附录A MIPS-C 指令集

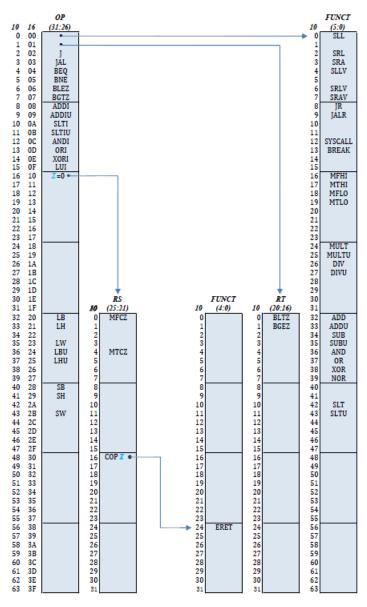
A.1 MIPS-C 指令表

本书从 MIPS 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 11 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	20H/24H	<pre>R[rt] = {24{Mem[GPR[rs]+sign_ext(offset)][7]},</pre>
	LBU	加载字节 (无符号)	24H	<pre>R[rt] = {24'b0, Mem[GPR[rs]+</pre>
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)	25Н	<pre>R[rt] = {16'b0, Mem[GPR[rs]+</pre>
	LW	加载字	23Н	R[rt] = Mem[GPR[rs]+sign_ext(offset)]
	SB	存储字节	28Н	<pre>Mem[GPR[rs]+sign_ext(offset)][7:0] = R[rt][7:0]</pre>
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	0/32Н	GPR[rd] = GPR[rs] + GRP[rt]
	ADDU	无符号加	0/33Н	GPR[rd] = GPR[rs] + GRP[rt]
	SUB	减	0/34Н	GPR[rd] = GPR[rs] - GRP[rt]
	SUBU	无符号减	0/35Н	GPR[rd] = GPR[rs] - GRP[rt]
	MULT	乘	0/24H	{HI, LO} = GPR[rs] × GRP[rt]
	MULTU	乘(无符号)	0/25Н	{HI, LO} = GPR[rs] × GRP[rt]
	DIV	除	0/26Н	{HI, LO} = GPR[rs] / GRP[rt]
	DIVU	除(无符号)	0/27Н	{HI, LO} = GPR[rs] / GRP[rt]
R-R	SLL	逻辑左移	0/0H	GPR[rd] = {GPR[rt][31-s:0], s{0}}
运算	SRL	逻辑右移	0/2H	GPR[rd] = {s{0}, GPR[rt][31:s]}
	SRA	算术右移	0/3H	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	0/6Н	GPR[rd] = {v{0}, GPR[rt][31:v]}
	SRAV	算术可变右移	0/7н	GPR[rd] = {v{GPR[rt][31]}, GPR[rt][31:v]}
	AND	与	0/36Н	GPR[rd] = GPR[rs] & GRP[rt]
	OR	或	0/37Н	GPR[rd] = GPR[rs] GRP[rt]
	XOR	异或	0/38Н	GPR[rd] = GPR[rs] ^ GRP[rt]
	NOR	或非	0/39Н	GPR[rd] = ~(GPR[rs] GRP[rt])
R-I	ADDI	加立即数	8Н	GPR[rd] = GPR[rs] + SignExtImm
运算	ADDIU	加立即数 (无符号)	9Н	GPR[rd] = GPR[rs] + SignExtImm

