第1章 计算机系统概述

**1.1 计算机发展历程**

1）第一代计算机（1946-1957年）——**电子管时代**：逻辑元件采用电子管，使用机器语言进行编程  
2）第二代计算机（1958-1964年）——**晶体管时代**：逻辑元件采用晶体管，软件开始使用高级语言，如Fortran  
3）第三代计算机（1965-1971年）——**中小规模集成电路时代**：逻辑元件采用中小规模集成电路，开始有了分时操作系统  
4）第四代计算机（1972-现在）——**超大规模集成电路时代**：逻辑元件采用大规模集成电路和超大规模集成电路，并产生了微处理器

摩尔定律  
32位、64位指的是机器字长，是指计算机进行一次整数运算所能处理的二进制数据的位数。

电子计算机可分为电子模拟计算机和电子数字计算机。数字计算机又可按用途分为专用计算机和通用计算机。通用计算机又分为巨型机、大型机、中型机、小型机、微型机和单片机6类。

计算机按指令和数据流还可分为：  
1）**单指令流和单数据流系统（SISD），也即冯诺依曼体系结构**  
2）单指令流和多数据流系统（SIMD），包括阵列处理器和向量处理器系统  
3）**多指令流和单数据流系统（MISD），这种计算机实际上不存在**  
4）多指令流和多数据流系统（MIMD），包括多处理器和多计算机系统

**1.2 计算机系统层次结构**

硬件系统和软件系统共同构成了一个完整的计算机系统。对于某一个功能来说，其既可以用软件实现，也可以用硬件实现，则称为软硬件在逻辑上是等效的。

早期的冯诺依曼机的特点：  
1）计算机硬件系统由**运算器、存储器、控制器、输入设备、输出设备**5大部件组成  
2）指令和数据以同等地位存于存储器内，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  
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

现代计算机已经发展为**以存储器为中心**，使I/O操作尽可能地绕过CPU，直接在I/O设备和存储器之间完成，以提高系统的整体运行效率。

**输入设备**：键盘、鼠标、扫描仪、摄像机等等  
**输出设备**：显示器、打印机  
**存储器**：主存储器和辅助存储器  
主存储器的工作方式是按存储单元的地址进行存取的，称为按地址存取方式。（相联存储器是按内容访问的）  
**地址寄存器MAR用于寻址，其位数对应着存储单元的个数，MAR的长度与PC的长度相等。  
数据寄存器MDR和存储字长相等。**  
MAR和MDR的位数分别为（地址码长度、存储字长）  
**注意：MAR和MDR虽然是存储器的一部分，但在现代CPU中却是存在于CPU中的，另外后文提到的高速缓存Cache也是存在于CPU中。**  
**运算器**：算术逻辑单元ALU、通用寄存器（累加器ACC、乘商寄存器MQ、操作数寄存器X、变址寄存器IX、基址寄存器BR）、程序状态寄存器PSW  
**控制器**：程序计数器PC、指令寄存器IR、控制单元CU

**系统软件、应用软件**  
注意：数据库管理系统DBMS和数据库系统DBS是有区别的，**DBMS是系统软件**，而DBS一般由数据库、数据库管理系统、数据库管理员DBA和应用系统构成

**机器语言是计算机唯一可以直接识别和执行的语言**；  
汇编语言的程序必须经过一个称为**汇编程序**的系统软件的翻译，将其转换为计算机的机器语言后，才能在计算机的硬件系统上执行；  
高级语言需要经过**编译程序编译**成汇编语言程序，然后经过**汇编操作**得到机器语言程序，或者**直接由高级语言程序翻译成机器语言程序**。

**注意：(PC)指程序计数器PC中存放的内容，括号不能省略，即(PC)+1->PC不能写成PC+1->PC**

**1.3 计算机的性能指标**

**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**，一般等于内部寄存器的大小，字长越长，数的表示范围越大，计算精度就越高。

**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。

吞吐量：系统吞吐量主要取决于主存的存取周期  
响应时间：通常包括CPU时间（运行一个程序所花费的时间）与等待时间  
**CPU时钟周期**：即主频的倒数，它是CPU中最小的时间单位  
**主频（CPU时钟频率）**：主频的倒数是CPU时钟周期，对于同一个型号的计算机，其主频越高，完成指令的一个执行步骤所用的时间越短，执行指令的速度越快。  
**CPI**：执行一条指令所需的时钟周期数  
CPU执行时间，指运行一个程序所花费的时间

**CPU执行时间=CPU时钟周期数/主频=（指令条数\*CPI）/主频  
CPU的性能（CPU执行时间）取决于三个要素：主频，每条指令执行所用的时钟周期数（CPI）、指令条数**

MIPS：每秒执行多少百万条指令。MIPS=指令条数/（执行时间\*）=主频/CPI  
MFLOPS：每秒执行多少百万次浮点计算，  
GFLOPS：每秒执行多少十亿次浮点计算，  
TFLOPS：每秒执行多少万亿次浮点计算，

PFLOPS：每秒执行多少一千万亿次浮点计算，

EFLOPS：每秒执行多少一百京次浮点计算，

ZFLOPS：每秒执行多少十万京次浮点计算，

**1.5 常见问题和易混淆知识点**

**在逻辑功能上，软件和硬件是等效的**。在计算机系统中，许多功能既可以由硬件直接实现，也可以在硬件的配合下由软件实现。

**翻译程序**是指把高级语言源程序翻译成机器语言程序（目标代码）的软件，可以分为两种：一种是**编译程序**，将高级语言源程序一次全部翻译成目标程序，只要源程序不变，就无须重新编译，目标程序与体系结构相关，但仍不是计算机硬件能够直接执行的程序；另一种是**解释程序**，翻译一句执行一句，并且不会生成目标程序。  
**汇编程序**把汇编语言源程序翻译成机器语言程序。

在计算机领域中，站在某一类用户的角度，如果感觉不到某个事物或属性的存在，即“看”不到某个事物或属性，则称为“对该用户而言，某个事物或属性是透明的”。**在CPU中，IR、MAR和MDR对各类程序员都是透明的。**

**机器字长：计算机能直接处理的二进制数据的位数，机器字长一般等于内部寄存器的大小，它决定了计算机的运算精度。  
指令字长：一个指令字中包含二进制代码的位数。  
存储字长：一个存储单元存储二进制代码的长度。**

第2章 数据的表示和运算

**2.1 数制与编码**

**二进制转换为八进制和十六进制**：以小数点为界，其整数部分，从小数点开始往左数，将一串二进制数分为3位（八进制）一组或4位（十六进制）一组，在数的最左边根据需要加“0”补齐；对于小数部分，从小数点开始往右数，将一串二进制数分为3位（八进制）一组或4位（十六进制）一组，在数的最右边根据需要加“0”补齐。  
**八进制或十六进制转换为二进制**：略。  
**任意进制转换为十进制**：略  
**十进制转换为任意进制**：基数乘除法。对于整数部分用除基取余法；对于小数部分用乘基取整法。

