总复习

第1章 计算机系统结构的基本概念

1.1 计算机系统的多级层次结构

1.2 计算机系统结构、组成与实现

1.3 软件取舍与计算机系统的设计思路

1.4 软件、应用、器件对系统结构的影响

1.5 系统结构中的并行性及系统的分类

1.1多级层次结构

*1.六级层次结构

应用语言机器 → 面向用户

高级语言机器 → 面向用户

汇编语言机器 → 面向用户

操作系统机器 → 面向上层机器

传统机器 → 面向上层机器

微指令机器 → 面向上层机器

2.层次结构的实现方式

根据性价比，软硬件逻辑是等同的

3.分层优点

1.2计算机系统结构、组成与实现

*1.结构、组成与实现的概念

1)系统结构：

系统结构(System Architecture)是对计算机系统中各机器之间界面的划分和定义，以及对各级界面上、下的功能进行分配。

2)透明性概念：

在计算机中，客观存在的事物或属性从某个角度看不到，称这些事物或属性对它是透明的。计算机重的“透明”与社会生活中的“透明”，含义正好相反。

3)计算机系统结构(Computer Architecture)

是系统结构中的一部分，指层次结构中传统机器级的系统结构，其界面之上的功能包括操作系统级、汇编语言级、高级语言级和应用语言级中所有软件的功能；界面之下的功能包括所有硬件和固件的功能。因此，这个界面实际是软件与硬件或固件的分界面。

4)计算机组成(Computer Organization)

是计算机系统结构的逻辑实现，包括机器级内的数据流和控制流的组成以及逻辑实现。

5)计算机实现(Computer Implementation)

指的是计算机组成的物理实现

2.结构、组成与实现之间的关系

1)具有相同系统结构(如指令系统相同)的计算机可以因速度等因素的要求不同而采用不同的组成。

2)相同的计算机组成可以采用多种不同实现方法。

3)不同的系统结构会使组成技术产生差异

4)计算机组成也会影响系统结构，组成的设计，其上取决于系统结构，其下又受限于所可以用的实现技术。

1.3软硬件取舍与系统的设计思想

1.软件取舍的基本原则

1)在现有的硬件和器件(主要是逻辑器件和存贮器件)的条件下，系统要有高的性价比。

2)充分考虑准备采用和可能要用的的组成技术，使它尽可能不要过多或不合理地限制各种组成、实现技术的采用。

3)不能仅从“硬”的角度去考虑如何便于应用组成技术的成果和发挥器件技术的进展，还应从“软”的角度为编译和操作系统的实现，以至高级语言程序的设计提供更多、更好的硬件支持。

硬件和软件在什么意义上是等效的？在什么意义上是不等效的？

硬件和软件在功能实现上是等效的，即一种功能可以由软件实现，也可以由硬件实现。在实现性能上是不等效的。软件实现的优点是设计容易、改进简单；硬件实现的优点是速度快。

2.计算机系统的设计思路

1)由上往下

a)方法：根据用户的要求，设计基本的命令、数据类型与格式等，然后再逐级往下设计，并考虑对上一级进行优化来实现。

b)优点：适用于专用机的设计，对所面对的具体应用，其效能是很好的。

c)缺点：不适用于通用机的设计

2)由下往上

方法：根据器件条件，先把微程序机器级及传统机器级研制出来，然后再配合不同的操作系统和编译系统软件，使应用人员根据所提供的条件来采用合适的算法满足相应的应用要求。

透明性：在计算机技术中，对本来存在的事物或属性，从某一角度来看又好像不存在的概念称为透明性。

3)中间法

方法：既考虑能拿到的硬件、器件，又考虑可能所需的算法和数据结构，先进行软、硬功能的合理分配 并定义好这个界面，然后从这一中间点分别往上、往下进行软、硬设计。

- 计算机系统设计中经常使用的4个定量原理是什么？并说出它们的含义。
- 答：（1）以经常性事件为重点。在计算机系统的设计中，对经常发生的情况，赋予它优先的处理权和资源使用权，以得到更多的总体上的改进。（2）Amdahl定律。加快某部件执行速度所获得的系统性能加速比，受限于该部件在系统中所占的重要性。（3）CPU性能公式。执行一个程序所需的CPU时间 = *IC* ×*CPI* ×时钟周期时间。（4）程序的局部性原理。程序在执行时所访问地址的分布不是随机的，而是相对地簇聚。
- 程序定位：
- 把一个程序交给处理机运行，必须首先把这个程序的指令和数据装入到主存储器中。一般情况下，程序所分配到的主存物理空间与程序本身的逻辑地址空间是不同的，把指令和数 据中的逻辑地址(相对地址)转变成主存物理地址(绝对地址)的过程称为程序定位。

1.4软件、应用、器件对系统结构的影响

1.软件的可移植性

1)概念：指软件可以不加修改或经少量修改，就可以由一台机器搬到另一台机器去运行，使得同一套软件可以应用于不同的硬件环境。

2)优点：可以大量节省重复工作量，是软件设计者可以集中精力更好的改进或开发全新的软件。

2.实现可移植性的技术

1)统一高级语言

2)系列机思想

3)模拟与仿真

- 模拟：
- 模拟是指用软件的方法在一台计算机上，实现另一台计算机的指令系统，被模拟的机器 是不存在的，称为虚拟机，执行模拟程序的机器称宿主机。

1.5系统中的并行性及其分类

* 1.并行性概念

1)并行性:解题中具有可以同时进行运算或操作的特性。目的是为了能并行处理，提高解题效率。

2)广义并行性:只要在同一时刻或是在同一时间间隔内完成两种或两种以上性质相同或不同的工作，在时间上能相互重叠，都称为并行性。

3)同时性：两个或多个事情在同一时刻发生。

4)并发性：两个或多个事情在同一时间间隔内发生。

2.并行性等级

1)从执行角度分

a)指令内部

b)指令之间

c)任务或进程之间

d)作业或进程之间

2)从数据处理角度

a)位串字串

b)位并字串

c)位片串字并

d)全并行

3)从信息加工角度

a)存贮器操作并行

b)处理器操作步骤并行

c)处理器操作并行

d)指令、任务、作业并行

*3.并行性开发途径

1)时间重叠

a)方法：在并行性中引入时间因素，让多个处理过程在时间上错开，轮流重复的使用同一套硬件设备的各个部分，以加快硬件周转而提高速度。

b)优点：不必增加额外硬件设备就可以提高计算机系统的性价比。

2)资源重复

a)方法：引入空间因素，通过重复设置硬件资源来提高可靠性或性能。如双工系统。

b)优点：系统的速度性能得到很大提高。

3)资源共享

a)方法：利用软件的办法让多个用户按一定时间顺序轮流使用同一套资源，以提高其利用率，这样可以提高整个系统的性能。例如，多道程序分时系统就是利用共享CPU、主存资源，以降低系统价格，提高设备的利用率。

b)优点：节省资源，效率高，介于上述两种之间。

4.并行处理系统结构与多机系统耦合度

1)结构

a)流水线计算机

b)阵列处理机

c)多处理机

d)数据流机

2)多级系统的耦合度

a)最低耦合——只是通过某种介质(如磁盘)交换信息，各机之间没有物理连接，无共享的联机硬件资源。

b)松散耦合——也叫间接耦合，通过通道或通信线路互连，只共享某些外围设备。又分两种形式：一种形式是多台计算机通过通道和共享的外围设备相联；另一种形式是各台计算机通过通信线路连接成计算机网络。这两种都是非对称，异步的。

c)紧密耦合——也叫直接耦合，通过总线或高速开关互连，共享主存，信息传输速率高，可以实现数据集一级、任务级、作业级的并行。大多是对称型多处理系统。

5.计算机系统的分类

1)弗林分类法

根据指令流和数据流的多倍性(是指在系统性能瓶颈部件上处于同一执行阶段的指令或数据的最大可能个数)状况对计算机进行分类。

a)单指令流单数据流SISD

b)单指令流多数据流SIMD

c)多指令流单数据流MISD

d)多指令流多数据流MDMD

2)库克分类法

根据指令流和执行流(Execution Stream)及其多倍性对计算机系统结构进行分类。

a)单指令流单执行流SISE

b)单指令流多执行流SIME

c)多指令流单执行流MISE

d)多指令流多执行流MIME

3)冯泽云分类法

根据数据处理并行度对计算机系统结构分类

a)字串位串

b)字串位并

c)字并位串

d)字并位并

本章重点

本章从定性知识和定量知识两个方面介绍计算机系统结构的基本概念。有关重点如下：

(1) Amdahl定律；

(2) 平均周期数CPI公式，程序执行时间Te公式；

(3) 每秒百万指令数MIPS公式，每秒百万浮点数MFLOPS公式。

1. 定量知识———3个性能公式

1.1 Amdahl定律（加快经常性事件原理，P9）

$$S_n = \frac{T_o}{T_n} = \frac{1}{(1-F_e)+\frac{F_e}{S_e}}$$

其中：S_n —— 全局加速比；
T_o —— 原执行时间（old）；
T_n —— 新执行时间（new）；
S_e —— 被改进部分的局部加速比；
F_e —— 被改进部分原执行时间占原来总时间的百分比。

Amdahl定律的推导

改进之前程序运行总时间可写为：$T_o = T_o(1 - F_e + F_e)$，

改进之后由于其中部分操作加快，总时间降为：

$$T_n = T_o(1 - F_e + \frac{F_e}{S_e})$$

根据加速比定义，有：$S_n = \frac{T_o}{T_n} = \frac{1}{(1 - F_e) + \frac{F_e}{S_e}}$

Amdahl定律的图形

从图1.2可以看出，增大Se和Fe对Sn都有提升作用；但当Fe固定时，一味增大Se对Sn的作用会越来越不显著。

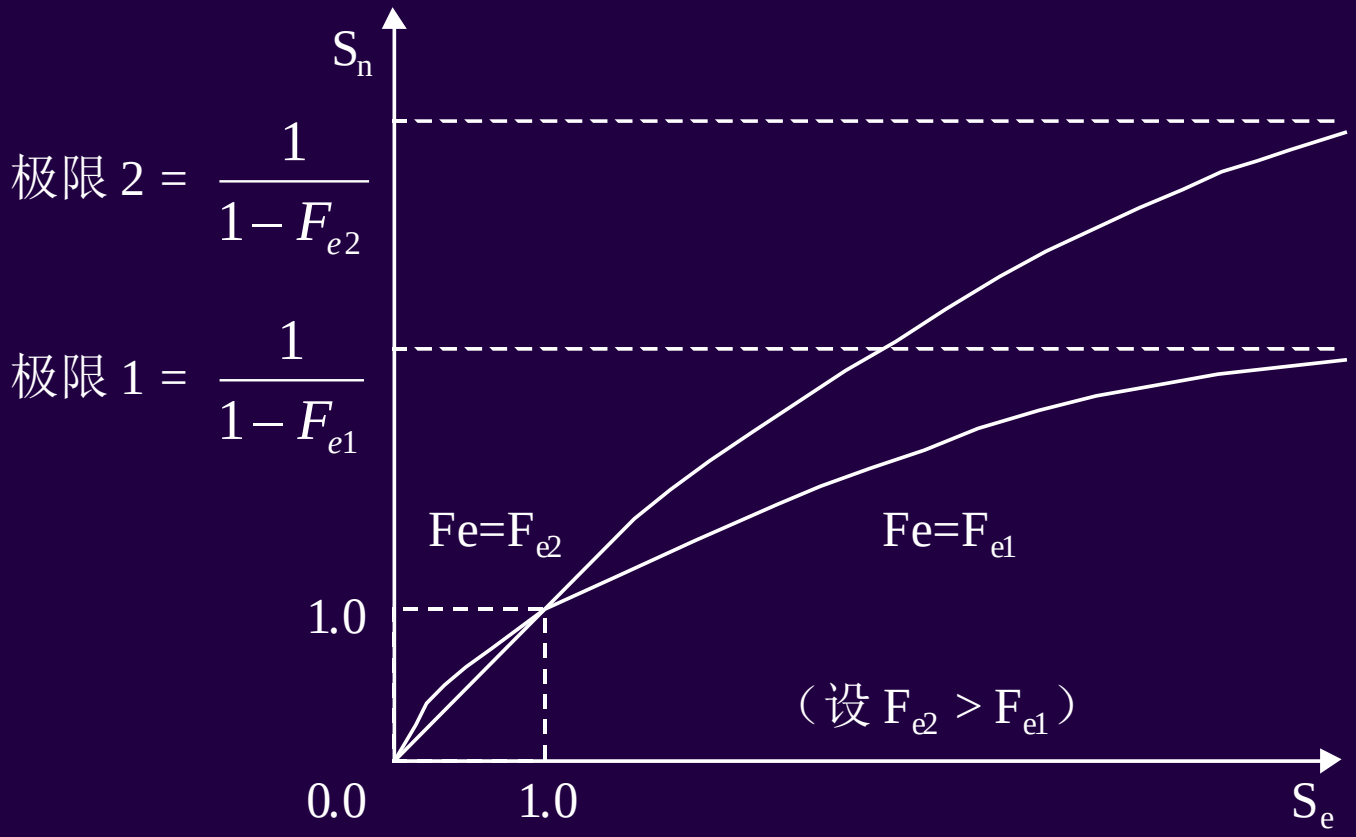


图 1.2 Amdahl 定律的图形

作1.12　假定利用增加向量模块来提高计算机的运算速度。计算机处理向量的速度比其通常的运算要快20倍，将可用向量处理部分所花费的时间占总时间的百分比称为可向量化百分比。

（1）求出加速比S和向量化百分比之间的关系式

作1.13 （2）当要得到加速比为2时的可向量化百分比F为多少？

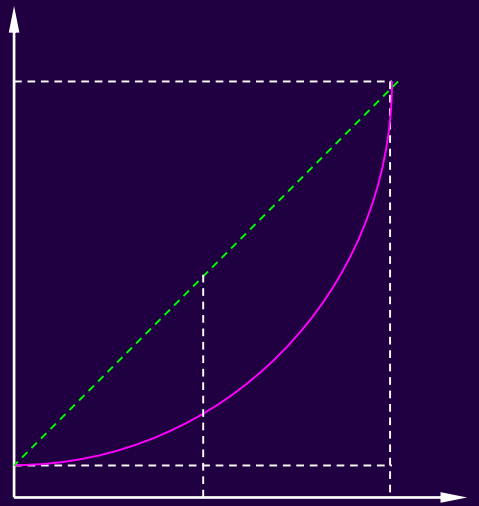
作1.14 （3）为了获得在向量模式所得到的最大加速比的一半，可向量化百分比F为多少？

解（1）：

由Amdahl定律知

$$S=\frac{1}{(1-F)+(F/20)}=\frac{20}{20-19*F}$$

(1)



(2) 由（1）式有

$$2=\frac{1}{(1-F)+(F/20)}$$

$$F=\frac{10}{19}=0.53$$

(3) 由题意可知

$$10 = \frac{1}{(1-F)+(F/20)}$$

$$F = \frac{18}{19} = 0.95$$

作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：f_e=0.9，r_e=5

$$S_P = \frac{T_e}{T_o} = \frac{1}{(1-0.9)+0.9/5} = \frac{1}{0.28} = 3.57$$

1.2 CPI与程序执行时间Te（P11）

CPI是衡量CPU执行指令效率的重要指标。让我们先考虑一个标准测速程序的全部执行时间Te和其中所有第i种指令的累计时间Ti，易知

$$T_e = IC \times CPI \times CYCLE$$

$$T_i = IC_i \times CPI_i \times CYCLE$$

其中：$CYCLE = \frac{1}{f}$，　　$IC = \sum_{i=1}^{n} IC_i$

另一方面，我们又可以写

$$T_e = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} (IC_i \times CPI_i \times CYCLE) = \left[\sum_{i=1}^{n} (IC_i \times CPI_i) \right] \times CYCLE$$

比较上面第一式与最后一式，可以得到CPI与CPI_i的关系

$$IC \times CPI = \sum_{i=1}^{n} (IC_i \times CPI_i)$$

或者写为　$CPI = \sum_{i=1}^{n} (\frac{IC_i}{IC} \times CPI_i)$，它表明*CPI*为所有$CPI_i$的加权平均值。

1.3 每秒百万指令数MIPS与每秒百万浮点数MFLOPS（P11）

$$MIPS = \frac{IC}{T_e} \times 10^{-6} = \frac{IC}{IC \times CPI \times CYCLE} \times 10^{-6} = \frac{f}{CPI} \times 10^{-6}$$，主要用于标量计算机；

$$MFLOPS = \frac{MIPS}{\text{每次浮点运算所需指令条数}}$$，主要用于向量计算机。

例题：P10，例1.1~例1.5。

P33，题12，题13，题14。

- 例1.19 用一台40MHz处理机执行标准测试程序，它含的混合指令数和相应所需的时钟周期数如下：

指令类型	指令条数	时钟周期数
整数运算	45000	1
数据传送	32000	2
浮点运算	15000	2
控制传送	8000	2

- 求有效CPI、MIPS速率和程序的执行时间。

- 解：依题意可知 $I_N=10^5$条，n=4

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

$$= \sum_{i=1}^{4}(1\times 0.45 + 2\times 0.32 + 2\times 0.15 + 2\times 0.08)$$

$$= 1.55$$

$$MIPS = \frac{f_C}{CPI\times 10^6} = \frac{40\times 10^6}{1.55\times 10^6} \approx 25.8$$

$$T_{CPU} = I_N \times CPI \times T_C$$

$$= 10^5 \times 1.55 \times 1/40\times 10^{-6} = 3.875(ms)$$

作1.20　某工作站采用时钟频率为15MHz、处理速率为10MIPS的处理机来执行一个已知混合程序。假定每次存储器存取为1周期延迟、试问：

（1）此计算机的有效CPI是多少?

（2）假定将处理机的时钟提高到30MHz，但存储器子系统速率不变。这样，每次存储器存取需要两个时钟周期。如果30%指令每条只需要一次存储存取，而另外5%每条需要两次存储存取，还假定已知混合程序的指令数不变，并与原工作站兼容，试求改进后的处理机性能。

解 (1)

$$CPI_{old} = \frac{f_{old}}{MIPS \times 10^6} = \frac{15 \times 10^6}{10 \times 10^6} = 1.5$$

(2) 依题意可知：30%的指令需要一次存储存取，则这些指令在处理器提高时钟频率之后需要增加1个时钟周期；另外5%的指令需要增加2个时钟周期。

$$CPI_{new} = CPI_{old} + 30\% \times 1 + 5\% \times 2 = 1.9$$

$$MIPS_{new} = \frac{f_{new}}{CPI_{new} \times 10^6} = \frac{30 \times 10^6}{1.9 \times 10^6} = 15.79$$

改进后性能提高情况可用CPU时间之比表示：

$$T_{CPU(old)} / T_{CPU(new)} = \frac{CPI_{old} \times I_N / f_{old}}{CPI_{new} \times I_N / f_{new}} \approx 1.58$$

作1.21　假设在一台40MHz处理机上运行200 000条指令的目标代码，程序主要由四种指令组成。根据程序跟踪实验结果，已知指令混合比和每种指令所需的指令数如下：

指令类型	CPI	指令混合百分比
算术和逻辑运算	1	60%
Cache命中的加载/存储	2	18%
转移	4	12%
Cache失效时访问主存	8	10%

(1)计算在单处理机上用上述踪数据运行程序的平均CPI
(2)根据(1)所得CPI，计算相应的MIPS 速率和程序的执行时间

- 解：依题意可知 $I_N=2\times10^5$条，n=4，

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

$$= \sum_{i=1}^{4}(1\times0.6 + 2\times0.18 + 4\times0.12 + 8\times0.1)$$

$$= 2.24$$

$$MIPS = \frac{f_C}{CPI\times10^6} = \frac{40\times10^6}{2.2\times10^6} \approx 17.86$$

$$T_{CPU} = I_N \times CPI \times T_C$$

$$= 2\times10^5 \times 2.24 \times \frac{1}{40} \times 10^{-6} = 11.2(ms)$$

第2章　数据表示与指令系统

2.1数据表示

2.2寻址方式

2.3指令系统的设计与改进

2.1数据表示

1.数据表示与数据结构

1)基本概念

a)数据表示：能由机器硬件直接识别和的引用的数据类型。

b)数据结构：各种数据元素或信息单元之间的结构关系。

2)两者关系

a)数据结构是通过软件映像将信息变换成数据表示来实现的，表示是结构的元素。

b)不同的表示为结构的实现提供不同的支持

c)结构和表示是软、硬件的交接面

2.高级数据表示

*1)自定义数据表示

a)带标志符的数据表示

b)数据描述符

目的：进一步减少标志符所占空间，对于向量、数组、记录等数据，每个元素具有相同属性，为此提出了数据描述符。

两者区别：

- 标志符：与每个数据相连，合存一个单元，描述单个数据的类型特征。
- 描述符：和数据分开，描述要访问数据是单个还是整块，及所需地址等信息。

2)向量数组数据表示

目的：为向量、数组的实现和快速运算提供更好的硬件支持，引入了向量数组数据表示，组成向量处理机。

3)堆栈数据表示

目的：通用机器对堆栈的实现支持很差，指令较少，功能单一，速度低，多用于保护子程序返回地址，少数用于参数传递。

3.引入数据表示的原则

1)看效率是否提高，即时间和空间是否减少

a)主存与CPU的通讯数据是否减少

b)是否节省了辅助性操作

2)看是否有利于通用性和利用率的提高

若只对某些数据结构的效率高，而其它很低，或引入后很少用到，必然导致性价比下降。

2.2寻址方式

1.寻址方式分析

1)不同机器的编址方式

a)把部件分类，各自从“0”开始单独编址，构成多个一维线性地址空间。

b)把所有部件统一编成一个从“0”开始的一维线性地址，对部件的访问反映为对地址的访问。

c)隐式地址，采用约定的编址方式，不必计算部件地址，加快访问速度。

寻址方式：指令系统中如何形成所要访问的数据的地址。一般来说，寻址方式可以指明指令中的操作数是一个常数、一个寄存器操作数或者是一个存储器操作数。

2)寻址方式

a)面向寄存器——速度快

b)面向堆栈——减轻编译负担

c)面向内存——针对大量数据操作

3)寻址方式的两种指明方式

a)占用操作码的某些位来指明

b)在地址码部分专门设置寻址方式位来指明

2.逻辑地址与主存物理地址

1)程序定位方法

a)静态再定位

思想:利用冯. 诺依曼型机器指令可修改的特点，程序装入内存时，用软件方法把逻辑地址变换成物理地址，执行时物理地址不再改变。

b)动态再定位

思想:基于变址思想，提出了基址寻址法，增加相应基址寄存器和地址加法器，执行时逻辑地址加基址即可形成物理地址而访问。

2)基址寻址与变址寻址

a)变址寻址:对向量、数组等数据块运算提供支持以利于实现程序的循环。

b)基址寻址:对逻辑地址空间到物理地址空间的支持，以利于实现程序的动态再定位。其实质是将静态再定位的软件实现用地址加法器硬件替换，以加快地址变换的速度。

3)界限保护

设置上下界寄存器，保护上下界地址；还可设置多对上下界寄存器，保证一道程序可以访问多个连续空间。

4)物理地址空间的信息分布

同台机器可存贮多种宽度信息:字节、字、半字及双字等。

a)存贮原则：存贮于主存中的各位信息必须是其信息位数的整数倍，即按整数边界存贮。

b)优点：减少了访问周期，有利于提高访问速度。

c)缺点：浪费存贮空间

2.3指令系统的设计和改进

1.指令格式的优化

1)基本概念

a)指令格式的优化：用最短的位数来表示指令的操作信息和地址信息，使程序中指令的平均字长最短。

b)哈夫曼压缩思想：当各种事件发生概率不等时，概率最高事件用最短位数表示，概率低事件用长位数表示，就会使平均位数缩短。可用于代码、程序、存贮空间、时间等的压缩。

*2)操作码的优化表示

a)目的:缩短指令字长度，减少程序总位数，增加指令字所能表示的操作信息和地址信息。

b)信息源熵H:

$$H=-\Sigma p_i \log_2 p_i$$

其中，p_i是指令的使用频度。

c)定长操作码的信息冗余量:

(实际平均长度－H)/实际平均长度

* d)哈夫曼编码

- 将指令的使用频度由小到大排序
- 每次选最小两个频度结合一个新节点
- 再按频度大小插入余下未结合的频度值中
- 如此重复直至全部结合完毕成根节点“1”
- 沿两个分支，分别用“0”或“1”来表示

这样，从根节点开始，沿线到达个频度指令的代码序列就是该指令的哈夫曼编码。

e)扩展操作码编码

结合哈夫曼编码与定长二进制编码思想，只用有限几种码长，仍是概率大的事件用长码，小的用短码。以此来缩短码长，降低冗余量，便于译码。

f)常用扩展方法

- 等长扩展法
- 15/15/15编码法

  适用于p_i在前15种指令中比较大，而在30种指令后急剧减小的情况。
- 8/64/512编码法

  适用于p_i在前8种指令中比较大，且之后的64种指令的p_i也不是过小时。
- 衡量标准：码长$\Sigma p_i l_i$最短

*3)指令字格式的优化

只对操作码表示优化，不对地址码表示和寻址方式采取措施，程序所需总位数难以减少。

a)主存按位编址，指令一条条挨着存贮，会使程序所需总位数减少，但是速度明显下降。为了不降低访存取指速度，就要按整数边界存贮。这样，操作码优化表示所带来l_i的缩短只是使指令字内出现空白浪费(冗余)。显然只有地址码也是可变长的，才能用上空白部分。

b)地址码长度优化

- 从访存范围考虑，地址码越长越好
- 在满足寻址范围前提下，可以缩短其位数

c)寻址方式

- 基址寻址：指明基址寄存器号
- 相对寻址：指明相对位移，相对位移不会太大。
- 地址分段：地址由段号和段内地址组成。段内转移，不需指明段号；断间转移，指明段号。
- 寄存器间接寻址：操作数存在寄存器内或经寄存器访存，则地址码宽度只需窄到指明寄存器号就可以了。

计算机指令集结构设计所涉及的内容有哪些？

(1) 指令集功能设计：主要有RISC和CISC两种技术发展方向

(2) 寻址方式的设计

(3) 操作数表示和操作数类型

(4) 寻址方式的表示：可以将寻址方式编码于操作码中，也可以将寻址方式作为一个单独的域来表示。

(5) 指令集格式的设计：有变长编码格式、固定长度编码格式和混合型编码格式三种。

2.按CISC方向发展与改进指令系统

1)改进思路

如何进一步增强原有指令的功能以及设置更为复杂的新指令取代原来由子程序完成的功能，实现软件功能的硬化。结果是系统愈加庞大、复杂，称为复杂指令系统计算机(Complex Instruction Set Computer,CISC)。

*2)CISC的改进途径(三个)

a)面向目标程序的优化实现来改进

- 目的：提高各机器语言目标程序的效率，减少存贮空间，提高运行速度，容易实现。

- 思路：

思路一：按统计出的指令和指令串的使用频度来分析改进

静态使用频度：对程序中出现的指令及指令串进行统计得到的百分比。目的是减少目标程序所占用的存贮空间。

动态使用频度：在目标程序执行中对指令和指令串统计的百分比。目的是减少目标程序的执行时间。

思路二：增设强功能的复合指令，取代原先由宏指令或子程序实现的功能，可以提高运算速度，减少程序调用的额外开销，也减少子程序所占用的空间。

- 原则：

  原则一：不删改原有指令，增设强功能指令替代对“瓶颈”有直接影响的常用指令，以满足软件向后兼容，从而新指令程序效率更高。

  原则二：尽量减少程序中如存取、传送、转移、比较等不执行数据变换的非功能型指令，增加真正执行数据变换的加、减、乘、除等功能型指令的比例。

b)面向高级语言的优化来实现

- 目的：尽可能缩短高级语言和机器语言的语义差距，以利于支持高级语言的编译系统，缩短编译程序的长度和编译所需的时间。
- 思路：

对源程序中各高级语言的使用频度进行分析

面向编译，优化代码生成来实现

改进指令系统

动态切换

发展高级语言机器

c)面向操作系统的优化来实现

- 目的：缩短OS与系统结构的语义差距，减少运行OS所需的辅助时间，节省OS软件所占用的存贮空间。
- 思路：
  - 统计和分析OS中常用指令(串)的使用频度
  - 增加专用于OS的新指令
  - 硬化或固化子程序的某些功能
  - 发展功能分布处理系统结构

3)CISC的结构和思路存在的问题

a)指令系统庞大，设计麻烦，周期长，成本高，可靠性低，查错和纠错代价大。

b)指令操作烦杂，执行速度很低。

c)难以优化编译生成真正高效的机器语言。

d)各种指令使用频度不高，且差别大，有些指令利用率很低，降低系统性价比。

3.按RISC方向发展与改进指令系统

1)RISC的改进思路

通过减少指令总数和简化指令功能来降低硬件设计的复杂度，提高指令执行速度。按这种途径和方向发展，使机器指令精练简单，因此称为精简指令系统计算机(Reduced Instruction Set Computer——RISC)。

2)RISC的设计原则

a)只选择使用频度高的指令，增加少量有效支持OS和高级语言实现的有用指令，减少条数。

b)减少系统可用的寻址方式，简化指令的格式，限于两种内，让全部指令等长。

c)让所有指令都在一个机器周期内完成。

d)增加通用寄存器以减少访存操作，所有指令只有存、取可以访存，其它都在寄存器间进行。

e)指令多数用硬联控制，少数用微程序控制。

f)以简单有效的方式支持高级语言的实现。

* 3)RISC结构采用的基本技术

a)遵循按RISC机器一般原则设计的技术。

b)在逻辑上采用硬联实现和微程序固件实现相结合的技术。

c)在CPU中设置数量较大的寄存器组，并采用重叠寄存器窗口的技术。

d)指令的执行采用流水和延迟转移技术。

e)采用认真设计和优化编译系统设计的技术。

2.1 Huffman压缩编码（P91）

(1)Huffman压缩概念（最佳编码定理）：当用n个长度不等的代码分别代表n种发生概率不等的事件时，按照短代码给高概率事件、把长代码给低概率事件的原则分配，可使平均码长达到最低。

(2) Huffman编码方法

这种编码方法由两个过程组成。

频度合并：将全部n个事件（在此即为n条指令）的频度值排序，选取其中最小的2个频度合并，然后将剩下的n-1个频度再次排序，再合并最小的2个频度，如此重复，直至剩下1个频度为止。记录所有的合并关系，形成一棵二叉树 —— Huffman树，所有原始频度值充当树叶，而最后剩下的总频度1为树根；

码元分配：从树根开始，对每个中间结点的左右2个分支边各赋予一位代码“0”和“1”（“0”在哪一侧不限）。读出从根结点到任一片树叶的路径上依次出现的代码位就排成了这个事件（即指令）的完整编码。由于频度高的事件较晚被合并，它的编码位数也就较少，符合Huffman压缩原则。

