第一章：

计算机软硬件 概念及其各自组成：

硬件：指计算机的实体部分（由看得见的各种电子元器件各类光 电 机设备的实物组成 如主机外设等）

软件：由人们事先编制的具有各类特殊功能的程序组成。常把这些程序寄寓于各类媒体（如ROM RAM 磁带 磁盘 光盘 纸带等）常存放在计算机的主存或辅存中

计算机的软件可分为 系统软件（管理整个计算机系统，监视服务，使系统资源合理调度，高效运行。包括 标准程序库、语言处理库、操作系统、数据库管理系统、网络软件等）与应用软件（应用程序 用户根据任务需要编制的各种程序）两大类

冯诺依曼计算机特点：

1.计算机由运算器、存储器、控制器、输入设备、输出设备五大部分构成

2.指令和数据以同等地位存放于存储器内，并可按地址寻访

3.指令和数据均已二进制数表示

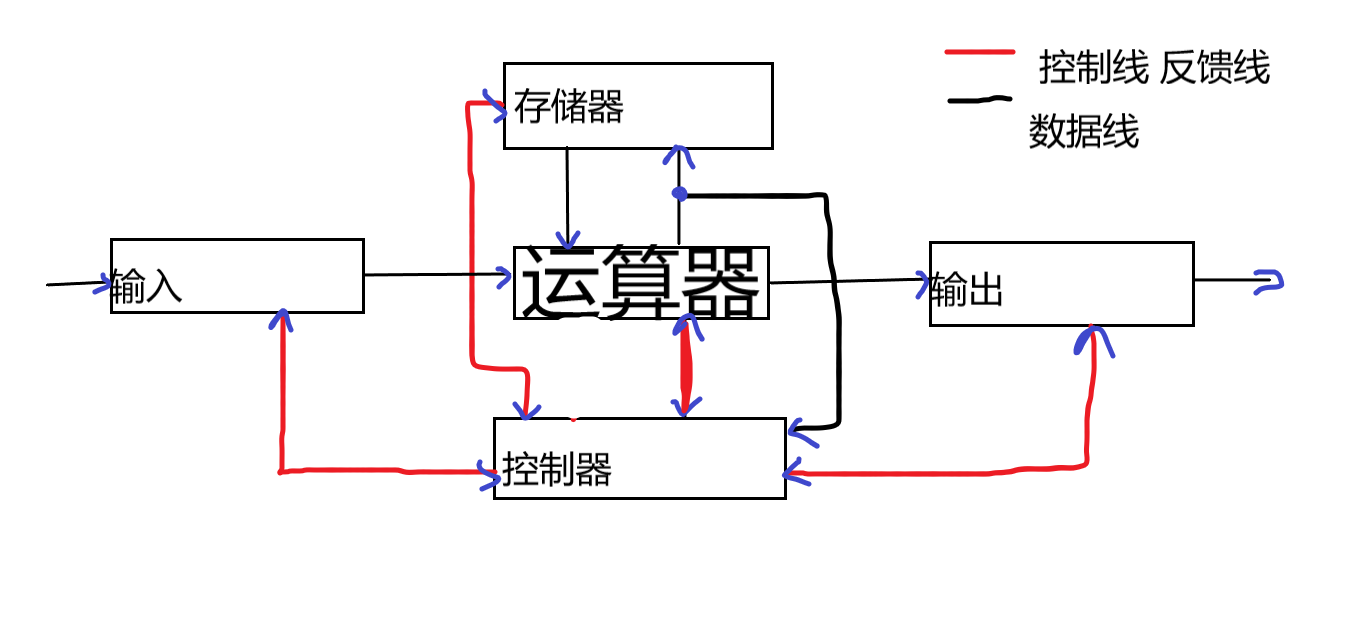
4.指令由操作码和地址码组成，操作码用来表示操作的性质，地址码表示操作数在存储器中的位置

5.指令在存储器中按顺序存放。通常指令是顺序执行的，在特定条件下，可根据运算结果或根据设定的条件改变执行顺序

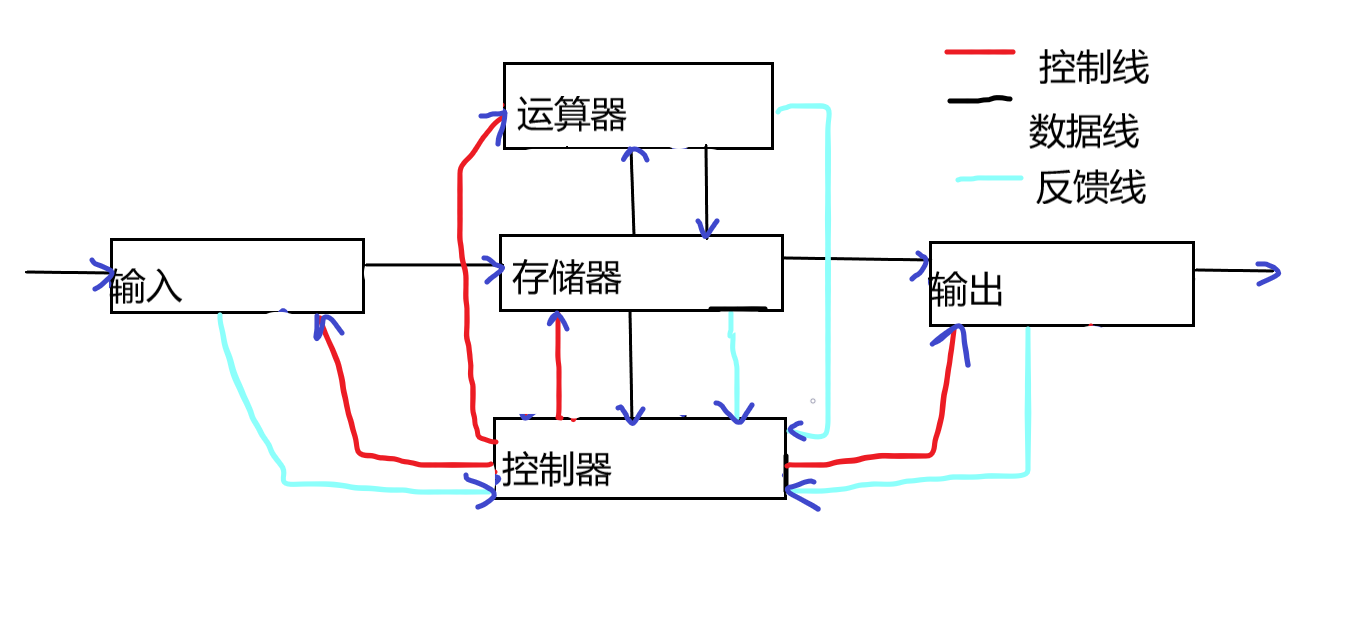
6.机器以运算器为中心，输入输出设备与存储器之间的数据传送通过运算器完成。

注：现代计算器以存储器为中心

结构框图：



（冯诺依曼）



（现代计算机结构）

区分指令和数的方法：

1. 访问内存的时间：取指周期取出的是指令 执行周期 取出的是操作数
2. 地址来源 ：地址寄存器 有效地址

计算机硬件的主要技术指标：

机器字长： 字长越长书的表示范围越大，精度越高。（机器字长数值上等于总线位数或ALU相加使得最长位数）

存储容量：包括主存容量与辅存容量 存储容量=存储单元个数\*存储字长

（一个字节是八位二进制数 一个字可能包含多个字节 B表示一个字节 例 2M = 2^21位 = 2^18字节（KB或B））

运算速度：与很多因素有关如主频 执行什么操作 主存本身速度（主存速度快 取指 取数就快）

吉普森法衡量运算速度（f为第i指令占全部操作的百分比 t为第i条指令时间）

现在机器的运算速度普遍采用单位时间内执行的指令的平均条数衡量

单位：MIPS（百万条指令每秒 Million Instruction Per Second）

或 CPI（Cycle Per Instruction执行一条指令所需的时钟周期 机器主频的倒数）衡量

或 FLOPS（Floating Point Operation Per Second 浮点运算次数每秒）衡量

第四章：存储器

三个性能指标：速度 容量 每位价格（位价） 一般 速度越大 价位越高 容量越大价位越低 （容量越大 速度必越低

主存层次结构：

1. 缓存-主存层次 （主要解决CPU与主存速度不匹配的问题 CPU中的寄存器速度最快 catch缓存可调节主存速度过慢的差距） 这一层次的速度接近于缓存高于主存
2. 主存-辅存层次

