计算机考研复试面试常问问题 计算机组成原理篇

计算机考研复试面试常问问题 计算机组成原理篇

第一章、计算机系统概述
第二章、数据的表示和运算（偏笔试一点的一章）
第三章、存储系统
第四章、指令系统
第五章、中央处理器
第六章、总线
第七章、输入输出系统

第一章、计算机系统概述

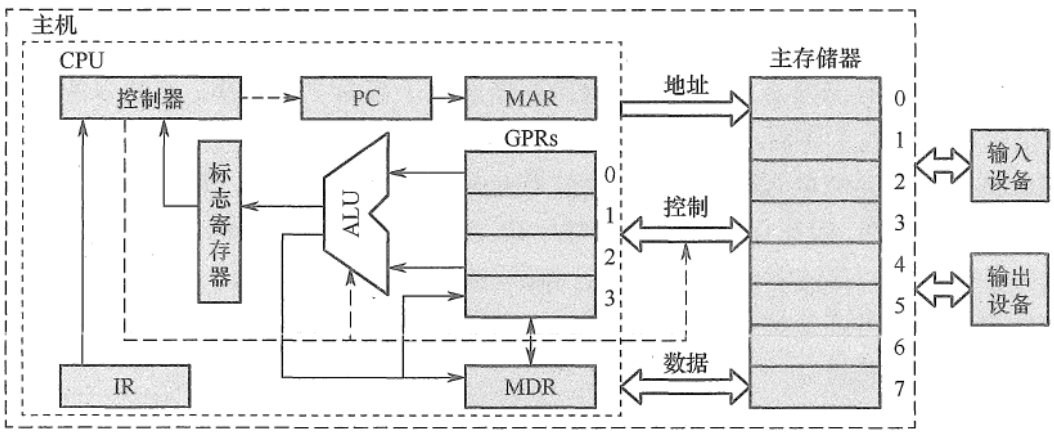
快速唤起记忆知识框架:

- 计算机系统概述
  - 发展历程
    - 第一代：电子管
    - 第二代：晶体管
    - 第三代：中小规模集成电路
    - 第四代：超大规模集成电路
  - 系统组成
    - 运算器：计算机的执行部件，用于对数据进行加工处理
    - 控制器：计算机的指挥中心，由其控制各部件自动协调地进行工作
    - 存储器：计算机的存储部件，用于存放程序和数据
    - 输入设备：将程序和数据以机器所能识别和接受的信息形式输入到计算机
    - 输出设备：将计算机处理的结果以某些的形式输出
  - 性能指标
    - 机器字长：计算机进行一次整数运算所能处理的二进制数据的位数
    - 数据通路带宽：数据总线一次所能并行传递信息的位数
    - 主存容量：主存储器所能存储信息的最大容量
    - 运算速度
      - 吞吐量：系统在单位时间内处理请求的数量
      - 响应时间：从用户提交请求到系统首次产生响应并获得其所需要的结果所用的时间
      - 时钟周期：节拍脉冲或T周期，CPU中最小的时间单位
      - 主频：机器内部主时钟的频率，时钟周期的倒数
      - CPI：执行一条指令所需的时钟周期数
      - 执行时间：运行一个程序所花费的时间
      - MIPS：每秒执行多少百万条指令
      - FLOPS：每秒执行多少次浮点运算，分为MFLOPS、GFLOPS等

1.冯诺依曼机和存储程序的概念?

> 冯·诺依曼在研究EDVAC 机时提出了“存储程序”的概念，“存储程序”的思想奠定了现代计算机的基本结构，以此概念为基础的各类计算机通称为冯·诺依曼机，其特点如下：
>
> 1) 计算机硬件系统由运算器、存储器、控制器、输入设备和输出设备5 大部件组成。
>
> 2) 指令和数据以同等地位存储在存储器中，并可按地址寻访。
>
> 3) 指令和数据均用二进制代码表示。
>
> 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，地址码用来表示操作数在存储器中的位置。
>
> 5) 指令在存储器内按顺序存放。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) 早期的冯诺依曼机以运算器为中心，输入 / 输出设备通过运算器与存储器传送数据。现代计算机以存储器为中心。

> “存储程序”的概念是指将指令以代码的形式事先输入计算机的主存储器，然后按其在存储器中的首地址执行程序的第一条指令，以后就按该程序的规定顺序执行其他指令，直至程序执行结束。

> 冯诺依曼结构的模型机：
>
> 
>

2.计算机的工作过程?

> 计算机的工作过程分为以下三个步骤：
> 1) 把程序和数据装入主存储器。
> 2) 将源程序转换成可执行文件。
> 3) 从可执行文件的首地址开始逐条执行指令。

3.在计算机系统结构中，什么是编译？什么是解释？

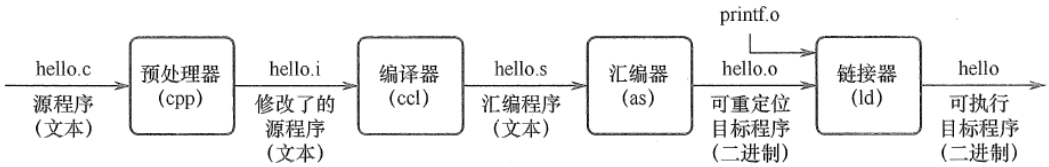
> 翻译的方式有两种，一个是编译，一个是解释。

> 编译型语言写的程序在执行之前，需要一个专门的编译过程，把程序编译成为机器语言的文件，比如exe文件，如果源程序不变以后要运行的话就不用重新翻译。

> 解释则不同，解释性语言的程序不需要编译，在运行程序的时候才翻译，翻译一句执行一句，不生成目标程序，这样解释性语言每执行一次就要翻译一次，效率比较低。

> .java文件->编译->.class文件，编译成.class字节码,.class需要jvm解释，然后解释执行。Java很特殊，Java程序需要编译但是没有直接编译成机器语言，即二进制语言，而是编译成字节码（.class）再用解释方式执行。java程序编译以后的class属于中间代码，并不是可执行程序exe，不是二进制文件，所以在执行的时候需要一个中介来解释中间代码，这就是所谓的java虚拟机（JVM）。

> C语言编译过程分成四个步骤：
>
> 1， 由.c文件到.i文件，这个过程叫预处理,将#include包含的头文件直接拷贝到hello.c当中；将#define定义的宏进行替换，同时将代码中没用的注释部分删除等
>
> 2， 由.i文件到.s文件，这个过程叫编译
>
> 3， 由.s文件到.o文件，这个过程叫汇编
>
> 4， 由.o文件到可执行文件，这个过程叫链接,将翻译成的二进制与需要用到库绑定在一块