上面所说的频度值就是各事件实际出现次数的百分比，它是理论出现概率的近似值。

- **平均码长**：各事件编码长度的数学期望。

$$\overline{L} = \sum_{i=1}^{n} (P_i \cdot l_i)$$

- **信息冗余量**：它表明消息编码中“无用成分”所占的百分比。

$$R = \frac{\overline{L} - H}{\overline{L}} \times 100\%$$

2.2 扩展编码方法（等长扩展法，P94-95）

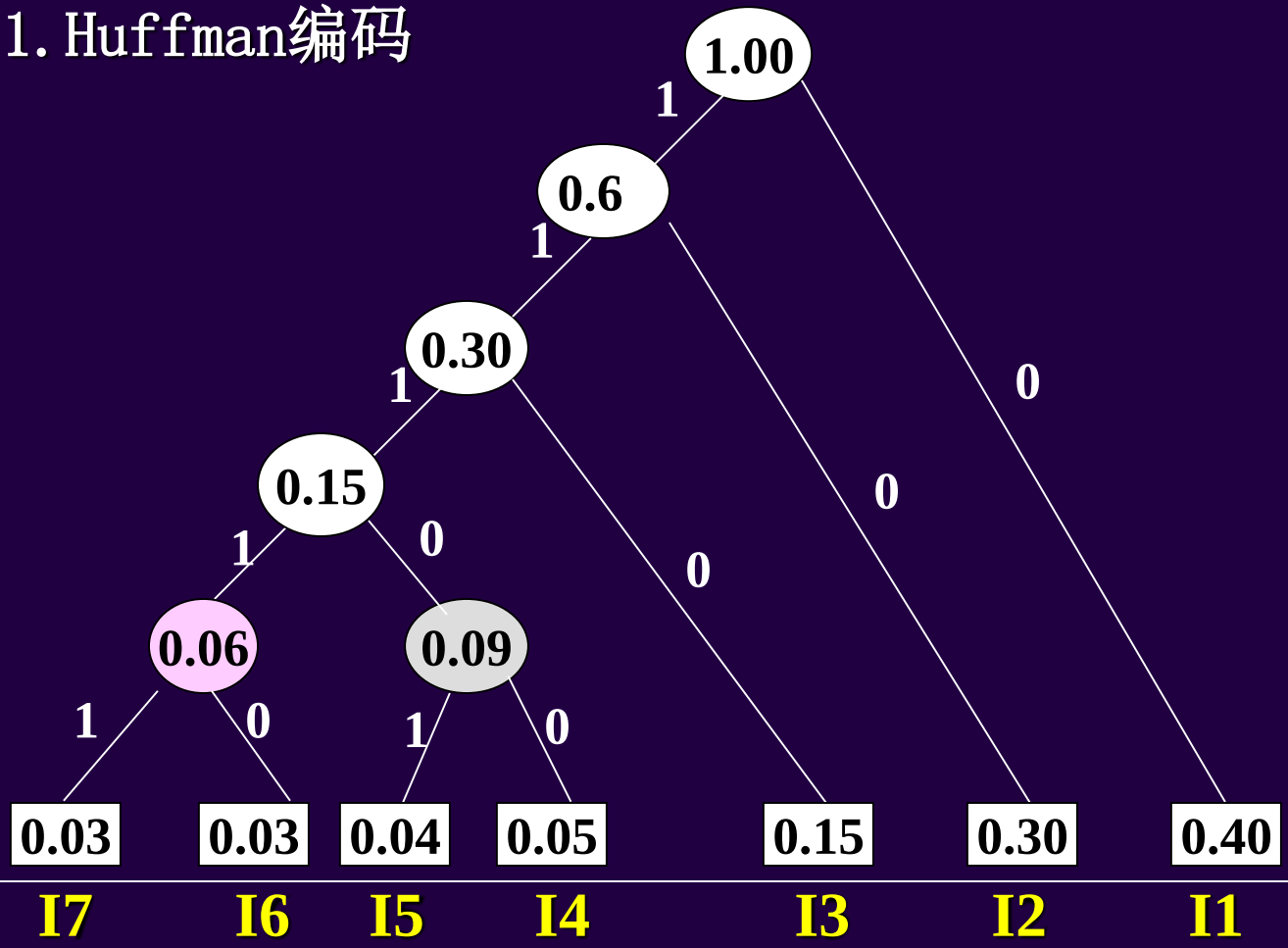
- 用码长表示：例如4-8-12法。这并不能说明具体编码方法，例如下面两种编码方法都是4-8-12法。

- 用码点数表示：例如15/15/15法，8/64/512法
  - 15/15/15法，每一种码长都有4位可编码位（前头可以有相同的扩展标识前缀），可产生16个码点（即编码组合），但是至多只能使用其中15个来表示事件，留下1个或多个码点组合作为更长代码的扩展标识前缀。已经用来表示事件的码点组合不能再作为其它更长代码的前导部分，否则接收者会混淆。这就是“非前缀原则”。
  - 8/64/512法，每一种码长按4位分段，每一段中至少要留下1位或多位作为扩展标识。各段剩下的可编码位一起编码，所产生的码点用来对应被编码事件。每一段中的标识位指出后面还有没有后续段。

例2.1　某指令系统各指令使用频度如下：

I1	I2	I3	I4	I5	I6	I7
0.4	0.3	0.15	0.05	0.04	0.03	0.03

1. Huffman编码



由此可得到哈夫曼编码如下：

I1：0　　I2：10　　I3：110　　I4：11100

I5：11101　　I6：11110　　I7：11111

平均码长

L=0.4*1+0.3*2+0.15*3+0.05*5+0.04*5

+0.03*5+0.03*5 = 2.20位

信息冗余量R=(2.20-2.17)/2.20=1.36%

指令长度个数=4

2. 扩展哈夫曼编码

- I1, I2, I3 用两位：00, 01, 10
- I4, I5, I6, I7 用四位：1100, 1101, 1110, 1111

L=(0.4+0.3+0.15)*2+(0.05+0.04+0.03+0.03)*4

= 2.30位

信息冗余量=(2.30-2.20)/2.30=0.0565=5.65%

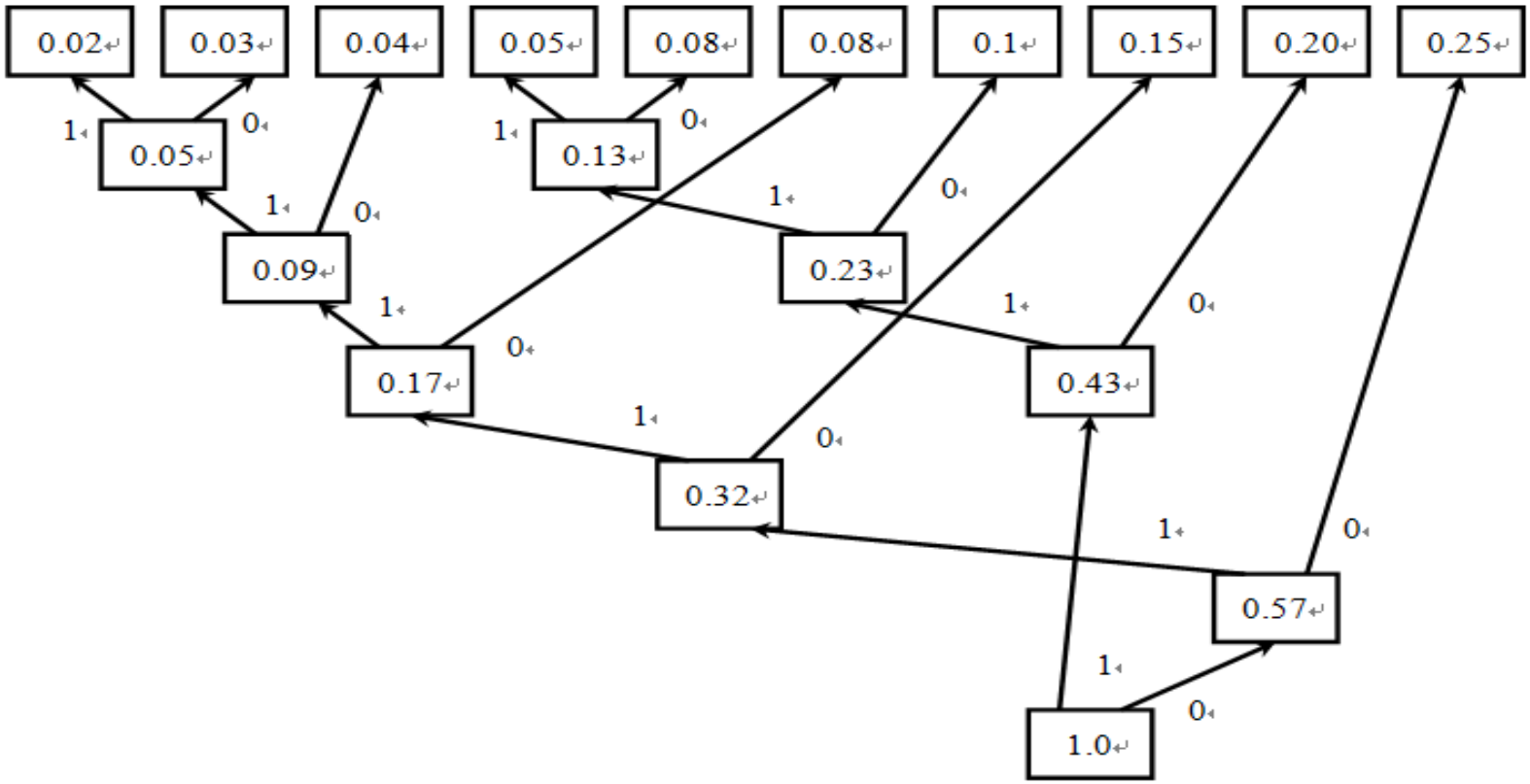
操作码的扩展（等长扩展）

指　令	频度（Pi）	操作码OP使用哈夫曼编码	OP长度li	huffman扩展编码	OP长度li
I1	0.40	0	1	0　0	2
I2	0.30	1　0	2	0　1	2
I3	0.15	1　1　0	3	1　0	2
I4	0.05	1　1　1　0　0	5	1　1　0 0	4
I5	0.04	1　1　1　0　1	5	1　1　0 1	4
I6	0.03	1　1　1　1　0	5	1　1　1 0	4
I7	0.03	1　1　1　1　1	5	1　1　1 1	4

平均码长：　　2. 2　　2.3

作 2.13 采用最优Huffman编码法（信息熵）的操作码最短平均长度为：

$$H = -\sum_{i=1}^{n} p_i \times \log_2^{p_i} \approx 2.957$$

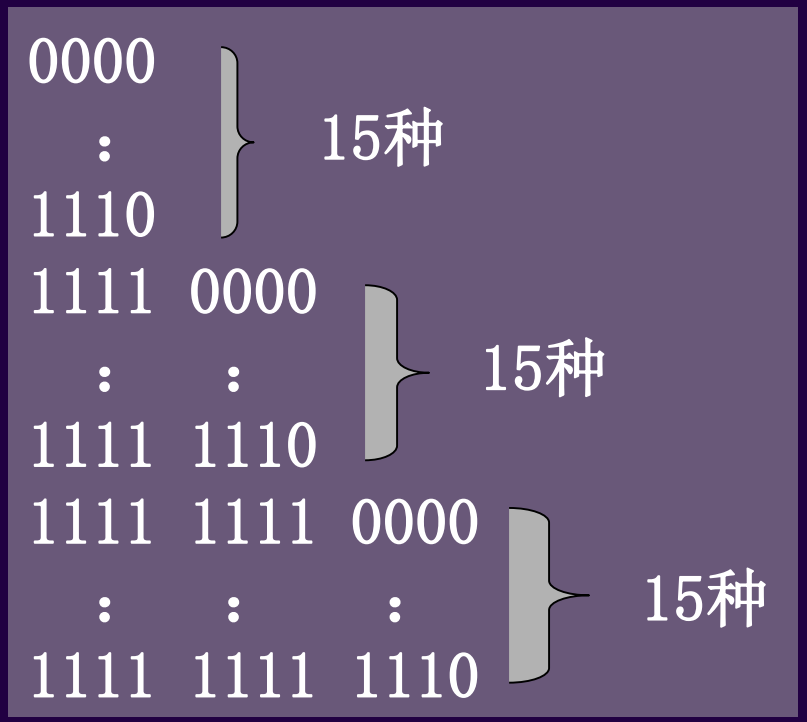


指令序号	出现的概率	Huffman 编码法	2/8 扩展编码法	3/7 扩展编码法
I_1	0.25	00	00	00
I_2	0.20	10	01	01
I_3	0.15	010	1000	10
I_4	0.10	110	1001	11000
I_5	0.08	0110	1010	11001
I_6	0.08	1110	1011	11010
I_7	0.05	1111	1100	11011
I_8	0.04	01110	1101	11100
I_9	0.03	011110	1110	11101
I_{10}	0.02	011111	1111	11110
操作码的平均长度		2.99	3.1	3.2
操作码的信息冗余量		1.1%	4.6%	7.6%

例2.2 指令系统共有42种指令，前15种使用频率平均为0.05，中间13种使用频率平均为0.015，最后14种使用频率平均为0.004。如何编码？

解：因频率分布有三种，故码长可有三种；

因每段指令数基本相同，故可采用等长扩展(4-8-12位)， 保留特征码的每段指令数相同(15-15-15)方法。结果如图所示；



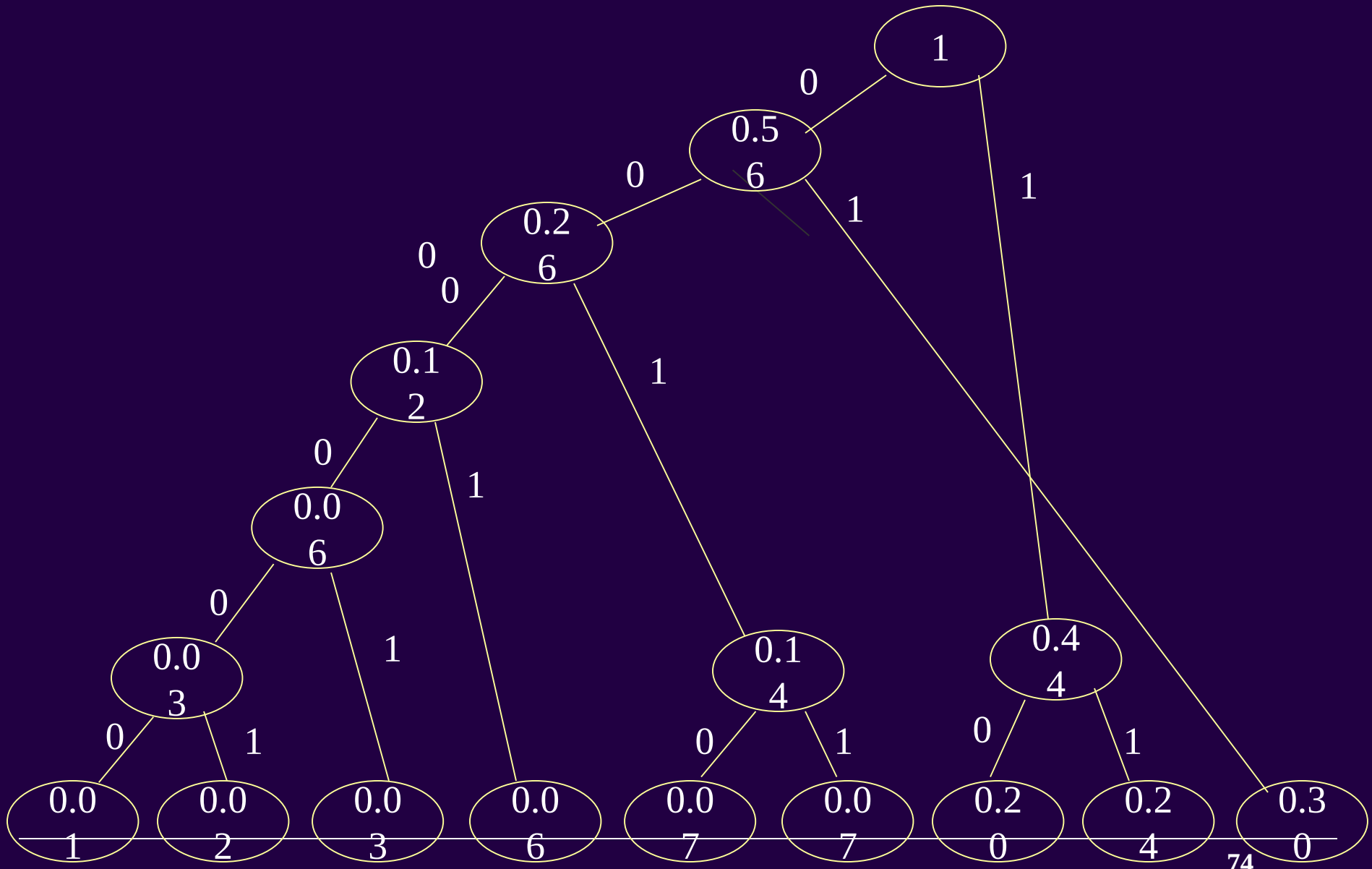
结果：采用15-15-15扩展方法，最后一种编码用于扩展，每段0000～1110用于编码，1111用于扩展。

例2.3 某模型机有9条指令，其使用频率为：

ADD（加）	30%	SUB（减）	24%
JOM（按负转移）	6%	STO（存）	7%
JMP（转移）	7%	SHR（右移）	2%
CIL（循环左移）	3%	CLA（清加）	20%
STP（停机）	1%		

要求有两种指令字长，都按双操作数指令格式编，采用扩展操作码，并限制只能有两种操作码码长。设该机有若干个通用寄存器，主存为16位宽，按字节编址，采用整数边界存贮，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。

解：(1) Huffman树的形式如图所示。



- 由上图可得到的Huffman编码为：

ADD(加)	30%	01
SUB(减)	24%	11
CLA(清加)	20%	10
JOM(按负转移)	6%	0001
STO(存)	7%	0011
JMP(转移)	7%	0010
CIL(循环左移)	3%	00001
SHR(右移)	2%	000001
STP(停机)	1%	000000

因此，操作码的平均码长为：

$$l = \sum_{i=1}^{9} p_i \times l_i = 2.61 \text{ 位}$$

- (2) 采用2-5扩展的操作码编码为：

ADD(加)	30%	00
SUB(减)	24%	01
CLA(清加)	20%	10
JOM(按负转移)	6%	11000
STO(存)	7%	11001
JMP(转移)	7%	11010
SHR(右移)	2%	11011
CIL(循环左移)	3%	11100
STP(停机)	1%	11101

因此，操作码的平均码长为：

$$\sum_{i=1}^{9} p_i \times l_i = (0.30+0.24+0.20)\times 2 + 0.26\times 5 = 2.78\text{位}$$

(3) 该机允许使用的可编址的通用寄存器个数为 $2^3=8$个

(4) 短指令为寄存器-寄存器型，格式如下：

OP(2位)	R1(3位)	R2(3位)

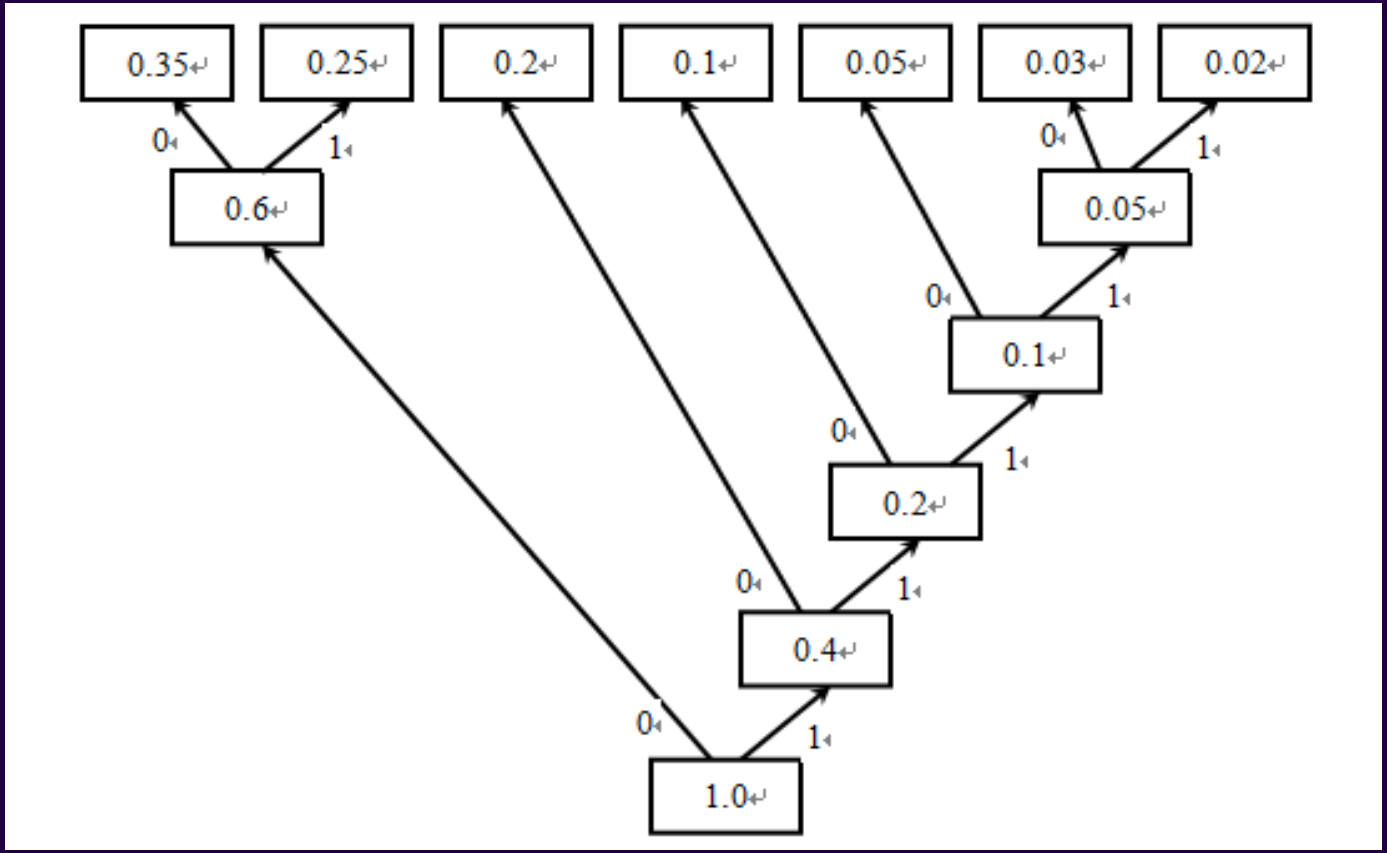
长指令为寄存器-主存型，格式如下：

OP(5位)	R1(3位)	X(2位)	相对位移d(6位)

(5) 访主存操作数地址寻址的最大相对位移量为64个字节(-32~+31个字节)

- 作2.14 一台模型机共有7条指令，各指令的使用频度分别是35％、25％、20％、10％、5％、3％、2％，有8个通用数据寄存器，2个变址寄存器。
- (1) 要求操作码的平均长度最短，请设计操作码的编码，并计算所设计操作码的平均长度。
- (2) 设计8位字长的寄存器－寄存器型指令3条，16位字长的寄存器－存储器型变址寻址方式指令4条，变址范围不小于正、负127。请设计指令格式，并给出各字段的长度和操作码的编码。

答：(1) 要使得到的操作码长度最短，应采用Huffman编码，Huffman树构造如下



由此可以得到7条指令的编码分别如下：

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，Huffman编码法得到的操作码的平均长度为：

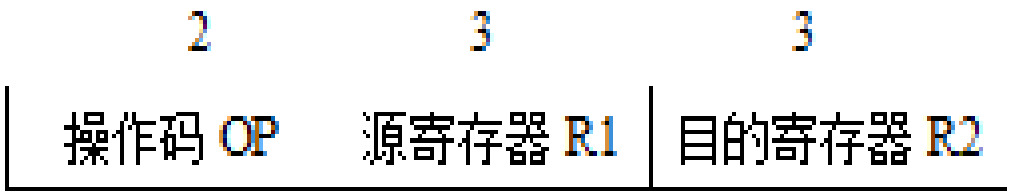
$l = 2\times(0.35+0.25+0.20) + 3\times0.10 + 4\times0.05 + 5\times(0.03 + 0.02)$

$= 1.6+0.3+0.2+0.25 = 2.35$

(2)

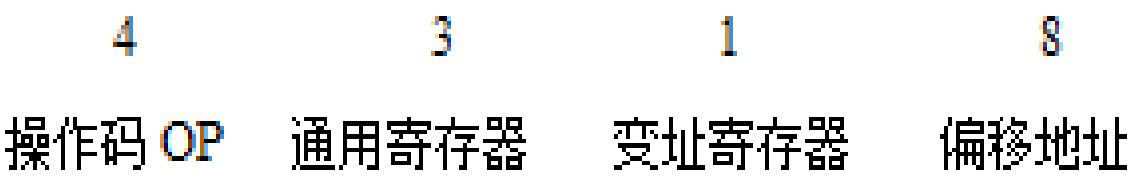
设计 8 位字长的寄存器-寄存器型指令如下：

因为只有 8 个通用寄存器，所以寄存器地址需 3 位，操作码只有两位，设计格式如下：



三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：



四条指令的操作码分别为 1100、1101、1110、1111。

作2.14 (改） 一台模拟机共有7条指令，各指令的使用频度分别为35%，25%，20%，10%，5%，3%，2%。该模拟机有8位和16位两种指令长，采用2-4扩展操作码，8位字长指令为寄存器-寄存器（R-R）二地址类型，16位字长指令为寄存器-存贮器（R-M）二地址变址寻址（-128≤变址范围≤127 ）类型。

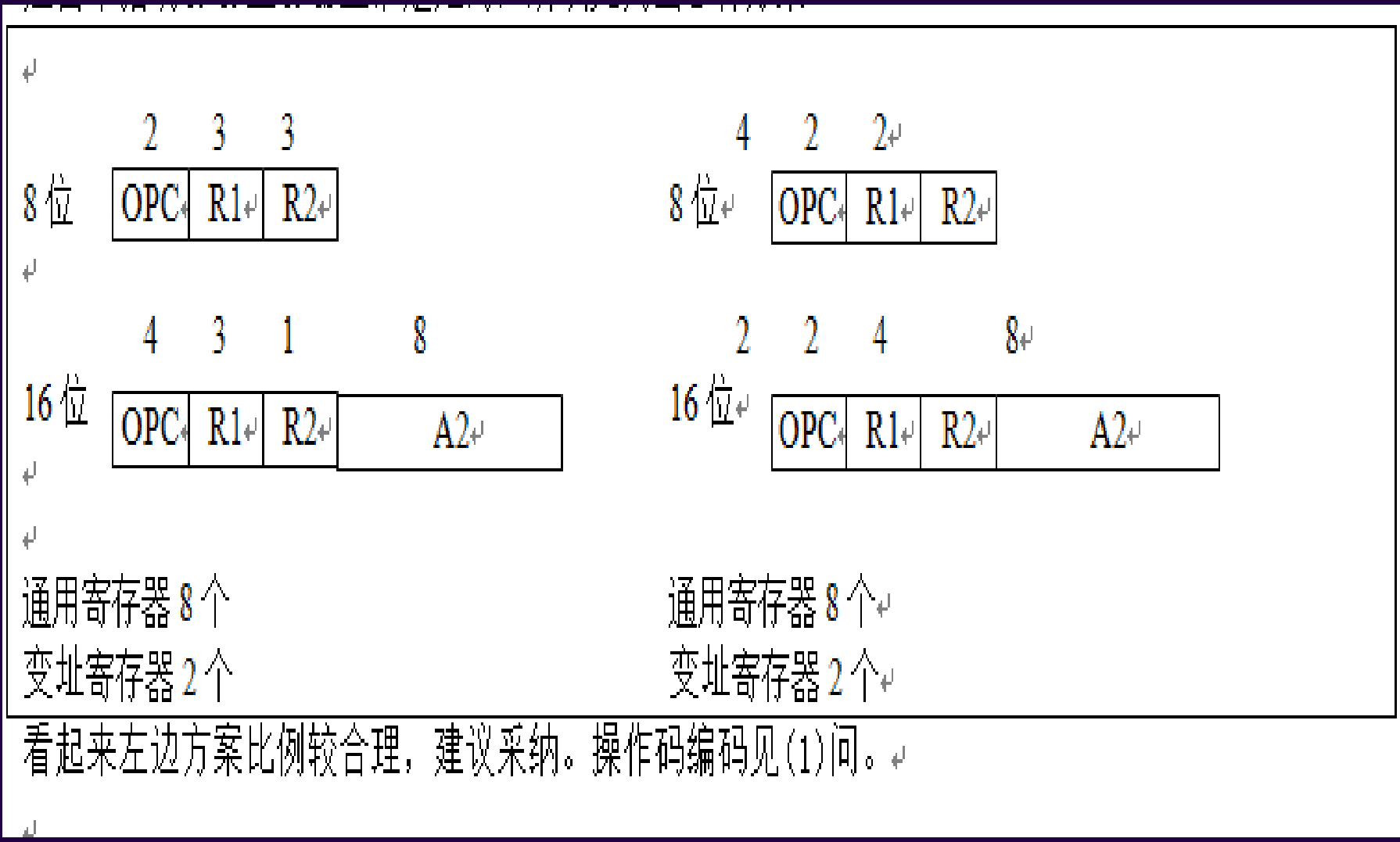
- (1) 计算操作码的平均码长
- (2) 该机允许使用多少个可编址的通用寄存器，多少个变址寄存器？
- 设计该机的两种指令格式，标出各字段位数并给出操作码编码。

- 扩展码有2种方案（下图）

00	00
01 3条	01 2条
10	1000
1100	1001
1101 4条	1010 5条
1110	1011
1111	1100
(3-7型 ）	（2-8 型）简版

- 2-4扩展码有2种方案（上图）
- 前一种方案里短指令占的比例较大，应该选用它。
- 平均码长/L = (0.35 + 0.25 + 0.20)×2 + (0.1 + 0.05 + 0.03 + 0.02) ×4 = 2.4

