

# 第二章 汇编语言



华中科技大学  
电子信息与通信学院  
School of Electronic Information and Communications



# 学习目标

- MIPS常用指令：
  - 逻辑运算指令、移位指令，跳转指令
- 理解常用C语句运行原理



# 逻辑运算指令

逻辑操作	C语言运算符	汇编指令	指令实例	指令含义
逻辑左移	<<	sll	sll \$s1,\$s2,10	将寄存器\$s2中的值左移10位，高10位移除，低10位补充0，结果保存到\$s1中
逻辑右移	>>	srl	srl \$s1,\$s2,10	将寄存器\$s2中的值右移10位，低10位移除，高10位补充0，结果保存到\$s1中
寄存器位与	&	and	and \$s1,\$s2,\$s3	将寄存器\$s2与\$s3中的值按位相与，结果保存到寄存器\$s1中
立即数位与	&	andi	andi \$s1,\$s2,40	将寄存器\$s2的值与40按位相与，结果保存到寄存器\$s1中
寄存器位或		or	or \$s1,\$s2,\$s3	将寄存器\$s2与\$s3中的值按位相或，结果保存到寄存器\$s1中
立即数位或		ori	ori \$s1,\$s2,40	将寄存器\$s2的值与40按位相或，结果保存到寄存器\$s1中
位或非	~	nor	nor \$s1,\$s2,\$s3	将寄存器\$s2与\$s3中的值按位相或非，结果保存到寄存器\$s1中
异或	^	xor	xor \$s1,\$s2,\$s3	将寄存器\$s2与\$s3中的值按位相异或，结果保存到寄存器\$s1中



- 移位指令机器指令格式

op (6位)	rs (5位)	rt (5位)	rd (5位)	shamt (5位)	func (6位)
操作码的编码	0	源操作数寄存器的编码	目的操作数寄存器的编码	移位次数的编码	操作码功能编码



# 程序控制类指令

- 条件跳转类指令：beq, bne

beq R1,R2,L1 #当寄存器R1的值与寄存器R2的值相等时，跳转到L1处执行指令。

bne R1,R2,L1 #当寄存器R1的值与寄存器R2的值不相等时，跳转到L1处执行指令。

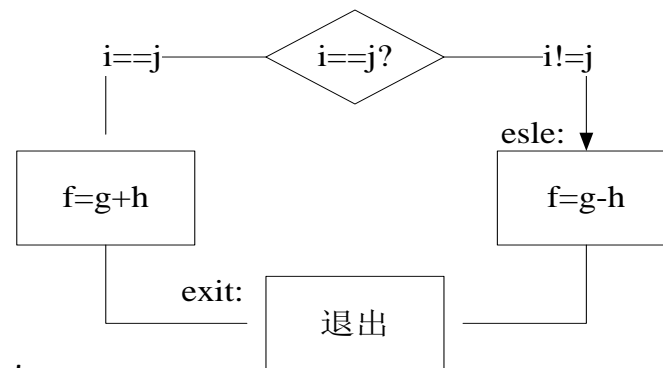
- 无条件跳转指令j

j L1#无条件跳转到L1处执行指令



## ● If控制

```
if (i==j)
    f=g+h;
else f=g-h;
```



变量i,j,f,g,h分别对应寄存器\$*s*0,\$*s*1,\$*s*2,\$*s*3,\$*s*4

```
bne $s0,$s1,else # i!=j,跳转到else处, 相等则顺序执行
add $s2,$s3,$s4
j exit      #处理完相等的情况后跳转到退出处
else: sub $s2,$s3,$s4
exit:
```




- for/while控制


```
while (save[i]==k)  
i+=1;
```

假定i, k为寄存器\$*s0*, \$*s2*, save的基地址保存在寄存器\$*s3*中

```
loop: sll $t1,$s0,2 #将i乘以4得到数组元素的偏移地址保存到暂存寄存器$t1中  
      add $t1,$t1,$s3 #将数组元素在内存中的地址保存到暂存寄存器$t1中  
      lw $t0,0($t1) #从内存中读取数组save中元素i的值保存到寄存器$t0中  
      bne $t0,$s2,exit #如果save[i]不等于k则跳转到退出处，否则顺序执行  
      addi $s0,$s0,1 #数组元素加1得到下一个元素  
      j loop #重复以上过程  
exit:
```

