计算机组成原理总复习

- 期末考试试卷题型分布
  - 1. 单项选择题（本大题共16小题，每小题2分，共32分）
  - 2. 名词解释（本大题共4小题，每小题5分，共20分）
  - 3. 计算题（本大题共2小题，每小题6分，共12分）
  - 4. 问答题（本大题共3小题，每小题12分，共36分）
- 注意事项
  - 认真答案，仔细检查，防止粗心大意造成丢分
  - 单项选择题需要将所选答案填写在相应的表格中
  - 闭卷考试120分钟，要求开考60分钟后才允许交卷

第一章 概论

- 计算机系统的层次结构
  - 按照计算机语言从低级到高级的次序可分五个层次级别：
  - 第1级：微程序机器级，由机器硬件直接执行微指令
  - 第2级：传统机器级，由微程序解释机器指令系统
  - 第3级：操作系统级，由操作系统程序实现，统一管理和调度计算机系统中的软硬件资源，支撑其他系统软件和应用软件。
  - 第4级：汇编语言机器级，由汇编程序支持和执行
  - 第5级：高级语言机器级，由高级语言编译程序支持和执行。
  - 计算机系统各个层次之间关系紧密，上层是下层功能的扩展，下层是上层的基础。

第一章 概论

- 冯·诺依曼计算机的特点是：
  - 计算机由运算器、存储器、控制器和输入设备、输出设备等五大部件组成。
  - 指令和数据以同等的地位存放于存储器内，并可以按地址寻访。
  - 指令和数据均可以用二进制代码表示。
  - 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所在存储器中的位置。
  - 指令在存储器内按顺序存放。通常，指令是顺序执行的，在特定情况下，可根据运算结果或根据设定的条件改变执行顺序。
  - 计算机以运算器为中心，输入输出设备与存储器的数据传送通过运算器。

第一章 概论

- 控制器：
  - 对当前指令进行译码分析其所需要完成的操作，产生并发送各部件所需要的控制信号，从而使整个计算机自动、协调地工作
- 运算器：
  - 用来完成算术和逻辑运算，并将运算的中间结果暂存在运算器内
- 存储器：
  - 存放指令和数据
- 输入设备：
  - 将外界信息转换为计算机能识别的二进制代码
- 输出设备：
  - 将计算机处理结果转换成人们或其他设备所能接收的形式

第一章 概论

- 主存：又称内存，用于存放计算机当前正在执行的数据和程序，可以被CPU直接存取
- CPU：中央处理器，是计算机硬件核心部件，由运算器和控制器构成
- 主机：CPU与主存合起来称为主机
- 外设：输入设备、输出设备的统称
- 计算机硬件技术指标
  - 机器字长：指CPU一次能处理数据的二进制位数，通常与CPU的寄存器位数有关
  - 指令字长：机器指令中含二进制代码的总位数
  - 存储字长：存储单元中二进制代码的个数
  - MIPS：每秒百万条指令数
  - MFLOPS：每秒百万条浮点运算指令数

第二章 计算机发展及应用

- Moore定律
  - Intel公司的缔造者Gordon Moore提出
  - 微芯片上集成的晶体管数目每三年翻两番
- 世界上第一台电子计算机 ENIAC(1946)
- 计算机发展的五个阶段
  - 电子管
  - 晶体管
  - 中小规模集成电路
  - 大规模集成电路
  - 超大规模集成电路

第三章 系统总线

- 系统总线：计算机系统各大部件如CPU、主存、I/O接口之间的信息传输线。按传输信息的不同，分为数据总线，地址总线，控制总线：
  - 数据总线：用来传输各功能部件之间的数据信息，是双向传输总线，其位数与机器字长，存储字长有关。一般为8位，16位或32位
  - 地址总线：用来指出数据总线上的源数据或目的数据在存储单元的地址，是单向传输的，其位数与存储单元的个数有关（几次幂的关系）
  - 控制总线：用来发出各种控制信号，对任一控制线而言，其传输都是单向的。与机器字长，存储字长，存储单元无关系

第三章 系统总线

- 总线仲裁：即总线判优，主要解决在多个主设备申请占用总线时，由总线控制器仲裁出优先级别最高的设备，允许其占用总线。
- 总线带宽：是指总线在单位时间内可以传输的二进制数据的总位数，即总线最大数据传输率，通常等于总线宽度与总线工作频率的乘积。
- 总线宽度：数据总线的根数
- 总线时钟频率：总线工作的时钟频率，即单位时间内发出的脉冲数

第三章 系统总线

- 集中式总线仲裁方式：
  - 链式查询方式：设备优先级次序固定，控制连线简单，易于扩充，对电路故障最敏感；
  - 计数器定时查询方式：设备优先级设置比较灵活，对电路故障不敏感，但增加了主控线数，控制过程比较复杂；
  - 独立请求方式：设备优先次序控制灵活，判优速度最快，但控制连线数量多，控制过程最复杂，硬件成本较高

