计算机组成原理
(总复习)

厦门大学信息学院软件工程系　曾文华

2021年6月11日

第一篇 概论

- 第 1 章　计算机系统概论
- 第 2 章　计算机的发展及应用

第二篇 计算机系统的硬件结构

- 第 3 章　系统总线
- 第 4 章　存储器
- 第 5 章　输入输出系统

第三篇 中央处理器

- 第 6 章　计算机的运算方法
- 第 7 章　指令系统
- 第 8 章　CPU 的结构和功能

第四篇 控制单元

- 第 9 章　控制单元的功能
- 第10章　控制单元的设计

第1章　计算机系统概论

主要介绍计算机的组成概貌及工作原理

1.1 计算机系统简介

1.2 计算机的基本组成

1.3 计算机硬件的主要技术指标

1.4 本书结构

第 2 章　计算机的发展及应用

简要介绍计算机的发展史以及它的应用领域

2.1 计算机的发展史

2.2 计算机的应用

2.3 计算机的展望

第 3 章　系统总线

主要介绍系统总线的
基本概念及其分类、结构和总线控制逻辑

3.1 总线的基本概念

3.2 总线的分类

3.3 总线特性及性能指标

3.4 总线结构

3.5 总线控制

本章小结（系统总线）

- 串行总线（1根）
- 并行总线（8根、16根、32根、...、等）

- 片内总线
- 系统总线（板级总线、板间总线）
  - 数据总线（DB，双向）
  - 地址总线（AB，单向）
  - 控制总线（CB，有入、有出）
- 通信总线

本章小结（系统总线）

- 总线特性：
  - 机械特性
  - 电气特性
  - 功能特性
  - 时间特性

- 总线性能指标：
  - 总线宽度：数据总线的位数
  - 总线带宽：标准传输率（**Mbps，MBps**）
  - 总线复用：数据总线与地址总线复用（AD_0-AD_7）

本章小结（系统总线）

- 总线标准：
  - **ISA**
  - **EISA**
  - **VESA（VL-BUS）**
  - **PCI：PCI Express、PCI-E 3.0、PCI-E 4.0、PCI-E 5.0**
  - **AGP**
  - **RS-232C**
  - **USB：USB 3.0、USB 3.1、USB 3.2、USB4**
  - **STD**
  - **MCA**
  - **SCSI**

本章小结（系统总线）

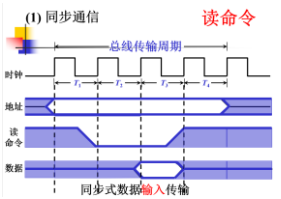
- 总线结构：
  - 单总线（系统总线）
  - 双总线（存储总线、I/O总线）
    （存储总线、系统总线）
  - 三总线（主存总线、DMA总线、I/O总线）
    （系统总线、局部总线、扩展总线）
  - 四总线（系统总线、局部总线、扩展总线、高速总线）

本章小结（系统总线）

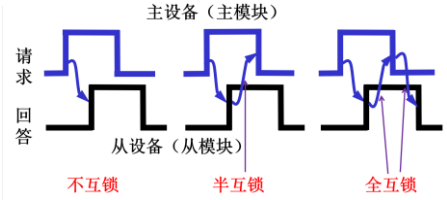
- 总线控制：
  - 总线判优控制：主要解决多个设备同时申请总线时的使用权分配问题，即决定哪个设备占用总线
    - 集中式
      - 链式查询：仅用3根线确定总线使用权属于哪个设备；优点是线路最少，但是对电路故障敏感。
      - 计数器定时查询：需要用$2+int(\log_2 n)$根线(如$n=16$，$\log_2 n=4$，则需要6根线)。
      - 独立请求方式：需采用$2n$根线；优点是响应时间快，但是线路最多。
    - 分布式

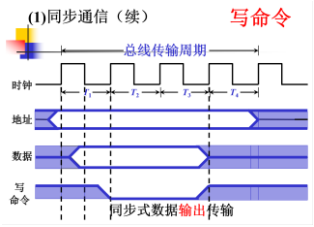
本章小结（系统总线）

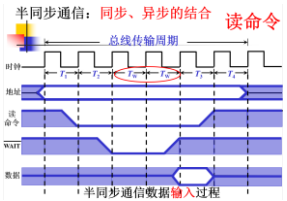
- **总线通信控制**：主要解决通信双方如何获知传输开始和传输结束，以及通信双方如何协调配合



  - 同步通信（1个总线传输周期分为4个时钟周期）
  - 异步通信
    - 不互锁方式
    - 半互锁方式
    - 全互锁方式







  - 半同步通信（同步和异步的结合，插入若干个T_W）
  - 分离式通信（将1个总线传输周期分解为2个子周期）

本章小结（系统总线）

- 总线周期（分为4个阶段）：
  - ①申请分配阶段
  - ②寻址阶段
  - ③传数阶段
  - ④结束阶段

例3.1：计算总线的数据传输率（带宽）

例3.2：计算波特率

例3.3：异步串行传输方式数据波形

例3.4：计算比特率

- 波特率、比特率：**bps**

- 时序图

第4章 存储器

4.1 概述

4.2 主存储器

4.3 高速缓冲存储器

4.4 辅助存储器

本章小结（存储器）

- 存储器分类
- 存储器层次结构
  - 缓冲-主存层次（Cache）：解决速度问题
  - 主存-辅存层次（虚拟存储器）：解决容量问题
  - 主存和缓存之间的数据调动是由硬件自动完成的，对系统程序员和应用程序员都是透明的
  - 主存和辅存之间的数据调动是由硬件和操作系统共同完成的，对系统程序员是不透明的、对应用程序员是透明的

本章小结（存储器）

- 主存储器
  - 主存的技术指标：
    - 存储容量
    - 存储速度
      - 存取时间
      - 存取周期
    - 存储器带宽
  - 提高存储器带宽的措施：
    - 缩短存取周期(MCT)
    - 增加存储字长，使每个存取周期可读/写更多的二进制位数（8位、16位、32位、……）
    - 增加存储体（见教材的4.2.7小节）
  - 半导体存储芯片的译码驱动方式：
    - 线选法
    - 重合法

本章小结（存储器）

- **RAM**
  - **SRAM**
    - **6个MOS管共同构成一个基本单元电路（存放1位二进制数）**
  - **DRAM**
    - 三管
    - 单管
  - **DRAM的刷新**
    - 集中刷新
    - 分散刷新
    - 异步刷新
  - **SRAM与DRAM的比较**

本章小结（存储器）

- **ROM**
  - **Mask ROM（掩模ROM）**
  - **PROM**
  - **EPROM**
  - **EEPROM**
  - **Flash ROM**
- **存储器与CPU的连接：难点是片选信号的产生**
  - **例4.1**
  - **例4.2**
  - **例4.3：按字访问，按字节访问（奇字节、偶字节）**
- **存储器的校验（汉明码）**

本章小结（存储器）

- 提高访存速度的措施
  - 高速器件（高性能存储芯片）
    - **SDRAM（DDR SDRAM）**
    - **RDRAM**
    - **CDRAM**
  - 采用层次结构（**Cache-**主存）
  - 调整主存结构
    - 单体多字
    - 多体并行
      - 高位交叉
      - 低位交叉
    - 例**4.6**：顺序存储与交叉存储比较（存储器带宽的比较）

本章小结（存储器）

- 高速缓冲存储器（Cache）
  - Cache是利用程序访问的局部性原理
  - Cache的命中率、效率、平均访问时间
  - Cache的基本结构（Cache存储体、地址映射变换机构、替换机构）
  - Cache的读写操作（写直达法、写回法）
  - Cache-主存地址映射
    - 直接映射
    - 全相联映射
    - 组相联映射
  - 例4.9：直接映射、全相联映射、二路组相联映射方式下，画出主存地址格式
  - 例4.10：计算平均访问时间
  - 例4.11：画出主存地址格式，计算命中率、平均访问时间、效率

本章小结（存储器）

- 辅助存储器
  - 磁表面存储器的主要技术指标
  - 磁表面存储器的记录方式
  - 硬盘存储器（HDD、HDC）
  - RAID：独立磁盘冗余阵列、廉价磁盘冗余阵列
  - 软盘存储器（FDD、FDC）
  - 磁带存储器
  - 光盘存储器（CD-ROM、CD-R、CD-RW、MO）
  - 循环冗余校验码（CRC）

第5章　输入输出系统

5.1 概述

5.2 外部设备

5.3 I/O接口

5.4 程序查询方式

5.5 程序中断方式

5.6 DMA方式

输入输出系统：I/O系统

外部设备：I/O设备

I/O系统=I/O接口+I/O设备

本章小结（输入输出系统）

- I/O系统（输入输出系统）=I/O接口 + I/O设备（输入输出设备）
- I/O 设备编址方式：
  - (1) 统一编址
  - (2) 不统一编址
- I/O设备与主机之间的联络方式：
  - (1) 立即响应方式
  - (2) 异步工作采用应答信号联络（异步并行，异步串行）
  - (3) 同步工作采用同步时标联络
- I/O设备与主机信息传送的控制方式（常用的是前3种）：
  - 1. 程序查询方式
  - 2. 程序中断方式
  - 3. DMA 方式
  - 4.I/O通道方式
  - 5.I/O处理机方式

