国家高技术研究发展计划课题（八六三计划）

超大规模集成电路设计专项

技术报告

浙大数芯

MediaDSP3201/3202 指令集用户手册

专题名称：嵌入式CPU开发和SOC设计平台技术研究

课题名称：SOC中 32 位数字信号处理器芯片开发和设计平台技术研究

课题编号：2002AA1Z1140

依托单位：浙江大学

参加单位：杭州士兰微电子股份有限公司

参加单位：杭州国芯科技有限公司

二零零四年一月

MediaDSP3201/3202 ISA User's Manual

January 2004

主要内容

☑ 引言

☑ MediaDSP3201/MediaDSP3202 指令格式

一、引言

1．编写目的：

提供给 MediaDSP3200 系列设计人员和软件人员参考，作为设计文档。

2．读者对象：

MediaDSP3200 系列设计人员和软件程序员。

3．文档背景：

MediaDSP3200 系列 ISA 设计。

4．参考文献：

[1]超大规模集成电路 SOC 重大专向预启动项目技术报告(863-SOC-Y-3-2)，浙江大学信息与通信工程研究所，2001 年 8 月。

[2]超大规模集成电路设计专项技术报告（863-2002AA1Z1140），浙江大学信息与通信工程研究所，2003 年 4 月。

[3]John L. Hennessy and David A. Patterson. ***Computer Architecture: A Quantitative Approach,*** 3rd Edition, Morgan Kaufmann Publishing Co., Menlo Park, CA, **2002.**

[4]John Paul Shen, and Mikko H.Lipasti, ***Modern Processor Design: Fundamentals of Superscalar Processors,*** Beta Edition, McGraw-Hill Companies, Inc, 2003.

[5]TMS320C4x User's Guide, Texas Instruments, 1996.

[6]http://www.mips.com/

[7]http://www.analog.com/

[8]http://www.ti.com/

二、MediaDSP3201 指令

MediaDSP3201(简称 MD32)的指令类型可分为 MDF，MDD 和 MDS 三种。MDF 格式有 3 类 IF-type（立即数指令），RF-type（寄存器指令），JF-type（跳转指令）；MDD 格式有 ID-type（立即数指令）， RD-type（寄存器指令）， PD-type（并行指令）3 类；MDS 格式有移位指令，运算类指令，数据传输和关于存储器的运算类指令 4 类。

MDF 指令列表（60 条指令）

表 2.1 MD-32 MDF 指令一览表

MDF 指令	指令描述
Load/store 指令(12)	
LB	装入字节
LBU	装入不带符号字节
LH	装入半字
LHU	装入不带符号半字
LW	装入字
LWL	左边装入字
LWR	右边装入字
SB	存入字节
SH	存入半字
SW	存入字
SWL	左边存入字
SWR	右边存入字
运算指令（立即数）(8)	
ADDI	加立即数
ADDIU	加立即数(不带符号)
SLTI	小于立即数时置数
SLTIU	小于不带符号立即数时置数
ANDI	立即数“与”
ORI	立即数“或”
XORI	立即数“异或”
LUI	装入上部立即数
运算指令（3 操作数）(10)	
ADD	加法
ADDU	加法(不带符号)
SUB	减法
SUBU	减法(不带符号)
SLT	小于时置数
SLTU	小于(不带符号)时置数
AND	“与”
OR	“或”
XOR	“异或”
NOR	“或非”

乘指令(6)	
MULT	乘法
MULTU	无符号乘法
MFHI	从媒体寄存器高位传送到通用寄存器
MTHI	从通用寄存器传送到媒体寄存器高位
MFLO	从媒体寄存器低位传送到通用寄存器
MTLO	从通用寄存器传送到媒体寄存器低位
跳转和转移指令(12)	
J	跳转
JAL	跳转与连接
JR	跳转到寄存器
JALR	跳转到连接寄存器
BEQ	相等时转移
BNE	不相等时转移
BLEZ	大于或等于零时转移
BGTZ	大于零时转移
BLTZ	小于零时转移
BGEZ	大于或等于零时转移
BLTZAL	小于零或连接时转移
BGEZAL	大于或等于零且连接时转移
移位指令(6)	
SLL	逻辑左移
SRL	逻辑右移
SRA	算术运算右移
SLLV	逻辑变量左移
SRLV	逻辑变量右移
SRAV	算术变量右移
系统控制指令(6)	
MTC0	传送到 CP0
MFC0	从 CP0 传送
TLBWI	写变址 TLB 入口
TLBR	读变址 TLB 入口
SYSCALL	系统调用
RFE	异常返回

MDD 指令列表（58 条指令）

表 2.2 MD-32 MDD 指令一览表

MDD 指令	指令描述
Load/store 指令(12)	
LB	装入字节
LBU	装入不带符号字节
LH	装入半字
LHU	装入不带符号半字
LW	装入字
LWL	左边装入字
LWR	右边装入字
SB	存入字节
SH	存入半字
SW	存入字
SWL	左边存入字
SWR	右边存入字
运算指令（立即数）(8)	
ADDI	加立即数
ADDIU	加立即数(不带符号)
SLTI	小于立即数时置数
SLTIU	小于不带符号立即数时置数
ANDI	立即数“与”
ORI	立即数“或”
XORI	立即数“异或”
LUI	装入上部立即数
运算指令（3 操作数）(10)	
ADD	加法
ADDU	加法(不带符号)
SUB	减法
SUBU	减法(不带符号)
SLT	小于时置数
SLTU	小于(不带符号)时置数
AND	“与”
OR	“或”
XOR	“异或”
NOR	“或非”
乘指令(2)	
MULT	乘法
MULTU	无符号乘法
循环指令（2 条）	
RPTS	单条指令循环
RPTB	程序块循环
移位指令(6)	
SLL	逻辑左移

SRL	逻辑右移
SRA	算术运算右移
SLLV	逻辑变量左移
SRLV	逻辑变量右移
SRAV	算术变量右移
运算和存储类并行指令(10)	
ADD_SW	整数加和存整数
SUB_SW	整数减和存整数
AND_SW	整数与和存整数
SRA_SW	整数代数右移和存整数
SRL_SW	整数逻辑右移和存整数
SLL_SW	整数逻辑左移和存整数
OR_SW	整数或和存整数
XOR_SW	整数异或和存整数
ABS_SW	整数绝对值和存整数
MULT_SW	整数乘和存整数
存取类并行指令(3)	
LW_SW	取整数和存整数
SW_SW	存整数和存整数
LW_LW	取整数和取整数
乘加类并行指令(3)	
MULT_ADD	整数乘和整数加
MULT_SUB	整数乘和整数减
MAC	乘累加

MDS 指令列表（MediaDSP3201 32 条指令, MediaDSP3202 41 条指令）

表 2.3 MD-32 MDS 指令一览表

MediaDSP3201 MDS 指令	指令功能描述
数据传输指令(6)	
PMTHI, PMTLO	传输 4-字节（*Rs* 到 *MRd*）从通用寄存器到 MDS 寄存器高/低端
PMFHI, PMFLO	传输 4-字节（*MRd* 到 *Rs*）从 MDS 寄存器高/低端到通用寄存器
PLOADO (MediaDSP3201 不支持)	传输 8-字节（*mem* 到 *MRd*）从 memory 到 MDS 寄存器
PSTOREO(MediaDSP3201 不支持)	传输 8-字节（*MRd* 到 *mem*）从 MDS 寄存器到 memory
转换指令(2)	
PACKSSDB/QD	将*MRt*（*mem*）和*MRs*操作数中打包的2-字节/4-字节数据转换为1-字节/2-字节数据，使用有符号饱和处理溢出
PACKUSDB/QD	将 *MRt*（*mem*）和 *MRs* 操作数中打包的 2-字节数据转换为 1-字节数据，使用无符号饱和处理溢出
解包指令(2)	
PUNPCKHBD/DQ/QO	将 *MRt*（*mem*）和 *MRs* 操作数中打包的 1-字节/2-字节/4-字节相交织，取高 64-bit 存入 *MRd* 操作数
PUNPCKLBD/DQ/QO	将 *MRt*（*mem*）和 *MRs* 操作数中打包的 1-字节/2-字节/4-字节相交织，取低 64-bit 存入 *MRd* 操作数
算术指令(17)	
PADDB/D/Q	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节/4-字节数据执行 SIMD 加法，不作溢出处理
PADDSB/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 加法，使用有符号饱和处理溢出
PADDUSB/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 加法，使用无符号饱和处理溢出
PSUBB/D/Q	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节/4-字节数据执行 SIMD 减法，不作溢出处理
PSUBSB/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 减法，使用有符号饱和处理溢出
PSUBUSB/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 减法，使用无符号饱和处理溢出
PMULLSD	*MRs*和*MRt*（*mem*）操作数中打包的2-字节数据执行SIMD有符号乘法，每个乘法结果取低16-bit。PMACLSD将每次乘法结果不断累加。
PMULHSD	*MRs*和*MRt*（*mem*）操作数中打包的2-字节数据执行SIMD有符号乘法，每个乘法结果取高16-bit。PMACHSD将每次乘法结果不断累加。
PMULLUD	*MRs*和*MRt*（*mem*）操作数中打包的2-字节数据执行SIMD无符号乘法，每个乘法结果取低16-bit。PMACLUD将每次乘法结果不断累加。
PMULHUD	*MRs*和*MRt*（*mem*）操作数中打包的2-字节数据执行SIMD无符号乘法，每个乘法结果取高16-bit。PMACHUD将每次乘法结果不断累加。
PMADDQD	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节数据执行 SIMD 有符号乘法，相邻 2 个结果两两相加

MediaDSP3201 MDS 指令	指令功能描述
PAVGB/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 平均值计算，小数四舍五入
PMAXUB	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节无符号数执行 SIMD 比较，结果取较大的数
PMAXSD	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节有符号数执行 SIMD 比较，结果取较大的数
PMINUB	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节无符号数执行 SIMD 比较，结果取较小的数
PMINSD	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节有符号数执行 SIMD 比较，结果取较小的数
PSADBD	*MRs* 和 *MRt*（*mem*）中打包的 1-字节无符号数执行 SIMD 减法，减法结果取绝对值，最后 8 个绝对差值相加
比较指令(2)	
PCMPEQB/D/Q	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节/4-字节数据执行 SIMD 比较，若相等则结果为全 1，否则为全 0
PCMPGTB/D/Q	*MRs* 和 *MRt*(*mem*)中打包的 1-字节/2-字节/4-字节有符号数执行 SIMD 比较，若大于则结果为全 1，否则为全 0
逻辑指令(4)	
PAND	*MRs* 和 *MRt*（*mem*）按位逻辑与
POR	*MRs* 和 *MRt*（*mem*）按位逻辑或
PXOR	*MRs* 和 *MRt*（*mem*）按位逻辑异或
PNOR	*MRs* 和 *MRt*（*mem*）按位逻辑或非
移位指令(3)	
PSLLD/Q/O	*MRs* 中打包的 2 字节/4 字节数据进行 SIMD 逻辑左移，移位量来自 *MRt*（*imm*）的最低 5bit
PSRLD/Q/O	*MRs* 中打包的 2 字节/4 字节数据进行 SIMD 逻辑右移，移位量来自 *MRt*（*imm*）的最低 5bit
PSRAD/Q	*MRs* 中打包的 2 字节数据进行 SIMD 算术右移，移位量来自 *MRt*(*imm*）的最低 5bit

文档常用符号说明

文档中用到的符号简述如下：

符号	意义
Src1 Src2 Src3 Src4	源寄存器 1 源寄存器 2 源寄存器 3 源寄存器 4
Dst Dst1 Dst2 D1 D2	目的寄存器（3bit） 目的寄存器 1（3bit） 目的寄存器 2（3bit） 目的寄存器 1（1bit） 目的寄存器 2（1bit）
rs rt rd Sa	源寄存器（5bit） 目标寄存器（5bit） 目的寄存器（5bit） 移位立即数或移位寄存器（5bit）
GPR Mod(ARn) Modn(ARn) Mem() Byte() Sign() Zero()	取出处理器通用寄存器的值 指出 ARn 的寻址模式 按照指令中的 Mod 位指出的模式取出 memory 中的值 取出 memory 中的值 对字中的字节进行操作 符号扩展 无符号扩展，零扩展
G T E P A	普通寻址标志位 三宗量寻址标志位 扩展寻址标志位 并行指令寻址标志位 并行指令选择标志位
ARn IRn	辅助寄存器 n（0～7） 索引寄存器 n（0 或 1）
Op1 || op2	操作 1 和操作 2 并行执行
X and y X or y X xor y X * y ~ x |x|	X 与 y 按位逻辑与 X 与 y 按位逻辑或 X 与 y 按位逻辑异或 X 与 y 按位相乘 X 按位逻辑反 X 取绝对值
X << y X >> y	X 左移 ybit X 右移 ybit
*++SP *SP--	SP 递增，递增后的值作为地址 SP 作为地址，然后 SP 递减

说明：

- 对于指令表中某些新增指令还有待于进一步讨论，所以在本文档中并未给出说明。
- 文档中列出的指令说明是按照指令表中的顺序给出的，可以对照查阅。
- 文档中主要对 MDD 指令编码作了说明。

寄存器名称	寄存器序列	说明
R0	0	0 寄存器(硬连线为零)
R1	1	
R2	2	
R3	3	
R4	4	
R5	5	
R6	6	
R7	7	
R8	8	辅助寄存器 m/n
R9	9	辅助寄存器 m/n
R10	10	辅助寄存器 m/n
R11	11	辅助寄存器 m/n
R12	12	辅助寄存器 m/n
R13	13	辅助寄存器 m/n
R14	14	辅助寄存器 m/n
R15	15	辅助寄存器 m/n
R16	16	
R17	17	
R18	18	
R19	19	
R20	20	
R21	21	
R22	22	
R23	23	
R24	24	索引寄存器 0(IR0)
R25	25	索引寄存器 1(IR1)
R26	26	
R27	27	
R28	28	
R29	29	
R30	30	
R31	31	

(a) 带偏移量的间接寻址

Mod field	Syntax	Operation	Description
10000	*+ARn(disp)	addr=ARn + disp	前加
10001	*-ARn(disp)	addr=ARn - disp	前减
10010	*++ARn(disp)	addr=ARn + disp ARn=ARn +disp	前加且更新辅助寄存器
10011	*--ARn(disp)	addr=ARn - disp ARn=ARn -disp	前减且更新辅助寄存器
10100	*ARn++(disp)	addr=ARn ARn=ARn +disp	后加只更新辅助寄存器
10101	*ARn--(disp)	addr=ARn ARn=ARn -disp	后减只更新辅助寄存器
10110	*ARn++(disp)%	addr=ARn ARn=circ(ARn +disp)	后加且以窗口寻址更新辅助寄存器
10111	*ARn--(disp)%	addr=ARn ARn=circ(ARn +disp)	后减且以窗口寻址更新辅助寄存器

(b) 用索引寄存器 IR0 的间接寻址

Mod field	Syntax	Operation	Description
00000	*+ARn(IR0)	addr=ARn + IR0	前加
00001	*-ARn(IR0)	addr=ARn - IR0	前减
00010	*++ARn(IR0)	addr=ARn + IR0 ARn=ARn +IR0	前加且更新辅助寄存器
00011	*--ARn(IR0)	addr=ARn - IR0 ARn=ARn -IR0	前减且更新辅助寄存器
00100	*ARn++(IR0)	addr=ARn ARn=ARn +IR0	后加只更新辅助寄存器
00101	*ARn--(IR0)	addr=ARn ARn=ARn -IR0	后减只更新辅助寄存器
00110	*ARn++(IR0)%	addr=ARn ARn=circ(ARn +IR0)	后加且以窗口寻址更新辅助寄存器
00111	*ARn--(IR0)%	addr=ARn ARn=circ(ARn +IR0)	后减且以窗口寻址更新辅助寄存器

(c) 用索引寄存器 IR1 的间接寻址

Mod field	Syntax	Operation	Description
01000	*+ARn(IR1)	addr=ARn + IR1	前加
01001	*-ARn(IR1)	addr=ARn - IR1	前减
01010	*++ARn(IR1)	addr=ARn + IR1 ARn=ARn +IR1	前加且更新辅助寄存器
01011	*--ARn(IR1)	addr=ARn - IR1 ARn=ARn -IR1	前减且更新辅助寄存器
01100	*ARn++(IR1)	addr=ARn ARn=ARn +IR1	后加只更新辅助寄存器
01101	*ARn--(IR1)	addr=ARn ARn=ARn -IR1	后减只更新辅助寄存器
01110	*ARn++(IR1)%	addr=ARn ARn=circ(ARn +IR1)	后加且以窗口寻址更新辅助寄存器
01111	*ARn--(IR1)%	addr=ARn ARn=circ(ARn +IR1)	后减且以窗口寻址更新辅助寄存器

(d) indirect addressing (special cases)

Mod field	Syntax	Operation	Description
11000	*ARn	addr=ARn	间接寻址
11001	*ARn++(IR0)!	addr=ARn ARn=!(ARn + IR0)	后加且比特反转

备注：

Addr = 内存地址

ARn = 辅助寄存器 AR0 ～ AR7

IRn = 索引寄存器 IR0 or IR1

Disp = 偏移量

++ = 加且更新

-- = 减且更新

! = 比特反转寻址

circ()% = 窗口寻址

指令格式说明

- IF-type 指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op						rs					rt					Imm																

- ID-type 指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Load/store						11111					rt					Modm					ARm			Disp								

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID-type						11111					00		Dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID-type						11111					01		Dst			Modm					ARm			Disp								

- RF-type 指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op						rs					rt					rd					Sa					Func					

- RD-type 指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			rt					rd					Disp						SRA,SLL,SRL				

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						10		Direct1			rt					Direct2											SRA,SLL,SRL				

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						11		ARm			rt					Modm					Disp						SRA,SLL,SRL				

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	RD-type						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						E=00		ARm			rt					T=01		Dst			Modm				1	RD-type						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						E=01		ARm			imm					T=01		Dst			Disp				1	RD-type						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					E=00		ARm			T=10		Dst			Modm				1	RD-type						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					E=01		ARm			T=10		Dst			Disp				1	RD-type						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		Dst			Modn				1	RD-type						

- JF-type 指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op						Index																									

- PD-type（运算和存储类并行）指令格式：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op						B1	B2	Src1			Modm		Src2			Modm		Dst1			B3	Modn			ARm			ARn			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op						A		Src1			P		Src2			0		Modm			D	Modn			ARm			ARn			

Op	B1B2B3	指令
110011	000	ADD_SW
	001	SUB_SW
	010	AND_SW
	011	OR_SW
	100	XOR_SW
	101	MULT_SW
	110	保留
	111	保留
111011	000	ABS_SW
	001	SRA_SW
	010	SRL_SW
	011	SLL_SW
	100	LW_SW
	101	SW_SW
	110	LW_LW
	111	保留

Op	A	指令
010011	00	MULT_ADD
	01	MULT_SUB
	10	保留
	11	保留

a) MDS 指令汇编助记符格式

MDS 指令的汇编助记符格式，即句型为

INST DEST，SRC

其中，INST 为指令助记符，DEST 为目的操作数，SRC 为源操作数。

b) MDS 指令机器码编码格式

MDS 指令的编码占用一个特殊操作码，构成 MDS 指令子集，子集中的每条 MDS 指令用功能码区分。MDS 指令格式的设计基于 Virgo 的 R—Type，参考基本指令的 RD—Type，基本相一致。

i. 一个操作数来自 MDS 寄存器 MR1，另一个操作数来自立即数 sa

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		000			gg		MR1			00000					sa					func_code						

ii. 一个操作数来自 MDS 寄存器 MR1，另一个操作数来自 MDS 寄存器 MR2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					func_code						

iii. 一个操作数来自 MDS 寄存器 MR1，另一个操作数来自通用寄存器 rs

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			gg		MR1			rs					00000					func_code						

iv. 一个操作数来自 MDS 寄存器 MR1，另一个操作数来自存储器

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					func_code						

上面 4 种指令编码格式中，gg 数据场表示 MDS 指令 SIMD 操作的数据颗粒度，其编码如下表所示。

gg = instr[7 6]	数据颗粒度	指令助记符中的特征字符
00	Packed Bytes	B
01	Packed Doublebytes	D
10	Packed QuadBytes	Q
11	Octabyte	O

在第四种操作数寻址方式中，第 2 个操作数来自存储器，对存储器地址的计算和对辅助寄存器的修改如下表所示。

Modm (5bit)	地址计算/辅助地址寄存器自修改	Modm (5bit)	地址计算/辅助地址寄存器自修改
00000	*+ARm(disp)	01000	*+ARm(IR0)
		10000	*+ARm(IR1)
00001	*-ARm(disp)	01001	*-ARm(IR0)
		10001	*-ARm(IR1)
00010	*++ARm(disp)	01010	*++ARm(IR0)
		10010	*++ARm(IR1)
00011	*--ARm(disp)	01011	*--ARm(IR0)
		10011	*--ARm(IR1)
00100	*ARm++(disp)	01100	*ARm++(IR0)
		10100	*ARm++(IR1)
00101	*ARm--(disp)	01101	*ARm--(IR0)
		10101	*ARm--(IR1)
00110	*ARm++(disp)%	01110	*ARm++(IR0)%
		10110	*ARm++(IR1)%
00111	*ARm--(disp)%	01111	*ARm--(IR0)%
		10111	*ARm--(IR1)%
11001	*ARm++(IR0)B	11000	*ARm

c) MDS 指令集的设计和编码

功能码	0	1	2	3	4	5	6	7
0	PSLL D/Q/O	*PSHUFD**	PSRL D/Q/O	PSRA D/Q				
1			PUNPCKH BD/DQ/QO	PUNPCKL BD/DQ/QO				
2	PMFHI	PMTHI	PMFLO	PMTLO	PACKSS DB/QD	PACKUS DB/QD		
3	PMULLSD	PMULLUD	*PMACLSD**	*PMACLUD**	PMULHSD	PMULHUD	*PMACHSD**	*PMACHUD**
4	PADDS B/D	PADDUS B/D	PSUBS B/D	PSUBUS B/D	PAND	POR	PXOR	PNOR
5	PMADDQD	PSADBD			PADD B/D/Q		PSUB B/D/Q	
6	PCMPGT B/D/Q				PCMPEQ B/D/Q			*PLOADO**
7	PMAXSD	PMAXUB	PMINSD	PMINUB	PAVG B/D			*PSTOREO**

注：斜体有上标*的表明在 MediaDSP3202 中支持。

按照功能，所设计的 MDS 指令可分为 7 组：

- 数据传输指令
- 转换指令
- 解包指令
- 算术指令
- 比较指令
- 逻辑指令
- 移位指令

描述格式：

句型： INST dst，src1，src2 或者

INST dst1，dst2，src1，src2，src3

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op						rs					rt					rd					Sa					Func						

操作： src1 + src2 → dst 或者

src1 + src2 → dst1 || src3 → dst2

操作数说明：

src1：

src2：

src3：

dst1：

dst2：

描述：

举例： INST R4, @98AEh，R5

操作：

指令详细信息

ABS_SW

句型： **ABS_SW** dst，mod(ARn)，mod(ARm)，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111011						0	0	Src1			Modm		Src2			Modm		Dst			0	Modn				ARm			ARn		

操作： |modm(ARm)| → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

ABS_SW R2，*+AR7(IR1)，*AR0--(IR0)，R3

操作： |mem(*AR0--(IR0))| → GPR(R2)，

GPR(R3) → mem(*+AR7(IR1))

ADD

句型：

ADD	rd，rs，rt	或者
ADD	dst，*+ARm(disp1)，*+ARn(disp2)	或者
ADD	dst，mod(ARm)，rt	或者
ADD	dst，*+ARm(disp)，Imm	或者
ADD	dst，rs，mod(ARm)	或者
ADD	dst，rs，*+ARm(disp)	或者
ADD	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		dst			Disp2				1	100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		dst			Modm				1	100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		dst			Disp				1	100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		dst			Modm				1	100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		dst			Disp				1	100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		dst			Modn				1	100000						

操作：

GPR(Rs) + GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) + mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) + GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) + sign(Imm) → GPR(dst)	或者
GPR(Rs) + mod(ARm) → GPR(dst)	或者
GPR(Rs) + mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) + modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
dst： 寄存器 （通用寄存器 0～7）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
ADD R5，R3，R7	GPR(R3) + GPR(R7) → GPR(R5)
ADD R5， *+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) + mem(*+AR2(8h)) → GPR(R5)
ADD R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) + GPR(R3) → GPR(R5)
ADD R5， *+AR1(1h)，08h	Mem(*+AR1(1h)) + sign(08h) → GPR(R5)
ADD R5，R3，*AR2++(IR1)	GPR(R3) + Mem(*AR2++(IR1)) → GPR(R5)
ADD R5，R3，*+AR1(1h)	GPR(R3) + mem(*+AR1(1h)) → GPR(R5)
ADD R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) + Mem(*AR2++(IR1)) → GPR(R5)

ADDI

句型： **ADDI** rt，rs，Imm 或者
ADDI dst，@Imm 或者
ADDI dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001000						rs					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001000						11111					00		Dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001000						11111					01		Dst			Modm					ARm			Disp								

操作：

sign(Imm) + GPR(Rs) → GPR(rt) 或者
mem(Imm) + GPR(dst) → GPR(dst) 或者
modm(ARm) + GPR(dst) → GPR(dst)

操作数说明：

rs： 源寄存器 （通用寄存器 0～30）
rt： 目标寄存器 （通用寄存器 0～31）
ARm： 间接寻址 （辅助寄存器 0～7）
Dst： 目的寄存器 （通用寄存器 0～7）
G： 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

ADDI 与 ADDIU 的区别在于前者产生 Overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
ADDI R5，R3，0840h	sign(0840h) + GPR(R3) → GPR(R5)
ADDI R5，@0840h	Mem(0840h) + GPR(R5) → GPR(R5)
ADDI R5，*AR2++(40h)	Mem(*AR2++(40h)) + GPR(R5) → GPR(R5)

ADDIU

句型： **ADDIU** rt，rs，Imm 或者
ADDIU dst，@Imm 或者
ADDIU dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001001						rs					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001001						11111					00		Dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001001						11111					01		Dst			Modm					ARm			Disp								

操作：

sign(Imm) + GPR(Rs) → GPR(rt) 或者
mem(Imm) + GPR(dst) → GPR(dst) 或者
modm(ARm) + GPR(dst) → GPR(dst)

操作数说明：

rs： 源寄存器 （通用寄存器 0～30）
rt： 目标寄存器 （通用寄存器 0～31）
ARm： 间接寻址 （辅助寄存器 0～7）
Dst： 目的寄存器 （通用寄存器 0～7）
G： 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

ADDI 与 ADDIU 的区别在于前者产生 Overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
ADDIU R5，R3，0840h	sign(0840h) + GPR(R3) → GPR(R5)
ADDIU R5，@0840h	Mem(0840h) + GPR(R5) → GPR(R5)
ADDIU R5，*AR2++(40h)	Mem(*AR2++(40h)) + GPR(R5) → GPR(R5)

ADDU

句型：

ADDU	rd，rs，rt	或者
ADDU	dst，*+ARm(disp1)，*+ARn(disp2)	或者
ADDU	dst，mod(ARm)，rt	或者
ADDU	dst，*+ARm(disp)，Imm	或者
ADDU	dst，rs，mod(ARm)	或者
ADDU	dst，rs，*+ARm(disp)	或者
ADDU	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		dst			Disp2				1	100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		dst			Modm				1	100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		dst			Disp				1	100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		dst			Modm				1	100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		dst			Disp				1	100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		dst			Modn				1	100001						

操作：

GPR(Rs) + GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) + mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) + GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) + sign(Imm) → GPR(dst)	或者
GPR(Rs) + mod(ARm) → GPR(dst)	或者
GPR(Rs) + mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) + modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31
dst： 寄存器 （通用寄存器 0～7）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod (4bit)	偏移地址的计算	Mod (4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

ADD 和 ADDU 的区别在于前者产生 overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
ADDU R5，R3，R7	GPR(R3) + GPR(R7) → GPR(R5)
ADDU R5， *+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) + mem(*+AR2(8h)) → GPR(R5)
ADDU R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) + GPR(R3) → GPR(R5)
ADDU R5， *+AR1(1h)，08h	Mem(*+AR1(1h)) + sign(08h) → GPR(R5)
ADDU R5，R3，*AR2++(IR1)	GPR(R3) + Mem(*AR2++(IR1)) → GPR(R5)
ADDU R5，R3，*+AR1(1h)	GPR(R3) + mem(*+AR1(1h)) → GPR(R5)
ADDU R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) + Mem(*AR2++(IR1)) → GPR(R5)

ADD_SW

句型： **ADD_SW** dst，mod(ARn)，mod(ARm)，src1，src2，

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110011						0	0	Src1			Modm		Src2			Modm		Dst			0	Modn				ARm			ARn		

操作： GPR(src1) + modm(ARm) → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）
Src2： 寄存器 （通用寄存器 0～7）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

ADD_SW R2，*+AR7(IR1)，*AR0--(IR0)，R5，R3
操作： mem(*AR0--(IR0)) + GPR(R5) → GPR(R2)，
GPR(R3) → mem(*+AR7(IR1))

AND

句型：

AND	rd，rs，rt	或者
AND	dst，*+ARm(disp1)，*+ARn(disp2)	或者
AND	dst，mod(ARm)，rt	或者
AND	dst，*+ARm(disp)，Imm	或者
AND	dst，rs，mod(ARm)	或者
AND	dst，rs，*+ARm(disp)	或者
AND	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	100100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		Dst			Modm				1	100100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	100100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		Dst			Modm				1	100100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		Dst			Disp				1	100100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	100100						

操作：

GPR(Rs) and GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) and mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) and GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) and sign(Imm) → GPR(dst)	或者
GPR(Rs) and mod(ARm) → GPR(dst)	或者
GPR(Rs) and mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) and modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
AND R5，R3，R7	GPR(R3) and GPR(R7) → GPR(R5)
AND R5， *+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) and mem(*+AR2(8h)) → GPR(R5)
AND R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) and GPR(R3) → GPR(R5)
AND R5， *+AR1(1h)，08h	Mem(*+AR1(1h)) and sign(08h) → GPR(R5)
AND R5，R3，*AR2++(IR1)	GPR(R3) and Mem(*AR2++(IR1)) → GPR(R5)
AND R5，R3，*+AR1(1h)	GPR(R3) and mem(*+AR1(1h)) → GPR(R5)
AND R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) and Mem(*AR2++(IR1)) → GPR(R5)

ANDI

句型： **ANDI** rt，rs，Imm 或者
ANDI dst，@Imm 或者
ANDI dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001100						rs					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001100						11111					00		dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001100						11111					01		dst			Modm					ARm			Disp								

操作：

GPR(Rs) and zero(Imm) → GPR(rt) 或者
GPR(dst) and mem(Imm) → GPR(dst) 或者
GPR(dst) and modm(ARm) → GPR(dst)

操作数说明：

rs: 源寄存器 （通用寄存器 0～30）
rt: 目标寄存器 （通用寄存器 0～31）
ARm: 间接寻址 （辅助寄存器 0～7）
Dst: 目的寄存器 （通用寄存器 0～7）
G: 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
ANDI R5，R3，0840h	GPR(R3) and zero(0840h) → GPR(R5)
ANDI R5，@0840h	GPR(R5) and mem(0840h) → GPR(R5)
ANDI R5，*AR2++(40h)	GPR(R5) and mem(AR2) → GPR(R5)，AR2=AR2+40h

AND_SW

句型： **AND_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110011						0	1	Src1			Modm		Src2			Modm		Dst			0	Modn				ARm			ARn		

操作： modm(ARm) AND GPR(src1) → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

AND_SW R2，*+AR7(IR1)，*AR0--(IR0)，R5，R3

操作： mem(*AR0--(IR0)) and GPR(R5) → GPR(R2)，

GPR(R3) → mem(*+AR7(IR1))

BEQ

句型： BEQ rs, rt, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000100						rs					rt					offset																

操作： sign(offset) || offset || 0^2 → target ， (GPR[rs] = GPR[rt])→ condition
if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
offset： 立即数
PC： 指令地址

描述：

转移的目标地址是延迟槽指令地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。寄存器 *rs* 和 *rt* 的内容相比较，若两寄存器相等则转移。此指令有一个指令的时延。条件转移指令

执行周期：1 cycle

举例：

BEQ r5, r6，0x100
操作： 0x400 → target ， (GPR[r5] = GPR[r6])→ condition
if condition then PC + 0x400 → PC

BLEZ

句型： BLEZ rs, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000110						rs					00000					offset																

操作： sign(offset) || offset || 0^2 → target ， $(GPR[rs]_{31} = 1)$ or $(GPR[rs] = 0^{32})$→ condition
if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
offset： 立即数
PC： 指令地址

描述：

转移的目标地址是延迟槽指令的地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。寄存器 *rs* 的内容同零相比较，若通用寄存器 *rs* 的符号位设置或等于零，则程序转移到目标地址。此指令有一个指令的时延。条件转移指令

执行周期：1 cycle

举例：

BLEZ r5, 0x100
操作： 0x400 → target ， $(GPR[r5]_{31} = 1)$ or $(GPR[r5] = 0^{32})$→ condition
if condition then PC + 0x400 → PC

BLTZ

句型： BLTZ rs, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000001						rs					00000					offset																

操作： sign(offset) || offset || 0^2 → target ， $(GPR[rs]_{31} = 1)$ → condition

if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）

offset： 立即数

PC： 指令地址

描述：

转移的目标地址是延迟槽指令的地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。寄存器 *rs* 的内容同零相比较，若通用寄存器 *rs* 的符号位设置，则程序转移到目标地址。此指令有一个指令的时延。条件转移指令

执行周期：1 cycle

举例：

BLTZ r5, 0x100

操作： 0x400 → target ， $(GPR[r5]_{31} = 1)$ → condition

if condition then PC + 0x400 → PC

BLTZAL

句型： BLTZAL rs, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000001						rs					10000					offset																

操作： sign(offset) || offset || 0^2 → target，
(GPR[rs]$_{31}$ = 1) → condition，
PC + 8 → GPR[31]
if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
offset： 立即数
PC： 指令地址

