1.1

1 层次结构：按照计算机语言从低级到高级的次序，把计算机系统按功能划分成多级层次结构，每一层以一种不同的语言为特征。这些层次依次为：微程序机器级，传统机器语言机器级，汇编语言机器级，高级语言机器级，应用语言机器级等。

计算机系统由硬件/器件和软件组成，按功能划分成多级层次结构

第 0 级级和第 1 级是具体实现机器指定功能的中央控制部分。

第 2 级是传统机器语言机器

第 3 级是操作系统机器。

第 4 级是汇编语言机器

第 5 级是高级语言机器

第 6 级是应用语言机器

2 计算机系统结构：传统机器程序员所看到的计算机属性，即概念性结构与功能特性。

3 计算机组成：计算机系统结构的逻辑实现，包含物理机器级中的数据流和控制流的组成以及逻辑设计等。

4 计算机实现：计算机组成的物理实现，包括处理机、主存等部件的物理结构，器件的集成度和速度，模块、插件、底板的划分与连接，信号传输，电源、冷却及整机装配技术等。

5 透明性：在计算机技术中，把这种本来存在的事物或属性，但从某种角度看又好像不存在的概念称为透明性。

6 由上往下设计：也称“由顶向低”设计。根据满足应用要求，定好面向应用的那个虚拟机器级的特性和工作环境，再逐级向下设计。由上向下设计师一种串行设计方法，设计周期长，是一种对环境要求比较稳定的专用机设计方法。

7 由下往上设计：也称“由底向顶”设计，根据目前能用的器件，参照、吸收已有各种机器的特点，将微程序机器级和传统机器级研制出来。然后加配适用的操作系统和编译系统软件。该类设计一体式串行设计，同样会延长设计周期，现在已经很少使用。

8 系列机：由同一厂家生产的具有相同系统结构、但具有不同组成和实现的一系列不同型号的计算机。

9 软件兼容：一个软件可以不经修改或者只需少量修改就可以由一台计算机移植到另一台计算机上运行。差别只是执行时间的不同。

10 兼容机：由不同公司厂家生产的具有相同系统结构的计算机。

11 模拟：用软件的方法在一台现有的计算机（称为宿主机）上实现另一台计算机（称为虚拟机）的指令系统。

12 仿真：用一台现有计算机（称为宿主机）上的微程序去解释实现另一台计算机（称为目标机）的指令系统。

13 虚拟机：用软件实现的机器。

14 宿主机：指要安装虚拟机软件的计算机。

如果 A 计算机上实现 B 计算机的指令系统，通常采用解释方法完成，即 B 机器的每一条指令用一段 A 机器的指令进行解释执行，如同 A 机器上也有 B 机器的指令系统一样。A 机器成为宿主机，被模拟的 B 机器成为虚拟机。

15 指令流：是指操作过程中涉及到的信息流动。 机器执行的指令序列

16 数据流：是一组有序，有起点和终点的字节的数据序列。

17Amdahl 定律：当对一个系统中的某个部件进行改进后，所能获得的整个系统性能的提高，受限于该部件的执行时间占总执行时间的百分比。

Amdahl 定律：加速比=（采用改进措施后的性能）/（没有采用改进措施前的性能）

=（没有采用改进措施前执行某任务的时间）/（采用改进措施执行某任

务的时间）

18CPI：每条指令执行的平均时钟周期数。 CPU 时钟周期数目/IC

19MIPS：单字长定点指令平均执行速度的缩写，每秒处理的百万级的机器语言指令数。每秒百万条指令数。=指令条数/执行时间 X10^6=时钟评率/CPIX10^6

20MFLOPS：衡量计算机系统的技术指标。程序中的浮点操作数/执行时间 X10^6 每秒百万次浮点数

1.2 解：这儿要注意的是第一级是最低的级别，而不是最高的级别。

第二级：NKns　　第三级：N^2Kns　　第四级：N^3Kns

1.3

可以加快操作系统操作命令解释的速度。同时也节省了存放解释操作命令这部分解释程序所占用的空间。简化了操作系统机器级的设计。也有利于减少传统机器级的指令条数。

1.4

答：第 2 级上等效程序需运行：(N/M)*Ks。第 3 级上等效程序需运行：(N/M)*(N/M)*Ks。第 4 级上等效程序需运行：(N/M)*(N/M)*(N/M)*Ks。

1.5

硬件和软件在实现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来实现用硬件实现的功能也可以用软件进行模拟来完成。但在速度、价格、实现的难易程度上是不同的。对于任何一种功能来说用软件实现的优点是设计容易、修改简单而且可以减少硬件成本。但其缺点是所实现的功能的速度较慢。用硬件实现的优点是速度快、性能高但它修改困难灵活性差。

等效：实现逻辑功能，原理上软件的功能可用硬件或者固件完成，硬件的功能也可以用软件模拟完成不等效：实现性能上

1.6

计算机系统结构、计算机组成和计算机实现是 3 个不同的概念。计算机系统结构是指令系统及其模型；计算机组成是计算机系统结构的逻辑实现；计算机实现是计算机组成的物理实现。它们各自包含不同的内容和采用不同的技术，但又有紧密的关系。

