第 2 章：计算机硬件基础　　作者：希赛教育软考学院　来源：希赛网　2014年05月19日

考点分析

第2章 计算机硬件基础

从历次考试试题来看，计算机硬件知识是软件设计师考试的一个重点。根据考试大纲，计算机硬件基础的考查知识包括以下几个方面：

（1）计算机系统的组成、体系结构分类及特性。

（2）存储系统。

（3）可靠性与系统性能评测基础知识。

2.1 考点分析

本节把历次考试中计算机硬件基础方面的试题进行汇总，得出本章的考点，如表2-1所示（括号中的数字表示知识点所考查的分数）。

表2-1 计算机硬件基础试题知识点分布

考试时间	分数	考查知识点
10.11	8	内存编址和容量（2）、中断响应（1）、流水线（1）、SIMD（1）、磁盘（1）、Cache（2）
11.05	9	数据的表示（2）、内存编址（1）、寻址方式（3）、系统可靠性（1）、磁盘（2）
11.11	5	多处理机（2）、可靠性（1）、流水线（2）
12.05	7	逻辑运算（1）、浮点数表示（1）、Cache（2）、可靠性（1）、流水线（1）、磁盘（1）
12.11	6	内存编址（1）、可靠性（1）、流水线（1）、时钟频率（2）、寻址方式（1）
13.05	5	控制器（1）、RISC（1）、流水线（1）、Cache（1）、可靠性（1）
13.11	6	寻址方式（2）、流水线（2）、存储器芯片的容量（1）、可靠性（1）
14.05	6	寻址方式（1）、流水线（1）、内存编址（1）、数据总线（1）、计算机分类（1）、可靠性计算（1）

根据表2-1,我们可以得出计算机硬件基础的考点主要有以下几个方面：

（1）计算机组成：包括计算机的基本组成、Flynn分类、RISC和CISC计算机的特点、多处理机、总线和接口等。

（2）数据运算：包括数据的表示（含浮点数的表示）、逻辑运算。

（3）寻址方式：包括指令的各种寻址方式。

（4）中断：主要考查中断的概念，以及中断响应的过程。

（5）存储体系：包括内存编址、内存容量的计算、Cache（高速缓冲存储器）、磁盘参数的计算。

（6）流水线：主要考查流水线的概念、性能，以及有关参数的计算。

（7）性能评估：主要考查系统可靠性的计算、时钟频率等。

对这些知识点进行归类，按照重要程度进行排列，如表2-2所示。其中的星号（*）代表知识点的重要程度，星号越多，表示越重要。

表2-2 计算机硬件基础各知识点重要程度

知识点	10.11	11.05	11.11	12.05	12.11	13.05	13.11	14.05	合计	比例	重要程度
计算机组成	1		2			2		2	7	13.46%	★★★
数据运算		2		2					4	7.69%	★★
寻址方式		3			1		2	1	7	13.46%	★★★
中断	1								1	1.92%	★
存储体系	5	3		3	1	1	1	1	15	28.85%	★★★★★
流水线	1		2	1	1	1	2	1	9	17.31%	★★★★
可靠性		1	1	1	3	1	1	1	9	17.31%	★★★★

在本章的后续内容中，我们将对这些知识点进行逐个讲解。

第 2 章：计算机硬件基础　作者：希赛教育软考学院　来源：希赛网　2014年05月19日

计算机组成

2.2 计算机组成

对于本知识点的考查，主要掌握计算机的基本组成、Flynn分类、RISC和CISC计算机的特点、多处理机的关键特性等。

第 2 章：计算机硬件基础　作者：希赛教育软考学院　来源：希赛网　2014年05月19日

计算机的基本组成

2.2.1 计算机的基本组成

在一台计算机中，主要有6种部件，分别是控制器、运算器、内存储器、外存储器、输入设备和输出设备，它们之间的合作关系如图2-1所示。

（1）控制器（Control Unit）：是分析和执行指令的部件，也是统一指挥并控制计算机各部件协调工作的中心部件，所依据的是机器指令。控制器的组成包含程序计数器（PC）、指令寄存器（IR）、指令译码器、时序部件、微操作控制信号形成部件（PSW）和中断机构。

（2）运算器：也叫做算术逻辑单元（Arithmetic and Logic Unit,ALU），对数据进行算术运算和逻辑运算。通常由ALU（算术/逻辑单元，包括累加器、加法器等）、通用寄存器（不包含地址寄存器）、多路转换器、数据总线组成。

（3）内存储器（Memory或Primary storage,简称内存或主存）：存储现场操作的信息与中间结果，包括机器指令和数据。

（4）外存储器（Secondary storage或Permanent storage,简称外存或辅存）：存储需要长期保存的各种信息。

（5）输入设备（Input devices）：接收外界向计算机输入的信息。

（6）输出设备（Output devices）：将计算机中的信息向外界输送。

现在的控制器和运算器是被制造在同一块超大规模集成电路中，统称为中央处理器，即CPU（Central Processing Unit）。

Flynn的分类

2.2.2 Flynn的分类

1966年，Michael.J.Flynn提出根据指令流、数据流的多倍性特征对计算机系统进行了分类（通常称为Flynn分类法），有关概念的定义如下：

（1）指令流：指机器执行的指令序列。

（2）数据流：指由指令流调用的数据序列，包括输入数据和中间结果，但不包括输出数据。

（3）多倍性：指在系统性能瓶颈部件上同时处于同一执行阶段的指令或数据的最大可能个数。

Flynn根据不同的指令流和数据流组织方式，把计算机系统分成如下4类：

（1）单指令流单数据流（Single Instruction stream and Single Data stream,SISD）：SISD其实就是传统的顺序执行的单处理器计算机，其指令部件每次只对一条指令进行译码，并只对一个操作部件分配数据。流水线方式的单处理机有时也被当作SISD.

（2）单指令流多数据流（Single Instruction stream and Multiple Data stream,SIMD）：SIMD以并行处理机（阵列处理机）为代表，并行处理机包括多个重复的处理单元，由单一指令部件控制，按照同一指令流的要求为它们分配各自所需的不同数据。相联处理机也属于这一类。

（3）多指令流单数据流（Multiple Instruction stream and Single Data stream,MISD）：MISD具有n个处理单元，按n条不同指令的要求对同一数据流及其中间结果进行不同的处理。一个处理单元的输出又作为另一个处理单元的输入。这类系统实际上很少见到。有文献把流水线看作多个指令部件，称流水线计算机是MISD.

（4）多指令流多数据流（Multiple Instruction stream and Multiple Data stream,MIMD）：MIMD是指能实现作业、任务、指令等各级全面并行的多机系统。多处理机属于MIMD.当前的高性能服务器与超级计算机大多具有多个处理机，能进行多任务处理，称为多处理机系统，不论是大规模并行处理机或对称多处理机，都属于MIMD.

