MIPS-C 指令集 (版本: 1.0)

文档编号: COCO01

高小鹏 北京航空航天大学计算机学院 2014年

修订记录

正式发布 1.0 版。

目录

A	.1 MI	IPS-C 指令表5
A	.2 MI	IPS-C 指令图7
A	.3 指·	令详解(按字母排列)8
	1.	ADD: 符号加8
	2.	ADDI: 符号加立即数8
	3.	ADDIU: 无符号加立即数8
	4.	ADDU: 无符号加9
	5.	AND: 与9
	6.	ANDI: 与立即数9
	7.	BEQ: 相等时转移10
	8.	BGEZ: 大于等于 0 时转移10
	9.	BGTZ: 大于 0 时转移10
	10.	BLEZ: 小于等于 0 时转移10
	11.	BLTZ: 小于 0 时转移11
	12.	BNE:不等于时转移11
	13.	BREAK: 断点11
	14.	DIV: 符号除12
	15.	DIVU: 无符号除
	16.	ERET: 异常返回12
	17.	J: 跳转 13
	18.	JAL: 跳转并链接13
	19.	JALR: 跳转并链接 13
	20.	JR: 跳转至寄存器 14
	21.	LB: 加载字节14
	22.	LBU: 加载无符号字节 14
	23.	LH:加载半字14

24.	LHU: 加载无符号半字15
25.	LUI: 立即数加载至高位15
26.	LW: 加载字
27.	MFC0: 读 CP0 寄存器16
28.	MFHI: 读 HI 寄存器16
29.	MFLO:读 LO 寄存器16
30.	MTCO: 写 CPO 寄存器
31.	MTHI: 写 HI 寄存器 17
32.	MTLO: 写 LO 寄存器17
33.	MULT: 符号乘
34.	MULTU: 无符号乘18
35.	NOR: 或非
36.	OR: 或
37.	ORI: 或立即数
38.	SB: 存储字节19
39.	SH: 存储半字节19
40.	SLL: 逻辑左移19
41.	SLLV: 逻辑可变左移20
42.	SLT: 小于置 1(有符号)20
43.	SLTI: 小于立即数置 1(有符号)20
44.	SLTIU: 小于立即数置 1(无符号)
45.	SLTU: 小于置 1(无符号)
46.	SRA: 算术右移
47.	SRAV: 算术可变右移21
48.	SRL: 逻辑右移
49.	SRLV: 逻辑可变右移22
50.	SUB: 符号减22
51.	SUBU: 无符号减23
52.	SW:存储字23
53.	SYSCALL: 系统调用

54.	XOR: 异或	. 24
55.	XORI: 异或立即数	. 24

A.1 MIPS-C 指令表

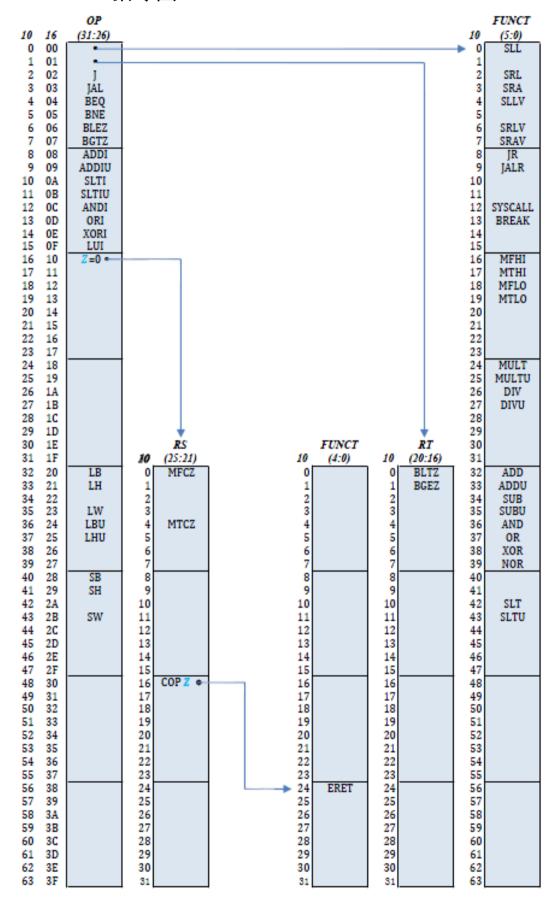
本附录从 MIPS32 指令集中选择了一些常用指令构成了 MIPS-C 指令集。MIPS-C 可以支持除浮点运算外的绝大多数定点类程序的运行,并且提供了包括 CPO、异常处理等指令,可以支持简单的操作系统的运行。MIPS-C 指令集共包括 55 条指令。从更细致的功能角度,MIPS-C 被划分为 9 个子类。

