

附录1、MIPS寄存器

编号	助记符	功能
0	\$zero	常量0，永远返回值为0
1	\$at	保留给汇编器作为暂时变量(Reserved for assembler)
2-3	\$v0-\$v1	保存过程调用返回的结果(values for results and expression evaluation)。
4-7	\$a0-\$a3	用作过程调用的参数(argument)。更多的参数采用栈传递。
8-15	\$t0-\$t7	暂时寄存器(temporary register)。过程使用它们时不需要保存与恢复
16-23	\$s0-\$s7	保存寄存器(saved register)。过程必须保存和恢复使用过的保存寄存器。
24-25	\$t8-\$t9	暂时寄存器(temporary register)。
26-27	\$k0-\$k1	通常被中断或异常处理程序使用作为保存一些系统参数。
28	\$gp	全局指针(global pointer)，指向保存"static"和"extern"变量的静态数据区。
29	\$sp	堆栈指针(stack pointer)，指向栈顶。
30	\$fp/\$s8	帧指针(frame pointer)，指向当前过程在栈中的帧的基址
31	\$ra	保存调用过程的返回地址(return address)
	cp0	在MIPS体系结构中，最多支持4个协处理器(Co-Processor)。其中，协处理器CPO是体系结构中必须实现的。它起到控制CPU的作用。MMU、异常处理、乘除法等功能，都依赖于协处理器CPO来实现。它是MIPS的精髓之一，也是打开MIPS特权级模式的大门。

- **\$0:** 即\$zero,该寄存器总是返回零, 为0这个有用常数提供了一个简洁的编码形式。

move \$t0,\$t1

实际为

add \$t0,\$0,\$t1

使用伪指令可以简化任务, 汇编程序提供了比硬件更丰富的指令集。

- **\$1:**即\$at, 该寄存器为汇编保留, 由于I型指令的立即数字段只有16位, 在加载一个32位立即数需要 lui (装入高位立即数) 和addi两条指令。汇编器可以用\$at来保存大常数来完成。

例子: 指令li \$t1,40是一条伪指令, 在汇编器中会转换成addi \$t1,\$zero,40

但是, 指令 li \$t1,-4000000 因为数字太大, 需要拆开, 则会被转换成

lui \$at,0xffc2

ori \$t1,\$at,0xf700

伪指令la \$a0, error的功能与上述li类似。

- **\$2..\$3:(\$v0-\$v1)**用于子程序的非浮点结果或返回值。MIPS约定堆栈中少数几个位置保留未做定义, 可以用于存放更多的返回值。
- **\$4..\$7:(\$a0-\$a3)**用来传递前四个参数给子程序, 不够的用堆栈。MIPS编译器总是为参数在堆栈中留有空间以防有参数需要存储。
- **\$8..\$15:(\$t0-\$t7)**临时寄存器, 子程序可以使用它们而不用保留。
- **\$16..\$23:(\$s0-\$s7)**保存寄存器, 被调用者需要保存和恢复使用的保存寄存器, 目的是减少了寄存器溢出(spilling), 即将不常用的变量放到存储器的过程。叶过程只有在临时寄存器不够用时使用保存寄存器。

- **\$24..\$25:(\$t8-\$t9):** 同\$t0-\$t7。
- **\$26..\$27:(\$k0,\$k1)**为操作系统/异常处理保留，至少要预留一个。
- **\$28:(\$gp)** 全局指针gp(global pointer)指向静态数据区，在存取位于gp值上下32KB范围内的数据时，只需要一条以gp为基指针的指令即可。
- **\$29:(\$sp)** MIPS硬件并不直接支持堆栈，堆栈是在内存中的一块区域。
- **\$30:(\$fp)** GNU MIPS C编译器使用了帧指针(frame pointer), 而SGI的C编译器没有使用，而把这个寄存器当作保存寄存器使用(\$s8),这节省了调用和返回开销，但增加了代码生成的复杂性。
- **\$31:(\$ra)**存放返回地址，MIPS的jal(jump-and-link)指令跳转到某个过程时，会把下一条指令的地址放到\$ra中。过程中执行jr \$ra返回。
- **cp0:(control processor)** 当执行指令发生异常时那条指令的地址会被保存在EPC(exception program counter)中。指令mfc0 (move from system control)可以将EPC中的地址复制到k0或k1中，再用jr跳转到异常指令处继续执行。