4.描述一下指令执行过程？

> 程序中第一条指令的地址置于PC 中，根据PC 取出第一条指令，经过译码、执行步骤等，控制计算机各功能部件协同运行，完成这条指令的功能，并计算下一条指令的地址。用新得到的指令地址继续读出第二条指令并执行，直到程序结束为止。下面以取数指令（即将指令地址码指示的存储单元中的操作数取出后送至运算器的ACC 中）为例进行说明，其信息流程如下：
>
> 1) 取指令： PC —>MAR—>M—>MDR—>IR
>
> 根据PC 取指令到IR, 将PC 的内容送MAR, MAR 中的内容直接送地址线，同时控制器将读信号送读 / 写信号线，主存根据地址线上的地址和读信号，从指定存储单元读出指令，送到数据线上，MDR 从数据线接收指令信息，并传送到IR 中。
>
> 2) 分析指令： OP(IR)—>CU
> 指令译码并送出控制信号。控制器根据IR 中指令的操作码，生成相应的控制信号，送到不同的执行部件。在本例中, IR 中是取数指令，因此读控制信号被送到总线的控制线上。
>
> 3) 执行指令: Ad(IR)—>MAR—>M—>MDR—>ACC
> 取数操作。将IR 中指令的地址码送MAR, MAR 中的内容送地址线，同时控制器将读信号送读 / 写信号线从主存指定存储单元读出操作数，并通过数据线送至MDR, 再传送到ACC 中。

> 此外，每取完一条指令，还须为取下一条指令做准备，形成下一条指令的地址，即(PC)+1 —> PC 。

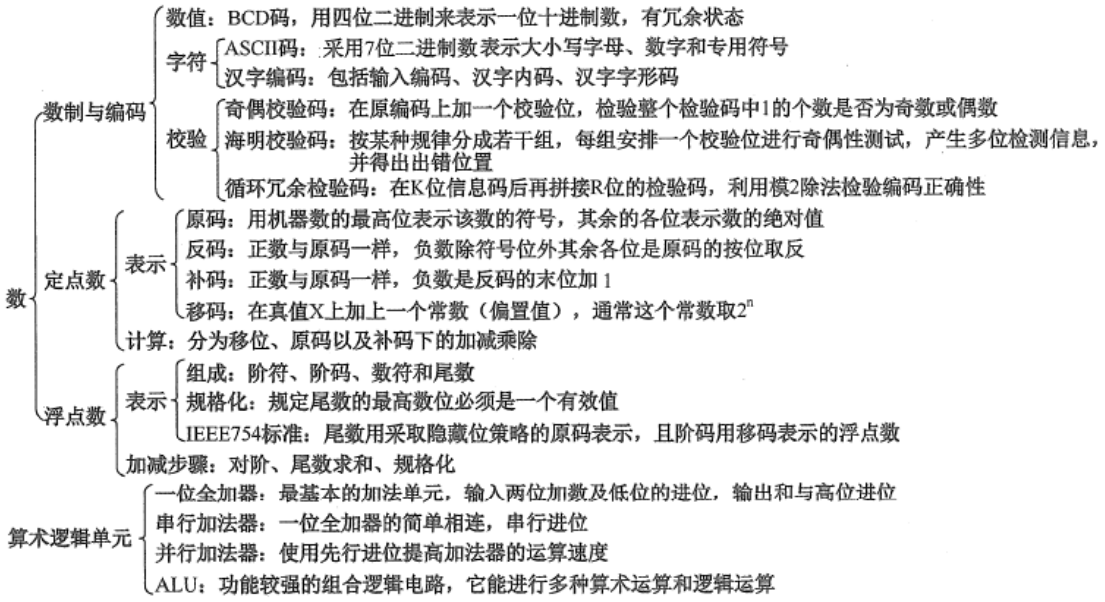
5.计算机的主要性能指标?

1. 机器字长
   机器字长是指计算机进行一次整数运算（即定点整数运算）所能处理的二进制数据的位数，通常与CPU 的寄存器位数、加法器有关。因此，机器字长一般等于内部寄存器的大小，字长越长，数的表示范围越大，计算精度越高。计算机字长通常选定为字节(8 位）的整数倍。
2. 数据通路带宽
   数据通路带宽是指数据总线一次所能并行传送信息的位数。这里所说的数据通路宽度是指外部数据总线的宽度，它与CPU 内部的数据总线宽度（内部寄存器的大小）有可能不同。各个子系统通过数据总线连接形成的数据传送路径称为数据通路。
3. 主存容量
   主存容量是指主存储器所能存储信息的最大容量，通常以字节来衡量，也可用字数x字长（如512Kx16 位）来表示存储容量。其中， MAR 的位数反映存储单元的个数， MAR 的位数反映可寻址范围的最大值（而不一定是实际存储器的存储容量）。
4. 运算速度
   (1) 吞吐量和响应时间。
   • 吞吐量：指系统在单位时间内处理请求的数量。它取决于信息能多快地输入内存， CPU能多快地取指令，数据能多快地从内存取出或存入，以及所得结果能多快地从内存送给一台外部设备。几乎每步都关系到主存，因此系统吞吐量主要取决于主存的存取周期。
   • 响应时间：指从用户向计算机发送一个请求，到系统对该请求做出响应并获得所需结果的等待时间。通常包括CPU 时间（运行一个程序所花费的时间）与等待时间（用于磁盘访问、存储器访问、I/0 操作、操作系统开销等的时间）。
   (2) 主频和CPU 时钟周期。
   • CPU 时钟周期：通常为节拍脉冲或T周期，即主频的倒数，它是CPU 中最小的时间单位，每个动作至少需要1 个时钟周期。
   •主频：机器内部时钟的频率。

   (3)CPI(Clock cycle Per Instruction),即执行一条指令所需的时钟周期数。

第二章、数据的表示和运算（偏笔试一点的一章）

快速唤起记忆知识框架：



6.IEEE754标准浮点数

按照 IEEE 754 标准，常用的浮点数的格式如图 2.14 所示。

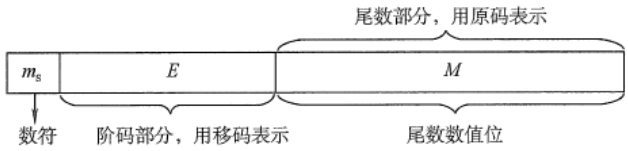


图 2.14　IEEE 754 标准浮点数的格式

IEEE 754 标准规定常用的浮点数格式有短浮点数（单精度、float 型）、长浮点数（双精度、double 型）、临时浮点数，见表 2.6。

表 2.6　IEEE 754 浮点数的格式

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

IEEE 754 标准的浮点数（除临时浮点数外），是尾数用采取隐藏位策略的原码表示，且阶码用移码表示的浮点数。

以短浮点数为例，最高位为数符位；其后是 8 位阶码，以 2 为底，用移码表示，阶码的偏置值为 $2^{8-1}-1=127$；其后 23 位是原码表示的尾数数值位。对于规格化的二进制浮点数，数值的最高位总是“1”，为了能使尾数多表示一位有效位，将这个“1”隐含，因此尾数数值实际上是 24 位。隐含的“1”是一位整数。在浮点格式中表示的 23 位尾数是纯小数。例如，$(12)_{10}=(1100)_2$，将它规格化后结果为 1.1×2^3，其中整数部分的“1”将不存储在 23 位尾数内。

