



计算机组成与结构习题讲解 (1)



计算机的逻辑部件

- 2.4 设计用若干个全加器和若干个与门、或门实现的8421码十进制加法器单元电路。
- 分析与解答：
 - BCD码：0000-1001
 - 二进制：0000-1111

计算机的逻辑部件

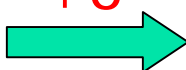
二进制结果

C	E3	E2	E1	E0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1

BCD结果

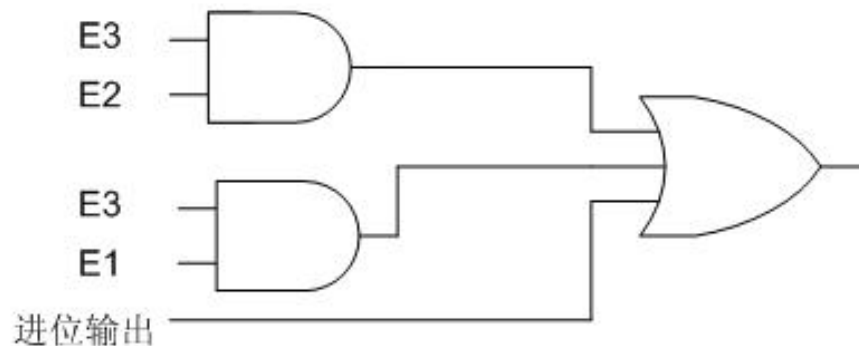
C	E3	E2	E1	E0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1

+6

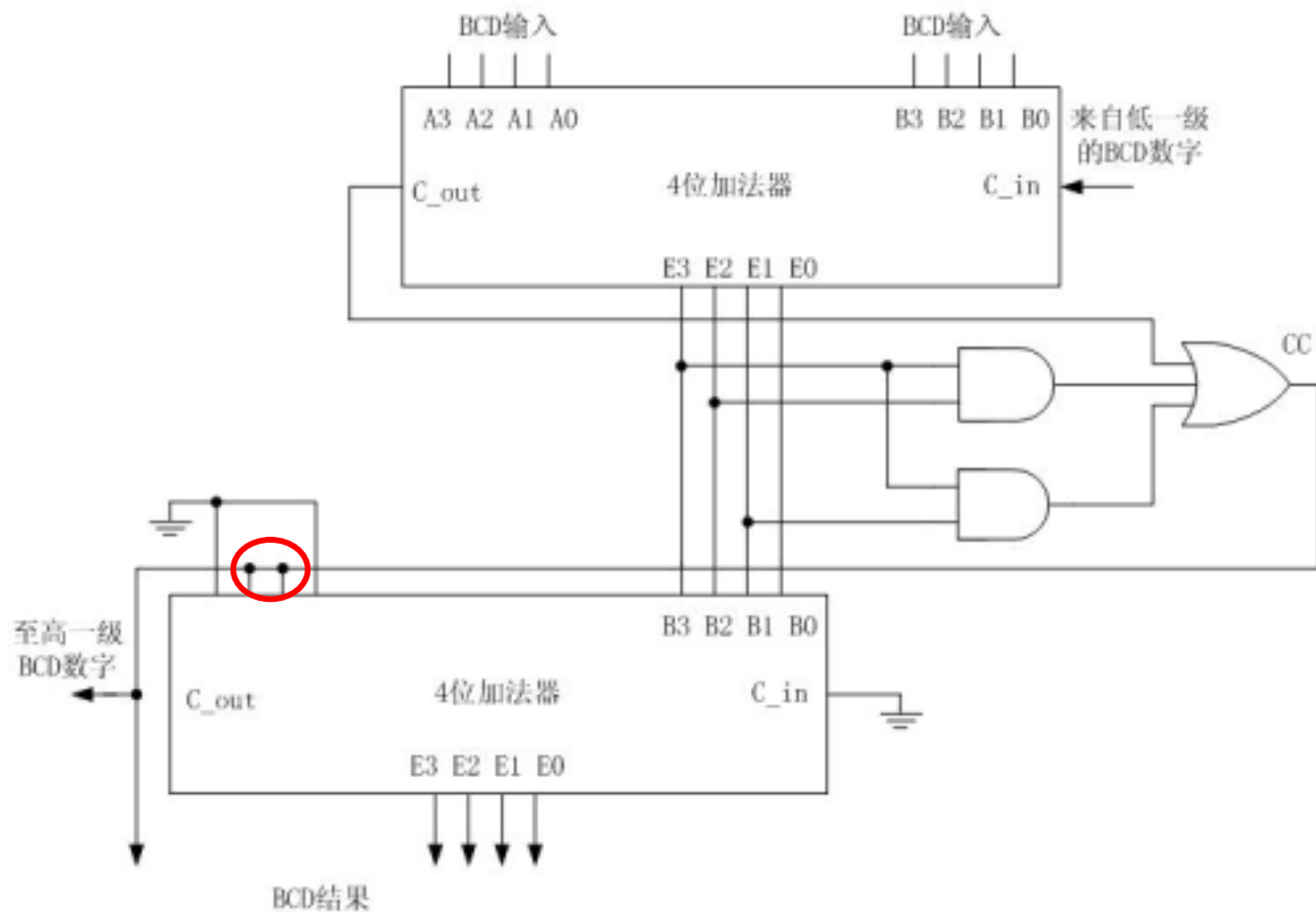


计算机的逻辑部件

- 何时要对结果作修正？
 - 当二进制加法的结果为 10_{10} 、 11_{10} 、 12_{10} 、 13_{10} 、 14_{10} 、 15_{10} 时
 - 二进制结果有进位时
- 由卡诺图，得到：
 - 结果 $=E3 \cdot E2 + E3 \cdot E1$



计算机的逻辑部件





计算机的算术运算

- 4.1 用32位二进制2的补码表示法表示数 512_{10}

- 分析与解答：

- 512_{10}
- $= (10\ 0000\ 0000)_2$
- $= (0000\ 0000\ 0000\ 0000\ 0000\ 0010\ 0000\ 0000)_2$

计算机的算术运算

- 4.2 用32位二进制2的补码表示法表示数 -1023_{10}

- 分析与解答：



- -1023_{10}
- $=(11\ 1111\ 1111)_2$
- $=(1000\ 0000\ 0000\ 0000\ 0000\ 0011\ 1111\ 1111)_{\text{原}}$
- $=(1111\ 1111\ 1111\ 1111\ 1111\ 1100\ 0000\ 0001)_{\text{补}}$



计算机的算术运算

- 4.4 给出如下用二进制2的补码表示法表示的数的十进制数：

1111 1111 1111 1111 1111 1110 0000 1100₂

- 分析与解答：

- (1111 1111 1111 1111 1111 1110 0000 1100)_补
- =(1000 0000 0000 0000 0000 0001 1111 0011)_反
- =(1000 0000 0000 0000 0000 0001 1111 0100)_原
- =-(1 1111 0100)₂
- =-500



计算机的算术运算

- 4.8 给出二进制数

1100 1010 1111 1110 1111 1010 1100 1110₂的十六进制数

- 分析与解答：

- (Hex)0-F \Leftrightarrow (B)0000-1111
- 1100 1010 1111 1110 1111 1010 1100 1110₂
- =(12 10 15 14 15 10 12 14)
- =(CAFEFACE)₁₆



计算机的算术运算

- 4.14 二进制数的各个数位本身并不是天生就有某种特定的含义。请考虑如下的二进制位串：

1000 1111 1110 1111 1100 0000 0000 0000

若它分别表示如下所示的三种数，那么他们的含义各是什么？

- 2的补码表示的整数
- 无符号整数
- 单精度浮点数



计算机的算术运算

■ 分析与解答：

■ 2的补码表示的整数

- $(1000\ 1111\ 1110\ 1111\ 1100\ 0000\ 0000\ 0000)_{\text{补}}$
- $= (1111\ 0000\ 0001\ 0000\ 0100\ 0000\ 0000\ 0000)_{\text{原}}$
- $= -(111\ 0000\ 0001\ 0000\ 0100\ 0000\ 0000\ 0000)$
- $= -1880113152_{10}$

■ 无符号整数

- $(1000\ 1111\ 1110\ 1111\ 1100\ 0000\ 0000\ 0000)$
- $= +(1000\ 1111\ 1110\ 1111\ 1100\ 0000\ 0000\ 0000)$
- $= +2414854144_{10}$



计算机的算术运算

■ 单精度浮点数

■ 1000 1111 1110 1111 1100 0000 0000 0000
S E F

■ $S = (-1)^1 = -1$

■ $E = 00011111 = 31_{10}$

■ $F' = 1_{10} + (110\ 1111\ 1100\ 0000\ 0000\ 0000)_2$

■ 单精度浮点数 = $S \times F' \times 2^E$

计算机的算术运算

- 4.26 请根据IEEE 754标准，写出 10.5_{10} 分别为单、双精度浮点数时，其二进制形式

- 分析与解答：

- 规格化： $10.5_{10} = (1010.1)_2 = (1.0101)_2 \times 2^3_2$
- 单精度浮点数公式（S：1位，E：8位，F：23位）
 - 移码偏移值=127
 - $S=0$
 - $E'=3 \Rightarrow E=3+127=130=(10000010)_2$
 - $F'=(1.0101)_2 \Rightarrow F=F'-1=(0101)_2$
- 0100 0001 0010 1000 0000 0000 0000 0000



计算机的算术运算

- $10.5_{10} = (1010.1)_2 = (1.0101)_2 \times 2^3_2$
- 双精度浮点数公式 (S : 1位, E : 11位, F : 53位)
 - 移码偏移值 = 1023
 - $S = 0$
 - $E' = 3 \Rightarrow E = 3 + 1023 = 1026 = (10000000010)_2$
 - $F' = (1.0101)_2 \Rightarrow F = F' - 1 = (0101)_2$
- 0100 0000 0010 0101 0000 0000 0000 0000 ...



运算方法和运算部件

- 3.9 设机器字长16位。定点表示时，数值15位，符号位1位；浮点表示时，阶码6位，其中阶符1位，尾数10位，其中，数符1位；阶码底为2。试求：
 - 1) 定点原码整数表示时，最大正数、最小负数各是多少？
 - 2) 定点原码小数表示时，最大正数、最小负数各是多少？
 - 3) 浮点原码表示时，最大浮点数和最小浮点数各是多少？绝对值最小的呢（非0）？估算表示的十进制值的有效数字位数。

运算方法和运算部件

■ 分析与解答：

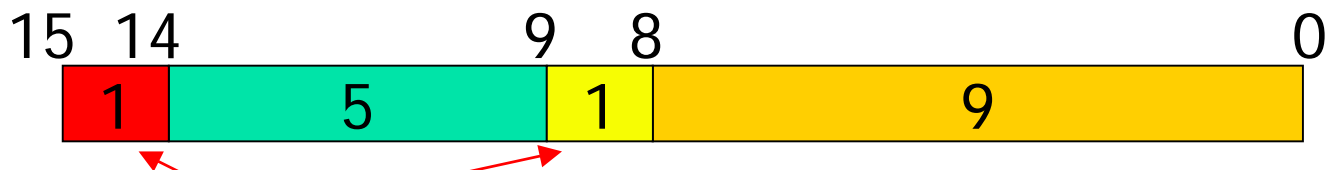
- 1) 定点整数：数值15位，符号位1位



- $-11\dots111 \sim +11\dots111$
- $-(2^{15}-1)_{10} \sim +(2^{15}-1)_{10}$
- 2) 定点小数：数值15位，符号位1位
- $-0.11\dots111 \sim +0.11\dots111$
- $-(1-2^{-15})_{10} \sim +(1-2^{-15})_{10}$

运算方法和运算部件

- 3) 浮点：阶码6位，其中阶符1位，尾数10位，其中数符1位



- $+2^{(+2^5-1)} \times (1-2^{-9}) = +2^{31} \times (1-2^{-9})$
- $-2^{(+2^5-1)} \times (1-2^{-9}) = -2^{31} \times (1-2^{-9})$
- $2^{(-(2^5-1))} \times 2^{-9} = 2^{-31} \times 2^{-9} = 2^{-40}$



运算方法和运算部件

■ 3.12 写出下列各数的移码

■ +01101101

■ -11001101

■ -00010001

■ +00011101



运算方法和运算部件

■ 分析与解答：

原码	反码	补码	移码
+01101101 (001101101)	001101101	001101101	101101101
-11001101 (111001101)	100110010	100110011	000110011
-00010001 (100010001)	111101110	111101111	011101111
+00011101 (000011101)	000011101	000011101	100011101



运算方法和运算部件

■ 3.19 用补码一位乘计算

$X=0.1010$, $Y=-0.0110$ 的积 $X \cdot Y$

■ 分析与解答：

- $X=0.1010 \rightarrow (00.1010)_{\text{原}} \rightarrow (00.1010)_{\text{补}}$
- $Y=-0.0110 \rightarrow (10.0110)_{\text{原}} \rightarrow (11.1010)_{\text{补}}$
- $-X=-0.1010 \rightarrow (10.1010)_{\text{原}} \rightarrow (11.0110)_{\text{补}}$

运算方法和运算部件

	00.0000	1 0 1 <u>0</u>	
+ 0	00.0000		
	00.0000		
	00.0000	0 1 0 <u>1</u>	→
+ [X] _补	00.1010		
	00.1010		
	00.0101	0 0 1 <u>0</u>	→
+ 0	00.0000		
	00.0101		
	00.0010	1 0 0 <u>1</u>	→
+ [X] _补	00.1010		
	00.1100		
	00.0110	0 1 0 0	→
+ [-X] _补	11.0110		
	11.1100	0 1 0 0	

[X•Y]_补 = 1.11000100

[X•Y] = -0.00111100

← 补码一位乘



运算方法和运算部件

- 3.21 $X=0.10110$, $Y=0.11111$, 用加减交替法补码一位除计算 X/Y 的商
- 分析与解答：
 - $X=0.10110 \rightarrow (00.10110)_{\text{原}}$
 - $Y=0.11111 \rightarrow (00.11111)_{\text{原}}$
 - $-Y=-0.11111 \rightarrow (10.11111)_{\text{原}}$
 $\rightarrow (11.00001)_{\text{补}}$

