第 1 章 计算机系统概论

1. 若某程序编译后生成的目标代码由 A、B、C、D 四类指令组成，它们在程序中所占比例分别为 40%、20%、15%、25%。已知 A、B、C、D 四类指令的 CPI 分别为 1、2、2、2。现需要对程序进行编译优化，优化后的程序中 A 类指令数减少了一半，而其他指令数量未发生变化。假设运行该程序的计算机 CPU 主频为 500MHz。完成下列各题：

（1）优化前后程序的 CPI 各为多少？

（2）优化前后程序的 MIPS 各为多少?

（3）通过上面的计算结果你能得出什么结论？

（1）优化前程序 CPI＝1*40%+2*20%+2*15%+2*25% = 1.6

优化后程序 CPI=1*(20% / (20%+20%+15%+25%)) +2*((20%+15%+25%)/(20%+20%+15%+25%)) =1.75

（2）优化前 MIPS ＝ f / (CPI * 10^6) = 312.5

优化后 MIPS ＝ f / (CPI * 10^6) = 285.7

（3）优化后，程序执行时间变短；但程序的 CPI 却增加，MIPS 减少。说明不能简单地只通过 CPI 或者 MIPS 来衡量计算机性能。

考虑下面两种处理器。P1 的时钟频率为 4GHz，平均 CPI 为 0.9，需要执行 5.0×10^9 条指令；P2 的时钟频率为 3GHz，平均 CPI 为 0.75，需要执行 1.0×10^9 条指令。

（1）一个常见的错误是，认为时钟频率最高的计算机具有最高的性能。请用 P1 和 P2 来验证这一说法。

（2）另一个错误是，认为执行指令最多的处理器需要更多的 CPU 时间。考虑 P1 执行 1.0×10^9 条指令序列所需的时间，P1 和 P2 的 CPI 不变，计算一下 P2 用同样的时间可以执行多少条指令？

（3）一个常见的错误是用 MIPS（每秒百万条指令）来比较两台不同的处理器的性能，并认为 MIPS 最大的处理器具有最高的性能。请用 P1 和 P2 验证这一说法。

（4）一个常见的错误是用 MFLOPS（每秒百万条浮点指令）来比较两台不同的处理器的性能，并认为 MFLOPS 最大的处理器具有最高的性能。请用 P1 和 P2 验证这一说法。假定 P1 和 P2 上执行的指令有 40%的浮点指令。

1. CPU 执行时间=指令数×CPI×时钟周期=指令数×CPI/时钟频率

   CPU 执行时间(P1) = 5 × 109 × 0.9 / (4 × 109) = 1.125 s

   CPU 执行时间(P2) = 109 × 0.75 / (3 × 109) = 0.25 s

   CPU 执行时间 (P1) > CPU 执行时间 (P2)，故性能(P1) < 性能(P2)

2. CPU 执行时间(P1) = 1.0 × 109 × 0.9 / (4 × 109) = 0.225 s

指令数（P2）= CPU 执行时间×时钟频率 / CPI=0.225×3 × 109 / 0.75=0.9 × 109

3. MIPS = 指令数/ (CPU 执行时间× 106) = 时钟频率 / (CPI × 106)

MIPS（P1）= 4 × 109 × 10—6/0.9 = 4.44 × 103

   MIPS(P2) = 3 × 109 × 10—6/0.75 = 4.0 × 103

   MIPS(P1) > MIPS(P2)，但性能(P1) < 性能(P2)

4. MFLOPS=浮点操作的数目 / （执行时间×（1× 106））

   MFLOPS（P1）= 5 × 109 × 40% /（1.125 s × 106）=1.78× 10^3

   MFLOPS（P1）= 1.0 × 109 × 40% /（0.25 s× 106）=1.6× 10^3

   MFLOPS(P1) > MFLOPS(P2), 但性能(P1) < 性能(P2)

习题 2-第 2 章 计算机中的数据表示

已知数的补码表示形式，求数的真值。

$[x]_{补}=0.10010$，　$[x]_{补}=1.10010$，　$[x]_{补}=1.11111$，

$[x]_{补}=1.00000$，　$[x]_{补}=0.10001$，　$[x]_{补}=1.00001$

- $[x]_{补}=0.10010$　$x=0.10010$，$x=2^{-1}+2^{-4}=0.5625$
- $[x]_{补}=1.10010$，$x=-0.01110$，$x=-(2^{-2}+2^{-3}+2^{-4})=-0.4375$
- $[x]_{补}=1.11111$，$x=-0.00001$，$x=-2^{-5}=-0.03125$
- $[x]_{补}=1.00000$，$x=-1.00000$，$x=-1$
- $[x]_{补}=0.10001$，$x=0.10001$，$x=2^{-1}+2^{-5}=0.53125$
- $[x]_{补}=1.00001$，$x=-0.11111$，$x=-(2^{-1}+2^{-2}+2^{-3}+2^{-4}+2^{-5})=-0.96875$

用 IEEE754 32 位单精度浮点数标准表示下列十进制数（要求写出详细的过程）。

(a) $-6\frac{5}{8}$

(b) -8.25

(a)

