# 附录A MIPS-C 指令集

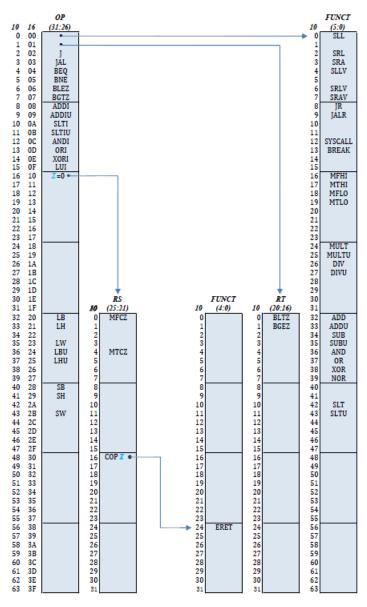
## A.1 MIPS-C 指令表

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	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)	25Н	R[rt] = {16'b0, Mem[GPR[rs]+ sign_ext(offset)][15:0]}
	LW	加载字	23H	R[rt] = Mem[GPR[rs]+sign_ext(offset)]
	SB	存储字节	28H	<pre>Mem[GPR[rs]+sign_ext(offset)][7:0] = R[rt][7:0]</pre>
保存	SH	存储半字	29Н	<pre>Mem[GPR[rs]+sign_ext(offset)][15:0] =     R[rt][15:0]</pre>
	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] X GRP[rt]
	MULTU	乘(无符号)	0/25Н	{HI, LO} = GPR[rs] X 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	0/0н	GPR[rd] = {GPR[rt][31-s:0], s{0}}
运算	SRL	逻辑右移	0/2Н	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	0/6Н	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	8H	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	4H	<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>
分支	BLEZ	小于等于0转移	6Н	<pre>if (GRP[rs] &lt;= 0)    PC = PC + 4 + BranchAddr</pre>
万又	BGTZ	大于0转移	7н	<pre>if (GRP[rs] &gt; 0)    PC = PC + 4 + BranchAddr</pre>
	BLTZ	小于0转移	特殊编码	<pre>if (GRP[rs] &lt;0)     PC = PC + 4 + BranchAddr</pre>
	BGEZ	大于等于0转移	特殊编码	<pre>if (GRP[rs] &gt;= 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]
	МТС0	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
гыл	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	16	15		0					
编码	lb 100000	base	r	t		offset						
	6	5	į	5		16						
格式	lb rt, offset(base)											
描述	GPR[rt] ←	memory[GP	R[base]	+offs	et]							
操作	Addr ← GPR[base] + sign_ext(offset)  memword ← memory[Addr]  byte ← Addr <sub>10</sub> GPR[rt] ← sign_ext(memword <sub>7+8*byte8*byte</sub> )											
示例	Ib \$v1, 3(\$s0)											

2. lbu: 加载无符号字节

	31	26	25	21	20		16	15		0		
编码	lbu 10010	00	ba	se		rt			offset			
	6		5			5			16			
格式	lb rt, offset(base)											
描述	GPR[rt	.] ←	memor	y[GPR	[bas	e]+c	offs	et]				
操作	Addr  GPR[base] + sign_ext(offset)  memword  memory[Addr]  byte  Addr <sub>10</sub> GPR[rt]  zero_ext(memword <sub>7+8*byte8*byte</sub> )											
示例	Ib \$v1, 3(\$s0)											

## 3. Ih: 加载半字

	31 26	25 21	20 10	15	0							
编码	lh 100001	base	rt		offset							
	6	5	5		16							
格式	lh rt, offset(base)											
描述	GPR[rt] ← memory[GPR[base]+offset]											
操作	memword ← byte ← Ac	PR[base] + s memory[Ado ddr <sub>1</sub> sign_ext(m	r]		;*byte)							
示例	lb \$v1, 3(\$s0)											
约束	Addr 必须是	2 的倍数(即 Add	dr <sub>o</sub> 必须为 0),	否则产	生地址错误异常							

