ch1 概论

计算机不是一个孤立的系统,要与外设和通信线路进行交互
计算机内部主要组成成分有:中央处理单元,主存储器,总线等系统互连,输入输出设备模块
CPU 内部组成为:高速缓存,算术逻辑单元,CPU 内部互连,控制单元
冯诺依曼计算机的主要特点有:

- 存储程序方式,采用程序控制
- 按地址访问，指令顺序执行
- 单元定长的一维线性空间存储器
- 使用低级机器语言，数据以二进制表示
- 单处理机结构，以运算器为中心
- 又称为普林斯顿体系结构,从原来的以运算器为中心演变为以存储器为中心,通过各种并行处理手段提高计算机系统性能
- 而哈佛结构:指令和数据有独立的存储空间

机器字长:CPU 一次能处理的数据位数
存储容量:存储单元个数×存储字长
硬件五大指标:存储器,控制器,运算器,输入设备,输出设备
性能指标:吞吐量,响应时间,利用率,处理机器字长,总线宽度,存储器容量,存储器带宽,主频
CPU 执行时间:

$$\text{执行一段程序耗时} = \text{CPU 周期数} \times \text{周期长} = \frac{\text{指令条数}}{\text{MIPS} \times 10^6}$$

CPI:每条指令的周期数
MIPS:

$$\text{每秒百万条指令数} = \frac{\text{时钟频率}}{\text{CPI} \times 10^6}$$

MFLOPS:每秒百万条浮点运算数
计算机软件系统:

- 系统软件
  - 管理、调度、监视和维护计算机系统软硬件资源的程序集合，使系统资源得到合理调度，确保高效
  - 包括操作系统,语言处理程序,数据库管理系统,服务性程序,网络管理软件
- 应用软件
  - 又称为应用程序
  - 用户在各自不同的应用领域根据具体的任务需要所开发编制的各种程序

计算机体系结构:

- 单指令单数据流:
  - 任何一个时钟周期内只执行一条指令并只处理一个数据
  - 单处理器机
- 单指令流多数据流:
  - 同一时间执行同一条指令但指令作用于多个不同的数据元素
  - 向量处理机,阵列处理机
- 多指令流单数据流:
  - 多个处理器,每个处理器独立地执行不同的指令流,但它们都作用于同一个数据流
  - 从未实现

- 多指令流多数据流
  - 多个处理器,每个处理器都独立地执行不同的指令流,并且处理不同的数据流
  - 多处理器机(紧耦合系统),机群系统(分布式系统,松耦合系统)

ch2 运算方法和运算器

进制转化:太简单了,不写,反正是重点之一
无符号数:每一位均表示一个数位
有符号数:

$$\text{定点数}\begin{cases}\text{整数} & \text{第一位符号,小数点隐藏于数值后}\\ \text{小数} & \text{第一位符号小数点隐藏于符号后}\end{cases}$$

$$\text{浮点数}\begin{cases}\text{单精度} & \text{1 位符号 8 位阶码(定点整数),23 位尾数}\\ \text{双精度} & \text{1 位符号,11 位阶码,52 位尾数(定点小数)}\end{cases}$$

原码:真值 0 有两种表示,符号位不可参与运算
补码:比原码多表示一个数且 0 只有一种表示,用 n 位二进制数表示整数补码(n 包括一位符号)

$$[X]_{\text{补}}=\begin{cases}X & 0\le X\le 2^{n-1}-1\\ 2^n+X & -2^{n-1}\le X<0\end{cases}$$

用 n 位二进制数表示小数补码

$$[X]_{\text{补}}=\begin{cases}X & 0\le X\le 1\\ 2+X & -1\le X<0\end{cases}$$

反码:符号位可参与运算,要循环进位,0 有两种表示
任意一个浮点数可被表示为

$$N=\text{基数}^{\text{阶码}}\times\text{尾数}$$

-∞ (上溢区间) 最小负数 (负区间) 最大负数 (下溢区间) 最小正数 (正区间) 最大正数 (上溢区间) +∞

算术移位:对象有符号数,符号位移动时不变
- 对于原码移位添补 0,高位丢 1 结果出错,低位丢 1 精度缺失
- 也就是反码只有在对负数右移的时候才引入 1,其他时候都是 0
- 补码左移补 0,右移补符号位,高位丢 0 出错低位丢 1 精度缺失

左 规

- 尾数运算结果为$11.1bb\ldots b$或$00.0bb\ldots b$:
  - 尾数每左移1位，阶码减1，直到尾数成为规格化数为止。
  - 还需要判断阶码是否下溢。若发生下溢（符号位为10），可认为结果为0。

右 规

- 尾数运算结果为$10.bb\ldots b$或$01.bb\ldots b$，即尾数运算结果溢出:
  - 尾数只需要右移1位，阶码加1。
  - 还需要判断阶码是否上溢。若发生上溢（符号位为01），可认为结果为∞。

加法器进位:
令

$$Y_n = A_n \cdot B_n, X_n = (A_n \oplus B_n)$$

进位位

$$C_{i+1} = Y_i + X_i \cdot C_i$$

$$C_1 = Y_0 + X_0 \cdot C_0$$

$$C_2 = Y_1 + X_1 \cdot C_1 \text{ (串行)} = Y_1 + X_1 \cdot (Y_0 + X_0 \cdot C_0) \text{ (并行)}$$