  用R代表寄存器编号，A代表变址偏移量。
- 题目未指明R-R型R-M型谁是短码，所以先列出2种方案：



看起来左边方案比例较合理，建议采纳。操作码编码见(1)问。

8 位: 操作码 2, 寄存器（8- 2)/2= 3 （右边为 操作码4 位）

16位：操作码 2, 寄存器 3，变址寻址（-128≤变址范围≤127 ）：2^7=128，符号1位，共 8位，变址器： 16-4-3-8=1

2.2.2 操作数优化———寻址方式比较（P95）

指令中操作数占用的位数由操作数的个数与寻址方式决定。

按操作数的个数划分，有零操作数指令、一操作数指令、二操作数指令、三操作数指令共四种形式。应该按机器用途来选择（P99，表2.20）。

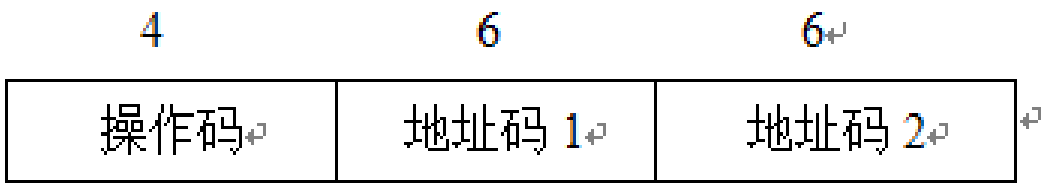
缩短操作数长度的常用方法是间址和变址（P99页末）。

作2.15 某处理机的指令字长为16位，有双地址指令、单地址指令和零地址指令三类，并假设每个地址字段的长度为16位。

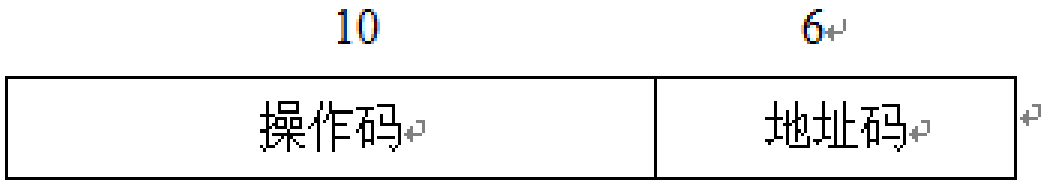
（1）如果双地址指令有15条，单地址和零地址指令的条数基本相同，问单地址指令和零地址指令各有多少条？并且为这三类指令分配操作码。

（2）如果三类指令的比例为1：9：9，问双地址指令、单地址指令和零地址指令各有多少条？并且为这三类指令分配操作码。

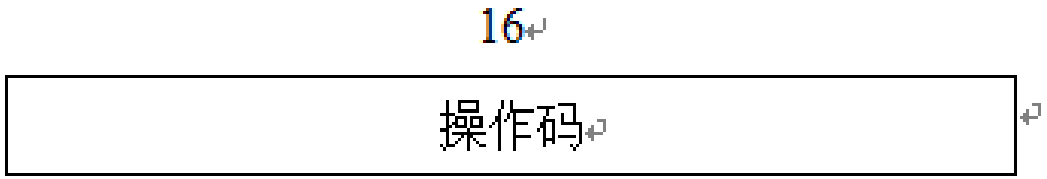
答：(1) 双地址指令格式为：



单地址指令格式为：



零地址指令格式为：



解：

（1）双地址指令15条，　地址码：　0000～1110

单地址指令$2^6-1=63$条，地址码：1111 000000
1111 111110

零地址指令64条，地址码：　1111 111111 000000
1111 111111 111111

(2)

假设双地址指令 x 条，则单地址、零地址分别为 9x 条：

$$[(2^4 - x) * 2^6 - 9x] * 2^6 = 9x$$

解之即得：x =14

∴ 双地址指令 14 条，操作码为：0000~1101；留出两个编码用于扩展。

单地址指令$(2^6\text{-}1)\times 2 = 126$ 条，操作码为：

1110 000000~1110 111110，1111 000000~1111 111110

零地址指令 126 条，操作码为：

1110 111111 000000~1110 111111 111110，1111 111111 000000~1111 111111 111110

第3章 存贮体系

3.1存贮体系的形成与性能

3.2虚拟存贮

3.3高速缓冲存贮器Cache

3.4主存保护

3.1 存贮体系的形成与性能

1.存贮器的性能要求

1)大容量

$S_M=W \cdot l \cdot m$

W：存贮体的字长，单位为bit或Byte。

l：每个存贮体的字数。

m：并行工作的存贮体的个数。

2)低价格

可以用总价格C或每位价格c来表示。具有S_M位的存贮器每位价格$c=C/S_M$。其中包括了存贮器本身

的价格和为该存贮器操作必须的外围电路的价格。

3)高速度

a)访问时间T_A

T_A是存贮器接到访存到信息被读到数据总线上所需的时间。是确定CPU与存贮器时间关系的重要指标。

b)存贮周期T_M

T_M是连续启动一个存贮体所需要的时间间隔。一般来说总比T_A大。

c)存贮器频宽

是指存贮器可以提供的数据传送率，一般用每秒钟所传送的信息位数来衡量。

最大频宽B_M(极限频宽)：是存贮器连续访问时能提供的频宽。

单体： $B_M = W/T_M$

m体并行工作：$B_M = mW/T_M$

实际频宽：实际频宽小于最大频宽B_M

4)结论

由于存贮器的价格、速度和容量的要求是相互矛盾的，为了同时满足三方面的要求，在一个完整的存贮体系中，必须采用不同工艺的存贮器，使得信息以各种方式分布于不同的存贮体。

2.并行主存系统频宽的分析

1)类型

a)单体单字

b)单体多字

c)多体单字交叉

d)多体多字交叉

2)分析结论

由于程序的转移概率不会很低，数据分布的离散性较大，所以单纯靠增大m来提高并行主存系统的频宽是有限的，且性价比还会随m的增大而下降。如果采用并行主存系统仍不能满足速度上的要求，就必须从系统结构上改进，采用存贮体系。

3.存贮体系的形成与分支

1)容量需求

主存——辅存存贮层次　程序局部性

2)速度需求

Cache——主存存贮层次　　程序局部性

3)多级存贮层次

4.存贮体系的性能参数

1)存贮体系的每位平均价格c

2)命中率$H=R_1/(R_1+R_2)$

3)等效访问时间

$T_A=HT_{A1}+(1-H)T_{A2}$

简述“Cache—主存”层次与“主存—辅存”层次的区别。

存储层次 比较项目	“Cache—主存”层次	“主存—辅存”层次
目的	为了弥补主存速度的不足	为了弥补主存容量的不足
存储管理的实现	全部由专用硬件实现	主要由软件实现
访问速度的比值 （第一级比第二级）	几比一	几万比一
典型的块（页）大小	几十个字节	几百到几千个字节
CPU对第二级的访问方式	可直接访问	均通过第一级
不命中时CPU是否切换	不切换	切换到其它进程

- **例3.1** 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：

$$S_p = \frac{T_{A2}}{T_A} = \frac{5T_{A1}}{HT_{A1} + (1-H)T_{A2}} = \frac{5}{0.9 + 0.1 \times 5} = 3.57$$

- **例3.2** 假设高速缓存Cache的访问周期为50ns，主存的访问周期为400ns，且Cache被访问命中的概率为95%，则采用Cache后，能使整个存储系统等效的访问周期为多少？获得多高的加速比？

解：

$$T_A = HT_{A1} + (1-H)T_{A2} = 0.95 \times 50 + 0.05 \times 400 = 67.5ns$$

$$S_p = \frac{T_{A2}}{T_A} = \frac{400}{67.5} = 5.9$$

存储层次的管理方式(P148)

根据程序的局部化性质，存储层次机构对用户文件的管理应该划分成较小的基本调度单位来进行。依划分标准不同，存在3种存储层次管理方式。

(1)**段式管理**(P148)。段是程序中的一个逻辑单位，可以是一个程序模块，或者是一个数据结构。段的长度不一，但段内所有数据的信息属性一般是相同的，便于统一进行信息保护。

每段使用独立的逻辑地址空间，即都从0开始计算地址。

段式管理方法的主要缺点是各段长短不一，调进调出之后容易形成大量不规则的零碎空间。

段式管理方法的虚实变换算法是查段表(P150)。

(2)**页式管理**(P151)。页是系统规定的固定长度单位。按页划分用户文件可以避免上述零碎空间浪费。

我们把用户文件划分得到的一个长度单位称为“**虚页**”，因为它的页号是在虚地址空间中编排的；实地址空间按页的大小划分得到的一个长度单位称为“**实页**”。

页式管理方法的主要缺点是按固定长度分出来的同一页内常有不同属性的信息，不便于信息保护的实现。

页式管理方法的虚实变换算法是查页表(P152)。

(3)**段页式管理**(P153)。它把上述两种管理方式结合起来，首先将整个文件分段，然后在各段内分页，所以有一个段表和若干个页表。其虚实变换算法是先查段表，查出该段的页表起始地址再查相应的页表(P154)。

段页式管理的主要缺点是多查一次表，虚实变换费时较多，占用空间也较大。

由于段页式管理方法的最小调度单位仍是页，或者说它是分段之后的分页管理，为了叙述简单，下面的分析还是以页式管理为模型。

3.3 地址映象与变换(P174)

基本术语：

逻辑地址（又称为**相对地址**、**虚地址**）是程序员在编写和编译一个程序模块时分配指令和数据的空间单位序号，总是从0开始（可以按字节编址、按CPU字编址等）。逻辑地址的取值范围称为**逻辑地址空间**、**虚空间**或**虚存**。

物理地址（又称为**绝对地址**、**实地址**）是任一级存储器为全部存储单元分配的序号。物理地址的取值范围称为**物理地址空间**、**实空间**或**实存**。

从M_1到M_n各层都有自己的物理地址空间，而对当前执行的程序模块来说，逻辑地址空间只有一个。

地址映象方式指的是虚页集合与实页集合的对应规则，或者说是约束关系。

地址变换（又叫**虚实变换**）指逻辑地址到物理地址的变换过程或者算法。

页失效指当前被访问存储级中没有所需的信息，也就是不命中现象。

实页争用又叫**实页冲突**，指虚页调入时，根据地址映象方式划定的实空间范围内已没有空闲实页的状况。

相联目录表技术

1. 页表占用空间过大问题

页表必须存放在实存M1里。实际上，命中情况下的访存时间等于查表时间加上访问目标数据的时间，所以页表不能放在M2。

页表占用空间 = 页表行数 × 每行宽度

其中，页表行数 = 虚存容量 / 页面大小

以PC机为例，页表行数 ≥ 60G / 4K = 2^{36} / 2^{12} = 2^{24} ≈ 1600万！按每行宽度6字节估算约需96MB。

减少页表空间的思路分减少行数和减少行宽两类。

2. 相联目录表方法（P158）

仅保留页表中已装入的虚页记录。为避免逐行比对，利用相联存储器存放此表，它具有并行比较功能，但价格远高于普通存储器。

3. 快慢表方法（P159）

4. 通过地址映象减少行宽

如下文所示

4种常见的地址映象方式

3.3.1 全相联(P174)

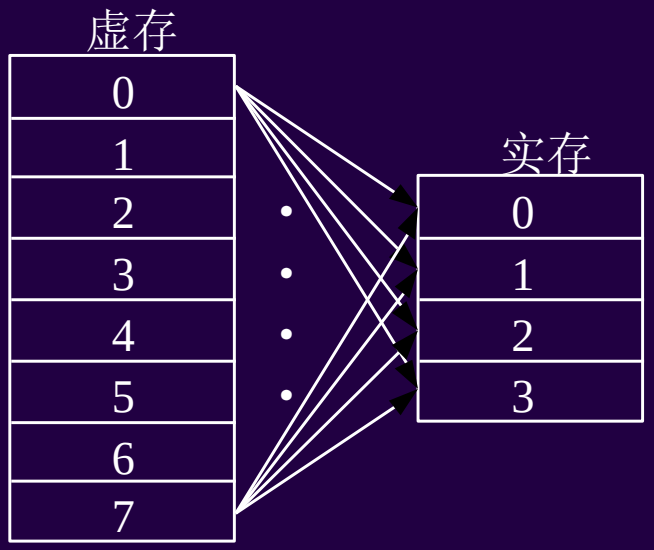
全相联就是无约束对应，或者说是一个完全关系，意思就是一个虚页可以调入任何一个实页。这种关系可用下页示意图(a)、(b)表示。

全相联的虚实变换信息完全来自于变换表，查表过程如图(c)所示。

全相联映象方式使虚页调入有最大的选择范围，发生实页争用的可能性最小，调入/调出的操作开销也最少，有利于命中率提高。但这种方式的页表占用空间和查表时间开销都比较大,也就是说实现成本比较高，在命中情况下花费在虚实变换上的时间也比较多。

由于页表必须常驻在实存中，而主存-辅存层次的实存（即主存）相对Cache-主存层次的实存（即Cache存储器）要低廉一些，所以全相联映象方式一般用于主存-辅存层次。

全相联的地址映象方式与地址变换原理示意图(a)(b)

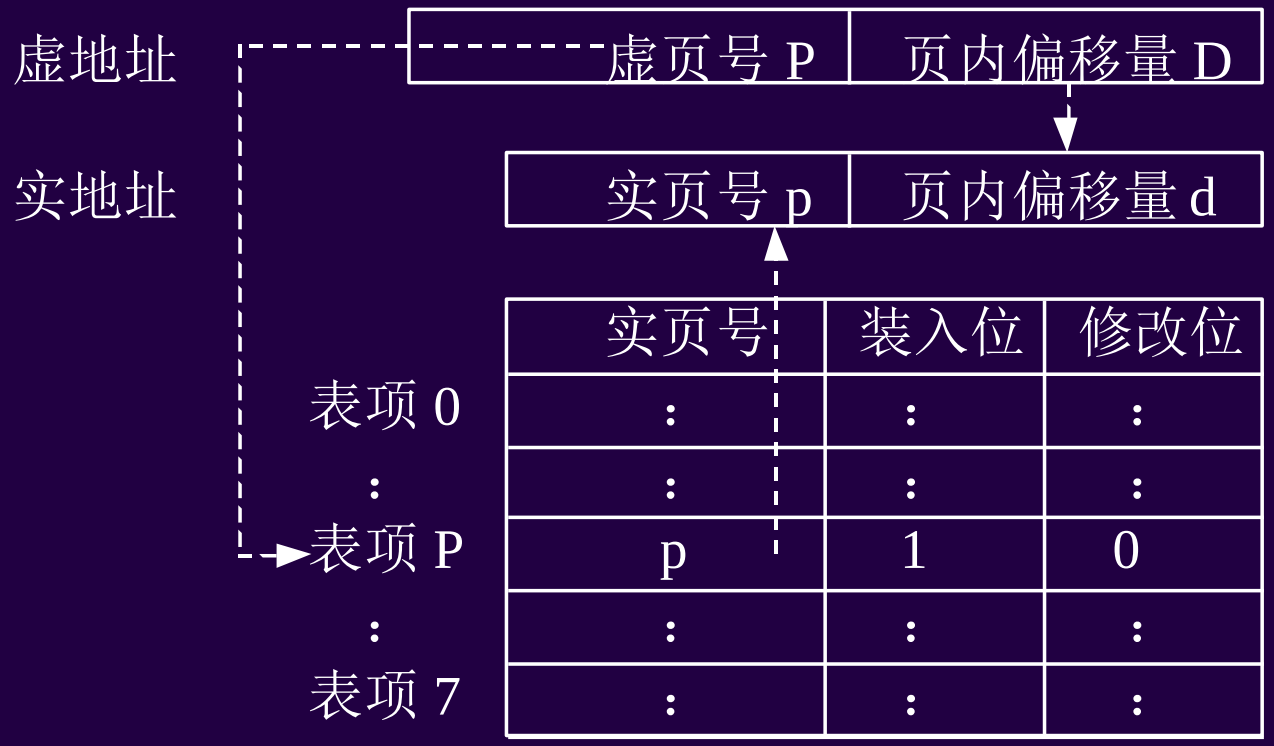


(a) 虚页集合与实页集合的对应关系

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

(b) 对应关系表（√为有关系）

全相联的地址映象方式与地址变换原理示意图(c)



(c) 通过查表进行虚实变换

3.3.2 直接相联(P176)

直接相联是一种最强的约束关系，它规定每个虚页只对应唯一的实页。为了便于虚实变换，用求模运算作为变换关系式：将虚页号对实页总数求模得到实页号。实现起来非常简单，因为在二进制中，任何数X对2的整次幂n求模等价于截取X的最低$\log_2 n$位，如下页示意图(c)所示。

- 例3.3 已知虚页号 = 7，实页总数 = 4，用直接相联求实页号。

解：

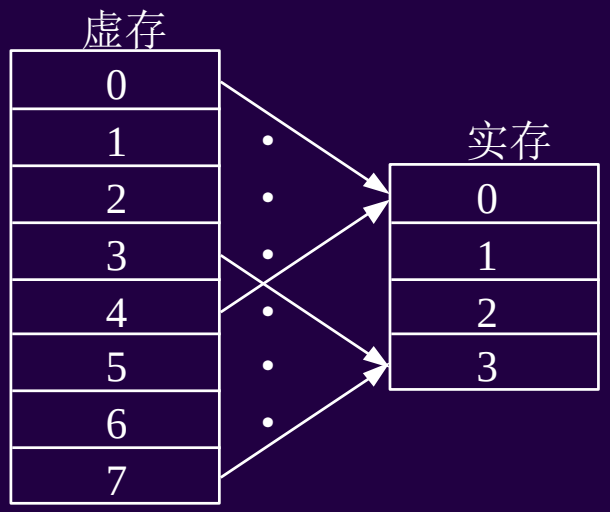
可用十进制形式求：7 mod 4 = 3；

也可用二进制形式求：由于n = 4，所以$\log_2 n$ = 2，取7的二进制形式111B的最低2位，得11B，即3。

直接相联映象方式不需要借助页表来进行虚实变换，显然大大节省了相应的空间与时间（当然页表中的装入位和修改位还得保留），但是由于每个虚页的选择范围太小，实页争用的发生频率较高，常出现明明实存有空闲空间却不得不调出一个现有虚页以腾出所在实页的情况，这使系统的命中率和运行效率大大下降。

这种映象方式主要用于一些对实存价格非常敏感的Cache-主存层次。

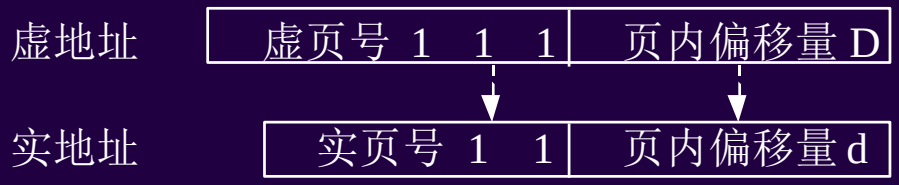
直接相联的地址映象方式与地址变换原理



(a) 虚页集合与实页集合的对应关系

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

(b) 对应关系表（√为有关系）



(c) 通过求模运算进行虚实变换示例

3.3.3 组相联(P178)

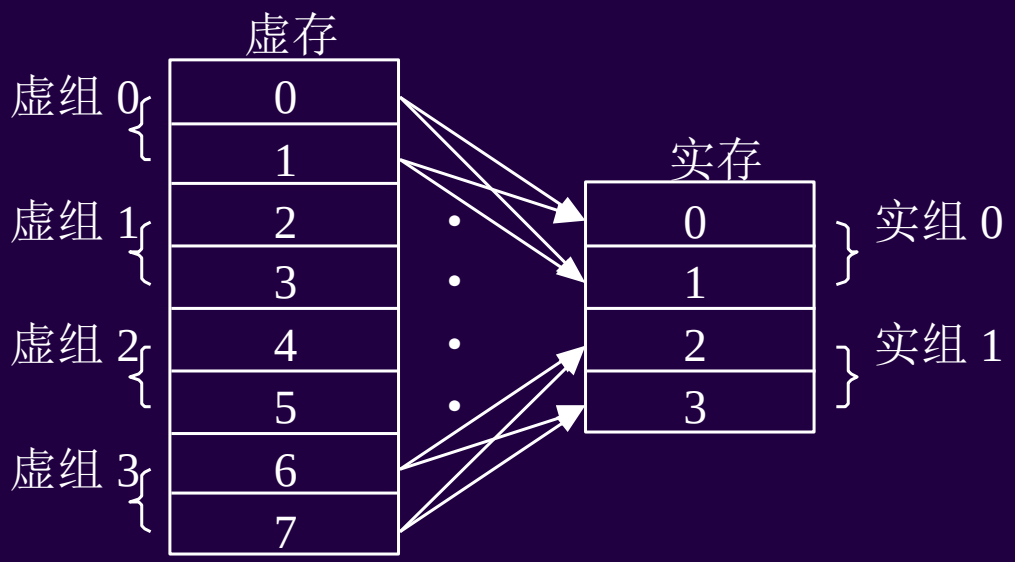
组相联映象方式是全相联与直接相联的一个折中方案，性能也是二者的折中。具体做法是先将实存分组，每组内有若干实页，然后将虚存空间也以同样大小分组。所有虚组按照直接相联方式映射到实组集合，对应的虚实组之间各页则用全相联映射，如下页示意图(a)、(b)所示（设实组数为2）。

由于包含了两层不同的映射关系，页表须按虚组划分成许多子表。在虚实变换时，首先根据虚页号所在的虚组号，通过求模运算确定实组号，再按虚组号在相应的子表内读出组内页号，拼接在一起就是实页号。简记为“组号计算、组内查表”。如图(c)所示。

采用组相联映象方式时，每个虚页在对应实组范围内有若干映象实页可供选择，实页争用的发生频率比直接相联要低；另一方面，由于页表内原来存放的实页号改成存组内页号，省略了实组号字段，所以页表占用空间也减少了。当然这两方面优点是互相抵触的：组内页数越多，实存空间划分的组数就越少，实组号字段所占位数也少，这时改善实页争用现象的效果较好，而节省页表空间的效果较差，反之亦然。实际使用中可根据性能要求选取合适参数。

这种映象方式性价比较好，在Cache-主存层次中被普遍使用。

组相联的地址映象方式与地址变换原理(a)(b)

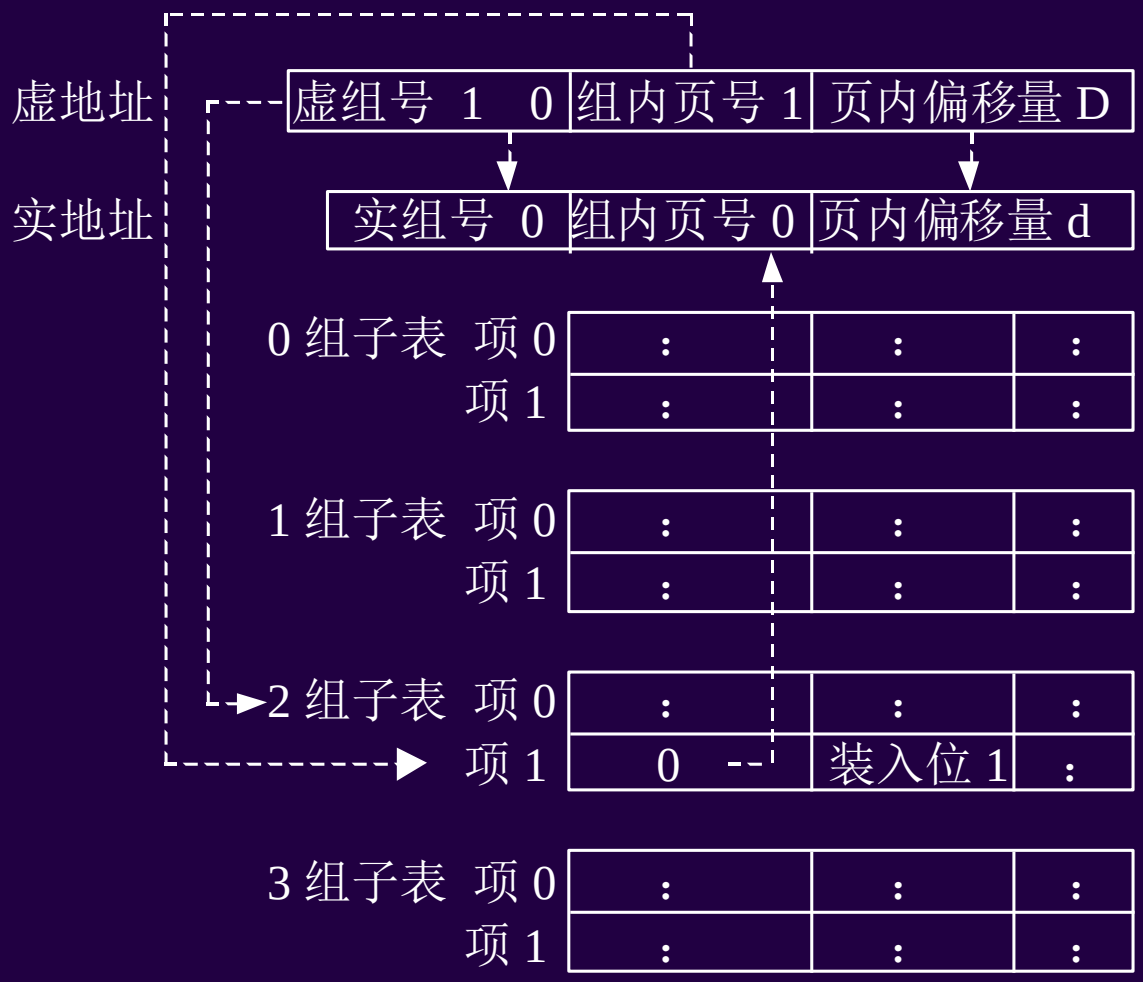


(a) 虚页集合与实页集合的对应关系

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

(b) 对应关系表（√为有关系）

组相联的地址映象方式与地址变换原理(c)



(c) 求模运算与分组查表结合进行虚实变换示例

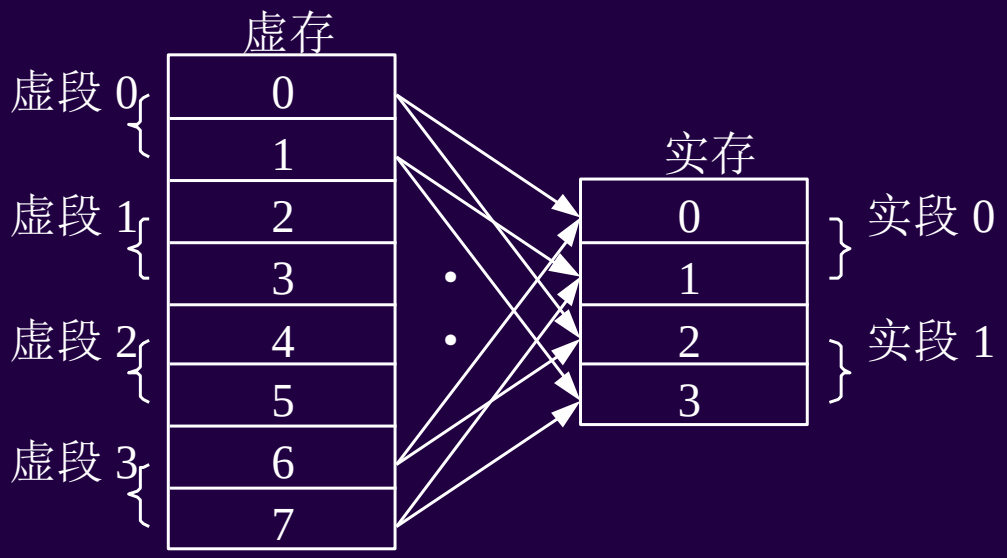
3.3.4 段相联(P184)

段相联映象方式也是全相联与直接相联的一个折中方案。它的分段方法与组相联相同，不同的是所有虚段按照全相联方式映射到实段集合，对应的虚实段之间各页则用直接相联映射（因为虚实段大小相同，所以实际上是一一对应），如下页示意图(a)、(b)所示（设实段数为2）。

段相联的虚实变换与组相联类似，不过可以通过计算来确定的部分不是在段外，而是在段内，即页表内只储存各虚页对应的实段号，段内页号则从虚页号中简单直接复制，拼接在一起就是实页号，简记为“段号查表、段内复制”。如图（c)所示。

段相联映象方式的虚实段内页号对应关系是固定的，每个虚页在调入时可以选择的只是实段号。由于虚实段大小相同，所以虚段号比实段号位数多，也就意味着“多→少”的映射（组相联是等量映射），其实页争用的发生频率比组相联要高。在节省页表存储空间方面，性能与组相联差不多。

段相联的地址映象方式与地址变换原理(a)(b)

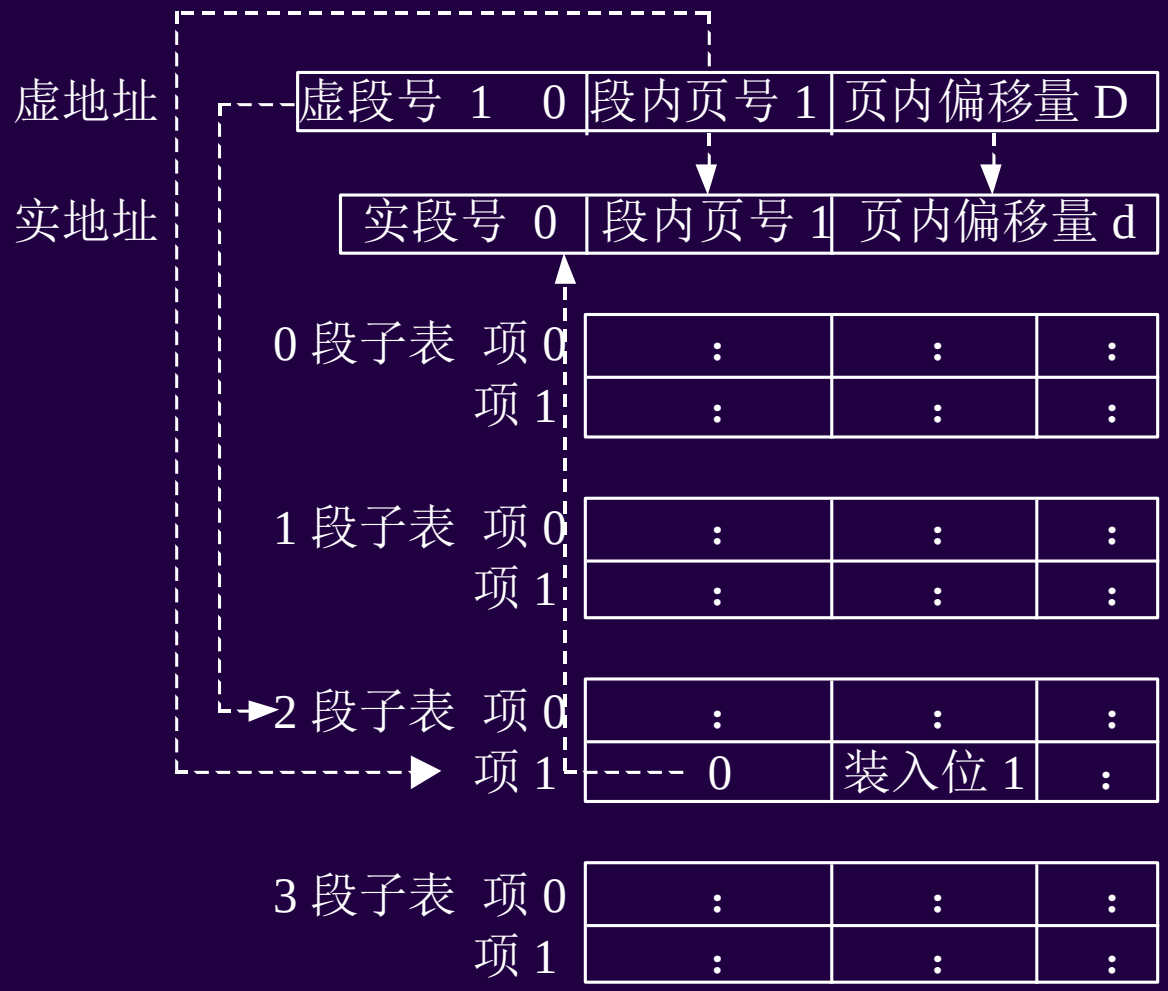


(a) 虚页集合与实页集合的对应关系

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

(b) 对应关系表（√为有关系）

段相联的地址映象方式与地址变换原理(c)