描述：

转移的目标地址是延迟槽指令的地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。延迟槽后面的指令地址放入连接寄存器 r31 中，若寄存器 *rs* 的符号位设置，则程序转移到目标地址。此指令有一个指令的时延。通用寄存器 rs 不是 r31。条件转移指令

执行周期：1 cycle

举例：

BLTZAL r5, 0x100
操作： 0x400 → target ，(GPR[r5]$_{31}$ = 1) → condition
PC + 8 → GPR[31]
if condition then PC + 0x400 → PC

BGEZ

句型： BGEZ rs, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000001						rs					00001					offset																

操作： sign(offset) || offset || 0^2 → target ， $(GPR[rs]_{31} = 0)$ → condition

if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）

offset： 立即数

PC： 指令地址

描述：

转移的目标地址是延迟槽指令的地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。寄存器 *rs* 的内容同零相比较，若通用寄存器 *rs* 的符号位清除，则程序转移到目标地址。此指令有一个指令的时延。条件转移指令

执行周期：1 cycle

举例：

BGEZ r5, 0x100

操作： 0x400 → target ， $(GPR[r5]_{31} = 0)$ → condition

if condition then PC + 0x400 → PC

BGEZAL

句型： BGEZAL rs, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000001						Rs					10001					offset																

操作： sign(offset) || offset || 0^2 → target，
$(GPR[rs]_{31} = 0)$ → condition，
PC + 8 → GPR[31]
if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
offset： 立即数
PC： 指令地址

描述：

转移的目标地址是延迟槽指令的地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。延迟槽后面的指令地址放入连接寄存器 r31 中，若寄存器 *rs* 的符号位清除，则程序转移到目标地址。此指令有一个指令的时延。通用寄存器 rs 不是 r31。条件转移指令

执行周期：1 cycle

举例：

BGEZAL r5, 0x100
操作： 0x400 → target ，$(GPR[r5]_{31} = 0)$ → condition
PC + 8 → GPR[31]
if condition then PC + 0x400 → PC

BGTZ

句型： BGTZ rs, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000111						rs					00000					offset																

操作： sign(offset) || offset || 0^2 → target ，
(GPR[rs]$_{31}$ = 0) and (GPR[rs] ≠ 0^{32}) → condition
if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
offset： 立即数
PC： 指令地址

描述：

跳转的目标地址是延迟槽指令的地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。寄存器 *rs* 的内容同零相比较，若通用寄存器 *rs* 的符号位清除且不等于零，则程序转移到目标地址。此指令有一个指令的时延。条件转移指令

执行周期：1 cycle

举例：

BGTZ r5, 0x100
操作： 0x400 → target ，
(GPR[r5]$_{31}$ = 0) and (GPR[r5] ≠ 0^{32}) → condition
if condition then PC + 0x400 → PC

BNE

句型： BNE rs, rt, offset

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000101						rs					rt					offset																

操作： sign(offset) || offset || 0^2 → target ， (GPR[rs] ≠ GPR[rt])→ condition
if condition then PC + target → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
offset： 立即数
PC： 指令地址

描述：

转移的目标地址是延迟槽指令地址同 16 位 *offset* 左移 2 位后的符号扩展到的 32 位的数之和。寄存器 *rs* 和 *rt* 的内容相比较，若两寄存器不相等则转移。此指令有一个指令的时延。条件转移指令

执行周期：1 cycle

举例：

BNE r5, r6，0x100
操作： 0x400 → target ， (GPR[r5] ≠ GPR[r6])→ condition
if condition then PC + 0x400 → PC

J

句型： **J target**

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000010						target																									

操作： $PC_{31\sim28}$ || target || 0^2 →PC

操作数说明：

target： 立即数

PC： 指令地址

描述： 26 位目标地址左移 2 位同延迟槽的 PC 地址的高 4 位组合成新的地址，程序无条件跳转到计算的地址。此指令有一个指令的时延。跳转指令

执行周期：1 cycle

举例：

J 0x00400000

操作： $PC_{31\sim28}$ || $100\_0000_{27\sim0}$→ PC

JAL

句型： **JAL target**

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000011						target																									

操作： $PC_{31\sim28}$ || target || 0^2 →PC 并且 PC + 8 → GPR[31]

操作数说明：

target： 立即数

PC： 指令地址

描述：

26 位目标地址左移 2 位同延迟槽的 PC 地址的高 4 位组合成新的地址，程序无条件跳转到计算的地址。此指令有一个指令的时延。延迟槽后面的指令地址读入寄存器 r31。跳转并连接指令

执行周期：1 cycle

举例：

JAL 0x00400000

操作： $PC_{31\sim28}$ || $100\_0000_{27\sim0}$ → PC 并且 PC + 8 → GPR[31]

JALR

句型： **JALR rs**

JALR rd, rs

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00000					rd					00000						001001					

操作： GPR[rs] →PC 并且 PC＋8 → GPR[31]

GPR[rs] →PC 并且 PC＋8 → GPR[rd]

操作数说明：

rs： 寄存器 （通用寄存器 0～31）

rd： 寄存器 （通用寄存器 0～31）

PC： 指令地址

描述：

程序无条件跳转到寄存器 rs 包含的地址，此指令有一个指令的延迟。延迟槽后面的指令地址读入寄存器 rd。跳转并连接指令

执行周期：1 cycle

举例：

JALR r5

JALR r6, r5

操作： GPR[r5] →PC 并且 PC＋8 → GPR[31] 或者

GPR[r5] →PC 并且 PC＋8 → GPR[r6]

JR

句型： **JR rs**

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					000000000000000															001000					

操作： GPR[rs] → PC

操作数说明：

rs： 寄存器 （通用寄存器 0～31）

PC： 指令地址

描述：

程序无条件跳转寄存器 *rs* 包含的地址。跳转指令

执行周期：1 cycle

举例：

JR r5

操作： GPR[r5] →PC

LB

句型： **LB** rt, offset(base) 或者

LB rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100000						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100000						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，

Sign(Byte(mem(Vaddr))) → GPR(rt)

操作数说明：

base： 寄存器 （通用寄存器 0～30）

rt： 寄存器 （通用寄存器 0～31）

ARm： 间接寻址（辅助寄存器 0～7）

Disp： 立即数

offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
LB R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，Sign(Byte(mem(Vaddr)))→ GPR(R5)
LB R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，Sign(Byte(mem(Vaddr))) → GPR(R5)

LBU

句型： **LBU** rt, offset(base) 或者

LBU rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100100						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100100						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，

zero(Byte(mem(Vaddr))) → GPR(rt)

操作数说明：

base： 寄存器 （通用寄存器 0～30）

rt： 寄存器 （通用寄存器 0～31）

ARm： 间接寻址（辅助寄存器 0～7）

Disp： 立即数

offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
LBU R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，0(Byte(mem(Vaddr))) → GPR(R5)
LBU R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，0(Byte(mem(Vaddr))) → GPR(R5)

LH

句型： **LH** rt, offset(base) 或者

LH rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100001						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100001						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，

Sign(Byte(mem(Vaddr))) → GPR(rt)

操作数说明：

base： 寄存器 （通用寄存器 0～30）

rt： 寄存器 （通用寄存器 0～31）

ARm： 间接寻址（辅助寄存器 0～7）

Disp： 立即数

offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
LH R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，Sign(Byte(mem(Vaddr)))→ RP(R5)
LH R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，Sign(Byte(mem(Vaddr))) → RP(R5)

LHU

句型： **LHU** rt, offset(base) 或者

LHU rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100101						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100101						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，

zero(Byte(mem(Vaddr))) → GPR(rt)

操作数说明：

base： 寄存器 （通用寄存器 0～30）

rt： 寄存器 （通用寄存器 0～31）

ARm： 间接寻址（辅助寄存器 0～7）

Disp： 立即数

offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
LHU R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，0(Byte(mem(Vaddr))) → GPR(R5)
LHU R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，0(Byte(mem(Vaddr))) → GPR(R5)

LUI

句型： **LUI** rt，Imm 或者
LUI dst，@Imm 或者
LUI dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001111						00000					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001111						11111					00		dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001111						11111					01		dst			Modm					ARm			Disp								

操作：

Imm | zero(0^{16}) → GPR(rt) 或者
mem(Imm) [15:0] | zero(0^{16})→ GPR(dst) 或者
modm(ARm) [15:0] | zero(0^{16})→ GPR(dst)

操作数说明：

rs： 源寄存器 （通用寄存器 0～30）
rt： 目标寄存器 （通用寄存器 0～31）
ARm： 间接寻址 （辅助寄存器 0～7）
Dst： 目的寄存器 （通用寄存器 0～7）
G： 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle
举例：

句型	操作
LUI R5，0840h	0840h	zero(0^{16}) → GPR(R5)
LUI R5，@0840h	mem(0840h) [15:0]	zero(0^{16})→ GPR(R5)
LUI R5，*AR2++(40h)	Mem(AR2) [15:0]	zero(0^{16})→ GPR(R5)

LW

句型： **LW** rt, offset(base) 或者

LW rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100011						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100011						11111					rt					Modn					ARm			Disp								

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，

mem(Vaddr) → GPR(rt)

操作数说明：

base： 寄存器 （通用寄存器 0～30）

rt： 寄存器 （通用寄存器 0～31）

ARm： 间接寻址（辅助寄存器 0～7）

Disp： 立即数

offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
LW R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，mem(Vaddr) → GPR(R5)
LW R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，mem(Vaddr) → GPR(R5)

LWL

句型： **LWL** rt, offset(base) 或者
LWL rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100010						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100010						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，
Left (mem(Vaddr)) → GPR(rt)

操作数说明：

base： 寄存器 （通用寄存器 0～30）
rt： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Disp： 立即数
offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
LWL R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，Left(mem(Vaddr)) → GPR(R5)
LWL R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，Left(mem(Vaddr)) → GPR(R5)

LW_LW

句型： **LW_LW** dst1，dst2，mod(ARm)，mod(ARn)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111011						1	1	000			Modm		Dst2			Modm		Dst1			0	Modn				ARm			ARn		

操作： modm(ARm) → GRP(dst1) || modn(ARn) → GPR(dst2)

操作数说明：

ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
dst1： 寄存器 （通用寄存器 0～7）
dst2： 寄存器 （通用寄存器 24～31）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

LW_LW R5，R3，*AR0--(IR0)，*+AR7(IR1)
操作： mem(*AR0--(IR0)) → GRP(R5)，
mem(*+AR7(IR1)) → GRP(R3)

LWR

句型： **LWR** rt, offset(base) 或者
LWR rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100110						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
100110						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，
right (mem(Vaddr)) → GPR(rt)

操作数说明：

base： 寄存器 （通用寄存器 0～30）
rt： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Disp： 立即数
offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
LWR R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，Right(mem(Vaddr)) → GPR(R5)
LWR R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，Right(mem(Vaddr)) → GPR(R5)

LW_SW

句型： **LW_SW** dst，mod(ARn)，mod(ARm)，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111011						1	0	000			Modm		Src2			Modm		Dst			0	Modn				ARm			ARn		

操作： modm(ARm) → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

LW_SW R0，*+AR7(IR1)，*AR0--(IR0)，R3

操作： mem(*AR0--(IR0)) → GPR(R0)，

GPR(R3) → mem(*+AR7(IR1))

MAC

句型： **MAC** srcA，srcB

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
010011						10		Src1			P		Src2			0	Modm				0	Modn					ARm			ARn		

操作： srcA * srcB + MR0 → MR0{HI，LO}

操作数说明：

srcA、srcB：必须两个为寄存器（通用寄存器 R0~R7，表示为 src1 和 src2），或为两个为间接寻址（辅助寄存器 0～7，表示为 Arm 和 ARn）

src1： 寄存器 （通用寄存器 0～7）
src2： 寄存器 （通用寄存器 0～7）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

P(2bit)	描述
00	Modm(ARm) * modn(ARn) ， src1 – src2
01	Modm(ARm)* src1，modn(ARn)– src2
10	Src1 * src2，Modm(ARm)–modn(ARn)
11	Modm(ARm)* src1，src2 –modn(ARn)

执行周期：4 cycles

举例：

MAC *AR0--(IR0)， *+AR7(IR1)

操作： mem(*AR0--(IR0)) * mem(*AR7(IR1) + MR0 → MR0{HI,LO}，

MFC0

句型：MFC0 rt, rd

指令编码：

31 26	25 21	20 16	15 11	10 0
010000	**00000**	**rt**	**rd**	**000_0000_0000**

操作：CPR(rd) → GPR(rt)

操作数说明：

rt： 寄存器 （通用寄存器 0～31）

rd： 寄存器 （系统寄存器 0～15）

描述：CP0 的寄存器 rd 的内容装入通用寄存器 rt 中。

执行周期：1 cycle

MFHI

句型：MFHI rd

指令编码：

31 26	25 16	15 11	10 6	5 0
000000	**00_0000_0000**	**rd**	**00000**	**010000**

操作：MR0{HI }→GPR(rd)

操作数说明：

rd： 寄存器 （通用寄存器 0～31）
MR0： MDS 寄存器 0

描述： MDS 寄存器 MR0 中 HI 的内容装入通用寄存器 *rd* 中。
执行周期：1 cycle

MFLO

句型：MFLO rd

指令编码：

31 26	25 16	15 11	10 6	5 0
000000	**00_0000_0000**	**rd**	**00000**	**010010**

操作：MR0{LO }→ GPR(rd)

操作数说明：rd： 寄存器 （通用寄存器 0～31）

MR0： MDS 寄存器 0

描述：MDS 寄存器 MR0 中 LO 的内容装入通用寄存器 *rd* 中。

执行周期：1 cycle

MULT

句型：

MULT	rs，rt	或者
MULT	*+ARm(disp1)，*+ARn(disp2)	或者
MULT	mod(ARm)，rt	或者
MULT	*+ARm(disp)，Imm	或者
MULT	rs，mod(ARm)	或者
MULT	rs，*+ARm(disp)	或者
MULT	mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					00000					00000					011000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		000			Disp2				1	011000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		000			Modm				1	011000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		000			Disp				1	011000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		000			Modm				1	011000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		000			Disp				1	011000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		000			Modn				1	011000					

操作：

GPR(Rs) * GPR(rt) → MR0{HI,LO}	或者
Mem(*+ARm(disp1)) * mem(*+ARn(disp2)) → MR0{HI,LO}	或者
mod(ARm) * GPR(Rt) → MR0{HI,LO}	或者
Mem(*+ARm(disp)) * sign(Imm) → MR0{HI,LO}	或者
GPR(Rs) * mod(ARm) → MR0{HI,LO}	或者
GPR(Rs) * mem(*+ARm(disp)) → MR0{HI,LO}	或者
Modm(ARm) * modn(ARn) → MR0{HI,LO}	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：3 cycles

举例：

句型	操作
MULT R5，R3，R7	GPR(R3) * GPR(R7) → GPR(R5)
MULT R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) * mem(*+AR2(8h)) → GPR(R5)
MULT R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) * GPR(R3) → GPR(R5)
MULT R5，*+AR1(1h)，08h	Mem(*+AR1(1h)) * sign(08h) → GPR(R5)
MULT R5，R3，*AR2++(IR1)	GPR(R3) * Mem(*AR2++(IR1)) → GPR(R5)
MULT R5，R3，*+AR1(1h)	GPR(R3) * mem(*+AR1(1h)) → GPR(R5)
MULT R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) * Mem(*AR2++(IR1)) → GPR(R5)

MULTU

句型：

MULTU	rs，rt	或者
MULTU	*+ARm(disp1)，*+ARn(disp2)	或者
MULTU	mod(ARm)，rt	或者
MULTU	*+ARm(disp)，Imm	或者
MULTU	rs，mod(ARm)	或者
MULTU	rs，*+ARm(disp)	或者
MULTU	mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					00000					00000					011001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		000			Disp2				1	011001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		000			Modm				1	011000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		000			Disp				1	011001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		000			Modm				1	011001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		000			Disp				1	011001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		000			Modn				1	011001					

操作：

GPR(Rs) * GPR(rt) → MR0{HI,LO}	或者
Mem(*+ARm(disp1)) * mem(*+ARn(disp2)) → MR0 {HI,LO}	或者
mod(ARm) * GPR(Rt) → MR0{HI,LO}	或者
Mem(*+ARm(disp)) * sign(Imm) → MR0{HI,LO}	或者
GPR(Rs) * mod(ARm) → MR0{HI,LO}	或者
GPR(Rs) * mem(*+ARm(disp)) → MR0{HI,LO}	或者
Modm(ARm) * modn(ARn) → MR0{HI,LO}	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

MULT 和 MULTU 的区别在于前者产生 Overflow 的异常，而后者不产生任何异常。

执行周期：3 cycles

举例：

句型	操作
MULTU R5，R3，R7	GPR(R3) * GPR(R7) → GPR(R5)
MULTU R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) * mem(*+AR2(8h)) → GPR(R5)
MULTU R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) * GPR(R3) → GPR(R5)
MULTU R5，*+AR1(1h)，08h	Mem(*+AR1(1h)) * sign(08h) → GPR(R5)
MULTU R5，R3，*AR2++(IR1)	GPR(R3) * Mem(*AR2++(IR1)) → GPR(R5)
MULTU R5，R3，*+AR1(1h)	GPR(R3) * mem(*+AR1(1h)) → GPR(R5)
MULTU R5 ， *AR1++(IR0) ， *AR2++(IR1)	Mem(*AR1++(IR0)) * Mem(*AR2++(IR1)) → GPR(R5)

MULT_ADD

句型： **MULT_ADD** D，srcA，srcB，srcC，srcD

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
010011						00		Src1			P		Src2			0	Modm				D	Modn				ARm			ARn			

操作： srcA * srcB → {HI，LO} || srcC + srcD → GPR(D)

操作数说明：

srcA、srcB、srcC 和 srcD：必须两个为寄存器（通用寄存器 R0~R7，表示为 src1 和 src2），另两个为间接寻址（辅助寄存器 0～7，表示为 Arm 和 ARn）

src1： 寄存器 （通用寄存器 0～7）
src2： 寄存器 （通用寄存器 0～7）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
D： 寄存器 （0 为 R1，1 为 R2）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

P(2bit)	描述
00	Modm(ARm) * modn(ARn) ， src1 + src2
01	Modm(ARm) * src1，modn(ARn) + src2
10	Src1 * src2，Modm(ARm) + modn(ARn)
11	Modm(ARm) * src1，src2 + modn(ARn)

执行周期：3 cycles

举例：

MULT_ADD R3，*AR0--(IR0)，*+AR7(IR1)，R5，R7
操作： mem(*AR0--(IR0)) * mem(*+AR7(IR1)) → {HI,LO}，GPR(R5) + GPR(R7) → GPR(R3)

MULT_SUB

句型： **MULT_SUB** D，srcA，srcB，srcC，srcD

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
010011						01		Src1			P		Src2			0	Modm				D	Modn				ARm			ARn			

操作： srcA * srcB → {HI，LO} || srcC – srcD → GPR(D)

操作数说明：

srcA、srcB、srcC 和 srcD：必须两个为寄存器（通用寄存器 R0~R7，表示为 src1 和 src2），另两个为间接寻址（辅助寄存器 0～7，表示为 Arm 和 ARn）

src1： 寄存器 （通用寄存器 0～7）
src2： 寄存器 （通用寄存器 0～7）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
D： 寄存器 （0 为 R1，1 为 R2）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

P(2bit)	描述
00	Modm(ARm) * modn(ARn) ， src1 – src2
01	Modm(ARm)* src1，modn(ARn)– src2
10	Src1 * src2，Modm(ARm)–modn(ARn)
11	Modm(ARm)* src1，src2 –modn(ARn)

执行周期：3 cycles

举例：

MULT_SUB R0，R3，*AR0--(IR0)，R5，*+AR7(IR1)，R7
操作： mem(*AR0--(IR0)) * GPR(R5) → {HI,LO}，
mem(*+AR7(IR1)) - GPR(R7) → GPR(R3)

MULT_SW

句型： **MULT_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110011						1	0	Dst			Modm		Src1			Modm		Src2			1	Modn				ARm			ARn		

操作： modm(ARm) * GPR(src1) → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：3 cycles

举例：

MULT_SW *+AR7(IR1)，*AR0--(IR0)，R5，R3

操作： mem(*AR0--(IR0)) * GPR(R5) → MR0(Hi,LO)，

GPR(R3) → mem(*+AR7(IR1))

MTC0

句型：MTC0 rt, rd

指令编码：

31 26	25 21	20 16	15 11	10 0
010000	**00100**	**rt**	**rd**	**000_0000_0000**

操作： GPR(rt) → rd

操作数说明：

rt： 寄存器 （通用寄存器 0～31）

rd： 寄存器 （系统寄存器 0～15）

描述：通用寄存器 rt 的内容装入 CP0 的寄存器 rd 中。

执行周期：1 cycle

MTHI

句型：MTHI rs

指令编码：

31 26	25 21	20 6	5 0
000000	**rs**	**000_0000_0000_0000**	**010001**

操作： GPR(rs) → MR0{HI }

操作数说明：rs： 寄存器 （通用寄存器 0～31）
MR0： MDS 寄存器 0

描述：通用寄存器 *rd* 的内容装入 MDS 寄存器 MR0 中 HI 寄存器。
执行周期：1 cycle

MTLO

句型：MTLO rd

指令编码：

31 26	25 21	20 6	5 0
000000	**rs**	**000_0000_0000_0000**	**010011**

操作：GPR(rs) → MR0{HI }

操作数说明：rd： 寄存器 （通用寄存器 0～31）
MR0： MDS 寄存器 0

描述：通用寄存器 *rd* 的内容装入 MDS 寄存器 MR0 中 LO 寄存器。
执行周期：1 cycle

NOR

句型：

NOR	rd，rs，rt	或者
NOR	dst，*+ARm(disp1)，*+ARn(disp2)	或者
NOR	dst，mod(ARm)，rt	或者
NOR	dst，*+ARm(disp)，Imm	或者
NOR	dst，rs，mod(ARm)	或者
NOR	dst，rs，*+ARm(disp)	或者
NOR	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	100111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		Dst			Modm				1	100111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	100111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		Dst			Modm				1	100111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		Dst			Disp				1	100111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	100111						

操作：

GPR(Rs) nor GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) nor mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) nor GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) nor sign(Imm) → GPR(dst)	或者
GPR(Rs) nor mod(ARm) → GPR(dst)	或者
GPR(Rs) nor mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) nor modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
NOR R5，R3，R7	GPR(R3) nor GPR(R7) → GPR(R5)
NOR R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) nor mem(*+AR2(8h)) → GPR(R5)
NOR R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) nor GPR(R3) → GPR(R5)
NOR R5，*+AR1(1h)，08h	Mem(*+AR1(1h)) nor sign(08h) → GPR(R5)
NOR R5，R3，*AR2++(IR1)	GPR(R3) nor Mem(*AR2++(IR1)) → GPR(R5)
NOR R5，R3，*+AR1(1h)	GPR(R3) nor mem(*+AR1(1h)) → GPR(R5)
NOR R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) nor Mem(*AR2++(IR1)) → GPR(R5)

OR

句型：

OR	rd，rs，rt	或者
OR	dst，*+ARm(disp1)，*+ARn(disp2)	或者
OR	dst，mod(ARm)，rt	或者
OR	dst，*+ARm(disp)，Imm	或者
OR	dst，rs，mod(ARm)	或者
OR	dst，rs，*+ARm(disp)	或者
OR	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	100101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		Dst			Modm				1	100101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	100101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		Dst			Modm				1	100101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		Dst			Disp				1	100101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	100101						

操作：

GPR(Rs) or GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) or mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) or GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) or sign(Imm) → GPR(dst)	或者
GPR(Rs) or mod(ARm) → GPR(dst	或者
GPR(Rs) or mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) or modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
OR R5，R3，R7	GPR(R3) or GPR(R7) → GPR(R5)
OR R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) or mem(*+AR2(8h)) → GPR(R5)
OR R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) or GPR(R3) → GPR(R5)
OR R5，*+AR1(1h)，08h	Mem(*+AR1(1h)) or sign(08h) → GPR(R5)
OR R5，R3，*AR2++(IR1)	GPR(R3) or Mem(*AR2++(IR1)) → GPR(R5)
OR R5，R3，*+AR1(1h)	GPR(R3) or mem(*+AR1(1h)) → GPR(R5)
OR R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) or Mem(*AR2++(IR1)) → GPR(R5)

ORI

句型： **ORI** rt，rs，Imm 或者
ORI dst，@Imm 或者
ORI dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001101						rs					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001101						11111					00		dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001101						11111					01		dst			Modm					ARm			Disp								

操作：

GPR(Rs) or zero(Imm) → GPR(rt) 或者
GPR(dst) or mem(Imm) → GPR(dst) 或者
GPR(dst) or modm(ARm) → GPR(dst)

操作数说明：

rs： 源寄存器 （通用寄存器 0～30）
rt： 目标寄存器 （通用寄存器 0～31）
ARm： 间接寻址 （辅助寄存器 0～7）
Dst： 目的寄存器 （通用寄存器 0～7）
G： 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
ORI R5，R3，0840h	GPR(R3) or zero(0840h) → GPR(R5)
ORI R5，@0840h	GPR(R5) or mem(0840h) → GPR(R5)
ORI R5，*AR2++(40h)	GPR(R5) or mem(AR2) → GPR(R5)，AR2=AR2+40h

OR_SW

句型： **OR_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110011						0	1	Src1			Modm		Src2			Modm		Dst			1	Modn				ARm			ARn		

操作： modm(ARm) OR GPR(src1) → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

OR_SW R2，*+AR7(IR1)，*AR0--(IR0)，R5，R3

操作： mem(*AR0--(IR0)) or GPR(R5) → GPR(R2)，
GPR(R3) → mem(*+AR7(IR1))

PACKSSDB/QD

句型： **PACKSSDB/QD** MR1，MR2

PACKSSDB/QD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					010100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					010100						

操作：

PACKSSDB:
DEST[7..0] ←SaturateSignedDouble-byteToSignedByte DEST[15..0];
DEST[15..8] ←SaturateSignedDouble-byteToSignedByte DEST[31..16];
DEST[23..16] ←SaturateSignedDouble-byteToSignedByte DEST[47..32];
DEST[31..24] ←SaturateSignedDouble-byteToSignedByte DEST[63..48];
DEST[39..32] ←SaturateSignedDouble-byteToSignedByte SRC[15..0];
DEST[47..40] ←SaturateSignedDouble-byteToSignedByte SRC[31..16];
DEST[55..48] ←SaturateSignedDouble-byteToSignedByte SRC[47..32];
DEST[63..56] ←SaturateSignedDouble-byteToSignedByte SRC[63..48];

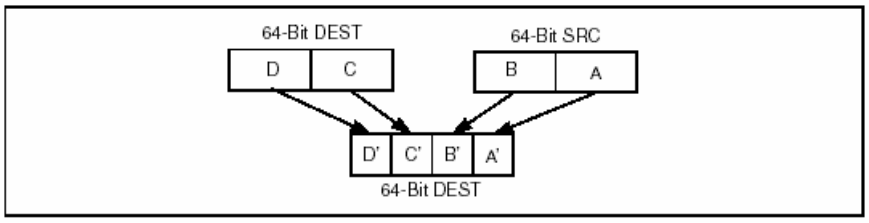
PACKSSQD:
DEST[15..0] ←SaturateSignedQuad-byteToSignedDouble-byte DEST[31..0];
DEST[31..16] ←SaturateSignedQuad-byteToSignedDouble-byte DEST[63..32];
DEST[47..32] ←SaturateSignedQuad-byteToSignedDouble-byte SRC[31..0];
DEST[63..48] ←SaturateSignedQuad-byteToSignedDouble-byte SRC[63..32];

操作数说明： MR1： MDS 寄存器
MR2： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PACKSSDB 将 64-bit DEST 操作数中打包的 4 个有符号 2 字节数和 64-bit SRC 操作数中打包的 4 个有符号 2 字节数转换为 8 个有符号字节数，采用有符号饱和法处理溢出，结果存入 DEST 操作数。

PACKSSQD 将 64-bit DEST 操作数中打包的 2 个有符号 4 字节数和 64-bit SRC 操作数中打包的 2 个有符号 4 字节数转换为 4 个有符号 2 字节数，采用有符号饱和法处理溢出，结果存入 DEST 操作数。

下图示例了 PACKSSQD 的操作过程，其它类推。



执行周期： **1** cycle

PACKUSDB/QD

句型： **PACKUSDB/QD** MR1，MR2

PACKUSDB/QD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					010101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					010101						

操作：

PACKUSDB:
DEST[7..0] ←SaturateSignedDouble-byteToUnsignedByte DEST[15..0];
DEST[15..8] ←SaturateSignedDouble-byteToUnsignedByte DEST[31..16];
DEST[23..16] ←SaturateSignedDouble-byteToUnsignedByte DEST[47..32];
DEST[31..24] ←SaturateSignedDouble-byteToUnsignedByte DEST[63..48];
DEST[39..32] ←SaturateSignedDouble-byteToUnsignedByte SRC[15..0];
DEST[47..40] ←SaturateSignedDouble-byteToUnsignedByte SRC[31..16];
DEST[55..48] ←SaturateSignedDouble-byteToUnsignedByte SRC[47..32];
DEST[63..56] ←SaturateSignedDouble-byteToUnsignedByte SRC[63..48];

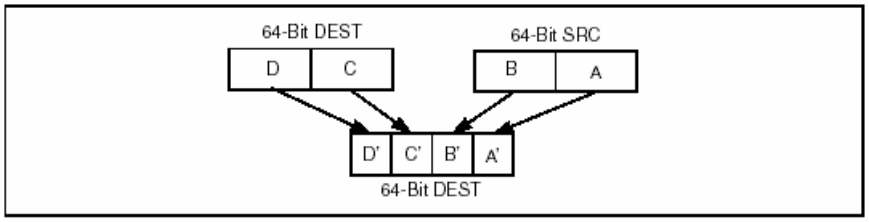
PACKUSQD:
DEST[15..0] ←SaturateSignedQuad-byteToUnsignedDouble-byte DEST[31..0];
DEST[31..16] ←SaturateSignedQuad-byteToUnsignedDouble-byte DEST[63..32];
DEST[47..32] ←SaturateSignedQuad-byteToUnsignedDouble-byte SRC[31..0];
DEST[63..48] ←SaturateSignedQuad-byteToUnsignedDouble-byte SRC[63..32];

操作数说明： MR1： MDS 寄存器
MR2： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PACKUSDB 将 64-bit DEST 操作数中打包的 4 个有符号 2 字节数和 64-bit SRC 操作数中打包的 4 个有符号 2 字节数转换为 8 个有符号字节数，采用无符号饱和法处理溢出，结果存入 DEST 操作数。

PACKUSQD 将 64-bit DEST 操作数中打包的 2 个有符号 4 字节数和 64-bit SRC 操作数中打包的 2 个有符号 4 字节数转换为 4 个有符号 2 字节数，采用无符号饱和法处理溢出，结果存入 DEST 操作数。

下图示例了 PACKUSQD 的操作过程，其它类推。



执行周期： **1** cycle

PADDB/D/Q

句型： **PADDB/D/Q** MR1，MR2

PADDB/D/Q *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					101100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					101100						

操作：

PADDB instruction with 64-bit operands:

DEST[7..0] ← DEST[7..0] + SRC[7..0];

* repeat add operation for 2nd through 7th byte *;

DEST[63..56] ← DEST[63..56] + SRC[63..56];

PADDD instruction with 64-bit operands:

DEST[15..0] ← DEST[15..0] + SRC[15..0];

* repeat add operation for 2nd and 3th double-byte *;

DEST[63..48] ← DEST[63..48] + SRC[63..48];

PADDQ instruction with 64-bit operands:

DEST[31..0] ← DEST[31..0] + SRC[31..0];

DEST[63..32] ← DEST[63..32] + SRC[63..32];

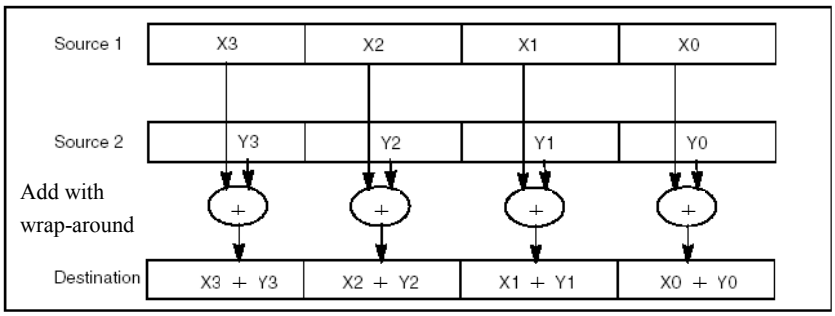
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PADDB/D/Q 对 64－bit DEST 操作数中打包的字节数/2 字节数/4 字节数和 64－bit SRC 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 加法，结果存入 DEST 操作数中，溢出被忽略。下图示例 PADDD 的操作过程，其它类推。



执行周期：1 cycle

PADDSB/D

句型： **PADDSB/D** MR1，MR2

PADDSB/D *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					100000						

操作：

PADDSB instruction with 64-bit operands:

DEST[7..0] ← SaturateToSignedByte(DEST[7..0] + SRC (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

DEST[63..56] ← SaturateToSignedByte(DEST[63..56] + SRC[63..56]);

PADDSD instruction with 64-bit operands:

DEST[15..0] ← SaturateToSignedDouble-byte(DEST[15..0] + SRC[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

DEST[63..48] ← SaturateToSignedDouble-byte(DEST[63..48] + SRC[63..48]);

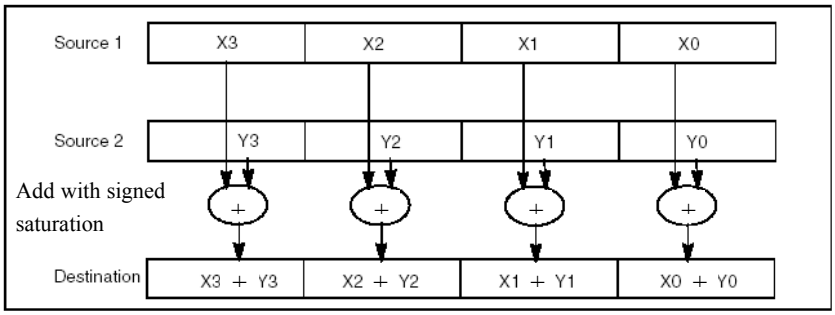
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PADDSB 对 64－bit DEST 操作数中 8 个打包的字节数和 64－bit SRC 操作数中 8 个打包的字节数，执行 SIMD 有符号加法，结果存入 DEST 操作数中相应的位置。PADDSD 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号加法，结果存入 DEST 操作数中相应的位置。下图示例 PADDSD 的操作过程，PADDSB 类推。



