# 第一次作业

6. 若机器 M1 和 M2 具有相同的指令集，其时钟频率分别为 1GHz 和 1.5GHz。在指令集中有五种不同类型的指令 A~E。下表给出了在 M1 和 M2 上每类指令的平均时钟周期数 CPI。

| 机器 | A | B | C | D | E |
|---|---|---|---|---|---|
| M1 | 1 | 2 | 2 | 3 | 4 |
| M2 | 2 | 2 | 4 | 5 | 6 |

请回答下列问题：

（1）M1 和 M2 的峰值 MIPS 各是多少？

（2）假定某程序 P 的指令序列中，五类指令具有完全相同的指令条数，则程序 P 在 M1 和 M2 上运行时，哪台机器更快？快多少？在 M1 和 M2 上执行程序 P 时的平均时钟周期数 CPI 各是多少？

参考答案：

（1）M1 上可以选择一段都是 A 类指令组成的程序，其峰值 MIPS 为 1000MIPS。

M2 上可以选择一段 A 和 B 类指令组成的程序，其峰值 MIPS 为 1500/2=750MIPS。

（2）5 类指令具有完全相同的指令条数，所以各占 20%。

在 M1 和 M2 上执行程序 P 时的平均时钟周期数 CPI 分别为：

M1：20%×(1+2+2+3+4)= 0.2×12 = 2.4

M2：20%×(2+2+4+5+6)= 0.2×19 = 3.8

假设程序 P 的指令条数为 N，则在 M1 和 M2 上的执行时间分别为：

M1：2.4× N×1/1G = 2.4N (ns)

M2：3.8×N×1/1.5G = 2.53 N (ns)

M1 执行 P 的速度更快，每条指令平均快 0.13ns，也即 M1 比 M2 快 0.13/2.53×100% ≈ 5%。

7. 假设同一套指令集用不同的方法设计了两种机器 M1 和 M2。机器 M1 的时钟周期为 0.8ns，机器 M2 的时钟周期为 1.2ns。某个程序 P 在机器 M1 上运行时的 CPI 为 4，在 M2 上的 CPI 为 2。对于程序 P 来说，哪台机器的执行速度更快？快多少？

参考答案：

假设程序 P 的指令条数为 N，则在 M1 和 M2 上的执行时间分别为：

M1：4 N×0.8 = 3.2N (ns)

M2：2 N×1.2 = 2.4 N (ns)

所以，M2 执行 P 的速度更快，每条指令平均快 0.8ns，比 M1 快 0.8/3.2×100%=25%。

8. 假设某机器 M 的时钟频率为 4GHz，用户程序 P 在 M 上的指令条数为 $8\times10^9$，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 4 秒，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. 假定某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。四类指令在 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？

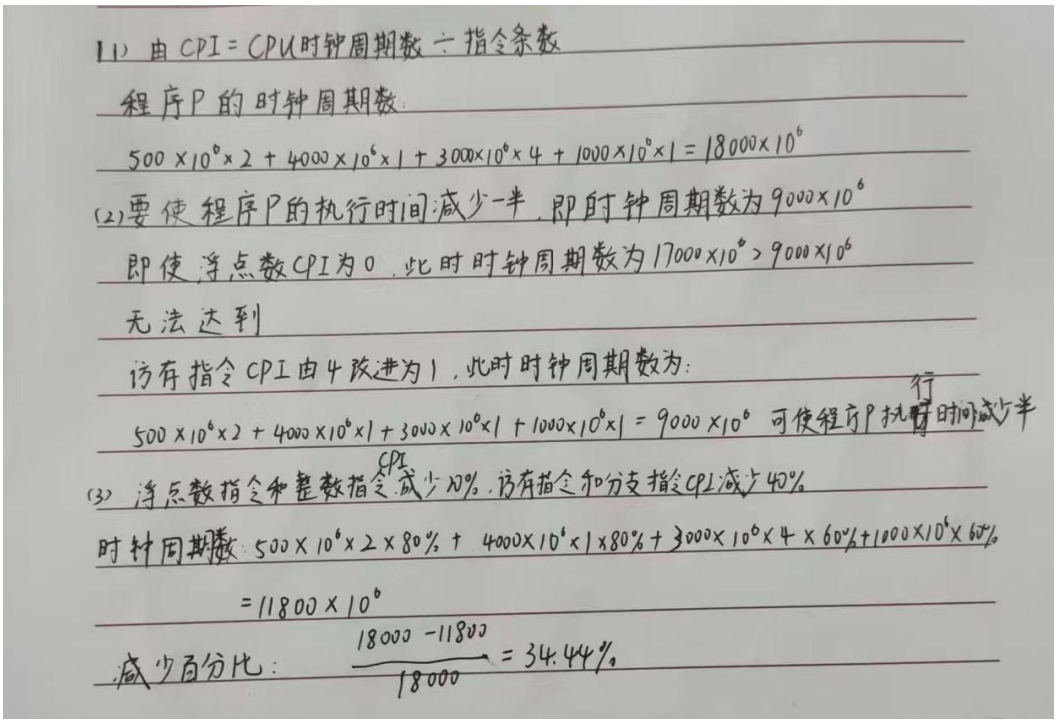
参考答案：

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。

（注：从上述结果来看，对于同一个高级语言源程序，在同一台机器上所生成的目标程序不同，其执行时间可能不同，而且，并不是指令条数少的目标程序执行时间就一定少。）

**11. （第 3 版新增习题）假定机器 M 在运行程序 P 的过程中，共执行了 $500\times10^6$ 条浮点数指令，$4000\times10^6$ 条整数指令、$3000\times10^6$ 条访存指令、$1000\times10^6$ 条分支指令，这 4 种指令的 CPI 分别是 2、1、4、1。若要使程序 P 的执行时间减少一半，则浮点数指令的 CPI 应如何改进？若要使程序 P 的执行时间减少一半，则访存指令的 CPI 应如何改进？若浮点数指令和整数指令的 CPI 减少 20%，访存指令和分支指令的 CPI 减少 40%，则程序 P 的执行时间会减少多少？**



# 第二次作业

5. 假定机器数为 8 位（1 位符号，7 位数值），写出下列各二进制数的补码和移码表示。
+1001，–1001，+1，–1，+10100，–10100，+0，–0

参考答案：

| | 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 |

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 R1：0000108BH，R2：8080108BH。不同指令对寄存器进行不同的操作，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 × $10^{-126}$。

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 × $10^{-126}$

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 **0000 0000 0010**B = +1. 0000 0000 001 × $2^{12}$

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。

15. 下表给出了有关 IEEE 754 浮点格式表示中一些重要数据的取值，表中已经有最大规格化数的相应内容，要求填入其他浮点数的相应内容。（注：表中 a 代表一个在 1 到 10 之间的正纯小数）

| 项目 | 阶码 | 尾数 | 单精度 | | 双精度 | |
|---|---|---|---|---|---|---|
| | | | 以 2 的幂次表示的值 | 以 10 的幂次表示的值 | 以 2 的幂次表示的值 | 以 10 的幂次表示的值 |
| 0 | 00000000 | 0…00 | 0 | 0 | 0 | 0 |
| 1 | 01111111 | 0…00 | 1 | 1 | 1 | 1 |
| 最大规格化数 | 11111110 | 1…11 | $(2-2^{-23})\times2^{127}$ | $a\times10^{38}$ | $(2-2^{-52})\times2^{1023}$ | $a\times10^{308}$ |
| 最小规格化数 | 00000001 | 0…00 | $1.0\times2^{-126}$ | $a\times10^{-38}$ | $1.0\times2^{-1022}$ | $a\times10^{-308}$ |
| 最大非规格化数 | 00000000 | 1…11 | $(1-2^{-23})\times2^{-126}$ | $a\times10^{-38}$ | $(1-2^{-52})\times2^{-1022}$ | $a\times10^{-308}$ |
| 最小非规格化数 | 00000000 | 0…01 | $2^{-23}\times2^{-126}=2^{-149}$ | $a\times10^{-44}$ | $2^{-52}\times2^{-1022}$ | $a\times10^{-?}$ |
| +∞ | 11111111 | 0…00 | – | – | – | – |
| NaN | 11111111 | 非全 0 | – | – | – | – |

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 IEEE754 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 = $-1.0\times2^{-3}$

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 |

# 第三次作业

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 |
| 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）。

