南京大学
NANJING UNIVERSITY

1-4章习题课

孟晴开
智能软件与工程学院
南雍楼东区228
qkmeng@nju.edu.cn，https://mengqingkai.github.io/

习题课

- **第1章 计算机系统概述**
- 第2章 数据的机器级表示
- 第3章 运算方法和运算部件
- 第4章 指令系统

(感谢计算机学院袁春风老师提供的许多习题答案！)

2. 简单回答下列问题。

(1) 冯·诺依曼计算机由哪几部分组成？各部分的功能是什么？采用什么工作方式？

(2) 摩尔定律的主要内容是什么？

(3) 计算机系统的层次结构如何划分？计算机系统的用户可分为哪几类？每类用户工作在哪个层次？

(4) 程序的 CPI 与哪些因素有关？

(5) 为什么说性能指标 MIPS 不能很好地反映计算机的性能？

参考答案：

（1） 由运算器、 控制器、 存储器、 输入设备和输出设备组成； 运算器用于算术运算和逻辑运算； 控制器用于控制指令的自动执行； 存储器用于存放数据和指令； 输入输出设备用于给操作人员使用计算机；采用 “存储程序” 的工作方式。

（2） 摩尔定律：对于半导体集成电路， 每18个月， 集成度将翻一番， 速度将提高一倍，而其价格将降低一半。

（3）计算机系统的层次化结构：

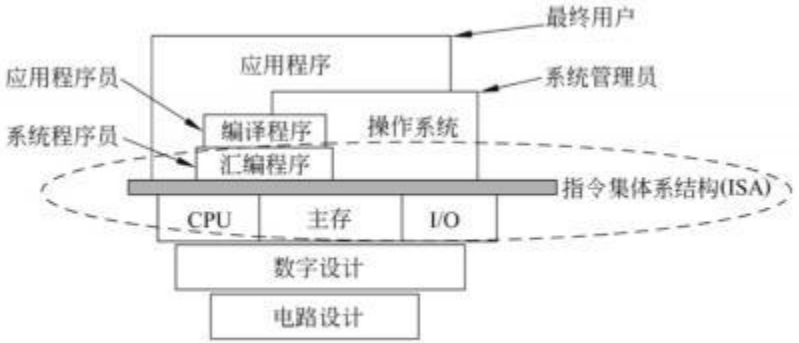


图 1.7　计算机系统的层次化结构

- 最终用户：使用应用程序完成特定任务的计算机用户，工作在最上面的应用程序层。
- 系统管理员：利用操作系统、数据库管理系统等软件对系统进行配置、管理和维护的操作人员，工作在操作系统层。
- 应用程序员：使用高级编程语言编制应用软件的程序员，工作在语言处理系统层。
- 系统程序员：设计和开发操作系统、编译器、数据库管理程序等系统软件的程序员，工作在指令集体系结构层。

2. 简单回答下列问题。

(1) 冯·诺依曼计算机由哪几部分组成？各部分的功能是什么？采用什么工作方式？

(2) 摩尔定律的主要内容是什么？

(3) 计算机系统的层次结构如何划分？计算机系统的用户可分为哪几类？每类用户工作在哪个层次？

(4) 程序的 CPI 与哪些因素有关？

(5) 为什么说性能指标 MIPS 不能很好地反映计算机的性能？

（4）程序的CPI是所有指令的平均时钟周期数，与每种指令占所有指令的比例以及每种指令所需的时钟周期数有关。

（5）由于不同机器的指令集、指令功能、CPI、时钟周期不同，且制造商经常将高于实际性能的峰值MIPS直接当作MIPS。

第1章 计算机系统概述

8. 假设机器 M 的时钟频率为 4GHz，程序 P 在 M 上的指令条数为 8×10^9，其 CPI 为 1.25，则 P 在 M 上的执行时间是多少？若在机器 M 上从程序 P 开始启动到执行结束所需的时间是 4s，则 P 占用的 CPU 时间的百分比是多少？

参考答案：

程序 P 在 M 上的执行时间为：$1.25\times8\times10^9\times1/4G = 2.5$ s，从启动 P 执行开始到执行结束的总时间为 4 秒，其中 2.5 秒是 P 在 CPU 上真正的执行时间，其他时间可能执行操作系统程序或其他用户程序。程序 P 占用的 CPU 时间的百分比为：2.5/4 = 62.5%。

9. 假定编译器对某段高级语言程序编译生成两种不同的指令序列 S1 和 S2，在时钟频率为 500MHz 的机器 M 上运行，目标指令序列中用到的指令类型有 A、B、C 和 D 四类。每类指令在 M 上的 CPI 和两个指令序列所用的各类指令条数如下表所示。

	A	B	C	D
各指令的 CPI	1	2	3	4
S1 的指令条数	5	2	2	1
S2 的指令条数	1	1	1	5

请问：S1 和 S2 各有多少条指令？CPI 各为多少？所含的时钟周期数各为多少？执行时间各为多少？

参考答案:

S1 有 10 条指令，CPI 为 (5×1+2×2+2×3+1×4)/10=1.9，所含的时钟周期数为 10×1.9=19，执行时间为 19/500M = 38ns。

S2 有 8 条指令，CPI 为 (1×1+1×2+1×3+5×4)/8 =3.25，所含的时钟周期数为 8×3.25=26，执行时间为 26/500M = 52ns。

10. 假定机器 M 的时钟频率为 1.2GHz，程序 P 在机器 M 上的执行时间为 12s。对 P 优化时，将其所有的乘 4 指令都换成了一条左移两位的指令，得到优化后的程序 P′。已知在 M 上乘法指令的 CPI 为 5，左移指令的 CPI 为 2，P 的执行时间是 P′执行时间的 1.2 倍，则 P 中有多少条乘法指令被替换成了左移指令被执行？

