兰州理工大学计算机与通信学院

讲　稿

课　程　名　称　计算机组成原理

主　讲　教　师　谢鹏寿 包仲贤 任学惠

前　言

《计算机组成原理》是计算机科学与技术、软件工程、物联网工程、网络空间安全、数据科学与大数据技术等信息类本科专业的核心专业基础课，为了做好该门课程的教学工作，教学团队通过分工协作，完成的《计算机组成原理讲稿》总体情况如下：

文档结构	内容简介	执笔人
第 1 章 计算机系统概论	计算机系统的组成、性能指标和层次结构	谢鹏寿
第 2 章 运算方法和运算器	数据的表示方法，定点加减法、乘除法运算，定点运算器的组成，浮点加减法、乘除法运算，浮点运算器	包仲贤
第 3 章 多层次的存储器	存储器概述，SRAM，DRAM，存储容量扩充，双端口存储器，多模块交叉存储器，高速缓冲存储器 cache，磁盘存储设备	谢鹏寿
第 4 章 指令系统	指令系统的性能要求，指令格式，指令和数据的寻址方式，典型指令系统	谢鹏寿
第 5 章 中央处理器	CPU 的功能与组成，指令周期，时序产生器和控制方式，微程序控制器，硬布线控制器，流水 CPU	任学惠
第 6 章 总线系统	总线的概念和结构形态，总线接口，总线的仲裁、定时和数据传送模式，HOST 总线和 PCI 总线	包仲贤
第 7 章 外设与输入/输出系统	外设概述及其信息交换方式，程序中断方式，DMA 方式，通道方式	任学惠

统稿人：谢鹏寿

该《讲稿》是在以下参考资料的基础上，结合教师们以往的教学经验完成的：

白中英. 计算机组成原理（第 5 版•立体化教材）. 北京：科学出版社，2013.

蒋本珊. 计算机组成原理（第 3 版）. 北京：清华大学出版社，2013.

王换招. 计算机组成与设计. 北京：清华大学出版社，2013.

王爱英. 计算机组成与结构（第 5 版）. 北京：清华大学出版社，2013.

唐朔飞. 计算机组成原理（第 2 版）. 北京：高等教育出版社，2008.

William Stallings. Computer Organization and Architecture: Design for Performance. 计算机组织与结构：性能设计(第 7 版)(影印版). 北京：高等教育出版社，2009.

杜谨泽、姚斌老师参加了该门课程混合式教学改革的相关工作，在此对教学团队的辛勤劳动表示衷心的感谢！

《计算机组成原理》教学团队

2020 年 2 月于兰州

目　　录

第1章　计算机系统概论（1 学时）

1.1 计算机系统的组成

（PPT11 第 2 页）同学们，从今天开始，我们将一起学习《计算机组成原理》这门课程。《礼记·学记》中讲到：“学然后知不足，教然后知困。知不足，然后能自反也；知困，然后能自强也。”这就是教学相长的出处，指明教与学是相辅相成的，希望我们能够密切配合，把这门课程教好学好。

（PPT11 第 2 页）也许有不少同学会问：为什么要学《计算机组成原理》？从中能学到什么？回答这个问题要结合日常现象和专业要求，首先，（动画 1）我提一个最为普通的问题：我们使用计算机已经好长时间，究竟对其知之多少？同学们的反应存在着明显的差异。（动画 2）该图表示的是一般用户观察到的计算机硬件系统，包括主机箱、显示器、键盘等等，我觉得好多同学截止今天仍然处在这个认知水平。

（PPT11 第 3 页）而专业用户观察到的计算机硬件系统就要比一般用户深入一些，如图所示，不仅看到了显示器、键盘、鼠标等外围设备，还看到了主机箱中的主板、CPU、硬盘、显卡、内存条等各种部件。而在座各位同学的水平不该止步于此，应该向计算机设计者看齐，那么计算机设计者对计算机硬件系统的认知是怎样的呢？

（PPT11 第 4 页）计算机设计者观察到的计算机硬件系统已经深入到系统的逻辑结构和器件级层次，如图所示，一个计算机硬件系统由控制器、ALU、存储器和外设构成，而 ALU 包含了许多的寄存器，寄存器由许多的门组成，门由晶体管组成，晶体管可以采用 MOS 制成，也可以采用 TTL 制成。这种认知采用了自顶向下、逐步分解的方法，使得一个笼统的整机最后具化到物理级，使得设计、工艺、制造、使用、维护等环节的复杂工作尽可能的简单化，因此，计算机的发展和应用都非常惊人。（动画 1）可见，不同对象观察到的计算机硬件系统有所不同。（动画 2）所以，为了提高我们对计算机的认知能力和水平，我们需要学习计算机组成原理这门课程。

（PPT11 第 5 页）计算机的应用无处不在，速度更快，精度更高，它的未来超乎人类的想象。计算机应用的范围几乎涉及人类社会的所有领域。在科学计算、自动控制和测量、信息处理、商务处理、管理应用、教育和卫生、家用电器、人工智能等领域的应用成就最为突出。

（PPT11 第 6 页）截止 2018 年 12 月份，PC 机配置的 CPU Intel i9-9980XE 的基础主频为 3GHz，最大睿频为 4.4GHz，三级缓存为 24.75MB，使得系统的速度更快。

（PPT11 第 7 页）（动画 1）让人赞叹的莫过于超级计算机的速度，已经达到了每秒 20 亿亿次的“浮点数操作”，这就是 2018 年上半年、全球排名第一的 Summit，由美国研制而成，（动画 2）排名第二的是中国研制的“神威·太湖之光”，其速度达到了每秒 12.5 亿亿次的“浮点数操作”，（动画 3）排名第三的是美国研制的 Sierra。

（PPT11 第 8 页）排名第四的是中国研制的“天河二号”，其速度达到了每秒 3.4 亿亿次的“浮点数操作”，2013 年 6 月全球排名第一。这些高性能计算机不仅速度超快，而且精度更高。由此可见计算机的发展给人类带来了惊喜，与此同时，难道不能激发我们对它的好奇心吗？我们有必要从它的组成、逻辑结构和工作原理入手，一探究竟。（动画 1）因此，学习这门课程，不仅能够为所学专业的其他相关后续课程打下扎实基础，（动画 2）更是为了将我们在座的学生培养成将来能够为计算机的发展和应用做出更大贡献的专业技术人才，推进计算机技术领域、国家和人类社会的发展。

（PPT11 第 9 页）通过这门课程的理论教学，同学们掌握计算机系统概论、运算方法和运算器、多层次的存储器、指令系统、中央处理器、总线系统、输入输出系统等内容，再通过实践教学，同学们完成运算器实验、存储器实验、基本模型机设计与实现，使大家有以下收获：

第一，同学们能够运用计算机硬件的基本概念和基础知识，描述计算机硬件系统的基本组成、基本工作原理，培养学生关于计算机硬件组成和功能的知识表达能力；

第二，同学们能够综合运用计算机硬件的基本原理，通过计算、分析、证明、分解、设计、对比等方法，选择计算机硬件设计方法，比较计算机硬件性能指标，制定或评估计算机硬件设计方案，培养学生关于计算机硬件的综合分析和设计论证能力；

第三，同学们能够根据计算机硬件系统设计的需求，选择科学的原理和方法，制定或选择运算器、存储器、微控制器的实验方案，设计基本模型机和复杂模型机的实验方案，利用可行的实验环境正确采集、整理实验数据，对实验结果进行分析，验证、优化和实现实验方案，由此培养学生解决计算机硬件设计领域复杂工程问题的创新意识和研究实践能力。

（PPT11 第 10 页）下面我们一起学习计算机系统概论这一章的内容，主要包括计算机系统的组成、计算机的性能指标和计算机系统的层次结构。

（PPT11 第 11 页）同学们在大一学完计算机基础之后就应该知道：计算机是由硬

件、软件组成的复杂的、能自动地高速处理各种信息的电子设备。（动画 1）也就是说，完整的计算机应包括配套的硬件和软件系统。（动画 2）硬件系统包括主机和外部设备。（动画 3）软件系统包括系统软件和应用软件。（动画 4）主机主要包括运算器、控制器和主存储器。（动画 5）外部设备主要包括辅助存储器、输入设备和输出设备。（动画 6）系统软件主要包括操作系统、语言处理程序和数据库管理系统。（动画 7）应用软件包括专业应用软件和通用应用软件。

软件必须在硬件的支持下才能运行。软件的作用在计算机系统中越来越重要。我们把没有软件的计算机称为“裸机”。

硬件是计算机系统的物质基础是计算机的躯体，软件是计算机的头脑和灵魂，只有将两者有效地结合起来，计算机系统才能有生命力，整个计算机系统的好坏，取决于软硬件功能的总和。

（PPT11 第 12 页）现代计算机发展所遵循的基本结构形式始终是冯·诺依曼体系结构。这种结构的创新就是“计算机存储程序的思想”，其核心理念是“存储程序和数据，按地址访问，顺序执行”，（动画 1）该结构具有以下五个特点，（动画 2）第一个特点是计算机硬件主要由存储器、运算器、控制器、输入输出设备和总线五部分组成。（动画 3）各部件之间通过系统总线相连，（动画 4）输入输出设备和总线之间通过适配器相连。（动画 5）第二个特点是采用“存储程序控制”的工作方式。（动画 6）第三个特点是采用二进制形式表示数据和指令。（动画 7）第四个特点是程序由指令序列构成。（动画 8）第五个特点是计算机以 CPU 为中心，CPU 主要由运算器和控制器组成。

（PPT11 第 13 页）（动画 1）运算器是计算机中用于信息加工的部件，又称执行部件。对数据进行算术运算和逻辑运算。（动画 2）存储器是计算机的记忆部件，用于存放程序和数据。（动画 3）控制器是计算机中发号施令的部件，它控制计算机的各部件有条不紊地进行工作。（动画 4）输入设备的作用是把人们所熟悉的某种信息形式变换为机器内部所能接收和识别的二进制信息形式。输出设备的作用是把计算机处理的结果变换为人或其他机器所能接收和识别的信息形式。也就是说，输入输出设备是实现人机交互的部件。（动画 5）适配器的作用相当于一个转换器，它可以保证外围设备用计算机所要求的形式发送或接受信息。（动画 6）系统总线是构成计算机系统的骨架，是多个系统部件之间进行信息传送的公共通路。借助系统总线，计算机在各系统部件之间实现传送地址、数据和控制信息的操作。

（PPT11 第 14 页）讲到这里，我们需要弄清楚以下几个问题。第一个问题是：冯

•诺依曼体系结构的第三个特点是采用二进制形式表示数据和指令。为什么计算机采用二进制表示信息？第二个问题是：计算机凭借什么找到需要的程序或数据？第三个问题是：指令和数据均存放在内存中，计算机如何区分它们是指令还是数据？

（PPT11 第 15 页）早期的计算机采用十进制：0~9 表示信息，硬件及其控制很复杂，实现相当困难。（动画 1）这个难题倒逼业内学者和工程师们探寻信息表示的更好方法，终于在中国三千五百年前的太极图上获得了灵感和启发，这幅看似简单的太极图中只有黑色和白色两种颜色，（动画 2）其二值逻辑思想告诉学者和工程师们，可以从 10 个数中抽取 2 个出来用于表示计算机中的信息，（动画 3）而电路原理中正好有正脉冲和负脉冲两种可用的电脉冲信号，这为实现提供了可行途径，于是以“化繁为简”作为指导思想，从 0~9 十个数字中选取了容易理解、记忆、表达、实现的数字 0 和 1，在正逻辑条件下，用低电平（负脉冲）表示 0，用高电平（正脉冲）表示 1。使得计算机的应用更广、速度更快。（动画 4）运算器一次运算二进制数的位数越多速度越快，成本也越高。目前计算机的运算器长度一般是 8 位、16 位、32 位或 64 位。

（PPT11 第 16 页）如图所示的单元格相当于存储器的存储单元，其中存储着程序指令或者数据，譬如第一个单元中存储着零零零一、一零零一，第四个单元存储着一零零零、一一一零，（动画 1）为了便于访存，每个存储单元都有一个十六进制表示的单元编号，称为存储单元地址，（动画 2）譬如第一个单元地址为 2000H，依此类推，每个存储单元都有明确的编号。（动画 3）我们学习过 C 程序，现在举一个例子来说明计算机是如何找到所需数据的。（动画 4）int i 这个 C 语句在程序编译时，系统会给变量 i 分配存储单元，其单元地址&i=2000H，（动画 5）当程序执行需要数据时，系统会访问地址为 2000H 的存储单元，（动画 6）于是读出的数据为零零零一、一零零一，转换成十进制就是 25。可见，计算机凭借存储单元地址就能找到需要的程序或数据。

（PPT11 第 17 页）遵循冯.诺依曼体系结构的思想，指令在存储器中顺序存放，（动画 1）因此，控制器的基本任务是：按照程序所排的指令序列，先从存储器取出一条指令放到控制器中，对该指令的操作码由译码器进行分析判别，然后根据指令性质，执行这条指令，进行相应的操作。接着从存储器取出第二条指令，再执行这第二条指令。依次类推。（动画 2）我们借用冯.诺依曼体系结构示意图来演示取指令和取数据的基本情况。（动画 3）在取指阶段（取指周期），从内存读出的二进制信息被送往控制器，我们称其为指令字；（动画 4）在执行阶段（执行周期），从内存读出的二进制信息被送往运算器，我们称其为数据字。（动画 5）这样以来，虽然从表面形式上

看都是二进制编码，但因读出时间和送达部件不同，便能区分该信息是指令还是数据。我们把这种方法称为指令周期法或者时空法，也就是从时间（取指或执行）和空间（控制器或运算器）两个维度加以区分。在这里我们还要弄懂几个基本概念，（动画 6）取指周期是指取指令的一段时间。（动画 7）执行周期是指执行指令的一段时间。（动画 8）指令字是指计算机字为一条指令。（动画 9）数据字是指计算机字代表要处理的数据。

现在，我们一起回顾总结一下刚刚学过的内容。完整的计算机应包括配套的硬件和软件系统。计算机的硬件是由有形的电子器件等构成的，它包括运算器、存储器、控制器、适配器、输入输出设备。早期将运算器和控制器合在一起称为 CPU(中央处理器)。目前的 CPU 包含了存储器，因此称为中央处理机。存储程序并按地址顺序执行，这是冯•诺依曼型计算机的工作原理，也是 CPU 自动工作的关键。

计算机的软件是计算机系统结构的重要组成部分，也是计算机不同于一般电子设备的本质所在。计算机软件一般分为系统软件和应用软件两大类。系统软件用来简化程序设计，简化使用方法，提高计算机的使用效率，发挥和扩大计算机的功能和用途，它包括：操作系统、语言类程序、各种服务性程序和数据库管理系统。应用软件是针对某一应用课题领域开发的软件。

我们经常讲一台计算机的主频是多少、内存容量是多少、硬盘容量及其转速是多少、缓存是多少、相应速度是多少、机器字长是多少，等等等等。下一节我们一起学习完计算机的性能指标之后，就能更准确的理解其含义，并且能够计算相关的指标值，继而评价设计方案的合理性，或者选择更优的方案。

（本节讲授完毕）

1.2 计算机的性能指标

（PPT12 第 1 页）同学们，衡量计算机性能的主要指标比较多，我们在这里集中学习 12 个，包括吞吐量、响应时间、利用率、处理机字长、总线宽度、存储器容量、存储器带宽、主频/时钟周期、CPU 执行时间、CPI、MIPS、MFLOPS。

（PPT12 第 2 页）（动画 1）吞吐量表征一台计算机在某一时间间隔内能够处理的信息量，单位是字节/秒（B/S）。（动画 2）响应时间——表征从输入有效到系统产生响应之间的时间度量，用时间单位来度量，例如微秒（10^{-6}S）、纳秒（10^{-9}S）。

（PPT12 第 3 页）（动画 1）利用率表示在给定的时间间隔内，系统被实际使用的时间所占的比率，一般用百分比表示。（动画 2）处理机字长是指处理机运算器中

一次能够完成二进制数运算的位数。当前处理机的字长有 8 位、16 位、32 位、64 位。字长越长，表示计算的精度越高。

（PPT12 第 4 页）（动画 1）总线宽度一般指 CPU 中运算器与存储器之间进行互连的内部总线二进制位数。（动画 2）存储器容量是指存储器中所有存储单元的总数目，通常用 KB、MB、GB、TB 来表示。其中 1K=2^{10}，1M=2^{20}，1G=2^{30}，1T=2^{40}，1B=8 位（1 个字节）。存储器容量越大，记忆的二进制数越多。

（PPT12 第 5 页）（动画 1）存储器带宽表征了存储器的速度指标，是指单位时间内从存储器读出的二进制数信息量，一般用字节数/秒表示。（动画 2）主频/时钟周期是指 CPU 的工作节拍受主时钟控制，主时钟不断产生固定频率的时钟，主时钟的频率（f）叫 CPU 的主频。度量单位是 MHz（兆赫兹）、GHz（吉赫兹）。例如 Pentium 系列机为 60MHz ~ 266MHz，而 Pentium 4 提升至 3.6GHz。（动画 3）CPU 时钟周期（T）是主频的倒数，即 T=1/f，度量单位是微秒、纳秒。

（PPT12 第 6 页）（动画 1）CPU 执行时间表示 CPU 执行一段程序所占用的 CPU 时间。（动画 2）计算公式为：CPU 执行时间等于 CPU 时钟周期数乘以 CPU 时钟周期。（动画 3）CPI 表示每条指令周期数，即执行一条指令所需的平均时钟周期数。（动画 4）计算公式为：CPI 等于执行某段程序所需的 CPU 时钟周期数除以该程序包含的指令条数。

（PPT12 第 7 页）（动画 1）MIPS 表示每秒百万条指令数，（动画 2）计算公式为：MIPS 等于指令条数除以程序执行时间再除以十的六次方，也等于时钟频率除以 CPI 再除以十的六次方，在这个公式的分母中乘以十的六次方，就能将每秒的指令条数转换成百万条。（动画 3）由此可计算程序执行时间 Te 等于指令条数除以 MIPS 再除以十的六次方。（动画 4）MFLOPS 表示每秒百万次浮点操作次数。TFLOPS 表示每秒万亿次浮点操作次数，该技术指标一般在超级计算机中使用。（动画 5）MFLOPS 的计算公式为：MFLOPS 等于程序中的浮点操作次数除以程序执行时间再除以十的六次方。在这里需要说明一下：MIPS 是单位时间内的执行指令数，所以 MIPS 值越高说明机器速度越快。MFLOPS 是基于操作而非指令的，只能用来衡量机器浮点操作的性能，而不能体现机器的整体性能。

（PPT12 第 8 页）下面我们共同做一道例题，进一步理解计算机部分性能指标的含义和计算公式。题干告诉我们一些性能指标名称及其符号，要求写出执行该程序所需的 CPU 时间、每条指令的平均时钟周期数、每秒钟执行的百万条指令数和 CPU 时钟周期数。（动画 1）执行该程序所需的 CPU 时间 t_{CPU} 等于 CPU 时钟周期数 N_C 乘以

时钟周期 T，也等于每条指令的平均时钟周期数 CPI 乘以执行程序中的指令总数 I_N 再乘以时钟周期 T，考虑不同类型指令在程序中出现的次数，这个公式可以变换为 t_{CPU} 等于 i 指令所需的平均时钟周期数 CPI_i 乘以指令 i 在程序中执行的次数 I_i，然后将各类指令的这些乘积求累加和，再乘以时钟周期 T，由于时钟周期 T 和时钟频率 f 互为倒数，所以该公式还可以转换为 CPU 时钟周期数 N_C 除以时钟频率 f。（动画 2）每条指令的平均时钟周期数 CPI 等于 CPU 时钟周期数 N_C 除以执行程序中的指总数 I_N，而 N_C 等于各类指令所需的平均时钟周期数 CPI_i 和指令 i 在程序中执行的次数 I_i 乘积的累加和，所以该公式可以转换为各类指令所需的平均时钟周期数 CPI_i 和该类指令在程序中占的比例（I_i 除以 I_N）乘积的累加和。（动画 3）每秒钟执行的百万条指令数 MIPS 等于执行程序中的指令总数 I_N 除以执行该程序所需的 CPU 时间 t_{CPU} 再除以十的六次方，由第一个公式经过变换后，MIPS 也等于时钟频率 f 除以每条指令的平均时钟周期数 CPI 再除以十的六次方。（动画 4）CPU 时钟周期数 N_C 等于各类指令所需的平均时钟周期数 CPI_i 和指令 i 在程序中执行的次数 I_i 乘积的累加和，也等于每条指令的平均时钟周期数 CPI 乘以执行程序中的指令总数 I_N。

计算机的性能指标主要包括 CPU 性能指标、存储器性能指标和 I/O 吞吐率。掌握计算机的性能指标，对于评价计算机设计方案的合理性，或者选择更优的设计方案至关重要，请同学们课后多做习题、积极查阅相关资料，深入研究计算机性能优化的技术途径。如何了解乃至设计由硬件、软件组成的复杂的计算机，需要一种分级观点，这将是我们下一个知识点教学的主要任务。

（本节讲授完毕）

1.3 计算机系统的层次结构

（PPT13 第 1 页）同学们，从前面学习的内容，我们知道，计算机是由硬件、软件组成的复杂的、能自动地高速处理各种信息的电子设备。硬件是计算机系统的物质基础，软件是计算机系统的灵魂。硬件和软件是相辅相成的，不可分割的整体。（动画 1）作为一个复杂的整体，我们很难了解计算机的组成，至于设计计算机就会非常困难。怎么办呢？我们在生活中已经形成了一种思维方式，那就是将复杂的问题简单化，要做到这一点，可以采用分解的方法，结合计算机的特点，（动画 2）我们不直接称其为分解，而是称其为分级，也就是人为地将计算机看成是按功能划分的多级层次结构，这样以来，（动画 3）我们了解乃至设计计算机都会变得比较容易。

（PPT13 第 2 页）如图所示，（动画 1）我们将一个完整的计算机划分成六级。（动画 2）第 0 级是微程序设计级或逻辑电路级。**微程序由硬件直接执行**，如果某一

个应用程序直接用微指令来编写，那么可在这一级上运行应用程序。（动画 3）软硬件技术发展到今天，这一级相当于一个固件级。（动画 4）第 1 级是一般机器级，也称为机器语言级，它由**微程序**解释机器指令系统。（动画 5）这一级是一个硬件级。

（动画 6）第 2 级是操作系统级，它由**操作系统**程序实现。这些操作系统由机器指令和广义指令组成，广义指令是操作系统定义和解释的软件指令，所以这一级也称为混合级。（动画 7）第 3 级是汇编语言级，它给程序人员提供一种符号形式语言，以减少程序编写的复杂性。这一级由汇编程序支持和执行。如果应用程序采用汇编语言编写时，则机器必须要有这一级的功能；如果应用程序不采用**汇编语言**编写，则这一级可以不要。

（动画 8）第 4 级是高级语言级，它是面向用户的，为方便用户编写应用程序而设置的。这一级由各种高级语言**编译程序**支持和执行。（动画 9）第 5 级是应用语言级，为用户提供了解决问题的应用程序。第 2～第 5 级对程序的依赖性较强，因此，（动画 10）将其称为虚拟机器级。

在这个六级层次结构中，每一级上都能进行程序设计，且得到下面各级的支持。也就是说，下层是上层的基础，上层是下层的扩展。譬如：第 0 级是第 1 级的基础，第 1 级是第 0 级的扩展；第 1 级是第 2 级的基础，第 2 级是第 1 级的扩展。

（动画 11）第 0 级到第 2 级，编写程序采用的语言基本是二进制数字化语言，机器执行和解释容易。（动画 12）第 3 级到第 5 级，编写程序所采用的语言是符号语言，用英文字母和符号来表示程序，程序员不需要精通硬件，培养程序员要容易的多，甚至不了解硬件的人们都可以使用计算机。在座的大部分同学就得益于这一点。

（PPT13 第 3 页）请问同学们：计算机采用多级层次结构的好处是什么？（动画 1）采用计算机的多级层次结构有两个好处，（动画 2）第一个好处是，为了解计算机的组成，提供了一种好的结构和体制；（动画 3）第二个好处是，为设计良好的计算机系统结构，提供了一种分级的观点和帮助。

（PPT13 第 4 页）设计计算机的某个功能时，究竟采用硬件方案还是软件方案？这是值得我们考虑的一个问题。随着超大规模集成电路技术的发展和软件硬化的趋势，计算机系统的软、硬件界限已经变得越来越模糊。因为任何操作可以由软件来实现，也可以由硬件来实现；任何指令的执行可以由硬件完成，也可以由软件来完成。（动画 1）要弄清楚这个问题，我们需要从软件与硬件的逻辑等价性和方案选择的决定因素这两个方面入手进行分析。（动画 2）而软件与硬件的逻辑等价性主要体现在硬件软化和软件硬化上。（动画 3）硬件软化是指原来由硬件实现的操作改由软件来实现，

它可以增强系统的功能和适应性。（动画 4）软件硬化是指原来由软件实现的操作改由硬件来实现，它可以显著降低软件在时间上的开销。（动画 5）固件是软、硬件结合的产物，它是指那些存储在能永久保存信息的器件（如 ROM）中的程序，是具有软件功能的硬件。固件的性能指标介于硬件与软件之间，吸收了软、硬件各自的优点，其执行速度快于软件，灵活性优于硬件。（动画 6）对于某一机器功能采用硬件方案还是软件方案，取决于器件价格、速度、可靠性、存储容量和变更周期等因素。

总之，随着超大规模集成电路技术和计算机系统结构的发展，实体硬件机的功能范围在不断扩大。换句话说，第 0 级和第 1 级的边界范围，要向第 2 级乃至更高级扩展。这是因为容量大、价格低、体积小、可以改写的只读存储器提供了软件固化的**良好物质手段**。现在已经可以把许多复杂的、常用的程序制作成所谓固件。目前在一片硅单晶芯片上制作复杂的逻辑电路已经是实际可行的，这就为扩大指令的功能提供了**物质基础**，因此原本通过软件手段来实现的某种功能，现在可以通过硬件来直接解释执行。进一步的发展，就是设计所谓面向高级语言的计算机。这样的计算机，可以通过硬件直接解释执行高级语言的语句而不需要先经过编译程序的处理。因此传统的软件部分，今后完全有可能“固化”甚至“硬化”，计算机功能的固件化将成为计算机发展中的一个趋势。

计算机系统是一个由硬件、软件组成的多级层次结构，它通常由微程序级、一般机器级、操作系统级、汇编语言级、高级语言级组成，每一级上都能进行程序设计，且得到下面各级的支持。希望同学们在后续的学习中，要深刻领悟和运用这种观点。

（本节讲授完毕）

第2章　运算方法和运算器（5 学时）

2.1 数据的表示方法

2.1.1 数据格式

现在我们都知道计算机无所不在无所不能，它能处理各种各样的信息，比如数值数据，文字符号，形状，颜色，图形图像，音频视频等等。同时我们也知道计算机使用的是二进制，它的灵魂深处只有 0 和 1，中使用的是二进制，我们知道计算机中常用的数据表示格式有两种，一是定点格式，二是浮点格式。一般来说，定点格式容许的数值范围有限，但要求的处理硬件比较简单。而浮点格式容许的数值范围很大，但要求的处理硬件比较复杂。

【ppt p1】计算机中常用的数据表示格式有两种，一是定点格式，二是浮点格式。**【动画 1】**一般来说，定点格式容许的数值范围有限，但要求的处理硬件比较简单。**【动画 2】**而浮点格式容许的数值范围很大，但要求的处理硬件比较复杂。

1.定点数的表示方法

【ppt p2】定点表示：约定机器中所有数据的小数点位置是固定不变的。即固定小数点的位置。因为我们固定了小数点的位置，小数点就不再使用记号“.”来表示。首先，我们来看，定点数 $x = x0\ x1\ x2 \ldots xn$ 在定点机中表示如下(最高位 $x0$ 为符号位，0 代表正号，1 代表负号)：那么，小数点的位置固定在哪呢？**【动画 1】**有两个位置，第一个位置，将小数点的位置固定在符号位的后面，数据表示成**纯小数**，**【动画 2】**第二个位置，将小数点的位置固定在最低数值位的后面，数据表示成**纯整数**。

【ppt p3 动画 1】纯小数的表示范围为($x0\ x1\ x2 \ldots xn$ 各位均为 0 时最小；各位均为 1 时最大) $0 \leqslant |x| \leqslant 1-2^{-n}$ 。**【动画 2】纯整数**的表示范围为 $0 \leqslant |x| \leqslant 2^{n}-1$ 。**【动画 3】**目前计算机中多采用定点纯整数表示，因此将定点数表示的运算简称为**整数运算**。

【ppt p4 动画 1】定点数中大家需要注意，原码和补码表示的绝对值最大的负数时有区别。绝对值最大的负数：

【动画 2】定点整数：　原码：111……11　-（2^{n}-1）　补码：100……00　- 2^{n}

【动画 3】定点小数：　原码：111……11　-（1-2^{-n}）　补码：100……00　- 1

2.浮点数的表示方法

【ppt p5】电子的质量(9×10^{-28}克)和太阳的质量(2×10^{33}克)相差甚远，在定点计算机中无法直接来表示这个数值范围.要使它们送入定点计算机进行某种运算，必须对它们分别取不同的比例因，使其数值部分绝对值小于 1，即： $9 \times 10^{-28} = 0.9 \times 10^{-27}$

$2 \times 10^{33} = 0.2 \times 10^{34}$

这里的比例因子 10^{-27} 和 10^{34} 要分别存放在机器的某个存储单元中，以便以后对计算结果按这个比例增大。显然这要占用一定的存储空间和运算时间。因此得到浮点表示法如下:

浮点表示法：把一个数的有效数字和数的范围在计算机的一个存储单元中分别予以表示，这种把数的范围和精度分别表示的方法，数的小数点位置随比例因子的不同而在一定范围内自由浮动。

任意一个十进制数 N 可以写成 $N=10^{E}.M$

同样，在计算机中一个任意进制数 N 可以写成 $N=R^{e}.m$

【ppt p5 动画 1】R ：比例因子的**基数**，对于二进计数值的机器是一个常数，一般规定 R 为 2，8 或 16。**【动画 2】**e ：比例因子的指数，称为浮点的**指数**，是一个整数。**【动画 3】**m ：**尾数**，是一个纯小数。

【ppt p6】一个机器浮点数由阶码和尾数及它们各自的符号位组成，**【动画 1】阶码**：用整数形式表示，指明小数点在数据中的位置，决定了浮点数的表示范围。**【动画 6】尾数**：用定点小数表示，给出有效数字的位数决定了浮点数的表示精度。

【ppt p7】为便于软件移植，按照 IEEE754 的标准，32 位浮点数和 64 位浮点数的标准格式为：IEEE754 标准中，32 位的浮点数中规定:

【动画 1】S：浮点数的符号位，1 位，0 表示正数，1 表示负数。**【动画 2】**M:尾数，23 位，用小数表示，小数点放在尾数域的最前面。**【动画 3】**E：阶码，8 位阶符采用隐含方式，即采用移码方式来表示正负指数。移码方法对两个指数大小的比较和对阶操作都比较方便，因为阶码域值大者其指数值也大。

【ppt p8】规格化表示：为了提高数据的表示精度，当尾数 M 的值不为 0 时，其绝对值应≥0.5，即尾数域的最高有效位应为 1，M 用**原码表示时，若值为 1XX…XX，**称为浮点数的**规格化表示。**

【动画 1】规格化的浮点数在存储过程中，尾数域的最高有效位 1 省略，即存储 **1XX…XX 的红色部分，**尾数左移 1 位（小数点右移 1 位）低位补 1 位，同时修改指数的真值 e 使其减 1。指数 e 采用移码表示，我们写成 E，$E=2^{n}+e$。**【动画 2】**浮点数

的为 S，M，E，e（E 是 e 的移码），【动画 3】我们实际存储过程中尾数域的最高有效位 1 省略，尾数左移 1 位，【动画 4】指数减 1 变成了新的 S，M，E，e，

【ppt p9】 一个规格化的 32 位浮点数 x 的真值可表示为

$x=(-1)^{s}\times(1.M)\times 2^{E-127}$　　$e=E-127$

【ppt p10】一个规格化的 64 位浮点数 x 的真值为

$x=(-1)^{s}\times(1.M)\times 2^{E-1023}$　$e=E-1023$

【ppt p11】浮点数的规格化表示主要解决同一浮点数表示形式的不唯一性问题。

规定，当浮点数的尾数不为 0 时，其绝对值应≥0.5，即尾数域的最高有效位应为 1，否则尾数要进行左移或右移，使其变成这一表示形式。

【ppt p12 动画 1】IEEE754 标准 32 位浮点数有六中情况，用于表示规格化的浮点数和特殊数据：

1) 【动画 2，3】当 0＜E＜255 时，表示规格化浮点数：

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

2) 【动画 4，5】当 E = 0，且 M≠0 时，表示规格化浮点数：

$N=(-1)^{S}(1.M)2^{-127}=(-1)^{S}(0.M)\ 2^{-126}$

3) 【动画 6】当 E = 0，且 M = 0 时，表示浮点数零：(-1)S　0

4) 【动画 7，8】当 E = 255，且 M≠0 时，表示一个非数 NaN，NaN(Not-a-Number)可能是零除以零、求负数的平方根等情况产生的结果。

5) 【动画 9】当 E = 255，且 M = 0 时，表示一个无穷数：　$(-1)^{S}$　∞。

6) 【动画 10】当浮点数的尾数为 0，不论其阶码为何值，或者当阶码的值遇到比它能表示的最小值还小时，不管其尾数为何值，计算机都把该浮点数看成零值，称为**机器零**。

2.1.2 数的机器码表示

【ppt p1】在计算机中对数据进行运算操作时，符号位如何表示呢？是否也同数值位一道参加运算操作呢？为了妥善的处理好这些问题，就产生了把符号位和数字位一起编码来表示相应的数的各种表示方法，如原码、补码、反码、移码等。为了区别一般书写表示的数和机器中这些编码表示的数，通常将前者称为**真值**，后者称为**机器数**或**机器码**。

数的真值变成机器码时有四种表示方法：原码表示法、反码表示法、补码表示法移码表示码。其中移码主要用于表示浮点数的阶码E，以利于比较两个指数的大小和对阶操作。

1.原码表示法

【ppt p2 动画 1】*比较自然的表示法，最高位表示符号，0 为正，1 为负，*若定点小数的原码形式为 $x0\ x1\ x2\ldots xn$，则原码表示的定义是【动画 2】

【ppt p3 动画 1】若定点整数的原码形式为 $x_n\ x_{n-1}\ldots x_1\ x_0$，则原码表示的定义是**【动画 2】**

那么，**【动画 3，4】** $x=+0.1001$, 则 $[x]_{原}=?$　　$[x]_{原}=01001$

【动画 5，6】 $x=-0.1001$, 则 $[x]_{原}=?$　　$[x]_{原}=11001$

【动画 7，8】 $x=+1101$, 则 $[x]_{原}=?$　　$[x]_{原}=01101$

【动画 9，10】 $x=-1011$, 则 $[x]_{原}=?$　　$[x]_{原}=11011$

【ppt p4 动画 1】对于 0 来说，数学中可以从数轴的政府两个方向趋近于 0，因此 原码机器中往往有“+0”、“-0”之分，那么 $[+0]_{原}=?$　$[-0]_{原}=?$

【动画 2】根据原码定义

$[+0]_{原}=00000$

$[-0]_{原}=10000$

【动画 3】采用原码表示法优点是：简单易懂，但它的最大缺点是加法运算复杂。这是因为，当两数相加时，如果是同号则数值相加；如果是异号，则要进行减法。而在进行减法时还要比较绝对值的大小，然后大数减去小数，最后还要给结果选择符号。为了解决这些矛盾，人们找到了补码表示法。

2.补码表示法

【ppt p5 动画 0】-3 = +9，-7=+5 这两个等式成立。**【动画 1】**我这个结论是真理还是谬论？ **【动画 2】**假设现在的标准时间为 3 点正； 而有一只表目前指向 12 点，为了校准时间，可以采用两种方法：一是将时针沿着顺时针方向转 3 格；一是将时针逆时针方向转 9 格，我们定义逆时针方向为正方向，则顺时针方向为负方向，这两种方法都能对准到 3 点。我的结论-3 = +9 是成立的，同样-7=+5 也是成立的。**【动画 3】**这个结论的成立限于建立在这个表盘之上（采用 12 进制）；

在这表盘上，不管正转还是反转，超过 12 的，取余数；

我们在等式两边同时加一个数 A，在以 12 为模的情况下 A-3 = A+9，A-7=A+5 成立，则减 3 和加 9 是等价的，就是说 9 是(-3)对 12 的补码，减 7 和加 5 是等价的，就是说 5 是(-7)对 12 的补码。

【ppt p6 动画 1】从这里可以得到一个启示，就是负数用补码表示时，可以把减法转化为加法。这样，在计算机中实现起来就比较方便。

【ppt p7 动画 1】若定点小数补码形式为 $x0\ x1\ x2\ldots\ xn$，则补码表示的定义是**【动画 2】**。**【动画 3】***若定点小数补码形式为 $x_n\ x_{n-1}\ldots\ x_1\ x_0$，则补码表示的定义是*
【动画 4】

注意补码是分正负数的，**【ppt p8 动画 1】** 注意，0 的补码表示只有一种形式，那么，补码中多出一个编码 100…00。补码中 0 的表示是唯一的，定点小数中 100…00 表示-1 的补码，定点整数中 100…00 表示-2^n的补码。

但根据补码定义，定点小数求负数的补码要从 2 减去|x|。定点整数求负数的补码要从 2^{n+1} 减去|x|。求补码不方便。有没有简单的求补码的方法呢？答案是有，下面介绍的反码表示法可以解决负数的求补问题。

3. 反码表示法

【ppt p9 动画 1】所谓反码，就是二进制的各位数码 0 变为 1，1 变为 0。数字电路中非常容易实现。对定点小数，反码表示的定义为：**【动画 2】**其中 n 代表数的位数。**【动画 3】**对定点整数，反码表示的定义为：**【动画 4】**

在一些文献中，这种以 2 为基数的反码又称为 1 的补码。

【ppt p10 动画 1】对于正数，[x]原=[x]反=[x]补适用于定点整数和定点小数

x = + 0. x 1 x 2… x n，则 [x]原=[x]反=[x]补 = 0. x 1 x 2… x n

x = + x n x 2… x 1 x 0 ，则 [x]原=[x]反=[x]补 = 0 x n x 2… x 1 x 0

【ppt p10 动画 2】

【ppt p11 动画 1】、【动画 2】，对于 0，有$[+0]_{反}$和$[-0]_{反}$之分：$[+0]_{反} = 0.00...0$，$[-0]_{反} = 1.11...1$。**【动画 3】***反码**是为补码而设计的：*

对于正数，$[X]_{原}=[X]_{反}=[X]_{补}$

负数的补码，符号位置 1，其余各位 0 变 1，1 变 0，然后在最末位(2^{-n})上加 1，即该数的反码末尾加 1；

4.移码表示法

移码通常用于表示浮点数的阶码。由于阶码是个 n 位的整数，所以假定定点整数移码形式为 $x0\ x1\ x2\ldots xn$ 时，对定点整数，移码的定义是

【ppt p12 动画 2】 $[x]_{移}=2^n+x \qquad 2^n>x\geqslant -2^n$

【动画 3】若阶码数值部分为 5 位，以 x 表示真值，则

【动画 4】$[x]_{移}=2^5+x \qquad 2^5>x\geq -2^5$

【动画 5】例如，当正数 $x=+10101$ 时，$[x]_{移}=110101$；当负数 $x=-10101$ 时，$[x]_{移}=2^5+x=2^5-10101=0,01011$。移码中的逗号不是小数点，而是表示左边一位是符号位。

【ppt p13 动画 1】现在我们来看一下**移码和补码的关系**。【动画 2】显然，移码中符号位 $x0$ 表示的规律与原码、补码、反码相反。**移码和补码的关系：对同一个数值，其数值位完全相同，而符号位正好完全相反。**【动画 3】采用指数的实际值加上固定的偏移值的办法表示浮点数的指数，好处是可以用长度为 8 位的无符号整数来表示所有的指数取值，这使得两个浮点数的指数大小的比较更为容易，实际上可以按照字典序比较两个浮点表示的大小。

小结：上述四种数据机器表示法中，移码主要用于表示浮点数的阶码。由于补码表示对加减法运算十分方便，因此目前机器中广泛采用补码表示法。在这类机器中，数用补码表示，补码存储，补码运算。也有些机器，数用原码进行存储和传送，运算时改用补码。还有些机器在做加减法时用补码运算，在做乘除法时用原码运算。