本章小结（输入输出系统）

- **I/O接口的功能和组成**

I/O接口的功能

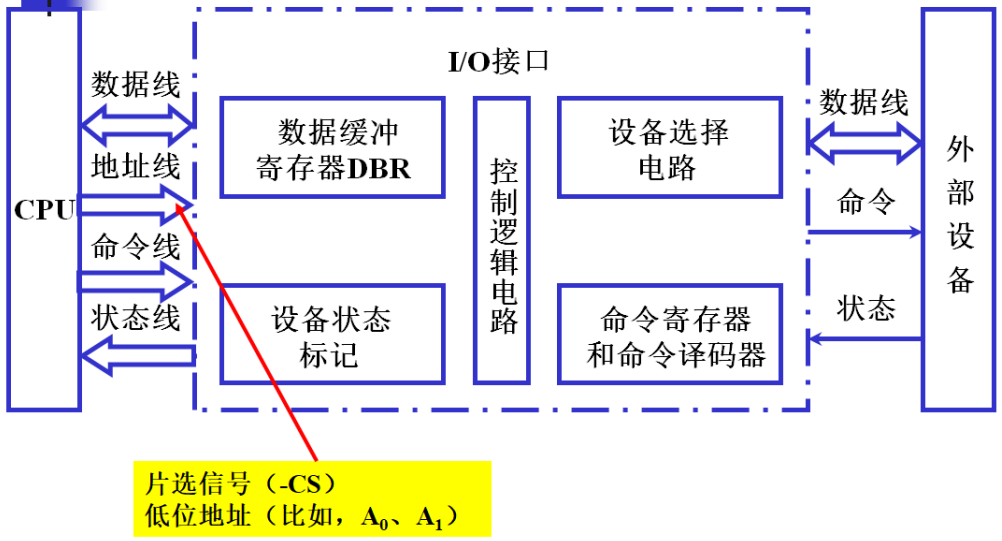
选址功能

传送命令的功能

传送数据的功能

反映设备状态的功能

I/O 接口的基本组成



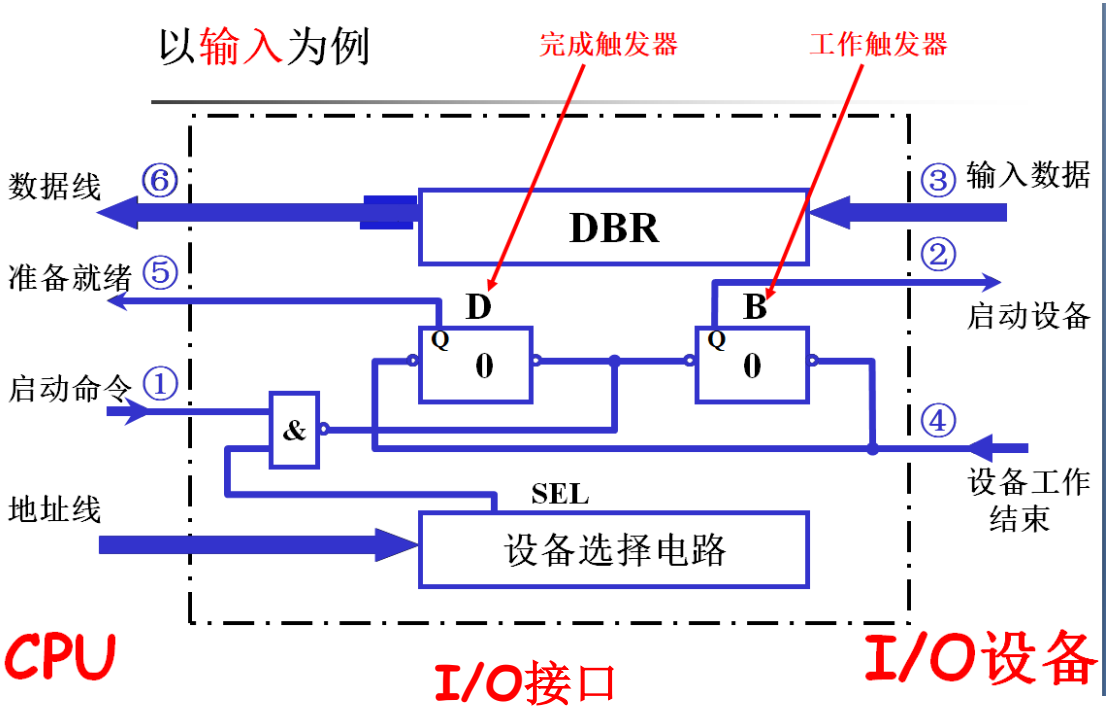
本章小结（输入输出系统）

- **I/O接口的类型**

1. 按数据 传送方式 分类
   - 并行接口　Intel 8255
   - 串行接口　Intel 8251、8250
2. 按功能 选择的灵活性 分类
   - 可编程接口　Intel 8255、 Intel 8251
   - 不可编程接口　Intel 8212　← 通用并行接口 数据输入锁存器
3. 按 通用性 分类
   - 通用接口　Intel 8255、 Intel 8251
   - 专用接口　Intel 8279、 Intel 8275　← 可编程CRT控制器接口
4. 按数据传送的 控制方式 分类
   - 中断接口　Intel 8259
   - DMA 接口　Intel 8257、8237

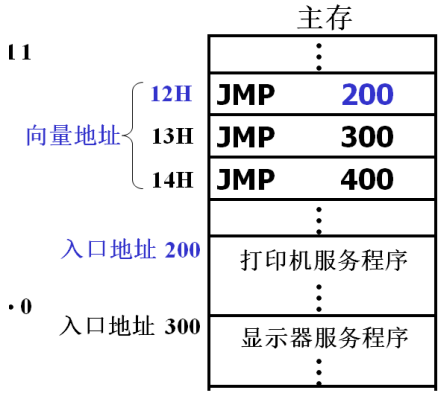
本章小结（输入输出系统）

- 程序查询方式的接口电路



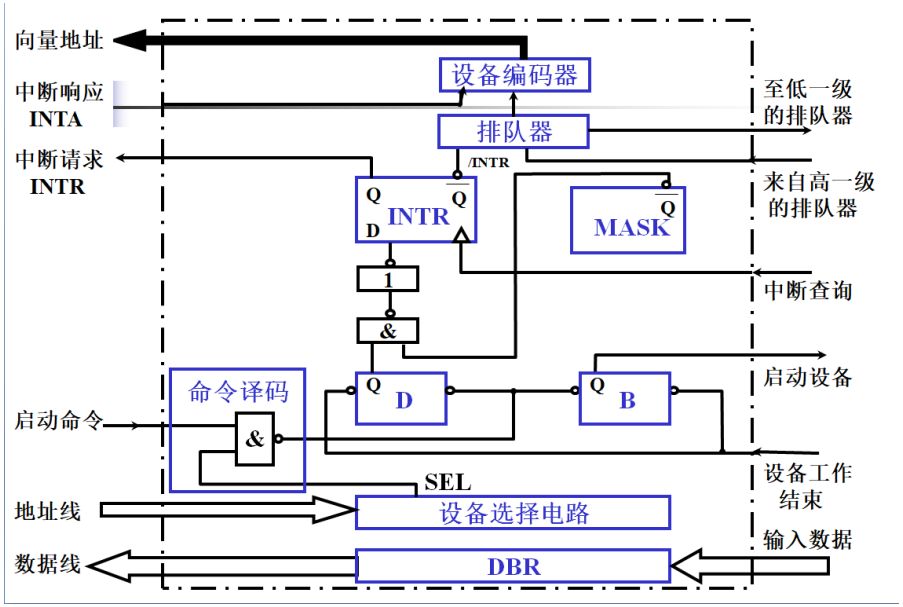
本章小结（输入输出系统）

- 程序中断方式
  - **INTR：中断请求信号（I/O送给CPU的）**
  - **INTP：中断优先级信号（排队器的输出）**
  - **INTA：中断响应信号（CPU送给I/O的）**
  - **EINT：中断允许信号（由中断指令控制，开中断指令EI使EINT=1，关中断指令DI使EINT=0）**
  - **MASK：中断屏蔽信号（MASK=0，表示没有屏蔽；MASK=1，表示有屏蔽）**
  - **向量地址（中断向量地址，中断向量）**
  - **入口地址（中断服务程序入口地址，中断服务程序首地址）**



本章小结（输入输出系统）

- 程序中断方式接口电路的基本组成



- 中断处理过程

1. 中断请求：**INTR=1**
2. 中断判优：排队器的**INTP=1**
3. 中断响应：**CPU**允许中断，则**INTA=1**，产生中断向量地址，形成中断服务地址
4. 中断服务：如完成数据的输入
5. 中断返回：通过**IRET**指令，返回到原程序的断点处

中断处理过程细分的化为10步（书上P198-199）

自下而上由4部分构成：图5.35、图5.37、图5.38、图5.39

本章小结（输入输出系统）

- **CPU响应中断的条件和时间：**
  - 条件：**CPU中的EINT（中断允许触发器）为“1”。注：EINT可以通过开中断指令（STI）置“1”，关中断指令（CLI）置“0”。**
  - 时间：**在每条指令执行阶段的结束时刻，CPU向I/O接口发中断查询信号，以获取I/O的中断请求（INTR）。**

本章小结（输入输出系统）

- 中断服务程序的流程

(1) 保护现场

{ 程序断点的保护　　中断隐指令完成
{ 寄存器内容的保护　进栈指令**PUSH**

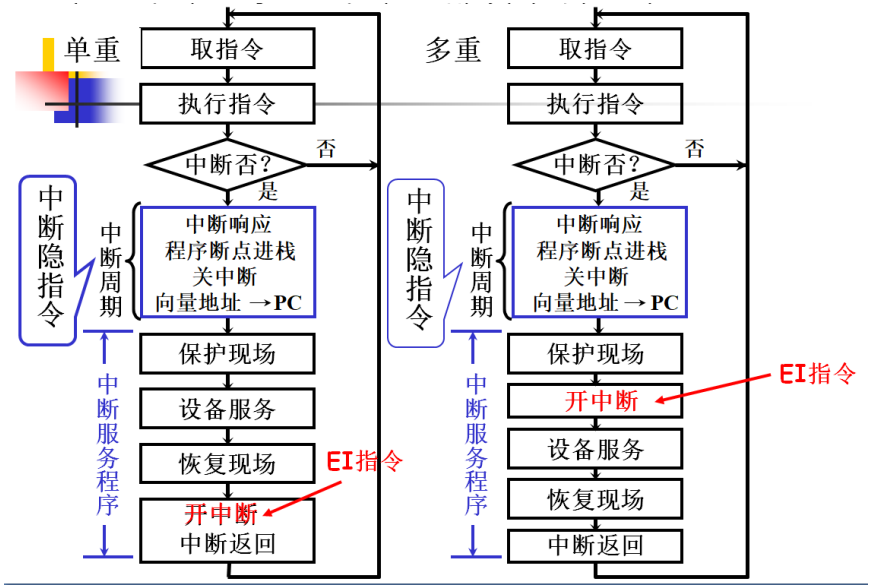
(2) 中断服务

对不同的 **I/O** 设备具有不同内容的设备服务

(3) 恢复现场　　出栈指令**POP**

(4) 中断返回　　中断返回指令**IRET**

- 单重中断、多重中断（中断嵌套）



本章小结（输入输出系统）

- DMA 与主存交换数据的三种方式
  - (1) 停止 CPU 访问主存
  - (2) 周期挪用（或周期窃取）
  - (3) DMA 与 CPU 交替访问

