第 2 章 课后习题

1、尾数用补码、小数表示，阶码用移码、整数表示，尾数字长 p=6（不包括符号位），阶码字长 q=6（不包括符号位），为数基值 r_m=16，阶码基值 r_e=2。对于规格化浮点数，用十进制表达式写出如下数据（对于前 11 项，还要写出 16 进值编码）。

（1）最大尾数　　（8）最小正数

（2）最小正尾数　　（9）最大负数

（3）最小尾数　　（10）最小负数

（4）最大负尾数　　（11）浮点零

（5）最大阶码　　（12）表数精度

（6）最小阶码　　（13）表数效率

（7）最大正数　　（14）能表示的规格化浮点数个数

2．一台计算机系统要求浮点数的精度不低于 $10^{-7.2}$，表数范围正数不小于 10^{38}，且正、负数对称。尾数用原码、纯小数表示，阶码用移码、整数表示。

(1) 设计这种浮点数的格式

(2) 计算（1）所设计浮点数格式实际上能够表示的最大正数、最大负数、表数精度和表数效率。

3．某处理机要求浮点数在正数区的积累误差不大于 2^{-p-1} ，其中，p 是浮点数的尾数长度。

(1) 选择合适的舍入方法。

(2) 确定警戒位位数。

(3) 计算在正数区的误差范围。

4．假设有 A 和 B 两种不同类型的处理机，A 处理机中的数据不带标志符，其指令字长和数据字长均为 32 位。B 处理机的数据带有标志符，每个数据的字长增加至 36 位，其中有 4 位是标志符，它的指令数由最多 256 条减少到不到 64 条。如果每执行一条指令平均要访问两个操作数，每个存放在存储器中的操作数平均要被访问 8 次。对于一个由 1000 条指令组成的程序，分别计算这个程序在 A 处理机和 B 处理机中所占用的存储空间大小（包括指令和数据），从中得到什么启发？

5．一台模型机共有 7 条指令，各指令的使用频率分别为 35%，25%，20%，10%，5%，3%和 2%，有 8 个通用数据寄存器，2 个变址寄存器。

(1) 要求操作码的平均长度最短，请设计操作码的编码，并计算所设计操作码的平均长度。

(2) 设计 8 字长的寄存器-寄存器型指令 3 条，16 位字长的寄存器-存储器型变址寻址方式指令 4 条，变址范围不小于±127。请设计指令格式，并给出各字段的长度和操作码的编码。

6．某处理机的指令字长为 16 位，有双地址指令、单地址指令和零地址指令 3 类，并假设每个地址字

段的长度均为 6 位。

(1) 如果双地址指令有 15 条，单地址指令和零地址指令的条数基本相同，问单地址指令和零地址指令各有多少条？并且为这 3 类指令分配操作码。

(2) 如果要求 3 类指令的比例大致为 1：9：9，问双地址指令、单地址指令和零地址指令各有多少条？并且为这 3 类指令分配操作码。

7. 别用变址寻址方式和间接寻址方式编写一个程序，求 C=A+B，其中，A 与 B 都是由 n 个元素组成的一维数组。比较两个程序，并回答下列问题：

(1) 从程序的复杂程度看，哪一种寻址方式更好？

(2) 从硬件实现的代价看，哪一种寻址方式比较容易实现？

(3) 从对向量运算的支持看，哪一种寻址方式更好？

8. 假设 X 处理机的数据不带标志符，其指令字长和数据字长均为 32 位。Y 处理机的数据带有标志符，每个数据的字长增加至 35 位，其中有 3 位是标志符，其指令字长由 32 位减少至 30 位。并假设一条指令平均访问两个操作数，每个操作数平均被访问 R 次。现有一个程序，它的指令条数为 I，分别计算在这两种不同类型的处理机中程序所占用的存储空间，并加以比较。

9. 一种浮点数表示方式的精度不低于 10^{-19}，能表示的最大正数不小于 10^{4000}，而且正负数对称。尾数用原码、小数表示，阶码用移码、整数表示，尾数和阶码的基值都是 2。