功能分类	助记符	功能	OPCODE/ FUNCT (16 进制)	操作 (VerilogHDL 语法描述)
	LB	加载字节	20Н	<pre>R[rt] = {24{Mem[GPR[rs]+sign_ext(offset)][7]},</pre>
	LBU	加载字节 (无符号)	24H	R[rt] = {24'b0, Mem[GPR[rs]+ sign_ext(offset)][7:0]}
加载	LH	加载半字	21H	<pre>R[rt] = {16{Mem[GPR[rs]+sign_ext(offset)][15]},</pre>
	LHU	加载半字 (无符号)	25H	R[rt] = {16'b0, Mem[GPR[rs]+ sign_ext(offset)][15:0]}
	LW	加载字	23Н	R[rt] = Mem[GPR[rs]+sign_ext(offset)]
_	SB	存储字节	28H	Mem[GPR[rs]+sign_ext(offset)][7:0] = R[rt][7:0]
保存	SH	存储半字	29Н	Mem[GPR[rs]+sign_ext(offset)][15:0] = R[rt][15:0]
	SW	存储字	2ВН	Mem[GPR[rs]+sign_ext(offset)] = R[rt]
_	ADD	加	0/20H	GPR[rd] = GPR[rs] + GPR[rt]
	ADDU	无符号加	0/21H	GPR[rd] = GPR[rs] + GPR[rt]
	SUB	减	0/22Н	GPR[rd] = GPR[rs] - GPR[rt]
	SUBU	无符号减	0/23Н	GPR[rd] = GPR[rs] - GPR[rt]
	MULT	乘	0/18Н	{HI, LO} = GPR[rs] × GPR[rt]
	MULTU	乘(无符号)	0/19Н	{HI, LO} = GPR[rs] × GPR[rt]
	DIV	除	0/1AH	{HI, LO} = GPR[rs] / GPR[rt]
	DIVU	除(无符号)	0/1BH	{HI, LO} = GPR[rs] / GPR[rt]
	SLT	小于置 1	0/2AH	GPR[rd] = (GPR[rs] < GPR[rt]) ? 1:0
R-R 运算	SLTU	小于置 1 (无符号)	0/2BH	GPR[rd] = (GPR[rs] < GPR[rt]) ? 1:0
心 异	SLL	逻辑左移	0/0H	GPR[rd] = {GPR[rt][31-s:0], s{0}}
	SRL	逻辑右移	0/2H	$GPR[rd] = {s{0}, GPR[rt][31:s]}$
	SRA	算术右移	0/3Н	GPR[rd] = {s{GPR[rt][31]}, GPR[rt][31:s]}
	SLLV	逻辑可变左移	0/4H	$GPR[rd] = \{GPR[rt][31-v:0], v\{0\}\}$
_	SRLV	逻辑可变右移	0/6H	$GPR[rd] = \{v\{0\}, GPR[rt][31:v]\}$
_	SRAV	算术可变右移	0/7H	GPR[rd] = {v{GPR[rt][31]}, GPR[rt][31:v]}
	AND	与	0/24H	GPR[rd] = GPR[rs] & GPR[rt]
	OR	或	0/25Н	GPR[rd] = GPR[rs] GPR[rt]
	XOR	异或	0/26Н	GPR[rd] = GPR[rs] ^ GPR[rt]
	NOR	或非	0/27Н	GPR[rd] = ~(GPR[rs] GPR[rt])
R-I	ADDI	加立即数	8Н	GPR[rt] = GPR[rs] + SignExt(Imm)

运算	ADDIU	加立即数 (无符号)	9н	GPR[rt] = GPR[rs] + SignExt(Imm)
	ANDI	与立即数	СН	GPR[rt] = GPR[rs] & ZeroExt(Imm)
	ORI	或立即数	DH	GPR[rt] = GPR[rs] ZeroExt(Imm)
	XORI	异或立即数	EH	GPR[rt] = GPR[rs] ^ ZeroExt(Imm)
	LUI	立即数加载至高 位	FH	GPR[rt] = {imm, 16'b0}
	SLTI	小于立即数置1	АН	GPR[rt] = (GPR[rs] < SignExt(Imm)) ? 1 : 0
	SLTIU	小于立即数置 1 (无符号)	ВН	GPR[rt] = (GPR[rs] < SignExt(Imm)) ? 1 : 0
	BEQ	等于转移	4H	<pre>if (GPR[rs] == GPR[rt]) PC = PC + 4 + BranchAddr</pre>
	BNE	不等转移	5Н	<pre>if (GPR[rs] != GPR[rt]) PC = PC + 4 + BranchAddr</pre>
/\ 	BLEZ	小于等于0转移	6Н	<pre>if (GPR[rs] <= 0) PC = PC + 4 + BranchAddr</pre>
分支	BGTZ	大于0转移	7н	<pre>if (GPR[rs] > 0) PC = PC + 4 + BranchAddr</pre>
	BLTZ	小于0转移	特殊编码①	<pre>if (GPR[rs] <0) PC = PC + 4 + BranchAddr</pre>
	BGEZ	大于等于0转移	特殊编码②	<pre>if (GPR[rs] >= 0) PC = PC + 4 + BranchAddr</pre>
	J	跳转	2Н	PC = JumpAddr
	JAL	跳转并链接	3н	PC = JumpAddr; GPR[31] = PC + 4
跳转	JALR	跳转并链接寄存 器	0/9н	PC = GPR[rs]; GPR[rd] = PC + 4
	JR	跳转寄存器	0/8H	PC = GPR[rs]
	MFHI	读 HI 寄存器	0/10Н	GPR[rd] = HI
传输	MFLO	读 LO 寄存器	0/12Н	GPR[rd] = LO
144期	MTHI	写 HI 寄存器	0/11H	HI = GPR[rs]
	MTLO	写 LO 寄存器	0/13н	LO = GPR[rs]
	ERET	异常返回	10/18H	PC = EPC; 还需要对 CPO 的其他寄存器做处理
特权	MFC0	读 CP0 寄存器	特殊编码③	GPR[rt] = CP0[rd]
	MTC0	写 CP0 寄存器	特殊编码④	CP0[rd] = GPR[rt]
陷阱	BREAK	断点异常	0/DH	EPC = PC+4; PC = 异常处理地址; CPO 的其他寄存器做处理
1 11/1	SYSCALL	系统调用异常	0/CH	EPC = PC+4; PC = 异常处理地址; CPO 的其他寄存器做处理

