第一章

重点：

1. P2 1.1.1 计算机系统层次结构（**1.1.2** 透明性概念）

2. P9 1.2.1 计算机系统设计的定量原理（**Amdahl law** 及 **CPU** 性能公式）

3. P22 1.4.1 Von Neumann 结构（模拟与仿真）

1. 习题

(2) Amdahl 定律：系统中某一部件由于采用某种更快的执行方式后整个系统性能的提高与这种执行方式的使用频率或占总执行时间的比例有关。

Fe =（改进前可改进部分占用的时间）/（改进前整个任务的执行时间）

Se =（改进前改进部分的执行时间）/（改进后改进部分的执行时间）

则：

① 改进后的整个任务的执行时间为：

$$T_n = T_0\left(1 - Fe + \frac{Fe}{Se}\right)$$

其中，T_0 为改进前的整个任务的执行时间

② 改进后的整个系统加速比为：

$$S_n = \frac{T_0}{T_n} = \frac{1}{(1 - Fe) + Fe/Se}$$

(3) CPU 性能公式

CPU 时间 = CPU 时钟周期数/频率

CPU 时间 = CPU 时钟周期数 × 时钟周期长

每条指令平均时钟周期数 CPI = CPU 时钟周期数/IC（指令的条数）

CPU 时间 =（$IC \times CPI$）/频率 f

I_i = 指令 i 在程序中执行的次数

$$\text{CPU 的时钟周期数} = \sum_{i=1}^{n}(CPI_i \times I_i)$$

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{IC} = \sum_{i=1}^{n}\left(CPI_i \times \frac{I_i}{IC}\right)$$

题 1.5　硬件和软件在什么意义上是等效的?在什么意义上又是不等效的?试举例说明。

[解答]　硬件和软件在逻辑功能上是等效的。在原理上，用软件实现的功能完全可以用硬件或固件(微程序解释)来完成。用硬件实现的功能也可以通过用软件进行模拟来完成，只是反映在速度、价格、实现的难易程度上，这两者是不同的。

例如，编译程序、操作系统等许多用机器语言软件子程序实现的功能完全可以用组合电路硬件或微程序固件来解释实现。它们的差别只是软件实现的速度慢，软件的编制复杂，编程工作量大，程序所占的存贮空间量较多，这些都是不利的；但是，这样所花硬件少，硬件实现上也就因此而简单容易，硬件的成本低，解题的灵活性和适应性较好，这些都是有利的。又如，乘除法运算可以经机器专门设计的乘法指令用硬件电路或乘除部件来实现，也可以通过执行一个使用相加、移位、比较、循环等机器指令组成的机器语言子程序来实现。向量、数组运算在向量处理机中是直接使用向量、数组类指令和流水或阵列等向量运算部件的硬件方式来实现，但在标量处理机上也可以通过执行用标量指令组成的循环程序的软件方式来完成。

浮点数运算可以直接通过设置浮点运算指令用硬件来实现，也可以用两个定点数分别表示浮点数的阶码和尾数，通过程序方法把浮点数阶码和尾数的运算映象变换成两个定点数的运算，用于程序软的方式来实现。十进制数的运算可以通过专门设置十进制运算类指令和专门的十进制运算部件硬的方式来完成，或者通过设置 BCD 数的表示和若干 BCD 数运算的校正指令来软硬结合地实现，也可以先经 10 转 2 的数制转换子程序将十进制数转成二进制数，再用二进制运算类指令运算，所得结果又调用 2 转 10 的数制转换子程序转换成十进制数结果，用全软的方式实现。

题 1.7　什么是透明性概念?对于计算机系统结构，下列哪些是透明的?哪些是不透明的?

存贮器的模 m 交叉存取：浮点数据表示：I / O 系统是采用通道方式还是外围处理机方式；数据总线宽度；字符行运算指令；阵列运算部件；通道是采用结合型还是独立型：PDP—11 系列中的单总线结构；访问方式保护；程序性中断；串行、重叠还是流水控制方式；堆栈指令；存贮器的最小编址单位；Cache 存贮器。

[分析]　所谓透明就是看不到，不属于其管理的部分。对计算机系统结构是否透明，首先要弄清教材 1.2.1 节中有关计算机系统结构的定义和所包含的属性内容。简单来说，凡是编写机器语言和汇编语言程序要用到的数据表示、指令系统、寻址方式、寄存器组织、机器级 I / O 结构、存贮容量及其编址方式、中断机构、系统管态和目态间的切换、信息保护方式和机构等对计算机系统结构都是不透明的。而全部由硬件实现，或是在机器语言、汇编语言编程中不会出现和不需要了解的部分，以及只影响机器的速度和价格的逻辑实现(计算机组成)和物理实现(计算机实现)的那些部分，对计算机系统结构都是透明的。

[解答]　客观存在的事物或属性。从某个角度去看，却看不到，称这些事物和属性对他是透明的。透明了就可以简化这部分的设计，然而因为透明而无法控制和干预，就会带来不利。因此，透明性的取舍要正确选择。

对计算机系统结构透明的有：存贮器的模 m 交叉存取，数据总线宽度，阵列运算部件，通道是采用结合型还是独立型，PDP 一 11 系列的单总线结构，串行、重叠还是流水控制方式，Cache 存贮器。

对计算机系统结构不透明的有：浮点数据表示，I / O 系统采用通道方式还是外围处理机方式，字符行运算指令，访问方式保护，程序性中断，堆栈指令，存贮器最小编址单位。

题 1.8　从机器(汇编)语言程序员的角度来看，以下哪些是透明的?