(1)设计这种浮点数的格式，给出各字段的名称和长度。

(2)计算(1)所设计的浮点数格式能够表示的最大正数、最大负数和表示数的精度。

(3)如果在运算器中没有设置硬件警戒位，则这种浮点数可能采用了哪一种舍入方法？给出这种舍入方法的舍入规则，在正数区的误差范围和积累误差。

10. 有研究人员指出，如果在采用通用寄存器结构的计算机里加入寄存器-存储器寻址方式可能提高计算机效率。做法是用：

```
ADD R2, 0(Rb)
```

代替指令序列

```
LOAD R1, 0(Rb)
ADD R2, R2, R1
```

假定使用新的指令能使时钟周期增加 10%，并且假定只对时钟产生影响，而不影响 CPI 那么：

(1) 采用新的指令，要达到与原来同样的性能需要去掉的 load 操作所占的百分比？(假定 load 指令占总指令的 22.8%)

(2) 举出一种多指令序列，该序列不能使用上述的寄存器-存储器寻址方式。即使得 load R1 后面紧接着执行对 R1 的操作（该操作可以是任意某一操作码），但这一指令序列不能被一条指令（假定存在

这条指令）代替。

11. 试比较下面 4 种不同类型的指令结构的存储效率：

(1) 累加型：所有的操作都在单个寄存器和单个内存地址之间进行

(2) 存储器-存储器型：每个指令的 3 个操作数都在内存中进行

(3) 堆栈型：所有的操作都在栈顶进行。只有 push 和 pop 操作会访问内存，其它的指令执行时都会删除栈中的操作数，然后写入执行结果。

(4) 通用寄存器型：所有的操作都在寄存器中进行。这些寄存器-寄存器指令中的每个指令都包含 3 个操作数。通用寄存器一共有 16 个，寄存器标志符占 4 位长。

为比较存储效率，我们对以上 4 种指令集作了如下约定：

操作码占一个字节（8 位）

内存地址占 2 个字节（16 位）

操作数占 4 字节（32 位）

所有指令的长度都以整数个字节计算

另外，还假定访问内存不使用其它的优化措施，变量 A、B、C 和 D 的初值都已经放在内存中。

针对以上 4 种不同的指令系统，回答下列问题：

(1) 分别用汇编指令写出下面 3 个赋值语句：

A = B + C;

B = A + C;

D = A – B;

(2) 分别计算所执行指令的字节数和转移内存数据的字节数，并指出如果根据代码的大小来计算的话，哪种结构的效率是最高的？如果按需要的总内存带宽（代码+数据）来计算，又是哪种结构的效率最高？

12. 考虑为 DLX 结构的计算机增加一个新的寻址模式。即使得地址模式增加两个寄存器和一个 11 位长的带符号的偏移量来得到有效地址。这样，编译器就会用新的寻址模式来代替

ADD R1, R1, R2

LW Rd, 0(R1) (或是 Store 指令)

如果已知在 DLX 结构的计算机上对测得一些程序的 load 和 store 指令分别平均占 26%和 9%，在此基础上，计算：

(1) 假定 10%的 load 和 store 指令可以用新的寻址模式代替，那么采用新的寻址模式后的指令计数与采用前之比为多少？

(2) 如果新的寻址模式使得时钟周期增长 5%，那么采用了新的寻址模式的机器和未采用新的寻址模式的机器相比，哪种机器会更快一些，快多少？

1、**解答：**

在尾数采用补码、小数表示且 p=6，阶码采用移码、整数表示且 q=6，尾数基 r_m 为 16，阶码基 r_e 为 2 的情况下：

(1) 最大尾数为：$1-r_m^{-p}=1-16^{-6}$，0.FFFFFF

(2) 最小正尾数为：$1/r_m=1/16$，0.100000

(3) 最小尾数为： -1， 1.000000

(4) 最大负尾数为：$-(r_m^{-1}+r_m^{-p})=(16^{-1}+16^{-6})$，1.EFFFFF

(5) 最大阶码为：$r_e^q-1=2^6-1=63$，7F，包括符号位共 7 个 1

(6) 最小阶码为：$-r_e^q=-2^6=-64$，00，包括符号位共 7 个 0

(7) 最大正数为：$(1-16^{-6})16^{63}$，7FFFFFF

(8) 最小正数为：16^{-65}，00100000

(9) 最大负数为：$-(16^{-1}+16^{-6})\ 16^{-64}$，80EFFFF

(10) 最小负数为：-16^{63}，FF000000

(11) 浮点零为：00000000

(12) 表数精度为：$16^{-5}/2=2^{-21}$

(13) 表数效率为：15/16＝93.75％

(14) 能表示的规格化浮点数个数为：$2\times15\times16^5\times2^7+1$

2、解答:

(1)取尾数和阶码的基都为 2，即：$r_m=2$ 且 $r_e=2$

根据表示数精度的要求：$2^{-p}<10^{-7.2}$　　$p>\dfrac{7.2}{\log 2}=23.9$

于是可以取 p=24；

根据表示数范围的要求：$(1-2^{-p})2^{2^q-1}>10^{38}$

即 $2^{2^q-1}>1.00000006\times10^{38}$

$$2^q>\frac{38+\log 1.00000006}{\log 2}+1$$

$$q>\frac{\log(127.33)}{\log 2}=6.99$$

因此可以取 q＝7

数据格式可以表示如下（尾数采用隐藏位）：

1 位	1 位	7 位	23 位
符号	阶符	阶码	尾数

(2)能够表示的最大正数：（$1-2^{-24}$）2^{127}，

能够表示的最大负数：-2^{-129}，

表示数的精度：2^{-24}，

表数效率：100%。

3、解答：

(1) 舍入方法：下舍上入法、查表法

(2) 警戒位位数：2 位

(3) 正数区的误差范围：$-2^{-p-1}(1-2^{-g+1}) \sim 2^{-p-1}$

4、解答:

我们可以计算出数据的大致数量：

1000 条指令访问的数据总数为 1000*2=2000 个；

每个数据平均访问 8 次，所以，不同的数据个数为：2000÷8=250 个

对于 A 处理机，所用的存储空间的大小为：

$Mem\_size = Mem_{instruction}+Mem_{data}$ = 1000×32+250×32=40000*bit*

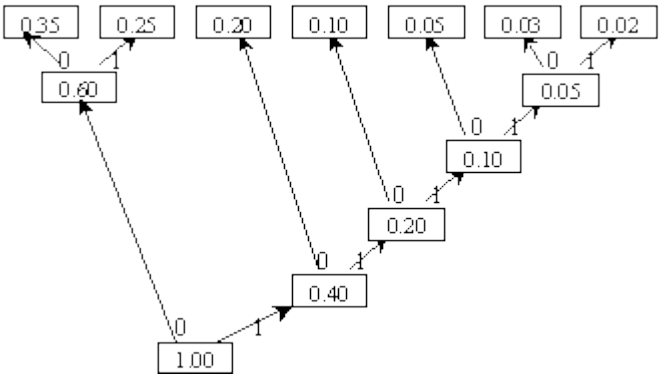
对于 B 处理机，指令字长由 32 位变为了 30 位（条数由 256 减少到 64），这样，所用的存储空间的大小为：

$Mem\_size = Mem_{instruction}+Mem_{data}$ = 1000×32+250×36=39000*bit*

由此我们可以看出，由于数据的平均访问次数要大于指令，所以，通过改进数据的格式来减少指令的长度，可以减少总的存储空间大小。

5、解答:

(1) 要使得到的操作码长度最短，应采用 Huffman 编码，构造 Huffman 树如下：



由此可以得到 7 条指令的编码分别如下：

指令号	出现的频率	编码
1	35%	00
2	25%	01
3	20%	10
4	10%	110
5	5%	1110
6	3%	11110
7	2%	11111

这样，采用 Huffman 编码法得到的操作码的平均长度为：

H = 2×(0.35+0.25+0.20) + 3×0.10 + 4 ×0.05 + 5×(0.03 + 0.02)

