

计算机组成原理 突击课

6.5小时突击

课程讲义

江苏博事达律师事务所

J I A N G S U B O O M S T A R L A W O F F I C E

中国 南京 奥体大街 68 号国际研发总部园 4A 栋 17 楼
17F 4A Building NO.68 Aoti Street, Nanjing, China
电话(Tel): (86)-25-82226685

邮编: 210019
P.C: 210000
传真(Fax): (86)-25-82226696

律 师 声 明

江苏博事达律师事务所接受蜂考品牌公司的委托,发表以下律师声明:

“蜂考系列课程”(含视频、讲义、音频等)内容均为蜂考原创,蜂考品牌公司对此依法享有著作权,任何单位或个人不得侵犯。

蜂考品牌公司为维护创作作品的合法权益,已与江苏博事达律师事务所开展长期法律顾问合作,凡侵犯课程版权等知识产权的,蜂考品牌公司将授权江苏博事达律师事务所依据国家法律法规提起民事诉讼。对严重的侵权盗版分子将报送公安部门采取刑事手段予以严厉打击。

感谢大家对蜂考品牌的长期支持,愿与各位携手共同维护知识产权保护。遵守国家法律法规,从自身做起,抵制盗版!

特此声明!

江苏博事达律师事务所
二〇二一年七月十四日



目录

| | |
|------------------------------|-----|
| 课时一 计算机概述..... | 1 |
| 1. 计算机重要的硬件部件..... | 1 |
| 2. 计算机系统的层次结构..... | 3 |
| 3. 计算机的性能指标..... | 4 |
| 课时一 练习题..... | 7 |
| 课时二 数据的表示..... | 8 |
| 1. 数制与编码..... | 8 |
| 2. 定点数的表示..... | 11 |
| 3. 浮点数的表示..... | 14 |
| 课时二 练习题..... | 18 |
| 课时三 数据的运算..... | 19 |
| 1. 定点数的运算..... | 19 |
| 2. 浮点数的计算..... | 26 |
| 3. 加法器与 <i>ALU</i> | 28 |
| 课时三 练习题..... | 33 |
| 课时四 主存储器..... | 35 |
| 1. 存储器概述..... | 35 |
| 2. 主存储器与 <i>CPU</i> 的连接..... | 41 |
| 课时四 练习题..... | 46 |
| 课时五 辅存与 <i>Cache</i> | 47 |
| 1. 磁盘存储器..... | 47 |
| 2. 高速缓冲存储器..... | 49 |
| 3. 虚拟存储器..... | 59 |
| 课时五 练习题..... | 62 |
| 课时六 指令系统..... | 64 |
| 1. 概述..... | 64 |
| 2. 指令格式..... | 67 |
| 3. 寻址方式..... | 69 |
| 课时六 练习题..... | 77 |
| 课时七 数据通路..... | 79 |
| 1. <i>CPU</i> 概述..... | 79 |
| 2. 数据通路..... | 81 |
| 3. 指令流水线..... | 88 |
| 课时七 练习题..... | 94 |
| 课时八 控制器..... | 96 |
| 1. 概括与设计..... | 96 |
| 2. 异常和中断..... | 104 |
| 课时八 练习题..... | 107 |
| 课时九 总线..... | 109 |

| | |
|------------------|-----|
| 1. 总线概括 | 109 |
| 2. 总线仲裁机制 | 114 |
| 3. 总线事务和定时 | 117 |
| 课时九 练习题 | 120 |
| 课时十 输入输出系统 | 122 |
| 1. 基本概念 | 122 |
| 2. I/O 接口 | 123 |
| 3. I/O 方式 | 127 |
| 课时十 练习题 | 137 |

课时一 计算机概述

| 考点 | 重要程度 | 占分 | 题型 |
|---------------|------|-----|-------|
| 1. 计算机重要的硬件部件 | ★★★★ | 2~6 | 选择、填空 |
| 2. 计算机系统的层次结构 | ★★★ | 2~4 | 选择、填空 |
| 3. 计算机的性能指标 | ★★★★ | 2~4 | 选择、填空 |

1. 计算机重要的硬件部件

1) 冯·诺依曼机的特点

- (1) 计算机由五大部件组成：运算器、控制器、存储器、输入设备、输出设备。
- (2) 指令和数据以同等地位存于存储器，可按地址寻访。
- (3) 指令和数据用二进制表示。
- (4) 指令由操作码和地址码组成。
- (5) 存储程序。
- (6) 以运算器为中心（现代计算机以存储器为中心）。

题 1. 冯·诺依曼结构计算机中数据采用二进制编码表示，其主要原因是（ ）。

(1) 二进制的运算规则简单；

(2) 制造两个稳态的物理器件比较容易；

(3) 便于用逻辑门电路实现算术运算。

A. 仅 (1) (2) B. 仅 (1) (3) C. 仅 (2) (3) D. (1) (2) (3)

答案：D

解析：对于 (3)，二进制的“0”和“1”刚好对应逻辑中的“真”和“假”。

2) 主存储器

(1) 主存储器的组成

- ① 存储体：数据在存储体内按地址存储。
- ② MAR（存储地址寄存器）：MAR 位数反映存储单元的个数。
- ③ MDR（存储数据寄存器）：MDR 位数等于存储字长。

(2) 相关概念

- ① 存储元：即存储二进制的电子元件，每个存储元可存储 1 bit。
- ② 存储单元：每个存储单元存放一串二进制代码。

- ③存储字：存储单位中二进制代码的组合。
- ④存储字长：存储单元中二进制代码的位数。
- ⑤1字节(Byte) = 8 bit, 1 B = 1个字节, 1 b = 1位二进制。

题 2. 下面关于半导体存储器的叙述中，错误的是（ ）。

- A. 存储器的核心部分是存储体，由若干个存储单元构成
- B. 一个存储单元有一个编号，就是存储单元的地址
- C. 存储单元由若干存放“0”和“1”的存储元件构成
- D. 同一个存储器中，每个存储单元的宽度可以不同

答案：D

解析：在同一个存储器中，每个存储单元的宽度，即存储字长是固定的。

3) 运算器

(1) 运算器的组成

- ①ACC（累加器）。
- ②MQ（乘商寄存器）。
- ③X（通用的操作数寄存器）。
- ④ALU（算术逻辑单元）

(2) 相关概念

| | 加 | 减 | 乘 | 除 |
|-----|-------|-------|---------|--------|
| ACC | 被加数、和 | 被减数、差 | 乘积高位 | 被除数、余数 |
| MQ | | | 乘数、乘积低位 | 商 |
| X | 加数 | 减数 | 被乘数 | 除数 |

题 3. 运算器的主要功能是进行（ ）。

- A. 逻辑运算
- B. 算术运算
- C. 逻辑运算和算术运算
- D. 初等函数的运算

答案：C

4) 控制器

(1) 控制器的组成

- ① *CU* (控制单元): 分析指令, 给出控制信号。
- ② *IR* (指令寄存器): 存放当前执行的指令。
- ③ *PC* (程序计数器): 存放下一条指令地址, 有自动加1的功能

(2) 硬件工作过程

初始时指令和数据存入主存, *PC* 指向第一条指令, 从主存中取出指令放入 *IR* 中, *PC* 自动加1, *CU* 分析指令并发出控制信号来控制其他部件来执行指令。

题 4. 在 *CPU* 中, 跟踪下一条将执行的指令的地址的寄存器是 ()。

- A. *MAR* B. *MDR* C. *PC* D. *IR*

答案: C

解析: (1) *CPU* = 运算器 + 控制器 (+ 寄存器)。

(2) *PC* (程序计数器) 的作用是存放下一条指令地址, 且有自动加1的功能。

2. 计算机系统的层次结构

1) 层次结构

微程序机器 M_0 (微指令系统) \rightarrow 传统机器 M_1 (用机器语言的机器) \rightarrow 虚拟机器 M_2 (操作系统机器) \rightarrow 虚拟机器 M_3 (汇编语言机器) \rightarrow 虚拟机器 M_4 (高级语言机器)。

2) 相关概念

(1) 高级语言 (*C/C++*、*Java*) $\xrightarrow[\text{(编译器)}]{\text{编译程序}}$ 汇编语言 $\xrightarrow[\text{(汇编器)}]{\text{汇编程序}}$ 机器语言 (二进制代码);

(2) 编译程序: 将高级语言编写的源程序全部语句一次全部翻译成机器语言程序, 而后再执行机器语言程序 (只需翻译一次);

解释程序: 将源程序的一条语句翻译成对应于机器语言的语句, 并立即执行。紧接着再翻译下一句 (每次执行都要翻译)。

题 1. 计算机硬件能够直接执行的是 ()。

- (1) 机器语言程序;
- (2) 汇编语言程序

(3) 硬件描述语言程序

A. 仅 (1) B. 仅 (1) (2) C. 仅 (1) (3) D. (1) (2) (3)

答案: A

解析: (2) (3) 都必须转换为 (1) 后才可直接被计算机执行。

3. 计算机的性能指标

1) 存储器的性能指标

(1) 存储器的容量 = 存储单元个数 \times 存储字长(bit) = 存储单元个数 \times 存储字长(8) (Byte)。

(2) 相关概念

① n 位二进制可表示 2^n 种状态,

例如: 2 位二进制可表示 4 种状态: 00, 01, 10, 11。

② $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, $2^7 = 128$,

$2^8 = 256$, $2^9 = 512$, $2^{10} = 1024$;

③ 在存储容量相关题中: $1K = 2^{10} = 1024$, $1M = 2^{20}$, $1G = 2^{30}$, $1T = 2^{40}$;

在与时间相关 (如传输速率) 题中: $1K = 10^3$, $1M = 10^6$, $1G = 10^9$, $1T = 10^{12}$ 。

题 1. 若计算机字长 16 位, 主存地址空间大小为 64KB, 按字节编址, 则主存寻址范围是()。

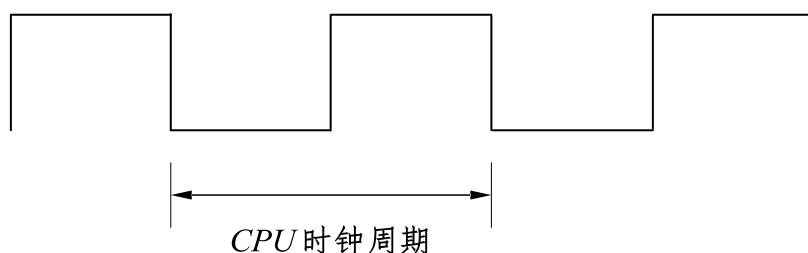
A. 64k B. 64kB C. 32k D. 32kB

答案: A

解析: (1) 要看清题中说的是按字节编址还是按字编址; (2) 寻址范围无单位, 存储容量有单位。

2) CPU 的性能指标

(1) CLK:



(2) CPU 主频 (时钟频率) = $\frac{1}{\text{CPU 时钟周期}}$;

(3) CPI: 执行一条指令所需的时钟周期数;

(4) IPS: 每秒执行的指令条数 ($IPS = \frac{\text{主频}}{\text{平均 CPI}}$);

(5) FLOPS: 每秒执行的浮点运算次数;

(6) 执行一条指令的耗时 = $\text{CPI} \times \text{CPU 时钟周期}$;

(7) CPU 执行时间 (整个程序的耗时) = $\frac{\text{CPU 时钟周期数}}{\text{主频}} = \frac{\text{指令条数} \times \text{CPI}}{\text{主频}}$ 。

题 2. 已知 CPU 主频为 1000Hz, 程序包含 100 条指令, 平均 $\text{CPI} = 3$, 则程序在 CPU 上执行需多久?

答案: $100 \times 3 \times \frac{1}{1000} = 0.3$ 。

题 3. 某计算机主频为 1.2GHz, 其指令分为 4 类, 它们在基准程序中所占比例及 CPI 如下表所示:

| 指令类型 | 所占比例 | CPI |
|------|------|-----|
| A | 50% | 2 |
| B | 20% | 3 |
| C | 10% | 4 |
| D | 20% | 5 |

该机的 MIPS 数是 ()。

A. 100 B. 200 C. 400 D. 600

答案: C

解析: 由表格可知 CPI 为:

$$50\% \times 2 + 20\% \times 3 + 10\% \times 4 + 20\% \times 5 = 3,$$

法一: 主频为 1.2GHz, 则时钟周期为 $\frac{1}{1.2\text{GHz}} = \frac{5}{6} \times 10^{-9} \text{s}$,

所以每秒执行指令数: $\frac{1}{\frac{5}{6} \times 10^{-9} \times 3} = 4 \times 10^8$ 条,

所以 $MIPS$ 为 400 ；

法二： CPI 为 3，主频为 $1.2GHz$ ，即 $1200MHz$ ，

则 $MIPS = \frac{1200}{3} = 400$ 。

课时一 练习题

- 计算机系统的层次结构从内到外依次为()。
 - 系统软件、硬件软件、应用软件
 - 硬件系统、系统软件、应用软件
 - 系统软件、应用软件、硬件系统
 - 应用软件、硬件系统、系统软件
- 将高级语言源程序转换为机器级目标代码文件的程序是()。
 - 汇编程序
 - 链接程序
 - 编译程序
 - 解释程序
- 目前的计算机,从原理上讲_____。
 - 指令以二进制形式存放,数据以十进制形式存放
 - 指令以十进制形式存放,数据以二进制形式存放
 - 指令和数据都以二进制形式存放
 - 指令和数据都以十进制形式存放
- 下列关于冯·诺伊曼结构计算机基本思想的叙述中,错误的是()。
 - 程序的功能都通过中央处理器执行指令实现
 - 指令和数据都用二进制数表示,形式上无差别
 - 指令按地址访问,数据都在指令中直接给出
 - 程序执行前,指令和数据需预先存放在存储器中
- 在冯·诺伊曼计算机中,CPU区分从存储器中取出的是指令还是数据的方法是()。
 - 指令和数据所在的存储单元地址不同
 - 访问指令和访问数据所处的指令执行阶段不同
 - 访问指令和访问数据的寻址方式不同
 - 指令和数据表示方式不同
- 下列选项中,描述浮点数操作速度指标的是()。
 - MIPS
 - CPI
 - IPC
 - MFLOPS
- 假定处理器 P_1 时钟频率为每秒 1.5GHz ,其对应的CPI为2,如果一个程序执行的时间为10秒,那么执行该程序的时钟周期和指令分别是()。
 - 15G, 7.5G
 - 30G, 15G
 - 7.5G, 30G
 - 15G, 30G
- 冯·诺伊曼计算机具有哪些基本特点?

课时二 数据的表示

| 考点 | 重要程度 | 占分 | 题型 |
|-----------|------|-------|-------|
| 1. 数制与编码 | ★★★ | 2 ~ 4 | 选择、填空 |
| 2. 定点数的表示 | ★★★ | 2 ~ 4 | 选择、填空 |
| 3. 浮点数的表示 | ★★ | 2 ~ 4 | 选择、填空 |

1. 数制与编码

1) 进位计数制

(1) $r(2, 8, 10, 16)$ 进制数

基数 = r 每个数码位可能出现 r 种字符, 逢 r 进 1

① 基数

每个数码位所用到的不同符号的个数, r 进制的基数为 r

② 符号

二进制: 0, 1;

八进制: 0, 1, 2, 3, 4, 5, 6, 7;

十进制: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;

十六进制: 0, 1, ..., 9, A(10), B(11), C(12), D(13), E(14), F(15);

③ 权重

符号与其位置共同影响权重 (例: 十进制中的个、十、百等)

④ 表示符号

| 二进制 | 八进制 | 十进制 | 十六进制 |
|-----|-----|-----|--------|
| B | O | D | $H/0x$ |

(2) r 进制 \rightarrow 十进制

各数码位与其位权的乘积之和

题 1. 二进制 10010.110 转化为十进制是_____。

答案: 18.75

解析: $1 \times 2^4 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-2} = 18.75$

(3) 二进制 \leftrightarrow 八进制

每3个二进制位对应1个八进制位

题 2. 11000010.01101B 转化为八进制是_____。

答案: $(302.32)_8$

解析: 以小数点为转化中心, 向两边看, 3位一组转换为对应的八进制符号, 不够3位则补0。

$$\begin{array}{ccccccc} 011 & 000 & 010 & . & 011 & 010 \\ \hline 3 & 0 & 2 & . & 3 & 2 \end{array}$$

因此 $11000010.01101B = (302.32)_8$

(4) 二进制 \leftrightarrow 十六进制

每4个二进制位对应1个十六进制位

题 3. C2.68H 转化为二进制是_____。

答案: 11000010.01101B

解析: 以小数点为中心向两边看, 1个十六进制符号可转换为4个二进制符号

$$\begin{array}{ccccccc} C & 2 & . & 6 & 8 & H \\ \hline 1100 & 0010 & . & 0110 & 1000 & B \end{array}$$

因此 $C2.68H = 11000010.01101B$

(5) 十进制 \rightarrow 二进制

将十进制数拆分为 $2^n (n = 0, 1, 2 \dots)$ 之和即可

题 4. 十进制357 转化为二进制是_____。

答案: 101100101B

解析: $357 = 2^8 + 2^6 + 2^5 + 2^2 + 2^0$

因此 $(357)_{10} = 101100101B$

注: 有的十进制小数无法用二进制精确表示, 如: 0.3

(6) 十进制 \rightarrow 八/十六进制

先将十进制数转化为二进制数，再将二进制数转化为八/十六进制

2) 编码

(1) BCD 编码

①概念

用4位二进制数来表示1位十进制中的数码(0~9)

②8421 码

a. 8421 码的映射关系

| | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 |

b. 8421 码的加法

例如: $8 + 4 = (12)_{10}$, 用8421 码表示:

$1000 + 0100 = 1100$ (不在映射表中, 则需 “+6” 进行修正) $\xrightarrow{+0110} 10010$

$$\begin{array}{r} 0001 \quad 0010 \\ \hline 1 \quad 2 \end{array}$$

注: 若相加结果在合法范围内, 则无需修正

(2) ASCII 码

①7 位二进制编码 (通常用8位表示一个字符, 最高位为0) 表示128 种字符;

②所有大写字母、小写字母、数字的编码都连续;

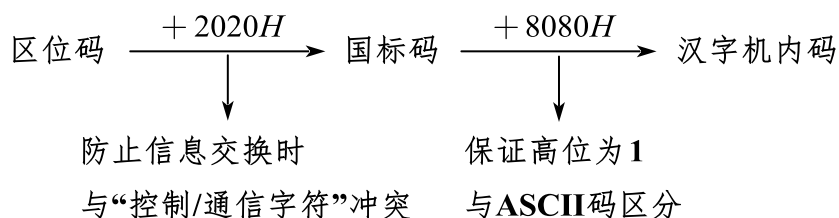
③常见的 ASCII 码: '0': $(48)_{10}$; 'A': $(65)_{10}$; 'a': $(97)_{10}$

题 5. 已知 'A' 的 ASCII 码为 65, 字符 'G' 存在存储单元 M 中, 则 M 中存放的内容为_____。

答案: 01000111

解析: A 的码值 $(65)_{10} = (100\ 0001)_2$, A 为第一个字母, G 为第七个字母, 所以 'G' 的码值为 $(100\ 0111)_2$, 因此 M 中存放的内容为 0100 0111 (存储单元为 8 位, 若问 G 的 ASCII 码, 则为 100 0111)

(3) 汉字的表示和编码



(4) 字符串

①从低地址到高地址逐个字符存储，采用'\0'作为结尾标志

②对于多字节的数据（如：汉字），可采用大/小端存储模式

大端模式：将数据的最高有效字节存放在低地址单元中。

小端模式：将数据的最高有效字节存放在高地址单元中

(5) 校验码

①概念

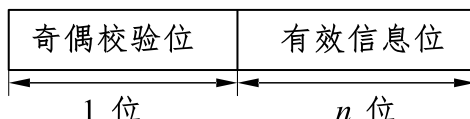
校验码是指能够发现或能自动校正错误的数据编码，其原理是通过增加一些冗余码，来检验或纠错编码。

②奇偶校验码

a. 基本原理：由若干位信息位再加一位二进制位组成校验码

b. 奇校验码：整个校验码（有效信息位和校验位）中“1”的个数为奇数

c. 偶校验码：整个校验码（有效信息位和校验位）中“1”的个数为偶数



题 6. 请写出编码1011010 和1100111 的奇校验码和偶校验码（设最高位为校验位）。

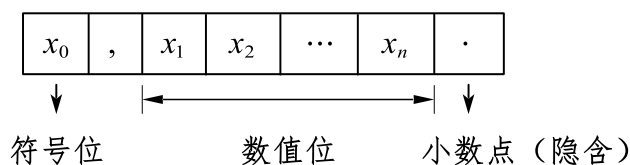
答案：1011010 的奇校验码为：1101 1010；偶校验码为：0101 1010

1100111 的奇校验码为：0110 0111；偶校验码为：1110 0111

解析：1011010 中已经有4个“1”，所以奇校验时最高位应为1，才符合“1”的个数为奇数，其他同理。

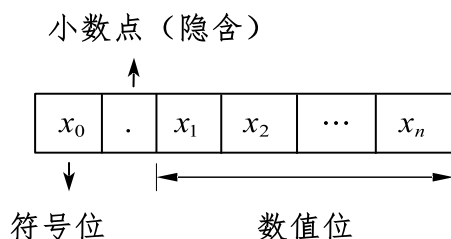
2. 定点数的表示

1) 定点整数为纯整数，约定小数点的位置在有效数值部分最低位之后。



2) 定点小数

定点小数是纯小数（即小于1），约定小数点位置在符号位之后，有效数值部分最高位之前。



3) 原、反、补、移码之间的转换

(1) 原码表示法

用机器数的最高位表示该数的符号（0：正；1：负），其余各位表示数的绝对值。

注：真值零的原码表示有2种： $[+0]_{\text{原}} = 0,000$ 和 $[-0]_{\text{原}} = 1,000$

(2) 反码表示法

正数：反码与原码相同；

负数：原码符号位不变，数值部分全部取反（ $0 \rightarrow 1; 1 \rightarrow 0$ ）

注：真值零的反码表示有2种， $[+0]_{\text{反}} = 0,000$ 和 $[-0]_{\text{反}} = 1,111$

(3) 补码表示法

正数：补码与原码相同；

负数：原码符号位不变，数值部分全部取反（ $0 \rightarrow 1; 1 \rightarrow 0$ ），末位加1（即“取反加1”）。

注：①此方法也可逆着用，即由 $[X]_{\text{补}}$ 求 $[X]_{\text{原}}$

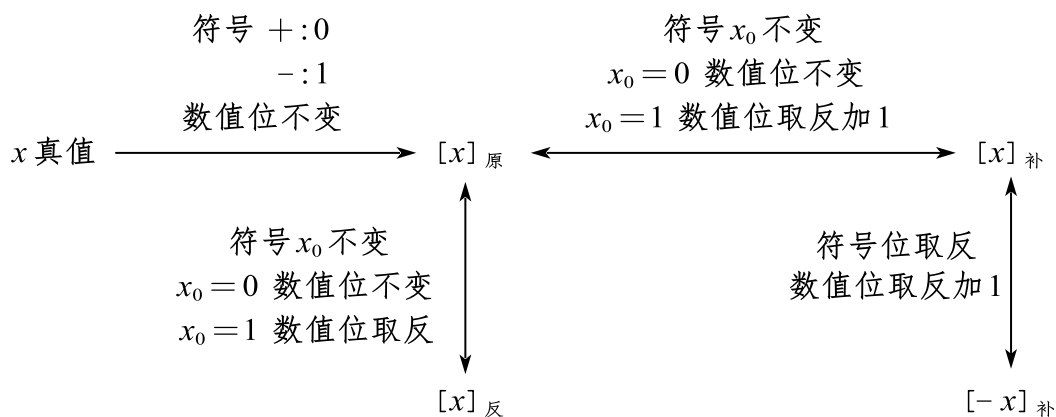
②真值零的补码表示只有1种， $[0]_{\text{补}} = 0,000$

(4) 移码表示法

将 $[X]_{\text{补}}$ 的符号位取反即得到 $[X]_{\text{移}}$

注：①移码只能用来表示定点整数，（移码多用于表示浮点数的阶码）

②真值零的移码表示只有1种， $[0]_{\text{移}} = 1,000$



题 1. 若 $[X]_{\text{补}} = 0.1101010$ ，则 $[X]_{\text{原}} = (\quad)$ 。

A. 1.0010101

B. 0.1101010

C. 0.0010110

D. 1.0010110

答案：B

解析：遇到 $[X]_{\text{原}}$ 与 $[X]_{\text{补}}$ 之间的转换，不要直接套口号“取反加1”，要先看符号！

题 2. 设寄存器内容为 FFH ，若其表示为 127，则为 码；若其表示为 -127，则为 码；

若其表示为 -1，则为 码；若其表示为 -0，则为 码。

答案：移；原；补；反

解析：遇到此类题，先将其转换为二进制

$$FFH = 1111\ 1111B$$

所以表示 127，则为移码；表示 -127，则为原码；表示 -1，则为补码；表示 -0，则为反码；

4) 原、反、补、移码的表示范围

(1) 定点整数的原、反、补、移码的表示范围

① $n+1$ 位定点整数（1 位符号位）用原码表示时，其表示范围为：

$$-(2^n - 1) \leq [X]_{\text{原}} \leq +(2^n - 1)$$

② $n+1$ 位定点整数（1 位符号位）用反码表示时，其表示范围为：

$$-(2^n - 1) \leq [X]_{\text{反}} \leq +(2^n - 1)$$

③ $n+1$ 位定点整数（1 位符号位）用补码表示时，其表示范围为：

$$-2^n \leq [X]_{\text{补}} \leq +(2^n - 1)$$

④ $n+1$ 位定点整数（1 位符号位）用移码表示时，其表示范围为：

$$-2^n \leq [X]_{\text{移}} \leq +(2^n - 1)$$

(2) 定点小数的原、反、补码的表示范围

① $n+1$ 位定点小数 (1 位符号位) 用原码表示时, 其表示范围为:

$$-(1 - 2^{-n}) \leq [X]_{\text{原}} \leq +(1 - 2^{-n})$$

② $n+1$ 位定点小数 (1 位符号位) 用反码表示时, 其表示范围为:

$$-(1 - 2^{-n}) \leq [X]_{\text{反}} \leq +(1 - 2^{-n})$$

③ $n+1$ 位定点小数 (1 位符号位) 用补码表示时, 其表示范围为:

$$-1 \leq [X]_{\text{补}} \leq +(1 - 2^{-n})$$

题 3. 用 16 位字长 (1 位符号位) 表示定点小数时, 用原码表示的数值范围为 ()。

A. $0 \leq |N| \leq 1 - 2^{-(16-1)}$

B. $0 \leq |N| \leq 1 - 2^{-16}$

C. $0 \leq |N| \leq 1 - 2^{-(16+1)}$

D. $0 \leq |N| \leq 1$

答案: A

解析: 注: ① 数值位为 $16-1$;

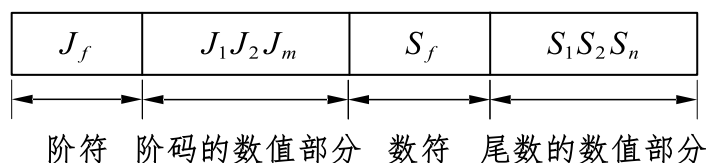
② n 位数值位, 原码所表示的定点小数范围为 $-(1 - 2^{-n}) \leq [X]_{\text{原}} \leq 1 - 2^{-n}$ (关于原点对称)

3. 浮点数的表示

1) 概念

浮点数的表示为: $N = r^E \times M$

其中, r 是浮点数阶码的底 (隐含), 与尾数的基数相同, 通常 $r=2$ 。 E 和 M 都是有符号的定点数, E 称为阶码, M 称为尾数。即浮点数由阶码和尾数两部分组成, 如下图所示。



注: ① 阶码是整数, 阶符 J_f 和阶码的位数 m 共同反映浮点数的表示范围及小数点的实际位置;

② 数符 S_f 代表浮点数的符号, 尾数的位数 n 反映浮点数的精度。

2) 规格化浮点数

(1) 概念

①浮点数规格化形式：为了提高运算精度，充分利用尾数的有效数位，规定尾数的最高数位上保证是一个有效值

②左规：当浮点数运算的结果为非规格化时，要进行规格化处理，将尾数算数左移一位，阶码减1的方法称为左规。左规可能要进行多次。

③右规：当浮点数运算的结果尾数出现溢出（双符号位为01或10）时，将尾数算数右移一位，阶码加1的方法称为右规。右规只需进行一次。

注：当 $r=2$ 时，规格化浮点数的尾数 M 满足 $\frac{1}{2} \leq |M| \leq 1$

(2) 尾数为原码时规格化

正数为 $0.1 \times \times \dots \times$ 的形式，其最大值表示为 $0.11 \dots 1$ ，最小值表示为 $0.100 \dots 0$ 。尾数的表示范围为 $\frac{1}{2} \leq M \leq (1 - 2^{-n})$ 。

负数为 $1.1 \times \times \dots \times$ 的形式，其最大值表示为 $1.10 \dots 0$ ，最小值表示为 $1.11 \dots 1$ 。尾数的表示范围为 $-(1 - 2^{-n}) \leq M \leq -\frac{1}{2}$ 。

(3) 尾数为补码时规格化

正数为 $0.1 \times \times \dots \times$ 的形式，其最大值表示为 $0.11 \dots 1$ ，最小值表示为 $0.100 \dots 0$ 。尾数的表示范围为 $\frac{1}{2} \leq M \leq (1 - 2^{-n})$ 。

负数为 $1.0 \times \times \dots \times$ 的形式，其最大值表示为 $1.01 \dots 1$ ，最小值表示为 $1.00 \dots 0$ 。尾数的表示范围为 $-1 \leq M \leq -\left(\frac{1}{2} + 2^{-n}\right)$ 。

注：这里补码规格化尾数的最大负数形式为 $1.01 \dots 1$ ，而不是原码的形式 $1.10 \dots 0$ ，因为 $1.10 \dots 0$ 不是补码规格数，所以规格化尾数的最大负数是

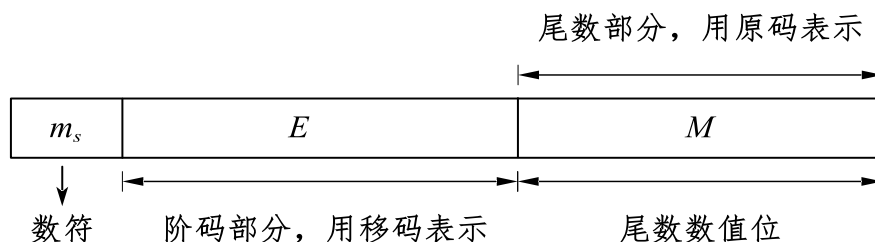
$-(0.10 \dots 0 + 0.0 \dots 01) = -0.10 \dots 01$ ，而 $(-0.10 \dots 01)_{\text{补}} = 1.01 \dots 1$

题 1. 若浮点数格式中基值一定，且尾数采用规格化表示，则浮点数的表示范围取决于____的位数，而精度取决于____的位数。

答案：阶码；尾数

3) IEEE 754 标准

按照 IEEE 754 标准，常用的浮点数的格式如下图所示。



IEEE 754 标准浮点数的格式

IEEE 754 标准规定常用的浮点数格式有短浮点数（单精确、float 型）、长浮点数（双精度、double 型）、临时浮点数，见下表

IEEE 754 浮点数的格式

| 类型 | 数符 | 阶码 | 尾数数值 | 总位数 | 偏置值 | |
|-------|----|----|------|-----|-------|-------|
| | | | | | 十六进制 | 十进制 |
| 短浮点数 | 1 | 8 | 23 | 32 | 7FH | 127 |
| 长浮点数 | 1 | 11 | 52 | 64 | 3FFH | 1023 |
| 临时浮点数 | 1 | 15 | 64 | 80 | 3FFFH | 16383 |

注：①短浮点数和长浮点数都是采用隐含尾数最高数位方法，因此可多表示一位尾数。临时浮点数无隐含位。

②阶码是以移码形式存储的，对于短浮点数，偏置值为127，因此其阶码真值 = 阶码 $E - 127$ 。

IEEE 754 标准中，规格化的短浮点数的真值为 $(-1)^s \times 1.M \times 2^{E-127}$ ，规格化长浮点数的真值为： $(-1)^s \times 1.M \times 2^{E-1023}$ 。

式中， $s=0$ 表示正数， $s=1$ 表示负数；短浮点数 E 的取值为1~254（8位表示）， M 为23位，共32位；长浮点数 E 的取值为1~2064（11位表示）， M 为52位，共64位。IEEE 754 标准浮点数的范围见下表。

IEEE 754 浮点数的范围

| 格式 | 最小值（绝对值） | 最大值（绝对值） |
|-----|---|--|
| 单精度 | $E = 1, M = 0, 1.0 \times 2^{1-127} = 2^{-126}$ | $E = 254, M = .111 \dots,$ $1.111 \dots 1 \times 2^{254-127} = 2^{127} \times (2 - 2^{-23})$ |
| 双精度 | $E = 1, M = 0, 1.0 \times 2^{1-1023} = 2^{-1022}$ | $E = 2046, M = .1111 \dots,$ $1.111 \dots 1 \times 2^{2046-1023} = 2^{1023} \times (2 - 2^{-52})$ |

题 2. *float* 类型 (即 IEEE 754 单精度浮点数格式) 能表示的最大正整数是 ()。

- A. $2^{126} - 2^{103}$ B. $2^{127} - 2^{104}$ C. $2^{127} - 2^{103}$ D. $2^{128} - 2^{104}$

答案: D

解析: ① IEEE 754 单精度浮点数是尾数用采取隐藏位策略的原码表示, 且阶码用移码 (偏置值为 127) 表示的浮点数。规格化短浮点数的真值为 $(-1)^S \times 1.M \times 2^{E-127}$, 其中 S 为符号位, 阶码 E 的取值为 1~254 (8 位表示) 尾数 m 为 23 位, 共 32 位; 因此 *float* 类型能表示的最大整数是 $1.111 \dots 1 \times 2^{254-127} = 2^{127} \times (2 - 2^{-23}) = 2^{128} - 2^{104}$, 因此选 D。

② IEEE 754 单精度浮点数的格式如下图所示。

| | | |
|--------|--------|---------|
| 数符 (1) | 阶码 (8) | 尾数 (23) |
|--------|--------|---------|

表示最大正整数时: 数符取 0; 阶码取最大值为 127; 尾数部分隐含了整数部分的 "1", 23 位尾数全取 1 时尾数最大, 为 $2 - 2^{-23}$, 此时浮点数的大小为

$$(2 - 2^{-23}) \times 2^{127} = 2^{128} - 2^{104}$$

课时二 练习题

1. 下列数中最小的数是 ()。

- A. $(10011001)_2$ B. $(227)_8$ C. $(98)_{16}$ D. $(152)_{10}$

2. 下列机器数中, 真值最小的数是 ()。

- A. $[x]_{\text{原}} = 0101101$ B. $[x]_{\text{反}} = 0101101$

- C. $[x]_{\text{补}} = 0101101$ D. $[x]_{\text{移}} = 0101101$

3. 下列编码为字符的奇偶校验码, 没有错误, 且采用偶校验编码的是 ()。

- A. 0110 1111 B. 1100 1011 C. 1101 0101 D. 1010 1101

4. 能够检出错误的校验码集的码距必须大于等于 ()。

- A. 1 B. 2 C. 3 D. 4

5. 关于奇偶校验的校验能力, 下列说法正确的是 ()。

- A. 奇校验能够检验出奇数位错误, 偶校验能够检验出偶数位错误

- B. 奇偶校验能检验并纠正1位错误

- C. 奇偶校验只能检验出1位错误, 不能纠正错误

- D. 奇偶校验能够检验出奇数位错误, 不能纠正错误

6. 设用8位二进制数表示一个定点纯小数, 最高位表示符号位, 其它位表示数值位。若用补码表示, 则可表示的小数范围为_____。

- A. $[-(1-2^{-7}), 1-2^{-7}]$ B. $[-(1-2^{-8}), 1-2^{-8}]$

- C. $[-1, 1-2^{-7}]$ D. $[-1, 1-2^{-8}]$

7. 用十六进制形式表示的 IEEE 754 标准32位单精度浮点数的规格化最大负数的机器数为 ()。

- A. 80C00000H B. 80800000H C. 80000000H D. 80000001H

8. IEEE 754 单精度浮点格式表示的数中, 最小的规格化正数是 ()。

- A. 1.0×2^{-126} B. 1.0×2^{-127} C. 1.0×2^{-128} D. 1.0×2^{-149}

课时三 数据的运算

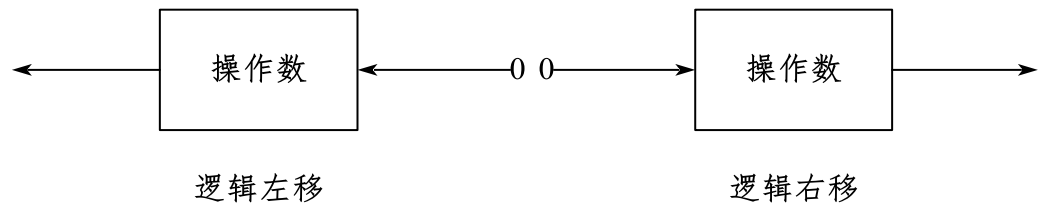
| 考点 | 重要程度 | 占分 | 题型 |
|-------------|-------|--------|--------|
| 1. 定点数的运算 | ★★★★★ | 4 ~ 10 | 解答题 |
| 2. 浮点数的运算 | ★★★★ | 5 ~ 8 | 选择、解答题 |
| 3. 加法器与 ALU | ★★★ | 2 ~ 6 | 选择、解答题 |

1. 定点数的运算

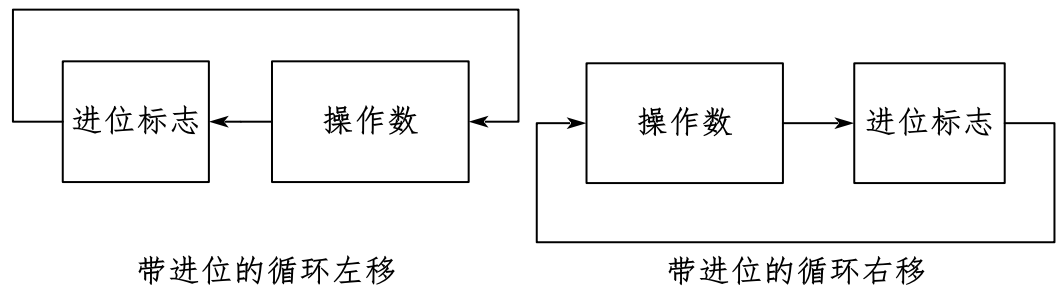
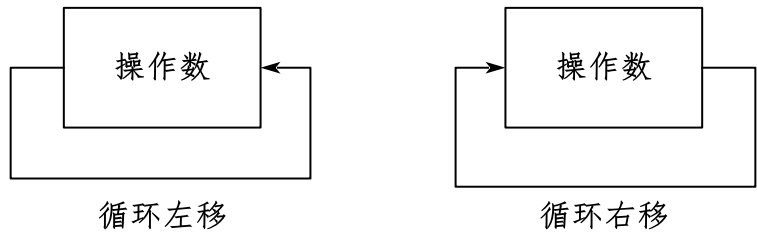
1) 移位运算与误差的舍入处理