第三章 系统总线

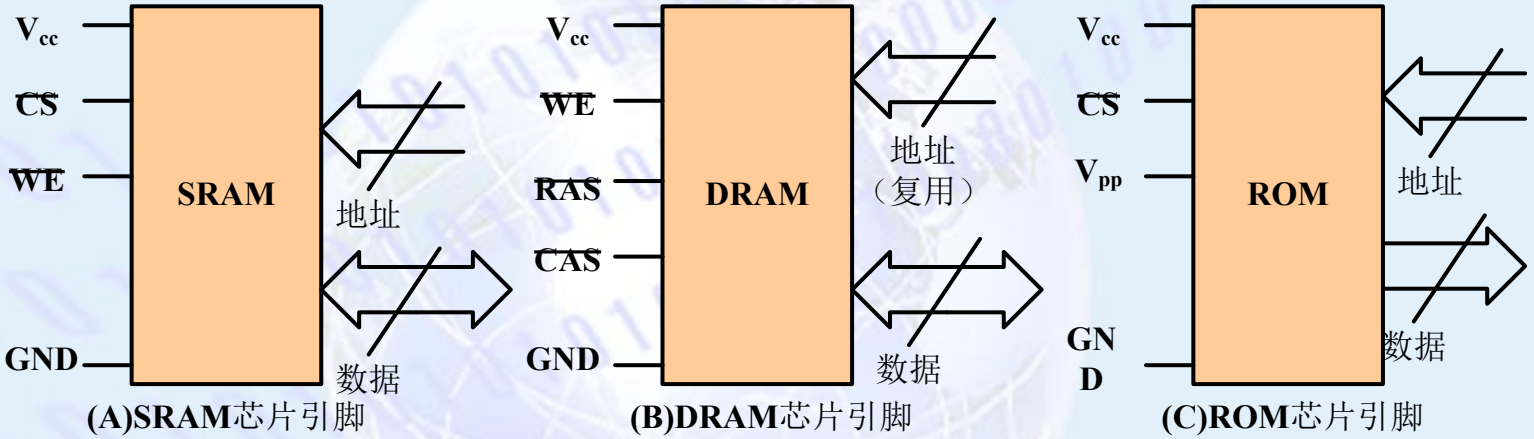
- 流行的总线标准：
  - ISA总线：工业标准体系结构总线
  - EISA总线：扩展工业标准体系结构总线
  - VESA总线：视频电子标准协会总线
  - PCI总线：外围设备互连总线
  - AGP总线：加速图形端口总线
  - RS232总线：串行通信总线
  - USB总线：通用串行总线

第四章 存储器

- **RAM：随机访问存储器**
  - 在程序执行过程中既可读出也可写入，而且存取时间与存储单元所在位置无关，但是保存的信息在掉电后会丢失。
- **ROM：只读存储器**
  - 只能对其存储的内容读出，而不能对其写入的只读存储器，信息在掉电后不会丢失。
- **SRAM：静态随机访问存储器**
  - 利用双稳态触发器来保存信息，只要不断电，信息不会丢失，存取速度快，集成度低，容量小，价格高，常用作Cache。
- **DRAM：动态随机访问存储器**
  - 利用电容存储电荷原理来保存信息，使用时需要不断给电容充电才能使信息保持，存取速度慢，集成度高，容量大，价格低，常用作内存条。

第四章 存储器

- 存储芯片的引脚封装
- 注意DRAM芯片采用行列地址分时复用，地址线引脚只引出了一半，没有片选信号。



第四章 存储器

- Cache：是为了解决CPU和主存的速度匹配，提高访存速度的一种存储器，它设在主存与CPU间，起缓冲作用。一般由SRAM构成。
- DRAM刷新：DRAM是靠电容存储电荷原理存储信息，电容上电荷要放电，造成信息丢失。为了维持所存信息，在一定时间(2ms)内，需要将所存信息读出再重新写入(恢复)，这一过程称作刷新，刷新是一行一行进行的，由CPU自动完成
- DRAM刷新方法
  - 集中刷新：在最大刷新间隔时间内，集中安排一段时间进行刷新
  - 分散刷新：在每个读/写周期之后插入一个刷新周期，无CPU访存死时间
  - 异步刷新：是集中式和分散式的折衷，在2ms内分散地把各行刷新一遍

第四章 存储器

- 存储器带宽：在单位时间中存储器传输数据的速率。
- 存储容量：一个存储器中可以容纳的存储单元总数。
- 存取时间：又称存储器访问时间，是指从启动一次存储器操作到完成该操作所经历的时间。
- 存取周期：是存储器指连续启动两次独立的存储器操作（如连续两次读操作）所需的最小间隔时间，通常存取周期大于存取时间。
- 存取周期和存取时间的主要区别是：
  - 存取时间仅为完成一次操作的时间，而存取周期不仅包含操作时间，还包含操作后线路的恢复时间。即：存取周期=存取时间+恢复时间

