

文件编号： ZDMC/MD64-002

项目名称  
MediaDSP3200/6400 IP Core  
MediaDSP32/64 ISA Manual (Draft)



编写人：浙江大学信息学院 SoC 研发小组

编写时间：2007 年 5 月

审核人：刘鹏

审核时间：2007 年 5 月

---

<b>MediaDSP32/64 ISA Manual (Draft)</b> .....	0
<b>MediaDSP32/64 ISA Manual</b> .....	2
一、 引言 .....	2
二、 MediaDSP32/64 指令格式 .....	3
NOTES .....	210

# MediaDSP32/64 ISA Manual

May 2007

---

主要内容

- 引言
  - MediaDSP32/MediaDSP64 指令格式
- 

## 一、引言

### 1. 编写目的:

提供给 MediaDSP3200 和 MediaDSP6400 IP core 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 和 MediaDSP6400 处理器设计人员和软件程序员。

### 3. 文档背景:

MediaDSP3200 和 MediaDSP6400 处理器 ISA 设计。

### 4. 参考文献:

- [1] 超大规模集成电路 SOC 重大专项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、MediaDSP32/64 指令格式

MediaDSP32/64(简称 MD32/64)的指令类型可分为 MDF, MDD, MDS, EMS 四种。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; EMS 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。MediaDSP ISA II 包括 200 条指令, MediaDSP ISA I 包括 159 条指令。

### MDF 指令列表 (60 条指令)

表 2.1 MD-32/64 ISA 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-ISA 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（43 条指令）

操作数位宽为 64bit.

表 2.3 MD-ISA MDS 指令一览表

MediaDSP ISA MDS 指令	指令功能描述
数据传输指令(6)	
PMTHI, PMTLO	传输 4-字节 ( $R_s$ 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 $R_s$ ) 从 MDS 寄存器高/低端到通用寄存器
<i>PLOADO</i>	传输 8-字节 ( $mem$ 到 $MRd$ ) 从 memory 到 MDS 寄存器
<i>PSTOREO</i>	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



MediaDSP ISA MDS 指令	指令功能描述
PAVGB/D	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 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	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑与
POR	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑或
PXOR	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑异或
PNOR	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑或非
移位指令(3)	
PSLLD/Q/O	<i>MRs</i> 中打包的 2 字节/4 字节数据进行 SIMD 逻辑左移, 移位量来自 <i>MRt(imm)</i> 的最低 5bit
PSRLD/Q/O	<i>MRs</i> 中打包的 2 字节/4 字节数据进行 SIMD 逻辑右移, 移位量来自 <i>MRt(imm)</i> 的最低 5bit
PSRAD/Q	<i>MRs</i> 中打包的 2 字节数据进行 SIMD 算术右移, 移位量来自 <i>MRt(imm)</i> 的最低 5bit
比特流操作指令 (2)	
GETBITS	从码流中取出 n 个比特, 并且码流指针也相应的偏移 n 个比特
SHOWBITS	从码流中取出 n 个比特, 但是码流的指针并不改变

## EMS 指令列表（41 条指令，支持 MediaDSP6400，MediaDSP3200 不支持）

操作数位宽为 128bit.

表 2.4 MD ISA EMS 指令一览表

MediaDSP ISA EMS 指令	指令功能描述
数据传输指令(4)	
<i>EPLOADO</i>	传输 16-字节 ( <i>mem</i> 到 <i>MRd</i> ) 从 memory 到 MDS 寄存器
<i>EPLOADOL</i>	传输 16-字节( <i>mem</i> 到 <i>MRd</i> )数据左对齐写入到 MDS 寄存器
<i>EPLOADOR</i>	传输 16-字节( <i>mem</i> 到 <i>MRd</i> )数据右对齐写入到 MDS 寄存器
<i>EPSTOREO</i>	传输 16-字节 ( <i>MRd</i> 到 <i>mem</i> ) 从 MDS 寄存器到 memory
转换指令(2)	
<i>EPACKSSDB/QD</i>	将 <i>MRt</i> ( <i>mem</i> ) 和 <i>MRs</i> 操作数中打包的 4-字节/8-字节数据转换为 2-字节/4-字节数据, 使用有符号饱和和处理溢出
<i>EPACKUSDB/QD</i>	将 <i>MRt</i> ( <i>mem</i> ) 和 <i>MRs</i> 操作数中打包的 4-字节数据转换为 2-字节数据, 使用无符号饱和和处理溢出
解包指令(2)	
<i>EPUNPCKHBD/DQ/QO</i>	将 <i>MRt</i> ( <i>mem</i> ) 和 <i>MRs</i> 操作数中打包的 1-字节/2-字节/4-字节相交织, 取高 128-bit 存入 <i>MRd</i> 操作数
<i>EPUNPCKLBD/DQ/QO</i>	将 <i>MRt</i> ( <i>mem</i> ) 和 <i>MRs</i> 操作数中打包的 1-字节/2-字节/4-字节相交织, 取低 128-bit 存入 <i>MRd</i> 操作数
算术指令(21)	
<i>EPADDB/D/Q</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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
<i>EPADDSB/D</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 操作数中打包的-1 字节/2-字节数据执行 SIMD 加法, 使用有符号饱和和处理溢出
<i>EPADDUSB/D</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 操作数中打包的 1-字节/2-字节数据执行 SIMD 加法, 使用无符号饱和和处理溢出
<i>EPSUBB/D/Q</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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
<i>EPSUBSB/D</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 操作数中打包的 1-字节/2-字节数据执行 SIMD 减法, 使用有符号饱和和处理溢出
<i>EPSUBUSB/D</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 操作数中打包的 1-字节/2-字节数据执行 SIMD 减法, 使用无符号饱和和处理溢出
<i>EPMULLSD</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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
<i>EPMULHSD</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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。
<i>EPMACHSD</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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, 将每次乘法结果不断累加
<i>EPMACLSD</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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, 将每次乘法结果不断累加
<i>EPMULLUD</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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
<i>EPMULHUD</i>	<i>MRs</i> 和 <i>MRt</i> ( <i>mem</i> ) 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

MediaDSP ISA EMS 指令	指令功能描述
PMACLUUD	<i>MRs</i> 和 <i>MRt(mem)</i> 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, PMACLUUD将每次乘法结果不断累加
PMACHUD	<i>MRs</i> 和 <i>MRt(mem)</i> 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
EPMADDQD	<i>MRs</i> 和 <i>MRt(mem)</i> 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
EPAVGB/D	<i>MRs</i> 和 <i>MRt(mem)</i> 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
EPMAXUB	<i>MRs</i> 和 <i>MRt(mem)</i> 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
EPMAXSD	<i>MRs</i> 和 <i>MRt(mem)</i> 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
EPMINUB	<i>MRs</i> 和 <i>MRt(mem)</i> 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
EPMINSD	<i>MRs</i> 和 <i>MRt(mem)</i> 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
EPSADBD	<i>MRs</i> 和 <i>MRt(mem)</i> 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 16 个绝对差值相加
比较指令(2)	
EPCMPEQB/D/Q	<i>MRs</i> 和 <i>MRt(mem)</i> 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。
EPCMPGTB/D/Q	<i>MRs</i> 和 <i>MRt(mem)</i> 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)	
EPAND	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑与
EPOR	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑或
EPXOR	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑异或
EPNOR	<i>MRs</i> 和 <i>MRt(mem)</i> 按位逻辑或非
移位指令(6)	
EPSLLD/Q	<i>MRs</i> 中打包的 2 字节/4 字节数据进行 SIMD 逻辑左移, 移位量来自 <i>MRt(imm)</i> 的最低 5bit
EPSRLD/Q	<i>MRs</i> 中打包的 2 字节/4 字节数据进行 SIMD 逻辑右移, 移位量来自 <i>MRt(imm)</i> 的最低 5bit
EPSRAD/Q	<i>MRs</i> 中打包的 2 字节数据进行 SIMD 算术右移, 移位量来自 <i>MRt(imm)</i> 的最低 5bit
EPSHUFQ	<i>MRs</i> 中打包的 4 字节数据作换位排列
EPSHUFLLD	<i>MRs</i> 中低 128 位做打包的 2 字节数据作换位排列, 换位控制来自 <i>Sa/MRt</i> 的最低 8bit. <i>MRd</i> 中低 128 位每个 16bit 数据取自 <i>MRs</i> 中低 128 位 4 个 16bit 数据的其中一个.
EPSHUFHLD	<i>MRs</i> 中高 128 位做打包的 2 字节数据作换位排列, 换位控制来自 <i>Sa/MRt</i> 的最低 8bit. <i>MRd</i> 中高 128 位每个 16bit 数据取自 <i>MRs</i> 中高 128 位 4 个 16bit 数据的其中一个.

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

**GRF 寄存器**为 32x32bit.

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	

**MRF 寄存器**为 8x64bit. (MediaDSP3200 支持)

寄存器名称	寄存器序列	说明
MR0	0	可用做 MUL 的 HI, LO 寄存器
MR1	1	
MR2	2	
MR3	3	
MR4	4	
MR5	5	
MR6	6	
MR7	7	

**ERF 寄存器**为 8x128bit.(MediaDSP6400 支持)

寄存器名称	寄存器序列	说明
ER0	0	
ER1	1	
ER2	2	
ER3	3	
ER4	4	
ER5	5	
ER6	6	
ER7	7	

## (a) 带偏移量的间接寻址

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

## (e) MDS/EMS 指令 Modm 位寻址模式

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

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

## MediaDSP 处理器指令格式

- 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						Displ	ARm	Displ	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

- MDS 指令格式:

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															

- EMS 指令格式:

EMS-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
111100					00	MRs	sa[9:5]				gg	MRd	sa[4:0]				func_code														

EMS-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
111100					01	MRs	00	MRt	gg	MRd	00000				func_code																

EMS-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
111100					10	MRs	Rs				gg	MRd	00000				func_code														

EMS-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
111100					11	MRs	disp	ARm	gg	MRd	Modm				func_code																

**a) MDS 指令编码**

功能码	0	1	2	3	4	5	6	7
0	PSLL D/Q/O	PSHUFDB	PSRL D/Q/O	PSRA D/Q				
1			PUNPCKH BD/DQ/QO	PUNPCKL BD/DQ/QO				GETBITS/ SHOWBITS
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

**b) EMS 指令编码**

功能码	0	1	2	3	4	5	6	7
0	EPSLL D/Q/O	EPSHUFDB	EPSRL D/Q/O	EPSRA D/Q		EPSHUFLD	EPSHUFHD	
1			EPUNPCKH BD/DQ/QO	EPUNPCKL BD/DQ/QO				
2					EPACKSS DB/QD	EPACKUS DB/QD		
3	EPMULLSD	EPMULLUD	EPMACLSD	EPMACLUD	EPMULHSD	EPMULHUD	EPMACHSD	EPMACHUD
4	EPADDS B/D	EPADDUS B/D	EPSUBS B/D	EPSUBUS B/D	EPAND	EPOR	EPXOR	EPNOR
5	EPMADDQD	EPSADBD			EPADD B/D/Q		EPSUB B/D/Q	
6	EPCMPGT B/D/Q				EPCMPEQ B/D/Q	EPLOADOL	EPLOADOR	EPLOADO
7	EPMAXSD	EPMAXUB	EPMINSD	EPMINUB	EPAVG B/D			EPSTOREO

## 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:  $|\text{modm}(\text{ARm})| \rightarrow \text{GPR}(\text{dst}) \quad || \quad \text{GPR}(\text{src2}) \rightarrow \text{modn}(\text{ARn})$

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:  $|\text{mem}(*\text{AR0--}(\text{IR0}))| \rightarrow \text{GPR}(\text{R2}),$

$\text{GPR}(\text{R3}) \rightarrow \text{mem}(*\text{+AR7}(\text{IR1}))$

## ADD

句型:	<b>ADD</b>	rd, rs, rt	或者
	<b>ADD</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>ADD</b>	dst, mod(ARm), rt	或者
	<b>ADD</b>	dst, *+ARm(displ), Imm	或者
	<b>ADD</b>	dst, rs, mod(ARm)	或者
	<b>ADD</b>	dst, rs, *+ARm(displ)	或者
	<b>ADD</b>	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						Displ		ARm		Displ		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(displ)) + mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) + GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) + sign(Imm) → GPR(dst)	或者
	GPR(Rs) + mod(ARm) → GPR(dst)	或者
	GPR(Rs) + mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00 间接寻址	寄存器
	E=01 *+ARn(displ)寻址	立即数
10	E=00 寄存器	间接寻址
	E=01 寄存器	*+ARn(displ)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:

$\text{sign}(\text{Imm}) + \text{GPR}(\text{Rs}) \rightarrow \text{GPR}(\text{rt})$        或者  
 $\text{mem}(\text{Imm}) + \text{GPR}(\text{dst}) \rightarrow \text{GPR}(\text{dst})$        或者  
 $\text{modm}(\text{ARm}) + \text{GPR}(\text{dst}) \rightarrow \text{GPR}(\text{dst})$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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	$\text{sign}(0840\text{h}) + \text{GPR}(\text{R3}) \rightarrow \text{GPR}(\text{R5})$
ADDI R5, @0840h	$\text{Mem}(0840\text{h}) + \text{GPR}(\text{R5}) \rightarrow \text{GPR}(\text{R5})$
ADDI R5, *AR2++(40h)	$\text{Mem}(*\text{AR2}++(40\text{h})) + \text{GPR}(\text{R5}) \rightarrow \text{GPR}(\text{R5})$