运算方法和运算部件

00.10110 + 11.00001	0 0 0 0 0	← + [-Y] _补
11.10111	0 0 0 0 0	
← 11.01110	0 0 0 0 <u>1</u>	← + [Y] _补
+ 00.11111		
00.01101	0 0 0 0 1	
← 00.11010	0 0 0 1 <u>1</u>	← + [-Y] _补
+ 11.00001		
11.11011	0 0 0 1 0	
← 11.10110	0 0 1 0 <u>1</u>	← + [Y] _补
+ 00.11111		
00.10101	0 0 1 0 1	
← 01.01010	0 1 0 1 <u>1</u>	← + [-Y] _补
+ 11.00001		
00.01011	0 1 0 1 1	
← 00.10110	1 0 1 1 <u>1</u>	← + [-Y] _补
+ 11.00001		
11.10111	1 0 1 1 0	
← 11.01110	0 1 1 0 <u>1</u>	
+ 00.11111		
00.01101		

$$[X]_{\text{补}} = 00.10110$$

$$[Y]_{\text{补}} = 00.11111$$

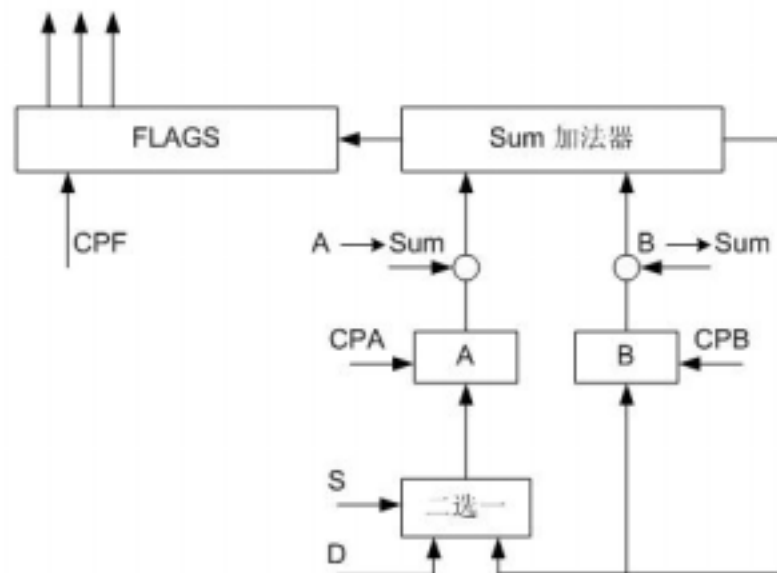
$$[-Y]_{\text{补}} = 11.00001$$

$$[X/Y]_{\text{补}} = 0.1011\mathbf{1}$$

$$[X/Y] = 0.10111$$

运算方法和运算部件

- **3.27** 设某运算器只由一个加法器和A、B两个D型边沿寄存器组成，A、B均可接收加法器输出，A还可接收外部数据，如图。





运算方法和运算部件

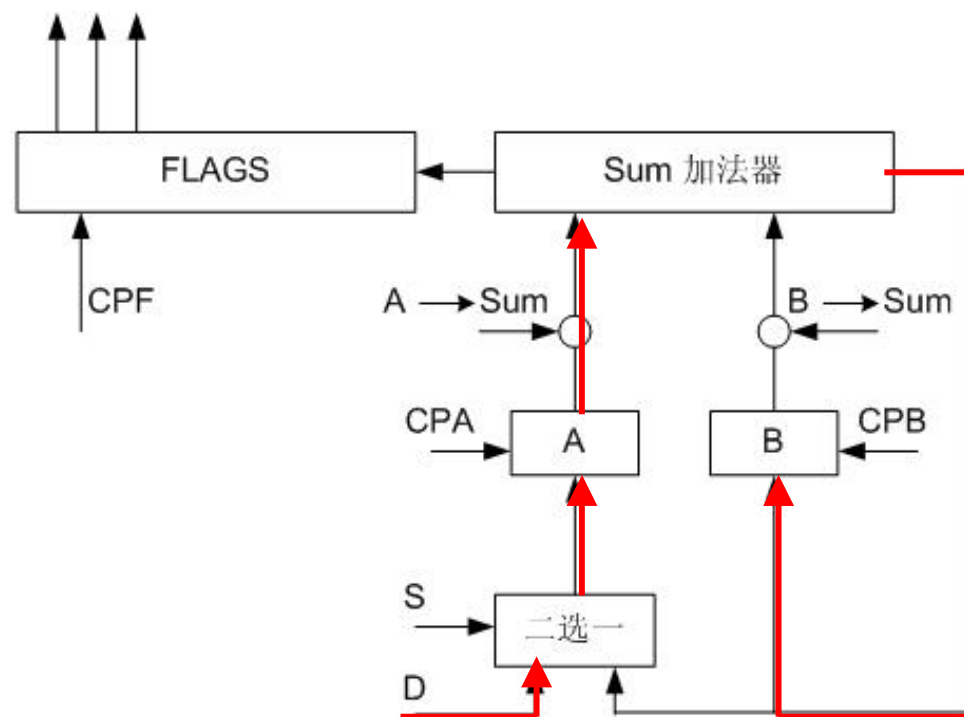
■ 问：

- 1) 外部数据如何才能传送到B？
- 2) 如何实现 $A+B \rightarrow A$ 和 $A+B \rightarrow B$ ？
- 3) 如何估算加法执行时间？
- 4) 若A、B均为锁存器，实现 $A+B \rightarrow A$ 和 $A+B \rightarrow B$ 有何问题？

运算方法和运算部件

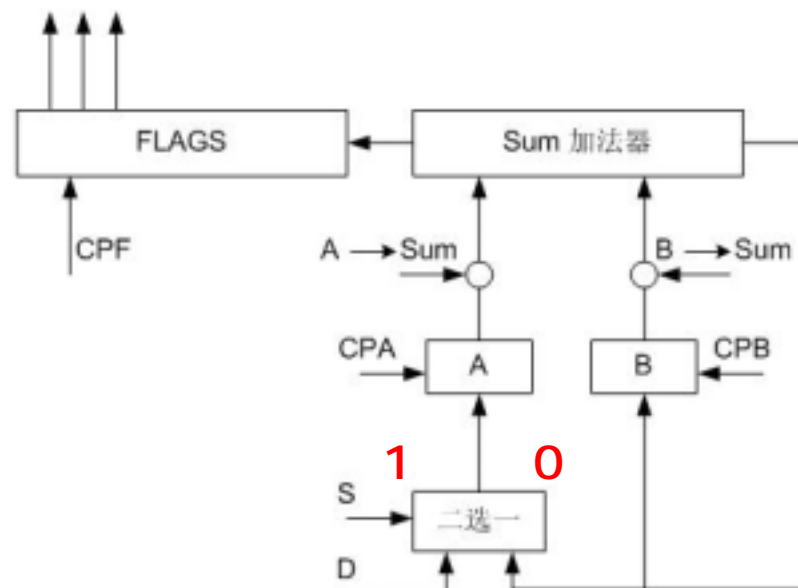
■ 分析与解答：

- 1) 外部数据如何才能传送到B？



运算方法和运算部件

- 2) 实现 $A+B \rightarrow A$
 - Load D
 - $S=1$: $D \rightarrow A$
 - CPA、CPB 脉冲:
 $A+B \rightarrow \text{Sum}$
 - $S=0$: $\text{Sum} \rightarrow A$
- 同理：实现 $A+B \rightarrow B$

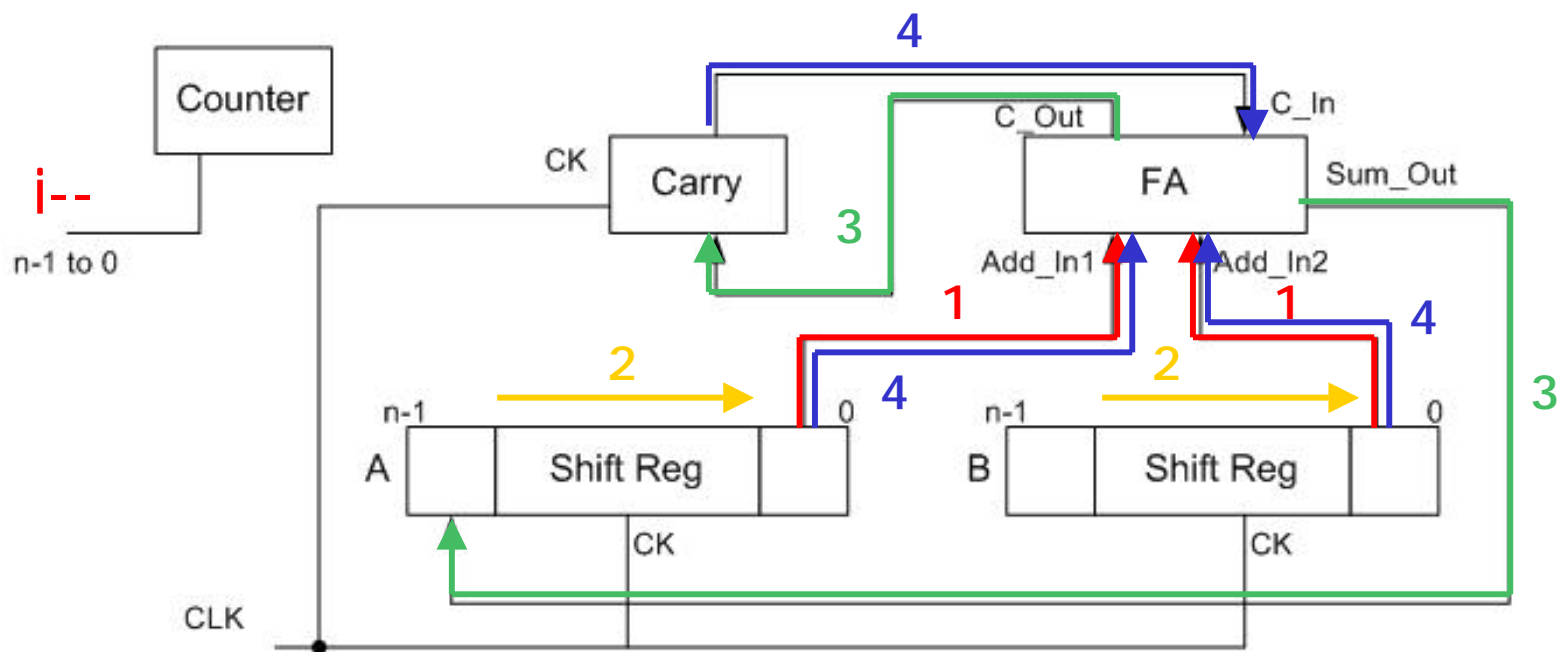




运算方法和运算部件

- **3.28** 今有一串行加法器，计算两个n位数据之和，已知相加两数存放在A、B寄存器中，请画出能实现 $(A) + (B) \rightarrow A$ 的逻辑图。图中只准用一个一位加法器，逐位进行计算
- **分析与解答：**
 - 一位加法器 \Rightarrow 各位串行计算
 - 寄存器要有移位功能
 - i 位进位和 $i+1$ 位的操作数一起计算 \Rightarrow 全加器
 - n 位数据加法 \Rightarrow 使用计数器确定加法是否完成

运算方法和运算部件



计算机系统结构基础知识

- 1-2. 如有一个经解释实现的计算机，可以按功能划分成4级。每一级为了执行一条指令需要下一级的N条指令解释。若执行第一级的一条指令需要K ns时间，那么执行第2、3、4级的一条指令各需要用多少时间？

分析与解答：NK ns、 N^2K ns、 N^3K ns

4
3
2
1



指令系统设计与优化习题

- 2-13. 某机14条指令的使用频度分别是：0.01、0.15、0.12、0.03、0.02、0.04、0.02、0.04、0.01、0.13、0.15、0.14、0.11、0.03。分别求出用等长二进制编码、Huffman编码、只能用两种码长的扩展操作码编码等3种方式的操作码平均长度。
- 分析与解答：
 - 等长编码时，二进制码位数： $\lceil \log_2 n \rceil$
 - Huffman编码，平均码长： $p_i \cdot l_i$



指令系统设计与优化习题

- 共14条指令
 - 等长编码：
 - 平均码长 $\lceil \log_2 14 \rceil = 4$
 - Huffman编码：
 - 平均码长 = 3.38
 - 扩展操作码（3/5扩展编码法）：
 - 000 ~ 101 : 0.15、0.15、0.14、0.13、0.12、0.11
 - 110XX和111XX : 0.03、0.02、0.04、0.02、0.04、0.01、0.03、0.01
 - 平均码长 $p_i \cdot l_i = 3 \times 0.8 + 5 \times 0.2 = 3.4$



指令系统设计与优化习题

- 2-14. 某模型机有9条指令，使用频度为：

ADD	0.30
SUB	0.24
CLA	0.20
JMP	0.07
STO	0.07
JOM	0.06
CIL	0.03
SHR	0.02
STP	0.01

要求：有两种指令字长，都按双操作数地址指令格式编排。采用扩展操作码，限制只能用两种码长。该机有若干个通用寄存器，主存16位宽，按字节编址，采用整数边界存储，任何指令都在一个主存周期中取得，短指令为寄存器-寄存器型，长指令为寄存器-主存型，主存地址应能变址寻址。



指令系统设计与优化习题

- 1) 仅根据使用频度，设计Huffman操作码，并计算平均码长；
- 2) 考虑题目其它要求，设计优化的指令操作码，并计算码长；
- 3) 该机允许使用多少可编址的通用寄存器？
- 4) 画出该机两种指令字格式，标出各字段之位數；
- 5) 访存操作数地址寻址的最大相对位移量为多少字节？