①BLTZ: INSTR_{31..26}/INSTR_{20..16}=01/00H ②BGEZ: INSTR_{31..26}/INSTR_{20..16}=01/01H ③MFCO: INSTR_{31..26}/INSTR_{25..21}=10/00H ④MTCO: INSTR_{31..26}/INSTR_{25..21}=10/04H

A.2 MIPS-C 指令图



A.3 指令详解(按字母排列)

55 条 MIPS-C 指令按字母排序。每条指令均包括指令编码(encoding)、格式(format)、描述(description)、操作(operation)、示例(example)和其他(note)。其中最为重要的描述和操作部分。描述部分用 RTL(Register Transfer Language)方式定义了指令的基本操作语义,操作部分则用 RTL 定义了指令的详细操作语义。

1. ADD: 符号加

	31	26	25	21	20	16	15	11	10	6	5	0
编码	spec		rs	rs		rt	r	rd))00	add 100000	
	6 5 5 5 6											
格式	add r	d, rs	, rt									
描述	GPR[r	d] ←	GPR[rs]+GP	R[rt]							
操作	<pre>temp ← (GPR[rs]₃₁ GPR[rs]) + (GPR[rt]₃₁ GPR[rt]) if temp₃₂ ≠ temp₃₁ then SignalException(IntegerOverflow) else GPR[rd] ← temp₃₁₀ endif</pre>											
示例	add \$s1, \$s2, \$s3											
其他	temp ₃₂ ≠ temp ₃₁ 代表计算结果溢出。 如果不考虑溢出,则 add 与 addu 等价。											

2. ADDI: 符号加立即数

	31	26	25	21	20	16	15		0				
编码	add 0010		rs			rt		immediate					
	6		5			5		16					
格式	addi rt, rs, immediate												
描述	GPR[rt] ← GPR[rs]+ immediate												
操作	if ten Si else	$GPR[rt] \leftarrow temp_{310}$											
示例	temp # temp 代表计算结果溢出												
其他													

3. ADDIU: 无符号加立即数

编码	31	26	25	21	20	10	15		0
編码	addit 00100			rs		rt		immediate	

	6	5	5	16
格式	addiu rt,	rs, immedia	ite	
描述	GPR[rt] ←	GPR[rs]+ i	mmediate	
操作	GPR[rt] ←	GPR[rs] +	sign_extend(i	mmediate)
示例	addiu \$s1,	, \$s2, 0x3FE	F	
其他	"无符号"是	一个误导,其为	本意是不考虑溢出	0

4. ADDU: 无符号加

	31	26	25	21	20		16	15		11	10		6	5		0
编码	spec 0000		rs			rt			rd		C	0 00000			addu 100001	
	6		5			5			5			5			6	
格式	addu 1															
描述	GPR[ro	d] ←	GPR[rs	3] +	GPR[rt]										
操作	GPR[ro															
示例	addu S															
其他																

5. AND: 与

	31	26	25	21	20		16	15		11	10	(5	5		0
编码	specia 00000		r	S		rt			rd		00	0 0000		1	and .00100	
	6		į	5		5			5			5			6	
格式	and rd	, rs	, rt													
描述	GPR[rd] ←	GPR[r	s] AN	D GP	R[rt	;]									
操作	GPR[rd] ←	GPR[r	s] AN	D GP	R[rt	:]									
示例	and \$s	1, \$	s2, \$	s3												
其他																

6. ANDI: 与立即数

	31 26	25	21	20	16	15		0			
编码	andi 001100	rs			rt	ir	nmediate				
	6	5			5		16				
格式	andi rt, rs, immediate										
描述	GPR[rt] ←	GPR[rs	s] AN	D imr	mediate						
操作	GPR[rt] ←	GPR[rs	s] AN	D zei	ro_exte	nd(immediate)					
示例	andi \$s1, \$s2, 0x55AA										
其他											