## 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：

$\text{sign}(\text{Imm}) + \text{GPR}(\text{Rs}) \rightarrow \text{GPR}(\text{rt})$  或者  
 $\text{mem}(\text{Imm}) + \text{GPR}(\text{dst}) \rightarrow \text{GPR}(\text{dst})$  或者  
 $\text{modm}(\text{ARm}) + \text{GPR}(\text{dst}) \rightarrow \text{GPR}(\text{dst})$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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	$\text{sign}(0840\text{h}) + \text{GPR}(\text{R3}) \rightarrow \text{GPR}(\text{R5})$
ADDIU R5, @0840h	$\text{Mem}(0840\text{h}) + \text{GPR}(\text{R5}) \rightarrow \text{GPR}(\text{R5})$
ADDIU R5, *AR2++(40h)	$\text{Mem}(*\text{AR2}++(40\text{h})) + \text{GPR}(\text{R5}) \rightarrow \text{GPR}(\text{R5})$

## ADDU

句型:	<b>ADDU</b>	rd, rs, rt	或者
	<b>ADDU</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>ADDU</b>	dst, mod(ARm), rt	或者
	<b>ADDU</b>	dst, *+ARm(displ), Imm	或者
	<b>ADDU</b>	dst, rs, mod(ARm)	或者
	<b>ADDU</b>	dst, rs, *+ARm(displ)	或者
	<b>ADDU</b>	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						Displ		ARm		Displ		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(displ)) + mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) + GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) + sign(Imm) → GPR(dst)	或者
	GPR(Rs) + mod(ARm) → GPR(dst)	或者
	GPR(Rs) + mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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	$GPR(R3) + GPR(R7) \rightarrow GPR(R5)$
ADDU R5, *+AR1(1h), *+AR2(8h)	$Mem(*+AR1(1h)) + mem(*+AR2(8h)) \rightarrow GPR(R5)$
ADDU R5, *AR2++(IR1), R3	$Mem(*AR2++(IR1)) + GPR(R3) \rightarrow GPR(R5)$
ADDU R5, *+AR1(1h), 08h	$Mem(*+AR1(1h)) + sign(08h) \rightarrow GPR(R5)$
ADDU R5, R3, *AR2++(IR1)	$GPR(R3) + Mem(*AR2++(IR1)) \rightarrow GPR(R5)$
ADDU R5, R3, *+AR1(1h)	$GPR(R3) + mem(*+AR1(1h)) \rightarrow GPR(R5)$
ADDU R5, *AR1++(IR0), *AR2++(IR1)	$Mem(*AR1++(IR0)) + Mem(*AR2++(IR1)) \rightarrow GPR(R5)$

## 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
110100				0	0	Src1		Modm	Src2		Modm	Dst		0	Modn		ARm		ARn												

操作:  $GPR(src1) + modm(ARm) \rightarrow GPR(dst)$  ||  $GPR(src2) \rightarrow modn(ARn)$

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:  $mem(*AR0--(IR0)) + GPR(R5) \rightarrow GPR(R2),$

$GPR(R3) \rightarrow mem(*+AR7(IR1))$

## AND

句型:	<b>AND</b>	rd, rs, rt	或者
	<b>AND</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>AND</b>	dst, mod(ARm), rt	或者
	<b>AND</b>	dst, *+ARm(displ), Imm	或者
	<b>AND</b>	dst, rs, mod(ARm)	或者
	<b>AND</b>	dst, rs, *+ARm(displ)	或者
	<b>AND</b>	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						Displ		ARm		Displ		ARn		00		Dst		Displ2					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(displ)) and mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) and GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) and sign(Imm) → GPR(dst)	或者
	GPR(Rs) and mod(ARm) → GPR(dst)	或者
	GPR(Rs) and mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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
110100				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:  $\text{sign}(\text{offset}) \parallel \text{offset} \parallel 0^2 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{rs}] = \text{GPR}[\text{rt}]) \rightarrow \text{condition}$   
if condition then  $\text{PC} + \text{target} \rightarrow \text{PC}$

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:  $0x400 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{r5}] = \text{GPR}[\text{r6}]) \rightarrow \text{condition}$   
if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$

## 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					<i>rs</i>					00000					<i>offset</i>																

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

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:  $0x400 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{r5}]_{31} = 1) \text{ or } (\text{GPR}[\text{r5}] = 0^{32}) \rightarrow \text{condition}$   
if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$

## 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						<i>rs</i>						00000						<i>offset</i>														

操作:  $\text{sign}(\text{offset}) \parallel \text{offset} \parallel 0^2 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{rs}]_{31} = 1) \rightarrow \text{condition}$   
if condition then  $\text{PC} + \text{target} \rightarrow \text{PC}$

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:  $0x400 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{r5}]_{31} = 1) \rightarrow \text{condition}$   
if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$

## 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:  $\text{sign}(\text{offset}) \parallel \text{offset} \parallel 0^2 \rightarrow \text{target}$ ,  
 $(\text{GPR}[\text{rs}]_{31} = 1) \rightarrow \text{condition}$ ,  
 $\text{PC} + 8 \rightarrow \text{GPR}[31]$   
 if condition then  $\text{PC} + \text{target} \rightarrow \text{PC}$

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:  $0x400 \rightarrow \text{target}$ ,  $(\text{GPR}[\text{r5}]_{31} = 1) \rightarrow \text{condition}$   
 $\text{PC} + 8 \rightarrow \text{GPR}[31]$   
 if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$

## 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						<i>rs</i>						00001						<i>offset</i>														

操作:  $\text{sign}(\text{offset}) \parallel \text{offset} \parallel 0^2 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{rs}]_{31} = 0) \rightarrow \text{condition}$   
if condition then  $\text{PC} + \text{target} \rightarrow \text{PC}$

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:  $0x400 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{r5}]_{31} = 0) \rightarrow \text{condition}$   
if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$

## 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:             $\text{sign}(\text{offset}) \parallel \text{offset} \parallel 0^2 \rightarrow \text{target}$ ,  
                    $(\text{GPR}[\text{rs}]_{31} = 0) \rightarrow \text{condition}$ ,  
                    $\text{PC} + 8 \rightarrow \text{GPR}[31]$   
                   if condition then  $\text{PC} + \text{target} \rightarrow \text{PC}$

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:            $0x400 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{r5}]_{31} = 0) \rightarrow \text{condition}$   
                    $\text{PC} + 8 \rightarrow \text{GPR}[31]$   
                   if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$

## 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					<i>rs</i>					00000					<i>offset</i>																

操作:  $\text{sign}(\text{offset}) \parallel \text{offset} \parallel 0^2 \rightarrow \text{target}$  ,  
 $(\text{GPR}[\text{rs}]_{31} = 0) \text{ and } (\text{GPR}[\text{rs}] \neq 0^{32}) \rightarrow \text{condition}$   
 if condition then  $\text{PC} + \text{target} \rightarrow \text{PC}$

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:  $0x400 \rightarrow \text{target}$  ,

$(\text{GPR}[\text{r5}]_{31} = 0) \text{ and } (\text{GPR}[\text{r5}] \neq 0^{32}) \rightarrow \text{condition}$

if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$

## 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:  $\text{sign}(\text{offset}) \parallel \text{offset} \parallel 0^2 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{rs}] \neq \text{GPR}[\text{rt}]) \rightarrow \text{condition}$   
if condition then  $\text{PC} + \text{target} \rightarrow \text{PC}$

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:  $0x400 \rightarrow \text{target}$  ,  $(\text{GPR}[\text{r5}] \neq \text{GPR}[\text{r6}]) \rightarrow \text{condition}$   
if condition then  $\text{PC} + 0x400 \rightarrow \text{PC}$



## EPACKSSDB/QD

句型:           **EPACKSSDB/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

---

和法处理溢出，结果存入 MRd 操作数。

执行周期: 1 cycle

## EPACKUSDB/QD

句型:           **EPACKUSDB/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:

$MRd[7..0] \leftarrow MRs[7..0] + MRt[7..0];$

\* repeat add operation for 2nd through 15th byte \*;

$MRd[127..120] \leftarrow MRs[127..120] + MRt[127..120];$

EPADDD instruction with 128-bit operands:

$MRd [15..0] \leftarrow MRs [15..0] + MRt [15..0];$

\* repeat add operation for 2nd and 7th double-byte \*;

$MRd[127..112] \leftarrow MRs[127..112] + MRt[127..112];$

EPADDQ instruction with 128-bit operands:

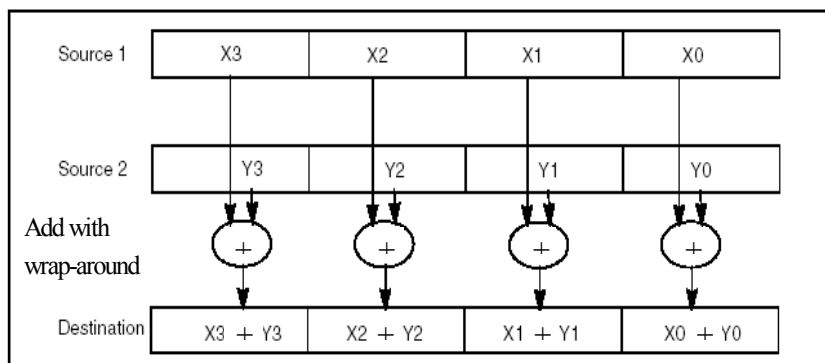
$MRd [31..0] \leftarrow MRs [31..0] + MRs [31..0];$

\* repeat add operation for 2nd and 3th word \*;

$MRd[127..96] \leftarrow MRs[127..96] + MRt[127..96];$

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。



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:

$MRd[7..0] \leftarrow \text{SaturateToSignedByte}(MRs[7..0] + MRt[7..0]);$   
 \* repeat add operation for 2nd through 15th bytes \*;  
 $MRd[127..120] \leftarrow \text{SaturateToSignedByte}(MRs[127..120] + MRt[127..120]);$

EPADDSB instruction with 128-bit operands:

$MRd[15..0] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[15..0] + MRt[15..0]);$   
 \* repeat add operation for 2nd and 3rd double-bytes \*;  
 $MRd[127..112] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[127..112] + MRt[127..112]);$

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, 使用有符号饱和和处理溢出, 结果存入 MRd 操作数中相应的位置。

EPADDSB 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:

$MRd[7..0] \leftarrow \text{SaturateToUnsignedByte}(MRs[7..0] + MRt[7..0]);$   
 \* repeat add operation for 2nd through 15th bytes \*;  
 $MRd[127..120] \leftarrow \text{SaturateToUnsignedByte}(MRs[127..120] + MRt[127..120]);$

EPADDUSD instruction with 128-bit operands:

$MRd[15..0] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[15..0] + MRt[15..0]);$   
 \* repeat add operation for 2nd and 3rd double-bytes \*;  
 $MRd[127..112] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[127..112] + MRt[127..112]);$

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:

$MRd \leftarrow MRs \text{ AND } MRt;$

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:

$MRt[7-0] \leftarrow (MRt[7-0] + MRs[7-0] + 1) \gg 1$ ; \* temp sum before shifting is 9 bits \*

\* repeat operation performed for bytes 2nd through 15th;

$MRt[127-120] \leftarrow (MRt[127-120] + MRs[127-120] + 1) \gg 1$ ;

EPAVGD instruction with 128-bit operands:

$MRt[15-0] \leftarrow (MRt[15-0] + MRs[15-0] + 1) \gg 1$ ; \* temp sum before shifting is 17 bits \*

\* repeat operation performed for double-bytes 2 and 7;

$MRt[127-112] \leftarrow (MRt[127-112] + MRs[127-112] + 1) \gg 1$ ;

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  $MRd[7..0] \leftarrow FFH$ ;

ELSE  $MRd[7..0] \leftarrow 0$ ;

\* Continue comparison of 2nd through 15th bytes in MRd and MRt \*

IF  $MRs[127..120] = MRt[127..120]$

THEN  $MRd[127..120] \leftarrow FFH$ ;

ELSE  $MRd[127..120] \leftarrow 0$ ;

EPCMPEQD instruction with 128-bit operands:

IF  $MRs[15..0] = MRt[15..0]$

THEN  $MRd[15..0] \leftarrow FFFFH$ ;

ELSE  $MRd[15..0] \leftarrow 0$ ;

\* Continue comparison of 2nd and 7th double-bytes in MRd and MRt \*

IF  $MRs[127..112] = MRt[127..112]$

THEN  $MRd[127..112] \leftarrow FFFFH$ ;

ELSE  $MRd[127..112] \leftarrow 0$ ;

EPCMPEQQ instruction with 128-bit operands:

IF  $MRs[31..0] = MRt[31..0]$

THEN  $MRd[31..0] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[31..0] \leftarrow 0$ ;

\* Continue comparison of 2nd and 3rd 4-bytes in MRd and MRt \*

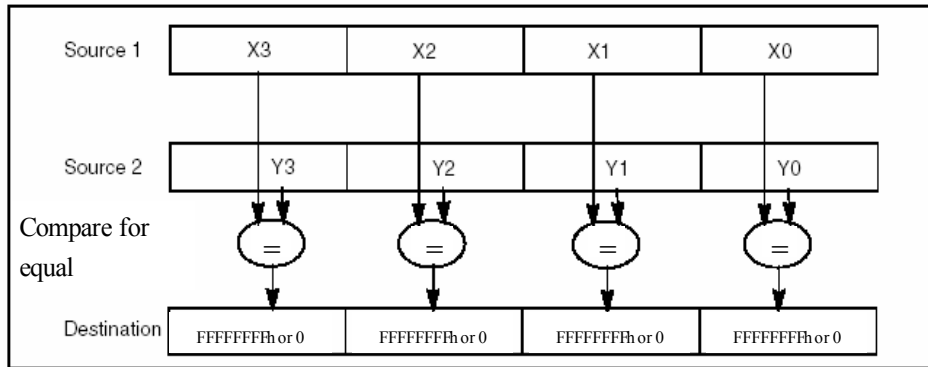
IF  $MRs[127..96] = MRt[127..96]$

THEN  $MRd[127..96] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[127..96] \leftarrow 0$ ;