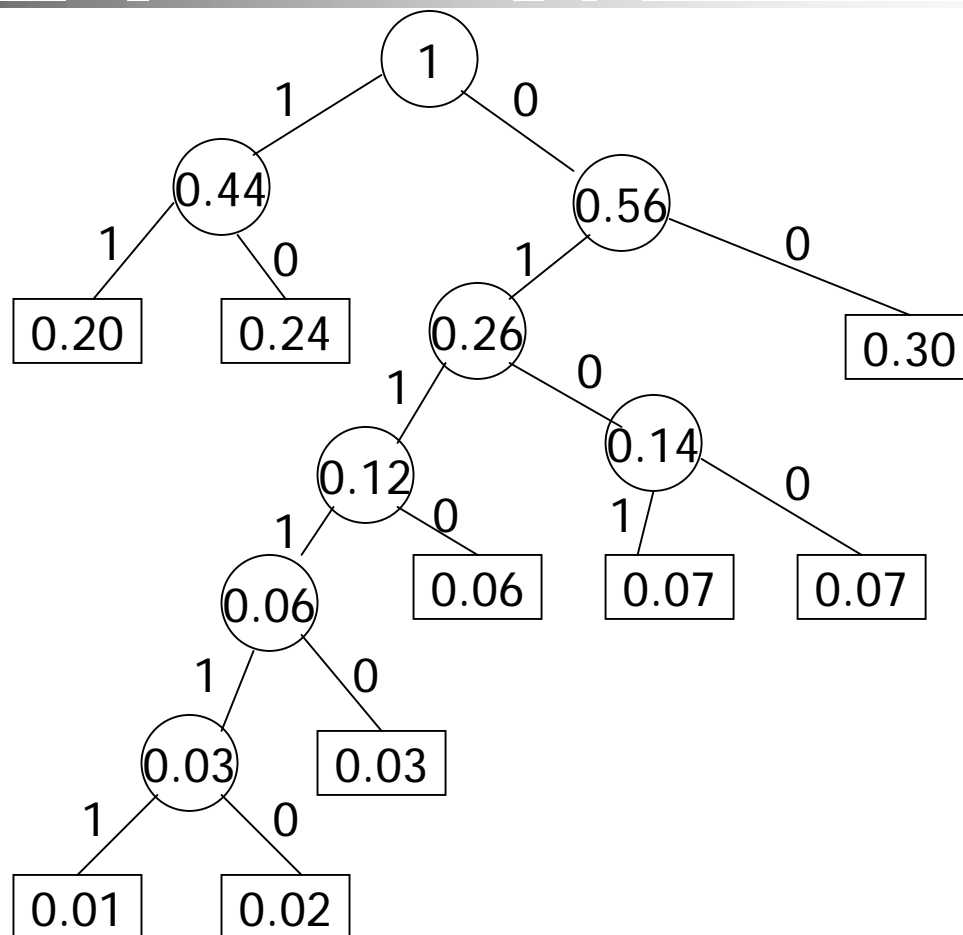
■ 分析与解答：

- 构造Huffman树，解出1)，后面各小题的解题关键是确定两种指令字的格式及其各字段的位数。

指令系统设计与优化习题

1) Huffman树

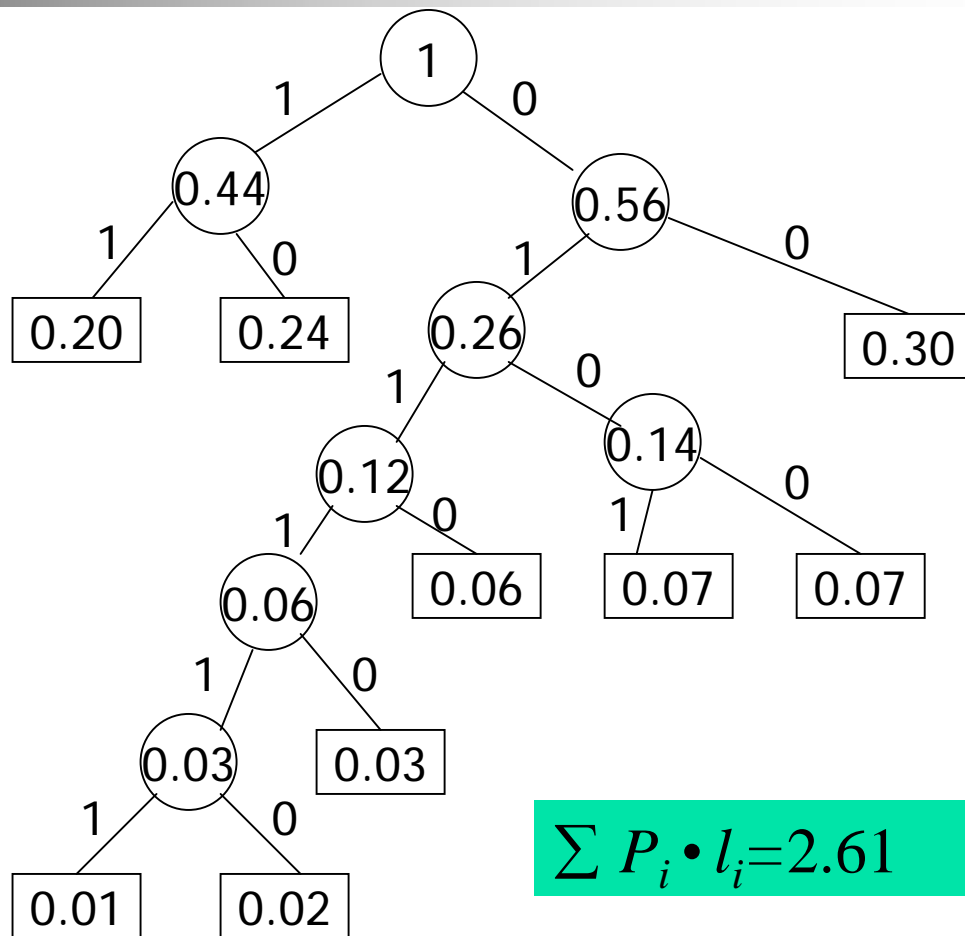
ADD	0.30
SUB	0.24
CLA	0.20
JMP	0.07
STO	0.07
JOM	0.06
CIL	0.03
SHR	0.02
STP	0.01



指令系统设计与优化习题

Huffman编码：

指 令	操作码码制
ADD	00
SUB	10
CLA	11
JMP	0100
STO	0101
JOM	0110
CIL	01110
SHR	011110
STP	011111



$$\sum P_i \cdot l_i = 2.61$$



指令系统设计与优化习题

2) 分析与解答：

- 要求用两种码长，ADD (0.30)、SUB (0.24) 和CLA (0.20) 3条指令频度相对较高，因此短码宜采用2位长，共 $2^2=4$ 个码点，剩下一个作为扩展标志码，有6条频度低的指令，所以需扩展出3位才可以满足。于是长操作码为5位。这样就得到扩展操作码。



指令系统设计与优化习题

Huffman编码：

指 令	操作码码制
ADD	00
SUB	10
CLA	11
JMP	0100
STO	0101
JOM	0110
CIL	01110
SHR	011110
STP	011111

扩展的操作码编码

指 令	操作码码制
ADD	00
SUB	01
CLA	10
JMP	11000
STO	11001
JOM	11010
CIL	11011
SHR	11100
STP	11101



指令系统设计与优化习题

后3小题分析与解答：

- 3) 该机允许使用多少可编址的通用寄存器？
 - 由已知条件：两种指令都在一个主存周期中取得、主存16位宽 \Rightarrow 长指令不超过16位。
 - 由已知条件：按字节编址、采用按整数边界存储 \Rightarrow 短指令只能是8位，长指令16位。
 - 由已知条件：短指令为寄存器-寄存器型，长指令为寄存器-主存型 \Rightarrow 指令按双操作数编排

指令系统设计与优化习题

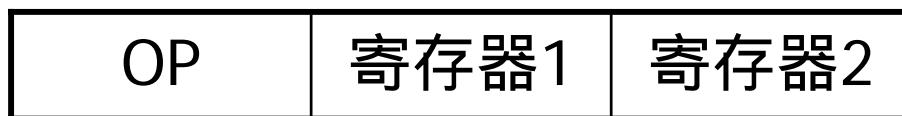
- 4) 画出该机两种指令字格式，标出各字段之位數：

- 短指令寄存器-寄存器型，其格式

2位

3位

3位



- 长指令为寄存器-主存型，主存地址应能变址寻址，格式为：

5位

3位

3位

5位





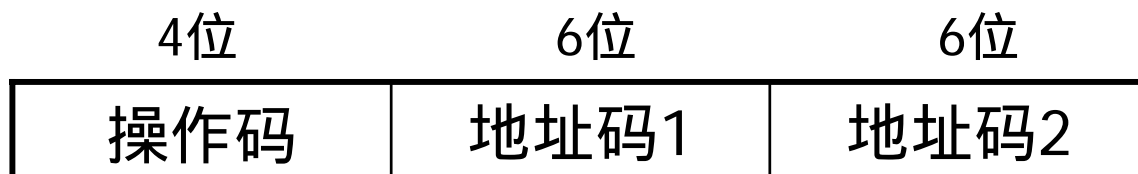
指令系统设计与优化习题

- 5) 访存操作数地址寻址的最大相对位移量为多少字节？
 - 允许通用寄存器数 $2^2=8$ 个；寻址最大相对位移量 $2^5=32$ 字节。

5位	3位	3位	5位
OP	寄存器号	变址寄存器	相对位移

指令系统设计与优化习题

- **2-15.** 某机指令字长16位。设有单地址指令和双地址指令两类。若每个地址字段均为6位，且双地址指令为x条，问单地址指令最多可以有多少条？
- **分析与解答：** 依据是扩展码中的短码不能是长码的前缀。
双地址指令： 格式为



操作码4位，共 $2^4=16$ 种短操作码，x条双地址指令占用了x个码点，剩 $16-x$ 个作为扩展标志。

单地址指令： 操作码10位，每个码扩展出6位操作码，所以，最多可以表示单地址指令 $(16-x) \cdot 2^6$ 条。



计算机组成与结构 习题讲解 (2)



主存储器

- 4.5 有一个 $512\text{K} \times 16$ 的存储器，由 $64\text{K} \times 1$ 的 2164RAM 芯片构成（芯片内是 4 个 128×128 结构），问：
 - (1) 总共需要多少个 RAM 芯片？
 - (2) 采用分散刷新方式，如单元刷新间隔不超过 2ms ，则刷新信号的周期是多少？
 - (3) 如采用集中刷新方式，设读/写周期 $T = 0.1 \mu\text{s}$ ，存储器刷新一遍最少用多少时间？



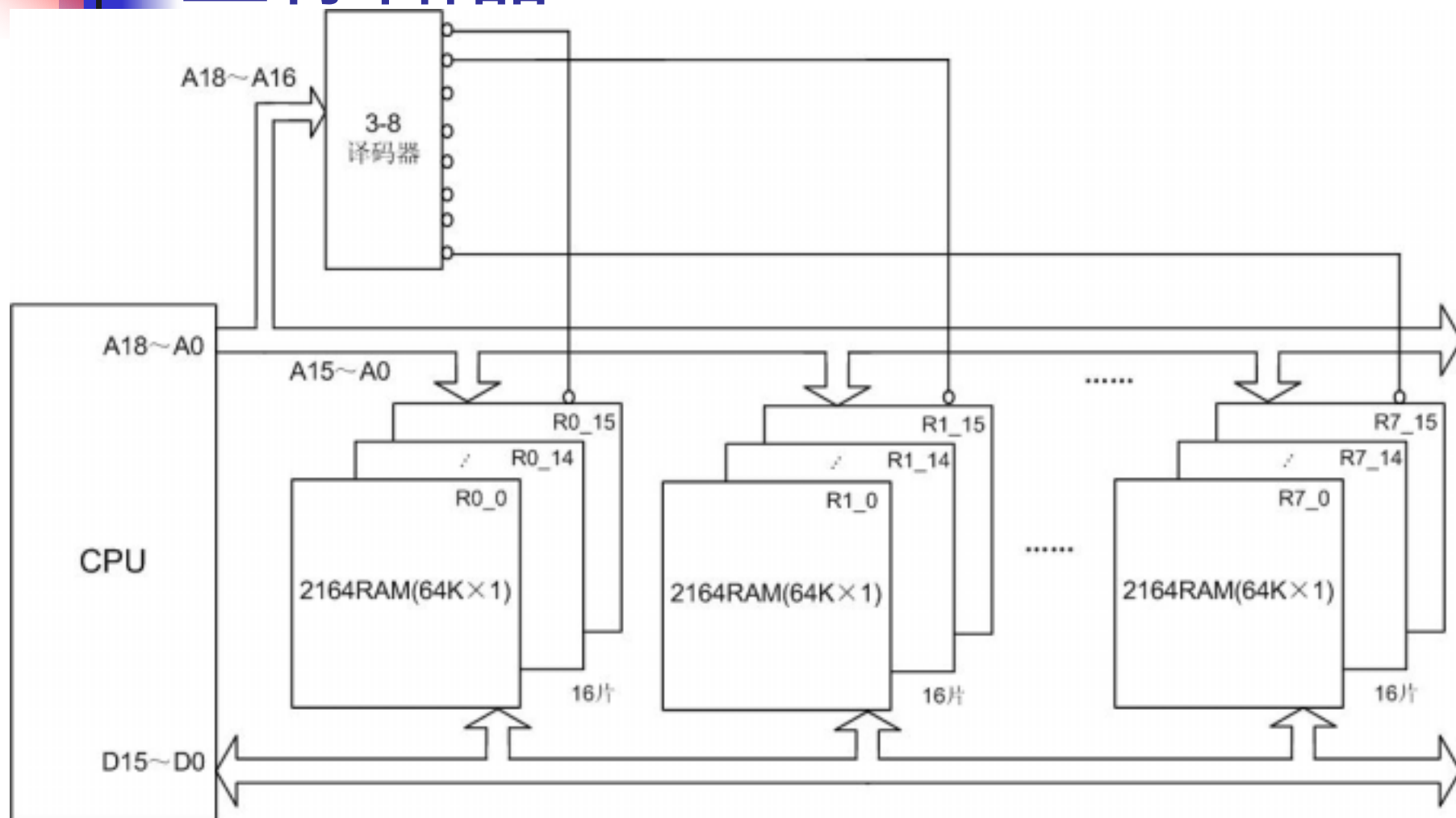
主存储器

- 分析与解答：

- (1) $64K \times 1 \Rightarrow 512K \times 16$

- 位扩展： $16/1=16$ 片
 - 字扩展： $512/64=8$ 片
 - 共要 $16 \times 8=128$ 片

主存储器





主存储器

- (2) 分散刷新
 - 每个2164RAM由4个 128×128 的芯片构成
 - $2\text{ms}/128 = 15.625 \mu\text{s}$
- (3) 集中刷新
 - $0.1 \mu\text{s} \times 128 = 12.8 \mu\text{s}$



主存储器

- 4.6 某机器中，已知道有一个地址空间为0000H ~ 1FFFH的ROM区域，现在再用RAM芯片（8K × 4）形成一个16K × 8的RAM区域，起始地址为2000H，假设RAM芯片有CS[#]和WE[#]信号控制端。CPU地址总线为A15 ~ A0，数据总线为D7 ~ D0，控制信号为R/W（读/写），MREQ[#]（当存储器进行读或写操作时，该信号指示地址总线上的地址是有效的）。要求画出逻辑图。



主存储器

```
ROM:  0000 0000 0000 0000
       0001 1111 1111 1111
RAM1:  0010 0000 0000 0000
       0011 1111 1111 1111
RAM2:  0100 0000 0000 0000
       0101 1111 1111 1111
```

■ 分析与解答：

ROM(0000H ~ 1FFFFH) + RAM(16K × 8)

■ ROM容量：8K × 8

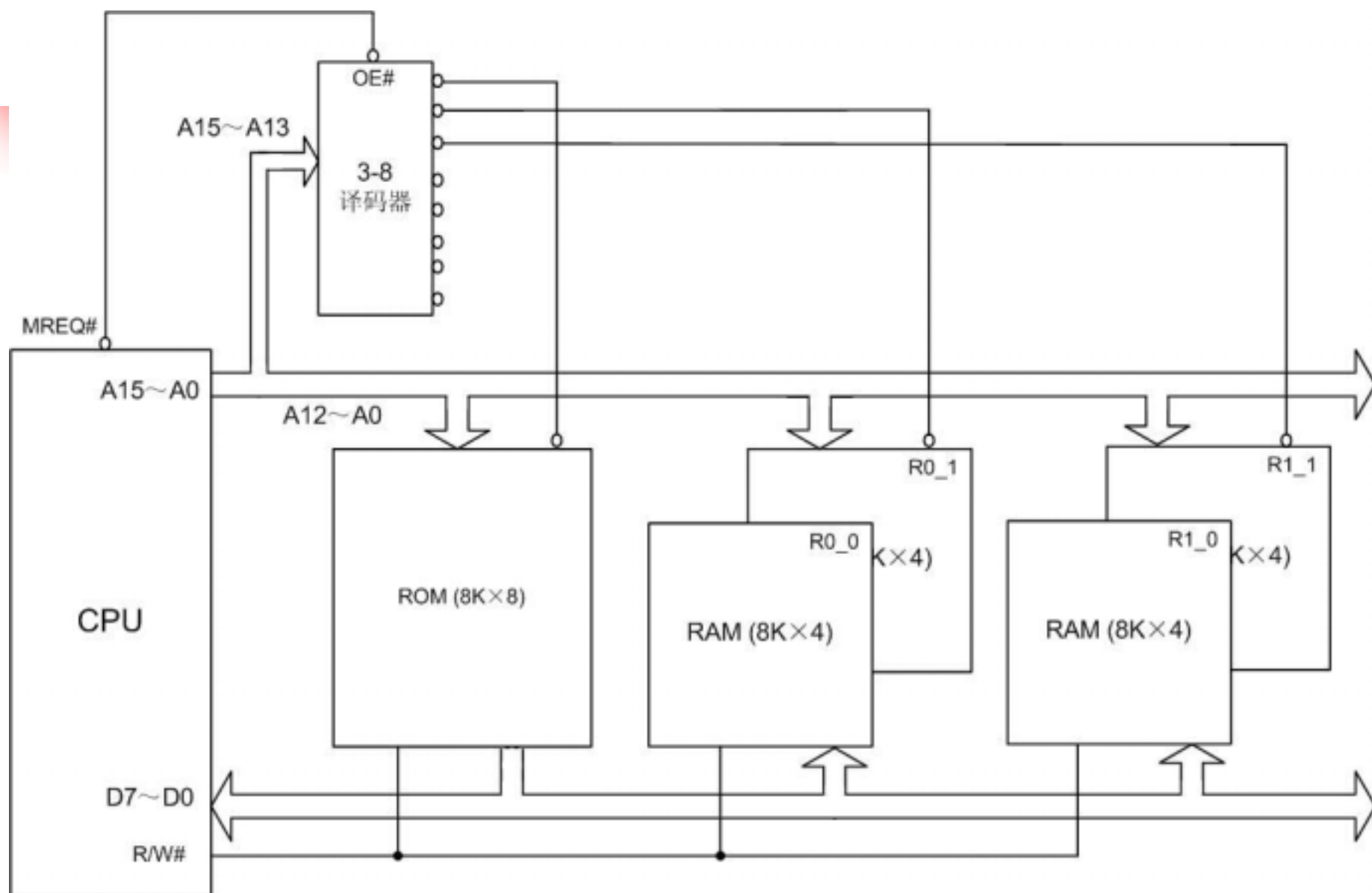
■ RAM由(8K × 4)的RAM芯片构成：

■ 位扩展：2片

■ 字扩展：2片

RAM1地址空间：2000H ~ 3FFFFH

RAM2地址空间：4000H ~ 5FFFFH



辅助存储器

- 8.3 设某磁盘存储器的平均找道时间为 t_s ，转速为每分 r 转，每磁道容量为 N 个字，每信息块为 n 个字。试推导读写一个信息块所需总时间 t_B 的计算公式。
- 分析与解答：
 - 找道时间： t_s
 - 找磁道时间： r 转/min
=> 每转一次的时间= $60/r$
=> 找磁道时间： $60/2r$
 - 读写每个字的时间： $60/2rN \times 2 = 60/rN$
=> 读写 n 个字的时间： $60n/rN$
 - 总时间 $t_B = t_s + 60/2r + 60n/rN$