注意：短浮点数与长浮点数都采用隐含尾数最高数位的方法，因此可多表示一位尾数。临时浮点数又称扩展精度浮点数，无隐含位。

7.C语言中的浮点数类型及类型转换

> C 语言中的float 和double 类型分别对应于IEEE 754 单精度浮点数和双精度浮点数。long double 类型对应于扩展双精度浮点数，但long double 的长度和格式随编译器和处理器类型的不同而有所不同。在C 程序中等式的赋值和判断中会出现强制类型转换，以char->int->long->double 和 float->double最为常见，从前到后范围和精度都从小到大，转换过程没有损失。

1) 从int 转换为float 时，虽然不会发生溢出，但int 可以保留32 位， float 保留24 位，可能有数据舍入，若从int 转换为double 则不会出现。

2) 从int 或float 转换为double 时，因为double 的有效位数更多，因此能保留精确值。

3) 从double 转换为float 时，因为float 表示范围更小，因此可能发生溢出。此外，由于有效位数变少，因此可能被舍入。

4) 从float 或double 转换为int 时，因为int 没有小数部分，所以数据可能会向0 方向被截断（仅保留整数部分），影响精度。另外，由于int 的表示范围更小，因此可能发生溢出。

8.在计算机中，为什么要采用二进制来表示数据?

从可行性来说，采用二进制，只有0 和1 两个状态，能够表示0 、1 两种状态的电子器件很多，如开关的接通和断开、晶体管的导通和截止、磁元件的正负剩磁、电位电平的高与低等，都可表示0 、1 两个数码。使用二进制，电子器件具有实现的可行性。

从运算的简易性来说，二进制数的运算法则少，运算简单，使计算机运算器的硬件结构大大简化（十进制的乘法九九口诀表有55 条公式，而二进制乘法只有4 条规则）。

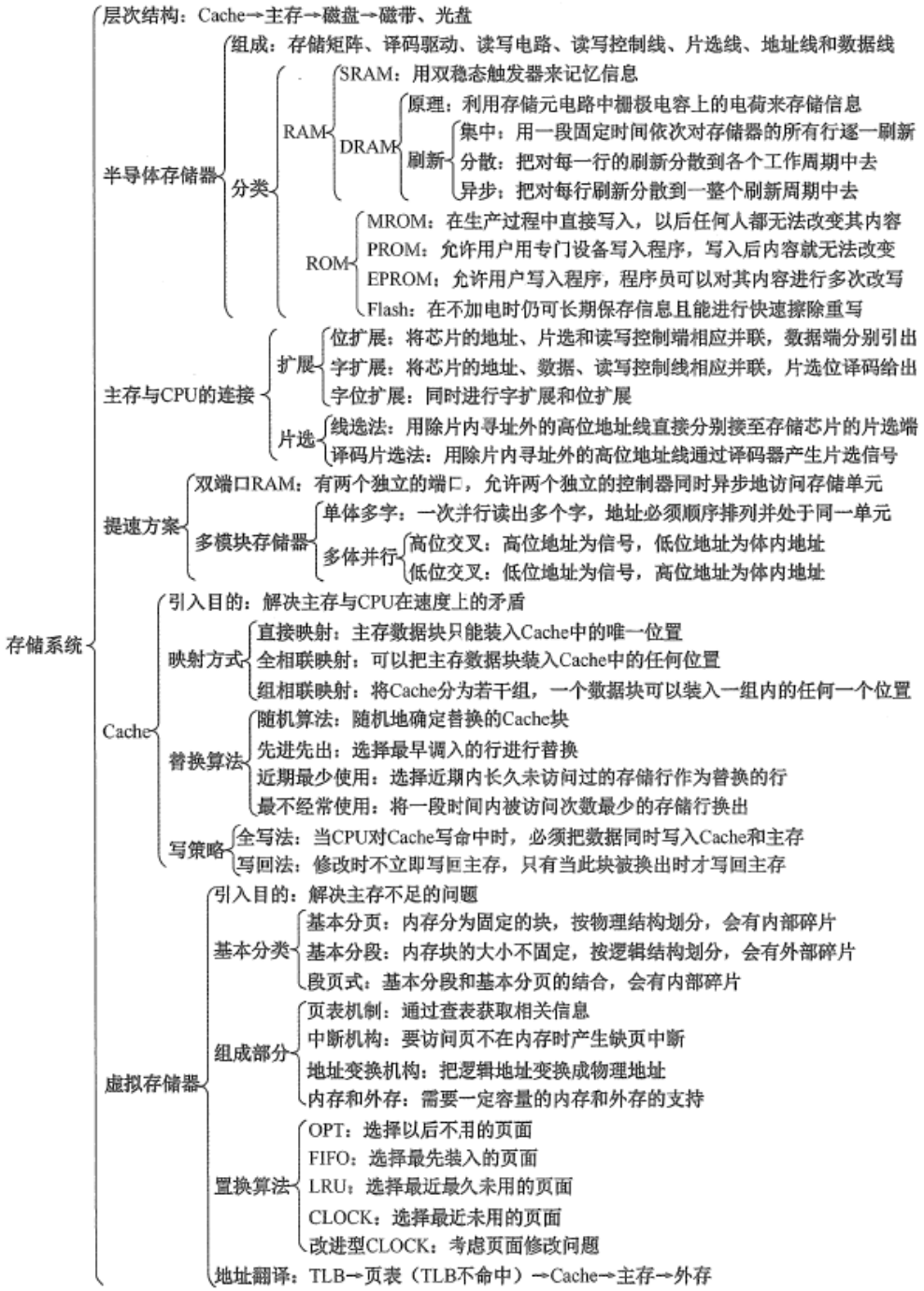
从逻辑上来说，由于二进制0 和1 正好和逻辑代数的假(false) 和真(true) 相对应，有逻辑代数的理论基础，用二进制表示二值逻辑很自然。

9.各编码方式的数值范围

编码方式	最小值编码	最小值	最大值编码	最大值	数值范围
无符号定点整数	0000…000	0	1111…111	$2^{n+1}-1$	$0 \le x \le 2^{n+1}-1$
无符号定点小数	0.00…000	0	0.11…111	$1-2^{-n}$	$0 \le x \le 1-2^{-n}$
原码定点整数	1111…111	-2^n+1	0111…111	2^n-1	$-2^n+1 \le x \le 2^n-1$
原码定点小数	1.111…111	$-1+2^{-n}$	0.111…111	$1-2^{-n}$	$-1+2^{-n} \le x \le 1-2^{-n}$
补码定点整数	1000…000	-2^n	0111…111	2^n-1	$-2^n \le x \le 2^n-1$
补码定点小数	1.000…000	-1	0.111…111	$1-2^{-n}$	$-1 \le x \le 1-2^{-n}$
反码定点整数	1000…000	-2^n+1	0111…111	2^n-1	$-2^n+1 \le x \le 2^n-1$
反码定点小数	1.000…000	$-1+2^{-n}$	0.111…111	$1-2^{-n}$	$-1+2^{-n} \le x \le 1-2^{-n}$
移码定点整数	0000…000	-2^n	1111…111	2^n-1	$-2^n \le x \le 2^n-1$
移码定点小数	小数没有移码定义				

