

Lily2 Processor Instruction Set Reference

Version **300**
Nov 2012

Institute of Microelectronics
Tsinghua University
Beijing 100084



CONTENTS

1 INTRODUCTION	12
Overview	12
Features	2
2 ABI	3
寄存器堆	3
基础数据类型	4
结构体、数组和联合的对齐	5
比特域	5
寄存器使用约定	5
函数调用堆栈	6
函数传参	7
函数返回值	7
函数调用指令的硬件设计	7
异常处理机制	8
虚拟地址空间	8
内存分页	8
虚拟地址空间分配	9
处理器执行模式	9
Magic Number	9
重定位	9
3 INSTRUCTION CLASSIFY	2
Lily Processor 16-Bit Instructions	2

Lily Processor 32-Bit Instructions	3
Lily Processor Instruction Classify	4
4 INSTRUCTION FORMAT	7
16-Bit Instruction Format	7
32-Bit Instruction Format	9
Fetch Packet and Execution Packet	12
Parallel Operations	12
5 CONTROL REGISTER FILE	14
6 ARITHMETIC OPERATIONS	18
Instruction Summary	18
ADD/ADD.U	19
ADD.2	22
ADDK	24
SADD	25
SADD.S/U.2/US.2	27
SUB/SUB.U	29
SUB.2	32
SSUB	34
SSUB.S.2/U.2	36
MAX	38
MAX.S.2/U.2	39
MIN	40
MIN.S.2/U.2	41
AVG.S.2/U.2	42
ABS	43
ABS.2	45
NEG	47
NEG.2	49

SAT	51
SAT.L	52
SAT.S.2/.U.2	54
DQUANT.....	56
DQUANT.2.....	57
7 LOGICAL OPERATIONS	58
Instruction Summary	58
AND	59
NAND	61
ANDN	63
OR.....	64
NOR.....	66
NOR.....	68
XOR.....	69
NXOR	71
XORN	73
XOR3.....	74
NOT	75
8 SHIFT/ROTATE OPERATIONS.....	76
Instruction Summary	76
ASL	77
ASL.2.....	79
ASR	81
ASR.2	83
LSR.....	85
LSR.2.....	87
SASL	89
SASL.2	90
ASRR.....	91

ASRR.2	93
XASL	95
XASR.....	96
XASRR.2.....	97
ROL	98
ROL.2	99
SAVSL	100
SAVSR	101
9 MULTIPLY OPERATIONS.....	102
Instruction Summary	102
MUL.S/.U	103
MUL.S/.US/.U/.SU.H.....	104
MUL.S/.US/.U/.SU.L.....	106
MUL.S/.U/.SU/.US.HL.....	108
MUL.S/.U.HI	112
MUL.S/.U.LI	113
MUL.S/.U/.US.2	114
MULR.HI.....	116
MULR.LI.....	117
MULR.L.....	118
MULR.2.....	119
SMULL.H	121
SMULL.L	123
SMULL.HL	125
SMULL.LH	127
SMULL.2.....	128
MAC.L	129
MAC.H.....	131
SMACCL.L.....	132

SMACL.H.....	133
MSU.L	134
MSU.H.....	136
SMSUL.L.....	137
SMSUL.H.....	138
DTP.S/.SU.2	139
DTPN.S/.SU.2.....	140
DTPR.S/.SU.2.....	141
DTPNR.S/.SU.2.....	142
BTFDTP.2	144
BTFDTPR.2.....	145
10 LOAD/STORE OPERATIONS	147
Instruction Summary	147
LD.B.S/.U LD.H.S/.U LD.W LD.D.....	148
ST.B ST.H ST.W ST.D.....	153
PUSH/POP	158
11 COMPARE OPERATIONS	162
Instruction Summary	162
CMPE	163
CMPE.2.....	165
CMPG.S/.U	167
CMPG.S/.U.2.....	169
CMPL.S/.U.....	171
CMPL.S/.U.2	173
12 MOVE OPERATIONS	175
Instruction Summary	175
MOV.R.....	176
MOV.K	178

MOV.KH	179
MOV.RC	180
MOV.CR	181
13 BIT PERMUTATION OPERATIONS	182
Instruction Summary	182
SETB	183
ZEROB	184
PICKBB	185
PUTB	186
SELB	187
FSL	188
PACK.L.2	189
PACK.H.2	191
PACK.LH.2	193
PACK.HL.2	195
PACK.L.4	197
PACK.H.4	198
UNPK.2	199
UNPK.L.4	200
UNPK.H.4	202
SPACK.2	204
SPACKU.4	205
SWAP.2	207
SWAP.4	209
SHLMB1/2/3	211
EXP.L/.H	213
EXPB.L/.H	215
EXT.S/.U	217
SXT1	219

ZXT1.....	220
SXT2.....	221
ZXT2.....	222
SXT8.....	223
NORM	224
NORM.L	226
LMBD0/1	228
DEAL	230
SHFL	231
ZERO.....	232
14 FLOW CONTROL OPERATIONS.....	240
Instruction Summary	240
B/.Z/.NZ.....	241
BD/.Z/.NZ.....	243
CALL/.Z/.NZ.....	245
CALLD/.Z/.NZ.....	248
PUSH.PC.....	250
RET.....	252
RETD.....	254
IRET.....	256
IRETD.....	257
NOP	258
NOPN	259
PAUSE	260
IDLE.....	261
SLEEP.....	262
LPSETUP.....	263
15 VECTOR OPERATIONS.....	267
Instruction Summary	267

ADD.V	268
SADD.S/.U.V	270
AVG.S/.U.V	272
SUB.V	274
SSUB.S/.U.V	276
SSA.S/.U.V	278
ABS.V	280
NEG.V	282
MAX.S/.U.V	284
MIN.S/.U.V	286
ASL.V	288
ASR.V	290
LSR.V	292
MUL.S/.US/.U.V	294
CMPE.V	296
CMPG.S/.U.V	298
CMPL.S/.U.V	300
16 64-BIT REGFILE G OPERATIONS	302
Instruction Summary	302
ADDG	303
SUBG	304
PACKGB0	305
PACKGB1	306
PACKGB2	307
PACKGB3	308
PACKGB4	309
PACKGB5	310
PACKGH0	311
PACKGH1	312

PACKGH2.....	313
PACKGH3.....	314
PACKGW0.....	315
PACKGW1.....	316
PACKGW2.....	317
PACKGW3.....	318
UNPACKGB0	319
UNPACKGB1	320
UNPACKGB2	321
UNPACKGB3	322
UNPACKGB4	323
UNPACKGB5	324
UNPACKGH0.....	325
UNPACKGH1.....	326
UNPACKGH2.....	327
UNPACKGH3.....	328
UNPACKGW0.....	329
UNPACKGW1.....	330
UNPACKGW2.....	331
UNPACKGW3.....	332
SWAPGH0.....	333
SWAPGH1.....	334
SWAPGH2.....	335
ASLG	336
ASRG	337
LSRG.....	338
ROLG	339
17 CONFIGED OPERATIONS	340
Instruction Summary	340

MUXD0/1	341
MUXW0/1/2/3	343
MBOXI8LO32L128	344
MBOXI8LO32H128	346
MBOXI8HO32L128	348
MBOXI8HO32H128	350
MBOXI8O16H128	352
MBOXI8O16L128	354
MBOXI9O16A128	356
MBOXI9O32A128	358
MBOXI10O32A64	360
MBOXI10O16A64	362
MBOXI11O32A32	364
MBOXI11O16A32	366
MBOXI12O16A16	368
18 CRYPTON OPERATIONS	370
Instruction Summary	370
DESIP	371
DESIIP	372
DESPC1	373
DESPC2	374
DESSL	375
DESRND	376
XTIME	377
19 FLOATING POINT OPERATIONS	378
Instruction Summary	378
20 FLOATING POINT VECTOR OPERATIONS	412
Instruction Summary	412



1 INTRODUCTION

Overview

Lily 处理器是清华大学微电子所基于自行提出的 **Gingko** 处理器结构开发的一款数字信号处理器。**Gingko** 处理器结构是清华大学微电子所自主开发的可配置的数字信号处理器结构，其目标是方便、快速地开发出适合于每一类应用的目标处理器。**Gingko** 处理器结构采用超长指令字结构，尤其适合于多媒体信号处理等方面的应用。**Gingko** 中采用的 **RFCC-VLIW** 寄存器访问结构是一种全新的通用寄存器结构，可以有效解决超长指令字结构处理器中通用寄存器堆的面积、功耗、访问延时等问题。

Lily 处理器采用完全自主知识产权的指令集，该指令集采用了单指令多数据（SIMD）技术，具有矢量操作的特性，并且包含了大量面向有些特定应用的专用指令。

Lily 处理器是清华大学基于 **Gingko** 结构开发的系列处理器中的第二款处理器。其目的是提供一款具有高性能的面向媒体信号处理的处理器。图 1.1 为 Lily 的结构框图。

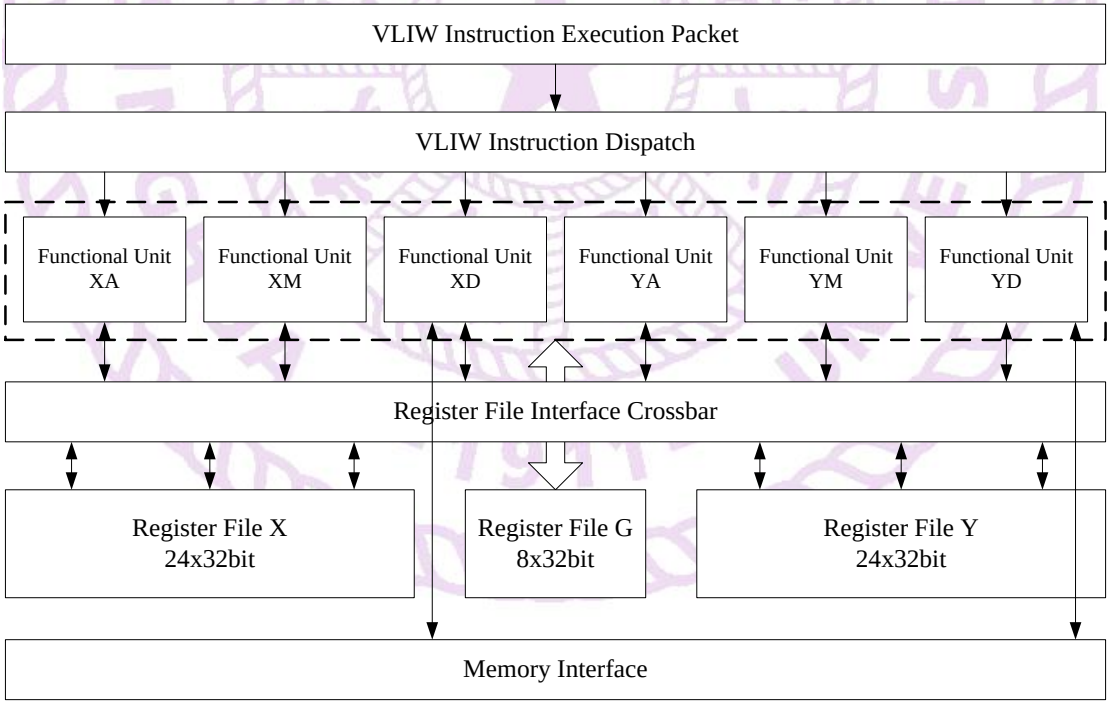


图 1.1 Lily 处理器总体架构

Features

Lily 作为一款高性能数字信号处理器，具有以下特点：

- 采用超长指令字结构，处理器中包含六个独立的功能单元，每个时钟周期可以并行执行六条指令。
- 处理器包含三个不同的功能单元类型，其中
 - .A:** 算术逻辑功能单元，执行加减法算术操作，移位指令和逻辑操作；
 - .M:** 乘法，浮点，浮点向量功能单元，执行乘法，乘累加，乘累减，点积，浮点计算，浮点向量等操作；
 - .D:** 数据传输功能单元，执行存储器访问指令以及比较指令。
- 片内集成了一个全局寄存器堆和两个本地寄存器堆，全局寄存器堆包含 16 个 128 位向量寄存器（包含了 8 个 32 位通用寄存器）；本地寄存器堆包含 24 个 32 位通用寄存器。
- 处理器支持 16 位和 32 位指令格式，16 位指令中每个功能单元具有 4bit 寄存器访问空间，可以访问 16 个寄存器，其中包括 4 个全局寄存器和 12 个本地寄存器；部分 16 位指令具有 5bit 寄存器访问空间，可以访问全部 32 个寄存器；32 位指令可以访问全局 32 个寄存器。
- 指令集中包含部分面向媒体信号处理的专用指令，可以显著提高处理器的性能。
- 所有 32 位指令均可以作为条件执行指令，可以有效提高应用程序的效率。
- 32 位指令集中包括矢量执行指令，通过 VSR 寄存器设置矢量长度 VL，以及矢量宽度 VW，可以进行矢量操作。系统默认矢量宽度为 8 位，矢量长度为 4，即一条矢量加法指令 ADD.V 可以并行执行 4 个 8 位数据。矢量宽度可以设置为 Byte 的整数倍，如 16 位，24 位，等。
- 16 位指令操作中部分专用寄存器用于标志特殊操作的寄存器，如 LBR (Load Base Address register)，SBR (Store Base Address register) 用于标志 Load/Store 的基地址；LOR (Load Offset Address register)，SOR (Store Offset Address register) 用于标志 Load/Store 的偏置；ACC 用于标志 MAC/MSU 指令的目标寄存器。
- 支持 10 个片外中断，包含 2 个不可屏蔽中断和 8 个可屏蔽中断。
- 支持 3 种休眠模式：Pause，Idle，以及 Sleep，分别不同的中断程序进行唤醒，其中 Sleep 模式只能用复位信号进行唤醒。

2 ABI

数据类型描述

寄存器堆

Lily2 处理器的寄存器堆，继承了 Lily 处理器的分簇结构，保留原有的 X、Y、G 三个寄存器堆的同时，对 G 寄存器堆的位宽和数量进行扩展，成为新的 V 寄存器堆，以适应向量计算的需求。Lily2 处理器的寄存器堆，以及各个功能单元的访问关系如下图：

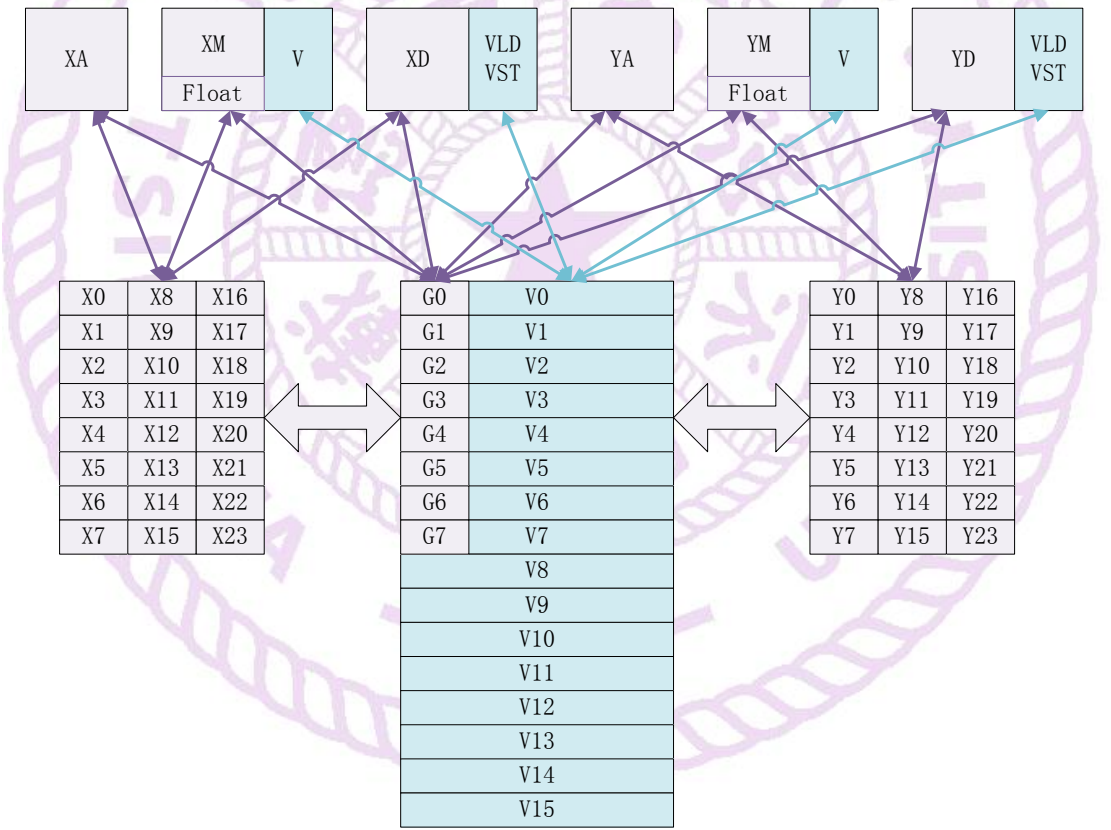


图 Lily2 处理器的功能单元和寄存器堆

寄存器堆的位宽：X、Y、G 寄存器堆均为 32bit，扩展后的向量寄存器为 128bit。

寄存器堆的数据交互：X、Y 寄存器堆不可直接传送数据，必须通过 G 寄存器堆中转。

G 寄存器堆与前 8 个向量寄存器的低 32 位重合。

功能单元访问规则：XA/XM/XD 功能单元的定点/浮点指令可以访问 X/G 寄存器堆，YA/YM/YD 功能单元的定点/浮点指令可以访问 Y/G 寄存器堆，M 单元的向量指令和 D 单元的向量存取指令只能访问扩展的 16 个向量寄存器。

基础数据类型

数据类别	C 语言数据类型	SIZEOF	字节对齐	Lily2 硬件实现
定点数据	char	1	1	signed byte
	signed char			
	unsigned char	1	1	unsigned byte
	short	2	2	signed halfword
	signed short			
	unsigned short	2	2	unsigned halfword
	int	4	4	signed singleword
	signed int			
	long			
指针	signed long			
	enum			
	unsigned int	4	4	unsigned singleword
	long long	8	8	signed doubleword (Register Pair)
浮点数据	signed long long			
	unsigned long long	8	8	unsigned doubleword (Register Pair)
指针	any-type *	4	4	unsigned singleword
	any-type (*) ()			
浮点数据	float	4	4	single precision float
	double	8	8	double precision float (Register Pair)

任何类型的 NULL 指针的值必须为 0。

浮点数据类型的格式符合 IEEE-754 标准。

在 Lily2 实现中，4 字节以下的数据类型将被存放在 32 位通用寄存器中，8 字节的数据类型将被存放在 32 位通用寄存器对中。

SIMD 类指令使用通用寄存器或寄存器对。向量类指令使用扩展后的 128 位寄存器。

结构体、数组和联合的对齐

结构体、数组和联合，其内部元素所要求的最大的字节对齐，将决定整体起始地址的对齐。

结构体和联合内部的数据，每个元素都放置于尽量靠前的对齐地址，元素与元素之间可能存在 padding。

比特域

与结构体和联合类似。

函数调用约定

寄存器使用约定

Lily2 处理器具有 X、Y、G、V 共 4 个寄存器堆，寄存器的使用约定如下：

寄存器	不受子函数调用影响	使用约定
X0/X1	否	PRI 寄存器，指令条件执行寄存器
X2~X9	是	Callee-Saved Register，发生函数调用时由被调用者负责保护，调用者不必考虑函数调用对这些寄存器的影响
X10/X11	否	Temporary Register，临时寄存器，发生函数调用时由调用者负责保护
X12~X17	否	函数传参寄存器 0~5
X18	否	RV—Return Value，函数返回值寄存器
X19	否	RVH—64 位函数返回值的高 32 位
X20	是	GP—Global Pointer，用于动态链接
X21	是	LR—Link Address Register，用于记录 CALL 指令的返回地址
X22	是	FP—Frame Pointer，函数栈帧指针
X23	是	SP—Stack Pointer，函数堆栈指针
Y0~Y23	否	Temporary Register，临时寄存器，发生函数调用时由调用者负责保护，Y0/Y1 为 PRI 寄存器

G0~G7	否	Temporary Register, 临时寄存器, 发生函数调用时由调用者负责保护, G0/G1/G2 为 PRI 寄存器
V0~V15	否	Temporary Register, 临时寄存器, 发生函数调用时由调用者负责保护

不受子函数调用影响的寄存器, 表示当前函数在使用这些寄存器的过程中, 调用其它子函数并返回之后, 这些寄存器的值仍保持子函数调用前的状态, 可以直接继续使用。

SP、FP: 在函数调用过程中, 函数调用堆栈将向低地址方向增长。SP 指针指向当前的栈顶, 低于 SP 的地址的栈空间是无效数据。SP 指针保持 8 字节对齐。FP 指针指向前一个栈帧, 以 FP 为基址可以获取函数参数等数据。

函数参数寄存器: 编译器可以选择是否启用寄存器传参规则。如果启动, 函数参数将保存在 X12~X17 中, 更多的参数将被保存在堆栈中; 如果不启动, 则全部参数保存在堆栈中, 使用 LD/ST 的方式来得到。

LR 寄存器: 即 X21 寄存器, 用于保存 call 指令的返回地址, 子函数返回时可以根据该寄存器的值来正确返回。嵌套函数的 LR 值将被保存到函数调用堆栈中, LR 寄存器中保存的永远是当前函数的返回地址。

RV 和 RVH: 函数返回值寄存器。函数返回的位宽小于 4 字节的数据, 将被存放于 RV 寄存器中; 函数返回 64bit 的值, 高 32 位将被保存于 RVH 寄存器; 函数返回的结构体和联合将被保存在堆栈中, RV 中保存指向该结构体或联合的指针; 如果函数无返回值, RV 和 RVH 中的值将处于未定义状态。

在处理器设计中, 可以使用一些额外的寄存器堆, 来实现进程上下文的快速切换, 供操作系统来使用。该部分内容待补充。

函数调用堆栈

每当一个函数调用它的子函数, 函数调用堆栈将开辟一个新的栈帧。

在子函数入口, 编译器将根据子函数中局部变量占用的空间, 将 SP 向低地址跳跃相应的距离, 并将 FP 指向恰当的位置以获取上层函数的传参, 这些操作称为 **prologue**。函数返回之前同样要进行一些 SP 和 FP 的恢复工作, 称为 **epilogue**。

如果一个函数内部没有其它的函数调用, 该函数被称为 **Leaf Function**。编译器对于这类函数有特殊的优化, 如可以直接在子函数中使用 SP 进行计算, 节省的 **prologue** 和 **epilogue**。

函数调用堆栈中一个栈帧中的地址对应关系:

地址	说明
FP+0, FP+4, ……	上层函数传递过来的参数
FP-4, FP-8, ……	当前函数的局部变量
SP+4	保存上层函数的 FP 的值
SP+0	保存上层函数的返回地址
SP-4, ……	可直接被 Leaf Function 使用，被 Exception Handler 使用

函数传参

32-bit 以下的函数参数，将存放在 X12~X17 中；

结构体和联合，将在堆栈中整体拷贝并传递对应的指针；

64-bit 的函数参数，也使用 X12~X17 来传递，但并不保证对齐，比如可能存放在 X12 和 X13 中，也可能存放在 X13 和 X14 中；

传参寄存器中放不下的函数参数，将按顺序存放在堆栈中，子函数使用 FP 为基址去获得实参。

函数返回值

32-bit 以下的函数返回值，将存放在 RV 寄存器中；

对于结构体和联合，RV 中返回指向该结构体和联合的指针；

64-bit 的函数返回值，低 32 位存放于 RV，高 32 位存放于 RVH；

void 类型的函数返回值，RV 和 RVH 将处理未定义状态。

函数调用指令的硬件设计

Lily2 处理器具有两种执行模式，risc 和 vliw，两种模式对于函数返回地址的计算有细微的差别。

在 risc 模式下，32 位 call 指令的返回地址为 PC+4，16 位 call 指令的返回地址为 PC+2；在 vliw 模式下，call 指令的返回地址为下一个并行指令组的起始地址。

在执行 call、calld、callr、calldr 等指令时，处理器应该能够将正确的返回地址存放到 LR 寄存器（X21）中。

操作系统接口

异常处理机制

异常发生及处理流程:

- 保存发生异常时的 PC、寄存器等各种处理器状态;
- 处理器切换到特权模式;
- 根据中断向量表, 跳到相应的位置执行相应的处理程序;

异常类型	中断向量入口	SIGNAL	示例
Reset	0x100	None	Reset
Bus Error	0x200	SIGBUS	不存在的物理地址
Data Page Fault	0x300	SIGSEGV	不允许的数据内存页访问
Instruction Page Fault	0x400	SIGSEGV	不允许的程序内存页访问
Tick Timer Interrupt	0x500	None	进程调度
Alignment	0x600	SEGBUS	非对齐的 LD/ST
Illegal Instruction	0x700	SEGILL	未定义的指令
External Interrupt	0x800	None	外设
D-TLB Miss	0x900	None	DTLB 更新
I-TLB Miss	0xA00	None	ITLB 更新
Range	0xB00	SIGSEGV	计算溢出
System Call	0xC00	None	syscall 指令调用
Floating Point	0xD00	None	浮点异常
Trap	0xE00	SIGTRAP	trap 指令调用和调试单元请求

虚拟地址空间

与 MMU 实现相关, 待补充。

内存分页

基本配置是 8KB 每页。待补充。

虚拟地址空间分配

每个进程占用一个独立的虚拟地址空间。

在虚拟地址空间中，需要保存进程的指令（.text 段），数据（.data、.bss、.rodata 段等），函数调用堆栈，堆，等等。它们的地址需要有一个合理的约定，链接器、操作系统需要对其进行支持。

函数调用堆栈，Stack，向低地址方向增长；

堆，Heap，使用 malloc 分配的空间，向高地址增长。

处理器执行模式

Lily2 处理器支持两种执行模式：用户模式（user）和特权模式（supervisor）。

用户程序运行在用户模式下，操作系统运行在特权模式下。

用户程序通过使用 syscall 指令来切换到操作系统的特权模式中。

Syscall 系统调用与普通的函数调用的传参、返回值等约定相同，唯一是需要额外指定一个 Syscall 的 id 号。

ELF 文件格式

Magic Number

在 ELF 文件的 Header 中，e_machine 的值，即处理器的 Magic Number，为 Lily2 处理器设定为 0xcaff。

重定位

Lily2 处理器的重定位除了标准的 NONE、32-bit、64-bit 等，其它由跳转类指令与 mov.kl/kh 指令来确定。

重定位类型	Size (bit)	说明
NONE	0	无计算操作
32-bit、64-bit	32、64	这类重定位用于数据段中的地址重定位, DWARF 调试信息支持等
mov.kl/kh	lo16、hi16	用于获取数据段变量、程序标签的绝对地址
b、call	20	计算跳转的相对偏移地址, VLIW 模式下并行指令组中不同位置的跳转指令具有不同的重定位



3 INSTRUCTION CLASSIFY

Lily Processor 16-Bit Instructions

----88 instructions

.A type unit		.M type unit	.D type unit	
ADD	SAT	MUL.S.H	LD.B.S	B
SUB	SAT.L	MUL.US.H	LD.B.U	B.Z
ADD.2	SAT.S.2	MUL.U.H	LD.H.S	B.NZ
SUB.2	SAT.U.2	MUL.SU.H	LD.H.U	ZERO
ASL	AND	MUL.S.L	LD.W	BD
ASL.2	ANDN	MUL.US.L	LD.D	BD.Z
LSR	OR	MUL.U.L	ST.B	BD.NZ
LSR.2	ORN	MUL.SU.L	ST.H	CALL
ASR	XOR	MUL.S.HL	ST.W	CALL.Z
ASR.2	XORN	MUL.U.HL	ST.D	CALL.NZ
ASRR	NOT	MUL.SU.HL	MOV.R	RET
ASRR.2		MUL.US.HL	PUSH	RETD
ABS		MUL.S.LH	POP	PAUSE
ABS.2		MUL.U.LH	SWAP.2	IDLE
NEG		MUL.SU.LH	SWAP.4	SLEEP
NEG.2		MUL.US.LH	EXP.L	NOP
		SMULL.H	EXP.H	NOPN
		SMULL.L	EXPB.L	ING
		SMULL.HL	EXPB.H	
		SMULL.LH	ADD	
		MUL.S.2	SUB	
		SMULL.2		
		ADD		
		SUB		

Lily Processor 32-Bit Instructions

----186 instructions

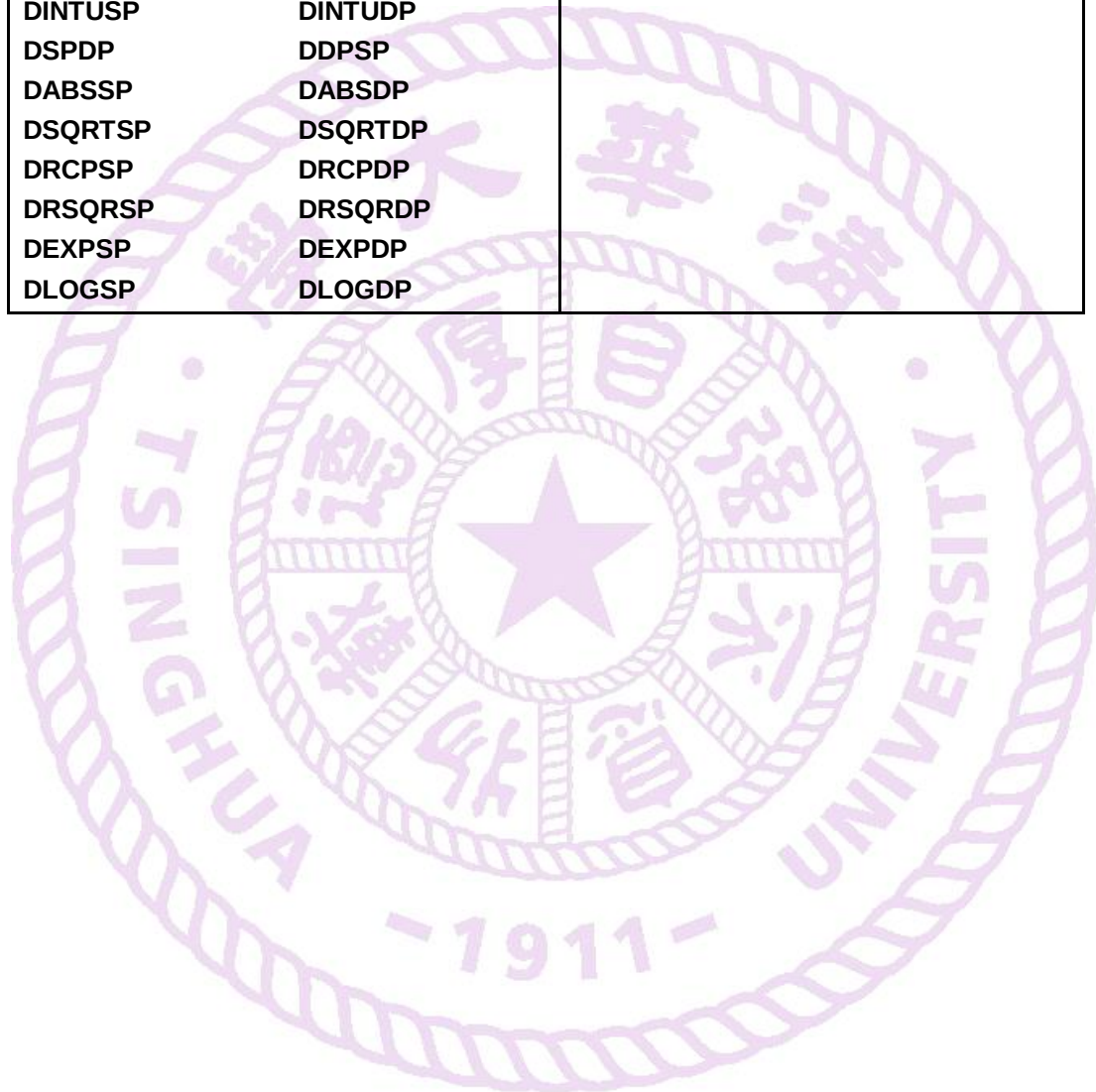
.A type unit		.M type unit		.D type unit	
ADD	MAX.S.2	MUL.S	MAC.H	LD.B.S	CMPE
ADD.U	MAX.U.2	MUL.U	SMACL.H	LD.B.U	CMPG.S
ADD.2	MIN.S.2	MUL.S.H	MSU.H	LD.H.S	CMPG.U
SADD	MIN.U.2	MUL.US.H	SMSUL.H	LD.H.U	CMPL.S
SADD.S.2	MAX.S.V	MUL.U.H	BTFDTP.2	LD.W	CMPL.U
SADD.U.2	MAX.U.V	MUL.S.L	BTFDTPR.2	LD.D	CMPE.2
SADD.US.2	MIN.S.V	MUL.US.L	ADD	ST.B	CMPG.S.2
AVG.S.2	MIN.U.V	MUL.U.L	SUB	ST.H	CMPG.U.2
AVG.U.2	ROL	MUL.S.HL	ADD.2	ST.W	CMPL.S.2
ADD.V	ROL.2	MUL.U.HL	SUB.2	ST.D	CMPL.U.2
SADD.S.V	AND	MUL.SU.HL	AND	PUSH	CMPE.V
SADD.U.V	ANDN	MUL.US.HL	ANDN	POP	CMPG.S.V
AVG.U.V	OR	MUL.S.HI	OR	PACK.L.2	CMPG.U.V
AVG.S.V	ORN	MUL.U.HI	ORN	PACK.H.2	CMPL.S.V
SUB.V	XOR	MUL.S.LI	XOR	PACK.LH.2	CMPL.U.V
SUB	XORN	MUL.U.LI	XORN	PACK.HL.2	B
SUB.U	ADD.SL1	MULR.HI	PACK.L.2	PACK.L.4	BD
SUB.2	ADD.SL2	MULR.LI	PACK.H.2	PACK.H.4	CALL
SSUB	ADD.SL3	MULR.L	PACK.LH.2	SPACK.2	LPSETUP
SSUB.S.2	SLADD1	SMULL.H	PACK.HL.2	SPACKU.4	ADD
SSUB.U.2	SLADD2	SMULL.L	PACK.L.4	SHLMB1	SUB
SSUB.S.V	SLADD3	SMULL.HL	PACK.H.4	SHLMB2	ADD.2
SSUB.U.V	ABS	MUL.S.2	SPACK.2	SHLMB3	SUB.2
SAA.S.V	ABS.2	MUL.US.2	SPACK.4	NORM	AND
SAA.U.V	ABS.V	MUL.U.2	SHLMB1	LMBD	ANDN
XASL	SAT	MULR.2	SHLMB2	DEAL	OR
XLSR	SATL	SMULL.2	SHLMB3	SHFL	ORN
ADDK	SAT.S.2	DTP.S.2	SWAP2	EXP.L	XOR
ASL	SAT.U.2	DTP.SU.2	SWAP4	EXP.H	XORN
ASR	NEG	DTPN.S.2	EXP.L	EXPB.L	MOV.RC
LSR	NEG.2	DTPN.SU.2	EXP.H	EXPB.H	MOV.CR
ASL.2	NEG.V	DTPR.S.2	EXPB.L	UNPK.L.4	MOV.R
ASR.2	NOT	DTPR.SU.2	EXPB.H	UNPK.H.4	
LSR.2	MOV.R	DTPNR.S.2	UNPK.2	SWAP.2	
SASL	EXT.S	DTPNR.SU.2	UNPK.L.4	SWAP.4	
SASL.2	EXT.U	MAC.L	UNPK.H.4		
ASRR	PACK.L.2	SMACL.L	MOV.R		
ASRR.2	PACK.H.2	MSU.L	MOV.K		
ASL.V	PACK.LH.2	SMSUL.L	MOV.KH		
ASR.V	PACK.HL.2		MUL.S.V		
LSR.V	UNPK.L.4		MUL.US.V		
SLAC.L	UNPK.H.4		MUL.U.V		
SLSU.L	SAVSL				
DQUANT	SAVSR				
DQUANT.2					

Lily Processor Instruction Classify

Instruction Name			
Arithmetic Operations		MAC Operations	
ADD	AVG.S.2	MUL.S	DTP.S.2
ADD.U	AVG.U.2	MUL.U	DTP.SU.2
ADD.2	NEG	MUL.S.H	DTPN.S.2
SADD	NEG.2	MUL.US.H	DTPN.SU.2
SADD.S.2	ABS	MUL.U.H	DTPR.S.2
SADD.U.2	ABS.2	MUL.S.L	DTPR.SU.2
SADD.SU.2	SAT	MUL.US.L	DTPNR.S.2
ADDK	SATL	MUL.U.L	DTPNR.SU.2
SUB	SAT.S.2	MUL.S.HL	DTP.S.4
SUB.U	SAT.U.2	MUL.U.HL	DTP.U.4
SUB.2	ADDSL1	MUL.SU.HL	DTP.SU.4
SSUB	ADDSL2	MUL.US.HL	MAC.S.L
SSUB.S.2	ADDSL3	MUL.S.HI	SMACL.S.L
SSUB.U.2	SLADD1	MUL.U.HI	MSU.S.L
MAX.S.2	SLADD2	MUL.S.LI	SMSUL.S.L
MAX.U.2	SLADD3	MUL.U.LI	MAC.S.H
MIN.S.2		MULR.HI	SMACL.S.H
MIN.U.2		MULR.LI	MSU.S.H
		SMULL.H	SMSUL.S.H
		SMULL.L	BTFDTP.2
		SMULL.HL	BTFDTPR.2
		MUL.S.2	
		MUL.US.2	
		MUL.U.2	
		MULR.2	
		SMULL.2	
Logical Operations		Move Operations	
AND		MOV.R	MOV.KL
ANDN		MOV.L	MOV.KH
OR		MOV.H	MOV.RC
ORN			
XOR		MOV.LH	MOV.CR
XORN			
NOT		MOV.HL	
Shift/Rotate Operations		Bit permutation Operations	
ASL	ASRR		
ASR	ASRR.2		
LSR	XASL		
ASL.2	XLSR		

ASR.2	ROL	NORM	PACK.L.2
LSR.2	ROL.2	LMBD	PACK.H.2
SASL		ONES	PACK.LH.2
SASL.2		DEAL	PACK.L.4
Load/Store Operations		SHFL	PACK.H.4
LD.B.S	ST.B	SHLMB1	UNPK.L.4
LD.B.U	ST.H	SHLMB2	UNPK.H.4
LD.H.S	ST.W	SHLMB3	SPACK.2
LD.H.U	ST.D	EXP.L	SPACKU.4
LD.W	FIFO.W	EXP.H	ZERO
LD.D	FIFO.R	EXP.BL	PACKG
POP	PUSH	EXP.BH	PACKGL2
LD.Q	ST.Q	SWAP2	PACKGH2
Compare Operations		UNPACKGH	UNPACKG
CMPE	CMPE.2	UNPACKGL	
CMPG.S	CMPG.S.2	Control Operations	
CMPG.U	CMPG.U.2	B	B.Z
CMPL.S	CMPL.S.2	BD	B.NZ
CMPL.U	CMPL.U.2	CALL	BD.Z
Vector Operations		RET	BD.NZ
ADD.V	MAX.S.V	RETN	NOP
SADD.S.V	MAX.U.V	LPSETUP	PAUSE
SADD.U.V	MIN.S.V		IDLE
AVG.U.V	MIN.U.V		SLEEP
AVG.S.V	MUL.S.V	Floating Point Operations	
SUB.V	MUL.US.V	MINSP	MINDP
SSUB.S.V	MUL.U.V	MAXSP	MAXDP
SSUB.U.V	CMPE.V	CMPLTSP	CMPLTDP
SAA.S.V	CMPG.S.V	CMPGTSP	CMPGTDP
SAA.U.V	CMPG.U.V	CMPEQSP	CMPEQDP
ASL.V	CMPL.S.V	ADDSP	ADDDP
ASR.V	CMPL.U.V	SUBSP	SUBDP
LSR.V		MPYSP	MPYDP
Floating Point Vector Operations		DIVSP	DIVDP
DMINSP	DMINDP	MPYADDSP	MPYADDDP
DMAXSP	DMAXDP	NMPYADDSP	NMPYADDDP
DCMPLTSP	DCMPLTDP	MPYSUBSP	MPYSUBDP
DCMPGTSP	DCMPGTDP	NMPYSUBSP	NMPYSUBDP
DCMPEQSP	DCMPEQDP	CMPYSP	CMPYDP
DADDSP	DADDDP	SPINT	DPINT
DSUBSP	DSUBDP	SPINTU	DPINTU
DMPYSP	DMPYDP	ROUNDSP	ROUNDDP
		INTSP	INTDP

DDIVSP	DDIVDP	INTUSP	INTUDP
DMPYADDSP	DMPYADDDP	SPDP	DPSP
DNMPYADDSP	DNMPYADDDP	ABSSP	ABSDP
DMPYSUBSP	DMPYSUBDP	SQRTSP	SQRTDP
DNMPYSUBSP	DNMPYSUBDP	RCPSp	RCPDP
DSPINT	DDPINT	RSQRSP	RSQRDP
DSPINTU	DDPINTU	EXPSP	EXPDP
DROUNDSP	DROUND DP	LOGSP	LOGDP
DINTSP	DINTDP		
DINTUSP	DINTUDP		
DSPDP	DDPSP		
DABSSP	DABSDP		
DSQRTSP	DSQRTDP		
DRCPSP	DRCPDP		
DRSQRSP	DRSQRDP		
DEXPSP	DEXPDP		
DLOGSP	DLOGDP		



4 INSTRUCTION FORMAT

16-Bit Instruction Format

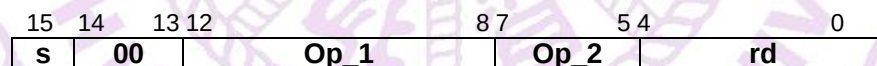
- **AL ins:** Arithmetic and Logical Instructions
 - **.A**
 - ADD, MAX, ASL, ... , etc
- **MA ins:** Multiply and Accumulate Instructions
 - **.M**
 - MUL,..., etc
- **DT ins:** Data transfer and Memory Load/Store Instructions
 - **.D**
 - LD, ST, MOV, ... , etc

AL.F ins

AL.F1



AL.F2



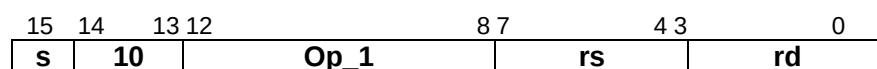
MA.F ins

MA.F1

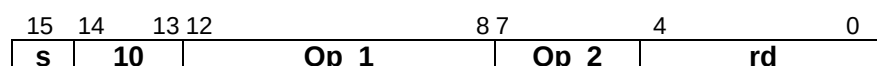


DT.F ins

DT.F1



DT.F2



DT.F3

15	14	13	12	10	9	5	4	0
s	10	Op_1	rs	rd				

DT.F4

15	14	13	12	8	7	5	4	0
s	10	Op_1	Op_2	Op_3				

DT.F5

15	14	13	12	8	7	5	4	3	2	0
s	10	Op_1	Op_2	Op_3	z	creg				

DT.F6

15	14	13	12	9	8	6	5	4	3	0
s	10	Op_1	mode	A	B	rd				

s: register file select ---“0” X register file; “1” Y register file
c: conditional execute --- “0” unconditional; “1” conditional
z: z-test for zero or nonzero when conditional instruction --- “0” unzero; “1” zero
A:选择存储器访问基址寄存器 LBR, SBR
— “0” LBR:X8/Y8 SBR: X9/Y9
— “1” LBR:X10/Y10 SBR: X11/Y11
B:选择存储器访问偏移量寄存器 LOR, SOR
— “0” LOR:X4/Y4 SOR: X5/Y5
— “1” LOR:X6/Y6 SOR: X7/Y7
Mode: 存储器访问模式选择

Mnemonic	Meaning
.S	有符号数操作
.U	无符号数操作
.K	立即数操作
.I	32 位整数操作
.L	低 16 位数操作
.H	高 16 位数操作
.B	字节操作
.H	半字操作
.W	字操作
.D	双字操作
.X	40 位 long 数操作
.2	2 个 16 位 subword 操作
.V	Byte 矢量操作
sint/uint	有符号/无符号 32 位整数
hw2s/hw2u	有符号/无符号 Packed-16bit
4Bytes/4Byteu	有符号/无符号 Packed-8bit
slong/ulong	有符号/无符号 40 位 long 型
sllong/ullong	有符号/无符号 64 位 llong 型
vecs/vecu	有符号/无符号矢量单元

Mnemonic	Meaning	Register
LBR	Load Base Address	X8/Y8 X10/Y10

SBR	Store Base Address	X9/Y9 X11/Y11
LOR	Load Offset Address Register	X4/Y4 X5/Y5
SOR	Store Offset Address Register	X6/Y6 X7/Y7

语法形式	Mode	基址修改方式	偏移地址对齐
*LBR	000	保持	是
*+LBR[LOR]	010	保持	是
*-LBR[LOR]	011	保持	是
*LBR++[LOR]	100	后加	是
*LBR--[LOR]	101	后减	是
*++LBR [LOR]	110	先加	是
*--LBR[LOR]	111	先减	是

语法形式	Mode	基址修改方式	偏移地址对齐
*SBR	000	保持	是
*+SBR[SOR]	010	保持	是
*-SBR[SOR]	011	保持	是
*SBR++[SOR]	100	后加	是
*SBR--[SOR]	101	后减	是
*++SBR[SOR]	110	先加	是
*--SBR[SOR]	111	先减	是

32-Bit Instruction Format

- **AL ins:** Arithmetic and Logical Instructions
 - .A
 - ADD, MAX, SHL, ... , etc
- **MA ins:** Multiply and Accumulate Instructions
 - .M
 - MUL, MAC, MSU, DTP, ..., etc
- **DT ins:** Data transfer and Memory Load/Store Instructions
 - .D
 - LD, ST, MOV, ... , etc

AL.D ins

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

31	30	29	28	27	26	25	24	8				4	3	2	0
s	11	00	Op	scst16				rd				z	creg		

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2/const5		rs1		rd		z	creg		

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				Op_2		rs1			rd			z	creg

31	30	29	28	27	26	24	23	21	20	5	4	0
s	11	01	Op	Op	3	Scst16					rd	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2/const5		rs1		rd		z	creg		

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				Op_2				rs1		rd		z	creg

31	30	29	28	27	26	23	22	19	18	14	13	9	8	4	3	2	0
s	11	11	Op			mode		ffset/cst5			baseR		rd		z	creg	

31	30	29	28	27	26	24	23					4	3	2	0
s	11	11	Op			cst20							z	creg	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				Op_2		rs		rd		z	creg		

31	30	29	28	27	26	19	18	3	2	0
s	11	10	Op	Op	1	cst16			LPRop	

- **N-LPSETUP select**

LPRop	LPSETUP select	
000	LPSETUP0a	将 Loop0_Begin: 地址的低 16 位赋值给 LPT0 的低 16 位
001	LPSETUP0b	将 Loop0_Begin: 地址的高 16 位赋值给 LPT0 的高 16 位
010	LPSETUP0c	将 Loop0_End: 地址加上 LPT0 的结果赋值给 LPB0
011	LPSETUP0d	将 Loop0_Num -1 赋值给 LPC0
100	LPSETUP1a	将 Loop1_Begin: 地址的低 16 位赋值给 LPT1 的低 16 位
101	LPSETUP1b	将 Loop1_Begin: 地址的高 16 位赋值给 LPT1 的高 16 位
110	LPSETUP1c	将 Loop1_End: 地址加上 LPT1 的结果赋值给 LPB1
111	LPSETUP1d	将 Loop1_Num -1 赋值给 LPC1

- **Creg-conditional register select**

Creg	Conditional register
000	Non conditional
001	G0
010	G1
011	G2
100	X0
101	X1
110	Y0
111	Y1

- **s: register file select ---“0” X register file; “1” Y register file**

- **z: Z-test for zero or nonzero when conditional instruction**

如果 z=1, 当条件寄存器为 0 时, 执行该指令; 反之, 如果 z=0, 则条件寄存器非零时, 执行该指令, 无条件执行时, z 位设为 0。

- **rd—address of destination register**

目标寄存器地址, 共 5bit, 地址 00h—17h 分别代表本地寄存器 X0—X23 或 Y0—Y23。地址 18h—1fh 分别代表全局寄存器 G0—G7。如果目标寄存器为寄存器对, 则 RD 中保存寄存器堆中地址较小的寄存器地址 (为偶数)。

- **rs1,rs2,rs3—address of source register**

源寄存器地址, 各 5bit, 地址 00h—17h 分别代表本地寄存器 X0—X23 或 Y0—Y23。地址 18h—1fh 分别代表全局寄存器 G0—G7。如果源寄存器为寄存器对, rs1 中保存寄存器堆中地址较小的寄存器地址 (为偶数)。Rs2 不支持寄存器对。

- **Load/Store Mode**

Syntax	Mode	基址修改方式	偏移地址对齐
*BaseR	0000	保持	是
*+BaseR[Offset]	0010	保持	是
*-BaseR[Offset]	0011	保持	是
*BaseR++[Offset]	0100	后加	是
*BaseR--[Offset]	0101	后减	是
*++BaseR[Offset]	0110	先加	是
*--BaseR [Offset]	0111	先减	是

*+BaseR(ucst5)	1000	保持	否
*-BaseR(ucst5)	1001	保持	否
*+BaseR[ucst5]	1010	保持	是
*-BaseR[ucst5]	1011	保持	是
*BaseR++[ucst5]	1100	后加	是
*BaseR--[ucst5]	1101	后减	是
*++BaseR[ucst5]	1110	先加	是
*--BaseR[ucst5]	1111	先减	是

Fetch Packet and Execution Packet

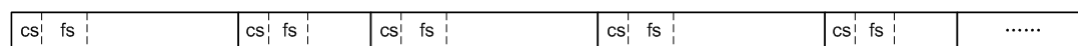
Lily 处理器每周期可以从指令缓存或指令 RAM 中读取 256bit(8 x 32bit)指令，这 256bit 指令被称为一个取指包(Fetch Packet)。取指包包含混合的 16 位指令和 32 位指令，可以由处理器分多个周期发送完成。

处理器每周期最多可以并行发送 6 条指令，这些被并行发送的指令组成一个执行包(Execution Packet)，在执行包的指令被并行执行。

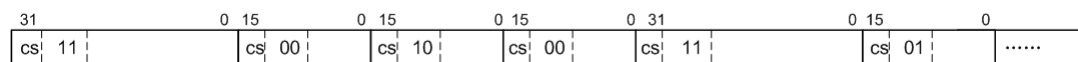
Parallel Operations

Lily 处理器根据一条指令最左端的第 2 位和第 3 位判断指令格式是 16 位指令还是 32 位指令，如果指令最左端的第 2 位和第 3 位为 2'b11，则相应指令为 32 位指令，否则为 16 位指令。

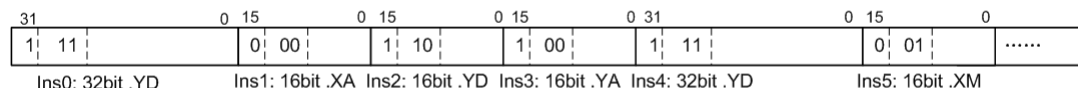
Lily 处理器没有独立的并行位用于标识指令的并行执行，而是通过指令中的簇选择位来完成指令并行的判断。在一个执行包内的指令按照一定的顺序排列 XA, XM, XD, YA, YM, YD, 即指令的高三位（一位寄存器堆选择位，两位功能单元选择位）按照升序排列；如果并行的指令按照一定的规定的顺序完成，则处理器认为可以将这组指令并行执行。



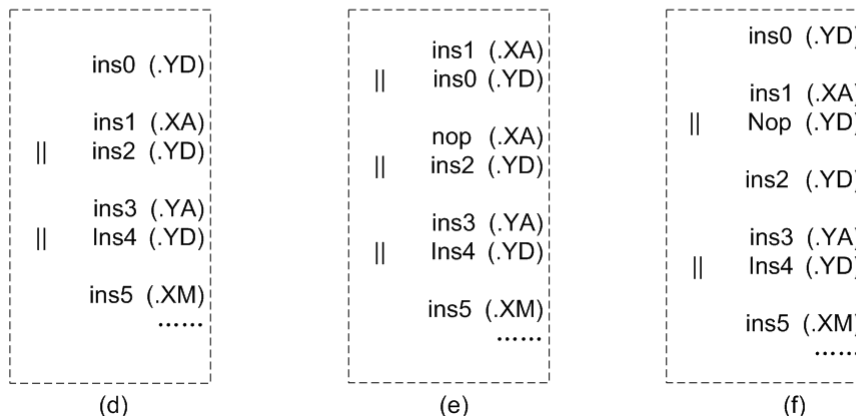
(a) Fetch Packet



(b) Instruction Format Judge



(c) Parallel Operation Judge



如上图示例,ins0 中最高三位为 111,根据最高第二位和第三位的"11",则判断 ins0 为 32 位指令。Ins1 中最高三位为 000,根据最高第二位和第三位的"00",则判断 ins1 为 16 位指令。同理可以得出其他指令的格式。

根据各条指令的寄存器堆选择位以及功能单元位,可以得出指令的并行关系,如 ins0 由 YD 执行,ins1 由 XA 执行。根据 Lily 处理器执行并行执行的原则,可以得出处理器 cycle0 只执行 ins0, cycle1 并行执行 ins1,ins2; cycle2 并行执行 ins3,ins4; cycle3 执行 ins5。

程序员写汇编程序的时候仍然采用并行执行符"||"来标识指令并行。如果并行符||两边的指令都由同一个簇完成,如 ins0 与 ins1,则汇编器会自动改变 ins0 和 ins1 的顺序,并在 ins2 之前增加一条 Nop 指令来实现 ins0 和 ins2 的分两个周期执行。Ins2 与 ins3 由于功能单元未按照升序排列,则 ins2 与 ins3 分两个周期进行,如(e)所示。如果两条升序排列的指令之间并没有并行执行符,如 ins1 与 ins2,则汇编器会自动的增加两条条 NOP 指令来串化执行 ins1 与 ins2,如 (f) 所示。

5 CONTROL REGISTER FILE

.D 功能单元可以访问控制寄存器堆。控制寄存器堆通过 **MOV.RC**, **MOV.CR** 指令与通用寄存器堆之间传输数据。其中，一些特殊的控制寄存器位可以被独立访问。例如，中断标志寄存器 IFR 中的中断标志位可以被相应的中断信号触发。

Addr	Name	Meaning	Description
00000	AMR	Addressing Mode Register	定义寻址类型，可以采用线性寻址或循环寻址，同时包含了循环寻址采用的 Block 的大小
00001	CSR	Control Status Register	包括中断使能，cache 控制，等状态位
01000	LPC0	LOOP Counter 0	LOOP 计数器
01010	LPB0	LOOP Bottom 0	LOOP 底寄存器
01011	LPT0	LOOP TOP 0	LOOP 头寄存器
01100	LPC1	LOOP Counter 1	LOOP 计数器
01110	LPB1	LOOP Bottom 1	LOOP 底寄存器
01111	LPT1	LOOP TOP 1	LOOP 头寄存器
00100	SP	STACK Point	STACK 指针
00110	SKB	STACK Bottom	栈底寄存器
00111	SKT	STACK Top	栈顶寄存器
10000	IGE	Interrupt Global Enable	全局中断使能
10001	IER	Interrupt Enable Register	中断使能寄存器
10010	IFR	Interrupt Flag Register	中断标志寄存器
10100	RETPC	Interrupt and Subroutine Return PC	中断返回寄存器
10101	VSR	Vector Status Register	设置矢量长度，矢量宽度，以及数据通道宽度
11000	PC	Program counter	程序指针
11001	PCE1	Program counter, E1 phase	E1 流水级

AMR

AMR: 存储器访问模式寄存器，处理器访问存储器时，如果基址寄存器是 X8, X9, X10, X11, Y8, Y9, Y10, Y11，则访问模式由 AMR 确定，AMR 的结构如图所示

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
MX8	MX9		MX10		MX11		MY8		MY9		MY10		MY11			
BLK1								BLK2								
15								8 7								0

MX*,MY*分别代表 X8, X9, X10, X11, Y8, Y9, Y10, Y11 的访问模式，访问模式如表所示

00	线性寻址
01	循环寻址，寻址空间大小由 blk1 决定
10	循环寻址，寻址空间大小由 blk2 决定
11	保留

CSR

CSR: 控制状态寄存器，包括控制和状态位。

31																	17		16	
Reserved																	PWRD			
SUF		SOF		Reserved										Z[6:0]						

Z[5:0]: 条件寄存器，每一位标志对应的条件寄存器的值是否为零

SOF: Stack Overflow flag 用于标志栈是否发生上溢。

SUF: Stack Underflow flag 用于标志栈是否发生下溢。

PWRD: Control power-down modes 功耗管理标志。

当处理器处于 idle 状态或 Sleep 状态时，该标志位置为“1”，并通过相应管脚输出。

Z[5:0]: Conditional execution register 条件标志

当相应的条件标志寄存器为 0 的时候，将相应的条件标志置为 0，否则置为 1

Z	Conditional register
Z0	G0
Z1	G1
Z2	X0
Z3	X1
Z4	Y0
Z5	Y1

R: Reserved 保留位

LPMR

LPMR 是用于控制 LOOP 的 LOOP 寄存器，其中包括 LPC 用于循环计数，处理器中最多可以同时开启 2 组循环，循环设置通过 XD,YD 功能单元的 **LPSETUP0, LPSETUP1, LPSETUP2, LPSETUP3** 指令进行设置循环参数，LOOP_BEGIN 指令以及 LOOP_END 伪指令来控制循环，当 LOOP_BEGIN 的时候，汇编器记录相应指令地址，并将指令地址用于 **LPSETUP0, LPSETUP1**，设置 Loop 起始地址 LPT； LOOP_END 的时候汇编器记录相应的循环结束指令地址，并将结束地址和循环起始地址的差用于 **LPSETUP2** 指令，计算得到 Loop 结束地址 LPB，系统要求循环起始地址和结束地址之间的差不能超过 $2^{32}-1$ 。**LPSETUP3** 指令设置 LPC，每个循环的最大循环次数是 65536 次。循环启动时，在处理器的取指阶段，判断取指地址和 LPT，如果 LPB 落在取指范围内，则取在 LPB 地址之前的指令和 LPT 地址的指令。并将 LPC 减 1，当 LPC 为 0 的时候跳出循环，取 LPB 后的指令。

15	0
LPC0	
31	0
LPT0	
31	0
LPB0	
15	0
LPC1	
31	0
LPT1	
31	0
LPB1	

举例说明:

```
LPSETUP0 // 将 LOOP BEGIN 的高 16 位赋给 LPT 的高 16 位
```

```
LPSETUP1      // 将 LOOP BEGIN 的低 16 位赋给 LPT 的低 16 位
```

```
LPSETUP2          // 将 LOOP END 加上 LPR.ADR.S 赋给 LPB
```

```
LPSETUP3          // 将循环次数减 1 后赋给 LPC
```

LOOP BEGIN 100: // 循环起始地址为 0x1000, 将 LOOP BEGIN 设为 0x1000

```
// 循环次数是 100
```

```
ADD (.A) X0, X1
```



```
||SUB (.M) X5, X6
```

```
ADD (.A) X0, X2
||SUB (.M) X5, X7
```

```
LOOP_END:
```

```
ADD (.A) X0, X3
||SUB (.M) X5, X8
```

```
// 循环结束地址为 0x1005, 结束地址和起始地址的差为 5, 将 LOOP_END 设为 5
```

SKMR

SKMR 是用于控制 STACK 的控制管理寄存器，其中包括栈指针 SP，栈顶地址 SKT，栈底地址 SKB。栈指针 SP 通过 **STACK_START** 指令进行初始化，并由 **PUSH** 和 **POP** 指令进行修改，栈顶地址和栈底地址由 **MOV.RC** 指令进行赋值。

31	0
SP	
* R	
31	0
SKT	
* R, W	
31	0
SKB	
* R, W	

IGE

IGE 寄存器中包含两个寄存器标志 IGE 和 PIGE，IGE 是全局中断使能信号，除了 **reset** 中断之外，所有中断（包括可屏蔽中断和不可屏蔽中断）均受 IGE 控制，只有当 IGE 为 1 时中断才可能被执行，当 IGE 为 0 时，除了 **reset** 之外所有的中断都将被屏蔽。

PIGE 是 IGE 的备份信号，当处理器相应中断是，当前 IGE 中的信号将被自动复制到 PIGE 当中，当执行中断返回程序时，PIGE 中的内容将被还原到 IGE 中。