# 条件判断

- $=, !=$  

- $<, >, <=, >=$  

- 比较设置指令

<u>Instruction</u>	<u>Example</u>	<u>Meaning</u>	<u>Comments</u>
set less than	<code>slt \$1, \$2, \$3</code>	$\$1 = (\$2 < \$3)$	comp less than signed
set less than imm	<code>slti \$1, \$2, 100</code>	$\$1 = (\$2 < 100)$	comp w/const signed
set less than uns	<code>sltu \$1, \$2, \$3</code>	$\$1 = (\$2 < \$3)$	comp < unsigned
set l.t. imm. uns	<code>sltiu \$1, \$2, 100</code>	$\$1 = (\$2 < 100)$	comp < const unsigned





if (i<j)

f=g+h;

else f=g-h;

slt \$t0, \$s0,\$s1      #若\$s0(i)小于\$s1(j), \$t0=1,否则\$t0=0.

beq \$t0,\$zero,else

add \$s2,\$s3,\$s4

j exit

#处理完小于的情况后跳转到退出处

else:      sub \$s2,\$s3,\$s4

exit:



## • case/switch 控制

```
switch (i)
{
case 0:j=j+1;
break;
case 1:j=j+2;
break;
case 2:j=j+3;
break;
case 3:j=j+4;
break;
case 4: j=j+5;
break;
};
```

- i,j为寄存器 \$s0,\$s1,
- 常数0,1,2,3,4 分别保存在寄存器 \$t0,\$t1,\$t2,\$t3,\$t4中,
- 每个case对应的标号为: ca0,ca1,ca2,ca3,ca4。

### 逐次比较法

```
beq $s0,$t0,ca0 #比较i=0, 则跳转到ca0
beq $s0,$t1,ca1 #比较i=1, 则跳转到ca1
beq $s0,$t2,ca2 #比较i=2, 则跳转到ca2
beq $s0,$t3,ca3 #比较i=3, 则跳转到ca3
beq $s0,$t4,ca4 #比较i=4, 则跳转到ca4
j exit
ca0:addi $s1,$s1,1
j exit
ca1:addi $s1,$s1,2
j exit
ca2:addi $s1,$s1,3
j exit
ca3:addi $s1,$s1,4
j exit
ca4:addi $s1,$s1,5
exit:
```



● case/switch 控制

跳转表法

Jumptable:

.word ca0

.word ca1

.word ca2

.word ca3

.word ca4

偏移地址	
0000	ca0
0004	ca1
0008	ca2
000C	ca3
0010	ca4

bltz \$s0,exit      #小于0退出

Slti \$t0,\$s0,5     #小于5设置为1

Beq \$t0,\$zero,exit #大于等于5退出

sll \$t0,\$s0,2      #将i\*4得到标号在内存中的偏移地址

add \$s3,\$s3,\$t0    #偏移地址与基地址相加得到标号在内存中的地址

lw \$s4,0(\$s3)      #取出标号地址存放到寄存器\$s4中

jr \$s4             #跳转到\$s4所指示的标号处

Exit:

# sum\_pow2代码的MIPS汇编指令实现

- 假定pow2的地址存储在寄存器\$*v*1中，*b*、*c*分别用寄存器\$*t*0,\$*t*1表示，*a*、*ret*用\$*t*2、\$*t*3表示

```
int sum_pow2(int b, int c)
{
    int pow2[8] = {1, 2, 4, 8, 16, 32, 64, 128};
    int a, ret;
    a = b + c;
    if (a < 8)
        ret = pow2[a];
    else
        ret = 0;
    return(ret);
}
```

add \$t2,\$t0,\$t1	# a = b + c,
slti \$v0,\$t2,8	# \$v0 = a < 8
beq \$v0,\$zero, Exceed	# 跳转到 Exceed if \$v0 == 0
sll \$v0,\$t2,2	# \$v0 = a*4
addu \$v0,\$v0,\$v1	# \$v0 = pow2 + a*4
lw \$t3,0(\$v0)	# \$t3 = pow2[a]
j Return	# 跳转到 Return
Exceed: addu \$t3,\$zero,\$zero	# \$t3 = 0
Return:	