241880334 闵振昌 第四章

3

由于每次CPU只能取1字节，并且这个指令占2字节，因此执行该转移指令之后PC变为202。要转移到PC为100的位置，偏移量应该是 $100-202=-102$。 将-102转化为8位补码即为第二部分的内容 因此是10011010

4

(1)

变址寻址，有效地址=形式地址+变址寄存器内容 变址器内容252=00FCH 因此有效地址是00FCH+1200H=12FCH

（2）

一次间接寻址，地址码指向的内存单元中存放的是有效地址，因此有效地址是12FCH

（3）

寄存器间接寻址，寄存器中存放的是操作数的有效地址，因此有效地址是1200H

6

对二地址指令：操作码占4位，最多有16种编码，$16-k_2$种可以用于扩展单地址指令 每一条二地址指令的扩展标志，可以扩展出$2^6=64$种单地址指令，但是还需要留一些给零地址指令作为扩展标志 每一条单地址指令的 “扩展标志” 可扩展出64条零地址指令，因此总共单地址指令的数量是 $(16-k_2)\times 64-\lfloor\frac{k_0}{64}\rfloor$

8

（1）

OP占4位，因此最多有$2^4=16$条指令 Rs 和 Rd 各占 3 位，因此最多有$2^3=8$个通用寄存器 主存地址空间是128KB，字长16位相当于2字节，因此主存的字数是 $128KB/2B=2^{16}$ 所以MAR至少需要16位，MDR需要存储一整个字，因此至少需要16位。

（2）

转移指令采取相对寻址方式，R[Rn] 是 16 位补码，所以目标地址的范围是 $0\sim 2^{16}-1$

（3）

对应的机器码：0010 001 100 010 101 十六进制是：0x2315 存储单元5678H的内容由1234H变为68ACH 寄存器R5的内容由5678H变为5679H

9

左移9位，把j推到最高位，放入s2，再右移14位，使位串位于低位，高位补0

```
sll $t0, $s0, 9
srl $s2, $t0, 14
```

10

```
add  $t0, $zero, $zero   # $t0 = 0
loop:
beq  $a1, $zero, finish  # 如果 $a1 == 0，跳转到 finish
add  $t0, $t0, $a0       # $t0 = $t0 + $a0
sub  $a1, $a1, 1         # a1自己减1
j    loop                # 跳回 loop
finish:
addi $t0, $t0, 100       # $t0 = $t0 + 100
add  $v0, $t0, $zero     # $v0 = $t0
```

功能分析： 计算a和b乘积，再加上100 $v_0 = a \times b + 100$

11

```
sll    $a2, $a2, 2        # 将数组长度转为字节长度
sll    $a3, $a3, 2
add    $v0, $zero, $zero  # 结果初始化为 0
add    $t0, $zero, $zero  # 外层循环下标 i 的字节偏移

outer:
    add    $t4, $a0, $t0  # 计算 &a[i]
    lw     $t4, 0($t4)
    add    $t1, $zero, $zero  # 内层循环下标 j 的字节偏移

inner:
    add    $t3, $a1, $t1  # 计算 &b[j]
    lw     $t3, 0($t3)
    bne    $t3, $t4, skip # 如果 b[j] != a[i]，跳转到 skip
    addi   $v0, $v0, 1    # 否则，$v0++

skip:
    addi   $t1, $t1, 4    # j++
```

```
    bne    $t1, $a3, inner # 如果 j != 数组 b 的字节长度，继续内层循环
    addi   $t0, $t0, 4    # i++
    bne    $t0, $a2, outer # 如果 i != 数组 a 的字节长度，继续外层循环
```

外层循环遍历数组 a，内层循环遍历数组 b。如果 a[i] == b[j]，则计数器 $v0 加 1。

所以功能是：计算两个数组的相同元素对 (i, j) 的数量

计算最坏情况的运行时间： 外层循环次数 n=2500，内层循环次数 m=2500，总迭代次数 = 6.25×10^6 最坏情况，每次内层循环有6条指令，总共9个周期，循环m次所以是22500周期 每次外层迭代有22507个周期，总共2500×22507 所以总共时间是总周期数 / 频率约等于28.13ms

12

25二进制是11001，表示为16位立即数，可以直接用ori指令

```
ori $t1, $t0, 25
```

但是65536太大了，没办法表示为16位立即数，分为两步

```
lui $at, 1           # $at = 1 << 16 = 0x00010000
or  $t1, $t0, $at    # $t1 = a | 0x00010000
```

13

修改后代码：

```
addi $v0, $zero, 0     # count = 0
loop:
lw   $v1, 0($a0)       # 读取数据
sw   $v1, 0($a1)       # 存储数据
beq  $v1, $zero, done  # 如果数据是 0，结束
addi $v0, $v0, 1       # 否则计数 +1
addi $a0, $a0, 4       # 源地址 +4
addi $a1, $a1, 4       # 目的地址 +4
j    loop              # 继续循环
done:
```

17

31:

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

65535:

```
lui   t2, 0x10
addi  t2, t2, -1
and   t1, t0, t2
```

MIPS：ORI 是零扩展 16 位立即数, lui 加载高 16 位，低 16 位用 ori 合并。

RISC-V：ORI / ANDI 是符号扩展 12 位立即数, lui 加载高 20 位，低 12 位用 addi 等合并，但 addi 是符号扩展，所以有时需要调整