1. 概论要点

1. 存储程序

- 定义
    1. 计算机的组成成分 :
        - 运算器
        - 控制器
        - 存储器
        - 输入设备
        - 输出设备
    2. 计算机内部使用二进制来表示指令和数据
    3. 将编写好的程序和原始数据预先存储在存储器中，然后启动计算机工作
- 弱点

  存储器的访问会成为瓶颈

2. CPU

1. 组成

   CPU = 运算器 + 控制器
2. 运算器

   核心是加法器，内部存在若干通用寄存器和累加寄存器(暂存运算结果)
    - 算术运算
    - 逻辑运算
3. 控制器

   按照预先的确定的操作步骤控制计算机的运行
    - 控制器从主存中取指分析
    - 包含有一些专用寄存器

3. 主机

1. 组成

   主机 = CPU + 主存

4. 总线

1. 定义 : 总线是一组能为多个部件服务的公共信息传送线路，它能分时地发送与接收各部件的信息
2. 特点
    - 分时
    - 共享
3. 单总线结构

   将各个部件连接在单一的一组总线上，称单总线是系统总线
4. 分类
    - 地址总线
        - 单方向
        - 多根
        - CPU向主存和外设传递地址信息
    - 控制总线

        - 双向
        - CPU送出的控制命令
        - 主存/外设反馈给CPU的状态信号
    - 数据总线
        - 双方向
        - 多根
        - CPU可以沿这些线从主存或外设读入数据，也可以沿这些线向主存或外设送出数据

5. 计算机系统

1. 成分
    - 计算机系统 = 硬件系统 + 软件系统
    - 软件和硬件在逻辑上是等价的，是相辅相成不可分割的整体
2. 兼容
    - 尽力做到向后兼容和向上兼容
3. 计算机系统的多层次结构
    0. 硬连逻辑级 : 计算机的内核
    1. 微程序级 : 机器语言的微指令级
    2. 机器语言级 : 机器的指令集
    3. 操作系统级
    4. 汇编语言级
    5. 高级语言级
    6. 应用语言级

    实际机器 : 0 ~ 2

    虚拟机器 : 3 ~ 6

2. 数据的机器层次表示

1. 无符号数 & 有符号数

1. 无符号数 :
    - 无符号数就是整个及其字长的全部二进制位均表示数值位(没有符号位的概念)
    - 相当于的数的绝对值
2. 带符号数
    - 机器字长的最高位(0, 1)表示符号位
3. 真值和机器数
    1. 机器数 : 0, 1表示数的正负性的二进制真值表示
    2. 真值 : 正号负号和数的绝对值表示

2. 原码，反码，补码

1. 原码 : 真值的表示方式
2. 反码 : 符号位不变，数值位取反
3. 补码 :
    - 整体取反 + 1
    - 从右向左数的第一个1之后的所有的数值位取反
4. 0的表示方法
    1. 原码
        - +0 : 00000
        - -0 : 10000

2. 反码
    - +0 : 00000
    - −0 : 11111
3. 补码

    +0 = −0 : 00000

5. 三种机器数的比较
    - 正数都等于数值的真值本身，负数表示方法都不同
    - 最高位都是符号位，但是补码和反码的符号位可以参与计算，但是原码的符号为必须分隔考虑
    - 对0的表示方法不同
    - 数据范围
        - 原码，反码表示正负数范围对称
        - 补码表示的正负数范围不对称，负数多表示一个最小负数

