目录

[**第1章 计算机系统概述** 3](#_Toc31273512)

[**1.1 计算机的发展历程** 3](#_Toc31273513)

[**1.2 计算机系统的基本组成** 3](#_Toc31273514)

[**1.3 计算机系统的层次结构** 3](#_Toc31273515)

[**1.4 程序开发与执行过程** 4](#_Toc31273516)

[**1.5 计算机系统性能评价** 4](#_Toc31273517)

[**第2章 数据的机器级表示** 5](#_Toc31273518)

[**2.1 数制和编码** 5](#_Toc31273519)

[**2.2 整数的表示** 6](#_Toc31273520)

[**2.3 实数的表示** 7](#_Toc31273521)

[**2.4 十进制数的表示** 7](#_Toc31273522)

[**2.5 非数值数据的编码表示** 8](#_Toc31273523)

[**2.6 数据的宽度和存储** 8](#_Toc31273524)

[**2.7 数据校验码** 9](#_Toc31273525)

[**第3章 运算方法和运算部件** 9](#_Toc31273526)

[**3.1 高级语言和机器指令中的运算** 9](#_Toc31273527)

[**3.2 基本运算部件** 10](#_Toc31273528)

[**3.3 定点数运算** 10](#_Toc31273529)

[**3.4 浮点数运算** 11](#_Toc31273530)

[**3.5 运算部件的组成** 12](#_Toc31273531)

[**第4章 指令系统** 12](#_Toc31273532)

[**4.1 指令格式设计** 12](#_Toc31273533)

[**4.2 指令系统设计** 12](#_Toc31273534)

[**4.3 指令系统实例** 15](#_Toc31273535)

[**4.4 程序的机器级表示** 15](#_Toc31273536)

[**第5章 中央处理器** 16](#_Toc31273537)

[**5.1 CPU概述** 16](#_Toc31273538)

[**5.2 单周期处理器设计** 18](#_Toc31273539)

[**5.3 多周期处理器设计** 19](#_Toc31273540)

[**5.4 微程序控制器设计** 20](#_Toc31273541)

[**5.5 异常和中断处理** 20](#_Toc31273542)

[**第6章 指令流水线** 21](#_Toc31273543)

[**6.1 流水线概述** 21](#_Toc31273544)

[**6.2 流水线处理器的实现** 21](#_Toc31273545)

[**6.3 流水线冒险及其处理** 22](#_Toc31273546)

[**6.4 高级流水线技术** 23](#_Toc31273547)

[**第7章 存储器分层体系结构** 25](#_Toc31273548)

[**7.1 存储器概述** 25](#_Toc31273549)

[**7.2 半导体随机存取存储器** 26](#_Toc31273550)

[**7.3 存储器芯片的扩展及其与CPU的连接** 27](#_Toc31273551)

[**7.4 半导体只读存储器和Flash存储器** 27](#_Toc31273552)

[**7.5 并行存储器结构技术** 28](#_Toc31273553)

[**7.6 高速缓冲存储器** 28](#_Toc31273554)

[**7.7 虚拟存储器** 29](#_Toc31273555)

[**第8章 互连及输入输出组织** 30](#_Toc31273556)

[**8.1 外部设备的分类与特点** 30](#_Toc31273557)

[**8.2 输入设备和输出设备** 31](#_Toc31273558)

[**8.3 外部存储设备** 32](#_Toc31273559)

[**8.4 外设与CPU、主存的互连** 33](#_Toc31273560)

[**8.5 I/O接口** 34](#_Toc31273561)

[**8.6 I/O数据传送控制方式** 34](#_Toc31273562)

[**8.7 I/O子系统概述** 35](#_Toc31273563)

[**8.8 内核空间I/O软件** 36](#_Toc31273564)

[**第9章 并行处理系统** 36](#_Toc31273565)

[**9.1 并行处理系统概述** 36](#_Toc31273566)

[**9.2 多处理器系统** 37](#_Toc31273567)

[**9.3 多计算机系统** 38](#_Toc31273568)

[**9.4 向量处理机和SIMD技术** 38](#_Toc31273569)

[**9.5 并行处理编程模式简介** 39](#_Toc31273570)

# **第1章 计算机系统概述**

## **1.1 计算机的发展历程**

1.1.1 通用电子计算机的诞生

ENIAC: Electronic Numerical Integrator and Computer

1.1.2 第一代计算机

电子管计算机：EDSAC、IAS、UMVAC-1、IBM 704

存储程序stored program通用电子计算机：冯诺依曼提出的现代计算机基本结构

1.1.3 第二代计算机

晶体管计算机：TRADIC、IBM 7070、IBM 7090、IBM 1401等

1.1.4 第三代计算机

集成电路计算机：IBM 360

1.1.5 第四代计算机

超大规模集成电路计算机

## **1.2 计算机系统的基本组成**

1.2.1 计算机硬件

内存：包括主存和高速缓存

外存：包括辅助存储器和后备存储器

辅存：运行时直接和主存交换信息的存储器，如硬盘

设备控制器：也称I/O控制器、I/O适配器、I/O模块或I/O接口。之所以名称繁多且都是I/O，因为这个概念非常笼统且没有统一标准，但所有的设备控制器本质工作都是管理I/O

寄存器堆register file：也称通用寄存器组GPRs或寄存器文件

1.2.2 计算机软件

略

## **1.3 计算机系统的层次结构**

1.3.1 计算机系统抽象层的转换

低级语言low-level language：也称机器级语言，包括机器语言和汇编语言

汇编语言assembly language：机器语言的符号化表示语言，和机器指令几乎是一一对应，在组成结构方面也是一一对应的。

机器语言machine language：用二进制编码的机器指令instruction

目标程序：目标程序不一定非得是机器语言，目标指的是翻译后的目标

汇编器assembler：将汇编语言翻译为机器语言

解释器interpreter：按顺序逐条翻译成机器指令并立即执行

指令集体系结构Instruction Set Architecture，ISA：也称体系结构或系统结构architecture。是硬件和软件之间的接口，也是软件可见部分。从某种意义上来说，ISA就是软件的终极接口。

微架构microarchitecture：也称微体系结构，是指令具体实现的组织方式，是硬件层面的实现细节，是软件不可见部分。

1.3.2 计算机系统的不同用户

（1）最终用户end user

工作于应用程序层

（2）系统管理员system administrator

工作于操作系统虚拟机

（3）应用程序员application programmer

工作于高级语言虚拟机

（4）系统程序员system programmer

工作于机器语言机器层面

透明：一个实际存在的事务或概念从某些角度看不到

## **1.4 程序开发与执行过程**

1.4.1 从源程序到可执行程序

略

1.4.2 可执行文件的启动和执行

·因为有I/O功能，所以大部分硬件都有自带的缓冲器来在共享总线上增加传输效率。

1.4.3 程序与指令的关系

指令通常分为若干字段：

操作码字段

地址码字段——操作数的内存地址或寄存器编号

立即数字段——直接给出操作数或偏移地址

指令的解码：指令执行时通过控制器将操作码字段解释成控制信号control signal来控制数据通路的执行。

1.4.4 指令的执行过程

·冯诺依曼简单模型机：p16/32

①CPU——

算数逻辑部件arithmetic and logic unit，ALU：执行算数或逻辑运算

GPRs：通用寄存器组

标志寄存器：ALU运算的标志信息，如是否为0或负数、进位或借位等

控制器：译码指令操作码，生成控制信号

指令寄存器instruction register，IR：存放主存读出的指令

程序计数器program counter，PC：存储下一条要被执行的指令的地址

存储器地址寄存器memory address register，MAR：和地址总线相连

存储器数据寄存器memory data register，MDR：和数据总线相连

②总线bus——

地址信号线

控制信号线：和控制器相连

数据信号线

定长/变长指令字系统：决定了如何在每次指令执行完毕后，如何更新PC计数器的内容，定长只需加上定值，变长需要根据指令的译码结果来计算

·注意这只是冯诺依曼模型，即简单模型。现代计算机为了性能已经添加多种功能模块，但是原理并未大变

时钟信号clock signal

## **1.5 计算机系统性能评价**

1.5.1 计算机性能的定义

吞吐率throughput：也称带宽bandwidth

响应时间response time：也称执行时间execution time或等待时间latency

1.5.2 计算机性能的测试

·执行时间：2部分

①CPU时间——也包括2部分

用户CPU时间：该进程自身花费时间

系统CPU时间：系统调用花费时间

②其他时间——等待I/O操作完成和并行处理其他进场的时间

CPU时间计算的指标：

①时钟周期clock cycle，tick，clock tick，clock

CPU用于同步各个控制操作的时钟定时信号（或主脉冲信号）的宽度

②时钟频率clock rate，主频

前者的倒数

③CPI，cycles per instruction

执行一个或全部程序的或全部指令集的指令所需的（平均）时钟周期数

计算机的性能可以看成是CPU时间的倒数。

1.5.3 用指令执行速度进行性能评估

·早期计算机指令的执行时间彼此差别不大，加法指令可以代表乘除性能，因而可以使用这个标准进行性能评估

MIPS：million instruction per second，每秒钟完成单个运算（指令）的数目，也称作定点指令运行速度

MFLOPS：million FLOating-point operations per second，每秒执行的浮点运算的操作次数，是用操作次数而非指令数目做指标，还有GFLOPS，TFLOPS，PFLOPS等单位。