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$[x\times y]_{原}$。

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 $X \times Y = -11\ 1100B = -60$

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。（假定对阶和右规时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（假定采用隐藏位）：

$X = (15/16) \times 2^7 = 0.111100B \times 2^7 = (1.111000)_2 \times 2^6$

$Y1 = (2/16) \times 2^5 = 0.001000B \times 2^5 = (1.000000)_2 \times 2^2$

$Y2 = (-2/16) \times 2^5 = -0.001000B \times 2^5 = (-1.000000)_2 \times 2^2$

$K = (15/16) \times 2^5 = 0.111100B \times 2^5 = (1.111000)_2 \times 2^4$

$J1 = (2/16) \times 2^7 = 0.001000B \times 2^7 = (1.000000)_2 \times 2^4$

$J2 = (-2/16) \times 2^7 = -0.001000B \times 2^7 = (-1.000000)_2 \times 2^4$

根据题目所给的各种位数，可以得到在机器中表示为：

$[X]_{浮}$ = 00　1110　(1)111000　　$[Y1]_{浮}$ = 00　1010　(1)000000　$[Y2]_{浮}$ = 11　1010　(1)000000

$[K]_{浮}$ = 00　1100　(1)111000　　$[J1]_{浮}$ = 00　1100　(1)000000　$[J2]_{浮}$ = 11　1100　(1)000000

所以，$E_x = 1110$，$M_x = 00\ (1).\ 111000$ ，$E_{y1} = 1010$，$M_y = 00(1).000000$，$E_{y2} = 1010$，$M_y = 11(1).000000$

$E_k = 1100$，$M_K = 00\ (1).\ 111000$ ，$E_{J1} = 1100$，$M_{J1} = 00(1).000000$，$E_{J2} = 1100$，$M_{J2} = 11(1).000000$

尾数 M 中小数点前面有三位，前两位为数符，表示双符号，第三位加了括号，是隐藏位“1”。

没有附加位时的计算：

(1)　X+Y1

$[\Delta E]_{补} = [E_x]_{移} + [-[E_{y1}]_{移}]_{补} \pmod{2^n} = 1110 + 0110 = 0100$

$\Delta E = 4$，根据对阶规则可知需要对 y1 进行对阶，结果为：$E_{y1} = E_x = 1110$，$M_{y1} = 000.000100$

尾数相加：$M_b = M_x + M_{y1} = 001.\ 111000 + 000.000100 = 001.111100$，两位符号相等，数值部分最高位为 1，不需要进行规格化，所以最后结果为：E=1110，M=00(1).111100，即 $(31/32) \times 2^7$

(2)　X+Y2

$[\Delta E]_{补} = [E_x]_{移} + [-[E_{y2}]_{移}]_{补} \pmod{2^n} = 1110 + 0110 = 0100$;

$\Delta E = 4$，根据对阶规则可知需要对 y2 进行对阶，结果为：$E_{y2} = E_x = 1110$，$M_{y2} = 111.111100$

尾数相加：$M_b = M_x + M_{y2} = 001.\ 111000 + 111.111100 = 001.110100$，两位符号相等，

数值部分最高为 1，不需要进行规格化，所以最后结果为：E=1110，M=00(1).110100，即$(29/32)\times 2^7$

（3） K+J1

$[\Delta E]_{补}=[E_K]_{移}+[-[E_{J1}]_{移}]_{补} \pmod{2^n}=1100+0100=0000$;

$\Delta E=0$，根据对阶规则可知不需要进行对阶。

尾数相加：$M_b=M_K+M_{J1}=001.111000+001.000000=010.111000$，两位符号不等，说明尾数溢出，需要进行右规，最后结果为：E=1101，M=00(1).011100，即$(23/32)\times 2^6$

（4） K+J2

$[\Delta E]_{补}=[E_K]_{移}+[-[E_{J2}]_{移}]_{补} \pmod{2^n}=1100+0100=0000$;

$\Delta E=0$，根据对阶规则可知不需要进行对阶。

尾数相加：$M_b=M_K+M_{J2}=00\ 1.111000+111.000000=000.111000$，两位符号相等，数值部分最高位为 0，需要进行左规，所以最后结果为：E=1011，M=00(1).110000，即$(7/8)\times 2^4$

如果有两位附加位精度上会有提高，在对阶的时候要注意小数点后就不是 6 位，而是 8 位，最后两位为保护位和舍入位。但是由于本题 6 位尾数已经足够，再加 2 位附加位，其结果是一样的。

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：

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

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：$00.000000110...000+11.000001010...000=11.000000100\ldots000$

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：$M_b=11.000000100\ldots0$

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。

# 第四次作业

3. 假定某计算机中有一条转移指令，采用相对寻址方式，共占两个字节，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个字节。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和操作数各是多少?

（1）操作数采用变址寻址，变址寄存器的内容为 12，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=000CH+1200H=120CH，操作数未知。

（2） 有效地址 EA=(1200H)=12FCH，操作数为 38B8H。

（3） 有效地址 EA=1200H，操作数为 12FCH。

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、单地址和零地址三类。若二地址指令有 k2 条，无地址指令有 k0 条，则单地址指令最多有多少条?

参考答案：

设单地址指令有 k1 条，则 $((24 – k2) \times 26 – k1) \times 26 = k0$，所以 $k1= (24 – k2) \times 26 – k0/26$

7. 某计算机字长 16 位，每次存储器访问宽度 16 位，CPU 中有 8 个 16 位通用寄存器。现为该机设计指令系统，要求指令长度为字长的整数倍，至多支持 64 种不同操作，每个操作数都支持 4 种寻址方式：立即（I）、寄存器直接（R）、寄存器间接（S）和变址（X），存储器地址位数和立即数均为 16 位，任何一个通用寄存器都可作变址寄存器，支持以下 7 种二地址指令格式（R、I、S、X 代表上述四种寻址方式）：RR 型、RI 型、RS 型、RX 型、XI 型、SI 型、SS 型。请设计该指令系统的 7 种指令格式，给出每种格式的指令长度、各字段所占位数和含义，并说明每种格式指令需要几次存储器访问?

参考答案：

指令格式可以有很多种，只要满足以下的要求即可。

操作码字段：6 位；寄存器编号：3 位；直接地址和立即数：16 位；变址寄存器编号：3 位；总位数是 16 的倍数。

指令格式例 1：

第 1 个部分（指示指令格式）只需要 3 位就可以满足 7 种指令格式的要求，取 4 位主要是满足总长度为字长的倍数。

指令格式例 2：分别在每个操作数前面用 2 位指定其寻址方式

寻址方式字段(2 位)----00：立即；01：寄直；10：寄间；11-变址

10.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$v0 中。假定每个数组有 2500 个字，其数组下标为 0 到 2499。两个数组的基地址分别存放在$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。因此，指令总条数为：5+2500×(3+2500×6+2)=37512505，

其中：add,addi 和 sll 的指令条数为：4+2500×(2+2500×3+1)=18757504

lw 和 bne 的指令条数为： 1+2500×(1+2500×3+1)=18755001

所以：程序执行的时间为：(2GHzclock 的 clock time=1/2G=0.5ns)

(18757504×1+18755001×2) × 0.5ns = 28133753ns≈0.028s

# 第五次作业

3. 右图30给出了某CPU内部结构的一部分，MAR和MDR直接连到存储器总线（图中省略）。在两个总线之间的所有数据传送都需经过算术逻辑部件ALU。ALU可实现的部分功能及其控制信号如下：

MOVa：F=A； MOVb：F=B；

a+1：F=A+1； b+1：F=B+1

a-1：F=A-1； b-1：F=B-1

其中A和B是ALU的输入，F是ALU的输出。假定JSR（转子指令）指令占两个字，第一个字是操作码，第二个字给出子程序的起始地址，返回地址保存在主存的栈中，用SP（栈指示器）指向栈顶，按字编址，每次从主存读取一个字。请写出读取并执行JSR指令所要求的控制信号序列（提示：当前指令地址在PC中）。