第四章 存储器

- 程序局部性原理：是指程序在执行时呈现出局部性规律，即在一段时间内，整个程序的执行仅限于程序中的某一部分。相应地，执行所访问的存储空间也局限于某个内存区域。
- 存储器层次结构体现在Cache—主存和主存—辅存这两个存储层次上。
- Cache—主存层次主要解决CPU和主存速度不匹配问题，对CPU访存起加速作用，即从整体运行的效果分析，CPU访存速度加快，接近于Cache的速度，而寻址空间和位价却接近于主存。
- 主存—辅存层次主要解决存储系统的容量问题，对主存来说起扩容作用，即从程序员的角度看，其所使用的存储器的容量和位价接近于辅存，而速度接近于主存。
- 综合上述两个存储层次的作用，从整个存储系统来看，就达到了速度快、容量大、位价低的优化效果。

第四章 存储器

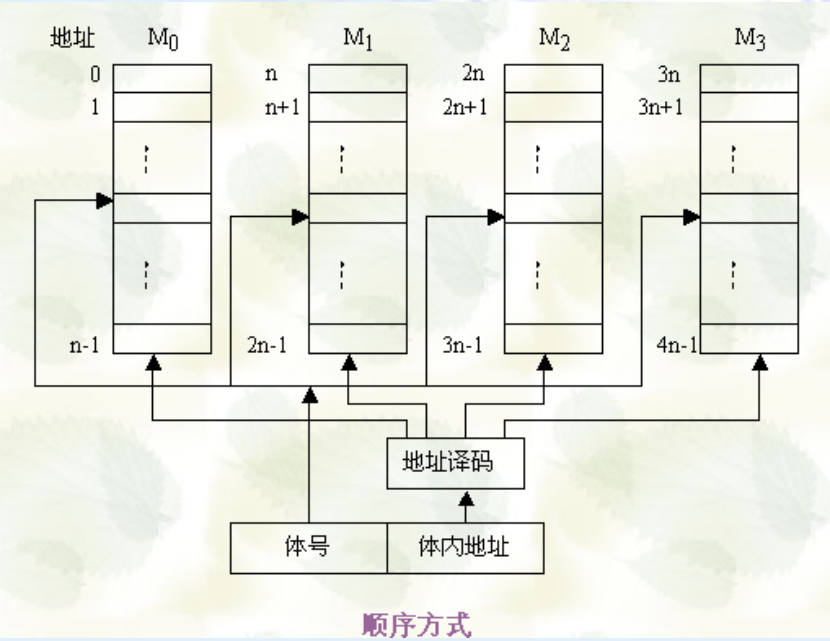
- 计算机对存储器层次结构的管理：
- Cache—主存层次的信息调度功能全部由硬件自动完成。
- 主存—辅存层次的调度目前广泛采用虚拟存储技术实现，即将主存与辅存的一部份通过软硬结合的技术组成虚拟存储器，程序员可以使用这个比主存的实际空间（物理地址空间）大得多的虚拟地址空间（逻辑地址空间）编程，当程序运行时，软、硬件自动配合完成虚拟地址空间与主存实际物理空间的转换。
- 因此，这两个层次上的调度或转换操作对于程序员来说都是透明的。

第四章 存储器

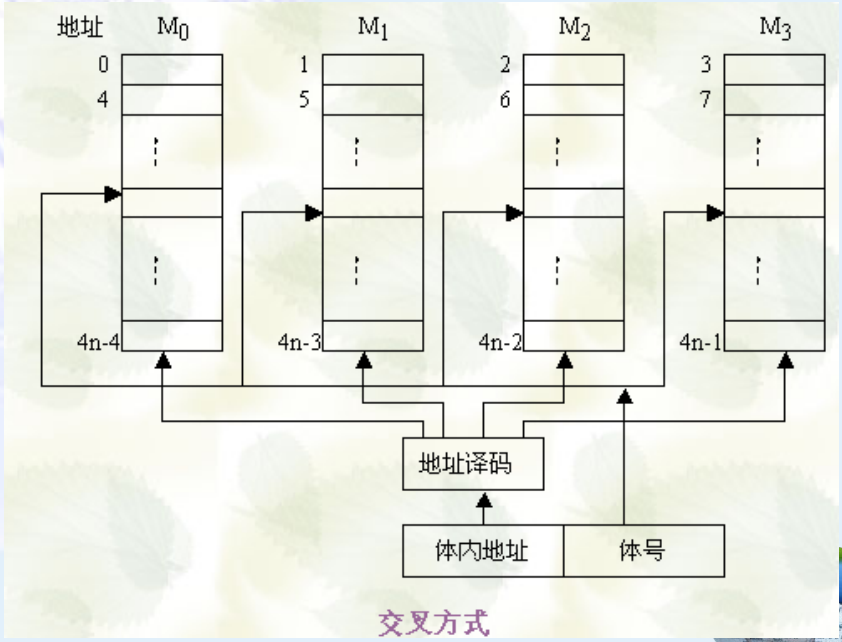
- 存储器容量扩展的三种方法：位扩展、字扩展、字位扩展
- 主存储器与CPU的连接：地址线、数据线、控制线
  - 1 根据CPU芯片提供的地址线数目，确定CPU访存的地址范围，并写出相应的二进制地址码；
  - 2 根据地址范围，确定各种类型存储器芯片的数目和扩展方法；
  - 3 分配CPU地址线。CPU地址线的低位（数量＝存储芯片的地址线数量）直接连接存储芯片的地址线；CPU高位地址线皆参与形成存储芯片的片选信号；
  - 4 连接数据线、R/W#等其他信号线，MREQ#信号一般可用作地址译码器的使能信号。