Reserved		PIGE	IGE
31	2	1	0

IER

IER 为中断使能寄存器。处理器中设计了 16 个中断，其中 **reset** 中断，NMI 中断，FIFO empty 中断和 full 中断，以及 IDLE 返回中断是不可屏蔽中断，而 int0—int7 均为可屏蔽中断。每个可屏蔽中断在 IER 中有一个对应的使能位，当某一个可屏蔽中断被触发时，只有对应的中断使能标志为 1 时才会被响应，否则该中断将被屏蔽。IER 中各位的信息如下图所示。

R	R	R	R	R	R	R	R	Int7	Int6	Int5	Int4	Int3	Int2	Int1	Int0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

IFR

IFR 是中断标志寄存器。除 **reset** 之外，每一个中断在处理器中都有一个对应的中断标志寄存器，当有中断触发时，对应的中断标志位将被置为 1。处理器将根据中断标志寄存器的内容、中断使能寄存器的内容以及各个中断的优先级进入相应的中断处理程序。

R	R	R	Idle1	Idle0	Ful	Emp	NMI	Int7	Int6	Int5	Int4	Int3	Int2	Int1	Int0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

VSR

VSR: 矢量状态寄存器，包括矢量长度，宽度，以及数据通道位宽限制等，每个功能单元对应一个 VSR。可以由 MOV.RC 设置以及清除还原成默认状态。

31																	24	23															18	17	16								
Revered																																VL_XD						VWM_XD					
VL_XM										VWM_XM																						VL_XA						VWM_XA					
15																	10	9	8	7													2	1	0								

31																	24	23															18	17	16								
Revered																																VL_YD						VWM_YD					
VL_YM										VWM_YM																						VL_YA						VWM_YA					
15																	10	9	8	7													2	1	0								

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: Datapath Width, $VL * VWM \leq VMAX$

VMIN: 矢量处理最小粒度，由数据通道来约束，例如，如果乘法单元内部乘法器可以支持 8bit 乘法操作，这样 VMIN 就为 Byte 粒度，而如果乘法单元内部乘法器不支持 8bit 乘法操作，只能支持最小 16bit 数据操作，则 VMIN 就为 16bit 粒度。

RETPC

当处理器进入可屏蔽中断处理程序时，RETPC 中将自动保存中断处理的返回地址。

RETPC																																			
31																																			0

6 ARITHMETIC OPERATIONS

Instruction Summary

- ADD/ ADD.U
- ADD.2
- ADDK
- SADD.S
- SADD.S.2/.U.2/.US.2
- SUB/ SUB.U
- SUB.2
- SSUB.S
- SSUB.S.2/ SSUB.U.2
- MAX
- MAX.S.2/ MAX.U.2
- MIN
- MIN.S.2/ MIN.U.2
- AVG.S.2/ AVG.U.2
- ABS
- ABS.2
- NEG
- NEG.2
- SAT
- SAT.L
- SAT.S.2/ SAT.U.2
- DQUANT
- DQUANT.2

Arithmetic Operations

ADD/ADD.U

Signed or Unsigned Integer Addition
Without Saturation

Arithmetic Operations

ADD/ADD.U

General Form **ADD** (.type) rd, rs /* (F) */
 ADD (.type) rd, rs1, rs2 /* (D) */
 ADD.U (.type) rd, rs1,rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1				rs/cst4		rd

DT.F1

15	14	13	12	8	7	4	3	0
s	10	Op_1				rs		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0	
s	11	00	Op	Op_1				rs2/const5					rs1		rd		z	creg

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0	
s	11	01	Op	Op 1				rs2/const5					rs1		rd		z	creg

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				rs2/const5			rs1		rd		z	creg	

ADD		(F) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	10110
rs	sint	.A	
rd	sint	.A	10111
rs	ucst4	.A	
rd	sint	.M	11010
rs	sint	.M	
rd	sint	.D	10100
rs	sint	.D	

ADD		(D) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	000 00000
rs1	sint	.A	
rs2	sint		
rd	slong	.A	000 00001
rs1	sint	.A	
rs2	sint		
rd	slong	.A	000 00010
rs1	slong	.A	
rs2	sint		
rd	sint	.A	000 00011
rs1	sint	.A	
rs2	ucst5		
rd	slong	.A	000 00100
rs1	sint	.A	
rs2	ucst5		
rd	slong	.A	000 00101
rs1	slong	.A	
rs2	ucst5		
rd	sint	.M	010 00000
rs1	sint	.M	
rs2	sint		
rd	sint	.M	010 00001
rs1	sint	.M	
rs2	ucst5		
rd	sint	.D	001 10100
rs1	sint	.D	
rs2	sint		
rd	sint	.D	001 10101
rs1	sint	.D	
rs2	ucst5		

ADD.U		(D) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	ulong	.A	000 01000
rs1	uint	.A	
rs2	uint		
rd	ulong	.A	000 01001
rs1	ulong	.A	
rs2	uint		

rd	ulong	.A	000	01010
rs1	uint	.A		
rs2	ucst5			
rd	ulong	.A	000	01011
rs1	ulong	.A		
rs2	Ucst5			

Description

(F)寄存器 rs2 与寄存器 rs1 中的有符号数相加，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

(D)寄存器 rs2 与寄存器 rs1 中的有符号数(或无符号数)相加，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

Operation

(F) $rd \leq rd + rs/ucst4;$
 (D) If (cond) $rd \leq rs1 + rs2/cst5$
 Else nop

Execute time

B. cycle

Example

```

ADD (.A)    X0, X1;           /*X0 = X0 + X1*/
[X0] ADD (.A) G5, X4, X5      /*if(X0==1) G5 = X4 + X5*/
ADD.U (.A) G6:G5, X5: X4, #16 /*G6:G5 = X5:X4 + 16*/
  
```

Arithmetic Operations

ADD.2

Two 16-Bit Integer Addition on Upper
and Lower Register Halves

Arithmetic Operations

ADD.2

General Form **ADD.2** (.type) rd, rs /* (F) */
ADD.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd		z	creg		

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2/const5		rs1		rd		z	creg		

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2/const5		rs1		rd		z	creg		

ADD.2**(F) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	00100
rs	2hws	.A	
rd	2hws	.M	11100
rs	2hws	.M	

ADD.2**(D) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	000 01100
rs1	2hws	.A	
rs2	2hws		
rd	2hws	.A	000 01101
rs1	2hws	.A	
rs2	ucst5		
rd	2hws	.M	010 00100
rs1	2hws	.M	
rs2	2hws		
rd	2hws	.D	001 11000
rs1	2hws	.D	
rs2	2hws		

Description

(F)寄存器 rs2 中的两个 16bit 有符号数分别与 rs1 中对应的 16bit 有符号数相加，结果分别存放于寄存器 rd 的两个半字中。

(D)寄存器 rs2 中的两个 16bit 有符号数(或无符号数)分别与 rs1 中对应的 16bit 有符号数(或无符号数)相加，结果分别存放于寄存器 rd 的两个半字中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

Operation

```
(F) {rd_hw2s <= rd_hw2s + rs_hw2s
     rd_hw1s <= rd_hw1s + rs_hw1s }
(D) If (cond)
{   rd_hw2s <= rs1_hw2s + rs2_hw2s
    rd_hw1s <= rs1_hw1s + rs2_hw1s
}
else nop
```

Execute time

B. cycle

Example

```
ADD.2 (.A) X0, X1; /*X0_h = X0_h + X1_h; X0_l = X0_l + X1_l */
[X0] ADD.2 (.A) G5, X4, #2 /*if(X0==1) G5_h = X4_h + 2; G5_l = X4_l + 2*
```

Arithmetic Operations

ADDK

Integer Addition Using Signed 16-Bit constant

Arithmetic Operations

ADDK

General Form **ADDK (.type) rd, cst16** */* (D) */*
 .type = .A,

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

ADDK

31	30	29	28	27	26	25	24		8		4	3	2	0
s	11	00	10					scst16		rd		z		creg

Description

寄存器 rd 中的有符号数与 16 位有符号立即数相加, 结果存放于寄存器 rd 中。

Operation

If (cond) rd_sint <= rd_sint + scst16
 Else nop

Execute time

B. cycle

Example

ADDK (.A) X0, #16'h1080; */*X0 = X0 + 16'h1080 */*

Arithmetic Operations

SADD Signed Integer Addition With
Saturation

Arithmetic Operations

SADD

General Form **SADD** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SADD

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	000	10000
rs1	sint	.A		
rs2	sint			
rd	slong	.A	000	10001
rs1	slong	.A		
rs2	sint			
rd	sint	.A	000	10010
rs1	sint	.A		
rs2	ucst5			
rd	slong	.A	000	10011
rs1	slong	.A		
rs2	ucst5			

Description

寄存器 rs2 与寄存器 rs1 中的有符号数(或无符号数)相加, 结果经过饱和处理后存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时, 立即数一定为无符号立即数。

Operation

If (cond) Rd(S) $\leq$ sat(rs2(S) + rs1(S))
Else nop

saturation rules:

If rd is a sint and $rs2 + rs1 > 2^{31}-1$, then $rd \leq 2^{31}-1$;

If rd is a sint and $rs2 + rs1 < -2^{31}$, then $rd \leq -2^{31}$;

If rd is a slong and $rs2 + rs1 > 2^{39}-1$, then $rd \leq 2^{39}-1$;

If rd is a slong and $rs2 + rs1 < -2^{39}$, then $rd \leq -2^{39}$;

Execute time

B. cycle

Example

SADD (.A) X1:X0, X3:X2, X5;



Arithmetic Operations SADD.S.2/.U.2

Arithmetic Operations SADD.S.2/.U.2

Signed or Unsigned Packed 16-Bit Addition With Saturation

General Form **SADD.S.2** (.type) rd, rs1, rs2 /* (D) */
 SADD.U.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SADD.S.2 (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	2hws	.A	000	10100
rs1	2hws	.A		
rs2	2hws			
rd	2hws	.A	000	10101
rs1	2hws	.A		
rs2	ucst5			

SADD.U.2 (D) Format

rd	2hwu	.A	000	10110
rs1	2hwu	.A		
rs2	2hwu			
rd	2hwu	.A	000	10111
rs1	2hwu	.A		
rs2	ucst5			

Description

寄存器 rs2 中的两个 16bit 有符号数(或者无符号数)分别与 rs1 中对应的 16bit 有符号数(或者无符号数)相加，结果经过饱和处理后分别存放于寄存器 rd 的两个半字中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

Operation

If (cond)

```
{  Rd_hw2s/u/u <= sat(rs2_hw2s/u/u + rs1_hw2s/u/s)
  Rd_hw1s/u/u <= sat(rs2_hw1s/u/u + rs1_hw1s/u/s)
}
```

else nop

Signed saturation rules:

if the sum of two adders is the range of -2^{15} to $2^{15}-1$, no saturation is performed

if the sum of two adders is greater than $2^{15}-1$, then the result is set to $2^{15}-1$;

if the sum of two adders is less than -2^{15} , then the result is set to -2^{15} ;

Unsigned saturation rules:

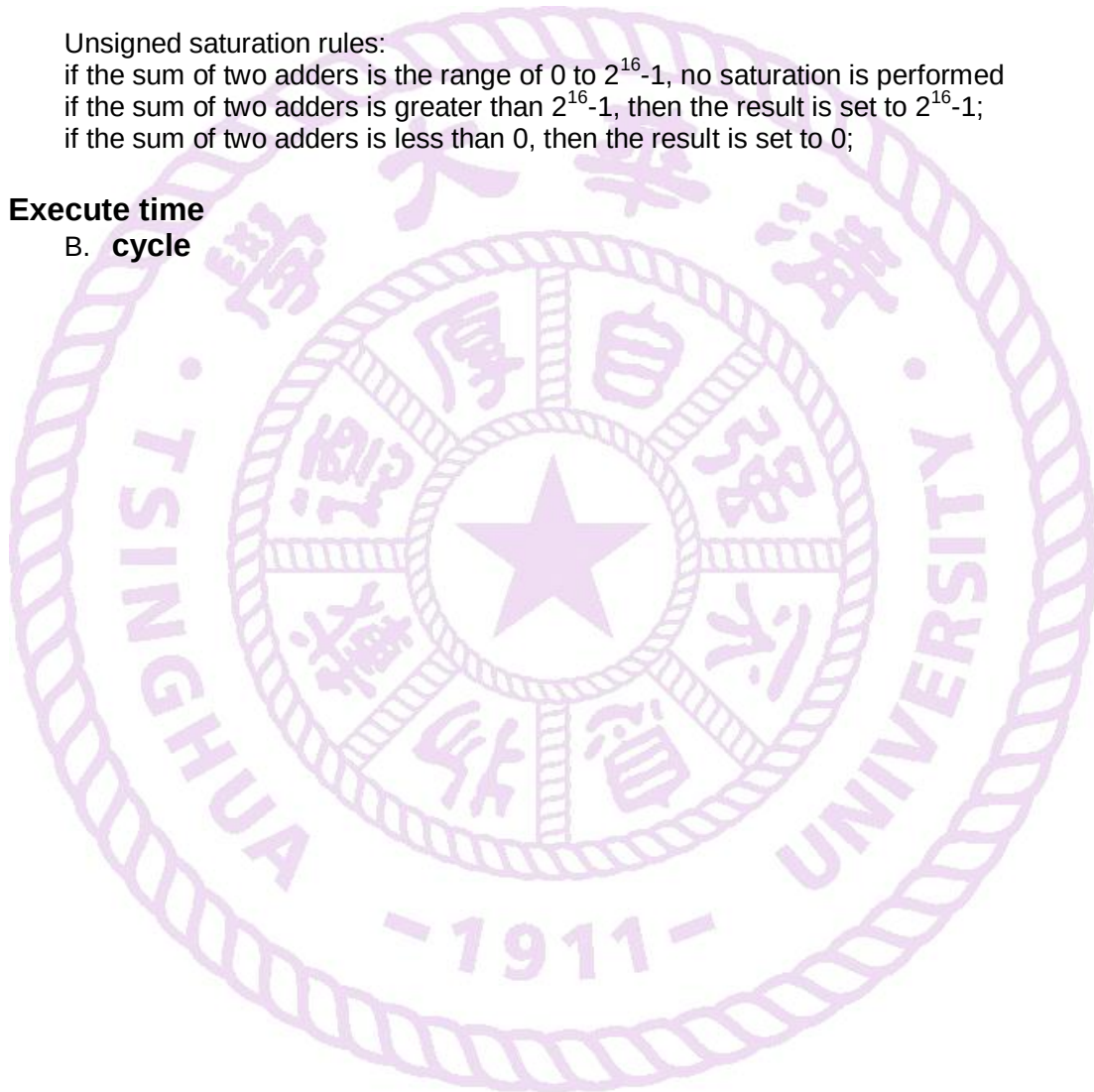
if the sum of two adders is the range of 0 to $2^{16}-1$, no saturation is performed

if the sum of two adders is greater than $2^{16}-1$, then the result is set to $2^{16}-1$;

if the sum of two adders is less than 0, then the result is set to 0;

Execute time

B. cycle



Arithmetic Operations

SUB/SUB.U

Signed or Unsigned Integer Subtraction
Without Saturation

Arithmetic Operations

SUB/SUB.U

General Form **SUB** (.type) rd, rs /* (F) */
 SUB (.type) rd, rs1, rs2 /* (D) */
 SUB.U (.type) rd, rs1,rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1		rs/cst4		rd		

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1		rs/cst4		rd		

DT.F1

15	14	13	12	8	7	4	3	0
s	10	Op_1		rs		rd		

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1	rs2/const5		rs1		rd		z	creg				

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op_1	rs2/const5		rs1		rd		z	creg				

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op		Op_1	rs2/const5		rs1		rd		z	creg				

SUB		(F) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	00010
rs	sint	.A	
rd	sint	.A	00011
rs	ucst4	.A	
rd	sint	.M	11011
rs	sint	.M	
rd	sint	.D	10101
rs	sint	.D	

SUB		(D) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	001 00000
rs1	sint	.A	
rs2	sint		
rd	slong	.A	001 00001
rs1	sint	.A	
rs2	sint		
rd	slong	.A	001 00010
rs1	slong	.A	
rs2	sint		
rd	sint	.A	001 00011
rs1	sint	.A	
rs2	ucst5		
rd	slong	.A	001 00100
rs1	sint	.A	
rs2	ucst5		
rd	slong	.A	001 00101
rs1	slong	.A	
rs2	ucst5		
rd	sint	.M	010 00010
rs1	sint	.M	
rs2	sint		
rd	sint	.M	010 00011
rs1	sint	.M	
rs2	ucst5		
rd	sint	.D	001 10110
rs1	sint	.D	
rs2	sint		
rd	sint	.D	001 10111
rs1	sint	.D	
rs2	ucst5		

SUB.U		(D) Format		Unit
Opcode map field used ...	For operand type ...		Opfield	
rd	ulong	.A	001 01000	
rs1	uint	.A		
rs2	uint			
rd	ulong	.A	001 01001	
rs1	ulong	.A		
rs2	uint			

rd	ulong	.A	001	01010
rs1	uint	.A		
rs2	ucst5			
rd	ulong	.A	001	01011
rs1	ulong	.A		
rs2	Ucst5			

Description

(F) 寄存器 rs2 与寄存器 rs1 中的有符号数相减，结果存放于寄存器 rd 中。
当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

(D) 寄存器 rs2 与寄存器 rs1 中的有符号数(或无符号数)相减，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

Operation

(F) $rd \leq rd - rs;$

(D) If (cond) $rd \leq rs1 - rs2/cst5$
Else nop

Execute time

B. cycle

Example

```

SUB (.A)X0, X1;          /*X0 = X0 - X1*/
[X0] SUB (.A)G5, X4, X5   /*if(X0==1) G5 = X4 - X5*/
SUB.U (.A) G6:G5, X5:X4, #16 /*G6:G5 = X5:X4 - 16*/

```

Arithmetic Operations

SUB.2

Two 16-Bit Integer Subtraction on
Upper and Lower Register Halves

Arithmetic Operations

SUB.2

General Form **SUB.2** (.type) rd, rs /* (F) */
SUB.2 (.type) rd, rs1, rs2 /* (D) */
.type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z		creg			

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1		rs2/const5		rs1		rd		z		creg			

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1		rs2/const5		rs1		rd		z		creg			

SUB.2**(F) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	00101
rs	2hws	.A	
rd	2hws	.M	11101
rs	2hws	.M	

SUB.2**(D) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	001 01100
rs1	2hws	.A	
rs2	2hws		
rd	2hws	.A	001 01101
rs1	2hws	.A	
rs2	ucst5		
rd	2hws	.M	010 00101
rs1	2hws	.M	
rs2	2hws		
rd	2hws	.D	001 11001
rs1	2hws	.D	
rs2	2hws		

Description

(F)寄存器 rs2 中的两个 16bit 有符号数分别与 rs1 中对应的 16bit 有符号数相减，结果分别存放于寄存器 rd 的两个半字中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

(D)寄存器 rs2 中的两个 16bit 有符号数(或无符号数)分别与 rs1 中对应的 16bit 有符号数(无符号数)相减，结果分别存放于寄存器 rd 的两个半字中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

Operation

```
(F) {rd_hw2s <= rd_hw2s - rs_hw2s
     rd_hw1s <= rd_hw1s - rs_hw1s }
(D) If (cond)
{   rd_hw2s <= rs1_hw2s - rs2_hw2s
    rd_hw1s <= rs1_hw1s - rs2_hw1s
}
else nop
```

Execute time

B. cycle

Example

```
SUB.2 (.A) X0, X1; /*X0_h = X0_h - X1_h; X0_l = X0_l - X1_l */
[X0] SUB.2 (.A) G5, X4, #2 /*if(X0==1) G5_h = X4_h - 2; G5_l = X4_l - 2*/
```

SSUB

SSUB

Signed or Unsigned Integer Subtraction With Saturation

General Form **SSUB** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd		z		creg	

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	001	10000
rs1	sint	.A		
rs2	sint			
rd	slong	.A	001	10001
rs1	slong	.A		
rs2	sint			
rd	sint	.A	001	10010
rs1	sint	.A		
rs2	ucst5			
rd	slong	.A	001	10011
rs1	slong	.A		
rs2	ucst5			

寄存器 **rs2** 与寄存器 **rs1** 中的有符号数(或无符号数)相减, 结果经过饱和处理后存放于寄存器 **rd** 中。当源操作数 **rs2** 为 5bit 立即数时, 立即数一定为无符号立即数。

Operation

If (cond) Rd(S) $\leq$ sat(rs2(S) – rs1(S))
Else nop

saturation rules:

If rd is a sint and $rs2 + rs1 > 2^{31}-1$, then $rd \leq 2^{31}-1$;

If rd is a sint and $rs2 + rs1 < -2^{31}$, then $rd \leq -2^{31}$;

If rd is a slong and $rs2 + rs1 > 2^{39}-1$, then $rd \leq 2^{39}-1$;

If rd is a slong and $rs2 + rs1 < -2^{39}$, then $rd \leq -2^{39}$;

Execute time

B. cycle

Example

SSUB (.A) X1:X0, X3:X2, X5;



Arithmetic Operations

SSUB.S.2/U.2

Signed or Unsigned Packed 16-Bit
Subtraction With Saturation

Arithmetic Operations

SSUB.S.2/U.2

General Form **SSUB.S.2** (.type) rd, rs1, rs2 /* (D) */
 SSUB.U.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SSUB.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	2hws	.A	001 10100
rs1	2hws	.A	
rs2	2hws		
Rd	2hws	.A	001 10101
rs1	2hws	.A	
rs2	ucst5		

SSUB.U.2

(D) Format

rd	2hwu	.A	001 10110
rs1	2hwu	.A	
rs2	2hwu		
rd	2hwu	.A	001 10111
rs1	2hwu	.A	
rs2	ucst5		

Description

寄存器 rs2 中的两个 16bit 有符号数(或无符号数)分别与 rs1 中对应的 16bit 有符号数(或无符号数)相减，结果经过饱和处理后分别存放于寄存器 rd 的两个半字中。当源操作数 rs2 为 5bit 立即数时，立即数一定为无符号立即数。

Operation

```
If (cond)
{   Rd_hw2s/u <= sat(rs2_hw2s/u - rs1_hw2s/u)
    Rd_hw1s/u <= sat(rs2_hw1s/u - rs1_hw1s/u)
}
else nop
```

Signed saturation rules:

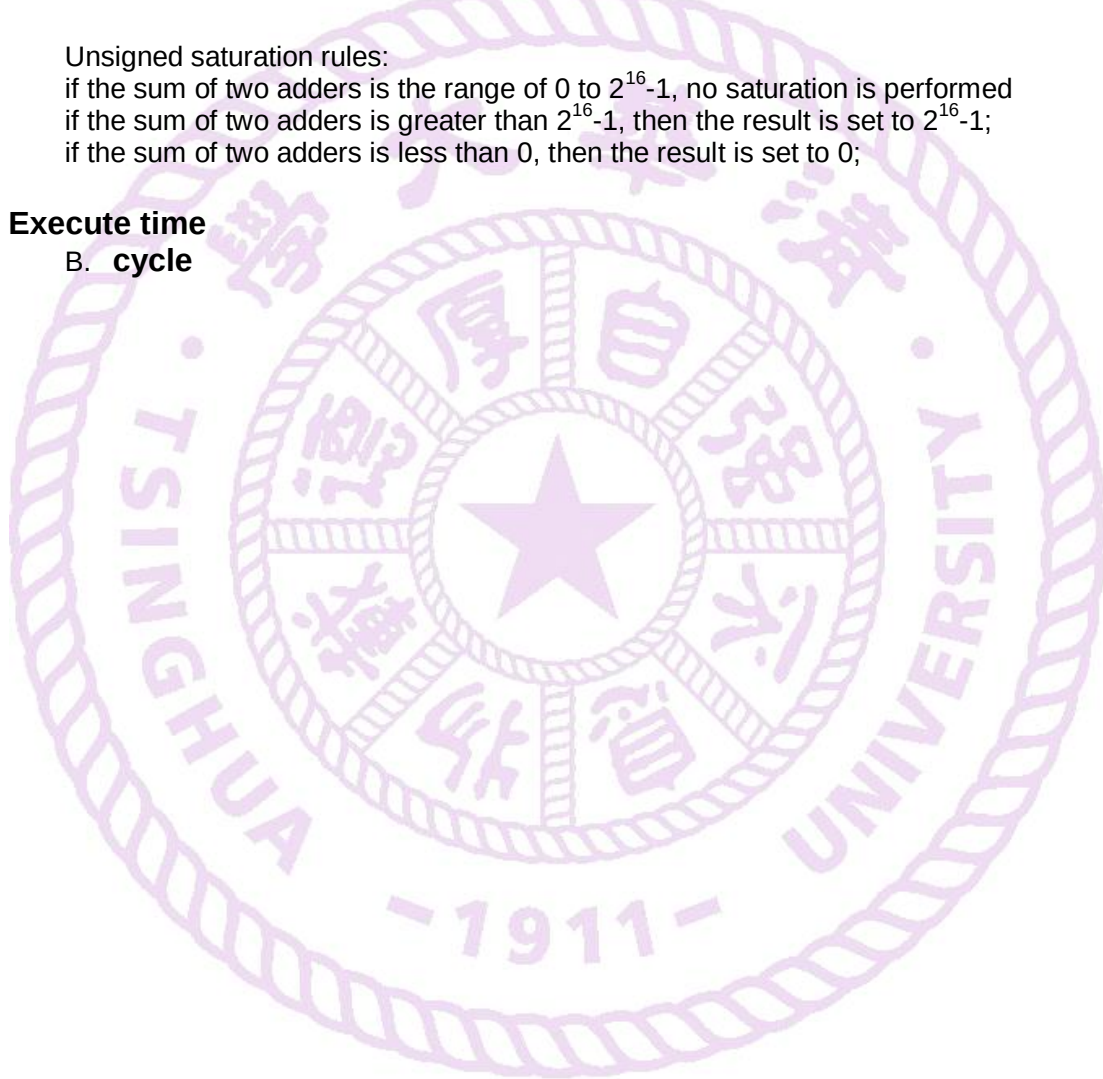
if the sum of two adders is the range of -2^{15} to $2^{15}-1$, no saturation is performed
if the sum of two adders is greater than $2^{15}-1$, then the result is set to $2^{15}-1$;
if the sum of two adders is less than -2^{15} , then the result is set to -2^{15} ;

Unsigned saturation rules:

if the sum of two adders is the range of 0 to $2^{16}-1$, no saturation is performed
if the sum of two adders is greater than $2^{16}-1$, then the result is set to $2^{16}-1$;
if the sum of two adders is less than 0, then the result is set to 0;

Execute time

B. cycle



MAX

Maximum, Signed or Unsigned 32-Bit

Arithmetic Operations

MAX

General Form **MAX** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd			z	creg	

MAX

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Sint	.A	011	11100
rs1	Sint	.A		
rs2	sint			

Description

寄存器 rs2 中的 32bit 有符号数与 rs1 中对应的 32bit 有符号数做比较取最大值，结果放在目标寄存器中。

Operation

```

if (cond)
{   rd <= max(rs2,rs1)
}
else nop

```

Execute time

B. cycle

Arithmetic Operations

MAX.S.2/.U.2

Maximum, Signed or Unsigned Packed
16-Bit

Arithmetic Operations

MAX.S.2/.U.2

General Form **MAX.S.2** (.type) rd, rs1, rs2 /* (D) */
 MAX.U.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MAX.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	2hws	.A	011 00000
rs1	2hws	.A	
rs2	2hws		

MAX.U.2

(D) Format

Rd	2hwu	.A	011 00001
rs1	2hwu	.A	
rs2	2hwu		

Description

寄存器 rs2 中的两个 16bit 有符号数或无符号数分别与 rs1 中对应的 16bit 有符号数或无符号数做比较取最大值，结果放在目标寄存器的两个 16Bit 中。

Operation

```

If (cond)
{
    rd_hw2s/u <= max(rs2_hw2s/u,rs1_hw2s/u)
    rd_hw1s/u <= max(rs2_hw1s/u,rs1_hw1s/u)
}
else nop
    
```

Execute time

B. cycle

MIN

Minimum, Signed or Unsigned 32-Bit

Arithmetic Operations

MIN

General Form **MAX** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd			z	creg	

MIN

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Sint	.A	011	11101
rs1	Sint	.A		
rs2	sint			

Description

寄存器 rs2 中的 32bit 有符号数与 rs1 中对应的 32bit 有符号数做比较取最小值，结果放在目标寄存器中。

Operation

```

If (cond)
{
    rd <= min(rs2,rs1)
}
else nop

```

Execute time

B. cycle

Arithmetic Operations

MIN.S.2/.U.2

Minimum, Signed or Unsigned Packed
16-Bit

Arithmetic Operations

MIN.S.2/.U.2

General Form **MIN.S.2** (.type) rd, rs1, rs2 /* (D) */
 MIN.U.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MIN.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	011 00010
rs1	2hws	.A	
rs2	2hws		

MIN.U.2

(D) Format

rd	2hwu	.A	011 00011
rs1	2hwu	.A	
rs2	2hwu		

Description

寄存器 rs2 中的两个 16bit 有符号数或无符号数分别与 rs1 中对应的 16bit 有符号数或无符号数做比较取最小值，结果放在目标寄存器的两个 16Bit 中。

Operation

```

If (cond)
{
    rd_hw2s/u <= min(rs2_hw2s/u,rs1_hw2s/u)
    rd_hw1s/u <= min(rs2_hw1s/u,rs1_hw1s/u)
}
else nop
    
```

Execute time

B. cycle

Arithmetic Operations

AVG.S.2/U.2

Average, Signed or Unsigned Packed
16-Bit

Arithmetic Operations

AVG.S.2/U.2

General Form **AVG.S.2** (.type) rd, rs1, rs2 /* (D) */
 AVG.U.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

AVG.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	000 11100
rs1	2hws	.A	
rs2	2hws		

AVG.U.2

(D) Format

rd	2hwu	.A	000 11101
rs1	2hwu	.A	
rs2	2hwu		

Description

寄存器 rs2 中的两个 16bit 有符号数(或无符号数)分别与 rs1 中对应的 16bit 有符号数(或无符号数)取平均值, 结果放在目标寄存器的两个 16Bit 中。

Operation

```

If (cond)
{
    rd_hw2s/u = (rs2_hw2s/u + rs1_hw2s/u + 1) >>1
    rd_hw1s/u = (rs2_hw1s/u + rs1_hw1s/u + 1) >>1
}
else nop
    
```

Execute time

B. cycle

Arithmetic Operations

ABS

Integer Absolute Value with Saturation

Arithmetic Operations

ABS

General Form **ABS** (.type) rd /* (F) */
 ABS (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F2

15	14	13	12	8	7	5	4	0
s	00	Op_1			Op_2		rd	

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1			Op_2			rs1		rd		z	creg	

ABS

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	11110	000
		.A		

ABS

(D) Format

rd	sint	.A	011	11111	00000
rs	sint	.A			
rd	slong	.A	011	11111	00001
rs	slong	.A			

Description

计算寄存器 rs 中的有符号数的绝对值，结果存放于寄存器 rd 中。如果源寄存器中的值为 -2^{31} ，则结果为 $2^{31}-1$ 。

Operation

(F) If($-2^{31}+1 \leq rd \leq 2^{31}-1$) $rd \leq |rd|$
else $rd \leq 2^{31}-1$

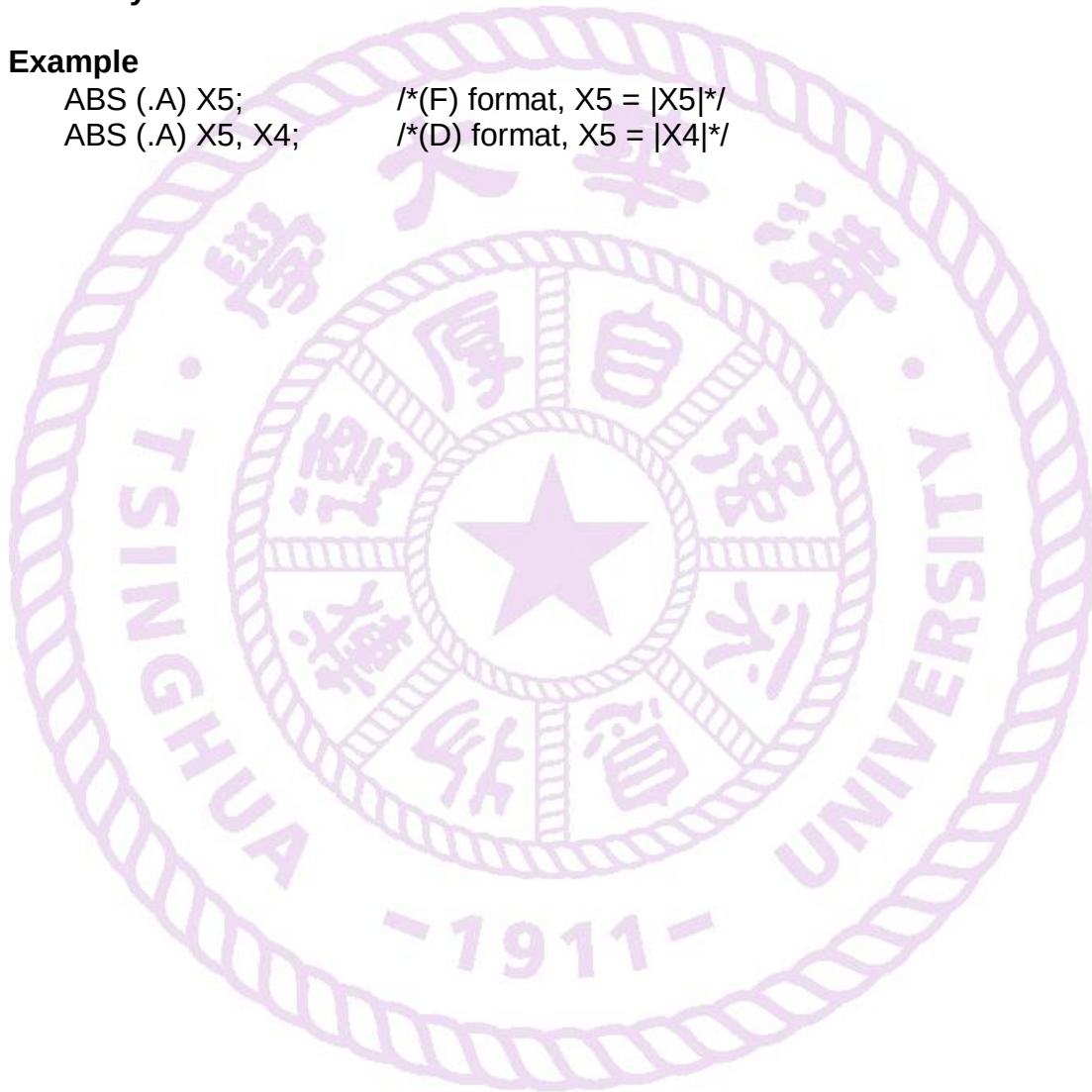
(D) If (cond)
If($-2^{31}+1 \leq rs \leq 2^{31}-1$) $rd \leq |rs|$
else $rd \leq 2^{31}-1$
Else nop

Execute time

B. cycle

Example

ABS (.A) X5; /*(F) format, X5 = |X5|*/
ABS (.A) X5, X4; /*(D) format, X5 = |X4|*/



Arithmetic Operations

ABS.2

Packed 16-Bit Absolute Value with
Saturation

Arithmetic Operations

ABS.2

General Form **ABS.2** (.type) rd /* (F) */
 ABS.2 (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F2

15	14	13	12	8	7	5	4	0
s	00	Op_1			Op_2		rd	

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1			Op_2			rs1		rd		z	creg	

ABS.2

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	11110 001
		.A	

ABS.2

(D) Format

rd	2hws	.A	011 1111 00010
rs	2hws	.A	

Description

计算寄存器 rs 中的两个 16Bit 有符号数的绝对值，结果存放于寄存器 rd 中。
 如果源寄存器中的值为 -2^{15} ，则结果为 $2^{15}-1$ 。

Operation

- If $(-2^{15}+1 \leq rd\_hw2s \leq 2^{15}-1)$ $rd\_hw2s \leq |rd\_hw2s|$;
 else $rd\_hw2s \leq 2^{15}-1$;
 If $(-2^{15}+1 \leq rd\_hw1s \leq 2^{15}-1)$ $rd\_hw1s \leq |rd\_hw1s|$;
 else $rd\_hw1s \leq 2^{15}-1$;

(D) If (cond)

- If $(-2^{15}+1 \leq rd\_hw2s \leq 2^{15}-1)$ $rd\_hw2s \leq |rd\_hw2s|$;
 else $rd\_hw2s \leq 2^{15}-1$;
 If $(-2^{15}+1 \leq rd\_hw1s \leq 2^{15}-1)$ $rd\_hw1s \leq |rd\_hw1s|$;
 else $rd\_hw1s \leq 2^{15}-1$;
 Else nop

Execute time

B. cycle



Arithmetic Operations

NEG

Negate

Arithmetic Operations

NEG

General Form **NEG** (.type) rd /* (F) */
 NEG (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F2

15	14	13	12	8	7	5	4	0
s	00	Op_1			Op_2		rd	

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1			Op_2			rs1		rd		z	creg	

NEG

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	11110 010
		.A	

NEG

(D) Format

rd	sint	.A	011	11111	01000
rs	sint	.A			
rd	slong	.A	011	11111	01001
rs	slong	.A			

Description

(F) 计算寄存器 rd 中的一个 32bit 有符号数的相反数，结果存放于寄存器 rd 中。

(D) 计算寄存器 rs 中的一个 32bit 有符号数的相反数，结果存放于寄存器 rd 中。

Operation

(F) rd_hw2s = -rd_hw2s
rd_hw1s = -rd_hw1s

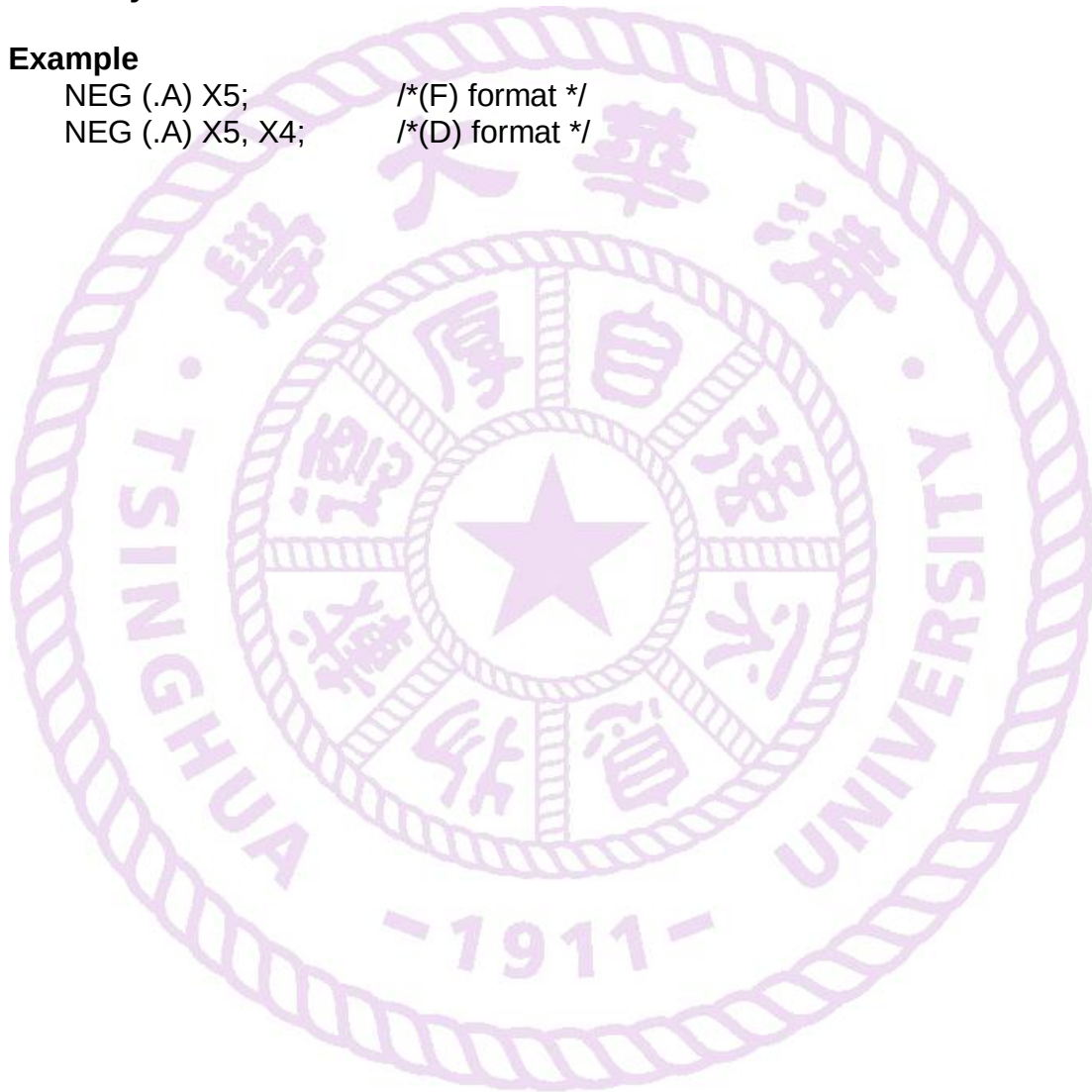
- If (cond)
rd_hw2s = -rd_hw2s
rd_hw1s = -rd_hw1s
Else nop

Execute time

B. cycle

Example

NEG (.A) X5; /*(F) format */
NEG (.A) X5, X4; /*(D) format */



Arithmetic Operations

NEG.2

Packed 16-Bit Negate

Arithmetic Operations

NEG.2

General Form **NEG.2** (.type) rd /* (F) */
 NEG.2 (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

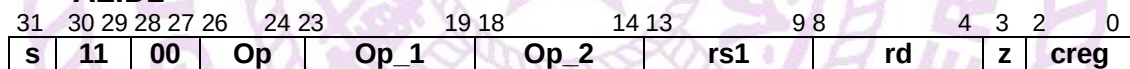
(F) Format

AL.F2



(D) Format

AL.D2



NEG.2 (F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	11110 011
		.A	

NEG.2 (D) Format

rd	2hws	.A	011 1111 01010
rs	2hws	.A	

Description

(F) 计算寄存器 rd 中的两个 16bit 有符号数的相反数，结果存放于寄存器 rd 中。

(D) 计算寄存器 rs 中的两个 16Bit 有符号数的相反数，结果存放于寄存器 rd 中。

Operation

- (F) $rd = -rd$
- (D) If (cond) $rd = -rs$
Else nop

Execute time

- B. cycle



Arithmetic Operations

SAT

Saturate a 40-Bit Integer to a 32-Bit Integer

SAT

General Form **SAT** (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				Op 2		rs1		rd		z	creg		

SAT

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1111 00100
rs	slong	.A	

Description

(D)对寄存器 rs 中的 40bit 有符号数做饱和处理, 结果存放于 32bit 寄存器 rd 中。

Operation

```
(D) If (cond) rd = sat(rs+1:rs)
    Else nop
```

Execute time

B. cycle

Arithmetic Operations

SAT.L

Saturate a 32-Bit Integer to a 16-Bit Half word

Arithmetic Operations

SAT.L

General Form **SAT.L** (.type) rd /* (F) */
 SAT.L (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F2

15	14	13	12	8	7	5	4	0
s	00	Op_1			Op_2		rd	

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1			Op_2			rs1		rd		z	creg	

SAT.L

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	11111 001
		.A	

SAT.L

(D) Format

rd	hw1s	.A	011 1111 00101
rs	sint	.A	

Description

(F)计算寄存器 rd 中的 32bit 有符号数的 16 位饱和值,结果存放于 32bit 寄存器 rd 的低 16bi 中, 高 16 位进行符号扩展。

(D)计算寄存器 rs 中的 32bit 有符号数的 16 位饱和值,结果存放于 32bit 寄存器 rd 的低 16bit 中, 高 16 位进行符号扩展。

Operation

- (F) $rd\_hw1s = sat(rd)$
- (D) If (cond) $rd\_hw1s = sat(rs)$
Else nop

Execute time

- B. cycle



Arithmetic Operations

SAT.S.2/U.2

Saturate Signed or Unsigned Packed
16-Bit to Bytes

Arithmetic Operations

SAT.S.2/U.2

General Form

SAT.S.2 (.type) rd	/* (F) */
SAT.S.2 (.type) rd, rs	/* (D) */
SAT.U.2 (.type) rd	/* (F) */
SAT.U.2 (.type) rd, rs	/* (D) */

.type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

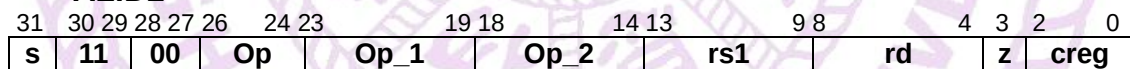
(F) Format

AL.F2



(D) Format

AL.D2



SAT.S.2

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	11111 010
		.A	

SAT.S.2

(D) Format

rd	2byte	.A	011 1111 00110
rs	2hws	.A	

SAT.U.2		(F) Format		
Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	2hws	.A	11111	011
		.A		

SAT.U.2		(D) Format		
rd	2byte	.A	011	11111
rs	2hws	.A	00111	

Description

(F)计算寄存器 rd 中的两个 16bit 有符号数的饱和值，结果存放于寄存器 rd 两个 16 比特的低 8bi 中。

(D)计算寄存器 rs 中的两个 16bit 有符号数的饱和值，结果存放于寄存器 rd 的两个 16 比特的低 8bit 中。

Operation

(F) $rd\_byte2s/u = sat(rd\_hw2s/u)$

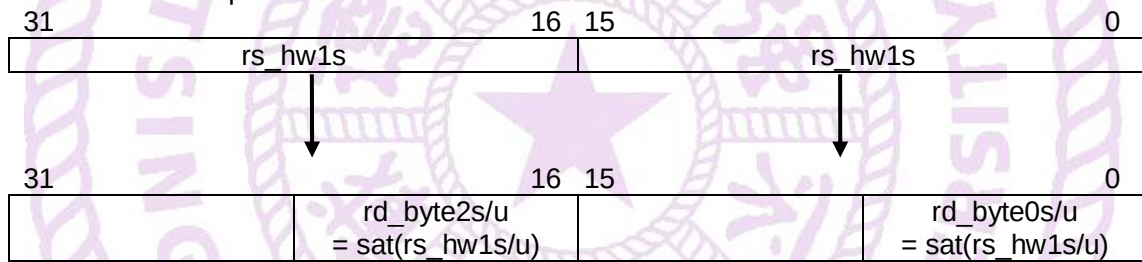
$rd\_byte0s/u = sat(rd\_hw1s/u)$

- If (cond)

$rd\_byte2s/u = sat(rs\_hw2s/u)$

$rd\_byte0s/u = sat(rs\_hw1s/u)$

Else nop



Execute time

B. cycle

Shift/Rotate Operations

DQUANT

De-quantization

Shift/Rotate Operations

DQUANT

General Form **DQUANT** (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

DQUANT

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11111 10010
rs	sint	.A	

Description

将寄存器 rs 中的有符号数的绝对值后左移一位，然后加 1，将结果进行和 rs1 相同的符号扩展后存放于寄存器 rd 中。

Operation

(D) If (cond)
 If(rs > 0) rd = 2*rs + 1;
 Else if(rs<0) rd = 2*rs -1;
 Else rd = 0;

Else nop

Execute time

B. cycle

DQUANT.2

Packed 16-bit De-quantization

Shift/Rotate Operations

DQUANT.2

General Form **DQUANT.2 (.type) rd, rs**
.type = .A

/* (D) */

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				Op 2			rs1		rd		z	creg	

DQUANT.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	2hws	.A	011	11111	10011
rs	2hws	.A			

Description

将寄存器 **rs** 中的低 16 位有符号数的绝对值后左移一位，然后加 1，将结果进行和 **rs** 的低 16 位相同的符号扩展后存放于寄存器 **rd** 的低 16 位中；
将寄存器 **rs** 中的高 16 位有符号数的绝对值后左移一位，然后加 1，将结果进行和 **rs** 的高 16 位相同的符号扩展后存放于寄存器 **rd** 的高 16 位中。

Operation

(D) If (cond)

```

If(rs_hw1s > 0)      rd_hw1s = 2* rs_hw1s + 1;
Else if(rs_hw1s <0)  rd_hw1s = 2* rs_hw1s -1;
Else                 rd_hw1s = 0;

```

```

If(rs_hw2s > 0)      rd_hw2s = 2* rs_hw2s + 1;
Else if(rs_hw2s < 0) rd_hw2s = 2* rs_hw2s -1;
Else                 rd_hw2s = 0;

```

Else nop

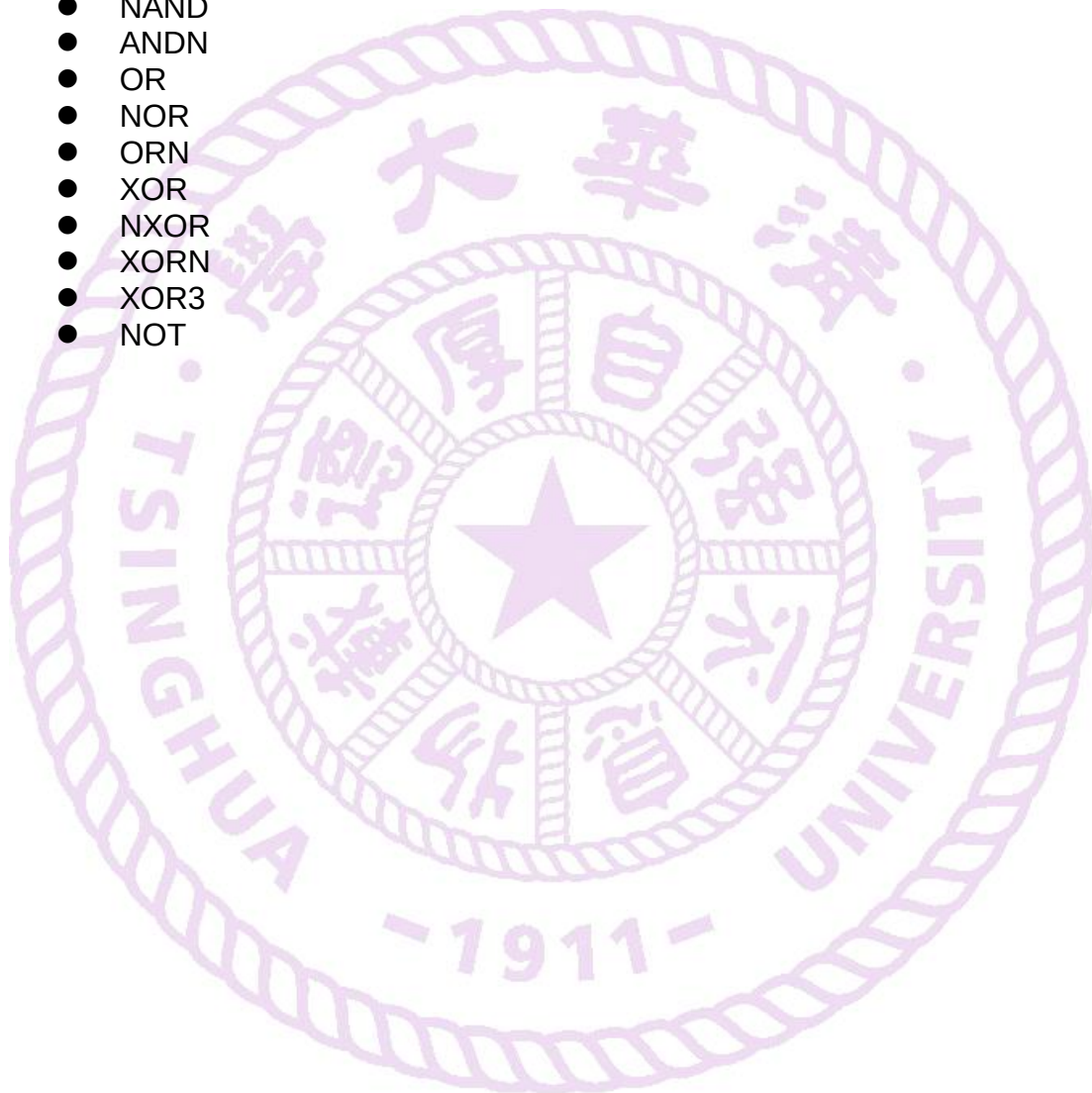
Execute time

B. cycle

7 LOGICAL OPERATIONS

Instruction Summary

- AND
- NAND
- ANDN
- OR
- NOR
- ORN
- XOR
- NXOR
- XORN
- XOR3
- NOT



Logical Operations AND

Bitwise AND

Logical Operations AND

General Form **AND** (.type) rd, rs /* (F) */
 .type = .A

AND (.type) rd, rs1, rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1			rs2/const5		rs1		rd		z	creg		

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op_1			rs2/const5		rs1		rd		z	creg		

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op		Op_1			rs2/const5		rs1		rd		z	creg		

AND		(F) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.A	11000
rs	int	.A	

AND		(D) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	011 01100
rs1	Int	.A	
rs2	Int		
rd	Int	.A	011 01101
rs1	Int	.A	
rs2	scst5		
rd	Int	.M	010 00110
rs1	Int	.M	
rs2	Int		
rd	Int	.D	001 11010
rs1	Int	.D	
rs2	Int		

Description

(F) 寄存器 rd 中的数据与寄存器 rs 中的数据进行与运算，结果存放于寄存器 rd 中。

(D) 寄存器 rs2 中的数据与寄存器 rs1 中的数据进行与运算，结果存放于寄存器 rd 中。

Operation

(F) rd = rd & rs

(D) If (cond)

{ rd = rs1 & rs2 }

else nop

Execute time

B. cycle

Example

```

    AND (.A)    X0, X1;           /*X0= X0 & X1 */
[X0] AND (.M)  G5, X4, X10       /*if(X0==1) G5 = X4 & X10*/

```


Logical Operations

NAND

Bitwise NAND

Logical Operations

NAND

General Form	NAND (.type) rd, rs	/* (F) */
	.type = .A	

```
NAND (.type) rd, rs1, rs2    /* (D) */
.type = .A, .M, .D
```

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

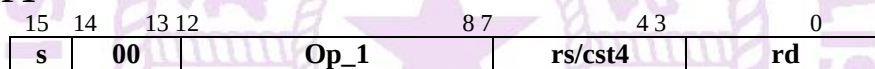
Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

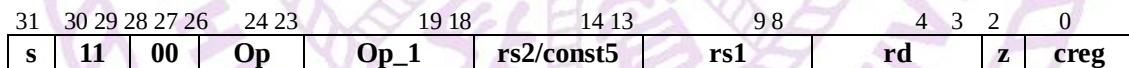
(F) Format

AL.F1

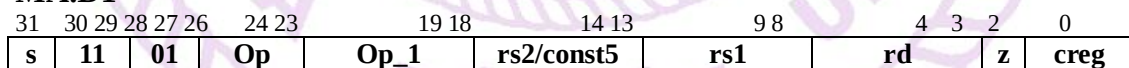


(D) Format

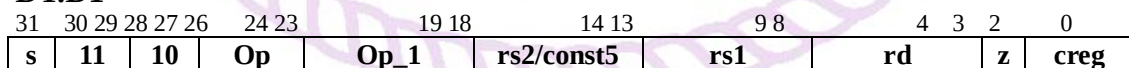
AL.D1



MA.D1



DT.D1



NAND**(F) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.A	11001
rs	int	.A	

NAND**(D) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	011 01110
rs1	Int	.A	
rs2	Int		
rd	Int	.A	011 01111
rs1	Int	.A	
rs2	scst5		
rd	Int	.M	010 00111
rs1	Int	.M	
rs2	Int		
rd	Int	.D	001 11011
rs1	Int	.D	
rs2	Int		

Description

(F) 寄存器 rd 中的数据与寄存器 rs 中的数据进行与非运算，结果存放于寄存器 rd 中。

(D) 寄存器 rs2 中的数据与寄存器 rs1 中的数据进行与非运算，结果存放于寄存器 rd 中。

Operation

(F) $rd = \sim(rd \& rs)$

(D) If (cond)

{ $rd = \sim(rs1 \& rs2)$ }

else nop

Execute time

B. cycle

Example

```

NAND (.A)  X0, X1;          /*X0= ~(X0 & X1) */
[X0]NAND (.M) G5, X4, X10    /*if(X0==1) G5 = ~(X4 & X10)*/

```

Logical Operations

ANDN

Bitwise ANDN

Logical Operations

ANDN

General Form **ANDN (.type) rd, rs1, rs2 /* (D) */**
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2	rs1	rd	z	creg								

ANDN

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	110 01101
rs1	Int	.A	
rs2	Int		

Description

(D)寄存器 rs2 中的数据取反后与寄存器 rs1 中的数据进行与运算,结果存放于寄存器 rd 中。

Operation

(D)If (cond)
{ rd = rs1 & ~rs2 }
else nop

Execute time

B. cycle

Example

[X0]ANDN (.M) G5, X4, X10 /*if(X0==1) G5 = X4 & ~X10*/

Logical Operations OR

Bitwise OR

Logical Operations OR

General Form **OR** (.type) rd, rs /* (F) */
 .type = .A

OR (.type) rd, rs1, rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1			rs2/const5		rs1		rd		z	creg		

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op_1			rs2/const5		rs1		rd		z	creg		

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op		Op_1			rs2/const5		rs1		rd		z	creg		

OR		(F) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.A	11010
rs	int	.A	

OR		(D) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	011 10000
rs1	Int	.A	
rs2	Int		
rd	Int	.A	011 10001
rs1	Int	.A	
rs2	scst5		
rd	Int	.M	010 01000
rs1	Int	.M	
rs2	Int		
rd	Int	.D	001 11100
rs1	Int	.D	
rs2	Int		

Description

(F)寄存器 rd 中的数据与寄存器 rs 中的数据进行或运算，结果存放于寄存器 rd 中。

(D)寄存器 rs2 中的数据与寄存器 rs1 中的数据进行或运算，结果存放于寄存器 rd 中。

Operation

(F) $rd = rd \mid rs$

(D) If (cond)

{ $rd = rs1 \mid rs2$
else nop

Execute time

B. cycle

Logical Operations NOR

Bitwise NOR

Logical Operations NOR

General Form **NOR (.type) rd, rs** /* (F) */
 .type = .A

NOR (.type) rd, rs1, rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1				rs2/const5			rs1		rd		z	creg

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op_1				rs2/const5			rs1		rd		z	creg

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op		Op_1				rs2/const5			rs1		rd		z	creg

NOR**(F) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.A	11011
rs	int	.A	

NOR**(D) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	011 10010
rs1	Int	.A	
rs2	Int		
rd	Int	.A	011 10011
rs1	Int	.A	
rs2	scst5		
rd	Int	.M	010 01001
rs1	Int	.M	
rs2	Int		
rd	Int	.D	001 11101
rs1	Int	.D	
rs2	Int		

Description

(F)寄存器 rd 中的数据与寄存器 rs 中的数据进行或非运算，结果存放于寄存器 rd 中。

(D)寄存器 rs2 中的数据与寄存器 rs1 中的数据进行或非运算，结果存放于寄存器 rd 中。

Operation

(F) $rd = \sim(rd \mid rs)$

(D) If (cond)

{ $rd = \sim(rs1 \mid rs2)$

else nop

Execute time

B. cycle

Logical Operations

ORN

Bitwise ORN

Logical Operations

ORN

General Form **ORN** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2	rs1	rd	z	creg								

NOR (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	110 01110
rs1	Int	.A	
rs2	Int		

Description

(D)寄存器 rs2 中的数据取反后与寄存器 rs1 中的数据进行或运算,结果存放于寄存器 rd 中。

Operation

(D)If (cond)
 { rd = rs1 | ~rs2}
 else nop

Execute time

B. cycle

Logical Operations XOR

Bitwise XOR

Logical Operations XOR

General Form **XOR** (.type) rd, rs /* (F) */
 .type = .A
 XOR (.type) rd, rs1, rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1



(D) Format

AL.D1



MA.D1



DT.D1



XOR		(F) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.A	11100
rs	int	.A	

XOR		(D) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	011 10100
rs1	Int	.A	
rs2	Int		
rd	Int	.A	011 10101
rs1	Int	.A	
rs2	scst5		
rd	Int	.M	010 01010
rs1	Int	.M	
rs2	Int		
rd	Int	.D	001 11110
rs1	Int	.D	
rs2	Int		