**注意：并不是每一个十进制小数都可以准确地用二进制表示，但任意一个二进制小数都可以用十进制小数表示。**

**真值**

**机器数：**

#机器数题目不说默认是补码，特别是对于负数。

二进制编码的十进制数（BCD码）：  
**8421码**：设其各位的数值为b3、b2、b1、b0，则权值从高到低依次为8/4/2/1，则它表示的十进制数为D=8b3+4b2+2b1+1b0。如果两个8421码相加之和小于等于(1001)，即(9)，则不需要修正；**如果相加之和大于等于(1010)，即(10)，则要加6修正**。  
**余3码**：在8421码的基础上加上(0011)形成的，即每个数都多余“3”  
**2421码**：权值由高到低分别为2/4/2/1，特点是大于等于5的4位二进制数中最高位为1，小于5的最高位为0。如5为1011而不是0101。

**奇偶校验码**：在原编码上加一个校验位，它的码距等于2，**可以检测出一位错误（或奇数位错误），但不能确定出错的位置，也不能够检测出偶数位错误**，增加的冗余位称为奇偶校验位。  
奇校验码：整个校验码（有效信息位和校验位）中“1”的个数为奇数  
偶校验码：整个校验码（有效信息位和校验位）中“1”的个数为偶数

**海明（汉明）校验码：不但可以发现错位，还能指出错位的位置，为自动纠错提供了依据。**  
L-1=D+C且D≥C，即编码最小码距L越大，则其检测错误的位数D越大，纠正错误的位数C也越大，且纠错能力恒小于或等于检错能力。  
**信息位n和校验位k应满足n+k≤2^k-1**  
求海明码的步骤（略）

**循环冗余校验码（CRC）**：在K为信息码后再拼接R位的校验码，整个编码的长度为N位

**2.2 定点数的表示与运算**

**原码表示法**：用机器数的最高位表示该数的符号，其余的各位表示数的绝对值  
**纯小数的原码定义**：若字长为n+1，则原码小数的表示范围为-(1-2^-n)≤x≤1-2 ^-n  
[x] = x，1>x≥0  
[x] = 1-x = 1+|x|，0≥x>-1  
**纯整数的原码定义**：若字长为n+1，则原码小数的表示范围为-(2^n-1)≤x≤2 ^n-1  
[x] = 0,x，2^n>x≥0  
[x] = 2^n-x = 2 ^n+|x|，0≥x>-2 ^n  
**注意：真值零的原码表示有正零和负零两种形式：[+0]=00000, [-0]=10000**

**补码表示法**：  
**纯小数的补码定义**：若字长为n+1，则原码小数的表示范围为-1≤x≤1-2 ^-n  
[x] = x，1>x≥0  
[x] = 2+x = 2-|x|，0≥x>-1  
**纯整数的补码定义**：若字长为n+1，则原码小数的表示范围为-2^n≤x≤2 ^n-1  
[x] = 0,x，2^n>x≥0  
[x] = 2^(n+1)+x = 2 ^(n+1)-|x|，0≥x≥-2 ^n  
**注意：真值零的补码表示是唯一的：[+0]=[-0]=0.0000**

**由原码求补码、由补码求原码：  
对于正数，补码与原码的表示相同。  
对于负数，原码符号位不变，数值部分按位取反，末位加1.**

**反码表示法**：  
**纯小数的原码定义**：若字长为n+1，则原码小数的表示范围为-(1-2^-n)≤x≤1-2 ^-n  
[x] = x，1>x≥0  
[x] = 2-2^-n +x，0≥x>-1  
**纯整数的原码定义**：若字长为n+1，则原码小数的表示范围为-(2^n-1)≤x≤2 ^n-1  
[x] = 0,x，2^n>x≥0  
[x] = 2^(n+1)-1+x，0≥x>-2 ^n  
**注意：真值零的原码表示有正零和负零两种形式：[+0]=0.0000, [-0]=1.1111**

**移码表示法**：移码常用来表示浮点数的阶码，它只能表示整数。  
[x] = 2^n+x（其中2 ^n>x≥-2 ^n，机器字长为n+1）  
**注意**：  
1）移码中零的表示唯一，[+0]=2^n+0=[-0]=2 ^n-0=100…0（n个0）  
2）一个真值的移码和补码仅差一个符号位，补码的符号位取反即得移码（1表示正，0表示负，这与其他机器数的符号位取值正好相反）  
3）移码全0时，对应真值的最小值-2^n；移码全1时，对应真值的最大值2 ^n -1  
4）移码大真值就大，移码小真值就小

**算术移位：移位对象是有符号数**

|  | **码制** | **添加代码** |
| --- | --- | --- |
| 正数 | 原码、补码、反码 | 0 |
| 负数 | 原码 | 0 |
| 负数 | 补码 | 左移添0，右移添1 |
| 负数 | 反码 | 1 |

**逻辑移位：将操作数当做无符号数看待，逻辑左移时，高位移丢，低位添0；逻辑右移时，低位移丢，高位添0.**  
注意：逻辑移位不管左移还是右移，都是添0。

**循环移位：**

**溢出**是指运算结果超过了数的表示范围，称大于机器所能表示的最大正数为**上溢**，小于机器所能表示的最小负数为**下溢**。  
**仅当两个符号相同的数相加，或两个符号相异的数相减才可能产生溢出。**

补码定点数加减运算溢出判断的方法有3种：  
1）**采用一位符号位**：只要参加运算的两个数符号相同，结果又与原操作数符号不同，则表示结果溢出。  
2）**采用双符号位**：运算结果的两个符号位相同，表示未溢出；运算结果的两个符号位不同，表示溢出，此时最高位符号位代表真正的符号。  
00，表示结果为正数，无溢出；  
01，表示结果正溢出；  
10，表示结果负溢出；  
11，表示结果为负数，无溢出。

3）**采用一位符号位根据数据位的进位情况判断溢出**：如果符号位的进位与最高数位的进位相同，说明没有溢出，否则表示发生溢出。

**小端模式：**先存储低位字节、后存储高位字节的顺序（即从低位字节向高位字节顺序）存放字符串的内容

**大端模式：**先存储高位字节、后存储低位字节的顺序（即从高位字节向低位字节顺序）存放字符串的内容

**2.3 浮点数的表示与运算**

IEEE754标准中，规格化的短浮点数的真值为(-1)^S*1.M*2 ^(E-127)  
规格化长浮点数的真值为(-1)^S*1.M*2 ^(E-1023)  
其中，s=0表示正数，s=1表示负数；短浮点数E的取值为1~254（8位表示），M为23位，共32位；长浮点数E的取值为1 ~2046（11位表示），M为52位，共64位。

**浮点数的加减运算：  
1）对阶  
2）尾数求和  
3）规格化  
4）舍入  
5）溢出判断**

**2.4 算术逻辑单元ALU**

运算器由算术逻辑单元ALU、累加器、状态寄存器和通用寄存器等组成。运算器的操作和操作种类由控制器决定，运算器处理的数据来自存储器，处理后的结果数据通常送回存储器，或暂存在运算器中。