- DMA 接口的功能

(1) 向 CPU 申请 DMA 传送

(2) 处理总线 控制权的转交

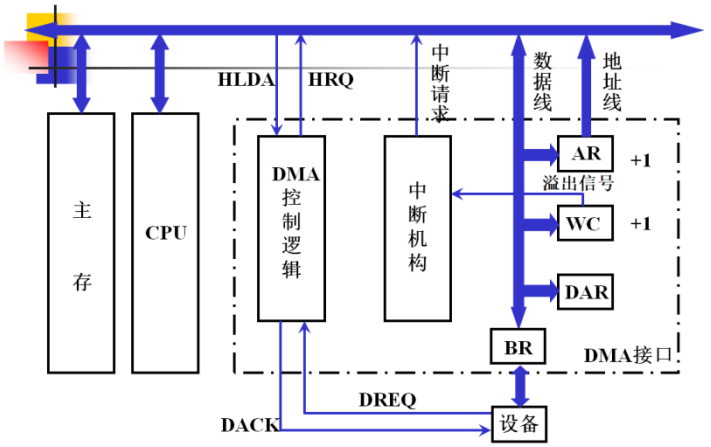
(3) 管理 系统总线、控制 数据传送

(4) 确定 数据传送的 首地址和长度

修正 传送过程中的数据 地址 和 长度

(5) DMA 传送结束时，给出操作完成信号

- DMA 接口的组成



- DREQ：外设向DMAC发出的DMA请求信号
- HRQ：DMAC向CPU发出的总线请求信号
- HLDA：CPU向DMAC发出的总线响应信号
- DACK：DMAC向外设发出的DMA应答信号

本章小结（输入输出系统）

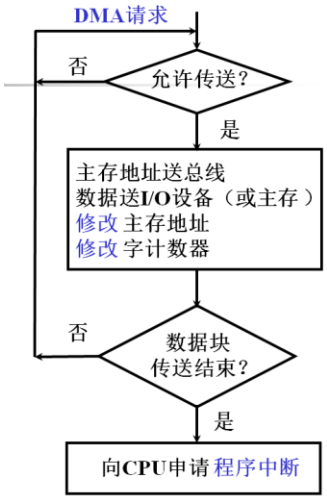
- **DMA 传送过程：预处理、数据传送、后处理**

(1) 预处理

通过几条输入输出指令预置如下信息

- 通知 **DMA** 控制逻辑传送方向（入/出）
- 设备地址——→**DMA** 的 **DAR**（I/O地址）
- 主存地址——→**DMA** 的 **AR**
- 传送字数——→**DMA** 的 **WC**

（2）数据传送



(3) 后处理

校验送入主存的数是否正确

是否继续用 **DMA**

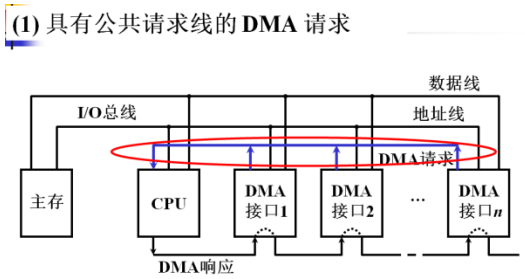
测试传送过程是否正确，如有错则转诊断程序

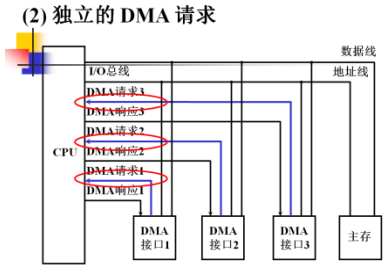
由中断服务程序完成

DMA数据传送细分的化为10步（书上P207-208）

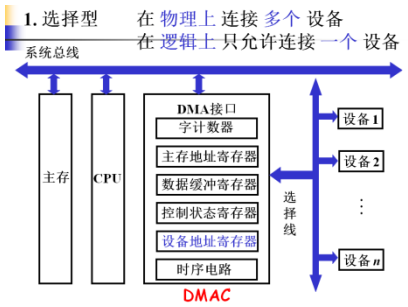
本章小结（输入输出系统）

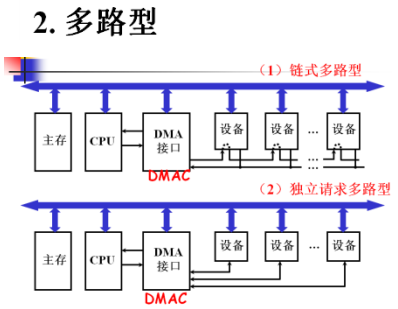
- **DMA 接口与系统的连接方式：**
  - **(1) 具有公共请求线的 DMA 请求**
  - **(2) 独立的 DMA 请求**





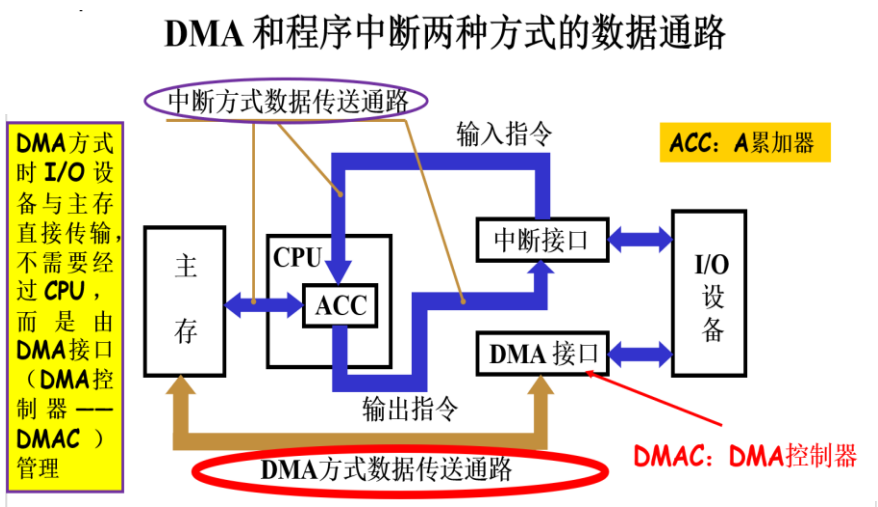
- **DMA 接口的类型：**
  - **1. 选择型**
  - **2. 多路型**
    - （1）链式多路型
    - （2）独立请求多路型





本章小结（输入输出系统）

- **DMA方式与中断方式的比较：**



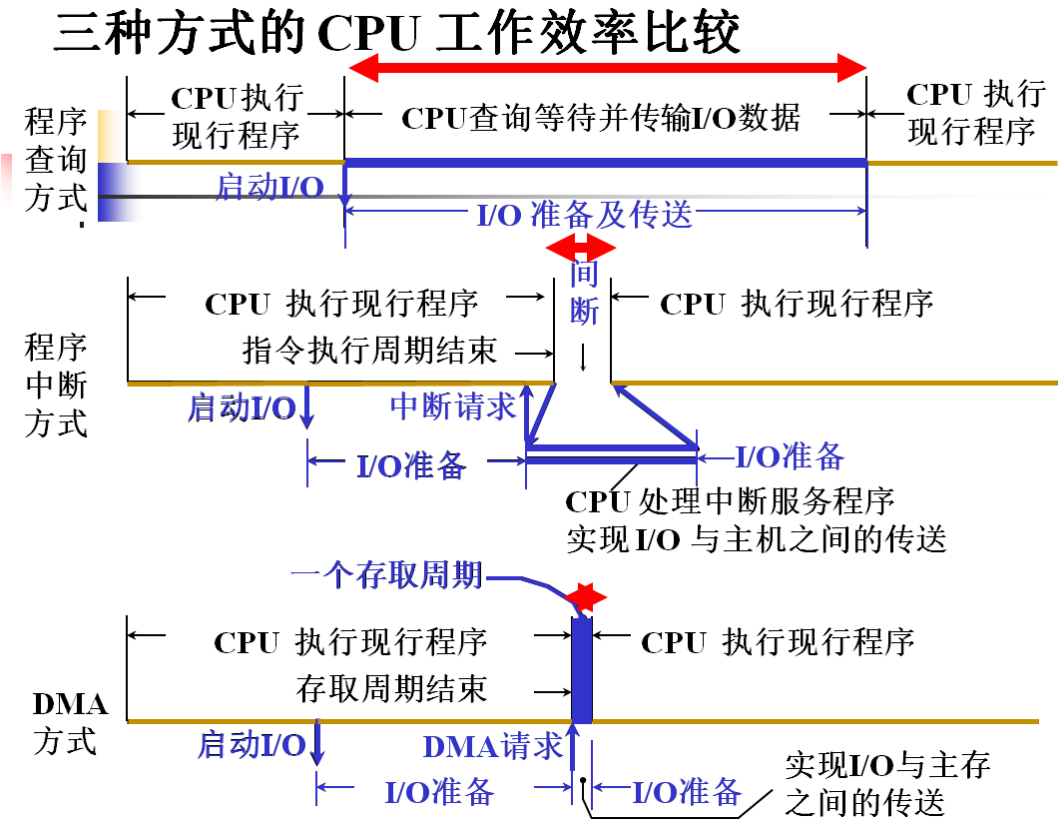
DMA控制器（DMAC）

	中断方式	DMA 方式
(1) 数据传送	程序	硬件
(2) 响应时间	指令执行结束	存取周期结束
(3) 处理异常情况	能	不能
(4) 中断请求	传送数据	后处理
(5) 优先级	低	高

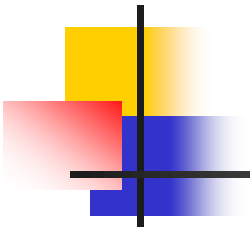
周期挪用（或周期窃取）

本章小结（输入输出系统）

- 查询方式、中断方式、DMA方式的CPU工作效率比较：



第 6 章　计算机的运算方法



6.1 无符号数和有符号数

6.2 数的定点表示和浮点表示

6.3 定点运算

6.4 浮点四则运算

6.5 算术逻辑单元（ALU）

本章小结（计算机的运算方法）

- 无符号数
  - 8位：00000000 ~ 11111111 （0-255）
- 有符号数
  - 真值
  - 机器码
    - 原码
    - 补码
    - 反码
    - 移码

