

附录A MIPS-C 指令集

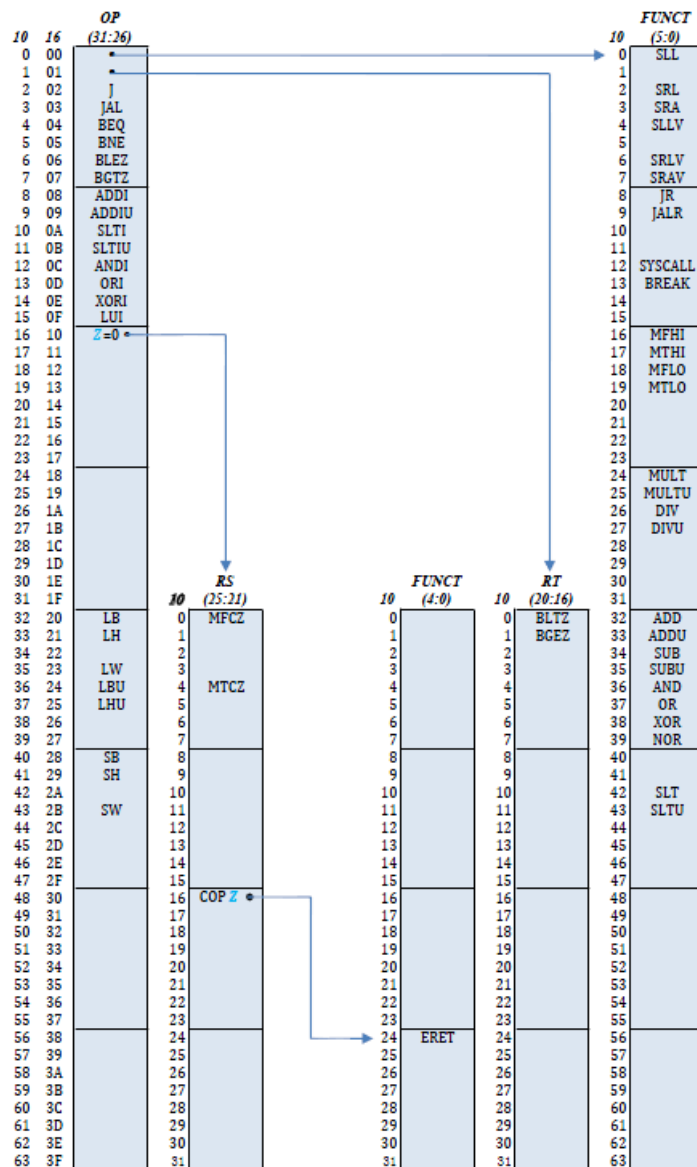
A.1 MIPS-C 指令表

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 CP0、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)
加载	LB	加载字节	20H/24H	$R[rt] = \{24\{\text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][7]\}, \text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][7:0]\}$
	LBU	加载字节 (无符号)	24H	$R[rt] = \{24'b0, \text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][7:0]\}$
	LH	加载半字	21H	$R[rt] = \{16\{\text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][15]\}, \text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][15:0]\}$
	LHU	加载半字 (无符号)	25H	$R[rt] = \{16'b0, \text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][15:0]\}$
	LW	加载字	23H	$R[rt] = \text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})]$
保存	SB	存储字节	28H	$\text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][7:0] = R[rt][7:0]$
	SH	存储半字	29H	$\text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})][15:0] = R[rt][15:0]$
	SW	存储字	2BH	$\text{Mem}[GPR[rs] + \text{sign_ext}(\text{offset})] = R[rt]$
R-R 运算	ADD	加	0/32H	$GPR[rd] = GPR[rs] + GRP[rt]$
	ADDU	无符号加	0/33H	$GPR[rd] = GPR[rs] + GRP[rt]$
	SUB	减	0/34H	$GPR[rd] = GPR[rs] - GRP[rt]$
	SUBU	无符号减	0/35H	$GPR[rd] = GPR[rs] - GRP[rt]$
	MULT	乘	0/24H	$\{HI, LO\} = GPR[rs] \times GRP[rt]$
	MULTU	乘(无符号)	0/25H	$\{HI, LO\} = GPR[rs] \times GRP[rt]$
	DIV	除	0/26H	$\{HI, LO\} = GPR[rs] / GRP[rt]$
	DIVU	除(无符号)	0/27H	$\{HI, LO\} = GPR[rs] / GRP[rt]$
	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	0/3H	$GPR[rd] = \{s\{GPR[rt][31]\}, GPR[rt][31:s]\}$
	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	0/6H	$GPR[rd] = \{v\{0\}, GPR[rt][31:v]\}$
	SRAV	算术可变右移	0/7H	$GPR[rd] = \{v\{GPR[rt][31]\}, GPR[rt][31:v]\}$
	AND	与	0/36H	$GPR[rd] = GPR[rs] \& GRP[rt]$
	OR	或	0/37H	$GPR[rd] = GPR[rs] \mid GRP[rt]$
R-I 运算	XOR	异或	0/38H	$GPR[rd] = GPR[rs] \wedge GRP[rt]$
	NOR	或非	0/39H	$GPR[rd] = \sim(GPR[rs] \mid GRP[rt])$
R-I 运算	ADDI	加立即数	8H	$GPR[rd] = GPR[rs] + \text{SignExtImm}$
	ADDIU	加立即数 (无符号)	9H	$GPR[rd] = GPR[rs] + \text{SignExtImm}$