Description

(F)寄存器 rd 中的数据与寄存器 rs 中的数据进行异或运算，结果存放于寄存器 rd 中。

(D)寄存器 rs2 中的数据与寄存器 rs1 中的数据进行异或运算，结果存放于寄存器 rd 中。

Operation

(F) $rd = rd \wedge rs$

(D) If (cond)

{ $rd = rs1 \wedge rs2$

else nop

Execute time

B. cycle

NXOR
Logical Operations
NXOR

Bitwise NXOR

Logical Operations
NXOR

General Form NXOR (.type) rd, rs /* (F) */
 .type = .A

 NXOR (.type) rd, rs1, rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1				rs2/const5			rs1		rd		z	creg

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op_1				rs2/const5			rs1		rd		z	creg

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op		Op_1				rs2/const5			rs1		rd		z	creg

NXOR**(F) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.A	11101
rs	int	.A	

NXOR**(D) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	011 10110
rs1	Int	.A	
rs2	Int		
rd	Int	.A	011 10111
rs1	Int	.A	
rs2	scst5		
rd	Int	.M	010 01011
rs1	Int	.M	
rs2	Int		
rd	Int	.D	001 11111
rs1	Int	.D	
rs2	Int		

Description

(F)寄存器 rd 中的数据与寄存器 rs 中的数据进行异或非运算，结果存放于寄存器 rd 中。

(D)寄存器 rs2 中的数据与寄存器 rs1 中的数据进行异或非运算，结果存放于寄存器 rd 中。

Operation

(F) $rd = \sim(rd \wedge rs)$

(D) If (cond)

{ $rd = \sim(rs1 \wedge rs2)$

else nop

Execute time

B. cycle

Logical Operations

XORN

Bitwise XORN

Logical Operations

XORN

General Form **XORN** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2	rs1	rd	z	creg								

XORN

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	Int	.A	110 10011
rs1	Int	.A	
rs2	Int		

Description

(D)寄存器 rs2 中的数据取反后与寄存器 rs1 中的数据进行异或运算，结果存放于寄存器 rd 中。

Operation

(D)If (cond)
 { rd = rs1 ^ ~rs2 }
 else nop

Execute time

B. cycle

Logical Operations

XOR3

Bitwise XOR3

Logical Operations

XOR3

General Form **XOR3** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2		rs1		rd		z		creg			

XOR3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	Int	.A	110	10100
rs1	Int	.A		
rs2	Int			

Description

(D)寄存器 rs2 中与 rs1 中与 rs1+1 中的数据进行异或运算，结果存放于寄存器 rd 中。

Operation

```
(D) If (cond)
{ rd = rs1 ^ rs1+1 ^ rs2 }
else nop
```

Execute time

B. cycle

Logical Operations

NOT

Bitwise NOT

Logical Operations

NOT

General Form **NOT** (.type) rd /* (F) */
 NOT (.type) rd, rs /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

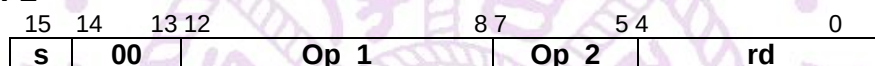
Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

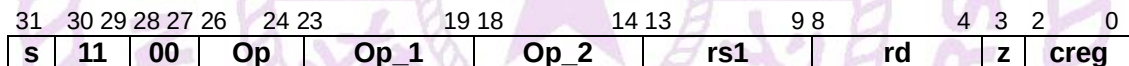
(F) Format

AL.F2



(D) Format

AL.D2



NOT

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	11110	100
		.A		

NOT

(D) Format

rd	sint	.A	011	11111	01110
rs	sint	.A			

Description

(F)计算寄存器 rd 中的有符号数的按位相反，结果存放于寄存器 rd 中。

(D)计算寄存器 rs 中的有符号数的按位相反，结果存放于寄存器 rd 中。

Operation

(F) rd = not rd

(D) If (cond) rd = not rd

Else nop

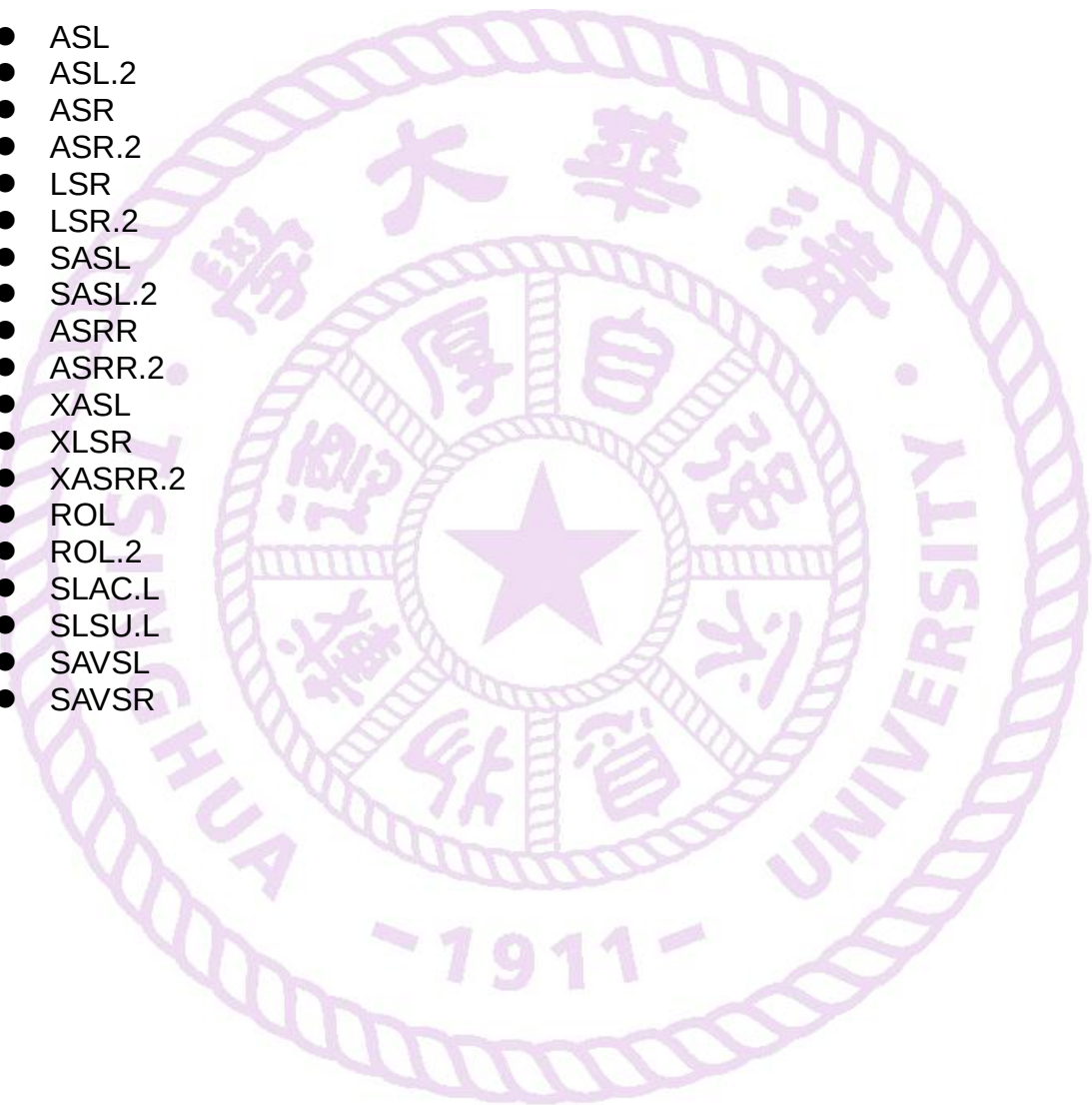
Execute time

B. cycle

8 SHIFT/ROTATE OPERATIONS

Instruction Summary

- ASL
- ASL.2
- ASR
- ASR.2
- LSR
- LSR.2
- SASL
- SASL.2
- ASRR
- ASRR.2
- XASL
- XLSR
- XASRR.2
- ROL
- ROL.2
- SLAC.L
- SLSU.L
- SAVSL
- SAVSR



Shift/Rotate Operations

ASL

Shift/Rotate Shift Left

Shift/Rotate Operations

ASL

General Form **ASL** (.type) rd, rs /* (F) */
 ASL (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1		rs2/const5		rs1		rd		z		creg			

ASL

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	00110
rs	sint	.A	
rd	sint	.A	00111
rs	ucst4	.A	

ASL

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	010 00000
rs1	sint	.A	
rs2	sint		
rd	sint	.A	010 00001
rs1	sint	.A	
rs2	ucst5		
rd	slong	.A	010 00010
rs1	slong	.A	
rs2	sint		
rd	slong	.A	010 00011
rs1	slong	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 **rs1** 中的有符号数向左移位，末尾添 0，移位结果存放在 **rd** 中，移位的位数由寄存器 **rs2/cst5** 决定。

如果 **rs1** 的数据类型为 40 位有符号数，由于移位的位数最大为 40，所以移位位数 **rs2** 只取最低有效的 6 位，且为无符号数，其余位将被忽略，如果 $39 < \text{rs1} < 63$ ，则移位位数取 40。

如果 **rs1** 的数据类型为 32 位有符号数，移位位数 **rs2** 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。

如果移位位数为无符号立即数 **cst5**，**cst5** 在 0 到 31 之间。

Operation

If (cond)

rd = rs1 << rs2/cst5

Else nop

Execute time

B. cycle



Shift/Rotate Operations

ASL.2

Packed 16-Bit Shift/Rotate Shift Left

Shift/Rotate Operations

ASL.2

General Form **ASL.2** (.type) rd, rs /* (F) */
ASL.2 (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1		rs/cst4		rd		

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z	creg				

ASL.2

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	01000
rs	sint	.A	
rd	2hws	.A	01001
rs	ucst4	.A	

ASL.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	010 01100
rs1	2hws	.A	
rs2	sint		
rd	2hws	.A	010 01101
rs1	2hws	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 **rs1** 中的高 16 位有符号数以及低 16 位有符号数向左移位，末尾添 0，移位结果分别存放在 **rd** 的高 16 位和低 16 位中，移位的位数由寄存器 **rs2/cst5** 决定。移位位数 **rs2** 只取最低有效的 4 位，且为无符号数，其余位将被忽略，移位位数取最大为 15。如果移位位数为无符号立即数 **cst5**，只取 **cst5** 的低 4 位，移位位数在 0 到 15 之间。

Operation

If (cond)

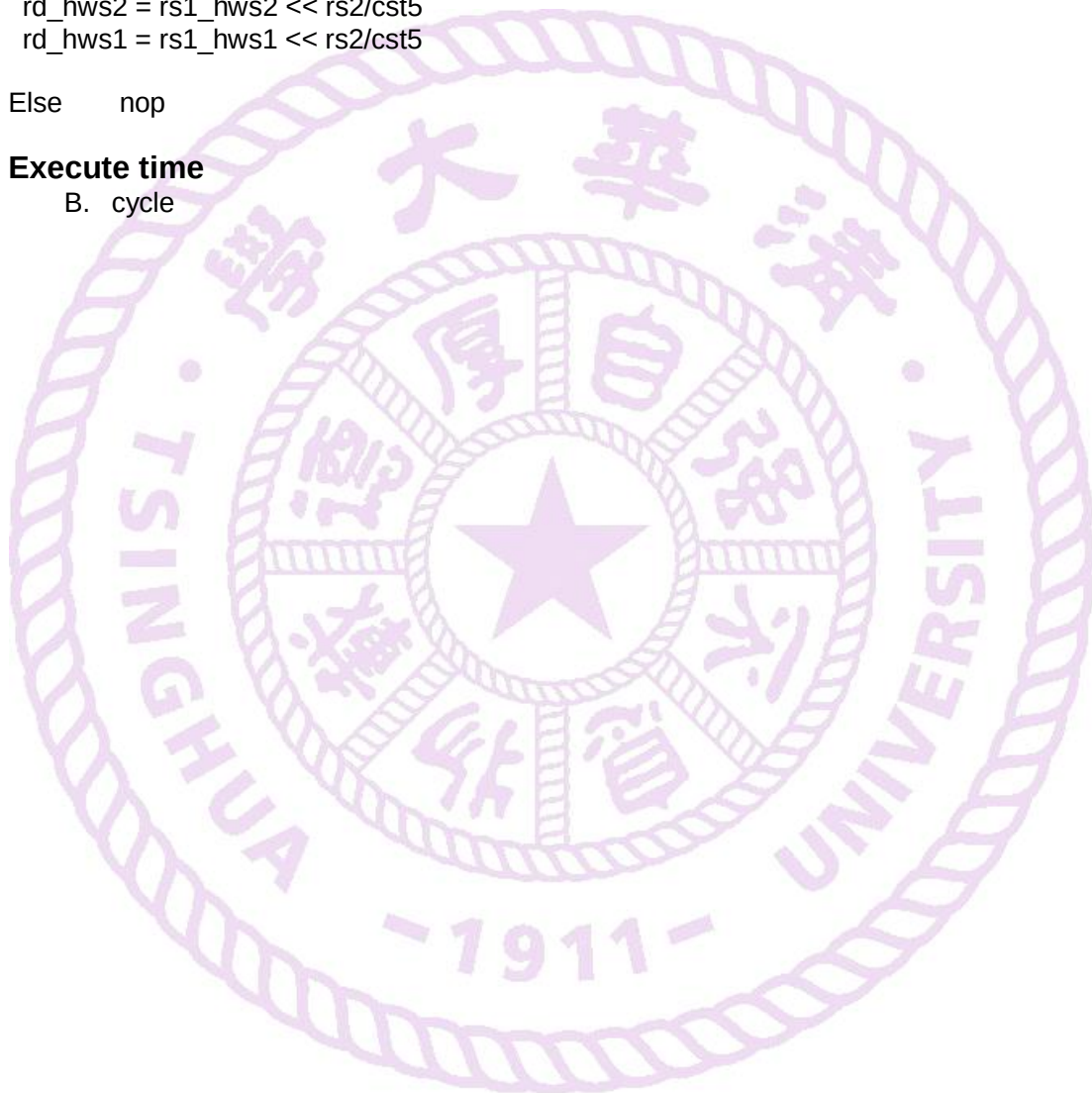
$rd\_hws2 = rs1\_hws2 \ll rs2/cst5$

$rd\_hws1 = rs1\_hws1 \ll rs2/cst5$

Else nop

Execute time

B. cycle



Shift/Rotate Operations

ASR

Shift/Rotate Shift Right

General Form

ASR (.type) rd, rs

/* (F) */

ASR (.type) rd, rs1, rs2

/* (D) */

.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1		rs/cst4		rd		

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z	creg				

ASR

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	01110
rs	sint	.A	
rd	sint	.A	01111
rs	ucst4	.A	

ASR

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	010 00100
rs1	sint	.A	
rs2	sint		
rd	sint	.A	010 00101
rs1	sint	.A	
rs2	ucst5		
rd	slong	.A	010 00110
rs1	slong	.A	
rs2	sint		
rd	slong	.A	000 00111
rs1	slong	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 **rs1** 中的有符号数向右算术移位，移位结果存放在 **rd** 中，移位的位数由寄存器 **rs2/cst5** 决定。

如果 **rs1** 的数据类型为 40 位有符号数，由于移位的位数最大为 40，所以移位位数 **rs2** 只取最低有效的 6 位，且为无符号数，其余位将被忽略，如果 $39 < rs1 < 63$ ，则移位位数取 40。

如果 **rs1** 的数据类型为 32 位有符号数，移位位数 **rs2** 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。

如果移位位数为无符号立即数 **cst5**，**cst5** 在 0 到 31 之间。

Operation

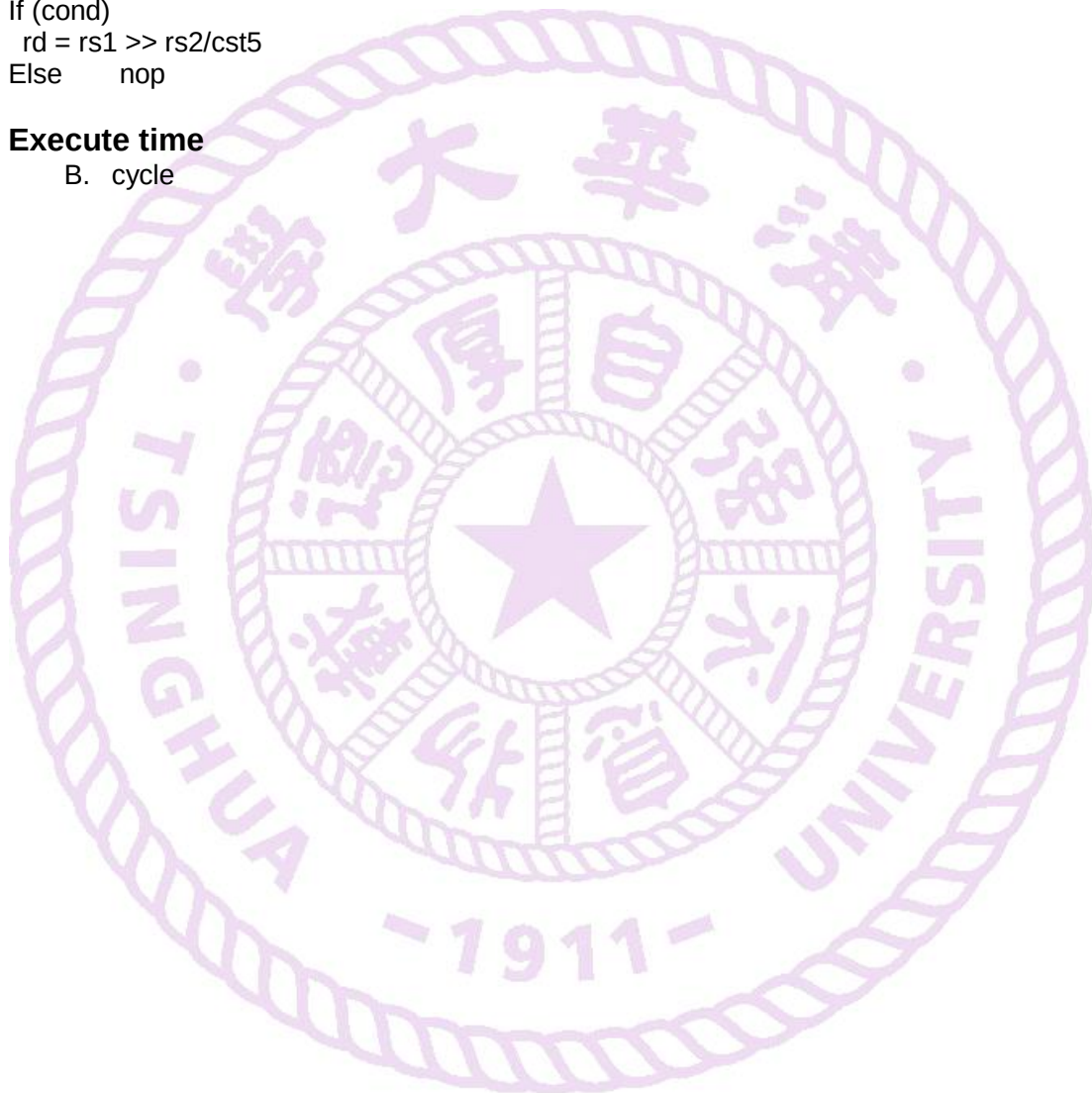
If (cond)

rd = rs1 >> rs2/cst5

Else nop

Execute time

B. cycle



Shift/Rotate Operations

ASR.2

Packed 16-Bit Shift/Rotate Shift Right

Shift/Rotate Operations

ASR.2

General Form **ASR.2** (.type) rd, rs /* (F) */
ASR.2 (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1		rs2/const5		rs1		rd		z	creg			

ASR.2

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	10000
rs	sint	.A	
rd	2hws	.A	10001
rs	ucst4	.A	

ASR.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	010 01110
rs1	2hws	.A	
rs2	sint		
rd	2hws	.A	010 01111
rs1	2hws	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 rs1 中的高 16 位有符号数以及低 16 位有符号数向右进行算术移位，移位结果分别存放在 rd 的高 16 位和低 16 位中，移位的位数由寄存器 rs2/cst5 决定。移位位数 rs2 只取最低有效的 4 位，且为无符号数，其余位将被忽略，移位位数取最大为 15。如果移位位数为无符号立即数 cst5，只取 cst5 的低 4 位，移位位数在 0 到 15 之间。

Operation

If (cond)

rd_hws2 = rs1_hws2 << rs2/cst5

rd_hws1 = rs1_hws1 << rs2/cst5

Else nop

Execute time

B. cycle



Shift/Rotate Operations

LSR

Logical Shift Right

General Form **LSR** (.type) rd, rs /* (F) */
LSR (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op	1	rs/cst4		rd		

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op	1	rs2/const5		rs1		rd	z	creg					

LSR

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	01010
rs	uint	.A	
rd	sint	.A	01011
rs	ucst4	.A	

LSR

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	010 01000
rs1	sint	.A	
rs2	uint		
rd	sint	.A	010 01001
rs1	sint	.A	
rs2	ucst5		
rd	slong	.A	010 01010
rs1	slong	.A	
rs2	uint		
rd	slong	.A	010 01011
rs1	slong	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 **rs1** 中的有符号数向右逻辑移位，移位结果存放在 **rd** 中，移位的位数由寄存器 **rs2/cst5** 决定。

如果 **rs1** 的数据类型为 40 位有符号数，由于移位的位数最大为 40，所以移位位数 **rs2** 只取最低有效的 6 位，且为无符号数，其余位将被忽略，如果 $39 < rs1 < 63$ ，则移位位数取 40。

如果 **rs1** 的数据类型为 32 位有符号数，移位位数 **rs2** 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。

如果移位位数为无符号立即数 **cst5**，**cst5** 在 0 到 31 之间。

Operation

If (cond)

rd = rs1 >> rs2/cst5

Else nop

Execute time

B. cycle



Shift/Rotate Operations

LSR.2

Packed 16-Bit Logical Shift Right

Shift/Rotate Operations

LSR.2

General Form **LSR.2** (.type) rd, rs /* (F) */
LSR.2 (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1		rs/cst4		rd		

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z	creg				

LSR.2

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	01100
rs	uint	.A	
rd	2hws	.A	01101
rs	ucst4	.A	

LSR.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	010 10000
rs1	2hws	.A	
rs2	uint		
rd	2hws	.A	010 10001
rs1	2hws	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 rs1 中的高 16 位有符号数以及低 16 位有符号数向右进行逻辑移位，移位结果分别存放在 rd 的高 16 位和低 16 位中，移位的位数由寄存器 rs2/cst5 决定。移位位数 rs2 只取最低有效的 4 位，且为无符号数，其余位将被忽略，移位位数取最大为 15。如果移位位数为无符号立即数 cst5，只取 cst5 的低 4 位，移位位数在 0 到 15 之间。

Operation

If (cond)

rd_hws2 = rs1_hws2 << rs2/cst5

rd_hws1 = rs1_hws1 << rs2/cst5

Else nop

Execute time

B. cycle



Shift/Rotate Operations

SASL

Shift/Rotate Shift Left with Saturation

Shift/Rotate Operations

SASL

General Form SASL (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SASL

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	010 10010
rs1	sint	.A	
rs2	uint		
rd	sint	.A	010 10011
rs1	sint	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 rs1 中的有符号数向左移位，末尾添 0，移位结果经过饱和处理后存放在 rd 中，移位的位数由寄存器 rs2/cst5 决定。Rs1 的数据类型为 32 位有符号数，移位位数 rs2 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。如果移位位数为无符号立即数 cst5，cst5 在 0 到 31 之间。

Operation

(D) If (cond) rd <= sat (rs1 << rs2/ucst5);
 Else nop

Execute time

B. cycle

Shift/Rotate Operations

SASL.2

Packed 16-Bit Shift/Rotate Shift Left
with Saturation

Shift/Rotate Operations

SASL.2

General Form SASL.2 (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SASL.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.A	010 00100
rs1	2hws	.A	
rs2	uint		
rd	2hws	.A	010 00101
rs1	2hws	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器 rs1 中的高 16 位有符号数以及低 16 位有符号数向左移位，末尾添 0，移位结果经过饱和处理后分别存放在 rd 的高 16 位和低 16 位中，移位的位数由寄存器 rs2/cst5 决定。移位位数 rs2 只取最低有效的 4 位，且为无符号数，其余位将被忽略，移位位数取最大为 15。如果移位位数为无符号立即数 cst5，只取 cst5 的低 4 位，移位位数在 0 到 15 之间。

Operation

If (cond)

```
{ rd_hw2s <= sat (rs1_hw2s << rs2/ucst5);
  rd_hw1s <= sat (rs1_hw1s << rs2/ucst5);
```

Else nop

Execute time

B. cycle

Shift/Rotate Operations

ASRR

Shift/Rotate Shift Right with Round

Shift/Rotate Operations

ASRR

General Form

ASRR (.type) rd, rs

/* (F) */

ASRR (.type) rd, rs1, rs2

/* (D) */

.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op_1				rs/cst4		rd

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op		Op_1			rs2/const5			rs1		rd		z	creg	

ASRR

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	10010
rs	uint	.A	
rd	sint	.A	10011
rs	ucst4	.A	

ASRR

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	010 10110
rs1	sint	.A	
rs2	uint		
rd	sint	.A	010 10111
rs1	sint	.A	
rs2	ucst5		

Description

(F)将寄存器 rd 中的有符号数向右进行带补偿移位，移位结果存放在 rd 中，移位的位数由寄存器 rs/cst4 决定，rs1 的数据类型为 32 位有符号数，移位位数 rs2 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。如果移位位数为无符号立即数 cst4，cst5 在 0 到 15 之间。

(D)将寄存器 rs1 中的有符号数向右进行带补偿移位，移位结果存放在 rd 中，移位的位数由

寄存器 `rs2/cst5` 决定。`Rs1` 的数据类型为 32 位有符号数，移位位数 `rs2` 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。如果移位位数为无符号立即数 `cst5`，`cst5` 在 0 到 31 之间。

Operation

- (F) `round = 1 << (rs2 - 1);`
`rd = (rd + round) >> rs;`
- (D) `if (cond)`
`round = 1 << (rs2 - 1);`
`rd = (rs1 + round) >> rs2;`

Execute time

- B. cycle



Shift/Rotate Operations

ASRR.2

Packed 16-Bit Shift/Rotate Shift Right
with Round

Shift/Rotate Operations

ASRR.2

General Form **ASRR.2** (.type) rd, rs /* (F) */
 ASRR.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

(F) Format

AL.F1

15	14	13	12	8	7	4	3	0
s	00	Op	1	rs/cst4		rd		

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op	1	rs2/const5		rs1		rd		z	creg				

ASRR.2

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	hw2s	.A	10100
rs	uint	.A	
rd	hw2s	.A	10101
rs	ucst4	.A	

ASRR.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	hw2s	.A	010 11000
rs1	hw2s	.A	
rs2	uint		
rd	hw2s	.A	010 11001
rs1	hw2s	.A	
rs2	ucst5		

Description

(F)将寄存器 rd 中的两个 16 位有符号数向右进行带补偿移位，移位结果存放在 rd 中，移位的位数由寄存器 rs/cst4 决定。移位位数 rs2 只取最低有效的 4 位，且为无符号数，其余位将被忽略，移位位数取最大为 15。如果移位位数为无符号立即数 cst5，只取 cst5 的低 4 位，移位位数在 0 到 15 之间。

(D)将寄存器 rs1 中的两个 16 位有符号数向右进行带补偿移位，移位结果存放在 rd 中，

移位的位数由寄存器 `rs2/cst5` 决定。移位位数 `rs2` 只取最低有效的 4 位，且为无符号数，其余位将被忽略，移位位数取最大为 15。如果移位位数为无符号立即数 `cst5`，只取 `cst5` 的低 4 位，移位位数在 0 到 15 之间。

Operation

```
round = 1 << (rs2 - 1);  
rd = (rs1 + round) >> rs2;
```

Execute time

B. cycle



Shift/Rotate Operations

XASL

64bit Bitstream Shift Left

Shift/Rotate Operations

XASL

General Form **XASL** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

XASL

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	110	00000
rs1	sllong	.A		
rs2	uint			
rd	sint	.A	110	00001
rs1	sllong	.A		
rs2	ucst5			

Description

这条指令的作用是将寄存器对 rs1+1: rs1 中的 64 位码流向左移位，末尾添 0，移位结果的前 32 位存放在 rd 中。移位位数 rs2 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。如果移位位数为无符号立即数 cst5，cst5 在 0 到 31 之间。

Operation

(D) If (cond) rd <= (rs1+1:rs1 << rs2/ucst5) [63:32];

Else nop

Execute time

B. cycle

Shift/Rotate Operations

XASR

64bit Bitstream Shift Right

General Form **XASR** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

XASR

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	110 00010
rs1	sllong	.A	
rs2	uint		
rd	sint	.A	110 00011
rs1	sllong	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器对 rs1+1: rs1 中的 64 位码流向右移位，首位添 0，移位结果的后 32 位存放在 rd 中。移位位数 rs2 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。如果移位位数为无符号立即数 cst5，cst5 在 0 到 31 之间。

Operation

(D) If (cond) rd <= (rs1+1:rs1 >> rs2/ucst5) [31:0];

Else nop

Execute time

B. cycle

Shift/Rotate Operations

XASRR.2

Two words Shift/Rotate Shift Right with
Round

Shift/Rotate Operations

XASRR.2

General Form XASRR.2 (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

XASRR.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	sint	.A	110	00100
rs1	sllong	.A		
rs2	uint			
Rd	sint	.A	110	00101
rs1	sllong	.A		
rs2	ucst5			

Description

这条指令的作用是将寄存器对 rs1+1: rs1 中的两个 32 位寄存器中的值，将寄存器 rs1+1，和寄存器 rs1 中的两个 32 位有符号数向右进行带补偿移位，移位结果存放在 rd 的高 16 位和低 16 位中，移位的位数由寄存器 rs2/cst5 决定。移位位数 rs2 只取最低有效的 5 位，且为无符号数，其余位将被忽略，移位位数取最大为 31。如果移位位数为无符号立即数 cst5，cst5 在 0 到 31 之间。

Operation

(D) If (cond) rd_smsb16 <= rs1+1_sint + round >> rs2/ucst5;
rd_slbsb16 <= rs1_sint + round >> rs2/ucst5;
Else nop

Execute time

B. cycle

Shift/Rotate Operations

ROL

Rotate Left

ROL

General Form **ROL (.type) rd, rs1, rs2** /* (D) */
.type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd			z	creg	

ROL

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	011	01000
rs1	sint	.A		
rs2	uint			
rd	sint	.A	011	01001
rs1	sint	.A		
rs2	ucst5			

Description

这条指令的作用是将寄存器对 rs1 中的 32 位数向左循环移位，移位结果存放在 rd 中。

Operation

(D) If (cond) $rd\_uint = (rs1\_uint \ll rs2\_sint) | (rs1\_uint \ll (32 - rs2\_sint))$

```
Else      nop
```

Execute time

B. cycle

Shift/Rotate Operations

ROL.2

Two 16-Bit Rotate Left

Shift/Rotate Operations

ROL.2

General Form **ROL.2** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

ROL.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 01010
rs1	sint	.A	
rs2	uint		
rd	sint	.A	011 01011
rs1	sint	.A	
rs2	ucst5		

Description

这条指令的作用是将寄存器对 rs1 中的两个 16 位数向左循环移位，移位结果存放在 rd 中。

移位位数 rs2 只取最低有效的 4 位，且为无符号数，其余位将被忽略，移位位数取最大为 15。如果移位位数为无符号立即数 cst5，只取 cst5 的低 4 位，移位位数在 0 到 15 之间。

Operation

If (cond)

```
{
    rd_hw2u = (rs1_hw2u<<rs2_sint)|(rs1_hw2u<<(32-rs2_sint));
    rd_hw1u = (rs1_hw1u<<rs2_sint)|(rs1_hw1u<<(32-rs2_sint));
}
```

Else nop

Execute time

B. cycle

Shift/Rotate Operations

SAVSL

Variable Shift Left with Saturation

Shift/Rotate Operations

SAVSL

General Form **SAVSL** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SAVSL

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	110 01010
rs1	sint	.A	
rs2	uint		

Description

这条指令的作用是将寄存器 rs1 中的有符号数进行移位，移位的位数由寄存器 rs2 决定，由于移位的位数的范围为-32~31，所以有符号数 rs2 只取最低有效的 6 位，第 6 位为符号位，其余位将被忽略。如果 rs2 的低 6 位大于 0，则 rs1 中的有符号数向左移位，末尾添 0，移位结果经过饱和处理后存放在 rd 中；如果 rs2 的低 6 位为负数，则 rs1 中的有符号数向右移位，移位的位数为 rs2 低 6 位有符号数的绝对值。

Operation

(D) If (cond)

 If(rs2[5:0] > 0) rd = sat (rs1 << rs2);

 else rd = rs1 >> (-rs2[5:0])

Else nop

Execute time

B. cycle

Shift/Rotate Operations

SAVSR

Variable Right Left with Saturation

SAVSR

General Form **SAVSR** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd		z		creg	

SAVSR

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	110 01011
rs1	sint	.A	
rs2	uint		

Description

这条指令的作用是将寄存器 **rs1** 中的有符号数进行移位，移位的位数由寄存器 **rs2** 决定，由于移位的位数的范围为-32~31，所以有符号数 **rs2** 只取最低有效的 6 位，第 6 位为符号位，其余位将被忽略。如果 **rs2** 的低 6 位为负数，则 **rs1** 中的有符号数向右移位，移位的位数为 **rs2** 低 6 位有符号数；如果 **rs2** 的低 6 位大于 0，则 **rs1** 中的有符号数向左移位，移位的位数为 **rs2** 低 6 位有符号数的绝对值，末尾添 0，移位结果经过饱和处理后存放在 **rd** 中。

Operation

(E) If (cond)

```
if(rs2[5:0] < 0 ) rd = sat (rs1 << (-rs2[5:0]));
```

```
else rd = rs1 >> rs2[5:0] ;
```

```
Else    nop
```

Execute time

B. cycle

9 MULTIPLY OPERATIONS

Instruction Summary

- MUL.S/.U
- MUL.S/.US/.U/.SU.H
- MUL.S/.US/.U/.SU.L
- MUL.S/.SU/.U/.US.HL
- MUL.S/.SU/.U/.US.LH
- MUL.S/U.HI
- MUL.S/U.LI
- MUL.S/.US/.U.2
- MULR.HI
- MULR.LI
- MULR.L
- MULR.2
- SMULL.H
- SMULL.L
- SMULL.HL
- SMULL.LH
- SMULL.2
- MAC.L
- MAC.H
- SMACL.L
- SMACL.H
- MSU.L
- MSU.H
- SMSUL.L
- SMSUL.H
- DTP.S/.SU.2
- DTPN.S/.SU.2
- DTPR.S/.SU.2
- BTFDTP.2
- BTFDTPR.2

Multiply Operations

MUL.S/.U Signed or Unsigned Integer Multiply

Multiply Operations

MUL.S/.U

General Form **MUL.S** (.type) rd, rs1, rs2 /* (D) */
 MUL.U (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MUL.S

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sllong	.M	001	11000
rs1	sint	.M		
rs2	sint			
rd	sllong	.M	001	11001
rs1	sint	.M		
rs2	scst5			

MUL.U

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	ullong	.M	001	11010
rs1	uint	.M		
rs2	uint			
rd	sllong	.M	001	11011
rs1	uint	.M		
rs2	scst5			

Description

这条指令的作用是将寄存器 rs1 与寄存器 rs2 中的有符号或无符号 32 位整数相乘，结果的高 32 位存放于寄存器 rd+1 中，低 32 位存放在寄存器 rd 中。

Operation

(D) If (cond) rd+1:rd <= rs1_sint x rs2_sint;

Else nop

Execute time

2 cycles

Multiply Operations MUL.S/.US/.U/.SU.H

Signed or Unsigned Integer Multiply 16msb x 16msb

Multiply Operations MUL.S/.US/.U/.SU.H

General Form

MUL.S.H (.type) rd, rs /* (F) */
 MUL.US.H (.type) rd, rs /* (F) */
 MUL.U.H (.type) rd, rs /* (F) */
 MUL.SU.H (.type) rd, rs /* (F) */
 MUL.S.H (.type) rd, rs1, rs2 /* (D) */
 MUL.US.H (.type) rd, rs1, rs2 /* (D) */
 MUL.U.H (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1	rs/cst4	rd				

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MUL.S.H

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	smsb16/sint	.M	00000
rs	smsb16	.M	
rd	smsb16/sint	.M	00001
rs	scst4	.M	

MUL.US.H

(F) Format

rd	umsb16/uint	.M	00010
rs	smsb16	.M	
rd	umsb16/uint	.M	00011
rs	scst4	.M	

MUL.U.H

(F) Format

rd	umsb16/uint	.M	00100
rs	umsb16	.M	

MUL.SU.H	(F) Format			
rd	smsb16/sint	.M	00101	
rs	umsb16	.M		
MUL.S.H	(D) Format			
rd	sint	.M	000	00000
rs1	smsb16	.M		
rs2	smsb16			
rd	sint	.M	000	00001
rs1	smsb16	.M		
rs2	scst5			
MUL.US.H	(D) Format			
rd	uint	.M	000	00010
rs1	umsb16	.M		
rs2	smsb16			
rd	uint	.M	000	00011
rs1	umsb16	.M		
rs2	scst5			
MUL.U.H	(D) Format			
rd	uint	.M	000	00100
rs1	umsb16	.M		
rs2	umsb16			

Description

(F)这条指令的作用是将寄存器 rd 与寄存器 rs 中的有高 16 位符号数相乘，结果存放于寄存器 rd 中

(D)这条指令的作用是将寄存器 rs1 与寄存器 rs2 中的有高 16 位符号数相乘，结果存放于寄存器 rd 中

Operation

(F) $rd\_ints/u \leq rd\_msb16s/u \times rs\_msb16s/u;$

(D) If (cond) $rd\_ints/u \leq rs1\_msb16s/u \times rs2\_msb16s/u;$

Else nop

Execute time

2 cycle

Multiply Operations MUL.S/.US/.U/.SU.L

Multiply Operations MUL.S/.US/.U/.SU.L

Signed or Unsigned Integer Multiply lsb16 x lsb16

General Form

MUL.S.L (.type) rd, rs /* (F) */
MUL.US.L (.type) rd, rs /* (F) */
MUL.U.L (.type) rd, rs /* (F) */
MUL.SU.L (.type) rd, rs /* (F) */
MUL.S.L (.type) rd, rs1, rs2 /* (D) */
MUL.US.L (.type) rd, rs1, rs2 /* (D) */
MUL.U.L (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1



(D) Format

MA.D1



MUL.S.L

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	slsb16/sint	.M	00110
rs	slsb16	.M	
rd	slsb16/sint	.M	00111
rs	scst4	.M	

MUL.US.L

(F) Format

rd	ulsb16/uint	.M	01000
rs	slsb16	.M	
rd	ulsb16/uint	.M	01001
rs	scst4	.M	

MUL.U.L

(F) Format

rd	ulsb16/uint	.M	01010
rs	ulsb16	.M	

MUL.SU.L	(F) Format			
rd	slsb16/sint	.M	01011	
rs	ulsb16	.M		
MUL.S.L	(D) Format			
rd	sint	.M	000	00101
rs1	slsb16	.M		
rs2	slsb16			
rd	sint	.M	000	00110
rs1	slsb16	.M		
rs2	scst5			
MUL.US.L	(D) Format			
rd	uint	.M	000	00111
rs1	ulsb16	.M		
rs2	slsb16			
rd	uint	.M	000	01000
rs1	ulsb16	.M		
rs2	scst5			
MUL.U.L	(D) Format			
rd	uint	.M	000	01001
rs1	ulsb16	.M		
rs2	ulsb16			

Description

(F)这条指令的作用是将寄存器 rd 与寄存器 rs 中的有低 16 位符号数相乘，结果存放于寄存器 rd 中

(D)这条指令的作用是将寄存器 rs1 与寄存器 rs2 中的有低 16 位符号数相乘，结果存放于寄存器 rd 中

Operation

(F) $rd\_ints/u \leq rd\_lsb16s/u \times rs\_lsb16s/u;$

(D) If (cond) $rd\_ints/u \leq rs1\_lsb16s/u \times rs2\_lsb16s/u;$

Else nop

Execute time

2 cycles

Multiply Operations MUL.S/.U/.SU/.US.HL

Signed or Unsigned Integer Multiply msb16 x lsb16

Multiply Operations MUL.S/.U/.SU/.US.HL

General Form

MUL.S.HL (.type) rd, rs /* (F) */
 MUL.U.HL (.type) rd, rs /* (F) */
 MUL.SU.HL (.type) rd, rs /* (F) */
 MUL.US.HL (.type) rd, rs /* (F) */
 MUL.S.HL (.type) rd, rs1, rs2 /* (D) */
 MUL.US.HL (.type) rd, rs1, rs2 /* (D) */
 MUL.U.HL (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1	rs/cst4	rd				

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MUL.S.HL

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	smsb16/sint	.M	01100
rs	slsb16	.M	

MUL.U.HL

(F) Format

rd	umsb16/uint	.M	01101
rs	ulsb16	.M	

MUL.SU.HL

(F) Format

rd	smsb16/sint	.M	01110
rs	ulsb16	.M	

MUL.US.HL

(F) Format

rd	umsb16/uint	.M	01111
rs	slsb16	.M	

MUL.S.HL	(D) Format			
rd	sint	.M	000	01010
rs1	smsb16	.M		
rs2	slsb16			

MUL.U.HL	(D) Format			
rd	uint	.M	000	01011
rs1	umsb16	.M		
rs2	ulsb16			

MUL.SU.HL	(D) Format			
rd	sint	.M	000	01100
rs1	smsb16	.M		
rs2	ulsb16			

MUL.US.HL	(D) Format			
rd	uint	.M	000	01101
rs1	umsb16	.M		
rs2	slsb16			

Description

(F)将寄存器 rd 的高 16 位有符号数与寄存器 rs 中的低 16 位有符号数相乘，结果存放于寄存器 rd 中

(D)将寄存器 rs1 的高 16 位有符号数与寄存器 rs2 中的低 16 位有符号数相乘，结果存放于寄存器 rd 中

Operation

(F) $rd\_ints/u \leq rd\_msb16s/u \times rs\_lsb16s/u;$

(D) If (cond) $rd\_ints/u \leq rs1\_msb16s/u \times rs2\_lsb16s/u;$

Else nop

Execute time

2 cycles

Multiply Operations MUL.S/.U/.SU/.US.LH

Signed or Unsigned Integer Multiply lsb16 x msb16

Multiply Operations MUL.S/.U/.SU/.US.LH

General Form **MUL.S.LH** (.type) rd, rs /* (F) */
 MUL.U.LH (.type) rd, rs /* (F) */
 MUL.SU.LH (.type) rd, rs /* (F) */
 MUL.US.LH (.type) rd, rs /* (F) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1				rs/cst4		rd

MUL.S.LH

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sldb16/sint	.M	10000
rs	smsb16	.M	

MUL.U.LH

(F) Format

rd	uldb16/uint	.M	10001
rs	umsb16	.M	

MUL.SU.LH

(F) Format

rd	sldb16/sint	.M	10010
rs	umsb16	.M	

MUL.US.LH

(F) Format

rd	uldb16/uint	.M	10011
rs	smsb16	.M	

Description

(F)将寄存器 rd 的低 16 位有符号数与寄存器 rs 中的高 16 位有符号数相乘，结果存放于寄存器 rd 中

(D)将寄存器 rs1 的低 16 位有符号数与寄存器 rs2 中的高 16 位有符号数相乘，结果存放于寄存器 rd 中

Operation

(F) $rd\_ints/u \leq rd\_lsb16s/u \times rs\_msb16s/u;$

(D) If (cond) $rd\_ints/u \leq rs1\_lsb16s/u \times rs2\_msb16s/u;$

Else nop

Execute time

2 cycles



Multiply Operations

MUL.S/.U.HI

Signed or Unsigned Integer Multiply
16msb x 32bit

Multiply Operations

MUL.S/.U.HI

General Form **MUL.S.HI** (.type) rd, rs1, rs2 /* (D) */
 MUL.U.HI (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op	1	rs2/const5	rs1	rd	z	creg							

MUL.S.HI

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sllong	.M	000 01110
rs1	smsb16	.M	
rs2	sint		

MUL.U.HI

(D) Format

rd	ullong	.M	000 01111
rs1	umsb16	.M	
rs2	uint		

Description

这条指令的作用是将寄存器 rs1 的高 16 位有符号数(或无符号数)与寄存器 rs2 中 32 位有符号数(或无符号数)相乘, 结果存放于寄存器 rd 中。

Operation

(F) rd+1:rd <= rd_msb16s/u x rs_ints/u;
(D) If (cond) rd+1:rd <= rs1_msb16s/u x rs2_ints/u;
 Else nop

Execute time

2 cycles

Multiply Operations

MUL.S/.U.LI

Signed or Unsigned Integer Multiply
16lsb x 32bit

Multiply Operations

MUL.S/.U.LI

General Form **MUL.S.LI** (.type) rd, rs1, rs2 /* (D) */
 MUL.U.LI (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MUL.S.LI

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sllong	.M	000 10000
rs1	slsb16	.M	
rs2	sint		

MUL.U.LI

(D) Format

rd	ullong	.M	000 10001
rs1	ulsb16	.M	
rs2	uint		

Description

这条指令的作用是将寄存器 rs1 的低 16 位有符号数(或无符号数)与寄存器 rs2 中 32 位有符号数(或无符号数)相乘，结果存放于寄存器 rd 中。

Operation

(F) rd+1:rd <= rd_lsb16s/u x rs_ints/u;
 (D) If (cond) rd+1:rd <= rs1_lsb16s/u x rs2_ints/u;
 Else nop

Execute time

2 cycles

Multiply Operations

MUL.S/.U/.US.2

Packed 16-Bit Signed or Unsigned Integer Multiply

Multiply Operations

MUL.S/.U/.US.2

General Form **MUL.S.2** (.type) rd, rs1, rs2 /* (D) */
MUL.US.2 (.type) rd, rs1, rs2 /* (D) */
MUL.U.2 (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MUL.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sllong	.M	000	11001
rs1	2hws	.M		
rs2	2hws			
rd	sllong	.M	000	11010
rs1	2hws	.M		
rs2	scst5			

MUL.U.2

(D) Format

rd	ullong	.M	000	11101
rs1	2hwu	.M		
rs2	2hwu			

MUL.US.2

(D) Format

rd	ullong	.M	000	11011
rs1	2hwu	.M		
rs2	2hws			
rd	ullong	.M	000	11100
rs1	2hwu	.M		
rs2	scst5			

Description

(D)将寄存器 `rs2` 中的低 16 位有符号数与寄存器 `rs1` 中的低 16 位有符号数相乘，结果存放于寄存器 `rd` 中；寄存器 `rs2` 中的高 16 位有符号数与寄存器 `rs1` 中的高 16 位有符号数相乘，结果存放于寄存器 `rd+1` 中。

Operation

```
(D) If (cond)
{   rd <= rs2_hw2s * rs1_hw2s
  rd+1 <= rs2_hw1s * rs1_hw1s
}
```

Elsenop

Execute time

2 cycles



Multiply Operations

MULR.HI Signed or Unsigned Integer Multiply 16msb x 32bit,
Shifted by 15 to Produce a Rounded 32-Bit Result

Multiply Operations

MULR.HI

General Form **MULR.HI** (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MULR.HI

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.M	000 10010
rs1	smsb16	.M	
rs2	sint		

Description

这条指令的作用是将寄存器 rs1 的高 16 位有符号数与寄存器 rs2 中 32 位有符号数相乘, 结果经过带补偿的 15 位移位之后存放于寄存器 rd 中。

Operation

(D) If (cond) rd <= (rs1_msb16s x rs2_ints + 0x4000) >> 15;

Else nop

Execute time

2 cycles

Multiply Operations

MULR.LI

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5		rs1		rd		z	creg		

Description

这条指令的作用是将寄存器 **rs1** 的低 16 位有符号数与寄存器 **rs2** 中 32 位有符号数相乘, 结果经过带补偿的 15 位移位之后存放于寄存器 **rd** 中。

```
(D) If (cond) rd <= (rs1_lsb16s x rs2_ints + 0x4000) >> 15;
Else      nop
```

2 cycles

Multiply Operations

Multiply Operations

MULR.L Signed or Unsigned 16-Bit Integer Multiply, Shifted by 15 to Product a Rounded Result **MULR.L**

General Form **MULR.L** (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MULR.L

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.M	001 10111
rs1	2hws	.M	
rs2	2hws		

Description

(D)将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘，结果经过带补偿的 15 位移位之后存放于寄存器 rd 的低 16 位中。

Operation

(D)If (cond)

```
{ rd_hw1s <= (rs2_hw1s * rs1_hw1s + round) >> 15;
}
```

Elsenop

Execute time

2 cycles

Multiply Operations

Multiply Operations

MULR.2 Packed 16-Bit Signed or Unsigned Integer Multiply,
Shifted by 15 to Product a Rounded Result

MULR.2

General Form **MULR.2** (.type) rd, rs /* (F) */
 MULR.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01		Op_1		rs/cst4		rd	

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op_1		rs2/const5		rs1		rd		z		creg		

MULR.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.M	000 11110
rs1	2hws	.M	
rs2	2hws		

Description

(F)将寄存器 rs 中的低 16 位有符号数与寄存器 rd 中的低 16 位有符号数相乘，结果经过带补偿的 15 位移位之后存放于寄存器 rd 的低 16 位中；寄存器 rs 中的高 16 位有符号数与寄存器 rd 中的高 16 位有符号数相乘，结果经过带补偿的 15 位移位之后存放于寄存器 rd 的高 16 位中。

(D)将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘，结果经过带补偿的 15 位移位之后存放于寄存器 rd 的低 16 位中；寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，结果经过带补偿的 15 位移位之后存放于寄存器 rd 的高 16 位中。

Operation

```
B. rd_hw2s <= (rd_hw2s * rs_hw2s + round) >> 15
rd_hw1s <= (rd_hw1s * rs_hw1s + round) >> 15
(D) If (cond)
{   rd_hw2s <= (rs2_hw2s * rs1_hw2s + round) >> 15;
    rd_hw1s <= (rs2_hw1s * rs1_hw1s + round) >> 15
}
Elsenop
```

Execute time

2 cycles



Multiply Operations

SMULL.H

Signed or Unsigned Integer Multiply 16msb x 16msb, with Left Shift and Saturate

Multiply Operations

SMULL.H

General Form **SMULL.H** (.type) rd, rs /* (F) */
 SMULL.H (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1



(D) Format

MA.D1



SMULL.H

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	smsb/sint	.M	10100
rs	smsb	.M	

SMULL.H

(D) Format

rd	sint	.M	000	10100
rs1	smsb	.M		
rs2	smsb			
rd	sint	.M	000	10101
rs1	smsb	.M		
rs2	scst5			

Description

(F)将寄存器 rs 中的高 16 位有符号数与寄存器 rd 中的高 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 rd 中。

(D)将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，结果经过

左移一位和饱和处理后存放于寄存器 **rd** 中。

Operation

(F) $rd \leftarrow \text{sat}[(rd\_msb16s * rs\_msb16s) \ll 1]$

(D) If (cond) $rd \leftarrow \text{sat}[(rs1\_msb16s * rs2\_msb16s) \ll 1]$

Elsenop

Execute time

2 cycles



Multiply Operations

SMULL.L

Signed or Unsigned Integer Multiply 16lsb x 16lsb, with Left Shift and Saturate

Multiply Operations

SMULL.L

General Form **SMULL.L** (.type) rd, rs /* (F) */
 SMULL.L (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op 1				rs/cst4		rd

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op 1			rs2/const5			rs1		rd		z	creg	

SMULL.L

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	slsb/sint	.M	10101
rs	slsb	.M	

SMULL.L

(D) Format

rd	sint	.M	000	10110
rs1	slsb	.M		
rs2	slsb			
rd	sint	.M	000	10111
rs1	slsb	.M		
rs2	scst5			

Description

(F)将寄存器 rd 中的低 16 位有符号数与寄存器 rs 中的低 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 rd 中。

(D)将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 rd 中。

Operation

(F) $rd \leq \text{sat}[(rd\_lsb16s * rs\_lsb16s) \ll 1]$

(D) If (cond) $rd \leq \text{sat}[(rs1\_lsb16s * rs2\_lsb16s) \ll 1]$

Elsenop

Execute time

2 cycles



Multiply Operations

SMULL.HL

Signed or Unsigned Integer Multiply 16msb x 16lsb, with Left Shift and Saturate

Multiply Operations

SMULL.HL

General Form **SMULL.HL** (.type) rd, rs /* (F) */
SMULL.HL (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1				rs/cst4		rd

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op		Op_1			rs2/const5		rs1		rd		z	creg		

SMULL.HL

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	smsb/sint	.M	10110
rs	slsb	.M	

SMULL.HL

(D) Format

rd	sint	.M	000	11000
rs1	smsb	.M		
rs2	slsb			

Description

(F)将寄存器 rd 中的高 16 位有符号数与寄存器 rs 中的低 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 rd 中。

(D)将寄存器 rs1 中的高 16 位有符号数与寄存器 rs2 中的低 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 rd 中。

Operation

(F) $rd \leftarrow \text{sat}[(rd\_msb16s * rs\_lsb16s) \ll 1]$

(D) If (cond) $rd \leftarrow \text{sat}[(rs1\_msb16s * rs2\_lsb16s) \ll 1]$

Elsenop

Execute time

2 cycles



Multiply Operations

SMULL.LH

Signed or Unsigned Integer Multiply 16lsb x 16msb, with Left Shift and Saturate

SMULL.LH

General Form

SMULL.LH (.type) rd, rs
.type = .M

$$/* (F) */$$

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1				rs/cst4		rd

SMULL.LH

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	slsb/sint	.M	10111
rs	smsb	.M	

Description

这条指令的作用是将寄存器 **rd** 中的低 16 位有符号数与寄存器 **rs** 中的高 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 **rd** 中。

Operation

```
B. rd <= sat[(rd lsb16s * rs lsb16s) <<1]
```

Execute time

2 cycles

Multiply Operations

Multiply Operations

SMULL.2 Packed 16-Bit Signed or Unsigned Integer Multiply, with Left Shift and Saturate SMULL.2

General Form SMULL.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SMULL.2 (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.M	000 11111
rs1	2hws	.M	
rs2	2hws		

Description

(D)将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 rd 中。寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，结果经过左移一位和饱和处理后存放于寄存器 rd+1 中。

Operation

If (cond)

```
{
    rd+1 <= sat[(rs1_hw2s * rs2_hw2s) <<1];
    rd <= sat[(rs1_hw1s * rs2_hw1s) <<1]
}
```

Elsenop

Execute time

2 cycles

Multiply Operations

MAC.L

Signed Integer Multiply and Accumulation
16lsb x 16lsb

Multiply Operations

MAC.L

General Form **MAC.L** (.type) ACC0, rs1, rs2 /* (F) */
 MAC.L (.type) ACC0, rs1, rs2 /* (D) */
 MAC.L (.type) ACC1, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, ACC0, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1

15	14	13	12	8	7	4	3	0
s	01	Op_1				Rs1		Rs2

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2/const5				rs1		0000a		z	creg

a: ACC select

-- "0" ACC0 X12/Y12

-- "1" ACC1 X22/Y22

MAC.L

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rs1	slsb	.M	001	01101
rs2	slsb	.M		
rs1	slsb	.M	001	01110
rs2	scst5	.M		

Description

将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘, 结果与累加寄存器 ACC0 或 ACC1 中的 32 位有符号数相加后产生 32 位有符号数存放于相应的累加寄存器 ACC 中。

Operation

(F) $ACC \leq ACC + rs1\_lsb16s * rs2\_lsb16s$

(D) If (cond) $ACC \leq ACC + rs1\_lsb16s * rs2\_lsb16s$
Elsenop

Execute time

2 cycles



Multiply Operations

MAC.H

Signed Integer Multiply and Accumulation
16msb x 16msb

Multiply Operations

MAC.H

General Form **MAC.H** (.type) ACC0, rs1, rs2 /* (D) */
 MAC.H (.type) ACC1, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, ACC0, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	0000a	z	creg								

a: ACC select

-- "0" ACC0 X12 /Y12

-- "1" ACC1 X22 /Y22

MAC.H

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs1	smsb	.M	001 10011
rs2	smsb	.M	

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，结果与累加寄存器 ACC0 或 ACC1 中的 32 位有符号数相加后产生 32 位有符号数存放于累加寄存器 ACC 中。

Operation

(D) If (cond) ACC<= ACC + rs1_msb16s * rs2_msb16s
 Elsenop

Execute time

2 cycles

Multiply Operations

SMACL.L

Signed Integer Multiply and Accumulation
16lsb x 16lsb, with Left Shift and Saturate

Multiply Operations

SMACL.L

General Form **SMACL.L** (.type) ACC0, rs1, rs2 /* (D) */
SMACL.L (.type) ACC1, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, ACC0, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	0000a	z	creg								

a: ACC select

-- "0" ACC0 X12 /Y12

-- "1" ACC1 X22 /Y22

SMACL.L

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs1	slsb	.M	001 01111
rs2	slsb	.M	

Description

这条指令的作用是将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘，结果左移一位并做饱和处理后，与累加寄存器 ACC0 或 ACC1 中的 32 位有符号数相加后产生 32 位有符号数，饱和运算后存放于累加寄存器 ACC 中。

Operation

(D) If (cond) ACC <=sat{ACC + sat [(rs1_lsb16s * rs2_lsb16s) <<1]}

Elsenop

Execute time

2 cycles

Multiply Operations

SMACL.H

Signed Integer Multiply and Accumulation
16msb x 16msb, with Left Shift and Saturate

Multiply Operations

SMACL.H

General Form **SMACL.H** (.type) ACC0, rs1, rs2 /* (D) */
SMACL.H (.type) ACC1, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, ACC0, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	0000a	z	creg								

a: ACC select

-- "0" ACC0 X12/Y12

-- "1" ACC1 X22/Y22

SMACL.H

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs1	smsb	.M	001 10100
rs2	smsb	.M	

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，结果左移一位并做饱和处理后，与累加寄存器 ACC0 或 ACC1 中的 32 位有符号数相加后产生 32 位有符号数，饱和运算后存放于累加寄存器 ACC 中。

Operation

(D) If (cond) ACC <= sat{ACC+ sat [(rs1_msb16s * rs2_msb16s) <<1]}
Elsenop

Execute time

2 cycles

Multiply Operations

MSU.L

Signed Integer Multiply and Subtraction 16lsb x 16lsb

Multiply Operations

MSU.L

General Form **MSU.L** (.type) rs1, rs2 /* (F) */
 MSU.L (.type) rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, ACC0, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.F1



(D) Format

MA.D1



a: ACC select

-- "0" ACC0 X12/Y12

-- "1" ACC1 X22/Y22

MSU.L

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rs1	slsb	.M	001	10000
rs2	slsb	.M		
rs1	slsb	.M	001	10001
rs2	scst5	.M		

Description

这条指令的作用是将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘，结果与累加寄存器 ACC 中的 32 位有符号数相减后产生 32 位有符号数，存放于累加寄存器 ACC 中。

Operation

(F) $ACC \leq ACC - rs1\_lsb16s * rs2\_lsb16s$

(D) If (cond) $ACC \leq ACC - rs1\_lsb16s * rs2\_lsb16s$
Elsenop

Execute time

2 cycles



Multiply Operations

MSU.H

Signed Integer Multiply and Subtraction 16msb
x 16msb

Multiply Operations

MSU.H

General Form **MSU.H** (.type) ACC0, rd, rs1, rs2 /* (D) */
 MSU.H (.type) ACC1, rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, ACC0, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	0000a	z	creg								

a: ACC select

-- "0" ACC0 X12 /Y12

-- "1" ACC1 X22 /Y22

MSU.H

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs1	smsb	.M	001 10101
rs2	smsb	.M	

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，结果与累加寄存器 ACC 中的 32 位有符号数相减后产生 32 位有符号数，存放于累加寄存器 ACC 中。

Operation

(D) If (cond) rd <= rd - rs1_msb16s * rs2_msb16s

Elsenop

Execute time

2 cycles

Multiply Operations

SMSUL.L

Signed Integer Multiply and Subtraction 16lsb x 16lsb, with Left Shift and Saturate

Multiply Operations

SMSUL.L

General Form **SMSUL.L** (.type) ACC0, rs1, rs2 /* (D) */
SMSUL.L (.type) ACC1, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, ACC0, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	0000a	z	creg								

a: ACC select

-- "0" ACC0 X12 /Y12

-- "1" ACC1 X22 /Y22

SMSUL.L

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs1	slsb	.M	001 10010
rs2	slsb	.M	