第四章 存储器

- 存储器字扩展一般采用高位交叉编址，顺序存储，其优点是一个存储体内的地址是连续的，有利于存储器的扩充。
- 存储器字扩展还可以采用低位交叉编址，交叉存储，其优点是可以使连续地址的字分布于不同的模块中，从而可对这些字并行访问，提高访存速度。



顺序方式



交叉方式

第四章 存储器

- 选用4K×4位的SRAM芯片构成32K×16位的主存储器，按字节编址，试问：
  - 1）CPU的数据寄存器最少应有多少位？
  - 2）CPU的地址寄存器最少应有多少位？
  - 3）主存储器共需要多少片SRAM芯片？
  - 4）片选逻辑需要多少位地址？
- 1）因为主存储器需要数据线16位，所以CPU的数据寄存器最少应有16位
- 2）因为主存储器的容量为32K×16=64KB，按字节编址，需要地址线16位，所以，CPU的地址寄存器最少应有16位
- 3）主存储器共需要（32K×16位）÷（4K×4位）= 8×4 = 32片SRAM芯片
- 4）主存储器在字方向扩展了32K ÷ 4K = 8倍，因而片选逻辑需要3位地址

第四章 存储器

- Cache和主存地址映射方式：
  - 直接映射：Cache的一种地址映象方式，一个主存块只能映象到Cache中的唯一的一个指定块
  - 全相联映射：Cache的一种地址映象方式，每个主存块都可映象到任何Cache块
  - 组相联映射：Cache的一种地址映象方式，将存储空间分成若干组，各组之间是直接映象，而组内各块之间则是全相联映象
- 写回法：是在CPU执行写操作时，信息只写入Cache，仅当需要被替换时，才将以被写入过的Cache块先送回主存，然后再调入新块
- 写直达：利用Cache—主存存储层次在处理机和主存之间的直接通路，每当处理机写入Cache的同时，也通过此通路直接写入主存

第五章 输入输出系统

- I/O接口的功能
  - 选址功能,实现设备的选择
  - 实现数据缓冲达到速度匹配
  - 实现数据串并格式的转换
  - 实现电平信号的转换
  - 执行CPU的控制命令
  - 返回外设的状态
  - 中断管理功能

第五章 输入输出系统

- I/O设备与主机信息传送的控制方式
  - 程序查询方式：CPU启动I/O后，时刻查询I/O是否准备好，若设备准备就绪，CPU便转入处理I/O与主机间传送信息的程序；若设备没有准备就绪，则CPU反复查询，踏步等待，直到I/O准备就绪为止。因此主机与I/O是串行工作，CPU效率很低。
  - 程序中断方式：CPU启动I/O后，不必时刻查询I/O是否准备好，而是继续执行程序，当I/O准备就绪时，向CPU发出中断信号，CPU在适当时候响应I/O的中断请求，暂停现行程序为I/O服务。因此主机与I/O并行工作，消除了踏步现象，提高了CPU效率
  - DMA方式：直接内存访问，一种完全由硬件执行I/O交换的工作方式，DMA控制器从CPU完全接管对总线的控制，数据交换不用经过CPU，而直接在内存和I/O设备之间进行。

第五章 输入输出系统

- DMA和主存交换数据的方式：
  - 停止CPU访问主存。这种方法DMA在传送一批数据时，独占主存，CPU放弃了地址线、数据线和有关控制线的使用权。在一批数据传送完毕后，DMA接口才把总线的控制权交回给CPU。显然，这种方法在DMA传送过程中，CPU基本处于不工作状态或保持原状态。
  - 周期挪用。这种方法CPU按程序的要求访问主存，一旦I/O设备有DMA请求，则由I/O设备挪用一个存储周期。此时CPU可完成自身的操作，但要停止访存。显然这种方法既实现了I/O传送，又较好地发挥了主存和CPU的效率，是一种广泛采用的方法。
  - DMA与CPU交替访存。这种方法适合于CPU的工作周期比主存的存取周期长的情况。如CPU的工作周期大于主存周期的两倍，则每个CPU周期的上半周期专供DMA接口访存，下半周期专供CPU访存。这种交替访问方式可使DMA传送和CPU工作效率最高，但相应的硬件逻辑更复杂。

第五章 输入输出系统

- DMA传送过程包括预处理、数据传送和后处理三个阶段。
- DMA接口的基本组成
  - 内存地址计数器---存放访问内存的地址
  - 字计数器---记录传送数据块的长度
  - 数据缓冲寄存器---暂存传送的数据
  - DMA请求触发器---保存外设发来的数据就绪信号（DMA请求）
  - 控制/状态逻辑---DMA接口的核心部分
  - 中断机构---向CPU发中断请求，请求进行后处理（结束处理）