指令地址寄存器；指令缓冲器；时标发生器；条件码寄存器；乘法器；主存地址寄存器；磁盘外设；先行进位链；移位器；通用寄存器；中断字寄存器。

[分析]　从机器(汇编)语言程序员看，实际上也就是从计算机系统结构看的内容。

指令地址寄存器就是程序计数器，汇编语言或机器语言程序都要用到它的，其位数多少会影响到可执行程序的空间大小。指令缓冲器、主存地址寄存器都属于计算机组成中的缓冲器技术，是由全硬件实现的，系统程序不参预对它们的管理。时标发生器、乘法器、先行进位链、移位器等都属于计算机组成中的专用部件配臀，它只影响机器的速度和价格，与软件编程无关。条件码寄存器是存放指令执行后生成反映结果状态或特征的标志码，它要供转移等指令使用，是编程要用到的。磁盘外设的种类、编址方式、容量等都是磁盘管理服务程序要用到的。通用寄存器的数量、位效、编址、使用规定在汇编语言程序和机器语言程序中都是会直接用到的。中断字寄存器是用来记录每一个中断类中，各个中断源发生中断请求的状况的，它是中断服务程序在处理中断时要用到的。

[解答] 从机器(汇编)语言程序员来看，透明的有：指令缓冲器，时标发生器，乘法器，主存地址寄存器，先行进位链，移位器。

题 1.9 下列哪些对系统程序员是透明的?哪些对应用程序员是透明的?

系列机各档不同的数据通路宽度；虚拟存贮器；Cache 存贮器；程序状态字；“启动 I／O”指令；“执行”指令；指令缓冲寄存器。

[分析] 系统程序员是编写诸如操作系统、编译程序等各种系统软件的人员。应用程序员是指利用计算机及所配的系统软件支持来编写解决具体应用问题的程序员。他们都可以使用汇编语言或机器语言来编写程序，当然也可以用高级语言来编写程序。所以，对系统程序员或应用程序员是不透明的，应包括计算机系统结构所包含的各个方面。而属全硬件实现的计算机组成所包含的方面，如系列机各档不同的数据通路宽度、Cache 存贮器、指令缓冲寄存器等，无论是对系统程序员，还是对应用程序员都应当是透明的。对目前高性能计算机系统来讲，大多数都是多用户环境，应用程序(也称算态，目态或用户态程序)中不允许使用管态(也林系统态，监督态)中所用的特权指令。

例如，大型多用户系统中，程序状态字是用于反映计算机系统在当前程序的各种关键状态(它并不是 IBM PC 计算机那种狭义的所谓程序状态字)的，它是操作系统用于管理计算机系统资源及其使用状况的，用户是不能直接对程序状态字内容进行读，写和访问的，只能由系统来管理。“启动 I／O”指令是大型机中的一种管态指令，属于特权指令，只能在操作系绕程序中使用(见教材中第 3 章 3.4.1 节的介绍)。用户程序是不能用它来直接启动 I／O 通道和设备的。虚拟存贮器(参看教材第 4 章 4.1.3 节)是一个主存——辅存两级存贮层次。它对应用程序员是完全透明的，使应用程序不必作任何修改就可以在系统上运行。但是，在操作系统中必须配置有相应的管理软件，能对其虚实外部地址的映象和变换、程序的换道、程序由辅存调入主存、主存页面的替换、存贮保护等进行管理，所以对系统程序员来说是不透明的。“执行”指令(参看教材中第 5 章 5.1.2 节)是 IBM 370 等系列机上用于解决程序在执行过程中不准修改指令，又允许将指令放在操作数区中作修改，以满足指令在执行过程中允许修改的要求。这种指令无论是用户程序，还是系统程序，都希望可以被使用，所以，“执行”指令应设计成对应用程序员和系统程序员都是不透明的。

[解答] 系列机各档不同的数据通路宽度、Cache 存贮器、指令缓冲寄存器属计算机组成，对系统程序员和应用程序员都是透明的。虚拟存贮器、程序状态字、“启动 I／O”指令，对系统程序员是不透明的，面对应用程序员却是透明的。“执行”指令则对系统程序员和应用程序员都是不透明的。

1.7

(1)从指定角度来看，不必要了解的知识称为透明性概念。

(2)见下表，“√”为透明性概念，“P”表示相关课文页数。

模 m 交叉，√，	浮点数据，×，P4	通道与 I/O 处理机，×，P4
总线宽度，√，	阵列运算部件，×，	结合型与独立型通道，√，
单总线，√，	访问保护，×，	中断，×，
指令控制方式，√，	堆栈指令，×，	最小编址单位，×，
Cache 存储器，√，		

1.8 见下表，“√”为透明性概念，“P”表示相关课文页数。

指令地址寄存器，×，	指令缓冲器，√，	时标发生器，√，
条件码寄存器，×，	乘法器，√，	主存地址寄存器，√，
磁盘，×，	先行进位链，√，	移位器，√，
通用寄存器 ，×，	中断字寄存器，×，	

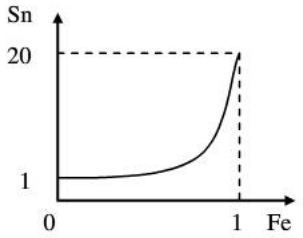
1.9 见下表，“√”表示都透明，“应”表示仅对应用程序员透明，“×”表示都不透明。

数据通路宽度，√，	虚拟存储器，应，	Cache 存储器，√，
程序状态字，×，	“启动 I/O”指令，应，	“执行”指令，×，
指令缓冲寄存器，√，		

1.12 已知 Se=20 ，求作 Fe-Sn 关系曲线。

将 Se 代入 Amdahl 定律得

$$S_n = \frac{1}{1 - \frac{19}{20} F_e}$$



1.13 上式中令 Sn=2，解出 Fe=10/19≈0.526

1.14 上式中令 Sn=10，解出 Fe=18/19≈0.947

1.15 已知两种方法可使性能得到相同的提高，问哪一种方法更好。

(1)用硬件组方法，已知 Se=40，Fe=0.7，解出 Sn=40/12.7≈3.1496（两种方法得到的相同性能）

(2)用软件组方法，已知 Se=20，Sn=40/12.7，解出 Fe=27.3/38≈0.7184（第二种方法的百分比）

(3)结论：软件组方法更好。因为硬件组需要将 Se 再提高 100%（20→40），而软件组只需将 Fe 再提高 1.84%（0.7→0.7184）。

1.17 $S_n = \frac{1}{0.1 + \frac{0.9}{5}} = \frac{5}{1.4} \approx 3.57$

1.18 记 f —— 时钟频率，T=1/f —— 时钟周期，B —— 带宽（Byte/s）。

方案一：$B_1 = \frac{1 \times 4}{T} = 4f\,(Byte\ /\ s)$

方案二：$B_2 = \frac{75\% \times 2 + 25\% \times 1}{2T} \times 4 = 3.5f\,(Byte\ /\ s)$

