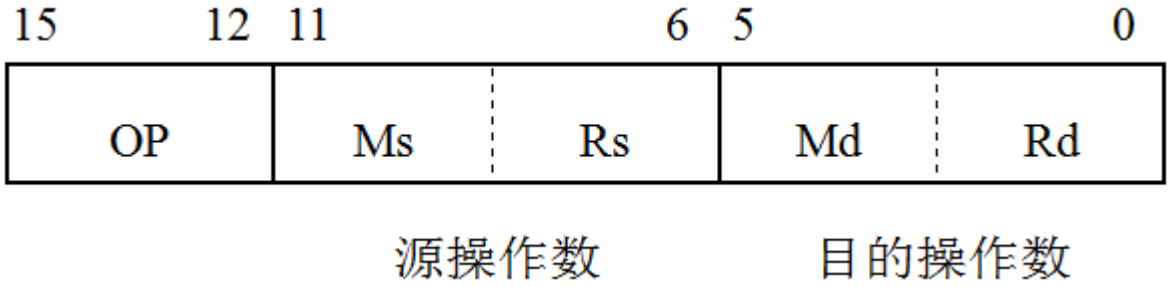
第4章 布置作业

- 补充题
- 课后3、4、6、7、8、9、10、11、12、13、14、15。

补充题

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



转移指令采用相对寻址方式，相对偏移量用补码表示。寻址方式定义如下：

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

注：(x) 表示存储器地址 x 或寄存器 x 的内容。

补充题 （续）

请回答下列问题：

（1）该指令系统最多可有多少条指令？该计算机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？

（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)+”（逗号前为源操作数，逗号后为目的操作数）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？

补充题　参考答案

（1）指令系统最多支持16条指令；支持8个通用寄存器；

MAR至少为16位；MDR至少为16位。

（2）转移指令的目标地址范围为0000H～FFFFH。

（3）汇编语句“add (R4), (R5)+”，对应的机器码为：0010 001 100 010 101B，用十六进制表示为2315H 。

“add (R4), (R5)+”指令执行后，R5和存储单元5678H的内容会改变。执行后，R5的内容为5679H。内存5678H单元的内容为68ACH。

P139 第3题 参考答案

执行到该转移指令时PC的内容为200，因此在CPU取指令过程中，取出第一字节的操作码后，PC的内容为201，取出第二字节的位移量后，PC的内容为202，因此计算转移目标地址时，PC已经是202了。