上一节 本书简介 下一节

并行处理

2.2.3 并行处理

本节主要介绍几种多处理机系统。

（1）超级标量处理机。在超级标量处理机中，配置了多个功能部件和指令译码电路，采取了多条流水线，还有多个寄存器端口和总线，因此可以同时执行多个操作，以并行处理来提高机器速度。它可以同时从存储器中取出几条指令同时送入不同的功能部件。超级标量机的硬件是不能重新安排指令的前后次序的，但可以在编译程序时采取优化的办法对指令的执行次序进行精心安排，把

能并行执行的指令搭配起来。

（2）超级流水线处理机。超级流水线处理机的周期比其他结构的处理机短。与超级标量计算机一样，硬件不能调整指令的执行次序，而由编译程序解决优先问题。

（3）超长指令字处理机。超长指令字处理机是一种单指令流多操作码多数据的系统结构，编译程序在编译时把这个能并行执行的操作组合在一起，成为一条有多个操作段的超长指令，由这条超长指令控制计算机中多个互相独立的功能部件，每个操作段控制一个功能部件，相当于同时执行多条指令。

（4）向量处理机。向量处理机是一种具有向量数据表示、并设置有相应的指令和硬件、能对向量的各个元素进行并行处理的计算机。当进行向量运算时，它的性能要比大型机好得多。向量处理机有巨型计算机和向量协处理机（或称为数组处理机）两种类型，巨型计算机能对大量的数据进行浮点运算，同时它还是可以进行标量计算和一般数据处理的通用计算机。向量处理机一般采用流水线工作，当它处理一条数组指令时，对数组中的每个元素执行相同的操作，而且各元素间是互相无关的，因此流水线不会阻塞，能以每个时钟周期送出一个结果的速度运行。为了存储系统能及时提供数据，向量处理器配有一个大容量的、分成多个模块交错工作的主存储器。为了提高运算速度，在向量处理机的运算部件中可采用多个功能部件，例如向量部件、浮点部件、整数运算部件和计算地址用的地址部件。向量协处理机是专门处理浮点和向量运算的数组处理机，它连接到主机总线上。

（5）多处理机系统。多处理机具有两个或两个以上的处理机，共享输入/输出子系统，在操作系统统一控制下，通过共享主存或高速通信网络进行通信，协同求解一个个复杂的问题。多处理机通过利用多台处理机进行多任务处理来提高速度，利用系统的重组能力来提高可靠性、适应性和可用性。多处理机结构多处理机具有共享存储器和分布存储器两种不同的结构。具有共享存储器的多处理机中，程序员无数据划分的负担，编程容易；系统处理机数目较少，不易扩充。具有分布式存储器的多处理机结构灵活；容易扩充；难以在各个处理单元之间实现复杂数据结构的数据传送；任务动态分配复杂；现有软件可继承性差，需要设计新的并行算法。多处理机系统属于MIMD系统，与SIMD的并行处理机相比，有很大的差别。其根源就在于两者的并行性的层次不同，多处理机要实现的是更高一层的作业任务间的并行。

（6）大规模并行处理机。并行处理机有时也称为阵列处理机，并行处理机使用按地址访问的随机存储器，以SIMD方式工作。主要用于要求大量高速进行向量矩阵运算的应用领域。并行处理机制并行性来源于资源重复，把大量相同的处理单元通过互联网连接起来，在统一的控制器控制下，对各自分配来的数据并行完成同一条指令所规定的操作。并行处理机有两种基本结构类型：采用分布式存储器的并行处理结构和采用集中式共享存储器的并行处理结构。分布式存储器的并行处理结构中，每一个处理机都有自己局部的存储器，只要控制部件将并行处理的程序分配至各处理机，它们便能并行处理，各自从自己的局部存储器中取得信息。而共享存储多处理机结构中的存储器是集中共享的，由于多个处理机共享，在各处理机访问共享存储器时会发生竞争。因此，需采取措施尽可能避免竞争的发生。大规模并行处理机（Massively Parallel Processor,MPP）是由众多的微处理器（从几百到上万）组成的大规模的并行系统。MPP的出现成为计算机领域中一个研发热点，被用作开发万亿次甚至更高速的巨型机的主要结构。MPP可以采用市场上的出售的RISC处理器，所以有很高的性价比。

（7）对称多处理机。对称多处理机（Symmetrical Multi Processor,SMP）目前也基于RISC微

处理器。它与MPP最大的差别在于存储系统。SMP有一个统一共享主存空间，而MPP则是每个微处理器都拥有自己的本地存储器。

精简指令系统计算机

2.2.4 精简指令系统计算机

RISC（Reduced Instruction Set Computer,精简指令系统计算机）是相对于传统的CISC（Complex Instruction Set Computer,复杂指令系统计算机）而言的。RISC不是简单地把指令系统进行简化，而是通过简化指令的途径使计算机的结构更加简单合理，以减少指令的执行周期数，从而提高运算速度。

在这个知识点，我们主要掌握RISC计算机的主要特点，列举如下：

（1）指令数量少：优先选取使用频率最高的一些简单指令以及一些常用指令，避免使用复杂指令。大多数指令都是对寄存器操作，对存储器的操作仅提供了读和写两种方式。

（2）指令的寻址方式少：通常只支持寄存器寻址方式、立即数寻址方式以及相对寻址方式。

（3）指令长度固定，指令格式种类少：因为RISC指令数量少，格式相对简单，其指令长度固定，指令之间各字段的划分比较一致，译码相对容易。

（4）只提供了Load/Store指令访问存储器：只提供了从存储器读数（Load）和把数据写入存储器（Store）两条指令，其余所有的操作都在CPU的寄存器间进行。因此，RISC需要大量的寄存器。

（5）以硬布线逻辑控制为主：为了提高操作的执行速度，通常采用硬布线逻辑（组合逻辑）来构建控制器。而CISC机的指令系统很复杂，难以用组合逻辑电路实现控制器，通常采用微程序控制。

（6）单周期指令执行：因为简化了指令系统，很容易利用流水线技术使得大部分指令都能在一个机器周期内完成。因此，RISC通常采用流水线组织。少数指令可能会需要多个周期执行，例如Load/Store指令因为需要访问存储器，其执行时间就会长一些。

（7）优化的编译器：RISC的精简指令集使编译工作简单化。因为指令长度固定、格式少、寻址方式少，编译时不必在具有相似功能的许多指令中进行选择，也不必为寻址方式的选择而费心，同时易于实现优化，从而可以生成高效率执行的机器代码。

大多数RISC采用了Cache方案，而且有的RISC甚至使用两个独立的Cache来改善性能。一个称为指令Cache,另一个称为数据Cache.这样取指和读数可以同时进行，互不干扰。

在理论上来看：CISC和RISC都有各自的优势，不能认为RISC就好，CISC就不好。事实上，这两种设计方法很难找到完全的界线，而且在实际的芯片中，这两种设计方法也有相互渗透的地方，表2-3是两者的简单对比。

表2-3 CISC和RISC的简单对比

	CISC	RISC
指令条数	多	只选取最常见的指令
指令复杂度	高	低
指令长度	变化	短、固定
指令执行周期	随指令变化大	大多在一个机器同期完成
指令格式	复杂	简单
寻址方式	多	极少
涉及访问主存指令	多	极小，大部分只有存两条指令
通用寄存器数量	一般	大量
译码方式	微程序控制	硬件电路
对编译系统要求	低	高

第 2 章：计算机硬件基础　作者：希赛教育软考学院　来源：希赛网　2014年05月19日

总线和接口

2.2.5 总线和接口

总线就是一组进行互连和传输信息（指令、数据和地址）的信号线，它好比连接计算机系统各个部件之间的桥梁。另外，我们广义上通常也把AGP接口、USB接口等也称为AGP总线、USB总线。可以说总线在计算机中无处不在。

1.总线的分类

按总线相对于CPU或其他芯片的位置可分为内部总线（Internal Bus）和外部总线（External Bus）两种。在CPU内部，寄存器之间和算术逻辑部件ALU与控制部件之间传输数据所用的总线称为内部总线；外部总线是指CPU与内存RAM、ROM和输入/输出设备接口之间进行通信的通路。由于CPU通过总线实现程序取指令、内存/外设的数据交换，在CPU与外设一定的情况下，总线速度是制约计算机整体性能的最大因素。

按总线功能来划分又可分为地址总线、数据总线、控制总线三类。我们通常所说的总线都包括上述三个组成部分，地址总线用来传送地址信息，数据总线用来传送数据信息，控制总线用来传送各种控制信号。例如ISA总线共有98条线，其中数据线有16条、地址线24条、其余为控制信号线、接地线和电源线。

按总线在微机系统中的位置可分为机内总线和机外总线（Peripheral Bus）两种。我们上面所说的总线都是机内总线，而机外总线是指与外部设备接口相连的，实际上是一种外设的接口标准。如目前计算机上流行的接口标准IDE、SCSI、USB和IEEE 1394等，前两种主要是与硬盘、光驱等IDE设备接口相连，后面两种新型外部总线可以用来连接多种外部设备。