1.19 由各种指令条数可以得到总条数，以及各百分比，然后代公式计算。

$$IC = \sum_{i=1}^{4} IC_i = 10^5$$

(1) $CPI = \sum_{i=1}^{4} (CPI_i \times \frac{IC_i}{IC}) = 1 \times 0.45 + 2 \times 0.32 + 2 \times 0.15 + 2 \times 0.08 = 1.55$

(2) $MIPS = \frac{f}{CPI \times 10^6} = \frac{40 \times 10^6}{1.55 \times 10^6} = \frac{40}{1.55} \approx 25.806$

(3) $T = \frac{IC}{MIPS \times 10^6} = \frac{1.55}{400} \approx 0.003876$ （秒）

1.21

(1) $CPI = 1 \times 0.6 + 2 \times 0.18 + 4 \times 0.12 + 8 \times 0.1 = 2.24$

(2) $MIPS = \frac{f}{CPI \times 10^6} = \frac{40 \times 10^6}{2.24 \times 10^6} \approx 17.86$

1.24 记 Tc —— 新方案时钟周期，已知 CPI = CPI_i = 1

原时间 = CPI × IC × 0.95Tc = 0.95IC×Tc

新时间 = （0.3×2/3+0.7）× IC × Tc = 0.9IC×Tc

二者比较，新时间较短。

第二章

重点：P91 2.3.2.2Huffman 编码法

$$H = -\sum_{i=1}^{n} p_i \cdot \log_2 p_i$$

其中 P_i 表示第 i 种操作码在程序中出现的概率

固定长操作码相对于 Huffman 操作码的信息冗余量为：

$$R = 1 - \frac{-\sum_{i=1}^{n} p_i \cdot \log_2 p_i}{\lceil \log_2 n \rceil}$$

采用 Huffman 编码法操作码的最短平均长度可以通过如下公式计算：

$$H = \sum_{i=1}^{n} p_i \times l_i$$，其中 l_i 为第 i 种操作码

Huffman 操作码的主要缺点：

① 操作码长度很不规整，硬件译码困难。

② 与地址码共同组成固定长的指令比较困难。

（5）扩展编码法：由固定长操作码与 Huffman 编码法相结合形成。

2.3（忽略 P124 倒 1 行 ～ P125 第 8 行文字，以简化题意）已知 2 种浮点数，求性能指标。

此题关键是分析阶码、尾数各自的最大值、最小值。

原图为数据在内存中的格式，阶码的小数点在其右端，尾数的小数点在其左端，遵守规格化要求。

由于尾数均为原码，原码的绝对值与符号位无关，所以最大正数与最小负数的绝对值相同，可用“±最大绝对值”回答；最小正数与最大负数的绝对值相同，可用“±最小绝对值”回答。

第 1 小问中，阶码全部位数为 8，作无符号数看待真值为 0～255，作移-127 码看待真值为-127～+128；尾数（不计符号位）有 23 位小数，另加 1 位整数隐藏位，所以尾数绝对值为 1.0～2.0 − 2^{-23}，有效位数 p=24；

第 2 小问中，阶码全部位数为 11，作无符号数看待真值为 0～2047，作移-1023 码看待真值为-1023～+1024；尾数（不计符号位）有 52 位小数，另加 1 位整数隐藏位，所以尾数绝对值为 1.0～2.0 − 2^{-52}，有效位数 p=53。

最大绝对值为最大阶码与最大尾数绝对值的组合，最小绝对值为最小阶码与最小尾数绝对值的组合。代入相关公式后得最终结果如下表。

	32 位	64 位
±最大绝对值	$\pm(1-2^{-24})\cdot 2^{129}$	$\pm(1-2^{-53})\cdot 2^{1025}$
±最小绝对值	$\pm 2^{-127}$	$\pm 2^{-1023}$
表数精度δ	2^{-24}	2^{-53}
表数效率η	100%	100%

2.5

(1) $r_n = 2$，$r_e = 2$，$p = 24$（隐藏最高位），$q = 7$。

(2) $N_{max} = 1.7\times10^{38}$，$-|N|_{min} = -1.47\times10^{-39}$

$\delta \leqslant 5.96\times10^{-8} \approx 10^{-7.22}$，$\eta = 100\%$

2.6

(1) $0.2 = 0.333333H\times16^{0}$

设阶码为移-63 码（即-2^6+1，原题未指明）

$0.2 = 0.11001100110011001100110 1B\times2^{-2}$

（其中最高有效位需隐藏）

阶码为移-127 码（即-2^7+1）

1 位	7 位	6 位
0	0111111	333333

1 位	8 位	23 位
0	01111101	10011001100110011001101

(2) 符号位不变，（阶码 − 63）×4 + 127；尾数左规，除去最高位；

(3) 符号位不变，（阶码 − 127）/ 4 + 63；尾数补最高位，按除法余数右移若干位，左补 0。

2.11 $$\frac{20\times8+30\times16+20\times32+30\times64}{100\times64}=\frac{3200}{6400}=50\%$$

从地址的整数倍位置开始访问

20% |字节 8 位|浪费 8 位|半字 16 位|单子 32 位

2.5% |半字 16 位|半字 16 位|半字 16 位|半字 16 位|

30% |双字 64 位|

$$\frac{20\times56+(100-20)\times64}{100\times64}=95\%$$

2.13 已知 10 条指令使用频度，求 3 种编码方法的平均码长与信息冗余量。

(1)此问中的“最优 Huffman 编码法”实际是指码长下限，即信源的平均信息量——熵，代公式得 H=2.9566。

(2)Huffman 编码性能如下表；

公式：$-\sum_{i}^{10} pi\ \log_2 pi$

(3)2/8 扩展编码是 8/64/512 法的变种，第一组 2 条指令，码长为 2（1 位扩展标志，1 位编码），第二组 8 条指令，码长为 4（1 位扩展标志，与第一组区别，加 3 位编码），编码性能如下表； 00；01；1***；