因为转移目标地址为100，所以此时位移量是100-202=-102,用补码表示为：-102= -1100110=9AH

P139 第4题 参考答案

(1)有效地址为 000CH+1200H=120CH。
(2)有效地址为 12FCH。
(3)有效地址为 1200H。

P139 第6题 参考答案

假设一地址指令最多x条，则$2^6*[(2^4-k2)*2^6-x]=k0$

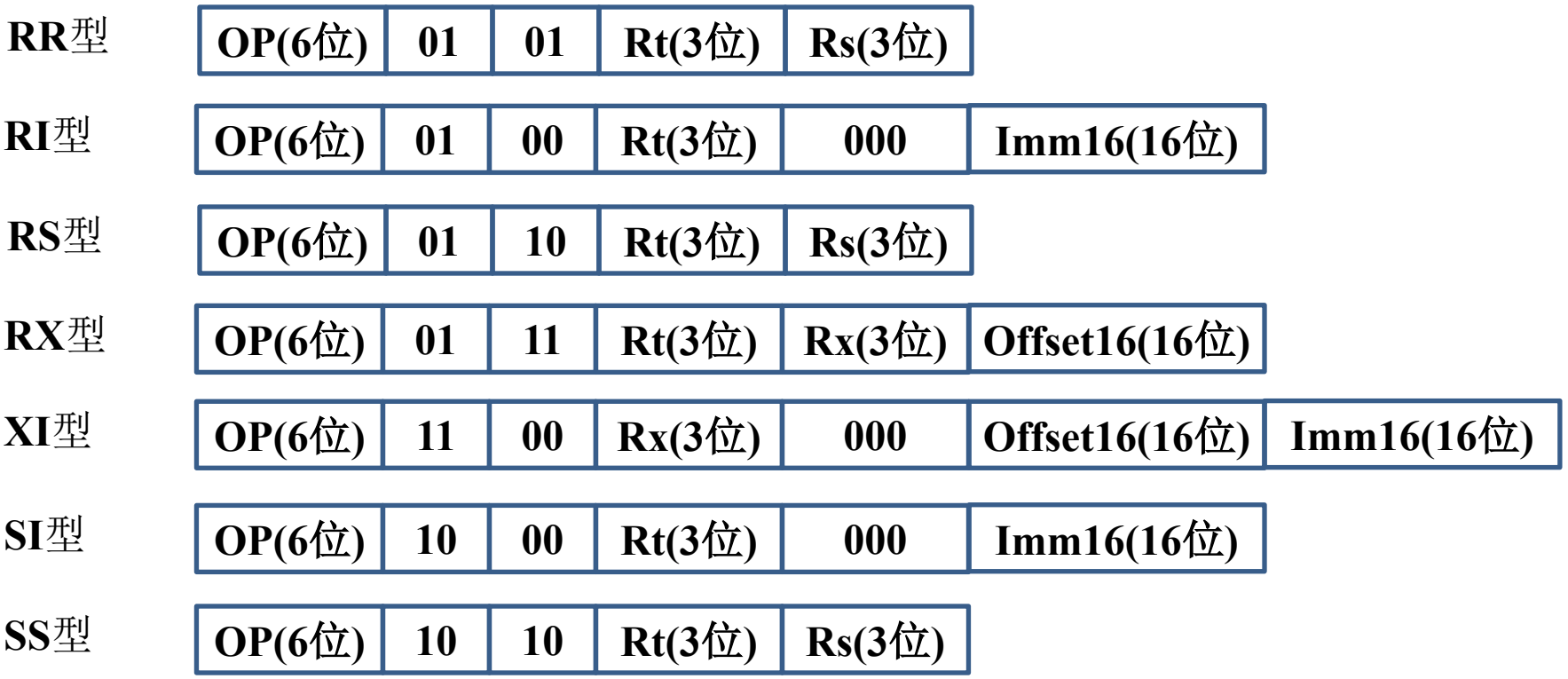
所以 $x=(16-k2)*2^6-k0/2^6$

P139 第7题 参考答案

因为至多有64种操作，所以**操作码字段**只需要6位；有8个通用寄存器，所以**寄存器编号**至少占3位；寻址方式有4种，所以**寻址方式位**至少占2位；存储器地址（直接地址）和立即数都是16位，任何通用寄存器都可作变址寄存器，所以指令中要明显指定变址寄存器，其编号占3位；指令总位数是16的倍数。此外，指令格式应尽量规整，指令长度应尽量短。

按照上述要求设计出的指令格式可以有很多种,例如下指令格式示例：

“寻址方式”字段：00-立即；01-寄存器直接；10-寄存器间接；11-变址。



P139 第7题 参考答案 （续）

存储器存取宽度为16位，每次从存储器取出16位。因此，读取16，32，48位指令分别需要1，2和3次存储器访问。各类指令的功能和访存次数分别说明如下：(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)

(1)RR型指令： 功能为R[Rt] ← R[Rt] op R[Rs]　　取指令时访存1次；
(2)RI型指令： 功能为R[Rt] ← R[Rt] op Imm16　　取指令时访存2次；
(3)RS型指令： 功能为R[Rt] ← R[Rt] op M[R[Rs]]
　　取指令和取第2个源操作数各访存1次，共访存2次；
(4)RX型指令： 功能为R[Rt] ← R[Rt] op M[R[Rx]+offset16]
　　取指令访存2次，取第2个源操作数访存1次，共访存3次；
(5)XI型指令： 功能为M[R[Rx]+offset16] ← M[R[Rx]+offset16] op Imm16
　　取指令访存3次，取第1个源操作数访存1次,写结果1次，共访存5次；
(6)SI型指令： 功能为M[R[Rt]] ← M[R[Rt]] op Imm16
　　取指令访存2次，取第1个源操作数访存1次,写结果1次，共访存4次；
(7)SS型指令： 功能为M[R[Rt]] ← M[R[Rt]] op M[R[Rs]]
　　取指令访存1次，取第1个源操作数各访存1次,取第2个源操作数和写结果各访存1次，共访存4次.

P140 第8题 参考答案

可以先左移9位，然后右移15位，MIPS指令序列为：

```
sll $s2,$s0,9      #将s0的内容左移9位，送s2
srl $s2,$s2,15     #将s2的内容右移15位，送s2
```

注意：不能使用算术右移指令sra

P140 第9题 参考答案

```
        add $t0,$zero,$zero     #将寄存器t0置0
loop:   beq  $a1, $zero, finish  #若a1的值等于0，则转finish处
        add $t0,$t0,$a0          #将ta0和a0的内容相加，送t0
        sub $a1,$a1,1            #将a1的值减1
        j  loop                  #无条件转到loop处
finish: addi $t0,$t0,100  #将t0的内容加100
        add $v0,$t0,$zero  #将t0的内容送v0
```

该过程的功能是计算a*b+100　(loop循环体里是b个a 相加)

