计算机组成与设计期末复习资料

（数据科学与计算机学院 3 班，2017 年 12 月）

期末课程设计要求：①自选课程知识体系范畴任一知识点，用硬件实现或软件模拟实现；②按论文格式要求完成课程论文 1 篇（包括选题背景、技术路线、实现过程、效果评价）；③准备第 18 周的课程设计交流（源程序及论文的电子版上传给学委，交论文打印稿）。

第一章　计算机概要与技术

1. 计算机的分类、划代和应用特性

早期：按功能（规模）、器件（电子管晶体管集成电路）、行业应用来划分

现代（后 PC 时代）：着重从应用特性来划分：桌面计算、服务器、嵌入式计算、个人移动设备、集群/仓库级计算机（Clusters/Warehouse Scale Computer，WSC，通过云计算实现，软件即服务，Software as a Service，SaaS）

理解行业应用的交叉渗透，计算机面临的瓶颈现状、变迁以及应用需求驱动的观点

2. 计算机设计的重要原则和思想（注意理解功能与性能的含义）

面向摩尔定律的设计

使用抽象简化的设计

大概率事件优先的原则

Amdahl 定律（性能改进递减规则）：**如果仅仅对计算任务中的一部分做性能改进，则改进得越多，所得到的总体性能的提升就越有限。**

重要推论：如果只针对整个任务的一部分进行改进和优化，那么所获得的加速比不超过 1/（1－可改进比例）

加速比：加快某部件执行速度所能获得的系统性能加速比，受限于该部件的执行时间占系统中总执行时间的百分比。

$$\text{加速比}=\frac{\text{系统性能}_{\text{改进后}}}{\text{系统性能}_{\text{改进前}}}=\frac{\text{总执行时间}_{\text{改进前}}}{\text{总执行时间}_{\text{改进后}}}$$

CPU 性能公式：执行一个程序所需的 **CPU 时间**

流水线、并行性以及预测对提高机器性能的意义

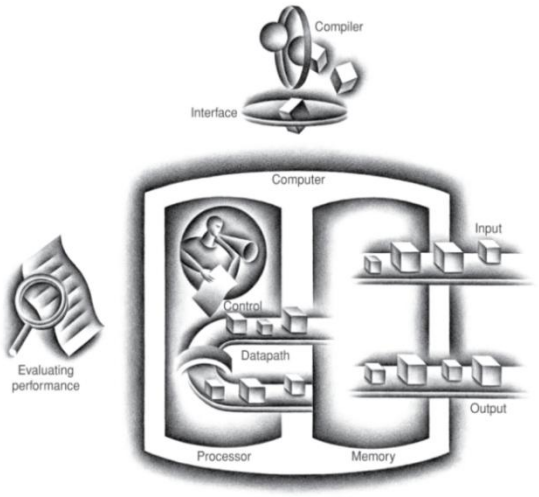
解决存储器容量、速度和价格三者矛盾的办法：引入存储器层次结构，理论依据：程序局部性（时间局部性和空间局部性）

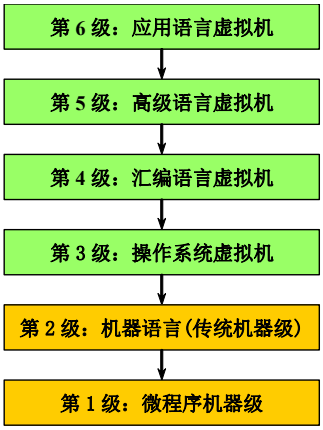
通过冗余提高可靠性（应对故障的主要方法）：冗余编码（提高信息传输的可靠性）、磁盘冗余阵（RAID(Redundant Array of Independent Disk）

3. 计算机系统组成：硬件设备+软件系统

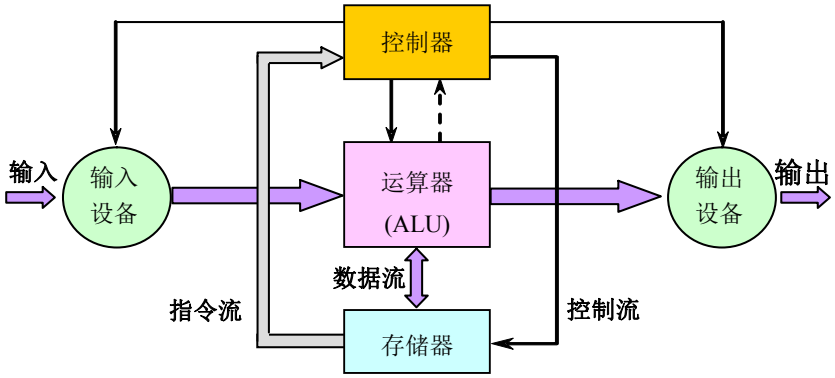
硬件设备：计算机系统中一切物理装置的总称。计算机发展的简史（需求驱动）、冯·诺依曼（John von Neumann, 1903-1957）的特点：按地址访问和顺序执行

计算机由运算器、存储器、控制器、输入设备和输入设备五大部件组成。





计算机系统层次结构



冯·诺依曼（John von Neumann, 1903-1957）结构

指令和数据以同等地位存入于存储器内，并可按地址寻访。

指令和数据均用二进制数表示。

指令由操作码和地址码组成，操作码用来表示操作的性质，地址码用来表示操作数在存储器中的位置。

指令在存储器内按顺序存放。通常，指令是顺序执行的，在特定条件下，可根据运算结果或根据设定的条件改变执行顺序。

机器以运算器为中心，输入输出设备与存储器间的数据传送通过运算器完成。

以运算器为中心→以存储器为中心，5 大部件→3 大部件，数据通路-分离式→总线式，单核 CPU→多核微处理器，依托 Cache 实现哈佛结构的存储体系。

软件：计算机系统中的程序、数据和文档（系统软件和应用软件）

系统软件（支撑环境）：操作系统、编译系统，解释和编译，高级语言到机器语言，以汇编语言为例：编辑（.txt）-编译（.obj）-链接（.com/exe）-运行

应用软件（工具软件和用户程序）

4. 计算机系统的功能和性能

功能：（Function），事物或方法所发挥的有利作用、效能。性能（performance）器物所具有的性质与效用。

从计算机用户或设计者的角度描述性能测量的度量指标和标准

响应时间（执行时间），计算机完成某任务的总时间

吞吐率（带宽），单位时间内完成的任务数量

性能度量公式：CPU 性能公式：**CPU 时间=指令数*CPI*时钟周期时间**

或 **CPU 时间=指令数*CPI/时钟频率**

指令数：执行某程序所需的指令数量；CPI：执行某个程序段时每条指令所需的时钟周期数；时钟周期时间：时钟频率的倒数

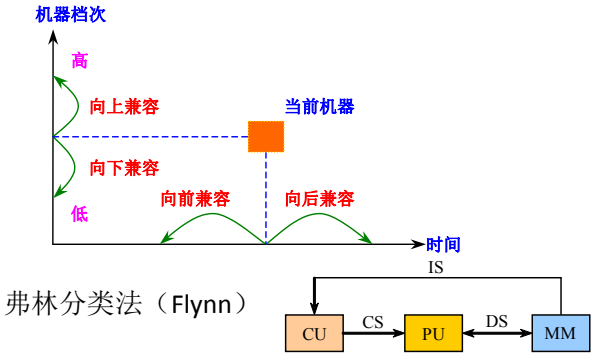
功耗墙：功耗（Power），功率的损耗，在单位时间中所消耗的能源的数量,单位为 W；能耗（Energy），能源转换实物量的损耗，单位是焦耳/秒

