计算机组成

- 第一章：计算机系统概述
  - 运算器、控制器、控存、CPU功能?
    - 运算器：算数、逻辑运算，暂存中间结果
    - 控制器：控制程序运行（取址、分析、执行）、数据输入输出、异常处理（中断、DMA)
    - 控存：解释指令，并发出操作命令执行指令
    - CPU：指令控制、操作控制、时间控制、数据加工、中断处理
  - 主存、控制器、运算器、CPU、主机组成?
    - 主存=存储体+MAR+MDR
    - 控制器=CU+IR+PC
    - 运算器=ACC+MQ+X操作数寄存器+ALU+PSW
    - CPU=运算器+控制器+MAR+MDR
    - 主机=CPU+内存
  - CPU中的寄存器?
    - 专用寄存器
      - 程序计数器PC、指令寄存器IR
      - 地址寄存器AR、数据寄存器DR
      - 程序状态字PSW
    - 通用寄存器X：累加寄存器
  - 计算机如何区分指令和数据?

- 通过不同的时间段，取指阶段取出的是指令，执行阶段取出的是数据
- 通过地址源区分，由PC提供存储单元地址，取出的是指令，指令地址码部分提供存储单元地址，取出的是操作数

- 编译、解释、汇编程序
  - 编译程序：高级语言——机器语言/汇编语言，一次性
  - 解释程序：高级语言——机器语言，一句一句解释
  - 汇编程序：汇编语言——机器语言
- 存储单元、存储字、存储字长、存储体?
  - 存储字：一个存储单元中存放的所有二进制数据
  - 存储单元：存储一个存储字、并具有特定存储地址的存储单位
  - 存储字长：存储字中存储的所有二进制数据的位数
  - 存储体：有多个存储单元构成的存储器件
- 冯诺依曼体系特点?
  - 二进制表示指令（操作码+地址码）和数据
  - 指令由操作码和地址码组成
  - 运算器为中心，存储程序，顺序执行

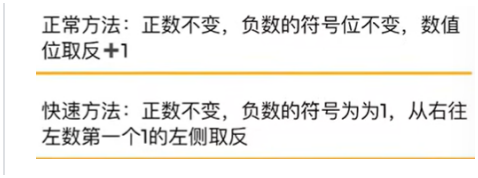
- 硬件系统由存储器、运算器、控制器、输入设备、输出设备
- 冯诺依曼体系结构?
  - 以二进制和存储程序为核心的计算机系统结构，称为“存储程序计算机”
  - 冯诺依曼体系以运算器为中心，现代计算机以存储器为中心
- 计算机分级层次结构?
  - 数字逻辑级、微程序级（控存）、一般机器级、操作系统级（虚存、辅存）、汇编语言级、高级语言级
- 计算机软件的定义及分类
  - 计算机数据和指令的集合
  - 系统软件（汇编/编译/解释程序；DBMS；OS）、应用软件
- MAR/MDR区别?
  - MDR主存数据寄存器：从主存取出数据、要输入主存数据
  - MAR主存地址寄存器：数据来源地址、数据目的地址
  - MAR位数——主存地址单元数
  - MDR位数——一个存储单元多大——存储字长
- 计算机性能指标

- 机器字长：CPU能进行多少位二进制的并行计算/数据总线位数/通用寄存器位数
- 主频（时钟频率）：1hz每秒一个时钟周期
- 主存容量
- 运算速度：
  - CPU执行时间 = 周期数/主频 =（指令条数IC ×CPI）/主频
  - CPI（一条指令几个周期）=时钟周期数/IC
  - MIPS每秒多少百万指令 = 主频/ (CPI×10^6)
  - MFLOPS每秒多少百万浮点运算
  - 吞吐量（单位时间处理请求数）
- 时钟周期?
  - 节拍，CPU最小时间单位，CPU脉冲信号的宽度，只完成一个最基本的动作
- 机器周期?
  - =CPU周期，完成一个阶段 JC
  - 通常以存取周期/读取一个指令字的最短时间为基准（由于CPU访问一次内存花费时间较长）
- 指令周期?
  - 取出、执行完一条指令，包括多个机器周期，指令不同，所需的机器周期也不同
- 存储周期、存取时间?
  - 存取周期：两次独立访存的最小时间间隔

