221900175-毛九弢-第四章

2023年11月1日 12:47

3. 假定某计算机中有一条转移指令，采用相对寻址方式，共占两字节，第一字节是操作码，第二字节是相对位移量(用补码表示)，CPU 每次从内存只能取一字节。假设执行到某转移指令时 PC 的内容为 200，执行该转移指令后要求转移到 100 开始的一段程序执行，则该转移指令第二字节的内容应该是多少？

$$ans = 100 - (200 + 2) = -102$$
$$= -(0110\ 0110)_2 = (1001\ 1010)_2$$

4. 假设地址为 1200H 的内存单元中的内容为 12FCH，地址为 12FCH 的内存单元的内容为 38B8H，而 38B8H 单元的内容为 88F9H。说明以下各情况下操作数的有效地址是多少？

(1) 操作数采用变址寻址，变址寄存器的内容为 252，指令中给出的形式地址为 1200H。

(2) 操作数采用一次间接寻址，指令中给出的地址码为 1200H。

(3) 操作数采用寄存器间接寻址，指令中给出的寄存器编号为 8，8 号寄存器的内容为 1200H。

(1)

有效地址：252 + (1200)H = 00C0 H + 1200 H= 12C0 H，操作数未知

(2)

有效地址：M[1200 H] = 12FC H， 操作数：M[12FC H] = 38B8 H

(3)

有效地址：1200 H，操作数：M[1200 H] = 12FC H

6. 某计算机指令系统采用定长指令字格式，指令字长 16 位，每个操作数的地址码长 6 位。指令分二地址、单地址和零地址 3 类。若二地址指令有 k2 条，零地址指令有 k0 条，则单地址指令最多有多少条？

二地址剩余: $2^{16-2*6} - k_2 = 16 - k_2$

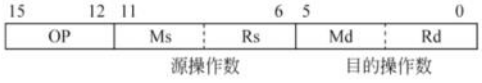
单地址剩余: $2^6 * (16 - k_2) - k_1 = (16 - k_2) * 64 - k_1$

零地址剩余: $2^6 * ((16 - k_2) * 64 - k_1) - k_0 \geq 0$

$\therefore k_1 \leq (16 - k_2) * 64 + \frac{k_0}{2^6}$

故而 单地址指令 最多有 $(16 - k_2) * 64 + \left\lfloor \frac{k_0}{64} \right\rfloor$ 向下取整

8. 某计算机字长为 16 位，主存地址空间大小为 128KB，按字编址。采用单字长定长指令格式，指令各字段定义如下：



转移指令采用相对寻址方式，相对位移量用补码表示，寻址方式定义如下表所示。

Ms/Md	寻址方式	助记符	含义
000B	寄存器直接	Rn	操作数=R[Rn]
001B	寄存器间接	(Rn)	操作数=M[R[Rn]]
010B	寄存器间接、自增	(Rn)+	操作数=M[R[Rn]]，R[Rn]←R[Rn]+1
011B	相对	D(Rn)	转移目标地址=PC+R[Rn]

注：M[x]表示存储器地址 x 中的内容，R[x]表示寄存器 x 中的内容。

请回答下列问题：

(1) 该指令系统最多可有多少条指令？最多有多少个通用寄存器？存储器地址寄存器(MAR)和存储器数据寄存器(MDR)至少各需要多少位？

(2) 转移指令的目标地址范围是多少？

(1)

最多的指令条数：$2^4 = 16$ 条；

最多有$2^3 = 8$ 个 通用寄存器

地址寄存器位数: $\lg\left(\frac{(128KB)}{\frac{16}{8}}\right) = 16$ 位

存储器数据寄存器位数 = 字长 = 16位

(2)