SPEC CPU 基准测试程序

事务处理（TP）性能测试基准程序：用于测试计算机在事务处理方面的能力，包括数据库访问和更新等。TPC(Transaction Processing Performance Council，事务处理性能委员会)，它的作用是制定商务应用基准程序(Benchmark)的标准规范、性能和价格度量，并管理测试结果的发布。

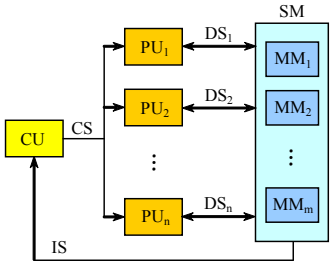
5. 例题与思考

兼容性：向上（下）兼容：按某档机器编制的程序，不加修改就能运行于比它高（低）档的机器。向前（后）兼容：按某个时期投入市场的某种型号机 器编制的程序，不加修改地就能运行于在它之前（后）投入市场的机器。系列机（向后兼容）、模拟和仿真、虚拟机、计算机性能随时间下移、透明性、计算机系统逻辑功能等效性、数据存放方式（大端和小端方式）、堆栈及其地址增长方式（后进先出，向上增长与向下增长）、计算机系统结构的生命周期（三个 5-7 年的划分）、第五代计算机（计算机分类的变化和划代的瓶颈）

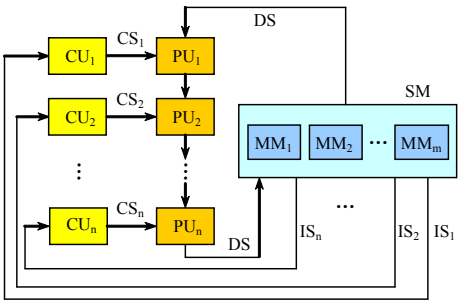


弗林分类法（Flynn）

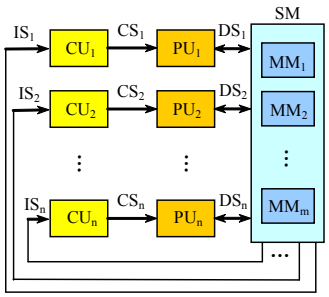
（a）SISD 计算机



（b）SIMD 计算机



（c）MISD 计算机



（d）MIMD 计算机

IS：指令流，DS：数据流，CS：控制流

CU：控制部件，PU：处理部件，MM 和 SM：存储器。

第二章　计算机的语言

1. 数据表示

数制：进位计数制（基数、位权）、数制转换（除基取余，乘基取整）

码制：机器数（符号数字化）、机器数中真值 0 的表示

原码、反码、补码和移码（0 的表示唯一）

ASCII 编码（字符 0，A，a 的编码）、汉字编码 GB2312-80（区位码-内码）、Unicode。

有符号数和无符号数（表示范围）、位、字节、字长、半字、双字的概念

2. 指令系统（指令集）

指令的表示：指令格式（操作码+操作数（或操作数地址）），指令操作（数据传送、算逻运算、分支与跳转等）

指令集结构分为堆栈、累加器和通用寄存器结构，根据数据来源又分为 RR 和 RM 型

指令格式设计(完整性、规整性、高效率、兼容性)、统一编址和独立编址（x86）

定长编码和变长编码及其优化编码（哈夫曼编码）

寻址方式（MIPS）：立即数寻址、寄存器寻址、基址寻址、PC 相对寻址、伪寻址

3. 实例讨论

①X86 指令集（CISC 结构、变长指令集）

②MIPS 指令集（RISC 结构，定长指令集）

③ARMV7(32 位)指令集（RISC 结构）

注意 ARMV7(32 位)指令集比较与 ARM 指令集的异同。

寻址方式（x86）：立即数寻址、寄存器寻址、直接寻址、寄存器间接寻址、存储器相对寻址、基址变址寻址、基址变址相对寻址、寄存器比例寻址（注意理解 x86 寻址方式中基址寻址和变址寻址的异同）

指令系统：数据传送、算逻运算、移位操作、串操作、控制转移、处理机控制

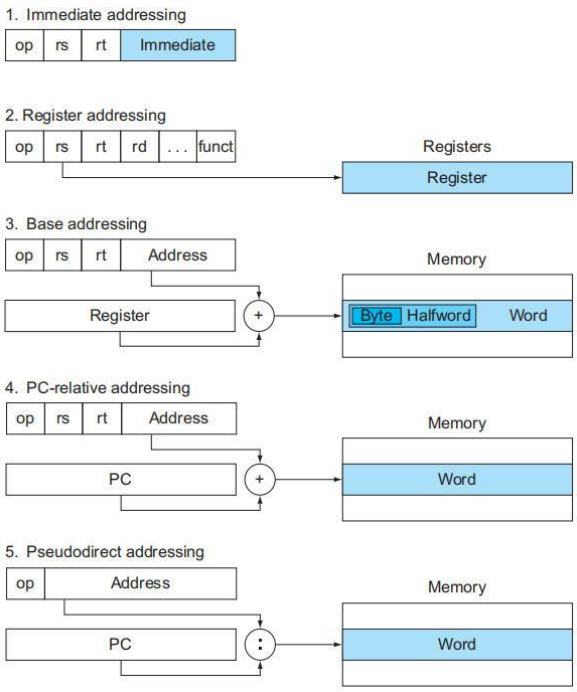
汇编程序设计

①汇编程序框架：COM 型和 EXE 型

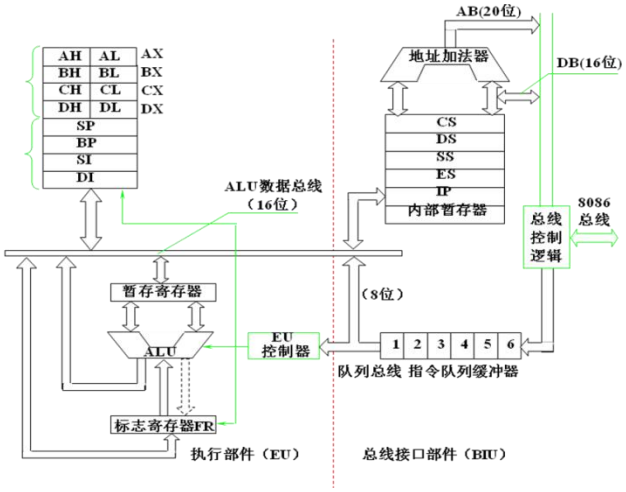
②汇编指令集

③常用中断调用和功能

④汇编程序设计



MIPS 寻址模式（立即数寻址、寄存器寻址、基址寻址、PC 相对寻址、伪寻址）



x86 CPU 内部结构框图

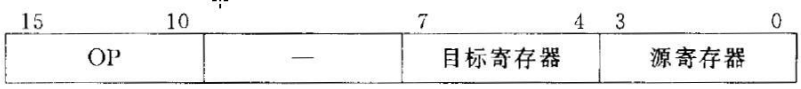
例题与思考

1. ASCII 码是 7 位，如果设计主存单元字长为 32 位，指令字长为 12 位，是否合理？为什么？

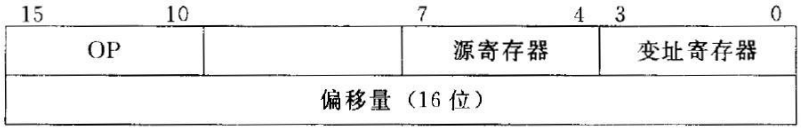
2. 假设某计算机指令长度为 20 位，具有双操作数、单操作数、无操作数三类指令形式，每个操作数地址规定用 6 位表示。问：