下面是具体比较：

(1) 计算机的系统结构相同，但可采用不同的组成。如 IBM370 系列

有 115、125、135、 158、 168 等 由低档到高档的多种型号机器。从汇编语言、机器语言程序设计者看到的概念性结构相同，均是由中央处理机/主存，通道、设备控制器，外设 4 级构成。其中，中央处理机都有相同的机器指令和汇编指令系统，只是指令的分析、执行在低档机上采用顺序进行，在高档机上采用重叠、流水或其它并行处理方式。

(2)相同的组成可有多种不同的实现。如主存器件可用双极型的，也可用 MOS 型的;可用 VLSI 单片，也可用多片小规模集成电路组搭。

(3)计算机的系统结构不同，会使采用的组成技术不同，反之组成也会影响结构。

如为实现 A:=B+CD:=E*F,可采用面向寄存器的系统结构，也可采用面向主存的三地址寻址方式的系统结构。要提高运行速度，可让相加与相乘并行，为此这两种结构在组成上都要求设置独立的加法器和乘法器。但对面向寄存器的系统结构还要求寄存器能同时被访问，而对面向主存的三地址寻址方式的系统结构并无此要求,倒是要求能同时形成多个访存操作数地址和能同时访存。又如微程序控制是组成影响结构的典型。通过改变控制存储器中的微程序，

就可改变系统的机器指令，改变结构。如果没有组成技术的进步，结构的进展是不可能的。

综上所述，系统结构的设计必须结合应用考虑，为软件和算法的实现提供更多更好的支持，同时要考虑可能采用和准备采用的组
能方便地在低档机上用简单便宜的组成实现，又能在高档机上用复杂较贵的组成实现，这样，结构才有生命力;组成设计上面决定于结构，下面受限于实现技术。然而，它可与实现折衷权衡。例如，为达到速度要求，可用简单的组成但却是复杂的实现技术，也可用复杂的组成但却是一般速度的实现技术。前者要求高性能的器件，后者可能造成组成设计复杂化和更多地采用专用芯片。
组成和实现的权衡取决于性能价格比等因素;结构、组成和实现所包含的具体内容随不同时期及不同的计算机系统会有差异。软件的硬化和硬件的软件都反映了这一事实。VLSI 的发展更使结构组成和实现融为一体，难以分开。

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，√，		

答：透明指的是客观存在的事物或属性从某个角度看不到。

透明的有：存储器的模 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-11 系列的单总线结构串行、重叠还是流水控制方式；Cache 存储器。

不透明的有：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；字符行运算指令；访问方式保护；程序性中断；；堆栈指令；存储器最小编址单位。

P.S.

属于计算机系统结构的属性有：数据表示、寻址方式、寄存器组织、指令系统、存储组织、中断机构、I/O 结构、保护机构等。

属于组成的属性有：数据通路宽度、专用部件设置、功能部件并行度、控制机构的组成方式，可靠性技术等。它着眼于机器内各事件的排序方式，控制机构的功能及部件间的关系。

属于实现的属性有：部件的物理结构、器件、模块的划分与连接、微组装技术、信号传输技术等，它着眼于器件技术和微组装技术。

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，√，		

1.9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；虚拟存储器；**cache** 存储器；程序状态字；“启动 **i/o**”指令；“执行”指令；指令缓冲器。答：①对系统程序员和应用程序员均透明的：是全用硬件实现的计算机组成所包含的方面。有：数据通路宽度、**cache** 存储器、指令缓冲器。

②仅对应用程序员透明的：是一些软硬件结合实现的功能。有：虚拟存储器、程序状态字、“启动 **i/o**”指令。③均不透明的：“执行”指令。

1.10

（1） 采用系列机方法，只能在具有相同系统结构的各种机器之间实现软件移植，一般是一个厂家生产的机器

（2） 采用模拟与仿真的方法，可在不同系统结构的机器之间相互移植软件，对于使用频率较高的指令，尽可能用仿真方法以提高运算速度，而对于使用频率低且难于用仿真实现的指令则用模拟方法来实现。

（3） 采用统一的高级语言方法，可以解决结构相同或完全不同的各种机器上的软件移植，

但是，要统一高级语言，语言的标准化很重要，但难以在短期内解决。

1.11 解：系列机是指由同一厂家生产并具有相同系统结构的计算机，但具有不同的计算机组成与实现。

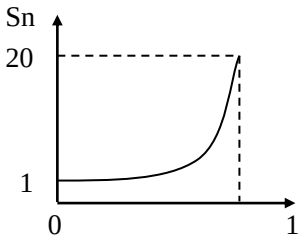
可行：(1) (3) (4) (6) (7)

不可行：(2) (5) (8)

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）。

1.16

解:

对该应用程序来说，在 90%的时间里，只有 50000*10%=5000 条指令在运行，其他的 45000 条指令的平均运行次数很少，因此，可以假设对它们来说，Cache 总是缺失的.