第五章 输入输出系统

- 程序查询方式和程序中断方式的区别
  - 程序查询方式和程序中断方式都通过“程序”传送数据
  - 程序查询方式通过“程序”传送数据时，程序对I/O的控制包括了I/O准备和I/O传送两段时间。由于I/O的工作速度比CPU低得多，因此程序中要反复询问I/O的状态，造成“踏步等待”，严重浪费了CPU的工作时间。
  - 而程序中断方式虽然也是通过“程序”传送数据，但程序仅对I/O传送阶段进行控制，I/O准备阶段不需要CPU查询。故CPU此时照样可以运行现行程序，与I/O并行工作，大大提高了CPU工作效率

第五章 输入输出系统

- 程序中断方式与DMA方式的比较
  - 从数据传送看，程序中断方式靠程序传送，DMA方式靠硬件传送
  - 从CPU响应时间看，程序中断方式在一条指令执行结束时响应，而DMA方式在存取周期结束时响应，即CPU将总线控制权让给DMA传送
  - 程序中断方式有处理异常事件的能力，DMA方式没有这种能力
  - 程序中断方式需要中断现行程序，故需保护现场，DMA方式不必中断现行程序，无需保护现场
  - DMA的优先级比程序中断高

第五章 输入输出系统

- DMA方式中的中断请求和程序中断方式中的中断请求的区别
  - DMA方式中的中断请求不是为了传送信息（信息是通过主存和I/O之间的直接数据通道传送的），只是为了报告CPU一组数据传送结束，有待CPU做一些其他处理工作，如测试传送过程中是否出错，决定是否继续使用DMA方式传送等。
  - 而程序中断方式的中断请求是为了传送数据，I/O设备和主机交换信息完全靠CPU响应中断后，转至中断服务程序完成的。

第五章 输入输出系统

- 比较程序查询、程序中断和DMA三种方式的综合性能。
- ① 程序查询、程序中断方式的数据传送主要依赖软件，DMA主要依赖硬件。
- ② 程序查询、程序中断传送数据的基本单位为字或字节，DMA为数据块。
- ③ 程序查询方式传送时，CPU与I/O设备串行工作；程序中断方式时，CPU与I/O设备并行工作，现行程序与I/O传送串行进行；DMA方式时，CPU与I/O设备并行工作，现行程序与I/O传送并行进行。
- ④ 程序查询方式时，CPU主动查询I/O设备状态；程序中断及DMA方式时，CPU被动接受I/O中断请求或DMA请求。
- ⑤ 程序中断方式由于软件额外开销时间比较大，因此传输速度最慢；程序查询方式软件额外开销时间基本没有，因此传输速度比中断快；DMA方式基本由硬件实现传送，因此速度最快；
- ⑥ 程序查询接口硬件结构最简单，因此最经济；程序中断接口硬件结构稍微复杂一些，因此较经济；DMA控制器硬件结构最复杂，因此成本最高；
- ⑦ 程序中断方式适用于中、低速设备的I/O交换；程序查询方式适用于中、低速实时处理过程；DMA方式适用于高速设备的I/O交换。

第五章 输入输出系统

- DMA访问主存的优先级和CPU访问主存的优先级哪一个高？
  - 一般情况下，如果DMA和CPU同时访问主存时，那么 DMA的优先级要比CPU的优先级高。
  - 主要原因在于DMA模块所连接的是高速外设，以数据块的方式与主存交换数据，若不及时处理，容易造成信息丢失。换言之，I/O访存有时间要求，前一个I/O数据必须在下一个访存请求到来前存取完毕。
  - 而对于CPU而言，只是访问主存延缓了一个存取周期而已，指令和数据仍然保存在主存中，不会造成信息丢失。

第六章 计算机的运算方法

- 无符号数：所有二进制数据位数均用来表示数值本身，没有正负之分
- 有符号数：其二进制数据位，包括符号位和数值位。计算机中的带符号数据又称为机器数。
- 机器数：把正负符号代码化，并保存在计算机中的数据。
- 真值：是指机器数所真正表示的数值，用数值并冠以+、-符号的方法来表示。
- 机器数的编码方法：原码、反码、补码、移码。

第六章 计算机的运算方法

- 原码编码方法
  - 原码为符号位加上数的绝对值，0正1负；
  - 原码零有两个编码，$[+0]_{原}=0.000\cdots0$ $[-0]_{原}=1.000\cdots0$
  - 若原码整数的位数是8位，其表示范围：-127 ～ 127
- 补码编码方法
  - 正数的补码在其二进制代码前加上符号位0
  - 负数的补码是将二进制代码前加0后，再全部按位取反，然后在最低位上加1
  - 0的补码表示只有一种形式，$[+0]_{补}=[-0]_{补}=0.000\cdots0$
  - 若补码整数的位数是8位，其表示范围：-128 ～127
  - 由$[X]_{补}$求$[-X]_{补}$，将$[X]_{补}$连同符号一起将各位取反，末位再加1