	ANDI	与立即数	СН	GPR[rd] = GPR[rs] & ZeroExtImm
	ORI	或立即数	DH	GPR[rd] = GPR[rs] ZeroExtImm
	XORI	异或立即数	EH	GPR[rd] = GPR[rs] ^ ZeroExtImm
	LUI	立即数加载至高 位	FH	GPR[rd] = {imm, 16'b0}
	SLTI	小于立即数置1	АН	GPR[rt] = (GPR[rs] < SignExtImm) ? 1 : 0
	SLTIU	小于立即数置 1 (无符号号)	ВН	GPR[rt] = (GPR[rs] < SignExtImm) ? 1 : 0
	BEQ	等于转移	4 H	<pre>if (GRP[rs] == GPR[rt]) PC = PC + 4 + BranchAddr</pre>
	BNE	不等转移	5Н	<pre>if (GRP[rs] != GPR[rt]) PC = PC + 4 + BranchAddr</pre>
ハナ	BLEZ	小于等于0转移	6Н	<pre>if (GRP[rs] <= 0) PC = PC + 4 + BranchAddr</pre>
分支	BGTZ	大于0转移	7н	<pre>if (GRP[rs] > 0) PC = PC + 4 + BranchAddr</pre>
	BLTZ	小于0转移	特殊编码	<pre>if (GRP[rs] <0) PC = PC + 4 + BranchAddr</pre>
	BGEZ	大于等于0转移	特殊编码	<pre>if (GRP[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	0/8Н	PC = GPR[rs]; GPR[rd] = PC + 4
	JR	跳转寄存器	0/9Н	PC = GPR[rs]
	MFHI	读 HI 寄存器	0/16Н	GPR[rd] = HI
传输	MFLO	读 LO 寄存器	0/17н	GPR[rd] = LO
144的	MTHI	写 HI 寄存器	0/18Н	HI = GPR[rd]
	MTLO	写 LO 寄存器	0/19Н	LO = GPR[rd]
	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 CPO 寄存器	特殊编码	GPR[rt] = CP0[rd]
	MTC0	写 CP0 寄存器	特殊编码	CP0[rd] = GPR[rt]
陷阱	BREAK	断点异常	0/13н	EPC = PC+4; PC = 异常处理地址; CPO 的其他寄存器做处理
гирт	SYSCALL	系统调用异常	0/12н	EPC = PC+4; PC = 异常处理地址; CPO 的其他寄存器做处理

A.2 MIPS-C 指令图



A.3 加载指令

1. lb: 加载字节

	31 26	25 21	20 1	6 15	0						
编码	lb 100000	base	rt		offset						
	6	16									
格式	lb rt, off	lb rt, offset(base)									
描述	GPR[rt] ←	memory[GPR	[base]+of:	set]							
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)										

2. Ibu: 加载无符号字节

	31 2	5 25	21	20	16	5	15			0	
编码	lbu 100100		base		rt			offs	et		
	6		5		5			16	5		
格式	lb rt, offset(base)										
描述	GPR[rt]	← memo	ory[GPR	[base	e]+off	se	t]				
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	lb \$v1, 3(\$s0)										

3. lh: 加载半字

	31	26 2	25 21	20	16	15		0				
编码	lh 100001		base	rt			offset					
	6		5	5			16					
格式	lh rt, c	lh rt, offset(base)										
描述	GPR[rt]	← m	nemory[GPR	[base]+	offs	et]						
操作	memword byte ←	← m Addr	[base] + s: nemory[Add: f ₁ sign_ext(me	r]			_{byte})					
示例	lb \$v1, 3(\$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											

4. lhu: 加载无符号半字

	31 2	26	25 21	20		16	15	0				
编码	lhu 100101		base		rt			offset				
	6		5			16						
格式	lhu rt,	lhu rt, offset(base)										
描述	GPR[rt]	← n	memory[GPI	[bas	e]+c	offse	et]					
操作	memword byte ← 1	← r Addi	[base] + s memory[Ado r ₁ zero ext(r	lr]				;*byte)				
示例	lb \$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											

5. lw: 加载字

	31 2	26 2	25 21	20		16	15	0			
编码	lh 100011	hase			rt			offset			
	6		5		5			16			
格式	lw rt, c	offse	et (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

A.4 保存指令

6. sb: 存储字节

	31	26	25	21	20		16	15	5 0		
编码	sb 101000	ı	base	base					offset		
	6 5 5 16										
格式	sb rt, offset(base)										
描述	GPR[rt]	+	memory	[GPR	[base	e]+c	offs	et]			
操作	Addr GPR[base] + sign_extend(offset) byte Addr10 memory[Addr] _{7+8*byte} GPR[rt] _{7:0}										
示例	sb \$v1, 3(\$s0)										

7. sh: 存储半字节

	31	26	25	21	20	16	15	0		
编码	sh 101001		base		rt		offset			
	6		5			16				
格式	sh rt, offset(base)									
描述	GPR[rt]	+	memory[GPR	[base]+offs	et]			
操作	Addr ← byte ← memory[Add	lr1							
示例	sh \$v1, 24(\$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									