指令平均执行时间：也称等效指令速度法或Gibson混合法，若以时钟周期数做单位就是CPI

peak MIPS：在特殊的指令组合下，可以得到最大的MIPS值

relative MIPS：以参考机型为标准的MIPS比值

1.5.4 用基准程序进行性能评估

基准程序benchmarks：一组用于测试计算机性能的程序集，标准上来说不同功能的计算机所用基准程序不同，也无法横向比较。SPEC测试程序集是最广泛的标准。

SPEC ratio：执行时间的归一化比值，是以Sun SPARCstation为参考机型计算的

·基准程序测试的缺点是可以被特殊优化处理的编译器蒙蔽，从而生成过于高的性能测试结果

# **第2章 数据的机器级表示**

## **2.1 数制和编码**

2.1.1 信息的二进制编码

二进制编码的十进制数binary coded decimal number，BCD：另一种计算机内部表示数值的方法，区别于直接用二进制表示

2.1.2 进位记数制

基数base：进制的名称，也是每个数位上可以使用的不同数字符号的个数，如10进制基数为10

（1）R进制数转换成十进制数

p28/44

（2）十进制数转换成R进制数

分为整数部分和小数部分

p29/45

（3）二、八、十六进制数的相互转换

p30/46

2.1.3 定点与浮点表示

（1）定点表示

定点表示一般都是针对浮点数的整数或小数部分，即一般小数点定位在最左或最右

（2）浮点表示

X = (-1)^SIGN \* 尾数 \* 基数^指数

尾数mantissa：浮点数的定点小数部分

阶或指数exponent：浮点数的定点整数部分，常用移码表示

基数radix，base：取值2,4,16

·此表示方法下，浮点数实际上是两个数（一个纯小数，一个纯整数或其倒数）的乘积

2.1.4 定点数的编码表示

机器数：数值在计算机内部的编码表示

真值：机器数真正的值

定点数编码表示方法（4种）：

①原码

缺点：算术计算不方便，虽然是人类常用

·一般只用定点原码小数来表示浮点数尾数

②补码

也称2-补码two’s complement

补码定义：一个数和其补码的概念是相对于一个参考数和一个循环数值系统而言的。该参考数称为模，而该循环系统称为模运算系统，该循环周期就是模。模运算系统和取余运算是等价的，而模就是除数。互为补码的数实则为同余的数，且往往二者都是绝对值小于模的数。

·计算机的有限位数和溢出性决定了其也是循环数值系统，其模数就是2^n，n为计数位数。则计算减法A-B时，只需加上-B的补码即可。实用中，只需将减法转化为加法计算，因此只需对负数进行补码转换，正数不变。

补码和真值之间的转换：p34/50

·注意，补码表示的一个优点（特点）就是补码将符号位融入到了其数值表示中。这可以从实际原理看出，补码表示将负数转化为一个相同字长的无符号正数，因为有符号位的限制该正数和带符号的正数区间不冲突。而该无符号正数因为模系统限制，与其他数相加时得到的结果和原负数可得结果相同。

·注意，尽管负数补码表示的首位和其他位在原理上是一体解读（不同于原码和直觉）的，但是仍然把首位作为符号位来参考和定义。

·注意，补码的定义有些复杂，“按位取反加1”的说法实际上是一种对操作结果的过程简化。如果按照该说法来理解补码，那对补码的符号位的理解也就无需上一条注意事项了。

③反码

p35/51，不常用

循环进位：反码计算中出现的操作，因为负数的反码比补码少1，所以计算时某些情况下必须结果加1才能得到正确结果

④移码

定义：将真值（无论正负）加上2^(n-1)，从而保证结果一定为正数（结果变为无符号整数）。只用来表示定点整数，主要是浮点数的阶。

阶码：用移码表示的浮点数的阶

## **2.2 整数的表示**

2.2.1 无符号整数的表示

略

2.2.2 带符号整数的表示

用补码表示

2.2.3 C语言中的整数类型

无符号unsigned short，unsigned int（unsigned），unsinged long

有符号short，int，long

·C编译器会隐式地将带符号整数强制类型转换为无符号

## **2.3 实数的表示**

2.3.1 浮点数的表示格式

·浮点数无法表示无限趋近于0的正或负数，这被称为正下溢或负下溢，而同理超过正负上限时称为正/负上溢

·浮点数的机器零值有+-之分，取值也不唯一，一般机器会将0值的阶码也强制置零

2.3.2 浮点数的规格化

规格化：即保证尾数的最高位非零，另一种表述是基数为R时，尾数部分真值的绝对值大于等于1/R（注意数制相同）。

2.3.3 IEEE 754浮点数标准

单精度32位：1位符号+8位阶码+23位尾数

双精度64位：1位符号+11位阶码+52位尾数

基数为2，尾数采用原码，隐藏首位尾数1且将其置于小数点之前

指数采用移码，但偏置常数为2^(n-1)-1

（1）全0阶码全0尾数：+0/-0

（2）全0阶码非0尾数：非规格化数

·非规格化数的特点是隐藏位为0，引入其的作用是在规格化数据无法触及的区间（正负下溢）内表示数据，具体原理见p41/57

（3）全1阶码全0尾数：正负无穷

略

（4）全1阶码非0尾数：非数NaN，not a number

非数分为2种：

不发信号quiet或静止的——尾数最高有效位为1

发信号signaling或通知的NaN——尾数最高有效位为0

（5）阶码非全0且非全1：规格化非0数

·浮点数也有扩展格式，即比双精度更长的浮点数格式，但是IEEE 754只对其进行了松散规定，因此浮点数扩展格式是因处理器而异的。

2.3.4 C语言中的浮点数类型

单精度float

双精度double

扩展双精度long double

## **2.4 十进制数的表示**

2.4.1 用ASCII码字符表示

ASCII：American Standard Code for Information Interchange

不同的数符表示方式：

前分隔数字串——符号位单独用一个字节表示，位于数串前

后嵌入数字串——符号位嵌入到最低一位数字的ASCII码的高3位中（十进制数的ASCII码的低四位就是8421码，高三位恒定为011，表示正数时不作为，表示负数时高三位变成111）

2.4.2 用BCD码表示

基2码：表示每个十进制数位的4个二进制数

（1）有权BCD码

8421码：也称自然BCD码，NBCD码，即按照习惯用等值的二进制数与十进制数对应，因此二进制数位每一位的权就是8421.

（2）无权BCD码

余3码

格雷码

## **2.5 非数值数据的编码表示**

2.5.1 逻辑值

逻辑数据和数值数据机器形式一致，是通过指令的操作码来区分的

2.5.2 西文字符character

码表：字符集的编码表

2.5.3 汉字字符

汉字代码的种类：

（1）汉字的输入码（外码）

定义：将汉字用英文键盘按键编码的方式

码元：组成编码的基本元素，如汉字输入码的码元是英文键盘按键（注意不是英文字符）

（2）字符集与汉字内码

内码：编码在计算机内部存储的形式（不一定和机器外一致）

国标码（国标交换码）：汉字的一种字符编码方式，注意和输入码与内码的区别

区位码：表示汉字在国标码码表中的位置的数据，具体描述见p49/65

Unicode-16和UCS-2编码是一致的，均使用2字节

（3）汉字的字模点阵码和轮廓描述

字形信息库：也称字库font，是描述一个字符集一种形式写法的形状描述信息集合

字形的描述方法2种：

点阵码——即bitmap

轮廓描述——用数学公式描述文字曲线，字形大小可任意变化，如Adobe Typel和True Type

## **2.6 数据的宽度和存储**

2.6.1 数据的宽度和单位

字word：一种类似比特和字节的数据计量单位，但是并非固定大小随平台而定。而数据的宽度也是如此，依赖编译器和机器的特性而异

字长：CPU内部用于整数运算的数据通路的宽度，也是数据运算、存储和传送的通路宽度。也就是机器的位数，32位或64位

·字和字长不是一个概念，在同一台机器上也不一定相等

存储容量：（从2^10字节开始以2^10递增）

kilobyte，megabyte，gigabyte，terabyte，petabyte，exabyte，zettabyte，yottabyte。

2.6.2 数据的存储和排列顺序

最低有效位least significant bit，LSB：表示数的最低位

最高有效位most significant bit，MSB：表示数的最高位

·注意最低/最高有效字节也是一样的缩写！！

·现代计算机都采用字节编址方式，即一个地址一个字节

大端big endian：MSByte在低地址，也是符合直觉书写习惯的方式

小端little endian：LSByte在低地址

·注意大端小端是以字节为单位的逆序，即字节序，而字节内部的顺序都是“大端”——即最高有效位bit在低地址。

## **2.7 数据校验码**

码距：码制中各码字间的最小距离

码字：由若干位代码组成的一个字，在编码制度（码制）中代表一个完整独立含义

海明距离：两个码字间不同代码的位的个数，也叫码字间的距离，但不能简称

2.7.1 奇偶校验码

只能发现奇数个位出错，不能发现偶数个位出错，也不能纠错。常用于校验一字节长的代码，且非常高效。

2.7.2 海明校验码

故障字syndrome word：对接收到的校验字和重新计算的校验字进行异或操作的结果

（1）校验位的位数的确定

略

（2）分组方式的确定

略，较复杂，见p57/73

（3）校验位的生成和检错、纠错

单纠错码SEC：只能对单个位进行纠错

双检错码DED：可发现两位错

·二者可以结合为SEC-DED

2.7.3 循环冗余校验码cyclic redundancy check，CRC

不基于奇偶检测，而是基于更加复杂的数学运算

（1）CRC码的检错方法

基于多项式系数的数学计算

（2）校验位的生成

略，算法复杂，此处只是简述

（3）CRC码的纠错

略

# **第3章 运算方法和运算部件**

## **3.1 高级语言和机器指令中的运算**

3.1.1 C程序中涉及的运算

（1）按位运算

主要用于实现掩码masking操作，而该操作主要用于置位功能或称为标记

（2）逻辑运算

略

（3）移位运算

逻辑移位：不考虑符号，无论左右移均移除补0；左移时如果移除的是1，则溢出

算数移位：考虑符号，左移时操作和逻辑移位一样，右移时高位补符号位；左移时如果操作前后符号位不同，则溢出

·大部分机器默认对无符号整数采用逻辑移位，对带符号整数采用算术移位

（4）位扩展和位截断运算

位扩展（不是一个显式运算符，发生在短数向长数的类型转换）

0扩展——用0补位，用于延长无符号整数

符号扩展——用符号补位，用于延长带符号整数

位截断（长数向短数转换）

直接截断

3.1.2 MIPS指令中涉及的运算

MIPS的定义：有Millions of Instructions Per Second的含义，作为指令集体系其含义是Microprocessor without Interlocked Pipeline Stages，同时也是应用其体系的计算机系统与开发公司的名称。