Description

这条指令的作用是将寄存器 rs2 中的低 16 位有符号数与寄存器 rs1 中的低 16 位有符号数相乘，结果左移一位并做饱和处理后，与累加寄存器 ACC0 或 ACC1 中的 32 位有符号数相减后产生 32 位有符号数，存放于累加寄存器 ACC 中。

Operation

(D) If (cond) ACC <= sat{ACC – sat[(rs1_lsb16s * rs2_lsb16s) <<1]}

Elseenop

Execute time

2 cycles

Multiply Operations

SMSUL.H

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, ACC0, ACC1 all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5		rs1		0000a		z	creg		

-- "1" ACC1 X22 /Y22

Description

这条指令的作用是将寄存器 **rs2** 中的高 16 位有符号数与寄存器 **rs1** 中的高 16 位有符号数相乘，结果右移一位并做饱和处理后，与累加寄存器 **ACC** 中的 32 位有符号数相减后产生 32 位有符号数，存放于相应的累加寄存器 **ACC** 中。

```
Else      nop
```

2 cycles

Multiply Operations

DTP.S/.SU.2

Dot Product, Signed or Unsigned Packed
16-Bit

Multiply Operations

DTP.S/.SU.2

General Form **DTP.S.2** (.type) rd, rs1, rs2 /* (D) */
 DTP.SU.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op	1	rs2/const5	rs1	rd	z	creg							

DTP.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.M	001 00000
rs1	2hws	.M	
rs2	2hws		

DTP.SU.2

(D) Format

rd	sint	.M	001 00001
rs1	2hws	.M	
rs2	2hwu		

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，低 16 位有符号数与低 16 位有符号数相乘，将结果相加存放于寄存器 rd 中。

Rs1_smsb16	rs1_slb16
DTP.S.2	
rs2_smsb16	rs2_slb16

rd = rs1_smsb16 x rs2_smsb16 + rs1_slb16 x rs2_slb16

Operation

(D) If (cond) rd = rs1_smsb16 x rs2_smsb16 + rs1_slb16 x rs2_slb16
 Elsenop

Execute time

2 cycles

Multiply Operations

DTPN.S/.SU.2

Dot Product with Negate, Signed or Unsigned Packed 16-Bit

Multiply Operations

DTPN.S/.SU.2

General Form **DTPN.S.2** (.type) rd, rs1, rs2 /* (D) */
DTPN.SU.2 (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op	1	rs2/const5	rs1	rd	z	creg							

DTPN.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.M	001 00010
rs1	2hws	.M	
rs2	2hws		

DTPN.SU.2

(D) Format

rd	sint	.M	001 00011
rs1	2hws	.M	
rs2	2hwu		

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，低 16 位有符号数与低 16 位有符号数相乘，将结果相减存放于寄存器 rd 中。

Rs1_smsb16	rs1_slb16
DTPN.S.2	
rs2_smsb16	rs2_slb16
rd = rs1_smsb16 x rs2_smsb16 – rs1_slb16 x rs2_slb16	

Operation

(D) If (cond) rd = rs1_smsb16 x rs2_smsb16 – rs1_slb16 x rs2_slb16
Elsenop

Execute time

2 cycles

Multiply Operations

Multiply Operations

DTPR.S/.SU.2 Dot Product, Shift and Round, Signed or Unsigned Packed 16-Bit DTPR.S/.SU.2

General Form **DTPR.S.2** (.type) rd, rs1, rs2 /* (D) */
DTPR.SU.2 (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op	1	rs2/const5	rs1	rd	z	creg							

DTPR.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.M	001 00100
rs1	2hws	.M	
rs2	2hws		

DTPR.SU.2

(D) Format

rd	sint	.M	001 00101
rs1	2hws	.M	
rs2	2hwu		

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，低 16 位有符号数与低 16 位有符号数相乘，将结果相加带补偿右移 16 位后，存放于寄存器 rd 中。

Rs1_smsb16	rs1_sl sb16
DTPR.S.2	
rs2_smsb16	rs2_sl sb16

rd_sint = (rs1_smsb x rs2_smsb + rs1_sl sb x rs2_sl sb + 0x8000) >>16

Operation

(D) If (cond) rd_sint = (rs1_smsb x rs2_smsb + rs1_sl sb x rs2_sl sb + 0x8000) >>16
Elsenop

Execute time

2 cycles

Multiply Operations

DTPNR.S/.SU.2 Dot Product with Negate, Shift and Round, Signed or Unsigned Packed 16-Bit

Multiply Operations

DTPNR.S/.SU.2

General Form **DTPNR.S.2** (.type) rd, rs1, rs2 /* (D) */
DTPNR.SU.2 (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

DTPNR.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.M	001 00110
rs1	2hws	.M	
rs2	2hws		

DTPNR.SU.2

(D) Format

rd	sint	.M	001 00111
rs1	2hws	.M	
rs2	2hwu		

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，低 16 位有符号数与低 16 位有符号数相乘，将结果相减后带补偿右移 16 位后，存放于寄存器 rd 中。

Rs1_smsb16	rs1_slb16
DTPNR.S.2	
rs2_smsb16	rs2_slb16

$rd\_sint = (rs1\_smsb \times rs2\_smsb - rs1\_slsb \times rs2\_slsb + 0x8000) \gg 16$

Operation

(D) If (cond) $rd\_sint = (rs1\_smsb \times rs2\_smsb + rs1\_slsb \times rs2\_slsb - 0x8000) \gg 16$
Elsenop

Execute time

2 cycles



Multiply Operations

BTFDTP.2

Butterfly Dot Product, Signed or Unsigned
Packed 16-Bit

Multiply Operations

BTFDTP.2

General Form BTFDTP.2 (.type) rd, rs1, rs2 /* (D) */
.type = .M

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

BTFDTP.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sllong	.M	001 01011
rs1	2hws	.M	
rs2	2hws		

Description

这条指令的作用是将寄存器 rs2 中的高 16 位有符号数与寄存器 rs1 中的高 16 位有符号数相乘，低 16 位有符号数与低 16 位有符号数相乘，将结果相加后存放于寄存器 rd+1 中，将乘法结果相减后保存在寄存器 rd 中，实现蝶型运算。

Rs1_smsb16	rs1_slb16
BTFDTP.2	
rs2_smsb16	rs2_slb16

$rd+1\_sint = rs1\_smsb \times rs2\_smsb + rs1\_slsb \times rs2\_slsb$
$rd\_sint = rs1\_smsb \times rs2\_slsb - rs1\_slsb \times rs2\_smsb$

Operation

If (cond)

$rd+1\_sint = rs1\_smsb \times rs2\_smsb + rs1\_slsb \times rs2\_slsb$

$rd\_sint = rs1\_smsb \times rs2\_slsb - rs1\_slsb \times rs2\_smsb$

Elsenop

Execute time

2 cycles

Multiply Operations

BTFDTPR.2

```
.type = .M
```

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5		rs1		rd		z	creg		

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.M	001 01100
rs1	2hws	.M	
rs2	2hws		

这条指令的作用是将寄存器 **rs2** 中的高 16 位有符号数与寄存器 **rs1** 中的高 16 位有符号数相乘，低 16 位有符号数与低 16 位有符号数相乘，将结果相加后进行带补偿右移 15 位存放于寄存器 **rd** 的高 16 位中，将结果相减后进行带补偿右移 15 位存放于寄存器 **rd** 的低 16 位中，实现蝶型运算。

Rs1_smsb16	rs1_sl sb16
BTFDTPR.2	
rs2_smsb16	rs2_sl sb16

rd_smsb16 = (rs1_smsb x rs2_smsb + rs1_slb x rs2_slb + 0x4000) >> 15
rd_slb16 = (rs1_smsb x rs2_slb - rs1_slb x rs2_smsb + 0x4000) >> 15

```
rd_smsb16 = (rs1_smsb x rs2_smsb + rs1_slb x rs2_slb + 0x4000) >> 15
rd_slb16 = (rs1_smsb x rs2_slb - rs1_slb x rs2_smsb + 0x4000) >> 15
Elsenop
```

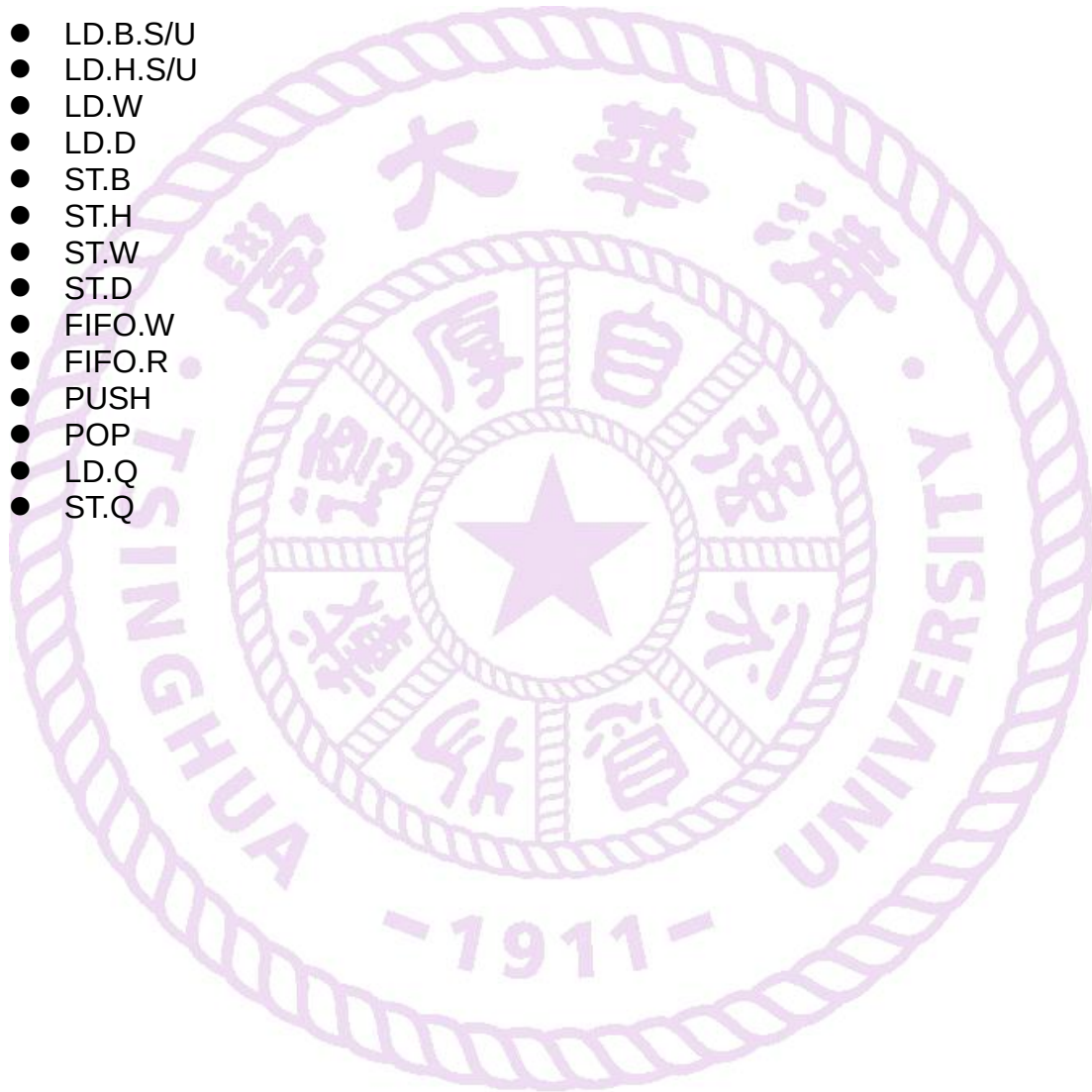
2 cycles



10 LOAD/STORE OPERATIONS

Instruction Summary

- LD.B.S/U
- LD.H.S/U
- LD.W
- LD.D
- ST.B
- ST.H
- ST.W
- ST.D
- FIFO.W
- FIFO.R
- PUSH
- POP
- LD.Q
- ST.Q



Load/Store Operations

LD.B.S/.U

LD.H.S/.U

LD.W

LD.D

LD.Q

Load Byte/ Halfword/ Word/ Doubleword
From Memory With an Unsigned
Constant Offset or Register Offset

Load/Store Operations

LD.B.S/.U

LD.H.S/.U

LD.W

LD.D

LD.Q

General Form

LD.*.* (.type) rd, *LBR /* (F) */
LD.*.* (.type) rd, *+LBR[LOR] /* (F) */
LD.*.* (.type) rd, *-LBR[LOR] /* (F) */
LD.*.* (.type) rd, *++LBR[LOR] /* (F) */
LD.*.* (.type) rd, *--LBR[LOR] /* (F) */
LD.*.* (.type) rd, *LBR++[LOR] /* (F) */
LD.*.* (.type) rd, *LBR--[LOR] /* (F) */

LD.*.* (.type) rd, *BaseR /* (D) */
LD.*.* (.type) rd, *+BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) rd, *-BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) rd, *++BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) rd, *--BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) rd, *BaseR++[OffsetR/ucst5] /* (D) */
LD.*.* (.type) rd, *BaseR--[OffsetR/ucst5] /* (D) */
LD.*.* (.type) rd, *+BaseR(ucst5) /* (D) */
LD.*.* (.type) rd, *-BaseR(ucst5) /* (D) */

LD.*.* (.type) gd, *BaseR /* (D) */
LD.*.* (.type) gd, *+BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) gd, *-BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) gd, *++BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) gd, *--BaseR[OffsetR/ucst5] /* (D) */
LD.*.* (.type) gd, *BaseR++[OffsetR/ucst5] /* (D) */
LD.*.* (.type) gd, *BaseR--[OffsetR/ucst5] /* (D) */
LD.*.* (.type) gd, *+BaseR(ucst5) /* (D) */
LD.*.* (.type) gd, *-BaseR(ucst5) /* (D) */

.type = .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

DT.F6

15	14	13	12	9	8	6	5	4	3	0
s	10	Op_1	mode	A	B	rd				

15	14	13	12	9	8	6	5	4	3	0
s	10	Op_1	mode	A	B	gd				

A:选择存储器访问基址寄存器 LBR

“0” LBR:X8/Y8 “1” LBR:X10/Y10

B:选择存储器访问偏移量寄存器 LOR

— “0” LOR:X4/Y4 “1” LOR:X6/Y6

Mnemonic	Meaning	Register
LBR	Load Base Address Register	X8/Y8 X10/Y10
LOR	Load Offset Address Register	X4/Y4 X6/Y6

Mode: 存储器访问模式选择

语法形式	Mode	基址修改方式	偏移地址对齐
*LBR	000	保持	
	001		
*+LBR[LOR]	010	保持	是
*-LBR[LOR]	011	保持	是
*LBR++[LOR]	100	后加	是
*LBR--[LOR]	101	后减	是
*++LBR [LOR]	110	先加	是
*--LBR[LOR]	111	先减	是

表 1 16 位指令地址产生方式

(D) Format

DT.D3

31	30	29	28	27	26	23	22	19	18	14	13	9	8	4	3	2	0
s	11	11	Op	mode	ffset/cst5	baseR	Rd/Gd	z	creg								

Load Mode

Syntax	Mode	基址修改方式	偏移地址对齐
*BaseR	0000	保持	
	0001		
*+BaseR[Offset]	0010	保持	否
*-BaseR[Offset]	0011	保持	否
*BaseR++[Offset]	0100	后加	否
*BaseR--[Offset]	0101	后减	否
*++BaseR[Offset]	0110	先加	否
*--BaseR [Offset]	0111	先减	否
*+BaseR(ucst5)	1000	保持	否
*-BaseR(ucst5)	1001	保持	否
*+BaseR[ucst5]	1010	保持	是
*-BaseR[ucst5]	1011	保持	是

*BaseR++[ucst5]	1100	后加	是
*BaseR-[ucst5]	1101	后减	是
+++BaseR[ucst5]	1110	先加	是
*--BaseR [ucst5]	1111	先减	是

表 2 32 位指令地址产生方式



LD.B.S		(F) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.D .D	0000
LD.B.U		(F) Format	
rd	sint	.D .D	0001
LD.B.S		(D) Format	
rd	Int	.D	100 0
BaseR	Int	.D	
OffsetR	Int		
LD.B.U		(D) Format	
rd	Int	.D	100 1
BaseR	Int	.D	
OffsetR	Int		
LD.H.S		(F) Format	
rd	sint	.D .D	0010
LD.H.U		(F) Format	
rd	sint	.D .D	0011
LD.H.S		(D) Format	
rd	Int	.D	101 0
BaseR	Int	.D	
OffsetR	Int		
LD.H.U		(D) Format	
rd	Int	.D	101 1
BaseR	Int	.D	
OffsetR	Int		
LD.W		(F) Format	
rd	sint	.D .D	0100
LD.W		(D) Format	
rd	Int	.D	110 0
BaseR	Int	.D	
OffsetR	Int		
LD.D		(F) Format	
rd	sint	.D .D	0101

LD.D	(D) Format			
rd	sllong	.D	110	0
BaseR	Int	.D		
OffsetR	Int			
Gd	128-bit	.D	100	0
BaseR	Int	.D		
OffsetR	int			

Description

这条指令的作用是从存储器中读取数据，存放于目的寄存器中，并对基址寄存器做相应的修改。数据的格式如表 3 所示。

指令	数据格式
LD.B.S	一个字节有符号数
LD.B.U	一个字节无符号数
LD.H.S	有符号半字
LD.H.U	无符号半字
LD.W	字
LD.D	双字
LD.Q	四字

表 3 不同指令的数据读取格式

如果读取的数据是一个字节的有符号数，则对目标寄存器的高 24 位进行符号扩展，如果读取的数据是一个字节的无符号数，则将目标寄存器的高 24 位清零。如果读取的数据是有符号半字，则对目标寄存器的高 16 位进行符号扩展，如果读取的数据是无符号半字，则将目标寄存器的高 16 位清零。

数据的访问地址是由基址寄存器和偏移地址共同产生的，16 位指令和 32 位指令的地址产生方法不同：

16 位 load 指令的基址寄存器和偏移寄存器为固定的几个本地寄存器，通过指令码中的两个 bit 来选择基址寄存器 LBR 和偏移寄存器 LOR，“A”用于选择 LBR：“0”→LBR:X3/Y3；“1”→LBR:X4/Y4；“B”用于选择 LOR：“0”→LOR:X8/Y8；“1”→LOR:X9/Y9。

16 位 load 指令地址的产生方法如表 1 所示。“后加”指的是以 LBR 作为读取数据的地址，数据读取结束后将 LBR 与 LOR 相加产生新的 LBR。“后减”指的是以 LBR 作为读取数据的地址，数据读取结束后将 LBR 与 LOR 相减产生新的 LBR。“先加”指的是首先将 LBR 与 LOR 相加产生新的 LBR，并作为数据读取的地址。“先减”指的是首先将 LBR 与 LOR 相减产生新的 LBR，并作为数据读取的地址。“后加 1”指的是以 LBR 作为读取数据的地址，数据读取结束后将 LBR 加 1 产生新的 LBR。“后减 1”指的是以 LBR 作为读取数据的地址，数据读取结束后将 LBR 减 1 产生新的 LBR。“先加 1”指的是首先将 LBR 加 1 产生新的 LBR，并作为数据读取的地址。“先减 1”指的是首先将 LBR 减 1 产生新的 LBR，并作为数据读取的地址。

32 位 load 指令地址的产生方法如表 2 所示。“后加”指的是以 BaseR 作为读取数据的地址，数据读取结束后将 Offset 与 BaseR 相加产生新的 BaseR。“后减”指的是以 BaseR 作为读取数据的地址，数据读取结束后将 BaseR 与 Offset 相减产生新的 BaseR。“先加”指的是首先将 Offset 与 BaseR 相加产生新的 BaseR，并作为数据读取的地址。“先减”指的是首先将 BaseR 与 Offset 相减产生新的 BaseR，并作为数据读取的地址。

Execute time

4 cycles

Load/Store Operations

ST.B Store Byte/ Halfword/ Word/ Doubleword to Memory
ST.H With an Unsigned Constant Offset or Register Offset
ST.W
ST.D
ST.Q

Load/Store Operations

ST.B
ST.H
ST.W
ST.D
ST.Q

General Form

ST.* (.type) *SBR, rs /* (F) */
ST.* (.type) *+SBR[SOR], rs /* (F) */
ST.* (.type) *-SBR[SOR], rs /* (F) */
ST.* (.type) *++SBR[SOR], rs /* (F) */
ST.* (.type) *--SBR[SOR], rs /* (F) */
ST.* (.type) *SBR-[SOR], rs /* (F) */

ST.* (.type) *BaseR, rs /* (D) */
ST.* (.type) *+BaseR[OffsetR/ucst5], rs /* (D) */
ST.* (.type) *-BaseR[OffsetR/ucst5], rs /* (D) */
ST.* (.type) *++BaseR[OffsetR/ucst5], rs /* (D) */
ST.* (.type) *--BaseR[OffsetR/ucst5], rs /* (D) */
*/
ST.* (.type) *BaseR++[OffsetR/ucst5], rs /* (D) */
ST.* (.type) *BaseR--[OffsetR/ucst5], rs /* (D) */
ST.* (.type) *+BaseR(ucst5), rs /* (D) */
ST.* (.type) *-BaseR(ucst5), rs /* (D) */

ST.* (.type) *BaseR, gs /* (D) */
ST.* (.type) *+BaseR[OffsetR/ucst5], gs /* (D) */
ST.* (.type) *-BaseR[OffsetR/ucst5], gs /* (D) */
ST.* (.type) *++BaseR[OffsetR/ucst5], gs /* (D) */
ST.* (.type) *--BaseR[OffsetR/ucst5], gs /* (D) */
ST.* (.type) *BaseR++[OffsetR/ucst5], gs /* (D) */
ST.* (.type) *BaseR--[OffsetR/ucst5], gs /* (D) */
ST.* (.type) *+BaseR(ucst5), gs /* (D) */
ST.* (.type) *-BaseR(ucst5), gs /* (D) */

.type = .D

Syntax

(F) rd, rs: X0~X11/Y0~Y11 : G0~G3, the low-order 16 Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

DT.F6

15	14	13	12	9	8	6	5	4	3	0
s	10	Op_1	mode	A	B	rs				

s	10	Op_1	mode	A	B	gs				

A:选择存储器访问基址寄存器 SBR

— “0” SBR:X9/Y9 “1” SBR:X11/Y11

B:选择存储器访问偏移量寄存器 SOR

— “0” SOR:X5/Y5 “1” SOR:X7/Y7

Mnemonic	Meaning	Register
SBR	Store Base Address Register	X9/Y9 X11/Y11
SOR	Store Offset Address Register	X5/Y5 X7/Y7

Mode: 存储器访问模式选择

语法形式	Mode	基址修改方式	偏移地址对齐
*SBR	000	保持	
	001		
*+SBR[SOR]	010	保持	是
*-SBR[SOR]	011	保持	是
*SBR++[SOR]	100	后加	是
*SBR--[SOR]	101	后减	是
*++SBR[SOR]	110	先加	是
*--SBR[SOR]	111	先减	是

表 1 16 位指令地址产生方式

(D) Format

DT.D3

31	30	29	28	27	26	23	22	19	18	14	13	9	8	4	3	2	0
s	11	11	Op	mode	ffset/cst5	baseR	rd	z	creg								

Store Mode

Syntax	Mode	基址修改方式	偏移地址对齐
*BaseR	0000		是
	0001		是
*+BaseR[Offset]	0010	保持	否
*-BaseR[Offset]	0011	保持	否
*BaseR++[Offset]	0100	后加	否
*BaseR--[Offset]	0101	后减	否
*++BaseR[Offset]	0110	先加	否
*--BaseR [Offset]	0111	先减	否
*+BaseR(ucst5)	1000	保持	否
*-BaseR(ucst5)	1001	保持	否
*+BaseR[ucst5]	1010	保持	是
*-BaseR[ucst5]	1011	保持	是

*BaseR++[ucst5]	1100	后加	是
*BaseR-[ucst5]	1101	后减	是
+++BaseR[ucst5]	1110	先加	是
*--BaseR [ucst5]	1111	先减	是

表 2 32 位指令地址产生方式



ST.B		(F) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
rs	sint	.D .D	0110
ST.B		(D) Format	
rs	Int	.D	111 0
BaseR	Int	.D	
OffsetR	Int		
ST.H		(F) Format	
rs	sint	.D .D	0111
ST.H		(D) Format	
rs	Int	.D	111 1
BaseR	Int	.D	
OffsetR	Int		
ST.W		(F) Format	
rs	sint	.D .D	1000
ST.W		(D) Format	
rs	Int	.D	011 0
BaseR	Int	.D	
OffsetR	Int		
ST.D		(F) Format	
rs	sint	.D .D	1001
ST.D		(D) Format	
rs	sllong	.D	011 1
BaseR	Int	.D	
OffsetR	Int		
ST.Q		128-bit	
BaseR	int	.D	100 1
OffsetR	int	.D	

Description

这条指令的作用是从源寄存器中提取数据，存放于存储器中，并对基址寄存器做相应的修改。数据的格式如表 3 所示。

指令	数据格式
ST.B	一个字节
ST.H	半字
ST.W	字
ST.D	双字

ST.Q	四字

表 3 不同指令的数据读取格式

如果保存的数据是一个字节，则寄存器中最低位字节的数据被存放于存储器，如果保存的数据时一个半字，则寄存器中低位半字的数据被保存于存储器，如果保存的数据为一个字，则整个寄存器的数据被保存于存储器中。

数据的访问地址是由基址寄存器和偏移地址共同产生的，16 位指令和 32 位指令的地址产生方法不同：

16 位 store 指令的基址寄存器和偏移寄存器为固定的几个本地寄存器，通过指令码中的两个 bit 来选择基址寄存器 SBR 和偏移寄存器 SOR，“A”用于选择 SBR：“0”→SBR:X3/Y3；“1”→SBR:X4/Y4；“B”用于选择 SOR：“0”→SOR:X8/Y8；“1”→SOR:X9/Y9。

16 位 load 指令地址的产生方法如表 1 所示。“后加”指的是以 SBR 作为保存数据的地址，数据保存结束后将 SBR 与 SOR 相加产生新的 SBR。“后减”指的是以 SBR 作为保存数据的地址，数据保存结束后将 SBR 与 SOR 相减产生新的 SBR。“先加”指的是首先将 SBR 与 SOR 相加产生新的 SBR，并作为数据保存的地址。“先减”指的是首先将 SBR 与 SOR 相减产生新的 SBR，并作为数据保存的地址。“后加 1”指的是以 SBR 作为保存数据的地址，数据保存结束后将 SBR 加 1 产生新的 SBR。“后减 1”指的是以 SBR 作为保存数据的地址，数据保存结束后将 SBR 减 1 产生新的 SBR。“先加”指的是首先将 SBR 加 1 产生新的 SBR，并作为数据保存的地址。“先减”指的是首先将 SBR 减 1 产生新的 SBR，并作为数据保存的地址。

32 位 store 指令地址的产生方法如表 3 所示。数据的访问地址是由基址寄存器和偏移地址共同产生的，地址的产生方法如表 2 所示。“后加”指的是以 BaseR 作为保存数据的地址，数据保存结束后将 Offset 与 BaseR 相加产生新的 BaseR。“后减”指的是以 BaseR 作为保存数据的地址，数据保存结束后将 BaseR 与 Offset 相减产生新的 BaseR。“先加”指的是首先将 Offset 与 BaseR 相加产生新的 BaseR，并作为数据保存的地址。“先减”指的是首先将 BaseR 与 Offset 相减产生新的 BaseR，并作为数据保存的地址。

Load/Store Operations

PUSH PUSH Stack and POP Stack
POP

Load/Store Operations

PUSH
POP

General Form **PUSH (.type) rs** /* (F) */
 PUSH (.type) rs /* (D) */
 POP (.type) rd /* (F) */
 POP (.type) rd /* (D) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

DT.F2

PUSH

15	14	13	12	8	7	5	4	0
s	10	10110	000	rd				

POP

15	14	13	12	8	7	5	4	0
s	10	10110	001	rd				

DT.D5

PUSH

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	000	11110	00010	rs	xxxxx	z	creg								

POP

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	000	11110	00011	xxxxx	rd	z	creg								

Description

这条指令的作用是将源寄存器中的 32 位数压栈或者将栈的数读取到目标寄存器中，并改变栈指针的值，同时判断栈指针是否在栈内，如果栈指针溢出，则这次栈操作无效，同时将 CSR 中的相关标志位 SOV 置高。可以通过 MOV.RC 指令来清除这一标志位。

SMR 是用于控制 STACK 的控制管理寄存器，其中包括栈指针 SP，栈顶地址 SKT，栈底地址 SKB。栈指针 SP 由 **PUSH** 和 **POP** 指令进行修改，栈顶地址和栈底地址由 **MOV.RC** 指令进行赋值，用于保护栈空间，如果栈指针小于栈底地址，则会激活栈下溢信号；当栈指针大于栈顶地址，则激活栈上溢信号。

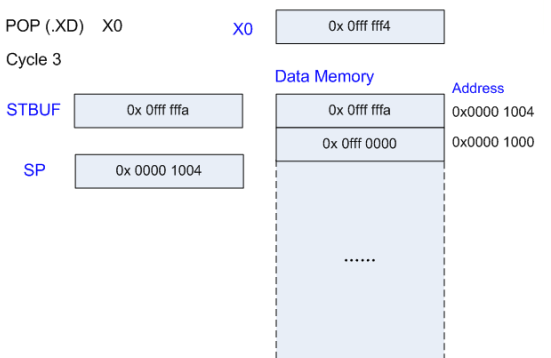
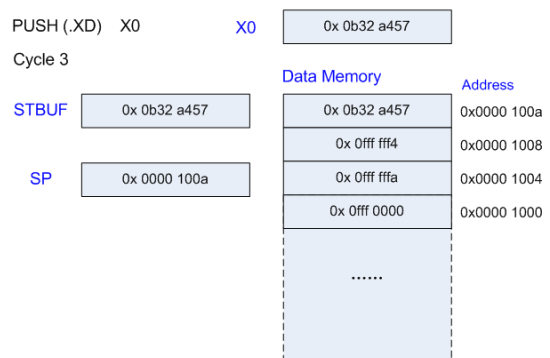
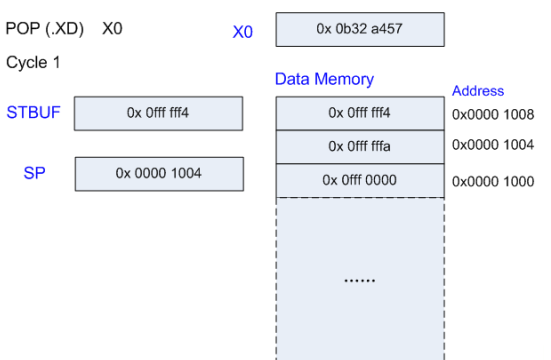
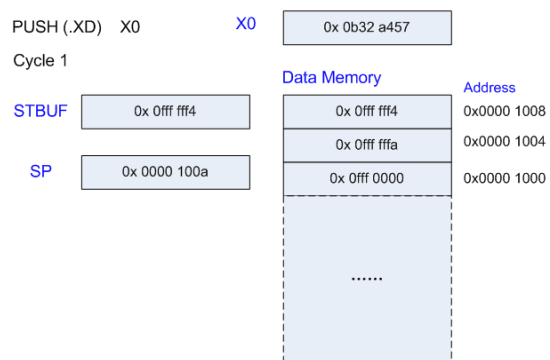
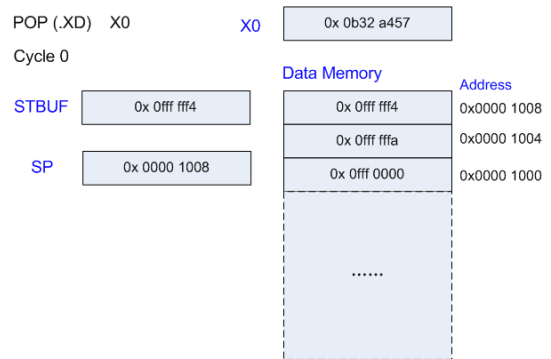
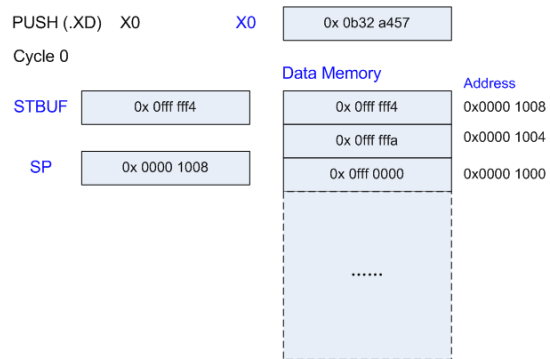
STBUF 寄存器的内容和栈顶数据相同，并在栈顶数据被更新的同时被更新。

31		0
SP		
31		0
SKT		
31		0
SKB		
31		0
STBUF		

Execute time

4 cycles

Example

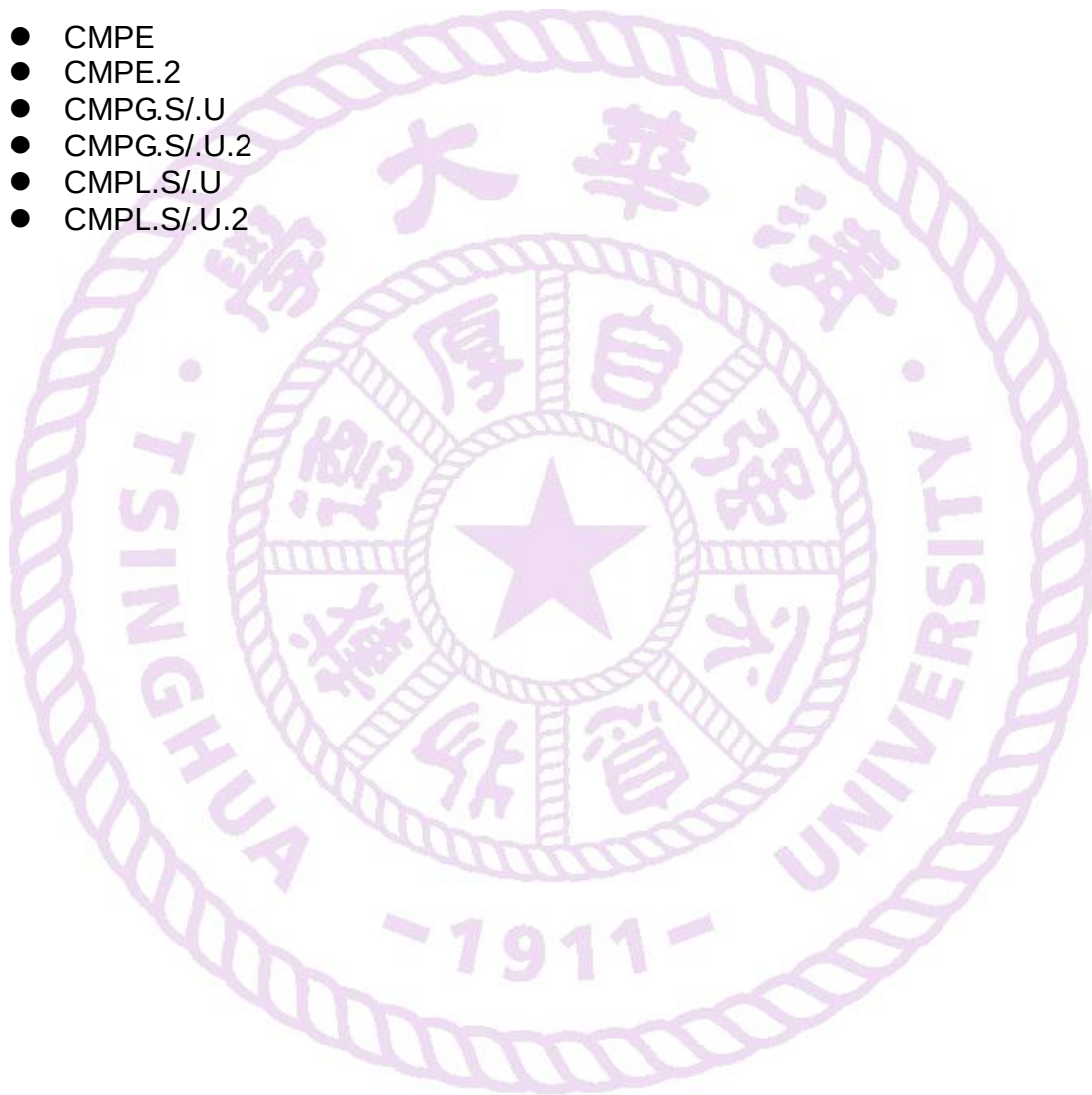




11 COMPARE OPERATIONS

Instruction Summary

- CMPE
- CMPE.2
- CMPG.S/.U
- CMPG.S/.U.2
- CMPL.S/.U
- CMPL.S/.U.2



Compare Operations

CMPE

Integer Compare for Equality

Compare Operations

CMPE

General Form **CMPE** (.type) rd, rs1, rs2 /* (D) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op	1	rs2/const5	rs1	rd	z	creg							

CMPE

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	Int	.D	000	10000
rs1	int	.D		
rs2	int			
rd	Int	.D	000	10001
rs1	sint	.D		
rs2	scst5			

Description

这条指令的作用是将寄存器 rs1 中的数据与寄存器 rs2 中的数据进行比较，如果两个数据相等，则将 rd 置全 1，如果两个数据不相等，则将 rd 置零。

Operation

If (cond)

```
{  if (rs1 == rs2/cst5    rd <= 32'hffffff  
    else    rd <= 0  
}
```

Elsenop

Execute time

B. cycle



CMPE.2

Integer Compare for Equality

Compare Operations

CMPE.2

General Form **CMPE.2** (.type) rd, rs1, rs2 /* (D) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				rs2/const5		rs1		rd			z	creg	

CMPE **(D) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.D	000 10100
rs1	2hws	.D	
rs2	2hws		

Description

这条指令的作用是将寄存器 **rs1** 中的数据与寄存器 **rs2** 中的高 16 位数据和低 16 位数据分别进行比较。如果高 16 位两个数据相等，则将 **rd** 的高 16 位置 **FFFF**，如果高 16 位两个数据不相等，则将 **rd** 的高 16 位置零；如果低 16 位两个数据相等，则将 **rd** 的低 16 位置 **FFFF**，如果低 16 位两个数据不相等，则将 **rd** 的低 16 位置零；

Rs1

1346	AB26

Rs2

2456	AB26

CMPE.2

Rd

16'b0000 0000 0000 0000	16'b1111 1111 1111 1111

Operation

```
If (cond_smsb16) { if (rs2_smsb16 == rs1_smsb16    rd_smsb16 <= 16'hfff  
                  else    rd_smsb16 <= 0 }  
If (cond_slsb16) { if (rs2_slsb16 == rs1_slsb16  rd_slsb16 <= 16'hfff  
                  else    rd_slsb16 <= 0 }
```

Elsenop

Execute time

B. cycle



Compare Operations

CMPG.S/.U

Signed or Unsigned Integer Compare for Greater Than

Compare Operations

CMPG.S/.U

General Form **CMPG.S** (.type) rd, rs1, rs2 /* (D) */
CMPG.U (.type) rd, rs1, rs2 /* (D) */
.type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

CMPG.S

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.D	000	00000
rs1	sint	.D		
rs2	sint			
rd	sint	.D	000	00001
rs1	sint	.D		
rs2	scst5			

CMPG.U

(D) Format

rd	uint	.D	000	00100
rs1	uint	.D		
rs2	uint			
rd	uint	.D	000	00101
rs1	uint	.D		
rs2	ucst5			

Description

这条指令的作用是将寄存器 rs1 中的数据与寄存器 rs2 中的数据进行比较，如果 rs1 > rs2，则将 rd 置全 1，如果 rs1 <= rs2，则将 rd 置零。

Operation

```
if (cond)
{
    if (rs1 > rs2/cst5 rd <= 32'hffffff)
        else    rd <= 0
}
Elsenop
```

Execute time

B. cycle



Compare Operations

CMPG.S/.U.2

Signed or Unsigned Integer Compare for Greater Than

Compare Operations

CMPG.S/.U.2

General Form **CMPG.S.2** (.type) rd, rs1, rs2 /* (D) */
 CMPG.U.2 (.type) rd, rs1, rs2 /* (D) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

CMPG.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	2hws	.D	000 10101
rs1	2hws	.D	
rs2	2hws		

CMPG.U.2

(D) Format

Rd	2hwu	.D	000 10110
rs1	2hwu	.D	
rs2	2hwu		

Description

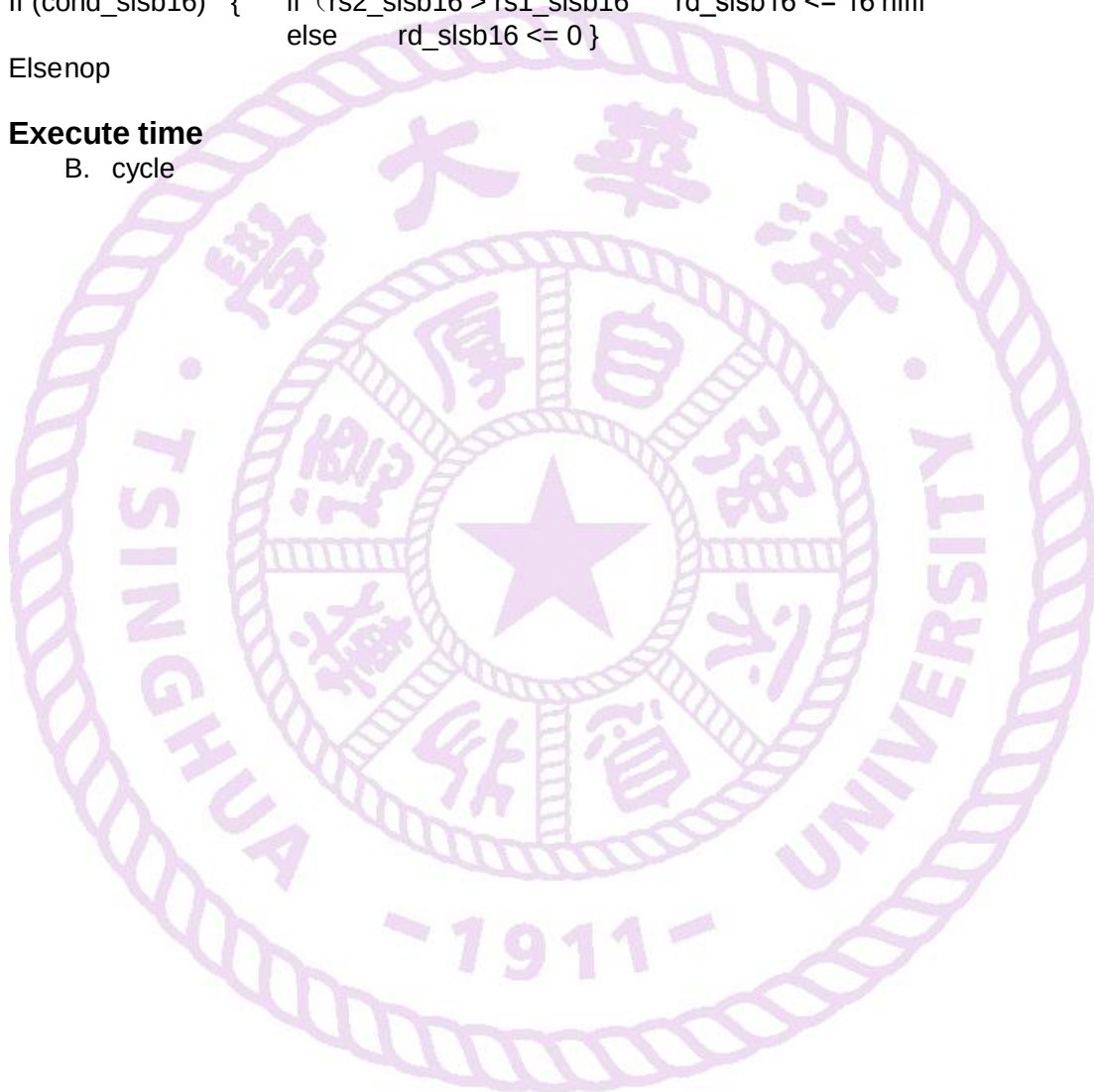
这条指令的作用是将寄存器 **rs1** 中的数据与寄存器 **rs2** 中的高 16 位数据和低 16 位数据分别进行比较。如果 **rs1** 的高 16 位大于 **rs2** 的高 16 位，则将 **rd** 的高 16 位置 **FFFF**，否则，则将 **rd** 的高 16 位置零；如果 **rs1** 的低 16 位大于 **rs2** 的低 16 位，则将 **rd** 的低 16 位置 **FFFF**，否则，则将 **rd** 的低 16 位置零；

Operation

```
If (cond_smsb16) { if (rs2_smsb16 > rs1_smsb16 rd_smsb16 <= 16'hfff
                    else rd_smsb16 <= 0 }
If (cond_slb16) { if (rs2_slb16 > rs1_slb16 rd_slb16 <= 16'hfff
                  else rd_slb16 <= 0 }
Elsenop
```

Execute time

B. cycle



Compare Operations

CMPL.S/.U

Signed or Unsigned Integer Compare for Less Than

Compare Operations

CMPL.S/.U

General Form **CMPL.S** (.type) rd, rs1, rs2 /* (D) */
CMPL.U (.type) rd, rs1, rs2 /* (D) */
.type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

CMPL.S

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.D	000	01000
rs1	sint	.D		
rs2	sint			
rd	sint	.D	000	01001
rs1	sint	.D		
rs2	scst5			

CMPL.U

(D) Format

rd	uint	.D	000	01100
rs1	uint	.D		
rs2	uint			
rd	uint	.D	000	01101
rs1	uint	.D		
rs2	ucst5			

Description

这条指令的作用是将寄存器 rs1 中的数据与寄存器 rs2 中的数据进行比较，如果 rs1 < rs2，则将 rd 置全 1，如果 rs1 >= rs2，则将 rd 置零。

Operation

If (cond)

```
{  if (rs1 < rs2/cst5 rd <= 32'hffffff  
    else      rd <= 0  
}
```

Elsenop

Execute time

B. cycle



Compare Operations

CMPL.S/.U.2

Signed or Unsigned Integer Compare for Less Than

Compare Operations

CMPL.S/.U.2

General Form **CMPL.S.2** (.type) rd, rs1, rs2 /* (D) */
CMPL.U.2 (.type) rd, rs1, rs2 /* (D) */
.type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

CMPL.S.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	2hws	.D	000 10111
rs1	2hws	.D	
rs2	2hws		

CMPL.U.2

(D) Format

rd	2hws	.D	000 11000
rs1	2hws	.D	
rs2	2hws		

Description

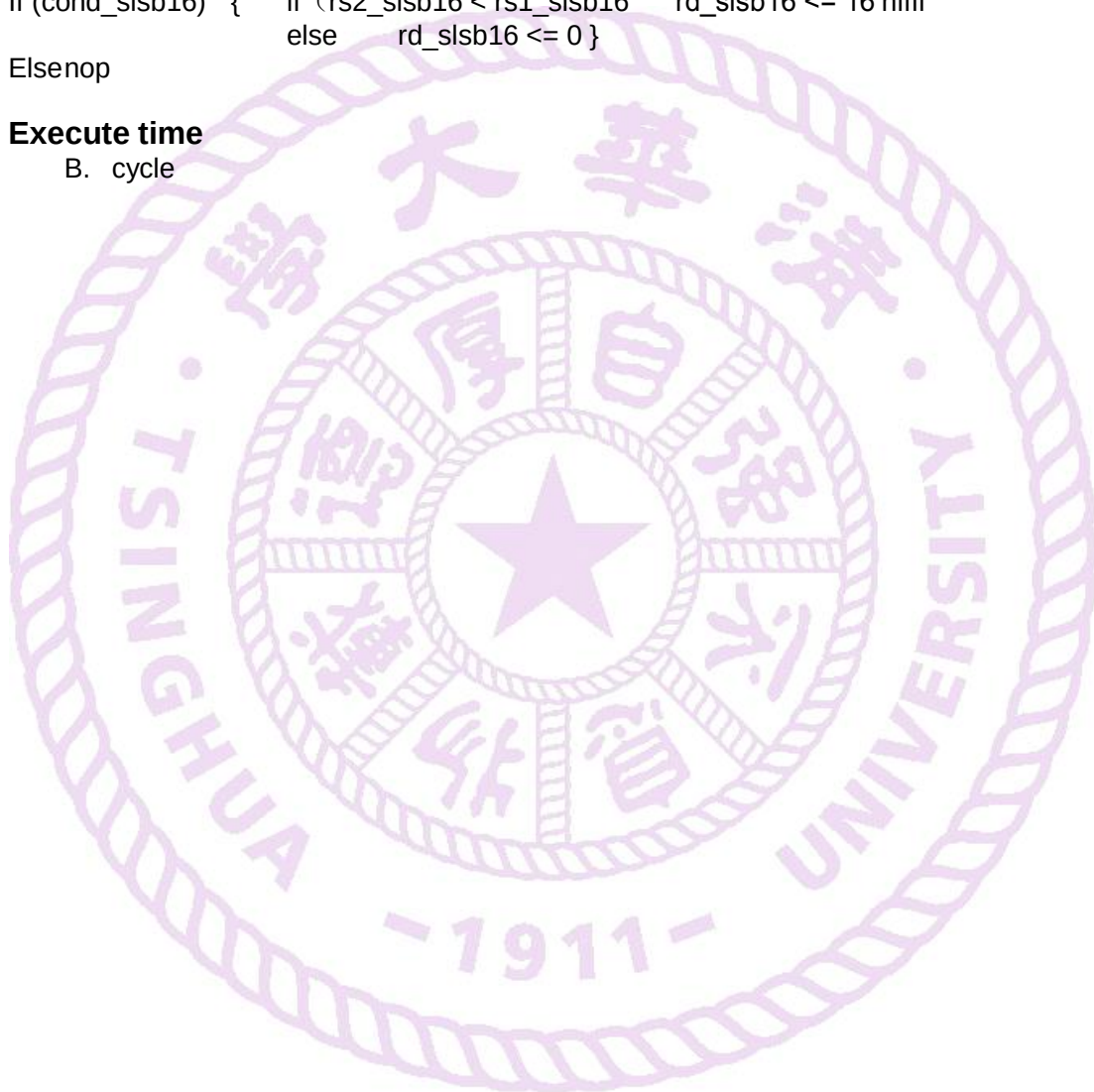
这条指令的作用是将寄存器 **rs1** 中的数据与寄存器 **rs2** 中的高 16 位数据和低 16 位数据分别进行比较。如果 **rs1** 的高 16 位小于 **rs2** 的高 16 位，则将 **rd** 的高 16 位置 **FFFF**，否则，则将 **rd** 的高 16 位置零；如果 **rs1** 的低 16 位小于 **rs2** 的低 16 位，则将 **rd** 的低 16 位置 **FFFF**，否则，则将 **rd** 的低 16 位置零；

Operation

```
If (cond_smsb16) { if (rs2_smsb16 < rs1_smsb16 rd_smsb16 <= 16'hfff
                    else rd_smsb16 <= 0 }
If (cond_slb16) { if (rs2_slb16 < rs1_slb16 rd_slb16 <= 16'hfff
                  else rd_slb16 <= 0 }
Elsenop
```

Execute time

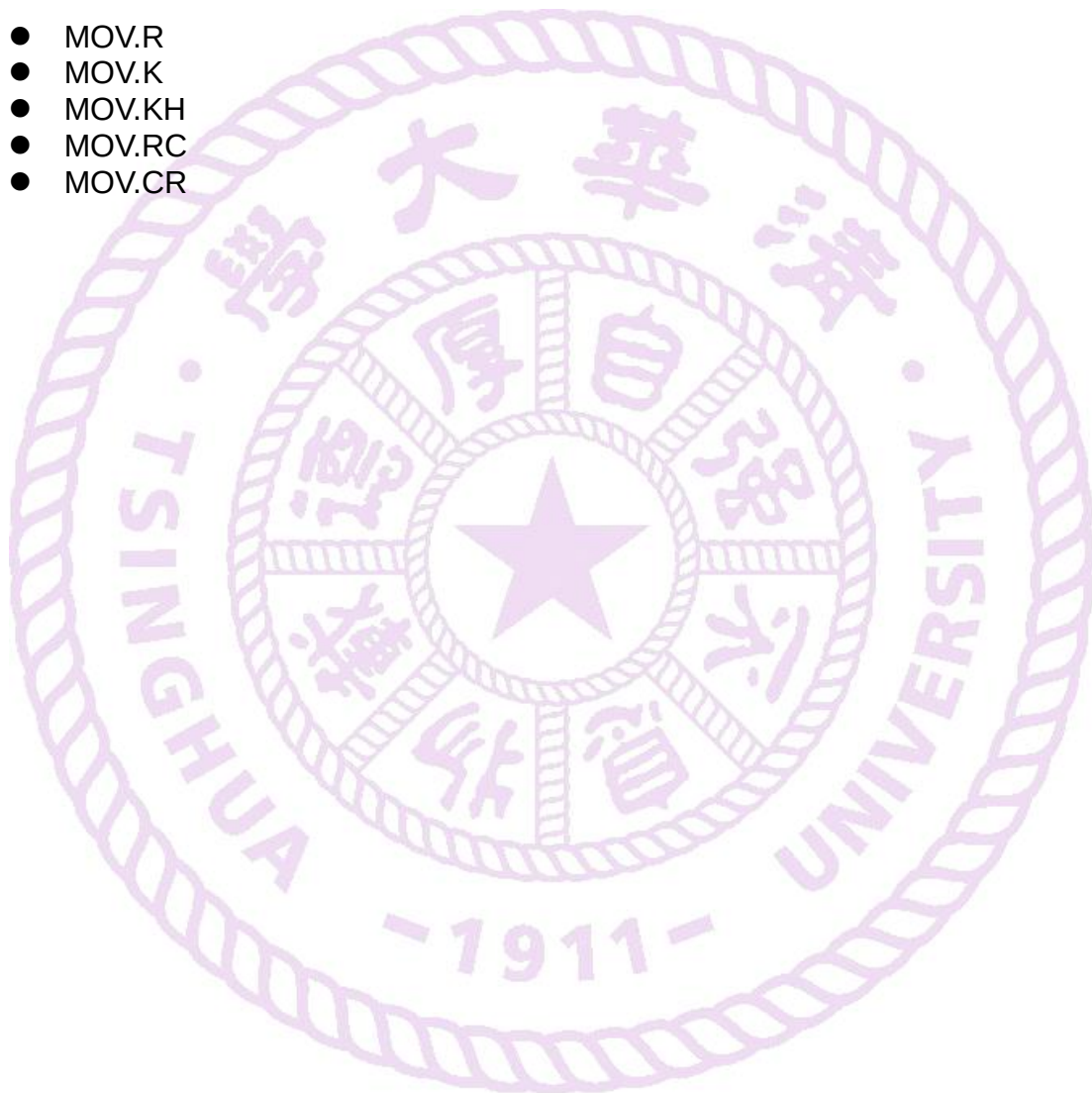
B. cycle



12 MOVE OPERATIONS

Instruction Summary

- MOV.R
- MOV.K
- MOV.KH
- MOV.RC
- MOV.CR



Move Operations MOV.R

Move Integer between Data Registers

Move Operations MOV.R

General Form **MOV.R** (.type) rd, rs /* (F) */
MOV.R (.type) rd, rs /* (D) */
.type = .A, .M, .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

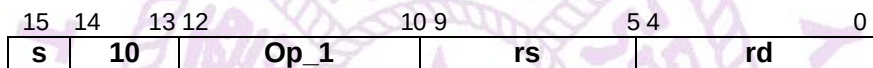
Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

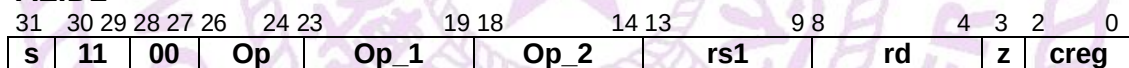
B. Format

DT.F3



(D) Format

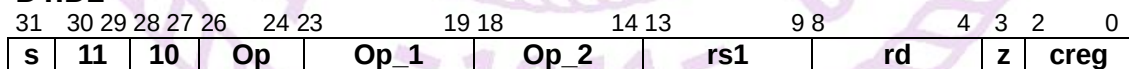
AL.D2



MA.D2



DT.D2



MOV.R

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.D	110
rs	int	.D	

MOV.R

(D) Format

rd	Int	.A	011	11111	10000
rs	int	.A			
rd	Int	.M	011	11111	10000
rs	int	.M			
rd	Int	.D	001	01000	10010
rs	int	.D			

Description

这条指令的作用是将寄存器 **rs** 中的数据复制到寄存器 **rd** 中。

Operation

```
rd <= rs;  
if(cond) rd <= rs;  
else NOP
```

Execute time

B. cycle



MOV.K

16lsb

MOV.K

/ * (D) * /

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	25	24	8				4	3	2	0	
s	11	01	op	Op 1	scst16								rd			

MOV. K	(D) Format		
Opcode map field used ...	For operand type ...	Unit	Opfield
rd	lsb16	.M	111 000
rs	scst16	.M	

这条指令的作用是将 16 位立即数复制到寄存器 `rd` 的低 16 位中，并将寄存器的高 16 位进行符号扩展。

```
(D) if(cond) rd_lsb16 <= cst16; rd_msb16 <= ext sign(cst16)
else NOP
```

B. cycle

MOV.KH

16msb

MOV,KH

```
MOV.K.H (.type) rd, cst16
.type = .M
```

/ * (D) * /

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

MA.D3

31	30	29	28	27	26	25	24	8				4	3	2	0	
s	11	01	op	Op_1	scst16								rd			

MOV. KH

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	msb16	.M	111 001
rs	cst16	.M	

Description

这条指令的作用是将 16 位立即数复制到寄存器 rd 的高 16 位中，并不改变寄存器 rd 低 16 位的值。

Operation

```
(D) if(cond) rd_msb16 <= cst16;
else NOP
```

Execute time

B. cycle

MOV.RC

Move Integer from Data Register to Control Register

Move Operations

MOV.RC

General Form **MOV.RC** (.type) rd, rs /* (D) */
 .type =.D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				Op_2		rs1			rd		z	creg	

MOV.RC

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	int	.D	0001	01000	10000
rs	Int	.D			

Description

这条指令的作用是将寄存器 **rs** 中的 32 位数据复制到控制寄存器 **rd** 中。

Operation

```
(D) if(cond) rd_ctrl_reg <= rs;
else NOP
```

Execute time

B. cycle

Move Operations

MOV.CR

Move Integer from Control Register to Data Register

Move Operations

MOV.CR

General Form **MOV.CR** (.type) rd, rs /* (D) */
 .type =.D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	rs1	rd	z	creg								

MOV.CR

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.D	001 01000 10001
rs	Int	.D	

Description

这条指令的作用是将控制寄存器 rs 中的 32 位数据复制到寄存器 rd 中。

Operation

(D) if(cond) rd <= rs_ctrl_reg;
 else NOP

Execute time

B. cycle

13 BIT PERMUTATION OPERATIONS

Instruction Summary

- SETB
- ZEROB
- PICKB
- PUTB
- SELB
- FSL
- PACK.L.2
- PACK.H.2
- PACK.LH.2
- PACK.HL.2
- PACK.L.4
- PACK.H.4
- UNPK.2
- UNPK.L.4
- UNPK.H.4
- SPACK.2
- SPACKU.4
- SWAP.2
- SWAP.4
- SHLMB1/2/3
- EXP.L/.H
- EXPB.L/H
- EXT.S/.U
- SXT1
- ZXT1
- SXT2
- ZXT2
- SXT8
- DEP
- NORM
- LMBD0/1
- DEAL
- SHFL
- ZERO
- PACKG
- PACKGL2
- PACKGH2
- UNPACKG
- UNPACKGH
- UNPACKGL

SETB

Set 1 bit to 1'b1

Bit Permutation Operations

SETB

General Form **SETB (.type) rd, rs1, rs2** /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				rs2/const5				rs1		rd		z	creg

SETB

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	Sint	.A	110	10101
rs1	Sint	.A		
rs2	Sint			
rd	Sint	.A	110	10110
rs1	Sint	.A		
rs2	Cst5			

Description

这条指令的作用是将寄存器 rs1 中由 rs2（或 5 位常数）确定的某位置 1，其它位取 rs1 对应位，结果存放于目标寄存器 rd 中。

Operation

Execute time

B. cycle

ZEROB

Set 1 bit to 1'b0

Bit Permutation Operations

ZEROB

General Form **ZEROB** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z		creg			