7. BEQ: 相等时转移

	31 2	6 25	21	20	16	15		0		
编码	beq 000100		rs	rt			offset			
	6		5		5		16			
格式	beq rs, rt, offset									
描述	if (GPR[rs] == GPR[rt]) then 转移									
操作	<pre>if (GPR[rs] == GPR[rt]) PC ← PC + 4 + sign_extend(offset 0²) else PC ← PC + 4</pre>									
示例	beq \$s1, \$s2, -2									
其他										

8. BGEZ: 大于等于 0 时转移

	31	26	25	21	20	16	15		0		
编码	000001		rs	bgez 00001				offset			
	6		5			16					
格式	bgez rs										
描述	if (GPR[rs] >= 0) then 转移										
操作	else	← P		sig	n_exten	d(of	fset $\ 0^2$)				
示例	bgez \$s1, -2										
其他											

9. BGTZ: 大于 0 时转移

	31 26	25	21	20	16	15		0					
编码	bgtz 000111	rs		000			offset						
	6	5	5 5				16						
格式	bgtz rs,												
描述	if (GPR[rs] > 0) then 转移												
操作	else	PC + 4	+ sig	n_exte	nd(of	fset	(0 ²)						
示例	bgtz \$s1, -2												
其他													

10. BLEZ: 小于等于 0 时转移

编码 31 26 25 21 20 16 15	0
-------------------------	---

	blez 000110	rs	0 00000	offset							
	6	5	5	16							
格式	blez rs, o	ffset									
描述	if (GPR[rs] <= 0) the	n 转移								
操作	if (GPR[rs PC ← PC else PC ← PC	C + 4 + sig	n_extend(of	ffset $\ 0^2$)							
示例	blez \$s1, -2										
其他											

11. BLTZ: 小于 0 时转移

	31 26	25 21	20 16	15 0							
编码	000001	rs	bltz 00000	offset							
	6	5	5	16							
格式	bltz rs, o										
描述	if (GPR[rs] < 0) then 转移										
操作	if (GPR[rs PC ← F else PC ← F	C + 4 + sig	gn_extend(of	$fset 0^2)$							
示例	bltz \$s1, -2										
其他											

12. BNE:不等于时转移

	31	26	25	21	20	16	15		0		
编码	bne 00010		rs			rt		offset			
	6		5			5	16				
格式	bne rs, rt, offset										
描述	if (GPR[rs] ≠ GPR[rt]) then 转移										
操作	PC else	← P(] ≠ GPI C + 4 + C + 4			end(of	fset 0 ²	²)			
示例	bne \$s1, \$s2, 8										
其他											

13. BREAK: 断点

	31 26	25 6	5		0
编码	SPECIAL 000000	code		BREAK 001101	
	6	20		6	

格式	break
描述	产生断点异常
操作	SignalException(breakpoint)
示例	break
其他	

14. DIV: 符号除

	31	26	25	21	20		16	15		6	5		0	
编码	spec 0000		r	5		rt			0 00 0000 0000			div 011010		
	6 5 5 10											6		
格式	div rs, rt													
描述	(HI, LO) ← GPR[rs] / GPR[rt] 商存放在LO寄存器,余数存放在HI寄存器													
操作	LO← GI	PR[rs]div (PR[rt	.]									
沐正	HI ← GE	PR[rs]mod (GPR[rt	.]									
示例	列 div \$s1, \$s2													
其他	如果 GPR[rt]为 0,则 HI/LO 结果不可预料。													

15. DIVU: 无符号除

	31	26	25	21	20		16	15		6	5		0
编码	spec 0000			rs		rt			0 00 0000 0000		(divu 011011	
	6			5		5			10			6	
格式	divu rs, rt												
描述	(HI, LO)← GPR[rs] / GPR[rt] 商存放在Lo寄存器,余数存放在HI寄存器												
操作	LO+ (0 GPR[rs]) div (0 GPR[rt]) HI+ (0 GPR[rs]) mod (0 GPR[rt])												
示例	divu	\$s1,	\$s2										
其他	因为 d:	ivu 为	无符号	除法,原	斤以对	其进	行 0	扩展 1	位后再进行运算				

16. ERET: 异常返回

	31	26	25	21	20	16	15	11	10	6	5	0
编码	COP0 010000				1000	800 00 0000 0		0 0000			eret 011000	
	6											
格式	eret											
描述		eret 将保存在 CP0 的 EPC 寄存器中的现场(被中断指令的下一条地址)写入 PC,从而实现从中断、异常或指令执行错误的处理程序中返回。										
操作	PC ← CP0[epc]											

示例	eret
其他	当程序被硬件中断、执行 sc 指令、指令执行异常(如除 0)时, PC 将被保存在 EPC 中。 【注意】如果是硬件中断和 SC, EPC 中保存的 PC+4; 如果是指令执行异常(如除零、 异常等),则保存 PC。

17. J: 跳转

	31 26	25 0								
编码	j 000010	instr_index								
	6	26								
格式	j target									
描述	j 指令是 PC 相关的转移指令。当把 4GB 划分为 16 个 256MB 区域, j 指令可以在当前 PC 所在的 256MB 区域内任意跳转。									
操作	PC ← PC31	28 instr_index 0 ²								
示例	j Loop_End									
其他	如果需要跳转范围超出了当前 PC 所在的 256MB 区域内时,可以使用 JR 指令。									