2.2 定点加法、减法运算

2.2.1 定点加减法运算

一、 补码加法

现在计算机大多数都是补码计算机，我们知道补码是为了把减法用加法来实现二设计的机器码，从而实现加法打天下，用加法器来实现算数运算的乘法和除法运算。现在我们来学习补码加减法运算，首先在我来看补码加法运算。

【ppt p1 动画 1】补码加法的公式是$[x]_{补}+[y]_{补}=[x+y]_{补}$

【ppt p2 动画 1】假设采用定点小数表示，因此证明的先决条件是

$|x|<1,\ |y|<1,\ |x+y|<1$。

【ppt p2 动画 2】现分四种情况来证明。(1) $x>0,\ y>0$，则 $x+y>0$。

相加两数都是正数，故其和也一定是正数。正数的补码和原码是一样的，可得:

$[x]_{补}+[y]_{补}=x+y=[x+y]_{补}$

【ppt p3 动画 0】(2) $x>0$， $y<0$，则 $x+y>0$ 或 $x+y<0$。

相加的两数一个为正，一个为负，因此相加结果有正、负两种可能。根据补码定义， ∵ $[x]_{补}=x$， $[y]_{补}=2+y$

∴ $[x]_{补}+[y]_{补}=x+2+y=2+(x+y)$

【ppt p3 动画 1】 当 $x+y>0$ 时，$2+(x+y)>2$，进位 2 必丢失，又因($x+y$)>0， 故 $[x]_{补}+[y]_{补}=x+y=[x+y]_{补}$ (mod 2)

【动画 2】当 $x+y<0$ 时，$2+(x+y)<2$，又因($x+y$)<0，

故 $[x]_{补}+[y]_{补}=2+(x+y)=[x+y]_{补}$ (mod 2)

【ppt p4】(3) $x<0$， $y>0$，则 $x+y>0$ 或 $x+y<0$。

这种情况和第 2 种情况一样，把 x和 y的位置对调即得证。

【ppt p5 动画 1】(4) $x<0$， $y<0$，则 $x+y<0$。

相加两数都是负数，则其和也一定是负数。

∵ $[x]_{补}=2+x$， $[y]_{补}=2+y$

∴ $[x]_{补}+[y]_{补}=2+x+2+y=2+(2+x+y)$

【ppt p5 动画 2】上式右边分为”2”和(2 + x + y)两部分.既然(x + y)是负数，而其绝对值又小于 1，那么(2 + x + y)就一定是小于 2 而大于 1 的数，进位”2”必丢失。因(x + y)<0，**所以 2 + x + y 是[x + y]$_{补}$**等式成立。

$[x]_{补}+[y]_{补}=2+(x+y)=[x+y]_{补}$ (mod 2)

至此我们四个方面证明了，在模 2 意义下，任意两数的补码之和等于该两数之和的补码.这是补码加法的理论基础，其结论也适用于定点整数

【ppt p6 动画 1 】现在我们来看加法实例， $x=+1011$， $y=-0101$，求 $x+y$。**【动画 2 】**$x_{补}$=01011，$y_{补}$=11011，我用红色标识了符号位，蓝色为进位，**【动画 3 、4】** 列竖式进行加法运算。**【动画 5、6】**结果为 100110，其中符号位的进位舍弃。**【动画 7】**计算结果： x + y =00110 最高位为符号位。

【ppt p7 动画 1 】现在我们来看加法第二个实例， $x=+0.1001$， $y=+0.0101$，求 $x+y$。**【动画 2 】**$x_{补}$=01001，$y_{补}$=00101，红色标识了符号位，蓝色为进位。**【动画 3 、4】** 列竖式进行加法运算。**【动画 5、6】**结果为 01110。**【动画 7】**计算结果: x + y =01110 最高位为符号位。

【ppt p8】由以上两例看到，补码加法的特点：

1. 参加运算的操作数是补码，运算结果是和的补码；

2. 符号位和数值位一起参加运算；

3. 在模 2（2n+1）的意义下相加，即 2（2n+1）的位舍弃，要丢掉符号位的进位舍弃。（定点小数进位 2 舍弃，定点整数数进位 2n+1 舍弃）

二、补码减法

【ppt p9】负数的减法运算也要设法化为加法来做，其所以使用这种方法而不使用直接减法，是因为它可以和常规的加法运算使用同一加法器电路，从而简化了计算机的设计。数用补码表示时，减法运算的公式为$[x-y]_{补}=[x]_{补}-[y]_{补}=[x]_{补}+[-y]_{补}$。**【动画 1 】**要想证明减法公式成立，我们只要证明$[-y]_{补}=-[y]_{补}$，即可。

【ppt p10】我们运用加法公式即可证明，从$[x+y]_{补}$得$[y]_{补}$ 从$[x-y]$补推导出$[-y]_{补}$，从而得$[-y]_{补}+[y]_{补}=0$。因此补码减法公式成立。

【ppt p11】 我们证明补码减法公式成立，**【动画 1 】**把$[x]_{补}-[y]_{补}$转换成$[x]_{补}+[-y]_{补}$，**【动画 2 】**那么$[-y]_{补}$怎么求啊？**【动画 3】**大家要牢记这个方法，从$[y]_{补}$求$[-y]_{补}$的法则是：对$[y]_{补}$包括符号位“求反且最末位加 1”，即可得到。这句话中，我有没有说正负数啊？没有，也就是说从 $[y]_{补}$求$[-y]_{补}$正负数一视同仁。

【ppt p12 动画 1 】 现在我们来看加法第二个实例，$x=+1101$，$y=+0110$。**【动画 2 】**$x_{补}=01101$，$y_{补}=00101$，$[-y]_{补}=11010$ 依然是红色标识了符号位，蓝色为进位。**【动画 3 、4】** 列竖式进行加法运算，用的是$[x]_{补}+[-y]_{补}$。**【动画 5】**结果为 100101。符号位的进位为舍弃。**【动画 6】**计算结果：$x+y=00101$ 最高位为符号位。补码减法运算的特点：

1. 参加运算的操作数是补码，运算结果是和的补码；

2. 符号位和数值位一起参加运算；

3. 在模 2（2^{n+1}）的意义下相加，即 2（2^{n+1}）的位舍弃，要丢掉符号位的进位舍弃。（定点小数进位 2 舍弃，定点整数数进位 2^{n+1} 舍弃）

4. $-[y]_{补}$用$+[-y]_{补}$，转换成加法实现。

5. $[-y]_{补}$从$[y]_{补}$求得，方法是：

对$[y]_{补}$包括符号位“求反且最末位加 1”，即可得到。

2.2.2 溢出概念与检测方法

【ppt p1 动画 1、2】

[例] $x = +1011$， $y = +1001$，求 $x + y$。

[解:] $[x+y]_{补}=10100$

正数加正数怎么的负数了？

【ppt p1 动画 3、4】

[例] $x = -1101$， $y = -1011$，求 $x + y$。

$[x+y]_{补}=01000$

负数加负数怎么的正数了？很显然，这两个计算结果出错了！

【ppt p1 动画 5】

为什么？怎么会这样？

【ppt p2 动画 1】 之所以发生错误，是因为运算结果产生了溢出。

在定点小数机器中，数的表示范围为$|x|<1$. 在运算过程中如出现大于 1 的现象，称为“溢出”。在定点整数数机器中，数的表示范围为$|x|<2^n-1$. 在运算过程中如出现大于 2^n-1 的现象，称为“溢出”。

【动画 2】溢出就是运算结果超出了定点数的表示范围，两个正数相加，结果大于机器所能表示的最大正数，称为**上溢**。而两个负数相加，结果小于机器所能表示的最小负数，称为**下溢**。

在定点机中，正常情况下溢出是不允许的，如果法神溢出，运算器必须报告溢出状态。为了判断“溢出”是否发生，可采用两种检测的方法：单符号位法 和双符号位法。 ***【ppt p3】***首先，我们来看第一种溢出检测方法是采用单符号位法。

Cf 为符号位产生的进位; C_0 为最高数值位产生的进位，从这三个例子中可以看到，最高数值位和符号位同时产生进位（$C_f=C_o=1$），或都不进位（$C_f=C_o=0$）时，无溢出。

【ppt p4】当最高数值位产生进位而符号位无进位时（$C_f=1$，$C_o=0$），产生上溢；当最高数值位无进位而符号位有进位时（$C_f=0$，$C_o=1$），产生下溢。

由此我们知道，$C_f=C_o$ 无溢出，$C_f \neq C_o$ 发生溢出，***【ppt p5 动画 1】***根据异或运算的规律，两个输入端值相同，则输出为 0，两个输入端值不相同，则输出为 1，***【动画 2】***故溢出逻辑表达式为 $V = C_f \oplus C_o$，其中 C_f 为符号位产生的进位，C_o 为最高数值位产生的进位。

$V=1$（$Cf \neq C_0$），溢出　　$V=Cf \oplus C_0 = 1 \oplus 0 = 0 \oplus 1=1$

$V=0$（$Cf = C_0$），无溢出　　$V= Cf \oplus C_0 = 0 \oplus 0 = 1 \oplus 1=0$

$Cf=1$，$C_0=0$，上溢　　$Cf=0$，$Co=1$，下溢

在定点机中当运算结果发生溢出时，机器通过逻辑电路自动检查出溢出，并进行中断处理。

【ppt p6 动画 1】 第二种方法是采用双符号位法，称为“变形补码”或“模 2^{n+2} 补码”(定点小数采用“模 4 补码”)，从而可使模 2^{n+1} 补码所能表示的数的范围扩大一倍。变形补码定义为：　$[x]_{补} = 2^{n+2}+x \pmod{2^{n+2}}$。下式也同样成立：$[x]_{补}+[y]_{补}=[x+y]_{补} \pmod{2^{n+2}}$。

【动画 2】 为了得到两数变形补码之和等于两数之和的变形补码，同样必须：1. 高两位为符号位，两个符号位都看作数码一样参加运算；2. 两数进行以 2^{n+2} 为模的加法，即最高符号位上产生的进位要丢掉。**【动画 3】** 采用变形补码后，正数的符号位为 00，负数的符号位为 11，如果两个数相加后，其结果的符号位出现“01”或“10”两种组合时，表示发生溢出，最高符号位永远表示结果的正确符号。

【ppt p7 动画 1 】现在我们看两双符号位溢出检验的实例， $x = +1100$，$y = +1000$，求 $x + y$。$x_{补}=001100$，$y_{补}=001000$，**【动画 2 】**列竖式计算结果得 **010100**，**【动画 3 】** $V=S_{f1} \oplus S_{f2}=0 \oplus 1=1$ 发生溢出，双符号为 01 正溢出。

【ppt p8 动画 1 】现在我们看两双符号位溢出检验的另一实例 $x = -1100$，$y = -1000$，求 $x + y$。$x_{补}=110100$，$y_{补}=111000$，**【动画 2 】**列竖式计算结果得 **101100 进位舍弃。【动画 3 】** $V=S_{f1} \oplus S_{f2}=1 \oplus 0=1$ 发生溢出，双符号为 10 负溢出。

【ppt p9】由此可以得出如下结论：

1. 当以变形补码运算，运算结果的二符号位相异时，表示溢出；相同时，表示未溢出。故溢出逻辑表达式为 $V=S_{f1} \oplus S_{f2}$，其中 S_{f1} 和 S_{f2} 分别为最高符号位和第二符号位。此逻辑表达式可用异或门实现。

2. 变形补码补码相加的结果，不论溢出与否，最高符号位始终指示正确的符号。

2.2.3 基本的二进制加法/减法器

【ppt p1】进行两个二进制数字 A_i，B_i 加法运算时，还要考虑低位的进位输入 C_i，产生的结果为一个和输出 S_i，以及一个进位输出 C_{i+1}。这是一个一位二进制全加器真值表，在表中列出一位全加器进行加法运算的输入输出真值表，**【动画 1】**根据这个

真值表以及数字电路的知识，列出全加器布尔运算表达式并化简，可得到如下的逻辑方程：$S_i = A_i \oplus B_i \oplus C_i$，$C_{i+1} = A_iB_i + B_iC_i + C_iA_i$ = AiBi + Ci（Ai⊕Bi）。**【动画 2】**按此表达式组成的一位全加器示图。**【动画 3】**但是，我们还不满足，我们先让这运算器不仅实现加法运算，还可以实现减法运算。要求我们的 Bi 输入端即可输入$[B]_{补}$也可输入$[-y]_{补}$，那么我们 Bi 输入端进行改造，**【动画 4】**在每个 Bi 输入端连接一个异或门（包括符号位），异或门的一个为 0 时，输出时另一输入端的原码，异或门的一个为 1 时，输出时另一输入端的反码。

这是一个 *n* 位行波进位加减器。**【ppt p3 动画 1】**由图看到，*n* 个 1 位的全加器(FA)可级联成一个 *n* 位的行波进位加减器。低位的进位输出连接到高位的进位输入，就可以实现 N 位的加法运算。**【动画 2】**同时，我们在每个 Bi 输入端接入一个异或门，异或门的一个输入端为 **Bi**，另一个输入端方式控制输入线 M，用于控制加减运算. 当 M = 0 时，作加法(A + B)运算；当 M = 1 时，作减法(A − B)运算，A − B 运算转化成$[A]_{补} + [-B]_{补}$运算，求$[-B]_{补}$过程由 B 通过异或门按位取反来实现，末位加 1 来实现。【ppt p3 动画 2】为了实现末尾加 1，把 M 连接到最低位的进位输入端，当减法发运算时，M=1，实现 B 按位取反后的末尾加 1。**【动画 4】**采用单符号位法进行溢出检测，Cn、Cn − 1 经异或门产生溢出信号。当 Cn = Cn − 1 时，运算无溢出；而当 Cn≠Cn − 1 时，运算有溢出，经异或门产生溢出信号。**【动画 5，动画 6，动画 7】**这个行波进位的加减法器从操作数输入到产生运算结果所需的时间应包括**【动画 5】**最低位 B 输入端与或门，**【动画 6】**各位级联的进位链以及**【动画 7】**溢出检验电路的相应时间总和。采用 T 为单级逻辑电路的单位门延迟时间，*n* 位行波进位加减器响应时间值为（2n+9）T。从图中可以看出，运算位数越多，响应时间越长。

【ppt p4 动画 1】总结一下：行波进位的补码加法／减法器，将加减法用一个加法器实现。它的特点是：……。**【动画 2，3，4】**第一、M 为方式控制输入线：当 M = 0 时，作加法(A + B)运算；当 M = 1 时，作减法(A − B)运算。**【动画 5】**第二、作减法运算，A − B 运算转化成[A]补 + [− B]补运算，求补过程由 B + 1 来实现。**【动画 6】**第三、功能方式线 M 连接到起始进位输入端上，作减法时 M = 1，相当于在加法器的最低位上加 1。**【动画 7】**第四、采用单符号位法进行溢出检测；当 Cn = Cn − 1 时，运算无溢出；而当 Cn≠Cn − 1 时，运算有溢出，经异或门产生溢出信号。**【动画 8】**第五、运算位数越多，响应时间越长。

2.3 定点乘法、除法运算

2.3.1 原码乘法

1、定点乘法运算的运算方法

【ppt p1】在定点计算机中，两个原码表示的数相乘的运算规则是:乘积的符号位由两数的符号位异或运算得到，而乘积的数值部分则是两个正数相乘之积。设 n 位被乘数和乘数用定点整数表示(定点小数也同样适用) 【动画 1、2、3】

被乘数　　$[X]_{原} = X_f X_{n-1} \cdots X_1 X_0$

乘数　　$[Y]_{原} = Y_f Y_{n-1} \cdots Y_1 Y_0$

则乘积　　$[Z]_{原} = (X_f \oplus Y_f) + (X_{n-1} \cdots X_1 X_0)(Y_{n-1} \cdots Y_1 Y_0)$

其中，X_f为被乘数符号，Y_f为乘数符号。

人工运算时乘积符号的运算法则是：同号相乘为正，异号相乘为负。由于被乘数和乘数和符号组合只有四种情况($X_f Y_f$ = 00，01，10，11)，因此计算机中乘积的符号可按“异或”运算得到。

数值部分的运算方法与普通的十进制乘法类似，不过对于用二进制表达式的数来说，其乘法规则更为简单一些，0 乘任何数都得 0（0 × 0=0 × 1=0），1 乘一个数还得这个数（1×0=0，1×1=1）。

【ppt p2 动画 1】设 X = 1101，Y = 1011. 【动画 2、3】让我们先用人工方法求其乘积，其过程如下：……。运算的过程与十进制乘法相似:【动画 4】从乘数 Y 的最低位开始，【动画 5】若这一位为“1”，则将被乘数 X 写下；若这一位为“0”，则写下全 0。【动画 6、7】然后在对乘数 y 的最高一位进行乘法运算，其规则同上，不过这一位乘数的权与最低位乘数的权不一样，因此被乘数 X 要左移一位。以此类推，直到乘数所有位乘完为止，【动画 8】最后将它们统统加起来，变得到最后乘积 z，此处的加为竖直方向的加法运算。其规则可以归纳为：按位相乘，移位求和。

【ppt p3】设计定点乘法器时，人们习惯的算法对机器并不完全适用。原因有两个:【动画 1】1、机器通常只有 n 位长，两个 n 位数相乘，乘积可能为 2n 位。【动画 2】2、只有两个操作数相加的加法器难以胜任将 n 各位积一次相加起来的运算。

早期计算机中为了简化硬件结构，采用串行的 1 位乘法方案，即多次执行“加法—移位”操作来实现。这种方法需要器件数较少。然而速度太慢，目前已经被淘汰了，自从大规模集成电路问世以来，出现了各种形式的流水式阵列乘法器，它们属于并行

乘法器。

下面我们学习并行乘法器：要用硬件实现上述的乘法运算，应风两部分进行处理

2、不带符号的阵列乘法器

【ppt p3】设有两个不带符号的二进制整数：$A=a_{m-1}\ldots a_1a_0$, $B=b_{n-1}\ldots b_1b_0$

它们的数值分别为 a 和 b，即

在二进制乘法中，被乘数 A 与乘数 B 相乘，产生 $m+n$ 位乘积 P: $P=p_{m+n-1}\ldots p_1p_0$

【ppt p4】乘积 P 的数值为……。

在 m×n 位不带符号的阵列乘法器中，$m\times n$ 个被加数$\{a_ib_j|0\le i\le m-1$ 和 $0\le j\le n-1\}$可以用 $m\times n$ 个“与”门并行地产生。

【ppt p5 动画 1】实现这个乘法过程所需要的操作和人们的习惯方法非常类似:

上述过程说明了在 m 位乘 n 位不带符号整数的阵列乘法中，“加法—移位”操作的被加数矩阵。每一个部分乘积项(位积)a_ib_j 叫做一个被加数。**【动画 2】**解决这个问题的关键在框起来这一部分。因此，**【动画 3】**我们需要解决 两个问题：被加数的产生问题，一位求和问题。

【ppt p6 动画 1，2】这两个问题我们用两个不见来解决。1、在 m×n 位不带符号的阵列乘法器中，$m\times n$ 个被加数 a_ib_j $\{|0\le i\le m-1$ 和 $0\le j\le n-1\}$可以用 $m\times n$ 个“与”门并行地产生。**【动画 3，4】**2、移位求和问题，可以用 n 行，m-1 列全加器阵列排列，正确进位来实现。

显然，设计高速并行乘法器的基本问题，就在于缩短被加数矩阵中每列所包含的 1 的加法时间。

【ppt p7 动画 1】这种乘法器要实现 n 位×n 位时，需要 $n(n-1)$个全加器和 n^2 个“与”门。**【动画 2】**图中 FA 是一位全加器，所有被加数项的排列和 A×B=P 乘法过程中的被加数矩阵相同。由 n×n 个“与”门并行地产生。**【动画 3，4，5】**阵列全加器的斜线方向为进位输出，竖线方向为和输出。**【动画 6，7】**虚线围住的阵列中最后一行构成了一个行波进位加法器。**【动画 8，9】**该乘法器的总的乘法时间可以估算，考虑最坏情况下延迟途径，即是沿着矩阵最右边的对角线和最下面的一行，n 位×n 位不带符号的阵列乘法器总的乘法时间为：　　$t_m=(4n-2)\times 2T$

这样我们利用与门和行列排列的全加器实现了并行原码乘法运算，整个实现过程与人工习惯算法，有相同也有不同，希望同学们在深入理解一下。

2.3.2 补码乘法

1、对 2 求补器电路

【ppt p1 动画 1】我们先来看看算术运算部件设计中经常用到的求补电路。***【动画 2】***这是一个具有使能控制的二进制对 2 求补器电路图其逻辑表达式如下:***【动画 3，4】***在对 2 求补时，要采用按位扫描技术来执行所需要的求补操作。

【ppt p2】对 2 求补电路的功能是:***【动画 1，2】***当控制信号线 E 为“1”时，启动一个负数的对 2 求补的操作。***【动画 3，4】***当控制信号线 E 为“0”时，输出等于输入，我们理解为正数的求补。

【ppt p3 动画 1】我们可以利用符号位来作为控制信号。最右端的起始链式输入 C－1 必须永远置成“0”。***【动画 2】***对于一个 n + 1 位带符号的数 A = an…a1a0,***【动画 3】***要求确定它的补码形式。进行求补的方法:

（1） 符号位连接到控制信号线 E，C_{-1}=0;

（2） 正数 E=0，输出等于输入，整数的补码=原码。

（3） 负数 E=1，从数的最右端 a0 开始，由右向左，直到找出第一个 “1”，例如 ai = 1，则以该位 ai 为界，ai 以左的每一个输入位都求反，即 1 变 0，0 变 1。该位 ai 以右含该位不变。

例如，***【ppt p4 动画 1】***在一个 4 位的对 2 求补器中，输入正数的原码 01010，***【动画 2】***那么输出数应是 01010;***【ppt p5 动画 1】***输入负数原码 11010，***【动画 2】***那么输出数是 10110，其中从右算起的第 2 位，就是所遇到的第一个“1”的位置。

注意：对 2 求补电路输入一个数的原码，可以得到补码；输入一个数的补码，可以得到原码。

2、间接补码阵列乘法器

利用对 2 求补电路，可以实现(n + 1)×(n + 1)位带求补器的阵列乘法，逻辑方框图如图所示。通常，把包括这些求补级的乘法器称为符号求补的阵列乘法器，又称间接补码阵列乘法器。在这种逻辑结构中，共使用三个求补器。

【ppt p6 动画 1，2】两个算前求补器的作用是：将两个操作数 A 和 B 在被不带符号的乘法阵列(核心部件)相乘以前，先变成正整数。***【动画 3，4】***算后求补器的作用则是：当两个输入操作数的符号不一致时，把运算结果变成带符号的数。

设 A 和 B 为用定点表示的(n + 1)位带符号整数。在必要的求补操作以后，A 和 B

的码值输送给 n×n 位不带符号的阵列乘法器，并由此产生 2n 位真值乘积: A·B = P = p2n－1…p1p0。其中 P2n 为符号位 p2n = an ⊕ bn。

【ppt p7 动画 1】带求补级的阵列乘法器既适用于原码乘法，也适用于间接的补码乘法。不过在原码乘法中，算前求补和算后求补都不需要，因为输入数据都是立即可用的。而间接的补码阵列乘法所需要增加的硬件较多。为了完成所必需的乘法操作，时间大约比原码阵列乘法增加 1 倍。

3、直接补码阵列乘法器

【ppt p8 动画 1】补码包含符号位直接参加运算，涉及符号位参加加法运算时，它为负权，并且这个负权可能会沿着和输出或进位输出进行传递，也可能两个方向同时传递。**【动画 2】**此时我们用到一般化的全加器。

【动画 3】常规的一位全加器可假定它的 3 个输入和 2 个输出都是正权。这种加法器通过把正权或负权加到输入/输出端，可以归纳出四类加法单元 0-3 类全加器，每一类全加器用他负权输入的个数命名起类型。**【动画 4】** $[N]_{补}$ = 01101 具有的数值为:

图中给出了一般化全加器的类型、逻辑符号和输入输出情况。

【ppt p9】采用原码阵列乘法器的方法，利用混合型的全加器，就可以构成直接补码数阵列乘法器。5 位乘 5 位的直接补码阵列乘法器逻辑原理，设被乘数 *A* 和乘数 *B* 是两个 5 位的二进制补码数。图中可以看出，直接补码阵列乘法器和原码阵列乘法器结构一样，用到了 0 类，1 类和 2 类全加器。

现在我们可以用加法器阵列实现原码乘法和补码乘法的并行运算。

2.3.3 原码除法运算原理

【ppt p1 动画 1】两个原码表示的数相除时，商的符号由两数的符号按位相加求得，商的数值部分由两数的数值部分相除求得。

【ppt p2 动画 1】设有 *n* 位定点小数(定点整数也同样适用): 被除数 x，其原码为 $[x]_{原} = x_f . x_{n-1} \ldots x_1 x_0$

除数 y，其原码为 $[y]_{原} = y_f . y_{n-1} \ldots y_1 y_0$

【动画 2】则有商 $q = x/y$，其原码为

$[q]_{原} = (x_f \oplus y_f) + (0.x_{n-1} \ldots x_1 x_0 / 0.y_{n-1} \ldots y_1 y_0)$

商的符号运算 $q_f = x_f \oplus y_f$ 与原码乘法一样得到。商的数值部分的运算，实质上是两个正数求商的运算。

根据我们所熟知的十进制除法运算方法，很容易得到二进制数的除法运算方法，所不同的只是在二进制中，商的每一位不是“1”就是“0”，其运算法则更简单一些。

【ppt p3 动画 1】下面仅讨论数值部分的运算。设被除数 x = 0.1001，除数 y = 0.1011，模仿十进制除法运算，以手算方法求 $x \div y$ 的过程如下：

【动画 2】1. 判断 x 是否小于 y？现在 $x < y$，故商的整数位商“0”， x 的低位补 0，得余数 r_0。

2. 比较*余数*和除数（*右移一位的除数*），因*余数>除数*，表示够减，小数点后第一位商“1”，作*减法*得余数*新的余数*。

3. 比较*余数*和除数（*右移一位的除数*），因*余数<除数*，表示不够减，小数点后第一位商“0”，不作减法，*余数不变*。

求四位商，至此除法完毕。

但是，人会心算，一看就知道够不够减。但机器却不会心算，必须先作减法，若余数为正，才知道够减；若余数为负，才知道不够减。不够减时必须恢复原来的余数，以便再继续往下运算。这种方法称为**恢复余数法**。

恢复余数方便，只要当前的余数加上除数即可。但恢复余数致使除法进行过程的步数不固定，因此控制比较复杂。

现在使用的都是不恢复余数除法

【ppt p5 动画 1】分析恢复余数除法推导出来：

（1） 第 i-1 次求商的余数 r_{i-1} 时，下一次求商的方法是:$r_i=2r_{i-1}-|y|$；

（2） 当 ri>=0 时，商 1，下一步 $r_{i+1}=2r_i-|y|$；

（3） 当 ri<0 时，回复余数，ri+|y|本次商 0，下一步 2ri-|y|，$r_i+1=2(r_i+|y|)-|y|=2r_i+|y|$。

【动画 2】因此，当某次出上不够减时，本次商为 0，继续求下次商时，可以不必恢复余数，而是直接将负的差值左移后再加上除数，效果与与恢复余数后在左移一位减除数是等效的。这种方法称为不恢复余数除法。

【ppt p6 动画 1】不恢复余数的除法规则是：

当余数为正时，商为 1，做 $2r_i-|y|$；

当余数为负时，商为 0，做 $2r_i+|y|$；

【动画 2】不恢复余数的除法特点是：除法进行过程的步数固定，因此控制简单。

2.3.4 并行除法器

【ppt p1】和阵列乘法器非常相似，阵列式除法器也是一种并行运算部件，采用大规模集成电路制造.与早期的串行除法器相比，阵列除法器不仅所需的控制线路少，而且能提供令人满意的高速运算速度。

阵列除法器有多种多样形式，如不恢复余数阵列除法器，补码阵列除法器等等。

1、可控加法/减法(CAS)单元

【ppt p2 动画 1】为了实现加减交替运算，这里首先介绍可控加法/减法(CAS)单元，这是由 1 位全加器进行改造得到的。

【动画 2】这是一个 1 位全加器 FA，【ppt p1 动画 3】在它的一个输入端连接了异或门，异或门的 P 输入 0 时，输出时另一输入端的原码，异或门的 P 输入 1 时，输出时另一输入端 Bi 的反码。

【动画 4】可控加法/减法(CAS)单元输入端有四个：A_i，B_i，C_i，P

【动画 5】可控加法/减法(CAS)单元输出端有四个：S_i，B_i，C_{i+1}，P

【动画 6】其中 ***B_i，和 P 输入输出直通。***它将用于并行除法流水逻辑阵列中时，***B_i 输出用于进行除数移位操作***，P 用于控制加减法操作。

它将用于并行除法流水逻辑阵列中，它有四个输出端和四个输入端。当输入线 P = 0 时，CAS 作加法运算；当 P = 1 时，CAS 作减法运算。

【ppt p3 动画 1】当 P = 0 时，方程式(2.32)就等于式(2.23)，即得我们熟悉的一位全加器(FA)的公式：……。***【动画 2】***当 P = 1 时，则得求差公式：……。

【ppt p4】在减法情况下，输入 C_i 称为借位输入，而 C_{i+1} 称为借位输出。　　为说明 CAS 单元的实际内部电路实现，将方程式加以变换，可得如下形式：……。

2、不恢复余数的阵列除法器

【ppt p5】这是一个不恢复余数的阵列除法器，用若干个 CAS 单元，按行列排列起来实现原码不恢复余数除法。大家看他们之间的连接。我们用定点小数参加运算，x，y 是定点小数，且 $|x|\leq|y|$，求 x ÷ y，符号位单独运算 $q_f=(x_f \oplus y_f)$。

【动画 1】第 1 行 P=1 ，做减法，同一行首尾相连，最低位的输出连接到自己的进位输入，***【动画 2】***第二行以后，用上一行商的输出连接 P，根据上决定下一步的加减运算。***【动画 3】***以后的行 P 都如此连接。

【ppt p6 动画 1】接下来我们看，Bi 的输入输出，沿斜线方向连接到下一行的 Bi

输入，实现除数移位。

假定所有被处理的数都是正的小数。

在不恢复余数的除法阵列中，每一行所执行的操作究竟是加法还是减法，取决于前一行输出的符号与被除数的符号是否一致。当出现不够减时，部分余数相对于被除数来说要改变符号。这时应该产生一个商位“0”，除数首先沿对角线右移，然后加到下一行的部分余数上。当部分余数不改变它的符号时，即产生商位“1”，下一行的操作应该是减法。下图示出了 4 位除 4 位的不恢复余数阵列除法器的逻辑原理图。其中，被除数……、除数……、商……、余数……、字长 n + 1 = 4。

【ppt p7】由图看出，该阵列除法器是用一个可控加法/减法(CAS)单元所组成的流水阵列来实现的。推广到一般情况，一个(n + 1)位除(n + 1)位的加减交替除法阵列由(n + 1)^2 个 CAS 单元组成，其中两个操作数(被除数与除数)都是正的。

单元之间的互连是用 n = 3 的阵列来表示的。这里被除数 x 是一个 6 位的小数(双倍长度值)：　$x = 0.x_1 x_2 x_3 x_4 x_5 x_6$

它是由顶部一行和最右边的对角线上的垂直输入线来提供的。

除数 y 是一个 3 位的小数：　$y = 0.y_1 y_2 y_3$

它沿对角线方向进入这个阵列。这是因为，在除法中所需要的部分余数的左移，可以用下列等效的操作来代替：即让余数保持固定，而将除数沿对角线右移。商 q 是一个 3 位的小数：$q = 0.q_1q_2q_3$

它在阵列的左边产生。　　余数 r 是一个 6 位的小数：$r = 0.00r_3r_4r_5r_6$

它在阵列的最下一行产生。

【ppt p8】原码除法（加减交替运算规则）

（1）　初始操作做减法。最上面一行的控制线 P 固定置成 “1”

（2）　被除数（或余数）减除数，当余数为正时商上 1，余数左移减除数；当余数为负时商上 0，余数左移加除数；

（3）　原码运算时，商的符号运算 $q_f = x_f \oplus y_f$，商的值为 $q = |x|/|y|$，采用双符号位，运算中余数左移 n 位，则 $r=2^{-n}.[r]$

（4）　对于 n 位数除法需进行 n+1 次加法和 n 次移位。

【ppt p9】这是一个具体的例子

一个整数除法例子：X=0.101001, y=0.111, 求X/Y?

解：　[-Y]补=1.001

被除数X	0.101001	
减Y	1.001	
余数为负	1.110001	<0 ⟹ $q_0=0$
移位	1.10001	
加Y	0.111	
余数为正	0.01101	>0 ⟹ $q_1=1$
移位	0.1101	
减Y	1.001	
余数为负	1.1111	<0 ⟹ $q_2=0$
移位	1.111	
加Y	0.111	
余数为正	0.110	>0 ⟹ $q_3=1$

故得：　商$q=q_0.q_1q_2q_3=0.101$

余数$r=(0.00r_3r_4r_5r_6)=0.000110$

【ppt p10】不恢复余数出发用于补码运算时，符号和数值位直接进行运算。

2.4 定点运算器的组成

2.4.1 逻辑运算

【ppt p1】计算机中除了进行加、减、乘、除等基本算术运算外，还可对两个或一个逻辑数进行逻辑运算。所谓逻辑数，是指不带符号的二进制数。利用逻辑运算可以进行两个数的比较，或者从某个数中选取某几位等操作。计算机中的逻辑运算，主要是指**逻辑非**、**逻辑加**、**逻辑乘**、**逻辑异**四种基本运算。

1、逻辑非运算

【ppt p2 动画 1】逻辑非也称**求反**。0 的非等于 1，1 的非等于 0，对某数进行逻辑非运算，就是按位求它的反，常用变量上方加一横来表示。

【动画 2】 设一个数 x 表示成：　$x = x_0\ x_1\ x_2 \ldots x_n$

对 x 求逻辑非，则有：　$z = \overline{x} = z_0\ z_1\ z_2 \ldots z_n$

$$z_i = \overline{x_i}, \quad (i = 0, 1, 2, \ldots n)$$

【动画 3】[例] $x_1 = 01001011$，　$x_2 = 11110000$，求 x_1 ，　x_2

【动画 4】 $\overline{x_1}=10110100, \overline{x_2}=00001111$

$x_1 = 10110100$，　$x_2 = 00001111$

2、逻辑加运算

【ppt p3 动画 1】 **逻辑加运算规则**，0∨0 = 0，0∨1 = 1，1∨0=1，1∨1=1；归纳见 1 就的 1，全 0 才得 0。 对两个数进行逻辑加，就是按位求它们的“或”，所以逻辑加又称**逻辑或**，常用记号“V”或“+”来表示。

【动画 2】设有两数 ，它们表示为 $x = x_0 x_1 \ldots x_n$， $y = y_0 y_1 \ldots y_n$

若 $x \vee y = z = z_0 z_1 z_2 \ldots z_n$ 则 $z_i = x_i \vee y_i$， (i = 0，1，2，…，n)

【动画 3】[例] x = 10100001， y = 10011011，求 $x \vee y$。

【动画 4】$x \vee y$ = 10100001 ∨ 10011011 =10111011

3、逻辑乘运算

【ppt p4 动画 1】 **逻辑乘运算规则** *0·0 = 0，0·1 = 0，1·0=0，1·1=1；归纳见 0 就的 0，全 1 才得 1*。对两数进行逻辑乘，就是按位求它们的“与”，所以逻辑乘又称“逻辑与”，常用记号“∧”或“·”来表示。

【动画 2】设有两数 x 和 y，它们表示为 $x = x_0 x_1 \ldots x_n$， $y = y_0 y_1 \ldots y_n$ 。若 $x \wedge y = z = z_0 z_1 z_2 \ldots z_n$，则 $z_i = x_i \wedge y_i$，(i = 0，1，2，…，n)

【动画 3】[例 23] x = 10111001， y = 11110011，求 $x \wedge y$。

【动画 4】 [解:] $x \wedge y$ = 10111001∧ y 11110011 = 10110001

4、逻辑异运算

【ppt p5 动画 1】逻辑乘运算规则 0⊕0 = 0，0⊕1 = 1，1⊕0=1，1⊕1=0；相同就得 0，不同得 1。对两数进行异就是按位求它们的模 2 和，所以逻辑异又称“按位加”，常用记号“⊕”表示。

【动画 2】设有两数 x 和 y： $x = x_0 x_1 \ldots x_n$， $y = y_0 y_1 \ldots y_n$

若 x 和 y 的逻辑异为 z： $x \oplus y = z = z_0 z_1 z_2 \ldots z_n$

则 $z_i = x_i \oplus y_i$， (i = 0，1，2，…，n)

【动画 3】[例 24] x = 10101011， y = 11001100，求 $x \oplus y$。

【动画 4】 $x \oplus y$ = 10101011 ⊕ 11001100 = 01100111

【ppt p6】另外，逻辑加还可以通过逻辑乘和逻辑非来实现：……。
同样，逻辑乘也可以用逻辑加和逻辑非来实现：……。

2.4.2 多功能算术/逻辑运算单元(ALU)

【ppt p1 动画 1】由一位全加器(FA)构成的 n 位行波进位加法器，它可以实现补码数的加法运算和减法运算。【动画 2】但是这种加法/减法器存在两个问题：***第一***，由于

串行进位，运算位数越多，运算时间很长，对于提高计算速度是不利的。***第二，***行波进位加法器只能完成加法和减法两种操作，不能完成逻辑操作。我们想让他既可以进行算术运算的加减运算，又可以实现逻辑运算。想要解决这两个问题，即是从功能和速度上来解决行波进位加法器存在问题，如何解决呢。现在我们来看一种解决方案。多功能算术/逻辑运算单元(ALU)，它不仅具有多种算术运算和逻辑运算的功能，而且具有先行进位逻辑，从而能实现高速运算。

1、能扩充的思想—函数发生器

【ppt p2 动画 1，2】一位全加器(FA)的逻辑表达式为：……。***【动画 3，4】***我们对全加器的输入端进行改造，加入函数发生器，函数发生器输出 Xi，Yi 是 Ai，Bi 的函数，具体什么输出什么函数用 S0，S1，S2，S3 的值的状态控制，如图所示。也就是将 Ai 和 Bi 先组合成由控制参数 S0，S1，S2，S3 控制的组合函数 Xi 和 Yi，然后再将 Xi，Yi 和下一位进位数通过全加器进行相加。

【ppt p3 动画 1】**X_i，Y_i与控制参数和输入量的关系如表所示。表中紫色为该数的非运算。**这样，***【动画 2】***原来一位全加器(FA)的逻辑表达式为……。

不同的控制参数可以得到不同的组合函数，因而能够实现多种算术运算和逻辑运算。***【ppt p4 动画 1】***根据控制参数 S_0，S_1，S_2，S_3 分别控制输入 A_i 和 B_i，产生 Y 和 X 的函数。其中 Y_i 是受 S_0，S_1 控制的 A_i 和 B_i 的组合函数，而 X_i 是受 S_2，S_3 控制的 A_i 和 B_i 组合函数，其函数关系如表所示。

根据上面所列的函数关系，即可列出 X_i 和 Y_i 的逻辑表达式

【ppt p4 动画 2】进一步化简并代入前面的求和与进位表达式，

【ppt p4 动画 3】***进一步化简***可得函数发生器的逻辑表达式如下：

【ppt p5 动画 1】下面我们来解决速度问题---进位方式改进

【动画 2】将行波进位改为，采用先行进位方式，以 4 位为例，若输入数据 4 位，4 位之间采用先行进位公式，根据全加器的逻辑表达式，每一位的进位公式可递推如下：第 0 位向第 1 位的进位公式为：$C_{n+1} = Y_0 + X_0C_n$ 其中 C_n 是向第 0 位（末位）的进位。第 1 位向第 2 位的进位展开公式为 $C_{n+2} = Y_1 + X_1C_{n+1} = Y_1 + Y_0X_1 + X_0X_1C_n$

第 2 位向第 3 位的进位展开公式为 $C_{n+3} = Y_2 + X_2C_{n+2} = Y_2 + Y_1X_1 + Y_0X_1X_2 + X_0X_1X_2C_n$ 第 3 位的进位输出（即整个 4 位运算进位输出）公式为

$$C_{n+4} = Y_3 + X_3C_{n+3} = Y_3 + Y_2X_3 + Y_1X_2X_3 + Y_0X_1X_2X_3 + X_0X_1X_2X_3C_n$$

【动画 3】设与 C_n 无关部分进位产生 $G = Y_3 + Y_2X_3 + Y_1X_2X_3 + Y_0X_1X_2X_3$

C_n 的系数进位传送 $P = X_0X_1X_2X_3$

则可以得到 4 位运算的进位输出： $C_{n+4} = G + PC_n$

这样，对一片 ALU 来说，可有三个进位输出。C_{n+4} 是本组的最后进位输出，G 称为**进位发生输出**，P 称为**进位传送输出**。

2、4 位的 ALU.