若操作码字段固定为 8 位，现已设计出 m 条双操作数指令，n 条无操作数指令，在此情况下，这台计算机最多可以设计出多少条单操作数指令？

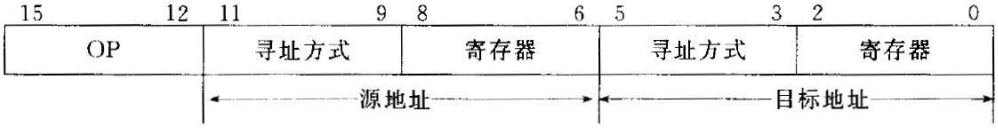
3. 指令格式结构如下所示，试分析指令格式及寻址方式特点。



4. 指令格式结构如下所示，试分析指令格式及寻址方式特点。



5. 指令格式结构如下所示，试分析指令格式寻址方式特点。



第三章　计算机的算术运算

1. 定点数的加减运算

溢出（上溢，超过最大表示范围；下溢，小于最小表示范围）

移位（算术移位、逻辑移位、循环移位），算术右移（最高位保持不变）

2. 定点数乘除运算流程图与结构框图

3. 浮点数的表示和运算

浮点数运算步骤：对阶（小阶看齐大阶）、尾数求和（差）、规格化、舍入、判溢出，移位时的向左规格化（规格化）和向右规格化（溢出校正）

4. IEEE754 标准及其转换

5. 浮点运算流程图与结构框图

6. 多功能算逻运算单元设计（ALU）

一位全加器-四位全加器-溢出判断（异或电路）-加减控制（求补电路）-先行进位（进位链设计）-函数发生器设计

7. 并行性和计算机算术：子字并行

子字并行（在一个宽字内部进行的并行操作）

例题与思考

例题：已知2[X]补＝1.0101001，1/2[Y] 原＝1.0101100，

⑴ 用变形补码计算：[X] 补＋[Y] 补，判断结果有无溢出。

⑵ 画出实现补码定点加减法（具有溢出判断）的硬件结构图。

【分析与解答】（1）[X]补＝1.1010100，$1/2[Y]_{补}$=1.1010100，[Y]补＝1.0101000

[X]补＋[Y]补＝11.1010100＋11.0101000＝10.1111100，溢出

（2）补码定点加减法硬件结构图（用加法器、求补电路、溢出判断构成框图则可）

4．在 IEEE754 标准中，一个规格化的 32 位浮点数 x 的真值表示为：

$$X = (-1)^{S} \times (1.M) \times 2^{E-127}$$

（1）$27/64=0.011011=1.1011\times 2^{-2}$

E= -2+127 = 125= 0111 1101　　S= 0　　M= 1011 0000 0000 0000 0000 000

最后表示为：0 01111101 10110000000000000000000

（2）$-27/64=-0.011011=1.1011\times 2^{-2}$

E= -2+127 = 125= 0111 1101　　S= 1　　M= 1011 0000 0000 0000 0000 000

最后表示为：1 01111101 10110000000000000000000

【例 2-14】 将十进制数-0.75 表示成单精度的 IEEE 754 标准代码。

解： -0.75 可表示成-3/4。即二进制的-0.11。在浮点表示法中为

-0.11×2^{0}

在 IEEE 754 标准中规格化数的表示法为

-1.1×2^{-1}

根据 IEEE 754 的单精度表示公式

$(-1)^{s}\times1.f\times2^{e-127}$

这个数可表示为

$(-1)^{1}\times(1+0.1000\ 0000\ 0000\ 0000\ 0000\ 000)\times2^{-1}$

$=(-1)^{1}\times(1+0.1000\ 0000\ 0000\ 0000\ 0000\ 000)\times2^{126-127}$

即符号位 s=1，阶码部分值 $e=126_{10}=01111110_{2}$，尾数部分 f= 1000 … 000。将其按照浮点数的编码格式表示，即为：

1 01111110 10000000000000000000000

【例 2-15】 将如下 IEEE 754 单精度浮点数用十进制数表示：

1 10000001 01000000000000000000000

解： 符号位 $s=1$，阶码部分值 $e=129$，尾数部分 $f=1/4=0.25$，根据 IEEE 754 标准的表示公式，它的数值为：

$(-1)^{1}\times(1+0.25)\times2^{129-127}$

$=-1\times1.25\times2^{2}$

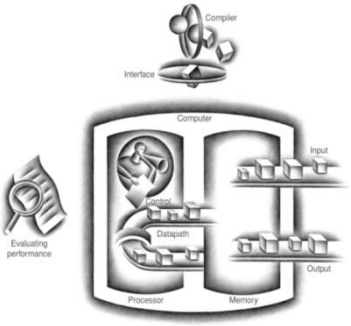
$=-1.25\times4$

$=-5.0$

第四章　处理器

1. 数据通路

2. 流水线技术



①基本原理：流水线把一个处理过程分解为若干个子过程（段），每个子过程由一个专门的功能部件来实现。流水线中各段的时间应尽可能相等，否则将引起流水线堵塞、断流。时间长的段将成为流水线的瓶颈。

流水线每一个功能部件的后面都要有一个缓冲寄存器（锁存器），称为流水寄存器。作用是在相邻的两段之间传送数据，以保证提供后面要用到的数据，并把各段的处理工作相互隔离。流水技术适合于大量重复的时序过程，只有在输入端不断地提供任务，才能充分发挥流水线的效率。流水线需要有通过时间和排空时间。流水线的分类。

②流水线的性能指标

吞吐率：在单位时间内流水线所完成的任务数量或输出结果的数量。$TP=n/T_k$

n：任务数；Tk：处理完成 n 个任务所用的时间

流水线的实际吞吐率和最大吞吐率　$TP=[n/(k+n-1)]TP_{max}$

解决流水线瓶颈问题的常用方法：细分瓶颈段和重复设置瓶颈段

加速比：完成同样一批任务，不使用流水线所用的时间与使用流水线所用的时间之比。$S=T_s/T_k$, 实际加速比和最大加速比：E=S/K，效率：E=n/(k+n-1)