附录2、MIPS32汇编指令

CPU Arithmetic Instructions

助记符	指令说明	助记符	指令说明
ADD	Add Word	SLT	Set on Less Than
ADDI	Add Immediate Word		
ADDIU	Add Immediate Unsigned Word	SLTI	Set on Less Than Immediate
ADDU	Add Unsigned Word	SLTIU	Set on Less Than Immediate Unsigned
SUB	Subtract Word	SLTU	Set on Less Than Unsigned
SUBU	Subtract Unsigned Word	DIV	Divide Word
MUL	Multiply Word to GPR	DIVU	Divide Unsigned Word
MULT	Multiply Signed Word		
MULTU	Multiply Unsigned Word		

* GPR -- General Purpose Register

CPU Branch and Jump Instructions

助记符	指令说明	助记符	指令说明
BEQ	Branch on Equal	JAL	Jump and Link
BNE	Branch on Not Equal	JR	Jump Register
J	Jump		

CPU Load, Store, and Memory Control Instructions

助记符	指令说明	助记符	指令说明
LB	Load Byte	SB	Store Byte
LBU	Load Byte Unsigned	SH	Store Halfword
LH	Load Halfword	SW	Store Word
LHU	Load Halfword Unsigned	SC	Store Conditional Word
LW	Load Word		
LL	Load Linked Word		

* LBU和LB - 分别采用有符号和无符号扩展

CPU Logical Instructions

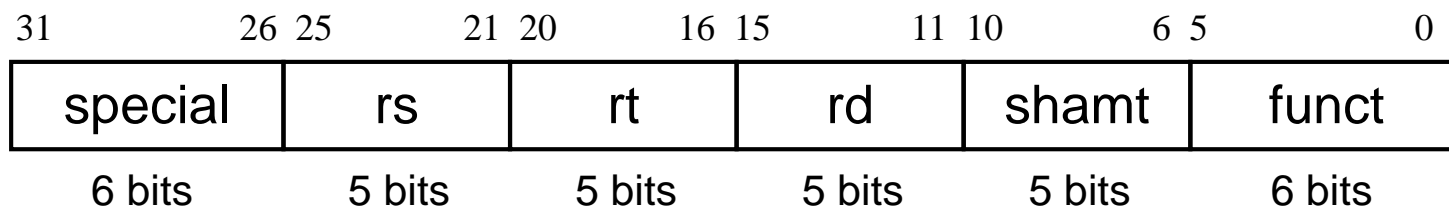
助记符	指令说明	助记符	指令说明
AND	And	XOR	Exclusive Or
ANDI	And Immediate	XORI	Exclusive Or Immediate
OR	Or	NOR	Not Or
ORI	Or Immediate	LUI	Load Upper Immediate

CPU Shift Instructions

助记符	指令说明	助记符	指令说明
SLL	Shift Word Left Logical		
SRL	Shift Word Right Logical		

附录3、MIPS32机器指令

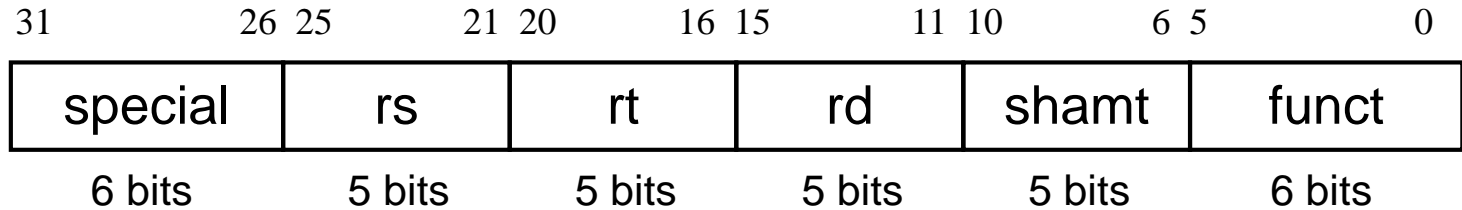
R-format Instructions (1)



format	description	special	shamt	funct
ADD rd, rs, rt	$rd \leftarrow rs + rt$	000000	00000	100000
ADDU rd, rs, rt	$rd \leftarrow rs + rt$	000000	00000	100001
SUB rd, rs, rt	$rd \leftarrow rs - rt$	000000	00000	100010
SUBU rd, rs, rt	$rd \leftarrow rs - rt$	000000	00000	100011
MUL rd, rs, rt	$rd \leftarrow rs \times rt$	011100	00000	000010
MULT rs, rt	$(HI, LO) \leftarrow rs \times rt$	000000	00000	011000
MULTU rs, rt	$(HI, LO) \leftarrow rs \times rt$	000000	00000	011001
DIV rs, rt	$(HI, LO) \leftarrow rs / rt$	000000	00000	011010
DIVU rs, rt	$(HI, LO) \leftarrow rs / rt$	000000	00000	011011