推广至加法器进位链(74181-ALU):
加法器中每一位的进位C_{i+1}可以表示为

$$C_{i+1} = A_iB_i + (A_i \oplus B_i)C_i$$

设进位生成

$$Y_n = A_nB_n$$

设进位传播

$$X_n = A_n \oplus B_n$$

则进位表达式变为

$$C_{n+1} = Y_n + X_nC_n$$

以此类推,对于 4 位加法器(如 74181):

$$C_1 = Y_0 + X_0C_0$$

$$C_2 = Y_1 + X_1C_1 = Y_1 + X_1Y_0 + X_1X_0C_0$$

$$C_3 = Y_2 + X_2C_2 = Y_2 + X_2Y_1 + X_2X_1Y_0 + X_2X_1X_0C_0$$

$$C_4 = Y_3 + X_3C_3 = Y_3 + X_3Y_2 + X_3X_2Y_1 + X_3X_2X_1Y_0 + X_3X_2X_1X_0C_0$$

每串等式前半部分称为串行,后半部分称为并行,正如上文普通加法器所言
为了加速进位传播,引入了进位生成G和进位传播P的概念,它们是 4 位 74181 的进位输出:令

$$G = Y_3 + X_3Y_2 + X_3X_2Y_1 + X_3X_2X_1Y_0$$

$$P = X_3X_2X_1X_0$$

则最终的 4 位进位输出 C_{n+4} 可以表示为:

$$C_{n+4} = G + PC_n$$

先行进位部件(74182 CLA 负责并行计算多组进位，其进位逻辑式如下C_0 表示初始进位 C_n
C_4 的进位逻辑(第一组 4 位的进位输出):

$$C_4 = G_0 + P_0C_0$$

这里 G_0 和 P_0 是第一片 74181 的 G 和 P 输出
C_8的进位逻辑(第二组 4 位的进位输出):

$$C_8 = G_1 + P_1C_4$$

将 C_4 代入得:

$$C_8 = G_1 + P_1G_0 + P_1P_0C_0$$

这里 G_1 和 P_1 是第二片 74181 的 G 和 P 输出
我们可以得到C_8, C_{16}将 C_{16} 进一步简化,引入更高层次的组进位生成和组进位传播

$$G' = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$

$$P' = P_3P_2P_1P_0$$

这样，C_{16}可以表示为:

$$C_{16} = G' + P'C_0$$

对于 74182,其进位输出的一般表达式为

$$C_{n+x} = G_0 + P_0C_n$$

$$C_{n+y} = G_1 + P_1G_0 + P_1P_0C_n$$

$$C_{n+z} = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_n$$

$$G' = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0$$

$$P' = P_3P_2P_1P_0$$

如果是对于 ppt 来说的话,每四位的 74181 就将自己的 P 和 G 给到 74182,74182 将这个超前进位信息给到下一个 74181,一个 74182 是 16 位的
至于 74181 的具体引脚,我只能说 A 是输入 1,B 是输入 2,S 是功能选择,M 是模式控制(A or L),输出是 F,进位输入输出是 C,P 是传播进位输出,G 是生成进位输出,其他的会给手册参数表
汉字编码
字符串:

- 每个 ASCII 码字符占 1 个字节
- 每个 Unicode 编码占用 2 个字节

汉字输入码:

- 也称外码,是代表某一汉字的一串键盘符号
- 分为数字编码,拼音编码,字形编码,音形编码
- 通过汉字输入法程序后转化为国标码或区位码

汉字交换码:
- 不同计算机系统之间在交换汉字信息时所使用的代码标准
- 区位码(把汉字表示成二维数组,位于第几片区第几个,结果为 D),国标码(区位码加 2020H,结果为 H)

汉字内码:
- 由国标码+8080H 得来
- 用于汉字信息的存储、交换、检索等操作的机内代码,一般采用两个字节表示
- 内码在计算机中是唯一的
- 为了与英文字符能相互区别,汉字机内代码中两个字节的最高位均规定为 1

汉字字形码:
- 内码通过字形检索程序得来
- 将汉字字形经过点阵数字化后形成的一串二进制数,用于汉字的显示和打印
- 分为简易型,普通型,提高型

校验码:
- 基本思想:冗余校验,即通过在有效信息代码的基础上,添加一些冗余位来构成整个校验码
- 构成:有效信息＋校验位
- 常用:奇偶校验,重点,计算机概论讲过,略

ch3 存储器

存储性能:MAR(access)的位数看存储单元数量 MDR(data)位数看存储字长
容量 = 数量×字长
译码结构:①单译码:2^n ②双译码:$2^{\frac{n}{2}}, 2^{\frac{n}{2}}$
机器字长:CPU 一次能处理的位数 存储字长:MDR 位数

SRAM:
- 双稳态触发器,半导体互锁机制
- 只要通电,内部不需周期刷新便不变,读不破坏原有 data,速度快,但用晶体管多,用作寄存器,cache
- 存储元拓展:行列各 n 条,存储元 n^2 个
- 存储阵列拓展:$n^2 \times x$片存储阵列摞在一起,行列各 n 条同时控制集成后使用 WE,CS 引脚控制读写

DRAM:用作主存,需读复写(读后重新写入),刷新(电容逐渐放电),一般按行刷新
读写:预充电→访问→信号检测→数据恢复→数据输入/出
集中刷新:CPU 不访问 DRAM,称为死区时间,连续
分散刷新:存在周期后加上刷新 1 行时间作为新周期
异步刷新:平均分,分散时多刷新几行
发展→地址复用,行列均由一组线控制,加上行、列选通线、IO、读写 ctrl

ROM 分类:
- MROM:只读不改,掩膜位写
- PROM:可读多次,编程改写一次,电写
- EPROM:可读多次,可多次紫外线擦除后脱机写入
- E^2PROM:进化 EPROM,电擦除改写,在线改写,精准对某一单字
- 闪存:EEPROM,只是不能单字修改

