计算机系统结构

1. 概论

1. 多级层次结构

5	应用语言机器级	应用程序包（翻译）	高级语言
4	高级语言机器级	编译程序	汇编语言、机器语言
3	汇编语言机器级	汇编程序	机器语言
2	操作系统机器级	被解释	机器语言
软硬件分界面			
1	传统机器语言机器级		微程序指令
0	微程序机器级		硬件执行

翻译：高级程序整个变换成低级机器级上的等效程序

解释：用低级机器级上的一串语句或指令仿真高级机器上的一条语句或指令的功能，对高级机器级程序逐条指令或逐语句解释

翻译快，占用空间多

2. 透明性
   a) 客观存在的事物，从某个角度看不到
   b) 好处：简化设计
   c) 坏处：无法控制
   d) 汇编级：透明-主存地址寄存器，不透明-外设+寄存器
3. 计算机系统结构
   a) 狭义：软硬界面的确定

      数据表示、寻址方式、寄存器组织（数量、字长、使用等）、存储系统、指令系统、中断机构、机器级 I/O 结构、管态用户态、信息保护机构
   b) 广义：软硬件之间的功能分配及如何更合理地实现分配给硬件的功能
   c) 计算机组成
      i. 计算机系统结构的逻辑实现，包括机器级内部的数据流，控制流组成及其逻辑设计
      ii. 数据通路宽度、专用功能部件设置、各种操作对部件的共享程度、缓冲和排队技术、各功能并行度、预测技术、可靠性技术、控制机构组成方式
   d) 计算机实现：计算机组成的物理实现
   e) 举例

	系统结构	组成	实现
指令系统	确定指令系统	指令的实现	电路设计
乘法指令	是否有乘法指令	加法器实现还是乘法器实现	加法器（乘法器）的类型、价格、数量等
主存系统	主存容量与编址方式	主存速度，单体多字、多体单字等逻辑结构	器件选择、电路设计、组装技术

   f) 关系：
      i. 狭义系统结构是组成的抽象，组成是实现的抽象；一种体系结构可以有多种组成、一种组成可以有多种物理实现

ii. 不同系统结构影响组成技术
iii. 计算机组成对系统结构起推动作用
iv. 系统结构的设计要考虑到准备采用和可能采用的组成与实现技术，尽可能不对组成、实现技术的采用和发展加以限制，要为组成和实现发展留有余地。
v. 组成设计向上决定于系统结构，向下受限于实现技术，组成可与实现折中权衡

4. 软硬件在逻辑上等效：软件的功能可以用硬件完成，硬件的功能可以用软件模拟完成
5. 软硬件取舍原则
   a) 考虑现有硬件或器件，系统有较高的性价比
   b) 从硬件角度，考虑到准备采用和可能采用的组成与实现技术，不过多对组成、实现技术的采用加以限制。
   c) 从软件角度考虑，把为编译、OS 以及高级语言的设计与实现提供更多、更好的硬件支持放在首位。
6. 计算机性能：系统速度，用响应时间衡量：从任务送入计算机处理到得到结果
   a) CPI：平均每条指令时钟周期数
   b) MIPS：计算机单位时间执行的指令条数（百万条指令数/秒）
   c) MFLOPS：每秒百万次浮点运算
7. 计算机定量设计原理
   a) 哈夫曼压缩原理：尽可能加速高概率事件比加速低概率事件对性能提升要显著
   b) Amdahl 定律：确定对系统中性能瓶颈部件采取措施提高速度后，能得到系统性能改进的程度，即系统加速比 S_p.
      性能可改进比 f_{new}:可改进部分占总时间比
      部件加速比 r_{new}：可改进部分，改进后提高比值

$$S_p = \frac{T_{old}}{T_{new}} = \frac{T_{old}}{(1 - f_{new})T_{old} + f_{new}T_{old}/r_{new}}$$