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  $MRd[7..0] \leftarrow FFH$ ;

ELSE  $MRd[7..0] \leftarrow 0$ ;

\* Continue comparison of 2nd through 15th bytes in MRd and MRt \*

IF  $MRs[127..120] > MRt[127..120]$

THEN  $MRd[127..120] \leftarrow FFH$ ;

ELSE  $MRd[127..120] \leftarrow 0$ ;

EPCMPGTD instruction with 128-bit operands:

IF  $MRs[15..0] > MRt[15..0]$

THEN  $MRd[15..0] \leftarrow FFFFH$ ;

ELSE  $MRd[15..0] \leftarrow 0$ ;

\* Continue comparison of 2nd and 7th double-bytes in MRd and MRt \*

IF  $MRs[127..112] > MRt[127..112]$

THEN  $MRd[127..112] \leftarrow FFFFH$ ;

ELSE  $MRd[127..112] \leftarrow 0$ ;

EPCMPGTQ instruction with 128-bit operands:

IF  $MRs[31..0] > MRt[31..0]$

THEN  $MRd[31..0] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[31..0] \leftarrow 0$ ;

\* Continue comparison of 2nd and 3rd 4-bytes in MRd and MRt \*

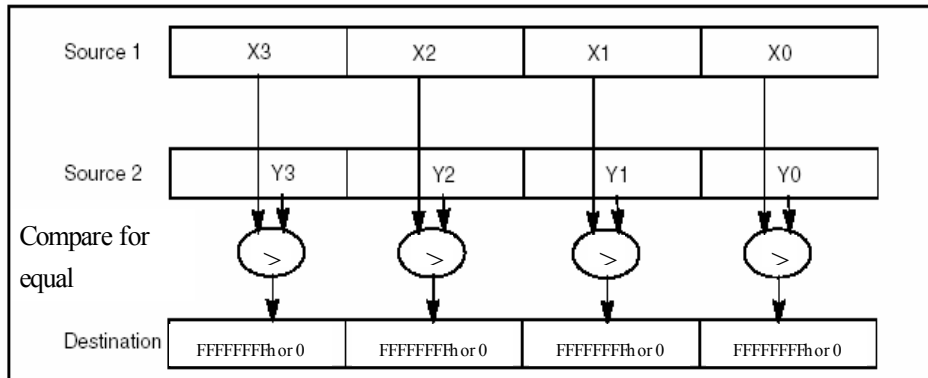
IF  $MRs[127..96] > MRt[127..96]$

THEN  $MRd[127..96] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[127..96] \leftarrow 0$ ;

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:

$MRd[127-0] \leftarrow \text{memory}$

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:

$Vaddr = Mod(ARm),$

$MRd[127:0] \leftarrow Left(mem(Vaddr))$

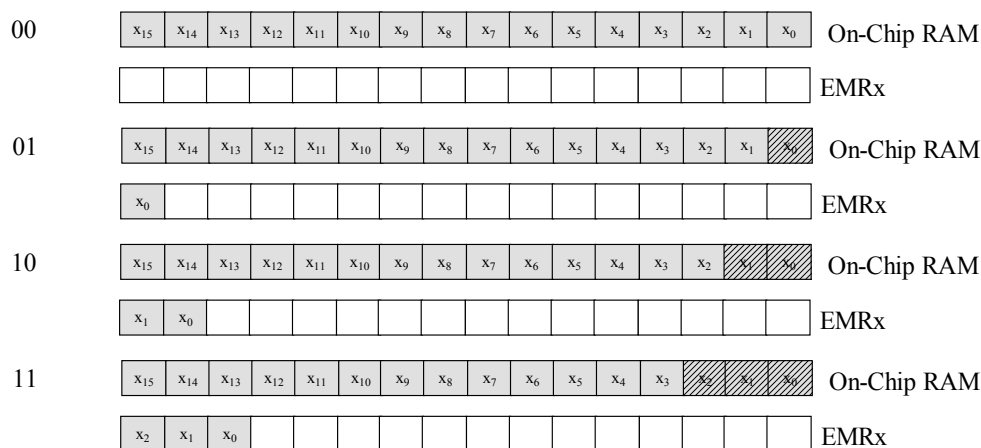
操作数说明: MRd: MDS 寄存器

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:

$Vaddr = Mod(ARm)$ ,

$MRd[127-0] \leftarrow Right(mem(Vaddr))$

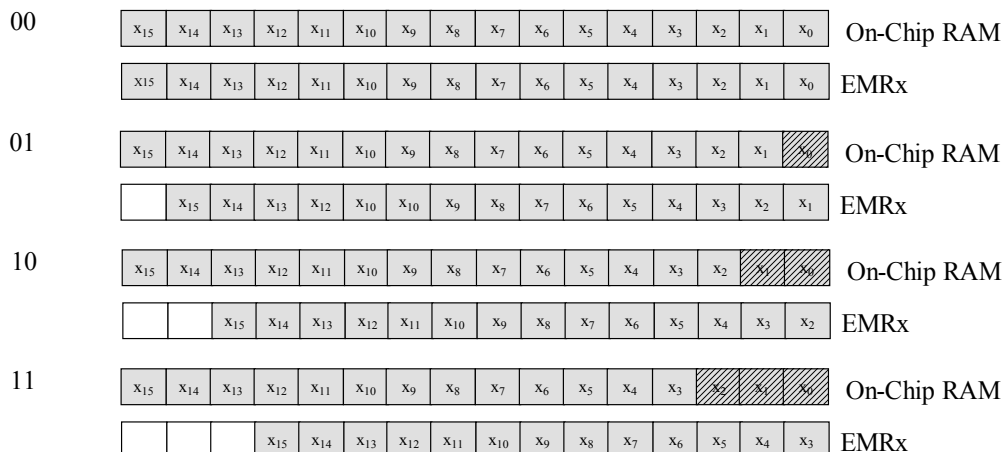
操作数说明: MRd: MDS 寄存器

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:

$$MRd[31..0] \leftarrow (MRs[15..0] \times MRt[15..0]) + (MRs[31..16] \times MRt[31..16]);$$

\* repeat operation performed for bytes 2nd through 3rd \*;

$$MRd[127..96] \leftarrow (MRs[111..96] \times MRt[111..96]) + (MRs[127..112] \times MRt[127..112]);$$

\* 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

( $MRd[15-0] \leftarrow MRd[15-0]$ );

ELSE

( $MRd[15-0] \leftarrow MRt[15-0]$ );

FI

\* repeat operation for 2nd and 7th double-bytes in source and destination operands \*

IF  $MRs[127-112] > MRt[127-112]$  THEN

( $MRd[127-112] \leftarrow MRd[127-112]$ );

ELSE

( $MRd[127-112] \leftarrow MRt[127-112]$ );

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

( $MRd[7-0] \leftarrow MRd[7-0]$ );

ELSE

( $MRd[7-0] \leftarrow MRt[7-0]$ );

FI

\* repeat operation for 2nd through 15th bytes in source and destination operands \*

IF  $MRs[127-120] > MRt[127-120]$  THEN

( $MRd[127-120] \leftarrow MRd[127-120]$ );

ELSE

( $MRd[127-120] \leftarrow MRt[127-120]$ );

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

$(MRd[15-0] \leftarrow MRd[15-0]);$

ELSE

$(MRd[15-0] \leftarrow MRt[15-0]);$

IF

    \* repeat operation for 2nd and 7th double-bytes in source and destination operands \*

IF  $MRs[127-112] < MRt[127-112]$  THEN

$(MRd[127-112] \leftarrow MRd[127-112]);$

ELSE

$(MRd[127-112] \leftarrow MRt[127-112]);$

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

$MRd[7-0] \leftarrow MRd[7-0]$ ;

ELSE

$MRd[7-0] \leftarrow MRt[7-0]$ ;

IF

    \* repeat operation for 2nd through 15th bytes in source and destination operands \*

IF  $MRs[127-120] < MRt[127-120]$  THEN

$MRd[127-120] \leftarrow MRd[127-120]$ ;

ELSE

$MRd[127-120] \leftarrow MRt[127-120]$ ;

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:

$TEMP0[31-0] \leftarrow MRs[15-0] \times MRt[15-0];$

\* Signed multiplication \*

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$TEMP7[31-0] \leftarrow MRs[127-112] \times MRt[127-112];$

$MRd[15-0] \leftarrow TEMP7[31-16];$

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$MRd[127-112] \leftarrow TEMP7[31-16];$

EPMACHSD:

$TEMP0[31-0] \leftarrow MRs[15-0] \times MRt[15-0];$

\* Signed multiplication \*

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$TEMP7[31-0] \leftarrow MRs[127-112] \times MRt[127-112];$

$MRd[15-0] \leftarrow MRd[15-0] + TEMP0[31-16];$

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$MRd[127-112] \leftarrow MRd[127-112] + TEMP7[31-16];$

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:

$TEMP0[31-0] \leftarrow MRs[15-0] \times MRt[15-0];$

\* Signed multiplication \*

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$TEMP7[31-0] \leftarrow MRs[127-112] \times MRt[127-112];$

$MRd[15-0] \leftarrow TEMP0[15-0];$

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$MRd[127-112] \leftarrow TEMP7[15-0];$

EPMACLSD:

$TEMP0[31-0] \leftarrow MRs[15-0] \times MRt[15-0];$

\* Signed multiplication \*

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$TEMP7[31-0] \leftarrow MRs[127-112] \times MRt[127-112];$

$MRd[15-0] \leftarrow MRd[15-0] + TEMP0[15-0];$

\* repeat operation performed for 2-bytes 2nd through 7th \*;

$MRd[127-112] \leftarrow MRd[127-112] + TEMP7[15-0];$

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:

$MRd \leftarrow MRd \text{ NOR } MRt;$

**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:

$MRd \leftarrow MRd \text{ OR } MRt;$

**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  $\leftarrow$  ABS(MRs[7-0] - MRt[7-0]);  
 \* repeat operation for bytes 2 through 15 \*;  
 TEMP15  $\leftarrow$  ABS(MRs[127-120] - MRt[127-120]);  
 MRd[23:0]  $\leftarrow$  SUM(TEMP0...TEMP15);  
 MRd[127:24]  $\leftarrow$  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 2<sup>nd</sup>, 3<sup>rd</sup>, 4<sup>rd</sup> quad-bytes;

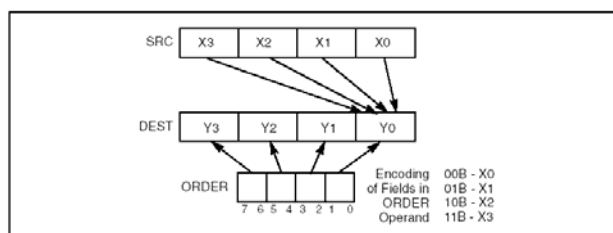
操作数说明: MRs:   MDS 寄存器

              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

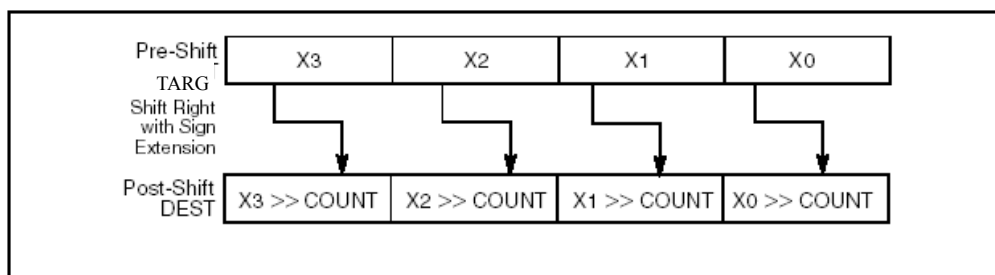
MRd: MDS 寄存器

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

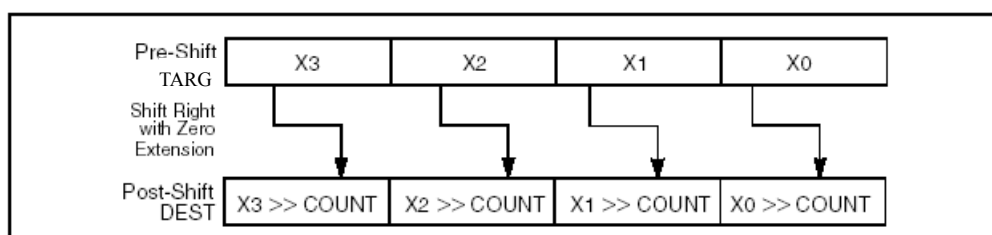
Sa: 立即数

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
111100						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
111100						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