Ps：CPU与主存进行信息交换时 先在catch’中寻找找不到才去主存中寻找（cache的命中率极高 95%以上 虽然不知道为什么）

主存的基本组成：

数据总线

MDR

读写电路

存储体

驱动器

控制电路

译码器

读 写

MAR

地址总线

主存中存储单元的地址分配：

（主存各存储单元的空间位置是由单元地址号来表示的，而地址总线是用来指出存储单元地址号的根据地址可读出一个存储字 八位二进制表示一个字节存储字长都是8的整数倍 通常计算机可按字寻址也可按字节寻址）

小端存储： 高字节存放在高地址 最低地址->最低字节

大端存储：高字节存放在低地址 最低地址->最高字节

对齐方式？？？

主存的主要技术指标： 存储容量 存储速度

存储容量 可用二进制代码位数表示 也可用存储的字节总数表示

存储速度： 由存取时间和存取周期表示

存取时间：存储器的访问时间 启动一次存储器操作到完成操作所需的全部时间

存取周期：存储器进行连续两次独立的存储器操作所需的最小间隔

通常存取周期大于存取时间

存储器的带宽： 单位时间内存储器存取的信息量 单位：字/秒 字节/秒 位/秒

提高带宽可采取的措施：

1. 缩短存取周期
2. 增加存储字长（每个存取周期可读/写更多的二进制位数
3. 增加存储体

存储芯片：

通过地址总线 数据总线 控制总线与外部连接

地址线：单向 控制线： 分为读写控制线与片选线两种 数据线：双向

译码驱动方式：

1. 线选法：用一根字选择线选择存储器的各位 只适用于容量较小的
2. 重合法：两个方向均有译码器 组合起来选择地址

RAM按存储信息原理可分为 静态RAM（SRAM） 动态RAM（DRAM）

静态RAM 信息读出后仍保持原状态不需要再生 掉电丢失

DRAM：、P87

刷新： 先将原存信息读出再有刷新放大器形成原信息并重新写入的再生过程

由于存储器随机存储有些信息可能很长时间访问不到得不到刷新 因此采用定时刷新的方法 一般取2ms（刷新周期

1. 集中刷新： 在一个刷新周期内 对全部存储单元集中一段时间逐步进行刷新 此时必须停止读写操作（这段时间被称为访存“死区”
2. 分散刷新： 对每行存储单元的刷新分散到每个存取周期内完成 一个存取周期一半用来读写或维持信息一半用来刷新，这样不存在“死”时间但存取周期变长 系统速度降低
3. 异步刷新： 结合集中于分散 即缩短“死”时间又最大利用刷新间隔

每隔一段时间刷新一行，刷新一行只暂停一个存取周期 “死”时间缩短

如 128\*128 2ms间隔 0.5微秒存取周期 集中刷新“死”时间：128\*0.5 = 64微秒

异步刷新 对于每行2ms刷新一次 “死”时间 0.5微秒

Ps：若DRAM刷新安排在CPU对指令的译码阶段，这个阶段不访存这样即克服分散刷新独占0.5微秒使存取周期加长缺点又不会出现集中刷新的放存“死区“问题 从根本提高了效率

DRAM与SRAM比较

DRAM应用广泛

DRAM优点：

1. 集成度比SRAM高
2. 行列按先后顺序输送减少了芯片引脚 封装尺寸减小
3. 功耗小于SRAM
4. 价格低

缺点：

1. 使用动态元件（电容）速度低于SRAM
2. DRAM需要再生，再生电路也需要消耗功率

通常容量不大的高速缓冲存储器大多用SRAM实现

存储器与CPU连接：

1. 地址线的连接
2. 数据线连接
3. 读写命令线连接
4. 片选线的l连接

位扩展与字扩展：

位扩展： 如2片1K\*4位可组成1K\*8位存储器 （只有数据线不在一起，其他都在一起）

字扩展： 如2篇1K\*8位可促成2K\*8位存储器

一般可以位扩展不选择字扩展 因为字扩展片选线不一致需要二次译码

连接时要注意片与片之间地址线 数据线 和 控制线的连接

地址线连接：

存储芯片容量不同地址线也不同，通常CPU地址线多余存储芯片 常将CPU低位与存储芯片地址线相连 CPU地址线高位作片选等功能

数据线的连接：

CPU数据线与存储芯片位数不同时 必须对存储芯片位扩展

读写命令线连接：

一般可直接连接 通常高电平度低电平写 有些CPU读写命令线是分开的

片选线的连接：

片选有效信号与CPU访存控制信号(MREQ非 （低电平有效）)有关

使用译码器进行片选信号的选择 片选选的是存储芯片

提高访存速度的措施：

1.调整主存结构：低位交叉、高位交叉

2.采用层次结构Cache-主存

3.采用高速器件：如高性能存储芯片（SDRAM（步DRAM）、DRAM、带cache的DRAM（CDRAM））

主存结构：

1. 单体多字系统：程序和数据在存储体内是连续存放的 如一个存储体内存储四条指令那么读一次就会读出四条指令，提高了带宽4倍（单位时间内存储器存取的信息量），但是主存内必须是连续存放的，一旦遇到转移指令或者操作数不能连续存放这种方法就无法提高速度了。
2. 多体并行系统：采用多体模块组成的存储器。每个模块有相同的容量和存取速度，各模块各自有独立的地址寄存器（MAR）数据寄存器（MDR）地址译码、驱动电路、读写电路他们可并行工作也可交叉工作。

两种编址方式：

1. 高位交叉编址（顺序存储）：使不同请求源访问不同体即可实现并行 （高位地址表示体号 低位地址为体内地址）

此方式由于一个体内的地址是连续的有利于存储器的扩充

1. 低位交叉编制（交叉存储/模M编址 M为模块数）：由于程序连续存放在相邻体中故低位地址表示体号 高位地址为体内地址 （此方法可在不改变没个模块存取周期的前提下，提高存储器的带宽。 除第一个存取周期存储器可在一个存取周期内提供M个存储字） 课本P106 例4.6

排队器 P106

高速缓冲存储器：

程序访问的局部性原理：指令和数据在主存的地址分布不是随机的，而是相对的簇聚，使得CPU在执行程序时访存具有相对的局部性。 （->CPU从主存存取指令或取数据，在一定时间内，只是对主存局部地址的访问）

Cache工作原理： 主存n位被分为两部分（高m位表示块地址低b位表示块内地址）Cache也被分为两部分（高c位表示块地址低b位表示块内地址 2^m>>2^c），任何时间都有一些主存块在缓存块中，CPU读取某字时有两种可能一种是在Cache内则直接访问Cache（CPU与Cache之间一次传送一个字），一种是不再Cache中此时则访问主存取出字并将主存对应的整个子块一次全部调入Cache中（Cache与主存之间块传送）。若主存块已调入缓存块，成该主存块与缓存块建立了联系。

（Cache块何便是对应的哪个主存块）

字在Cache中称为Cache命中不在称为不命中，由于缓存块数远小于主存块数因此需要一个标记来表示当前存放的是哪一个主存块。CPU读信息时需将高m位（主存块地址）与

Cache标记作比较判断在不在Cache中

命中率 h = Cache命中总数/（Cache命中总数+访存次数）

Cache-主存系统平均访存时间 t = h\*命中时Cache访问时间 + （1-h）\* 没命中时主存访问时间

访问效率 e = 命中时Cache访问时间/Cache-主存系统平均访存时间 \* 100%

Ps： 一般而言，Cache容量越大命中率越高。但不需要太大一是成本问题 二是Cache容量达到一定值时命中率随容量增加提高不显著。Cache容量是命中率与总成本价的折中值

e.g: 80386主存最大容量4GB 但配套Cache 16KB或32KB 命中率达95%以上。

块长与Cache容量相同 不好确定：一般每块取4至8个可编址单元（字或字节）较好，也可取一个主存存取周期能调出主存的信息长度。

Cache基本结构：

主要由Cache存储体、地址映射变换机构、Cache替换机构几大模块组成。

Cache存储体：以块为单位与主存交换信息，为加速Cache与主存之间的调动，主存大多采用多体结构，且Cache访问优先级最高

地址映射变换机构：将CPU送来的主存地址转换为Cache地址。（主存与Cache的块大小相同 块内地址都是相对于快的起始地址的偏移量，即地位地址相同，因此地址变换主要是主存的块号与Cache块号之间的转换