(1) 逻辑移位：是指移位过程中不考虑数据符号位含义，包括逻辑左移、逻辑右移，通常用于对无符号数或逻辑数进行移位。

方法：不管逻辑左移还是逻辑右移，对移位空出位一律补0，对于移出位，不管0,1，一律丢掉。



(2) 循环移位：将机器字的首尾相接进行移位，其移位通路构成一个环路，不管左移还是右移，其移出位都会自动填补到空出位中。



(3) 算数移位：针对带符号的移位操作，对于不同表示法的机器码，规则不同，具体如下：

原码、补码、反码的移位规则

| | 码制 | 符号位 | 右移 MSB 位填补植 | 左移 LSB 位填补植 |
|----|----------|-----|---------------|---------------|
| 正数 | 原码、反码、补码 | 0 | 0 | 0 |
| 负数 | 原码 | 1 | 0 | 0 |
| | 补码 | 1 | 1 | 0 |
| | 反码 | 1 | 1 | 1 |

(4) 误差的舍入处理

①截断法：当运算结果超出机器字长时，无条件丢掉超出的位数，且保留下来的数值不做任何改变。

②末位置1法：在舍去结果最低位之后超出数值位的同时，将机器数末位（即 LSB 位）置1。（当超出部分全是0时，末位不变）

③0舍1入法

- 原码/补码正数：若舍去部分最高位为0，则“舍”，且保留数值不做任何变化；若舍去部分最高位为1，则“入”，对保留部分的最低位加1
- 补码负数：
 - 若舍去部分最高位为0，则“舍”，且保留数值不做任何变化
 - 若舍去部分最高位为1，舍去部分其余各位全为0，则“舍”，且保留部分不做任何变化；
 - 若舍去部分最高位为1，舍去部分其余各位不全为0，则“入”，对保留部分的最低位+1

题 1. 某8位无符号数10010101 右移 1 位后的值为（ ）。

A. 01001010B

B. 00101010B

C. 10101010B

答案：A

解析：无符号数右移即逻辑右移。

2) 溢出的判别方法

(1) 概念

溢出：指运算结果超出了数的表示范围。通常，称大于机器所能表示的最大正数为上溢，称小于机器所能表示的最小负数为下溢。

(2) 采用双符号位判断溢出

双符号位法也称模4补码。运算结果的两个符号位 $S_{s1}S_{s2}$ 相同，表示未溢出；运算结果的

两个符号位 $S_{S1}S_{S2}$ 不同，表示溢出，此时最高位符号位代表真正的符号。

符号位 $S_{S1}S_{S2}$ 的各种情况如下：

① $S_{S1}S_{S2} = 00$ ：表示结果为正数，无溢出。

② $S_{S1}S_{S2} = 01$ ：表示结果正溢出。

③ $S_{S1}S_{S2} = 10$ ：表示结果负溢出。

④ $S_{S1}S_{S2} = 11$ ：表示结果为负数，无溢出。

3) 补码定点数加减运算

补码运算的特点如下（设机器字长为 $n+1$ ）：

a. 参与运算的两个操作数均用补码表示。

b. 按二进制运算规则运算，逢二进一。

c. 符号位与数值位按同样的规则一起参与运算，符号位运算产生的进位要丢掉，结果的符号位由运算得出。

d. 补码加减运算依据下面的公式进行。当参加的数是定点小数时，模 $M = 2$ ；当参加运算的数是定点整数时，模 $M = 2^{n+1}$ 。

$$[A+B]_{\text{补}} = [A]_{\text{补}} + [B]_{\text{补}} \pmod{M}$$

$$[A-B]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} \pmod{M}$$

注： \pmod{M} 运算是为了将溢出位丢掉。

也就是说，若做加法，则两数的补码直接相加；若做减法，则将被减数与减数的机器负数相加。

e. 补码运算的结果亦为补码。

题 2. 设机器字长为 8 位（含 1 位符号位）， $A = 15$ ， $B = 24$ ，求 $[A+B]_{\text{补}}$ 和 $[A-B]_{\text{补}}$ 。

答案： $[A+B]_{\text{补}} = 00100111$ ， $[A-B]_{\text{补}} = 11110111$ 。

解析： $A = +15 = +0001111$ ， $B = +24 = +0011000$ ，得：

$$[A]_{\text{补}} = 00001111, [B]_{\text{补}} = 00011000$$

$$\text{求得：} [-B]_{\text{补}} = 11101000$$

所以， $[A+B]_{\text{补}} = 00001111 + 00011000 = 00100111$ ，其符号位为 0，对应真值为 $+39$ 。

$[A - B]_{\text{补}} = [A]_{\text{补}} + [-B]_{\text{补}} = 00001111 + 11101000 = 11110111$, 其符号位为1, 对应真值为-9。

4) 原/补码的乘法运算

(1) 原补码的一位乘法

原码的一位乘法是符号位与数值位分开计算, 乘积符号由两个数的符号位“异或”形成, 而乘积的数值部分由两个数的绝对值相乘而得, 具体方法如下:

$|X| \times |Y|$, 按 $|Y|$ 的每一位从低到高计算, 遇到“1”则部分积 $+|X|$, 右移一位, 遇到“0”则部分积 $+0$, 右移一位, 右移次数=乘积 $|Y|$ 的位数。

题 3. 用原码一位乘计算 $X * Y$ 。 $X = 0.1100$, $Y = -0.1001$ 。

答案:

| 部分积 | 乘数 $ Y $ | | | |
|----------|----------|-----------|---------------------------------|------|
| 000000 | 1001 | 初始状态 | 000001 | 1001 |
| + 001100 | | + X | + 001100 | |
| 001100 | 1001 | 右移1位 | 001101 | 1001 |
| 000110 | 0100 | + 0, 右移1位 | 000110 | 1100 |
| 000011 | 0010 | + 0, 右移1位 | 符号位 $1 \oplus 0 = 1$ | |
| 000001 | 1001 | + X | $\therefore X * Y = 0.01101100$ | |

注: ①右移次数=乘积 $|Y|$ 的位数

②不管 X, Y 的正负, 计算数值时用绝对值来求, 即符号位都为0或00, 最终符号由 X, Y 的符号异或求得。

(2) 补码一位乘法 (Booth 算法)

这是一种有符号数的乘法, 采用相加和相减操作计算补码数据的乘积。

设 $[X]_{\text{补}} = x_s x_1 x_2 \cdots x_n$, $[Y]_{\text{补}} = y_s y_1 y_2 \cdots y_n$, 则运算规则如下:

①符号位参与运算, 运算的数均以补码表示。

②被乘数一般取双符号位参与运算, 部分积取双符号位, 初值为0, 乘数可取单符号位。

③乘数末位增设附加位 y_{n+1} , 且初值为0。

④根据 (y_n, y_{n+1}) 的取值来确定操作，见下表。

Booth 算法的移位规则

| y_n (高位) | y_{n+1} (低位) | 操作 |
|------------|----------------|------------------------------|
| 0 | 0 | 部分积右移一位 |
| 0 | 1 | 部分积加 $[X]_{\text{补}}$ ，右移一位 |
| 1 | 0 | 部分积加 $[-X]_{\text{补}}$ ，右移一位 |
| 1 | 1 | 部分积右移一位 |

⑤移位按补码右移规则进行。

⑥按照上述算法进行 $n+1$ 步操作，但第 $n+1$ 步不再移位（共进行 $n+1$ 次累加和 n 次右移），仅根据 $y_n + y_{n+1}$ 的比较结果做相应的运算。

题 4. 设机器字长为 5 位（符号位为 1 位， $n=4$ ）， $x = -0.1101$ ， $y = 0.1011$ ，采用 *Booth* 算法求 $X * Y$ 。

答案： $X * Y = -0.10001111$

解析： $[x]_{\text{补}} = 11.0011$ ， $[-x]_{\text{补}} = 00.1101$ ， $[y]_{\text{补}} = 0.1011$ 。*Booth* 算法的求解过程如下。

| (高位部分积) | | (低位部分积/乘数) | 说明 |
|---------|---------------------------|-------------------|---|
| | 00.0000 | 0.1011 | 0 丢失位 起始情况 |
| + | $[-x]_{\text{补}}$ 00.1101 | | $Y_4 Y_5 = 10, Y_5 - Y_4 = -1$, 则 $+[-x]_{\text{补}}$ |
| | 00.1101 | | |
| 右移 | 00.0110 | ----- 10.10110 | 右移部分积和乘数 |
| + | 00.0000 | | $Y_4 Y_5 = 11, Y_5 - Y_4 = 0$, 则 $+0$ |
| | 00.0110 | | |
| 右移 | 00.0011 | ----- 010.10110 | 右移部分积和乘数 |
| + | $[x]_{\text{补}}$ 11.0011 | | $Y_4 Y_5 = 01, Y_5 - Y_4 = 1$, 则 $+ [x]_{\text{补}}$ |
| | 11.0110 | | |
| 右移 | 11.1011 | ----- 0010.10110 | 右移部分积和乘数 |
| + | $[-x]_{\text{补}}$ 00.1101 | | $Y_4 Y_5 = 10, Y_5 - Y_4 = -1$, 则 $+[-x]_{\text{补}}$ |
| | 00.1000 | | |
| 右移 | 00.0100 | ----- 00010.10110 | 右移部分积和乘数 |
| + | $[x]_{\text{补}}$ 11.0011 | | $Y_4 Y_5 = 01, Y_5 - Y_4 = 1$, 则 $+ [x]_{\text{补}}$ |
| | 11.0111 | | |
| | | | 构成 $[x \cdot y]_{\text{补}}$ |

所以 $[x \cdot y]_{\text{补}} = 1.01110001$ ，得 $x \cdot y = -0.10001111$

(3) 乘法运算总结

| 乘法类型 | 符号位 | | | 累加次数 | 移位 | | |
|--------|------|-----|-----|-------|----|-----|------|
| | 参与运算 | 部分积 | 乘数 | | 方向 | 次数 | 每位次数 |
| 原码一位乘法 | 否 | 2 位 | 0 位 | n | 右 | n | 1 |
| 补码一位乘法 | 是 | 2 位 | 1 位 | $n+1$ | 右 | n | 1 |

5) 原补码的除法运算（定点小数的除法，规定被除数要小于除数）

(1) 原码除法运算（不恢复余数法）

原码除法主要采用原码不恢复余数法，也称原码加减交替除法。特点是商符和商值是分开进行的，商符由两个操作数的符号位“异或”形成。求商值的规则如下：

设被除数 $[x]_{\text{原}} = x_s x_1 x_2 \cdots x_n$ ，除数 $[y]_{\text{原}} = y_s y_1 y_2 \cdots y_n$ ，则：

① 商的符号： $Q_s = x_s \oplus y_s$ 。

② 商的数值： $|Q| = |X| / |Y|$ 。

求 Q 的不恢复余数法运算规则如下：

① 符号位不参加运算。

② 先用被除数减去除数 ($|X| - |Y| = |X| + |-Y| = |X| + [-|Y|]_{\text{补}}$)，当余数为正时，商上 1，余数和商左移一位，再减去除数；当余数为负时，商上 0，余数和商左移一位，再加上除数。

③ 当第 $n+1$ 步余数为负时，需加上 $|Y|$ 得到第 $n+1$ 步正确的余数（余数与被除数同号）。

题 5. 设机器字长为 5 位（符号位为 1 位， $n=4$ ）， $x=0.1011$ ， $y=0.1101$ ，采用原码加减交替除法求 x/y 。

答案： $x/y = +0.1101$ ，余 0.0111×2^{-4} （因为将数值左移一位，相当于乘 2^1 ，左移 4 位，即乘 2^4 ，现在要还原）

解析： $x=0.1011$ ， $y=0.1101$ ， $[|y|]_{\text{补}} = 0.1101$ ， $[-|y|]_{\text{补}} = 1.0011$ ，原码不恢复余数法的求解过程如下。

| 被除数/乘数 | 商 | 说明 |
|------------------------------------|--------|--|
| 0.1011 | 0.0000 | 起始情况 |
| $+ [- y]_{\text{补}} \quad 1.0011$ | | $- y $ 即 $+ [- y]_{\text{补}}$ |
| 1.1110 | 0.0000 | 部分余数为负，商上0 |
| 左移 | 0.0000 | 余数和商左移一位 $+ y $ |
| $+ [y]_{\text{补}} \quad 0.1101$ | | |
| 0.1001 | 0.0001 | 部分余数为正，商上1 |
| 左移 | 0.0010 | 余数和商左移一位， $- y $ 即 $+ [- y]_{\text{补}}$ |
| $+ [- y]_{\text{补}} \quad 1.0011$ | | |
| 0.0101 | 0.0011 | 部分余数为正，商上1 |
| 左移 | 0.0110 | 余数和商左移一位， $- y $ 即 $+ [- y]_{\text{补}}$ |
| $+ [- y]_{\text{补}} \quad 1.0011$ | | |
| 1.1101 | 0.0110 | 部分余数为负，商上0 |
| 左移 | 0.1100 | 余数和商左移一位 $+ y $ |
| $+ [y]_{\text{补}} \quad 0.1101$ | | |
| 0.0111 | 0.1101 | 部分余数为正，商上1 |
| ↑ 余数 | ↑ 商 | |

因此 $Q_s = x_s \oplus y = 0 \oplus 0 = 0$ ，得 $x/y = +0.1101$ ，余 0.0111×2^{-4}

(2) 补码除法运算（加减交替法）

补码一位除法的特点是，符号位与数值位一起参加运算，商符自然形成，除法第一步根据被除数和除数的符号决定是做加法还是减法：上商的原则根据余数和除数的符号位共同决定，同号上商“1”，异号上商“0”；最后步商恒置“1”。

加减交替法的规则如下：

- ①符号位参加运算，除数与被除数均用补码表示，商和余数也用补码表示。
- ②若被除数与除数同号，则被除数减去除数；若被除数与除数异号，则被除数加上除数。
- ③若余数与除数同号，则商上1，余数左移一位减去除数；若余数与除数异号，则商上0，余数左移一位加上除数。
- ④重复执行第③步操作 n 次。
- ⑤若对商的精度没有特殊要求，则一般采用“末位恒置1”法。

题 6. 设机器字长为5位（符号位为1位， $n=4$ ）， $x=0.1000$ ， $y=-0.1011$ ，采用补码加减交替除法求 x/y 。

答案： $[x/y]_{\text{补}} = 1.0101$ ，余 0.0111×2^{-4} 。

解析：采用两位符号表示。 $[x]_{\text{原}} = 00.1000$ ，则 $[x]_{\text{补}} = 00.1000$ 。 $[y]_{\text{原}} = 11.1011$ ，

$[y]_{\text{补}} = 11.0101$ ， $[-y]_{\text{补}} = 00.1011$ 。补码加减交替法的求解过程如下。

| 被除数 | 商 | 说明 |
|--------------------------------|--------|--|
| 00.1000 | 0.0000 | 起始情况 |
| $+ [y]_{\text{补}}$ 11.0101 | | $[x]_{\text{补}}$ 、 $[y]_{\text{补}}$ 异号则 $+ [y]_{\text{补}}$ |
| 11.1101 | 0.0001 | 部分余数与 $[y]_{\text{补}}$ 同号，则商上 1 |
| 左移 | 0.0010 | 左移一位 $+ [-y]_{\text{补}}$ |
| $+ [-y]_{\text{补}}$ 00.1011 | | |
| 00.0101 | 0.0010 | 部分余数与 $[y]_{\text{补}}$ 异号，则商上 0 |
| 左移 | 0.0100 | 左移一位 $+ [y]_{\text{补}}$ |
| $+ [y]_{\text{补}}$ 11.0101 | | |
| 11.1111 | 0.0101 | 部分余数与 $[y]_{\text{补}}$ 同号，则商上 1 |
| 左移 | 0.1010 | 左移一位 $+ [-y]_{\text{补}}$ |
| $+ [-y]_{\text{补}}$ 00.1011 | | |
| 00.1001 | 0.1010 | 部分余数与 $[y]_{\text{补}}$ 异号，则商上 0 |
| 左移 | 1.0100 | 左移一位 $+ [y]_{\text{补}}$ |
| $+ [y]_{\text{补}}$ 11.0101 | | |
| 00.0111 | 1.0101 | 末位恒置 1 |
| ↑ 余数 | ↑ 商 | |

所以有 $[x/y]_{\text{补}} = 1.0101$ ，余 0.0111×2^{-4} 。

(3) 除法运算总结

| 除法类型 | 符号位参与运算 | 加减次数 | 移位 | | 说明 |
|---------|---------|-------------------|----|-----|---------------|
| | | | 方向 | 次数 | |
| 原码加减交替法 | 否 | $N + 1$ 或 $N + 2$ | 左 | N | 若最终余数为负，需恢复余数 |
| 补码加减交替法 | 是 | $N + 1$ | 左 | N | 商末位恒置 1 |

2. 浮点数的计算

1) 浮点数的加减运算

浮点数运算的特点是阶码和尾数分开运算，浮点数的加减运算一律采用补码，主要分为

以下几步：

(1) 对阶

对阶的目的是使两个操作数的小数点位置对齐，即使得两个数的阶码相等，为此，先求阶差，然后以小阶向大阶看齐的原则，将阶码小的尾数右移一位（基数为2），阶加1，直到两个数的阶码相等为止。尾数右移时，舍弃掉有效位会产生误差，影响精度。

(2) 尾数求和

将对阶后的尾数按定点数加（减）运算规则运算。

(3) 规格化

以双符号位为例，当尾数大于0时，其补码规格化形式为：

$$[S]_{\text{补}} = 00.1 \times \times \cdots \times$$

当尾数小于0时，其补码规格化形式为：

$$[S]_{\text{补}} = 11.0 \times \times \cdots \times$$

可见，当尾数的最高数值与符号位不同时，即为规格化形式。规格化分为左规与右规两种。

a. 左规：当尾数出现 $00.0 \times \times \cdots \times$ 或 $11.1 \times \times \cdots \times$ 时，需左规，即尾数左移1位，和的阶码减1，直到尾数为 $00.1 \times \times \cdots \times$ 或 $11.0 \times \times \cdots \times$ 。

b. 右规：当尾数求和结果溢出（如尾数为 $10. \times \times \cdots \times$ 或 $01. \times \times \cdots \times$ ）时，需右规，即尾数右移一位，和的阶码加1。

注：①对于左规和右规，不应死记。考察尾数的大小，左规一次相当于乘2，右规一次相当于除以2；

② $[-1/2]_{\text{补}} = 1.1000$ 不是规格化数，需左规一次， $[-1]_{\text{补}} = 1.0000$ 才是规格化数。

(4) 舍入

在对阶和右规的过程中，可能会将尾数低位丢失，引起误差，影响精度。常见的舍入方法有：“0”舍“1”入法和恒置“1”法。

“0”舍“1”入法：类似于十进制数运算中的“四舍五入”法，即在尾数右移时，被移去的最高数值位为0，则舍去；被移去的最高数值位为1，则在尾数的末位加1。这样做可能会使尾数又溢出，此时需再做一次右规。

恒置“1”法：尾数右移时，不论丢掉的最高数值位是“1”还是“0”，都使右移后的尾数末位恒置“1”。这种方法同样有使尾数变大或变小的两种可能。

(5) 溢出判断,

与定点数加减法一样, 浮点数加减运算最后一步也需判断溢出。

在浮点数规格化中已指出当尾数之和(差)出现 $01.\times\times\cdots\times$ 或 $10.\times\times\cdots\times$ 时, 并不表示溢出, 只能将此数右规后, 再根据阶码来判断浮点数运算结果是否溢出。

浮点数的溢出与否是由阶码的符号决定的。以双符号位补码为例, 当阶码的符号位出现“01”时, 即阶码大于最大阶码时, 表示上溢, 进入中断处理; 当阶码的符号位出现“10”时, 即阶码小于最小阶码时, 表示下溢, 按机器零处理。实际上原理还是阶码符号位不同表示溢出, 且真实符号位和高位符号位一致。

题 7. 已知十进制数 $X = -5/256$ 、 $Y = +59/1024$, 按机器补码浮点运算规则计算 $X - Y$, 结果用二进制表示, 浮点数格式如下: 阶符取 2 位, 阶码取 3 位, 数符取 2 位, 尾数取 9 位。

答案: $(X - Y)$ 的真值为 $2^{-3} \times (-0.1001111)_2$

解析: 浮点数的格式如下:

| 阶符 2 | 阶码 3 | 数符 2 | 尾数 9 |
|------|------|------|------|
|------|------|------|------|

$$X = -5/256 = (-101)_2/2^8 = 2^{-101} \times (-0.101000000)_2$$

$$Y = +59/1024 = (111011)_2/2^{10} = 2^{-100} \times (0.111011000)_2$$

$$[X]_{\text{补}} = 11011, 11.011000000, [Y]_{\text{补}} = 11100, 00.111011000。$$

$$\textcircled{1} \text{求阶差: } [\Delta E]_{\text{补}} = 11011 + 00100 = 11111, \text{ 知 } \Delta E = -1$$

$$\textcircled{2} \text{对阶: } [X]_{\text{补}} = 11100, 11.101100000$$

$\textcircled{3}$ 尾数求差:

$$\begin{array}{r} 11.101100000 \\ + 11.000101000 \\ \hline 10.110001000 \end{array}$$

$$[X - Y]_{\text{补}} = 11100, 10.110001000$$

$$\textcircled{4} \text{结果右规一次: } [X - Y]_{\text{补}} = 11101, 11.011000100。$$

$$\textcircled{5} \text{常阶码, 无溢出, 结果真值为 } 2^{-3} \times (-0.1001111)_2。$$

3. 加法器与 ALU

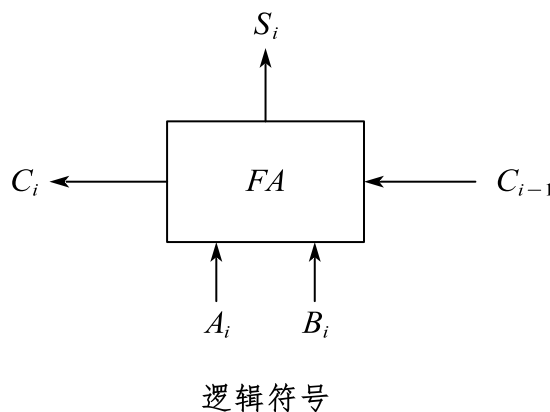
1) 加法器

(1) 一位全加器 (FA)

全加器的逻辑表达式如下。

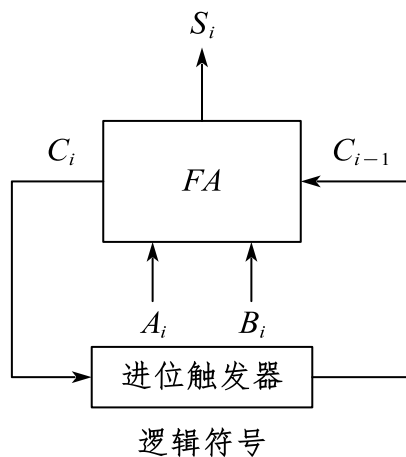
和表达式: $S_i = A_i \oplus B_i \oplus C_{i-1}$ (A_i 、 B_i 、 C_{i-1} 中有奇数个1时, $S_i = 1$; 否则 $S_i = 0$)

进位表达式: $C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$



(2) 串行加法器

在串行加法器中, 只有一个全加器, 数据逐位串行送入加法器中进行运算。若操作数长 n 位, 则加法器就要分 n 次进行, 每次产生一位和, 并且串行逐位地送回寄存器。进位触发器用来寄存进位符号, 以便参与下一次运算。



(3) 并行加法器

进位表达式为: $C_i = G_i + P_i C_{i-1}$ ($G_i = 1$ 或 $P_i G_{i-1} = 1$ 时, $C_{i-1} = 1$)

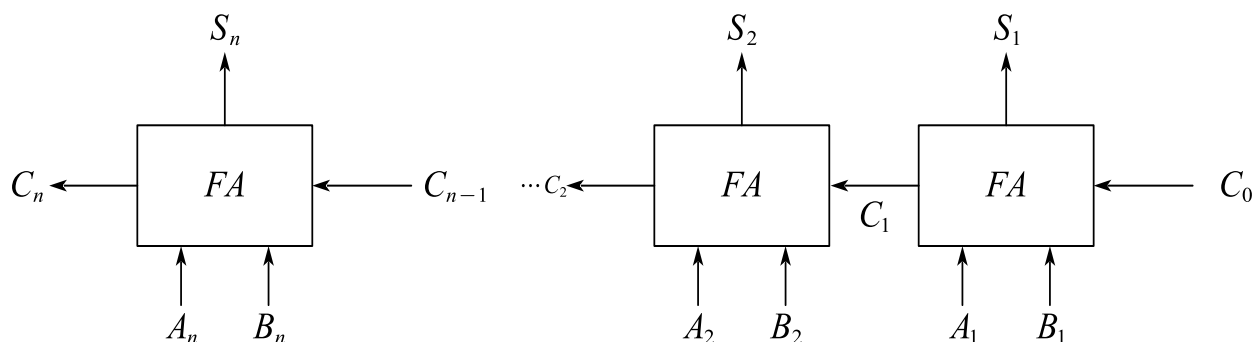
式中, G_i 是进位产生函数, $G_i = A_i B_i$;

P_i 是进位传递函数, $P_i = A_i \oplus B_i$;

并行加法器的进位分为串行进位和并行进位。

a. 串行进位 (行波进位)

把 n 个全加器串接起来,就可进行两个 n 位数的相加,每级进位直接依赖于前一级的进位,即进位信号是逐级形成的。



串行进位的并行加法器

其中,

$$C_1 = A_1 B_1 + (A_1 \oplus B_1) C_0 \text{ 或 } (C_1 = G_1 + P_1 C_0)$$

$$C_2 = A_2 B_2 + (A_2 \oplus B_2) C_1 \text{ 或 } (C_2 = G_2 + P_2 C_1)$$

...

$$C_n = A_n B_n + (A_n \oplus B_n) C_{n-1} \text{ 或 } (C_n = G_n + P_n C_{n-1})$$

注: 低位运算产生进位所需要的时间将影响高位的运算时间。

b. 并行进位 (先行进位、同时进位)

其特点是各级进位信号同时形成, 各级进位信号表达式如下:

$$C_1 = G_1 + P_1 C_0$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 C_0$$

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$$

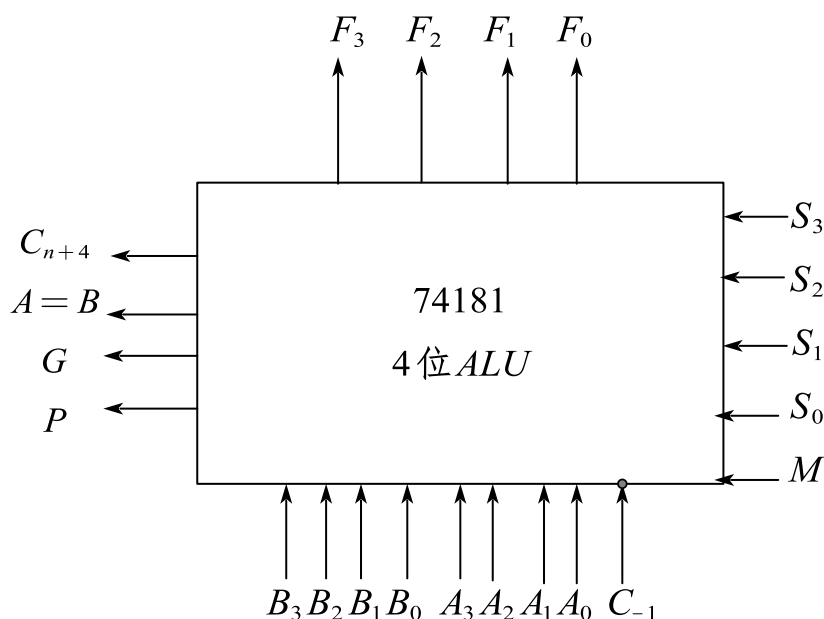
...

注: 上述所有进位的输出仅由 G_i 、 P_i 和最低进位输入 C_0 决定, 而不依赖于其低位的进位输入 C_{i-1} , 因此各级进位输出可以同时产生。

2) ALU

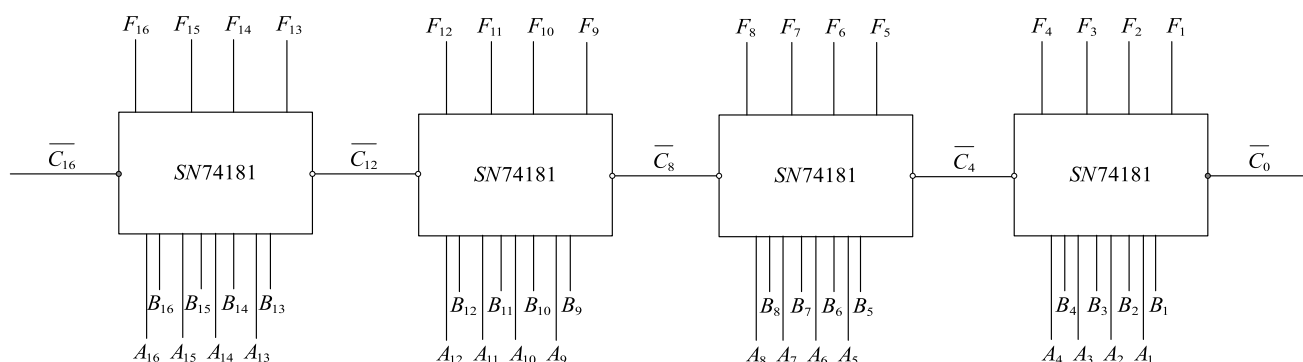
ALU是一种功能较强的组合逻辑电路, 能进行多种算术运算和逻辑运算。

以典型的4位ALU芯片(74181)为例介绍ALU结构:

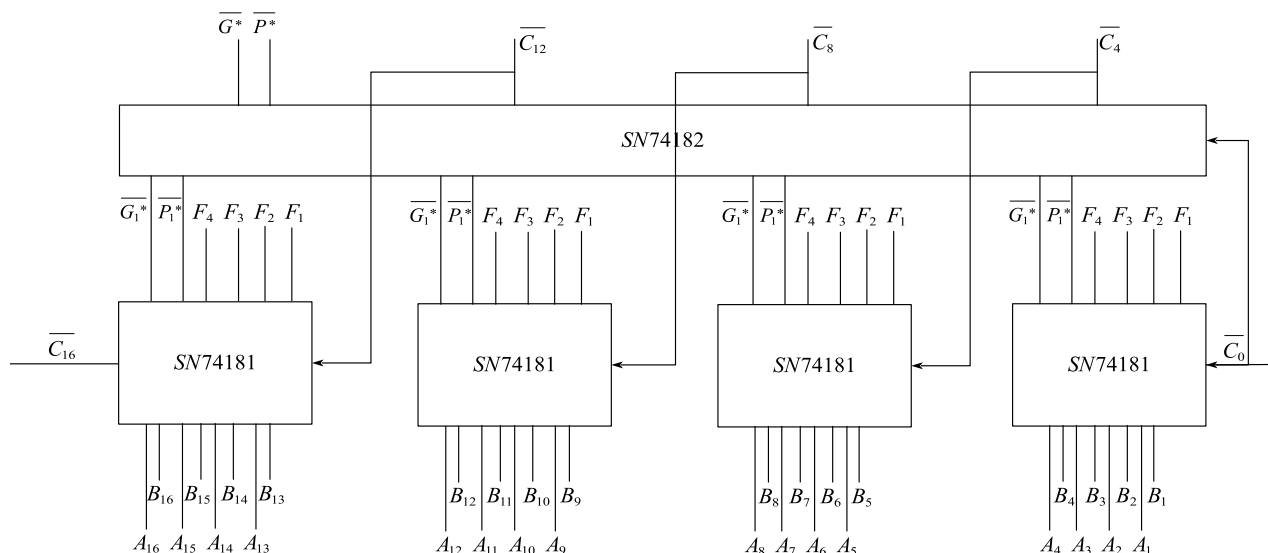


74181 外特性示意图

74181 为 4 位并行加法器，其 4 位进位是同时产生的，用 4 片 74181 芯片可组 16 位 ALU 。其片内进位是快速的，但片间进位是逐片传递的，即组内并行（74181 片内），组间串行（74181 片间），如下图所示。因此，总的形成时间还是比较长的。

16 位组内并行、组间串行进位 ALU

若把 16 位 ALU 中的每 4 位作为一组，即将 74181 与 74182 芯片（先行进位芯片）配合，用类似位间快速进位的方法来实现 16 位 ALU （4 片 ALU 组成），则能得到 16 位的两级先行进位 ALU ，即组内并行（74181 片内），组间并行（74181 片间），如下图所示。



16位组内并行、组间并行进位ALU

题 8. 某加法器进位链小组信号为 C_4 、 C_3 、 C_2 、 C_1 ，低位来的进位信号为 C_0 ，请分别按下述两种方式写出 C_1 、 C_2 、 C_3 和 C_4 的逻辑表达式。

1) 串行进位方式。

2) 并行进位方式。

解析：1) 串行进位方式：

$$C_1 = G_1 + P_1 C_0 \quad \text{其中, } G_1 = A_1 B_1, P_1 = A_1 \oplus B_1$$

$$C_2 = G_2 + P_2 C_1 \quad G_2 = A_2 B_2, P_2 = A_2 \oplus B_2$$

$$C_3 = G_3 + P_3 C_2 \quad G_3 = A_3 B_3, P_3 = A_3 \oplus B_3$$

$$C_4 = G_4 + P_4 C_3 \quad G_4 = A_4 B_4, P_4 = A_4 \oplus B_4$$

2) 并行进位方式：

$$C_1 = G_1 + P_1 C_0$$

$$C_2 = G_2 + P_2 G_1 + P_2 P_1 C_0$$

$$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0$$

$$C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 C_1 + P_4 P_3 P_2 P_1 C_0$$

其中， $G_1 \sim G_4$ 、 $P_1 \sim P_4$ 表达式与串行进位方式相同。

课时三 练习题

- 浮点加减中的对阶是（ ）。
 - 将较小的一个阶码调整到与较大的一个阶码相同
 - 将较大的一个阶码调整到与较小的一个阶码相同
 - 将被加数的阶码调整到与加数的阶码相同
 - 将加数的阶码调整到与被加数的阶码相同
- 四片74181ALU和一片74182CLA器件相配合，具有如下进位传递功能（ ）。
 - 行波进位
 - 组内先行进位，组间先行进位
 - 组内先行进位，组间行波进位
 - 组内行波进位，组间先行进位
- 下列有关浮点数加减运算的叙述中，正确的是（ ）。
 - 对阶操作不会引起阶码上溢或下溢
 - 右规和尾数舍入都可能引起阶码上溢
 - 左规时可能引起阶码下溢
 - 尾数溢出时结果不一定溢出
 - 仅II、III
 - 仅I、II、IV
 - 仅I、III、IV
 - I、II、III、IV
- 若 $x=103$ ， $y=-25$ ，则下列表达式采用8位定点补码运算实现时，会发生溢出的是（ ）。
 - $x+y$
 - $-x+y$
 - $x-y$
 - $-x-y$
- 整数 x 的机器数为11011000，分别对 x 进行逻辑右移1位和算术右移1位操作，得到的机器数各是（ ）。
 - 11101100、11101100
 - 01101100、11101100
 - 11101100、01101100
 - 01101100、01101100
- 设某浮点数的阶码采用移码表示，尾数采用原码表示，判断该浮点数是否为规格化数的方法是（ ）。
 - 尾数的最高位为1，其余位任意。
 - 尾数的最高位为0，其余位任意。
 - 尾数的最高位和数符相同，其余位任意。
 - 尾数的最高位和数符相异，其余位任意。
- 为什么用算术逻辑单元ALU和移位器能够实现定点数和浮点数的加减乘除运算？

8. 设浮点数 $X = 0.110101 \times 2^{010}$, $Y = -0.101010 \times 2^{100}$, 若阶码取3位, 尾数取6位, (均不包括符号位), 按补码运算步骤计算 $X + Y$ 和 $X - Y$ 。

课时四 主存储器

| 考点 | 重要程度 | 占分 | 题型 |
|----------------|------|-------|-------|
| 1. 存储器概述 | ★★ | 2 ~ 4 | 选填、填空 |
| 2. 主存储器与CPU的连接 | ★★★★ | 4 ~ 8 | 选填、大题 |

1. 存储器概述

基本概念

1) 存储器的分类

(1) 按在计算机中的层次分类

①主存储器，又称主存、内存。CPU可直接随机对其访问，其也可与高速缓冲存储器（Cache）和辅助存储器交换数据。

②辅助存储器，又称辅存、外存，辅存的内容需调入主存才能被CPU访问。

③高速缓冲存储器，简称Cache，位于主存与CPU之间，用来存放当前CPU经常使用的指令和数据，以便CPU能高速访问它们。

(2) 按存储介质分类

分为磁表面存储器（磁盘、磁带）、半导体存储器和光存储器（光盘）等。

(3) 按存取方式分类

①随机存储器（RAM）。存储器的任何一个存储单元都可以随机存取，且存取时间与存储单元的物理位置无关，RAM又分为静态RAM（SRAM）和动态RAM（DRAM）。

②只读存储器（ROM）。存储器的内容只能随机读出而不能写入，ROM与RAM的存取方式均为随机存取，信息写入之后就固定不变，即使断电，也不会丢失。

③串行访问存储器。包括顺序存取存储器（如磁带）和直接存取存储器（如磁盘）。

(4) 按信息的可保存性分类

①易失性存储器（断电后，存储信息消失）：RAM。

②非易失性存储器（断电后信息仍保持）：ROM、磁盘、光盘。

③破坏性读出（某个存储单元的信息被读出时，原存储信息被破坏）：半导体存储器。

④非破坏性读出（某个存储单元的信息被读出时，原存储信息不被破坏）：磁盘。

题 1. 在下列几种存储器中，CPU不能直接访问的是（ ）

A. 硬盘 B. 内存 C. Cache D. 寄存器

答案：A

题 2. 磁盘属于（ ）类型的存储器。

A. 随机存取存储器 (RAM) B. 只读存储器 (ROM)
C. 顺序存取存储器 (SAM) D. 直接存取存储器 (DAM)

答案：D

2) 存储器的性能指标

(1) 存储容量 = 存储字数 × 字长 (如 $1K \times 8$ 位)

存储字数表示存储器的地址空间大小，字长表示一次存取操作的数据量。

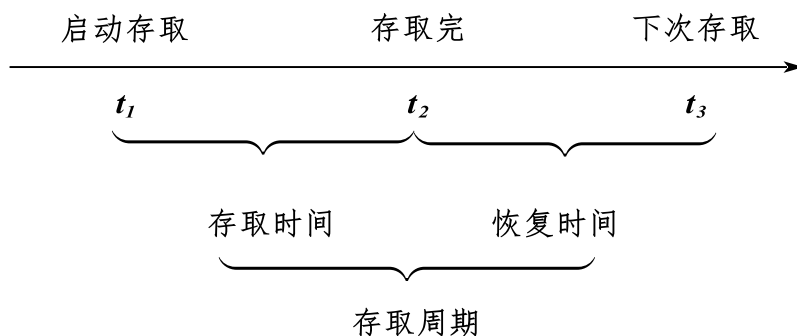
(2) 存储速度：数据传输率 = 数据的宽度 / 存储周期

① 存取时间 (T_a)：存取时间是指从启动一次存储器操作到完成该操作所经历的时间，分为读出时间和写入时间。

② 存取周期 (T_m)：存取周期又称读写周期或访问周期。它是指存储器进行一次完整的读写操作所需的全部时间，即连续两次独立访问存储操作 (读或写操作) 之间所需的最小时间间隔。