辅助存储器

- 8.5 设磁盘组有11个盘片，每片有两个记录面；存储区域内直径2.36英寸，外直径5.00英寸；道密度为1250TPI（每英寸磁盘数），内层位密度52400bpi（每英寸位数），转速为2400rpm。问：
 - (1) 共有多少个存储面可用？
 - (2) 共有多少柱面？
 - (3) 每道存储多少字节？盘组总存储容量是多少？
 - (4) 数据传输率是多少？
 - (5) 每扇区存储2KB数据，在寻址命令中如何表示磁盘地址？
 - (6) 如果某文件长度超过了一个磁道的容量，应将它记录在同一个存储面上，还是记录在同一个柱面上？



辅助存储器

■ 分析与解答：

- (1) 11个盘片，每片有两个记录面
=> $11 \times 2 - 2 = 20$ 个存储面可用
- (2) 柱面数 = $(5.00 - 2.36) / 2 \times 1250 \text{TPI} = 1650$
- (3) 每道存储字节 = 内层位密度 \times 内层磁道长度
= $52400 \text{ bpi} \times 2.36 \times \text{PI} = 48.5 \text{ KB}$
 盘组总容量 = 每道存储字节 \times 道数 \times 存储面数
= $48.5 \text{ KB} \times 1650 \times 20 = 1600500 \text{ KB} = 1.6 \text{ GB}$
- (4) 转速为 $2400 \text{ rpm} = 40 \text{ rps}$
 数据传输率 = 每道存储字节 \times 转速
= $48.5 \text{ KB} \times 40 \text{ rps} = 1940 \text{ KB} = 1.94 \text{ MB}$



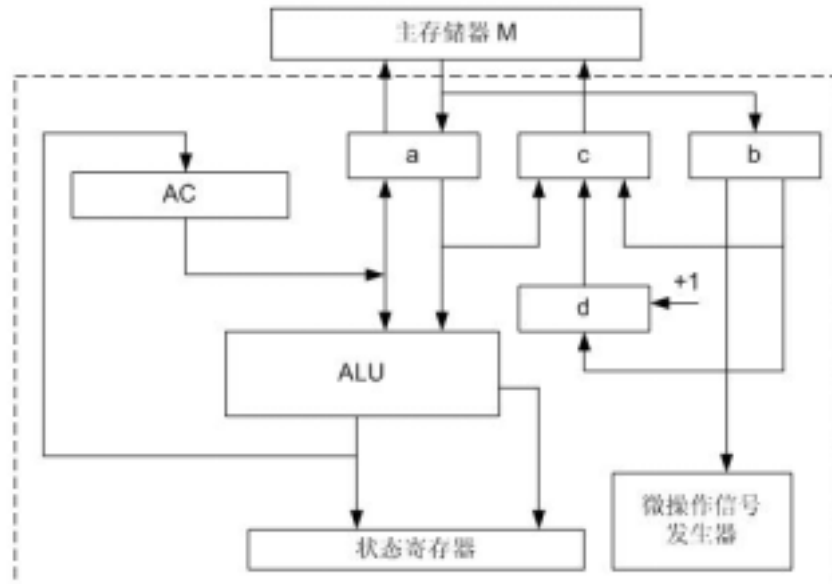
辅助存储器

- (5) 已知：每扇区存储2KB数据
 - 找存储面：20个存储面=>5位
 - 找柱面（磁道）：1650个柱面=>11位
 - 找扇区：
每柱面上扇区数= $48.5\text{KB}/2\text{KB}=25$ 个扇区=>5位
 - 总共需要： $5+11+5=21$ 位（磁盘地址）
- (6) 如果某文件长度超过了一个磁道的容量，应将它记录在同一个存储面上，还是记录在同一个柱面上？
 - 同一个柱面上：使得一次访存可以读写文件的所有内容

中央处理部件CPU

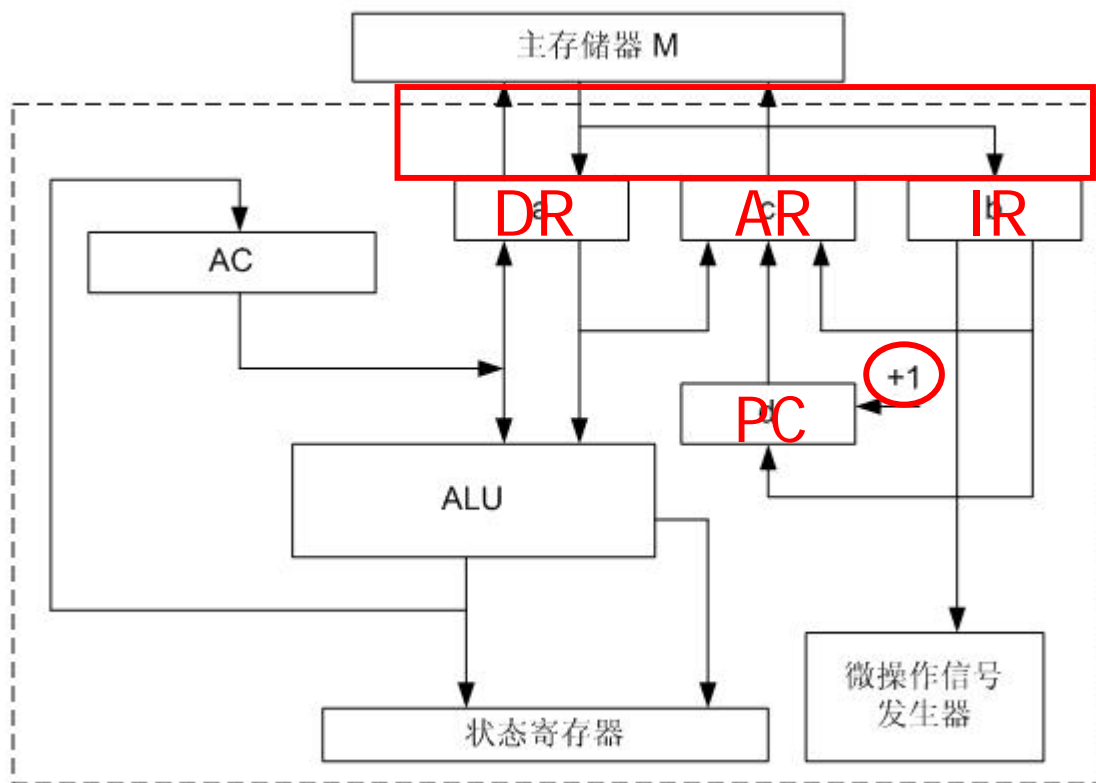
- 6.1 CPU结构如图所示，其中有一个累加器AC、一个状态条件寄存器和其他四个寄存器，各部分之间的连线表示数据通路，箭头表示信息传送方向。要求：

- (1) 标明图中a、b、c、d四个寄存器的名称
- (2) 简述指令从主存取到控制器的数据通路
- (3) 简述数据在运算器和主存之间进行存/取访问的数据通路



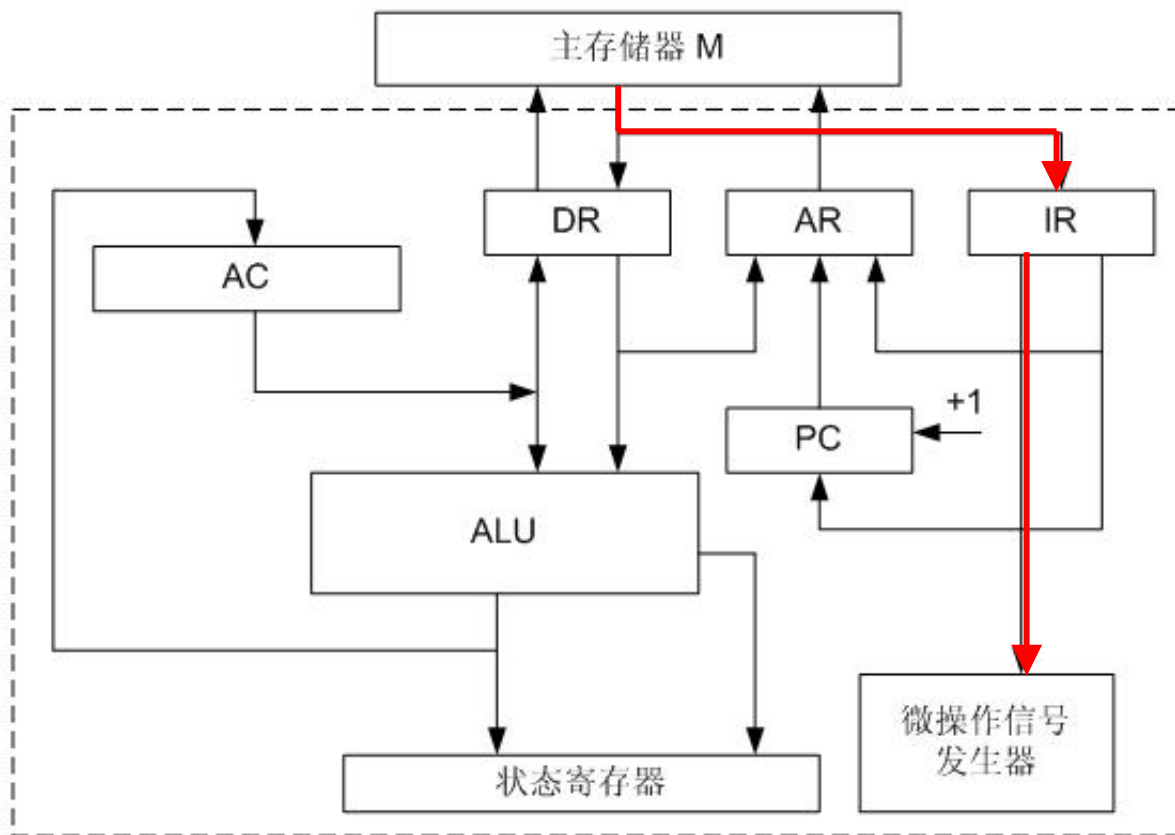
中央处理部件CPU

- (1) 标明图中a、b、c、d四个寄存器的名称



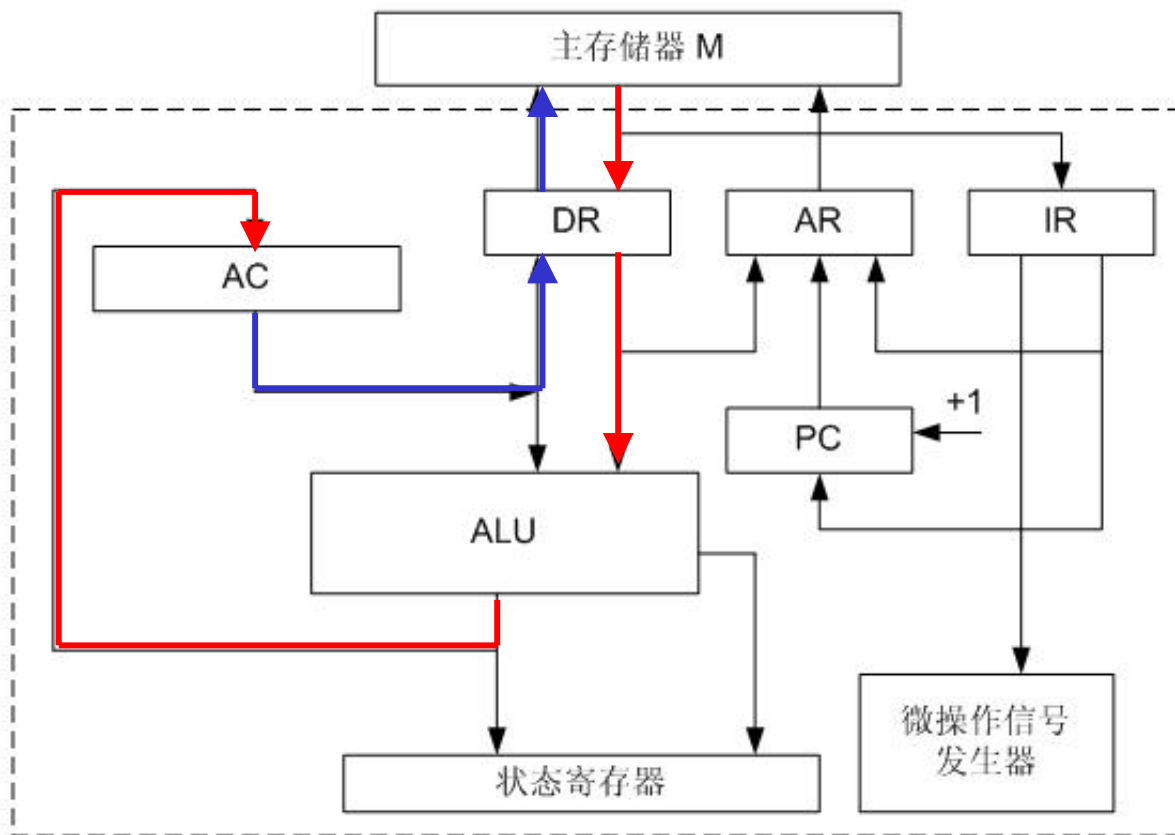
中央处理部件CPU

- (2) 简述指令从主存取到控制器的数据通路



中央处理部件CPU

- (3) 简述数据在运算器和主存之间进行存/取访问的数据通路



中央处理部件CPU

- 6.2 设某计算机运算控制器逻辑图如图6.8，控制信号意义见表6.1，指令格式和微指令格式如下：

指令格式	操作码	rs,rd	rs1	imm或disp
------	-----	-------	-----	----------

微指令格式

1	2	...	23	24	...	25
←—— 控制字段 ——→				← 下址字段 →		

- 其中1-23位代表的1-23控制信号见表6.1。
- 试写出下述三条指令的微程序编码：
 - (1) JMP (无条件转移到(rs1)+disp)
 - (2) Load (从(rs1)+disp指示的内存单元取数，送rs保存)
 - (3) Store (把rs内容送到(rs1)+disp指示的内存单元)

中央处理部件CPU

■ 分析与解答：

	PC->AB	ALU->PC	PC+1	imm(dispatch)->ALU	DB->IR	DB->DR	DR->DB	rs1->GR	rs,rd->GR	(rs1)->ALU	(rs)->ALU	DR->ALU	+	-	>	<	ALU->GR	ALU->DR	ALU->AR	AR->AB	ADS	M/IO#	W/R#
JMP:		1		1				1		1			1						1			X	X
LOAD:				1				1		1			1						1			X	X
						1														1	1	1	
									1			1	1				1					X	X