在电路中多加这两个进位输出的目的，是为了便于实现多片（组）ALU 之间的先行进位，为此还需一个配合电路，称之为**先行进位发生器**(CLA)，下面还要介绍。

C_{n+4} 是本片(组)的最后进位输出。逻辑表达式表明，这是一个先行进位逻辑。换句话说，第 0 位的进位输入 C_n 可以直接传送到最高位上去，因而可以实现高速运算。

3、逻辑运算的实现

【ppt p6】逻辑运算与算术运算的区别在进位，逻辑运算按位运算，不产生进位，逻辑运算时，所有进位均为 0，即屏蔽进位。

4、算术逻辑运算单元的实现

【ppt p7】这是用正逻辑表示的 4 位算术/逻辑运算单元(ALU)的逻辑电路图。这个期间的商业标号为 74181ALU。***【动画 1，2】***4 个函数发生器，***【动画 3】***4 部分先行进位，***【动画 4】***算术逻辑控制端 M。

【ppt p8 动画 1】图中控制输入端除了 S_0－S_3 四个控制端外，***【动画 2】***还有一个控制端 *M*，它使用来控制 ALU 是进行算术运算还是进行逻辑运算的。***【动画 3】*** 当 *M* = 0 时，*M*对进位信号没有任何影响。此时 F 不仅与本位的被操作数 Y 和操作数 X 有关，而且与本位的进位输出，即 C 有关，因此 *M* = 0 时，进行**算术操作**。***【动画 4】***当 *M* = 1 时，封锁了各位的进位输出，即 C =0，因此各位的运算结果 F 仅与 Y 和 X 有关，故 *M* = 1 时，进行**逻辑操作**。

【ppt p9】把这个逻辑电路封装起来，就是一片 4 位 ALU 芯片 74181ALU。有正逻辑和负逻辑两种工作方式。***【ppt p10】***这是 **74181ALU 的运算功能表，*M*控制 ALU 是进行算术运算还是进行逻辑运算，S_0－S_3 值的组合可以实现 16 种运算。**

注意，表中算术运算操作是用补码表示法来表示的。其中“加”是指算术加，运算时要考虑进位，而符号“＋”是指“逻辑加”。

其次，减法是用补码方法进行的，其中数的反码是内部产生的，而结果输出“A 减 B 减 1”，因此做减法时需在最末位产生一个强迫进位(加 1)，以便产生“A 减 B”的结果。另外，“A = B”输出端可指示两个数相等，因此它与其他 ALU 的“A = B"输出端按“与”

逻辑连接后，可以检测两个数的相等条件。

5、两级先行进位的 ALU

【ppt p11】前面说过，74181ALU 设置了 P 和 G 两个本组先行进位输出端。如果将四片 74181 的 P，G 输出端送入到 74182 先行进位部件（CLA），又可实现第二级的先行进位，即组与组之间的先行进位。***【动画 1】*** 假设 4 片（组）74181 的先行进位输出依次为 P0，G0，G1P1，P2，G2，P3，G3，根据前面的进位逻辑表达式，先行进位部件 74182CLA 所提供的进位逻辑关系推导如下：***【动画 2】***

【ppt p12】根据以上表达式，用 TTL 器件实现的成组先行进位部件 74182 的逻辑电路图如图所示其中 G^*称为**成组进位发生输出**，P^*称为**成组进位传送输出**。

若干个 74181ALU 位片，与配套的 74182 先行进位部件 CLA 在一起，可构成一个组内先行进位，组间先行进位的全字长 ALU。

【ppt p13】这里给出了用八个 74181ALU 和两个 74182CLA 组成的 32 位 ALU 逻辑方框图。很显然，对一个 16 位来说，CLA 部件构成了第二级的先行进位逻辑，即实现四个小组（位片）之间的先行进位，从而使全字长 ALU 的运算时间大大缩短。

2.4.3 定点运算器的基本结构

【ppt p1】运算器包括 ALU\阵列乘除器\寄存器\多路开关\三态缓冲器\数据总线等逻辑部件。

运算器的设计，主要是围绕 ALU 和寄存器同数据总线之间如何传送操作数和运算结果进行的。

在决定方案时，需要考虑数据传送的方便性和操作速度，在微型机和单片机中还要考虑在硅片上制作总线的工艺。 计算机的运算器大体有如下三种结构形式。

1、单总线结构的运算器

【ppt p2】这是一个单总线结构的运算器。由于所有部件都接到同一总线上，所以数据可以在寄存器与寄存器之间，寄存器和 ALU 之间传送。如果具有阵列乘法器或除法器，那么它们所处的位置应与 ALU 相当。***【动画 1】***这种结构的运算器，在同一时间内，只能有一个操作数放在单总线上。***【动画 2】***运算时需要需要分两次传送两个操作数到 ALU，而且还需要 A，B 两个缓冲寄存器，***【动画 3】***运算完成送操作结果，需要三次串行的选通操作。这种结构的主要缺点是操作速度较慢。但是它只控制一条总线，故控制电路比较简单。

2、双总线结构的运算器

双总线结构的运算器如所示。在这种结构中，两条总线各自把其数据送至 ALU 的输入端，两个操作数同时进行，只需一次操作控制，而且马上就可以得到运算结果，但结果不能直接加到总线上去，此时，两条总线都被输入数占据，因而必须在 ALU 输出端设置缓冲寄存器。为此，操作的控制要分两步完成:

【ppt p3 动画 1，2】1.在 ALU 的两个输入端输入操作数，形成结果并送入缓冲寄存器; ***【动画 3】***2.把结果送入目的寄存器。

这里特殊寄存器分为两组，它们分别与一条总线交换数据。这样，通用寄存器中的数就可进入到任一组特殊寄存器中去，从而使数据传送更为灵活。

3、三总线结构的运算器

【ppt p4】现在我们来看三总线结构的运算器。***【动画 1，2】***在三总线结构中，ALU 的两个输入端分别连接两条总线，而 ALU 的输出则与第三条总线相连。这样，算术逻辑操作就可以在一步的控制之内完成。由于 ALU 本身有时间延迟，所以打入输出结果的选通脉冲必须考虑到包括这个延迟。

另外，设置了一个总线旁路器。如果一个操作数不需要修改，而直接从总线 2 传送到总线 3，那么可以通过控制总线旁路器把数据传出;如果一个操作数传送时需要修改，那么就借助于 ALU。很显然，三总线结构的运算器的特点是操作时间快。

2.5 浮点运算方法和浮点运算器

2.5.1 浮点加法、减法运算

【ppt p1 动画 1】设有两个浮点数 x 和 y，它们分别为 $x=2^{Ex}\cdot M_x$ $y=2^{Ey}\cdot M_y$。其中 E_x 和 E_y 分别为数 x 和 y 的阶码，M_x 和 M_y 为数 x 和 y 的尾数。**【动画 2】**两浮点数进行加法和减法的运算规则是……

【ppt p2 动画 1】这是浮点加减法浮点加减运算的操作流程图，浮点加减法运算从操作数输入到输出结果，**【动画 2】**我们把浮点加减运算的操作过程大体分为四步: 1. 0 操作数的检查; 2. 比较阶码大小并完成对阶; 3. 尾数进行加或减运算; 4. 结果规格化并进行舍入处理。

【ppt p3】下面我们来分步看一下浮点加减运算过程:

(1) 0 操作数检查

浮点加减运算过程比定点运算过程复杂。如果判知两个操作数 x 或 y 中有一个数为 0，即可得知运算结果而没有必要再进行后续的一系列操作以节省运算时间。**【动画 1】**0 操作数检查步骤则用来完成这一功能。

(2) 比较阶码大小并完成对阶

两浮点数进行加减，首先要看两数的阶码是否相同，即小数点位置是否对齐。若二数阶码相同，表示小数点是对齐的，就可以进行尾数的加减运算。反之，若二数阶码不同，表示小数点位置没有对齐，此时必须使二数阶码相同，这个过程叫作**对阶**。**【动画 2】**要对阶，首先应求出两数阶码 E_x 和 E_y 之差，即 $\triangle E = E_x - E_y$

若 $\triangle E = 0$，表示两数阶码相等，即 $E_x = E_y$；若 $\triangle E>0$，表示 $E_x<E_y$；若 $\triangle E<0$，表示 $E_x>E_y$。

当 $E_x \neq E_y$ 时，要通过尾数的移动以改变 E_x 或 E_y，使之相等。原则上，既可以通过 M_x 移位以改变 E_x 来达到 $E_x = E_y$，也可以通过 M_y 移位以改变 E_y 来实现 $E_x = E_y$。但是，由于浮点表示的数多是规格化的，尾数左移会引起最高有效位的丢失，造成很大误差。尾数右移虽引起最低有效位的丢失，但造成误差较小。因此，对阶操作规定使尾数右移，尾数右移后阶码作相应增加，其数值保持不变。

一个增加后的阶码与另一个阶码相等，增加的阶码的一定是小阶。因此在对阶时，总是使**小阶向大阶看齐**，即小阶的尾数向右移位(相当于小数点左移）每右移一位，其阶码加 1，直到两数的阶码相等为止，右移的位数等于阶差 $\triangle E$。

(3) 尾数求和运算

【动画 3】对阶结束后，即可进行尾数的求和运算。不论加法运算还是减法运算，都按加法进行操作，其方法与定点加减法运算完全一样。

(4) 结果规格化

在浮点加减运算时，尾数求和的结果也可以得到 01.ϕ…ϕ 或 10.ϕ…ϕ，即两符号位不等，这在定点加减法运算中称为溢出，是不允许的。但在浮点运算中，它表明尾数求和结果的绝对值大于 1，向左破坏了规格化。此时将运算结果右移以实现规格化表示，称为**向右规格化**。规则是：尾数右移 1 位，阶码加 1。当尾数不是 1.M 时需向左规格化。**【动画 4】**

(5) 舍入处理

在对阶或向右规格化时，尾数要向右移位，这样，被右移的尾数的低位部分会被丢掉，从而造成一定误差，因此要进行**舍入处理**。简单的舍入方法有两种：一种是"0

舍 1 入"法，即如果右移时被丢掉数位的最高位为 0 则舍去，为 1 则将尾数的末位加"1"。另一种是"恒置一"法，即只要数位被移掉，就在尾数的末尾恒置"1"。

【ppt p4】小阶向大阶看齐：一个增加后的阶码与另一个阶码相等，增加的阶码的一定是小阶。因此在对阶时，总是使小阶向大阶看齐，即小阶的尾数向右移位(相当于小数点左移）每右移一位，其阶码加 1，直到两数的阶码相等为止，右移的位数等于阶差△E。

【ppt p5】向右规格化。但在浮点运算中，它表明尾数求和结果的绝对值大于 1，向左破坏了规格化。规则是：尾数右移 1 位，阶码加 1。当尾数不是 1.M 时需向左规格化，尾数左移 1 位，阶码减 1 ，右边补 1 位。

【ppt p6】在 IEEE754 标准中，舍入处理提供了四种可选方法：

就近舍入 其实质就是通常所说的"四舍五入"。例如，尾数超出规定的 23 位的多余位数字 是 10010，多余位的值超过规定的最低有效位值的一半，故最低有效位应增 1。若多余的 5 位 是 01111，则简单的截尾即可。对多余的 5 位 10000 这种特殊情况：若最低有效位现为 0，则截尾；若最低有效位现为 1，则向上进一位使其变为 0。

朝 0 舍入 即朝数轴原点方向舍入，就是简单的截尾。无论尾数是正数还是负数，截尾都使取值的绝对值比原值的绝对值小。这种方法容易导致误差积累。

朝 + ∞舍入 对正数来说，只要多余位不全为 0 则向最低有效位进 1;对负数来说则是简单的 截尾。

朝 − ∞舍入 处理方法正好与 朝 + ∞舍入情况相反。对正数来说，只要多余位不全为 0 则 简单截尾;对负数来说，向最低有效位进 1。

浮点数的溢出

【ppt p7】下图表示了浮点机器数在数轴上的分布情况。

当机器浮点数值大于最大正数 A 值，或小于最小负数 B 值时，称为上溢，这两种情况意味着阶码运算值超出了它所表示的范围，机器必须做中断处理。

当机器浮点数值小于最小正数 a 值，或大于最大负数 b 值时，称为下溢。下溢不是一个严重问题，通常看作为机器零。

浮点数的溢出是以其阶码溢出表现出来的。在加\减运算过程中要检查是否产生了溢出：若阶码正常，加(减)运算正常结束；若阶码溢出，则要进行相应处理。另外对尾数的溢出也需要处理。

阶码上溢 超过了阶码可能表示的最大值的正指数值，一般将其认为是 + ∞和 − ∞。

阶码下溢 超过了阶码可能表示的最小值的负指数值，一般将其认为是 0。**尾数上溢** 两个同符号尾数相加产生了最高位向上的进位，将尾数右移，阶码增 1 来重新对齐。**尾数下溢** 在将尾数右移时，尾数的最低有效位从尾数域右端流出，要进行舍入处理。

2.5.2 浮点乘法、除法运算

1、浮点乘法、除法运算规则

【ppt p1 动画 1】设有两个浮点数 x 和 y：$x=2^{E_x}\cdot M_x$　　$y=2^{E_y}\cdot M_y$

【动画 2】浮点乘法运算的规则是……即乘积的尾数是相乘两数的尾数之积，乘积的阶码是相乘两数的阶码之和。当然，这里也有规格化与舍入等步骤。**【动画 3】**浮点除法运算的规则是…… 商的尾数是相除两数的尾数之商，商的阶码是相除两数的阶码之差。也有规格化和舍入等步骤。

2、浮点乘、除法运算步骤

【ppt p2】浮点数的乘除运算大体分为四步：第一步，0 操作数检查；第二步，阶码加/减操作；第三步，尾数乘/除操作；第四步，结果规格化及舍入处理。

(1) 浮点数的阶码运算

【ppt p3 画 1】对阶码的运算有 +1、−1、两阶码求和、两阶码求差四种，运算时还必须检查结果是否溢出。**【动画 2】**在计算机中，阶码通常用补码或移码形式表示。补码运算规则和判定溢出的方法，前面已经讲过。这里只对移码的运算规则和判定溢出的方法进行讲解。移码的定义为 $[x]_{移}=2^n+x \quad 2^n>x\geq -2^n$。按此定义，则有 $[x]_{移}+[y]_{移}=2^n+x+2^n+y=2^n+(2^n+(x+y))=2^n+[x+y]_{移}$

即直接用移码实现求阶码之和时，结果的最高位多加了个 1，要得到正确的移码形式结果，必须对结果的符号再执行一次求反。

解码也可采用移码和补码混合运算 **【ppt p4 动画 1】**……**【动画 2】**即当混合使用移码和补码时，考虑到移码和补码的关系：对同一个数值，其数值位完全相同，而符号位正好完全相反。而$[y]_{补}$的定义为 $[y]_{补}=2^{n+1}+y$，则求阶码和用如下方式完成：$[x]_{移}+[y]_{补}=2^n+x+2^{n+1}+y=2^{n+1}+(2^n+(x+y))$同理**【动画 3】**……混合使用移码和补码时执行阶码加减时，对加数或减数 y 来说，应送移码符号位正常值的反码。

【动画 4】 如果阶码运算的结果溢出，上述条件则不成立。此时，**【动画 5】**使用双符号位的阶码加法器，并规定移码的第二个符号位，即最高符号位恒用 0 参加加减运算，则溢出条件是结果的最高符号位为 1。**【动画 6】**此时，当低位符号位为 0 时，

表明结果上溢，为 1 时，表明结果下溢。当最高符号位为 0 时，表明没有溢出；低位符号位为 1，表明结果为正；为 0 时，表明结果为负。

(2) 尾数处理

【ppt p5】 浮点加减法对结果的规格化及舍入处理也适用于浮点乘除法。

2.5.3 浮点运算流水线

1、流水线原理

【ppt p1 动画 1】我们都知道大规模工业生产过程中，流水线技术极大的提高了生产效率，得到广泛应用。计算机技术中的流水处理过程中引入流水技术。为了实现流水，首先必须把输入的任务分割为一系列的子任务，使各子任务能在流水线的各个阶段并发地执行。将任务连续不断地输入流水线，从而实现了子任务的并行。因此流水处理大幅度地改善了计算机的系统性能，是在计算机上实现**时间并行性**的一种非常经济的方法。**【动画 2】**现在假定有作业 T 被分成 k 个子任务，可以用这个公式表达：

$T = \{T_1, T_2, \cdots, T_k\}$

各个子任务之间有一定的先后关系：若 i<j，则必须在 T_i 完成以后，T_j 才能开始工作。具有这种线性优先关系的流水线称为**线性流水线**。

【ppt p2】这是线性流水线处理的硬件基本结构图。图中，处理一个子任务的过程为**过程段**(S_i)。**【动画 2】**线性流水线由一系列串联的过程段组成，各个过程之间设有高速的缓冲寄存器(L)，进行隔离和缓冲。**【动画 3】**在一个统一的时钟(C)控制下，数据从一个过程段流向相邻的过程段。

2、线性流水线的时钟周期

【ppt p3】设过程段 S_i 所需的时间为 τ_i，缓冲寄存器的延时为 τ_l，线性流水线的时钟周期定义为 $\tau = \max\{\tau_i\} + \tau_l = \tau_m + \tau_l$。故流水线处理的频率为 $f = 1/\tau$。

在流水线中，原则上要求各个阶段的处理时间都相同。若某一阶段的处理时间较长，势必造成其他阶段的空转等待。因此对子任务的划分，是决定流水线性能的一个关键因素，它取决于操作部分的效率、所期望的处理速度，以及成本价格等等。

在流水线处理中，当任务饱满时，任务源源不断的输入流水线，不论有多少级过程段，每隔一个时钟周期都能输出一个任务。

3、线性流水线的加速比

【ppt p4 动画 1】从理论上说，一个具有 k 级过程段的流水线处理 n 个任务需要

的时钟周期数为$T_k=k+(n-1)$。其中 k 个时钟周期用于处理第一个任务。k 个周期后，流水线被装满，剩余的 n－1 个任务只需 n－1 个周期就完成了。***【动画 2】***如果用非流水线的硬件来处理这 n 个任务，时间上只能串行进行，则所需时钟周期数为 $T_L=n\cdot k$ ***【动画 3】***我们将 T_L 和 T_k 的比值定义为 k 级线性流水线的加速比: ……。当 n>>k 时，C_k－>k 。***【动画 4】***这就是说，理论上 k 级线性流水线处理几乎可以提高 k 倍速度。但实际上由于存储器冲突、数据相关，这个理想的加速比不一定能达到。

4、流水线浮点加法器实例

【ppt p5】浮点数加减法由 0 操作数检查、对阶操作、尾数操作、结果规格化及舍入处理共 4 步完成，因此流水线浮点加法器可由 4 个过程段组成。***【动画 1】***这是一个 3 段流水线浮点加法器框图，从图可以看出，左侧是一个除 0 操作数检查之外的 3 段流水线浮点加法器框图。***【动画 2】***有两个规格化的浮点数 $X=1.1000\times2^2$ $Y=1.1100\times2^4$ 进行加法运算，流水线浮点加法器框图右侧，标出了上述例子在每一个过程段和锁存器 L 中保存的流水运算结果值。

第3章　多层次的存储器（6 学时）

3.1 存储器概述

3.1.1 存储器的分类

（PPT311 第 1 页）计算机能够运行程序、处理数据，从而解决相关的问题。请问同学们，程序和数据从哪儿来，处理的结果又要放到哪儿去？（动画 1）有些同学会坚定的回答，凭借的就是存储器。因为存储器是附图 311.1 中的五大主要部件之一，附图 311.1 表示的是冯·诺依曼型计算机的硬件，存储器是计算机系统中的记忆部件，用来存放系统程序、应用程序和数据。如果拔掉计算机的内存条，系统还能正常工作吗？如果一台计算机没有硬盘，情况又会怎样？很显然，没有存储器的支持，计算机是无法正常工作的。讲到这里，同学们还会有一系列的疑问，譬如：存储器为什么能够存储程序和数据？它的逻辑结构是什么？计算机中究竟运用了哪些存储器？等等等等。为了弄清这些问题，我们下面共同学习第三章——多层次的存储器。

（PPT311 第 2 页）多层次的存储器这一章的重点主要包括存储器技术指标、随机读写存储器、并行存储器、高速缓冲存储器 cache，难点主要包括存储器扩展、cache 的映射规则、cache 的替换算法。需要说明的是，硬磁盘存储器将安排在这一章进行讲解，整章内容量大，请同学们认真做好预习、课堂笔记和课后练习。

（PPT311 第 3 页）计算机中究竟部署了哪些类型的存储器呢？这个问题可以概括为附表 311.1 来回答。由附表 311.1 可知，存储器按照存储介质、存储方式、存储器的读写功能、信息的易失性、在计算机系统中的作用等五个标准进行分类。

内存条属于半导体存储器，其存储介质是半导体器件，能满足速度快的要求；硬盘和磁带属于磁表面存储器，其存储介质是磁性材料，能满足容量大的要求。

半导体存储器属于随机存储器，它采用随机存取方式，任何存储单元的内容都能被随机存取，且存取时间和存储单元的物理位置无关；磁带属于顺序存储器，它采用顺序存取方式，只能按某种顺序来存取，存取时间和存储单元的物理位置有关。

内存条是一种既能读出又能写入的随机读写存储器（RAM），其内容是可以更改的；计算机也需要一种只能读出而不能写入的只读存储器（ROM），其内容是固定不变的，ROM 用来存放一些硬件的驱动程序，可避免用户有意或无意的修改。

内存条是一种断电后信息就会消失的易失性存储器；硬盘是一种断电后信息不会

消失的非易失性存储器。

（选讲）（讲到这里，我和同学们分享一个真实的故事。十五年前，我给一个勘察设计院讲 CAD 培训课程，当时有一位工作不久的小张学的很认真、也很快，有天下午，他集中精力绘制一张施工图，花了一个半小时完成了 80%，有几位同事在旁边羡慕地看着他绘图的过程，不知谁把插线板用脚碰了一下，有三台计算机突然断电关机，其中包括小张用的计算机。当他们七手八脚弄好电源、小张再次打开那个图文档时，文档中只有他刚开始画的两条线，其余的东西都不见了，他都快要急哭了。）

我告诉他，以后要养成及时“保存”文档的良好习惯，这个简单的操作会将驻留在内存的信息写入硬盘，能够永久记忆下来，除非硬盘出现故障，否则不会丢失。

同学们应该知道两个常用的名称，一个是内存，另一个是外存（或辅存）。内存的全称是内部存储器，要求速度快，半导体存储器属于内存；外存的全称是外部存储器，辅存的全称是辅助存储器，要求容量大，硬盘属于外存。主存储器和高速缓冲存储器（Cache）合在一起被称为内存。在学习第五章中央处理器时，同学们将会接触到一种名为控制存储器的硬件，它是用来存放微程序的，容量小、速度快，是控制器的主要组成部件之一，它的容量主要与微指令的格式和长度有关。

（本节讲授完毕）

3.1.2 存储器的分级

（PPT312 第 1 页）如果有一种存储器，集容量大、速度快、成本低于一身的话，计算机系统的结构便会简单很多。但事实是，很难设计制造出这样的存储器，而计算机的运行速度越来越快，要求能够高速存取指令和数据，计算机的应用越来越广泛，可以解决更加复杂的问题，随之而来的是大量的程序和数据需要更大容量的存储器来存储，考虑到整机性价比的问题，存储器的价格还需控制在合理的范围之内。为此，计算机领域的专家、学者和工程师们经过反复研究实验，发现计算机采用多级存储体系结构这一方案，能够有效解决速度、容量、价格这三方面的矛盾。

（PPT312 第 2 页）多级存储体系结构可以用图 3.1 来表示，也可以用附表 312.1 作进一步的说明。（动画 1）其中 cache 是一种小容量的半导体存储器，主要强调快速存取，其存取速度要和 CPU 的运算速度相匹配，利用它能够高速存取指令和数据；磁盘、光盘、磁带属于外存，主要强调大的存储容量，以满足计算机的大容量存储要求，位成本低；主存储器由 MOS 半导体存储器组成，介于 cache 与外存之间，它能和 cache

交换数据和指令，要求选取适当的存储容量和存取周期，使它能容纳系统的核心软件和较多的用户程序。

需要强调两个容易混淆的知识点，那就是：cache 和主存储器属于内存储器，它们能被 CPU 直接访问，磁盘、磁带、光盘属于外存储器，它们不能被 CPU 直接访问。

（本节讲授完毕）

3.1.3 主存储器的技术指标

（PPT313 第 1 页）一个主存储器能存放多少二进制信息、其速度如何，需要用存储容量、存取时间、存储周期和存储器带宽等主要的技术指标来衡量。我们将这四个指标的含义、表现、单位归纳之后，可以用附表 313.1 来表达。

存储容量指一个存储器中可以容纳的存储单元总数。它是衡量存储空间大小的指标，单位通常用字数或字节数。目前，台式机的内存容量是 4G～32G（DDR4）、笔记本电脑的内存容量是 2G～32G。DDR 的全称是 Double Data Rate SDRAM，中文名称是双倍速率同步动态随机存储器。

存取时间又称存储器访问时间，是指一次读操作命令发出到该操作完成，将数据读出到数据总线上所经历的时间。单位通常用纳秒。

存储周期指连续启动两次读操作所需间隔的最小时间。单位通常用纳秒。半导体存储器的存储周期一般为 60 纳秒至 100 纳秒。

存储器带宽指的是单位时间里存储器所存取的信息量，通常以位/秒或字节/秒做度量单位。它决定了以存储器为中心的机器获取信息的传输速度。

其通用计算公式是：存储器带宽=存储器工作频率 × 存储器数据总线位数/8

存取时间、存储周期和存储器带宽是反映主存速度的指标。

（本节讲授完毕）

3.2 SRAM 存储器

3.2.1 基本的静态存储元阵列

（PPT321 第 1 页）同学们知道，当下 PC 机中的内存条是 DDR4，属于半导体存储器，速度快、存储体积小、节能可靠、价格便宜，它是动态随机存储器 DRAM 的换代升级产品，中文全称是双倍速率同步动态随机存储器第四代。事实上，还有一种比 DRAM 更快的静态随机存储器 SRAM，只不过它的容量很难做的像 DRAM 那么大。

图 3.2 表示了基本的静态存储元阵列。该图中，（动画 1）行列交叉处的小方框代

表着一个存储元，（动画 2）右上角有其局部放大图，实际上是一个锁存器，也称为触发器，这是记忆信息的关键器件，它用来存储一位二进制信息 0 或 1。只要直流供电电源一直加在这个记忆电路上，它就无限期地保持记忆的 1 状态或 0 状态。如果电源断电，那么存储的数据(1 或 0)就会丢失。（动画 3）任何一个 SRAM，都有地址线、数据线、控制线这三组信号线与外部打交道。

（动画 4）图 3.2 中有 6 条地址线，用于传送 A0~A5 共 6 位地址信号，（动画 5）经地址译码器输出 64 条选择线，也称为行线，它的作用是打开每个存储位元的输入与非门，可寻址 2^6=64 个存储单元。这里需要提醒一下，教材 66 页的图 3.2 中，选择线序标是有问题的，正确的表示应该是 0～63，不是 64，请同学们修改一下。

（动画 6）图 3.2 中有 4 条数据线 I/O_0~I/O_3，它指定了存储器的字长是 4 位，因此存储位元的总数是 64 × 4=256 个。

（动画 7）图 3.2 中有 1 条控制线 R/W 非，通过施加高电平或低电平，控制存储器的读或写操作。（动画 8）需要注意的是，存储器通常采用读/写互锁逻辑，即读/写不能同时进行。

（本节讲授完毕）

3.2.2 基本的 SRAM 逻辑结构

（PPT322 第 1 页）前面学习的图 3.2 表示了基本的静态存储元阵列，（动画 1）采用的是单译码方式，配有一个地址译码器，（动画 2）传送 6 位地址信号的情况下，选择线（也称为译码线）有 64 条，寻址存储容量为 2^6=64 个单元。（动画 3）请问同学们，若采用单译码方式，要寻址 4K 存储容量，需要多少条选择线？由 4K=4096 个单元=2^{12} 可知，需要 4096 根选择线。哪 8K、1M、1G 的存储芯片呢？（动画 4）8K 的需要 8192 根选择线、13 位地址信号，1M 的需要 100 多万（1048576）根选择线、20 位地址信号，1G 的需要 10 亿多（1073741824）根选择线、30 位地址信号。可见，随着芯片容量的增大，选择线的数量呈几何级数增长，硬件会变得很复杂，难以实现。所以，单译码适用于小容量存储器。我们试想一下，那大容量的存储芯片该怎么办啊？（动画 5）采用双译码方式可以解决这个问题。

（PPT322 第 2 页）图 3. 3(a)表示存储容量为 32K × 8 位的 SRAM 逻辑结构图。（动画 1）它采用了双译码方式，也就是将地址分成 x 向、y 向两部分，第一级进行 x 向(行译码)和 y 向(列译码)的独立译码，然后在存储阵列中完成第二级的交叉译码。因为该

存储器容量是 32K，需要 15 位地址信号（即 15 条地址线），x 方向分配 8 条(A0～A7)，（动画 2）经行译码输出 2^8=256 行，y 方向分配 7 条(A8～A14)，经列译码输出 2^7=128 列，行、列译码线加起来总共是 384 条，比采用单译码要少很多很多，单译码需要译码线 32768 条。（动画 3）双向数据线有 8 条，即 I/O_0～I/O_7。（动画 4）这样以来，形成了 256 行×128 列×8 位的三维存储阵列结构。

（PPT322 第 3 页）（动画 1）图 3. 3(a)中，通过控制信号 CS 非、WE 非、OE 非和两个与门 G1、G2 以及输入/输出缓冲器（也称为输入/输出三态门）实现读/写互锁逻辑。控制信号中 CS 非是片选信号，WE 非是写允许信号，OE 非为读出使能信号，这三个信号都是低电平有效。（动画 2）当 CS 非有效、OE 非有效、WE 非=1 时，CS 非和 OE 非圈后都为 1，WE 非为 1，这三个信号将门 G2 开启，同时门 G2 的输出信号 1 将 8 个输出缓冲器打开，WE 非圈后为 0 将门 G1 关闭，同时门 G1 的输出信号 0 将 8 个输入缓冲器关闭，（动画 3）存储器进行读操作，从存储阵列中读出的数据 10011101 被送到 8 条 I/O 数据线上。

（PPT322 第 4 页）（动画 1）当 CS 非有效、WE 非有效、OE 非=1 时，CS 非和 WE 非圈后都为 1，这二个信号将门 G1 开启，同时门 G1 的输出信号 1 将 8 个输入缓冲器打开，WE 非为 0 将门 G2 关闭，同时门 G2 的输出信号 0 将 8 个输出缓冲器关闭，（动画 2）存储器进行写操作，8 条 I/O 数据线上的数据写入到存储阵列中去。（动画 3）需要注意的是：门 G1 和 G2 是互锁的，输出、输入缓冲器不仅有方向性，而且是互锁的，当一个开启时另一个必定是关闭的，这样保证了读时不写、写时不读。

掌握了“基本的 SRAM 逻辑结构”，还有一个问题需要我们弄清楚，那就是 SRAM 工作的时间关系，这就要求同学们盯住后面讲解的“SRAM 读写周期波形图”。

（本节讲授完毕）

3.2.3SRAM 读写周期波形图

（PPT323 第 1 页）前面我们通过学习 SRAM 的读/写互锁逻辑，基本上知道了读操作和写操作的数据通路是如何形成的，但很多细节并不清楚，譬如地址信号、控制信号、数据何时有效，它们之间的先后顺序是什么样的？即 SRAM 工作的时间关系如何？要弄清这个问题，需要我们一起研究它的读/写周期波形图。

（动画 1）在读周期中，地址信号先有效，以便进行地址译码，选中存储单元。（动画 2）为了读出数据，片选信号 CS 非和读出使能信号 OE 非也必须有效(由高电平变

为低电平)。（动画 3）从地址有效开始经 t_{AQ} 时间，数据总线 I/O 上出现了有效的读出数据。（动画 4）因此，我们把 t_{AQ} 称为读出时间，它是从给出有效地址到外部数据总线上稳定地出现所读出的数据信息所经历的时间。

（动画 5）之后 CS 非、OE 非信号恢复高电平，（动画 6）t_{RC} 以后才允许地址总线发生改变。（动画 7）t_{RC} 称为读周期时间，它是存储芯片进行两次连续读操作时所必须间隔的时间，（动画 8）它总是大于或等于读出时间 t_{AQ}。也就是它们之间要有一个蓝色线条所标记的时间窗口。

（PPT323 第 2 页）在写周期中，（动画 1）也是地址信号先有效，（动画 2）接着片选信号 CS 非有效，（动画 3）写命令 WE 非有效(低电平)，此时数据总线 I/O 上必须置写入数据，（动画 4）在 t_{WD} 时间段将数据写入存储器。（动画 5）之后撤消写命令 WE 非和 CS 非。（动画 6）为了写入可靠，I/O 线的写入数据要有维持时间 t_{hD}, CS 非的维持时间也比读周期长。（动画 7）t_{WC} 时间叫做写周期时间。为了控制方便，一般取 t_{RC}=t_{WC}，通常叫做存取周期。

（动画 8）要实现写操作，要求片选 CS 非和写命令 WE 非信号都为低，并且 CS 非信号与 WE 非信号相“与”的宽度至少应为“写数时间”。

（动画 9）我们要看懂这两个周期波形图，关键要弄清楚地址线、控制线、数据线三组信号线何时有效，以及它们的先后顺序。

（本节讲授完毕）

3.3 DRAM 存储器

3.3.1 DRAM 存储位元的记忆原理

（PPT331 第 1 页）SRAM 的存储元是一个触发器，它具有两个稳定的状态，其优点是速度快，但其容量很难做的太大。而 DRAM 的存储元是由一个 MOS 晶体管和电容器组成的记忆电路，其中 MOS 管做为开关使用，而所存储的信息 1 或 0 则是由电容器上的电荷量来体现，当电容器充满电荷时，代表存储了 1，当电容器放电没有电荷时，代表存储了 0。其存储容量极大，通常用作计算机的主存储器。

下面用四个图分别说明 DRAM 存储元的写、读、刷新操作。在图 3.6(a)中，（动画 1）输出和刷新缓冲器关闭，输入缓冲器打开(R/W 非为低)，（动画 2）输入数据 1 通过输入线 D_{IN} 送到存储元位线上，而行选线为高，打开 MOS 管，于是位线上的高电平给电容器充电，表示存储了 1。该图表示了写 1 到存储元的原理。

（PPT331 第 2 页）在图 3.6(b)中，（动画 1）输出和刷新缓冲器关闭，输入缓冲器打开，（动画 2）输入数据 0 通过输入线 D_{IN} 送到存储元位线上，行选线为高，打开 MOS 管，于是电容上的电荷通过 MOS 管和位线放电，表示存储了 0。该图表示了写 0 到存储元的原理。

（PPT331 第 3 页）在图 3.6(c)中，（动画 1）输入和刷新缓冲器关闭，输出缓冲器打开(R/W 非为高)。行选线为高，打开 MOS 管，（动画 2）电容上所存储的 1 送到位线上，通过输出缓冲器/读出放大器发送到输出线 D_{OUT} 上，即 D_{OUT}=1。该图表示了从存储元读出 1 的原理。

（PPT331 第 4 页）在图 3. 6(d)中，（动画 1）输入缓冲器关闭，刷新和输出缓冲器打开，（动画 2）图 3.6(c)中读到输出线 D_{OUT} 上的数据 1 又经刷新缓冲器送到位线上，再经 MOS 管写到电容上。该图表示了刷新操作的原理。

提醒同学们注意二点：第一点是 DRAM 的读出操作伴随着刷新操作。第二点是输入和输出缓冲器总是互锁的，这是因为读操作和写操作是互斥的，不会同时发生。请同学们课后认真复习前面讲过的“基本的 SRAM 逻辑结构”这部分内容，其中详细说明了读/写互锁逻辑，有助于大家进一步深入理解 DRAM 芯片的读/写互锁机制。

“DRAM 存储位元的记忆原理”就讲到这里，后面将要学习“DRAM 芯片的逻辑结构”，同学们要关注它与 SRAM 芯片在逻辑结构上的不同。

（本节讲授完毕）

3.3.2 DRAM 芯片的逻辑结构

（PPT332 第 1 页）同学们，请问 DRAM 存储芯片是如何存储二进制信息 1 或 0 的？DRAM 的存储元是由一个 MOS 晶体管和电容器组成的记忆电路，其中 MOS 管做为开关使用，电容器上的电荷量体现存储的信息是 1 还是 0，当电容器充满电荷时，代表存储了 1，当电容器放电没有电荷时，代表存储了 0。

掌握了 DRAM 存储芯片的记忆原理之后，我们还需要弄清楚 DRAM 存储器的逻辑结构。图 3.7 表示的是 1M×4 位 DRAM 的逻辑结构。

图 (a)示出了 1M×4 位 DRAM 芯片的管脚图，其中有 2 个电源脚（1 号和 12 号）、2 个地线脚（13 号和 24 号），为了对称，还有 1 个空脚 NC（6 号），有 11 个地址脚（8～11 号对应 A0~A3，14～19 号对应 A4~A9，7 号对应 A10），有 4 个数据脚（2 号、3 号、22 号、23 号分别对应 D0~D3），有 4 个控制脚（4 号、5 号、20 号、21 号分别对

应 WE 写允许、RAS 行选通、OE 读出允许、CAS 列选通信号）。

（PPT332 第 2 页）图 (b)是该芯片的逻辑结构图。DRAM 存储器芯片由存储体与外围电路构成。DRAM 与 SRAM 的不同主要表现在以下三点：

（PPT332 第 3 页）第一点是：**集成度更高，但外围电路更复杂。**这一点通过分析六管 SRAM 存储元电路和四管 DRAM 存储元电路便会理解的更为透彻。附图 332.1 是六管 SRAM 存储元电路示意图。T3、T4 为负载管；T1、T2 为工作管，在 T1 截止、T2 导通或 T2 截止、T1 导通时，为两个稳定状态，且 A、B 两个电位始终相反；T5、T6 、T7、T8 为门控管，控制按地址选择存储元，输出到 I/O、I/O 非线上。由于工作管 T1、T2 总是存在着一定的电容，信息暂存于它们的栅极，而负载管 T3、T4 主要用于补充电荷，且 MOS 的栅极电阻很高，所以泄露电流很小，于是，T1、T2 上的信息电荷可以维持一定的时间。它的外围电路简单，速度快，但其使用的器件多，集成度不高。为了减少管子以提高集成度，把负载管 T3、T4 去掉，就变成四管动态存储元电路，结构如附图 332.2 所示。T1、T2 为工作管，T5、T6、T7、T8 为门控管，T9、T10 为预充管。当 T5、T6、T7、T8 导通时，A、B 点与位线相连、与 I/O、I/O 非相通。从而提高了芯片的集成度，但外围电路有预充管 T9、T10 等其他器件，导致外围电路更加复杂。

（PPT332 第 4 页）第二点是：（动画 1）**增加了行地址锁存器和列地址锁存器。**由于 DRAM 存储器容量很大，地址线宽度相应要增加，这势必增加芯片地址线的管脚数目。为避免这种情况，采取的办法是分时传送地址码。若地址总线宽度为 10 位，先传送地址码 A0～A9，由行选通信号 RAS 非打入到行地址锁存器；然后传送地址码 A10～A19，由列选通信号 CAS 非打入到列地址锁存器。芯片内部两部分合起来，地址线宽度达 20 位，存储容量为 1M×4 位。

第三点是：（动画 2）**增加了刷新计数器和相应的控制电路。**DRAM 读出后必须刷新，而未读写的存储元也要定期刷新，并且要按行刷新，所以刷新计数器的长度等于行地址锁存器。刷新操作与读/写操作是交替进行的，（动画 3）所以通过 2 选 1 MUX 多路开关来提供刷新行地址或正常读/写的行地址。