拓展技术:
位拓展:存储芯片位宽小于数据线(CPU)位宽时,称为数据位扩展或位长扩展
字拓展:芯片容量小于 CPU 寻址范围，多次片寻址应用于片选
双端口存储器:同一个存储体左右两个端口各自有相互独立的读写控制电路,通过左右两个独立的控制电路和寄存器，允许同时进行独立的读写访问
高位多体交叉(顺序编址):与字拓展相似,分为模块+块内地址,程序的指令、数据连续分布在同个单元中,无法与主存模块并行,带宽仅为$\frac{1}{T}$
低位多体交叉(交叉编址):分为块内+模块地址,并行性提高,流水线存取

cache:
CDRAM:带 cache 的 dram
某程序命中 cache 的次数为n_c从主存中访问信息次数n_m,命中率

$$h = \frac{n_c}{n_c + n_m}$$

平均访问时间

$$t_a = h \cdot t_c + (1 - h) \cdot t_m$$

效率

$$e = \frac{t_c}{t_a}$$

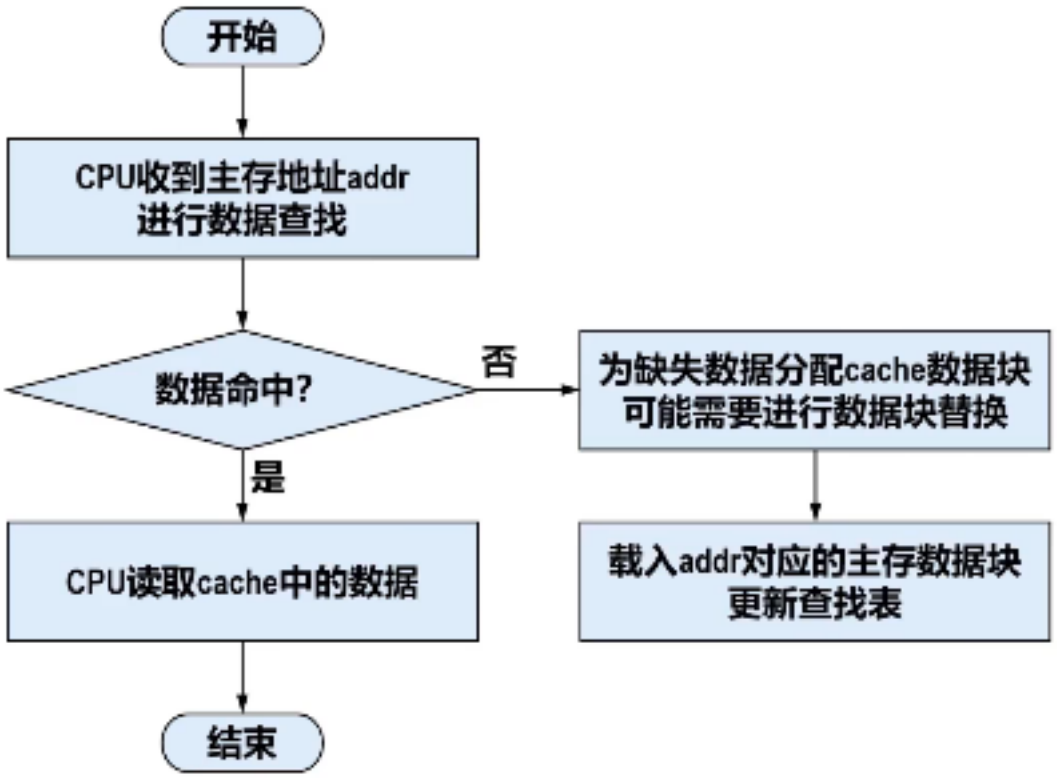


Figure 1: cache 的读流程

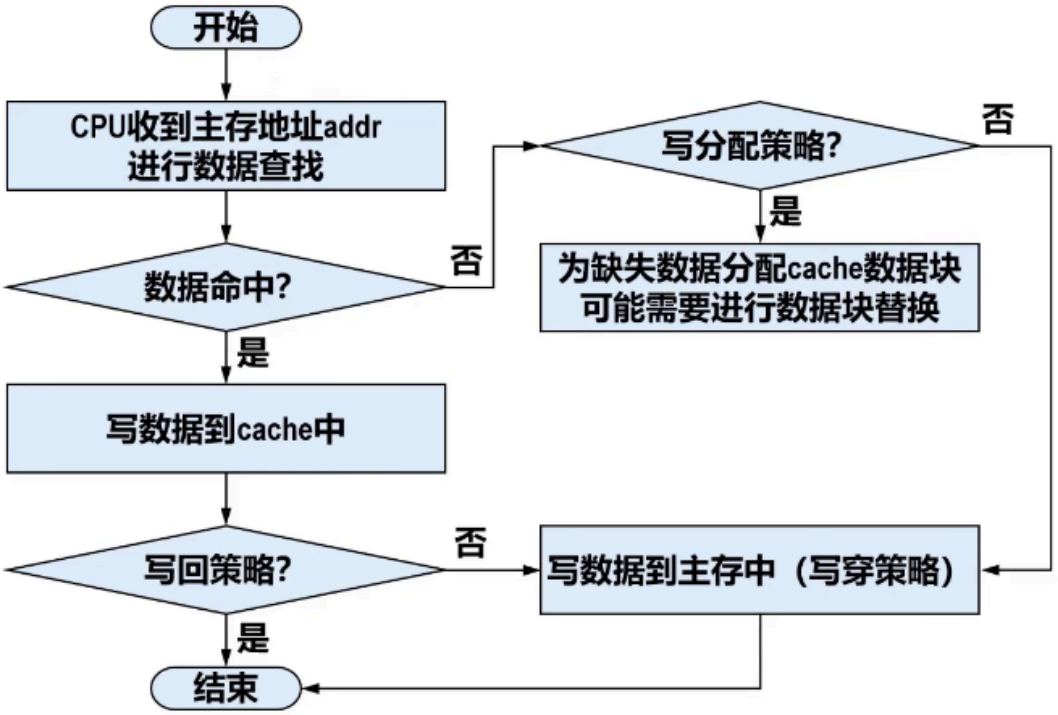


Figure 2: cache 的写流程

地址映射

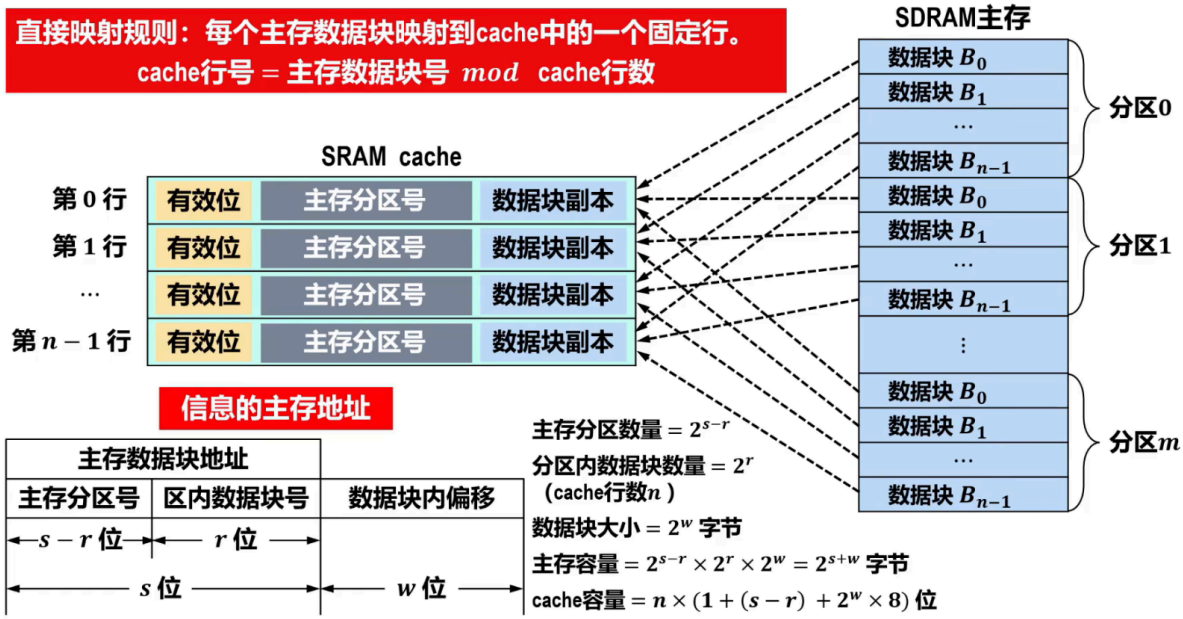


Figure 3: 直接映射

数据块副本存了一整个数据块,适合大容量 cache 使用

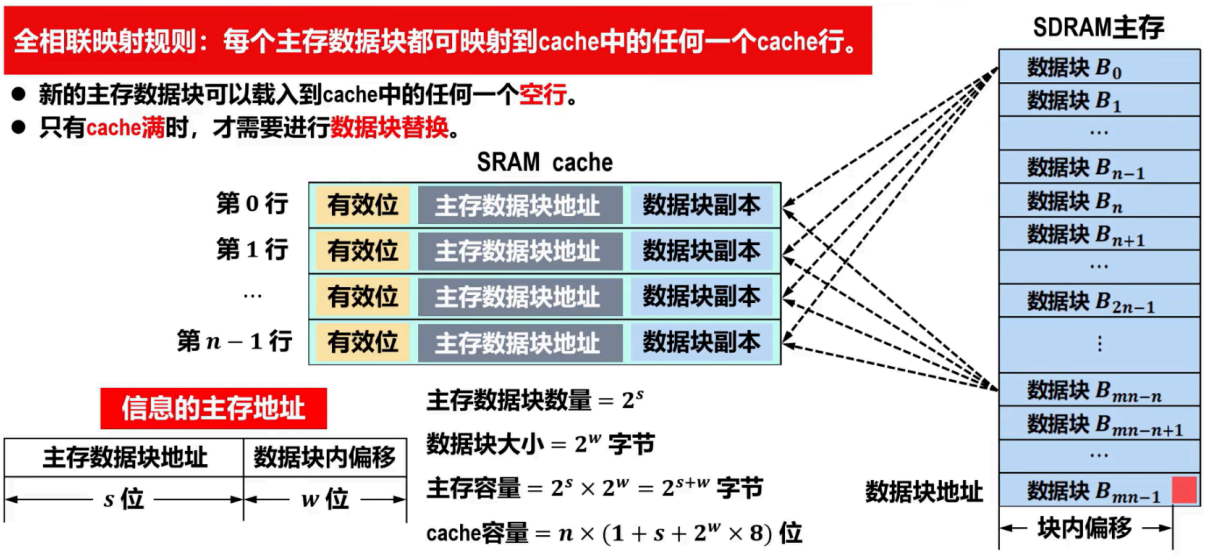


