

第一章 计算机系统结构导论

- 【简答题】计算机系统结构定义，它和组成和实现三者关系:P9

计算机系统结构是程序设计者所看到的计算机属性，即概念性结构和功能特性。计算机组成是计算机系统结构的逻辑实现，计算机实现是计算机组成的物理实现。

- 计算机系统的功能层级：P7
 - 应用软件
 - 系统软件：高级语言、汇编语言、操作系统
 - 指令系统：机器语言程序员
- 软件兼容：指同一个软件可以不加修改地运行于系统结构相同的各机器上，所得结果一致。P11
 - 必须向后兼容，尽量向上兼容。
 - 向上兼容：低档机器的目标程序不加修改就可以运行于高档机器。
 - 向下兼容：高档机器的目标程序不加修改就可以运行于低档机器。
 - 向后兼容：新系统能运行旧软件。
- 模拟与仿真：P11
 - 概念：
 - 模拟：用机器语言程序解释，实现程序移植。
 - 仿真：用微程序直接解释另一种机器的指令系统。
 - 区别：
 - 仿真用微程序解释，更快，其解释程序在微程序存储器。
 - 模拟用机器语言程序解释，更慢，解释程序在主存。
- 冯·诺依曼型计算机：P21
 - 组成：由运算器、控制器、存储器和输入/输出设备组成。
 - 特点：
 - 存储程序顺序执行。
 - 以运算器、控制器为中心。
 - 固定的存储器、顺序线性编址的一维结构。
 - 指令由操作码和地址码构成；在存储器中顺序存放；指令和数据被同等对待；指令和数据二进制编码。
- Amdahl定律：系统中某部件采用某种更快的执行方法后，整个系统性能的提高与这种执行方法的使用频率或占总执行时间的比例有关。(公式：改进前后整个系统加速比)P13

$$S_n = \frac{1}{(1 - F_e) + \frac{F_e}{S_e}}$$

S_n 整个系统加速比， F_e 改进部分在整个系统占比， S_e 改进部分的局部加速比。

第二章 处理器及其相关技术

- 哈夫曼编码作用：压缩指令操作码。P26
- 并行的三种方式：指令级并行、线程级并行和数据并行。
- 【简答】Intel环形总线好处：P35
 - 各模块交互方便，不用通过CPU内核中转。
 - 双环设计保证任意两个环形接入点之间的距离不超过环形接入点总数一半，延迟较低。
 - 高速环形总线保证了高性能。
 - 方便扩展CPU内核。
- GPU：P47
 - 流多处理器：
 - GPU中基本运算单元：流处理器(Stream Processor, SP)。
 - SM流多处理器是GPU执行指令的基本单位，同一SM内的SP具有共享的高速一级缓存。
 - CPU与GPU的混合调度：
 - GPU适合处理计算粒度大，通信量较小，数据并行度较高的任务。
 - GPU的存储层次：P48
 - 寄存器：位于SM中，存放线程执行变量，有线程私有性质。
 - 共享内存：位于SM中，访问速度仅次于寄存器，由一个SM中所有线程共享。
 - 全局内存：位于GPU片外存储体，即显存。
- cache一致性问题：不能用写回法（仅在被替换时才将修改过的cache写回主存），要用写直达法（每次写cache时，同步写入主存）。
- RAID（独立磁盘冗余阵列）：通过数据备份、分布式存储方式将数据写入多个磁盘。

第三章 存储系统结构

- 地址映像方式：段式、页式、段页式。
 - 组相联映像：**各组间直接映像，组内是全相联映像。**
 - 全相联映像：任何虚存页均可映像到实存的任何页面。
 - 直接相联映像：每个虚存页只能映像到实存的一个页面。
- cache透明性：cache的地址变换和块替换算法均由硬件实现，故cache-主存层次对系统程序员和用户都是透明的。P87
- 【简答】哪些是堆栈型替换算法，哪些不是，说明理由：P74

堆栈性质：也就是当分配给算法的资源增加时，命中率不会下降的算法。也就是对于n个资源数，算法在n+1个资源下的访问集合，完全包含在n个资源下的访问集合。LRU,OPT均是堆栈算法，FIFO不是堆栈算法。
- cache性能计算：P92
 - 数据cache的真实不命中率要根据实际数据访问次数计算，而非总指令数。
 - 平均存储器访问时间=指令所占比例*（命中时间+指令不命中率*不命中代价）+数据所占比例*（命中时间+数据不命中率*不命中代价）
 - CPU时间=指令数*（每条指令执行的时钟周期数+不命中率*存储器存取次数/指令数*不命中代价）*时钟周期时间
 - 【大题】多级cache性能计算：P92，看下3-1，3-2，3-3，3-5