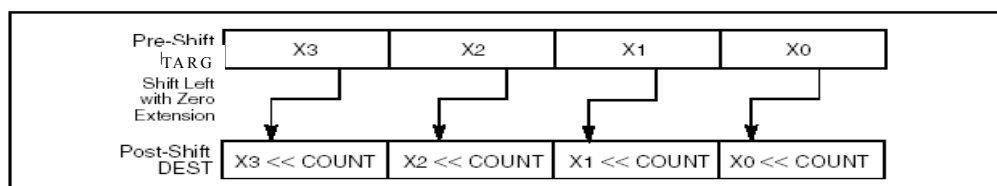
MRd: MDS 寄存器

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  $\leftarrow$  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:

$MRd[7..0] \leftarrow MRs[7..0] - MRt[7..0];$   
     \* repeat sub operation for 2nd through 15th byte \*;  
 $MRd[127..120] \leftarrow MRs[127..120] - MRt[127..120];$

EPSUBD instruction with 128-bit operands:

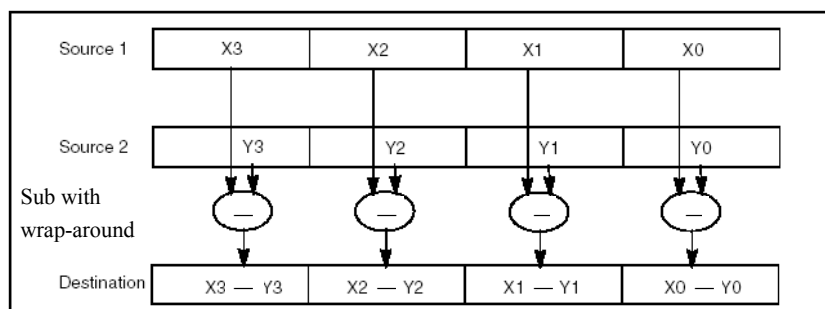
$MRd[15..0] \leftarrow MRs[15..0] - MRt[15..0];$   
     \* repeat add operation for 2nd and 7th double-byte \*;  
 $MRd[127..112] \leftarrow MRs[127..112] - MRt[127..112];$

EPSUBQ instruction with 128-bit operands:

$MRd[31..0] \leftarrow MRs[31..0] - MRt[31..0];$   
     \* repeat add operation for 2nd and 3rd double-byte \*;  
 $MRd[127..96] \leftarrow MRs[127..96] - MRt[127..96];$

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:

$MRd[7..0] \leftarrow \text{SaturateToSignedByte}(MRs[7..0] - MRt(7..0));$   
     \* repeat sub operation for 2nd through 15th bytes \*;  
 $MRd[127..120] \leftarrow \text{SaturateToSignedByte}(MRs[127..120] - MRt[127..120]);$

EPSUBSD instruction with 128-bit operands:

$MRd[15..0] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[15..0] - MRt[15..0]);$   
     \* repeat sub operation for 2nd and 7th double-bytes \*;  
 $MRd[127..112] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[127..112] - MRt[127..112]);$

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:

$MRd[7..0] \leftarrow \text{SaturateToUnsignedByte}(MRs[7..0] - MRt(7..0));$

\* repeat sub operation for 2nd through 15th bytes \*;

$MRd[127..120] \leftarrow \text{SaturateToUnsignedByte}(MRs[127..120] - MRt[127..120]);$

EPSUBUSD instruction with 128-bit operands:

$MRd[15..0] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[15..0] - MRt[15..0]);$

\* repeat sub operation for 2nd and 7th double-bytes \*;

$MRd[127..112] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[127..112] - MRt[127..112]);$

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:

$MRd[7..0] \leftarrow MRs[71..64];$   
 $MRd[15..8] \leftarrow MRt[71..64];$   
 $MRd[23..16] \leftarrow MRs[79..72];$   
 $MRd[31..24] \leftarrow MRt[79..72];$   
 $MRd[39..32] \leftarrow MRs[87..80];$   
 $MRd[47..40] \leftarrow MRt[87..80];$   
 $MRd[55..48] \leftarrow MRs[95..88];$   
 $MRd[63..56] \leftarrow MRt[95..88];$   
 $MRd[71..64] \leftarrow MRs[103..96];$   
 $MRd[79..72] \leftarrow MRt[103..96];$   
 $MRd[87..80] \leftarrow MRs[111..104];$   
 $MRd[95..88] \leftarrow MRt[111..104];$   
 $MRd[103..96] \leftarrow MRs[119..112];$   
 $MRd[111..104] \leftarrow MRt[119..112];$   
 $MRd[119..112] \leftarrow MRs[127..120];$   
 $MRd[127..120] \leftarrow MRt[127..120];$

EPUNPCKHDQ:

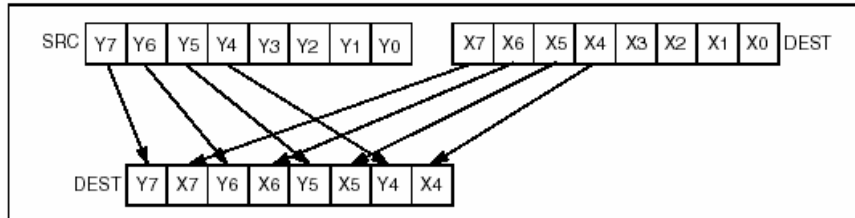
$MRd[15..0] \leftarrow MRs[79..64];$   
 $MRd[31..16] \leftarrow MRt[79..64];$   
 $MRd[47..32] \leftarrow MRs[95..80];$   
 $MRd[63..48] \leftarrow MRt[95..80];$   
 $MRd[79..64] \leftarrow MRs[111..96];$   
 $MRd[95..80] \leftarrow MRt[111..96];$   
 $MRd[111..96] \leftarrow MRs[127..112];$   
 $MRd[127..112] \leftarrow MRt[127..112];$

EPUNPCKHQO:

$MRd[31..0] \leftarrow MRs[95..64]$   
 $MRd[63..32] \leftarrow MRt[95..64];$   
 $MRd[95..64] \leftarrow MRs[127..96]$   
 $MRd[127..96] \leftarrow MRt[127..96];$

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:

$MRd[7..0] \leftarrow MRs[7..0];$   
 $MRd[15..8] \leftarrow MRt[7..0];$   
 $MRd[23..16] \leftarrow MRs[15..8];$   
 $MRd[31..24] \leftarrow MRt[15..8];$   
 $MRd[39..32] \leftarrow MRs[23..16];$   
 $MRd[47..40] \leftarrow MRt[23..16];$   
 $MRd[55..48] \leftarrow MRs[31..24];$   
 $MRd[63..56] \leftarrow MRt[31..24];$   
 $MRd[71..64] \leftarrow MRs[39..32];$   
 $MRd[79..72] \leftarrow MRt[39..32];$   
 $MRd[87..80] \leftarrow MRs[47..40];$   
 $MRd[95..88] \leftarrow MRt[47..40];$   
 $MRd[103..96] \leftarrow MRs[55..48];$   
 $MRd[111..104] \leftarrow MRt[55..48];$   
 $MRd[119..112] \leftarrow MRs[63..56];$   
 $MRd[127..120] \leftarrow MRt[63..56];$

EPUNPCKLDQ:

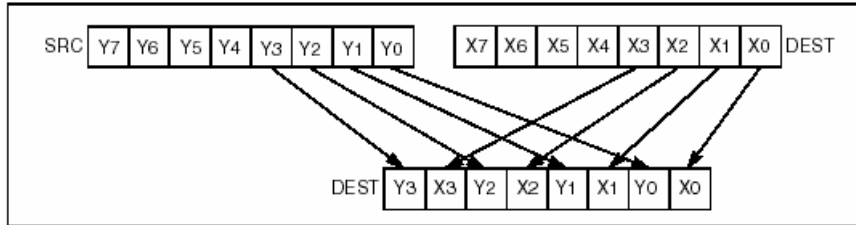
$MRd[15..0] \leftarrow MRs[15..0];$   
 $MRd[31..16] \leftarrow MRt[15..0];$   
 $MRd[47..32] \leftarrow MRs[31..16];$   
 $MRd[63..48] \leftarrow MRt[31..16];$   
 $MRd[79..64] \leftarrow MRs[47..32];$   
 $MRd[95..80] \leftarrow MRt[47..32];$   
 $MRd[111..96] \leftarrow MRs[63..48];$   
 $MRd[127..112] \leftarrow MRt[63..48];$

EPUNPCKLQO:

$MRd[31..0] \leftarrow MRs[31..0];$   
 $MRd[63..32] \leftarrow MRt[31..0];$   
 $MRd[95..64] \leftarrow MRs[63..32];$   
 $MRd[127..96] \leftarrow MRt[63..32];$

**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:

$MRd \leftarrow MRs \text{ XOR } MRt;$

**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

## GETBITS

句型:       **GETBITS**       rd, ra, sa       或者  
              **GETBITS**       rd, ra, rn

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	0	留	01	ra	sa	11	rd	sa				001111													

31	30	29	28	27	26	25	24	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	1	留	01	ra	sa	11	rd	rn				001111													

操作:

GETBITS:

$n = sa$  or  $ValueOfReg(rn)$

IF ( $n > 32$ )

THEN  $n \leftarrow 32$ ;

$rd = HeadBitsOfVLDBuffer(n)$ ;

$ShiftBitsOfVLDBuffer(n)$ ;

IF ( $BitsNumOfVLDBuffer < 33$ )

THEN

$Fill32bitToVLDBuffer(ra) \parallel ValueOfReg(ra) \leftarrow ValueOfReg(ra) + 4$ ;

操作数说明: rd:       寄存器   (通用寄存器 0~7)  
              ra:       寄存器   (辅助寄存器 0~3)  
              sa:       立即数  
              rn:       寄存器   (通用寄存器 0~31)

描述:        VLD 指令 GetBits 完成的操做是从码流中取出  $n$  个比特, 并且码流指针也相应的偏移  $n$  个比特。其中  $rd$  是目标寄存器,  $ra$  是地址寄存器,  $rn$  是取比特数寄存器,  $sa$  是取 bit 立即数。

执行周期: 1 cycle

## 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:            $PC_{31-28} \parallel \text{target} \parallel 0^2 \rightarrow PC$

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:        $PC_{31-28} \parallel 100\_0000_{27-0} \rightarrow PC$

## 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:  $PC_{31\sim28} \parallel target \parallel 0^2 \rightarrow PC$  并且  $PC + 8 \rightarrow GPR[31]$

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:          $PC_{31\sim28} \parallel 100\_0000_{27\sim0} \rightarrow PC$  并且  $PC + 8 \rightarrow GPR[31]$

## 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						<i>rs</i>						0000000000000000												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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,    或者     $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{Sign}(\text{Byte}(\text{mem}(\text{Vaddr}))) \rightarrow \text{GPR}(\text{rt})$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{Sign}(\text{Byte}(\text{mem}(\text{Vaddr}))) \rightarrow \text{GPR}(R5)$
LB R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{Sign}(\text{Byte}(\text{mem}(\text{Vaddr}))) \rightarrow \text{GPR}(R5)$

## 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,    或者     $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{zero}(\text{Byte}(\text{mem}(Vaddr))) \rightarrow \text{GPR}(\text{rt})$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $0(\text{Byte}(\text{mem}(Vaddr))) \rightarrow \text{GPR}(R5)$
LBU R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $0(\text{Byte}(\text{mem}(Vaddr))) \rightarrow \text{GPR}(R5)$



## 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,    或者     $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{Sign}(\text{Byte}(\text{mem}(\text{Vaddr}))) \rightarrow \text{GPR}(\text{rt})$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{Sign}(\text{Byte}(\text{mem}(\text{Vaddr}))) \rightarrow \text{RP}(R5)$
LH R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{Sign}(\text{Byte}(\text{mem}(\text{Vaddr}))) \rightarrow \text{RP}(R5)$

## 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ , 或者  $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{zero}(\text{Byte}(\text{mem}(Vaddr))) \rightarrow \text{GPR}(\text{rt})$

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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $0(\text{Byte}(\text{mem}(Vaddr))) \rightarrow \text{GPR}(R5)$
LHU R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $0(\text{Byte}(\text{mem}(Vaddr))) \rightarrow \text{GPR}(R5)$

## 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( $0^{16}$ )  $\rightarrow$  GPR(rt) 或者  
 mem(Imm) [15:0] | zero( $0^{16}$ )  $\rightarrow$  GPR(dst) 或者  
 modm(ARm) [15:0] | zero( $0^{16}$ )  $\rightarrow$  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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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( $0^{16}$ ) $\rightarrow$ GPR(R5)
LUI R5, @0840h	mem(0840h) [15:0]   zero( $0^{16}$ ) $\rightarrow$ GPR(R5)
LUI R5, *AR2++(40h)	Mem(AR2) [15:0]   zero( $0^{16}$ ) $\rightarrow$ 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,        或者         $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{mem}(Vaddr) \rightarrow \text{GPR}(\text{rt})$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{mem}(Vaddr) \rightarrow \text{GPR}(R5)$
LW R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{mem}(Vaddr) \rightarrow \text{GPR}(R5)$

## 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ , 或者  $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{Left}(\text{mem}(Vaddr)) \rightarrow \text{GPR}(\text{rt})$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*--ARn(displ)	00001	*--ARn(IR0)
		01001	*--ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{Left}(\text{mem}(Vaddr)) \rightarrow \text{GPR}(R5)$
LWL R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{Left}(\text{mem}(Vaddr)) \rightarrow \text{GPR}(R5)$

## 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ , 或者  $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{right}(\text{mem}(Vaddr)) \rightarrow \text{GPR}(\text{rt})$

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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{Right}(\text{mem}(Vaddr)) \rightarrow \text{GPR}(R5)$
LWR R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{Right}(\text{mem}(Vaddr)) \rightarrow \text{GPR}(R5)$