(4)3/7 扩展编码是 15/15/15 法的变种，第一组 3 条指令，码长为 2（共有 4 种组合，其中 3 种组合分别代表 3 条指令，留 1 种组合作为扩展前缀标志），第二组 7 条指令，码长为 5（2 位固定的前缀扩展标志，与第一组区别，加 3 位编码，只用其中 7 种组合），编码性能如下表。 00；01；10；11***（只用 7 种）；

	Huffman 编码	2/8 扩展编码	3/7 扩展编码
平均码长 L	2.99	3.1	3.2
信息冗余量 R	1.10%	4.61%	7.59%

2.14

题意分析 本题目的是考察用 Huffman 编码设计指令系统，为此要构造 Huffman 树。

解答

(1) 要使得到的操作码长度最短，应采用 Huffman 编码，构造 Huffman 树如图 2－1 所示。

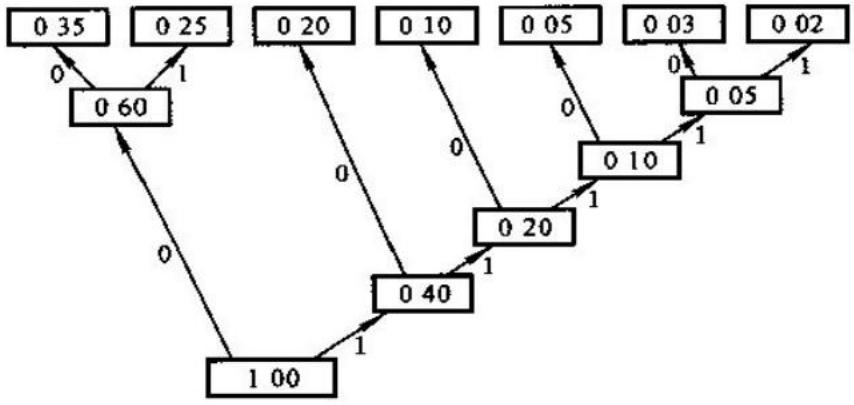


图 2－1 利用 Huffman 树进行操作码编码

由此可以得到 7 条指令的编码分别列于表 2－1。

表 2－1 指 令 编 码

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

这样，采用 Huffman 编码法得到的操作码的平均长度为：

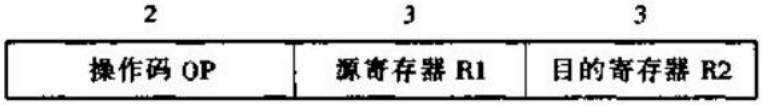
$$H = 2 \times (0.35 + 0.25 + 0.20) + 3 \times 0.10 + 4 \times 0.05 + 5 \times (0.03 + 0.02)$$

$= 1.6 + 0.3 + 0.2 + 0.25$

$= 2.35$

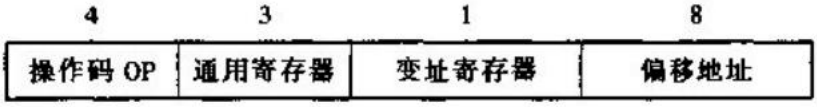
(2) 设计 8 位字长的寄存器—寄存器型变址寻址方式指令如下：

因为只有 8 个通用寄存器，所以寄存器地址需 3 位，操作码只有 2 位，设计格式如下：



3 条指令的操作码分别为 00,01,10。

设计 16 位字长的寄存器—存储器型变址寻址方式指令如下：



4 条指令的操作码分别为 1100，1101，1110，1111。

2.15

(1) 15 条/63 条/64 条　　(2) 14 条/126 条/128 条

说明：每种扩展刘两种组合：

共 14 { 0000 1101

1110 { 1110 000000 ..... 共 2^6-1 1110 111110　　扩充码 1110 111111 { 1110 111111 000000 ..... 共 2^6 1110 111111 111111

1111 { 1111000000 ..... 共 2^6-1 1111111110　　扩充码 1111 111111 { 1111 111111 000000 ....共 2^6 1111 111111 111111

2.18　P117

2.20　向后转移

```
(1) start:   move as,r1
             Mov num,r2
             dec r1
             inc r1
Loop:        move (r1),ad-as(r1)
             Dec r2
             Bgt loop
             Inc r1
             Halt
```

```
Num:        N
```
(2)N=100，循环 100 次，节省 100 个周期，循环体前后浪费 3 个周期，故能节省 97 个指令周期

```
（3）start:   move as,r1
              Mov num,r2
              Dec r2
              Dec r1
              Inc r1
Loop:         move (r1),ad-as(r1)
              Bgt loop
              Inc r1
              Dec r2
              Halt
Num:          N
```

第三章

难点：3.1.4.2 交叉访问存储器

重点：地址映射及替换算法

P146 3.2 虚拟存储器　　P174 3.3 Cache

3.2 T=H1T1+H2T2+H3T3；S=S1+S3+S2；C=(C1S1+C2S2+C3S3)/S

3.3 直接代公式计算存储层次性能指标。

(1)74ns，38ns，23.6ns　　H*t1+(1-h)*t2

(2)0.258，0.315，0.424　　(c1s1+c2s2)/(s1+s2)

(3)T256K < T128K < T64K

c256K > c128K > c64K

(4)T*C 分别得 19.092，11.97，10.0064。答案是 256K 方案最优。

3.5 已知 $\overline{K_n} = \frac{1-(1-g)^n}{g}$，其中 g=0.1

依题意有 $\overline{K_{n+1}} = \frac{1-(1-g)^{n+1}}{g} \geq \overline{K_n} + 0.2 = \frac{1-(1-g)^n}{g} + 0.2$

整理得 $0.9^n \geqslant 0.2$，解出 $n \leq \frac{\lg 0.2}{\lg 0.9} \approx 15.28$，向下取整，得 15；

按另一种题意理解是向上取整，得 16，也对。

3.7

3.9 $g = \frac{\log_2 Nv - \log_2 Np}{\log_2 Np - \log_2 Nd}$(向上取整　) =2,Nv:虚存大小；Np：页面大小；Nd：页表存储字大小