(c) 查表求实段号进行虚实变换示例

例 3.3

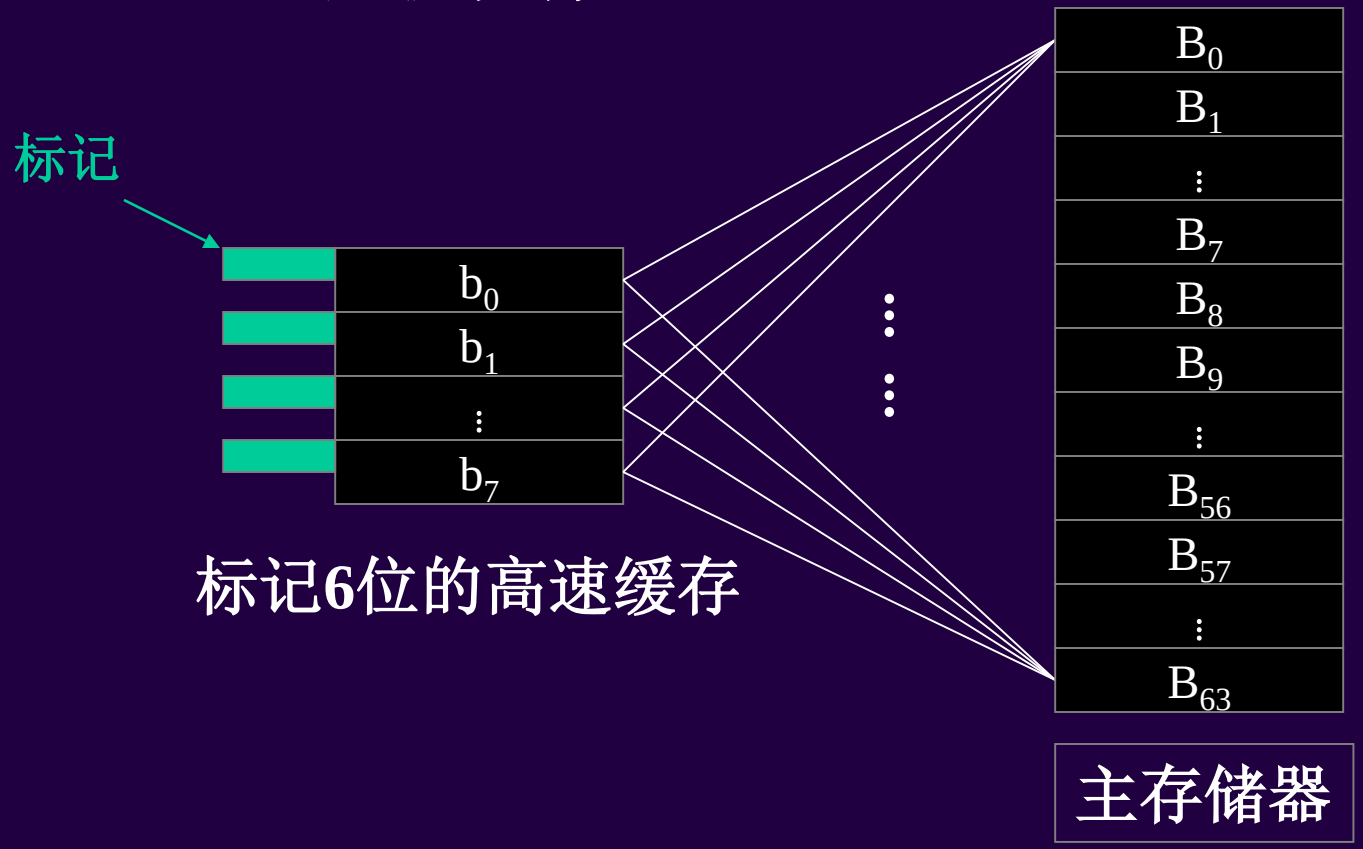
假若一计算机的主存储器按64块组织，块大小为8个字，高速缓存有8块。试按下述要求画图回答问题。

（1）画出全相联映像示意图、指定标记字段和主存地址格式。

（2）画出直接映像示意图、指定标记字段和主存地址格式。

（3）画出2路组相联映像示意图、指定标记字段和主存地址格式。

解:（1）全相联映像

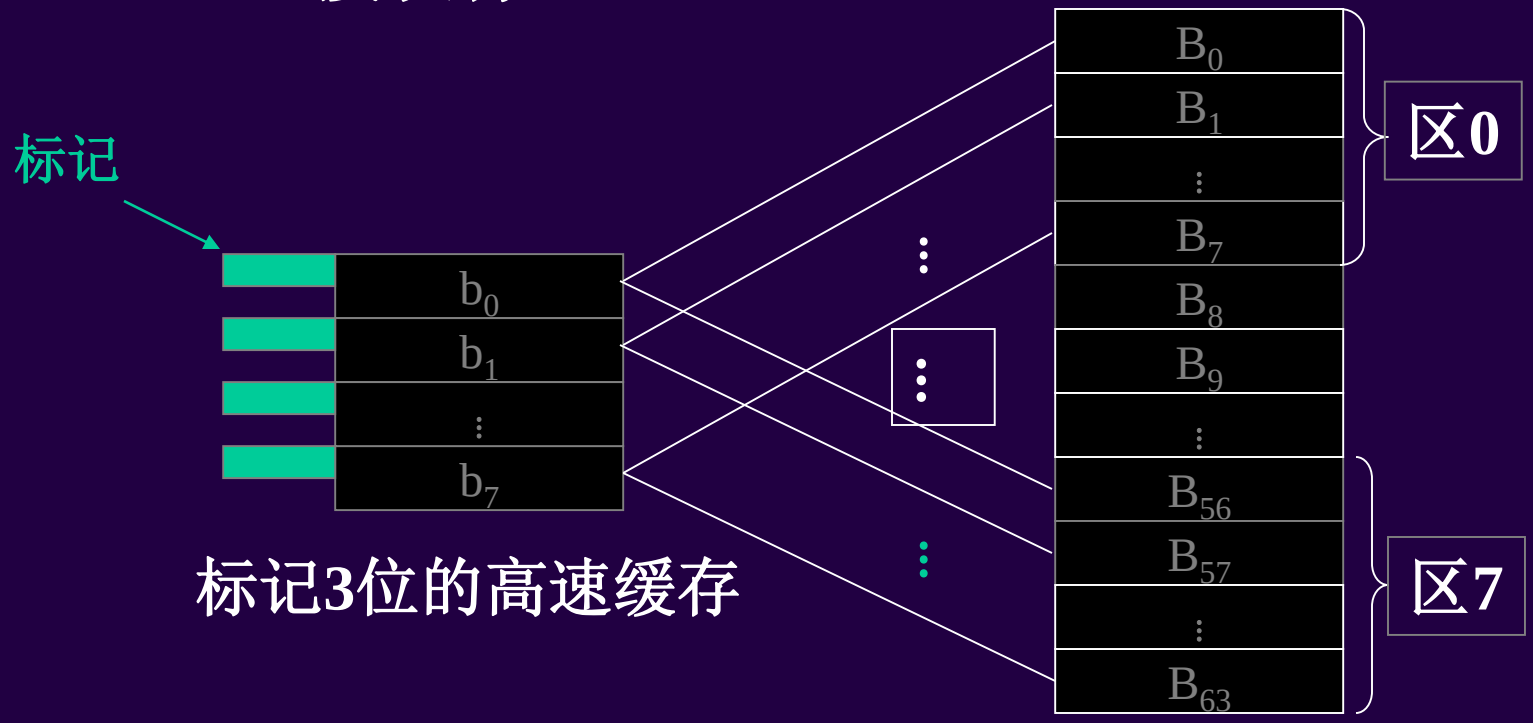


标记6位的高速缓存

主存地址 （块号）标记（6位）字（3位）

缓存地址 行号（3位） 字（3位）

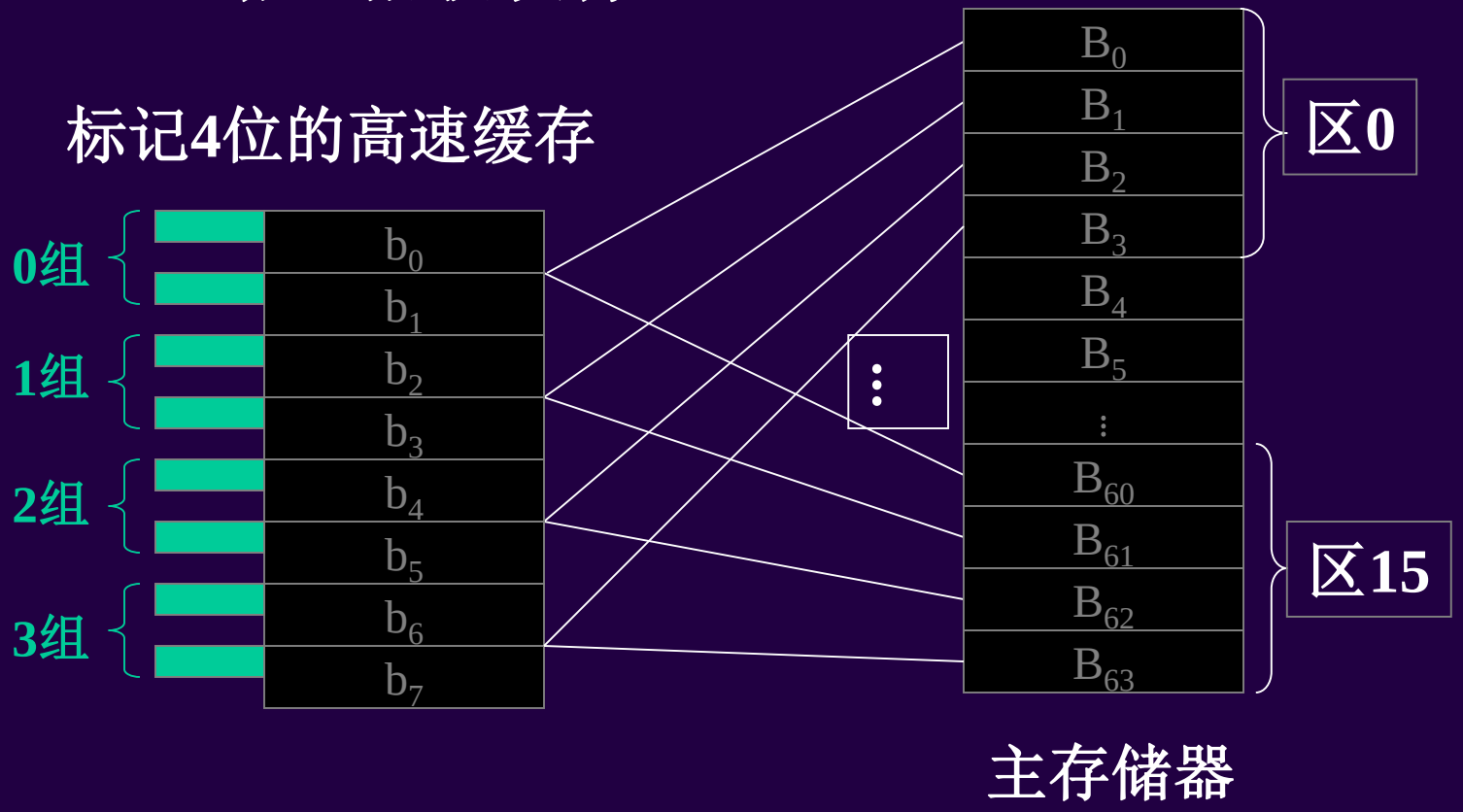
（2）直接映像



	3位	3位	3位
主存地址	区号E（标记）	块号B	字W

	3位	3位
缓存地址	行号L	字w

（3）2路组相联映像



4位 2位 3位

主存地址 | 区号E（标记） | 组号G | 字W |

2位 1位 3位

缓存地址 | 组号g | 行号L | 字w |

3.4 替换算法(P164)

上面所讲的地址映象方式是在虚页调入时的“选址”规则，而地址变换方法则是命中时获得实地址的手段。

不命中时需要增加的操作就是首先调出一页，调出之后再调入称为“替换”。

替换算法要解决的是选择调出对象的问题。

替换算法的目的是在发生实页争用（即根据地址映象方式，将要调入的虚页被允许进入的所有实页均被其它虚页占用）时，选择将来不太可能使用或者使用最晚的虚页作为调出对象，以腾出一个实页来。

3.4.1 几种常用的替换算法(P164)

(1)随机算法RAND —— 在比较范围内任取一页作为淘汰页；

(2)先进先出算法FIFO —— 在比较范围内选取调入最早的一页作为淘汰页；

(3)最不经常使用算法LFU —— 在比较范围内选取最近单位时间内使用次数最少的一页作为淘汰页；

(4)最不接近使用算法LRU —— 在比较范围内选取最后一次使用离现在最久的一页作为淘汰页；

(5)最优替换算法OPT —— 在比较范围内选取下一次使用时间离现在最久的一页作为淘汰页。

实例：实存状况图(P166图3.32)

例如 LRU 算法（其中*号表示被选中的淘汰页）：

已访问次数 t		0	1	2	3	4	5	6	7	8	9	10	命中总次数
被访问虚页号		无	1	2	1	5	4	1	3	4	2	4	
实存空间使用情况（实页号为0、1、2）	0	空	1	1	1	1	1*	1	1	1*	2	2	4次
	1	空	空	2	2	2*	4	4	4*	4	4	4	
	2	空	空	空	空	5	5	5*	3	3	3*	3*	
操作名称		初态(空)	调入	调入	命中	调入	替换	命中	替换	命中	替换	命中	
被访问实页号			0	1	0	2	1	0	2	1	0	1	

[作业3.14 P205]为在页式虚拟存储器中，一个程序由P1～P5共5个页面组成。在程序执行过程中依次访问的页面如下：P2，P3，P2，P1，P5，P2，P4，P5，P3，P2，P5，P2

假设系统分配给这个程序的主存有3个页面，分别采用FIFO、LFU和OPT三种页面替换算法对这3页主存进行调度。

(1)画出主存页面调入、替换和命中的情况表。

(2)统计三种页面替换算法的页命中率。

解：三种替换算法的替换过程：

页地址流	2	3	2	1	5	2	4	5	3	2	5	2
FIFO 命中3次 H=3/12 =1/4	2	2 3	2 3	2* 3 1	5 3* 1	5 2 1*	5* 2 4	5* 2 4	3 2* 4	3 2* 4	3 5 4*	3* 5 2
	调进	调进	命中	调进	替换	替换	替换	命中	替换	命中	替换	替换
LRU 命中5次 H = 5/6	2	2 3	2 3	1 2 3*	5 1 2*	2 5 1*	4 2 5*	5 4 2*	3 5 4*	2 3 5*	5 2 3*	2 5 3*
	调进	调进	命中	调进	替换	命中	替换	命中	替换	替换	命中	命中
OPT 命中6次 H= 1/2	2	2 3	2 3	2 3 1*	2 3* 5	2* 3 5	4* 3 5	4* 3 5	4* 3 5	2 3* 5	2 3 5	2 3 5
	调进	调进	命中	调进	替换	命中	替换	命中	命中	替换	命中	命中

[作业3.15 p205]一个程序由五个虚页组成，采用LFU替换算法，在程序执行过程中依次访问的地址流如下：

4，5，3，2，5，1，3，2，3，5，1，3

(1) 可能的最高页命中率是多少？

(2) 至少要分配给该程序多少个主存页面才能获得最高的命中率。

(3) 如果在程序执行过程中访问一个页面，平均要对该页面内的存储单元访问1024次，求访问存储单元的命中率。

解：（1）由于在页地址流中互不相同的页共有5页，因此最多分配5个主存页面就可获得最高页中命中率，可能的最高命中率为:

$$H=\frac{12-5}{12}=\frac{7}{12}$$

（2）因为LFU替换算法为堆栈型换算法，即随着分配给该程序的主存页面数的减少，其命中率单调递减，所以为获得最高命中率H＝7/12，可采用逐步减少所分配的主存页数的方法来推算，若分配n个主存页面时可获得最高命中率，但分配n－1个页面时命中率却减少，则此时我们可以得出这样的结论：至少要分配给该程序n个主存页面才能获得最高的命中率。

由表可知，至少要分配给该程序4个主存页面才能获得最高的命中率。

页地址流		4	5	3	2	5	1	3	2	2	5	1	3
堆栈内容	S(1) S(2) S(3) S(4) S(5) S(6)	4	5 4	3 5 4	2 3 5 4	5 2 3 4	1 5 2 3 4	3 1 5 2 4	2 3 1 5 4	2 3 1 5 4	5 2 3 1 4	1 5 2 3 4	3 1 5 2 4
实页数	n=1 n=2 n=3 n=4 n>=5					H H H		H H	H H	H H H H H	H H	H H	H H

（3）访问存储单元的命中率为:

$$H = \frac{1024 \times 12 - 5}{1024 \times 12} \approx 0.99959$$

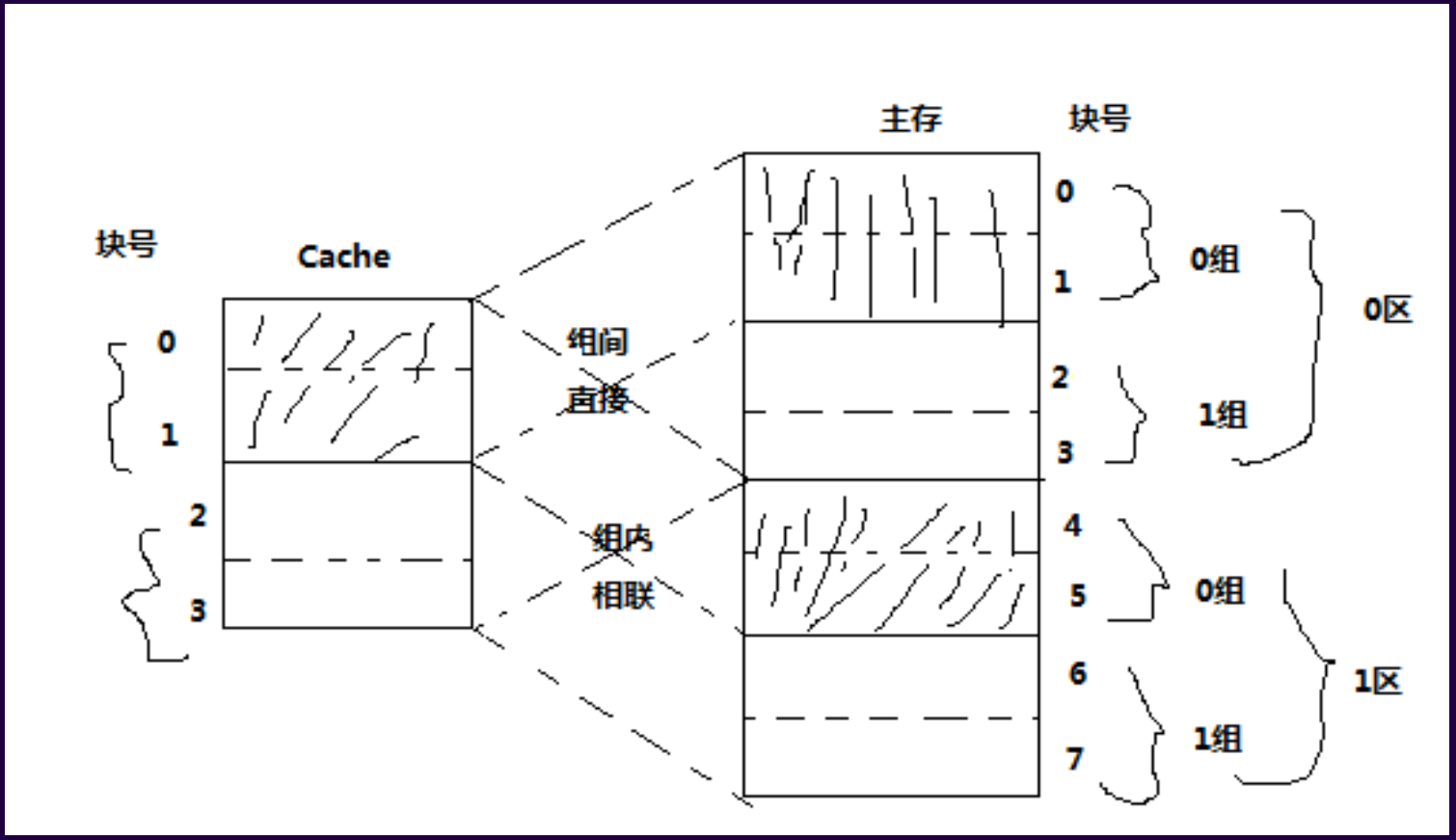
值得说明的是，在此例中，尽管LFU属于堆栈替换算法，但是分配的实际页数n也并不是越多越好，当命中率H达到饱和后，实际页数n的增加不仅不会提高命中率，反而会使实存的利用率下降。

- 例（半期考）： 有一个Cache -主存存储层次，主存共8个块(0~7)，Cache有4个块(0~3)，采用组相联映象，组内块数为2块，替换算法为LRU 算法。

(1)画出主存、Cache空间块的映象对应关系示意图;

(2)对于如下主存块地址流：0、3、5、7、0、1、3、2、7、0、6、4，画出Cache内各块的实际替换过程图，并计算此期间的Cache命中率Hc。

1.主存、Cache空间块的映象对应关系示意图



（2）对于如下主存块地址流：0、3、5、7、0、1、3、2、7、0、6、4，画出Cache内各块的实际替换过程图如下：

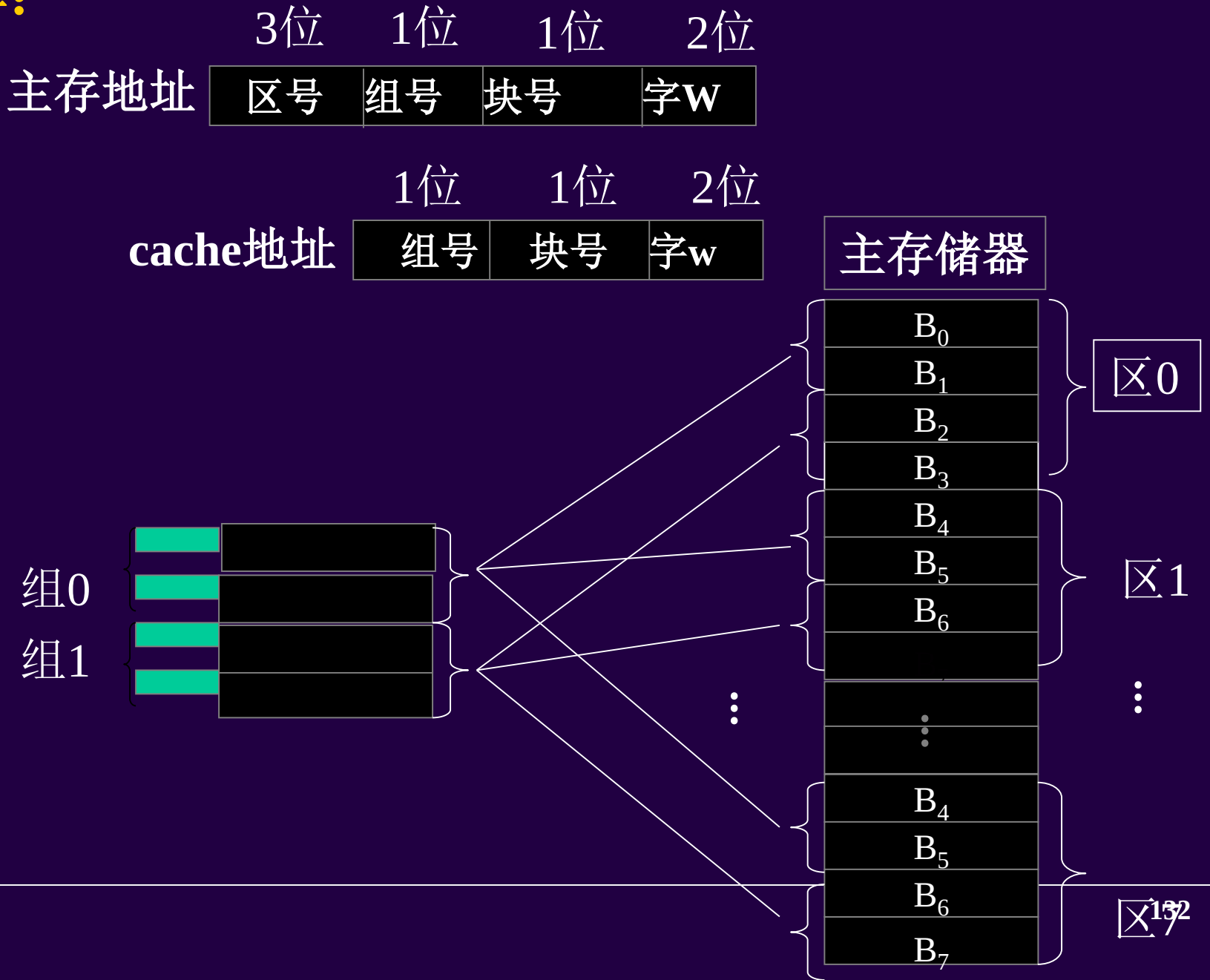
时间t		1	2	3	4	5	6	7	8	9	10	11	12
主存块地址		0	3	5	7	0	1	3	2	7	0	6	4
Cache块	0	0	0	0*	0*	0	0*	0*	0*	0*	0	0	0*
	1			5	5	5*	1	1	1	1	1*	1*	4
	2		3	3	3*	3*	3*	3	3*	7	7	7*	7*
	3				7	7	7	7*	2	2*	2*	6	6
命中情况		失	失	失	失	命中	替	命中	替	替	命中	替	

Hc=3/12

- **题3.4** 对于下述访存字节地址序列：

1,14,50,89,20,17,19,56,19,11,14,43,15,16,9,17
标出每次访存后的cache存储空间的分配情况和命中情况。假定cache是2路组相联的，采用FIFO替换策略，每块是4个32位的字。Cache的容量是16字，初始cache为空。

解：


3位
1位
1位
2位
主存地址
区号
组号
块号
字W
1位
1位
2位
cache地址
组号
块号
字w
主存储器
B0
B1
B2
B3
B4
B5
B6
B4
B5
B6
B7
区0
区1
区7
组0
组1

（2路组相联FIFO）

时间 t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
字地址流	**1**	**14**	**50**	**89**	**20**	**17**	**19**	**56**	**19**	**11**	**14**	**43**	**15**	**16**	**9**	**17**
组0	1	1	1	1*	20	20*	19	19	19	19	19	19	19	19	19*	17
			50	50	50*	17	17	17	17	17	17	17	17*	16	16	16
组1		14	14	14	14	14	14*	56	56	56*	14	14*	15	15	15	15
				89	89	89	89	89	89*	11	11*	43	43	43*	9	9
	调进	调进	调进	调进	替换	替换	替换	替换	命中	替换	替换	替换	替换	替换	替换	替换

命中1次

多用户虚地址格式

在多用户或多进程并发环境下，由于机器中同时保存并交替运行多个程序模块，各模块中的相同虚页号会发生混淆。这时从CPU发出的虚地址还需要在前面拼接上一个“当前用户号”字段，形成“多用户虚地址”，如下图所示(参见P154)。

在虚实变换时，上面所说的各种查表操作之前还得先去查一个“段表基址寄存器组”或“页表基址寄存器组”的小表格(P150，P152)，确定现在该查哪一张段表或页表。这个小表格建立在CPU里，读写时间很短。

- 例3.5 设某用户虚存共有8页, 主存有4页, 每页大小为1KB. 试根据页表计算出虚地址1023和6800的主存实地址。

页表

虚页号	实页号	装入位
0	3	1
1	1	1
2	2	0
3	3	0
4	2	1
5	1	0
6	0	1
7	0	0

提示：注意页表中虚、实页对应关系

解：页号与地址对应关系

第0页	0—1023
第1页	1024—2047
第2页	2048—3071
第3页	3072—4095
第4页	4096—5119
第5页	5120—6143
第6页	6144—7167
第7页	7168--8191

虚页号＝虚地址 / 1024

虚地址1023，虚页号为0，页内位移为1023；根据虚页号查页表得知实页号为3，且装入位为1。
主存实地址PA=3072+1023=4095

虚地址6800，虚页号为6，页内位移为656；根据虚页号查页表得知实页号为0，且装入位为1。
主存实地址PA=0+656=656

每页首地址=页号 X 每页大小

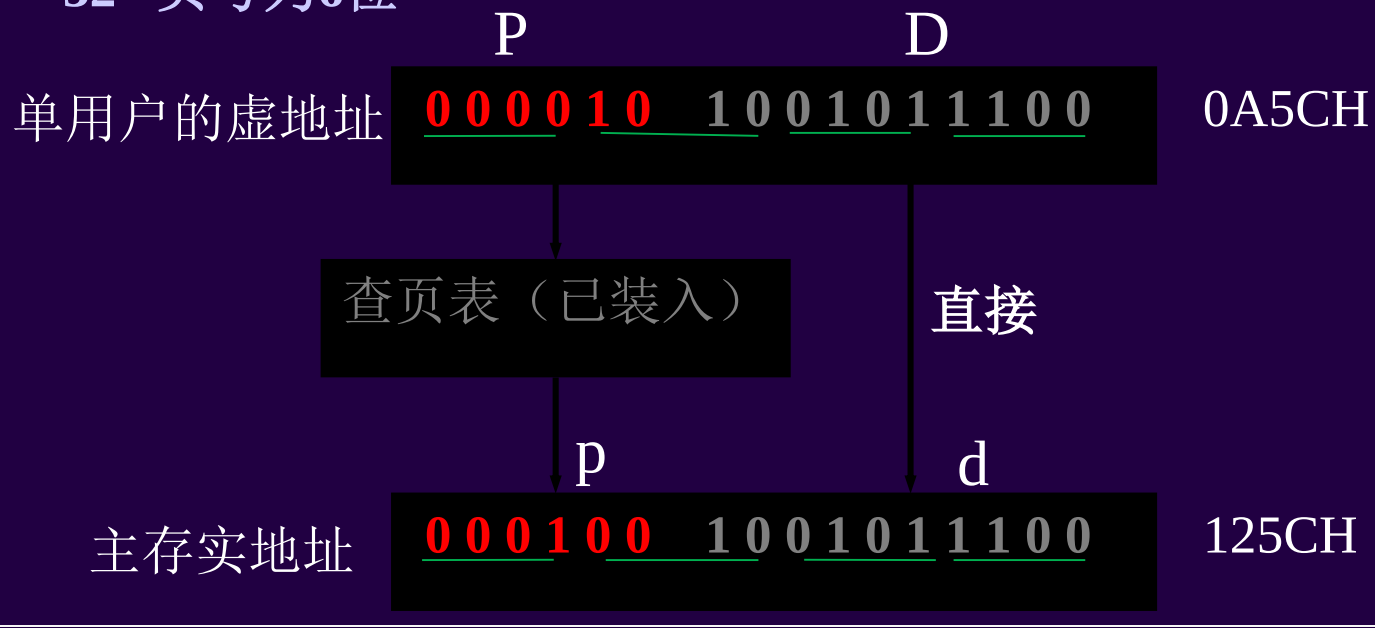
- 例3.6 虚存32页, 主存16页, 每页为1KB。某时刻已调入主存的实页与虚页对应如下:

  虚页号　0　1　2　8

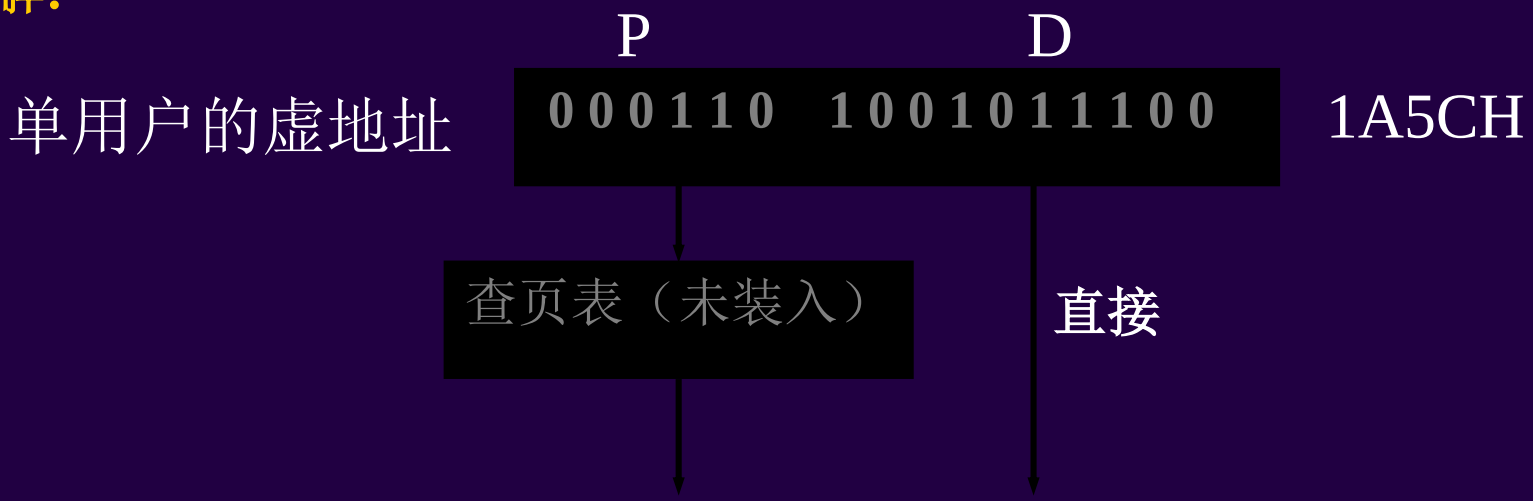
  实页号　5　10　4　7

  求虚地址0A5CH和1A5CH对应的实地址。

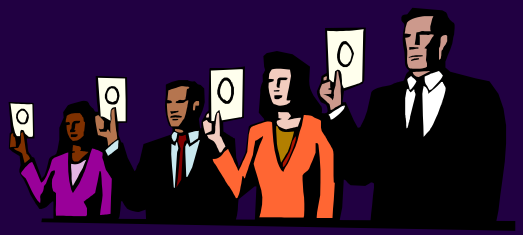
解：2^6 =32　页号为6位



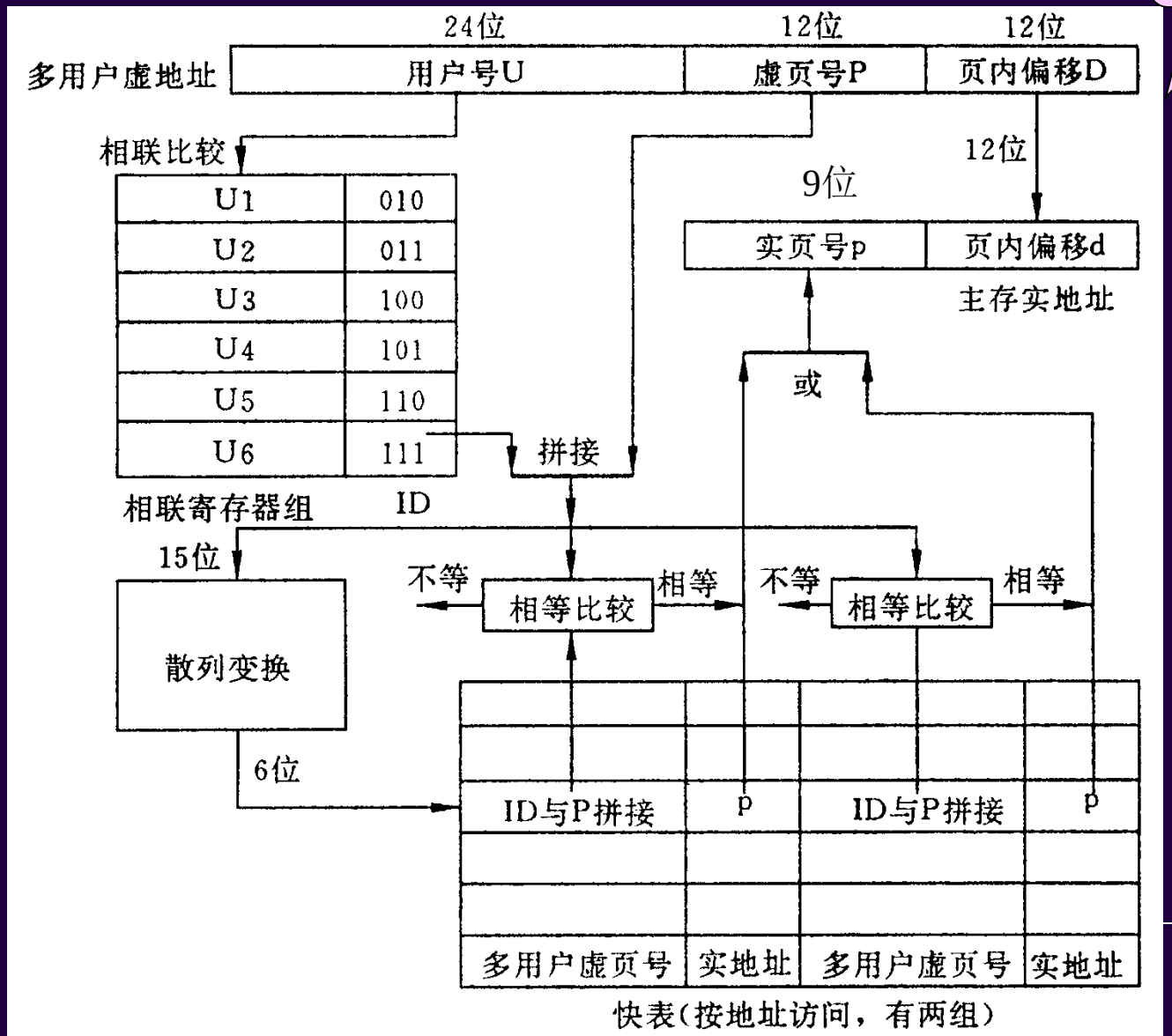
解：



用户的虚页并未装入主存中，当执行该虚页程序时，找不到对应的实页。



例3.7 虚拟存储器举例—IBM37



2^{12}=4K

2^{12}=4KB

15位

64行

24+24位

用

每 页数？

每

相

相

快表行

快表每行的位数？

如何减少散列冲突？

快表为何采用两组？

相等比较的作用？

例 3.8

有1K个任务，短期内只有4个用户，
每个用户程序空间可达4096页，每页512B，
主存地址长度为20位，虚实地址经快表变换，
问相关设计：

（1）实页位数 ？

（2）每个相联寄存器的相联比较位数

（3）每个相联寄存器的总位数

（4）散列变换硬件的输入和输出位数

（5）每个相等比较器的位数为

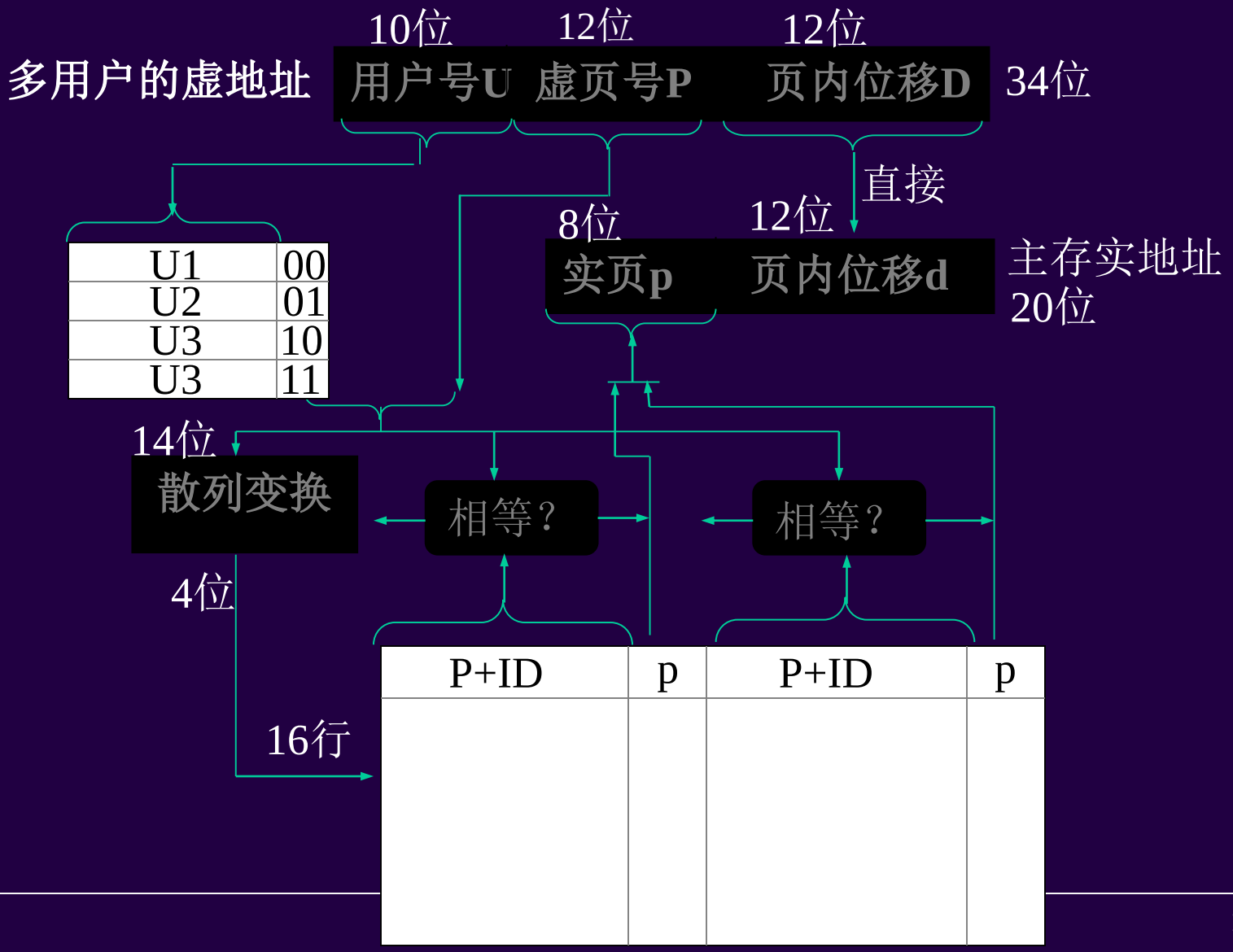
（6）快表的总容量

●3.8 解：

（1）∵可有1K个任务，短期内只有4个用户，
∴指示用户号的地址字段U=10位；ID=2位；
又∵每个用户程序空间可达4096页，∴P=12,
∵每页512B，512*8= 4096 位
∴页内位移D=(9+3)=12位；
又∵主存地址长度为20位，　∴ 实页号p=20-12=8位

（2）每个相联寄存器的相联比较位数为U=10位（用户号）
（3）每个相联寄存器的总位数为U+ID=12位；
（4）散列变换硬件的输入和输出位数为14位和4位
（5）每个相等比较器的位数为P+ID=12+2=14位；
（6）快表的总容量为16×(12+2+8)×2=704（位）

虚实地址经快表变换的逻辑示意图



3.2.2 性能指标（P132-P134）

(1) 容量：$S=S_2$ （理论上）
(2) 单价：（美分/bit）

$$c=\frac{c_1\cdot S_1+c_2\cdot S_2}{S_1+S_2}=\frac{\frac{S_1}{S_2}\cdot c_1+c_2}{\frac{S_1}{S_2}+1}$$

它的最小值是 $\lim\limits_{\frac{S_1}{S_2}\to 0} c=c_2$

（3）速度：表现访问速度的参数很多

- 命中率：反映被访问数据事先已在M_1的发生概率

$$H = \frac{N_1}{N_1 + N_2} \quad , \quad 0 \le H \le 1$$

- 等效访问时间：命中时的访问时间为T_1，不命中时的访问时间为T_2，等效访问时间则是它们的概率均值

$$T = H \cdot T_1 + (1 - H) \cdot T_2$$

$$\lim_{H \to 100\%} T = T_1$$

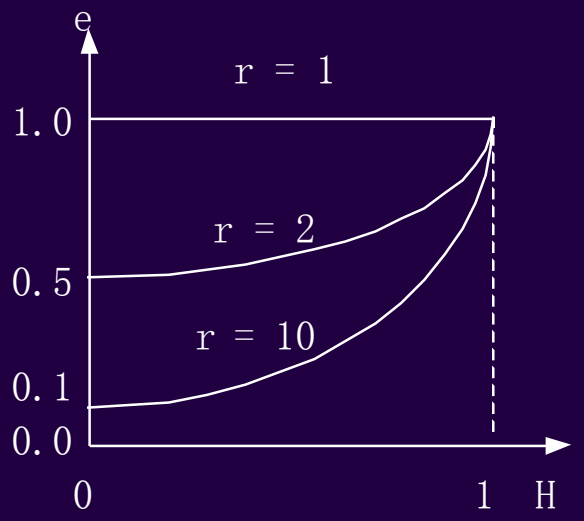
- 访问效率：这是一个相对值，便于不同系统之间的比较。

$$e = \frac{T_1}{T}$$

$$= \frac{T_1}{H \cdot T_1 + (1-H) \cdot T_2}$$

$$= \frac{1}{H + (1-H) \cdot r}$$

$$= \frac{1}{r + (1-r) \cdot H}$$



H 和 r 对 e 的作用

其中$0 \le e \le 1$，$r = \frac{T_2}{T_1} \ge 1$（$r$ 是邻级速度比）。

访问效率e受H和r的影响（参见右图）：

$H \uparrow \rightarrow e \uparrow$，$r \uparrow \rightarrow e \downarrow$。

- Cache预取技术对命中率的提高作用（P134）：

这里所说的“预取”技术，并不是根据对程序执行的未来趋势进行猜测以提前调入数据，而仅仅是在发生不命中情况时把调入1个数据字改为调入1个数据块的策略。根据程序的局部化原理，在当前使用数据周围的其它数据未来被使用的几率大于远处数据，所以该数据块中被提前调入的邻近数据很可能成为未来的命中点，从而提高命中率。

采用这种预取技术后新的命中率为

$$H' = \frac{H + n - 1}{n}$$

其中：H —— 原命中率（即按照不命中时取入1字的策略）；
H’ —— 新命中率（即按照不命中时取入1块的策略）；
n —— 每块数据平均被访问次数。

H’的推导：

按照定义，原不命中率 $1-H=\frac{N_2}{N_1+N_2}$，新不命中率 $1-H'=\frac{N_2'}{N_1'+N_2'}$，并且有 $N_1+N_2=N_1'+N_2'$ 。

由于预取使得每块数据中的不命中次数由n次降低到1次，所以有 $N_2'=\frac{1}{n}N_2$。此式可改写为 ，

整理得 $H'=\frac{H+n-1}{n}$ 。

- 加速比（P193）

Cache-主存层次的主要作用是提高访问速度，系统的等效速度应高于主存（即M_2）的原有速度，两个速度之比称为加速比。

$$S_p = \frac{\text{等效速度}}{M_2\text{速度}} = \frac{M_2\text{时间}}{\text{等效时间}} = \frac{T_2}{T}$$

$$= \frac{T_2}{H \cdot T_1 + (1-H) \cdot T_2}$$

$$= \frac{1}{(1-H) + H/r}$$

- 增加中间层对e的影响

- **3.9** 在一个Cache存储系统中， Cache的访问周期为10ns，主存储器的访问周期为60ns，每个数据在Cache中平均重复使用4次。当块大小为1个字节时，存储系统的访问效率只有0.5，现在要通过增加块大小，使存储系统的访问效率达到0.94。
- （1）当存储系统的访问效率为0.5时，计算命中率和等效访问周期。
- （2）为了使存储系统的访问效率达到0.94，命中率和等效访问周期应该提高到多少？
- （3）为了使存储系统的访问效率从0.5提高到0.94，块的大小至少增加到几个字？

解:（1）当存储系统的访问效率为0.5时，由表达式

$$e=\frac{T_{A1}}{T_A}=\frac{T_{A1}}{HT_{A1}+(1-H)T_{A2}}=\frac{1}{H+(1-H)T_{A2}/T_{A1}}$$

可求出命中率为

$$0.5=\frac{1}{H_1+(1-H_1)60/10}\qquad H_1=0.8$$

等效访问周期为

$$T_1=T_c\times H_c+(1-H_c)\times T_m$$

$$=10\times 0.8+60\times 0.2=20(ns)$$

或由 $e=\frac{T_{A1}}{T_A}=\frac{T_c}{T_1}=\frac{10}{T_1}=0.5$ 得 $T_1=20(ns)$

（2）当存储系统的访问效率提高到0.94时，命中率应该提高到H_2

$$0.94=\frac{1}{H_2+(1-H_2)60/10} \qquad H_2=0.987$$

等效访问周期应提高为

$$T_2=10\times 0.987+60\times(1-0.987)=10.64(ns)$$

或由 $e=\frac{T_{A1}}{T_A}=\frac{T_c}{T_2}=\frac{10}{T_2}=0.94$ 得 $T_1=10.64(ns)$

（3）为了使存储系统的访问效率由0.5提高到0.94，块大小应为B个字。则有

$$H_2 = \frac{H_1 + n - 1}{n}$$

其中n为Cache的块大小与数据重复使用次数的乘积，H_1是原来的命中率，H是块大小增加后的命中率。

$$H_2 = \frac{H_1 + 4B - 1}{4B}$$

在上式中代入相关参数，可求出

$$B = 4$$

3.5 虚拟存储器与Cache的特点(P146，P172)

- 虚拟存储器与Cache的主要区别

  (P173表3.4)

- Cache的主要组成与工作流程

  (P173图3.38)

第4章总线、中断与I/O系统

4.1输入输出系统概述

4.2总线设计

3.3中断系统

4.4通道处理机

4.5外围处理机

4.1　基本输入输出方式(P212)

4.1.1 程序控制I/O方式

4.1.2 中断I/O方式

4.1.3 DMA方式

4.1.4 通道方式

4.1.5 I/O处理机方式

4.2 中断优先级管理(P219)

中断是为实时任务优先获得处理机资源而采用的一种调度技术，当系统中存在多个中断源时必须根据实时性强弱设定优先顺序，这也被称为中断的分级。为了兼顾中断响应的时效与配置的灵活，通常采用两套机制结合组成中断优先序管理体系。

(1)硬件响应优先序：未被屏蔽的几个中断源同时提出申请时，CPU选择服务对象的顺序。它由硬件电路实现，用户不能修改。如P226图4.11所示。

(2)软件服务优先序：在各中断服务程序开头，用软件设置自己的中断屏蔽字（在主程序中也设置）。以此改变实际服务顺序(P230)。

例如某个硬件响应优先级高的中断源，其中断服务程序执行中屏蔽了自身，而开放了某个硬件响应优先级比它低的中断源，后者就可以在前者刚开放中断时就打断它，从而在实际上先得到服务。

中断服务过程示意图如P231图4.14所示。

由于常规用户主程序对处理机的需求紧迫性最低，所以它的中断屏蔽字是“全部开放”。

(3)实例分析：屏蔽字表、中断服务过程图。

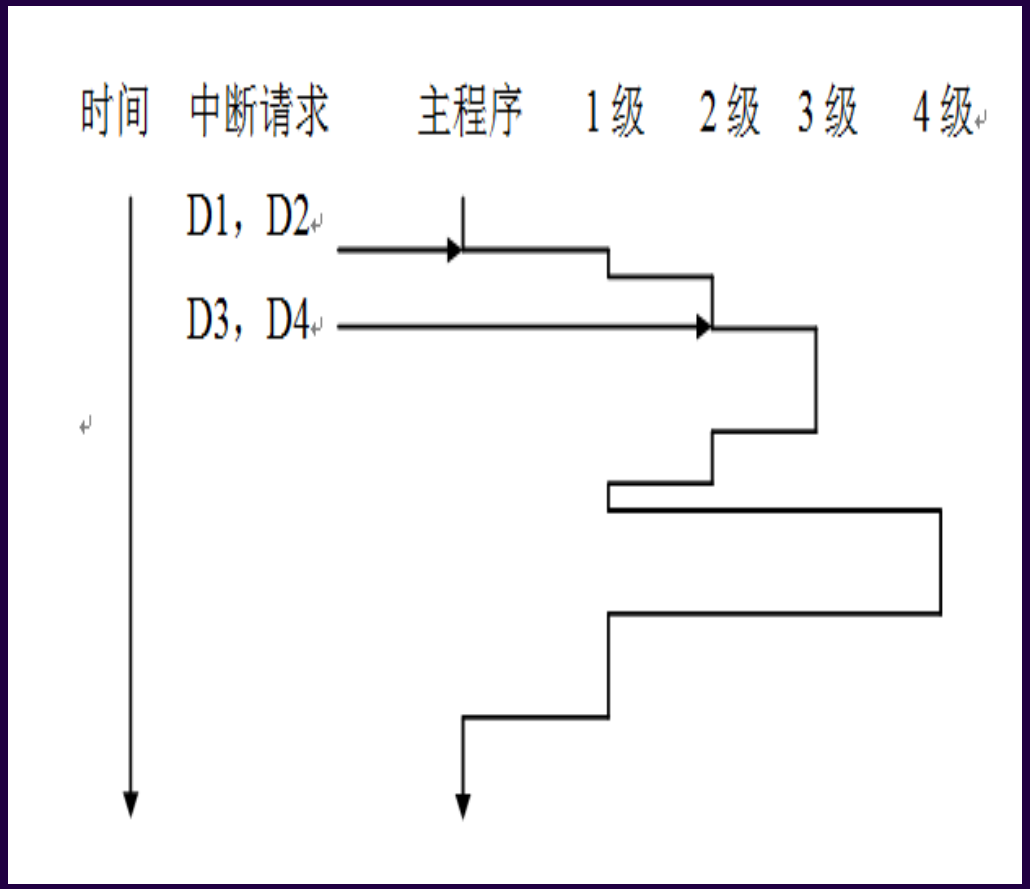
- 例4.1（P230倒数第8行开始）

作业4.5 已知中断服务请求次序为 1-2-3-4，现改为3-2-4-1，（1）设计中断屏蔽码（2）处理机运行主程序时，同时D1和D2q请求中断，而在运行中断源D2时，D3和D4又同时请求，请画出程序运行过程示意图
解：(1)中断屏蔽字表如下图；(令1对应屏蔽，0对应开放)

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

(2)中断过程示意图如下图。

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



例 4.1 设中断级屏蔽位为“0”对应于开放，1对应于屏蔽，各级中断处理程序的中断级屏蔽位设置如下：

中断处理程序级别	中断级屏蔽位			
	1级	2级	3级	4级
第1级	1	1	1	1
第2级	0	1	0	0
第3级	0	1	1	1
第4级	0	1	0	1

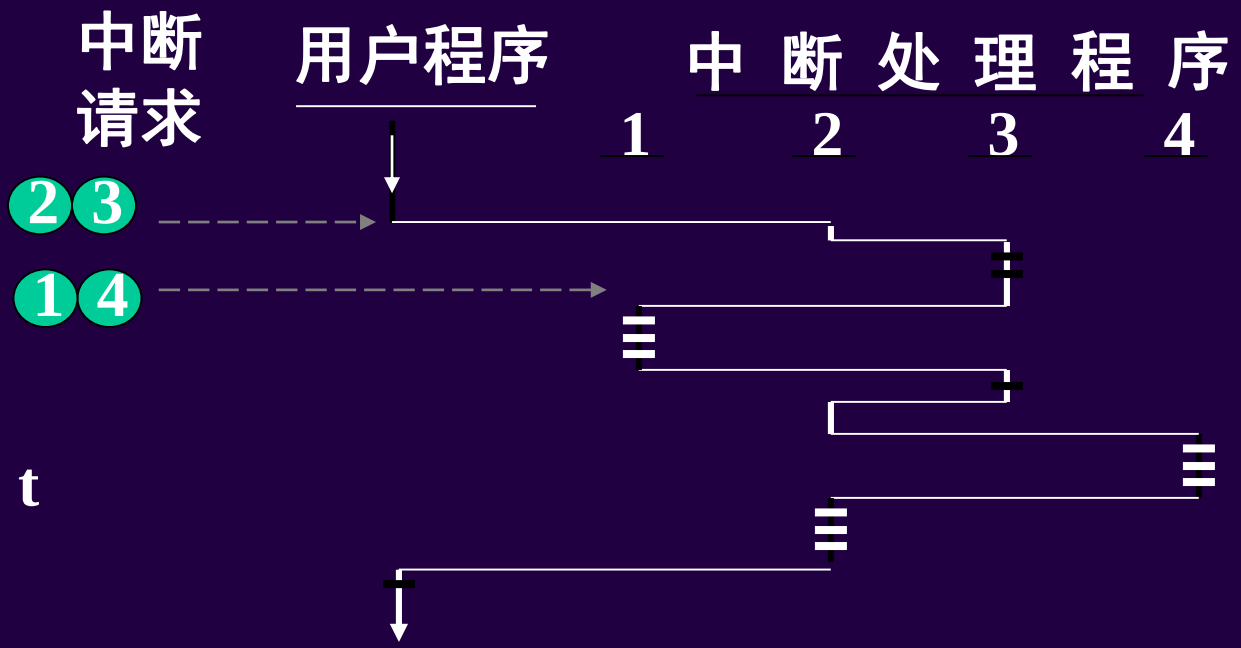
（1）当中断响应的优先次序为1、2、3、4，其中断处理次序是什么？

（2）所有的中断处理各需要3个单位时间，当正在运行用户程序时，同时出现第2、3级中断请求，过2个单位时间，又同时出现第1、4级中断请求，请画出程序运行过程示意图。

【解答】

(1) 当中断响应的优先次序为1、2、3、4，其中断处理次序是1、3、4、2。

(2) 运行过程示意图如下：



4.3 通道处理机(P233)

(1)定义：

通道处理机（简称"通道"）是隶属于主处理机的输入输出专用协处理机。

(2)特点：

- 有一套输入输出功能很强的专用指令系统；
- 与主处理机共享主存，存放相应的程序和数据；
- 一个通道可以连接多台外部设备；
- 主处理机可用"启动I/O"指令来启动一个通道；
- 当通道访存与主处理机冲突时，存控部件赋予通道较高的优先权；
- 通道程序执行完毕自动转入休眠状态，同时向主处理机发出一个特定的中断申请，通知该事件。

(3)地位：

从属于主处理机。

(4)分类(P238)：

- 字节多路通道：以字节为单位交叉为多台设备传输。子通道的概念。
- 选择通道：完成一台设备的全部传输再去为另一台设备服务。
- 数组多路通道：以数组为单位交叉为多台设备传输。

(5)通道传输过程的时间分配 (P241，其中P是设备台数)：

- 字节多路通道：$T_{BYTE} = (T_S + T_D) \cdot P \cdot n$ ，其中n是单台设备的数据传输量；

- 选择通道：$T_{SELECT} = (\frac{T_S}{n} + T_D) \cdot P \cdot n$

- 数组多路通道：$T_{BLOCK} = (\frac{T_S}{k} + T_D) \cdot P \cdot n$ ，其中k是块尺寸，$k < n$ 。

(6)通道流量分析(P243):

- 通道最大能力流量:

$$f_{MAX.BYTE} = \frac{1}{T_S + T_D}$$

$$f_{MAX.SELECT} = \frac{1}{\frac{T_S}{n} + T_D}$$

$$f_{MAX.BLOCK} = \frac{1}{\frac{T_S}{k} + T_D}$$

- 通道实际最大负荷流量：

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

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

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

- 通道正常工作条件：

$$f_j \leq f_{MAX.j}$$

• 实例分析：

通道时间关系图

•P243 例 4.1 倒数第2行开始

已知：$f_1=\frac{1}{10\times10^{-6}秒}$，$f_2=\frac{1}{30\times10^{-6}秒}$，$f_3=\frac{1}{30\times10^{-6}秒}$，

$f_4=\frac{1}{50\times10^{-6}秒}$，$f_5=\frac{1}{75\times10^{-6}秒}$。求通道实际流量$f_{BYTE}$，并作时间图。

解：（见教材$P243-P244$）

例4.1　某字节交叉多路通道连接6台设备，其数据传送速率如下表所示。

设备号	1	2	3	4	5	6
传送速率(B/ms)	50	50	40	25	25	10
二次请求的间隔时间(μS)						