**一位全加器**：  
**和表达式**：Si=Ai⊕Bi⊕Ci-1（Ai、Bi、Ci-1有奇数个1，则Si=1，否则Si=0）  
**进位表达式**：Ci=AiBi+(Ai⊕Bi)Ci-1

串行加法器：只有一个全加器  
**并行加法器**：由多个全加器组成，提高并行加法器速度的关键是尽量加快进位产生和传递的速度。  
**进位表达式**：Ci=Gi+PiCi-1，Gi是进位产生函数，Gi=AiBi；Pi是进位传递函数，Pi=Ai⊕Bi  
并行加法器的进位通常分为串行进位和并行进位  
分组并行进位方式：  
**单级先行进位方式又称为组内并行、组间串行进位方式；多组先行进位方式又称为组内并行、组间并行进位方式。**

第3章 存储系统

**3.1 存储器的概述**

按在计算机中的作用（层次）分类：**主存储器、辅助存储器、高速缓冲存储器Cache**  
按存储介质分类：**磁表面存储器**（磁盘、磁带）、**磁芯存储器半导体存储器**（MOS型存储器、双极型存储器）、**光存储器**（光盘）  
按存取方式分类：**随机存储器**（静态RAM、动态RAM）、**只读存储器**（ROM）、**串行访问存储器**（顺序存取存储器（磁带）、直接存取存储器（磁盘））  
按信息的可保存性分类：**易失性存储器**（RAM）、**非易失性存储器**（ROM、磁表面存储器、光存储器）

**存储容量=存储字数\*字长，存储字数表示存储器的地址空间大小，字长表示一次存取操作的数据量**

**存储速度：数据传输率=数据的宽度/存储周期**  
**存取时间Ta**：指从启动一次存储器操作到完成该操作所经历的时间，分为读出时间和写入时间  
**存取周期Tm**：指存储器进行一次完整的读写操作所需的全部时间，即连续两次独立地访问存储器操作（读或写操作）之间所需的最小时间间隔  
**主存带宽Bm**：又称数据传输率，表示每秒从主存进出信息的最大数量  
**存取时间不等于存储周期，通常存储周期大于存取时间**。

**3.2 存储器的层次化结构**

**“Cache-主存”：解决CPU和主存速度不匹配的问题  
“主存-辅存”：解决存储系统的容量问题**

**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、辅存都嗯呢该交换信息**

**注意**：**主存和Cache**之间的数据调动是由**硬件**自动完成的，**对所有程序员均是透明的**；而**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，**对应用程序员是透明的**。  
Cache中的内容只是主存中内容的一部分，主存中的内容只是辅存中内容的一部分

**3.3 半导体随机存储器**

1）存储矩阵  
2）译码驱动  
3）读写电路  
4）读写控制线  
**5）片选线：确定哪个存储芯片被选中  
6）地址线：单向输入  
7）数据线：双向的，数据线数和地址线数共同反映存储芯片容量的大小。如地址线10根，数据线8根，则芯片容量=2^10\*8=8K位**

**静态随机存储器SRAM**：用双稳态触发器（六管MOS）来记忆信息，属于易失性半导体存储器，一般用来组成高速缓冲存储器  
**动态随机存储器DRAM**：利用存储元电路中栅极电容上的电荷来存储信息，一般用来组成大容量主存系统。刷新方式有三种：集中刷新、分散刷新、异步刷新

|  | **SRAM** | **DRAM** |
| --- | --- | --- |
| 存储信息 | 触发器 | 电容 |
| 破坏性读出 | 非 | 是 |
| 需要刷新 | 不要 | 需要 |
| 送行列地址 | 同时送 | 分两次送 |
| 运行速度 | 快 | 慢 |
| 集成度 | 低 | 高 |
| 发热量（功耗） | 大 | 小 |
| 存储成本 | 高 | 低 |
| 主要用途 | 高速缓存 | 主机内存 |

**只读存储器（ROM）**

掩膜式只读存储器MROM：写入以后任何人都无法改变其内容  
一次可编程只读存储器PROM：允许用户用专门设备写入程序，写入后内容就无法改变  
可擦除可编程只读存储器EPROM：允许用户写入信息，而且可以对其内容进行多次改写  
闪速存储器Flash Memory：在不加电时仍可长期保存信息且能进行快速擦除重写  
固态硬盘SSD：用固态电子存储芯片阵列而制成的硬盘，由控制单元和存储单元（Flash芯片）组成

**3.4 主存储器与CPU的连接**

1）主存储器通过数据总线、地址总线和控制总线与CPU连接  
2）数据总线的位数与工作频率的乘积正比于数据传输率  
3）地址总线的位数决定了可寻址的最大内存空间  
4）控制总线（读/写）指出总线周期的类型和本次输入/输出操作完成的时刻

主存容量的扩展：**位扩展法**、**字扩展法**、**字位同时扩展法**

CPU要实现对存储单元的访问，首先要选择存储芯片，即进行片选，然后再为选中的芯片依地址码选择相应的存储单元，以进行数据的存取，即进行字选。  
片选信号的产生分为**线选法**和**译码片选法**。

**3.5 双口RAM和多模块存储器**

**双端口RAM**：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。  
1）两个端口不同时对同一地址单元存取数据（正常）  
2）两个端口同时对同一地址单元读出数据（正常）  
3）两个端口同时对同一地址单元写入数据（可能会写入错误）  
4）两个端口同时对同一地址单元，一个写入数据，另一个读出数据（出现读出错误）

**多模块存储器**：  
**单体多字存储器**：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  
**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，既能并行工作，又能交叉工作  
多体并行存储器由分为高位交叉编址（顺序方式）和低位交叉编址（交叉方式）  
**高位交叉编址**：高位地址表示体号，低位地址为体内地址  
**低位交叉编址**：低位地址为体号，高位地址为体内地址  
流水线方式，存储器交叉模块数应大于等于**m=T/r**（T为存取周期，r为总线传送周期），连续读取m个字所需时间为**t1=T+(m-1)r**，而顺序方式连续读取m个字所需时间为t2=mT

**3.6 高速缓冲存储器**

高速缓冲技术就是利用程序访问的**局部性原理**。  
当CPU发出读请求时，如果访存地址在Cache中命中，就将此地址转换成Cache地址，直接对Cache进行读操作，与主存无关；如果Cache不命中，则仍需访问主存，并把此字所在的块一次从主存调入Cache内。若此时Cache已满，则需根据某种替换算法，用这个块替换掉Cache中原来的某块信息。  
**注意：CPU与Cache之间的数据交换以字位单位，而Cache与主存之间的数据交换以Cache块为单位**