（2）4KB/4B=1K，故而二级页表空间为：4GB/1K=4MB，需 4MB/4KB=1024 页；

4MB/1K=4KB，故一级页表空间为 4KB，即 1 页

（3）一级页表必须驻立主存

3.10 令 TM 为主存的平均访问时间，TD 为硬盘的访问时间，则

T=HTM+(1-H)TD=(10000-9999*0.9999)TM=1.9999TM

D=TM/T=1/1.9999=50.0025%

3.12 (1) $U=\log_2 64=6$;$P=\log_2 1024=10$;$D=\log_2 4K=12$

(2)总数为 $\log_2 8M=23$;$D=\log_2 4K=12$,故实页号 p=23-12=11；

（3）快表：多用户虚页号（U+P）+实页号 p，即 16+11=17

（4）每个实页在页表中都存在一行与之对应，故共需 $2^{11}=2K=2048$（个存储字）；慢表包括主存页号（实页号）+装入位及其它标志位，即 11+1+其它

（5）P159 图 3.27

3.14

P=	2	3	2	1	5	2	4	5	3	2	5	2	命中次数
FIFO	2	2	2	2*	5	5	5*	5*	3	3	3	3	3
		3	3	3	3*	2	2	2	2*	2*	5	5	25%
				1	1	1*	4	4	4	4*	4*	2	
	入	入	中	入	换	换	换	中	换	中	换	换	

向每行回看，最大的为待换出的

LFU	2	2	2	2	2*	2	2	2*	3	3	3*	3*	5
		3	3	3*	5	5	5*	5	5	5*	5	5	41.67%
				1	1	1*	4	4	4*	2	2	2	
	入	入	中	入	换	中	换	中	换	换	中	中	

向页地址流回看，最后出现的为待换出的

OPT	2	2	2	2	2	2*	4*	4*	4*	2	2	2	6
		3	3	3	3*	3	3	3	3	3*	3*	3*	50%
				1*	5	5	5	5	5	5	5	5	
	入	入	中	入	换	中	换	中	中	换	中	中	

向页地址流后看，最远才访问的为待换出的

注：最好的办法是堆栈模拟。

3.15 欲知可能的最高命中率及所需的最少主存页数，较好的办法是通过“堆栈模拟法”，求得命中次数随主存页数变化的函数关系。下图就是“堆栈模拟图”，其中“√”表示命中。

P=	4	5	3	2	5	1	3	2	3	5	1	3	命中次数
	4	5	3	2	5	1	3	2	3	5	1	3	
		4	5	3	2	5	1	3	2	3	5	1	
			4	5	3	2	5	1	1	2	3	5	
				4	4	3	2	5	5	1	2	2	
						4	4	4	4	4	4	4	

n=1													0
n=2									√				1
n=3					√				√			√	3
n=4					√		√	√	√	√	√	√	7
n=5					√		√	√	√	√	√	√	7

(1)$H_{max}=7/12\approx58.3\%$

(2)n=4

(3)当 1 次页面访问代表连续 1024 次该页内存储单元访问时，后 1023 次单元访问肯定是命中的，而第 1 次单元访问的命中情况与这 1 次页面访问的命中情况相同。根据上图中最高命中情况，共有 7 次页命中（折算为 7×1024 次单元命中），5 次页不命中（折算为 5×1023 次单元命中，也可写为 5×1024-5），单元访问总次数为 12×1024，故有：

$H_{cell}=(12\times1024-5)/(12\times1024)=12283/12288\approx99.96\%$

改 LRU 替换算法：[分析] 由于 LRU 替换算法是堆栈型的替换算法，因而随着分配给该程序的实页数增加，实页命中率只会上升，至少是不会下降的。但是，当实页数增加到一定程度之后，其命中率就不会再提高了．如要再增加分配给该道程序的实页数，只会导致实存空间的利用率下降．所以，只要分别求出分配给该道程序不同实页数时的页命中率，找出达到最高命中率时所分配的最少实页数即可．

既然 LRU 替换算法是堆栈型的替换算法，对虚页地址流只需要用堆栈处理技术处理一次，就可以同时求出不同实页数时各自的命中率．这样，可以大大减少模拟的工作量。

[解答] 用堆栈对页地址流处理一次的过程见表 4．6 所示，其中 H 表示命中。

页地址流	4	5	3	2	5	1	3	2	2	5	1	3
S(1)	4	5	3	2	5	1	3	2	2	5	1	3
S(2)	4	5	3	2	5	1	3	3	2	5	1	
S(3)	4	5	3	2	5	1	1	3	2	5		
S(4)	4	4	3	2	5	5	1	3	2			
S(5)												
S(6)												
n=1									H			
实 n=2									H			
页 n=3					H				H			
数 n=4					H		H	H	H	H	H	H
n=5					H		H	H	H	H	H	H

模拟结果表明，使用 LRU 替换算法替换，对该程序至少应分配 4 个实页．如果只分配 3 个实页，其页命中率只有 2／12，太低：而分配实页数多于 4 页后，其页命中率不会再有提高．所以，分配给该程序 4 个实页即可，其可能的最高命中串为 H＝7／12.

3.15 加 1 题 一个二级存储层次，采用全相联映象和最久没有使用算法，实存共 5 页，为 2 道程序分享，页地址流分别如下

P_1 = 1 2 3 4 1 3 2 1

P_2 = 1 2 3 4 2 2 3 3

试作 2 个实存分配方案，分别使 2 道程序满足

(1)命中率相同；

(2)命中次数之和最大。

解：分别为 2 道程序作“堆栈模拟图”，其中“√”表示命中。

$P_1=$	1	2	3	4	1	3	2	1	命中次数 $N_{(1)}$
	1	2	3	4	1	3	2	1	
		1	2	3	4	1	3	2	
			1	2	3	4	1	3	
				1	2	2	4	4	

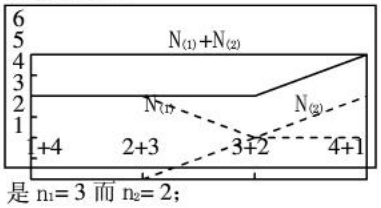
$n_1=1$									0
$n_1=2$									0
$n_1=3$						√		√	2
$n_1=4$					√	√	√	√	4

$P_2=$	1	2	3	4	2	2	3	3	命中次数 $N_{(2)}$
	1	2	3	4	2	2	3	3	
		1	2	3	4	4	2	2	
			1	2	3	3	4	4	
				1	1	1	1	1	