ZEROB

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	Sint	.A	110	10111
rs1	Sint	.A		
rs2	Sint			
rd	Sint	.A	110	11000
rs1	Sint	.A		
rs2	Cst5			

Description

这条指令的作用是将寄存器 rs1 中由 rs2（或 5 位常数）确定的某位置 0，其它位取 rs1 对应位，结果存放于目标寄存器 rd 中。

Operation

Execute time

B. cycle

PICKB

Get 1 bit from rs1

Bit Permutation Operations

PICKB

General Form **PICKB** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				rs2/const5				rs1		rd		z	creg

PICKB

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Sint	.A	110	11001
rs1	Sint	.A		
rs2	Sint			
Rd	Sint	.A	110	11010
rs1	Sint	.A		
rs2	Cst5			

Description

这条指令的作用是将寄存器 rs1 中由 rs2（或 5 位常数）确定的某位存入 rd 最低位，rd 其它位置零。

Operation

Execute time

B. cycle

PUTB

Get byte from rs1

Bit Permutation Operations

PUTB

General Form **PUTB** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z		creg			

PUTB

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	Sint	.A	110 10010
rs1	Sint	.A	
rs2	Sint		

Description

这条指令的作用是把 rs1 的 LSB 字节复制到目标寄存器的指定位置，其它位复制 rs2 的对应位，指定字节的位置由 rs1 的最高字节（MSB）的末两位指定，如 rs1 的最高字节（MSB）的末两位为 00，则把 rs1 的 LSB 复制到目标寄存器的 LSB，……，如果 rs1 的最高字节（MSB）的末两位为 11，则把 rs1 的 LSB 复制到目标寄存器的 MSB。

Operation

Execute time

B. cycle

SELB

Get byte from rs1

Bit Permutation Operations

SELB

General Form **SELB** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z		creg			

SELB		(D) Format	
Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	Sint	.A	110 01111
rs1	Sint	.A	
rs2	Sint		

Description

这条指令的作用是把 rs1 中指定的字节复制到目标寄存器的 LSB，其它位复制 rs2 的对应位，指定字节的位置由 rs2 的 LSB 指定，如 rs2 的 LSB=0x00，则把 rs1 的 LSB（0 字节）复制到目标寄存器的 LSB，如果 rs2 的 LSB>=0x03，则把 rs1 的 MSB（第 3 字节）复制到目标寄存器的 LSB。

Operation

Execute time

B. cycle

FSL

Byte shift

Bit Permutation Operations

FSL

General Form	FSL (.type) rd, rs1, rs2	/* (D) */
	.type = .A	

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z		creg			

SELB

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	Sint	.A	110 10000
rs1	Sint	.A	
rs2	Sint		

Description

这条指令的作用是把寄存器对 rs1 中的 64 位无符号数左移 8 位, 低 8 位由寄存器 rs2 中的高 8 位填充, 结果存于寄存器对 rd 中。

Operation

Execute time

B. cycle

Bit Permutation Operations

PACK.L.2

Pack 16 LSB, 16 LSB Into Packed 16-Bit

Bit Permutation Operations

PACK.L.2

General Form **PACK.L.2** (.type) rd, rs1, rs2 /* (D) */
 .type = .A, .M, .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

PACK.L.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	2hws	.A	011	11000
rs1	lsb16	.A		
rs2	lsb16			
Rd	2hws	.M	010	10000
rs1	lsb16	.M		
rs2	lsb16			
Rd	2hws	.D	001	00000
rs1	lsb16	.D		
rs2	lsb16			

Description

这条指令的作用是将寄存器 rs1 中的低 16 位数据与寄存器 rs2 中的低 16 位数据分别存放在目标寄存器 rd 的高 16 位和低 16 位中。

Rs1

Rs1_msb16	Rs1_lsb16

Rs2

Rs2_msb16	Rs2_lsb16

PACK.L.2

Rd

Rs1_lsb16	Rs2_lsb16

Operation

If (cond) rd_msb16 <= rs1_lsb16; rd_lsb16 <= rs2_lsb16;
Elsenop

Execute time

B. cycle



Bit Permutation Operations

PACK.H.2

Operation

If (cond) rd_msb16 <= rs1_msb16; rd_lsb16 <= rs2_msb16;
Elsenop

Execute time

B. cycle



Bit Permutation Operations

Pack 16 LSB, 16 MSB Into Packed 16-Bit

PACK.LH.2

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd			z	creg	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5		rs1		rd			z	creg	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				rs2/const5		rs1		rd			z	creg	

PACK.LH.2		(D) Format		
Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	2hws	.A	011	11010
rs1	lsb16	.A		
rs2	msb16			
Rd	2hws	.M	010	10010
rs1	lsb16	.M		
rs2	msb16			
Rd	2hws	.D	001	00010
rs1	lsb16	.D		
rs2	msb16			

Description

这条指令的作用是将寄存器 **rs1** 中的低 16 位数据与寄存器 **rs2** 中的高 16 位数据分别存放在目标寄存器 **rd** 的高 16 位和低 16 位中。

Rs1

Rs1_msb16	Rs1_lsb16

Rs2

Rs2_msb16	Rs2_lsb16

PACK.LH.2

Rd

Rs1 sb16	Rs2 msb16

Operation

If (cond) rd_msb16 <= rs1_lsb16; rd_lsb16 <= rs2_msb16;
Elsenop

Execute time

B. cycle



Bit Permutation Operations

Pack 16MSB, 16 LSB Into Packed 16-Bit

PACK.HL.2

.type =.A, .M, .D

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		rd			z	creg	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5		rs1		rd			z	creg	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				rs2/const5		rs1		rd			z	creg	

PACK.HL.2		(D) Format		
Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	2hws	.A	011	11011
rs1	lsb16	.A		
rs2	lsb16			
Rd	2hws	.M	010	10011
rs1	lsb16	.M		
rs2	lsb16			
Rd	2hws	.D	001	00011
rs1	lsb16	.D		
rs2	lsb16			

这条指令的作用是将寄存器 **rs1** 中的高 16 位数据与寄存器 **rs2** 中的低 16 位数据分别存放在目标寄存器 **rd** 的高 16 位和低 16 位中。

Rs1_msb16	Rs1_lsb16

Rs2 msb16	Rs2 lsb16

Rd

Rs1 msb16	Rs2 lsb16

Operation

If (cond) rd_msb16 <= rs1_msb16; rd_lsb16 <= rs2_lsb16;
Elsenop

Execute time

B. cycle



Bit Permutation Operations

Pack Low Bytes of Four Half-Words Into Packed 8-Bit

PACK.L.4

```
.type = .M,.M,.D
```

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5		rs1		rd		z		creg	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				rs2/const5		rs1		rd		z	creg		

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	4Byte	.M	010	10100
rs1	4Byte	.M		
rs2	4Byte			
Rd	4Byte	.D	001	00100
rs1	4Byte	.D		
rs2	4Byte			

这条指令的作用是将寄存器 rs1 中的 Byte2, Byte0 与寄存器 rs2 中的 Byte2, Byte0 数据分别存放在目标寄存器 rd 的 Byte3, Byte2, Byte1, Byte0 中。

a 3	a 2	a 1	a 0

b 3	b 2	b 1	b 0

a 2	a 0	b 2	b 0

```
If (cond)    rd_Byte3 <= rs1_Byte2;
            rd_Byte2 <= rs1_Byte0;
            rd_Byte1 <= rs2_Byte2;
            rd_Byte0 <= rs2_Byte0;
```

Elsenop

B. cycle

Bit Permutation Operations

Pack High Bytes of Four Half-Words Into Packed 8-Bit

PACK.H.4

```
.type = .M,.M,.D
```

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5			rs1		rd		z	creg	

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2/const5		rs1		rd			z	creg	

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	4Byte	.M	010	10101
rs1	4Byte	.M		
rs2	4Byte			
Rd	4Byte	.D	001	00101
rs1	4Byte	.D		
rs2	4Byte			

这条指令的作用是将寄存器 rs1 中 Byte3,Byte1 分别存放在目标寄存器 rd 的 Byte3,Byte2,Byte1,Byte0 中

a 3	a 2	a 1	a 0

b 3	b 2	b 1	b 0

Rd

a 3	a 1	b 3	b 1

```
If (cond)    rd_Byte3 <= rs1_Byte3;
            rd_Byte2 <= rs1_Byte1;
            rd_Byte1 <= rs2_Byte3;
            rd_Byte0 <= rs2_Byte1;
```

Elsenop

B. cycle

Bit Permutation Operations

UNPK.2

Unpack Low Signed Packed 16-Bit to
Signed Packed 32-Bit

Bit Permutation Operations

UNPK.2

General Form **UNPK.2** (.type) rd, rs /* (D) */
 .type = .M,.M

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPK.2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	sllong	.M	011 1111 01000
rs1	2hws	.M	

Description

这条指令的作用是将寄存器 rs1 中的低 16 位数据经过符号扩展后, 存放在目标寄存器 rd 的中, 将寄存器 rs1 的高 16 位数据经过符号扩展, 存放在目标寄存器 rd+1 中。

Rs

a_1	a_0

UNPK.2

Rd+1

a_1

Rd

a_0

Operation

If (cond) rd_Byte3 <= 8'b0 ;
 rd_Byte2 <= rs_Byte1;
 rd_Byte1 <= 8'b0;
 rd_Byte0 <= rs_Byte0;

Else nop

Execute time

B. cycle

Bit Permutation Operations

UNPK.L.4

Unpack Low Unsigned Packed 8-Bit to
Unsigned Packed 16-Bit

Bit Permutation Operations

UNPK.L.4

General Form **UNPK.L.4** (.type) rd, rs /* (F) */
UNPK.L.4 (.type) rd, rs /* (D) */
.type = .A, .M,.M,.D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

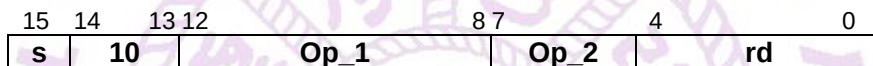
Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

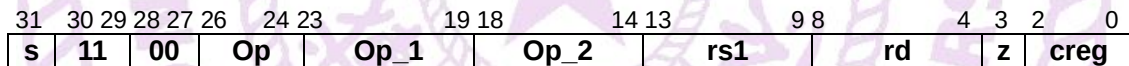
B. Format

DT.F2

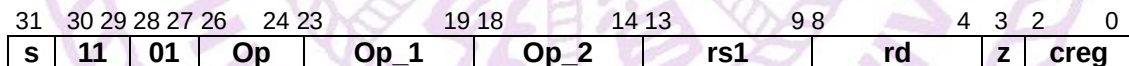


(D) Format

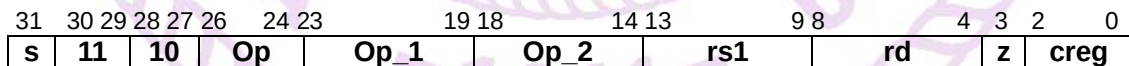
AL.D2



MA.D2



DT.D2



UPACK.L.4

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	4Byte	.D	10111 110
rs1	4Byte	.D	

UPACK.L.4

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	4Byte	.A	011 11111 10100
rs1	4Byte	.A	
Rd	4Byte	.M	011 11111 00110
rs1	4Byte	.M	
Rd	4Byte	.D	001 01000 11100
rs1	4Byte	.D	

Description

这条指令的作用是将寄存器 `rs1` 中的低 16 位数据:Byte1,Byte0 存放在目标寄存器 `rd` 的 Byte2,Byte0 中, 并且将 `rd` 的 Byte3,Byte1 置零。

Rs

a_3	a_2	a_1	a_0

UNPK.L.4

Rd

00000000	a_1	00000000	a_0

Operation

If (cond) `rd_Byte3 <= 8'b0 ;`
 `rd_Byte2 <= rs_Byte1;`
 `rd_Byte1 <= 8'b0;`
 `rd_Byte0 <= rs_Byte0;`

Elsenop

Execute time

B. cycle

Bit Permutation Operations

UNPK.H.4

Unpack High Unsigned Packed 8-Bit to
Unsigned Packed 16-Bit

Bit Permutation Operations

UNPK.H.4

General Form **UNPK.H.4** (.type) rd, rs /* (F) */
 UNPK.H.4 (.type) rd, rs /* (D) */
 .type = .A, .M,.M,.D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

DT.F2

15	14	13	12	8	7	4	0
s	10	Op_1				Op_2	rd

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				Op_2				rs1		rd		z	creg

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				Op_2				rs1	rd	z	creg		

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				Op 2				rs1		rd		z	creg

UPACK.H.4

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	4Byte	.D	10111	111
rs1	4Byte	.D		

UPACK.H.4

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
Rd	4Byte	.A	011	11111	10101
rs1	4Byte	.A			
Rd	4Byte	.M	011	11111	00111
rs1	4Byte	.M			
Rd	4Byte	.D	001	01000	11101
rs1	4Byte	.D			

Description

这条指令的作用是将寄存器 `rs1` 中的低 16 位数据:Byte3,Byte2 存放在目标寄存器 `rd` 的 Byte2,Byte0 中, 并且将 `rd` 的 Byte3,Byte1 置零。

Rs

a_3	a_2	a_1	a_0

UNPK.H.4

Rd

00000000	a_3	00000000	a_2

Operation

If (cond) `rd_Byte3 <= 8'b0 ;`
 `rd_Byte2 <= rs_Byte3;`
 `rd_Byte1 <= 8'b0;`
 `rd_Byte0 <= rs_Byte2;`

Else nop

Execute time

B. cycle

Bit Permutation Operations

Saturate and Pack into Signed Packed 16-Bit

SPACK.2

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5		rs1		rd			z	creg	

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				rs2/const5		rs1		rd			z	creg	

SPACK.2

(D) Format

Opcode used ...	map	field	For operand type ...	Unit	Opfield
Rd			2hws	.M	010 10111
rs1			sint	.M	
rs2			sint		

Description

这条指令的作用是将寄存器 **rs1** 中的 32 位数据与寄存器 **rs2** 中的 32 位数据分别经饱和处理后存放在目标寄存器 **rd** 的高 16 位和低 16 位中。

Rs1

00000000ABCDEFGH	IJKLMNOPQRSTUVWXYZ

Rs2

0000000000000000	00YZ123456789ABC

SPACK.2

Rd

0111111111111111	00YZ123456789ABC

Operation

```

If (cond)      rd_msb16 <= sat(rs1); rd_lsb16 <= sat(rs2);
Elsenop

```

Execute time

B. cycle

Bit Permutation Operations

SPACKU.4

Saturate and Pack into Unsigned Packed
8-Bit

Bit Permutation Operations

SPACKU.4

General Form **SPACKU.4** (.type) rd, rs1, rs2 /* (D) */
 .type = .M,.M

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SPACKU.4

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	4bytes	.M	010	11000
rs1	2hws	.M		
rs2	2hws			

Description

这条指令的作用是将寄存器 rs1 中的高 16 位数据进行无符号 8 位饱和处理后放在寄存器 rd 的高 16 位的高 8 位中；rs1 中的低 16 位数据进行无符号 8 位饱和处理后放在寄存器 rd 的高 16 位的低 8 位中；将寄存器 rs2 中的高 16 位数据进行无符号 8 位饱和处理后放在寄存器 rd 的低 16 位的高 8 位中；rs2 中的低 16 位数据进行无符号 8 位饱和处理后放在寄存器 rd 的低 16 位的低 8 位中

Rs1

00000000ABCDEFGH	00000001IJKLMNOP

Rs2

00000000YZ123456	11111111QRSTUVWXYZ

SPACKU.4

Rd

ABCDEFGH	11111111	YZ123456	00000000

Operation

```
if (cond) {  
    if rs1_msb16 > 0x00FF,  
    then rd_byte3 = 0xFF  
    if rs1_msb16 < 0,  
    then rd_byte3 = 0  
    else  
    then rd_byte3 = rs1_byte2  
  
    if rs1_lsb16 > 0x00FF,  
    then rd_byte2 = 0xFF  
    if rs1_lsb16 < 0,  
    then rd_byte2 = 0  
    else  
    then rd_byte2 = rs1_byte0  
  
    if rs2_msb16 > 0x00FF,  
    then rd_byte1 = 0xFF  
    if rs2_msb16 < 0,  
    then rd_byte1 = 0  
    else  
    then rd_byte1 = rs2_byte2  
  
    if rs2_lsb16 > 0x00FF,  
    then rd_byte0 = 0xFF  
    if rs2_lsb16 < 0,  
    then rd_byte0 = 0  
    else  
    then rd_byte0 = rs2_byte0  
}  
else nop
```

Execute time

B. cycle

Bit Permutation Operations

SWAP.2

Swap Half-Word

Bit Permutation Operations

SWAP.2

General Form **SWAP.2** (.type) rd /* (F) */
SWAP.2 (.type) rd, rs /* (D) */
.type = .M, .M, .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

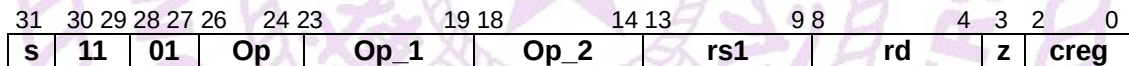
B. Format

DT.F2

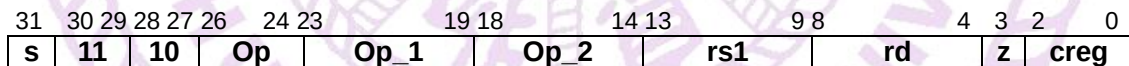


(D) Format

MA.D2



DT.D2



SWAP.2

(F) Format

Opcode used ...	map	field	For operand type ...	Unit	Opfield
Rd			2hws	.D	10111 000
				.D	

SWAP.2

(D) Format

Rd	2hws	.M	011	11111	00000
Rs	2hws	.M			
Rd	2hws	.D	001	01000	11110
Rs	2hws	.D			

Description

这条指令的作用是将寄存器 **rs** 中的低 16 位数据与高 16 位数据分别存放在目标寄存器 **rd** 的高 16 位和低 16 位中。

Rs1

Rs_msb16

Rs_lsb16

SWAP.2

Rd

Rs_lsb16

Rs_msb16

Operation

(F) rd_msb16 <= rd_lsb16; rd_lsb16 <= rd_msb16;

(D) If (cond) rd_msb16 <= rs_lsb16; rd_lsb16 <= rs_msb16;

Elsenop

Execute time

B. cycle



Bit Permutation Operations

SWAP.4

Swap Bytes in Each Half-Word

Bit Permutation Operations

SWAP.4

General Form **SWAP.4** (.type) rd /* (F) */
SWAP.4 (.type) rd, rs /* (D) */
.type = .M, .M, .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

DT.F2

15	14	13	12	8	7	4	0
s	10	Op_1	Op_2	rd			

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs1	rd	z	creg								

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	rs1	rd	z	creg								

SWAP.4

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	4Bytes	.D	10111 001
		.D	

SWAP.4

(D) Format

Rd	4Bytes	.M	011	11111	00001
Rs	4Bytes	.M			
Rd	4Bytes	.D	001	01000	11111
Rs	4Bytes	.D			

Description

这条指令的作用是将寄存器 **rs** 的低 16 位数据中的两个 **Byte** 互换后存在 **rd** 的低 16 位中，同时将 **rs** 高 16 位数据中的两个 **Byte** 互换后存放在目标寄存器 **rd** 的高 16 位中。

Rs

ub_3	ub_2	ub_1	ub_0

SWAP.4

Rd

ub_2	ub_3	ub_0	ub_1

Operation

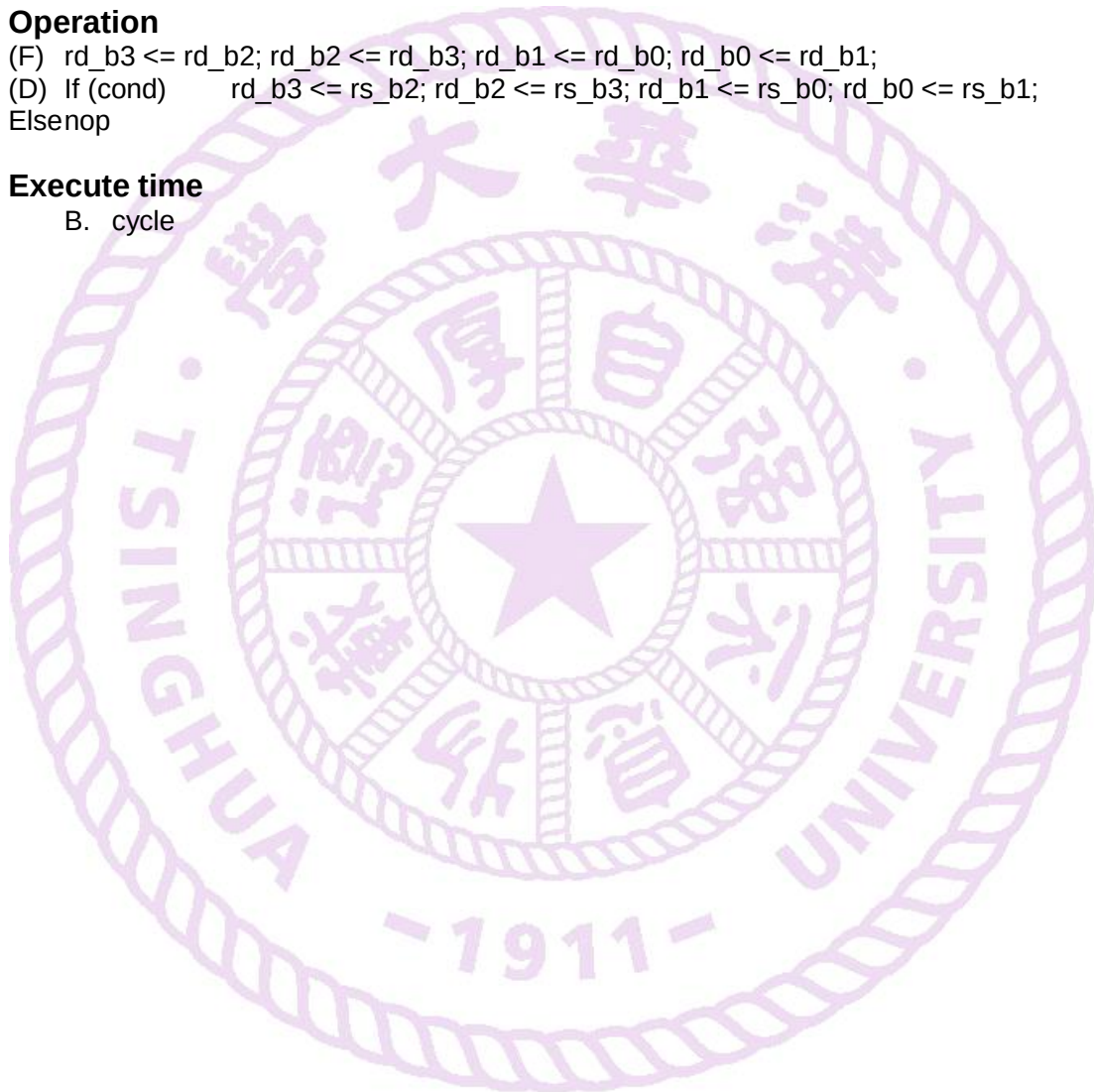
(F) rd_b3 <= rd_b2; rd_b2 <= rd_b3; rd_b1 <= rd_b0; rd_b0 <= rd_b1;

(D) If (cond) rd_b3 <= rs_b2; rd_b2 <= rs_b3; rd_b1 <= rs_b0; rd_b0 <= rs_b1;

Elsenop

Execute time

B. cycle



Bit Permutation Operations

SHLMB1/2/3

Shift and Merge Bytes

Bit Permutation Operations

SHLMB1/2/3

General Form **SHLMB1** (.type) rd, rs1, rs2 /* (D) */
 SHLMB2 (.type) rd, rs1, rs2 /* (D) */
 SHLMB3 (.type) rd, rs1, rs2 /* (D) */
 .type = .M, .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2/const5	rs1	rd	z	creg								

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SHLMB1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	int	.M	010 11100
rs1	Int	.M	
rs2	Int		
Rd	int	.D	001 01100
rs1	Int	.D	
rs2	Int		

SHLMB2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	int	.M	010 11101
rs1	Int	.M	
rs2	Int		
Rd	int	.D	001 01101
rs1	Int	.D	
rs2	Int		

SHLMB3**(D) Format**

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	int	.M	010	11110
rs1	Int	.M		
rs2	Int			
Rd	int	.D	001	01110
rs1	Int	.D		
rs2	Int			

Description

这条指令的作用是将寄存器 rs1 与寄存器 rs2 中的 8 个 Byte 做相应的处理并合并，将结果存放在 rd 中。

Rs1

ua_3	ua_2	ua_1	ua_0

Rs2

ub_3	ub_2	ub_1	ub_0

SHLMB1

Rd

ua_0	ub_3	ub_2	ub_1

SHLMB2

Rd

ua_1	ua_0	ub_3	ub_2

SHLMB3

Rd

ua_2	ua_1	ua_0	ub_3

Operation

SHLMB1

If (cond) rd <= {ua_0 : ub_3: ub_2: ub_1};

Elsenop

SHLMB2

If (cond) rd <= {ua_1 : ua_0: ub_3: ub_2};

Elsenop

SHLMB3

If (cond) rd <= {ua_2 : ua_1: ua_0: ub_3};

Elsenop

Execute time

B. cycle

Bit Permutation Operations

EXP.L/.H

Expand Low or High Halfword to Packed
16-Bit

Bit Permutation Operations

EXP.L/.H

General Form

EXP.L (.type) rd /* (F) */
EXP.L (.type) rd, rs /* (D) */
EXP.H (.type) rd /* (F) */
EXP.H (.type) rd, rs /* (D) */
.type = .M, .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

DT.F2

15	14	13	12	8	7	4	0
s	10	Op_1			Op_2		rd

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				Op_2		rs1			rd		z	creg	

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				Op_2		rs1			rd		z	creg	

EXP.L

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	2hws	.D	10111 010
		.D	

EXP.L

(D) Format

Rd	2hws	.M	011	11111	00010
Rs	2hws	.M			
Rd	2hws	.D	001	01000	01010
Rs	2hws	.D			

EXP.H	(F) Format			
Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	2hws	.D	10111	011
		.D		

EXP.H	(D) Format				
Rd	2hws	.M	011	11111	00011
Rs	2hws	.M			
Rd	2hws	.D	001	01000	01011
Rs	2hws	.D			

Description

这组指令的作用是将寄存器 **rs** 中的低 16 位数据或高 16 位进行扩展，结果存放在目标寄存器 **rd** 的高 16 位和低 16 位中。

Rs

Rs_msb16	Rs_lsb16

EXP.L

Rd

Rs_lsb16	Rs_lsb16

EXP.H

Rd

Rs_msb16	Rs_msb16

Operation

EXP.L

(F) $rd\_msb16 \leq rd\_lsb16; rd\_lsb16 \leq rd\_lsb16;$

(D) If (cond) $rd\_msb16 \leq rs\_lsb16; rd\_lsb16 \leq rs\_lsb16;$

Elsenop

EXP.H

(F) $rd\_msb16 \leq rd\_msb16; rd\_lsb16 \leq rd\_msb16;$

(D) If (cond) $rd\_msb16 \leq rs\_msb16; rd\_lsb16 \leq rs\_msb16;$

Elsenop

Execute time

B. cycle

Bit Permutation Operations

EXPB.L/.H

Expand Low or High Bytes to Packed
8-Bit

Bit Permutation Operations

EXPB.L/.H

General Form

EXPB.L (.type) rd /* (F) */

EXPB.L (.type) rd, rs /* (D) */

EXPB.H (.type) rd /* (F) */

EXPB.H (.type) rd, rs /* (D) */

.type = .M, .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

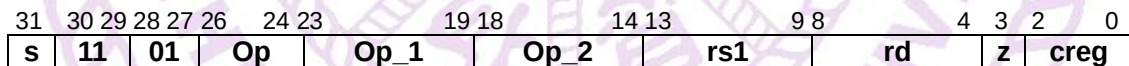
B. Format

DT.F2

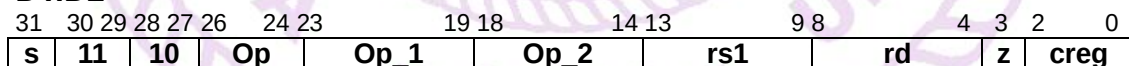


(D) Format

MA.D2



DT.D2



EXPB.L

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	4bytes	.D	10111 100
		.D	

EXPB.L

(D) Format

Rd	4bytes	.M	011	11111	00100
Rs	4bytes	.M			
Rd	4bytes	.D	001	01000	01100
Rs	4bytes	.D			

EXPB.H	(F) Format			
Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	2hws	.D	10111	101
		.D		

EXPB.H	(D) Format				
Rd	2hws	.M	011	11111	00101
Rs	2hws	.M			
Rd	2hws	.D	001	01000	01101
Rs	2hws	.D			

Description

这组指令的作用是将寄存器 **rs** 中的低 8 位数据或高 8 位进行扩展，结果存放在目标寄存器 **rd** 中。

Rs

ua_3	ua_2	ua_1	ua_0

EXPB.L

Rd

ua_0	ua_0	ua_0	ua_0

EXPB.H

Rd

ua_3	ua_3	ua_3	ua_3

Operation

EXPB.L

(F) $rd\_b3 \leq rd\_b0; rd\_b2 \leq rd\_b0; rd\_b1 \leq rd\_b0; rd\_b0 \leq rd\_b0;$

(D) If (cond) $rd\_b3 \leq rs\_b0; rd\_b2 \leq rs\_b0; rd\_b1 \leq rs\_b0; rd\_b0 \leq rs\_b0;$

Elsenop

EXPB.H

(F) $rd\_b3 \leq rd\_b3; rd\_b2 \leq rd\_b3; rd\_b1 \leq rd\_b3; rd\_b0 \leq rd\_b3;$

(D) If (cond) $rd\_b3 \leq rs\_b3; rd\_b2 \leq rs\_b3; rd\_b1 \leq rs\_b3; rd\_b0 \leq rs\_b3;$

Elsenop

Execute time

B. cycle

Bit Permutation Operations

EXT.S/.U

Signed or Unsigned Extract

Bit Permutation Operations

EXT.S/.U

General Form **EXT.S** (.type) rd, rs1, rs2 /* (D) */
 EXT.U (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

EXT.S

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	int	.A	001 11100
rs1	Int	.A	
rs2	Int		

EXT.U

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	int	.A	001 11101
rs1	Int	.A	
rs2	Int		

Description

这条指令的作用是按照 rs2 中的定义，将寄存器 rs1 中的的特定长度的位码经过符号扩展或者无符号扩展，存放在目标寄存器 rd 中。

Rs1

SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS SSSS

Rs2

xxxxxxxxxxxxxxxxxx	xxxp pppp	xxxL LLLL
31	16 15	8 7 0

rs2 中, bits 15~8 定义了需要 extract 的 bit 的起始位置, bits 7~0 定义了需要 extract 的 bit 的长度

rs2 中的长度和位置的有效范围都是 0 到 32, 如果(长度 + 位置) > 32, 那么, EXT.S 指令, 目标寄存器则为全 1, EXT.U 指令, 目标寄存器则为全 0。

Rs1

32'b 1010 0101 1010 0101 1100 0 100 1010 1010

Rs2

xxxxxxxxxxxxxxxxxxx	0000 0111	0000 0100
31	16 15	8 7 0

EXT.S

Rd

32'b1111 1111 1111 1111 1111 1111 1111 1001

EXT.U

Rd

32'b 0000 0000 0000 0000 0000 0000 0000 1001

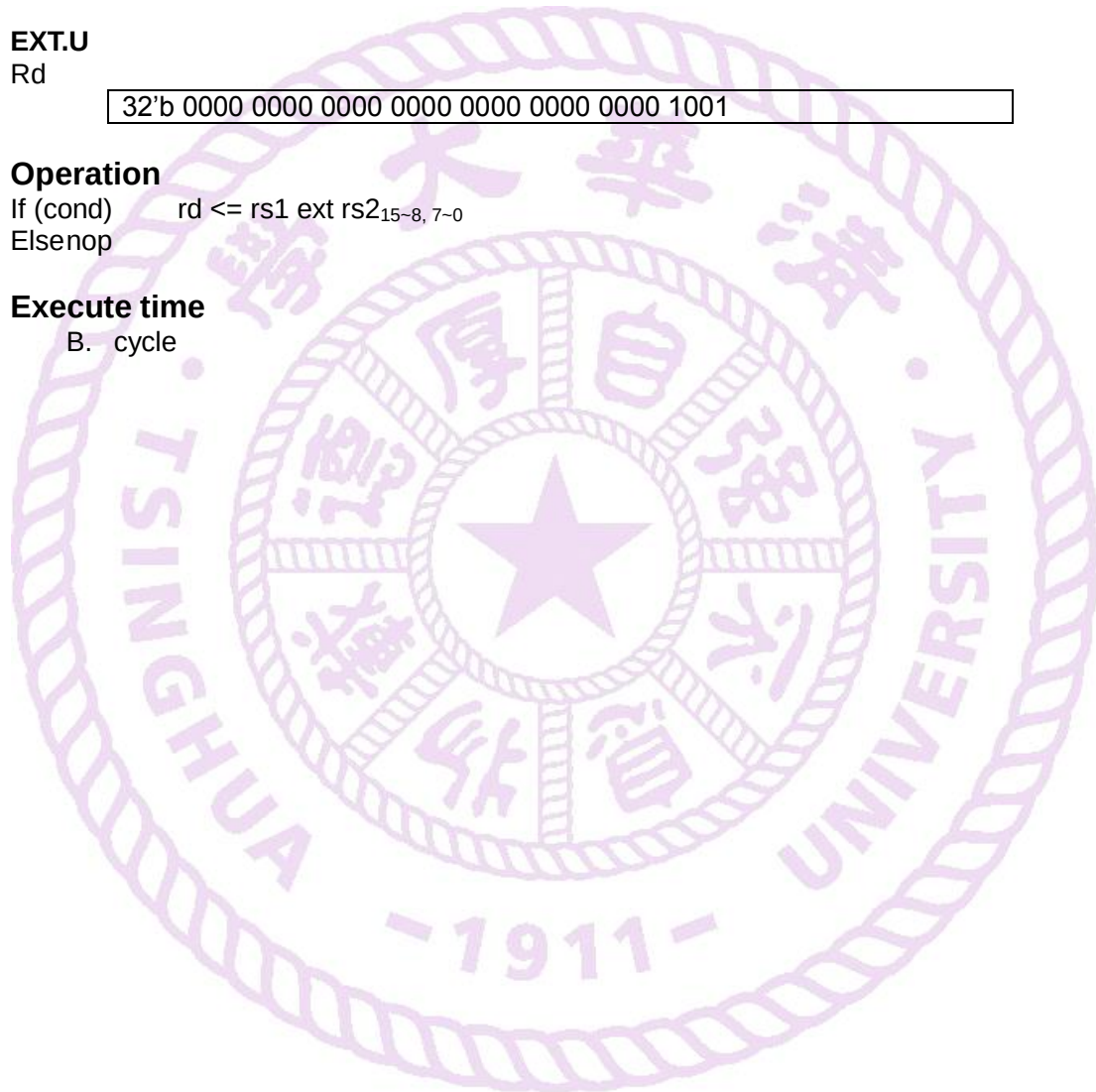
Operation

If (cond) rd <= rs1 ext rs2_{15-8, 7-0}

Elseenop

Execute time

B. cycle



General Form **SXT1** (.type) rd, rs /* (D) */
 .type = .M

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				Op_2				rs1		rd		z	creg

SXT1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	int	.M	011	1111	01011
rs	Int	.M			

Description

这条指令的作用是按照将 rs 的高 24 位按 rs2 第 8 位的值符号扩展，结果放入 rd 中。

Rs

24'bx	saaaaaaaa

Rd

24'bs	saaaaaaa
31	8 7 0

Operation

Execute time

B. cycle

Bit Permutation Operations

ZXT1

32 To 8 Unsigned Bit Length Conversion

ZXT1

General Form	ZXT1 (.type) rd, rs	<i>/* (D) */</i>
	.type = .M	

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		Op_2		rs1		rd		z		creg			

ZXT1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	int	.M	011	1111	01100
rs	Int	.M			

Description

这条指令的作用是按照将 rs 的高 24 位置 0，结果放入 rd 中。

Rs

24'bx	saaaaaaaa

Rd

24'b0	saiaaaaa

Operation

Execute time

B. cycle

SXT2

Bit Permutation Operations

SXT2

32 To 16 Signed Bit Length Conversion

General Form **SXT2 (.type) rd, rs** **/* (D) */**
.type = .M

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				Op_2				rs1		rd		z	creg

SXT2

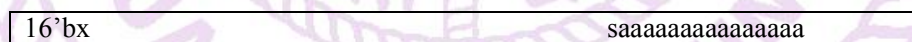
(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	int	.M	011	1111	01101
rs	Int	.M			

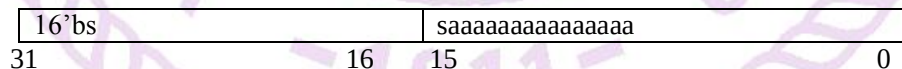
Description

这条指令的作用是按照将 rs 的高 16 位按 rs 第 16 位的值符号扩展，结果放入 rd 中。

Rs



Rd



Operation

Execute time

B. cycle

ZXT2**ZXT2**

32 To 16 Unsigned Bit Length Conversion

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				Op_2				rs1		rd		z	creg

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	int	.M	011	1111	01101
rs	Int	.M			

这条指令的作用是按照将 rs 的高 16 清零，结果放入 rd 中。

16'bx	saaaaaaaaaaaaaaaa

16'b0																saaaaaaaaaaaaaaaaa															
31																0															

Execute time

B. cycle

SXT8

SXT8

```
SXT2 (.type) rd, rs      /* (D) */
.type = .M
```

Syntax

Instruction Length

Opcode

MA.D1

SXT8

Opcode map field used ...

Unit

rd

int

.M

011

11111

01101

rs

Int

.M

这条指令的作用是按照将 rd+1 按 rs 第 32 位的值符号扩展，rd 值不变。

Saaaaaaaaaaaaaaaaaaaaaaaaaaaaa

32'bs

31

0

Execute time

223

Bit Permutation Operations

NORM Normalize Integer

Bit Permutation Operations

NORM

General Form **NORM** (.type) rd, rs /* (D) */
 .type = .A, .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	rs1	rd	z	creg								

NORM

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	int	.A	011	11111	11000
rs	Int	.A			
rd	int	.D	001	01000	00000
rs	Int	.D			

Description

这条指令的作用是提取源操作数 rs 中的冗余位,并将冗余位的数量存放在目标寄存器 rd 中。

Rs

32'b01xx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx

NORM 返回 0

rs

32'b0000_1xxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx_xxxx

NORM 返回 3

rs

32'b1111_1111_1111_1111_1111_1111_1111_1110

NORM 返回 30

rs

32'b1111_1111_1111_1111_1111_1111_1111_1111

NORM 返回 31

rs

32'b0000_0000_0000_0000_0000_0000_0000_0000

NORM 返回 31

Operation

If (cond) rd <= norm rs
Else nop

Execute time

B. cycle



Bit Permutation Operations

NORM.L

Normalize 16-bit Integer

Bit Permutation Operations

NORM.L

General Form **NORM.L** (.type) rd, rs /* (D) */
 .type = .A,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

NORM

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	int	.A	011 1111 11001
rs	Int	.A	

Description

这条指令的作用是提取源操作数 rs 的低 16 位中的冗余位, 并将冗余位的数量存放在目标寄存器 rd 中。

Rs

32'bxxxx xxxx xxxx xxxx 01xx xxxx xxxx xxxx

NORM 返回 0

rs

32'bxxxx xxxx xxxx xxxx 0000 1xxx xxxx xxxx

NORM 返回 3

rs

32'bxxxx xxxx xxxx xxxx 1111 1111 1111 1110

NORM 返回 14

rs

32'bxxxx xxxx xxxx xxxx 1111 1111 1111 1111

NORM 返回 15

rs

32' bxxxx xxxx xxxx xxxx 0000 0000 0000 0000

NORM 返回 15

Operation

If (cond) rd = norm rs_hw1s
Else nop

Execute time

B. cycle



Bit Permutation Operations

LMBD0/1

Leftmost Bit 0 or 1 Detection

Bit Permutation Operations

LMBD0/1

General Form **LMBD0** (.type) rd, rs /* (D) */
LMBD1 (.type) rd, rs /* (D) */
.type = .A, .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	rs1	rd	z	creg								

LMBD0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield				
rd	int	.A	011	11111	11100		
rs	Int	.A					
rd	int	.D	001	01000	00100		
rs	Int	.D					

LMBD1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield				
rd	int	.A	011	11111	11101		
rs	Int	.A					
rd	int	.D	001	01000	00101		
rs	Int	.D					

Description

这条指令的作用是检测源操作数 rs 中的最左端的 0 或 1 所在的位置，并将检测的位置存放在目标寄存器 rd 中。

Rs

32'b01xx xxxx xxxx xxxx xxxx xxxx xxxx

LMBD0 返回 0

LMBD1 返回 1

rs

32'b0000 1xxx xxxx xxxx xxxx xxxx xxxx

LMBD0 返回 0

LMBD1 返回 4

rs

32'b1111 1111 1111 1111 1111 1111 1111 1110

LMBD0 返回 31

LMBD1 返回 0

rs

32'b1111 1111 1111 1111 1111 1111 1111 1111

LMBD0 返回 32

LMBD1 返回 0

Operation

If (cond) rd <= lmbd0/1 rs

Else nop

Execute time

B. cycle

DEAL

De-Interleave and Pack

Bit Permutation Operations

DEAL

General Form **DEAL** (.type) rd, rs /* (D) */
 .type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op 1				Op 2			rs1		rd		z	creg	

DEAL

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	int	.D	001 01000 00010
Rs	Int	.D	

Description

这条指令的实现功能如下:

rs

aAbBcCdDeEfGgHhIiJjKkLlMmNnOoPp

DEAL

rd

abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

Operation

```
If (cond)      rd <= deal rs
```

```

Else      nop

```

Execute time

B. cycle

Bit Permutation Operations
SHFL Shuffle

Bit Permutation Operations
SHFL

General Form **SHFL** (.type) rd, rs /* (D) */
 .type = .D

Syntax
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length
 In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	rs1	rd	z	creg								

SHFL	(D) Format
Opcode map field used ...	For operand type ... Unit Opfield
Rd	int .D 001 01000 00011
Rs	Int .D

Description

这条指令的实现功能如下:

rs

abcdefghijklmnoABCDEFGHIJKLMN

SHFL

rd

aAbBcCdDeEfGhHilJkKlLmMnNoOpP

Operation

If (cond) rd <= shfl rs
Else nop

Execute time

B. cycle

Bit Permutation Operations

ZERO Set all ZERO's

ZERO

General Form **ZERO (.type) rd** */* (F) */*
 .type = .D

Syntax

(D) rd: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

ZERO

15	14	13	12	8	7	4	0
s	10	10110		111		rd	

Description

这条指令的作用是将目标寄存器 rd 设为全零。

Operation

```
rd <= 32'b0
```

Execute time

B. cycle

Bit Permutation Operations

PACKG

Pack Register Pair to G

PACKG

```
General Form   PACKG (.type) Gd, Rs_Pair1, Rs_Pair2    /* (D) */
                 .type = .D
```

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				Rs_Pair1		Rs_Pair2		Gd		z	creg		

LMBD0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Gd	int	.D	001	01000
Rs_Pair1	Int	.D		
Rs_Pair2	int	.D		

Description

将 Rs_Pair1 中的值放到 Gd 的高 64 位，把 Rs_Pair2 中的值放到 Gd 的低 64 位。

Operation

If (cond) Gd = [Rs_Pair1 : Rs_Pair2]

```
Else      nop
```

Execute time

B. cycle

Bit Permutation Operations

PACKGL2

Pack Register G

PACKGL2

General Form **PACKGL2** (.type) Gd, Gs1,Gs2 /* (D) */
 .type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				Gs1		Gs2		Gd		z	creg		

LMBD0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Gd	int	.D	001	01001
Gs1	Int	.D		
Gs2	int	.D		

Description

将 Gs1 中的低 64 位放到 Gd 的高 64 位，把 Gs2 中的低 64 位放到 Gd 的低 64 位。

Operation

```

If (cond)      Gd = [lsb(Gs1) :lsb(Gs2)]
Else          nop

```

Execute time

B. cycle

Bit Permutation Operations

PACKGH2

Pack Register G

Bit Permutation Operations

PACKGH2

General Form

PACKGH2 (.type) Gd, Gs1,Gs2

/* (D) */

.type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;

comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31

30

29

28

27

26

24

23

19

18

14

13

9

8

4

3

2

0

s

11

10

Op

Op_1

Gs1

Gs2

Gd

z

creg

LMBD0

(D) Format

Opcode map field used ...

For operand type ...

Unit

Opfield

Gd

int

.D

001

01010

Gs1

Int

.D

Gs2

int

.D

Description

将 Gs1 中的高 64 位放到 Gd 的高 64 位，把 Gs2 中的高 64 位放到 Gd 的低 64 位。

Operation

If (cond) Gd = [msb(Gs1) :msb(Gs2)]

Else nop

Execute time

B. cycle

Bit Permutation Operations UNPACKGH2 Unpack Register G

Bit Permutation Operations UNPACKGH2

General Form **UNPACKGH2** (.type) Gd, Gs /* (D) */
 .type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	Gs	Gd	z	creg								

LMBD0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Gd	int	.D	001 01000 01110
Gs	Int	.D	

Description

将 Gs 中的高 64 位分别放到 Gd 的高 64 位和低 64 位。

Operation

If (cond) Gd = [msb(Gs) :msb(Gs)]
Else nop

Execute time

- B. cycle
- C.

Bit Permutation Operations
UNPACKGL2 Unpack Register G

Bit Permutation Operations
UNPACKGL2

General Form **UNPACKGL2** (.type) Gd, Gs /* (D) */
 .type = .D

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	Gs	Gd	z	creg								

LMBD0		(D) Format			
Opcode map field used ...		For operand type ...		Unit	Opfield
Gd		int		.D	001 01000 01111
Gs		Int		.D	

Description
将 Gs 中的低 64 位分别放到 Gd 的高 64 位和低 64 位。

Operation
If (cond) Gd = [lsb(Gs) :lsb(Gs)]
Else nop

Execute time
B. cycle

Bit Permutation Operations
UNPACKGW0/1/2/3

Unpack Register G to Register

Bit Permutation Operations
UNPACKGW0/1/2/3

General Form
UNPACKGW0/1/2/3 (.type) Rd, Gs
/* (D) */
.type = .D

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	Gs	Rd	z	creg								

LMBD0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	int	.D	001 01000 10101
Gs	Int	.D	
Rd	Int	.D	001 01000 10110
Gs	Int	.D	
Rd	Int	.D	001 01000 10111
Gs	Int	.D	
Rd	Int	.D	001 01000 11000
Gs	int	.D	

Description
根据 Op_2，将 Gs 中的某一段 32-bit 放入到寄存器 Rs 中。

Operation
If (cond) Rd = Gs[127: 96]
Else nop
If (cond) Rd = Gs[95: 64]
Else nop
If (cond) Rd = Gs[63: 32]
Else nop
If (cond) Rd = Gs[31: 0]
Else nop

Execute time
B. cycle

Bit Permutation Operations

UNPACKGD0/1 Unpack Register G to Register Pair

Bit Permutation Operations

UNPACKGD0/1

General Form **UNPACKGD0/1** (.type) Rd_Pair, Gs /* (D) */
 .type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	Gs	Rd_Pair	z	creg								

LMBD0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield		
Rd_Pair	int	.D	001	01000	11001
Gs	Int	.D			
Rd_Pair	int	.D	001	01000	11010
Gs	Int	.D			

Description

根据 Op_2, 将 Gs 的高 64 位或者低 64 位放到寄存器对 Rd_Pair 中。

Operation

If (cond) Rd = Gs[127:64]
 Else nop
 If (cond) Rd = Gs[63:0]
 Else nop

Execute time

B. cycle

14 FLOW CONTROL OPERATIONS

Instruction Summary

- B/.Z/.NZ
- BD/.Z/.NZ
- CALL/.Z/.NZ
- RET
- RETD
- IRET
- IRETD
- NOP
- NOPN
- PAUSE
- IDLE
- SLEEP
- LPSETUP
- ING



Flow Control Operations

B/.Z/.NZ Branch

Flow Control Operations

B/.Z/.NZ

General Form

B (.type) rs /* (F) */
B.Z (.type) rs /* (F) */
B.NZ (.type) rs /* (F) */
B (.type) rs /* (D) */
B (.type) lable /* (D) */
.type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

DT.F2

15	14	13	12	8	7	4	0
s	10	Op_1	Op_2	rs			

(D) Format

DT.D4

31	30	29	28	27	26	24	23	4	3	2	0
s	11	11	Op	cst20	z	creg					

DT.D5

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	Op_2	rs	xxxxx	z	creg								

B (F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs	sint	.D	11111 000
B.Z	(F) Format	.D	11111 001
rs	sint	.D	11111 001
B.NZ	(F) Format	.D	11111 010
rs	sint	.D	11111 010

B (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs	sint	.D	000 11111 01000

		.D	
lable	scst20	.D	000
		.D	

Description

B.Z 是指当条件寄存器 G0 为 0 的时候执行；B.NZ 是指当条件寄存器 G0 为非 0 的时候执行。

根据目标寄存器或偏移量，产生新的执行程序的地址并将其存放于程序计数器 PC，同时清除已经进入流水线的指令。

如果采用寄存器的方式，则将寄存器中的值直接导入程序计数器 PC；如果采用偏移量的方式，则将 $\text{const20} \ll 1 + \text{PCE1}$ 所得结果导入 PC。PCE1 是包含跳转指令的取指包中第一条指令的地址。

跳转指令的执行顺序如下图所示

Cycle0	执行跳转指令，产生新的 PC
Cycle1	
Cycle2	
Cycle3	
Cycle4	
Cycle5	
Cycle6	
Cycle7	执行新的指令

Operation

(F) if (! G0) PC <= rd

(D) If (cond) {PC <= rd or $\text{const20} \ll 1 + \text{PCE1}$ }

Elsenop

Execute time

7 cycle

Flow Control Operations

BD/.Z/.NZ

Branch Delay

Flow Control Operations

BD/.Z/.NZ

General Form

BD (.type) rs	/* (F) */
BD.Z (.type) rs	/* (F) */
BD.NZ (.type) rs	/* (F) */
BD (.type) rs	/* (D) */
BD (.type) lable	/* (D) */

.type = **.D**

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

DT.F2



(D) Format

DT.D4



DT.D5



BD

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs	sint	.D	11111 100
		.D	
BD.Z	(F) Format		
rs	sint	.D	11111 101
		.D	
BD.NZ	(F) Format		
rs	sint	.D	11111 110
		.D	

BD		(D) Format		
Opcode map field used ...	For operand type ...	Unit	Opfield	
rs	sint	.D	000	11111 01001
		.D		
lable	scst20	.D	001	
		.D		

Description

BD.Z 是指当条件寄存器 G0 为 0 的时候执行；BD.NZ 是指当条件寄存器 G0 为非 0 的时候执行。

根据目标寄存器或偏移量，产生新的执行程序地址并将其存放于程序计数器 PC，同时继续执行已经进入流水线的指令。

如果采用寄存器的方式，则将寄存器中的值直接导入程序计数器 PC；如果采用偏移量的方式，则将 $\text{const20} \ll 1 + \text{PCE1}$ 所得结果导入 PC。PCE1 是包含跳转指令的取指包中第一条指令的地址。

延迟跳转指令的执行顺序如下图所示

Cycle0	执行延迟跳转指令，产生新的 PC
Cycle1	继续执行指令 0
Cycle2	继续执行指令 1
Cycle3	继续执行指令 2
Cycle4	继续执行指令 3
Cycle5	继续执行指令 4
Cycle6	继续执行指令 5
Cycle7	执行跳转目标指令

Operation

(F) if (Z) $\text{PC} \leq \text{rd}$

(D) If (cond) $\{\text{PC} \leq \text{rd or } \text{const20} \ll 1 + \text{PCE1}\}$

Elsenop

Execute time

7 cycle

Flow Control Operations

CALL/.Z/.NZ Call

Flow Control Operations

CALL/.Z/.NZ

General Form

CALL (.type) rs /* (F) */

CALL.Z (.type) rs /* (F) */

CALL.NZ (.type) rs /* (F) */

CALL (.type) rs /* (D) */

CALL (.type) lable /* (D) */

.type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

DT.F2



DT.D4



DT.D5



CALL (F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs	sint	.D	11110 100
		.D	

CALL.Z (F) Format

rd	sint	.D	11110 101
		.D	

CALL.NZ (F) Format

rs	sint	.D	11110 110
		.D	

CALL (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs	sint	.D	000 1111 01010
		.D	
lable	scst20	.D	010
		.D	

Description

CALL.Z 是指当条件寄存器 **G0** 为 0 的时候执行；**CALL.NZ** 是指当条件寄存器 **G0** 为非 0 的时候执行。

根据目标寄存器或偏移量，产生新的执行程序的地址并将其存放于程序计数器 **PC**，并将当前 **PCE1** 值压栈，同时将 **PCE1** 存放在 **STBUF**。同时继续执行已经进入流水线的指令。

如果采用寄存器的方式，则将寄存器中的值直接导入程序计数器 **PC**；如果采用偏移量的方式，则将 $\text{const20} \ll 1 + \text{PCE1}$ 所得结果导入 **PC**。**PCE1** 是包含跳转指令的取指包中第一条指令的地址。

延迟跳转指令的执行顺序如下图所示

Cycle0	执行函数调用指令，产生新的 PC
Cycle1	
Cycle2	
Cycle3	
Cycle4	
Cycle5	
Cycle6	
Cycle7	执行函数调用目标指令

Operation

(F) if (Z) $\text{PC} = \text{rd}$, push **STBUF**, $\text{STBUF} = \text{PCE1}$

(D) If (cond) $\{\text{PC} = \text{rd or } \text{const20} \ll 1 + \text{PCE1}$

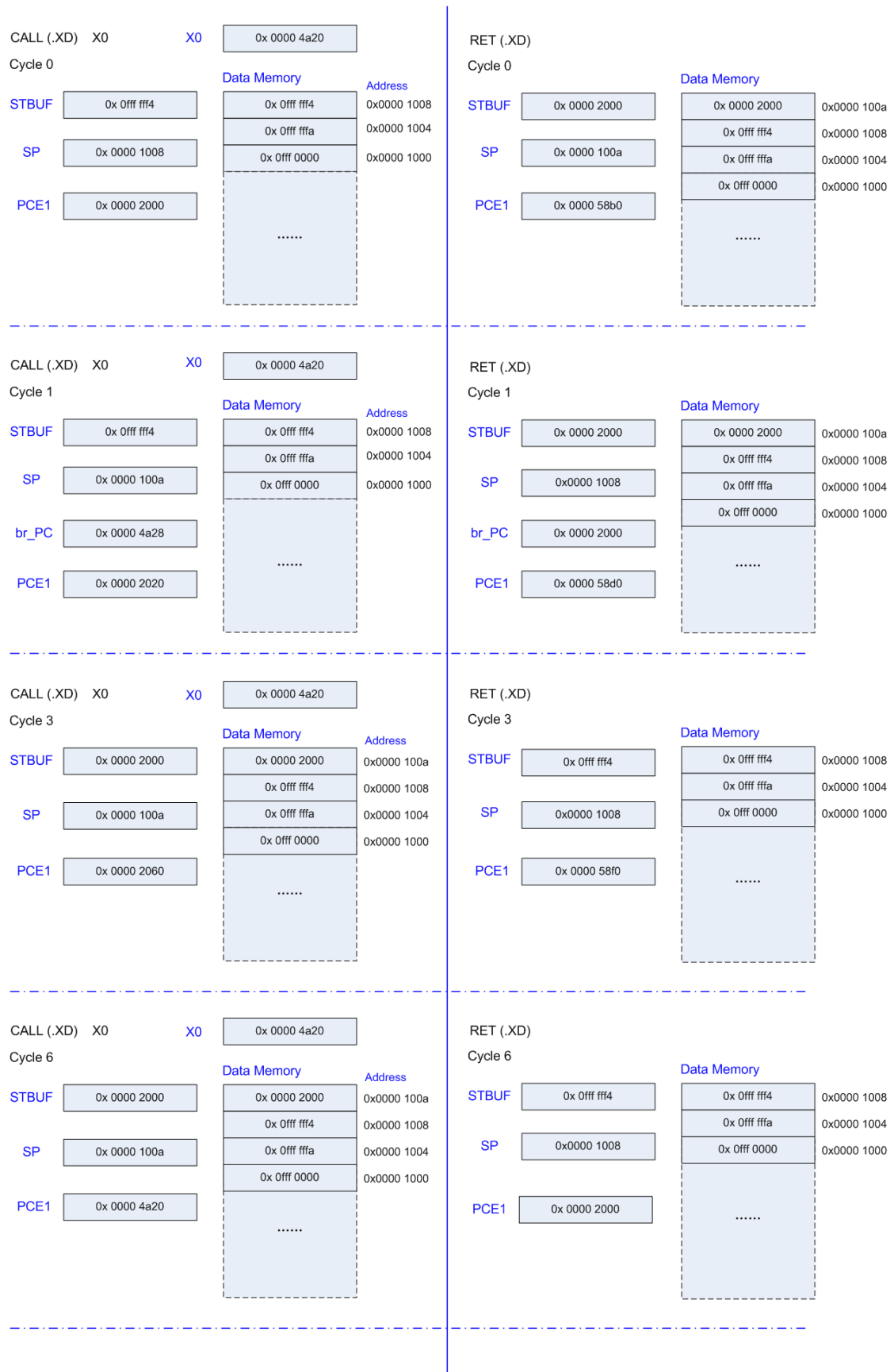
push **STBUF**, $\text{STBUF} = \text{PCE1}\}$

Elsenop

Execute time

7 cycle

Example



Flow Control Operations

CALLD/.Z/.NZ Call Delay

Flow Control Operations

CALLD/.Z/.NZ

General Form

CALLD (.type) rs /* (F) */
CALLD.Z (.type) rs /* (F) */
CALLD.NZ (.type) rs /* (F) */
CALLD (.type) rs /* (D) */
CALLD (.type) lable /* (D) */
.type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(F) Format

DT.F2



(D) Format

DT.D4



DT.D5



CALLD

(F) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rs	sint	.D	11111 100
		.D	
CALLD.Z	(F) Format		
rs	sint	.D	11111 101
		.D	
CALLD.NZ	(F) Format		
rs	sint	.D	11111 110
		.D	

CALLD		(D) Format		
Opcode map field used ...	For operand type ...	Unit	Opfield	
rs	sint	.D	000	11111 01001
		.D		
lable	scst20	.D	001	
		.D		

Description

CALLD.Z 是指当条件寄存器 G0 为 0 的时候执行；CALLD.NZ 是指当条件寄存器 G0 为非 0 的时候执行。

根据目标寄存器或偏移量，产生新的执行程序地址并将其存放于程序计数器 PC，同时继续执行已经进入流水线的指令。

如果采用寄存器的方式，则将寄存器中的值直接导入程序计数器 PC；如果采用偏移量的方式，则将 $\text{const20} \ll 1 + \text{PCE1}$ 所得结果导入 PC。PCE1 是包含跳转指令的取指包中第一条指令的地址。

与 BD 指令采用相同的指令码，而与 BD 指令不同的是 CALLD 指令需要在指令后五个执行包之内存在一条 PUSH.PC 指令，将返回地址压栈。

延迟跳转指令的执行顺序如下图所示

Cycle0	执行延迟跳转指令，产生新的 PC
Cycle1	继续执行指令 0
Cycle2	继续执行指令 1
Cycle3	继续执行指令 2
Cycle4	继续执行指令 3
Cycle5	继续执行指令 4
Cycle6	继续执行指令 5
Cycle7	执行跳转目标指令

Operation

(F) if (Z) PC <= rd

(D) If (cond) {PC <= rd or $\text{const20} \ll 1 + \text{PCE1}$ }

Elsenop

Execute time

7 cycle

Flow Control Operations

PUSH.PC

Push Return PC address

Flow Control Operations

PUSH.PC

General Form **PUSH.PC** (.type) lable /* (D) */
 PUSH.PC (.type) /* (D) */
 .type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D4

31	30	29	28	27	26	24	23					4	3	2	0
s	11	11	Op											z	creg

PUSH.PC

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
lable	scst20	.D .D	000 1

Description

PUSH.PC 指令用于将 CALLD 的返回地址压栈。CALLD 指令与 BD 指令采用相同的指令码,而与 BD 指令不同的是 CALLD 指令需要在指令后五个执行包之内存在一条 PUSH.PC 指令,将返回地址压栈。