   c) 程序访问的局部性原理
      i. 时间上：现在使用的信息，不久后可能还要使用，因为程序存在循环
      ii. 空间上：当前访问存储项的临近项很可能很快被访问，因为程序顺序存放，顺序执行，数据以向量阵列等形式聚簇存放
8. 计算机系统的设计方法
   a) 由上往下（专用机）
      i. 先考虑如何满足用户需求，定好面向应用的虚拟机器级的特性和工作环境然后逐级向下设计
      ii. 串行设计、设计周期长
      iii. 无法用于通用机的设计，通用机要求应用对象和环境不断改变
   b) 由下往上（通用机）
      i. 不考虑应用需求，
      ii. 硬件不改变情况下，设计软件来适应需求，软硬件脱节，软件得不到硬件支持。
   c) “从中间开始”向两边设计
      i. 从层次结构软硬界面开始

ii. 并行设计，周期短
iii. 软硬件设计人员可交流协调，交互式设计，软硬件分配更合理
iv. 性价比更高

9. 软件可移植性：软件不经修改或少量修改，就可以由一台机器移到另一台机器上运行，同一软件可以应用于不同环境。

技术：

a) 统一高级语言：面向题目和算法，和具体结构关系不大
一种完全通用的高级语言，为所有程序员使用，并能在完全不同的机器之间实现程序的软件移植。

b) 系列机、兼容机：具有相同结构的机器之间
在一定范围内的不同型号机器之间统一汇编语言
系列机：一个厂家生产的具有相同的体系结构，但具有不同组成和实现（不同档次）的计算机。（阻碍系统结构突破性进展）
兼容机：不同厂家生产的具有相同指令集结构的计算机
软件兼容：按指令集结构编制的机器语言程序及编译程序可以通用于各档机器
向上（下）兼容：程序不加修改就能在更高（低）档机器上运行（性能和速度）
向前（后）兼容：程序不加修改就能在之前（后）投入市场的机器上运行（时间）
系列机软件兼容：保证向后兼容，力争向上兼容

c) 模拟与仿真：软件在不同体系结构之间的移植
i. 目标：在一种机器的系统结构上实现另一种机器的系统结构，即在一种机器的系统结构上实现另一种机器的指令系统，即机器语言。
ii. 模拟：用 A 机（宿主机）机器语言程序解释 B（虚拟机）的机器语言程序
iii. 仿真：用 A（宿主机）的微程序解释 B（目标机）的机器语言程序

	模拟	仿真
主要区别：解释用的语言	模拟程序	仿真微程序
存放介质	主存	控制存储器
实现	复杂	简单
速度	慢	快
灵活性	大	小
适用范围	实时性要求不高	机器结构差异不大
解释次数	双重解释	单重，两套解释程序

10. 软件、应用、器件对系统结构的影响

a) 软件是促使计算机系统结构发展的最重要的因素
b) 应用需求是促使计算机系统结构发展的最根本的动力，不同的应用对计算机系统结构的设计提出了不同的要求
c) 器件是促使计算机系统结构发展的最活跃的因素。器件速度、集成度、可靠性等的提高促进了系统结构的发展
器件集成度、速度——机器速度
器件可靠性——流水技术
半导体存储器——cache、虚拟存储器
器件性价比提高——组成技术下移

11. 并行性：可以同时进行运算或操作
    a) 同时性：两个或多个事件同一时刻发生
    b) 并发性：两个或多个事件同一时间间隔发生
    c) 等级划分（低到高）：
        i. 执行程序：指令内部、指令之间、任务或进程之间、作业或程序之间
        ii. 处理信息：位串字串、位并字串（同字的全部位）、位片串字并（多字同位）、全并行
        iii. 信息加工各步骤和阶段：存储器操作并行、处理机操作步骤并行、处理器操作并行，指令、任务、作业并行
    d) 技术
        i. 时间重叠（时间上并行）：多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分
            流水线处理机（异构）
        ii. 资源重复（空间上并行）：重复设置硬件资源来提高可靠性或性能
            阵列处理机（同构）
        iii. 资源共享：利用软件的方法让多个用户按一定时间顺序轮流地使用同一套资源，以提高其利用率
            多机系统、分时操作系统、分布处理系统
12. 多机系统
    a) 多处理机系统：多台处理机组成单一计算机系统，逻辑上受统一操作系统控制
    b) 多计算机系统：多台独立计算机组成系统，各计算机受在逻辑上独立的操作系统控制
    c) 耦合度：反映多级系统物理连接的紧密性和交叉作用能力的强弱
        i. 最低耦合系统：脱机系统
        ii. 松散（间接）耦合系统：共享外围设备
        iii. 紧密（直接）耦合系统：共享主存
13. 计算机系统弗林分类法
        指令流：指令序列
        数据流：数据序列
        多倍性：系统性能瓶颈部件上处于同一执行阶段的指令或数据的最大可能个数。
    a) 多指令流单数据流（MISD）宏流水、脉动阵列
    b) 多指令流多数据流（MIMD）多机系统
    c) 单指令流单数据流（SISD）
    d) 单指令流多数据流（SIMD）阵列处理机、相联处理机
14. 3T 性能目标：1TFLOPS 运算能力、1TB 主存容量、1TB/s I/O 系统带宽

