Chapter 1

- **计算机发展历程**
    - **计算机四代变化**
    - **计算机元件更新换代**
    - **计算机软件发展**
    - **计算机分类与发展方向**

计算机系统层次结构

冯·诺伊曼机

- **计算机硬件系统组成：**
    - **运算器**
    - **存储器**
    - **控制器**
    - **输入设备**
    - **输出设备**
- **MIPS计算机硬件系统组成**
    - **存储器**
    - **寄存器堆**
    - **ALU部件**
    - **控制部件**
- **指令数据同等地位存储在存储器中，可按地址寻访**
- **指令和数据均用二进制码表示**
- **指令由操作码和地址码组成**
- **指令在存储器内按顺序存放**
- **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**

现代计算机组织结构

- **功能部件**

> CPU = 运算器 + 控制器

- **输入设备**
- **输出设备**
- **存储器**
  - **地址寄存器（MAR）：存放访存地址**
  - **数据寄存器（MDR）：暂存读出/写入存储器的数据**
- **运算器**
  - **算术逻辑单元（ALU）**
  - **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（标志寄存器，PSW）**
- **控制器**
  - **控制单元（CU）**
  - **程序计数器（PC）：存放当前欲执行指令的地址（自增1，与MAR有直接通路）**
  - **指令寄存器（IR）：存放当前指令**

计算机软件的分类

- **系统软件和应用软件**
- **三个级别的语言**
  - **机器语言**
  - **汇编语言**
  - **高级语言**

计算机工作过程

- **从源程序到可执行文件：.c => .i => .s => .o => .exe**
- **指令执行过程**

> 王道书p8

计算机系统的多级层级结构

- **第一级：微程序机器M0（微指令系统）：由硬件直接执行微指令**
- **第二级：传统机器M1（用机器语言的机器）：用微程序解释机器的指令**
- **第三级：虚拟机器M2（操作系统机器）：用机器语言解释操作系统**
- **第四级：虚拟机器M3（汇编语言机器）：用汇编程序翻译成机器语言程序**
- **第五级：虚拟机器M4（高级语言机器）：用编译程序翻译成汇编语言程序**
- **TRICKS：**

  > 王道书p6

  - **解释：边翻译边执行**
  - **编译：高级语言 => 汇编**
  - **汇编：汇编 => 机器码**
  - **机器字长：计算机能直接处理的二进制数据位数**
  - **存储字长：一个存储单元存储的二进制代码长度**
  - **指令字：完整的一条指令的二进制表示**
  - **指令字长：一个指令字包含的二进制代码的位数**

计算机性能指标

> s

- **机器字长**
- **数据通路带宽**
- **主存容量**
- **运算速度**
  - **吞吐量和响应时间**
  - **主频和CPU时钟周期**
- **CPI（Clock Cycle Per Institution）：执行一条指令所需时钟周期数**
- **CPU执行时间：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 x CPI）/主频**

- **MIPS、MFLOPS、GFLOPS、TFLOPS**
    - **MIPS（Million Instructions Per Second）：每秒执行多少百万条指令**
    - **MFLOPS（Mega Floating-point Operations Per Second）：每秒执行多少百万次浮点运算**
    - **GFLOPS（Giga Floating-point Operations Per Second）：每秒执行多少十亿次浮点运算**
    - **TFLOPS（Tera Floating-point Operations Per Second）：每秒执行多少万亿次浮点运算**
- **TRICKS：**
    - **CPU时钟周期 = 1/主频，主频以Hz为单位，1Hz表示每秒1次**
    - **K、M、G、T描述存储容量时，以2的幂次表示；描述速率、频率时，以10的幂次表示**
    - **熟知各种性能指标含义并会计算即可**

Chapter 2

> 数据的表示和运算

数制与编码

- **进制转换**

  > 特别是小数部分的进制转换

    - **二进制转八进制或十六进制：三位/四位组合**
    - **任意进制转十进制：加权求和**
    - **十进制转换任意进制：整数部分除基取余，小数部分乘基取余（王道书p27）**
- **真值和机器数**
    - **BCD码**
        - **8421码：书p28**
    - **字符编码ASCII：ASCII码最高位为0**
- **校验码**