替换机构：（要将新的主存块放入Cache内是Cache已满 则需要采用替换策略）

Ps：Cache纯硬件实现 用户看不见主存地址与它的转换

Cache读操作：

CPU发出主存地址后判断是否在Cache中，命中则直接访问Cache并发送至CPU，不命中访问主存，将字送入CPU同时将字所在块放入Cache中，Cache满替换腾出空位调入新主存块，Cache有空位直接调入。

Cache写操作：写回法、写直达法

1. 写直达法（存直达法）：写操作是数据即写入Cache又写入主存，随时保证Cache与主存数据一致

缺点：增加访存次数

时间是访问主存时间

1. 写回法（拷回法）：写操作时仅把数据写入Cache当Cache数据被替换出去时才写入主存。

Cache可能会与主存数据不一致

为了识别数据是否一致，在Cache中增设标志位，两个状态：“清”（未修改过 一致）与“浊”（修改过 不一致）替换时若标志位为浊则应该将Cache写回主存同时将标志位改为“清“。

时间较快 但增加Cache复杂性

Cache——主存的地址映射

1. 直接映射：每个主存块仅与一个Cache对应，用主存的某几位直接判断

映射关系：I = j mod C 或 I = j mod 2^C （i表示缓存块号 j表示主存块号 C为缓存块数）

主存分为三部分 t+c+b （t位表示主存标记 c位表示对应Cache子块地址（m = t + c） b位表示字块地址） 主存块数 2^m

比较时 根据中间c位找到Cache对应块 将Cache对应的标记与主存高t位对比 相同并且Cache有效位为“1“说明命中，否则不命中，前往主存读取数据 并将块写入Cache此时若有效位为”0“则改为”1“

Ps：Cache内数据可能无效 如初始时Cache数据应为空 其中的内容无意义

缺点：肯回造成Cache大面积不使用 块冲突率高 命中率低

主存可投影到第i块上的有 i、C+i、……、2^m-C+i

1. 全相联映射：允许主存中每一字块映射到Cache的任意位置 冲突率降低 命中率提高

主存分为 两部分 m+b位 m（t+c）为主存字块标记 b字块内地址

每次寻找时 需要与Cache的每一块的标记做比较，直到命中或全部访问结束不命中

缺点：所需逻辑电路多 成本较高 实际Cache需采用各种措施减少地址比较次数

1. 组相联映射：直接映射与全相联映射的折中

把Cache分成Q组 组内进行全相联 各组之间直接映射 每个主存块只对应某个组

I = j mod Q（i缓存组好 j主存组号 Q组数）

主存被分为三部分 s+q+b （s = t + r 主存字块标记 q = c-r 组地址 b 字块内地址） 2^q 表示Cache内的组数 2^r 表示组内包含的块数

访问Cache时 现根据中间 q位找到组号 再用高s位与组内所有块的标记比较若命中则访问Cache不命中读取主存 并将 块替换到对应组

替换策略：

1. 先进先出算法（FIFO： 不改变命中率
2. 近期最少使用算法（LRU： 平均命中率最高
3. 随机法 不改变命中率

第五章：输入输出系统

I/O指令是机器指令（CPU指令）的一类

I/O指令的一般格式： 操作码+命令码+设备码

（操作码可作为I/O指令与其他指令（如访存指令、算逻指令、控制指令等）的判别代码；命令码体现I/O设备的具体操作；设备码是多台I/O设备的选择码

通道指令：对具有通道的I/O系统专门设置的指令，这类指令一般用一指明参与传送（写入或读取）的数据组在主存中的首地址；指明需要传送的字节数或所传送的数据组的末地址；指明所选设备的设备码及完成某种操作的命令码。 (这类指令的位数一般较长)

Ps：通道指令又称通道控制字（channel control word，CCW） 通道指令是通道自身的指令用来执行I/O操作，在具有通道结构的计算机中，I/O指令不是先I/O数据传送，主要完成启动/停止I/O设备，查询通道和I/O设备的状态及控制通道所做的其他操作，具有通道指令的计算机，一旦CPU执行了启动I/O设备的指令，就有通道来代替CPU对I/O设备的管理。

I/O设备码编址方式：（通常将I/O设备码看作地址码）

1. 统一编制：将I/O地址看作是存储地址的一部分
2. 独立编址：I/O地址与存储器地址是分开的，所有对I/O设备的访问都必须有专用的I/O指令

Ps：统一编址占用了存储空间，减少了主存容量，但无需专用的I/O指令。不统一编址由于不占用主存空间，故不影响主存容量，但需设I/O专用指令。

RISC指令集 统一编址 CISC指令集 独立编址

设备寻址：每台设备都有一个设备号，当要启动时有I/O指令的设备码直接指出改设备的设备号。通过接口电路中的设备选择电路，便可选中要交换信息的设备。（地址译码电路）

传送方式：（网络时一位一位传输（串行）

1. 并行：同一瞬间 n位信息同时从CPU输送至I/O设备，或由I/O设备输入到CPU

特点：出书速度快 但数据线要求多

1. 串行：同一瞬间只传送一位信息，在不同时刻连续逐位传送一串信息

特点：传送速度慢 但只需要一根数据线一根地线

常用作远距离传输 如网络

联络方式：

1. 立即响应方式：（常用于惰性器件） 工作速度比较慢，当他们与CPU发生联系时通常都已处于某种等待状态，因此只要CPU的I/O指令一到就立即响应 （如：指示灯的亮灭、开关的通断、A/D转换器缓变信号的输入等。）
2. 异步工作采用应答信号联络：I/O设备与主机工作速度不匹配时，采用此方法。

此方式交换信息前，I/O设备与CPU各自完成自身的任务，一旦出现联络信号彼此才准备交换信息

1. 同步工作采用同步时标联络：要求I/O设备与CPU工作速度完全同步

相互之间需配有专用电路，用以产生同步时标来控制同步工作

I/O设备与住机的连接方式：

1. 辐射式：每台I/O设备都有一套控制线路和一组信号线
2. 总线式：通过一组总线（包括地址线、数据线、控制线等）将所有I/O设备与主机相连（现代计算机多采用此方式

I/O设备与主机信息传送的控制方式：

1. 程序查询方式：CPU与I/O设备串行工作，CPU会出现踏步等待

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

（I/O接口上有一个反应是否就绪的状态标记

启动I/O设备CPU开始查询，直到设备准备就绪（CPU的踏步等待），将数据从I/O接口送至CPU再由CPU送入主存

1. 程序中断方式：CPU启动设备后继续自己的工作，当I/O设备准备就绪并向CPU发送中断请求后才响应

大部分时间CPU与I/O设备并行工作 I/O设备准备好后 CPU停止工作

CPU上有INIR接口 CPU每完成一项命令的时钟周期结束时都会检查INIR的状态

1. 直接存储器获取方式（DMA direct mamery access）：主存与I/O设备hi见有一条数据通路，主存与I/O设备交换信息时，不需要调用中断程序，若出现CPU与DMA同时访问主存，CPU总将总线占有权交给DMA，通常把DMA的这种占用称为占有或挪用，又由于窃取的时间一般为一个存取周期，因此又把DMA占用的存取周期的现象叫做窃取周期或挪用周期。

在DMA窃取存取周期时CPU仍能做内部操作（如乘法运算等）

1. I/O通道方式：
2. I/O处理机方式：

Ps：输入设备完成输入程序、数据和操作命令等功能。

I/O接口的功能和组成：

接口可以看做是两个系统或部件之间的交换部分，它即可以是两种硬件设备之间的连接电路，也可以是两个软件之间的共同逻辑边界。

I/O接口通常是指主机与I/O设备之间设置的一个硬件电路及器相应的软件控制。

主机与I/O设置接口的原因：

1. 实现I/O设备的选择
2. 通过接口实现数据缓冲达到I/O设备与主机速度的匹配
3. 通过接口可实现数据串-并格式的转换
4. 通过接口实现电平的转换
5. CPU启动I/O设备需要发送控制信号，通过接口可实现传送控制命令
6. 监视设备的工作状态，并可保存状态信息，供CPU查询

接口与端口区别：

端口使之接口电路中的一些寄存器，这些寄存器分别用来存放数据信息、控制信息和状态信息，相应的端口被称为数据端口、控制端口和状态端口。若干个端口加上相应的控制逻辑才能组成接口。CPU通过输入指令，从端口读入信息，通过输出指令，可将信息写入到端口中。