PUSH.PC 压栈返回地址有两种方式,一种是不加 lable,汇编器可以自动计算 CALLD 之后第 7 个执行包的地址,计算与 PUSH.PC 指令 PCE1 地址的偏移量提供给处理器。处理器可以根据 PUSH.PC 指令 PCE1 地址与返回地址偏移量计算得到返回地址并将返回地址压栈。第二种方式是加 lable,汇编器计算 lable 所示指令地址,计算与 PUSH.PC 指令 PCE1 地址的偏移量提供给处理器。处理器可以根据 PUSH.PC 指令 PCE1 地址与返回地址偏移量计算得到返回地址并将返回地址压栈。

延迟调用指令的执行顺序如下图所示, PUSH.PC 指令必须出现在 cycle1 到 cycle5 之间

Cycle0	执行延迟调用指令,产生新的 PC
Cycle1	继续执行指令 0
Cycle2	继续执行指令 1
Cycle3	继续执行指令 2
Cycle4	继续执行指令 3
Cycle5	继续执行指令 4
Cycle6	继续执行指令 5

Cycle6	执行跳转目标指令

Operation

(D) If (cond) {PUSH (const20<<1 + PCE1) }
Elsenop

Execute time

7 cycle



Flow Control Operations

RET Return

Flow Control Operations

RET

General Form **RET (.type)** /* (F) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

DT.F5

15	14	13	12	8	7	5	4	3	2	0
s	10	11110			111	0	z	creg		

Description

RET 为函数返回指令, 这条指令的作用是将函数返回程序寄存器 STBUF 中的地址导入 PC, 同时 POP 栈中的值到 STBUF 中。其余操作同跳转指令的执行顺序如下图所示。

Cycle0	执行跳转指令, 产生新的 PC
Cycle1	
Cycle2	
Cycle3	
Cycle4	
Cycle5	
Cycle6	
Cycle7	执行新的指令

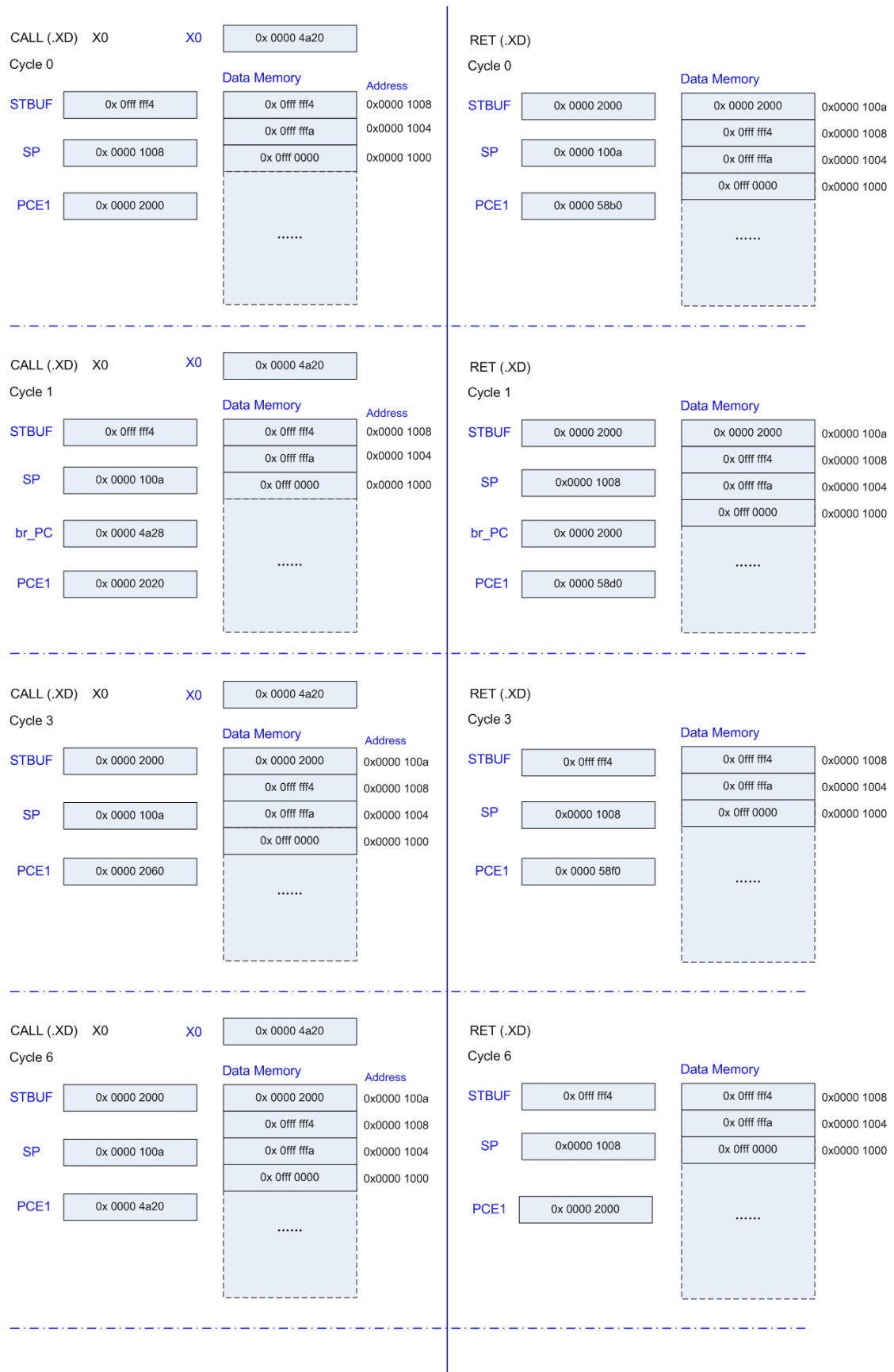
Operation

(F) If (cond) { PC <= RETPC}
 Elsenop

Execute time

7 cycle

Example



Flow Control Operations

RETD

Return Delay

Flow Control Operations

RETD

General Form **RETD (.type)** */* (F) */*
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

DT.F5

15	14	13	12	8	7	5	4	3	2	0
s	10	11110			111	1	z	creg		

Description

RET 为函数返回指令, 这条指令的作用是将函数返回程序寄存器 RETPC 中的地址导入 PC, 同时 POP 栈中的值到 RETPC 中。其余操作同延迟跳转指令的执行顺序如下图所示。

Cycle0	执行跳转指令, 产生新的 PC
Cycle1	继续执行指令 0
Cycle2	继续执行指令 1
Cycle3	继续执行指令 2
Cycle4	继续执行指令 3
Cycle5	继续执行指令 4
Cycle6	继续执行指令 5
Cycle7	执行新的指令

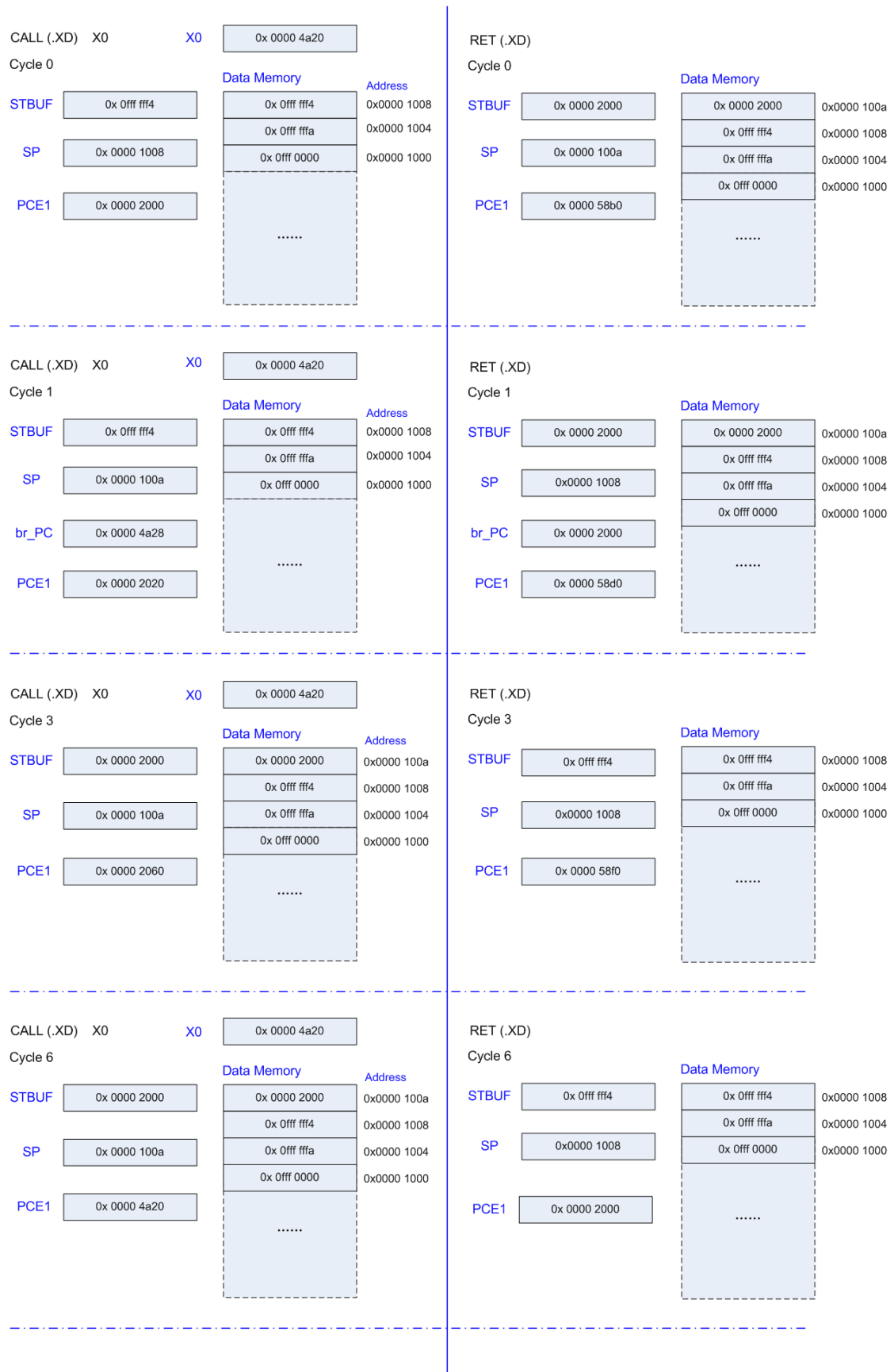
Operation

(F) If (cond) { PC <= RETPC }
 Elsenop

Execute time

7 cycle

Example



Flow Control Operations
IRET Return

Flow Control Operations
IRET

General Form IRET (.type) /* (F) */
 .type = .D

Syntax
(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode
DT.F5

15	14	13	12	8	7	5	4	3	2	0
s	10	11110			010		0	z	creg	

Description
IRET 为中断返回指令, 这条指令的作用是将中断返回程序寄存器 STBUF 中的地址导入 PC, 同时 POP 栈中的值到 STBUF 中, 并给中断控制器发送中断返回信号。其余操作同跳转指令的执行顺序如下图所示。

Cycle0	执行跳转指令, 产生新的 PC
Cycle1	
Cycle2	
Cycle3	
Cycle4	
Cycle5	
Cycle6	
Cycle7	执行新的指令

Operation
(F) If (cond) { PC <= RETPC}
Elsenop

Execute time
7 cycle

Example

Flow Control Operations

IRETD

Return Delay

Flow Control Operations

IRETD

General Form **IRETD (.type)** */* (F) */*
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

DT.F5

15	14	13	12	8	7	5	4	3	2	0
s	10	11110			010	1	z	creg		

Description

RET 为中断返回指令, 这条指令的作用是将中断返回程序寄存器 RETPC 中的地址导入 PC, 同时 POP 栈中的值到 RETPC 中并发送中断返回信号到中断控制器。其余操作同延迟跳转指令的执行顺序如下图所示。

Cycle0	执行跳转指令, 产生新的 PC
Cycle1	继续执行指令 0
Cycle2	继续执行指令 1
Cycle3	继续执行指令 2
Cycle4	继续执行指令 3
Cycle5	继续执行指令 4
Cycle6	继续执行指令 5
Cycle7	执行新的指令

Operation

(F) If (cond) { PC <= RETPC }
 Elsenop

Execute time

7 cycle

Example

Flow Control Operations

NOP

NOP

Flow Control Operations

NOP

General Form **NOP (.type)** /* (F) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

NOP

15	14	13	12	8	7	4	0
s	10	11111	111	00000			

Description

NOP 指令，处理器执行一个周期的空指令。

Operation

(F) NOP

Execute time

1 cycle

Example

NOP (.D)

Flow Control Operations

NOPN

Syntax
(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

NOPN ucst5

15	14	13	12	8	7	4	0
s	10	11111			111	ucst5 - 1	

NOP 指令，处理器执行 N 个周期的空指令。

(F) NOP N

N cycle

```
NOPN (.D) #5 // NOP NOP NOP NOP NOP
```

Flow Control Operations

PAUSE

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

PAUSE

15	14	13	12	8	7	4	0
s	10	11111	111	11111			

PAUSE 指令不停时钟，处理器不执行任何指令，通过所有中断进行唤醒。

(F) NOP ... NOP

1 cycles

IDLE

Idle

IDLE

General Form

IDLE

/* (F) */

```
.type = .D
```

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

IDLE

15	14	13	12			8	7		5	4	3	2		0
s	10	11110					000		0	z	creg			

Description

IDLE 指令暂停时钟，处理器核处于休眠状态，通过所有中断进行唤醒。

Operation

(F) IDLE

Execute time

10 cycles

Flow Control Operations

SLEEP

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

SLEEP

15	14	13	12			8	7		5	4	3	2		0
s	10	11110				000			1	z	creg			

SLEEP 指令清空缓存，暂停处理器时钟，处理器处于休眠状态，只能通过复位信号唤醒。

(F) SLEEP

Flow Control Operations

LPSETUP

Loop Setup

Flow Control Operations

LPSETUP

General Form

LPSETUP0a /* (D) */
 LPSETUP0b /* (D) */
 LPSETUP0c /* (D) */
 LPSETUP0d /* (D) */
 LPSETUP1a /* (D) */
 LPSETUP1b /* (D) */
 LPSETUP1c /* (D) */
 LPSETUP1d /* (D) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

31	30	29	28	27	26	19	18	3	2	0
s	11	10	001	10000	cst16					LPRop

N-LPSETUP select

LPRop	LPSETUP select	
000	LPSETUP0a	将 Loop0_Begin: 地址的低 16 位赋值给 LPT0 的低 16 位
001	LPSETUP0b	将 Loop0_Begin: 地址的高 16 位赋值给 LPT0 的高 16 位
010	LPSETUP0c	将 Loop0_End: 地址加上 LPT0 的结果赋值给 LPB0
011	LPSETUP0d	将 Loop0_Num - 1 赋值给 LPC0
100	LPSETUP1a	将 Loop1_Begin: 地址的低 16 位赋值给 LPT1 的低 16 位
101	LPSETUP1b	将 Loop1_Begin: 地址的高 16 位赋值给 LPT1 的高 16 位
110	LPSETUP1c	将 Loop1_End: 地址加上 LPT1 的结果赋值给 LPB1
111	LPSETUP1d	将 Loop1_Num - 1 赋值给 LPC1

Description

设置 LOOP 参数

Operation

LPT0 [15:0] <= #Loop_Begin [15:0]

LPT0 [31:16] <= #Loop_Begin [31:16]

LPB0 <= #Loop_End + LPT0

LPC0 <= #Loop_Num - 1

Execute time

1 cycle

Example

```
LPSETUP0a      // 将 LOOP_BEGIN 的高 16 位赋给 LPT 的高 16 位
LPSETUP0b      // 将 LOOP_BEGIN 的低 16 位赋给 LPT 的低 16 位
LPSETUP0c      // 将 LOOP_END 加上 LPR.ADR.S 赋给 LPB
LPSETUP0d      // 将循环次数减 1 后赋给 LPC
LOOP0_BEGIN: lable, #100
                // 循环起始地址为 0x1000, 将 LOOP_BEGIN 设为 0x1000
                // 循环次数是 100

    ADD (.A) X0, X1
    ||SUB (.M) X5, X6

    ADD (.A) X0, X2
    ||SUB (.M) X5, X7

    ADD (.A) X0, X3
    ||SUB (.M) X5, X8

LOOP0_END: lable
// 循环结束地址为 0x1005, 结束地址和起始地址的差为 5, 将 LOOP_END 设为 5
```


Flow Control Operations

ING

ING

Flow Control Operations

ING

General Form **ING (.type)** /* (F) */
 .type = .D

Syntax

(F) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

NOP

15	14	13	12	8	7	4	0
s	10	11111	011	00000			

Description

Operation

(F) NOP

Execute time

1 cycle

Example

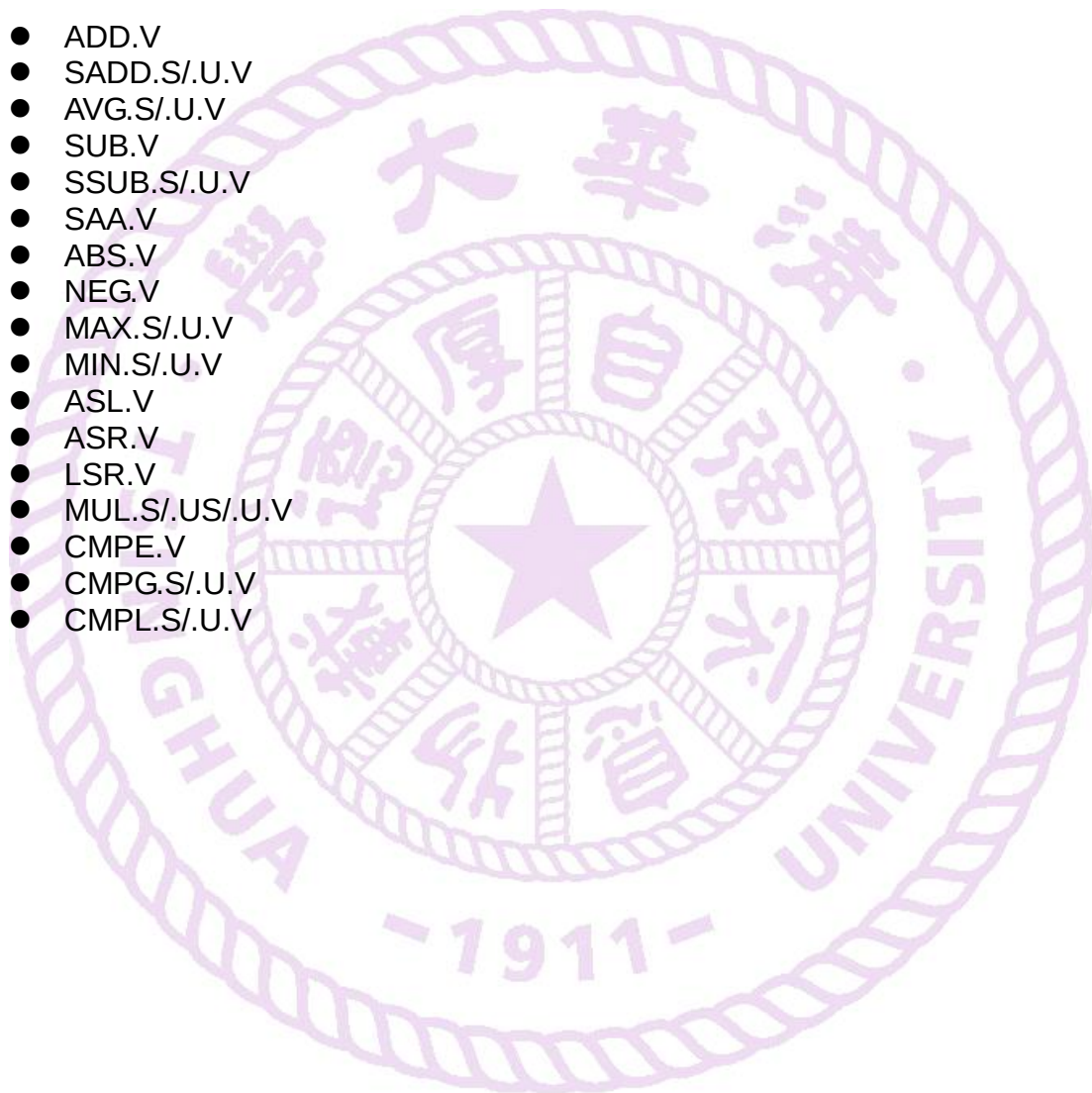
NOP (.D)



15 VECTOR OPERATIONS

Instruction Summary

- ADD.V
- SADD.S/.U.V
- AVG.S/.U.V
- SUB.V
- SSUB.S/.U.V
- SAA.V
- ABS.V
- NEG.V
- MAX.S/.U.V
- MIN.S/.U.V
- ASL.V
- ASR.V
- LSR.V
- MUL.S/.US/.U.V
- CMPE.V
- CMPG.S/.U.V
- CMPL.S/.U.V



Vector Operations

ADD.V

Signed Vector Addition Without Saturation

ADD.V

General Form **ADD.V** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				rs2/const5		rs1		rd		z		creg	

ADD.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	000 01110
rs1	svec	.A	
rs2	svec		
rd	svec	.A	000 01111
rs1	svec	.A	
rs2	scst5		

Description

根据矢量控制寄存器中的设置，将寄存器 rs2 与寄存器 rs1 中的有符号矢量相加，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数为 5bit 有符号立即数。

VSR 设置

31	24	23	18	17	16
Revered			VL_XD		VWM_X D
VL_XM	VWM_X M	VL_XA		VWM_X A	
15	10	9	8	7	2 1 0

31	24	23	18	17	16
Revered			VL_YD		VWM_Y D
VL_YM	VWM_Y M	VL_YA		VWM_Y A	
15	10	9	8	7	2 1 0

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond) $rd\_vec \leq rs1\_svec + rs2\_svec/scst5$
Else nop

Execute time

1 cycle

Vector Operations

SADD.S/.U.V Signed or Unsigned Vector Addition With Saturation

Vector Operations

SADD.S/.U.V

General Form **SADD.S.V** (.type) rd, rs1, rs2 /* (D) */
 SADD.U.V (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SADD.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	000 11010
rs1	svec	.A	
rs2	svec		

SADD.U.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	uvec	.A	000 11011
rs1	uvec	.A	
rs2	uvec		

Description

根据矢量控制寄存器中的设置，将寄存器 rs2 与寄存器 rs1 中的有符号矢量相加，结果经过饱和处理后存放于寄存器 rd 中。

VSR 设置

31	24	23	18	17	16
Revered	VL_XD		VWM_X D		
VL_XM	VWM_X M	VL_XA	VWM_X A		
15	10	9	8	7	2
31	24	23	18	17	16
Revered	VL_YD		VWM_Y D		
VL_YM	VWM_Y M	VL_YA	VWM_Y A		
15	10	9	8	7	2
					1
					0

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$=1$

Operation

If (cond) rd_vec $\leq$ sat(rs1_vec + rs2_vec)
Else nop

VWM		Saturation rules	
2'b00	8bit (default)	Signed	rs2 + rs1 $> 2^7-1$, then rd = 2^7-1 ; rs2 + rs1 $< -2^7$, then rd = -2^7 ;
		Unsigned	rs2 + rs1 $> 2^8-1$, then rd = 2^8-1 ; rs2 + rs1 < 0 , then rd = 0;
2'b01	16bit	Signed	rs2 + rs1 $> 2^{15}-1$, then rd = $2^{15}-1$; rs2 + rs1 $< -2^{15}$, then rd = -2^{15} ;
		Unsigned	rs2 + rs1 $> 2^{16}-1$, then rd = $2^{16}-1$; rs2 + rs1 < 0 , then rd = 0;
2'b10	32bit	Signed	rs2 + rs1 $> 2^{31}-1$, then rd $\leq 2^{31}-1$; rs2 + rs1 $< -2^{31}$, then rd $\leq -2^{31}$;
		Unsigned	rs2 + rs1 $> 2^{32}-1$, then rd $\leq 2^{32}-1$; rs2 + rs1 < 0 , then rd ≤ 0 ;

Execute time

1 cycle

Vector Operations

AVG.S/.U.V

Signed or Unsigned Vector Average

Vector Operations

AVG.S/.U.V

General Form **AVG.S.V** (.type) rd, rs1, rs2 /* (D) */
AVG.U.V (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

AVG.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	000 11111
rs1	svec	.A	
rs2	svec		

AVG.U.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	uvec	.A	000 11110
rs1	uvec	.A	
rs2	uvec		

Description

根据矢量控制寄存器中的设置，寄存器 rs2 中的有符号或无符号矢量分别与 rs1 中对应的有符号或无符号矢量取平均值，结果放在目标寄存器 rd 的矢量中。

VSR 设置

31	24	23	18	17	16
Revered	VL_XD			VWM_X D	
VL_XM	VWM_X M	VL_XA			VWM_X A
15	10	9	8	7	2
					1
					0

31	24	23	18	17	16
Revered	VL_YD			VWM_Y D	
VL_YM	VWM_Y M	VL_YA			VWM_Y A
15	10	9	8	7	2
					1
					0

VWM: Vector Width Mode:

00: 8bit (default)
01: 16bit
10: 32bit
11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位, 即 VMAX 为 32, 这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit, 矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond) rd_vec $\leq$ (rs1_vec + rs2_vec + 1) $\gg$ 1 (signed extend or unsigned extend)
Else nop

Execute time

1 cycle

Vector Operations

SUB.V

Signed Vector Subtract Without Saturation

Vector Operations

SUB.V

General Form **SUB.V** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SUB.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	001 01110
rs1	svec	.A	
rs2	svec		
rd	svec	.A	001 01111
rs1	svec	.A	
rs2	scst5		

Description

根据矢量控制寄存器中的设置，将寄存器 rs2 与寄存器 rs1 中的有符号矢量相减，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数为 5bit 有符号立即数。

VSR 设置

31	24				23	18				17	16
Revered					VL_XD				VWM_X D		
VL_XM			VWM_X M		VL_XA				VWM_X A		
15	10		9	8	7	2		1	0		
31	24				23	18				17	16
Revered					VL_YD				VWM_Y D		
VL_YM			VWM_Y M		VL_YA				VWM_Y A		
15	10		9	8	7	2		1	0		

VWM: Vector Width Mode:

00: 8bit (default)

01: 16bit

10: 32bit

11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位, 即 VMAX 为 32, 这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit, 矢量长度是 4。

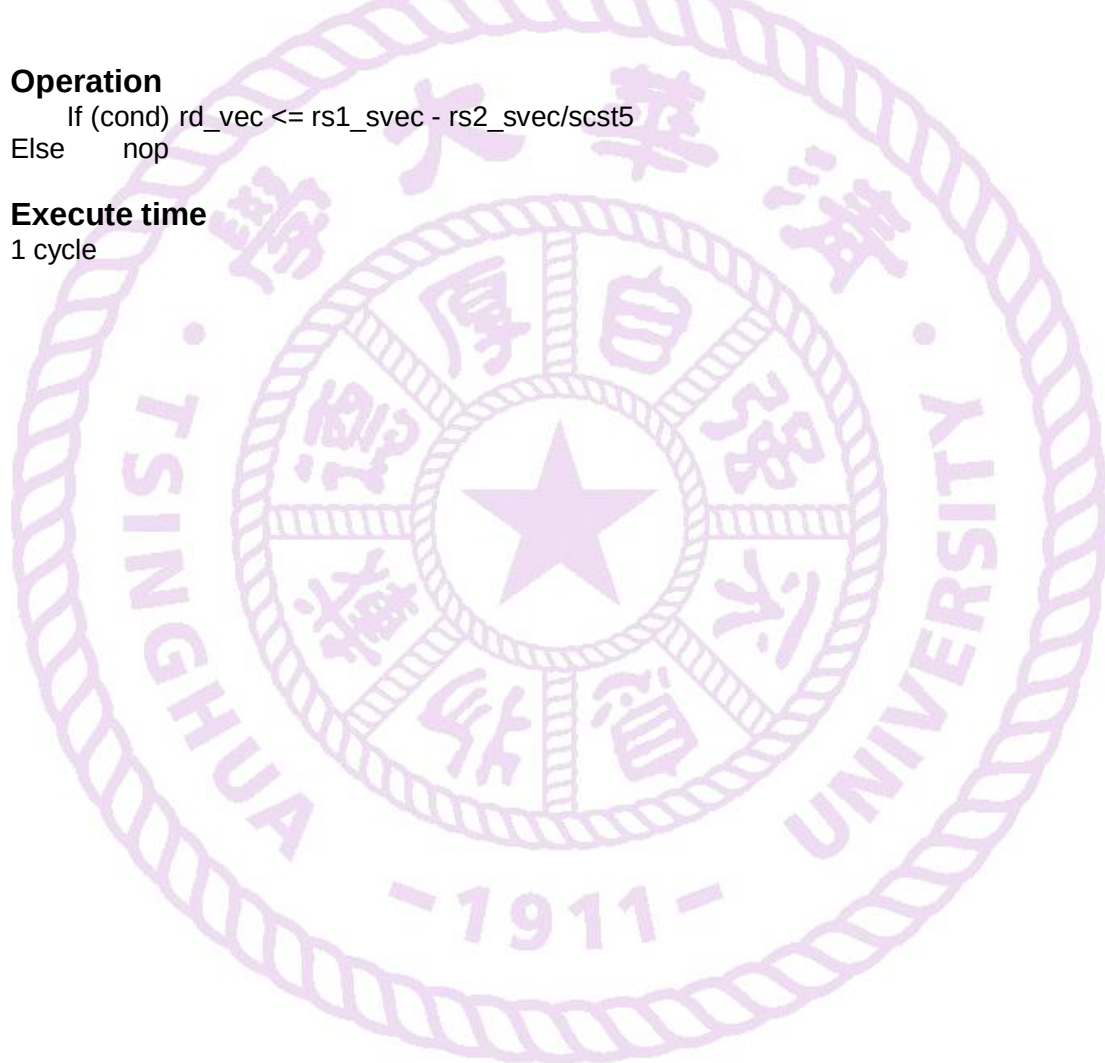
VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	=1

Operation

If (cond) rd_vec $\leq$ rs1_svec - rs2_svec/scst5
Else nop

Execute time

1 cycle



Vector Operations

SSUB.S/.U.V Signed or Unsigned Vector Subtract With Saturation

Vector Operations

SSUB.S/.U.V

General Form **SSUB.S.V** (.type) rd, rs1, rs2 /* (D) */
SSUB.U.V (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

SSUB.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	001 11000
rs1	svec	.A	
rs2	svec		

SSUB.U.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	uvec	.A	001 11001
rs1	uvec	.A	
rs2	uvec		

Description

根据矢量控制寄存器中的设置，将寄存器 rs2 与寄存器 rs1 中的有符号矢量相加，结果经过饱和处理后存放于寄存器 rd 中。

VSR 设置

31	24	23	18	17	16
Revered	VL_XD		VWM_X D		
VL_XM	VWM_X M	VL_XA	VWM_X A		
15	10	9	8	7	2
					1
					0

31	24	23	18	17	16
Revered	VL_YD		VWM_Y D		
VL_YM	VWM_Y M	VL_YA	VWM_Y A		
15	10	9	8	7	2
					1
					0

VWM: Vector Width Mode:
 00: 8bit (default)
 01: 16bit
 10: 32bit
 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$=1$

Operation

If (cond) rd_vec $\leq$ sat(rs1_vec - rs2_vec)
 Else nop

VWM		Saturation rules	
2'b00	8bit (default)	Signed	rs2 + rs1 $> 2^7-1$, then rd = 2^7-1 ; rs2 + rs1 $< -2^7$, then rd = -2^7 ;
		Unsigned	rs2 + rs1 $> 2^8-1$, then rd = 2^8-1 ; rs2 + rs1 < 0 , then rd = 0;
2'b01	16bit	Signed	rs2 + rs1 $> 2^{15}-1$, then rd = $2^{15}-1$; rs2 + rs1 $< -2^{15}$, then rd = -2^{15} ;
		Unsigned	rs2 + rs1 $> 2^{16}-1$, then rd = $2^{16}-1$; rs2 + rs1 < 0 , then rd = 0;
2'b10	32bit	Signed	rs2 + rs1 $> 2^{31}-1$, then rd $\leq 2^{31}-1$; rs2 + rs1 $< -2^{31}$, then rd $\leq -2^{31}$;
		Unsigned	rs2 + rs1 $> 2^{32}-1$, then rd $\leq 2^{32}-1$; rs2 + rs1 < 0 , then rd ≤ 0 ;

Execute time

1 cycle

Vector Operations

SSA.S/.U.V

Vector Subtract and Sum With Absolution

SSA S/.U.V

General Form	SSA.S.V (.type) rd, rs1, rs2	/* (D) */
	SSA.U.V (.type) rd, rs1, rs2	/* (D) */
	.type = .A	

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				rs2/const5		rs1		0000a		z	creg		

a: ACC select

```
-- "0" ACC0  X12 /Y12
```

```
-- "1" ACC1  X22 /Y22
```

SSA.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	sint	.A	001 11010
rs1	svec	.A	
rs2	svec		

SSA.U.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	uint	.A	001	11011
rs1	uvec	.A		
rs2	uvec			

Description

根据矢量控制寄存器中的设置，寄存器 **rs2** 中的有符号或无符号矢量分别与 **rs1** 中对应的有符号或无符号矢量相减后取绝对值，并将结果进行累加，累加后的结果与累加寄存器 **ACC** 中的 **32** 位数相加后放在目标寄存器 **ACC** 中。

VSR 设置

31				24	23	18		17	16	
Revered				VL_XD			VWM_X D			
VL_XM			VWM_X M	VL_XA			VWM_X A			
15				10	9	8	7	2	1	0
31				24	23	18		17	16	

Revered				VL_YD				VWM_Y D		
VL_YM			VWM_Y M	VL_YA				VWM_Y A		
15		10	9	8	7			2	1	0

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位, 即 VMAX 为 32, 这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit, 矢量长度是 4。

VWM		VL
2'b00	8bit (default)	4 (default VL=4)
2'b01	16bit	2
2'b10	32bit	1

Operation

If (cond)
 $ACC = ACC + \sum \{ \text{abs}[rs1\_vec(n) - rs2\_vec(n)] + \dots + \text{abs}[rs1\_vec(0) - rs2\_vec(0)] \}$
 Else nop

Execute time

1 cycle

Vector Operations

ABS.V

Signed Vector Integer Absolute Value with Saturation

ABS.V

General Form **ABS.V** (.type) rd, rs1 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				Op 2				rs1		rd		z	creg

ABS.V (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	011 1111 00011
rs1	svec	.A	

Description

根据矢量控制寄存器中的设置，将寄存器 **rs2** 与寄存器 **rs1** 中的有符号矢量相加，结果存放于寄存器 **rd** 中。当源操作数 **rs2** 为 5bit 立即数时，立即数为 5bit 有符号立即数。

VSR 设置

31				24	23					18	17	16
Revered					VL_XD					VWM_X D		
VL_XM				VWM_X M	VL_XA					VWM_X A		
15				10	9	8	7			2	1	0

31				24		23		18		17		16	
Revered						VL_YD				VWM_Y		D	
VL_YM				VWM_Y		VL_YA				VWM_Y		A	
15				10		9		8		7		2	
										1		0	

VWM: Vector Width Mode:

00: 8bit (default)

01: 16bit

10: 32bit

```
11: reserved
```

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位, 即 VMAX 为 32, 这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit, 矢量长度是 4。

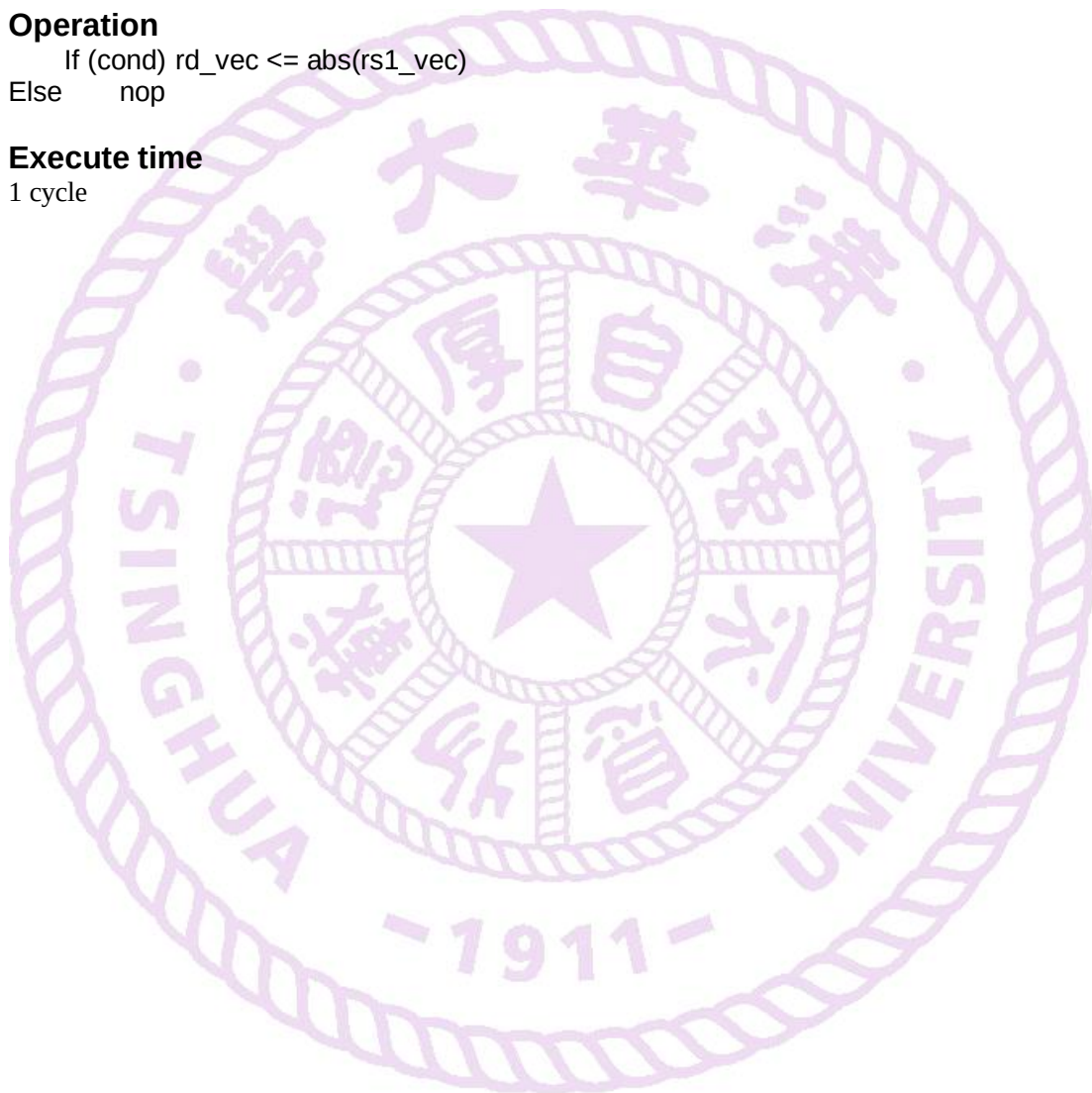
VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	=1

Operation

If (cond) rd_vec $\leq$ abs(rs1_vec)
Else nop

Execute time

1 cycle



Vector Operations

NEG.V

Signed Vector Integer Negate

Vector Operations

NEG.V

General Form **NEG.V** (.type) rd, rs1 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

NEG.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	011 1111 01011
rs1	svec	.A	
rs2	svec		

Description

根据矢量控制寄存器中的设置，将寄存器 rs2 与寄存器 rs1 中的有符号矢量相加，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数为 5bit 有符号立即数。

VSR 设置

31	24	23	18	17	16
Revered	VL_XD		VWM_X D		
VL_XM	VWM_X M	VL_XA	VWM_X A		
15	10	9	8	7	2

31	24	23	18	17	16
Revered	VL_YD		VWM_Y D		
VL_YM	VWM_Y M	VL_YA	VWM_Y A		
15	10	9	8	7	2

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位, 即 VMAX 为 32, 这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit, 矢量长度是 4。

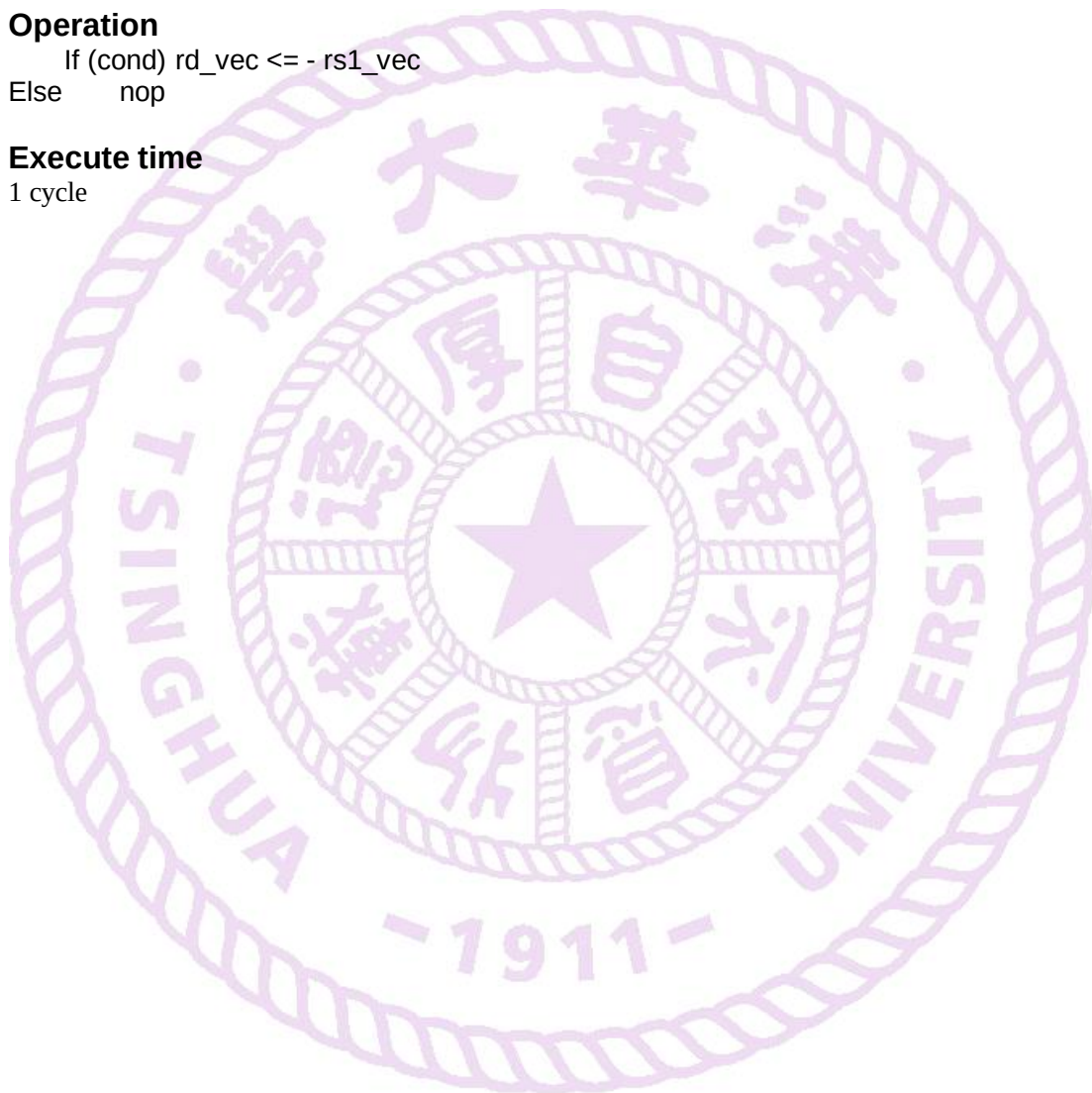
VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	=1

Operation

If (cond) rd_vec $\leq$ - rs1_vec
Else nop

Execute time

1 cycle



Vector Operations MAX.S/.U.V

Vector Maximum, Signed or Unsigned

Vector Operations MAX.S/.U.V

General Form **MAX.S.V (.type)** rd, rs1, rs2 /* (D) */
MAX.U.V (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MAX.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	2hws	.A	011	00100
rs1	2hws	.A		
rs2	2hws			

MAX.U.V

(D) Format

Rd	2hwu	.A	011	00101
rs1	2hwu	.A		
rs2	2hwu			

Description

寄存器 **rs2** 中的矢量分别与 **rs1** 中对应的矢量做比较取最大值，结果放在目标寄存器 **rd** 的矢量中。

VSR 设置

31	24	23	18	17	16
Revered	VL_XD			VWM_XD	
VL_XM	VWM_XM	VL_XA			VWM_XA
15	10	9	8	7	2
					1
					0

31	24	23	18	17	16
Revered	VL_YD			VWM_YD	
VL_YM	VWM_YM	VL_YA			VWM_YA
15	10	9	8	7	2
					1
					0

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM	VL	
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond)

```
{ rd_vec <= max(rs2_vec,rs1_vec) }
  else nop
```

Execute time

1 cycle

Vector Operations MIN.S/.U.V

Vector Minimum, Signed or Unsigned

Vector Operations MIN.S/.U.V

General Form **MIN.S.V** (.type) rd, rs1, rs2 /* (D) */
 MIN.U.V (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

MIN.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	2hws	.A	011	00110
rs1	2hws	.A		
rs2	2hws			

MIN.U.V

(D) Format

Rd	2hwu	.A	011	00111
rs1	2hwu	.A		
rs2	2hwu			

Description

寄存器 **rs2** 中的矢量分别与 **rs1** 中对应的矢量做比较取最小值，结果放在目标寄存器 **rd** 的矢量中。

VSR 设置

31	24	23	18	17	16
Revered	VL_XD			VWM_XD	
VL_XM	VWM_XM	VL_XA			VWM_XA
15	10	9	8	7	2
					1
					0

31	24	23	18	17	16
Revered	VL_YD			VWM_YD	
VL_YM	VWM_YM	VL_YA			VWM_YA
15	10	9	8	7	2
					1
					0

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM	VL
2'b00	8bit (default) <= 4 (default VL=4)
2'b01	16bit <= 2
2'b10	32bit ==1

Operation

If (cond)

```
{ rd_vec <= min(rs2_vec,rs1_vec) }
  else nop
```

Execute time

1 cycle

Vector Operations

ASL.V

Vector Arithmetic Shift Left

Vector Operations

ASL.V

General Form **ASL.V** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

ASL.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	010 11010
rs1	svec	.A	
rs2	svec		
rd	svec	.A	010 11011
rs1	svec	.A	
rs2	ucst5		

Description

根据矢量控制寄存器中的设置，将寄存器 rs1 中的有符号矢量根据 rs2 中的有符号数进行向左移位，末尾添 0，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数为 5bit 无符号立即数。移位位数 rs2 只取最低有效的 3 位，且为无符号数，其余位将被忽略，移位位数取最大为 7。如果移位位数为无符号立即数 cst5，移位位数只取 cst5 的低 3 位，移位位数在 0 到 7 之间。

VSR 设置

31	24				23	18				17	16
Revered					VL_XD				VWM_X D		
VL_XM			VWM_X M		VL_XA				VWM_X A		
15	10		9	8	7	2				1	0
31	24				23	18				17	16
Revered					VL_YD				VWM_Y D		
VL_YM			VWM_Y M		VL_YA				VWM_Y A		
15	10		9	8	7	2				1	0

VWM: Vector Width Mode:

00: 8bit (default)
01: 16bit
10: 32bit
11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位, 即 VMAX 为 32, 这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit, 矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

```
If (cond)
{
  If (rs2 > 0) rd_vec <= rs1_vec << rs2;
  else      rd_vec <= rs1_vec >> abs(rs2)
}
Else      nop
```

Execute time

1 cycle

ASR.V

Vector Arithmetic Shift Right

Vector Operations

ASR.V

General Form **ASR.V** (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				rs2/const5		rs1		rd		z	creg		

ASR.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	svec	.A	010	11100
rs1	svec	.A		
rs2	svec			
rd	svec	.A	010	11101
rs1	svec	.A		
rs2	ucst5			

Description

根据矢量控制寄存器中的设置，将寄存器 **rs1** 中的有符号矢量根据 **rs2** 中的有符号数进行向右移位，并进行符号扩展，结果存放于寄存器 **rd** 中。当源操作数 **rs2** 为 **5bit** 立即数时，立即数为 **5bit** 无符号立即数。移位位数 **rs2** 只取最低有效的 **3** 位，且为无符号数，其余位将被忽略，移位位数取最大为 **7**。如果移位位数为无符号立即数 **cst5**，移位位数只取 **cst5** 的低 **3** 位，移位位数在 **0** 到 **7** 之间。

VSR 设置

31								24				23				18				17		16	
Revered								VL_XD								VWM_X D							
VL_XM						VWM_X M		VL_XA						VWM_X A									
15								10		9		8		7		2		1		0			

31								24				23				18				17		16	
Revered												VL_YD								VWM_Y D			
VL_YM								VWM_Y M				VL_YA						VWM_Y A					
15						10		9		8		7				2		1		0			

VWM: Vector Width Mode:

00: 8bit (default)
01: 16bit
10: 32bit
11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond)

```
{  
If (rs2 > 0) rd_vec <= rs1_vec >> rs2;  
else      rd_vec <= rs1_vec << abs(rs2)  
}
```

Else nop

Execute time

1 cycle

Vector Operations

LSR.V

Vector Logical Shift Right

Vector Operations

LSR.V

General Form **LSR.V** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

LSR.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.A	010 11110
rs1	svec	.A	
rs2	svec		
rd	svec	.A	010 11111
rs1	svec	.A	
rs2	ucst5		

Description

根据矢量控制寄存器中的设置，将寄存器 rs1 中的有符号矢量根据 rs2 中的有符号数进行向右移位，首位添 0，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数为 5bit 无符号立即数。具体移位方式参造 LSR

VSR 设置

31	24	23	18	17	16
Revered	VL_XD		VWM_X D		
VL_XM	VWM_X M	VL_XA	VWM_X A		
15	10	9	8	7	2
31	24	23	18	17	16
Revered	VL_YD		VWM_Y D		
VL_YM	VWM_Y M	VL_YA	VWM_Y A		
15	10	9	8	7	2

VWM: Vector Width Mode:

00: 8bit (default)

01: 16bit

10: 32bit
11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$=1$

Operation

```
If (cond)
{
  If (rs2 > 0) rd_vec <= rs1_vec >> rs2;
  else      rd_vec <= rs1_vec << abs(rs2)
}
Else      nop
```

Execute time

1 cycle

Vector Operations

MUL.S/.US/.U.V

Signed or Unsigned Vector Multiply Without Saturation

Vector Operations

MUL.S/.US/.U.V

General Form **MUL.S.V** (.type) rd, rs1, rs2 /* (D) */
 MUL.US.V (.type) rd, rs1, rs2 /* (D) */
 MUL.U.V (.type) rd, rs1, rs2 /* (D) */
 .type = .M

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op	1	Op	2	rs1	rd	z	creg						

MUL.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.M	001 11100
rs1	svec	.M	
rs2	svec		

MUL.US.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.M	001 11110
rs1	uvec	.M	
rs2	svec		

MUL.U.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	uvec	.M	001 11101
rs1	uvec	.M	
rs2	uvec		

Description

根据矢量控制寄存器中的设置，将寄存器 rs2 与寄存器 rs1 中的矢量相乘，结果存放于寄存器 rd 中。当源操作数 rs2 为 5bit 立即数时，立即数为 5bit 有符号立即数。

VSR 设置

31	24	23	18	17	16
Revered	VL_XD			VWM_X D	
VL_XM	VWM_X M	VL_XA			VWM_X A
15	10	9	8	7	2
					1
					0

31	24	23	18	17	16
Revered	VL_YD			VWM_Y D	
VL_YM	VWM_Y M	VL_YA			VWM_Y A
15	10	9	8	7	2
					1
					0

VWM: Vector Width Mode:

00: 8bit (default)

01: 16bit

10: 32bit

11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位, 即 VMAX 为 32, 这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit, 矢量长度是 4。

VWM	VL	
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond) $rd\_vec \leq rs1\_svec \times rs2\_svec / scst5$

Else nop

Execute time

2 cycle

Example

VL: 4 VWM:2'b00 (VW: 8bit)

Rs1

ua_3	ua_2	ua_1	ua_0

Rs2

ub_3	ub_2	ub_1	ub_0

MUL.U.V

Rd+1:Rd

63	48	47	32
ua_3 x ub_3		ua_2 x ub_2	
ua_1 x ub_1		ua_0 x ub_0	
31	16	15	0

Vector Operations

CMPE.V

Vector Compare for Equality

Vector Operations

CMPE.V

General Form **CMPE.V** (.type) rd, rs1, rs2 /* (D) */
 .type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

CMPE.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.D	000 11001
rs1	svec	.D	
rs2	svec		

Description

这条指令的作用是将寄存器 rs1 中的数据与寄存器 rs2 中的矢量数据进行比较，如果两个数据相等，则将 rd 相应的矢量置 1，如果两个数据不相等，则将 rd 相应的矢量置零。

VL:4 VWM:2'b00 (VW: 8bit)

Rs1

13	46	AB	26

Rs2

24	56	AB	26

CMPE.V

Rd

8'b0	8'b0	8'b1	8'b1

VSR 设置

31				24		23		18		17		16	
Revered								VL_XD				VWM_XD	
VL_XM						VWM_XM		VL_XA				VWM_XA	
15				10		9		8		7		2	
												1	
												0	

31	24	23	18	17	16
Revered	VL_YD		VWM_YD		
VL_YM	VWM_YM	VL_YA	VWM_YA		
15	10	9	8	7	2
					1
					0

VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond)

```
{
  if (rs1_vec == rs2_vec) rd_vec <= 1
  else rd_vec <= 0
}
```

Elsenop

Execute time

1 cycle

Vector Operations

CMPG.S/.U.V Vector Compare for Greater Than

Vector Operations

CMPG.S/.U.V

General Form **CMPG.S.V** (.type) rd, rs1, rs2 /* (D) */
CMPG.U.V (.type) rd, rs1, rs2 /* (D) */
.type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

CMPG.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.D	000 11010
rs1	svec	.D	
rs2	svec		

CMPG.U.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	uvec	.D	000 11011
rs1	uvec	.D	
rs2	uvec		

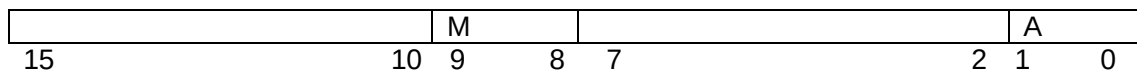
Description

这条指令的作用是将寄存器 rs1 中的数据与寄存器 rs2 中的矢量数据进行比较，rs1 > rs2，则将 rd 相应的矢量置 1，如果两个数据不相等，则将 rd 相应的矢量置零。

VSR 设置

31	24	23	18	17	16
Revered		VL_XD		VWM_X	D
VL_XM	VWM_X	VL_XA		VWM_X	A
15	10	9	8	7	2
					1
					0

31	24	23	18	17	16
Revered		VL_YD		VWM_Y	D
VL_YM	VWM_Y	VL_YA		VWM_Y	



VWM: Vector Width Mode:

- 00: 8bit (default)
- 01: 16bit
- 10: 32bit
- 11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond)

```
{
  if (rs1_vec > rs2_vec) rd_vec <= 1
  else rd_vec <= 0
}
```

Elsenop

Execute time

1 cycle

Vector Operations

CMPL.S/.U.V Vector Compare for Less Than

Vector Operations

CMPL.S/.U.V

General Form **CMPL.S.V** (.type) rd, rs1, rs2 /* (D) */
CMPL.U.V (.type) rd, rs1, rs2 /* (D) */
.type = .D

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

DT.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

CMPL.S.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	svec	.D	000 11100
rs1	svec	.D	
rs2	svec		

CMPL.U.V

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	uvec	.D	000 11101
rs1	uvec	.D	
rs2	uvec		

Description

这条指令的作用是将寄存器 rs1 中的数据与寄存器 rs2 中的矢量数据进行比较，rs1 < rs2，则将 rd 相应的矢量置 1，如果两个数据不相等，则将 rd 相应的矢量置零。

VSR 设置

31							24	23							18	17	16
Revered								VL_XD						VWM_X D			
VL_XM				VWM_X M		VL_XA						VWM_X A					
15	10			9	8	7							2	1	0		

31					24	23					18	17	16		
Revered								VL_YD				VWM_Y D			
VL_YM						VWM_Y M		VL_YA				VWM_Y A			
15					10	9	8	7					2	1	0

VWM: Vector Width Mode:

00: 8bit (default)

01: 16bit

10: 32bit

11: reserved

VL: Vector Length, default is 4.