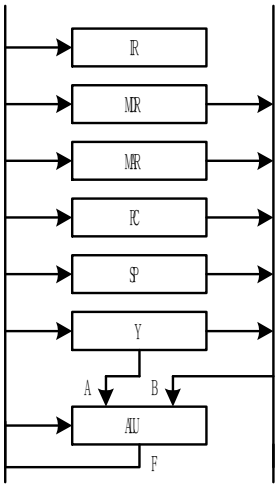



图3

参考答案：

**假定采用同步方式（若为异步，则只需在read和Write后加一个等待信号WMFC)**

分三个阶段：

1. 取指令操作码：PCout, MOVb, MARin
   Read, b+1, PCin （假设PC输出到总线后，下个时钟周期只要没有其他信号输出到总线，该信号一直有效）
   MDRout, MOVb, IRin
2. 取子程序首址：PCout, MOVb, MARin
   Read, b+1, Yin (返回地址送Y中)
   MDRout, MOVb, **PCin**(子程序首址送PC中)
3. 保存返址至栈：SPout, MOVb, MARin
   Yout, MOVb, MDRin
   Write, SPout, b-1, SPin

(注：若按最长的存储访问时间作为CPU时钟周期，则上述每个阶段都需三个时钟周期)

能否用更少的时钟周期完成上述功能？不能！以下是另一种方式）

1. 取指令操作码：PCout, MOVb, MARin
   Read, b+1, Yin
   MDRout, MOVb, IRin
2. 取子程序首址：Yout, MOVb, MARin
   Read, a+1, Yin （用b+1也行）
   MDRout, MOVb, PCin
3. 保存返址至栈：SPout, MOVb, MARin
   Yout, MOVb, MDRin
   Write, SPout, b-1, Spin

4. 假定某计算机字长16位，标志寄存器Flag中的ZF、NF和VF分别是零、负和溢出标志，采用双字节定长指令字。假定Bgt (大于零转移) 指令的第一个字节指明操作码和寻址方式，第二个字节为偏移地址Imm8，其功能是：

若(ZF+(NF ⊕ VF)=0) 则 PC=PC+2+Imm8 否则 PC=PC+2

（1） 该计算机的编址单位是什么？

（2） 画出实现Bgt指令的数据通路。

参考答案：

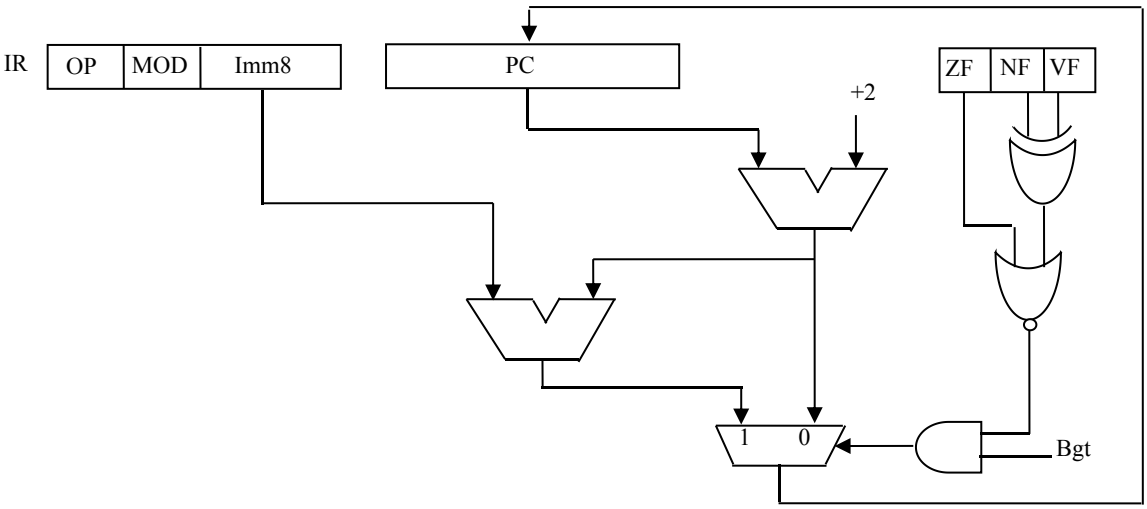
（1）该计算机的编址单位是字节。

因为 PC 的增量是 2，且每条指令占 2 个字节，所以编址单位是字节。

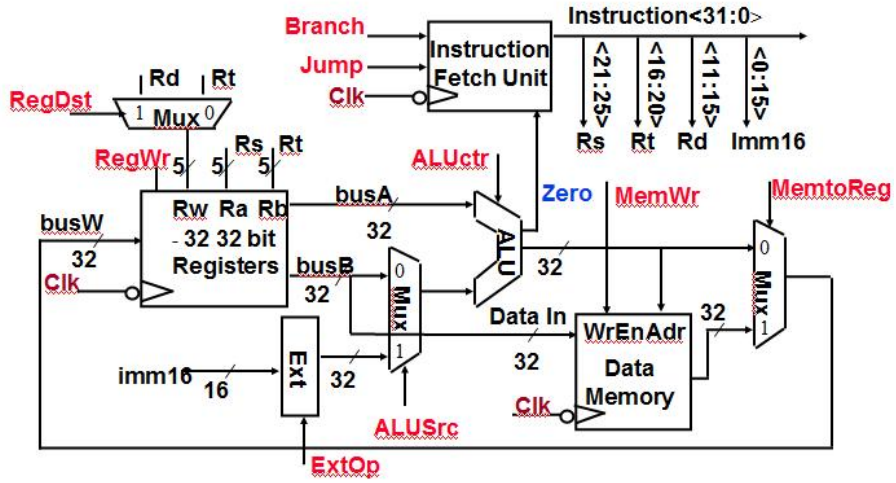
（2）实现Bgt指令的数据通路如下

根据“大于”条件判断表达式，可以看出该bgt指令实现的是带符号整数比较。因为无符号数比较时，其判断表达式中没有溢出标志OF。偏移地址Imm8为补码表示，转移目标地址可能在bgt指令之前，也可能在bgt指令之后。计算转移目标地址时，偏移量为Imm8，范围为-128～127，故转移目标地址的范围是PC+2+(-128)～PC+2+127

如果偏移量为Imm8x2，转移目标地址的范围是PC+2+(-128×2)～PC+2+127×2，其实意味着相对于bgt指令的前127条指令到后128条指令之间。



5. 假定图 5.16 所示单周期数据通路对应的控制逻辑发生错误，使得控制信号 RegWr、RegDst、ALUSrc、Branch、MemWr、ExtOp、R-type、MemtoReg 中某一个在任何情况下总是为 0，则该控制信号为 0 时哪些指令不能正确执行？要求分别讨论。




参考答案：

| | 总是0 | 总是1 |
|---|---|---|
| RegWr | 则所有需写结果到寄存器的指令（如：R-Type指令、load指令等）都不能正确执行，因为寄存器不发生写操作 | 不需写结果到寄存器的指令可能会出错（如store,分支,转移指令等） |
| RegDst | 则所有R-Type指令都不能正确执行，因为目的寄存器指定错误 | 所有非R-Type指令都不能正确执行 |
| ALUSrc | 则立即数运算类指令和访问存储器的指令都不能正确执行，因为立即数imm16送不到ALU中。 | 所有R型指令中具有rt寄存器的指令都不能正确指令 |
| Branch | Branch指令可能出错，因为永远不会发生转移 | 非Branch指令都出错,因为下条指令的地址计算错误 |
| MemWr | Store指令不能正确执行，因为存储器不能写入所需数据 | 非Store指令都会出错,因为存储器内会写入错误数据 |
| ExtOp | 需要符号扩展的指令（如Beq、lw/sw,addiu等）发生错误 | 必须0扩展的指令会出错(比如ori) |
| R-type | 所有具有func字段的指令（R型指令）不能正确执行。 | 所有非R型指令中需要在ALU中运算的指令不能正确执行。 |
| MemtoReg | 从存储器取数的指令不能正确执行。因此不能通过多路转换器送到寄存器组的写端口。 | 所有在ALU中运算后需要写入寄存器的指令都不能正确执行，包括移位指令 |

6. 假定图 5.16 所示单周期数据通路对应的控制逻辑发生错误，使得控制信号 RegWr、RegDst、ALUSrc、Branch、MemWr、ExtOp、R-type、MemtoReg 中某一个在任何情况下总是为 1，则该控制信号为 1 时哪些指令不能正确执行？要求分别讨论。