- * DIV执行结果的余数和商分别保存在32位内部寄存器HI和LO中，它们要用MFHI和MFLO才能取出内容。
- * 指令中确实的rs、rt或rd的值位000000。
- * ADDU不会产生溢出例外（exception），而ADD则会，溢出时ADD不会把结果保存在rd中。

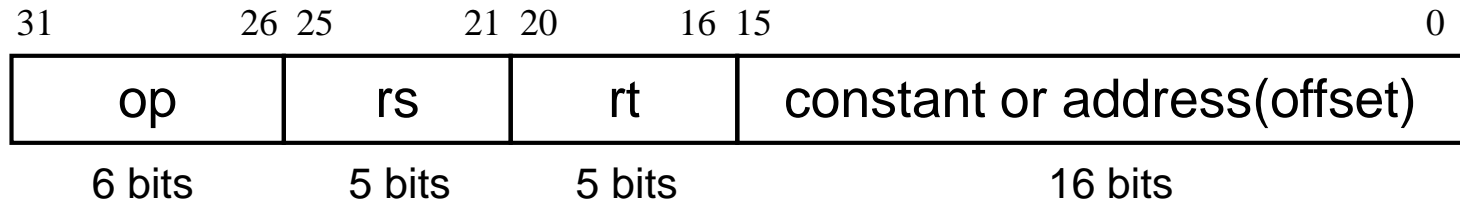
R-format Instructions (2)



format	description	special	shamt	funct
AND rd, rs, rt	$rd \leftarrow rs \text{ AND } rt$	000000	00000	100100
OR rd, rs, rt	$rd \leftarrow rs \text{ or } rt$	000000	00000	100101
NOR rd, rs, rt	$rd \leftarrow rs \text{ NOR } rt$	000000	00000	100111
XOR rd, rs, rt	$rd \leftarrow rs \text{ XOR } rt$	000000	00000	100110
SLL rd, rt, sa	$rd \leftarrow rt \ll sa \text{ (logical)}$	000000	sa	000000
SRL rd, rt, sa	$rd \leftarrow rt \gg sa \text{ (logical)}$	000000	sa	000010
SLT rd, rs, rt	$rd \leftarrow (rs < rt)$	000000	00000	101010
SLTU rd, rs, rt	$rd \leftarrow (rs < rt)$	000000	00000	101011
JR rs	go to an address in a register	000000	00000	001000
MFHI rd	$rd \leftarrow HI$	000000	00000	010000
MFLO rd	$rd \leftarrow LO$	000000	00000	010010

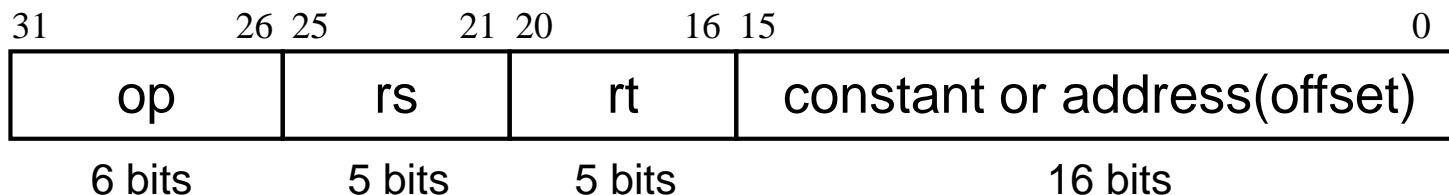
* mthi/mtlo: $rd \leftarrow HI/LO$

I-format Instructions (1)