掌握了 SRAM 和 DRAM 在存储原理和逻辑结构方面的异同，我们就能充分理解 SRAM 为啥速度快、容量却不易做的太大，而 DRAM 为啥能够作为大容量内存条的核心颗粒一直在不断换代升级，尽管它的速度没有 DRAM 快，关于这个问题，同学们学完下一个知识点：DRAM 读/写周期、刷新周期后，不仅会会更加明白 DRAM 的速

度为什么慢于 SRAM，而且还会弄清楚 DRAM 的工作时间关系。

（本节讲授完毕）

3.3.3 DRAM 读/写周期、刷新周期

（PPT333 第 1 页）与 SRAM 一样，要搞清楚 DRAM 的工作时间关系，就需要研究它的读/写周期波形图，同学们从前面学习的内容知道，DRAM 读出时会伴随刷新操作，因此，还需要研究它的刷新周期和刷新方式。

图 3.8(a)为 DRAM 的读周期波形图。当地址线上行地址有效后，（动画 1）用行选通信号 RAS 非(低电平有效)打入行地址锁存器；接着地址线上传送列地址，（动画 2）并用列选通信号 CAS 非(低电平有效)打入列地址锁存器。此时经行、列地址译码，（动画 3）读/写命令 R/W 非=1(高电平有效)，（动画 4）数据线上便有输出数据。由波形图可知，读周期的定义是从行选通信号 RAS 非下降沿开始，到下一个 RAS 非信号的下降沿为止的时间，也就是连续两个读操作的时间间隔。

（PPT333 第 2 页）图 3.8(b)为 DRAM 的写周期波形图。（动画 1）此时读/写命令 R/W 非=0(低电平有效)，在此期间，（动画 2）数据线上必须送入欲写入的数据 D_{IN}(1 或 0)。写周期的定义类似读周期，通常为控制方便，写周期和读周期时间相等。

DRAM 存储位元是基于电容器上的电荷量存储，这个电荷量随着时间和温度而减少，因此必须定期地刷新，以保持它们原来记忆的正确信息。

为什么要定期刷新呢？同学们想知道原因吗？既然大家想知道，那就一起分析一下其主要原因。一次读操作会自动刷新选中行中的所有存储位元，这是前面讲过的内容，然而通常情况下，不能准确地预知读操作出现的频率，也就是说，啥时候读、啥时候写，很难准确预判，把希望寄托在读操作伴随的刷新操作，很容易出现数据丢失，怎么办呢？隔段时间刷新一遍是可行的，这就形成了定期刷新的模式。这个方法看似“笨拙”，但做到了“以防万一”，它能够有效保证数据的完整性。

（PPT333 第 3 页）这样以来，就有了**刷新周期**这个概念，它指的是：从上一次对整个存储器刷新结束到下一次对整个存储器全部刷新一遍为止的时间间隔。

DRAM 的刷新操作有集中式刷新和分散式刷新两种方式。

（PPT333 第 4 页）附图 333.1 表示的是 DRAM 的集中式刷新。对于一个 128×128 阵列的 DRAM 存储器，假定 t_C = 0.5 μ s，则存储器读/写周期为 0.5 μ s，刷新周期为 2ms，可等分为 4000 个存储周期。在一个刷新周期内，（动画 1）前一段时间 3872 个周期(1936

μs）重复进行读/写或维持操作，等到需要进行刷新操作时，便暂停读/写或维持操作，（动画 2）花费 128 个周期（64 μs）逐行刷新整个存储器，这段时间内数据线输出被封锁。等所有行刷新结束后，又开始正常的读/写周期。在集中式刷新方式中，出现了较长的“死时间”（既不能读也不能写），附图 333.1 出现了连续 64 μs 的“死时间”，这对于以读/写为主要操作的存储器来说，显然是很悲哀的现象。因此，这种方式适用于高速存储器。

（PPT333 第 5 页）附图 333.2 表示的是 DRAM 的分散式刷新。对于一个 128×128 阵列的 DRAM 存储器，假定 t_C = 0.5 μs，则存储器读/写周期为 0.5 μs，刷新周期为 2ms，，可等分为 4000 个存储周期。在一个刷新周期内，把每一行的刷新插入到正常的读/写周期之中，从附图 333.2 可知，（动画 1）在 2000/128=15.5 μs 的时间段内花费 0.5 μs 刷新一行，另外 15 μs 进行重复读/写或维持操作。（动画 2）在一个刷新周期内，所有行的刷新操作是分散的，没有出现较长的“死时间”，因此，这种方式成为 DRAM 存储芯片刷新的首选。

（PPT333 第 6 页）附图 333.3 表示了 CAS 非先于 RAS 非的刷新操作。由于 DRAM 芯片内部有刷新计数器，可利用 CAS 非信号比 RAS 非信号提前动作来实现刷新操作——这种时序能保证内部刷新计数器加 1，产生一个需要刷新的行地址，经 2 选 1 MUX 多路开关送到行地址译码器，从而刷新这一行。这种刷新操作巧妙地利用了行列地址分时传送、行列选通存在着时差以及行列地址互锁的特点，而且充分发挥了内部刷新计数器的作用，不需要增加外部刷新地址计数器。

（PPT333 第 7 页）SRAM 和 DRAM 都是半导体随机读写存储器，二者的优点是体积小，可靠性高，价格低廉，缺点是断电后不能保存信息。我们可以用附表 333.1 概括性地对二者进行 7 个方面的比较，SRAM 存储器的存储位元是一个触发器，它具有两个稳定的状态。而 DRAM 存储器的存储位元是由一个 MOS 晶体管和电容器组成的记忆电路。SRAM 没有刷新操作，因而速度比 DRAM 快。而 DRAM 的芯片引脚少于 SRAM，集成度比 SRAM 高，功耗小、价格便宜。因此 DRAM 通常用作计算机的主存储器。

目前，台式机、笔记本电脑的内存容量已达到 32G，而单个半导体存储芯片很难做到这么大的容量，这必然要采取一些方法达到这个目的，究竟如何去做，请同学们关注下一个知识点“存储器容量的扩充”。

（本节讲授完毕）

3.3.4 存储器容量的扩充

（PPT334 第 1 页）由前面所学的内容可知，单个半导体存储芯片的容量比我们需要的存储器容量小，要获得大容量的存储器，解决的办法就是利用若干个存储芯片通过合理的方法进行扩展。在学习存储容量扩充方法之前，同学们首先要弄清楚存储器容量的规范表示形式，通常将存储器容量表示为：字存储容量 × 字长位数，譬如 1M × 32 位。

当存储器的字长位数大于存储芯片的字长位数时，采用字长位数扩展法对存储芯片进行扩充，当存储器的字存储容量大于存储芯片的字存储容量时，采用字存储容量扩展法对存储芯片进行扩充，当存储器的字长位数和字存储容量大于存储芯片的字长位数和字存储容量时，采用字长位数存储容量同时扩展法对存储芯片进行扩充。在学习这部分内容的过程中，同学们要特别注意地址线、数据线和控制线的连接方式。

（PPT334 第 2 页）**1、字长位数扩展**

譬如，用 8K × 1 位的存储芯片扩充形成 8K × 8 位的存储器，需要采用字长位数扩展法，如附图 334.1 所示，需要存储芯片 8 位/1 位=8 个。地址信号 13 位（A0~A12），地址线 13 根，由 8 个芯片公用，控制线也是公用的。每个芯片有 1 根双向数据线，扩充后总共有 8 条双向数据线（D0~D7），与存储器的 I/O 端相链接。由此可见，8 个芯片同时工作，对芯片没有选片要求，存储器的容量和存储芯片的容量一样大，存储的二进制信息并没有增加，只是扩充了字长，提高了存储器的传输能力。

（PPT334 第 3 页）**2、字存储容量扩展**

如果想要存储更多的二进制信息，就得采用字存储容量扩展法对存储芯片进行扩充。譬如，用 16K × 8 位的存储芯片扩充形成 64K × 8 位的存储器，如附图 334.2 所示，需要存储芯片 64K/16K=4 个。存储芯片的地址信号是 $\log_2$（16K）=14 位（A0~A13），地址线是 14 根，扩充后的存储器的地址信号是 $\log_2$（64K）=16 位（A0~A15），地址线是 16 根，其中低位地址信号 A0~A13 由 4 个存储芯片公用，高位地址信号 A14、A15 送到 2：4 译码器获得 4 个信号（00、01、10、11）用作片选，也就是决定 4 个芯片中的哪一个芯片工作，其余的 3 个芯片不能和被选中的芯片同时工作。双向数据线 8 位由 4 个芯片公用，连接着每个芯片的 8 条数据线，亦即字长位数没有变化。读/写控制信号 WE 非由 4 个芯片公用，但片选信号 CE 非不能公用。（动画 1）当 A14 为 1、A15 为 0 时，片选信号为 10，选中的是编号为 2 的存储芯片。当 A14 为 0、A15 为 0

时，片选信号为 10，选中的是编号为 0 的存储芯片，其他 2 个芯片的选择以此类推。

（PPT334 第 4 页）**3、字长位数存储容量同时扩展**

如果既要增大存储容量，也要增加字长，就得采用字长位数存储容量同时扩展法对存储芯片进行扩充。譬如，用 L×K 位的存储芯片扩充形成 M×N 位的存储器，其中 L＜M、K＜N，此时共需要(M/L)×(N/K)个存储芯片。（动画 1）我们要明确这种方法的基本思路，首先利用存储芯片通过位扩展形成芯片组，然后用芯片组通过字位同时扩展或者字扩展形成模块板，最后用模块板通过字扩展形成所需的存储器。关于这个扩充方法，我们可以共同做一道例题来理解其基本思路和做法。（动画 2）在做这类题目时，需要关注三个重点，第一个重点是所需芯片数量的计算，第二个重点是译码器输入、输出的确定，第三个重点是 CPU 如何选择模块板或芯片组。

（PPT334 第 5 页）【附例 334.1】的题干告诉我们：16 位计算机，地址码 18 位，SRAM 芯片是 4K×8 位的，模块板是 32K×16 位的。要求我们：计算模块板的数量、模块板内 SRAM 芯片的数量、整个存储器中 SRAM 芯片的数量，陈述 CPU 选择模块板的基本方法。

（动画 1）由地址码 18 位可知主存容量为 2^{18}=256K，主存共需 256K/32K=8 块模块板。关于已知地址码位数、计算存储容量，或者已知存储容量、计算地址码位数、确定地址线数量的方法，请同学们务必熟练掌握。（动画 2）32K 模块板的地址信号是 $\log_2$（32K）=15 位（A0～A14），256K 存储器的地址信号是 $\log_2$（256K）=18 位（A0～A17），其中低位地址信号（A0~A14）用于模块板寻址，高位地址信号 A17、A16、A15 通过 3：8 译码器输出进行模块板的选择。

（动画 3）由 16 位/8 位=2 可知，用 2 片 4K×8 位的存储芯片形成 4K×16 位的芯片组，容量没变，不存在片选问题，然后由 32K/4K=8 可知，用 8 个 4K×16 位的芯片组形成 32K×16 位的模块板，所以模块板内共需（32K/4K）×（16 位/8 位）=8×2=16 片 SRAM。（动画 4）4K×16 位芯片组的地址信号是 $\log_2$（4K）=12 位（A0～A11），32K×16 位模块板的地址信号是 $\log_2$（32K）=15 位（A0～A14），其中低位地址信号（A0~A11）用于芯片组寻址，高位地址信号 A12、A13、A14 通过 3：8 译码器输出进行芯片组的选择。

（动画 5）由上述计算可知，整个主存需要 8 个模块板，而每个模块板内共有 16 片 SRAM 芯片，所以主存共需 8（块）×16（片/块）=128 片 SRAM 芯片。

（PPT334 第 6 页）**4、存储器模块条**

存储器通常以插槽用模块条形式供应市场。这种模块条常称为内存条，它们是在一个条状形的小印制电路板上，用一定数量的存储器芯片，组成一个存储容量固定的存储模块，CPU 可通过总线寻址访问它，并进行读写操作，如图 3.11 所示。

自 1991 年起，内存条的发展主要经历了 EDO DRAM、SDRAM、DDR、DDR2、DDR3 和 DDR4 六个阶段。

EDO DRAM（Extended Date Out RAM，外扩充数据模式存储器）内存，这是 1991 年到 1995 年之间盛行的内存条。它取消了扩展数据输出内存与传输内存两个存储周期之间的时间间隔，在把数据发送给 CPU 的同时去访问下一个页面，故而速度要比普通 DRAM 快 15~30%。

自 Intel Celeron 系列以及 AMD K6 处理器以及相关的主板芯片组推出后，EDO DRAM 内存性能再也无法满足需要了，内存技术必须彻底得到个革新才能满足新一代 CPU 架构的需求，此时内存开始进入比较经典的 SDRAM 时代。SDRAM 内存前后经历了 PC66、PC100、PC133、PC150、PC166 规范。

DDR SDRAM(Dual Data Rate SDRAM)简称 DDR，也就是“双倍速率 SDRAM“的意思。DDR 可以说是 SDRAM 的升级版本，DDR 在时钟信号上升沿与下降沿各传输一次数据，这使得 DDR 的数据传输速度为传统 SDRAM 的两倍。由于仅多采用了下降缘信号，因此并不会造成能耗增加。至于定址与控制信号则与传统 SDRAM 相同，仅在时钟上升缘传输。

DDR2（Double Data Rate 2）SDRAM 与 DDR 内存最大的不同就是，虽然都采用了在时钟的上升/下降延同时进行数据传输的基本方式，但 DDR2 内存却拥有两倍于上一代 DDR 内存预读取能力（即：4bit 数据读预取）。换句话说，DDR2 内存每个时钟能够以 4 倍外部总线的速度读/写数据，并且能够以内部控制总线 4 倍的速度运行。

DDR3 相比 DDR2 有更低的工作电压，从 DDR2 的 1.8V 降落到 1.5V，性能更好更为省电; DDR2 的 4bit 预读升级为 8bit 预读。DDR3 最高能够达到 2400Mhz 的速度。

DDR4 内存将会拥有两种规格。其中使用传统 SE 信号（Single-ended Signaling）的 DDR4 内存的传输速率为 1.6～3.2Gbps，而基于差分信号技术的 DDR4 内存的传输速率达到 6.4Gbps。由于通过一个 DRAM 实现两种接口基本上是不可能的，因此 DDR4 内存将会同时存在基于传统 SE 信号和差分信号的两种规格产品。

目前的 PC 机配置的就是 DDR4 内存，容量已达到 32GB。但计算机对于存储器速度的要求从未停歇，我们将在下一个知识点学习并行存储器，从中能够体会 CPU 和存

储器之间的速度匹配问题为何成为计算机领域的研究热点之一。

（本节讲授完毕）

3.4 并行存储器

3.4.1 双端口存储器

（PPT341 第 1 页）前面我们共同学习了 SRAM 和 DRAM，这两种半导体随机读写存储器的缺点是断电后不能保存信息。同学们自学的只读存储器和闪速存储器，正好弥补了 SRAM 和 DRAM 的缺点，即使断电也仍然保存原先写入的数据。特别是闪速存储器能提供高性能、低功耗、高可靠性以及移动性，是一种全新的存储器体系结构。同学们知道，存储器的四个技术指标中，其中用存取时间、存储周期、存储器带宽这三个指标来衡量存储器的速度性能，可见，存储器速度是人们关注的重中之重。

这么多年来，存储器的速度在提升，而 CPU 的速度提升更快，导致传统存储器的和 CPU 的速度不匹配，仍然感觉存储器的速度还是慢。（动画 1）造成这种情况的主要原因有两个，（动画 2）第一个原因是 CPU 和主存储器是用不同的材料制成的；（动画 3）第二个原因是一个 CPU 周期可能需要几个存储器字。（动画 4）这种情况便成为限制高速计算的主要问题，主要的解决方法有四种，（动画 5）第一种方法是采用更高速的主存或增加主存的字长；（动画 6）第二种方法是采用空间并行的双端口存储器；（动画 7）第三种方法是在 CPU 和主存之间插入一个高速缓冲存储器；（动画 8）第四种方法是采用时间并行的多体交叉存储器。

（PPT341 第 2 页）**1、双端口存储器的逻辑结构**

双端口存储器具有两组相互独立的读写控制电路，是一种高速工作的存储器。它属于并行存储器，采用空间并行技术。

图 3.24 表示了 2K × 16 位双端口存储器 IDT7133 的逻辑框图。这是一个存储容量为 2K 字长 16 位的 SRAM，它提供了两个相互独立的端口，即左端口和右端口。它们分别具有各自的地址线（A0 ~ A10）、数据线(I/O_0 ~ I/O_{15})和控制线(R/W 非，CE 非，OE 非，BUSY 非)，可以对存储器中任何位置上的数据进行独立的存取操作。图 3.24 中，相关标识符的下标中 L 表示左端口，R 表示右端口，LB 表示低位字节，UB 表示高位字节。这里需要说明一下，DRAM 也能做成双端口存储器。

（PPT341 第 3 页）**2、无冲突读写控制**

当两个端口传送的地址不相同时，在两个端口上进行读写操作，一定不会发生冲

突。对于图 3.24 来说，当任一端口被选中驱动时，就可对整个存储器进行存取，每一个端口都有自己的片选控制(CE 非)和输出驱动控制(OE 非)。读操作时，端口的 OE 非(低电平有效)打开输出驱动器，由存储矩阵读出的数据就出现在 I/O 线上。

我们用附图 341.1 作为双端口存储器无冲突读写动画演示的底图，譬如，（动画 1）左端口输入地址 2、右端口输入地址 1，便会在两个端口独立的控制信号作用下，（动画 2）分别从左右端口将地址 2 的数据 00010001、地址 1 的数据 10010011 输出到左右端口的数据线上。

（PPT341 第 4 页）再譬如，（动画 1）左端口输入地址 1、右端口输入地址 3，便会在两个端口独立的控制信号作用下，（动画 2）分别从左右端口将地址 1 的数据 10010011、地址 3 的数据 11111000 输出到左右端口的数据线上。这种情况下，不会发生冲突。

（PPT341 第 5 页）双端口存储器无冲突的读/写条件可概括为表 3.4，该表中符号 1 代表高电平，0 代表低电平，X 代表任意，Z 代表高阻态。

从表 3.4 的 $I/O_0 \sim I/O_7$ 和 $I/O_8 \sim I/O_{15}$ 两列一眼可以看出来，有 6 个“数据入”，有 4 个“数据出”，有 6 个“Z”。这是怎么回事呢？现在我告诉大家其中蕴含的规则。当片选控制 CE 非为 1 时，端口不用。当片选控制 CE 非为 0 时，可进行读写操作。再当读写控制 R/W 非为 0 时，将对应端口的低位字节数据线($I/O_0 \sim I/O_7$)或高位字节数据线($I/O_8 \sim I/O_{15}$)上的数据写入存储器。当 R/W 非和输出驱动控制 OE 非分别为 1 和 0 时，存储器中的数据从对应端口的低位字节数据线上或高位字节数据线上输出。

当 R/W 非和 OE 非分别为 1 和 1 时，对应端口的低位字节数据线或高位字节数据线呈高阻态。譬如，表中第三行，当 CE 非、OE 非分别为 0、0，R/W_{LB} 非为 0 时，$I/O_{0\sim7}$ 数据入，低位字节数据写入存储器；（动画 1）当 CE 非、OE 非分别为 0、0，R/W_{UB} 非为 1 时，$I/O_{8\sim15}$ 数据出，存储器中数据输出至高位字节。其他情况不在课堂上逐一进行具体分析，请同学们按照我刚才讲的规则自己进行分析。

（PPT341 第 6 页）**3、有冲突的读写控制**

当两个端口均为开放状态(BUSY 非为高电平)，且同时访问存储器同一个存储单元进行存取时，便发生读写冲突。在这种情况下，片上的判断逻辑可以决定对哪个端口优先进行读写操作，而对另一个被延迟的端口置 BUSY 非标志(BUSY 非变为低电平)，即暂时关闭此端口。换句话说，读写操作对 BUSY 非变为低电平的端口是不起作用的。一旦优先端口完成读写操作，才将被延迟端口的 BUSY 非标志复位(BUSY 非变

为高电平)，开放此端口，允许延迟端口进行存取。

我们再用附图 341.1 作为双端口存储器有冲突读写动画演示的底图，譬如，（动画 1）左右端口输入的都是地址 2，请问同学们，会从哪个端口读出地址 2 的数据呢？也许有些同学会说，两个都可以。显然，这样的回答有点随意了。此时，需要片上的判断逻辑来做决定。我们来看动画演示，（动画 2）右端口的 BUSY 非变为低电平，左端口的 BUSY 非变为高电平，可知片上的判断逻辑优先决定左端口作为读写操作的端口，右端口就会被关闭，（动画 3）地址 2 的数据 00010001 便会输出到左端口的数据线上。

（PPT341 第 6 页）双端口存储器片上的判断逻辑通常有 CE 非判断和地址有效判断两种方式。

（动画 1）第一种是 CE 非判断，它指的是如果地址匹配且在 CE 非之前有效，片上的控制逻辑在 CE_L 非和 CE_R 非之间进行判断来选择端口。图 3.25(b) 表示用 CE 非判断的冲突周期时序波形，地址匹配先于 CE 非有效，而 CE_L 非先于 CE_R 非有效，因此，$BUSY_R$ 非置 0，右端口被关闭，左端口获得了优先读写操作权限。

（动画 2）第二种是地址有效判断，它指的是如果 CE 非在地址匹配之前变低，片上的控制逻辑在左、右地址间进行判断来选择端口。

无论采用哪种判断方式，延迟端口的 BUSY 非标志都将置位而关闭这个端口，而当允许存取的端口完成操作时，延迟端口的 BUSY 非标志才进行复位而打开这个端口。

有些显示存储器是双端口的，也就是在显存中增加了一个端口，它可以从显示芯片中得到数据的同时向数/模转换电路输送数据，提高了显示带宽。这种形式的显示存储器价格高，多用在图形处理工作站上。

本节学习的双端口存储器采用了空间并行技术，同学们课后通过温习这部分内容以及查阅资料，进一步弄明白双端口存储器发明的科学意义和工程意义。

下一个知识点是关于多模块交叉存储器的，它采用了时间并行技术，对于提高计算机的访存速度也做出了重要贡献，希望同学们用心关注其逻辑结构和工作原理。

（本节讲授完毕）

3.4.2 多模块交叉存储器

1、存储器的模块化组织

（PPT342 第 1 页）同学们已经学习了采用空间并行技术的双端口存储器，它具有两组相互独立的读写控制电路，是一种高速工作的存储器。当两个端口传送的地址不

相同时，在两个端口上进行读写操作，不会发生冲突。可是当两个端口均为开放状态，且同时访问存储器同一个存储单元进行存取时，便会发生读写冲突。为此，研究人员沿着时间并行这条技术路线，假设同时从主存取出多条指令或多个存储字，从而提高计算机的运行速度，以此作为基本思想，研究实现了采用模块化组织的多模块交叉存储器，它采用线性编址，能并行执行多个独立的读写操作，属于并行存储器。

多模块交叉存储器线性编址的方式有两种，一种是顺序方式，另一种是交叉方式。

（PPT342 第 2 页）图 3.26(a)表示的是顺序编址方式。（动画 1）图中的存储器容量为 32 字，分成 M0、M1、M2、M3 四个模块，每个模块存储 8 个字。（动画 2）访问地址按顺序分配给一个模块后，譬如线性地址 0～7 分配在 M0 模块，接着又按顺序为下一个模块分配访问地址，8～15 分配在 M1 模块，16～23 分配在 M2 模块，24～31 分配在 M3 模块。这样，（动画 3）存储器的 32 个字可由 5 位地址寄存器来表示，其中高 2 位选择 4 个模块中的一个，（动画 4）低 3 位选择每个模块中的 8 个字。譬如，0～7 号字的高两位都为 00，与模块号 0 是一致的，低三位从 000 到 111，5 位地址正好从 0～7；8～15 号字高两位都为 01，与模块号 1 是一致的，低三位从 000 到 111，5 位地址正好从 8～15，其他的依次类推。

图 3.26(a)表明，在顺序方式中某个模块进行存取时，其他模块不工作。而某一模块出现故障时，其他模块可以照常工作，另外通过增添模块来扩充存储器容量也比较方便。但其缺点是各模块一个接一个串行工作，因此存储器的带宽受到了限制。

（PPT342 第 3 页）为了解决顺序组织方式中带宽受限的问题，研究形成了交叉组织方式，如图图 3.26(b)所示。（动画 1）图中的存储器容量也是 32 字，也分成四个模块，每个模块存储 8 个字。连续地址分布在相邻的不同模块内，先将 4 个线性地址 0、1、2、3 依次分配给 M0、M1、M2、M3 模块，再将线性地址 4、5、6、7 依次分配给 M0、M1、M2、M3 模块，依此类推，直到全部线性地址分配完毕为止。这样以来，同一个模块内的地址都是不连续的。（动画 2）当存储器寻址时，用高 3 位选择模块中的 8 个字，（动画 3）用地址寄存器的低 2 位选择 4 个模块中的一个。地址码的低位字段经过译码选择不同的模块，而高位字段指向相应模块内的存储字。

譬如，模块 M0 中存放的 8 个字的地址低两位都为 00，与模块号 0 是一致的，高三位从 000 到 111，5 位地址分别是 0、4、8、12、16、20、24、28；模块 M1 中存放的 8 个字的地址低两位都为 01，与模块号 1 是一致的，高三位从 000 到 111，5 位地址分别是 1、5、9、13、17、21、25、29，其他的依次类推。由此可以发现，线性地

址对模块数求模即求得的余数便是该地址所在的模块号。譬如线性地址 0、4、8、12、16、20、24、28 对模块数 4 求余的结果都为 0，因此，它们被分配到模块 M0 中；线性地址 1、5、9、13、17、21、25、29 对模块数 4 求余的结果都为 1，因此，它们被分配到模块 M1 中，其他的依此类推。

在交叉组织方式中，对连续字的成块传送可实现**多模块流水式并行存取**，能够大大提高存储器的带宽，弥补了顺序组织方式存在的带宽受限的缺点。

2、多模块交叉存储器的基本结构

（PPT342 第 4 页）图 3.27 表示的是四模块交叉存储器结构框图。（动画 1）主存被分成 4 个相互独立、容量相同的模块 M0、M1、M2、M3，每个模块都有自己的读写控制电路、地址寄存器和数据寄存器，各自以等同的方式与 CPU 传送信息。存储器地址交叉的方式是采用模除的方法，即二进制地址的低位表示该单元所在的模块。

对于每一个存储模块来说，从 CPU 给出访存命令直到读出信息仍然使用了一个存取周期时间，而对 CPU 来说，它可以在一个存取周期内连续访问四个模块，由存储器控制部件控制它们分时使用数据总线进行信息传递。各模块的读写过程将重叠进行，在理想情况下，如果程序段或数据块都是连续地在主存中存取，那么将大大提高主存的访问速度，所以多模块交叉存储器是一种并行存储器结构。

（PPT342 第 5 页）前面所学内容表明，顺序组织方式的带宽受限，而交叉组织方式弥补了这一缺点。关于这个问题，我们可以用定量分析方法来证明。

当模块存取一个字的存储周期为 T，总线传送周期为 τ，存储器的交叉模块数为 m，图 3.28 表示了 m=4 的流水线方式存取示意图。成块传送可按 τ 间隔流水方式进行，M0 模块启动后，经 τ 时间延迟后启动 M1，依此类推。m 个模块不能在同一时刻启动，只有在一个存储周期内分时使用总线传送数据，才能满足交叉存取的要求。这种情况下应当满足 $m \geqslant T/\tau$，以保证启动某模块后经 mτ 时间再次启动该模块时，它的上次存取操作已经完成。T/τ 称为交叉存取度，它受限于存储器和总线的性能，也就成为交叉存储器选择模块数量的依据。

基于流水线方式交叉存取的思想，连续读取 m 个字所需的时间为 $t1 = T+(m-1)\tau = (2m-1)\tau$，而顺序方式存储器连续读取 m 个字所需时间为 $t2 = mT = m^2\times\tau$，由此可得到交叉存取方式的带宽 $W1 = 1/t1 = 1/(2m-1)\,\tau$，顺序存取方式的带宽 $W2 = 1/t2 = 1/m^2 \times \tau$，因为 $t1 < t2$，所以 $W1 > W2$。

（PPT342 第 6 页）下面我们做一下例 5，用计算所得的量化结果帮助我们理解上

述结论。题目告诉我们：设存储器容量为 32 字，字长 64 位，模块数 m = 4，分别用顺序方式和交叉方式进行组织。若存储周期 T = 200ns，数据总线宽度为 64 位，总线传送周期 τ = 50ns，若连续读出 4 个字，问：顺序存储器和交叉存储器带宽各是多少？

解题步骤如下：

信息总量：q = 64 位字长 × 模块数 4 =256 位

交叉存储器和顺序存储器读出 4 个字的时间分别是：

t1 = T + (m – 1)τ = 200 + 3×50 = 3.4 ×10 –7 (s)

t2 = m T = 4×200ns =8×10 –7 (s)

交叉存储器带宽：W1 = q / t1 = 73 × 107 （位/ S）

顺序存储器带宽：W2 = q / t2 = 32 × 107 （位/ S）

由计算结果可知交叉存储器带宽大于顺序存储器带宽。

3、二模块交叉存储器举例

（PPT342 第 7 页）图 3. 29 表示了二模块交叉存储器方框图。（动画 1）每个模块的容量为 1MB（256K × 32 位），由 8 片 256K × 4 位的 DRAM 芯片组成(位扩展)。二模块的总容量为 2MB (512K × 32 位)。（动画 2）数据总线宽度为 32 位，地址总线宽度为 24 位。为简化，将 2 片 DRAM 芯片用一个 256K × 8 位的长条框表示。

（PPT342 第 8 页）DRAM 存储器有读周期、写周期和刷新周期。存储器读/写周期时，（动画 1）在行选通信号 RAS 非有效下输入行地址，在列选通信号 CAS 非有效下输入列地址，于是芯片中行列矩阵中的某一位组被选中。如果是读周期，此位组内容被读出；如果是写周期，将总线上数据写入此位组。（动画 2）刷新周期是在 RAS 非有效下输入刷新地址，此地址指示的一行所有存储元全部被再生。刷新周期比读/写周期有高的优先权，当对同一行进行读/写与刷新操作时，存储控制器对读/写请求予以暂存，延迟到此行刷新结束后再进行。

（PPT342 第 9 页）由图 3.29 可以看出，（动画 1）24 位的存储器物理地址指定的系统主存总容量可达 16MB，按照“存储体–块–字”进行寻址。（动画 2）其中高 3 位用于存储体选择(字扩展)，1 个体为 2MB，全系统有 8 个 2MB 存储体。（动画 3）A3 ~ A20 的 18 位地址用于模块中 256K 个存储字的选择。读/写周期时，它们分为行、列地址两部分送至芯片的 9 位地址引脚。（动画 4）一个模块内所有芯片的 RAS 非引脚连接到一起，模块 0 由 RAS0 驱动，模块 1 由 RAS1 驱动。（动画 5）A2 用于模块选择(字扩展)，在读/写周期时，（动画 6）主存地址中 A2=0，RAS0 有效；（动画 7）A2=1，

RAS1 有效。连续的存储字(32 位，双字)交错分布在两个模块上，偶地址在模块 0，奇地址在模块 1。

（PPT342 第 10 页）（动画 1）CPU 给出的主存地址中没有 A1、A0 位，替代的是 4 个字节允许信号 BE3～BE0，对应着 CAS3～CAS0，以允许对 A2～A23 指定的存储字(双字)中的字节或字完成读/写访问。当 BE3～BE0 全有效时，即完成双字存取。

DRAM 存储器需要逐行定时刷新，以使不因存储信息的电容漏电而造成信息丢失。另外，DRAM 芯片的读出是一种破坏性读出，因此在读取之后要立即按读出信息予以充电再生。这样，若 CPU 先后两次读取的存储字使用同一 RAS 选通信号的话，CPU 在接收到第一个存储字之后必须插入等待状态，直至前一存储字再生完毕才开始第二个存储字的读取。为避免这种情况，模块 0 由 RAS0 驱动，模块 1 由 RAS1 驱动。

通过前面两节的学习，我们知道双端口存储器和多模块交叉存储器属于（并行）存储器结构。双端口存储器采用（空间）并行技术，多模块交叉存储器采用（时间）并行技术。这两种类型的存储器在科研和工程中得到了广泛的使用。双端口存储器在（左端口与右端口的地址码相同）情况下会发生读/写冲突。交叉存储器实质上是一种（模块式）存储器，它能（并行）执行（多个）独立的读写操作。

解决存储器和 CPU 速度不匹配问题的方法有四种，同学们还记得第三种方法是什么吗？只有部分同学记得，是在 CPU 和主存之间插入一个高速缓冲存储器，这种存储器有个英文名称叫 Cache，它在解决主存和 CPU 之间速度不匹配问题上发挥了至关重要的作用，具体情况如何，学完下一个知识点，同学们便会晓其厉害。

（本节讲授完毕）

3.5 cache 存储器

3.5.1 cache 的基本原理

（PPT351 第 1 页）**1、Cache 的功能**

早期的计算机只有主存和外存，没有 Cache，如附图 35.1 所示，数据和指令在 CPU 和主存中进行交换，而主存是 MOS 半导体性质，它的存取速度明显慢于 CPU 的运算速度，也就是说，系统读、写一次主存的过程中，CPU 往往要等待一段时间，使得 CPU 的高速运算能力没有得到充分发挥，出现了 CPU 和主存之间速度不匹配的问题。用什么技术可以解决这一问题呢？

（动画 1）使用 Cache。形成了高速缓冲存储器（即 Cache）、主存储器和外存储

器的分级存储体系结构，如图 3.31（P.90）所示。因为 Cache 是一种高速缓冲存储器，半导体性质，由高速 SRAM 组成，加上控制逻辑，全部功能由硬件实现，容量虽小，但存取速度快，能高速向 CPU 提供指令和数据，加快程序的执行速度。（动画 2）现在的 Cache 分片内 Cache 和片外 Cache，片内 Cache 的速度已接近 CPU 速度。

由此可见，Cache 是为了解决 CPU 和主存之间速度不匹配而采用的一项重要的硬件技术，并且发展为多级 cache 体系，（动画 3）指令 cache 与数据 cache 分设体系。

（PPT351 第 2 页）**2、Cache 基本原理**

Cache 是如何解决 CPU 和主存之间速度匹配这一问题的呢?

这要从 Cache 的理论基础——程序访问的局部性原理谈起。所谓程序访问的局部性原理是指在某一个较短的时间间隔内，CPU 对局部范围的内存地址频繁访问，而对此地址范围之外的地址访问很少。这样以来，降低了 CPU 的运算速度，出现了 CPU 和主存之间速度匹配的问题。如果将频繁访问的地址空间中存储的内容放到 Cache 之中，让 CPU 从 Cache 中读取指令或数据，那么就能提高 CPU 访存的速度。（动画 1）程序的局部性包括时间局部性、空间局部性等，例如，循环程序。

（动画 2）譬如，1 加到 100，可以用下面的循环程序来实现。

```
int i;
long s=0;
for(i=1;i<=100;i++)
{s=s+i;}
```

在这个程序编译时，系统会给整型变量 i、长整型变量 s 和加法指令在主存中分配存储地址，在程序执行过程中，这几个存储地址所指向的存储单元被频繁访问，而主存的速度相对较慢，很显然，程序执行的速度会因此下降，如果将频繁访问的 i、s 和加法指令放到 Cache 之中，让 CPU 从 Cache 中读取指令或数据，那么就能提高 CPU 访存的速度，继而提高程序的运行速度。

（PPT351 第 3 页）图 3.32（P.91）是 Cache 的工作原理图。（动画 1）CPU 与 cache 之间的数据交换是以字为单位，（动画 2）而 cache 与主存之间的数据交换是以块为单位。（动画 3）图中 cache 分为 4 行，每行 4 个字。（动画 4）分配给 cache 内所有的地址存放在一个相联存储器 CAM 中，它是按内容寻址的。（动画 5）当 CPU 执行访存指令时，就把所要访问的字的地址送到 CAM 和主存。（动画 6）如果 CAM 指出所要访问的字 W 在 cache 中，则把 W 从 cache 传送到 CPU;（动画 7）如果 W 不在 cache 中，

则将 W 从主存传送到 CPU，与此同时，把包含 W 的由前后相继的 4 个字所组成的一行数据块送入 cache。（动画 8）替换算法采用 LRU。

通过 cache 的功能和工作原理可知，cache 能提高存储系统的速度，然而提高到什么程度，用什么指标来衡量呢？

（PPT351 第 4 页）**3. cache 的命中率**

用命中率（h）、cache/主存系统的平均访问时间（ta）、访问效率（e）来衡量。（动画 1）所谓命中率是指在一个程序执行期间，cache 完成存取的总次数与 cache/主存完成存取总次数之和的比值。（动画 2）即 $\mathbf{h=N_c/(N_c+N_m)}$ （公式 3.5）

（动画 3）cache/主存系统的平均访问时间 $\mathbf{t_a=ht_c+(1-h)t_m}$ （公式 3.5） 其中 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。

（动画 4）访问效率 $\mathbf{e=t_c/t_a=1/[h+(1-h)r]=1/[r+(1-r)h]}$（公式 3.7）（动画 5）其中 $r=t_m/t_c$

结论：h→1，e↑，(r=5 ~ 10)

（动画 6）h 与程序的行为、cache 的容量、组织方式、块的大小有关。（譬如循环结构的程序要被多次重复地执行。程序中大部分指令是顺序存储和顺序执行，数据一般也簇聚地存储在一起。）

（PPT351 第 5 页）下面我们共同完成【附例 351.1】的求解，帮助我们理解刚才所讲的三个指标的计算方法。题目告诉我们：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 45 纳秒，主存的存取周期为 200 纳秒。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：(1) cache 的命中率是多少？(2) CPU 访问内存的平均时间是多少纳秒？(3) Cache-主存系统的效率是多少？

解题步骤如下：

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

（动画 2）(2) CPU 访存的平均时间

$$Ta=H\cdot Tc+(1-H)Tm=0.92\times 45+(1-0.92)\times 200=57.4ns$$

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

（动画 4）从上例可以看出，加入 cache 之后，cache/主存系统的平均访问时间为 57.4ns，没加 cache 时，主存访问时间为 200ns，相比而言，加入 cache 之后使存储系统的速度大幅度提高。我们期望 cache 的命中率接近于 1，这成为业界为之奋斗的动力。

讲到这里，我们就能掌握 cache 的基本原理，但其中有个这内容似乎给我们一点启示，那就是 CPU 与 cache 之间的数据交换是以字为单位，而 cache 与主存之间的数据交换是以块为单位，这种不同要求主存与 cache 的地址之间必须遵守某种映射规则，否则，无法将主存块放到 cache 中合适的位置。这便是下一个要学的知识点——主存与 cache 的地址映射。

（本节讲授完毕）

3.5.2 主存与 cache 的地址映射

（PPT352 第 1 页）同学们，上一节学习了 cache 的基本原理，从学过的内容可知，cache 的容量很小，它保存的内容只是主存内容的一个子集，它与主存的数据交换是以块为单位，并将主存块放到 cache 中，究竟放到什么位置，须要地址映射来实现。所谓地址映射就是应用某种方法把主存地址定位到 cache 中，从而把主存块放到 cache 相应位置。（动画 1）地址映射过程全部由硬件实现，对程序员透明。

（动画 2）地址映射如何实现有以下三种方式：全相联、直接映射、组相联。

1、全相联映射方式

（PPT352 第 2 页）首先来学习全相联映射方式。图 3.33（a）（P.93）是全相联映射的 cache 组织。（动画 1）假定示意图中 cache 为 8 行，主存为 256 块，（动画 2）主存的每一块都可映射到 cache 任意一行中，譬如，（动画 3）B0 主存块可映射到 cache 的第 0 行，也可影射到 cache 的第 1 行，也可映射到 cache 的第 7 行，依次类推，（动画 4）B1、B2…B255 均可映射到 cache 的任意一行。

（PPT352 第 3 页）检索过程如图 3.33（b）（P.93）所示：（动画 1）CPU 访存指令指定一个内存地址，它由块号（高位）和字 （低位）组成。（动画 2）为了加快检索，指令中的块号与 cache 中所有行的标记同时在比较器中比较，（动画 3）如果块号命中，则按字节从 cache 中读取一个字，（动画 4）如果块号不命中，则按内存地址读取这个字，同时把内存块读入 cache 行中。