**Cache命中率**

**平均访问时间**

**Cache和主存的映射方式（3种）**

**直接映射**：j = i mod 2^c（j是Cache的块号，又称行号，i是主存的块号，2 ^c是Cache的总块数）  
地址结构为：

| **主存字块标记** | **Cache字块地址（行号）** | **字块内地址** |
| --- | --- | --- |

**全相联映射**：  
地址结构为：

| **主存字块标记** | **字块内地址** |
| --- | --- |

**组相联映射**：将Cache空间分成大小相同的组，主存的一个数据块可以装入到一组内的任何一个位置，即组间采取直接映射，而组内采取全相联映射。  
j = i mod Q（j是Cache的组号，i是主存的块号，Q是Cache的组数，Q=1变为全相联映射，Q=Cache块数变为直接映射）  
地址结构为：

| **主存字块标记** | **组地址（组号）** | **字块内地址** |
| --- | --- | --- |

**置换算法**：  
1）随机算法RAND：没有依据程序访问的局部性原理  
2）先进先出算法FIFO：没有依据程序访问的局部性原理  
3）近期最少使用算法LRU  
4）最不经常使用算法LFU

**Cache写策略**  
**对于Cache写命中**：  
1）**全写法**（写直通法write-through）：必须把数据同时写入Cache和主存  
2）**写回法**（write-back）：只修改Cache的内容，而不立即写入主存，只有当此块被换出时才写回主存  
**对于Cache写不命中**：  
1）**写分配法**（write-allocate）：加载主存中的块到Cache中，然后更新这个Cache块  
2）**非写分配法**（not-write-allocate）：只写入主存，不进行调块  
**非写分配法通常与全写法合用，写分配法通常和写回法合用。**

**3.7 虚拟存储器**

主机和联机工作的辅存共同构成了虚拟存储器，对于应用程序而言，**虚拟存储器是透明的**。

**页式虚拟存储器**：以页为基本单位，虚拟空间和主存空间都被划分成同样大小的页，主存的页称为实页，虚存的页称为虚页。**把虚拟地址分为两个字段：虚页号和页内地址**。  
**优点**是页面的长度固定，页表简单，调入方便。**缺点**是由于程序不可能正好是页面的整数倍，最后一页的零头将无法利用而造成浪费，并且页不是逻辑上独立的实体，所以处理、保护和共享都不及段式虚拟存储器方便。

**段式虚拟存储器**：段是按程序的逻辑结构划分的，各个段的长度因程序而异。**把虚拟地址分为两部分：段号和段内地址**。  
**优点**是段的分界与程序的自然分界相对应，因而具有逻辑独立性，使它易于编译、管理、修改和保护，也便于多道程序的共享；**缺点**是因为段长度可变，分配空间不便，容易在段间留下碎片，不好利用，造成浪费。

**段页式虚拟存储器**：把程序按逻辑结构分段，每段再划分为固定大小的页，主存空间也划分为大小相等的页，程序对主存的调入、调出仍以页为基本传送单位。每个程序对应一个段表，每段对应一个页表，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。**虚地址分为段号、段内页号、页内地址三部分**。  
**优点**是兼具页式和段式虚拟存储器的优点，可以按段实现共享和保护。**缺点**是在地址变换过程中需要两次查表，系统开销比较大。

**快表TLB**依据程序执行的局部性原理，快表只是慢表Page的一个副本，而且只存放了慢表中很少的一部分。  
查找时，快表和慢表同时进行，快表由于根据内容指定地址，一般使用相联存储器。  
**注意：TLB是Page的一个很小的副本，所以若TLB命中则Page一定命中。在同时具有虚拟页式存储器（有TLB）和Cache的系统中，访问顺序为TLB->页表->Cache->主存。TLB，Page，Cache命中情况组合共有五种，分别为111，110，011，010，000。**

**虚拟存储器和Cache的不同之处**：  
1）Cache主要解决系统速度，而虚拟存储器是为了解决主存容量  
2）Cache全由硬件实现，是硬件存储器，对所有程序员透明；而虚拟存储器由OS和硬件共同实现，是逻辑上的存储器，对系统程序员不透明，但对应用程序员透明。  
3）虚拟存储器系统不命中时对系统性能影响更大。  
4）Cache不命中时主存能和CPU直接通信，同时将数据调入Cache中；而虚拟存储器系统不命中时，只能先由硬盘调入内存中，而不能直接和CPU通信。

第4章 指令系统

**4.1 指令格式**

指令的长度是指一条指令中所包含的二进制代码的位数。指令字长取决于操作码的长度、操作数地址码的长度和操作数地址的个数。指令长度与机器字长没有固定的关系。

**零地址指令**：  
1）不需要操作数的指令：空操作指令、停机指令、关中断指令  
2）零地址的运算类指令仅用在堆栈计算机中  
**一地址指令**：  
1）只有目的操作数的单操作数指令，按A1地址读取操作数，进行OP操作后，结果存回原地址，即OP(A1)->A，如加1，减1，求反、求补等  
2）隐含约定目的地址的双操作数指令，按指令地址A1可读取源操作数，指令可隐含约定另一个操作数由ACC（累加器）提供，运算结果也存放在ACC中，即(ACC)OP(A1)->ACC  
**二地址指令**：给出目的操作数和源操作数的地址，其中目的操作数地址还用于保存本次的运算结果，即(A1)OP(A2)->A1  
**三地址指令**：(A1)OP(A2)->A3  
**四地址指令**：(A1)OP(A2)->A3，A4=下一条将要执行指令的地址

**扩展操作码的指令格式**

**4.2 指令寻址方式**

**指令寻址**

**1）顺序寻址方式**：(PC)+1（1为指令字长）

**2）跳跃寻址方式**：通过转移类指令实现。就是计算下一条指令的地址，然后修改程序计数器（PC）。

**数据寻址**

**1）隐含寻址**：在指令中隐含着操作数的地址，累加器ACC对单地址指令格式来说是隐含地址  
**优点**：有利于缩短指令字长  
**缺点**：需增加存储操作数或隐含地址的硬件

**2）立即（数）寻址**：指出的是操作数本身，又称为立即数  
**优点**：指令在执行阶段不访问主存，指令执行时间最短  
**缺点**：A的位数限制了立即数的范围

**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，即EA=A  
**优点**：简单，指令在执行阶段仅访问一次主存，不需要专门计算操作数的地址  
**缺点**：A的位数决定了该指令操作数的寻址范围，操作数的地址不易修改

**4）间接寻址**：指令的地址字段给出的形式地址不是操作数的真正地址，而是操作数有效地址所在的存储单元的地址，也就是操作数地址的地址，即EA=(A)  
**优点**：可扩大寻址范围，便于编制程序（用间接寻址可以方便地完成子程序返回）  
**缺点**：指令在执行阶段要多次访存（一般问到扩大寻址范围通常都是寄存器间接寻址）

**5）寄存器寻址**：在指令字中直接给出操作数所在的寄存器编号，即EA=Ri，其操作数在由Ri所指的寄存器内  
**优点**：指令在执行阶段不访问主存，只访问寄存器，执行速度快，支持向量/矩阵运算  
**缺点**：寄存器价格昂贵，寄存器个数有限

**6）寄存器间接寻址**：在寄存器Ri中给出的不是一个操作数，而是操作数所在主存单元的地址，即EA=(Ri)  
**优点**：与一般间接寻址相比速度更快  
**缺点**：需要访问主存

**7）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A而形成操作数的有效地址，即EA=(PC)+A  
**优点**：操作数的地址不是固定的，随着PC的变化而变化，并且与指令地址之间总是相差一个固定值，因此便于程序浮动，广泛应用于转移指令