$n_2=1$						√		√	2
$n_2=2$						√		√	2
$n_2=3$					√	√	√	√	4
$n_2=4$					√	√	√	√	4

将两图结果综合，得到 4 个分配方案的命中率情况表如下

n_1	1	2	3	4
$N_{(1)}$	0	0	2	4
n_2	4	3	2	1
$N_{(2)}$	4	4	2	2
$N_{(1)}+N_{(2)}$	4	4	4	6



结论如下

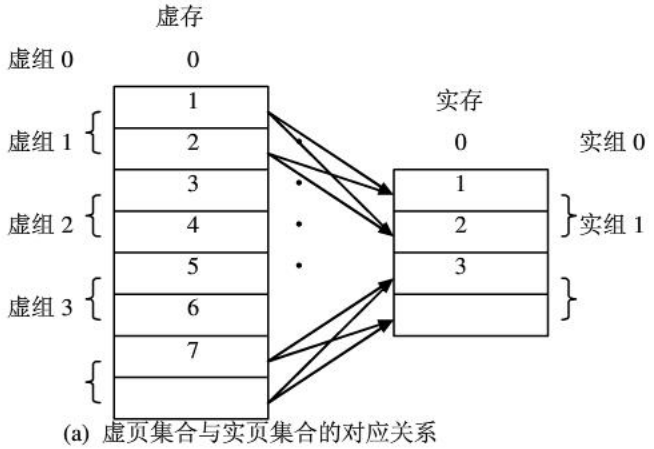
(1) 命中率相同的方案是 $n_1=3$ 而 $n_2=2$；

(2) 命中次数之和最大的方案是 $n_1=4$ 而 $n_2=1$。

3.19（1）区号：$\log_2 \frac{8}{2\times 2}=1$；格式为：|1E|1G|1B|4W|

（2）cache 格式为：|组号 g：1 位|组内块号：1 位|块内地址 W：4|

(3)



(a) 虚页集合与实页集合的对应关系

	实页	0	1	2	3
	0	√	√		
	1	√	√		
	2			√	√
虚页	3			√	√
	4	√	√		
	5	√	√		
	6			√	√
	7			√	√

(b) 对应关系表（√为有关系）

(4)通过作“实存状况图”模拟各虚块的调度情况，可获得 Cache 的块地址流序列。

P=	6	2	4	1	4	6	3	0	4	5	7	3
C0			4	4*	4	4	4	4*	4	4*	4*	4*
C1				1	1*	1*	1*	0	0*	5	5	5
C2	6	6*	6*	6*	6*	6	6*	6*	6*	6*	7	7*
C3		2	2	2	2	2*	3	3	3	3	3*	3
	入	入	入	入	中	中	替	替	中	替	替	中
C=	2	3	0	1	0	2	3	1	0	1	2	3

此问最容易出错的地方是忽略“组相联”地址约束，将虚页装错实组。另外没有及时标注“*”号也容易导致淘汰对象错误。

（5）

P=	6	2	4	1	4	6	3	0	4	5	7	3
C0			4	4*	4*	4*	4*	0	0*	5	5	5
C1				1	1	1	1	1*	4	4*	4*	4*
C2	6	6*	6*	6*	6*	6*	3	3	3	3	3*	3*
C3		2	2	2	2	2	2*	2*	2*	2*	7	7
	入	入	入	入	中	中	替	替	替	替	替	中

（7）LFU

P=	6	2	4	1	4	6	3	0	4	5	7	3
C0	6	6	6	6*	6*	6	6	6	6*	5	5	5
C1		2	2	2	2	2*	3	3	3	3*	7	7
C2			4	4	4	4	4	4*	4	4	4	4*
C3				1	1	1	1*	0	0	0	0*	3
	入	入	入	入	中	中	替	替	中	替	替	替

(6)H=4/12≈33%

(8)做法同 3.15 题(3)问，$H_{cell}=(12\times16-8)/(12\times16)\approx95.8\%$

3.21（2）一个主存周期从主存中取出数据为：4 体交叉×体字长 4B=16B，故 cache 块大小为 16B，共 1KB/16B=64 块，每组 4 块，故共 64/4=16 组，故 cache 的地址为：

|组号 g：4 位|组内块号 b：2 位|块内地址 W：4 位|

（1）主存的块与 cache 的组是直接映像，故主存每区有 16 块（cache 共 16 组），每块大小 16B，共有区数 1MB/16×16B=4K 个区。故主存地址为：|区号 E：12 位|区内块号 B：4 位|块内地址 W：4 位|

（3）cache 的每一组在块表中要有一行，故要有 16 行，由于有四个比较电路，故每行如下：

|区号 E|块号 b|区号 E|块号 b|区号 E|块号 b|区号 E|块号 b|；其中区号 E12 位，块号 b2 位

注：块表中并不包含所有的区，只有在 cache 中的区才在块表中有对应记录。

（4）每个比较电路的位数 12 位

（5）P183 图 3.48

3.23

引 1：在一个页式二级虚拟存贮器中，采用 FIFO 算法进行页面替换，发现命中率 H 太低，因此有下列建议：

(1) 增大辅存容量

(2) 增大主存容量(页数)

(3) 增大主、辅存的页面大小

(4) FIFO 改为 LRU

(5) FIFO 改为 LRU，并增大主存容量(页数)

(6) FIFO 改为 LRU，且增大页面大小

试分析上述各建议对命中率的影响情况。

[解答] (1) 增大辅存容量，对主存命中率 H 不会有什么影响。因为辅存容量增大，并不是程序空间的增大，程序空间与实主存空间的容量差并未改变。所以，增大物理辅存容量，不会对主存的命中率 H 有什么影响。

(2) 如果主存容量(页数)增加较多时，将使主存命中率有明显提高的趋势。但如果主存容量增加较少，命中率片可能会略有增大，也可能不变，甚至还可能会有少许下降。这是因为其前提是命中率 H 太低。如果主存容量显着增加，要访问的程序页面在主存中的机会会大大增加，命中率会显著上升。但如果主存容量(页数)增加较少，加上使用的 FIFO 替换算法不是堆栈型的替换算法，所以对命中率的提高可能不明显，甚至还可能有所下降。