#### 4. lhu: 加载无符号半字

	31	26	25	21	20	16	15		0		
编码	lhu 100101		base		rt						
	6		5		5			16			
格式	lhu rt, offset(base)										
描述	GPR[rt]	+	memory[G	PR [	base]+	offs	et]				
操作	memword byte ←	<b>←</b> Add	R[base] + memory[A dr <sub>1</sub> zero_ext	ddr	:]			,)			
示例	lb \$v1, 2(\$s0)										
约束	Addr 必须:	是 2	的倍数(即 /	Addr	。必须为	0),	测产生地:	址错误异常			

## 5. lw: 加载字

	31 26	25 21	20 16	15 (	0							
编码	lh 100011	base	rt	offset								
	6	5	5	16								
格式	lw rt, off	set(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 <sub>10</sub> 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 2	6 25	21	20		16	15		0			
编码	sb 101000	offset										
	6		5		5			16				
格式	sb rt, offset(base)											
描述	GPR[rt]	← mer	nory[GPR	[base	e]+c	offs	et]					
操作	Addr  GPR[base] + sign_extend(offset) byte  Addr10 memory[Addr] <sub>7+8*byte8*byte</sub> GPR[rt] <sub>7:0</sub>											
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	5				16			
格式	sh rt, offset(base)										
描述	GPR[rt]	_ ←	memory	[GPR	[base	e]+c	offse	et]			
操作	Addr ← byte ← memory	Add	dr1								
示例	sh \$v1, 24(\$s0)										
约束	Addr 必须	页是 2	的倍数(图	Add	r <sub>o</sub> 必須	页为 <b>C</b>	)),	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 101011 base rt offset												
	6		5		5		16						
格式	sh rt, offset(base)												
描述	GPR[rt]	] ←	memory[GP]	R[bas	e]+off	set]							
操作			R[base] + ; r] ← GPR[:		ext(of	fset	)						
示例	sw \$v1, 8(\$s0)												
约束	Addr 必须	页是 4	的倍数(即 Ad	dr <sub>10</sub> 业	必须为 00	,否	则产生地址错误异常						

# 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	special 000000		rs			rt		rd			0 000	add 100000		
	6	j	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	emp <sub>32</sub> <del>7</del> ignalE PR[rd]	PR[rs] <sub>3</sub> temp <sub>3</sub> Excepti ← tem	the on(I	n			] 31  0	GPR[	rt])				
示例	add \$s1, \$s2, \$s3													
其他	temp <sub>32</sub> ≠ temp <sub>31</sub> 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	(	ŝ	5		0
编码	specia 00000			rs		rt			rd		00	0 0000			addu 100001	
	6			5		5 5						6				
格式	addu r	ddu rd, rs, rt														
描述	GPR[rd	PR[rd]  GPR[rs] + GPR[rt]														
操作	GPR[rd	GPR[rd] ← GPR[rs] + GPR[rt]														
示例	addu \$	addu \$s1, \$s2, \$s3														
其他																

## 11. and: 与

	31	26	25	21	20		16	15		11	10	(	6	5		0
编码	specia 00000		1	S		rt			rd		0	0 0000			addu 100100	
	6			5	5 5 5									6		
格式	and ro	and rd, rs, rt														
描述	GPR[rd	GPR[rd] ← GPR[rs] AND GPR[rt]														
操作	GPR[rd	GPR[rd] ← GPR[rs] AND GPR[rt]														
示例	and \$s	and \$s1, \$s2, \$s3														
其他																

#### 12. div: 符号除

	31 26	25	21 20	1	.6	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[rt]										
	商存放在LO智	F存器,余数存放在HI寄存器									

操作	LO ← GPR[rs] div GPR[rt]
1米1ト	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		16	15		6	5		0
编码	speci 0000			rs		rt			0 00 0000 0000		0	div 11011	
	6			5		5			10			6	
格式	divu	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	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 di	.vu 为	无符号	}除法,原	<b></b> 「以对	其进	行 0	扩展 1	位后再进行运算。	D.			