即流水线中的设备实际使用时间与整个运行时间的比值，即流水线设备的利用率。

流水线设计中的若干问题：瓶颈问题、流水线的额外开销（流水寄存器延迟和时钟偏移开销）、冲突问题

描述流水线的工作，最常用的方法是时间-空间图（时空图）

③相关与流水线冲突：

相关的 3 种类型：数据相关（也称真数据相关）、名相关（反相关和输出相关）、控制相关（由分支指令引起的相关）

流水线的 3 种冲突类型：结构冲突：因硬件资源满足不了指令重叠执行的要求而发生的冲突。数据冲突：当指令在流水线中重叠执行时，因需要用到前面指令的执行结果而发生的冲突。控制冲突：流水线遇到分支指令和其他会改变 PC 值的指令所引起的冲突。由分支指令引起的延迟称为分支延迟。可采取两种措施来减少分支延迟：在流水线中尽早判断出分支转移是否成功；尽早计算出分支目标地址。

3. CPU 中断逻辑

（1）基本概念

中断源（人为设置的中断（访管指令）、程序性事故、硬件故障、I/O 设备、外部事件）

中断的分类：（可屏蔽中断和不可屏蔽中断、内部中断和外部中断、硬中断和软中断）

（2）中断系统需要解决的 7 个问题

①各中断源如何向 CPU 提出中断请求；

②当多个中断源同时提出中断请求时，中断系统如何优先响应哪个中断源的请求；

中断判优逻辑：硬件排队电路和软件轮询方法

③CPU 在什么条件、什么时候、以什么方式来响应中断；

中断服务程序入口地址的寻找：硬件向量法和软件查询法；中断响应的条件：中断允许触发器必须为 1；中断响应的时间：当前指令周期结束

中断隐指令（CPU 在中断周期由硬件自动完成）

CPU 相应中断后，进入中断周期。在中断周期，CPU 要自动完成一系列操作，包括：保护程序断点、寻找中断服务程序入口地址；关中断。

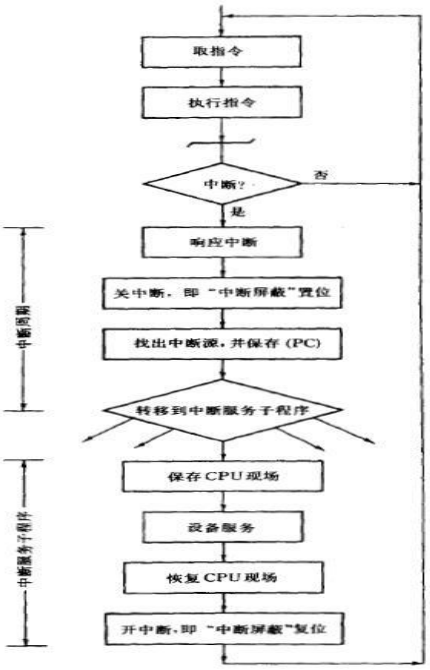


图 8.3 中断处理过程流程图

④CPU 响应中断后如何保护现场；

保护现场应该包括保护程序断点和保护 CPU 内部各寄存器的现场两个方面。程序断点的现场由中断隐指令完成，各寄存器的内容可在终端服务程序中由用户（或系统）用机器指令编程实现。恢复现场是指在中断返回前，必须将寄存器的内容恢复到终端处理前的状态，这部分工作也由中断服务程序完成。

⑤CPU 响应中断后，如何停止原程序的执行转入中断服务程序的入口地址；

⑥中断处理结束后，CPU 如何恢复现场，如何返回到原程序的间断处；

⑦在中断处理过程中又出现了新的中断请求，CPU 该如何处理。

要解决上述七个问题，只有在中断系统中配置相应的硬件和软件，才能完成中断处理任务。

（3）中断屏蔽技术

1）实现多重中断的条件

①提前设置“开中断”指令；

②优先级别高的中断源有权屏蔽优先级别低的中断源

2）中断屏蔽技术

①屏蔽触发器与屏蔽字； ②改变中断优先等级以及运行轨迹的变化

3）多重中断的断点保护

例：根据中断源优先级的先后顺序以及中断请求响应的先后次序，画出响应的中断响应轨迹图以及中断屏蔽字。改变优先级的先后顺序后有何变化？

4. 指令级并行

并行性的概念：计算机系统在同一时刻（同时性）或者同一时间间隔内（并发性）进行多种运算或操作。实现并行性的三种技术途径：时间重叠、资源重复、资源共享。从处理数据的角度来看，并行性等级从低到高可分为：字串位串、字串位并、字并位串、全并行。从执行程序的角度来看，并行性等级从低到高可分为：指令内部并行、指令级并行、线程级并行、任务级或过程级并行、作业或程序级并行。从计算机体系结构的角度来看：现代计算机的并行性等级可分为：（指令级并行 Instruction-Level Parallelism ）、数据级并行（Data-Level Parallelism）、多处理机和线程级并行（Multiprocessors and Thread-Level Parallelism）、需求级并行（Request-Level）。

几乎所有的处理机都利用流水线来使指令重叠并行执行，以达到提高性能的目的。这种指令之间存在的潜在并行性称为指令级并行（ILP：Instruction-Level Parallelism）。

流水线处理机的实际 CPI：理想流水线的 CPI 加上各类停顿的时钟周期数：

CPI 流水线 = CPI 理想 + 停顿结构冲突 + 停顿数据冲突 + 停顿控制冲突

理想 CPI 是衡量流水线最高性能的一个指标。

IPC（Instructions Per Cycle）：每个时钟周期完成的指令条数

基本程序块：一段除了入口和出口以外不包含其他分支的线性代码段。

循环级并行：使一个循环中的不同循环体并行执行。

最基本的开发循环级并行的技术：

循环展开（loop unrolling）技术、采用向量指令和向量数据表示

指令的动态调度（记分牌算法和 Tomasulo 算法）

动态分支预测技术（预测分支是否成功；尽快找到分支目标地址或指令）

基于硬件的前瞻执行

技术	降低CPI的哪一组成部分
转发和旁路	潜在的数据冒险停顿
延迟分支和简单分支调度	控制冒险停顿
基本编译器流水线调度	数据冒险停顿
基本动态调度（记分牌）	由真相关引起的数据冒险停顿
循环展开	控制冒险停顿
分支预测	控制停顿
采用重命名的动态调度	由数据冒险、输出相关和反相关引起的停顿
硬件预测	数据冒险和控制冒险停顿
动态存储器消除二义性	涉及存储器的数据冒险停顿
每个周期发出多条指令	理想CPI
编译器 相关性分析、软件流水线、踪迹调试	理想CPI、数据冒险停顿
硬件支持编译器推测	理想CPI、数据冒险停顿、分支冒险停顿

ILP大体有两种不同的开发方法：（1）依靠硬件来帮助动态发现和开发并行；（2）依靠软件技术在编译时发现并行。

流水线CPI=理想流水线CPI+结构化停顿+数据冒险停顿+控制停顿

理想流水线CPI可以用来度量能够实现的最佳性能。通过缩短上式右侧各项，可以降低总流水线CPI，也就是提高IPC（每个时钟周期执行的指令数）。利用上面的公式，可以看到一项技术能够缩小总CPI的哪一部分，以此来刻画各种技术的特征。