Figure 4: 全相联映射

适用于小容量 cache,每个数据块都存储在 cache 的任意位置

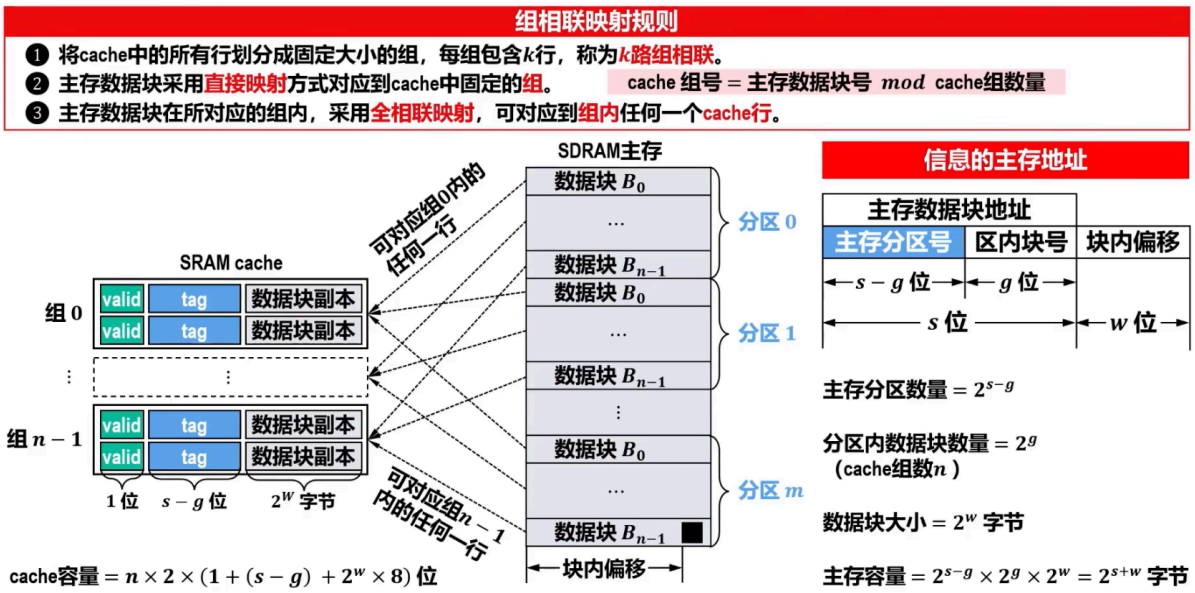


Figure 5: 组相联映射

主存直接映射至某组,之后采用组内全相联,主存地址构成方式与直接映射相同

cache 替换算法

- 先进先出:FIFO,最早进入的被替换,记录载入时刻作为指标
- 最近最少使用:LRU,淘汰计数器在命中自己时清零,记数最小的淘汰
- LFU:为每个 cache 行设置淘汰计数器,计数最少的淘汰

写策略:

- 命中时
  - 写回法:只写入 cache 行而不写入主存,只有该 cache 行中数据被替换时才将脏数据写回主存,置有一个 dirty 位,DMA 操作可能数据错误
  - 写穿法:命中时同时写入 cache 行和主存,不需要 dirty 位,但会增加主存访问次数,降低性能

- 未命中时
  - 写分配法:
    - ①写穿方式:将新数据写入到主存中相应数据块的位置,在 cache 中分配行载入
    - ②写回方式:在 cache 中分配行后载入新数据所在主存数据块
  - 非写分配法:仅写入主存中对应位置不载入 cache

虚拟页式存储器

虚拟存储器:主存和辅存构成的、单一的、可供 CPU 直接访问的大容量主存,虚页 VP 称逻辑页,实页 PP 称页帧

请求分页:活跃的页面在主存中,不活跃的页面在辅存中,当 CPU 访问辅存中的页面时,发生缺页异常,将辅存中的页面调入主存