**8）基址寻址**：将CPU中基址寄存器（BR）的内容加上指令格式中的形式地址A，而形成操作数的有效地址，即EA=(BR)+A  
**优点**：可扩大寻址范围，用户不必考虑自己的程序位于主存的哪一空间区域，故有利于多道程序设计，以及用于编制浮动程序

**9）变址寻址**：有效地址EA等于指令字中的形式地址A与变址寄存器IX的内容相加之和，即EA=(IX)+A  
**优点**：可扩大寻址范围，用于处理数组问题

**10）堆栈寻址**：硬堆栈、软堆栈

| **寻址方式** | **有效地址** | **访存次数** |
| --- | --- | --- |
| 隐含寻址 | 程序指定 | 0 |
| 立即寻址 | A即是操作数 | 0 |
| 直接寻址 | EA=A | 1 |
| 一次间接寻址 | EA=(A) | 2 |
| 寄存器寻址 | EA=Ri | 0 |
| 寄存器间接一次寻址 | EA=(Ri) | 1 |
| 相对寻址 | EA=(PC)+A | 1 |
| 基址寻址 | EA=(BR)+A | 1 |
| 变址寻址 | EA=(IX)+A | 1 |

X86汇编指令入门

**4.3 CISC和RISC的基本概念**

| **类别** | **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 | 固定 |
| 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 |

第5章 中央处理器

**5.1 CPU的功能和基本结构**

具体功能：指令控制、操作控制、时间控制、数据加工、中断处理

**运算器**：主要由算术逻辑单元、暂存寄存器、累加寄存器、通用寄存器组、程序状态字寄存器、移位器、计数器等组成  
**控制器**：主要由程序计数器、指令寄存器、指令译码器、存储器地址寄存器、存储器数据寄存器、时序系统和微操作信号发生器等组成

**注意：通用寄存器组、程序状态字寄存器、程序计数器对用户是可见的，存储器地址寄存器、存储器数据寄存器、指令寄存器对用户是透明的，不可见的。**

**5.2 指令执行过程**

CPU从主存中取出并执行一条指令所需的全部时间称为**指令周期**，也就是CPU完成一条指令的时间。一个指令周期通常包含若干机器周期，一个机器周期又包含若干时钟周期。

一个完整的指令周期应包括**取指、间址、执行和中断**四个周期。  
**注意**：中断周期中进栈操作是**将SP减1**，和传统意义上的进栈操作相反，因为计算机的堆栈中都是向低地址增加，所以进栈操作是减1，不是加1.

**取指周期**是根据PC中的内容从主存中取出指令代码并存放在指令寄存器IR中，取指令的同时，PC加1  
**间址周期**是取操作数有效地址  
**执行周期**是根据IR中的指令字的操作码和操作数通过ALU操作产生执行结果  
**中断周期**是处理中断请求，假设程序断点存入堆栈中，并用SP表示栈顶地址，而且进栈操作是先修改栈顶指针，后存入数据。

指令执行方案

**单指令周期**：对所有指令都选用相同的时间来完成，指令之间串行执行，指令周期取决于执行时间最长的指令的执行时间。  
**多指令周期**：指令之间串行执行，不再要求所有指令占用相同的执行时间

**流水线方案**：指令之间并行执行，尽量让多条指令同时执行，但各自处在不同的执行步骤中

**5.3 数据通路的功能和基本结构**

数据在功能部件之间传送的路径称为**数据通路**，其功能是**实现CPU内部的运算器和寄存器以及寄存器之间的数据交换**。

基本结构：  
1）**CPU内部单总线方式**：所有寄存器的输入端和输出端都连接到一条公共的通路上。结构简单，但数据传输中存在较多冲突现象，性能较低

2）**CPU内部三总线方式**：所有寄存器的输入端和输出端都连接到多条公共的通路上。效率CPU内部单总线方式相对有所提高

3）**专用数据通路方式**：根据指令执行过程中的数据和地址的流动方向安排连接线路，避免使用共享的总线。性能较高，但硬件量大

前两者为内部总线，第三个为专用数据总线

#区分内部总线和系统总线（地址总线，数据总线，控制总线）

**5.4 控制器的功能和工作原理**

控制器的结构：

1）运算器部件通过数据总线与内存储器、输入设备和输出设备传送数据  
2）输入设备和输出设备通过接口电路与总线相连接  
3）内存储器、输入设备和输出设备从地址总线接收地址信息，从控制总线得到控制信号，通过数据总线与其他部件传送数据  
4）控制器部件从数据总线接收指令信息，从运算器部件接收指令转移地址，送出指令地址到地址总线，还要向系统中的部件提供它们运行所需要的控制信号

**控制器的主要功能**有：  
1）从主存中取出一条指令，并指出下一条指令在主存中的位置  
2）对指令进行译码或测试，产生相应的操作控制信号，以便启动规定的动作  
3）指挥并控制CPU、主存、输入和输出设备之间的数据流动方向

**1）硬布线控制器由复杂的组合逻辑门电路和一些触发器构成，因此又称为组合逻辑控制器**。  
通常以存取周期作为基准时间，即内存中读取一个指令字的最短时间作为机器周期。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。

**2）微程序控制器采用存储逻辑实现**，也就是把微操作信号代码化，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，**微操作控制信号由微指令产生**。

微程序设计思想是将每一条机器指令编写成一个微程序，每一个微程序包含若干条微指令，每一条微指令对应一个或几个微操作命令。  
微命令是微操作的控制信号，微操作是微命令的执行过程。  
**注意**：在组合逻辑控制器中也存在微命令与微操作这两个概念，它们并非只是微程序控制器的专有概念。  
微指令是若干微命令的集合，存放微指令的控制存储器的单元地址称为微地址。  
微周期通常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。  
主存储器用于存放程序和数据，在CPU外部，用RAM实现；控制存储器CM用于存放微程序，在CPU内部，用ROM实现。  
**注意：若指令系统中具有n种机器指令，则控制存储器中的微程序数至少是n+1个（1为公共的取指微程序）**

**#微命令（控制序列的最小单位），微操作（最小操作），微指令（若干微命令的集合），微周期（读取并执行微指令的时间），微程序（微指令的有序集合）**

**微指令的编码方式又称为微指令的控制方式**，指如何对微指令的控制字段进行编码，以形成控制信号。目标是在保证速度的情况下，尽量缩短微指令字长。  
1）**直接编码（直接控制）方式**  
2）**字段直接编码方式**：将微指令的微命令字段分成若干个小字段，把互斥性微命令组合在同一字段中，把相容性微命令组合在不同的字段中  
3）**字段间接编码方式，又称隐式编码**

**#微命令的编码方式**

**微指令的地址形成方式**：  
1）**直接由微指令的下地址字段指出（又称为断定方式）**：由微指令的下地址字段直接指出后继微指令的地址  
2）**根据机器指令的操作码形成**：当机器指令取至指令寄存器后，微地址的地址由操作码经微地址形成部件形成

