计组复习

大二下复习

- 计组复习
  - 第一章 计算机系统概论
    - 计算机系统简介
    - 计算机系统的层次结构
    - 计算机结构与组成
    - 计算机系统基本组成
      - 冯·诺依曼模型计算机
      - 计算机硬件的基本组成
        - 现代计算机硬件结构
    - 计算机系统的性能指标
      - 性能指标
        - 硬件性能参数
        - 计算机性能指标
      - 计算机硬件的性能设计
      - 计算机硬件的性能设计
  - 第四章 指令系统
    - 指令系统的组成
      - 指令功能
      - 指令格式
    - 数据存放与寻址方式
      - 操作数的存放方式
        - 数据在寄存器中的存放方式
        - 数据在存储器中的存放方式
        - 堆栈存取方式
      - 寻址方式
        - 指令寻址
        - 数据寻址
    - 指令格式举例
      - Pentium指令系统（变长指令字结构）
      - Power PC指令系统（定长指令字结构）
    - 指令系统发展

- 第五章 中央处理器
  - CPU的结构和工作原理
    - CPU的功能
    - CPU的基本结构
    - CPU的寄存器组织
    - CPU的工作流程
    - 指令执行过程
    - 数据通路组织
  - 控制器的组成与工作原理
    - 控制器的基本结构
    - 时序系统组成
      - CPU工作流程的相关周期及时序
      - 时序系统的基本组成
    - 微操作控制信号的时序控制方式
      - 同步控制方式
      - 异步控制方式
      - 联合控制方式
    - 微操作控制信号的形成
      - CPU基本操作的实现与微操作命令序列
      - 微操作控制信号的形成
  - 硬布线控制器
    - 微操作控制信号形成电路的设计方法
    - 例子
  - 微程序控制器
    - 微程序控制思想
    - 微程序控制器的组成原理
    - 微指令格式和编码方式
      - 微指令格式
      - 微指令编码方式
    - 微指令地址形成方式
      - 指令地址形成方式
      - 应用
    - 微程序控制器设计
    - 其它微程序设计方法
  - CPU举例
  - 指令流水技术
    - 指令流水线基本原理
      - 指令流水线的工作原理
      - 指令流水线的基本要求

流水线的分类
流水线的性能

指令流水线的相关和处理
高性能指令流水线

第六章 总线
总线概述
总线的分类
按总线信号线功能分类
按总线连接部件匪类

总线的特性
总线的性能指标

总线传输与控制
总线操作
总线仲裁
集中式仲裁
分布式仲裁

总线定时
总线标准

总线互联结构
总线互联的结构
单总线结构
多总线结构

总线互联的实现

第七章 输入/输出系统
I/O系统概述
I/O系统的基本组成
I/O系统的硬件
I/O系统的软件

I/O设备与主机的连接方式
I/O设备与主机的连接方式
I/O设备的编址方式
I/O设备的寻址方式
I/O设备与主机的联络方式

I/O设备与主机的传送控制方式

外部设备
输入设备
输出设备

存储设备
辅存的性能指标
磁盘存储器
磁盘阵列

I/O接口
I/O接口功能
I/O接口的硬件组成
I/O接口的软件组成

程序查询方式
程序查询方式的流程
程序查询方式的接口组织
无条件传送方式

程序中断方式
中断的分类
按中断请求分
按中断源识别及中断服务程序入口地址获得方法分
按中断处理能否重叠分类

I/O中断的过程
I/O中断的组织
I/O接口的组织
中断源识别的组织
中断控制器的组织
中断系统组成
多重中断的组织
中断屏蔽的组织

DMA方式
DMA的传送方式
DMA接口的基本组成
DMA的数据传送过程
DMA的组织

通道方式

第一章 计算机系统概论

问题：

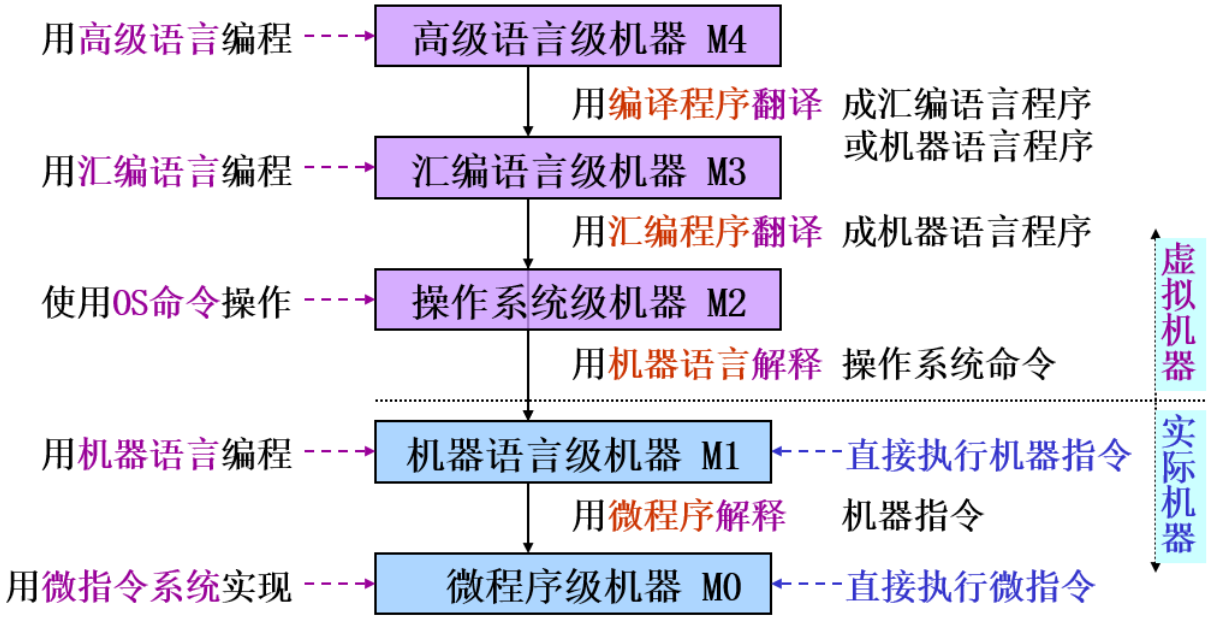
1. 冯·诺依曼模型中，为何存储单元长度应该是指令和数据长度的最小者的长度？

计算机系统简介

计算机的基本功能：

1. 数据处理
2. 数据存储
3. 数据传送
4. 过程控制

计算机系统的层次结构

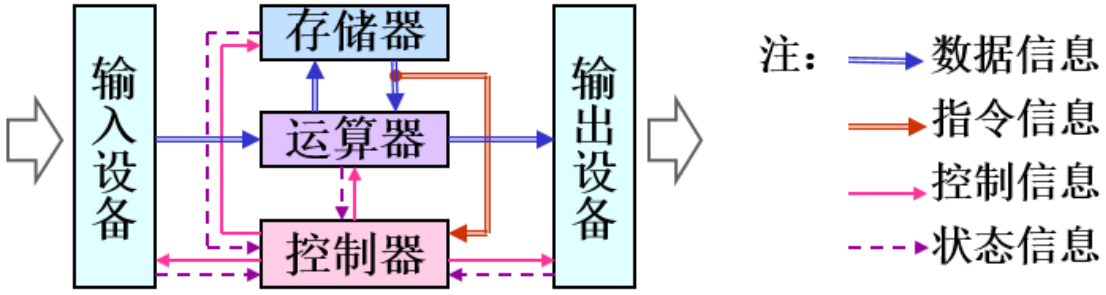


计算机结构与组成

- **计算机系统结构**是指**程序员**看到的计算机系统的属性，包括概念性结构和功能特性。精确地讲，计算机系统结构是机器语言程序员或编译程序编写者所看到的传统级机器的属性，包括指令集、数据表示、寻址方式、存储系统组织、I/O系统组织等抽象属性或参数。主要研究计算机系统软、硬件交界面的定义及其上下的功能分配。
- **计算机组成**是指计算机**硬件设计人员**所看到的属性，它包含了许多层次程序员来说是透明的硬件属性，包括专用部件及数据通路宽度设计、功能部件并行度、控制机构组成方式等。主要研究如何合理地**逻辑实现**计算机系统结构分配给硬件的功能。
- **计算机实现**则主要研究**机器和微组装**技术。

计算机系统基本组成

冯·诺依曼模型计算机



> 存储程序原理：程序和数据预先放在存储器中，机器工作时自动按程序的逻辑顺序从存储器中逐条取出指令并执行。

- 以运算器为中心

计算机硬件的基本组成

现代计算机硬件结构

CPU = ALU + CU

1. 以存储器为中心的结构
    - 为了实现数据传送与数据处理可以并行执行
2. 多种存储器共存的结构
    - 主存+辅存
3. 采用总线互连的结构

计算机系统的性能指标

性能指标

> 计算机性能指运行在计算机**硬件**上的计算机**软件**的性能。

硬件性能参数

> 计算机硬件的性能只能称为计算机系统的性能参数，而不能称为性能指标。

1. 机器字长（CPU字长）：CPU一次能处理数据的二进制位数。决定了ALU、数据通路、寄存器的长度。
2. 机器主频（CPU主频）：主时钟的脉冲频率 。
3. 主时钟周期：CPU主时钟脉冲长度($1/f$)，常简称为时钟周期。
4. 主存容量：计算机主存中能够存放信息的最大二进制位数。

计算机性能指标

1. 响应时间：计算机完成某任务从任务输出到结果输出的全部时间

    - $T_{响应} = T_{CPU} + T_{IO等待}$
    - $T_{CPU} = I_N \times CPI \times T_C$，$I_N$为机器指令的条数，CPU的时钟周期为$T_C$，任务对应程序中每个机器指令执行时平均需要$CPI$个时钟周期。

2. 吞吐量（吞吐率）：单位时间内计算机能够完成的工作量。

    - **MIPS**：每秒百万条指令数，对于给定的n个程序$MIPS = \frac{n个程序的指令总条数}{n个程序的总执行时间 \times 10^6}$
    - **MFLOPS**：每秒百万次浮点操作次数，对于给定的n个程序
      $MIPS = \frac{n个程序的浮点操作总次数}{n个程序的总执行时间 \times 10^6}$

计算机硬件的性能设计

1. 冯·诺依曼模型计算机的性能瓶颈

    1. CPU-MEM瓶颈：由摩尔定律，CPU与MEM间的速度差异日益突出
        - 需从结构与技术方面解决速度匹配问题。

    2. 指令串行执行瓶颈：无法并行化处理，导致性能受限。
        - 快速串行处理，受限于器件技术，需要从器件技术方面提高性能，但效果有限。
        - 并行处理，受限于指令间的相关性，需要从结构和技术方面解决问题，是主流方向。

2. 性能平衡设计

    - 目标是解决CPU-MEM访问瓶颈，减少访存延迟，提高访存效率。可采用多种存储器构成层次结构的寄存系统，采用多级总线互连，提高存储器本身的速度（采用并行结构存储器）。

3. CPU性能设计

    - 目标是解决指令串行执行瓶颈。主要指通过多种手段优化CPU结构，减轻指令串行执行方式对采用并行处理技术后CPU性能的影响。
    - 流水线技术，超标量技术，超线程技术，多核技术等并行处理技术。

计算机硬件的性能设计

第四章 指令系统

问题：

1. 霍夫曼扩展编码？
2. 地址码字段指明各个操作数的源/目标属性？
3. 间接寻址为什么能扩大寻址范围？

指令系统的组成

> 通常将要求硬件直接实现某种运算或操作的命令成为**机器指令**，将所有机器指令的集合称为**机器指令系统**，简称**指令系统**。

1. 应用需求必须按照指令系统约定的功能，形成计算机软件（程序）。
2. 计算机硬件必须按照指令系统约定的功能，进行硬件组成设计。

- 指令系统实际上是软硬件之间的一个“约定”。

指令功能

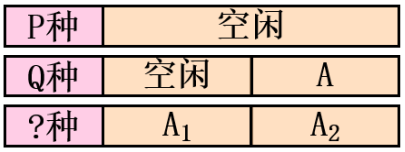
1. 指令的操作数
2. 指令的操作
    1. 数据传送：寄存器-寄存器、寄存器-存储单元、存储单元-存储单元
    2. 算逻运算：算术运算和逻辑运算
    3. 转移操作：条件转移、非条件转移、调用、返回

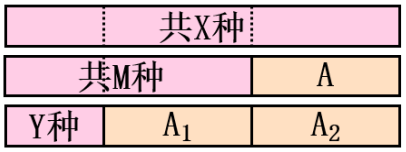
指令格式

> 操作码+地址码。

1. 操作码字段
    - 霍夫曼编码和**霍夫曼扩展编码。**
2. 地址码字段
    - 操作数地址以及下条指令地址，各个操作数的源/目标属性（？）。

例4.3 设某机器指令字长为16位，每个地址码有6位，指令有零地址、单地址、双地址三种格式。
（1）操作码采用定长编码，零地址指令、单地址指令分别为P种和Q种，则双地址指令最多有多少种？（2^4-P-Q种）
（2）操作码采用变长扩展编码，零地址指令、单地址指令分别为X种和Y种，则单地址指令最多有多少种？（$X=[(2^4-Y)\times 2^6-M]\times 2^6$，$M$为单地址指令的种数）





3. 指令字

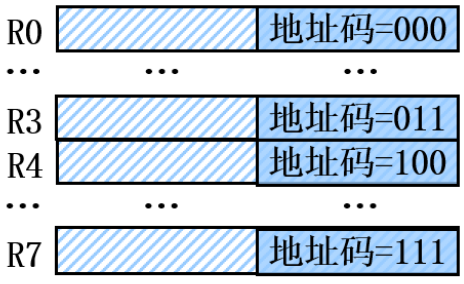
    - 指令字由操作码字段和地址码字段组成。
    - 指令字长指的是指令字中包含的二进制信息位数。
    - 机器字长指CPU能够直接处理的二进制位数。
    - 如果系统中所有指令的指令字长都相等，称这种指令系统为定长指令字结构。对应的变长指令字结构。