## 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
011011				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
<b>010000</b>		<b>00000</b>			<b>rt</b>	<b>rd</b>		<b>000_0000_0000</b>	

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:	<b>MULT</b>	rs, rt	或者
	<b>MULT</b>	*+ARm(displ), *+ARn(displ2)	或者
	<b>MULT</b>	mod(ARm), rt	或者
	<b>MULT</b>	*+ARm(displ), Imm	或者
	<b>MULT</b>	rs, mod(ARm)	或者
	<b>MULT</b>	rs, *+ARm(displ)	或者
	<b>MULT</b>	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						Displ		ARm		Displ		ARn		00		000		Displ2				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(displ)) * mem(*+ARn(displ2)) → MR0{HI,LO}	或者
	mod(ARm) * GPR(Rt) → MR0{HI,LO}	或者
	Mem(*+ARm(displ)) * sign(Imm) → MR0{HI,LO}	或者
	GPR(Rs) * mod(ARm) → MR0{HI,LO}	或者
	GPR(Rs) * mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	寄存器
	E=01	*+ARn(displ)寻址
10	E=00	寄存器
	E=01	*+ARn(displ)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:	<b>MULTU</b>	rs, rt	或者
	<b>MULTU</b>	*+ARm(displ), *+ARn(displ2)	或者
	<b>MULTU</b>	mod(ARm), rt	或者
	<b>MULTU</b>	*+ARm(displ), Imm	或者
	<b>MULTU</b>	rs, mod(ARm)	或者
	<b>MULTU</b>	rs, *+ARm(displ)	或者
	<b>MULTU</b>	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						Displ						ARm						Displ						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:	$GPR(Rs) * GPR(rt) \rightarrow MR0\{HI,LO\}$	或者
	$Mem(*+ARm(displ)) * mem(*+ARn(displ2)) \rightarrow MR0\{HI,LO\}$	或者
	$mod(ARm) * GPR(Rt) \rightarrow MR0\{HI,LO\}$	或者
	$Mem(*+ARm(displ)) * sign(Imm) \rightarrow MR0\{HI,LO\}$	或者
	$GPR(Rs) * mod(ARm) \rightarrow MR0\{HI,LO\}$	或者
	$GPR(Rs) * mem(*+ARm(displ)) \rightarrow MR0\{HI,LO\}$	或者
	$Modm(ARm) * modn(ARn) \rightarrow MR0\{HI,LO\}$	

## 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(displ)寻址	*+ARn(displ)寻址
01	E=00 间接寻址	寄存器
	E=01 *+ARn(displ)寻址	立即数
10	E=00 寄存器	间接寻址
	E=01 寄存器	*+ARn(displ)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
011011				00		Src1		P	Src2		0	Modm			D	Modn		ARm		ARn											

操作:  $\text{srcA} * \text{srcB} \rightarrow \{\text{HI}, \text{LO}\}$       ||       $\text{srcC} + \text{srcD} \rightarrow \text{GPR}(\text{D})$

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:  $\text{mem}(*\text{AR0--}(\text{IR0})) * \text{mem}(*\text{+AR7}(\text{IR1})) \rightarrow \{\text{HI}, \text{LO}\}$ ,  
 $\text{GPR}(\text{R5}) + \text{GPR}(\text{R7}) \rightarrow \text{GPR}(\text{R3})$

## 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
011011					01	Src1	P	Src2	0	Modm	D	Modn	ARm	ARn																	

操作:  $\text{srcA} * \text{srcB} \rightarrow \{\text{HI}, \text{LO}\}$  ||  $\text{srcC} - \text{srcD} \rightarrow \text{GPR}(\text{D})$

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	$\text{Modm}(\text{ARm}) * \text{modn}(\text{ARn})$ , $\text{src1} - \text{src2}$
01	$\text{Modm}(\text{ARm}) * \text{src1}$ , $\text{modn}(\text{ARn}) - \text{src2}$
10	$\text{Src1} * \text{src2}$ , $\text{Modm}(\text{ARm}) - \text{modn}(\text{ARn})$
11	$\text{Modm}(\text{ARm}) * \text{src1}$ , $\text{src2} - \text{modn}(\text{ARn})$

执行周期: 3 cycles

举例:

MULT\_SUB R0, R3, \*AR0--(IR0), R5, \*+AR7(IR1), R7

操作:  $\text{mem}(*\text{AR0--}(\text{IR0})) * \text{GPR}(\text{R5}) \rightarrow \{\text{HI}, \text{LO}\}$ ,

$\text{mem}(*\text{+AR7}(\text{IR1})) - \text{GPR}(\text{R7}) \rightarrow \text{GPR}(\text{R3})$

## 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
110100				1	0	Src1			Modm	Src2			Modm	Dst		1	Modn			ARm		ARn									

操作:  $\text{modm}(\text{ARm}) * \text{GPR}(\text{src1}) \rightarrow \text{GPR}(\text{dst}) \quad || \quad \text{GPR}(\text{src2}) \rightarrow \text{modn}(\text{ARn})$

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:  $\text{mem}(*\text{AR0--}(\text{IR0})) * \text{GPR}(\text{R5}) \rightarrow \text{MR0}(\text{Hi,LO}),$

$\text{GPR}(\text{R3}) \rightarrow \text{mem}(*\text{AR7}(\text{IR1}))$

## MTCO

句型: **MTCO** rt, rd

指令编码:

31	26	25	21	20	16	15	11	10	0
<b>010000</b>		<b>00100</b>		<b>rt</b>		<b>rd</b>		<b>000_0000_0000</b>	

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:	<b>NOR</b>	rd, rs, rt	或者
	<b>NOR</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>NOR</b>	dst, mod(ARm), rt	或者
	<b>NOR</b>	dst, *+ARm(displ), Imm	或者
	<b>NOR</b>	dst, rs, mod(ARm)	或者
	<b>NOR</b>	dst, rs, *+ARm(displ)	或者
	<b>NOR</b>	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						Displ		ARm		Displ		ARn		00		Dst		Displ2					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(displ)) nor mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) nor GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) nor sign(Imm) → GPR(dst)	或者
	GPR(Rs) nor mod(ARm) → GPR(dst)	或者
	GPR(Rs) nor mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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:	<b>OR</b>	rd, rs, rt	或者
	<b>OR</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>OR</b>	dst, mod(ARm), rt	或者
	<b>OR</b>	dst, *+ARm(displ), Imm	或者
	<b>OR</b>	dst, rs, mod(ARm)	或者
	<b>OR</b>	dst, rs, *+ARm(displ)	或者
	<b>OR</b>	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						Displ		ARm		Displ		ARn		00		Dst		Displ2		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(displ)) or mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) or GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) or sign(Imm) → GPR(dst)	或者
	GPR(Rs) or mod(ARm) → GPR(dst)	或者
	GPR(Rs) or mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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
110100					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**            *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:

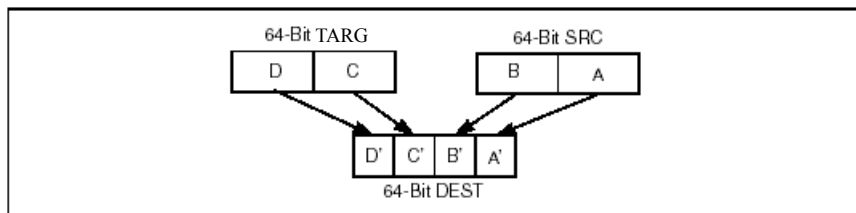
MRd[15..0] ← SaturateSignedQuad-byteToSignedDouble-byte MRs[31..0];  
 MRd[31..16] ← SaturateSignedQuad-byteToSignedDouble-byte MRs[63..32];  
 MRd[47..32] ← SaturateSignedQuad-byteToSignedDouble-byte MRt[31..0];  
 MRd[63..48] ← SaturateSignedQuad-byteToSignedDouble-byte MRt[63..32];

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:

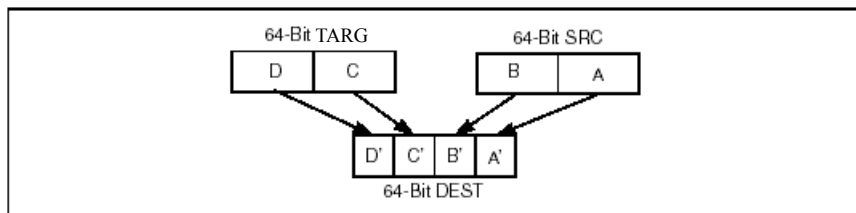
MRd[15..0] ← SaturateSignedQuad-byteToUnsignedDouble-byte MRs[31..0];  
 MRd[31..16] ← SaturateSignedQuad-byteToUnsignedDouble-byte MRs[63..32];  
 MRd[47..32] ← SaturateSignedQuad-byteToUnsignedDouble-byte MRt[31..0];  
 MRd[63..48] ← SaturateSignedQuad-byteToUnsignedDouble-byte MRt[63..32];

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:

$MRd[7..0] \leftarrow MRs[7..0] + MRt[7..0];$

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

$MRd[63..56] \leftarrow MRs[63..56] + MRt[63..56];$

PADDD instruction with 64-bit operands:

$MRd[15..0] \leftarrow MRs[15..0] + MRt[15..0];$

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

$MRd[63..48] \leftarrow MRs[63..48] + MRt[63..48];$

PADDQ instruction with 64-bit operands:

$MRd[31..0] \leftarrow MRs[31..0] + MRt[31..0];$

$MRd[63..32] \leftarrow MRs[63..32] + MRt[63..32];$

操作数说明: MRs: MDS 寄存器

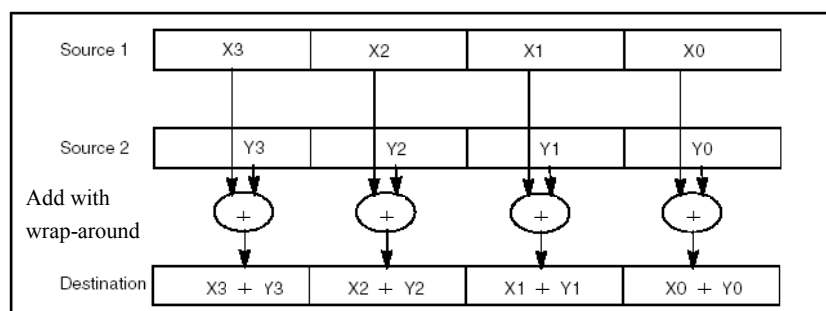
MRt: MDS 寄存器

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:

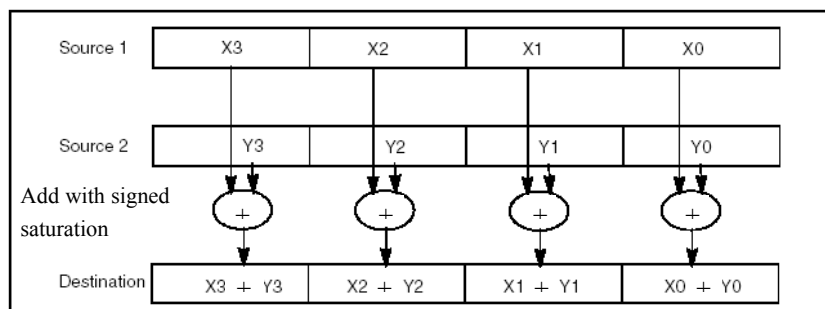
$MRd[7..0] \leftarrow \text{SaturateToSignedByte}(MRs[7..0] + MRt[7..0]);$   
 \* repeat add operation for 2nd through 7th bytes \*;  
 $MRd[63..56] \leftarrow \text{SaturateToSignedByte}(MRs[63..56] + MRt[63..56]);$

PADDSB instruction with 64-bit operands:

$MRd[15..0] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[15..0] + MRt[15..0]);$   
 \* repeat add operation for 2nd and 3rd double-bytes \*;  
 $MRd[63..48] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[63..48] + MRt[63..48]);$

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。PADDSB 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 PADDSB 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:

$MRd[7..0] \leftarrow \text{SaturateToUnsignedByte}(MRs[7..0] + MRt[7..0]);$

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

$MRd[63..56] \leftarrow \text{SaturateToUnsignedByte}(MRs[63..56] + MRt[63..56]);$

PADDUSD instruction with 64-bit operands:

$MRd[15..0] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[15..0] + MRt[15..0]);$

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

$MRd[63..48] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[63..48] + MRt[63..48]);$

操作数说明: MRs: MDS 寄存器

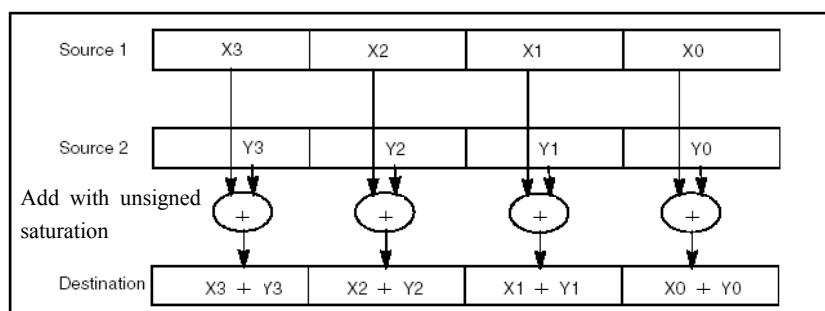
MRt: MDS 寄存器

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:

$MRd \leftarrow MRs \text{ AND } MRt;$

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:

$MRt[7-0] \leftarrow (MRt[7-0] + MRs[7-0] + 1) \gg 1$ ; \* temp sum before shifting is 9 bits \*

\* repeat operation performed for bytes 2 through 6;

$MRt[63-56] \leftarrow (MRt[63-56] + MRs[63-56] + 1) \gg 1$ ;

PAVGD instruction with 64-bit operands:

$MRt[15-0] \leftarrow (MRt[15-0] + MRs[15-0] + 1) \gg 1$ ; \* temp sum before shifting is 17 bits \*

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

$MRt[63-48] \leftarrow (MRt[63-48] + MRs[63-48] + 1) \gg 1$ ;

操作数说明: MRs: MDS 寄存器

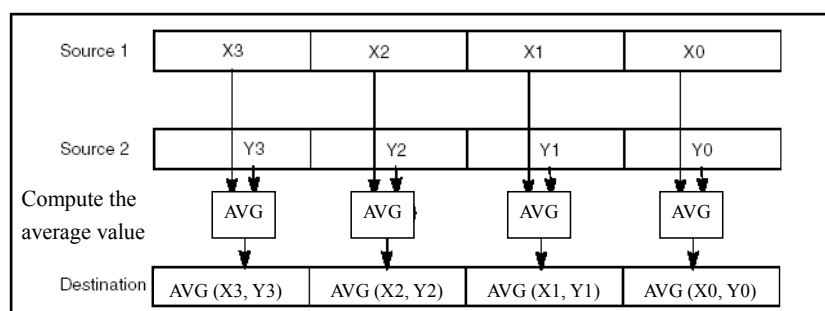
MRt: MDS 寄存器

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  $MRd[7..0] \leftarrow FFH$ ;

ELSE  $MRd[7..0] \leftarrow 0$ ;

\* Continue comparison of 2nd through 7th bytes in MRd and MRt \*

IF  $MRs[63..56] = MRt[63..56]$

THEN  $MRd[63..56] \leftarrow FFH$ ;

ELSE  $MRd[63..56] \leftarrow 0$ ;

PCMPEQD instruction with 64-bit operands:

IF  $MRs[15..0] = MRt[15..0]$

THEN  $MRd[15..0] \leftarrow FFFFH$ ;

ELSE  $MRd[15..0] \leftarrow 0$ ;

\* Continue comparison of 2nd and 3rd double-bytes in MRd and MRt \*

IF  $MRs[63..48] = MRt[63..48]$

THEN  $MRd[63..48] \leftarrow FFFFH$ ;

ELSE  $MRd[63..48] \leftarrow 0$ ;

PCMPEQQ instruction with 64-bit operands:

IF  $MRs[31..0] = MRt[31..0]$

THEN  $MRd[31..0] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[31..0] \leftarrow 0$ ;

IF  $MRs[63..32] = MRt[63..32]$

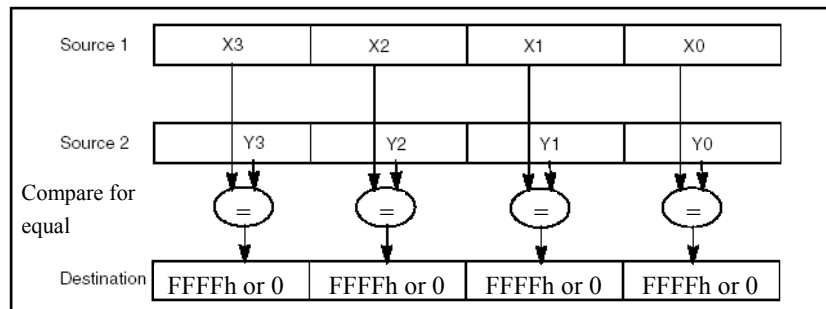
THEN  $MRd[63..32] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[63..32] \leftarrow 0$ ;

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  $MRd[7..0] \leftarrow FFH$ ;

ELSE  $MRd[7..0] \leftarrow 0$ ;

\* Continue comparison of 2nd through 7th bytes in MRd and MRt \*

IF  $MRs[63..56] > MRt[63..56]$

THEN  $MRd[63..56] \leftarrow FFH$ ;

ELSE  $MRd[63..56] \leftarrow 0$ ;

PCMPGTD instruction with 64-bit operands:

IF  $MRs[15..0] > MRt[15..0]$

THEN  $MRd[15..0] \leftarrow FFFFH$ ;

ELSE  $MRd[15..0] \leftarrow 0$ ;

\* Continue comparison of 2nd and 3rd double-bytes in MRd and MRt \*

IF  $MRs[63..48] > MRt[63..48]$

THEN  $MRd[63..48] \leftarrow FFFFH$ ;

ELSE  $MRd[63..48] \leftarrow 0$ ;

PCMPGTQ instruction with 64-bit operands:

IF  $MRs[31..0] > MRt[31..0]$

THEN  $MRd[31..0] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[31..0] \leftarrow 0$ ;

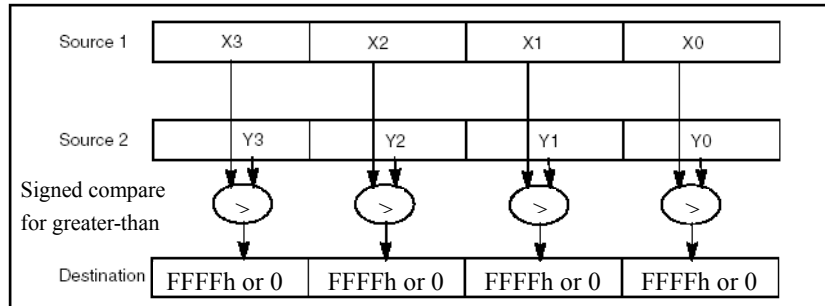
IF  $MRs[63..32] > MRt[63..32]$

THEN  $MRd[63..32] \leftarrow FFFFFFFFH$ ;

ELSE  $MRd[63..32] \leftarrow 0$ ;

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:

$MRd[63-0] \leftarrow \text{memory}$

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:

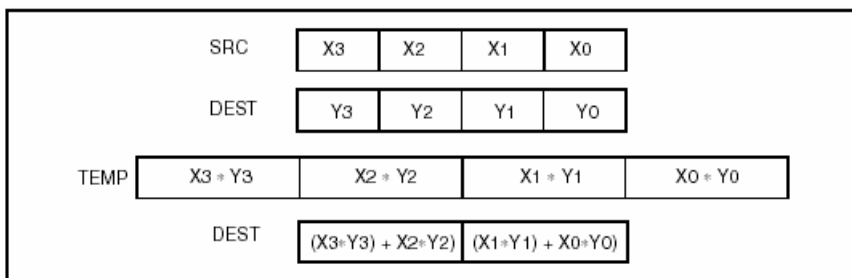
PMADDQD instruction with 64-bit operands:

$MRd[31..0] \leftarrow (MRs[15..0] \times MRt[15..0]) + (MRs[31..16] \times MRt[31..16]);$

$MRd[63..32] \leftarrow (MRs[47..32] \times MRt[47..32]) + (MRs[63..48] \times MRt[63..48]);$

\* 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

( $MRd[15-0] \leftarrow MRd[15-0]$ ;

ELSE

( $MRd[15-0] \leftarrow MRt[15-0]$ ;

FI

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

IF  $MRs[63-48] > MRt[63-48]$  THEN

( $MRd[63-48] \leftarrow MRd[63-48]$ ;

ELSE

( $MRd[63-48] \leftarrow MRt[63-48]$ ;

FI

操作数说明: MRs: MDS 寄存器

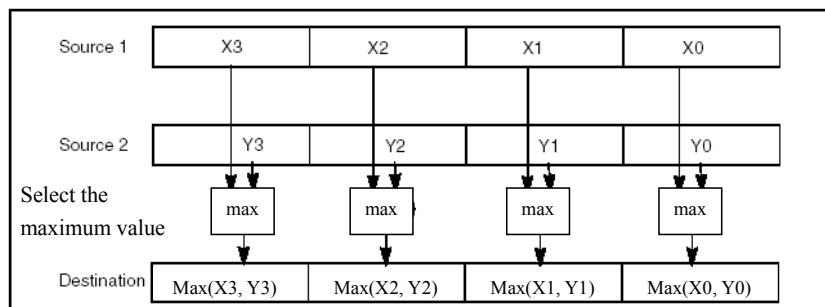
MRt: MDS 寄存器

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

( $MRd[7-0] \leftarrow MRd[7-0]$ );

ELSE

( $MRd[7-0] \leftarrow MRt[7-0]$ );

FI

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

IF  $MRs[63-56] > MRt[63-56]$  THEN

( $MRd[63-56] \leftarrow MRd[63-56]$ );

ELSE

( $MRd[63-56] \leftarrow MRt[63-56]$ );

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:

$$Rt[31-0] \leftarrow MRs[63-32]$$

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:

$Rt[31-0] \leftarrow MRs[31-0]$

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]  $\leftarrow$  MRd[15-0];

ELSE

(MRd[15-0]  $\leftarrow$  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]  $\leftarrow$  MRd[63-48];

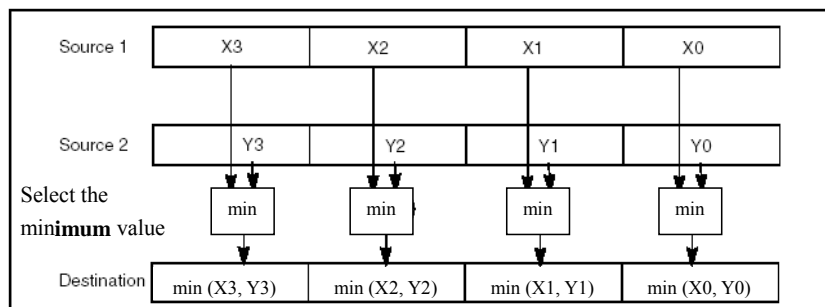
ELSE

(MRd[63-48]  $\leftarrow$  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:

$MRd[63-32] \leftarrow Rs[31-0]$

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:

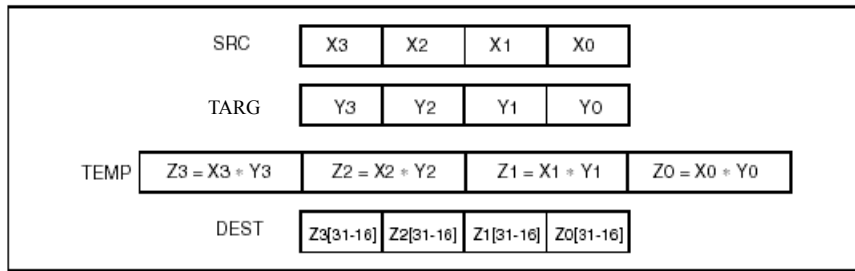
TEMP0[31-0] ← MRs[15-0] × MRt[15-0]; \* Signed multiplication \*  
 TEMP1[31-0] ← MRs[31-16] × MRt[31-16];  
 TEMP2[31-0] ← MRs[47-32] × MRt[47-32];  
 TEMP3[31-0] ← MRs[63-48] × MRt[63-48];  
 MRd[15-0] ← TEMP0[31-16];  
 MRd[31-16] ← TEMP1[31-16];  
 MRd[47-32] ← TEMP2[31-16];  
 MRd[63-48] ← TEMP3[31-16];

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:

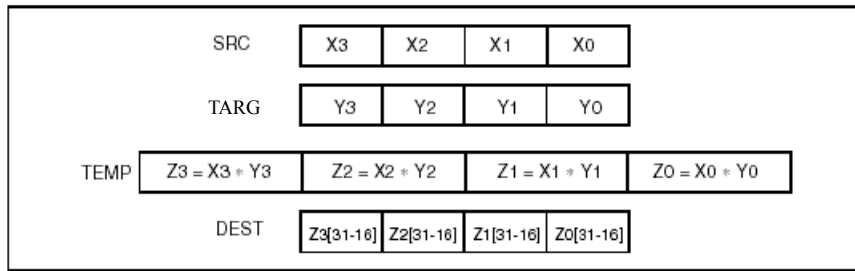
TEMP0[31-0] ← MRs[15-0] × MRt[15-0]; \* Unsigned multiplication \*  
 TEMP1[31-0] ← MRs[31-16] × MRt[31-16];  
 TEMP2[31-0] ← MRs[47-32] × MRt[47-32];  
 TEMP3[31-0] ← MRs[63-48] × MRt[63-48];  
 MRd[15-0] ← TEMP0[31-16];  
 MRd[31-16] ← TEMP1[31-16];  
 MRd[47-32] ← TEMP2[31-16];  
 MRd[63-48] ← TEMP3[31-16];

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:

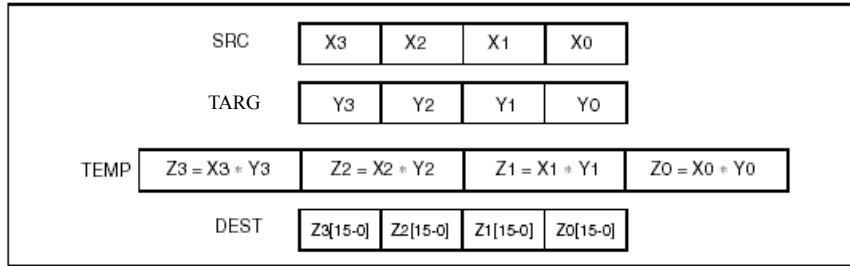
$TEMP0[31-0] \leftarrow MRs[15-0] \times MRt[15-0];$  \* Signed multiplication \*  
 $TEMP1[31-0] \leftarrow MRs[31-16] \times MRt[31-16];$   
 $TEMP2[31-0] \leftarrow MRs[47-32] \times MRt[47-32];$   
 $TEMP3[31-0] \leftarrow MRs[63-48] \times MRt[63-48];$   
 $MRd[15-0] \leftarrow TEMP0[15-0];$   
 $MRd[31-16] \leftarrow TEMP1[15-0];$   
 $MRd[47-32] \leftarrow TEMP2[15-0];$   
 $MRd[63-48] \leftarrow TEMP3[15-0];$