$-110.101=-1.10101\times 2^2$

阶码 = 2 + 127 = 129= 1000 0001

尾数= 10101

1 1000 0001 1010 1000 0000 0000 0000 000

=1100 0000 1101 0100 0000 0000 0000 0000＝（C0D40000）16

(b) $-1000.01=-1.000\ 01\times 2^3$

阶码=3+127=130=1000 0010

尾数=00001

1 1000 0010 0000 1000 0000 0000 0000 000 =C104000H

求下列单精度浮点数对应的十进制数。（要求：写出详细求解过程）

(a) 43940000H

(b) C640 0000H

（1）

4394 0000H = 0100 0011 1001 0100 0000 0000 0000

数符 S=0，阶码=100 0011 1 = 8*16+7=135，E=135-127=8

尾数= 00101，M=1.00101，所以表示数：$1.00101*2^8$ = 100101000

十进制数= 296。

（2）C640 0000H = 1100 0110 0100 0000 0000 0000 0000 0000

S=1，$E=10001100_2-127_{10}$ = 10001100 − 01111111 = 0000 1101 = 13

$M=1.1_2=1.5$

因此，浮点数的值为-1.5×2^{13}。

1 3.4 已知x和y，用变形补码计算$x+y$，并判断结果是否溢出。

x=0.11010，y=0.101110

x=0.11101，y=－0.10100

x=－0.10111，y=－0.11000

正确答案：

	（1） x=0.11010 y=0.101110	（2） x=0.11101 y=－0.10100	（3） x=－0.10111 y=－0.11000
$X_{补}$	00.11010	00.11101	11.01001
$Y_{补}$	00.101110	11.01100	11.01000
$[X+Y]_{补}$	01.100010	00.01001	10.10001
是否溢出	正溢出	未溢出	负溢出

2. 用原码一位乘法计算 x×y=？ $x=-0.11111$，$y=0.11101$

$|x|_{原}$= 00.11111，$|y|_{原}$= 0.11101

迭代次数	步骤	部分积	乘数
0	初始值	00. 00000	11101
1	y_n=1，+ |x|	+ 00.11111	
		00.11111	11101
	部分积与乘数逻辑右移1位	00.01111	11110
2	y_n=0，+ 0	+ 00.00000	
		00.01111	11110
	部分积与乘数逻辑右移1位	00.00111	11111
3	y_n=1，+ |x|	+ 00.11111	
		01.00110	11111
	部分积与乘数逻辑右移1位	00.10011	01111
4	y_n=1，+ |x|	+ 00.11111	
		01.10010	01111
	部分积与乘数逻辑右移1位	00.11001	00111
5	y_n=1，+ |x|	+ 00.11111	
		01.11000	00111
	部分积与乘数逻辑右移1位	00.11100	00011

$P_f = x_0 \oplus y_0 = 1 \oplus 0 = 1$，所以，x×y= - 0.1110000011

3. 采用 IEEE754 单精度浮点数格式计算下列表达式的值。0.625 + (－12.25)

$0.625 = (0.101)_2 = 1.01 \times 2^{-1}$

$-12.25 = (-1100.01)_2 = -1.10001 \times 2^3$

① 对阶：$1.01 \times 2^{-1} = 0.000101 \times 2^3$

② 尾数相加：0.000101 -1.10001 = -1.011101

③ 结果规格化：尾数符合1.X的形式，无须规格化。

④ 判溢出：阶码正常，无溢出。

③ 舍入处理：尾数无须舍入。

$0.625 + (-12.25) = -1.011101 \times 2^3 = (1\ 10000010\ 01110100000000000000000)_2 = (C13A0000)_{16}$

4 3.7 用补码一位乘法计算x×y=？

$x=-0.011010$，$y=-0.011101$

正确答案：

$[X]_{补}=11.100110$，$[-X]_{补}=00.011010$，$[Y]_{补}=1.100011$

迭代次数	步骤	部分积	乘数
0	初始值	00.000000	11000110
1	$y_n y_{n+1}$ =10，+$[-X]_{补}$	+ 00.011010	
		00.011010	11000110
	部分积与乘数算术右移1位	00.001101	01100011
2	$y_n y_{n+1}$ =11，+0	+ 00.000000	
		00.001101	01100011
	部分积与乘数算术右移1位	00.000110	10110001
3	$y_n y_{n+1}$ =10，+$[X]_{补}$	+ 11.100110	
		11.101100	10110001
	部分积与乘数算术右移1位	11.110110	01011000
4	$y_n y_{n+1}$ =00，+0	+ 00.000000	
		11.110110	01011000
	部分积与乘数算术右移1位	11.111011	00101100
5	$y_n y_{n+1}$ =00，+0	+ 00.000000	
		11.110110	01011000
	部分积与乘数算术右移1位	11.111101	10010110
6	$y_n y_{n+1}$ =10，+$[-X]_{补}$	+ 00.011010	
		00.010111	10010110
	部分积与乘数算术右移1位	00.001011	11001011
7	$y_n y_{n+1}$ =11，+0	+ 00.000000	
	最后一步不右移，得乘积	00.001011	110010