③ 主存带宽 (B_m)：主存带宽又称数据传输率，表示每秒从主存进出信息的最大数量，单位字/秒、字节/秒 (B/s) 或位/秒 (b/s)。

注：存取时间不等于存储周期，通常存储周期大于存取时间。



存取时间与存储周期的关系

题 3. 存储器的存取周期是指（ ）

A. 存储器的读出时间

B. 存储器的写入时间

C. 存储器进行连续读写或写操作所允许的最短时间间隔

D. 存储器进行一次读或写操作所需的平均时间

答案：C

解析：注意C中的“连续”。

题 4. 若某存储器存储周期为 $250ns$ ，每次读出16位，该存储器的数据传输率是（ ）。

A. $4 \times 10^6 B/s$

B. $4MB/s$

C. $8 \times 10^6 B/s$

D. $8 \times 2^{20} B/s$

答案：C

解析：每个存储周期读出 $16bit = 2B$ ，因此数据传输率 = $\frac{2B}{(250 \times 10^{-9})s} = 8 \times 10^6 B/s$ 。注：

数据传输中的 M 指 10^6 而非 2^{20} 。

3) 存储系统的层次

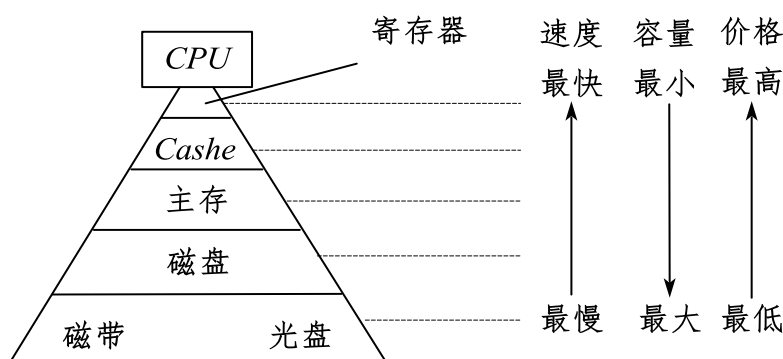


图1 多级存储器结构

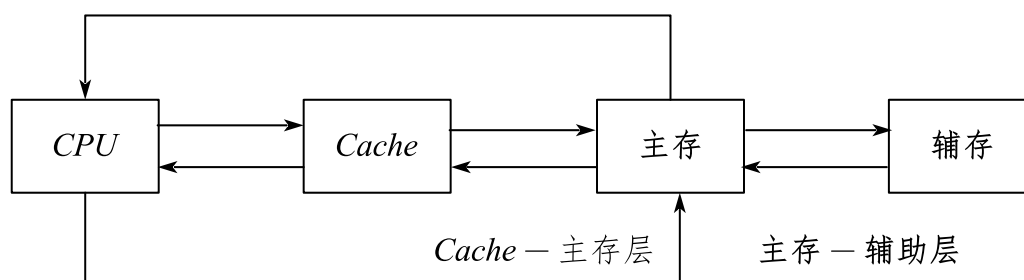


图2 三级存储系统的层次结构及其构成

注：①Cache—主存层速度接近于Cache，容量和价格接近于主存；主存—辅存层速度接近于主存，容量和价格接近于辅存。

②主存与Cache之间的数据调动由硬件自动完成，对所有程序员是透明的（透明的意思

是感受不到其存在);主存与辅存之间的数据调动是由硬件和操作系统共同完成的,对应用程序员是透明的。

③Cache—主存层与主存—辅存层中,上一层的内容只是下一层内容的副本。

题 5. 在多级存储体系中“Cache—主存层”结构的作用时解决()的问题。

- A. 主存容量不足
- B. 主存与辅存速度不匹配
- C. 辅存与CPU速度不匹配
- D. 主存与CPU速度不匹配

答案: D

题 6. 存储器分层体系结构中,存储器从速度最快到最慢的排列顺序是()

- A. 寄存器—主存—Cache—辅存
- B. 寄存器—主存—辅存—Cache
- C. 寄存器—Cache—辅存—主存
- D. 寄存器—Cache—主存—辅存

答案: D

半导体随机存储器

1) SRAM 芯片和 DRAM 芯片

(1) SRAM 的工作原理

- ①存储元: 存放一个二进制位的物理器件。
- ②存储单元: 由地址码相同的多个存储元构成。
- ③存储体: 若干个存储单元的集合称为存储体。

静态随机存储器(SRAM)的储元是用双稳态触发器(元晶体管MOS)来记忆信息的,因此即使信息被读出后,它仍保持其原状态而不需要再生(非破坏性读出)。

SRAM的存取速度快,但集成度低,功耗大,价格高,一般用于高速缓冲存储器。

(2) DRAM 的工作原理

动态随机存储器(DRAM)是利用存储元电路中栅极电容上的电荷来存储信息的,但其电容上的电荷一般只能维持 $1\sim 2ms$,因此即使电源不断电,信息也会自动消失,为此,每隔一定时间必须刷新(即将存储单元的信息读出在立马写入),通常取 $2ms$,称为刷新周期。

常用的刷新方式有3种:

- ①集中刷新: 指在一个刷新周期内,利用一段固定的时间,依次对存储器的所有行进行

逐一再生，在此期间停止对存储器的读写操作，称为“死时间”，又称访存“死区”。优点是读写操作时不受刷新工作的影响；缺点是在集中刷新期间（死区）不能访问存储器。

②分散刷新：把对每行的刷新分散到各个工作周期中。这样，一个存储器的系统工作周期分为两部分：前半部分用于正常读、写或保持；后半部分用于刷新。这种刷新方式增加了系统的存取周期，如存储芯片的存取周期为 $0.5\mu s$ ，则系统的存取周期为 $1\mu s$ 。优点是没有死区；缺点是加长了系统的存取周期，降低了整机的速度。

③异步刷新：异步刷新是前两种方法的结合，它既可缩短“死时间”，又能充分利用最大刷新间隔为 $2ms$ 的特点。具体做法就是将刷新周期除以行数，得到两次刷新操作之间的时间间隔 t ，利用逻辑电路每隔时间 t 产生一次刷新请求。这样可以避免使CPU连续等待过长的时间，而且减少了刷新次数，从根本上提高了整机的工作效率。

注：①刷新对CPU是透明的，即刷新不依赖于外部的访问；

②动态RAM的刷新单位是行，由芯片内部自行生成行地址；

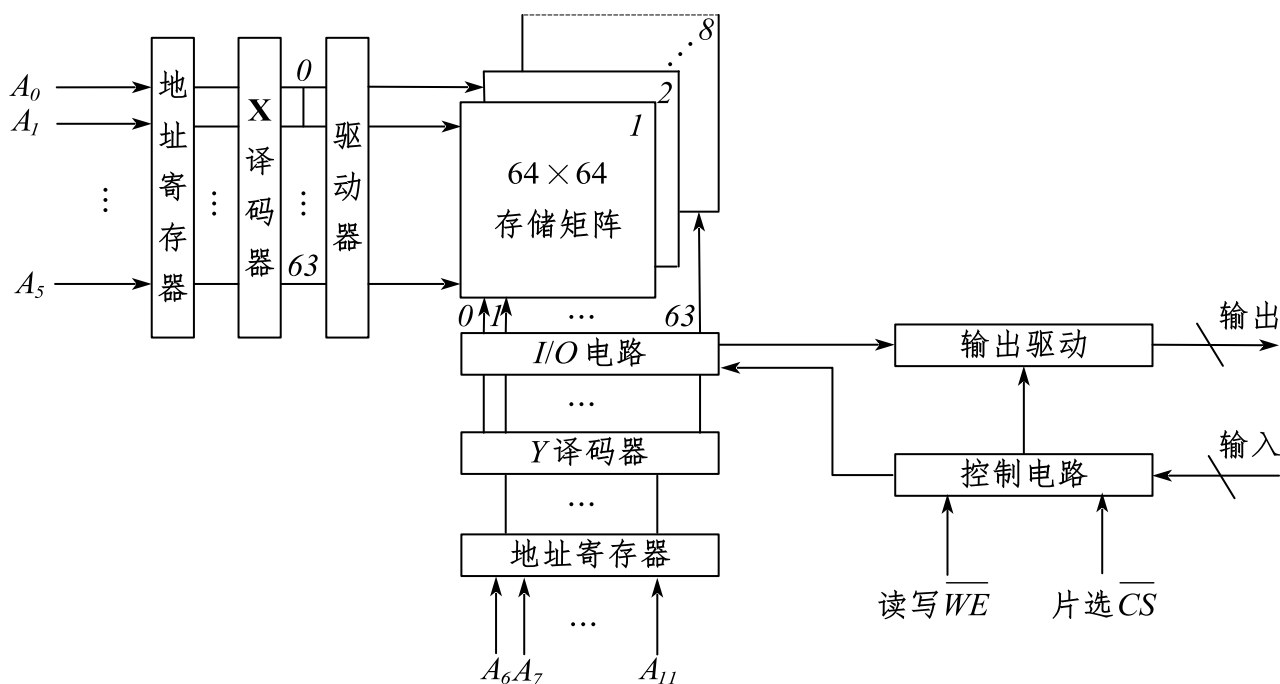
③刷新操作类似于读操作，但又有所不同，刷新时不需要选片，即整个存储器中的所有芯片同时被刷新。

（3）SRAM和DRAM的比较

| 特点 \ 类型 | SRAM | DRAM |
|---------|------|------|
| 存储信息 | 触发器 | 电容 |
| 破坏性读出 | 非 | 是 |
| 需要刷新 | 不要 | 需要 |
| 送行列地址 | 同时送 | 分两次送 |
| 运行速度 | 快 | 慢 |
| 集成度 | 低 | 高 |
| 存储成本 | 高 | 低 |
| 主要用途 | 高速缓存 | 主机内存 |

（4）存储器芯片的内部结构

如下图所示，存储芯片由存储体、I/O读写电路、地址译码和控制电路等部分组成。



存储芯片结构图

①存储体(存储矩阵)。存储体是存储单元的集合,它有行选择线(X)和列选择线(Y)来选择所访问单元,存储体的相同行、列上的位同时被读出或写入。

②地址译码器。用来将地址转换为译码输出线上的高电平,以便驱动相应的读写电路。

③I/O 控制电路。用以控制被选中的单元的读出或写入,具有放大信息的作用。

④片选控制信号。单个芯片容量太小,往往满足不了计算机对存储器容量的要求,因此需用一定数量的芯片进行存储器的扩展。在访问某个字时,必须“选中”该存储字所在的芯片,而其他芯片不被“选中”,因此需要有片选控制信号。

⑤读/写控制信号。根据CPU给出的读命令或写命令,控制被选中单元进行读或写。

题 7. 下面是有关SRAM与DRAM存储芯片的叙述

I. DRAM 芯片的集成度比SRAM 芯片的高

II. DRAM 芯片的成本比SRAM 芯片的高

III. DRAM 芯片的速度比SRAM 芯片的快

IV. DRAM 芯片工作时需要刷新, SRAM 片工作时不需要刷新

通常情况下, 错误的是 ()

A. I 和 II B. II 和 III C. III 和 IV D. I 和 IV

答案: B

解析：II. SRAM 芯片的成本比 DRAM 高 III. SRAM 芯片的速度比 DRAM 快

2) 只读存储器

(1) 只读存储器 (ROM) 的特点

- ①结构简单，位密度比可读写存储器高
- ②具有非易失性，可靠性高

(2) ROM 的类型

①掩模式只读存储器 (MROM)

其内容在生产时写入，不可改变，优点是可靠性强，集成度高；缺点灵活性差。

②一次可编程只读存储器 (PROM)

允许用户用专门的设备写入内容，但一旦写入，内容无法改变

③可擦除可编程只读存储器 (EPROM)

可多次写入、修改其内容，但写入时间长。

④Flash 存储器 (U 盘)

特点是既可在不加电的情况下长期保存信息，又可在线快速擦除与重写。

⑤固态硬盘 (SSD)

基于闪存的固态硬盘是用固态电子存储芯片阵列制成的硬盘，由控制单元和存储单元 (Flash 芯片) 组成，优点为读写速度更快，低功耗，缺点是价格高。

题 8. 下列几种存储器中，() 是易失性存储器。

- A. Cache B. EPROM C. Flash 存储器 D. CD - ROM

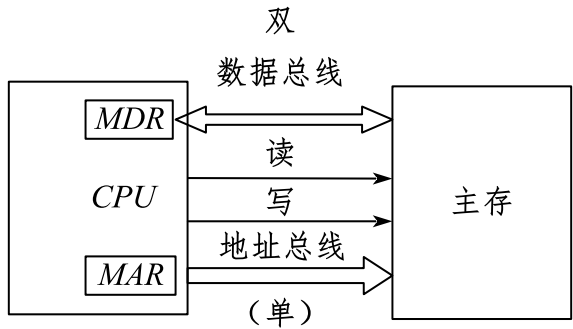
答案：A

2. 主存储器与 CPU 的连接

1) 连接原理

- ①主存储器通过数据总线、地址总线和控制总线与 CPU 连接。
- ②数据总线的位数与工作频率的乘积正比于数据传输率。
- ③地址总线的位数决定了可寻址的最大内存空间。

④控制总线（读/写）指出总线周期的类型和本次输入/输出操作完成的时刻。主存储器与CPU的连接如图所示。



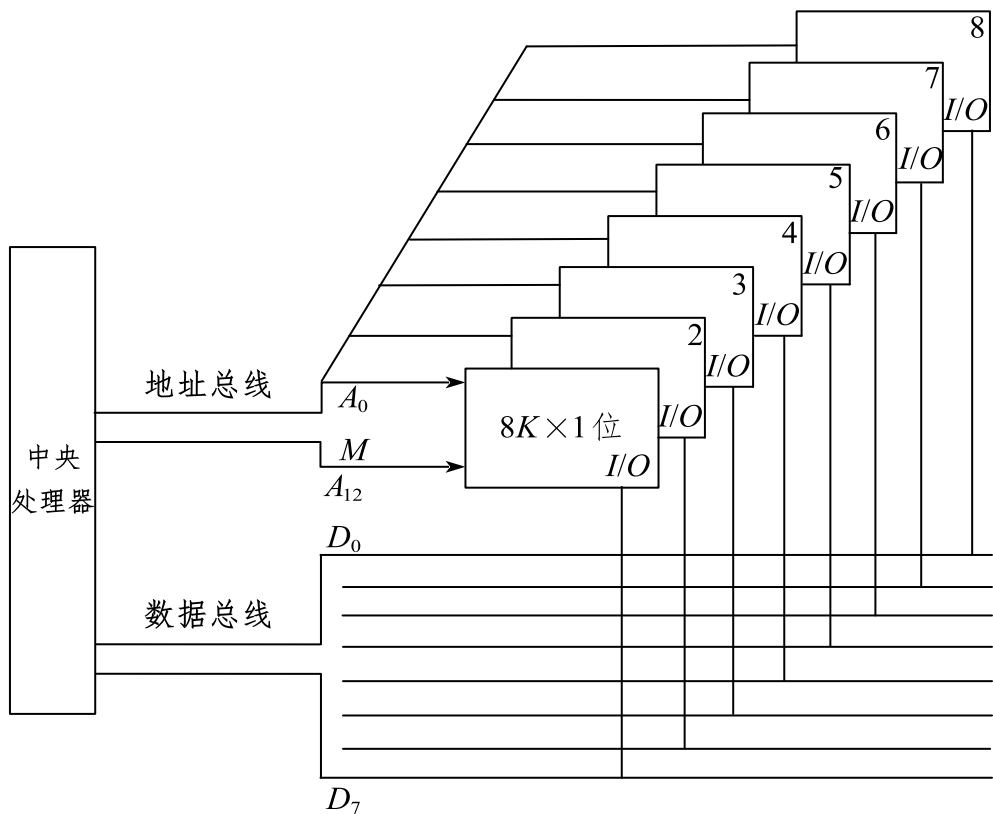
2) 主存容量的扩展

(1) 位扩展法

目的：使存储芯片的数据位数与CPU的数据线数相等。

方法：将多个存储芯片的地址端、片选端和读写控制端相应并联，数据端分别引出与CPU相连。

如下图所示，用8片 $8K \times 1$ 位的RAM芯片组成 $8K \times 8$ 位的存储器，8片RAM芯片的地址线 $A_{12} \sim A_8$ 、 \overline{CS} 、 \overline{WE} 都分别连在一起，每片数据线依次作为CPU数据线的一位。



位扩展连接示意图

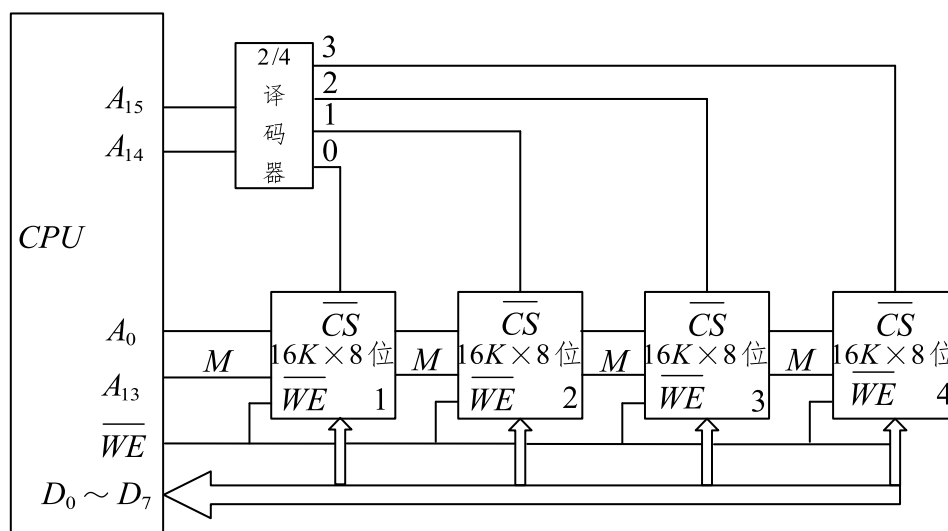
注：仅采用位扩展时，各芯片连接地址线的方式相同，但连接数据线的方式不同，在某一时刻选中所有的芯片，所以片选信号 \overline{CS} 要连接到所有芯片。

(2) 字扩展法

目的：增加存储器中字的数量，而位数不变。

方法：将芯片的地址线、数据线、读写控制线相并联，由片选信号来区分各芯片的地址范围。

如下图所示，用4片 $16K \times 8$ 位的 RAM 芯片组成 $64K \times 8$ 位的存储器。4片 RAM 芯片的数据线 $D_0 \sim D_7$ 和 \overline{WE} 都分别连在一起。将 $A_{15} A_{14}$ 用作片选信号， $A_{15} A_{14} = 00$ 时，译码器输出端0有效，选中最左边的1号芯片； $A_{15} A_{14} = 01$ 时，译码器输出端1有效，选中2号芯片，以此类推（在同一时间内只能有一个芯片被选中）。各芯片的地址分配如下：



字扩展连接示意图

第1片，最低地址：0000 0000 0000 0000；最高地址：0011 1111 1111 1111

第2片，最低地址：0100 0000 0000 0000；最高地址：0111 1111 1111 1111

第3片，最低地址：1000 0000 0000 0000；最高地址：1011 1111 1111 1111

第4片，最低地址：1100 0000 0000 0000；最高地址：1111 1111 1111 1111

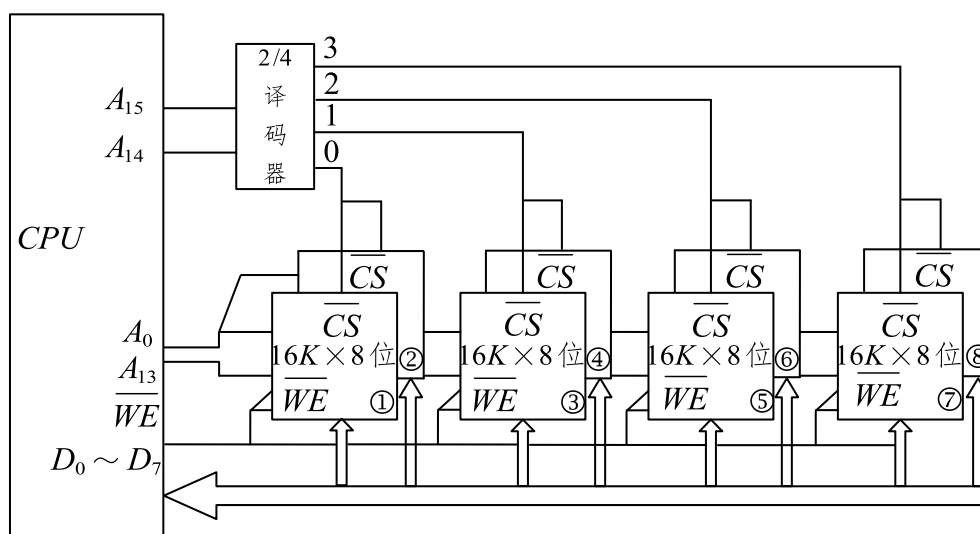
注：仅采用字扩展时，各芯片连接地址线的方式相同，连接数据线的方式也相同，但在某一时刻只需选中部分芯片，所以通过片选信号 \overline{CS} 或采用译码器设计连接到相应的芯片。

(3) 字位同时扩展法

目的：既增加存储字的数量，又增加存储字长。

如下图所示，用8片 $16K \times 4$ 位的 RAM 芯片组成 $64K \times 8$ 位的存储器。每两片构成一组

16K×8位的存储器（位扩展），4组便构成64K×8位的存储器（字扩展）。地址线 $A_{15}A_{14}$ 经译码器得到4个片选信号， $A_{15}A_{14}=00$ 时，输出端0有效，选中第一组芯片（①和②）； $A_{15}A_{14}=01$ 时，输出端1有效，选中第二组芯片（③和④），以此类推。



字位同时扩展及CPU的连接图

注：用字位同时扩展时，各芯片连接地址线的方式相同，但数据线的连接方式不同，而且需要通过片选信号 \overline{CS} 或采用译码器设计连接到相应的芯片。

3) 存储芯片的地址分配和片选

(1) 概念

片选：CPU要实现对存储单元的访问，首先要选择存储芯片，即片选。

字选：接着根据地址码在所选中的芯片中选择相应的存储单元，即字选。

注：①片选信号的产生主要为译码片选法；

②字选通常是由CPU送出的 N 条低地址线完成的。

(2) 译码片选法（ n 条线可连 2^n 个芯片）

译码片选法指用除片内寻址外的高位地址线通过地址译码器芯片产生信号。

例如：用8片8K×8位存储芯片组成64K×8位存储器（地址线16位，数据线8位），需要8个片选信号，即只需要3个高位地址，如 $A_{15}A_{14}A_{13}=000$ 时，选中第一个芯片， $A_{15}A_{14}A_{13}=001$ 时，选中第二个芯片，以此类推。

(3) 片选线的连接

片选线的连接是CPU与存储芯片连接的关键。存储器由许多存储芯片叠加而成，哪一片

被选中完全取决于该存储芯片的片选控制端 \overline{CS} 是否能接收到来自 CPU 的片选有效信号。

片选有效信号与 CPU 的访存控制信号 \overline{MREQ} （低电平有效）有关，因为只有当 CPU 要求访存时，才要求选中存储芯片。若 CPU 访问 I/O，则 \overline{MREQ} 为高，表示不要求存储器工作。

题 1. 用存储容量为 $16K \times 1$ 位的存储器芯片来组成一个 $64K \times 8$ 位的存储器，则在字的方向和位方向分别扩展了（ ）倍。

A. 4, 2 B. 8, 4 C. 2, 4 D. 4, 8

答案：D

解析：字方向： $\frac{64}{16} = 4$ ；位方向： $\frac{8}{1} = 8$ 。

课时四 练习题

- 下列各类存储器中，不采用随机存取方式的是（ ）。
A. EPROM B. CDROM C. DRAM D. SRAM
- 以下对于存储器刷新操作的描述中，正确的是（ ）。
A. 动态和静态 RAM 都需要刷新 B. 刷新是按行进行的
C. 刷新是按一个芯片接着一个芯片的顺序进行的 D. 所有的刷新方式都存在“死区”
- 动态存储器 DRAM 的刷新原则是（ ）。
A. 各 DRAM 芯片轮流刷新 B. 各 DRAM 芯片同时刷新，片内逐位刷新
C. 各 DRAM 芯片同时刷新，片内逐字刷新 D. 各 DRAM 芯片同时刷新，片内逐行刷新
- 某存储器容量为 64KB，按字节编址，地址 400H ~ 5FFFH 为 ROM 区，其余为 RAM。若采用 $8K \times 4$ 位的 SRAM 芯片进行设计，则需要该芯片的数量是（ ）。
A. 7 B. 8 C. 14 D. 16
- 某容量为 256MB 的存储器由若干 $4M \times 8$ 位的 DRAM 芯片构成，该 DRAM 芯片的地址引脚和数据引脚总数是（ ）。
A. 19 B. 22 C. 30 D. 36
- 某计算机主存容量为 64KB，其中 ROM 区为 4KB，其余为 RAM 区，按字节编址。现要用 $2K \times 8$ 位的 ROM 芯片和 $4K \times 4$ 位的 RAM 芯片来设计该存储器，则需要上述规格的 ROM 芯片数和 RAM 芯片数分别是（ ）。
A. 1、15 B. 2、15 C. 1、30 D. 2、30
- 已知某 16 位机的主存采用半导体存储器，地址码为 18 位，若使用 $8K \times 8$ 位 SRAM 芯片组成该机所允许的最大主存空间，并选用模板块结构形式。问：
(1) 若每个模板块为 $32K \times 16$ 位，共需要几个模板块？
(2) 每个模板块内共有多少片 RAM 芯片？
(3) 主存共需多少 RAM 芯片？CPU 如何选择模板块？
- 有一个 $1024K \times 32$ 位的存储器，由 $128K \times 8$ 位的 DRAM 芯片构成。问：
(1) 总共需要多少 DRAM 芯片？
(2) 设计此存储体组成框图。

课时五 辅存与Cache

| 考点 | 重要程度 | 占分 | 题型 |
|------------|-------|-------|-------|
| 1. 磁盘存储器 | ★★★★ | 4 ~ 6 | 选择、大题 |
| 2. 高速缓冲存储器 | ★★★★★ | 4 ~ 8 | 选择、大题 |
| 3. 虚拟存储器 | ★★★ | 2 ~ 4 | 选择 |

1. 磁盘存储器

1) 磁盘设备的组成

- (1) 磁盘驱动器：核心部件为磁头组件和盘片组件。
- (2) 磁盘控制器：硬盘存储器和主机的接口。
- (3) 存储区域：一块硬盘有若干记录面，每个记录面划分为若干磁道，每条磁道又划分为若干扇区，扇区（也称块）是磁盘读写的最小单位，即磁盘按块存取。

①磁头数(Heads)：即记录面数，表示硬盘共有多少个磁头，磁头用于读取/写入盘片记录面的信息，一个记录面对应一个磁头。

②柱面数(Cylinders)：表示硬盘每面盘片上有多少条磁道。在一个盘组中，不同记录面的相同编号(位置)的诸磁道构成一个圆柱面。

③扇区数(Sectors)：表示每条磁道上有多少个扇区。

2) 磁盘的性能指标

(1) 记录密度

- ①道密度：沿磁盘半径方向单位长度上的磁道数；
- ②位密度：磁道单位长度上能记录的二进制代码位数；
- ③面密度：位密度和道密度的乘积。

(2) 磁盘的容量

- ①非格式化磁盘容量 = 磁盘记录面数 × 磁道数 × 磁道容量
= 磁盘记录面数 × 道密度 × 盘片有效半径 × 位密度 × 磁道的周长
- ②格式化磁盘容量 = 磁盘记录面数 × 磁道数 × 每道扇区数 × 扇区容量

注：①格式化指为磁盘划分扇区，并增加管理位；

②格式化后的容量要比非格式化容量要小。

(3) 平均存取时间

平均存取时间 = 寻道时间 + 旋转延迟时间 + 传输时间

寻道时间：磁头移动到目的磁道的时间；

旋转延迟时间：磁头定位到要读写扇区的时间

传输时间：传输数据所花费的时间

(4) 数据传输率

假设磁盘转速为 r 转/秒, 每条磁道容量为 N 字节, 则数据传输率为: $D_r = rN$ 。

3) 磁盘存储器的优缺点

优点: ①存储容量大, 位价格低;

②记录介质可重复使用;

③记录信息可长期保存而不丢失, 甚至可脱机存档;

④非破坏性读出, 读出时不需再生。

缺点: 存取速度慢, 机械结构复杂。

4) 磁盘地址

主机向磁盘控制器发送寻址信息, 磁盘地址一般如下图所示:

| | | | |
|------|-----------|-----|-----|
| 驱动器号 | 柱面 (磁道) 号 | 盘面号 | 扇区号 |
|------|-----------|-----|-----|

例如, 若系统中有4个驱动器, 每个驱动器带一个磁盘, 每个磁盘256个磁道, 16个盘面, 每个盘面划分为16个扇区, 则每个扇区地址需要18位二进制代码, 其格式如下图所示。

| | | | |
|-----------|----------------|----------|----------|
| 驱动器号 (2位) | 柱面 (磁道) 号 (8位) | 盘面号 (4位) | 扇区号 (4位) |
|-----------|----------------|----------|----------|

5) 磁盘的工作过程

硬盘的主要操作是寻址、读盘、写盘。每个操作都对应一个控制字, 硬盘工作时, 第一步是取控制字, 第二步是执行控制字。

硬盘属于机械式部件, 其读写操作是串行的, 不可能在同一时刻既读又写, 也不可能在同一时刻读两组数据或写两组数据。

题 1. 下列关于磁盘存储器的叙述中, 错误的是 ()。

- A. 磁盘的格式化容量比非格式化容量小
- B. 扇区中包含数据、地址和校验等信息
- C. 磁盘存储器的最小读写单位为1字节
- D. 磁盘存储器由磁盘控制器、磁盘驱动器和盘片组成

答案: C

解析：磁盘存储器的最小读写单位为一个扇区，即磁盘按块存取，*C* 错误。磁盘存储数据之前需要进行格式化，将磁盘分成扇区，并写入信息，因此磁盘的格式化容量比非格式化容量小，*A* 正确。磁盘扇区中包含数据、地址和校验等信息，*B* 正确。磁盘存储器由磁盘控制器、磁盘驱动器和盘片组成，*D* 正确。

题 2. 设某硬盘的磁盘组有 16 个数据记录面，存储区域的最内圈直径为 22cm ，最外圈直径为 33cm ，道密度为 $80\text{道}/\text{cm}$ ，则该硬盘共有_____个柱面。

答案：440

解析： $\frac{1}{2}(33 - 22) = 5.5\text{cm}$ ， $5.5\text{cm} \times 80 = 440\text{道}$ ，所以该磁盘共有 440 个柱面。（16 个数据记录面在此为干扰信息）

2. 高速缓冲存储器

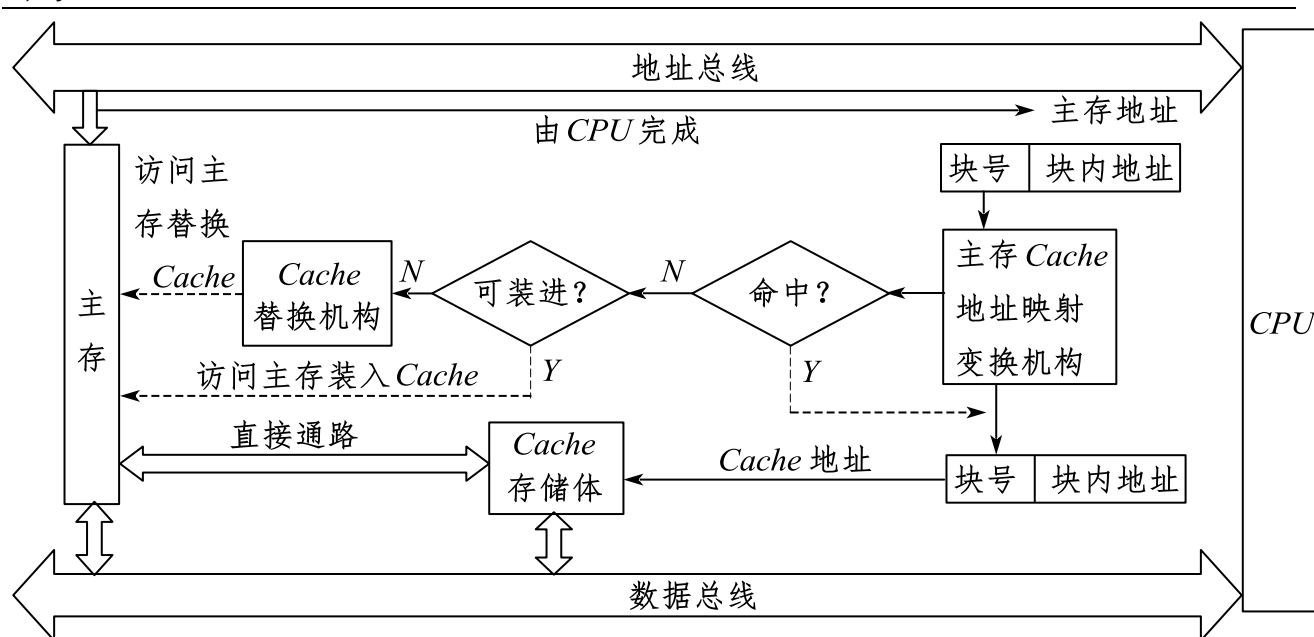
Cache — 主存层次结构通常来解决 *CPU* 和主存速度不匹配的问题，而支撑该方法的根本原理是：程序访问的局部性原理。

程序访问的局部原理：

（1）时间局部性：最近的未来要用的信息，很可能是现在正在使用的信息。如：程序中的循环。

（2）空间局部性：最近的未来要用的信息，很可能与现在正在使用的信息在存储空间上是邻近的。如：指令的顺序存放，数据的数组形式等。

1) *Cache* 的基本工作原理



高速缓冲存储器的工作原理

相关概念：

(1) 块：为了便于Cache和主存间交换信息，Cache和主存都被划分为相等的块，Cache块又称为Cache行，每个块由若干字节组成，块的长度称为块长（Cache行长）。

(2) 读操作：当CPU发出读请求时，若访问地址在Cache中命中，就将此地址换成Cache地址，直接对Cache进行读操作，与主存无关；若Cache不命中，则仍需访问主存，并把此字所在的块一次性地从主存中调入Cache。若此时Cache已满，则需根据某种替换算法，用这个块替换Cache中原来的某块信息。整个过程全部由硬件实现。值得注意的是，CPU与Cache之间的数据交换以字为单位，而Cache与主存之间的数据交换则以Cache块为单位。

(3) 写操作：当CPU发出写请求时，若Cache命中，有可能遇到Cache与主存中内容不一致的问题。例如，由于CPU写Cache，把Cache某单元中的内容从 X 修改成了 X' ，而主存对应单元中的内容仍然是 X ，没有改变。所以若Cache命中，需要按照一定的写策略处理，常见的处理方法有全写法和回写法，详见本节的Cache写策略部分。

(4) 命中率：CPU欲访问的信息已在Cache中的比率称为Cache的命中率。设一个程序执行期间，Cache的总命中次数为 N_c ，访问主存的总次数为 N_m ，则命中率 H 为：

$$H = N_c / (N_c + N_m)$$

(5) Cache主存系统的平均访问时间：

设 t_c 为命中时的 *Cache* 访问时间, t_m 为未命中时的主存访问时间, $1-H$ 表示未命中率, 则 *Cache* 主存系统的平均访问时间 T_a 为:

$$T_a = Ht_c + (1-H)t_m$$

根据 *Cache* 的读、写流程, 实现 *Cache* 时需解决以下关键问题:

- (1) 数据查找。如何快速判断数据是否在 *Cache* 中。
- (2) 地址映射。主存块如何存放在 *Cache* 中, 如何将主存地址转换为 *Cache* 地址。
- (3) 替换策略。*Cache* 满后, 使用何种策略对 *Cache* 块进行替换或淘汰。
- (4) 写入策略。如何既保证主存块和 *Cache* 块的数据一致性, 又尽量提升效率。

题 1. 假设 *Cache* 的速度是主存的 5 倍, 且 *Cache* 的命中率为 95%, 则采用 *Cache* 后, 存储器的性能提高多少 (设 *Cache* 和主存同时被访问, 若 *Cache* 命中则中断访问主存)?

答案: 3.17 倍

解析: 设 *Cache* 的存取周期为 t , 主存的存取周期为 $5t$, 由 $H = 95\%$ 得系统的平均访问时间为: $T_a = 0.95 \times t + 0.05 \times 5t = 1.2t$

可知性能为原来的 $5t/1.2t = 4.17$ 倍, 即提高了 3.17 倍。(注: 题中问性能提高了多少)

思考: 若采用先访问 *Cache* 再访问主存的方式, 则提高的性能又是多少?