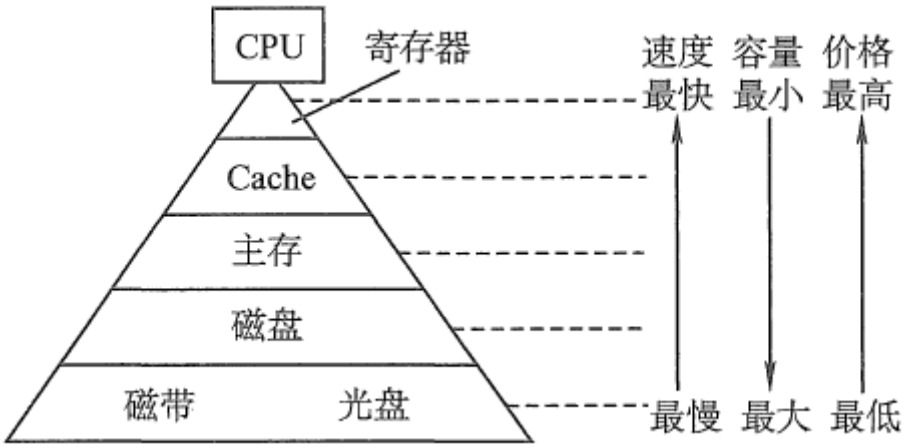
第三章、存储系统

快速唤起记忆知识框架：



10.多级存储系统?

> 为了解决存储系统大容量、高速度和低成本3 个相互制约的矛盾，在计算机系统中，通常采用多级存储器结构，在图中由上至下，位价越来越低，速度越来越慢，容量越来越大， CPU 访问的频度也越来越低。



实际上，存储系统层次结构主要体现在"Cache-主存"层次和"主存 - 辅存"层次。前者主要解决CPU和主存速度不匹配的问题，后者主要解决存储系统的容量问题。在存储体系中， Cache 、主存能与CPU 直接交换信息，辅存则要通过主存与CPU 交换信息；主存与CPU 、Cache 、辅存都能交换信息。

存储器层次结构的主要思想是上一层的存储器作为低一层存储器的高速缓存。从CPU 的角度看，"Cache—主存"层次速度接近于Cache, 容量和位价却接近千主存。从"主存—辅存"层次分析，其速度按近于主存，容址和位价动接近于辅存。这就解决了速度、容量、成本这三者之间的矛盾。

在"主存—辅存"这一层次的不断发展中，逐渐形成了虚拟存储系统，在这个系统中程序员编程的地址范围与虚拟存储器的地址空间相对应。对具有虚拟存储器的计算机系统而言，编程时可用的地址空间远大于主存间。

11.半导体随机存储器?

主存储器由DRAM 实现，靠处理器的那一层(Cache) 则由SRAM 实现，它们都属于易失性存储器，只要电源被切断，原来保存的信息便会丢失。DRAM 的每比特成本低千SRAM, 速度也慢于SRAM, 价格差异主要是因为制造DRAM 需要更多的硅。而ROM 属千非易失性存储器。

1.SRAM 的工作原理

通常把存放一个二进制位的物理器件称为存储元，它是存储器的最基本的构件。地址码相同时多个存储元构成一个存储单元。若干存储单元的集合构成存储体。

静态随机存储器(SRAM) 的存储元是用双稳态触发器（六晶体管MOS) 来记忆信息的，因此即使信息被读出后，它仍保持其原状态而不需要再生（非破坏性牍出）。SRAM 的存取速度快，但集成度低，功耗较大，所以一般用来组成高速缓冲存储器。

2 .DRAM 的工作原理

与SRAM 的存储原理不同，动态随机存储器(DRAM) 是利用存储元电路中栅极电容上的电荷来存储信息的， DRAM 的基本存储元通常只使用一个晶体管，所以它比SRAM 的密度要高很多。DRAM 采用地址复用技术，地址线是原来的1/2, 且地址信号分行、列两次传送。相对于SRAM 来说， DRAM 具有容易集成、位价低、容量大和功耗低等优点，但DRAM 的存取速度比SRAM 的慢，一般用来组成大容量主存系统。DRAM 电容上的电荷一般只能维持1~2ms, 因此即使电源不断电，信息也会自动消失。为此，每隔一定时间必须刷新，通常取2ms, 这个时间称为刷新周期。常用的刷新方式有3 种：集中刷新、分散刷新和异步刷新。

3.只读存储器(ROM) 的特点

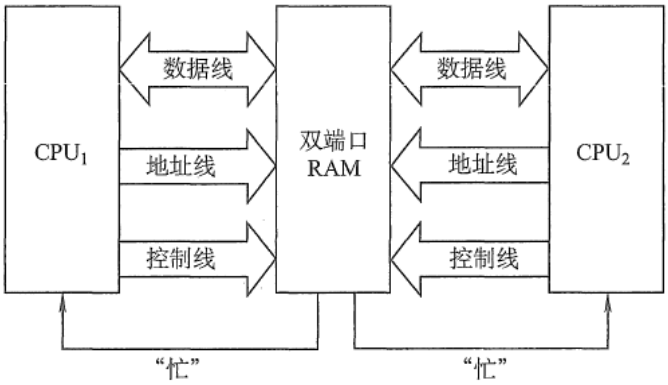
ROM 和RAM 都是支持随机存取的存储器，其中SRAM 和DRAM 均为易失性半导体存储器。而ROM 中一旦有了信息，就不能轻易改变，即使掉电也不会丢失，它在计算机系统中是只供读出的存储器。ROM 器件有两个显著的优点：

1) 结构简单，所以位密度比可读写存储器的高。
2) 具有非易失性，所以可靠性高。

12.有哪些技术能够提高CPU访存速度?