8. sw: 存储字

	31	26	25	21	20		16	15	0					
编码	sw 10101	1	bas	base rt				offset						
	6 5 5 16													
格式	sh rt, offset(base)													
描述	GPR[rt] ←	memory	[GPR	[bas	e]+c	offs	et]						
操作	Addr ← memory					ext	(off:	set)						
示例	sw \$v1, 8(\$s0)													
约束	Addr 必須	页是 4	的倍数(即 Add	r ₁₀ 必	须为	00),	否则	川产生地址错误异常					

A.5 R-R 运算指令

9. add: 符号加

	31	26	25	21	20	16	15		11	10	6	5		0
编码	spec 0000		rs	rs		rt		rd		0 00000		add 100000		
	6		5			5	5				5		6	
格式	add rd, rs, rt													
描述	GPR[rd] ← GPR[rs]+GPR[rt]													
操作	if te Si else	mp ₃₂ / IgnalE PR[rd]	PR[rs] ₃ temp ₃₁ Excepti ← tem	the	n		-] 31 0	GPR[rt])				
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。													

10. addu: 无符号加

	31	26	25	21	20		16	15		11	10	6	5	5		0
编码	speci 00000			rs		rt			rd		00	0 0000			addu .00001	
	6			5		5			5			5			6	
格式	addu r	addu rd, rs, rt														
描述	GPR[rd	GPR[rd] ← GPR[rs] + GPR[rt]														
操作	GPR[rd	GPR[rd] ← GPR[rs] + GPR[rt]														
示例	addu \$s1, \$s2, \$s3															
其他																

11. and: 与

	31	26	25	21	20		16	15		11	10	6	ô	5		0
编码	specia 00000		rs			rt			rd		0	0 0000			addu 100100	
	6		5			5			5			5			6	
格式	and rd	d rd, rs, rt														
描述	GPR[rd	rd] ← GPR[rs] AND GPR[rt]														
操作	GPR[rd	.] ←	GPR[rs] AN	D GP	R[rt	:]									
示例	and \$s	1, \$	s2, \$s3	3												
其他																

12. div: 符号除

	31 26	25 21	20	16	15		6	5	0			
编码	special 000000	rs	rt			0 00 0000 0000		div 011010				
	6	5	5			10		6				
格式	div rs, rt	;										
描述	(HI, LO) •	► GPR[rs] ,	GPR[rs] / GPR[rt]									
畑处	商存放在LO含	序存器,余数 有	器,余数存放在HI寄存器									

操作	LO GPR[rs] div GPR[rt]
採作	HI ← GPR[rs] mod GPR[rt]
示例	div \$s1, \$s2
其他	如果 GPR[rt]为 0,则 HI/LO 结果不可预料。

13. divu: 无符号除

	31	26	25	21	20	1	6	15		6	5		0
编码	specia 00000		rs			rt			0 00 0000 0000			div 011011	
	6		5			5			10			6	
格式	divu r	s, r	t										
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	同仔放住LO奇仔裔, 宗数仔放住HI奇仔裔 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 divu 为无符号除法,所以对其进行 0 扩展 1 位后再进行运算。												

14. mult: 符号乘

	31	26	25	21	20	16	15		6	5		0	
编码	spec 0000		rs			rt		0 00 0000 0000			mult 011000		
	6		5			5		10			6		
格式	mult :	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。所有操作数均为有符号数。												
操作	乘积低32位存放在LO寄存器,高32位存放在HI寄存器。所有操作数均为有符号数。 prod ← GPR[rs] × GPR[rt] HI ← prod ₆₃₃₂ LO ← prod ₃₁₀₀												
示例	mult :	\$s1,	\$s2										
其他													

15. multu: 无符号乘

	31 26	25	21	20	16	15	(6	5	0		
编码	special 000000	rs			rt	00 0	0 0000 0000		multu 011001			
	6	5 5 10 6										
格式	multu 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。所有操作数均为无符号数。											
操作	飛標版32世存成在LU可存益,同32世存成在HI可存益。所有採作数均为允符与数。 prod ← (0 GPR[rs]) × (0 GPR[rt]) HI ← prod ₆₃₃₂ LO ← prod ₃₁₀₀											
示例	multu \$s1,	, \$s2										