- 正数； 原码、补码、反码 都一样
- 补码 = 原码取反加1
- 移码 = 补码的符号位取反
- 0：补码、移码唯一，原码、反码有二个
- 小数没有移码
- 原码（8位）：-127 ~ +127
- 补码（8位）：-128 ~ +127
- 反码（8位）：-127 ~ +127
- 移码（8位）：-128 ~ +127

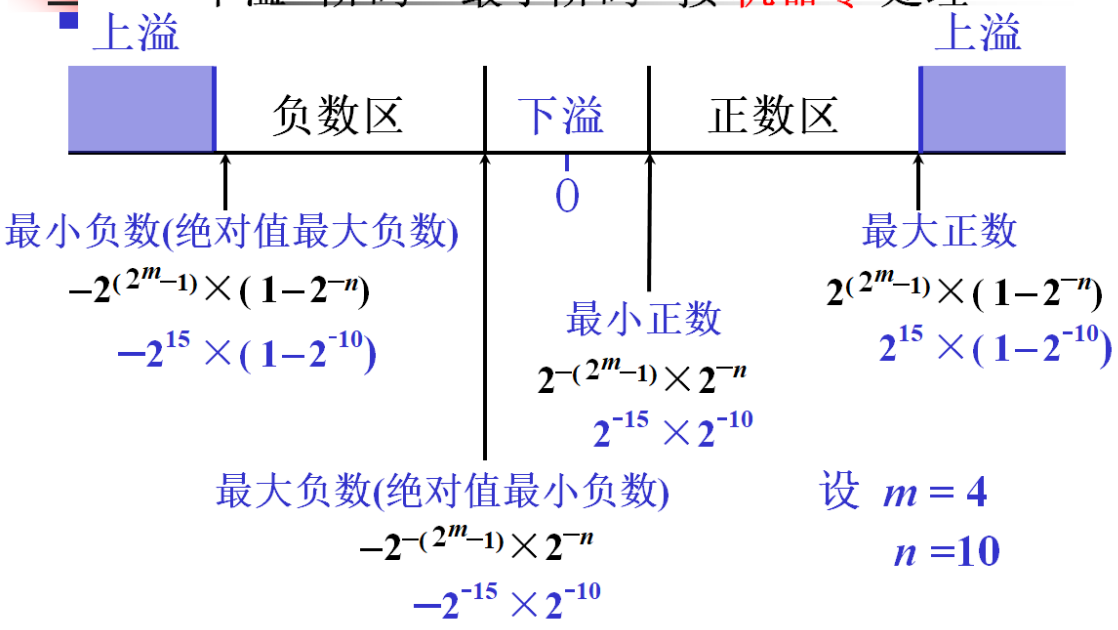
本章小结（计算机的运算方法）

- 定点数　　8位 = 1位符号位 + 7位有效位
  - 纯小数：0.101 0110　　1.010 1001
  - 纯整数：0,101 0110　　1,010 1001

- 浮点数
  - 浮点数的定义
    - 16位 = 1位阶符 + 4位阶码数值部分 + 1位尾数符号 + 10位尾数数值部分
  - 浮点数的表示范围
    - 上溢：阶码是一个很大的正数，超出表示范围（比如>+127）
    - 下溢（当0处理）：阶码是一个很小的负数，超出表示范围（比如<-128）
  - 浮点数的规格化
    - 判断尾数是不是规格化?（尾数绝对值>1，或尾数绝对值<0.5）
    - 对非规格化的尾数进行左规（尾数绝对值<0.5时，尾数左移1位，阶码减1），或右规（尾数绝对值>1时，尾数右移1位，阶码加1），使其变为规格化的浮点数

上溢　阶码＞最大阶码

下溢　阶码＜最小阶码　按 机器零 处理



浮点数的表示范围

■尾数：

■正数：

■原码、反码、补码：　0（符号位）　1（最高有效位）

X=0.1100	$X_{原}$=0.1100	$X_{补}$=0.1100	$X_{反}$=0.1100

■负数：

■原码：　1（符号位）　1（最高有效位）

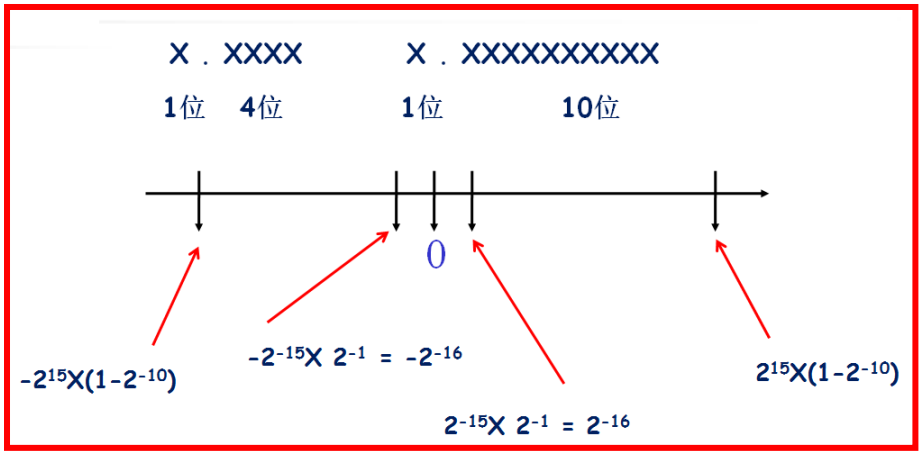
■反码、补码：　1（符号位）　0（最高有效位）

X=-0.1100	$X_{原}$=1.1100	$X_{补}$=1.0100	$X_{反}$=1.0011

判断尾数是不是规格化？

本章小结（计算机的运算方法）

- 浮点数
  - 规格化的浮点数的表示范围

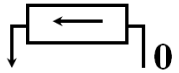
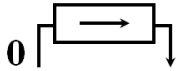


  - 机器零（浮点数）：当阶码用移码表示、尾数用补码表示时；如果尾数全为0（包括符号位），或者阶码全为0（包括符号位），则为机器零

  - IEEE 754 标准： 符号位 *S* 阶码（含阶符） 尾数

本章小结（计算机的运算方法）

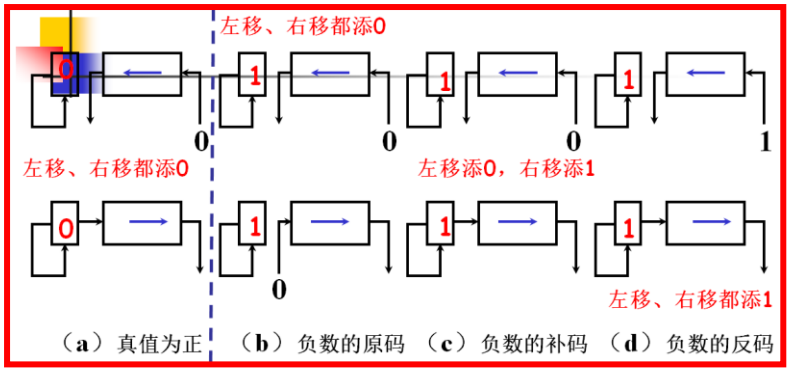
- 定点运算（定点数的运算）
  - 逻辑运算（书上没有讲）：与、或、非、与非、或非、异或
  - 移位运算
    - 逻辑移位（左移、右移、循环移位——循环左移和循环右移）

逻辑左移	低位添 0，高位移丢	
逻辑右移	高位添 0，低位移丢	

- 算术移位（移码的算术移位规则同补码）

算术左移

算术右移



本章小结（计算机的运算方法）

- 定点运算（加减运算）
  - 加法运算
  - 减法运算：**A-B = A + [-B] = A + B取反 + 1**
  - 溢出
    - 什么是溢出？
      - 结果超出表示范围（8位）　　-128 ～ +127
      - 正溢出：　正数 + 正数　　正数 - 负数
      - 负溢出：　负数 + 负数　　负数 - 正数
    - 怎么判断运算结果是否溢出？
      - 一位符号位判断溢出：

        最高有效位的进位 ⊕ 符号位的进位 = 1　　溢出

        如　$1 \oplus 0 = 1$, $0 \oplus 1 = 1$ } 有 溢出

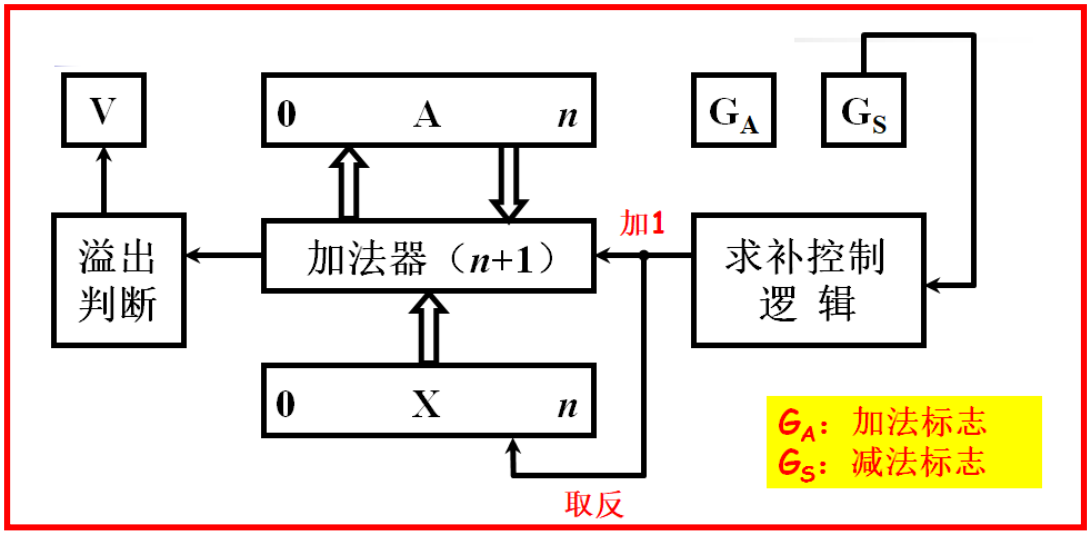
        $0 \oplus 0 = 0$, $1 \oplus 1 = 0$ } 无 溢出
      - 两位符号位判断溢出（变形补码）：