3. 定点小数的表示范围

小数点位数是n位

1. 原码

$$-(1-2^{-n}) \rightarrow 1-2^{-n}$$

2. 补码

$$-1 \rightarrow 1-2^{-n}$$

4. 定点整数的表示范围

数值位n位

1. 原码

$$-(2^n-1) \rightarrow 2^n-1$$

2. 补码

$$-2^n \rightarrow 2^n-1$$

5. 浮点数

浮点数表示范围

- 限定
    1. 阶码部分 k 位
    2. 尾数部分 n 位
    3. 阶码和位数部分都是用补码表示
- 浮点数的表示范围

$$-1 \times 2^{2^k-1} \rightarrow (1-2^{-n}) \times 2^{2^k-1}$$

规格化浮点数

1. 限定了尾数的最高位必须是一个有效的数据位
2. 限定
    - 原码 : 尾数的最高数位必须是1
    - 补码 : 尾数的最高数位必须和符号位不同

3. 规格化最小正数 : $2^{-1} \times 2^{-2^k}$

6. 移码

1. 移码将所有的真值都映射到了正数域，可以讲移码视为是无符号的数
2. 移码和补码只有符号位不同

7. 非数值

1. 国标码 = 区位码（十六进制）+ 2020H
2. 汉字机内码 = 汉字国标码 + 8080H

3. 指令系统

1. 非规整型指令的操作码（扩展操作码）

- 操作码字段的位数不固定，且分散地放在指令字的不同位置上
- 定长指令字可以表示不同的操作长度的操作码，但是控制器的设计变得更加的复杂
- 实现方法 : 扩展操作码法
    1. 让操作数地址个数多的指令（如三地址指令）的操作码字段短些
    2. 操作数地址个数少的指令（如一或零地址指令）的操作码字段长些
    3. 注意事项 :
        - 不允许短码是长码的前缀，否则无法保证解码的唯一性和实时性
        - 指令的操作码一定不可以重复，保证指令的格式安排的统一和规整

2. 编址方式

1. 编址
    - 通用寄存器
    - 主存
    - 输入输出设备
2. 编制单位
    - 字编址
        - 编址单位=访问单位
        - 每一个编制单位的包含的信息量和访问一次寄存器和主存获得的信息量是一致的
    - 字节编址
        - 编址单位＜访问单位
        - 为了适应非数值计算的需要
        - 通常访问的单位是编制单位的几倍
    - 地址码的位数
        - 和最小寻址单位是有关联的
        - 如果以字节为最小寻址单位，地址码的位数就需要长些
        - 如果以字为最小寻址单位(机器字长)，地址码的位数可以减少

3. 数据寻址和指令寻址

1. 速度快慢和访存次数
    1. 立即寻址 : 0
    2. 寄存器寻址 : 0
    3. 直接寻址 : 1
    4. 寄存器间接寻址 : 1, 寄存器取数
    5. 页面寻址 : 1, 拼接操作
    6. 变址寻址（基址寻址、相对寻址）: 1, 累加计算地址
    7. 一级间接寻址多级间接寻址 : ≥ 2

4. 存储器堆栈操作

1. 操作顺序
    1. 先修改指针
    2. 再压入或者弹出数据

5. 程序控制类指令

1. 转移指令
    - 无条件转移指令
    - 条件转移指令
    - 相对寻址

      虽然PC计数器会自动的累增地址，但是我们的相对的地址的值不改变
    - 绝对寻址
    - 有符号和无符号的大小比较
        - 有符号

          G / E / L
        - 无符号

          A / E / B
2. 子程序调用指令
    - 子程序是一段可以公用的指令序列，只要知道子程序的入口地址即可调用
    - 转子指令 : `CALL` 是一个地址指令
    - 子程序调用指令和转移指令的区别
        - 返回调用程序
        - 转移时同一程序内转移，但是子程序调用实在不同的程序之间的转移
    - 返回地址的保存方法
        - 子程序的第一个字节用来保存返回地址
            1. 间接转移到调用程序
            2. 可以实现嵌套但是不可以实现递归
        - 寄存器存放返回地址
            1. 转子指令将返回地址放到一个寄存器中，子程序再将寄存器中的内容保护起来(转移到主存中)
            2. 但是增加了子程序的复杂程度
        - 堆栈存放返回地址
            1. 先进先出的计算模式符合我们的子程序调用的顺序
            2. 可以实现子程序嵌套和子程序的递归调用
            3. 不会增加子程序的复杂程度，使用的最广泛的方式
    - 返回指令
        - 子程序的最后一条指令是返回指令
        - 返回指令存放的位置决定了返回指令的格式

          零地址的返回执行 → 堆栈存放格式

6. 输入输出类指令

1. 目的

   输入/输出（I/O）类指令用来实现主机与外部设备之间的信息交换
2. 编址方式

- 独立编址方式
  - 外设端口和主存单元独立编址
  - 必须使用特定的指令 IN/OUT
- 统一编址方式
  - 外设寄存器和准村单元统一编址
  - 没有特定的指令，统一使用 MOV 实现数据的转移
  - 但是外设如果过大会挤占内存编址空间