其他 因为 multu 为无符号乘法,所以对其进行 0 扩展 1 位后再进行运算。

16. nor: 或非

	31	26	25	21	20		16	15		11	10		6	5		0
编码	specia 00000				rt			rd		0	0 0000			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[rd] ←	GPR[1	s] NO	R GP	R[rt	:]									
示例	nor \$s	1, \$	s2, \$	s3												
其他																

17. or: 或

	31	26	25	21	20		16	15		11	10	6	5		0
编码	specia 00000		r	S		rt			rd		0(0 0000		or 100101	
	6		į	5		5			5			5		6	
格式	or rd,	or rd, rs, rt													
描述	GPR[rd	[rd] ← GPR[rs] OR GPR[rt]													
操作	GPR[rd	[] ←	GPR[r	s] OR	GPR	[rt]									
示例	or \$s1	, \$s	2, \$s	3											
其他															

18. sll: 逻辑左移

	31 26	25	21	20		16	15		11	10		6	5		0
编码	special 000000	0			rt			rd			S			sll 000000	
	6	0000	0		5			5			5			6	
格式	sll rd, r	t, s													
描述	GPR[rd] +	PR[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	sl1 \$0, \$0, 0 对应的指令码是 0x0000_0000, 也被认为是 NOP(空操作指令)。 该指令有时被用于空循环,有时被编译器用于与体系结构相关的编译优化。														

19. sllv: 逻辑可变左移

	31 2	5 25	21	20		16	15		11	10		6	5	0
编码	special 000000		rs		rt			rd		0	0 0000		sllv 00010	00
	6		5 5 5 5								6			
格式	sllv rd,	rd, rt, rs												
描述	GPR[rd] ← GPR[rt] << GPR[rs]													
操作	s ← GPR[rs] ₄₀													

	GPR[rd] ← GPR[rt] _{(31-s)0} 0 ^s
示例	sllv \$s1, \$s2, \$s3
其他	GPR[rs]的位 31 至位 5 被忽略。

20. slt: 小于置 1(有符号)

	31	26	25	21	20		16	15		11	10		6	5		0
编码	special 000000		0			rt			rd			S			slt 101010	
	6		0000)		5			5			5			6	
格式	slt rd,	rs	, rt													
描述	GPR[rd]	+	(GPR[r	s] <	GPR	[rt])									
操作	GPR[rd]	+	(GPR[r	s] <	GPR	[rt]) ?	031	1	: 0	32					
示例	slt \$s1	, \$	s2, \$s3													
其他																

21. sltu: 小于置 1(无符号)

	31	26	0 00000 rs, rt		20		16	15		11	10		6	5		0
编码	specia 00000		0			rt			rd			S			sltu 101011	
	6		000	00		5			5			5			6	
格式	sltu r	d, r	s, rt													
描述	sltu rd, rs, rt GPR[rd] ← (GPR[rs] < GPR[rt])															
操作	GPR[rd] ←	(0 GPF	[rs]	< 0	GPR	[rt]) ?	031	1 :	032					
示例	sltu \$	s1,	\$s2, \$	s3												
其他																

22. sra: 算术右移

	31	26	25	21	20		16	15		11	10		6	5		0
编码	specia 00000			0		rt			rd			S		(sra 000011	
	6		0	0000		5			5			5			6	
格式	sra rd	· · ·														
描述	GPR[rd]	sra rd, rt, s GPR[rd] ← GPR[rt] >> s														
操作	GPR[rd]] ←	GPR[rt] ₃₁ ^s	GPR	[rt]	31s									
示例	sra \$si	1, \$	s2,	5												
其他																

23. srav: 算术可变右移

	31	26	25	2:	20		16	15		11	10	6	5		0
编码	specia 00000			rs		rt			rd		000	000		srav 000111	
	6			5		5			5		5	;		6	
格式	srav r	d, r	t, rs	3											