**微指令的格式**：  
1）**水平型微指令**：指令字中的一位对应一个控制信号，有输出时为1，否则为0。一条水平型微指令定义并执行几种并行的基本操作。优点是微程序短，执行速度快；缺点是微指令长，编写微程序较麻烦。  
2）**垂直型微指令**：类似机器指令操作码的方式，设置微操作码字段，由微操作码规定微指令的功能。一条垂直型微指令只能定义并执行一种基本操作。优点是微指令短、简单、规整，便于编写微程序；缺点是微程序长，执行速度慢，工作效率低。  
3）**混合型微指令**

| **类别** | **微程序控制器** | **硬布线控制器** |
| --- | --- | --- |
| 工作原理 | 微操作控制信号以微程序的形式存放在控制存储器中，执行指令时读出即可 | 微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生 |
| 执行速度 | 慢 | 快 |
| 规整性 | 较规整 | 烦琐、不规整 |
| 应用场合 | CISC CPU | RISC CPU |
| 易扩充性 | 易扩充修改 | 困难 |

**5.5 指令流水线**

一条指令的执行过程：  
**取指**：根据PC内容访问主存储器，取出一条指令送到IR中  
**分析**：对指令操作码进行译码，按照给定的寻址方式和地址字段中的内容形成操作数的有效地址EA，并从有效地址EA中取出操作数  
**执行**：根据操作码字段，完成指令规定的功能，即把运算结果写到通用寄存器或主存中

1）顺序执行方式：前一条指令执行完后，才启动下一条指令，T=3nt  
2）一次重叠执行方式：第k条指令的执行阶段和第k+1条指令的取指阶段同时进行，T=(1+2n)t  
3）二次重叠执行方式：把取第k+1条指令提前到分析第k条指令的期间完成，而将分析第k+1条指令与执行第k条指令同时进行，T=(2+n)t

流水线的分类

按使用级别分：部件功能级流水线、处理机级流水线、处理机间级流水线  
按完成功能分：单功能流水线、多功能流水线  
按连接方式分：动态流水线、静态流水线  
按有无反馈信号分：线性流水线、非线性流水线

**影响流水线的因素**  
1）**结构相关（资源冲突）：由于多条指令在同一时刻争用同一资源而形成的冲突**  
解决办法：  
a. 前一指令访存时，使后一条相关指令（以及其后续指令）暂停一个时钟周期  
b. 单独设置数据存储器和指令存储器，使两项操作各自在不同的存储器中进行，这属于资源重复配置

2）**数据相关（数据冲突，数据冒险）：必须等前一条指令执行完才能执行后一条指令的情况**  
a. 把遇到数据相关的指令及其后续指令都暂停一至几个时钟周期，可分为硬件阻塞（stall）和软件插入“NOP”指令两种方法  
b. 设置相关专用通路，即不等前一条指令把计算结果返回寄存器组，下一条指令也不再读寄存器组，而是直接把前一条指令的ALU的计算结果作为自己的输入数据开始计算过程，使本来需要暂停的操作可以继续执行，称为数据旁路技术  
c. 通过编译器对数据相关的指令编译优化的方法，调整指令顺序来解决数据相关

3）**控制相关（控制冲突，控制冒险）：遇到转移指令和其他改变PC值得指令而造成断流**  
a. 对转移指令进行分支预测，分为简单（静态）预测和动态预测，静态预测总是预测条件不满足，即继续执行后续指令；动态预测根据程序执行的历史情况，进行动态预测调整。  
b. 预取转移成功和不成功两个控制流方向上的目标指令  
c. 加快和提前形成条件码  
d. 提高转移方向的猜准率

#两条连续的指令读取相同的寄存器时，就会产生RAR（读后读）相关，这种相关不会影响流水线；  
当某条指令要读取上一条指令所写入的寄存器时，就会产生RAW（写后读）相关，这种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。  
当某条指令的上条指令要读/写该指令的输出寄存器时，就会产生WAR（读后写）和WAW（写后写）相关。在非按序流动的流水线中，既可能发生RAW，也可能发生WAR和WAW相关。  
对流水线影响最严重的指令相关是数据相关。

**流水线的性能指标**  
1）**流水线的吞吐率**：，n是任务数，Tk是处理完成n个任务所用的时间；一条k段线性流水线能够在k+n-1个时钟周期内完成n个任务，此时，t为时钟周期；当n趋于无穷时，得到最大吞吐率  
2）**流水线的加速比**：完成同样一批任务，不使用流水线所用的时间与使用流水线所用的时间之比。，当n趋于无穷时，得到最大加速比  
3）**流水线的效率**：=n个任务占用k时空区有效面积/n个任务所用的时间与k个流水段所围成的时空区总面积=，当n趋于无穷时，最高效率为

超标量流水线技术：每个时钟周期内可并发多条独立指令，即以并行操作方式将两条或多条指令编译并执行，为此需配置多个功能部件  
超流水线技术：在一个时钟周期内再分段，在一个时钟周期内一个功能部件使用多次  
超长指令字：将多条能并行操作的指令组合成一条具有多个操作码字段的超长指令字，为此需要采用多个处理部件

第6章 总线

**6.1 总线概述**

**总线**是一组能为多个部件**分时共享**的公共信息传送线路。**分时和共享**是总线的两个特点。  
分时是指同一时刻只允许由一个部件向总线发送信息；共享是指总线上可以挂接多个部件，各个部件之间互相交换的信息都可以通过这组线路分时共享。在某一时刻只允许由一个部件向总线发送信息，但多个部件可以同时从总线上接收相同的信息。

主设备：是指获得总线控制权的设备  
从设备：是指被主设备访问的设备，只能响应从主设备发来的各种总线命令

总线特性：机械特性、电气特性、功能特性、时间特性

**猝发传输方式**：在一个总线周期内传输存储地址连续的多个数据字的总线传输方式叫猝发传送

**总线的分类**：  
1）**片内总线**：CPU芯片内部寄存器与寄存器之间、寄存器与ALU之间的公共连接线  
2）**系统总线**：计算机系统内各功能部件（CPU、主存、I/O接口）之间相互连接的总线  
a. **数据总线**：用来传输各功能部件之间的数据信息，双向传输，其位数与机器字长、存储字长有关  
b. **地址总线**：用来指出数据总线上的源数据或目的数据所在的主存单元或I/O接口的地址，单向传输，其位数与主存地址空间有关  
c. **控制总线**：传输控制信息，包括CPU送出的控制命令和主存（或外设）返回CPU的反馈信号  
3）**通信总线**：用于计算机系统之间或计算机系统与其他系统（如远程通信设备、测试设备）之间信息传送的总线，也称为**外部总线**

系统总线的结构：  
1）单总线结构：CPU、主存、I/O设备（通过I/O接口）都挂在一组总线上  
2）双总线结构：主存总线、I/O总线  
3）三总线结构：主存总线、I/O总线、DMA总线