	ANDI	与立即数	CH	$GPR[rd] = GPR[rs] \& ZeroExtImm$
	ORI	或立即数	DH	$GPR[rd] = GPR[rs] \mid ZeroExtImm$
	XORI	异或立即数	EH	$GPR[rd] = GPR[rs] \wedge ZeroExtImm$
	LUI	立即数加载至高 位	FH	$GPR[rd] = \{imm, 16'b0\}$
	SLTI	小于立即数置 1	AH	$GPR[rt] = (GPR[rs] < SignExtImm) ? 1 : 0$
	SLTIU	小于立即数置 1 (无符号号)	BH	$GPR[rt] = (GPR[rs] < SignExtImm) ? 1 : 0$
分支	BEQ	等于转移	4H	if (GRP[rs] == GPR[rt]) PC = PC + 4 + BranchAddr
	BNE	不等转移	5H	if (GRP[rs] != GPR[rt]) PC = PC + 4 + BranchAddr
	BLEZ	小于等于 0 转移	6H	if (GRP[rs] <= 0) PC = PC + 4 + BranchAddr
	BGTZ	大于 0 转移	7H	if (GRP[rs] > 0) PC = PC + 4 + BranchAddr
	BLTZ	小于 0 转移	特殊编码	if (GRP[rs] < 0) PC = PC + 4 + BranchAddr
	BGEZ	大于等于 0 转移	特殊编码	if (GRP[rs] >= 0) PC = PC + 4 + BranchAddr
跳转	J	跳转	2H	PC = JumpAddr
	JAL	跳转并链接	3H	PC = JumpAddr; GPR[31] = PC + 4
	JALR	跳转并链接寄存 器	0/8H	PC = GPR[rs]; GPR[rd] = PC + 4
	JR	跳转寄存器	0/9H	PC = GPR[rs]
传输	MFHI	读 HI 寄存器	0/16H	GPR[rd] = HI
	MFLO	读 LO 寄存器	0/17H	GPR[rd] = LO
	MTHI	写 HI 寄存器	0/18H	HI = GPR[rd]
	MTLO	写 LO 寄存器	0/19H	LO = GPR[rd]
特权	ERET	异常返回	10/18H	PC = EPC; 还需要对 CP0 的其他寄存器做处理
	MFC0	读 CP0 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	0/13H	EPC = PC+4; PC = 异常处理地址; CP0 的其他寄存器做处理
	SYSCALL	系统调用异常	0/12H	EPC = PC+4; PC = 异常处理地址; CP0 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

编码	31	26	25	21	20	16	15	0
	lb 100000		base		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 _{1..0}							
	GPR[rt] ← sign_ext(memword _{7+8*byte..8*byte})							
示例	lb \$v1, 3(\$s0)							

2. lbu: 加载无符号字节

	$\text{GPR}[\text{rt}] \leftarrow \text{memory}[\text{Addr}]$
示例	<code>lw \$v1, 8(\$s0)</code>
约束	Addr 必须是 4 的倍数(即 $\text{Addr}_{1..0}$ 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

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

7. sh: 存储半字节

编码	31	26	25	21	20	16	15	0
	sh 101001		base		rt		offset	
	6		5		5		16	
格式	sh rt, offset(base)							
描述	GPR[rt] ← memory[GPR[base]+offset]							
操作	Addr ← GPR[base] + sign_extend(offset) byte ← Addr1 memory[Addr] _{15+16*byte..16*byte} ← GPR[rt] _{15:0}							
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

编码	31	26	25	21	20	16	15	0
	sw 101011		base		rt		offset	
	6		5		5		16	
格式	sh rt, offset(base)							
描述	GPR[rt] ← 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 _{1..0} 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

27. **subu**: 无符号减

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000		rs		rt		rd		0 00000		subu 100011	
	6		5		5		5		5		6	
格式	sub rd, rs, rt											
描述	GPR[rd] ← GPR[rs] - GPR[rt]											
操作	GPR[rd] ← GPR[rs] - GPR[rt]											
示例	sub \$s1, \$s2, \$s3											
其他	subu 不考虑减法溢出。例如 0x0000_0000 - 0xFFFF_FFFF = 0x0000_0001，即结果为非负值。											

28. **xor**: 异或

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000		rs		rt		rd		0 00000		xor 100110	
	6		5		5		5		5		6	
格式	xor rd, rs, rt											
描述	GPR[rd] ← GPR[rs] XOR GPR[rt]											
操作	GPR[rd] ← GPR[rs] XOR GPR[rt]											
示例	xor \$s1, \$s2, \$s3											
其他												