I/O接口功能及组成：

1. 选址功能：CPU选择设备时将设备码发向所有设备的接口，当设备选择线上的设备码与本设备设备码相符时应发出设备选中信号SEL（通过接口内的设备选择电路实现）
2. 传送命令功能：I/O接口内含命令寄存器及命令译码器，响应CPU的命令
3. 传送数据功能：数据通路，同时还应具有缓冲功能（能将数据暂存在接口内），接口内常设有数据缓冲寄存器，用来暂存I/O设备与主句准备交换的i西南西，与I/O设备中的数据线是相连的。
4. 反映I/O设备工作状态的功能：通过触发器实现

基本组成：数据缓冲寄存器 DBR、设备状态标记、、设备选择电路、命令寄存器和命令译码器、控制逻辑电路。

程序查询方式：

查询流程：

单个I/O设备的查询流程：首先检查状态标记，若准备就绪则交换数据，若没有再次检查状态标记直到准备就绪

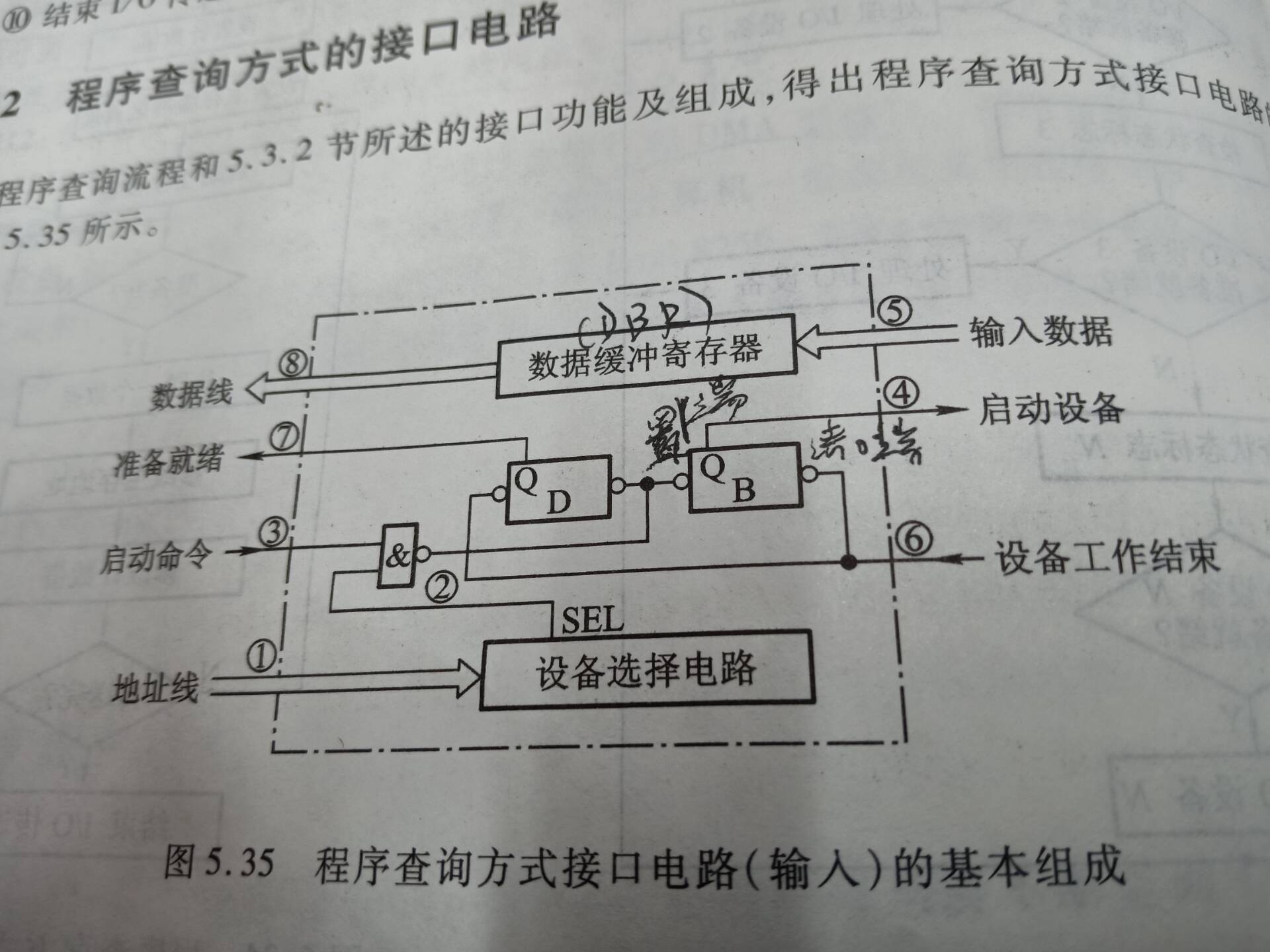
1. 测试指令
2. 传送指令
3. 转移指令

当设备较多时CPU按各个设备在系统中的优先级别进行逐级查询（当所有设备均没有准备就绪时才会进行下一轮查询，若有一个设备准备就绪就处理此设备）

程序查询方式的程序流程：

1. 保护寄存器内容 ->由于此方式需要占用CPU中的寄存器，因此要将寄存器原内容保存起来
2. 设置计数值 ->往往传输一批数据，因此需要有计数值
3. 设置主存缓冲区首址 ->设置欲传送数据在主存缓冲区的首地址
4. 启动I/O设备 ->CPU启动I/O设备
5. 是否准备好（准备好就继续 没准备好反复查询准备状态） ->对于输入 准备就绪代表接口电路数据缓存器已装满欲传送数据成为输入缓冲满 对于输出 准备就绪代表接口电路已被设备取走 输出缓冲空，CPU可再次将数据传送到接口，设备可再次从接口接收数据
6. 传送一个数据 ->CPU执行I/O指令或从I/O接口的数据缓存器读出一个数据或吧一个数据写入I/O接口的数据缓存器内，同时将接口中的状态标志复位
7. 修改主存地址
8. 修改计数值
9. 是否传送结束（若结束 继续 若没结束 返回4 ->计数值位0时代表一批数据已经传送完毕 不为零 重新启动外设继续传送）
10. 结束I/O传送

程序查询的接口电路



（设备选择电路用以识别本设备地址，当地址线上的设备号与本设备号符合时SEL有效可以接收命令；数据缓冲寄存器用于存放欲传送的数据；D时完成触发器，B时工作触发器）

以输入设备为例：

1. CPU通过I/O指令启动输入设备时，指令的设备码字段通过地址线送至设备选择电路
2. 若地址号吻合，其输出SEL有效
3. I/O指令的启动命令经过“与非”门将工作触发器B置“1”，将完成触发器D置“0“
4. 由触发器B启动设备工作
5. 输入设备将数据送至数据缓冲寄存器
6. 由设备发设备工作结束信号将D置“1“B“置“0”，表示外设准备就绪
7. D触发器以“准备就绪“状态通知CPU，表示”数据缓冲满“
8. CPU执行输入指令，将数据缓冲寄存器中的数据送至CPU的通用寄存器，再存入主存相关单元

程序中断方式：

中断的六大步骤：

1. 中断请求：
2. 中断屏蔽：
3. 中断判优：
4. 中断响应：
5. 中断处理：
6. 中断返回：

前四步纯硬件实现后两步软件实现

中断响应：

1. 保护断点
2. 转移中断服务入口
3. 关中断

中断接口电路：

相关线路：

1. 中断请求触发器（INIR）和中断屏蔽触发器（MASK）：INIR为“1”时表示该设备向CPU提出中断请求，且设备欲提出中断请求时，其设备必须准备就绪，即接口内的完成触发器D的状态必须为“1”

把能向CPU提出中断请求的各种因素统称为中断源。

一个原则：多个中断源同时提出请求时，在任何瞬间只能接受一个中断源的请求

当多个中断源同时请求，CPU对他们及逆行排队，只接受级别最高的请求，并通过I/O接口内的屏蔽触发器MASK当其为一时表示被屏蔽，即封锁中断源请求（当设备就绪（D=1）为被屏蔽（MASK = 0）CPU的中断查询信号可将中断请求触发器置“1”。

1. 排队器：就I/O中断而言 速度越高的I/O设备，优先级越高

设备优先权的处理可采用硬件方法也可采用软件方法

1. 中断向量地址形成部件（设备编码器）：CPU一旦响应中断就要执行中断服务程序，每个服务有不同的中断服务程序，每个服务程序都有一个入口地址，CPU必须找到这个入口地址。