18. JAL: 跳转并链接

	31 26	25 0										
编码	jal 000011	instr_index										
	6	26										
格式	jal target											
描述	jal 指令是函数指令, PC 转向被调用函数,同时将当前 PC+4 保存在 GPR[31]中。当把 4GB 划分为 16 个 256MB 区域,jal 指令可以在当前 PC 所在的 256MB 区域内任意跳 转。											
操作	PC ← PC31 GPR[31] ←	$28 \ instr_index \ 0^2$ PC + 4										
示例	jal my_function_name											
其他	jal 与 jr 配套使用。jal 用于调用函数,jr 用于函数返回。当所调用的函数地址超出了当前 PC 所在的 256MB 区域内时,可以使用 jalr 指令。											

19. JALR: 跳转并链接

	31 26 25 21		20	16	15		11	10	6	5		0		
编码	special 000000		rs		0	0 00000		rd		0 00000		jalr 001001		
	6			5		5	5			5		6		
格式	jalr rd, rs													
描述	jalr 指令是函数指令,PC 转向被调用函数(函数入口地址保存在 GPR[rs]中),同时将当前 PC+4 保存在 GPR[rd]中。													
操作	PC ← GPR[rs] GPR[rd] ← PC + 4													
示例	jalr \$s1, \$31													

20. JR: 跳转至寄存器

	31 26	25	21	20		11	10	6	5		0	
编码	special 000000		rs		0 00 0000 0000	-				jr 001000		
	6		5		10			5		6		
格式	jr rs											
描述	PC ← GPR[rs]											
操作	PC ← GPR	[rs]										
示例	jr \$31											
其他	jr 与 jal/jalr 配套使用。jal/jalr 用于调用函数,jr 用于函数返回。											

21. LB: 加载字节

	31 26	25	21	20		16	15		0			
编码	lb 100000	bas	se		rt			offset				
	6	5	5					16				
格式	lb rt, offset(base)											
描述	GPR[rt] ← memory[GPR[base]+offset]											
操作	Addr ← GPR[base] + sign_ext(offset) memword ← memory[Addr] byte ← Addr ₁₀ GPR[rt] ← sign_ext(memword _{7+8*byte8*byte})											
示例	lb \$v1, 3(\$s0)											

22. LBU: 加载无符号字节

	31 26	25 21	20 16	15 0							
编码	lbu 100100	base	rt	offset							
	6	5	5	16							
格式	lbu rt, offset(base)										
描述	GPR[rt] ← memory[GPR[base]+offset]										
操作	Addr GPR[base] + sign_ext(offset) memword memory[Addr] byte Addr ₁₀ GPR[rt] zero_ext(memword _{7+8*byte8*byte})										
示例	Ibu \$v1, 3(\$s0)										

23. LH: 加载半字

(-) T-	31	26	25	21	20	16	15		0
编码	lh 1000		ba	ase		rt		offset	

	6	5	5	16							
格式	lh rt,	offset(base)									
描述	GPR[rt] ← memory[GPR[base]+offset]										
操作	Addr GPR[base] + sign_ext(offset) memword memory[Addr] byte Addr ₁ GPR[rt] sign_ext(memword _{15+16*byte16*byte})										
示例	lh \$v1,	3(\$s0)									
约束	Addr 必须	ī是 2 的倍数(即 Add	dr ₀ 必须为 0),否则	产生地址错误异常							

24. LHU:加载无符号半字

	31 20	5 25	21	20	16	15		0			
编码	lhu 100101	hase			rt		offset				
	6	6 5				5 16					
格式	lhu rt, offset(base)										
描述	GPR[rt] ← memory[GPR[base]+offset]										
操作	Addr GPR[base] + sign_ext(offset) memword memory[Addr] byte Addr ₁ GPR[rt] zero_ext(memword _{15+16*byte16*byte})										
示例	lhu \$v1, 2(\$s0)										
约束	Addr 必须是 2 的倍数(即 Addr ₀ 必须为 0), 否则产生地址错误异常										

25. LUI: 立即数加载至高位

	31	26	25	21	20		16	15	0		
编码	lui 0011		0	0 0000		rt			immediate		
	6	6 5 5 16									
格式	lui rt, immediate										
描述	GPR[rt	t] ←	imme	ediate C)16						
操作	GPR[rt	t] ←	imme	ediate C)16						
示例	lui \$s1, 0x55AA										
其他											

26. LW: 加载字

	31 26	25 21	20 16	15	0					
编码	lw 100011	base	rt	offset						
	6	5	5	16						
格式	lw rt, offset(base)									
描述	GPR[rt] ← memory[GPR[base]+offset]									
操作	Addr GPR[base] + sign_ext(offset)									

	GPR[rt] ← memory[Addr]
示例	lw \$v1, 8(\$s0)
约束	Addr 必须是 4 的倍数(即 Addr ₁₀ 必须为 00), 否则产生地址错误异常