- 存取时间：启动一次存储器操作，到完成该操作的时间
- 存取时间+恢复时间

- 总线周期?
  - CPU通过总线对IO设备或主存的一次访问
  - IO设备，CPU，主存都挂在总线上

- 第二章：数据的表示和运算（PC中保存补码）
  - 原反补移 范围?
    - 原码反码对称
    - 补码移码左边多1，0的表示唯一
  - 原补转换?
    - 原码——>补码



    - [x]补——[-x]补：全部取反+1
  - 机器码比大小?
    - 原码：符号位、数值位分开考虑
    - 反码、补码：正负分别看，数值位大的大
    - 移码：整体直接比，大的大，小的小
  - 三种移位方法?
    - 逻辑移位（无符号数）：补0
    - 算数移位（有符号数）：左移补0，右移补符号位

- 循环移位
- 符号扩展?
  - 无符号数使用零扩展即可
  - 有符号数，符号扩展复制符号位
- 大端存储、小端存储?
  - 大端：高位——低地址
  - 小端：高位——高地址，高位是左边第一位
  - 大端相反，小端相同
  - 大端方便人类阅读，小端方便机器处理（机器按地址递增顺序读入，先读入个位）大人
- 边界对齐存储?
  - 数据储存时，按照数据类型大小 的整数倍，进行地址分配，方便取指令
  - char存在任意地址编号上，short存在0、2、4，int存在0、4、8
- C语言char, short , int位数?
  - char 1字节，short 2字节，int 4字节，一字节8bit
  - 计算机按字节编址，可按字、半字、字节寻址
- 溢出（运算结果超过机器数可表示范围）判断?
  - 双符号位：00，11不溢出；01正溢出，10负溢出
  - 一位符号位直接判断：符号相同两数想加，结果符号相反（只有正+正才会上溢出，负+负才

会下溢出）

- 一位符号位进位判断

	符号位的进位C_S	最高数值位的进位C_1
上溢	0	1
下溢	1	0

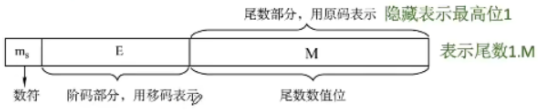
- 标志位含义
  - 标志位含义：溢出、符号标志只对有符号数有意义 衣服

OF（Overflow Flag)溢出标志。溢出时为1,否则置0。
SF（Sign Flag）符号标志。结果为负时置1,否则置0.
ZF（Zero Flag)零标志，运算结果为0时ZF位置1,否则置0.
CF（Carry Flag)进位/借位标志，进位/借位时置1,否则置0.

- 串行、并行进位加法器?
  - 串行...：n个全加器串联，进行两个n位数的相加，信号逐级产生，每一级进位依赖于前一级进位输出；设计简单，但总延迟时间长
  - 并行...：在低位数据运算之前，高位已知进位多少；可实现同步运算，提高运算速度
  - 影响速度的关键因素是进位传递速度
- 进位产生函数、进位传递函数的含义?
  - 产生函数：两个输入都为1，一定产生进位 G=X·Y
  - 传递函数：一个输入为1，且低位进位为1，则产生进位 P=X+Y
- 三种校验技术?
  - 奇偶校验码?
    - 一位或奇数位检错，无纠错能力
    - 使得整个校验码中1的个数为奇数或偶数

- 海明码?
  - 多重奇偶校验，实现一位纠错检错，两位检错
  - 分块校验，逐步确定出错位置
- CRC循环冗余校验码?
  - 发现并纠正连续多位错误
  - CRC码（数据码+冗余码）与生成多项式相除，余数为0，数据正确，不为0有错，且余数与CRC码出错位的对应关系一定，可以纠错
  - 生成多项式的满足的条件?
    - 1位出错，余数不为0；不同位出错，余数不同；对余数模2除，应循环
- 移码定义?
  - 移码=真值+偏置值（n位移码偏置值为2^n-1）
- 浮点数乘除累加、移动次数（n数值位）？
  - 原码一位乘：n次累加，n次右移
  - 补码一位乘：校正法、booth法均为n+1次累加，n次右移
  - 补码两位乘（booth法)
    - 奇数数值位，(n+1)/2次求部分积，最后一次右移一位

- 偶数数值位，增加符号位，(n/2+1)次求部分积，不右移；末位补0，(n/2+1)次求部分积，最后一次右移一位
- 原码一位除：上商n+1次，左移n次
  - 恢复余数法
  - 加减交替法：加减n+1/n+2次，左移n次，最后余数为负需要恢复余数
- 补码一位除法：
  - 加减交替法：加减n+1次，左移n次
- IEEE 754
  - 移码表示阶码：n位移码偏置值为2^(n-1)-1