解析: 若为先访问 *Cache* 再访问主存, 则当 *Cache* 未命中时, 所消耗的时间为 *Cache* + 主存的时间

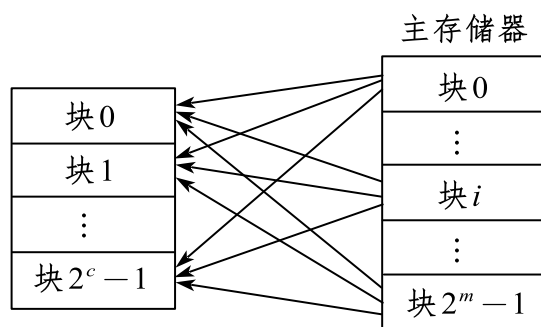
$$\therefore T_a = t + 0.05 \times 5t = 1.25t$$

性能为原来的 $\frac{5t}{1.25t} = 4$ 倍, 即提高了 3 倍。

2) *Cache* 和主存的映射方式

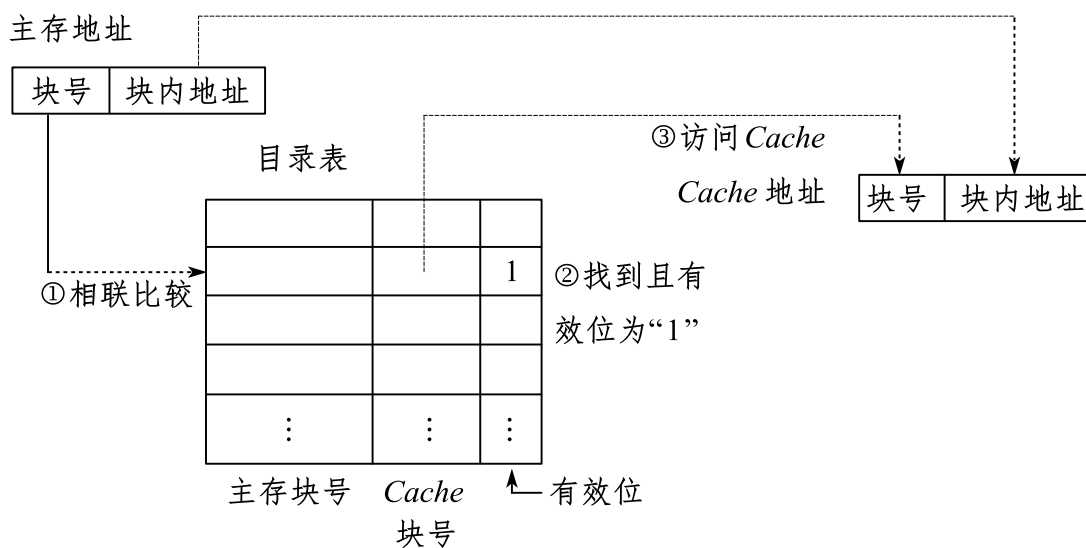
(1) 全相联映射及其地址变换

全相联地址映射是指主存中的每一块都可以映射到 *Cache* 中的任意块, 如下图所示。这种映射方法是最灵活的, 也是 *Cache* 利用率最高的一种方式, 但同时也是成本最高的一种方式。



全相联映射方式

在全相联映射方式下，主存地址被分为两个部分：高 m 位表示主存块地址，低 b 位表示块内地址。同样，Cache的地址也分为两个部分：高 c 位表示Cache块地址，低 b 位表示块内地址。通常采用目录表记录主存块之间的映射关系，并将目录表存放在一个相联存储器中。目录表中的每个存储字主要包括三个部分：主存块号、Cache块号和有效位。有效位表示目录表中主存块号和Cache块号建立的映射关系是否有效。目录表共有 2^c 个存储字，即Cache中每个块对应一个存储字。



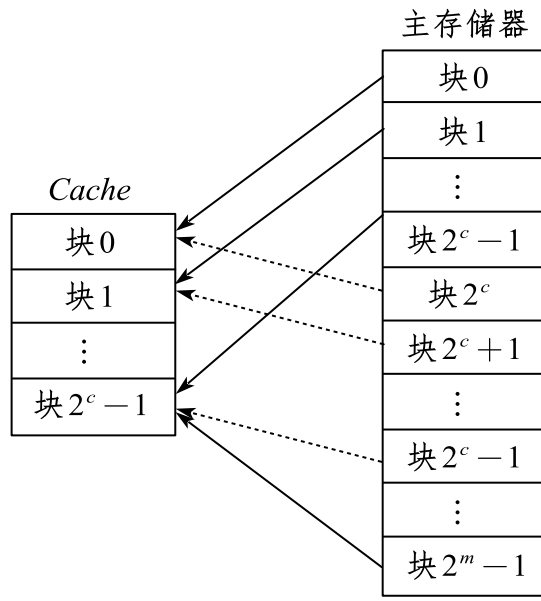
全相联映射的地址变换

(2) 直接映及其地址变换

直接地址映射是指主存中的块只能映射到Cache中某个固定的块中，主存和Cache块的对应关系可用如下公式表示：

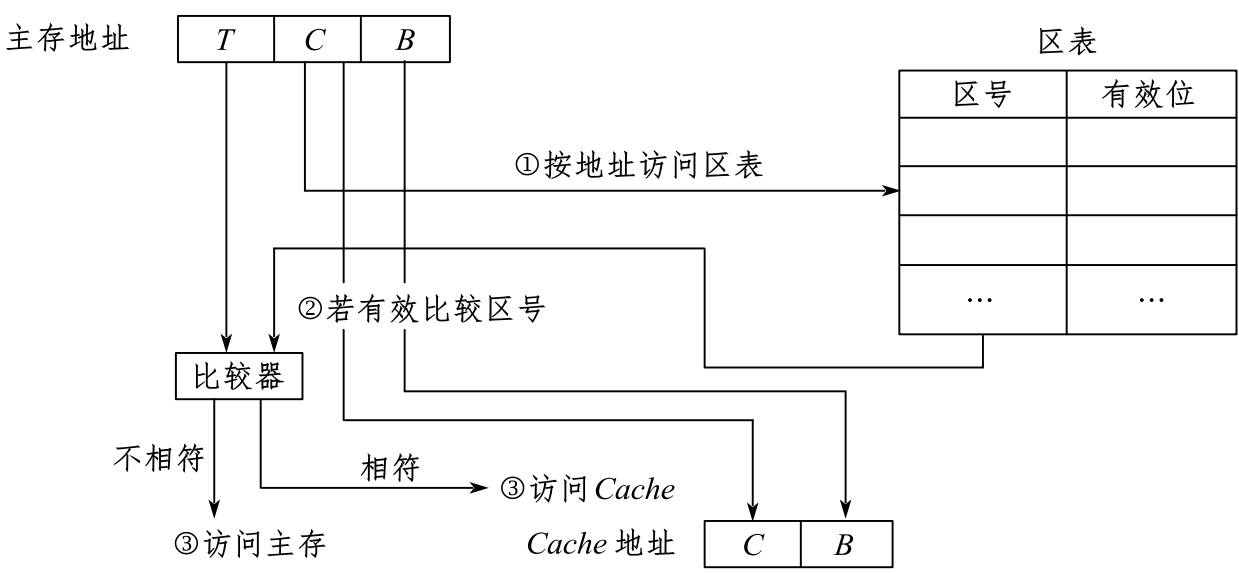
$$j = i \bmod 2^c$$

其中， j 为数据在Cache中的块号， i 为数据在主存中的块号。在这种映射方式中，主存的第0块，第 2^c 块，...只能映射到Cache的第0块，而主存的第1块，第 $2^c + 1$ 块，...只能映射到Cache的第1块，以此类推。如下图所示。



直接地址映射方式

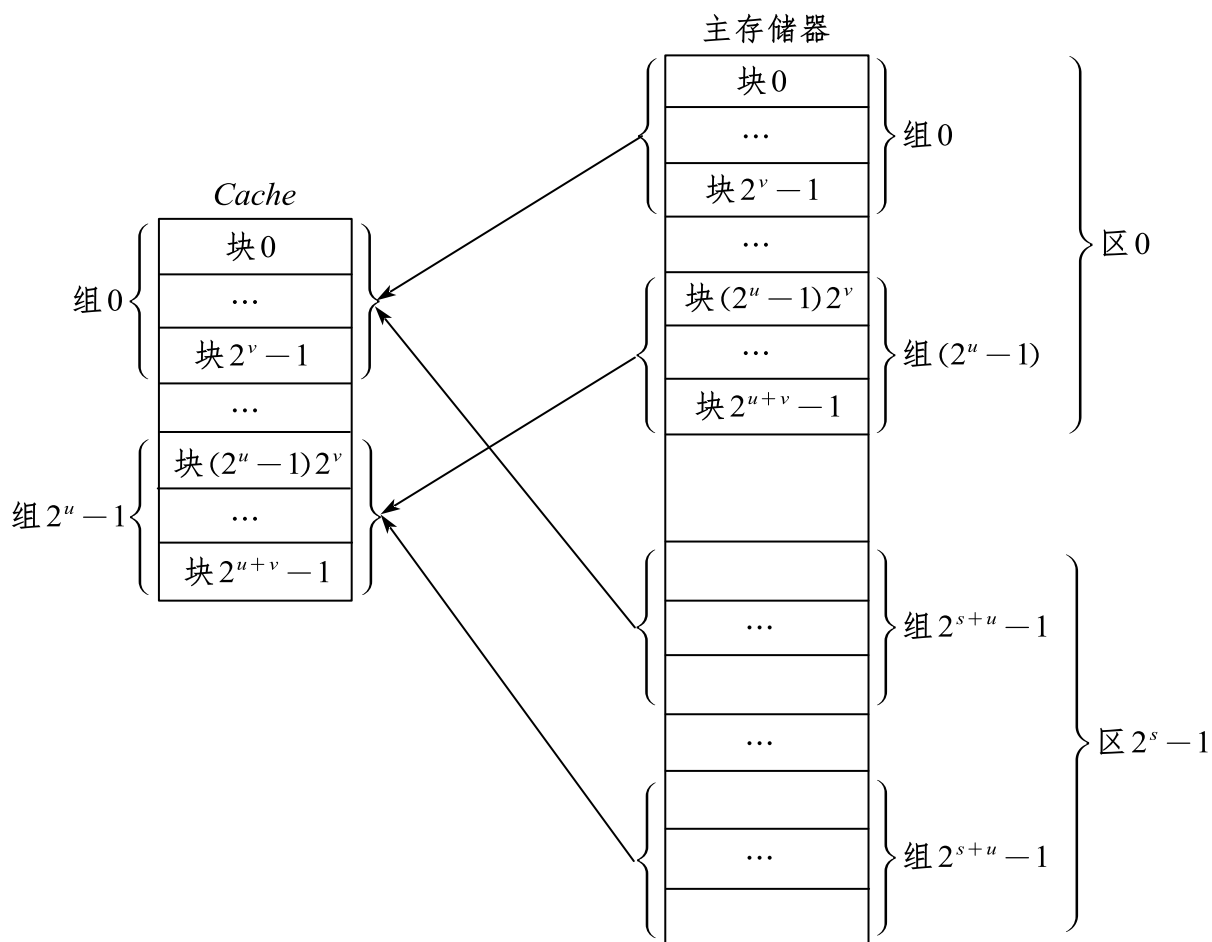
在直接地址映射方式下，主存地址由三部分组成：区号（ t 位）、区内块号（ c 位）和块内地址（ b 位），其中， $m = t + c$ 。通常用区表来保存主存块与 *Cache* 块的映射关系。区表中的每个存储字主要包括两个部分：主存区号和有效位。有效位表示区表中的主存块是否已经装入 *Cache* 中。区表中共有 2^c 个存储字。区表通常存放在一个小容量高速存储器中，按地址进行访问。



直接地址变换方法

(3) 组相联映射及其地址变换

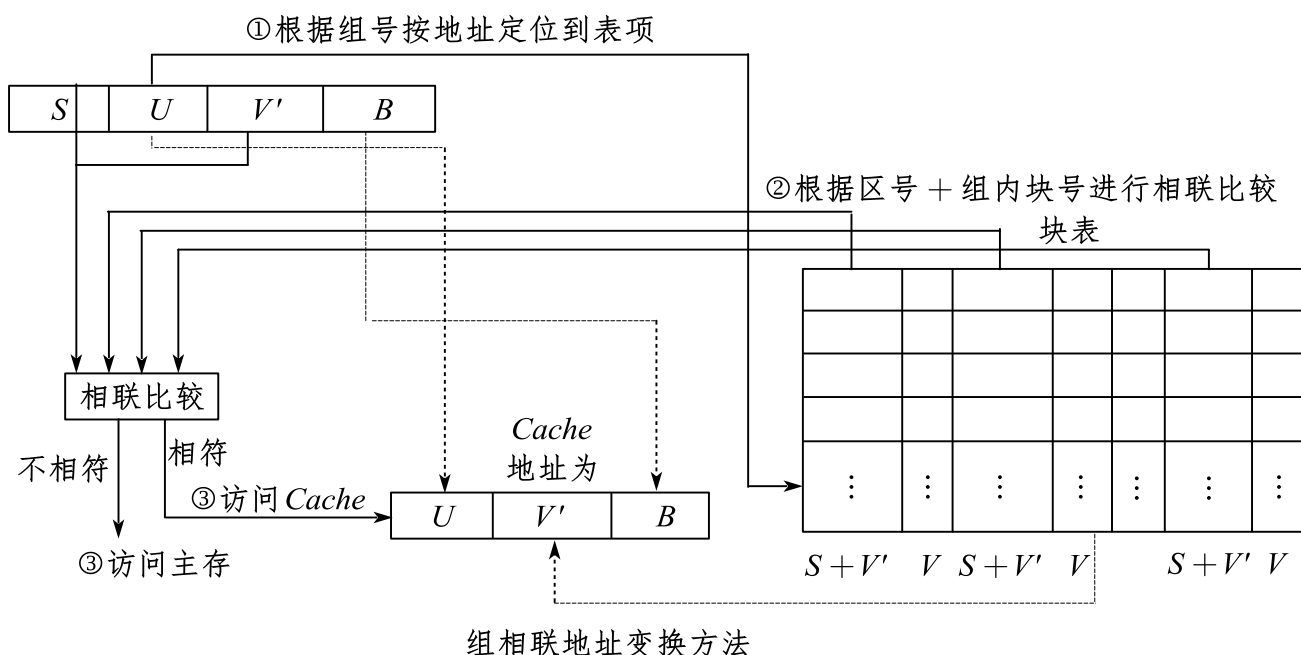
组相联映射 = 全相联映射 + 直接相联映射。如下图所示，主存和 *Cache* 的块先进行分组，（主存和 *Cache* 每组块数相同），在地址映射时，组间为直接相联映射，组内为全相联映射。



组相联映射方式

在上图中, *Cache* 被分为 2^u 组, 每组 2^v 块; 主存中共有 2^s 个区, 每个区有 2^u 个组, 即共有 2^{s+u} 个组。主存中的每个区的第 i 组都只能映射到 *Cache* 中的第 i 组, 在组内块采用全相联方式, 即每个块都可以映射到 *Cache* 的第 i 组的任一块。

CPU 发出的访存地址被分解为: 区号 S , 组号 U , 组内块号 V' 和块内地址 B 四个部分。而 *Cache* 的地址可分解为: 组号 U , 组内块号 V 和块内地址 B 三个部分。下图给出了组相联地址变换方式。



注： n 路组相联是指：每组有 n 块。

(4) 三种映射方式的对比

三种映射方式中，直接映射的每个主存块只能映射到 $Cache$ 中的某一固定行；全相联映射可以映射到所有 $Cache$ 行； N 路组相联映射可以映射到 N 行。当 $Cache$ 大小、主存块大小一定时，

①直接映射的命中率最低，全相联映射的命中率最高。

②直接映射的判断开销最小、所需时间最短，全相联映射的判断开销最大、所需时间最长。

③直接映射标记所占的额外空间开销最少，全相联映射标记所占的额外空间开销最大。

题 2. 主存与 $Cache$ 的地址映射有_____、_____和_____三种方式，其中_____的成本最高。

答案：直接映射，全相联映射，组相联映射；全相联映射

题 3. 设某机主存容量为 $4MB$ ， $Cache$ 容量为 $4096B$ ，字块长度为 8 字，字长 32 位，试使用直接映像，全相联映像和四路组相连映像（即 $Cache$ 每组内共有 4 个字块）三种方式的 $Cache$ 组织，要求：分别画出上述三种方式中主存地址字段的组成及各段位数。

答案：直接映射：

| | | |
|----|----|-------|
| 10 | 7 | 5 |
| 标记 | 块号 | 字块内地址 |

全相联

| | |
|----|-------|
| 17 | 5 |
| 标记 | 字块内地址 |

4路组相联:

| | | |
|----|-----|-------|
| 12 | 5 | 5 |
| 标记 | 组地址 | 字块内地址 |

解析: ①Cache 容量 $4096B = 2^{12}B$, Cache 字块地址12位: 块长 $8 \times 32b = 2^5B$, 即 $b = 5$ 且 Cache 块共 $4096/2^5 = 2^7$ 块, 即 $C = 7$; 主存容量 $4MB = 2^{22}B$, 主存主节地址为22位; 在直接映射下, 主存块标记为 $22 - 12 = 10$ 。

| | | |
|----|----|-------|
| 10 | 7 | 5 |
| 标记 | 块号 | 字块内地址 |

②全相连: 标记 $22 - 5 = 17b$

| | |
|----|-------|
| 17 | 5 |
| 标记 | 字块内地址 |

③组相联: 分组 $\frac{4096B}{8 \times 32b \times 4} = 2^5$ 组, 即占 $5b$, 标记 $22 - 5 - 5 = 12b$ 。

| | | |
|----|-----|-------|
| 12 | 5 | 5 |
| 标记 | 组地址 | 字块内地址 |

3) Cache 中主存的替换算法

常用的替换算法有随机 (RAND) 算法、先进先出 (FIFO) 算法、近期最少使用 (LRU) 算法和最不经常使用 (LFU) 算法。其中最常考查的是 LRU 算法。

(1) 随机算法: 随机地确定替换的 Cache 块。它的实现比较简单, 但未依据程序访问的局部性原理, 因此可能命中率较低。

(2) 先进先出算法: 选择最早调入的行进行替换。它比较容易实现, 但也未依据程序访问的局部性原理, 因为最早进入的主存块也可能是目前经常要用的。

(3) 近期最少使用算法 (LRU): 依据程序访问的局部性原理, 选择近期内长久未访问过的 Cache 行作为替换的行, 平均命中率要比 FIFO 的高, 是堆栈类算法。

LRU算法对每个Cache行设置一个计数器，用计数值来记录主存块的使用情况，并根据计数值选择淘汰某个块，计数值的位数与Cache组大小有关，2路时有一个LRU位，4路时有两个LRU位。假定采用四路组相联，有5个主存块{1, 2, 3, 4, 5}映射到Cache的同一组，对于主存访问序列{1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5}，采用LRU算法的替换过程如下图所示。图中左边阴影的数字是对应Cache行的计数值，右边数字是存放在该行中的主存块号。

| 1 | 2 | 3 | 4 | 1 | 2 | 5 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 2 | 1 | 3 | 1 | 0 | 1 | 1 | 1 |
| | 0 | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 0 | 2 | 1 |
| | | | 0 | 3 | 1 | 3 | 2 | 3 | 3 | 3 | 0 |
| | | | | 0 | 4 | 1 | 4 | 2 | 4 | 3 | 4 |

LRU算法的替换过程示意图

计数器的变化规则：

- ①命中时，所命中的行的计数器清零，比其低的计数器加1，其余不变；
- ②未命中且还有空闲行时，新装入的行的计数器置0，其余全加1；
- ③未命中且无空闲行时，计数值为3的行的信息块被淘汰，新装行的块的计数器置0，其余全加1。

当集中访问的存储区超过Cache组的大小时，命中率可能变得很低，如上例的访问序列变为1, 2, 3, 4, 5, 1, 2, 3, 4, 5, …，而Cache每组只有4行，那么命中率为0，这种现象称为抖动。

(4) 最不经常使用算法：将一段时间内被访问次数最少的存储行换出。每行也设置一个计数器，新建行后从0开始计数，每访问一次，被访问的行计数器加1，需要替换时比较各特定行的计数值，将计数值最小的行换出。这种算法与LRU类似，但不完全相同。

4) Cache写策略

因为Cache中内容为主存的副本，当对Cache内容更改时，需用写操作使Cache内容与主存内容一致，此时分两种情况：

(1) Cache写命中

- ①全写法：当CPU对Cache写命中时，必须把数据同时写入Cache和主存中。

优点：能随时保持主存数据正确性。

缺点：增加了访问次数，降低了 *Cache* 效率。

②回写法：当 *CPU* 对 *Cache* 写命中时，只把数据写入 *Cache*，而不立即写入主存中，只有当此块被换出时才写回主存。

优点：减少了访存次数。

缺点：存在数据不一致的隐患。

(2) *Cache* 写不命中

①写分配法：加载主存的块到 *Cache* 中，然后更新此 *Cache* 块。

②非写分配法：只写入主存，不进行调块。

注：非写分配法通常与全写法合用；写分配法通常与回写法合用。

题 4. 在 *Cache* 中，常用的替换策略有随机法 (*RAND*)、先进先出法 (*FIFO*)、近期最少使用法 (*LRU*)，其中与局部性原理有关的是 ()。

A. 随机法 (*RAND*)

B. 先进先出法 (*FIFO*)

C. 近期最少使用法 (*LRU*)

D. 都不是

答案：C

解析：*LRU* 算法根据程序访问局部性原理选择近期使用得最少的存储块作为替换的块。

题 5. 某虚拟存储器系统采用页式内存管理，使用 *LRU* 页面替换算法，考虑下面的页面访问地址流（每次访问在一个时间单位中完成）：1 8 1 7 8 2 7 2 1 8 3 8 2 1 3 1 7 1 3 7

假定内存容量为 4 个页面，开始时是空的，则页面失效率是 ()。

A. 30%

B. 5%

C. 1.5%

D. 15%

答案：A

解析：*LRU* 表如下：

| | | | | | | | | | | | | | | | | | | | | |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 单元 1 | | | | | | 2 | 7 | 2 | 1 | 8 | 3 | 8 | 2 | 1 | 3 | 1 | 7 | 1 | 3 | 7 |
| 单元 2 | | | | 7 | 8 | 8 | 2 | 7 | 2 | 1 | 8 | 3 | 8 | 2 | 1 | 3 | 1 | 7 | 1 | 3 |
| 单元 3 | | 8 | 1 | 1 | 7 | 7 | 8 | 8 | 7 | 2 | 1 | 1 | 3 | 8 | 2 | 2 | 3 | 3 | 7 | 1 |
| 单元 4 | 1 | 1 | 8 | 8 | 1 | 1 | 1 | 1 | 8 | 7 | 2 | 2 | 1 | 3 | 8 | 8 | 2 | 2 | 2 | 2 |
| 命中否 | 否 | 否 | | 否 | 否 | | | | | 否 | | | | | | 否 | | | | |

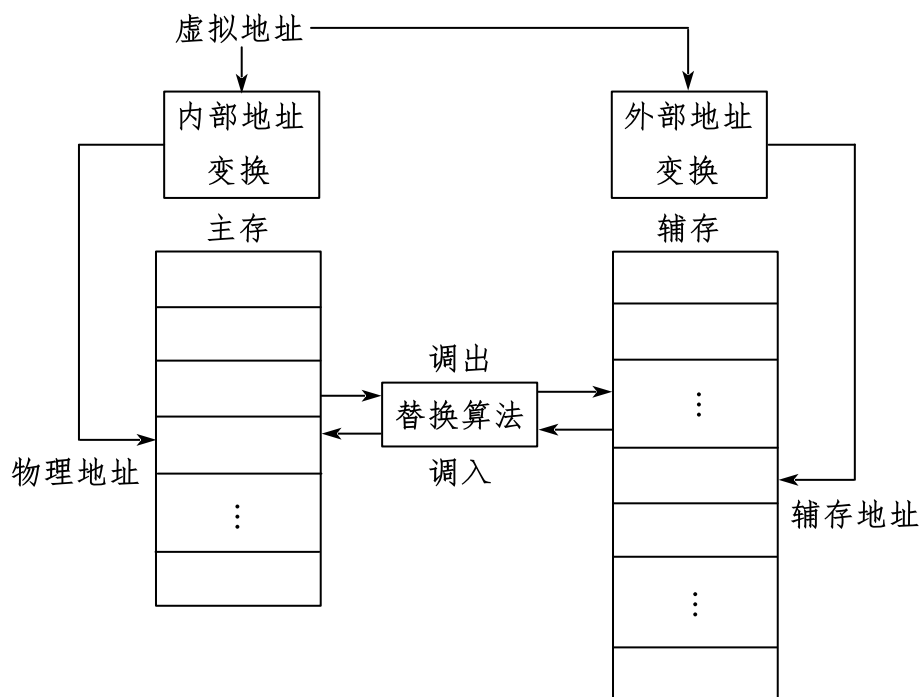
可见页面失效率是 $6/20 = 30\%$

3. 虚拟存储器

1) 基本原理

虚拟存储器技术是将一部分硬磁盘空间作为主存来使用。此时程序员使用的地址为虚拟地址（逻辑地址），对应空间也被称为虚拟地址空间（逻辑地址空间）；主存地址称为物理地址（实地址），对应空间称为物理地址空间（实地址空间）。

虚拟存储的基本思想是：在程序执行时，按照执行的顺序将程序的一部分调入主存，其他部分则保存在辅存中。当需要执行存放在辅存中的程序段时，由CPU按照某种调度算法以页、段为单位将其调入主存。



虚拟存储器的工作过程

2) 虚拟存储器与 *Cache* 的区别

| 区别 \ 存储层次 | 虚拟存储器 | 高速缓冲存储器 (<i>Cache</i>) |
|--------------|--|--|
| 设计目的 | 扩大容量 | 提高速度 |
| 实现手段 | 操作系统和硬件共同实现 | 硬件实现 |
| 透明性 | 对应用程序员透明, 对系统程序员不透明 | 对系统程序员和应用程序员都不透明 |
| 数据块大小 | 几十至几千字节 (可变) | 几个至几十字节 (固定) |
| 交换频率 (访问时间比) | 低 (一般100:1 至1000:1) | 高 (一般为10:1) |
| 数据通路 | 辅存与 <i>CPU</i> 之间不存在直接的数据通路, 当主存不命中时, 只能通过先将辅存的数据调入主存, 然后 <i>CPU</i> 对主存进行访问 | <i>CPU</i> 与 <i>Cache</i> 和主存之间均有直接访问通路, <i>Cache</i> 不命中时, 可以直接访问主存 |

题 1. 下列关于虚存的叙述中, 正确的是 ()。

- A. 对应用程序员透明, 对系统程序员不透明
- B. 对应用程序员不透明, 对系统程序员透明
- C. 对应用程序员、系统程序员都不透明
- D. 对应用程序员、系统程序员都透明

答案: A

解析: 虚存需要通过对操作系统实现地址映射, 因此对操作系统的设计者即系统程序员是不透明的。而应用程序员写的程序所使用的是逻辑地址 (虚地址), 因此对其是透明的。

题 2. 在一个页式虚拟存储器中, 假设有 8 个页面, 页面大小为 1024 字, 主存有 4 个页, 页表的内容如下所示, 问虚拟地址 4098 对应的主存地址是什么?

| 虚页号 | 实页号 |
|-----|-----|
| 0 | 3 |
| 1 | 1 |
| 2 | |
| 3 | |
| 4 | 2 |
| 5 | |
| 6 | 0 |
| 7 | |

答案：2050

解析：4098 \div 1024 = 4 余 2，因此虚页号为4，页内地址为2。从表中查得虚页号4对应的实页号为2，则实地址为 $1024 \times 2 + 2 = 2050$ 。

课时五 练习题

1. 高速缓冲存储器 *Cache* 一般采用 ()。
A. 随机存取方式 B. 顺序存取方式 C. 半顺序存取方式 D. 只读不写方式
2. *CPU* 执行一段程序时, *Cache* 完成存取的次数为1800次, 主存完成存取的次数为200。已知 *Cache* 存取周期为50ns, 主存的为250ns, 则 *Cache* / 主存系统的效率为 ()。
A. 0.625 B. 0.714 C. 0.375 D. 0.276
3. 采用虚拟存储器的主要目的是 ()。
A. 提高主存储器的存取速度
B. 扩大主存储器的存贮空间, 并能进行自动管理和调度
C. 提高外存储器的存取速度
D. 扩大外存储器的存贮空间
4. 直接映射 *Cache* 的主要优点是实现简单。这种方式的主要缺点是 ()。
A. 它比其他 *Cache* 映射方式价格更贵
B. 如果使用中的2个或多个块映射到 *Cache* 同一行, 命中率则下降
C. 它的存取时间大于其它 *Cache* 映射方式
D. *Cache* 中的块数随着主存容量增大而线性增加。
5. 如果 *Cache* 与主存之间采用的是组相联映射方式, 那么以下说法正确的是 ()。
A. 如果替换策略采用 *LRU* 算法, 那么 *Cache* 组内的行数越多则命中率越高
B. 如果替换策略采用 *FIFO* 算法, 那么 *Cache* 组内的行数越多则命中率越高
C. *Cache* 组的大小与命中率没有关系
D. 无论采用哪种算法, *Cache* 的组越大则命中率越高
6. 某计算机的 *Cache* 共有16块, 采用2路组相联映射方式 (即每组2块)。每个主存块大小为32B, 按字节编址。主存129号单元所在主存块应装入到的 *Cache* 组号是 ()。
A. 0 B. 1 C. 4 D. 6
7. 假设某计算机按字编址, *Cache* 有4个行, *Cache* 和主存之间交换的块大小为1个字。若 *Cache* 的内容初始为空, 采用2路组相联映射方式和 *LRU* 替换策略。访问的主存地址依次为0, 4, 8, 2, 0, 6, 8, 6, 4, 8时, 命中 *Cache* 的次数是 ()。
A. 1 B. 2 C. 3 D. 4

8. 设主存容量为 $1MB$ ， $Cache$ 容量为 $16KB$ ，每个字块有16个字，每字32位。

(1) 若 $Cache$ 采用直接相联映像，求出主存地址字段中各段位数。

(2) 若 $Cache$ 采用四路组相联映像，求出主存地址字段中各段位数。

课时六 指令系统

| 考点 | 重要程度 | 占分 | 常见题型 |
|---------|-------|-----|-------|
| 1. 概述 | ★★ | 2~4 | 选择、填空 |
| 2. 指令格式 | ★★★★ | 2~4 | 选择、填空 |
| 3. 寻址方式 | ★★★★★ | 4~8 | 选择、大题 |

1. 概述

1) 基本概念

(1) 指令系统是软件和硬件的接口，反映了计算机所具有的基本功能。

(2) 计算机的指令有宏指令、机器指令和微指令。宏指令是由若干条机器指令组成的软件指令，属于软件；微指令是微程序级的命令，属于硬件；机器指令（简称指令）介于微指令和宏指令之间。

2) 指令系统的发展

根据指令系统来分类计算机，主要有两类：复杂指令系统计算机（CISC）和精简指令系统计算机（RISC）

CISC 举例：IA-32

RISC 举例：MIPS 32

(1) CISC 的主要特点

①软件硬化：由于CPU访问主存的速度明显低于访问寄存器的速度，为了减少读取指令引起的大量主存访问。用一条功能复杂的新指令来取代原先需一串指令完成的功能。

②支持高级语言：当使用高级语言取代汇编语言后，增加新的复杂指令以及复杂的寻址方式来支持高级语言程序的高效实现。

③系统软件要求向上兼容和向后兼容，使得。指令系统不断扩大

④指令系统庞大，指令功能复杂，指令格式和寻址方式多样性，导致编译程序复杂，程序编译速度慢，特别是难以用优化编译技术生成高效的目标代码程序

⑤大多数指令功能复杂，且各种指令都可访问存储器，使得绝大多数指令需要多个机器周期才能完成。

⑥为了实现复杂的指令系统，通常采用微程序控制技术设计控制器，由微程序解释执行机器指令，也影响了指令的执行速度。

⑦由80%的指令只在20%的CPU运行时间内才被用到。

(2) RISC 的主要特点

RISC 的主要思想是只包含使用频率高的少量指令,并提供一些必要的指令以支持高级语言

①优先选取使用频率较高的简单指令,避免复杂指令。

②指令长度固定,指令格式种类少,寻址方式种类少。指令各字段的划分比较一致,各字段的功能也比较规整。

③只有取数/存数指令(*LOAD/STORE*)访问存储器,数据在寄存器和存储器之间传送。其余指令的操作都在寄存器之间进行。

④CPU种通用寄存器数量相当多,算术运算指令的操作数都在通用寄存器中存取。

⑤CPU采用流水线结构,大部分指令可以在一个机器周期(时钟周期)内完成。

⑥控制单元设计以硬布线控制逻辑为主,不用或少用微程序控制。

⑦采用编译优化技术,以减少程序执行时间。

3) 指令系统的功能

(1) 数据传送指令

功能:在寄存器、主存单元之间传送数据

举例:存储器读(*LOAD*)、存储器写(*STORE*)。

(2) 数据运算指令

①算术运算指令:加、减、乘、除等。

②逻辑运算指令:与、或、非等。

③移位指令:算术移位、逻辑移位、循环移位。

④位操作指令:位清除、位求反等。

注: a. *CISC* 指令系统支持源操作或目的操作数在内存单元的运算类指令:

b. *RISC* 指令系统只有存数(*LOAD*)和取数(*STORE*)可以访问内存单元,运算类指令操作数均在寄存器中。

(3) 程序控制指令(转移指令)

①无条件转移:不受任何条件约束,可直接把程序转移到某一条需要执行的指令。

②条件转移:根据当前指令的执行结果判断是否转移。

a. 条件满足(如零标志位(*Z*),结果为0, $Z=1$),则转移

b. 条件不满足,则继续顺序执行

③调用与返回。即调用子程序指令(*CALL*)和返回原程序断点指令(*RETURN*)。

注:调用子程序时返回地址可存放在:

a. 寄存器内：机器内设有主用寄存器，专门用来放返回地址。

特点：此方法难以支持子程序嵌套。

b. 子程序的入口单元内：将返回地址存入子程序的第一个单元，然后转到第二个单元开始执行子程序。

特点：此方法支持子程序嵌套，但无法支持子程序的递归调用（即子程序调用它本身）

c. 堆栈栈顶：调用子程序时（*CALL*）将返回地址压入堆栈，子程序返回时（*RETURN*），可自动从栈顶取出相应的返回地址。

特点：此方法可实现递归调用。

题 1. 下列关于 *RISC* 的说法中，错误的是（ ）

A. *RISC* 普遍采用微程序控制器

B. *RISC* 大多数指令在一个时钟周期内完成

C. *RISC* 的内部通用寄存器大数量相对 *CISC* 多

D. *RISC* 的指令数、寻址方式和指令格式种类相对 *CISC* 少

答案：A

解析：相对于 *CISC*，*RISC* 的特点是：指令条数少；指令长度固定，指令格式和寻址种类少；

只有取数/存数指令访问存储器，其余指令的操作均在寄存器之间进行；*CPU* 中通用寄存器多；大部分指令在一个或小于一个机器周期内完成；以硬布线逻辑为主，不用或少用微程序控制。选项 B、C、D 都是 *RISC* 的，选项 A 是错误的，因为 *RISC* 的速度快，所以普遍采用硬布线控制器，而非微程序控制器。

题 2. 下面描述的 *RISC* 机器基本概念中正确的句子是（ ）

A. *RISC* 机器不一定是流水 *CPU*

B. *RISC* 机器一定是流水 *CPU*

C. *RISC* 机器有复杂的指令系统

D. *CPU* 配置很少的通用寄存器

答案：B

解析：*RISC* 指令长度固定，采用流水线技术是为了提高 *CPU* 的性能。

2. 指令格式

1) 指令的组成

机器指令的基本格式如下：

| | |
|----------|----------|
| 操作码 (OP) | 地址码 (Ad) |
|----------|----------|

(1) 操作码：指出指令的操作性质，即指令完成的功能。

(2) 地址码：指出操作数的地址，即操作对象所在位置，在主存储中的地址。

2) 指令分类

按指令中地址码字段的个数分类：三地址指令，二地址指令，一地址指令。

(1) 三地址指令，指令格式：

| | | | |
|----|-------|-------|-------|
| OP | A_1 | A_2 | A_3 |
|----|-------|-------|-------|

$A_3 \leftarrow (A_1)OP(A_2)$ ， A_1 、 A_2 为源操作数地址， A_3 为目的操作数地址。

特点：指令执行后，两个源操作数的内容不变：指令字长较长，难以支撑 SS 型指令。

(2) 二地址指令：

| | | |
|----|-------|-------|
| OP | A_1 | A_2 |
|----|-------|-------|

$A_1 \leftarrow (A_1)OP(A_2)$ ， A_1 既是源操作数地址也是目的操作数地址： A_2 为另一个源操作数地址。

特点：

① 二地址指令和三地址指令能完成同样的数据处理操作。

② 二地址指令用过减少一个地址码字段，可以减少指令字长度或增加地址码字段位数或为操作码扩展提供支持。

③ 二地址指令执行后一个源操作数内容会被替代，影响了程序的灵活性。

(3) 一地址指令：在二地址指令格式基础上，将一个地址设定到专用寄存器中（通常为 AC），便形成了一地址指令格式。

| | |
|----|-------|
| OP | A_1 |
|----|-------|

$AC \leftarrow (AC) OP(A)$

特点：一地址指令字长进一步缩短；

执行该指令的前提是一个操作数已存入特定位置，如 AC 中。

(4) 零地址指令：OP

例如：堆栈机类，空操作（*NOP*），停机（*HLT*）等指令。

3) 指令字长

(1) 等长指令字结构：指令系统中所有指令字长均相等，通常为机器字长

特点：指令的读取和分析的硬件结构简单

举例：*MIPS32* 采用单字长的等长指令字结构，指令字长为32位。

(2) 变长指令字结构：各种指令长度不等，如：半字长，单字长，双字长。

特点：结构灵活，能充分利用指令信息位，但指令读取控制及分析的逻辑复杂

举例：*LA-32* 采用变长指令字结构，指令长度取1~16倍字节。

4) 操作码扩展技术

(1) 定长操作码

优点：硬件设计，指令译码结构简单，译码时间短；

缺点：浪费了许多信息位

举例：*RISC* 普遍为此结构，*MIPS32* 操作码固定位6位。

(2) 变长操作码

等长扩展法（保留一个码点/状态、标志）

以4-8-12扩展法为例：

| 指令格式及操作码编码 | | | | 说明 |
|------------|------|------|------|-------------------|
| OP_Code | $A1$ | $A2$ | $A3$ | 4 位操作码的三地址指令共15 条 |
| 0000 | | | | |
| 0001 | | | | |
| ... | | | | |
| 1110 | | | | |
| OP_Code | $A1$ | $A2$ | | 8 位操作码的二地址指令共15 条 |
| 1111 | 0000 | | | |
| 1111 | 0001 | | | |
| ... | ... | | | |
| 1111 | 1110 | | | |

| OP_Code | $A1$ | | 12 位操作码的一地址指令共15 条 |
|------------|------|-----------|--------------------|
| 1111 | 1111 | 0000 | |
| 1111 | 1111 | 0001 | |
| ... | ... | ... | |
| 1111 | 1111 | 1110 | |
| OP_Code | | | 16 位操作码的零地址指令共16 条 |
| 1111 | 1111 | 1111 0000 | |
| 1111 | 1111 | 1111 0001 | |
| ... | ... | ... | |
| 1111 | 1111 | 1111 1110 | |

题 3. 设某机器指令字长为16 位，每个地址码长为4 位，用扩展操作码方法设计指令格式。若该指令系统中已定义了12 条三地址指令，62 条二地址指令，30 条一地址指令，则该指令系统最多可以有（ ）条零地址指令

答案：32

解析：此类题无特殊说明，都是指上文介绍的方法

| | | | |
|------|-------|-------|-------|
| OP | A_1 | A_2 | A_3 |
|------|-------|-------|-------|

三地址指令剩余的码点状态： $2^4 - 12 = 4$ 种

二地址指令剩余的码点状态： $4 \times 2^4 - 62 = 2$ 种

一地址指令剩余的码点状态： $2 \times 2^4 - 30 = 2$ 种

零地址指令剩余的码点状态： $2 \times 2^4 = 32$ 条

3. 寻址方式

1) 基本概念：

- (1) 形式地址（符号地址）：指令中显式给出的地址
- (2) 有效地址（逻辑地址）：操作数或指令实际的地址
- (3) 有效地址（ EA ）：由寻址方式和形式地址共同确定

2) 指令寻址

(1) 顺序寻址

不需要在指令中显式给下一条指令地址的信息，而是通过程序计数器（PC）加“1”，自动形成下一条指令的地址。

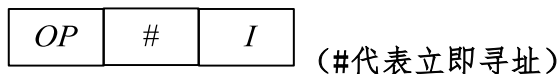
注：此处加“1”并不是真正的加1，而是加 n （ n 为正数，取决于当前指令的字长）。

