计算机组成原理

Sec1 基础

- 概念
  - Input/Output
  - 编译器
  - 控制器
  - 数据通路
  - 处理器
  - 存储器
- 8个伟大思想
  - 面向摩尔定理的设计
  - 使用抽象简化设计
  - 加速大概率事件
  - 通过并行提高性能
  - 通过流水线提高性能
  - 通过预测提高性能
  - 存储器层次
  - 通过冗余提高可靠性
- 性能度量
  - 响应事件 (执行时间)
  - 吞吐率 (带宽)
  - 时间
    - CPU执行时间
    - 用户CPU时间
    - 系统CPU时间
    - 时钟周期时间
      - 时间频率
  - CPI
    - Cycle per Instruction
    - IPC = 1 / CPI

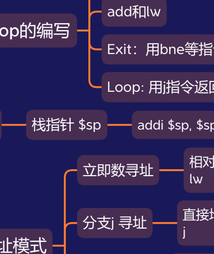
Sec2 计算机的语言

- 汇编语言
  - 算数 — add, sub, addi
  - 数据传输 — lw, sw
  - 逻辑 — and, or, nor, andi, sll, srl
  - 条件分支 — bne, beq, slt,
  - 无条件跳转 — j, jr, jal
- 操作数
  - 存储器操作数
    - 数据传输指令
      - big end, little end
      - spilling(寄存器溢出) — 将不常使用的变量存回寄存器的过程叫做~
- 数的操作
  - 二进制位
    - 字 — 字: 计算机的基本访问单位
    - 字节
    - 最高/低有效位 — most/least significant bit
  - 补码 — 取反加1
- MIPS字段
  - op | rs | rt | rd | shamt | funct
  - op | rs | rt | constant or address
  - sw/lw的字段中
lw $t0, 1200($t1)
$t0 为rt, $t1 为rs
  - add等字段中
sub $t2, $t1, $t0
$t2为rd, $t1为rs, $t0为rt
  - bne等的字段中
beq reg1, reg2, L1
- Loop的编写
  - 对i操作 — sll 2位 等于乘4（字节寻址）
  - add和lw
  - Exit：用bne等指令判断出口
  - Loop: 用指令返回
- 栈
  - 栈指针 $sp — addi $sp, $sp, -12 建立空间
- 寻址模式
  - 立即数寻址 — 相对地址
lw
  - 分支j 寻址 — 直接地址
j
  - 跳转寻址 — PC相对寻址
bne — PC和常数的和
  - 寄存器寻址 — 操作是寄存器

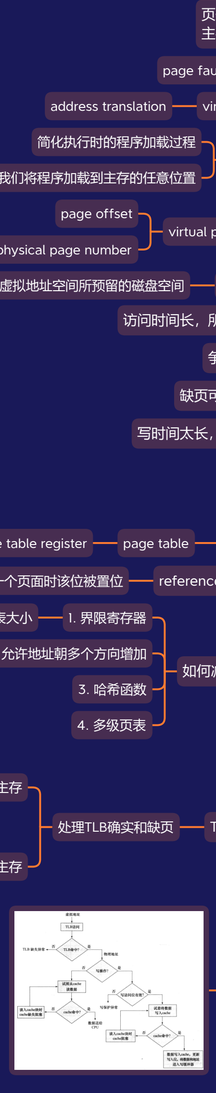
Sec3 计算机的算术运算

- 加法和减法 — 上溢下溢
- 乘法
  - 顺序乘法
    - multiplicand
    - multiplier
  - 乘法器硬件
    - version1
    - version2 — 将乘积和乘数放一块，一起右移
同时右移，被乘数，乘数寄存器和ALU改位32位
只有乘积寄存器是64位，ALU加左边就行
    - version3 — 直接通过加法器栈 — 乘数每一位为分配log32高的寄存器栈
- 除法
  - 操作
    - dividend
    - divisor
    - remainder
  - 思路
    - 除数寄存器64位
每次右移
    - 余数寄存器：开始为被除数
    - 不断用余数减去除数
根据大于0小于0情况商左移补1或者0
若小于0，则余数恢复原值
    - 最后无论如何除数都要右移
  - 除法硬件结构
    - 普通 — 64位除数，余数和ALU
32位商
    - 改进 — 只有余数为64位，商为余数寄存器右半部分
根据情况左移或者右移余数
将源操作数和商移位与减法同时进行
- 浮点数
  - 浮点表示
    - fraction 尾数 23位
    - exponent 指数 8位
  - biased notation
    - $(-1)^S \times (1 + Fraction) \times 2^{(Exponent-Bias)}$
    - 单精度：bias：127
双精度：bias：1023
  - 浮点数操作
    - 加法 — 规格化-> 指数小的数进行右移，变成指数大的数一样的指数 -> 尾数相加 -> 规格化再舍入
    - 乘法 — 指数相加(注意偏阶) -> 尾数相乘 -> 舍入并规格化