对频繁访问的这 10%的指令，假设它们访问均匀，这样，Cache 的行为便可以认为是均匀覆盖了这些指令. 所以，

10%的指令承担了 90%的指令访问, 指令访问次数(50000*10%)/90%

命中次数 2000

Cache 的命中率为：H=2000/[(50000*10%)/90%]=0.36

1.17

假设高速缓存 Cache 工作速度为主存的 5 倍，且 Cache 被访问命中的概率为 90%，则采用 Cache 后，能使整个存储系统获得多高的加速比？

解：

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

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

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

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

$$\text{实际带宽} = \frac{25\%\times 1+75\%\times 2}{2} = 0.875\text{字/时钟周期}$$

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

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

1.20

解：(a) f==15MHz ，MIPS=10，每次存取时间为 2 个时钟周期

有效$CPI = \dfrac{f}{MIPS \times 10^6} = \dfrac{15 \times 10^6}{10 \times 10^6} = 1.5$

(b) $f = 30MHz$,存储系统的速率不变，但每次存取为2个时钟周期

30%指令每条只需要一次存储存取，改进前共需 1 周期，改进后共需 2 周期
而另外 5%每条需要两次存储存取，改进前共需 2 周期，改进后共需 4 周期

$$CPI_{新} = CPI_{原} + 30\% \times (2-1) + 5\% \times (4-2) = 1.9$$

$$MIPS = \frac{f}{CPI \times 10^6} = \frac{30 \times 10^6}{1.9 \times 10^6} = 15.8$$

$$Sn = \frac{T_{原}}{T_{新}} = \frac{I_C \times CPI_{原} \times \tau_{原}}{I_C \times CPI_{新} \times \tau_{新}} = \frac{1.5 \times 30}{1.9 \times 15} = 1.58$$

1.21

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

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

1.22

1.（6 分）

因为 MIPS=$\frac{IC}{T * 10^6}$

所以每台计算机每个程序得 MIPS 速率如下表所示：

程序	MIPS 速率（百万指令/秒）		
	计算机 A	计算机 B	计算机 C
程序 1	50	5	2.5
程序 2	0.05	0.5	2.5
程序 3	0.1	0.05	1
程序 4	0.5	0.0625	0.5

由上述 MIPS 速率可知，每个计算机对四个程序有不同的处理时间，而且大小顺序不同，所以不能得出明确结论。

2.（4 分）

可以采取平均的方法来比较各计算机的相对性能：

平均执行时间	MIPS 速率（百万指令/秒）		
	计算机 A	计算机 B	计算机 C
算术平均（AM）	12.65	1.405	1.625
几何平均（GM）	0.595	0.295	1.33
调和平均（HM）	0.125	0.10	1.05

如果按照算术平均 AM 比较性能，计算机 A 最快，计算机 C 最慢，如果按照调和平均 HM 比较性能，结果恰好相反。

1.23

算术平均=（1/10）

（10.07+8.9+8.3+5.0+8.7+9.0+9.7+11.1+7.8+5.6）

=8.48MFLOPS

几何平均=$\sqrt[10]{(\Pi(ETR_i))}$ =8.247MFLPOS

调和平均=7.99

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。

1.28 解：原始 MFLOPS ＝ 195578/（10.8×106）＝0.018

正则化后 MFLOPS ＝195578/（13.6×106）＝0.014

指令正则化后的具体值 ＝ f/CPI ＝ 16.6M/（6×106）＝ 2.77

2.2 解：

注意：位数用补码、小数表示，阶码用移码、整数表示。

1)最大正尾数：1－16 - 6

2)最小正尾数：16-1

3)最小负尾数：－1

4)最大负尾数：－(16-1＋16 - 6)

5)最大阶码：26－1

6)最小阶码: －2 6

7)最大正数： (1－16 - 6)* 16 63

8)最小正数：16-1 * 16-64

9)最大负数：-(16-1＋16 - 6)*16-64

10)最小负数：－16 63

11)浮点零：0

12)表数精度：1/2×16-(6-1)

13)表数效率：15/16

14)能表示的规格数浮点数个数：2×15×165×2×26+1

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)设计浮点数的格式：$2^{-P}=10^{-7.2}$, $P=-\log_2 10^{-7.2}=7.2\times\log_2 10$

尾数为 24 位（隐藏最高位），阶码为 7+1 位（-128－127）。

2)计算：

①最大正数：（1－2-24）×2127

②最大负数：-2-1×2-128

③表数精度：1/2×2-23＝2-24＝10-7.22

④表数效率：50%（如果尾数采用隐藏位，那么表数效率为 100%）

2.6

(1) 0.2 = 0.333333H$\times 16^0$

设阶码为移-63 码（即-2^6+1，原题未指明）

0.2 = 0.11001100110011001100 1101B$\times 2^{-2}$

（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。

2.10 解：

要点：指令数由 256 减少到 64，减少了两位指令码。

在 A 处理机中所占的空间为：

MA = 1000*32 + （1000*2*32）/8 = 40000bit

在 B 处理机中所占的空间：39000bit

MB = 1000*30 + （1000*2*36）/8 = 39000bit

2.11 $\dfrac{20\times 8+30\times 16+20\times 32+30\times 64}{100\times 64}=\dfrac{3200}{6400}=50\%$