为了提高CPU 访问存储器的速度，可以采用双端口存储器、多模块存储器等技术，它们同属并行技术，前者为空间并行，后者为时间并行。

1.双端口RAM
双端口RAM 是指同一个存储器有左、右两个独立的端口，分别具有两组相互独立的地址线、数据线和读写控制线，允许两个独立的控制器同时异步地访问存储单元，如图所示。当两个端口的地址不相同时，在两个端口上进行读写操作一定不会发生冲突。



2.多模块存储器
为提高访存速度，常采用多模块存储器，常用的有单体多字存储器和多体低位交叉存储器。

注意： CPU 的速度比存储器的快，若同时从存储器中取出n 条指令，就可充分利用CPU 资源，提高运行速度。多体交叉存储器就是基于这种思想提出的。

(1)单体多字存储器

单体多字系统的特点是存储器中只有一个存储体，每个存储单元存储m 个字，总线宽度也为m 个字。一次并行读出m 个字，地址必须顺序排列并处于同一存储单元。单体多字系统在一个存取周期内从同一地址取出m条指令，然后将指令逐条送至CPU执行，即每隔1/m 存取周期， CPU 向主存取一条指令。显然，这增大了存储器的带宽，提高了单体存储器的工作速度。

缺点：指令和数据在主存内必须是连续存放的，一旦遇到转移指令，或操作数不能连续存放，这种方法的效果就不明显。

(2)多休并行存储器
多体并行存储器由多体模块组成。每个模块都有相同的容量和存取速度，各模块都有独立的读写控制电路、地址寄存器和数据寄存器。它们既能并行工作，又能交叉工作。多体并行存储器分为高位交叉编址（顺序方式）和低位交叉编址（交叉方式）两种。

13.Cache

Cache存储器：电脑中为高速缓冲存储器，是位于CPU和主存储器DRAM（Dynamic Random Access Memory）之间，规模较小，但速度很高的存储器，通常由SRAM（Static Random Access Memory静态存储器）组成。

Cache的功能是提高CPU数据输入输出的速率。Cache容量小但速度快，内存速度较低但容量大，通过优化调度算法，系统的性能会大大改善，仿佛其存储系统容量与内存相当而访问速度近似Cache。

Cache通常采用相联存储器。

使用Cache改善系统性能的依据是程序的局部性原理

替换算法：

> 当Cache产生了一次访问未命中之后，相应的数据应同时读入CPU和Cache。但是当Cache已存满数据后，新数据必须替换（淘汰）Cache中的某些旧数据。最常用的替换算法有随机算法、先进先出算法（FIFO）和近期最少使用算法（LRU）。

> 写操作：
>
> 因为需要保证缓存在Cache中的数据与内存中的内容一致，Cache的写操作比较复杂，常用的有写直达法、写回法和标记法。

> 与主存的映射方式：
>
> 直接映射：主存数据块只能装入Cache中的唯一位置
>
> 全相联映射：可以把主存数据块装入Cache 中的任何位置
>
> 组相联映射：将Cache分为若干组，一个数据块可以装入一组内的任何一个位置

14.虚拟存储器

> **虚拟存储器的基本概念**
>
> 虚拟存储器是指具有请求调入和置换功能，能从逻辑上对内存容量加以扩存的一种存储器系统

> **页式虚拟存储器**
>
> 页式管理:是把虚拟存储空间和实际空间等分成固定大小的页,各虚拟页可装入主存中的不同实际页面位置.页式存储中,处理机逻辑地址由虚页号和页内地址两部分组成,实际地址也分为页号和页内地址两部分,由地址映射机构将虚页号转换成主存的实际页号.
>
> 页式管理用一个页表,包括页号,每页在主存中起始位置,装入位等.页表是虚拟页号与物理页号的映射表.页式管理由操作系统进行,对应用程序员的透明的.

> **段式虚拟存储器**
>
> 段式管理: 把主存按段分配的存储管理方式.它是一种模块化的存储管理方式,每个用户程序模块可分到一个段,该程序模块只能访问分配给该模块的段所对应的主存空间.段长可以任意设定,并可放大和缩小.
>
> 系统中通过一个段表指明各段在主存中的位置.段表中包括段名(段号),段起点,装入位和段长等.段表本身也是一个段.段一般是按程序模块分的.

> **段页式虚拟存储器**
>
> 段页式管理:是上述两种方法的结合,它将存储空间按逻辑模块分成段,每段又分成若干个页,访存通过一个段表和若干个页表进行.段的长度必须是页长的整数倍,段的起点必须是某一页的起点.

> **TLB(快表)**
>
> 在虚拟存储器中进行地址变换时,需要虚页号变换成主存中实页号的内部地址变换这一过程
>
> 缓存时首先要到主存查页表，然后才能根据主存物理地址访问主存的存取指令或数据。因此采用虚拟存储器机制后，访存的次数增加了。为了减少访存的次数，往往将页表中最活跃的几个页表项复制到高速缓存中。这种在高速缓存中的页表项称为快表（translation look aside buffer）
>
> 查表时,根据虚页表同时查找快表和慢表,当在快表中查到该虚页号时,就能很快找到对应的实页号,将其送入主存实地址寄存器,同时使慢表的查找作废,这时主存的访问速度没降低多少.如果在快表中查不到,则经过一个访主存的时间延迟后,将从慢表中查到的实页送入实地址寄存器,同时将此虚页号和对应的实页号送入快表。

第四章、指令系统

快速唤起记忆知识框架:

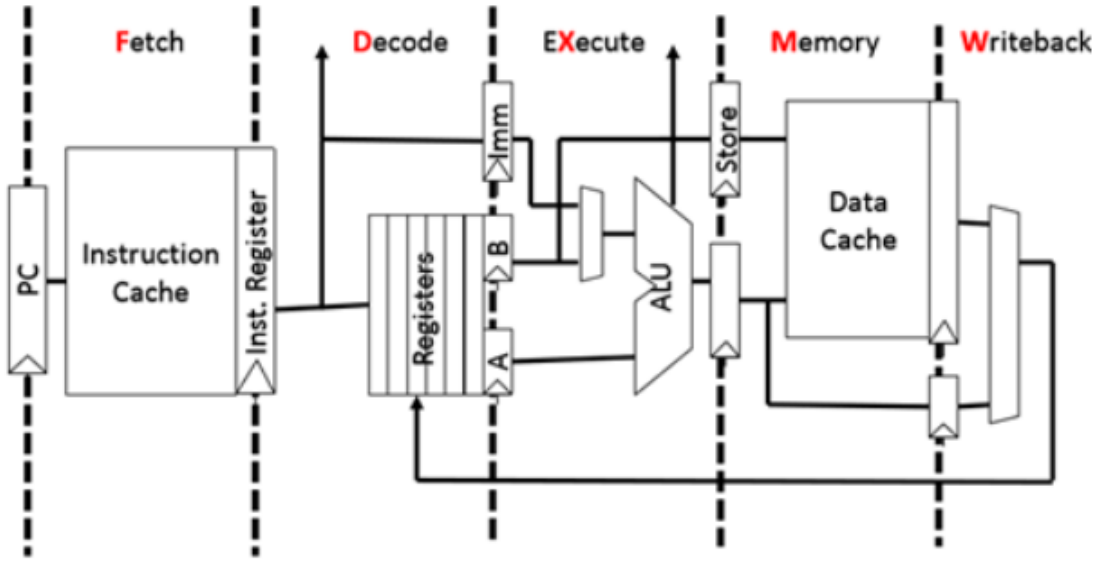
- 指令系统
  - 指令格式
    - 基本格式
      - 操作码：指出指令中该指令应该执行什么性质的操作和具有何种功能
      - 地址码：给出被操作的信息的地址或操作数本身，可以有多个地址码
    - 定长操作码：在指令字的最高位部分分配固定的若干位（定长）表示操作码
    - 扩展操作码：全部指令的操作码字段的位数不固定
  - 寻址方式
    - 指令寻址
      - 顺序寻址：通过程序计数器PC加1，自动形成下一条指令的地址
      - 跳跃寻址：通过转移类指令直接或间接给出下一条指令的地址
    - 数据寻址
      - 隐含寻址：操作数地址不明显给出，隐含在指令中
      - 立即寻址：给出的不是操作数的地址，而是操作数本身
      - 直接寻址：直接给出操作数的真实地址
      - 间接寻址：给出操作数有效地址所在的存储单元的地址
      - 寄存器寻址：直接给出操作数所在的寄存器编号
      - 寄存器间接寻址：给出存有操作数所在主存单元的地址的寄存器编号
      - 相对寻址：把程序计数器PC的内容加上指令格式中的形式地址
      - 基址寻址：将基址寄存器的内容加上指令格式中的形式地址
      - 变址寻址：将变址寄存器的内容加上指令格式中的形式地址
      - 堆栈寻址：从规定的堆栈中取出操作数
  - CISC：指令数目多、字长不固定、寻址方式多、寄存器数量少、一般为微程序控制
  - RISC：指令数目少、字长固定、寻址方式少、寄存器数量多、一般为组合逻辑控制