参考答案：

显然，P'的执行时间为 10 秒，因此，P 比 P'多用了 2 秒，因此，执行时被换成左移指令的乘法指令的条数为 1.2G×2/(5–2) = 800M。

习题课

- 第1章 计算机系统概述
- **第2章 数据的机器级表示**
- 第3章 运算方法和运算部件
- 第4章 指令系统

2. 简单回答下列问题。

(1) 为什么计算机内部采用二进制表示信息？既然计算机内部所有信息都用二进制表示，为什么还要学习十六进制表示？

(2) 常用的定点数编码方式有哪几种？通常它们各自用来表示什么信息？

(3) 为什么现代计算机中大多用补码表示带符号整数？

(4) 在浮点数的基数和总位数一定的情况下，浮点数的表示范围和精度分别由什么决定？两者如何相互制约？

(5) 为什么要对浮点数进行规格化？有哪两种规格化操作？

(6) 为什么计算机处理汉字时会涉及不同的编码(如输入码、内码、字模码)？说明这些编码中哪些用二进制编码，哪些不用二进制编码。为什么？

参考答案:

（1）①二进制只有两种基本状态，制造有两个稳定状态的物理器件容易；

②二进制的编码和运算规则都很简单；

③两个符号1和0与逻辑命题的两个值真和假相对应，便于逻辑运算，也可通过逻辑电路实现算术运算。

学习十六进制：便于阅读和书写。

（2）原码：表示浮点数中的尾数
补码：表示带符号整数
反码：很少被使用
移码：表示浮点数的阶

（3）实现加减运算的统一

（4）表示范围由阶码决定，精度由尾数决定；阶码位数增多（减少），尾数位数减少（增多），需要综合考虑平衡二者

（5）得到尽量多的有效数位，使浮点数表示具有唯一性
左规和右规

（6）

- 输入码：由于汉字字数多，无法使每个汉字与西文键盘上的一个键相对应，必须使每个汉字用一个或几个键来表示，这种编码方式称为“输入码”；（按键组合，不是二进制）
- 内码：汉字被输入到计算机内部后，采用内码进行存储、查找、传送等；（唯一标识，二进制）
- 字模码：描述汉字字模点阵或轮廓，用于显示/打印。（其中，字模点阵码是0/1方阵，是二进制；轮廓描述是关于直线和曲线的公式，不是二进制）

3. 实现下列各数的转换。

(1) $(25.8125)_{10}=(?)_2=(?)_8=(?)_{16}$

(2) $(101101.011)_2=(?)_{10}=(?)_8=(?)_{16}$

(3) $(4E.C)_{16}=(?)_{10}=(?)_2$

参考答案：

(1) $(25.8125)_{10}=(1\ 1001.1101)_2=(31.64)_8=(19.D)_{16}$

(2) $(101101.011)_2=(45.375)_{10}=(55.3)_8=(2D.6)_{16}$

(3) $(4E.C)_{16}=(78.75)_{10}=(0100\ 1110.11)_2$

5. 假定机器数为 8 位(1 位符号,7 位数值),写出下列各二进制整数的补码和移码(偏置常数为 128)表示。

+1001,−1001,+1,−1,+10100,−10100,+0,−0

参考答案:(前面添 0)

	移码	补码
+1001:	10001001	00001001
−1001:	01110111	11110111
+1:	10000001	00000001
−1:	011111111	11111111
+10100:	10010100	00010100
−10100:	01101100	11101100
+0:	10000000	00000000
−0:	10000000	00000000

6. 已知 $[x]_{补}$，求 x。

(1) $[x]_{补}=1.1100111$　　(2) $[x]_{补}=10000000$

(3) $[x]_{补}=0.1010010$　　(4) $[x]_{补}=11010011$

参考答案：

(1) $[x]_{补}=1.1100111$　　$x=-0.0011001B$

(2) $[x]_{补}=10000000$　　$x=-10000000B=-128$

(3) $[x]_{补}=0.1010010$　　$x=+0.101001B$

(4) $[x]_{补}=11010011$　　$x=-101101B=-45$

7. 假定一台 32 位字长的机器中带符号整数用补码表示，浮点数用 IEEE 754 标准表示，寄存器 R1 和 R2 的内容分别为 R1：0000 108BH，R2：8080 108BH。不同指令对寄存器内容进行不同的操作，因而，不同指令执行时寄存器内容对应的真值不同。假定执行下列运算指令时，操作数为寄存器 R1 和 R2 的内容，则 R1 和 R2 中操作数的真值分别为多少？

（1）无符号整数加法指令。

（2）带符号整数乘法指令。

（3）单精度浮点数减法指令。

参考答案：

R1 = 0000108BH = 0000 0000 0000 0000 0001 0000 1000 1011b

R2 = 8080108BH = 1000 0000 1000 0000 0001 0000 1000 1011b

（1）对于无符号数加法指令，R1 和 R2 中是操作数的无符号数表示，因此，其真值分别为 R1：108BH，R2：8080108BH。

（2）对于带符号整数乘法指令，R1 和 R2 中是操作数的带符号整数补码表示，由最高位可知，R1 为正数，R2 为负数。R1 的真值为+108BH，R2 的真值为–(0111 1111 0111 1111 1110 1111 0111 0100b + 1b) = –7F7FEF75H。

（3）对于单精度浮点数减法指令，R1 和 R2 中是操作数的 IEEE754 单精度浮点数表示。在 IEEE 754 标准中，单精度浮点数的位数为 32 位，其中包含 1 位符号位，8 位阶码，23 位尾数。

由 R1 中的内容可知，其符号位为 0，表示其为正数，阶码为 0000 0000，尾数部分为 000 0000 0001 0000 1000 1011，故其为非规格化浮点数，**指数为–126**，尾数中没有隐藏的 1，用十六进制表示尾数为+0.002116H，故 R1 表示的真值为**+0.002116H** × 2^{-126}

由 R2 中的内容可知，其符号位为 1，表示其为负数，阶码为 0000 0001，尾数部分为 000 0000 0001 0000 1000 1011，故其为规格化浮点数，**指数为 1–127 = –126**，尾数中有隐藏的 1，用十六进制表示尾数为–1.002116H，故 R2 表示的真值为**–1.002116H** × 2^{-126}

9. 以下是一个C语言程序，用来计算一个数组 a 中每个元素的和。当参数 len 为 0 时，返回值应该是0，但是在机器上执行时，却发生了存储器访问异常。请问这是什么原因造成的，并说明程序应该如何修改。

```
1  float sum_elements(float a[],unsigned len)
2  {
3      int i;
4      float result=0;
5
6      for (i=0; i<=len-1; i++)
7          result+=a[i];
8
9      return result;
10 }
```