例：将十进制数 -0.75 转换为 IEEE 754 的单精度浮点数格式表示。

$(-0.75)_{10} = (-0.11)_2 = (-1.1)_2 \times 2^{-1}$

数符 = 1
尾数部分 = .1000000..... （隐含最高位1）
阶码真值 = -1
单精度浮点型偏移量 = 127D
移码 = 阶码真值+偏移量 = -1 + 111 1111 = 0111 1110 （凑足8位）

→ 1 01111110 10000000000000000000000

- 浮点数加减运算流程
  - 对阶：小阶向大阶看齐
  - 尾数求和
  - 规格化：原码（小数点后第一个为1），补码（符号位与小数点后第一位不同）
  - 舍入：尾数右移时，移去最高位为0舍去；最高位为1，末尾加1

- 溢出判断：溢出看阶码（符号位为01或10），上溢出中断，下溢出零处理
- 原码一位乘、补码一位乘?
  - 原码一位乘：符号单独算，部分积初始为0；Y不带符号，为1加x，为0不加；右移
  - 补码一位乘
    - 校正法：Y为正数，按原码一位乘计算即可，Y为负数，最后要加上[-X]补
    - booth比较法：Y单符号位，辅助位0，每次加x补码、[-x]补码，右移
- 原码一位除、补码一位除?
  - 原码一位除
    - 恢复余数法：先减[|Y|]补，正商1，负商0，再恢复余数，一次操作逻辑左移一次，最后一次不左移
    - 加减交替法：先减[|Y|]补，正商1，负商0，一次操作左移一次，为负下次加，为正下次减，最后为负要再加
  - 补码一位除：
    - 加减交替法：第一次都加[|Y|]补，X、Y都用双符号位，同号商1，下次减，异号商0下次加，每次操作移位，最后一位恒商1
- 浮点数（小数点位置漂浮不定）？

- 1、浮点数用科学计数法表示，在计算机中存储时，只需要确认符号位、指数、尾数、基数即可
- 2、为统一格式，提出IEEE754浮点数标准，提供两种浮点数格式：单精度浮点数float 32位，双精度浮点数double 64位
- 3、浮点数存在精度损失（十进制小数无法精确转换成二进制）
- 4、浮点数范围、精度大，小数在计算机中一般用此存储

- 定点数、浮点数?
  - 1、定点数（小数点位置固定不变），浮点数（小数点位置漂浮不定）
  - 2、相同字长浮点数、定点数，定点数精度更高
  - 3、定点数所需硬件简单，浮点数复杂
  - 4、定点运算超出范围即溢出，浮点运算需要规格化后看阶码溢出才溢出

- 第三章：存储系统
  - 主存性能指标?
    - 主存容量
    - 存取时间：启动一次存储器操作到完成该操作所用的时间
    - 存储周期：两次独立存取操作最短间隔

- 存储体系?
  - 把>=2种不同存储容量、存取速度、价格的存储器组成层次结构
  - 并通过管理软件、辅助硬件组合成有机整体
  - 现在采用cache-主存-辅存体系
- 主存储器分类
  - RAM：
    - SRAM：双稳态触发器；加电源信息一直保持；集成度低、价格高、行列地址同时送、快、cache
    - DRAM：栅极电容；动态刷新；集成度高、价格低、行列地址两次送、慢、主存
  - ROM：
    - 不可在线改写内容的ROM：MROM（出厂写入、只读）、PROM（一次写入、无法改变）、EPROM（Erasable Programmable可读可写、写入时间长）、EEPROM（electrically erasable，programmable可电擦除）
    - 闪存（相对于EEPROM擦除重写快、便宜集成度高、电擦除、用作BIOS）
- DRAM三种刷新（定期向电容补充电荷）方式?
  - 集中刷新：一个刷新周期（上一次对整个存储器刷新结束到下一次刷新一遍为止）内，留出