中央处理部件CPU

STORE:

PC->AB	ALU->PC	PC+1	imm(displ)->ALU	DB->IR	DB->DR	DR->DB	rs1->GR	rs,rd->GR	(rs1)->ALU	(rs)->ALU	DR->ALU	+	-	>	<	ALU->GR	ALU->DR	ALU->AR	AR->AB	ADS	M/IO#	W/R#
			1				1		1			1						1			X	X
								1	1		1						1				X	X
						1													1	1	1	1

中央处理部件CPU

- 6.6 某机有8条微指令I1-I8，每条微指令所包含的微命令控制信号如表所示。

微指令	微命令信号									
	a	b	c	d	e	f	g	h	i	j
I1	✓	✓	✓	✓	✓					
I2	✓			✓		✓	✓			
I3		✓						✓		
I4			✓							
I5			✓		✓		✓		✓	
I6	✓							✓		✓
I7			✓	✓				✓		
I8	✓	✓						✓		

a-j分别对应10种不同性质的微命令信号。假设一条微指令的控制字段为8位，请安排微指令的控制字段格式。

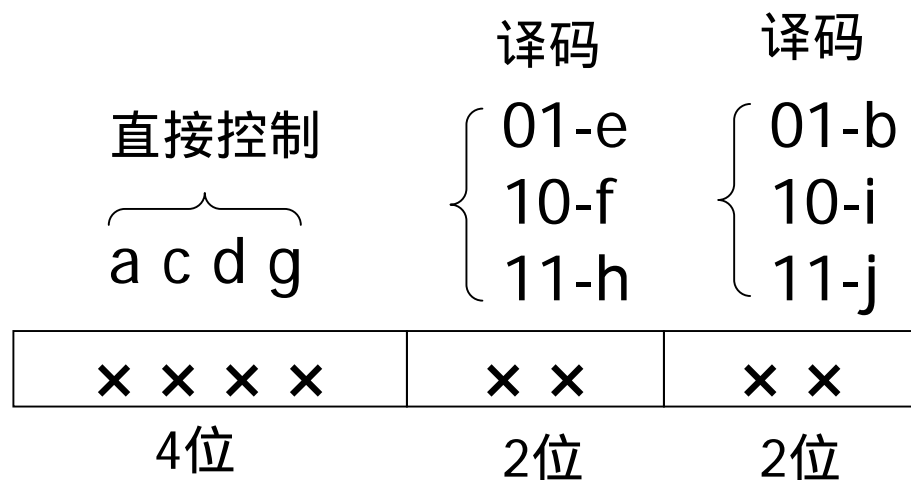
中央处理部件CPU

- 分析与解答：
 - 8位控制字段 < 10条微命令 => 使用“译码法”

微指令	微命令信号									
	a	b	c	d	e	f	g	h	i	j
I1	✓	✓	✓	✓	✓					
I2	✓			✓		✓	✓			
I3		✓						✓		
I4			✓							
I5			✓		✓		✓		✓	
I6	✓							✓		✓
I7			✓	✓				✓		
I8	✓	✓						✓		

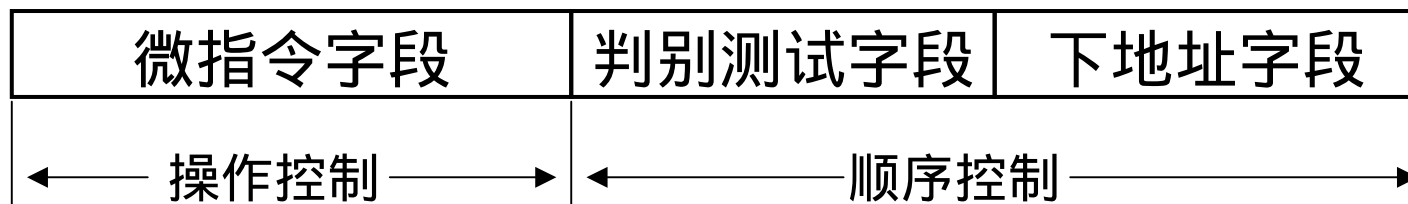
中央处理部件CPU

- 当a、c、d、g都为0时，产生译码信号



中央处理部件CPU

- 6.7 已知某机采用微程序控制方式，其控制存储器容量为 512×48 （位）。微指令字长为48位，微程序可在整个控制存储器中实现转移，可控制微程序转移的条件共4个（直接控制），微指令采用水平型格式，如图所示。



- (1) 微指令中的三个字段分别对应多少位？
- (2) 画出围绕这种微指令格式的微程序控制器逻辑框图



中央处理部件CPU

■ 分析与解答：

- 控制存储器容量为 512×48 、微指令字长位48位
=>控制存储器共有512个存储单元，完全寻址需要9位
- 4个直接控制的转移条件
=>占用4位

微指令字段	判别测试字段	下地址字段
← 35位 →	← 4位 →	← 9位 →



中央处理部件CPU

- 6.15 设有主频为16MHz的微处理器，平均每条指令的执行时间为两个机器周期，每个机器周期由两个时钟脉冲组成。问：
 - (1) 存储器为“0等待”，求机器速度
 - (2) 假如每两个机器周期中有一个是访存周期，需插入1个时钟周期的等待时间，求机器速度
(“0等待”表示存储器可在一个机器周期完成读/写操作，因此不需要插入等待时间)



中央处理部件CPU

■ 分析与解答：

- 平均每条指令的执行时间为两个机器周期，每个机器周期由两个时钟脉冲组成
- (1) 存储器为“0等待”时：
 $16\text{MHz} \Rightarrow 16\text{M脉冲/s} \Rightarrow 8\text{M机器周期/s} \Rightarrow 4\text{M指令周期/s} \Rightarrow 4\text{MIPS}$
- (2) 每两个机器周期中有一个是访存周期，需插入1个时钟周期的等待时间：
一个访存周期需要2个机器周期+另一个机器周期=3个机器周期 $\Rightarrow 6$ 个时钟脉冲
 $\Rightarrow 16/6\text{MIPS} = 2.67\text{MIPS}$

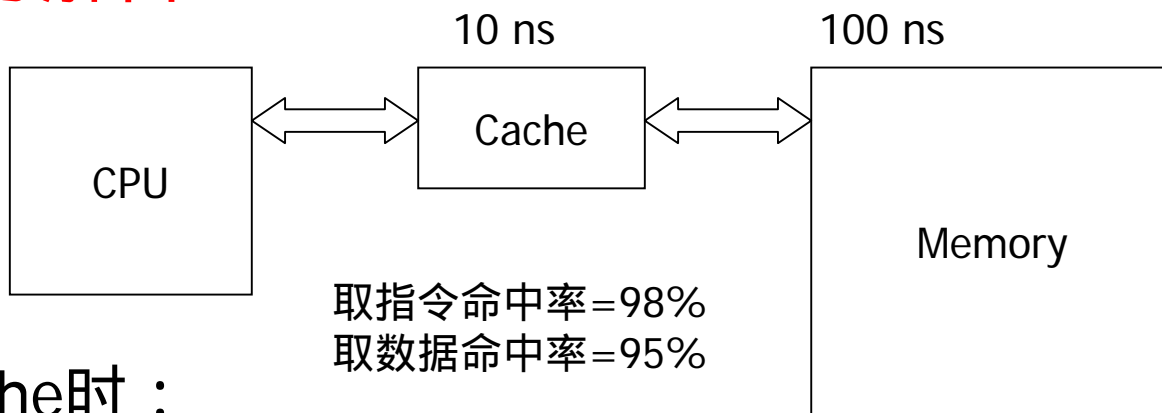


存储系统

- 7.3 设某流水线计算机有一个指令和数据合一的cache，已知cache的读/写时间为10ns，主存的读/写时间为100ns，取指的命中率为98%，数据的命中率为95%，在执行程序时，约有1/5指令需要存/取一个操作数，为简化起见，假设指令流水线在任何时候都不阻塞。问设置cache后，与无cache比较，计算机的运算速度可提高多少倍？

存储系统

■ 分析与解答：



有cache时：

- 取指令时间
 $10\text{ns} \times 98\% + (10\text{ns} + 100\text{ns}) \times 2\% = 12\text{ns}$
- 取数据时间
 $(10\text{ns} \times 95\% + (10\text{ns} + 100\text{ns}) \times 5\%) \times 1/5 = 3\text{ns}$
- 平均访存时间=取指令时间+取数据时间=15ns



存储系统

无cache时：

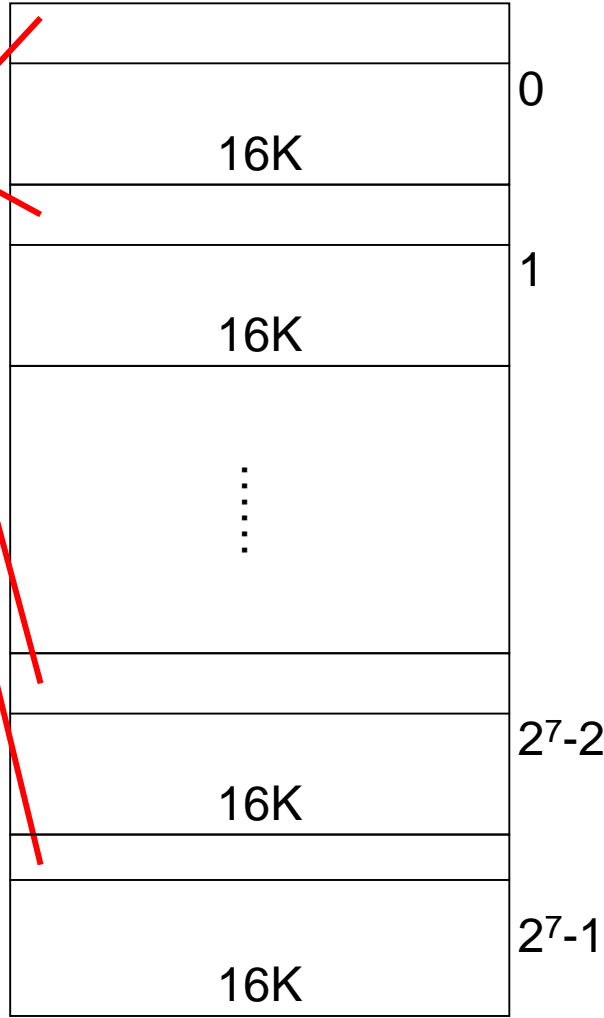
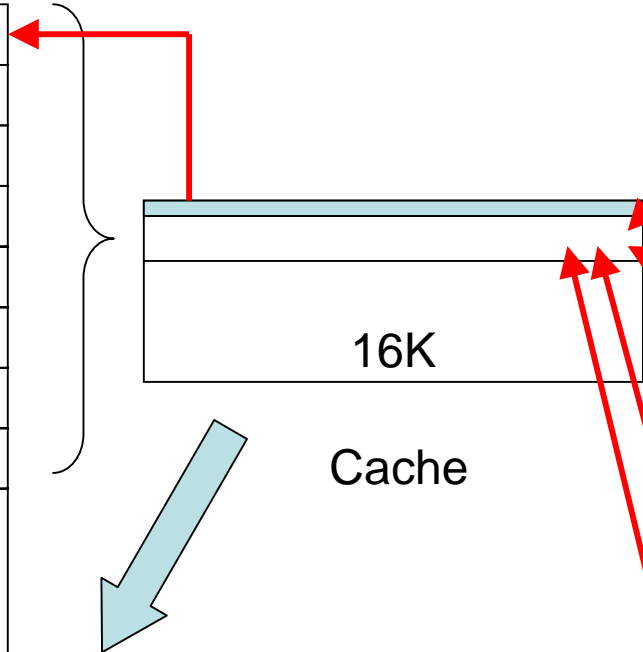
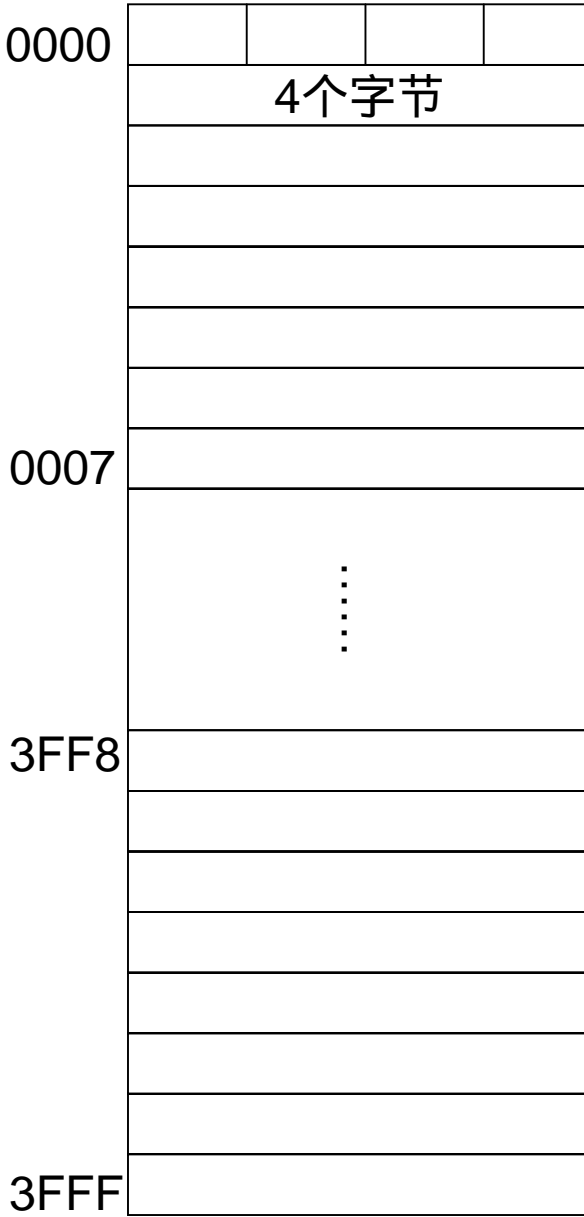
- 平均访存时间 = $100\text{ns} + 100\text{ns} \times 1/5 = 120\text{ns}$

运算速度提高 = $120\text{ns} / 15\text{ns} = 8\text{倍}$



存储系统

- 7.5 设某计算机的cache采用4路组相联映像，已知cache容量为16KB，主存容量为2MB，每个字块有8个字，每个字有32位。请回答：
 - (1) 主存地址多少位（按字节编址），各字段如何划分（各需多少位）？
 - (2) 设cache起始为空，CPU从主存单元0，1，...，100依次读出101个字（主存一次读一个字），并重复按此次序数读11次，问命中率为多少？若cache速度是主存的5倍，问采用cache与无cache比较速度提高多少倍？



Memory



存储系统

■ 分析与解答：

- (1) 主存容量 $2\text{MB} = 2 \times 2^{20}\text{B} = 2^{21}\text{B}$

=>主存地址21位

- 已知每个字块有8个字，每个字有32位，cache容量为16KB，主存容量为2MB：

- 每个字占4个字节 = 2^2 字节
- 每个字块有8个字 = 2^3 个字
- 把主存分成 $2\text{MB}/16\text{KB} = 2^7$ 块

主存高位地址	组号	块内地址	字节
9	7	3	2



存储系统

■ (2)

主存高位地址	组号	块内地址	字节
9	7	3	2

- 第1次读不命中，后10次读命中
=>命中率=10/11=91%
- 采用cache比无cache速度提高
$$=(11 \times 5)/(10 \times 1 + 1 \times 5) = 55/15 = 3.67 \text{倍}$$