执行周期：1 cycle

PADDUSB/D

句型： **PADDUSB/D** MR1，MR2

PADDUSB/D *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					100001						

操作：

PADDUSB instruction with 64-bit operands:

DEST[7..0] ← SaturateToUnsignedByte(DEST[7..0] + SRC (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

DEST[63..56] ← SaturateToUnsignedByte(DEST[63..56] + SRC[63..56]);

PADDUSD instruction with 64-bit operands:

DEST[15..0] ← SaturateToUnsignedDouble-byte(DEST[15..0] + SRC[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

DEST[63..48] ← SaturateToUnsignedDouble-byte(DEST[63..48] + SRC[63..48]);

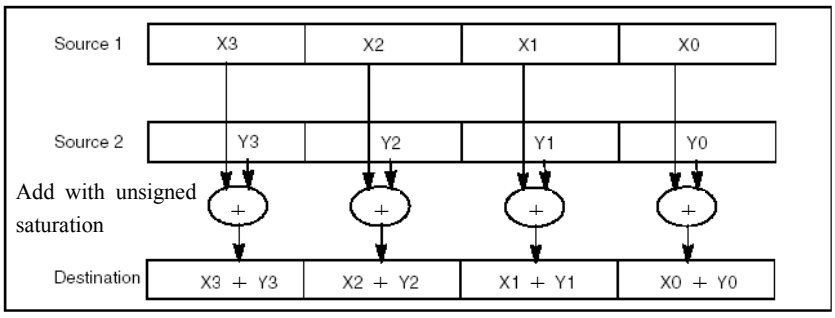
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PADDUSB 对 64－bit DEST 操作数中 8 个打包的字节数和 64－bit SRC 操作数中 8 个打包的字节数，执行 SIMD 无符号加法，结果存入 DEST 操作数中相应的位置。PADDUSD 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 无符号加法，结果存入 DEST 操作数中相应的位置。下图示例 PADDUSD 的操作过程，PADDUSB 类推。



执行周期：1 cycle

PAND

句型： **PAND** MR1，MR2

PAND *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			11		MR1			00000					00000					100100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm					disp					100100						

操作：

PAND：

DEST ← DEST AND SRC;

描述： PAND 对 64－bit DEST 操作数和 64－bit SRC 操作数，执行按位逻辑与运算，结果存入 DEST 操作数。

执行周期：1 cycle

PAVGB/D

句型： **PAVGB/D** MR1，MR2

PAVGB/D MR1，Modm(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					111100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					111100						

操作：

PAVGB instruction with 64-bit operands:

SRC[7-0] ← (SRC[7-0] + DEST[7-0] + 1] >> 1; * temp sum before shifting is 9 bits *

* repeat operation performed for bytes 2 through 6;

SRC[63-56] ← (SRC[63-56] + DEST[63-56] + 1) >> 1;

PAVGD instruction with 64-bit operands:

SRC[15-0] ← (SRC[15-0] + DEST[15-0] + 1) >> 1; * temp sum before shifting is 17 bits *

* repeat operation performed for double-bytes 2 and 3;

SRC[63-48] ← (SRC[63-48] + DEST[63-48] + 1] >> 1;

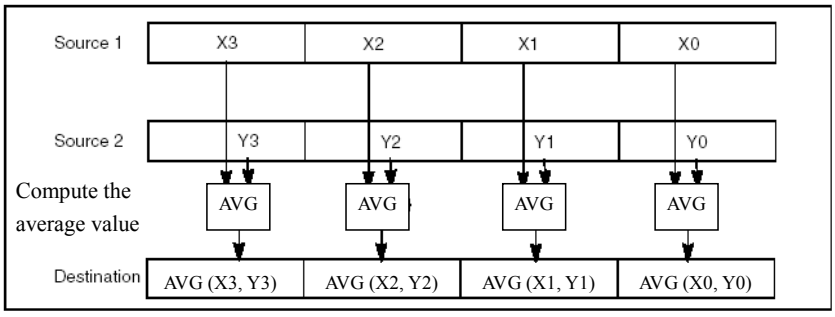
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PAVGB/D 对 64－bit DEST 操作数中打包的字节数/2 字节数和 64－bit SRC 操作数中打包的字节数/2 字节数，执行 SIMD 加法，每个和值再加 1，相应结果右移 1bit 作为 2 个数的平均值，存入 DEST 操作数中。下图示例了 PAVGD 的操作过程，PAVGB 类推。



执行周期：1 cycle

PCMPEQB/D/Q

句型： **PCMPEQB/D/Q** MR1，MR2

PCMPEQB/D/Q *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					110100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					110100						

操作：

PCMPEQB instruction with 64-bit operands:
IF DEST[7..0] = SRC[7..0]
THEN DEST[7 0] ← FFH;
ELSE DEST[7..0] ← 0;
* Continue comparison of 2nd through 7th bytes in DEST and SRC *
IF DEST[63..56] = SRC[63..56]
THEN DEST[63..56] ← FFH;
ELSE DEST[63..56] ← 0;

PCMPEQD instruction with 64-bit operands:
IF DEST[15..0] = SRC[15..0]
THEN DEST[15..0] ← FFFFH;
ELSE DEST[15..0] ← 0;
* Continue comparison of 2nd and 3rd double-bytes in DEST and SRC *
IF DEST[63..48] = SRC[63..48]
THEN DEST[63..48] ← FFFFH;
ELSE DEST[63..48] ← 0;

PCMPEQQ instruction with 64-bit operands:
IF DEST[31..0] = SRC[31..0]
THEN DEST[31..0] ← FFFFFFFFH;
ELSE DEST[31..0] ← 0;
IF DEST[63..32] = SRC[63..32]
THEN DEST[63..32] ← FFFFFFFFH;
ELSE DEST[63..32] ← 0;

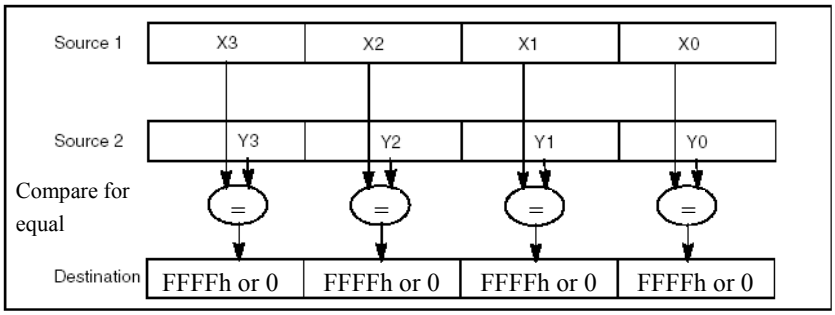
操作数说明：MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PCMPEQB/D/Q 对 64－bit DEST 操作数中打包的字节数/2 字节数/4 字节数和 64－bit SRC 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 相等比较，如果相等结果全置 1，否则全置 0，结果存入 DEST 操作数中。下图示例了 PCMPEQD 的操作过程，其它类推。



执行周期：1 cycle

PCMPGTB/D/Q

句型： **PCMPGTB/D/Q** MR1，MR2

PCMPGTB/D/Q *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					110000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					110000						

操作：

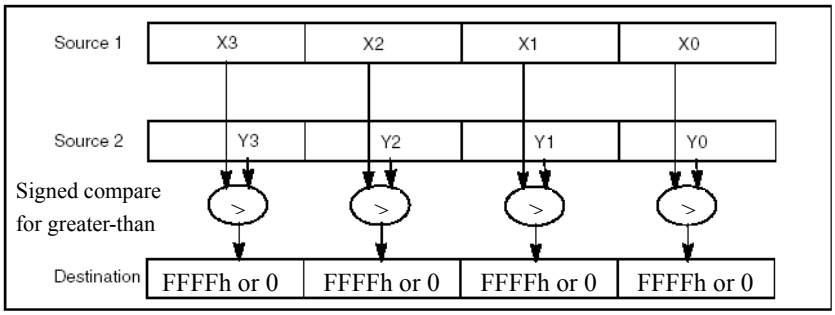
PCMPGTB instruction with 64-bit operands:
IF DEST[7..0] > SRC[7..0]
THEN DEST[7 0] ← FFH;
ELSE DEST[7..0] ← 0;
* Continue comparison of 2nd through 7th bytes in DEST and SRC *
IF DEST[63..56] > SRC[63..56]
THEN DEST[63..56] ← FFH;
ELSE DEST[63..56] ← 0;

PCMPGTD instruction with 64-bit operands:
IF DEST[15..0] > SRC[15..0]
THEN DEST[15..0] ← FFFFH;
ELSE DEST[15..0] ← 0;
* Continue comparison of 2nd and 3rd double-bytes in DEST and SRC *
IF DEST[63..48] > SRC[63..48]
THEN DEST[63..48] ← FFFFH;
ELSE DEST[63..48] ← 0;

PCMPGTQ instruction with 64-bit operands:
IF DEST[31..0] > SRC[31..0]
THEN DEST[31..0] ← FFFFFFFFH;
ELSE DEST[31..0] ← 0;
IF DEST[63..32] > SRC[63..32]
THEN DEST[63..32] ← FFFFFFFFH;
ELSE DEST[63..32] ← 0;

操作数说明：MR1： MDS 寄存器
MR2： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PCMPGTB/D/Q 对 64－bit DEST 操作数中打包的字节数/2 字节数/4 字节数和 64－bit SRC 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 有符号比较，如果大于结果全置 1，否则全置 0，结果存入 DEST 操作数中。下图示例了 PCMPGTD 的操作过程，其它类推。



执行周期：1 cycle

PLOADO

句型： *PLOADO MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm						disp					110111					

操作：

PLOADO：

MR1[63-0] ← memory

操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PLOADO 从 memory 中指定的位置读取 64bit 数据，写入到 MDS 寄存器 MR1 中。

执行周期：1 cycle

PMADDQD

句型： **PMADDQD** MR1，MR2

PMADDQD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			11		MR1			00000					00000					101000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm					disp					101000						

操作：

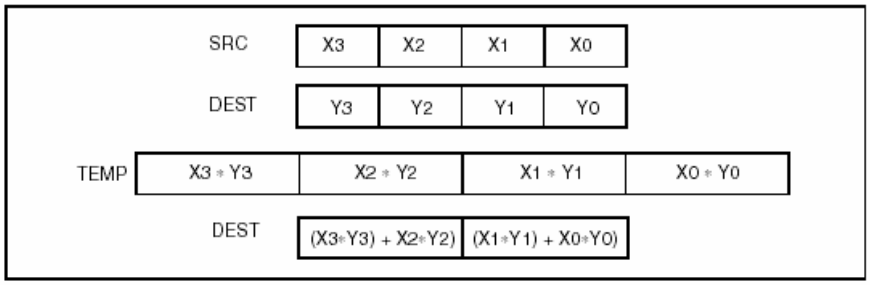
PMADDQD instruction with 64-bit operands:

DEST[31..0] ← (DEST[15..0]×SRC[15..0]) + (DEST[31..16]×SRC[31..16]);

DEST[63..32] ← (DEST[47..32]×SRC[47..32]) + (DEST[63..48]×SRC[63..48]);

* Signed multiplication *

描述： PMADDQD 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，然后相邻的 2 个 32－bit 结果相加成 1 个 32－bit 结果，最后结果存入 DEST 操作数。下图示例了 PMADDQD 的操作过程。



执行周期：3 cycles

PMAXSD

句型： **PMAXSD** MR1，MR2

PMAXSD MR1，Modm(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			01		MR1			00000					00000					111000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			01		MR1			Modm					disp					111000						

操作：

PMAXSD instruction for 64-bit operands:

IF DEST[15-0] > SRC[15-0]) THEN

(DEST[15-0] ← DEST[15-0];

ELSE

(DEST[15-0] ← SRC[15-0];

FI

* repeat operation for 2nd and 3rd double-bytes in source and destination operands *

IF DEST[63-48] > SRC[63-48]) THEN

(DEST[63-48] ← DEST[63-48];

ELSE

(DEST[63-48] ← SRC[63-48];

FI

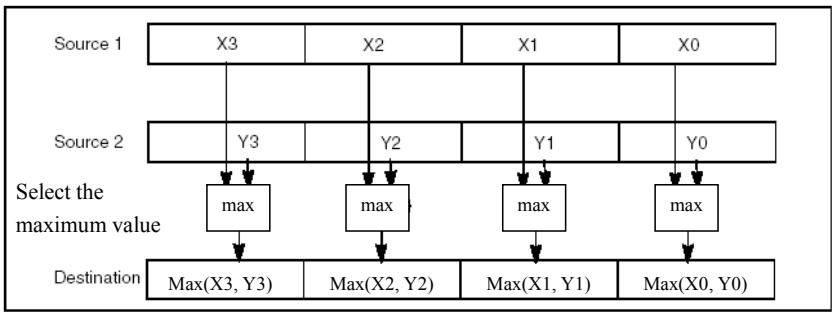
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMAXSD 对 64－bit DEST 操作数中 4 个打包的有符号 2 字节数和 64－bit SRC 操作数中 4 个打包的有符号 2 字节数，执行 SIMD 有符号比较，相应较大的数存入 DEST 操作数中。下图示例了 PMAXSD 的操作过程。



执行周期：1 cycle

PMAXUB

句型： **PMAXUB** MR1，MR2

PMAXUB MR1，Modm(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			00		MR1			00000					00000					111001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			00		MR1			Modm					disp					111001						

操作：

PMAXUB instruction for 64-bit operands:

IF DEST[7-0] > SRC[17-0]) THEN

(DEST[7-0] ← DEST[7-0];

ELSE

(DEST[7-0] ← SRC[7-0];

FI

* repeat operation for 2nd through 7th bytes in source and destination operands *

IF DEST[63-56] > SRC[63-56]) THEN

(DEST[63-56] ← DEST[63-56];

ELSE

(DEST[63-56] ← SRC[63-56];

FI

操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMAXUB 对 64－bit DEST 操作数中 8 个打包的无符号字节数和 64－bit SRC 操作数中 8 个打包的无符号字节数，执行 SIMD 比较，相应较大的数存入 DEST 操作数中。PMAXUB 操作过程类似 PMAXSD。

执行周期：1 cycle

PMFHI

句型： **PMFHI** MR1，rs

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			10		MR1			rs						00000					010000					

操作：

PMFHI:

rs[31-0] ← MR1[63-32]

操作数说明： MR1： MDS 寄存器

rs： 通用寄存器

描述： PMFHI 将 MDS 寄存器 MR1 的高 32 位的值写入到通用寄存器 rs。

执行周期：1 cycle

PMFLO

句型： **PMFLO** MR1，rs

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			10		MR1			rs						00000					010010					

操作：

PMFHI:

rs[31-0] ← MR1[31-0]

操作数说明： MR1： MDS 寄存器

rs： 通用寄存器

描述： PMFLO 将 MDS 寄存器 MR1 的低 32 位的值写入到通用寄存器 rs。

执行周期： **1** cycle

PMINSD

句型： **PMINSD** MR1，MR2

PMINSD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			01		MR1			00000					00000					111010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			01		MR1			Modm					disp					111010						

操作：

PMINSD instruction for 64-bit operands:

IF DEST[15-0] < SRC[15-0]) THEN

(DEST[15-0] ← DEST[15-0];

ELSE

(DEST[15-0] ← SRC[15-0];

FI

* repeat operation for 2nd and 3rd double-bytes in source and destination operands *

IF DEST[63-48] < SRC[63-48]) THEN

(DEST[63-48] ← DEST[63-48];

ELSE

(DEST[63-48] ← SRC[63-48];

FI

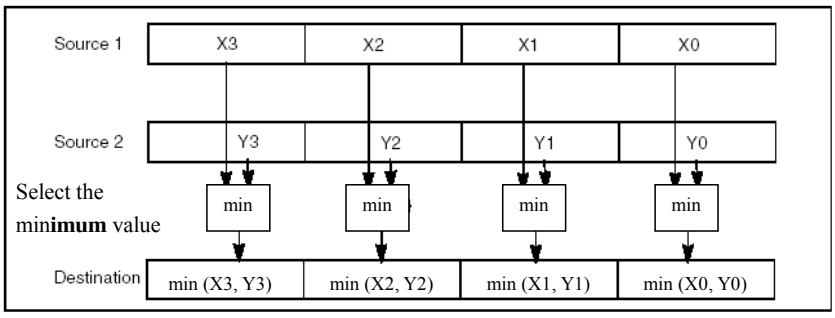
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMINSD 对 64－bit DEST 操作数中 4 个打包的有符号 2 字节数和 64－bit SRC 操作数中 4 个打包的有符号 2 字节数，执行 SIMD 有符号比较，相应较小的数存入 DEST 操作数中。下图示例了 PMINSD 的操作过程。



执行周期：1 cycle

PMINUB

句型： **PMINUB** MR1，MR2

PMINUB MR1，Modm(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			00		MR1			00000					00000					111011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			00		MR1			Modm					disp					111011						

操作：

PMINUB instruction for 64-bit operands:

IF DEST[7-0] < SRC[17-0]) THEN

(DEST[7-0] ← DEST[7-0];

ELSE

(DEST[7-0] ← SRC[7-0];

FI

* repeat operation for 2nd through 7th bytes in source and destination operands *

IF DEST[63-56] < SRC[63-56]) THEN

(DEST[63-56] ← DEST[63-56];

ELSE

(DEST[63-56] ← SRC[63-56];

FI

操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMINUB 对 64－bit DEST 操作数中 8 个打包的无符号字节数和 64－bit SRC 操作数中 8 个打包的无符号字节数，执行 SIMD 比较，相应较小的数存入 DEST 操作数中。PMINUB 操作过程类似 PMINSD。

执行周期：1 cycle

PMTLO

句型： **PMTLO** MR1，rs

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			10		MR1			rs						00000					010011					

操作：

PMTLO:

MR1[31-0] ← rs[31-0]

操作数说明： MR1： MDS 寄存器

rs： 通用寄存器

描述： PMTLO 将通用寄存器 rs 的值写入到 MDS 寄存器 MR1 的低 32 位。

执行周期： **1** cycle

PMTHI

句型： **PMTHI** MR1，rs

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			10		MR1			rs						00000					010001					

操作：

PMTHI:

MR1[63-32] ← rs[31-0]

操作数说明： MR1： MDS 寄存器

rs： 通用寄存器

描述： PMTHI 将通用寄存器 rs 的值写入到 MDS 寄存器 MR1 的高 32 位。

执行周期： **1** cycle

PMULHSD

句型： **PMULHSD** MR1，MR2

PMULHSD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					011100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					011100						

操作：

PMULHSD:

TEMP0[31-0] ← DEST[15-0]×SRC[15-0]; * Signed multiplication *

TEMP1[31-0] ← DEST[31-16]×SRC[31-16];

TEMP2[31-0] ← DEST[47-32]×SRC[47-32];

TEMP3[31-0] ← DEST[63-48]×SRC[63-48];

DEST[15-0] ← TEMP0[31-16];

DEST[31-16] ← TEMP1[31-16];

DEST[47-32] ← TEMP2[31-16];

DEST[63-48] ← TEMP3[31-16];

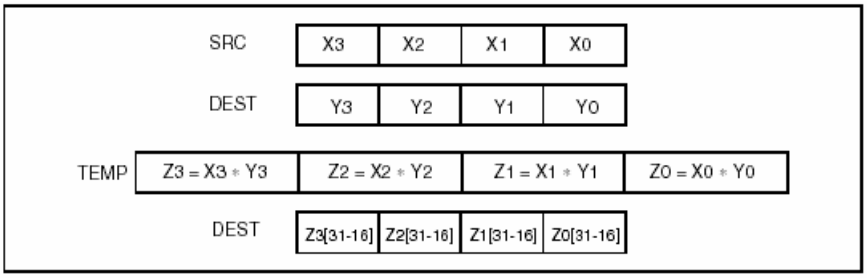
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的高 16－bit 存入 DEST 操作数中相应的位置。下图示例 PMULHSD 的操作过程。



执行周期：2 cycles

PMULHUD

句型： **PMULHUD** MR1，MR2

PMULHUD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					011101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					011101						

操作：

PMULHUD:

TEMP0[31-0] ← DEST[15-0]×SRC[15-0]; * Unsigned multiplication *

TEMP1[31-0] ← DEST[31-16]×SRC[31-16];

TEMP2[31-0] ← DEST[47-32]×SRC[47-32];

TEMP3[31-0] ← DEST[63-48]×SRC[63-48];

DEST[15-0] ← TEMP0[31-16];

DEST[31-16] ← TEMP1[31-16];

DEST[47-32] ← TEMP2[31-16];

DEST[63-48] ← TEMP3[31-16];

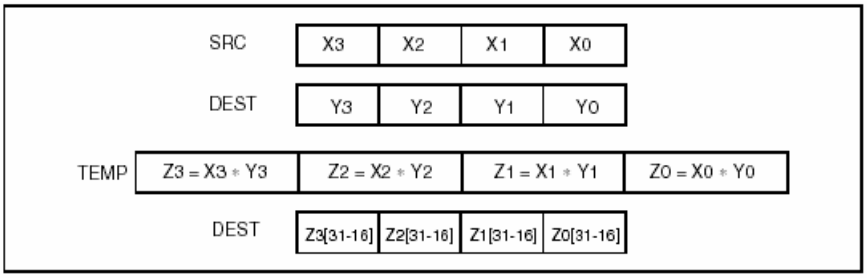
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的高 16－bit 存入 DEST 操作数中相应的位置。下图示例 PMULHUD 的操作过程。



执行周期：2 cycles

PMULLSD

句型： **PMULLSD** MR1，MR2

PMULLSD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					011000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					011000						

操作：

PMULLSD:

TEMP0[31-0] ← DEST[15-0]×SRC[15-0]; * Signed multiplication *

TEMP1[31-0] ← DEST[31-16]×SRC[31-16];

TEMP2[31-0] ← DEST[47-32]×SRC[47-32];

TEMP3[31-0] ← DEST[63-48]×SRC[63-48];

DEST[15-0] ← TEMP0[15-0];

DEST[31-16] ← TEMP1[15-0];

DEST[47-32] ← TEMP2[15-0];

DEST[63-48] ← TEMP3[15-0];

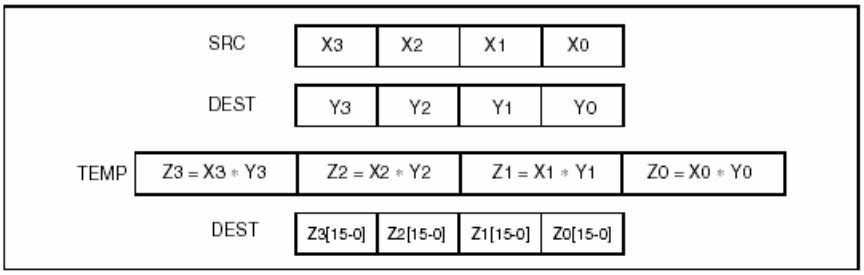
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的低 16－bit 存入 DEST 操作数中相应的位置。下图示例 PMULLSD 的操作过程。



执行周期：2 cycles

PMULLUD

句型： **PMULLUD** MR1，MR2

PMULLUD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					011001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					011001						

操作：

PMULLUD:

TEMP0[31-0] ← DEST[15-0]×SRC[15-0]; * Unsigned multiplication *

TEMP1[31-0] ← DEST[31-16]×SRC[31-16];

TEMP2[31-0] ← DEST[47-32]×SRC[47-32];

TEMP3[31-0] ← DEST[63-48]×SRC[63-48];

DEST[15-0] ← TEMP0[15-0];

DEST[31-16] ← TEMP1[15-0];

DEST[47-32] ← TEMP2[15-0];

DEST[63-48] ← TEMP3[15-0];

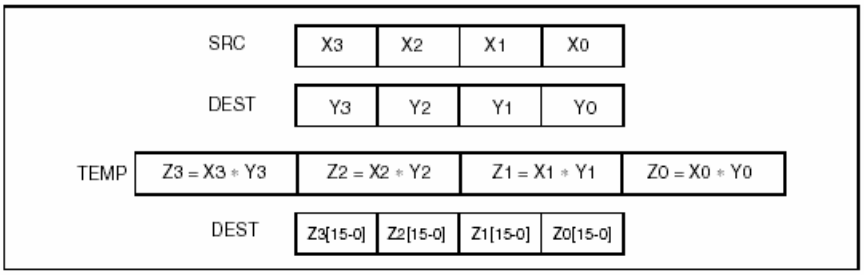
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 无符号乘法，每个 32－bit 结果的低 16－bit 存入 DEST 操作数中相应的位置。下图示例 PMULLUD 的操作过程。



执行周期：2 cycles

PNOR

句型： **PNOR** MR1，MR2

PNOR *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			11		MR1			00000					00000					100111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm					disp					100111						

操作：

PNOR：

DEST ← DEST NOR SRC;

描述： PNOR 对 64－bit DEST 操作数和 64－bit SRC 操作数，执行按位逻辑或非运算，结果存入 DEST 操作数。

执行周期：1 cycle

POR

句型： **POR** MR1，MR2

POR MR1，Modm(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			11		MR1			00000					00000					100101						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm					disp					100101						

操作：

POR：

DEST ← DEST OR SRC;

描述： POR 对 64－bit DEST 操作数和 64－bit SRC 操作数，执行按位逻辑或运算，结果存入 DEST 操作数。

执行周期：1 cycle

PSADBD

句型： **PSADBD** MR1，MR2

PSADBD *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			11		MR1			00000					00000					101001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm					disp					101001						

操作：

PSADBD instructions when using 64-bit operands:

TEMP0 ← ABS(DEST[7-0] － SRC[7-0]);

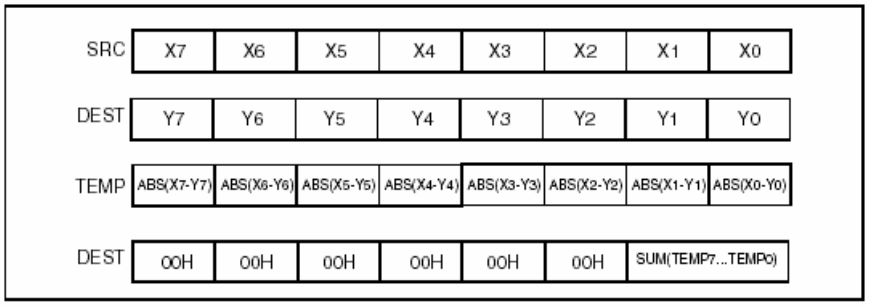
* repeat operation for bytes 2 through 6 *;

TEMP7 ← ABS(DEST[63-56] － SRC[63-56]);

DEST[15:0] ← SUM(TEMP0…TEMP7);

DEST[63:16] ← 000000000000H;

描述： PSADBD 对 64－bit DEST 操作数中 8 个打包的字节数和 64－bit SRC 操作数中 8 个打包的字节数，执行 SIMD 减法，取绝对值得到绝对差值，然后 8 个绝对差值相加成 1 个 16－bit 无符号数，存入 DEST 操作数的低 16－bit，DEST 操作数的高 48bit 置 0。下图示例了 PSADBD 的操作过程。



执行周期：2 cycles

PSRAD/Q

句型： **PSRAD/Q** MR1，sa
PSRAD/Q MR1，MR2
PSRAD/Q *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		000			gg		MR1			00000					sa					000011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					000011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					000011						

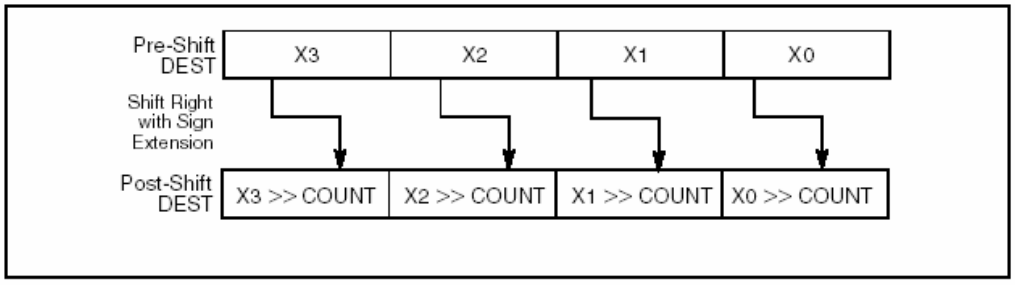
操作：

```
PSRAD:
IF (COUNT > 15)
THEN COUNT ← 16;
FI;
DEST[15..0] ← SignExtend(DEST[15..0] >> COUNT);
* repeat shift operation for 2nd and 3rd double-bytes *;
DEST[63..48] ← SignExtend(DEST[63..48] >> COUNT);

PSRAQ:
IF (COUNT > 31)
THEN COUNT ← 32;
FI;
ELSE
DEST[31..0] ← SignExtend(DEST[31..0] >> COUNT);
DEST[63..32] ← SignExtend(DEST[63..32] >> COUNT);
```

操作数说明： MR1： MDS 寄存器
MR2： MDS 寄存器
Sa： 立即数
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64bit DEST 操作数中打包的 2 字节/4 字节进行 SIMD 算术右移，结果存入 DEST 操作数。下图示例了 PSRAQ 的操作过程，其它类推。



执行周期：1 cycle

PSRLD/Q

句型： **PSRLD/Q** MR1，sa
PSRLD/Q MR1，MR2
PSRLD/Q *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		000			gg		MR1			00000					sa					000010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					000010						

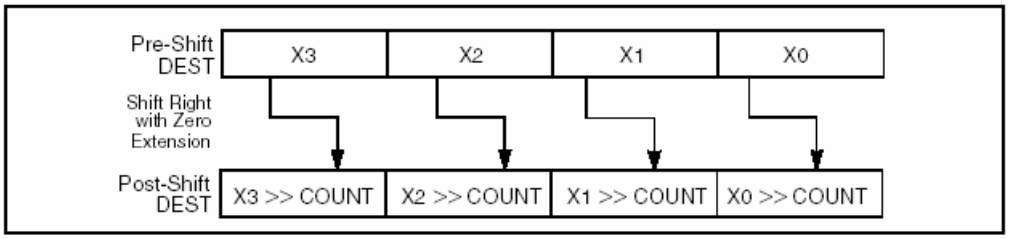
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					000010						

操作：

```
PSRLD:
        IF (COUNT > 15)
        THEN
                DEST[64..0] ← 0000000000000000H
        ELSE
                DEST[15..0] ← ZeroExtend(DEST[15..0] >> COUNT);
                * repeat shift operation for 2nd and 3rd double-bytes *;
                DEST[63..48] ← ZeroExtend(DEST[63..48] >> COUNT);
        FI;
PSRLQ:
       IF (COUNT > 31)
       THEN
              DEST[64..0] ← 0000000000000000H
       ELSE
              DEST[31..0] ← ZeroExtend(DEST[31..0] >> COUNT);
              DEST[63..32] ← ZeroExtend(DEST[63..32] >> COUNT);
       FI;
```

操作数说明： MR1： MDS寄存器
MR2： MDS 寄存器
Sa： 立即数
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64bit DEST 操作数中打包的 2 字节/4 字节/8 字节进行 SIMD 逻辑右移，结果存入 DEST 操作数。下图示例了 PSRLQ 的操作过程，其它类推。



执行周期： **1** cycle

PSLLD/Q

句型： **PSLLD/Q** MR1，sa
PSLLD/Q MR1，MR2
PSLLD/Q/O *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		000			gg		MR1			00000					sa					000000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					000000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					000000						

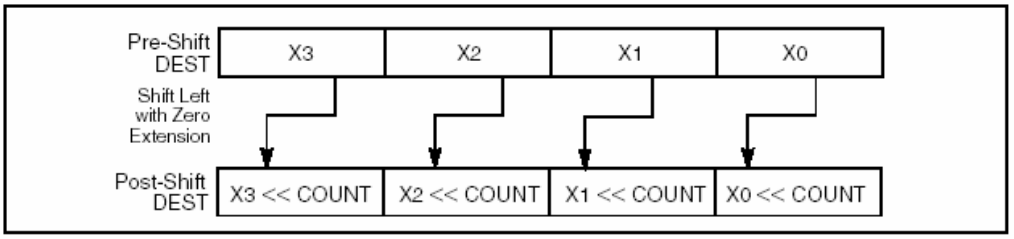
操作：

```
PSLLD:
        IF (COUNT > 15)
                THEN
                        DEST[64..0] ← 0000000000000000H
        ELSE
                DEST[15..0] ← ZeroExtend(DEST[15..0] << COUNT);
                * repeat shift operation for 2nd and 3rd double-bytes *;
                DEST[63..48] ← ZeroExtend(DEST[63..48] << COUNT);
        FI;

PSLLQ:
        IF (COUNT > 31)
                THEN
                    DEST[64..0] ← 0000000000000000H
        ELSE
                DEST[31..0] ← ZeroExtend(DEST[31..0] << COUNT);
                DEST[63..32] ← ZeroExtend(DEST[63..32] << COUNT);
        FI;
```

操作数说明： MR1： MDS 寄存器
MR2： MDS 寄存器
Sa： 立即数
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64bit DEST 操作数中打包的 2 字节/4 字节/8 字节进行 SIMD 逻辑左移，结果存入 DEST 操作数。下图示例了 PSLLQ 的操作过程，其它类推。



执行周期： **1** cycle

PSTOREO

句型： *PSTOREO MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm						disp					111111					

操作：

PSTOREO：

Memory ← MR1[63-0]

操作数说明： MR1： MDS 寄存器
MR2： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PSTOREO 将 MDS 寄存器 MR1 中的 64bit 数据写入到 memory 中指定的位置。

执行周期：1 cycle

PSUBB/D/Q

句型： **PSUBB/D/Q** MR1，MR2

PSUBB/D/Q *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					101110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					101110						