数据存放与寻址方式

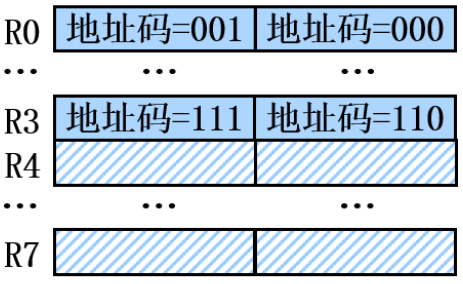
操作数的存放方式

数据在寄存器中的存放方式

1. 使用寄存器低端。
2. 使用部分寄存器*(OK)*。



方案1—使用REG低端

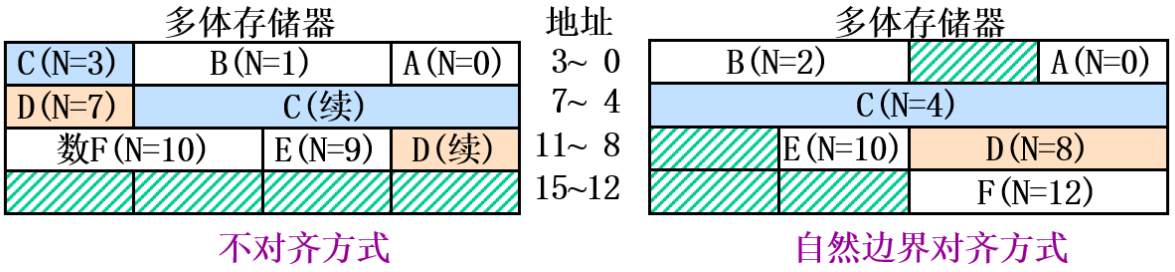


方案2—使用部分REG

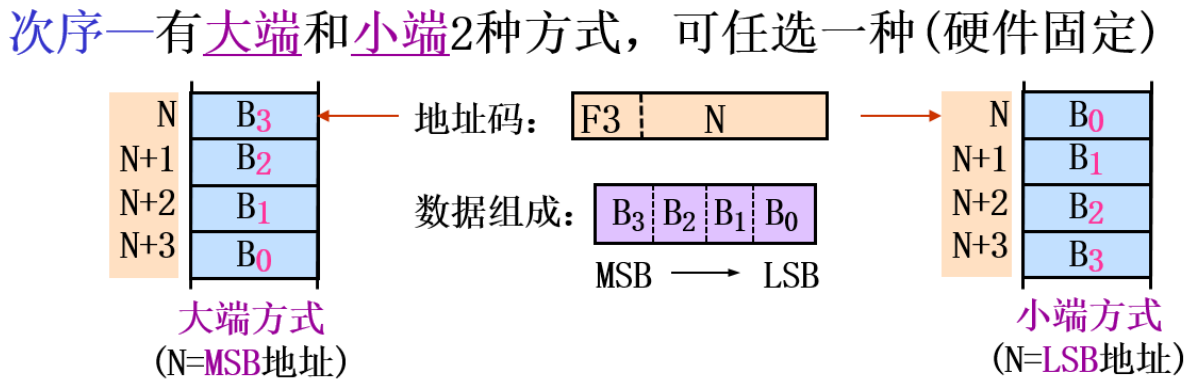
数据在存储器中的存放方式

- **边界**
- **次序**

1. 边界不对齐（节省空间，访问效率低）。
2. 边界对齐（保证访问性能，空间利用率低）。



1. 大端次序。
2. 小端次序。（Intel CPU采用小端次序）



堆栈存取方式

- 新元素入栈时，**SP(Stack Point)**向约定方向移动一个地址单位过后的地址才是元素入栈的起始地址。
- 栈底在首个元素入栈的起始地址。

寻址方式

指令寻址

> 形成下条**指令地址**的方法，有顺序寻址和跳跃寻址两种方法。
>
> 顺序型指令格式 | 操作码 | 操作数地址码 | 下条指令地址码(隐含)
>
> 转移型指令格式 | 操作码 | 操作数地址码(无) | 下条指令地址码

数据寻址

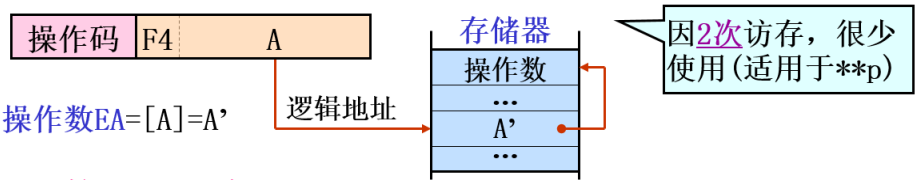
> 形成**操作数或操作数地址**的方法。地址码一般为寻址方式位I和形式地址A（得到数据所需要的参数，为数据或者寄存器地址、存储器地址等）组成

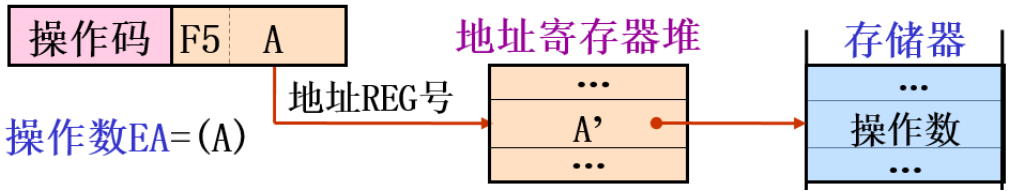
1. 立即寻址
    - 操作数 = A

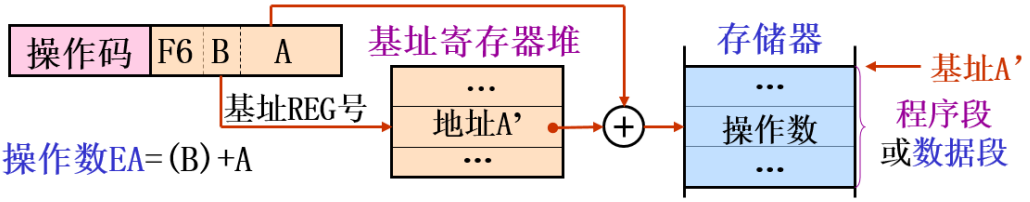
- 适合对常数的操作

2. 寄存器（直接）寻址
   - A为存放操作数的寄存器编号
   - 操作数 = (A)
   - 寄存器读写速度快，A位数少，指令字长短。

3. 直接寻址
   - A为存放操作数的存储器存储单元地址。
   - 操作数 = $[A]$
   - 形成简单，但指令字长较长，适合地址固定的存储器单元的操作。

4. 间接寻址
   - A为存放操作数的存储单元地址的存储单元地址。
   - 操作数 = $[[A]]$
   - 扩大了操作数的寻址范围(?)，适合用于指针操作。
   - 

5. 寄存器间接寻址
   - A为存放操作数的存储单元地址的寄存器编号。
   - 操作数 = $[(A)]$
   - 具有间接寻址的最大寻址访问、直接寻址的最少访存次数、寄存器寻址的最短A长度，是使用频率最高的寻址方式之一。
   - 

6. 基址寻址
   - 操作数在存储器中的存放地址 = A + 基址寄存器中的内容B
   - 此时A称为偏移地址
   - 操作数 = $(A+(B))$
   - 

   - 只有一个基址寄存器时（专用基址寄存器），不用显示表示REG号。

7. 变址寻址
   - 基址寄存器->变址寄存器、
   - 操作数 = $((I)+A)$
   - 往往是A不变，变址寄存器内容有规律地变化。

8. 相对寻址
    - 操作数地址为将A与程序计数器PC的内容（当前指令地址）相加的结果。
    - 操作数 = $((PC)+A)$
    - A称为偏移量。
    - 由于该方法是相对于PC寄存器的，故只能用于转移指令的跳跃寻址，不能用于数据寻址。

9. 隐含寻址
    - 由指令操作码隐含约定操作数地址或下条指令地址的形成方式。
    - $(PC)+1$、返回指令RET隐含约定返回栈顶等。

指令格式举例

Pentium指令系统（变长指令字结构）

Power PC指令系统（定长指令字结构）

指令系统发展

> $T_{CPU}=I_N\times CPI\times T_C$，$I_N$是程序执行时的指令条数，$CPI$为每条指令执行所需要的时钟周期数，$T_C$是每个时钟周期的延迟，$T_C$只与器件速度有关，与指令系统无关。

1. 从指令格式角度看，可以采取操作码为扩展编码、支持多种寻址方式、提高指令格式规整性等方法，为软件和硬件提供良好的支持。
2. 从指令功能角度看，由CPU时间公式可知，同时减小I_N和CPI时系统性能提高最快，但要减小I_N就必须增强指令功能，则会令CPI较大；要减小CPI就必须精简指令功能，而会导致I_N较大。

指令系统由此向两个方向发展：

1. 复杂指令集计算机(*Complex Instruction Set Computer,CISC*)，侧重增强指令系统功能，通过减少程序中指令条数来提高系统性能。Intel公司的Pentium是CISC结构处理器的典型代表。
2. 精简指令集计算机(*Reduced Instruction Set Computer,RISC*)，侧重精简指令系统功能，通过减小程序中指令执行时间来提高系统性能。RISC研究者发现CISC指令系统的不同指令使用频率相差悬殊，20%的指令使用了80%的运行时间。SUN公司的SPARC、IBM公司的RT PC等都是RISC结构初期的典型代表。

第五章 中央处理器

问题：

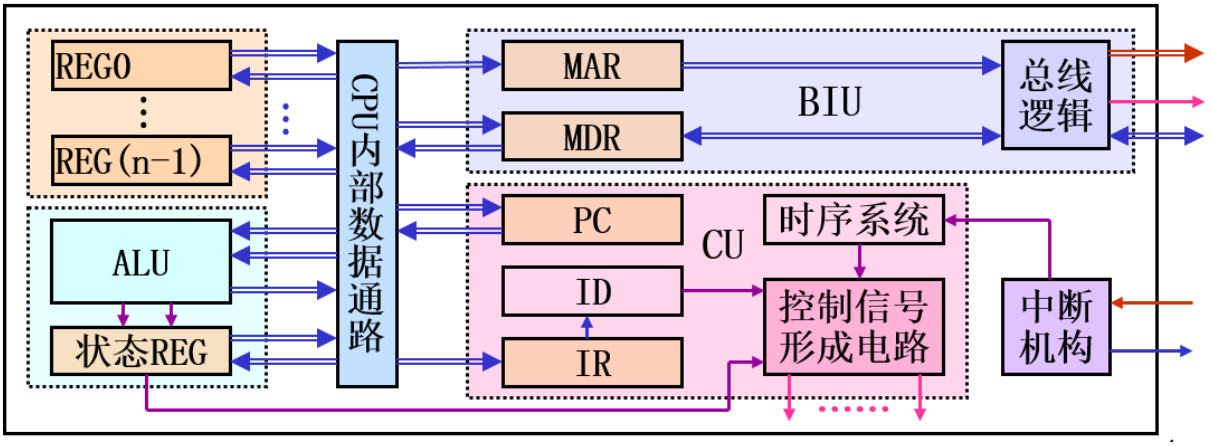
1. P183“机器周期的长度相对**固定**”和P184“机器周期长度可采用定长或**变长**方法组织”？
2. P184提到节拍周期可以按**定长**和**变长**方式组织，但又说相对**固定**？
3. P184“节拍周期所用时钟信号成为CPU主时钟信号，又称为CPU主时钟”，但图5.16中展示的CPU主时钟周期却和节拍脉冲是相同的？
    - CPU主时钟信号一个周期的长度是节拍周期的**有效时间。**
4. 什么叫“微操作控制所需的时间单位”？

CPU的结构和工作原理

CPU的功能

1. 指令控制
2. 操作控制
3. 时间控制
4. 数据加工
5. 中断处理

CPU的基本结构



- *CU(Control Unit)*，控制单元
- *BIU(Bus Interface Unit)*，总线接口单元

CPU的寄存器组织

寄存器是一种可稳定存储信息、**速度最快**的部件。大致可分为：

1. 用户可见寄存器
    - 用户编程时可以使用，来减少程序访存次数
    - 可分为数据寄存器（存放指令操作数，同时存放源和目标操作数的寄存器称为累加寄存器*AC*）、地址寄存器、通用寄存器、条件寄存器（存放仅为标志位*C*、负标志位*N*、零标志位*Z*等）

2. 控制和状态寄存器：
    1. 程序计数器（*PC*）
    2. 指令寄存器（*IR*）
    3. 存储器地址寄存器（*MAR*）
    4. 存储器数据寄存器（*MDR*）
    5. 程序状态字寄存器（*PSW*）：用于存放程序执行状态的寄存器，如是否为单步执行状态、是否为设置断点状态，**通常只允许操作系统设置** ，大多数计算机常将条件码寄存器和*PSW*合二为一，统称**状态寄存器或标志寄存器**

CPU的工作流程

> **指令周期**：CPU取出并执行一条指令所需的全部时间。
> **CPU周期**：通常将取值周期、间址周期、执行周期、DMA周期、中断周期这些**基本功能段**称为CPU周期。

1. 总的来讲指令周期由取值周期和执行周期组成。但不同指令的指令周期都可能不同，为了统一，必须按最长的来进行循环。
2. 当前指令周期结束后应检测是否有I/O设备的DMA请求，若有，需**让出总线控制权**，进入**DMA周期。**
3. 当前指令周期结束后应检测是否有I/O设备的中断请求，有则进入**中断周期**，准备**在下一个指令周期执行中断服务程序。**