2. 数据表示、寻址方式、指令系统

1. 数据表示：可以由硬件直接识别和引用的数据类型。
   表现在有对这种类型的数据进行操作的指令和运算部件。
2. 数据类型：一组值的集合+作用于这个集合上的操作集
    a) 基本数据类型
    b) 结构数据类型：由相互有关的数据元素复合而成
    c) 访问指针

d) 抽象数据

3. 数据结构：应用中相互有关的数据元素的集合，结构数据类型的组织方式，反映用用中数据元素或信息单元之间的结构关系
   a) 通过软件映像，将信息变成机器中所具有的数据表示来实现
   b) 数据表示是数据结构的子集，数据表示位数据结构的实现提供支持，影响效率和方便性
   c) 数据表示的确定实质是软硬件的取舍
   d) 数据表示和数据结构的关系问题是软硬件界面的问题
4. 高级数据表示
   a) 自定义数据表示
      i. 标志符数据表示：为缩短高级语言与机器语言之间的语义差距，让机器中每个数据都带类型表示
         问题：
            数据字增设标志符，增加程序占用空间
            降低程序执行速度
      ii. 数据描述符：进一步减少标志符所占存储空间
            描述符和数据分开存放，表示要访问的数据是整块还是单个及地址信息
            不必每次访存取元素都访问描述符
      iii. 区别
         标志符与数据合存于同一个存储单元中，描述单个数据；
         数据描述符与数据分开存储，主要描述成块数据。
   b) 向量、数组数据表示——向量机、通用机
      为向量、数组数据结构的实现和快速运算提供硬件支持
      便于成块预取向量、数组，用一条指令对向量的全部元素进行运算
   c) 堆栈数据表示——堆栈机器
      堆栈机器：
         高速寄存器组横的硬件堆栈，与主存堆栈在逻辑上构成整体，使堆栈速度是寄存器级别，容量是主存级别
         丰富的堆栈指令
         有利于高级语言的编译
         程序总存储量短
         支持程序嵌套和递归调用，支持中断处理
5. 引入数据表示的原则
   a) 基本数据表示必不可少
   b) 数据表示的引入是否高了系统效率（时间空间）
   c) 数据表示的引入，是否提高通用性和利用率
   d) 基本数据表示也有可挖掘的细节
6. 浮点数（p+1 位阶码：阶符+p 位阶值；m 位尾数）
   a) p：影响表示范围（阶码都是二进制）
   b) m：影响精度
7. 浮点数尾数基值 r_m
   a) 机器中一个 r_m 的数位用$\lceil \log_2 r_m \rceil$个机器位来表示

b) 非负阶，正规格化尾数（规格化尾数：小数点后第一个 r_m 进制的数不是 0

$m' = m/\lceil \log_2 r_m \rceil$

可表示最小尾数值	r_m^{-1}	可表示最小数	r_m^{-1}
可表示最大尾数值	$1 - r_m^{-m'}$	可表示最大数	$(1 - r_m^{-m'})(2^p - 1)$
可表示最小阶	0	可表示尾数个数	$r_m^{m'} - r_m^{m'-1}$
可表示最大阶	$2^p - 1$	可表示数的个数	$(r_m^{m'} - r_m^{m'-1})2^p$

r_m 是 2 的整数次幂时，$r_m^{m'} = 2^m$

c) 尾数基址 r_m 增加：可表示数的范围增大，可表示数的个数增加，数轴上分布稀疏，精度下降右移机会小，精度损失机会少，运算速度增加

8. 浮点数尾数下溢处理方法
   a) 截断法：去掉尾数，总是产生负误差，平均误差最大
   b) 舍入法：增设附加位，0 舍 1 入
      优点：增加的硬件开销小，平均误差接近 0，误差最小
      缺点：处理速度慢
   c) 恒置“1”：最低位恒置“1”
      优点：实现简单，平均误差小，中高速机器
      缺点：最大误差最大
   d) 查表舍入法：尾数 k-1 位+准备舍弃位，查表得到最后 k-1 位
      下溢处理表内容：全“1”截断法，其余舍入法
      优点：平均误差可以调节到 0（人为调节），避免多次进位
      缺点：需要硬件配合，最花费硬件