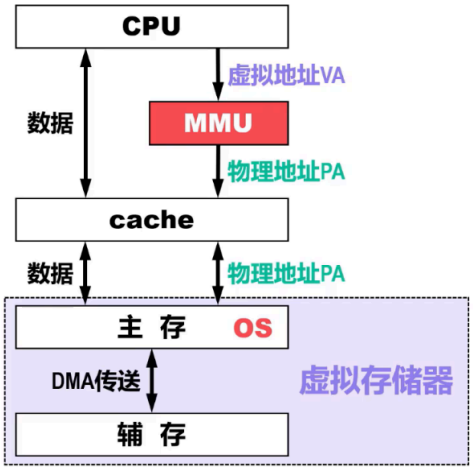


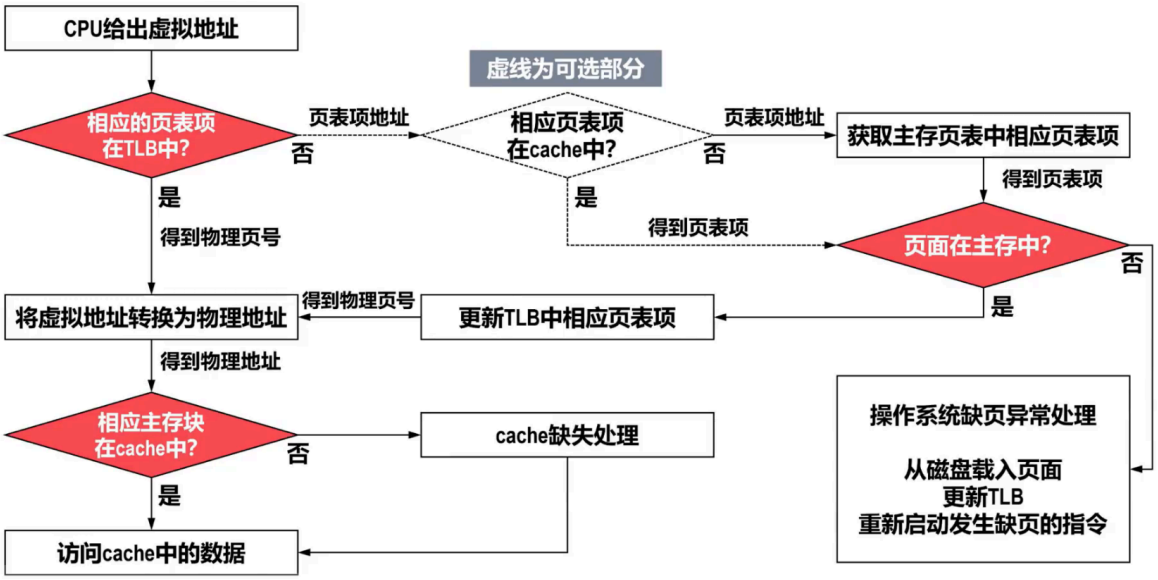
Figure 6: 虚拟存储器的组织结构

物理 addr:物理页号+页内偏移

虚拟 addr:虚页号+页内偏移

虚拟页号比物理页号长,作为表索引,两者页内偏移相同

TLB:一个容量较小的 cache 仅用于存页表项,用组相联与全相联(与主存中的页表(慢)类似, 性能更好)



ch4 指令系统

指令系统分为

- CISC:
  - 8/2 原则:80% 的程序只使用 20% 的指令
  - 大多数程序只使用少量的指令就能够运行
- RISC:
  - 一个有限的简单的指令集
  - CPU 配备大量的通用寄存器
  - 强调对指令流水线的优化

一个完善的指令系统应满足

- 完备性
- 有效性
- 规整性
  - 对称性:所有的寄存器和存储单元地位相同
  - 匀齐性:一个性质的指令平等看待各种数据类型
  - 指令格式和数据格式的一致性:指令长度和数据长度有一定的关系
- 兼容性:向上兼容

按形式划分:

- 三地址指令:每条指令有三个操作数,源操作数 1、源操作数 2、一个目标操作数
- 两地址指令:每条指令有两个操作数,源操作数 1 或一个目标操作数、源操作数 2
- 一地址指令:每条指令有一个操作数,源操作数 1 或一个目标操作数(或由 OP 约定隐含某个特定寄存器)
- 零地址指令:每条指令没有操作数　e.g.　NOP,HLT etc.