**总线的性能指标**：  
1）传输周期，简称总线周期  
2）时钟周期  
3）工作频率：总线周期的倒数，总线工作频率=时钟频率/N（N表示N个时钟周期）  
4）时钟频率  
5）**总线宽度**，又称总线位宽，是总线上同时能够传输的数据位数，通常是指数据总线的根数  
6）**总线带宽**，总线的数据传输率，即单位时间内总线上可传输数据的位数，总线带宽=总线工作频率 \* (总线宽度/8)

总线带宽=总线宽度 \* 总线频率

**6.2 总线仲裁**

为解决多个主设备同时竞争总线控制权的问题，应当采用总线仲裁部件，以某种方式选择一个主设备优先获得总线控制权。

**集中仲裁方式**：将所有的总线请求集中起来，利用一个特定的裁决算法进行裁决  
1）**链式查询方式**：离总线控制器越近的部件，其优先级越高。优点：链式查询方式优先级固定，结构简单，扩充容易；缺点：对故障敏感，优先级不能改变  
2）**计数器定时查询方式**：优点：设备使用总线的优先级相等，优先次序可以改变，对故障不敏感；缺点：增加了控制线数，相对复杂  
3）**独立请求方式**：优点：响应速度快，对优先次序的控制相当灵活；缺点：控制线数量多，逻辑更复杂

|  | **链式查询** | **计数器定时查询** | **独立请求** |
| --- | --- | --- | --- |
| 控制线数 | 3 |  | 2n+1 |
| 优点 | 优先级固定；结构简单；扩充容易 | 优先级灵活 | 响应速度快；优先级灵活 |
| 缺点 | 对电路故障敏感；优先级不灵活 | 控制线多；控制复杂 | 控制线多；控制复杂 |

**分布仲裁方式**：不需要中央仲裁器，每个潜在的主模块都有自己的仲裁号和仲裁器。如果仲裁总线上的号优先级高，则它的总线请求不予响应，并撤销它的仲裁号，最后获胜者的仲裁号保留在仲裁总线上。

**6.3 总线操作和定时**

一个总线周期可分为4个阶段：  
1）**申请分配阶段**：由需要使用总线的主模块（或主设备）提出申请，经总线仲裁机构决定将下一传输周期的总线使用权授予某一申请者。  
2）**寻址阶段**：主模块通过总线发出本次要访问的从模块（或从设备）的地址及有关命令，启动参与本次传输的从模块。  
3）**传输阶段**：主模块和从模块进行数据交换，可单向或双向进行数据传送  
4）**结束阶段**：主模块的有关信息均从系统总线上撤除，让出总线使用权

**同步定时方式**：系统采用一个**统一的时钟信号**来协调发送和接收双方的传递定时关系。适用于总线长度较短及总线所接部件的存取时间比较接近的系统。  
优点：传送速度快，具有较高的传输速率；总线控制逻辑简单  
缺点：主从设备属于强制性同步；不能及时进行数据通信的有效性检验，可靠性较差

**异步定时方式**：没有统一的时钟，也没有固定的时间间隔，**完全依靠传送双方相互制约的“握手”信号**来实现定时控制。  
优点：总线周期长度可变，能保证两个工作速度相差很大的部件或设备之间可靠地进行信息交换，自动适应时间的配合  
缺点：比同步控制方式稍复杂一些，速度比同步定时方式慢  
1）**不互锁方式**：主设备自动撤销“请求”信号，从设备自动撤销“回答”信号，双方不存在互锁关系  
2）**半互锁方式**：主设备发出“请求”信号后，必须待接到从设备的“回答”信号后，才撤销“请求”信号，有互锁关系；而从设备在接到“请求”信号后，发出“回答”信号，但不必等待获知主设备的“请求”信号已经撤销，而是隔一段时间后自动撤销“回答”信号，不存在互锁关系。  
3）**全互锁关系**：主设备发出“请求”信号后，必须待从设备“回答”后，才撤销“请求”信号；从设备发出“回答”信号，必须待获知主设备“请求”信号已撤销后，再撤销其“回答”信号。双方存在互锁关系。

**6.4 总线标准**

**ISA总线**：最早出现的微型计算机的**系统总线**标准，应用在IBM的AT机上  
**EISA总线**：为配合32位CPU而设计的总线扩展标准，EISA对ISA完全兼容  
**VESA总线**：32位标准的计算机**局部总线**，是针对多媒体PC要求**高速传送活动图像的大量数据**应运而生的  
**PCI总线**：高性能的32位或64位总线，专为**高度集成的外围部件、扩充插板和处理器/存储器系统而设计的互联机制**。目前常用的PCI适配器有**显卡、声卡、网卡**等。PCI总线**支持即插即用**，与处理器时钟频率无关，属于**局部总线**，可以通过**桥连接**实现多层PCI总线  
**PCI-Express（PCI-E）是最新的总线和接口标准，将全面取代现行的PCI和AGP，最终实现总线标准的统一**  
**AGP**：是一种**视频接口标准**，专用于**连接主存和图形存储器**，属于**局部总线**。AGP技术为**传输视频和三维图形数据**提供了切实可行的解决方案。  
**RS-232C总线**是由美国电子工业协会（EIA）推荐的一种**串行通信总线标准**，是应用于**串行二进制交换**的数据终端设备（DTE）和数据通信设备（DCE）之间的标准接口  
**USB总线**：连接外部设备的I/O总线标准，属于**设备总线**，具有**即插即用、热插拔**等优点，有很强的连接能力  
**PCMCIA**：广泛**应用于笔记本电脑**中的一种接口标准，是一个小型的**用于扩展功能的插槽，可即插即用**  
**IDE总线**：ATA，是一种IDE接口磁盘驱动器接口类型，**硬盘和光驱通过IDE接口与主板连接**  
**SCSI**：用于**计算机和智能设备之间（硬盘、软驱、光驱、打印机等）系统级接口**的独立处理器标准  
**SATA**：基于行业标准的**串行硬件驱动器接口**，**硬盘接口规范**

第7章 输入/输出系统

**7.1 I/O系统基本概念**

I/O控制方式主要有一下4种：  
1）**程序查询方式**：由CPU通过程序不断查询I/O设备是否已做好准备，从而控制I/O设备与主机交换信息  
2）**程序中断方式**：只在I/O设备准备就绪并向CPU发出中断请求时才予以响应  
3）**DMA方式**：主存和I/O设备之间有一条直接数据通路，当主存和I/O设备交换信息时，无需调用中断服务程序  
4）**通道方式**：在系统中设有通道控制部件，每个通道都挂接若干外设，主机在执行I/O命令时，只需启动有关通道，通道将执行通道程序，从而完成I/O操作。

**7.2 外部设备**

显示存储器VRAM：  
VRAM容量 = 分辨率 \* 灰度级位数  
VRAM带宽 = 分辨率 \* 灰度级位数 \* 帧频

一块磁盘含有若干个记录面，每个记录面划分为若干条磁道，而每条磁道又划分为若干个扇区，扇区（也称为块）是磁盘读写的最小单位，也就是说磁盘按块读取。  
**磁头数**：即记录面数，磁头用于读取/写入盘片上记录面的信息，一个记录面对应一个磁头  
**柱面数**：表示硬盘每一面盘片上有多少条磁道  
**扇区数**：表示每一条磁道上有多少个扇区