   最省硬件、最快：截断法、恒置“1”

9. 寻址方式三种面向

	主要	少量访问	
面向主存的寻址	主存	寄存器	
面向寄存器的寻址	寄存器	主存和堆栈	不用访存
面向堆栈的寻址	堆栈	主存和寄存器	利于减轻对高级语言编译的负担，利于支持函数嵌套调用、递归。

10. 寻址方式在指令中指明方式
    a) 占用操作码位
    b) 在地址码部分设置寻址方式位字段

11. 定位技术
    a) 静态再定位：目标程序装入主存时，装入程序用软件方法把目标程序中的指令和数据进行修改，即把程序的逻辑地址改成实际物理地址
       程序执行过程中，物理地址不变
    b) 动态再定位：装入主存时，装入主存起始地址到基址寄存器，程序执行时，每取出一条指令借助重定位机构（地址加法器和基址寄存器）形成物理地址访存。
       指令执行时才形成物理地址

12. 基址寻址与变址寻址
    a) 基址寻址：支持逻辑地址到物理地址变换，以利于实现程序的动态再定位
       地址加法器形成物理地址，速度快
       配有程序越界的保护措施

b) 变址寻址：对向量、数组等数据块运算的支持，便于程序的循环

13. 物理主存中信息的存储分布
    a) 编址单位：字节
    b) 信息在存储器中按整数边界存储
       存放的地址是该信息宽度（字节数）的整数倍，避免信息跨主存边界
14. 指令系统的设计
    a) 指令系统是程序设计者看到的机器的主要属性，是软、硬件的主要界面
    b) 指令功能+指令格式
    c) 指令类型：特权型、非特权型
15. 指令格式优化：用最短位数表示指令的操作信息和地址信息，使程序中指令平均字长最短
    a) 指令：操作码+地址码
    b) 目标：存储空间占用少、规整
    c) 方式
       i. 扩展操作码
       ii. 指令字在存储器中按整数边界存储的前提下，使用变长指令字格式
       iii. 多种寻址方式，以便缩短地址码的长度
16. 操作码优化
    a) 三种编码方法

固定长度	规整性好	解码简单	空间大
哈夫曼编码	不好	复杂	小
扩展编码			

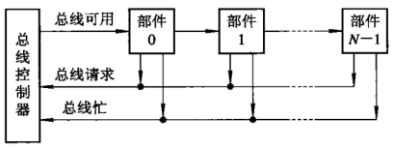
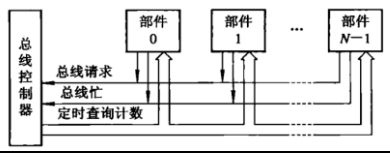
    b) 信息源熵：$H = -\sum p_i \log_2 p_i$
       信息冗余量：（实际平均码长-H）/实际平均码长
    c) 哈夫曼编码
       i. 短码不能是长码前缀
       ii. 缺点：
           不唯一（哈夫曼树、编码），平均码长不唯一
           不规整，硬件译码困难
           难以与地址码组成定长指令
17. 地址码个数，长度
18. 指令系统——软硬件功能分配
    a) CISC
       i. 面向目标程序优化改进
       ii. 面向高级语言的优化改进
       iii. 面向操作系统的优化
    b) RISC
       i. 按 RISC 一般原则设计的技术
       ii. 逻辑上，硬联和微程序固件结合
       iii. 设置数量较大的寄存器组，采用重叠寄存器窗口技术
       iv. 流水和延迟转移技术
       v. 优化编译系统设计的技术
    c) 超长指令字