(3) 因为前提是主存的命中率 H 很低，在增大主、辅存的页面大小时，如果增加量较小，主存命中率可能没有太大的波动。因为 FIFO 是非堆栈型的替换算法，主存命中事可能会有所增加，也可能降低或不变。而当页面大小增加量较大时，可能会出现两种相反的情况。当原页面大小较小时，在显著增大了页面大小之后，一般会使主存命中率有较大提高。但当原页面大小已较大时，再显著增大页面大小后，由于在主存中的页面数过少，将会使主存命中宰继续有所下降。

(4) 页面替换算法由 FIFO 改为 LRU 后，一般会使主存的命中率提高。因为 LRU 替换算法比 FIFO 替换算法能更好地体现出程序工作的局部性特点。然而，主存命中率还与页地址流、分配给主存的实页数多少等有关，所以，主存命中率也可能仍然较低，没有明显改进。

(5) 页面替换算法由 FIFO 改为 LRU，同时增大主存的容量(页数)，一般会使主存命中率有较大的提高。因为 LRU 替换算法比 FIFO 替换算法更能体现出程序的局部性，又由于原先主存的命中宰太低，现增大主存容量(页数)，一般会使主存命中率上升。如果主存容量增加量大些，主存命中率 H 将会显著上升。

(6) FIFO 改为 LRU，且增大页面大小时，如果原先页面大小很小，则会使命中率显著上升；如果原先页面大小已经很大了，因为主存页数进一步减少而使命中率还会继续有所下降。

引 2：采用组相联映象、LRU 替换算法的 Cache 存贮器，发现等效访问速度不高，为此提议：

(1）增大主存容量

(2）增大 Cache 中的块数(块的大小不变)

(3）增大组相联组的大小(块的大小不变)

（4）增大块的大小（组的大小和 Cache 总容量不变）

（5）提高 Cache 本身器件的访问速度

试问分别采用上述措施后，对等效访问速度可能会有什么样的显著变化？其变化趋势如何？如果采取措施后并未能使等效访问速度有明显提高的话，又是什么原因？

[分析] Cache 存储器的等效访问时间 $t_a = H_c t_c + (1-H_c)t_m$

等效访问速度不高，就是 t_a 太长。要想缩短 t_a，一是要使 H_c 命中率尽可能提高，这样 $(1-H_c)t_m$ 的分量就会越小，使 t_a 缩短，越来越接近于 t_c。但如果 t_a 已非常接近于 t_c 时，表明 H_c 已趋于 1，还想要提高等效访问速度，则只有减小 t_c，即更换成更高速的 Cache 物理芯片，才能缩短 t_a。另外，还应考虑 Cache 存贮器内部，在查映象表进行 Cache 地址变换的过程时，是否是与访物理 Cache 流水地进行，因为它也会影响到 t_a。当 H_c 命中率已很高时，内部的查映象表与访 Cache 由不流水改成流水，会对 t_c 有明显的改进，可缩短近一半的时间。所以，分析时要根据不同情况做出不同的结论。

如果 Cache 存贮器的等效访问速度不高是由于 H_c 太低引起的，在采用 LRU 替换算法的基础上，就要设法调整块的大小、组相联映象中组的大小，使之适当增大，这将会使 H_c 有所提高．在此基础上再考虑增大 Cache 的容量．Cache 存贮器中，只要 Cache 的容量比较大时，由于块的大小受调块时间限制不可能太大，增大块的大小一般总能使 Cache 命中率得到提高．

[解答] (1)增大主存容量，对 H_c 基本不影响．虽然增大主存容量可能会使 t_m 稍微有所加大，如果 H_c 已很高时，这种 t_m 的增大，对 t_a 的增大不会有明显的影响·

(2)增大 Cache 中的块数，而块的大小不变，这意味着增大 Cache 的容量．由于 LRU 替换算法是堆栈型的替换算法，所以，将使 H_c 上升，从而使 t_a 缩短．t_a 的缩短是否明显还要看当前的 H_c 处在什么水平上．如果原有 Cache 的块数较少，H_c 较低，则 t_a 会因 H_c 迅速提高而显著缩短．但如果原 Cache 的块数已较多，H_c 已很高了，则增大 Cache 中的块数，不会使 H_c 再有明显提高，此时其 t_a 的缩短也就不明显了。

(3)增大组相联组的大小，块的大小不变，从而使组内的块数有了增加，它会使块冲突突概率下降，这也会使 Cache 块替换次数减少．而当 Cache 各组组内的位置已全部装满了主存的块之后，块替换次数的减少也就意味着 H_c 的提高。所以，增大组的大小能使 H_c 提高，从而可提高等效访问速度。不过，Cache 存贮器的等效访问速度改进是否明显还要看目前的 H_c 处于什么水平。如果原先组内的块数太少，增大组的大小，会明显缩短 t_a；如果原先组内块数已较多，则 t_a 的缩短就不明显了。

(4)组的大小和 Cache 总容量不变，增大 Cache 块的大小，其对 t_a 影响的分析大致与(3)相同，会使 t_a 缩短，但要视目前的 H_c 水平而定。如果 H_c 已经很高了，则增大 Cache 块的大小对 t_a 的改进也就不明显了。

(5)提高 Cache 本身器件的访问速度，即减小 t_a，只有当 H_c 命中率已很高时，才会显著缩短 t_a。如果 H_c 命中率较低时，对减小 t_a 的作用就不明显了．

第四章

重点：P238 4.3.3 通道种类

(2) 通道可分为 3 类：字节多路通道、选择通道和数组多路通通。字节多路通道常用于连接低或中速的设备，选择通道和数据多路通道用于连接高速设备。

(3) 对于以上的 3 种通道，当每个通道上连接有 P 台外围设备，每台设备都传送 n 个字节时，总共所需的时间分别为：

$$T_{BYTE} = (T_S + T_D) \cdot P \cdot n$$

$$T_{SELECT} = (T_S/n + T_D) \cdot P \cdot n$$

$$T_{BLOCK} = (T_S/n + T_D) \cdot P \cdot n$$

其中 T_S 指设备选择时间，T_D 指传送一个字节所需的时间。

(4) 通道的流量是指一个通道在数据传送期间内，单位时间内能够传送的最大数据量又叫通道吞吐率。一个通道在满负荷工作下的流量称为通道最大流量。3 种通道的最大流量计算公式如下：

$$f_{MAX \cdot BYTE} = (P \cdot n)/[(T_S + T_D) \cdot P \cdot n] = 1/(T_S + T_D)$$

$$f_{MAX \cdot SELETE} = (P \cdot n)/[(T_S/n + T_D) \cdot P \cdot n] = 1/(T_S/n + T_D)$$

$$f_{MAX \cdot BLOCK} = (P \cdot n)/[(T_S/k + T_D) \cdot P \cdot n] = 1/(T_S/k + T_D)$$