操作：

PSUBB instruction with 64-bit operands:

DEST[7..0] ← DEST[7..0] − SRC[7..0];

* repeat add operation for 2nd through 7th byte *;

DEST[63..56] ← DEST[63..56] − SRC[63..56];

PSUBD instruction with 64-bit operands:

DEST[15..0] ← DEST[15..0] − SRC[15..0];

* repeat add operation for 2nd and 3th double-byte *;

DEST[63..48] ← DEST[63..48] − SRC[63..48];

PSUBQ instruction with 64-bit operands:

DEST[31..0] ← DEST[31..0] − SRC[31..0];

DEST[63..32] ← DEST[63..32] − SRC[63..32];

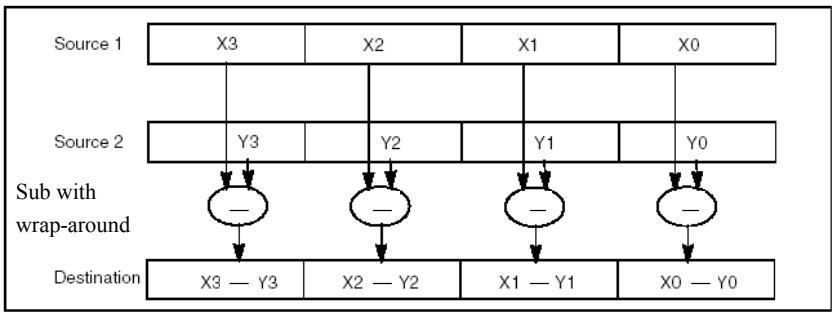
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSUBB/D/Q 对 64－bit DEST 操作数中打包的字节数/2 字节数/4 字节数和 64－bit SRC 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 减法，结果存入 DEST 操作数中，溢出被忽略。下图示例 PSUBD 的操作过程，其它类推。



执行周期：1 cycle

PSUBSB/D

句型： **PSUBSB/D** MR1，MR2

PSUBSB/D *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					100010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					100010						

操作：

PSUBSB instruction with 64-bit operands:

DEST[7..0] ← SaturateToSignedByte(DEST[7..0] − SRC (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

DEST[63..56] ← SaturateToSignedByte(DEST[63..56] − SRC[63..56]);

PSUBSD instruction with 64-bit operands:

DEST[15..0] ← SaturateToSignedDouble-byte(DEST[15..0] − SRC[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

DEST[63..48] ← SaturateToSignedDouble-byte(DEST[63..48] − SRC[63..48]);

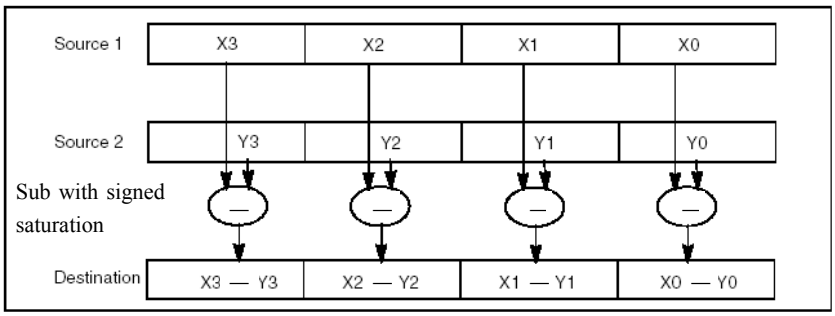
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSUBSB 对 64－bit DEST 操作数中 8 个打包的字节数和 64－bit SRC 操作数中 8 个打包的字节数，执行 SIMD 有符号减法，结果存入 DEST 操作数中相应的位置。PSUBSD 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号减法，结果存入 DEST 操作数中相应的位置。下图示例 PSUBSD 的操作过程，PSUBSB 类推。



执行周期：1 cycle

PSUBUSB/D

句型： **PSUBUSB/D** MR1，MR2

PSUBUSB/D *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					100011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					100011						

操作：

PSUBUSB instruction with 64-bit operands:

DEST[7..0] ← SaturateToUnsignedByte(DEST[7..0] − SRC (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

DEST[63..56] ← SaturateToUnsignedByte(DEST[63..56] − SRC[63..56]);

PSUBUSD instruction with 64-bit operands:

DEST[15..0] ← SaturateToUnsignedDouble-byte(DEST[15..0] − SRC[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

DEST[63..48] ← SaturateToUnsignedDouble-byte(DEST[63..48] − SRC[63..48]);

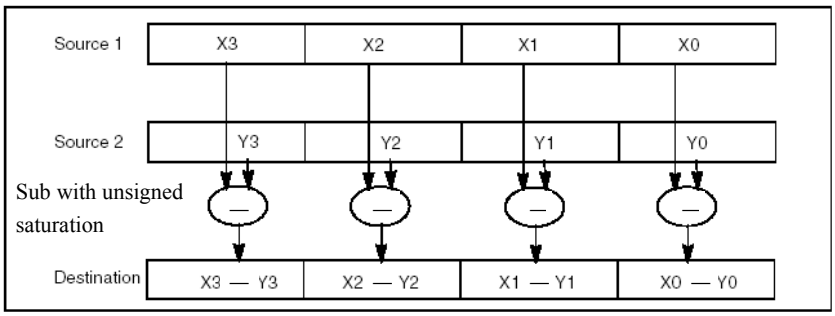
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSUBUSB 对 64－bit DEST 操作数中 8 个打包的字节数和 64－bit SRC 操作数中 8 个打包的字节数，执行 SIMD 无符号减法，结果存入 DEST 操作数中相应的位置。PSUBUSD 对 64－bit DEST 操作数中 4 个打包的 2 字节数和 64－bit SRC 操作数中 4 个打包的 2 字节数，执行 SIMD 无符号减法，结果存入 DEST 操作数中相应的位置。下图示例 PSUBUSD 的操作过程，PSUBUSB 类推。



执行周期：1 cycle

PUNPCKHBD/DQ/QO

句型： **PUNPCKHBD/DQ/QO** MR1，MR2

PUNPCKHBD/DQ/QO *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					001010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					001010						

操作：

PUNPCKHBD:

DEST[7..0] ←DEST[39..32];

DEST[15..8] ←SRC[39..32];

DEST[23..16] ←DEST[47..40];

DEST[31..24] ←SRC[47..40];

DEST[39..32] ←DEST[55..48];

DEST[47..40] ←SRC[55..48];

DEST[55..48] ←DEST[63..56];

DEST[63..56] ←SRC[63..56];

PUNPCKHDQ:

DEST[15..0] ←DEST[47..32];

DEST[31..16] ←SRC[47..32];

DEST[47..32] ←DEST[63..48];

DEST[63..48] ←SRC[63..48];

PUNPCKHQO:

DEST[31..0] ←DEST[63..32]

DEST[63..32] ←SRC[63..32];

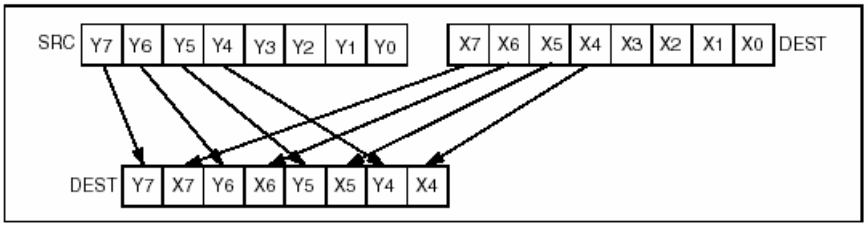
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 将 64bit SRC 和 DEST 操作数中打包的字节/2 字节/4 字节相交织，取高 64bit 存入 DEST 操作数。下图示例了 PUNPCKHBD 的操作过程，其它类推。



执行周期： **1** cycle

PUNPCKLBD/DQ/QO

句型： **PUNPCKLBD/DQ/QO** MR1，MR2

PUNPCKLBD/DQ/QO *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			gg		MR1			00000					00000					001011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MR1			Modm					disp					001011						

操作：

PUNPCKLBD:

DEST[63..56] ←SRC[31..24];

DEST[55..48] ←DEST[31..24];

DEST[47..40] ←SRC[23..16];

DEST[39..32] ←DEST[23..16];

DEST[31..24] ←SRC[15..8];

DEST[23..16] ←DEST[15..8];

DEST[15..8] ←SRC[7..0];

DEST[7..0] ←DEST[7..0];

PUNPCKLDQ:

DEST[63..48] ←SRC[31..16];

DEST[47..32] ←DEST[31..16];

DEST[31..16] ←SRC[15..0];

DEST[15..0] ←DEST[15..0];

PUNPCKLQO:

DEST[63..32] ←SRC[31..0];

DEST[31..0] ←DEST[31..0];

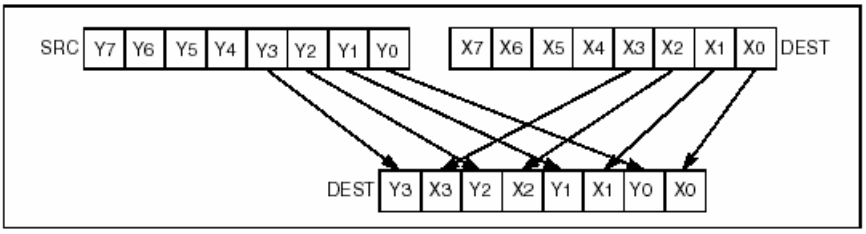
操作数说明： MR1： MDS 寄存器

MR2： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 将 64bit SRC 和 DEST 操作数中打包的字节/2 字节/4 字节相交织，取低 64bit 存入 DEST 操作数。下图示例了 PUNPCKLBD 的操作过程，其它类推。



执行周期： **1** cycle

PXOR

句型： **PXOR** MR1，MR2

PXOR *MR1，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MR2			11		MR1			00000					00000					100110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			11		MR1			Modm					disp					100110						

操作：

PXOR：

DEST ← DEST XOR SRC;

描述： PXOR 对 64－bit DEST 操作数和 64－bit SRC 操作数，执行按位逻辑异或运算，结果存入 DEST 操作数。

执行周期：1 cycle

RFE

句型：RFE

指令编码：

31 26	25	24 6	5 0
010000	1	000_0000_0000_0000_0000	010000

操作：$SR_{31..4}|SR_{5..2}$→ SR

操作数说明：

SR：状态寄存器

描述：恢复先前中断标志和状态寄存器的核心/用户太模式位（IEp 和 KUp）到当前对应的状态位（IEc 和 KUc），恢复先前的状态位（IEo 和 KUo）到对应的状态位（IEp 和 KUp），先前的状态位保持不变。

执行周期：1 cycle

RPTB

句型： **RPTB** direct

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000									direct																0	110111						

操作：

unsigned(direct × 4) + PC → RE

(PC of RPTB) + 4 → RS

操作数说明：

direct： 16bit 立即数

RE： 重复结束地址寄存器

RS： 重复开始地址寄存器

描述：

RPTB 指令可以对指定的一段程序块重复执行若干次，不需要进行跳转判断，无跳转开销。

在 RPTB 指令之前要有指令对 RC 寄存器进行赋值，因为 RPTB 指令本身只对 RE 和 RS 寄存器进行赋值，并不对 RC 赋值。

举例：

addi RC, $0, 2

RPTB @3h

Inst A

Inst B

Inst C

Inst D

指令 A\B\C\三条指令循环三次，块大小为 3。

执行周期：1 cycle

RPTS

句型： **RPTS** direct

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						direct																			1	110111					

操作：

unsigned(direct x 4) → RC

PC → RE

PC → RS

操作数说明：

direct： 16bit 立即数

RE： 重复结束地址寄存器

RS： 重复开始地址寄存器

描述：

RPTS 指令对其后的一条指令重复执行，不需要进行跳转判断，无跳转开销。执行的次数在指令中由 Direct = n 指出，执行 n+1 次；不需要对 RC 寄存器进行赋值。

执行周期：1 cycle

举例：

RPTS @1

Inst A

Inst B

InstC

指令 A 循环 2 次。

SB

句型： **SB** rt, offset(base) 或者

SB rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101000						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101000						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，

Byte(GPR(rt)) → mem(Vaddr)

操作数说明： base： 寄存器 （通用寄存器 0～30）

rt： 寄存器 （通用寄存器 0～31）

ARm： 间接寻址（辅助寄存器 0～7）

Disp： 立即数

offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SB R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，Byte(GPR(R5)) → mem(Vaddr)
SB R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，Byte(GPR(R5)) → mem(Vaddr)

SH

句型： **SH** rt, offset(base) 或者
SH rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101001						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101001						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，
Byte(GPR(rt)) → mem(Vaddr)

操作数说明： base： 寄存器 （通用寄存器 0～30）
rt： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Disp： 立即数
offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SH R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，Byte(GPR(R5)) → mem(Vaddr)
SH R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，Byte(GPR(R5)) → mem(Vaddr)

SLL

句型： **SLL** rd，rt，sa 或者
SLL rd，rt，*+ARm(disp) 或者
SLL rt，@direct 或者
SLL rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00000					Rt					Rd					Sa						000000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		14	13	12	11	10	9	8	7	6		4	3	2	1	0
000000						01		ARm			Rt					Rd					Disp						000000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
000000						10		Direct1			Rt					Direct2										000000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						11		ARm			Rt					Modm					Disp						000000					

操作：

GPR(rt) << sa→ GPR(rd) 或者
GPR(rt) << mem(*+ARm(disp))→ GPR(rd) 或者
GPR(rt) << mem({Direct1,Direct2})→ GPR(rt) 或者
GPR(rt) << mod(ARm)→ GPR(rt)

操作数说明：

rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Sa： 立即数
Disp： 立即数
T： 寻址模式选择位。

T	源操作数 1	源操作数 2
00	寄存器	寄存器
01	寄存器	*+ARm(disp)
10	寄存器	直接寻址
11	寄存器	间接寻址

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SLL R3，R7，04h	GPR(R7) << 04h→ GPR(R3)
SLL R3，R7，*+AR1(04h)	GPR(R7) << mem(*+AR1(04h))→ GPR(R3)
SLL R3，0840h	GPR(R3) << mem(0840h)→ GPR(R3)
SLL R3，*AR2++(IR1)	GPR(R3) << mod(AR2)→ GPR(R3)

SLLV

句型：

SLLV	rd，rt，rs	或者
SLLV	dst，*+ARn(disp2)， *+ARm(disp1)	或者
SLLV	dst，rt，mod(ARm)	或者
SLLV	dst，Imm，*+ARm(disp)	或者
SLLV	dst，mod(ARm)，rs	或者
SLLV	dst，*+ARm(disp)，rs	或者
SLLV	dst，mod(ARn)，mod(ARm)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					Rt					Rd					00000					000100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	000100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			Rt					01		Dst			Modm				1	000100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	000100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					00		ARm			10		Dst			Modm				1	000100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					01		ARm			10		Dst			Disp				1	000100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		Dst			Modn				1	000100						

操作：

GPR(rt) << GPR(Rs)[4:0] → GPR(rd)	或者
mem(*+ARn(disp2)) << Mem(*+ARm(disp1))[4:0] → GPR(dst)	或者
GPR(Rt) << mod(ARm)[4:0] → GPR(dst)	或者
sign(Imm) << Mem(*+ARm(disp)) → GPR(dst)	或者
mod(ARm) << GPR(Rs) → GPR(dst)	或者
mem(*+ARm(disp)) << GPR(Rs) → GPR(dst)	或者
modn(ARn) << Modm(ARm) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
SLLV R5，R3，R7	GPR(R3) << GPR(R7) [4:0]→ GPR(R5)
SLLV R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) << mem(*+AR2(8h))[4:0] → GPR(R5)
SLLV R5，R3，*AR2++(IR1)	Mem(*AR2++(IR1)) << GPR(R3) [4:0] → GPR(R5)
SLLV R5，08h， *+AR1(1h)	sign(08h) << Mem(*+AR1(1h)) → GPR(R5)
SLLV R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) << GPR(R3) [4:0] → GPR(R5)
SLLV R5，*+AR1(1h)，R3	mem(*+AR1(1h)) << GPR(R3) [4:0] → GPR(R5)
SLLV R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) << Mem(*AR2++(IR1))[4:0] → GPR(R5)

SLL_SW

句型： **SLL_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111011						0	1	Src1			Modm		Src2			Modm		Dst			1	Modn				ARm			ARn		

操作： sa = GPR(src1)[4:0]

modm(ARm) << sa → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

SLL_SW R2，*+AR7(IR1)，*AR0--(IR0)，R5，R3

操作： mem(*AR0--(IR0)) << GPR(R5) [4:0]→ GPR(R2)，

GPR(R3) → mem(*+AR7(IR1))

SLT

句型：

SLT	rd，rs，rt	或者
SLT	dst，*+ARm(disp1)，*+ARn(disp2)	或者
SLT	dst，mod(ARm)，rt	或者
SLT	dst，*+ARm(disp)，Imm	或者
SLT	dst，rs，mod(ARm)	或者
SLT	dst，rs，*+ARm(disp)	或者
SLT	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					101010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	101010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		Dst			Modm				1	101010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	101010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		Arm			10		Dst			Modm				1	101010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		Arm			10		Dst			Disp				1	101010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	101010						

操作：

if GPR(Rs) < GPR(rt) then GPR(rd)=1 else GPR(rd)=0 或者

if Mem(*+ARm(disp1)) < mem(*+ARn(disp2)) then GPR(dst) =1 else GPR(dst) 或者

if mod(ARm) < GPR(Rt) then GPR(dst) =1 else GPR(dst)=0 或者

if Mem(*+ARm(disp)) < sign(Imm) then GPR(dst) =1 else GPR(dst)=0 或者

if GPR(Rs) < mod(Arm) then GPR(dst) =1 else GPR(dst)=0 或者

if GPR(Rs) < mem(*+Arm(disp)) then GPR(dst) =1 else GPR(dst)=0 或者

if Modm(ARm) < modn(ARn) then GPR(dst)=1 else GPR(dst)=0

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

SLT 和 SLTU 的区别在于前者产生 overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
SLT R5，R3，R7	If GPR(R3) < GPR(R7) then GPR(R5)=1 else GPR(R5)=0
SLT R5， *+AR1(1h)，*+AR2(8h)	If Mem(*+AR1(1h)) < mem(*+AR2(8h)) then GPR(R5) =1 else GPR(R5)=0
SLT R5，*AR2++(IR1)，R3	If Mem(*AR2++(IR1)) < GPR(R3) then GPR(R5) =1 else GPR(R5)=0
SLT R5， *+AR1(1h)，08h	If Mem(*+AR1(1h)) < sign(08h) then GPR(R5) =1 else GPR(R5)=0
SLT R5，R3，*AR2++(IR1)	If GPR(R3) < Mem(*AR2++(IR1)) then GPR(R5) =1 else GPR(R5)=0
SLT R5，R3，*+AR1(1h)	If GPR(R3) < mem(*+AR1(1h)) then GPR(R5) =1 else GPR(R5)=0
SLT R5，*AR1++(IR0)，*AR2++(IR1)	If Mem(*AR1++(IR0)) < Mem(*AR2++(IR1)) then GPR(R5) =1 else GPR(R5)=0

SLTI

句型： **SLTI** rt，rs，Imm 或者
SLTI dst，@Imm 或者
SLTI dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001010						rs					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001010						11111					00		Dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001010						11111					01		Dst			Modm					ARm			Disp								

操作：

If GPR(Rs) < sign(Imm) then GPR(rt)=1 else GPR(rt)=0 或者
If GPR(dst) < mem(Imm) then GPR(dst)=1 else GPR(dst)=0 或者
If GPR(dst) < modm(ARm) then GPR(dst)=1 else GPR(dst)=0

操作数说明：

rs： 源寄存器 （通用寄存器 0～30）
rt： 目标寄存器 （通用寄存器 0～31）
ARm： 间接寻址 （辅助寄存器 0～7）
Dst： 目的寄存器 （通用寄存器 0～7）
G： 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

SLTI 与 SLTIU 的区别在于前者产生 Overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
SLTI R5，R3，0840h	If GPR(R3) < sign(0840h) then GPR(R5)=1 else GPR(R5)=0
SLTI R5，@0840h	If GPR(R5) < mem(0840h) then GPR(R5)=1 else GPR(R5)=0
SLTI R5，*AR2++(40h)	If GPR(R5) < mem(AR2) then GPR(R5)=1 else GPR(R5)=0 AR2=AR2+40h

SLTIU

句型： **SLTIU** rt，rs，Imm 或者
SLTIU dst，@Imm 或者
SLTIU dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001011						rs					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001011						11111					00		Dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001011						11111					01		Dst			Modm					ARm			Disp								

操作：

If GPR(Rs) < sign(Imm) then GPR(rt)=1 else GPR(rt)=0 或者
If GPR(dst) < mem(Imm) then GPR(dst)=1 else GPR(dst)=0 或者
If GPR(dst) < modm(ARm) then GPR(dst)=1 else GPR(dst)=0

操作数说明：

rs： 源寄存器 （通用寄存器 0～30）
rt： 目标寄存器 （通用寄存器 0～31）
ARm： 间接寻址 （辅助寄存器 0～7）
Dst： 目的寄存器 （通用寄存器 0～7）
G： 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

SLTI 与 SLTIU 的区别在于前者产生 Overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
SLTI R5，R3，0840h	If GPR(R3) < sign(0840h) then GPR(R5)=1 else GPR(R5)=0
SLTI R5，@0840h	If GPR(R5) < mem(0840h) then GPR(R5)=1 else GPR(R5)=0
SLTI R5，*AR2++(40h)	If GPR(R5) < mem(AR2) then GPR(R5)=1 else GPR(R5)=0 AR2=AR2+40h

SLTU

句型：

SLTU	rd，rs，rt	或者
SLTU	dst，*+ARm(disp1)，*+ARn(disp2)	或者
SLTU	dst，mod(ARm)，rt	或者
SLTU	dst，*+ARm(disp)，Imm	或者
SLTU	dst，rs，mod(ARm)	或者
SLTU	dst，rs，*+ARm(disp)	或者
SLTU	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					101011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	101011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		Dst			Modm				1	101011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	101011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		Dst			Modm				1	101011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		Dst			Disp				1	101011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		Dst			Modn				1	101011						

操作：

if GPR(Rs) < GPR(rt) then GPR(dst)=1 else GPR(dst)=0 或者

if Mem(*+ARm(disp1)) < mem(*+ARn(disp2)) then GPR(dst) =1 else GPR(dst) 或者

if mod(ARm) < GPR(Rt) then GPR(dst) =1 else GPR(dst)=0 或者

if Mem(*+ARm(disp)) < sign(Imm) then GPR(dst) =1 else GPR(dst)=0 或者

if GPR(Rs) < mod(ARm) then GPR(dst) =1 else GPR(dst)=0 或者

if GPR(Rs) < mem(*+ARm(disp)) then GPR(dst) =1 else GPR(dst)=0 或者

if Modm(ARm) < modn(ARn) then GPR(dst)=1 else GPR(dst)=0

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

SLT 和 SLTU 的区别在于前者产生 overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
SLTU R5，R3，R7	If GPR(R3) < GPR(R7) then GPR(R5)=1 else GPR(R5)=0
SLTU R5， *+AR1(1h)，*+AR2(8h)	If Mem(*+AR1(1h)) < mem(*+AR2(8h)) then GPR(R5) =1 else GPR(R5)=0
SLTU R5，*AR2++(IR1)，R3	If Mem(*AR2++(IR1)) < GPR(R3) then GPR(R5) =1 else GPR(R5)=0
SLTU R5， *+AR1(1h)，08h	If Mem(*+AR1(1h)) < sign(Imm) then GPR(R5) =1 else GPR(R5)=0
SLTU R5，R3，*AR2++(IR1)	If GPR(R3) < Mem(*AR2++(IR1)) then GPR(R5) =1 else GPR(R5)=0
SLTU R5，R3，*+AR1(1h)	If GPR(R3) < mem(*+AR1(1h)) then GPR(R5) =1 else GPR(R5)=0
SLTU R5，*AR1++(IR0)，*AR2++(IR1)	If Mem(*AR1++(IR0)) < Mem(*AR2++(IR1)) then GPR(R5) =1 else GPR(R5)=0

SRA

句型： **SRA** rd，rt，sa 或者
SRA rd，rt，*+ARm(disp) 或者
SRA rt，@(direct) 或者
SRA rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00000					Rt					Rd					Sa						000011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			Rt					Rd					Disp						000011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						10		Direct1			Rt					Direct2											000011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						11		ARm			Rt					Modm					Disp						000011					

操作：

GPR(rt) >> sa→ GPR(rd) 或者
GPR(rt) >> mem(*+ARm(disp))→ GPR(rd) 或者
GPR(rt) >> mem({Direct1,Direct2})→ GPR(rt) 或者
GPR(rt) >> mod(ARm)→ GPR(rt)

操作数说明：

rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Sa： 立即数
Disp： 立即数
T： 寻址模式选择位。

T	源操作数 1	源操作数 2
00	寄存器	寄存器
01	寄存器	*+ARm(disp)
10	寄存器	直接寻址
11	寄存器	间接寻址

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SRA R3，R7，04h	GPR(R7) >> 04h→ GPR(R3)
SRA R3，R7，*+AR1(04h)	GPR(R7) >> mem(*+AR1(04h))→ GPR(R3)
SRA R3，0840h	GPR(R3) >> mem(0840h)→ GPR(R3)
SRA R3，*AR2++(IR1)	GPR(R3) >> mod(AR2)→ GPR(R3)

SRAV

句型：

SRAV	rd，rt，rs	或者
SRAV	dst，*+ARn(disp2)， *+ARm(disp1)	或者
SRAV	dst，rt，mod(ARm)	或者
SRAV	dst，Imm，*+ARm(disp)	或者
SRAV	dst，mod(ARm)，rs	或者
SRAV	dst，*+ARm(disp)，rs	或者
SRAV	dst，mod(ARn)，mod(ARm)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					Rt					Rd					00000					000111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	000111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			Rt					01		Dst			Modm				1	000111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	000111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					00		ARm			10		Dst			Modm				1	000111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					01		ARm			10		Dst			Disp				1	000111						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	000111						

操作：

GPR(rt) >> GPR(Rs)[4:0] → GPR(rd)	或者
mem(*+ARn(disp2)) >> Mem(*+ARm(disp1))[4:0] → GPR(dst)	或者
GPR(Rt) >> mod(ARm)[4:0] → GPR(dst)	或者
sign(Imm) >> Mem(*+ARm(disp)) → GPR(dst)	或者
mod(ARm) >> GPR(Rs) → GPR(dst)	或者
mem(*+ARm(disp)) >> GPR(Rs) → GPR(dst)	或者
modn(ARn) >> Modm(ARm) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
SRAV R5，R3，R7	GPR(R3) >> GPR(R7) [4:0]→ GPR(R5)
SRAV R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) >> mem(*+AR2(8h))[4:0] → GPR(R5)
SRAV R5，R3，*AR2++(IR1)	Mem(*AR2++(IR1)) >> GPR(R3) [4:0] → GPR(R5)
SRAV R5，08h， *+AR1(1h)	sign(08h) >> Mem(*+AR1(1h)) → GPR(R5)
SRAV R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) >> GPR(R3) [4:0] → GPR(R5)
SRAV R5，*+AR1(1h)，R3	mem(*+AR1(1h)) >> GPR(R3) [4:0] → GPR(R5)
SRAV R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) >> Mem(*AR2++(IR1))[4:0] → GPR(R5)

SRA_SW

句型： **SRA_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111011						0	0	Src1			Modm		Src2			Modm		Dst			1	Modn				ARm			ARn		

操作： sa = GPR(src1)[4:0]

modm(ARm) >> sa → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

SRA_SW R2，*+AR7(IR1)，*AR0--(IR0)，R5，R3

操作： mem(*AR0--(IR0)) >> GPR(R5) [4:0]→ GPR(R2)，

GPR(R3) → mem(*+AR7(IR1))

SRL

句型： **SRL** rd，rt，sa 或者
SRL rd，rt，*+ARm(disp) 或者
SRL rt，@(direct) 或者
SRL rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00000					Rt					Rd					Sa						000010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			Rt					Rd					Disp						000010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						10		Direct1			Rt					Direct2											000010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						11		ARm			Rt					Modm					Disp						000010					

操作：

GPR(rt) >> sa→ GPR(rd) 或者
GPR(rt) >> mem(*+ARm(disp))→ GPR(rd) 或者
GPR(rt) >> mem({Direct1,Direct2})→ GPR(rt) 或者
GPR(rt) >> mod(ARm)→ GPR(rt)

操作数说明：

rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Sa： 立即数
Disp： 立即数
T： 寻址模式选择位。

T	源操作数 1	源操作数 2
00	寄存器	寄存器
01	寄存器	*+ARm(disp)
10	寄存器	直接寻址
11	寄存器	间接寻址

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SRL　R3，R7，04h	GPR(R7) >> 04h→ GPR(R3)
SRL　R3，R7，*+AR1(04h)	GPR(R7) >> mem(*+AR1(04h))→ GPR(R3)
SRL　R3，0840h	GPR(R3) >> mem(0840h)→ GPR(R3)
SRL　R3，*AR2++(IR1)	GPR(R3) >> mod(AR2)→ GPR(R3)

SRLV

句型：

SRLV	rd，rt，rs	或者
SRLV	dst，*+ARn(disp2)， *+ARm(disp1)	或者
SRLV	dst，rt，mod(ARm)	或者
SRLV	dst，Imm，*+ARm(disp)	或者
SRLV	dst，mod(ARm)，rs	或者
SRLV	dst，*+ARm(disp)，rs	或者
SRLV	dst，mod(ARn)，mod(ARm)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					Rt					Rd					00000					000110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	000110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			Rt					01		Dst			Modm				1	000110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	000110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					00		ARm			10		Dst			Modm				1	000110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Rs					01		ARm			10		Dst			Disp				1	000110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	000110						

操作：

GPR(rt) >> GPR(Rs)[4:0] → GPR(rd)	或者
mem(*+ARn(disp2)) >> Mem(*+ARm(disp1))[4:0] → GPR(dst)	或者
GPR(Rt) >> mod(ARm)[4:0] → GPR(dst)	或者
sign(Imm) >> Mem(*+ARm(disp)) → GPR(dst)	或者
mod(ARm) >> GPR(Rs) → GPR(dst)	或者
mem(*+ARm(disp)) >> GPR(Rs) → GPR(dst)	或者
modn(ARn) >> Modm(ARm) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
SRLV R5，R3，R7	GPR(R3) >> GPR(R7) [4:0]→ GPR(R5)
SRLV R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) >> mem(*+AR2(8h))[4:0] → GPR(R5)
SRLV R5，R3，*AR2++(IR1)	Mem(*AR2++(IR1)) >> GPR(R3) [4:0] → GPR(R5)
SRLV R5，08h， *+AR1(1h)	sign(08h) >> Mem(*+AR1(1h)) → GPR(R5)
SRLV R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) >> GPR(R3) [4:0] → GPR(R5)
SRLV R5，*+AR1(1h)，R3	mem(*+AR1(1h)) >> GPR(R3) [4:0] → GPR(R5)
SRLV R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) >> Mem(*AR2++(IR1))[4:0] → GPR(R5)

SRL_SW

句型： **SRL_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111011						0	1	Src1			Modm		Src2			Modm		Dst			0	Modn				ARm			ARn		

操作： sa = GPR(src1)[4:0]

modm(ARm) >> sa → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

SRL_SW R2，*+AR7(IR1)，*AR0--(IR0)，R5，R3

操作： mem(*AR0--(IR0)) >> GPR(R5) [4:0]→ GPR(R0)，

GPR(R3) → mem(*+AR7(IR1))

SUB

句型：

SUB	rd，rs，rt	或者
SUB	dst，*+ARm(disp1)，*+ARn(disp2)	或者
SUB	dst，mod(ARm)，rt	或者
SUB	dst，*+ARm(disp)，Imm	或者
SUB	dst，rs，mod(ARm)	或者
SUB	dst，rs，*+ARm(disp)	或者
SUB	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		dst			Disp2				1	100010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			Rt					01		dst			Modm				1	100010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		dst			Disp				1	100010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		dst			Modm				1	100010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		dst			Disp				1	100010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm		ARn			11		dst			Modn				1	100010						

操作：

GPR(Rs) - GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) - mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) - GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) - sign(Imm) → GPR(dst)	或者
GPR(Rs) - mod(ARm) → GPR(dst)	或者
GPR(Rs) - mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) - modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
SUB R5，R3，R7	GPR(R3) - GPR(R7) → GPR(R5)
SUB R5， *+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) - mem(*+AR2(8h)) → GPR(R5)
SUB R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) - GPR(R3) → GPR(R5)
SUB R5， *+AR1(1h)，08h	Mem(*+AR1(1h)) - sign(08h) → GPR(R5
SUB R5，R3，*AR2++(IR1)	GPR(R3) - Mem(*AR2++(IR1)) → GPR(R5)
SUB R5，R3，*+AR1(1h)	GPR(R3) - mem(*+AR1(1h)) → GPR(R5)
SUB R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) - Mem(*AR2++(IR1)) → GPR(R5)

SUBU

句型：

SUBU	rd，rs，rt	或者
SUBU	dst，*+ARm(disp1)，*+ARn(disp2)	或者
SUBU	dst，mod(ARm)，rt	或者
SUBU	dst，*+ARm(disp)，Imm	或者
SUBU	dst，rs，mod(ARm)	或者
SUBU	dst，rs，*+ARm(disp)	或者
SUBU	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	100011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		Dst			Modm				1	100011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	100011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		Arm			10		Dst			Modm				1	100011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		Arm			10		Dst			Disp				1	100011						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	100011						

操作：

GPR(Rs) - GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) - mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) - GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) - sign(Imm) → GPR(dst)	或者
GPR(Rs) - mod(ARrm) → GPR(dst)	或者
GPR(Rs) - mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) - modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