计算机的总线按其功用来划分主要有局部总线、系统总线、通信总线三种类型。其中局部总线是在传统的ISA总线和CPU总线之间增加的一级总线或管理层，它的出现是由于计算机软硬件功能的不断发展，系统原有的ISA/EISA等已远远不能适应系统高传输能力的要求，而成为整个系统的主要瓶颈。系统总线是计算机系统内部各部件（插板）之间进行连接和传输信息的一组信号线，例如ISA、EISA、MCA、VESA、PCI、AGP等。通信总线是系统之间或微机系统与设备之间进行通信的一组信号线。

2.总线标准

总线标准是指计算机部件各生产厂家都需要遵守的系统总线要求，从而使不同厂家生产的部件能够互换。总线标准主要规定总线的机械结构规范、功能结构规范和电气规范。总线标准可以分为正式标准和工业标准，其中正式标准是由IEEE等国际组织正式确定和承认的标准，工业标准是首先由某一厂家提出，然后得到其他厂家广泛使用的标准。

3.接口的分类

根据外部设备与I/O模块交换数据的方式，系统接口可以分为串行接口和并行接口两种。串行接口一次只能传送1位信息，而并行接口一次就可传送多位信息（一般为8的倍数）。串行通信又可分为异步通信方式和同步通信方式两种。并行接口数据传输率高，控制简单，通常用于高速数据通道接口；但是所需连线很多，不适于远距离传送。串行通信连线少，适于长距离传送；但是控制复杂而且传输速度较慢。

4.常见接口

常见的设备接口有以下几种：

（1）ST506.主要用于温盘，结构简单，只完成磁盘信息的读写放大，把数据的编码解码、数据的格式转换等功能都留给I/O模块处理。其传输速率为5Mbps~7Mbps,最多可支持2个硬盘，最大支持盘空间为150MB.

（2）ESDI.一种通用的标准接口，不仅适用于小型温盘，还适用于磁带机和光盘存储器。该接口除了完成信息的读写放大外，还要完成数据的编码解码。数据传输率5Mbps~10Mbps,最多可支持4个硬盘，硬盘空间最大可达600MB.

（3）IDE.IDE是最常用的磁盘接口，分为普通IDE和增强型IDE（EIDE）接口。普通IDE数据传输率不超过1.5Mbps,数据传输宽度为8位，最多可连接4个IDE设备，每个IDE硬盘容量不超过528MB.EIDE的传输率有UDMA-33、UDMA-66、UDMA-133三种，数据传输率可达12Mbps~18Mbps,数据传输宽度32位，最多可连接4个IDE设备，每个IDE硬盘可超过528MB.

（4）SCSI.数据宽度为8位、16位和32位，是大容量存储设备、音频设备和CD-ROM驱动器的一种标准。SCSI接口通常被看作是一种总线，可用于连接多个外设，这些SCSI设备以雏菊链（Mode daisy chain）形式接入，并被分配给惟一的ID号（0~7），其中7号分配给SCSI控制器。某些SCSI控制器可以提供多达35个SCSI通道。SCSI设备彼此独立运作，相互之间可以交换数据，也可以和主机进行交互。数据以分组消息的形式进行传输。最初的SCSI标准（目前又称为SCSI Ⅰ）的最大同步传输速率为5Mbps,后来的SCSI II规定了2种提高速度的选择。一种为提高数据传输的频率，即Fast SCSI,由于频率提高了一倍，即使数据通路仍和SCSI I同为8位宽，其最大同步传输速率也提高了一倍，达10Mbps.另一种提高速度的选择是传输频率提高一倍的同时也增大数据通路的宽度，由8位增至16位，这就是Wide SCSI,其最大同步传输速率为20Mbps.

（5）PCMCIA.PCMCIA是一种广泛用于笔记本电脑的接口标准，体积小，扩展较方便灵活。最初PCMCIA主要用于笔记本电脑扩展内存，目前常用作一种存储器卡接口或进行传真、调制解调器功能扩展接口。现在用PCMCIA代表个人计算机储器卡国际协会，而PCMCIA接口更名为PC Card接口。PC Card接口具有以下特点：电源管理服务，允许系统控制PC Card的工作状态（开/关），支持3.3V/5V电压，可降低功耗，支持多功能卡、扩充卡的信息结构，以提高其兼容性，规定了直接内存访问规范，增加了一个32位的Card Bus接口。

（6）P1394.P1394是一种高速的串行总线，用以连接众多的外部设备。P1394有许多优于SCSI等其他外设接口的特点：数据传输率高、价格低且容易实现，所以不仅应用于计算机系统中，也广

泛用于消费类电子产品，诸如数码相机、VCD等。P1394的数据速度可达400Mbps,新的标准是800Mbps.P1394接口使用雏菊链式的设备连接方式，一个端口可以支持63个设备；而且使用桥互联的方式，以树形结构配置，可以支持的设备数高达1022.P1394支持设备的热插拔，即允许计算机在未关机带电的情况下插入或拔除所连接的外部设备而不会造成损害。

（7）USB.USB接口是一种串行总线式的接口，在串行接口中可达到较高的数据传输率，并且也允许设备以雏菊链形式接入，最多可连接127个设备。USB的最大特点是允许热插拔，目前在便携式计算机和台式计算机中已成为标准配置。许多数码相机、闪存、视频摄像头以及打印机等都可通过USB口接入计算机。USB1.0的速度是1.2Mbps,USB2.0的速度达到了480Mbps.

上一节　本书简介　下一节

数据运算

2.3 数据运算

对于本知识点的考查，主要是数据的各种码制的表示和运算、浮点数的表示和运算，以及逻辑运算。

上一节　本书简介　下一节

各种码制

2.3.1 各种码制

本节主要掌握原码、反码、补码和移码的概念，以及各自的用途和优点。

1.原码

将最高位用做符号位（0表示正数，1表示负数），其余各位代表数值本身的绝对值的表示形式。这种方式是最容易理解的。例如：假设用8位表示一个数，则+11 的原码是00001011,-11 的原码是10001011.

直接使用原码在计算时会有麻烦。例如，$(1)_{10}+(-1)_{10}=0$.如果直接使用原码，则：

$(00000001)_2+(1000001)_2=(10000010)_2$

这样计算的结果是-2,也就是说，使用原码直接参与计算可能会出现错误的结果。所以，原码的符号位不能直接参与计算，必须和其他位分开，这样会增加硬件的开销和复杂性。

2.反码

正数的反码与原码相同。负数的反码符号位为1,其余各位为该数绝对值的原码按位取反。例如，-11的反码为11110100.

同样，对上面的加法，使用反码的结果是：

$(00000001)_2+(11111110)_2=(11111111)_2$

这样的结果是负0,而在人们普遍的观念中，0是不分正负的。反码的符号位可以直接参与计算，而且减法也可以转换为加法计算。

3.补码

正数的补码与原码相同。负数的补码是该数的反码加1,这个加1就是"补".例如，-11的补码为11110100+1 = 11110101

再次做以上的加法，是这样的：

$(00000001)_2+(11111111)_2=(00000000)_2$

这说明，直接使用补码进行计算的结果是正确的。

对一个补码表示的数，要计算其原码，只要对它再次求补。由于补码能使符号位与有效值部分一起参加运算，从而简化了运算规则，同时它也使减法运算转换为加法运算，进一步简化计算机中运算器的电路，这使得在大部分计算机系统中，数据都使用补码表示。

4.移码

移码又称为增码，移码的符号表示和补码相反，1表示正数，0表示负数。也就是说，移码是在补码的基础上把首位取反得到的，这样使得移码非常适合于阶码的运算，所以移码常用于表示阶码。

上一节　本书简介　下一节

作者：希赛教育软考学院　来源：希赛网　2014年05月19日

定点数和浮点数

2.3.2 定点数和浮点数

定点数和浮点数的区别在于如何对待小数点，在运算方式上也不相同，衡量一个计算机系统，定点运算和浮点运算是两个重要的指标。定点数的小数点是隐含的，固定在某个位置。如果该位置是在数的最低位之后，就是定点整数。定点数表示比较简单，运算规则也比较容易实现，但是当数值范围变化大时，使用定点数表示和运算就比较困难。为了表示更大范围的数值，可以使用浮点数表示法。