描述	GPR[rd] ← GPR[rt] >> GPR[rs]
操作	s ← GPR[rs] ₄₀
沐田	GPR[rd] ← GPR[rt] ₃₁ ^s GPR[rt] _{31s}
示例	srav \$s1, \$s2, \$s3
其他	GPR[rs]的位 31 至位 5 被忽略。

24. 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		000	00		5			5			5			6	
格式	srl rd	rl rd, rt, s														
描述	GPR[rd	rl rd, rt, s PR[rd] GPR[rt] >> s														
操作	GPR[rd	<u>-</u>	0° GPI	R[rt]	31s											
示例	srl \$s	1, \$	s2, 5													
其他																

25. srlv: 逻辑可变右移

	31	26	25	21	20		16	15		11	10	6	5		0
编码	spec 0000		r	5		rt			rd		0	0000	(srlv 000110	
	6		5			5			5			5		6	
格式															
描述	GPR[rd] ← GPR[rt] >> GPR[rs]														
操作	s ← G	PR[r	5]40												
1米1下	GPR[ro	d] ←	0° GP	R[rt]	31s										
示例	srlv :	\$s1,	\$s2, \$	ss3											
其他	GPR[r	s]的位	31 至位	立 5 被	忽略。										

26. sub: 符号减

	31	26	25	21	20		16	15		11	10	6	;	5	0
编码	spec 0000		rs			rt			rd			0 0000		sub 1000:	
	6		5			5			5			5		6	
格式	sub r	d, rs	, rt												
描述	<pre>GPR[rd] ← GPR[rs] - GPR[rt] temp ← (GPR[rs]₃₁ GPR[rs]) - (GPR[rt]₃₁ GPR[rt])</pre>														
操作	if te Si else	mp ₃₂ / .gnalE	PR[rs] ₃ 4 temp ₃₃ Excepti ← tem	the	n			-] 31 0	GPR[rt])				
示例	sub \$	s1, \$	s2, \$s	3											
其他			emp ₃₁ 代ā 出,则 s				۲.								

27. subu: 无符号减

	31 26	25	21	20		16	15		11	10		6	5		0
编码	special 000000	rs			rt			rd		0	0 0000			subu 100011	
	6	5			5			5			5			6	
格式	sub rd, r	s, rt													
描述	GPR[rd] ←	Br[rd] ← GPR[rs] - GPR[rt]													
操作	GPR[rd] ←	GPR[rs]	- (GPR[1	rt]										
示例	sub \$s1,	\$s2, \$s3													
其他	subu 不考虑即结果为非负		。例如	如 0x	0000	00	00 -	0x	FFFF	_FFF	F = ()x0	000	_0001	,

28. xor: 异或

	31	26	25		21	20		16	15		11	10		6	5		0
编码	specia 00000			rs			rt			rd		0	0 0000			xor 100110	
	6			5			5			5			5			6	
格式	xor rd	, rs	, rt														
描述	GPR[rd]	—	GPR[rs]	XOI	R GP	R[rt	:]									
操作	GPR[rd]	_ ←	GPR[rs]	XOI	R GP	R[rt	:]									
示例	xor \$si	1, \$	s2, S	\$s3													
其他																	

A.6 R-I 运算指令

29. addi: 符号加立即数

	31 20	5 25	21	20	16	5 15	0							
编码	addi 001000		rs	r	t	immediate								
	6		5	5	5	16								
格式	addi rt,	rs, i	mmediat	.e										
描述	<pre>GPR[rt] ← GPR[rs] + immediate temp ← (GPR[rs]₃₁ GPR[rs]) + sign_extend(immediate)</pre>													
操作	if temp ₃₂ Signa else	≠ tem	p ₃₁ the tion(I	n	-									
示例	addi \$s1	\$s2,	-1											
其他	temp ₃₂ ≠ 如果不考虑					`•								

30. addiu: 无符号加立即数

	31 2	6	25	21	20		16	15	0
编码	addi 001001		rs			rt		immediate	
	6		5			5		16	

格式	addiu rt, rs, immediate
描述	GPR[rt] ← GPR[rs] + immediate
操作	<pre>GPR[rt] ← GPR[rs] + sign_extend(immediate)</pre>
示例	addiu \$s1, \$s2, 0xFFFF
其他	"无符号"是一个误导,其本意是不考虑溢出。