(1) 在上表中填出设备相应二次请求传送字节的间隔时间。

(2) 当所有设备同时要传送数据时，求其对通道要求的总流量f_{byte}

(3)让通道以极限流量$f_{max.byte}=f_{byte}$的工作周期工作，通道的工作周期(即TS+TD的时间间隔)是多少？

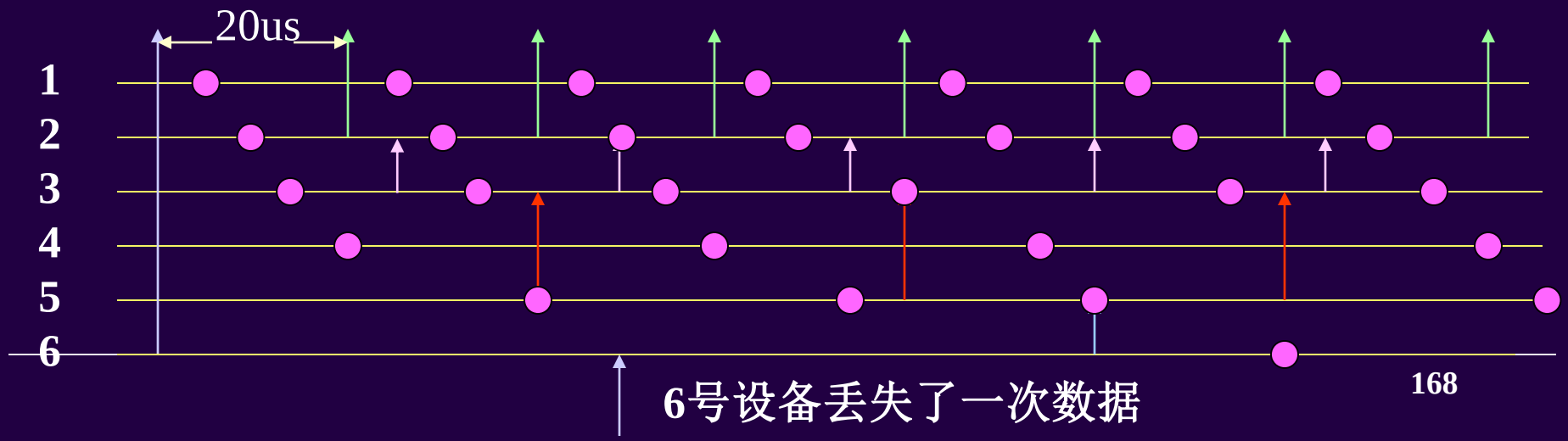
(4)让通道中所挂设备速率越高的数据传送请求被响应的优先级越高。画出6台设备同时发送请求到下次同时发送请求期间里，通道响应和处理完各设备请求时刻的示意图。哪个设备丢失了信息？提出一种不丢失信息的解决办法。

解：（1）

设备号	1	2	3	4	5	6
传送速率(B/ms)	50	50	40	25	25	10
二次请求的间隔时间(μS)	20	20	25	40	40	100

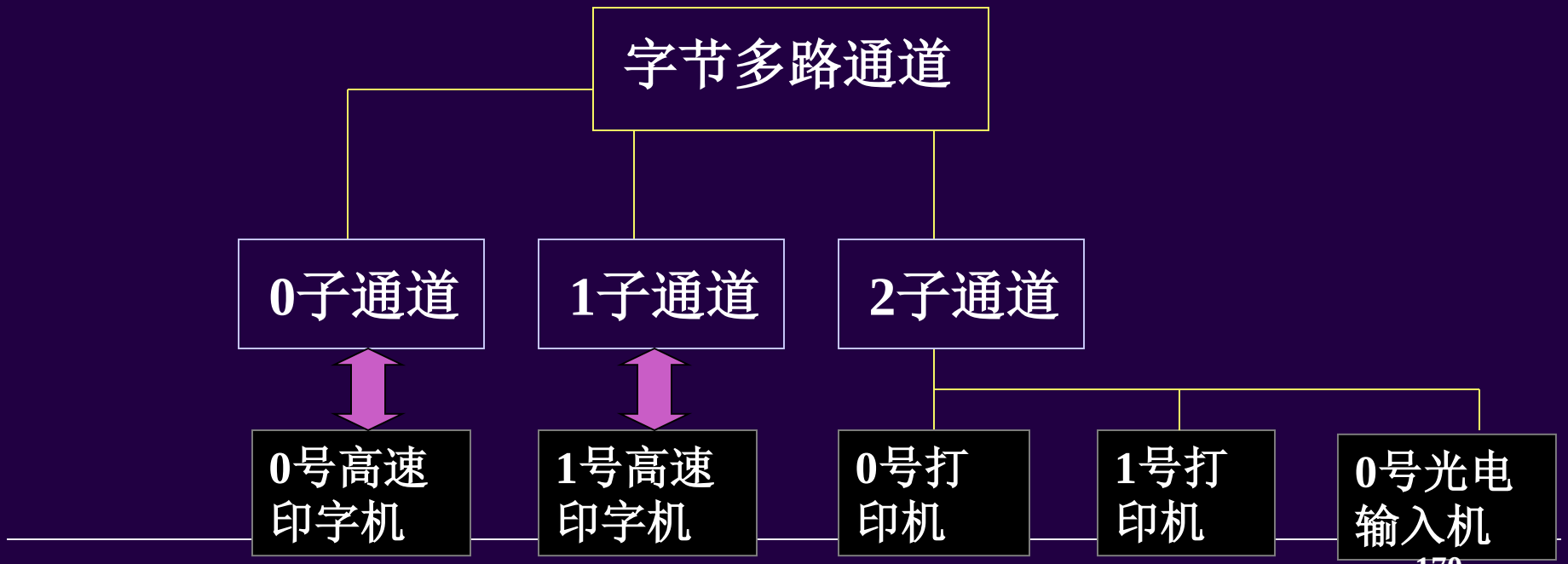
（2）总容量 $f_{byte}=\sum_{i=1}^{6} f_i = 200B/ms$

（3）传送周期 $T_S+T_D=1ms/200B=5\ \mu S$



- 方法1：增加通道的最大流量，保证连接在通道上的所有设备的数据传送请求能够及时得到通道的响应
- 方法2：动态改变设备的优先级
- 方法3：增加一定数量的数据缓冲器，特别是对优先级比较低的设备

例4.2 印字机各占一个子通道，0号打印机、1号打印机和0号光电输入机合用一个子通道。假定数据传送期内高速印字机每隔25us发一个请求，低速打印机每隔150us发一个字节请求，光电输入机每隔800us发一个字节请求，则这5台设备要求通道的实际流量为多少？



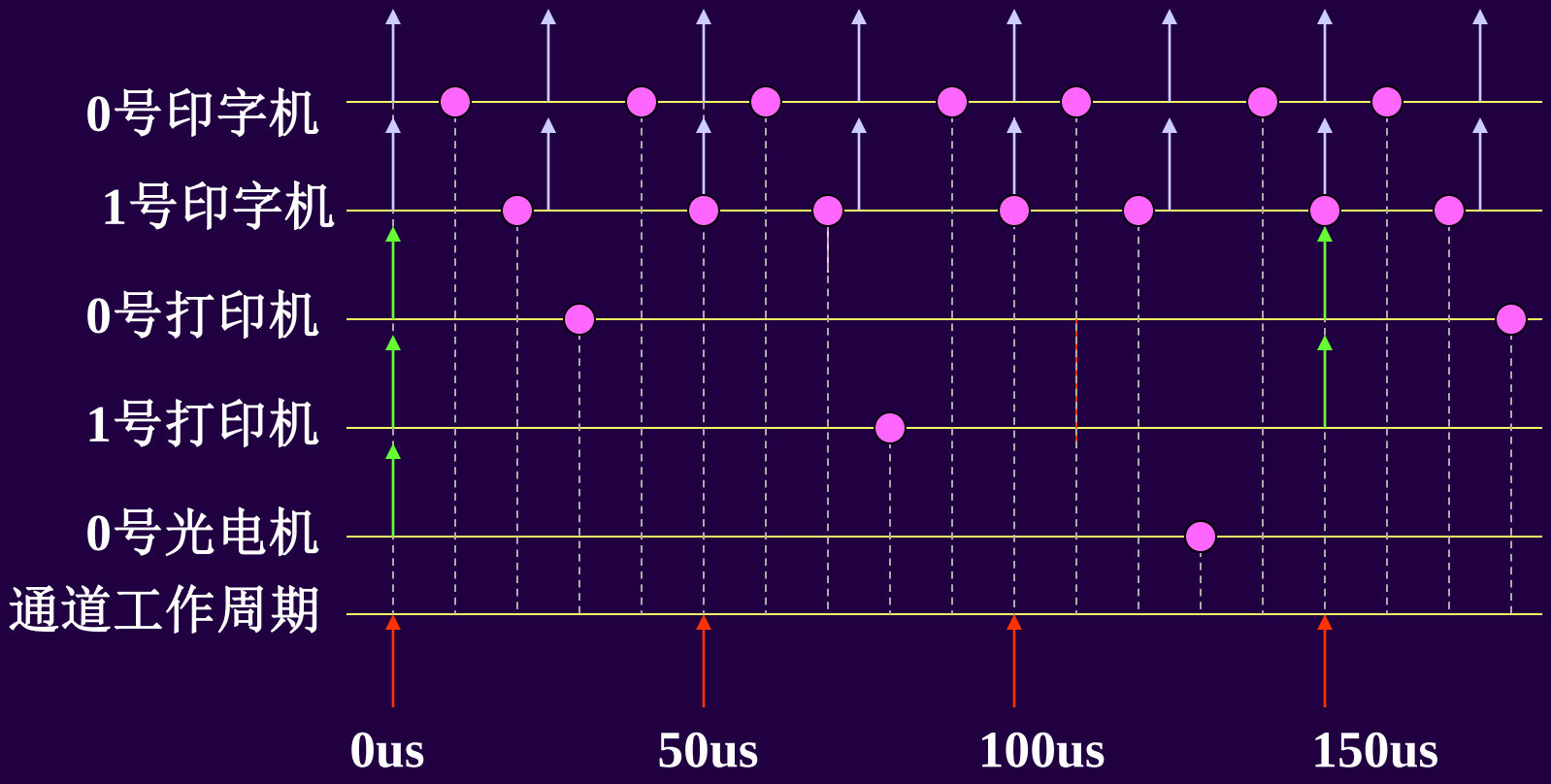
解：5台设备要求通道的数据流量为：

$$f_{byte}=\sum f=\frac{1}{25}+\frac{1}{25}+(\frac{1}{150}+\frac{1}{150}+\frac{1}{800})=0.095MB/s$$

可将该通道设计成0.1MB/s，即所设计的工作周期为：

$$t=\frac{1}{f_{byte}}=\frac{1}{T_S+T_D}=10us$$

这样各设备的请求就能及时得到响应和处理，不会丢失信息。



↑ 表示设备提出申请的时刻

● 表示通道处理完设备申请的时刻

优先级次序：0号印字机、1号印字机、0号打印机、1号打印机、0号光电机

例4.3 设通道在数据传送期中，选择设备需4.9μS，传送一个字节数据需0.1μS。

（1）其低速设备每隔250μS发出一个字节数据传送请求，问最多可接多少台这种设备？

（2）若有A～E共5种高速设备，要求字节传送的间隔时间如下表所示，其时间单位为μS。若一次通信传送的字节数不少于1024个字节，问哪些设备可挂在此通道上？哪些则不能？

设备	A	B	C	D	E
时间间隔（μS）	0.13	0.1	0.11	0.2	0.3

解：(1)低速设备应接字节多路通道

$$f_{\max \bullet byte}=\frac{1}{T_S+T_D}=\frac{1}{4.9+0.1}=\sum_{i=1}^{n} f_{byte \bullet i}=\frac{n}{250}$$

n ＝250/5=50台，所以，n≤50台，即最多可接50台这种设备

(2)根据题意，此通道为选择通道

$$f_{\max \bullet select}=\frac{1}{T_D+\frac{T_S}{n}}=\frac{1}{0.1+\frac{4.9}{n}}$$

其中，n≥1024，应使$f_{select \bullet i} \leq f_{max \bullet select}$

由此可得出通道工作周期为：T≈0.1048(us)

所以，只有A、C、D、E可挂在此通道上，B则不行。

第五章　标量流水线技术(P253)

本章学习标量计算机上使用的流水加速技术。主要内容有流水技术的分类、流水线性能指标计算、非线性流水线的调度算法。

标量计算机指只能直接进行标量运算的计算机，与能够直接进行向量运算的向量计算机相对应。

流水处理方式的特征，是让多个依次启动的任务，尽量同时使用系统的不同部件，通过时间重叠来提高处理速率。这种技术理论上不增加成本。

标量计算机上使用的流水加速技术属于指令级并行技术。

每条指令的处理过程，可以划分为取指、译码、取数、运算、送结果5个子过程，也可以分得更细或更粗一些。划分的原则是各部分时间长度大致相等、并使用CPU中不同的部件，这样才有利于多任务重叠处理。

5.2 流水处理与逻辑相关的概念

CPU中的各个部件按流水处理顺序连接起来，就称为一条流水线。

5.2.1 流水线工作原理

处理机解释程序的方式有*顺序方式*、*重叠方式*、*流水方式*等。

- *顺序方式*是解释完一条指令再开始解释下一条(P254)；
- *流水方式*是把一个重复的过程分解为若干个子过程，每个子过程可以与其它子过程同时进行，以此提高单位时间内解释指令的数目(P277)；
- *重叠方式*是一种简单的流水方式，它把指令分成2个子过程，每条指令只与下一条指令相重叠(P255)。

• 流水线结构图(P278)

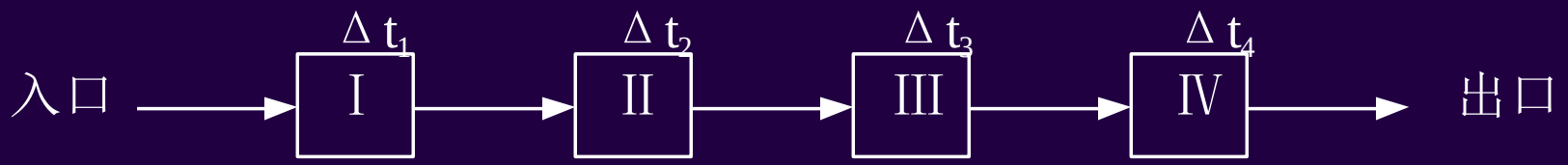


图 5.1 流水线结构图

• **流水线工作时空图(P278—P279)**

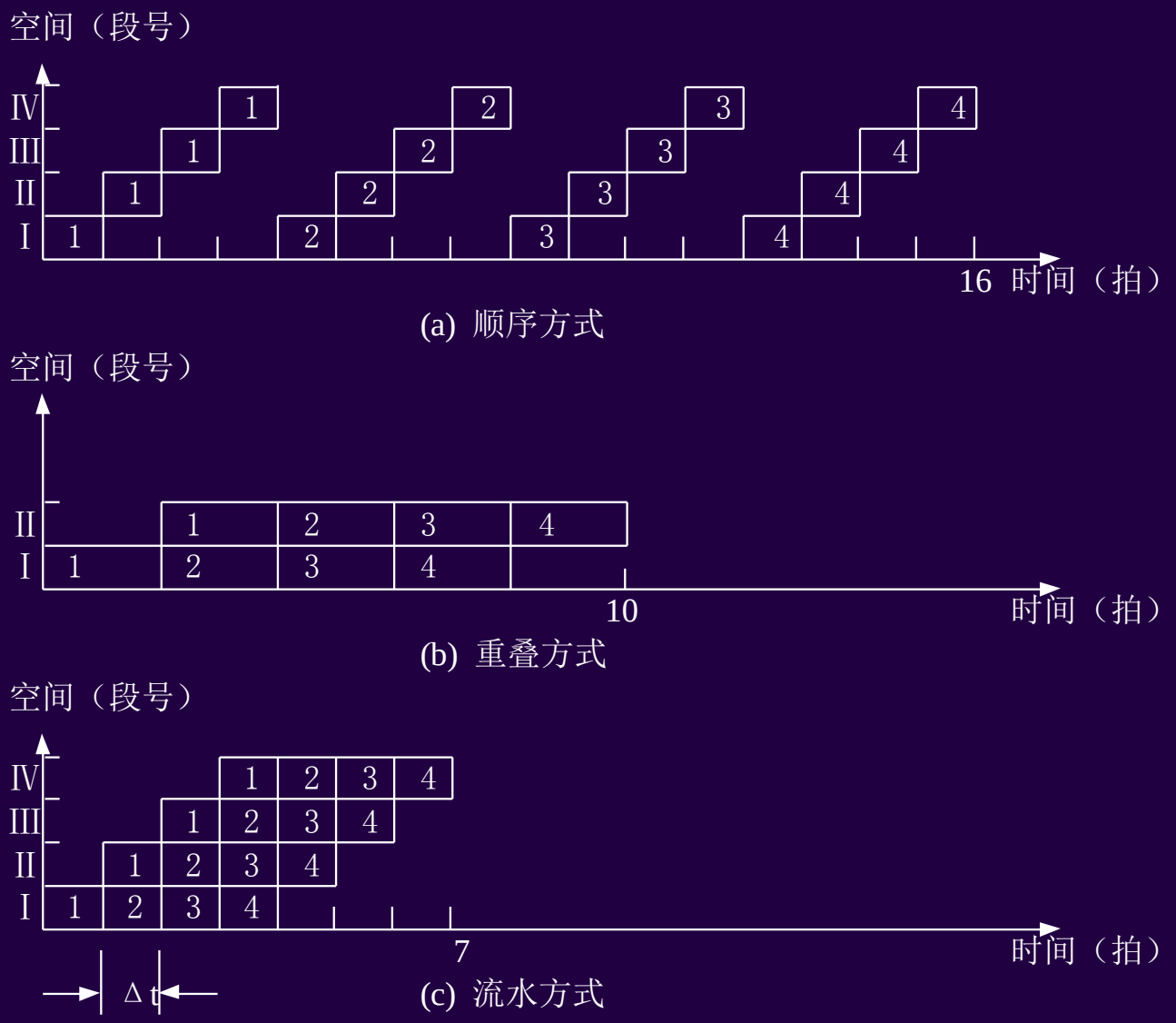


图 3.2 CPU 工作时空图

5.2.2 逻辑相关(P263-276)

- 相关的定义：(P263倒数第4段)

  一条指令必须等待前一条指令解释完成才能开始解释。

- 相关的分类及其对策

1. 全局性相关/局部性相关(P312、P269/P263、P303)；
2. 指令相关/数相关(P264/P263)；
3. 主存数相关/寄存器数相关(P265/P266)；
4. 数值相关/变址值相关(P266/P268)。

5.2.2 逻辑相关(P263-276)

减少流水线分支延迟的静态方法有哪些？

答：（1）预测分支失败：沿失败的分支继续处理指令，就好象什么都没发生似的。当确定分支是失败时，说明预测正确，流水线正常流动；当确定分支是成功时，流水线就把在分支指令之后取出的指令转化为空操作，并按分支目标地址重新取指令执行。

（2）预测分支成功：当流水线ID段检测到分支指令后，一旦计算出了分支目标地址，就开始从该目标地址取指令执行。

（3）延迟分支：主要思想是从逻辑上“延长”分支指令的执行时间。把延迟分支看成是由原来的分支指令和若干个延迟槽构成。不管分支是否成功，都要按顺序执行延迟槽中的指令。

3种方法的共同特点：它们对分支的处理方法在程序的执行过程中始终是不变的。它们要么总是预测分支成功，要么总是预测分支失败。

流水线中有哪三种相关？各是什么原因造成的？

（1）结构相关：当硬件资源满足不了指令重叠执行的要求,而发生资源冲突时,就发生了结构相关。

（2）数据相关：当一条指令需要用到前面指令的执行结果，而这些指令均在流水线中重叠执行时，就可能引起数据相关。

（3）控制相关：当流水线遇到分支指令和其它能够改变PC值的指令时，就会发生控制相关。

解决流水线数据相关的方法有哪些？

（1）定向技术：在某条指令产生一个结果之前，其他指令并不真正需要该计算结果，如果将该计结果从其产生的地方直接送到其他指令需要它的地方，就可以避免暂停。

（2）暂停技术：设置一个“流水线互锁”的功能部件，一旦流水线互锁检测到数据相关，流水线暂停执行发生数据相关指令后续的所有指令。直到该数据相关解决为止。

（3）采用编译器调度。

（4）重新组织代码顺序。

5.3 流水技术的分类(P280)

- 线性/非线性(P280)：
- 部件级/处理机级/处理机间级（宏流水线） (P281)：
- 单功能/多功能(P282)：
- 静态/动态(P283)：
- 标量/向量(P285)：
- 同步/异步(P285)：
- 顺序/乱序(P285、P304)：

5.4 线性流水线性能分析(P285)

5.4.1 吞吐率TP(P285)

吞吐率（**TP——ThroughPut**）指流水线在单位时间内执行的任务数，可以用输入任务数或输出任务数表示。

$TP = \frac{n}{T_k}$ ，其中**k**表示流水线划分的段数。

当满足 $\Delta t_i \equiv \Delta t$ 条件时，有 $T_k = (n + k - 1) \cdot \Delta t$ 。

5.4.2 加速比(P288)

$$S = \frac{T_o}{T_k}$$

其中 $$T_o = n \cdot \sum_{i=1}^{k} \Delta t_i$$

5.4.3 效率（设备利用率，P289）

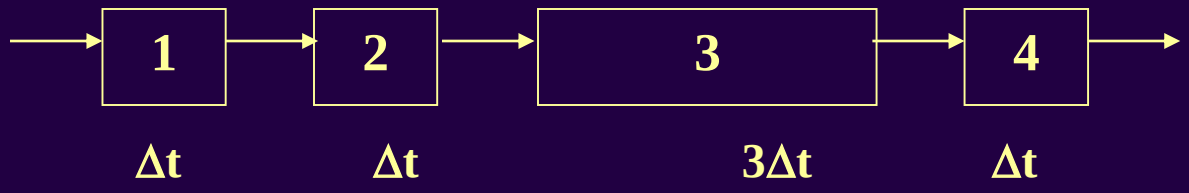
段效率：$E_i = \dfrac{n \cdot \Delta t_i}{T_k}$ ，　　各段平均效率：$E = \sum_{i=1}^{k} (\alpha_i \cdot E_i)$

其中 α_i 表示第i段设备量占整条流水线全部设备量的百分比。

当满足 $\Delta t_i \equiv \Delta t$ 和 $\alpha_i \equiv \dfrac{1}{k}$ 条件时，有：

$$E = \frac{1}{k} \cdot \sum_{i=1}^{k} (E_i) = \frac{n}{k \cdot T_k} \cdot \sum_{i=1}^{k} \Delta t_i = \frac{T_o}{k \cdot T_k}$$

例5.1 带有瓶颈部件的4功能段流水线, $\Delta t_1=\Delta t_2=\Delta t_4=\Delta t$, $\Delta t_3=3\Delta t$, 4个任务、10个任务时TP，E、SP 。

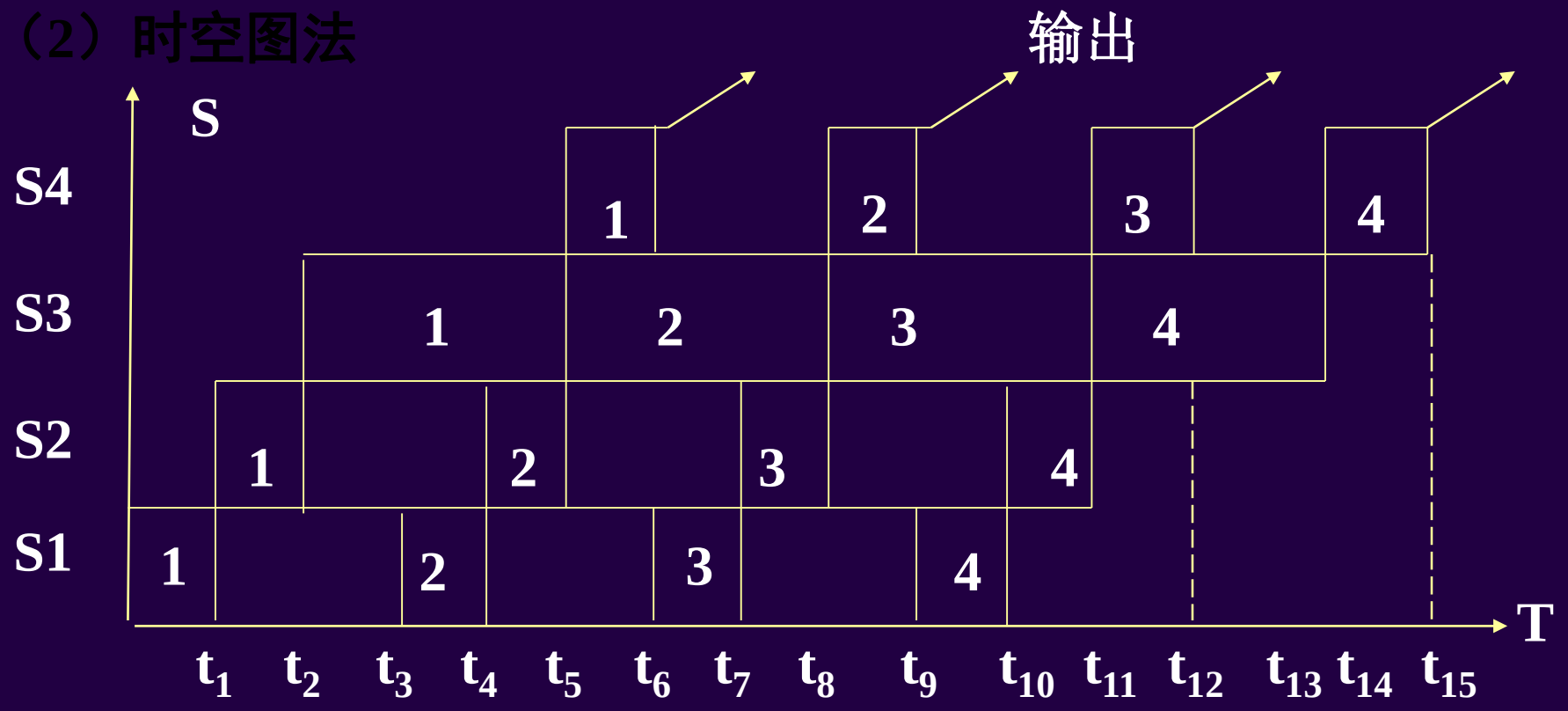


（1）分析法: 各段时间不等

$$TP=\frac{n}{\sum \Delta t_i+(n-1)\Delta t_{max}}$$

$$n=4\text{时}, \quad TP=\frac{4}{(6+9)\Delta t}=\frac{4}{15\Delta t}$$

（2）时空图法



$$E=\frac{\text{n个任务实际占用的时-空区}}{\text{M各段总的时-空区}}=\frac{4\times 6\Delta t}{4\times 15\Delta t}=\frac{6}{15}=0.4=40\%$$

$$Sp=\frac{n*\sum_{i=1}^{m}\triangle t_i}{\sum_{I=1}^{m}\triangle t_i+(n-1)*\triangle t_j}=\frac{4*6\ \triangle t}{15\ \triangle t}=\frac{24}{15}=1.6$$

例5. 2 以浮点加法运算为例（四段流水线）各段时间相等，求吞吐率、效率。

求Z=A+B+C+D+E+F+G+H，TP、E、Sp (注意有相关)

Z=A+B+C+D+E+F+G+H

1 2 3 4

5 6

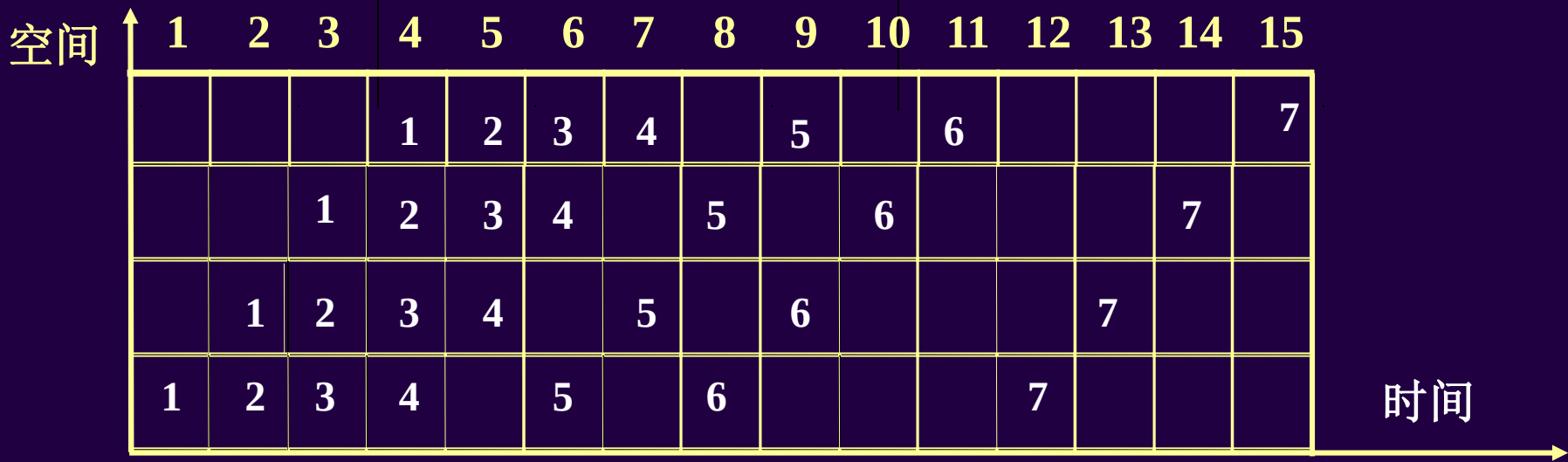
7

解:

TP=7/15△t

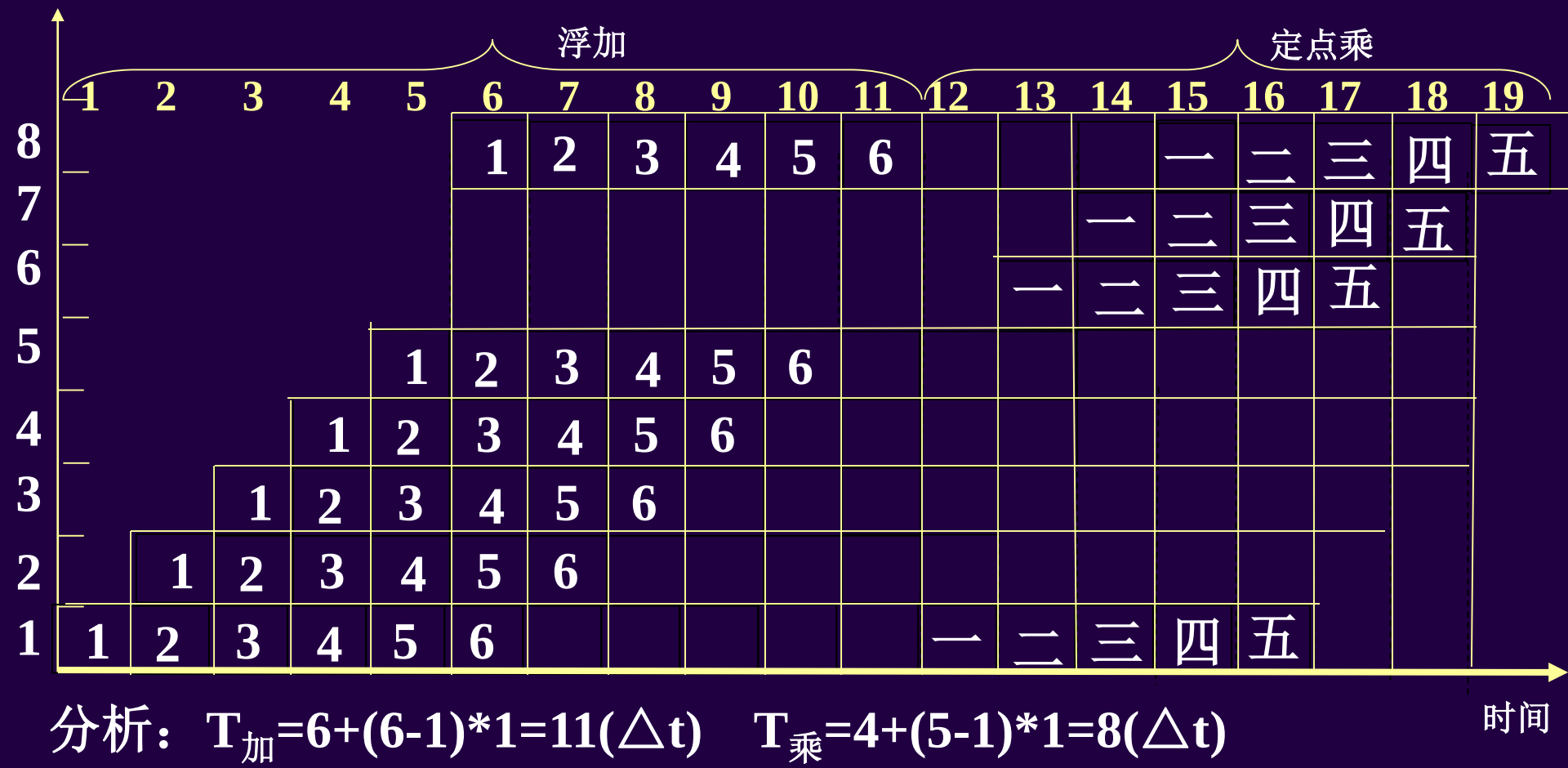
E=7*4/(15*4)=7/15＝46%

Sp=4*7/15=28/15=1.87



流水线的效率不高，原因在于存在着数据相关，有空闲功能段。

例5.3 ASC计算机多功能算术运算流水线各段时间相等，6次浮点加、 5次定点乘的吞吐率，效率，加速比　　m=8，n=11



分析：$T_{加}$=6+(6-1)*1=11(△t)　$T_{乘}$=4+(5-1)*1=8(△t)

