90-复习

01 概述

组织架构

	对程序员	特点	包括	例子
组织	不可见	操作单元及其相互连接（硬件的内部实现）	控制信号、存储技术	使用乘法器/加法器完成乘法
结构	可见	直接影响程序逻辑执行的属性（硬件给编程者的“接口”）	指令集、各类数据类型的大小	提供/不提供乘法指令

冯诺依曼架构

- 运算器、控制器、存储器、I/O

评估指标

- $I_C = \sum_{i=1}^{n} I_i$: 指令条数
- $CPI = \frac{\sum_{i=1}^{n} CPI_i \times I_i}{I_C}$
- $T = I_C \times CPI \times t$: 给定程序的执行时间
- $\text{MIPS} = \frac{I_C}{T \times 10^6} = \frac{f}{CPI \times 10^6}$

- $\text{MFLOPS} = \frac{N_{FLO}}{T \times 10^6}$

错题

- 注意是速度还是时间的比值
- 区分指令数据取决于指令周期的不同阶段
- ALU 和 通用寄存器 一定和机器字长相同

03 数据表示

- 浮点数的非规格化数指数还是`-126`，尾数是`0.xx`

舍入

- 就近：多余部分0开头掐掉，1开头进1
- 朝正无穷：正数进1，负数掐掉（因为负数尾数越大值越小）
- 朝负无穷：负数进1，正数掐掉
- 朝0：掐掉

错题

- IEEE754的`float`类型中，对应到十进制有效位数为**6-7位**，如果超过7位的整数转为`float`类型，会丢失精度
- 注意规格化/非规格化
- IEEE754 能表示的最大数是$2^{128} - 2^{104}$
- 十六进制表示最后加个h

04 校验码

奇偶校验

- 奇校验多xor一个1

海明码

校验码长度

- 假设数据长度为M 校验码长度为K 最多有1位发生错误
- 那么所有情况是 M+K+1
- 为了覆盖所有情况，$2^K \geq M + K + 1$

数据位划分

- 二进制下只有1个1的位置留给校验码
- 校验码异或所有对应位置为1的数据

错误情形

- 若$C'' \oplus C'$
  - 全为0：没有错误
  - 仅1位1：校验码出错（因为数据位不会只有1个1）
  - 多位1：数据出错，对应数据取反即可

CRC

- 适合用流格式传输存储的数据
- 步骤
  1. 提前约定好生成多项式
  2. 假设数据有M位，左移数据K位（右侧补0），并用K+1位生成多项式除它(模2运算)（0-1=1）
  3. 采用K位余数作为校验码，把校验码放在数据（不含补的0）后面，一同存储或传输
- 如果M+K位内容可以被生成多项式除尽，则没有检测到错误

05-07 运算

- 浮点数乘除要加减偏移量
- NBCD 加减进位要加减6

延迟

- 与/或门：1ty
- 异或门：3ty
- 复杂组件中，能预先计算的都可以预先计算，总时间取决于最慢的那个
- 串行进位加法器
  - $C_i = C_{i-1} + 2\ \text{ty} \to C_n = 2n\ \text{ty}$
  - $S_1 = S_2 = 6$
  - $S_n = C_{i-1} + 3\ \text{ty} \to S_n = 2n + 1\ \text{ty}\ (n \geq 3)$
- 全先行加法器：能算的都算完再一起算
  - 计算G_i, P_i：$1\ \text{ty}$
  - 求出所有的C_i：$2\ \text{ty}$
    - 先并行计算所有的与：$1\ \text{ty}$
    - 再并行计算所有的或：$1\ \text{ty}$
  - 最后通过异或求出S_i：$3\ \text{ty}$
- 部分先行加法器：串联全先行加法器
  - $8n$位加法，串联n个8-bit的CLA
  - 第一个计算出给下一个的C_8需要：$1 + 2 = 3\ \text{ty}$，S_i不与后续元件相关，时间忽略
  - 此后每一个的G_i, P_i都预先算好，计算出给下一个的C_{8i}需要$2\ \text{ty}$
  - 最后一个计算出C_{8n}需要$2\ \text{ty}$，S_{8n}需要$3\ \text{ty}$
  - 因此共需$2n + 4\ \text{ty}$
  - 例如32位就是$12\ \text{ty}$

错题

- 逻辑右移补0，算术右移补符号位
- 判断溢出：异号相加不会溢出，同号相加若符号位不同则溢出
- `unsigned short`转`unsigned int`时，高位补0
- 左规是阶码减小，右规是阶码增大