31. andi: 与立即数

	31	26	25	21	20		16	15		0		
编码	andi 001100		rs			rt			immediate			
	6		!	5	5			16				
格式	andi rt, rs, immediate											
描述	GPR[rt	:] ←	GPR[r	s] AN) im	media	ate					
操作	GPR[rt	:] ←	GPR[r	s] AN) ze:	ro_e	xte	nd(im	mediate)			
示例	andi \$s1, \$s2, 0x55AA											
其他												

32. lui: 立即数加载至高位

	31	26	25	21	20		16	15	0	
编码	lui 00111					rt			immediate	
	6 5 5 16									
格式	lui rt, immediate									
描述	lui rt	, im	media	ate $\parallel 0^1$	6					
操作	lui rt	, im	media	ate $\parallel 0^1$	6					
示例	lui \$s1, 0x55AA									
其他										

33. ori: 或立即数

	31	26	25	21	20		16	15		0	
编码	andi 001101		rs			rt			immediate		
	6		5			5			16		
格式	ori rt, rs, immediate										
描述	GPR[rt] ← GPR[rs] OR immediate										
操作	GPR[rt] ←	GPR[rs] OR	zer	o_ext	tend	d(immediate)			
示例	ori \$s1, \$s2, 0x55AA										
其他											

34. 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	immediate
	6	5	5	16

格式	slti rt, rs, immediate
描述	GPR[rt] ← (GPR[rs] < immediate)
操作	GPR[rt] \leftarrow (GPR[rs] < sign_extend(immediate)) ? 0^{31} 1 : 0^{32}
示例	slti \$s1, \$s2, 0x55AA
其他	

35. 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	sltiu rt, rs, immediate										
描述	GPR[rt] ←	(GPR[rs] <	imm	edia	te)					
操作	GPR[rt] ←	(0 GPR	[rs]	< 0	sign	_ex	tend	$d(immediate)) ? 0^{31} 1 : 0^{32}$			
示例	sltiu \$s1, \$s2, 0xAABB											
其他	"无符号	"无符号"是误导										

36. xori: 异或立即数

	31 2	26 25	5 21	20	16	15	0				
编码	xori 001110		rs		rt		immediate				
	6 5 5 16										
格式	xori rt, rs, immediate										
描述	GPR[rt] ← GPR[rs] XOR immediate										
操作	GPR[rt]	← GP	R[rs] XO	R ze:	ro_ext	end(immediate)				
示例	xori \$s1, \$s2, 0x55AA										
其他											

A.7 分支指令

37. beq: 相等时转移

	31 26 25 21 20 16 15										
编码	beq 000100	r	5		rt		offset				
	6	5			5	16					
格式	beq rs, rt, offset										
描述	if (GPR[rs] == GPR[rt]) then 转移										
操作	else	PC + 4	-	- /	(offset	: 0²)					
示例	beq \$s1, \$s2, -2										
其他											

38. bgez: 大于等于 0 时转移

	31	26	25	21	20	0					
编码	0000	01	rs		bge 000			offset			
	6		5		5			16			
格式	bgez rs, offset										
描述	if (GPR[rs] >= 0) then 转移										
操作	PC else	← P] >= 0 C + si	•	tend(o	ffset	0 ²)				
示例	bgez \$s1, -2										
其他											

39. bgtz: 大于 0 时转移

	31	26	25	21	20	16	0				
编码	bgt: 0001		rs	rs 0 00000				offset			
	6		5		Ţ	5		16			
格式	bgtz rs, offset										
描述	if (GPR[rs] > 0) then 转移										
操作	PC else	← P] > 0) C + si C + 4	gn_ex	tend(offset	0 ²)				
示例	bgtz \$s1, -2										
其他											

40. blez: 小于等于 0 时转移

	31	26	25	21	20	16	15	0					
编码	blez 00011		rs	rs 0 00000				offset					
	6 5 5							16					
格式	blez rs, offset												
描述	if (GPR[rs] <= 0) then 转移												
操作	PC else	← P(] <= 0) C + sig C + 4	n_ex	tend (c	ffset	0 ²)						
示例	bgtz \$s1, -2												
其他													