7. 指令系统

1. CISC / RISC
   - CISC : 复杂指令系统计算机
   - RISC : 精简指令系统计算机
2. VLIW / EPIC
   - VLIW : 指令集并行
   - EPIC : 多条指令混入一个指令字中，提高CPU计算功能部件的利用效率和性能

5. 存储系统和结构

1. 主存储器的基本结构

组成

读写操作都是在控制器的控制下执行的

- 存储体 : 主存储器的核心，存放程序和数据
- 地址译码驱动电路 :
  - 译码器 : 地址码转换成对应的有效输出线的有效电平
  - 驱动器 : 驱动相应的读写电路
- I/O读写电路 : 完成被选中的存储单元的读写操作

2. 主存储器的存储单元

1. 位 : 二进制数的基本单位，存储器存储信息的最小单位
2. 存储字 : 一个二进制数由若干位组成，二进制数被作为整体存入写出的时候称之为是存储字
3. 存储单元 : CPU可以主存访问的最小存储单位，字或者双字都是由计算机的结构决定的
4. 存储体 : 大量存储单元的集合构成一个存储体,存储器的核心

3. 主存储器的主要技术指标

1. 存取时间 T_a : 从启动一次存储器操作到完成该操作所经历的时间
2. 存取周期 T_m : 存储器进行一次完整的读写操作所需的全部时间
3. 关系

   $$T_m > T_a$$

   - 任何一种存储器，读写操作之后都需要一定的时间恢复内部状态
   - 如果是破坏性的读出的话 : $T_m = 2T_a$

4. 数据在主存中的存放

1. 不浪费存储资源
   - 不浪费主存储器的资源
   - 访问的时候可能会需要两个存储周期，读写电路比较复杂
2. 起始位置存放

- 浪费大量的主存储器资源
- 只需要一个读写周期

3. 边界对齐
    - 浪费少量的主存储器资源
    - 主需要一个读写周期

5. RAM记忆单元电路

1. 主存储器通常分为RAM和ROM两大部分,RAM可读可写，ROM只能读不能写
2. 存放一个二进制位的物理器件称为记忆单元，它是存储器的最基本构件
3. 分类
    - SRAM : 静态RAM，其存储电路以双稳态触发器为基础
    - DRAM : 动态RAM，其存储电路以栅极电容为基础
4. DRAM的刷新方式
    - 最大刷新间隔 :
        1. 为了维持MOS型动态记忆单元的存储信息，每隔一定时间必须对存储体中的所有记忆单元的栅极电容补充电荷，这个过程就是刷新
        2. 最大刷新间隔为2ms，应在2ms内，将全部存储体刷新一遍
    - 刷新不同于重写
        - 刷新定时，读写随机
        - 刷新行为单位，读写是一存储单元为单位
    - 刷新周期 = 读写周期
    - 刷新时间 = 存储矩阵行数 × 刷新周期
    - 刷新分类

      P.S : m 是存储矩阵的行数，t 是读写周期(刷新周期)

        1. 集中刷新方式
            - 读写不受刷新的影响
            - 但是存在大量的死区(存储容量越大，死区越大，死区中暂停读写操作)
            - 死区长度 : $m \times t$
        2. 分散刷新方式
            - 系统存储周期 = 读写周期 + 刷新周期
            - 不存在死区，但是系统的存储周期变长，降低机器的速度
            - 刷新过于频繁，没有充分利用最大刷新间隔
        3. 异步刷新方式
            - 相邻两行的刷新间隔 $\Delta t = \frac{t}{m}$
            - Δt 中只有一次刷新周期(死区)，其他都是读写周期
            - 死区变短，减少刷新次数

6. ROM类型

1. 掩膜式ROM : 它的内容是由半导体生产厂家写入后任何人都无法改变其内容
2. 一次编程ROM : PROM允许用户利用专门的设备写入自己的程序
3. 可擦除ROM
    - 紫外线擦除 : 用紫外线灯进行擦除的，所以只能对整个芯片擦除，而不能对芯片中个别需要改写的存储单元单独擦除和重写
    - 电擦除 : 可以擦除部分的存储单元
4. 闪存 : 既可在不加电的情况下长期保存信息，又能在线进行快速擦除与重写