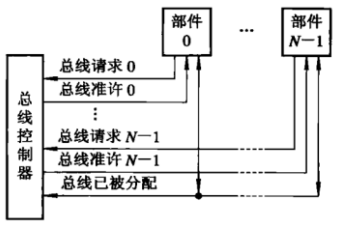
19. 重叠寄存器窗口
    a) 目标：缩短 CALL、RETURN 耗时
    b) 设置的大量的寄存器，分成多个组和全局区；每个组中分高、本地、低三个区；相邻组的高、低区重叠，加速参数与结果的传递，节省保护和恢复现场时间

3. 存储、中断、总线、I/O

1. 存储系统
    a) 要求：大容量、高速度、低价格
    b) 容量 S_w
    c) 存取速度：访问时间，存储周期，带宽
    d) 价格，每位价格
2. 并行主存系统：能并行读取多个 CPU 字的单体多字，多体单字或多体多字的交叉存储系统
    a) 单体多字：增加字宽
        i. 前提：指令和数据在主存中连续存放
    b) 多体单字：m 个地址不发生分体冲突
        i. 高位交叉：分体对应位地址低位一致
        ii. 低位交叉：分体高位一致
    c) 多体多字：结合单体多字，多体单字
3. 中断
    a) 中断源：引起中断的事件，可以来自机器外部，也可以来自机器内部
    b) 中断可以是软件引起也可以是硬件引起的，
    c) 中断请求：中断源向中断系统发出的请求中断的申请
    d) 中断响应：允许中断 CPU 现行程序，转去对该请求进行预处理，包括保存断点及现场，调出中断服务程序，准备运行（实现：交换新旧程序 PSW）
4. 中断分类
    a) 机器校验中断：设备故障，包括电源故障、运算电路出错、主存出错、通道动作故障，处理器硬件故障
    b) 访管中断：用户程序需要操作系统介入
    c) 程序性中断：
    d) 外部中断
    e) I/O 中断：I/O 操作完成、I/O 通道或设备故障
    f) 重新启动中断：CPU 不能禁止
5. 中断分级（优先级从高到低）
    a) 紧急机器校验
    b) 管理程序调用、程序性
    c) （可抑制的机器校验）
    d) 外部中断
    e) I/O 中断
    f) 重启
6. 中断响应次序
    a) 同时发生多个不同类中断请求时，中断响应硬件所决定的响应次序
    b) 硬件，固定

7. 中断处理次序：中断处理程序实际执行完的次序
   a) 增设中断级屏蔽位寄存器
   b) 软件
8. 中断过程软硬件功能分配：中断处理软件与中断响应硬件
   a) 必须硬件：保存断点，进入服务程序
   b) 必须软件：中断服务程序，断点返回
9. 总线系统分类
   a) 传输方向：单工、半双工、全双工
   b) 用法：
      i. 专用总线：只连接一对物理部件
      优点：
      多个部件可以同时收发信息
      不会争用总线，系统流量高
      控制简单
      总线失效只造成总线连接的两部件不能直接通信，还可以间接通信，系统可靠性高
      缺点：
      线数多，成本高，N 台 N*(N-1)/2 根
      不利于模块化
      总线利用率低
      ii. 非专用总线：多个部件共享，同时只能有一对设备使用总线
      优点：
      总线数目少
      总线接口标准化、模块化强
      可扩充能力强
      易用多重总线来提高总线的带宽和可靠性，使故障弱化。
      缺点：
      系统流量小，经常会出现争用，使未获得总线控制权的部件等待，降低效率
10. 总线控制方式（非专用总线，保证只有一个优先级高的申请者得到总线控制权）
    a) 分散式总线控制：控制逻辑分散到连接总线的各个部件
    b) 集中式总线控制：总线控制逻辑集中放到连接总线的某部件或单独设置的硬件设备中

串行链接方式	3	优先级固定	
定时查询方式	$2+\lceil \log_2 N \rceil$	可被程序改变	

独立请求方式	2N+1	可被程序改变	

i. 串行链接方式
ii. 定时查询方式
每次总线分配前，计数器清“0”：优先级同串行方式
每次总线分配前，计数器不清“0”：是一种循环优先级，各部件有相同的使用总线机会
每次分配前，计数器置初值：设置某个部件位最高优先级
每次分配前，重设部件号：指定各部件优先级
iii. 独立请求方式