多指令流出技术：超标量（Superscalar）、超长指令字 VLIW（Very Long Instruction Word）、超流水处理机

5．处理器（CPU）的功能和设计

CPU 的功能：指令控制、操作控制、时间控制、数据加工

CPU 的控制方式：同步控制、异步控制、联合控制

CPU 中的主要寄存器（通用寄存器和专用寄存器）

X86：数据寄存器（DR）、指令寄存器（IR）、程序计数器（PC）、地址寄存器（AR）、累加器（AC）、状态寄存器（PSW）

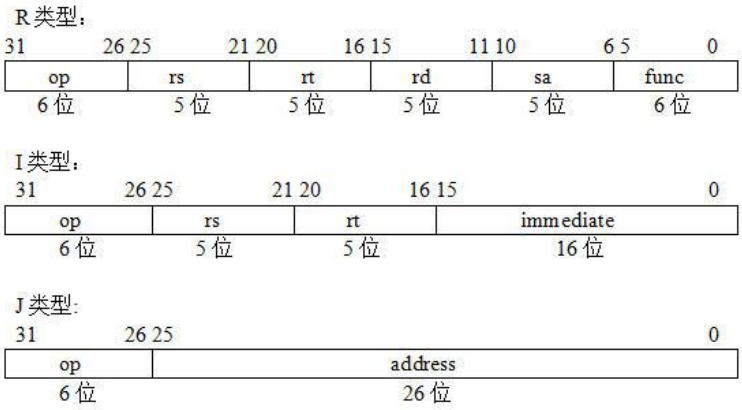
CU 设计（硬联技术和微程序设计）

MIPS 数据通路部件

指令周期（取指时间+执行时间）；CPU 周期（机器周期、总线周期）；时钟周期（节拍周期、T 周期）；相互关系：1 个指令周期 = 若干个 CPU 周期 ；1 个 CPU 周期 = 若干 T 周期

1）MIPS 单周期 CPU（一个时钟周期执行一条指令的实现机制）**设计**：

①设计或选取 MIPS32 核心指令集中选取要实现的合适指令集



例：MIPS 简单的核心指令

存储器访问指令：取字（lw）、存字（sw）

算术逻辑指令：加法（add）减法（sub）、与运算（AND）、或运算（OR）、小于则设置（slt）

分支指令：相等则分支（beq）、跳转（j）

②设计建立数据通路所需要的元素：

数据通路部件：指令存储器（指令地址和指令存储器）、寄存器堆（读写数据和读写寄存器）、ALU 和加法器；程序计数器、数据存储器、符号扩展单元

③主控单元设计：根据每条指令如何使用数据通路设计相应的控制单元逻辑

④单周期 CPU 仿真测试

单周期 CPU 实现方式的特点和缺陷：每条指令占用 1 个 CPU 周期（CPI=1）；逻辑设计简单，时钟设计也简单。但各组成部分的利用率不高（各部分大部分时间都在保持信号），并且时钟周期对所有指令等长，这样时钟周期要由执行时间最长的那条指令决定（取数指令）。现代设计中不采取这种方式。

2）MIPS 多周期 CPU 设计

多周期 CPU 的基本思想是把每条指令的执行分成多个大致相等的阶段，每个阶段在一个时钟周期内完成；各阶段功能相对完整，产生下一阶段需要的数据并保存；比如完成一次存储器访存、一次寄存器读写或一次 ALU 操作等；各阶段的执行结果在下一个时钟到来时保存到相应存储单元或稳定地保持在组合电路中；时钟的宽度以最复杂阶段所用时间为准，通常取一次存储器读写时间。

每阶段的执行由控制器输出不同控制信号，用于控制器中各单元电路的有序运行。控制器由有限状态机实现，状态机在每个时钟改变一个状态，输出相应控制信号，控制一个特定的执行阶段的具体操作。

①MIPS 多周期 CPU 数据通路设计

多周期数据通路设计思路：

在组合逻辑中插入寄存器，切分数据通路；大组合逻辑被别分为若干小组合逻辑；大延迟变为多核分段小延迟；不同执行执行占用不同的功能单元（允许不必顺序走完 5 个部件）

②MIPS 多周期 CPU 控制器设计

- 确定数据通路
- 划分指令执行步骤
  - 指令流程图
- 安排每条指令每个步骤的功能,并给出相应的控制信号
  - 指令流程表
- 为指令执行步骤设计状态机
- 为每个步骤的控制信号设计控制信号生成逻辑

③仿真测试

多周期 CPU 设计的特点和存在问题：指令的执行划分为多个步骤；每个步骤占用 1 个 CPU 周期；不同指令的指令周期不同；指令串行执行；整体性能提高，但各部件的利用率偏低（CPI>1）

4）MIPS 流水 CPU 设计

①划分流水线（5 级流水）

每条指令的实现需要 5 个时钟周期：取指令周期（IF）、指令译码 / 读寄存器周期（ID）、执行 / 有效地址计算周期（EX）、存储器访问 / 分支完成周期（MEM）、写回周期（WB），不同类型的指令在以上 5 个时钟周期中进行的操作各不相同。

在流水线的各个流水段之间加入被称为流水线寄存器的寄存器堆，并在这些寄存器堆上标明所连接的流水段。

PC 多路选择器被移到 IF 段，这样做的目的是保证对 PC 值的写操作只出现在一个流水段内，否则当分支转移成功的时候，流水线中两条指令都试图在不同的流水段修改 PC 值，从而发生写冲突。

每个时刻，每条指令都只在一个流水段上是活动的，因此，任何指令所作的任何动作都发生在一对流水线寄存器之间，具体操作由指令类型决定。

②流水线的控制

由于每个部件执行的不是同一条指令，但必须控制到每个功能部件。解决方案：把控制信号和数据一样流动起来。为区分起见，把控制信号前面加上标记，如_IF 等。每个时钟周期往下一步骤传递控制信号，保证正确的控制信号到达正确的位置。

③冲突时的解决方案

结构冲突（因资源冲突而无法使用某种指令组合）：暂停流水线执行，插入等待周期、增加资源，解决资源冲突。

数据冲突（EXE 段冲突、MEM 段冲突）：由数据的缺失引发。旁路技术，将结果尽快传送到需要使用它的位置（需要对数据通路的修改）；检测（指令译码阶段）与暂停。

控制冲突（全局冲突），由 PC 的缺失引发。程序执行转移类指令而引起的冲突。转移类指令如无条件转移、条件转移、子程序调用、中断等，它们属于分支指令，执行中可能改变程序的方向，从而造成流水线断流。解决方案是：①暂停流水线（直到有了正确的转移地

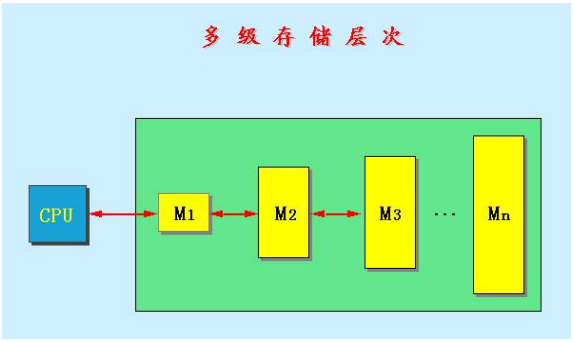
址，造成性能的降低）；②预测分支不成功（顺序执行下一条指令，清除错误启动的指令）；③预测分支成功（计算转移目的地址）；④动态预测（硬件根据上次分支的结果进行本次预测）；⑤编译器处理（延迟槽）