7. 主存容量的扩展

1. 位扩展 : 共用CS / WE数据线
2. 字扩展 : 加入地址译码器，一次只能选中一个芯片
3. 字位同时扩展 :

8. 存储器地址分配和片选

1. 访问存储单位的流程 :
    1. 片选 : 选定存储芯片
    2. 字选 : 从选定的片中选择出相应的存储单元
2. 片选信号的产生
    - 线选法 : 不需要地址译码器，适合连接存储器少的情况
    - 全译码法 : 不会出现地址重叠，需要译码器
    - 部分译码法 : 只有部分高位地址(片内寻址外)的地址译码，会出现地址重叠

9. 主存储器和CPU连接

1. 总线
    - 地址总线
    - 数据总线
    - 控制总线
2. 重要寄存器
    - MAR : 存储器地址寄存器，可以接受来自程序计数器的指令地址或来自运算器的操作数地址，以确定要访问的单元
    - MDR : 存储器数据寄存器，向主存写入数据或从主存读出数据的缓冲部件

6. CPU 中央控制器

1. 控制器的功能

1. 控制器是计算机系统的指挥中心，将运算器，存储器，输入输出设备等部件组成一个有机的整体，然后根据指令的要求指挥全机的工作
2. 在控制器的控制下逐条执行程序中各指令
    - 指令流 : CPU执行的指令序列
    - 数据流 : 根据指令操作要求依次存取数据的序列
    - 关系 : 数据流是由指令流驱动的
3. 控制器的基本功能 : 控制器的基本功能是对指令流和数据流在时间与空间上实施正确的控制

2. CPU中的主要寄存器

- 通用寄存器 :
    1. 通用寄存器可用来存放原始数据和运算结果，有的还可以作为变址寄存器、计数器、地址指针等
    2. 通用寄存器一般可以由CPU直接访问
- 专用寄存器 :
    1. 专用寄存器是专门用来完成某一种特殊功能的寄存器
    2. 组成
        - 程序计数器PC : 程序计数器用来存放正在执行的指令地址或接着要执行的下条指令地址
            - 增量式 + 1
            - 跳转式
        - 指令寄存器IR : 指令寄存器用来存放从存储器中取出的指令
        - 存储器地址寄存器MAR : 存储器地址寄存器用来保存当前CPU所访问的主存单元的地址
        - 存储器数据寄存器MDR : 存储器数据寄存器用来暂时存放由主存储器读出的一条指令或一个数据字

        - 状态标志寄存器PSWR : 状态标志寄存器用来存放程序状态字的。程序状态字的各位表征程序和机器运行的状态

3. 控制器的组成

1. 指令部件
    - 指令部件的主要任务是完成取指令并分析指令
    - 组成
        1. 程序计数器
        2. 指令寄存器
        3. 指令译码器 :
            - 输入 : 指令寄存器的指令
            - 输出 : 产生相应的控制信号提供给微操作信号发生器
        4. 地址形成部件 : 根据指令的不同寻址方式，用来形成操作数的有效地址，一般使用运算器来进行计算
2. 时序部件
    - 时序部件能产生一定的时序信号，以保证机器的各功能部件有节奏地进行信息传送、加工及信息存储
    - 组成
        1. 脉冲源 : 为整个机器提供基准信号
        2. 启停控制逻辑 : 启停控制逻辑保证启动时输出的第一个脉冲和停止时输出的最后一个脉冲都是完整的脉冲
        3. 节拍信号发生器
3. 微操作信号发生器
4. 中断控制逻辑

4. 控制器的硬件实现方法

1. 组合逻辑型
    - 它是采用组合逻辑技术来实现的
    - 速度快
    - 设计调试困难，不可增加新功能
2. 存储逻辑型
    - 使用 11 条展示的微程序控制器实现
    - 它是采用存储逻辑来实现的,使每条机器指令转化成为一段微程序并存入一个专门的存储器（控制存储器）中
    - 利于设计和调整和自动化设计
    - 增加了一级控制存储器，速度慢
3. 结合型

5. 时序系统

1. 指令周期 :