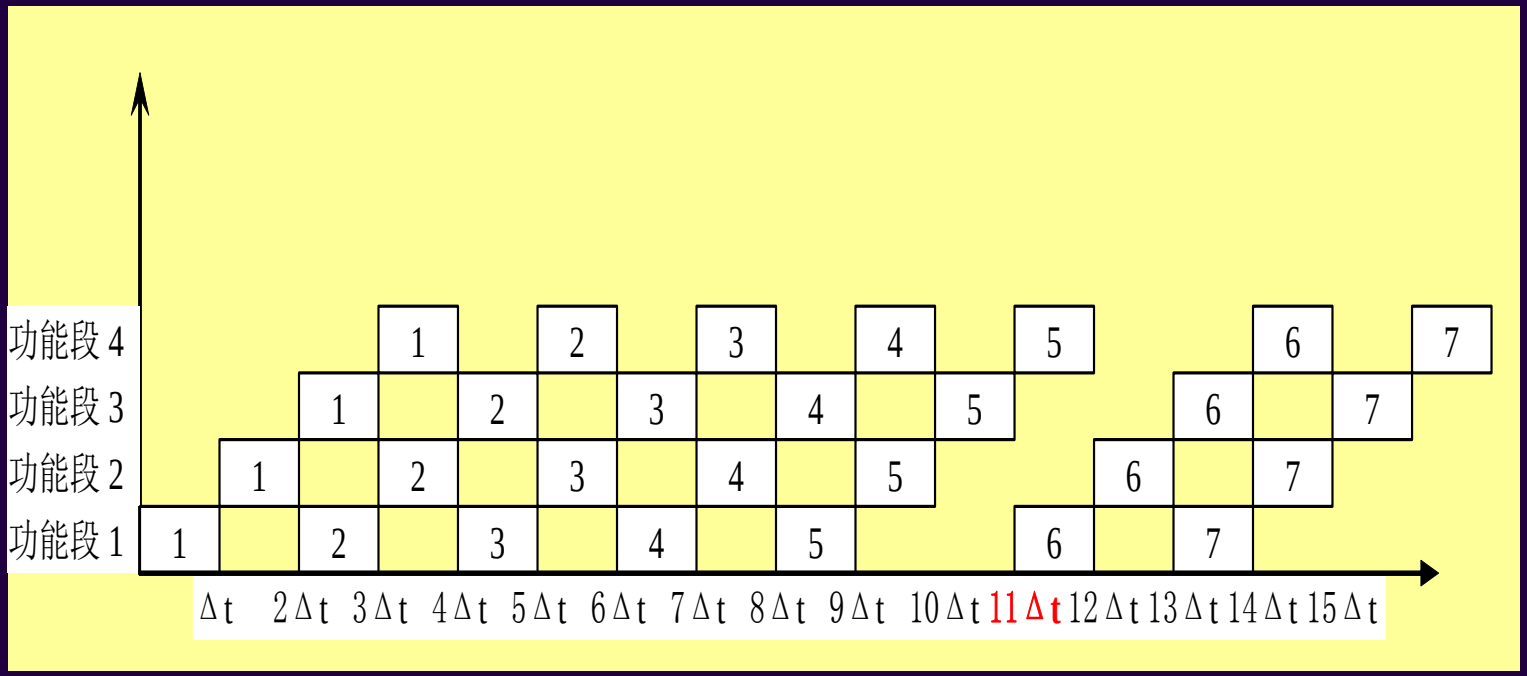
则 TP=（6+5）/(11+8)△t=11/19△t

E=(6*6+5*4)△t/(19*8△t)=36.8%

Sp=(6*6+5*4)△t/19△t=56/19=2.94

作业5.7 p343 一条线性流水线有4个功能段组成，每个功能段的延迟时间都相等，都为Δt。开始5个Δt，每间隔一个Δt向流水线输入一个任务，然后停顿2个Δt，如此重复。求流水线的实际吞吐率、加速比和效率。

[解答]流水线的时空图如下：



我们可以看出，在（11n+1）Δt的时间内，可以输出5n个结果，如果指令的序列足够长（n→∞），并且指令间不存在相关，那么，吞吐率可以认为满足：

$$Tp = \frac{5n}{(11n+1)\Delta t} = \frac{5}{(11+1/n)\Delta t} = \frac{5}{11\Delta t}(n \to \infty)$$

加速比为：

$$S = \frac{5n \times 4\Delta t}{(11n+1)\Delta t} = \frac{20n}{11n+1} = \frac{20}{11+1/n} = \frac{20}{11}(n \to \infty)$$

从上面的时空图很容易看出，效率为：

$$E = \frac{20n\Delta t}{4 \times (11n+1)\Delta t} = \frac{5}{11+1/n} = \frac{5}{11}(n \to \infty)$$

作业 5.8 用一条5个功能段的浮点加法器流水线计算

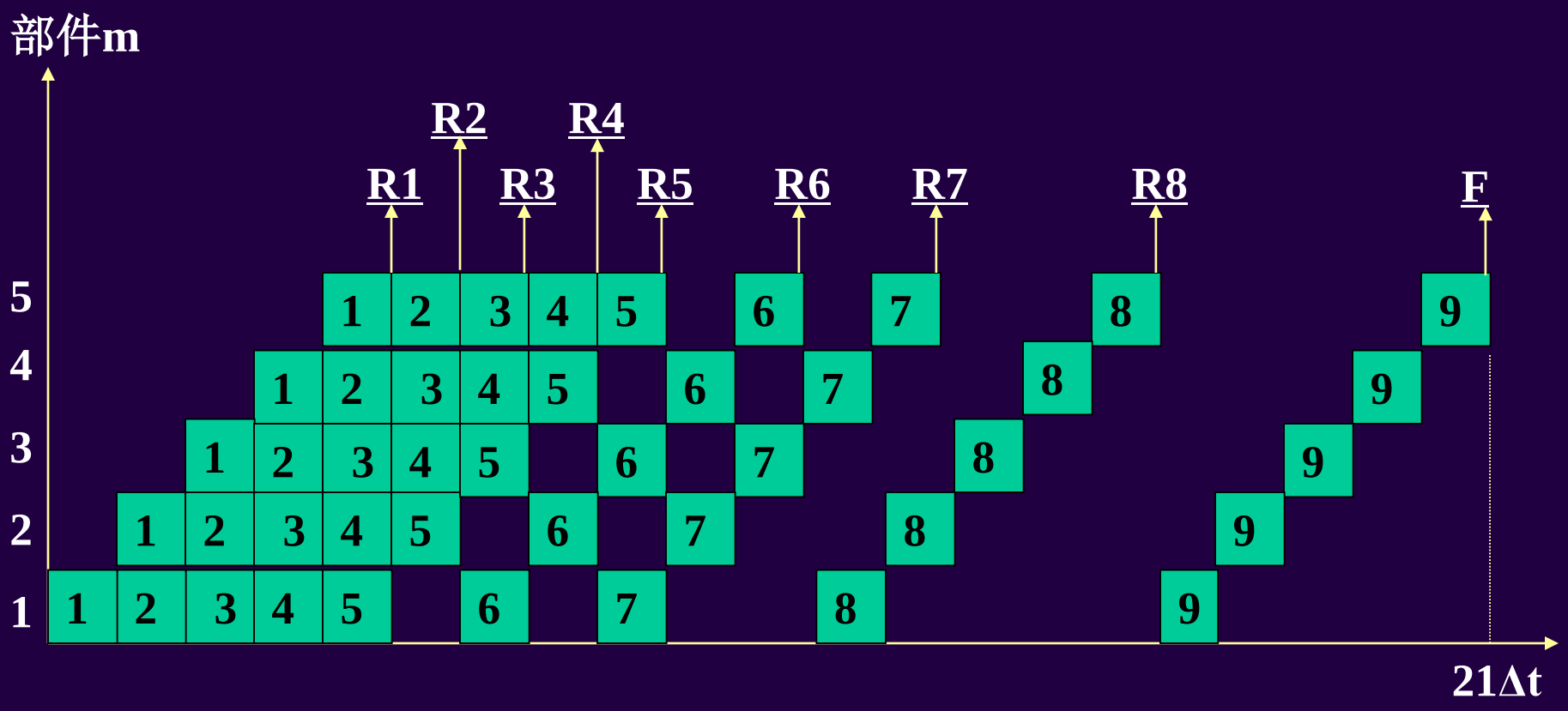
$$F = \sum_{i=1}^{10} A_i$$

每个功能段的延迟时间均相等，流水线的输出端和输入端之间有直接数据通路，而且设置有足够的缓冲寄存器。要求用尽可能短的时间完成计算，画出流水线时空图，并计算流水线的实际吞吐率、加速比和效率。

[解答]首先需要考虑的是，10个数的的和最少需要做几次加法。我们可以发现，加法的次数是不能减少的：9次；于是我们要尽可能快的完成任务，就只有考虑如何让流水线尽可能充满，这需要消除前后指令之间的相关。由于加法满足交换率和结合率，我们可以调整运算次序如以下的指令序列，我们把中间结果寄存器称为R，源操作数寄存器称为A，最后结果寄存器称为F，并假设源操作数已经在寄存器中，则指令如下：

- I1：　　R1←A1+A2
- I2：　　R2←A3+A4
- I3：　　R3←A5+A6
- I4：　　R4←A7+A8
- I5：　　R5←A9+A10
- I6：　　R6←R1+R2
- I7：　　R7←R3+R4
- I8：　　R8←R5+R6
- I9：　　F←R7+R8

这并不是唯一可能的计算方法。假设功能段的延迟为Δt。时空图如下，图中的数字是指令号。



R1=A1+A2

R2=A3+A4

R3=A5+A6

R4=A7+A8

R5=A9+A10

R6=R1+R2

R7=R3+R4

R8=R5+R6

F=R7+R8

整个计算过程需要21Δt，所以吞吐率为：

$$Tp = \frac{9}{21\Delta t}$$

加速比为：

$$S = \frac{9 \times 5\Delta t}{21\Delta t} = \frac{45}{21} = 2.14$$

效率为：

$$E = \frac{9 \times 5\Delta t}{5 \times 21\Delta t} \approx 42.86\%$$

作业 5.9（类似题5.8）一条线性静态多功能流水线由6个功能段组成，加法操作使用其中的1、2、3、6功能段，乘法操作使用其中的1、4、5、6功能段，每个功能段的延迟时间均相等。流水线的输入端与输出端之间有直接数据通路，而且设置有足够的缓冲寄存器。现在用这条流水线计算：

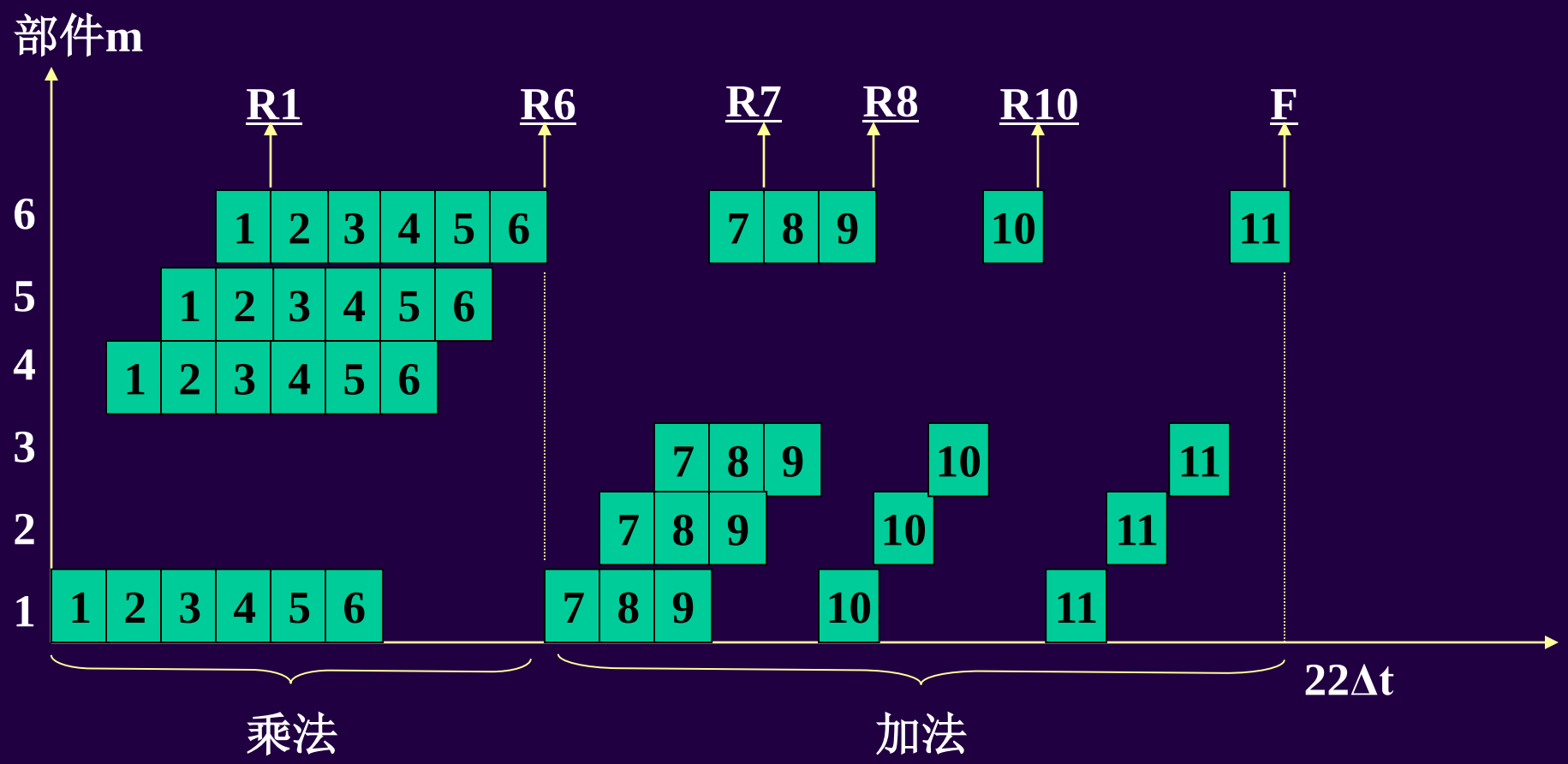
$$F = \sum_{i=1}^{6} (A_i \times B_i)$$

画出流水线时空图，并计算流水线的实际吞吐率、加速比和效率。

解：为了取得较高的速度，我们需要一次将乘法作完，设源操作数存放在寄存器A、B中，中间结果存放在寄存器R中，最后结果存放在寄存器F中，则执行的指令序列如下所示：

I1：　　R1←A1*B1
I2：　　R2←A2*B2
I3：　　R3←A3*B3
I4：　　R4←A4*B4
I5：　　R5←A5*B5
I6：　　R6←A6*B6
I7：　　R7←R1+R2
I8：　　R8←R3+R4
I9：　　R9←R5+R6
I10：　　R10←R7+R8
I11：　　F←R9+R10

这并不是唯一可能的计算方法。假设功能段的延迟为Δt。时空图（不完全）如下，图中的数字是指令号。



R1=A1*B1

R2=A2*B2 → R7=R1+R2

R3=A3*B3

R4=A4*B4 → R8=R3+R4 → R10=R7+R8

R5=A5*B5

R6=A6*B6 → R9=R5+R6 → F=R9+R10

整个计算过程需要22Δt，所以吞吐率为：

$$Tp = \frac{9}{22\Delta t}$$

加速比为：

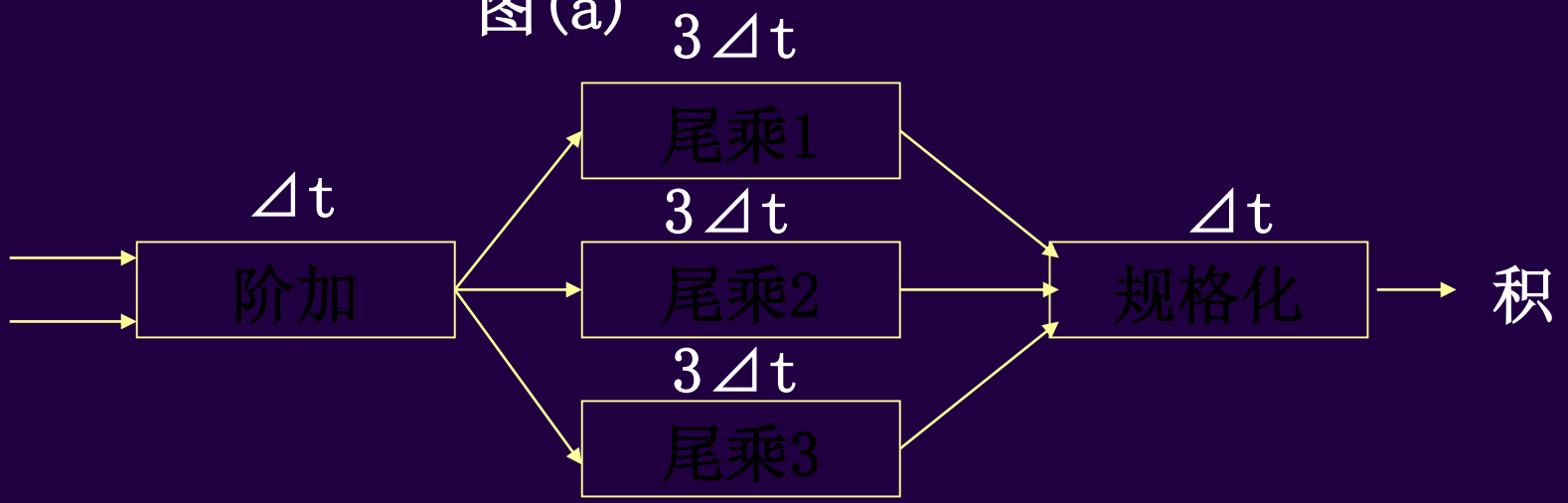
$$S = \frac{(6+5)\times 4\Delta t}{22\Delta t} = \frac{44}{22} = 2$$

效率为：

$$E = \frac{(6+5)\times 4\Delta t}{6\times 22\Delta t} \approx 33.3\%$$

例5.4 有一个浮点乘流水线如下图(a)所示，其乘积可直接返回输入端或暂存于相应缓冲寄存器中，画出实现A*B*C*D的时空图以及输入端的变化，并求出该流水线的吞吐率和效率；当流水线改为下图(b)形式实现同一计算时，求该流水线的效率及吞吐率。

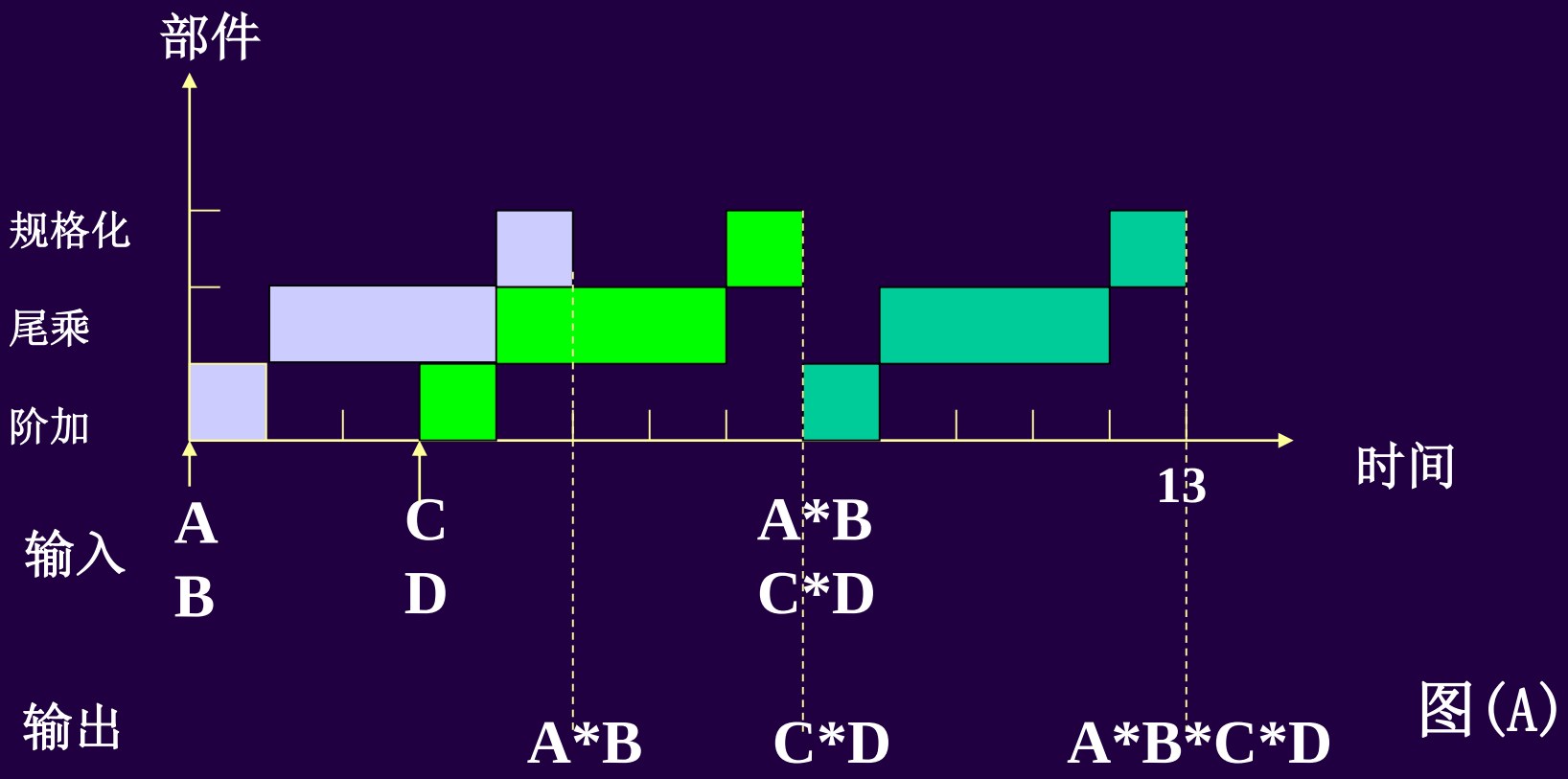
图(a)



图(b)

[分析]为了减少运算过程中的操作数相关，A*B*C*D应改为采用((A*B)*(C*D)) 的算法步骤进行运算。

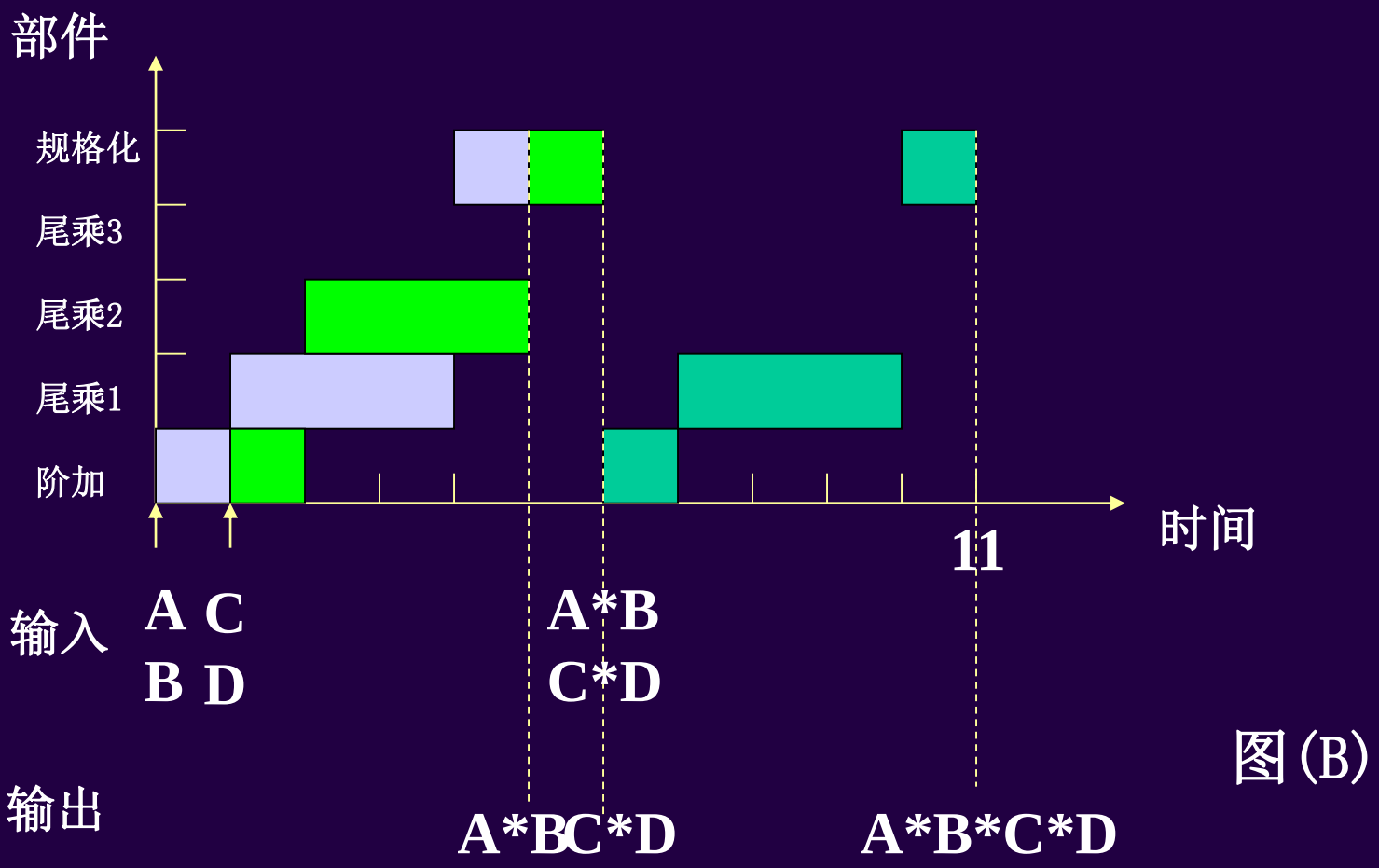
[解]按图(a)组织，实现A*B*C*D的时空关系如下图(A)所示。



图(A)

吞吐率TP=3/(13⊿t)

效率E=(3×5⊿t)/(3×13⊿t)=5/13=38.5%



图(B)

流水线按图(b)组织时，实现A*B*C*D的时空关系如图(B)

吞吐率TP=3/(11⊿t)

效率E =(3×5⊿t)/(5×11⊿t)=3/11=27.3%

在CRAY-1机上，设向量长度均为64，所有浮点功能部件的执行时间分别为：相加需6拍，相加需7拍，求倒数近似值需14拍，从存储器读数据需6拍，打入寄存器机启动功能部件各需1拍，问下列个指令组，组内的哪些指令可以链接？，哪些指令不可以链接？不能链接的原因是什么？并分别计算各指令组全部完成所需的拍数。

（1）
$V_0 \leftarrow$存储器
$V_1 \leftarrow V_2 + V_3$
$V_4 \leftarrow V_5 \times V_6$

（2）
$V_2 \leftarrow V_0 \times V_1$
$V_3 \leftarrow$存储器
$V_4 \leftarrow V_2 + V_3$

（3）
$V_0 \leftarrow$存储器
$V_2 \leftarrow V_0 \times V_1$
$V_3 \leftarrow V_2 + V_0$
$V_5 \leftarrow V_3 + V_4$

（4）
$V_2 \leftarrow$存储器
$V_1 \leftarrow 1/V_0$
$V_3 \leftarrow V_1 \times V_2$
$V_5 \leftarrow V_3 + V_4$

解：

组（1）三条指令可并行执行。

T=1+7+1+64-1=72(拍)。

组（2）前二条指令可并行执行，前两条与第三条指令可链接执行。

T=（1+7+1+1+6+1）+63=80（拍）。

组（3） 前3条指令可链接执行，后一条指令只能串行（加法部件冲突）

T=（8+9+8+63）+8+63=159（拍）。

组（4）前二条指令可并行执行，后两条可链接，这样前两条与后两条指令可链接执行。

T=（14）+（9）+(8)+63=94（拍）。

计算机系统结构模拟试题

模拟试题一

一.简答题(40分)

1.什么是透明性概念？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；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。

2.当浮点数尾数基值减小时，对机器数的表示会产生哪些方面的影响(至少列举5点)？

3.试述CISC的改进途径及其目的。

4.描述总线控制方式中采用集中式串行链接时，总线的分配过程。

5.试区别中断响应次序与中断处理次序。

6.在页式虚拟存贮器中，什么叫页面失效？什么叫页面争用？什么时候两者同时发生？什么时候两者不会同时发生？

7.描述Cache存贮器的组相联映象及其变换。

8.段式管理的思想是什么？

二.问答题(30分)

1.浮点数尾数下移处理有哪些方法？各种方法的思想、最大误差、误差的分布及正负情况是什么？

2.三种通道的极限流量与实际最大流量是怎么计算的？设计时应满足什么式子？

3.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，有下来建议：

(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。

三.计算题(30分)