转移指令的目标地址范围 0000H ~ FFFFH （字长16位）

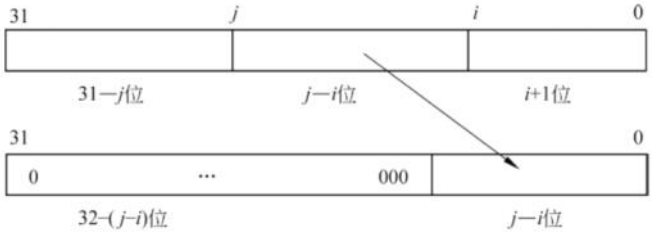
(3) 若操作码 0010B 表示加法操作(助记符为 add)，寄存器 R4 和 R5 的编号分别为 100B 和 101B，R4 的内容为 1234H，R5 的内容为 5678H，地址 1234H 中的内容为 5678H，地址 5678H 中的内容为 1234H，则汇编语句"add(R4)，(R5)+"(逗号前为第一源操作数，逗号后为第二源操作数和目的操作数)对应的机器码是什么(用十六进制表示)？该指令执行后，哪些寄存器和存储单元的内容会改变？改变后的内容是什么？

机器码：0010 001 100 010 101 = 0010 0011 0001 0101 = 2315 H

执行后, R4寄存器中的值不变，R5寄存器中的值变为5679 H

地址1234H中的值不变，地址5678H中的值变为 1234H + 5678H = 68AC H

9. 有些计算机提供了专门的指令，能从一个 32 位寄存器中抽取其中任意一个位串置于另一个寄存器的低位有效位上，并在高位补 0，如下图所示。MIPS 指令系统中没有这样的指令，请写出最短的一个 MIPS 指令序列来实现这个功能，要求 $i=5, j=22$，操作前后的寄存器分别为 $s0 和 $s2。



假设需要将$s0中[j,i)位移到$s2中

```
sll  $s2, $s0, 9
srr $s2, $s2, 16
```

10. 以下程序段是某个过程对应的指令序列。入口参数 int a 和 int b 分别置于 $a0 和 $a1 中，返回参数是该过程的结果，置于 $v0 中。要求为以下 MIPS 指令序列加注释，并简单说明该过程的功能。

```
          add    $t0, $zero, $zero
loop:     beq    $a1, $zero, finish
          add    $t0, $t0, $a0
          sub    $a1, $a1, 1
          j      loop
finish:   addi   $t0, $t0, 100
          add    $v0, $t0, $zero
```

// 计算$v0(返回值) = $a0(a)*$a1(b) + 100

// 结束后 $a1 = 0

```
          add    $t0, $zero, $zero    # 初始化$t0 = 0
loop:     beq    $a1,$zero, finish    # 判断$a1是否为0,是则结束循环，至finish
          add    $t0, $t0, $a0        #  $t0 += $a0(a)
          sub    $a1, $a1, 1          #  $a1(b) -= 1
          j      loop                 #  跳转到loop 继续循环
finish:   addi   $t0, $t0, 100        # $t0 += 100
          add    $v0, $t0, $zero      #  $v0(结果) = $t0
```