(2) 跳跃寻址

由转移类指令给出下一条指令的地址信息（寻址方式和形式地址），类似数据寻址多种寻址方式。

3) 数据寻址：根据数据所在位置分为：

(1) 立即寻址：操作数直接在指令的地址码字段给出。

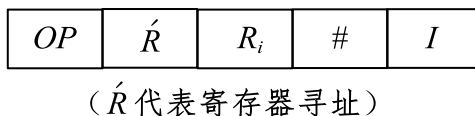


特点：①形式地址 I 不是操作数的地址，而是操作数本身

②有效地址（EA）为该指令在主存单元的地址，

③执行阶段不必访存，执行速度快。

(2) 寄存器寻址：源操作数已经在通用寄存器中，或者操作结果要存回寄存器中。



特点：① $EA = R_i$ （ R_i 为通用寄存器编号）

②寄存器寻址方式执行过程中也无需访问，执行时间短

③寄存器编号位数小，指令字长短，节省了存储空间

(3) 存储器寻址

数据存储在存储器中，根据有效地址的不同形成方式，分为：

①直接寻址：指令中地址码字段给出的形式地址 A 就是操作数或下一条指令的有效地址。



特点：

a. $EA = A$

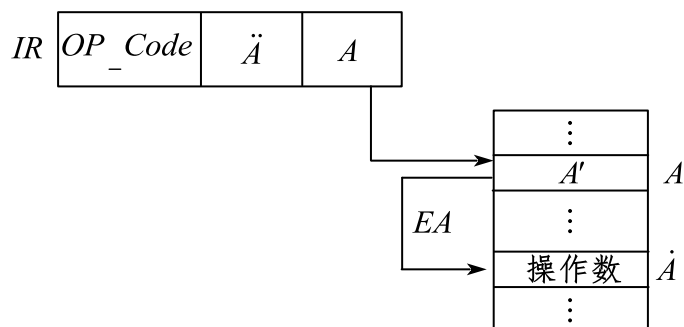
b. 获取有效地址比较简单

c. 指令执行期间，需访问一次主存

d. 指令字长限制了形式地址字段的位数，可寻址范围较小

②间接寻址（存储器间接寻址）

指令中的形式地址是操作数地址的地址，需访问两次或两次以上主存才能得到操作数（或指令）



一次间接寻址

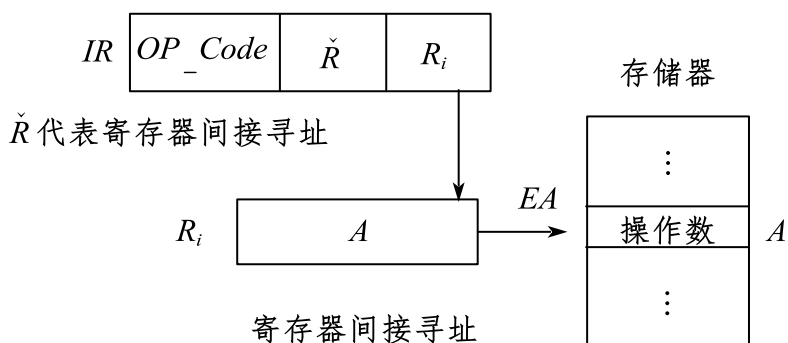
特点：

a. 一次简介是， $EA = (A)$ ；

b. 寻址范围取决于主存字长 m ，一次间接寻址最大范围是 2^m 字

c. 简介寻址为程序设计提供很好的灵活性，但要多次访存，指令执行速度较慢

③寄存器间接寻址：形式地址给出的是通用寄存器的编号为 R



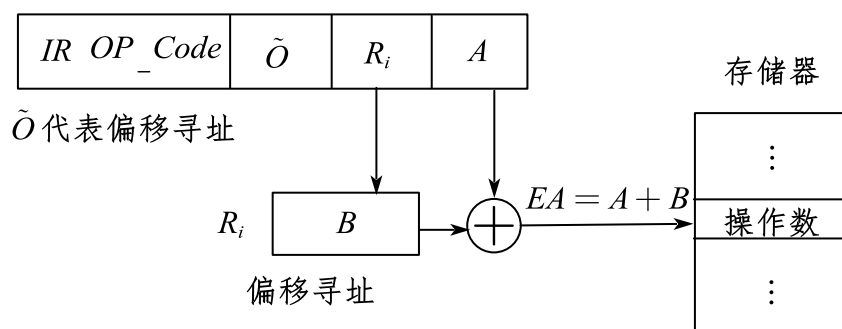
特点：

a. $EA = (R_i)$

b. 寄存器间接寻址的寻址范围取决于寄存器的位数（即机器字长 n ）

c. MIPS 及其受流水线结构限制，其指令系统不支持间接寻址方式

④偏移寻址：直接寻址和寄存器间接寻址的结合



特点：偏移寻址方式既要给出形式地址 A （偏移量），也要指出引用哪一个寄存器 R_i 的内容实现偏移，即 $EA = (R_i) + A$ 。（此外 R 可是用寄存器也可可是通用寄存器， A 为有符号整数（通常用补码表示））。

相对寻址：

特点：

a. 使用专门的程序计数器（ PC ）

b. $EA = (PC) + A$

c. 寻址范围取决于偏移量 A 的位数 k ，即 $(PC) - 2^{k-1} \sim (PC) + 2^{k-1} - 1$

d. 偏移量 A 为补码：首位为 1，代表负数，进行符号扩展时，高位补 1；首位为 0，代表正数，进行符号扩展时，高位补 0

e. 偏移量 A 是相对于该条指令的下一条指令而言（因为 PC 自动加 1）

f. 只要操作数或下一条指令与当前指令的相对距离不变，无论程序放在主存中哪段区域都可正确执行，既有利于程序在内存中浮动。

变址寻址：

特点：

a. 采用专用变址寄存器（ R_x ）或通用寄存器（ R_i ）

b. $EA = (R_x) + A$ 或 $EA = (R_i) + A$

c. 变址寻址常用于数组或字符串操作，即，寄存器中存放的地址为修改量（变址量），形式地址 A 为基本地址值（起始地址，用无符号整数表示）

d. 变址寻址完成后，变址寄存器的内容会自动调整，即 $R_x = (R_x) + \Delta$

e. 变址寻址的地址范围由贬值寄存器的位数 n 决定，即最大寻址空间为 2^n 个字

基址寻址：

特点：

a. 基址寄存器(R_b)为专用或通用的

b. $EA = (R_b) + A$

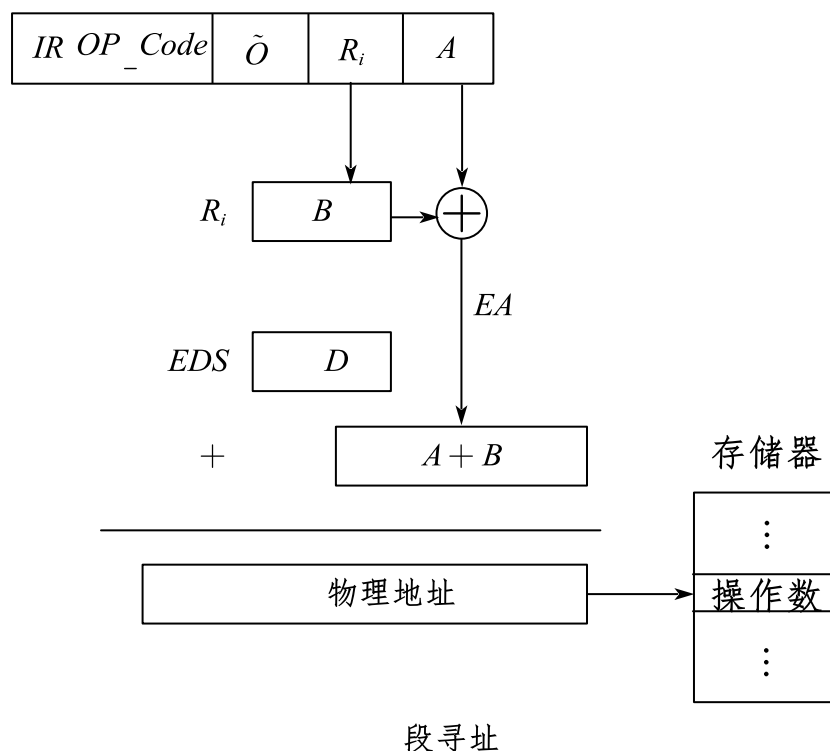
c. 基址寻址通常用于实现操作系统对用户程序的动态定位

d. 基址寄存器中存放的是基本地址值(基地址), 一般不可修改, 偏移量 A 通常为正向偏移, 即 A 为无符号整数

e. 寻址范围由偏移量的位数决定

⑤段寻址: 基址寻址的一种特例, 用于地址长度超过及其字长的场合

基本思想: 将主存空间在逻辑上划分为若干段, 一个程序可以占用多段



段寻址

特点:

a. 与机器字长相等的段地址和段内偏移量错位相加, 以获得更长的存储器地址

b. 段地址放在专用的段地址寄存器 R_s 中, 整个段寻址过程中由硬件自动完成, 对用户是透明的

c. 段寻址方式中的段内偏移量就是其他寻址方式形成的有效地址 EA , 段寻址后形成的地址就是实际的主存单元地址, 即物理地址

(4) 堆栈寻址

存储器堆栈实在主存中, 开辟一块区域, 该区域一段圆定, 称为栈底, 另一端浮动, 称为栈顶。栈顶是数据唯一的出入口

堆栈指针（ SP ）始终指向栈顶，依据“先进后出（ $FILO$ ）”的原则存储数据

栈底与栈顶的地址由两种设定方法：

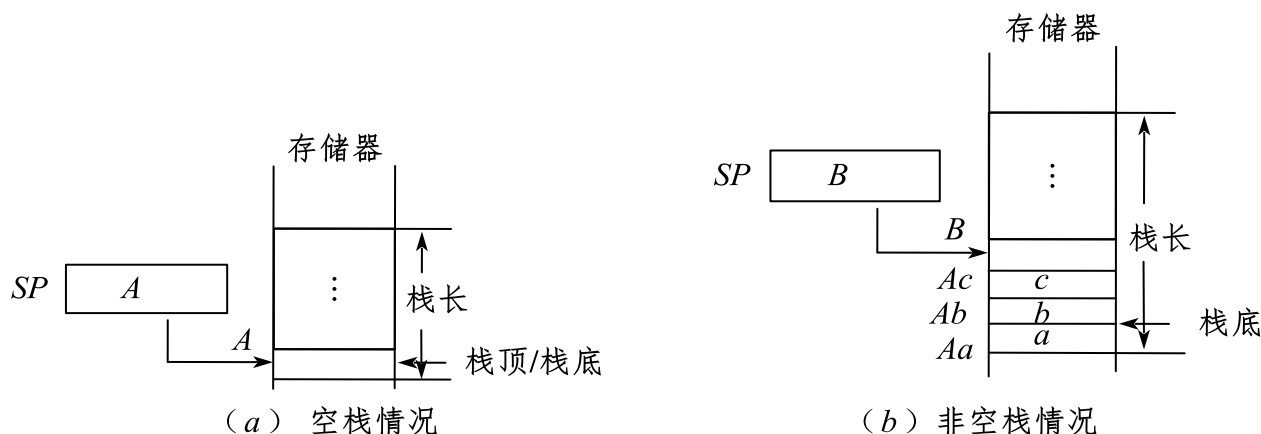
①栈底设在堆栈区域的低地址端，栈顶设在高地址端，堆栈向上生长；

②栈底设在堆栈区域的高地址端，栈顶设在低地址端，堆栈向下生长

存储器堆栈主要有两种实现方法：

① SP 指向栈顶的一个空单元

当堆栈为空时，栈底和栈顶为同一单元，如图（ a ）所示。入栈时将数据存入栈顶，并将 SP 指向栈顶空单元，如图（ b ）所示



SP 指向栈顶空单元

入栈（ $PUSH$ ）时，硬件完成的具体操作包括：

$(SP) \leftarrow \text{数据}$

$SP \leftarrow (SP) + 1$ （向上生长），或者 $SP \leftarrow (SP) - 1$ （向下生长）

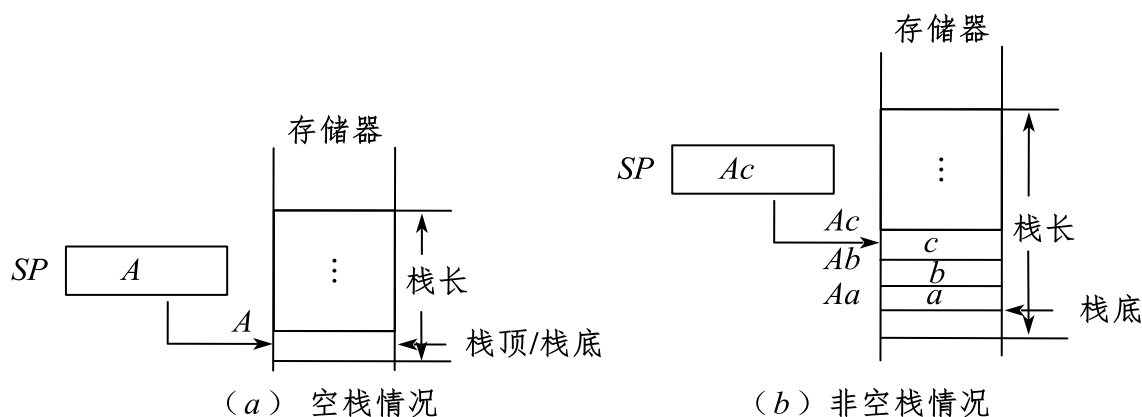
出栈（ POP ）时硬件完成的具体操作包括：

$SP \leftarrow (SP) - 1$ （向上生长），或者 $SP \leftarrow (SP) + 1$ （向下生长）

$[(SP)]$ 出栈

② SP 指向栈顶的一个非空单元

当堆栈为空时，栈底和栈顶为同一个单元，如图（ a ）所示，入栈时，将数据存入栈顶，并将 SP 指向栈顶非空单元，如图 b 所示



SP 指向栈顶非空单元

入栈 ($PUSH$) 时, 硬件完成的具体操作包括:

$SP \leftarrow (SP) + 1$ (向上生长), 或者 $SP \leftarrow (SP) - 1$ (向下生长)

$(SP) \leftarrow \text{数据}$

出栈 (POP) 时硬件完成的具体操作包括:

$[(SP)]$ 出栈

$SP \leftarrow (SP) - 1$ (向上生长), 或者 $SP \leftarrow (SP) + 1$ (向下生长)

(5) 复合寻址

将两种以上寻址方式联合起来使用。

注: 复合寻址要解决的关键问题是地址的计算问题, 一般习惯于从名称上加以反映。

举例:

①变址间接寻址: 先变址后间接的寻址顺序: $EA = [(R_x) + A]$, 且 $R_x = (R_x) + \Delta$

②间接变址寻址: 先间接后变址的寻址顺序: $EA = (R_x) + (A)$, 且 $R_x = (R_x) + \Delta$

题 4. 对某个寄存器中操作数的寻址方式称为 () 寻址

A. 直接 B. 间接 C. 寄存器 D. 寄存器间接

答案: C

题 5. 偏移寻址通过将某个寄存器内容与一个形式地址相加而生成有效地址。下列寻址方式中, 不属于偏移寻址方式的是 ()

A. 间接寻址 B. 基址寻址 C. 相对寻址 D. 变址寻址

答案: A

题 6. 指令系统中采用不同的寻址方式的主要目的是 ()

I 简化指令译码电路

II 缩短指令长度，扩大寻址范围

III 提高访问内存的速度

IV 提高程序的灵活性

V 实现程序控制

VI 增加指令系统中的指令数量

A. I、II、VI

B. I、III

C. IV、V、VI

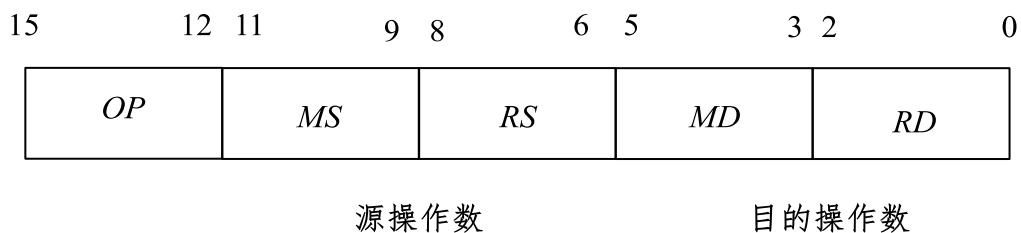
D. II、IV

答案：*D*

解析：主要目的：扩大寻址空间，提高编程灵活性

课时六 练习题

- 下面哪一个不是 *RISC* 的特点 ()
 - 指令长度固定, 指令格式和寻址方式的种类少
 - 只有存数/取数两条指令访问存储器, 其他指令均访问寄存器
 - CPU* 中通用寄存器数量多
 - 为提高执行速度, 尽量采用微程序控制实现指令
- 某机器的指令长度为 12 位, 包括 4 位基本操作码字段和 2 个 4 位地址字段, 则炒菜用可变长操作码时, 二地址指令, 一地址指令, 领地址指令可能存在的条数依次为: ()
 - 15, 16, 16
 - 14, 32, 16
 - 15, 14, 32
 - 15, 15, 28
- 某机器字长为 16 位, 主存按字节编址, 转移指令采用相对寻址, 由 2 个字节组成, 第一个字节为操作码字段, 第二个字节为相对移位量字段。假定取指令时, 每取一个字节 *PC* 自动加 1。若某转移指令所在主存地址为 $2000H$, 相对位移量字段的内容为 $06H$, 则该转移指令成功转以后的目标地址为 ()
 - $2006H$
 - $2007H$
 - $2008H$
 - $2009H$
- 直接寻址方式中, 操作数在 () 中。
 - 指令
 - 寄存器
 - 内存单元
 - 硬盘
- 寄存器间接寻址方式中, 操作数在 () 中。
 - 指令
 - 寄存器
 - 内存单元
 - 硬盘
- 某计算机字长为 16 位, 贮存地址空间大纤维 $128KB$, 按字编址。采用单字长格式, 每个地址都标记了寻址方式。指令各字段定义如下:



部分寻址方式的编码及含义如下:

| <i>MS/MD</i> | 寻址方式 | 助记符 | 含义 |
|--------------|-------|-------|---------------|
| $(000)_2$ | 寄存器直接 | R_n | 操作数 = (R_n) |

| | | | |
|-----------|-------|---------|-----------------|
| $(001)_2$ | 寄存器间接 | (R_n) | 操作数 $= ((R_n))$ |
|-----------|-------|---------|-----------------|

(1) 该指令系统最多可有多少指令？该计算机最多有多少个通用寄存器？

(2) 主存地址寄存器 (MAR) 和贮存期数据寄存器 (MDR) 至少各需要多少位？

(3) 若操作码 $(0010)_2$ 表示加法操作 (助记符为 ADD)，寄存器 $R4$ 和 $R5$ 的编号分别为 $(100)_2$ 和 $(101)_2$ ，则汇编语句 “ $ADD(R4), R5$ ” 对应的机器码是什么？（注：结果用十六进制表示。该指令逗号前的属于源操作数，逗号后的为目的操作数）

7. 某机器存储字长、指令字长和及其字长均为 16 位，指令格式如下：

| | | |
|------|-----|-----|
| 5 | 3 | 8 |
| OP | M | D |

其中， D 为形式地址，补码表示（包括一位符号位）：

M 为寻址模式：

$M=0$ ：立即寻址；

$M=1$ ：直接寻址（此时 D 视为无符号数）

$M=2$ ：间接寻址（此时 D 视为无符号数）

$M=3$ ：变址寻址（变址寄存器为 R_x ）

$M=4$ ：相对寻址

(1) 写出各种寻址模式计算有效地址的表达式：

(2) 当 $M=1、2、4$ 时，能访问的最大主存区为多少机器字（主存容量为 $64K$ 字）

课时七 数据通路

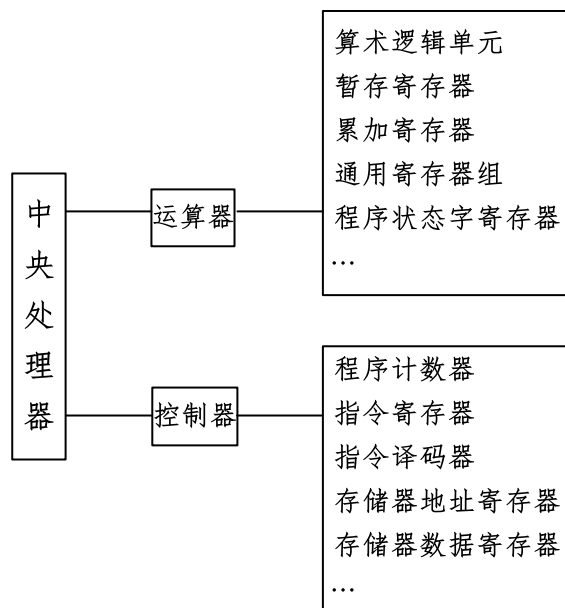
| 考点 | 重要程度 | 占分 | 题型 |
|-----------|-------|-------|-------|
| 1. CPU 概述 | ★★ | 2 ~ 4 | 选填、填空 |
| 2. 数据通路 | ★★★★★ | 4 ~ 8 | 选填、大题 |
| 3. 指令流水线 | ★★★★★ | 4 ~ 8 | 选填、大题 |

1. CPU 概述

1) CPU 的功能

- ①指令控制: 即程序的顺序控制, 包括取指令、分析指令、执行指令操作。
- ②操作控制: 指令的执行由多个操作信号控制, 这些操作信号即由 CPU 产生。
- ③时间控制: 对各种操作加以时间控制。
- ④数据加工: 数据的算术运算和逻辑运算。
- ⑤中断处理: 处理计算机允许时发生的异常及特殊情况。

2) CPU 的结构



中央处理器的组成

(1) 运算器: 计算机对数据进行加工处理的中心。

- ①算术逻辑单元 (ALU): 运算器的核心部件, 主要功能是进行算术/逻辑运算。
- ②暂存寄存器: 暂存从主存读来的数据, 对应用程序员是透明的。
- ③累加寄存器: 是一个通用寄存器, 可作为加法运算的一个输入端, 也可用于暂存 ALU 运算的结果。

④通用寄存器组：用于存放操作数和各种地址信息。

⑤程序状态字寄存器（*PSW*）：保留由算术逻辑运算指令或测试指令的结果而建立的各种状态信息，如溢出标志（*OF*）、符号标志（*SF*）、零标志（*ZF*）、进位标志（*CF*）等。

⑥移位器：对操作数或运算结果进行移位运算。

⑦计数器：控制乘除运算的操作步数。

（2）控制器：基本功能为通过一系列微操作来执行指令

①程序计数器。用于指出下一条指令在主存中的存放地址。*CPU*根据*PC*的内容去主存中取指令。因程序中指令（通常）是顺序执行的，所以*PC*有自增功能。

②指令寄存器。用于保存当前正在执行的那条指令。

③指令译码器。仅对操作码字段进行译码，向控制器提供特定的操作信号。

④存储器地址寄存器。用于存放要访问的主存单元的地址。

⑤存储器数据寄存器。用于存放向主存写入的信息或从主存读出的信息。

⑥时序系统。用于产生各种时序信号，它们都由统一时钟（*CLOCK*）分频得到。

⑦微操作信号发生器。根据*IR*的内容（指令）、*PSW*的内容（状态信息）及时序信号，产生控制整个计算机系统所需的各种控制信号，其结构有组合逻辑型和存储逻辑型两种。

注：对用户透明的寄存器：*MAR*、*MDR*、*IR*

对用户不透明的寄存器：通用寄存器组、程序状态字寄存器

题 1. *CPU*从主存取指令时，（ ）。

A. 总是根据程序计数器*PC*得到主存地址。

B. 有时根据*PC*得到主存地址，有时根据转移指令得到主存地址。

C. 总是根据指令寄存器得到主存地址。

D. 有时根据*PC*得到主存地址，有时根据指令寄存器得到主存地址。

答案：A

解析：转移指令也是通过改变*PC*的值来进行转移。

题 2. 下列寄存器中，汇编语言程序员可见的是（ ）。

A. 存储器地址寄存器（*MAR*） B. 程序计数器（*PC*）

C. 存储器数据寄存器（*MDR*） D. 指令寄存器（*IR*）

答案：B

解析：汇编语言程序员可见的为 PC ，因为其可以通过汇编程序对某个寄存器进行访问，而 IR 、 MAR 、 MDR 为 CPU 内部工作寄存器，对程序员不可见。

2. 数据通路

指令执行过程

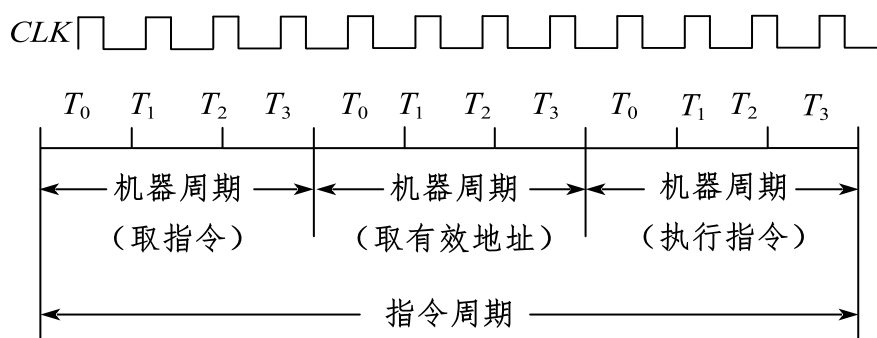
1) 指令周期

(1) 定义： CPU 从主存中取出并执行一条指令的时间。

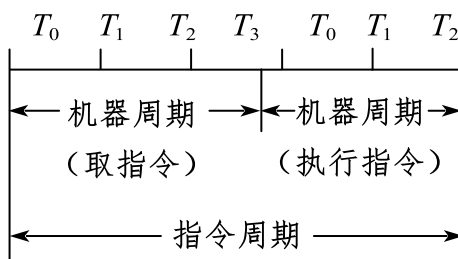
(2) 特点：

①指令周期通常用若干机器周期表示，一个机器周期又包含若干时钟周期(节拍或 T 周期，为 CPU 操作的最基本单位)

②每个指令周期内的机器周期数可以不等，每个机器周期内的节拍也可以不等。



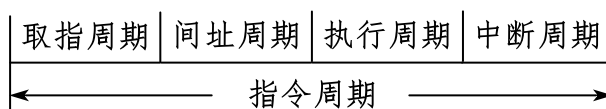
(a) 定长的机器周期



(b) 不定长的机器周期

指令周期、机器周期、节拍和时钟周期的关系

③完整的指令周期包括：取指、间址、执行、中断4个周期。



带有间址周期、中断周期的指令周期

取指周期：取指令

间址周期：取有效地址

执行周期：取操作数

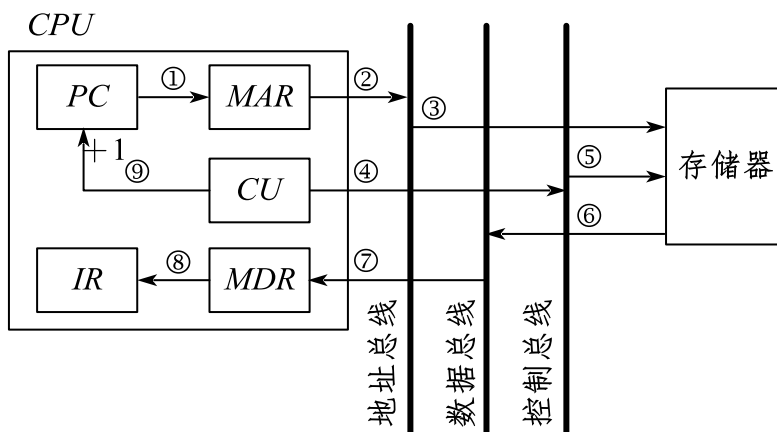
中断周期：保存程序断点

④CPU通过4个标志触发器 FE 、 IND 、 EX 和 INT 来区分取指、间址、执行、中断周期。

2) 指令周期的数据流

(1) 定义：即在指令执行的不同阶段，根据指令要求依次访问的数据序列。

(2) 取指周期：根据 PC 中的地址从主存中取出指令代码并存放于 IR 中。



取指周期的数据流

数据流：

① $PC \rightarrow MAR$ ② 地址总线 ③ 主存。

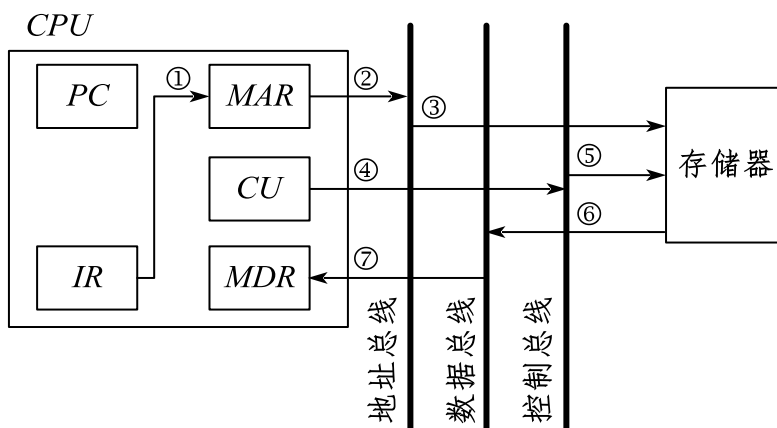
② CU 发出读命令 ④ 控制总线 ⑤ 主存。

③ 主存 ⑥ 数据总线 ⑦ MDR ⑧ IR (存放指令)。

④ CU 发出控制信号 ⑨ PC 内容加1。

(3) 间址周期

取操作数的有效地址。



一次间址周期的数据流

数据流：

① $Ad(IR)$ (或 MDR) ① MAR ② 地址总线 ③ 主存。

② CU 发出读命令 ④ 控制总线 ⑤ 主存。

③ 主存 ⑥ 数据总线 ⑦ MDR (存放有效地址)。

其中, $Ad(IR)$ 表示取出 IR 中存放的指令字的地址字段

(4) 执行周期

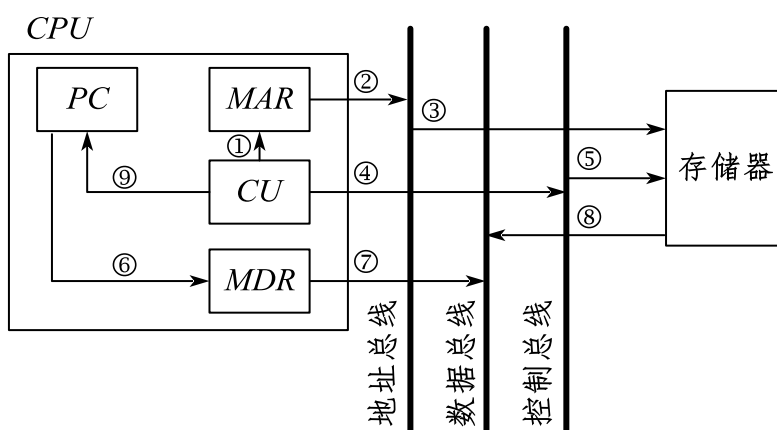
取操作数, 并根据 IR 中指令字的操作码通过 ALU 操作产生执行结果。

数据流: 不同指令执行周期操作不同, 无统一的数据流。

(5) 中周期

处理中断请求。

假设程序断点存入堆栈中, 并用 SP 指向栈顶地址, 且进栈操作作为先修改栈顶指针, 后存入数据, 且默认为向下生长型。



中断周期的数据流

数据流：

① CU 控制将 SP 减1, SP ① MAR ② 地址总线 ③ 主存

② CU 发出写命令 ④ 控制总线 ⑤ 主存。

③ PC ⑥ MDR ⑦ 数据总线 ⑧ 主存 (程序断点存入主存)。

④ CU (中断服务程序的入口地址) ⑨ PC 。

3) 指令执行方案

(1) 单指令周期: 对于所有指令都选用相同的执行时间来完成。

特点:

①指令之间串行执行，即下一条指令只能在前一条指令执行结束后才能启动。

②指令周期取决于执行时间最长的指令的执行时间，降低了整个系统的运行速度。

(2) 多指令周期：对不同类型的指令选用不同的执行步骤。

特点：

①指令之间串行操作。

②不同指令的周期数可不同，不同指令的时钟周期数也可不同。

(3) 流水线方案：指令之间并行执行，力争在每个时钟周期完成一条指令的执行过程。

题 1. CPU 从主存取出一条指令并执行该指令的时间叫做 ()

A. 机器周期 B. 指令周期 C. 时钟周期 D. 总线周期

答案：B

题 2. 下列说法中，正确的是 ()。

I. 指令字长等于机器字长的前提下，取指周期等于机器周期

II. 指令字长等于存储字长的前提下，取指周期等于机器周期

III. 指令字长和机器字长的长度没有任何关系

IV. 为了硬件设计方便，指令字长都和存储字长一样大

A. II、III B. II、III、IV C. I、III、IV D. I、IV

答案：A

解析：①机器字长是 CPU 一次运算的最多位数。

②指令字长取决于操作码位数和操作表长度，与机器字长无必然联系。

③指令字长一般取字节或存储字长的整数倍，不一定等于存储字长。

数据通路

1) 数据通路的功能

(1) 数据通路的定义

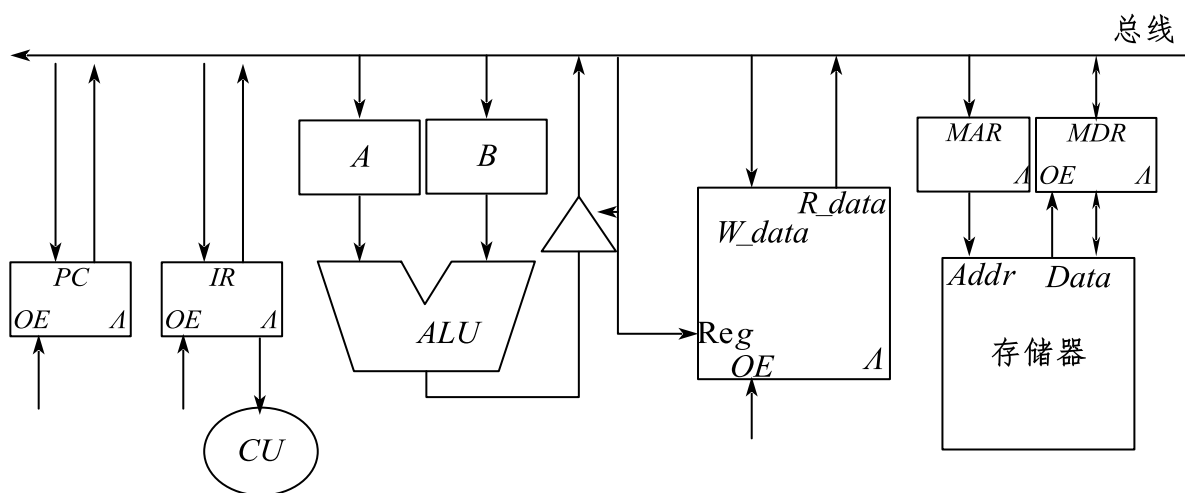
数据在功能部件之间传送的路径，包括数据通路上流经的部件和中断处理逻辑等。

(2) 数据通路的功能

实现 CPU 内部的运算器与寄存器及寄存器之间的数据交换。

2) 数据通路的基本结构

(1) 单总线结构：连接各部件的总线只有一条。

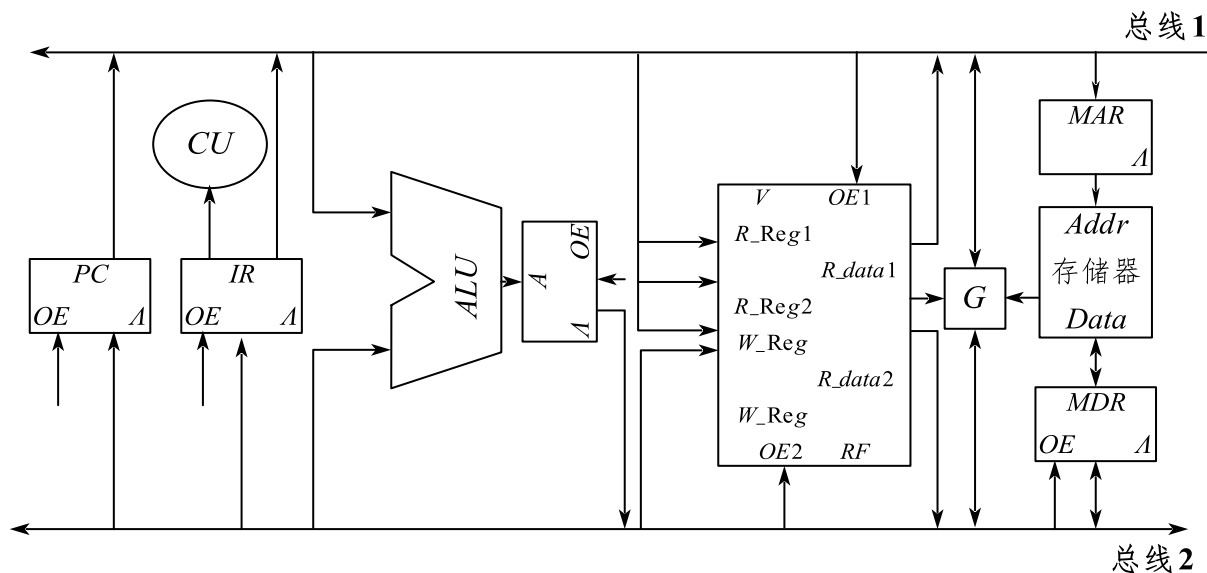


单总线CPU数据通路框图

特点：

- ①结构相对简单，但数据传输存在较多的冲突现象，性能较低。
- ②降低了操作间的并行性，导致指令周期变长。
- ③ALU的两个数据输入端各要设置一个暂存器（A，B）或在ALU的一个数据输入端和数据输出端设置暂存器。

（2）双总线结构：连接各个部件的总线有2条。

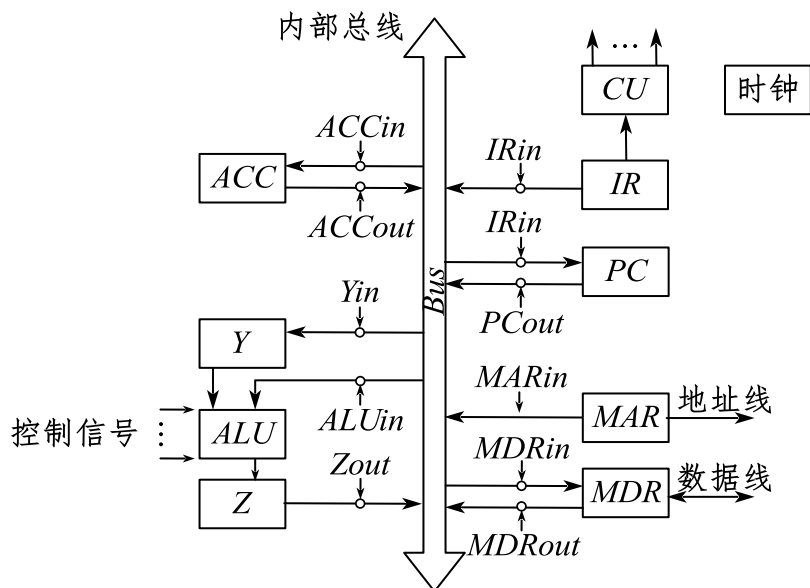


双总线CPU数据通路基本框架

特点：

- ①提高了操作间的并行性，缩短了指令周期，但增加了硬件复杂性和成本。
- ②ALU的两个操作数直接分别从两条总线上获得，因此两个输入端不需要暂存器，但输出端仍需要暂存器（ALU结果要输出到目的寄存器中）。