P140 第10题 参考答案

```
        sll $a2,$a2,2     #将a2的内容左移2位，即乘4
        sll $a3,$a3,2     #将a3的内容左移2位，即乘4
        add $v0,$zero,$zero #将v0初始化为0
        add $t0,$zero,$zero #将t0初始化为0
outer:  add  $t4, $a0,$t0  #计算数组1当前元素的地址
        lw $t4,0($t4)       #数组1当前元素存放在$t4
        add $t1,$zero,$zero          #将t1初始化为0
inner:  add $t3,$a1,$t1       #计算数组2当前元素的地址
        lw $t3,0($t3)       #数组2当前元素存放在$t3
        bne $t3,$t4,skip   #当t3和t4的内容不相等，则转skip
        addi $v0,$v0,1    #将v0加1
skip:   addi $t1,$t1,4      #将t1加4
        bne $t1,$a3,inner   #数组2未处理完，继续转inner执行
        addi $t0,$t0,4    #将t0加4
        bne $t0,$a2,outer   #数组1未处理完，继续转outer执行
```

该过程的功能是统计两个数组中相同元素的个数，多次相等则重复计数。

P140 第10题 参考答案 （续）

该过程的功能是**统计两个数组中相同元素的个数，多次相等则重复计数。**

执行过程中最坏的情况是两个数组的所有元素都相等，这样指令“addi $v0,$v0,1 ”在每次循环中都被执行。因为add,addi和sll指令的CPI都为1，lw和bne的CPI为2，所以在最坏情况下所需的时钟周期总数{4+[4+(6+3)*2500+3]*2500}=56267506 时钟周期为1/2GHz=0.5ns 因此，执行该过程的总执行时间最多为56267506 *0.5ns=28133753ns ≈0.028s

P140 第11题 参考答案

实现b=25|a，指令：ori $t1,$t0,25 。

因为65536=1 0000 0000 0000 0000B 不能使用16位立即数表示，可以如下方式实现b=65536|a

```
lui  $t1,1          #将0001 0000H置于寄存器t1     LUI：立即数加载至高位
or  $t1,$t0,$t1     #将t0和t1的内容进行“或”运算，送t1
```

P140 第12题 参考答案

修改后的代码如下：

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

P141 第13题 参考答案

在MIPS指令系统中，分支指令beq是I-型指令，转移目标地址采用相对寻址方式，16位偏移量offset用补码表示，因此可以跳转到当前指令前或指令后。其转移目标地址的计算公式为：PC+4+offset*4，因此当offset为0000H~7FFFH，向后跳，相对于本条指令，向后正跳1~2^{15}条指令，其跳转地址范围：4~4+(2^{15}-1)*4=2^{17}。当为8000H~FFFFH，向前跳，相对于本条指令，向前负跳1~2^{15} -1条指令，其跳转地址范围：-4*(2^{15}-1)~ -4 。

当上述指令序列中here和there表示的地址相差超过上述给定范围时，beq指令会发生汇编错误，可以改为以下指令序列：　（无条件转移指令j的跳转范围足够大）

```
here:  bne  $s0,$s2,skip
       j  there
skip:  …
       …
there:  addi  $s1,$s0,4
```

P141 第14题 参考答案

（1）过程 sum_array: 本过程使用的array是入口参数，而不是局部变量，无须在该过程中的栈帧中给它分配空间，array数组的首地址作为入口参数被传递给过程sum_array(假定在$a0中)。

此外，还有另一个入口参数为num (假定在$a1中)，最后一个返回参数($s0)。

因为该过程又调用了compare过程，因此它不是叶子过程。所以在其栈帧中除了保留所用的保存寄存器外，还必须保留返回地址$ra，以免在调用过程compare时被覆盖。

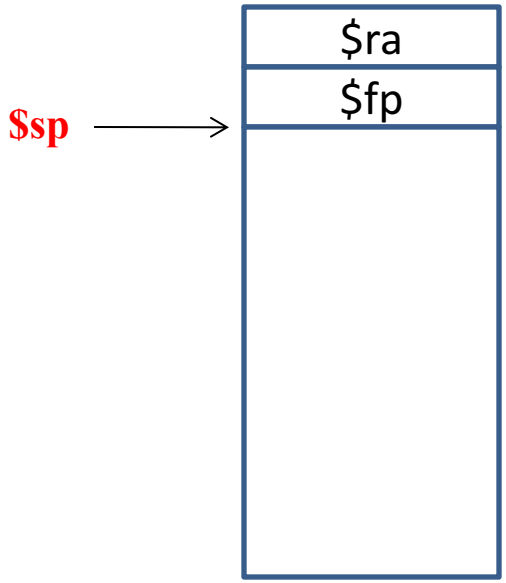
是否保存$fp要看具体情况，如果确保后面一直都用不到$fp，则可以不保存它，但编译器往往无法预测这种情况，所以通常采用统一的做法，即总是保存$fp的值。因而，该过程的栈帧中要保存的信息有返回地址$ra和桢指针$fp，其栈帧空间大小至少为4*2=8B。