        结果的双符号位 相同　　未溢出　　00, ×××××　11, ×××××

        结果的双符号位 不同　　溢出　　10, ×××××　01, ×××××

本章小结（计算机的运算方法）

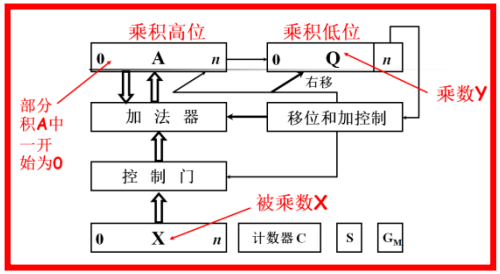
- 定点运算（加减运算）
  - 补码加减法运算部件



A - X = A + [-X] = A + X取反 + 1 -> A

本章小结（计算机的运算方法）

- 定点运算（乘法运算：加法、右移）
  - 原码一位乘法
    - 数值部分（绝对值）相乘；符号位单独处理——异或
    - 原码一位乘的硬件配置

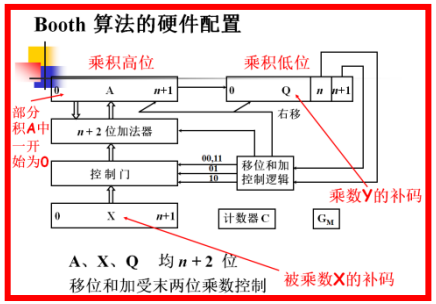


  - 原码二位乘法
    - 数值部分（绝对值）相乘；符号位单独处理——异或
    - 根据乘数的后2位及标志位的不同情况（8种情况），进行操作

乘数判断位 $y_{n-1}y_n$	标志位 C_j	操 作 内 容
0 0	0	$z\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 保持“0”
0 1	0	$z+x^*\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 保持“0”
1 0	0	$z+2x^*\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 保持“0”
1 1	0	$z-x^*\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 置“1”
0 0	1	$z+x^*\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 置“0”
0 1	1	$z+2x^*\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 置“0”
1 0	1	$z-x^*\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 保持“1”
1 1	1	$z\rightarrow 2,\ y^*\rightarrow 2,\ C_j$ 保持“1”

本章小结（计算机的运算方法）

- 定点运算（乘法运算：加法、右移）
  - 补码一位乘法（乘积的符号位自然形成）
    - （1）被乘数符号任意，乘数为正数：同原码乘
    - （2）被乘数符号任意，乘数为负数：校正法（最后要加$[-x]_{补}$）
    - （3）被乘数符号任意，乘数符号任意： Booth 算法（比较法，4种情况，3种操作）
    - Booth算法的硬件配置



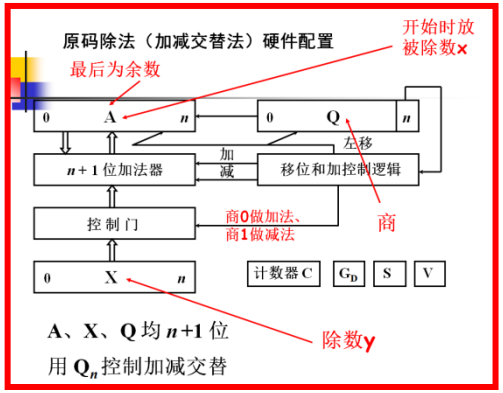
y_i y_{i+1}	$y_{i+1}-y_i$	操作
0 0	0	-1
0 1	1	$+[x]_{补}$ -1
1 0	-1	$+[-x]_{补}-1$
1 1	0	-1

  - 补码二位乘法
    - 根据$y_{n-1}y_ny_{n+1}$进行判断（8种情况）

判断位 $y_{n-1}y_ny_{n+1}$	操作内容
000	$[z_{i+1}]_{补}=2^{-2}[z_i]_{补}$
001	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+[x]_{补}\}$
010	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+[x]_{补}\}$
011	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+2[x]_{补}\}$
100	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+2[-x]_{补}\}$
101	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+[-x]_{补}\}$
110	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+[-x]_{补}\}$
111	$[z_{i+1}]_{补}=2^{-2}[z_i]_{补}$

本章小结（计算机的运算方法）

- 定点运算（除法运算：减法、左移）
  - 原码一位除法
    - （1）恢复余数法：需要恢复余数
    - （2）加减交替法（不恢复余数法）：不需要恢复余数；一会儿加除数，一会儿减除数，故称为加减交替法
    - 加减交替法硬件配置



  - 补码一位除法
    - 加减交替法：并且末位恒置1（余数会产生误差，最大误差为2^{-n}）

本章小结（计算机的运算方法）

- 浮点运算（加减运算，共**5**步，实际上是前**3**步）

1. 对阶：小阶向大阶对齐
   - 小阶的尾数右移**1**次，阶码加**1**，直到与大阶的阶码相同
   - 小阶尾数右移时，可能需要进行舍入操作（ **0**舍**1** 入）
2. 尾数求和：加减——补码加减运算
3. 尾数规格化
   - 左规（尾数绝对值**<0.5**）：尾数左移**1**位，阶码减**1**；可能需要左规多次，直到尾数变为规格化的数
   - 右规（尾数绝对值**>1**）：尾数右移**1**位，阶码加**1**；只需右规**1**次
4. 舍入：尾数右规时（对阶、规格化）可能需要进行舍入操作——**0**舍**1**入
5. 溢出判断：上溢，下溢（当**0**处理）

例6.29：浮点加法运算

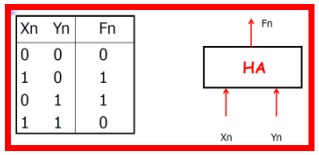
例6.30：浮点减法运算

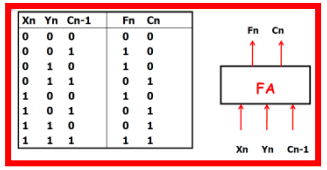
本章小结（计算机的运算方法）

- 浮点运算（乘除运算）
  - 浮点乘法运算：尾数相乘，阶码相加
  - 浮点除法运算：尾数相除，阶码相减
- 协处理器（**NPU，FPU**）：**8087，80287，80387**
- 算术逻辑部件（**ALU，Arithmetic Logic Unit**）
  - 74LS181：4位运算器
    - 输入：A（4位），B（4位），C_{-1}（来自低位的进位）
    - 输出：F（4位），C_{n+4}（送到高位的进位）　　A=B　　G，P（用于超前进位，送到74LS182）
    - 控制信号：S（4位，16种情况），M（0：算术运算，1：逻辑运算）
  - 74LS182：超前进位器件
  - 29C101：16位运算器
  - AM2901：4位运算器（功能类似于29C101，但只是4位运算器）
  - AM2902：超前进位器件（功能类似于74LS182）

本章小结（计算机的运算方法）

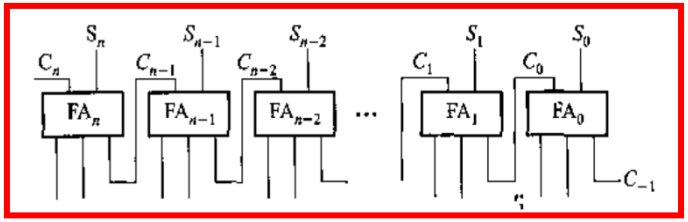
- 快速进位链（超前进位）
  - 半加器（**HA**）
  - 全加器（**FA**）





- 并行加法器（**4**位，串行进位）：产生全部进位的时间为**4*2=8ty**

图6-19



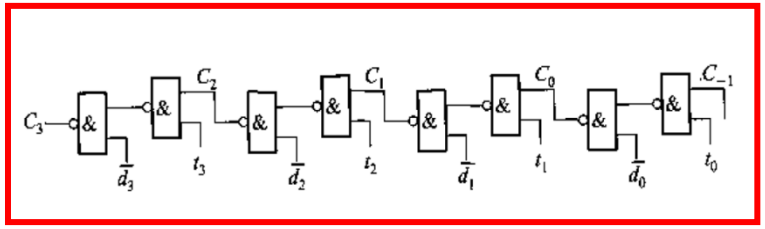
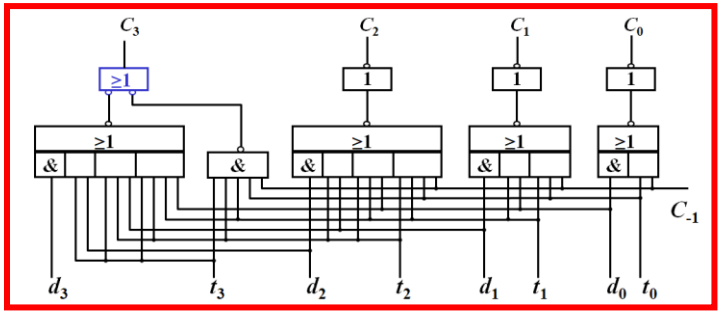


图6-20

- 并行加法器（**4**位，先行进位）：产生全部进位的时间为**2.5ty**

图6-21



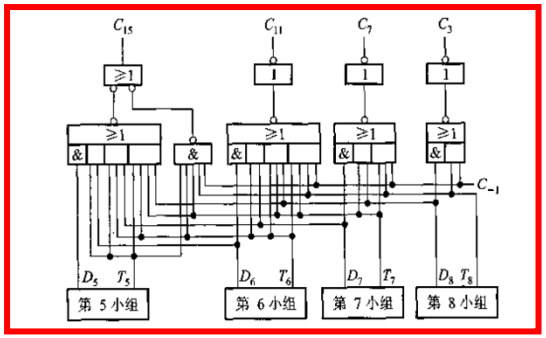


图6-24

双重分组的第2重

本章小结（计算机的运算方法）

- 快速进位链（超前进位）
  - 并行加法器（**16**位，串行进位）：产生全部进位的时间为**16*2ty=32ty**
  - 并行加法器（**16**位，单重分组）：产生全部进位的时间为**4*2.5ty=10ty**