（动画 5）优点：可以使主存的一个块直接拷贝到 cache 中的任意一行上，非常灵活，不易产生冲突。

（动画 6）缺点：比较器电路复杂（假定 cache 有 n 行，比较器就有 n+1 路输入），难于设计和实现，效率低，速度慢。因此适合于小容量 cache 采用。

但是，在实际设计中，我们需要 cache 的容量大一些，硬件简单一些，cache 的命中率高一些，为此，我们采用一种多对一，并且一个主存块只能拷贝到 cache 的一个特定位置上去的映射方式，即直接映射方式，用以简化比较器电路。

2、直接映射方式

（PPT352 第 4 页）图 3.34（a）（P.94）是直接映射的 cache 组织。（动画 1）假定示意图中 cache 为 8 行，主存为 256 块，（动画 2）主存的每一块只可映射到 cache 特定一行中，cache 的行号 i 和主存的块号 j 有如下函数关系：　i=j mod m　（m 为 cache 中的总行数）。譬如，（动画 3）B0、B8…B8k 主存块只可映射到 cache 的第 0 行，（动画 4）B7、B15…B8k+7 只可映射到 cache 的第 7 行。至于 B0、B8…B8k 映射到 L0，而不是 B0、B1…B31 映射到 L0 是因为基于程序局部性原理。

（PPT352 第 5 页）检索过程如图 3.34（b）（P.94）:（动画 1）CPU 访存指令指定一个内存地址，它由 tag、行号和字组成。相当于全相联映射中的块号分解成两部分：tag 和行号。（动画 2）按地址 r 位行号找到 cache 中的此一项，然后用地址中的（s-r）位标记（tag）部分与此行的标记在比较器中做比较。如果相符即命中，在 cache 中找到所要求的块，然后用地址中最低的 W 位读取所需的字，（动画 3）如果不符合则不命中，从主存中读取所需要的块。

（动画 4）优点：硬件简单，成本低。

（动画 5）缺点：容易产生冲突，不能有效利用 Cache 空间。

由此可见，前面两种映射方式的优缺点正好相反，为了取长补短，结合二者的优点又尽量避免其缺点，形成另外一种映射方式——组相联映射方式。从灵活性、命中率、硬件投资来说较为理想，因而得到了普遍采用。

3、组相联映射方式

（PPT352 第 6 页）图 3.35（a）（P.95）是组相联映射的 cache 组织。（动画 1）它将 cache 分成 u 组，每组 v 行，主存块存放到哪个组是固定的，至于存到该组哪一行是灵活的，即有如下函数关系：m = u × v　组号　q = j mod u 。（动画 2）假定示意图中 cache 为 8 行，主存为 256 块，cache 分为 4 组，每组 2 行，主存的每一块可以存入 cache 特定一组的任意一行中，譬如，（动画 3）B0、B4…B252 可以存入 cache 的 S0 组的任意一行中。（动画 4）B1、B5…B253 可以存入 cache 的 S1 组的任意一行中。

同理，B2、B6…B254 及其它。

（PPT352 第 7 页）检索过程如图 3.35（b）（P.95）所示:（动画 1）CPU 访存指令指定一个内存地址，它由 tag、组号和字组成。（动画 2）首先用块号域的第 d 位找到 cache 的相应组，（动画 3）然后将块号域的高（s-d）位与该组行中的所有标记同时进行比较。（动画 4）如果有一行的标记与之相符，则此行命中，以内存地址的 W 位字域部分检索此行的具体字，并完成所需要求的存取操作。（动画 5）如果任意行的标记与之不相符，则 cache 不命中，按内存地址访问主存。组间是直接映射方式，组内是全相联映射方式。

（动画 6）组相联映射方式中的每组行数 v 一般取值较小，这种规模的 v 路比较器容易设计和实现。而块在组中的排放又有一定的灵活性，冲突减少。

主存与 cache 的地址映射已经全部讲完，你能说出上述三种映射方式的优缺点吗?在下一节学习替换策略时，同学们就会进一步明白三种映射方式的特点。

（本节讲授完毕）

3.5.3 替换策略

（PPT353 第 1 页）同学们，请问 cache 的工作原理是什么？通过前面的学习，我们知道 cache 遵循的是程序访问的局部性原理，而且它是一类速度快、容量小的存储器，这就要求它尽量保存最新数据，新旧数据必然要产生替换。

探讨替换这一问题，不得不考虑 cache 与主存之间的三种映射规则，（动画 1）对直接映射而言，只要把某特定位置上的原主存块换出 cache 即可。

（动画 1）对全相联的 cache 来说，按照映射规则，新主存块可以存放在 cache 的任意一行，可事实不能这么做，因为某些行可能是新调入行且存放着当前最重要的数据，一旦被替换便会导致程序运行错误。对组相联的 cache 来说，按照映射规则，新主存块可以存放在 cache 特定一组中的任意一行，同样不能这么去做，原因和全相联的 cache 相似。这就要求赋予计算机某种替换策略（也称为替换算法），使其选取特定的某一行用来存放新主存块。（动画 3）常用的 cache 替换策略有：最不经常使用(LFU，Least Frequently Used)算法、近期最少使用(LRU，Least Recently Used)算法和随机替换。

（动画 4）我们先简单了解一下随机替换策略。随机替换策略是指从 cache 的特定行中随机地选取一行换出。这种策略在硬件上容易实现，而且速度比前两种策略都快。缺点是一旦新调入行的数据被替换，再次访问这一行时容易出现未命中的情况，从而

导致 cache 的命中率和工作效率降低。

（PPT353 第 2 页）下面我们重点学习 LFU 和 LRU 策略。这两种策略的英文缩写只有一个字母之差，但它们的原理有着很大的区别，请同学们仔细琢磨其间的异同。

LFU 和 LRU 策略都给 cache 的每行设置了一个计数器，标记为 Pi。起初，LFU 给所有的新行 Pi 置 0，而 LRU 不这么做。

当行序为 k 的某行被访问一次，LFU 就会将其 Pk 加 1，而 LRU 则将其置 0，其他各行计数器加 1。很显然，这两种策略沿着相反的两个方向标记着被访行。

当需要替换时，对这些特定行的计数值进行比较，LFU 将计数值最小的行换出，同时将该行的计数器清零，将其标记为新调入行，而 LRU 则将计数值最大的行换出。

LFU 换出计数值小的行，而新行的计数值小，容易被替换出去，它将计数周期限定在对这些特定行两次替换之间的时间间隔内，因而不能严格反映近期访问情况。LRU 换出计数值大的行，而新行的计数值小，因而保护了刚拷贝到 cache 中的新数据行，符合 cache 工作原理，因而使 cache 有较高的命中率。

根据以往的经验，会有相当一部分同学将 cache 的写操作策略与替换策略混淆，自认为没什么区别，请同学们在下一节学习 cache 的写操作策略时，仔细琢磨二者之间的区别和联系，先提示大家，首先要弄清楚为什么需要替换策略和写操作策略，然后其他的细节问题理解起来就容易多了。

（本节讲授完毕）

3.5.4 cache 的写操作策略

（PPT354 第 1 页）上一节学习的替换策略主要用于解决新旧数据替换的问题，以保证 cache 尽量保存最新数据。可是，还有一个主存与 cache 内容一致性的问题需要解决。

因为 cache 的内容只是主存部分内容的拷贝，它应当与主存内容保持一致。CPU 对 cache 的写入更改了 cache 的内容。可选用写操作策略使 cache 内容和主存内容保持一致。如何与主存内容保持一致，可选用写回法、全写法、写一次法三种写操作策略。

（动画 1）写回法是指当 CPU 写 cache 命中时，只修改 cache 的内容，而不立即写入主存；只有当此行被换出时才写回主存。（动画 2）这种方法减少了访问主存的次数，但是存在不一致性的隐患。实现这种方法时，每个 cache 行必须配置一个修改位，以反映此行是否被 CPU 修改过。

（动画 3）全写法是指当写 cache 命中时，cache 与主存同时发生写修改，当写 cache 未命中时，直接向主存进行写入。（动画 4）cache 中每行无需设置一个修改位以及相应的判断逻辑。缺点是降低了 cache 的功效。

（动画 5）写一次法是指写命中与写未命中的处理方法与写回法基本相同，只是第一次写命中时要同时写入主存。（动画 6）这便于维护系统全部 cache 的一致性。

讲完了 cache 的写操作策略，请问同学们，它主要用于解决什么问题？cache 的写操作策略用于维护主存与 cache 内容的一致性，可避免因数据不一致导致的程序运行错误，还可避免因 cache 命中率下降导致存储系统的效率损失。

只有一级 cache 的情况下，一旦未命中，就需要访问主存，访问 cache 的时间相当于浪费，尽管比访问主存的时间要少得多，但仍然造成了缺失损失，如何解决这个问题，我们将在下一节通过一个实例来弄懂其技术途径。

（本节讲授完毕）

3.5.5 使用多级 cache 减少缺失损失

（PPT355 第 1 页）为进一步缩小现代处理器高时钟频率和访问 DRAM 相对较慢之间的差距，高性能微处理器可支持附加一级的 cache。（动画 1）这种二级的 cache，位于处理器芯片内或是位于处理器芯片外单独的一组 SRAM，当访问主 cache 缺失后就会访问它。如果二级 cache 包含所请求的数据，缺失损失就是二级 cache 的访问时间，这要比主存的访问时间少得多。如果第一级 cache、第二级 cache 都不包含这个数据，就需要访问主存储器，产生更大的缺失损失。

（PPT355 第 2 页）我们通过【例 10】的求解分析，来掌握这一技术途径。题目告诉我们：现有一处理器，基本 CPI 为 1.0，所有访问在第一级 cache 中命中，时钟频率 5GHz。假定访问一次主存储器的时间为 100ns，其中包括所有缺失处理。设平均每条指令在第一级 cache 中产生的缺失率为 2%。若增加一个二级 cache，命中或缺失的访问时间都为 5ns，且容量大到可使必须访问主存的缺失率降为 0.5%，问处理器速度提高多少。

解题步骤如下：

（动画 1）必须访问主存储器的缺失损失为：$\dfrac{\textbf{100ns}}{\textbf{0.2ns/时钟周期}} = \textbf{500个时钟周期}$

（动画 2）只有一级 cache 的 CPU：

总的 CPI = 基本 CPI + 每条指令中存储器停顿的时钟周期

= 1.0 + 2% × 500 = 11.0

（动画 3）对二级 cache，主 cache 中发生缺失后可以被第二级 cache 或主存处理，访问第二级 cache 的缺失损失为：$\frac{\mathbf{5ns}}{\mathbf{0.2ns/时钟周期}} = \mathbf{25个时钟周期}$

（动画 4）如果第二级 cache 能处理全部缺失，那么这就是整个的缺失损失。如果缺失要求访问主存储器，那么总的缺失损失为访问第二级 cache 和访问主存储器的时间之和。

（动画 5）有二级 cache 的 CPU:

总的 CPI = 基本 CPI + 每条指令的一级停顿 + 每条指令的二级停顿

= 1.0 + 2% × 25 + 0.5% × 500 = 4.0

后者是前者 CPU 性能的：11.0 ÷ 4.0 = 2.8 倍

cache 是一种高速缓冲存储器，是为了解决 CPU 和主存之间速度不匹配而采用的一项重要的硬件技术，并且发展为多级 cache 体系，指令 cache 与数据 cache 分设体系。要求 cache 的命中率接近于 1。主存与 cache 的地址映射有全相联、直接、组相联三种方式。其中组相联方式是前二者的折衷方案，适度地兼顾了二者的优点又尽量避免其缺点，从灵活性、命中率、硬件投资来说较为理想，因而得到了普遍采用。

多级存储体系结构中常见的是 cache、主存和外存三级体系，同学们可以看到，我们已经学习了主存和 Cache，是不是需要懂得外存是怎么回事？至少需要弄清楚 PC 机中所用的硬盘是怎么工作的，它有什么样的结构等等问题。关于这些内容，我们可以在下一节课一起来学习。

（本节讲授完毕）

3.6 磁盘存储设备

3.6.1 磁记录原理

（PPT361 第 1 页）同学们，我们使用的 PC 机中有个大容量的硬盘，能存储很多的数据、程序和资料。请问你的硬盘容量有多大？500G，1T，8T？容量如此之大的硬盘究竟具有什么样的结构，它是怎样记录信息的？下面学习的内容将为我们逐一解答这些问题。（动画）在多层次的存储器体系结构中，硬盘是一种磁盘存储设备，属于外存层次。

（PPT361 第 2 页）而磁盘属于磁表面存储器，是用某些磁性材料薄薄地涂在金属铝或塑料表面作载磁体来存储信息。（动画 1）它的 4 个优点分别是：存储容量大，位价格低；记录介质可以重复使用；记录信息可以长期保存而不丢失，甚至可以脱机存档；非破坏性读出，读出时不需要再生信息。（动画 2）它的缺点是：存取速度较慢，机械结构复杂，对工作环境要求较高。

1、磁性材料的物理特性

（PPT361 第 3 页）磁性材料作为载磁体记录信息，这一功能与它的物理特性有关。我们知道计算机中用二进制表示信息 0 和 1，无独有偶，自然界就是如此奇妙，磁性材料的磁滞回线表明，磁性材料被磁化以后，会产生稳定的正剩磁状态（+Br）或者负剩磁状态（-Br），而且与所加电流的方向密切相关，这种情况恰好与我们想要的 1 和 0 吻合，因此，可规定+Br 代表 1，-Br 代表 0，需要 1 的时候加正向脉冲电流，需要 0 的时候加负向脉冲电流即可。磁性材料上呈现剩磁状态的地方形成了一个磁化元或存储元，它是记录一个二进制信息位的最小单位。

2、磁表面存储器的读写原理

上述内容说明磁性材料能够表示 0 或 1，仅知道这一点还不够，因为我们都知道存储器通常有读写两种操作，在这个问题上磁表面存储器是如何处理的呢？

（PPT361 第 4 页）大家以前学习物理的时候就明白电-磁变换规则，也就是利用磁头写线圈中的脉冲电流，可把一位二进制代码转换成载磁体存储元的不同剩磁状态。当载磁体相对于磁头运动时，就可以连续写入一连串的二进制信息。

（PPT361 第 5 页）反过来，通过磁-电变换，利用磁头读出线圈，可将由存储元的不同剩磁状态表示的二进制代码转换成电信号输出。当载磁体相对于磁头运动时，就可以连续读出一连串的二进制信息。

（PPT361 第 6 页）形成不同写入电流波形的方式，称为记录方式。记录方式是一种编码方式，它按某种规律将一串二进制数字信息变换成磁层中相应的磁化元状态，用读写控制电路实现这种转换。在磁表面存储器中，由于写入电流的幅度、相位、频率变化不同，从而形成了不同的记录方式。常用记录方式可分为：不归零制(NRZ)、调相制(PM)、调频制(FM)。

（PPT361 第 7 页）如图所示，表示了磁表面存储器记录方式的写读波形图。给写线圈加上一定方向的脉冲电流，在磁体被磁化后，其不同的磁化状态所产生的感应电势方向不同，这样，不同方向的感应电势经读出放大器放大鉴别，就可判知读出的信

息是“1”还是“0”。

由上述内容可知，磁表面存储器的记录原理是：通过电-磁变换，将电脉冲信号转换成剩磁状态，实现写信息；通过磁-电变换，将剩磁状态转换成电脉冲信号，实现读信息。并且按某种规律将一串二进制数字信息变换成磁层中相应的磁化元状态，实现信息编码。掌握了这些原理之后，磁盘的结构和组成便是我们的下一个探究重点。

（本节讲授完毕）

3.6.2 磁盘的组成和分类

（PPT362 第 1 页）请问哪些同学拆开 PC 机硬盘，研究过它的结构？（动画 1、动画 2）专业用户眼里拆开后的硬盘如图所示，（动画 3）而我们需要从计算机设计者的角度认识拆开后的硬盘。

（PPT362 第 2 页）硬盘的全称叫硬磁盘机，它是指记录介质为硬质圆形盘片的磁表面存储器。（动画 1）主要由磁记录介质、磁盘控制器、磁盘驱动器三部分组成。（动画 2）磁盘控制器指的是红色线条圈起来的这一部分，包括控制逻辑与时序、数据并-串变换电路和串-并变换电路。（动画 3）磁盘驱动器指的是蓝色线条圈起来的这一部分，包括写入电路与读出电路、读写转换开关、读写磁头与磁头定位伺服系统等。

写入时，将计算机并行送来的数据取至并-串变换寄存器，变为串行数据，然后一位一位地由写电流驱动器作功率放大并加到写磁头线圈上产生电流，从而在盘片磁层上形成按位的磁化存储元。读出时，当记录介质相对磁头运动时，位磁化存储元形成的空间磁场在读磁头线圈中产生感应电势，该读出信息经放大检测就可还原成原来存入的数据。由于数据是一位一位串行读出的，故要送至串-并变换寄存器变换为并行数据，再并行送至计算机。

（PPT362 第 3 页）硬盘的分类主要取决于盘片结构和磁头类型，其中盘片有**可换盘片式**与**固定盘片式**两种，磁头有**可移动磁头**和**固定磁头**两种。（动画 1）二者进行合理的搭配之后，硬盘就有可移动磁头固定盘片的磁盘机、固定磁头磁盘机、可移动磁头可换盘片的磁盘机三种类型。（动画 2）在可移动磁头固定盘片的磁盘机中，一片或一组盘片固定在主轴上，盘片不可更换 。盘片每面只有一个磁头，存取数据时磁头沿盘面径向移动。（动画 3）在固定磁头磁盘机中，磁头位置固定，磁盘的每一个磁道对应一个磁头，盘片不可更换。优点是存取速度快，省去磁头找道时间，缺点是结构复杂。（动画 4）在可移动磁头可换盘片的磁盘机中，盘片可以更换，磁头可沿盘面径向

移动。优点是盘片可以脱机保存，同种型号的盘片具有互换性。（动画 5）温彻斯特磁盘机是一种可移动磁头、固定盘片的磁盘机，它将磁头、盘片、电机等驱动部件、读写电路等组装成一个不可拆卸的机电一体化整体，防尘性能好，可靠性高，对使用环境要求不高，因而得到了广泛的应用，成为最有代表性的硬磁盘存储器，简称“温盘”。

（PPT362 第 4 页）磁盘的组成和分类就讲到这里，请同学们课后思考以下问题：温盘的发明具有划时代意义，你能说说为什么？

（本节讲授完毕）

3.6.3 磁盘驱动器和控制器

（PPT363 第 1 页）通过前一节的学习，我们简单了解了硬盘的主要组成，请问同学们：硬盘主要由哪三部分组成？（动画 1）我们可以用一个比前面见到的逻辑结构图更简单的示意图来表达这三个主要的组成部分。其中，（动画 2）盘片由硬质铝合金材料制成，（动画 3）其盘面涂上磁性材料，便可以作为磁记录介质。

（PPT363 第 2 页）各类磁盘驱动器的具体结构虽然有差别，但基本结构相同，主要由定位驱动系统、主轴系统和数据转换系统组成。磁盘驱动器是一种精密的电子和机械装置，因此各部件的加工安装有严格的技术要求。对温盘驱动器，还要求在超净环境下组装。在这里提醒大家，我们在搬移主机箱时要轻拿轻放，主要为了避免明显的振动。振动会导致主机箱中的部件接触不良或者受损，其中硬盘对振动更为敏感。

（动画 1）磁盘驱动器的结构如图所示，在可移动磁头的磁盘驱动器中，磁头定位驱动系统由驱动部件、传动部件、运载部件(磁头小车)组成，能够驱动磁头沿盘面径向位置运动以寻找目标磁道位置。当磁盘存取数据时，磁头小车的平移运动驱动磁头进入指定磁道的中心位置，并精确地跟踪该磁道。目前磁头小车的驱动方式主要采用步进电机和音圈电机两种。步进电机靠脉冲信号驱动，控制简单，整个驱动定位系统是开环控制，因此定位精度较低，一般用于道密度不高的硬磁盘驱动器。音圈电机是线性电机，可以直接驱动磁头作直线运动，整个驱动定位系统是一个带有速度和位置反馈的闭环控制系统，驱动速度快，定位精度高，因此用于较先进的磁盘驱动器。

主轴系统的作用是安装盘片，并驱动它们以额定转速稳定旋转。其主要部件是主轴电机和有关控制电路。数据转换系统的作用是控制数据的写入和读出，包括磁头、磁头选择电路、读写电路以及索引、区标电路等。

（PPT363 第 3 页）磁盘控制器是主机与磁盘驱动器之间的接口。（动画 1）由于

磁盘存储器是高速外存设备，故与主机之间采用成批交换数据方式。作为主机与驱动器之间的控制器，它需要有两个方面的接口：（动画 2）一个是与主机的接口，控制外存与主机总线之间交换数据；（动画 3）另一个是与设备的接口，根据主机命令控制设备的操作。（动画 4）前者称为系统级接口，后者称为设备级接口。（动画 5）磁盘控制器的功能是：接受主机发来的命令，转换成磁盘驱动器的控制命令；实现主机和驱动器之间的数据格式转换；控制磁盘驱动器读写。

（PPT363 第 4 页）主机与磁盘驱动器交换数据的控制逻辑如图所示，（动画 1）磁盘上的信息经读磁头读出以后送至读出放大器，（动画 2）然后进行数据与时钟的分离，（动画 3）再进行串-并变换、（动画 4）格式变换，（动画 5）最后送入数据缓冲器，经 DMA（直接存储器传送）控制将数据传送到主机总线。（动画 6）写数据时，经过 DMA 传送、数据缓冲、格式变换、并-串变换、数据编码、写数放大保真这一系列过程，将数据从主机写到磁盘之中。（动画 7）由此可见，磁盘控制器的功能全部转移到设备中，主机与设备之间采用标准的通用接口，从而使设备相对独立。

（PPT363 第 5 页）弄清磁盘驱动器的组成及功能，硬盘控制器的逻辑结构及功能，我们基本上就能明白硬盘是如何工作的。请同学们思考一个问题，那就是磁盘与我们前面学习的半导体存储器的编址方式是否相同？为什么？以便学好下一小节的内容。

（本节讲授完毕）

3.6.4 磁盘上信息的分布

（PPT364 第 1 页）半导体存储器的存储介质是半导体器件，它是一种随机存储器，它的任何存储单元的内容都能被随机存取，且存取时间和存储单元的物理位置无关。而磁盘存储器的存储介质是磁性材料，它是一种半顺序存储器。何谓半顺序存储器？首先我们共同回顾一下第一节学过的一个概念，那就是顺序存储器。所谓顺序存储器是指存储器只能按某种顺序来存取，存取时间和存储单元的物理位置有关，它的存取周期较长。磁盘存储器对数据的访问，在不同柱面和磁头上是随机选择的，但在一个扇区内部是进行顺序读写的，所以磁盘属于一种半顺序存储器。这取决于磁盘上的信息分布方式。

（PPT364 第 1 页）盘片的上下两面都能记录信息，通常把磁盘片表面称为**记录面**。如图所示，记录面上一系列同心圆称为**磁道**。每个盘片表面通常有几十到几百个磁道，每个磁道又分为若干个**扇区**。（动画 1）磁道的编址是**从外向内**依次编号，最外一个同

心圆叫 0 磁道，最里面的一个同心圆叫 n 磁道，n 磁道里面的圆面积并不用来记录信息。扇区的编号有多种方法，可以连续编号，也可间隔编号。（动画 2）磁盘地址：记录面号(磁头号)、磁道号和扇区号。磁道的起始位置称为索引。通过索引标志定出磁道的起始位置，便于读/写操作。多个记录面上相同编号的磁道就会形成一个柱面。

（PPT364 第 2 页）如图所示，磁盘存储器的每个扇区记录定长的数据，因此读/写操作是以扇区为单位一位一位串行进行的。每一个扇区记录一个记录块。每个记录块由头部空白段、序标段、数据段、校验字段及尾部空白段组成。其中空白段用来留出一定的时间作为磁盘控制器的读写准备时间，序标段被用来作为磁盘控制器的同步定时信号，数据段就是记录的有效数据，校验字段用来校验磁盘读出的数据是否正确。

弄清楚磁盘地址格式、数据在磁盘上的记录格式，我们对磁盘存储器是一种半顺序存储器应该有了基本的认识，它的速度比顺序存储器要快一些，但比随机存储器要慢很多，而且我们还知道它的容量很大，所有这些特征都需要相应的技术指标来描述。请同学们对比主存储器的技术指标来学习下一节的内容，重点要搞懂不同的指标。

（本节讲授完毕）

3.6.5 磁盘存储器的技术指标

（PPT365 第 1 页）磁盘存储器的主要技术指标包括存储密度、存储容量、存取时间和数据传输率。（动画 1）存储密度分为：道密度、位密度和面密度。（动画 2）道密度是指沿磁盘半径方向单位长度上的磁道数，(道/英寸)。（动画 3）位密度—磁道单位长度上能记录的二进制代码位数，(位/英寸)。（动画 4）面密度—位密度和道密度的乘积，(位/平方英寸)。（动画 5）存储容量是指一个磁盘存储器所能存储的字节总数。（动画 6）存储容量有格式化容量和非格式化容量之分。（动画 7）格式化容量是指按照某种特定的记录格式所能存储信息的总量。（动画 8）非格式化容量是指磁记录表面可以利用的磁化单元总数。同学们知道，我们买一个 500GB 的硬盘，通过分区以及格式化之后，发现各分区容量的总和不到 500GB，前者是非格式化容量，后者是格式化容量。

（PPT365 第 2 页）（动画 1）存取时间是指从发出读写命令后，磁头从某一起始位置移动至新的记录位置，到开始从盘片表面读出或写入信息加上传送数据所需要的时间。它由找道时间、等待时间、数据传送时间三部分组成。其中，（动画 2）找道时间是指将磁头定位至所要求的磁道上所需的时间。（动画 3）等待时间是指找道完成后至磁道上需要访问的信息到达磁头下的时间。（动画 4）这两个时间都是随机变化的，

因此往往使用平均值来表示，平均找道时间是最大找道时间与最小找道时间的平均值。平均等待时间和磁盘转速有关，它用磁盘旋转一周所需时间的一半来表示。（动画 5）平均存取时间等于平均定位时间、平均等待时间、数据传送时间之和。

（PPT365 第 3 页）数据传输率是指磁盘存储器在单位时间内向主机传送数据的字节数。（动画 1）传输率与存储设备和主机接口逻辑有关。从主机接口逻辑考虑，应有足够快的传送速度向设备接收/发送信息。（动画 2）数据传输率就等于磁盘转速 r 乘以每条磁道的容量 N，或者等于位密度 D 乘以磁盘旋转的线速度 v。数据传输率的单位是字节/秒。在这里强调一下，磁盘转速的单位是转/秒。

（PPT365 第 4 页）下面我们一起来做这道例题，进而帮助同学们消化理解前面所学的一些知识点。题目告诉我们一些已知数据，提出了 5 个问题。

（PPT365 第 5 页）（动画 1）我们首先来解决第一个问题：每台磁盘组共有多少个柱面?一个记录面的有效存储区域等于 0 磁道的半径 16.5 减去 n 磁道的半径 11，等于 5.5(cm)，可见，记录面上的有效存储区域是一个环面。而道密度等于 40 道/cm，所以一个记录面上总共有 40 乘以 5.5，等于 220 道，前面我们讲过：多个记录面上相同编号的磁道就会形成一个柱面，由此可见，磁盘共有 220 个柱面。

（动画 2） 第二个问题：每台磁盘组的总存储容量是多少?题目给出了内层位密度，所以我们从 n 磁道（即 219 磁道）入手首先计算一个磁道上的信息量，它等于磁道的周长乘以位密度，内层磁道的周长算出来等于 69.08(cm)，每道的信息量就等于内层位密度 400 乘以内层磁道的周长 69.08，等于 2 万 7 千 6 百 32 位，也可折算成 3 千 4 百 54 字节。每个记录面的信息量就等于每道的信息量 3 千 4 百 54 字节乘以 220 道，等于 75 万 9 千 8 百 80 字节。一个盘组的总容量就等于每个记录面的信息量 75 万 9 千 8 百 80 字节乘以记录面的数量 10，等于 759 万 8 千 8 百字节。

（动画 3）第三个问题：每台磁盘组的数据传输率是多少？数据传输率等于磁盘转速 r 乘以每条磁道的容量 N，题目告诉我们磁盘转速为 2400 转/分，在这里需要转换成每秒多少转，因此参与计算的转速 r 为每秒 40 转，每条磁道的容量为 3 千 4 百 54 字节，这两个数相乘之后等于每秒 1 万 3 千 8 百 16 字节。

（PPT365 第 6 页）（动画 1）第四个问题：采用定长数据块记录格式，直接寻址的最小单位是什么？寻址命令中如何表示磁盘地址?采用定长数据块格式，直接寻址的最小单位是一个记录块(一个扇区)，每个记录块记录固定字节数目的信息，在定长记录的数据块中，该磁盘组的编址方式如图所示，（动画 2）该计算机有 4 台磁盘组，所

以台号占 2 位，（动画 3）每台磁盘组有 10 个记录面，所以盘面磁头号占 4 位，（动画 4）每个记录面有 220 个磁道，所以柱面磁道号占 8 位，（动画 5）每道有 16 个扇区，所以扇区号占 4 位。

（动画 2）第五个问题：如果某文件长度超过一个磁道的容量，应将它记录在同一个存储面上，还是记录在同一个柱面上?如果某文件长度超过一个磁道的容量，应将它记录在同一个柱面上，因为不需要重新找道，数据读/写速度快。如果记录在同一个记录面的不同磁道上（即不同柱面上），磁头需要沿着径向移动，找寻相应的磁道，这是需要时间开销的，就会导致磁盘的读/写速度下降，是不可取的。

（PPT365 第 7 页）通过这部分的学习，我们知道硬盘主要由磁记录介质、磁盘驱动器和磁盘控制器三部分组成。普遍被使用的温彻斯特盘是将各个机械部件和电子器件组装密封的一种不可随意拆卸的存储装置。（动画 1）硬盘凭借盘片记录面上磁性材料形成的磁化元记录二进制信息，具体来讲，通过电-磁变换写信息，通过磁-电变换读信息。（动画 2）硬盘的主要技术指标有：存储密度、存储容量、存取时间、数据传输率。（动画 3）硬盘的主要特点是存储容量大、存取速度较慢。（动画 4）存储容量大主要是因为记录面多、面密度大。（动画 5）存取速度较慢主要是因为它的平均存取时间等于找道时间加等待时间再加数据传送，很显然前两部分的时间开销影响较大。

（PPT365 第 8 页）讲到这里，多层次存储器的主要内容已经讲授完毕，从前面学过的内容，我们知道 DRAM 存储器由于刷新操作，使得它的速度比 SRAM 的要慢。借着这个例子的启发和你们积极学习的热情，请同学们课后综合运用所学知识，深入思考一个问题，那就是：磁盘的存取速度较慢，这是为什么？另外，提一个建议：请同学们找个合适的机会和闲置的硬盘，拆开它仔细研究研究。

（本节讲授完毕）

第4章　指令系统（3 学时）

4.1 指令系统的性能要求

（PPT41 第 2 页）同学们，我们都知道，利用计算机解决问题需要程序设计，而计算机程序是由一系列的机器指令组成的。（动画）一台计算机中所有机器指令的集合构成了这台计算机的指令系统。它是反映计算机性能的重要因素之一，它是设计一台计算机的硬件与低层软件的接口，它的格式与功能不仅直接影响到机器的硬件结构，而且也影响到系统软件，影响到机器的适用范围。因此，进行指令格式和功能的设计，需要结合整机系统对指令系统性能的要求。

（PPT41 第 3 页）在学习指令系统的性能要求之前，首先要弄清楚**指令**、**微指令**、**宏指令**、**机器指令**、**指令系统**、**系列机**这几个关于指令系统的基本概念。（动画 1）指令是指计算机执行某种操作的命令。从计算机组成的层次结构来说，计算机的指令有微指令、宏指令和机器指令之分。（动画 2）微指令是微程序级的命令，它属于硬件。（动画 3）宏指令是由若干条机器指令组成的软件指令，它属于软件。（动画 4）机器指令介于微指令与宏指令之间，通常简称为指令，每一条指令可完成一个独立的算术运算或逻辑运算操作。（动画 5）指令系统是指一台计算机中所有机器指令的集合。（动画 6）系列机是指基本指令系统相同、基本体系结构相同的一系列计算机。

（PPT41 第 4 页）指令系统的性能如何，决定了计算机的基本功能，因而指令系统的设计是计算机系统设计中的一个核心问题，它不仅与计算机的硬件结构紧密相关，而且直接关系到用户的使用需要。一个完善的指令系统应满足**完备性**、**有效性**、**规整性**、**兼容性**这四个方面的要求。

（动画 1）**完备性**是指用汇编语言编写各种程序时，指令系统直接提供的指令足够使用，而不必用软件来实现。完备性要求指令系统丰富、功能齐全、使用方便。一台计算机中最基本、必不可少的指令是不多的。许多指令可用最基本的指令编程来实现。例如，乘除运算指令、浮点运算指令可直接用硬件来实现，也可用基本指令编写的程序来实现。采用硬件指令的目的是提高程序执行速度，便于用户编写程序。

（动画 2）**有效性**是指利用该指令系统所编写的程序能够高效率地运行。高效率主要表现在程序占据存储空间小、执行速度快。一般来说，一个功能更强、更完善的指令系统，必定有更好的有效性。

（动画 3）规整性包括指令系统的对称性、匀齐性、指令格式和数据格式的一致性。（动画 4）对称性是指：在指令系统中所有的寄存器和存储器单元都可同等对待，所有的指令都可使用各种寻址方式；（动画 5）匀齐性是指：一种操作性质的指令可以支持各种数据类型，如算术运算指令可支持字节、字、双字整数的运算，十进制数运算和单、双精度浮点数运算等；（动画 6）指令格式和数据格式的一致性是指：指令长度和数据长度有一定的关系，以方便处理和存取。例如指令长度和数据长度通常是字节长度的整数倍。

（动画 7）系列机各机种之间具有相同的基本结构和共同的基本指令集，因而指令系统是兼容的，即各机种上基本软件可以通用。但由于不同机种推出的时间不同，在结构和性能上有差异，做到所有软件都完全兼容是不可能的，只能做到“向上兼容”，即低档机上运行的软件可以在高档机上运行。

一台计算机中所有机器指令的集合，称为这台计算机的指令系统。指令系统是表征一台计算机性能的重要因素，它的格式与功能不仅直接影响到机器的硬件结构，而且也影响到系统软件。因此，下一节我们将学习指令格式的分析与设计。

（本节讲授完毕）

4.2 指令格式

4.2.1 指令字长度

（PPT421 第 1 页）通过前一节的学习，我们知道，表示一条指令的机器字，就称为指令字，通常简称指令。（动画 1）指令能反映以下信息：做什么操作；如果需要操作数，从哪里取；结果送到哪里；下一条指令从哪里取。（动画 2）指令格式的分析和设计是指令系统设计的重要内容之一，影响计算机指令格式的因素主要有：机器的字长，存储器的容量，指令的功能。因此，我们首先要熟悉指令格式的定义和内涵。（动画 3）指令格式是指令字用二进制代码表示的结构形式。（动画 4）一条指令的结构可表示为操作码字段和地址码字段。

（PPT421 第 2 页）设计计算机时，对指令系统的每一条指令都要规定一个操作码。（动画 1）指令的操作码 OP 表示该指令应进行什么性质的操作，即表征指令的操作特性与功能，如进行加法、减法、乘法、除法、取数、存数等等。不同的指令用操作码字段的不同编码来表示，每一种编码代表一种指令。组成操作码字段的位数一般取决于计算机指令系统的规模。较大的指令系统就需要更多的位数来表示每条特定的指令。对于一个机器的指令系统，在指令字中操作码字段和地址码字段的长度通常是固定的。

（动画 2）等长（固定长度）编码的优点是：指令规整，译码简单；主要缺点是：信息的冗余极大，使程序的总长度增加。

（动画 3）地址码字段指示操作数的地址。根据一条指令中有几个操作数地址，可将该指令称为几操作数指令或几地址指令。（动画 4）一般的操作数有被操作数、操作数及操作结果这三种数，因而就形成了三地址指令格式、二地址指令格式、一地址指令格式和零地址指令格式。

（PPT421 第 2 页）（动画 1）三地址指令字中 OP A1　A2　A3 的含义是：地址 A1 的源操作数与地址 A2 的操作数进行 OP 操作，将其操作结果存放在地址 A3 指向的存储单元，与此同时程序计数器 PC 的值增 1，确定了下一条指令的地址。三地址指令中，通常将 A1、A2、A3 指定为运算器中通用寄存器的地址，能够加快指令的执行速度。三地址指令只在字长较长的大、中型计算机中使用，而小型、微型机中很少使用。（动画 2）二地址指令字中 OP A1　A2 的含义是：地址 A1 的源操作数与地址 A2 的操作数进行 OP 操作，将其操作结果存放在地址 A1 指向的存储单元，与此同时程序计数器 PC 的值增 1，确定了下一条指令的地址。在这类指令中，A1 兼做存放操作结果的地址，指令执行之后，A1 中原来的内容已经被新的运算结果替换了。

（动画 3）二地址指令根据操作数的物理位置分为：存储器－存储器（SS）型指令，寄存器－存储器（RS）型指令，寄存器－寄存器（RR）型指令。（动画 4）需要注意的是：SS 型指令速度最慢，RR 型指令速度最快，RS 型指令速度居中。这对于我们设计指令时将操作数的位置指定在内存还是寄存器有着至关重要的影响。

（动画 5）一地址指令只有一个地址码，它指定一个操作数，另一个操作数地址是隐含的，一般情况下，以运算器中累加寄存器 AC 中的数据作为隐含的被操作数。所以，OP A 的含义是：地址 A 的源操作数与累加寄存器 AC 中的数据进行 OP 操作，将其操作结果存放在 AC 中，与此同时程序计数器 PC 的值增 1，确定了下一条指令的地址。（动画 6）零地址指令的指令字中只有操作码，而没有地址码。例如，“停机”、“空操作”、“清除”等控制类指令。

（PPT421 第 4 页）指令字长度指的是一个指令字中包含二进制代码的位数。由前面所学的内容可知，不同种类的指令其长度不尽相同。指令字长度与机器字长存在着一定的比例关系。机器字长指的是计算机能直接处理的二进制数据的位数，它决定了计算机的运算精度。（动画 1）指令字长度等于机器字长度的指令，称为单字长指令；（动画 2）指令字长度等于半个机器字长度的指令，称为半字长指令；（动画 3）指令字长度等于两个机器字长度的指令，称为双字长指令。譬如，机器字长是 64 位，若指

令字长是 64 位，则该指令是单字长指令；若指令字长是 32 位，则该指令是半字长指令；若指令字长是 128 位，则该指令是双字长指令。（动画 4）多字长指令的优点是：（动画 5）提供足够的地址位来解决访问内存任何单元的寻址问题；（动画 6）缺点是：必须两次或多次访问内存以取出一整条指令，降低了 CPU 的运算速度，又占用了更多的存储空间。早期的计算机多采用这种多字长指令结构。

（PPT421 第 5 页）（动画 0）根据指令字长度是否相等或者固定，将其分为等长指令字结构和变长指令字结构。（动画 1）在一个指令系统中，如果各种指令字长度是相等的，称为等长指令字结构，这种指令字的优点是结构简单，且指令字长度是不变的，（动画 2）缺点是信息冗余大，占用存储空间多。（动画 3）如果各种指令字长度随指令功能而异，就称为变长指令字结构。（动画 4）这种指令字的优点是结构灵活，能充分利用指令长度，（动画 5）缺点是指令译码存在难度，指令的控制较复杂。（动画 6）高档微型机中多采用大于 32 位的固定长度。

指令在计算机中是用二进制信息表示的，程序员在写程序的时候，如果要大量记忆由 0 和 1 组成的指令，那是相当困难而且是不现实的，怎么办呢？下一节将要学习的指令助记符将会帮助我们更好地编写计算机程序。（本节讲授完毕）

4.2.2 指令助记符

（PPT422 第 1 页）计算机能够直接识别的是二进制信息 0 或 1，但程序员编写程序记忆二进制表示的指令就非常困难，所以，为了便于书写和记忆而设定的、与机器指令一一对应的英文缩写码，我们将其称为**指令助记符**。（动画 1）其命名规则是：用 3～4 个英文字母来表示操作码，一般为英文缩写。（动画 2）我们需要注意问题有两个，第一个是：不同的计算机系统，规定不一样。（动画 3）第二个是：必须用汇编语言翻译成二进制代码。（动画 4）该表中列出了几条典型的指令助记符，譬如，存数指令的助记符为英文缩写 STO，其二进制操作码为 110，我们都学过英语，记忆、阅读和理解 STO 要比 110 容易很多，也许有些同学觉得，110 三位二进制信息有什么难记的？同学们可以考虑一下，如果一条指令字长度是 16 位，除了操作码，还有地址码等信息，都是用 0 或 1 表示的，请问这样的用二进制 0 或 1 表示的指令，你能记多少呢？