VMAX: (系统设置) Datapath Width, $VL * VWM \leq VMAX$

Lily 处理器中.A 单元的最大数据通道为 32 位，即 VMAX 为 32，这就约束了该功能单元的 VL 和 VW 如下表所示。系统默认矢量宽度为 8bit，矢量长度是 4。

VWM		VL
2'b00	8bit (default)	≤ 4 (default VL=4)
2'b01	16bit	≤ 2
2'b10	32bit	$= 1$

Operation

If (cond)

```
{  if (rs1_vec < rs2_vec  rd_vec <= 1
    else  rd_vec <= 0
}
```

Elsenop

Execute time

1 cycle

16 64-BIT REGFILE G OPERATIONS

Instruction Summary

- ADDG
- SUBG
- PACKGB0
- PACKGB1
- PACKGB2
- PACKGB3
- PACKGB4
- PACKGB5
- PACKGH0
- PACKGH1
- PACKGH2
- PACKGH3
- PACKGW0
- PACKGW1
- PACKGW2
- PACKGW3
- UNPACKGB0
- UNPACKGB1
- UNPACKGB2
- UNPACKGB3
- UNPACKGB4
- UNPACKGB5
- UNPACKGH0
- UNPACKGH1
- UNPACKGH2
- UNPACKGH3
- UNPACKGW0
- UNPACKGW1
- UNPACKGW2
- UNPACKGW3
- SWAPGH0
- SWAPGH1
- SWAPGH2
- ASLG
- ASRG
- LSRG
- ROLG

64-bit G Operations

ADDG

64-bit G ADDG

64-bit G Operations

ADDG

General Form **ADDG** (.type) rd, rs1, rs2 /* (D) */
 .type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length
 In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	creg								

ADDG

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	Int	.A	000	00110
rs1	Int	.A		
rs2	Int			
rd	Int	.A	000	00111
rs1	Int	.A		
rs2	cst5			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的无符号数(或 5 位常数)相加，结果放入 G 寄存器 rd 中。

Operation

If (cond) rd <= rs1 + rs2/cst5
Else nop

Execute time

1 cycle

64-bit G Operations

SUBG

64-bit G SUBG

64-bit G Operations

SUBG

General Form SUBG (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax
(D) rd, rs: G0~G7

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1	rs2/const5	rs1	rd	z	Creg								

SUBG

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	Int	.A	001	00110
rs1	Int	.A		
rs2	Int			
rd	Int	.A	001	00111
rs1	Int	.A		
rs2	cst5			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的无符号数(或 5 位常数)相减, 结果放入 G 寄存器 rd 中。

Operation

If (cond) rd <= rs1 - rs2/cst5
Else nop

Execute time

1 cycle

PACKGB0

64-bit G PACKGB0

64-bit G Operations

PACKGB0

General Form **PACKGB0** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2			rs1			rd		z	Creg

PACKGB0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	Int	.A	111	00000
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字节进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63								0
7	6	5	4	3	2	1	0	

Rs2:

f	e	d	c	b	a	9	8

Rd:

d	9	5	1	c	8	4	0

Operation

Execute time

1 cycle

PACKGB1

64-bit G PACKGB1

64-bit G Operations

PACKGB1

General Form **PACKGB1** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z	Creg		

PACKGB1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	00001
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字节进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63							0
7	6	5	4	3	2	1	0

Rs2:

f	e	d	c	b	a	9	8

Rd:

f	b	7	3	e	a	6	2

Operation

Execute time

1 cycle

PACKGB2

64-bit G PACKGB2

64-bit G Operations

PACKGB2

General Form **PACKGB2** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd			z	Creg	

PACKGB2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	00010
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字节进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63								0
7	6	5	4	3	2	1	0	

Rs2:

f	e	d	c	b	a	9	8

Rd:

e	c	a	8	6	4	2	0

Operation

Execute time

1 cycle

PACKGB3

64-bit G PACKGB3

64-bit G Operations

PACKGB3

General Form **PACKGB3** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1		rs2		rs1		rd		z		Creg			

PACKGB3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	00011
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字节进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63								0
7	6	5	4	3	2	1	0	

Rs2:

f	e	d	c	b	a	9	8

Rd:

f	d	b	9	7	5	3	1

Operation

Execute time

1 cycle

PACKGB5

64-bit G PACKGB5

64-bit G Operations

PACKGB5

General Form **PACKGB5** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z	Creg		

PACKGB5

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	00101
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字节进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63							0
7	6	5	4	3	2	1	0

Rs2:

f	e	d	c	b	a	9	8

Rd:

f	7	e	6	d	5	c	4

Operation

Execute time

1 cycle

PACKGH0

64-bit G PACKGH0

64-bit G Operations

PACKGH0

General Form **PACKGH0** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z	Creg		

PACKGH0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	00110
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按半字进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63				0
3	2	1	0	

Rs2:

7	6	5	4

Rd:

6	4	2	0

Operation

Execute time

1 cycle

PACKGH1

64-bit G PACKGH1

64-bit G Operations

PACKGH1

General Form **PACKGH1** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2			rs1			rd		z	Creg

PACKGH1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	Int	.A	111 00111
rs1	Int	.A	
rs2	Int		

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按半字进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63				0
3	2	1	0	

Rs2:

7	6	5	4

Rd:

7	5	3	1

Operation

Execute time

1 cycle

PACKGH2

64-bit G PACKGH2

64-bit G Operations

PACKGH2

General Form **PACKGH2 (.type) rd, rs1, rs2** */* (D) */*

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z	Creg		

PACKGH2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	01000
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按半字进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63				0
3	2	1	0	

Rs2:

7	6	5	4

Rd:

5	1	4	0

Operation

Execute time

1 cycle

PACKGH3

64-bit G PACKGH3

64-bit G Operations

PACKGH3

General Form **PACKGH3** (.type) rd, rs1, rs2 /* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z	Creg		

PACKGH3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	01001
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按半字进行压缩操作，结果放入 G 寄存器 rd 中，压缩方式如下图所示。

Rs1:

63				0
3	2	1	0	

Rs2:

7	6	5	4

Rd:

7	3	6	2

Operation

Execute time

1 cycle

PACKGW0

64-bit G PACKGW0

64-bit G Operations

PACKGW0

General Form **PACKGW0** (.type) rd, rs1, rs2

/* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z	Creg		

PACKGW0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	Int	.A	111 01010
rs1	Int	.A	
rs2	Int		

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字进行压缩操作, 结果放入 G 寄存器 rd 中, 压缩方式如下图所示。

Rs1:

63	0
1	0

Rs2:

3	2

Rd:

2	0

Operation

Execute time

1 cycle

PACKGW1

64-bit G PACKGW1

64-bit G Operations

PACKGW1

General Form **PACKGW1** (.type) rd, rs1, rs2

/* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z		Creg	

PACKGW1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
Rd	Int	.A	111 01011
rs1	Int	.A	
rs2	Int		

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字进行压缩操作, 结果放入 G 寄存器 rd 中, 压缩方式如下图所示。

Rs1:

63	0
1	0

Rs2:

3	2

Rd:

3	1

Operation

Execute time

1 cycle

PACKGW2

64-bit G PACKGW2

64-bit G Operations

PACKGW2

General Form **PACKGW2** (.type) rd, rs1, rs2

/* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z		Creg	

PACKGW2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	01100
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字进行压缩操作, 结果放入 G 寄存器 rd 中, 压缩方式如下图所示。

Rs1:

63	0
1	0

Rs2:

3	2

Rd:

3	0

Operation

Execute time

1 cycle

PACKGW3

64-bit G PACKGW3

64-bit G Operations

PACKGW3

General Form **PACKGW3** (.type) rd, rs1, rs2

/* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	10	Op	Op_1				rs2		rs1		rd		z		Creg	

PACKGW3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	Int	.A	111	01100
rs1	Int	.A		
rs2	Int			

Description

这条指令的作用是将 G 寄存器 rs1 中的数据与 G 寄存器 rs2 中的数据按字进行压缩操作, 结果放入 G 寄存器 rd 中, 压缩方式如下图所示。

Rs1:

63	0
1	0

Rs2:

3	2

Rd:

2	1

Operation

Execute time

1 cycle

64-bit G Operations
UNPACKGB0

64-bit G UNPACKGB0

64-bit G Operations
UNPACKGB0

General Form UNPACKGB0 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
(D) rd, rs: G0~G7

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPACKGB0	(D) Format
Opcode map field used ...	For operand type ... Unit Opfield
rd	sint .A 011 11110 00000
rs	sint .A

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字节进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字节填 0。

Rs:

63	7	6	5	4	3	2	1	0

Rd:

		5	1			4	0

Rd+1:

		7	3			6	2

Operation

Execute time
1 cycle

64-bit G Operations UNPACKGB1

64-bit G UNPACKGB1

64-bit G Operations UNPACKGB1

General Form **UNPACKGB1** (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2																	
31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1			Op_2			rs1			rd			z	creg

UNPACKGB1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 00001
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字节进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字节填 0。

Rs:							
63							0
7	6	5	4	3	2	1	0

Rd:									
5	1				4	0			

Rd+1:									
7	3				6	2			

Operation

Execute time

1 cycle

64-bit G Operations
UNPACKGB2

64-bit G UNPACKGB2

64-bit G Operations
UNPACKGB2

General Form UNPACKGB2 (.type) rd+1:rd, rs /* (D) */
 .type = .A

Syntax
(D) rd, rs: G0~G7

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPACKGB2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 00010
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字节进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字节填 0。

Rs:

63																	0
7	6	5	4	3	2	1	0										

Rd:

	3		2		1		0

Rd+1:

	7		6		5		4

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGB3

64-bit G UNPACKGB3

64-bit G Operations UNPACKGB3

General Form UNPACKGB3 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPACKGB3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 00011
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字节进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字节填 0。

Rs:									0
63	7	6	5	4	3	2	1	0	

Rd:	3	2	1	0				

Rd+1:	7	6	5	4				

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGB4

64-bit G UNPACKGB4

64-bit G Operations UNPACKGB4

General Form UNPACKGB4 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2									
31	30	29	28	27	26	24	23	19	18
14	13	9	8	4	3	2	0		
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg

UNPACKGB4

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 00100
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字节进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字节填 0。

Rs:									
63									0
7	6	5	4	3	2	1	0		

Rd:									
				6	4	2	0		

Rd+1:									
				7	5	3	1		

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGB5

64-bit G UNPACKGB5

64-bit G Operations UNPACKGB5

General Form UNPACKGB5 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2									
31	30	29	28	27	26	24	23	19	18
14	13	9	8	4	3	2	0		
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg

UNPACKGB5

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 00101
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字节进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字节填 0。

Rs:									
63									0
7	6	5	4	3	2	1	0		

Rd:									
6	4	2	0						

Rd+1:									
7	5	3	1						

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGH0

64-bit G UNPACKGH0

64-bit G Operations UNPACKGH0

General Form **UNPACKGH0** (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2									
31	30	29	28	27	26	24	23	19	18
14	13	9	8	4	3	2	0		
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg

UNPACKGH0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 00110
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按半字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的半字填 0。

Rs:			
63			0
3	2	1	0

Rd:			
	1		0

Rd+1:			
	3		2

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGH1

64-bit G UNPACKGH1

64-bit G Operations UNPACKGH1

General Form UNPACKGH1 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2									
31	30	29	28	27	26	24	23	19	18
14	13	9	8	4	3	2	0		
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg

UNPACKGH1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 00111
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按半字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的半字填 0。

Rs:			
63			0
3	2	1	0

Rd:			
1		0	

Rd+1:			
3		2	

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGH2

64-bit G UNPACKGH2

64-bit G Operations UNPACKGH2

General Form

UNPACKGH2 (.type) rd+1:rd, rs
.type = .A

/* (D) */

Syntax

(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPACKGH2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1110 01000
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按半字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的半字填 0。

Rs:

63	3	2	1	0

Rd:

2	0

Rd+1:

3	1

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGH3

64-bit G UNPACKGH3

64-bit G Operations UNPACKGH3

General Form UNPACKGH3 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPACKGH3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 01001
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按半字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的半字填 0。

Rs:

63	3	2	1	0

Rd:

2	0		

Rd+1:

3	1		

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGW0

64-bit G UNPACKGW0

64-bit G Operations UNPACKGW0

General Form UNPACKGW0 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPACKGW0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1110 01010
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字填 0。

Rd:	63	1	0

Rd:	0

Rd+1:	1

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGW1

64-bit G UNPACKGW1

64-bit G Operations UNPACKGW1

General Form UNPACKGW1 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

UNPACKGW1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 11110 01011
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字填 0。

Rd:	63	0
	1	0

Rd:	0	

Rd+1:	1	

Operation

Execute time

1 cycle

64-bit G Operations

UNPACKGW2

64-bit G UNPACKGW2

64-bit G Operations

UNPACKGW2

General Form **UNPACKGW2 (.type) rd+1:rd, rs** */* (D) */*
.type = .A

Syntax
(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		Op_2		rs1		rd		z		creg			

UNPACKGW2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1110 01100
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字填 0。

R _s : 63	1	0

Rd:	0

Rd+1:	1	

Operation

Execute time

1 cycle

64-bit G Operations UNPACKGW3

64-bit G UNPACKGW3

64-bit G Operations UNPACKGW3

General Form UNPACKGW3 (.type) rd+1:rd, rs /* (D) */
.type = .A

Syntax
 (D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2									
31	30	29	28	27	26	24	23	19	18
14	13	9	8	4	3	2	0		
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg

UNPACKGW3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1110 01101
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按字进行扩展操作，结果放入 G 寄存器 rd+1 和 rd 中，扩展方式如下图所示，rd 和 rd+1 中未写入的字填 0。

Rs:									
63									0
								1	

Rd:									
								0	

Rd+1:									
								1	

Operation

Execute time

1 cycle

SWAPGH0

64-bit G SWAPGH0

64-bit G Operations

SWAPGH0

```
General Form      SWAPGH0 (.type) rd+1:rd, rs          /* (D) */
                   .type = .A
```

Syntax
(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		Op_2		rs1		rd		z		creg			

SWAPGH0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1111 10001
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按半字进行交换操作，结果放入 G 寄存器 rd 中。

Rs:

63				0
3	2	1	3	

Rd:

3	1	2	0

Operation

Execute time

1 cycle

SWAPGH1

64-bit G SWAPGH1

SWAPGH1

```
General Form      SWAPGH1 (.type) rd+1:rd, rs          /* (D) */
                   .type = .A
```

Syntax
(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		Op_2		rs1		rd		z		creg			

SWAPGH1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1111 10110
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按半字进行交换操作，结果放入 G 寄存器 rd 中。

Rs:

63				0
3	2	1	3	

Rd:

2	3	0	1

Operation

Execute time

1 cycle

64-bit G Operations

SWAPGH2

64-bit G SWAPGH2

64-bit G Operations

SWAPGH2

```
General Form      SWAPGH2 (.type) rd+1:rd, rs          /* (D) */
                   .type = .A
```

Syntax
(D) rd, rs: G0~G7

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		Op_2		rs1		rd		z		creg			

SWAPGH2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	011 1111 1011
rs	sint	.A	

Description

这条指令的作用是将 G 寄存器 rs 中的数据按半字进行交换操作，结果放入 G 寄存器 rd 中。

Rs:

63				0
3	2	1	3	

Rd:

0	2	1	3

Operation

Execute time

1 cycle

64-bit G Operations

ASLG

G register Shift/Rotate Shift Left

64-bit G Operations

ASLG

General Form ASLG (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

ASLG

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	111 01110
rs1	sint	.A	
rs2	sint		

Description

这条指令的作用是将 G 寄存器 rs1 中的有符号数向左算术移位，移位结果存放在 rd 中，移位的位数由寄存器 rs2/cst5 决定。

由于 rs1 的数据长度为 64 位，移位的位数最大为 63，所以移位位数 rs2 只取最低有效的 6 位，且为无符号数，其余位将被忽略。

如果移位位数为无符号立即数 cst5，cst5 在 0 到 31 之间。

Operation

If (cond)

rd = rs1 << rs2/cst5

Else nop

Execute time

1 cycle

64-bit G Operations

ASRG

G register Shift/Rotate Shift Right

64-bit G Operations

ASRG

General Form ASRG (.type) rd, rs1, rs2 /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	rs2/const5	rs1	rd	z	creg								

ASRG

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	sint	.A	111 01111
rs1	sint	.A	
rs2	sint		

Description

这条指令的作用是将 G 寄存器 rs1 中的有符号数向右算术移位，移位结果存放在 rd 中，移位的位数由寄存器 rs2/cst5 决定。

由于 rs1 的数据长度为 64 位，移位的位数最大为 63，所以移位位数 rs2 只取最低有效的 6 位，且为无符号数，其余位将被忽略。

如果移位位数为无符号立即数 cst5，cst5 在 0 到 31 之间。

Operation

If (cond)

rd = rs1 >> rs2/cst5

Else nop

Execute time

1 cycle

LSRG

G register Shift/Rotate Shift Right

64-bit G Operations

LSRG

General Form **LSRG (.type) rd, rs1, rs2** */* (D) */*
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1		rs2/const5		rs1		rd		z		creg			

LSRG

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	111	10000
rs1	sint	.A		
rs2	sint			

Description

这条指令的作用是将 G 寄存器 rs1 中的有符号数向右逻辑移位，移位结果存放在 rd 中，移位的位数由寄存器 rs2/cst5 决定。

由于 rs1 的数据长度为 64 位，移位的位数最大为 63，所以移位位数 rs2 只取最低有效的 6 位，且为无符号数，其余位将被忽略。

如果移位位数为无符号立即数 `cst5`，`cst5` 在 0 到 31 之间。

Operation

If (cond)

```
rd = rs1 >> rs2/cst5
```

```
Else      nop
```

Execute time

1 cycle

ROLG

G register Shift/Rotate Shift Left

64-bit G Operations

ROLG

```
General Form      ROLG (.type) rd, rs1, rs2      /* (D) */
                   .type = .A
```

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				rs2/const5				rs1		rd		z	creg

ROLG

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
Rd	sint	.A	111	10010
rs1	sint	.A		
rs2	sint			

Description

这条指令的作用是将 G 寄存器 rs1 中的有符号数向左循环移位，移位结果存放在 rd 中，移位的位数由寄存器 rs2/cst5 决定。

由于 rs1 的数据长度为 64 位，移位的位数最大为 63，所以移位位数 rs2 只取最低有效的 6 位，且为无符号数，其余位将被忽略。

如果移位位数为无符号立即数 `cst5`，`cst5` 在 0 到 31 之间。

Operation

If (cond)

$$rd = rs1 \ll rs2/cst5$$

```
Else      nop
```

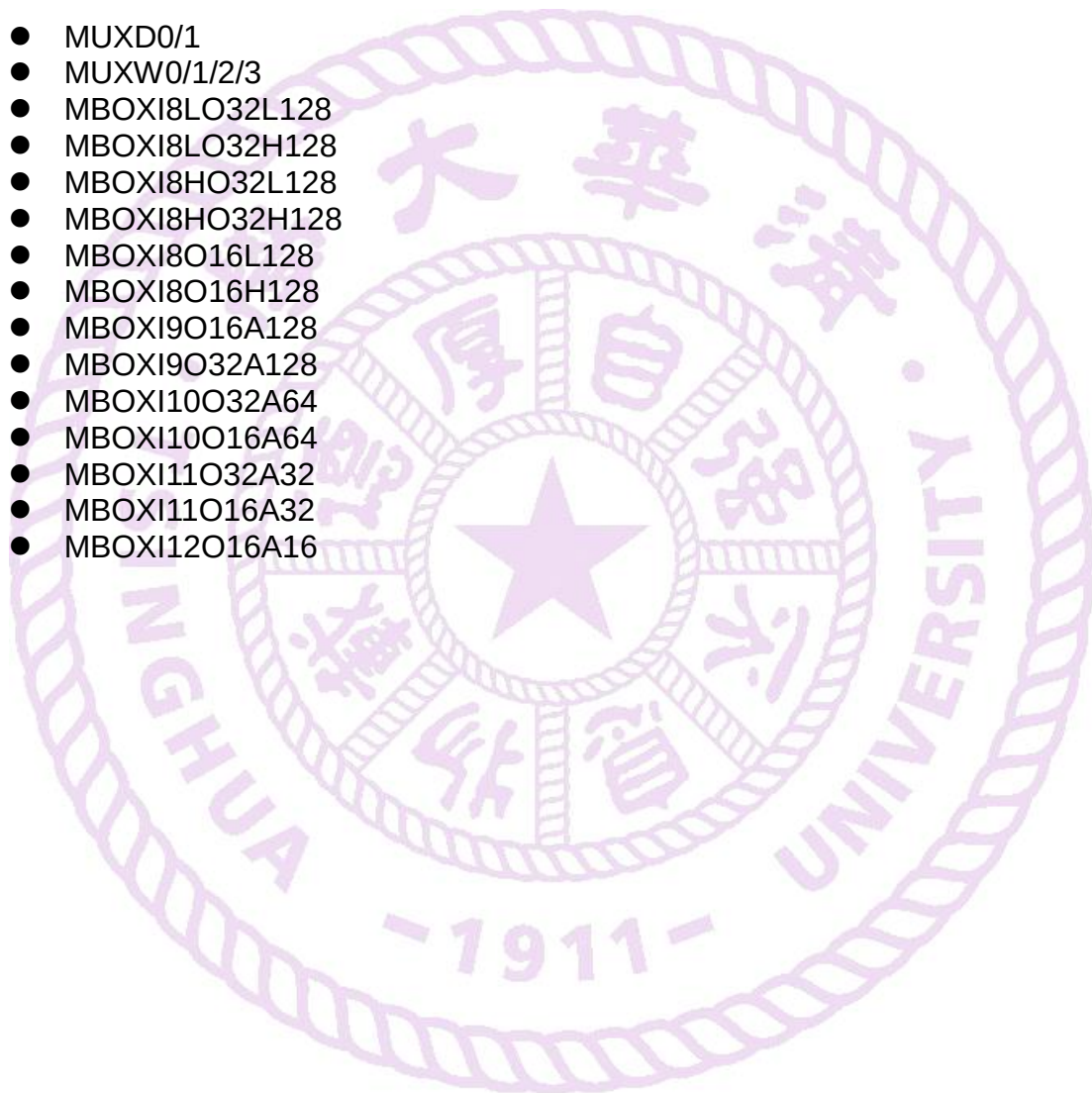
Execute time

1 cycle

17 CONFIGED OPERATIONS

Instruction Summary

- MUXD0/1
- MUXW0/1/2/3
- MBOXI8LO32L128
- MBOXI8LO32H128
- MBOXI8HO32L128
- MBOXI8HO32H128
- MBOXI8O16L128
- MBOXI8O16H128
- MBOXI9O16A128
- MBOXI9O32A128
- MBOXI10O32A64
- MBOXI10O16A64
- MBOXI11O32A32
- MBOXI11O16A32
- MBOXI12O16A16



MUXD0/1

MUXD0/1

Configured Operations

MUXD0/1

General Form **MUXD0/1** (.type) rd, rs /* (D) */
 .type = .A

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op 1				Op 2				rs1		rd		z	creg

MUXD0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.A	011 1110 10110
rs	64-bit	.A	

MUXD1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.A	011 1110 10111
rs	64-bit	.A	

Description

MUXD0/1 指令为可配置的多路选择指令，根据配置编码选择 64 输入 64 输出。

每条 MUXD_x 指令由 16 个 32 位配置寄存器进行配置，分别为 D_x_REG0, D_x_REG1 到 D_x_REG15，每个字节 8 bit 配置 1 位输出，REG0 配置输出最低 4 位，以此类推，REG15 配置输出最高 4 位。对于输出最低位，配置方式如下。

case (Dx REG0[5:0])

```
6'h0 : temp=in[0];
```

```
6'h1 : temp=in[1];
```

```
6'h2 : temp=in[2];
```

```
6'h3 : temp=in[3];
```

• • • • •

```
6'h3d : temp=in[61];
```

```
6'h3e : temp=in[62];
```

```
6'h3f : temp=in[63];
```

endcase

```
out[0] = Dx REG0[6]?~temp:temp;
```

配置寄存器映射到处理器的数据存储空间,可由处理器自身动态配置。

X 簇 D0 配置寄存器地址为 0xXXXXXXXX 开始的 16 个地址

X 簇 D1 配置寄存器地址为 0xXXXXXXXX 开始的 16 个地址

Y 簇 D0 配置寄存器地址为 0xXXXXXXXX 开始的 16 个地址

Y 簇 D1 配置寄存器地址为 0xXXXXXXXX 开始的 16 个地址

Operation
Execute time
1 cycle



Configured Operations

MUXW0/1/2/3

MUXW0/1/2/3

Configured Operations

MUXW0/1/2/3

General Form MUXW0/1/2/3 (.type) rd, rs /* (D) */
.type = .A

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

MUXW0

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	32-bit	.A	011 11110 11000
rs	32-bit	.A	

MUXW1

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	32-bit	.A	011 11110 11001
rs	32-bit	.A	

MUXW2

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	32-bit	.A	011 11110 11010
rs	32-bit	.A	

MUXW3

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	32-bit	.A	011 11110 11011
rs	32-bit	.A	

Description

MUXW0/1/2/3 指令为可配置的多路选择指令，其复用 MUXD_x 的多路选择器，即分别访问 MUXD_x 的高低 32 位，MUXW0 访问 DMUX0 的低 32 位，MUXW1 访问 MUXD0 的高 32 位，MUXW2 访问 MUXD1 的低 32 位，MUXW3 访问 MUXD1 的高 32 位。多路选择器的配置地址和编码和 MUXD_x 一致。

Operation

Execute time

1 cycle

Configured Operations
MBOXI8LO32L128

MBOXI8LO32L128

Configured Operations
MBOXI8LO32L128

General Form MBOXI8LO32L128 (.type) rd, rs /* (D) */
 .type = .M

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

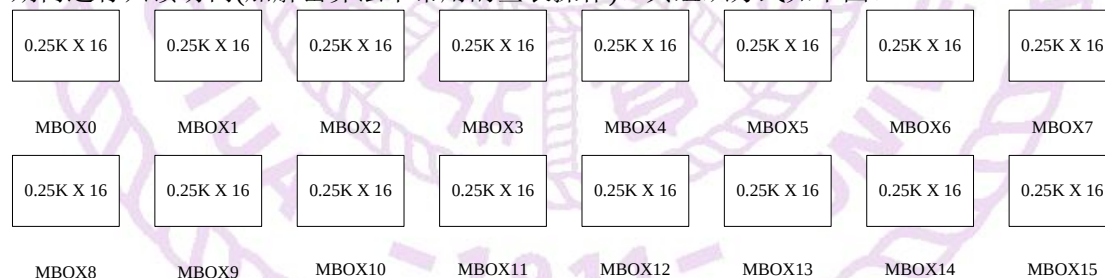
MBOXI8LO32L128

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 10000
rs	64-bit	.M	

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI8LO32L128 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 8 表示输入地址宽度为 8 位(即存储器深度为 0.25KB), 第一个 L 表示输入地址来自源操作数的低 4 字节, O 表示输出, 32 表示每个地址查询的输出数据宽度(即两块存储器使用同一个地址输入, 两个 16 位输出组成一个 32 位输出), 第二个 L 表示输出由存储器的低部分输出(即 MBOX0/1 输出最低 32 位, MBOX2/3 输出次低 32 位, MBOX4/5 输出较高 32 位, MBOX6/7 输出最高 32 位)。最后一个 128 表示输出数据总宽度, 对于 128 位输出, 输出的低 64 位存放于 G 寄存器 rd 中, 高 64 位存放于 G 寄存器 rd+1 中。

Operation
Execute time
2 cycle



Configured Operations

MBOXI8LO32H128

MBOXI8LO32H128

Configd Operations
MBOXI8LO32H128

General Form **MBOXI8LO32H128** (.type) rd, rs /* (D) */
 .type = .M

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

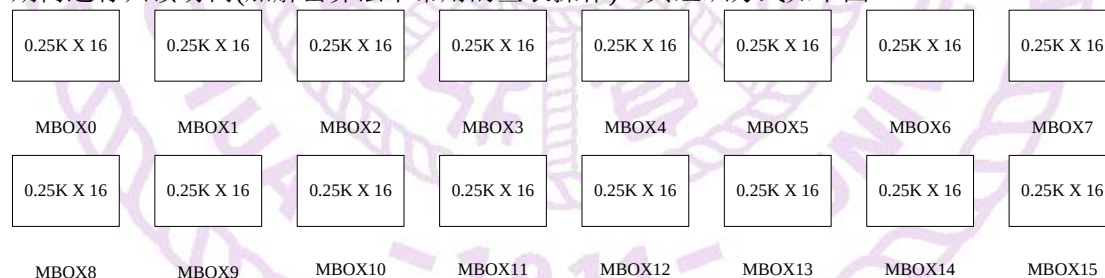
MBOXI8LO32H128

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 10001
rs	64-bit	.M	

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI8LO32H128 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 8 表示输入地址宽度为 8 位(即存储器深度为 0.25KB), 第一个 L 表示输入地址来自源操作数的低 4 字节, O 表示输出, 32 表示每个地址查询的输出数据宽度(即两块存储器使用同一个地址输入, 两个 16 位输出组成一个 32 位输出), 第二个 H 表示输出由存储器的高部分输出(即 MBOX8/9 输出最低 32 位, MBOX10/11 输出次低 32 位, MBOX12/13 输出较高 32 位, MBOX14/15 输出最高 32 位)。最后一个 128 表示输出数据总宽度, 对于 128 位输出, 输出的低 64 位存放于 G 寄存器 rd 中, 高 64 位存放于 G 寄存器 rd+1 中。

Operation
Execute time
2 cycle



Configured Operations
MBOXI8HO32L128

MBOXI8H032L128

Configd Operations
MBOXI8HO32L128

General Form MBOXI8HO32L128 (.type) rd, rs /* (D) */
 .type = .M

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

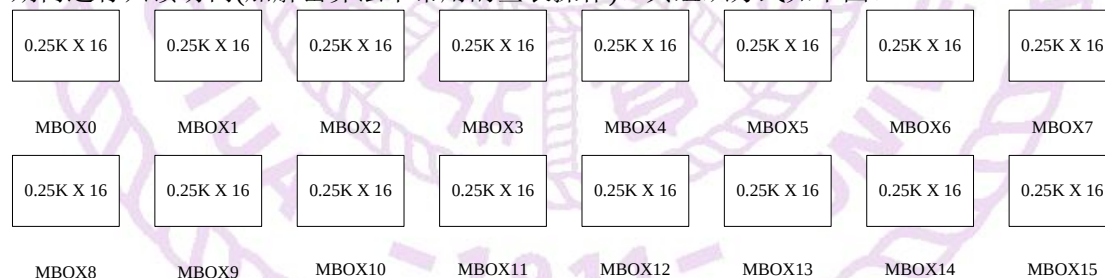
MBOXI8HO32L128

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 10010
rs	64-bit	.M	

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI8LO32L128 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 8 表示输入地址宽度为 8 位(即存储器深度为 0.25KB), 第一个 H 表示输入地址来自源操作数的高 4 字节, O 表示输出, 32 表示每个地址查询的输出数据宽度(即两块存储器使用同一个地址输入, 两个 16 位输出组成一个 32 位输出), 第二个 L 表示输出由存储器的低部分输出(即 MBOX0/1 输出最低 32 位, MBOX2/3 输出次低 32 位, MBOX4/5 输出较高 32 位, MBOX6/7 输出最高 32 位)。最后一个 128 表示输出数据总宽度, 对于 128 位输出, 输出的低 64 位存放于 G 寄存器 rd 中, 高 64 位存放于 G 寄存器 rd+1 中。

Operation
Execute time
2 cycle



Configured Operations

MBOXI8HO32H128

Configured Operations

General Form **MBOXI8LO32H128** (.type) rd, rs
 .type = .M

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

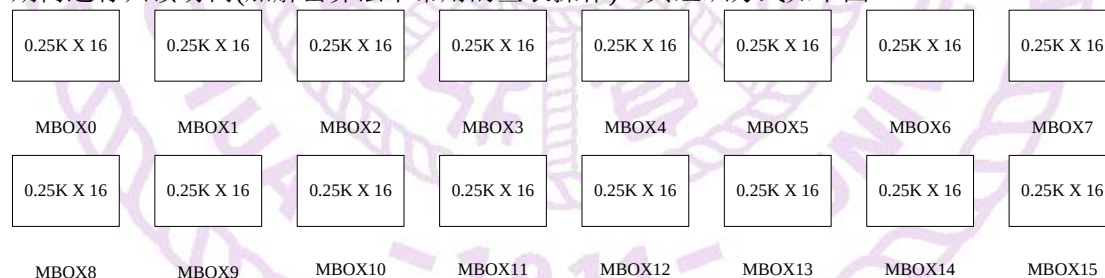
MBOXI8H032H128

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 10011
rs	64-bit	.M	

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI8LO32H128 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 8 表示输入地址宽度为 8 位(即存储器深度为 0.25KB), 第一个 H 表示输入地址来自源操作数的高 4 字节, O 表示输出, 32 表示每个地址查询的输出数据宽度(即两块存储器使用同一个地址输入, 两个 16 位输出组成一个 32 位输出), 第二个 H 表示输出由存储器的高部分输出(即 MBOX8/9 输出最低 32 位, MBOX10/11 输出次低 32 位, MBOX12/13 输出较高 32 位, MBOX14/15 输出最高 32 位)。最后一个 128 表示输出数据总宽度, 对于 128 位输出, 输出的低 64 位存放于 G 寄存器 rd 中, 高 64 位存放于 G 寄存器 rd+1 中。

Operation
Execute time
2 cycle



Configd Operations
MBOXI8O16L128

MBOXI8O16L128

Configured Operations
MBOXI8O16L128

General Form	MBOXI8016L128	(.type) rd, rs	/* (D) */
	.type = .M		

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

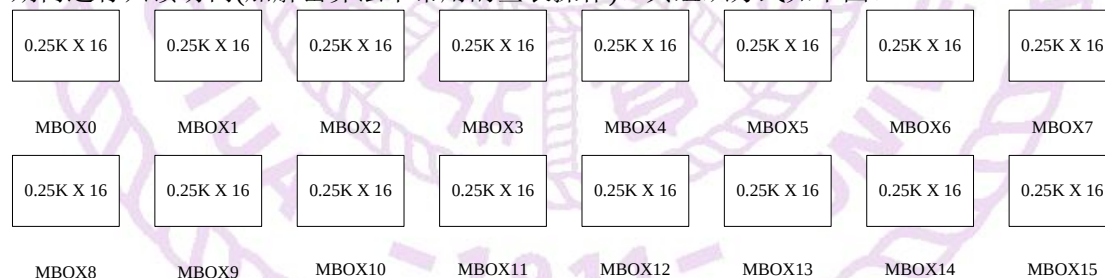
MBOXI8O16L128

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 10100
rs	64-bit	.M	

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI8O16L128 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器 (查表操作), I 表示输入, 8 表示输入地址宽度为 8 位(即存储器深度为 0.25KB), 为输入 G 寄存器的 8 个字节, O 表示输出, 16 表示每个地址查询的输出数据宽度 (即每块存储器使用一个地址输入, 输出 8 个 16 位数据), 第二个 L 表示输出由存储器的低部分输出 (即 MBOX0/1/2/3 输出最低 64 位, MBOX5/6/7/8 输出高 64 位)。最后一个 128 表示输出数据总宽度, 对于 128 位输出, 输出的低 64 位存放于 G 寄存器 rd 中, 高 64 位存放于 G 寄存器 rd+1 中。

Operation

Execute time
2 cycle



Configured Operations
MBOXI8O16H128

Execute time
2 cycle



MBOXI9O16A128

Configured Operations
MBOXI9016A128

General Form	MBOXI9016A128	(.type) rd, rs	/* (D) */
	.type = .M		

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode (D) Format

MA.D1

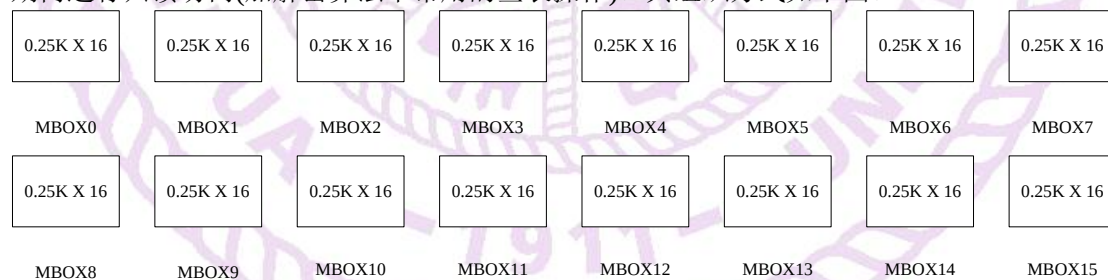
31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				rs2/const5			rs1		rd		z	creg	

MBOXI9016A128 (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	64-bit	.M	010	01100
rs1	64-bit	.M		
rs2	64-bit			

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1ffff000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI9016A128 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器 (查表操作), I 表示输入, 9 表示输入地址宽度为 9 位(即存储器深度为 0.5KB, bit[7:0] 作为两个 MBOX 的地址输入 bit[8] 作为 MBOX 选择位), 为输入 G 寄存器 rs1 和 rs2 的每 16 位的低 9 位, O 表示输出, 16 表示每个地址查询的输出数据宽度 (每两块存储器使用一个地址输入, 16 块存储器输出 8 个 16 位数据), 第二个 A 表示输出由所有存储器输出。rs1[15:0] 的低 9 位选择 MBOX0/1 输出 16 位数据, 其产生伪代码如下:

rs1[7:0]作为 MBOX0/1 的输入地址, rs1[8]作为 MBOX0/1 选择位。

```
MBOX0_addr = rs1[7:0];
```



```
MBOX1_addr = rs1[7:0];
```

```
if(rs1[8])  
    rd[15:0] = MBOX1_out;  
else  
    rd[15:0] = MBOX0_out;
```

与之类似，rs1 分别选择 MBOX0/1，MBOX2/3，MBOX4/5，MBOX6/7 输出低 64 位数据，rs2 分别选择 MBOX8/9，MBOX10/11，MBOX12/13，MBOX14/15 输出高 64 位数据。

最后一个 128 表示输出数据总宽度，对于 128 位输出，输出的低 64 位存放于 G 寄存器 rd 中，高 64 位存放于 G 寄存器 rd+1 中。

Operation

Execute time

2 cycle



Configured Operations MBOXI9O32A128

MBOXI9O32A128

Configured Operations MBOXI9O32A128

General Form MBOXI9O16A128 (.type) rd, rs /* (D) */
.type = .M

Syntax
(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode (D) Format

MA.D2

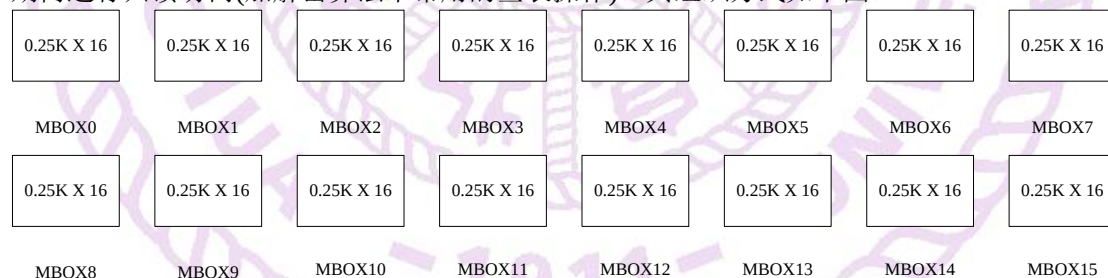
31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs1	rd	z	creg								

MBOXI9O32A128 (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 10110
rs	64-bit	.M	

Description

在 EDSP 内部，有 16 块 $0.25K \times 16$ 共 8KB 的 SRAM，初始化后可由 M 单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化，DMA 初始化 MBOX 时目标地址为数据存储空间 $0x1fff0000$ 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI9O32A128 指令，MBOX 为这一类指令的通用名，表示访问 M 单元内部存储器（查表操作），I 表示输入，9 表示输入地址宽度为 9 位(即存储器深度为 0.5KB，bit[7:0] 作为两组 MBOX 的地址输入，bit[8] 作为 MBOX 选择位)，为输入 G 寄存器 rs 的每 16 位的低 9 位，O 表示输出，32 表示每个地址查询的输出数据宽度（每块 MBOX 输出宽度为 16 位，因此需要两块 MBOX 构成一组输出 32 位），第二个 A 表示输出由所有存储器输出。rs1[15:0] 的低 9 位选择 MBOX0/1/2/3 输出 32 位数据，其产生伪代码如下：

```
rs1[7:0] 作为 MBOX0/1/2/3 的输入地址，rs1[8] 作为组 MBOX0/1, MBOX2/3 选择位。
MBOX0_addr = rs1[7:0];
MBOX1_addr = rs1[7:0];
MBOX2_addr = rs1[7:0];
```

```
MBOX3_addr = rs1[7:0];
```

```
if(rs1[8])  
    rd[31:0] = {MBOX3_out, MBOX1_out};  
else  
    rd[31:0] = {MBOX2_out, MBOX0_out};
```

与之类似，rs 每个 16 位的低 9 位分别 MBOX0/1/2/3，MBOX4/5/6/7，MBOX8/9/10/11，MBOX12/13/14/15 输出 128 位数据。

最后一个 128 表示输出数据总宽度，对于 128 位输出，输出的低 64 位存放于 G 寄存器 rd 中，高 64 位存放于 G 寄存器 rd+1 中。

Operation

Execute time

2 cycle



Configured Operations
MBOXI10O32A64

MBOXI10O32A64

Configured Operations
MBOXI10O32A64

```
General Form      MBOXI10032A64  (.type) rd, rs          /* (D) */
                  .type = .M
```

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode (D) Format

MA.D2

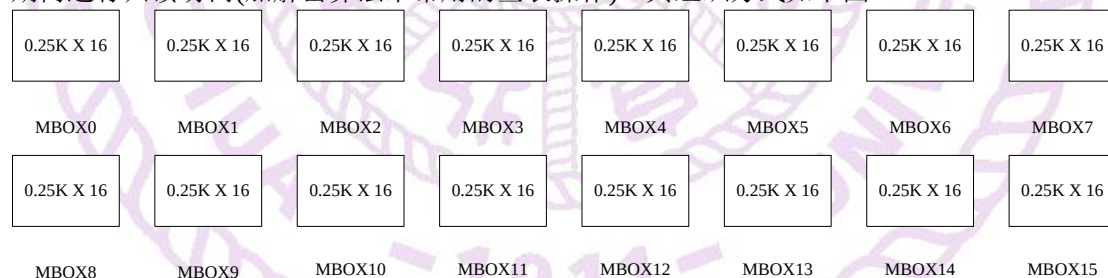
31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

MBOXI10032A64 (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 1011
rs	64-bit	.M	

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(如解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI10032A64 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 10 表示输入地址宽度 10 位(即存储器深度为 1KB, bit[7:0] 作为四组 MBOX 的地址输入, bit[9:8] 作为四组 MBOX 选择位), 为输入 G 寄存器 rs 的每 16 位的低 10 位, O 表示输出, 32 表示每个地址查询的输出数据宽度(每块 MBOX 输出宽度为 16 位, 因此需要两块 MBOX 构成一组输出 32 位), 第二个 A 表示输出由所有存储器输出。rs1[15:0] 的低 10 位选择 MBOX0/1/2/3/4/5/6/7 输出 32 位数据, 其产生伪代码如下:

rs1[7:0]作为 MBOX0/1/2/3/4/5/6/7 的输入地址, rs1[9:8]作为组 MBOX0/4, MBOX1/5, MBOX2/6, MBOX3/7 选择位。

```
MBOX0_addr = rs1[7:0];
```



```

MBOX1_addr = rs1[7:0];
MBOX2_addr = rs1[7:0];
MBOX3_addr = rs1[7:0];
MBOX4_addr = rs1[7:0];
MBOX5_addr = rs1[7:0];
MBOX6_addr = rs1[7:0];
MBOX7_addr = rs1[7:0];

```

```

case (rs[9:8])
  2'b11 : rd[31:0] = {MBOX7_out, MBOX3_out};
  2'b10 : rd[31:0] = {MBOX6_out, MBOX2_out};
  2'b01 : rd[31:0] = {MBOX5_out, MBOX1_out};
  2'b00 : rd[31:0] = {MBOX4_out, MBOX0_out};
endcase

```

endcase

与之类似，rs[31:16]位的低 10 位分别 MBOX8/9/10/11/12/13/14/15 输出 32 位数据。
 最后一个 64 表示输出数据总宽度存放于 G 寄存器 rd 中。

Operation

Execute time

2 cycle



MBOXI10O16A64

Configured Operations
MBOXI10O16A64

```
General Form      MBOXI10016A64  (.type) rd, rs          /* (D) */
                  .type = .M
```

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

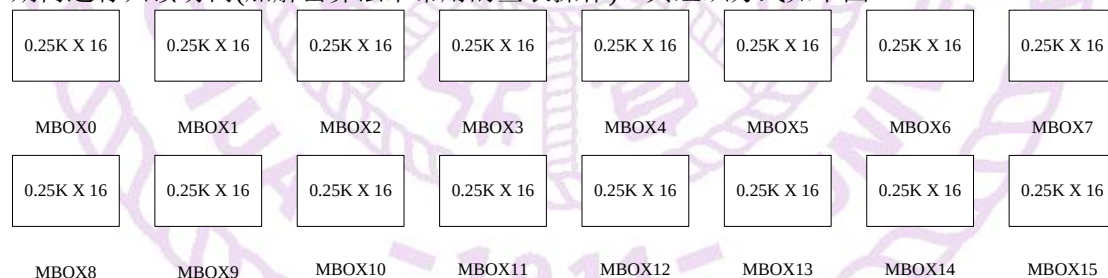
MBOXI10O16A64

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 1111 11000
rs	64-bit	.M	

Description

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI10O16A64 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 10 表示输入地址宽度 10 位(即存储器深度为 1KB, bit[7:0] 作为四组 MBOX 的地址输入, bit[9:8] 作为四组 MBOX 选择位), 为输入 G 寄存器 rs 的每 16 位的低 10 位, O 表示输出, 32 表示每个地址查询的输出数据宽度(每块 MBOX 输出宽度为 16 位, 因此单块 MBOX 作为一组输出 16 位数据), 第二个 A 表示输出由所有存储器输出。rs[15:0] 的低 10 位选择 MBOX0/1/2/3 输出 16 位数据, 其产生伪代码如下:

rs[7:0]作为 MBOX0/1/2/3 的输入地址, rs1[9:8]作为组 MBOX0/1/2/3 选择位。

```
MBOX0_addr = rs1[7:0];
```

```
MBOX1_addr = rs1[7:0];
```

```
MBOX2_addr = rs1[7:0];
```

```
MBOX3_addr = rs1[7:0];
```

```
case (rs[9:8])
```

```
  2'b11 : rd[15:0] = MBOX3_out;
```

```
  2'b10 : rd[15:0] = MBOX2_out;
```

```
  2'b01 : rd[15:0] = MBOX1_out;
```

```
  2'b00 : rd[15:0] = MBOX0_out;
```

```
endcase
```

与之类似，rs 每 16 位的低 10 位分别 MBOX0/1/2/3，MBOX4/5/6/7，MBOX8/9/10/11，MBOX12/13/14/15 输出 64 位数据。

最后一个 64 表示输出数据总宽度存放于 G 寄存器 rd 中。

Operation

Execute time

2 cycle



MBOXI11O32A32

General Form	MBOXI11O32A32 (.type) rd, rs	<i>/* (D) */</i>
	.type = .M	

(D) rd, rs: G0~G7, G Registers

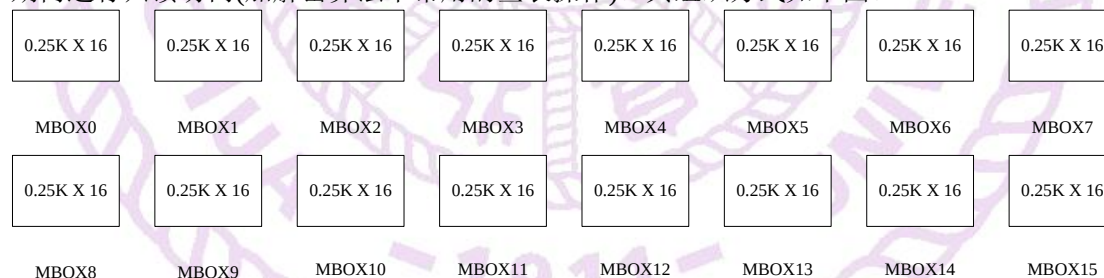
In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	64-bit	.M	011	1111	11001
rs	64-bit	.M			

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI11O32A32 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 11 表示输入地址宽度 11 位(即存储器深度为 2KB, bit[7:0] 作为八组 MBOX 的地址输入, bit[10:9] 作为八组 MBOX 选择位), 为输入 G 寄存器 rs 的每 16 位的低 11 位, O 表示输出, 32 表示每个地址查询的输出数据宽度(每块 MBOX 输出宽度为 16 位, 因此需要两块 MBOX 构成一组输出 32 位), 第二个 A 表示输出由所有存储器输出。rs1[15:0] 的低 11 位选择所有 MBOX 输出 32 位数据, 其产生伪代码如下:

rs1[7:0]作为所有 MBOX 的输入地址，rs1[10:8]作为八组 MBOX 的选择位。

```
MBOX0 addr = rs1[7:0];
```

```
MBOX1_addr = rs1[7:0];
```

```
MBOX2_addr = rs1[7:0];
```



```

MBOX3_addr = rs1[7:0];
MBOX4_addr = rs1[7:0];
MBOX5_addr = rs1[7:0];
MBOX6_addr = rs1[7:0];
MBOX7_addr = rs1[7:0];
MBOX8_addr = rs1[7:0];
MBOX9_addr = rs1[7:0];
MBOXa_addr = rs1[7:0];
MBOXb_addr = rs1[7:0];
MBOXc_addr = rs1[7:0];
MBOXd_addr = rs1[7:0];
MBOXe_addr = rs1[7:0];
MBOXf_addr = rs1[7:0];

```

```

case (rs[10:8])
  2'b111 : rd[31:0] = {MBOXf_out, MBOX7_out};
  2'b110 : rd[31:0] = {MBOXe_out, MBOX6_out};
  2'b101 : rd[31:0] = {MBOXd_out, MBOX5_out};
  2'b100 : rd[31:0] = {MBOXc_out, MBOX4_out};
  2'b011 : rd[31:0] = {MBOXb_out, MBOX3_out};
  2'b010 : rd[31:0] = {MBOXa_out, MBOX2_out};
  2'b001 : rd[31:0] = {MBOX9_out, MBOX1_out};
  2'b000 : rd[31:0] = {MBOX8_out, MBOX0_out};

```

endcase

最后一个 32 表示输出数据总宽度存放于 rd 中。

Operation

Execute time

2 cycle

MBOXI11O16A32

General Form	MBOXI11O16A32 (.type) rd, rs	<i>/* (D) */</i>
	.type = .M	

(D) rd, rs: G0~G7, G Registers

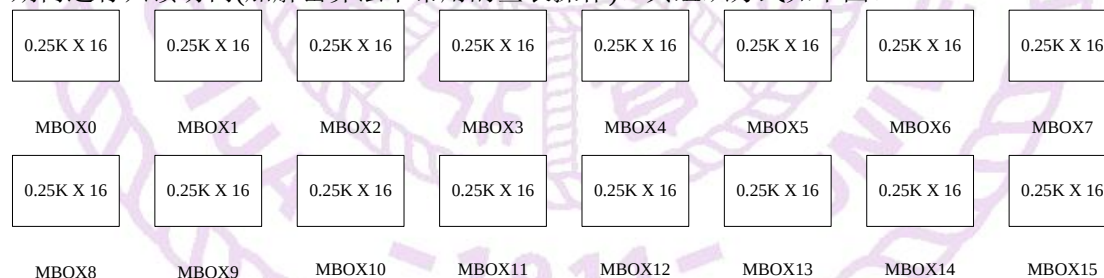
In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op 1				Op 2				rs1		rd		z	creg

Opcode map field used ...	For operand type ...	Unit	Opfield		
rd	64-bit	.M	011	1111	11010
rs	64-bit	.M			

在EDSP内部,有16块 $0.25K \times 16$ 共8KB的SRAM,初始化后可由M单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化, DMA 初始化 MBOX 时目标地址为数据存储空间 0x1fff0000 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI11O16A32 指令, MBOX 为这一类指令的通用名, 表示访问 M 单元内部存储器(查表操作), I 表示输入, 11 表示输入地址宽度 11 位(即存储器深度为 2KB, bit[7:0] 作为八组 MBOX 的地址输入, bit[10:8] 作为八组 MBOX 选择位), 为输入 G 寄存器 rs 的每 16 位的低 11 位, O 表示输出, 32 表示每个地址查询的输出数据宽度(每块 MBOX 输出宽度为 16 位, 因此单块 MBOX 作为一组输出 16 位数据), 第二个 A 表示输出由所有存储器输出。rs[15:0] 的低 10 位选择 MBOX0/1/2/3/4/5/6/7 输出 16 位数据, 其产生伪代码如下:

rs[7:0]作为 MBOX0/1/2/3/4/5/6/7 的输入地址, rs1[10:8]作为八组 MBOX 选择位。

```
MBOX0  addr = rs1[7:0];
```

```
MBOX1_addr = rs1[7:0];
```

```
MBOX2_addr = rs1[7:0];
```

```

MBOX3_addr = rs1[7:0];
MBOX4_addr = rs1[7:0];
MBOX5_addr = rs1[7:0];
MBOX6_addr = rs1[7:0];
MBOX7_addr = rs1[7:0];

```

```

case (rs[10:8])
  2'b111 : rd[31:0] = MBOX7_out;
  2'b110 : rd[31:0] = MBOX6_out;
  2'b101 : rd[31:0] = MBOX5_out;
  2'b100 : rd[31:0] = MBOX4_out;
  2'b011 : rd[31:0] = MBOX3_out;
  2'b010 : rd[31:0] = MBOX2_out;
  2'b001 : rd[31:0] = MBOX1_out;
  2'b000 : rd[31:0] = MBOX0_out;

```

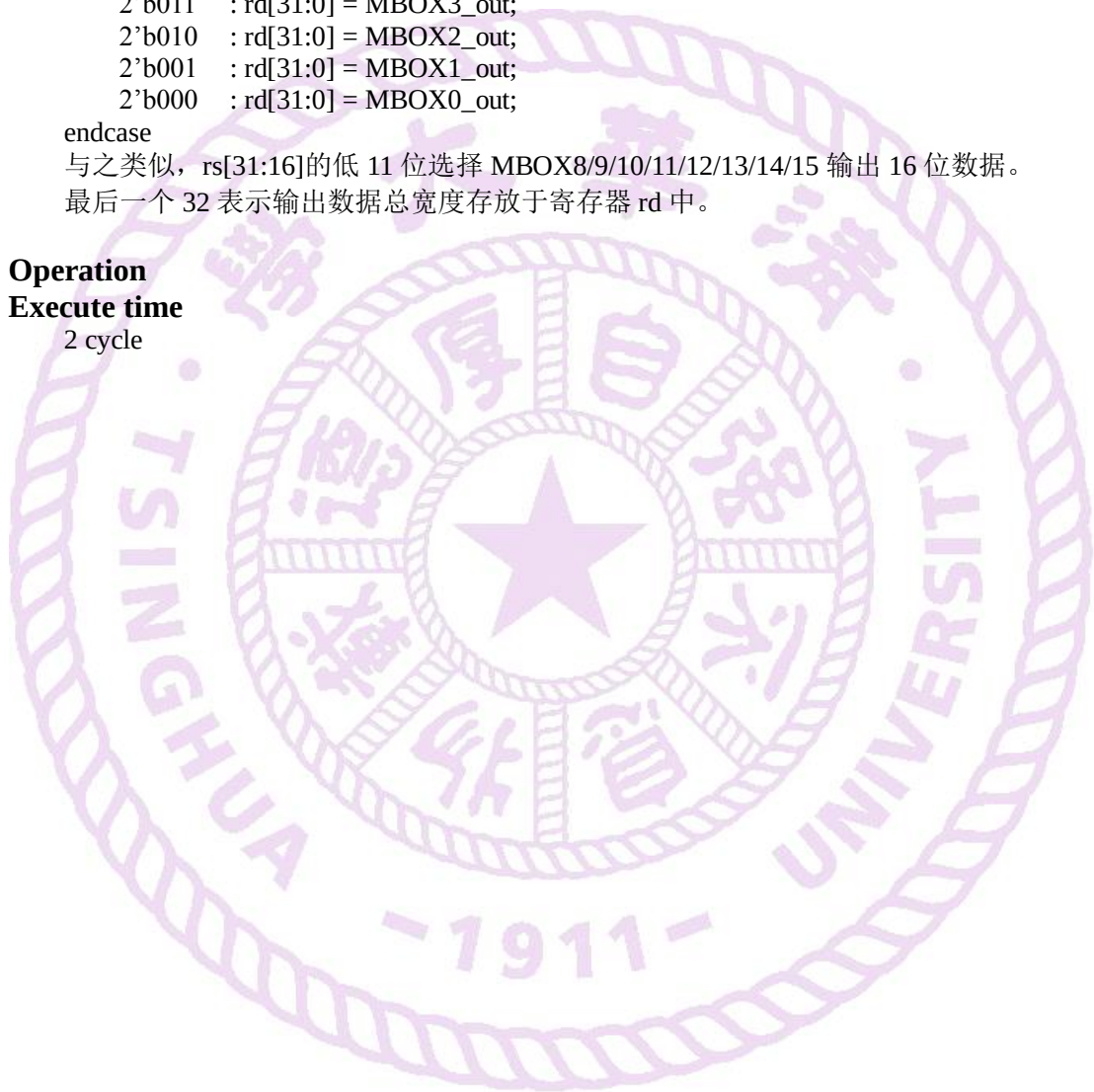
endcase

与之类似，rs[31:16]的低 11 位选择 MBOX8/9/10/11/12/13/14/15 输出 16 位数据。
 最后一个 32 表示输出数据总宽度存放于寄存器 rd 中。

Operation

Execute time

2 cycle



Configured Operations MBOXI12O16A16

MBOXI12O16A16

Configured Operations MBOXI12O16A16

General Form MBOXI12O16A16 (.type) rd, rs /* (D) */
.type = .M

Syntax
(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode (D) Format

MA.D2

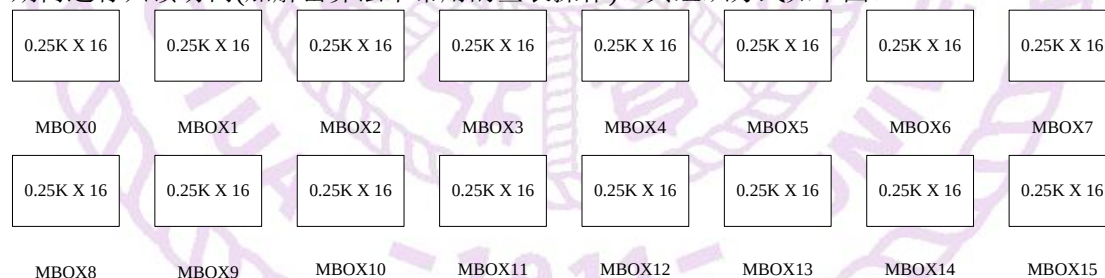
31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs1	rd	z	creg								

MBOXI12O16A16 (D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.M	011 11111 11011
rs	64-bit	.M	

Description

在 EDSP 内部，有 16 块 $0.25K \times 16$ 共 8KB 的 SRAM，初始化后可由 M 单元在两个周期内进行只读访问(加解密算法中常用的查表操作)。其组织方式如下图。



MBOX 存储器在处理器初始化阶段由 HOST 配置 DMA 进行初始化，DMA 初始化 MBOX 时目标地址为数据存储空间 $0x1fff0000$ 开始的 8KB 地址。地址增长方式为 MBOX0[0], MBOX1[0], ..., MBOX15[0], MBOX0[1], MBOX1[1], ..., MBOX15[1], ..., MBOX0[255], MBOX1[255], ..., MBOX15[255]。

对于 MBOXI12O16A16 指令，MBOX 为这一类指令的通用名，表示访问 M 单元内部存储器（查表操作），I 表示输入，12 表示输入地址宽度 12 位(即存储器深度为 4KB, bit[7:0] 作为十六组 MBOX 的地址输入，bit[11:9]作为十六组 MBOX 选择位)，为输入 G 寄存器 rs 的每 16 位的低 12 位，O 表示输出，32 表示每个地址查询的输出数据宽度（每块 MBOX 输出宽度为 16 位，因此每组只需一块 MBOX 构成输出 16 位），第二个 A 表示输出由所有存储器输出。rs1[15:0]的低 12 位选择所有 MBOX 输出 16 位数据，其产生伪代码如下：

```
rs1[7:0]作为所有 MBOX 的输入地址，rs1[11:8]作为十六组 MBOX 的选择位。
MBOX0_addr = rs1[7:0];
MBOX1_addr = rs1[7:0];
MBOX2_addr = rs1[7:0];
```



```

MBOX3_addr = rs1[7:0];
MBOX4_addr = rs1[7:0];
MBOX5_addr = rs1[7:0];
MBOX6_addr = rs1[7:0];
MBOX7_addr = rs1[7:0];
MBOX8_addr = rs1[7:0];
MBOX9_addr = rs1[7:0];
MBOXa_addr = rs1[7:0];
MBOXb_addr = rs1[7:0];
MBOXc_addr = rs1[7:0];
MBOXd_addr = rs1[7:0];
MBOXe_addr = rs1[7:0];
MBOXf_addr = rs1[7:0];

```

```

case (rs[11:8])
  2'b1111 : rd[15:0] = MBOXf_out;
  2'b1110 : rd[15:0] = MBOXe_out;
  2'b1101 : rd[15:0] = MBOXd_out;
  2'b1100 : rd[15:0] = MBOXc_out;
  2'b1011 : rd[15:0] = MBOXb_out;
  2'b1010 : rd[15:0] = MBOXa_out;
  2'b1001 : rd[15:0] = MBOX9_out;
  2'b1000 : rd[15:0] = MBOX8_out;
  2'b0111 : rd[15:0] = MBOX7_out;
  2'b0110 : rd[15:0] = MBOX6_out;
  2'b0101 : rd[15:0] = MBOX5_out;
  2'b0100 : rd[15:0] = MBOX4_out;
  2'b0011 : rd[15:0] = MBOX3_out;
  2'b0010 : rd[15:0] = MBOX2_out;
  2'b0001 : rd[15:0] = MBOX1_out;
  2'b0000 : rd[15:0] = MBOX0_out;

```

endcase

最后一个 16 表示输出数据总宽度存放于 rd 中。

Operation

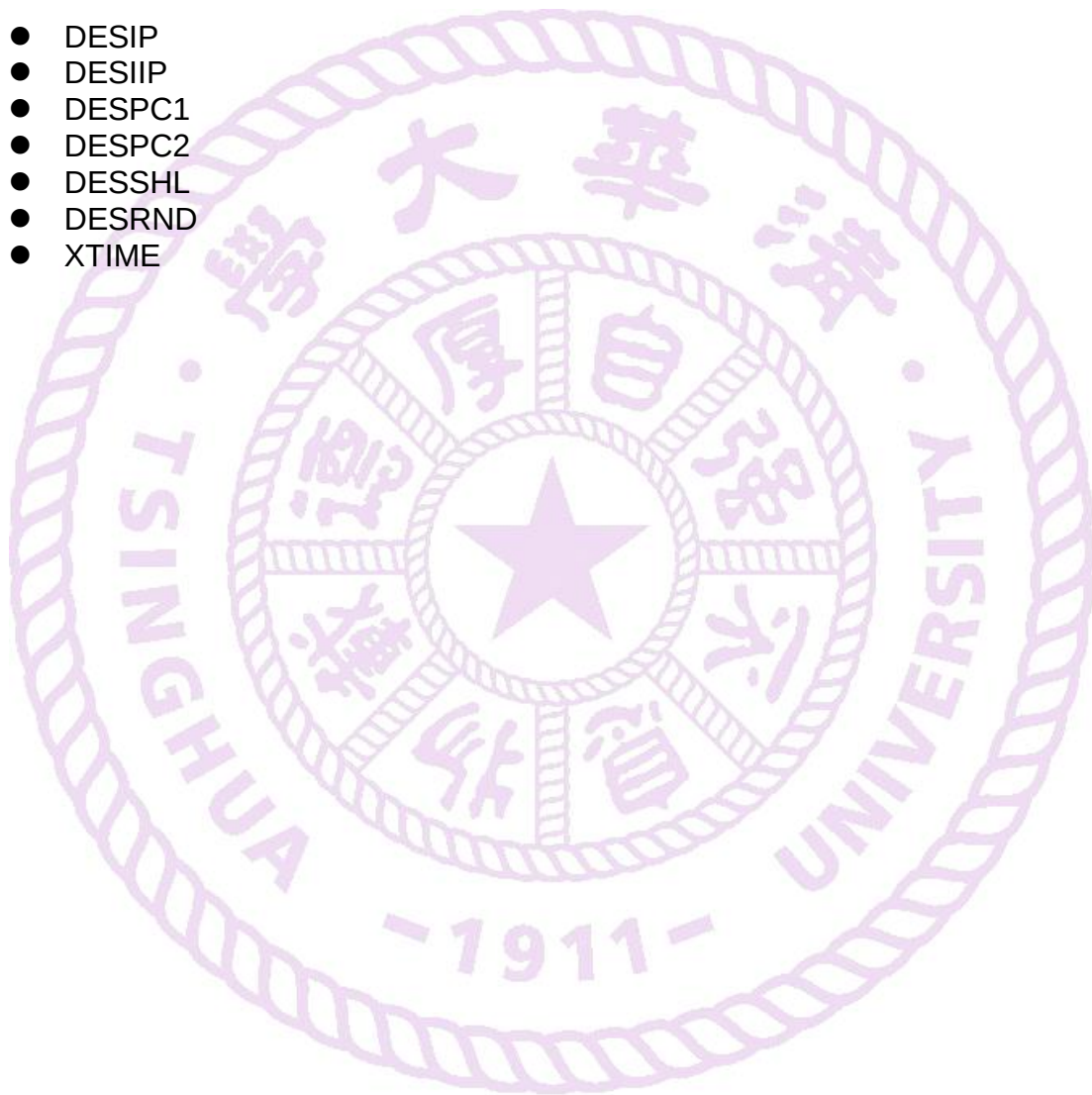
Execute time

2 cycle

18 CRYPTON OPERATIONS

Instruction Summary

- DESIP
- DESIIP
- DESPC1
- DESPC2
- DESSHL
- DESRND
- XTIME



Cryption Operations
DESIP

DESIP

Cryption Operations
DESIP

General Form **DESIP** (.type) rd, rs /* (D) */
 .type = .A

Syntax
 (D) rd, rs: G0~G7, G Registers

Instruction Length
 In the syntax, comment (F) identifies 16-bit instruction length; comment (D)
 identifies 32-bit instruction length.