11. 下列指令序列用来对两个数组进行处理，并产生结果存放在 $v0 中，两个数组的基地址分别存放在 $a0 和 $a1 中，数组长度分别存放在 $a2 和 $a3 中。要求为以下 MIPS 指令序列加注释，并简单说明该过程的功能。假定每个数组有 2500 个字，其数组下标为 0～2499，该指令序列运行在一个时钟频率为 2GHz 的处理器上，add、addi 和 sll 指令的 CPI 为 1；lw 和 bne 指令的 CPI 为 2，则最坏情况下运行所需时间是多少秒？

```
          sll    $a2, $a2, 2          $a2 <<= 2  # 一个数占4个二进制位
          sll    $a3, $a3, 2          $a3 <<= 2  # 一个数占4个二进制位
          add    $v0, $zero, $zero    $v0 = 0    # 记录数组$a0和$a1中有相同元素对的个数
          add    $t0, $zero, $zero    $t0 = 0    # 记录遍历数组$a0的相对位置
outer:    add    $t4, $a0, $t0        $t4 = $a0 + $t0 # 遍历数组$a0的位置
```

```
        add     $v0, $zero, $zero   $v0 = 0     # 记录数组$a0和$a1中有相同元素对的个数
        add     $t0, $zero, $zero   $t0 = 0     # 记录遍历数组$a0的相对位置
outer:  add     $t4, $a0, $t0       $t4 = $a0 + $t0 # 遍历数组$a0的位置
        lw      $t4, 0($t4)         $t4 = M[$t4]   # 遍历数组$a0的值
        add     $t1, $zero, $zero   $t1 = 0  # 记录遍历数组$a1的相对位置
inner:  add     $t3, $a1, $t1       $t3 = $a1 + $t1 # 遍历数组$a1的位置
        lw      $t3, 0($t3)         $t3 = M[$t3]    # 遍历数组$a1的值
        bne     $t3, $t4, skip      if($t3 != $t4) goto skip # 两数不相等 跳过计数++
        addi    $v0, $v0, 1         $v0 += 1 # 两数相等,个数加一
skip:   addi    $t1, $t1, 4         $t1 += 4  # 下一个数
        bne     $t1, $a3, inner     if($t1 != $a3) goto inner  # 如果未到头继续循环
        addi    $t0, $t0, 4         $t0 += 4  # 下一个数
        bne     $t0, $a2, outer     if($t0 != $a2) goto outer # 如果未到头继续循环
```

等效果的c程序: 功能：统计两个数组的相同元素对个数

```
int res = 0;
for(int i = 0; i < arr1Size; ++i){
    for(int j = 0; j < arr2Size; ++j){
        if(arr1[i] != arr2[j]) continue;
        ++res;
    }
}
return res;
```

首先有 StartTimes = sll*2 + add*2 = 4

最坏的情况是每个数都一样，

从inner-bne inner 每次得2500次，InnerTimes = (add+lw+bne+addi+addi+bne)*2500 = 9*2500 = 22500 ;

从outer-bne outer 要2500次, outerTimes = (add + lw + add +InnerTime + addi + bne)*2500

= (7+9*2500)*2500 = 56267500;

TotalTimes = outerTimes + StartTimes = 56267504;

时间为 56267504/(2*10^9) = 0.0281 (s)

12. 用一条 MIPS 指令或最短的 MIPS 指令序列实现以下 C 语言语句：b=25|a。假定编译器将 a 和 b 分别分配到 $t0 和 $t1 中。如果把 25 换成 65536，即 b=65536|a，则用 MIPS 指令或指令序列如何实现？

b=25|a ⇒ ori $t1, $t0, 25

65536 = 1 0000 0000 0000 0000

b= 65536|a ⇒

```
lui $t1, 1
ori $t1, $t1, $t0
```

13. 以下程序段是某个过程对应的 MIPS 指令序列，其功能为复制一个存储块数据到另一个存储块中，存储块中每个数据的类型为 float，源数据块和目的数据块的首地址分别存放在 $a0 和 $a1 中，复制的数据个数存放在 $v0 中，作为返回参数返回给调用过程。假定在复制过程中遇到 0 就停止复制，最后一个 0 也需要复制，但不被计数。已知程序段中有多个错误，请找出它们并修改之。

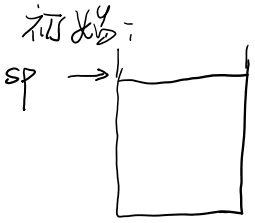
```
        addi    $v0, $zero, 0
loop:   lw      $v1, 0($a0)
        sw      $v1, 0($a1)
        addi    $a0, $a0, 4
        addi    $a1, $a1, 4
        beq     $v1, $zero, loop
```

```
    addi    $v0, $zero, 0
loop: lw    $v1, 0($a0)
```