11. 总线通信技术
a) 同步通信：统一时标同步
i. 信息传输速度高；不受总线长度影响
ii. 对同步信号要求高；部件速度差异大时，会影响效率；缺少正确接收回答
b) 异步通信：应答方式（不同速度的 I/O 共享总线）
i. 单向控制：只由一方控制
单向源控：源部件把数据放到总线上，延迟一段时间发“数据请求信号”，作为目的部件接收数据选通信号
单向目控：目的部件发“数据请求”使源部件放数据到总线，目的部件校验后决定发“数据出错”还是“数据请求”
传送速率随距离增加而降低
ii. 请求/回答双向控制：
最广泛：异步双向互锁通信方式

12. 数据宽度
a) 数据宽度：I/O 设备取得总线控制权后传送的数据总量
b) 数据通路宽度：数据传送的物理宽度，一个时钟周期传送的信息量
c) 种类
i. 单字（单字节）：打印机（低速设备
ii. 定长块：磁盘（高速
iii. 可变长块：高优先级，中高速
iv. 单字加定长块：低速，优先级高
v. 单字加可变长块：最快，费用最高

13. 总线线数
a) 线数越多，成本越高，干扰越大，可靠性越差，占空间越大，传送速度越快
b) 总线越长，成本越高，干扰越大，可靠性越差

14. 输入输出系统
a) 组成：输入输出设备、设备控制器与输入输出有关的软硬件
b) 输入输出系统硬件对应用程序员透明，面向操作系统
c) 工作方式
i. 程序控制方式（全软件、程序查询方式、中断方式）

查询方式：I/O 受 CPU 控制，并行度低
中断方式：

ii. DMA 方式：传送过程不需要 CPU 干预
iii. I/O 处理机方式（通道、外围处理机）

15. 通道处理机
    a) 用户在目态无权使用 I/O 指令，需要访管
    b) 一次 I/O，两次访管
    c) 通道系统：中央处理机 CPU/主存—通道—设备控制器—外设
    d) 种类

设备选择时间 T_S
传送一个字节时间 T_D
P:连接在通道上同时工作设备数
n：每个设备传送字节数
k:数据块字节数

字节多路通道	$T=(T_S+T_D)*P*n$	单字节	字符类低速设备	字节交叉方式
数组多路通道	$T=\left(\frac{T_S}{k}+T_D\right)*P*n$	定长数据块	高速，磁盘	成组交叉方式
选择通道	$T=\left(\frac{T_S}{n}+T_D\right)*P*n$	变长数据块	高优先级，高速	设备独占通道

    e) 通道流量：单位时间传送字节数
        i. 通道极限流量
        ii. $f_{max}=P*n/T$
        iii. 通道实际流量（不大于最大流量）

字节多路通道：$f_{byte}=\sum f_i$
数组多路通道：$f_{block}=\max\{f_i\}$
选择通道：$f_{select}=\max\{f_i\}$
I/O 系统总流量（极限流量）$f=f_{byte}+f_{block}+f_{select}$

4. 存储体系

1. 存储体系：构成存储系统的不同存储器之间配上辅助软、硬件，使从应用程序员看来，是一个逻辑整体
    a) 主存容量：虚拟存储器
    b) 主存速度：寄存器、多体交叉并行存取、cache
    c) 基本二级存储体系
        i. 虚拟存储器：速度是主存的，容量是辅存的（主要软件实现），对应用程序员透明，对系统程序员不透明
        ii. Cache：速度是 cache，容量是主存（主要硬件实现），对应用程序员和系统程序员透明
    d) 构成依据——局部性
    e) 虚拟存储器只用于多用户环境，要访问的信息不在主存，申请访问的程序暂停执行或被挂起
    f) Cache：多用户或单用户，要访问的信息不在 cache，暂停执行