指令执行过程

1. 指令的执行过程
    1. 取数指令的执行过程
    2. 存数指令的执行过程
    3. 加法指令的执行过程
    4. 条件转移指令的执行过程
    5. 无条件转移指令的执行过程（双字长指令）

2. CPU的基本操作

    > 将CPU内部实现基本功能的操作称为**CPU的基本操作**。

    1. 寄存期间数据传送：实现两个寄存器间的数据传送，$R_{源}->R_{目标}$
    2. 存储器读：$MAR->MEM(读)->MDR$

3. 存储器写：$(MAR+MDR)->MEM(写)$
4. 算逻运算：$R_{源1}和R_{源2}->ALU(运算)->R_{结果}$

3. 微操作

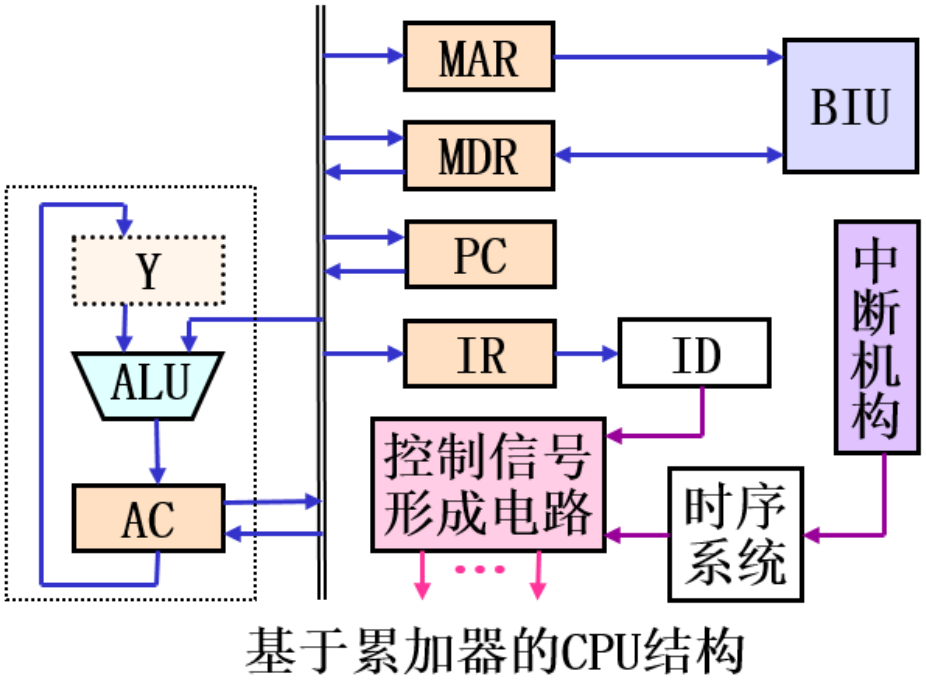
> 将CPU内部的原子操作称为**微操作**。

数据通路组织

> - 数据通路是CPU内部部件间传送数据的物理通道。
> - 将能够并行执行的**微操作**(uOP)称为**微操作步**$(uOPs)$，将用微操作步表示的**微操作序列**$(uOP序列)$称为**微操作步序列**$(uOPs序列)$。

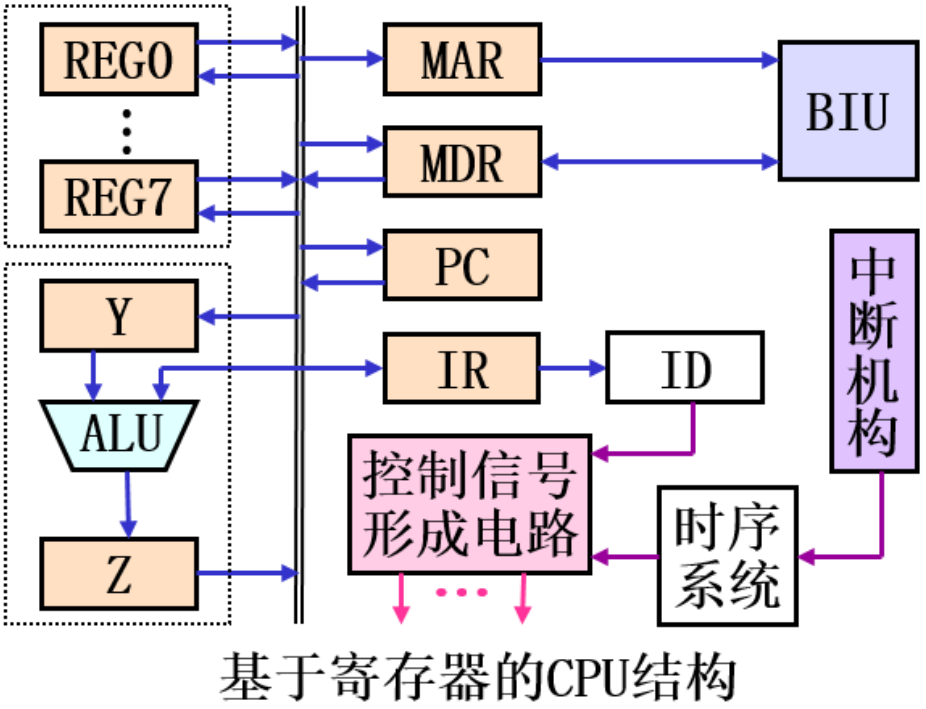
- uOP序列：每个uOP一行。
- $uOPs$序列：不冲突的可以合并。

1. 总线通路
2. 单总线通路。
   - 基于累加器的CPU



基于累加器的CPU结构

   - 基于寄存器的CPU



3. 多总线通路
    - 目标是解决ALU的两个入端和一个出端与源操作数及目标操作数存放部件间的数据传送冲突问题。

4. 专用通路

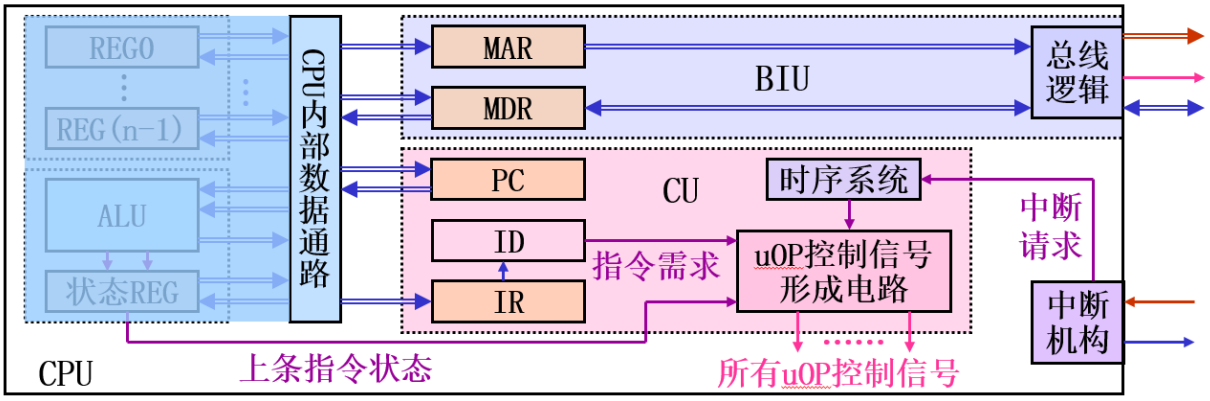
- 只要不存在部件冲突的数据传送都可以同时进行。
    1. 单总线通路CPU的指令执行过程

控制器的组成与工作原理

- CPU主要有指令控制、操作控制、时间控制、数据加工、中断处理五大功能，除数据加工外均由控制器实现。
- CPU的工作流程就是循环执行微操作步序列中的所有微操作。要执行一个微操作，只要给相关部件发出控制信号即可，通常将这些信号称作**微操作控制信号**。执行一个微操作步的过程，就是同时发出微操作步内各个微操作控制信号的过程。因此，**控制器的功能**就是自动、依次发出CPU工作流程对应微操作步序列中的所需的微操作控制信号。

控制器的基本结构

控制器由控制单元CU、中断机构及总线接口单元BIU组成。



时序系统组成

CPU工作流程的相关周期及时序

由于指令功能、寻址方式都会严重影响指令的执行时间，若指令周期用固定长度的基准信号表示，则CPU性能特别差，因此，指令周期一般不作为某级时序，而用End信号表示指令周期是否已结束。

1. 机器周期

> 机器周期是按CPU工作流程中的基本过程划分的时间，如取值周期、间址周期、DMA周期及中断周期。

- 每个机器周期设置一个触发器表是指示其状态，时序系统轮流使其有效来表示当前处于哪一个周期。
- 由于机器周期长度较大 ，时序系统组织当前CPU工作流程中包含哪些机器周期时，通常采用**按需组织**的方法。

2. 节拍周期

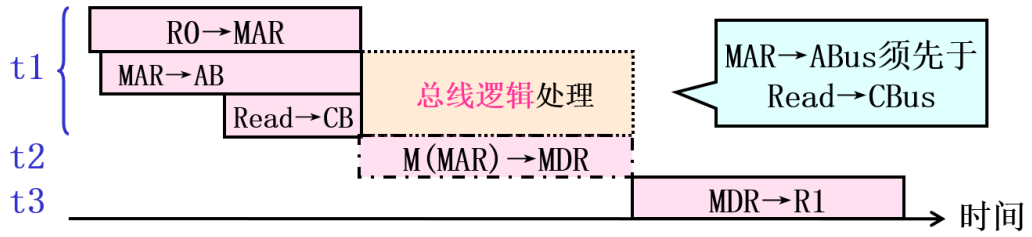
> 节拍周期是指完成CPU内部最基本操作$\boldsymbol{uOP}$所需要的时间。

- 节拍周期所用时钟信号成为CPU主时钟（信号），CPU主时钟的频率称为**CPU主频**或**机器主频**。
- 每个节拍周期设置一个触发器表示其状态，时序系统轮流使其有效。
- 定长按$\boldsymbol{max\{uOP_i\}}$设置长度，变长按某种微操作步序列的需要组织。