一个举例表格

## **3.2 基本运算部件**

包括加法器、算术逻辑部件ALU和移位器，ALU的核心部件也是加法器

3.2.1 串行进位加法器p71/87

全加和：单独计算加法本位的结果

全加进位：单独计算加法进位数的结果

串行进位加法器：也称行波进位加法器Carry Ripple Adder，CRA。因为进位数从低到高逐位传递得名。这样速度慢，不实用。

3.2.2 并行进位加法器p72/88

进位传递函数：一个中间表达式函数

进位生成函数：同上

先行进位部件：也称超前进位，Carry Lookahead Unit，CLA。实际上只是通过推导代数式子直接得出最终进位情况。并行进位加法器在数字位数较多时，高位的输入电路非常庞杂，可以采用多级先行进位器的方法。

3.2.3 带标志加法器p73/89

没有很详细的介绍标志种类，在下一节3.3中详细介绍包括ALU在内的器件细节。

3.2.4 算术逻辑部件

ALU的核心是带标志加法器，ALU简介p74/90

多路选择器MUX：接收硬件的控制信号，并且根据该信号选择性允许部分通路联通。具体来说仍然是一个逻辑组合电路。

## **3.3 定点数运算**

3.3.1 补码加减运算

Cin/Cout-进位输入/输出端

ZF-零标志

SF-符号标志

CF-进/借位标志

OF-溢出标志

·OF对无符号整数运算无意义，CF对带符号整数运算无意义，也可以理解为无符号时的“溢出位”。CF的具体作用方式（为何减法时与Cout取反）此处不明。

补码计算溢出的判断方法（2种）：p76/92

3.3.2 原码加减运算

p77/93

3.3.3 移码加减运算

移码的和差等于和差的补码，p77/93

3.3.4 原码乘法运算

①原码一位乘法

p78/94

②原码二位乘法

即将2位原码的4种取值情况来决定算法内部走向

3.3.5 补码乘法运算

①补码一位乘法

Booth乘法：p81/97

②补码两位乘法

也称改进布斯算法，modified booth algorithm，MBA，或基-4布斯算法，Radix-4

原理同原码二位乘法，用两位取值来加速计算，减少部分积的计算次数

3.3.6 快速乘法器

阵列乘法器array multiplier：利用空间错位而非电子移位，以及利用大量并行的硬件而非单独器件的串行循环，来加速计算。

基于行波进位加法器CRA的阵列乘法器：加法进位串行计算，慢

基于进位保留加法器CSA的整列乘法器：carry save adder，进位向下级传输而非同一级下一位，更快。

Wallace Tree，WT：华莱士树，一种用于减少求和次数的数据结构。

3.3.7 原码除法运算

·类似乘法，通过模拟和简化手算过程。

·除法运算之前，需要对被除数进行扩展称为2倍宽度。

单精度除法：即n位被除数扩展2n后进行除法，扩展出的n位全为0，但是可能在前或后

双精度除法：被除数本身就有2n长度，n为除数宽度

·若2n位被除数的前n位和n位除数相减后大于0，说明商的结果可能超过n位，进而溢出。除非是浮点数尾数，可以通过改变阶码来处理。

①恢复余数除法

每次中间余数减去除数为负时，再将除数加回以恢复，效率较低

②不恢复余数除法

又称加减交替法，因为上一步若中间余数减除数为负后，根据递推公式，下一步就须加上除数。但是最后一位若为负，则必须加回除数，因为此时余数为最终结果而非中间结果。

·对于两个n位定点整数除法，可以省略第一次循环，只做n次循环，因为商一定不会超过n位。但是这只是一种溢出方式，如果是两个负数相除，可能使补码上溢。后一种溢出一般在做计算之前就判定了。

3.3.8 补码除法运算

①补码恢复余数除法

p91/107

②补码不恢复余数除法

p92/108

## **3.4 浮点数运算**

3.4.1 浮点数加减运算

①对阶

移码的运算p93/109

原则：小阶向大阶对齐，以使尾数不出现更多整数（原本有个省略的1）

②尾数加减

经过适当调整的定点原码小数运算

③尾数规格化

略

④尾数的舍入处理

附加位——

·IEEE754标准不仅仅规定了浮点数本身的形式，也规定了运算中间结果要增加浮点数之外的附加位以保障精度。尾数右边的附加位依次是保护位/警戒位guard，舍入位round，和粘位sticky。