所以，$[x\times y]_{补}=0.001011110010$

5.用原码不恢复余数法计算 x÷y =？ x = －0.10101，y = 0.1101

$|y|_{补}=00.11011$，$[-|y|]_{补}=11.00101$，$|x|_{原}=00.10101$

迭代次数	步骤	余数R	商Q
0	初始值	00.10101	0.00000
1	第一次减|y|比较，+$[-|y|]_{补}$	+ 11.00101	
		11.11010	
	R<0，商上0		0.00000
	R、Q同步左移1位	11.10100	0.0000
2	加|y|比较，+$[|y|]_{补}$	+ 00.11011	
		00.01111	
	R>0，商上1		0.00001
	R、Q同步左移1位	00.11110	0.0001
3	减|y|比较，+$[-|y|]_{补}$	+ 11.00101	
		00.00011	
	R>0，商上1		0.00011
	R、Q同步左移1位	00.00110	0.0011
4	减|y|比较，+$[-|y|]_{补}$	+ 11.00101	
		11.01011	
	R<0，商上0		0.00110
	R、Q同步左移1位	10.10110	0.0110
5	加|y|比较，+$[|y|]_{补}$	+ 00.11011	
		11.10001	
	R<0，商上0		0.01100
	R、Q同步左移1位	11.00010	0.1100
6	加|y|比较，+$[|y|]_{补}$	+ 00.11011	
		11.11101	
	R<0，商上0		0.11000
	最后一步不左移 但余数<0，需恢复余数	+ 00.11011	
		00.11000	

商=$(X_0 \oplus Y_0).11000$ =1.11000

余数= - $(00.11*2^{-5})$=1.0000011

1 给定2个16位半精度格式IEEE754编码的浮点数（1位符号位，5位指数，10位尾数）：$A=2.6125\times10^{1}$，$B=4.150390625\times10^{-1}$。计算A+B，假定有1位保护位，1位舍入位，1位粘贴位，并采用向最靠近的偶数舍入模式。要求：给出详细的浮点加法步骤，并说明保护位、舍入位、粘贴位的值分别为多少。

正确答案：

解答：

$2.6125\times 10^{1} = 26.125 = 11010.001 = 1.1010001000\times 2^{4}$

$4.150390625\times10^{-1} = 0.4150390625 = .0110101001 = 1.10101001\times2^{-2}$

① 对阶：$1.10101001\times2^{-2} = 0.0000011010\,1001 \times 2^{4}$

② 尾数相加：

$$\begin{array}{rl} & 1.1010001000 \\ +\ & 0.0000011010\ 10\ 01 \quad \text{（保护位1，舍入位0，粘贴位1）} \\ \hline = & 1.1010100010\ 10\ 1 \quad \text{（保护位1，舍入位0，粘贴位1）} \end{array}$$

③ 规格化：无须规格化

④ 判溢出：无溢出，$1.1010100010\,101\times2^{4}$

⑤ 采用向最靠近的偶数舍入模式，得 $1.1010100011 \times 2^{4}$

$A+B=1.1010100011\times2^{4}=11010.100011=26.546875=2.6546875\times10^{1}$

2 3.5 已知x和y，用变形补码计算$x-y$，并判断结果是否溢出。

x=0.11011，y=0.11101

x=0.10111，y=0.11110

x=−0.11111，y=−0.11001

正确答案：

	（1） x=0.11011，y=0.11101	（2） x=0.10111，y=0.11110	（3） x=−0.11111，y=−0.11001
$X_{补}$	00.11011	00.10111	11.00001
$Y_{补}$	00.11101	00.11110	11.00111
$[X-Y]_{补}$	11.11110	11.11001	11.11010
是否溢出	未溢出	未溢出	未溢出

习题 4-第 4 章 指令系统

5. 6 假设某计算机的指令长度固定为 16 位，具有双操作数，单操作数和无操作数 3 类指令，每个操作数地址规定用 6 位表示。

（1）若操作码字段不固定，现已设计出 m 条双操作数指令，n 条无操作数指令，在此情况下，这台计算机最多可以设计出多少条单操作数指令？

（2）若操作码字段不固定，当双操作数指令取最大数，且在此基础上，单操作数指令条数也取最大值，试计算这 3 类指令最多可拥有多少条指令？

(1) 双操作数指令中，操作数 6 位，则操作码=16-6-6=4 位，最多可表示双操作数指令 2^4 条；而实际有 m 条双操作数指令，则剩余状态数=2^4 – m=16-m。

单操作数指令中，操作数 6 位，最多可表示单操作数指令（16-m）*2^6 条；设实际有 x 条单操作数指令，则剩余状态数=（16-m）*2^6 –x。

无操作数指令可有 （（16-m）*2^6 –x）*2^6 条；即：（（16-m）*2^6 –x）*2^6 = n，推导出：单操作数指令条数=(16-m)*64- n/64 条。

(2) 也就是剩余状态数仅为 1 时，双操作数指令=2^4 – 1 =15 条，单操作数= 1*2^6 -1= 63 条，无操作数指令=64 条

5.7 设相对寻址的转移指令占三个字节，第一个字节是操作码，第二个字节是相对位移量(补码表示)的低 8 位，第三个字节是相对位移量(补码表示)的高 8 位，每当 CPU 从存储器取一个字节时，便自动完成(PC)+1 →PC:

（1）若 PC 当前值为 256(十进制)，要求转移到 290(十进制)，则转移指令第二、三字节的机器代码是什么(十六进制)?