在表示一个很大的数时，我们常常使用一种称为科学计数法的方式：

其中M称为尾数，e是指数（阶码），R为基数。

浮点数就是使用这种方法来表示大范围的数，其中基数一般是2,8,16.而且对于特定机器而言，基数是固定不变的，所以在浮点数中基数并不出现。从这个表达式可以看出：浮点数表示的精读取决于尾数的宽度，范围取决于基数的大小和指数的宽度。

1.格式化数

使用格式化数是提高浮点数有效位的方法。格式化的意思是把尾数前面加0,同时修改指数，这样，在尾数位数固定的情况下，能提供最多的有效位来表示尾数。当指数小于能够表示的最小值

时，这个数称为机器零，此时会把尾数和指数同时清零。

2.定点数的溢出处理

计算机中通常使用补码进行计算。两个正数相加，如果结果的符号位变成了1,则表示有溢出；两个负数相加，如果结果的符号位变成了0,那么也意味着溢出。如果是正数和负数相加，则不会出现溢出的情况。

判断处理的方法可以再增加一个符号位，称之为第一符号位，原来那个符号位变成了第二符号位。两个符号位都参与计算，如果计算结果的两个符号位相同，表示没有溢出，如果不同，就表示出现了溢出。而第一符号位才是真正的符号。

也可以通过进位信号来判断，当结果的最高位和符号位的进位信号一致时（都有进位信号或都没有进位信号），则没有溢出，否则表示有溢出。

3.浮点数的运算

浮点数运算过程比定点数复杂，包括以下过程。

（1）对阶。首先计算两个数的指数差，把指数小的向指数大的对齐，并将尾数右移指数差的位数。这样，两个浮点数就完成了对阶的操作。可以看出：对阶的过程可能使得指数小的浮点数失去一些有效位。如果两个浮点数阶数相差很大，这个差大于指数小的浮点数的尾数宽度，则对阶后指数小的浮点数的尾数就变成了0,即当做机器零处理了。

（2）尾数计算。对阶完成后，两个浮点数的尾数就和定点数一样进行计算。

（3）结果格式化。尾数计算后，可能会产生溢出，此时将尾数右移，同时指数加1,如果指数加1后发生了溢出，则表示两个浮点数的运算发生了溢出。如果尾数计算没有溢出，则尾数不断左移，同时指数减1,直到尾数为格式化数。在这个过程中，指数小于机器能表达的最小数，则将结果置机器零，这种情况称为下溢。

上一节　本书简介　下一节

逻辑运算

2.3.3 逻辑运算

在计算机中，运算可以分为算术运算和逻辑运算。二进制数1和0在逻辑上可以代表"真"与"假"、"是"与"否"、"有"与"无".这种具有逻辑属性的变量就称为逻辑变量，逻辑变量之间的运算称为逻辑运算。

逻辑运算与算术运算的主要区别是：逻辑运算是按位进行的，位与位之间不像加减运算那样有进位或借位的联系。

逻辑运算主要包括4种基本运算，分别是逻辑加法（或运算）、逻辑乘法（与运算）、逻辑否定（非运算、否运算）和异或运算（半加运算）。

1.逻辑加法

逻辑加法通常用符号"+"或"∨"来表示。逻辑加法运算规则如下：

0+0=0, 0∨0=0

0+1=1, 0∨1=1

1+0=1, 1∨0=1

1+1=1, 1∨1=1

对于逻辑加法，在给定的逻辑变量中A和B中，只要有一个为1,其逻辑加的结果就为1.只有两者都为0时，逻辑加的结果才为0.

例如，某逻辑电路有两个输入端分别是X和Y,其输出端为Z.当且仅当两个输入端X和Y同时为0时，输出Z才为0,则该电路输出Z的逻辑表达式为X+Y.

2.逻辑乘法

逻辑乘法通常用符号"×"或"∧"或"·"来表示。逻辑乘法运算规则如下：

0×0=0, 0∧0=0, 0·0=0

0×1=0, 0∧1=0, 0·1=0

1×0=0, 1∧0=0, 1·0=0

1×1=1, 1∧1=1, 1·1=1

对于逻辑乘法，当参与运算的逻辑变量都同时取值为1时，其逻辑乘积才等于1.只要有一个逻辑变量为0,其结果就为0.

例如，用二进制数0与累加器X的内容进行与运算，并将结果放在累加器X中，一定可以完成对X的"清0"操作。

3.逻辑否定

逻辑非的规则为把变量取反，即：

$\bar{0}=1, \bar{1}=0$

4.异或运算

异或运算通常用符号"⊕"表示，其运算规则为：

0⊕0=0,0⊕1=1,1⊕0=1,1⊕1=0

也就是说，只有两个逻辑变量相异（一个为0,另一个为1），结果才为1.如果两个逻辑变量相同，则结果为0.

例如，在进行定点原码乘法运算时，乘积的符号位是被乘数的符号位和乘数的符号位通过异或运算来获得。因为原码的符号位表示数的正负，0表示正数，1表示负数。被乘数和乘数都是正数时，值为正数；都为负数时，值也为正数；只有当一个数是正数，另一个数是负数时，值才为负数。

我们在前面所举的例子都是一位数的操作，事实上，多位数进行逻辑运算时，也是按照"逐位运算"的规则进行的。例如，8位累加器A中的数据为FCH,若将其与7EH相异或，则累加器A中的数据为82H.因为将FCH和7EH转换为二进制数，得到11111100和01111110,根据异或的运算规则，可以得到10000010,然后将10000010转换成十六进制，得到82H.

上一节　本书简介　下一节

寻址方式

2.4 寻址方式

在计算机中，CPU都会定义出自己特定的指令系统，不过都遵循着统一的标准格式。指令的基本格式是由操作码和地址码两部分组成。操作码指出该指令要完成什么操作，地址码则是提供原始的数据。指令系统中定义操作码的方式可以分为规整型（即定长编码）和非规整型（变长编码）两种，如表2-4所示。

表2-4 指令系统中的操作码

编码方式	编码方式	平均码长
定长编码	采用相等码长，每个操作码的长度相等	码长≥$\log_2 X$ 其中 X 为操作码数。例如，14 个操作码，就应该是 4 位。因为 2^3=8，不够；2^4=16，多 2 个
变长编码	根据使用频度选择不同长度的编码	将长度分为几类，然后再对每类进行编码。 平均码长：将每个码长乘以频度，再累加其和

在指令系统中用来确定如何提供操作数或提供操作数地址的方式称为寻址方式（编址方式）。操作数可以存放在CPU中的寄存器（用寄存器名操作）、主存储器（指出存储单元地址）、堆栈（先进后出的存储机制，用栈顶指针SP来标出其当前位置）、外存储器或外围设备中。不过在运算时，数据均在主存储器中，操作数可以采用以下几种寻扯方式：

（1）立即寻址：直接给出操作数，而非地址。

（2）直接寻址：直接给出操作数地址或所在寄存器号（寄存器寻址）。

（3）间接寻址：给出的是指向操作数地址的地址，称之为间接寻址。

（4）变址寻址：给出的地址需与特定的地址值累加从而得出操作数地址，称之为变址寻址。

通过采用不同的寻址方式，能够达到缩短指令长度、扩大寻址空间和提高编程灵活性等目的。

例如，某计算机字长为16位，运算器为16位，有16个16位通用寄存器，8种寻址方式，主存容量为64K个字。指令中地址码由寻址方式字段和寄存器字段组成，采用单字长指令。则要表示8种寻址方式需要3位，要表示16个通用寄存器则需要4位，所以地址码一共需要7位，而又采用单字长指令，字长为16位，因此，操作码的位数就只有16-7=9位。也就是说，可以表示的指令种类是2^9条，即512条。因为每个寄存器是16位的，所以，可以表示的地址范围是2^{16}个字，即64K个字。

