參考資料

Name	Register number	Usage	Preserved on call?
$zero	0	The constant value 0	n.a.
$v0–$v1	2–3	Values for results and expression evaluation	no
$a0–$a3	4–7	Arguments	no
$t0–$t7	8–15	Temporaries	no
$s0–$s7	16–23	Saved	yes
$t8–$t9	24–25	More temporaries	no
$gp	28	Global pointer	yes
$sp	29	Stack pointer	yes
$fp	30	Frame pointer	yes
$ra	31	Return address	yes

1

MIPS instruction encoding

op (31:26)

28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	Bltz/gez	jump (j)	jump & link (jal)	branch eq (beq)	branch ne (bne)	blez	bgtz
1(001)	add immediate	addiu	set less than imm.	sltiu	andi	ori	xori	load upper imm
2(010)	TLB	FIPt						
3(011)								
4(100)	load byte	lh	lwl	load word (lw)	lbu	lhu	lwr	
5(101)	store byte	sh	swl	store word (sw)			swr	
6(110)	lwc0	lwc1						
7(111)	swc0	swc1						

MIPS instruction encoding

op (31:26) = 000000 (R-format) funct(5:0)

2-0 5-3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	sll		srl	sra	sllv		srlv	srav
1(001)	jump reg.	jalr			syscall	break		
2(010)	mfhi	mthi	mflo	mtlo				
3(011)	mult	multu	div	divu				
4(100)	add	addu	subtract (sub)	subu	and	or	xor	nor
5(101)			set l.t.	sltu				

名稱	欄位						註解
欄位大小	6 位元	5 位元	5 位元	5 位元	5 位元	6 位元	所有 MIPS 的指令都是 32 位元
R 格式	op	rs	rt	rd	shamt	funct	算術指令格式
I 格式	op	rs	rt	位址 / 立即值			傳輸、條件式跳躍、 imm 格式
J 格式	op	目標位址					跳躍指令格式

名稱	欄　位						註　解
欄位大小	6 位元	5 位元	5 位元	5 位元	5 位元	6 位元	所有 MIPS 指令的長度都是 32 位元
R 格式	op	rs	rt	rd	shamt	funct	算術指令格式
I 格式	op	rs	rt	位址／立即值			傳輸、分支、imm 格式
J 格式	op	目標位址					跳躍指令格式

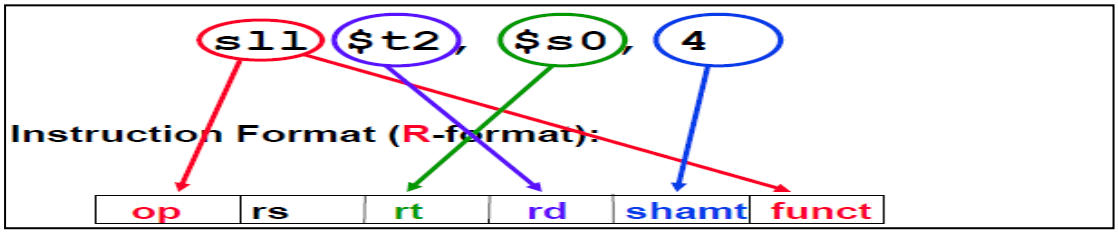
參考資料

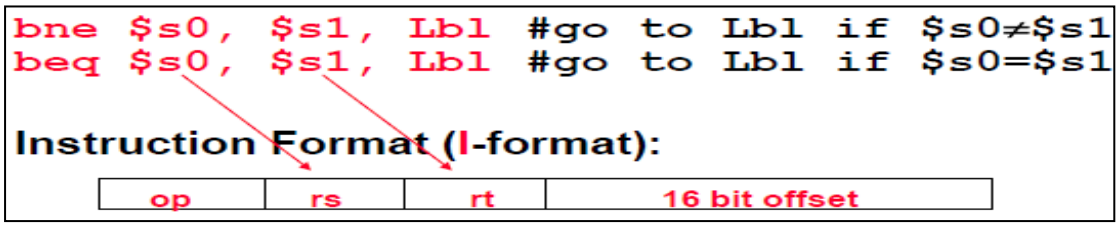
2

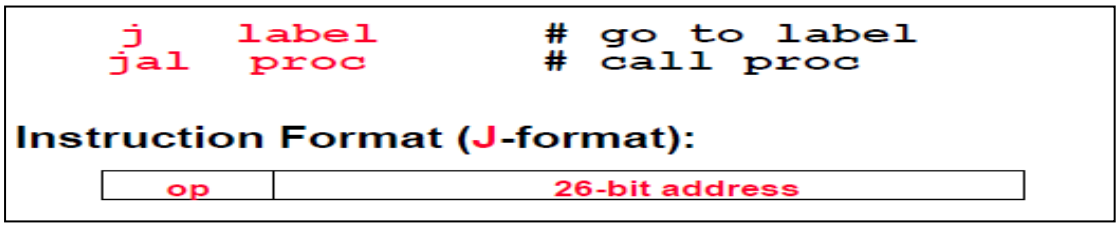
R-type (register) format

• General form:　op rd, rs, rt　**(ex.　Add/Sub　rd, rs, rt)**

• Some MIPS R-type Instructions:　rs=0







MIPS machine language

Name	Format	Example						Comments
add	R	0	18	19	17	0	32	add $s1,$s2,$s3
sub	R	0	18	19	17	0	34	sub $s1,$s2,$s3
addi	I	8	18	17	100			addi $s1,$s2,100
lw	I	35	18	17	100			lw $s1,100($s2)
sw	I	43	18	17	100			sw $s1,100($s2)
Field size		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	R	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	I	op	rs	rt	address			Data transfer format