（2）若 PC 当前值为 128(十进制)，要求转移到 110(十进制)，则转移指令第二、三字节的机器代码又是什么(十六进制)?

(1) PC=256+3，新 PC=290，故：290-（256+3）=31= 001FH 转移指令第 2 字节为 1FH，第 3 字节为 00H。

(2) PC=128+3，故 110-（128+3）= - 21 = - 15H = FFEBH， 转移指令第 2 字节为 EBH，第 3 字节为 FFH。

5.8 计算机的指令格式包括操作码 OP、寻址方式特征位 I 和形式地址 D 等三个字段，其中 OP 字段 6 位，寻址方式特征位字段 I 为 2 位，形式地址字段 D 为 8 位。I 的取值与寻址方式的对应关系为：

I=00：变址；

I=01：用变址寄存器 X1 进行变址；

I=10：用变址寄存器 X2 进行变址；

I=11：相对寻址。

设(PC)=1234H，(X1)=0037H，(X2)=1122H，以下四条指令均采用上述格式，请确定这些指令的有效地址：

(1)4420H (2)2244H (3)1322H (4)3521H

(1) 指令 4420H = 010001 00 0010 0000B，即寻址方式特征位字段=00，故 EA=D=20H

(2) 寻址方式特征位字段=10，用变址寄存器 X2 进行变址，故 EA=(X2)+D=1122H+44H=1166H

(3) 寻址方式特征位字段=11，相对寻址，故 EA=PC+2+D=1234H+2H+22H=1258H

(4) 寻址方式特征位字段=01，用变址寄存器 X1 进行变址，故 EA=(X1)+D=0037H+21H=0058H

5.9 某计算机 A 有 60 条指令，指令的操作码字段固定为 6 位，从 000000-111011，该机器的后续机型 B 中需要增加 32 条指令，并与 A 保持兼容。

1) 试采用操作码扩展方法为计算机 B 设计指令操作码。

2) 计算机 B 中操作码的平均长度。

解：(1) 可以采用扩展操作码方式，保留 4 个状态用于扩展指令，将操作码扩展到地址字段，只需要占用地址字段 3 位即可表示 32 条新指令。（$4\times2^n=32$ ➔ n=3）

(2)平均长度为=7.04 位。（$60\times6+32\times9$）÷（60+32）=7.04 位

以下 MIPS 指令代表什么操作？写出它的 MIPS 汇编指令格式。0000 0000 1010 1111 1000 0000 0010 0000

解：汇编指令为：add $s0,$a1,$t7

操作码 op=000000，扩展操作码 funct=100000，add 指令

rs=00101 = $a1，rt=01111 = $t7，rd=10000 = $s0

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

15~12	11~060806		05~0002~00	
OP	Ms	Rs	Md	Rd
	源操作数		目的操作数	

转移指令采用相对寻址方式，相对偏移量用补码表示，寻址方式定义见下表。

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

注：（X）表示存储器地址 X 或寄存器 X 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？ 该计算机最多有多少个通用寄存器？2) 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？3) 转移指令的目标地址范围是多少？4) 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，则汇编语言为“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中的内容会改变？改变后的内容是什么？

正确答案：

解：

(1) 该指令系统最多可有16 条指令；该机最多有8 个通用寄存器；

(2) 主存地址空间大小为 128KB = 217B，按字编址，所以主存地址为16位，故存储器地址寄存器MAR需16位；
存储器按字编址，所以每一个单元为16位，存储器数据寄存器MDR至少需16 位。

(3) 由于寄存器是16位，转移指令的目标地址范围是0000H～FFFFH（0～2^{16}-1）。

(4) 汇编语句“add (R4), (R5) +”，对应的机器码为0010 0011 0001 0101B=2315H。
该指令执行后，寄存器R5 和存储单元5678H 的内容会改变。执行后R5 的内容从5678H 变成5679H。存储单元5678H 中的内容变成该加法指令计算的结果5678H＋1234H=68ACH。

2 考虑如下的MIPS循环：

```
LOOP :  slt   $t2, $zero, $t1
        beq $t2, $zero, DONE
        subi $t1, $t1, 1
        addi $s2, $s2, 2
        j LOOP
DONE:
```

(1) 假设寄存器$t1的初始值为10，假设$s2初始值为0，则循环执行完毕时寄存器$s2的值是多少？

(2) 对于上面的循环体，写出等价的C代码。假定寄存器$s1、 $s2、$t1和$t2分别为整数A、B、i和temp。

（20分）

我的答案：

(1)LOOP总共循环十次，$s2的值每次都增加2，最后s2=20

(2)

```
while(i-->0){
    B+=2;
}
```

```
int fact (int n)
{    if (n < 1) return 1;
     else return (n * fact (n-1));
}
```