■ 核心公式：

- 每次访问平均存储器访问时间=命中时间+不命中率*不命中代价
- 平均每条指令停顿周期数=（平均存储器访问时间-命中时间）*每条指令访问次数
- 平均存储器访问时间=L1命中时间+L1不命中率*(L2命中代价+L2不命中率*L2不命中代价)
- 平均每条指令的存储器停顿周期数=L1平均每条指令的不命中次数*L2命中时间+L2平均每条指令不命中次数*L2不命中代价

第四章 流水线结构

• 【简答】流水线技术特点：

- 将指令执行划分为若干功能段，每个功能段由专用功能部件实现。
- 每个功能段实现尽可能相等，避免因不等产生处理瓶颈。
- 形成流水线需要一段“通过时间”，在此过后流水过程才稳定。
- 指令流不顺序执行时，会断流，再形成流水需要一段时间，不应经常断流。
- 流水线技术适用于大量重复的程序过程，只有输入端能连续提供服务，流水线效率才高。

【判断】指令流水线不减少单次指令执行时间。【T】

• 流水线分类：

◦ 按级别分类：

- 部件级流水线（运算器流水线）
- 处理机级流水线（指令流水线）

◦ 连接方式分类：

- 静态流水线：同一时间段，流水线各段只能按同一种连接方式工作。
- 动态流水线：同一时间段，流水线各段按不同运算的连接方式工作，控制更复杂。

- 流水线中为避免时间段不等引起干扰，保证各段之间数据通路宽度匹配，各段之间增加**锁存器** (Latch)，由一个同一的时钟同步，采用集中控制方式。

• 流水线性能指标：

◦ 吞吐率：

$$TP_{\max} = \frac{\text{任务数}}{\text{执行时间}}$$

最大吞吐量取决于流水线中最慢的功能段，也称为**瓶颈**。

设流水线由m段组成，要完成n个任务，所需时间为T，实际吞吐率为：

$$TP = \frac{n}{T} = \frac{n}{n \times \Delta t_0 + (m-1) \times \Delta t_0} = \frac{TP_{\max}}{1 + \frac{m-1}{n}}$$

◦ 加速比：

$$S = \frac{T_{\text{非流水}}}{T_{\text{流水}}} = \frac{mn\Delta t_0}{n \times \Delta t_0 + (m-1) \times \Delta t_0} = \frac{mn}{m+n-1} = \frac{m}{1 + \frac{m-1}{n}}$$

◦ 效率：

$$E = \frac{\text{任务占用的时空区}}{\text{整个矩形时空区}}$$

- 超标量处理机和超流水线处理机：P143
 - 超标量处理机 $T(m,1)$ ：K表流水线级数

$$T(m,1) = (K + \frac{N-m}{m}) \times \Delta t$$

$(m,1)$ (表示同时发射m条指令，一个时钟周期的并行度)超标量处理机的加速比：

$$S(m,1) = \frac{T(1,1)}{T(m,1)} = \frac{m(K+N-1)}{N+m(K-1)}$$

同时发射的指令条数为m。

- 超流水线处理机 $T(1,n)$ ：
ILP(Instruction Level Parallelism)指令级并行度

$$T(1,n) = (K + \frac{N-1}{n}) \times \Delta t$$

加速比：

$$S(1,n) = \frac{T(1,1)}{T(1,n)} = \frac{n(K+N-1)}{nK+N-1}$$

指令并行度为n，每 $\frac{1}{n}$ 个周期发射一条指令，同时发射的指令条数为1。

- 【大题】非线性流水线：P148习题3
 - 求禁止向量，冲突向量，画状态图，求所有简单循环，求最小平均等待时间。
- 数据相关：P148习题4
 - 写写（不易判断），读写，写读。
- 流水相关处理：
 - 标量流水线中出现跳转指令的处理方式：照常执行，跳转时钟周期后改为idle，直到跳转位置。

第五章 并行处理机与多处理机系统

- 并行性：
 - 定义：同一时刻或同一时间间隔内完成两种或两种以上性质相同或不同的工作，只要在时间上重叠，均存在并行性。
 - 并行性从执行角度的等级从低到高划分：
 - 指令内部并行（运算部件流水线）
 - 指令间并行（指令级流水线）
 - 任务级或过程级并行
 - 作业级或程序级并行
 - 【简答】提高并行性的三种措施：
 - 时间重叠（流水线技术）
 - 资源重复（多核CPU,GPU,超标量处理机）
 - 资源共享（如多道程序、分时系统）
- 【简答】并行处理机基本结构：P151
 - 按弗林(Flynn)分类法：SIMD类型。