  > **码距（最小码距）：是指任意两个合法码之间至少有几个二进制位不相同（或：从一个合法码变成另一个合法码至少要改变几位码的值，称为码距）**

> 若用4个二进制位表示16种状态，则所有可能的编码都用来表示合法码，此时码距为1，也就是说任意一位或几位的出错都会变成另一个合法码，从而没有检错能力

> **对于码距不小于2的数据校验码，开始具有检错的能力。码距越大，检错、纠错能力就越强，而且检错能力总是大于纠错能力**

> 假设用4个二进制位表示8种状态，则有8种状态是非法的，此时可以使合法编码的码距为2，如果任何一位出错都会变成非法编码，从而有了具有检查出一位错的能力

    - **奇偶校验码**
        - **用于并行数据传送中**
        - **码距为2**
    - **海明校验码**
        - **用于并行数据传送中**
        - **码距为4**
    - **循环冗余校验码（CRC）**
        - **用于串行数据传送中**

定点数的表示与运算

- **原码、补码、反码**
    - **正数：三者都相同**
    - **负数：原码与补码通过符号位不变，数值位取反加1进行互转。负数补码的反码为数值位取反**
    - **两个真值互为相反数的补码通过整体取反加1互转**
    - **补码表数范围：负数部分比整数部分多了1**
- **移码**
- **算法移位 & 逻辑移位 & 循环移位**
- **原码加减法**
- **补码加减法**
- **符号扩展**
- **判断溢出**
    - **符号位改变（无论加减乘除，只要符号位不变，尽管丢弃高位，也不是溢出）**
- **定点数 乘法 & 除法**

- **C语言中的整数类型及类型转换**
- **大端 & 小端**
    - **大端：数据高位存地址低位（大端是正常的）**
- **数据对齐**
    - **对齐原则是任何K字节的基本对象的地址必须是K的倍数**

IEEE 754 浮点数的表示与运算

> 王道书p64

- **IEEE 754 单精度浮点数（正数）**
    - **最小值：2^(-126)**
    - **最大值：2^(127)x(2 - 2^(-23))**
- **IEEE 754 双度浮点数（正数）**
    - **最小值：2^(-1022)**
    - **最大值：2^(1023)x(2 - 2^(-52))**
- **浮点数加减**
- **TRICKS**
    - **十进制小数转IEEE 754单精度浮点数**

        > CSDN

        > CSDN

        > **1位符号位 + 8位阶码 + 23位尾数**

        - **先将整数与小数部分分别转换为二进制数**
        - **确定2^E，使之不超过整数部分的大小，其中E = e - bias(127)，可以算出e，将e写成原码即为8位阶码部分**
        - **将二进制数的小数点往左移E位（即小数点左边只有一位数），小数点右边的二进制数位即为23位小数部分**
    - **IEEE 754单精度浮点数加减**

算术逻辑单元（ALU）

> 王道书p82，讲数字逻辑的，考的几率很小

> 这里清本课件还有一些数字逻辑的内容，来的及再看，考的机率很小

Chapter 3

> 存储系统

局部性原理

- **空间局部性**
- **时间局部性**

层次存储器结构层次之间应满足的原则

- **一致性原则**

> 处在不同层次存储器中的同一个信息应保持相同的值

- **包含性原则**

> 处在内层的信息一定被包含在其外层的存储器中，反之则不成立，即内层存储器中的全部信息，是其相邻外层存储器中的一部分信息的复制品

地址总线 & 数据总线 & 控制总线

- **数据线数和地址线数共同反映存储体容量的大小**

主存储器的读写过程

- **读过程**
    - **给出地址**
    - **给出片选与读命令**
    - **保存读出内容**
- **写过程**
    - **给出地址**
    - **给出片选与数据**
    - **给出写命令**

半导体随机存储器

- **静态随机存储器（SRAM）**

    > **工作特点：非破坏性读出、无需定期刷新、同时送行列地址**

    - **使用双稳态触发器（六晶体管MOS）来记忆信息（非破坏性读出）**

    - 无需刷新
- 动态随机存储器（DRAM）

  > 工作特点：破坏性读出、需定期刷新、分两次送行列地址

    - 使用栅极电容的电荷存储信息
    - 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
    - DRAM电荷1~2ms就会消失，所以必须刷新
        - 集中刷新
        - 分散刷新
        - 异步刷新
    - 动态存储器的读/写过程
- 只读存储器（ROM）
    - MROM
    - PROM
    - EPROM
- 主存容量的扩展
    - 位扩展法
    - 字扩展法
    - 字位同时扩展法
- 存储芯片的地址分配和片选
- 单体多字存储器 & 多体并行存储器