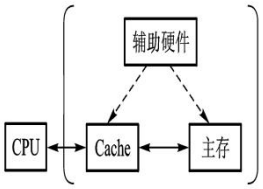
4 支持流水 CPU 的仿真测试

流水 CPU 的特点：流水线的每个阶段完成一条指令执行过程的一部分不同阶段并行完成不同指令执行过程的不同部分；多条指令同时运行，占用 CPU 不同的资源；流水线并没有缩短单条指令的时间，但提高了整个系统的吞吐率；连续不断地提供指令才能发挥流水线的效率。

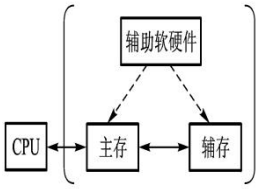
第五章　存储器

1. 存储器的层次结构





(a) “Cachc-主存” 层次



(b) “主存-辅存” 层次

2. 存储器技术

主存技术：

RAM-SRAM-DRAM-RSRAM-DDR；

ROM-PROM-EPROM-EEROM-闪存（Flash memory）

存储体扩展技术：位扩展、字扩展、位字扩展

- 存储体芯片选择：假定芯片容量mxn，组成的存储容量为M字，每字N位，则所需要的芯片数为：

$$\frac{M}{m}\times\frac{N}{n}$$

- 注意位扩展和字扩展时地址线的选择

存储器带宽的计算

Tm：存储周期，W：数据宽度（位数）

$$B_m=\frac{W}{T_m}$$

总线带宽的计算：Dr=D/T

3）高速缓冲存储器（Cache）

存储层次要解决的四个问题

（1）当把一个块调入高一层(靠近 CPU)存储器时，可以放在哪些位置上?（地址映像）：全相联、直接相联（块地址 mod 块数）、组相联（块地址 mod 组数）

（2）当所要访问的块在高一层存储器中时，如何找到该块?（查找算法）：通过查找目录表实现

（3）当发生失效时，应替换哪一块？（替换算法）：随机法、FIFO 法、LRU 法

（4）当进行写访问时，应进行哪些操作？（写策略）：写直达法（含写缓冲）、写回法

Cache 性能分析

失效率（与硬件速度无关）

3C 模型：强制性失效、容量失效、冲突失效

平均访存时间： 平均访存时间 ＝ 命中时间＋失效率×失效开销

程序执行时间：CPU 时间＝(CPU 执行周期数+存储器停顿周期数)× 时钟周期时间

其中：存储器停顿时钟周期数＝

=“读”的次数×读失效率×读失效开销＋“写”的次数×写失效率×写失效开销

存储器停顿时钟周期数＝访存次数×失效率×失效开销

$$CPU\ 时间= IC\times\left(CPI_{execution}+\frac{访存次数}{指令数}\times 失效率\times 失效开销\right)\times 时钟周期时间$$

$$每条指令的平均失效次数=\frac{失效次数}{指令数}=\frac{访存次数\times 失效率}{指令数}$$

$$CPU\ 时间= IC\times\left(CPI_{execution}+\frac{存储器停顿周期数}{指令数}\right)\times 时钟周期时间$$

Cache 失效对于一个 CPI 较小而时钟频率较高的 CPU 来说，影响是双重的：$CPI_{execution}$越低，固定周期数的 Cache 失效开销的相对影响就越大；在计算 CPI 时，失效开销的单位是时钟周期数。因此，即使两台计算机的存储层次完全相同，时钟频率较高的 CPU 的失效开销较大，其 CPI 中存储器停顿这部分也就较大。因此 Cache 对于低 CPI、高时钟频率的 CPU 来说更加重要

Cache 性能优化技术：根据公式：平均访存时间＝命中时间＋失效率×失效开销，可以从三个方面改进 Cache 的性能：降低失效率；减少失效开销；减少 Cache 命中时间

3. cache 的命中率

从 CPU 来看，增加一个 cache 的目的，就是在性能上使主存的平均读出时间尽可能接近 cache 的读出时间。为了达到这个目的，在所有的存储器访问中由 cache 满足 CPU 需要的部分应占很高的比例，即 cache 的命中率应接近于 1。由于程序访问的局部性，实现这个目标是可能的。

在一个程序执行期间，设 N_c 表示 cache 完成存取的总次数，N_m 表示主存完成存取的总次数，h 定义为**命中率**，则有

$$h=\frac{N_c}{N_c+N_m} \tag{3.4}$$

若 t_c 表示命中时的 cache 访问时间，t_m 表示未命中时的主存访问时间，$1-h$ 表示未

命中率，则 cache/主存系统的平均访问时间 t_a 为：

$$t_a = ht_c + (1-h)t_m \tag{3.5}$$

我们追求的目标是，以较小的硬件代价使 cache/主存系统的平均访问时间 t_a 越接近 t_c 越好。设 $r = t_m/t_c$ 表示主存慢于 cache 的倍率，e 表示访问效率，则有

$$e = \frac{t_c}{t_a} = \frac{t_c}{ht_c + (1-h)t_m} = \frac{1}{h + (1-h)r} = \frac{1}{r + (1-r)h} \tag{3.6}$$

由表达式看出，为提高访问效率，命中率 h 越接近 1 越好，r 值以 5—10 为宜，不宜太大。

命中率 h 与程序的行为、cache 的容量、组织方式、块的大小有关。

4）可信存储器层次

对外设可靠性能的评价参数：可靠性（Reliability）;可用性（Availability）;可信性（Dependability）

系统的可靠性：系统从某个初始参考点开始一直连续提供服务的能力。用 MTTF 衡量。

平均无故障时间（mean time in failure，MTTF），MTTF 的倒数就是系统的失效率。如果系统中每个模块的生存期服从指数分布，则系统整体的失效率是各部件的失效率之和

年失效率（annual failure，AFR）

系统的可用性：系统正常工作的时间在连续两次正常服务间隔时间中所占的比率。

$$\text{可用性} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

平均修复时间（Mean Time To Restoration，MTTR）

平均失效间隔时间（Mean Time Between Failure，MTBF），即 MTTF+MTTR

系统的可信性：服务的质量。即在多大程度上可以合理地认为服务是可靠的。**（不可以度量）**

例题：磁盘的 MTTF 和 AFR

解：整个系统的失效率为：

$$\text{系统失效率} = 10 \times \frac{1}{1000000} + \frac{1}{500000} + \frac{1}{200000} + \frac{1}{200000} + \frac{1}{1000000} = \frac{23}{1000000}$$

系统的 MTTF 为系统失效率的倒数，即：

$$\text{MTTF} = \frac{1000000}{23} = 43500 \text{ 小时}$$

即将近 5 年。

纠一位错，检测两位错的海明编码

海明码是广泛采用的一种有效的校验码，它实际上是一种多重奇偶校验码，其实现原理是：在有效信息位中加入几个校验位形成海明码，并把海明码的每一个二进制位分配到几个奇偶校验组中。当某一位出错后，就会引起有关的几个校验位的值发生变化，这不但可以发现错误，还能指出错误的位置，为自动纠错提供了依据。