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|

$\dfrac{20\times 56+(100-20)\times 64}{100\times 64}=95\%$

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：$-\sum_{i}^{10} pi\log_2 pi$

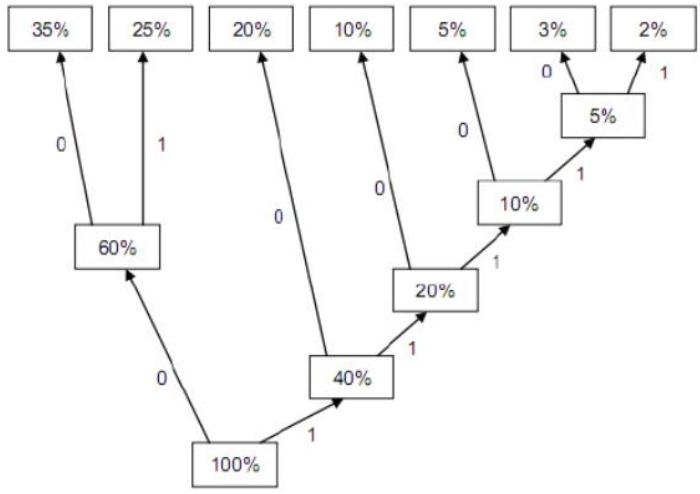
(3)2/8 扩展编码是 8/64/512 法的变种，第一组 2 条指令，码长为 2（1 位扩展标志，1 位编码）,第二组 8 条指令，码长为 4（1 位扩展标志，与第一组区别，加 3 位编码），编码性能如下表；　00；01；1***；

(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

1) 让操作的平均长度最短,用 Huffman 编码



得到的 Huffman 操作码编码是:

指令序号	使用的频度	Huffman 编码	操作码长度
1	35%	00	2
2	25%	01	2
3	20%	10	2
4	10%	110	3
5	5%	1110	4
6	3%	11110	5
7	2%	11111	5

操作码的平均长度 = 35%*2 + 25%*2 + 20%*2 + 10%*3 + 5%*4 + 3%*5 + 2%*5

= 2.35

2）8 位字长的寄存器-寄存器型指令:

已知有 8 个通用寄存器，用一个 3 位的地址码可表示一个通用数据寄存器。

指令格式: 操作码 地址码 1 地址码 2

操作码 2 位 地址码均 3 位

具体的 3 条指令格式:

00 xxx xxx

01 xxx xxx

10 xxx xxx

16 位字长的寄存器-存储器型变址寻址方式指令:

已知有 8 个通用数据寄存器，用一个 3 位的地址码可表示一个通用数据寄存器

已知有 2 个变址寄存器，用一个 1 位的地址码可表示一个变址寄存器

已知变址范围不小于正负 127，用一个 8 位的立即数可以表示一个变址寄存器内的偏移量。

指令格式：操作码 地址码 1 地址码 2 立即数

其中操作码 4 位 地址码 1 为 3 位 地址码 2 为 1 位 立即数 8 位

具体 4 条指令:

1100 xxx x xxxxxxxx

1101 xxx x xxxxxxxx

1110 xxx x xxxxxxxx

1111 xxx x xxxxxxxx

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 ... 1110 111110 （共 2^6-1）

扩充码 1110 111 { 1110 111111 000000 ... 1110 111111 111111 （共 2^6）

1111 { 1111000000 ... 1111111110 （共 2^6-1）

扩充码 1111 111 { 1111 111111 000000 ... 1111 111111 111111 （共 2^6）

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

3.1 解：

（1） 当 S2>>S1 时，平均价格接近 C2。

（2） ta = h*t1+(1-h)*t2

（3） e = 1/[h+(1-h)r]