Cache

> Cache采用静态存储器（SRAM）实现（硬件实现）

- Cache参数
    - 块
    - 命中
        - 命中率
        - 命中时间：访问在较高层次中数据的时间

    - 失效
        - 失效率
        - 失效损失：替换较高层次数据块的时间+将该块交付给处理器的时间
    - 平均访问时间 = 命中率 * 命中时间 + （1 - 命中率）*失效损失
- Cache和主存的映射方式（W-I）
    - 直接映射
        - 物理地址结构：标记 + Cache行号 + 块内偏移
        - Cache行结构：有效位（1位） + 标记位 + 数据位
    - 全相联映射
        - 物理地址结构：标记（主存块号） + 块内偏移
        - Cache行结构：有效位（1位） + 标记位 + 数据位
    - 组相联映射
        - 物理地址结构：标记 + Cache组号 + 块内偏移
        - Cache行结构：有效位（1位） + 标记位 + 数据位
- Cache写策略
    - Cache写命中（Write hit）
        - 写直达（全写法，Write through）
        - 拖后写（写回法，Write back）
    - Cache写不命中（Write miss）
        - 写分配法（Write allocate）
        - 非写分配法（not Write allocate）
- Cache缺失的四类原因
    - 必然缺失
        - 对策：预取
    - 容量缺失
        - 对策：增加Cache容量
    - 冲突缺失

        - **对策：增加Cache容量、增加相联的组数**
    - **无效缺失**
- **影响Cache缺失率的因素**
    - **经验总结：容量为N，采用直接映射方式Cache的缺失率和容量为N/2、采用2路组相联映射方式Cache的缺失率相当**
- **Cache主存块替换算法**
    - **LRU**
    - **FIFO**
    - **RAND**
- **多级Cache**
    - **采用多级Cache提高命中率**
    - **将Cache分解为指令Cache和数据Cache**
- **Cache接入系统的体系结构**
    - **侧接法**
    - **隔断法**
- **Cache一致性保证策略（MESI）**
    - **修改态（M）**
    - **独占态（E）**
    - **共享态（S）**
    - **无效态（I）**

虚拟存储器

由操作系统管理

- **减少页表本身所占空间**
    - **层次页表**
    - **反转页表**
- **对磁盘的访问总是由缺页引起的**

Chapter 4

> **指令（机器指令）是计算机运行的最小功能单位**

指令系统的设计要求

- **完备性**
- **规整性**
- **高效性**
- **兼容性**

指令格式

- **指令由操作码和地址码（操作数地址）构成：**

  > **指令操作码的位数限制指令系统中的指令条数**

  - **操作码：指出指令的功能，如算术加减、程序转移、返回操作等**
  - **地址码：给出被操作的信息的地址，如数的地址、转移地址、入口地址等**

- **指令长度（指令字长）：指令包含二进制码的位数**
  - **指令字长和机器字长没有固定关系**
  - **指令字长等于机器字长的指令称为单字长指令...**

- **指令格式**
  - **零地址指令：只有操作码，没有显式地址码**
    - **零地址的运算类指令仅用在堆栈计算机中（栈顶和次栈顶保存有操作数）**
  - **一地址指令：一个操作码 + 一个地址码（操作数）**
    - **另一个操作数可隐含地由ACC给出**
  - **二地址指令：一个操作码 + 两个地址码（目的操作数 + 源操作数）**
    - **目的操作数地址默认保存本次运算的结果**
  - **三地址指令：一个操作码 + 三个地址码（目的操作数 + 源操作数 + 运算结果保存地址）**
  - **四地址指令：一个操作码 + 四个地址码（目的操作数 + 源操作数 + 运算结果保存地址 + 下一条指令地址）**