   指令周期是取指指令、分析指令到执行完该指令所需的全部时间

    - 指令周期 $= i \times$ 机器周期
    - 每一个机器周期都已有个对应的周期状态触发器，任何时刻只有一个状态触发器启动
2. 机器周期 :

   机器周期通常又称CPU周期，通常把一条指令划分为若干个机器周期，每个机器周期完成一个基本操作

6. CPU的控制方式

1. 同步控制方式

- 同步控制方式即固定时序控制方式，各项操作都由统一的时序信号控制
- 同步控制方式应以最复杂指令的操作时间作为统一的时间间隔标准
- 对于简单指令来说存在时间浪费，影响指令的执行速度

2. 异步控制方式
    - 异步控制方式即可变时序控制方式。
    - 需要多少时间，就占用多少时间。
    - 没有时间上的浪费，因而提高了机器的效率
    - 但是控制比较复杂。
3. 联合控制方式
    - 在功能部件内部采用同步方式或以同步方式为主的控制方式
    - 在功能部件之间采用异步方式

7. 指令执行的基本过程

1. 流程
    1. 取指令阶段 : 取指令阶段完成的任务是将现行指令从主存中取出来并送至指令寄存器,公共操作

$$
\begin{gathered}
(PC) \rightarrow MAR \\
READ \\
M(MAR) \rightarrow MDR \\
(MDR) \rightarrow IR \\
(PC)+1 \rightarrow PC
\end{gathered}
$$

    2. 分析取数阶段
        1. 指令译码器ID可识别和区分不同的指令类型及各种获取操作数的方法
        2. 指令不同，该阶段各不相同
    3. 执行阶段
        1. 执行阶段完成指令规定的各种操作，形成稳定的运算结果，并将其存储起来

8. 指令的微操作序列

1. 控制器在实现一条指令的功能时，总要把每条指令分解成为一系列时间上先后有序的最基本、最简单的微操作，即微操作序列。
2. 微操作序列是与CPU的内部数据通路密切相关的，不同的数据通路就有不同的微操作序列
3. 都需要取指公共操作

9. 微程序控制的基本概念

程序 → 指令周期 → 微程序(机器周期) → 微指令 → 微命令(微操作)

10. 微指令编码法

1. 正如上式显示
    - 控制存储器中存放的是微指令序列，每一个微指令序列都是由对微命令的编码构成的
2. 微指令的编码方法
    1. 直接控制法
        - 并行性强
        - 微指令字太长
    2. 最短编码法
        - 并行性最差
        - 微指令字太短
    3. 字段编码法
        - 段内最短编码法 : 互斥的微命令

        - 段间直接控制法：兼容的微命令
        - 分类
            - 字段直接编码
            - 字段间接编码

11. 微程序控制器的组成和工作过程

1. 组成
    - 控制存储器（CM）：存放微程序
    - 微指令寄存器（μIR）：用来存放从CM取出的正在执行的微指令
    - 微地址形成部件：用来产生初始微地址和后继微地址
    - 微地址寄存器（μMAR）：它接受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
2. 工作过程
    1. 取指令公共操作：
        - 取指令的公共操作通常由一段取指微程序来完成
        - 取指微程序：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IR中了
        - 工作过程
            1. 自动将0地址(取指微程序入口地址)加入μMAR中执行取指微程序
    2. 由机器指令的操作码字段通过微地址形成部件产生出该机器指令所对应的微程序的入口地址，并送入μMAR
    3. 从CM中逐条取出对应的微指令并执行之
    4. 完成一个微程序(机器周期)返回取指微指令首地址准备下一次的机器周期的执行

12. 微程序入口地址的形成

1. 由机器指令的操作码字段指出各段微程序的入口地址（初始微地址）
2. 分类
    - 一级功能转换：如果机器指令操作码字段的位数和位置固定，可以直接使操作码与入口地址码的部分位相对应
    - 二级功能转换：不固定的时候
        1. 第一次先按指令类型标志转移，以区分出指令属于哪一类，如：是单操作数指令，还是双操作数指令等
        2. 因为每一类机器指令中操作码字段的位数和位置是固定的，可以直接找到
    - PLA电路实现：
        1. 可以采用PLA电路将每条机器指令的操作码翻译成对应的微程序入口地址
        2. 转换速度快