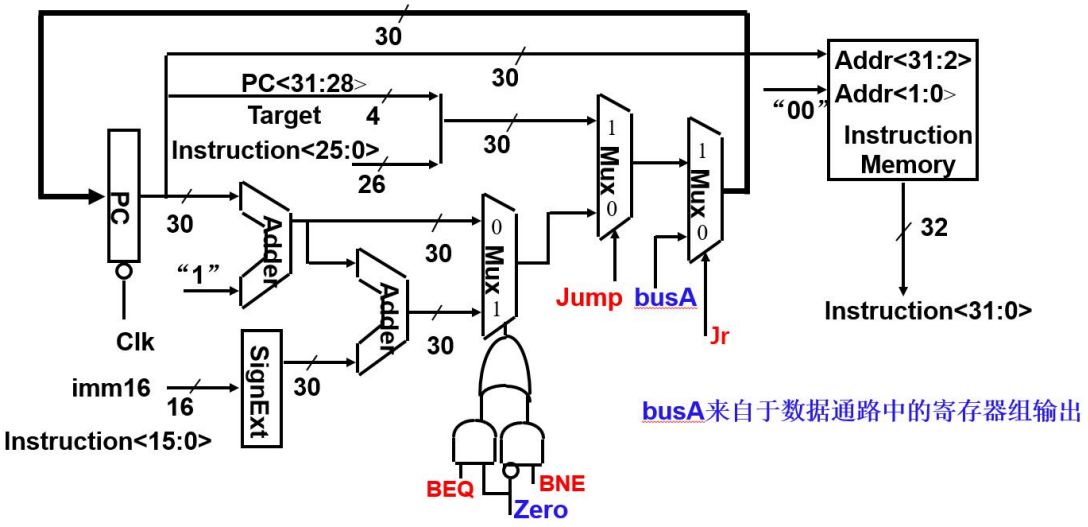
参考答案：见第5题的表格.

**补充题 1**

在课件介绍的 7 条指令基础上，增加 I 型指令 bne（不等转移）和 R 型指令 jr（寄存器跳转）两条指令后，画出取指部件的逻辑结构图。

参考答案

设置控制信号 BEQ 和 BNE，BEQ 对应相等转移指令，BNE 对应不等转移指令。原来的控制信号 Branch 取消。也可以保留 Branch 信号，另外增加一个区分相等转移和不等转移的指令的控制信号 EorNE。设置控制信号 Jr 对应寄存器跳转指令 jr




**补充题 2**

在单周期 MIPS 处理器的数据通路中，将各指令执行时所需要的控制信号的取值填入下表中。

| 指令 | RegDst | ALUSrc | ALUctr | MemWr | ExtOP | RegWr | MemtoReg |
|---|---|---|---|---|---|---|---|
| addu $10,$11,$12 | | | | | | | |
| addi $10, $15, AAH | | | | | | | |
| sw $10, 100($11) | | | | | | | |
| bne $10, $15, 1234H | | | | | | | |

addu $10, $11, $12 //R[rd] <= R[rs] + R[rt], 在 ALU 中执行不带溢出的加法 addu。
addi $10, $15, AAH //R[rt] <= R[rs] + SignExt[Imm16], ALU 执行具有溢出判断的加法 add。

| 指令 | RegDst | ALUSrc | ALUctr | MemWr | ExtOP | RegWr | MemtoReg |
|---|---|---|---|---|---|---|---|
| addu $10,$11,$12 | 1 | 0 | Addu | 0 | x | 1 | 0 |
| addi $10, $15, AAH | 0 | 1 | Add | 0 | 1 | 1 | 0 |
| sw $10, 100($11) | x | 1 | Addu | 1 | 1 | 0 | x |
| bne $10, $15, 1234H | x | 0 | Subu | 0 | x | 0 | x |

# 第六次作业

6. 以下指令序列中，哪些指令对发生数据相关？假定采用“取指、译码/取数、执行、访存、写回”五段流水线方式，那么不用“转发”技术的话，需要在发生数据相关的指令前加入几条 nop 指令才能使这段程序避免数据冒险？如果采用“转发”是否可以完全解决数据冒险？不行的话，需要在发生数据相关的指令前加入几条 nop 指令才能使这段程序不发生数据冒险？

```
add   $s3, $s1, $s0
sub   $t2, $s0, $s3
lw    $t1, 0($t2)
add   $t1, $t1, $t2
```

**参考答案：**

发生数据相关的有：第 1 和 2 间关于$s3、第 2 和 3 间关于$t2、第 2 和 4 间关于$t2、第 3 和 4 间关于$t1。

不进行“转发”处理的话，需要分别在第 2、3、4 条指令前加三条 nop 指令才能避免数据冒险。而通过“转发”可以避免 1 和 2、2 和 3、2 和 4 间的数据相关；但第 3 和 4 间是 load-use 数据相关，所以无法用“转发”消除冒险，因此，需在第 4 条指令前加入一条 nop 指令。

寄存器写口和寄存器读口分别安排在一个时钟周期的前、后半个周期内独立工作呢？

——2、3、4 条之前分别插入 2 条 nop 就可以

7. 假定以下 MIPS 指令序列在图 7.18 所示的流水线数据通路中执行：

```
addu  $s3, $s1, $s0
subu  $t2, $s0, $s3
lw    $t1, 0($t2)
add   $t3, $t1, $t2
add   $t1, $s4, $s5
```

请问：（1）上述指令序列中，哪些指令的哪个寄存器需要转发，转发到何处？

（2）上述指令序列中，是否存在 load-use 数据冒险？

（3）第 5 周期结束时，各指令执行状态是什么？哪些寄存器的数据正被读出？哪些寄存器将被写入？

**参考答案：**

（1）发生数据相关的有：第 1 和 2 间关于$s3、第 2 和 3 间关于$t2、第 2 和 4 间关于$t2、第 3 和 4 间关于$t1。通过“转发”可以避免 1 和 2、2 和 3、2 和 4 间的数据相关；

（2）第 3 和 4 间是 load-use 数据相关，所以无法用“转发”消除冒险。

（3）第五个时钟内各条指令的执行情况如下：

指令 1 在“WB”阶段，控制信息等在 MEM/WB.Reg 中，$s3 正在被写，结束时写完

指令 2 在“MEM”阶段，控制信息等在 EX/MEM.Reg 中。sub 指令在该阶段进行的是空操作；在转发检测单元中，因为流水段寄存器 Ex/Mem 中的目的寄存器 RegRd 为$t2，流水段寄存器 ID/Ex 中的源寄存器 Rs 也为$t2，同时，流水段寄存器 Ex/Mem 中的 RegWr 控制信号为 1，所以检测到转发条件满足，因而，此时，sub 指令在上一个时钟周期中的执行结果（在流水段寄存器 Ex/Mem 中的 ALU 输出结果）正被回送到 ALU 的输入端；结束时转发完成

指令 3 在“EXE”阶段，ALU 正在执行“add”操作，进行地址运算，ALU 输出结果将被写入流水段寄存器 Ex/Mem 中；结束时运算完成。控制信息等在 ID/EX.Reg 中,正在检测是否 loaduse 冒险

指令 4 在“ID/REG”阶段，指令在 IF/ID.Reg 中，$t1 和$t2 正在被读出。在 **load-use 冒险检测单元**中，因为流水段寄存器 IF/ID 中源操作数寄存器 Rs 为$t1，流水段寄存器 ID/Ex 中目的操作数寄存器 Rt 也为$t1，同时，因为上条指令是 lw，故流水段寄存器 ID/Ex 中的 MemRead 控制信号为 1，所以在该阶段**检测到 load-use 冒险条件满足**，此时，需要进行 load-use 冒险处理，在流水线中插入一个“气泡”，**将指令的执行阻塞一个时钟周期**。包括以下三个步骤：① 将流水段寄存器 ID/Ex 中的控制信号全部清 0，以保证第 4 条指令被阻塞一个时钟周期执行；② 将流水段寄存器 IF/ID 中的指令维持不变，以保证第 4 条指令重新译码后执行；③ 将 PC 的值维持不变，以保证根据 PC 的值重新取出第 5 条指令。结束时完成上述工作。