- **定长操作码指令格式**
  - **指令字最高位分配固定的若干位操作码**

- **扩展操作码指令格式**

- **扩展操作码有一些限制，见书p161**

指令寻址方式

- **指令寻址：寻址下一条将要执行的指令地址**
  - **顺序寻址：程序计数器（PC）自增1形成下一条指令地址**
  - **跳跃寻址：通过转移类指令实现，由本条指令给出下条指令地址的计算方式。**
    - **绝对地址（由标记符直接得到）**
    - **和相对地址（相对于当前指令的偏移量）**
    - **跳跃的结果还是修改PC值，下一条指令的地址最终仍然由PC给出**
- **数据寻址：寻找操作数的地址**
  - **隐含寻址：隐含地规定ACC中为第二个操作数地址**
    - **有利于缩短指令字长**
  - **立即（数）寻址：指令的地址字段直接给出操作数**
    - **在指令的执行阶段无需访存，指令执行时间最短，但地址字段的位数会限制立即数的范围**
  - **直接寻址：指令的地址字段给出操作数的真实地址**
    - **指令在执行阶段仅访问一次主存**
  - **间接寻址：指令的地址字段给出操作数的真实地址的地址**
    - **可以是一次间接寻址，也可以是多次间接寻址**
    - **可扩大寻址范围**
  - **寄存器寻址：指令的地址字段给出操作数所在的寄存器编号**
    - **指令在执行阶段无需访存，只访问寄存器**
  - **寄存器间接寻址：指令的地址字段给出操作数的真实地址所在的寄存器编号**
  - **相对寻址：指令的地址字段给出形式地址，真实地址由 (PC) + 形式地址 给出**
    - **注意：当读取完本条指令时，PC中的地址已经是下一条指令的地址了，所以真实地址是相对于下一条地址的偏移**
  - **基址寻址：指令的地址字段给出形式地址，真实地址由基址寄存器（BR）的内容加上指令格式中的形式地址给出**
    - **基址不能改动，形式地址可改动**

    - **变址寻址：指令的地址字段给出形式地址，真实地址由变址寄存器（IX）的内容和指令格式中的形式地址给出**
        - **形式地址不能改动，变址寄存器可以改动**
        - **便于处理数组问题**
    - **堆栈寻址**
        - **硬堆栈**
        - **软堆栈**
- **X86汇编指令：书P171**

CISC & RISC

- **复杂指令系统计算机（CISC）**
    - **指令字长：不固定**
    - **可访存指令：不加限制**
    - **各种指令执行时间：相差较大**
    - **各种指令使用频度：相差很大**
    - **控制方式：微程序控制**
    - **指令流水线：可以实现**
- **精简指令系统计算机（RISC）**
    - **指令字长：定长**
    - **可访存指令：只有Load/Store指令**
    - **各种指令执行时间：绝大多数在一个周期内完成**
    - **各种指令使用频度：都比较常用**
    - **控制方式：组合逻辑控制**
    - **指令流水线：必须实现**
- **TRICKS**
    - **程序计数器（PC）一般每读取主存一个字节，自增1。所以从主存中读完当前指令的时候，PC中保存的已经不是本指令的地址了**
    - **内存的地址编号是无符号整数，注意原码和补码的互转**

Chapter 5

> 中央处理器

CPU功能和基本结构

- **CPU由运算器和控制器组成**
    - **控制器：负责协调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**
        - **程序计数器（PC）、指令寄存器（IR）、指令译码器、存储器地址寄存器（MAR）、存储器数据寄存器（MDR）、时序系统和微操作信号发生器**
    - **运算器：对数据加工**
        - **算术逻辑单元（ALU）、暂存寄存器、累加寄存器（ACC）、通用寄存器组、程序状态字寄存器（PSW）、移位器、计数器（CT）**

指令执行过程

> **CPU从主存中取出并执行一条指令的时间称为指令周期**

- **指令周期常用若干机器周期表示，一个机器周期又包含若干时钟周期（也称T周期或节拍），每个指令周期内的机器周期数可以不等，每个机器周期内的时钟周期数也可以不等**
- **指令周期：**
    - **取指周期**
    - **间址周期**
    - **执行周期**
    - **中断周期**
    - **对于无条件转移指令，执行时不需要访问主存，其指令周期只包含取指周期和执行周期**
    - **对于间址寻址的指令，还需包括间址周期**
    - **当CPU采用中断方式实现主机和I/O设备的信息交换时，CPU在每条指令执行结束前，都要发出中断查询信号，若有中断请求，则CPU进入中断响应阶段**
- **指令周期的数据流：书p204**
- **指令执行方案**
    - **单指令周期**
        - **串行执行**