13. 后继微地址的形成

1. 增量方式
    - 累计
    - 可以使用μMAR代替微程序计数器技术
    - 要求一段微程序必须存放在连续的存储单元中
2. 断定方式
    - 可以由微程序设计者指定，比较灵活
    - 非测试段：可由设计者指定，一般是微地址的高位部分，用来指定后继微地址在CM中的某个区域内
    - 测试段：

这相当于在指定区域内断定具体的分支,测试段如果只有一位，则微地址将产生两个分支，若有两位，则最多可产生四个分支，依此类推，测试段为n位最多可产生2n个分支

7. 总线

1. 总线的性能指标

1. 总线宽度 : 总线的线数，决定了总线所占的物理空间和成本
2. 总线带宽 :
   - $B = \frac{W}{T \times N}$
   - 总线的最大数据传输率 : 总线每秒传输的字节数
3. 总线负载 : 连接在总线上的最大设备的数量
4. 总线复用 :
   - 总线分时复用，在不用的时间段上利用总线上的同一个信号线传递不同的信号
   - 减少总线的数量，提高总线的利用率
5. 总线的猝发传送 : 一种总线的传输方式，在一个总线周期内传输存储地址连续的多个数据

8. 外部设备

1. 磁表面存储原理

1. 磁记录介质 : 信息记录在一层薄层磁性材料上的(磁层),磁层和所依附着的载体称之为记录介质
2. 磁头 : 磁记录设备的关键部件之一，是一种电磁转换设备
   - 写 : 电到磁
   - 读 : 磁到电

   分类
   - 接触式磁头
   - 浮动式磁头

   工作方式 : 记录截止运动，磁头不动
3. 写入 : 写磁头线圈上通上正负两个不同方向的写电流可以表示1,0两种不同的剩磁状态
4. 读出 : 磁头在运动的时候产生对应的感应电动势可以生成读出信号

2. 磁表面存储器的技术指标

1. 记录密度
   - 记录密度是指磁介质存储器单位长度或单位面积磁层表面所能存储的二进制信息量
   - 通常以道密度和位密度表示，也可用两者的乘积面密度来表示
2. 存储容量
   - 存储容量是指整个磁介质存储器所能存储的二进制信息的总量，一般用位或字节为单位表示
   - 它与存储介质尺寸和记录密度直接相关
3. 平均存取时间

   平均存取时间 = 寻道时间 + 旋转延迟时间 + 读写时间 + 磁盘控制器的开销
4. 数据传送率

   磁介质存储器在单位时间内向主机传送数据的位数或字节数

3. 数字磁记录方式

1. 直接记录方式
   - 归零制 : 每次都会恢复到电平上
   - 不归零制 : 见变就翻

    - 不归零制-1：见1就翻
2. 按位编码记录方式
    - PE：1必上升，0必下降
    - FM：1中间两边翻，0只在边界翻
    - MFM
    - MMFM

4. 硬盘的信息分布 and 磁盘地址

硬盘信息分布

1. 驱动器号
2. 柱面
3. 磁头号
4. 扇区号

9. 输入输出系统

1. 输入/输出接口

1. 是主机和外设的交接界面，通过接口实现主机和外设的信息交换
2. 传递的信息
    - 数据信息：双向的
    - 控制信息：这是CPU对外设的控制信息或管理命令,单向的
    - 状态信息：表示外设的工作状态
    - 联络信息：配合工作(按顺序执行操作)
    - 外设识别信息：帮助CPU识别出需要使用的外设的信息

2. 接口的基本组成

1. 接口中要分别传送数据信息、控制信息和状态信息，数据信息、控制信息和状态信息都通过数据总线来传送。大多数计算机都把I/O设备的状态信息视为输入数据，而把控制信息看成输出数据，并在接口中分设各自相应的寄存器，赋以不同的端口地址，各种信息分时地使用数据总线传送到各自的寄存器中
2. 端口是指接口电路中可以进行读/写的寄存器，若干个端口加上相应的控制逻辑电路才组成接口

3. I/O编址方式

1. 独立编址
    - 需要额外的输入输出指令
    - 不需要占用主存空间
2. 统一编址
    - 占用主存空间
    - 不需要额外的指令
3. 外设