= 1.6+0.3+0.2+0.25

=2.35

(2) 设计 8 位字长的寄存器-寄存器型变址寻址方式指令如下：

因为只有 8 个通用寄存器，所以寄存器地址需 3 位，操作码只有两位，设计格式如下：

2 位	3 位	3 位
操作码 OP	源寄存器 R1	目的寄存器 R2

3 条指令的操作码分别为 00，01，10

设计 16 位字长的寄存器-存储器型变址寻址方式指令如下：

4 位	3 位	1 位	8 位
操作码 OP	通用寄存器	变址寄存器	偏移地址

4 条指令的操作码分别为 1100， 1101，1110，1111

6、解答：

(1) 首先，可以根据指令地址的数量来决定各种指令在指令空间上的分布：

如果按照从小到大的顺序分配操作码，并且按照指令数值从小到大的顺序，分别为双地址指令、单地址指令和零地址指令。

其次可以根据指令的条数来大致的估计操作码的长度：

双指令 15 条，需要 4 位指令来区分，剩下的 12 位指令平均分给单地址和零地址指令，每种指令可以用 6 位指令来区分，这样，各指令的条数为：

双地址指令 15 条，地址码：0000~1110；

单地址指令 26-1=63 条，地址码：1111 000000~1111 111110；

零地址指令 64 条，地址码：1111 111111 000000~1111 111111 111111。

(2) 与上面的分析相同，可以得出答案：

双地址指令 14 条，地址码：0000~1101；

单地址指令 2^6*2-2 = 126 条，1110 000000~1110 111110，1111 000000~1111 111110；

零地址指令 128 条 1110 111111.000000~1110 111111.111111，1111

7、解答：

(1) 变址寻址方式

(2) 间接寻址方式

(3) 变址寻址方式

8、解答：

X 处理机程序占用的存储空间总和为：$B_x = 32I + \frac{2\times 32I}{R}$，

Y 处理机程序占用的存储空间总和为：$B_y = 30I + \frac{2\times 35I}{R}$，

Y 处理机与 X 处理机的程序占用存储空间的比值：$\frac{B_y}{B_x} = \frac{30I + \frac{2\times 35I}{R}}{32I + \frac{2\times 32I}{R}} = \frac{15R + 35}{16R + 32}$

当 R＞3 时，有 $\frac{B_y}{B_x} < 1$，即对于同样的程序，在 Y 处理机中所占用的存储空间比在 X 处理机中所占用的存储空间要小。在实际应用中经常是 R＞10，所以带标志符的处理机所占用的存储空间通常要小。

9、解答：

(1)根据表示数精度的要求：$2^{-p} < 10^{-19} \qquad p > \frac{19}{\log 2} = 63.12$

根据表示数范围的要求：$2^{2^{q-1}} > 10^{4000}$

$2^q > \frac{4000}{\log 2} + 1$

$q > \frac{\log(4000/\log 2 + 1)}{\log 2} = 13.70$

取 p＝64, q＝14

1 位	1 位	14 位	64 位
符号	阶符	阶码	尾数

(2)能够表示的最大正数：（$1-2^{-64}$）2^{16383}，

能够表示的最大负数：-2^{-16385}，

表示数的精度：2^{-64}。

(3)采用的舍入方法为恒置法，

舍入规则：在规格化之后，尾数的最低位置为 1，

在正数区的误差范围：$-2^{-64}(1-2^{-g})$～$+2^{-64}$，

在正数区的积累误差：$+2^{-64}$。

10、解答：

(1) 计算 CPU 时间为：$T_{CPU} = I_C \times CPI \times T_C$

对原来的指令: $T_{CPU\,old} = I_{C\,old} \times CPI_{old} \times T_{C\,old}$ **(1)**

$$T_{CPU_{new}} = I_{C_{new}} \times CPI_{new} \times T_{C_{new}}$$

$$= (I_{C_{old}} - R) \times CPI_{old} \times (1.10 \times T_{C_{old}}) \qquad \textbf{(2)}$$

对修改后的指令序列:

在等式（2）中，$CPI_{new} = CPI_{old}$，$T_{C_{new}} = (1.10 \times T_{C_{old}})$，R 为新的指令设计方案中比原来的方案中减少的指令数。

要使去掉一些 load 操作，使得修改指令后的性能和原来的性能相同，必须满足：

$$I_{C_{old}} \times CPI_{old} \times T_{C_{old}} = (I_{C_{old}} - R) \times CPI_{old} \times (1.10 \times T_{C_{old}})$$

即 $$\frac{I_{C_{old}} - R}{I_{C_{old}}} = 0.91$$

又已知 load 指令占总指令的 22.8%

$$\therefore \frac{1-91\%}{22.8\%} = 39.5\%$$

即要达到原来的性能，39.5%的 load 指令必须去掉。

((2) 给出下面的两条指令：

ld r1, 0(r1)

add r1, r1, r1

在上面的指令中，我们将题目中的寄存器 r2 和 rb 用寄存器 r1 来代替。如果假定 r1 的值为 47，指令执行前内存中地址为 47 的位置存放了整数 4，那么上面指令序列执行后 r1 存放的值将变为 8。

然而，如果我们使用了寄存器-存储器模式后，上面的指令序列成为：

add r1, 0(r1)