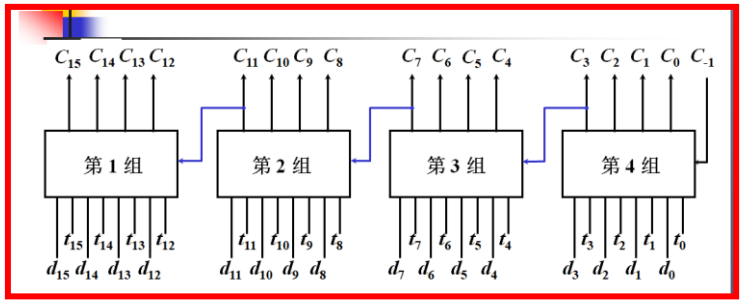


图6-22

  - 并行加法器（**16**位，双重分组）：产生全部进位的时间为**3*2.5ty=7.5ty**

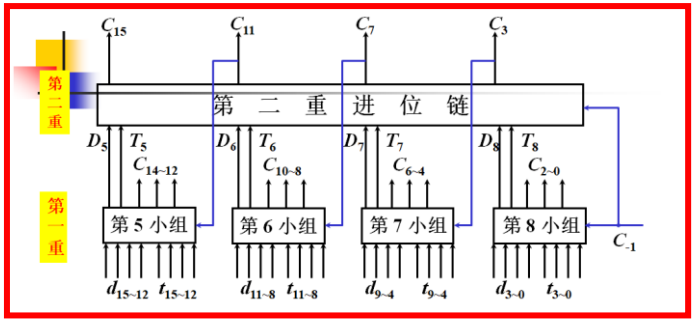


图6-26

本章小结（计算机的运算方法）

- 快速进位链（超前进位）
  - 并行加法器（**32**位，串行进位）：产生全部进位的时间为**32*2ty=64ty**
  - 并行加法器（**32**位，单重分组）：产生全部进位的时间为**8*2.5ty=20ty**
  - 并行加法器（**32**位，双重分组）：产生全部进位的时间为**4*2.5ty=10ty**

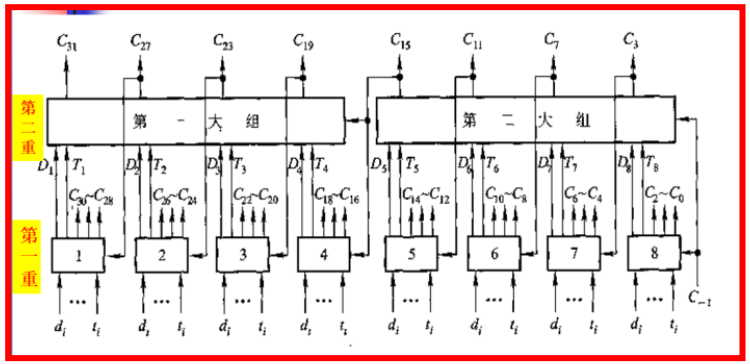


图6-23

本章小结（计算机的运算方法）

- 快速进位链（超前进位）
  - **74LS182**（先行进位部件，超前进位部件）
    - 输入：**74LS181**的快速进位G_i、P_i，以及最低位的进位C_n
    - 输出：送到**74LS181**的C_{n+X}、C_{n+Y}、C_{n+Z}，以及送到上一层**74LS182**的快速进位**P**、**G**

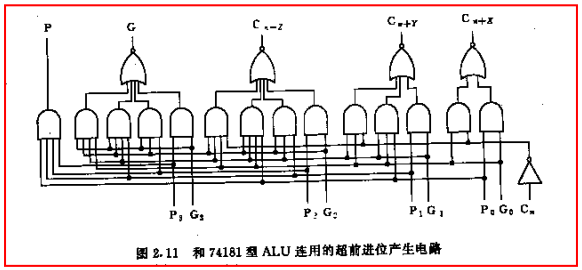


图 2.11　和 74181 型 ALU 连用的超前进位产生电路

  - **8片74LS181**（4位运算器）和**2片74LS182**，构成**32**位并行加法器（超前进位）

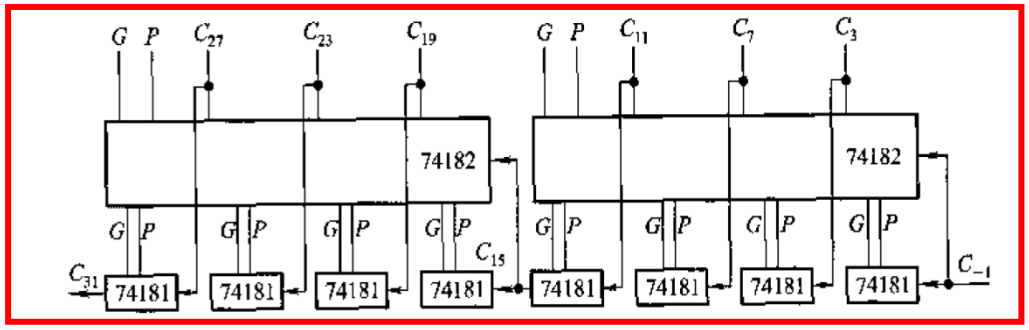
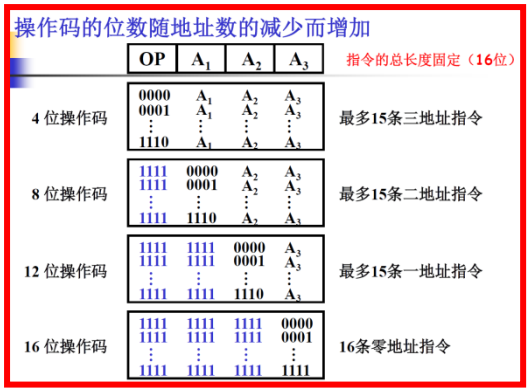


图6-27

第7章　指令系统

本章小结（指令系统）

- 指令=操作码（OP） + 操作数（地址码）
- 扩展操作码技术



- 指令字长（指令翻译成机器码的长度）
  - 指令字长固定（**RISC**计算机）：如**32**位
  - 指令字长可变（**CISC**计算机）：**8**位、**16**位、**24**位……

本章小结（指令系统）

- 数据在存储器中的存放方式
  - 边界对准：(例如，32位的字存放在0000H-0003H存储单元中)
  - 边界未对准：(例如，32位的字存放在0002H-0005H存储单元中)

- 指令的类型（操作类型、操作码的类型）
  - 数据传送指令
  - 算术逻辑指令
  - 移位指令
  - 转移指令
    - 无条件转移
    - 条件转移
    - 子程序调用和返回
    - 陷进指令（例如，INT 21H）
  - 输入输出指令
  - 其他指令

本章小结（指令系统）

- 寻址方式
  - 指令寻址——确定下一条指令的地址
  - 数据寻址——确定操作数的地址
- 指令寻址
  - **PC+1 -> PC　　PC+2->PC　　PC+3->PC**
  - 跳转：转移指令、子程序调用与返回指令、陷进指令、中断

本章小结（指令系统）

- 数据寻址
  1. 立即数寻址： **MOV AX, 2300H**
  2. 直接寻址：**MOV BX, [1000H]**
  3. 隐含寻址：**MUL BL　　AL*BL -> AX**
  4. 间接寻址：**MOV AL, [[2300H]]**
  5. 寄存器寻址：**MOV AX, BX**
  6. 寄存器间接寻址： **MOV CH, [SI]**
  7. 基址寻址：**MOV AX, ARRY[BP]　　MOV AX, ARRY[BX]**
  8. 变址寻址：**MOV AX, 10H[SI]　　MOV TABLE[DI], 12H**
  9. 基址变址寻址：**MOV AX, 200H[BX][SI]**
  10. 相对寻址：**JMP LOOP**　　怎么计算转移指令的位移量？例**7.2**
  11. 堆栈寻址：**PUSH AX　　POP AX**　　例**7.3**（计算**PC**、**SP**、堆栈栈顶的内容）

例7.2：计算相对转移指令的位移量

例7.3：计算子程序调用指令的PC、SP、栈顶内容

本章小结（指令系统）

- 指令系统的兼容性：向上兼容（**80286**的指令系统要兼容**8086**的指令系统）

- **RISC：Reduced Instruction Set Computer，**精简指令系统计算机
- **CISC：Complex Instruction Set Computer，**复杂指令系统计算机

- 计算机执行程序所需要的时间 **P = I * C * T**
  - **I**：高级语言程序编译后在机器上运行的指令数
  - **C**：执行每条机器指令所需的平均周期（需要多少个时钟周期？） **CPI**
  - **T**：每个机器周期的执行时间 （即时钟周期，主频的倒数）

表 7.6 RISC/CISC 的 *I*、*C*、*T* 统计比较

	I	*C*	*T*
RISC	1.2～1.4	1.3～1.7	<1
CISC	1	4～10	1

- **RISC** 计算机的性能优于 **CISC** 计算机 **2-5**倍

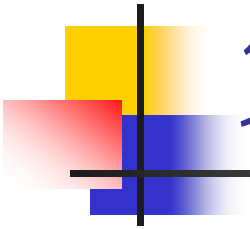
本章小结（指令系统）

- **RISC的主要特征（7个方面）**
  - 选用使用频度较高的一些 简单指令，复杂指令的功能由简单指令来组合实现
  - 指令 长度固定 、指令格式简单、指令种类少 、寻址方式少
  - 只有 **LOAD / STORE**指令访存
  - 采用 流水技术，一个时钟周期 内能完成（执行）一条指令
  - 采用 组合逻辑实现控制器，不用 微程序控制器
  - **CPU** 中有多个通用寄存器
  - 采用优化的编译 程序

- **CISC的主要特征（7个方面）**
  - 系统指令复杂庞大，各种指令使用频度相差大
  - 指令长度不固定 、指令格式复杂、指令种类多 、寻址方式多
  - 访存指令不受限制
  - 大多数指令需要 多个时钟周期 才能执行完毕
  - 采用微程序控制器
  - **CPU** 中设有专用寄存器
  - 难以用优化编译生成高效的目代码

本章小结（指令系统）

- **RISC计算机与CISC计算机的比较（4个优点、1个缺点）**
  - **RISC** 更能充分利用 **VLSI** 芯片的面积
  - **RISC** 更能提高计算机运算速度
    - 指令数 少、指令格式简单 、寻址方式少
    - 通用寄存器多，采用组合逻辑控制，采用寄存器窗口重叠技术
    - 便于实现指令流水
  - **RISC** 便于设计，可降低成本 ，提高可靠性
  - **RISC** 有利于编译程序代码优化
  - **RISC** 不易实现指令系统兼容