g) 性能参数：

命中率H

等效访问时间$T_A = HT_{A_1} + (1-H)T_{A_2}$

访问效率$e = T_{A_1}/T_A$

2. 虚拟存储器管理方式（增加地址映像表机构
   a) 段式管理
      i. 段表，常驻内存，可以在辅存
         段名、地址、装入位、段长、访问方式
      ii. 段表基址寄存器
      iii. 特点：
         便于程序模块化
         便于多程序共用主存中的数据和程序
         容易以段为单位实现存储保护
         缺点：
         地址变换所用时间长，两次加法
         主存利用率低
   b) 页式管理
      i. 页表特点
         （虚页号）、实页号、装入位、访问方式
         页表项简单，速度快
         页表过大效率低
      ii. 优点
         内存利用率高
         页表相对简单
         地址变换速度快
         易管理磁盘
      iii. 缺点
         模块化不好
         页表长，占空间
      iv. 浪费空间大（装入位为“0”多）
         页表中装入位为“0”的行，用实页号字段存放此虚页辅存中的实地址
         相联目录表法：页表压缩成只存放装入主存的页
         （相联存储器-按内容访问）
   c) 段页式管理
      i. 主存分页，程序分段，段再分页
3. 页式虚拟存储器
   a) 虚地址：$N_s = u + N'_v + N_r$
      实地址：$n_p = n_v + n_r$
      页内地址 n_r 直接由 N_r 得到
   b) 地址映像：建立虚地址与实地址之间的对应关系
      地址变换：虚地址如何变换为实地址

c) 实页冲突（争用）：先装入一个，剩下等待
d) 页式管理方法：
全相联映像（虚存远大于实存）：冲突概率最低
e) 地址变换：页表法（地址映像表）

4. 页面替换算法
a) 目标：命中率，便于实现（软硬件成本）
b) 随机算法
c) 先进先出（FIFO）
d) 近期最少使用（LRU）最近最久未被使用
实现：随机期法，历史位法
修改位
e) 优化替换算法（OPT）理想化最优

5. 影响命中率
a) 虚拟存储器
i. 替换算法
ii. 地址流
iii. 主存页数
堆栈替换型算法：页数越多，命中率可能会越高（不会降低），LRU，OPT
在任何时刻 t，n 个实页中的虚页集合总是 n+1 个实页时集合的子集
其他如 FIFO，可能低也可能高
iv. 主存容量
v. 调页策略
b) Cache
i. 块大小：不出现先增后减，但块过大导致 CPU 空等时间长
ii. 块数目（cache 容量）
iii. 组相联组数
iv. 替换算法
v. 页地址流

6. 页式虚拟存储器实现中的问题
a) 页面失效
i. 一般中断在指令末尾检测中断，页面失效式故障应该立刻响应处理
ii. 原因：
指令、操作数、字符串跨页
间接寻址跨页
iii. 后援寄存器法：保存全部现场
预判技术：预判是否都在主存，不在则调页后执行
iv. 替换算法：减少颠簸现象
v. 页面大小：页面过大，页数少，页面失效概率高
页面过小，跨页概率高
b) 提高虚存等效访问速度
i. 提高命中率
ii. 加快访存速度
小容量快速存储器或寄存器组存页表

快慢表（相联存储器，容量大，查找速度慢）
大容量快表（散列函数，相联访问变成按地址访问

7. Cache
   a) 映像规则：数据块可放位置
      实现难易，地址变换速度，主存利用率高，冲突概率小
      i. 全相联映像：主存任意一块可以映射到 cache 任意一块
         相联目录表法
         冲突概率低，查表速度慢
      ii. 直接相联映像：主存按 cache 大小分区，每区各块按位置对应 cache 各块，主存第 i 块唯一映像到 i mod 2^{ncb} 块 cache
         硬件实现简单、快速；冲突概率高，空间利用率低
      iii. 组相联映像：主存和 cache 分组，主存分区，组与组之间直接映像，组内全相联映像（主存地址=区号：组号：组内块号：块内位移）
      iv. 段相联映像：组间全相联，组内直接相联
   b) 地址变换：找到块位置
   c) 替换算法：
      i. FIFO
      ii. LRU
      iii. 实现方式：
         堆栈法：硬件堆栈法、寄存器法
         比较对法：触发器
   d) 写策略：写访问操作
      CPU 写 cache，没有立即写主存
      IO 写主存，cache 没有更新
      i. 写直达法：写 cache 时，写主存
      ii. 标志交换法（写回法）：只写 cache，替换回主存时写回
      iii. 写不命中
         不按写分配算法：只写主存——写直达法
         按写分配算法：写主存，块调入 cache——写回法