(3) 典型单总线结构的数据通路:



CPU 内部总线的数据通路和控制信号

①寄存器之间的数据传送

寄存器之间的数据传送可通过CPU内部总线完成。在上图中,某寄存器AX的输出和输入分别由AXout和AXin控制。这里以PC寄存器为例,把PC内容送至MAR,实现传送操作的流程及控制信号为:

$PC \rightarrow Bus$ $PCout$ 有效, PC 内容送总线

$Bus \rightarrow MAR$ $MARin$ 有效, 总线内容送 MAR

②主存与CPU之间的数据传送

主存与CPU之间的数据传送也要借助CPU内部总线完成。现以CPU从主存读取指令为例说明数据在数据通路中的传送过程。实现传送操作的流程及控制信号为:

$PC \rightarrow Bus \rightarrow MAR$ $PCout$ 和 $MARin$ 有效, 现行指令地址 $\rightarrow MAR$

$I \rightarrow R$ CU发读命令

$MEM(MAR) \rightarrow MDR$ $MDRin$ 有效

$MAR \rightarrow Bus \rightarrow IR$ $MDRout$ 和 $IRin$ 有效, 现行指令地址 $\rightarrow IR$

③执行算术或逻辑运算

执行算术或逻辑操作时,由于ALU本身是没有内部存储功能的组合电路,因此如要执行加法运算,相加的两个数必须在ALU的两个输入端同时有效。上图中的暂存器Y即用于该目的。先将一个操作数CPU内部总线送入暂存器Y保存,Y的内容在ALU的左输入端始终有效,

再将另一个操作数经总线直接送到 *ALU* 的右输入端。这样两个操作数都送入了 *ALU*，运算结果暂存在暂存器 *Z* 中。

$$AD(IR) \rightarrow Bus \rightarrow MAR \quad MDRout \text{ 和 } MARin \text{ 有效}$$

| | |
|-------------------|-----------|
| $I \rightarrow R$ | CU 发读命令 |
|-------------------|-----------|

$MEM \rightarrow \text{数据线} \rightarrow MDR$ 操作数从存储器 $\rightarrow \text{数据线} \rightarrow MDR$

$$MDR \rightarrow Bus \rightarrow Y \quad MDRout \text{ 和 } Yin \text{ 有效, 操作数 } \rightarrow Y$$

$(ACC)+(Y) \rightarrow Z$ $ACCout$ 和 $ALUin$ 有效, CU 向 ALU 发加命令, 结果 $\rightarrow Z$

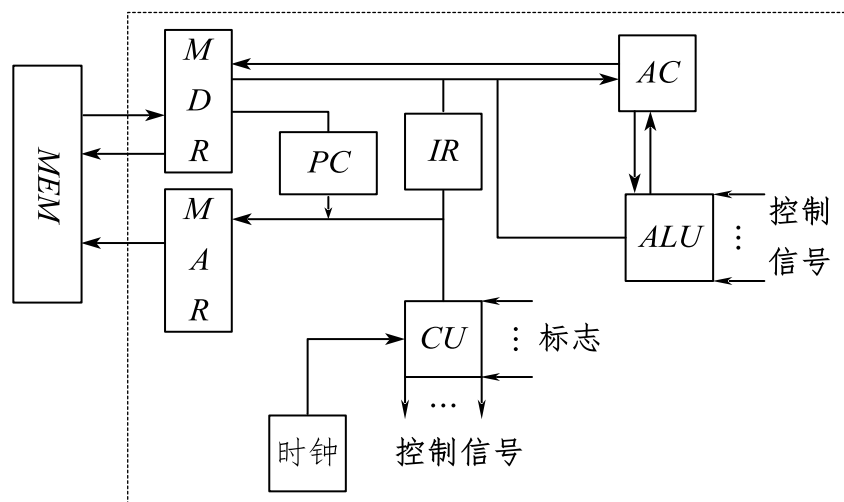
$Z \rightarrow ACC$ Z_{out} 和 ACC_{in} 有效, 结果 $\rightarrow ACC$

注:

① “in”表示该部件的允许输入信号;“out”表示该部件的允许输出信号。

②各部件用大写字母表示。

题 3. 某模型机的主机结构如下图所示，其中 MEM 为主存， AC 为累加器， CU 为控制单元， STA 指令的功能是将 AC 中的数取出送到主存，如果每个工作周期都是 3 个节拍，请分别写出取址和执指周期的微操作序列。



解析：取指： $T_0 \quad PC \rightarrow MAR, PC_{out}, MAR_{in}$

$$T_1 \quad M[MAR] \rightarrow MDR, MAR_{out}, MdRin$$
$$T, \quad MDR \rightarrow IR, MDR_{out}, IR_{in}$$

执行: $T_0 \quad Ad(IR) \rightarrow MAR, MARin$

$$T_1 \quad AC \rightarrow MDR, ACout, MdRin$$

3. 指令流水线

基本概念

1) 提高处理机并行性方法:

①时间上的并行技术，将一个任务分解为几个不同的子阶段，每个阶段在不同的功能部件上并行执行，以便在同一时刻能够同时执行多个任务，进而提升系统的性能，此方法称为流水线技术。

②空间上的并行性，在一个处理机内设置多个执行相同任务的功能部件，并让这些功能部件并行工作，这样的处理机称为超标量处理机。

2) 指令流水的定义

将一条指令的执行过程分为若干阶段，每个阶段都由相应的功能部件完成，视其为流水段，则指令的执行过程就构成了一条指令流水线。

例如：将一条指令的执行过程分为如下5个阶段：

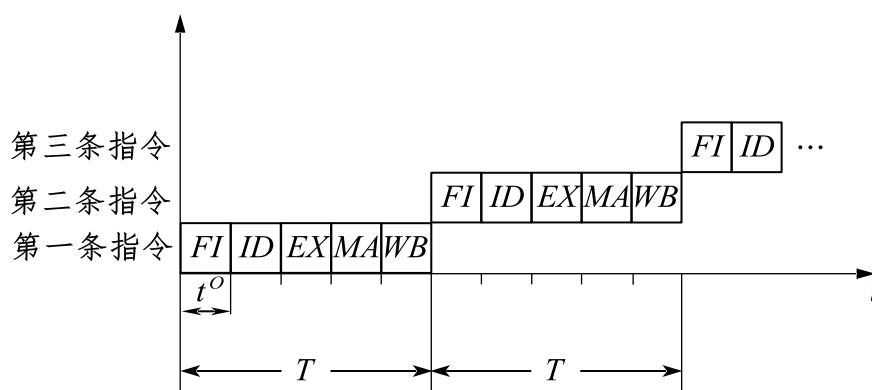
①取指（*IF*）：从指令存储器或*Cache*中取指令。

②译码/读寄存器（*ID*）：操作控制器对指令进行译码，同时从寄存器堆中取操作数。

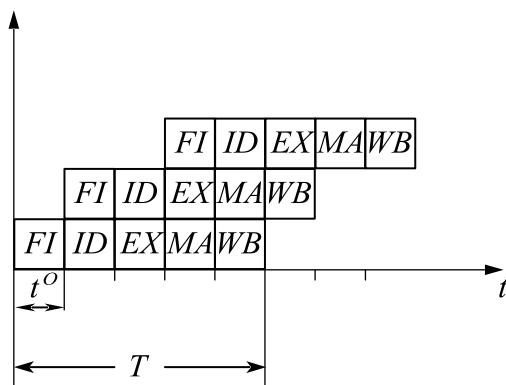
③执行/计算地址（*EX*）：执行运算操作或计算地址。

④访存（*MEM*）：对存储器进行读写操作。

⑤写回（*WB*）：将指令执行结果写回寄存器堆。



(a) 非流水线



(b) 流水线

指令执行过程

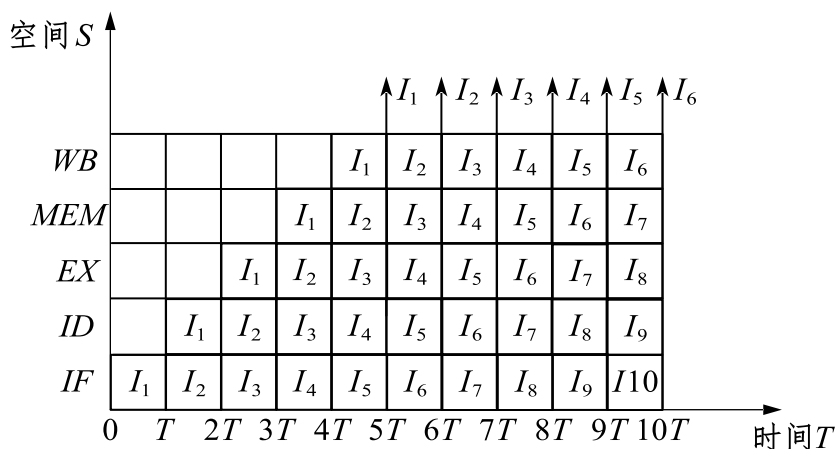
3) 指令流水线的设计原则

①指令流水段的个数以最复杂指令所用的功能段个数为准；

②流水段的长度以最复杂的操作所花的时间为准。

注：流水线方式并不能缩短单条指令的执行时间，但对整个程序来说，执行效率得到大幅提高。

4) 流水线的表示方法：时空图



一个5段指令流水线时空图

流水线的冒险与处理

流水线冒险：指在指令流水线中，遇到一些情况使得流水线无法正确执行后续指令而引起流水线阻塞或停顿的现象。

1) 结构冒险

由于多条指令在同一时刻争用同一资源而形成的冲突，也称为资源冲突，即由硬件资源竞争造成的冲突，有以下两种解决办法：

①前一指令访存时，使后一条相关指令（以及其后续指令）暂停一个时钟周期。

②单独设置数据存储器 and 指令存储器,使取数和取指令操作各自在不同的存储器中进行。事实上,现代计算机都引入了 *Cache* 机制,而 *L1 Cache* 通常采用数据 *Cache* 和指令 *Cache* 分离的方式,因而也就避免了资源冲突的发生。

2) 数据冒险

指令流水线相对于指令串行执行来说,可能会改变不同指令操作间的先后顺序,当一个操作必须等待另一个操作完成后才能进行时,将引发流水线停顿现象,称之为数据冒险。

数据冒险的类型:

①写后读 (*Read After Write, RAW*) 相关:表示当前指令将数据写入寄存器后,下一条指令才能从该寄存器读取数据。否则,先读后写,读到的就是错误(旧)数据。

②读后写 (*Write After Read, WAR*) 相关:表示当前指令读出数据后,下一条指令才能写该寄存器。否则,先写后读,读到的就是错误(新)数据。

③写后写 (*Write After Write, WAW*) 相关:表示当前指令写入寄存器后,下一条指令才能写该寄存器。否则,下一条指令在当前指令之前写,将使寄存器的值不是最新值。

解决的办法有以下几种:

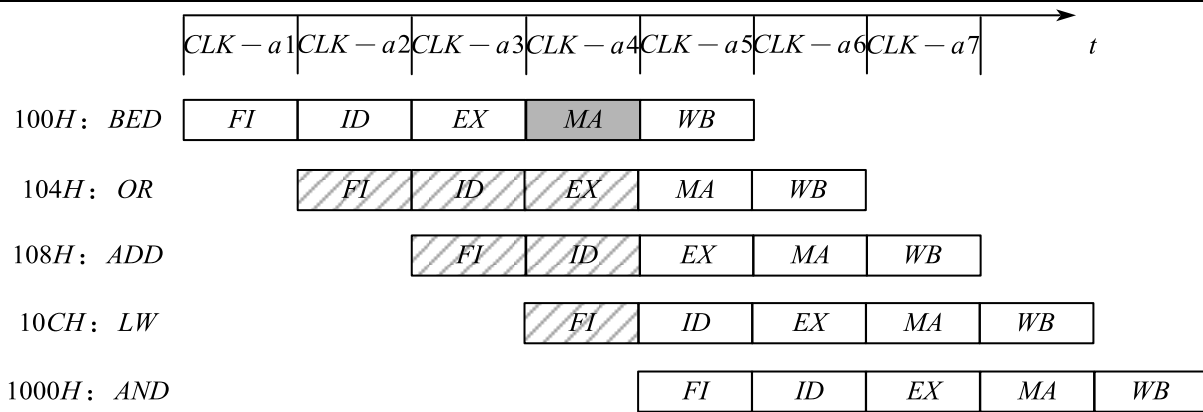
①把遇到数据相关的指令及其后续指令都暂停一至几个时钟周期,直到数据相关问题消失后再继续执行,可分为硬件阻塞 (*stall*) 和软件插入 “*NOP*” 指令两种方法。

②设置相关专用通路,即不等前一条指令把计算结果写回寄存器组,下一条指令也不再读寄存器组,而直接把前一条指令的 *ALU* 的计算结果作为自己的输入数据开始计算过程,使本来需要暂停的操作变得可以继续执行,这称为数据旁路技术。

③通过编译器对数据相关的指令编译优化的方法,调整指令顺序来解决数据相关。

(3) 控制冒险

在指令流水线中,条件转移指令做出决定之前,其后续指令已经出现在流水线中。这时,如果决定的结果与已经进入流水线的指令序列不同的话,在条件转移指令之后进入流水线的指令执行就是无效的。这种由于指令间执行顺序的约束关系引起的流水线冒险称为控制冒险。



分支指令引起的流水线冒险

解决方法:

①对转移指令进行分支预测, 尽早生成转移目标地址。分支预测分为简单(静态)预测和动态预测。静态预测总是预测条件不满足, 即继续执行分支指令的后续指令。动态预测根据程序执行的历史情况, 进行动态预测调整, 有较高的预测准确率。

②预取转移成功和不成功两个控制流方向上的目标指令。

③加快和提前形成条件码。

④提高转移方向的猜准率。

注意: Cache 缺失的处理过程也会引起流水线阻塞, 在不过多增加硬件成本的情况下, 如何尽可能地提高指令流水线的运行效率是选用指令流水线技术必须解决的关键问题。

3. 流水线的性能指标

1) 流水线的吞吐率 (TP): 指在单位时间内流水线所完成的任务数量, 或输出结果的量。

$$TP = \frac{n}{T_k} = \frac{n}{(k+n-1)\Delta t}$$

式中: n 为任务数;

T_k 是处理完 n 个任务用的总时间;

k 为流水段的段数;

Δt 为时钟周期。

当连续输入的任务数 $n \rightarrow \infty$ 时, 最大吞吐率 $TP_{\max} = \frac{1}{\Delta t}$

2) 流水线的加速比 (S): 完成同一批任务, 不使用流水线与使用流水线所用的时间比。

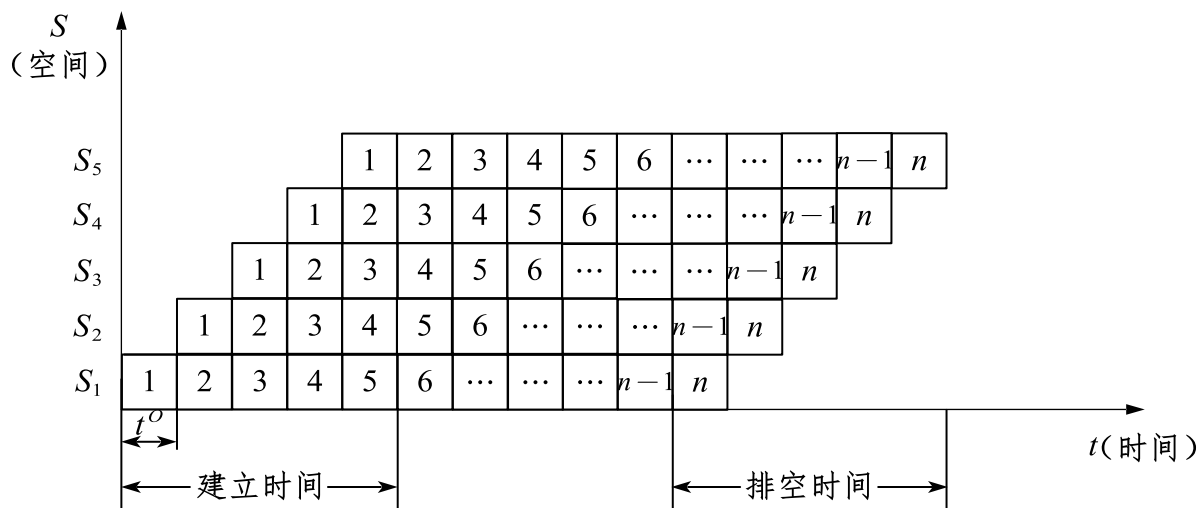
$$S = \frac{T_o}{T_k} = \frac{kn\Delta t}{(k+n-1)\Delta t} = \frac{kn}{(k+n-1)} = \frac{k}{1 + \frac{(k-1)}{n}}$$

式中： T_o ：不使用流水线的总时间；

T_k ：使用流水线的总时间。

连续输入的任务数 $n \rightarrow \infty$ 时，得最大加速比 $S_{\max} = k$ 。

3) 流水线的效率 (E)：指流水线的设备利用率，在时空图上定义为 n 个任务占用的时空与 k 个流水线总的时空区之比。



n 条指令在 5 段流水线上执行的时空图

$$E = \frac{k \cdot n \cdot \Delta t}{k(k+n-1)\Delta t} = \frac{n}{k+n-1} = \frac{1}{k} S = TP \cdot \Delta t$$

题 1. 设指令由取指、分析、执行 3 个子部件完成，并且每个子部件的时间均为 Δt ，若采用常规标量单流水线处理机（即处理机的度为 1），连续执行 12 条指令，共需（ ）。

- A. $12\Delta t$ B. $14\Delta t$ C. $16\Delta t$ D. $18\Delta t$

答案：B

解析： $[3 + (12 - 1)]\Delta t = 14\Delta t$

题 2. 在无转发机制的五段基本流水线（取指、译码/读寄存器、运算、访写回寄存器）中，下列指令序列存在数据冒险的指令对是（ ）。

I1: $addR1, R2, R3; (R2) + (R3) \rightarrow R1$

I2: $addR5, R2, R4; (R2) + (R4) \rightarrow R5$

I3: $addR4, R5, R3; (R5) + (R3) \rightarrow R4$

I4: $addR5, R2, R6; (R2) + (R6) \rightarrow R5$

- A. I1 和 I2 B. I2 和 I3 C. I2 和 I4 D. I3 和 I4

答案：B

解析：数据冒险即数据相关，指在一个程序中存在必须等前一条指令执行完才能执行后一条指令的情况，此时，这两条指令即为数据相关。当多条指令重叠处理时就会发生冲突。首先这两条指令发生写后读相关，且两条指令在流水线中的执行情况(发生数据冒险)如下表所示。

| 时钟 指令 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|----------|----|---------|---------|----|----|----|---|
| I2 | 取指 | 译码/读寄存器 | 运算 | 访存 | 写回 | | |
| I3 | | 取指 | 译码/读寄存器 | 运算 | 访存 | 写回 | |

指 令
I2 在

时钟5时将结果写入寄存器（R5），但指I3在时钟3时读寄存器（R5）。本来指令I2应先写入R5，指令I3后读R5，结果变成指令I3先读R5，指I2后写入R5，因而发生数据冲突。

课时七 练习题

1. CPU 从主存取出一条指令并执行该指令的时间叫做 ()
A. 机器周期 B. 指令周期 C. 时钟周期 D. 总线周期
2. 组成指令流水线的各功能段的执行时间最好 ()。
A. 与指令周期一致 B. 按指令功能不同分配不同的时间 C. 尽量不等 D. 尽量相等
3. 下列选项中, 不会引起指令流水阻塞的是 ()
A. 数据旁路 (转发) B. 数据相关 C. 条件转移 D. 资源冲突
4. 下列给出的指令系统特点中, 有利于实现指令流水线的是 ()
I. 指令格式规整且长度一致
II. 指令和数据按边界对齐存放
III. 只有 Load/Store 指令才能对操作数进行存储访问
A. 仅 I、II B. 仅 II、III C. 仅 I、III D. I、II、III
5. 在采用“取指、译码/取数、执行、访存、写回”5段流水线的处理器中, 执行如下指令序列, 其中 $s0$ 、 $s1$ 、 $s2$ 、 $s3$ 和 $t2$ 表示寄存器编号。

$I1: add\ s2, s1, s0 \quad //R[s2] \leftarrow R[s1] + R[s0]$

$I2: load\ s3, 0(t2) \quad //R[s3] \leftarrow M[R[t2] + 0]$

$I3: add\ s2, s2, s3 \quad //R[s2] \leftarrow R[s2] + R[s3]$

$I4: store\ s2, 0(t2) \quad //M[R[t2] + 0] \leftarrow R[s2]$

下列指令对中, 不存在数据冒险的是 ()

- A. $I1$ 和 $I3$ B. $I2$ 和 $I3$ C. $I2$ 和 $I4$ D. $I3$ 和 $I4$
6. 某计算机最复杂指令的执行需要完成5个子功能, 分别由功能部件 $A \sim E$ 实现, 各功能部件所需时间分别为 $80ps$ 、 $50ps$ 、 $50ps$ 、 $70ps$ 和 $50ps$, 采用流水线方式执行指令, 流水段寄存器延时为 $20ps$, 则 CPU 时钟周期至少为 ()。
A. $60ps$ B. $70ps$ C. $80ps$ D. $100ps$
7. 某 CPU 主频为 $1.03GHz$, 采用4级指令流水线, 每个流水段的执行需要1个时钟周期。假定 CPU 执行了100条指令, 在其执行过程中, 没有发生任何流水线阻塞, 此时流水线的吞吐率为 ()
A. 0.25×10^9 条指令/秒 B. 0.97×10^9 条指令/秒

C. 1.0×10^9 条指令/秒

D. 1.03×10^9 条指令/秒

8. 采用流水线技术的计算机在流水处理过程中，会出现哪三种相关冲突？简述各冲突的主要特征。

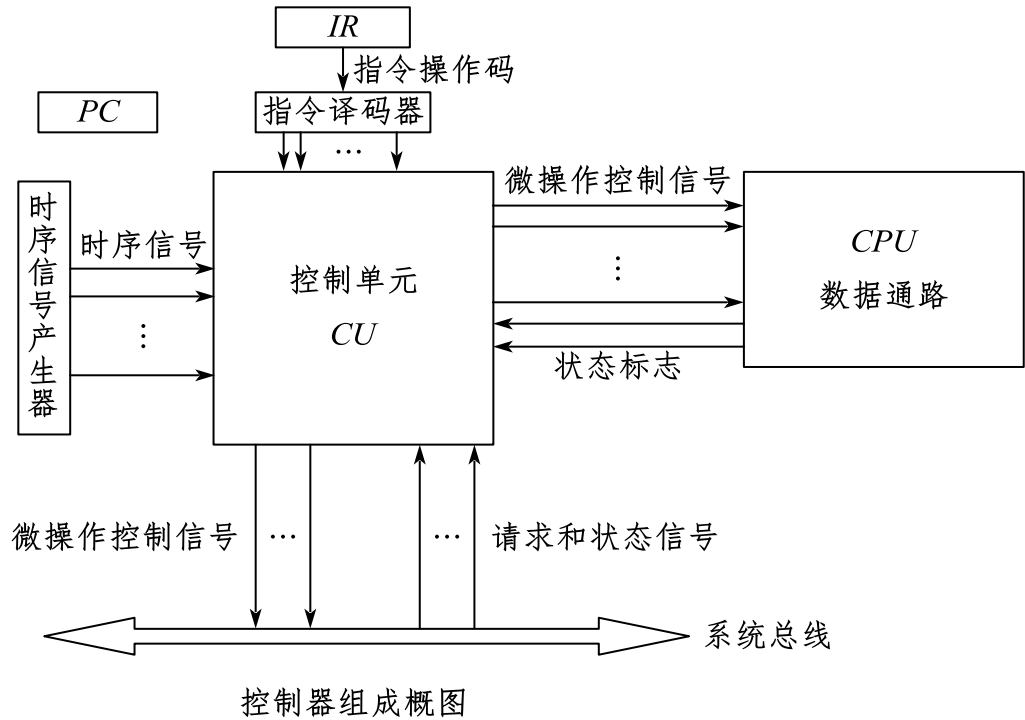
课时八 控制器

| 考点 | 重要程度 | 占分 | 题型 |
|----------|------|-----|-------|
| 1. 概括与设计 | ★★★★ | 4~8 | 选择、大题 |
| 2. 异常和中断 | ★★★★ | 4~8 | 选择、大题 |

1. 概括与设计

1) 结构和功能

(1) 控制器的结构：



组成：

- ①控制寄存器和译码器，包括PC、IR和指令译码器(ID)；
- ②时序信号产生器；
- ③控制单元(CU)，是控制器的核心，发出整机运行所需的全部控制信号。

(2) 控制器的功能

- ①从主存中取出一条指令，并指出下一条指令在主存中的位置；
- ②对指令进行译码或测试，产生相应的操作控制信号，以便启动规定的动作；
- ③指挥并控制CPU、主存、输入和输出设备之间的数据流动方向。

控制单元的设计主要有两种方法(它们的区别在于如何产生微操作)：硬布线(组合逻辑)设计方法和微程序(存储逻辑)设计方法。

2) 控制方式

(1) 同步控制方式

指系统有一个统一的时钟，所有的控制信号均来自统一的时钟信号。通常以最长的微操作序列和最繁琐的微操作作为标准，采取完全统一的，具有相同时间间隔和相同数目的节拍作为机器周期来运行不同的指令。

优点：控制电路简单

缺点：运行速度慢

(2) 异步控制方式

异步控制方式不存在基准时标信号，各部件按自身固有的速度工作，通过应答方式进行联络。

异步控制方式的优点是运行速度快，缺点是控制电路比较复杂。

(3) 联合控制方式

联合控制方式是介于同步、异步之间的一种折中。这种方式对各种不同的指令的微操作实行大部分采用同步控制、小部分采用异步控制的办法。

3) 微程序控制器

(1) 基本思想

将每条机器指令编写成一个微程序，每个微程序包含若干个微指令，每条微指令对应一个或几个微操作命令。将微程序存入控制存储器中，然后用寻址一般指令的办法寻址每个微程序中的微指令。

(2) 相关概念

①微操作：一条机器指令可以分解成一系列微操作，其是计算机中最基本的不可再分解的操作。

②微命令：在微程序控制的计算机中，将控制部件向执行部件发出的各种控制命令称为微命令，其是构成控制序列的最小单位。

微命令分为相容性微命令和互斥性微命令：

- { 相容性微命令：可以同时产生，共同完成某一些微操作的微命令；
- { 互斥性微命令：不允许同时出现的微命令。

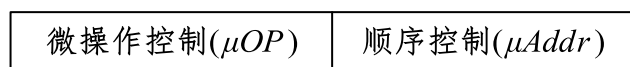
注：a. 微命令和微操作是一一对应的，微命令是微操作的控制信号，微操作是微命令的执行过程。

b. 微命令的相容性和互斥性是相对的，一个微命令可以与一些微命令相容，和另一些微命令互。

③微指令：若干微命令的集合，存放微指令的控制存储器的单元地址称为微地址。

一条微指令通常包含两大部分：

- a. 操作控制字段(微操作字段)：用于产生某一步操作所需的各种操作控制信号。
- b. 顺序控制字段(微地址码字段)：用于控制产生下一条要执行的微指令地址。



微指令基本格式

④微周期：指从控制存储器中读取一条微指令并执行相应微操作所需的时间。

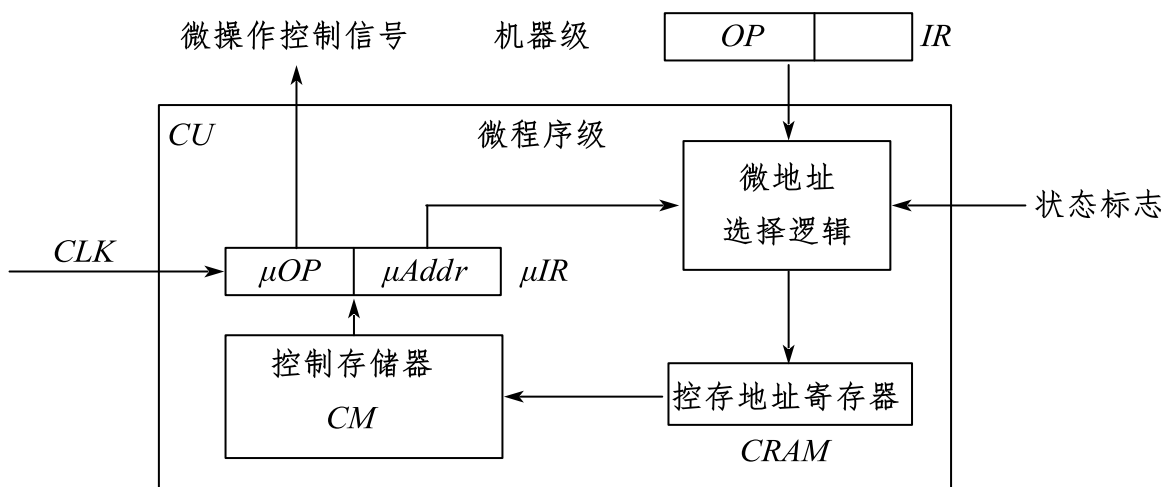
⑤控制存储器 (CM)：用于存放微程序，其在CPU内部，用ROM实现。

⑥程序和微程序：程序是指令的有序集合，用于完成特定的功能；微程序是微指令的有序集合，一条指令的功能由一段微程序来完成。

注：通常，一条机器指令对应一个微程序。由于任何一条机器指令的取指令操作都是相同的，因此可将取指令操作的微命令统一编成一个微程序，这个微程序只负责将指令从主存单元中取出并送至指令寄存器。

(3) 微程序控制器的组成和工作过程

①微程序控制器的基本组成



微程序控制单元组成框图

a. 控制存储器 (CM)：其是微程序控制器的核心部件，用于存放各指令对应的微程序，由ROM构成。

b. 微指令寄存器：用于存放从CM取出的微指令，其位数等于微指令字长。

c. 微地址寄存器：接收微地址形成部件送来的微地址，为在CM中读取微指令作准备。

d. 微地址形成部件：用于产生初始微地址和后继微地址，以保证微指令的连续执行。

注：a. 地址寄存器(MAR)。用于存放主存的读/写地址。

- b. 微地址寄存器(*CMAR*)。用于存放控制存储器的读/写微指令的地址。
- c. 指令寄存器(*IR*)。用于存放从主存中读出的指令。
- d. 微指令寄存器(*CMDR* 或 μIR)。用于存放从控制存储器中读出的微指令。

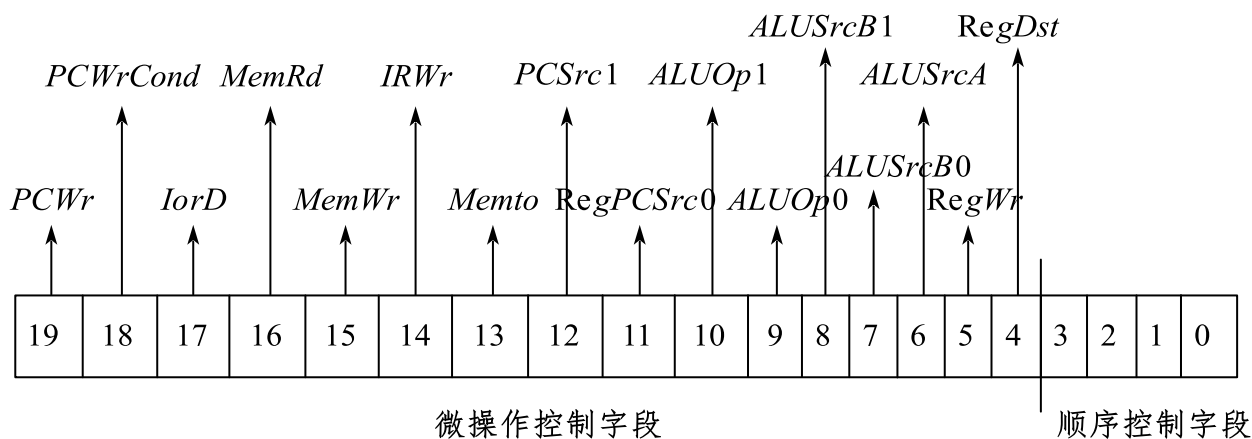
②微程序控制器的工作过程

- a. 执行取微指令公共操作。具体的执行是：在机器开始运行时，自动将取指微程序的入口地址送入*CMAR*，并从*CM*中读出相应的微指令送入*CMDR*。取指微程序的入口地址一般为*CM*的0号单元，当取指微程序执行完后，从主存中取出的机器指令就已存入指令寄存器中。
- b. 由机器指令的操作码字段通过微地址形成部件产生该机器指令所对应的微程序的入口地址，并送入*CMAR*。
- c. 从*CM*中逐条取出对应的微指令并执行。
- d. 执行完对应于一条机器指令的一个微程序后，又回到取指微程序的入口地址，继续第a步，以完成取下一条机器指令的公共操作。

(4) 微指令的格式

①水平型微指令

定义：水平型微指令可以同时给出多个并行操作的命令，每个微命令需要由微指令操作控制字段中的一位来指出，而且微指令中还设置后继地址字段来进行微程序的顺序控制。



②垂直型微指令：其格式类似于普通的机器指令格式，只能同时给出1~2个微指令，不支持微操作的并行性。微操作控制字段采用微命令编码法，由编码规定微指令的功能。

| 操作编码 | 地址码 | | 其他 | 微指令类型及功能 |
|------|--------|--------|-------|-----------------------------------|
| | 3~7 | 8~12 | 13~15 | |
| 000 | 源寄存器 | 目的寄存器 | 其他控制 | 传送型微指令 |
| 001 | ALU左输入 | ALU右输入 | ALU | 运算控制型指令 按ALU字段所规定的功能执行，其结果送暂存器 |
| 010 | 寄存器 | 移位次数 | 移位方式 | 移位控制型微指令 按移位方式对寄存器中的数据移位 |

③微指令格式的比较

- 水平型微指令比垂直型微指令并行操作能力强，效率高、灵活性强。
- 水平型微指令执行一条机器指令所需的微指令数目少，因此速度比垂直型微指令的速度快。
- 垂直型微指令字长较短，但用以解释机器指令的微程序较长；相反，水平型微指令的字长较长，但用以解释机器指令的微程序较短。
- 垂直型微指令与机器指令相似，所以垂直型微程序设计较容易；水平型微指令与机器指令差别较大，水平型微程序设计较困难。

通过比较可知，水平型微指令格式更为实用。

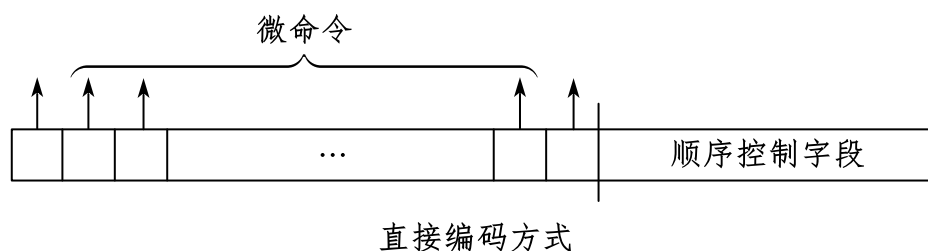
(5) 微指令格式设计

以水平型微指令格式为例，说明微命令编码技术和微地址形成方法：

①微命令的编码方式：用来解决微指令操作控制字段的格式安排

a. 直接编码

主要思想：用微操作控制字段的每一位分别表示一个微命令。该位为“0”表示对应的微命令无效，该位为“1”表示对应的微命令有效。

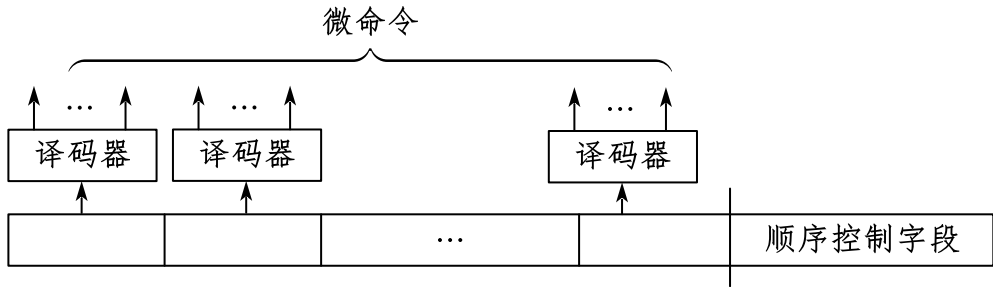


特点：微指令结构简单直观，微指令执行速度快，但微指令字长较长，适用于结构比较

简单的机器。

b. 字段直接编码

基本思想：利用微命令之间的相容和互斥关系，将微操作控制字段分为若干小字段，把一组互斥微命令组织在一起，用一个小字段编码表示；将相容的微命令安排在不同字段内。



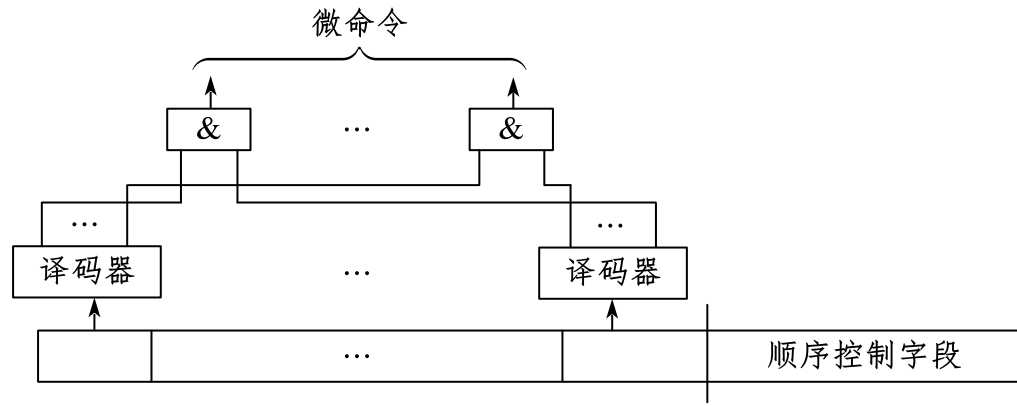
字段直接编码方式

特点：可以有效地压缩微指令字长，但译码过程会影响微指令的执行速度。

注：为每个字段编码时，应考虑无操作的情况(通常把全“0”编码作为“无操作”)

c. 字段间接编码

基本思想：在字段直接编码的基础上，规定一个字段的某些微命令，要兼由另一个字段中的某些微命令来解释，则称为字段间接编码方式。

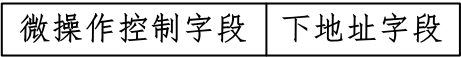


字段间接编码方式

特点：可以进一步压缩微指令的长度，但由于字段间的关联可能减弱微指令的并行控制能力，且译码级数增加使微指令执行更慢(通常作为直接编码法的辅助手段)。

②微地址的形成方法：在微程序执行的不同阶段获得下一条微指令地址(后继微地址)的方法不同：

a. 直接表示方法(断定方式)：由每条微指令的顺序控制字段直接给出后继微地址。



b. 根据机器指令的操作码形式：将机器指令取至指定寄存器后，微指令的地址由操作

码经微地址形成部件形成。

- c. 增量计数器法：即： $(CMAR) + 1 \rightarrow CMAR$ ，适用于后继微指令的地址连续的情况。
- d. 根据各种标志决定微指令分支转移的地址。
- e. 通过测试网络形式。
- f. 由硬件直接产生微程序入口地址(如取指周期微程序入口地址)。