参考答案：

参数 len 的类型是 unsigned，所以，当 len=0 时，执行 len-1 的结果为 11…1，是最大可表示的无符号数，因而，任何无符号数都比它小，使得循环体被不断执行，引起数组元素的访问越界，发生存储器访问异常。

只要将 len 声明为 int 型，或循环的测试条件改为 i<len。

10. 设某浮点数格式为

数符	阶码	尾数
1位	5位移码	6位补码数值部分

其中，移码的偏置常数为16，补码采用一位符号位，基数为4。

(1) 用这种格式表示下列十进制数：+1.75，+19，−1/8。

(2) 写出该格式浮点数的表示范围，并与12位定点补码整数和定点补码小数表示范围比较。

参考答案： (假定采用0舍1入法进行舍入)

(1) $+1.75=+1.11B=0.0111\times2^2=(0.13)_4\times4^1$，故阶码为1+16=17=10001B，尾数位+0.0111的补码，即尾数是0.011100，所以+1.75的表示为0 10001 011100.

$+19=+10011B=0.010011\times2^6=(0.103)_4\times4^3$，故阶码为3 + 16 = 19 = 10011B, 尾数为0.010011，所以+19表示为0 10011 010011。

$-1/8=-0.125=-0.001B=-0.100000\times2^{-2}=(0.1)_4\times4^{-1}$，阶码为 −1 + 16 = 15 = 01111B，尾数为− 0.100000的补码，即1.100000，所以−1/8 表示为1 01111 100000。

(2) 该格式浮点数表示的范围如下：

正数最大值：$0.111111B\times4^{11111}=(0.333)_4\times4^{15}$ ($\approx2^{30}\approx10^9$)

正数最小值：$0.000001B\times4^{00000}=(0.001)_4\times4^{-16}=(\approx2^{-34}\approx10^{-10})$

负数最大值：$-0.000001B\times4^{00000}=(-0.001)_4\times4^{-16}$

负数最小值：$-1.000000B\times4^{11111}=(-1.000)_4\times4^{15}$

因此，该格式浮点数的数量级在10^{-10}~10^9之间。

12位定点补码整数表示范围：-2^{11}~$(2^{11}-1)$，即 -2048~2047

12位定点补码小数表示范围：1.0000~0.1111，即 -1~$(1-2^{-11})$

由此可见，定点数和浮点数的表示范围相差非常大。

12. 以 IEEE 754 单精度浮点数格式表示下列十进制数。

+1.75，+19，−1/8，258

参考答案：

+1.75 = +1.11B = 1.11B × 2^0，故阶码为 0+127=01111111B，数符为 0，尾数为 1.110…0，小数点前为隐藏位，所以+1.7 表示为 0 01111111 110 0000 0000 0000 0000 0000，用十六进制表示为 3FE00000H。

+19 = +10011B = +1.0011B × 2^4，故阶码为 4+127 = 10000011B，数符为 0，尾数为 1.00110…0，所以+19 表示为 0 10000011 001 1000 0000 0000 0000 0000，用十六进制表示为 41980000H。

–1/8 = – 0.125 = – 0.001B = – 1.0 × 2^{-3}，阶码为–3+127 = 01111100B，数符为 1，尾数为 1.0…0，所以–1/8 表示为 1 01111100 000 0000 0000 0000 0000 0000，用十六进制表示为 BE000000H。

258=100000010B=1.0000001B × 2^8，故阶码为 8+127=10000111B，数符为 0，尾数为 1.0000001，所以 258 表示为 0 10000111 000 0001 0000 0000 0000 0000，用十六进制表示为 43810000H。

13. 设一个变量的值为 4098，要求分别用 32 位补码整数和 IEEE 754 单精度浮点格式表示该变量（结果用十六进制表示），并说明哪段二进制序列在两种表示中完全相同，为什么会相同？

参考答案:

$4098 = +1\ \mathbf{0000\ 0000\ 0010}B = +1.0000\ 0000\ 001 \times 2^{12}$

32 位 2-补码形式为：0000 0000 0000 0000 0001 **0000 0000 0010** （00001002H）

IEEE754 单精度格式为：0 10001011 **0000 0000 0010** 0000 0000 000 （45801000H）

粗体部分为除隐藏位外的有效数字，因此，在两种表示中是相同的序列。

第2章 数据的机器级表示

17. 假定在一个程序中定义了变量 x、y 和 i，其中，x 和 y 是 float 型变量(用 IEEE 754 单精度浮点数表示)，i 是 16 位 short 型变量(用补码表示)。程序执行到某一时刻，$x=-0.125$、$y=7.5$、$i=100$，它们都被写到了主存(按字节编址)，其地址分别是 100、108 和 112。请分别画出在大端机器和小端机器上变量 x、y 和 i 在内存的存放位置。

参考答案：

$-0.125=-0.001B=-1.0\times 2^{-3}$

x 在机器内部的机器数为：1　01111100　00...0 (BE00 0000H)

$7.5=+111.1B=+1.111\times 2^{2}$

y 在机器内部的机器数为：0　10000001　11100...0 (40F0 0000H)

100=64+32+4=1100100B

i 在机器内部表示的机器数为：0000 0000 0110 0100（0064H）

	大端机	小端机
地址	内容	内容
100	BEH	00H
101	00H	00H
102	00H	00H
103	00H	BEH
108	40H	00H
109	F0H	00H
110	00H	F0H
111	00H	40H
112	00H	64H
113	64H	00H

习题课

- 第1章 计算机系统概述
- 第2章 数据的机器级表示
- **第3章 运算方法和运算部件**
- 第4章 指令系统

3. 考虑以下 C 语言程序代码：

```
int func1(unsigned word)
{
    return (int)((word<<24)>>24);
}
int func2(unsigned word)
{
    return ((int) word<<24)>>24;
}
```

假设在一个 32 位机器上执行这些函数，该机器使用二进制补码表示带符号整数。无符号数采用逻辑移位，带符号整数采用算术移位。请填写下表，并说明函数 func1 和 func2 的功能。

w		func1(w)		func2(w)	
机器数	值	机器数	值	机器数	值
	127				
	128				
	255				
	256				

参考答案:

W		func1(w)		func2(w)	
机器数	值	机器数	值	机器数	值
0000 007FH	127	0000 007FH	+127	0000 007FH	+127
0000 0080H	128	0000 0080H	+128	FFFF FF80H	–128
0000 00FFH	255	0000 00FFH	+255	FFFF FFFFH	–1
0000 0100H	256	0000 0000H	0	0000 0000H	0

函数 func1 的功能是把无符号数高 24 位清零（左移 24 位再逻辑右移 24 位），结果一定是正的有符号数；而函数 func2 的功能是把无符号数的高 24 位都变成和第 25 位一样，因为左移 24 位后进行算术右移，高 24 位补符号位（即第 25 位）。

4. 填写下表，注意对比无符号整数和带符号整数（用补码表示）的乘法结果，包括截断操作前后的结果。

模　式	x		y		$x \times y$（截断前）		$x \times y$（截断后）	
	机器数	值	机器数	值	机器数	值	机器数	值
无符号整数	110		010					
带符号整数	110		010					
无符号整数	001		111					
带符号整数	001		111					
无符号整数	111		111					
带符号整数	111		111					

参考答案:

模式	x		y		x×y（截断前）		x×y（截断后）	
	机器数	值	机器数	值	机器数	值	机器数	值
无符号数	110	6	010	2	001100	12	100	4
二进制补码	110	–2	010	+2	111100	–4	100	–4
无符号数	001	1	111	7	000111	7	111	7
二进制补码	001	+1	111	–1	111111	–1	111	–1
无符号数	111	7	111	7	110001	49	001	1
二进制补码	111	–1	111	–1	000001	+1	001	+1

5. 以下是两段 C 语言代码，函数 arith()是直接用 C 语言写的，而 optarith()是对 arith()函数以某个确定的 M 和 N 编译生成的机器代码反编译生成的。根据 optarith()推断函数 arith() 中 M 和 N 的值各是多少？

```
#define M
#define N
int arith(int x, int y)
{
    int result=0;
    result=x*M+y/N;
    return result;
}

int optarith(int x, int y)
{
    int t=x;
    x<<=4;
    x-=t;
    if(y<0) y+=3;
    y>>=2;
    return x+y;
}
```

参考答案:

可以看出 x*M 和“int t = x; x <<= 4; x-=t;”三句对应，这些语句实现了 x 乘 15 的功能（左移 4 位相当于乘以 16，然后再减 1），因此，M 等于 15；

y/N 与“if (y < 0) y += 3; y>>2;”两句对应，第二句“y 右移 2 位”实现了 y 除以 4 的功能，因此 N 是 4。而第一句“if (y < 0) y += 3;”主要用于对 y= –1 时进行调整，若不调整，则–1>>2= –1 而–1/4=0，两者不等；调整后 –1+3=2，2>>2=0，两者相等。

7. 已知 $x=10, y=-6$，采用 6 位机器数表示。请按如下要求计算，并把结果还原成真值。

(1) 求 $[x+y]_{补}$，$[x-y]_{补}$。

(2) 用原码一位乘法计算 $[x \times y]_{原}$。

(3) 用 MBA(基 4 布斯算法)计算 $[x \times y]_{补}$。

(4) 用不恢复余数法计算 $[x/y]_{原}$ 的商和余数。

(5) 用不恢复余数法计算 $[x/y]_{补}$ 的商和余数。

参考答案:

$[10]_{补}=001010$　　$[-6]_{补}=111010$　　$[6]_{补}=000110$　　$[10]_{原}=001010$　　$[-6]_{原}=100110$

（1） $[10+(-6)]_{补}=[10]_{补}+[-6]_{补}=001010+111010=000100\ (+4)$

$[10-(-6)]_{补}=[10]_{补}+[-(-6)]_{补}=001010+000110=010000\ (+16)$

（2） 先采用无符号数乘法计算 001010× 000110 的乘积，原码一位乘法过程（前面两个 0 省略）如下：

C	P	Y	说明
0	0 0 0 0	0 1 1 0	$P_0=0$
	+0 0 0 0		$y_4=0$，+0
0	0 0 0 0		C, P 和 Y 同时右移一位
0	0 0 0 0	0 0 1 1	得 P_1
	+1 0 1 0		$y_3=1$，+X
0	1 0 1 0		C, P 和 Y 同时右移一位
0	0 1 0 1	0 0 0 1	得 P_2
	+1 0 1 0		$y_2=1$，+X
0	1 1 1 1	0 0 0 0	C, P 和 Y 同时右移一位
0	0 1 1 1	1 0 0 0	得 P_3
	+0 0 0 0		$y_1=0$，+0
0	0 1 1 1		C, P 和 Y 同时右移一位
0	0 0 1 1	1 1 0 0	得 P_4

若两个 6 位数相乘的话，则还要右移两次，得 000000 111100

符号位为：$0\oplus 1=1$，因此，$[X\times Y]_{原}=1000\ 0011\ 1100$

即 $X\times Y=-11\ 1100B=-60$

（3） $[-10]_{补}=110110$，布斯乘法过程如下：

P	Y	y_{-1}	说明
000000	111010	0	设 $y_{-1}=0$，$[P_0]_{补}=0$
			$y_0y_{-1}=00$，P、Y 直接右移一位
000000	011101	0	得$[P_1]_{补}$
+110110			$y_1y_0=10$，$+[-X]_{补}$
110110			P、Y 同时右移一位
111011	001110	1	得$[P_2]_{补}$
+001010			$y_2y_1=01$，$+[X]_{补}$
000101			P、Y 同时右移一位
000010	100111	0	得$[P_3]_{补}$
+110110	100111	0	$y_3y_2=10$，$+[-X]_{补}$
111000			P、Y 同时右移一位
111100	010011	1	得$[P_4]_{补}$
+000000	010011	1	$y_4y_3=11$，+0
111100			P、Y 同时右移一位
111110	001001	1	得$[P_5]_{补}$
+000000	001001	1	$y_5y_4=11$，+0
111110			P、Y 同时右移一位
111111	000100	1	得$[P_6]_{补}$

因此，$[X\times Y]_{补}$=1111 1100 0100，即 $X\times Y=-11\ 1100B=-60$

（4） 因为除法计算是 2n 位数除 n 位数，所以$[6]_{原}$=0110，$[10]_{原}$=0000 1010，$[-6]_{补}$=1010，商的符号位：0⊕1=1，运算过程（前面两个 0 省略）如下：