5. 标量处理机（重叠流水）

1. 重叠
   a) 在解释第 k 条指令的操作完成之前，就可以开始解释第 k+1 条指令
   b) 不能加快一条指令的实现
   c) 要求
      i. 访存冲突（取指访存，取操作数也可能访存，内存访问冲突）
         操作数、指令分开存放
         多体交叉结构（不在同体）
         指令缓冲寄存器，预取指令
      ii. 分析与执行并行：独立的硬件部件
      iii. 分析与执行操作控制上的同步：一次重叠（只有相邻两条指令重叠解释）
   d) 相关
      i. 转移指令：第 k 条指令是条件转移指令
         延迟转移：编译程序生成目标程序时，将转移指令与与条件转移无关

的第 k-1 条指令交换位置
猜测法
加快和提前形成条件码
加快短循环程序处理

ii. 数相关：k、k+1 条指令的数据地址之间有关联（主存空间，通用寄存器组）
主存空间数相关：推后读（分析 $_{k+1}$），到 k 的执行结束
通用寄存器组相关：
推后读：推后“分析 $_{k+1}$”到“执行 $_{k}$”结束或写入寄存器后
增设“相关专用通路”：直接将运算结果送到运算器供 k+1 条指令用

iii. 指令相关：第 k 条指令执行结果形成第 k+1 条指令
程序运行过程中，不允许修改指令
设置执行指令，将指令相关转化为数相关

e) 寄存器组基址（变址）相关
分析前半段取得基址，运算结果在执行末尾送入寄存器组
可能存在一次相关和二次相关

2. 并行性
a) 空间并行性：多个独立的操作部件
b) 时间并行性：流水线

3. 重叠与流水
a) 一次重叠：只是把一条指令的解释分解成两个子过程
b) 流水线：是重叠的延伸，将指令的解释分解成更多的子过程

4. 流水：同时解释多条指令
a) 表示方法：连接图、时空图、预约表
b) 流水线扩展
i. 向上：多个处理机流水
ii. 向下：流水线各段进一步细分
iii. 子过程细分需增设锁存器而减慢速度，增加用时

c) 分类
i. 级别：部件级（子部件流水），处理机级（各部件流水），系统级（处理机）
ii. 功能：
单功能流水线
多功能流水线（多种连接方式，可以实现不同功能）
连接方式：静态多功能流水线，动态多功能流水线
同一段时间，各段能否按不同方式连接
iii. 按机器具有的数据表示：标量流水、向量流水
iv. 是否有反馈回路：线性，非线性
v. 控制方式：同步、异步

d) 性能
i. 吞吐率TP：单位时间能流出的任务数或结果数
流水线最大吞吐率：$TP = \frac{1}{\max\{\Delta t_i\}}$
实际吞吐率：任务量/总时间
耗时最长——瓶颈子过程

细分子过程；设置多套部件交叉并行

ii. 加速比S_p：流水线速度与等效的非流水线速度之比

iii. 效率η：流水线的设备实际使用时间占整个运行时间之比，也称流水线设备时间利用率

e) 相关处理：相邻两条指令不能同时解释

i. 局部性相关：指令相关、访存操作数相关、通用寄存器组相关（基本块内）
推后读、设置专用通路

ii. 全局性相关：转移指令和后续指令的相关（基本块间相关）

f) 中断处理（断点现场保护和恢复）

g) 非线性流水线调度——二维预约表

i. 连接图可以唯一表示线性流水线，不能唯一表示非线性流水线

ii. 启动距离：向非线性流水线输入端输入两个任务之间的间隔
禁止启动距离：引起流水段冲突的启动距离

iii. 禁止表：每一行任意两个“x”之间的距离的集合（去重）

iv. 冲突向量：N-1 位的向量表示和普及任务间隔各拍是否发生冲突
初始冲突向量：流水线开始工作的冲突向量

v. 状态转移图
当前冲突向量：第一个任务冲突向量按间隔拍数向右移位与第二个任务初始冲突向量按位或

vi. 启动循环：使任一流水段在任一时钟周期都不冲突的循环数列
恒定循环：只有一个启动距离
平均启动距离：循环数列平均数
最小平均启动距离：（3，4）更优，（4，3）