SUB 和 SUBU 的区别在于前者产生 overflow 异常，而后者不产生任何异常。

执行周期：1 cycle

举例：

句型	操作
SUBU R5，R3，R7	GPR(R3) - GPR(R7) → GPR(R5)
SUBU R5， *+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) - mem(*+AR2(8h)) → GPR(R5)
SUBU R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) - GPR(R3) → GPR(R5)
SUBU R5， *+AR1(1h)，08h	Mem(*+AR1(1h)) – sign08h) → GPR(R5)
SUBU R5，R3，*AR2++(IR1)	GPR(R3) - Mem(*AR2++(IR1)) → GPR(R5)
SUBU R5，R3，*+AR1(1h)	GPR(R3) - mem(*+AR1(1h)) → GPR(rd)
SUBU R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) - Mem(*AR2++(IR1)) → GPR(R5)

SUB_SW

句型： **SUB_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110011						0	0	Src1			Modm		Src2			Modm		Dst			1	Modn				ARm			ARn		

操作： modm(ARm) - GPR(src1) → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）
Src2： 寄存器 （通用寄存器 0～7）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod (4bit)	偏移地址的计算	Mod (4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

SUB_SW R2，*+AR7(IR1)，R5，*AR0--(IR0)，R3
操作： mem(*AR0--(IR0)) - GPR(R5) → GPR(R0)，
GPR(R3) → mem(*+AR7(IR1))

SW

句型： **SW** rt, offset(base) 或者
SW rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101011						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101011						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，
GPR(rt) → mem(Vaddr)

操作数说明： base： 寄存器 （通用寄存器 0～30）
rt： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Disp： 立即数
offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SW R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，GPR(R5) → mem(Vaddr)
SW R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，GPR(R5) → mem(Vaddr)

SWL

句型： **SWL** rt, offset(base) 或者
SWL rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101010						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101010						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，
Left(GPR(rt)) → mem(Vaddr)

操作数说明： base： 寄存器 （通用寄存器 0～30）
rt： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Disp： 立即数
offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SWL R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，left(GPR(R5)) → mem(Vaddr)
SWL R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，left(GPR(R5)) → mem(Vaddr)

SWR

句型： **SWR** rt, offset(base) 或者
SWR rt，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101110						base					rt					offset																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
101110						11111					rt					Modm					ARm			Disp							

操作：

Vaddr = sign(offset) + GPR(base)， 或者 Vaddr = Mod(ARm)，
Right(GPR(rt)) → mem(Vaddr)

操作数说明： base： 寄存器 （通用寄存器 0～30）
rt： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
Disp： 立即数
offset： 立即数

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
SWR R5，0840h(R3)	Vaddr = sign(0840h) + GPR(R3)，Right(GPR(R5)) → mem(Vaddr)
SWR R5，*AR0--(IR0)	Vaddr = *AR0--(IR0)，Right(GPR(R5)) → mem(Vaddr)

SW_SW

句型： **SW_SW** mod(ARm)，mod(ARn)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111011						1	0	Src1			Modm		Src2			Modm		000			1	Modn				ARm			ARn		

操作： GPR(src1) → modm(ARm) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

SW_SW *AR0--(IR0)，*+AR7(IR1)，R5，R3

操作： GPR(R5) → mem(*AR0--(IR0))，

GPR(R3) → mem(*+AR7(IR1))

SYSCALL

句型：SYSCALL

指令编码：

31 26	25 6	5 0
000000	0000_0000_0000_0000_0000	001100

操作：SystmeCallException

操作数说明：

描述：系统异常。

执行周期：1 cycle

TLBR

句型：TLBR

指令编码：

31 26	25	24 6	5 0
010000	**1**	**000_0000_0000_0000_0000_0000**	**00_0001**

操作： TLB[Index]$_{127..96}$ → PageMask；
TLB[Index] $_{95..64}$ → EntryHi ；
TLB[Index] $_{63..32}$→ EntryLo1；
TLB[Index] $_{31..0}$ → EntryLo0

操作数说明：

EntryLo0: 表项 Lo0 寄存器
EntryLo1: 表项 Lo1 寄存器
PageMask: Mask 寄存器
Index:索引寄存器

描述：TLB 变址寄存器所指内容装入 EntryHi，EntryLo0, EntryLo1 和 MASK 寄存器。

执行周期：1 cycle

TLBWI

句型：TLBWI

指令编码：

31 26	25	24 6	5 0
010000	**1**	**000_0000_0000_0000_0000_0000**	**00_0010**

操作：PageMask || (EntryHi and not PageMask) || EntryLo1 || EntryLo0) → TLB[Index]

操作数说明：

EntryLo0:表项 Lo0 寄存器

EntryLo1: 表项 Lo1 寄存器

PageMask: Mask 寄存器

Index:索引寄存器

描述：TLB 变址寄存器所指内容装入 EntryHi，EntryLo0, EntryLo1 和 MASK 寄存器。

执行周期：1 cycle

XOR

句型：

XOR	rd，rs，rt	或者
XOR	dst，*+ARm(disp1)，*+ARn(disp2)	或者
XOR	dst，mod(ARm)，rt	或者
XOR	dst，*+ARm(disp)，Imm	或者
XOR	dst，rs，mod(ARm)	或者
XOR	dst，rs，*+ARm(disp)	或者
XOR	dst，mod(ARm)，mod(ARn)	

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					rt					rd					00000					100110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Disp1		ARm			Disp1		ARn			00		Dst			Disp2				1	100110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						00		ARm			rt					01		Dst			Modm				1	100110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						01		ARm			imm					01		Dst			Disp				1	100110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					00		ARm			10		Dst			Modm				1	100110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						rs					01		ARm			10		Dst			Disp				1	100110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000000						Modm		ARm			Modm			ARn		11		Dst			Modn				1	100110						

操作：

GPR(Rs) xor GPR(rt) → GPR(rd)	或者
Mem(*+ARm(disp1)) xor mem(*+ARn(disp2)) → GPR(dst)	或者
mod(ARm) xor GPR(Rt) → GPR(dst)	或者
Mem(*+ARm(disp)) xor sign(Imm) → GPR(dst)	或者
GPR(Rs) xor mod(ARm) → GPR(dst)	或者
GPR(Rs) xor mem(*+ARm(disp)) → GPR(dst)	或者
Modm(ARm) xor modn(ARn) → GPR(dst)	

操作数说明：

rs： 寄存器 （通用寄存器 0～31）
rt： 寄存器 （通用寄存器 0～31）
rd： 寄存器 （通用寄存器 0～31）
ARm： 间接寻址（辅助寄存器 0～7）
ARn： 间接寻址（辅助寄存器 0～7）
Dst： 寄存器 （通用寄存器 0～7）
T： 寻址模式选择位。

T		源操作数 1	源操作数 2
00		*+ARn(disp)寻址	*+ARn(disp)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(disp)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(disp)寻址
11		间接寻址	间接寻址

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

句型	操作
XOR R5，R3，R7	GPR(R3) xor GPR(R7) → GPR(R5)
XOR R5，*+AR1(1h)，*+AR2(8h)	Mem(*+AR1(1h)) xor mem(*+AR2(8h)) → GPR(R5)
XOR R5，*AR2++(IR1)，R3	Mem(*AR2++(IR1)) xor GPR(R3) → GPR(R5)
XOR R5，*+AR1(1h)，08h	Mem(*+AR1(1h)) xor sign(08h) → GPR(R5)
XOR R5，R3，*AR2++(IR1)	GPR(R3) xor Mem(*AR2++(IR1)) → GPR(R5)
XOR R5，R3，*+AR1(1h)	GPR(R3) xor mem(*+AR1(1h)) → GPR(R5)
XOR R5，*AR1++(IR0)，*AR2++(IR1)	Mem(*AR1++(IR0)) xor Mem(*AR2++(IR1)) → GPR(R5)

XORI

句型： **XORI** rt，rs，Imm 或者
XORI dst，@Imm 或者
XORI dst，mod(ARm)

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001110						rs					rt					Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001110						11111					00		dst			Imm																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
001110						11111					01		dst			Modm					ARm			Disp								

操作：

GPR(Rs) xor zero(Imm) → GPR(rt) 或者
GPR(dst) xor mem(Imm) → GPR(dst) 或者
GPR(dst) xor mod(ARm) → GPR(dst)

操作数说明：

rs: 源寄存器 （通用寄存器 0～30）
rt: 目标寄存器 （通用寄存器 0～31）
ARm: 间接寻址 （辅助寄存器 0～7）
Dst: 目的寄存器 （通用寄存器 0～7）
G: 寻址模式选择位。G=00 为直接寻址，G=01 为间接寻址。

描述：

Mod(5bit)	偏移地址的计算	Mod(5bit)	偏移地址的计算
10000	*+ARn(disp)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(disp)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(disp)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(disp)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(disp)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(disp)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(disp)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(disp)%	00111	*ARn--(IR0)%
		01111	*ARn--(IR1)%
11001	*ARn++(IR0)!	11000	*ARn

执行周期：1 cycle

举例：

句型	操作
XORI R5，R3，0840h	GPR(R3) xor zero(0840h) → GPR(R5)
XORI R5，@0840h	GPR(R5) xor mem(0840h) → GPR(R5)
XORI R5，*AR2++(40h)	GPR(R5) xor mem(AR2) → GPR(R5)，AR2=AR2+40h

XOR_SW

句型： **XOR_SW** dst，mod(ARn)，mod(ARm)，src1，src2

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
110011						1	0	Src1			Modm		Src2			Modm		Dst			0	Modn				ARm			ARn		

操作： modm(ARm) XOR GPR(src1) → GPR(dst) || GPR(src2) → modn(ARn)

操作数说明：

src1： 寄存器 （通用寄存器 0～7）

Src2： 寄存器 （通用寄存器 0～7）

ARm： 间接寻址（辅助寄存器 0～7）

ARn： 间接寻址（辅助寄存器 0～7）

Dst： 寄存器 （通用寄存器 0～7）

描述：

Mod(4bit)	偏移地址的计算	Mod(4bit)	偏移地址的计算
0000	*+ARn(IR0)	1000	*+ARn(IR1)
0001	*-ARn(IR0)	1001	*-ARn(IR1)
0010	*++ARn(IR0)	1010	*++ARn(IR1)
0011	*--ARn(IR0)	1011	*--ARn(IR1)
0100	*ARn++(IR0)	1100	*ARn++(IR1)
0101	*ARn--(IR0)	1101	*ARn--(IR1)
0110	*ARn++(IR0)%	1110	*ARn++(IR1)%
0111	*ARn--(IR0)%	1111	*ARn--(IR1)%

执行周期：1 cycle

举例：

XOR_SW R2，*+AR7(IR1)，*AR0--(IR0)，R5，R3

操作： mem(*AR0--(IR0)) xor GPR(R5) → GPR(R2)，
GPR(R3) → mem(*+AR7(IR1))

NOTES

- 2003, May,1 更新一些错别字错误。更改了 MDD 指令 LW_LW 的编码，增加了 MDS 指令 PMTLO, PMTHI, PMFLO, PMFHI, 删去了 PSTOREQ, PLOADQ 指令。相对于 Ver1.01.(by Peng LIU)
- 2004, Jan.18 按照字顺排列指令，同时按照 TSMC 流片的版本进行的修正。(by Peng LIU).MediaDSP3201 不支持 MDS 类型 4（以斜体表示）。
- 2004, Feb,9 按照视频系统芯片的设计实现更新指令集，以便第二次流片（MediaDSP3202 版本）.根据李东晓的文档。（见附录 1）
- 2004, Feb.12 经过李东晓的反馈，进行了更新。Ver1.3
- 2004,Feb.19 经过赖莉雅的反馈，进行了更新。Ver1.4
- 2004,Feb,19RISC3200 的指令集请参考 MDF、MDS 指令。

三、MDS 指令集修改方案（MediaDSP3202 中实现）

根据对 IDCT 算法及其在 MD32 上软件实现的深入研究，提出了一些对 MDS 指令集的修改意见，可以大大提高处理性能，经过讨论，整理为如下的修改方案。

1 运算类指令由 2 操作数改成 3 操作数

目前，MDS 指令共有 36 条（不同粒度相同功能归为 1 条），按照功能可分为 6 组：

- 数据传输指令（6 条）；
- 数据转换指令（打包 2 条，解包 2 条）；
- 算术指令（17 条）；
- 比较指令（2 条）；
- 逻辑指令（4 条）；
- 移位指令（3 条）。

后面 5 组指令为*运算类指令*。

原来的 MDS 指令设计为 2 操作数指令，汇编助记符格式为

INST MRd，SRC

其中，INST 为指令助记符；MRd 为 MDS 寄存器，既是目的操作数又是第一个源操作数；SRC 为第二个源操作数，有 4 种可能的来源（立即数、MDS 寄存器、GPR 寄存器和 memory）。例外的是 3 条数据传输指令（PMTHI，PMTLO 和 PSTOREO），MRd 为数据源，而 SRC 为目的地。

相应的 MDS 指令的编码格式可归为 4 种格式，如图 1 所示。

MDF R-type （与 MIPS R4000 之 R-type 一致）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
op_code						Rs					Rt					Rd					sa					func_code					

MDS-type1：DEST 操作数来自 MDS 寄存器 MRt，SRC 操作数来自立即数 sa

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		000			gg		MRd			00000					sa					func_code					

MDS-type2：DEST 操作数来自 MDS 寄存器 MRt，SRC 操作数来自 MDS 寄存器 MRs

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			gg		MRd			00000					00000					func_code					

MDS-type3：DEST 操作数来自 MDS 寄存器 MRt，SRC 操作数来自通用寄存器 Rs

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			gg		MRd			Rs					00000					func_code					

MDS-type4：DEST 操作数来自 MDS 寄存器 MRt，SRC 操作数来自 memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		ARm			gg		MRd			Modm					disp					func_code					

图 1 原 MDS 指令的 4 种编码格式

经过研究，决定将 MDS 运算类指令修改为 3 操作数指令，汇编助记符格式为

INST MRd，MRs，SRC

其中，INST 为指令助记符；MRd 为 MDS 寄存器，为目的操作数；MRs 为 MDS 寄存器，为

第一个源操作数；SRC 为第二个源操作数，有 4 种可能的来源（立即数、MDS 寄存器、GPR 寄存器和 memory）。实质上就是，原来的目的操作数与第一个源操作数为同一个 MDS 寄存器，而现在将它们分离，目的操作数可以是不同的 MDS 寄存器。当然，在软件编程时，允许 MRd、MRs 取为同一个 MDS 寄存器地址，甚至 3 个操作数全取为同一个 MDS 寄存器地址，硬件上也不会出现问题。因此，3 操作数的扩展肯定包容了原 2 操作数指令的所有功能。

相应地，MDS 指令的编码格式调整为如图 2 所示。4 种编码格式以 Instr[25:24]来区分。

MDF R-type （与 MIPS R4000 之 R-type 一致）

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
op_code						Rs					Rt					Rd					sa					func_code						

MDS-type1：目标操作数为 MRd，第一源操作数为 MRs，第二源操作数为立即数 sa

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					func_code					

MDS-type2：目标操作数为 MRd，第一源操作数为 MRs，第二源操作数为立即数 MRt

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					func_code					

MDS-type3：用于 MDS 寄存器 MRd 和通用寄存器 Rs 之间的数据传输

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		MRs			Rs					gg		MRd			00000					func_code					

MDS-type4：目标操作数为 MRd，第一源操作数为 MRs，第二源操作数来自 memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					func_code					

图 2 3 操作数格式的 MDS 指令编码格式

1. MDS-type1 只用于*移位指令*，汇编格式为

INST MRd，MRs，sa

MRd 为目的 MDS 寄存器，MRs 为源 MDS 寄存器。Sa 为立即数，10bit 编码，对于移位指令只有低 5bit 有效，高 5bit 填 0。

2. MDS-type2 适用于*所有的运算类指令*，汇编格式为

INST MRd，MRs，MRt

MRd 为目的 MDS 寄存器，MRs 为第一源 MDS 寄存器，MRt 为第二源 MDS 寄存器。

3. MDS-type3 只用于 MDS 寄存器和通用寄存器之间的*数据传输*，汇编格式为

PMTHI / PMTLO MRd，Rs

PMFHI / PMFLO MRs，Rs

对于 PMTHI、PMTLO 指令，不用的 MRs 数据场填 0；

对于 PMFHI、PMFLO 指令，不用的 MRd 数据场填 0。

4. MDS-type4 适用于*除移位外所有的运算类指令*，汇编格式为

INST MRd，MRs，mem

MRd 为目的 MDS 寄存器，MRs 为第一源 MDS 寄存器，mem 为第二源操作数。

第二源操作数 mem 来自 memory，寻址模式用 5-bit Modm 编码；地址计算辅助寄存器用 3-bit 编码。*偏移立即数用 2-bit 编码，因为 MDS 指令使用的 memory 数据肯定为 64bit 宽度，在 memory 寻址时地址也必须指向 64bit 边界，即地址的最低 3bit 默认为 0，disp 编码的是第 4 第 5bit*。mem 寻址方式与 MDD 指令一样，如下表 1。

表 1 MDS 指令中存储器操作数的寻址方式

MODM（5BIT）	地址计算 / 辅助地址寄存器自修改	MODM（5BIT）	地址计算 / 辅助地址寄存器自修改
00000	*+ARm（disp X 8）	01000	*+ARm（IR0）
		10000	*+ARm（IR1）
00001	*-ARm（disp X 8）	01001	*-ARm（IR0）
		10001	*-ARm（IR1）
00010	*++ARm（disp X 8）	01010	*++ARm（IR0）
		10010	*++ARm（IR1）
00011	*--ARm（disp X 8）	01011	*--ARm（IR0）
		10011	*--ARm（IR1）
00100	*ARm++（disp X 8）	01100	*ARm++（IR0）
		10100	*ARm++（IR1）
00101	*ARm--（disp X 8）	01101	*ARm--（IR0）
		10101	*ARm--（IR1）
00110	*ARm++（dispX8）%	01110	*ARm++（IR0）%
		10110	*ARm++（IR1）%
00111	*ARm--（disp X 8）%	01111	*ARm--（IR0）%
		10111	*ARm--（IR1）%
11001	*ARm++（IR0X8）B	11000	*ARm

5. 对于 MDS 寄存器和 memory 之间的数据传输指令，只需要 2 个操作数，汇编格式为

PLOADO MRd，mem

和 PSTOREO MRs，mem

为了编码的规整性和译码的方便性，还是采用 MDS-type4 编码格式。对于 PLOADO，MRs 数据场不用，置为全 0；对于 PSTOREO，MRd 数据场不用，置为全 0。

2 新增 16 比特 MAC 指令 4 条

新增指令的功能码编码如下表斜体所示：PMACLSD、PMACLUD、PMACHSD 和 PMACHUD。在软件编程时，应该与 PMUL*指令相应配合使用，比如，要做乘累加时，第一条发送 PMULLSD 指令，乘法结果作为累加的初值，然后*连续发送* PMACLSD 指令，将本条指令的乘法结果与累加值相加成新的累加结果。具体地说，在软件编程时，连续 PMACLSD 指令序列应前缀 PMULLSD 指令；连续 PMACHSD 指令序列应前缀 PMULHSD 指令；连续 PMACLUD 指令序列应前缀 PMULLUD 指令；连续 PMACHUD 指令序列应前缀 PMULHUD 指令。

汇编器、指令集仿真器、反汇编模块上增加对新指令的支持。

表 2 MDS 指令功能码表

Instr[2:0] / Instr[5:3]	0	1	2	3	4	5	6	7
0	PSLL D/Q/O	*PSHUFD*	PSRL D/Q/O	PSRA D/Q				
1			PUNPCKH BD/DQ/QO	PUNPCKL BD/DQ/QO				
2	PMFHI	PMTHI	PMFLO	PMTLO	PACKSS DB/QD	PACKUS DB/QD		
3	PMULLSD	PMULLUD	*PMACLSD*	*PMACLUD*	PMULHSD	PMULHUD	*PMACHSD*	*PMACHUD*
4	PADDS B/D	PADDUS B/D	PSUBS B/D	PSUBUS B/D	PAND	POR	PXOR	PNOR
5	PMADDQD	PSADBD			PADD B/D/Q		PSUB B/D/Q	
6	PCMPGT B/D/Q				PCMPEQ B/D/Q			PLOADO
7	PMAXSD	PMAXUB	PMINSD	PMINUB	PAVG B/D			PSTOREO

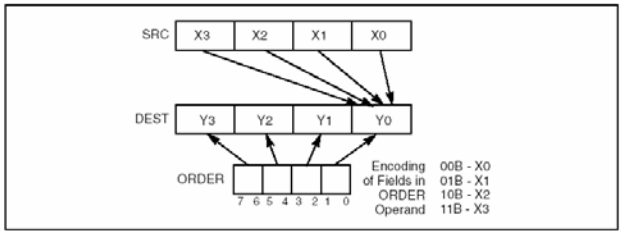
3 增加其它指令

增加数据的换位指令，以及适用于矩阵转角的专用指令等。经过讨论，先增加一条数据换位指令，汇编格式为：

PSHUFD MRd，MRs，sa

PSHUFD MRd，MRs，MRt

功能码编码见表 2，与移位指令构成一组。PSHUFD 指令的功能是，MRs 为源操作数，MRd 为目标操作数，均为 4 个 16bit 数据打包格式，MRd 中每个 16bit 数据取自 MRs 中 4 个 16bit 数据的其中一个。这样，每个结果 16bit 数据的产生需要 2bit 选择信号，总共需要 8bit 选择信号，由立即数 sa 的最低 8 个 bit 表示（高 2bit 汇编时置 0）或 MRt 操作数的最低 8 个 bit 表示。



4 MDS 指令简表（MediaDSP3202）

指令分组	MD-32 MDS 媒体指令	类似功能的 INTEL 指令	指令功能简要描述
数据传输指令	PMTHI，PMTLO *MRd，Rs*	MOVD PINSRW SSE	传输 4-字节（*Rs* 到 *MRd*） 从通用寄存器到 MDS 寄存器高/低端
	PMFHI，PMFLO *MRd，Rs*	MOVD PEXTRW SSE	传输 4-字节（*MRd* 到 *Rs*） 从 MDS 寄存器高/低端到通用寄存器
	PLOADO *MRd，mem*	MOVD PINSRW SSE	传输 8-字节（*mem* 到 *MRd*） 从 memory 到 MDS 寄存器
	PSTOREO *MRd，mem*	MOVD PEXTRW SSE	传输 8-字节（*MRd* 到 *mem*） 从 MDS 寄存器到 memory
数据转换指令	PACKSSDB/QD *MRd，MRs，MRt* *MRd，MRs，mem*	PACKSSWB PACKSSDW	将*MRt*（*mem*）和*MRs*操作数中打包的2-字节/4-字节数据转换为1-字节/2-字节数据，使用有符号饱和处理溢出
	PACKUSDB *MRd，MRs，MRt* *MRd，MRs，mem*	PACKUSWB	将 *MRt*（*mem*）和 *MRs* 操作数中打包的 2-字节数据转换为 1-字节数据，使用无符号饱和处理溢出
	PUNPCKHBD/DQ/QO *MRd，MRs，MRt* *MRd，MRs，mem*	PUNPCKH BW/WD/DQ	将 *MRt*（*mem*）和 *MRs* 操作数中打包的 1-字节/2-字节/4-字节相交织，取高 64-bit 存入 *MRd* 操作数
	PUNPCKLBD/DQ/QO *MRd，MRs，MRt* *MRd，MRs，mem*	PUNPCKL BW/WD/DQ	将 *MRt*（*mem*）和 *MRs* 操作数中打包的 1-字节/2-字节/4-字节相交织，取低 64-bit 存入 *MRd* 操作数
算术指令	PADDB/D/Q *MRd，MRs，MRt* *MRd，MRs，mem*	PADDB/W/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节/4-字节数据执行 SIMD 加法，不作溢出处理
	PADDSB/D *MRd，MRs，MRt* *MRd，MRs，mem*	PADDSB/W	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 加法，使用有符号饱和处理溢出
	PADDUSB/D *MRd，MRs，MRt* *MRd，MRs，mem*	PADDUSB/W	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 加法，使用无符号饱和处理溢出
	PSUBB/D/Q *MRd，MRs，MRt* *MRd，MRs，mem*	PSUBB/W/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节/4-字节数据执行 SIMD 减法，不作溢出处理
	PSUBSB/D *MRd，MRs，MRt* *MRd，MRs，mem*	PSUBSB/W	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 减法，使用有符号饱和处理溢出
	PSUBUSB/D *MRd，MRs，MRt* *MRd，MRs，mem*	PSUBUSB/W	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 减法，使用无符号饱和处理溢出

指令分组	MD-32 MDS 媒体指令	类似功能的 INTEL 指令	指令功能简要描述
	PMULLSD PMACLSD *MRd，MRs，MRt* *MRd，MRs，mem*	PMULLW 无	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节数据执行 SIMD 有符号乘法，每个乘法结果取低 16-bit。PMACLSD 将每次乘法结果不断累加。
	PMULHSD PMACHSD *MRd，MRs，MRt* *MRd，MRs，mem*	PMULHW 无	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节数据执行 SIMD 有符号乘法，每个乘法结果取高 16-bit。PMACHSD 将每次乘法结果不断累加。
	PMULLUD PMACLUD *MRd，MRs，MRt* *MRd，MRs，mem*	PMULHUW SSE 无	*MRs*和*MRt*（*mem*）操作数中打包的2-字节数据执行SIMD无符号乘法，每个乘法结果取低16-bit。PMACLUD将每次乘法结果不断累加。
	PMULHUD PMACHUD *MRd，MRs，MRt* *MRd，MRs，mem*	PMULHUW SSE 无	*MRs*和*MRt*（*mem*）操作数中打包的2-字节数据执行SIMD无符号乘法，每个乘法结果取高16-bit。PMACHUD将每次乘法结果不断累加。
	PMADDQD *MRd，MRs，MRt* *MRd，MRs，mem*	PMADDWD	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节数据执行 SIMD 有符号乘法，相邻 2 个结果两两相加
	PAVGB/D *MRd，MRs，MRt* *MRd，MRs，mem*	PAVGB/W SSE	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节数据执行 SIMD 平均值计算，小数四舍五入
	PMAXUB *MRd，MRs，MRt* *MRd，MRs，mem*	PMAXUB SSE	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节无符号数执行 SIMD 比较，结果取较大的数
	PMAXSD *MRd，MRs，MRt* *MRd，MRs，mem*	PMAXSW SSE	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节有符号数执行 SIMD 比较，结果取较大的数
	PMINUB *MRd，MRs，MRt* *MRd，MRs，mem*	PMINUB SSE	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节无符号数执行 SIMD 比较，结果取较小的数
	PMINSD *MRd，MRs，MRt* *MRd，MRs，mem*	PMINSW SSE	*MRs* 和 *MRt*（*mem*）操作数中打包的 2-字节有符号数执行 SIMD 比较，结果取较小的数
	PSADBD *MRd，MRs，MRt* *MRd，MRs，mem*	PSADBW SSE	*MRs* 和 *MRt*（*mem*）中打包的 1-字节无符号数执行 SIMD 减法，减法结果取绝对值，最后 8 个绝对差值相加
比较指令	PCMPEQB/D/Q *MRd，MRs，MRt* *MRd，MRs，mem*	PCMPEQ B/W/D	*MRs* 和 *MRt*（*mem*）操作数中打包的 1-字节/2-字节/4-字节数据执行 SIMD 比较，若相等则结果为全 1，否则为全 0
	PCMPGTB/D/Q *MRd，MRs，MRt* *MRd，MRs，mem*	PCMPGT B/W/D	*MRs* 和 *MRt*（*mem*）中打包的 1-字节/2-字节/4-字节有符号数执行 SIMD 比较，若大于则结果为全 1，否则为全 0
	PAND *MRd，MRs，MRt* *MRd，MRs，mem*	PAND	*MRs* 和 *MRt*（*mem*）按位逻辑与

指令分组	MD-32 MDS 媒体指令	类似功能的 INTEL 指令	指令功能简要描述
逻辑指令	POR *MRd，MRs，MRt* *MRd，MRs，mem*	POR	*MRs* 和 *MRt*（*mem*）按位逻辑或
	PXOR *MRd，MRs，MRt* *MRd，MRs，mem*	PXOR	*MRs* 和 *MRt*（*mem*）按位逻辑异或
	PNOR *MRd，MRs，MRt* *MRd，MRs，mem*	无	*MRs* 和 *MRt*（*mem*）按位逻辑或非
移位换位指令	PSLLD/Q *MRd，MRs，MRt* *MRd，MRs，imm*	PSLLW/D/Q	*MRs* 中打包的 2 字节/4 字节数据进行 SIMD 逻辑左移，移位量来自 *MRt*（*imm*）的最低 5bit
	PSRLD/Q *MRd，MRs，MRt* *MRd，MRs，imm*	PSRLW/D/Q	*MRs* 中打包的 2 字节/4 字节数据进行 SIMD 逻辑右移，移位量来自 *MRt*（*imm*）的最低 5bit
	PSRAD *MRd，MRs，MRt* *MRd，MRs，imm*	PSRAW/D	*MRs* 中打包的 2 字节数据进行 SIMD 算术右移，移位量来自 *MRt*（*imm*）的最低 5bit
	PSHUFD *MRd，MRs，MRt* *MRd，MRs，imm*	PSHUFD	*MRs* 中打包的 2 字节数据作换位排列，换位控制来自 *MRt*（*imm*）的最低 8bit

PACKSSDB/QD

句型： **PACKSSDB/QD** *MRd，MRs，MRt*

PACKSSDB/QD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					010100					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					010100					

操作：

PACKSSDB:
MRd[7..0] ←SaturateSignedDouble-byteToSignedByte MRs[15..0];
MRd[15..8] ←SaturateSignedDouble-byteToSignedByte MRs[31..16];
MRd[23..16] ←SaturateSignedDouble-byteToSignedByte MRs[47..32];
MRd[31..24] ←SaturateSignedDouble-byteToSignedByte MRs[63..48];
MRd[39..32] ←SaturateSignedDouble-byteToSignedByte MRt[15..0];
MRd[47..40] ←SaturateSignedDouble-byteToSignedByte MRt[31..16];
MRd[55..48] ←SaturateSignedDouble-byteToSignedByte MRt[47..32];
MRd[63..56] ←SaturateSignedDouble-byteToSignedByte MRt[63..48];

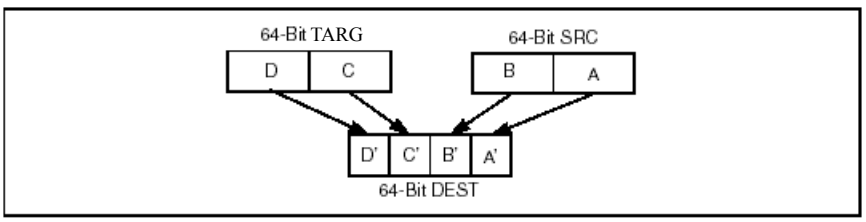
PACKSSQD:
MRd[15..0] ←SaturateSignedQuad-byteToSignedDouble-byte MRs[31..0];
MRd[31..16] ←SaturateSignedQuad-byteToSignedDouble-byte MRs[63..32];
MRd[47..32] ←SaturateSignedQuad-byteToSignedDouble-byte MRt[31..0];
MRd[63..48] ←SaturateSignedQuad-byteToSignedDouble-byte MRt[63..32];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd: MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PACKSSDB 将 64-bit MRs 操作数中打包的 4 个有符号 2 字节数和 64-bit MRt 操作数中打包的 4 个有符号 2 字节数转换为 8 个有符号字节数，采用有符号饱和法处理溢出，结果存入 MRd 操作数。

PACKSSQD 将 64-bit MRs 操作数中打包的 2 个有符号 4 字节数和 64-bit MRt 操作数中打包的 2 个有符号 4 字节数转换为 4 个有符号 2 字节数，采用有符号饱和法处理溢出，结果存入 MRd 操作数。

下图示例了 PACKSSQD 的操作过程，其它类推。



执行周期： **1** cycle

PACKUSDB/QD

句型： **PACKUSDB/QD** *MRd，MRs，MRt*

PACKUSDB/QD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					010101					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					010101					

操作：

PACKUSDB:
MRd[7..0] ←SaturateSignedDouble-byteToUnsignedByte MRs[15..0];
MRd[15..8] ←SaturateSignedDouble-byteToUnsignedByte MRs[31..16];
MRd[23..16] ←SaturateSignedDouble-byteToUnsignedByte MRs[47..32];
MRd[31..24] ←SaturateSignedDouble-byteToUnsignedByte MRs[63..48];
MRd[39..32] ←SaturateSignedDouble-byteToUnsignedByte MRt[15..0];
MRd[47..40] ←SaturateSignedDouble-byteToUnsignedByte MRt[31..16];
MRd[55..48] ←SaturateSignedDouble-byteToUnsignedByte MRt[47..32];
MRd[63..56] ←SaturateSignedDouble-byteToUnsignedByte MRt[63..48];

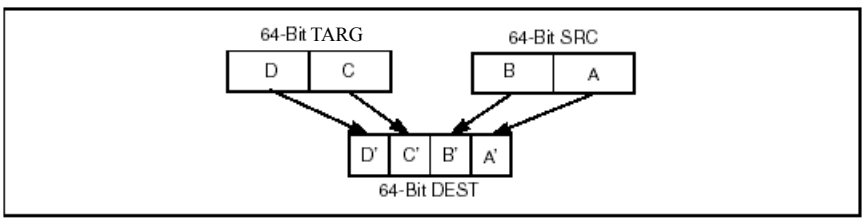
PACKUSQD:
MRd[15..0] ←SaturateSignedQuad-byteToUnsignedDouble-byte MRs[31..0];
MRd[31..16] ←SaturateSignedQuad-byteToUnsignedDouble-byte MRs[63..32];
MRd[47..32] ←SaturateSignedQuad-byteToUnsignedDouble-byte MRt[31..0];
MRd[63..48] ←SaturateSignedQuad-byteToUnsignedDouble-byte MRt[63..32];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PACKUSDB 将 64-bit MRs 操作数中打包的 4 个有符号 2 字节数和 64-bit MRt 操作数中打包的 4 个有符号 2 字节数转换为 8 个有符号字节数，采用无符号饱和法处理溢出，结果存入 MRd 操作数。

PACKUSQD 将 64-bit MRs 操作数中打包的 2 个有符号 4 字节数和 64-bit MRt 操作数中打包的 2 个有符号 4 字节数转换为 4 个有符号 2 字节数，采用无符号饱和法处理溢出，结果存入 MRd 操作数。