余数寄存器 R	余数/商寄存器 Q	说　明
0 0 0 0	1 0 1 0 □	开始 $R_0=X$
+1 0 1 0		$R_1=X-Y$
1 0 1 0	1 0 1 0 0	$R_1<0$，则 $q_4=0$，没有溢出
0 1 0 1	0 1 0 0 □	$2R_1$（R 和 Q 同时左移，空出一位商）
+0 1 1 0		$R_2=2R_1+Y$
1 0 1 1	0 1 0 0 0	$R_2<0$，则 $q_3=0$
0 1 1 0	1 0 0 0 □	$2R_2$ （R 和 Q 同时左移，空出一位商）
+0 1 1 0		$R_3=2R_2+Y$
1 1 0 0	1 0 0 0 0	$R_3<0$，则 $q_2=0$
1 0 0 1	0 0 0 0 □	$2R_3$ （R 和 Q 同时左移，空出一位商）
+0 1 1 0		$R_3=2R_2+Y$
1 1 1 1	0 0 0 0 0	$R_4<0$，则 $q_1=0$
1 1 1 0	0 0 0 0 □	$2R_4$ （R 和 Q 同时左移，空出一位商）
+0 1 1 0		$R_5=2R_4+Y$
0 1 0 0	0 0 0 0 1	$R_5>0$，则 $q_0=1$

商的数值部分为：00001。所以，$[X/Y]_{原}$=00001 (最高位为符号位)，余数为 0100。

（5） 将 10 和–6 分别表示成补码形式为：$[10]_{补}=0\ 1010$，$[-6]_{补}=1\ 1010$，计算过程如下：

先对被除数进行符号扩展，$[10]_{补}=00000\ 01010$，$[6]_{补}=0\ 0110$

余数寄存器 R	余数/商寄存器 Q	说　明
00000	01010	开始 $R_0=[X]$
+11010		$R_1=[X]+[Y]$
11010	01010	R_1 与[Y]同号，则 $q_5=1$
10100	10101	$2R_1$（R 和 Q 同时左移，空出一位上商 1）
+00110		$R_2=2R_1+[-Y]$
11010	10101	R_2 与[Y]同号，则 $q_4=1$，
10101	01011	$2R_2$（R 和 Q 同时左移，空出一位上商 1）
+00110		$R_3=2R_2+[-Y]$
11011	01011	R_3 与[Y]同号，则 $q_3=1$
10110	10111	$2R_3$（R 和 Q 同时左移，空出一位上商 1）
+00110		$R_4=2R_3+[-Y]$
11100	10111	R_4 与[Y]同号，则 $q_2=1$
11001	01111	$2R_4$（R 和 Q 同时左移，空出一位上商 0）
+00110		$R_5=2R_4+[-Y]$
11111	01111	R_5 与[Y]同号，则 $q_1=1$，
11110	11111	$2R_5$（R 和 Q 同时左移，空出一位上商 1）
+00110		$R_6=2R_5+[-Y]$
00100	11110	R_6 与[Y]异号，则 $q_0=0$，Q 左移，空出一位上商 1
+00000	+　　1	商为负数，末位加 1；余数不需要修正
00100	11111	

所以，$[X/Y]_{补}=11111$，余数为 00100。

即：X/Y=–0001B=–1，余数为 0100B=4

将各数代入公式“除数×商+余数= 被除数”进行验证，得：(–6)×(–1) +4= 10。

11. 假设浮点数格式为：阶码是 4 位移码，偏置常数为 8，尾数是 6 位补码（采用双符号位），用浮点运算规则分别计算以下表达式在不采用任何附加位和采用 2 位附加位（保护位、舍入位）两种情况下的值（假定采用就近舍入到偶数方式）。

(1) $(15/16)\times 2^{7}+(2/16)\times 2^{5}$　　(2) $(15/16)\times 2^{7}-(2/16)\times 2^{5}$

(3) $(15/16)\times 2^{5}+(2/16)\times 2^{7}$　　(4) $(15/16)\times 2^{5}-(2/16)\times 2^{7}$

参考答案：

将上述各式中的数据用相应的变量 A、B、C、D 代替。

$A=(15/16)\times2^7=0.1111B\times2^7$，$[A]_{浮}=00.1111,1111$。

$B=(2/16)\times2^5=0.0010B\times2^5=0.1000B\times2^3$，$[B]_{浮}=00.1000,1011$。

$C=(15/16)\times2^5=0.1111B\times2^5$，$[C]_{浮}=00.1111,1101$。

$D=(2/16)\times2^7=0.0010B\times2^7=0.1000B\times2^5$，$[D]_{浮}=00.1000,1101$。

不采用任何附加位时的计算结果如下。

（1）计算 A+B：$[\Delta E]_{补}=[E_A]_{补}+[-[E_B]_{补}]_{补}=1111+0101=0100\ (\mathrm{mod}\ 2^n)$，因此 $\Delta E=4$，故需对 B 进行对阶，因为采用就近舍入到偶数方式，所以，B 的尾数右移四位后直接舍去 1000，又因为“1000”是中间值，因此尾数取偶数 00.0000，故对阶后结果为 $[B]_{浮}=00.0000,1111$。由于 B 的尾数为 0，因此，$[A+B]_{浮}=[A]_{浮}=00.1111,1111$。故 $A+B=A=(15/16)\times2^7$。

（2）计算 A–B：对阶结果与（1）相同，故 $[A-B]_{浮}=[A]_{浮}=00.1111,1111$。故 $A-B=A=(15/16)\times2^7$。

（3）计算 C+D：$[\Delta E]_{补}=[E_C]_{补}+[-[E_D]_{补}]_{补}=1101+0011=0000\ (\mathrm{mod}\ 2^n)$，因此 $\Delta E=0$，故无需对阶。尾数直接加：$[M_C]_{补}+[M_D]_{补}=00.1111+00.1000=01.0111$，因为补码的两个符号位不同，所以尾数溢出，需要右规。右规时需对尾数进行舍入，阶码加 1。舍入最后一位的“1”是中间值，因此尾数取偶数 00.1100，阶码 1101 加 1 后为 1110，所以，$[C+D]_{浮}=00.1100,1110$。故 $C+D=(12/16)\times2^6$。