存储系统

- 7.6 设某计算机采用直接映像cache，已知容量是4096B。
 - (1) 若CPU依次从主存单元0, 1, ..., 99和4096, 4097, ..., 4195交替取指令，循环执行10次，问命中率为多少？
 - (2) 如cache存取时间为10ns，主存存取时间为100ns，cache命中率为95%，求平均存取时间。

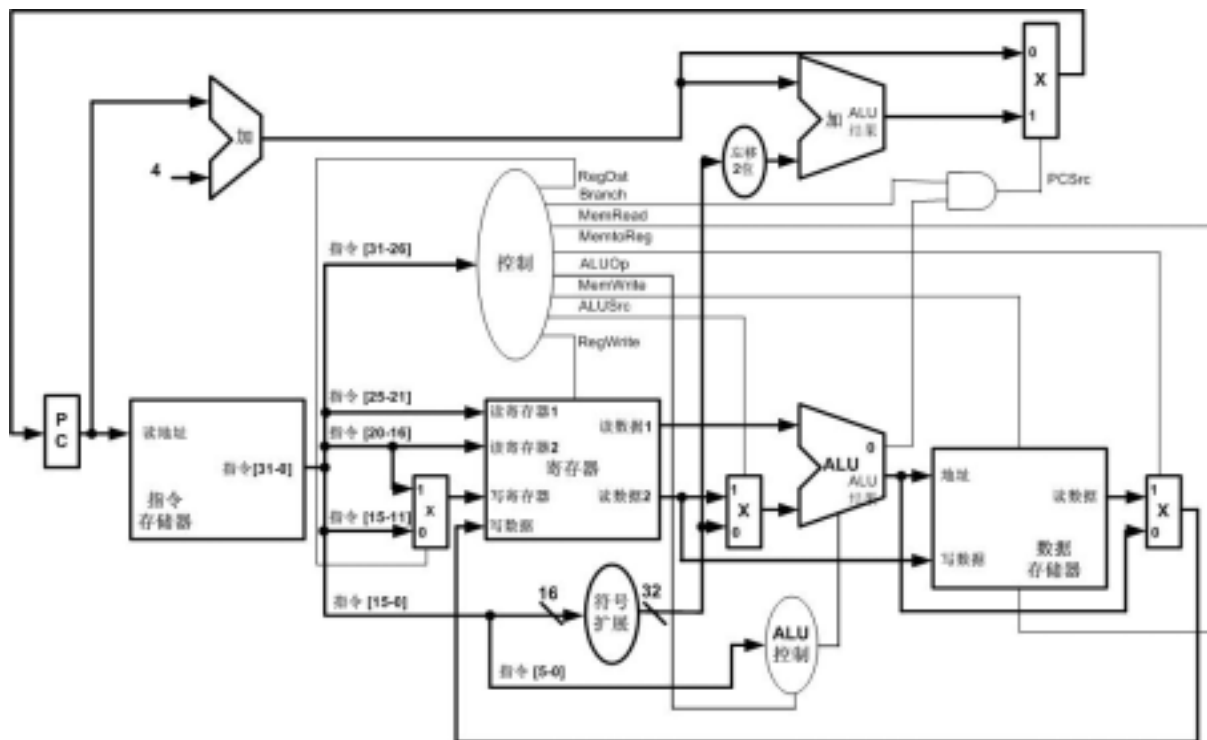


存储系统

- 分析与解答：
 - (1) 命中率=0
 - (2) 平均存取时间
 $= 10 \times 95\% + (100 + 10) \times 5\% = 15\text{ns}$

处理器：数据通路及其控制

- 5.5 希望给本章描述的单周期数据通路加入addi（立即数加）指令。给下图的单周期数据通路加入必要的数据通路和控制信号。





处理器：数据通路及其控制

- 分析与解答：
 - addi指令格式：

addi rt, rs, imm	8	rs	rt	imm
	6	5	5	16

$rs + imm \rightarrow rt$

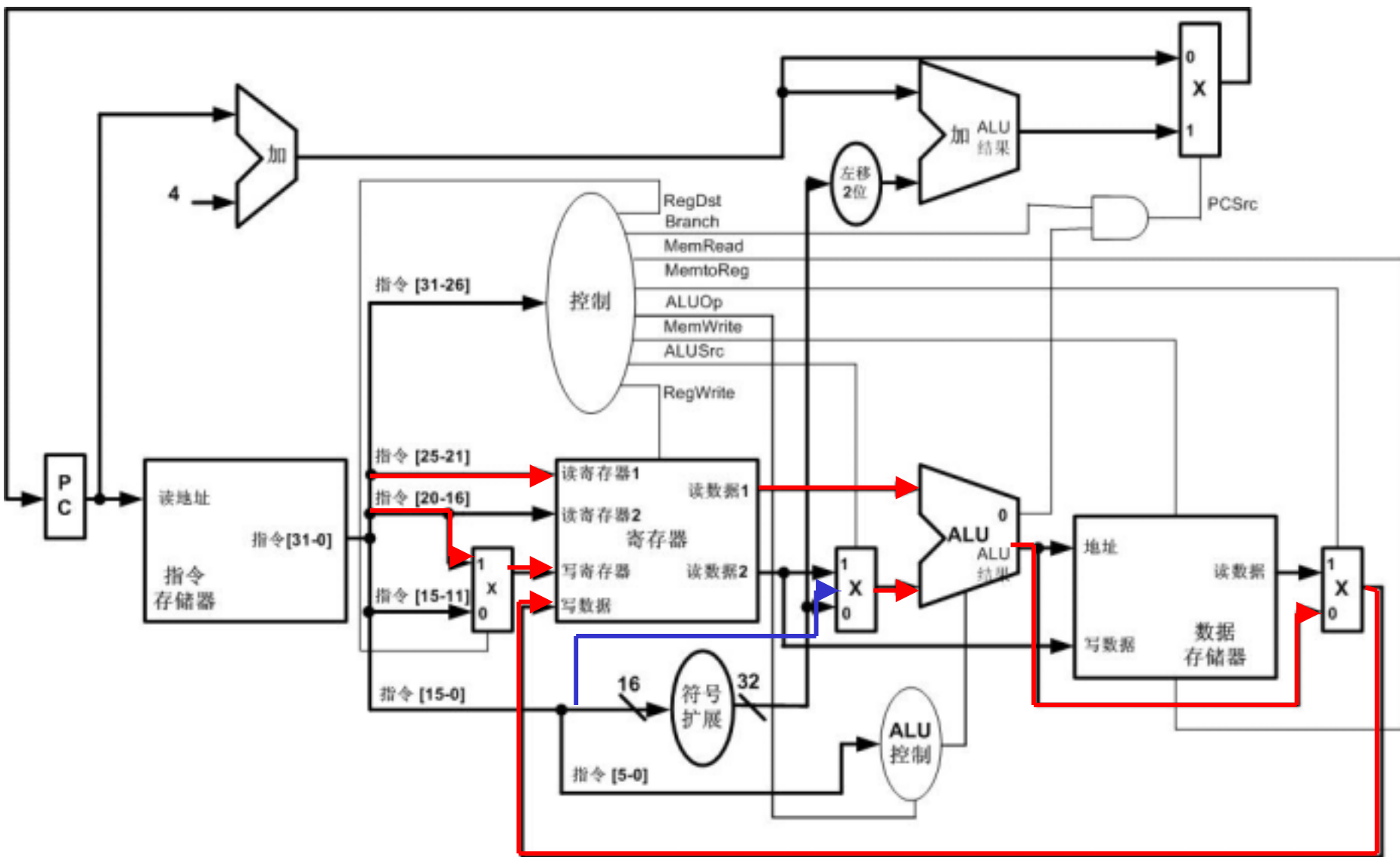
■ 分析与解答：

8

rs

rt

imm



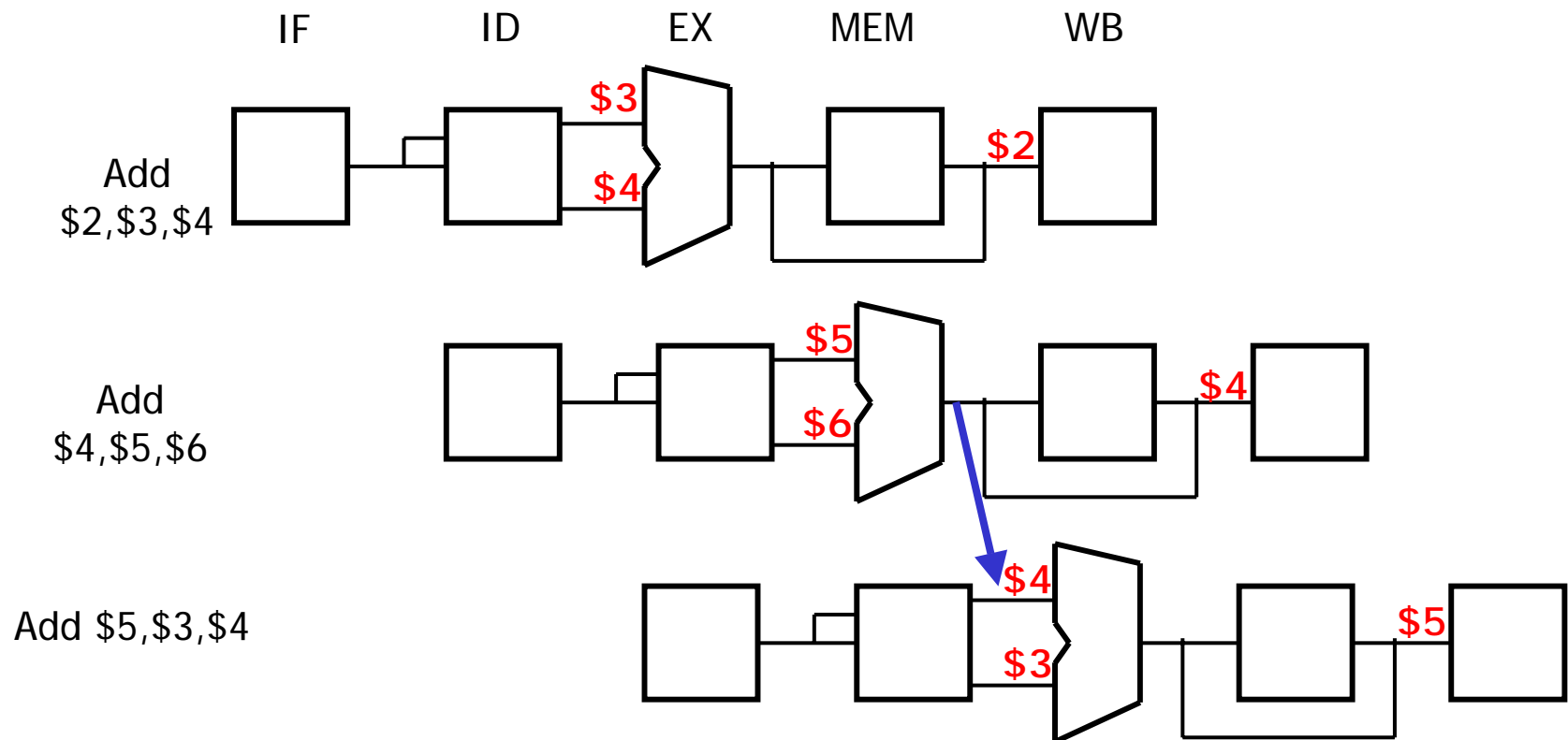


利用流水线提高性能

- 6.2 请使用流水线示意图，说明执行下列三条指令所需的转发路径：
 - Add \$2,\$3,\$4
 - Add \$4,\$5,\$6
 - Add \$5,\$3,\$4

利用流水线提高性能

■ 分析与解答：





Computer Architecture

—A Quantitative Approach

■ 1.10

Assume the two program in Figure 1.15 each **execute 100 million floating-point operations** during execution on each of the three machines.

If performance is expressed as a rate, then the average that tracks total execution time is the *harmonic mean*.

$$\frac{n}{\sum_{i=1}^n (1/\text{Rate}_i)}$$

where Rate_i is a function of $1/\text{Time}_i$, the execution time for the i th of n programs in the workload.



Computer Architecture

—A Quantitative Approach

- **Qa.** Calculate the MFLOPS rating of each program.
- **Qb.** Calculate the arithmetic, geometric, and harmonic means of MFLOPS for each machine.
- **Qc.** Which of the three means matches the relative performance of total execution time?

	Computer A	Computer B	Computer C
Program P1 (secs)	1	10	20
Program P2 (secs)	1000	100	20
Total Time	1001	110	40

Figure 1.15



Computer Architecture

—A Quantitative Approach

■ Answer a.

Number of floating-point operations in a program

$$\text{MFLOPS} = \frac{\text{Execution time in seconds} \times 10^6}{\text{Number of floating-point operations in a program}}$$

- Each executes 100 million floating-point operations

	Computer A		Computer B		Computer C	
Program	Time	MFLOPS	Time	MFLOPS	Time	MSLOPS
P1	1	100	10	10	20	5
P2	1000	0.1	100	1	20	5



Computer Architecture

—A Quantitative Approach

- **Answer** b.

$$\text{Arithmetic Mean} = \frac{\sum_{i=1}^n a_i}{n}$$

$$\text{Geometric Mean} = \left(\prod_{i=1}^n a_i \right)^{1/n}$$

$$\text{Harmonic Mean} = n / \sum_{i=1}^n (1/a_i)$$

	Computer		
Mean	A	B	C
Arithmetic	50.1	5.5	5.0
Harmonic	0.2	1.8	5.0
Geometric (normalized to A)	1.0	1.0	1.6
Geometric (normalized to B)	1.0	1.0	1.6
Geometric (normalized to C)	0.6	0.6	1.0



Computer Architecture

—A Quantitative Approach

- **Answer c.**
 - The arithmetic mean of MFLOPS rates trends inversely with total execution time.
 - The geometric means, regardless of which normalization is used, do not show each difference in total execution time.
 - The harmonic mean tracks total execution time **best**.

	Computer A	Computer B	Computer C
Program P1 (secs)	1	10	20
Program P2 (secs)	1000	100	20
Total Time	1001	110	40

Figure 1.15

Computer Architecture

—A Quantitative Approach

■ Appendix A.2

Use the following code fragment:

```
Loop: LD      F0,0(R2)
      LD      F4,0(R3)
      MUL.D   F0,F0,F4
      ADD.D   F2,F0,F2
      DADDUI  R2,R2,#8
      DADDUI  R3,R3,#8
      DSUBU   R5,R4,R2
      BNEZ    R5,Loop
```

$792/8 = 99$
iteration = 99 times

- Assume that the initial value of R4 is R2+792



Computer Architecture

—A Quantitative Approach

- For this exercise assume the standard **five stage** integer pipeline and the MIPS FP pipeline as described in section A.5. If structural hazards are due to write-back contention, assume the earliest instruction gets priority and other instructions are stalled.

Qa. Show the timing of this instruction sequence for the MIPS FP pipeline **without any forwarding** or bypassing hardware but assuming a register read and a write in the same clock cycle “forward” through the register file. Assume that the branch is handled by flushing the pipeline. If all memory references hit in the cache, how many cycles does this loop take to execute?