假定参数 n 存储在$a0 中，结果存储在$v0 中, 对应的部分 MIPS 代码如下，请补充完整！（50 分）

```
fact:
        [1]_____________________    # 调整栈指针用于存 2 项
        sw    $ra, 4($sp)           # 保存返回地址
        sw    $a0, 0($sp)           # 保存参数寄存器内容
        slti   $t0, $a0, 1          # n < 1 则$t0=1 否则=0
        beq   $t0, $zero, L1        # $t0=0 则跳转到 L1
        addi $v0, $zero, 1          # $t0=1 阶乘结果为 1→$v0
        addi $sp, $sp, 8            # 两项出栈
        jr   $ra                    # 返回
L1:    addi $a0, $a0, -1            #n=n-1
        [2]_____________________    # 递归调用 fact(n-1)
        [3]_____________________    # 取出之前的 n→$a0
        [4]_____________________    # 取出之前的返回地址→$ra
        addi $sp, $sp, 8            # 两项出栈
        mul   $v0, $a0, $v0         # 计算 n*fact(n-1)
       [5]_____________________     # 返回
```

第一空：addi $sp, $sp, -8

第二空：jal fact

第三空：lw $a0, 0($sp)

第四空：lw $ra, 4($sp)

第五空：jr $ra

假定 int 型变量 i 和 j，依次分配到寄存器$s3, $s4 中，假设 int 型数组 A 和 B 的基地址依次放置寄存器$s6 和 $s7 中，请把下面的 C 代码翻译成 MIPS 指令： B[8] = A[i] + A[j]。

```
sll $t1, $s3, 2
sll $t2, $s4, 2 #将 i,j 分别左移两位，得到 4*i,4*j,方便之后取数
add $t3, $s6, $t1
add $t4, $s6, $t2 #得到 A[i]与 A[j]对应的地址
lw $t1, 0($t3)
lw $t2, 0($t4) #将 A[i]与 A[j]对应的数值取到寄存器中
add $t1, $t1, $t2 #将 A[i]与 A[j]相加，存入 t1
sw $t1, 32($s7) #将结果存入 B[8]
```

习题 5-第 5 章 中央处理器

1. 假设图 6.25 所示的单周期 MIPS 处理器中，操作控制器输出某个控制信号发生了恒 0 故障，表 6.2 中的哪些指令会发生错误呢？如果是恒 1 故障呢？(1) RegWrite (2) RegDst (3) MemWrite

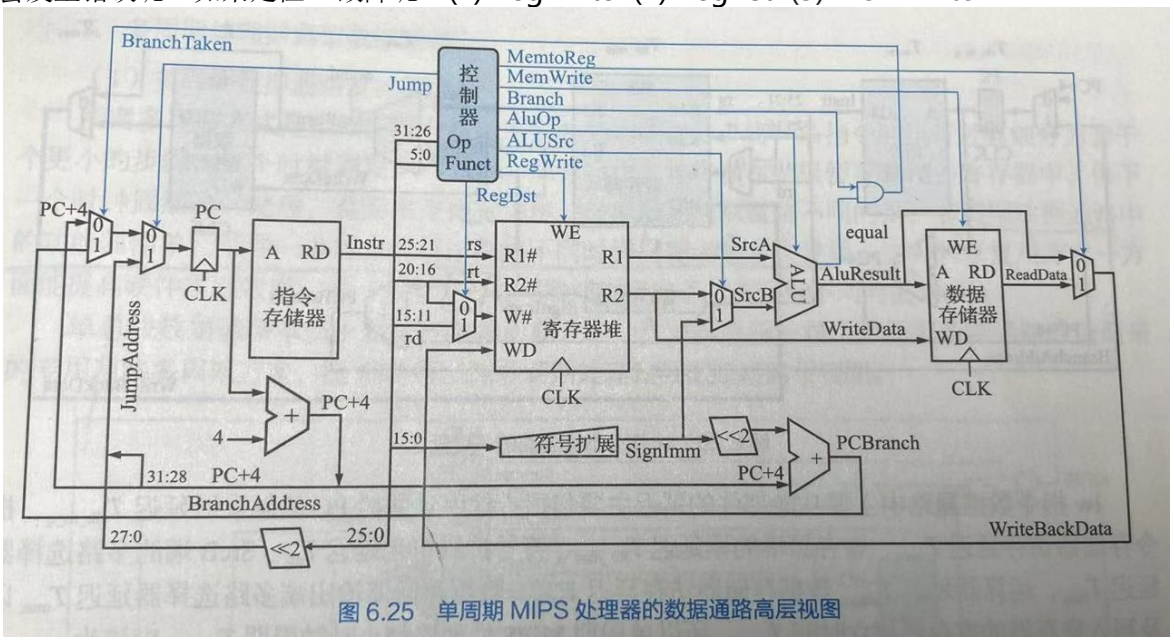

图 6.25 单周期 MIPS 处理器的数据通路高层视图

(1) RegWrite 恒 0 故障：lw、addi、add、slt 指令发生错误。
RegWrite 恒 1 故障：sw、beq、j 指令发生错误。
(2) RegDst 恒 0 故障：add、slt 指令错误。
RegDst 恒 1 故障：lw、addi 指令错误。
(3) MemWrite 恒 0 故障：sw 指令错误。
MemWrite 恒 1 故障：lw、beq、addi、add、slt、j 指令错误。