（PPT422 第 2 页）该例子给出了一段由 6 条指令组成的程序。我们一眼就能看出来第一、第四、第六条指令是加法指令，第二条是减法指令，第三条是存数指令，第五条是乘法指令。如果换成二进制操作码，估计很多人会混淆不清的。下面我们简单分析其中的几条指令的功能。第一条指令是将通用寄存器 R2 和 R3 的数据相加，然后将其结果存入通用寄存器 R1。第二条指令是将通用寄存器 R3 和 R5 的数据相减，然

后将其结果存入通用寄存器 R4。第三条指令是将通用寄存器 R4 的数据存入到 M(x)指向的存储器单元。第五条指令是将通用寄存器 R1 和 R2 的数据相乘，然后将其结果存入通用寄存器 R3。同学们要注意每条指令后边的注释，其中的通用寄存器符号加上圆括弧是有含义的，意思就是取出该寄存器中存放的值。箭头线后边标明了操作结果写入的目标地址，所以相应的寄存器符号不带圆括弧。

再次强调一下，在不同的计算机中，指令助记符的规定是不一样的。计算机硬件只能识别二进制语言。因此，指令助记符还必须转换成与它们相对应的二进制操作码。这种转换借助汇编器可以自动完成，汇编器的作用相当于一个“翻译”。要设计指令格式，首先要学会分析已有的指令格式，关于这个问题我们将在下一节学习。

（本节讲授完毕）

4.2.3 指令格式分析

（PPT423 第 1 页）指令格式如图所示，OP 为操作码字段，试分析指令格式的特点。（动画 1）操作码字段为（15-8）等于 7 位，可指定 2 的 7 次方等于 128 种操作，也就是可以表示 128 条指令。（动画 2）由该示意图可知一条指令字 16 位，可用一个机器字 16 位表示，指令中的操作数地址有源寄存器和目标寄存器两个，因此是单字长二地址指令。（动画 3）两个操作数均在寄存器中，源寄存器和目标寄存器都是通用寄存器，所以是 RR 型指令。表示目标寄存器和源寄存器的位数分别是 0～3、4～7，都是 4 位，所以，可指定 2 的 4 次方等于 16 个通用寄存器分别作为源和目标寄存器。

（PPT423 第 2 页）指令格式如图所示，OP 为操作码字段，试分析指令格式的特点。（动画 1）操作码字段为（15-9）等于 6 位，可指定 2 的 6 次方等于 64 种操作，也就是可以表示 64 条指令。（动画 2）由该示意图可知一条指令字 32 位，可用两个机器字 16 位表示，指令中的操作数地址有两个，因此是双字长二地址指令。（动画 3）一个操作数在源寄存器，另一个操作数在存储器中，由变址寄存器内容加上偏移量决定该操作数在存储器中的位置，所以是 RS 型指令。表示变址寄存器和源寄存器的位数分别是 0～3、4～7，都是 4 位，所以，可指定 2 的 4 次方等于 16 个寄存器分别作为源和变址寄存器。

（PPT423 第 3 页）指令格式如图所示，OP 为操作码字段，试分析指令格式的特点。（动画 1）操作码字段为（31-25）等于 6 位，可指定 2 的 6 次方等于 64 种操作，也就是可以表示 64 条指令。（动画 2）由该示意图可知一条指令字 32 位，可用一个机器字 32 位表示，指令中的操作数地址有两个，因此是单字长二地址指令。（动画 3）一个操作数在源寄存器，另一个操作数在存储器中，由变址寄存器内容加上偏移量决

定该操作数在存储器中的位置，所以是 RS 型指令。表示变址寄存器的位数是 16～17，共 2 位，所以，可指定 2 的 2 次方等于 4 个寄存器作为变址寄存器；表示源寄存器的位数是 18～22，共 5 位，所以，可指定 2 的 5 次方等于 32 个寄存器作为源寄存器。

指令格式分析主要从三个方面入手，首先弄清楚操作码的位数，从而确定它能表示多少个操作或者多少条指令；其次弄清楚指令字长与机器字长的倍数，以及操作数地址的数量；然后弄清楚操作数从何而来。

指令格式是指令字用二进制代码表示的结构形式，通常由操作码字段和地址码字段组成。操作码字段表征指令的操作特性与功能，而地址码字段指示操作数的地址。目前多采用二地址、单地址、零地址混合方式的指令格式。指令字长度分为：单字长、半字长、双字长三种形式。高档微机采用 32 位长度的单字长形式。

计算机运行程序需要取指令和取操作数，它们究竟在什么地方存放，怎样才能获得它们的有效地址，我们在下一节的指令寻址和数据寻址学习中可获得解决办法。

（本节讲授完毕）

4.3 指令和数据的寻址方式

4.3.1 指令的寻址方式

（PPT431 第 1 页）我们在学完第一章之后，就知道了一种现象，那就是指令和数据均存放在内存中，计算机采用指令周期法（也称为时空法）区分指令和数据。（动画 1）在指令执行阶段，需要确定本条指令中各操作数的地址，同时要确定下一条指令的地址。（动画 2）操作数或指令字在存储器中读/写的方式有三种：地址指定方式、相联存储方式和堆栈存取方式。（动画 3）当采用地址指定方式时，形成操作数或指令地址的方式，称为寻址方式。

（PPT431 第 2 页）指令系统采用不同寻址方式的目的是：缩短指令长度，扩大寻址空间，提高编程灵活性。（动画 1）指令寻址方式比较简单，数据寻址方式比较复杂。（动画 2）形成指令地址的方式，称为指令寻址方式。（动画 3）指令的寻址方式有两种，一种是顺序寻址方式，另一种是跳跃寻址方式。

（PPT431 第 3 页）由于指令地址在内存中按顺序安排，当执行一段程序时，通常是一条指令接一条指令的顺序进行。如图所示，（动画 1）当程序计数器 PC 的值为 0 时，从地址为 0 的内存单元中取出取数 LDA 指令，与此同时程序计数器 PC 的值自增 1，确定了下一条指令的地址为 1。（动画 2）当程序计数器 PC 的值为 1 时，从地址为 1 的内存单元中取出加法 ADD 指令，与此同时程序计数器 PC 的值自增 1，确定了

下一条指令的地址为 2。（动画 3）当程序计数器 PC 的值为 2 时，从地址为 2 的内存单元中取出自增一 INC 指令，与此同时程序计数器 PC 的值自增 1，确定了下一条指令的地址为 3。依此类推。指令的顺序寻址方式简单直观。然而，解决现实中的问题，无法完全依赖简单的顺序结构，还是会出现程序执行顺序转移的情况，这时就需要另一种寻址方式，即指令的跳跃寻址方式。

（PPT431 第 4 页）所谓跳跃，是指下条指令的地址码不是由程序计数器给出，而是由本条指令给出。（动画 1）当程序计数器 PC 的值为 3 时，从地址为 3 的内存单元中取出跳转 JMP 指令，与此同时程序计数器 PC 的值自增 1，更新为 4，（动画 2）随即被本条跳转指令中的 6 更新，所以下一步取出的指令不是地址为 4 的取数 LDA 指令，而是地址为 6 的自增一 INC 指令。采用指令的跳跃寻址方式，可以实现程序转移或构成循环程序，从而能缩短程序长度，或将某些程序作为公共程序引用。指令系统中的各种条件转移或无条件转移指令，就是为了实现指令的跳跃寻址而设置的。

形成指令地址的方式，称为指令寻址方式。有顺序寻址和跳跃寻址两种，由指令计数器来跟踪。指令执行阶段需要的操作数从哪儿来，结果存到哪儿去，需要确定操作数的有效地址，下一节我们将一起学习操作数的寻址方式。

（本节讲授完毕）

4.3.2 操作数基本寻址方式

（PPT432 第 1 页）形成操作数有效地址的方式，称为数据寻址方式。由于指令中操作数的地址码字段是由形式地址和寻址方式特征位等组合形成，因此，一般来说，指令中所给出的地址码，并不是操作数的有效地址。（动画 1）寻址过程就是把操作数的**形式地址**，变换为操作数的**有效地址**的过程。操作数可放在专用寄存器、通用寄存器、内存、指令和 I/O 设备的端口中。（动画 2）因此，就形成了隐含寻址、立即寻址、直接寻址、间接寻址、寄存器寻址、寄存器间接寻址、偏移寻址、段寻址、堆栈寻址等九种基本寻址方式。

（PPT432 第 2 页）（动画 1）**隐含寻址**指的是：在指令中不是明显的给出而是隐含着操作数的地址。单地址指令中为了完成两个数的算术运算，除地址码指明的一个操作数外，另一个常需采用隐含寻址方式。例如，单地址的指令格式，没有在地址字段中指明第二操作数地址，而是规定累加寄存器 AC 作为第二操作数地址，AC 对单地址指令格式来说是隐含地址。（动画 2）**立即寻址**指的是：指令的地址码字段指出的不

是操作数的地址，而是操作数本身。也就是说数据就包含在指令中，只要取出指令，就取出了可以立即使用的操作数，因此，这样的操作数被称为立即数。可以描述为操作数 D 等于指令的地址码字段所表示的数据 A。这种方式的特点是指令执行时间很短，不需要访问内存取数。在取指令时，操作码和操作数被同时取出，不必再次访问存储器，从而提高了指令的执行速度。但是，因为操作数是指令的一部分，不能被修改；而且对于定长指令格式，操作数的大小将受到指令长度的限制，所以这种寻址方式灵活性最差，通常用于给某一寄存器或主存单元赋初值，或者用于提供一个常数。

（PPT432 第 3 页）（动画 1）直接寻址指的是：在指令格式的地址字段中直接指出操作数在内存的地址。即操作数的有效地址 EA 等于形式地址 A，操作数 D 等于地址 A 所指向内存单元中的数据。如图所示，（动画 1）指令译码后将获得操作数的地址 A，通过直接地址 A 访问对应的内存单元，（动画 2）然后获取指令执行所需要的操作数 D。（动画 3）在这种寻址方式中，操作数地址是不能修改的，与程序本身所在的位置无关，所以又叫做绝对寻址方式。（动画 4）指令中地址码的位数是固定的，这将不能满足整个主存空间寻址的要求，因此直接寻址方式受到了很大的限制。另外，在指令的执行过程中，为了取得操作数，必须进行访存操作，降低了指令的执行速度。

（PPT432 第 4 页）间接寻址指的是：指令的地址码部分给出的形式地址 A 不是操作数的有效地址，而是存放操作数有效地址的主存单元的地址，简称操作数地址的地址。即操作数的有效地址 EA 等于形式地址 A 所指内存单元的地址数据，操作数 D 等于以形式地址 A 为指向两次访问内存所得的数据。如图所示，（动画 1）指令译码后将获得操作数的形式地址 A，通过 A 访问对应的内存单元，获得操作数的有效地址 EA，（动画 2）然后通过 EA 访问对应的单元即可获取指令执行所需要的操作数 D。（动画 3）因为操作数的有效地址在主存储器中，可以被灵活的修改而不必修改指令，从而使间接寻址要比直接寻址灵活得多。但是，间接寻址在指令执行过程中至少需要两次访问主存储器才能取出操作数，严重降低了指令执行的速度。

（PPT432 第 5 页）寄存器寻址指的是：在指令的地址码部分给出 CPU 内某一通用寄存器的编号，指令的操作数存放在相应的寄存器中。即操作数的有效地址 EA 等于通用寄存器的编号 R。（动画 1）由于寄存器在 CPU 的内部，指令在执行时从寄存器中取操作数比访问主存要快得多；由于寄存器的数量较少，因此寄存器编号所占位数也较少，从而可以有效减少指令的地址码字段的长度。

（PPT432 第 5 页）（动画 2）寄存器间接寻址指的是：为了克服间接寻址中多次

访存的缺点，可采用寄存器间接寻址，即将操作数放在主存储器中，而操作数的地址放在某一通用寄存器中，然后在指令的地址码部分给出该通用寄存器的编号，（动画 3）这时有操作数的有效地址 EA 等于某一通用寄存器 R 的内容，操作数 D 等于访问一次寄存器 R 获得有效地址后再访问一次内存所得到的数据。指令格式中的寄存器内容不是操作数，而是操作数的地址，该地址指明的操作数在内存中。（动画 4）这种寻址方式的指令较短，并且在取指令后只需一次访存便可得到操作数，因此指令执行速度较前述的间接寻址方式要快，也是目前在计算机中使用较为广泛的一种寻址方式。

（PPT432 第 6 页）前面学习的直接寻址方式能够寻址的主存空间受限、灵活性差，间接寻址方式需要访存两次、速度较慢，为了解决这些问题，研究形成了偏移寻址方式。（动画 1）**偏移寻址**指的是：指令中的形式地址 A 和寄存器 R 所指向的地址数据，通过地址运算器相加之后形成操作数的有效地址 EA，（动画 2）其中 A 是显式的，又称为偏移量；R 或基于地址码显式给出，或基于操作码隐含引用。（动画 3）偏移寻址实质上是直接寻址和寄存器间接寻址的有效结合。（动画 4）R 可以是程序计数器 PC，可以是基址寄存器 Rb，也可以是变址寄存器 Rx，（动画 5）因此形成了相对寻址、基址寻址和变址寻址三种具体的偏移寻址方式。

（PPT432 第 7 页）**相对寻址**方式指的是：（动画 1）把指令格式中的形式地址 A 加上程序计数器 PC 的内容，形成操作数的有效地址 EA，（动画 2）再通过 EA 访问内存中对应的单元，获得相应的操作数。程序计数器的内容就是当前指令的地址。“相对”寻址，就是相对于当前的指令地址而言。（动画 3）相对寻址的好处是程序员无须用指令的绝对地址编程，所编程序可以放在内存任何地方。此时形式地址 A 通常称为偏移量，其值可正可负，相对于利用当前指令地址进行操作数地址的浮动。

（PPT432 第 8 页）**基址寻址**方式指的是：（动画 1）将指令格式中的形式地址 A 加上 CPU 中基址寄存器 Rb 的内容，形成操作数的有效地址 EA，（动画 2）再通过 EA 访问内存中对应的单元，获得相应的操作数。优点是可以扩大寻址能力。同形式地址相比，基址寄存器的位数可以设置得很长，从而可以在较大的存储空间中寻址。

（PPT432 第 9 页）**变址寻址**方式指的是：把指令格式中的偏移量 A 加上 CPU 中某个变址寄存器 Rx 的内容，形成操作数的有效地址 EA。（动画 1）但使用变址寻址方式的目的不在于扩大寻址空间，而在于实现程序块的规律性变化。（动画 2）用哪一个寄存器作为变址寄存器必须在硬件设计时就事先规定，变址寄存器 Rx 中的内容称为**变址值**，该值可正可负。（动画 3）变址寻址方式是一种被广泛采用的寻址方式，最典

型的应用就是将指令的地址码部分给出的地址 A 作为基准地址，而将变址寄存器 Rx 中的内容作为修改量。在遇到需要频繁修改操作数地址时，无须修改指令，只要修改 Rx 中的变址值就可以了，这对于数组运算、字符串操作等一些进行成批数据处理的指令是很有用的。

（PPT432 第 10 页）**段寻址**：如图所示，（动画 1）当指令格式中的偏移量 A 是 16 位、寄存器也是 16 位的时候，直接寻址 1M 存储空间，很显然是有问题的，因为 1M 存储空间需要 20 位地址，而 16 位地址只能直接寻址 64K 存储空间。（动画 2）为此将整个 1M 空间存储器以 64K 为单位划分成若干段，配上 16 位的段寄存器。（动画 3）在寻址一个内存具体单元时，段寄存器中的 16 位地址数据自动左移 4 位之后，（动画 4）再加上一个 16 位基地址（逻辑地址），（动画 5）形成实际的 20 位物理地址，从而具有 1M 存储空间的直接寻址能力。这种寻址方式的实质还是基址寻址。

（PPT432 第 11 页）（动画 1）**堆栈**是一组能存储和取出数据的**暂时存储单元**。很多计算机把存储器的一部分用作堆栈。（动画 2）堆栈和其他形式的存储器之间的差别就在于，它们对数据的存取方法或寻址方法有所不同。（动画 3）堆栈是一种特殊的数据寻址方式，采用**“先进后出”**原理。（动画 4）按结构不同，分为：寄存器堆栈、存储器堆栈。（动画 5）寄存器堆栈是由 CPU 当中的一组专门的寄存器构成。（动画 6）在寄存器堆栈中，所有的数据传送是在栈顶和某个通用寄存器之间进行的，通用寄存器仅和堆栈的顶部单元相联系。它的优点是速度快。（动画 7）通常，CPU 通过“进栈”指令 PUSH 把数据送入堆栈，（动画 8）而通过“出栈”指令 POP 把数据从堆栈中取出。（动画 9）这种结构通常称作“下压堆栈”。

（PPT432 第 12 页）在寄存器堆栈中，每当一个新的数据被压进栈顶时，数据串联的向下通过栈顶下压。如图所示，（动画 1）数据 A 首先进栈，（动画 2）当 B 进栈时，A 必须下移，（动画 3）当 C 进栈时，A、B 必须下移，（动画 4）当 D 进栈时，A、B、C 必须下移。出栈时，最后进栈的数据首先从堆栈中取出。需要数据 A 出栈的话，（动画 5）首先 D 出栈，（动画 6）紧接着 C 出栈，（动画 7）然后 B 出栈，A 移到了栈顶，（动画 8）此时恰好有一个数据 E 要进栈，A 又被下移，（动画 9）若想 A 出栈，刚进栈的数据 E 先出栈，（动画 10）最后 A 才能出栈。（动画 11）因此这种结构通常又称为“后进先出”堆栈，其读出具有破坏性。（动画 12）正因如此，不能将太多的数据压入堆栈，否则，进栈容易出栈难，将会大幅度降低寻址取数的效率，也就是说容量不宜太大。在生活中，我们用一个封底纸箱装书和取书，箱子越大，装的书就会

越多，但取出一本压箱底的好书就没装进去那么容易。同学们，是不是很相像？（动画 13）为了突破寄存器堆栈的这种局限性，计算机中使用了存储器堆栈中。

（PPT432 第 13 页）为了突破寄存器堆栈的这种限制，计算机中使用了存储器堆栈中。（动画 1）这时，需要在 CPU 中配置一个堆栈指示器 SP，其内容指向的存储单元就是栈顶，如图动画所示，（动画 2）当 SP 的内容从 302 变化到 274，再变化到 300，最后变化到 277，栈顶也跟随着这个内容实时变化。（动画 3）进栈时，SP 为 300，栈顶移至存储单元 300，通用寄存器中的数据 a 被存放到存储器 300 指向的单元，随着 SP 的变化，栈顶在变，一直到通用寄存器中的数据 c 被存放到存储器 276 指向的单元，栈顶指向了存储单元 275。（动画 4）出栈时，SP 为 276，栈顶指向存储单元 276，栈顶的数据 c 被取到通用寄存器，以此类推，直到栈顶的数据 a 被取到通用寄存器，栈顶指向了存储单元 301。（动画 5）可见，进栈时先存入数据，后修改堆栈指示器；出栈时，先修改堆栈指示器，然后取出数据。这个现象可以用两个表达式来表示，其中（A）表示通用寄存器 A 的内容，SP 表示堆栈指示器，Msp 表示堆栈指示的存储器栈顶单元。（动画 6）寄存器堆栈和存储器堆栈的操作方式不同，在寄存器堆栈中，移动的是数据，而在存储器堆栈中，移动的是栈顶。这主要和两种堆栈的速度有关。

（PPT432 第 14 页）我们总共学习了 9 种寻址方式，现在对它们的优缺点进行一个简单的对比小结，如表所示，隐含寻址、立即寻址、寄存器寻址、堆栈寻址的优点是无存储器访问，偏移寻址和段寻址具有很好的灵活性，间接寻址和寄存器间接寻址具有大的地址范围，直接寻址是一种最简单的寻址方式。隐含寻址和立即寻址的缺点是数据范围或幅值有限，直接寻址和寄存器寻址的地址范围有限，间接寻址和寄存器间接寻址因多重或额外访问存储器，使得速度较慢，偏移寻址和段寻址方式复杂，堆栈寻址的应用有限。掌握了各种寻址方式的优缺点，我们在设计指令格式时，就懂得如何选择其寻址模式。

（PPT432 第 15 页）下面我们共同做两道指令格式设计的例题，帮助我们消化和巩固前面所学的主要内容。第一道例题给出了一种二地址 RS 型指令的结构示意图和 6 种寻址方式算法及其说明的表格，各种寻址方式取决于间接寻址标志位 I、寻址模式字段 X、偏移量字段 A 的组合，要求我们写出 6 种寻址方式的名称。我们首先分析有效地址 EA 的算法，第一种方式的有效地址 EA 等于偏移量 A，说明这是一种直接寻址方式；第二种方式的有效地址 EA 等于程序计数器的内容加减偏移量 A，说明这是一种相对寻址方式；第三种方式的有效地址 EA 等于变址寄存器 R2 的内容加减偏移量 A，

说明这是一种变址寻址方式；第四种方式的有效地址 EA 等于寄存器 R3 的内容，说明这是一种寄存器间接寻址方式；第五种方式的有效地址 EA 等于偏移量 A 所指存储单元的内容，说明这是一种间接寻址方式；第六种方式的有效地址 EA 等于基址寄存器 R1 的内容加减偏移量 A，说明这是一种基址寻址方式。回过头来我们再分析一下间接寻址标志位 I，从表中可知，只有寄存器间接寻址和间接寻址的 I 为 1，其余均为 0，说明 I 为 1 时，对应的寻址方式便是间接寻址，为 0 时，对应的寻址方式不是间接寻址。最后再分析一下寻址模式字段 X，该字段有 2 位，有四种组合，当 I 为 0 时，这四种组合分别区分了第 1、2、3、6 这四种寻址方式，当 I 为 1 时，只使用了 00、11 这两种组合分别表达了第 5、第 4 种寻址方式。

（PPT432 第 16 页）第二道例题给出了机器字长、主存容量、通用寄存器数量、指令的条数，以及明确的六种寻址方式，要求我们设计单字长双地址指令格式。题目已知条件告诉我们，有 32 条指令，所以操作码 OP 字段占 5 位；通用寄存器有 16 个，所以源寄存器和目的寄存器各占 4 位；有 6 种寻址方式，所以寻址模式字段 X 占 3 位；剩余字段 A 占 32 减去 5 减去 4 减去 4 减去 3 等于 16 位，可作为立即数和直接寻址使用。指令格式如图所示。除此之外，还有一件事情必须要做，那就是寻址模式字段 X 的定义，千万不能忘记。定义 X 等于 000 时表示立即寻址方式，操作数 D 等于 A；定义 X 等于 001 时表示直接寻址方式，有效地址 EA 等于 A；定义 X 等于 010 时表示寄存器直接寻址方式，有效地址 EA 等于寄存器 R 的编号；定义 X 等于 011 时表示寄存器间接寻址方式，有效地址 EA 等于寄存器 R 的内容；定义 X 等于 100 时表示变址寻址方式，有效地址 EA 等于寄存器 R 的内容加上 A；定义 X 等于 101 时表示相对寻址方式，有效地址 EA 等于程序计数器 PC 的内容加上 A。

可见，寻址模式的定义是指令格式设计的主要内容之一，我们在课后有必要通过习题练习，更加深入的理解和掌握数据寻址方式的内涵。不同机种的指令系统不尽相同，但一些基本指令还是相同的，所以我们有必要全面了解基本指令系统的范围、功能和特性，这些内容我们在下一节课共同学习。

（本节讲授完毕）

4.4 典型指令

4.4.1 基本指令系统

（PPT441 第 1 页）同学们，我们花了较多时间学习了指令和数据的寻址方式，现在，我们一起回顾一下这部分的主要内容。形成指令地址的方式，称为指令寻址方式。（动画 1）有顺序寻址和跳跃寻址两种，由指令计数器来跟踪。（动画 2）形成操作数地址的方式，称为数据寻址方式。（动画 3）操作数可放在专用寄存器、通用寄存器、内存和指令中。因此，数据寻址有隐含寻址、立即寻址、直接寻址、间接寻址、寄存器寻址、寄存器间接寻址、相对寻址、基址寻址、变址寻址、段寻址、堆栈寻址等多种方式。（动画 4）按操作数的物理位置不同，常用的二地址指令有 RR 型和 RS 型。前者比后者执行的速度快。（动画 5）堆栈是一种特殊的数据寻址方式，采用“先进后出”原理，按结构不同，分为寄存器堆栈和存储器堆栈。

（PPT441 第 2 页）不同机器的指令系统是各不相同的。从指令的操作码功能来考虑，（动画 1）一个较完善的指令系统，应当有数据处理、数据存储、数据传送、程序控制四大类指令，（动画 2）具体有数据传送类指令、算术运算类指令、逻辑运算类指令、程序控制类指令、输入输出类指令、字符串类指令、特权指令和其他指令共 8 种类型。各类指令都有相应的范围、各自的功能和特性。

（PPT441 第 3 页）它们的范围、功能和特性可以概括起来，如表所示。**数据传送指令**主要包括取数指令、存数指令、传送指令等，用来实现主存和寄存器之间，或寄存器和寄存器之间的数据传送。**算术运算指令**主要包括二进制定点（或浮点）加、减、乘、除等运算类指令，以及十进制加、减运算指令，用于定点或浮点的算术运算，大型机中有向量运算指令，直接对整个向量或矩阵进行求和、求积运算。**逻辑运算指令**主要用于无符号数的位操作、代码的转换、判断及运算。 移位指令用来对寄存器的内容实现左移、右移或循环移位。**程序控制指令**主要包括条件转移指令、无条件转移指令等，条件转移指令根据不同结果进行转移，从而改变程序原来执行的顺序。转移指令的转移地址一般采用直接寻址和相对寻址方式来确定。

（PPT441 第 4 页）**输入输出指令**用来启动外围设备，检查测试外围设备的工作状态，并实现外部设备和 CPU 之间，或外围设备与外围设备之间的信息传送。**字符串处理指令**主要包括字符串传送、转换、替换指令，是一种非数值处理指令，在文字编辑中对大量字符串进行处理。**特权指令**只用于操作系统或其他系统软件，一般不直接提供给用户使用。 在多用户、多任务的计算机系统中特权指令必不可少。它主要用于系

统资源的分配和管理。**其他指令**主要包括状态寄存器置位、复位指令、测试指令、暂停指令，空操作指令，以及其他一些系统控制用的特殊指令。

（PPT441 第 5 页）复杂指令系统计算机的指令系统也遵循 2：8 规则，也就是说大约有 20%的指令使用频率高，占据了 80%的处理机时间，而且是一些最简单最基本的指令，另外 80%的不常用指令只占用处理机的 20%时间。主要原因是：大规模集成电路工艺要求规整性，而大量复杂指令控制逻辑极其不规整，给大规模集成电路工艺造成了很大的困难；现在用微程序实现复杂指令与用简单指令组成的子程序相比，没有多大的区别。因为现在控制存储器和主存的速度差别缩小；复杂指令系统计算机中，通过增强指令系统的功能，简化了软件，增加了硬件的复杂程度。然而指令复杂了，指令的执行时间必然加长，从而使整个系统的执行时间反而增加，因而在计算机体系结构设计中，软硬件的功能分配必须恰当。关于软硬件功能分配的问题在另外一门课程《计算机系统结构》中交待的很详尽，部分专业的学生在选修该门课程。

简化计算机的指令系统的要求因上述现象和原因，变得非常迫切，于是精简指令系统计算机 RISC 诞生了，关于精简指令系统具有哪些特点，我们在下一节学习。

（本节讲授完毕）

4.4.2 精简指令系统

（PPT442 第 1 页）复杂指令系统计算机的指令系统包括的指令多达几百条，硬件结构越来越复杂，不但使计算机的研制周期变长，而且由于采用了大量使用频率很低的复杂指令而造成硬件资源浪费，出现了典型的 2：8 规律，（动画 1）为了解决这一问题，人们经过反复研制，诞生了便于大规模集成电路技术实现的精简指令系统计算机 RISC。（动画 2）精简指令系统的特点概括起来有 5 个，（动画 3）第一个是指令条数少；（动画 4）第二个是指令长度固定；（动画 5）第三个是指令格式种类少；（动画 6）第四个是寻址方式种类少；（动画 7）第五个是只有取数／存数指令访问存储器，其余的指令操作都在寄存器之间进行。

（PPT442 第 2 页）现在我们从指令系统的规模、指令数目、指令格式的种类、寻址方式的种类、软件系统开发时间等十二个方面对复杂指令系统和精简指令系统进行比较，比较结果如表所示。在下面的讲解中，我们称复杂指令系统为 C 系，精简指令系统为 R 系。对整个指令系统而言，C 系复杂、庞大复杂，R 系简单、精简。在指令数目方面，C 系一般大于 200 条，R 系一般小于 100 条。在指令格式种类和寻址方式种类方面，C 系一般大于 4，R 系一般小于 4。在指令字长方面，C 系的不固定，R 系的固定。在可访存指令方面，C 系不加限定，R 系只有 LOAD/STORE 指令。在各种

指令使用频率方面，C 系相差很大，R 系相差不大。在各种指令执行时间方面，C 系相差很大，R 系的绝大多数指令执行在一个周期内完成。在优化编译实现方面，C 系很难，R 系较容易。在程序源代码长度方面，C 系的较短，R 系的较长。在控制器实现方式方面，绝大多数 C 系采用微程序控制方式，绝大部分的 R 系采用硬布线控制方式。在软件系统开发时间方面，C 系的较短，R 系的较长。

现在我们对这一节的内容进行简要回顾，不同机器有不同的指令系统。一个较完善的指令系统应当包含数据传送类指令、算术运算类指令、逻辑运算类指令、程序控制类指令、I/O 类指令、字符串类指令、系统控制类指令。RISC 指令系统是目前计算机发展的主流，也是 CISC 指令系统的改进，它的最大特点是：①指令条数少；②指令长度固定，指令格式和寻址方式种类少；③只有取数/存数指令访问存储器，其余指令的操作均在寄存器之间进行。至此，指令系统这一章已全部讲解完毕。

希望同学们做好第五章中央处理器的预习，为后面学习 CPU 的功能与组成、指令周期流程图、微程序控制器工作原理、微命令编码方法、微地址形成方式、微指令格式设计、流水线时空图及其相关问题，做好充分的课前准备。

（本节讲授完毕）

第5章　中央处理器（4 学时）

同学们，通过前面章节的学习，我们知道，程序是由一系列计算机指令组成，它们按顺序存放在存储器当中。那么程序中的不同指令是如何执行的？具体的执行过程由计算机的哪个部件完成呢？它又是如何控制程序执行的呢？对，这些都是由 CPU 实现的。CPU 要实现这些功能，必须具备什么样的组成和结构，以及相应部件的工作原理如何，用什么方法进行设计……，针对这些问题，我们一起来认真学习第五章中央处理器。

5.1 CPU 的功能和组成

5.1.1 CPU 的功能

（PPT511 第 1 页）同学们，当我们用计算机解决某个问题时，首先必须为它编写程序，使用指令明确告诉计算机应该执行什么操作，在什么地方找到所需要的操作数据。（动画 1）一旦把程序装入内存，就可以由相应的部件来自动完成取出指令和执行指令的任务。专门用来完成此项工作的计算机部件称为中央处理器，简称 CPU。

因此，CPU 对整个计算机系统的运行是极其重要的，它具有如下四个基本功能:

（动画 2）程序的顺序控制，称为指令控制。由于程序是一个指令序列，指令间的相互顺序不能任意颠倒，必须严格按程序规定的顺序执行，因此，保证机器按顺序执行程序是 CPU 的首要任务。

（动画 3）一条指令的功能往往是由若干个操作信号的组合来实现的，因此，CPU 管理并产生由内存取出的每条指令的操作信号，把它们送往相应的部件，从而控制这些部件按指令的要求进行动作。

（动画 4）对各种操作实施时间上的定时，称为时间控制。因为在计算机中，各种指令的操作信号均受到时间的严格定时。另一方面，一条指令的整个执行过程也受到时间的严格定时。只有这样，计算机才能有条不紊地自动工作。

（动画 5）所谓数据加工，就是对数据进行算术或逻辑运算处理。完成数据的加工处理，是 CPU 的根本任务。因为，原始信息只有经过加工处理后才能对人们有用。

5.1.2 CPU 的基本组成

（PPT512 第 1 页）（动画 1）早期的 CPU 由运算器和控制器两大部分组成。（动画 2）但是随着超大规模集成电路技术的发展，可将早期放在 CPU 芯片外部的一些逻

辑功能部件，如浮点运算器、Cache、总线仲裁器等纷纷移入 CPU 内部，这样 CPU 的基本组成变成运算器、Cache 和控制器三大部分。

（PPT512 第 2 页）从教学目的出发，本章以 CPU 执行指令为主线来组织教学内容。为便于同学们建立计算机的整机概念，突出主要矛盾，给出如图所示的 CPU 模型。它主要包括三部分:

（动画 1）一、运算器：包括算术逻辑单元(ALU)、通用寄存器、数据缓冲寄存器 DR 和状态条件寄存器 PSW 组成。它是数据加工处理部件，接受控制器的命令而进行动作。（动画 2）其主要功能包括：1）执行所有的算术运算；2）执行所有的逻辑运算，并进行逻辑测试，如零值测试或两个值的比较。

（动画 3）二、Cache：分为指令 Cache 和数据 Cache 两部分。前者存放程序中的指令，简称指存，后者存放指令执行时所需要的数据，简称数存。

（PPT512 第 3 页）（动画 1）三、控制器：包括程序计数器 PC、指令寄存器 IR、指令译码器、时序产生器和操作控制器 OC 组成，它是发布命令的“决策机构”，即完成协调和指挥整个计算机系统的操作，其主要功能有（动画 2）:

1）从指令 Cache 中取出一条指令，并指出下一条指令在指令 Cache 中的位置；

2）对指令进行译码或测试，并产生相应的操作控制信号，以便启动规定的动作。比如一次数据 Cache 的读/写操作，一个算术/逻辑运算操作，或一个输入/输出操作。

3）指挥并控制 CPU、数据 Cache 和输入/输出设备之间数据流动的方向。

前面章节已详细讨论了运算器和 Cache，所以本章重点放在控制器上。

5.1.3 CPU 中的主要寄存器

（PPT513 第 1 页）（动画 0）各种计算机的 CPU 可能有这样或那样的不同，但是在 CPU 中至少要有六类寄存器，包括：指令寄存器（IR）；程序计数器（PC）；数据地址寄存器（AR）；数据缓冲寄存器（DR）；通用寄存器（R0-R3）；状态条件寄存器（PSW）。下面我们详细介绍这些寄存器的功能和结构。

1、数据缓冲寄存器（DR），用来暂时存放 ALU 的运算结果，或由数存读出的一个数据字，或来自外部接口的一个数据字。因此，DR 的作用是：①作为 ALU 运算结果和通用寄存器之间信息传送中时间上的缓冲；②补偿 CPU 和内存、外设之间在操作速度上的差别。

2、指令寄存器（IR），用来保存当前正在执行的一条指令。当执行一条指令时，

先把它从指令 cache 中读出，然后再传送至 IR。而指令包括操作码和地址码两部分，由二进制数字组成。为了执行任何给定的指令，必须由指令译码器对 OP 字段进行测试，以便识别所要求的操作。操作码经译码后，即可向操作控制器 OC 发出具体操作的特定信号。

3、程序计数器（PC），为了保证程序能够连续地执行下去，CPU 必须具有某些手段来确定下一条指令的地址。而程序计数器(PC)正是起到这种作用，所以它又称为指令计数器。在程序开始执行前，必须将它的起始地址（或入口地址），即程序的第一条指令所在的指存单元地址送入 PC。当执行指令时，CPU 将自动修改 PC 的内容，以便使其保持的总是将要执行的下一条指令的地址。由于大多数指令都是顺序执行的，所以修改的过程通常只是简单的对 PC 加 1。

但是，当遇到转移指令如 JMP 指令时，后继指令的地址就必须从指令寄存器中的地址码字段取得转移目标地址。在这种情况下，下一条从指存取出的指令将由转移指令来规定。因此程序计数器 PC 的结构应当是具有寄存器和计数两种功能的结构。

4、数据地址寄存器（AR） 用来保存当前 CPU 所访问的数据 cache 相应单元的地址。因为访存时要对存储器阵列进行地址译码，所以必须使用 AR 来保持地址信息，直到一次读/写操作完成为止。

AR 的结构和 DR、IR 一样， 通常使用单纯的寄存器结构。信息的存入一般采用电位—脉冲方式，即电位输入端对应数据信息位，脉冲输入端对应控制信号，在控制信号作用下，瞬时地将信息打入寄存器。

5、通用寄存器（R0-R3）

本 CPU 模型中，通用寄存器有 4 个(R0~R3)，其功能是：当 ALU 执行算术或逻辑运算时，为 ALU 提供两个工作区。

6、状态字寄存器（PSW） 保存由算术/逻辑指令运算或测试结果建立的各种条件代码，如进位标志(C)、溢出标志(V)、结果为零标志(Z)、结果为负标志(N)等等。这些标志位通常分别由 1 位触发器保存。

除此之外，PSW 还保存中断和系统工作状态等信息，以便使 CPU 和系统能及时了解机器运行状态和程序运行状态。

5.1.4 操作控制器与时序发生器

（PPT514 第 1 页）前面我们讲了 CPU 中的 6 类主要寄存器，每一类完成一种特

定的功能。然而信息怎样才能在各寄存器之间传送呢？也就是说，数据的流动是由什么部件控制的呢？（动画 1）仔细看图 5.1，我们还有哪些部件没有介绍呢？对，操作控制器和时序发生器。

通常把许多寄存器之间传送信息的通路，称为数据通路，如图中数据总线 DBUS、指令总线 IBUS 等。信息从什么地方开始，中间经过哪个寄存器或三态门，最后传送到哪个寄存器，都要加以控制。（PPT514 第 2 页）（动画 1）在各寄存器之间建立数据通路的任务，是由称为**操作控制器 OC** 的部件来完成的。它的功能就是根据指令译码结果和时序信号，产生各种操作控制信号 C_1，C_2，...，C_n，以便正确地选择数据通路，把有关数据打入到一个寄存器，从而完成取指令和执行指令的控制。

（动画 2）根据设计方法不同，**操作控制器**可分为**时序逻辑型**和**存储逻辑型**两种。前者称为硬布线控制器，它是采用时序逻辑技术来实现的；后者称为微程序控制器，它是采用存储逻辑来实现的。本书重点介绍微程序控制器。

（动画 3）操作控制器产生的控制信号必须定时，为此必须有**时序产生器**。因为计算机高速地进行工作，每一个动作的时间是非常严格的，不能太早也不能太晚。时序产生器的作用，就是对各种操作信号实施时间上的控制。

CPU 中除了上述组成部分外，还有中断系统、总线接口等其他功能部件，这些内容将在以后各章中陆续展开。

5.2 指令周期

5.2.1 指令周期的基本概念

学习汇编语言课程时，我们知道，指令和数据从形式上看，都是二进制代码， 所以人们很难区分出这些代码是指令还是数据。然而，CPU 却能识别这些二进制代码，它能准确地判别出哪些是指令字，哪些是数据字，并将它们送往相应的地方。本节我们将讨论在一些典型的指令周期中，CPU 的各部分是怎样工作的，从而能加深对这一问题的理解和体验。

（PPT521 第 1 页）（动画 1）计算机之所以能自动地工作，是因为 CPU 能从存放程序的内存中取出一条指令并执行这条指令，紧接着又是取指令，执行指令……如此周而复始，构成了一个封闭的循环。除非遇到停机指令，否则这个循环将一直继续下去，其过程如图所示。

CPU 每取出一条指令并执行这条指令，都要完成一系列的操作，这一系列操作所

需的时间通常叫做一个**指令用期**，换言之（动画 2）**指令周期**是取出一条指令并执行这条指令的时间。（动画 3）由于各种指令的操作功能不同，因此各种指令的指令周期是不尽相同的。指令周期常常用若干个 **CPU 周期**来表示。

（PPT521 第 2 页）（动画 1）CPU 周期又称为**机器周期**。CPU 访问一次内存所花的时间较长，因此通常用从内存中读取一个指令字的最短时间来规定 **CPU 周期**。也就是说，取出一条指令需要一个 CPU 周期时间。（动画 2）而一个 CPU 周期时间又包含有若干个 T 周期(通常称为节拍脉冲，它是处理操作的最基本单位)。这些 T_i 周期的总和则规定了一个 CPU 周期的时间宽度。