    外设至少需要两种寄存器
    - 控制状态寄存器
    - 数据缓冲寄存器

4. 中断的基本概念

- 允许主机和外设置键的并行工作，提高了CPU的使用效率
- 适合中低速的设备

5. 程序中断与调用子程序指令的区别

1. 随机性：子程序的执行是程序员事先安排好的，但是中断的到来时随机的
2. 主从无关性：子程序的执行和主程序是之间有关系的，但是中断和当前的程序执行之间不存在任何的关系
3. 并发同时性：不存在同事调用子程序的情况，但是中断的到来可以是并发的

6. 中断的基本类型

1. 自愿中断和强迫中断 :
    - 自愿中断 : 程序的自中断，程序中安排的相关的中断
    - 强迫中断 : 随机产生的中断，和程序的执行没有任何的关系
2. 程序中断和简单中断
    - 程序中断 : 执行中断处理程序的中断请求
    - 简单中断 : DMA传送方式的前称，不执行相应的中断处理程序的，不破坏线性程序的执行
3. 内中断和外中断
    - 内中断 : CPU内部引发的中断
    - 外中断 : CPU外部的设备引发的中断
        - 屏蔽中断
        - 不可屏蔽中断
4. 向量中断和非向量中断
    - 向量中断 : 中断服务程序的入口地址是中断提供的中断
    - 非向量中断 : 不直接的提供中断成立程序的入口地址，需要CPU直接的是被找到入口地址
5. 单重中断和多重中断
    - 单 : 中断处理程序不可以被打扰
    - 中断可以嵌套

7. CPU响应中断的条件

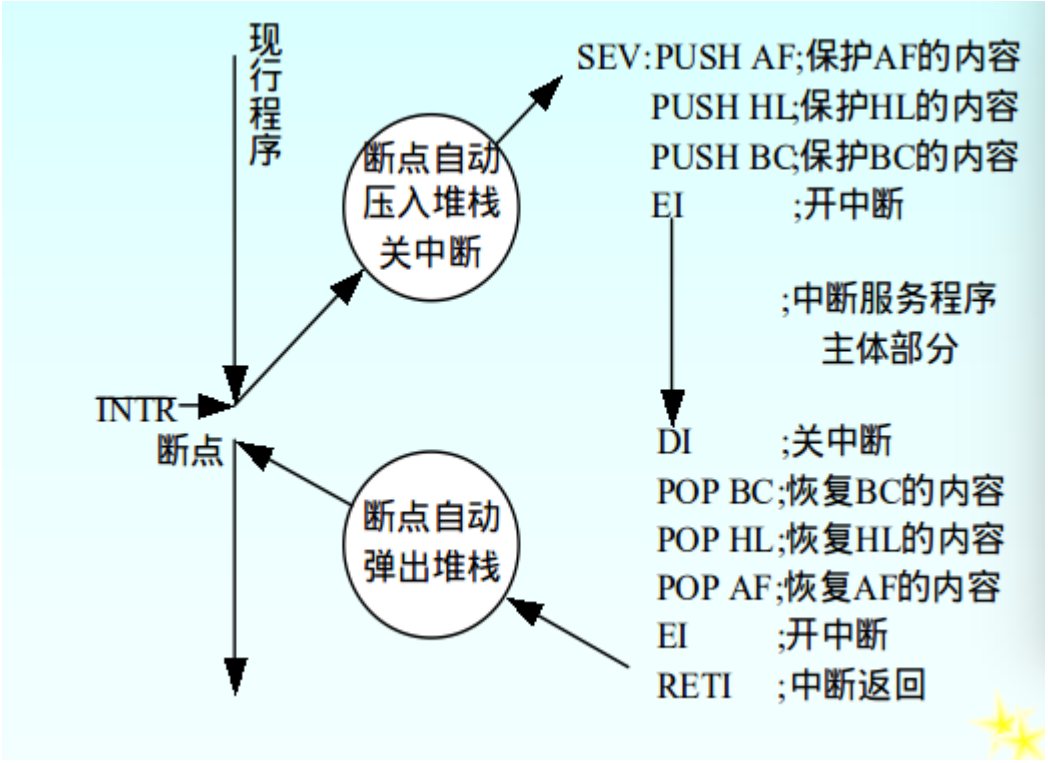
1. 中断源发出中断信号，cpu接收中断信号
2. cpu允许中断
3. 一条指令执行完毕

8. 中断隐指令

1. 保存断点
2. 关中断 : 防止保护的现场信息被新来的中断打断破坏
3. 引出中断处理程序

9. 中断现场的保护和恢复

1. 现代计算机一般都先采用硬件方法来自动快速的保护和恢复部分重要的现场，其余寄存器的内容再由软件完成保护和恢复，这种方法的硬件支持是堆栈。

2. 