41. 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, c	ffset		

描述	if (GPR[rs] < 0) then 转移
操作	<pre>if (GPR[rs] < 0) PC ← PC + sign_extend(offset 0²) else PC ← PC + 4</pre>
示例	bltz \$s1, -2
其他	

42. bne: 不等于时转移

	31 20	6 25	21	20	16	15	0	
编码	bne 000101		rs	r	t		offset	
	6		5	5	;		16	
格式	bne rs,	rt, of	fset					
描述	if (GPR[:							
操作	<pre>if (GPR[rs] ≠ 0) PC ← PC + sign_extend(offset 0²) else PC ← PC + 4</pre>							
示例	bne \$s1,	\$s2, 8	3					
其他								

A.8 跳转指令

43. 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	$PC \leftarrow PC3128 \ instr_index\ 0^2$											
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。											

44. 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	$PC \leftarrow PC3128 \ instr_index\ 0^2$ $GPR[31] \leftarrow PC + 4$										

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。

45. jalr: 跳转并链接

	31	26	25	21	20	16	15		11	10	6	5		0	
编码	specia 00000		rs		0 0000	00		rd		00	0 0000	jalr 001001			
	6		5		5			5			5		6		
格式	jalr r	d, r	s												
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	jal my_function_name														
其他	jalr 与 jr 配套使用。jal 用于调用函数,jr 用于函数返回。														

46. jr: 跳转至寄存器

	31	26	25	21	20		11	10	6	5		0
编码	speci 00000			rs		0 00 0000 0000		00	0 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。											

A.9 数据传输指令

47. mfhi: 读 HI 寄存器

	31	26	25	21	20		15	11	10	6	5		0
编码	spe 0000				0000 0000			rd	C	0 00000			
	6	i		1	.0			5		5		6	
格式	mfhi rd												
描述	GPR[rd] ← HI												
操作	GPR[rd] ← HI												
示例	mfhi	\$s1											
其他	当乘法/除法计算完毕后,需要用 mfhi 读取相应的结果。												

48. mflo: 读 LO 寄存器

	(-)	31 26		25 21 20		15 11		10 6		5 0		0	
000000 000000000 00000 010010	编码	'		0 00 0000 0000			rd		0 00000		mfhi 010010		

	6	10	5	5	6
格式	mflo rd				
描述	GPR[rd] ← LO				
操作	GPR[rd] ← LO				
示例	mflo \$s1				
其他	当乘法/除法计算完毕后,	需要用 mflo 读取相应	的结果。		

49. mthi: 写 HI 寄存器

	31	26	25	21	20		15	11	10	6	6	5		0
编码	spec 0000			rs		0	0 0 0000 00		00			(mthi 010001	
	6			5			15	5					6	
格式	mthi	rs												
描述	HI ← GPR[rd]													
操作	HI ← GPR[rd]													
示例	mthi \$s1													
其他	mthi/mtlo 只在进行中断响应是需要使用。此时与 mfhi/mflo 配套使用确保被中断程序的 乘除法运算在中断响应结束后能够得到正确结果。													

50. mtlo: 写 LO 寄存器

	31	26	25	21	20		15	1	1	10	(6	5		0
编码	special rs					0 000 0000 0000 0000									
	6	6 5 15 6													
格式	mtlo	rs													
描述	LO ←	LO ← GPR[rd]													
操作	LO ←	LO ← GPR[rd]													
示例	mtlo	mtlo \$s1													
其他		mthi/mtlo 只在进行中断响应是需要使用。此时与 mfhi/mflo 配套使用确保被中断程序的 乘除法运算在中断响应结束后能够得到正确结果。													

A.10CP0 指令

51. eret: 异常返回

	31 26	25	25 21 20 16 15 11 10								
编码	COP0 010000										
	6		20 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										

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)否则保存 PC。

52. mfc0: 读 CPO 寄存器

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

53. mtc0: 写 CPO 寄存器

	31	26	25	21	20		16	15		11	10		0		
编码	COP 01000	-		ntc0 0100		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															

A.11系统指令

54. 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									

55. syscall: 系统调用

	31 26	25 6	5		0
编码	SPECIAL	code		BREAK	
,,, , ,,,,	000000		(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						