（4） 略

（5） 当 r = 100 时，h>0.99947

（6） P134 公式，H’ = (H+n-1)/n=(0.96+5D-1)/5D=0.99947 计算得：D>15.05，取 D=16

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。

//（1）t = ht1 + (1 – h)t2,

cache 为 64k 时，t = 0.7*20ns + (1-0.7)*200ns = 74ns；

cache=128k 时，t = 38ns；

cache=256k 时，t=23.6ns

（2）按照公式：

cache=64k, c=0.2585 美元/k 字节；

cache=128k, c=0.3152 美元/k 字节；

cache=256k，c=0.4235 美元/k 字节

（3）按等效访问时间由小到大排序，容量分别为：256k，128k，64k

按每字节平均价格由小到大排序，分别为：64k, 128k, 256k

（4）①19.129 ns.美元/k 字节；

②11.9776 ns.美元/k 字节；

9.9946 ns.美元/k 字节；

256k 的 cache 最优

3.4

（1）、由 c1*s1+c2*s2≤15000 0.01*512+0.5*s2≤1500 s2≤14.6MB

（2）、由 t=t1*h+t2*(1-h) 40=20*0.95+(1-0.95)*t2 t2=420ns

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（1）

方式一、体号：4 位；体内地址：20 位；

方式二、存贮地址：20 位；多路选择器：4 位；

方式三、体内地址：20 位；存储器体号：4 位；

方式四、高位体号：1 位；低位体号：3 位；体内地址：20 位；

方式五、高位体号：2 位；低位体号：2 位；体内地址：20 位；

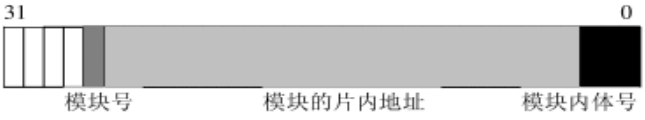
方式六、体内地址：20 位；多路选择器：2 位；低位体号：2 位；

（2）①扩大容量；②比较简单；③速度比较快；④速度快，容量大；⑤速度快，容量大；⑥提高速度

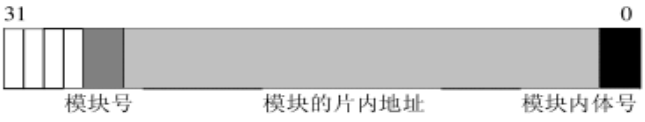
（3）① 1；② 16；③ 接近 16；④ 接近 8；⑤ 接近 4；⑥ 接近 16

方式 2：16 个模块并行访问

方式 4：2 路高位交叉 8 路低位交叉
16 个存储模块每 8 个组成一个大的模块：



方式 5：4 路高位交叉 4 路低位交叉
16 个存储模块每 4 个组成一个大的模块：



（1）这几种存储器都能够并行工作，因此可以提高频带宽度。总的来说，并行访问存储器的优点是实现简单、容易，缺点是访问冲突大；
高位交叉访问存储器的优点是扩充方便，缺点是访问效率不高；低位交叉访问存储器可以用分时的方法来提高速度，但扩充不方便。
（2）各种存储器的频带宽度和他们的工作频率有关，在不考虑冲突的情况下，如果有足够多的独立控制电路和寄存器，那么，他们的频带宽度是相同的。
（3）存储器的逻辑示意图略。
注意，并行访问存储器和低位交叉访问存储器很相象，只不过，并行访问存储器使用存储模块号（存储体号）来对已经输出的结果进行选择，而低位交叉访问存储器则用来生成对存储模块（存储体）的片选信号，他通过流水的方式来提高访问的速度。

3.8 由 P.Budnik 和 D.J.Kuck 提出的方法可知并行存储体的个数至少为 m=17;再由 ^(2*p)+1=17 得 p=2;按公式 体号=(2^p)*i+j mod m , 体内地址=i 得出无冲突的访问图如下:

3.9

(1)由公式：g= (log2Nv-log2Np)/(log2Np-log2Nd) 得：g= (log24G-log24K)/(log24k-log24)=2,故需用 2 级页表；

（2）一级页表存储容量为 1KB，共 256 个页面；二级页表为 4K 个页面，共 1M。

（3）一级页表及目前正在进行中的程序的页表驻留主存,其它可以放在辅存中。

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

ɒ=TM/T=1/1.9999=50.0025%

3.11 （1）2、5、9

（2）2098、1084、无、0060、1124、无、3116、2128、1124、无

（3）2

（4）1、3、4、6

虚地址	虚页号	实页号	页内偏移	实地址	操作	合法性
0	204	0	2	50	2098	合法
1	3060	3	1	60	1084	非法
2	6600	6	无	无	无	无
3	5860	5	0	60	0060	非法
4	3740	3	1	100	1124	非法
5	4616	4	无	无	无	无
6	1680	1	3	80	3116	非法
7	460	0	2	64	2128	合法
8	3200	3	1	100	1124	合法
9	4856	4	无	无	无	无

3.12

(1) U=$\log_2$64=6;P=$\log_2$1024=10;D=$\log_2$4K=12 用户号6位,虚页号10 位,页内偏移地址12 位

(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=27 多用户虚页号：16 位，实页号：11 位

（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+其他　另解：慢表容量：64k 个存储字（26*210），每个字长：装入位 1 位＋实页号 11 位＝12

（5）P159 图 3.27

3.13

(1)多用户虚地址:用户号-8 位;虚页号-12 位;页内偏移地址-10 位;

实地址格式：实页号-14 位；页内偏移-10 位；

问题实质：（用户号-8 位；虚页号-12 位）->（实页号：14 位）

(2) 输入位：20（8+12）；输出位：5；

(3) 相等比较电路的位数：20；

(4) 快表存储字长度：68 位，每组分为：多用户虚页号：20 位；实页号：14 位；注意：有 2 套独立的比较电路

(5) 略（P160）

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≈58.3%

(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}=(12×1024-5)/(12×1024)=12283/12288≈99.96%

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	l	3	3	2	5	1	
S(3)	4	5	3	2	5	l	1	3	2	5		
S(4)	4	4	3	2	5	5	1	3	2			
S(5)												
S(6)												

n=1 实 n=2 页 n=3 数 n=4 n=5	H H H H H H H H H 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 $N_{(1)}$