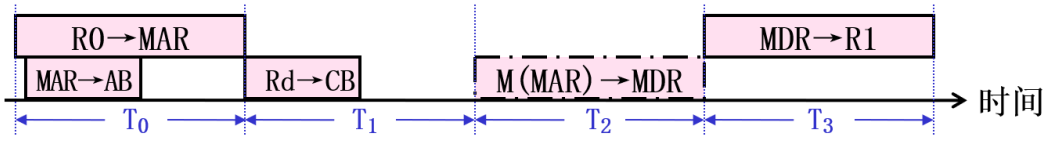
3. 节拍脉冲

> 节拍脉冲指节拍周期内部微操作控制所需要的时间单位。一个节拍周期通常包含两个节拍脉冲。

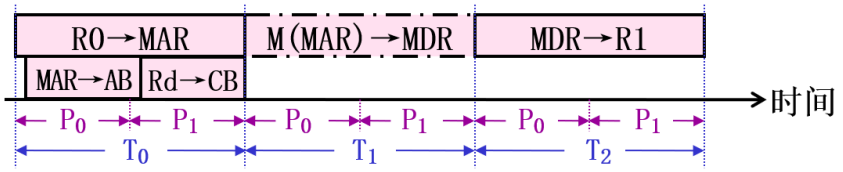
例—指令R1←[R0]执行阶段的uOPs序列如下



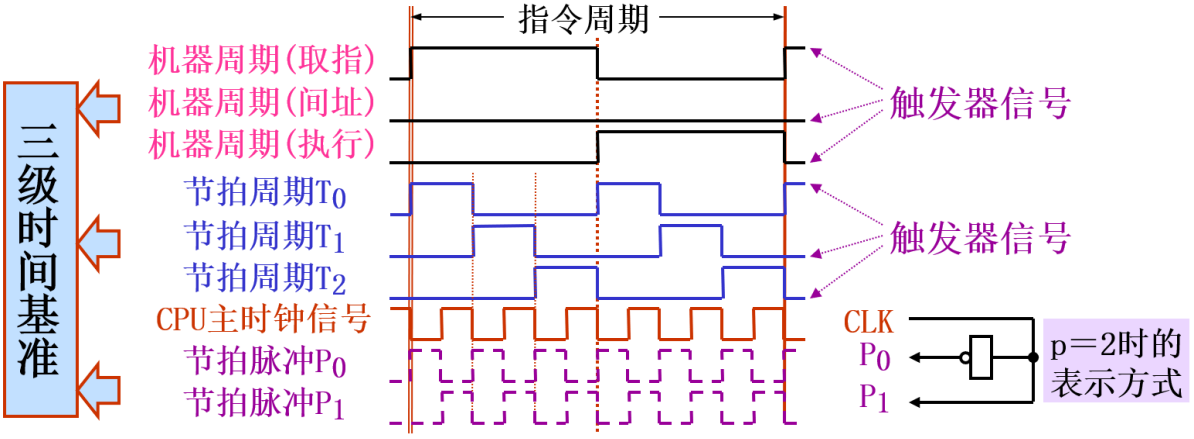
机器周期组成方案一：4个节拍周期



机器周期组成方案二：3个节拍周期、2个节拍脉冲



- 由于CPU主时钟CLK频率固定，因此，节拍周期及节拍脉冲长度固定。而机器周期长度可以采用定长或变长方式组织。

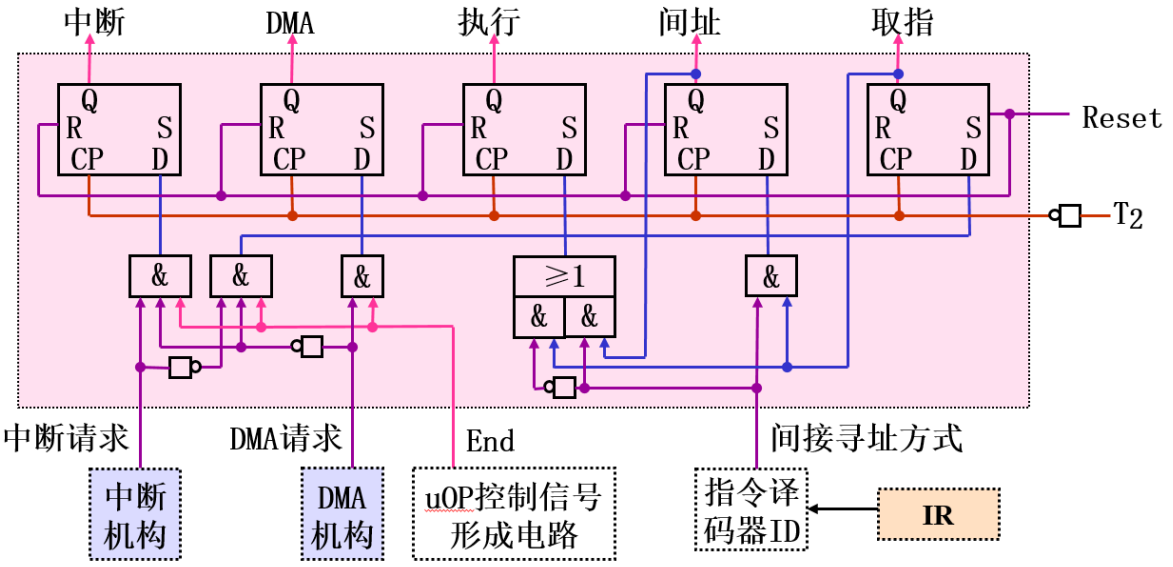


*状态表示：用门电路或触发器表示其状态

（p=2时） （p>2时）

时序系统的基本组成

- 硬布线控制器的时序系统是由机器周期、节拍周期和节拍脉冲组成的三级时序系统。
- 微程序控制器是由节拍脉冲组成的单级时序系统。



微操作控制信号的时序控制方式

> 通常将微操作控制信号时序(时长以及顺序)的确定方式称为控制器的控制方式。

同步控制方式

- 顺序：微操作控制信号根据微操作步序列的需要与某些时标信号绑定，来决定出场时间。如取值周期的触发器信号为IF，则取指令阶段$PC->MAR$微操作的有效条件为$IF\&T_0+\ldots$。
- 时长：
    1. 采用一种长度的定长机器周期。
    2. 采用几种长度的定长机器周期。如分别统计取值、间址、执行、DMA机器周期所需节拍周期的最大值，分别采用即可。
    3. 采用变长的节拍周期：中央控制和局部控制相结合的方法，局部控制起到延长中央控制的机器周期的效果。

异步控制方式

> 微操作控制信号的时序只受专门的应答线路控制。

- **应答协议/握手协议**：CU发出某微操作控制信号后，等待相关部件完成后发回反馈信号，CU再及时发出下个微操作控制信号。

联合控制方式

> 微操作开始时转入异步控制方式，微操作结束时转回同步控制方式。

实现思路：延长节拍周期（通过WMFC信号封锁/释放节拍周期信号发生器的CP）。

微操作控制信号的形成

> 微操作的功能可以通过同时使相关部件的操作控制信号有效来实现，这些有效的操作控制信号又称为**微操作命令**，$uOPs$序列的实现需求可以用**微操作命令序列**表示。

CPU基本操作的实现与微操作命令序列

1. 寄存期间数据传送的实现：$R_{X_{out}}$，$R_{Y_{in}}$
2. 存储器读和存储器写的实现：
    - 读：①$Read$ ②$WMFC$
    - 写：①$Write$ ②$WMFC$
3. 算逻运算的实现：①$R1_{out}$、Y_{in} ②$R2_{out}$、ADD、Z_{in}
4. 其他功能操作的实现：PC_{+1}、End等

> 微操作步序列到微操作命令序列的转换，仅仅是**微操作到微命令**的转换。

微操作控制信号的形成

> 微操作命令的有效条件应该是“序列名称&微操作步序号”

硬布线控制器

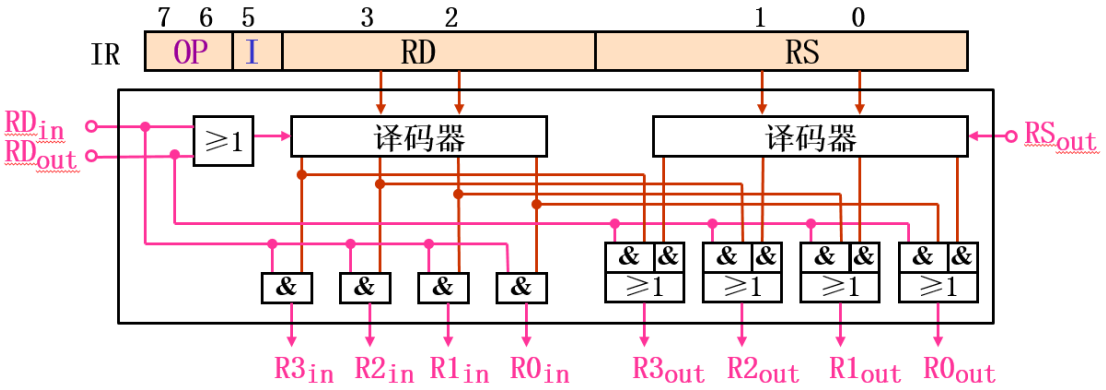
> - 硬布线控制器的时序系统有机器周期、节拍周期以及接拍脉冲三级时序信号。
> - 硬布线控制器又分为组合逻辑控制器（组合逻辑电路）和PLA控制器（可编程逻辑阵列）。

微操作控制信号形成电路的设计方法

1. 列出所有微操作命令序列，并按取值周期、间址周期、执行周期划分成子序列。
2. 确定时序系统相关参数，如机器周期、节拍周期、节拍脉冲的长度。
3. 形成所有微操作命令的使用时间表以及有效逻辑表达式。
4. 画出微操作控制信号形成电路以及相关部件的连接图。

例子

- 注意寄存器微操作控制信号的形成



- RISC通常采用硬布线控制器，CISC通常采用微程序控制器。

微程序控制器

微程序控制思想

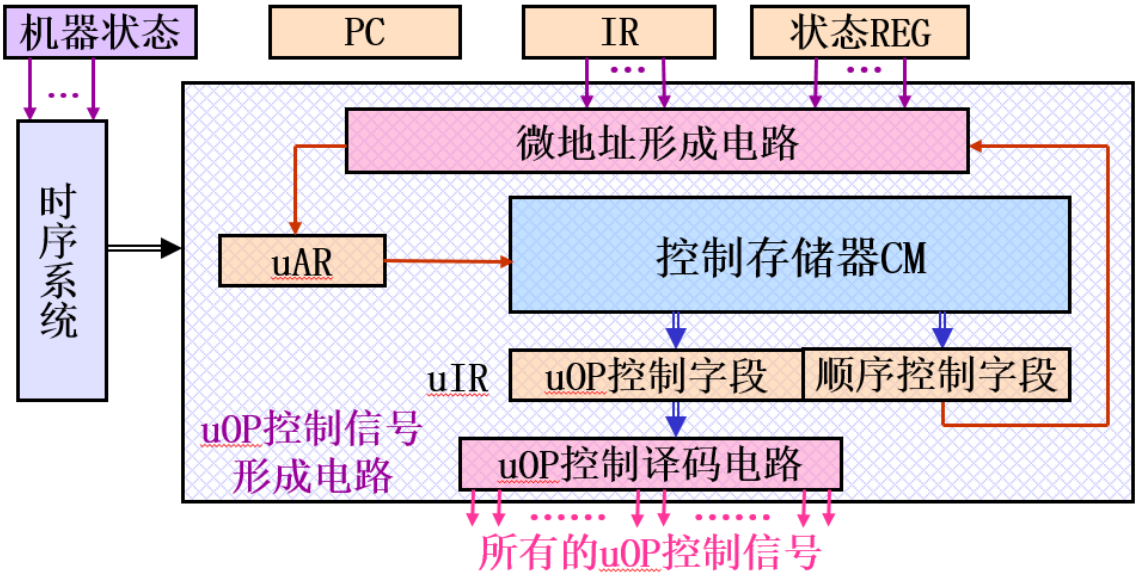
> - 微命令：实现位操作所需的部件控制信号，微操作是微命令实现的操作功能。
> - 微指令：按一定格式编排、用二进制编码表示、可同时执行的**一组微命令**。
> - 微程序：指完成特定功能的**微指令序列**，如取值微程序、*ADD*指令操作微程序等。
> - 控制存储器：专用于存放微程序的存储器，简称控存*(Control Memory,CM)*。
> - 微指令周期：取出并执行一条微指令所用的时间。

将CPU工作流程对应的微操作命令序列编写成微程序，所有的微程序存放在控存中，控制器自动按照微程序的逻辑顺序，逐条取出微指令并执行，来实现CPU工作流程的控制。

微程序控制器的组成原理

- 为了便于控制和减少微指令周期时长，微指令均采用**定长指令格式**。
- 微程序控制器的时序系统只需要**节拍脉冲**信号一级时序。

1. 微程序控制器的基本组成



2. 微程序控制器的工作原理

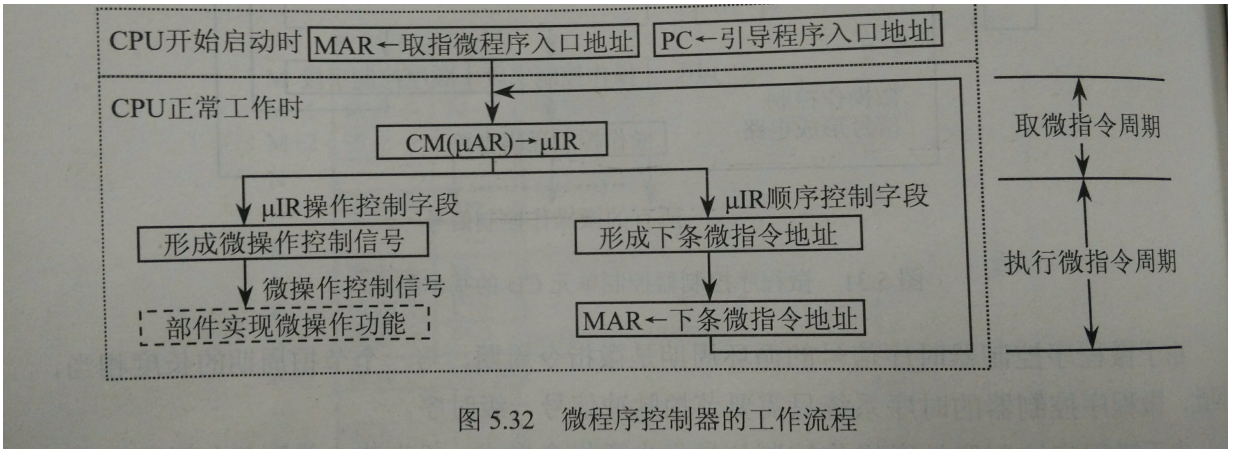
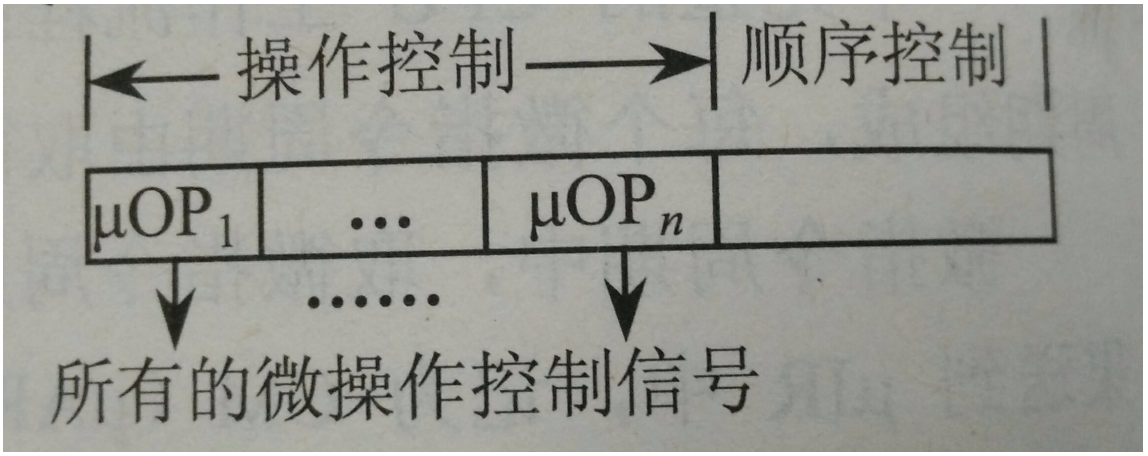