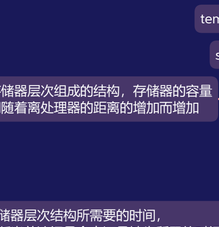
Sec4 处理器

- 处理器
  - 控制信号
    - RegDst — 写寄存器的目标寄存器号来自rd字段
    - RegWrite — 寄存器堆写使能有效
    - ALUSrc — 第二个ALU操作数为指令低16位的符号扩展
    - PCSrc — PC由分支目标地址取代
    - MemRead — 数据存储器读使能有效
    - MemWrite — 将写入数据输入端的数据
写入到用地址指定存储单元中取
    - MemtoReg — 写寄存器的数据来自数据存储器
  - 
- 流水线 pipelining
  - 5个操作
    - IF — instruction fetch
    - ID — instruction decode
read register heap
    - EX — execution
address calculate
    - MEM — memory
    - WB — write back
  - 流水线冒险
    - structural hazard
    - data hazard
      - forwarding/ bypassing
      - load-use data hazard — pipeline stall / bubble
    - control/branch hazard
      - 假定分支不发生 — 预测错了就flush
      - 缩短分支的延迟 — 把分支提前到ID级
原本为MEM级
        - 可以将分支预测错误减少到1条指令
        - 操作：
旁路单元
阻塞流水线
      - 动态分支预测
dynamic branch prediction — 观察上一次分支发不发生
        - branch prediction buffer
        - branch history table
  - 图形化表示 — IM - Reg - ALU - DM - Reg
- 异常处理
  - EPC(exception program counter)
  - 向量中断（vectored interrupt）
- 指令级并行
  - static multiple issue
  - dynamic multiple issue
  - 问题
    - issue slot
    - 数据冒险和控制冒险
  - speculation
  - 矩阵乘法加速? — 循环展开（loop unrolling） — 能获得更高程度的指令并行

Sec5 虚拟存储器

- idea
  - program address -> physical address
protection
  - 页偏移一致
主要是虚页号到物理页号
  - 概念
    - page fault — page
    - address translation — virtual address
    - relocation
      - 简化执行时的程序加载过程
      - 允许我们将程序加载到主存的任意位置
    - virtual page number
      - page offset
      - physical page number
    - swap space — 为进行的全部虚拟地址空间所预留的磁盘空间
- 考虑的因素
  - 访问时间长，所以页要足够大
  - 争取降低缺页率
  - 缺页可以用软件处理
  - 写时间太长，常用写回操作
- 关于写
  - dirty bit
  - dirty page
- 缺页故障
  - 页的存放和查找 — page table — page table register
  - reference bit（used bit） — 每当访问一个页面时该位被置位
  - 如何减少页表的容量? — 限定页表大小
    - 1. 界限寄存器
    - 2. 允许地址朝多个方向增加
    - 3. 哈希函数
    - 4. 多级页表
  - TLB 加快地址转换
    - 处理TLB确实和缺页
      - 页不在主存
        - 缺页故障
        - 数据访问引起的缺页 — restartable
指令要可以重新启动
异常前指令没有结束
      - 页在主存 — 替换即可
- 
- 
- 3C 模型
  - cold-start miss — compulsory miss
  - 全相联都容纳不了所有块的时候 — capacity miss
  - 组相联和直接映射 / 全相联没有 — collision miss — conflict miss

Sec5 存储器层次结构

- 引言
  - 目的
    - temporal locality
    - spatial locality
  - memory hierarchy — 一种由多存储器层次组成的结构，存储器的容量和访问时间随着离处理器的距离的增加而增加
  - 指标
    - hit rate/ratio
    - miss rate
    - hit time — 访问某存储器层次结构所需要的时间，
包括了判断当前访问是命中还是缺失所需的时间
    - miss penalty — 将相应的块从低层存储器替换到高层存储器所需要的时间，包括访问块、将数据逐层传输、将数据插入发生缺失的层和将信息块传送给请求者的时间
- 存储器技术
  - DRAM — 主存
速度更慢
  - SRAM — 靠近处理器那层
  - EEPROM — wear leveling
损耗均衡
- 磁盘存储器
  - track
  - sector
  - seek
  - rotational latency
- cache的基本原理
  - 概念
    - tag
    - valid bit
    - dirty bit
    - block
    - index
  - 原理
    - 
    - 地址：标记-索引-块偏移-字节偏移
  - 块的大小和miss rate，miss penalty 的关系
    - 越大
    - 越小
  - cache miss
    - pipeline stall
    - 处理步骤
      - 1. 保存PC
      - 2. 通知主存执行一次读操作
      - 3. 写cache项，将从主存取回的数据写入cache中存放数据的部分，并将地址的高位写入标记域，设置有效位
      - 4. 重启指令执行第一步，重新IF
    - 多级cache结构减少缺失代价
  - 写操作处理
    - write through
    - write back
    - write buffer
    - 策略
  - 性能评估和改进
    - 性能
      - CPI计算
      - AMAT = 命中时间+缺失率x缺失代价
    - 策略改进
      - 直接映射
      - 组相联
      - 全相联
    - 
  - 替换块的选择策略 — LRU（least recently used）
  - 分块进行软件优化（矩阵乘法） — blocking