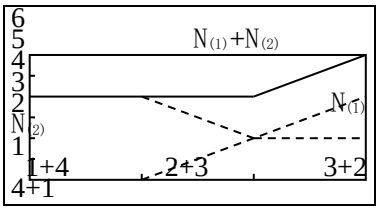
	1	2	3	4	1	3	2	1
		1	2	3	4	1	3	2
			1	2	3	4	1	3
				1	2	2	4	4
n_1= 1									0
n_1= 2									0
n_1= 3						√		√	2
n_1= 4					√	√	√	√	4

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。

3.16（1）、页地址流为：0，0，1，1，0，3，1，2，2，4，4，4；

（2）、页地址流为：0，0，2，2，1，6，3，4，4，8，9，7；

（3）、页地址流为：0，0，0，0，0，1，0，1，1，1，1，1；

（4）、页面越大，则命中的可能性比较大。

3.17

设置虚拟存储器的主要目的是扩展存储空间，设置 Cache 的主 要目的是提高访问速度。两种存储系统实现时的主要差别有：

（1）、实现方法不同，Cache 全部用硬件实现，而虚拟存储器则以软件为主，硬件为辅；

（2）、两级存储器的速度比不一样；

（3）、页块大小不一样，Cache 系统一般 1 字——16 字，虚拟存储系统则为 1KB——16KB；（4）、等效存储容量不同，Cache 等效于主存，而虚拟存储系统等效于虚存；

（5）、不命中进处理方式不同，Cache 系统采用等待方式，而虚拟系统采用任务切换。

3.18

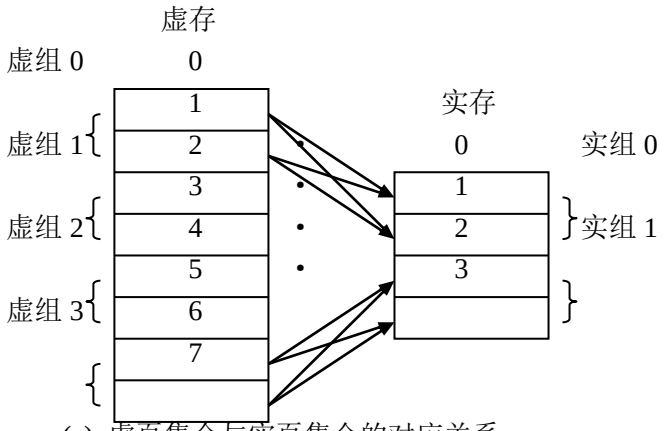
直接映象的优点在于最简单、最直接，不需要联想查找，不需要页面替换算法，因而成本较低且速度较高。缺点是命中率较低，而且它还禁止并行虚拟地址转换。全相联的优点是提供最大的灵活性，可以实现较好的块替换策略，以减少块的冲突。缺点是检索过程昂贵，需要的硬件成本比较高。组相联可以提供较好的性能价格比，替换策略可以更加经济地实现，联想检索实现容易，设计灵活可以获得更高的高速缓存的命中率。。段相联的优点在于，实现各种块替换算法比较灵活和对有限数量区段标记完成全联想检索比较经济。位选择组相联映象在块表中存放参与相联比较的只有主存地址中的区号，而没有组内块号 B，使 B 实现起来比组相联要容易.

直接映象<段相联<位选择<组相联<全相联。直接映射不需要联想查找，不需要页面替换算法；区段映射只对有限数量区段标记，所以完成全联想检索比较经济；组联想在做 k 路联想检索中，由于 k 实际上是相当小的，因此比全联想要经济得多；全联想标记长度大，检索耗时。

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) 主存与 Cache 中各个块的映象对应关系：



(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）

B6 B2 B4 B1 B4 B6 B3 B0 B4 B5 B7 B3
C2 C3 C0 C1 C0 C2 C3 C1 C0 C1 C2 C3

(5) FIFO 中 Cache 的块命中率：3/12＝25%

(6) LFU 中 Cache 的块命中率：4/12＝33.3%

(7) 改为全相联映象后：

FIFO 中块命中率： 4/12＝33.3%

LFU 中块命中率： 3/12＝25%

(8) 这时 Cache 的命中率：1－8/(16×12)＝95.8%

3.20

(1)主存 8×8＝64MB，每个存储体为 8M/16K＝512 区，每区 16K/(32×4)＝128 组。区号：9 位，组号：7 位，组内块号：2 位，偏移地址：5 位，存储体号：3 位

(2)Cache 中每块 32Byte，共 16K/32＝512 块，512/4＝128 组

组号：7 位，组内块号：2 位，块内偏移：5 位。

(3)相联目录表共有 128 行。

(4) 相联目录表如下：

19	2	1	19	2	1		19	2	1
E，B1	b	e	E，B2	b	e	.....	E，B8	b	e

其中，（E，B）表示区号和区内组号，b 表示组内块号，e 表示有效位。

(5) 比较电路的位数：19（其中 1 为为有效位标志）

(6) P181 图 3.46

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，井增大主存容量(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