Computer Architecture

—A Quantitative Approach

- **Qb.** Show the timing of this instruction sequence for the MIPS FP pipeline **with normal forwarding** or bypassing hardware. Assume that the branch is handled by predicting it as not taken. If all memory references hit in the cache, how many cycles does this loop take to execute?
- See 3 Hazards & Forwarding
 - Structural Hazard
 - Data Hazard
 - Control Hazard

Computer Architecture

—A Quantitative Approach

■ Answer a. (without forwarding)

instruction	Clock cycle																										
	1	2	3	4	5	6	7	8	...	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27			
LD F0,0(R2)	F	D	E	M	W																						
LD F4,0(R3)		F	D	E		M	W																				
MUL.D F0,F0,F4			F	D	S	S	E	E	...	E	M	W															
ADD.D F2,F0,F2			F		S	S	D	S	...	S	S	S	E	E	E	E	M	W									
DADDUI R2,R2,#8					F	S			...	S	S	S	D	E	M	W											
DADDUI R3,R3,#8												F	D	E	M	W											
DSUBU R5,R4,R2												F	D	S	E	M	W										
<u>BNEZ R5,Loop</u>													F	S	D	S	r										
L.D F0,0(R2)																F	S	S	F	D	E	M	W				

Computer Architecture

—A Quantitative Approach

■ Answer a. (without forwarding)

	Clock cycle																										
instruction	1	2	3	4	5	6	7	8	...	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27			
LD F0,0(R2)	F	D	E	M		W																					
LD F4,0(R3)		F	D	E		M	W																				
MUL.D F0,F0,F4			F	D	s	s	E	E	...	E	M	W															
ADD.D F2,F0,F2			F		s	s	D	s	...	s	s	s	E	E	E	E	M	W									
DADDUI R2,R2,#8					F	s	...	s	s	s	D	E	M	W													
DADDUI R3,R3,#8											F	D	E	M	W												
DSUBU R5,R4,R2												F	D	s	E	M	W										
BNEZ R5,Loop													F	s	D	s	r										
L.D F0,0(R2)																F	s	s	F	D	E	M	W				

Loop 1

Loop 2 ...



Computer Architecture

—A Quantitative Approach

- **Answer** a.

total loop execution time
 $= 22 \times 99 = 2178$ clock cycles

Computer Architecture

—A Quantitative Approach

■ Answer b. (with normal forwarding)

instruction	Clock cycle																						
	1	2	3	4	5	6	7	...	12	13	14	15	16	17	18	19	20	21	22	23			
LD F0,0(R2)	F	D	E	M	W																		
LD F4,0(R3)		F	D	E	M	W																	
MUL.D F0,F0,F4			F	D	s	E	E	...	E	M	W	W											
ADD.D F2,F0,F2			F		s	D	s	...	s	E	E	E	E		M	W							
DADDUI R2,R2,#8					F	s	...		s	D	E	M	W										
DADDUI R3,R3,#8									F	D	E	M	W										
DSUBU R5,R4,R2									F	D	s	E	M	W									
<u>BNEZ R5,Loop</u>									F	s	D	r											
L.D F0,0(R2)														F	s	F	D	E	M	W			

Computer Architecture

—A Quantitative Approach

■ Answer b. (with normal forwarding)

instruction	Clock cycle																						
	1	2	3	4	5	6	7	...	12	13	14	15	16	17	18	19	20	21	22	23			
LD F0,0(R2)	F	D	E	M	W																		
LD F4,0(R3)		F	D	E	M	W																	
MUL.D F0,F0,F4			F	D	s	E	E	...	E	M	W	W											
ADD.D F2,F0,F2			F		s	D	s	...	s	E	E	E	E	E	M	W							
DADDUI R2,R2,#8					F	s	...		s	D	E	M	W										
DADDUI R3,R3,#8									F	D	E	M	W										
DSUBU R5,R4,R2										F	D	s	E	M	W								
BNEZ R5,Loop											F	s	D	r									
L.D F0,0(R2)														F	s	F	D	E	M	W			

Loop 1

Loop 2 ...

Loop 1

Loop 2 ...



Computer Architecture

—A Quantitative Approach

- **Answer** b.

total loop execution time

$$= 18 \times 98 + 19 = 1783 \text{ clock cycles}$$



Computer Architecture

—A Quantitative Approach

■ Appendix A.3

Suppose the branch frequencies (as percentage of all instructions) are as follows:

- Conditional branches 15%
- Jumps & Calls 1%
- Conditional branches 60% are taken

We are examining a **four-deep pipeline** where the branch is resolved at the end of the second cycle for unconditional branches and at the end of the third cycle for conditional branches. Assuming that **only the first pipe stage can always be done independent** of whether the branch goes and ignoring other pipeline stalls, how much faster would the machine be without any branch hazards?



Computer Architecture

—A Quantitative Approach

■ Answer

- Pipeline CPI = Ideal pipeline CPI + (Structural Hazard Stalls + Data Hazard Stalls + Control Hazard Stalls)

$$\text{Pipeline speedup} = \frac{1}{1 + \text{Pipeline stalls}} \times \frac{\text{clock cycle unpipelined}}{\text{clock cycle pipelined}}$$

$$= \text{Pipeline Depth} / (1 + \text{Pipeline stalls})$$

- No Control Hazard:

$$\text{Pipeline speedup}_{\text{ideal}} = 4 / (1 + 0) = 4$$

Computer Architecture

—A Quantitative Approach

- Having Control Hazard:
Assume 4 stage: IF, ID, EX and WB
 - Handle Jump & Call:

Instruction	Clock cycle					
	1	2	3	4	5	6
Jump or Call	IF	ID	EX	WB		
i+1		IF	IF	ID	EX	...
i+2			stall	IF	ID	...
i+3				stall	IF	...

Computer Architecture

—A Quantitative Approach

- Handle taken conditional branch:

Instruction	Clock cycle					
	1	2	3	4	5	6
Taken Branch	IF	ID	EX	WB		
i+1		IF	stall	IF	ID	...
i+2			stall	stall	IF	...
i+3				stall	stall	...



Computer Architecture

—A Quantitative Approach

- Handle not-taken conditional branch:

Instruction	Clock cycle					
	1	2	3	4	5	6
Not-taken Branch	IF	ID	EX	WB		
i+1		IF	stall	ID	EX	...
i+2			stall	↔ IF	ID	...
i+3				stall	IF	...



Computer Architecture

—A Quantitative Approach

- Summary of above 3 control flow instructions:

Control flow type	Frequency (per instruction)	Stall (cycles)
Jump & Call	1%	1
Conditional (taken)	$15\% \times 60\% = 9\%$	2
Conditional (not taken)	$15\% \times 40\% = 6\%$	1

$$\begin{aligned} \text{Pipeline Stall}_{\text{real}} \\ &= (1 \times 1\%) + (2 \times 9\%) + (1 \times 6\%) = 0.25 \end{aligned}$$

$$\begin{aligned} \text{Pipeline Speedup}_{\text{real}} \\ &= 4 / (1 + 0.25) = 3.2 \end{aligned}$$



Computer Architecture

—A Quantitative Approach

- Pipeline Speedup_{without control hazard}
= $4/3.2 = 1.25$
—— 25% speedup



Computer Architecture

—A Quantitative Approach

■ 5.19

Some memory systems handle TLB missed in software (as an exception), while others use hardware for TLB (Translation Lookaside Buffer) misses.



Computer Architecture

—A Quantitative Approach

- Qa.** What are the trade-off between two methods for handling TLB misses?
- Qb.** Will TLB miss handling in software always be slower than TLB miss handling in hardware? Explain.
- Qc.** Are there page table structures that would be difficult to handle in hardware, but possible in software? Are there any such structure that would be difficult for software to handle but easy for hardware to manage?



Computer Architecture

—A Quantitative Approach

Qd. Use the data from Figure 5.45 to calculate the penalty to CPI for TLB misses on the following workload **assuming hardware TLB handlers require 10 cycles per miss and software TLB handlers takes 30 cycles per miss**: (50% gcc, 25% perl, 25%ijpeg), (30% swim, 30% wave5, 20% hydro2d, 10% gcc).

Qe. Are the TLB miss times in part(d) realistic? Discuss.

Qf. Why are TLB miss rate for floating-point program generally higher then those for integer program?



Computer Architecture

—A Quantitative Approach

- **Answer a.**
Software is slower because of the overhead switch to the handler code, but the replacement algorithm can be higher than hardware and a wider variety of virtual memory organizations can be readily accommodated.
Hardware - faster but less flexible
- **Answer b.**
Factors affecting on the handling time include:
 - Page table – paged?
 - More efficient page table searching algorithm —— software
 - TLB entry prefetching —— hardware
- **Answer c.**
Page table structure that change dynamically would be difficult to handle in hardware but possible in software.



Computer Architecture

—A Quantitative Approach

		Cache misses per 1000 instructions		TLB misses per 1000 instr.
Program	CPI	I-Cache	L2-Cache	I-TLB
gcc	0.63	3.43	0.25	0.30
ijpeg	0.49	0.03	0.02	0.10
perl	0.56	1.66	0.09	0.26
swim	0.40	0.00	5.99	0.10
wave5	0.74	0.17	1.72	0.89
hydro2d	0.64	0.01	0.46	0.19

Adapted from Figure 5.45



Computer Architecture

—A Quantitative Approach

- **Answer** d.

Program	Weight	TLB misses/1000 instructions
gcc	50%	0.3
perl	25%	0.26
jpeg	25%	0.10



Computer Architecture

—A Quantitative Approach

- Workload miss rate
$$= \text{Weight}_i \times (\text{TLB misses}/1000_i)$$
$$= 50\% \times 0.3 + 25\% \times 0.26 + 25\% \times 0.1$$
$$= 0.24/1000 \text{ instructions}$$
- Penalty (Hardware)
$$= \text{WMR} \times \text{TLB miss handling time (10 cycles)}$$
$$= 2.4 \text{ cycles}/1000 \text{ instructions}$$
$$\text{CPI} = 0.0024 \text{ clocks/instruction}$$
- Penalty (Software)
$$= \text{WMR} \times \text{TLB miss handling time (30 cycles)}$$
$$= 7.2 \text{ cycles}/1000 \text{ instructions}$$
$$\text{CPI} = 0.0072 \text{ clocks/instruction}$$



Computer Architecture

—A Quantitative Approach

Program	Weight	TLB misses/1000 instructions
swim	30%	0.1
wave5	30%	0.89
hydro2d	20%	0.19
gcc	10%	0.3



Computer Architecture

—A Quantitative Approach

- Workload miss rate
$$= \text{Weight}_i \times (\text{TLB misses}/1000_i)$$
$$= 30\% \times 0.1 + 30\% \times 0.89 + 20\% \times 0.19 + 10\% \times 0.3$$
$$= 0.37/1000 \text{ instructions}$$
- Penalty (Hardware)
$$= \text{WMR} \times \text{TLB miss handling time (10 cycles)}$$
$$= 3.7 \text{ cycles}/1000 \text{ instructions}$$
$$\text{CPI} = 0.0037 \text{ clocks/instruction}$$
- Penalty (Software)
$$= \text{WMR} \times \text{TLB miss handling time (30 cycles)}$$
$$= 11.1 \text{ cycles}/1000 \text{ instructions}$$
$$\text{CPI} = 0.0111 \text{ clocks/instruction}$$



Computer Architecture

—A Quantitative Approach

- **Answer e.**

The TLB miss times are too small. Handling a TLB miss requires finding and transferring a page table entry in main memory to the TLB. A main memory access typically takes on the order of 100 clocks, already much greater than the miss time in part (d).

- **Answer f.**

Floating-point programs often traverse large data structures and thus more often reference a large number of pages. It is thus more likely that the TLB will experience a higher rate of capacity misses.



Computer Architecture

—A Quantitative Approach

■ 3.2

Consider the following four MIPS code fragments each containing two instructions:

- i. DADDI R1,R1,#4
 LD R2,7(R1)
- ii. DADD R3,R1,R2
 SD R2,7(R1)
- iii. SD R2,7(R1)
 SD F2,200(R7)
- iv. BEZ R1,place
 SD R1,7(R1)



Computer Architecture

—A Quantitative Approach

- **a.** For each fragment (i) to (iv) identify each type of dependence that exists or that may exist (a fragment may have no dependence) and describe what data flow, name reuse, or control structure causes or would cause the dependence. For a dependence that may exist, describe the source of the ambiguity and identify the time at which that uncertainty is resolved.
- **b.** For each code fragment, discuss whether dynamic scheduling is, may be, or is not sufficient to allow out-of-order execution of the fragment.



Computer Architecture

—A Quantitative Approach

■ Answer a.

Code fragment	Data Dependence?	Dynamic scheduling sufficient for out-of-order execution?
DADDI R1,R1, # 4 LD R2,7(R1)	True dependence of R1	No. Changing instruction order will break program semantics
DADD R3,R1,R2 SD R2,7(R1)	None	Yes
SD R2,7(R1) SD F2,200(R7)	Output dependence may exist	Maybe. If the hardware computes the effective addresses early enough, then the store order may be exchanged.
BEZ R1,place SD R1,7(R1)	None	No. Changing instruction order is speculative until the branch resolved



Computer Architecture

—A Quantitative Approach

■ 3.9

Increasing the size of a branch-prediction buffer means that it is less likely that two branch in a program will share the same predictor. A single predictor predicting a single branch instruction is generally more accurate than is that same predictor serving more than one branch instruction.

Qa. List a sequence of branch taken and not taken action to show a simple example of 1-bit predictor sharing that **reduces** misprediction rate.

Qb. List a sequence of branch taken and not taken action to show a simple example of how sharing a 1-bit predictor **increases** misprediction rate.

Computer Architecture

—A Quantitative Approach

■ Answer a.

	P	B1	P			B1	P			B1	P			B1		
	NT	<u>T</u>	T			<u>NT</u>	NT			<u>T</u>	T			<u>NT</u>		
Correct prediction?	--	no	--	--	--	no	--	--	--	no	--	--	--	no	--	--

	P	B1	P	B2	P	B1	P	B2	P	B1	P	B2	P	B1	P	B2
	NT	<u>T</u>	T	<u>NT</u>	NT	<u>NT</u>	NT	<u>T</u>	T	<u>T</u>	T	<u>NT</u>	NT	<u>NT</u>	NT	<u>T</u>
Correct prediction?	--	no	--	no	--	yes	--	no	--	yes	--	no	--	yes	--	no

■ Prediction Accuracy increases: 0% -> 50%



Computer Architecture

—A Quantitative Approach

■ Answer b.