入口地址寻址方式也可用硬件或软件的方法来完成，

硬件法时通过向量地址来寻找设备的中断服务入口地址

程序中断方式接口电路的基本组成：P198

CPU响应中断的条件和时刻：

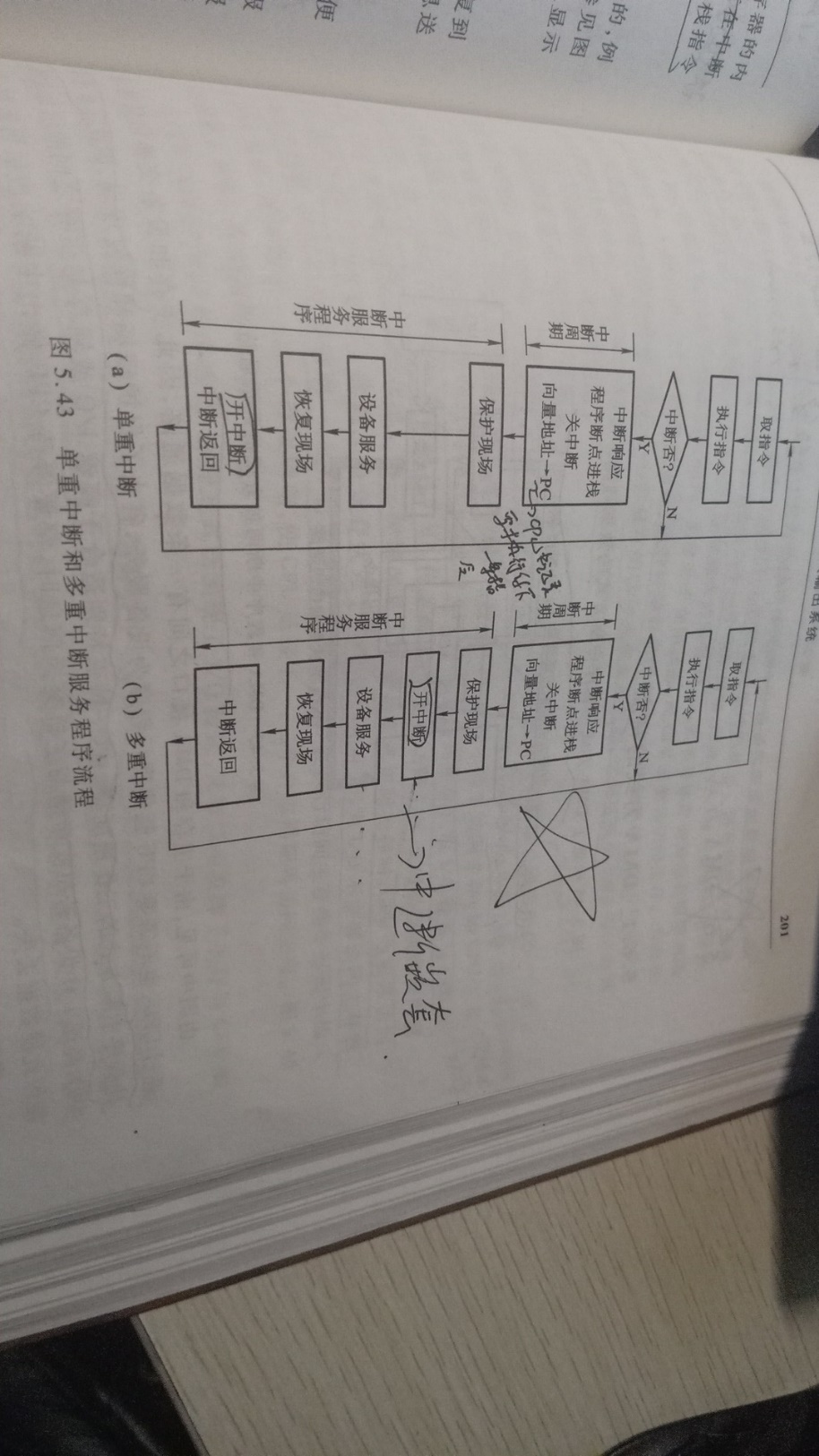
条件：中断触发器EINT为“1” （该触发器可用中断指令置位（称为开中断））；也可用关中断指令或硬件自动使其复位（称为关中断）

时间：每条指令执行阶段的结束时刻（最后一个时钟周期）

中断服务流程：

1. 保护现场：保存程序断点，保存通用寄存器（中断隐指令完成）和状态寄存器的内容（中断服务程序完成）
2. 中断服务（设备服务）
3. 恢复现场：通常可用取数指令或出栈指令（POP）将保存在存储器（或堆栈）中的信息送回到原来的寄存器
4. 中断返回：使程序返回原程序断点处，继续执行原程序

分为单重中断和多重中断：（单重中断不允许中断嵌套 多重中断允许中断嵌套，区别在于开中断的位置，单重中断的开中断在中断返回前恢复现场后，多重中断开中断在设备服务前保护现场后）



附：中断周期： 中断响应->程序断点进栈->关中断->向量地址传给PC

P201

DMA方式：

DMA与主存交换时采用三种方法：P202

1. 停止CPU访问主存：外设传送数据时DMA接口向CPU发送停止信号 CPU放弃地址线、数据线、和有关控制线的使用权。DMA获得控制权后开始数据传送。

优点：控制简单

缺点：DMA访问主存时CPU基本不工作或保持原状态，而且即使I/O设备高速运行，两个数据之间的准备间隔时间总大于一个存取周期CPU对主存利用率没有充分发挥

1. 周期挪用（周期窃取）：DMA请求时I/O设备挪用或窃取总线占有权一个或几个主存周期DMA不请求时CPU继续访问主存。

I/O设备请求DMA传送的三种情况：

* 1. CPU不需要访问主存，不冲突
  2. CPU正在访问主存，此时等待存取周期结束后，让出总线占有权
  3. 同时访问主存，出现访问冲突，I/O访存优于CPU（I/O不立即访问可能丢失数据），I/O设备窃取一、二个存取周期，CPU在执行访问主存指令过程中插入DMA请求，并挪用几个存取周期，使CPU延缓访存

1. DMA与CPU交替访问：适用于CPU工作周期比主存存取周期长的情况

DMA接口的功能和组成：

功能：

1. 向CPU申请DMA传送
2. 在CPU允许DMA工作时，处理总线控制权的转交，避免因进入DMA巩固走而影响CPU正常活动或引起总线竞争
3. 在DMA器件管理系统总线，控制数据传送
4. 确定数据传送的起始地址和数据长度，修正数据传送过程中的数据地址和数据长度
5. 在数据块传送结束时，给出DMA操作完成的信号

DMA接口的基本组成：P205

1. 主存地址寄存器（AR）：存放主存中需要交换数据的地址
2. 字计数器（WC）：记录传送数据的总字数
3. 数据缓冲寄存器（BR）：暂存每次传送的数据
4. DMA控制逻辑：
5. 中断机构
6. 设备地址寄存器（DAR）：存放I/O设备的设备码或表示设备信息存储区的寻址地址的寻址信息

传送过程：

1. 预处理：

DMA接口开始工作前CPU给他预置如下信息：

1. 给DMA控制逻辑指明数据传送方向时输入（写主存）还是输出（读主存）
2. 向DMA设备地址寄存器送入设备号，启动设备
3. 向DMA主存地址寄存器送入交换数据的主存起始地址
4. 对字计数器赋予交换数据的个数
5. 数据传送：DMA以数据块为单位传送

示意图P207

1. 后处理：中断服务程序，做DMA结束处理

第六章 计算机的运算方法

通常称寄存器的位数为机器字长 无符号数范围 0到65535 有符号数范围 -32768到32767

有符号数：

1.真值：

2.原码：

整数：

0，x 2^n>x>=0

2^n-x 0>=x>-2^n

小数

x 1>x>=0

1-x 0>=x>-1

3.反码：

原码除符号位各位取反

4.补码：

整数：

0,x 2^n>X>=0

2^(n+1) + n 0>x>-2^n (mod 2^(n+1))

小数：

x 1>x>=0

2+x 0>x>=-1 (mod 2)

5.移码：最小真值的移码全为零 当浮点数的阶码用移码表示时，就能很方便地判断阶码的大小

X移 = 2^n + x (2^n>x>=-2^n)