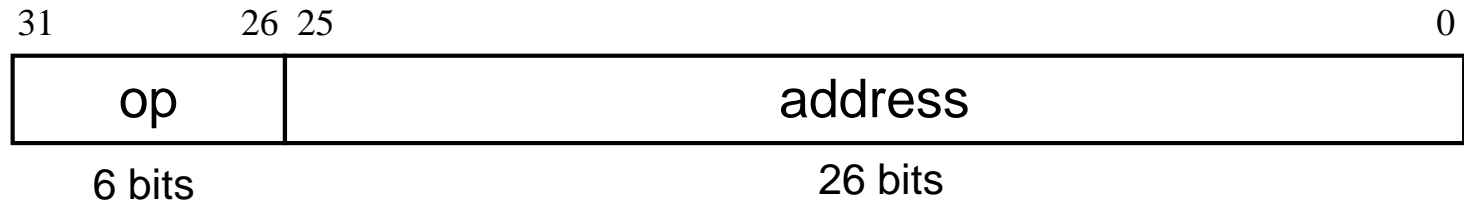
format	description	op
ADDI rt, rs, immediate	$rt \leftarrow rs + \text{immediate}$	001000
ADDIU rt, rs, immediate	$rt \leftarrow rs + \text{immediate}$	001001
ANDI rt, rs, immediate	$rt \leftarrow rs \text{ AND immediate}$	001100
ORI rt, rs, immediate	$rt \leftarrow rs \text{ or immediate}$	001101
XORI rt, rs, immediate	$rt \leftarrow rs \text{ XOR immediate}$	001110
SLTI rt, rs, immediate	$rt \leftarrow (rs < \text{immediate})$	001010
SLTIU rt, rs, immediate	$rt \leftarrow (rs < \text{immediate})$	001011
BEQ rs, rt, offset	if $rs = rt$ then branch ($PC \leftarrow PC + \text{offset} \times 4$)	000100
BNE rs, rt, offset	if $rs \neq rt$ then branch ($PC \leftarrow PC + \text{offset} \times 4$)	000101

I-format Instructions (2)



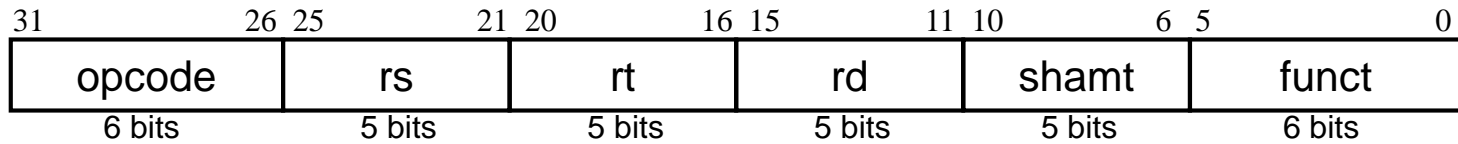
format	description	op
LW rt, offset(base)	$rt \leftarrow \text{memory}[\text{base}(rs)e+\text{offset}]$	100011
SW rt, offset(base)	$\text{memory}[\text{base}(rs)+\text{offset}] \leftarrow rt$	101011
LB rt, offset(base)	$rt \leftarrow \text{memory}[\text{base}(rs)+\text{offset}]$ (8位, 进行有符号扩展)	100000
LBU rt, offset(base)	$rt \leftarrow \text{memory}[\text{base}(rs)+\text{offset}]$ (8位, 进行无符号扩展)	100100
LH rt, offset(base)	$rt \leftarrow \text{memory}[\text{base}(rs)+\text{offset}]$ (16位, 进行有符号扩展)	100001
LHU rt, offset(base)	$rt \leftarrow \text{memory}[\text{base}(rs)+\text{offset}]$ (16位, 进行无符号扩展)	100101
SB rt, offset(base)	$\text{memory}[\text{base}(rs)+\text{offset}] \leftarrow rt$ (保存最低有效字节)	101000
SH rt, offset(base)	$\text{memory}[\text{base}+\text{offset}] \leftarrow rt$ (half word) (保存最低有效16位)	101001
LUI rt, immediate	$rt \leftarrow \text{immediate}(16\text{bits}) + 16's\ 0$ (把16位立即数送入高16位)	001111
LL rt, offset(base)	$rt \leftarrow \text{memory}[\text{base}(rs)+\text{offset}]$ (Load and Link)	110000
SC rt, offset(base)	if atomic_update (Store Conditionally) then $\text{memory}[\text{base}(rs)+\text{offset}] \leftarrow rt, rt \leftarrow 1$ else $rt \leftarrow 0$	111000

J-format Instructions

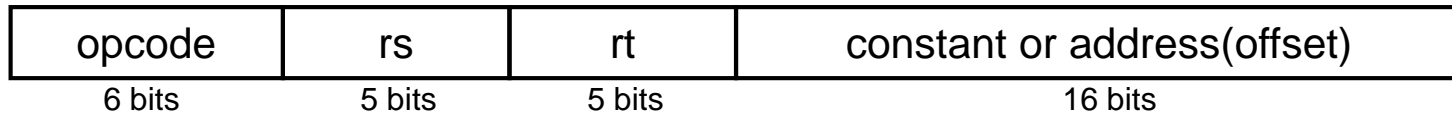


format	description	op
J target	To branch within the current 256 MB-aligned region	000010
JAL target	To execute a procedure call within the current 256 MB-aligned region	000011

R-format Instructions



I-format Instructions



J-format Instructions