第六章 计算机的运算方法

- 逻辑移位：无符号数移位
  - 逻辑左移：各位依次左移，末位补0，最高位移丢。
  - 逻辑右移：各位依次右移，最高位补0，最低位移丢。
- 算术移位：有符号数移位，符号位必须要保持不变
  - 补码左移：符号位不变，数值部分依次左移，高位移丢，低位补0
  - 补码右移：符号位不变，数值部分依次左移，低位移丢，高位补符号位

第六章 计算机的运算方法

- 补码加减法的公式:
  - $[x+y]_{补} = [x]_{补} + [y]_{补}$
  - $[x-y]_{补} = [x]_{补} + [-y]_{补}$
  - 不需要事先判断符号，符号位与码值位一起参加运算
  - 符号位相加后若有进位，则舍去该进位数字
- 补码加减运算结果判溢出:
  - 单符号位判溢出：当最高有效位产生的进位和符号位产生的进位不同时，加减运算发生了溢出。
  - 双符号位判溢出：采用变形补码运算，当运算结果两位符号相同时无溢出，不同时有溢出，其中高位符号位表示真正的符号，01表示上溢，10表示下溢。

第六章 计算机的运算方法

- 一个机器浮点数由阶码E和尾数M及其符号位组成。约定：
  - 尾数M：用定点小数表示，给出有效数字的位数，决定了浮点数的表示精度
  - 阶码E：用整数形式表示，指明小数点在数据中的位置，决定了浮点数的表示范围
- 浮点数的一般形式为：

E_s	$E_1\ E_2\ \cdots\cdots\ E_m$	M_s	$M_1\ M_2\ \cdots\cdots\ M_n$
阶符	阶码	数符	尾数

第六章 计算机的运算方法

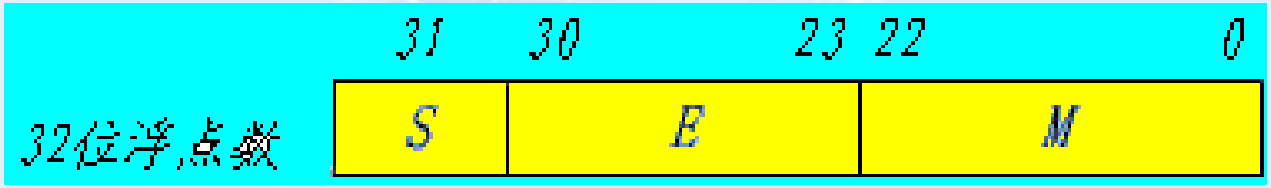
- 设浮点数的格式为：阶码5位，包含一位符号位，尾数5位，包含一位符号位，阶码和尾数均用补码表示，排列顺序为：

阶符1位	阶值4位	数符1位	数值4位

- 则按上述浮点数的格式：
  - ① 若X=15/32，Y=-1.75，则求X和Y的规格化浮点数表示形式。
  - ② 若数Z的浮点数的16进制形式为0ABH，求Z的十进制的真值。
- $X = 15/32 = 0.01111\ B = 0.1111 \times 2^{-1}$ $[X]_{浮} = 1,1111\ \ 0.1111$
- $Y = -1.75 = -1.11\ B = -0.1110 \times 2^{+1}$ $[Y]_{浮} = 0,0001\ \ 1.0010$
- $[Z]_{浮} = 0ABH = 00\ 1010\ 1011$
- $[Z]_{浮} = 0,0101\ \ 0.1011$
- $Z = 0.1011 \times 2^{+5} = 10110$ $Z = 22$

第六章 计算机的运算方法

- IEEE754标准　32位浮点数格式
  - S数的符号位，1位，在最高位，0表示正数，1表示负数。
  - M是尾数，23位，在低位部分，采用原码(纯小数)表示
  - E是阶码，8位，采用移码表示。移码比较大小方便。
  - 规格化：尾数域最左位(最高有效位)总是1，故这一位不予存储，而认为隐藏在小数点的左边。
  - 将浮点数指数真值e变成阶码E时，应将指数e加上一个固定偏移值127(01111111)，即E=e+127。
  - 一个规格化的32位浮点数x的真值表示为$x=(-1)^S\times(1.M)\times2^{E-127}$



第六章 计算机的运算方法

- 将数$(-1.25)_{10}$转换成IEEE754 标准的32位浮点数的二进制格式
  - 首先分别将整数和分数部分转换成二进制数：
  - $(-1.25)_{10} = -1.01$
  - 然后移动小数点，使其在第1，2位之间
  - $-1.01 = -1.01 \times 2^0$
  - e=0 于是得到：S=1，E=0+127=127=01111111，M=01
  - 最后得到32位浮点数的二进制存储格式为：
  - 1011 1111 1010 0000 0000 0000 0000 0000 $=(BFA00000)_{16}$