2.假设构成 CPU 的功能部件的时间延迟如表 6.21 功能部件时间延迟表 6.21 所示，试分别计算求单周期、多周期 MIPS 处理器的最小时钟周期和最大时钟频率，假设某 MIPS 程序包含 1000 亿条指令，其中 *lw*、*sw*、beq、R 型算术逻辑运算、I 型算术逻辑运算指令比例分别为 10%，10%，10%，50%，20%，试分别计算该程序在单周期 MIPS、多周期 MIPS 处理器上的 CPI 及执行时间。

功能部件	参数	延迟	功能部件	参数	延迟
寄存器延迟	$T_{clk\_to\_q}$	20 *ps*	运算器ALU	T_{alu}	90 *ps*
存储器读	T_{mem}	150 *ps*	多路选择器	T_{mux}	20 *ps*
寄存器堆读	T_{RF_read}	90 *ps*	寄存器建立时间	T_{setup}	10 *ps*

单周期 MIPS 处理器 CPI=1，Tcpu=530 s
多周期 MIPS 处理器 CPI=4，Tcpu=800 s

3. 请分析：如图 6.25 所示的单周期 MIPS 处理器，使其能够支持如下 MIPS 指令。试描述需要增加修改哪些数据通路和控制信号。
（1）addi rt,rs,imm16 指令功能：R[rt]← R[rs]+SignExt(imm16)
（2）ori rt, rs, imm16 指令功能：R[rt] ← R[rs] |ZeroExtend(imm16)

（1）不需修改，不需增加
（2）需增加一个同时支持符号扩展和零扩展的功能部件，且增加一个控制信号 ExtOp 进行选择进行符号扩展还是零扩展。

习题 6-第 6 章 指令流水线

1.气泡流水线执行下述程序，请给出类似图 7.18 的流水时空图，注意时空图最后一个时钟周期第 5 条指令进入 ID 段。

```
addi $s0, $s0, 4
lw   $s1, ($s0)
add  $s2, $s2, $s1
and  $s3, $s1, $s2
sub  $s4, $s2, $s2
```

CLKs	取指 IF	译码 ID	执行 EX	访存 MEM	写回 WB
1	addi $s0, $s0,4				
2	lw $s1, ($s0)	addi $s0, $s0,4			
3	add $s2, $s2, $s1	lw $s1, ($s0)	addi $s0, $s0,4		
4	add $s2, $s2, $s1	lw $s1, ($s0)	Bubble	addi $s0, $s0,4	
5	add $s2, $s2, $s1	lw $s1, ($s0)	Bubble	Bubble	addi $s0, $s0,4
6	and $s3, $s1, $s2	add $s2, $s2, $s1	lw $s1, ($s0)	Bubble	Bubble
7	and $s3, $s1, $s2	add $s2, $s2, $s1	Bubble	lw $s1, ($s0)	Bubble
8	sub $s4, $s2, $s2	add $s2, $s2, $s1	Bubble	Bubble	lw $s1, ($s0)
9	Next Instr	sub $s4, $s2, $s2	add $s2, $s2, $s1	Bubble	Bubble

2. 如果采用重定向流水线执行上述程序，请给出流水时空图？注意时空图最后一个时钟周期第 5 条指令进入 ID

CLKs	取指 IF	译码 ID	执行 EX	访存 MEM	写回 WB
1	addi $s0, $s0,4				
2	lw $s1, ($s0)	addi $s0, $s0,4			
3	add $s2, $s2, $s1	lw $s1, ($s0)	addi $s0, $s0,4		
4	and $s3, $s1, $s2	add $s2, $s2, $s1	lw $s1, ($s0)	addi $s0, $s0,4	
5	and $s3, $s1, $s2	add $s2, $s2, $s1	Bubble	lw $s1, ($s0)	addi $s0, $s0,4
6	sub $s4, $s2, $s2	and $s3, $s1, $s2	add $s2, $s2, $s1	Bubble	lw $s1, ($s0)
7	Next Instr	sub $s4, $s2, $s2	and $s3, $s1, $s2	add $s2, $s2, $s1	Bubble

3. 重定向流水线中所有分支跳转指令均在 EX 执行，假设无分支预测、无延迟槽技术，尝试计算下述程序的执行周期。

```
add $s0,$0,100        #  i=100
while_loop:
    beq $s0,$0,done   # while (i>0)
    addi $s0,$s0,-1   #  i=i-1
    j while_loop      # 继续循环
done:
```

程序执行周期数=508

4. 重定向流水线中所有分支跳转指令均在 EX 执行，假设采用动态分支预测技术，beq、j 指令都可以进行预测，计算最优情况下上述程序的执行周期。

程序执行周期数=310

5. 对于表 7.8 给出的 MIPS 指令流水线时间参数，如果可以优化流水线一个功能部件的关键延迟以提升处理器整体性能，应该选择哪个部件进行优化？如果这种优化与成本是线性关系，如何优化才能使得处理器性能达到最优，且成本最低。