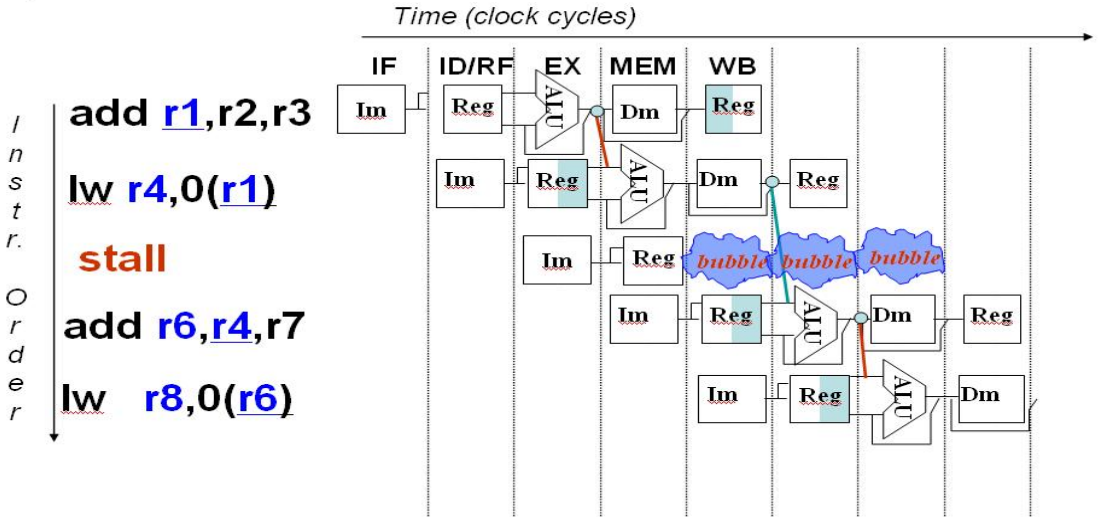
指令 5 在“IF”阶段，指令正被读出。结束时已送到流水段寄存器 IF/ID 的输入端。因为之前发生了 load-use 数据冒险，所以该指令将在随后的第 6 个时钟周期内重新被读出。

8. 假定有一个程序的指令序列为“lw, add, lw, add, …”。add 指令仅依赖它前面的 lw 指令，而 lw 指令也仅依赖它前面的 add 指令，寄存器写口和寄存器读口分别在一个时钟周期的前、后半个周期内独立工作。请问：（1）在带转发的五段流水线中执行该程序，其 CPI 为多少?

（2）在不带转发的五段流水线中执行该程序，其 CPI 为多少?

参考答案：（1）因为 lw 指令和 add 指令之间存在一个 load- use 数据冒险，所以每个 lw 指令和 add 指令之间要有一次流水线阻塞。而 add 指令和 lw 指令之间的数据冒险可通过数据转发解决。即：CPI 为 1.5

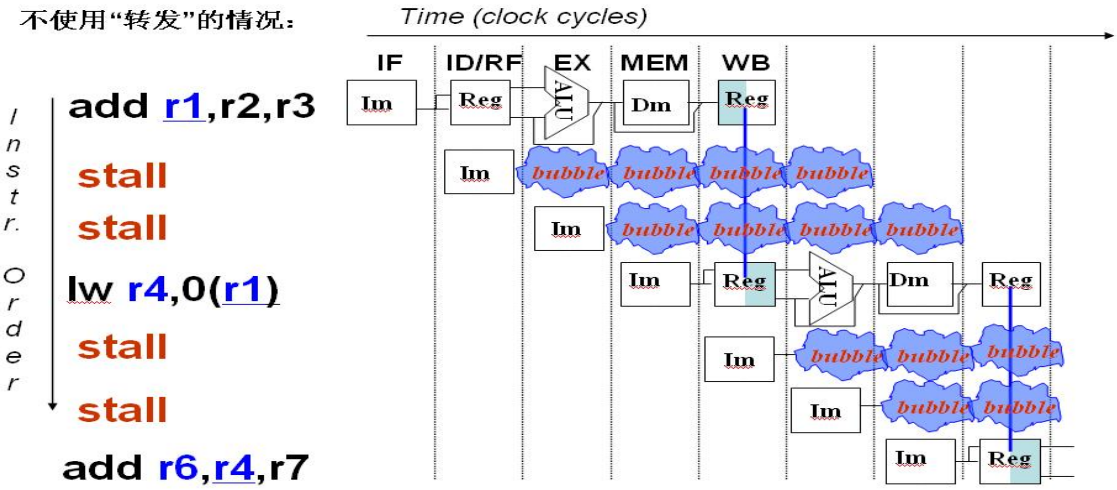
使用“转发”的情况：




使用“转发”时，只有lw指令后需要一次阻塞！

（2）如果没有转发，而寄存器写口和寄存器读口分别在一个时钟周期的前、后半个周期内工作，则在每条 lw 指令和 add 指令之间将会有两个阻塞，这样每条指令相当于都要有三个时钟才能完成。即：CPI 为 3

不使用“转发”的情况：




通过寄存器写口/读口分别安排在前半/后半周期，在不使用“转发”时使得每条指令之间只要阻塞两次就可解决！

9. 假定在一个带转发功能的五段流水线中执行以下程序段，则可以怎样调整以下指令序列使其性能达到最好？

参考答案：

```
1    lw    $2, 100($6)
2    add   $2, $2, $3
3    lw    $3, 200($7)
4    add   $6, $4, $7
5    sub   $3, $4, $6
```

```
6        lw     $2, 300($8)
7        beq    $2, $8, Loop
```

因为采用“转发”技术，所以，只要对 load-use 数据冒险进行指令序列调整。从上述指令序列来看，第 1 和第 2 条指令、第 6 和第 7 条指令之间存在 load-use 数据冒险，所以，可将与第 2 和第 3 条指令无关的第 4 条指令插入第 2 条指令之前；将无关的第 5 条指令插入第 7 条指令之前。调整顺序后的指令序列如下（粗体部分为变换了位置的指令）。

```
lw     $2, 100($6)
add        $6, $4, $7
add        $2, $2, $3
lw       $3, 200($7)
lw         $2, 300($8)
sub        $3, $4, $6
 beq       $2, $8, Loop
```

10. 在一个采用“取指、译码/取数、执行、访存、写回”的五段流水线中，若检测结果是否为“零”的操作在执行阶段进行，则分支延迟损失时间片（即分支延迟槽）为多少？以下一段 MIPS 指令序列中，在考虑数据转发的情况下，哪些指令执行时会发生流水线阻塞？各需要阻塞几个时钟周期？

```
1   loop:       add    $t1, $s3, $s3
 2              add    $t1, $t1, $t1
 3              add    $t1, $t1, $s6
 4              lw     $t0, 0($t1)
 5              bne    $t0, $s5, exit
 6              add    $s3, $s3, $s4
 7              j      loop
 8    exit:
```

参考答案：

若检测操作在执行阶段进行，则分支延迟损失时间片（即分支延迟槽）为 2。

分析：发生数据相关的是：第 1 和第 2 条指令之间关于$t1，第 2 和第 3 条指令之间关于$t1，第 3 和第 4 条指令之间关于$t1，第 4 和第 5 条指令之间关于$t0，以及第 6 和第 1 条指令之间关于$s3。此外，第 5 和第 7 条指令的执行都会发生控制相关。

对于数据冒险，如果不采用“转发”，而是简单地通过加入 nop 指令来避免冒险的话，那么应该在第 2、3、4、5 条指令前各加两条 nop 指令，以消除数据相关；对于第 6 条和第 1

条指令之间的数据相关，则可通过在第 7 条“j loop”指令后面加一条或两条 nop 指令消除（这样同时还能解决第 7 条“j loop”指令的控制冒险）；

此处，第 2、3、4 条指令所需的操作数可通过“转发”得到，无需加 nop 指令。第 5 条 bne 指令所需的操作数$t0 是 load-use 冒险，不能用“转发”解决问题，需要在第 5 条指令前加一条 nop 指令，或通过硬件将第 5 条指令的执行阻塞一个时钟周期。

j 指令如果在**译码阶段**就根据译码结果计算跳转目标地址，那么 j 指令后面指令会被阻塞 1 个时钟周期，若在**执行阶段**计算，则要阻塞 2 个时钟周期。

---------- 其它

11. 假设数据通路中各主要功能单元的操作时间为：存储单元：200ps；ALU 和加法器：100ps；寄存器堆读口或写口：50ps。程序中指令的组成比例为：取数 25%、存数 10%、ALU52%、分支 11%、跳转 2%。假设时钟周期取存储器存取时间的一半，MUX、控制单元、PC、扩展器和传输线路等的延迟都忽略不计，则下面的实现方式中，哪个更快？快多少？

    （1）单周期方式：每条指令在一个固定长度的时钟周期内完成；
    （2）多周期方式：每类指令时钟数：取数-7，存数-6，ALU-5，分支-4，跳转-4；
    （3）流水线方式：取指 1、取指 2、取数/译码、执行、存取 1、存取 2、写回 7 段流水线；没有结构冒险；数据冒险采用“转发”技术处理；load 指令与后续各指令之间存在依赖关系的概率分别 1/2、1/4、1/8、…；分支延迟损失时间片为 2，预测准确率为 75%；不考虑异常、中断和访问失效引起的流水线冒险。

参考答案：

单周期：存储器操作变为两个时钟周期后，其数据通路的时钟周期不变，为 **600ps**

多周期：CPI=0.25x7+0.10x6+0.52x5+0.11x4+0.02x4 = 5.47

存储器操作变为两个时钟周期后，多周期数据通路的时钟周期为 100ps，

故一条指令的执行时间为 100x5.47=**547ps**

流水线：存储器操作变为两个时钟周期后，其流水线包含了 7 个阶段.