（4）计算 C–D：$[\Delta E]_{补}=[E_C]_{补}+[-[E_D]_{补}]_{补}=1101+0011=0000\ (\mathrm{mod}\ 2^n)$，因此 $\Delta E=0$，故无需对阶。尾数直接减：$[M_C]_{补}+[-M_D]_{补}=00.1111+11.1000=00.0111$。显然，尾数需左规。左规时，尾数左移一位，阶码减 1。因此，最终尾数为 00.1110，阶码 1101 减 1 后为 1100。因此，$[C-D]_{浮}=00.1110,1100$，故 $C-D=(14/16)\times2^4$。

采用两位附加位时的计算结果如下。

（1）计算 A+B：$[\Delta E]_{补} = [E_A]_{补} + [-[E_B]_{补}]_{补} = 1111+0101 = 0100 \pmod{2^n}$，因此 $\Delta E = 4$，故需对 B 进行对阶，对阶后结果为 $[B]_{浮} = 00.0000\ 10, 1111$。尾数相加结果为：$[M_A]_{补}+[M_B]_{补} = 00.1111\ 00 + 00.0000\ 10 = 00.1111\ 10$，因此，$[A+B]_{浮} = 00.1111\ 10, 1111$。最后对尾数附加位 10 进行舍入，因为舍入的是中间值，所以尾数结果强迫为偶数，即尾数末位加 1，得尾数为 01.0000，因此，尾数需右规为 00.1000，同时，阶码 1111 加 1，产生阶码上溢，因而导致结果溢出。因此，A+B 的结果溢出。

（2）计算 A–B：$[\Delta E]_{补} = [E_A]_{补} + [-[E_B]_{补}]_{补} = 1111+0101 = 0100 \pmod{2^n}$，因此 $\Delta E = 4$，故需对 B 进行对阶，对阶后结果为 $[B]_{浮} = 00.0000\ 10, 1111$。尾数相减结果为：$[M_A]_{补}+[-M_B]_{补} = 00.1111\ 00 + 11.1111\ 10 = 00.1110\ 10$，因此，$[A-B]_{浮} = 00.1110\ 10, 1111$。最后对尾数附加位 10 进行舍入，因为舍入的是中间值，所以尾数结果强迫为偶数，得尾数为 00.1110，因此，$[A-B]_{浮} = 00.1110, 1111$。故 $A-B=(14/16)\times 2^7$。

（3）计算 C+D：$[\Delta E]_{补} = [E_C]_{补} + [-[E_D]_{补}]_{补} = 1101+0011 = 0000 \pmod{2^n}$，因此 $\Delta E = 0$，故无需对阶。尾数直接加：$[M_C]_{补}+[M_D]_{补} = 00.1111\ 00+00.1000\ 00 = 01.0111\ 00$，因为补码的两个符号位不同，所以尾数溢出，需要右规。右规时需对尾数进行舍入，阶码加 1。舍入的“100”是中间值，因此尾数取偶数 00.1100，阶码 1101 加 1 后为 1110，所以，$[C+D]_{浮} = 00.1100, 1110$。故 $C+D=(12/16)\times 2^6$。

（4）计算 C–D：$[\Delta E]_{补} = [E_C]_{补} + [-[E_D]_{补}]_{补} = 1101+0011 = 0000 \pmod{2^n}$，因此 $\Delta E = 0$，故无需对阶。尾数直接减：$[M_C]_{补}+[-M_D]_{补} = 00.1111\ 00+11.1000\ 00 = 00.0111\ 00$。显然，尾数需左规。左规时，尾数左移一位，阶码减 1。因此，最终尾数为 00.1110，阶码 1101 减 1 后为 1100。因此，$[C-D]_{浮} = 00.1110, 1100$，故 $C-D=(14/16)\times 2^4$。

12. 采用 IEEE 754 单精度浮点数格式计算下列表达式的值。

(1) 0.75+(−65.25)　　(2) 0.75−(−65.25)

参考答案:

$x = 0.75 = 0.110...0B = (1.10...0)_2 \times 2^{-1}$

$y = -65.25 = -1000001.01000...0B = (-1.00000101...0)_2 \times 2^6$

用 IEEE 754 标准单精度格式表示为:

$[x]_{浮} = 0$　01111110　10...0　　$[y]_{浮} = 1$　10000101　000001010...0

所以，$E_x = 01111110$，$M_x = 0\ (1).\ 1...0$ ，$E_y = 10000101$，$M_y = 1(1).000001010...0$

尾数 M_x 和 M_y 中小数点前面有两位，第一位为数符，第二位加了括号，是隐藏位“1”。

以下是计算机中进行浮点数加减运算的过程（假定保留 2 位附加位：保护位和舍入位）

（1）0.75+ (– 65.25)

① 对阶： $[\Delta E]_{补} = [E_x]_{移} + [-[E_y]_{移}]_{补} \pmod{2^n} = 0111\ 1110 + 0111\ 1011 = 1111\ 1001$

$\Delta E = -7$，根据对阶规则可知需要对 x 进行对阶，结果为：$E_x = E_y = 10000101$，$M_x = 00.000000110...000$

x 的尾数 M_x 右移 7 位，符号不变，数值高位补 0，隐藏位右移到小数点后面，最后移出的 2 位保留

② 尾数相加：$M_b = M_x + M_y = 00.\ 000000110...000 + 11.\ 000001010\ ...000$ （注意小数点在隐藏位后）

根据原码加/减法运算规则，得：$00.000000110...000 + 11.000001010...000 = 11.000000100...000$

上式尾数中最左边第一位是符号位，其余都是数值部分，尾数后面两位是附加位（加粗）。

③ 规格化：根据所得尾数的形式，数值部分最高位为 1，所以不需要进行规格化。

④ 舍入：把结果的尾数 M_b 中最后两位附加位舍入掉，从本例来看，不管采用什么舍入法，结果都一样，都是把最后两个 0 去掉，得：$M_b = 11.000000100...0$

⑤ 溢出判断：在上述阶码计算和调整过程中，没有发生“阶码上溢”和“阶码下溢”的问题。因此，阶码 $E_b = 10000101$。

最后结果为 $E_b = 10000101$，$M_b = 1(1).00000010...0$，即：– 64.5。

（2） 0.75–(– 65.25）

① 对阶： $[\Delta E]_{补} = [E_x]_{移} + [-[E_y]_{移}]_{补} \pmod{2^n} = 0111\ 1110 + 0111\ 1011 = 1111\ 1001$