$T_{min\_clk} = \max(T_{if\_max}, T_{id\_max}, T_{ex\_max}, T_{ex\_max}, T_{mem\_max}, T_{wb\_max})$，所以应该优化最慢的功能段，这里 ID 段最慢，应该优化其中的寄存器堆读延迟 T_{RF_read}=150ps，当这个时间延迟优化到 100 时，ID 段时延和 IF、EX、MEM 相同，再进一步优化没有意义，只会增加成本。

习题 7-第 7 章 存储系统

设 cache 的容量为 2^{14} 块，每块是一个 32 位字，主存容量是 cache 容量的 256 倍，其中有如下表所示数据(地址和数据均采用十六进制表示)。

地　址	数　据	地　址	数　据
000000	87568536	01FFFC	4FFFFC68
000008	87792301	FFFFF8	01BF2460
010004	9ABEFCD0		

将主存中这些数据装入到 cache 后，cache 各块中的数据内容及相应的标志是什么？

1)　全相联映射　2) 直接相联映射　3)四路组相联映射

1）全相联映射

每块是一个32位=4*8=2^2字节，故块内偏移地址：2位

Cache容量2^{14}块，而主存容量是cache容量的256倍，则主存容量=2^{14}*256块=2^{22}块

故：主存块地址：22位

主存地址	数　据	主存块地址（tag）
000000_{16} =0000 0000 0000 0000　0000 0000_2	87568536	0000 0000 0000 0000 0000 00_2 =000000_{16}
000008_{16}=0000 0000 0000 0000　0000 1000_2	87792301	0000 0000 0000 0000 0000 10_2 =000002_{16}
010004_{16}=0000 0001 0000 0000　0000 0100_2	9ABEFCD0	0000 0001 0000 0000 0000 01_2 =004001_{16}
$01FFFC_{16}$=0000 0001 1111 1111　1111 1100_2	4FFFFC68	0000 0001 1111 1111 1111 11_2=$007FFF_{16}$
$FFFFF8_{16}$=1111 1111 1111 1111　1111 1000_2	01BF2460	1111 1111 1111 1111 1111 10_2 =$3FFFFE_{16}$

cache行	标志	数据
0	000000	87568536
1	000002	87792301
2	004001	9ABEFCD0
3	007FFF	4FFFFC68
4	3FFFFE	01BF2460

2）直接相联映射

每块是一个32位=4*8=2^2字节，故块内偏移地址：2位

Cache容量2^{14}块，行索引：14位

而主存容量是cache容量的256倍，则主存容量=2^{14}*256块=2^{22}块

故：区地址tag：22-14=8位

主存地址	数　据	主存块地址（tag）
000000_{16} =0000 0000 0000 0000 0000 0000_2	87568536	0000 0000_2 =00_{16}
000008_{16}=0000 0000 0000 0000 0000 1000_2	87792301	0000 0000_2 =00_{16}
010004_{16}=0000 0001 0000 0000 0000 0100_2	9ABEFCD0	0000 0001=01_{16}
$01FFFC_{16}$=0000 0001 1111 1111 1111 1100_2	4FFFFC68	00000 0001_2=01_{16}
$FFFFF8_{16}$=1111 1111 1111 1111 1111 1000_2	01BF2460	1111 1111_2 =FF_{16}

cache行	标志	数据
0000	00	87568536
0002	00	87792301
0001	01	9ABEFCD0
3FFF	01	4FFFFC68
3FFE	FF	01BF2460

3）四路组相联映射

每块是一个32位=4*8=2^2字节，故块内偏移地址：2位

Cache容量2^{14}块，每组4块=2^2，组索引：12位

而主存容量是cache容量的256倍，则主存容量=2^{14}*256块=2^{22}块

故：tag：22-12=10位

主存地址	数　据	主存块地址（tag）
000000_{16} =0000 0000　0000 0000 0000 0000_2	87568536	0000 0000 00_2 =000_{16}
000008_{16}=0000 0000　0000 0000 0000 1000_2	87792301	0000 0000 00_2 =000_{16}
010004_{16}=0000 0001　0000 0000 0000 0100_2	9ABEFCD0	0000 0001 00_2 =004_{16}
$01FFFC_{16}$=0000 0001　1111 1111 1111 1100_2	4FFFFC68	0000 0001 11_2=007_{16}
$FFFFF8_{16}$=1111 1111　1111 1111 1111 1000_2	01BF2460	1111 1111 11_2 =$3FF_{16}$

cache组	标志	数据
000	000	87568536
002	000	87792301
001	004	9ABEFCD0
0FFF	007	4FFFFC68
0FFE	3FF	01BF2460

某计算机中主存容量为 4MB，*cache* 容量为 16KB，每块包含 8 个字，每字 32 位，映射方式采用四路组相联。设 *cache* 的初始状态为空，CPU 依次从主存第 0，1，2，…，99 号单元读出 100 个字(每次读一个字)，并重复此操作 10 次，替换算法采用 LRU。