4.1 P208

4.2 P250 下半部分

4.3 P222 图 4.8

4.4

原题表述有错。“1”表示被屏蔽，“0”表示开放。

(1) 正常中断屏蔽码：

响应：D1 D2 D3 D4 D5（硬件优先级）

处理：D1 D2 D3 D4 D5（屏蔽后的优先级）

(2)修改后：

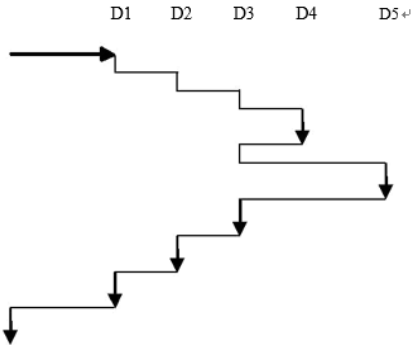
响应：D1 D2 D3 D4 D5

处理：D4 D5 D3 D2 D1

(3)处理示意图：

中断请求：D1 D2 D3 D4 D5

处理：



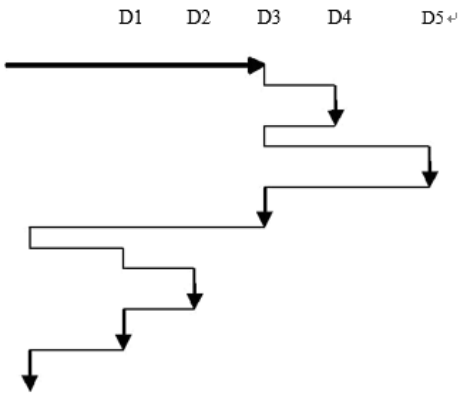
(4)处理示意图：

中断请求：D3 D4 D5 D1 D2

(4)处理示意图：

中断请求：D3 D4 D5 D1 D2

处理：



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；

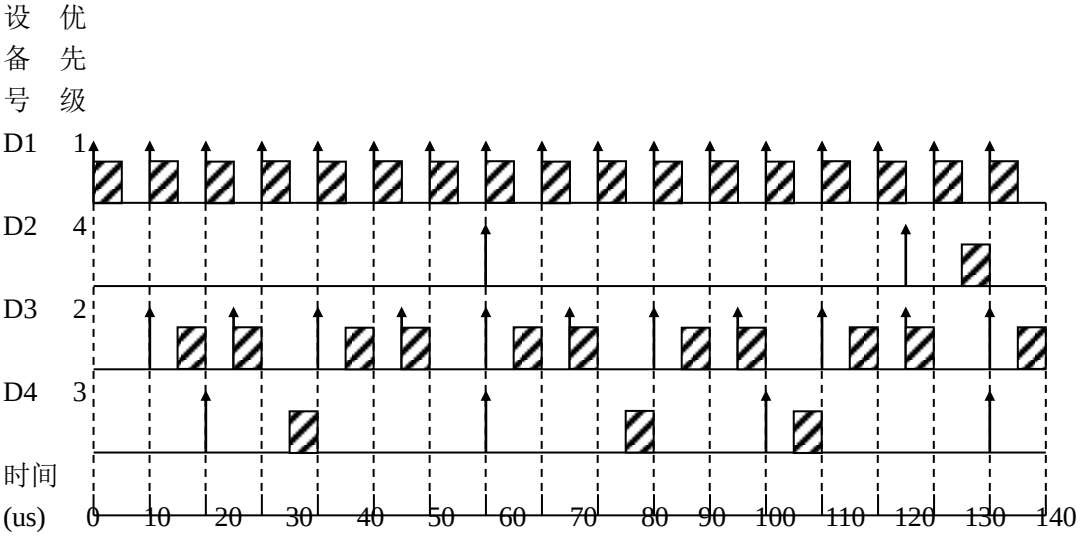
(2)中断过程示意图如右图。

	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)$f=2\times10^5$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；