        - **每条指令都在固定的时钟周期内完成**
        - **下一条指令只能在前一条指令执行结束后才能启动**
    - **多指令周期**
        - **串行执行**
        - **每条指令可在不同时钟周期内完成**
        - **下一条指令只能在前一条指令执行结束后才能启动**
    - **流水线方案**

数据通路的功能和基本结构

> **书p211**

控制器的功能和工作原理

- **硬布线控制器：书p224**
- **微程序控制器：书p228**

> **微程序设计思想就是将每条机器指令编写成一个微程序，每个微程序包含若干微指令，每条微指令对应一个或几个微操作命令。这些微程序存储在CPU中的控制存储器中**

> **微程序核心观点是：将一条指令分为若干微指令，这些微指令存于CM中，只是在CPU内部的指令执行流程发生了改变，宏观上，对于每条指令，还是基本不变**

    - **微命令是微操作的控制信号，微操作是微命令的执行过程**
    - **控制存储器（CM）用于存放微程序，在CPU内部，用ROM实现**
    - **机器指令的操作码字段通过微地址形成部件产生该机器指令对应的微程序的入口地址**
    - **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（1为公共的取值微程序）**

指令流水线

> **必考重点，书p241**

- **提高计算机内部的并行性**
    - **空间并行性**
        - **在一个处理机内设置多个独立的操作部件，并且使这些部件并行工作**
    - **时间并行性**
        - **就是采用流水线技术**

- **流水线的实现原理**

  > **前提：大多数机器都是将指令和数据保存在同一存储器中，且只有一个访问口，一个时钟周期只能由一条指令进行访问**

  - **MIPS指令流水实现**
    - **取指令（IF）：涉及访存（指令存储器）操作**
    - **指令译码（ID）：读取寄存器Reg数据**
    - **指令执行（EXE）：ALU运算。若无利用寄存器的访存操作，EXE末尾实际上ALU已经形成计算结果数据，**
    - **读存储器（MEM）：涉及访存（数据寄存器）操作。若有利用寄存器寻址的操作，则在MEM末尾实际上已经形成计算结果数据**
    - **写回（WB）：写回寄存器Reg**
  - **MIPS指令集**
    - **访存指令**
      - **LW rt rs imm**
        - **Addr <- R[rs] + SignExt(imm)**
        - **R[rt] <- MEM[Addr]**
      - **SW rt rs imm**

        > **SW指令未改变寄存器的值**

        - **Addr <- R[rs] + SignExt(imm)**
        - **MEM[Addr] <- R[rt]**
    - **算术逻辑运算指令**
      - **ADDU rd rs rt**
        - **R[rd] <- R[rs] + R[rt]**
      - **SUBU rd rs rt**
        - **R[rd] <- R[rs] - R[rt]**
      - **ORI rt rs imm**
        - **R[rt] <- R[rs] or ZeroExt(imm)**
    - **转移指令**

- BEQ rs rt imm
  - cond <- R[rs] - R[rt]
  - if (cond eq 0) then
  - pc <- pc + 4 + SignExt(imm)*4 else
  - pc <- pc + 4
- J target
  - pc[31: 0] <- pc[31: 28] || target[25: 0] || 00