图 5.32　微程序控制器的工作流程

微指令格式和编码方式

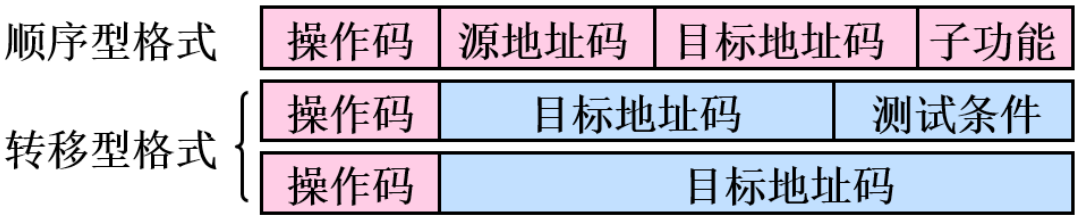
微指令格式

1. 水平型微指令



- 下条微指令地址的形成方式多样，顺序控制字段必须显示表示。
- 同时定义并执行多个微指令。

2. 垂直型微指令

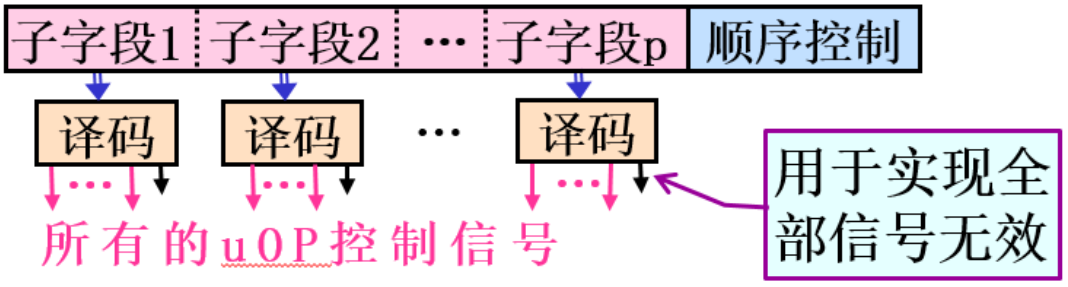


- 同时定义并执行一个或几个微命令的微指令格式。
- 顺序型微指令的顺序控制字段缺省表示、转移型微指令顺序控制字段显示表示。

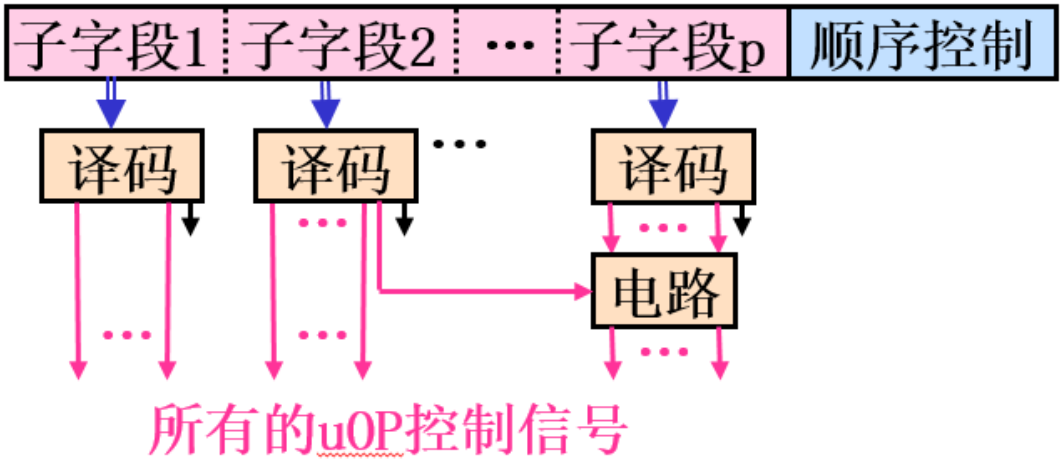
1. 水平型格式微指令的微操作能力强、灵活性强、效率高。
2. 水平型格式微指令的微程序执行速度快。
3. 水平型格式微指令的微程序代码效率低。（存储单元浪费，微程序所占CM的存储空间大，控制器成本增加。但**不影响CM中微指令的访问速度**，因为访存速度只与地址长度有关、与存储单元长度无关）。

微指令编码方式

1. 直接编码方式：每个位表示一个微命令，1有效，0无效。操作控制字段长度等于总的微命令个数。
2. 字段直接编码方式：操作控制字段分成若干个**子字段**，子字段的每个编码表示一组**互斥**微命令中的一个。



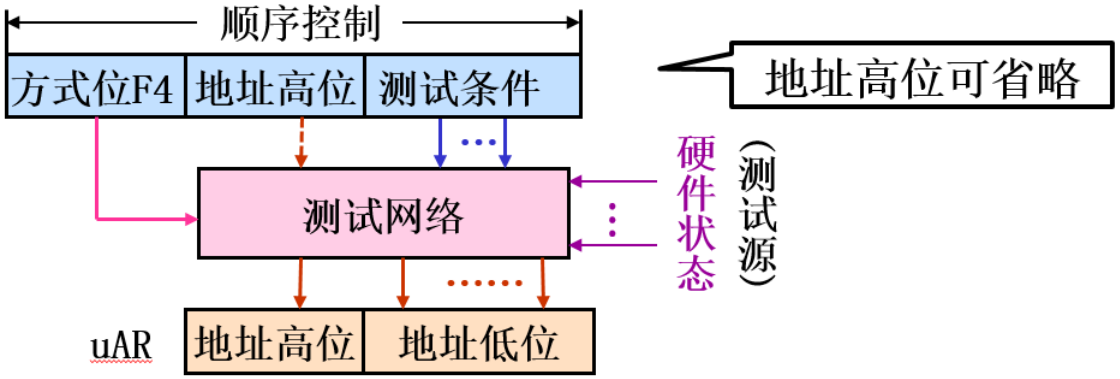
3. 字段间接编码方式：在字段直接编码方式上，约定部分微命令需要用多个子字段中的编码组合表示。



微指令地址形成方式

指令地址形成方式

1. 计数器法：$\mu AR = (\mu AR) + 1$。
2. 下址法：$\mu AR =$下址子字段。
3. 测试网络法：$\mu AR = func$(测试源，测试条件)



4. 硬件产生法：$\mu AR =$固定地址。

应用

1. 垂直型微指令格式
   - 顺序型微指令只能采用计数器法。（顺序控制字段必须隐式表示）
   - 转移型微指令采用下址法较优，顺序控制字段只由下址子字段组成。
   - 条件转移微指令的转移条件通常放到操作控制字段中。

2. 水平型微指令格式
   - 顺序型微指令采用计数器法、下址法均可。（顺序控制字段必须显示表示）

顺序型：	方式位00	空闲	方式位00	下址	
无条件转移型：	方式位01	空闲	方式位00	下址	
条件转移型：	方式位10	测试条件	方式位01	地址高位	测试条件
多路分支型：	方式位11	测试条件	方式位10	地址高位	测试条件
	顺序型采用计数器法		顺序型采用下址法		

微程序控制器设计

1. 列出所有微操作命令序列。
2. 设计微指令集格式。
3. 编制微程序。
4. 设计相关电路。

其它微程序设计方法

1. 毫微程序设计
   - 用来解释微指令的毫微指令序列。
   - 目的是解决垂直型微指令并行操作能力弱的问题。
   - 采用两级微程序的设计方法：第一级为垂直型微指令，第二级为水平型微指令。

2. 动态微程序设计
    - 用EPROM构成CM。

CPU举例

指令流水技术

指令流水线基本原理

指令流水线的工作原理

1. 将流水线中细化的操作阶段称为**段**，将指令通过某个段的时间长度称为**拍**。

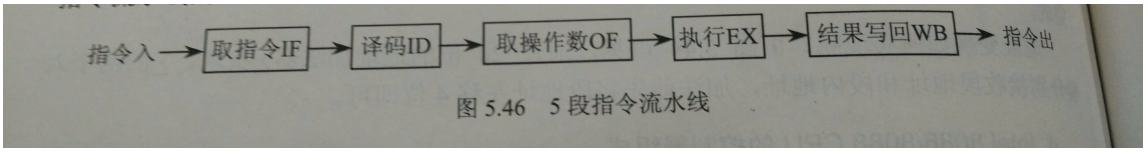
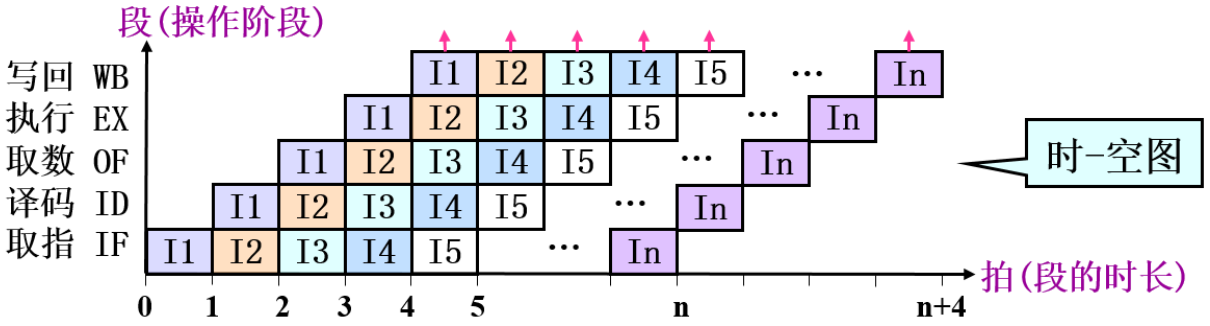


图 5.46　5 段指令流水线



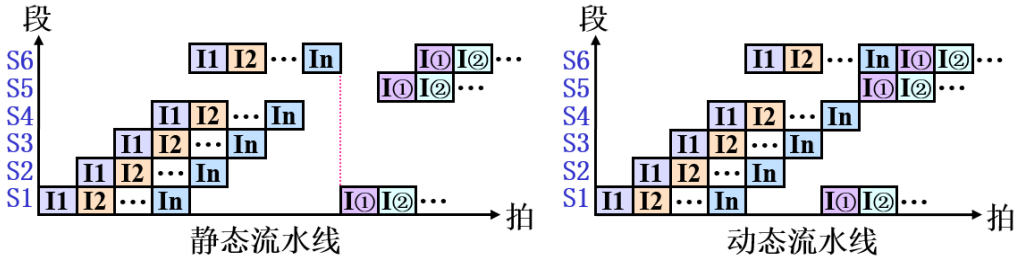
指令流水线的基本要求

1. 流水线各个段的操作互相独立。
2. 流水线的各个段的操作同步

流水线的分类

1. 单功能流水线和多功能流水线
    - 单功能流水线是只能完成一种功能的流水线（如浮点加法流水线）
    - 多功能流水线是指各个段可以在不同时段或同一时段内进行不同的连接。

2. 静态流水线和动态流水线
    - 静态流水线是指同一时段内，流水线的各个段只能按一中功能的连接方式工作，功能切换需串行进行（排空后再换）。
    - 动态流水线是指同一时段内，某些段在实现一种功能时，其他段可以实现其它功能。

- 

3. 线性流水线和非线性流水线
    - 线性流水线没有反馈回路，指令随时可以流入线性流水线。
    - 非线性流水线指各个段间存在反馈回路，常用于递归操作。

4. 标量流水线和向量流水线。
    - 标量数据可表示定点数、浮点数、字符等单个数据。
    - 向量数据可表示标量数据中的多个连续元素。

流水线的性能

1. 吞吐率：单位时间内流水线完成指令或输出结果的数量。
2. 加速比：程序在流水线上执行与在等功能非流水线上的执行速度之比。
3. 效率：一定时间段内，流水线所有段处于工作状态的比率。

指令流水线的相关和处理

1. 结构相关及处理
    - 结构相关是指流水线中不同指令因操作重叠，引起某些段在同一拍**争用**同一功能部件或同一数据通路的现象，结构相关又称资源相关。
    - 数据通路冲突：采用多总线结构或专用通路结构。
    - 部件使用冲突：重复设置功能部件/轮流使用功能部件。
    - 访存冲突：增设存储器/增设数据存储缓冲器/增设指令预缓冲期/各个段串行访问存储器

2. 数据相关及处理
    - 流水线中不同指令因为操作重叠，引起对统一寄存器或存储单元访问次序的改变，而部分改变会产生冲突现象。
    - 因指令内数据关联产生冲突称为数据相关，将读必须在写操作之后的数据相关称为写后读（**RAW**）相关
    - 处理RAW相关：后推法、重定向法（不用等到写入寄存器后再读，而是直接从产生结果的部件读）、乱序流动法（在等待期间让后续无相关的指令先执行）。

3. 控制相关及处理
    - 无法立即给出分支目标处指令地址、而取值段又要求立即获得后继指令地址导致的冲突。
    - 冻结发（ID发现指令为转移型指令时暂停IF段工作）、延迟转移法（延长转移指令的执行时间，在延长期间可尽量做些有用的指令）、分支预测法（先猜一个，结果出来猜错后就重头开始）

高性能指令流水线

1. 超级流水线（减小拍长）
2. 超标量流水线（同时流入/流出多条标量指令，条数称为**路数**）
3. 超长指令字（VLIW）流水线。（增加流水线同时进行的操作数，流入一条指令流出多个操作结果）

第六章 总线

总线概述

> 总线上连接的部件或设备进行信息传输操作时，能够发起操作的部件或设备称为**主设备**，如CPU；只可响应操作的部件或设备称为**从设备**，如主存、I/O设备等。

总线的分类

按总线信号线功能分类

1. 数据总线（DB），双向传输总线
2. 地址总线（AB），单向传输总线（主设备->从设备）
3. 控制总线（CB），根据信号由主设备发出还是由从设备发出，有控制信号线和状态信号线良好总，都是单向的。

按总线连接部件匪类

1. 片内总线：芯片内部的总线，用于连接芯片内部多个元件，如CPU内部的总线通路。按照信号功能分类，只有DB，没有CB和AB。
2. 系统总线：指计算机内部连接CPU、主存、I/O设备（通过标准化接口）五大部件的总线。又称为板级总线，如ISA总线。通常DB、CB、AB均有。
3. 通信总线：指计算机之间或计算机与其它系统之间通信的总线。如远距离通信的RS-485总线、快速串行通信的USB总线。通常只有DB、CB。

> 由于系统总线连接计算机内各个部件，不同部件的速度相差很大，所以现代计算机往往采用多总线结构，将系统总线分为**CPU总线**和**局部总线**。CPU总线指与CPU直接相连的总线，又称存储器总线或HOST总线；局部总线指不与CPU直接相连的总线，又称I/O总线，如PCI总线。

总线的特性

1. 物理特性：总线上部件在物理连接时表现出的一些特性。

2. 功能特性：每一根信号线的功能。
3. 电气特性：每一根信号线上的信号方向及表示信号有效的电平范围。通常，由主设备发出的信号称为**输出信号**。
4. 时间特性：又称逻辑特性，指在总线操作过程中每根信号线上信号什么时候有效。