按功能划分可分为:

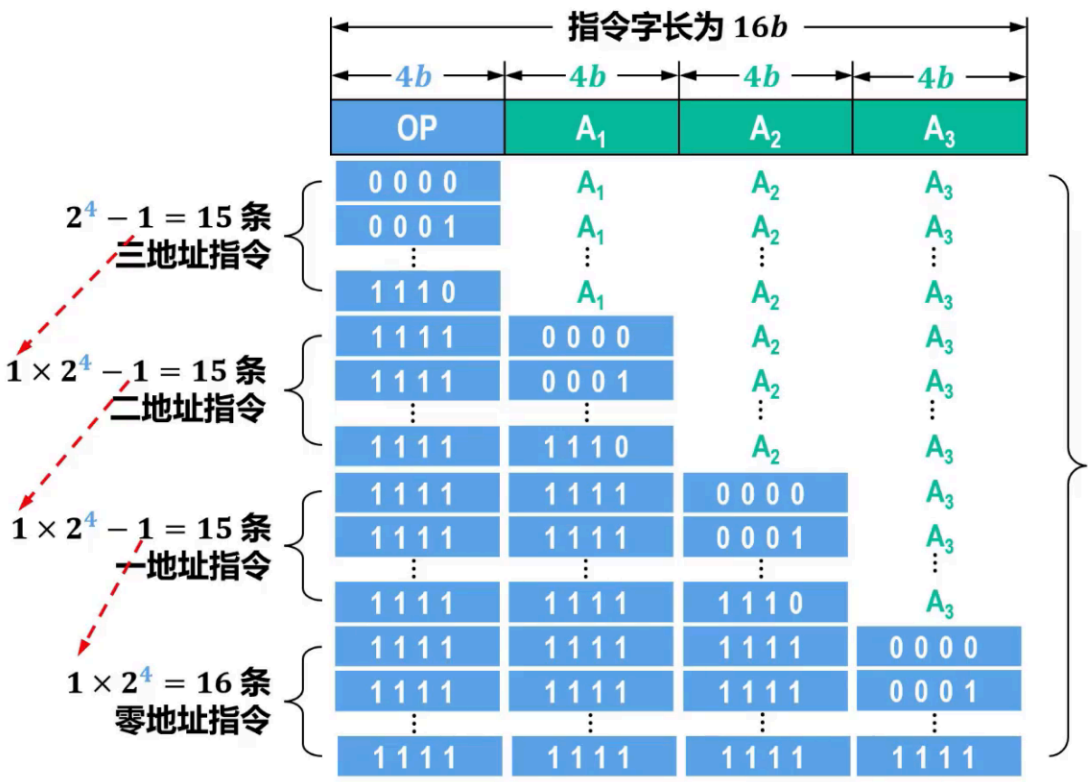
- 数据传送指令
- 算术逻辑运算指令
- 程序控制指令
- IO 指令
- 其他指令·

二地址指令按操作数的来源又可分为:

- RR:两个操作数均来自寄存器的指令
- SS:两个操作数均来自内存的指令
- RS:操作数分别来自寄存器和内存的指令

扩展操作码:取三地址指令前四位如 1111 前缀 OP,二地址指令视作 OP 为 1111xxxx

【举例】



顺序寻址:每读一条指令，自动 PC + 1(指令字长算下一条指令的有效地址)
跳跃寻址:分为条件、无条件
隐含寻址:一地址指令规定的东西
立即寻址:OP + 寻址方式码 + ADDR/DATA,操作数与指令一起存在主存,直接从指令寄存器 IR 中获取操作数
直接寻址:操作数与指令分开存，形式地址填操作数的主存地址
寄存器寻址:形式地址填所在(操作数)寄存器编号
间接寻址:形式地址填操作数间接地址，该间址指向主存操作数有效地址
寄存器间接寻址:形式地址写操作数地址通用寄存器的编号
相对寻址:形址填相对于当前 PC 的偏移量(补码表示)(找下一条指令的偏移)
变址寻址:形址填寄存器 n 内容与地址,两者相加得操作数主存地址
基址寻址:变址 n 可变，形址不变，基址反之
堆栈寻址:在主存里开一块高地为栈底栈顶(看方向),SP ± 1(SP 是寄存器/指针)