Ps：同一个真值的移码和补码仅差一个符号位，若将补码的符号位由“0”变为“1”或从“1”变为“0”即可得到该真值的移码

定点数：小数点固定在某一位置的数称为定点数

当小数点位于数符和第一数值之间时，机器内的数为纯小数；当小鼠带你位于数值位之后时机器内的数位纯整数。

采用定点数的机器称为定点机。数值部分的位数n据欸的那个了定点机中数的表示范围

若机器数原码表示 小数定点机范围 -（1-2^(-n)）到 (1-2^(-n)) 整数定点机范围 -（2^n-1）到 (2^n-1)

在定点机中，小数点位置固定不变，当机器处理的数不是纯小数或纯整数时，必须乘上一个比例因子，否则会产生“溢出”

浮点数由阶码j和位数s两部分组成 阶码时是整数，阶符和阶码的位数m合起来反应浮点数的表示范围及小数点的实际位置；尾数是小数，其位数n反映了浮点数的精度；尾数的符号Sf代表浮点数的正负。

浮点数与真值之间的相互转换：P231 规格化…… 例题

定点运算： 包括移位 加 减 乘 除

移位 正数 移位补零（原 反 补）

负数 原码 补零 反码 补一 补码 左移补零 右移补一

1. 补码加减运算及溢出判断规则：减法就是加法

整数：A补 + B补 = （A+B）补（mod 2^(n+1)）

小数：A补 + B补 = （A+B）补（mod 2)

溢出判断：一位符号位 （加法只有在两个数符号位相同时才可能溢出 减法只有正减负 或 负减正 才可能溢出）结果的符号位与加数不同时溢出

两位符号位 结果的两位符号位 异或值为0时溢出 10表示负溢出 01表示正溢出

1. 原码一位乘法规则：符号位另行计算（异或值）只计算数值位的绝对值的乘积 判断位为0部分积加零结果右移1判断位为1部分积加被乘数绝对值结果右移1 右移次数为乘数的有效位数
2. 补码一位乘法规则： 应用两位符号位 以便判断溢出
3. 乘数是正数：部分积初值为零判断位为0部分积直接右移判断位为1部分积加x补并右移一位（左移乘数的有效位数次）
4. 乘数是负数：先不考虑乘数符号位按正数计算最后加上（-x）补进行修正

比较法（Booth算法）：部分积仍取双符号位，乘数因符号位参与运算，故多一位

附加位初始为0，由附加位于乘数最后一位共同判断 10 部分积加（-x）补然后右移一位，11部分积直接右移一位，01部分积加x补然后右移一位，00部分积直接右移一位。（注：最后一步若出现10 或01不移位 因此移位次数为乘数有效位数）

浮点加减运算规则：浮点数 阶码；尾数

1. 对阶：使阶码相同 小阶向大阶对齐 小阶的尾数右移阶码的差值次
2. 尾数求和：按定点数规则直接相加减
3. 规格化： 1/2<=abs(S)<1 小数点左右必须不同 相同时左规 尾数左移一次阶码减一 符号位不同时 右规一次 尾数右移阶码加一 ？？？？？？？？直到符号位相同
4. 舍入： 规格化时尾数低位丢失 （处理方法 两种 1 “0舍1入”法：右移时最高位为1尾数末加一为零不做处理直到不溢出 2 “恒置1”法：无论怎么样将尾数置1.
5. 溢出判断：规格化右规之后再根据阶码判断是否溢出

浮点数溢出

阶码范围 -（2^m-1） < 阶码 <2^m-1

阶码大于上限为上溢 小于下限为下溢

第七章 指令系统

操作码：指明该指令所要完成的操作 其位数反映了机器的操作种类 如操作码占7位 最多有2^7条指令

1. 固定编码：将操作码放在指令字的一个字段内，便于硬件设计，指令译码时间断，广泛用于字长较长的、大中型计算机和超级小型计算机及RISC中
2. 扩展编码技术：使操作码的长度随地址数的减少而增加

（原则：使用频度高的分配短的操作数，短码不能是长码的前缀

设计指令原则：

访存次数：越少越好

指令寻址范围：越大越好（扩大指令寻址范围）

与数据有关的寻址方式：

1. 立即寻址：操作数本身设在指令字内，即形式地址 三部分组成 OP+立即寻址特征+A（立即数）
2. 寄存器寻址：指令的地址码字段直接指出了寄存器的编号 操作数在次寄存器内
3. 直接寻址：指令字中的形式地址A就是操作数的真实地址EA
4. 间接寻址：指令字的形式地址A指向的地址中是操作数有效地址所在的存储单元
5. 寄存器间接寻址：指向寄存器 寄存器内地址是操作数有效地址所在的存储单元
6. 基址寻址：含基址寄存器BR 操作数的有效地址是 形式地址A与基址寄存器内的内容（基地址）相加。
7. 变址寻址：变址寄存器（IX）有效地址是形式地址A与变址寄存器内容相加
8. 相对寻址：将程序计数器PC的内容（即当前指令字的地址）与指令字中的形式地址A相加而成

EA = （PC）+ A

相对寻址移位量 （PC）+ 2^（m-1） ~ (PC) + 2^（m-1） -1（m为操作码位数）

广泛用于转移指令

“\*”：相对寻址特征

1. 隐含寻址：指令字中不明显的给出操作数的地址，其操作数的地址隐含在操作码或某个寄存器中

基址寻址与变址寻址的区别：

基址寻址主要用于位程序或数据分配存储空间，故计指寄存器的内容通常由操作系统或管理程序设定，在程序的执行过程中其值是不可变的，而A是可变的。在变址寻址中，变址寄存器的内容是由用户设定的，在程序执行过程中其值是可变的，而指令字中的A是不可变的。变址寻址主要用于处理数组问题（可设定A为数组首地址，改变IX内容就可形成数组中任一数据的地址）

第八章 CPU结构和功能

指令周期流程：

取指周期->间址周期（需判断是否存在 存在才进入）->执行周期->中断周期（需判断是否存在）

取指周期：取指令：

间址周期：取有效地址

执行周期：取操作数（当指令为访存指令时）

中断周期：保存程序断点

（为区分上述四个周期 CPU内可设置四个标志触发器）

引起中断的各种因素：

1. 人为设置的中断：一般称为自愿中断
2. 程序性事故：程序设计不周引起的中断 （如定点溢出 浮点溢出 操作码不能识别等）
3. 硬件故障：如插件接触不良，通风不良，磁盘表面损坏，电源掉电等
4. I/O设备：I/O设备被启动以后，一旦准备就绪，便向CPU发出中断请求
5. 外部事件：用户通过键盘来中断现行程序属于外部事件中断

中断请求标记和中断判优逻辑：

1. 中断请求标记：中断请求标记触发器，检查中断请求触发器，记作INIR，为“1“时表示中断源有请求
2. 中断判优逻辑：任何一个中断系统，在任一时刻，只能相应一个中断源的请求。（因此需要中断判优，中断判优可以硬件实现也可软件实现
   1. 硬件排队：
      1. 链式排队器：对应中断请求触发器分散在各个接口电路中的情况，每一个接口电路中都设有一个非门和一个与非门，它们向链条一样串接起来。（P196 图5.38）
      2. 设在CPU内：P361 图8.25
   2. 软件排队：编写查询程序实现

中断隐指令：

CPU响应中断后，即进入中断周期。在中断周期内要自动完成下列操作：

1. 保护程序断点：将当前程序计数器PC的内容（程序断点）保存到存储器中。它可以存在存储器的特定单元（如0号地址）内也可以存入堆栈。
2. 寻找服务程序入口地址：中断周期结束后进入下条指令（中断服务程序的第一条指令）的取指周期，因此在中断周期内必须设法找到中断服务周期的入口地址。入口地址有两种方法获得，因此中断周期内有两种方法寻找入口地址：
   1. 中断周期内，将向量地址送至PC（对应硬件向量法），使CPU执行下一条无条件转移指令，转至中断服务程序的入口地址。
   2. 中断周期内，将软件查询入口地址的地址（又称中断识别程序）首地址送至PC，使CPU执行中断识别程序，找到入口地址（对应软件查询法）。
3. 关中断：关中断以禁止CPU再次响应新的中断请求

Ps：上述保护断点、寻找入口地址和关中断都是在中断周期内由一条隐指令完成的，所谓中断隐指令，即在机器指令系统中没有的指令，它是CPU在中断周期内由硬件自动完成的一条指令。