对于 ALU 指令，随后的数据相关指令都可通过转发解决，故 CPI=1

对于 Store 指令，不会发生数据冒险，故 CPI=1

对于 Jump 指令，总要等到译码结束才能确定转移地址，故 CPI=3（取指 1,2，译码）

对于 beq，若预测正确，则为 1 个周期，若预测错误，则为 3 个周期，故

CPI=1/4x3+3/4x1=1.5

对于 load，随后第一条则为 3 个（阻塞 2 个）周期；随后第二条则为 2 个（阻塞 1 个）周期，以后的指令都不需要阻塞，故 CPI=1/2x3+1/4x2+2/8x1=2.25

**平均 CPI 为：2.25x25%+1x10%+1x52%+1.5x11%+3x2%=1.41**

所以， 1 条指令的执行时间为 1.41x100=**141(ps)**

12.假设有一段程序的核心模块中有五条分支指令，该模块将会被执行成千上万次，在其中一次执行过程中，五条分支指令的实际执行情况如下（T：Taken；N：not Taken）。

分支指令 1（B1）：T – T – T。

分支指令 2（B2）：N – N – N – N。

分支指令 3（B3）：T – N – T – N – T – N。

分支指令 4（B4）：T – T – T – N – T。

分支指令 5（B5）：T – T – N – T – T – N – T。

假定各个分支指令在每次模块执行过程中实际执行情况都一样，并且动态预测时，每个分支指令都有各自的预测表项，每次执行时的初始预测位都相同。请给出以下几种预测方案的预测准确率。

（1）静态预测，总是预测转移 （Taken）。

（2）静态预测，总是预测不转移（not Taken）。

（3）一位动态预测，初始预测转移 （Taken）。

（4）二位动态预测，初始预测弱转移（Taken）。

**【分析解答】**

预测准确率=预测正确次数/总预测次数×100%。以下 R 表示正确预测次数，W 表示错误预测次数。

（1）B1: R-3, W-0； B2: R-0, W-4； B3: R-3, W-3； B4: R-4, W-1； B5: R-5, W-2；60%

（2）B1: R-0, W-3； B2: R-4, W-0； B3: R-3, W-3； B4: R-1, W-4； B5: R-2, W-5；40%

（3）B1: R-3, W-0； B2: R-3, W-1； B3: R-1, W-5； B4: R-3, W-2； B5: R-3, W-4；52%

（4）B1: R-3, W-0； B2: R-3, W-1； B3: R-3, W-3； B4: R-4, W-1； B5: R-5, W-2；72%

| B4 | T | T | T | N | T |
|---|---|---|---|---|---|
| 静态预测，总是预测转移 | 对 | 对 | 对 | 错 | 对 |
| 总是预测不转移 | 错 | 错 | 错 | 对 | 错 |
| 一位动态预测，初始预测转移 | 对 | 对 | 对 | 错(N) | 错 |

| | | | | | |
|---|---|---|---|---|---|
| 二位动态预测，初始预测弱转移 | 对(强转移) | 对(强转移) | 对(强转移) | 错(弱转移) | 对 |

# 第七次作业

**3. 已知某主机主存的最大寻址空间为 4GB,按字节编址。假定用64M × 8位的具有 8 个位平面的 DRAM 芯片构成容量位 512MB、传输宽度为 64 位的内存条。问：**

(1)每个内存条需要多少个这样的 DRAM 芯片?

(2)构建容量为 2GB 的主存时，需要几个内存条?

(3)主存地址共多少位?其中几位用于 DRAM 芯片内地址?哪几位时片内行地址?哪几位是片内列地址?哪几位用于选择芯片?

参考答案:

(1)每个内存条需要多少个这样的 DRAM 芯片?

因为 DRAM 芯片容量为64M × 8 位 = 64MB

所以每个内存条需要512MB ÷ 64MB = 8 个

很多同学这里计算结果为 64，把内存条的容量算作了 512M × 64位了

(2)构建容量为 2GB 的主存时，需要几个内存条?

因为每个内存条容量位 512MB,所以需要2GB ÷ 512MB = 4 个

(3)主存地址共多少位?其中几位用于 DRAM 芯片内地址?哪几位是片内行地址?哪几位是片内列地址?哪几位用于选择芯片?

因为4GB = $2^{32}B$,按字节编址则地址需要 32 位

芯片内总共 64MB=$2^{26}$B=$2^{13}$*$2^{13}$B 所以芯片内部需要 26 位，其中低 13 位为行地址，高 13 位为列地址。即第 0-12 位行地址，第 13-25 位为列地址。

（这里很多同学计算芯片内部的时候是按照内存条容量 512MB 来算的，用 14 位选择行，15 位选择列）

主存总共有 32 位地址，所以需要 32-26=6 位，即第 26-31 位用于选择芯片。

（因为前面的计算错误，所以导致这里的结果也有误。）

这是用高 6 位做片选，存储单元地址在一个芯片内连续分配。如果用最低 6 位做片选，就要用多体交叉分配地址，这种结构较适合总线传输带宽是多字节的情况，比如 64 位宽度，可以由 8 个芯片同时读出放到总线上，而这 8 个芯片读出的 8 个字节单元应该形成一个连续的地址范围。

**6.某计算机中已配有 0000H～7FFFH 的 ROM 区域，现在再用 8K×4 位的 RAM 芯片形成 32K×8 位的存储区域，CPU 地址总线为 A0-A15，数据总线为 D0-D7，控制信号为 R/W#（读/写）、MREQ#（访存）。要求说明地址译码方案，并画出 ROM 芯片、RAM 芯片与 CPU 之间的连接图。假定上述其他条件不变，只是 CPU 地址线改为 24 根，地址范围 000000H～007FFFH 为 ROM 区，剩下的所有地址空间都用 8K×4 位的 RAM 芯片配置，则需要多少个这样的 RAM 芯片?**

**参考答案:**

**CPU 地址线共 16 位，故存储器地址空间为 0000H～FFFFH，其中，8000H～FFFFH 为 RAM 区，共 $2^{15}$=32K 个单元，其空间大小为 32KB，故需 8K×4 位的芯片数为 32KB/8K×4 位= 4×2 = 8 片。**

**因为 ROM 区在 0000H～7FFFH，RAM 区在 8000H～FFFFH，所以可通过最高位地址 $A_{15}$ 来区分，当 $A_{15}$ 为 0 时选中 ROM 芯片；为 1 时选中 RAM 芯片，此时，根据 $A_{14}$ 和 $A_{13}$ 进行译码，得到 4 个译码信号，分别用于 4 组字扩展芯片的片选信号。(图略，**

可参照图 4.15)

若 CPU 地址线为 24 位, ROM 区为 000000H ~ 007FFFH, 则 ROM 区大小为 32KB, 总大小为 16MB=$2^{14}$KB=512×32KB，所以 RAM 区大小为 511×32KB，共需使用 RAM 芯片数为 511×32KB/8K×4 位=511×4×2 个芯片。

11. 假定某机主存空间大小1GB，按字节编址。cache的数据区（即不包括标记、有效位等存储区）有64KB，块大小为128字节，采用直接映射和全写（write-through）方式。请问：

（1）主存地址如何划分？要求说明每个字段的含义、位数和在主存地址中的位置。

（2）cache的总容量为多少位？

参考答案：

（1）主存空间大小为 1GB，按字节编址，说明主存地址为 30 位。cache 共有 64KB/128B=512 行，因此，行索引（行号）为 9 位；块大小 128 字节，说明块内地址为 7 位。因此，30 位主存地址中，高 14 位为标志（Tag）；中间 9 位为行索引；低 7 位为块内地址。