PMACLSD:

$TEMP0[31-0] \leftarrow MRs[15-0] \times MRt[15-0];$  \* Signed multiplication \*  
 $TEMP1[31-0] \leftarrow MRs[31-16] \times MRt[31-16];$   
 $TEMP2[31-0] \leftarrow MRs[47-32] \times MRt[47-32];$   
 $TEMP3[31-0] \leftarrow MRs[63-48] \times MRt[63-48];$   
 $MRd[15-0] \leftarrow MRd[15-0] + TEMP0[15-0];$   
 $MRd[31-16] \leftarrow MRd[31-16] + TEMP1[15-0];$   
 $MRd[47-32] \leftarrow MRd[47-32] + TEMP2[15-0];$   
 $MRd[63-48] \leftarrow MRd[63-48] + TEMP3[15-0];$

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];

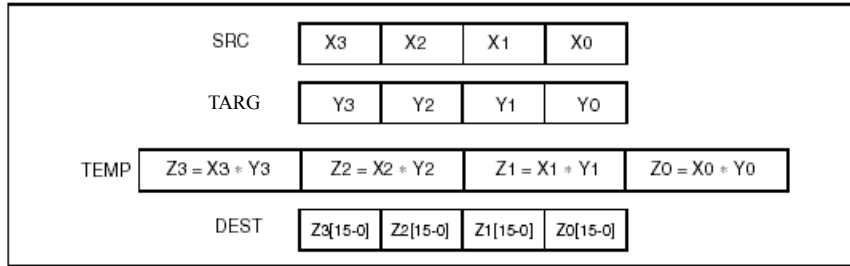
PMACLUD:

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:

$MRd \leftarrow MRd \text{ NOR } MRt;$

**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:

$MRd \leftarrow MRd \text{ OR } MRt;$

**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														

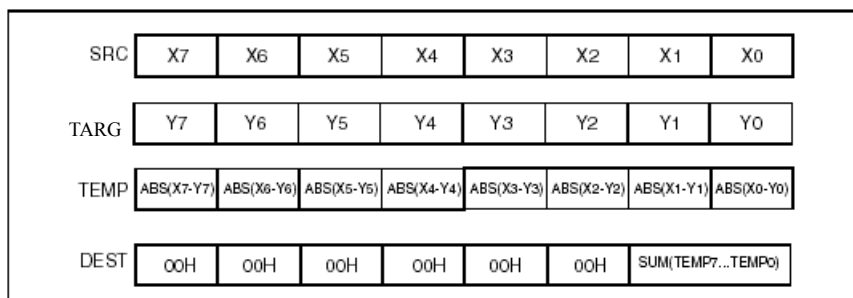
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
111111							11	MRs	disp	ARm	11	MRd	Modm				101001														

操作:

PSADBD instructions when using 64-bit operands:

TEMP0  $\leftarrow$  ABS(MRs] - MRt[7-0]);  
 \* repeat operation for bytes 2 through 6 \*;  
 TEMP7  $\leftarrow$  ABS(MRs[56] - MRt[63-56]);  
 MRd[15:0]  $\leftarrow$  SUM(TEMP0...TEMP7);  
 MRd[63:16]  $\leftarrow$  000000000000H;

描述: PSADBD 对 64-bit MRs 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 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		000000				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 2<sup>nd</sup>, 3<sup>rd</sup>, 4<sup>th</sup> double-bytes;

操作数说明: MRs: MDS 寄存器

MRt: MDS 寄存器

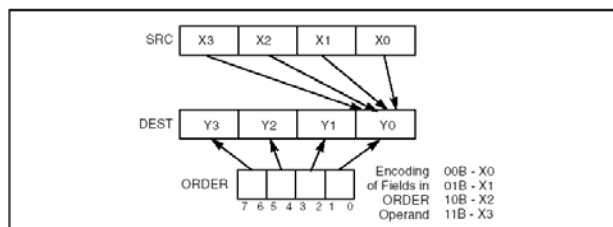
MRd: MDS 寄存器

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

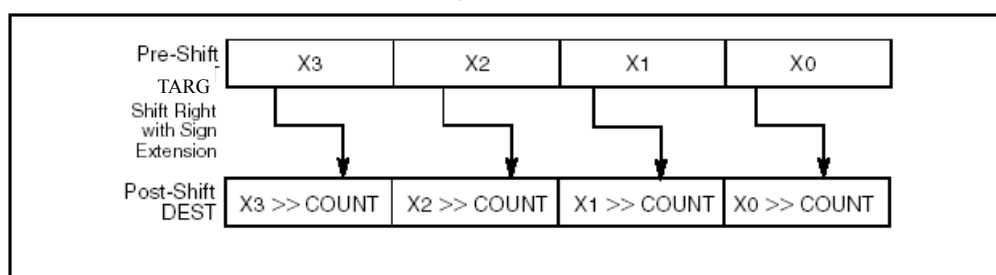
                  MRd:   MDS 寄存器

                  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

句型: **PSRLD/Q** *MRd, MRs, imm*

**PSRLD/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]				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

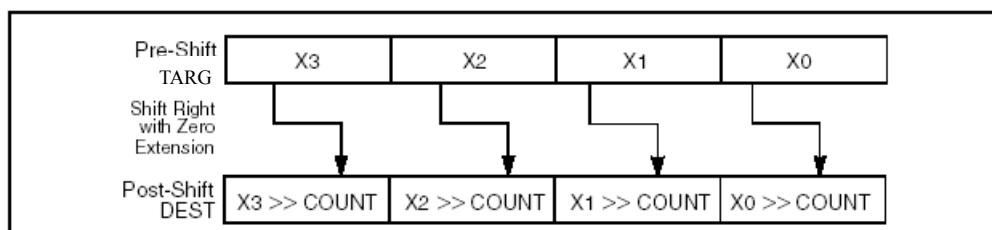
MRd: MDS寄存器

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

句型: **PSLLD/Q** *MRd, MRs, imm*  
**PSLLD/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]				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		000000				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

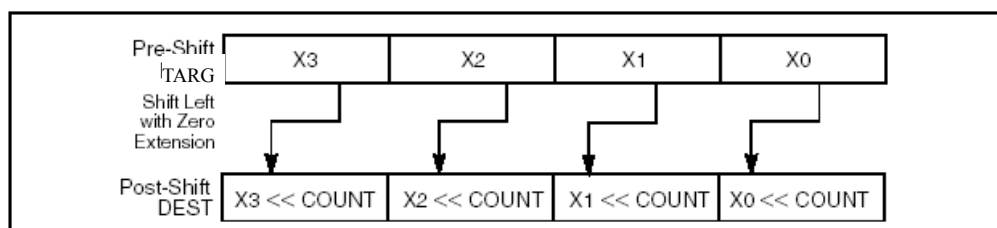
MRd: MDS 寄存器

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  $\leftarrow$  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:

$MRd[7..0] \leftarrow MRs[7..0] - MRt[7..0];$   
 \* repeat add operation for 2nd through 7th byte \*;  
 $MRd[63..56] \leftarrow MRs[63..56] - MRt[63..56];$

PSUBD instruction with 64-bit operands:

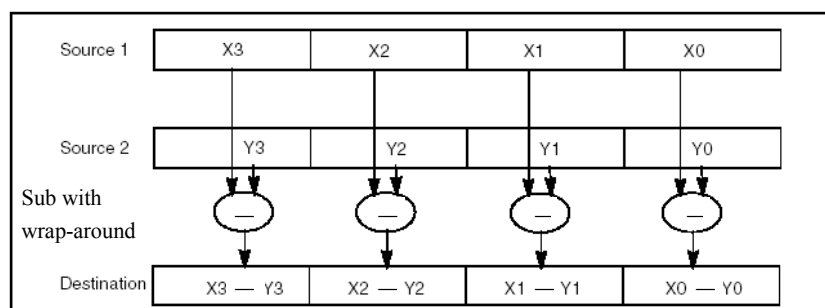
$MRd[15..0] \leftarrow MRs[15..0] - MRt[15..0];$   
 \* repeat add operation for 2nd and 3th double-byte \*;  
 $MRd[63..48] \leftarrow MRs[63..48] - MRt[63..48];$

PSUBQ instruction with 64-bit operands:

$MRd[31..0] \leftarrow MRs[31..0] - MRt[31..0];$   
 $MRd[63..32] \leftarrow MRs[63..32] - MRt[63..32];$

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*  
                   **PSUBSD**            *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:

$MRd[7..0] \leftarrow \text{SaturateToSignedByte}(MRs[7..0] - MRt[7..0]);$

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

$MRd[63..56] \leftarrow \text{SaturateToSignedByte}(MRs[63..56] - MRt[63..56]);$

PSUBSD instruction with 64-bit operands:

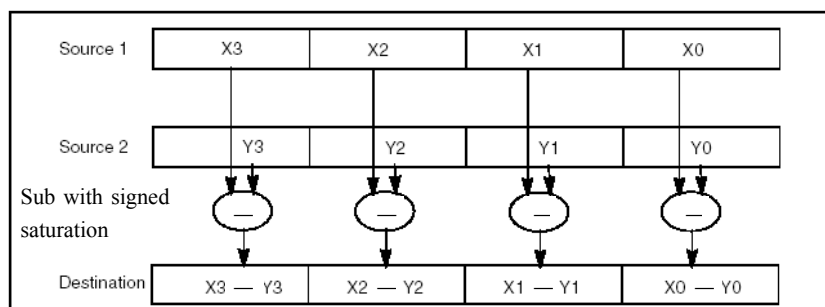
$MRd[15..0] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[15..0] - MRt[15..0]);$

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

$MRd[63..48] \leftarrow \text{SaturateToSignedDouble-byte}(MRs[63..48] - MRt[63..48]);$

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:

$MRd[7..0] \leftarrow \text{SaturateToUnsignedByte}(MRs[7..0] - MRt(7..0));$

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

$MRd[63..56] \leftarrow \text{SaturateToUnsignedByte}(MRs[63..56] - MRt[63..56]);$

PSUBUSD instruction with 64-bit operands:

$MRd[15..0] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[15..0] - MRt[15..0]);$

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

$MRd[63..48] \leftarrow \text{SaturateToUnsignedDouble-byte}(MRs[63..48] - MRt[63..48]);$

操作数说明: MRs: MDS 寄存器

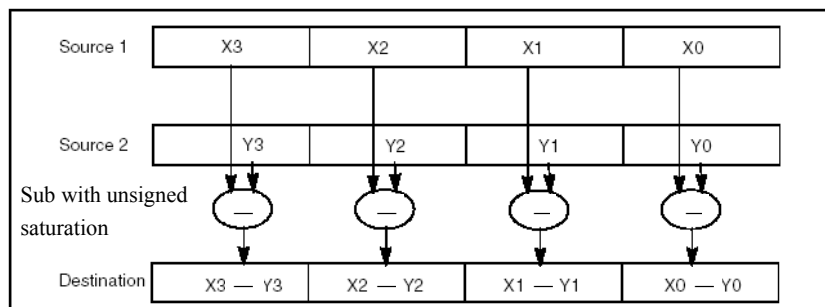
MRt: MDS 寄存器

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:

$MRd[7..0] \leftarrow MRs[39..32];$   
 $MRd[15..8] \leftarrow MRt[39..32];$   
 $MRd[23..16] \leftarrow MRs[47..40];$   
 $MRd[31..24] \leftarrow MRt[47..40];$   
 $MRd[39..32] \leftarrow MRs[55..48];$   
 $MRd[47..40] \leftarrow MRt[55..48];$   
 $MRd[55..48] \leftarrow MRs[63..56];$   
 $MRd[63..56] \leftarrow MRt[63..56];$

PUNPCKHDQ:

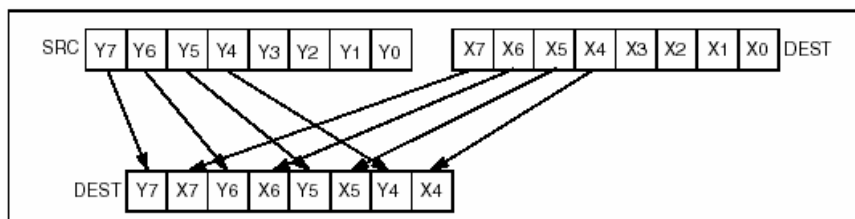
$MRd[15..0] \leftarrow MRs[47..32];$   
 $MRd[31..16] \leftarrow MRt[47..32];$   
 $MRd[47..32] \leftarrow MRs[63..48];$   
 $MRd[63..48] \leftarrow MRt[63..48];$

PUNPCKHQO:

$MRd[31..0] \leftarrow MRs[63..32]$   
 $MRd[63..32] \leftarrow MRt[63..32];$

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:

$MRd[63..56] \leftarrow MRt[31..24];$   
 $MRd[55..48] \leftarrow MRs[31..24];$   
 $MRd[47..40] \leftarrow MRt[23..16];$   
 $MRd[39..32] \leftarrow MRs[23..16];$   
 $MRd[31..24] \leftarrow MRt[15..8];$   
 $MRd[23..16] \leftarrow MRs[15..8];$   
 $MRd[15..8] \leftarrow MRt[7..0];$   
 $MRd[7..0] \leftarrow MRs[7..0];$

PUNPCKLDQ:

$MRd[63..48] \leftarrow MRt[31..16];$   
 $MRd[47..32] \leftarrow MRs[31..16];$   
 $MRd[31..16] \leftarrow MRt[15..0];$   
 $MRd[15..0] \leftarrow MRs[15..0];$

PUNPCKLQO:

$MRd[63..32] \leftarrow MRt[31..0];$   
 $MRd[31..0] \leftarrow MRs[31..0];$

操作数说明: MRs: MDS 寄存器

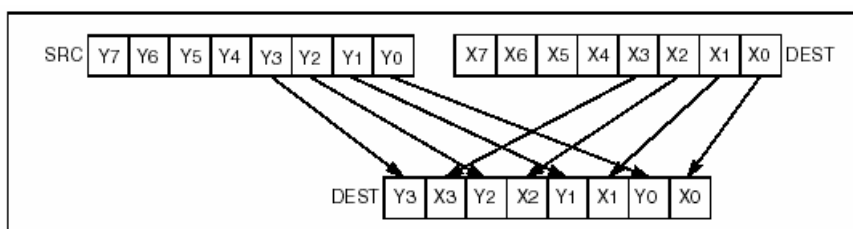
MRt: MDS 寄存器

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:

$MRd \leftarrow MRs \text{ XOR } MRt;$

**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

## 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:  $SR_{31..4} | SR_{5..2} \rightarrow SR$

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:

$\text{unsigned}(\text{direct} \times 4) + \text{PC} \rightarrow \text{RE}$   
 $(\text{PC of RPTB}) + 4 \rightarrow \text{RS}$

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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,   或者    $Vaddr = \text{Mod}(\text{ARm})$ ,  
Byte(GPR(rt))  $\rightarrow$  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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , Byte(GPR(R5)) $\rightarrow$ mem(Vaddr)
SB R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , Byte(GPR(R5)) $\rightarrow$ 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,    或者     $Vaddr = \text{Mod}(\text{ARm})$ ,  
 Byte(GPR(rt))  $\rightarrow$  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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	* -ARn(displ)	00001	* -ARn(IR0)
		01001	* -ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , Byte(GPR(R5)) $\rightarrow$ mem(Vaddr)
SH R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , Byte(GPR(R5)) $\rightarrow$ mem(Vaddr)

## SHOWBITS

句型:       **SHOWBITS**   rd, ra, sa       或者  
              **SHOWBITS**   rd, ra, rn

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	0	留	00	ra	sa	11	rd	sa			001111														

31	30	29	28	27	26	25	24	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	1	留	00	ra	sa	11	rd	rn			001111														

操作:

SHOWBITS:

n = sa or ValueOfReg(rn)

IF (n > 32)

THEN n ← 32;

rd = HeadBitsOfVLDbuffer( n );

操作数说明: rd:       寄存器   (通用寄存器 0~7)  
              ra:       寄存器   (辅助寄存器 0~3)  
              sa:       立即数  
              rn:       寄存器   (通用寄存器 0~31)

描述:       VLD 指令 ShowBits 完成的操做是从码流中取出 n 个比特, 但是码流的指针并不改变。其中 rd 是目标寄存器, ra 是地址寄存器, rn 是取比特数寄存器, sa 是取比特立即数。

执行周期: 1 cycle

## SLL

句型:       **SLL**    rd, rt, sa                                或者  
               **SLL**    rd, rt, \*+ARm(dis)                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	15	14	13	12	11	10	9	8	7	6	5	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	0
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(dis)) → 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(dis)
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:	<b>SLLV</b>	rd, rt, rs	或者
	<b>SLLV</b>	dst, *+ARn(dis2), *+ARm(dis1)	或者
	<b>SLLV</b>	dst, rt, mod(ARm)	或者
	<b>SLLV</b>	dst, Imm, *+ARm(dis2)	或者
	<b>SLLV</b>	dst, mod(ARm), rs	或者
	<b>SLLV</b>	dst, *+ARm(dis2), rs	或者
	<b>SLLV</b>	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						Displ		ARm		Displ		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(dis2)) << Mem(*+ARm(dis1))[4:0] → GPR(dst)	或者
	GPR(Rt) << mod(ARm)[4:0] → GPR(dst)	或者
	sign(Imm) << Mem(*+ARm(dis2)) → GPR(dst)	或者
	mod(ARm) << GPR(Rs) → GPR(dst)	或者
	mem(*+ARm(dis2)) << 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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:	<b>SLT</b>	rd, rs, rt	或者
	<b>SLT</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>SLT</b>	dst, mod(ARm), rt	或者
	<b>SLT</b>	dst, *+ARm(displ), Imm	或者
	<b>SLT</b>	dst, rs, mod(ARm)	或者
	<b>SLT</b>	dst, rs, *+ARm(displ)	或者
	<b>SLT</b>	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						Displ		ARm		Displ		ARn		00		Dst		Displ2					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(displ)) < mem(*+ARn(displ2)) 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(displ)) < 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(displ)) 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	* -ARn(displ)	00001	* -ARn(IR0)
		01001	* -ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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(displ), *+ARn(displ2)	或者
	SLTU	dst, mod(ARm), rt	或者
	SLTU	dst, *+ARm(displ), Imm	或者
	SLTU	dst, rs, mod(ARm)	或者
	SLTU	dst, rs, *+ARm(displ)	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						Displ		ARm		Displ		ARn		00		Dst		Displ2		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(displ)) < mem(*+ARn(displ2)) 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(displ)) < 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(displ)) 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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(dis)                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(dis)) → 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(dis)
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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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:	<b>SRAV</b>	rd, rt, rs	或者
	<b>SRAV</b>	dst, *+ARn(dis2), *+ARm(dis1)	或者
	<b>SRAV</b>	dst, rt, mod(ARm)	或者
	<b>SRAV</b>	dst, Imm, *+ARm(dis2)	或者
	<b>SRAV</b>	dst, mod(ARm), rs	或者
	<b>SRAV</b>	dst, *+ARm(dis2), rs	或者
	<b>SRAV</b>	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(dis2)) >> Mem(*+ARm(dis1))[4:0] → GPR(dst)	或者
	GPR(Rt) >> mod(ARm)[4:0] → GPR(dst)	或者
	sign(Imm) >> Mem(*+ARm(dis2)) → GPR(dst)	或者
	mod(ARm) >> GPR(Rs) → GPR(dst)	或者
	mem(*+ARm(dis2)) >> 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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(dis)                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(dis)) → 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(dis)
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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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:	<b>SRLV</b>	rd, rt, rs	或者
	<b>SRLV</b>	dst, *+ARn(dis2), *+ARm(dis1)	或者
	<b>SRLV</b>	dst, rt, mod(ARm)	或者
	<b>SRLV</b>	dst, Imm, *+ARm(dis2)	或者
	<b>SRLV</b>	dst, mod(ARm), rs	或者
	<b>SRLV</b>	dst, *+ARm(dis2), rs	或者
	<b>SRLV</b>	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(dis2)) >> Mem(*+ARm(dis1))[4:0] → GPR(dst)	或者
	GPR(Rt) >> mod(ARm)[4:0] → GPR(dst)	或者
	sign(Imm) >> Mem(*+ARm(dis2)) → GPR(dst)	或者
	mod(ARm) >> GPR(Rs) → GPR(dst)	或者
	mem(*+ARm(dis2)) >> 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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:	<b>SUB</b>	rd, rs, rt	或者
	<b>SUB</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>SUB</b>	dst, mod(ARm), rt	或者
	<b>SUB</b>	dst, *+ARm(displ), Imm	或者
	<b>SUB</b>	dst, rs, mod(ARm)	或者
	<b>SUB</b>	dst, rs, *+ARm(displ)	或者
	<b>SUB</b>	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						Displ		ARm		Displ		ARn		00		dst		Displ2					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(displ)) - mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) - GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) - sign(Imm) → GPR(dst)	或者
	GPR(Rs) - mod(ARm) → GPR(dst)	或者
	GPR(Rs) - mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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:	<b>SUBU</b>	rd, rs, rt	或者
	<b>SUBU</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>SUBU</b>	dst, mod(ARm), rt	或者
	<b>SUBU</b>	dst, *+ARm(displ), Imm	或者
	<b>SUBU</b>	dst, rs, mod(ARm)	或者
	<b>SUBU</b>	dst, rs, *+ARm(displ)	或者
	<b>SUBU</b>	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						Displ		ARm		Displ		ARn		00		Dst		Displ2				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(displ)) - mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) - GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) - sign(Imm) → GPR(dst)	或者
	GPR(Rs) - mod(ARm) → GPR(dst)	或者
	GPR(Rs) - mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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
110100							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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,    或者     $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{GPR}(\text{rt}) \rightarrow \text{mem}(Vaddr)$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{GPR}(R5) \rightarrow \text{mem}(Vaddr)$
SW R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{GPR}(R5) \rightarrow \text{mem}(Vaddr)$



## 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ , 或者  $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{Left}(\text{GPR}(\text{rt})) \rightarrow \text{mem}(Vaddr)$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{left}(\text{GPR}(R5)) \rightarrow \text{mem}(Vaddr)$
SWL R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{left}(\text{GPR}(R5)) \rightarrow \text{mem}(Vaddr)$

## 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:

$Vaddr = \text{sign}(\text{offset}) + \text{GPR}(\text{base})$ ,    或者     $Vaddr = \text{Mod}(\text{ARm})$ ,  
 $\text{Right}(\text{GPR}(\text{rt})) \rightarrow \text{mem}(Vaddr)$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	* -ARn(displ)	00001	* -ARn(IR0)
		01001	* -ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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)	$Vaddr = \text{sign}(0840h) + \text{GPR}(R3)$ , $\text{Right}(\text{GPR}(R5)) \rightarrow \text{mem}(Vaddr)$
SWR R5, *AR0--(IR0)	$Vaddr = *AR0--(IR0)$ , $\text{Right}(\text{GPR}(R5)) \rightarrow \text{mem}(Vaddr)$

## 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: SystemCallException

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]<sub>127..96</sub> → PageMask;

TLB[Index]<sub>95..64</sub> → EntryHi ;

TLB[Index]<sub>63..32</sub> → EntryLo1;

TLB[Index]<sub>31..0</sub> → 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:	<b>XOR</b>	rd, rs, rt	或者
	<b>XOR</b>	dst, *+ARm(displ), *+ARn(displ2)	或者
	<b>XOR</b>	dst, mod(ARm), rt	或者
	<b>XOR</b>	dst, *+ARm(displ), Imm	或者
	<b>XOR</b>	dst, rs, mod(ARm)	或者
	<b>XOR</b>	dst, rs, *+ARm(displ)	或者
	<b>XOR</b>	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						Displ		ARm		Displ		ARn		00		Dst		Displ2		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(displ)) xor mem(*+ARn(displ2)) → GPR(dst)	或者
	mod(ARm) xor GPR(Rt) → GPR(dst)	或者
	Mem(*+ARm(displ)) xor sign(Imm) → GPR(dst)	或者
	GPR(Rs) xor mod(ARm) → GPR(dst)	或者
	GPR(Rs) xor mem(*+ARm(displ)) → 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(displ)寻址	*+ARn(displ)寻址
01	E=00	间接寻址	寄存器
	E=01	*+ARn(displ)寻址	立即数
10	E=00	寄存器	间接寻址
	E=01	寄存器	*+ARn(displ)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：

$GPR(Rs) \text{ xor zero}(Imm) \rightarrow GPR(rt)$       或者  
 $GPR(dst) \text{ xor mem}(Imm) \rightarrow GPR(dst)$       或者  
 $GPR(dst) \text{ xor mod}(ARm) \rightarrow GPR(dst)$

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(displ)	00000	*+ARn(IR0)
		01000	*+ARn(IR1)
10001	*-ARn(displ)	00001	*-ARn(IR0)
		01001	*-ARn(IR1)
10010	*++ARn(displ)	00010	*++ARn(IR0)
		01010	*++ARn(IR1)
10011	*--ARn(displ)	00011	*--ARn(IR0)
		01011	*--ARn(IR1)
10100	*ARn++(displ)	00100	*ARn++(IR0)
		01100	*ARn++(IR1)
10101	*ARn--(displ)	00101	*ARn--(IR0)
		01101	*ARn--(IR1)
10110	*ARn++(displ)%	00110	*ARn++(IR0)%
		01110	*ARn++(IR1)%
10111	*ARn--(displ)%	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	$GPR(R3) \text{ xor zero}(0840h) \rightarrow GPR(R5)$
XORI    R5, @0840h	$GPR(R5) \text{ xor mem}(0840h) \rightarrow GPR(R5)$
XORI    R5, *AR2++(40h)	$GPR(R5) \text{ xor mem}(AR2) \rightarrow GPR(R5), AR2=AR2+40h$

## 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
110100							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。
- 2004, Feb.19 经过赖莉雅的反馈,进行了更新。
- 2004, Feb,19 RISC3200 的指令集请参考 MDF、MDS 指令。
- 2007, May 张奇修改了部分错误,同时根据小组讨论结果,进行 MediaDSP ISA 指令的整理,形成了指令体系。Ver1.0
- 2007, May,31 修改了 VLD 类指令和 PI 类指令的编码,避免与 MIPS 指令冲突。