一段时间对整个存储器全部刷新一遍（不能读写，死时间）
    - 分散刷新：每行刷新分散到每个存取周期
    - 异步刷新：每隔一段时间刷新一行（最大刷新间隔/行数）
- RAM和ROM?
    - RAM随机存取，断电易失，指内存
        - 外存数据读入内存才能处理
    - ROM可随机存取，断电非易失，指外存（结构更简单、位密度高） while
- 主存容量扩展?
    - 位扩展：每个芯片接一根数据线，使得数据位数与CPU数据线数相等 （芯片并联）胃病
    - 字扩展：地址线增加，选芯片（芯片串联）
    - 字位同时扩展
- 容量扩展算地址线?
    - 数据线，地址线，片选线，读写控制线
    - 低位连地址线，高位连片选线
- 缓解主存、CPU速度不匹配问题（提高访存速度）的方法?
    - 提高存储芯片性能
        - 双端口存储器：一个存储器有两套独立的读写控制电路，支持两个CPU同时访存，但可能会发生冲突

- 多体交叉存储器：由多个存储体组成，每个存储体有自己的读写控制电路、寄存器
  - 高位交叉编址（顺序存储器）：模块内地址连续，体号+体内地址
  - 低位交叉编址：连续地址分布在相邻存储体中，同一存储体内地址不连续，横向编址，并行存取，可以实现流水线；体内地址+体号（存取时间结束即可读下一块，不用等恢复时间）
  - 为使流水线不间断，模块数>=存取周期/存取时间
- 优化体系结构
  - cache：cache的存储和管理完全由硬件实现
- 全相联映射、直接映射、组相联映射含义、优缺点
  - 全相联：主存块可以放到cache任意位置
    - 优点：cache存储空间利用充分，命中率高
    - 缺点：查找标记慢，可能需要对比所有行的标记
  - 直接：主存块只能放到特定cache行
    - 优点：速度最快
    - 缺点：cache存储空间利用不充分，命中率低
  - 组相联：主存块可以放特定分组任意位置

    - 优点：综合效果最好，目录表短，实现成本低
- 什么是cache命中率?
    - CPU要访问的信息已在cache中的比例
- 影响cache命中率的因素?
    - cache容量：容量越大，命中率越高
    - 块大小：块增大，先上升后下降（块增大，因局部性原理命中率提高，太大，装入缓存的有用数据少于被替换掉的有用数据）
    - 地址映像方式：全相联命中率最高，但硬件多不采用；直接映像命中率最低，但简单；组相联合适，且组数增加，命中率降低（分组增加、组内块数减少、主存某一块映射到的cache块数减少、命中率下降）
    - 替换算法：LRU替换算法命中率高于FIFO
- cache替换算法?
    - 随机算法RAND
    - 最近最久未使用LRU
    - 先进先出算法FIFO
    - 近期最少使用LFU
- cache一致性问题?
    - 某段时间内，主存某单元内容和cache对应单元内容不同
    - 解决方法：cache更新算法

- 写命中：全写法（每次修改、写回主存和cache）、回写法（先写cache、cache换出再写主存、需要设置脏位）
- 写不命中：写分配（写主存、加载主存块到cache）、非写分配（写主存、不调块）

- 全写+非写分配原因?
  - 若搭配写分配，不命中、写主存调块、还想改该块、命中、每次该都要全写
  - 不命中、写主存不调块、还想改该块、依然不命中、只改主存即可
- 回写+写分配原因?
  - 搭配非写分配，不命中、写内存不调块、每次都不命中、修改同一行都要访问主存
  - 搭配写分配，不命中、写主存调块、修改同一行、改cache即可

- 数据cache行结构?
  - 有效位+脏位+替换控制位+标记位+数据
  - 有效位：在不在内存中
  - 脏位：回写法一位，换出时要写回
  - 替换控制位：控制替换算法，直接映射、随机替换算法不需要
- cache/高速缓存/数据缓存?
  - 位于CPU内部，由SRAM制成
  - 存储内存中部分数据的副本

- 虚拟存储器容量=主存+辅存
- 引入cache为了解决CPU与主存速度不匹配问题

- 快表/TLB/地址缓存?
  - 快表是页表数据的一部分，页表位于内存中
  - 引入快表为了加快地址转换
- 相联存储器?
  - 按内容寻址的存储器，查询速度快
  - cache和快表都属于相联存储器
- 基于快表的虚拟存储器访问顺序?
  - TLB，页表，cache，主存
- 如何提高页式虚拟存储器主存利用率
  - 增加主存容量（虚存=主存+辅存，利用率=主存/虚存）
  - 主存容量一定，页面增大，命中率先增后减（相邻两次访存逻辑地址间距小于页面大小，页面大，在同一页概率大，大于页面大小，页数少，概率减小）
- cache与虚存对比?
  - cache解决速度问题，虚存解决容量问题
  - cache所有程序员不可见，虚存对系统程序员可见
  - cache全硬件实现，虚存由硬件和OS共同实现
  - 虚存不命中对系统影响更大