08 内部储存器

分类

类型	种类	可擦除性	擦除级别	写机制	易失性
随机存取储存器 RAM	读-写储存器	电可擦除	字节级	电	易失
只读储存器 ROM	只读储存器	不可擦除	-	掩膜（光刻）	非易失
可编程ROM PROM	主要进行读操作的储存器	不可擦除	-	电	非易失
可擦除 PROM EPROM	主要进行读操作的储存器	紫外线可擦除	芯片级	电	非易失
电可擦除 PROM EEPROM	主要进行读操作的储存器	电可擦除	字节级	电	非易失
快闪储存器	主要进行读操作的储存器	电可擦除	块级	电	非易失

- 地址空间：单元总数
- 寻址能力：单元储存信息的位数

- RAM：易失，供电中断后数据消失
  - DRAM：周期性充电刷新 **用于内存**（SRAM 集成度太低）
  - SRAM：有电源就可以维持数据 **用于 Cache**（DRAM 速度慢）
- 大小端
  - 大端：高位在前
  - 小端：低位在前
- 随机访问：无论访问哪个地址，时间都是一样的
- 位拓展：增加每一个字节的位数，会改变数据长度，不会改变地址
- 字拓展：增加地址位数，会改变地址，不会改变数据长度

错题

- 交叉编址：多路储存器，错开访存，在连续读写时提高效率，在随机读写时可能会产生冲突
- 数据引脚的数量是数据位数，不是 log2 数据位数
- 刷新操作次数：单个芯片的行数=总容量开根号（默认是方形的）
- DRAM采用地址复用，地址线数要减半

09 Cache

计算

- p：命中率、T_C：Cache访问时间、T_M：主存访问时间，则
- 使用Cache的平均访问时间：$T_A = T_C + (1-p) \times T_M$
- 多级的平均访问时间：L_i需T_i时间访问，L_1 - L_i联合命中率为P_i，$T_A = T_1 + \sum(1-P_i) \times T_{i+1} + (1-P_N) \times T_M$（注意：无论是否hit都要一级一级访问下去）

映射方式

- 直接映射：内存行号取模Cache行数，tag为取模部分以外的部分

- 全关联映射：无规则，tag为内存行号
- 组关联映射：X路组相连，Y组，内存行号取模Y

替换方式

- LRU：最近最少使用（例如两路中最近访问的设为1，另一个设为0，下次替换0）
- FIFO：先进先出，设置时间戳
- LFU：最少使用，设置计数器
- 随机：随机替换

写回方式

- 写直达：写Cache的同时写主存
- 写回法：只写Cache，替换时写回主存，使用 Dirty Bit

错题

- 字、块都是从0开始
- Cache行位数 = Tag + 有效位（1） + 脏位（仅限写回） +Used Bit（仅限LRU）+ 数据位（Byte*8）

10 外部储存器

- 扇区：512B
- 光盘是螺旋线，磁盘是同心圆
- 平均访问时间：$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$
  - b：传送的字节数，r：旋转速率，转/秒，N：每磁道的字节数（扇区数和扇区字节数之积）
  - T_s **平均**寻道时间
  - $\frac{1}{2r}$ **平均**旋转延迟

- $\frac{b}{rN}$ **实际**传送时间 若有数据传输率的数值直接用数据传输率来算 $\frac{b}{N}$可以是传输/磁道的扇区数的比值，也可以是数据大小的比值
- 若单位为rpm，转换为rps
- 读取多个相邻磁道，只需考虑1次寻道时间，每个磁道都需考虑旋转延迟

错题

- 数据相关的K M用2^{10} 2^{20} 其他用 10^3 10^6
- `Mbps` = `M` * `bps` 此处`M`是10^6
- 注意`MBps` or `Mbps`
- 最大数据传输率：不需要寻道，不需要转，只读一个扇区，不需要考虑间隙
- 当给出读/写的数据尺寸时，涉及到多个磁道时，读写前后的磁头位置**不同**
  - 切换磁道需要额外加上寻道时间
- 随机存取：到任何一个位置的时间差不多

11 RAID

级别	描述	磁盘数量	种类	数据可用性	大I/O输出传输能力
0	非冗余	N	条带化	最低	很高
1	镜像	$2N$	镜像	比2，3，4，5高，比6低	读：比单盘高 写：类似
2	海明码冗余	$N+m$	并行存取	高	最高
3	位交错奇偶校验	$N+1$	并行存取	高	最高
4	块交错奇偶校验	$N+1$	独立存取	高	读：RAID0 写：低于单盘
5	块交错分布式奇偶校验	$N+1$	独立存取	高	读：RAID0 写：低于单盘
6	块交错分布式奇偶校验	$N+2$	独立存取	最高	读：RAID0 写：低于RAID 5