15.指令流水线的基本概念

> **流水线基本原理：**

> 流水线技术是一种显著提高指令执行速度与效率的技术。方法是:指令取指完成后,不等该指令执行完毕即可取下一条指令。如果把一条指令的解释过程进一步细分,例如分成**取指,译码, 访存， 执行, 和写回**五个子过程,并用五个子部件分别处理这五个子过程.这样只需在上一指令的第一子过程处理完毕进入第二子过程处理时,在第一子部件中就开始对第二条指令的第一子过程进行处理.随着时间推移,这种重叠操作最后可达到五个子部件同时对五条指令的子过程进行操作.

> **典型的五级流水线的数据通路：**



> **流水线方式的特点:**

> 与传统的串行执行方式相比，采用流水线方式具有如下特点：

> 1) 把一个任务（一条指令或一个操作）分解为几个有联系的子任务，每个子任务由一个专门的功能部件来执行，并依靠多个功能部件并行工作来缩短程序的执行时间。

> 2) 流水线每个功能段部件后面都要有一个缓冲寄存器，或称锁存器，其作用是保存本流水段的执行结果，供给下一流水段使用。
>
> 3) 流水线中各功能段的时间应尽量相等，否则将引起堵塞、断流。
>
> 4) 只有连续不断地提供同一种任务时才能发挥流水线的效率，所以在流水线中处理的必须是连续任务。在采用流水线方式工作的处理机中，要在软件和硬件设计等多方面尽量为流水线提供连续的任务。
>
> 5) 流水线需要有装入时间和排空时间。装入时间是指第一个任务进入流水线到输出流水线的时间。排空时间是指最后一个任务进入流水线到输出流水线的时间。

> **影响流水线性能的因素**

> 1）结构相关是当多条指令同一时刻争用同一资源形成冲突

> 解决方案：（1）暂停一个时钟周期（2）单独设置数据存储器和指令存储器

> 2）数据相关是指令在流水线中重叠执行时,当后继指令需要用到前面指令的执行结果时发生的.

> 解决方案：（2）暂停一个时钟周期（2）数据旁路：把前一条指令的ALU计算结果直接输入到下一条指令

> 3）控制相关是当流水线遇到分支指令和其他改变PC值的指令时引起的.
>
> 解决方案：(1)延迟转移技术。将转移指令与其前面的与转移指令无关的一条或几条指令对换位置，让成功转移总是在紧跟的指令被执行之后发生，从而使预取的指令不作废。
>
> (2)转移预测技术。

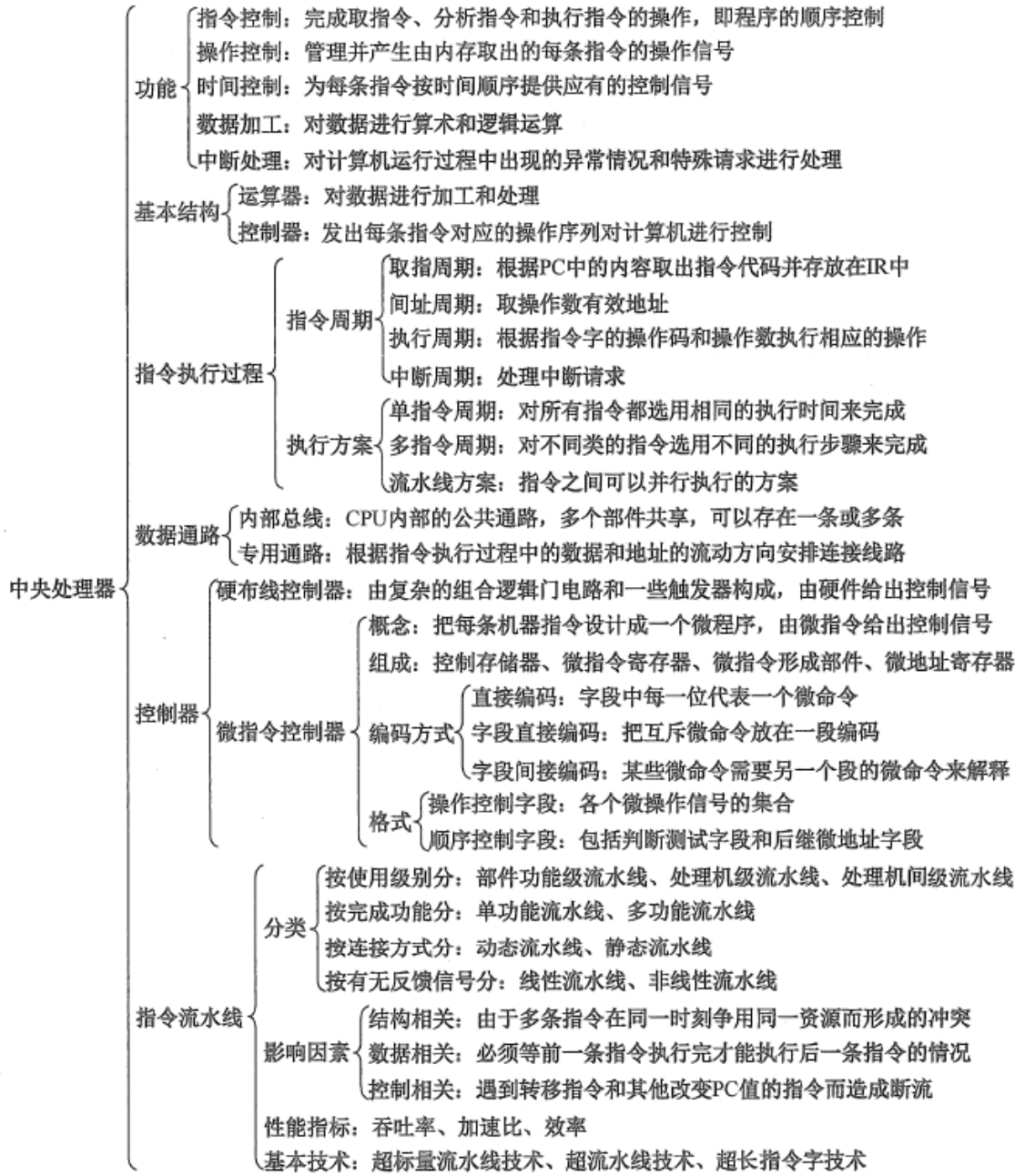
16.CISC和RISC（复杂指令集和精简指令集）的对比?