#### 14. 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) 乘积低32位	-	-			EHI寄存器。所有操作数	:均为7	有符号数。	
操作	prod ← G HI ← prod LO ← prod	d <sub>6332</sub>	GPF	[rt]					
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	0 00 0000 0000		multu 011001	
	6		5		5	10		6	
格式	multu rs,	rt							
描述	(HI, LO) 乘积低32位					EHI寄存器。所有操作数比	り为う	无符号数。	
操作	prod ← ( HI ← pro LO ← pro	d <sub>6332</sub>		(0  GI	PR[rt]				
示例	multu \$s1	, \$s	2						

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		rs				rt			rd		0	00000		nor 100111		
	6	6 5 5							5 5							6	
格式	nor rd,	nor rd, rs, rt															
描述	GPR[rd]	GPR[rd] ← GPR[rs] NOR GPR[rt]															
操作	GPR[rd]	GPR[rd] ← GPR[rs] NOR GPR[rt]															
示例	nor \$si	nor \$s1, \$s2, \$s3															
其他																	

#### 17. or: 或

	31	26	25	21	20		16	15		11	10	6	5		0
编码	specia 00000	00 rs				rt			rd		0(	0 0000	or 100101		
	6		Ţ	5		5			5			5		6	
格式	or rd,	or rd, rs, rt													
描述	GPR[rd	.] ←	GPR[r	s] 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 0000		
	6	00000 5 5 5 6											
格式	sll rd, rt	ll rd, rt, s											
描述	GPR[rd] ←	GPR[rd] ← GPR[rt] << s											
操作	GPR[rd] ←	GPR[rd] ← GPR[rt] <sub>(31-s)0</sub>    0 <sup>s</sup>											
示例	sll \$s1, \$s2, 5												
其他	sll \$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 26	25 2	21	20	16	15		11	10	6	5	0
编码	special 000000	rs		rt			rd		00	0 0000	sll <sup>-</sup> 0001	
	6	5		5		5			5		6	
格式	sllv rd, r	sllv rd, rt, rs										
描述	GPR[rd] ← GPR[rt] << GPR[rs]											
操作	s ← GPR[rs] <sub>40</sub>											

	GPR[rd] ← GPR[rt] <sub>(31-s)0</sub>    0 <sup>s</sup>
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[rs	s] <	GPR	[rt]	)									
操作	GPR[rd]	+	(GPR[rs	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	25	21	20		16	15		11	10		6	5		0
编码	speci 00000			0		rt			rd			s			sltu 101011	
	6		00	000		5			5			5			6	
格式	sltu r	ltu rd, rs, rt														
描述	GPR[ro	PR[rd] ← (GPR[rs] < GPR[rt])														
操作	GPR[ro	$PR[rd] \leftarrow (0  GPR[rs] < 0  GPR[rt]) ? 0^{31}  1 : 0^{32}$														
示例	sltu \$	ltu \$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		(	)		rt			rd			S			sra 000011	
	6		000	000		5			5			5			6	
格式	sra rd	ra rd, rt, s														
描述	GPR[rd	PR[rd] ← GPR[rt] >> s														
操作	GPR[rd	PR[rd] ← GPR[rt] >> s  PR[rd] ← GPR[rt] <sub>31</sub> <sup>s</sup>    GPR[rt] <sub>31s</sub>														
示例	sra \$s	ra \$s1, \$s2, 5														
其他																

#### 23. srav: 算术可变右移

	31	26	25		21	20		16	15		11	10	6	5		0
编码	specia 00000			rs			rt			rd		0000	00		srav 000111	
	6			5			5			5		5			6	
格式	srav ro	d, r	t, r	s												