中断屏蔽技术（主要用于多重中断）P366

1. 改变屏蔽字，改变中断处理顺序：
2. CPU执行程序轨迹：（根据硬件排队优先级 和 屏蔽字可画出CPU执行轨迹

第九章、第十章 控制单元

指令周期、机器周期和时钟周期 概念及关系：

指令周期：CPU每取出并执行一条指令所需的全部时间称为指令周期，即CPU完成一条指令的时间

机器周期：机器周期可看做使所有指令执行过程中的一个基准时间，机器周期取决于指令的功能及器件的速度

（通常以访问一次存储区的时间定为基准时间，又由于不论执行什么指令都需要访问存储器取出指令，因此在存储字长等于指令字长的前提下，取指周期也可看做机器周期。

时钟周期（节拍 状态）：时钟信号可由机器主振电路发出的脉冲信号经整形后产生，时钟信号的频率即为CPU主频。用时钟信号控制节拍发生器，就可产生节拍。

根据CPU的数据通路，写出指令微操作：

1. 取指周期：
   1. PC->MAR
   2. 1->R
   3. M(MAR)->MDR
   4. MDR->IR
   5. OP(IR)->CU(指令操作码送给CU译码)
   6. (PC)+1->PC
2. 间址周期：完成取操作数有效地址的任务
   1. ad(IR)->MAR
   2. 1->R
   3. M(MAR)->MDR
   4. MDR->ad(IR) (有些机器省略此步)
3. 执行周期：不同执行周期的微操作是不同的，下分别讨论非访存指令、访存指令、转移类指令的微操作：
   1. 非访存指令：
      1. 清除累加器指令CLA： 该指令只完成清除累加器操作 0->ACC
      2. 累加器取反指令COM：该指令只完成累加器内容取反，结果返送累加器操作 记作 ACC（非 ，上面加横线）->ACC
      3. 算数右移一位指令SHR：该指令在执行阶段只完成累加器内容算数右移一位操作
         1. L（ACC）-> R(ACC)
         2. ACC o-> ACCo (ACC符号位不变）
      4. 循环左移一位指令CSL：该指令在执行阶段只完成累加器内容循环左移一位操作
         1. R（ACC）->L(ACC)
         2. ACC o-> ACCn (或p^-1（ACC）
      5. 停机指令STP：计算机中有一个运行标志触发器G当G=1时表示机器运行G=0表示停机，STP指令在执行阶段只需将运行标志触发器置“0“，记作 0->G
   2. 访存指令：需要访问存储器 这里只考虑直接寻址方式（形式地址A就是操作数地址 不考虑其他
      1. 加法指令ADD X：累加器内容与主存地址X内容相加结果送至累加器
         1. ad(IR)->MAR
         2. 1->R
         3. M(MAR)->MDR
         4. (MDR）+ （ACC）-> ACC (向ALU发送加命令，将ACC内容与MDR内容相加结果存ACC
      2. 存数指令STA X：该指令在执行阶段将ACC内容存至贮存单元X
         1. ad(IR)->MAR
         2. 1->W
         3. ACC->MDR
         4. MDR->M(MAR)
      3. 取数指令 LDA X
         1. ad(IR)->MAR
         2. 1->R
         3. M(MAR)->MDR
         4. MDR->ACC
   3. 转移类指令：执行阶段也不访问存储器
      1. 无条件转移指令JMP X：该指令在执行阶段完成将地址码部分X送至PC 记作 Ad（IR）-> PC
      2. 条件转移（负则转）指令BAN X：根据上一条指令运行的结果决定下一条指令的地址，若结果为负（累加器最高位为1即Ao = 1），则指令的地址码送至PC，否则程序按原顺序执行。由于在取址阶段已经完成了（PC）+1->PC，所以当累加器结果不为负（即Ao=0）时，就按取指阶段形成的PC执行，记作 Ao·Ad（IR）+ A（非）·（PC）-> PC.
4. 中断周期：
   1. 0->MAR （将特定地址“0“送至存储器地址寄存器）
   2. 1->W (向主存发出写命令，启动存储器作写操作)
   3. PC->MDR (将PC内容（程序断点）送至MDR)
   4. MDR->M(MAR) (将MDR内容写入到MAR所指示的主存单元（o地址单元）)
   5. 向量地址->PC （将向量地址形成部件的输出送至PC，为下一条指令的取指周期作准备
   6. 0->EINT （关中断 将允许中断触发器清零 该操作可由硬件线路完成）

Ps：如果程序断点存入堆栈，而且进栈操作时先修改栈指针，后存入数据至需将 第一步变为 （SP）-1->SP 且 SP->MAR

上述所有操作都是在控制单元发出的控制信号（即微操作命令）控制下完成的。

配合节拍以及微命令

微操作的节拍安排： 采用同步控制方式 一个机器周期内由三个节拍（时钟周期）

原则：

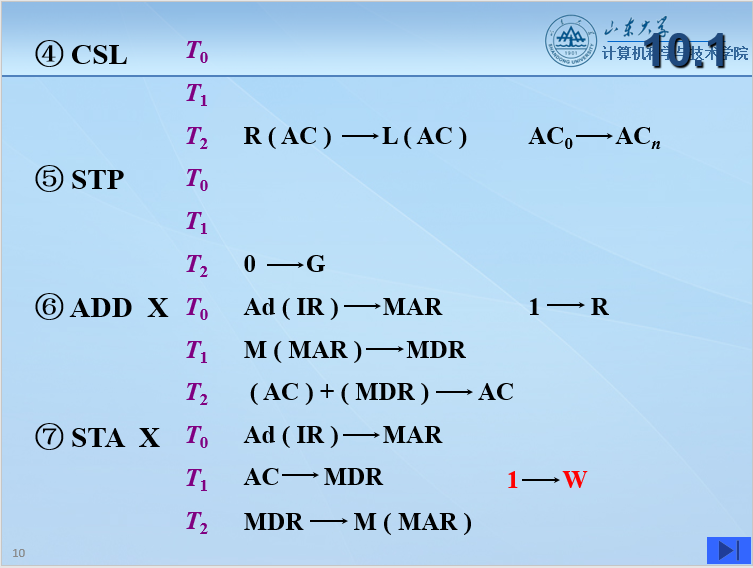
1. 微操作的先后顺序不得随意更改
2. 被控对象不同的微操作 尽量安排在一个节拍内完成
3. 占用时间较短的微操作，尽量安排在一个节拍内完成，并允许有先后顺序

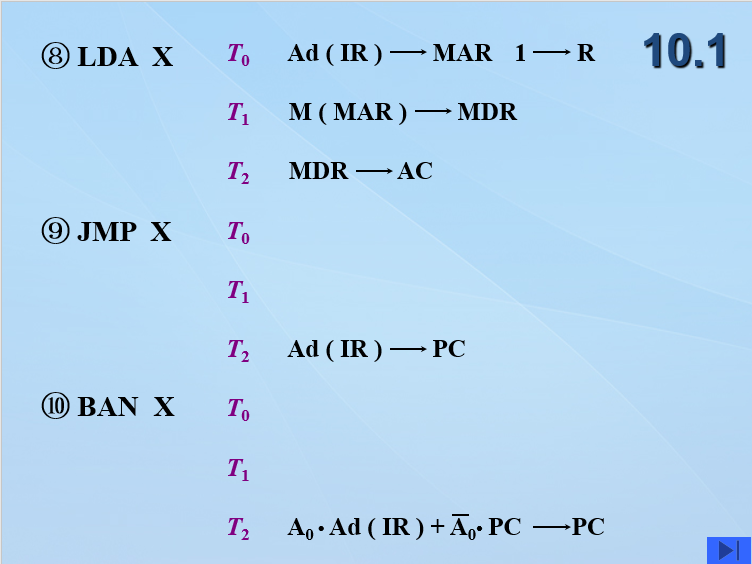
微程序控制器的基本概念：

1. 微命令：构成控制信号序列的最小单位
2. 微操作：由微命令控制实现的最基本操作
3. 微指令：若干个微命令的组合
4. 微周期：指从控制存储器中读取一条微指令并执行相应的微操作所需的时间
5. 微程序：一系列指令的有序集合
6. 控制存储器：存放微程序的制度存储器