第六章 计算机的运算方法

- 浮点数加减运算的步骤：
  - 对阶、尾数求和、规格化、舍入、溢出判断
- 对阶
  - 在浮点运算作加法或减法时，若两数阶码不等，需要对阶操作。
  - 在浮点数的表示中，尾数采用定点小数，故阶码的大小直接反映尾数有效值小数点的实际位置，因此，当两浮点数的阶码不等时，尾数部分无法直接进行加减运算，需要先进行对阶操作。
  - 对阶操作采用小阶向大阶看齐的原则，原因是：如果是大阶向小阶看齐，那么随着大阶码的值减少，为保持浮点数的值不变，则其尾数必须左移相应的位数，有可能发生符号位及尾数高位丢失的错误，这是不允许的。而增大小阶码同时其尾数右移，有可能发生尾数低位的丢失，这只影响精度，不会产生错误
  - 对阶方法：将小阶加1，其尾数相应右移1位，直至两数阶码相等。

第六章 计算机的运算方法

- 规格化
  - 为了提高数据的表示精度，当尾数值不为0时，其绝对值|M|≥0.5，即尾数绝对值域的最高有效位应为1，否则通过修改阶码同时左右移小数点办法，使其变成这一表示形式，称为浮点数规格化表示。
  - 原码：不论正数、负数，第一数位为1
  - 补码：符号位和第 1 数位不同
- 舍入处理（对阶和右规时）
  - 0舍1入：如 丢弃的最高位为1，进1
  - 恒置1法：尾数的末位恒置1
- 溢出判断
  - 阶码溢出：上溢按中断溢出处理，下溢按机器零处理
  - 尾数溢出：需要右归，最低位移出，要进行舍入处理

第七章 指令系统

- 寻址方式：确定本条指令的操作数地址，以及下一条将要执行的指令地址的方法。寻址方式与硬件结构紧密相关，而且直接影响指令格式和指令功能
- 间接寻址：倘若指令字中的形式地址不直接给出操作数的地址，而是指出操作数有效地址所在的存储单元的地址，也就是说有效地址是由形式地址间接提供，即为间接寻址
- 基址寻址：操作数有效地址等于形式地址加上基址寄存器的内容；基址寄存器的内容由操作系统给定，且在程序的执行过程中不可变，支持多道程序技术的应用
- 变址寻址：操作数有效地址等于形式地址加上变址寄存器的内容，变址寄存器的内容由用户给定且在程序的执行过程中可变，常常用于处理数组程序

第七章 指令系统

- 基址寻址和变址寻址的特点：
  - 两种寻址方式都可以扩大指令寻址范围，计算操作数有效地址的方法类似，都等于形式地址A加上寄存器的内容
  - 基址寻址的基准地址由基址寄存器给出，通常由操作系统或管理程序设定，地址改变反映在形式地址A的取值上
  - 而变址寻址的基准地址由形式地址A给出，地址改变反映在变址值的自动修改上，变址值由变址寄存器给出，其内容通常是由用户设定
  - 基址寻址面向系统，解决程序在主存中动态重定位问题，变址寻址面向用户，适用于数组或字符串处理

第七章 指令系统

- 某计算机指令系统具有直接、间接、变址、基址、相对、立即等六种寻址方式，请分析哪一种执行时间最短，哪一种执行时间最长，为什么？哪一种便于程序的浮动，哪一种适合处理数组问题？
- 立即寻址指令执行时间最短，因为立即数是由指令直接给出，故在指令执行阶段不需要访问主存。间接寻址指令执行时间最长，因为指令的地址字段给出的形式地址不是操作数的真实地址，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)，因此，在指令执行阶段需要多次访存。
- 相对寻址把程序计数器PC的内容（即当前指令的地址）加上指令字中的形式地址A而形成操作数的有效地址，即EA=(PC)+A。相对寻址常被用于转移类指令，其中转移后的目标地址与当前指令有一段距离，称为相对位移量，它由指令形式地址A给出，可正可负，通常用补码表示。在相对寻址中，转移地址不固定，它随着PC值的变化而变化，因此，无论程序在主存的哪段区域，都可以正确运行，对于编写浮动程序特别有利。
- 变址寻址时，其操作数有效地址EA等于指令的形式地址A与变址寄存器IX的内容相加之和，即EA=(IX)+A。变址寄存器是面向用户的，在程序执行过程中，变址寄存器的内容可由用户改变（作为偏移量），而形式地址A不变（作为基地址）。在数组处理过程中，可设定A为数组的首地址，不断改变变址寄存器IX的内容，便可很容易形成数组中任意数据的地址。故适合编制循环程序，处理数组问题。