上一节　本书简介　下一节

中断

2.5 中断

在计算机中，I/O系统可以有三种不同的工作方式，分别是程序控制方式、程序中断方式和DMA工作方式。

1.程序控制方式

在程序控制方式模式下，输入输出完全由CPU控制，在整个I/O过程中CPU必须等待其完成，限制了CPU的高速能力。不过这种方式下，是由程序主动查询外设，完成主机与外设间的数据传送，方法简单，硬件开销小。

在这种方式下，I/O设备有2种编码方式：

（1）存储器映射：即I/O设备和主存储器统一编址，使用相同的机器指令来访问内存和外设，这种方式下，CPU是采用地址的不同来区分访问的是外设还是存储器的。

（2）独立编址：I/O设备和主存储器的地址空间相互独立，CPU使用专门的I/O指令来访问外设。

程序查询时，CPU既可以进行串行点名，也可以进行并行查询。

（1）串行点名：CPU依次对所有的外设进行查询，不过每次只查询一台。

（2）并行查询：对各个外设的状态位集中起来，由CPU通过一个专用的端口来读取，每次可以同时查询多个外设的状态。

2.程序中断方式

在I/O控制中引入中断，是为了解决"程序控制输入输出"方法中CPU低效等待的缺陷。采用该机制，它将无须定期查询I/O系统的状态，而可以抽身处理其他途径。当I/O系统完成后，则以中断信号通知CPU,之后CPU保存正在执行程序的现场（包括程序计数器PC,记住执行到哪个指令），然后转入I/O中断服务程序完成数据交换。而收到中断请求后，停止正在执行的代码，保存现场的时间称之为中断响应时间，这个时间应该尽可能短。

在系统中有多个中断源时，常见的处理方法包括如下所示：

（1）多中断信号线法：就是给每个中断源"拉一根电话线",做到专线专用。

（2）中断软件查询法：CPU收到中断后转到中断服务程序，由该程序来确认中断源。

（3）雏菊链法：硬件查询法，所有的I/O模块共享一条共同的中断请求线。

（4）总线仲裁法：一个I/O设备在发出中断请求前，必须先获得总线控制权。由总线仲裁机制来决定谁有权发出中断信号。

（5）中断向量表法：中断向量表用来保存各个中断源的中断服务程序的入口地址，当外设发出中断后，由中断控制器确定其中继号。

3.DMA方式

中断法虽然比程序控制法要更加有效，但由于都是由软件来完成工作的，因此难以满足高速传输的要求。而DMA方式则使用DMA控制器来控制和管理数据传输。DMAC与CPU共享系统总线，并且具有可以独立访问存储器的能力。

在进行DMA时，CPU放弃对系统总线的控制，改由DMAC控制总线；由DMAC提供存储器地址及必须的读写控制信号，实现外设与存储器的数据交换。

（1）向CPU申请DMA传送。

（2）获得CPU允许后，DMA控制器接管系统总线的控制权。

（3）在DMA控制器的控制下，在存储器和外设之间进行数据传送，在传送过程中无须CPU参与，开始时需要提供传送数据的长度和起始地址。

（4）传送结束后，向CPU返回DMA操作完成信号。

DMAC获取系统总线的控制权可以采用暂停方式（CPU交出控制权到DMA操作结束）、周期窃取方式（CPU空闲时暂时放弃总线时，插入一个DMA周期）、共享方式（CPU不使用系统总线时，由DMAC来进行DMA传输）。

第 2 章：计算机硬件基础 作者：希赛教育软考学院 来源：希赛网 2014年05月19日

存储体系

2.6 存储体系

计算机中，用于存放程序或数据的存储部件有CPU内部寄存器、高速缓冲存储器（Cache）、主存储器（内存储器、内存）、辅存（外存储器、外存）。它们的存取速度不一样，从快到慢依次为寄存器->Cache->内存->辅存。一般来讲，速度越快，成本就会越高。因为成本高，所以容量就会越小。严格来说，CPU内部寄存器不算存储系统。因此，在计算机的存储系统体系中，Cache是访问速度最快的层次。

在存储体系这个知识点，主要考查内存编址、内存容量的计算、Cache（高速缓冲存储器），以及磁盘参数的计算。

第 2 章：计算机硬件基础 作者：希赛教育软考学院 来源：希赛网 2014年05月19日

主存储器

2.6.1 主存储器

内存采用的是随机存取方式，因此简称为RAM（Random Access Memory）。如果计算机断电，则RAM中的信息会丢失。内存需对每个数据块进行编码，即每个单元有个地址，这就是所谓的内存编址问题。内存一般按照字节编址或按照字编址，通常采用的是十六进制表示。例如，假设某内存储器按字节编址，地址从A4000H到CBFFFH,则表示该存储器有（CBFFF-A4000）+1个字节（28000H字节），也就是163840个字节（160KB）。

要注意的是，编址的基础可以是字节，也可以是字（字是由1个或多个字节组成的），要算地址位数，首先应计算要编址的字或字节数，然后求2的对数即可得到。例如，上述内存的容量为160KB,则需要18位地址来表示（2^{17}=131072,2^{18}=262144）。

在内存这个知识点的另外一个问题，就是求存储芯片的组成问题。实际的存储器总是由一片或多片存储器配以控制电路构成的。设其容量为W?B,W是存储单元的数量，B表示每个单元由多少位组成。如果某一芯片规格为w×b,则组成W×B的存储器需要用（W/w）×（B/b）块芯片。例如：上述例子中的存储器容量为160KB,若用存储容量为32K×8bit的存储芯片构成，因为1B=8b（一个字节由8位组成），则至少需要（160K/32K）×（1B/8）=5块。

作者：希赛教育软考学院　来源：希赛网　2014年05月19日

高速缓冲存储器

2.6.2 高速缓冲存储器

Cache的功能是提高CPU数据输入输出的速率，突破所谓的"冯？诺依曼瓶颈",即CPU与存储系统间数据传送带宽限制。高速存储器能以极高的速率进行数据的访问，但因其价格高昂，如果计算机的内存完全由这种高速存储器组成则会大大增加计算机的成本。通常在CPU和内存之间设置小容量的高速存储器 Cache.Cache容量小但速度快，内存速度较低但容量大，通过优化调度算法，系统的性能会大大改善，仿佛其存储系统容量与内存相当而访问速度近似 Cache.

1.Cache基本原理

使用Cache改善系统性能的依据是程序的局部性原理。依据局部性原理，把内存中访问概率高的内容存放在Cache中，当CPU需要读取数据时就首先在Cache中查找是否有所需内容，如果有，则直接从Cache中读取；若没有，再从内存中读取该数据，然后同时送往CPU和Cache.如果CPU需要访问的内容大多都能在Cache中找到（称为访问命中），则可以大大提高系统性能。

如果以h代表对Cache的访问命中率（"1-h"称为失效率，或者称为未命中率），t_1表示Cache的周期时间，t_2表示内存的周期时间，以读操作为例，使用"Cache+主存储器"的系统的平均周期为t_3.则：

$$t_3 = t_1 \times h + t_2 \times (1-h)$$

系统的平均存储周期与命中率有很密切的关系，命中率的提高即使很小也能导致性能上的较大改善。

例如：设某计算机主存的读/写时间为100ns,有一个指令和数据合一的Cache,已知该Cache 的读/写时间为10 ns,取指令的命中率为98%,取数的命中率为95%.在执行某类程序时，约有1/5指令需要存/取一个操作数。假设指令流水线在任何时候都不阻塞，则设置Cache后，每条指令的平均访存时间约为：

（2%×100ns + 98%×10ns）+ 1/5×（5%×100ns + 95%×10ns） = 14.7ns

2.映射机制

当CPU发出访存请求后，存储器地址先被送到Cache控制器以确定所需数据是否已在Cache中，若命中则直接对Cache进行访问。这个过程称为Cache的地址映射（映像）。在Cache的地址映射中，主存和Cache将均分成容量相同的块（页）。常见的映射方法有直接映射、全相联映射和组相联映射。

（1）直接映射

直接映射方式以随机存取存储器作为Cache存储器，硬件电路较简单。直接映射是一种多对一的映射关系，但一个主存块只能够复制到Cache的一个特定位置上去。Cache的块号i和主存的块号j有函数关系：i=j%m（其中m为Cache总块数）。

例如：某Cache容量为16KB（即可用14位表示），每块的大小为16B（即可用4位表示），则说明其可分为1024块（可用10位表示）。则主存地址的最低4位为Cache的块内地址，然后接下来的中间10位为Cache块号。如果内存地址为1234E8F8H的话（一共32位），那么最后4位就是1000（对应十六进制数的最后一位"8"），而中间10位，则应从E8F（1110 1000 1111）中获取，得到10 1000 1111.因此，内存地址为1234E8F8H的单元装入的Cache地址为10 1000 1111 1000.