假设海明码长 m 位(其中校验位为 K 位、信息位为 N 位)，能检测和自动校正一位错，并能发现两位错。若海明码的最高位号为 m，最低位号为 1，即有 $H_m H_{m-1} \cdots H_2 H_1$，则海明码的编码规则为：

(1) 校验位和信息位之和为 m，每个校验位 P_i 在海明码中被分到位号 2^{i-1} 的位置上，其余各位为信息位。

(2) 海明码每一位 H_i 由多个校验位校验，其关系是被校验的每一位的位号等于校验它的各校验位的位号之和。这样安排的目的是希望校验的结果能正确地反映出错位的位号。

(3) 在增大码距时，应使所有编码的码距尽量均匀地增大，以保证对所有代码的检测能力能均衡地提高。

虚拟机（Virtual Machine，VM）:在二进制指令系统结构（ISA）的层次上提供一个完整的系统级环境（系统虚拟机）

虚拟监视器（virtual machine monitor，VMM）,支持虚拟机的软件,虚拟机技术的核心，底层的硬件平台称为主机，它的资源被客户端虚拟机共享。VMM 决定如何将细腻资源映射到物理资源，而物理资源则可能是分时共享、划分，甚至通过软件模拟。

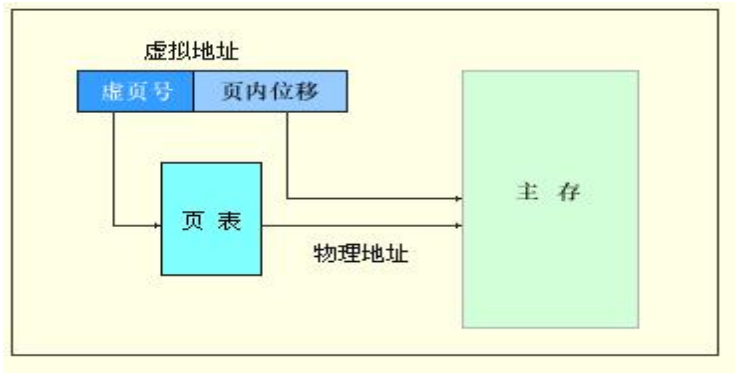
虚拟存储器：一种将主存用作辅助存储器高速缓存的技术

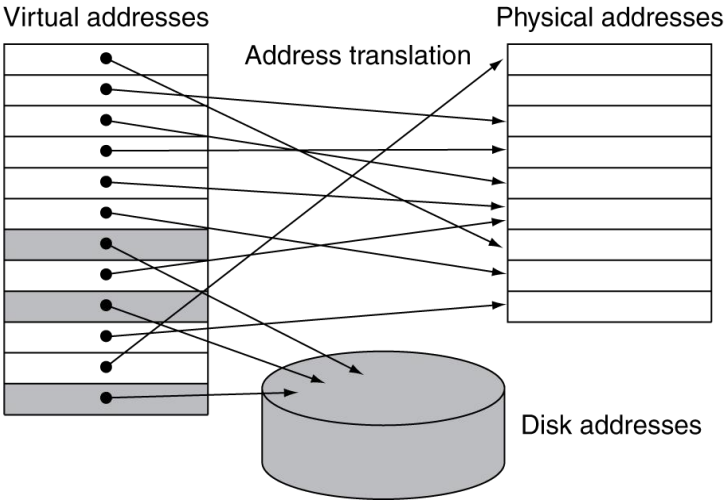
基本原理：⑴虚拟存储器是“主存－辅存”层次进一步发展的结果；

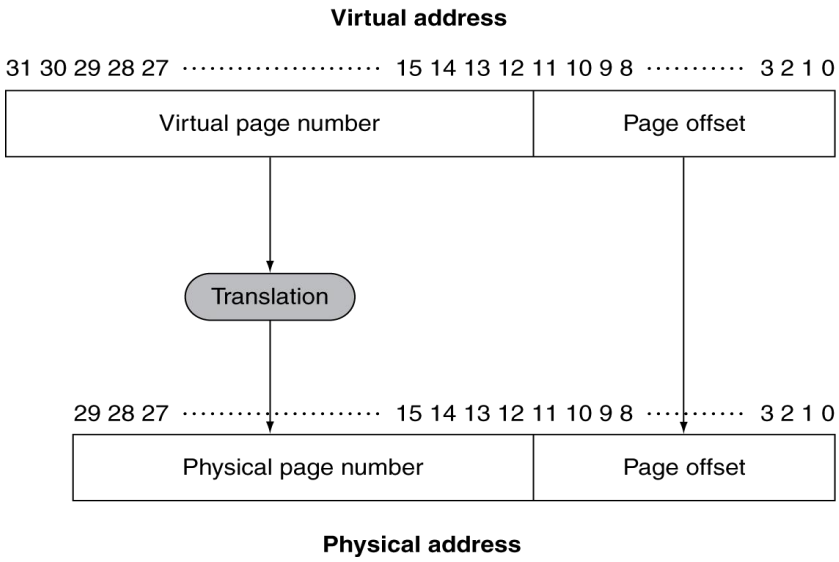
⑵虚拟存储器可以分为两类：页式和段式；

⑶有关虚拟存储器的四个问题：映象规则：全相联；查找算法：页表，段表，TLB；替换算法：LRU；写策略：写回法

用页表实现虚拟地址到物理地址的映射







快表（TLB）:TLB 是一个专用的高速缓冲器，用于存放近期经常使用的页表项；TLB 中的内容是页表部分内容的一个副本；TLB 也利用了局部性原理。

进程保护与虚存实例

进程：程序呼吸的空气和生存的空间。也就是说，一个正在运行的程序加上它继续执行所需的任何状态就是进程。 进程保护：最简单的保护机制是用一对寄存器来检查每一个地址，以确保地址在两个界限之间。基地址和上界地址。检测条件：基地址≤地址≤上界地址。在有些系统中，地址被看作是无符号的整数。检测条件就变为：（基地址＋地址）≤上界地址

计算机硬件设计者为操作系统设计者提供支持

1）提供至少两种模式：区分正在运行的进程是用户进程还是操作系统进程；称后者为内核进程、超级用户进程或管理进程

2）使 CPU 状态的一部分成为用户进程，可读但不可写。包括：基地址/上界地址寄存器；用户/管理模式位；异常许可/禁止位

3）提供一种机制，使得 CPU 能从用户模式进入管理模式和从管理模式进入用户模式

一个简单的 Cache 控制器的有限状态机：空闲-标志比较-写回-分配

并行与存储器层次结构：Cache 一致性

迁移与复制

Cache 一致性协议：监听协议和目录协议

例题与思考

例题:有一个具有 14 位地址和 8 位字长的存储器,问:⑴ 该存储器能存储多少字节的信息？⑵ 如果存储器由 1K×4 位 SRAM 芯片组成，需要多少片？⑶ 需要地址多少位作芯片选择？