第 8 章　CPU 的结构和功能

8.1 CPU 的结构

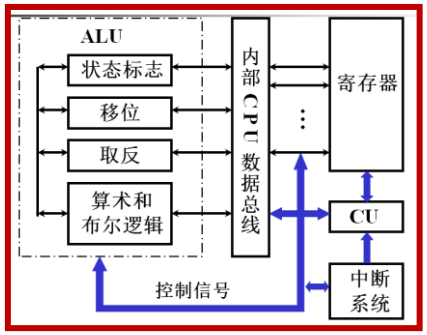
8.2 指令周期

8.3 指令流水

8.4 中断系统

本章小结（CPU的结构和功能）

- **CPU的功能**
  - 控制器的功能
    - 取指令、分析指令、执行指令
  - 运算器的功能
- **CPU的内部结构**



- **CPU的寄存器**
  - 用户可见的寄存器
  - 控制和状态寄存器（用户不可见的寄存器）
- 控制器的设计方法
  - 组合逻辑设计（硬布线控制器）
  - 微程序设计（微程序控制器）

本章小结（CPU的结构和功能）

- 指令周期
  - 取指周期
  - 间址周期
  - 执行周期
  - 中断周期

- 指令周期的数据流
  - 取指周期的数据流：6步
  - 间址周期的数据流：4步
  - 执行周期的数据流：不同指令的执行周期数据流不一样
  - 中断周期的数据流：6步

本章小结（CPU的结构和功能）

- 并行
  - 并发：两个或两个以上事件在 同一时间段 发生
  - 同时：两个或两个以上事件在 同一时刻 发生
- 并行性的等级
  - 过程级（程序、进程）：粗粒度，软件实现
  - 指令级（指令之间、指令内部）：细粒度，硬件实现
- 影响指令流水线性能的因素
  - 结构相关：资源相关
  - 数据相关：
    - 写后读相关（RAW）　　先写后读
    - 读后写相关（WAR）　　先读后写
    - 写后写相关（WAW）　　写后再写
  - 控制相关：转移指令引起的

本章小结（CPU的结构和功能）

- 流水线的性能
  - 吞吐率(Throughput Rate)
  - 加速比 S_p (Speedup Ratio)
  - 效率(Efficient)

例8.1：画指令周期流程、时空图；计算吞吐率、加速比、效率

- 流水线的多发技术
  - 超标量技术（Superscalar）
  - 超流水线技术（Superpiplined）
  - 超长指令字技术（VLIW）

本章小结（CPU的结构和功能）

- **中断系统需解决的问题**（有些是通过硬件解决，有些是通过软件解决）
  - 各中断源如何向CPU提出请求？
  - 各中断源同时提出请求怎么办？（优先级排队）
  - CPU什么条件、什么 时间、以什么 方式响应中断？
  - 如何保护现场？
  - 如何寻找入口地址？（中断服务程序入口地址）
  - 如何恢复现场？如何返回？
  - 处理中断的过程中又 出现新的中断 怎么办？(中断嵌套)

本章小结（CPU的结构和功能）

- 中断请求标记：INTR
- 中断判优逻辑
  - 硬件实现：链式排队器
  - 软件实现

图5.38：在I/O接口中
图8.25：在CPU中

- 中断服务程序入口地址的寻找
  - 硬件向量法
  - 软件查询法

图5.39：在I/O接口中
图8.27：在CPU中

本章小结（CPU的结构和功能）

- 中断响应
  - 响应中断的 条件
  - 响应中断的 时间
  - 中断隐指令(中断周期CPU自动完成的操作)
    - 保护程序断点
    - 寻找中断服务程序入口地址
    - 硬件关中断 (自动关中断)

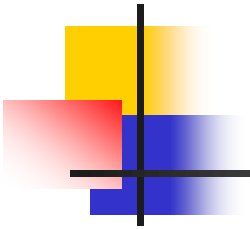
保护到地址为0的存储器中

保护到堆栈中

- 保护现场和恢复现场

- 中断屏蔽技术：多重中断
  - 屏蔽字
  - 多重中断的断点保护

例8.2：画CPU执行程序的轨迹



第9章　控制单元的功能

9.1 微操作命令的分析

9.2 控制单元的功能

本章小结（控制单元的功能）

- 取指周期——从存储器中取出指令的机器码（放到IR中）

① PC → MAR → 地址线

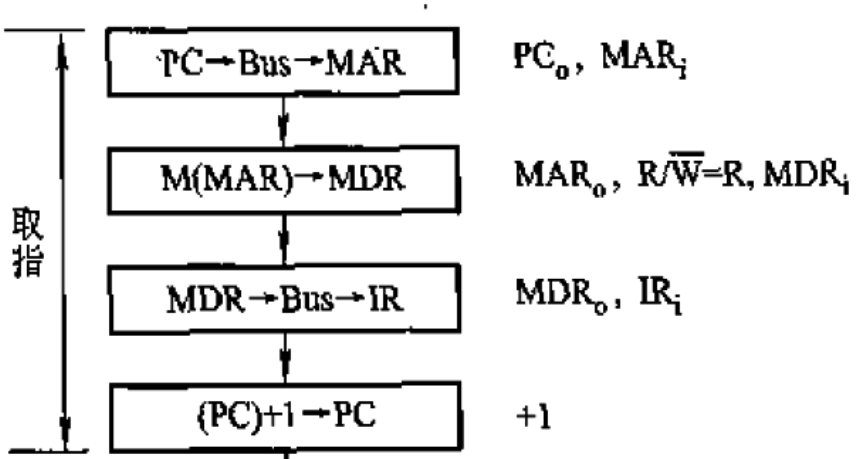
② 1 → R

③ M (MAR) → MDR

④ MDR → IR

⑤ OP（IR）→ CU

⑥ (PC) + 1 → PC



本章小结（控制单元的功能）

- 间址周期——根据形式地址从存储器中得到有效地址

① 指令形式地址 ⟶ MAR

即：Ad (IR) ⟶ MAR

② 1⟶R

③ M (MAR) ⟶MDR

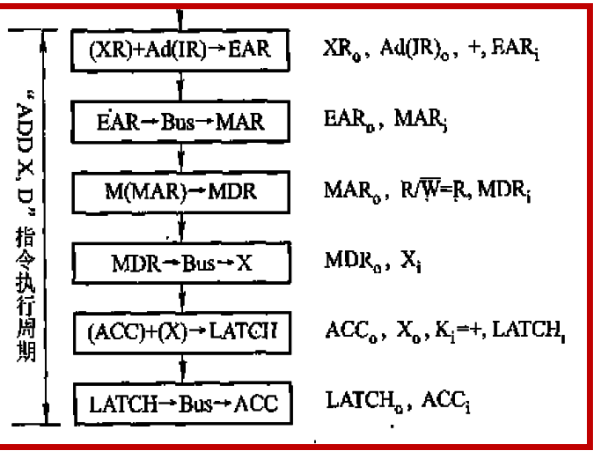
④ MDR⟶Ad (IR)

- Ad(IR)->MAR
- 1->R
- M(MAR)->MDR

本章小结（控制单元的功能）

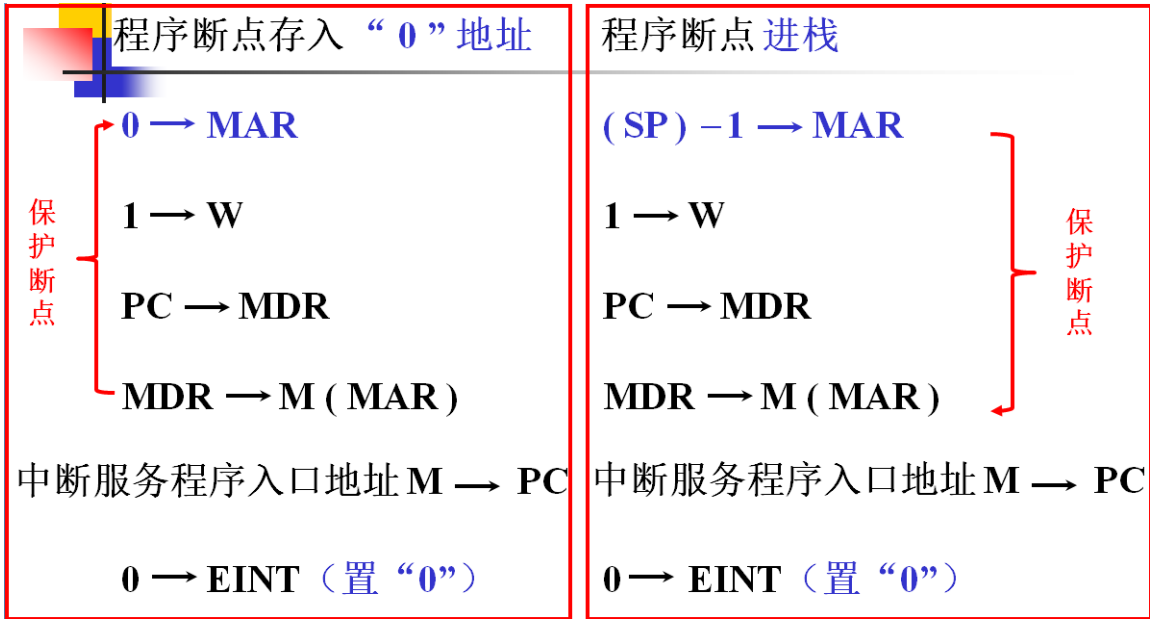
- 执行周期
  - 非访存指令
  - 访存指令
  - 转移指令

- ①取数指令“LDA M”执行阶段所需的全部微操作
  - Ad(IR)->MAR　指令的地址码字段->MAR
  - 1->R　命令存储器读
  - M(MAR)->MDR　操作数从存储器中读至MDR
  - MDR->ACC　操作数->IR
- ②存数指令“STA M”执行阶段所需的全部微操作
  - Ad(IR)->MAR　指令的地址码字段->MAR
  - 1->W　命令存储器写
  - ACC->MDR　欲写入的数据->MDR
  - MDR-> M(MAR)　数据写至存储器中
- ③加法指令“ADD M”执行阶段所需的全部微操作
  - Ad(IR)->MAR　指令的地址码字段->MAR
  - 1->R　命令存储器读
  - M(MAR)-MDR　操作数从存储器中读至MDR
  - (ACC)+(MDR)->ACC　两数相加结果送ACC