（2）因为采用直接映射，所以cache中无需替换算法所需控制位，全写方式下也无需修改（dirty）位，而标志位和有效位总是必须有的，所以，cache总容量为 512×(128×8+14+1)=519.5K位。

18. 假设某计算机的主存地址空间大小为 64MB，采用字节编址方式。其 cache 数据区容量为 4KB，采用 4 路组相联映射方式、LRU 替换和回写（write back）策略，块大小为 64B。请问：

（1）主存地址字段如何划分？要求说明每个字段的含义、位数和在主存地址中的位置。

（2）该 cache 的总容量有多少位？

（3）若 cache 初始为空，CPU 依次从 0 号地址单元顺序访问到 4344 号单元，重复按此序列共访问 16 次。若 cache 命中时间为 1 个时钟周期，缺失损失为 10 个时钟周期，则 CPU 访存的平均时间为多少时钟周期？

参考答案：

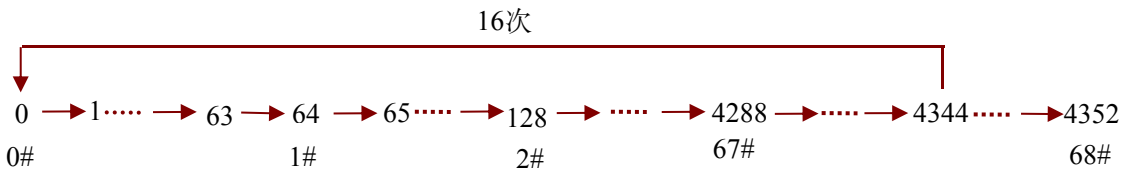
（1）cache 的划分为：4KB = $2^{12}$B = $2^4$ 组×$2^2$ 行/组×$2^6$ 字节/行，所以，cache 组号（组索引）占 4 位。

主存地址划分为三个字段：高 16 位为标志字段、中间 4 位为组号、最低 6 位为块内地址。

即主存空间划分为：64MB = $2^{26}$B = $2^{16}$ 组群×$2^4$ 块/组群×$2^6$ 字节/块

（2）cache 共有 64 行，每行中有 16 位标志、1 位有效位、1 位修改(dirty)位、2 位 LRU 位，以及数据 64B。故总容量为 64×(16+1+1+2+64×8)=34048 位。

(3) 因为每块为 64B, CPU 访问的单元范围为 0 ~ 4344, 共 4345 个单元, 4345/64=67.89, 所以 CPU 访问的是主存前 68 块（第 0 ~ 67 块），也即 CPU 的访问过程是对前 68 块连续访问 16 次，总访存次数为 16×4345 = 69520。



cache 共有 16 组，每组 4 行，采用 LRU 算法的替换情况如下图所示：

| | 第0行 | 第1行 | 第2行 | 第3行 |
|---|---|---|---|---|
| 0组 | 0/64/48 | 16/0/64 | 32/16 | 48/32 |
| 1组 | 1/65/49 | 17/1/65 | 33/17 | 49/33 |
| 2组 | 2/66/50 | 18/2/66 | 34/18 | 50/34 |
| 3组 | 3/67/51 | 19/3/67 | 35/19 | 51/35 |
| 4组 | 4 | 20 | 36 | 52 |
| … | … | … | … | … |
| … | … | … | … | … |
| 15组 | 15 | 31 | 47 | 63 |

根据图中所示可知，第一次循环的每一块只有第一次未命中，其余都命中；以后 15 次循环中，有 20 块的第一字未命中，其余都命中。所以命中率 p 为 (69520–68–15×20)/69520 = 99.47%

平均访存时间为：Hit Time + (1–p) × Miss Penalty

=1+10×(1–p) = 1+0.0053×10 = 1.053 个时钟周期

**24.** 假定一个虚拟存储系统的虚拟地址为40位，物理地址为36位，页大小为16KB，按字节编址。若页表中有有效位、存储保护位、修改位、使用位，共占4位，磁盘地址不在页表中，则该存储系统中每个进程的页表大小为多少？如果按计算出来的实际大小构建页表，则会出现什么问题？

参考答案：

因为每页大小有 16KB，所以虚拟页数为 $2^{40}$B/16KB=$2^{(40-14)}$=$2^{26}$ 页。

物理页面和虚拟页面大小相等，所以物理页号的位数为 36–14=22 位。

页表项位数为：有效位+保护位+修改位+使用位+物理页号位数=4+22=26 位。

为简化页表访问，每项大小取 32 位。因此，每个进程的页表大小为：$2^{26}$×32b=256MB。

如果按实际计算出的页表大小构建页表，则页表过大而导致页表无法一次装入内存。

**25.** 假定一个计算机系统中有一个TLB和一个L1 data cache。该系统按字节编址，虚拟地址16位，物理地址12位；页大小为128B，TLB为四路组相联，共有16个页表项；L1 data cache采用直接映射方式，块大小为4B，共16行。在系统运行到某一时刻时，TLB、页表和L1 data cache中的部分内容（用十六进制表示）如下：

| 组号 | 标记 | 页框号 | 有效位 | 标记 | 页框号 | 有效位 | 标记 | 页框号 | 有效位 | 标记 | 页框号 | 有效位 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 03 | – | 0 | 09 | 0D | 1 | 00 | – | 0 | 07 | 02 | 1 |
| 1 | 03 | 2D | 1 | 02 | – | 0 | 04 | – | 0 | 0A | – | 0 |
| 2 | 02 | – | 0 | 08 | – | 0 | 06 | – | 0 | 03 | – | 0 |
| 3 | 07 | – | 0 | 63 | 0D | 1 | 0A | 34 | 1 | 72 | – | 0 |

(a) TLB（四路组相联）：四组、16 个页表项

| 虚页号 | 页框号 | 有效位 |
|---|---|---|
| 00 | 08 | 1 |
| 01 | 03 | 1 |
| 02 | 14 | 1 |
| 03 | 02 | 1 |
| 04 | – | 0 |
| 05 | 16 | 1 |
| 06 | – | 0 |
| 07 | 07 | 1 |
| 08 | 13 | 1 |
| 09 | 17 | 1 |
| 0A | 09 | 1 |
| 0B | – | 0 |
| 0C | 19 | 1 |
| 0D | – | 0 |
| 0E | 11 | 1 |
| 0F | 0D | 1 |

(b) 部分页表：（开始 16 项）

| 行索引 | 标记 | 有效位 | 字节 3 | 字节 2 | 字节 1 | 字节 0 |
|---|---|---|---|---|---|---|
| 0 | 19 | 1 | 12 | 56 | C9 | AC |
| 1 | 15 | 0 | – | – | – | – |
| 2 | 1B | 1 | 03 | 45 | 12 | CD |
| 3 | 36 | 0 | – | – | – | – |
| 4 | 32 | 1 | 23 | 34 | C2 | 2A |
| 5 | 0D | 1 | 46 | 67 | 23 | 3D |
| 6 | – | 0 | – | – | – | – |
| 7 | 16 | 1 | 12 | 54 | 65 | DC |
| 8 | 24 | 1 | 23 | 62 | 12 | 3A |
| 9 | 2D | 0 | – | – | – | – |
| A | 2D | 1 | 43 | 62 | 23 | C3 |
| B | – | 0 | – | – | – | – |
| C | 12 | 1 | 76 | 83 | 21 | 35 |
| D | 16 | 1 | A3 | F4 | 23 | 11 |
| E | 33 | 1 | 2D | 4A | 45 | 55 |
| F | 14 | 0 | – | – | – | – |