描述	GPR[rd] ← GPR[rt] >> GPR[rs]
操作	s  GPR[rs] <sub>40</sub>
1米11-	GPR[rd] ← GPR[rt] <sub>31</sub> <sup>s</sup>    GPR[rt] <sub>31s</sub>
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	ı
	6		0000	00		5			5			5			6	
格式	srl rd	rd, rt, s														
描述	GPR[rd	R[rd] ← GPR[rt] >> s														
操作	GPR[rd	.] ←	0°    GPF	[rt]	31s											
示例	srl \$s	1 \$s1, \$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	speci 0000		r	5		rt			rd		0	0000		srl\ 0001	
	6		5			5			5			5		6	
格式	srlv	rlv rd, rt, rs													
描述	GPR[ro	GPR[rd] ← GPR[rt] >> GPR[rs]													
操作	s ← G	s + GPR[rs] <sub>40</sub>													
1米1下	$GPR[rd] \leftarrow 0^{s}   GPR[rt]_{31s}$														
示例	srlv \$s1, \$s2, \$s3														
其他	GPR[rs	s]的位	7.31 至亿	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			0 000			sub 100010	
	6		5			5			5			5			6	
格式	sub r	d, rs	, rt													
描述	GPR[r															
操作	if ten Si else	GPR[rd] ← temp														
示例	sub \$	sub \$s1, \$s2, \$s3														
其他			mp <sub>31</sub> 代》 出,则 s													

## 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		00	0 0000		subu 100011	
	6	5	5			5			5		6	
格式	sub rd, rs	rd, rs, rt										
描述	GPR[rd] ←	rd]  GPR[rs] - GPR[rt]										
操作	GPR[rd] ←	[rd] ← GPR[rs] - GPR[rt]  [rd] ← GPR[rs] - GPR[rt]										
示例	sub \$s1, \$	b \$s1, \$s2, \$s3										
其他	subu 不考虑 即结果为非负	减法溢出。例 值。	如 0×000	0_00	00 -	- 0x	FFFF	_FFFI	F = 0x	000	0_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		6	
格式	xor rd	r rd, rs, rt													
描述	GPR[rd	R[rd]  GPR[rs] XOR GPR[rt]													
操作	GPR[rd	.] ←	GPR[	rs] XO	R GP	R[rt	:]								
示例	xor \$s	r \$s1, \$s2, \$s3													
其他															

# A.6 R-I 运算指令

## 29. addi: 符号加立即数

	31	26	25	21	20	16	15	0				
编码	add 00100		rs			rt		immediate				
	6		5			5		16				
格式	addi r	t, r	s, imme	ediat	:e							
描述	GPR[rt	:] ←	GPR[rs	] +	immed	iate.						
操作	if tem Sig else	GPR[rt] ← temp										
示例	addi \$	addi \$s1, \$s2, -1										
其他			emp <sub>31</sub> 代表 出,则 a				,					

## 30. addiu: 无符号加立即数

	31 26	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	GPR[rt] ← GPR[rs] + sign_extend(immediate)
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[rs] AND immediate										
操作	GPR[rt] ←	GPR[rs]	AN	D zei	ro_e	xte	end(immediate)				
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	1	0 1 00000			rt			immediate	
	6 5 5 16									
格式	lui rt, immediate									
描述	lui rt, immediate $\parallel 0^{16}$									
操作	lui rt	, im	media	te    0 <sup>1</sup>	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 00110:	1	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]	<b>+</b>	GPR[rs]	OR	zer	о_ех	ten	d(immed	iate)		
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 rt, rs, immediate											
描述	<pre>GPR[rt] ← (GPR[rs] &lt; immediate)</pre>											
操作	GPR[rt	] ←	(0  GE	R[rs]	< 0	sig	n_ex	tend(immediate)) ? $0^{31} \  1 : 0^{32}$				
示例	sltiu \$s1, \$s2, 0xAABB											
其他	"无符号"是误导											