1.某机器有10条指令，使用频度分别为：

0.01，0.15，0.12，0.07，0.08，

0.13，0.15，0.03，0.17，0.09。

(1)计算用等长操作码编码的平均码长；

(2)构造Huffman树；

(3)写出Huffman的一种编码，并计算其平均码长；

(4)只有二种码长，求平均码长最短的扩展操作码编码及其平均码长。

2.Cache分$S_1=2^4$块，主存分$S_2=2^8$块，让主存第i块映象装入到Cache中的第j块，$j=i \bmod S_1$。

(1)这是什么映象规则？

(2)画出主存地址到Cache地址变换过程的示意图，图中应标示出主存与Cache地址各字段位数及对应关系；指出用于地址映象变换的辅助表存贮器的宽度和单元数；并对变换过程作简单的文字说明。

模拟试题一参考答案

一.简答题

1.【解答】客观存在的事物或属性，从某个角度看去却看不到，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；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。

2.【解答】(1)数的可表示范围缩小；

(2)可表示数的总个数减少；

(3)数在数轴上的分布变密；

(4)机器数的精度提高；

(5)运算过程中的精度损失增大；

(6)运算速度有所降低。

3.【解答】(1)面向目标程序的优化实现来改进

目的：提高各机器语言目标程序的效率，减少存贮空间，提高运行速度，容易实现。

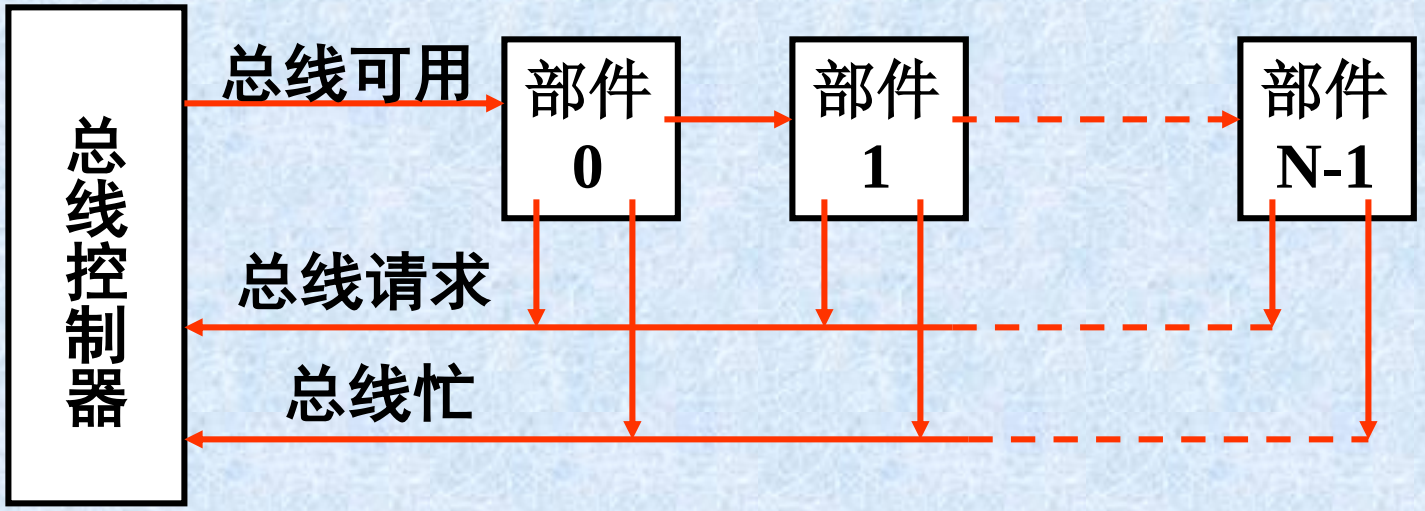
(2)面向高级语言的优化来实现

目的：尽可能缩短高级语言和机器语言的语义差距，以利于支持高级语言的编译系统，缩短编译程序的长度和编译所需的时间。

(3)面向操作系统的优化来实现

目的：缩短OS与系统结构的语义差距，减少运行OS所需的辅助时间，节省OS软件所占用的存贮空间。

4.【解答】集中式串行链接总线的原来如下图：



集中式串行链接总线控制原理图

各部件经公共的“总线请求”线向总线控制器发出使用总线的请求。仅当“总线忙”未建立时，控制器在“总线可用”线上发出“总线可用”信号，串行送往各个部件。如果某部件未发“总线请求”信号，就将

“总线可用”信号顺链下去；否则，停止下传，向“总线忙”送出信号，并取出该部件的“总线请求”，此次总线分配结束，该部件获得总线使用权。等到该部件数据传送完毕后，由部件去除“总线忙”信号，“总线可用”信号就随之去除。如果系统仍有总线请求，就开始新的总线分配过程。

5.【解答】(1)中断响应次序是靠中断响应的硬件排队器事先固定好的。它总是对进入了中断响应排队器的中断级请求按由高到低的次序响应其中的一个高优先级的中断级请求，除非某些中断级请求未进入中断响应排队器排队。

(2)为了能动态地调节中断处理程序实际执行的次序，即中断处理次序，在中断级请求源与中断响应排队器的入口端之间又加设了一个中断级屏蔽字寄存器和\相应的控制门电路硬件。中断级屏蔽字寄存器中的每一个中断级屏蔽位可以控制让相应等级的中断请求能否进入中断响应排队器去参加排队。只要能进入中断响应排队器的中断请求，总是让其中断级别相对高的优先得到响应。

操作系统可以通过修改各中断级处理程序的中断级屏蔽位的状况，来使中断处理(完)的次序符合我们所希望的次序。

6.【解答】要访问的虚页不在实际主存时，就会发生页面失效。当页面调入主存，主存中的页面位置全部已被其它虚页占用时，就会发生页面争用。当分配给程序的内存已被全部占用之后，只要发生页面失效，就一定会发生页面争用。反之，发生页面失效，并不会发生页面争用。

7.将Cache空间和主存空间分组，每组S块($S=2^s$)。Cache一共2^{ncb}个块，分成Q组($Q=2^q$)，整个Cache是一区。主存分成与Cache一样大小的2^{nd}个区，其地址按区号、组号、组内块号、块内地址分成对应的4个字段。主存地址的组号、组内块号分别用q、s' 字段表示，它们的宽度和位置与Cache地址的q、s是一致的。

组相联映象指的是各组之间直接映象，但组内各块间则是全相联映象。

8.【解答】根据程序的模块性，把一个复杂的大程序分解成多个逻辑上相对独立的模块。模块可以是主程序、子程序或过程，也可以是表格、数组以及树、向量等某类数据元素的集合。其大小可以互不相同，甚至事先无法确定。每个模块都是一个单独的程序段，都以该段的起点为0相对编址。调入主存时，系统按照基址(该段程序在主存中的起始地址)和每个单元在段内的相对位移组合来形成各单元在主存的实际地址。

二.问答题

1.【解答】(1)截断法

方法:将尾数超出机器字部分简单截去

最大误差：整数接近1，二进制分数接近2^{-m}

正负：正数　产生负误差

分布：间隔相同，均匀分布

(2)舍入法

方法：在规定字长外增设一位附加位，存放溢出部分高位，处理时该位加1。

最大误差：整数二进制为0.5，分数为$2^{-(m+1)}$

正负：有正有负

分布：统计平均误差接近零，稍偏正，无法调节

(3)恒置“1”法

方法:在规定字长之最低位恒置成“1”状态

最大误差：整数二进制为1，分数为2^{-m}

正负：有正有负。

分布：统计平均误差接近零，稍偏正，无法调节

(4)查表舍入法

方法:基于存贮逻辑思想，用ROM或PLA存放下溢处理表。当k-1位全位“1”时，以截断法处理；其它以舍入法处理。

最大误差： 最大为1

正负：有正有负

分布：不均匀，平均误差为零，可调节

2.【解答】(1)极限流量

a)字节多路通道:每选一台设备传送一个字节

$$f_{max.byte}=1/(T_S+T_D)$$

b)数组多路通道:每选一条设备传送K个字节

$$f_{max.block}=k/(T_S+kT_D)=1/(T_S/k+T_D)$$

c)选择通道:每选一台设备就把N个字节传送完

$$f_{max.select}=N/(T_S+NT_D)=1/(T_S/N+T_D)$$

(2)实际最大流量

a)字节多路通道:

$$f_{byte.j}=\sum_{i=1}^{p_j} f_{i.j}$$

b)数组多路通道:

$$f_{block.j}=\max_{i=1}^{p_j} f_{i.j}$$

c)选择通道:

$$f_{select.j}=\max_{i=1}^{p_j} f_{i.j}$$

(3)满足式子

$$f_{max.byte.j} >= f_{byte.j}$$

$$f_{max.block.j} >= f_{block.j}$$

$$f_{max.select.j} >= f_{select.j}$$

3.【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比较少，命中率H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

显著增加，要访问的程序页面在主存中的机会大大增加，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算法不是堆栈型替换算法，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，主存命中率可能会有所增加，也可能降低和不变。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H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，将会使主存命中率继续有所下降。

(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

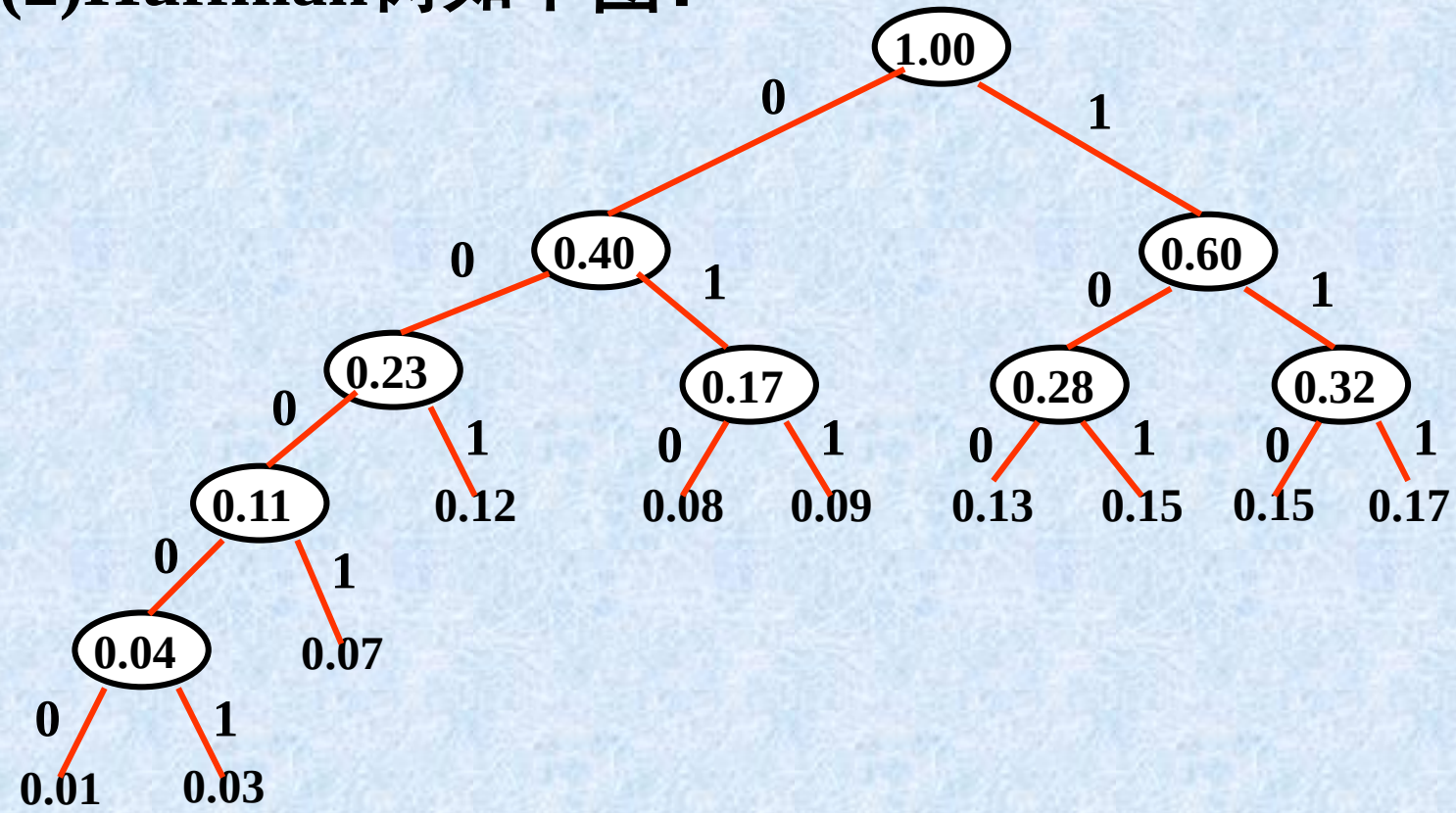
存的容量(页数)，一般会使主存命中率有较大的提高。因为LRU替换算法比FIFO替换算法能更好地体现出程序工作的局部性，又由于原先主存的命中率太低，现在增大主存容量，一般会使主存命中率上升。如果主存容量增加量大些，主存命中率H将会显著上升。

(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。

三.计算题

1.【解答】(1)4位

(2)Huffman树如下图：



(3)、(4)结果如下：

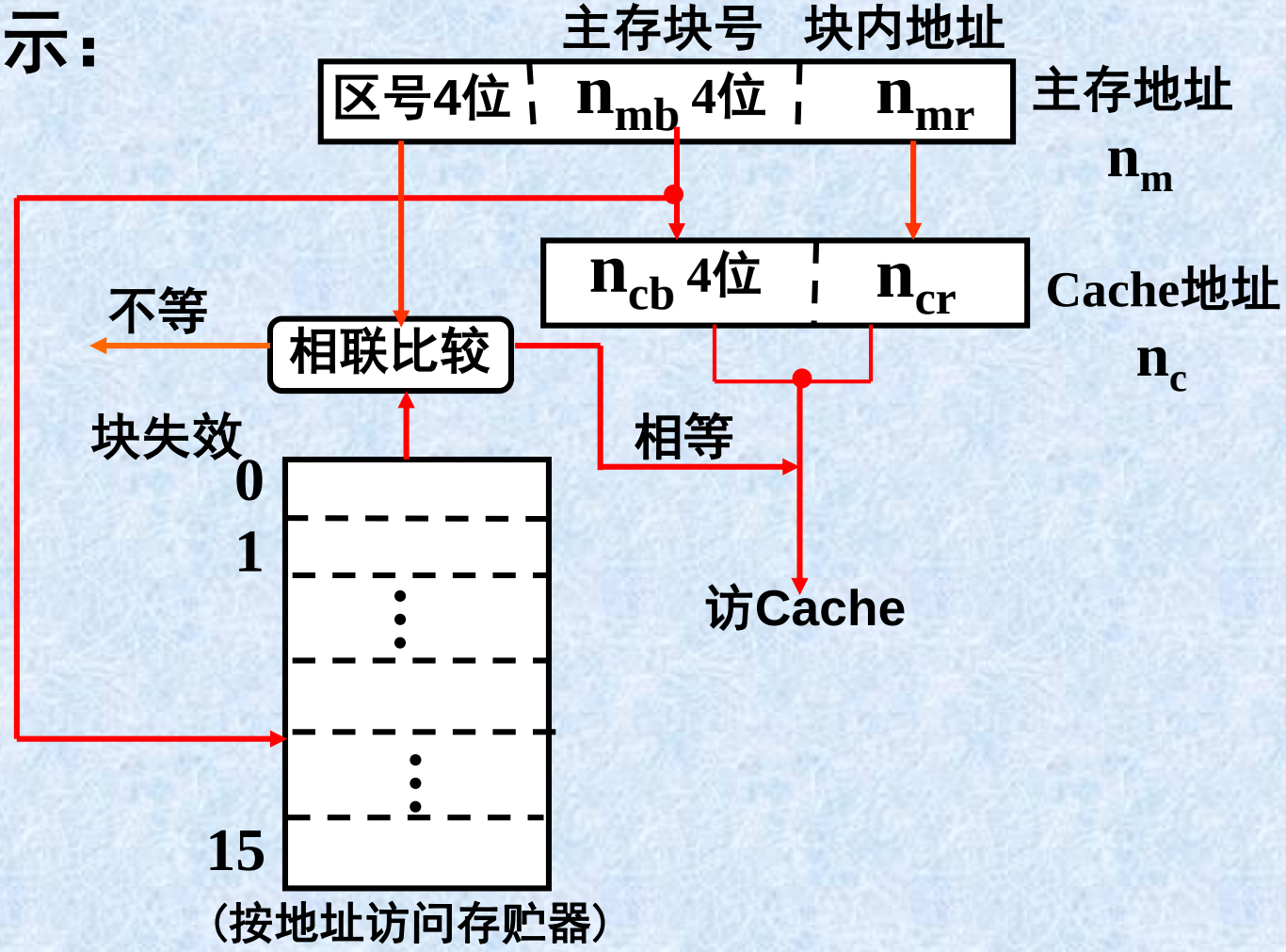
使用频度	Huffman编码(不唯一)	扩展编码(不唯一)
0.01	00 00 0	00 00
0.03	00 00 1	00 01
0.07	00 01	00 10
0.08	01 0	00 11
0.09	01 1	01 0
0.12	00 1	01 1
0.13	10 0	10 0
0.15	10 1	10 1
0.15	11 0	11 0
0.17	11 1	11 1

Huffman编码平均码长＝$\sum p_i l_i$＝3.15位

扩展编码平均码长＝$\sum p_i l_i$＝3.19位

2.【解答】(1)直接映象

(2)主存地址到Cache地址变换过程的示意图如下图所示：



辅助映象表存贮器用按地址访问的存贮器构成，共16个单元，每个单元4位宽。

变换过程：主存地址中截取对应Cache地址部分的字段来访Cache，同时用区内(Cache)块号访问辅助映象表存贮器，将其内容读出，与主存区号字段进行比较。若比较相等，让访Cache操作继续进行下去，表示Cache命中；否则，表示访Cache失效，由主存中调块。

模拟试题二

一.简答题(40分)

1.分别解释结构、组成与实现。

2.提高计算机系统并行性的技术途径有哪三个？简要解释并各举一系统类型的例子。

3.设计RISC机器的基本技术是什么？

4.简要说明构造Huffman树的过程。

5.通道分为哪3种类型？各适合连接什么类型的设备？满负荷时，设备对通道要求的实际流量与所连的设备有什么关系？

6.列举定时查询中确定部件优先级次序的方法。

7.CPU写Cache时，会发生Cache与主存的对应副本内容不一致的现象，解决这个问题有哪些方法？各需要增加什么开销？

8.LRU块替换算法的硬件实现有哪2种方法？各自的思路是什么？

二.问答题(30分)

1.什么叫“程序的动态再定位”？

2.中断分类的依据是什么？目的是什么？具体分为哪几类？各自是如何定义的？

3.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？

三.计算题(30分)

1.机器有5级中断，中断响应次序为1 2 3 4 5，现要求中断处理次序为2 3 1 5 4。

(1)设计各级中断处理程序的中断级屏蔽位的状态，令“0”表示开放，“1”表示屏蔽。

(2)若在运行用户程序时，同时发生第1、3级中断请求，而在第1级中断未完成时，又发生2、3、4、5级中断，请画出处理机执行程序的全过程示意图(标出交换PSW的时间)。

2.设某虚拟存贮器上运行的程序包含5个页面，其页地址流依次为4，5，3，2，5，1，3，2，5，1，3用LRU替换算法。

(1)用堆栈对该页地址流模拟一次，画出这一模拟过程，并标出实页数为3，4，5时的命中情况。

(2)为获得最高的命中率，应分配给该程序几个实页即可？其可能的最高命中率是多少？

模拟试题二参考答案

一.简答题

1.【解答】(1)系统结构：是对计算机系统中各机器之间界面的划分和定义，以及对各级界面上、下的功能进行分配。

(2)计算机系统结构：是系统结构中的一部分，指层次结构中传统机器级的系统结构，其界面之上的功能包括操作系统级、汇编语言级、高级语言级和应用语言级中所有软件的功能；界面之下的功能包括所有硬件和固件的功能。因此，这个界面实际是软件与硬件或固件的分界面。

(3)计算机组成：是计算机系统结构的逻辑实现，包括机器级内的数据流和控制流的组成以及逻辑实现。

(4)计算机实现：指的是计算机组成的物理实现。

2.【解答】(1)时间重叠：在并行性中引入时间因素，让多个处理过程在时间上错开，轮流重复的使用同一套硬件设备的各个部分，以加快硬件周转而提高速度。例如，流水线处理机。

(2)资源重复：引入空间因素，通过重复设置硬件资源来提高可靠性或性能。例如，双工系统。

(3)资源共享：利用软件的办法让多个用户按一定时间顺序轮流使用同一套资源，以提高其利用率，这样可以提高整个系统的性能。例如，多道程序分时系统，多处理机，分布处理系统，计算机网络。

3.【解答】(1)遵循RISC机器一般原则设计的技术。

(2)在逻辑上采用硬联实现和微程序固件实现相结合的技术。

(3)在CPU中设置数量较大的寄存器组，并采用重叠寄存器窗口的技术。

(4)指令的执行采用流水和延迟转移技术。

(5)采用认真设计和优化编译系统设计的技术。

4.【解答】(1)将指令的使用频度由小到大排序

(2)每次选最小两个频度结合一个新节点

(3)再按频度大小插入余下未结合的频度值中

(4)如此重复直至全部结合完毕成根节点“1”

(5)沿两个分支，分别用“0”或“1”来表示

这样，从根节点开始，沿线到达个频度指令的代码序列就是该指令的哈夫曼编码。

5.【解答】通道分为字节多路通道、数组多路通道以及选择通道3种。

(1)字节多路通道适合连接大量低速的字符设备。满负荷时，设备对通道要求的实际流量应是所连各

设备的流量之和。

(2)数组多路通道适合于连接高速的设备。满负荷时，设备对通道要求的实际流量应是所连各个设备中，流量最大的那个。

(3)选择通道适合于连接中、高速的高优先级的设备。满负荷时，设备对通道要求的实际流量应是所连各个设备中，流量最大的那个。

6.【解答】(1)总线分配前计数器清“0”,从“0”开始查
询，优先级排序类似串行链接。

(2)总线分配前不清“0”，从中止点继续查询，是循环优先级，部件使用总线机会均等。

(3)总线分配前将计数器设置初值，可以指定某个部件为最高优先级。

(4)总线分配前将部件号重新设置，可以为各部件指定任意希望的优先级。

7.【解答】(1) 写回法：在CPU执行写操作时，只是把信息写入Cache，仅当需要被替换时，才将已经被写入过的Cache块先送回主存，然后再调入新块进行替换。这种方法要求对每个Cache块增加一个修改位的资源开销。

(2)写直达法：也称为存直达法，它利用Cache——主存存贮层次在处理机和主存之间的直接通路，每当处理机写入Cache的同时，也通过此通路直接写入主存。这种方法要增加写主存的时间开销。

8.【解答】(1)堆栈法：栈顶恒存放近期最久访问过的页的页号，而栈底恒存放近期最久没有访问过的页的页号，即准备被替换掉的页的页号。按此思想组成一个硬件堆栈。

(2)比较对法：让各个块成对组合，用一个触发器的状态来表示该比较对内两块访问的远近次序，再经门电路就可以找到LRU块。

二.问答题

1.【解答】程序在实际主存空间中的位置可以动态移动的定位技术。一种做法是，在硬件上设置基址寄存器和地址加法器，即在程序装入主存时，只把程序装在主存中的起始地址(基址)装入基址寄存器中，对指令各地址字段不作修改；在程序运行时，由逻辑地址根据需要，加上基址寄存器中的基址来形成访存有效地址。另一种做法是，设置逻辑地址到主存物理有效地址的映象表硬件，即程序装入主存时，在映象表中建立起逻辑地址与主存物理地址

的映象关系；程序执行时，由逻辑地址查映象表来获得访主存的物理有效地址。这样，只需修改基址寄存器中的基址或地址映象表的内容，就可以使程序在主存中动态改变所存贮的位置。

2.【解答】(1)中断分类的依据是：把中断源性质相近、中断处理过程类似的归为一类。

(2)中断分类的目的是：为了减少中断处理程序的入口，每一类给一个中断服务程序总入口，可以减少中断服务程序入口地址形成的硬件数量。

(3)中断分为中断(Interrupt)和异常(Exception)两类，而异常又细分为自陷(Trap)、故障(Fault)和失

败(Abort)三类。具体定义如下：

中断：专指与当前进程运行无关的请求暂停的事件，如机器故障中断请求、外设中断请求、定时中断请求等。中断可以被屏蔽，暂时保存在中断寄存器，屏蔽解除后继续得到响应和处理。

异常：专指由现行指令引起的暂停事件，如页面失效、溢出等，一般不能屏蔽，立即得到响应和处理。

自陷：发生在引起异常的指令执行的末尾，处理后返回原先正常程序的下一条指令继续执行。

故障：发生在执行指令的过程中，处理后返回原先发生故障的那条指令出重复执行。

失败：也发生在指令执行过程中，需强制干预或系统复位才可以使指令再正确执行下去。

3.【分析】Cache存贮器的等效访问时间为：

$$t_a = H_c\, t_c + (1 - H_c) t_m$$

等效访问速度不高，就是t_a太长。要想缩短t_a，一是要使命中率H_c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物理存贮器芯片，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。当命中率H_c已经很高时，内部的查映象表与访Cache由不流水改为流水，会对t_a有明显的改进，可以缩短接近一半的时间。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命中率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的限制不可能太大，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稍微有所增大，如果H_c已经很高时，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

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的块数已经较多， H_c已经很高了，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，块的大小保持不变，从而使组内的块数有了增加，它会使块冲突的概率下降，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，从而可以提高等效访问速度。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，但要看目前的H_c水平而定。如果H_c已经很高，则增大Cache块的大小对t_a的改进也就不明显了。

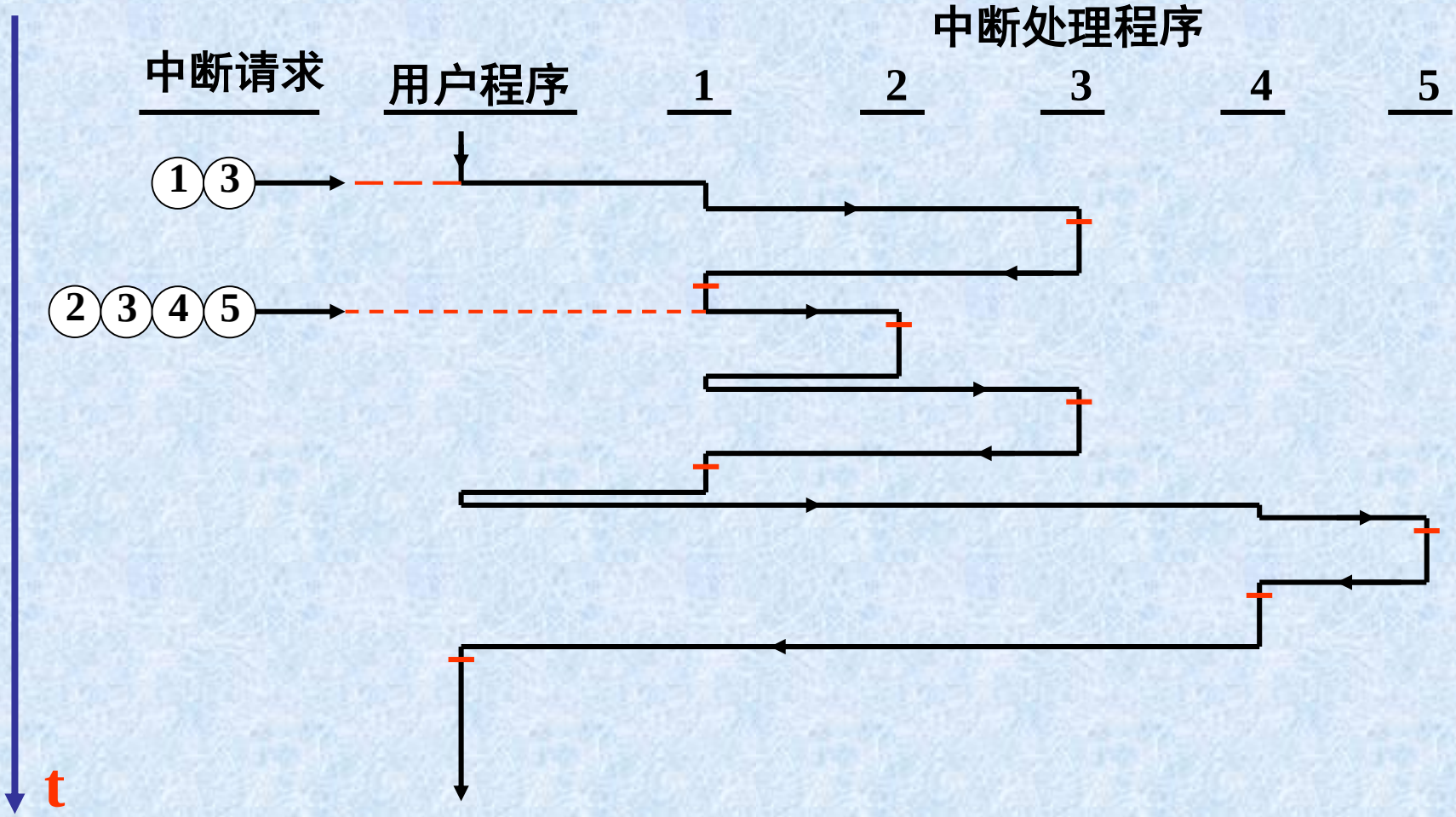
(5)提高Cache本身器件的访问速度，即减小t_c，只有当命中率H_c已很高时，才会显著缩短t_a。如果命中率H_c较低时，对减小t_a的作用就不明显了。

三.应用题

1.【解答】(1)各级中断处理程序的中断级屏蔽位状态如下表：

中断处理程序级别	中断级屏蔽位				
	1级	2级	3级	4级	5级
第1级	1	0	0	1	1
第2级	1	1	1	1	1
第3级	1	0	1	1	1
第4级	0	0	0	1	0
第5级	0	0	0	1	1

(2)处理机执行程序的全过程如下图：



2.【分析】由于LRU替换算法是堆栈型替换算法，因而随着分配给该道程序的实页数增加，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后，其命中率就不会再提高了。如果要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命中率，找出大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】(1)模拟过程及命中情况如下表：

时间t		1	2	3	4	5	6	7	8	9	10	11
页地址流		4	5	3	2	5	1	3	2	5	1	3
堆栈内容	S(1)	4	5	3	2	5	1	3	2	5	1	3
	S(2)		4	5	3	2	5	1	3	2	5	1
	S(3)			4	5	3	2	5	1	3	2	5
	S(4)				4	4	3	2	5	1	3	2
	S(5)						4	4	4	4	4	4
实页数	n=3					H						
	n=4					H		H	H	H	H	H
	n≥5					H		H	H	H	H	H

(2)为获得最高命中率，应分配给该程序4个实页即可，最高命中率为H＝6/11。