- 基本结构：用一个控制器控制多个处理单元构成的阵列，也称阵列处理机。在单一CU控制下，阵列内各个PE对各自所分配的不同数据并行执行**同一条指令**规定的操作。(CU, Control Unit), (PE, Processing Element)
- SIMD计算机的两种基本结构：分布式、共享式存储器结构
- 单级互连函数：
 - 恒等(Identity)：一一对应互连
 - 交换(Exchange)：第0位取反
 - 方体(Cube)：沿着x,y,z方向，第0, 1, 2位取反
 - PM2I：加减 2^i 再取模
 - 混洗(Shuffle)：将最高位换到最低位。除了地址码全为0和1外，其余节点都有与其他节点连接的机会。
 - 蝶式(Butterfly)：最高位与最低位互换。
- 多级立方体互连网络：Cube方式，
- 多级混洗交换网络（Omega网络）：
 - 每一级采用全混选方式

第六章 集群、网格和云计算

- 【简答】集群定义：通过**高速互连网络**把通用计算机连接起来，采用消息传递机制(MPI等)，向用户提供**单一并行编程环境和计算资源**。

编程题

- openmp常见函数API：

```

1  # pragma omp parallel for num_threads(4) // 指定并行执行时使用的线程数
2
3  # pragma omp critical{} // 对{}代码所有线程串行执行
4
5  omp_get_thread_num // 返回线程号
6
7  omp_set_num_threads // 设置后续并行域中的线程个数
8
9  omp_get_num_threads // 返回当前并行域中的线程数
10
11 omp_get_num_procs // 返回系统中处理器的个数

```

- 使用OPENMP，实现两个向量点积运算：编译运行时加上-fopenmp

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <omp.h>
4
5  int main() {
6      float a[100], b[100], result;
7      int n = 100;
8      result = 0.0;
9      for(int i = 0; i < n ; i ++ ) {
10         a[i] = rand()%5;
11         b[i] = rand()%9;

```

```

12     }
13     /*# pragma omp parallel for
14     for(int i = 0; i < n; i ) {
15         # pragma omp critical // 同一时间只有一个线程执行被保护的代码块
16         {
17             result += a[i] * b[i];
18         }
19
20     }*/
21
22     # pragma omp parallel for reduction(+:result) // 或者使用归约方法
23     for(int i = 0; i < n; i ++ ) {
24         result += a[i] * b[i];
25     }
26     printf("%f\n",result);
27     return 0;
28 }

```

- MPI常见API函数:

```

1  MPI_init(int* argc, char** argv[])
2  // 第一个被调用的MPI函数，作用是初始化并行环境，main函数必须带参数。
3
4  MPI_Comm_size(MPI_Comm comm, int *size)
5  // 获得通信域comm中规定的group包含的进程的数量。
6
7  MPI_Comm_rank(MPI_Comm comm, int* rank)
8  // 获取本进程在通信空间中的rank值。
9
10 MPI_Finalize(void)
11 // 退出MPI系统，释放占用的资源。
12
13 MPI_Send(buf, count, datatype, dest, tag, comm)
14 // 将从buf开始的count个数据发送给进程编号为dest的进程。
15
16 MPI_Recv(buf, count, datatype, source, tag, comm, status)
17 // 将接收到的数据保存在buf里。
18
19 MPI_Barrier(comm)
20 // 同步所有进程，阻塞通信域中所有调用本函数的进程，直到所有调用者都调用了它。
21
22 MPI_Reduce(sendbuf, recvbuf, count, datatype, op, rank, comm)
23 // 所有进程对从sendbuf开始的count个元素做op运算，并依次存放在进程rank上recvbuf开始的缓冲区。
24
25 MPI_Gather()
26 // 收集所有进程数据到rank
27
28 MPI_Scatter()
29 // 将进程rank上数据散发给所有进程

```

- linpack测试:

- 全称: Linear Equations Package计算机系统性能测试线性方程程序包。
- 核心程序包:

- HPL(High Performance Linpack): 主测试程序。
- BLAS(Basic Linear Algebra Subprograms): 提供基础线性代数运算。
- MPI: 分布式内存系统的并行计算通信。
- OpenMP: 共享内存系统的并行加速。
- LAPACK(Linear Algebra Package): 高层线性代数库, 包含求解线性方程组的算法。

题型: 选择15分、判断10分、简答25分、(应用、编程(写个OPENMP程序))25分、综合25分(含一个论述题)

- 计算题:
 - 非线性流水线
 - 单级互连函数
 - 多级互连函数
 - cache性能计算
 - 流水线相关性处理

论述题:

- 计算机架构从单核、流水线到多核各自的特点, 适合于什么计算, 这种变化是否合理:
是硬件资源利用率和计算需求共同驱动的结果。未来向异构并行和专用架构继续发展。
 - 单核架构: 串行执行, CPI接近1, 性能受限于时钟频率。
 - 流水线架构: 将指令分为多阶段(取指、译码、执行等), 各阶段重复执行。适用标量计算。
 - 多核架构: 多个物理核心共享内存, 并行执行多个线程。