总线的性能指标

1. 总线宽度：数据总线的位数。
2. 总线时钟频率：控制同步总线操作时序的基准时钟频率。越快越好。
3. 总线带宽：总线最大数据传输率，单位时间内总线上可传输数据的最大位数。
4. 总线负载能力：总线上保持信号逻辑电平在正常范围内时所能直接连接的部件或设备数量。

总线传输与控制

总线操作

> 总线周期：完成一次总线操作的时间。
> 总线周期中的操作可分为：
>
> 1. 总线申请及仲裁阶段
> 2. 寻址阶段（地址期）
> 3. 数据传送阶段（数据期）
> 4. 结束阶段（数据期）

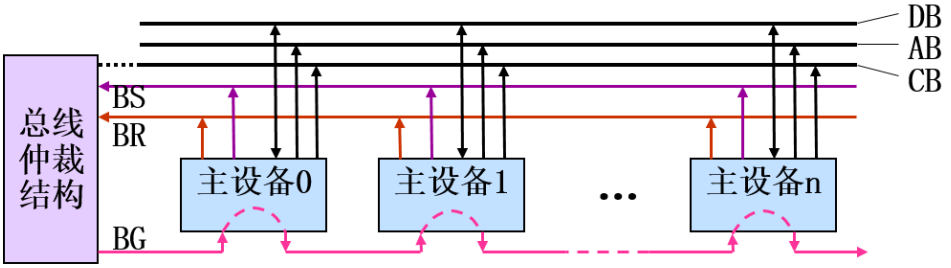
- 隐藏式仲裁：将下次仲裁放在上个总线周期的结束阶段，提高了总线带宽，但需要仲裁机构监视总线状态、决定何时开始仲裁。

总线仲裁

集中式仲裁

> 采用集中式仲裁时，总线上的主设备起码有两根信号线连接到总线仲裁机构：总线请求线RB和总线允许信号线BG。

1. 链式查询方式：添加总线忙信号线BS，菊花链，可扩展性强，易断，距离CPU越近的主设备优先级越高。

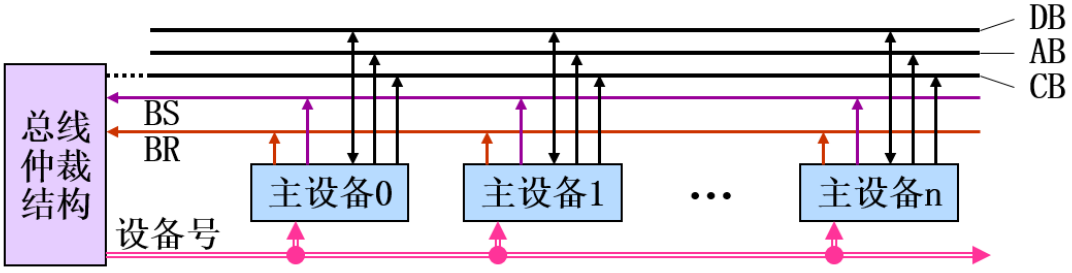


总线请求线BR— $BR=\sum BR_i$，BR=1时有总线请求

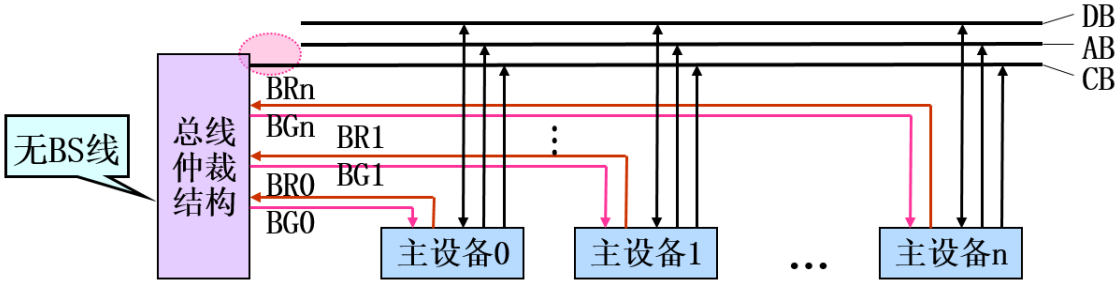
总线允许线BG— $BG_{(i+1)IN}=BG_{iOUT}$

总线忙线BS— $BS=\sum BS_i$，BS=0总线空闲、=1传输周期

2. 计数器定时查询方式：为避免菊花链断掉，采用设备地址信号线代替BG信号线与主设备连接。总线仲裁机构及各主设备均需用$2+log_2n$根信号线，实现的是**循环优先级**。



3. 独立请求方式：各主设备独立地向总线仲裁机构提出总线请求，由总线仲裁机构自身进行仲裁，然后将结果通知到相应的主设备，仲裁电路通常是一个**排队器**。主设备只需要一对BR、BG信号线，但总线仲裁机构需要$2n$根信号线实现仲裁。仲裁速度快、优先次序灵活但信号线数量多、电路复杂。

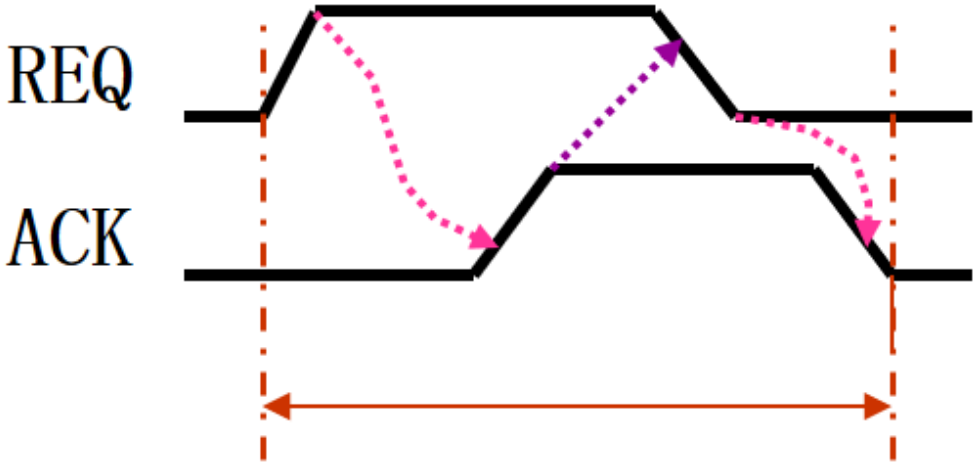


分布式仲裁

1. 自举式仲裁
2. 并行竞争式仲裁

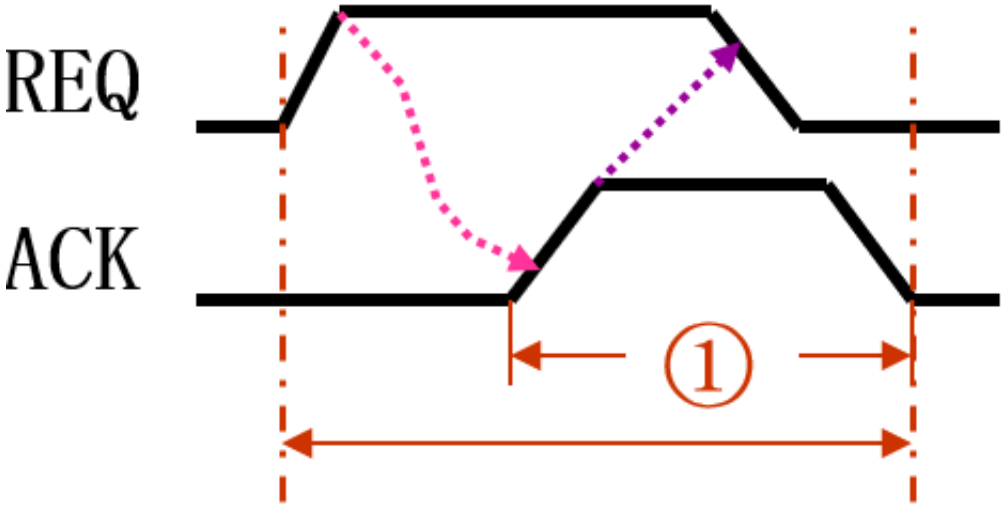
总线定时

1. 同步定时方式：使用同一时钟信号。增加时钟信号线CLK。
2. 异步定时方式：主设备发出请求信号后，一直等待从设备反馈，收到反馈信号后才继续通信。增加请求信号线REQ和应答信号线ACK。
    - 允许主从设备速度不一致。
    - 全互锁方式：主设备及从设备发出信号后，均必须等待对方撤销信号后，才能撤销自己的请求。



(a)全互锁方式

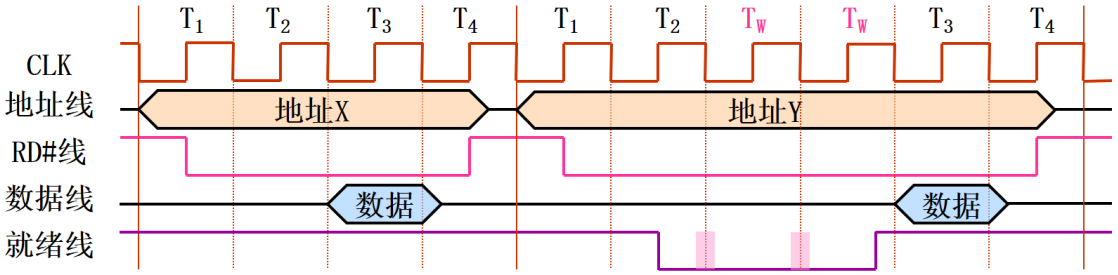
- 半互锁方式：主设备发出信号后必须等从设备反馈才能撤销，从设备不必等待主设备撤销信号，自行一段时间后撤销。①为估计的时间



(b)半互锁方式

- 不互锁方式：两边均估计一段时间后自行撤销。
- **起止式异步串行通信协议**是一个面向字符串送的、无请求/应答信号的、采用不互锁方式的异步通信协议。通信前需要约定**波特率**，即单位时间内传送的二进制信息的位数。

3. 半同步定时方式：采用同步方式定时，通常在时钟上升沿发送信号，在时钟下降沿接收信号。为速度慢的通信方（主/从设备都可能）增设一条等待线$\overline{WAIT}$信号线或者就绪(*Ready*)信号线。



总线标准

1. ISA总线*(Instruction Standard Architecture,*工业标准体系结构*)*
    - 不支持多CPU的总线，总线没有仲裁逻辑。
    - 16位总线，后在此基础上推出了32bit、兼容ISA总线的EISA总线。

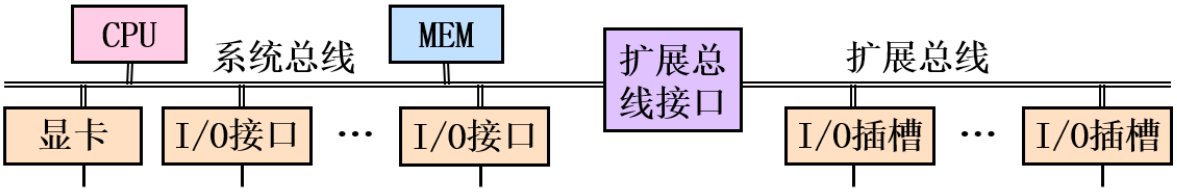
2. PCI总线*(Peripheral Component Interconnect,*外围部件互联*)*
    - Intel、IBM公司联合制定的一种先进的局部总线。
    - 支持多主设备。

总线互联结构

总线互联的结构

单总线结构

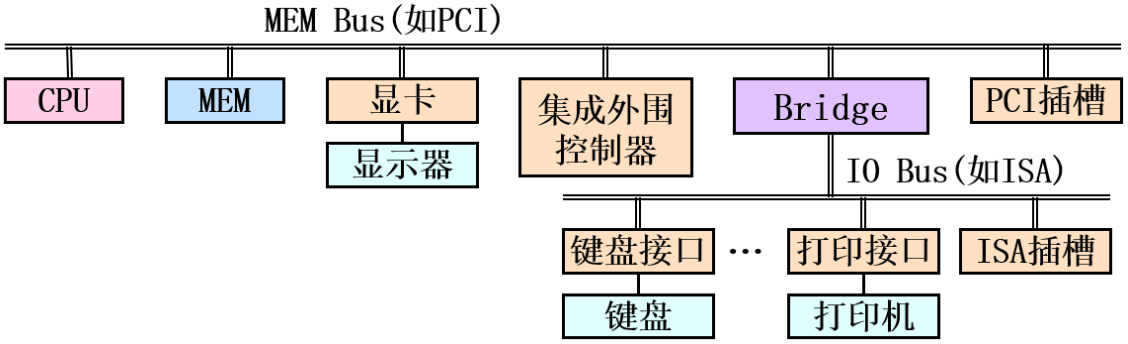
> CPU、主存、I/O设备都连接在同一总线上。



扩展总线是为了增强总线长度及信号稳定性设置的，扩展总线与系统总线同为一个总线标准，因而称为单总线结构。

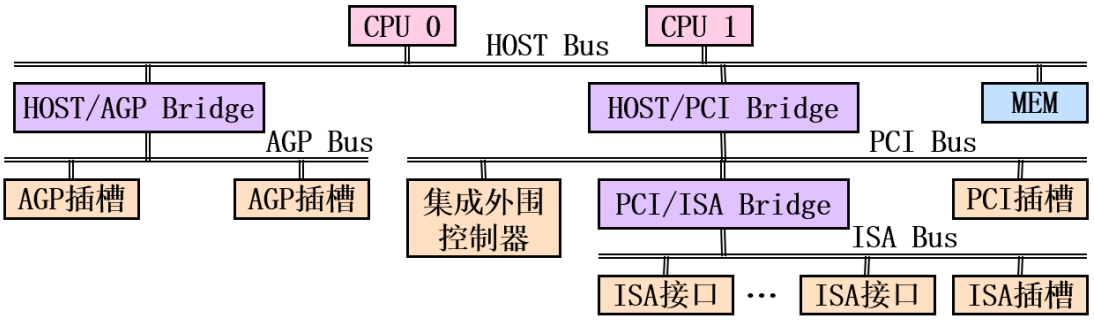
多总线结构

1. 双总线结构



- **总线桥**既可以作为所连总线的总线操作中转机构（转换时钟频率），也可以作为所辖总线的总线仲裁机构。

2. 三总线结构



- 将CPU和主存从双总线结构的存储总线中分离出来，就形成了CPU总线（HOST 总线）、局部总线及I/O总线的三总线结构。
- 不同总线之间还是通过总线桥连接，总线桥通常按照两端的总线标准命名。如HOST/PCI桥（**北桥**）、PCI/ISA桥（**南桥**）。