舍入方法——

·就近舍入

·正向或朝上舍入

·负向或朝下舍入

·朝0舍入或截取、截断、恒舍法

⑤溢出判断

浮点数的溢出即指数或阶数溢出

3.4.2 浮点数乘除运算

①浮点数乘法运算

略，p97/113

尾数（原码定点小数）相乘，指数（移码）相加

②浮点数除法运算

略，p98/114

## **3.5 运算部件的组成**

数据通路=运算器+传输

3.5.1 定点运算部件

中央控制部件CCU：遵循指令操作码译码进行控制操作的部件

通用寄存器组General register set，GRS

锁存器Latch：用于暂存寄存器中的数据

多路选择器MUX

Q乘商寄存器

移位器：不同的寄存器可能使用不同的专属移位器

3.5.2 浮点运算部件

p102/118

# **第4章 指令系统**

指令的种类：

微指令：微程序级命令，硬件范畴

伪（宏）指令：若干机器指令的序列，软件范畴

机器指令：即通常意义上的指令

## **4.1 指令格式设计**

4.1.1 指令地址码的个数

指令包含的信息：

操作码

源操作数或地址

结果地址

下一条指令地址（不一定所有指令都有）

指令分类（地址个数）

三地址、二地址、一地址、零地址

4.1.2 指令格式设计原则

p108/124

## **4.2 指令系统设计**

4.2.1 基本设计问题

p109/125

4.2.2 操作数类型

略

4.2.3 寻址方式

有效地址EA：可能是物理地址或虚拟地址，但是必须对应数据本身。任何形式的间接地址不算有效地址。

形式地址A：指令中的地址码字段给出的地址，即名义上或形式上的地址，可能是中间地址、偏移量（相对寻址）或基准地址（变址寻址）等。

1）立即寻址

立即数

2）直接寻址

直接地址或绝对地址

3）间接寻址

4）寄存器寻址

5）寄存器间接寻址

寄存器中存储有效地址

6）变址寻址

基准地址+偏移量的方式，此时A为基准地址

·变址寄存器I：存放偏移量（I），括号表示I寄存器中的内容

EA=（I）+A

·主要用于线性表或数组的访问

7）相对寻址

即在指令之间的相对移动，A为偏移量，参考点为程序计数器内容

EA=（PC）+A

8）基址寻址

类似变址寻址，但是A为偏移量

·基址寄存器B：存放基准地址的寄存器

EA=（B）+A

·主要用于程序的动态重定位

·变址、基址、相对3种寻址统称偏移寻址

9）其他寻址方式

如隐含地址码

4.2.4 操作类型

1）算术和逻辑运算指令

·常见：

加ADD，减SUB，乘MUL，除DIV，比较CMP，与AND，或OR，取反NOT，取负NEG，异或XOR，加1INC，减1DEC。

·特殊：

带进位的加ADC，带借位的减SBB

2）移位指令

各种移位操作的含义：p113/129

算术左右移

逻辑左右移

小循环左右移

大循环左右移

半字交换

3）传送指令

寄存器之间MOV，内存读取LOAD，存入内存STORE

4）串指令

对字符串进行操作的指令

串传动，串比较，检索，传送转换

5）顺序控制指令

条件转移BRANCH

无条件转移JMP

跳步SKIP：跳过下一条指令

调用CALL：也称转子指令，和转移指令的区别在于必须保存调用位置

返回RET：调用返回

6）CPU控制指令

也称管态指令，一般只在内核使用

停机、开关中断，系统模式切换等

7）输入输出指令

用于完成外设数据交换，若I/O地址与主存地址统一编址，可以用访存指令代替

4.2.5 操作码编码

1）定长操作码编码

固定长度的操作码，但指令字总长度可变。指令可能有几种不同的离散长度，但一般均为1字节倍数。

优点是译码方便，速度快，但是空间有冗余。

2）扩展操作码编码

即变长操作码，进一步可分等步长扩展和不等长扩展。前者操作码的长度变化离散且均匀递变。

·扩展操作码编码：p115/131

4.2.6 标志信息的生成与使用

条件码condition codes，CC：也称状态位status，标志信息

条件码寄存器：也称状态寄存器，标志寄存器，程序状态字寄存器program status word register，PSWR或PSR

·标志信息主要用于硬件层面实现条件或关系表达式的判定。各种不同的标志信息的组合可以明确显示出判定结果，进而可以用于条件转移指令。

4.2.7 指令系统设计风格

1）按操作数位置指定风格来分

①累加器accumulator型指令系统

其中一个操作数往往隐含在累加器AC中，指令较短，但是所有运算步骤都需要经过累加器，效率低。只有早期系统采用。

②栈stack型指令系统

指令的操作数默认来自栈顶，指令较短，但是编译不够灵活。JVM采用此方式。

③通用寄存器general purpose register型指令系统

使用寄存器来存放临时数据，包括立即数I，寄存器R，存储单元S中的4种组合RR、RS、SI、SS形式的指令。

④Load/Store型指令系统

是通用寄存器型的特殊类型，即只有取数或存数才能访存，或只有RR、RI型指令。

2）按指令格式的复杂度来分

①CISC（complex instruction set computer）

复杂指令集计算机：复杂之处在于指令格式复杂，用以支持更复杂的集中操作，以便更好的支持高级语言或软件产品。

·特点：指令复杂多变且不利于流水线化，甚至不能完全依赖硬件支持实现。难以进行编译优化，因为指令可选择太多。

②RISC（reduced instruction set computer）

精简指令集计算机：只保留最常用的简单指令。

·特点：采用Load/Store型风格，指令规整可流水线化

·Intel架构的计算机一般均采用CISC为主，或混合RISC方案。这是带有历史兼容和商业原因的。但总体而言，RISC更具优势，尤其是在移动设备中。

## **4.3 指令系统实例**

4.3.1 IA-32指令系统

1）指令格式

①前缀p121/137

类似于指令的属性参数的设置

·指令前缀

·段前缀

·操作数长度前缀

·地址长度前缀

②指令

·操作码

·寻址方式

·SIB：变址或基址寻址信息

·位移

·直接数据

2）寻址方式

p122/138

4.3.2 ARM指令系统

p123/139

1）数据处理指令格式

2）装入/存储指令格式

3）多数据装入/存储指令格式

4）跳转指令格式

·ARM的所有指令都有一个条件码前缀，用于控制标志或条件判断。而且移位操作也被包含在各种指令中。

·ARM提供9种寻址方式，p125/141

## **4.4 程序的机器级表示**

4.4.1 MIPS汇编语言和机器语言

1）MIPS指令中数据的表示

p125/141（包含MIPS通用寄存器名称与编号表）

·寄存器一般用$符号标识，后加编号（一个正整数）或名称（如a0,t1,s2,k3,zero等）表示，如$a0或$4。

2）MIPS指令格式和寻址方式

p126/142

常用寄存器代号：

Rs：双目运算符下，R型、I型和分支指令的第一源操作数；加载存储指令中的基址

Rt：双目时R型和分支的第二源操作数；移位指令的源操作数；I型的目的操作数

Rd：R型指令中的目的操作数

shamt：R型指令中的移位数

func：R型指令中的操作类型

·指令是32位定长，即4字节，因此指令的地址必为4的倍数，即最后两位为00，因而可以省略，以创造更多可用性。

3）MIPS汇编语言

指令解码表：机器代码和汇编表示之间的对应表

汇编：从汇编表示转换为机器代码

反汇编：上述反过程

4.4.2 选择结构的机器代码表示

p129/145

4.4.3 循环结构的机器代码表示

p130/146

4.4.4 过程调用的机器代码表示

1）MIPS中用于过程调用的指令

jal 跳转并链接jump and link

jr 寄存器跳转jump register

2）过程调用时MIPS寄存器的使用约定

这只是惯例性约定：p132/148

3）MIPS中的栈和栈帧

p132/148

4）MIPS过程调用协议

p133/149

①调用过程P在过程调用前的执行步骤

②被调用程序Q中的执行步骤

开始段、本体段（过程体）、结尾段

·示例：p135/151

# **第5章 中央处理器**

CPU中控制指令执行的部件为控制器，可以用硬连线路或微程序实现。

## **5.1 CPU概述**

5.1.1 指令执行过程

指令周期：CPU取出并执行一条指令的时间

寄存器传送语言register transfer language，RTL：形式化描述寄存器相关的基本操作的表征方式

R[r]：寄存器r的内容

M[addr]：读取存储单元addr

→：传送方向

PC：程序计数器的内容（不须加中括号）

PC<end:begin>：寄存器或控制信号等（二进制内容）的部分内容，起止位由begin和end指定，一般最低起始位为0，起止边界是包含性的或[begin,end]。

5.1.2 CPU的基本功能和基本组成

CPU组成：

数据通路datapath

执行部件execution unit或功能部件function unit

控制部件control unit

CPU的基本组成：p143/159

5.1.3 数据通路的基本结构

指令执行元件：

组合逻辑元件或操作元件

存储元件或状态元件

连线方式：

总线方式和分散连接方式

1）组合逻辑元件

定义：输出只取决于当前输入，不受时钟信号的控制。

包括：多路选择器MUX，加法器Adder，算术逻辑部件ALU，译码器decoder

2）状态元件

D触发器：最简单的状态元件种类，有时钟输入Clk，状态输入D和状态输出Q

·时钟周期：

建立时间setup time

保持时间hold time

延迟时间“Clock-to-Q” time

·不同类型的寄存器

①暂存器

暂存器：锁存器组成的寄存器，通常用做指令寄存器IR，GRS

锁存器latch：带“写使能write enable，WE”输入信号的触发器

写使能：当WE=0时，寄存器不工作

通用寄存器组general register set，GRS：工作原理p146/162

②输出端带有一个三态门

③带复位（清零）功能的寄存器

④带计数（自增）功能的寄存器

⑤带移位功能的寄存器

3）数据通路与时序控制

①早期计算机的三级时序系统

包括机器周期、节拍、脉冲

机器周期：指令周期的一个基本部分，如取指令、存储器读/写

节拍：机器周期的一个部分动作，如送地址、发命令，检测数据

脉冲：更低层的时钟信号

②现代计算机的时钟信号

现代时钟信号不区分三级，只称时钟周期。

锁存延迟latch prop：即Clk-to-Q

时钟偏移clock skew：由于工艺、走线原因导致的时钟同步误差

·最长操作延迟和最短操作延迟的约束：p148/164

最短延迟的约束在于操作元件的输出改变不能早于hold time结束

4）单总线数据通路

·CPU内部总线（数据通路）和外部总线（存储器总线）不是一个概念。

存储器数据寄存器MDR：所访问地址读取/写入的存储器数据的暂留地

存储器地址寄存器MAR：用于访问存储器某个位置的地址

①在通用寄存器之间传送数据

p149/165

②完成算术、逻辑运算

p149/165

控制逻辑延迟：Clk-to-Signal

③从内存读取一个字（指令或数据或数据的地址）

异步方式：主存读取数据的时间不定，需要返回信号通知CPU

同步方式：读取数据的过程需要固定几个时钟周期完成

④把一个字（数据）写入主存

p151/167

5）三总线数据通路

p151/167

·不管单总线或三总线，都难以实现流水线化，这里只是介绍。

## **5.2 单周期处理器设计**

处理器设计过程（概述）：p152/168

5.2.1 指令功能的描述

p153/169

5.2.2 数据通路的设计

1）算术逻辑部件的设计

p154/170

2）取指令部件的设计

p157/173

3）R-型指令的数据通路

p157/173

4）立即数运算指令的数据通路

p158/174

5）Load/Store指令的数据通路

p159/175

6）分支指令的数据通路

p159/175

7）无条件转移指令的数据通路

p161/177

8）综合11条指令的完整数据通路

p162/178

5.2.3 控制逻辑单元的设计

根据每个指令功能所需的控制信号，来决定操作码和实际控制信号之间的译码过程，以及实现。

1）控制信号取值分析

p162/178

2）控制单元设计

①主控制器的设计

可编程逻辑阵列Programmable Logic Array，PLA

p167/183

②ALU局部控制器的设计

略

5.2.4 时钟周期的确定

计算机性能或程序执行速度由3个因素决定：

指令集内指令数目

时钟周期——单周期处理器通常取最复杂指令所用的时间

CPI——单周期处理器的CPI恒为1

## **5.3 多周期处理器设计**

多周期处理器思想：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，每个阶段一般只包括一个基本操作。

5.3.1 信号竞争问题

写使能信号、地址、数据信号之间的同步关系必须满足，写使能信号到达前，另外两者必须稳定。因此，但凡需要写使能（即不一定每个周期均需要写入）的情况，必定需要两个周期来协同。

5.3.2 指令执行状态分析

状态：多周期条件下，每个时钟周期或一个指令执行阶段

1）取指令、指令译码/取数阶段

①取指令状态IFetch

②译码/取数状态RFetch/ID

投机计算：在某些阶段，某些元件本属于空闲，但是具备计算某些内容的条件，然而并不能确定这些内容是否一定需要。但如果需要则可以加速执行，反之并不损失额外性能。

2）R-型指令运算阶段

①执行RExec

②结束RFinish

3）I-型指令立即数运算阶段

①执行oriExec

②结束oriFinish

4）lw指令执行阶段

①访存地址计算状态MemAdr

②存储器取数状态MemFetch

③结果写回寄存器状态lwFinish

5）sw指令执行阶段

①访存地址计算状态MemAdr

②存储器存数swFinish

6）分支指令执行阶段

BrFinish

7）无条件跳转指令执行阶段

jumpFinish

5.3.3 硬连线路控制器设计

多周期控制器采用2种方式实现：

基于有限状态机描述：也称组合逻辑控制器，PLA控制器，硬连线路控制器

基于微程序描述

摩尔机：有限状态机的一种，特点是信号的输出结果仅依赖于当前状态，而与其他输入无关。

·多周期控制器和单周期最大的区别在于多了一个“下一状态”计算的逻辑。都含有操作码（加本状态）计算控制信号的逻辑。

## **5.4 微程序控制器设计**

微程序：由若干条微指令（也称微操作μop，但是这个术语似乎也用在微命令层）组成的程序，每条微指令相当于有限状态机的一个状态，通常每个时钟周期执行一条微指令。每个指令由一个微程序表示。