(5) 字节多路通道的实际流量是指连接在这个通道上的所有设备的数据传输率之和。而选择通道和数据多路通道的实际流量是指连接在这个通道上的所有设备数据传输率的最大值。

$$f_{BYTE} = \sum_{i=1}^{P} f_i$$

$$f_{SELETE} = \max_{i=1}^{P} f_i$$

$$f_{BLOCK} = \max^{P} f_i$$

4.5 已知中断服务次序为 3-2-4-1，。

(1) 中断屏蔽字表如下图；

(2) 中断过程示意图如右图。

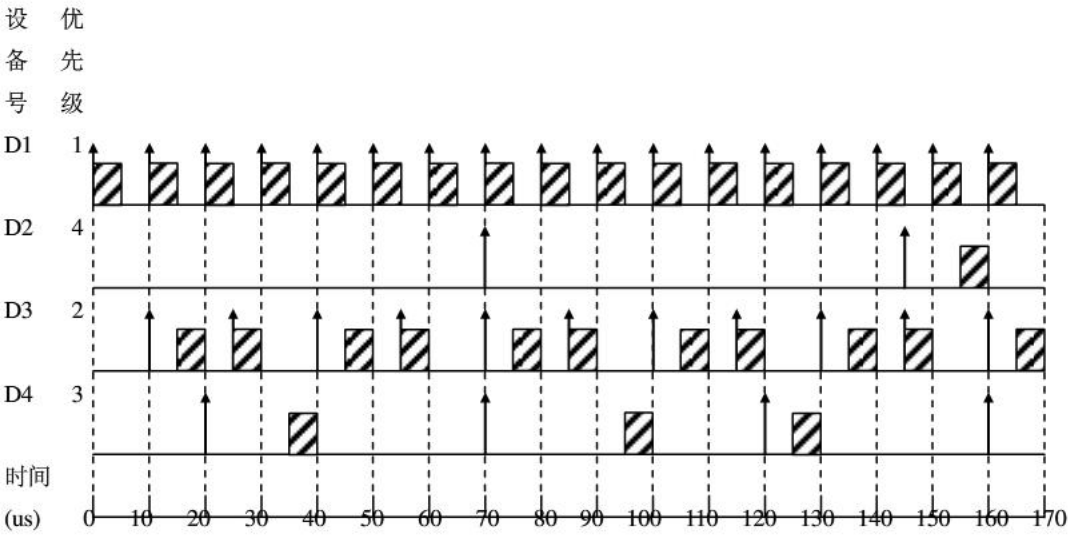
	D1	D2	D3	D4
D1	0	1	1	1
D2	0	0	1	0
D3	0	0	0	0
D4	0	1	1	0

4.8

(1) f=2×10^5字节/秒，T=5us

(2) Ts+Td=5us，通道时间图如下。作图时注意：至少要画到最慢设备的第

二次请求出现，才能确定是否丢失数据（因为响应优先级低的设备较易丢失数据）。



(3)5，160，20，40；

(4)D2 丢失第一次请求的数据；

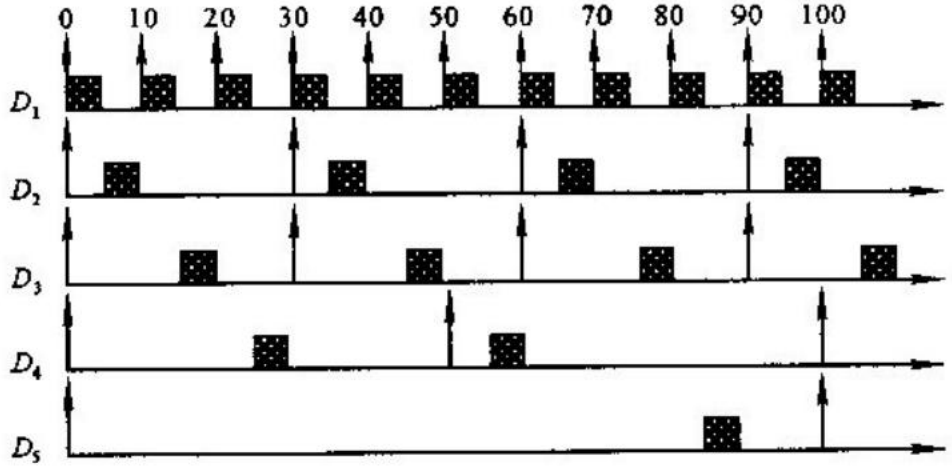
(5)参见 P245。

4.7

(1) 196.6 KB/s。 100+33.3+33.3+20+10

(2) 200 KB/s;5 μs。

(3) 通道分时工作的时间关系图如图 4－10 所示。通道处理完各设备第一次数据服务请求的时刻分别为:5 μs、10 μs、20 μs、30 μs、90 μs。



4.9

道道名称		实际流量(MB/s)	工作周期(μs)
字节多路通道	子通道 1	0.25	4
	子通道 2	0.25	4
	子通道 3	0.5	2
数组多路通通	数组多路通道 1	4	0.25
	数组多路通道 2	4	0 25
选择通通	选择通道 1	5	0.2
	选择通道 2	6	0.167

（2）I/O 流量：0.25+0.25+0.5+4+4+5+6=20MB/S

主存流量的另一部分包含指令的执行带宽需求，执行指令的速度 1GIPS×32/4=8GB/S，则取指令和取数据的带宽需求为： 1%×8GB/S=，

综合得，主存的数据传输率应为，周期为

第五章

重点：**P285** 线性流水线的性能分析

P294 非线性流水线的调度技术