- 指令执行步骤——单周期CPU
  - 每条指令占用一个时钟周期
  - 时钟周期的长度需满足最慢的指令（LW）
- 指令执行步骤——多周期CPU

  > 注意时钟周期的时间取决最长的步骤

  - 每个步骤占用一个时钟周期
  - MIPS 7个指令集多周期分析

    > IF ID EXE MEM WB

    - ADDU、SUBU、ORI

      > 仅需4个步骤（因为ADDU类指令无需在MEM阶段访存）

      - IF——ID——EXE——WB
    - LW

      > 5个步骤

      - IF——ID——EXE——MEM——WB
    - SW

      > 4个步骤（SW没有数据需要在WB阶段写回寄存器堆）

      - IF——ID——EXE——MEM
    - BEQ

      > 3个步骤

      - IF——ID——EXE

- J
  > 2个步骤
  - IF——ID

- 关于MIPS五级流水线各类型指令CPI的讨论
  - LW
    - 若无数据冒险产生，则n条指令总共需要n + 4个时钟周期（找规律即可），平均每条指令需要（n + 4） / n = 1个时钟周期
    - 当发送数据冒险且采用转发技术处理，则n条指令总共需要2n + 3个时钟周期（同样找规律），平均每条指令需要（2n + 3） / n = 2个时钟周期
  - SW、ADDU类
    - 由于SW不改变寄存器值，ADDU不访存且采用转发技术，故n条指令总共需要n + 4个时钟周期，平均每条指令需要（n + 4） / n = 1个时钟周期
  - BEQ
    - ---
  - J
- 分支预测
  - 静态预测
    - 每次都预测“Token”或“Not Token”，不随预测准确率动态改变
  - 动态预测
    - 1位预测位
      - 只要预测错误就转换预测方向
    - 2位预测位
      - 连续两次预测错误才转变预测方向
- TRICK：
  - 结构相关
    - 若机器存储器只有一个访问口，则IF与MEM阶段会产生结构相关（若未特殊说明，默认存储器有多个访问口）
  - 数据相关

        - LW R1，(0)R2 => SUB R4，R1，R5
            - 第一条指令的计算结果最快在MEM阶段的末尾得出，此时就算利用旁路技术也必须暂停流水线
        - LW R1，R2 => SUB R4，R1，R5
            - 第一条指令的计算结果最快在EXE阶段末尾得出，此时若利用旁路则无需暂停流水线

- 指令执行方式

    > 指令多次重叠执行方式实际上就是指令流水线

    > 假设每条指令有k个阶段，每段的时间都为t

    - 顺序执行方式
        - n条指令执行所用时间：T = knt
    - 一次重叠执行方式
        - n条指令执行所用时间：T = knt - (n - 1)t
    - 两次重叠执行方式
        - n条指令执行所用时间：T = knt - (n - 2)t
    - m次重叠执行方式
        - n条指令执行所用时间：T = knt - (n - m)t
    - 典型情况

        > 假设每个指令有k=3个阶段，每段时间为t

        - 顺序执行方式
            - n条指令执行所用时间：T = 3nt
        - 一次重叠执行方式
            - n条指令执行所用时间：T = (2n + 1)t
        - 两次重叠执行方式
            - n条指令执行所用时间：T = (n + 2)t

- 流水线的特点
    - 流水线的每一个功能部件后面都要有一个缓冲寄存器（锁存器），其作用是保存流水段的中间结果、中间参数、控制信号

- **各功能段的时间尽量相等。流水线的时钟周期不能短于最慢的流水段**
- **流水线中处理的必须是连续任务**
- **装入时间：第一个任务进入流水线到输出流水线的时间**
- **排空时间：第n个（最后一个）任务进入流水线到输出流水线的时间**

- **流水线的实现**
  - **PC值多路选择器被放到IF段（取指令），这样做的目的是保证对PC值的写操作只出现在一个流水段内，防止发送写冲突**

- **影响流水线的因素**
  - **流水线中的冲突**

    > **清本课件，以此为准**

    - **结构（资源）冲突**
      - **解决方法**
        - **暂停流水线执行，插入等待周期**
        - **增加资源，解决资源冲突**
    - **数据冲突：局部冲突**

      > **对于两条指令i和j，假设指令i在j之前进入流水线，下面讨论几种不同的数据冲突**

      - **写后读冲突（RAW）**

        > 指令j的执行需要使用指令i的计算结果，但是当它们在流水线中重叠执行时，指令j可能在指令i将其计算结果写入之前就先行对保存该计算结果的寄存器进行了读操作，这样指令j读出的寄存器值就是错误的

        - **最常见**
        - **解决方法**
          - **旁路（定向）技术**
      - **写后写冲突（WAW）**

        > 指令j和指令i的目的操作数相同，但是当它们在流水线中重叠执行时，指令j可能在指令i将其计算结果写入之前就先行对保存该计算结果的寄存器进行了写操作，这样就导致了寄存器写入顺序的错误，此时，目的寄存器的内容是指令i写入的值，而不是指令j写入的值

        - **MIPS指令流水不会发送WAW冲突**
    - **读后写冲突（WAR）**

      > 指令j可能在指令i读取某个源寄存器的内容之前就对该寄存器进行了写操作，结果就是导致了指令i后来读取的值是错误的

        - **MIPS指令流水不会发生WAR冲突**
    - **解决方法**
        - **旁路（定向）技术**
        - **暂停流水线**
        - **编译器调度**
            - **动态调度**
            - **静态调度**
  - **控制冲突：全局冲突**