- SSD固态硬盘特性?
  - EP-ROM可电擦除，随机访问
  - 读页写块，读快写慢，但写多了会磨损
- 第四章：指令系统
- 取指令的微操作?
  - 程序计数器PC内容送到MAR（主存地址寄存器）
  - 主存读信号有效
  - 根据地址从主存中读指令到MDR（主存数据寄存器）
  - 程序计数器自增
- load/store指令的区别?
  - load加载，读内存，内存读到寄存器
  - store存储，写内存，寄存器写入内存
- 指令周期各个阶段的任务?
  - 取指周期：根据PC，从主存中取指令放入IR中，(PC)+1
  - 间址周期：取操作数的有效地址
  - 执行周期：取操作数，并根据指令操作码，产生运算结果
  - 中断周期：改栈顶指针、存PC、PC指向中断服务程序入口
- CISC和RISC?
  - CISC复杂指令集系统

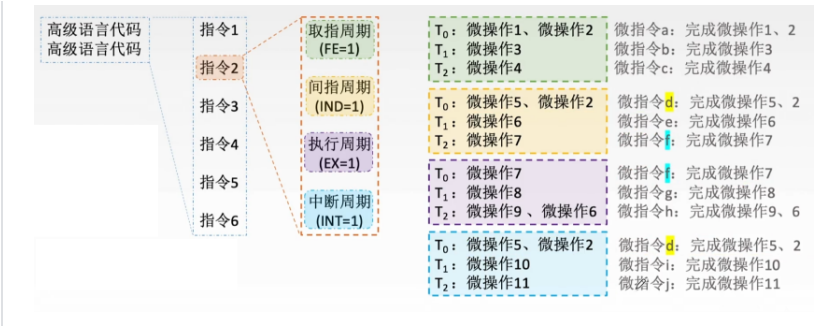
- 指令多，长度不定，使用频率差异大
- 可以实现流水线；微程序控制，x86
- RISC精简指令集系统
  - 指令少，长度短且固定，使用频率高
  - 必须采用流水线；硬布线/组合逻辑控制，ARM
  - 只允许Load/Store指令访问存储器
- 指令的两种寻址方式?
  - 指令寻址：寻找下一条要执行指令的地址
    - 顺序寻址：PC+1(1为一个指令字长)
    - 跳跃寻址：指令地址由指令本身给出
  - 数据寻址：寻找操作数地址
    - 隐含寻址
    - 立即寻址：所需操作数直接包含在指令代码中
    - 直接寻址、一次间接寻址
    - 寄存器寻址、寄存器间接寻址
      - 优点：扩大指令字的寻址范围、缩短指令字长度、减少访存次数
    - 相对寻址（相对于下一条指令）、基址寻址（多道程序设计）、变址寻址（数组问题）变数
    - 堆栈寻址
- 基本指令系统五大类型

- 数据传送类MOV、运算类ADD
- 程序控制类JMP（改变程序执行顺序，转移+程序调用+循环控制）、输入输出LOAD
- 处理机控制和调试（软硬件调试）

- 采用不同寻址方式的原因?
  - 缩短指令长度
  - 扩大寻址空间
  - 提高编程灵活性

- 第五章：中央处理器
  - 控制器的时序控制方式?
    - 同步控制：指令所需机器周期数、节拍数固定不变
    - 异步控制：控制器发出控制信号，执行部件执行完应答，不需要统一周期/节拍控制，但需要应答电路
    - 联合控制：同步异步结合
    - 人工控制
  - 两类数字电路?
    - 组合逻辑电路：任意时刻的输出只取决于该时刻的输入，与电路原来的状态无关
    - 时序逻辑电路：任意时刻的输出不仅取决于当时输入，也取决于原来的电路状态（有记忆单元，即包含触发器）

- 运算器是组合逻辑电路，控制器是时序逻辑电路
- 两种控制器?
  - 硬布线控制器：又称组合逻辑控制器，微操作控制信号由组合逻辑电路产生；执行速度快；难扩充；适用于RISC CPU
  - 微程序控制器：微操作控制信号以微程序形式放在控存中；执行速度慢；易扩充；适用于CISC CPU
- 微程序控制器思想
  - 每条机器指令编写一个微程序
  - 一个微程序有多个微指令
  - 一个微指令对应一个或多个微操作
  - 一个微命令对应一个微操作