下图示例了 PACKUSQD 的操作过程，其它类推。



执行周期：1 cycle

PADDB/D/Q

句型： **PADDB/D/Q** *MRd，MRs，MRt*

PADDB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					101100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					101100						

操作：

PADDB instruction with 64-bit operands:

MRd[7..0] ← MRs[7..0] + MRt[7..0];

* repeat add operation for 2nd through 7th byte *;

MRd[63..56] ← MRs[63..56] + MRt[63..56];

PADDD instruction with 64-bit operands:

MRd[15..0] ← MRs[15..0] + MRt[15..0];

* repeat add operation for 2nd and 3th double-byte *;

MRd[63..48] ← MRs[63..48] + MRt[63..48];

PADDQ instruction with 64-bit operands:

MRd[31..0] ← MRs[31..0] + MRt[31..0];

MRd[63..32] ← MRs[63..32] + MRt[63..32];

操作数说明： MRs： MDS 寄存器

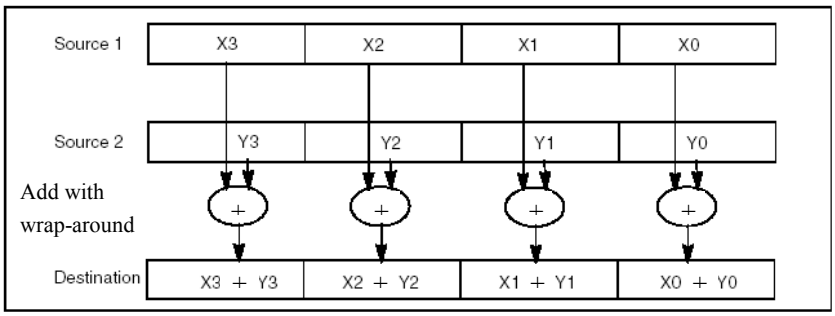
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PADDB/D/Q 对 64－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 64－bit MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 加法，结果存入 MRd 操作数中，溢出被忽略。下图示例 PADDD 的操作过程，其它类推。



执行周期：1 cycle

PADDSB/D

句型： **PADDSB/D** *MRd，MRs，MRt*

PADDSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					100000						

操作：

PADDSB instruction with 64-bit operands:

MRd[7..0] ← SaturateToSignedByte(MRs[7..0] + MRt (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

MRd[63..56] ← SaturateToSignedByte(MRs[63..56] + MRt[63..56]);

PADDSD instruction with 64-bit operands:

MRd[15..0] ← SaturateToSignedDouble-byte(MRs[15..0] + MRt[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

MRd[63..48] ← SaturateToSignedDouble-byte(MRs[63..48] + MRt[63..48]);

操作数说明： MRs： MDS 寄存器

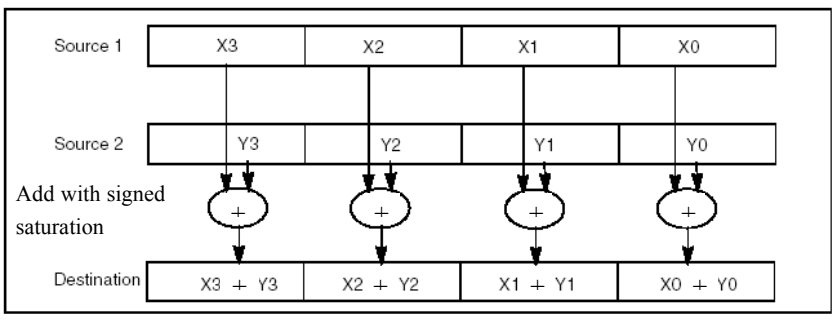
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PADDSB 对 64－bit MRs 操作数中 8 个打包的字节数和 64－bit MRt 操作数中 8 个打包的字节数，执行 SIMD 有符号加法，结果存入 MRd 操作数中相应的位置。PADDSD 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号加法，结果存入 MRd 操作数中相应的位置。下图示例 PADDSD 的操作过程，PADDSB 类推。



执行周期：1 cycle

PADDUSB/D

句型： **PADDUSB/D** *MRd，MRs，MRt*

PADDUSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					100001						

操作：

PADDUSB instruction with 64-bit operands:

MRd[7..0] ← SaturateToUnsignedByte(MRs[7..0] + MRt (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

MRd[63..56] ← SaturateToUnsignedByte(MRs[63..56] + MRt[63..56]);

PADDUSD instruction with 64-bit operands:

MRd[15..0] ← SaturateToUnsignedDouble-byte(MRs[15..0] + MRt[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

MRd[63..48] ← SaturateToUnsignedDouble-byte(MRs[63..48] + MRt[63..48]);

操作数说明： MRs： MDS 寄存器

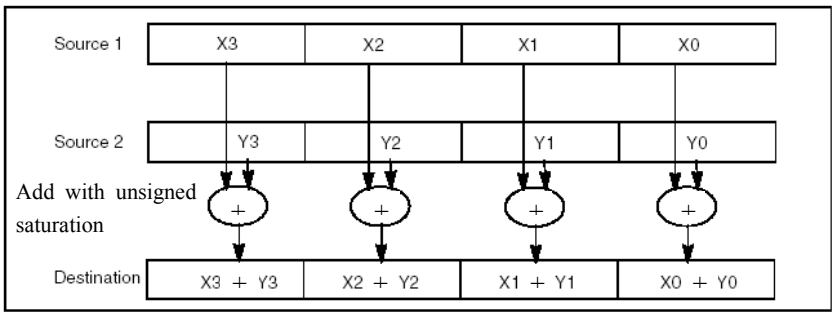
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PADDUSB 对 64－bit MRs 操作数中 8 个打包的字节数和 64－bit MRt 操作数中 8 个打包的字节数，执行 SIMD 无符号加法，结果存入 MRd 操作数中相应的位置。PADDUSD 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 无符号加法，结果存入 MRd 操作数中相应的位置。下图示例 PADDUSD 的操作过程，PADDUSB 类推。



执行周期：1 cycle

PAND

句型： **PAND** *MRd，MRs，MRt*

PAND *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					100100					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					100100					

操作：

PAND：

MRd ← MRs AND MRt;

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PAND 对 64－bit MRs 操作数和 64－bit MRt 操作数，执行按位逻辑与运算，结果存入 MRd 操作数。

执行周期：1 cycle

PAVGB/D

句型： **PAVGB/D** *MRd，MRs，MRt*

PAVGB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					111100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					111100						

操作：

PAVGB instruction with 64-bit operands:

MRt[7-0] ← (MRt[7-0] + MRs[7-0] + 1] >> 1; * temp sum before shifting is 9 bits *

* repeat operation performed for bytes 2 through 6;

MRt[63-56] ← (MRt[63-56] + MRs[63-56] + 1) >> 1;

PAVGD instruction with 64-bit operands:

MRt[15-0] ← (MRt[15-0] + MRs[15-0] + 1) >> 1; * temp sum before shifting is 17 bits *

* repeat operation performed for double-bytes 2 and 3;

MRt[63-48] ← (MRt[63-48] + MRs[63-48] + 1) >> 1;

操作数说明： MRs： MDS 寄存器

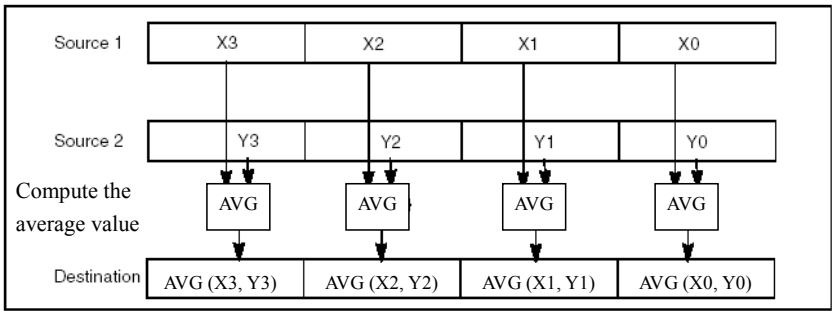
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PAVGB/D 对 64－bit MRs 操作数中打包的字节数/2 字节数和 64－bit MRt 操作数中打包的字节数/2 字节数，执行 SIMD 加法，每个和值再加 1，相应结果右移 1bit 作为 2 个数的平均值，存入 MRd 操作数中。下图示例了 PAVGD 的操作过程，PAVGB 类推。



执行周期：1 cycle

PCMPEQB/D/Q

句型： **PCMPEQB/D/Q** *MRd，MRs，MRt*

PCMPEQB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					110100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					110100						

操作：

PCMPEQB instruction with 64-bit operands:
IF MRs[7..0] = MRt[7..0]
THEN MRd[7 0] ← FFH;
ELSE MRd[7..0] ← 0;
* Continue comparison of 2nd through 7th bytes in MRd and MRt *
IF MRs[63..56] = MRt[63..56]
THEN MRd[63..56] ← FFH;
ELSE MRd[63..56] ← 0;

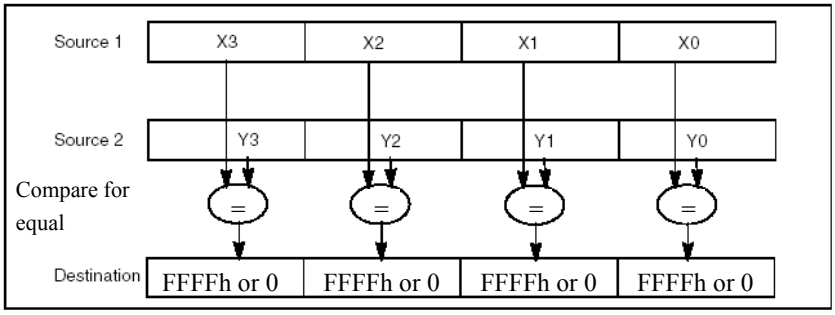
PCMPEQD instruction with 64-bit operands:
IF MRs[15..0] = MRt[15..0]
THEN MRd[15..0] ← FFFFH;
ELSE MRd[15..0] ← 0;
* Continue comparison of 2nd and 3rd double-bytes in MRd and MRt *
IF MRs[63..48] = MRt[63..48]
THEN MRd[63..48] ← FFFFH;
ELSE MRd[63..48] ← 0;

PCMPEQQ instruction with 64-bit operands:
IF MRs[31..0] = MRt[31..0]
THEN MRd[31..0] ← FFFFFFFFH;
ELSE MRd[31..0] ← 0;
IF MRs[63..32] = MRt[63..32]
THEN MRd[63..32] ← FFFFFFFFH;
ELSE MRd[63..32] ← 0;

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PCMPEQB/D/Q 对 64－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 64－bit

MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 相等比较，如果相等结果全置 1，否则全置 0，结果存入 MRd 操作数中。下图示例了 PCMPEQD 的操作过程，其它类推。



执行周期：1 cycle

PCMPGTB/D/Q

句型： **PCMPGTB/D/Q** *MRd，MRs，MRt*

PCMPGTB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					110000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					110000						

操作：

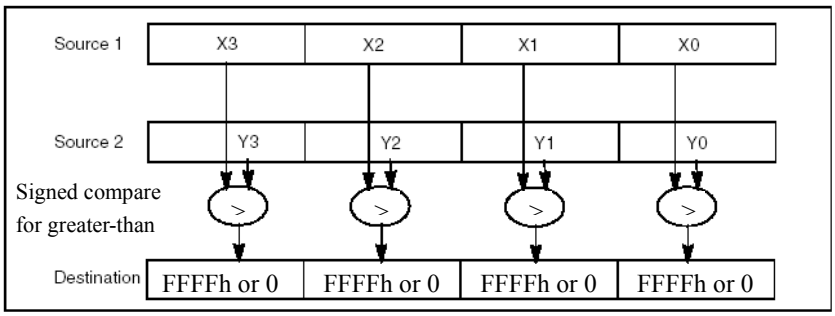
PCMPGTB instruction with 64-bit operands:
IF MRs[7..0] > MRt[7..0]
THEN MRd[7 0] ← FFH;
ELSE MRd[7..0] ← 0;
* Continue comparison of 2nd through 7th bytes in MRd and MRt *
IF MRs[63..56] > MRt[63..56]
THEN MRd[63..56] ← FFH;
ELSE MRd[63..56] ← 0;

PCMPGTD instruction with 64-bit operands:
IF MRs[15..0] > MRt[15..0]
THEN MRd[15..0] ← FFFFH;
ELSE MRd[15..0] ← 0;
* Continue comparison of 2nd and 3rd double-bytes in MRd and MRt *
IF MRs[63..48] > MRt[63..48]
THEN MRd[63..48] ← FFFFH;
ELSE MRd[63..48] ← 0;

PCMPGTQ instruction with 64-bit operands:
IF MRs[31..0] > MRt[31..0]
THEN MRd[31..0] ← FFFFFFFFH;
ELSE MRd[31..0] ← 0;
IF MRs[63..32] > MRt[63..32]
THEN MRd[63..32] ← FFFFFFFFH;
ELSE MRd[63..32] ← 0;

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： PCMPGTB/D/Q 对 64－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 64－bit MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 有符号比较，如果大于结果全置 1，否则全置 0，结果存入 MRd 操作数中。下图示例了 PCMPGTD 的操作过程，其它类推。



执行周期：1 cycle

PLOADO

句型： **PLOADO** *MRd， Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		000			disp		ARm			gg		MRd			Modm						110111					

操作：

PLOADO：

MRd[63-0] ← memory

操作数说明： MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PLOADO 从 memory 中指定的位置读取 64bit 数据，写入到 MDS 寄存器 MRd 中。

执行周期：1 cycle

PMADDQD

句型： **PMADDQD** *MRd，MRs，MRt*

PMADDQD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					101000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					101000						

操作：

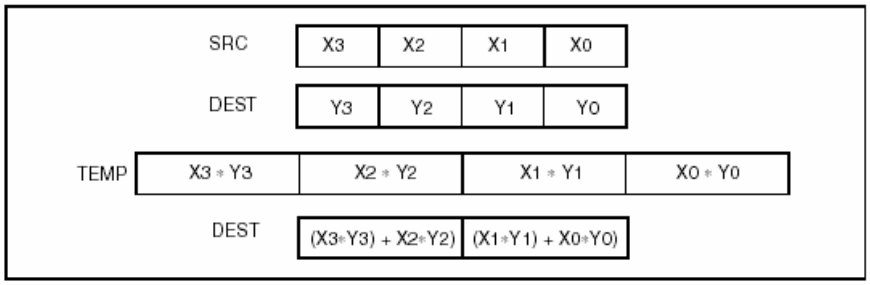
PMADDQD instruction with 64-bit operands:

MRd[31..0] ← (MRs[15..0]×MRt[15..0]) + (MRs[31..16]×MRt[31..16]);

MRd[63..32] ← (MRs[47..32]×MRt[47..32]) + (MRs[63..48]×MRt[63..48]);

* Signed multiplication *

描述： PMADDQD 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，然后相邻的 2 个 32－bit 结果相加成 1 个 32－bit 结果，最后结果存入 MRd 操作数。下图示例了 PMADDQD 的操作过程。



执行周期：4 cycles

PMAXSD

句型： **PMAXSD** *MRd，MRs，MRt*

PMAXSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					111000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					111000						

操作：

PMAXSD instruction for 64-bit operands:

IF MRs[15-0] > MRt[15-0]) THEN

(MRd[15-0] ← MRd[15-0];

ELSE

(MRd[15-0] ← MRt[15-0];

FI

* repeat operation for 2nd and 3rd double-bytes in source and destination operands *

IF MRs[63-48] > MRt[63-48]) THEN

(MRd[63-48] ← MRd[63-48];

ELSE

(MRd[63-48] ← MRt[63-48];

FI

操作数说明： MRs： MDS 寄存器

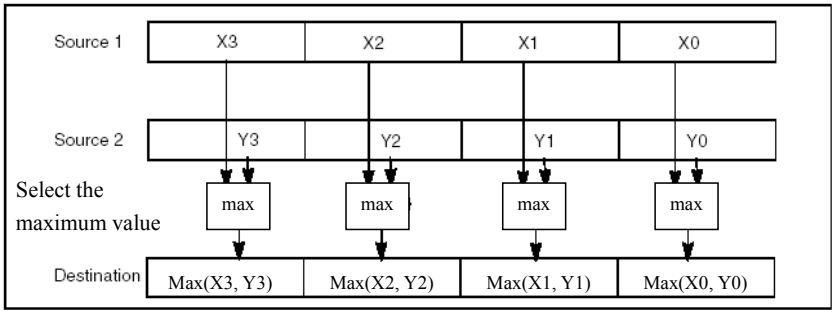
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMAXSD 对 64－bit MRs 操作数中 4 个打包的有符号 2 字节数和 64－bit MRt 操作数中 4 个打包的有符号 2 字节数，执行 SIMD 有符号比较，相应较大的数存入 MRd 操作数中。下图示例了 PMAXSD 的操作过程。



执行周期：1 cycle

PMAXUB

句型： **PMAXUB** *MRd，MRs，MRt*

PMAXUB *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					111001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					111001						

操作：

PMAXUB instruction for 64-bit operands:

IF MRs[7-0] > MRt[17-0]) THEN

(MRd[7-0] ← MRd[7-0];

ELSE

(MRd[7-0] ← MRt[7-0];

FI

* repeat operation for 2nd through 7th bytes in source and destination operands *

IF MRs[63-56] > MRt[63-56]) THEN

(MRd[63-56] ← MRd[63-56];

ELSE

(MRd[63-56] ← MRt[63-56];

FI

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMAXUB 对 64－bit MRs 操作数中 8 个打包的无符号字节数和 64－bit MRt 操作数中 8 个打包的无符号字节数，执行 SIMD 比较，相应较大的数存入 MRd 操作数中。PMAXUB 操作过程类似 PMAXSD。

执行周期：1 cycle

PMFHI

句型： **PMFHI** *MRs，Rt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		MRs			Rt					10		000			00000					010000					

操作：

PMFHI:

Rt[31-0] ← MRs[63-32]

操作数说明： MRs： MDS 寄存器

Rt： 通用寄存器

描述： PMFHI 将 MDS 寄存器 MRs 的高 32 位的值写入到通用寄存器 Rt。

执行周期： **1** cycle

PMFLO

句型： **PMFLO** *MRs，Rt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		MRs			Rt					10		000			00000					010010					

操作：

PMFHI:

Rt[31-0] ← MRs[31-0]

操作数说明： MRs： MDS 寄存器

Rt： 通用寄存器

描述： PMFLO 将 MDS 寄存器 MRs 的低 32 位的值写入到通用寄存器 Rt。

执行周期： **1** cycle

PMINSD

句型： **PMINSD** *MRd，MRs，MRt*

PMINSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					111010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					111010					

操作：

PMINSD instruction for 64-bit operands:

IF MRs[15-0] < MRt[15-0]) THEN

(MRd[15-0] ← MRd[15-0];

ELSE

(MRd[15-0] ← MRt[15-0];

FI

* repeat operation for 2nd and 3rd double-bytes in source and destination operands *

IF MRs[63-48] < MRt[63-48]) THEN

(MRd[63-48] ← MRd[63-48];

ELSE

(MRd[63-48] ← MRt[63-48];

FI

操作数说明： MRs： MDS 寄存器

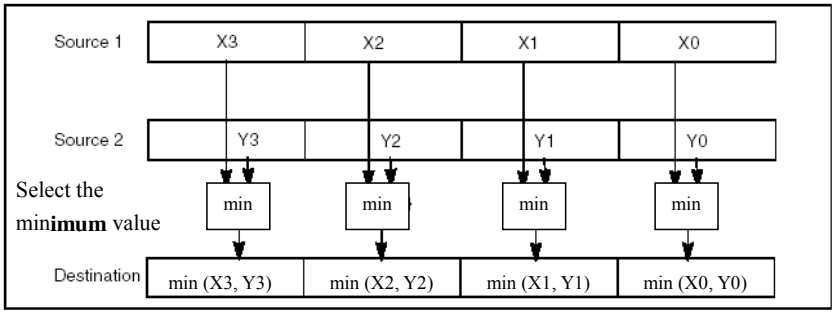
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMINSD 对 64－bit MRs 操作数中 4 个打包的有符号 2 字节数和 64－bit MRt 操作数中 4 个打包的有符号 2 字节数，执行 SIMD 有符号比较，相应较小的数存入 MRd 操作数中。下图示例了 PMINSD 的操作过程。



执行周期：1 cycle

PMINUB

句型： **PMINUB** *MRd，MRs，MRt*

PMINUB *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					111011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					111011					

操作：

PMINUB instruction for 64-bit operands:

IF MRs[7-0] < MRt[17-0]) THEN

(MRd[7-0] ← MRd[7-0];

ELSE

(MRd[7-0] ← MRt[7-0];

FI

* repeat operation for 2nd through 7th bytes in source and destination operands *

IF MRs[63-56] < MRt[63-56]) THEN

(MRd[63-56] ← MRd[63-56];

ELSE

(MRd[63-56] ← MRt[63-56];

FI

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PMINUB 对 64－bit MRs 操作数中 8 个打包的无符号字节数和 64－bit MRt 操作数中 8 个打包的无符号字节数，执行 SIMD 比较，相应较小的数存入 MRd 操作数中。PMINUB 操作过程类似 PMINSD。

执行周期：1 cycle

PMTLO

句型： **PMTLO** MRd，Rs

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			Rs					10		MRd			00000					010011					

操作：

PMTLO:

MRd[31-0] ← Rs[31-0]

操作数说明： MRd： MDS 寄存器

Rs： 通用寄存器

描述： PMTLO 将通用寄存器 Rs 的值写入到 MDS 寄存器 MRd 的低 32 位。

执行周期： **1** cycle

PMTHI

句型： **PMTHI** MRd，Rs

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						10		000			Rs					10		MRd			00000					010001					

操作：

PMTHI:

MRd[63-32] ← Rs[31-0]

操作数说明： MRd： MDS 寄存器

Rs： 通用寄存器

描述： PMTHI 将通用寄存器 Rs 的值写入到 MDS 寄存器 MRd 的高 32 位。

执行周期：1 cycle

PMULHSD

PMACHSD

句型： **PMULHSD** *MRd，MRs，MRt*
PMACHSD *MRd，MRs，MRt*
PMULHSD *MRd，MRs，Modm(ARm)*
PMACHSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					011100/011110					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					011100/011110					

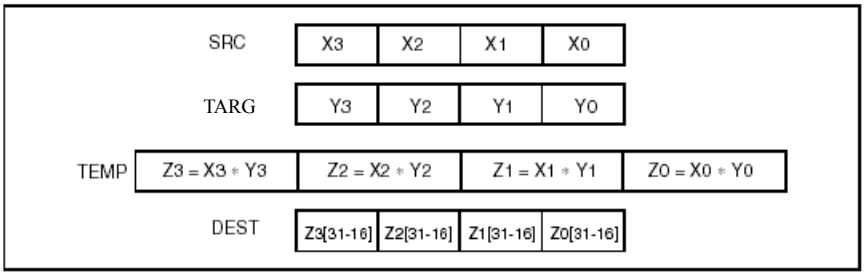
操作：

PMULHSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Signed multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← TEMP0[31-16];
MRd[31-16] ← TEMP1[31-16];
MRd[47-32] ← TEMP2[31-16];
MRd[63-48] ← TEMP3[31-16];

PMACHSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Signed multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← MRd[15-0] + TEMP0[31-16];
MRd[31-16] ← MRd[31-16] + TEMP1[31-16];
MRd[47-32] ← MRd[47-32] + TEMP2[31-16];
MRd[63-48] ← MRd[63-48] + TEMP3[31-16];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的高 16－bit 存入 MRd 操作数中相应的位置。下图示例 PMULHSD 的操作过程。PMACHSD 将每次乘法结果不断累加。



执行周期：2 cycles

PMULHUD

PMACHUD

句型： **PMULHUD** *MRd，MRs，MRt*
PMACHUD *MRd，MRs，MRt*
PMULHUD *MRd，MRs，Modm(ARm)*
PMACHUD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					011101/011111					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					011101/011111					

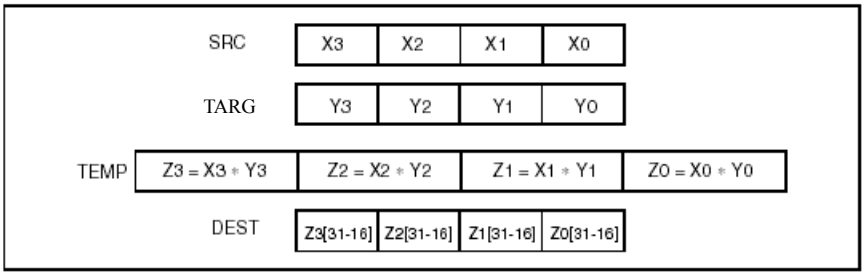
操作：

PMULHUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Unsigned multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← TEMP0[31-16];
MRd[31-16] ← TEMP1[31-16];
MRd[47-32] ← TEMP2[31-16];
MRd[63-48] ← TEMP3[31-16];

PMACHUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Unsigned multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← MRd[15-0] + TEMP0[31-16];
MRd[31-16] ← MRd[31-16] + TEMP1[31-16];
MRd[47-32] ← MRd[47-32] + TEMP2[31-16];
MRd[63-48] ← MRd[63-48] + TEMP3[31-16];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的高 16－bit 存入 MRd 操作数中相应的位置。下图示例 PMULHUD 的操作过程。PMACHUD 将每次乘法结果不断累加。



执行周期：2 cycles

PMULLSD

PMACLSD

句型： **PMULLSD** *MRd，MRs，MRt*
PMACLSD *MRd，MRs，MRt*
PMULLSD *MRd，MRs，Modm(ARm)*
PMACLSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					011000/011010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					011000/011010					

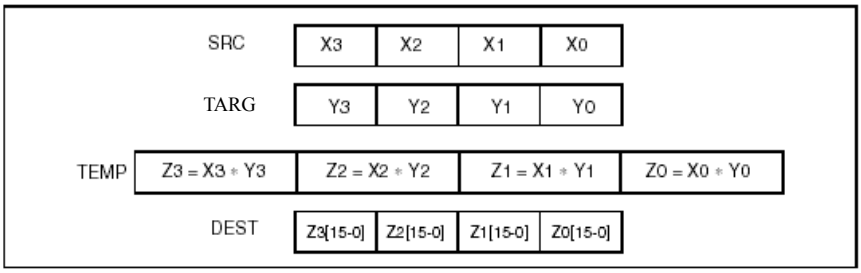
操作：

PMULLSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Signed multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← TEMP0[15-0];
MRd[31-16] ← TEMP1[15-0];
MRd[47-32] ← TEMP2[15-0];
MRd[63-48] ← TEMP3[15-0];

PMACLSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Signed multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← MRd[15-0] + TEMP0[15-0];
MRd[31-16] ← MRd[31-16] + TEMP1[15-0];
MRd[47-32] ← MRd[47-32] + TEMP2[15-0];
MRd[63-48] ← MRd[63-48] + TEMP3[15-0];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的低 16－bit 存入 MRd 操作数中相应的位置。下图示例 PMULLSD 的操作过程。PMACLSD 将每次乘法结果不断累加。



执行周期：2 cycles

PMULLUD

PMACLUD

句型： **PMULLUD** *MRd，MRs，MRt*
PMACLUD *MRd，MRs，MRt*
PMULLUD *MRd，MRs，Modm(ARm)*
PMACLUD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					011001/011011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					011001/011011					

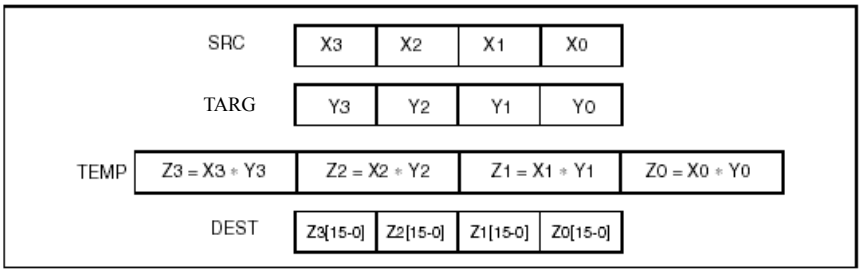
操作：

PMULLUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Unsigned multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← TEMP0[15-0];
MRd[31-16] ← TEMP1[15-0];
MRd[47-32] ← TEMP2[15-0];
MRd[63-48] ← TEMP3[15-0];

PMACLUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0]; * Unsigned multiplication *
TEMP1[31-0] ← MRs[31-16]×MRt[31-16];
TEMP2[31-0] ← MRs[47-32]×MRt[47-32];
TEMP3[31-0] ← MRs[63-48]×MRt[63-48];
MRd[15-0] ← MRd[15-0] + TEMP0[15-0];
MRd[31-16] ← MRd[31-16] + TEMP1[15-0];
MRd[47-32] ← MRd[47-32] + TEMP2[15-0];
MRd[63-48] ← MRd[63-48] + TEMP3[15-0];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 无符号乘法，每个 32－bit 结果的低 16－bit 存入 MRd 操作数中相应的位置。下图示例 PMULLUD 的操作过程。PMACLUD 将每次乘法结果不断累加。



执行周期：2 cycles

PNOR

句型： **PNOR** *MRd，MRs，MRt*

PNOR *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			11		MRd			00000					100011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			11		MRd			Modm					100011					

操作：

PNOR：

MRd ← MRd NOR MRt;

描述： PNOR 对 64－bit MRs 操作数和 64－bit MRt 操作数，执行按位逻辑或非运算，结果存入 MRd 操作数。

执行周期：1 cycle

POR

句型： **POR** *MRd，MRs，MRt*

POR *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			11		MRd			00000					100101					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			11		MRd			Modm					100101					

操作：

POR：

MRd ← MRd OR MRt;

描述： POR 对 64－bit MRd 操作数和 64－bit MRt 操作数，执行按位逻辑或运算，结果存入 MRd 操作数。

执行周期：1 cycle

PSADBD

句型： **PSADBD** *MRd，MRs，MRt*

PSADBD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			11		MRd			00000					101001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			11		MRd			Modm					101001					

操作：

PSADBD instructions when using 64-bit operands:

TEMP0 ← ABS(MRs] － MRt[7-0]);

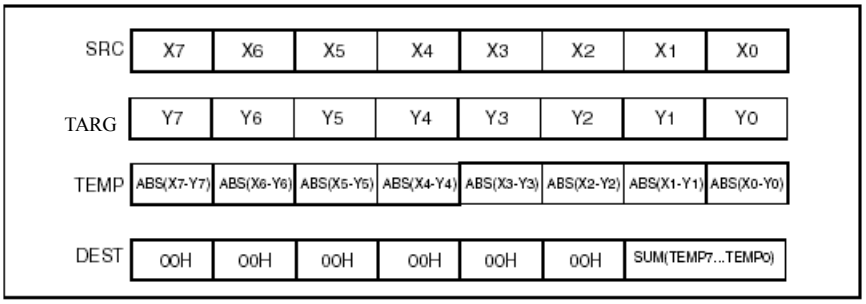
* repeat operation for bytes 2 through 6 *;

TEMP7 ← ABS(MRs[56] － MRt[63-56]);

MRd[15:0] ← SUM(TEMP0…TEMP7);

MRd[63:16] ← 000000000000H;

描述： PSADBD 对 64－bit MRs8 个打包的字节数和 64－bit MRt 操作数中 8 个打包的字节数，执行 SIMD 减法，取绝对值得到绝对差值，然后 8 个绝对差值相加成 1 个 16－bit 无符号数，存入 MRd 操作数的低 16－bit，MRd 操作数的高 48bit 置 0。下图示例了 PSADBD 的操作过程。



执行周期：2 cycles

PSHUFD

句型： **PSHUFD** *MRd，MRs，imm*

PSHUFD *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					000001					

操作：

PSHUFD:

IF (ORDER = 0)

THEN MRd[15:0]← MRs[15:0];

IF (ORDER = 1)

THEN MRd[15:0] ← MRs[31:16];

IF (ORDER = 2)

THEN MRd[15:0] ← MRs[46:32];

IF (ORDER = 3)

THEN MRd[15:0] ← MRs[63:47];

*Repeat operation for 2nd, 3rd, 4rddouble-bytes;

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

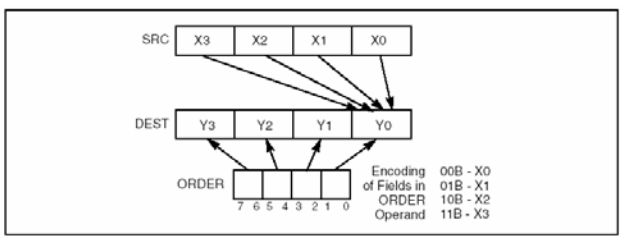
MRd： MDS 寄存器

Sa： 立即数

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： *MRs* 中打包的 2 字节数据作换位排列，换位控制来自 *MRt*（*imm*）的最低 8bit。MRd 中每个 16bit 数据取自 MRs 中 4 个 16bit 数据的其中一个。这样，每个结果 16bit 数据的产生需要 2bit 选择信号，总共需要 8bit 选择信号，由立即数 sa 的最低 8 个 bit 表示（高 2bit 汇编时置 0）或 MRt 操作数的最低 8 个 bit 表示。



执行周期：1 cycle

PSRAD/Q

句型： **PSRAD/Q** *MRd，MRs，imm*

PSRAD/Q *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					000011					

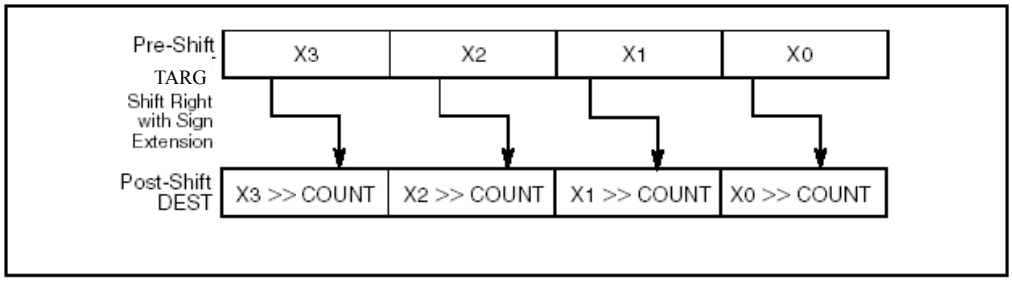
操作：

PSRAD:
IF (COUNT > 15)
THEN COUNT ← 16;
FI;
MRd[15..0] ← SignExtend(MRs[15..0] >> COUNT);
* repeat shift operation for 2nd and 3rd double-bytes *;
MRd[63..48] ← SignExtend(MRs[63..48] >> COUNT);