    > **转移类指令改写PC造成的冲突**

    - **解决方法**
        - **暂停流水线**
        - **预测分支**
        - **分支目标缓冲技术**
        - **延迟槽**
- **王道上的冲突**
    - **资源冲突**
    - **相关问题**
        - **数据相关：下一条指令会用到这一条指令计算出的结果，则这两条指令数据冒险**
            - **解决方法：数据旁路、暂停流水、编译器处理**
        - **控制相关**

- **流水线的性能指标**
    - **吞吐率：单位时间执行指令的数量**
    - **加速比：与串行执行时速度提高的比率**

    - 效率
- 超标量流水线
    - 超标量流水线
    - 超流水线

Chapter 6

总线，书p264

- 特点：分时与共享
- 总线特性：机械特性、电器特性、功能特性、时间特性
- 总线的猝发传输方式
    - 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，称为猝发传送
- 总线的最高速度主要由下列因素决定
    - 总线长度
    - 总线负载的设备数
    - 负载设备的特性
- 系统总线结构
    - 单总线结构
    - 双总线结构
    - 三总线结构
- 总线类型
    - 处理器-主存总线（专用）
    - 输入/输出总线（行业标准）
        - 和处理器-主存总线通过桥连接（或通过主板总线）
    - 主板总线（行业标准或专门设计）
- 总线组成
    - 片内总线：CPU内部
    - 系统总线

        - 数据总线：双向传输
        - 地址总线：单向传输（CPU => 地址总线）
        - 控制总线
- 总线相关概念
    - 总线主设备：有能力控制总线，发起总线事务
        - 总线事务：发起命令（和地址）、传输数据
    - 总线从设备：响应主设备请求
    - 总线通信协议
    - 异步总线传输：控制信号（请求，应答）作为总控信号
    - 同步总线传输：使用共同的时钟信号
- 总线仲裁
    - 集中仲裁
        - 菊链仲裁
        - 集中平行仲裁
    - 分布仲裁
        - 通过自我选择进行分布式仲裁
        - 碰撞检测
    - 按优先级仲裁或轮循仲裁
- 增加总线带宽
    - 增加总线的宽度
    - 分别设置数据总线和地址总线
    - 采用成组传送方式
- 多主设备总线提高事务数量
    - 仲裁重叠
    - 总线占用
    - 地址、数据传送重叠
- 总线性能指标

    - **总线带宽**
    - **总线频率**
    - **总线宽度**
- **总线操作和定时**
    - **同步定时**
    - **异步定时**
- **总线标准**
    - **PCI：并行**
        - **集中仲裁方式-集中平行仲裁（和上一事务重叠）**
        - **32位地址和数据线互用**
    - **PCI-E：串行**
    - **USB：串行**
        - **由4根线组成（电源、接地和双数据线）**
        - **USB帧**
            - **控制帧**
            - **同步帧**
            - **块传送帧**
            - **中断帧**
    - **SATA：串行**
- **TRICKS**
    - **数据通路：各个功能部件通过数据总线连接形成的数据传输路径称为数据通路**

Chapter 7

输入/输出系统

- **外部设备**
    - **输入设备**
    - **输出设备**

        - 显示器：了解一下VRAM计算相关（颜色深度、灰度等概念）
    - 外存储器
        - 磁盘存储器
            - 磁盘设备组成
                - 存储区域：磁头数、柱面数、扇区数
                - 硬盘存储器组成：磁盘驱动器、磁盘控制器、盘片
            - 磁记录原理
                - PMR
                - CMR
                - SMR
            - 磁盘性能指标
                - 磁盘的容量
                    - 格式化后容量比格式化前小
                - 记录密度
                - 平均存取时间
                - 数据传输率
            - 磁盘地址
                - 驱动器号 + 柱面（磁道）号 + 盘面号 + 扇区号
        - 磁盘阵列（RAID）

          CSDN

            - RAID 0（条带化）
                - ---
                - 数据分布式存储于各磁盘中
                - 容量翻倍，读写速度翻倍，无冗余
            - RAID 1（镜像）
                - ---
                - 50%数据通过镜像备份于完整磁盘中
                - 磁盘容量利用率50%（容量减半），读取速度为单磁盘两倍，写入速度与单磁盘相近