假设寄存器和内存初值相同的话，指令执行结束后，寄存器的值将变为 51（r1+MEM[0+r1]即 47+4）。所以，在这种情况下，不能做指令替换。

11、解答：

一台采用累加型指令集结构计算机实现的程序段如下：

```
Start:  loada   B       ; accumulator←B
        adda    C       ; accumulator←B+C
        storea  A       ; store B+C in A
        add     C       ; accumulator←A+C
        storea  B       ; store A+C in B
        negatea         ; negate accumulator
        adda    A       ; accumulator← -B+A
        storea  D       ; store A-B in D
```

在上面的代码中，loada，storea 和 adda 每个都是 24 位长（操作码 8 位，操作数地址 16 位），negatea 指令只需 8 位，所以整个代码一共需要 22 个字节。对存取数据操作来说，每条指令访问操作数的值需要访问 4 字节数据，所以，整个程序需要在内存和 CPU 之间转移 28 字节的数据。

一台采用存储器-存储器型指令集结构计算机实现的程序段如下：

```
Start:  add     A, B, C     ; A=B+C
        add     B, A, C     ; B=A+C
        sub     D, A, B     ; D=A-B
```

在上面的代码中，每个指令为 56 位长（操作码 8 位，3 个操作数 48 位），这样程序需要 21 个字节。对存取数据操作来说，每条指令执行 3 个各访问 4 字节数据的操作，这样，整个程序在内存和 CPU 之间共转移 36 字节的数据。

一台采用堆栈型指令集结构计算机实现的程序段如下：

```
Stack:  push    B       ; push B onto stack
        push    C       ; push C onto stack
        add             ; top←B+C
        pop     A       ; A=B+C
        push    A       ; push A onto stack
        push    C       ; push C onto stack
        add             ; top←A+C
        pop     B       ; B=A+C
        push    A       ; push A onto stack
        push    B       ; push B onto stack
        sub             ; top←A-B
        pop     A       ; D=A-B
```

上面的代码中，每条 load 和 store 指令都为 28 位长（操作码 8 位，地址 16 位，寄存器 4 位），每条 add 和 sub 指令都为 20 位长（操作码 8 位，3 个寄存器 12 位），所以整个代码一共需要 29 个字节存储。对存取数据操作来说，每条 load 和 store 指令访问操作数的值需要访问 4 字节数据，所以整个程序在内存和 CPU 之间共转移 20 字节的数据。

根据上面的讨论，可以得到如下表格：

结构类型	指令带宽	数据带宽	总计
累加型	22	28	50
存储器-存储器型	21	36	57
堆栈型	30	36	66
通用寄存器型	29	20	49

由上图我们可以看出采用不同的结构指令带宽和数据带宽之间的关系。其中采用存储器-存储器型指令集结构的指令带宽较小，但数据带宽较大，而采用通用寄存器指令集结构正好相反。

根据写出的代码，从总的指令和数据带宽的角度来看，采用通用寄存器指令集结构的计算机是效率最高的，其次是采用累加型指令集结构的计算机。

12、解答：

(1) 设 IC 为指令条数，$I_{Cclassic}$ 和 $I_{C\,new}$ 分别表示未采用新的寻址模式和采用新的寻址模式所执行的指令数。

$$\therefore \quad \frac{I_{Cnew}}{I_{Cclassic}} = \frac{I_{Cclassic} - I_{Cclassic}[(10\% \times 26\%) + (10\% \times 9\%)]}{I_{Cclassic}}$$

$$= \frac{0.965\, I_{Cclassic}}{I_{Cclassic}} = 0.965$$

(2) 为比较哪一种机器更快一些，需要考虑 CPU 时间。计算 CPU 的时间的公式为： $T_{CPU} = I_c \times CPI \times T_c$

由（1）可知，采用新寻址模式的机器执行的指令数占未采用新的寻址模式机器执行指令数的 96.5%，又因为要支持新的寻址模式，时钟周期增长 5%

∴ 加速比为

$$\frac{I_{Classic} \times CPI_{classic} \times T_{Classic}}{I_{New} \times CPI_{new} \times T_{New}}$$

$$= \frac{I_{Classic} \times CPI_{classic} \times T_{Classic}}{0.965 I_{Classic} \times CPI_{classic} \times 1.050 T_{Classic}}$$

$$= 0.987$$

由此我们可以看出，尽管未采用新寻址模式的机器比采用新寻址模式的机器执行的指令数多，但由于时钟周期的影响，采用了新的寻址模式的机器比未采用的机器实际上慢了 1-98.7%=1.3%。