(6) 微程序控制单元的设计步骤

①写出对应机器指令的微操作命令及节拍安排。

例如：取指阶段的微操作命令及节拍：

$T_0: PC \rightarrow MAR, 1 \rightarrow R$

$T_1: Ad(CMDR) \rightarrow CMAR$

$T_2: M(MAR) \rightarrow MDR, (PC) + 1 \rightarrow PC$

$T_3: Ad(CMDR) \rightarrow CMAR$

$T_4: MDR \rightarrow IR$

$T_5: OP(IR) \rightarrow \text{微地址形成部件} \rightarrow CMAR$

②确定微指令格式：包括其编码方式，后继微指令地址的形成方式、微指令字长等。

注：

- a. 微操作个数决定采用何种编码方式，以确定微指令的操作控制字段的位数。
- b. 由微指令数确定微指令的顺序控制字段的位数。

③编写微指令码点：根据操作控制字段每位代表的微操作命令，编写每条微指令的码点。

(7) 硬布线与微程序控制器的对比

| 类别 对比项目 | 微程序控制器 | 硬布线控制器 |
|------------|-----------------------------------|-----------------------------------|
| 工作原理 | 微操作控制信号以微程序的形式存放在控制存储器中，执行指令时读出即可 | 微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生 |
| 执行速度 | 慢 | 快 |
| 规整性 | 较规整 | 烦琐、不规整 |
| 应用场合 | CISC CPU | RISC CPU |
| 易扩充性 | 易扩充修改 | 困难 |

题 1. 微程序控制器中的控制存储器用来存放 ()。

- A. 机器指令和数据 B. 微程序和数据
C. 机器指令和微程序 D. 微程序

答案: D

解析: 控制存储器中只有微程序。

题 2. 水平型微指令的特点是 ()。

- A. 一次可以完成多个操作 B. 微指令的操作控制字段不进行编码
C. 微指令的格式简短 D. 微指令的格式较长

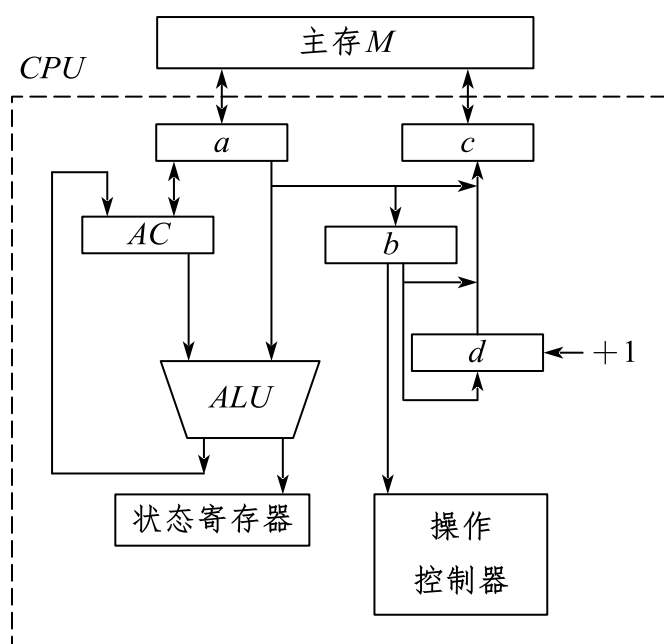
答案: A

题 3. CPU 结构如图所示, 其中有一个累加寄存器 AC, 一个状态寄存器, 各部分之间的连线表示数据通路, 箭头表示信息传送方向:

(1) 写出图中 a、b、c、d 四个寄存器的名称;

(2) 写出指令从主存取到 CPU 内的数据通路;

(3) 写出数据在主存和运算器间读出和写入的数据通路。



解析: (1) MDR、IR、MAR、PC

(2) $(PC) \rightarrow MAR$

$M(MAR) \rightarrow MDR$

$(MDR) \rightarrow IR$

(3) 读出 $Ad(IR) \rightarrow MAR$

写入 $Ad(IR) \rightarrow MAR$

$M(MAR) \rightarrow MDR$ $(AC) \rightarrow MDR$ $(MDR) \rightarrow (AC)$ $(MDR) \rightarrow M(MAR)$

2. 异常和中断

1) 基本概念

(1) 异常：*CPU* 内部产生的意外事件，也叫内中断(与正在执行的指令相关)。

(2) 中断：*CPU* 外部的设备向 *CPU* 发出的中断请求，通常用于信息的输入与输出，也称外中断(与正在执行的指令无关)。

2) 分类

(1) 异常的分类

①故障：指在引起故障的指令启动后，执行结束前被检测的异常事件。

如：取数据时的“缺段”、“缺页”；执行整数除法指令时，发现“除数为0”。

②自陷：即自己设计的陷阱，是被预先安排的一种“异常”事件，如：调试程序中的“断点设置”。

注：故障异常和自陷异常属于程序性异常(软件中断)。

③终止：指在执行指令的过程中发生了使计算机无法继续执行的硬件故障(随机发生的)。

如：控制器出错

注：终止异常和外中断属于硬件中断。

(2) 中断的分类

①可屏蔽中断：*CPU* 可通过中断屏蔽字来屏蔽某些中断，此类中断一般并非紧急中断。

②不可屏蔽中断：指非常紧急的硬件故障，如电源掉电等。

(3) 中断与异常的不同：

①“缺页”或“溢出”等异常事件是由特定指令在执行过程中产生的，而中断不和任何指令相关联，也不阻止任何指令的完成。

②异常的检测由 *CPU* 自身完成，不必通过外部的某个信号通知 *CPU*。对于中断，*CPU* 必须通过中断请求线获取中断源的信息，才能知道哪个设备发生了何种中断。

3) 响应过程

CPU 对异常和中断响应过程分为：关中断、保存断点和程序状态、识别异常和中断并转到相应的处理程序。

(1) 关中断

在保存断点和程序状态期间，不能被新的中断打断，因此要禁止响应新的中断，即关中断。

(2) 保存断点和程序状态

将程序的断点(返回地址)和被中断时的程序状态字寄存器 *PSWR* 的内容保存在栈中，以便返回被中断的程序继续执行。

(3) 识别异常和中断并转到相应的处理程序

异常和中断的识别有软件识别和硬件识别两种：

①软件识别方式是指 *CPU* 设置一个异常状态寄存器，用于记录异常原因。操作系统使用一个统一的异常或中断查询程序，按优先级顺序查询异常状态寄存器，以检测异常和中断类型，先查询到的先被处理，然后转到内核中相应的处理程序。

②硬件识别方式又称向量中断，异常或中断处理程序的首地址称为中断向量，所有中断向量都存放在中断向量表中。每个异常或中断都被指定一个中断类型号。在中断向量表中，类型号和中断向量一一对应，因而可以根据类型号快速找到对应的处理程序。

题 1. 以下关于异常和中断响应的叙述中，错误的是 ()。

- A. 异常事件检测由 *CPU* 在执行每一条指令的过程中进行
- B. 中断请求检测由 *CPU* 在每条指令执行结束，取下条指令之前进行
- C. *CPU* 检测到异常事件后所做的处理和检测到中断请求后所做的处理完全相同
- D. *CPU* 在中断响应时会关中断、保存断点和程序状态并转到相应的中断服务程序

答案：C

解析：*CPU* 检测到异常事件后所做的处理和检测到中断请求后所做的处理基本是相同的，但有些地方可能不同。例如，对于故障类异常，因为其断点为发生故障是的指令地址，所以要重新计算 *PC* 值，而中断的断点为下条指令地址(即 *PC* 值)，因此无须重新计算 *PC* 值。

题 2. 下列关于“自陷”(Trap，也称陷阱)的叙述中，错误的是 ()。

- A. 自陷是通过陷阱指令预先设定的一类外部中断事件
- B. 自陷可用于实现程序调试时的断点设置和单步跟踪
- C. 自陷发生后 *CPU* 将转去执行操作系统内核相应程序
- D. 自陷处理完成后返回到陷阱指令的下一条指令执行

答案: *A*

解析: 自陷是一种内部异常, *A* 错误。在 *x86* 中, 用于程序调试的“断点设置”功能是通过自陷机制实现的, *B* 正确。执行到自陷指令时, 无条件或有条件地自动调出操作系统内核程序进行执行, *C* 正确。*CPU* 执行陷阱指令后, 会自动地根据不同陷阱类型进行相应的处理, 然后返回到陷阱指令的下一条指令执行, *D* 正确。

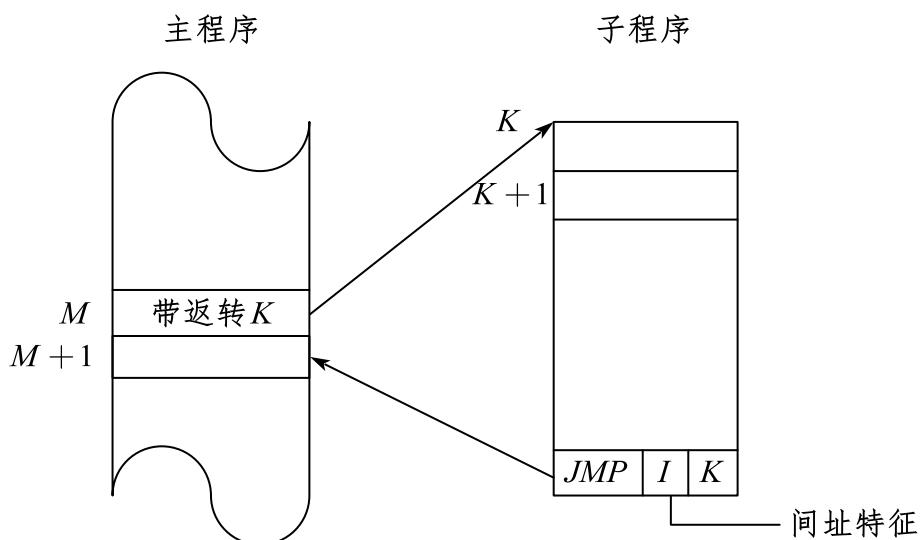
课时八 练习题

1. 微程序控制器中，机器指令与微指令的关系是()。
 - A. 每条机器指令由一条微指令来执行
 - B. 每条机器指令出一段微指令编成的微程序来解释执行
 - C. 一段机器指令组成的程序可出一条微指令来执行
 - D. 一条微指令由若干条机器指令组成
2. 为确定下一条微指令的地址，通常采用断定方式，其基本思想是()。
 - A. 用程序计数器 PC 来产生后继微指令地址
 - B. 用微程序计数器 μPC 来产生后继微指令地址
 - C. 通过微指令顺序控制字段由设计者指定或由设计者指定的判别字段控制产生后继微指令地址
3. 中断向量地址是()。
 - A. 子程序入口地址
 - B. 中断服务子程序入口地址
 - C. 中断服务子程序入口地址的地址
 - D. 中断返回地址
4. 相对于微程序控制器，硬布线控制器的特点是()。
 - A. 指令执行速度慢，指令功能的修改和扩展容易
 - B. 指令执行速度慢，指令功能的修改和扩展难
 - C. 指令执行速度快，指令功能的修改和扩展容易
 - D. 指令执行速度快，指令功能的修改和扩展难
5. 某计算机的控制器采用微程序控制方式，微指令中的操作控制字段采用字段直接编码法，共有33个微命令，构成5个互斥类，分别包含7、3、12、5和6个微命令，则操作控制字段至少有()。
 - A. 5位
 - B. 6位
 - C. 15位
 - D. 33位
6. 下面有关微指令、机器指令和微程序、程序的说法中，正确的是()。
 - A. 每一条机器指令由一段微程序解释、执行
 - B. 程序是指令的有序集合，而机器指令是微程序的有序集合
 - C. 微指令是一段机器指令的有序集合
 - D. 微程序存放在主存中

7. 某计算机采用微程序控制器，共有32条指令，公共的取指令微程序包含2条微指令，各指令对应的微程序平均由4条微指令组成，采用断定法(下地址字段法)确定下条微指令地址，则微指令中下地址字段的位数至少是()。

- A. 5 B. 6 C. 8 D. 9

8. 已知带返转指令的含义如下图所示，写出机器在完成带返转指令时，取指阶段和执行阶段所需的全部微操作命令及节拍安排。如果采用微程序控制，需增加哪些微操作命令？



课时九 总线

| 考点 | 重要程度 | 占分 | 题型 |
|------------|------|-------|-------|
| 1. 总线概括 | ★★ | 2 ~ 4 | 选择、大题 |
| 2. 总线仲裁机制 | ★★★★ | 4 ~ 8 | 选择、大题 |
| 3. 总线事务和定时 | ★★★★ | 4 ~ 8 | 选择、大题 |

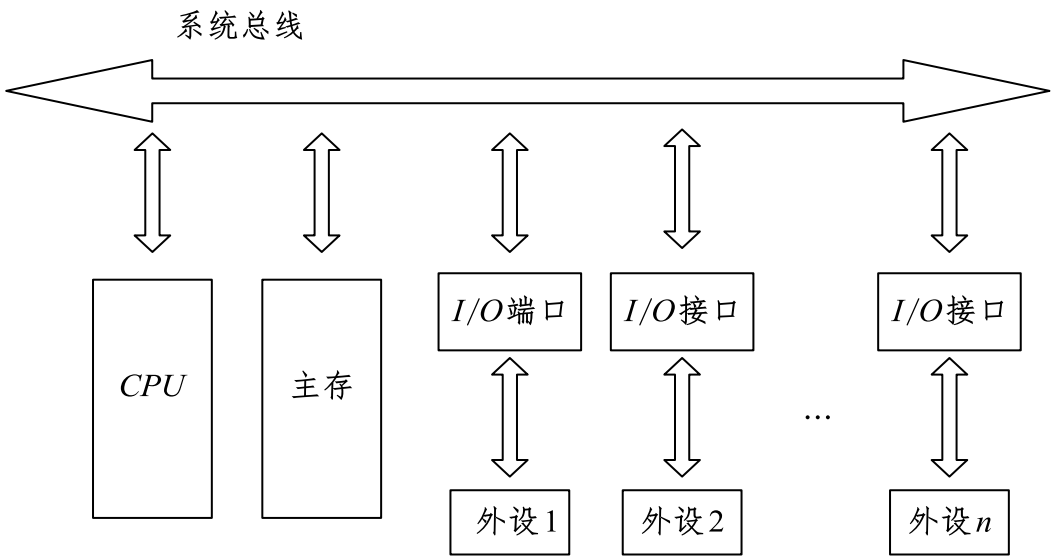
1. 总线概括

1) 基本概念

(1) 定义：总线是组能为多个部件分时共享的公共信息传送线路，即总线的基本组成，是一组导线加上接收和发送控制逻辑。

分时：指同一时刻只允许有一个部件向总线发送信息。

共享：各个部件可以通过总线共享信息。



总线结构的计算机组成框图

(2) 总线设备：总线上所连接的设备

主设备：指的是获得总线控制权的设备，CPU 通常为主模块，有些 I/O 接口既可作为主模块，也可作为从模块。

从设备：指的是被主设备访问的设备，它只能相应从主设备发来的各种总线命令，主存只能作为从模块。

2) 总线的分类（系统总线）

(1) 数据总线：用来在各功能部件之间传输数据信息（数据和机器指令）。

特点：①数据总线为双向性传输总线；

②数据总线的位数称为总线宽度，即总线宽度表示系统总线能同时传送的数据位数，与机器字长，存储字长有关。

(2) 地址总线：用来指出数据总线上的源数据或目的数据在主存中的地址或者 I/O 设备的地址。

特点：①地址来源于总线传输操作的发起者，即地址传输的方向为发起者到响应者，因此可以认为地址总线为单向总线。

②地址总线的宽度决定了系统总线的寻址范围。例如，若地址为 32 位，则按字节寻址的主存储器可达到的最大容量为 $2^{32} B = 4GB$ 。

(3) 控制总线：用来传送各种控制命令信号以及各部件之间协同操作所需要的时序信号和操作状态信号。

特点：①对于每根信号线来说，它是单向的；对于控制总线来说，它是双向总线。

②控制总线给出的信号类型决定了系统总线所支持的控制方式及通信方式。

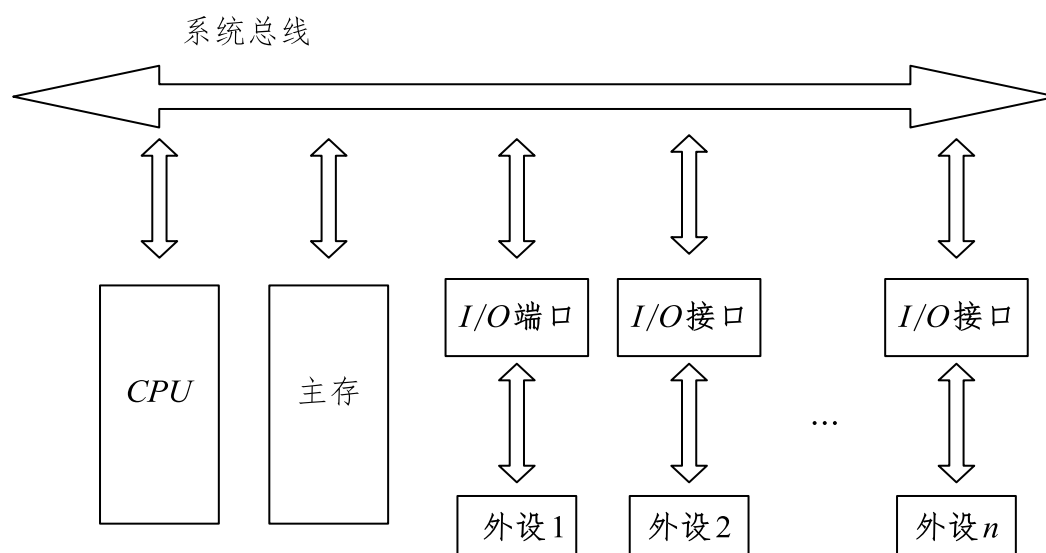
题 1. 系统总线中地址总线的作用是 ()。

- A. 用于选择存储单元
- B. 用于选择进行信息传输的设备
- C. 用于指定主存单元和 I/O 设备接口电路的地址
- D. 用于传送主存物理地址和逻辑地址

答案：C

3) 系统总线的结构

(1) 单总线结构：将 CPU 、主存、 I/O 设备都挂在一组总线上。



总线结构的计算机组成框图

注：单总线并不是指只有一根信号线，系统总线按传送信息的不同可细分为地址总线、数据总线和控制总线。

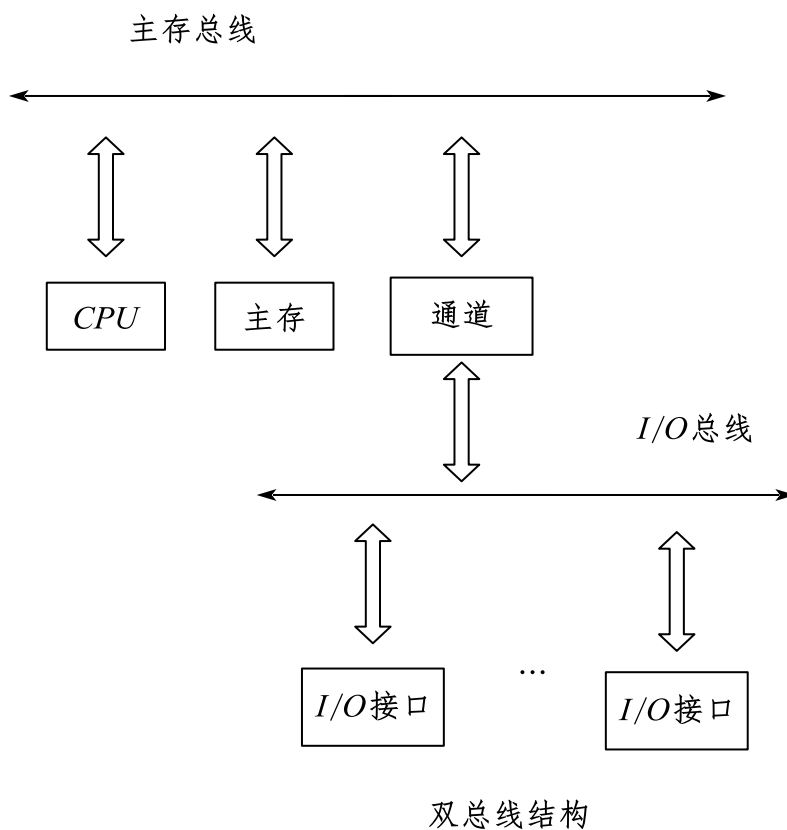
优点：结构简单，成本低，易于接入新的设备。

缺点：带宽低、负载重，多个部件只能争用唯一的总线，且不支持并发传送操作。

(2) 双总线结构

主存总线：用于在CPU、主存和通道之间传送数据。

I/O总线：用于在多个外部设备与通道之间传送数据。



优点：将低速 *I/O* 设备从单总线上分离出来，实现了存储器总线和 *I/O* 总线分离。

缺点：需要增加通道及硬件设备。

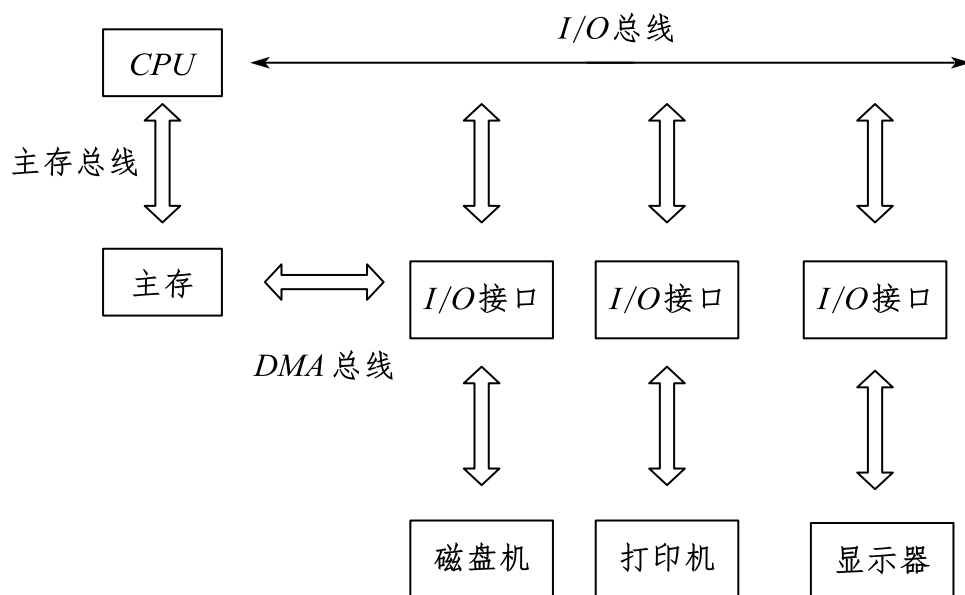
（3）三总线结构

在计算机系统各部件之间采用三条各自独立的总线（主存总线、*I/O* 总线、直接内存访问 *DMA* 总线）。

①主存总线：用于在 *CPU* 和主存之间传送地址、数据和控制信息。

② *I/O* 总线：用于 *CPU* 和各类外设之间通信。

③ *DMA* 总线：用于在内存和高速外设之间直接传送数据。



三总线结构

优点：提高了 I/O 设备的性能，使其更快地响应命令，提高系统吞吐量。

缺点：系统工作效率较低。

题 2. 所谓三总线结构的计算机是（ ）。

- A. 地址线、数据线和控制线三组传输线
- B. I/O 总线、主存总线和总线三组传输线
- C. I/O 总线、主存总线和系统总线三组传输线
- D. 以上都不对

答案：B

4) 性能指标

(1) 总线传输周期

指一次总线操作所需的时间，包括申请阶段、寻址阶段、传输阶段和结束阶段。总线传输周期通常由若干总线时钟周期构成。

(2) 总线时钟周期

即机器的时钟周期。计算机中有一个统一的时钟，以控制整个计算机的各个部件，总线也要受此时钟的控制。

(3) 总线工作频率

总线上各种操作的频率，为总线周期的倒数。实际上指 1 秒内传送几次数据，若总线周期

$= N$ 个时钟周期，则总线的工作频率 = 时钟频率 / N 。

(4) 总线时钟频率

即机器的时钟频率，它为时钟周期的倒数。

(5) 总线宽度

又称总线位宽，它是总线上同时能够传输的数据位数，通常指数据总线的根数，如 32 根称为 32 位总线。

(6) 总线带宽

可来理解为总线的最大数据传输率，即单位时间内总线上最多可传输数据的位数，通常用每秒传送信息的字节数来衡量，单位可用字节/秒(B/s)表示。总线带宽 = 总线工作频率 × (总线宽度/8)。

注：总线带宽和总线宽度应加以区别。

题 3. 某总线在一个总线周期并行传送 64 位数据，假设一个总线周期为一个总线时钟周期，总线时钟周期频率为 66MHz。问：

(1) 总线宽度是多少？

(2) 分析哪些因素影响宽度？

解析：(1) $66b \cdot 66MHz = 528MB/s$ ；

(2) 总线宽度、传输距离、总线发送和接收电路工作频率的限制以及数据传输形式。

2. 总线仲裁机制

1) 基本概念

当有多个主模块同时发出总线请求时，由总线仲裁逻辑决定优先响应哪一个主模块的请求，并将总线控制权移交给这个部件。

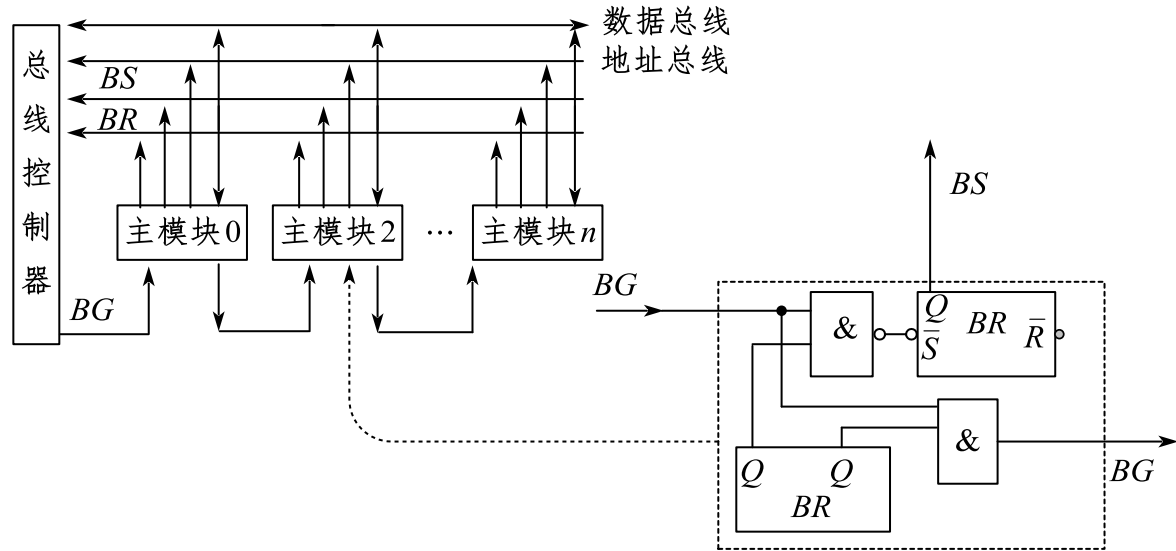
总线仲裁方式有集中式和分布式。

集中式：仲裁逻辑集中一起（例如，由 CPU 中的控制器承担），分为链式查询、计数器定时查询和独立请求三种方式。

分布式：仲裁逻辑分布在总线连接的各个部位中。

2) 链式查询方式

过程：通过一条判优链路对所有主模块逐个串行查询（通常，离总线控制器物理连接最近的模块优先级最高）。



链式查询框图

BR (Bus Request) 是总线请求信号。所有主模块都通过此线向总线控制器发请求信号，该信号有效时表示系统中至少有一个主模块请求使用总线。

BG (Bus Grant) 是总线允许信号。总线控制器通过此线向请求总线的主模块发出允许使用总线信号，该信号有效时表示总线控制器已经响应了总线请求。

BS (Bus Busy) 是总线忙信号。所有主模块都通过此线向总线控制器发送“忙”信号，该信号有效时表示总线正在被占用。

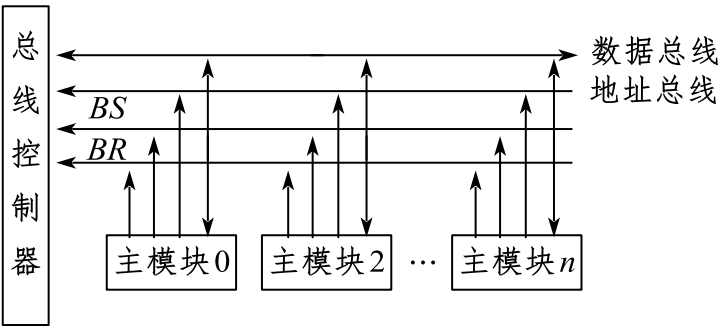
优点：控制信号线数量少，结构简单，易于扩充。

缺点：查询速度慢，对传播过程中电路故障敏感；各模块优先级固定不变，使用灵活性差。

3) 计数器定时查询

(1) 过程

在总线控制器中设置一个查询计数器。开始查询时，启动计数器计数。每计数一次，就将计数值作为模块地址发往各个主模块。每个申请总线的主模块对地址进行识别，地址相符的模块就获得了总线的控制权，并且通过设置总线忙信号 BS 有效使查询计数器停止计数。



计数器定时查询框图

(2) 主模块的优先级取决于计数器的工作方式

①若每次查询时计数器都是从0开始计数，则0号（模块地址）主模块的优先级最高，其他主模块优先级按照模块地址依次降低。

②若每次查询时计时器都是从上一次查询的计数终止点开始计数，则终止点对应的模块优先级最高。通常这种方式下计数器循环计数，这样主模块的优先级也就循环递减。

③若计数初值由程序设定，则各主模块的优先级可以通过编程来改变。

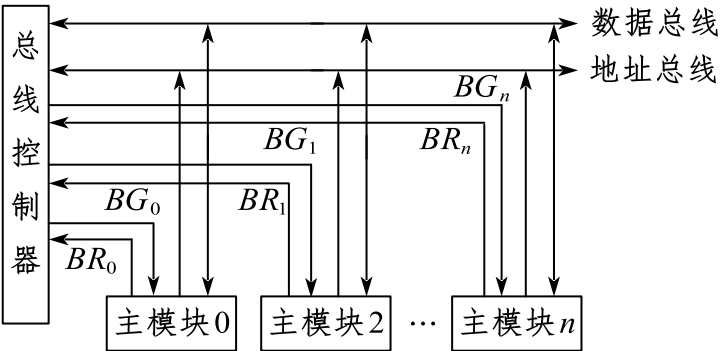
优点：优先级设置比较灵活；查询过程对电路故障敏感度较低。

缺点：控制过程相当复杂，硬性开销大。

4) 独立请求

(1) 过程

每一个主模块专用一根总线请求信号线，和一根总线允许信号线 BG_i ，各自独立地向总线控制器发出请求。总线控制器中设置并行排队线路，对各模块发来的总线请求信号 BR_i 同时进行排队判优，然后通过各自独立的总线允许信号线 BG_i ，向优先级最高的申请模块发送总线允许信号。



独立请求框图

优点：查询速度快。

缺点：控制逻辑复杂，硬性开销大。

题 4. 集中式总线仲裁方式中, () 方式对电路故障最敏感。

- A. 菊花链
- B. 独立请求
- C. 计数器定时查询
- D. 菊花链式和计数器定时查询

答案: A

解析: 菊花链方式就是链式查询方式。

题 5. 在集中式总线仲裁方式中, () 方式响应时间最快。

- A. 菊花链
- B. 独立请求
- C. 计数器定时查询
- D. 计数器定时查询和链式查询

答案: B

3. 总线事务和定时

1) 总线事务

(1) 定义

从请求总线到完成总线使用的操作序列称为总线事务, 它是在一个总线周期中发生的一系列活动。

(2) 典型的总线事务

- ①请求阶段: 主设备 (CPU 或 DMA) 发出总线传输请求, 并且获得总线控制权。
- ②仲裁阶段: 总线仲裁机构决定将下一个传输周期的总线使用权授予某个申请者。
- ③寻址阶段: 主设备通过总线给出要访问的从设备地址及有关命令, 启动从模块。
- ④传输阶段: 主模块和从模块进行数据交换, 可单向或双向进行数据传送。
- ⑤释放阶段: 主模块的相关信息均从系统总线上撤除, 让出总线使用权。

注: ①在总线事务的传输阶段, 主、从设备之间一般只能传输一个字长的数据。

②突发 (猝发) 传送方式能够进行连续成组数据的传送, 其寻址阶段发送的是连续数据单元的首地址, 在传输阶段传送多个连续单元的数据, 每个时钟周期可以传送一个字长的信息, 但是不释放总线, 直到一组数据全部传送完毕后, 再释放总线。

2) 总线定时

(1) 定义

总线定时是指总线在双方交换数据过程中需要时间上配合关系的控制，其实质是一种协议或规则。

(2) 主要分类

同步定时、异步定时、半同步定时、分离定时。

(3) 同步定时方法

①定义：指系统采用一个统一的时钟信号来协调发送和接收双方的传送定时关系。

②特点：

a. 一个总线周期中固定分配若干个时钟周期。

b. 需按照最慢速模块和最长距离来安排时钟周期。

③优点：传送速度快，具有较高的传输速率；总线控制逻辑简单。

缺点：主从设备属于强制性同步；不能及时进行数据通信的有效性检验，可靠性较差。

注：同步通信适用于总线长度较短及总线所接部件的存取时间比较接近的系统。

(4) 异步定时方法

①定义：

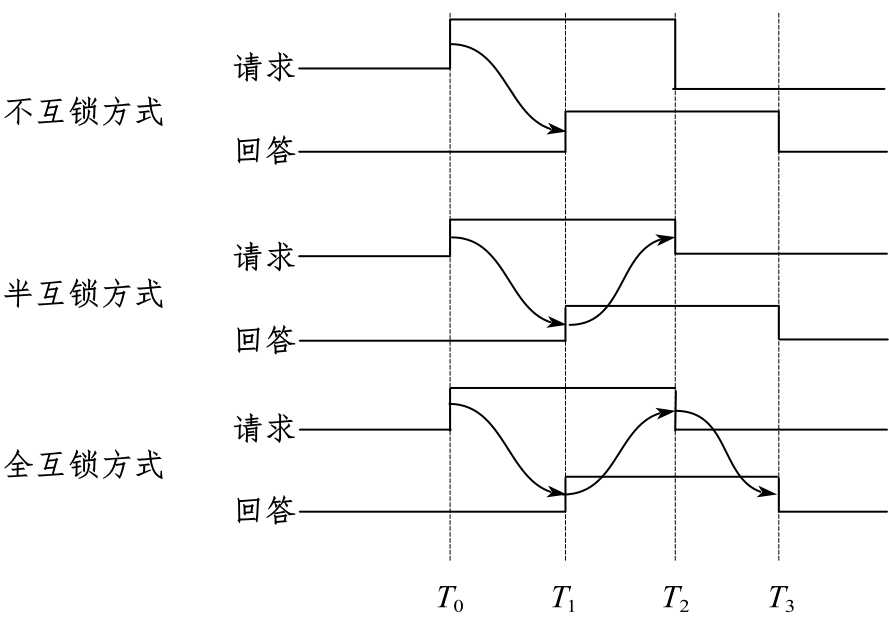
在异步定时方式中，没有统一的时钟，也没有固定的时间间隔，完全依靠传送双方相互制约的“握手”信号来实现定时控制。

②异步定时方式分为：

a. 不互锁方式：主设备发出“请求”信号后，不必等到接到从设备的“回答”信号，而是经过一段时间便撤销“请求”信号。而从设备在接到“请求”信号后，发出“回答”信号，并经过一段时间后自动撤销“回答”信号，双方不存在互锁关系。

b. 半互锁方式：主设备发出“请求”信号后，必须在接到从设备的“回答”信号后，才撤销“请求”信号，有互锁关系。而从设备在接到“请求”后，发出“回答”信号，但不必等待获知主设备的“请求”信号已经撤销，而是隔一段时间后自动撤销“回答”信号，不存在互锁关系。

c. 全互锁方式：主设备发出“请求”信号后，必须在从设备“回答”后才撤销“请求”信号；从设备发出“回答”信号后，必须在获知主设备“请求”信号已经撤销后，再撤销其“回答”信号，双方存在互锁关系。



T_0 发出请求； T_1 请求激励回答。

T_2 回答激励请求结束； T_3 请求结束激励回答结束。

课时九 练习题

1. 下面有关总线的叙述，正确的是（ ）。
 - A. 异步定时适用于各功能模块存取时间相差很大的情况
 - B. 对电路故障最敏感的仲裁方式是独立请求方式
 - C. 系统总线连接CPU和内存，而PCI总线则连接各种如键盘之类的低速I/O设备
 - D. CPU和内存之间主要采用串行传送方式传输相关数据
2. 连接在总线上的多个部件（ ）。
 - A. 只能分时向总线发送数据，并只能分时从总线接收数据
 - B. 只能分时向总线发送数据，但可同时从总线接收数据
 - C. 可同时向总线发送数据，并同时从总线接收数据
 - D. 可同时向总线发送数据，但只能分时从总线接收数据
3. 集中式总线控制中，（ ）方式对电路故障最敏感。
 - A. 链式查询
 - B. 计数器定时查询
 - C. 独立请求
 - D. 总线式
4. 同步通信之所以比异步通信具有较高的传输频率，是因为同步通信（ ）。
 - A. 不需要应答信号
 - B. 总线长度较短
 - C. 用一个公共时钟信号进行同步
 - D. 各部件存取时间比较接近
5. 假定一个同步总线的工作频率为 200MHz ，总线中有64位数据线，每个总线时钟周期传输两次数据，则该总线的最大数据传输率为（ ）。
 - A. 800MB/s
 - B. 1600MB/s
 - C. 3200MB/s
 - D. 6400MB/s
6. 假定某同步总线的工作频率为 33MHz ，总线中有32位数据线，每个总线时钟传输一次数据，则该总线的最大数据传输率为（ ）。
 - A. 66MB/s
 - B. 132MB/s
 - C. 528MB/s
 - D. 1056MB/s
7. 总线的依次数据传送过程大致分几个阶段？
8. 总线链式判优是集中式总线仲裁方案的一种，请回答以下问题：