控制存储器control storage，CS：也称控存，是一种专门存储微程序的ROM。

5.4.1 微程序控制器的结构

微命令：微操作中的一个控制信号，即是微操作的一部分。可以说，微指令相当于精简指令的指令，微命令就是指令包含的控制信号，分别对应一个基本运算或传送。

控制字control word，CW：也称微指令字

微指令寄存器μIR：即存储微指令的IR

微程序寄存器μPC：存储微指令地址的计数器

5.4.2 微命令编码——微操作码格式

微指令：类似的，由微操作码和微地址码两部分组成。

1）不译法（直接控制法）

即一位对应一个微命令，执行快，不必译码，但是浪费编码空间

2）字段直接编码法

将微指令划分为若干字段，每个字段包括若干彼此互斥（不能并行）的微命令，对每个字段进行编码，然后直接组合即可。

3）字段间接编码法

略

4）最少（最短、垂直）编码法

所需微操作码位数最少，但是没有考虑并行性，微程序一般较长

5.4.3 微指令地址的确定

下址：下一条微指令的控存地址，简称

1）计数器法

类似宏观层面的指令调度，但是不同之处在于必须临时插入转移微指令，因为微程序虽然是指令的实现，但是不像指令在程序中一样顺序存储，而是孤立的。

2）断定法（下址字段法）

每个微指令都包含下地址字段，代价是微指令变长，好处是定址逻辑简化。

固件firmware：固化了的微程序，或存储于ROM的微程序，不可更改。

## **5.5 异常和中断处理**

5.5.1 基本概念

1）内部异常exception

分为：

硬故障中断：掉线，断电，线路出错

程序性异常：也称软中断、例外，包括以下3种

①故障

也称失效，如“缺页”，“除数为0”，“非法操作码”

②自陷

也称陷阱或陷入，不是意外性的错误或异常，而是预先设定和安排的

系统调用，条件自陷，都是自陷的例子

③终止

和硬故障中断类似，但是是软件层面的反应

2）外部中断interrupt

也称I/O中断或外部中断

·异常和中断的定义随着时代和书籍的变化有区别。不必太在乎定义。

5.5.2 异常处理过程

包括两部分

1）保护断点和程序状态

程序状态字寄存器PSWR，program status word

2）识别异常事件并转异常处理

cause：专门记录异常原因的寄存器

5.5.3 带异常处理的处理器设计

略

# **第6章 指令流水线**

## **6.1 流水线概述**

6.1.1 流水线的执行效率

指令流水段：

取指令IF

指令译码ID

取操作数OF

执行EX

写回WB

·每个指令的流水段显然都是可能不一样的，不等长的，一般拿最长最多步的指令来作为参考。

6.1.2 适合流水线的指令集特征

1）指令长度应尽量一致

便于取指令和指令译码

2）指令格式应尽量规整

便于加速译码过程，不同字段可同时译码

3）采用load/store型指令风格

除了load/store指令外的指令不访问存储器，便于规整指令执行过程

4）数据和指令在存储器中对齐存放

有利于减少访存次数

## **6.2 流水线处理器的实现**

6.2.1 每条指令的流水段分析

功能段/流水段

0）共有的两个：Ifetch取指令并计算PC，Reg/Dec寄存器取数并译操作码

1）R-型指令功能段划分

Exec执行ALU逻辑，Write写回寄存器

2）I-型运算类指令功能段划分

同上

3）lw指令功能段划分

Exec用ALU计算地址，Mem从存储器读取，Write写回

4）sw指令功能段划分

Exec用ALU计算地址，Mem在存储器写入

5）beq指令功能段划分

Exec用于计算逻辑表达式和转移地址，Mem（或WrPC功能段）用于写入PC

6）j指令功能段划分

WrPC写入PC

·为了统一指令的执行过程，需要加入“空”功能段，但是不能随意添加。必须保证流水线执行的不冲突——每个部件每个指令只能一次性使用，且必须在相同的阶段使用。

6.2.2 流水线数据通路的设计

流水段寄存器：流水线处理器的特有寄存器特征，用于保存每个处理阶段的中间信号与结果。

1）Ifetch（IF）段

p199/215

2）Reg/Dec（ID）段

p199/215

3）Exec（Ex）段

p200/216

4）Mem段

p201/217

5）Wr段

p202/218

6.2.3 流水线控制器的设计

类似于单周期控制器，一旦译码之后产生全部阶段的控制信号且不会改变，所有信号依次传递到流水段寄存器中，直到在特定周期该信号被执行成操作便不再向下传。

## **6.3 流水线冒险及其处理**

6.3.1 结构冒险structural hazard

也称硬件资源冲突hardware resource conflict，通过满足特定约束和分离独立硬件的设计方式即可避免。如寄存器的读写分离（同一周期前后分离），指令存储器IM和数据存储器DM的分离，cache缓存的数据和代码分离等

p203/219

6.3.2 数据冒险data hazard

也称数据相关data dependency。

写后读read after write，RAW：由于前面指令写回结果之前后续指令就要读取结果产生的冒险。在非乱序执行的基本流水线中，所有数据冒险都属于RAW。

1）插入空操作指令

软件上延迟执行，实现简单但是低效

2）插入气泡bubble

硬件上阻塞冲突指令执行，实现复杂但是仍然低效

3）采用转发技术forwarding

也称旁路bypassing，即在指令未完全执行完毕但结果已经生成的情况下提前将结果发出给后续指令的相应阶段。p205/221

·这是非常复杂的更多的控制逻辑，了解即可。转发技术可以解决大部分RAW，但不包括Load-use。

4）Load-use数据冒险的检测和处理

即lw指令之后紧跟相关运算类指令，运算指令前期需要源操作数，但lw指令后期才能备好数据。

最好的办法是通过编译器重排指令顺序避免出现Load-use，是为一种软件优化技术。

硬件处理Load-use冒险的策略：p211/227

6.3.3 控制冒险control hazard

即由各种转移、调用、中断引发的指令顺序被打乱的情况

1）转移指令引起的控制冒险

也称分支冒险branch hazard

延迟损失时间片C：由于流水线阻塞而带来的延迟执行周期数

①简单预测

是一种静态预测方式，或者说就是盲目固定选项的方式，如所有循环分支总是预测执行分支。正确率和实际分布一致。

②动态预测dynamic prediction

根据转移历史经验性指导预测进行，平均正确率可达90%，但也和自身属性和实际程序逻辑相关

分支历史记录表branch history table，BHT

分支预测缓冲branch prediction buffer，BPB

分支目标缓冲branch target buffer，BTB

·以上3者都是记录转移发生历史情况的表

预测位：用于提示控制逻辑进行预测的信息

1位预测位：1时预测转移，0时预测不转移，每次预测错误都将预测位取反

2位预测位：实际上用1和0的序列来记录转移发生的历史，可以区分更多的预测情况。准确率略有提高。

③延迟分支delayed branch

是编译器调度指令顺序的另一种优化形式，将原本计划在分支指令前执行的代码移到分支后（前提是不产生副作用）来填补延迟损失时间片，不够时用nop（空操作）补位。

·减少分支延迟损失时间片C的数目是提高流水线执行效率的关键，也严重影响到延迟分支等优化操作的效果。这只能通过将分支指令的条件检测所处的阶段提前来实现。

2）异常或中断引起的控制冒险

·异常或中断在指令序列中的优先级显然是越早执行的指令越高，但是流水线中可能会发生后面的指令出现异常而前面的指令还没到出错的阶段。因此所有异常和中断响应都在出错指令执行结束后才解决。

举例：单周期、多周期和流水线处理器的工作效率比较p218/234

注意，这里体现出多周期和流水线技术严重依赖阶段划分的均匀性。

## **6.4 高级流水线技术**

指令级并行ILP：包括2种策略

超流水线super-pipelining：增加流水段数目，减少时钟周期

多发射流水线multiple issue pipelining：同时启动多条独立指令来并行，如不同种类的运算。

超标量superscalar处理器：采用多发射流水线技术的处理器

IPC，instructions per cycle：每周期完成指令数，用于衡量CPI<1时的性能，也是CPI的倒数。

指令打包：将能够并行处理的多条指令同时发送，或合并为一个长指令中。

6.4.1 静态多发射处理器

发射包：也称长指令，静态多发射指令最初也称超长指令字very long instruction word，VLIW

1）2-发射MIPS处理器

同时处理两条流水线（指令），p220/236

循环展开：将循环体的内容显式表达成指令或代码，重写几次循环体，则循环次数要除以几倍。这样做为的是代码调度有更多对象，但是增长了程序和寄存器的使用。

2）Intel的IA-64的结构特点

显式并行指令计算机explicitly parallel instruction computer，EPIC：intel为其VLIW计算机的命名

寄存器窗口重叠技术：

指令组instruction group：IA-64的编译器用来标识并行指令的方式，所有同组指令无寄存器级数据依赖。指令组长度任意。

指令包bundle：执行时，并行指令被重新编码整合的结果，包长度固定128位，固定有3个41位长的指令。

谓词化技术：

谓词：分支指令中的条件

x86-64：后更名AMD64，兼容IA-32的64位指令集，不同于IA-64，它保留了IA-32指令集，因而兼容性更好

IA32-EM64T：全名extended memory 64 technology，64位内存扩展技术。后更名Intel 64。是Intel公司在IA-64失败后推出的新指令集系统，与IA-32和x86-64兼容。

6.4.2 动态多发射处理器

也需要编译器进行指令顺序调整，但是不由其决定具体的并行打包细节。

动态流水线调度dynamic pipeline scheduling：处理器通过指令相关性检测和动态分支预测，投机性地不按指令顺序执行，尤其是流水线阻塞时自动寻找无依赖指令提前执行。也称乱序执行out-of-order scheduling。