27. MFCO: 读 CPO 寄存器

	31	26	25	21	20		16	15		11	10		0
编码	COP 0100	-		mfc0 0000		rt			rd			0 0 0000 0000	
	6			5		5			5			11	
格式	mfc0 rt, rd												
描述	GPR[rt] ← CP0[rd]												
操作	GPR[r	t] ←	CP0[[rd]									
示例	mfc0	\$s1,	\$1										
其他													

28. MFHI: 读 HI 寄存器

	31	26	25	21	20	16	15	11	10	6	5	0		
编码	special 000000				0 00 0000			rd		0 0000	mf 0100			
	6			1	10			5		5	6			
格式	mfhi rd													
描述	GPR[rd]← HI													
操作	GPR[rd]	⊢ I	ΗI											
示例	mfhi \$s1													
其他	当乘法/除	去计	算完与	毕后,需	要用 mfh	i 读取材	目应的	J结果。						

29. MFLO: 读 LO 寄存器

	31	26	25	21	20	16	15	11	10	6	5	C)
编码	spec 0000			00 0	0000 0000			rd		0 0000		mflo 10010	
	6				10			5		5		6	
格式	mflo	rd											
描述	GPR[r	:d] ←	LO										
操作	GPR[r	:d] ←	LO										
示例	mflo	\$s1											
其他	当乘法	:/除法	计算完毕	毕后,	需要用 mf	lo 读取木	目应的	结果。					

30. MTCO: 写 CPO 寄存器

	31	26	25	21	20		16	15		11	10		0
编码	COP(01000	-		ntc0)100		rt			rd			0 0 0000 0000	
	6			5		5			5			11	
格式	mtc0 rt, rd												
描述	CP0[rd] ← GPR[rt]												
操作	CP0[rd	[] ←	GPR[rt]									
示例	mtc0 \$	s1,	\$1										
其他													

31. MTHI: 写 HI 寄存器

	31	26	25	21	20		15	11	10	6	5		0	
编码	specia 000000			rs		0	0000 000	0000 00	00			mthi 010001		
	6			5			1	.5				6		
格式	mthi rs	5												
描述	HI ← GPR[rs]													
操作	HI ← G	PR[r	îs]											
示例	mthi \$5	s1												
其他	mthi/mtlo 乘除法运			中断响应; 向应结束后				nfhi/mflo	o 配套	使用确保		中断程序	的	

32. MTLO: 写 LO 寄存器

	31	26	25	21	20		15		11	10		6	5		0
编码	spec 0000			rs			000 000	0 0 000	00 00	00				mtlo 010011	
	6			5				15						6	
格式	mtlo	rs													
描述	LO ← GPR[rs]														
操作	LO ←	GPR[1	rs]												
示例	mtlo	\$s1													
其他				中断响应 响应结束原				mfh	i/mflo	配套	使用码	角保	:被中	断程序	·的

33. MULT: 符号乘

	31 26	25	21	20		16	15		6	5	0			
编码	special 000000	rs			rt			0 00 0000 0000		mult 011000				
	6	5			5			10		6				
格式	mult rs, rt													
描述	(HI, LO) ← GPR[rs] × GPR[rt]													

	乘积低32位存放在LO寄存器,高32位存放在HI寄存器。所有操作数均为有符号数。
	prod ← GPR[rs] × GPR[rt]
操作	HI ← prod ₆₃₃₂
	LO ← prod ₃₁₀
示例	mult \$s1, \$s2
其他	

34. MULTU: 无符号乘

	31 2	6	25	21	20		16	15	6	5		0			
编码	special 000000		rs			rt		0 00 0000 0000			nultu 11001				
	6		5			5		10			6				
格式	multu rs	, r	t												
描述	(HI, LO) ← GPR[rs] × GPR[rt] 乘积低32位存放在LO寄存器,高32位存放在HI寄存器。所有操作数均为无符号数。														
操作	乘积低32位存放在LO寄存器,高32位存放在HI寄存器。所有操作数均为无符号数。 prod ← (0 GPR[rs])×(0 GPR[rt]) HI ← prod ₆₃₃₂ LO ← prod ₃₁₀														
示例	multu \$s	1,	\$s2												
其他	因为 mult	1 为	无符号乘	法,	所以邓	付其进	行(扩展 1 位后再进行运算	0						

35. NOR: 或非

	31	26	25	21	20		16	15		11	10	6	5	5		0
编码	spec 0000		-	rs		rt			rd		0	0 0000		1	nor 100111	
	6			5		5			5			5			6	
格式	nor rd, rs, rt															
描述	GPR[rd] ← GPR[rs] NOR GPR[rt]															
操作	GPR[r	d] ←	GPR[r	s] NO	R GP	R[rt]									
示例	nor \$	s1, \$	s2, \$	s3												
其他												•			•	

36. OR: 或

	31 2	6 2	25 21		20		16	15		11	10		6	5		0
编码	special 000000		rs			rt			rd		0	0 0000		:	or 100101	
	6		5			5			5			5			6	
格式	or rd, r	s, r	rt .													
描述	GPR[rd]	← G1	PR[rs]	OR	GPR	[rt]										
操作	GPR[rd] ← GPR[rs] OR GPR[rt]															