MIPS 組合語言

分類	指 令	舉 例	意 義	註 解
資料傳輸	載入字組	lw $s1,20($s2)	$s1=**Memory**[$s2+**20**]	字組由記憶體載入至暫存器
	儲存字組	sw $s1,20($s2)	**Memory**[$s2+**20**]=$S1	字組由暫存器儲存至記意體
	載入半字組	lh $s1,20($s2)	$s1=**Memory**[$s2+**20**]	半字組由記憶體載入至暫存器
	載入無號半字組	lhu $s1,20($s2)	$s1=**Memory**[$s2+**20**]	無號的半字組由記憶體載入至暫存器
	儲存半字組	sh $s1,20($s2)	**Memory**[$s2+**20**]=$s1	半字組由暫存器儲存至記憶體
	載入位元組	lb $s1,20($s2)	$s1=**Memory**[$s2+**20**]	位元組由記憶體載入至暫存器
	載入無號位元組	lbu $s1,20($s2)	$s1=**Memory**[$s2+**20**]	無號的位元組由記憶體載入至暫存器
	儲存位元組	sb $s1,20($s2)	**Memory**[$s2+**20**]=$s1	位元組由暫存器儲存至記憶體
	載入連結的字元組	ll $s1,20($s2)	$s1=**Memory**[$s2+**20**]	作為不可分割的（記憶體與儲存器內容）交換中第一部份的載入字元組
	條件式儲存字元組	sc $s1,20($s2)	**Memory**[$s2+**20**]=$s1; $s1=0 或 1	作為不可分割的（記憶體與暫存器內容）交換中第二部份的儲存字組
	載入上半部立即值	lui $s1,20	$s1=**20*2**^{16}	載入常數至較高的 16 位元
邏輯	及	and $s1,$s2,$s3	$s1=$s2 **&** $s3	三個暫存器運算元：逐位元的及運算
	或	or $s1,$s2,$s3	$s1=$s2|$s3	三個暫存器運算元：逐位元的或運算
	反或	nor $s1,$s2,$s3	$s1=～($s2|$s3)	三個暫存器運算元：逐位元的反或運算
	及立即值	andi $s1,$s2,20	$s1=$s2 **& 20**	暫存器與常數做逐位元的及運算
	或立即值	ori $s1,$s2,20	$s1=$s2|**20**	暫存器與常數做逐位元的或運算
	邏輯左移	sll $s1,$s2,10	$s1=$s2<<**10**	左移常數個位元位置
	邏輯右移	srl $s1,$s2,10	$s1=$s2>>**10**	右移常數個位元位置

3

參考資料

MIPS 組合語言

分類	指 令	舉 例	意 義	註 解
條件式分支	若等於則分支	beq $s1,$s2,25	**若**($s1==$s2)**則前往 PC+4+100**	等於測試：PC 相對的分支
	若不等於則分支	bne $s1,$s2,25	**若**($s1!=$s2)**則前往 PC+4+100**	不等於測試：PC 相對的分支
	若小於則設定	slt $s1,$s2,$s3	**若**($s2<$s3) $s1=**1**; **否則** $s1=**0**	小於比較；用於 beq、bne
	無號若小於則設定	sltu $s1,$s2,20	**若**($s2<$s3) $s1=**1**; **否則** $s1=**0**	無號數的小於比較
	若小於立即值則設定	slti $s1,$s2,20	**若**($s2<**20**) $s1=**1**; **否則** $s1=**0**	小於某常數的比較
	無號若小於立即值則設定	sltiu $s1,S2,20	**若**($s2<**20**) $s1=**1**; **否則** $s1=**0**	無號數的小於某常數的比較
非條件式跳躍	跳躍	j 2500	**前往 10000**	跳至目的位址
	透過暫存器跳躍	jr $ra	**前往** $ra	用於 switch 敘述、程序反回
	跳躍並連結	jal 2500	$ra=**PC+4 前往 10000**	用於程序呼叫

MIPS 組合語言

分類	指 令	舉 例	意 義	註 解
算術	加法	add $s1,$s2,$s3	$s1=$s2+$s3	三個暫存器運算元
	減法	sub $s1,$s2,$s3	$s1=$s2-$s3	三個暫存器運算元
	加立即值	addi $s1,$s2,20	$s1=$s2+**20**	加上常數

參考資料

- $$\text{CPU Time} = \text{CPU Clock Cycles} \times \text{Clock Cycle Time}$$
- $$= \frac{\text{CPU Clock Cycles}}{\text{Clock Rate}}$$

$$\text{Clock Cycles} = \text{Instruction Count} \times \text{Cycles per Instruction}$$

$$\text{CPU Time} = \text{Instruction Count} \times \text{CPI} \times \text{Clock Cycle Time}$$

$$= \frac{\text{Instruction Count} \times \text{CPI}}{\text{Clock Rate}}$$

- 中央處理器時間=指令數×CPI×時脈週期，
- $$\text{中央處理器時間} = \frac{\text{指令數} \times \text{CPI}}{\text{時脈速率}}$$

- $$\text{中央處理器執行某程式需時} = \text{中央處理器執行該程式所需時脈數} \times \text{時脈週期時間}$$
- $$\text{中央處理器執行某程式需時} = \frac{\text{中央處理器執行該程式所需時脈數}}{\text{時脈速率}}$$