	P	B1	P			B1	P			B1	P			B1		
	NT	<u>T</u>	T			<u>T</u>	T			<u>T</u>	T			<u>T</u>		
Correct prediction?	--	no	--	--	--	yes	--	--	--	yes	--	--	--	yes	--	--

	P	B1	P	B2	P	B1	P	B2	P	B1	P	B2	P	B1	P	B2
	NT	<u>T</u>	T	<u>NT</u>	NT	<u>T</u>	T	<u>NT</u>	NT	<u>T</u>	T	<u>NT</u>	NT	<u>T</u>	T	<u>NT</u>
Correct prediction?	--	no	--	no	--	no	--	no	--	no	--	no	--	no	--	no

■ Prediction Accuracy decreases: 100% -> 0%



Computer Architecture

—A Quantitative Approach

■ 3.14

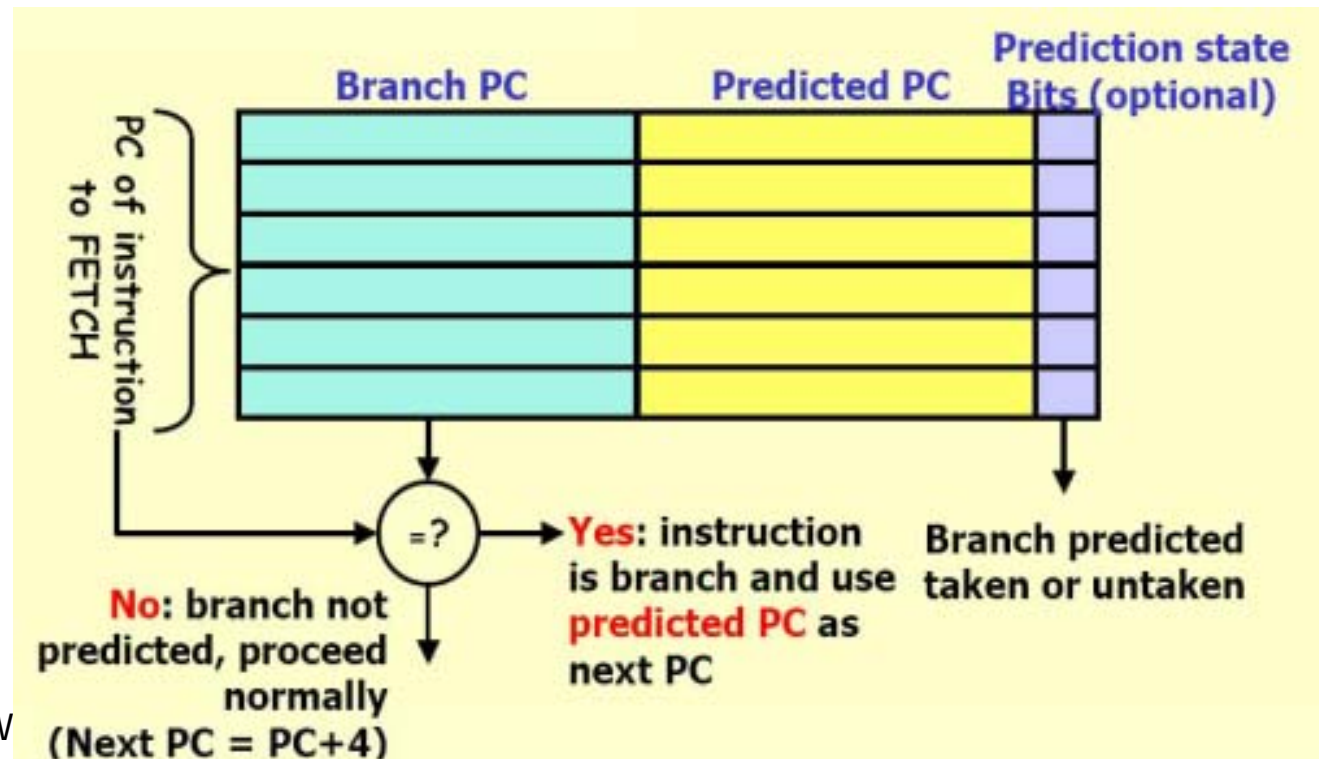
Suppose we have a deeply pipeline processor. For which we implement a **branch-target buffer** for the conditional branches only. Assume that the misprediction penalty is always 4 cycles and the buffer miss penalty is always 3 cycles. **Assume 90% hit rate and 90% accuracy and 15% branch frequency.**

Q. How much faster is the processor with the branch-target buffer versus a processor that has a **fixed 2 cycle branch penalty**? Assume a base CPI without branch stall of 1.

Computer Architecture

—A Quantitative Approach

- **Branch-Target Buffer (BTB):**
Address of branch index to get prediction AND
branch address (if taken)





Computer Architecture

—A Quantitative Approach

■ Answer

- CPI_{BTB} - System with a branch-target buffer
 CPI_{NBTB} - System without a branch-target buffer

- $$\text{Speedup} = \frac{CPI_{NBTB}}{CPI_{BTB}} = \frac{CPI_{\text{base}} + \text{Stall}_{NBTB}}{CPI_{\text{base}} + \text{Stall}_{BTB}}$$

$CPI_{\text{base}} = 1$ — exercise statement



Computer Architecture

—A Quantitative Approach

$$\text{Stall} = \sum_s \text{Stall}_s \times \text{Frequency}_s \times \text{Penalty}_s$$

$$\text{Stall}_{\text{NBTB}} = 15\% \times 2 = 0.3$$

$$\text{Stall}_{\text{BTB}} = 1.5\% \times 3 + 1.3\% \times 4 = 0.097$$

BTB result	BTB prediction	Frequency (per instruction)	Penalty (cycle)
Miss	--	$15\% \times 10\% = 1.5\%$	3
Hit	Correct	$15\% \times 90\% \times 90\% = 12.1\%$	0
Hit	Incorrect	$15\% \times 90\% \times 10\% = 1.3\%$	4

Assume 90% hit rate and 90% accuracy and 15% branch frequency



Computer Architecture

—A Quantitative Approach

$$\text{Speedup} = \frac{\text{CPI}_{\text{base}} + \text{Stall}_{\text{NBTB}}}{\text{CPI}_{\text{base}} + \text{Stall}_{\text{BTB}}} = \frac{1 + 0.3}{1 + 0.097} = 1.2$$

—— 20% faster

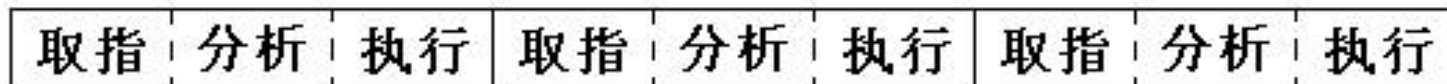


标量处理机习题

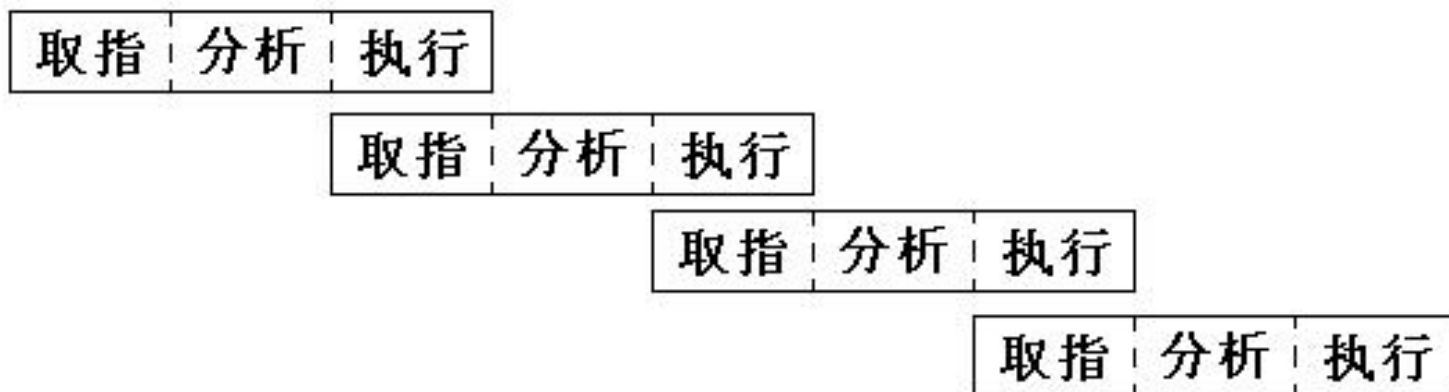
- 5.3 假设一条指令的执行过程分为“取指令”、“分析”和“执行”三段，每一段的时间分别是 t 、 $2t$ 和 $3t$ 。在下列各种情况下，分别写出连续执行 n 条指令所需要的时间表达式。
 - (1) 顺序执行方式。
 - (2) 仅“取指令”和“执行”重叠。
 - (3) “取指令”、“分析”和“执行”重叠。
 - (4) 先行控制方式。

标量处理机习题

■ 分析与解答



顺序方式工作的时间关系图



“执行”和“取指”重叠方式工作的时间关系图



标量处理机习题

- 顺序方式：

执行 n 条指令的时间

$$= n \times (t_{\text{取指}} + t_{\text{分析}} + t_{\text{执行}})$$

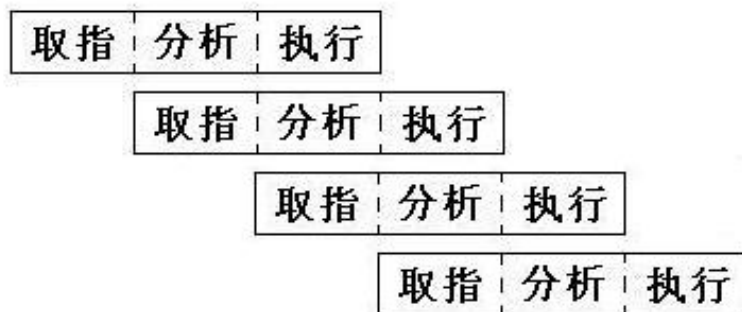
- “执行”和“取指”重叠：

执行 n 条指令的时间

$$= t_{\text{取指}} + n \times t_{\text{分析}} + \\ (n-1) \times \text{MAX} \{t_{\text{取指}}, t_{\text{执行}}\} + t_{\text{执行}}$$

标量处理机习题

■ 分析与解答



“执行”、“取指”和“分析”重叠方式工作的时间关系图



先行控制方式工作的时间关系图



标量处理机习题

- “执行”、“分析”和“取指”重叠：

执行 n 条指令的时间

$$= t_{\text{取指}} + \text{MAX} \{t_{\text{取指}}, t_{\text{分析}}\} + \\ (n-2) \times \text{MAX} \{t_{\text{取指}}, t_{\text{分析}}, t_{\text{执行}}\} + \\ \text{MAX} \{t_{\text{分析}}, t_{\text{执行}}\} + t_{\text{执行}}$$

- 先行控制：

执行 n 条指令的时间

$$= t_{\text{取指}} + t_{\text{分析}} + n \times t_{\text{执行}}$$



网络互连习题

- 7.3 设 16 个处理器编号分别为 0、1、...、15 , 要用单级互连网络。当互连函数分别为
 - (1) Cube_3
 - (2) PM2_{+3}
 - (3) PM2_{-0}
 - (4) Shuffle
 - (5) Shuffle (Shuffle)
- 时，第 13 号处理器各与哪一个处理器相连？



网络互连习题

■ 分析与解答

16 个处理器可用 4 位 2 进制 $P_3P_2P_1P_0$ 表示：

- (1) Cube_3 : $P_3P_2P_1P_0 \rightarrow P_3P_2P_1P_0$
- (2) PM2_{+3} : $P(j) \rightarrow P(j+2^3 \bmod 16)$
- (3) PM2_{-0} : $P(j) \rightarrow P(j-2^0 \bmod 16)$
- (4) Shuffle: $P_3P_2P_1P_0 \rightarrow P_2P_1P_0P_3$
- (5) Shuffle (Shuffle): $P_3P_2P_1P_0 \rightarrow P_1P_0P_3P_2$



网络互连习题

■ 分析与解答

第 13 号处理器 : 1101

- (1) Cube_3 : $1101 \rightarrow 0101 \Rightarrow (5)_{10}$
- (2) PM2_{+3} : $P(13) \rightarrow P(13+2^3 \bmod 16) = 5$
- (3) PM2_{-0} : $P(13) \rightarrow P(13-2^0 \bmod 16) = 12$
- (4) Shuffle: $1101 \rightarrow 1011 \Rightarrow (11)_{10}$
- (5) Shuffle (Shuffle): $1101 \rightarrow 0111 \Rightarrow (7)_{10}$



网络互连习题

- 7.4 在编号分别为 0、1、2、...、F 的 16 个处理器之间，要求按下列配对通信：
 - (B、1)，(8、2)，(7、D)，(6、C)，
 - (E、4)，(A、0)，(9、3)，(5、F)。

试选择所用互连网络类型、控制方式，并画出该互连网络的拓扑结构和各级交换开关状态图。

网络互连习题

■ 分析与解答

(B、 1)	(<u>1</u> <u>0</u> <u>1</u> <u>1</u> 、 <u>0</u> <u>0</u> <u>0</u> <u>1</u>)	
(8、 2)	(<u>1</u> <u>0</u> <u>0</u> <u>0</u> 、 <u>0</u> <u>0</u> <u>1</u> <u>0</u>)	
(7、 D)	(<u>0</u> <u>1</u> <u>1</u> <u>1</u> 、 <u>1</u> <u>1</u> <u>0</u> <u>1</u>)	
(6、 C)	(<u>0</u> <u>1</u> <u>1</u> <u>0</u> 、 <u>1</u> <u>1</u> <u>0</u> <u>0</u>)	$P_3P_2P_1P_0$
(E、 4)	(<u>1</u> <u>1</u> <u>1</u> <u>0</u> 、 <u>0</u> <u>1</u> <u>0</u> <u>0</u>)	\rightarrow
(A、 0)	(<u>1</u> <u>0</u> <u>1</u> <u>0</u> 、 <u>0</u> <u>0</u> <u>0</u> <u>0</u>)	$/P_3P_2/P_1P_0$
(9、 3)	(<u>1</u> <u>0</u> <u>0</u> <u>1</u> 、 <u>0</u> <u>0</u> <u>1</u> <u>1</u>)	
(5、 F)	(<u>0</u> <u>1</u> <u>0</u> <u>1</u> 、 <u>1</u> <u>1</u> <u>1</u> <u>1</u>)	



网络互连习题

■ 分析与解答

- $P_3P_2P_1P_0 \rightarrow /P_3P_2/P_1P_0$
 - 采用 Cube 网络，4 级控制
 - 第1、3级：交换状态
 - 第0、2级：直连状态
 - 级控制信号：0101（从右至左分别控制第 0 级至第 3 级）



网络互连习题

- 7.12 并行处理机有 16 个处理机，要实现相当于先 4 组 4 元交换，然后 2 组 8 元交换，再次是 1 组 16 元交换的交换函数功能，请写出此时各处理器之间所实现之互连函数的一般式，画出相应多级网络拓扑结构图，标出各级交换开关的状态。




网络互连习题

■ 分析与解答

	(0123 4567 89AB CDEF)
4组4元交换	(3210 7654 BA98 FEDC)
2组8元交换	(4567 0123 CDEF 89AB)
1组16元交换	(BA98 FEDC 3210 7654)

网络互连习题

■ 分析与解答

(0、 B)	(<u>0000</u> 、 <u>1011</u>)	
(1、 A)	(<u>0001</u> 、 <u>1010</u>)	
(2、 9)	(<u>0001</u> 、 <u>1001</u>)	
(3、 8)	(<u>0011</u> 、 <u>1000</u>)	 $P_3P_2P_1P_0$ -> $/P_3P_2/P_1/P_0$
(4、 F)	(<u>0100</u> 、 <u>1111</u>)	
(5、 E)	(<u>0101</u> 、 <u>1110</u>)	
(6、 D)	(<u>0110</u> 、 <u>1101</u>)	
(7、 C)	(<u>0111</u> 、 <u>1100</u>)	



SIMD 计算机

- 8.10 在 16 台 PE 的并行处理机上，要对存放在 M 分体并行存储器中的 16×16 二维数组实现行、列、主对角线、次对角线上各元素均无冲突访问，
 - 要求 M 至少为多少？
 - 此时数组在存储器中应如何存放？写出其一般规则。
 - 证明这样存放同时也可以无冲突地访问该二维数组中任意 4×4 子阵列的各元素。



SIMD 计算机

■ 分析与解答

- n 台 PE 的并行处理机，要对 $n \times n$ 二维数组实现行、列、主对角线、次对角线上各元素的同时无冲突访问，

要求：

存储器模数 M 是一个 $\geq n$ 的质数，

且 $M = 2^{2p} + 1$

数组中，同一列上两个相邻行的元素其地址错开的体号距离 d_1 为 2^p ，同一行上两个向量的元素其地址错开的体号距离 d_2 为 1



SIMD 计算机

- 分析与解答

- 本例中, $n = 16$,

- 模数 $= 17$,

- 又 $17 = 2^{2 \times 2} + 1$,

- $_1 = 2^2 = 4,$

 - $_2 = 1$