PSRAQ:
IF (COUNT > 31)
THEN COUNT ← 32;
FI;
ELSE
MRd[31..0] ← SignExtend(MRs[31..0] >> COUNT);
MRd[63..32] ← SignExtend(MRs[63..32] >> COUNT);

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
Sa： 立即数
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 64bit MRs 操作数中打包的 2 字节/4 字节进行 SIMD 算术右移，结果存入 MRd 操作数。下图示例了 PSRAQ 的操作过程，其它类推。



执行周期：1 cycle

PSRLD/Q/O

句型： **PSRLD/Q/O** *MRd，MRs，imm*

PSRLD/Q/O *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00			MRt		gg		MRd			00000					000010					

操作：

PSRLD:

```
IF (COUNT > 15)
THEN
        MRd[64..0] ← 0000000000000000H
ELSE
        MRd[15..0] ← ZeroExtend(MRs[15..0] >> COUNT);
        * repeat shift operation for 2nd and 3rd double-bytes *;
        MRd[63..48] ← ZeroExtend(MRs[63..48] >> COUNT);
FI;
```

PSRLQ:

```
IF (COUNT > 31)
THEN
        MRd[64..0] ← 0000000000000000H
ELSE
        MRd[31..0] ← ZeroExtend(MRs[31..0] >> COUNT);
        MRd[63..32] ← ZeroExtend(MRd[63..32] >> COUNT);
FI;
```

操作数说明： MRs： MDS寄存器

MRt： MDS 寄存器

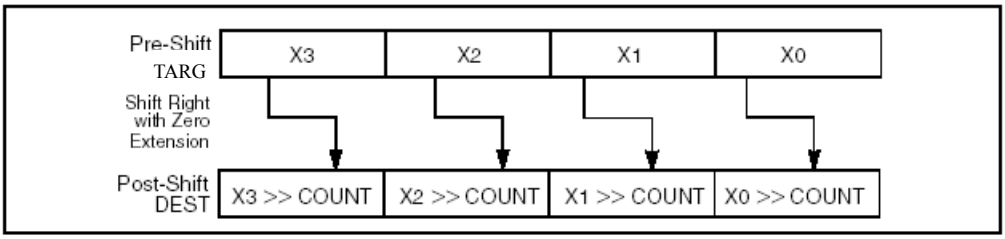
MRd： MDS 寄存器

Sa： 立即数

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 64bit MRs 操作数中打包的 2 字节/4 字节/8 字节进行 SIMD 逻辑右移，结果存入 MRd 操作数。下图示例了 PSRLQ 的操作过程，其它类推。



执行周期： **1** cycle

PSLLD/Q/O

句型： **PSLLD/Q/O** *MRd，MRs，imm*

PSLLD/Q/O *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00			MRt		gg		MRd			00000					000000					

操作：

```
PSLLD:
    IF (COUNT > 15)
        THEN
            MRd[64..0] ← 0000000000000000H
    ELSE
        MRd[15..0] ← ZeroExtend(MRs[15..0] << COUNT);
        * repeat shift operation for 2nd and 3rd double-bytes *;
        MRd[63..48] ← ZeroExtend(MRs[63..48] << COUNT);
    FI;
PSLLQ:
    IF (COUNT > 31)
        THEN
            MRd[64..0] ← 0000000000000000H
    ELSE
        MRd[31..0] ← ZeroExtend(MRs[31..0] << COUNT);
        MRd[63..32] ← ZeroExtend(MRs[63..32] << COUNT);
    FI;
```

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

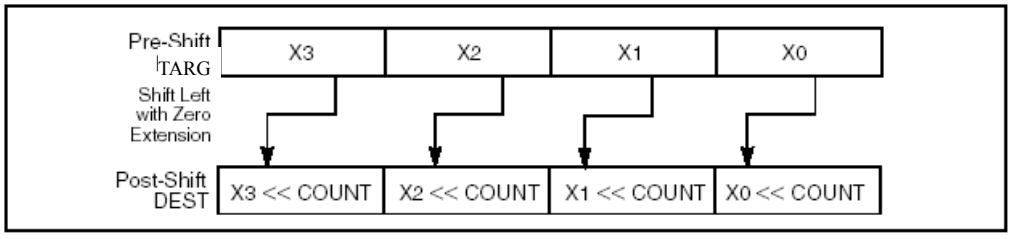
MRd： MDS 寄存器

Sa： 立即数

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 64bit MRs 操作数中打包的 2 字节/4 字节/8 字节进行 SIMD 逻辑左移，结果存入 MRd 操作数。下图示例了 PSLLQ 的操作过程，其它类推。



执行周期：1 cycle

PSTOREO

句型： **PSTOREO** *MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			11		000			Modm					111111					

操作：

PSTOREO：

Memory ← MRs[63-0]

操作数说明： MRs： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSTOREO 将 MDS 寄存器 MRd 中的 64bit 数据写入到 memory 中指定的位置。

执行周期：1 cycle

PSUBB/D/Q

句型： **PSUBB/D/Q** *MRd，MRs，MRt*

PSUBB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					101110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					101110						

操作：

PSUBB instruction with 64-bit operands:

MRd[7..0] ← MRs[7..0] － MRt[7..0];

* repeat add operation for 2nd through 7th byte *;

MRd[63..56] ← MRs[63..56] － MRt[63..56];

PSUBD instruction with 64-bit operands:

MRd[15..0] ← MRs[15..0] － MRt[15..0];

* repeat add operation for 2nd and 3th double-byte *;

MRd[63..48] ← MRs[63..48] － MRt[63..48];

PSUBQ instruction with 64-bit operands:

MRd[31..0] ← MRs[31..0] － MRt[31..0];

MRd[63..32] ← MRs[63..32] － MRt[63..32];

操作数说明： MRs： MDS 寄存器

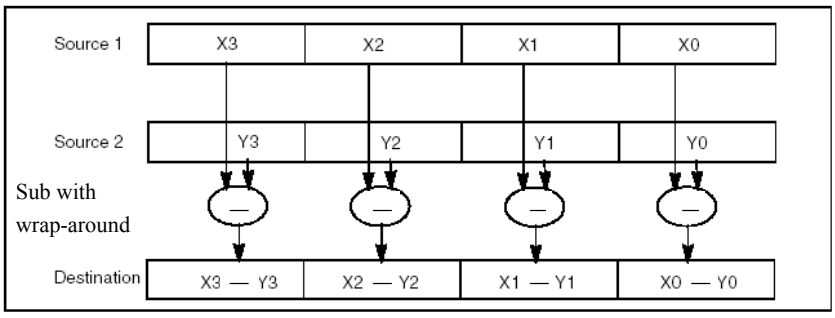
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSUBB/D/Q 对 64－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 64－bit MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 减法，结果存入 MRd 操作数中，溢出被忽略。下图示例 PSUBD 的操作过程，其它类推。



执行周期：1 cycle

PSUBSB/D

句型： **PSUBSB/D** *MRd，MRs，MRt*

PSUBSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					100001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					100001					

操作：

PSUBSB instruction with 64-bit operands:

MRd[7..0] ← SaturateToSignedByte(MRs[7..0] － MRt (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

MRd[63..56] ← SaturateToSignedByte(MRs[63..56] － MRt[63..56]);

PSUBSD instruction with 64-bit operands:

MRd[15..0] ← SaturateToSignedDouble-byte(MRs[15..0] － MRt[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

MRd[63..48] ← SaturateToSignedDouble-byte(MRs[63..48] － MRt[63..48]);

操作数说明： MRs： MDS 寄存器

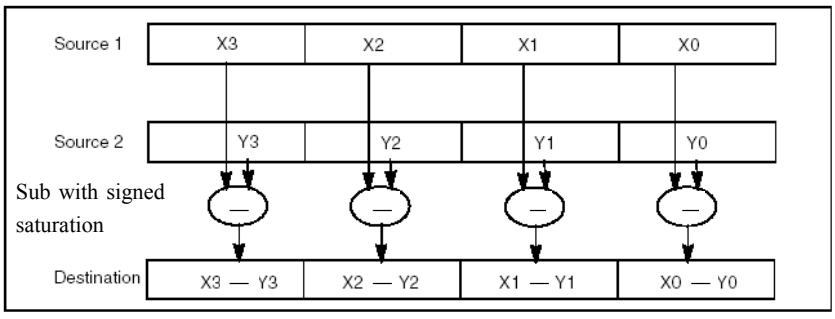
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSUBSB 对 64－bit MRs 操作数中 8 个打包的字节数和 64－bit MRt 操作数中 8 个打包的字节数，执行 SIMD 有符号减法，结果存入 MRd 操作数中相应的位置。PSUBSD 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 有符号减法，结果存入 MRd 操作数中相应的位置。下图示例 PSUBSD 的操作过程，PSUBSB 类推。



执行周期：1 cycle

PSUBUSB/D

句型： **PSUBUSB/D** *MRd，MRs，MRt*

PSUBUSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					100011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					100011					

操作：

PSUBUSB instruction with 64-bit operands:

MRd[7..0] ← SaturateToUnsignedByte(MRs[7..0] － MRt (7..0)) ;

* repeat add operation for 2nd through 7th bytes *;

MRd[63..56] ← SaturateToUnsignedByte(MRs[63..56] － MRt[63..56]);

PSUBUSD instruction with 64-bit operands:

MRd[15..0] ← SaturateToUnsignedDouble-byte(MRs[15..0] － MRt[15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

MRd[63..48] ← SaturateToUnsignedDouble-byte(MRs[63..48] － MRt[63..48]);

操作数说明： MRs： MDS 寄存器

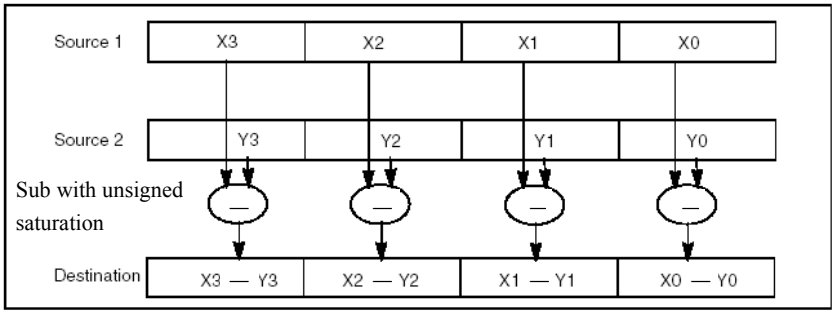
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSUBUSB 对 64－bit MRs 操作数中 8 个打包的字节数和 64－bit MRt 操作数中 8 个打包的字节数，执行 SIMD 无符号减法，结果存入 MRd 操作数中相应的位置。PSUBUSD 对 64－bit MRs 操作数中 4 个打包的 2 字节数和 64－bit MRt 操作数中 4 个打包的 2 字节数，执行 SIMD 无符号减法，结果存入 MRd 操作数中相应的位置。下图示例 PSUBUSD 的操作过程，PSUBUSB 类推。



执行周期：1 cycle

PUNPCKHBD/DQ/QO

句型： **PUNPCKHBD/DQ/QO** *MRd，MRs，MRt*

PUNPCKHBD/DQ/QO *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					001010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					001010					

操作：

PUNPCKHBD:

MRd[7..0] ←MRs[39..32];
MRd[15..8] ←MRt[39..32];
MRd[23..16] ←MRs[47..40];
MRd[31..24] ←MRt[47..40];
MRd[39..32] ←MRs[55..48];
MRd[47..40] ←MRt[55..48];
MRd[55..48] ←MRs[63..56];
MRd[63..56] ←MRt[63..56];

PUNPCKHDQ:

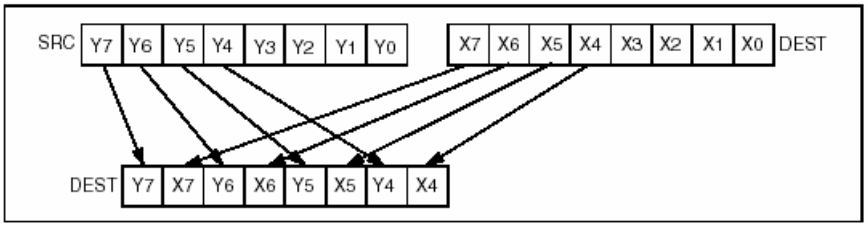
MRd[15..0] ←MRs[47..32];
MRd[31..16] ←MRt[47..32];
MRd[47..32] ←MRs[63..48];
MRd[63..48] ←MRt[63..48];

PUNPCKHQO:

MRd[31..0] ←MRs[63..32]
MRd[63..32] ←MRt[63..32];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 将 64bit MRt 和 MRs 操作数中打包的字节/2 字节/4 字节相交织，取高 64bit 存入 MRd 操作数。下图示例了 PUNPCKHBD 的操作过程，其它类推。



执行周期： **1** cycle

PUNPCKLBD/DQ/QO

句型： **PUNPCKLBD/DQ/QO** *MRd，MRs，MRt*

PUNPCKLBD/DQ/QO *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					001011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					001011					

操作：

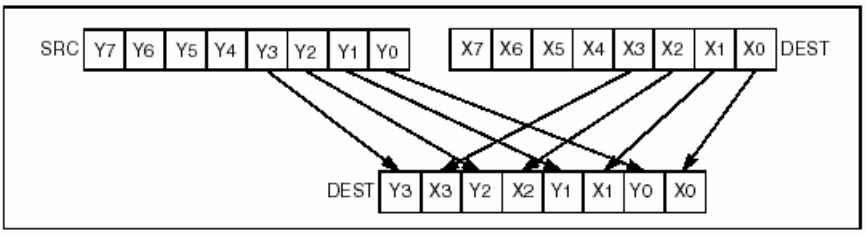
PUNPCKLBD:
MRd[63..56] ←MRt[31..24];
MRd[55..48] ←MRs[31..24];
MRd[47..40] ←MRt[23..16];
MRd[39..32] ←MRs[23..16];
MRd[31..24] ←MRt[15..8];
MRd[23..16] ←MRs[15..8];
MRd[15..8] ←MRt[7..0];
MRd[7..0] ←MRs[7..0];

PUNPCKLDQ:
MRd[63..48] ←MRt[31..16];
MRd[47..32] ←MRs[31..16];
MRd[31..16] ←MRt[15..0];
MRd[15..0] ←MRs[15..0];

PUNPCKLQO:
MRd[63..32] ←MRt[31..0];
MRd[31..0] ←MRs[31..0];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 将 64bit MRt 和 MRs 操作数中打包的字节/2 字节/4 字节相交织，取低 64bit 存入 MRd 操作数。下图示例了 PUNPCKLBD 的操作过程，其它类推。



执行周期：1 cycle

PXOR

句型： **PXOR** *MRd，MRs，MRt*

PXOR *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					100110					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						11		MRs			disp		ARm			gg		MRd			Modm					100110					

操作：

PXOR：

MRd ← MRs XOR MRt;

描述： PXOR 对 64－bit MRs 操作数和 64－bit MRt 操作数，执行按位逻辑异或运算，结果存入 MRd 操作数。

执行周期：1 cycle

EPACKSSDB/QD

句型： **EPACKSSBD/QD** *MRd，MRs，MRt*

EPACKSSDB/QD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					010100					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					010100					

操作：

EPACKSSDB:

MRd[7..0] ← SaturateSignedDouble-byteToSignedByte MRs[15..0];
MRd[15..8] ← SaturateSignedDouble-byteToSignedByte MRs [31..16];
MRd[23..16] ← SaturateSignedDouble-byteToSignedByte MRs[47..32];
MRd[31..24] ← SaturateSignedDouble-byteToSignedByte MRs[63..48];
MRd[39..32] ← SaturateSignedDouble-byteToSignedByte MRs[79..64];
MRd[47..40] ← SaturateSignedDouble-byteToSignedByte MRs[95..80];
MRd[55..48] ← SaturateSignedDouble-byteToSignedByte MRs[111..96];
MRd[63..56] ← SaturateSignedDouble-byteToSignedByte MRs[127..112];
MRd[71..64] ← SaturateSignedDouble-byteToSignedByte MRt[15..0];
MRd[79..72] ← SaturateSignedDouble-byteToSignedByte MRt [31..16];
MRd[87..80] ← SaturateSignedDouble-byteToSignedByte MRt[47..32];
MRd[95..88] ← SaturateSignedDouble-byteToSignedByte MRt[63..48];
MRd[103..96] ← SaturateSignedDouble-byteToSignedByte MRt[79..64];
MRd[111..104] ← SaturateSignedDouble-byteToSignedByte MRt[95..80];
MRd[119..112] ← SaturateSignedDouble-byteToSignedByte MRt[111..96];
MRd[127..120] ← SaturateSignedDouble-byteToSignedByte MRt[127..112];

EPACKSSQD:

MRd[15..0] ←SaturateSignedDwordToSignedword MRs[31..0];
MRd[31..16] ←SaturateSignedDwordToSignedword MRs[63..32];
MRd[47..32] ←SaturateSignedDwordToSignedword MRs[95..64];
MRd[63..48] ←SaturateSignedDwordToSignedword MRs[127..96];
MRd[79..64] ←SaturateSignedDwordToSignedword MRt[31..0];
MRd[95..80] ←SaturateSignedDwordToSignedword MRt[63..32];
MRd[111..96] ←SaturateSignedDwordToSignedword MRt[95..64];
MRd[127..112]←SaturateSignedDwordToSignedword MRt[127..96];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： EPACKSSDB 将 128-bit MRs 操作数中打包的 8 个有符号 2 字节数和 128-bit MRt 操作数中打包的 8 个有符号 2 字节数转换为 16 个有符号字节数，采用有符号饱和法处理溢出，结果存入 MRd 操作数。

EPACKSSQD 将 128-bit MRs 操作数中打包的 4 个有符号 4 字节数和 128-bit MRt 操作数中打包的 4 个有符号 4 字节数转换为 8 个有符号 2 字节数，采用有符号饱

和法处理溢出，结果存入 MRd 操作数。

执行周期：1 cycle

EPACKUSDB/QD

句型： **EPACKUSBD/QD** *MRd，MRs，MRt*

EPACKUSDB/QD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					010101					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					010101					

操作：

EPACKUSDB:

MRd[7..0] ← SaturateUnsignedDouble-byteToUnsignedByte MRs[15..0];
MRd[15..8] ← SaturateUnsignedDouble-byteToUnsignedByte MRs [31..16];
MRd[23..16] ← SaturateUnsignedDouble-byteToUnsignedByte MRs[47..32];
MRd[31..24] ← SaturateUnsignedDouble-byteToUnsignedByte MRs[63..48];
MRd[39..32] ← SaturateUnsignedDouble-byteToUnsignedByte MRs[79..64];
MRd[47..40] ← SaturateUnsignedDouble-byteToUnsignedByte MRs[95..80];
MRd[55..48] ← SaturateUnsignedDouble-byteToUnsignedByte MRs[111..96];
MRd[63..56] ← SaturateUnsignedDouble-byteToUnsignedByte MRs[127..112];
MRd[71..64] ← SaturateUnsignedDouble-byteToUnsignedByte MRt[15..0];
MRd[79..72] ← SaturateUnsignedDouble-byteToUnsignedByte MRt [31..16];
MRd[87..80] ← SaturateUnsignedDouble-byteToUnsignedByte MRt[47..32];
MRd[95..88] ← SaturateUnsignedDouble-byteToUnsignedByte MRt[63..48];
MRd[103..96] ← SaturateUnsignedDouble-byteToUnsignedByte MRt[79..64];
MRd[111..104] ← SaturateUnsignedDouble-byteToUnsignedByte MRt[95..80];
MRd[119..112] ← SaturateUnsignedDouble-byteToUnsignedByte MRt[111..96];
MRd[127..120] ← SaturateUnsignedDouble-byteToUnsignedByte MRt[127..112];

EPACKUSQD:

MRd[15..0] ←SaturateUnsignedDwordToUnsignedword MRs[31..0];
MRd[31..16] ← SaturateUnsignedDwordToUnsignedword MRs[63..32];
MRd[47..32] ← SaturateUnsignedDwordToUnsignedword MRs[95..64];
MRd[63..48] ← SaturateUnsignedDwordToUnsignedword MRs[127..96];
MRd[79..64] ← SaturateUnsignedDwordToUnsignedword MRt[31..0];
MRd[95..80] ← SaturateUnsignedDwordToUnsignedword MRt[63..32];
MRd[111..96] ← SaturateUnsignedDwordToUnsignedword MRt[95..64];
MRd[127..112] ← SaturateUnsignedDwordToUnsignedword MRt[127..96];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： EPACKUSDB 将 128-bit MRs 操作数中打包的 8 个无符号 2 字节数和 128-bit MRt 操作数中打包的 8 个无符号 2 字节数转换为 16 个无符号字节数，采用无符号饱和法处理溢出，结果存入 MRd 操作数。

EPACKUSQD 将 128-bit MRs 操作数中打包的 4 个无符号 4 字节数和 128-bit MRt 操作数中打包的 4 个无符号 4 字节数转换为 8 个无符号 2 字节数，采用无符号饱

和法处理溢出，结果存入 MRd 操作数。

执行周期：1 cycle

EPADDB/D/Q

句型： **EPADDB/D/Q** *MRd，MRs，MRt*

EPADDB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					101100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					101100						

操作：

EPADDB instruction with 128-bit operands:

MRd[7..0] ← MRs[7..0] + MRt[7..0];

* repeat add operation for 2nd through 15th byte *;

MRd[127..120] ← MRs[127..120] + MRt[127..120];

EPADDD instruction with 128-bit operands:

MRd [15..0] ← MRs [15..0] + MRt [15..0];

* repeat add operation for 2nd and 7th double-byte *;

MRd[127..112] ← MRs[127..112] + MRt[127..112];

EPADDQ instruction with 128-bit operands:

MRd [31..0] ← MRs [31..0] + MRs [31..0];

* repeat add operation for 2nd and 3th word *;

MRd[127..96] ← MRs[127..96] + MRt[127..96];

操作数说明： MRs： MDS 寄存器

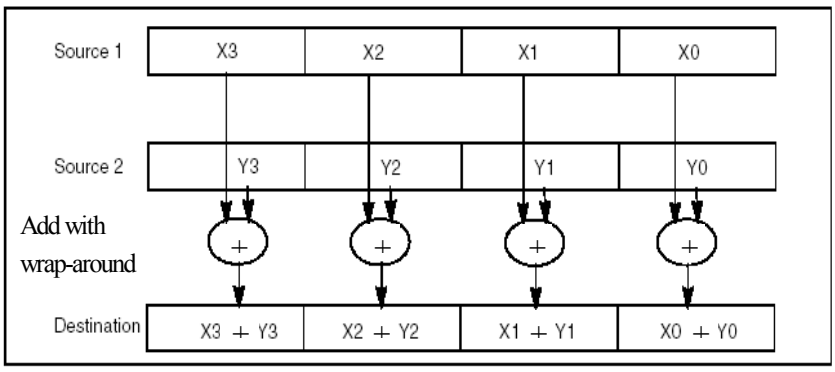
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPADDB/D/Q 对 128－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 128－bit MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 加法，结果存入 MRd 操作数中，溢出被忽略。下图示例 EPADDQ 的操作过程，其它类推。


Source 1
X3
X2
X1
X0
Source 2
Y3
Y2
Y1
Y0
Add with
wrap-around
+
+
+
+
Destination
X3 + Y3
X2 + Y2
X1 + Y1
X0 + Y0

执行周期：1 cycle

EPADDSB/D

句型： **EPADDSB/D** *MRd，MRs，MRt*

EPADDSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1111100						01		MRs			00		MRt			gg		MRd			00000					100000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					100000						

操作：

EPADDSB instruction with 128-bit operands:

MRd[7..0] ← SaturateToSignedByte(MRs[7..0] + MRt [7..0]) ;

* repeat add operation for 2nd through 15th bytes *;

MRd[127..120] ← SaturateToSignedByte(MRs[127..120] + MRt[127..120]);

EPADDSD instruction with 128-bit operands:

MRd [15..0] ← SaturateToSignedDouble-byte(MRs [15..0] + MRt [15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

MRd [127..112] ← SaturateToSignedDouble-byte(MRs [127..112] + MRt [127..112]);

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPADDSB 对 128－bit MRs 操作数中 16 个打包的字节数和 128－bit MRt 操作数中 16 个打包的字节数，执行 SIMD 有符号加法，使用有符号饱和处理溢出，结果存入 MRd 操作数中相应的位置。

EPADDSD 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 有符号加法，使用有符号饱和处理溢出，结果存入 MRd 操作数中相应的位置。

执行周期：1 cycle

EPADDUSB/D

句型： **EPADDUSB/D** *MRd，MRs，MRt*

EPADDUSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					100001						

操作：

EPADDUSB instruction with 128-bit operands:

MRd[7..0] ← SaturateToUnsignedByte(MRs[7..0] + MRt [7..0]) ;

* repeat add operation for 2nd through 15th bytes *;

MRd[127..120] ← SaturateToUnsignedByte(MRs[127..120] + MRt[127..120]);

EPADDUSD instruction with 128-bit operands:

MRd [15..0] ← SaturateToUnsignedDouble-byte(MRs [15..0] + MRt [15..0]);

* repeat add operation for 2nd and 3rd double-bytes *;

MRd [127..112] ← SaturateToUnsignedDouble-byte(MRs [127..112] + MRt [127..112]);

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPADDUSB 对 128－bit MRs 操作数中 16 个打包的字节数和 128－bit MRt 操作数中 16 个打包的字节数，执行 SIMD 无符号加法，使用无符号饱和处理溢出，结果存入 MRd 操作数中相应的位置。

EPADDUSD 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 无符号加法，使用无符号饱和处理溢出，结果存入 MRd 操作数中相应的位置。

执行周期：1 cycle

EPAND

句型： **EPAND** *MRd，MRs，MRt*

EPAND *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					100100					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp			ARm			gg		MRd			Modm					100100					

操作：

PAND：

MRd ← MRs AND MRt;

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PAND 对 128－bit MRs 操作数和 128－bit MRt 操作数，执行按位逻辑与运算，结果存入 MRd 操作数。

执行周期：1 cycle

EPAVGB/D

句型： **EPAVGB/D** *MRd，MRs，MRt*

EPAVGB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					111100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					111100						

操作：

EPAVGB instruction with 128-bit operands:

MRt[7-0] ← (MRt[7-0] + MRs[7-0] + 1] >> 1; * temp sum before shifting is 9 bits *

* repeat operation performed for bytes 2nd through 15th;

MRt[127-120] ← (MRt[127-120] + MRs[127-120] + 1) >> 1;

EPAVGD instruction with 128-bit operands:

MRt[15-0] ← (MRt[15-0] + MRs[15-0] + 1) >> 1; * temp sum before shifting is 17 bits *

* repeat operation performed for double-bytes 2 and 7;

MRt[127-112] ← (MRt[127-112] + MRs[127-112] + 1) >> 1;

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPAVGB/D 对 128－bit MRs 操作数中打包的字节数/2 字节数和 128－bit MRt 操作数中打包的字节数/2 字节数，执行 SIMD 加法，每个和值再加 1，相应结果右移 1bit 作为 2 个数的平均值，存入 MRd 操作数中。

执行周期：1 cycle

EPCMPEQB/D/Q

句型： **EPCMPEQB/D/Q** *MRd，MRs，MRt*

EPCMPEQB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					110100						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					110100						

操作：

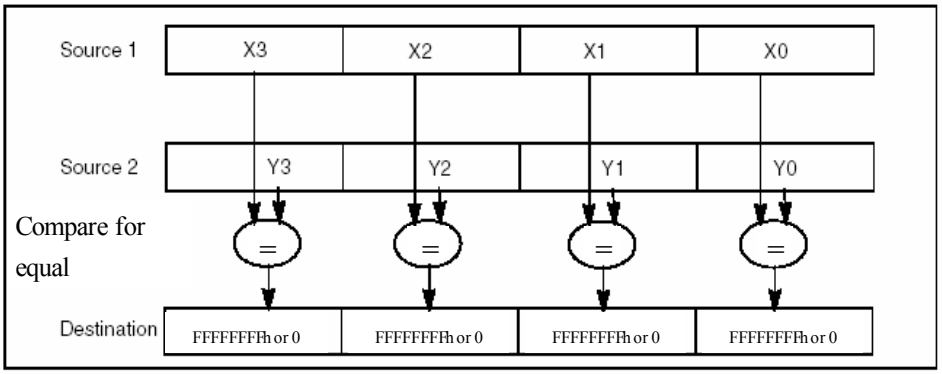
EPCMPEQB instruction with 128-bit operands:
IF MRs[7..0] = MRt[7..0]
THEN MRd[7 0] ← FFH;
ELSE MRd[7..0] ← 0;
* Continue comparison of 2nd through 15th bytes in MRd and MRt *
IF MRs[127..120] = MRt[127..120]
THEN MRd[127..120] ← FFH;
ELSE MRd[127..120] ← 0;

EPCMPEQD instruction with 128-bit operands:
IF MRs[15..0] = MRt[15..0]
THEN MRd[15..0] ← FFFFH;
ELSE MRd[15..0] ← 0;
* Continue comparison of 2nd and 7th double-bytes in MRd and MRt *
IF MRs[127..112] = MRt[127..112]
THEN MRd[127..112] ← FFFFH;
ELSE MRd[127..112] ← 0;

EPCMPEQQ instruction with 128-bit operands:
IF MRs[31..0] = MRt[31..0]
THEN MRd[31..0] ← FFFFFFFFH;
ELSE MRd[31..0] ← 0;
* Continue comparison of 2nd and 3rd 4-bytes in MRd and MRt *
IF MRs[127..96] = MRt[127..96]
THEN MRd[127..96] ← FFFFFFFFH;
ELSE MRd[127..96] ← 0;

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： EPCMPEQB/D/Q 对 128－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 128－bit MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 相等比较，如果相等结果全置 1，否则全置 0，结果存入 MRd 操作数中。下图示例了 EPCMPEQQ 的操作过程，其它类推。



执行周期：1 cycle

EPCMPGTB/D/Q

句型： **EPCMPGTB/D/Q** *MRd，MRs，MRt*

EPCMPGTB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					110000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					110000						

操作：

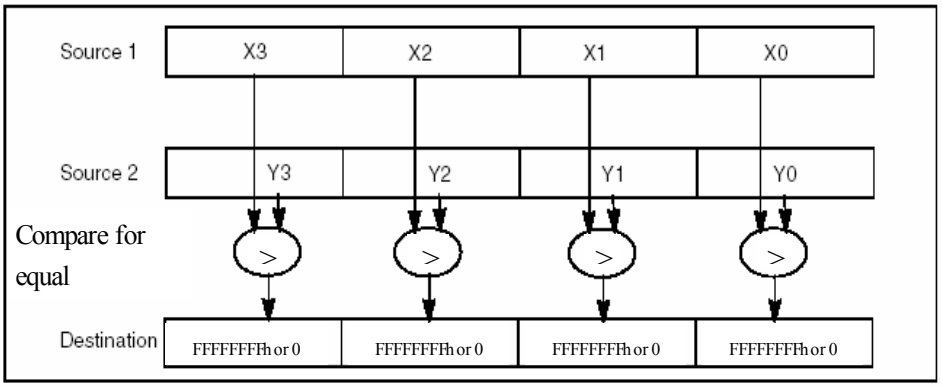
EPCMPGTB instruction with 128-bit operands:
IF MRs[7..0] >MRt[7..0]
THEN MRd[7 0] ← FFH;
ELSE MRd[7..0] ← 0;
* Continue comparison of 2nd through 15th bytes in MRd and MRt *
IF MRs[127..120] > MRt[127..120]
THEN MRd[127..120] ← FFH;
ELSE MRd[127..120] ← 0;

EPCMPGTD instruction with 128-bit operands:
IF MRs[15..0] > MRt[15..0]
THEN MRd[15..0] ← FFFFH;
ELSE MRd[15..0] ← 0;
* Continue comparison of 2nd and 7th double-bytes in MRd and MRt *
IF MRs[127..112] > MRt[127..112]
THEN MRd[127..112] ← FFFFH;
ELSE MRd[127..112] ← 0;

EPCMPGTQ instruction with 128-bit operands:
IF MRs[31..0] > MRt[31..0]
THEN MRd[31..0] ← FFFFFFFFH;
ELSE MRd[31..0] ← 0;
* Continue comparison of 2nd and 3rd 4-bytes in MRd and MRt *
IF MRs[127..96] > MRt[127..96]
THEN MRd[127..96] ← FFFFFFFFH;
ELSE MRd[127..96] ← 0;

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： EPCMPGTB/D/Q 对 128－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 128－bit MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 有符号比较，如果大于结果全置 1，否则全置 0，结果存入 MRd 操作数中。下图示例了 EPCMPGTQ 的操作过程，其它类推。



执行周期：1 cycle

EPLOADO

句型： **EPLOADO** *MRd， Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		000			disp		ARm			gg		MRd			Modm						110111					

操作：

EPLOADO：

MRd[127-0] ← memory

操作数说明： MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： EPLOADO 从 memory 中指定的位置读取 128bit 数据，写入到 MDS 寄存器 MRd 中。

执行周期：1 cycle

EPLOADOL

句型： **EPLOADOL** *MRd， Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		000			disp		ARm			gg		MRd			Modm						110101					

操作：

EPLOADOL：

Vaddr = Mod(ARm)，

MRd[127-0] ← Left (mem(Vaddr))