（动画 3）图中给出了采用定长 CPU 周期的指令周期示意图。从中可以看出，取出和执行任何一条指令所需的最短时间为两个 CPU 周期。此外，同学们还需知道两个概念，即单周期和多周期。所谓单周期，就是在一个 CPU 周期中完成取指和执行操作(少数指令可实现)。大多数指令需要在多个 CPU 周期中完成指令周期的全部操作。

（PPT521 第 3 页）表中列出了由 6 条指令组成的一个简单程序。这 6 条指令是有意安排的，因为它们是非常典型的，既有 RR 型指令，又有 RS 型指令；既有算术/逻辑指令，又有访存指令，还有程序转移指令。我们将在下一节通过 CPU 取出一条指令并执行这条指令的分解动作，来具体认识每条指令的指令周期。

5.2.2 五种典型指令的指令周期

同学们，本节课我们一起来分析五种典型指令的指令周期。

一、MOV 指令的指令周期

（PPT522 第 1 页）首先看第一条指令，即 MOV 指令的指令周期。（动画 1）它是一条 RR 型指令。其指令周期如图 5.4 所示，需要两个 CPU 周期。其中，取指周期需要一个 CPU 周期，执行周期也需要一个 CPU 周期。（动画 2-4）取指周期中 CPU 完成三件事：①从指存取出指令；②对程序计数器 PC 加 1，以便为取下一条指令做好准备；③对指令操作码进行译码或测试。执行周期中 CPU 根据对 OP 字段的译码结果，进行指令所要求的操作。对 MOV 指令来说，就是将寄存器 R1 的内容传送给 R0。

（PPT522 第 2 页）下面我们通过动画看看 MOV 指令的取指周期和执行周期，CPU 中具体操作过程。

假定表 5.1 的程序已装入指存中，因而 MOV 指令的取指周期内， CPU 的动作如下（动画 1-5）：1)程序计数器 PC 中装入第一条指令地址 101(八进制)；2)PC 的内容被

放到指令地址总线 ABUS(I)上，对指存进行译码，并启动读命令；3）从 101 号单元读出的 MOV 指令通过指令总线 IBUS 装入指令寄存器 IR；4）PC 内容加 1，变成 102，为取下一条指令做好准备；5)指令寄存器中的操作码被译码；6）CPU 识别出是 MOV 指令。至此，取指周期即告结束。

（动画 6-9）MOV 指令的执行周期中，CPU 的动作如下：1)操作控制器(OC)送出控制信号到通用寄存器，选择 R1 作源寄存器，其内存有值 10，选择 R0 作目标寄存器；2）OC 送出控制信号到 ALU，指定 ALU 做传送操作；3)OC 送出控制信号，打开 ALU 的输出三态门，将其输出送到数据总线 DBUS 上。注意，任何时候 DBUS 上只能有一个数据。4) OC 送出控制信号，将 DBUS 上的数据打入到数据缓冲寄存器 DR；5) OC 送出控制信号，将 DR 中的数据 10 打入到目标寄存器 R0，R0 的内容由 00 变为 10。至此，MOV 指令执行结束。

二、指令 LAD 的指令周期

（PPT522 第 3 页）接下来我们看第二条指令 LAD 的指令周期。（动画 1-6）LAD 指令是 RS 型指令，它先从指存中取出指令，然后从数存 6 号单元取出数据 100 装入寄存器 R1，R1 中原来存放的数据 10 被更换成 100。由于一次访问指存，一次访问数存，LAD 指令的指令周期需要 3 个 CPU 周期，如图 5.7 所示。

（PPT522 第 4 页）（动画 1-4）在 LAD 指令的取指周期中，CPU 的动作完全与 MOV 指令的取指周期中一样，只是 PC 提供的指令地址为 102，按此地址从指存中读出“LDA R1，6”放入 IR 中，然后将 PC+1，使 PC 内容变成 103，为取下一条 ADD 指令做好准备。以下 ADD、STO、JMP 三条指令的取指周期中，CPU 的动作完全与 MOV 指令一样，后面将不再细述。

LAD 指令执行周期的第一个 CPU 周期中，送操作数地址，CPU 执行的动作如下：（动画 5）1)操作控制器 OC 发出控制命令打开 IR 输出三态门，将指令中的直接地址码 6 放到数据总线 DBUS 上；2)OC 发出操作命令，将地址码 6 装入数存地址寄存器 AR。LAD 指令执行周期的第二个 CPU 周期中，取出操作数，装入通用寄存器，CPU 执行的动作如下：1)OC 发出读命令，将数存 6 号单元中的数 100 读出到 DBUS 上；2）OC 发出命令，将 DBUS 上的数据 100 装入 DR；3）OC 发出命令，将 DR 中的数 100 装入 R1， R1 中原来的数 10 被冲掉。至此，LAD 指令执行周期结束。

注意，数据总线 DBUS 上分时进行了地址传送和数据传送，所以需要 2 个 CPU 周期。

三、ADD 指令的指令周期

（PPT522 第 5 页）（动画 1-4）ADD 是 RR 型指令，在运算器中，两个寄存器 R1 和 R2 的数据进行加法运算，指令周期只需两个 CPU 周期。

（PPT522 第 6 页）（动画 1-4）ADD 指令的取指周期中 CPU 的操作与 MOV、LAD 指令相同。下面只讲执行周期，CPU 完成的动作如下（动画 5-7）：1)OC 送出控制命令到通用寄存器，选择 R1 做源寄存器，R2 既做源寄存器又做目标寄存器；2)OC 送出控制命令到 ALU，指定 ALU 做 R1(100)和 R2(20)的加法操作；3)OC 送出控制命令，打开 ALU 输出三态门，运算结果 120 放到 DBUS 上；4)OC 送出控制命令，将 DBUS 上的数据打入 DR，ALU 加法操作产生的进位信号保存在状态字寄存器 PSW 中；5)OC 送出控制命令，将 DR(120)装入 R2，R2 中原来的内容 20 被冲掉。至此，ADD 指令执行周期结束。

四、STO 指令的指令周期

（PPT522 第 7 页）（动画 1-7）STO 指令是 RS 型指令，它先访问指存取出 STO 指令，然后按(R3)=30，以寄存器间接方式形成操作数有效地址 30，访问数存，将 R2 中的 120 写入数存 30 号单元。由于一次访问指存，一次访问数存，因此指令周期需要 3 个 CPU 周期，其中执行周期占 2 个。

（PPT522 第 8 页）（动画 1-4）STO 指令的取指周期 CPU 的操作与 MOV 指令相同。下面也只讲执行周期，CPU 完成的动作如下：（动画 5）1)操作控制器 OC 送出操作命令到通用寄存器，选择(R3)=30 做数存的地址单元；（动画 6）2)OC 发出操作命令，打开通用寄存器输出三态门(注意，此处不经 ALU 以节省时间)，将地址 30 放到 DBUS 上：3)OC 发出操作命令，将地址 30 打入 AR，并进行数存地址译码；（动画 7）4)OC 发出操作命令到通用寄存器，选择 R2 中的 120，作为数存的写入数据；5)OC 发出操作命令，打开通用寄存器输出三态门，将数据 120 放到 DBUS 上；6)OC 发出操作命令，将数据 120 写入数存 30 号单元，它原先的数据 40 被冲掉。至此，STO 指令执行周期结束。

注意，DBUS 是单总线结构，先送地址(30)，后送数据(120)，必须分时传送。

五、JMP 指令的指令周期

（PPT522 第 9 页）（动画 1-4）JMP 指令是一条无条件转移指令，用来改变程序的执行顺序。指令周期为两个 CPU 周期，其中取指周期和执行周期各为 1 个 CPU 周期。（PPT522 第 10 页）（动画 1-5）下面也只看其执行周期，CPU 完成的动作如下：

1)OC 发生操作控制命令，打开指令寄存器 IR 的输出三态门，将 IR 中的转移目标地址 101 发送到 DBUS 上；2)OC 发出操作控制命令，将 DBUS 上的地址码 101 打入到程序计数器 PC 中，PC 中的原内容 106 被更换。于是下一条指令不是从指存的 106 号单元取出，而是转移到 101 号单元取出。至此，JMP 指令执行周期结束。

应当指出，执行“JMP 101”指令时，我们所给的五条指令组成的程序进入了死循环，除非人为停机，否则这个程序将无休止地运行下去。当然，我们此处所举的转移目标地址 101 是随意的，仅仅用来说明转移指令能够改变程序的执行顺序而已。

5.2.3 用方框图语言表示指令周期

上一节介绍了五条典型指令的指令周期，从而使我们对一条指令的**取指过程**和**执行过程**有了一个较深刻的印象。然而我们是通过画示意图或数据通路图，来解释这些过程的。这样做主要是为了教学的目的。但是，在进行计算机设计时，若用这种办法来表示指令周期，就显得过于烦琐，而且也没有必要。

（PPT523 第 1 页）通常在进行计算机设计时，可采用方框图语言来表示指令的指令周期。（动画 1）一个**方框**代表一个 **CPU 周期**，（动画 2）方框中的内容表示**数据通路的操作**或**某种控制操作**。（动画 3）除了方框以外，还需要一个**菱形**符号，它通常用来表示某种判别或测试，不过时间上它依附于紧接它的前面一个方框的 CPU 周期，而不单独占用一个 CPU 周期。

（动画 4）我们把前面的五条典型指令加以归纳，用方框图语言表示的指令周期如图所示。可以明显地看到，所有指令的取指周期是完全相同的，而且只需一个 CPU 周期。但是指令的执行周期，由于各条指令的功能不同，所需的 CPU 周期是各不相同的，其中 MOV、ADD、JMP 指令是一个 CPU 周期；LAD 和 STO 指令是两个 CPU 周期。框图中 DBUS 代表数据总线，ABUS(D)代表数存地址总线，ABUS(I)代表指存地址总线，RD(D)代表数存读命令，WE(D)代表数存写命令，RD(I)代表指存读命令。

图中，还有一个“~”符号，我们称它为公操作符号。它表示一条指令已执行完毕，转入公操作。所谓公操作，就是一条指令执行完毕后，CPU 所开始进行的一些操作，主要是 CPU 对外设请求的处理，如中断处理、通道处理等。如果外设没有向 CPU 请求交换数据，那么 CPU 又转向从指存取下一条指令。由于所有指令的取指周期是完全一样的，因此，取指令也可以认为是公操作。这是因为，一条指令执行结束后，如果没有外设请求，CPU 一定转入“取指令”操作。

（PPT523 第 2 页）（动画 1）下面我们通过一个例子，来看看如何根据数据通路和指令功能，用方框图语言给出指令周期流程图，并列出相应的微操作控制信号序列。

（动画 2）**[例 1]**如图所示为双总线结构机器的数据通路，有四个专用寄存器 IR、PC (具有自增功能)、AR 和 DR，还有四个通用寄存器 R0-R3，M 为主存(受 R/W 非信号控制)，它既存放指令又存放数据，ALU 由+、-控制信号决定完成加、减法操作，控制信号 G 控制一个门电路，它相当于两条总线之间的桥。另外，线上标注有小圆表示有控制信号，例如，Y_i 表示 Y 寄存器的输入控制信号，$R1_o$ 为寄存器 R1 的输出控制信号，未标字符的线为直通线，不受控制。

（PPT523 第 3 页）（动画 1）(1) ADD R2，R0；假设该指令在内存 M 的地址已放入 PC 中。

（动画 2）解：(1) “ADD R2，R0”指令是一条加法指令，参与运算的两个数放在寄存器 R2 和 R0 中，指令周期流程图包括取指阶段和执行阶段两部分(为简单起见，省去了“->”号左边各寄存器代码上应加的括号)。根据给定的数据通路图，“ADD R2，R0”指令的详细指令周期流程图如图中（a）所示，图的右边部分标注了每个机器期中用到的微操作控制信号序列。

取指阶段需要 3 个 CPU 周期，1)按 PC 中加法指令的地址，经 B 总线、G 信号控制的门电路、A 总线，打入 AR；2）按 AR 中的地址读存储器 M，取加法指令到 DR；3）将 DR 中的加法指令经 B 总线、G 门电路、A 总线，打入 IR，操作码译码。至此，加法指令取指阶段结束；

执行阶段也需要 3 个 CPU 周期，1）R2 寄存器的值，经 B 总线、G 门电路、A 总线，打入 ALU 的 Y 寄存器；2）R0 寄存器的值，经 B 总线、G 门电路、A 总线，打入 ALU 的 X 寄存器；3）ALU 完成加法，结果经 B 总线、G 门电路、A 总线，打入 R0 寄存器；至此，加法指令执行阶段结束。

从中，我们可以看出两点：1）只是一种陈旧的 CPU 的结构，A 总线负责输入，B 总线负责输出；2）由于指令和数据存放在存储器，而不是 CPU 片内 Cache，故而，指令周期较长，机器效率较低。

（动画 3）(2) SUB R1, R3；其指令周期流程如图（b）所示，具体细节留给同学们自己分析。

5.3 时序产生器和控制方式

5.3.1 时序信号的作用和体制

我们知道，在日常生活中，人们学习、工作和休息都有一个严格的作息时间。比如，早晨 6:00 起床；8:00~12:00 上课，12:00~14:00 午休，……，每个教师和学生都必须严格遵守这一规定，在规定的时间里上课，在规定的时间里休息，不得各行其是，否则就难以保证正常的教学秩序。

（PPT531 第 1 页）（动画 1）CPU 中也有一个类似“作息时间”的东西，称为时序信号。计算机之所以能够准确、迅速、有条不紊地工作，正是因为在 CPU 中有一个时序信号产生器。机器一旦被启动，即 CPU 开始取指令并执行指令时，操作控制器就利用定时脉冲的顺序和不同的脉冲间隔，有条理、有节奏地指挥机器的动作，规定在这个脉冲到来时做什么，在那个脉冲到来时又做什么，给计算机各部分提供工作所需的时间标志。为此，需要采用如同年、月、日、时、分、秒类似的多级时序体制。

从 5.2 节指令周期中我们知道，指令周期包含若干个 CPU 周期，而一个 CPU 周期又包含若干个 T 周期，以便规定在每个周期内 CPU 干什么。这种时间约束对 CPU 来说是非常必要的，否则就可能造成丢失信息或导致错误的结果。

（动画 2）总之，计算机的协调动作需要时间标志，而时间标志是用时序信号来体现的。（动画 3）一般来说，操作控制器发出的各种控制信号都是时间因素(时序信号)和空间因素(部件位置)的函数。如果忽略了时间因素，那么我们学习计算机硬件时往往就会感到困难，这一点务请同学们加以注意。

（动画 4）组成计算机硬件的器件特性决定了时序信号最基本的体制是电位-脉冲制。这种体制最明显的一个例子，就是当实现寄存器之间的数据传送时，数据加在触发器的电位输入端，而打入数据的控制信号加在触发器的时钟输入端。电位的高低，表示数据是 1 还是 0，而且要求打入数据的控制信号到来之前，电位信号必须已稳定。这是因为，只有电位信号先建立，打入到寄存器中的数据才是可靠的。当然，计算机中有些部件，如 ALU 只用电位信号工作就可以了。但尽管如此，运算结果还是要送入通用寄存器，所以最终还是需要脉冲信号来配合。

（PPT531 第 2 页）（动画 1-2）微程序控制器中，时序信号比较简单，一般采用节拍电位-节拍脉冲二级体制。就是说，它有一个节拍电位，在节拍电位中又包含若干个节拍脉冲(T 周期)。节拍电位表示一个 CPU 周期的时间，而节拍脉冲把一个 CPU 周

期划分成几个较小的时间间隔。根据需要，这些时间间隔可以相等，也可以不相等。

（动画 3-4）而硬布线控制器中，时序信号往往采用主状态周期-节拍电位-节拍脉冲三级体制。主状态周期可包含若干个节拍电位，所以它是最大的时间单位。主状态周期可以用一个触发器的状态持续时间来表示。一个节拍电位表示一个 CPU 周期，它表示了一个较大的时间单位；在一个节拍电位中又包含若干个节拍脉冲，用以表示较小的时间单位。

5.3.2 时序信号产生器

前面已分析了指令周期中需要的一些典型时序。（PPT532 第 1 页）（动画 1-2）时序信号产生器的功能是用逻辑电路来实现这些时序。

（动画 3）各种计算机的时序信号产生电路是不尽相同的。（动画 4）一般来说，大型计算机的时序电路比较复杂，而微型机的时序电路比较简单，这是因为前者涉及的操作动作较多，后者涉及的操作动作较少。另一方面，从设计操作控制器的方法来讲，硬布线控制器的时序电路比较复杂，而微程序控制器的时序电路比较简单。然而不管是哪一类，时序信号产生器最基本的构成是一样的。

（PPT532 第 2 页）（动画 1-2）下面我们来看看微程序控制器中使用的时序信号产生器的结构图，（动画 3-6）它由时钟源、环形脉中发生器、节拍脉冲和读写时序译码逻辑、启停控制逻辑等部分组成。

1)时钟源　用来为环形脉冲发生器提供频率稳定且电平匹配的方波时钟脉冲信号。它通常由石英晶体振荡器和与非门组成的正反馈振荡电路组成，其输出送至环形脉冲发生器。

2)环形脉冲发生器　它的作用是产生一组有序的间隔相等或不等的脉冲序列，以便通过译码电路来产生最后所需的节拍脉冲，其电路同学们查阅《数字逻辑》(第五版)相关内容。

（PPT532 第 3 页）3)节拍脉冲和存储器读/写时序　我们假定在一个 CPU 周期中产生四个等间隔的节拍脉冲 T_1^*~ T_4^*，每个节拍脉冲的脉冲宽度均为 200ns。因此一个 CPU 周期便是 800ns，在下一个 CPU 周期中，它们又按固定的时间关系重复。

不过注意，节拍电位与节拍脉冲时序关系图中，画出的节拍脉冲信号是 T_1~T_4，它们在逻辑关系上与 T_1^*~ T_4^*是完全一致的，是后者经过启停控制逻辑中与门以后的输出，图中忽略了一级与门的时间延迟细节。

存储器读/写时序信号 $\overline{RD}$、$\overline{WE}$用来进行存储器的读/写操作。

在硬线控制器中，节拍电位信号是由时序产生器本身通过逻辑电路产生的，一个节拍电位持续时间正好包容若干个节拍脉冲。然而在微程序设计的计算机中，节拍电位信号可由微程序控制器提供。一个节拍电位持续时间，通常也是一个 CPU 周期的时间。（PPT532 第 4 页）（动画 1）例如，起停控制逻辑中的 $\overline{RD}$、$\overline{WE}$信号持续时间均为 800ns，而一个 CPU 周期的时间也正好是 800ns。

4)启停控制逻辑 机器一旦接通电源，就会自动产生原始的节拍脉冲信号 $\overline{T}_1$~ $\overline{T}_4$，然而，只有在启动机器运行的情况下，才允许时序产生器发出 CPU 工作所需的节拍脉冲 T_1~T_4。为此，需要由启停控制逻辑来控制 $\overline{T}_1$~ $\overline{T}_4$的发送。同样，对读/写时序信号也需要由启停逻辑加以控制。

启停控制逻辑的核心是一个运行标志触发器 Cr。当运行触发器为“1”时，原始节拍脉冲 $\overline{T}_1$~ $\overline{T}_4$和读/写时序信号 $\overline{RD}$、$\overline{WE}$通过门电路发送出去，变成 CPU 真正需要的节拍脉冲信号 T1~T4 和读/写时序 $\overline{RD}$， $\overline{WE}$。反之，当运行触发器为“0”时，就关闭时序产生器。

（动画 2-4）由于启动计算机是随机的，停机也是随机的，为此必须要求：当计算机启动时，一定要从第一个节拍脉冲前沿开始工作，而在停机时一定要在第 4 个节拍脉中结束后关闭时序产生器。只要这样，才能使发送出去的脉冲都是完整的脉冲。图中，在 Cr（D 触发器）下面加上一个 SR 触发器，且用 $\overline{T}_4$信号作 Cr 触发器的时钟控制端，那么就可以保证在 T_1 的前沿开启时序产生器，而在 T_4 的后沿关闭时序产生器。

5.3.3 控制方式

同学们，从 5.2 节我们知道，机器指令的指令周期是由数目不等的 CPU 周期数组成，CPU 周期数的多少反映了指令动作的复杂程度，即操作控制信号的多少。对一个 CPU 周期而言，也有操作控制信号的多少与出现的先后问题。这两种情况综合在一起， 说明每条指令和每个操作控制信号所需的时间各不相同。

（PPT533 第 1 页）（动画 1-2）控制不同操作序列时序信号的方法，称为控制器的控制方式。（动画 3）常用的有同步控制、异步控制、联合控制三种方式，其实质反映了时序信号的定时方式。

（动画 4）第一种是同步控制方式，即在任何情况下，已定的指令在执行时所需的机器周期数和时钟周期数都是固定不变的。

根据不同情况，同步控制方式可选取如下方案:

(1)采用完全统一的机器周期执行各种不同的指令。这意味着所有指令周期具有相同的节拍电位数和相同的节拍脉冲数。显然，对简单指令和简单的操作来说，将造成时间浪费。

(2)采用不定长机器周期。将大多数操作安排在一个较短的机器周期内完成，对某些时间紧张的操作，则采取延长机器周期的办法来解决。

(3)中央控制与局部控制结合。将大部分指令安排在固定的机器周期完成，称为中央控制。对少数复杂指令(乘、 除、浮点运算)采用另外的时序进行定时，称为局部控制。 简单说，这三种方案分别总结为“一视同仁”、“需要多少给多少”、“分类区别对待”。

（动画 5）第二种是异步控制方式，其特点是：每条指令、每个操作控制信号需要多少时间就占用多少时间。这意味着每条指令的指令周期可由多少不等的机器周期数组成；也可以是当控制器发出某一操作控制信号后，等待执行部件完成操作后发回“回答”信号，再开始新的操作。显然，用这种方式形成的操作控制序列没有固定的 CPU 周期数或严格的时钟周期与之同步。

（动画 6）第三种是联合控制方式，就是把同步控制和异步控制相结合的方式。一种情况是，大部分操作序列安排在固定的机器周期中，对某些时间难以确定的操作则以执行部件的“回答”信号作为本次操作的结束。例如，CPU 访问主存时，依靠其送来的“READY”信号作为读/写周期的结束。另一种情况是，机器周期的节拍脉冲数固定，但是各条指令周期的机器周期数不固定。例如下一节将要介绍的微程序控制就是这样。

5.4 微程序控制器

5.4.1 微程序控制器原理

同学们，这一讲我们学习微程序控制器工作原理及微程序实例。（PPT541 第 1 页）（动画 1）微程序控制器同硬布线控制器相比较，具有规整性、灵活性、可维护性等优点。因而在计算机设计中逐渐取代了早期采用的硬步线控制器，并得到了广泛应用。（动画 2）在计算机系统中，微程序设计技术是利用软件方法来设计硬件的一门技术。

（动画 3）微程序控制器的基本思想，就是仿照通常的解题程序的方法，把操作控制信号编成所谓的“微指令”，存放到一个只读存储器里。当机器运行时，一条又一条地读出这些微指令，从而产生全机所需要的各种操作控制信号，使相应部件执所规定的操作。（动画 4）微指令是由若干个微命令组成的。

在这里，我们要给大家讲清楚两组概念，一组是微命令和微操作；另一组是微指令和微程序。

（PPT541 第 2 页）（动画 1）首先，我们来看微命令和微操作。（动画 2）一台数字计算机基本上可以划分为两大部分—控制部件和执行部件。控制器就是控制部件，而运算器、存储器、外围设备相对控制器来讲，就是执行部件。那么两者之间是怎样进行联系的呢？（动画 3）控制部件与执行部件的一种联系，就是通过控制线。控制部件通过控制线向执行部件发出各种控制命令，通常把这种控制命令叫做微命令，而执行部件接受微命令后所进行的操作，叫做微操作。（动画 4）控制部件与执行部件之间的另一种联系是反馈信息。执行部件通过反馈线向控制部件反映操作情况，以便使控制部件根据执行部件的“状态”来下达新的微命令，这也叫做“状态测试”。

（PPT541 第 3 页）（动画 1）微命令和微操作的关系是一一对应的。（动画 2）微操作在执行部件中是最基本的操作。由于数据通路的结构关系，微操作可分为相容性和相斥性两种。（动画 3）所谓相容性的微操作，是指在同时或同一个 CPU 周期内可以并行执行的微操作。（动画 3）所谓相斥性的微操作，是指不能在同时或不能在同一个 CPU 周期内并行执行的微操作。

（PPT541 第 4 页）图中给出了一个简单运算器模型，其中，ALU 为算术/逻辑单元，R1、R2、R3 为三个寄存器，其内容都可以通过多路开关从 ALU 的 X 输入端或 Y 输入端送至 ALU，而 ALU 的输出可以送往任何一个寄存器或同时送往 R1、R2、R3 三个寄存器。

多路开关的每个控制门仅是一个常闭的开关，它的一个输入端代表来自寄存器的信息，而另一个输入端则作为操作控制端。一旦两个输入端都有输入信号，它才产生一个输出信号，从而在控制线能起作用的一个时间宽度来控制信息在部件中流动。图中每个开关由控制器中相应的微命令来控制，例如，开关门 4 由控制器中编号为 4 的微命令控制，开关门 6 由编号为 6 的微命令控制，如此等等。三个寄存器 R1，R2，R3 的时钟输入端 1，2，3 也需要加以控制，以便在 ALU 运算完毕而输出公共总线上电平稳定时，将结果打入到某一寄存器。另外，我们假定 ALU 只有+，-，M(传送)三

种操作。Cy 为最高进位触发器，有进位时该触发器状态为“1”。

（动画 1）ALU 的操作（加、减、传送）在同一个 CPU 周期中只能选择一种，不能并行，所以+，-，M（传送）三个微操作是相斥性的微操作。类似的 4，6，8 三个微操作是相斥性的，5，7，9 三个微操作也是相斥性的。微操作 1，2，3 是可以同时进行的，所以是相容性的微操作。另外，ALU 的 X 端输入微操作 4，6，8 分别与 Y 输入的 5，7，9 任意两个微操作，也都是相容性的。

（PPT541 第 5 页）接下来，我们看看另一组概念:（动画 1）微指令和微程序。（动画 2）在机器的一个 CPU 周期中，一组实现一定操作功能的微命令的组合，构成一条微指令。（动画 3）图中给出了一个具体的微指令结构，微指令字长为 23 位，它由操作控制和顺序控制两大部分组成。

操作控制部分用来发出管理和指挥全机工作的控制信号。在本微指令实例中，该字段为 17 位，每一位表示一个微命令。每个微命令的编号同图 5.20 所示的数据通路相对应，具体功能示于微指令格式的左上部。（动画 4）当操作控制字段某位信息为“1”时，表示发出微命令；而某位信息为“0”时，表示不发出微命令。例如，当微指令字第 1 位信息为“1”时，表示发出 LDR1'的微命令，那么运算器将执行 ALU—>R1 的微操作，把公共总线上的信息打入到寄存器 R1。同样，当微指令第 10 位为“1”时，则表示向 ALU 发出进行“+”的微命令，因而 ALU 就执行加法运算。

微指令格式中的顺序控制部分用来决定产生下一条微指令的地址。图中，顺序控制字段设置 6 位，其中，（动画 5）第 18，19 两位作为判别测试标志 P1，P2。当 P1，P2 为“0”时表示不进行测试，按后面 4 位（第 20-23 位，即（动画 6）直接地址）给出的地址取下一条微指令；当 P1 或 P2 某一位置“1”时，表示要进行 P1 或 P2 的判别测试，根据测试结果，需要对直接地址的某一位或几位进行修改，然后按修改后的地址取下一条微指令。

（动画 7）下面我们将会知道，一条机器指令的功能是用许多条微指令组成的序列来实现的，这个微指令序列通常叫做微程序。

（PPT541 第 6 页）这里，（动画 1）需要注意的是:（动画 2）图 5.21 中微指令给出的控制信号都是节拍电位信号，它们的持续时间都是一个 CPU 周期。如果要用来控制图 5.20 所示的运算器数据通路，势必会出现问题，因为前面的三个微命令信号(LDR1'，LDR2'，LDR3')既不能来得太早，也不能来得太晚。为此，（动画 3）要求这些微命令信号还要加入时间控制，（动画 4）例如同节拍脉冲 T4 相与而得到

LDR1~LDR3 信号，如图中右边所示。在这种情况下，控制器最后发给运算器的 12 个控制信号中，3 个是节拍脉冲信号(LDR1，LDR2，LDR3)，其他 9 个都是节拍电位信号，从而保证运算器在前 600ns 时间内进行运算，运算完毕后，公共总线上输出稳定的运算结果，由 LDR1(或 LDR2，LDR3)信号打入到相应的寄存器，其时间关系如图左边所示。

下面，（PPT541 第 7 页）（动画 1）我们一起来看看微程序控制器的原理框图。微程序控制器主要由控制存储器、微指令寄存器和地址转移逻辑三大部分组成，其中，微指令寄存器分为微地址寄存器和微命令寄存器两部分。

（动画 2）我们首先看控制存储器，它用来存放实现全部指令系统的微程序，它是一种只读型存储器。一旦微程序固化，机器运行时则只读不写。其工作过程是:每读出一条微指令，则执行这条微指令；接着又读出下一条微指令，又执行这一条微指令……。读出一条微指令并执行微指令的时间总和称为一个微指令周期。通常，在串行方式的微程序控制器中，微指令周期就是只读存储器的工作周期。控制存储器的字长就是微指令字的长度，其存储容量视机器指令系统而定，即取决于微程序的数量。对控制存储器的要求是速度快，读出周期短。

（动画 3）其次，我们来看微指令寄存器。微指令寄存器用来存放由控制存储器中读出的一条微指令。其中，（动画 4）微地址寄存器决定将要访问的下一条微指令的地址，而（动画 5）微命令寄存器则保存一条微指令的操作控制字段和判别测试字段的信息。

（动画 6）最后是地址转移逻辑。在一般情况下，微指令由控制存储器读出后直接给出下一条微指令的地址，即微地址，这个微地址信息就存放在微地址寄存器中。如果微程序不出现分支，那么下一条微指令的地址就直接由微地址寄存器给出。当微程序出现分支时，意味着微程序出现条件转移。在这种情况下，通过判别测试字段 P 和执行部件的“状态条件反馈信息”，去修改微地址寄存器的内容并按改好的内容去读下一条微指令。因此，地址转移逻辑就承担自动完成修改微地址的任务。

5.4.2 微程序举例

（PPT542 第 1 页）一条机器指令是由若干条微指令组成的序列来实现的。因此，一条机器指令对应着一个微程序，而微程序的总和便可实现整个指令系统。（动画 1）现在我们以“十进制加法”指令为例，具体看看微程序控制的过程。

（动画 2） “十进制加法”指令的功能是用 BCD 码来完成十进制数的加法运算。在十进制运算时，当两数相加之和大于 9 时，便产生进位。可是用 BCD 码完成十进制数运算时，当和数大于 9 时，必须对和数进行加 6 修正。这是因为，采用 BCD 码后，在两数相加的和数小于等于 9 时，十进制运算的结果是正确的；而当两数相加的和数大于 9 时，结果不正确，必须加 6 修正后才能得出正确结果。

（动画 3）假定指令存放在指存中，数据 a、b 及常数 6 已存放在图中的 R1、R2、R3 三个寄存器中。因此，（动画 4）完成十进制加法的微程序流程图如图 5.24。执行周期要求先进行 a+b+6 运算，然后判断结果有无进位：当进位标志 Cy=1 时，不减 6；当 Cy=0，减去 6，从而获得正确结果。

可以看到，十进制加法微程序流程图由四条微指令组成，每一条微指令用一个长方框表示。第一条微指令为“取指”周期，它是一条专门用来取机器指令的微指令，负责从内存取出一条机器指令，放到指令寄存器 IR，然后程序计数器 PC 加 1，做好取下一条机器指令的准备，最后，对机器指令的操作码用 P1 进行判别测试，然后修改微地址寄存器内容，给出下一条微指令的地址。

在微程序流程图中，每一条微指令的地址用数字示于长方框的右上角。注意，菱形代表判别测试，它的动作依附于第一条微指令。第二条微指令完成 a+b 运算。第三条微指令完成 a+b+6 运算，同时又用 P2 测试进位标志 Cy。根据测试结果，微程序或者转向公操作，或者转向第四条微指令。当微程序转向公操作时，如果没有中断请求，则继续取下一条机器指令。与此相对应，第三条微指令和第四条微指令的下一个微地址就又指向第一条微指令，即 “取指”微指令。

假设我们已经按微程序流程图编好了微程序，并已事先存放到控制存储器中。同时假定用图所示的运算器做执行部件。机器启动时，只要给出控制存储器的首地址，就可调出所需要的微程序。为此，首先给出第一条微指令的地址 0000，经地址译码，访问控制存储器读取该地址所对应的“取值”微指令，并将其读到微指令寄存器中。（PPT542 第 2 页）下面我们按照微指令格式来分别给出四条微指令的编码。（动画 1）第一条微指令的二进制编码如动画所示，它的操作控制字段需要发出五个微命令：

第 16 位发出 PC→ABUS(I)，将 PC 内容送到指存地址总线 ABUS(I)；第 13 指存读命令 RD(I)，从指存单元取出“十进制加法”指令放到指令总线 IBUS 上；第 15 位发出 LDIR'，将指令总线 IBUS 上的“十进制加法”指令打入到指令寄存器 IR。假定 “十进制加法” 指令的操作码为 1010，那么指令寄存器的 OP 字段现在是 1010。第 17 位

发出 PC+1 微命令，使程序计数器加 1，做好取下一条机器指令的准备。

另外，顺序控制字段中直接地址是 0000，但是，需要进行 P1 测试，故第 18 位为 1。因此，0000 并不是下一条微指令的真正地址。P1 测试的“状态条件”是指令寄存器 IR 的操作码字段，即用 OP 字段形成下一条微指令的地址。于是微地址寄存器的内容修改成 1010。

在第二个 CPU 周期开始时，按照 1010 这个微地址访问控制存储器，读出第二条微指令，（动画 2）它的二进制编码如动画 2 所示。其操作控制字段发出如下四个微命令: R1→X，R2→Y，+，LDR2'，即第 4、7、10、2 这 4 位置 “1”，其余为 0；不做判别测试，直接地址为 1001。

在第三个 CPU 周期开始时，按照微地址 1001 读出第三条微指令，（动画 3）它的二级制编码如动画 3 所示。它的操作控制部分发出 R2→X，R3→Y，+，LDR2'四个微命令，运算器完成 R2+R3→R2 的操作。顺序控制部分由于判别字段中 P2 为 1，表明进行 P2 测试，测试的 “状态条件” 为进位标志 Cy。换句话说，此时直接地址 0000 需要进行修改，我们假定用 Cy 的状态来修改微地址寄存器的最后一位：当 Cy=0 时，下一条微指令的地址为 0001，需要执行第四条微指令，将多加的 6 再减去；当 Cy=1 时，下一条微指令的地址为 0000，指向 “取指” 微指令。此时十进制加法指令执行结束，转入公操作。

当 Cy=0 时，在第四个 CPU 周期开始时，按微地址 0001 读出第四条微指令，（动画 4）其编码如动画 4 所示，微指令发出 R2→X，R3→Y，-，LDR2'的微命令，运算器完成了 R2-R3→R2 的操作功能。顺序控制部分直接给出下一条微指令的地址为 0000，按该地址取出的微指令是 “取指” 的微指令。

当下一个 CPU 周期开始时，“取指” 微指令又从内存读出第二条机器指令。如果这条机器指令是 STO 指令，那么经过 P1 测试，就转向执行 STO 指令的微程序。

以上是由四条微指令序列组成的简单微程序。从这个简单的控制模板中，我们可以看到微程序控制的主要思想及大概过程。

（PPT542 第 3 页）前面讲解四条微指令时，（动画 1）我们提到 “CPU 周期” 和 “微指令周期”，那么，它们之间的关系是怎样的？我们一起来看看。

（动画 2）在串行方式的微程序控制器中，微指令周期等于读出微指令的时间加上执行该条微指令的时间。为了保证整个机器控制信号的同步，可以将一个微指令周期时间设计得恰好和 CPU 周期时间相等。（动画 3）图中给出了某计算机中 CPU 周期

与微指令周期的时间关系。

假定一个 CPU 周期为 800ns，它包含四个等间隔的节拍脉冲 T1~T4，每个脉冲宽度为 200ns。用 T4 作为读取微指令的时间，用 T1+T2+T3 作为执行微指令的时间。例如，在前 600ns 时间内运算器进行运算，在 600ns 时间的末尾运算器已经运算完毕，可用 T4 上升沿将运算结果打入某个寄存器。与此同时可用 T4 间隔读取下一条微指令，经 200ns 时间延迟，下一条微指令又从控制存储器中读出，并用 T1 上升沿打入到微指令寄存器。如忽略触发器的翻转延迟，那么下一条微指令的微命令信号就从 T1 上升沿起就开始有效，直到下一条微指令读出后打入微指令寄存器为止。因此一条微指令的保持时间恰好是 800ns，也就是一个 CPU 周期的时间。

（PPT542 第 4 页）（动画 1）经过上面的讲述，应该说，我们能够透彻地了解机器指令微地址寄存器与微指令的关系。也许同学们会问:一会儿取机器指令，一会儿取微指令，它们之间到底是什么关系？现在让我们把前面内容归纳一下，作为对此问题的问答。

（动画 2）第一，一条机器指令对应一个微程序，它由若干条微指令序列组成的。因此，一条机器指令的功能是由若干条微指令组成的序列来实现的。第二，从指令与微指令，程序与微程序，地址与微地址的一一对应关系来看，（动画 3）前者与内存储器有关，（动画 4）后者与控制存储器有关。与此相关，也有相对应的硬设备，如图所示（动画 5）。

第三，在本章 5.2 节中，曾讲述了指令与机器周期概念，并归纳了五条典型指令的指令周期(参见图 5.14)。现在我们看到，图 5.14 就是这五条指令的微程序流程图，每一个 CPU 周期就对应一条微指令。这就告诉我们如何设计微程序，也将使我们进一步体验到机器指令与微指令的关系。

5.4.3 微程序设计技术

同学们，上一讲我们学习了微程序控制器的原理。（PPT543 第 1 页）（动画 1）这使我们认识到，如何确定微指令的结构，乃是微程序设计的关键。

（动画 2）设计微指令结构时追求的五个目标依次是:（动画 3-7）。

（PPT543 第 2 页）要想设计微指令格式，首先看看如何设置微指令中的操作控制字段，（动画 1）即微命令编码。通常有以下三种方法。

1、直接表示法（动画 2-3）如图所示，其特点是操作控制字段中的每一位代表一

个微命令。（动画 4）优点是输出直接用于控制，需要发出某个微命令就将对应位置“1”，如上节课中的四条微指令。缺点是当微命令较多时，微指令字较长，从而增大控制存储器容量。（动画 5）如图所示的微指令格式，仅仅实现了一个简单运算器的功能（+，-，M），操作控制字段就需要 17 位。

（PPT543 第 3 页）（动画 1）2、编码表示法 就是把一组相斥性的微命令信号组成一个小组(即一个字段)，然后通过字段译码器对每一个微命令信号进行译码，译码输出作为操作控制信号，其微指令结构如图所示（动画 2）。

（动画 3）采用字段译码的方法，可用较少的二进制信息位表示较多的微命令信号。例如 3 位二进制位译码后可表示 7 个微命令（其中，去掉的 000 表示本字段不发出命令），4 位二进制位译码后可表示 15 个微命令。与直接表示相比，字段译码法可使微指令字大大缩短，但由于增加译码电路，使微程序的执行速度稍稍减慢。目前在微程序控制器设计中，字段直接译码法使用较普遍。

（PPT543 第 4 页）（动画 1）3、混合表示法 （动画 2）这种方法是把直接表示法与字段编码法混合使用，以便能综合考虑微指令字长、灵活性、执行微程序速度等方面的要求。

（动画 3）另外，在微指令中还可附设一个常数字段。该常数可作为操作数送入 ALU 运算，也可作为计数器初值用来控制微程序循环次数。

（PPT543 第 5 页）下面我们来看微指令格式中的顺序控制字段的设置方法，（动画 1）即微地址的形成方法。

微指令执行的顺序控制问题，实际上是如何确定下一条微指令的地址问题。（动画 2）通常，产生后继微地址有两种办法。

（动画 3）1)计数器方式 如同程序计数器 PC+1 产生下一条机器指令地址，在顺序执行微指令时，后继微地址可由现行微地址加上一个增量来产生，可将微地址寄存器改为计数器，顺序执行的微指令序列就必须安排在控制存储器的连续单元中。