3. 多总线结构的应用

总线互联的实现

> 习惯上，将I/O接口及总线桥统称为**总线接口单元**，或**总线接口电路**。

总线接口单元应具有如下功能：

1. 侦测总线状态：获得当前总线状态：空闲、地址期、数据期等。
2. 总线侧操作控制：按照当前总线状态，决定是否按总线标准协议发出或响应总线操作，并按要求完成与总线的操作过程。
3. 设备侧操作控制：按照设备操作协议要求，向设备发送操作信息或接收来自设备的信息，并完成与设备的操作过程。
4. 记录设备状态：监视设备工作状态并保存，以便快速完成来自总线的设备状态查询操作。有些状态还需要触发相关动作，如发出总线使用请求信号，或仲裁总线请求后发出总线允许信号。
5. 数据缓冲：利用缓冲期暂存来自总线或设备的数据，解决设备与总线的速度差异问题。
6. 格式转换：串并转换、电平转换、时序转换等。

> 总线接口单元实际上是一个信号及时序转换器。

第七章 输入/输出系统

问题：

1. I/O指令的地址码？命令码的作用？
2. “CPU启动I/O设备后”中启动I/O设备的意思？
3. 程序查询方式中的“控制命令位”和“数据缓冲器”的作用？
4. 周期挪用的适用情况？
5. $WC=0$的溢出信号是什么意思？

I/O系统概述

I/O系统的基本组成

I/O系统的硬件

- 早期：所有I/O设备采用星型结构直接与CPU相连，系统可扩展性差，I/O硬件由I/O设备组成。
- 中期：所有I/O设备通过各自的I/O接口与标准化总线相连接，CPU与存储器也连接到总线上。I/O硬件由I/O设备及I/O接口组成。
- 现代计算机：增设一些I/O管理部件，协助或部分代替CPU实现信息交换。

I/O系统的软件

> I/O软件的主要任务是表示以及实现信息交换。为了提高系统性能，它另一个任务是协调主机与I/O设备的工作。

I/O指令的组成：操作码（输入or输出） + 设备码（哪个设备） + 命令码（操作内容）

- I/O指令与其它指令通过（如算逻运算）操作码区分
- 设备码及命令码为指令格式中的**地址码**
- 命令码通常有数据传送、工作状态测试、操作命令传送等几种类型。

I/O设备与主机的连接方式

I/O设备与主机的连接方式

> I/O设备与主机的连接方式有辐射和总线两种，目前全部采用总线连接方式，这种方式具有可扩展性好、能够实现操作标准化等特点。

- I/O接口以标准化逻辑连接到总线
- 具体设备以非标准化逻辑连接到I/O接口

I/O设备的编址方式

1. 统一编址

   > I/O设备地址与主存单元地址统一编码，不同部件所对应的地址范围不重叠。

   - 统一编址时，指令系统无须设置I/O指令，CPU对I/O设备的访问可借用存储器访问进行。总线上控制信号只需要存储器读$\overline{MEMR}$、存储器写$\overline{MEMW}$两种。
   - 不增加机器指令数，但指令的地址码长，主存空间不易扩展。

2. 独立编址

   > I/O设备地址与主存单元地址均从零开始独立编码，不同部件所对应的地址范围重叠。

   - 独立编址时，指令系统需要设置I/O指令。总线上的控制信号线需要$\overline{MEMR}$、$\overline{MEMW}$、$\overline{IOR}$、$\overline{IOW}$四种。
   - 指令的地址码较短，容易扩展主存空间和I/O设备空间，只需增加两条机器指令，因此目前使用较多的就是独立编址方式。

I/O设备的寻址方式

> I/O设备通过硬件的形式保存了这个设备号，CPU进行I/O操作时，通过I/O指令中的地址码字段指明目标设备，通过BIU将当前指令的操作需求转换成相应的地址信号和控制信号并发送到总线上。

- I/O设备或接口随时监视总线状态。

I/O设备与主机的联络方式

> 联络方式是指I/O与主机间信息传送的定时方式。

1. 异步联络方式：I/O设备与主机之间异步工作，需设置请求（如$\overline{Strobe}$）、应答（如$\overline{Ack}$或$Ready$）联络信号线。

2. 同步联络方式：I/O设备与主机之间同步工作，需设置同步时钟。
3. 立即响应方式：I/O设备随时响应主机操作，无联络信号线，如控制指示灯等简单I/O设备。

I/O设备与主机的传送控制方式

> 传送控制方式是指主机对I/O操作的管理方式，又称信息交换方式，或I/O方式。

1. 程序查询方式：CPU在启动I/O设备后，不断地查询I/O设备是否已做好传送准备，只有在I/O设备准备就绪时，才进行信息传送
    - 需要反映对I/O设备当前操作类型的控制命令位、反映I/O设备当前状态的标志位、存放与I/O设备间所传送数据的数据缓冲器。

2. 程序中断方式：CPU在启动I/O设备后，不主动查询I/O设备状态，而是继续执行现有程序，由I/O设备在就绪后**主动**提出中断请求，再由CPU处理。
    - 需要在程序查询方式接口的功能上，再添加提出、**撤销**请求的功能。
    - 每次响应中断请求都有一定开销，且每次中断服务只能进行一次信息传送（I/O设备速度跟不上，“一次”是相对DMA而言来理解）

3. DMA方式：I/O设备直接与主存间进行信息传送，不需要CPU的干预和中转。
    - 需要CPU让出总线使用权。
    - 在全部数据传送结束时，才由I/O设备提出中断请求，让CPU进行后续处理。
    - 每次DMA传送的I/O设备启动还是需要CPU亲自完成。

4. 通道方式：CPU将部分I/O控制权利下放给称为通道的**专用处理器**，由CPU根据传送需求用通道指令编写通道程序，通道独立地执行通道程序实现信息传送。
5. I/O处理机（IOP）方式：IOP除了能够实现通道的功能外，还能代替CPU实现数据校验、码制转换等与I/O有关的工作。

外部设备

输入设备

1. 键盘：非编码键盘、编码键盘
2. 鼠标：机械式鼠标、光电式鼠标

输出设备

1. 显示器
2. 打印机：点阵针式打印机、激光打印机

存储设备

辅存的性能指标

1. 存储密度：单位面积所能存储的二进制信息量。
    - 道密度：相邻两个道的距离的倒数
    - 位密度：相邻两个二进制位的距离的倒数
    - **存储密度 = 道密度×位密度。**

2. 存储容量：辅存所能存储的二进制信息总量。
    - **存储容量 = 道数×位数/道**，位数/道与位密度并不一一对应。
    - 存储容量有格式化容量和非格式化容量两方面。格式化容量指按某种特定的记录格式存储时的容量，通常只有非格式化容量的70%左右。

3. 寻址时间：读/写头从起始位置移动到读/写位置的全部时间。由寻道时间和道内寻址时间组成。
    - 通常用平均寻址时间表示$(max + min)/2$

4. 数据传输率：单位时间内辅存能向主机传送数据的位数或字节数。
    - 记录密度D_b（即位数/道），盘面转速V（即转速/秒），$D_r \approx D_b \times V$。

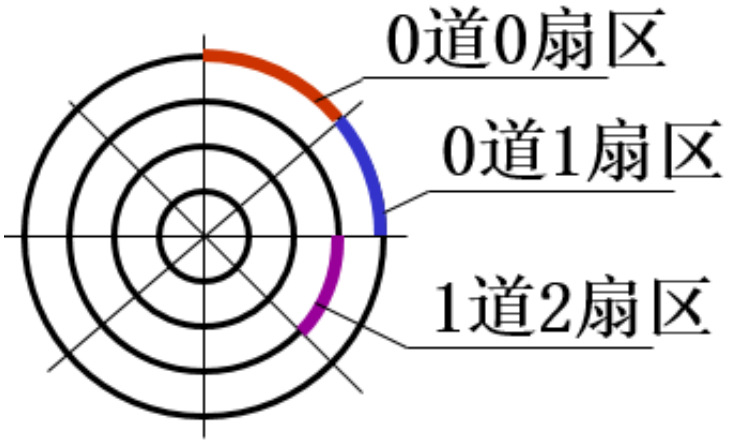
5. 误码率：从辅存读出信息时出错的频率。

磁盘存储器

1. 磁记录原理
2. 磁记录方式
3. 磁盘的信息分布

> 磁盘盘片由若干各磁道组成，每个磁道可以存储若干个字节的数据。每次访问磁盘时均为一批数据，称为**记录块**，每个磁道可存放若干个记录块。根据磁道内**记录块长度是否可变**，磁道记录格式有定长和不定长两种格式。

1. 定长记录格式：将盘面的所有磁道划分成若干个大小相同的扇区。



    - 每个扇区的扇角和可存放信息量均相同
    - 扇区是磁盘读写的基本单位。
    - 磁盘地址：台号+磁道号+盘面号+扇区号

        - 解题时一定要注意**每个盘片是双面的，即有两个盘面，如道密度（道的个数/cm）是双面的，算每个盘面的道数时别忘了除2。**

    2. 不定长记录格式：根据需要来确定记录块的长度。

4. 硬盘存储器组成
    1. 磁盘盘片
    2. 磁盘驱动器
    3. 磁盘控制器

磁盘阵列

> 通常将多个磁盘组织成阵列，在增加存储容量的同时，提高数据传输率及I/O请求响应速度，同时设置冗余信息盘来提高可靠性，这种磁盘组织方式被称为**廉价磁盘冗余阵列*Reundant Array of Inexpensive Disks,RAID***，常简称为**磁盘阵列**。

1. RAID0：无冗余磁盘阵列，并行工作。
2. RAID1：镜像磁盘阵列。
3. RAID2~RAID6等。

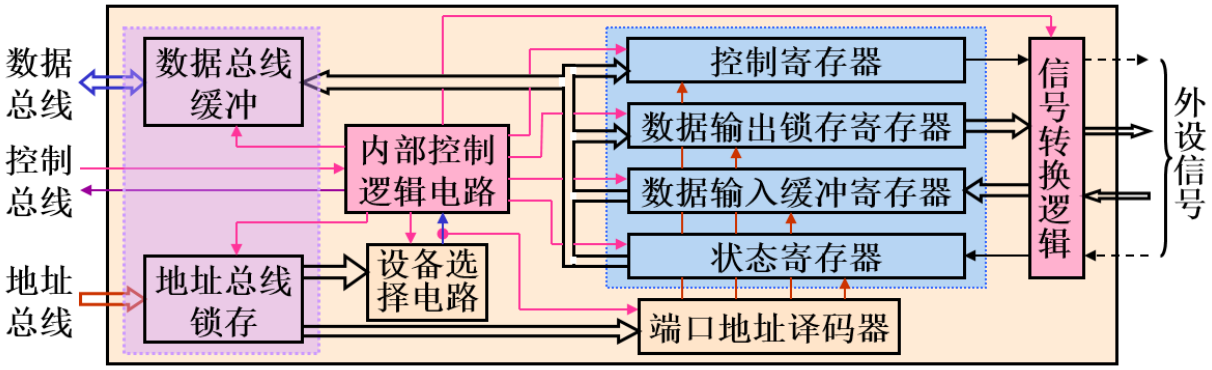
I/O接口

I/O接口功能

1. 设备寻址：总线要找的设备是不是我的。
2. 数据缓冲：用寄存器暂存来自总线或I/O设备的数据（总线和I/O设备速度不同）。
3. 操作中转：暂存来自总线的命令，根据设备的状态适时转发。
4. 信号转换：串-并格式、电平及时序等转换。
5. 设备状态监视：监视并保存状态到寄存器中，供CPU查询或主动向CPU报告，以支持I/O传送方式（如程序查询、DMA、中断）的实现。

    - **端口**是指**接口硬件中**那些可直接与总线交换信息的**寄存器。**如存放数据信息、控制信息以及状态信息的寄存器分别称为数据端口、控制端口及状态端口。

I/O接口的硬件组成



I/O接口的软件组成

- I/O接口与总线间的信息交换部件是I/O端口，每个设备有多个I/O端口。
- I/O指令格式：操作码+I/O端口地址+CPU中寄存器地址
- I/O端口地址：设备码+端口序号。

程序查询方式

程序查询方式的流程

> CPU不断地查询I/O设备的状态，只有在I/O设备准备就绪时才进行数据传送

- 查询程序至少包含四条指令（两条I/O指令、两条非I/O指令）：读I/O接口状态的I/O指令、测试数据中某位值的比较指令、条件转移指令、读或写I/O接口数据口的I/O指令。

程序查询方式的接口组织

- 完成触发器RD、工作触发器BS。
  - $RD=0,BS=0$，表示I/O设备未初始化（暂停）
  - $RD=1,BS=0$，表示I/O设备准备就绪。
  - $RD=0,BS=1$，表示I/O设备正在工作。
  - 通过触发器的Set和Reset端口实现。