直接映射方式的优点是比较容易实现，缺点是不够灵活，有可能使Cache的存储空间得不到充分利用。例如：假设Cache有8块，则主存的第1块与第17块同时复制到Cache的第1页，即使Cache其他页面空闲，也有一个主存页不能写入Cache.

（2）全相联映射

全相联映射使用相联存储器组成的Cache存储器。在全相联映射方式中，主存的每一页可以映射到Cache的任一页。如果淘汰Cache中某一页的内容，则可调入任一主存页的内容，因而较直接映射方式灵活。

在全相联映射方式中，主存地址不能直接提取Cache页号，而是需要将主存页标记与Cache各页的标记逐个比较，直到找到标记符合的页（访问Cache命中），或者全部比较完后仍无符合的标记（访问Cache失败）。因此这种映射方式速度很慢，失掉了高速缓存的作用，这是全相联映射方式的最大缺点。如果让主存页标记与各Cache标记同时比较，则成本又太高。全相联映像方式因比较器电路难于设计和实现，只适用于小容量Cache.

（3）组相联映射

组相联映射是直接映射和全相联映射的折中方案。它将Cache中的块再分成组，通过直接映射方式决定组号，通过全相联映射的方式决定Cache中的块号。在组相联映射方式中，主存中一个组内的块数与Cache的分组数相同。

例如：容量为64块的Cache采用组相联方式映像，每块大小为128个字，每4块为一组。若主存容量为4096块，且以字编址，那么主存地址应该为多少位？主存区号（组号）为多少位？这样的题目，首先根据主存与Cache块的容量需一致，即每个内存块的大小也是128个字，因此共有128×4096个字（219个字），即主存地址需要19位。因为Cache的容量为64块，所以内存需要分为4096/64个组，即26,因此主存组号需6位。

在组相联映射中，由于Cache中每组有若干可供选择的页，因而它在映像定位方面较直接映像方式灵活；每组页数有限，因此付出的代价不是很大，可以根据设计目标选择组内页数。

3.淘汰算法

当Cache产生了一次访问未命中之后，相应的数据应同时读入CPU和Cache.但是当Cache已存满数据后，新数据必须淘汰Cache中的某些旧数据。最常用的淘汰算法有随机淘汰法、先进先出法（FIFO）和近期最少使用淘汰法（LRU）。其中平均命中率最高的是LRU算法。

4.写操作

因为需要保证缓存在Cache中的数据与内存中的内容一致，相对读操作而言，Cache的写操作比较复杂，常用的有以下几种方法。

（1）写直达（write through）。当要写Cache时，数据同时写回内存，有时也称为写通。

（2）写回（write back）。 CPU修改Cache的某一行后，相应的数据并不立即写入内存单元，而是当该行从cache中被淘汰时，才把数据写回到内存中。

（3）标记法。对Cache中的每一个数据设置一个有效位。当数据进入Cache后，有效位置1;而当CPU要对该数据进行修改时，数据只需写入内存并同时将该有效位清0.当要从Cache中读取数据时需要测试其有效位：若为l则直接从Cache中取数，否则从内存中取数。

磁盘

2.6.3 磁盘

磁盘是最常见的一种外部存储器，它是由1至多个圆形磁盘组成的，其结构如图2-2所示。

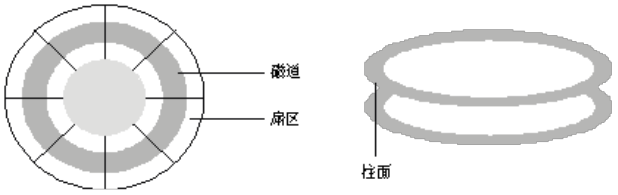

图2-2 磁盘主要术语示意图

磁盘的常见技术指标如下：

（1）计算磁道数：（外半径-内半径）×道密度×记录面数。

说明：硬盘的第一面与最后一面是起保护作用的，一般不用于存储数据，所以在计算的时候要减掉。例如，6个双面的盘片的有效记录面数是6×2-2=10.

（2）非格式化容量=位密度×3.14 ×最内圈直径×总磁道数。

说明：每个磁道的位密度是不相同的，但每个磁道的容量却是相同的。一般来说，0磁道是最外面的磁道，其位密度最小。

（3）格式化容量 = 每道扇区数×扇区容量×总磁道数。

（4）平均数据传输速率 = 每道扇区数×扇区容量×盘片转速。

说明：盘片转速是指磁盘每秒钟转多少转。

（5）存取时间 = 寻道时间 + 等待时间

说明：寻道时间是指磁头移动到磁道所需的时间；等待时间为等待读写的扇区转到磁头下方所用的时间。显然，寻道时间与磁盘的转速没有关系，而是取决于磁盘移动臂的速度。

上一节　本书简介　下一节

RAID

2.6.4 RAID

廉价磁盘冗余阵列（Redundant Array of Inexpensive Disks,RAID）技术旨在缩小日益扩大的CPU速度和磁盘存储器速度之间的差距。其策略是用多个较小的磁盘驱动器替换单一的大容量磁盘驱动器，同时合理地在多个磁盘上分布存放数据以支持同时从多个磁盘进行读写，从而改善了系统的I/O性能。最初，inexpensive一词主要针对当时另一种技术（single large expensive disk,SLED）而言，但随着技术的发展，SLED已是昨日黄花，RAID和non-RAID皆采用了类似的磁盘技术。因此 RAID现在代表独立磁盘冗余阵列（Redundant Array of Independent Disks），用"Independent"来强调RAID技术所带来的性能改善和更高的可靠性。

RAID机制中共分8个级别，RAID应用的主要技术有分块技术、交叉技术和重聚技术。

（1）RAID0级（无冗余和无校验的数据分块）：具有最高的I/O性能和最高的磁盘空间利用率，易管理，但系统的故障率高，属于非冗余系统，主要应用于那些关注性能、容量和价格而不是可靠性的应用程序。

（2）RAID1级（磁盘镜像阵列）：由磁盘对组成，每一个工作盘都有其对应的镜像盘，上面保存着与工作盘完全相同的数据拷贝，具有最高的安全性，但磁盘空间利用率只有50%.RAID1主要用于存放系统软件、数据以及其他重要文件。它提供了数据的实时备份，一旦发生故障所有的关键数据即刻就可使用。

（3）RAID2级（采用纠错海明码的磁盘阵列）：采用了海明码纠错技术，用户需增加校验盘来提供单纠错和双验错功能。对数据的访问涉及阵列中的每一个盘。大量数据传输时I/O性能较高，但不利于小批量数据传输。实际应用中很少使用。

（4）RAID3和RAID4级（采用奇偶校验码的磁盘阵列）：把奇偶校验码存放在一个独立的校验盘上。如果有一个盘失效，其上的数据可以通过对其他盘上的数据进行异或运算得到。读数据很快，但因为写入数据时要计算校验位，速度较慢。