- **RAID 2（并行访问）**

  > **根据n + k <= 2^k - 1判断每个条带数据位和校验位个数**

  - **通过汉明码实现冗余**
  - **每个数据的每一位分别存储在不同磁盘中，所以是“并行访问”**
  - **磁盘数量取决于数据宽度（即汉明码的数据位数n）：例如数据宽度为4，则需要4块磁盘存储数据，根据n + k <= 2^k - 1，所以需要3块磁盘存储校验位数据，总共需要7块磁盘；再如，数据宽度为64，则需要64块磁盘存储数据，根据n + k <= 2^k - 1，所以需要7块磁盘存储校验位数据**

- **RAID 3（并行访问）**

  > **每个条带一个校验位，至少需要3块磁盘**

  - **交错位奇偶校验**
  - **只需一块磁盘作为校验磁盘，其余存数据**
  - **当一块数据盘损坏，可根据其他盘的校验位和数据位恢复数据**

- **RAID 4（独立访问）**

  > **每个条带一个校验位，至少3块磁盘**

  - **交错块奇偶校验**
  - **其余和RAID3一样，区别在于条带化方式的不同**
  - **当一块数据盘损坏，可根据其他盘的校验位和数据位恢复数据**

- **RAID 5（独立访问）**

  > **每个条带一个校验位，至少需3块磁盘**

  - **交错块分布奇偶校验**
  - **不采用单独的校验磁盘，校验位分布在不同磁盘上**
  - **当一块数据盘损坏，可根据其他盘的校验位和数据位恢复数据**
  - **目前综合性能最佳**

- **RAID 6（独立访问）**

  > **每个条带两个校验位，至少需要4块磁盘**

  - **交错块双重分布奇偶校验**

        - 每个条带有两位校验位（RAID3、4、5均只有一位），所以当两块磁盘损坏时，可以恢复数据
    - RAID 7
        - ---
        - 实时事件操作系统
        - 采用Cache技术提高访问速度，并在Cache中完成校验
    - RAID 10
        - ---
        - 两块磁盘先组成RAID1，然后再将这两块磁盘整体和另两块磁盘组RAID0
    - RAID 01：
        - ---
        - 两块磁盘先组成RAID0，然后再将这两块磁盘整体和另两块磁盘组RAID1

- I/O接口
    - 编址
        - 统一编址
        - 独立编址

I/O方式

- 程序直接控制
    - CPU与I/O串行工作
- 程序中断方式
    - 内中断 & 外中断
        - 内中断：处理器和内存内部产生的中断
        - 外中断：外部设备产生的中断
    - 硬中断 & 软中断
        - 硬中断：
        - 软中断：
    - 非屏蔽中断 & 可屏蔽中断
        - 非屏蔽中断：内中断不可屏蔽

        - 可屏蔽中断：
    - 中断判优
    - 中断隐指令（由硬件自动完成）
        - 关中断
        - 保存断点
        - 引出中断服务程序
    - 中断向量：中断服务程序的入口地址，存放在中断向量表中
    - 中断处理过程：和操作系统的差不多
    - 多重中断和中断屏蔽技术
    - 中断设备接口组成
        - 中断请求寄存器
        - 中断屏蔽寄存器
        - 优先级排队线路
- DMA方式

    > 与设备一对一

    - DMA完整过程：DMA控制器接收外设发出DMA请求，并向CPU发出总线请求，CPU让出总线，DMA完全接管总线控制权，进入传输数据阶段，该阶段完全由DMA控制，传输结束后向发出中断，告知CPU数据传输完毕
    - DMA工作方式
        - 独占总线
        - 周期窃取
    - DMA传送过程
        - 预处理
        - 数据传送（成组传送）
        - 后处理
- I/O通道

    > 与设备一对多，可并行工作

    - 通道类型

        - **字节多路通道**
            - **共享、分时**
        - **选择通道**
            - **独占**
        - **数组多路通道**
            - **前两种结合**
- **外围处理机**
    - **通道型处理机**
        - **共享内存**
    - **外围处理机**
        - **通过通道方式与主机交互**
- **TRICKS**
    - **CPU响应中断的时间是在每条指令执行阶段的结束时刻**
    - **CPU响应DMA请求的时间是在每个机器周期结束时**