- (1) 简述总线式判优控制方式的优缺点；
- (2) 除了链式判优外还有哪两种方案属于集中式总线仲裁？
- (3) 设总线的时钟频率为 80MHz ，一个总线周期等于一个时钟周期。如果一个总线周期中并行传送 32 位数据，求该总线的带宽。

课时十 输入输出系统

| 考点 | 重要程度 | 占分 | 题型 |
|-----------|-------|-------|-------|
| 1. 基本概念 | ★★ | 2 ~ 4 | 选择、填空 |
| 2. I/O 接口 | ★★★★ | 2 ~ 4 | 选择、填空 |
| 3. I/O 方式 | ★★★★★ | 4 ~ 8 | 选择、大题 |

1. 基本概念

1) I/O 控制方式

(1) 程序查询方式

由CPU通过程序不断查询I/O设备是否已做好准备，从而控制I/O设备与主机交换信息。

(2) 程序中断方式

只在I/O设备准备就绪并向CPU发出中断请求时才予以响应。

(3) DMA 方式

主存和I/O设备之间有一条直接数据通路，当主存和I/O设备交换信息时，无需调用中断服务程序。

(4) 通道方式

在系统中设有通道控制部件，每个通道都挂接若干外设，主机在执行I/O命令时，只需启动有关通道，通道将执行通道程序，从而完成I/O操作。

注：① 1、2方式主要用于数据传输率低的外部设备。

② 3、4方式主要用于数据传输率高的外部设备。

2) 常考的外部设备

(1) 显示器

按所用的显示器件分类，有阴极射线管（CRT）显示器、液晶显示器（LCD）、发光二极管（LED）显示器等。显示器属于用点阵方式运行的设备，有以下主要参数。

a. 屏幕大小：以对角线长度表示，常用的有12 ~ 29英寸等。

b. 分辨率：能表示的像素个数，屏幕上的每个光点就是一个像素，以宽和高的像素数的乘积表示，如 800×600 、 1024×768 和 1280×1024 等。

c. 灰度级：指黑白显示器中所显示的像素点的亮暗差别，在彩色显示器中则表现为颜色的不同，灰度级越多，图像层次越清楚、逼真，典型的有8位（256级）、16位等。

d. 刷新：光点只能保持极短的时间便会消失，为此必须在光点消失之前再重新扫描显示

一遍，这个过程称为刷新。

e. 刷新频率：指单位时间内扫描整个屏幕内容的次数。按照人的视觉生理，刷新频率大于 30Hz 时才不会感到闪烁，通常显示器的刷新频率为 $60 \sim 120\text{Hz}$ 。

f. 显示存储器（*VRAM*）：也称刷新存储器，为了不断提高刷新图像的信号，必须把一帧图像信息存储在刷新存储器中。其存储容量由图像分辨率和灰度级决定，分辨率越高，灰度级越多，刷新存储器容量越大。

$$\text{VRAM 容量} = \text{分辨率} \times \text{灰度级位数}$$

$$\text{VRAM 带宽} = \text{分辨率} \times \text{灰度级位数} \times \text{帧频}$$

题 1. 下列关于 *I/O* 指令的说法中，错误的是（ ）。

- A. *I/O* 指令是 *CPU* 系统指令的一部分
- B. *I/O* 指令是机器指令的一类
- C. *I/O* 指令反映 *CPU* 和 *I/O* 设备交换信息的特点
- D. *I/O* 指令的格式和通用指令格式相同

答案：D

解析：*I/O* 指令是指令系统的一部分，是机器指令的一类，但其为了反映与 *I/O* 设备交互的特点，格式和其他通用指令相比有所不同。

题 2. 假定一台计算机的显示存储用 *DRAM* 芯片实现，若要求显示分辨率为 1600×1200 ，颜色深度为 24 位，帧频为 85Hz ，显存总带宽的 50% 用来刷新屏幕，则需要的显存总带宽至少约为（ ）。

- A. 245Mb/s
- B. 979Mb/s
- C. 1958Mb/s
- D. 7834Mb/s

答案：D

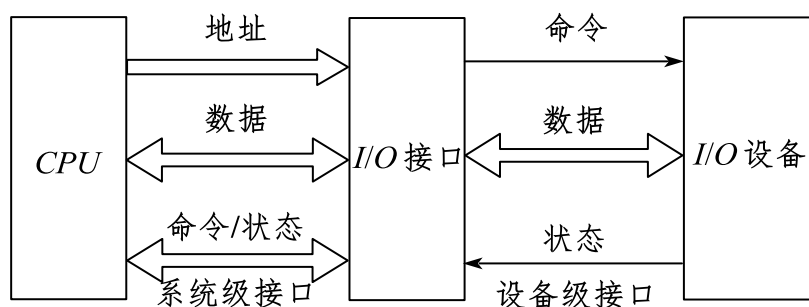
解析：刷新所需带宽 = 分辨率 \times 色深 \times 帧频 = $1600 \times 1200 \times 24 \times 85\text{Hz} = 3916.8\text{Mb/s}$ ，显存总带宽的 50% 用来刷新屏幕，于是需要的显存总带宽至少为 $3916.8 / 0.5 = 7833.6\text{Mb/s} \approx 7834\text{Mb/s}$ 。

2. *I/O* 接口

I/O 接口是主机与外设之间的交接界面，通过接口可以实现主机和外设之间的信息交换。

1) I/O 接口的功能

- (1) 通过数据缓冲寄存器，实现CPU与I/O设备之间的速度匹配。
- (2) 通过串—并（或并—串）转换电路，实现CPU与I/O设备之间的数据格式转换。
- (3) 通过电平匹配逻辑，实现CPU与I/O设备之间的电气转换。
- (4) 通过接收与传达CPU的控制命令，实现CPU对I/O设备的操作控制。
- (5) 通过保存与传送I/O设备的状态，实现CPU对I/O设备的状态查询。
- (6) 通过设备选择电路，实现CPU对I/O设备的寻址功能。



I/O 接口与CPU和I/O设备的关系

2) I/O 端口及其编址

(1) 概述

把I/O接口中每个能被CPU直接访问的寄存器称做端口，并且为每个端口都分配一个地址，称为端口地址。

(2) 端口编址方式

①统一编址：指把I/O端口当作存储器的单元进行地址分配，此方式不需要专门的I/O指令，用统一的访存指令就可访问I/O端口。

优点：不需要专门的I/O指令，使CPU访问I/O更灵活，更方便；端口有较大的编址空间。

缺点：端口占用存储器地址，使内存容量减少；执行速度较慢。

②独立编址(I/O映射方式)：I/O端口的地址空间与主存地址空间是两个独立的地址空间，需设置专门的I/O指令来访问I/O端口。

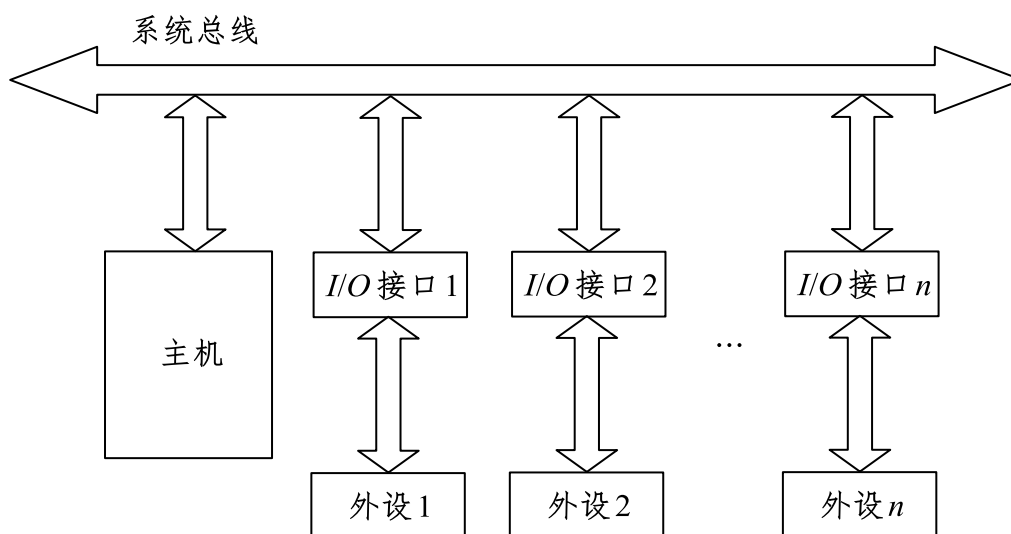
优点：程序编制清晰，便于理解。

缺点：使CPU控制变得更加复杂。

3) I/O 端口的通信方式

(1) I/O 接口与主机连接方式

I/O 接口通过系统总线与主机联络， I/O 设备和接口之间通过专用的通路连接。



以总线为中心的计算机结构

(2) I/O 接口和 I/O 设备间的数据传送方式

① 并行传送： n 位数据信息同时传送，通常按字或字节并行传送。

优点：传送速度快：

缺点：数据线用量多：

应用：适合于近距离，高速数据传送的场合。

② 串行传送：每次只传送一位二进制信息，数据从低位开始逐位传送。

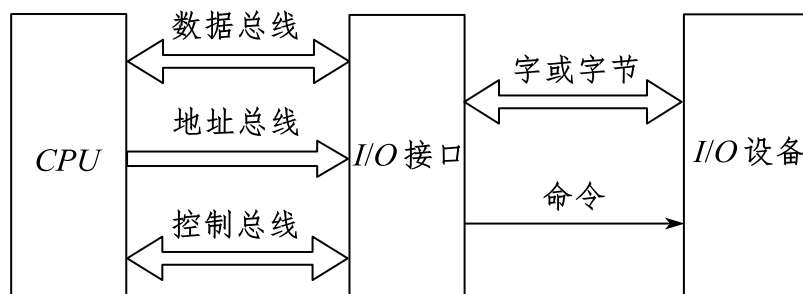
优点：只需一根数据线和一根地址线：

缺点：数据传送速度慢：

应用：适合于远距离，传送速度要求不高的场合。

(3) I/O 接口与 I/O 设备的通信方式

① 同步通信： I/O 接口与 I/O 设备之间按照统一的时钟信号进行通信，无需任何应答信号。

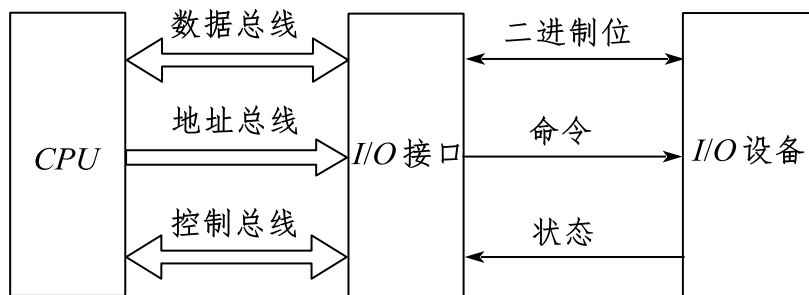


同步并行接口与 I/O 设备及 CPU 的连接方式

特点：采用该通信方式时， I/O 设备的操作时间是事先安排好的，预定的时间到了 CPU

就知道相关操作已经完成，便可进行下一个操作。

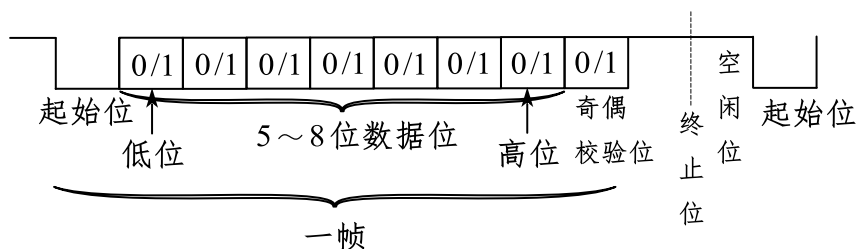
②异步通信：*I/O* 接口与 *I/O* 设备之间采用应答方式进行联络，不需设置统一的时钟信号。



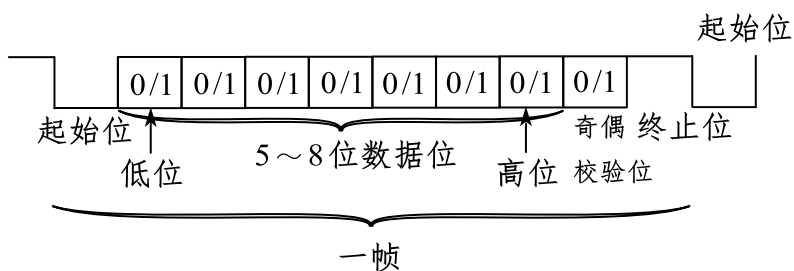
异步并行接口与 *I/O* 设备及 *CPU* 的连接方式

特点：一旦 *CPU* 通过 *I/O* 接口启动 *I/O* 设备后，*I/O* 设备通过状态反馈告知接口自己的状态，接口再将该状态告知 *CPU*。

数据传送：将一个完整的数据格式称为一个数据帧（有起始、终止标识信息）



(a) 两帧间有空闲



(b) 两帧间无空闲

注：在异步串行通信中，每一个数据都要附加起始位和终止位作为字符开始和结束标志，以至于占用了比较多的信道时间。同步串行通信是在每个数据块前附加1~2个同步字符，标识传送开始，并使收发双方同步，所以，数据传送的效率比较高。

相关计算：波特率：指单位时间内传送的二进制位表，单位为波特。

比特率：指单位时间内传送的有效数据位，单位为位/秒。

题 3. 在统一编址的方式下，区分存储单元和 *I/O* 设备是靠（ ）。

- A. 不同的地址码 B. 不同的地址线
C. 不同的控制线 D. 不同的数据线

答案：A

解析：在统一编址的情况下，没有专门的 *I/O* 指令，因此用访存指令来实现 *I/O* 操作，区分存储单元和 *I/O* 设备是靠它们各自不同的地址码。

题 4. 在异步串行传输系统中，假设每秒传输 120 个数据帧，其字符格式规定包含 1 个起始位、7 个数据位、1 个奇偶校验位、1 个终止位，试计算波特率和比特率。

解析：一帧中包含的二进制位数为 $1 + 7 + 1 + 1 = 10$ 位

$$\text{波特率} = 120 \text{ 帧/秒} \times 10 \text{ 位} = 1200 \text{ 波特}$$

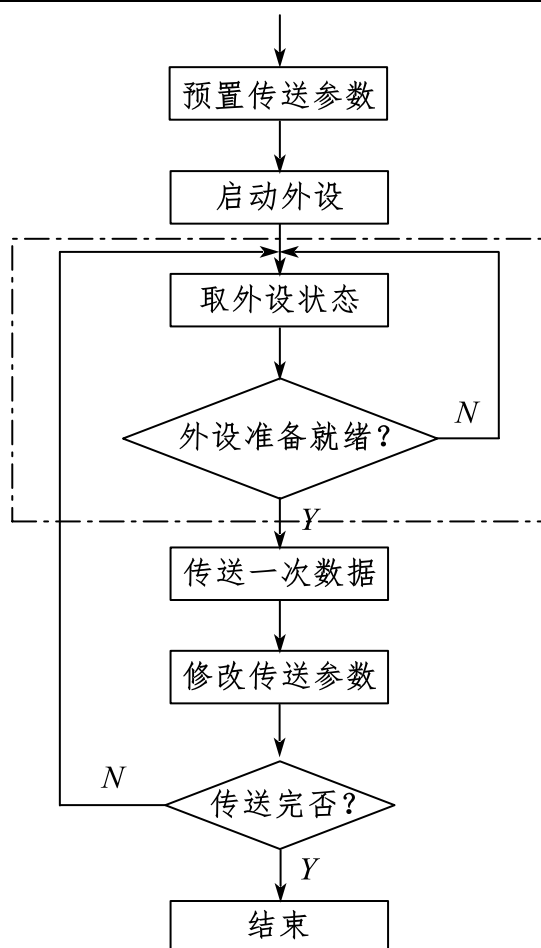
$$\text{比特率} = 120 \text{ 帧/秒} \times 7 = 840 \text{ b/s}$$

3. *I/O* 方式

1) 程序查询方式

(1) 工作流程

- ① CPU 执行初始化程序，并预置传送参数；
- ② 向 *I/O* 接口发出命令字，启动 *I/O* 设备；
- ③ 从外设接口读取其状态信息；
- ④ CPU 不断查询 *I/O* 设备状态，直到外设准备就绪；
- ⑤ 传送一次数据；
- ⑥ 修改地址和计数器参数；
- ⑦ 判断传送是否结束，若未结束转第③步，直到计数器为 0。



程序查询方式流程图

特点：①信息交换的控制完全由CPU执行程序实现；

②CPU一旦启动I/O，必须停止现行程序，并在现行程序中插入该段程序；

③CPU有“踏步”等待现象。

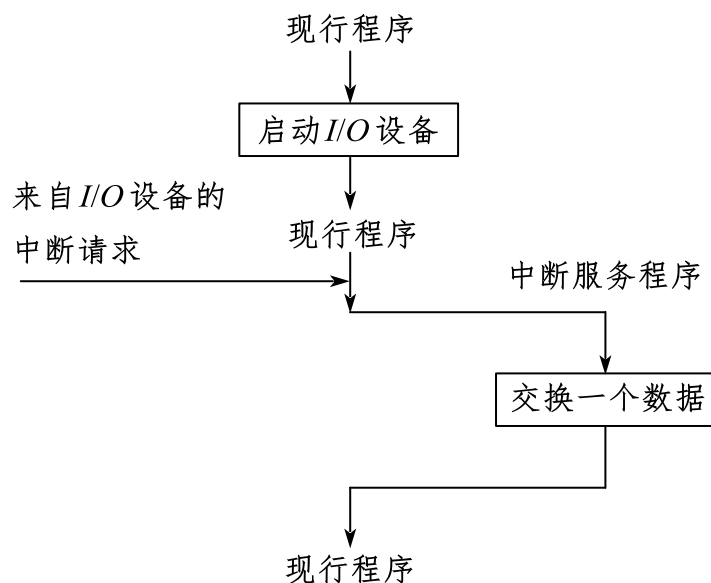
优点：接口设计简单，设备量少；

缺点：CPU存在“踏步”等待现象，且一段时间内只能和一台外设交换信息，效率大大降低。

2) 程序中断方式

(1) 工作流程

CPU启动I/O设备后，继续执行原来的程序，直到I/O接口发来中断请求为止。



用程序中断方式交换单个数据的程序流程

特点：① *I/O* 设备的准备数据过程与 *CPU* 现行程序并行执行，*CPU* 不用踏步等待：

② 外设具有申请服务的主动权。

优点：*CPU* 效率高；

缺点：硬件结构相对复杂，服务开销时间较大。

(2) 完整的中断处理过程

一次完整的中断过程依次经历5个阶段：中断请求、中断判优、中断响应、中断服务、中断返回。

① 中断请求

当 *I/O* 设备操作完成时，通过发中断请求信息来通知 *CPU*。

中断请求触发器 *INTR*：标记中断请求状态。

$INTR = 0$ ：无请求；

$INTR = 1$ ：有请求。

中断屏蔽触发器 *MASK*：由 *CPU* 执行指令来设置，决定 *CPU* 是否理睬此中断请求。

$MASK = 0$ ：表示该中断开放，*CPU* 对该接口发出的中断请求信号予以响应；

$MASK = 1$ ：表示该中断被屏蔽，*CPU* 对该接口发出的中断请求不予理睬。

中断屏蔽字：所有中断屏蔽触发器可以构成一个中断屏蔽寄存器，把 *CPU* 为中断屏蔽寄存器设置的值称为中断屏蔽字。

② 中断判优

当多个中断源同时向CPU发出中断请求时，需要按其紧急程度进行优先级排队。

排队方法：串行排队和并行排队。

注：a. 响应优先级一般为：

不可屏蔽中断 > 内部异常 > 可屏蔽中断；

内部异常中，硬件故障 > 软件中断；

DMA 中断请求优先于 I/O 设备传送的中断请求；

在 I/O 传送类中断请求中，高速设备优先于低速设备，输入设备优先于输出设备，实时设备优先于普通设备。

b. 中断优先级包括响应优先级和处理优先级，此外优先级指处理优先级。响应优先级为固定的，处理优先级可根据中断屏蔽字动态调整。

③ 中断响应

a. 中断响应的条件：

CPU 允许响应中断，即 CPU 中的中断允许触发器 $EINT = 1$ (开中断)；若 $EINT = 0$ ，表示禁止中断 (关中断) (异常和不可屏蔽中断不受此影响)；

中断源有中断请求。

b. 开始中断响应的过程为：当 $EINT = 1$ 时，CPU 在每条指令执行未发中断查询信号，对中断请求状态进行登记，若有中断请求，则发出中断响应信号 (INTA)，进入中断响应过程。

注：CPU 在一条指令执行结束响应中断的理由是：此时 CPU 现行程序的现场最简单且最稳定。

c. 中断响应的操作：

关中断。CPU 响应中断后，首先要保护程序的断点和现场信息，在保护断点和现场的过程中，CPU 不能响应更高级中断源的中断请求。否则，若断点或现场保存不完整，在中断服务程序结束后，就不能正确地恢复并继续执行现行程序。

保存断点。为保证在中断服务程序执行完后能正确地返回到原来的程序，必须将原程序的断点 (指令无法直接读取的 PC 和 PSW 的内容) 保存在栈或特定寄存器中。

注意异常和中断的差异：异常指令通常并没有执行成功，异常处理后要重新执行，所以其断点是当前指令的地址。中断的断点则是下一条指令的地址。

引出中断服务程序。识别中断源，将对应的服务程序入口地址送入程序计数器 PC。有两种方法识别中断源：硬件向量法和软件查询法。

注：这3个操作均由硬件自动完成，称之为中断隐指令。

d. 硬件向量法

中断向量：中断服务程序的入口地址

中断向量表：将系统中的全部中断向量集中存放在存储器的某区域内，称之为中断向量表。

中断类型号：每个中断都有一个唯一的类型号，每个中断类型号都对应一个中断服务程序。

具体过程：*CPU*响应中断后，通过识别中断源获得中断类型号，然后据此计算出对应中断向量的地址；再根据该地址从中断向量表中取出中断服务程序的入口地址，并送入程序计数器*PC*，以转而执行中断服务程序。

④ 中断服务

中断服务处理主要完成三项工作：保护现场和屏蔽字、中断处理和恢复现场和屏蔽字。

a. 保存现场和屏蔽字：现场信息是指用户可见的工作寄存器的内容，存放着程序执行到断点处的现行值。

注：现场信息的保存通常由软件实现；断点信息的保存通常由硬件实现。

b. 中断处理：对引起中断的事件进行处理。*I/O*中断主要完成*I/O*与主机间的数据交换。

c. 恢复现场和屏蔽字：将其恢复至原有状态。

⑤ 中断返回

中断返回由中断返回指令完成（即中断服务程序的最后一条指令）。

中断返回指令的工作内容为：

a. 将原程序断点出栈；

b. 开中断。

3) *DMA* 方式

直接存储器访问(*DMA*)是在高速*I/O*设备和主存储器间进行自动成批数据传送，而尽量减少*CPU*干预的*I/O*控制方式。

(1) *DMA* 数据交换流程

一次完整的*DMA*数据传送过程分为预处理、数据传送和后处理三个阶段。

① 预处理

*CPU*通过运行一段程序向*DMA*接口发送初始参数（如主存地址、传送字数等）和操作

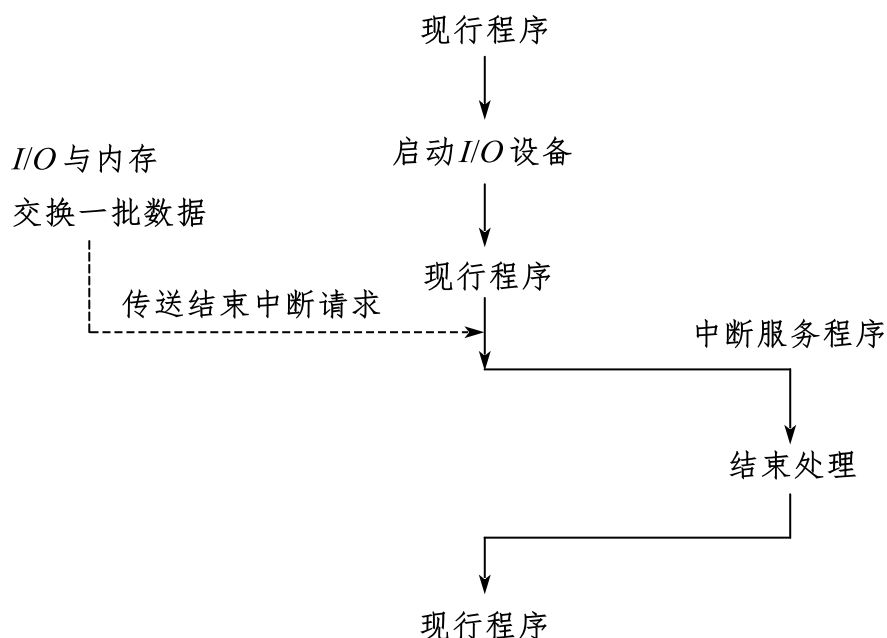
命令，之后CPU继续执行原来的程序。

②数据传送

DMA接口在I/O设备准备好一个数据后，向CPU发出总线请求，取得总线控制权后与主存进行一次数据传送。每传送完一个数据后，DMA接口修改主存地址和字数计数器值，且检查一批数据是否传送结束。若未传送结束，则继续传送；若传送结束，则向CPU发出中断请求。在此过程中，CPU完全不参与数据传送，而是继续执行原来的程序，从而实现了CPU和I/O设备之间较高的并行性。

③后处理

一批数据传送结束后，DMA接口向CPU发出DMA结束中断请求，CPU响应中断后，通过中断服务程序进行DMA数据传送的结束工作，例如数据校验、关闭I/O等。



采用DMA方式交换成批数据的程序流程

注：a. 预处理和后处理有CPU参与；

b. DMA方式通过程序进行预处理；结束时通过中断方式进行后处理。

(2) DMA接口和CPU访存冲突

当CPU在执行主程序时访问存储器与DMA接口访问存储器冲突时，有3种解决方法。

①停止CPU访问内存：当DMA接口要和主存交换数据时，CPU暂停现行程序的运行，等待DMA接口将一批数据全部传送完才继续执行原来的程序。(即总线控制权由CPU→DMA接口→CPU)。

a. 与程序查询的区别：虽然 I/O 接口与主存交换数据时，CPU 会踏步等待，但 CPU 不需要查询 I/O 设备的状态

b. 与中断方式的区别：CPU 不需要进行程序转移，没有保存现场和恢复现场的额外时间开销。

②周期挪用：当 I/O 设备有 DMA 请求时，有 3 种情况：

a. CPU 不在访存：即 I/O 访存请求与 CPU 未冲突；

b. CPU 正在访存：必须待此存取周期结束后，CPU 再将总线控制权交给 DMA 接口；

c. I/O 与 CPU 同时请求访存：I/O 优先访存，DMA 接口获得总线控制权后占用 1~2 个主存同期与主存交换一个数据，交换完就释放总线（单字传送方式）

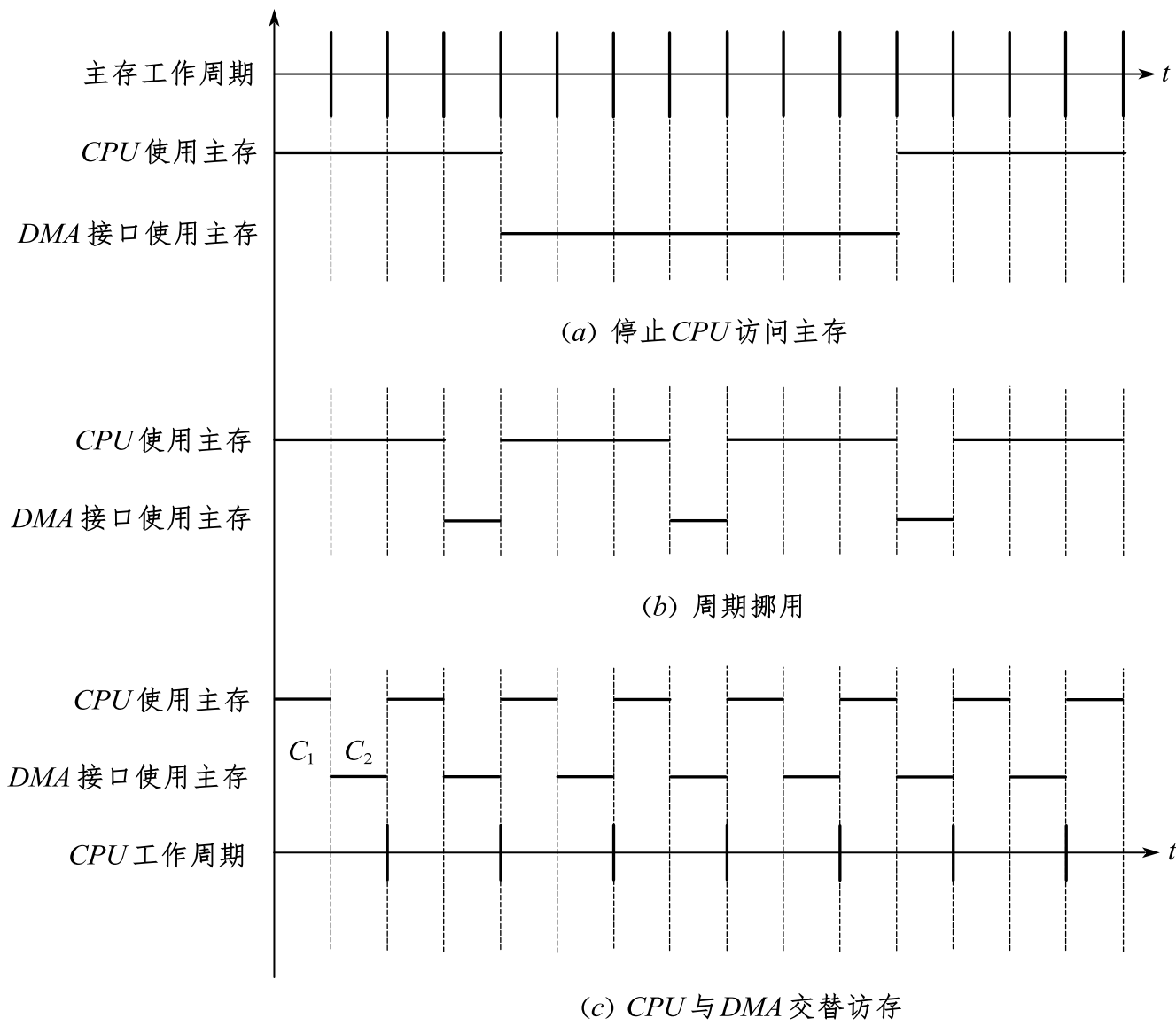
总结：CPU 不用访存时 DMA 使用；CPU 访存时 DMA 不可打断；二者同时申请时，DMA 访存。

③DMA 与 CPU 交替访存

此方法将 CPU 的工作周期比主存存取周期长一倍的情况。

例如：若 CPU 的工作周期为 $1.2\mu s$ ，主存的存取周期为 $0.6\mu s$ ，可将一个 CPU 工作周期分为 C_1 和 C_2 两个周期，其中 C_1 专供 DMA 访存， C_2 专供 CPU 访存（两者互不干扰）。

特点：此方法不需要总线使用权的申请、建立和归还过程，总线使用权是通过 C_1 和 C_2 分时控制的。



CPU和DMA接口使用主存情况

(3) DMA方式与中断方式的区别

- ①中断方式是程序的切换，需要保护和恢复现场；而DMA方式除了预处理和后处理，其他时候不占用CPU的任何资源；
- ②对中断请求的响应只能发生在每条指令执行完毕时(即指令的执行周期后)；而对DMA请求的响应可以发生在每个机器周期结束时(在取指周期、间址周期、执行周期后均可)，只要CPU不占用总线就可被响应；
- ③中断传送过程需要CPU的干预；而DMA传送过程不需要CPU的干预，因此数据传输率非常高，适合于高速外设的成组数据传送；
- ④DMA请求的优先级高于中断请求；

- ⑤中断方式具有对异常事件的处理能力，而DMA方式仅局限于传送数据块的I/O操作；
- ⑥从数据传送来看，中断方式靠程序传送，DMA方式靠硬件传送。

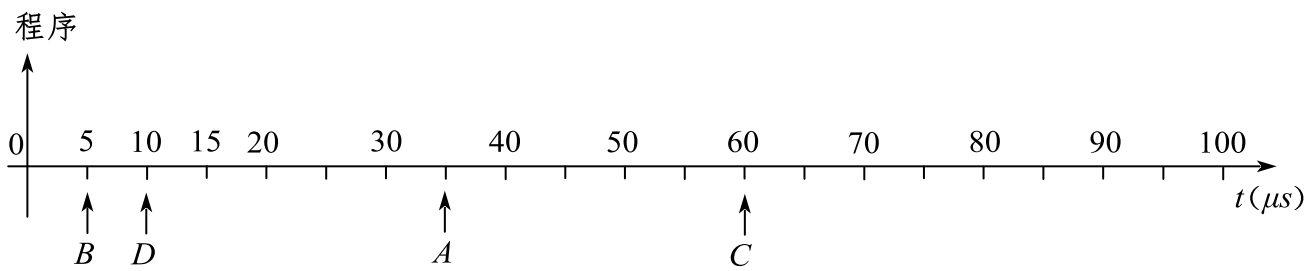
题 5. 在DMA 传送方式中，由_____发出DMA 请求。

- A. 外部设备 B. DMA 控制器 C. CPU D. 主存

答案：B

题 6. 设某机有四个中断源A、B、C、D，其硬件排队优先次序为A>B>C>D，现要求将中断处理次序改为D>A>C>B。

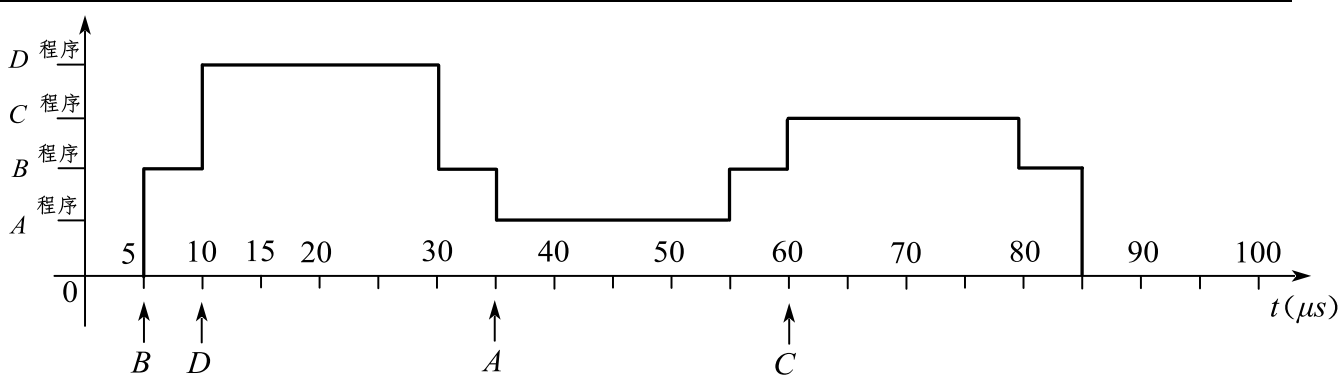
- (1) 写出每个中断源对应的屏蔽字；
- (2) 按下图时间轴给出的四个中断源的请求时刻，画出CPU 执行程序的轨迹。设每个中断源的中断服务程序时间均为20μs。



解析：(1)

| | 屏蔽字 | | | |
|-----|-----|---|---|---|
| 中断源 | A | B | C | D |
| A | 1 | 1 | 1 | 0 |
| B | 0 | 1 | 0 | 0 |
| C | 0 | 1 | 1 | 0 |
| D | 1 | 1 | 1 | 1 |

(2)



课时十 练习题

1. 下列选项中, 在 I/O 总线的数据线上传输的信息包括 ()。

I. I/O 接口中的命令字 II. I/O 接口中的状态字 III. 中断类型号

A. 仅 I、II B. 仅 I、III C. 仅 II、III D. I、II、III

2. 下面有关 I/O 接口的叙述中, 错误的是 ()。

A. 状态端口和控制端口可以合用同一个寄存器

B. I/O 接口中 CPU 可访问的寄存器称为 I/O 端口

C. 采用独立编址方式时, I/O 端口地址和主存地址可能相同

D. 采用统一编址方式时, CPU 不能用访存指令访问 I/O 端口

3. 单级中断系统中, 中断服务程序内的中断处理顺序是 ()。

I. 保护现场 II. 开中断 III. 关中断 IV. 保存断点

V. 中断事件处理 VI. 恢复现场 VII. 中断返回

A. I→V→I→II→VII B. III→I→V→VII

C. III→IV→V→VI→VII D. IV→I→V→VI→VII

4. DMA 方式的特点是 ()。

A. 与 CPU 的工作串行, 控制实现主存与高速外设之间的单个数据交换

B. 与 CPU 的工作并行, 控制实现主存与高速外设之间的成批数据交换

C. DMA 控制器通过执行程序, 控制主存与高速外设之间的单个数据交换

D. DMA 控制器通过执行程序, 控制主存与高速外设之间的成批数据交换

5. 在中断发生时, 由硬件保护并更新程序计数器 PC, 而不由软件完成, 主要是为 ()。

A. 能进入中断处理程序并能正确返回原程序

B. 节省内存

C. 使中断处理程序易于编制, 不易出错

D. 提高处理机速度

6. 下面有关 I/O 方式的叙述中, 错误的是 ()。

A. 程序查询方式和中断方式, 数据传输都通过执行指令来完成

B. DMA 方式下, 外设接口中的数据和主存单元中的内容直接交换

C. 中断 I/O 方式下, 外设接口中的数据和通用寄存器的内容直接交换

D. 中断方式下的额外开销 (额外指令执行时间) 比程序查询方式下的更小

7. 假定一台计算机的显示存储器用 DRAM 芯片实现, 若要求显示分辨率为 1600×1200 , 颜色

深度为24位,帧频为85Hz,显存总带宽的50%用来刷新屏幕,则需要的显存总带宽至少约为()。

- A.245Mbit/s B.979Mbit/s C.1958Mbit/s D.7834Mbit/s

8. 某计算机的CPU主频为500MHz, CPI为5 (即执行每条指令平均需要5个时钟周期)。假定某外设的数据传输率为0.5MB/s,采用中断方式与主机进行数据传送,以32位为传输单位,对应的中断服务程序包含18条指令,中断服务的其他开销相当于2条指令的执行时间。请回答下列问题,要求给出计算过程。

(1) 在中断方式下,CPU用于该外设I/O的时间占整个CPU时间的百分比是多少?

(2) 在该外设的数据传输率达到5MB/s时,改用DMA方式传送数据。假定每次DMA传送块大小为5000B,且DMA预处理和后处理的总开销为500个时钟周期,则CPU用于该外设I/O的时间占整个CPU时间的百分比是多少(假设DMA和CPU之间没有访存冲突)?

恭喜你完成本课程学习!

丰富校园资讯

精彩大学生生活

更多课程和学习资料

请关注公众号【蜂考】



一起学习，答疑解惑
请加蜂考学习微信群