(1) 求 *cache* 的命中率。

(2) 若 *cache* 比主存快 10 倍，分析采用 cache 后存储访问速度提高了多少？

解：（1） 主存 100 个字单元分 13 块，而 Cache 有 128 组，故访问主存前 100 号单元不发生替换调度。初态为空，每块第一次不命中，后 9 次访问均命中。

100 号单元对应 13 块，第一轮访问 13 次不命中，后 9 轮访问均命中。

循环 10 次的总命中率为:

(100*10-13)/(10*100) = 98.7%

(2) 设 Cache 的存取周期为 t，则主存存取周期为 10t

直接从主存读取所有数据所需时间为: 1000*10t = 10000t

通过 Cache 访问 1000 个数据的时间为: Tc= 13*(10t+1t) +(1000-13)*t = 1130t

使用了 cache 后速度提高了：10000t/1130t=8.85 倍

例 4.4 某计算机的主存地址空间大小为 256MB，按字节编址。指令 cache 和数据 cache 分离，均有 8 个 cache 行，每个 cache 行大小均为 64B，数据 cache 采用直接相联映射方式。现有两个功能相同的程序 A 和 B，其伪代码如下。

程序 A:

```
int a[256][256];
int sum_array1()
{
    int i,j,sum=0; for(i=0;i<256;i++)
    for(j=0;j<256;j++) sum+=a[i][j];
    return sum;
}
```

程序 B:

```
int a[256][256];
int sum_array2()
{
    int i,j,sum=0; for(j=0;j<256;j++)
    for(i=0;i<256;i++) sum+=a[i][j];
    return sum;
}
```

假定 int 型数据用 32 位补码表示，程序编译时 i、j、sum 均分配在寄存器中，数组 a 按行优先方式存放，其首地址为 320（十进制数）。请回答下列问题，要求说明理由或给出计算过程。

（1）若不考虑用于 cache 一致性维护和替换算法的控制位，则数据 cache 的总容量为多少？

（2）数组元素 a[0][31] 和 a[1][1] 各自所在的主存块对应的 cache 行号分别是多少（cache 行号从 0 开始）？

（3）程序 A 和 B 的数据访问命中率各是多少？哪个程序的执行时间更短？

解：

（1）数据 cache 总容量 = cache 行数 × 行大小，不考虑一致性维护和替换算法的控制位，每个 cache 行主要包括有效位 valid、区地址标记字段、数据块 3 部分。

主存地址为 256MB，因此主存地址位宽为 28，数据 cache 有 8 个 cache 行。index 字段位宽 $r=3$；每个 cache 行大小为 64B，故块内偏移地址位宽 $w=6$。区地址 tag 字段位宽为 28−3−6 = 19 位。因此，数据 cache 的总容量应为：$8\times[(1+19)+64\times8]/8 = 532B$。

（2）数组按行优先方式存放，首地址为 320，数组元素占 4 个字节，行优先和列优先时二维数组元素在内存中的分布如表 4.6 所示。

表 4.6 二维数组元素在内存中的分布

内存地址	320	324	328	…	1340	1344	1348
行优先	a[0][0]	a[0][1]	a[0][2]	…	a[0][255]	a[1][0]	a[1][1]
列优先	a[0][0]	a[1][0]	a[2][0]	…	a[255][0]	a[0][1]	a[1][1]

根据表 4.6 可知：

a[0][31] 所在的主存地址为 (320+31×4) = 444；

a[1][1] 所在的主存地址为 320+256×4+1×4 = 1348。

根据直接相联映射规则可知：

a[0][31] 所在的主存块对应的 cache 行号 = 块号 mod 8 = (444/64)mod 8 = 6；

a[1][1] 所在的主存块对应的 cache 行号 = 块号 mod 8 = (1348/64)mod 8 = 5。

（3）由于 i、j、sum 均分配在寄存器中，故数据访问命中率仅考虑数组 a 的情况。A、B 两程序的功能都是对二维数组累加求和，数组中的每一个元素仅被使用一次。数组按行优先存放，数据 cache 的容量为 8×64B = 512B = 128 字，可以放下数组半行的数据。

① 程序 A 中数据的访问顺序与存储顺序相同，具有较好的空间局部性，每个 cache 数据块可以存储 16 个 int 型数据，顺序访问时，第一次访问缺失，载入数据块，后续 15 次访问都会命中。程序 A 中的所有数据访问都符合这一规律，故命中率为 15/16，即程序 A 的数据访问命中率是 93.75%。

② 程序 B 按照数组的列执行外层循环，在执行内层循环的过程中，将连续访问不同行但同一列的数据，由于数组中一行数据大小为 256×4B=256 字，是 cache 容量的 2 倍，因此不同行的同一列数组元素对应同一个 cache 行，第一次访问不命中，载入数据块，且后续访问仍然不命中，载入新的数据块到同一行，这样所有数据都无法命中，故命中率是 0。由于从 cache 读数据比从主存读数据快很多，因此程序 A 的执行速度比程序 B 快得多。