示例	or \$s1, \$s2, \$s3
其他	

37. ORI: 或立即数

	31	26	25	21	20		16	15		0
编码	ori 00110:	1	rs			rt			immediate	
	6		5			5			16	
格式	ori rt,	rs	, imme	diate	<u> </u>					
描述	GPR[rt]	+	GPR[rs] OR	imme	edia	te			
操作	GPR[rt]	+	GPR[rs] OR	zer	o_ex	tend	d(immediate)		
示例	ori \$s1	L , \$	s2, 0x	55AA						
其他										

38. SB: 存储字节

	31 26	25	21	20	1	.6	5 15	0				
编码	sb 101000	ba	se		rt		offset					
	6	!	5		5		16					
格式	sb rt, offset(base)											
描述	memory[GPR[base]+offset]											
操作	Addr ← GPR[base] + sign_extend(offset) byte← Addr ₁₀ memory[Addr] _{7+8*byte8*byte} ← GPR[rt] _{7:0}											
示例	sb \$v1, 3	(\$s0)										

39. SH: 存储半字节

	31 2	5 25	21	20	16	j	15 0					
编码	sh 101001	b	ise		rt		offset					
	6		5		5		16					
格式	sh rt, offset(base)											
描述	<pre>memory[GPR[base]+offset]</pre>											
操作	Addr ← G byte ← A memory[A	ddr_1										
示例	sh \$v1, 24(\$s0)											
约束	Addr 必须是	2 的倍数	即 Add	r ₀ 必须	页为 0),	否具	5则产生地址错误异常					

40. SLL: 逻辑左移

	special 000000	0	rt	rd	S	sll 000000						
	6	00000	5	5	5	6						
格式	sll rd, rt, s											
描述	GPR[rd] ← GPR[rt] << s											
操作	GPR[rd] ←	GPR[rd] ← GPR[rt] _{(31-s)0} 0 ^s										
示例	sll \$s1, \$s2, 5											
其他	·) , 0 对应的指 用于空循环,不		_								

41. SLLV:逻辑可变左移

	31	26	25	21	20		16	15		11	10	6	5		0
编码	spec 0000		rs			rt			rd		00	0 0000		sllv 000100	
	6 5					5		5				5			
格式	sllv rd, rt, rs														
描述	GPR[rd] ← GPR[rt] << GPR[rs]														
操作	s ← (GPR[rs	3]40												
沐正	GPR[r	d] ←	GPR[r	t] _{(31-s})0 0	ງຶ									
示例	sllv \$s1, \$s2, \$s3														
其他	GPR[rs]的位 31 至位 5 被忽略。														

42. SLT: 小于置 1(有符号)

	31	26	25	21	20		16	15		11	10		6	5		0
编码	special 000000		rs			rt			rd		(00000			slt 101010	
	6		00000			5			5			5			6	
格式	slt rd,	rs	, rt													
描述	GPR[rd]	+	(GPR[rs] <	GPR	[rt])									
操作	GPR[rd]	+	(GPR[rs] <	GPR	[rt]) ?	031 1	1:	032						
示例	slt \$s1	, \$	s2, \$s3													
其他																

43. SLTI: 小于立即数置 1(有符号)

	31 26	25 21	20 16	15		0						
编码	slti 001010	rs	rt	imn	nediate							
	6	5	5		16							
格式	slti rt, rs, immediate											
描述	GPR[rt] ← (GPR[rs] < immediate)											
操作	GPR[rt] ←	(GPR[rs] <	sign_exten	d(immediate)) ?	0 ³¹ 1 : 0 ³²							

示例	slti \$s1, \$s2, 0x55AA
其他	

44. SLTIU: 小于立即数置 1(无符号)

	31 26	25	21	20	16	15	0					
编码	sltiu 001011	rs		rt		immediate						
	6	5		5		16						
格式	sltiu rt, rs, immediate											
描述	GPR[rt] ←	(GPR[rs]	< :	immedi	ate)							
操作	GPR[rt] ←	(0 GPR[r	s]<	0 sign	_ext	tend(immediate)) ? $0^{31} \ 1 : 0^{32}$						
示例	sltiu \$s1, \$s2, 0xAABB											
其他	"无符号"是	误导										

45. SLTU: 小于置 1(无符号)

	31	26	25	21	20		16	15		11	10	6	5		0
编码	speci 00000		rs			rt		rd		00000			sltu 101011		
	6		0000	0		5			5			5		6	
格式	sltu rd, rs, rt														
描述	GPR[rd] ← (GPR[rs] < GPR[rt])														
操作	$GPR[rd] \leftarrow (0 GPR[rs] < 0 GPR[rt]) ? 0^{31} 1 : 0^{32}$														
示例	sltu \$s1, \$s2, \$s3														
其他															

46. SRA: 算术右移

	31	26	25	21	20		16	15		11	10		6	5		0
编码	speci 00000		0			rt			rd			S			sra 000011	
	6		000	00		5			5			5			6	
格式	sra ro	sra rd, rt, s														
描述	GPR[rd] ← GPR[rt] >> s															
操作	GPR[rd] ← GPR[rt] ₃₁ ° GPR[rt] _{31s}															
示例	sra \$s1, \$s2, 5															
其他																