$\Delta E = -7$，根据对阶规则可知需要对 x 进行对阶，结果为：$E_x = E_y = 10000110$，$M_x = 00.000000110...000$

x 的尾数 M_x 右移一位，符号不变，数值高位补 0，隐藏位右移到小数点后面，最后移出的位保留

② 尾数相加：$M_b = M_x - M_y = 00.\ 000000110...000 - 11.\ 000001010...000$ （注意小数点在隐藏位后）

根据原码加/减法运算规则，得：00.000000110...000 – 11.000001010...000=01.00001000…000

上式尾数中最左边第一位是符号位，其余都是数值部分，尾数后面两位是附加位（加粗）。

③ 规格化：根据所得尾数的形式，数值部分最高位为 1，不需要进行规格化。

④ 舍入：把结果的尾数 M_b 中最后两位附加位舍入掉，从本例来看，不管采用什么舍入法，结果都一样，都是把最后两个 0 去掉，得：$M_b = 01.00001000…0$

⑤ 溢出判断：在上述阶码计算和调整过程中，没有发生“阶码上溢”和“阶码下溢”的问题。因此，阶码 $E_b = 10000101$。

最后结果为 $E_b = 10000101$，$M_b = 0(1).00001000…0$，即：+66。

习题课

- 第1章 计算机系统概述
- 第2章 数据的机器级表示
- 第3章 运算方法和运算部件
- **第4章 指令系统**

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？

参考答案：

因为执行到该转移指令时 PC 为 200，所以说明该转移指令存放在 200 单元开始的两个字节中。因为 CPU 每次从内存只能取一个字节，所以每次取一个字节后 PC 应该加 1。

该转移指令的执行过程为：取 200 单元中的指令操作码并译码→PC+1→取 201 单元的相对位移量→PC+1→计算转移目标地址。假设该转移指令第二字节为 Offset，则 100=200+2+Offset，即 Offset = 100–202 = –102 = 10011010B

（注：没有说定长指令字，所以不一定是每条指令占 2 个字节。）

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。

参考答案：

(1) 有效地址EA=00FCH+1200H=12FCH;

(2) 有效地址EA=(1200H)=12FCH;

(3) 有效地址EA=1200H。

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？

参考答案：

二地址操作码位数： $16-6\times2=4$

除去二地址指令，未使用的4位操作码数目： $2^4-k2=16-k2$

单地址操作码位数： $16-6=10$ (比二地址多6位)

设单地址指令为k1条，除去单地址指令，未使用的10位操作码数目： $(16-k2)\times2^6-k1$

零地址操作码位数： 16 (比单地址多6位)

零地址指令条数： $k0=((16-k2)\times2^6-k1)\times2^6$

所以： $k1=(16-k2)\times2^6-k0/2^6$

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？

(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？

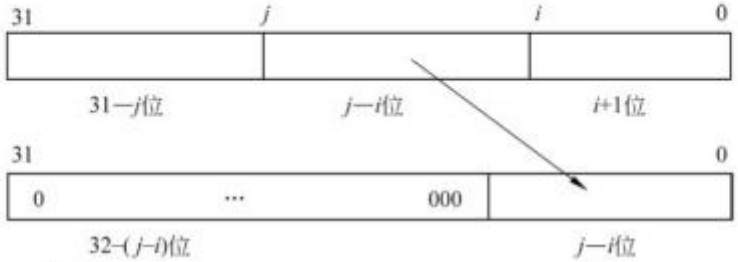
参考答案:

(1) 因为采用单字长指令格式,操作码字段占 4 位,所以最多有 16 条指令;指令中通用寄存器编号占 3 位,所以,最多有 8 个通用寄存器;因为地址空间大小为 128KB,按字编址,故共有 64K 个存储单元,因而地址位数为 16 位,所以 MAR 至少为 16 位;因为字长为 16 位,所以 MDR 至少为 16 位。

(2) 因为地址位数和字长都为 16 位,所以 PC 和通用寄存器位数均为 16 位,两个 16 位数据相加其结果也为 16 位,即转移目标地址位数为 16 位,因而能在整个地址空间转移,即目标转移地址的范围为 0000H～FFFFH。

(3) 要得到汇编语句"add(R4),(R5)+"对应的机器码,只要将其对应的指令代码各个字段拼接起来即可。显然,add 对应 op 字段,为 0010B;(R4)的寻址方式字段为 001B,R4 的编号为 100B;(R5)+的寻址方式字段为 010B,R5 的编号为 101B;因此,对应的机器码为 0010 001 100 010 101B,用十六进制表示为 2315H。指令"add(R4),(R5)+"的功能为:M[R5]←M[R[R5]]+M[R[R4]],R[R5]←R[R5]+1。已知 R[R4]=1234H,R[R5]=5678H,M[1234H]=5678H,M[5678H]=1234H,因为 1234H+5678H=68ACH,所以 5678H 单元中的内容从 1234H 改变为 68ACH,同时 R5 中的内容从 5678H 变为 5679H。

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。



参考答案：

可以先左移 9 位，然后右移 15 位，即：

```
sll $s2, $s0, 9     #把高位非 0 值撞出去
srl $s2, $s2, 15    #把低位非 0 值撞出去
```

思考：(1) 第二条用算术右移指令 sra 行不行？

不行，因为不能保证高位补 0！

(2) 若第一条指令中的$s2 改成其他寄存器 R，则会带来什么问题？

所用寄存器 R 的值被破坏！

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
```

参考答案：

1: 将 t0 寄存器置零

2: 如果 a1 的值等于零则程序转移到 finish 处

3: 将 t0 和 a0 的内容相加，结果存放于 t0

4: 将 a1 的值减 1

5: 无条件转移到 loop 处

6: 将 t0 的内容加上 100，结果存放于 t0

7: 将 t0 的值存放在 v0

该程序的功能是计算“100+a×b”

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
        sll    $a3, $a3, 2
        add    $v0, $zero, $zero
        add    $t0, $zero, $zero
outer:  add    $t4, $a0, $t0
        lw     $t4, 0($t4)
        add    $t1, $zero, $zero
inner:  add    $t3, $a1, $t1
        lw     $t3, 0($t3)
        bne    $t3, $t4, skip
        addi   $v0, $v0, 1
skip:   addi   $t1, $t1, 4
        bne    $t1, $a3, inner
        addi   $t0, $t0, 4
        bne    $t0, $a2, outer
```