- RAID 1 是镜像，更可靠
- 不用 RAID 2：海明码冗余盘太多
- 不用 RAID 3：位交错奇偶校验，只能同时处理一个请求
- 不用 RAID 4：奇偶校验在一块盘上，有瓶颈
- RAID 5 校验分布在所有磁盘上，通过异或恢复错误
- RAID 6 两个校验盘，最可靠
- 需要计算校验码的 RAID 写入时间较长
- RAID 5 读写时间：中间存在异或的过程
  - 最坏：2读2写，计算需要已有数据
  - 最好：1读1写，已有数据全都不复存在

错题

- 并行存取：每个I/O请求由所有磁盘同时响应
- 独立存取：并行处理多个请求

12 虚拟内存

TLB	Page	Cache	说明	主存访问	硬盘访问
Hit	Hit	Hit	直接从Cache获取数据	0	0
Hit	Hit	Miss	将数据从主存读入Cache后获取	1 写Cache	0
Miss	Hit	Hit	将页表读入快表 直接从Cache获取数据	1 读页表	0
Miss	Hit	Miss	将页表读入快表 将数据从主存读入Cache后获取	2 读页表+写Cache	0
Miss	Miss	Miss	将页表读入快表 将数据同时读入主存与Cache后获取	2 读页表+写Cache	1 主存读数据（同时主存自己写入页表相关）
Hit	Miss	Miss	不可能		
Hit	Miss	Hit	不可能		
Miss	Miss	Hit	不可能		

- 页表的行数：$2^{\text{虚拟地址位数}-\text{页面大小位数}}$
- 页缺失，主存中一定不存在对应的数据，Cache中一定没有对应缓存

- 页缺失，页表中一定不存在对应的映射，TLB中一定没有对应缓存
- Cache Miss：硬件处理
- Page Miss：OS处理
- TLB Miss：硬件/OS处理均可
- TLB 缺失了 页表不可能缺失
- TLB 使用全关联映射，使用虚页号做 Tag
- TLB使用写回策略：磁盘访问开销大

13 指令

- 寻址方式
  - 立即寻址：立即数
  - 直接寻址：跳一次，访问指令中的地址
  - 间接寻址：跳两次，扩大寻址范围
  - 寄存器寻址：直接访问寄存器
  - 寄存器间接寻址：跳一次，访问寄存器中的地址
  - 偏移寻址
    - 相对寻址 `EA` = `PC` + `A`
    - 基址寄存器寻址：`EA` = `Base` + `A`
    - **变址寻址**：`EA` = `A` + `(R)`
  - 栈寻址：栈指针
- 指令集设计，不能有二义性
- 指令周期

14 流水线

寄存器

- PC 程序计数器：存有待取指令的地址

- IR 指令寄存器：存有最近取来的指令
- MAR 存储地址寄存器：存有存储器位置的地址
- MBR 存储缓冲寄存器 / MDR 存储数据寄存器：存即将写入/刚刚读出存储器的字
- PSW 程序状态字：包含状态信息的寄存器

六阶段方法

简称	全程	任务
FI	取指 Fetch Instruction	读下一条预期的指令到MBR
DI	译码 Decode Instruction	确定操作码、操作数指定符
CO	计算操作数 Calculate Operands	计算源操作数的有效地址
FO	取操作数 Fetch Operands	从存储器中取操作数
EI	执行 Evaluate Instruction	完成指定操作，若需写入寄存器则写入
WO	写操作数 Write Operand	将结果存入存储器

流水线性能

- 假设
  - 流水线第i段电路延迟时间 t_i
  - 最大段延迟 t_m
  - 指令流水线段数 k
  - 锁存延时 d 数据和信号从上一段传到下一段所需的段间锁存接收时间
- 周期时间 $t = \max[t_i] + d = t_m + d$
- k阶段流水线指令执行n条指令耗时 $T_{k,n} = [k + (n-1)]\,t$
- 加速比 $S_k = \frac{nkt}{[k+(n-1)]t} = \frac{n}{1+\frac{n-1}{k}}$
- 误解：阶段数越多，执行速度越快
- 原因：阶段间移动数据&控制逻辑存在开销，随阶段增加急剧增加

冒险

- 结构冒险：多个指令需要同一硬件
- 数据冒险：指令需要的数据还未准备好（前一条指令还未写入更新后的数据，后一条指令就需要这个数据），最多影响3条指令
- 控制冒险：分支预测错误
    - 动态预测的状态转移

15 控制器

- 输入：指令、标志、时钟、来自控制总线的信号
- 输出：控制信号
- 实现方式
    - 硬布线
    - 微程序：控制存储器
        - 微指令：控制信号+地址（1-2个）

微操作

取指周期

```
t1: MAR <- (PC)
t2: MBR <- Memory(MAR)
 PC  <- (PC) + I
t3: IR  <- MBR
```