36. xori: 异或立即数

	31 26	25 2	1 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] ←	GPR[rs] >	OR zer	o_exte	nd(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	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 + sign_extend(offset  0²) else   PC ← PC + 4</pre>											
示例	beq \$s1, \$s2, -2											
其他												

38. bgez: 大于等于 0 时转移

	31	26	25	21	20	16	15	0				
编码	000001		rs	rs bgez 00001				offset				
	6 5				5			16				
格式	bgez rs, offset											
描述	if (GPR[rs] >= 0) then 转移											
操作	<pre>if (GPR[rs] &gt;= 0)    PC ← PC + sign_extend(offset  0²) else    PC ← PC + 4</pre>											
示例	bgez \$s1, -2											
其他												

## 39. bgtz: 大于 0 时转移

	31 2	6 25	21	20	16	15		0				
编码	bgtz 000111	rs		0000	00		offset					
	6	5		5			16					
格式	bgtz rs, offset											
描述	if (GPR[rs] > 0) then 转移											
操作	else	rs] > 0) PC + sig PC + 4	ın_ex	tend(o	ffset	0 <sup>2</sup> )						
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 000110		rs		0 00000			offset				
	6		5		5	5		16				
格式	blez rs, offset											
描述	if (GPR[rs] <= 0) then 转移											
操作	PC else	← P	C + sig		tend(c	offset	0 <sup>2</sup> )					
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, o	ffset		

描述	if (GPR[rs] < 0) then 转移
操作	<pre>if (GPR[rs] &lt; 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 26	25 21	20	16	15 0					
编码	bne 000101	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	<pre>if (GPR[rs] ≠ 0)    PC ← PC + sign_extend(offset  0²) else    PC ← PC + 4</pre>									
示例	bne \$s1, \$s2, 8									
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	-	目关的转移指令。当把 4GB 划分为 16 个 256MB 区域, j 指令可以在当前 6MB 区域内任意跳转。										
操作	PC ← PC31.	28  instr_index  0 <sup>2</sup>										
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 ← PC31. GPR[31] ←	28  instr_index  0 <sup>2</sup> 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	special 0000000         rs         0 000000         rd         0 00000         jalr 00000														
	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	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	specia 00000			rs		0 00 0000 0000		00	0 0000	jr 001000		
	6 5					10		5				
格式	jr rs											
描述	PC ← (	GPR[1	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	1	11	10	6	5		0	
编码	spe 0000				00000		rd			0(	0 0000		mfhi 010000		
	6	i		1	.0		5				5				
格式	mfhi rd														
描述	GPR[r	rd] ←	HI												
操作	GPR[r	rd] ←	HI												
示例	mfhi \$s1														
其他	当乘法	/除法计	算完毕	后,需	要用 mfl	ni 读取材	目应的	结果。							

#### 48. mflo: 读 LO 寄存器

12	31	26	25	21	20		15	11	10	6	5		0
编码	special 000000		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	的结果。		

#### 49. mthi: 写 HI 寄存器

	31 2	6 2	25	21	20	1	.5	11	10	6	5		0
编码	special 000000		rs			0 000 0000 0000 0000							
	6		5				15					6	
格式	mthi rs												
描述	HI ← GP	₹[rd	.]										
操作	HI ← GP	R[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	11	10	6	5		0		
编码	special 000000		rs		000 0000 00	00 00	00			mthi 10011			
	6		5		15				6				
格式	mtlo rs												
描述	LO ← GPR[rd]												
操作	LO ← GPR[:	rd]											
示例	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	21	20	16	15	11	10	0
编码	COP0 010000								eret 011000
	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	COP 0100			nfc0 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[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	COP0 mtc0 010000 00100		rt			rd			0 0 0000 0000				
	6			5		5			5				
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	SignalExce	ption(breakpoint)							
示例	break								
其他									

#### 55. syscall: 系统调用

	31 26	25 6	5		0
编码	SPECIAL 000000	code		BREAK 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					