第八章 CPU的结构和功能

- 主频：CPU工作的时钟频率，即单位时间内发出的脉冲数
- 时钟周期：是指计算机主时钟的周期时间，是计算机运行时最基本的时序单位，是控制计算机操作的最小时间单位，通常时钟周期等于主频的倒数。
- 机器周期：也称CPU周期，计算机为了便于管理，常把一条指令的执行过程划分为若干个阶段如取指、译码、执行等，每一阶段完成一个基本操作。我们把完成一个基本操作所需要的时间称为机器周期，通常等于取指时间。
- 指令周期：是取出一条指令并执行这条指令的时间。一般由若干个机器周期组成，是从取指令、分析指令到执行完所需的全部时间。
- CPI：执行一条指令所需时钟周期数
- 1GHz=1000MHz、1MHz=1000KHz、1KHz=1000Hz
- 1s=1000ms、1ms=1000μs、1μs=1000ns

第八章 CPU的结构和功能

- CPU五大功能
  - 指令控制：用于控制指令程序的顺序执行
  - 操作控制：负责管理并产生每条指令所需操作信号
  - 时间控制：对各种操作加以时间的实施控制
  - 数据加工：对数据进行算术运算和逻辑运算处理
  - 中断处理：处理响应中断
- 一个完整的CPU指令周期所包含4个机器周期：
  - 取指周期：完成取指令和分析指令的操作
  - 间址周期：访问存储器取出操作数的有效地址
  - 执行周期：完成取出操作数、执行指令的操作
  - 中断周期：保护程序断点、寻找中断服务程序入口地址、关中断

第八章 CPU的结构和功能

- CPU中各个寄存器的作用如下：
  - PC：程序计数器，存放当前欲执行指令的地址，并可自动计数形成下一条指令地址。
  - IR：指令寄存器，存放当前正在执行的指令的寄存器。
  - MAR：存储器地址寄存器，用来存放CPU欲访问存储单元地址。
  - MDR：存储器数据寄存器，用来存放从某存储单元读出、或写入某存储单元的数据。
  - ACC：累加寄存器，运算前存放操作数、运算后存放运算结果。
  - PSW：程序状态字寄存器，记录由算术指令和逻辑指令运行或测试结果建立的各种状态信息。

第八章 CPU的结构和功能

- 向量中断：中断源在提出中断请求同时，通过硬件向主机提供中断服务程序入口地址，即向量地址。
- 中断隐指令：是在计算机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，其功能包括保护程序断点、寻找中断服务程序的入口地址、关中断等功能。
- 多重中断：是指CPU执行某个中断服务程序的过程中，发生了更高级、更紧迫的事件，CPU暂停现行中断服务程序的执行，转去处理该事件的中断，处理完返回现行中断服务程序继续执行的过程。
- 中断屏蔽：是指当产生中断请求后，用程序方式有选择地封锁部分中断，而允许其余部分中断仍得到响应，称为中断屏蔽。实现方法是为每个中断源设置一个中断屏蔽触发器来屏蔽该设备的中断请求。

第八章 CPU的结构和功能

- CPU响应处理一次中断的具体步骤：
  - ①关中断
  - ②保存断点
  - ③识别中断源，将向量地址送PC
  - ④保存现场
  - ⑤中断事件处理（开中断、执行中断服务程序、关中断）
  - ⑥恢复现场
  - ⑦开中断
  - ⑧中断返回
  - ①～③由硬件完成，即中断隐指令
  - ④～⑧由中断服务程序完成

第八章 CPU的结构和功能

- I/O设备向CPU提出中断请求的条件是：
  - 在I/O接口中，设备工作完成状态为1（D=1），中断屏蔽码为0（MASK=0），且CPU查询中断时，中断请求触发器状态为1（INTR=1）。
- CPU响应I/O设备中断请求的条件和时间是：
  - 中断允许状态为1（EINT=1），且至少有一个中断请求被查到，则在一条指令执行完时，响应中断。

第八章 CPU的结构和功能

- 计算机为了管理中断，在硬件上设有专门处理中断的机构——中断系统。
- 中断系统通常包括：中断请求寄存器、中断优先级排队器、向量编码器、中断允许触发器（EINT）、中断标记触发器（INT）、中断屏蔽触发器（寄存器）等。
- 中断系统各个部件的功能如下：
- 中断请求寄存器——对中断源发来的一过性中断请求信号进行登记；
- 中断优先级排队器——对同时提出的多个中断请求信号进行裁决，选出一个最紧迫的进行响应；
- 向量编码器——向量中断时，用来产生向量地址；
- 中断允许触发器（EINT）——CPU中的中断总开关，完成开、关中断状态的设置；中断标记触发器（INT）——用来建立中断周期状态。INT=1，表示进入中断周期，即开始执行中断隐指令；
- 中断屏蔽触发器——对于可屏蔽的中断源进行开、关中断操作，可视为各中断源的中断分开关；
- 采用程序中断技术时，指令系统中往往有相关指令支持。常见的指令有：开中断、关中断、中断返回等。

谢谢大家


AUSTRALIA
OCEANIA
Northern Territory
Queensland
New South Wales
Victoria
NEW ZEALAND
VANUATU
Port-Vila
TONGA
Chesterfield
New Caledonia
Brisbane
Sydney
Canberra, ACT
Lord Howe
Norfolk I.
Tasman Sea
Coral Sea
Cairns
Alice Springs
Darwin
Adelaide
Hobart