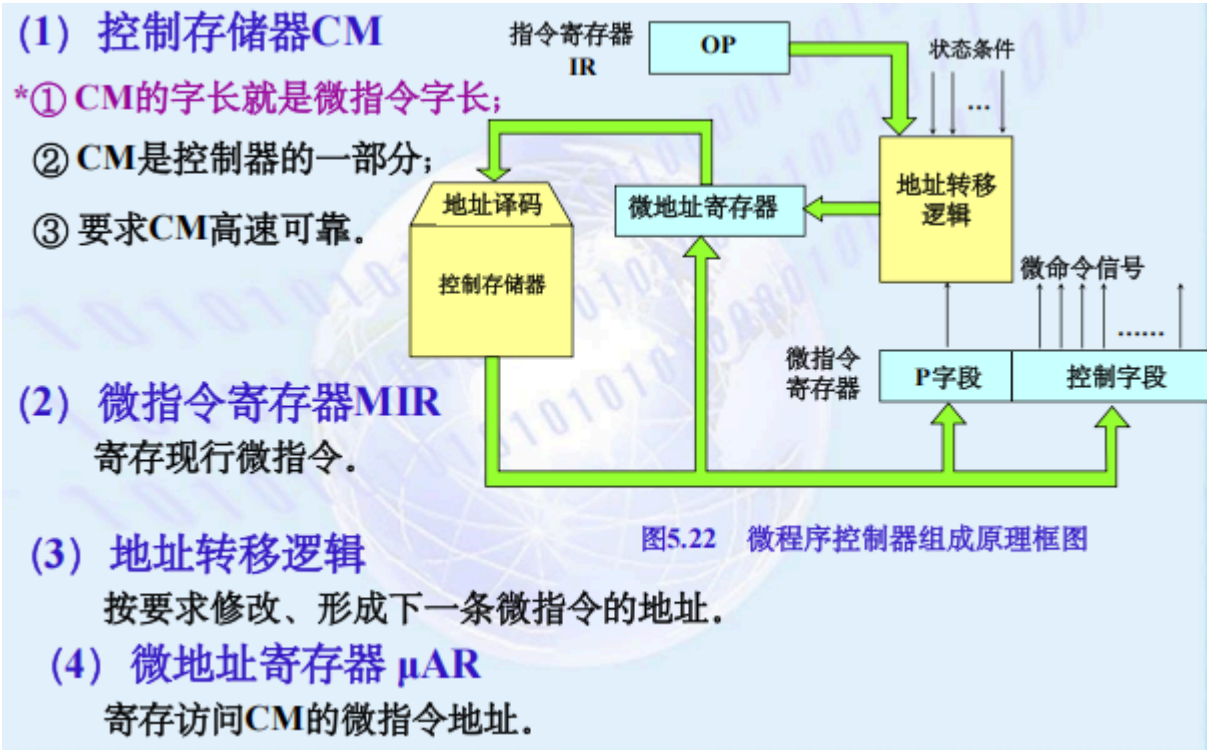
ch5 中央处理器

CPU 的基本部分由运算器,cache,控制器组成
CPU 的基本功能:

- 指令控制
- 操作控制
- 时间控制
- 数据加工

节拍周期:CPU 执行一个微操作命令的最小时间单位
机器(CPU)周期:CPU 从内存中读或写一条指令字所需要的最短时间
微指令周期:从控存取出并执行一条微指令所需要的时间,一般与一个 CPU 周期相当
指令周期:CPU 从主存中取出并执行一条指令所需时间

无条件转移:取指 译码 执行
间址:取指 间址 执行
执行后出现中断请求，会进入中断响应
取指:根据 PC 从主存中取出指令送入 IR
间址:根据指令中的地址码从主存中获得操作数的有效地址
中断:保存断点,将中断服务程序的入口地址送入 PC(堆栈寻址)
取指:PC→MAR　1→R　$M_{访存}$(MAR)→MDR　MDR→IR　OP(IR)→指令译码器　(PC)+1→PC
间址:(IR)→MAR　1→R　M(MAR)→MDR　MDR→(IR)
微程序:由若干条微指令构成(同一时钟能并行)
微程序控制器:



水平微指令:一条能定义多个可并行的微指令，执行快但编写麻烦
- 直接方式:操作控制字段每一个位表一个命令,n 个命令要 n 位字段
- 字段直接编码:①互斥性放在同一段内 ②每段信息位不宜过多 ③每一小段留一个方式代表(000)无操作
- 字段间接编码:多过一层译码，并行控制更弱

垂直型:一条微指令只定义一个命令,如 OP + 目的地址 + 源地址
CPU 的主要寄存器:
- 数据缓冲寄存器 DR
- 指令寄存器 IR
  - 取出一条指令时,指令总线从 cache 取到 IR
  - 对于 OP 字段,从 IR 到指令译码器到操作控制器
- 程序计数器 PC
- 数据地址寄存器 AR
- 累加器 AC
- 状态条件寄存器 PSW,就是 ctm3-xPSR

- 指令缓冲寄存器 SIR

流水线:
标量流水线:在每个时钟周期只发射一条指令,要求每个时钟周期只从流水线流出一条指令的结果
超标量流水线:标量流水线的一改成多
流水线的性能指标:
- 吞吐率:设任务数为 n,处理 n 个任务用时 T_k, $\mathrm{TP} = \frac{n}{T_k}$
- 装入时间:指令第 1 个阶段从开始到最后放入
- 排空时间:同上，改为最后 1 个阶段
- 加速比:并行比串行时间
- 效率:设备利用率(看面积)

流水线断流影响因素,前两个称为局部性相关,后面的称为全局性相关:
①访问资源冲突
②数据冲突(未写回上一步内容就被主存读)
③控制冲突(遇到转移指令等改变 PC 取值而断流,或者触发中断)
解决方案:
①暂停相关指令或资源重复配置
②暂停相关指令(NOP),调整指令顺序,数据旁路
③分支预测,预取多指令,推测取状态码
五段式:兼并耗时更长的段,分为 IF、ID、EX、M、WB 段,每段后均有一个锁存器

ch6 总线:
分为片内总线,系统总线,IO 总线
总线上信息传送分为五个阶段:
- 请求总线
- 总线仲裁
- 寻址
- 信息传送
- 状态返回

总线周期:一次总线操作所需时间
总线时钟周期:机器的时钟周期
工作频率:总线周期的倒数
时钟频率:总线时钟周期的倒数
总线宽度:数据总线位数
总线带宽:每秒传输数据位数
猝发传输:只读一次地址自由向后读多位
集中仲裁:由总线控制器判优先级
①链式查询:分为　BG 总线允许,BS 总线忙(获得总线设备发出),BR 总线请求 三个信号
离控制器越近优先级越高,对故障敏感,共$\lceil \log_2^n \rceil + 2$
②计数器查询:多了一根 BG,多了一组设备地址线,计数从上次终点开始
③独立请求:每个设备均有一对 BR 与 BG 与单条 BS(共 2n + 1)
分布仲裁:类似抽奖号码(优先级越大越容易被抽中)
四阶段:申请分配、寻址、传输、结束