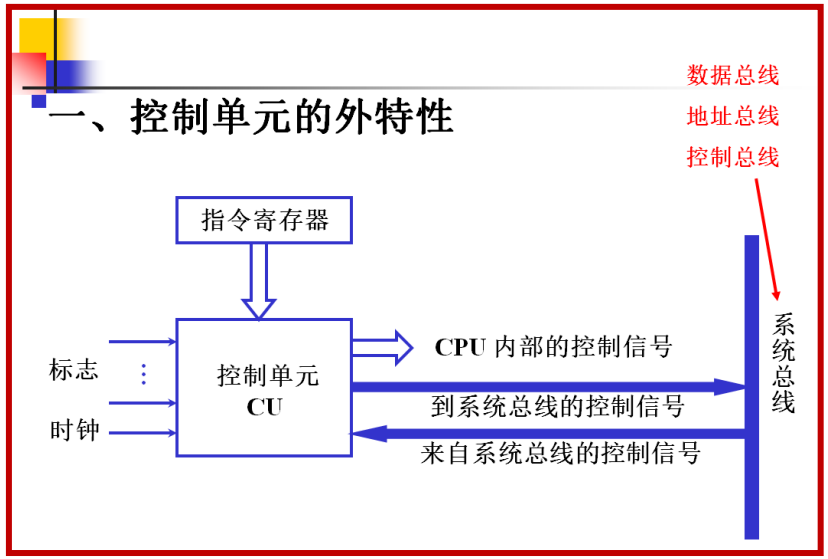
本章小结（控制单元的功能）

- **中断周期**——由中断隐指令自动完成：（1）保护断点；（2）寻找中断服务程序入口地址；（3）硬件关中断



本章小结（控制单元的功能）

- 控制单元的输入和输出
  - 输入：
    - 指令的操作码
    - 标志
    - 来自系统总线的控制信号
    - 时钟信号
  - 输出：
    - CPU内容的控制信号
    - 到系统总线的控制信号

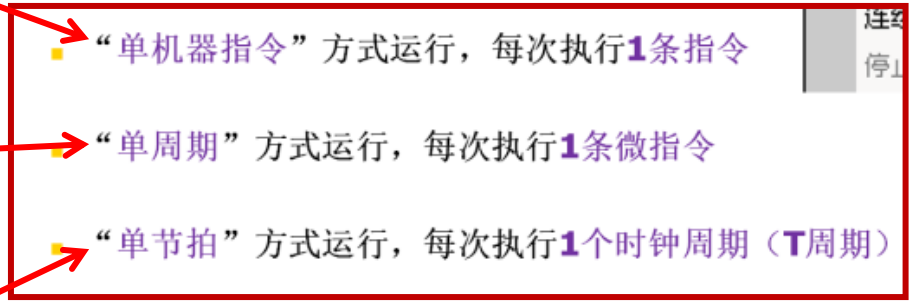


本章小结（控制单元的功能）

- 控制信号举例
  - 不采用CPU内部总线的方式（图9.3）
  - 采用CPU内部总线的方式（图9.4）
  - 例9.1（不采用CPU内部总线的方式）
    - 写出取指周期的全部微操作
    - 写出“LDA M”、“STA M”、“ADD M”指令执行周期的全部微操作
    - 写出间址周期的全部微操作
    - 写出“JMP Y”、“BAZ Y”指令执行周期的全部微操作
  - 例9.2（采用单总线计算机结构）
    - 画出“ADD X, D”、“STA *D”指令的指令周期信息流程图，并列出相应的控制信号序列

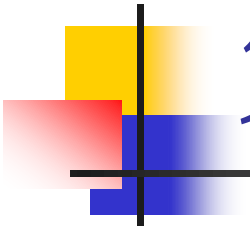
本章小结（控制单元的功能）

- 指令周期
- 机器周期
- 时钟周期



本章小结（控制单元的功能）

- 控制方式
  - 同步控制方式
    - 采用定长的机器周期
    - 采用不定长的机器周期
    - 采用中央控制和局部控制相结合的方法
  - 异步控制方式
  - 联合控制方式：同步与异步的结合
  - 人工控制方式：**RESET**、单步执行、连续执行



第10章　控制单元的设计

10.1　组合逻辑设计

10.2　微程序设计

本章小结（控制单元的设计）

- 组合逻辑设计（硬布线控制器）
  - 安排微操作时序的原则

原则一　微操作的先后顺序不得随意更改

原则二　被控对象不同的微操作
尽量安排在一个节拍内完成　　节省时间

原则三　占用时间较短的微操作
尽量安排在一个节拍内完成
并允许有先后顺序

  - 微操作命令及节拍安排

取指周期
- T0 PC->Bus->MAR, 1->R
- T1 M(MAR)->MDR, (PC)+1->PC
- T3 MDR->Bus->IR, OP(IR)->微操作命令形成部件(CU)

间址周期 微操作的 节拍安排

T_0　Ad(IR) ⟶ MAR
　　1 ⟶ R
T_1　M(MAR) ⟶ MDR
T_2　MDR ⟶ Ad(IR)

执行周期

ADD AL,80H

(2)立即寻址的加法指令执行周期的微操作命令及节拍安排如下：
- T0 Ad(IR)->Bus->R1　　;立即数送R1
- T1 (ACC)+(R1)->ALU->R2
- T2 R2->Bus->ACC

5. 中断周期 微操作的 节拍安排

T_0　0 ⟶ MAR　1 ⟶ W　　硬件关中断
T_1　PC ⟶ MDR
T_2　MDR ⟶ M(MAR)　　向量地址 ⟶ PC

本章小结（控制单元的设计）

- 组合逻辑设计（硬布线控制器）
  - 例10.1：
    - 写出取指周期全部微操作命令及节拍安排
    - 写出“ADD #a”指令在执行阶段所需的全部微操作及节拍安排
  - 例10.2：
    - 写出“ADD B, C”、“SUB E, @H”、“STA @mem”指令组合逻辑控制单元所发出的微操作命令及节拍安排
  - 例10.3：
    - 写出采用Booth算法实现“MUL a”指令取指周期全部微操作命令及节拍安排

本章小结（控制单元的设计）

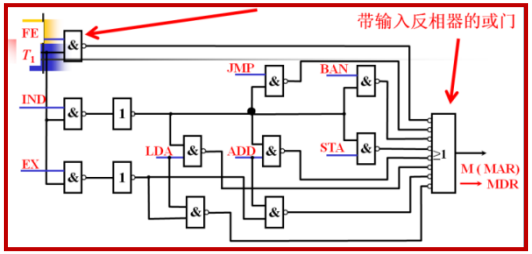
- 组合逻辑设计（硬布线控制器）
  - 组合逻辑设计步骤
    - 1. 列出每个微操作命令的操作时间表
    - 2. 写出每个微操作命令的最简表达式
    - 3. 画出逻辑图

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
FE 取指	T_0		PC⟶MAR	1	1	1	1	1	1
			1⟶R	1	1	1	1	1	1
	T_1		M(MAR)⟶MDR	1	1	1	1	1	1
			(PC)+1⟶PC	1	1	1	1	1	1
	T_2		MDR⟶IR	1	1	1	1	1	1
			OP(IR)⟶ID	1	1	1	1	1	1
		I	1⟶IND			1	1	1	1
		$\bar{I}$	1⟶EX	1	1	1	1	1	1

以 M(MAR)⟶MDR 微操作命令 为例

$$= FE \cdot T_1 + IND \cdot T_1 (ADD + STA + LDA + JMP + BAN) + EX \cdot T_1 (ADD + LDA)$$

$$= T_1\{FE + IND (ADD + STA + LDA + JMP + BAN) + EX (ADD + LDA)\}$$



本章小结（控制单元的设计）

- 微程序设计
  - 一条机器指令对应一段微程序（若干条微指令），微程序存储在控制存储器（控存）中
  - 微指令 = 操作控制字段 + 顺序控制字段（下地址）
  - 微指令的编码方式（控制方式）：主要是前面2种方式
    1. 直接编码（直接控制）方式
    2. 字段直接编码方式：实验箱模型机采用的方式
    3. 字段间接编码方式
    4. 混合编码
    5. 其他

本章小结（控制单元的设计）

- 微程序设计
  - 微指令序列地址的形成：主要是前面2种方式
    1. 微指令的 下地址字段 指出
    2. 根据机器指令的 操作码 形成
    3. 增量计数器：(CMAR) + 1 -> CMAR
    4. 分支转移
    5. 通过测试网络
    6. 由硬件产生微程序入口地址

本章小结（控制单元的设计）

- 微程序设计
  - 水平型微指令
  - 垂直型微指令

微操作码	地址码	其他

  - 静态微程序设计
  - 动态微程序设计
  - 毫微程序设计：采用两级微程序设计方法，第一级为垂直型微指令，第二级为水平型微指令
  - 串行微程序控制
  - 并行微程序控制

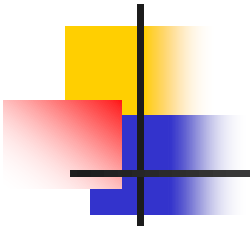
本章小结（控制单元的设计）

- 微程序设计举例（微程序设计步骤）：
  - 写出机器指令对应的全部微操作及节拍安排（例**10.1**、**10.2**、**10.3**；例**9.1**、**9.2**）
  - 确定微指令格式：采用什么编码方式（直接控制？字段直接编码）？微命令字段几位？微地址字段几位？
  - 编写每条微指令的二进制代码（微指令码点，微指令代码化）

关于期末考试

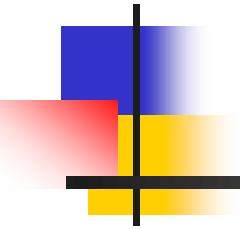
- 考试时间：**2021年6月21日14:00-16:00**
- 考试地点：**1号楼（学武楼）A201**

- 题型：
  - 填空题（**25空，25分**）
  - 判断题（**10小题，10分**）
  - 问答题（**4小题，30分**）
  - 综合题（**3小题，35分**）



祝同学们期末考试

取得好成绩！



The End

Thanks