（5）RAID5（无独立校验盘的奇偶校验码磁盘阵列）：与RAID4类似，但没有独立的校验盘，校验信息分布在组内所有盘上，对于大批量和小批量数据的读写性能都很好。RAID4和RAID5使用了独立存取技术，阵列中每一个磁盘都相互独立地操作，所以I/O请求可以并行处理。所以，该技术非常适合于I/O请求率高的应用而不太适合于要求高数据传输率的应用。与其他方案类似，RAID4、RAID5也应用了数据分块技术，但块的尺寸相对大一些。

（6）RAID6（具有独立的数据硬盘与两个独立的分布式校验方案）：在RAID6级的阵列中设置了一个专用的、可快速访问的异步校验盘。该盘具有独立的数据访问通路，但其性能改进有限，价格却很昂贵。

（7）RAID7:（具有最优化的异步高I/O速率和高数据传输率的磁盘阵列）：是对RAID6的改进。在这种阵列中的所有磁盘，都具有较高的传输速度，有着优异的性能，是目前最高档次的磁盘阵列。

（8）RAID10:（高可靠性与高性能的组合）：由多个RAID等级组合而成，建立在RAID0和RAID1基础上。RAID1是一个冗余的备份阵列，而RAID0是负责数据读写的阵列，因此又称为RAID 0+1.由于利用了RAID0极高的读写效率和RAID1较高的数据保护和恢复能力，使RAID10成为了一种性价比较高的等级，目前几乎所有的RAID控制卡都支持这一等级。

上一节　本书简介　下一节

流水线

2.7 流水线

流水线技术是通过并行硬件来提高系统性能的常用方法，它其实是一种任务分解的技术，把一件任务分解为若干顺序执行的子任务，不同的子任务由不同的执行机构来负责执行，而这些执行机

构可以同时并行工作。

在流水线这个知识点，主要考查流水线的概念、性能，以及有关参数的计算。

参数计算

2.7.1 参数计算

假定有某种类型的任务，共可分成n个子任务，每个子任务需要时间t,则完成该任务所需的时间即为n?t.若以传统的方式，则完成k个任务所需的时间是knt;而使用流水线技术执行，则花费的时间是（n+k-1）×t.也就是说，除了第一个任务需要完整的时间外，其他都通过并行，节省了大量的时间，只需一个子任务的单位时间就够了。

另外要注意的是，如果每个子任务所需的时间不同，则其速度取决于其执行顺序中最慢的那个（也就是流水线周期值等于最慢的那个指令周期），要根据实际情况进行调整。

例如：若指令流水线把一条指令分为取指、分析和执行三部分，且三部分的时间分别是取指2 ns,分析2 ns,执行1 ns.那么，最长的是2 ns.要注意的是，在设计流水线的周期时，是以执行时间最长的那一个部分为标准的。因此100条指令全部执行完毕需要的时间就是：（2+2+1）+（100-1）×2 = 203 ns.

另外，还应该掌握几个关键的术语：流水线的吞吐率、加速比。流水线的吞吐率（ThroughPut rate,TP）是指在单位时间内流水线所完成的任务数量或输出的结果数量。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称为流水线的加速比（Speedup Ratio）。

例如：在上述例子中，203 ns的时间内完成了100条指令，则从指令的角度来看，该流水线的吞吐率为（100×10^9）/203=4.93×10^8/s（1s=10^9ns），加速比为500/203=2.46（如果不采用流水线，则执行100条指令需要500 ns）。

影响流水性的主要因素

2.7.2 影响流水性的主要因素

流水线的关键在于"重叠执行",因此如果这个条件不能够满足，流水线就会被破坏。这种破坏主要来自三种情况：

1.转移指令

因为前面的转移指令还没有完成，流水线无法确定下一条指令的地址，因此也就无法向流水线

中添加这条指令。从这里的分析可以看出：无条件跳转指令是不会影响流水线的。

2.共享资源访问的冲突

也就是后一条指令需要使用的数据，与前一条指令发生的冲突，或者相邻的指令使用了相同的寄存器，这也会使得流水线失败。为了避免冲突，就需要把相互有关的指令进行阻塞，这样就会引起流水线效率的下降。一般来说，指令流水线级数越多，越容易导致数据相关，阻塞流水线。

当然，也可以在编译系统上进行设置，当发现相邻的语句存在资源共享冲突的时候，在两者之间插入其他语句，将两条指令进入流水线的时间拉开，以避免错误。

3.响应中断

当有中断请求时，流水线也会停止。流水线响应中断有两种方式：一种是立即停止现有的流水线，称为精确断点法，这种方法能够立即响应中断，缩短了中断响应时间，但是增加了中央处理器的硬件复杂度。还有一种是在中断时，在流水线内的指令继续执行，停止流水线的入口，当所有流水线内的指令全部执行后，再执行中断处理程序。这种方式中断响应时间较长，这种方式称为不精确断点法，优点是实现控制简单。

上一节　本书简介　下一节

非线性流水线

2.7.3 非线性流水线

在本节的介绍中，都是以线性流水线为例的，即假设系统中只存在一条流水线。如果系统中同时存在多条流水线，则需要进行变通处理。

例如：设指令由取指、分析、执行3个子部件完成，并且每个子部件的时间均为Δt。若采用常规标量单流水线处理机（即该处理机的度为1），连续执行12条指令，根据前面的介绍，则共需（12+3-1）×Δt=14.若采用度为4的超标量流水线处理机，连续执行上述12条指令，则因为同时运行4条流水线，平均每条流水线只需执行3条指令，因此只需（3+3-1）×Δt=5.

上一节　本书简介　下一节

性能评估

2.8 性能评估

在性能评估方面，主要考查系统可靠性、指令周期等。

可靠性相关概念

2.8.1 可靠性相关概念

与可靠性相关的概念主要有：平均无故障时间、平均故障修复时间以及平均故障间隔时间等。

1.平均无故障时间

可靠度为R（t）的系统的平均无故障时间MTTF定义为从t=0时到故障发生时系统的持续运行时间的期望值，计算公式如下：

$$MTTF = \int_0^{\infty} R(t)\,dt$$

如果 $R(t) = e^{-\lambda t}$，则 $MTTF = 1/\lambda$。λ 为失效率，是指器件或系统在单位时间内发生失效的预期次数，在此处假设为常数。

例如：假设同一型号的1000台计算机，在规定的条件下工作1000小时，其中有10台出现故障。这种计算机千小时的可靠度R为（1000-10）/1000=0.99.失效率为10/（1000×1000=1×10^{-5}）。因为平均无故障时间与失效率的关系为MTTF=1/λ,因此，MTTF=10^5小时。

2.平均故障修复时间

可用度为A（t）的系统的平均故障修复时间MTTR可以用类似于求MTTF的方法求得。设 $A_1(t)$ 是在风险函数Z（t）=0且系统的初始状态为1状态的条件下A（t）的特殊情况，则

此处假设修复率 $\mu(t) = \mu$（常数），修复率是指单位时间内可修复系统的平均次数，则：

$$MTTR = \int_0^{\infty} A_1(t)\,dt$$

MTTR=1/μ

3.平均故障间隔时间

平均故障间隔时间MTBF常常与MTTF发生混淆。因为两次故障（失败）之间必然有修复行为，因此，MTBF中应包含MTTR.对于可靠度服从指数分布的系统，从任一时刻到达故障的期望时间都是相等的，因此有：

MTBF = MTTR + MTTF

在实际应用中，一般MTTR很小，所以通常认为MTBF≈MTTF.