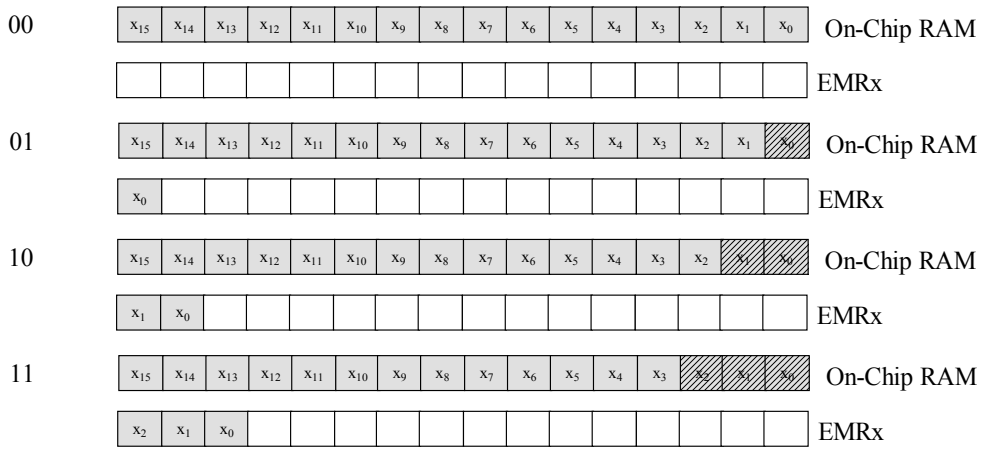
操作数说明： MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPLOADOL 从 memory 中指定的位置读取不大于 128bit 数据，左对齐写入到 MDS 寄存器 MRd 中。如图所示：（空白部分保持不变，每个方格表示一个字节）

(ARm[1:0])



执行周期：1 cycle

EPLOADOR

句型： **EPLOADOR** *MRd， Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		000			disp		ARm			gg		MRd			Modm						110110					

操作：

EPLOADOR：

Vaddr = Mod(ARm)，

MRd[127-0] ← Right (mem(Vaddr))

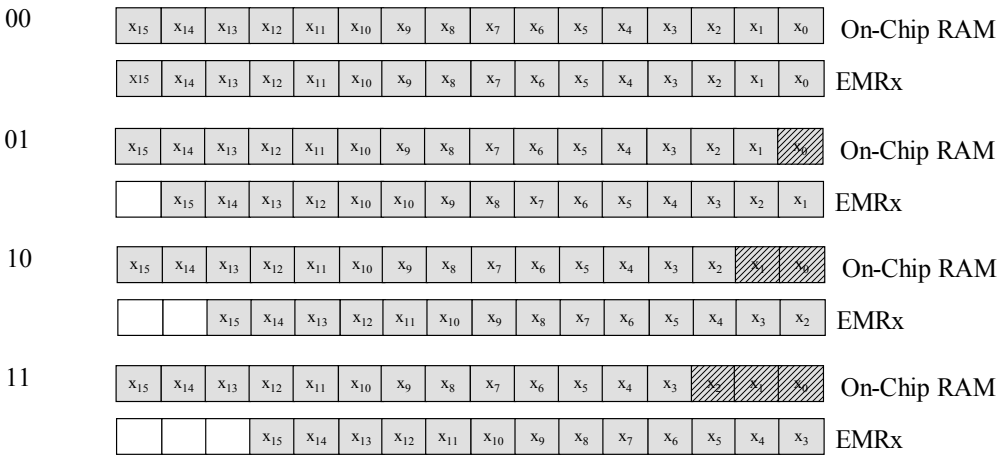
操作数说明： MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPLOADOR 从 memory 中指定的位置读取不大于 128bit 数据，右对齐写入到 MDS 寄存器 MRd 中。如图所示：（空白部分保持不变，每个方格表示一个字节）

(ARm[1:0])



执行周期：1 cycle

EPMADDQD

句型： **EPMADDQD** *MRd，MRs，MRt*

EPMADDQD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					101000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					101000						

操作：

EPMADDQD instruction with 128-bit operands:

MRd[31..0] ← (MRs[15..0]×MRt[15..0]) + (MRs[31..16]×MRt[31..16]);

* repeat operation performed for bytes 2nd through 3rd *;

MRd[127..96] ← (MRs[111..96]×MRt[111..96]) + (MRs[127..112]×MRt[127..112]);

* Signed multiplication *

描述： EPMADDQD 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 有符号乘法，然后相邻的 2 个 32－bit 结果相加成 1 个 32－bit 结果，最后结果存入 MRd 操作数。

执行周期：4 cycles

EPMAXSD

句型： **EPMAXSD** *MRd，MRs，MRt*

EPMAXSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					111000						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					111000						

操作：

EPMAXSD instruction for 128-bit operands:

IF MRs[15-0] > MRt[15-0]) THEN

(MRd[15-0] ← MRd[15-0];

ELSE

(MRd[15-0] ← MRt[15-0];

FI

* repeat operation for 2nd and 7th double-bytes in source and destination operands *

IF MRs[127-112] > MRt[127-112]) THEN

(MRd[127-112] ← MRd[127-112];

ELSE

(MRd[127-112] ← MRt[127-112];

FI

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPMAXSD 对 128－bit MRs 操作数中 8 个打包的有符号 2 字节数和 128－bit MRt 操作数中 8 个打包的有符号 2 字节数，执行 SIMD 有符号比较，相应较大的数存入 MRd 操作数中。

执行周期：1 cycle

EPMAXUB

句型： **EPMAXUB** *MRd，MRs，MRt*

EPMAXUB *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					111001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					111001						

操作：

EPMAXUB instruction for 128-bit operands:

IF MRs[7-0] > MRt[17-0]) THEN

(MRd[7-0] ← MRd[7-0];

ELSE

(MRd[7-0] ← MRt[7-0];

FI

* repeat operation for 2nd through 15th bytes in source and destination operands *

IF MRs[127-120] > MRt[127-120]) THEN

(MRd[127-120] ← MRd[127-120];

ELSE

(MRd[127-120] ← MRt[127-120];

FI

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPMAXUB 对 128－bit MRs 操作数中 16 个打包的无符号字节数和 128－bit MRt 操作数中 16 个打包的无符号字节数，执行 SIMD 比较，相应较大的数存入 MRd 操作数中。EPMAXUB 操作过程类似 EPMAXSD。

执行周期：1 cycle

EPMINSD

句型： **EPMINSD** *MRd，MRs，MRt*

EPMINSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					111010						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					111010						

操作：

EPMINSD instruction for 128-bit operands:

IF MRs[15-0] < MRt[15-0]) THEN

(MRd[15-0] ← MRd[15-0];

ELSE

(MRd[15-0] ← MRt[15-0];

IF

* repeat operation for 2nd and 7th double-bytes in source and destination operands *

IF MRs[127-112] < MRt[127-112]) THEN

(MRd[127-112] ← MRd[127-112];

ELSE

(MRd[127-112] ← MRt[127-112];

IF

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPMINSD 对 128－bit MRs 操作数中 8 个打包的有符号 2 字节数和 128－bit MRt 操作数中 8 个打包的有符号 2 字节数，执行 SIMD 有符号比较，相应较小的数存入 MRd 操作数中。

执行周期：1 cycle

EPMINUB

句型： **EPMINUB** *MRd，MRs，MRt*

EPMINUB *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					111011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					111011					

操作：

EPMINUB instruction for 128-bit operands:

IF MRs[7-0] < MRt[17-0]) THEN

MRd[7-0] ← MRd[7-0];

ELSE

MRd[7-0] ← MRt[7-0];

IF

* repeat operation for 2nd through 15th bytes in source and destination operands *

IF MRs[127-120] < MRt[127-120]) THEN

MRd[127-120] ← MRd[127-120];

ELSE

MRd[127-120] ← MRt[127-120];

IF

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPMINUB 对 128－bit MRs 操作数中 16 个打包的无符号字节数和 128－bit MRt 操作数中 16 个打包的无符号字节数，执行 SIMD 比较，相应较小的数存入 MRd 操作数中。EPMINUB 操作过程类似 EPMINSD。

执行周期：1 cycle

EPMULHSD

EPMACHSD

句型： **EPMULHSD** *MRd，MRs，MRt*
EPMACHSD *MRd，MRs，MRt*
EPMULHSD *MRd，MRs，Modm(ARm)*
EPMACHSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					011100/011110					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					011100/011110					

操作：

EPMULHSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
* Signed multiplication *
* repeat operation performed for 2-bytes 2nd through 7th *;
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← TEMP7[31-16];
* repeat operation performed for 2-bytes 2nd through 7th *;
MRd[127-112] ← TEMP7[31-16];

EPMACHSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
* Signed multiplication *
* repeat operation performed for 2-bytes 2nd through 7th *;
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← MRd[15-0] + TEMP0[31-16];
* repeat operation performed for 2-bytes 2nd through 7th *;
MRd[127-112] ← MRd[127-112] + TEMP7[31-16];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的高 16－bit 存入 MRd 操作数中相应的位置。EPMACHSD 将每次乘法结果不断累加。

执行周期：2 cycles

EPMULHUD

EPMACHUD

句型： **EPMULHUD** *MRd，MRs，MRt*
EPMACHUD *MRd，MRs，MRt*
EPMULHUD *MRd，MRs，Modm(ARm)*
EPMACHUD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					011101/011111					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					011101/011111					

操作：

EPMULHUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
　　* Unsigned multiplication *
　　* repeat operation performed for 2-bytes 2nd through 7th *；
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← TEMP0[31-16];
　　* repeat operation performed for 2-bytes 2nd through 7th *；
MRd[127-112] ← TEMP7[31-16];

EPMACHUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
* Unsigned multiplication *
　　* repeat operation performed for 2-bytes 2nd through 7th *；
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← MRd[15-0] + TEMP0[31-16];
　　* repeat operation performed for 2-bytes 2nd through 7th *；
MRd[127-112] ← MRd[127-112] + TEMP7[31-16];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的高 16－bit 存入 MRd 操作数中相应的位置。EPMACHUD 将每次乘法结果不断累加。

执行周期：2 cycles

EPMULLSD

EPMACLSD

句型： **EPMULLSD** *MRd，MRs，MRt*
EPMACLSD *MRd，MRs，MRt*
EPMULLSD *MRd，MRs，Modm(ARm)*
EPMACLSD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					011000/011010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					011000/011010					

操作：

EPMULLSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
　　* Signed multiplication *
　　* repeat operation performed for 2-bytes 2nd through 7th *;
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← TEMP0[15-0];
　　* repeat operation performed for 2-bytes 2nd through 7th *;
MRd[127-112] ← TEMP7[15-0];

EPMACLSD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
　　* Signed multiplication *
　　* repeat operation performed for 2-bytes 2nd through 7th *;
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← MRd[15-0] + TEMP0[15-0];
　　* repeat operation performed for 2-bytes 2nd through 7th *;
MRd[127-112] ← MRd[127-112] + TEMP7[15-0];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 有符号乘法，每个 32－bit 结果的低 16－bit 存入 MRd 操作数中相应的位置。EPMACLSD 将每次乘法结果不断累加。

执行周期：2 cycles

EPMULLUD

EPMACLUD

句型： **EPMULLUD** *MRd，MRs，MRt*
EPMACLUD *MRd，MRs，MRt*
EPMULLUD *MRd，MRs，Modm(ARm)*
EPMACLUD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					011001/011011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					011001/011011					

操作：

EPMULLUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
　　* Unsigned multiplication *
　　* repeat operation performed for 2-bytes 2nd through 7th *；
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← TEMP0[15-0];
　　* repeat operation performed for 2-bytes 2nd through 7th *；
MRd[127-112] ← TEMP7[15-0];

EPMACLUD:
TEMP0[31-0] ← MRs[15-0]×MRt[15-0];
　　* Unsigned multiplication *
　　* repeat operation performed for 2-bytes 2nd through 7th *；
TEMP7[31-0] ← MRs[127-112]×MRt[127-112];
MRd[15-0] ← MRd[15-0] + TEMP0[15-0];
　　* repeat operation performed for 2-bytes 2nd through 7th *；
MRd[127-112] ← MRd[127-112] + TEMP7[15-0];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 无符号乘法，每个 32－bit 结果的低 16－bit 存入 MRd 操作数中相应的位置。EPMACLUD 将每次乘法结果不断累加。

执行周期：2 cycles

EPNOR

句型： **EPNOR** *MRd，MRs，MRt*

EPNOR *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			11		MRd			00000					100011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			11		MRd			Modm					100011					

操作：

EPNOR：

MRd ← MRd NOR MRt;

描述： EPNOR 对 128－bit MRs 操作数和 128－bit MRt 操作数，执行按位逻辑或非运算，结果存入 MRd 操作数。

执行周期：1 cycle

EPOR

句型： **EPOR** *MRd，MRs，MRt*

EPOR *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			11		MRd			00000					100101					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			11		MRd			Modm					100101					

操作：

EPOR：

MRd ← MRd OR MRt;

描述： EPOR 对 128－bit MRd 操作数和 128－bit MRt 操作数，执行按位逻辑或运算，结果存入 MRd 操作数。

执行周期：1 cycle

EPSADBD

句型： **EPSADBD** *MRd，MRs，MRt*

EPSADBD *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			11		MRd			00000					101001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			11		MRd			Modm					101001						

操作：

EPSADBD instructions when using 128-bit operands:

TEMP0 ← ABS(MRs[7-0] － MRt[7-0]);

* repeat operation for bytes 2 through 15 *;

TEMP15 ← ABS(MRs[127-120] － MRt[127-120]);

MRd[23:0] ← SUM(TEMP0…TEMP15);

MRd[127:24] ← 0;

描述： EPSADBD 对 128－bit MRs 操作数中 16 个打包的字节数和 128－bit MRt 操作数中 16 个打包的字节数，执行 SIMD 减法，取绝对值得到绝对差值，然后 16 个绝对差值相加成 1 个 24－bit 无符号数，存入 MRd 操作数的低 24－bit，MRd 操作数的高 104bit 置 0。

执行周期：3 cycles

EPSHUFQ

句型： **EPSHUFQ** *MRd，MRs，imm*

EPSHUFQ *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000001					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					000001					

操作：

EPSHUFQ:

IF (ORDER = 0)

THEN MRd[31:0]← MRs[31:0];

IF (ORDER = 1)

THEN MRd[31:0] ← MRs[63:32];

IF (ORDER = 2)

THEN MRd[31:0] ← MRs[95:128];

IF (ORDER = 3)

THEN MRd[31:0] ← MRs[127:96];

IF;

*Repeat operation for 2nd, 3rd, 4rdquad-bytes;

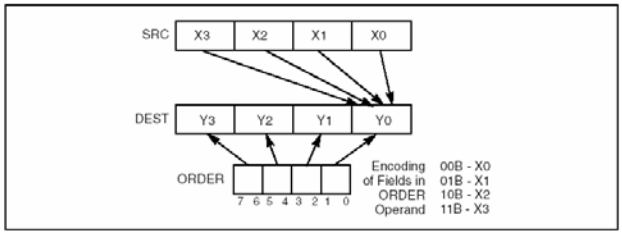
操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

Sa： 立即数

描述： *MRs* 中打包的 4 字节数据作换位排列，换位控制来自 Sa/MRt 的最低 8bit。MRd 中每个 32bit 数据取自 MRs 中 4 个 32bit 数据的其中一个。这样，每个结果 32bit 数据的产生需要 2bit 选择信号，总共需要 8bit 选择信号，由立即数 sa 的最低 8 个 bit 表示（高 2bit 汇编时置 0）。



图中寄存器都是 128 比特的，所以X_i,Y_i都是 32 比特数。

执行周期：1 cycle

EPSHUFLD

句型： **EPSHUFLD** *MRd，MRs，imm*

EPSHUFLD *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000101					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00			MRt		gg		MRd			00000					000101					

操作：

EPSHUFLD:

MRd[15-0] <– (MRs >> (ORDER[1-0] * 16))[15-0]

MRd[31-16] <– (MRs >> (ORDER[3-2] * 16))[15-0]

MRd[47-32] <– (MRs>> (ORDER[5-4] * 16))[15-0]

MRd[63-48] <– (MRs>> (ORDER[7-6] * 16))[15-0]

MRd[127-128] <– MRs[127-128]

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

Sa： 立即数

描述： *MRs* 中低 128 位做打包的 2 字节数据作换位排列，换位控制来自 Sa/MRt 的最低 8bit。MRd 中低 128 位每个 16bit 数据取自 MRs 中低 128 位 4 个 16bit 数据的其中一个。这样，每个结果 16bit 数据的产生需要 2bit 选择信号，总共需要 8bit 选择信号，由立即数 sa 的最低 8 个 bit 表示（高 2bit 汇编时置 0）。MRd 中高 128 位数据直接复制 MRs 中高 128 位数据。

执行周期：1 cycle

EPSHUFHD

句型： **EPSHUFHD** *MRd，MRs，imm*

EPSHUFHD *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000110					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00			MRt		gg		MRd			00000					000110					

操作：

EPSHUFHD:

MRd[63-0] <– MRs[63-0]

MRd[79-128] <– (MRs >> (ORDER[1-0] * 16))[15-0]

MRd[95-80] <– (MRs >> (ORDER[3-2] * 16))[15-0]

MRd[111-96] <– (MRs >> (ORDER[5-4] * 16))[15-0]

MRd[127-112] <– (MRs >> (ORDER[7-6] * 16))[15-0]

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

Sa： 立即数

描述： *MRs* 中高 128 位做打包的 2 字节数据作换位排列，换位控制来自 Sa/MRt 的最低 8bit。MRd 中高 128 位每个 16bit 数据取自 MRs 中高 128 位 4 个 16bit 数据的其中一个。这样，每个结果 16bit 数据的产生需要 2bit 选择信号，总共需要 8bit 选择信号，由立即数 sa 的最低 8 个 bit 表示（高 2bit 汇编时置 0）。MRd 中低 128 位数据直接复制 MRs 中低 128 位数据。

执行周期：1 cycle

EPSRAD/Q

句型： **EPSRAD/Q** *MRd，MRs，imm*

EPSRAD/Q *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					000011					

操作：

EPSRAD:

IF (COUNT > 15)

THEN COUNT ← 16;

MRd[15..0] ← SignExtend(MRs[15..0] >> COUNT);

* repeat shift operation for 2nd and 7th double-bytes *;

MRd[127..112] ← SignExtend(MRs[127..112] >> COUNT);

EPSRAQ:

IF (COUNT > 31)

THEN COUNT ← 32;

MRd[31..0] ← SignExtend(MRs[31..0] >> COUNT);

* repeat shift operation for 2nd and 3rd 4-bytes *;

MRd[127..96] ← SignExtend(MRs[127..96] >> COUNT);

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

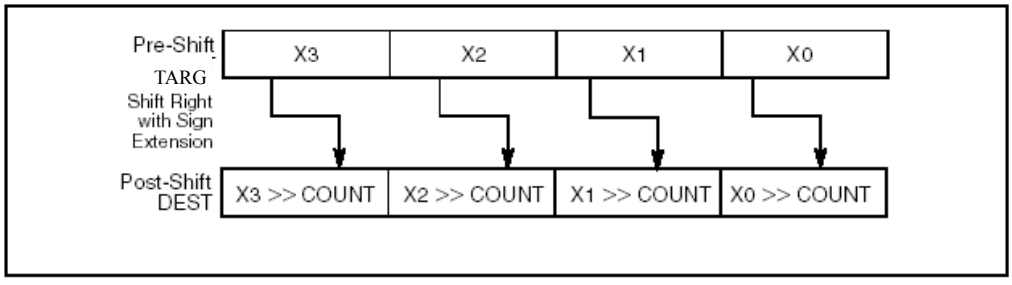
MRd： MDS 寄存器

Sa： 立即数

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 128bit MRs 操作数中打包的 2 字节/4 字节进行 SIMD 算术右移，结果存入 MRd 操作数。下图示例了 EPSRAQ 的操作过程，其它类推。



执行周期：1 cycle

EPSRLD/Q

句型： **EPSRLD/Q/O** *MRd，MRs，imm*

EPSRLD/Q/O *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00			MRt		gg		MRd			00000					000010					

操作：

EPSRLD:

```
IF (COUNT > 15)
THEN
        MRd[128..0] ← 0
ELSE
        MRd[15..0] ← ZeroExtend(MRs[15..0] >> COUNT);
            * repeat shift operation for 2nd and 7th double-bytes *;
        MRd[127..112] ← ZeroExtend(MRs[127..112] >> COUNT);
```

EPSRLQ:

```
IF (COUNT > 31)
THEN
        MRd[128..0] ← 0
ELSE
        MRd[31..0] ← ZeroExtend(MRs[31..0] >> COUNT);
            * repeat shift operation for 2nd and 3rd   4-bytes *;
        MRd[127..96] ← ZeroExtend(MRd[127..96] >> COUNT);
```

操作数说明： MRs： MDS寄存器

MRt： MDS 寄存器

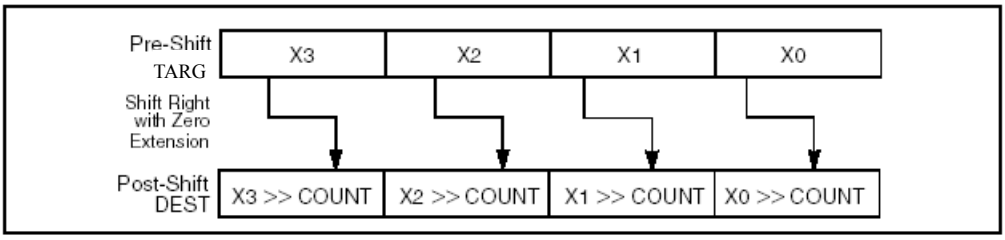
MRd： MDS 寄存器

Sa： 立即数

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 128bit MRs 操作数中打包的 2 字节/4 字节进行 SIMD 逻辑右移，结果存入 MRd 操作数。下图示例了 EPSRLQ 的操作过程，其它类推。



执行周期： **1** cycle

EPSLLD/Q

句型： **EPSLLD/Q/O** *MRd，MRs，imm*

EPSLLD/Q/O *MRd，MRs，MRt*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						00		MRs			sa[9:5]					gg		MRd			sa[4:0]					000000					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111						01		MRs			00		MRt			gg		MRd			00000					000000					

操作：

EPSLLD:

```
IF (COUNT > 15)
THEN
        MRd[128..0] ← 0
ELSE
        MRd[15..0] ← ZeroExtend(MRs[15..0] << COUNT);
            * repeat shift operation for 2nd and 7th double-bytes *;
        MRd[127..112] ← ZeroExtend(MRs[127..112] << COUNT);
```

EPSLLQ:

```
IF (COUNT > 31)
THEN
        MRd[128..0] ← 0
ELSE
        MRd[31..0] ← ZeroExtend(MRs[31..0] << COUNT);
            * repeat shift operation for 2nd and 3rd   4-bytes *;
        MRd[127..96] ← ZeroExtend(MRd[127..96] << COUNT);
```

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

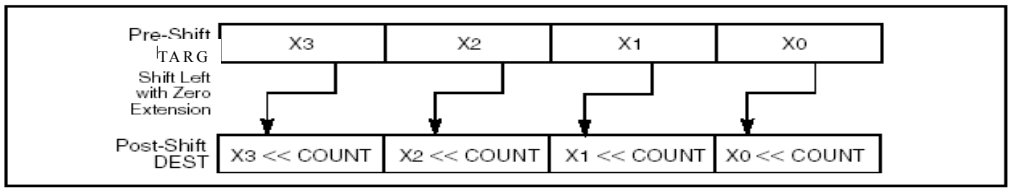
MRd： MDS 寄存器

Sa： 立即数

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： 对 128bit MRs 操作数中打包的 2 字节/4 字节进行 SIMD 逻辑左移，结果存入 MRd 操作数。下图示例了 EPSLLQ 的操作过程，其它类推。



执行周期：1 cycle

EPSTOREO

句型： **EPSTOREO** *MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			11		000			Modm					111111					

操作：

EPSTOREO：

Memory ← MRs[127-0]

操作数说明： MRs： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： PSTOREO 将 MDS 寄存器 MRs 中的 128bit 数据写入到 memory 中指定的位置。

执行周期：1 cycle

EPSUBB/D/Q

句型： **EPSUBB/D/Q** *MRd，MRs，MRt*

EPSUBB/D/Q *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					101110						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					101110						

操作：

EPSUBB instruction with 128-bit operands:

MRd[7..0] ← MRs[7..0] − MRt[7..0];

* repeat sub operation for 2nd through 15th byte *;

MRd[127..120] ← MRs[127..120] − MRt[127..120];

EPSUBD instruction with 128-bit operands:

MRd[15..0] ← MRs[15..0] − MRt[15..0];

* repeat add operation for 2nd and 7th double-byte *;

MRd[127..112] ← MRs[127..112] − MRt[127..112];

EPSUBQ instruction with 128-bit operands:

MRd[31..0] ← MRs[31..0] − MRt[31..0];

* repeat add operation for 2nd and 3rd double-byte *;

MRd[127..96] ← MRs[127..96] − MRt[127..96];

操作数说明： MRs： MDS 寄存器

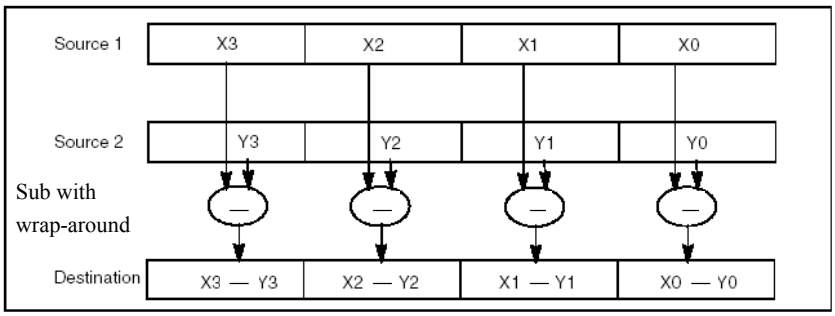
MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPSUBB/D/Q 对 128－bit MRs 操作数中打包的字节数/2 字节数/4 字节数和 128－bit MRt 操作数中打包的字节数/2 字节数/4 字节数，执行 SIMD 减法，结果存入 MRd 操作数中，溢出被忽略。下图示例 EPSUBQ 的操作过程，其它类推。



执行周期：1 cycle

EPSUBSB/D

句型： **EPSUBSB/D** *MRd，MRs，MRt*

EPSUBSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					100001						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					100001						

操作：

EPSUBSB instruction with 128-bit operands:

MRd[7..0] ← SaturateToSignedByte(MRs[7..0] － MRt (7..0)) ;

* repeat sub operation for 2nd through 15th bytes *;

MRd[127..120] ← SaturateToSignedByte(MRs[127..120] － MRt[127..120]);

EPSUBSD instruction with 128-bit operands:

MRd[15..0] ← SaturateToSignedDouble-byte(MRs[15..0] － MRt[15..0]);

* repeat sub operation for 2nd and 7th double-bytes *;

MRd[127..112] ← SaturateToSignedDouble-byte(MRs[127..112] － MRt[127..112]);

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPSUBSB 对 128－bit MRs 操作数中 16 个打包的字节数和 128－bit MRt 操作数中 16 个打包的字节数，执行 SIMD 有符号减法，结果存入 MRd 操作数中相应的位置。

EPSUBSD 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 有符号减法，结果存入 MRd 操作数中相应的位置。

执行周期：1 cycle

EPSUBUSB/D

句型： **EPSUBUSB/D** *MRd，MRs，MRt*

EPSUBUSB/D *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					100011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					100011					

操作：

EPSUBUSB instruction with 128-bit operands:

MRd[7..0] ← SaturateToUnsignedByte(MRs[7..0] － MRt (7..0)) ;

* repeat sub operation for 2nd through 15th bytes *;

MRd[127..120] ← SaturateToUnsignedByte(MRs[127..120] － MRt[127..120]);

EPSUBUSD instruction with 128-bit operands:

MRd[15..0] ← SaturateToUnsignedDouble-byte(MRs[15..0] － MRt[15..0]);

* repeat sub operation for 2nd and 7th double-bytes *;

MRd[127..112] ← SaturateToUnsignedDouble-byte(MRs[127..112] － MRt[127..112]);

操作数说明： MRs： MDS 寄存器

MRt： MDS 寄存器

MRd： MDS 寄存器

ARm： 间接寻址辅助寄存器

Disp： 地址偏移立即数

描述： EPSUBUSB 对 128－bit MRs 操作数中 16 个打包的字节数和 128－bit MRt 操作数中 16 个打包的字节数，执行 SIMD 无符号减法，结果存入 MRd 操作数中相应的位置。EPSUBUSD 对 128－bit MRs 操作数中 8 个打包的 2 字节数和 128－bit MRt 操作数中 8 个打包的 2 字节数，执行 SIMD 无符号减法，结果存入 MRd 操作数中相应的位置。

执行周期：1 cycle

EPUNPCKHBD/DQ/QO

句型： **EPUNPCKHBD/DQ/QO** *MRd，MRs，MRt*

EPUNPCKHBD/DQ/QO *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					001010					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					001010					

操作：

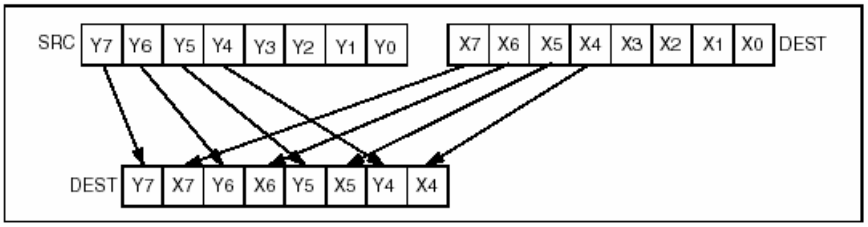
EPUNPCKHBD:
MRd[7..0] ←MRs[71..64];
MRd[15..8] ←MRt[71..64];
MRd[23..16] ←MRs[79..72];
MRd[31..24] ←MRt[79..72];
MRd[39..32] ←MRs[87..80];
MRd[47..40] ←MRt[87..80];
MRd[55..48] ←MRs[95..88];
MRd[63..56] ←MRt[95..88];
MRd[71..64] ←MRs[103..96];
MRd[79..72] ←MRt[103..96];
MRd[87..80] ←MRs[111..104];
MRd[95..88] ←MRt[111..104];
MRd[103..96] ←MRs[119..112];
MRd[111..104] ←MRt[119..112];
MRd[119..112] ←MRs[127..120];
MRd[127..120] ←MRt[127..120];

EPUNPCKHDQ:
MRd[15..0] ←MRs[79..64];
MRd[31..16] ←MRt[79..64];
MRd[47..32] ←MRs[95..80];
MRd[63..48] ←MRt[95..80];
MRd[79..64] ←MRs[111..96];
MRd[95..80] ←MRt[111..96];
MRd[111..96] ←MRs[127..112];
MRd[127..112] ←MRt[127..112];

EPUNPCKHQO:
MRd[31..0] ←MRs[95..64]
MRd[63..32] ←MRt[95..64];
MRd[95..64] ←MRs[127..96]
MRd[127..96] ←MRt[127..96];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 将 128bit MRt 和 MRs 操作数中打包的字节/2 字节/4 字节相交织，取高 128bit 存入 MRd 操作数。下图示例了 EPUNPCKHDQ 的操作过程，其它类推。



执行周期：1 cycle

EPUNPCKLBD/DQ/QO

句型： **EPUNPCKLBD/DQ/QO** *MRd，MRs，MRt*

EPUNPCKLBD/DQ/QO *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					001011					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					001011					

操作：

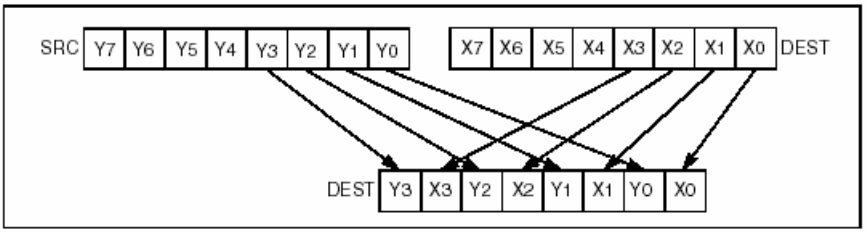
EPUNPCKLBD:
MRd[7..0] ←MRs[7..0];
MRd[15..8] ←MRt[7..0];
MRd[23..16] ←MRs[15..8];
MRd[31..24] ←MRt[15..8];
MRd[39..32] ←MRs[23..16];
MRd[47..40] ←MRt[23..16];
MRd[55..48] ←MRs[31..24];
MRd[63..56] ←MRt[31..24];
MRd[71..64] ←MRs[39..32];
MRd[79..72] ←MRt[39..32];
MRd[87..80] ←MRs[47..40];
MRd[95..88] ←MRt[47..40];
MRd[103..96] ←MRs[55..48];
MRd[111..104] ←MRt[55..48];
MRd[119..112] ←MRs[63..56];
MRd[127..120] ←MRt[63..56];

EPUNPCKLDQ:
MRd[15..0] ←MRs15..0];
MRd[31..16] ←MRt[15..0];
MRd[47..32] ←MRs[31..16];
MRd[63..48] ←MRt[31..16];
MRd[79..64] ←MRs[47..32];
MRd[95..80] ←MRt[47..32];
MRd[111..96] ←MRs[63..48];
MRd[127..112] ←MRt[63..48];

EPUNPCKLQO:
MRd[31..0] ←MRs[31..0]
MRd[63..32] ←MRt[31..0];
MRd[95..64] ←MRs[63..32]
MRd[127..96] ←MRt[63..32];

操作数说明： MRs： MDS 寄存器
MRt： MDS 寄存器
MRd： MDS 寄存器
ARm： 间接寻址辅助寄存器
Disp： 地址偏移立即数

描述： 将 128bit MRt 和 MRs 操作数中打包的字节/2 字节/4 字节相交织，取低 128bit 存入 MRd 操作数。下图示例了 EPUNPCKLDQ 的操作过程，其它类推。



执行周期：1 cycle

EPXOR

句型： **EPXOR** *MRd，MRs，MRt*

EPXOR *MRd，MRs，Modm(ARm)*

指令编码：

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						01		MRs			00		MRt			gg		MRd			00000					100110					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111100						11		MRs			disp		ARm			gg		MRd			Modm					100110					

操作：

EPXOR：

MRd ← MRs XOR MRt;

描述： EPXOR 对 128－bit MRs 操作数和 128－bit MRt 操作数，执行按位逻辑异或运算，结果存入 MRd 操作数。

执行周期：1 cycle