间址周期

- 完成间址周期后，IR的间址被替换为操作数，可正常执行

```
t1: MAR <- (IR(Addr)) // 指的是取 IR Addr的这几位，后同
t2: MBR <- Memory(MAR)
t3: IR(Address) <- (MBR(Addr))
```

执行周期

- 取决于不同指令

```
// ADD R1, X
t1: MAR <- (IR(Addr))
t2: MBR <- Memory(MAR)
t3: R1  <- (R1) + (MBR)

// BSA X 转移并保存地址 将地址保存在X，从X+1继续执行
t1: MAR <- (IR(Addr))
 MBR <- (PC)
t2: PC  <- (IR(Addr))
 Memory <- (MBR)
t3: PC  <- (PC) + I
```

中断周期

```
t1: MBR <- (PC)
 MAR <- Save_Addr
t2: PC  <- Routine_Addr
 Memory <- MBR
```

错题

- 微指令有x个互斥类，每个类有a_i条指令，则控制字段需要$\sum\lceil\log_2 a_i\rceil$位
- 控制器不在指令流水线数据通路中

16 总线

- 分类

    - 内部总线：CPU内部
    - 系统总线：CPU与主存与控制器
    - 通信总线：系统间
- 仲裁方式
    - 集中式：菊花链、独立请求、计数器
    - 分布式
- 时序
    - 同步时序：时钟信号
    - 异步时序：握手
    - 半同步：时钟+握手
    - 分离事务：将时间前后拆分
- 带宽：理想值，最大速率
- 数据传输率：实际值，每秒传输的数据量
    - 同步时序中每一步须向上取整到时钟周期的整数倍
    - 连续取数据时可以边取边传，取最大值
    - 异步时序中2-4步和访存时间取最大值

步骤	内容	意图
1	CPU设置地址 ReadReq线	CPU告知存储器需读取
2	存储器读取地址 设置Ack线	存储器告知CPU地址读取完毕
3	CPU释放地址线和ReadReq线	CPU告知存储器已收到存储器的回应
4	存储器释放Ack线	完成地址传输，即将数据传输
5	CPU将数据传到数据线 设置DataRdy线	存储器告知CPU读取数据
6	CPU读取数据 设置Ack线	CPU告知存储器数据读取完毕
7	存储器释放数据线 DataRdy线	存储器告知CPU收到CPU的回应
8	CPU释放Ack线	完成数据传输

错题

- 一个总线事务需要至少x个周期时：至少有1个周期用来传输，至少有$x-1$个周期用来准备、传送其他信息
- 突发传输：连续传输多个数据
- 并行总线的速度不一定比串行快
- PCI-Express采用串行传输
- 计算带宽注意是否全双工、是否多通道

17 I/O

- I/O 模块的设计
- 编程式
  - I/O 命令：指定外设地址，发出命令

    - 计算轮询频率：通过传输速率和缓冲区大小计算
- 中断式
    - 若中断处理速度比I/O速度慢，则无法使用中断
- DMA：I/O设备直接访问内存
    - DMA 访问主存优先级高于CPU：防止缓冲区溢出造成数据丢失

解决CPU/DMA争用主存

- CPU停止法：DMA访存时向CPU发送信号，使CPU停止访问主存
    - 优点：控制简单
    - 缺点：影响CPU、没有充分利用内存
    - 适用：高速I/O设备的块传输
- 周期窃取：DMA单字传送，需传送时CPU让出1个周期，周期结束传送完立刻释放总线
    - 优点：充分利用CPU、内存，及时响应I/O
    - 缺点：DMA每次都请求总线
    - 适用：I/O周期大于储存周期
- 交替分时访问：每个周期分为两部分，前半部分给CPU，后半部分给DMA
    - 优点：CPU未停止/等待，DMA不请求总线
    - 缺点：CPU周期大于储存周期

> 例 假设中断系统中有4个中断源，其响应优先级为L1>L2>L3>L4，处理优先级为L1>L4>L3>L2。如果在主程序执行时同时发生L1、L3和L4中断，并且在处理L3中断的过程中发生L2中断，写出掩码字和所有中断服务程序的过程。

掩码字	L1	L2	L3	L4
L1	1	1	1	1
L2	0	1	0	0
L3	0	1	1	0
L4	0	1	1	1

错题

- 在单级(或单重)中断系统中，不允许中断嵌套。中断处理过程为：① 关中断；② 保存断点；③ 识别中断源；④ 保存现场；⑤ 中断事件处理；⑥恢复现场；⑦ 开中断；⑧ 中断返回。其中1-3由硬件完成，4-8由中断服务程序

上一页
17-输入输出

下一页
机考

最后更新于11个月前