功能单元：两个性能指标p224/240

执行周期数latency

发射时间issue time

重排序缓冲reorder buffer：也称ROB。用于保存指令完成的结果，等待写回寄存器堆，也可以转发到功能部件。

执行模式：例子p225/241

按序发射、按序完成

按序发射、无序完成

无序发射、无序完成

·动态调度的优势在于减轻了编译器编写的压力，更好的封装了硬件细节和平台相关性，在编译器软件和处理器硬件之间达到了对并行性更好的平衡。

6.4.3 Pentium 4处理器的流水线结构

踪迹高速缓存trace cache，TC：奔4用来实现指令cache

·奔4采用硬件译码和微程序（CISC壳，RISC核）结合，静态和动态两级分支预测

奔4流水线功能段

1）TC下指针，TC next IP

相当于更新TC的PC计数器

2）TC取，TC fetch

3）分配，allocate

4）换名，renaming

5）μop排队，queuing

6）μop调度，scheduling

7）分派，dispatching

8）寄存器文件，register file

9）执行，execute

地址生成部件AGU

SSE指令：流式SIMD处理

MMX指令：多媒体信号处理

10）标志，flags

包括寄存器写回

11）分支检测branch check

# **第7章 存储器分层体系结构**

## **7.1 存储器概述**

7.1.1 存储器的分类

1）按存储元件分类

半导体存储器

磁性材料存储器

光介质存储器

2）按存取方式分类

①随机存取存储器random access memory，RAM

按地址访问存储单元，包括半导体存储器

②顺序存取存储器sequential access memory，SAM

按顺序存取，速度慢，容量大，如磁带存储器

③直接存取存储器direct access memory，DAM

前两者的集合体，如磁盘

④相联存储器associative memory

也称按内容访问存储器content addressed memory，CAM

3）按信息的可更改性分类

读写存储器read/write memory，包括RAM

只读存储器read only memory，ROM

4）按断电后信息的可保存性分类

非易失（不挥发）性存储器nonvolatile memory

易失（挥发）性存储器volatile memory

5）按功能分类

①高速缓冲存储器cache

②主存储器main memory，MM——MOS管半导体存储器

③辅助存储器

④海量后备存储器

后两者统称外存

7.1.2 主存储器的组成和基本操作

记忆单元cell：存储元，位元，都表示一个位

存储阵列bank：存储体，存储矩阵

编址单位addressing unit

编址方式addressing mode：有字节和字两种，通用计算机采用字节编址

7.1.3 存储器的主要性能指标p237/253

容量：略

价格：总价格，每位价格

速度：访问时间、存储周期、带宽

7.1.4 存储器的层次化结构

数据一般只在相邻层之间复制传送，且由慢而大的存储器到快而小的，因此不存在跨层传送。

## **7.2 半导体随机存取存储器**

简称RWM，习惯上多称RAM，分为：

双极型RAM

MOS型RAM：静态RAM，static RAM，SRAM和动态RAM，dynamic RAM，DRAM

7.2.1 基本存储元件p239/255

用来存1位信息

1）六管静态MOS管存储元件

①信息的保持

②读出

③写入

2）单管动态MOS管存储元件

①读出

②写入

③刷新

3）静态存储元件和动态存储元件的比较

SRAM：体积大，功耗大，无需刷新和再生，读写速度快，价格贵，用作cache

DRAM：体积小，功耗小，但必须刷新和再生，读写速度慢，价格低，用作主存

7.2.2 SRAM芯片和DRAM芯片

1）存储器芯片的内部结构

①存储体/存储矩阵

位平面：4096个存储单元排成的64×64阵列

②地址译码器

一维译码：也称线选法或单译码法，适用于SRAM，其芯片称为字片式芯片

二维译码：也称重合法或双译码法，适用于DRAM，其芯片称为位片式芯片

③驱动器

起到放大强化信号的作用，因为一条信号线被分成多个信号继续使用

④I/O控制电路

⑤片选控制信号

选取存储芯片的信号

⑥读写控制信号

引脚复用技术：将一个多位信号分成多个部分先后从相同的引脚输入，以减少引脚数和芯片体积。

2）DRAM芯片的刷新

刷新计数器：实际上是一个行地址生成器，用于自动生成需要刷新的行地址。刷新周期一般为64ms

①集中刷新

停工作来刷新，不用

②分散刷新

利用每个存储周期刷新一部分，少用

③异步刷新

结合前两者

3）DRAM芯片读写周期

p243/259

7.2.3 SDRAM芯片技术

即synchronous DRAM，包括DDR1到3系列

1）SDRAM芯片技术

行缓冲器row buffer：由SRAM实现，用于缓存SDRAM一次读取的一块连续信息

模式寄存器：也称工作方式寄存器，设置BL和CL

突发长度burst length，BL：传送数据的长度

CAS潜伏期CAS latency，CL：从收到读命令到开始传送数据的延迟，也称读取潜伏期read latency，RL

2）DDR SDRAM芯片技术

即double data rate，是对标准SDRAM的改进设计

3）DDR2 SDRAM芯片技术

4）DDR3 SDRAM芯片技术

I/O缓冲：存储器阵列之外的一个缓冲器，可能就是row buffer或DDR SDRAM的内部变种

n-bit预取：若不深究会有些违反常识，n-bit可以理解为内存核心向外部I/O缓冲传递一次数据，可供外部I/O传输给总线n次的量

·DDR1~3仅在于预取位数的升级，因而提升了数据传输率或带宽

## **7.3 存储器芯片的扩展及其与CPU的连接**

7.3.1 内存条和内存条插槽

总线接口部件：也称总线控制逻辑，CPU中处理总线I/O的部件

前端总线front side bus，FSB：也称处理器总线，Intel芯片中特指CPU连接到北桥芯片的总线

双通道内存插槽：还有三、四通道变种，即支持多条总线同时进行传输的内存条插槽

存储控制器memory controller：存储器相连的控制逻辑

7.3.2 存储器芯片的扩展

1）位扩展

即将每个存储地址的内容扩充，即编址方式

2）字扩展

将整个地址空间扩充，扩大总容量

3）字、位同时扩展

略

## **7.4 半导体只读存储器和Flash存储器**

7.4.1 半导体只读存储器

1）掩膜只读存储器mask ROM，MROM

分为双极型和MOS型

2）可编程只读存储器programmable ROM，PROM

分为熔丝型和反向二极管型（P-N结破坏型）

3）可擦除可编程只读存储器erasable programmable ROM，EPROM

采用MOS工艺，必须全部擦除

4）电擦除电改写只读存储器electrically erasable programmable ROM，EEPROM或E2PROM

可单独修改，但写入过程慢

7.4.2 半导体Flash存储器

也称闪存、快擦型电可擦除重编程ROM。采用EPROM的简单结构和浮栅/热电子注入的编程写入方式。

1）Flash存储元

工作原理p249/265

2）Flash存储器的基本操作p249/265

①编程（充电）

②擦除（放电）

重写过程实际上是先全部擦除为1（浮空栅不带电），再写入

③读取

·读取速度和半导体RAM相当，写入速度和硬磁盘存储器速度相当

## **7.5 并行存储器结构技术**

7.5.1 双口存储器

两个读写端口和电路并行

7.5.2 多模块存储器

和双口或多口存储器类似的空间并行技术，但是不同的口进行的访问可以是完全独立的，但是多模块之间的访问一般是相关的或协同的。

1）连续编址方式

也称按高位地址划分方式，主存地址高位表示模块号或体号

2）交叉编址方式

也称按低位地址划分方式，主存地址低位表示模块号。因为CPU进行的多为顺序读写，交叉编址方式才支持多模块并行访问。

①轮流启动

②同时启动

·取决于数据线总宽度能够承载多少数据的

## **7.6 高速缓冲存储器**

7.6.1 程序访问的局部性

略

7.6.2 cache的基本工作原理

主存块block的大小和cache的行line或槽slot的大小一般相等

1）cache的有效位valid bit

冲刷flush：将有效位置0，每个cache行均有一个有效位

2）CPU在cache中的访问过程

cache的访问过程必须在一条指令执行过程中完成，只能依靠硬件实现

3）cache-主存层次的平均访问时间

命中率hit rate和命中时间hit time

缺失率miss rate缺失损失miss penalty

7.6.3 cache行和主存块之间的映射方式

1）直接direct映射p257/273

也称模映射，每个主存块映射到固定行中，类似于一般的散列表

缺点：面对映射到同一行频繁的情况不够灵活

2）全相联full associate映射p260/276

是按内容比较查找式访问，时间开销等比较大

3）组相联set associate映射p261/277

前两种方法结合，即组间模映射，组内全映射

n路组相联即为每组n行，n一般为2的幂

4）三种映射方式比较p263/279

关联度的性质

7.6.4 cache中主存块的替换算法

1)先进先出算法fist-in-first-out，FIFO

2）最近最少用算法least-recently-used，LRU

栈算法：同一时刻小组中的块集合必然是大组中块集合的子集

颠簸pingpong或抖动thrashing：程序中的局部化变化范围大于cache组的大小，频繁切换但不命中

3）最不经常用least-frequently used，LFU

类似LRU，使用计数器相关算法实现

4）随机替换算法

无差别随机替换，性能只差一点点，但实现代价低

7.6.5 cache的一致性问题

1）全写法write through

也称通写法或直写法或写直达法，一次性同步完成所有副本的写操作，原理简单但性能差

①写分配法write allocate

主存和cache一起写入并加载cache

②非写分配法not write allocate

只写主存不加载到cache

写缓冲write buffer：cache和主存之间的一个临时存储写入内容的地点，保证CPU的高速运行和写操作的异步进行。但是不适合大量写操作同时进行，会产生阻塞。