类别 / 对比项目	CISC	RISC
指令系统	复杂，庞大	简单，精简
指令数目	一般大于 200 条	一般小于 100 条
指令字长	不固定	定长
可访存指令	不加限制	只有 Load/Store 指令
各种指令执行时间	相差较大	绝大多数在一个周期内完成
各种指令使用频度	相差很大	都比较常用
通用寄存器数量	较少	多
目标代码	难以用优化编译生成高效的目标代码程序	采用优化的编译程序，生成代码较为高效
控制方式	绝大多数为微程序控制	绝大多数为组合逻辑控制
指令流水线	可以通过一定方式实现	必须实现

17.寻址方式在本章知识框架

第五章、中央处理器

快速唤起记忆知识框架：



18.CPU 的功能?

> 中央处理器(CPU) 由运算器和控制器组成。其中，控制器的功能是负责协调并控制计算机各部件执行程序的指令序列，包括取指令、分析指令和执行指令；运算器的功能是对数据进行加工。
> CPU 的具体功能包括:
>
> 1) 指令控制。完成取指令、分析指令和执行指令的操作，即程序的顺序控制。
> 2) 操作控制。一条指令的功能往往由若干操作信号的组合来实现。CPU 管理并产生由内存取出的每条指令的操作信号，把各种操作信号送往相应的部件，从而控制这些部件按指令的要求进行动作。
> 3) 时间控制。对各种操作加以时间上的控制。时间控制要为每条指令按时间顺序提供应有的控制信号。
> 4) 数据加工。对数据进行算术和逻辑运算。
> 5) 中断处理。对计算机运行过程中出现的异常情况和特殊请求进行处理。

19.流水线越多，并行度就越高。是否流水段越多，指令执行越快?

> 错误，原因如下:
> 1) 流水段缓冲之间的额外开销增大。每个流水段有一些额外开销用于缓冲间传送数据、进行各种准备和发送等功能，这些开销加长了一条指令的整个执行时间，当指令间逻辑上相互依赖时，开销更大。

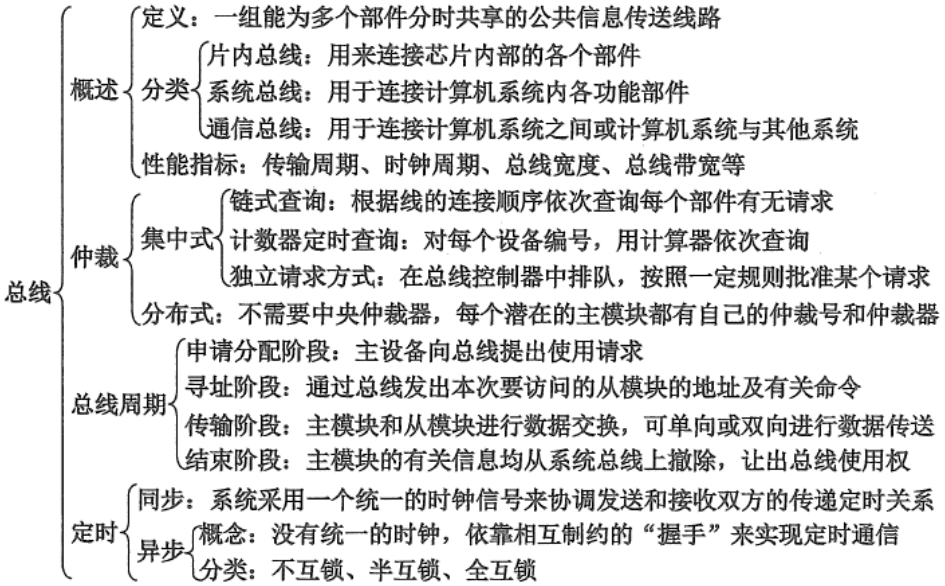
> 2) 流水段间控制逻辑变多、变复杂。用于流水线优化和存储器（或寄存器）冲突处理的控制逻辑将随流水段的增加而大增，这可能导致用于流水段之间控制的逻辑比段本身的控制逻辑更复杂。

20.有关指令相关、数据相关的几个概念

> 1) 两条连续的指令读取相同的寄存器时，会产生读后读(Read After Read, RAR) 相关，这种相关不会影响流水线。
>
> 2) 某条指令要读取上一条指令所写入的寄存器时，会产生写后读(Read After Write, RAW)相关，它称数据相关或真相关，影响流水线。按序流动的流水线只可能出现RAW 相关。
>
> 3) 某条指令的上条指令要读 / 写该指令的输出寄存器时，会产生读后写(Write After Read,WAR)和写后写(Write After Write, WAW) 相关。在非按序流动的流水线中，既可能发生RAW 相关，又可能发生WAR 相关和WAW 相关。
>
> 对流水线影响最严重的指令相关是数据相关。

第六章、总线

快速唤起记忆知识框架：



21.引入总线结构有什么好处?

> 引入总线结构主要有以下优点：
> 1）简化了系统结构，便于系统设计制造。
> 2）大大减少了连线数目，便于布线，减小体积，提高系统的可靠性。
> 3）便于接口设计，所有与总线连接的设备均采用类似的接口。
> 4）便于系统的扩充、更新与灵活配置，易千实现系统的模块化。
> 5）便于设备的软件设计，所有接口的软件对不同的接口地址进行操作。
> 6）便于故障诊断和维修，同时也能降低成本。