- 程序、微程序的区别?
  - 程序由程序员编写，由指令组成
  - 微程序由微指令组成，用于实现指令功能
- 主存VS控存?
  - 主存存放程序和数据，CPU外部，RAM实现

- 控存存放微程序（中的微指令），CPU内部，ROM实现
- 微地址?
  - 存放微指令的控存地址
  - 操作码+微地址码（下一条微指令地址）
- 指令编码方式（如何对操作码进行编码）
  - 固定长编码：指令操作码等长——浪费很多信息量；使用频率不同的指令长度相同
  - 哈夫曼编码：概率高的指令用短码，概率低的用长码——操作码的平均长度最短、信息冗余量最小；形成的操作码很不规整
  - 扩展编码：结合定长编码和哈夫曼编码
  - 注：地址码的位数一般设置为0、1、2、3
- 微指令编码方式（如何对微指令的控制字段编码）
  - 直接编码：每个操作对应控制字段的一位。
  - 字段直接编码：控制字段分为多个小段，互斥微命令同一字段，相容不同——缩短指令字长，表示更多指令
  - 字段间接编码：一个字段中某些微命令需要另一个字段中某些微命令解释——进一步缩短指令字长
- 微指令两种格式?
  - 水平型微指令：一条指令定义多个微操作

- 优点：并行能力强；灵活性强；执行速度快
- 缺点：微指令字长、微程序短；用户难掌握**
- 直接编码，字段直接编码均属于
- 垂直型微指令：一条指令定义一个微操作
  - 优点：微指令字短、微程序长；用户好掌握
  - 缺点：并行能力弱；执行速度慢
  - 微操作码+目的地址+源地址
- 下一条微指令形成方式？
  - 断定方式：操作码+后继微指令地址；速度快、指令长
  - 计数器方式：操作码+标志位（顺序or跳转），简单、速度慢
    - 指令顺序放在控存中
  - 结合方式：操作码+标志位+后继微指令地址
- 机器指令、微指令区别？
  - 机器指令：CPU能直接识别并执行的二进制指令
  - 微指令：一个机器周期内，一组实现一定操作功能的、微命令组合，一条微指令对应一个或多个微操作
- 指令流水线？
  - 一种通过优化系统结构，提高处理机速度的方法

- 指令分阶段，每个阶段与其它阶段并行执行
- 影响流水线的因素
  - 资源冲突：多条指令 同一时刻 争用同一资源 互斥
  - 数据冲突：在一个程序中，一条指令执行完，才能执行后一条指令，但由于指令重叠操作，可能改变对操作数的读写访问顺序 同步
  - 控制冲突：流水线遇到转移指令等改变PC值的指令，造成断流
- 解决资源冲突?
  - 1、完成前一条指令对资源的访问时，先暂停（一个周期）取后一条指令的操作（取指和取操作数会发生冲突)
  - 2、设置两个独立的存储器存放操作数和指令
- 解决数据冲突?
  - 1、后推法：遇到数据相关，停顿后续指令的运行，直到前面的结果生成
  - 2、数据旁路：不必等某条指令执行结果送回寄存器后，再从寄存器中取出结果作为下条指令源操作数，而是直接将执行结果送到其他指令需要的地方
- 解决控制冲突?
  - 1、预测转移是否发生，尽早形成目标转移地址

- 2、预取转移成功、不成功两个方向的指令
- 流水线性能评估指标?
  - 吞吐率：单位时间内流水线完成指令，输出结果的数量
  - 加速比：m段流水线与等功能非流水线的速度之比
  - 效率：流水线处于工作时间的时空区/各段总时空区
- 怎么提高流水线性能?
  - 降低相关干扰
  - 流水线多发技术
  - 重新组织指令执行顺序
  - 好的指令调度算法、优化编译
- 三种流水线多发技术
  - 超标量流水：一个机器周期内发射多条指令，需要设置多个相同的部件（空间并行性）
  - 超流水线：一个机器周期内分时发射多条指令（时间并行性）
  - 超标量超流水：结合
- 第六章：总线（概念）
- 总线分类?
  - 片内总线：芯片内部（寄存器—寄存器，寄存器—ALU）