2）回写法write back

也称一次性写方式或写回法，写命中时只修改cache，最后统一同步主存

修改位dirty bit：也称脏位，cache行中标记是否被修改过的位

7.6.6 cache性能评估

结论：处理器性能越高，cache的性能就越重要

7.6.7 影响cache性能的因素

1）单级/多级cache、联合/分离cache的选择问题

一般L1 cache最接近CPU，且采用数据和指令cache分离的设计，更关注速度；L2等高层cache速度逐渐降低，远离CPU，采用联合设计，关注命中率。

2）主存-总线-cache间的连接结构问题

①窄形结构

②宽形结构

③交叉存储器结构

3）DRAM结构、总线事务类型与cache的配合问题p271/287

7.6.8 cache结构举例p272/288

MESI协议：用于解决cache一致性问题，将每个cache行的状态分为更新modified、独占exclusive、共享shared、无效invalid。

## **7.7 虚拟存储器**

7.7.1 进程与进程的上下文切换context switching

用户级上下文：p274/290

系统级上下文：p274/290

寄存器上下文：p274/290

7.7.2 存储器管理memory management

1）分区方式partitioning

简单固定分区fixed-size partition

可变长分区variable-length partition

2）分页方式paging

虚拟地址virtual address：也称逻辑地址logical address

物理地址physical address：也称实地址

请求分页demand paging

7.7.3 虚拟地址空间

Linux虚拟地址空间的分配示意：p278/294

7.7.4 虚拟存储器的实现

1)分页式虚拟存储器

虚拟页virtual page，VP：也称逻辑页、虚页，即虚拟地址空间中的页

页框page frame，PF：页帧、物理页PP或实页

①页表

一级页表、二级或多级页表、倒置页表，用以解决页表项过多不适于全部至于主存的情况

②地址转换address translation

存储器管理部件memory management unit，MMU：CPU中

③快表p282/298

定义：由高速缓存中的页表项组成的页表，也称后备转换缓冲器translation lookaside buffer，TLB。主存中的页表原体称为慢表

④CPU访存过程

2）分段式虚拟存储器p285/301

略

3）段页式虚拟存储器p286/302

·计算机字长，亦即常说的32位或64位CPU（或计算机）是指CPU运算和数据传输的宽度，不是物理地址线的宽度。因此一个进程所允许的虚拟空间大小一般来说和机器字长相关，因为只是一个“数学”概念（当然也要看具体CPU的实现）。而物理内存的可用大小是CPU地址线位数决定的或称寻址空间，直接安装更大的内存并不一定能扩大物理内存。

7.7.5 存储保护

1）访问方式保护

2）存储区域保护

①加界重定位

②键保护方式

③环保护方式

# **第8章 互连及输入输出组织**

## **8.1 外部设备的分类与特点**

8.1.1 外设的分类

信息传输方向：

1）输入设备

媒体输入设备

交互式输入设备

2）输出设备

3）输入输出设备

功能：

1）人机交互设备

2）存储设备

3）机-机通信设备

8.1.2 外设的特点

1）异步性

2）实时性

3）多样性

## **8.2 输入设备和输出设备**

8.2.1 键盘

按键结构：

触点式——开关断通

非触点式——电信号变化，计算机中常用

输入主机信息不同：

编码键盘——键盘内部硬件已经将按键的扫描码译成ASCII码，速度快

非编码键盘——输入主机的是原扫描码，需要软件程序翻译结果，速度较慢（但并非不可接受），但是可以对按键自由重编码

硬件接口：

AT接口、大五芯接口

PS/2接口、小五芯接口

USB接口

8.2.2 鼠标器mouse

传感器技术不同：

机械式鼠标——滚轮机械运动

光电式鼠标器——光学镜头拍摄，帧变化分析计算

数字信号处理器DSP：用以根据拍摄的帧变化计算光电式鼠标位移的

硬件接口：

USB

PS/2

8.2.3 打印机

1）针式打印机

也称击打式打印机，包括：

整字形打印设备

点阵打印设备

打印控制器（打印适配器）：

数据锁存器

命令译码器

控制锁存器

状态锁存器

不可打印字符：

ASCII特定动作字符

ESC命令序列（换码序列）

2）激光打印机

打印语言：

页面描述语言PDL

Escape码语言

8.2.4 显示器

1）阴极射线管CRT显示器

行帧：水平扫描周期的倒数

帧频：即刷新频率，垂直扫描周期的倒数

2）液晶LCD显示器

两种模式：

字符模式

图形模式

显示存储器：也称显存，VRAM，刷新存储器

显示控制器CRTC：即显卡，图形显示卡

绘图处理器graphics processing unit，GPU：显卡的核心，专门计算图形的硬件

## **8.3 外部存储设备**

8.3.1 磁表面存储原理

1）磁层和磁头p301/317

2）磁表面存储器的读写过程p301/317

3）磁表面存储器的性能指标p302/318

①记录密度

②存储容量

③平均存取时间average access time

④数据传输速率

内部传输速率internal transfer rate：也称持续传输速率sustained transfer rate，指在外设内部进行读写操作时处理数据的速率，和外设种类、采用的工艺种类均有关

外部传输速率external transfer rate：也称突发数据传输速率burst data transfer rate或接口传输速率，指在外设外部（由对应控制器或CPU）对外设缓存进行读写操作的速率，一般由外设采用的接口类型而定

4）数据记录方式p303/319

8.3.2 硬盘存储器

1）硬盘驱动器

寻道操作

旋转等待操作

读写操作

·驱动器是硬盘内部运行的控制装置

2）硬盘控制器p305/321

·控制器是主机和硬盘内部逻辑之间的接口，主要处理数据信号的转换、缓存和对外传输，实际上每个外设都有对应的外设控制器，也基本都是指主机和外设接口的部件。

·然而，控制器和驱动器之间的划分并不是明确标准，只是泛泛概念。

3）磁盘的记录格式p305/321

定长记录格式

不定长记录格式

4）硬磁盘的主要技术指标p306/322

硬盘与主机接口：

SCSI、并行ATA（IDE），串行ATA（SATA）

8.3.3 冗余磁盘阵列redundant arrays of inexpensive disk，RAID

思想：利用类似于多模块交叉并行的方式将数据并行存储在多个磁盘，并行工作和相互校验可以提高性能与可靠性

1）RAID 0 — 8）RAID 7

不同的实现方案代表不同的设计结构，p307/323

8.3.4 U盘和固态硬盘solid state disk，SSD

略，p309/325

## **8.4 外设与CPU、主存的互连**

互连结构：连接各部件的通路的集合

8.4.1 总线概述

互连方式：

分散连接：各部件单独互连

总线连接：多部件公共连接

总线结构：

控制线——包括时钟线、复位线、操作命令线、总线请求和总线回答线

数据线

地址线

同步并行总线和异步串行总线：后者常用p311/327

1）总线宽度

2）总线工作频率

=时钟频率的倍数

3）总线带宽

4）总线寻址能力

地址线的位数（若无分时多次传送地址）

5）总线定时方式

6）总线传送方式

非突发、突发方式

7）总线负载能力

8.4.2 基于总线的互连结构

北桥芯片：是一个主存控制器集线器memory controller hub，MCH，本质上是一个DMA，direct memory access控制器。专门负责和主存、显存等和CPU的互连。

南桥芯片：I/O控制器集线器I/O controller hub，ICH芯片。可集成USB控制器、磁盘控制器、网卡、声卡和显卡等的控制。

·北桥和南桥也称桥接器，将处理器-存储器总线和I/O总线连接在一起。形成超大规模集成层面的芯片，因此不同的系统结构可能对功能单元的划分不同。两种桥接器一般常见于早期系统构造。

1）处理器总线

也称前端总线front side bus，FSB。可由多核共享

QPI：quick path interconnect，一种core i7采用的多核之间传输数据的总线与其采用的协议。i7中，北桥芯片集成在了CPU中，处理器-存储器总线合并为1条。

2）存储器总线

3）I/O总线p315/331

第一代：XT、ISA、EISA、VESA

第二代：PCI、AGP、PCI-X

第三代：PCI-Express

主板芯片组结构：（北桥集成在CPU后）

双芯片——

CPU、PCH（platform controller hub，包括南桥和北桥的部分功能）

三芯片——

CPU、IOH（I/O hub，支持更高级I/O接口的部件）、ICH

NVRAM，non-volatile random access memory：

管理引擎management engine，ME：

DMI，direct media interface：总线类型和协议的一种

设备控制器：外设的I/O接口，也称I/O控制器、I/O控制接口、I/O模块，可以集成于主板或插在I/O总线扩展槽，如磁盘控制器，网卡，声卡，显卡，USB接口

## **8.5 I/O接口**

8.5.1 I/O接口的功能

1）数据缓冲

2）错误或状态检测

3）控制和定时

4）数据格式转换

8.5.2 I/O接口的通用结构p318/334

8.5.3 I/O端口及其编址

端口：实际就是寄存器，如数据缓存寄存器就是数据端口

1）独立编址方式

I/O地址空间和主存地址空间独立，只能通过特别的I/O指令来区分

·Intel处理器一般都采用独立编址

2）统一编址方式（存储器映射方式）

I/O地址空间和主存地址空间共享，优点是指令简单灵活，但是导致指令译码过程更加复杂，且主存空间减少。

## **8.6 I/O数据传送控制方式**

8.6.1 程序直接控制I/O方式

直接通过查询程序来控制主机和外设之间的数据交换

1）无条件传送方式

也称同步传送方式，无定时信号（所谓定时全凭程序逻辑，也并不一定是周期性）和状态查询，只适合特殊的一些简单外设（且外设工作时间往往固定且短暂），不适合高速设备的I/O

2）条件传送方式

也称异步传送方式或程序查询方式，虽然无条件传送方式也有查询程序，但是条件传送方式的程序更加复杂，需要检测各种状态与控制信号，过程简述p321/337

根据查询程序被启动的方式：

①定时查询

即每隔固定时间周期运行查询程序，每次查询一定等到条件满足并完成传输后才返回。

②独占查询

即CPU和外设完全串行工作

8.6.2 程序中断I/O方式

1）中断的概念

2）中断系统的基本职能和结构

Intel处理器的中断系统功能简述p326/342

INTR：intel处理器的中断请求信号引脚

可编程中断控制器programmable interrupt controller，PIC：Intel的中断控制器

IRQ：每个外设连接到中断请求寄存器的引线，每个外设拥有不同编号的IRQ

中断系统中的两种优先级（可能不不同，可能相近）：

中断响应优先级——由中断判优电路决定

中断处理优先级——由中断屏蔽字动态反映

3）中断过程

①中断响应

·保存好被中断的程序的断点处的关键性信息

现场：用户可见的工作寄存器内容，可以由指令访问，并且由软件负责管理保存

断点信息：程序计数器PC和状态字寄存器PSWR等内容，不可由指令访问，由硬件负责

·识别中断源并根据中断响应优先级进行判优

软件查询方法：利用软件程序来查询判优

硬件判优方法：也称向量中断方式，此时中断服务程序的首地址PC和初始PSW称为中断向量IV，interrupt vector。存放所有中断向量（亦即中断服务程序的入口）的是中断入口地址表/中断向量表。表中的编号是为中断类型号。