参考答案:

1: 将 a2 的内容左移 2 位，即乘 4
2: 将 a3 的内容左移 2 位，即乘 4
3: 将 v0 置零
4: 将 t0 置零
5: 将第一个数组的首地址存放在 t4
6: 取第一个数组的第一个元素存放在 t4
7: 将 t1 置零
8: 将第二个数组的首地址存放在 t3
9: 取第二个数组的第一个元素存放在 t3
10: 如果 t3 和 t4 不相等，则跳转到 skip
11: 将 v0 的值加 1，结果存于 v0
12: 将 t1 的值加 4，结果存于 t1
13: 如果 t1 不等于 a3，即还未取完数组中所有元素，则转移到 inner
14: 将 t0 的值加 4
15: 如果 t0 不等于 a2，即还未取完数组中所有元素，则转移到 outer
该程序的功能是统计两个数组中相同元素的个数。
程序最坏的情况是：两个数组所有元素都相等，这样每次循环都不会执行 skip。

指令总条数为：4+2500×(3+2500×6+2)=37512504
其中：add, addi和sll指令条数为：4+2500×(2+2500×3+1)=18757504
lw和bne的指令条数为：2500×(1+2500×3+1)=18755000
时钟周期为：1/(2G)=0.5ns
所以：程序执行时间为：(18757504×1+18755000×2)×0.5ns=28133753ns≈0.028s

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？

参考答案:

只要用一条指令 ori $t1, $t0, 25 就可实现。

如果把 25 换成 65536，则不能用一条指令 ori $t1, $t0, 65536 来实现，因为 65536 （1 0000 0000 0000 0000B）不能用 16 位立即数表示。可用以下两条指令实现。

```
lui  $t1, 1
or   $t1, $t0, $t1
```

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
```

参考答案：

修改后的代码如下：

```
        addi  Sv0, Szero, 0
loop:   lw    Sv1, 0(Sa0)
        sw    Sv1, 0(Sa1)
        beq   Sv1, Szero, exit
        addi  Sa0, Sa0, 4
        addi  Sa1, Sa1, 4
        addi  Sv0, Sv0, 1
        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;
}
```

参考答案:

程序由两个过程组成，全局静态变量 sum 分配给$s0。

为了尽量减少指令条数，并减少访问内存次数。在每个过程的过程体中总是先使用临时寄存器$t0~$t9，临时寄存器不够或者某个值在调用过程返回后还需要用，就使用保存寄存器$s0~$s7。

MIPS 指令系统中没有寄存器传送指令，为了提高汇编表示的可读性，引入一条伪指令 move 来表示寄存器传送，汇编器将其转换为具有相同功能的机器指令。伪指令“move $t0, $s0”对应的机器指令为“add $t0,$zero,$s0”。

（1）过程 set_array：该过程和教材中例 5.10 中的不同，在例 5.10 中 array 数组是过程 sum_array 的局部变量，应该在过程栈帧中给数组分配空间，但该题中的数组 array 是在其他过程中定义的，仅将其数组首地址作为参数传递给过程 sum_array（假定在在$a0 中），因此，无需在其栈帧中给数组分配空间。此外，还有一个入口参数为 num（假定在$a1 中），有一个返回参数 sum，被调用过程为 compare。因此，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，以免在调用过程 compare 时被覆盖。是否保存$fp 要看具体情况，如果确保后面都不用到$fp，则可以不保存，但为了保证$fp 的值不被后面的过程覆盖，通常情况下，应该保存$fp 的值。

栈帧中要保存的信息只有返回地址Sra 和帧指针Sfp，其栈帧空间为 4×2=8B。

汇编表示如下：

```
            move  Ss0, Szero        # sum=0
set-array:  addi  Ssp, Ssp, –8      # generate stack frame
            sw    Sra, 4(Ssp)       # save Sra on stack
            sw    Sfp, 0(Ssp)       # save Sfp on stack
            addi  Sfp, Ssp, 4       # set Sfp
            move  St2, Sa0          # base address of array
            move  St0, Sa1          # St0=num
            move  St3, Szero            # i=0
loop:       slt   St1, St3, St0         # if i<num, St1=1; if i>=num, St1= 0
            beq   St1, Szero, exit1     # if St1= 0, jump to exit1
            move  Sa0, St0          # Sa0=num
            move  Sa1, St3          # Sa1=i
            addi  Sa1, Sa1, 1       # Sa1=i+1
            jal   compare               # call compare
            beq   Sv0, Szero, else  # if Sv0 = 0, jump to else
            sll   St1, St3, 2       # i×4
            add   St1, St2, St1     # St1=array[i]
            lw    St4, 0(St1)       # load array[i]
            add   Ss0, Ss0, St4     # sum+=array[i]
```

```
        j       loop
exit1:  move    $v0, $s0        # return sum
        lw      $ra, 4($sp)     # restore $ra
        lw      $fp, 0($sp)     # restore $fp
        addi    $sp, $sp, 8     # free stack frame
        jr      $ra             # return to caller
```

（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:  move  $v0, $zero      # return 0
          beq   $a0, $a1, exit2 # if $a0=$a1, jump to exit2
          slt   $t1, $a0, $a1   # if $a0<$a1, $t1=1; if $a0>=$a1, $t1= 0
          bne   $t1, $zero, exit2 # if $a0<$a1, jump to exit2
          ori   $v0, $zero,1    # return 1
exit2:    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？

参考答案:

RV32I指令：按位与的指令andi的立即数用12比特表示

当立即数是31时，可表示:

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

当立即数是65535=2^{16}-1时，超出表示范围，转换为RV32I指令序列:

```
lui $t1, 65535
srli $t1, 12
and $t1, $t0, $t1
```

与12题相比，RV32I指令中的立即数表示存在位数和位置限制

提问

Q & A

孟晴开
智能软件与工程学院
南雍楼东区228
qkmeng@nju.edu.cn，https://mengqingkai.github.io/

南京大学 NANJING UNIVERSITY