47. SRAV: 算术可变右移

编码	31	26	25	21	20	16	15	11	10	6	5		0
細竹	spe	ecial		rs		rt		rd	00	0000		srav	

	000000					000111						
	6	5	5	5	5	6						
格式	srav rd, r	t, rs										
描述	GPR[rd] ← GPR[rt] >> GPR[rs]											
操作	s ← GPR[rs] ₄₀											
沐正	GPR[rd] ←	$\texttt{GPR[rt]}_{31}^{\text{s}} \big\ $	GPR[rt] _{31s}									
示例	srav \$s1, \$s2, \$s3											
其他	GPR[rs]的位 31 至位 5 被忽略。											

48. SRL: 逻辑右移

	31	26	25	21	20		16	15		11	10		6	5		0
编码	specia 00000			0		rt			rd			S			srl 000010	
	6		0	0000		5			5			5			6	
格式	srl rd,	srl rd, rt, s														
描述	GPR[rd] ← GPR[rt] >> s															
操作	GPR[rd]	_ ←	0° 0	GPR[rt]	31s											
示例	srl \$s1, \$s2, 5															
其他																

49. SRLV:逻辑可变右移

	31 26	25 2	1 20		16	15		11	10	6	5		0
编码	special 000000	rs		rt			rd		000	00		srlv 000110	
	6	5		5			5		5	5		6	
格式	srlv rd, r	t, rs											
描述	GPR[rd] ← GPR[rt] >> GPR[rs]												
操作	s ← GPR[rs	5]40											
沐正	GPR[rd] ←	0° GPR[rt] _{31s}										
示例	srlv \$s1, \$s2, \$s3												
其他	GPR[rs]的位 31 至位 5 被忽略。												

50. SUB: 符号减

	31	26	25	21	20		16	15		11	10	6	5		0
编码	special 000000		rs			rt			rd		000))00	1	sub 00010	
	6		5			5			5		5	5		6	
格式	sub rd,	rs	, rt												
描述	GPR[rd] ← GPR[rs] - GPR[rt]														

操作	<pre>temp ← (GPR[rs]₃₁ GPR[rs]) - (GPR[rt]₃₁ GPR[rt]) if temp₃₂ ≠ temp₃₁ then SignalException(IntegerOverflow) else GPR[rd] ← temp₃₁₀ endif</pre>
示例	sub \$s1, \$s2, \$s3
其他	temp ₃₂ ≠ temp ₃₁ 代表计算结果溢出。 如果不考虑溢出,则 sub 与 subu 等价。

51. SUBU: 无符号减

	31	26	25	21	20		16	15		11	10	6	6	5		0
编码	speci 00000		r	5		rt			rd		00	0 0000		_	ubu 0011	
	6		5			5			5			5			6	
格式	subu r	ubu rd, rs, rt														
描述	GPR[ro	GPR[rd] ← GPR[rs] - GPR[rt]														
操作	GPR[ro	1] ←	GPR[r	s] -	GPR[rt]										
示例	subu \$s1, \$s2, \$s3															
其他	subu 不考虑减法溢出。例如 0x0000_0000 - 0xFFFF_FFF = 0x0000_0001, 即结果为非负值。															

52. SW: 存储字

	31	26	25	21	20	-	16	15		0		
编码	sw 101011		base			rt			offset			
	6		5			5			16			
格式	sw rt, offset(base)											
描述	memory[GPR[base]+offset]											
操作	Addr ← GPR[base] + sign_ext(offset) memory[Addr] ← GPR[rt]											
示例	sw \$v1, 8(\$s0)											
约束	Addr 必须是 4 的倍数(即 Addr ₁₀ 必须为 00), 否则产生地址错误异常											

53. SYSCALL: 系统调用

	31 26	25 6	5		0						
编码	SPECIAL 000000	code		SYSCALL 001100							
	6	20		6							
格式	syscall										
描述	产生系统调用异常										
操作	SignalException(systemcall)										
示例	syscall										

其他

54. XOR: 异或

	31	26	25	21	20		16	15		11	10	6	5		0
编码	speci 00000			rs		rt			rd		00	0 0000		xor 100110	
	6			5		5			5			5		6	
格式	xor ro	xor rd, rs, rt													
描述	GPR[rd] ← GPR[rs] XOR GPR[rt]														
操作	GPR[rd	l] ←	GPR[rs] XO	R GP	R[rt	:]								
示例	xor \$s1, \$s2, \$s3														
其他															

55. XORI: 异或立即数

	31	26	25	21	20		16	15	0		
编码	xori 00111	0		rs		rt		imme	diate		
	6			5		5		1	6		
格式	xori r	t, r	s, in	mediat	.e						
描述	GPR[rt] ← GPR[rs] XOR immediate										
操作	GPR[rt] ← GPR[rs] XOR zero_extend(immediate)										
示例	xori \$s1, \$s2, 0x55AA										
其他		•	•			•					