10. 允许禁止中断和中断屏蔽

1. 允许中断
   - 返回主程序
   - 保护中断现场之后
2. 关中断
   - 禁止其他中断打扰的情况
   - 中断服务程序的保护和回复现场的阶段
3. 中断屏蔽
4. 触发器
   - 中断屏蔽触发器 : 构成寄存器
   - 中断允许触发器
   - 中断屏蔽触发器 : 构成寄存器

11. DMA

1. DMA的特点
   - 它使主存与CPU的固定联系脱钩，主存既可被CPU访问，又可被外设访问
   - 在数据块传送时，主存地址的确定，传送数据的计数等等都用硬件电路直接实现
   - 主存中要开辟专用缓冲区，及时供给和接收外设的数据
   - DMA传送速度快，CPU和外设并行工作，提高了系统的效率
   - DMA在开始前和结束后要通过程序和中断方式进行预处理和后处理
2. DMA接口(控制器)

   在**DMA**传送过程中，**DMA**控制器将接管**CPU**的地址总线、数据总线和控制总线，**CPU**的主存控制信号被禁止使用,而当**DMA**传送结束后，将恢复**CPU**的一切权利并开始执行其操作。由此可见，**DMA**控制器必须具有控制系统总线的能力，即能够像**CPU**一样输出地址信号，接收或发出控制信号，输入或输出数据信号,完全代替**CPU**进行工作

   - 接受外设发出的DMA请求，并向CPU发出总线请求
   - 当CPU响应此总线请求，发出总线响应信号后，接管对总线的控制，进入DMA操作周期
   - 确定传送数据的主存单元地址及传送长度，并能自动修改主存地址计数值和传送长度计数值
   - 规定数据在主存与外设之间的传送方向，发出读/写或其他控制信号，并执行数据传送的操作。

- 向CPU报告DMA操作的结束

3. DMA传送方法
    - CPU停止访问主存法 :
        1. 用DMA请求信号迫使CPU 让出总线控制权
        2. 在DMA操作期间，CPU处于保持状态，停止访问主存，仅能进行一些与总线无关的内部操作，直到DMA将操作权返还给CPU
    - 存储器分时法 :
        1. 把原来的一个存取周期分成两个时间片，一片给CPU，一片给DMA，使CPU和DMA交替地访问主存
        2. 不需要申请和归还总线
        3. CPU既不停止现行程序的运行，也不进入保持状态
        4. 但是因为大多数的外设速度的底下，容易导致很多的周期外设执行很多的空操作
    - 周期挪用法
        1. 一旦外设有DMA请求并获得CPU批准后，CPU让出一个周期的总线控制权，由DMA控制器控制系统总线，挪用一个存取周期进行一次数据传送，传送一个字节或一个字，然后，DMA控制器将总线控制权交回CPU，CPU继续进行自己的操作， 等待下一个DMA请求的到来；
        2. 如果在同一时刻，发生**CPU**与**DMA**的访存冲突，那么优先保证**DMA**工作，而CPU等待一个存取周期。若DMA传送时CPU不需要访存，则外设的周期挪用对CPU 执行程序无任何影响

4. DMA传送过程
    1. DMA预处理
        1. 打扰CPU一次
        2. 完成DMA控制器的初始化操作
    2. 数据传送
    3. DMA后处理
        1. 打扰CPU一次
        2. 执行相关的中断处理程序

题

2. 数据的机器层次表示

1. n位字长的二进制定点整数，其中一位是符号位，分别写出对于两种情况下的补码和反码的情况下的模数是多少

    $2^{n-1}/2^n$

5. 存储系统和结构

1. $5-16$