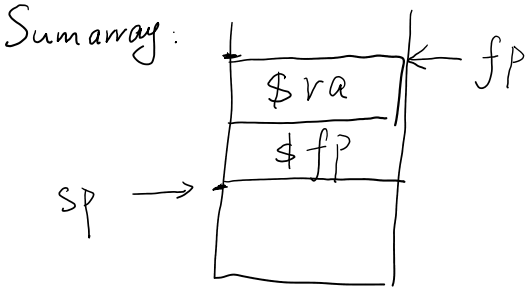
```
        sw      $v1, 0($a1)
        beq     $v1, $zero, exit
        addi    $v0, $v0, 1
        addi    $a0, $a0, 4
        addi    $a1, $a1, 4
        j       loop
exit:   ...
```

15. 以下 C 语言程序段中有两个函数 sum_array 和 compare，假定 sum_array 函数先被调用，全局变量 sum 分配在寄存器 $s0 中。要求按照 MIPS 过程调用协议写出每个函数对应的 MIPS 汇编语言程序，并画出每个函数调用前、后栈中的状态、帧指针和栈指针的位置。

```
int sum=0;

int sum_array(int array[], int num)
{
    int i;
    for(i=0; i<num; i++)
        if compare(num, i+1) sum+=array[i];
    return sum;
}

int compare(int a, int b)
{
    if (a>b)
        return 1;
    else
        return 0;
}
```




```
        and     $s0, $zero, $zero       # sum = 0
Sumarray:  addi $sp, $sp, -8            # callee saver
        sw      $ra,4($sp)              # callee saver
        sw      $fp,0($sp)              # callee saver
        addi    $fp,$sp,4               # callee saver
        add     $t0, $a0, $zero         # $t0 存放array的地址
        add     $t1, $a1, $zero         # $t1 为num的值
        and     $t2, $zero, $zero       # $t2 = i = 0;
Sumloop:   slt  $t3, $t2, $t1           # $t3 = 1 if $t2 < $t1, 判断i是否小于num
        beq     $t3, $zero, Sumexit     # 如果 i >= num, 跳出循环
        add     $a0, $t1, $zero         # 传参
        add     $a1, $t2, $zero         # 传参
        addi    $a1, $a1, 1             #传参
        jal     compare                   # 调用compare
        beq     $v0, $zero, Sumelse     # if compare return false, goto Sumelse
        lw      $t3, 0($t0)             # $t4 存放 array[i]
        add     $s0, $s0, $t4         # sum += array[i]
Sumelse:   addi $t2, $t2, 1           # i++
        addi    $t0, $t0, 4           # arr++
        j       Sumloop               # 继续循环
Sumexit:   lw   $ra, 4($sp)                 # 恢复
        lw      $fp, 0($sp)                 # 恢复
        addi    $sp, $sp, 8                 # 恢复
        jr      $ra                         # 返回

compare:   and  $v0, $zero, $zero         # 将返回值 初始化为0
        slt     $v0, $a1, $a0               # 如果 a1 < a0 => $v0=1;
```



Compare 为叶子过程 栈帧 为空
且 compare 中未使用临时寄存器
故 Sumarray 过程中不用保存.

```
jr      $ra
```

17. 假定编译器将 a 和 b 分别分配到 t0 和 t1 中，用一条 RV32I 指令或最短的 RV32I 指令序列实现以下 C 语言语句：b=31&a。如果把 31 换成 65535，即 b=65535&a，则用 RV32I 指令或指令序列如何实现？对比第 12 题 RV32I 与 MIPS 之间在立即数处理上有何不同？

b = 31 & a
31 = 0000 0001 1111

```
        andi    $t1, $t0, 31
```

b = 65536 & a
65535 = 1111 1111 1111 1111
0x 0000FFFF

```
        lui     $t1, 0x00010  也就是 lui     $t1, 16
        addi    $t1,$t1, -1
        and     $t1,$t1,$t0
```

mips lui为16位立即数放在高16位，低16位填0
riscv lui为20位立即数放在高20位，低12位填0

并都有立即数的符号拓展