·调出中断服务程序

中断请求寄存器IRR：用于存放当前所有外设中断请求位信息的寄存器，连接IRQ引线

中断屏蔽寄存器IMR：存放当前被屏蔽中断的信息，决定了哪些中断现在可以响应

中断服务寄存器ISR：

②中断处理

中断方式较程序直接控制方式更优，但仍然不适合超高速外设，尤其是存储器型外设。毕竟中断造成的上下文切换成本仍然在数据传输频率较大时显得非常高。

8.6.3 DMA方式direct memory access

即直接存储器存取，用于控制外设和主存的直接数据交换（如磁盘等），不经过CPU。

DMA方式进行磁盘数据传送过程实际上包括很多细节步骤，而这些步骤中的核心步骤虽然是DMA传送，但其余初始和辅助步骤也仍需要中断和查询方式工作。另外，除了高速存储器访问外，一般的外设仍然多采用扩展性和通用性强的中断方式工作。p333/349

1）三种DMA方式

基于DMA需要和CPU争夺总线控制权以及主存资源的情况——

①CPU停止法

二者串行

②周期挪用法

单字传送，一次CPU让出一个总线事务周期

③交替分时访问法

每个存储周期分给CPU和DMA各一部分

2）DMA操作步骤

①DMA控制器的预置（初始化）

准备内存区

设置传送参数

发送启动传送的命令

②DMA数据传送

③DMA结束处理

3）DMA与存储系统

主要是虚拟存储器结合和cache一致性的新问题讨论p335/351

## **8.7 I/O子系统概述**

I/O子系统层次结构：

·用户空间I/O软件——

用户程序中的I/O请求

运行时系统

·内核空间I/O软件——

与设备无关的I/O软件

设备驱动程序

中断服务程序

·I/O硬件

Linux系统中write操作的执行过程（例）p338/354

## **8.8 内核空间I/O软件**

8.8.1 与设备无关的I/O软件

1）设备驱动程序统一接口

2）缓冲区处理

缓冲区存在在多个层面和范围内，有用户空间缓冲区、内核空间缓冲区、设备缓冲区。

3）错误报告

设备无关I/O软件负责上层和部分通用错误处理

4）打开与关闭文件

5）逻辑块大小的处理

隐藏设备中数据块大小差异

8.8.2 设备驱动程序

程序直接控制方式：

内核驱动程序（即CPU）和外设完全串行，但用户进程可以不被阻塞（因为不需要，并不代表并行）

中断方式：

驱动程序阻塞进程，外设和CPU并行工作

DMA方式：

类似中断方式，只是中断间歇因DMA变长

8.8.3 中断服务程序

中断过程：p342/358

中断响应

中断处理——准备阶段、处理阶段、恢复阶段

# **第9章 并行处理系统**

## **9.1 并行处理系统概述**

9.1.1 并行处理的主要技术问题

1）互连

互连线路的结构

2）数据一致性

3）同步控制

共享/集中式同步控制

分布式同步控制

4）任务划分

5）并行程序设计

设计方法：

共享内存式（多线程并行）

消息传递式

MapReduce

6）资源调度和管理

7）容错性和安全性

8）性能分析与评估

加速比

9.1.2 并行处理系统的分类

1）Flynn分类

①单指令流单数据流SISD结构single instruction stream and single data stream

允许流水线方式和多模块交叉存储

②单指令流多数据流SIMD结构multiple data

多用于多媒体应用技术，因为存在大量数据需要进行反复相同的操作

·扩展原有的指令集以支持数据级并行（Intel系列）：

MMX，multi media extension多媒体扩展指令集（64位）

SSE，stream SIMD extension指令集（128位）

AVX，advanced vector extension全新的指令扩展集（256位）

③多指令流单数据流MISD结构

只是理论模型，现实不存在

④ 多指令流多数据流MIMD结构

目前最流行的并行处理方式

2）按地址空间的访问方式划分

①多计算机系统multicomputers

也称消息传递系统，因为不同的计算机使用不同的存储器。

②多处理器系统multiprocessor

SMP，shared memory multiprocessor，也称共享存储系统

3）按存储访问时间是否一致划分

①一致性内存访问UMA结构uniform memory access

所有处理器访问任何存储单元的时间相同

②非一致性内存访问NUMA结构non-uniform

CC-NUMA，cache-coherent NUMA：高速缓存一致的非一致性内存访问

4）按处理单元的位置及其互连方式划分

以下划分并非互斥关系：

①多核芯片multi-core

LLC，last-level cache

也称片级多处理器，chip-level multiprocessing，CMP

②众核many core

一般用于形容GPU芯片，包含的多为简单构造的核（达不到CPU的功能标准），但数量众多

③对称多处理器SMP，symmetric multiprocessor

同时也是UMA结构，对称即平等、相同地位

④大规模并行处理机MPP，massive parallel processing

用网络互连的对称多处理机系统

⑤集群cluster

紧密耦合的同构多计算机系统

⑥网格grid

松散耦合的异构多计算机系统

云计算cloud computing

## **9.2 多处理器系统**

9.2.1 UMA多处理器结构

①基于总线连接的UMA多处理器系统p354/370

②基于交叉开关网络的UMA多处理器系统p355/371

③基于多级交换网络的UMA多处理器系统p356/372

类似于路由，比交叉开关节省线路，但是也需要调度和阻塞算法

9.2.2 NUMA多处理器结构

对于大型系统和超多处理器结构，很难保证访存时间一致性。NUMA系统中，每个处理器都带有一个本地存储模块，视为本地内存，和cache不是一个概念。所有本地存储器统一编址，共同构成一个整体的逻辑存储器。

NC-NUMA，no cache NUMA：无cache的NUMA

CC-NUMA：带有cache

9.2.3 多处理器系统中的互连网络

p358/374

9.2.4 片级多处理器和多线程技术

·多核或片级多处理器只是广义上多处理器系统的一种特殊类型，而分布式云计算等也是多处理器系统的例子。现在通用的多核技术，实质上等同于微观意义的UMA。

指令级并行instruction-level parallelism，ILP

Tejas和Jayhawk：Intel的失败处理器研发项目，因为解决不了主频过大的散热问题，也标志着升频和集成技术的瓶颈。

多线程multithreading：一种共享单处理器功能部件的技术，包括3种实现方式

·细粒度多线程——每个时钟周期可切换线程

·粗粒度多线程——流水线阻塞时才会切换线程

·同时多线程simultaneous multithreading，SMT——可以说是超细粒度多线程，在同一周期内发射不同线程的指令

超线程hyper-threading：Intel处理器中的SMT技术

## **9.3 多计算机系统**

9.3.1 集群多计算机系统

采用局域网连接，通过消息传递进行通信，紧密耦合。适用于数据库系统、文件服务器、Web服务器等。

1）集群的互连与网卡p361/377

2）集群的交换机制p361/377

3）Google集群系统

OC48互联网连接的线路

OC12内部连接的传输线路

9.3.2 网格多计算机系统

中间件：网格中的一个特殊的协调性程序，负责管理用户认证、资源和作业分配等。

## **9.4 向量处理机和SIMD技术**

9.4.1 向量处理机

p364/380

9.4.2 Intel架构中的SIMD技术

p366/382

9.4.3 GPU架构

显存：显示存储器或GPU存储器，对于集成GPU来说，显存是从主存中分离的

GPGPU，general-purpose computing on GPU：通用GPU

NVIDIA Tesla GPU架构：

* + SPA，streaming processor array流处理器阵列
    - 线程处理器群thread processor cluster，TPC
      * 流多处理器SM
        + 标量流处理器stream processor，SP

也称线程处理器thread processor，TP

* + - * + SP各自的通用寄存器组register file，RF
        + 特殊功能部件special function unit，SFU
        + 双精度单元double precision unit，DPU
    - 共享的L1纹理cache和L2指令及常数cache
  + 存储控制系统
    - 光栅操作单元raster operation processor，ROP
    - 存储控制器
    - L2纹理cache

单指令多线程single instruction multiple thread，SIMT：NVIDIA GPU采用的不同于Intel CPU的SIMD方式。后者为数据级并行，SIMT同样是并行技术，但是形式上更像是超多线程并行，因为每个指令元操作都是由独立的SP核与RF执行的。同时，SIMT也必须考虑同步，SIMD不需要。

## **9.5 并行处理编程模式简介**

计算应用问题的类型：

数据密集型data-intensive

计算密集型computation-intensive

数据密集和计算密集混合型

1）共享存储变量方式shared memory variable

也称多线程并行程序设计方式，如Pthread，OpenMP

2）消息传递方式message passing

也称多进程并行程序设计方式，如消息传递接口message passing interface，MPI

3）MapReduce并行程序设计方式

由Google开发，面向大规模数据处理的并行处理模式和方法

4）CUDA并行程序设计方式computer unified device architecture

由NVIDIA开发，面向GPU+CPU系统，基于C++的可扩展并行编程模型

包含3个关键抽象概念：

层次结构线程组——网格grid，线程块thread block，线程

同步栅synchronization barrier

共享内存——局部存储器（显存中）、共享存储器（SM中）、全局存储器（显存中）

核函数kernel：可并行执行的函数

网格：执行相同核函数的一组线程块，相当于一个子处理任务

线程阵列collaborative thread arrays，CTA：线程块对应的执行模型

单程序多数据SPMD：CUDA在网格上的编程模型特点

计算调度器computer scheduler：负责计算任务的分配工作