**平均存取时间由寻道时间（磁头移动到目的磁道）、旋转延迟时间（磁头定位到所在扇区）和传输时间（传输数据所花费的时间）三部分构成**

硬盘属于机械式部件，其读写操作是串行的，不可能在同一时刻既读又写，也不可能在同一时刻读两组数据或写两组数据。

RAID0没有容错能力，RAID1-RAID5的几种方案中，无论何时有磁盘损坏，都可以随时拔出受损的磁盘再插入好的磁盘，而数据不会损坏。  
RAID0：无冗余和无校验的磁盘阵列  
RAID1：镜像磁盘阵列  
RAID2：采用纠错的海明码的磁盘阵列  
RAID3：位交叉奇偶校验的磁盘阵列  
RAID4：块交叉奇偶校验的磁盘阵列  
RAID5：无独立校验的奇偶校验磁盘阵列  
RAID通过同时使用多个磁盘，提高了传输率；通过在多个磁盘上并行存取来大幅提高存储系统的数据吞吐量；通过镜像功能，可以提高安全可靠性；通过数据校验，可以提供容错能力。

**7.3 I/O接口**

**I/O接口的功能**：  
1）实现主机和外设的通信联络控制  
2）进行地址译码和设备选择  
3）实现数据缓冲  
4）信号格式的转换  
5）传送控制命令和状态信息

I/O接口的基本结构

**内部接口**：内部接口与系统总线相连，实质上是与内存、CPU相连，数据的传输方式只能是**并行传输**  
外部接口：外部接口通过接口电缆与外设相连，外部接口的数据传输可能是串行方式，因此I/O接口需具有串/并转换功能。  
**注意**：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才可以组成接口。

I/O接口的类型

**I/O端口**是指接口电路中可以被CPU直接访问的寄存器，主要有数据端口、状态端口和控制端口，若干个端口加上相应的控制逻辑电路组成接口。**CPU能对数据端口执行读写操作，对状态端口只能执行读操作，对控制端口只能执行写操作。**

**I/O端口的编址方式**：  
1）**统一编址**，又称存储器映射方式，是指把I/O端口当做存储器的单元进行地址分配，这种方式CPU不需要设置专门的I/O指令，用**统一的访存指令**就可以访问I/O端口  
2）**独立编址**，又称I/O映射方式，是指I/O端口地址与存储器地址无关，独立编址CPU**需要设置专门的输入/输出指令访问端口**

**7.4 I/O方式**

**程序查询方式**：CPU一旦启动I/O，必须停止现行程序的运行，并在现行程序中插入一段程序，有“踏步”现象，**CPU与I/O串行工作**  
1）CPU执行初始化程序，并预置传送参数  
2）向I/O接口发出命令字，启动I/O设备  
3）从外设接口读取其状态信息  
4）CPU不断查询I/O设备状态，直到外设准备就绪  
5）传送一次数据  
6）修改地址和计数器参数  
7）判断传送是否结束，若没结束转第3步，直到计数器为0

**程序中断方式：CPU与I/O并行工作**  
**程序中断**是指在计算机执行现行程序的过程中，出现某些急需处理的异常情况或特殊请求，CPU暂时中止现行程序，而转去对这些异常情况或特殊请求进行处理，在处理完毕后CPU又自动返回到现行程序的断点处，继续执行原程序。

**程序中断方式工作流程**：  
1）中断请求：中断源向CPU发出中断信号  
**外中断**：来自处理器和内存以外的部件引起的中断  
**内中断**：在处理器和内存内部产生的中断  
**硬件中断属于外中断；软件中断属于内中断。  
非屏蔽中断是一种硬件中断，属于外中断；可屏蔽中断也是一种硬件中断，属于外中断。**

2）中断判优：判断多个中断源的优先级，可由硬件实现或软件实现  
硬件中断属于最高级，其次是软件中断，非屏蔽中断优于可屏蔽中断，DMA请求优于I/O设备传送的中断请求，高速设备优于低速设备，输入设备优于输出设备，实时设备优于普通设备。

3）CPU响应中断的条件  
有中断源提出中断请求  
CPU允许中断及开中断  
一条指令执行完毕且没有更紧迫的任务

4）中断隐指令：完成关中断、保存断点、引出中断服务程序，由硬件直接实现，不允许、也不可能为用户使用的特殊指令。

5）中断向量：查询中断服务程序的入口地址  
**注意：中断向量是中断服务程序的入口地址，中断向量地址地址是指中断服务程序的入口地址的地址。**

6）中断处理：执行中断服务程序最后恢复现场，中断返回  
中断服务程序的最后一条指令通常是一条**中断返回指令**，使其返回到原程序的断点处，以便继续执行原程序。

**多重中断**：当CPU处理中断时，又有更高优先级的中断请求  
条件：  
1）在中断服务程序中提前设置开中断指令  
2）优先级高的中断源有权中断优先级低的中断源

**DMA方式**：完全由**硬件**进行成组信息传送的控制方式，主存和DMA接口之间有一条直接数据通路，**I/O与主机并行工作，程序和传送并行工作**

**组成**：  
主存地址计数器：存放要交换数据的主存地址  
传送长度计数器：用来记录传送数据的长度  
数据缓冲寄存器：用于暂存每次传送的数据  
DMA请求触发器：I/O设备准备好数据后使DMA请求触发器置位  
控制/状态逻辑：由控制和时序电路及状态标志组成  
中断机构：数据块传送完毕后触发中断机构，提出中断请求

**传送方式**：  
**停止CPU访存**：当需要传送数据时，停止CPU对主存的访问  
**交替访存**：将CPU周期分为DMA访存和CPU访存两个部分  
**周期挪用（周期窃取）**：I/O设备需要访存时，挪用一个或几个存取周期

**传送过程**：  
**预处理**：完成寄存器置初始值之类的准备工作，I/O设备向DMA控制器发送DMA请求，再由DMA控制器向CPU发送总线请求  
**数据传送**：占用总线传输数据，数据传送**完全由DMA（硬件）控制**  
**后处理**：CPU执行中断服务程序做结束DMA处理

**DMA方式和中断方式的区别**：  
1）中断方式是程序的切换，需要保护和恢复现场；而DMA方式除了预处理和后处理，其他时候不占用CPU的任何资源  
2）对中断请求的响应只能发生在每条指令执行完毕时（即指令的执行周期之后）；而对DMA请求的响应可以发生在每个机器周期结束时（在取指、间址、执行周期之后均可），只要CPU不占用总线就可以被响应  
3）中断传送过程需要CPU的干预；而DMA传送过程不需要CPU的干预，故数据传输速率非常高，适合于高速外设的成组数据传送  
4）DMA请求的优先级高于中断请求  
5）中断方式具有对异常事件的处理能力，而DMA方式仅限于传送数据块的I/O操作  
6）从数据传送来看，中断方式靠程序传送，DMA方式靠硬件传送