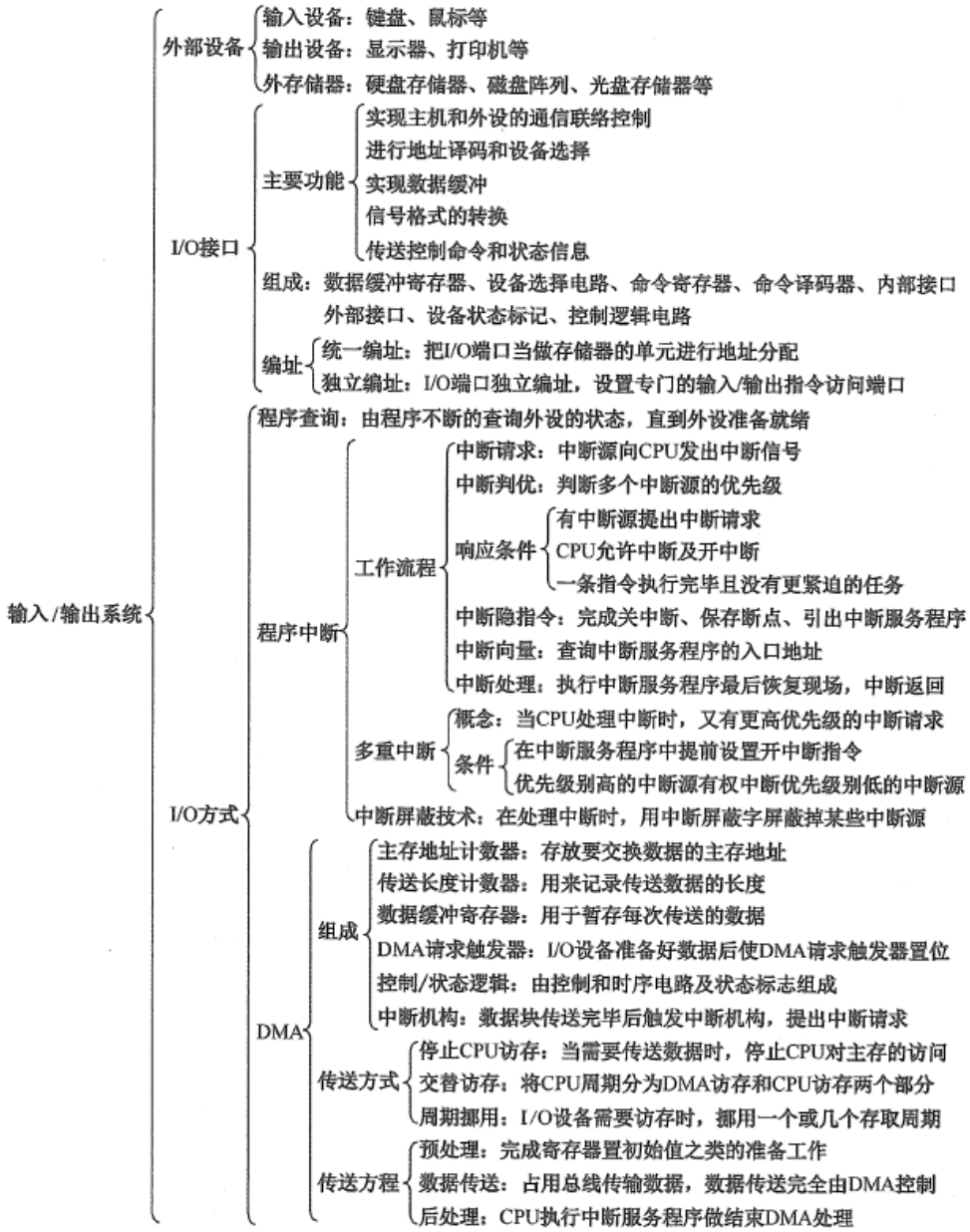
22、总线相关概念

> 1、系统总线按照传输信息的不同，分成哪几类？是单向的，还是双向的?
> 1）分成数据总线、地址总线以及控制总线。
> 2）数据总线：各个功能部件之间传送数据信息，双向传输；
> 3）地址总线：用来指明数据总线上，源数据或目的数据所在的主存单元的地址。单向：由CPU发出
> 4）控制总线：用来发送各种控制信号。对于控制总线中的单根线，是单向的，即只能由一个部件发向另一　个部件。而一组控制总线中，有输入也有输出，因此，控制总线也可以看成是双向的。

> 2、什么是总线宽度、总线带宽、总线复用、信号线数?
> 1）总线宽度：数据总线的根数，一般是8的倍数。是衡量计算机系统性能的重要指标；
> 2）总线带宽：即总线数据传输速率，总线上每秒能够传输的最大字节量。
> 3）总线复用：一条信号线上分时传送两种信号。例如数据总线和地址总线的分时复用；
> 4）信号线数：地址总线、数据总线和控制总线三种总线的线数之和。

第七章、输入输出系统

快速唤起记忆知识框架：



23.CPU 响应中断应具备哪些条件?

> 1)在CPU 内部设置的中断屏蔽触发器必须是开放的。
>
> 2)外设有中断请求时，中断请求触发器必须处于"1" 状态，保持中断请求信号。
>
> 3)外设（接口）中断允许触发器必须为"1"'这样才能把外设中断请求送至CPU 。
>
> 具备上述三个条件时， CPU 在现行指令结束的最后一个状态周期响应中断。

24.中断响应优先级和中断处理优先级分别指什么?

> 中断响应优先级是由硬件排队线路或中断查询程序的查询顺序决定的，不可动态改变；而中断处理优先级可以由中断屏蔽字来改变，反映的是正在处理的中断是否比新发生的中断的处理优先级低（屏蔽位为"0", 对新中断开放），若是，则中止正在处理的中断，转到新中断去处理，处理完后再回到刚才被中止的中断继续处理。

25.向量中断、中断向量、向量地址三个概念是什么关系?

> 1)中断向量：每个中断源都有对应的处理程序，这个处理程序称为中断服务程序，其入口地址称为中断向量。所有中断的中断服务程序入口地址构成一个表，称为中断向量表；也有的机器把中断服务程序入口的跳转指令构成一张表，称为中断向量跳转表。
>
> 2)向量地址：中断向量表或中断向量跳转表中每个表项所在的内存地址或表项的索引值，称为向量地址或中断类型号。
>
> 3)向量中断：指一种识别中断源的技术或方式。识别中断源的目的是找到中断源对应的中断服务程序的入口地址的地址，即获得向量地址。

26.程序中断和调用子程序有何区别?

> 两者的根本区别主要表现在服务时间和服务对象上不一样。
>
> 1) 调用子程序过程发生的时间是已知的和固定的，即在主程序中的调用指令(CALL) 执行时发生主程序调用子程序过程，调用指令所在位置是已知的和固定的。而中断过程发生的时间一般是随机的， CPU 在执行某个主程序时收到中断源提出的中断申请，就发生中断过程，而中断申请一般由硬件电路产生，申请提出时间是随机的。也可以说，调用子程序是程序设计者事先安排的，而执行中断服务程序是由系统工作环境随机决的。
>
> 2) 子程序完全为主程序服务，两者属于主从关系。主程序需要子程序时就去调用子程序，并把调用结果带回主程序继续执行。而中断服务程序与主程序二者一般是无关的，不存在谁为谁服务的问题，两者是平行关系。
>
> 3) 主程序调用子程序的过程完全属千软件处理过程，不需要专门的硬件电路；而中断处理系统是一个软 / 硬件结合的系统，需要专门的硬件电路才能完成中断处理的过程。
>
> 4) 子程序嵌套可实现若干级，嵌套的最多级数受计算机内存开辟的堆栈大小限制；而中断嵌套级数主要由中断优先级来决定，一般优先级数不会很大。

27.I/O控制方式在操作系统篇有了，这里就不重复了。