【分析解答】 （1）容量：16KB 或 16384B；

（2）所需芯片数：32 片

（3）需要 4 位地址作芯片选择。

例题：某计算机系统的内存储器由 cache 和主存构成，cache 的存取周期为 45ns，主存的存取周期为 200ns。已知在一段给定的时间内，CPU 共访问内存 4500 次，其中 340 次访问主存。问：⑴ cache 的命中率是多少？⑵ CPU 访问内存的平均时间是多少？⑶ Cache-主存系统的效率是多少？

【分析与解答】

（1）cache 的命中率 H=$\frac{Nc}{Nc+Nm}=\frac{4500-340}{4500}$=0.92

（2）CPU 访存的平均时间 Ta=H·Tc+(1-H)Tm=0.92×45+(1-0.92)×200=57.4ns

（3）Cache-主存系统的效率 e=$\frac{Tc}{Ta}=\frac{45}{57.4}$=0.78=78%

5. 要求用 256K×16 位 SRAM 芯片设计 1024K×32 位的存储器。SRAM 芯片有两个控制端：当$\overline{CS}$有效时，该片选中。当 $\overline{W}$/R=1 时执行读操作，当 $\overline{W}$/R=0 时执行写操作。

6. 用 32K×8 位的 EPROM 芯片组成 128K×16 位的只读存储器，试问：

（1）数据寄存器多少位？

（2）地址寄存器多少位？

（3）共需多少个 EPROM 芯片？

（4）画出此存储器组成框图。

7. 某机器中，已知配有一个地址空间为 0000H—3FFFH 的 ROM 区域。现在再用一个 RAM 芯片(8K×8)形成 40K×16 位的 RAM 区域，起始地为 6000H。假设 RAM 芯片有$\overline{CS}$和$\overline{WE}$信号控制端。CPU 的地址总线为 A_{15}—A_0，数据总线为 D_{15}—D_0，控制信号为 R/W(读/写)，$\overline{MREQ}$(访存)，要求：

（1）画出地址译码方案。

（2）将 ROM 与 RAM 同 CPU 连接。

例 3.3 某存储器数据总线宽度为 32 bit，存取周期为 250 ns。试问该存储器的带宽是多少？

解 由题目可知，T_m=250 ns，W=32 bit

$$B_m=\frac{W}{T_m}=\frac{32\text{ bit}}{250\text{ ns}}=\frac{32\text{ bit}}{250\times10^{-9}\text{ s}}=128\times10^6\text{ bit/s}=16\times10^6\text{ B/s}$$

所以该存储器的带宽为 16×10^6 B/s。

【例 1】（1）某总线在一个总线周期中并行传送 4 个字节的数据，假设一个总线周期等于一个总线时钟周期，总线时钟频率为 33MHz，总线带宽是多少？（2）如果一个总线周期中并行传送 64 位数据，总线时钟频率升为 66MHz，总线带宽是多少？

解：（1）设总线带宽用 D_r 表示，总线时钟周期用 $T=\frac{1}{f}$ 表示，一个总线周期传送的数据量用 D 表示，根据定义可得

$$D_r=D/T=D\times\frac{1}{T}=D\times f=4\text{B}\times33\times10^6/\text{s}=132\text{MB/s}$$

（2）64 位=8B，

$$D_r=D\times f=8\text{B}\times66\times10^6/\text{s}=528\text{MB/s}$$

第六章　从客户端到云的并行处理器

$$\frac{1}{(1-\text{可加速部分比例})+\frac{\text{可加速部分比例}}{\text{理论加速比}}}$$

并行处理面临的挑战：程序中的并行性有限、相对较高的通信开销

加速比的挑战的例子：要达到某个加速比，原有程序中允许的顺序执行部分，计算规模和负载均衡等。

SISD、MISD、SIMD、SPMD（单程序流多数据流）和向量机

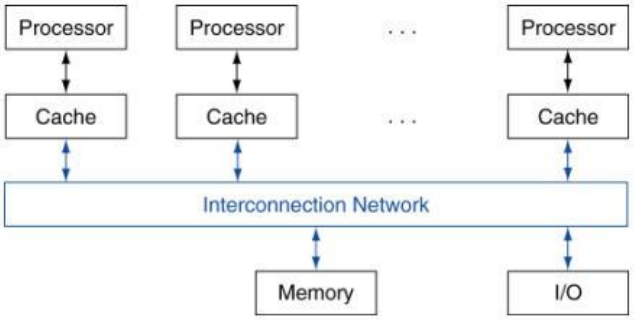
硬件多线程（粗粒度多线程、细粒度多线程、同时多线程）

多核和其他共享内存多处理器

图形处理单元（GPU）与 CPU 的主要区别

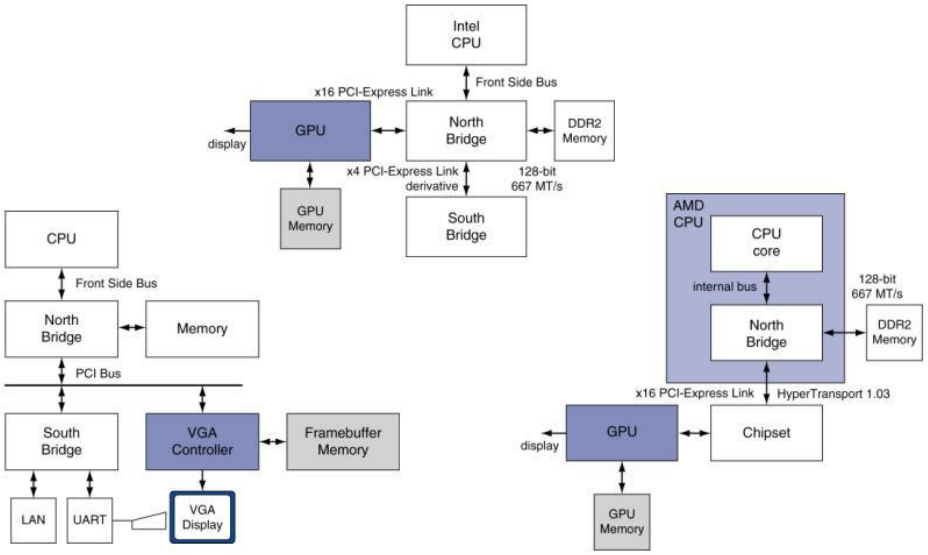
统一存储访问（UMA）和非统一存储访问（MUMA）

共享内存多处理器的典型组成



对多个处理器维护一致性协议称为 Cache 一致性协议。目前有两类采用了不同的共享数据状态跟踪技术的协议，它们分别是：①目录协议：物理存储器中共享数据块的状态及相关信息均被保存在一个称为目录的地方；②监听协议：每个 Cache 除了包含物理存储器中块的数据拷贝之外，也保存着各个块的共享信息。Cache 通常连在共享存储器的总线上，各个 Cache 控制器通过监听总线来判断它们是否有总线上请求的数据块。

GPU 在计算机系统中的位置



期末笔试题型分布

一、概念类（40%）

1. 填空题（每空 1 分，共 20 分）
2. 选择题（每空 1 分，共 10 分）
3. 判断题（每空 1 分。共 10 分）

二、综合应用类（60%）

三、总评=期末 60%+平时 40%（含期中的设计作业和期末的综合实践）