(c) L1 data cache：直接映射，共 16 行，块大小为 4B

请回答下列问题：

（1）虚拟地址中哪几位表示虚拟页号？哪几位表示页内偏移量？虚拟页号中哪几位表示 TLB 标记？哪几位表示 TLB 索引？

（2）物理地址中哪几位表示物理页号？哪几位表示页内偏移量？

（3）主存（物理）地址如何划分成标记字段、行索引字段和块内地址字段？

（4）CPU 从地址 067AH 中取出的值为多少？说明 CPU 读取地址 067AH 中内容的过程。

参考答案：

（1）16 位虚拟地址中低 7 位为页内偏移量，高 9 位为虚页号；虚页号中高 7 位为 TLB 标记，低 2 位为 TLB 组索引。

（2）12 位物理地址中低 7 位为页内偏移量，高 5 位为物理页号。

（3）12 位物理（主存）地址中，低 2 位为块内地址，中间 4 位为 cache 行索引，高 6 位为标记。

（4）地址 067AH=0000 0110 0111 1010B，所以，虚页号为 0000011 00B，映射到 TLB 的第 00 组，将 0000011B=03H 与 TLB 第 0 组的四个标记比较，虽然和其中一个相等，但对应的有效位为 0，其余都不等，所以 TLB 缺失，需要访问主存中的慢表。直接查看 0000011 00B =00CH 处的页表项，有效位为 1，取出物理页号 19H=11001B，和页内偏移 111 1010B 拼接成物理地址：11001 111 1010B。根据中间 4 位 1110 直接找到 cache 第 14 行(即：第 E 行)，有效位为 1，且标记为 33H=110011B，正好等于物理地址高 6 位，故命中。根据物理地址最低两位 10，取出字节 2 中的内容 4AH=01001010B。

# 第八次作业

4. 假定一个程序重复完成将磁盘上一个 4KB 的数据块读出，进行相应处理后，写回到磁盘的另外一个数据区。各数据块内信息在磁盘上连续存放，并随机地位于磁盘的一个磁道上。磁盘转速为 7200RPM，平均寻道时间为 10ms，磁盘最大数据传输率为 40MBps，磁盘控制器的开销为 2ms，没有其他程序使用磁盘和处理器，并且磁盘读写操作和磁盘数据的处理时间不重叠。若程序对磁盘数据的处理需要 20000 个时钟周期，处理器时钟频率为 500MHz，则该程序完成一次数据块“读出-处理-写回”操作所需的时间为多少？每秒钟可以完成多少次这样的数据块操作？

**参考答案：**

**平均旋转等待时间：(1s / (7200/60)) / 2 ≈ 8.33/2 ≈ 4.17ms**

**因为块内信息连续存放，所以数据传输时间：4KB / 40MBps ≈ 0.1ms**

**平均存取时间 T ： 寻道时间 + 旋转等待时间 + 数据传输时间**

**= 10ms + 4.17ms + 0.1ms = 14.27ms**

**读出时间(写回时间)：14.27ms+2ms = 16.27ms**

**数据块的处理时间：20000 / 500MHz ≈ 0.04ms**

因为数据块随机存放在某个磁道上，所以，每个数据块的“读出-处理-写回”操作时间都是相同的，**所以完成一次操作时间：16.27ms x 2+0.04ms = 32.58ms**

**每秒中可以完成这样的数据块操作次数：1s / 32.58ms ≈ 30 次**

8. 某终端通过 RS-232 串行通信接口与主机相连，采用起止式异步通信方式，若传输速率为 1200 波特，采用两相调制技术。通信协议为 8 位数据、无校验位、停止位为 1 位。则传送一个字节所需时间约为多少？若传输速度为 2400 波特，停止位为 2 位，其他不变，则传输一个字节的时间为多少？

**参考答案：**

**采用两相调制技术，所以，波特率=比特率，且每个字符都有一个起始位，**

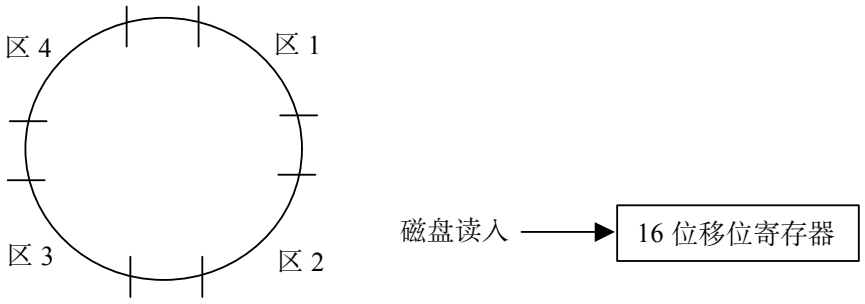
**(a) 1200 波特时，一个字符共占：1+8+1=10 位**

**所以一个字符所需时间约为：10x(1÷1200)=8.3 毫秒**

**(b) 2400 波特时，一个字符共占：1+8+2=11 位**

**所以一个字符所需时间约为：11x(1/2400)=4.6 毫秒**

10. 假设有一个磁盘，每面有 200 个磁道，盘面总存储容量为 1.6 兆字节，磁盘旋转时间为 25ms/圈，每道有 4 个区，每两个区之间有一个间隙，磁头通过每个间隙需 1.25ms。(1) 问：从该磁盘上读取数据时的最大数据传输率是多少（单位为字节/秒）？ (2) 假如有人为该磁盘设计了一个与计算机之间的接口，如下图所示，磁盘每读出一位，串行送入一个移位寄存器，每当移满 16 位后向处理器发出一个请求交换数据的信号。在处理器响应该请求信号并读取移位寄存器内容的同时，磁盘继续读出一位一位数据并串行送入移位寄存器，如此继续工作。已知处理器在接到请求交换的信号以后，最长响应时间是 3 微秒，这样设计的接口能否正确工作？若不能则应如何改进？




**参考答案：** **每个磁道的存储容量：$1.6x10^6 / 200=8000B$**

**→ 每个区容量为：8000 / 4 = 2000B**

**而当仅读取一个区内数据的时候，转过一个区只需要： (25-1.25x4) / 4 = 5ms**

**所以最大数据传输率：$2000B / 5ms = 4x10^5$ 字节/秒**

---

**因此，传送 1 位的最短时间为：$1 / (8x4x10^5) =0.31\mu s$ <<3μs**

**因此，当处理器经过 3μs 来读取移位寄存器中的数据时，磁盘已经读出了新的数据位，并将原先请求被读的移位寄存器中的数据冲刷掉了。所以这样的设计接口不能正确工作。**

**改进方法：传送 16 位数据需 0.31 x 16 = 5μs > 3μs**

**所以可以增加一个 16 位数据缓冲器。当 16 位移位寄存器装满后，先送入数据缓冲寄存器，在读出下一个 16 位数据期间（5μs），上次读出的 16 位数据从数缓器中被取走（3μs）。**

**16. （在第 3 版教材中为第 11 题）**

参考答案：

设备与主机传 1000 字节所需时间：1000B/50KBps=0.01953s=19.53ms

（1）采用定时查询方式

查询状态并处理一个字节所需时间: (1000+60)/1GHz=1.06us

传送 1000 字节所需 CPU 时间：1.06us*1000=1.06ms

故时间占比：1.06/19.53=0.0543=5.43%

（2）独占查询方式

此种方式下 CPU 全部用于 I/O，CPU 所用时间：19.53ms；时间占比 100%

（3）采用中断方式

传 1000 字节需 1000 次中断，所需 CPU 总时间：

1000*（2+1200）/1GHz=1.202ms

时间占比：1.202/19.53=6.15%

（4）传 1000 字节 CPU 占用时间只是 DMA 初始化和后处理的 2000 个时钟周期：

2000/1GHz=2us

时间占比：2us/19.53ms=2/19.53*$10^{-3}$=0.1024*$10^{-3}$=0.01024%

（5）传输速率提高到 5MBps 后传 1000 字节所需时间：1000/5MBps=1000/(5*1024*1024)=0.19073ms

=190.73us

对于前述方式（1），CPU 传 1000 字节需 1.06ms 大于设备传输时间 0.19073ms，CPU 处理速度跟不上设备传输速度，故该方式不可行。

对于前述方式（2），查询和传送 1000 字节所需时间大于 1.06ms，故该方式也是不可行。

对于前述方式（3），传 1000 字节所需时间 1.202ms 大于设备传输时间 0.19073ms，故不可行。

对于方式（4），传 1000 字节所需 CPU 时间 2us，小于设备传输时间 0.19073ms，故该方式可行。

CPU 时间占比：2/190.73=1.05%

（6）对于方式（1）：查询状态并处理一个字节所需时间: (1000+60)/1GHz=1.06us,

故时间占比：1.06us/0.02ms=0.053=5.3%

对于方式（2）：100%

对于方式（3）：1 次中断传 1 字节所需 CPU 时间：1.202ms/1000=1.202us

故时间占比：1.202us/0.02ms=6.01%