A.6 R-I 运算指令

29. **addi**: 符号加立即数

编码	31	26	25	21	20	16	15	0
	addi 001000		rs		rt		immediate	
	6		5		5		16	
格式	addi rt, rs, immediate							
描述	GPR[rt] ← GPR[rs] + immediate							
操作	temp ← (GPR[rs] ₃₁ GPR[rs]) + sign_extend(immediate) if temp ₃₂ ≠ temp ₃₁ then SignalException(IntegerOverflow) else GPR[rt] ← temp endif							
示例	addi \$s1, \$s2, -1							
其他	temp ₃₂ ≠ temp ₃₁ 代表计算结果溢出。 如果不考虑溢出，则 addi 与 addiu 等价。							

30. **addiu**: 无符号加立即数

编 码	31 26 25 21 20 16 15									
-----	--	--	--	--	--	--	--	--	--	--

格式	addiu rt, rs, immediate
描述	$GPR[rt] \leftarrow GPR[rs] + immediate$
操作	$GPR[rt] \leftarrow GPR[rs] + sign_extend(immediate)$
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

编码	31	26	25	21	20	16	15	0
	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] AND zero_extend(immediate)							
示例	andi \$s1, \$s2, 0x55AA							
其他								

32. lui: 立即数加载至高位

编码	31	26	25	21	20	16	15	0
	lui 001111		0 00000		rt		immediate	
	6		5		5		16	
格式	lui rt, immediate							
描述	lui rt, immediate 0 ¹⁶							
操作	lui rt, immediate 0 ¹⁶							
示例	lui \$s1, 0x55AA							
其他								

33. ori: 或立即数

编码	31	26	25	21	20	16	15	0
	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 zero_extend(immediate)							
示例	ori \$s1, \$s2, 0x55AA							
其他								

34. slti: 小于立即数置 1(有符号)

编码	31	26	25	21	20	16	15	0
	slti 001010	rs			rt		immediate	
	6	5			5		16	

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

35. sltiu: 小于立即数置 1(无符号)

编码	31	26	25	21	20	16	15	0
	sltiu 001011		rs		rt		immediate	
	6		5		5		16	
格式	sltiu rt, rs, immediate							
描述	$\text{GPR}[\text{rt}] \leftarrow (\text{GPR}[\text{rs}] < \text{immediate})$							
操作	$\text{GPR}[\text{rt}] \leftarrow (0 \parallel \text{GPR}[\text{rs}] < 0 \parallel \text{sign_extend}(\text{immediate})) ? 0^{31} \parallel 1 : 0^{32}$							
示例	sltiu \$s1, \$s2, 0xAABB							
其他	“无符号”是误导							

36. xori: 异或立即数

编码	31	26	25	21	20	16	15	0
	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] XOR zero_extend(immediate)							
示例	xori \$s1, \$s2, 0x55AA							
其他								

A.7 分支指令

37. beq: 相等时转移

编码	31	26	25	21	20	16	15	0
	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	if (GPR[rs] == GPR[rt]) PC ← PC + sign_extend(offset 0 ²) else PC ← PC + 4							
示例	beq \$s1, \$s2, -2							
其他								

38. bgez: 大于等于 0 时转移

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

42. bne: 不等于时转移

编码	31	26	25	21	20	16	15	0
	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	if (GPR[rs] ≠ 0) PC ← PC + sign_extend(offset 0 ²) else PC ← PC + 4							
示例	bne \$s1, \$s2, 8							
其他								

A.8 跳转指令

43. j: 跳转

编 码	31	26	25	0
	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	$PC \leftarrow PC_{31..28} instr_index 0^2$			
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。			

44. jal: 跳转并链接

编码	31	26	25	0
	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	$PC \leftarrow PC_{31..28} instr_index 0^2$ $GPR[31] \leftarrow PC + 4$			

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

编码	31	26	25	21	20	16	15	11	10	6	5	0
	special 000000		rs		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	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

编码	31	26	25	21	20	11	10	6	5	0
	special 000000	rs		0 00 0000 0000			0 00000		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

编码	31	26	25	21	20	15	11	10	6	5	0
	special 000000	0 00 0000 0000				rd	0 00000		mfhi 010000		
	6	10				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

编码	31	26	25	21	20	15	11	10	6	5	0
	special 000000	0 00 0000 0000			rd			0 00000	mflo 010010		
	6	5			5			5	6		

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)否则保存 PC。
----	--

52. mfc0: 读 CP0 寄存器

编码	31	26	25	21	20	16	15	11	10	0
	COP0 010000		mfc0 00000		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 CP0 寄存器

编码	31	26	25	21	20	16	15	11	10	0
	COP0 010000		mtc0 00100		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

编码	31	26	25	6	5	0
	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

编码	312625		650	
	SPECIAL 000000	code		BREAK 001100
	620		6	
格式	syscall			

描述	产生系统调用异常
操作	SignalException(systemcall)
示例	syscall
其他	