- 系统总线：数据总线（双向），地址总线（单向），控制总线（单向）
- 通信总线：连接计算机系统，或计算机系统与其它系统

- 为什么使用总线？总线两大特征？
  - 在冯诺依曼结构中，部件和部件之间都有单独连线，连线多而复杂、I/O设备的扩展困难，从而引入了总线连接方式，将所有设备连接在一组总线上，构成设备之间的共享传输通道。
  - 总线具有共享和分时两大特征，共享是指多个部件连接在一组总线上，都通过该总线进行数据的传输和交换；分时是指在同一时刻只能有一个部件使用总线传输信息、
- 引入总线结构的好处？
  - 1、简化了系统结构、减少了连线数目
  - 2、便于故障诊断和维修，同时也能降低成本
  - 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
- 总线宽度、总线带宽、总线复用、信号宽度？
  - 总线宽度：数据总线的根数，一般是8的倍数
  - 总线带宽：总线数据最大传输速率，总线上每秒能够传输的最大字节量。（=工作频率×总线宽度）

- 总线复用：一条信号线上分时传送两种信号
- 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。

- 总线操作流程?
  - 主设备请求总线
  - 总线仲裁：选个设备分配总线
  - 寻址：找从设备
  - 传送信息
  - 状态返回
- 总线定时协议?
  - 同步定时：公共时钟信号
  - 异步定时：基于应答方式的定时协议
- 总线仲裁/判优方式?
  - 集中仲裁方式
    - 链式查询：离总线近，优先级高，故障敏感
    - 计数器查询：总线控制部件维护一个计数器，每次从初值开始，与设备号对比，可通过更改初值改变设备优先次序
    - 独立请求：使用一个中心裁决器从请求总线的设备中选一个。各个设备独立请求总线，最快，线路复杂
  - 分布仲裁方式：每个设备有自己的仲裁器和仲裁号，与仲裁总线上的仲裁号比较优先级

- 第七章：IO系统

- **IO接口类型?**
  - **按照外设和接口侧的数据传送方式可分为：并行接口和串行接口**
  - **按主机访问IO设备的控制方式可分为：程序查询接口，中断接口、DMA接口**
  - **按照功能灵活性可分为：可编程接口、不可编程接口**
- **IO控制方式有哪些?**
  - **程序查询：一次传一个字；CPU轮询检查设备是否就绪**IO与CPU串行
  - **程序中断：一次传一个字；CPU启动外设，做其他事，外设准备好发出中断信号，CPU执行中断服务程序，进行一次数据的输入**
  - **DMA（高速设备与主机）：一次传一个或多个块；CPU接收DMA请求，控制启动DMA控制器，由DMA控制器与主存直接交互，一个一个字的传送，直到传完整块数据，再向CPU发出中断信号**IO与CPU并行
  - **通道控制：一次读一组块；CPU向通道发出一条IO指令，由通道控制**IO与CPU并行
  - **外围处理机方式**
- **IO接口作用、组成?**
  - **IO接口=IO控制器=设备控制器**
  - **作用：协调主机和外设之间的数据传输**

- 组成：包括控制逻辑和IO端口（数据缓冲、状态、控制寄存器）
- IO接口具体功能?
  - 数据缓冲寄存器：协调主机和外设的速度不匹配问题——数据缓冲
  - 状态寄存器：记录设备状态（错误、完成、就绪），供CPU查用——错误、状态检测
  - 控制寄存器：接收CPU发出的控制信号——控制和定时
  - 数据格式转换（串行数据并行数据互转）
  - 主机和设备的通信
- IO端口及编址方式?
  - 接口中可以读写的寄存器
  - 编址方式：独立编址（独立IO指令操作，外设地址与主存地址无关）、统一编址（使用访存指令完成IO功能）
- 什么是中断?
  - 计算机执行程序时，出现异常状况
  - 终止程序，处理异常，再恢复原程序执行
- 程序中断和子程序调用的区别?
  - 程序中断：
    - 发生时间随机；
    - 与主程序无关，平行关系；
    - 软硬件结合；

- 中断嵌套级数受优先级限制
- 子程序调用：
  - 发生时间已知；
  - 为主程序服务，主从关系；
  - 软件处理过程；
  - 子程序嵌套可以若干级
- 中断分类（王道分类）？
  - 内中断/异常/同步中断：中断信号来自CPU内部，与当前指令相关
    - 陷入：trap指令，自愿中断（由于系统调用引起处理机中断的指令）
    - 终止：由致命错误引起，不可修复，直接终止程序。比如：地址越界、算术溢出、非法操作码
    - 故障：由错误条件引起，可修复，比如缺页
  - 外中断/异步中断：中断信号来自CPU外部,与当前执行指令无关
    - IO设备：IO设备启动发出中断请求
    - 外部事件：通过键盘终止现行程序
- 中断分类（软硬分类）？
  - 软中断：软件程序引发的中断，如：中断指令、CPU运算错误、debug设置断点（可概括为执行异常、中断指令）