汇编代码如下：

P141 第14题 参考答案 (续)

(1)过程sum_array汇编代码如下：

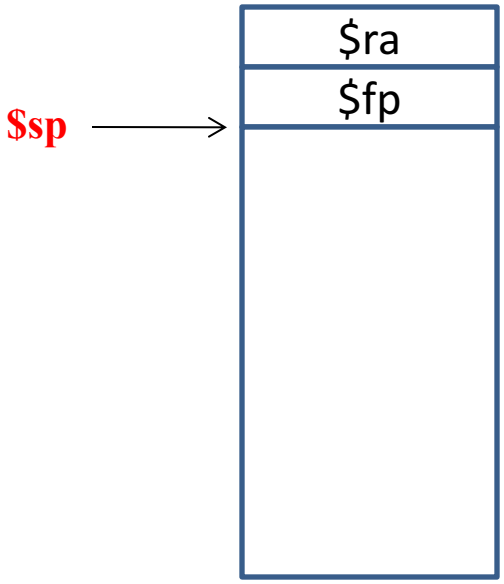
```
            add $s0,$zero,$zero      #sum=0
sum_array:  addi $sp,$sp,-8          #生成栈帧
            sw $ra, 4($sp)           #保存ra到栈
            sw $fp, 0($sp)           #保存fp到栈
            addi  $fp, $sp, 4        #设置fp
            add $t2, $zero, $a0      #数组起始地址
            add $t0, $zero, $a1      # t0=sum
            add $t3, $zero, $zero    #  i=0
loop:       slt $t1,$t3,$t0          #  if (i<num)  t1=1  else  t1=0
            beq  $t1,$zero, exit1    #  if (t1=0)  jump to exit1
            add $a0, $zero, $t0      # a0=num
            add $a1, $zero, $t3      # a1=i
            addi $a1, $a1, 1         # a1=i+1
            jal  compare             #call compare
            beq  $v0, $zero, else    # if (v0=0)  jump to else
            sll $t1,$t3, 2           #i=i*4
            add $t1,$t2,$t1          # t1=array[i]
             lw $t4, 0($t1)          # load array[i]
             add $s0, $s0, $t4       # sum+=array[i]
  else:     addi $t3,$t3,1          # i=i+1
             j loop
  exit1:    lw  $ra, 4($sp)        #重置ra
            lw  $fp, 0($sp)        #重置fp
            addi  $sp, $sp, 8       #释放栈帧
            jr $ra                  #返回调用者
```



P141 第14题 参考答案 (续)

（2）过程compare: 入口参数为a和b，分别在$a0和$a1中，有一个返回参数，没有局部变量，是叶子过程，且过程体中没有用到任何保存寄存器，所以栈帧中不需要保留任何信息。

```
compare:    add $v0,$zero,$zero      # 返回值设定为0
            beq  $a0,$a1, exit2      # if ($a0=$a1) jump to exit2
            slt $t1, $a0, $a1        # if ($a0<$a1) $t1=a else $t1=0
            bne $t1, $zero, exit2    # if ($a0<$a1) jump to exit2
            ori $v0,$zero, 1         #返回值设定为1
  exit2:    jr $ra
```



P141 第15题 参考答案

过程fact有一个输入参数，按MIPS过程调用协议，n应在$a0中，返回参数应在$v0中。过程内没有局部变量，故无须在其栈帧中保留局部变量所用空间；需递归调用fact过程，所以必须在其栈帧中保留返回地址$ra。过程体内全部使用临时寄存器$t0-$t9,因而无须在其栈帧中保存通用寄存器。因为是递归调用，所以需在栈帧中保留输入参数，所以该过程的栈帧中要保存的信息有返回地址$ra，帧指针$fp和输入参数$a0，栈帧空间大小至少为4*3=12B。

```
fact:   addi $sp, $sp, -12      #产生本过程栈帧
        sw  $ra, 8($sp)         #保存ra
        sw $fp, 4($sp)          #保存fp
        addi $fp, $sp, 0        #设置fp
        sw $a0, 0($fp)          #保存a0 到栈中
        slti $t0,$a0, 1         # if(n<1)  $t0=1  else  $t0=0
        bne $t0,$zero, exit1    # if(n<1)  goto exit1
        addi $a0, $a0, -1       # n=n-1
        jal  fact               #  call fact
        lw  $t1, 0($fp)         #  恢复n
        mul $v0, $t1,$v0        # $v0=n*fact(n-1)
        j exit
exit1:addi $v0, $zero, 1        # $v0=1
 exit:  lw $ra, 8($sp)          # 恢复ra
        lw $fp , 4($sp)         # 恢复fp
        addi $sp, $sp, 12       # 释放栈帧
        jr  $ra                 # 返回调用者
```