Opcode
(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

DESIP	(D) Format
Opcode map field used ...	For operand type ... Unit Opfield
rd	64-bit .A 011 1111 1110
rs	64-bit .A

Description
 这条指令的作用是将 G 寄存器 rs 中的 64 位数按下表所示方式重排，下表中第一个 58
 为目标第 0 位，最后一个 7 为目标第 63 位。rs 低 32 位对应下表中的高 32 个数字，rs 高 32
 位对应下表中低 32 个数字，rd 低 32 位存放目标高 32 位，rd 高 32 位存放目标低 32 位。
 the initial permutation *IP*:

IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Operation
Execute time
 1 cycle

Cryption Operations
DESIIP

DESIIP

Cryption Operations
DESIIP

General Form DESIIP (.type) rd, rs /* (D) */
 .type = .A

Syntax
 (D) rd, rs: G0~G7, G Registers

Instruction Length
 In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode
(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

DESIIP	(D) Format
Opcode map field used ...	For operand type ...
rd	64-bit
rs	64-bit

Description
 这条指令的作用是将 G 寄存器 rs 中的 64 位数按下表所示方式重排，下表中第一个 58 为目标第 0 位，最后一个 7 为目标第 63 位。rs 低 32 位对应下表中的高 32 个数字，rs 高 32 位对应下表中低 32 个数字，rd 低 32 位存放目标高 32 位，rd 高 32 位存放目标低 32 位。

IP¹

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Operation
Execute time
 1 cycle

Cryption Operations
DESPC1

DESPC1

Cryption Operations
DESPC1

General Form DESPC1 (.type) rd, rs /* (D) */
 .type = .A

Syntax
 (D) rd, rs: G0~G7, G Registers

Instruction Length
 In the syntax, comment (F) identifies 16-bit instruction length; comment (D)
 identifies 32-bit instruction length.

Opcode
(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

DESPC1	(D) Format
Opcode map field used ...	For operand type ... Unit Opfield
rd	64-bit .A 011 1111 11010
rs	64-bit .A

Description
 这条指令的作用是将 G 寄存器 rs 中的 64 位数按下表所示方式重排，下表中第一个 58 为目标第 0 位，最后一个 7 为目标第 63 位。rs 低 32 位对应下表中的高 32 个数字，rs 高 32 位对应下表中低 32 个数字，rd 低 32 位存放目标高 32 位，rd 高 32 位存放目标低 32 位。

PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Operation
Execute time
 1 cycle

Cryption Operations
DESPC2

DESPC2

Cryption Operations
DESPC2

General Form DESPC2 (.type) rd, rs /* (D) */
 .type = .A

Syntax
 (D) rd, rs: G0~G7, G Registers

Instruction Length
 In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode
(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1	Op_2	rs1	rd	z	creg								

DESPC2	(D) Format
Opcode map field used ...	For operand type ...
rd	64-bit
rs	64-bit

Description
 这条指令的作用是将 G 寄存器 rs 中的 64 位数按下表所示方式重排，下表中第一个 58 为目标第 0 位，最后一个 7 为目标第 63 位。rs 低 32 位对应下表中的高 32 个数字，rs 高 32 位对应下表中低 32 个数字，rd 低 32 位存放目标高 32 位，rd 高 32 位存放目标低 32 位。

PC-2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Operation
Execute time
 1 cycle

DESSLDESSL

Cryption Operations

DESSL

General Form **DESSL** (.type) rd, rs1, rs2

/* (D) */

```
.type = .A
```

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				rs2/const5				rs1		rd		z	creg

DESSL

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.A	110	01100
rs1	sint	.A		
rs2	sint			

Description

这条指令的作用是把寄存器 rs1 中的 32 位无符号数的低 28 位循环左移 1 位或 2 位, rs2 的最低位为 0 则左移 1 位, 否则左移 2 位。

Operation

Execute time

1 cycle

DESRND

DESRND

Cryption Operations

DESRND

General Form	DESRND (.type) rd, rs1, rs2	/* (D) */

```
.type = .M
```

Syntax

(D) rd, rs: G0~G7, G Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2/const5				rs1		rd		z	creg

DESRND

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield	
rd	sint	.M	010	01101
rs1	sint	.M		
rs2	sint			

Description

输入 rs1 为 DES 每轮操作的中间数, rs2 为 DES 每轮操作的轮密钥, 这条指令的作用是将完成一轮 DES 运算, 并将结果写入目标寄存器 rd。

Operation

Execute time

1 cycle

XTIME

XTIME**XTIME**

General Form	XTIME (.type) rd, rs	/* (D) */
	.type = .A	

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

(D) Format

AL.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	00	Op	Op_1				Op_2				rs1		rd		z	creg

XTIME

(D) Format

Opcode map field used ...	For operand type ...	Unit	Opfield
rd	64-bit	.A	011 1111 01101
rs	64-bit	.A	

Description

这条指令的作用是将 rs 的高 8 位经过 xtime() 运算后结果写入 rd, xtime 运算定义如下:

```
reg[7:0]      RC;
xtime(RC)=(RC<<1)^{3'b0,{2{RC[7]}},1'b0,{2{RC[7]}}};//xtime(RC);
```

Operation

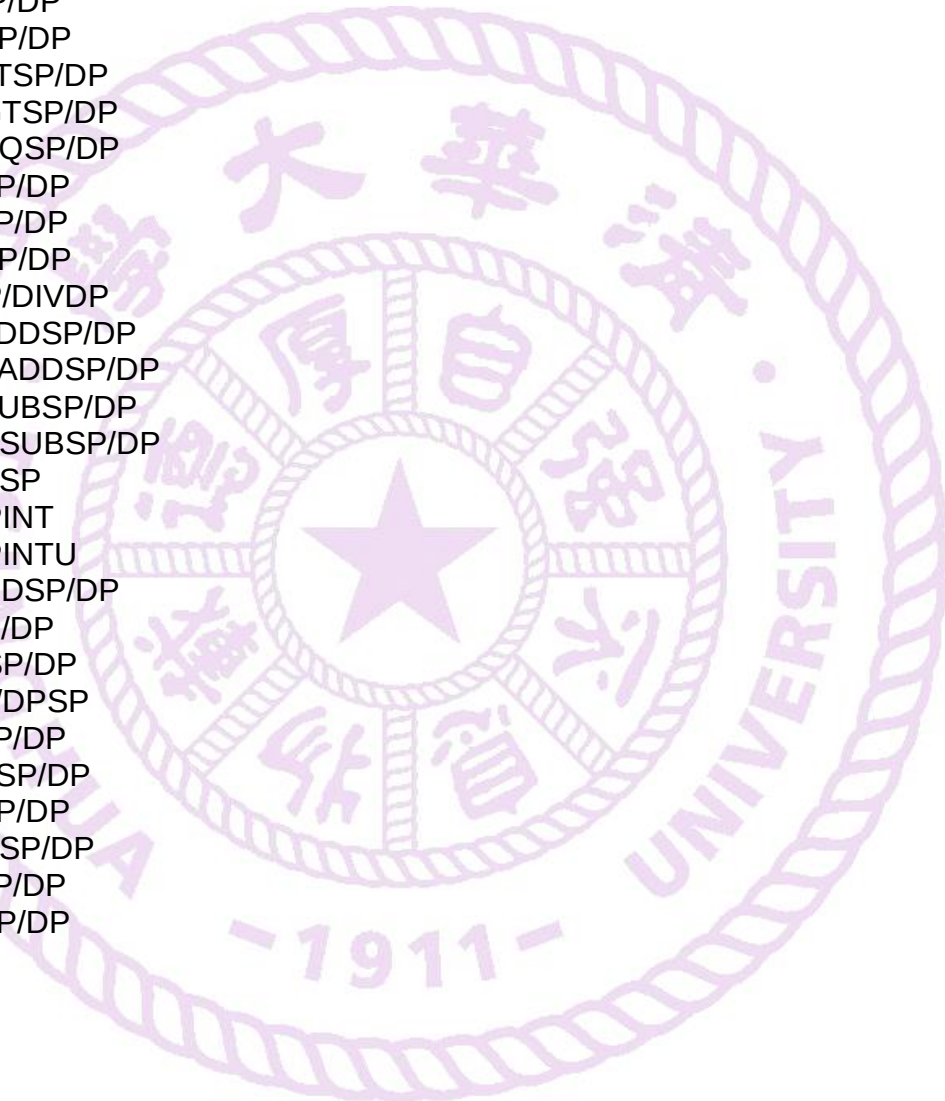
Execute time

1 cycle

19 FLOATING POINT OPERATIONS

Instruction Summary

- MINSP/DP
- MAXSP/DP
- CMPLTSP/DP
- CMPGTSP/DP
- CMPEQSP/DP
- ADDSP/DP
- SUBSP/DP
- MPYSP/DP
- DIVSP/DIVDP
- MPYADDSP/DP
- NMPYADDSP/DP
- MPYSUBSP/DP
- NMPYSUBSP/DP
- CMPYSP
- SP/DPINT
- SP/DPINTU
- ROUNDSP/DP
- INTSP/DP
- INTUSP/DP
- SPDP/DPSP
- ABSSP/DP
- SQRTSP/DP
- RCPSP/DP
- RSQRSP/DP
- EXPSP/DP
- LOGSP/DP



Floating Point Operations
MINSP/DP Minimum Floating Point

Floating Point Operations
MINSP/DP

General Form **MINSP** (.type) rd, rs1, rs2 /* (D) */
MINDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		rs2		rs1		rd		z	creg				

SWAP.2 (D) Format

Rd	sp	.M	100	00000
Rs1	sp			
Rs2	sp			
Rd_Pair	dp	.M	100	10000
Rs_Pair1	dp			
Rs_Pair2	dp			

Description
单精度情况下，将 Rs1 和 Rs2 中较小的数放在 Rd 中。
双精度情况下，将 Rs_Pair1 和 Rs_Pair2 中的较小的数放在 Rd_Pair 中。

Operation
(D) If (cond) rd = min(rs1, rs2)
Elsenop
(D) If (cond) rd_pair = min(rs_pair1, rs_pair2)
Elsenop

Execute time
C. cycle

Floating Point Operations
MAXSP/DP Maximum Floating Point

Floating Point Operations
MAXSP/DP

General Form **MAXSP** (.type) rd, rs1, rs2 /* (D) */
MAXDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		rs2		rs1		rd		z	creg				

SWAP.2		(D) Format			
Rd	sp	.M	100	00001	
Rs1	sp				
Rs2	sp				
Rd_Pair	dp	.M	100	10001	
Rs_Pair1	dp				
Rs_Pair2	dp				

Description
单精度情况下，将 Rs1 和 Rs2 中较大的数放在 Rd 中。
双精度情况下，将 Rs_Pair1 和 Rs_Pair2 中的较大的数放在 Rd_Pair 中。

Operation
(D) If (cond) rd = max(rs1, rs2)
Elsenop
(D) If (cond) rd_pair = max(rs_pair1, rs_pair2)
Elsenop

Execute time
D. cycle

Floating Point Operations

CMPLTSP/DP

Compare for Less Than Floating Point

Floating Point Operations

CMPLTSP/DP

General Form **CMPLTSP** (.type) rd, rs1, rs2 /* (D) */
CMPLTDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2		rs1		rd		z		creg	

SWAP.2

(D) Format

Rd	sp	.M	100	00010
Rs1	sp			
Rs2	sp			
Rd_Pair	dp	.M	100	10010
Rs_Pair1	dp			
Rs_Pair2	dp			

Description

单精度情况下，若 Rs1 小于 Rs2，则 Rd 为 1，反之为 0

双精度情况下，若 Rs_Pair1 小于 Rs_Pair2，则 Rd_Pair 为 1，反之为 0

Operation

(D) If (cond) rd =(rs1 < rs2) ? 1 : 0

Elsenop

(D) If (cond) rd_pair =(rs_pair1 < rs_pair2) ? 1 : 0

Elsenop

Execute time

E. cycle

Floating Point Operations

CMPGTSP/DP

Compare for Greater Than Floating Point

Floating Point Operations

CMPGTSP/DP

General Form **CMPGTSP** (.type) rd, rs1, rs2 /* (D) */
CMPGTDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	rs2	rs1	rd	z	creg								

SWAP.2

(D) Format

Rd	sp	.M	100	00011
Rs1	sp			
Rs2	sp			

Rd_Pair	dp	.M	100	10011
Rs_Pair1	dp			
Rs_Pair2	dp			

Description

单精度情况下，若 Rs1 大于 Rs2，则 Rd 为 1，反之为 0

双精度情况下，若 Rs_Pair1 大于 Rs_Pair2，则 Rd_Pair 为 1，反之为 0

Operation

(D) If (cond) rd =(rs1 > rs2) ? 1 : 0
Elsenop

(D) If (cond) rd_pair =(rs_pair1 > rs_pair2) ? 1 : 0
Elsenop

Execute time

F. cycle

Floating Point Operations

CMPEQSP/DP Compare for Equality Floating Point

Floating Point Operations

CMPEQSP/DP

General Form **CMPGTSP** (.type) rd, rs1, rs2 /* (D) */
CMPGTDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2		rs1		rd		z		creg	

SWAP.2

(D) Format

Rd	sp	.M	100	00100
Rs1	sp			
Rs2	sp			
Rd_Pair	dp	.M	100	10100
Rs_Pair1	dp			
Rs_Pair2	dp			

Description

单精度情况下，若 Rs1 等于 Rs2，则 Rd 为 1，反之为 0

双精度情况下，若 Rs_Pair1 等于 Rs_Pair2，则 Rd_Pair 为 1，反之为 0

Operation

(D) If (cond) rd =(rs1 == rs2) ? 1 : 0

Elsenop

(D) If (cond) rd_pair =(rs_pair1 == rs_pair2) ? 1 : 0

Elsenop

Execute time

G. cycle

Floating Point Operations

ADDSP/DP

Addition Floating Point

Floating Point Operations

ADDSP/DP

General Form **ADDSP** (.type) rd, rs1, rs2 /* (D) */
ADDDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2		rs1				rd	z	creg	

SWAP.2

(D) Format

Rd	sp	.M	100	00101
Rs1	sp			
Rs2	sp			
Rd_Pair	dp	.M	100	10101
Rs_Pair1	dp			
Rs_Pair2	dp			

Description

单精度情况下，将 Rs1 与 Rs2 相加，结果存放在 Rd 之中。

双精度情况下，将 Rs_Pair1 与 Rs_Pair2 相加，结果存放在 Rd 之中。

Operation

(D) If (cond) rd = rs1 + rs2

Elsenop

(D) If (cond) rd_pair = rs_pair1 + rs_pair2

Elsenop

Execute time

H. cycle

Floating Point Operations

SUBSP/DP

Subtraction Floating Point

Floating Point Operations

SUBSP/DP

General Form

SUBSP (.type) rd, rs1, rs2 /* (D) */
SUBDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
.type = .M,

Syntax

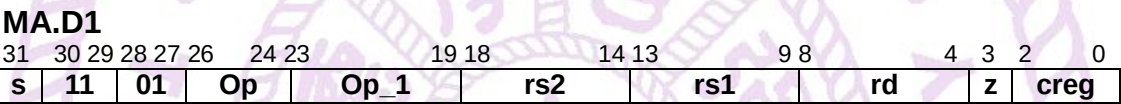
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format



SWAP.2		(D) Format			
Rd		sp		.M	100 00110
Rs1		sp			
Rs2		sp			
Rd_Pair		dp		.M	100 10110
Rs_Pair1		dp			
Rs_Pair2		dp			

Description

单精度情况下，将 Rs1 与 Rs2 相减，结果存放在 Rd 之中。
双精度情况下，将 Rs_Pair1 与 Rs_Pair2 相减，结果存放在 Rd 之中。

Operation

(D) If (cond) rd = rs1 - rs2
Elsenop
(D) If (cond) rd_pair = rs_pair1 - rs_pair2
Elsenop

Execute time

l. cycle

Floating Point Operations

MPYSP/DP

Multiplication Floating Point

Floating Point Operations

MPYSP/DP

General Form

MPYSP (.type) rd, rs1, rs2 /* (D) */
 MPYDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		rs2		rs1		rd		z	creg				

SWAP.2

(D) Format

Rd	sp	.M	100	00111
Rs1	sp			
Rs2	sp			

Rd_Pair	dp	.M	100	10111
Rs_Pair1	dp			
Rs_Pair2	dp			

Description

单精度情况下，将 Rs1 与 Rs2 相乘，结果存放在 Rd 之中。
 双精度情况下，将 Rs_Pair1 与 Rs_Pair2 的相乘，结果存放在 Rd_Pair 之中。

Operation

(D) If (cond) rd = rs1 * rs2
 Elsenop
 (D) If (cond) rd_pair = rs_pair1 * rs_pair2
 Elsenop

Execute time

J. cycle

Floating Point Operations

DIVSP/DP

Division Floating Point

Floating Point Operations

DIVSP/DP

General Form

DIVSP (.type) rd, rs1, rs2

/* (D) */

DIVDP (.type) rd_pair, rs_pair1, rs_pair2

/* (D) */

.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2		rs1				rd	z	creg	

SWAP.2

(D) Format

Rd	sp	.M	100	01000
Rs1	sp			
Rs2	sp			

Rd_Pair	dp	.M	100	11000
Rs_Pair1	dp			
Rs_Pair2	dp			

Description

单精度情况下，将 Rs1 与 Rs2 相除，结果存放在 Rd 之中。

双精度情况下，将 Rs_Pair1 与 Rs_Pair2 的相除，结果存放在 Rd_Pair 之中。

Operation

(D) If (cond) rd = rs1 / rs2

Elsenop

(D) If (cond) rd_pair = rs_pair1 / rs_pair2

Elsenop

Execute time

K. cycle

Floating Point Operations
MPYADDSP/DP Fused Mac Floating Point

Floating Point Operations
MPYADDSP/DP

General Form **MPYADDSP** (.type) rd, rs1, rs2 /* (D) */
MPYADDDP (.type) rd_pair, rs_pair1, rs_pair2 /* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2		rs1		rd		z		creg	

SWAP.2		(D) Format			
Rd	sp	.M	100	01001	
Rs1	sp				
Rs2	sp				
Rd_Pair	dp	.M	100	11001	
Rs_Pair1	dp				
Rs_Pair2	dp				

Description
单精度情况下，将 Rs1 与 Rs2 相乘再和 Rd 相加，结果存放在 Rd 之中。
双精度情况下，将 Rs_Pair1 与 Rs_Pair2 的相乘再和 Rd_Pair 相加，结果存放在 Rd_Pair 之中。

Operation
(D) If (cond) rd += rs1 * rs2
 Elsenop
(D) If (cond) rd_pair += rs_pair1 * rs_pair2
 Elsenop

Execute time
L. cycle

Floating Point Operations
NMPYADDSP/DP

Negative Fused Mac Floating Point

Floating Point Operations
NMPYADDSP/DP

General Form
NMPYADDSP (.type) rd, rs1, rs2
/* (D) */
NMPYADDSP (.type) rd_pair, rs_pair1, rs_pair2/* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2		rs1				rd	z	creg	

SWAP.2

(D) Format

Rd	sp	.M	100	01010
Rs1	sp			
Rs2	sp			
Rd_Pair	dp	.M	100	11010
Rs_Pair1	dp			
Rs_Pair2	dp			

Description
单精度情况下，将 Rs1 与 Rs2 相乘再和 Rd 相加之后再取相反数，结果存放在 Rd 之中。
双精度情况下，将 Rs_Pair1 与 Rs_Pair2 的相乘再和 Rd_Pair 相加之后再取相反数，结果存放在 Rd_Pair 之中。

Operation
(D) If (cond)
Elsenop
rd = -(rd + rs1 * rs2)
(D) If (cond)
Elsenop
rd_pair = -(rd_pair + rs_pair1 * rs_pair2)

Execute time
M. cycle

Floating Point Operations
MPYSUBSP/DP Fused Mac Floating Point

Floating Point Operations
MPYSUBSP/DP

General Form **MPYSUBSP** (.type) rd, rs1, rs2 /* (D) */
MPYSUBDP (.type) rd_pair, rs_pair1, rs_pair2/* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. **Format**

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1				rs2		rs1		rd		z		creg	

SWAP.2		(D) Format			
Rd	sp	.M	100	01011	
Rs1	sp				
Rs2	sp				
Rd_Pair	dp	.M	100	11011	
Rs_Pair1	dp				
Rs_Pair2	dp				

Description
单精度情况下，将 Rs1 与 Rs2 相乘再和 Rd 相减，结果存放在 Rd 之中。
双精度情况下，将 Rs_Pair1 与 Rs_Pair2 的相乘再和 Rd_Pair 相减，结果存放在 Rd_Pair 之中。

Operation
(D) If (cond) rd = -(rd - rs1 * rs2)
Elsenop
(D) If (cond) rd_pair = -(rd_pair - rs_pair1 * rs_pair2)
Elsenop

Execute time
N. cycle

Floating Point Operations

Floating Point Operations

NMPYSUBSP/DP

Negative Fused Mac Floating Point

NMPYSUBSP/DP

General Form

NMPYSUBSP (.type) rd, rs1, rs2

/* (D) */

NMPYSUBDP (.type) rd_pair, rs_pair1, rs_pair2/* (D) */

.type = .M,

Syntax

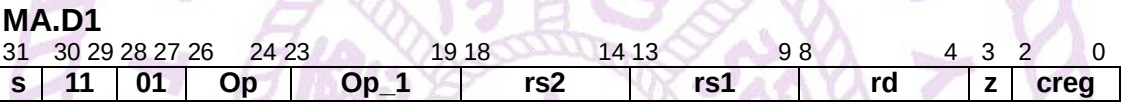
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format



SWAP.2		(D) Format			
Rd	sp	.M	100	01100	
Rs1	sp				
Rs2	sp				
Rd_Pair	dp	.M	100	11100	
Rs_Pair1	dp				
Rs_Pair2	dp				

Description

单精度情况下，将 Rs1 与 Rs2 相乘再和 Rd 相减之后再取相反数，结果存放在 Rd 之中。

双精度情况下，将 Rs_Pair1 与 Rs_Pair2 的相乘再和 Rd_Pair 相减之后再取相反数，结果存放在 Rd_Pair 之中。

Operation

(D) If (cond)

rd = (rd - rs1 * rs2)

Elsenop

(D) If (cond)

rd_pair = (rd_pair - rs_pair1 * rs_pair2)

Elsenop

Execute time

O. cycle

Floating Point Operations

ABSSP/DP

Absolute Value Floating Point

Floating Point Operations

ABSSP/DP

General Form **ABSSP** (.type) rd, rs /* (D) */
 ABSDP (.type) rd_pair, rs_pair /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	sp	.M	011	00100	00000
Rs	sp				
Rd_Pair	dp	.M	011	00110	00000
Rs_Pair	dp				

Description

单精度情况下，求 Rs 的绝对值，将结果存放在 Rd 中。

双精度情况下，求 Rs_Pair 的绝对值，将结果存放在 Rd_Pair 中。

Operation

(D) If (cond) rd =abs(rs)
 Elsenop

(D) If (cond) rd_pair = abs(rs_pair)
 Elsenop

Execute time

P. cycle

Floating Point Operations SQRTSP/DP Square Root Floating Point

Floating Point Operations SQRTSP/DP

General Form **SQRTSP** (.type) rd, rs /* (D) */
SQRTDP (.type) rd_pair, rs_pair /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	sp	.M	011	00100	00001
Rs	sp				

Rd_Pair	dp	.M	011	00110	00001
Rs_Pair	dp				

Description

单精度情况下，求 Rs 的平方根，将结果存放在 Rd 中。

双精度情况下，求 Rs_Pair 的平方根，将结果存放在 Rd_Pair 中。

Operation

(D) If (cond) rd =sqrt(rs)

Elsenop

(D) If (cond) rd_pair = sqrt(rs_pair)

Elsenop

Execute time

Q. cycle

Floating Point Operations
RCPSP/DP

Reciprocal Approximation Floating Point

Floating Point Operations
RCPSP/DP

General Form **RCPSP** (.type) rd, rs /* (D) */
 RCPDP (.type) rd_pair, rs_pair /* (D) */
 .type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2 (D) Format

Rd	sp	.M	011	00100	00010
Rs	sp				
Rd_Pair	dp	.M	011	00110	00010
Rs_Pair1	dp				

Description

单精度情况下，求 Rs 的倒数，将结果存放在 Rd 中。
双精度情况下，求 Rs_Pair 的倒数，将结果存放在 Rd_Pair 中。

Operation

(D) If (cond) rd =1/rs
 Elsenop
(D) If (cond) rd_pair = 1/rs_pair
 Elsenop

Execute time

R. cycle

Floating Point Operations
RSQRSP/DP

Reciprocal Square Root Floating Point
RSQRSP/DP

General Form **RSQRSP** (.type) rd, rs /* (D) */
RSQRDP (.type) rd_pair, rs_pair /* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31

30

29

28

27

26

24

23

19

18

14

13

9

8

4

3

2

0

s

11

01

Op

Op_1

Op_2

rs

rd

z

creg

SWAP.2		(D) Format					
Rd	sp	.M	011	00100	00011		
Rs	sp						
Rd_Pair	dp	.M	011	00110	00011		
Rs_Pair	dp						

Description
单精度情况下，求 Rs 的倒数平方根，将结果存放在 Rd 中。
双精度情况下，求 Rs_Pair 的倒数平方根，将结果存放在 Rd_Pair 中。

Operation
(D) If (cond) rd =sqrt(1/rs)
Elsenop
(D) If (cond) rd_pair = sqrt(1/rs_pair)
Elsenop

Execute time
S. cycle

General Form **EXPSP** (.type) rd, rs /* (D) */
 EXPDP (.type) rd_pair, rs_pair /* (D) */
 .type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2 (D) Format

Rd	sp	.M	011	00100	00100
Rs	sp				
Rd	sp	.M	011	00100	00101
Rs	sp				
Rd	sp	.M	011	00100	00110
Rs	sp				
Rd_Pair	dp	.M	011	00110	00100
Rs_Pair	dp				
Rd_Pair	dp	.M	011	00110	00101
Rs_Pair	dp				
Rd_Pair	dp	.M	011	00110	00110
Rs_Pair	dp				

Description

Op_2 编码为 00100 时，底数为 2
Op_2 编码为 00101 时，底数为 e
Op_2 编码为 00110 时，底数为 10
单精度情况下，将 Rs 作为指数计算，将结果放入 Rd 中。
双精度情况下，将 Rs_Pair 作为指数计算，将结果存放 Rd_Pair 中。

Operation

(D) If (cond) $rd = 2^{rs}$ or e^{rs} or 10^{rs}
 Elsenop

(D) If (cond) $rd\_pair = 2^{rs\_pair}$ or e^{rs_pair} or 10^{rs_pair}
 Elsenop

Execute time

T. cycle



Floating Point Operations
LOGSP/DP Log Floating Point

Floating Point Operations
LOGSP/DP

General Form **LOGSP** (.type) rd, rs /* (D) */
LOGDP (.type) rd_pair, rs_pair/* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	sp	.M	011	00100	00111
Rs	sp				
Rd	sp	.M	011	00100	01000
Rs	sp				
Rd	sp	.M	011	00100	01001
Rs	sp				
Rd_Pair	dp	.M	011	00110	00111
Rs_Pair	dp				
Rd_Pair	dp	.M	011	00110	01000
Rs_Pair	dp				
Rd_Pair	dp	.M	011	00110	01001
Rs_Pair	dp				

Description

Op_2 编码为 00111 时，底数为 2
Op_2 编码为 01000 时，底数为 e
Op_2 编码为 01001 时，底数为 10
单精度情况下，将 Rs 作为真数计算，将结果存放入 Rd 中。
双精度情况下，将 Rs_Pair 作为真数计算，将结果存放 Rd_Pair 中。

Operation

(D) If (cond)	$rd = \log_2 rs$ or $\ln rs$ or $\lg rs$
Elsenop	
(D) If (cond)	$rd = \log_2 rs\_pair$ or $\ln rs\_pair$ or $\lg rs\_pair$
Elsenop	

Execute time

U. cycle



Floating Point Operations

SP/DPINT

convert Floating Point to integer

Floating Point Operations

SP/DPINT

General Form

```

SPINT (.type) rd, rs          /* (D) */
SPINT (.type) rd_pair, rs     /* (D) */
DPINT (.type) rd, rs_pair     /* (D) */
DPINT (.type) rd_pair, rs_pair /* (D) */
.type = .M,
    
```

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	word	.M	011	00000	00000
Rs	sp				
Rd	word	.M	011	00000	00001
Rs	sp				
Rd	word	.M	011	00000	00010
Rs	sp				
Rd	word	.M	011	00000	00011
Rs	sp				
Rd_Pair	doubleword	.M	011	00000	00100
Rs	sp				
Rd_Pair	doubleword	.M	011	00000	00101
Rs	sp				
Rd_Pair	doubleword	.M	011	00000	00110
Rs	sp				
Rd_Pair	doubleword	.M	011	00000	00111
Rs	sp				
Rd	word	.M	011	00001	00000
Rs_Pair	dp				
Rd	word	.M	011	00001	00001
Rs_Pair	dp				
Rd	word	.M	011	00001	00010

Rs_Pair	dp				
Rd	word	.M	011	00001	00011
Rs_Pair	dp				
Rd_Pair	doubleword	.M	011	00001	00100
Rs_Pair	dp				
Rd_Pair	doubleword	.M	011	00001	00101
Rs_Pair	dp				
Rd_Pair	doubleword	.M	011	00001	00110
Rs_Pair	dp				
Rd_Pair	doubleword	.M	011	00001	00111
Rs_Pair	dp				

Description

Op_2 编码末 2 位为 00 时，Round Near

Op_2 编码末 2 位为 01 时，Round Zero

Op_2 编码末 2 位为 10 时，Round Positive Infinite

Op_2 编码末 2 位为 11 时，Round Negative Infinite

单精度情况下，将 Rs 取整以整数形式存放入 Rd 或者 Rd_Pair 中。

双精度情况下，将 Rs_Pair 取整以整数形式存放入 Rd 或者 Rd_Pair 中。

Operation

- (D) If (cond) rd = *(int *)&(round(rs))
 Elsenop
- (D) If (cond) rd_pair = *(int *)&(round(rs))
 Elsenop
- (D) If (cond) rd = *(int *)&(round(rs_pair))
 Elsenop
- (D) If (cond) rd_pair = *(int *)&(round(rs_pair))
 Elsenop

Execute time

V. cycle

Floating Point Operations

SP/DPINTU

convert Floating Point to unsigned integer

Floating Point Operations

SP/DPINTU

General Form

```

SPINTU (.type) rd, rs          /* (D) */
SPINTU (.type) rd_pair, rs     /* (D) */
DPINTU (.type) rd, rs_pair     /* (D) */
DPINTU (.type) rd_pair, rs_pair /* (D) */
.type = .M,
    
```

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	word	.M	011	00000	01000
Rs	sp				
Rd	word	.M	011	00000	01010
Rs	sp				
Rd	word	.M	011	00000	01011
Rs	sp				
Rd_Pair	doubleword	.M	011	00000	01100
Rs	sp				
Rd_Pair	doubleword	.M	011	00000	01110
Rs	sp				
Rd_Pair	doubleword	.M	011	00000	01111
Rs	sp				
Rd	word	.M	011	00001	01000
Rs_Pair	dp				
Rd	word	.M	011	00001	01010
Rs_Pair	dp				
Rd	word	.M	011	00001	01011
Rs_Pair	dp				
Rd_Pair	doubleword	.M	011	00001	01100
Rs_Pair	dp				

Rd_Pair	doubleword	.M	011	00001	01110
Rs_Pair	dp				
Rd_Pair	doubleword	.M	011	00001	01111
Rs_Pair	dp				

Description

Op_2 编码末 2 位为 00 时, Round Near

Op_2 编码末 2 位为 10 时, Round Positive Infinite

Op_2 编码末 2 位为 11 时, Round Negative Infinite

单精度情况下, 将 Rs 取整以整数形式存放入 Rd 或者 Rd_Pair 中。

双精度情况下, 将 Rs_Pair 取整以整数形式存放入 Rd 或者 Rd_Pair 中。

Operation

(D) If (cond) rd = *(int *)&(round(rs))

 Elsenop

(D) If (cond) rd_pair = *(int *)&(round(rs))

 Elsenop

(D) If (cond) rd = *(int *)&(round(rs_pair))

 Elsenop

(D) If (cond) rd_pair = *(int *)&(round(rs_pair))

 Elsenop

Execute time

W. cycle

Floating Point Operations

ROUNDSP/DP Round Floating Point

Floating Point Operations

ROUNDSP/DP

General Form **ROUNDSP** (.type) rd, rs /* (D) */
ROUNDSP (.type) rd_pair, rs_pair /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	sp	.M	011	00000	10000
Rs	sp				
Rd	sp	.M	011	00000	10001
Rs	sp				
Rd	sp	.M	011	00000	10010
Rs	sp				
Rd	sp	.M	011	00000	10011
Rs	sp				

Rd	dp	.M	011	00001	10000
Rs_Pair	dp				
Rd	dp	.M	011	00001	10001
Rs_Pair	dp				
Rd	dp	.M	011	00001	10010
Rs_Pair	dp				
Rd	dp	.M	011	00001	10011
Rs_Pair	dp				

Description

Op_2 编码末 2 位为 00 时, Round Near

Op_2 编码末 2 位为 01 时, Round Zero

Op_2 编码末 2 位为 10 时, Round Positive Infinite

Op_2 编码末 2 位为 11 时, Round Negative Infinite

单精度情况下, 将 Rs 取整以浮点形式放入 Rd 中。

双精度情况下, 将 Rs_Pair 取整以浮点形式放入 Rd_Pair 中。

Operation

(D) If (cond) $rd = (\text{round}(rs))$

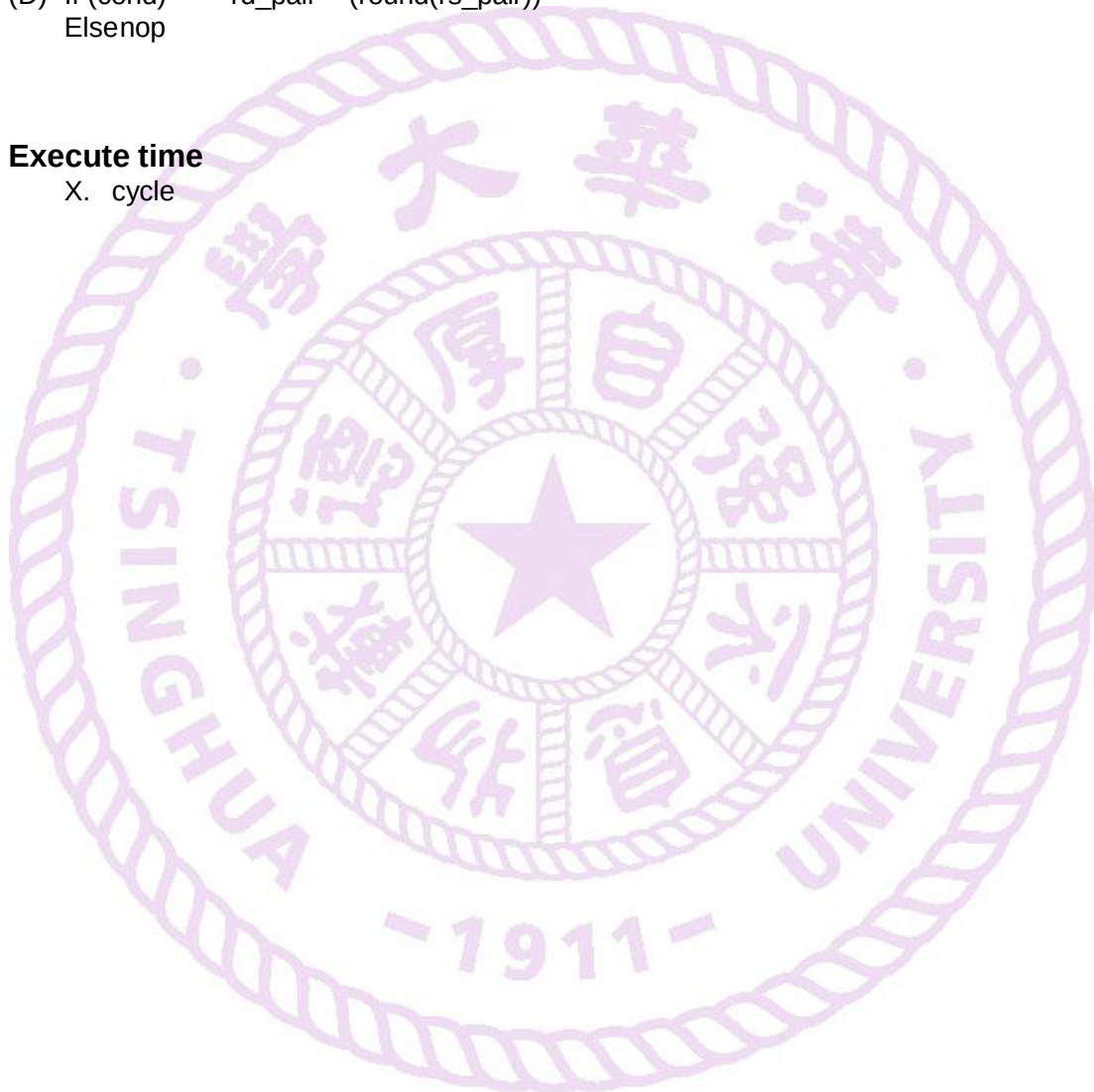
 Elsenop

(D) If (cond) $rd\_pair = (\text{round}(rs\_pair))$

 Elsenop

Execute time

X. cycle



Floating Point Operations

INTSP/DP

Convert Integer to Floating Point

Floating Point Operations

INTSP/DP

General Form

```

INTSP (.type) rd, rs          /* (D) */
INTSP (.type) rd, rs_pair     /* (D) */
INTDP (.type) rd_pair, rs     /* (D) */
INTDP (.type) rd_pair, rs_pair /* (D) */
.type = .M,
    
```

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	sp	.M	011	00000	10100
Rs	word				
Rd	sp	.M	011	00000	10101
Rs_Pair	doubleword				
Rd_Pair	dp	.M	011	00001	10100
Rs	word				
Rd_Pair	dp	.M	011	00001	10101
Rs_Pair	doubleword				

Description

单精度情况下，将 Rs 或者 Rs_Pair 以浮点形式存放入 Rd 中。

双精度情况下，将 Rs 或者 Rs_Pair 以浮点形式存放入 Rd_Pair 中。

Operation

(D) If (cond) $rd = *(float *)\&(rs)$

```

Elsenop
(D) If (cond)    rd = *(float *)&(rs_pair)
Elsenop
(D) If (cond)    rd_pair = *(double *)&(rs)
Elsenop
(D) If (cond)    rd_pair = *(double *)&(rs_pair)
Elsenop

```

Execute time

Y. cycle



Floating Point Operations

INTUSP/DP

Convert Unsigned Integer to Floating Point

Floating Point Operations

INTUSP/DP

General Form

```

INTSP (.type) rd, rs          /* (D) */
INTSP (.type) rd, rs_pair     /* (D) */
INTDP (.type) rd_pair, rs     /* (D) */
INTDP (.type) rd_pair, rs_pair /* (D) */
.type = .M,
    
```

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd	sp	.M	011	00000	10110
Rs	word				
Rd	sp	.M	011	00000	10111
Rs_Pair	doubleword				

Rd_Pair	dp	.M	011	00001	10110
Rs	word				
Rd_Pair	dp	.M	011	00001	10111
Rs_Pair	doubleword				

Description

单精度情况下，将 Rs 或者 Rs_Pair 以浮点形式存放入 Rd 中。

双精度情况下，将 Rs 或者 Rs_Pair 以浮点形式存放入 Rd_Pair 中。

Operation

(D) If (cond) $rd = *(float *)\&(rs)$


```

Elsenop
(D) If (cond)    rd = *(float *)&(rs_pair)
Elsenop
(D) If (cond)    rd_pair = *(double *)&(rs)
Elsenop
(D) If (cond)    rd_pair = *(double *)&(rs_pair)
Elsenop

```

Execute time

Z. cycle



Floating Point Operations

SPDP/DPSP

Convert Floating Point Format

Floating Point Operations

SPDP/DPSP

General Form **INTSP** (.type) rd, rs_pair /* (D) */
 INTDP (.type) rd_pair, rs /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

B. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	rs	rd	z	creg								

SWAP.2

(D) Format

Rd_Pair	dp	.M	011	00000	11000
Rs	sp				
Rd	sp	.M	011	00001	11000
Rs_Pair	dp				

Description

将 Rs 中的单精度数转化为双精度模式存入 Rd_Pair 中。
 或者将 Rs_Pair 中的双精度数转化为单精度模式存入 Rd 中。

Operation

(D) If (cond) rd = *(float *)&(rs)
 Elsenop
 (D) If (cond) rd = *(float *)&(rs_pair)
 Elsenop
 (D) If (cond) rd_pair = *(double *)&(rs)
 Elsenop
 (D) If (cond) rd_pair = *(double *)&(rs_pair)
 Elsenop

Execute time

AA. cycle



20 FLOATING POINT VECTOR OPERATIONS

Instruction Summary

- QMINSP/DMINDP
- QMAXSP/DMAXDP
- QCMPLTSP/DCMPLTDP
- QCMPGTSP/DCMPGTDP
- QCMPEQSP/DCMPEQDP
- QADDSP/DADDDP
- QSUBSP/DSUBDP
- QMPYSP/DMPYDP
- QDIVSP/DDIVDP
- QMPYADDSP/DMPYADDDP
- QNMPYADDSP/DNMPYADDDP
- QMPYSUBSP/DMPYSUBDP
- QNMPYSUNSP/DMPYSUBDP
- CMPYSP
- QABSSP/DABSDP
- QSQRTSP/DSQRTDP
- QRCPS/DRCPDP
- QRSQRSP/DRSQRP
- QEXPSP/DEXPDP
- QLOGSP/DLOGDP
- DSP/DPINT
- DSP/DPUINT
- DROUNDSP/DP
- DINTSP/DP
- DINTUSP/DP
- DSPDP/DDPSP

Floating Point Operations
QMINSP/DMINDP

2/4 Way Minimum Floating Point

Floating Point Operations
QMINSP/DMINDP

General Form

QMINSP (.type) gd, gs1, gs2 /* (D) */
DMINDP (.type) gd, gs1, gs2 /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	gs2	gs1	gd	z	creg								

SWAP.2 (D) Format

Gd	sp	.M	101	00000
Gs1	sp			
Gs2	sp			

Gd	dp	.M	101	10000
Gs1	dp			
Gs2	dp			

Description

SIMD 指令，将 Gs1 和 Gs2 中较小的数放在 Gd 中。

Operation

(D) If (cond) Gd = min(Gs1, Gs2)

Execute time

BB. cycle

Floating Point Operations

QMAXSP/DMAXDP 2/4 Ways Maximum Floating Point

Floating Point Operations

QMAXSP/DMAXDP

General Form **QMAXSP** (.type) gd, gs1, gs2 /* (D) */
DMAXDP (.type) gd, gs1, gs2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	gs2	gs1	gd	z	creg								

SWAP.2

(D) Format

gd	sp	.M	100	00001
gs1	sp			
gs2	sp			

gd	dp	.M	100	10001
gs1	dp			
gs2	dp			

Description

SIMD 指令，将 Gs1 和 Gs2 中较大的数放在 Gd 中。

Operation

(D) If (cond) gd = max(gs1, gs2)
 Elsenop
 (D) If (cond) gd = max(gs1, gs2)
 Elsenop

Execute time

CC. cycle

Floating Point Operations

QCMPLTSP/DCMPLTDP 2/4 Ways Compare for Less Than Floating Point

Floating Point Operations

QCMPLTSP/DCMPLTDP

General Form **QCMPLTSP** (.type) gd, gs1, gs2 /* (D) */
DCMPLTDP (.type) gd, gs1, gs2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	gs2	gs1	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	100	00010
Gs1	sp			
Gs2	sp			
Gd	dp	.M	100	10010
Gs1	dp			
Gs2	dp			

Description

单精度情况下，若 Rs1 小于 Rs2，则 Rd 为 1，反之为 0

双精度情况下，若 Rs_Pair1 小于 Rs_Pair2，则 Rd_Pair 为 1，反之为 0

Operation

(D) If (cond) rd =(rs1 < rs2) ? 1 : 0
 Elsenop

(D) If (cond) rd_pair =(rs_pair1 < rs_pair2) ? 1 : 0
 Elsenop

Execute time

DD. cycle

Floating Point Operations

QCMPGTSP/QCMPGTDP

2/4 Ways
Compare for
Greater Than
Floating Point

Floating Point Operations

QCMPGTSP/DCMPGTDP

General Form **QCMPGTSP** (.type) gd, gs1, gs2 /* (D) */
 DCMPGTDP (.type) gd, gs1, gs2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		gs2		gs1		gd		z	creg				

SWAP.2

(D) Format

Gd	sp	.M	100	00011
Gs1	sp			
Gs2	sp			
Gd	dp	.M	100	10011
Gs1	dp			
Gs2	dp			

Description

单精度情况下，若 Rs1 大于 Rs2，则 Rd 为 1，反之为 0

双精度情况下，若 Rs_Pair1 大于 Rs_Pair2，则 Rd_Pair 为 1，反之为 0

Operation

(D) If (cond) rd =(rs1 > rs2) ? 1 : 0

Elseenop

(D) If (cond) rd_pair =(rs_pair1 > rs_pair2) ? 1 : 0

Elseenop

Execute time

EE. cycle



Floating Point Operations

QCMPEQSP/DCMPEQDP

2/4 Ways
Compare for
Equality
Floating Point

Floating Point Operations

QCMPEQSP/DCMPEQDP

General Form **QCMPEQSP** (.type) gd, gs1, gs2 /* (D) */
 DCMPEQDP (.type) gd, gs1, gs2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	gs2	gs1	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	100	00100
Gs1	sp			
Gs2	sp			
Gd	dp	.M	100	10100
Gs1	dp			
Gs2	dp			

Description

SIMD 指令，若 Gs1 等于 Gs2，则 Gd 为 1，反之为 0

Operation

(D) If (cond) gd =(gs1 == gs2) ? 1 : 0
 Elsenop

Execute time

FF. cycle

Floating Point Operations

QADDSP/DADDDP Addition Floating Point

Floating Point Operations

QADDSP/DADDDP

General Form **QADDSP** (.type) gd, gs1, gs2 /* (D) */
DADDDP (.type) gd, gs1, gs2 /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		gs2		gs1		gd		z	creg				

SWAP.2

(D) Format

Gd	sp	.M	100	00101
Gs1	sp			
Gs2	sp			

Gd	dp	.M	100	10101
Gs	dp			
Gs	dp			

Description

SIMD, 将 Gs1 与 Gs2 相加, 结果存放在 Gd 之中。

Operation

(D) If (cond) gd = gs1 + gs2
Elsenop

Execute time

GG. cycle

Floating Point Operations
QSUBSP/DSUBDP

2/4 Ways Subtraction Floating Point

Floating Point Operations
QSUBSP/DSUBDP

General Form

QSUBSP (.type) gd, gs1, gs2 /* (D) */
DSUBDP (.type) gd, gs1, gs2 /* (D) */
.type = .M,

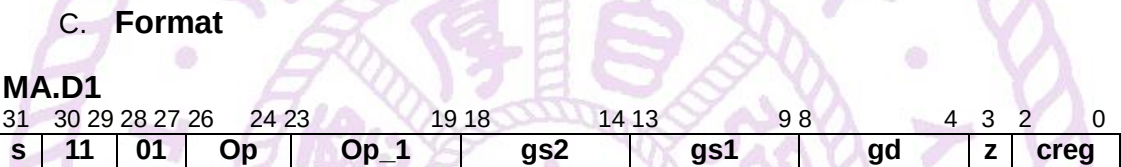
Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length; comment (D) identifies 32-bit instruction length.

Opcode



SWAP.2

(D) Format

Gd	sp	.M	100	00110
Gs1	sp			
Gs2	sp			

Gd	dp	.M	100	10110
Gs	dp			
Gs	dp			

Description

SIMD 指令，将 Gs1 与 Gs2 相减，结果存放在 Gd 之中。

Operation

(D) If (cond) Gd = Gs1 - Gs2
Elsenop

Execute time

HH. cycle

Floating Point Operations

QMPYSP/DMPYDP

2/4 Ways Multiplication Floating Point

Floating Point Operations

QMPYSP/DMPYDP

General Form **QMPYSP** (.type) gd, gs1, gs2 /* (D) */
DMPYDP (.type) gd, gs1, gs2 /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	gs2	gs1	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	100	00111
Gs1	sp			
Gs2	sp			

Gd	dp	.M	100	10111
Gs	dp			
Gs	dp			

Description

SIMD 指令，将 Gs1 与 Gs2 相乘，结果存放在 Gd 之中。

Operation

(D) If (cond) gd = gs1 * gs2
Elsenop

Execute time

11. cycle

Floating Point Operations
QDIVSP/DDIVDP

2/4 Ways Division Floating Point

Floating Point Operations
QDIVSP/DDIVDP

General Form

QDIVSP (.type) gd, gs1, gs2 /* (D) */
DDIVDP (.type) gd, gs1, gs2 /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		gs2		gs1		gd		z	creg				

SWAP.2

(D) Format

Gd	sp	.M	100	01000
Gs1	sp			
Gs2	sp			

Gd	dp	.M	100	11000
Gs1	dp			
Gs2	dp			

Description

SIMD 指令，将 Gs1 与 Gs2 相除，结果存放在 Gd 之中。

Operation

(D) If (cond) gd = gs1 / gs2
Elsenop

Execute time

JJ. cycle

Floating Point Operations

QMPYADDSP/DMPYADDDP

2/4 Ways
Fused
Mac
Floating
Point

Floating Point Operations

QMPYADDSP/DMPYADDDP

General Form **QMPYADDSP** (.type) gd, gs1, gs2 /* (D) */
 DMPYADDDP (.type) gd, gs1, gs2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		gs2		gs1		gd		z	creg				

SWAP.2

(D) Format

Gd	sp	.M	100	01001
Gs1	sp			
Gs2	sp			
Gd	dp	.M	100	11001
Gs1	dp			
Gs2	dp			

Description

SIMD 指令，将 Gs1 与 Gs2 相乘再和 Gd 相加，结果存放在 Gd 之中。

Operation

(D) If (cond) gd += gs1 * gs2
 Elsenop

Execute time

KK. cycle

Floating Point Operations

QNMPYADDSP/DNMPYADDDP

2/4

Ways
Negative
Fused
Mac
Floating
Point

Floating Point Operations

QNMPYADDSP/DNMPYADDDP

General Form **QNMPYADDSP** (.type) gd, gs1, gs2 /* (D) */
 DNMPYADDDP (.type) gd, gs1, gs2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	gs2	gs1	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	100	01010
Gs1	sp			
Gs2	sp			

Gd	dp	.M	100	11010
Gs1	dp			
Gs2	dp			

Description

SIMD 指令，将 Gs1 与 Gs2 相乘再和 Gd 相加之后再取相反数，结果存放在 Gd 之中。

Operation

(D) If (cond) $gd = -(gd + gs1 * gs2)$
 Elsenop

Execute time

LL. cycle

Floating Point Operations

QMPYSUBSP/DMPYSUBDP

2/4 Ways
Fused
Mac
Floating
Point

Floating Point Operations

QMPYSUBSP/DMPYSUBDP

General Form **QMPYSUBSP** (.type) gd, gs1, gs2 /* (D) */
 DMPYSUBDP (.type) gd, gs1, gs2 /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. **Format**

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		gs2		gs1		gd		z	creg				

SWAP.2

(D) **Format**

Gd	sp	.M	100	01011
Gs1	sp			
Gs2	sp			
Gd	dp	.M	100	11011
Gs1	dp			
Gs2	dp			

Description

SIMD 指令，将 Gs1 与 Gs2 相乘再和 Gd 相减，结果存放在 Gd 之中。

Operation

(D) If (cond) gd = -(gd - gs1 * gs2)
 Elsenop

Execute time

MM. cycle

Floating Point Operations

QNMPYSUBSP/DNMPYSUBDP

2/4
Ways
Negative
Fused
Mac
Floating
Point

Floating Point Operations

QNMPYSUBSP/DNMPYSUBDP

General Form **QNMPYSUBSP** (.type) gd, gs1, gs2 /* (D) */
DNMPYSUBDP (.type) gd, gs1, gs2 /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D1

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	gs2	gs1	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	100	01100
Gs1	sp			
Gs2	sp			

Gd	dp	.M	100	11100
Gs1	dp			
Gs2	dp			

Description

SIMD 指令，将 Gs1 与 Gs2 相乘再和 Gd 相减之后再取相反数，结果存放在 Gd 之中。

Operation

(D) If (cond) $gd = (gd - gs1 * gs2)$
Elsenop

Execute time

NN. cycle

Floating Point Operations

QABSSP/DABSDP

2/4 Ways Absolute
Floating Point

Floating Point Operations

QABSSP/DABSDP

General Form

QABSSP (.type) gd, gs /* (D) */
DABSDP (.type) gd, gs /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	gs	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	011	00101	00000
Gs	sp				
Gd	dp	.M	011	00111	00000
Gs	dp				

Description

SIMD 指令，求 Gs 的绝对值，将结果存放在 Gd 中。

Operation

(D) If (cond) gd =abs(gs)
Elsenop

Execute time

OO. cycle

Floating Point Operations
QSQRTSP/DSQRTDP

2/4 Ways Square Root
Floating Point

Floating Point Operations
QSQRTSP/DSQRTDP

General Form **QSQRTSP** (.type) gd, gs /* (D) */
DSQRTDP (.type) gd, gs /* (D) */
.type = .M,

Syntax
(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. **Format**

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		Op_2		gs		gd		z	creg				

SWAP.2

(D) Format

Gd	sp	.M	011	00101	00001
Gs	sp				
Gd	dp	.M	011	00111	00001
Gs	dp				

Description
SIMD 指令，求 Gs 的平方根，将结果存放在 Gd 中。

Operation
(D) If (cond) gd =sqrt(gs)
Elsenop

Execute time
PP. cycle

Floating Point Operations

QRCPSP/DRCPDP 2/4

Ways

Reciprocal

Approximation Floating Point

Floating Point Operations

QRCPSP/DRCPDP

General Form **QRCPSP** (.type) gd, gs /* (D) */
DRCPDP (.type) gd, gs /* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	gs	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	011	00101	00010
Gs	sp				

Gd	dp	.M	011	00111	00010
Gs	dp				

Description

SIMD 指令，求 Gs 的倒数，将结果存放在 Gd 中。

Operation

(D) If (cond) gd = 1/gs
Elsenop

Execute time

QQ. cycle

Floating Point Operations

QRSQRSP/DRSQRDP

2/4 Ways Reciprocal
Square Root Floating
Point

Floating Point Operations

QRSQRSP/DRSQRDP

General Form **QRSQRSP** (.type) gd, gs /* (D) */
DRSQRDP (.type) gd, gs /* (D) */
 .type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	gs	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	011	00101	00011
Gs	sp				

Gd	dp	.M	011	00111	00011
Gs	dp				

Description

SIMD 指令，求 Gs 的倒数平方根，将结果存放在 Gd 中。

Operation

(D) If (cond) gd =sqrt(1/gs)
 Elsenop

Execute time

RR. cycle

Floating Point Operations

QEXPSP/DEXPDP

2/4 Ways Exponent Floating Point

Floating Point Operations

QEXPSP/DEXPDP

General Form **QEXPSP** (.type) gd, gs /* (D) */
 DEXPDP (.type) gd, gs /* (D) */
 .type = .M,

Syntax
 (D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length
 In the syntax, comment (F) identifies 16-bit instruction length;
 comment (D) identifies 32-bit instruction length.

Opcode

C. **Format**

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1		Op_2		gs		gd		z	creg				

SWAP.2

(D) Format

Gd	sp	.M	011	00101	00100
Gs	sp				
Gd	sp	.M	011	00101	00101
Gs	sp				
Gd	sp	.M	011	00101	00110
Gs	sp				
Gd	dp	.M	011	00111	00100
Gs	dp				
Gd	dp	.M	011	00111	00101
Gs	dp				
Gd	dp	.M	011	00111	00110
Gs	dp				

Description
 Op_2 编码为 00100 时，底数为 2
 Op_2 编码为 00101 时，底数为 e
 Op_2 编码为 00110 时，底数为 10
 SIMD 指令，将 Gs 作为指数计算，将结果存放入 Gd 中。

Operation
 (D) If (cond) gd =2^{gs} or e^{gs} or 10^{gs}
 Elsenop

Execute time

SS. cycle



Floating Point Operations

QLOGSP/DLOGDP 2/4 Ways Log Floating Point

Floating Point Operations

QLOGSP/DLOGDP

General Form **QLOGSP** (.type) gd, gs /* (D) */
DLOGDP (.type) gd, gs/* (D) */
.type = .M,

Syntax

(D) rd, rs: X0~X23/Y0~Y23 : G0~G7, all Data Registers or Register Pairs

Instruction Length

In the syntax, comment (F) identifies 16-bit instruction length;
comment (D) identifies 32-bit instruction length.

Opcode

C. Format

MA.D2

31	30	29	28	27	26	24	23	19	18	14	13	9	8	4	3	2	0
s	11	01	Op	Op_1	Op_2	gs	gd	z	creg								

SWAP.2

(D) Format

Gd	sp	.M	011	00101	00111
Gs	sp				
Gd	sp	.M	011	00101	01000
Gs	sp				
Gd	sp	.M	011	00101	01001
Gs	sp				
Gd	dp	.M	011	00111	00111
Gs	dp				
Gd	dp	.M	011	00111	01000
Gs	dp				
Gd	dp	.M	011	00111	01001
Gs	dp				

Description

Op_2 编码为 00111 时，底数为 2

Op_2 编码为 01000 时，底数为 e

Op_2 编码为 01001 时，底数为 10

SIMD 指令，将 Gs 作为真数计算，将结果存放入 Gd 中。

Operation

(D) If (cond) gd =log₂gs or ln gs or lg gs
Elsenop

Execute time

TT. cycle