- 硬中断：外设引发中断，分为可屏蔽（比如接口发出中断请求，可以使用软件屏蔽）、不可屏蔽（比如电源掉电，不能使用软件屏蔽）
- 中断流程
  - 中断请求
  - 中断响应：满足响应条件（CPU开中断状态、一条指令已经执行完毕）、中断判优
  - 中断处理：中断隐指令、中断服务程序
- 中断隐指令定义？
  - CPU响应中断后、转去执行中断服务程序的、一系列操作，CPU内硬件执行
- 中断隐指令/中断响应/中断周期做哪三件工作
  - 关中断：中断触发器置0
  - 保存断点PC
  - 跳转到中断服务程序入口
- 单重中断、多重中断定义及处理流程？
  - 单重中断：CPU执行中断服务程序时不能被打断
  - 多重中断/中断嵌套：CPU执行中断服务程序时，可去响应级别更高的中断请求
  - 现场：存放程序执行到断点处的现行值

	单重中断	多重中断
中断隐指令	关中断	关中断
	保存断点（PC）	保存断点（PC）
	送中断向量	送中断向量
中断服务程序	保护现场	保护现场和屏蔽字
	-	开中断
	执行中断服务程序	执行中断服务程序
	-	关中断
	恢复现场	恢复现场和屏蔽字
	开中断	开中断
	中断返回	中断返回

- **中断屏蔽技术?**
  - **用于多重中断，调整中断源优先级**
  - **每个中断源有一个屏蔽字触发器，1表示屏蔽该中断源请求，所有触发器组合在一起称为屏蔽字寄存器，内容为屏蔽字（1越多优先级越高，至少一个1以屏蔽自身中断）**
- **多重中断需要哪些硬件?**
  - **中断请求寄存器**
  - **中断判优部件：屏蔽字寄存器（可调整优先级）or 硬件排队器（优先级固定）or 软件查询**
  - **中断服务寄存器（堆栈、向量地址形成部件）**
  - **控制逻辑**
- **中断判优实现?**
  - **屏蔽字寄存器（可调整优先级）**
  - **硬件排队器（优先级固定）**
  - **软件查询**
- **中断判优优先级**
  - **1、硬件故障最高级**

- 2、非屏蔽>可屏蔽，DMA中断>IO设备中断
- 3、高速设备>低速设备，实时设备>普通设备
- 输入设备>输出设备（大量输入不处理会导致信息丢失）

- 通道?
  - 专门负责输入输出的处理机，指令单一，与CPU共享内存
- 通道种类?
  - 字节多路通道：连接大量低速设备，各设备分时共享总通道
  - 选择通道：某时间段内选一个设备工作，适合数据块方式成组高速传输
  - 成组多路通道：结合
- DMA、通道控制的区别?
  - DMA由CPU控制传送数据大小，主存位置；通道方式则由通道控制
  - 每个DMA控制器对应一个设备；一个通道可控制多个设备
- DMA与CPU访存冲突的传送方式/DMA三种工作方式?
  - 停止CPU访问
  - 周期挪用
    - IO设备与CPU同时申请访存
    - CPU延缓1-2个存储周期再访问

- DMA与主存交替访存
- DMA控制器的结构6
  - 1、主存地址计数器：读或写的主存单元在什么位置
  - 设备地址寄存器
  - 2、传送长度计数器：记录还有多少字要传送or已经传送多少字
  - 3、DMA控制逻辑（控制每一步做什么动作）
  - 4、数据缓冲寄存器
  - 5、中断机构：一个数据块传完，向DMA发出中断请求
  - ~~DMA请求触发器：IO设备准备好数据,将其置1~~、
- 中断、DMA的区别?
  - 1、指令结束后响应；任一传输周期结束后响应
  - 2、传送完一个字中断CPU；传送完一批数据中断CPU
  - 3、数据传送由CPU控制完成；数据传送由DMA控制器控制完成
  - 4、可处理异常；不可处理异常
  - 5、DMA中断请求优先级高（连接高速设备，不立即输入会丢失数据）