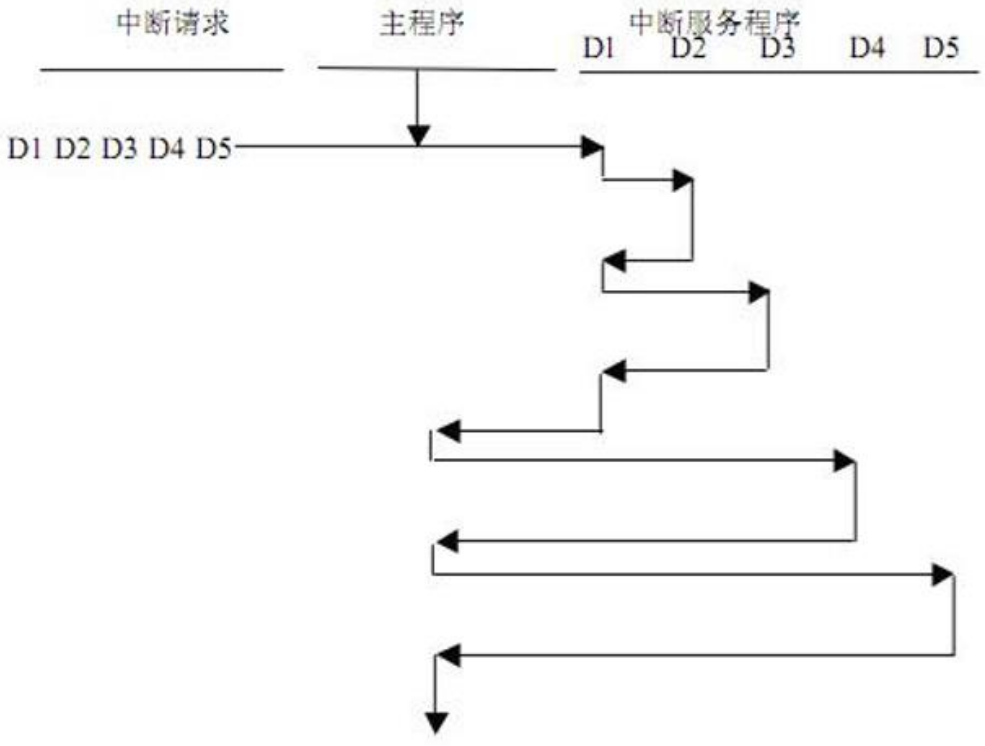
(5)参见 P245。

4.6

解：(1)有 5 个中断源，所以需要 3 位中断屏蔽码。

(2)中断响应的优先次序为：1、2、3、4、5；实际中断处理次序为：3、1、2、4、5；

(3)示意图如下。

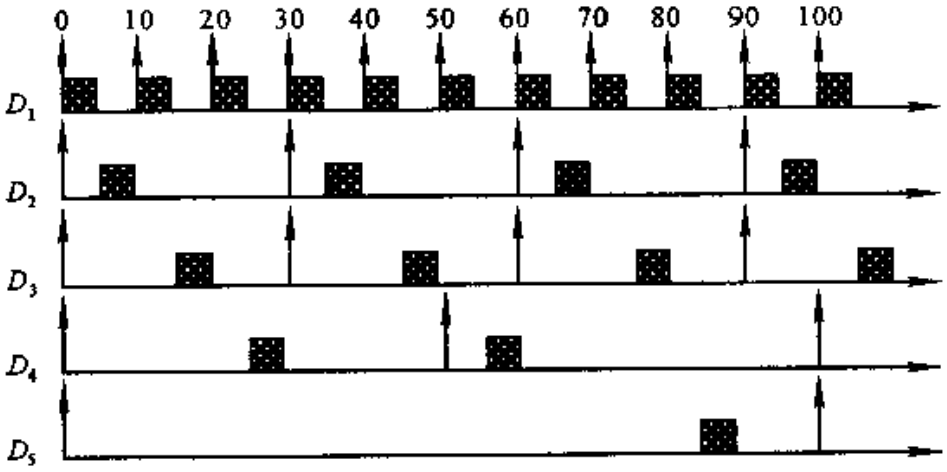


4.7

（1）196.6 KB/s。　　100+33.3+33.3+20+10

（2）200 KB/s;5 μs。

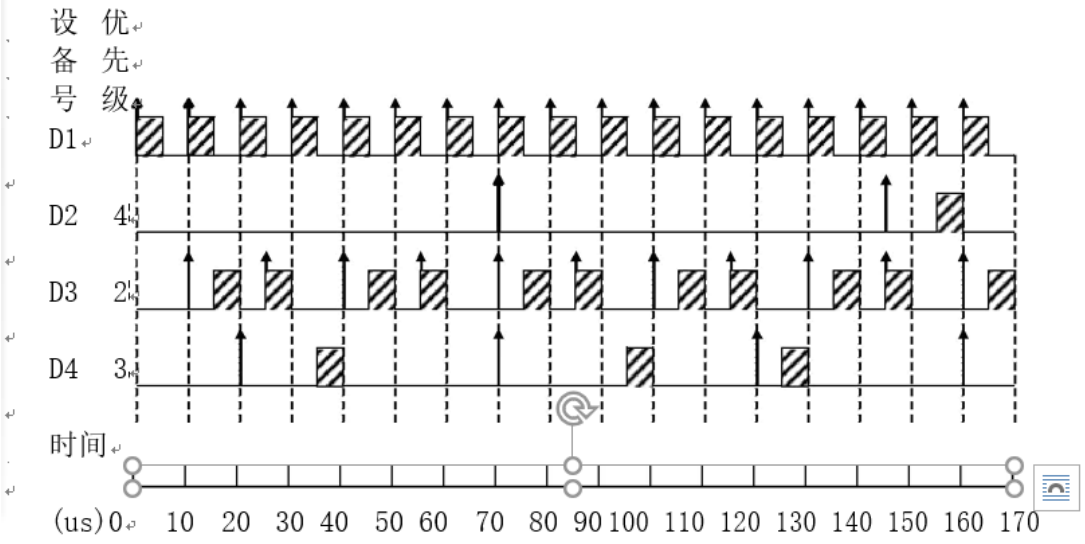
（3）通道分时工作的时间关系图如图 4－10 所示。通道处理完各设备第一次数据服务请求的时刻分别为:5 μs、10 μs、20 μs、30 μs、90 μs。



4.8

(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。

4.9

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