可靠性计算

2.8.2 可靠性计算

计算机系统是一个复杂的系统，而且影响其可靠性的因素也非常繁复，很难直接对其进行可靠性分析。但通过建立适当的数学模型，把大系统分割成若干子系统，可以简化其分析过程。

1.串联系统

假设一个系统由n个子系统组成，当且仅当所有的子系统都有能正常工作时，系统才能正常工作，这种系统称为串联系统，如图2-3所示。

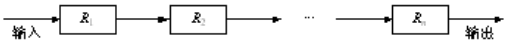

图2-3 串联系统

设系统各个子系统的可靠性分别用 $R_1, R_2, \ldots, R_n$ 表示，则系统的可靠性 $R = R_1 \times R_2 \times \ldots \times R_n$。

如果系统的各个子系统的失效率分别用 $\lambda_1, \lambda_2, \ldots, \lambda_n$ 来表示，则系统的失效率 $\lambda = \lambda_1 + \lambda_2 + \ldots + \lambda_n$ 。

2.并联系统

假如一个系统由n个子系统组成，只要有一个子系统能够正常工作，系统就能正常工作，如图2-4所示。

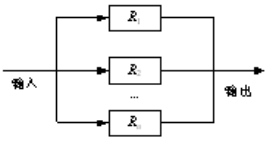

图2-4 并联系统

设系统各个子系统的可靠性分别用 $R_1, R_2, \ldots, R_n$ 表示，则系统的可靠性 $R = 1 - (1 - R_1) \times (1 - R_2) \times \ldots \times (1 - R_n)$

假如所有的子系统的失效率均为λ，则系统的失效率为μ：

$$\mu = \frac{1}{\frac{1}{\lambda}\sum_{j=1}^{n}\frac{1}{j}}$$

在并联系统中只有一个子系统是真正需要的，其余n-1个子系统称为冗余子系统，随着冗余子系统数量的增加，系统的平均无故障时间也增加了。

3.模冗余系统

m模冗余系统由m个（m=2n+1为奇数）相同的子系统和一个表决器组成，经过表决器表决后，m个子系统中占多数相同结果的输出作为系统的输出，如图2-5所示。

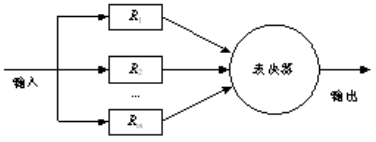

图2-5 模冗余系统

在m个子系统中，只有n+1个或n+1个以上子系统能正常工作，系统就能正常工作，输出正确结果。假设表决器是完全可靠的，每个子系统的可靠性为R_0 ,则m模冗余系统的可靠性为：

$$R = \sum_{i=n+1}^{m} C_m^j \times R_0^i \left(1 - R_0\right)^{m-i}$$

其中为从m个元素中取j个元素的组合数。

例如：某高可靠性计算机系统由图2-6所示的冗余部件构成。

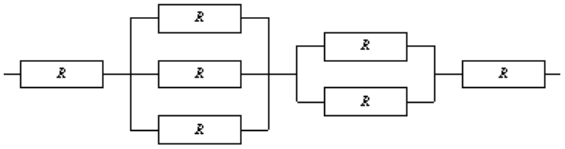

图2-6 某计算机系统

显然，该系统为一个串并联综合系统，我们可以先计算出中间2个并联系统的可靠度，根据并联

公式$R=1-(1-R_1)\times(1-R_2)\times\ldots\times(1-R_n)$，可得到3个部件并联的可靠度为$1-(1-R)^2$,2个部件并联的可靠度为$R=R_1\times R_2\times\ldots\times R_n$。

然后，再根据串联公式，可得到整个系统的可靠度为：$R\times(1-(1-R)^3)\times(1-(1-R)^2)\times R$。

容错

2.8.3 容错

提高计算机可靠性的技术可以分为避错技术和容错技术。避错是预防和避免系统在运行中出错。容错是指系统在其某一组件故障存在的情况下不失效，仍然能够正常工作的特性。简单地说，容错就是当计算机由于种种原因在系统中出现了数据、文件损坏或丢失时，系统能够自动将这些损坏或丢失的文件和数据恢复到发生事故以前的状态，使系统能够连续正常运行。容错功能一般通过冗余组件设计来实现。计算机系统的容错性通常可以从系统的可靠性、可用性和可测性等方面来衡量。

冗余技术是计算机容错技术的基础，一般可分为下列几种类型。

（1）硬件冗余：以检测或屏蔽故障为目的而增加一定硬件设备的方法。

（2）软件冗余：为了检测或屏蔽软件中的差错而增加一些在正常运行时所不需要的软件方法。

（3）信息冗余：在实现正常功能所需要的信息外，再添加一些信息，以保证运行结果正确性的方法。纠错码就是信息冗余的例子。

（4）时间冗余：使用附加一定时间的方法来完成系统功能。这些附加的时间主要用在故障检测、复查或故障屏蔽上。

在20世纪60年代主要利用双处理机或双机的方法来达到容错的目的。例如：把关键的元件（处理机、存储器等）或整个计算机设置两套：一套是系统运行时使用，另一套用于备份。根据系统的工作情况又可分为：热备份和冷备份两种。

（1）热备份（双重系统）：两套系统同时同步运行，当联机子系统检测到错误时，退出服务进行检修，而由热备份子系统接替工作。

（2）冷备份（双工系统）：处于冷备份的子系统平时停机或者运行与联机系统无关的运算，当联机子系统产生故障时，人工或自动进行切换，使冷备份系统成为联机系统。在运行冷备份时，不能保证从程序端点处精确地连续工作，因为备份机不能取得原来的机器上当前运行的全部数据。

20世纪在70年代中期出现了软件和硬件结构的容错方法。该方法在操作系统的层次上，支持联机维修，即故障部分退出后运行、进行维修并重新投入运行都不影响正在运行的应用程序。该结构特点是系统内包括双处理器、双存储器、双输入输出控制器、不间断工作的电源以及与之适应的操作系统等。因此上述硬件的任何一部分发生故障都不会影响系统的继续工作。系统容错是在操作系统控制下进行的，在每个处理机上都保持了反映所有系统资源状态的表格以及本机和它机的工作进程。

指令周期

2.8.4 指令周期

在这个考点中，我们主要掌握几个基本概念：时钟频率、时钟周期、机器周期、指令周期、指令执行速度。

时钟频率（时钟脉冲，主频）是计算机的基本工作脉冲，它控制着计算机的工作节奏。因此，计算机的时钟频率在一定程度上反映了机器速度。显然，对同一种机型的计算机，时钟频率越高，计算机的工作速度就越快。但是，由于不同的计算机硬件电路和器件的不完全相同，所以其所需要的时钟频率范围也不一定相同。相同频率、不同体系结构的机器，其速度和性能可能会相差很多倍。

时钟周期也称为振荡周期，定义为时钟频率的倒数。时钟周期是计算机中最基本的、最小的时间单位。在一个时钟周期内，CPU仅完成一个最基本的动作。

在计算机中，为了便于管理，常把一条指令的执行过程划分为若干个阶段，每一阶段完成一项工作。例如：取指令、存储器读、存储器写等，这每一项工作称为一个基本操作。完成一个基本操作所需要的时间称为机器周期。一般情况下，一个机器周期由若干个时钟周期组成。

指令周期是执行一条指令所需要的时间，一般由若干个机器周期组成。指令不同，所需的机器周期数也不同。对于一些简单的单字节指令，在取指令周期中，指令取出到指令寄存器后，立即译码执行，不再需要其他的机器周期。对于一些比较复杂的指令，例如转移指令、乘法指令，则需要两个或者两个以上的机器周期。

为了帮助读者搞清楚这些概念之间的关系，下面，我们通过一个例子来说明。

假设微机A和微机B采用同样的CPU,微机A的主频为20MHz,微机B为60MHz.如果两个时钟周期组成一个机器周期，平均三个机器周期可完成一条指令，则：

（1）微机A的时钟周期为1/（20M）=50ns.因为"两个时钟周期组成一个机器周期",则一个机器周期为2×50ns=100ns."平均三个机器周期可完成一条指令",则平均指令周期为3×100ns=300ns.也就是说，指令平均执行速度为1/（300ns）×3.33MIPS.其中MIPS的含义为"百万条指令/每秒".

（2）因为微机B的主频为60MHz,是微机A主频的60/20=3倍，所以，微机B的平均指令执行速度应该比微机A的快5倍，即微机B的指令平均执行速度为3.33×3×10MIPS。

考点分析