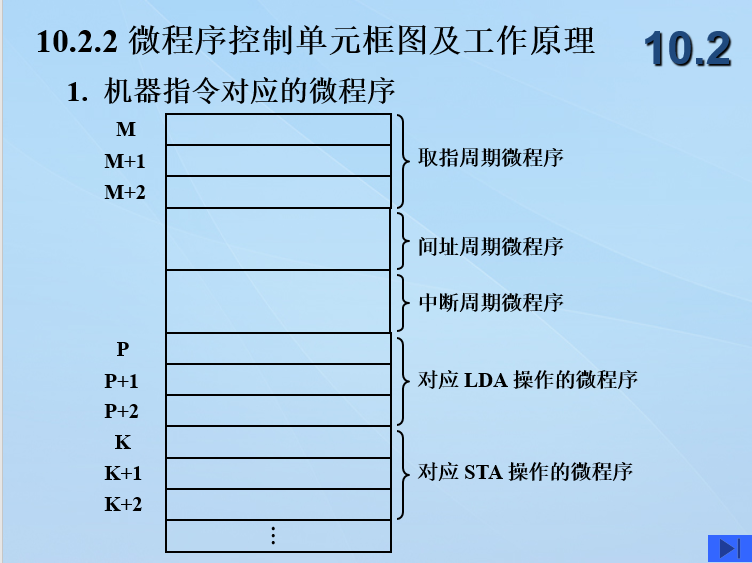


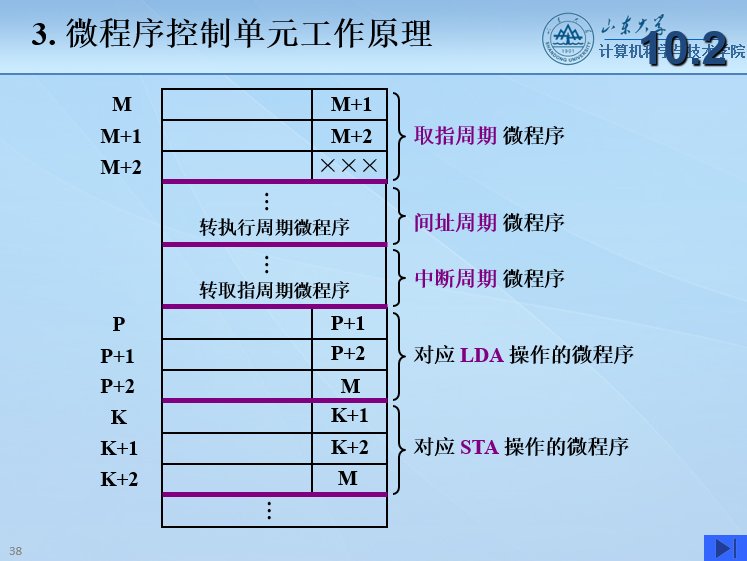




微指令和微操作的关系，微指令和机器指令的关系，微程序和程序之间的关系

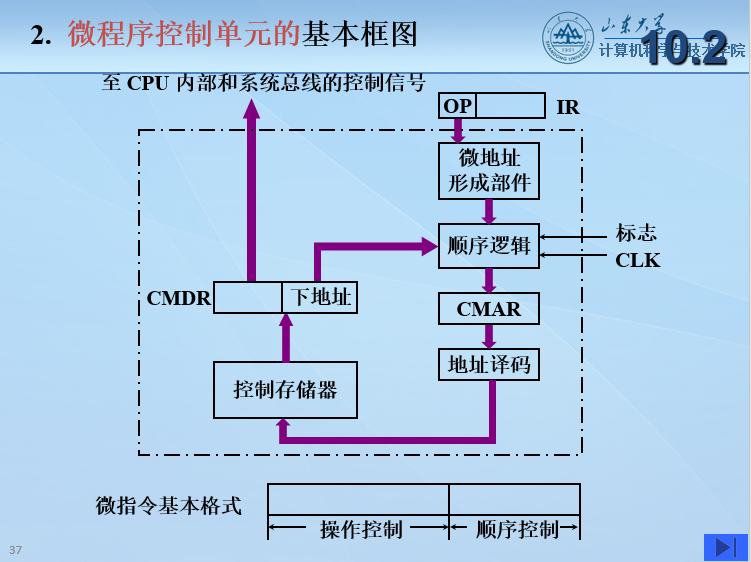
1. 微指令是若干个微操作的组合，微命令时构成控制信号序列的最小单位，微操作时有微命令控制实现的最基本单位
2. 微指令时若干个微命令的集合，微程序是一系列微指令的有序集合。微程序是机器指令的实时解释器，每一条机器指令都对应一个微命令
3. 微程序是由微指令组成的，用于描述机器指令，实际上是机器指令的实时解释器，微程序是由计算机的设计者事先编制好并存放在控制存储器中的，一般不提供给用户；程序是由机器指令组成的，有程序员事先编制好并存放在主存放器中。





微程序控制器的工作原理

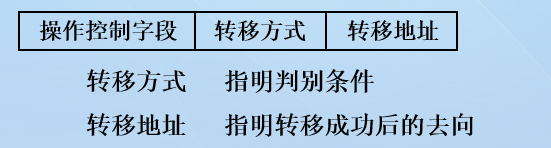
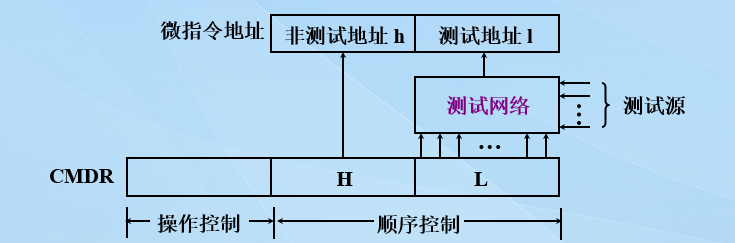
微程序控制单元的基本组成 图10.5

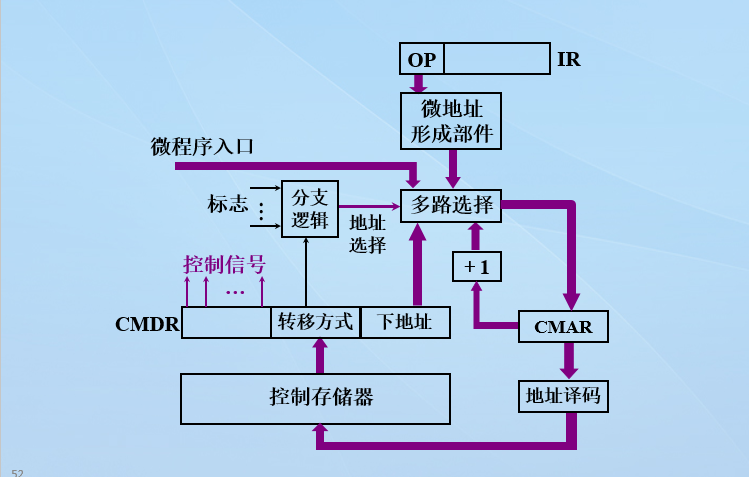


微指令的编码方式：

1. 直接编码：每一位代表一个微操作命令，这种编码方式即为直接编码方式
   1. 优点：含义清晰 速度快
   2. 缺点：微操作命令多，可能使微指令操作控制字段达几百位，造成控存容量极大
2. 字段直接编码方式：将为指令的控制字段分成若干“段”每段经译码后发出控制信号
   1. 显式编码
   2. 微程序执行速度较慢（译码电路
3. 字段间接编码方式：一个字段的某些微命令还需由另一个字段中的某些微命令来解释
   1. 隐式编码
   2. 进一步缩短指令字长，单削弱了微指令的并行控制能力
4. 混合编码：直接编码和字段编码（直接和间接）混合使用

微指令序列地址的形式：

1. 微指令的下地址字段指出：断定方式
2. 根据机器指令的操作码形成：微指令地址由操作码经微地址形成部件形成
3. 增量计数器：**( CMAR ) + 1 –> CMAR 对于顺序地址可采用此方法（多数情况下 后续微指令地址是连续的）**
4. 分支转移：若不成功则顺序执行
5. 通过测试网络：
6. 由硬件产生微程序入口地址：
   1. 第一条微指令 可有专门硬件电路产生，也可由外部直接向CMAR输入微指令的地址
   2. 中断周期：由硬件产生中断周期微程序的入口地址
   3. 间址周期：由硬件产生简直周期微程序的入口地址
7. 后序微指令地址形成方式原理图：



多路选择 可选下列四路地址：

1. （CMAR）-> CMAR
2. 微指令的下地址字段
3. 指令寄存器（通过微指令形成部件
4. 微程序入口地址