UART 总线时序:
- 起始位:先发出一个逻辑 0 表示传输字符的开始

- 数据位:可以是 5 至 8 位逻辑 0 或者逻辑 1,可以是 7 位 ASCII 码也可以是 8 位的 BCD 码,小端传输 LSB 先发 MSB 后发
- 校验位:举个简单的例子,如果传输机制设置为偶检验,1 的个数为奇数,则输出 1
- 停止位:是一个字符数据的结束标志,可以是若干位的高电平,用于同步,停止位越长容错率越高
- 空闲位:一直是 1 的话就表示无数据传送

ch7 外存与 IO 设备:

外部设备组成:

- 存储介质
- 驱动装置
- 控制电路

外部设备分类:

- 输入设备
- 输出设备
- 外存设备
- 数据通信设备
- 过程控制设备

VRAM 容量=分辨率×灰度级位数　VRAM 带宽=VRAM 容量×帧率(速度)

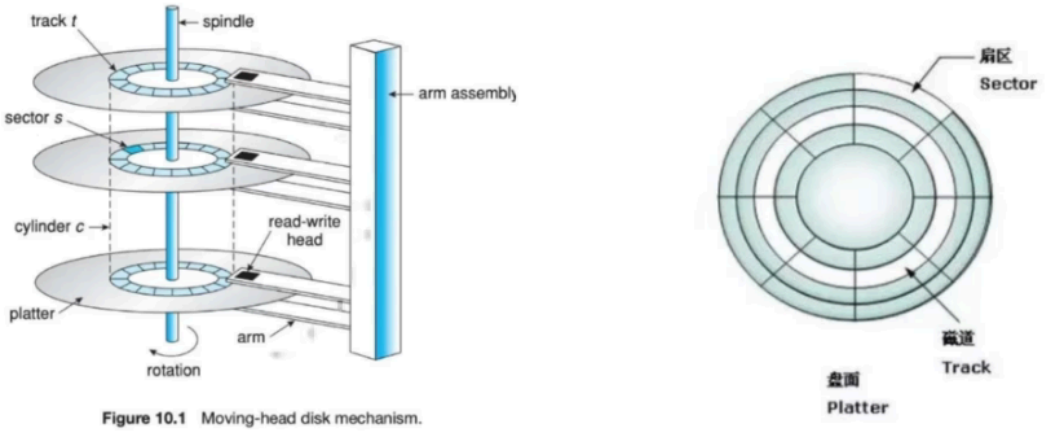


Figure 7: 磁盘结构图

最上和最下的记录面通常不用作存储(因为容易受灰尘和损伤影响，且读写头无法很好地贴近)
柱面:数同心圆
盘组总储存容量:全用最内道算,乘以柱面就行
寻道时间:

$$T = M(\text{相邻时间}) \times n(\text{间隔数}) + s(\text{磁头启动})$$

旋转延迟:

$$T = \frac{1}{2 \times r_{(\text{转速})}}$$

传输时间:

$$T = \frac{b}{r \cdot N}$$

其中 r 为转速,b 为需读取的数据块数量,N 为单条轨道总字节数
如果某文件长度超过了一个磁道的容量,应将它记录在同一个柱面上,这样就不用重新寻道,读写速度快

ch8I/O 系统

程序查询(轮询)方式:不断轮询 I/O 控制器中的状态寄存器直到任务完成
程序中断方式:可先去执行其他任务,I/O 完成后控制器发中断等 CPU 响应
单中断服务程序包含:保护现场,中断处理事件,恢复现场,开中断,中断返回
多级中断和嵌入式一样的不赘述了
DMA 方式:每完成一块大小的 I/O 再中断

- 在 DMA 工作时，停止 CPU 访问内存的操作
- 周期挪用:CPU 让出一个周期优先 DMA,但每次传送都要请求、响应和两次访问权限交换
- 交替访存:适用于 CPU 周期大于主存存储周期的 DMA

通道控制:弱化版本 CPU,CPU 不再直接连 I/O 设备
I/O 指令:(CPU 指令的一部分)操作码(识别 I/O 指令)　命令码(做何操作)　设备码(对何设备)
通道指令:提前放入主存中

ch9 并行组织与结构

访存模型包括:

- SIMD:单指令多数据流,所有处理器执行同一指令,处理不同数据
  - 特点：高效率处理向量/数组数据
  - 应用:向量处理器、DSP、GPU 等
  - 例子:FPGA 相比 DSP,在特定乘加运算上能实现更高的并行度和性能
- MIMD:多指令多数据流:每个处理器独立执行不同指令，处理不同数据
  - 多处理器(单地址空间):所有处理器共享一个逻辑上的地址空间
    - 集中式存储器(UMA):所有处理器访问任何存储单元的时间相同
    - 分布式存储器(NUMA):每个处理器有自己的本地存储器,访问本地存储快,访问远程存储慢
  - 多计算机:每个计算机有独立的地址空间,通过消息传递通信
    - 松耦合多计算机
    - 紧耦合多计算机

结构模型分为:

- 共享储存多处理机
  - 紧耦合 MIMD
  - 所有存储器构成单一地址空间,共享变量方式实现通信,处理机可以访问所有存储器且时间相同
- 并行向量处理机 PVP
  - 每个 VP 按地址可访问所有共享存储器
  - 同时实现任意节点互连,采用开关实现 n 入到 n 出
- 分布存储多处理机
  - 松耦合 MIMD
  - 所有存储器构成多个地址空间
  - 采用消息传递方式实现通信,处理机只能访问局部存储器
- 分布共享存储多处理机 DSM
  - 可以是松 or 紧耦合 MIMD
  - 其他同共享储存多处理机,就是由于互联网络延时会出现访问时间不同
- 对称多处理机 SMP

- 有两个以上功能相似的处理机,能完成一样的功能
- 处理机共享主存,IO,以某种内部互联机制连在一起(存储器对处理机时间大致相等)
- 系统被一个集中式操作系统 OS 控制
- 为了便利来自 IO 处理器和 DMA 的传送,具有寻址,仲裁,分时共享功能