无条件传送方式

> 无条件传送方式中，CPU**随时**与I/O设备交换数据，无须查询I/O设备的状态。只用于CPU与简单外设的传送（如信号灯、读取开关状态等）。

程序中断方式

> 当用中断方法实现I/O传送控制时，这种I/O传送控制方式称为程序中断I/O方式，简称程序中断方式。
>
> - 中断请求：表示存在异常情况或突发事件急需处理，通常用信号线有效表示。
> - 中断源：能产生中断请求的部件（如I/O接口）
> - 中断服务程序：处理某个中断请求的程序（不惟一）。
> - 中断响应：CPU从当前程序转入中断服务程序的过程。
> - 中断返回：CPU从中断服务程序返回当前程序的过程。
> - 中断处理：中断响应、中断服务和中断返回的综合。

中断的分类

按中断请求分

> - 内中断：来自CPU内部的中断，异常。
> - 外中断：来自CPU外部的中断，硬件中断。

- 可屏蔽中断/不可屏蔽中断（中断允许标志位IF，0屏蔽，1允许）。
- 可屏蔽中断在中断返回后继续执行断点的下条指令。
- 不可屏蔽中断在中断返回后**总是执行一条新指令**。

按中断源识别及中断服务程序入口地址获得方法分

- 识别中断源：取最紧急的请求处理。
- 向量中断：CPU响应中断时，中断机构用**硬件方法**识别中断源、获得中断服务程序入口地址（**中断向量**），存放所有中断向量的表格称为**中断向量表（*IVT*）**（基本放在主存中），中断向量在IVT中的行号称为**中断向量地址**。
- 非向量中断：CPU响应中断时，采用**软件查询方法**识别中断源和获得中断服务程序入口地址。所有的请求共用一个**公共的中断服务程序**，程序入口地址放在寄存器中。

按中断处理能否重叠分类

1. 单重中断：中断处理过程中不再响应新的I/O中断请求（即使更紧急）。
2. 多重中断：中断嵌套。
   利用“中断允许”位IF表示，0为单重，1为多重，由于IF位当前状态必须对中断服务程序可见，故一般都设置在CPU的状态寄存器中。

I/O中断的过程

1. 中断响应
    1. 识别中断源。

2. 保存现场。
    - 硬件现场：PC、PSW
    - PC可保存断点的指令地址，也可以用刚取得的中断源类型确定返回点的指令地址后再保存。
    - I/O请求得到响应的条件：I/O中断请求信号有效、当前指令结束（End有效）、“中断允许”标志位$IF=1$、无DMA请求或更紧急的中断请求。

3. 获得中断服务程序入口。
4. 转入中断服务阶段。

2. 中断服务
3. 中断返回
    1. 恢复现场
    2. 返回当前程序

例—某计算机的CPU主频为50MHz，机器指令平均CPI为5，中断响应需6个主时钟周期，某外设最大数据传输率为20KB/s，该外设接口中的数据缓冲器为16位，相应的中断服务程序包含10条机器指令。请回答下列问题：

（1）该外设是否可用中断方式I/O？若能，在该设备持续工作期间，CPU用于该设备I/O时间占整个CPU时间的百分比为多少？

（2）若该外设最大数据传输率为2MB/s，则是否可用中断方式进行I/O？

解—(1)中断方式I/O时，中断请求间隔为2B/20KBps=100us，

每次中断时间为(6＋10*5)/(50*10^6)=1.12us，

∴可采用中断方式I/O；I/O时间百分比为1.12%

(2)速率为2MBps时，中断请求间隔为2B/2MBps=1us，

1.12us＞1us，∴不可采用中断方式I/O

I/O中断的组织

I/O接口的组织

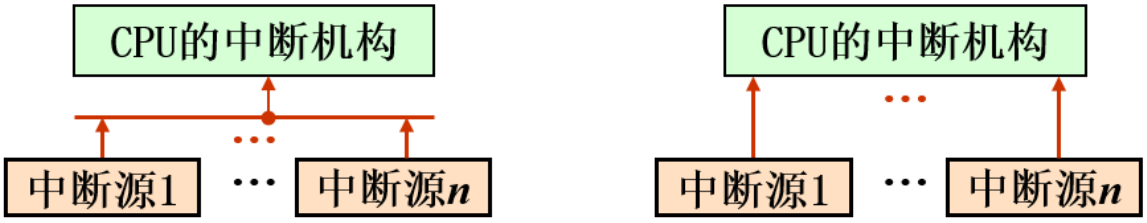
- 中断请求触发器$INTR$。
- 中断允许触发器EI。
- 完成触发器RD、工作触发器BS。
- 中断响应信号$\overline{INTA}$。
- 中断向量地址是约定好且惟一的。

$EI=1, RD=1$时$INTR \leftarrow 1$。

I/O接口在提供中断向量地址时撤销中断请求。

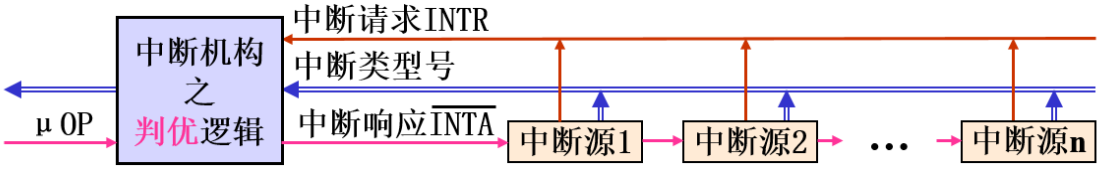
中断源识别的组织

两种中断请求的连接方法：共用式和独立式

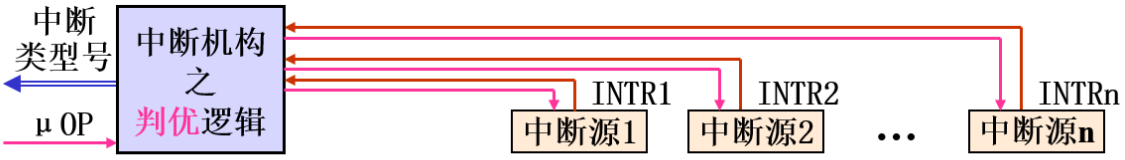


三种判定最紧急的方式：

1. 软件查询方式：适用于两种连接方式，在执行中断服务程序时按序查询中断请求标志，**查询顺序决定了优先级**。无需设置中断向量地址相关机构以及$\overline{INTA}$引脚。
2. 串行判优方式：适用于共用式，各中断源的连接顺序决定了优先级。



3. 并行判优：适用于独立请求连接方式，按请求引脚的次序来判定优先级并形成中断向量地址，可实现动态优先级。

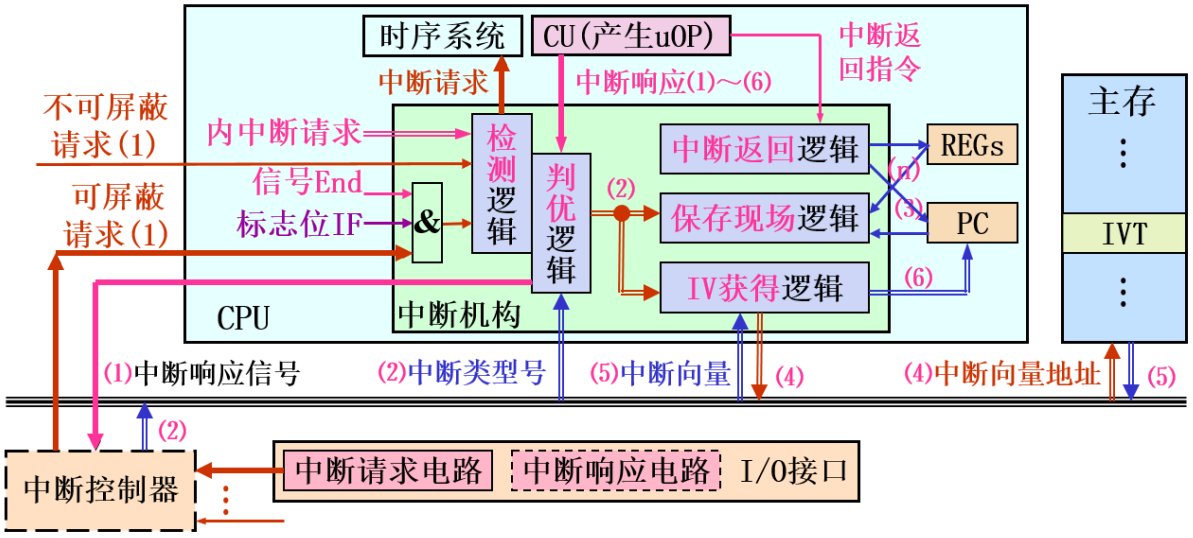


判优原理—算法控制，判优算法决定各请求优先级(动态)

中断控制器的组织

> 基本功能：
>
> - 自动检测并记录引脚的中断请求，能自动向CPU提出中断请求。
> - 能够处理来自CPU的中断响应操作
> - 能够用作常规I/O接口，接受并处理CPU的操作，如修改优先级等。

中断系统组成



多重中断的组织

1. 多重中断环境的组织
    - 在中断服务程序中把IF置1，表示还可以接收中断请求。

2. 新I/O中断请求检测的组织
    - 将I/O中断请求分为正在服务和尚未服务两大类。
    - 只有当后产生的尚未服务请求的优先级高于正在服务请求时，才会认为有新I/O中断请求产生。所以对两类中断请求分别排队，将排队后的最高级别的中断请求进行比较，再判断是否有新I/O中断请求。

3. 多重中断现场保存的组织
    - 用CPU内部的寄存器堆栈保存中断现场，而中断返回指令$IRET$则用出栈操作实现现场恢复。

中断屏蔽的组织

- 改变优先级。
- 屏蔽寄存器和屏蔽字$MASK$。

DMA方式

1. CPU需要**在主存中**开辟好缓冲区，CPU还负责缓冲区数据的准备工作或结束工作。
2. 数据传送时需要CPU让出总线控制权。

DMA的传送方式

1. 暂停CPU访问方式
    - 一批数据传送开始时DMA接口提出请求，CPU让出总线控制权，全部传送完成后才交回总线控制权。
    - 在I/O设备准备数据期间，DMA接口占用总线却不传送数据。

2. **周期挪用（窃取）方式**
    - 与暂停CPU访问方式基本相同，在I/O设备撤销DMA请求（去准备数据）时，DMA接口立即还回总线使用权。
    - 由于每次请求/应答只传送一个字，传送周期通常为多个主存周期，故适用于I/O设备读/写周期大于主存周期的情况。

3. 与CPU交替访问方式
    - 约定总线使用权轮流分配给CPU和DMA接口，DMA接口可直接使用总线而不申请。

DMA接口的基本组成

1. 主存地址计数器（MAC）：用于存放主存中需交换数据的地址。
2. 传送字数计数器（WC）：用于存放已传送数据的总字数，或尚需传送数据的总字数。
3. DMA请求逻辑：当数据准备就绪的情况下产生DMA请求$DREQ$。$DACK$再次启动I/O设备。
4. DMA控制逻辑：由控制电路、时序电路等组成，负责管理DMA传送过程。

DMA的数据传送过程

1. 预处理：CPU提前通知DMA接口本次传送的参数（缓冲区首地址、传送地址、传送方向和传送方式），并启动I/O设备。
2. 数据传送
    1. I/O设备准备就绪时，将数据传送到数据缓冲寄存器BR中，出发I/O接口部分的状态寄存器改变，产生DMA请求$DREQ$。
    2. DMA控制逻辑在$DREQ$有效时，向CPU发出请求总线使用权信号HRQ。
    3. CPU在当前访存周期结束时，向DMA接口发回允许拥有总线控制权信号$HLDA$。
    4. DMA接口通过总线实现写主存操作，$AB \leftarrow (MAC)$、$CB \leftarrow MEMW$、$DB \leftarrow (BR)$。
    5. DMA控制逻辑通知I/O接口部分上个字传送结束（$DACK$），准备传送下个字
    6. DMA控制逻辑触发MAC和WC修改主存地址及传送字数。
    7. 若传送字数计数器$WC \neq 0$，DMA控制逻辑在$DREQ$无效时撤销HRQ（向CPU交回总线控制权），在$DREQ$有效时继续传送（到2）。若为0，则表示数据传送已经结束，DMA控制逻辑撤销HRQ。
    8. I/O接口部分收到DMA传送结束信号（$WC = 0$的溢出信号）时，向CPU提出程序中断请求，通知CPU数据传送已经结束。

3. 后处理：校验送入主存的数据是否正确、是否继续采用DMA方式传送等。

> CPU负责预处理和后处理，DMA接口负责数据传送。

- 传送最大速度，中断方式为1字/指令周期，DMA方式为1字/总线周期。

- 中断方式的请求在一条指令完成时响应，DMA方式的请求在一个存取周期结束时响应。
- DMA请求的优先级高于中断请求。
- 中断方式可以处理传送异常，DMA不行。
- 中断方式靠程序传送、CPU每干预一次传送一个字，DMA方式靠硬件传送、CPU每干预一次传送一批数据。

DMA的组织

1. DMA接口的组织
2. DMA请求判优的组织
    - 共用请求式和独立请求式两种连接方法。
    - CPU仅发出DMA响应信号，无需获得任何信息，独立请求式的DMA接口都必须设置DMA响应信号。
    - 各DMA接口通常采用独立请求式连接到“DMA控制器”，由它再共用请求方式连接到CPU。与中断控制器类似，“DMA控制器”是一个并行判优逻辑部件及DMA请求部件。

3. 增强型DMA接口
    1. 选择性DMA接口：在物理上可以连接多个I/O设备，逻辑上只允许连接一个I/O设备（每次DMA传送只为一个I/O设备服务）。
    2. 多路DMA接口：物理上连接多个I/O设备，逻辑上也允许多个I/O设备。DMQA接口采用交叉方式为各个I/O设备传送数据。多路型DMA接口需要为每个所连I/O设备设置一套存放DMA传送参数的寄存器。

通道方式