（动画 4）2)多路转移方式 一条微指令具有多个转移分支的能力称为多路转移。（PPT543 第 6 页）例如，“取指”微指令根据操作码 OP 产生多路微程序分支而形成多个微地址。在多路转移方式中，当微程序不产生分支时，后继微地址直接由微指令的顺序控制字段给出；当微程序出现分支时，有若干“后选”微地址可供选择，即按“判别测试”标志和“状态条件”信息来选择其中一个微地址，其原理如图所示。如十进制加法指令中，（PPT543 第 7 页）根据 Cy 取值产生两路分支，用 Cy 控制微地址寄存器的

4 位地址中的最低位。Cy=1 时，微地址寄存器中的地址不变，仍为 0000；Cy=0 时，将微地址寄存器的最低位 0 修改为 1，后继微地址变为 0001，转去读取并执行第 4 条微指令。

（PPT543 第 8 页）（动画 1） 下面我们通过图示详细看看，“取指”微指令如何根据所取机器指令的操作码 OP 产生多路微程序分支进而形成多个微程序入微地址的。（动画 2）每条机器指令对应一段微程序。（动画 3）当公用的“取指” 微指令（或程序）从主存中取出机器指令之后，由机器指令的操作码字段指出各段微程序的入口地址。例如: 假定 ADD 指令的 OP 字段为 1000, 那么它在控存中的入口地址就为 100011; SUB 指令的 OP 字段若为 1001, 则它在控存中的入口地址就是 100111。由此可以看出，多路转移方式能以较短的顺序控制字段配合，实现多路并行转移，灵活性好，速度较快，但转移地址逻辑需要用组合逻辑方法设计。

（PPT543 第 9 页）（动画 1-5）下面我们通过【例 2】，来看看如何设计转移地址逻辑。

（PPT543 第 10 页）（动画 1）按所给设计条件，微程序有三个判别测试，分别为 P1，P2，P3。由于修改 μA5~μA0 内容具有很大灵活性，（动画 2）现分配如下：1）用 P1 和指令寄存器低 4 位修改微地址寄存器低 4 位；2）用 P2 和进位标志 C 修改微地址寄存器最低位；3）用 P3 和 IR5，IR4 修改 μA5，μA4。（动画 3）另外，还要考虑时间因素，一般在 CPU 周期最后一个节拍脉冲 T4，形成下一条微指令地址，读取该微指令。故转移逻辑表达式如下（动画 4）:

μA5=P3 · IR5 · T4　　μA4=P3 · IR4 · T4

μA3=P1 · IR3 · T4　　μA2=P1 · IR2 · T4

μA1=P1 · IR1 · T4　　μA 0=P1 · IR0 · T4+ P2 · C · T4

（动画 5）由于从微地址寄存器各位相关的触发器强置端修改，故前 5 个表达式可用“与非”门实现，最后一个用 “与或非”门实现。由此可画出微地址转移逻辑(作为作业同学们自己完成)。

（PPT543 第 11 页）（动画 1）接下来我们看看完整的微指令格式。

（动画 2）微指令的编译方法是决定微指令格式的主要因素。考虑到速度、成本等原因，在设计计算机时采用不同的编译法。因此，微指令的格式大体分成两类:水平型微指令和垂直型微指令。

（动画 3）一次能定义并执行多个并行操作微命令的微指令，叫做水平型微指令。

例如 5.4.1 节中所讲的微指令就是水平型微指令。水平型微指令的一般格式包括三个字段，操作控制字段、判别测试字段和下地址字段。按照操作控制字段的编码方法不同，水平型微指令又分为三种:一种是全水平型(直接表示法)微指令，第二种是字段译码法水平型微指令，第三种是直接和译码相混合的水平型微指令。

（动画 4）微指令中设置微操作码字段，采用微操作码编译法，由微操作码规定微指令的功能，称为垂直型微指令。垂直型微指令的结构类似于机器指令的结构。它有操作码，在一条微指令中只有 1~2 个微操作命令，每条微指令的功能简单，因此，实现一条机器指令的微程序要比水平型微指令编写的微程序长得多。它是采用较长的微程序结构去换取较短的微指令结构。

教材 P164 页给出了 4 条垂直型微指令的格式，请同学课后自学相关内容。

（PPT543 第 12 页）表格中给出了两种格式微指令的比较分析。我们可以看出，水平型微指令与机器指令差别很大，一般需要对机器的结构、数据通路、时序系统以及微命令很精通时才能设计。垂直型微指令的设计思想在 Pentium 4、安腾系列机中得到了应用。

（PPT543 第 13 页）（动画 1）最后，我们来看看动态微程序设计。微程序设计技术还有静态和动态之分。（动画 2）对应于一台计算机的机器指令只有一组微程序， 而且这一组微程序设计好之后，一般无需改变而且也不好改变，这种微程序设计技术称为静态微程序设计。本节前面讲述的内容基本上属于静态微程序设计的概念。（动画 3）采用 E^2PROM 作为控制存储器时，还可以通过改变微指令和微程序来改变机器的指令系统，该技术称为动态微程序设计。此时，微指令和微程序可根据需要加以改变，因而可在一台机器上实现不同类型的指令系统。这种技术又称为仿真，以便扩大机器的功能。

5.5 流水 CPU

5.5.1 并行处理技术

计算机有史以来，人们追求的目标之一就是很高的运算速度， 因此并行处理技术便成为计算机发展的主流。

从“运算方法和运算器”这一章中我们学到，为了提高浮点运算速度，浮点加法器采用流水线技术。从“存储系统”这一章，我们知道，双端口存储器设置两套独立的读\写电路，当左、右两个端口地址不相同的时候，可以同时进行读\写操作；还有，

低位交叉的多模块存储器，也可以流水访问各个模块，从而提高存取速度。

由此可见，计算机的并行处理技术可贯穿于信息加工的各个步骤和阶段。（PPT551 第 1 页）（动画 1）从广义上讲，并行性有着两种含义:一是同时性， 指两个以上事件在同一时刻发生。如双端口存储器左\右两个端口地址不同时，可同时读\写两个数据。二是并发性，指两个以上事件在同一时间间隔内发生。如“操作系统原理”课程中学到的，多道程序设计技术中，某个时刻 CPU 中只有一个进程在运行，而在一个时间段内，多个进程同时运行。

（动画 2）概括起来，并行性主要有三种形式:①时间并行； ②空间并行； ③时间并行+空间并行。

一、**时间并行**，是指时间重叠，就是让多个处理过程在时间上相互错开，轮流重叠地使用同一套硬件设备的各个部分，以加快硬件周转而赢得速度。其实现方式就是采用流水处理部件，这是一种非常经济而实用的并行技术，能够保证计算机系统具有较高的性价比。目前的高性能微型机几乎无一例外地都使用了流水线技术。

二、**空间并行**，指资源重复，在并行性概念中引入空间因素，以“数量取胜”为原则来大幅度提高计算机的处理速度。超大规模集成电路技术的迅速发展为空间并行技术带来了巨大生机，因而成为目前实现并行处理的一个主要途径。目前，空间并行技术主要体现在多处理器系统和多计算机系统。但在单机系统中也得到了广泛应用。

三、**时间并行+空间并行**，指时间重叠和资源重复的综合应用，既采用时间并行性又采用空间并行性。例如，我们即将学习的奔腾 CPU，采用了超标量指令流水线，在一个机器周期中以流水方式同时执行两条指令。

下面我们开始学习流水 CPU。

5.5.2 流水 CPU 的结构

我们知道，指令周期包括取指周期和执行周期，也就是说，执行一条指令可简单分为 “取指令”和“执行指令”两个阶段。（PPT552 第 1 页）（动画 1-4）因而，当处理机串行执行指令时，取指令和执行指令是周而复始地重复出现，各条指令按顺序串行执行。（动画 5）图中取指令操作由 “取指令部件”完成，（动画 6）执行指令的操作由“执行部件”完成，（动画 7）因而总有一个部件是空闲的。

（动画 8-11）如果指令执行阶段不访问主存，则完全可以利用这一段时间取下一条指令，这样就使取下一条指令的操作和执行当前指令的操作同时进行，这就是两条

指令的重叠，即指令的二级流水。

（动画 12）取指部件取出一条指令，并将它暂存起来，如果执行部件空闲，就将暂存指令传给执行部件执行。与此同时，取指部件又可取出下一条指令并暂存起来，这就是“指令预取”。（动画 13）显然，这种工作方式能加速指令的执行。如果取指和执行阶段在时间上完全重叠，相当于将指令周期减半，速度调高 1 倍。

（PPT552 第 2 页）然而，进一步分析指令的二级流水，我们就会发现，存在两个原因使得流水线执行效率加倍是不可能的。（动画 1）我们知道，功能不同的指令，执行周期所需时间也不同。通常，指令的执行时间一般大于取指时间，因此，（动画 2）取指部件取出的下一条指令不能立即传给执行部件，需要等待当前指令执行结束才能送入，因而，先送入取指部件缓冲器暂存。（动画 3）当遇到条件转移指令时，下一条指令是不可知的，（动画 4）因为必须等到当前指令执行结束后，才能知道条件是否成立，从而决定下一条指令的地址，因而造成时间损失。

解决上面两个问题的方法，我们将在后续课程中详细探讨，此处不再赘述。

（PPT552 第 3 页）（动画 1）要想以流水线方式执行指令，计算机系统将如何组成？下面，我们一起来学习流水 CPU 的组成。

（动画 1）图 5.30 给出了现代流水计算机的系统组成原理示意图。（动画 2）其中 CPU 按流水线方式组织，通常由三大部分组成:指令部件、指令队列、执行部件。这三个功能部件可以组成一个 3 级流水线。程序和数据存储在主存中，主存通常采用多体交叉存储器，以提高访问速度。高速缓冲存储器 cache， 用以弥补主存和 CPU 速度上的差异。

指令部件本身又构成一个流水线，即指令流水线，它由取指令、指令译码、计算操作数地址、取操作数等几个过程段组成。指令队列是一个先进先出(FIFO) 的寄存器栈，用于存放经过译码的指令和取来的操作数。它也是由若干个过程段组成的流水线。执行部件可以具有多个算术逻辑运算部件，这些部件本身又用流水线方式构成。由图可见，当执行部件正在执行第 I 条指令时，指令队列中存放着 I+1, I+2,…,I+k 条指令，而与此同时，指令部件正在取第 I+k+1 条指令。

为了使存储器的存取时间能与流水线的其他各过程段的速度相匹配，一般都采用多体交叉存储器。例如，IBM 360/91 计算机，根据一个机器周期输出一条指令的要求、存储器的存取周期、CPU 访问存储器的频率，采用了模 8 交叉存储器。在现有的流水线计算机中，存储器几乎都是采用交叉存取的方式工作。

（动画 3）执行段的速度匹配问题，通常采用并行的运算部件以及部件流水线的工作方式来解决。一般采用的方法包括:①将执行部件分为定点执行部件和浮点执行部件两个可并行执行的部分，分别处理定点运算指令和浮点运算指令；②在浮点执行部件中，又有浮点加法部件和浮点乘/除部件，它们也可以同时执行不同的指令；③浮点运算部件都以流水线方式工作。

计算机的流水处理过程非常类似于工厂中的流水装配线。为了实现流水，首先把输入的任务(或过程)分割为一系列子任务，并使各子任务能在流水线的各个阶段并发地执行。当任务连续不断地输入流水线时，在流水线的输出端便连续不断地吐出执行结果，从而实现了子任务级的并行性。

（PPT552 第 4 页）下面我们通过时空图来证明这个结论。（动画 1）图 5. 31(a)表示流水 CPU 中一个指令周期的任务分解。（动画 2-5）假设指令周期包含四个子过程:取指令(IF)、指令译码(ID)、执行运算(EX)、结果写回(WB)，每个子过程称为过程段(Si)，这样，一个流水线由一系列串联的过程段组成。各个过程段之间设有高速缓冲寄存器，以暂存上一过程段子任务处理的结果。在统一的时钟信号控制下，数据从一个过程段流向相邻的过程段。

（PPT552 第 5 页）图 5.31(b)表示非流水计算机的时空图。（动画 0）从图中可以看出，指令 I_1 经历 IF、ID、EX、WB 四个子过程全部执行完毕后才能开始下条指令 I_2。因此，每隔 4 个时钟周期才有一个输出结果。

（PPT552 第 6 页）图 5.31(c)表示标量流水计算机的时空图，（动画 0）对流水计算机来说，上条指令与下一条指令的四个子过程在时间上可以重叠执行。因此，当指令源源不断进入流水线，各过程段都满负荷工作时，每隔一个时钟周期就可以输出一个结果。（注意，标量是相对向量而言的。）

（PPT552 第 7 页）（动画 1）图 5.31(d)表示超标量流水计算机的时空图。（动画 2-3）所谓超标量流水，是指它具有两条以上的指令流水线。如图所示，当流水线满载时，每一个时钟周期可以执行 2 条指令。显然，超标量流水计算机是时间并行技术和空间并行技术的综合应用。Pentium 机就是一个超标量流水计算机。

（动画 4-7）直观比较后发现:标量流水计算机在 8 个时钟周期内执行完 5 条指令，超标量流水计算机在 8 个时钟周期内执行了 10 条指令，而非流水计算机在 8 个时钟周期内仅执行了 2 条指令。显然，流水技术的应用，使计算机的速度大大提高了。

一个计算机系统可以在不同的并行等级上采用流水线技术，请同学自学相关内容。

5.5.3 流水线中的主要问题

（PPT553 第 1 页）从上一节流水 CPU 的时空图对比分析中，我们可以看出，要使流水线具有良好的性能，必须使流水线畅通流动，不发生断流。但是，由于指令流水过程中会出现以下三种相关冲突，实现流水线的不断流是很困难的，（动画 1）这三种相关是资源相关、数据相关和控制相关。这一节我们主要学习引起三种相关的原因及其解决方法。

首先我们来看资源相关。（动画 2）所谓资源相关，是指多条指令进入流水线后在同一时钟周期内争用同一个功能部件所发生的冲突。（动画 3）假定一条指令流水线由五段组成，分别为取指令(IF)、指令译码(ID)并读寄存器堆、计算存储器操作数有效地址或执行(EX)、访存取操作数(MEM)、结果写寄存器堆(WB)。由表 5.2 看出，（动画 4）在时钟 4 时，指令 I_1 的 MEM 段与指令 I_4 的 IF 段都要访同存储器。当数据和指令放在同一个存储器且只有一个访问端口时，便发生两条指令争用存储器资源的相关冲突。（PPT553 第 2 页）解决资源冲突的办法，（动画 1-4）一是指令 I_4 停顿一拍后再启动，指令 I_5 紧随其后顺延。（动画 5）二是增设一个存储器，将指令和数据分别放在两个存储器中。

（PPT553 第 3 页）其次，我们来看数据相关。（动画 1）在一个程序中，如果必须等前一条指令执行完毕后，才能执行后一条指令，那么这两条指令就是数据相关的。在流水计算机中，指令执行是重叠进行的，前一条指令还没有结束，第二、三条指令就陆续地开始工作。由于多条指令的重叠执行，当后继指令所需要的操作数，刚好是前一指令的运算结果时，便发生了“先读后写”的数据相关冲突。例如下面三条指令（动画 2）：R2 和 R3 寄存器的内容相加之和存入 R1 寄存器；之后 R1 寄存器中的和，再减去 R5 寄存器的内容，将得到的差存入 R4 寄存器；R1 寄存器中的和，再跟 R7 寄存器的内容进行逻辑与运算，将结果存入 R6 寄存器。

经过分析发现，ADD 指令的运算结果正好是 SUB 指令和 AND 指令的一个源操作数。（动画 3）而如表 5.3 所示，ADD 指令在时钟 5 时将运算结果写入 R1 寄存器，但 SUB 指令在时钟 3 时读取寄存器 R1 的内容到 ALU 运算，AND 指令在时钟 4 时读取寄存器 R1 的内容到 ALU 运算。三条指令顺序执行时，ADD 指令应该先写 R1，SUB 指令后读 R1，但流水执行时，结果变成 SUB 指令先读 R1，ADD 指令后写 R1，因而发生了 SUB、 ADD 两条指令间先读后写的数据相关冲突；同样， AND、ADD 两条

指令间也发生了先读后写的数据相关冲突。

（PPT553 第 4 页）（动画 1）解决数据相关冲突的方法一:（动画 2）是推后后继指令对相关单元的读操作，直到当前指令 WB 段完成，（动画 3-4）流水线停顿 3 个时钟周期，即插入“气泡”，从而使得流水线断流，影响效率。

（PPT553 第 5 页）（动画 1-2）方法二是定向传送技术，又称为旁路技术，（动画 3）就是在流水 CPU 的运算器中，从 ALU 的输出端到其两个输入端之一，建立专用数据通路，将前一条指令的运算结果直接作为后继指令的一个源操作数，参与其相关运算。（动画 4）不需要经数据总线 DBUS→DR→R_i→ALU，去“长途跋涉”。也可以特意设置若干运算结果缓冲寄存器，暂时保留运算结果，以便于后继指令直接使用。

（PPT553 第 6 页）最后，我们来看控制相关。（动画 1）控制相关冲突是由转移指令引起的。当执行转移指令时，依据转移条件的产生结果，可能为顺序取下条指令;也可能转移到新的目标地址取指令，从而使流水线发生断流。为了减小转移指令对流水线性能的影响，常用以下两种转移处理技术:（动画 2）方法一为延迟转移法，由编译程序重排指令序列来实现。基本思想是“先执行再转移”，即发生转移取时并不排空指令流水线，而是让紧跟在转移指令 I_b 之后已进入流水线的少数几条指令继续完成。如果这些指令是与 I_b 结果无关的有用指令，那么延迟损失时间片正好得到了有效的利用。（动画 3）方法二是转移预测法，使用硬件方法来实现，依据条件转移指令过去的行为来预测将来的行为。通过使用转移取和顺序取两路指令预取队列以及目标指令 cache，可将转移预测提前到取指阶段进行，以获得良好的效果。关于控制相关更为详细的内容，我们将在后续课程“计算机系统结构”中探讨，此处不再赘述。

（PPT553 第 7 页）流水线中有三类数据相关冲突: 写后读(RAW)相关、读后写(WAR)相关、写后写(WAW)相关。

（PPT553 第 8 页）下面我们通过[例 4]，来分析判断以下三组指令各存在哪种类型的数据相关?

首先，分析第(1)组指令，I_1 指令运算结果应先写入 R_1，然后在 I_2 指令中读出 R_1 内容。由于 I_2 指令进入流水线，变成 I_2 指令在 I_1 指令写入 R_1 前就读出 R_1 内容，发生 RAW 相关。

其次，分析第(2)组指令，I_3 指令应先读出 R_3 内容并存入存储单元 M(x)，然后在 I_4 指令中将运算结果写 R_3。但如果 I_4 指令进入流水线，变成 I_4 指令在 I_3 指令读出 R_3 内容前就写入 R_3，发生 WAR 相关。

最后，分析第(3)组指令，如果 I_6 指令的加法运算完成时间早于 I_5 指令的乘法运算时间，变成指令 I_6 在指令 I_5 写入 R_3 前就写入 R_3,导致 R_3 的内容错误，发生 WAW 相关。

第6章　总线系统（2 学时）

6.1 总线的概念和结构形态

6.1.1 总线的基本概念

【ppt p1 动画 1，2，3】总线是构成计算机系统的互联机构，是多个系统功能部件之间进行数据传送的公共通路，并在争用资源的基础上进行工作。

1. 总线的分类

【ppt p2 动画 1】单机系统总线可分为三类:

【ppt p2 动画 2】 内部总线: CPU 内部连接各寄存器及运算器部件之间的总线。

【ppt p2 动画 3】 系统总线: 外部总线。CPU 和计算机系统中其他高速功能部件相互连接的总线。

【ppt p2 动画 4】 I/O 总线: 中低速 I/O 设备相互连接的总线。

2.总线的特性

总线的特性可分为:

【ppt p3 动画 1】物理特性: 指总线的物理连接方式，包括总线的根数，总线的插头、插座的形状，引脚线的排列方式等。

【ppt p3 动画 2】功能特性: 描述总线中每一根线的功能。

【ppt p3 动画 3】电气特性: 定义每一根线上信号的传递方向及有效电平范围。送入 CPU 的信号叫输入信号(IN)，从 CPU 发出的信号叫输出信号(OUT)。

【ppt p3 动画 4】时间特性: 定义了每根线在什么时间有效。规定了总线上各信号有效的时序关系，CPU 才能正确无误地使用。

3.总线的性能指标

【ppt p4 动画 1】总线带宽: 总线本身所能达到的最高传输速率。

【ppt p4 动画 2】总线宽度: 总线包含数据线的根数

【ppt p5】除了总线带宽和总线宽度，衡量总线性能的参数还包括:

标准传输率， 时钟同步/异步，总线复用，信号线数，总线控制方式，负载能力等，希望大家有所了解

总线带宽: 总线本身所能达到的最高传输速率。

影响总线带宽的主要因素有: 总线宽度、传送距离、总线发送和接收电路工作频

率限制以及数据传送形式。

【ppt p5 动画 1】影响总线带宽的主要因素有：总线宽度、传送距离、总线发送和接收电路工作频率限制以及数据传送形式。

【ppt p6】 **4. 总线的标准化**

【ppt p6 动画 1】为了使不同厂家生产的相同功能部件可以互换使用，就需要进行系统总线的标准化工作。目前，已经出现了很多总线标准，如 PCI、ISA、STD 等。

SA 总线(16 位，带宽 8MB/s)

EISA 总线(32 位，带宽 33.3MB/s)

VESA 总线(32 位，带宽 132MB/s)

PCI 总线(64 位，带宽 264MB/s) ***【ppt p5 动画 2】***

6.1.2 总线的连接方式

计算机的用途很大程度上取决于他所连接的外围设备的范围，而外围设备种类繁多，速度各异，所以，大量外设通过适配器连接大主机。***【ppt p1 动画 1】***计算机中总线的排列布置与其他各类不见的连接方式对计算机来说具有非常重要的作用。

【ppt p1 动画 2】根据连接方式不同，单机系统的总线结构分为两大类：单总线机构和多总线结构

【ppt p2】1.单总线结构： ***【ppt p2 动画 1，2，3】***使用一条单一的系统总线来连接 CPU、内存和 I/O 设备，相当于 CPU 引脚的延伸。

此时要求连接到总线上的逻辑部件必须高速运行，以便在某些设备需要使用总线时能迅速获得总线控制权；而当不再使用总线时，能迅速放弃总线控制权。

【ppt p2 动画 4】单总线系统负载较重，但是只要在系统总线上挂接多个 CPU，很容易扩展成多 CPU 系统。

【ppt p3】2. 多总线结构

现在，大多数的计算机采用分层次的多总线结构，在这种结构中速度差异较大的设备采用不同的总线，速度相同或相近的设备采用相同的总线。多总线结构分为双总线结构、三总线结构、四总线结构等

双总线：在 CPU 和主存之间专门设置了一组高速的存储总线。保持了单总线系统简单、易于扩充的优点，但又在 CPU 和主存之间专门设置了一组高速的存储总线，使

CPU 可通过专用总线与存储器交换信息，并减轻了系统总线的负担，这里有三种双总线结构。

【ppt p4】以存储器为中心的双总线结构

【ppt p5】面向 CPU 的双总线结构

【ppt p6】带有通道的双总线结构

当然这种双总线系统以增加硬件为代价。

【ppt p7】下面的三总线：它是在双总线系统的基础上增加 I/O 总线形成，【ppt p8】也可以增加局部总线形成的。

【ppt p9】也可以增加 DMA 总线等形成。

【ppt p10】多总线结构也可以在是在三总线结构的基础上，把 I/O 设备根据同速度不同分类管理连接，将 I/O 总线设计成高速总线和扩展总线从而形成四总线结构。

6.1.4 总线的内部结构

早期总线的内部按功能分类可分为：地址线（单向）、数据线（双向）和控制线（每一根是单向的）。

【ppt p1 动画 1】数据总线　双向　与机器字长、存储字长有关

【ppt p1 动画 2】地址总线　单向　与存储地址、 I/O 地址有关

【ppt p1 动画 3】控制总线　有出（存储器读、存储器写，总线允许、中断确认）；有入（中断请求、总线请求）

【ppt p1 动画 4】早期总线实际上就是 CPU 芯片引脚的延伸和驱动能力的增强，存在以下不足：

1） CPU 是总线上惟一的主控者。

2） 总线结构与 CPU 紧密相关，通用性较差。

【ppt p2 动画 1】当代总线的趋势是标准总线，与结构、CPU、技术无关，又被称为底板总线，能够满足包括多个 CPU 在内的主控者环境需求。

【ppt p2 动画 2】在当代总线结构中，CPU 和它私有的 cache 一起作为一个模块与总线相连。【ppt p2 动画 3】系统中允许有多个这样的处理器模块。而总线控制器完成几个总线请求者之间的协调与仲裁。

【ppt p2 动画 4】当代总线可分为四个部分：

- 数据传送总线：由地址线、数据线、控制线组成。

- 仲裁总线：包括总线请求线和总线授权线。
- 中断和同步总线：包括中断请求线和中断认可线。
- 公用线：时钟信号、电源等。

6.2 总线接口

6.2.1 信息的传送方式

【ppt p1 动画 1】数字计算机使用二进制数，它们或用电位的高、低来表示，或用脉冲的有、无来表示

【ppt p1 动画 2】计算机系统中，传输信息基本有三种方式：串行传送、并行传送、和分时传送。但是出于速度和效率上的考虑，系统总线上传送的信息必须采用并行传送方式。

分时传送即总线的分时复用。

串行传送：使用一条传输线，采用脉冲传送。主要优点是只需要一条传输线，这一点对长距离传输显得特别重要，不管传送的数据量有多少，只需要一条传输线，成本比较低廉。缺点就是速度慢。

并行传送：每一数据位需要一条传输线，一般采用电位传送。

分时传送：总线复用或是共享总线的部件分时使用总线。

1.串行传送

【ppt p2 动画 1】当信息以串行方式传送时，只有一条传输线，且采用脉冲传送。***【ppt p2 动画 2】***当在串行传送时，按顺序来传送表示一个数码的所有二进制位(bit)的脉冲信号，***【ppt p2 动画 3】***当每次一位，通常以第一个脉冲信号表示数码的最低有效位，最后一个脉冲信号表示数码的最高有效位。

【ppt p2 动画 4】在串行传送时，被传送的数据需要在发送部件进行并－－串变换，这称为拆卸；

【ppt p2 动画 5】在接收部件又需要进行串－－并变换，这称为装配。

串行传送的主要优点是只需要一条传输线，这一点对长距离传输显得特别重要，不管传送的数据量有多少，只需要一条传输线，成本比较低廉。缺点是速度慢。

2.并行传送

【ppt p3 动画 1】 用并行方式传送二进制信息时，对每个数据位都需要单独一条传输线，并行传送一般采用电位传送。***【ppt p3 动画 2】***信息有多少二进制位组成，就

需要多少条传输线，【ppt p3 动画 3】从而使得二进制数“0”或“1”在不同的线上同时进行传送。

【ppt p3 动画 4】由于所有的位同时被传送，所以并行数据传送比串行数据传送快得多。系统总线上传送的信息都采用并行传送方式。

【ppt p4】3.分时传送

分时传送有两种概念。

一是采用总线复用方式，某个传输线上既传送地址信息，又传送数据信息。为此必须划分时间片，以便在不同的时间间隔中完成传送地址和传送数据的任务。

分时传送的另一种概念是共享总线的部件分时使用总线。

6.2.2 接口的基本概念

【ppt p1】外设不能直接连接 CPU,必须通过接口设备。接口的典型功能：控制、缓冲、状态、转换、整理、程序中断。

一个适配器的两个接口：一个同系统总线相连，采用并行方式，另外一个同设备相连，可能采用并行方式或是串行方式。

【ppt p1 动画 1】接口即 I/O 设备适配器，具体指 CPU 和主存、外围设备之间通过总线进行连接的逻辑部件。

【ppt p1 动画 2】接口部件在它动态连接的两个部件之间起着“转换器”的作用，以便实现彼此之间的信息传送。

【ppt p2】CPU、接口和外围设备之间的连接关系

为了使所有的外围设备能够兼容，并能在一起正确地工作，CPU 规定了不同的信息传送控制方法。 一个标准接口可能连接一个设备，也可能连接多个设备。

【ppt p2 动画 1】典型的接口通常具有如下功能:

【ppt p2 动画 2】1.控制:接口靠程序的指令信息来控制外围设备的动作，如启动、关闭设备等。

【ppt p2 动画 3】2.缓冲:接口在外围设备和计算机系统其他部件之间用作为一个缓冲器，以补偿各种设备在速度上的差异。

【ppt p2 动画 4】3.状态:接口监视外围设备的工作状态并保存状态信息。状态信息包括数据“准备就绪”、“忙”、“错误”等等，供 CPU 询问外围设备时进行分析之用。

【ppt p2 动画 5】4.转换:接口可以完成任何要求的数据转换，例如并－－串转换或串－－并转换，因此数据能在外围设备和 CPU 之间正确地进行传送。

【ppt p2 动画 6】5.整理:接口可以完成一些特别的功能，例如在需要时可以修改字计数器或当前内存地址寄存器。

【ppt p2 动画 7】6.程序中断:每当外围设备向 CPU 请求某种动作时，接口即发生一个中断请求信号到 CPU。

【ppt p3 动画 1】事实上，一个适配器必有两个接口:

【ppt p3 动画 2，3】一是和系统总线的接口，CPU 和适配器的数据交换一定的是并行方式;

【ppt p3 动画 4，5】二是和外设的接口，适配器和外设的数据交换可能是并行方式，也可能是串行方式。根据外围设备供求串行数据或并行数据的方式不同，适配器分为串行数据接口和并行数据接口两大类。

6.3 总线的集中式仲裁

【ppt p1 动画 1】*仲裁的基本概念*

【ppt p1 动画 2】总线仲裁是总线系统的核心问题之一。

【ppt p1 动画 3】为了解决多个主设备竞争总线控制权的问题，必须具有总线仲裁部件。它通过采用优先级策略或公平策略进行仲裁，选择其中一个主设备作为总线的下一次主方，接管总线控制权。

【ppt p1 动画 4】主设备(模块) 对总线有 控制权

【ppt p1 动画 5】从设备(模块) 响应 从主设备发来的总线命令

【ppt p2 动画 1】总线占用期: 主方持续控制总线的时间。

连接到总线上的功能模块有主动和被动两种形态，其中主方可以启动一个总线周期，而从方只能响应主方请求。每次总线操作，只能有一个主方，但是可以有多个从方。

为了解决多个功能模块争用总线的问题，必须设置总线仲裁部件。

【ppt p2 动画 2】按照总线仲裁电路的位置不同，仲裁方式分为【ppt p2 动画 3】集中式和【ppt p2 动画 4 分布式两种。

【ppt p3】集中式仲裁的控制逻辑基本集中在一处,需要中央仲裁器在单机系统中，中央仲裁器即为总线控制器。

【ppt p3 动画 1】集中式仲裁：链式查询方式

BS -总线忙 BR-总线请求 BG-总线同意 各信号线上 1 有信号 0 有信号

【ppt p3 动画 2】从图上可以看到任何主方均可通过 BR 申请总线使用权，中央仲裁器收到 1 个或多个 BR 时，产生总线同意 BG 信号进行总线授权，**【ppt p3 动画 3】**BG 串行地从一个 I/O 接口传送到下一个 I/O 接口。假如 BG 到达的接口无总线请求，则继续往下查询；假如 BG 到达的接口有总线请求，便不再往下查询，**【ppt p3 动画 4】**置“1”BS 线，获得了总线使用权，启动总线周期。

【ppt p3 动画 5】特点：离中央仲裁器最近的设备具有最高优先级，“近水楼台先得月”，通过接口的优先级排队电路来实现。

优点：只用很少几根线就能按一定优先次序实现总线控制，并且这种链式结构很容易扩充设备。

缺点：是对询问链的电路故障很敏感，优先级固定。

【ppt p4 】 集中式仲裁：计数器定时查询方式

【ppt p4 动画 1，2】可方便的改变优先级。

【ppt p4 动画 3】总线上的任一设备要求使用总线时，通过 BR 线发出总线请求。中央仲裁器接到请求信号以后，在 BS 线为“0”的情况下让计数器开始计数，**【ppt p4 动画 4，5，6，7】**计数值通过一组地址线发向各设备。每个设备接口都有一个设备地址编号和地址判别电路，将地址线上的计数值与自己的设备地址比较，**【ppt p4 动画 8】**结果相同时，该设备 置“1”BS 线，获得了总线使用权，此时中止计数查询。

【ppt p4 动画 9】每次计数可以从“0”开始，也可以从中止点开始。

如果从“0”开始，各设备的优先次序与链式查询法相同，优先级的顺序是固定的。

如果从中止点开始，则每个设备使用总线的优先级相等。

计数器的初值也可用程序来设置，可以采用加计数也可采用加计数，这可以方便地改变优先次序，但这种灵活性是以增加线数为代价的。

计数器的初值也可用程序来设置，这可以方便地改变优先次序，但这种灵活性是以增加线数为代价的。

总线上的任一设备要求使用总线时，通过 BR 线发出总线请求。

【ppt p5 】集中式仲裁：独立请求方式

每一个共享总线的设备均有一对总线请求线 BRi 和总线授权线 BGi。

【ppt p5 动画 2】当设备要求使用总线时，便发出该设备的请求信号。

【ppt p5 动画 3】中央仲裁器中的排队电路决定首先响应哪个设备的请求，【ppt p5 动画 4】给设备以授权信号 BGi。

【ppt p5 动画 5】独立请求方式的优点：响应时间快，确定优先响应的设备所花费的时间少，用不着一个设备接一个设备地查询。

其次，对优先次序的控制相当灵活，可以预先固定也可以通过程序来改变优先次序；还可以用屏蔽(禁止)某个请求的办法，不响应来自无效设备的请求。

独立请求方式：优点是响应时间快，即确定优先响应的设备所花费的时间少。对优先次序的控制也是相当灵活的。

6.4 总线的定时和数据传送模式

6.4.1 总线的定时

【ppt p1 动画 1】为了同步主方、从方的操作，必须制订定时协议。

【ppt p1 动画 2】定时：事件出现在总线上的时序关系。

【ppt p1 动画 3】通常采用同步定时与异步定时两种方式。

【ppt p1 动画 4】同步定时：事件在总线上的时刻由总线时钟信号来确定，传输频率较高，适用于各功能模块速度相差不多的情况。

【ppt p1 动画 5】异步定时：应答方式或互锁机制。总线周期长度可变，适用于快速、慢速设备连接到同一总线。

【ppt p2 动画 1】总线的一次信息传送过程，大致可分为如下五个阶段：

请求总线，总线仲裁，寻址(目的地址)，信息传送，状态返回(或错误报告)

1.同步定时

【ppt p3 动画 1】在同步定时协议中，事件出现在总线上的时刻由总线时钟信号来确定。由于采用了公共时钟，每个功能模块什么时候发送或接收信息都由统一时钟规定，因此，同步定时具有较高的传输频率。

【ppt p3 动画 2】在同步定时适用于总线长度较短、各功能模块存取时间比较接近的情况。

这是一个读数的同步时序图。

【ppt p4 动画 1，2，3，4，5】发送方用系统时钟前沿发信号，接收方用系统时钟后沿判断和识别信号，大多事事件值占用一个时钟周期。

【ppt p4 动画 6，7，8】T1: CPU 送出地址信息也可以发一个启动信号，指示控制信号地址信息一出现在总线上;

【ppt p4 动画 9，10】T2: CPU 发出读信号，并延迟一个时钟周期用于存储模块地址译码;

【ppt p4 动画 11，12，13】T3:存储器将数据放到数据总线上，稳定并并保持一定时间，保证 CPU 正确读取;

【ppt p4 动画 14，15】依次撤销信号

2.异步定时

【ppt p5 动画 1】在异步定时协议中，后一事件出现在总线上的时刻取决于前一事件的出现，即建立在应答式或互锁机制基础上。在这种系统中，不需要统一的共公时钟信号。总线周期的长度是可变的。

在异步定时方式中，没有统一时钟，也没有固定时间间隔，完全依靠传送双方互相制约的握手信号来实现定时控制。

【ppt p5 动画 2】主设备提出交换信息的“请求”信号，经过接口传送到从设备，从设备接到主设备的请求后，通过接口向主设备发出 “回答”信号。

【ppt p5 动画 3】根据请求和回答信号的撤销是否互锁，异步定时分为三种类型:不互锁方式、半互锁方式、全互锁方式

【ppt p6 动画 1，2，3】1.不互锁方式：速度最快，可靠性最差

【ppt p6 动画 4，5】主设备发出请求信号后，不必等到从设备的回答信号，而是经过一段时间间隔，便撤销请求信号;

【ppt p6 动画 3】而从设备接到请求信号后，发出回答信号，并经过一段时间，自动撤销回答信号。双方不存在互锁关系。

【ppt p6 动画 6】**2.半互锁方式**

【ppt p6 动画 7，8，9】主设备发出请求信号后，必等接到从设备的回答信号后，才能撤销请求信号，有互锁关系;

而从设备接到请求信号后，发出回答信号，但不必等到获知主设备的请求信号已经撤销，而是经过一段时间后自动撤销回答信号，不存在互锁关系。

【ppt p6 动画 10】全互锁方式:

【ppt p6 动画 11，12，13，14】主设备发出请求信号后，必等接到从设备的回答信号后，才能撤销请求信号，有互锁关系;

而从设备接到请求信号后，发出回答信号，必须等到获知主设备的请求信号已经撤销后，才能撤销回答信号，存在互锁关系。

即双方存在互锁关系。

异步定时的优点是总线周期长度可变，不把响应时间强加到功能模块上，因而允许快速和慢速的功能模块都能连接到同一总线上。

但这以增加总线的复杂性和成本为代价。

6.4.2 总线数据传送模式

【ppt p1 动画 1】四类数据传送模式:

读、写操作，块传送操作，写后读、读修改写操作，广播、广集操作。

【ppt p2 动画 1】1.读、写操作

读操作是由从方到主方的数据传送；写操作是由主方到从方的数据传送。

一般，主方先以一个总线周期发出命令和从方地址，经过一定的延时再开始数据传送总线周期。为了提高总线利用率，减少延时损失，主方完成寻址总线周期后可让出总线控制权，以使其他主方完成更紧迫的操作。然后再重新竞争总线，完成数据传送总线周期。

【ppt p2 动画 2】2.块传送操作

只需给出块的起始地址，然后对固定块长度的数据一个接一个地读出或写入。

【ppt p2 动画 3】对于 CPU(主方)、存储器(从方)而言的块传送，常称为猝发式传送，其块长一般固定为数据线宽度(存储器字长)的 4 倍。

【ppt p3 动画 1】3.写后读、读修改写操作

只给出地址一次，或进行先写后读操作，或进行先读后写操作。

【ppt p3 动画 2】先写后读用于校验目的，***【ppt p3 动画 3】***先读后写用于多道程序系统中对共享存储资源的保护。这两种操作和猝发式操作一样，主方掌管总线直到整个操作完成。

【ppt p3 动画 4.5】4.广播、广集操作

一般而言，数据传送只在一个主方和一个从方之间进行。***【ppt p3 动画 6】***但有的总线允许一个主方对多个从方进行写操作，这种操作称为广播。***【ppt p3 动画 7】***与广播相反的操作称为广集，它将选定的多个从方数据在总线上完成 AND 或 OR 操作，用以检测多个中断源。

6.5 一种多总线结构

这是某 PC 机的主板总线框图，是一种典型的多总线结构。包括 HOST 总线，两组 PCI 总线，LAGACY 总线。板上有三种桥 HOST 桥，PCI/LAGACY 总线桥，PCI/PCI 桥。

其中：

【ppt p1 动画 1】HOST 总线：宿主总线，也称 CPU 总线、系统总线、主存总线等，各种叫法反映总线功能的一个方面。 “宿主”总线，也许更全面，因为 HOST 总线不仅连接主存，还可以连接多个 CPU。HOST 总线用于连接 CPU、Cache 和主存。

【ppt p1 动画 2】PCI 总线：连接各种高速的 PCI 设备。PCI 设备可以是主设备，也可以是从设备，或兼而有之。用 DMA 方式工作的设备可以移植到 PCI 总线上，采用主设备工作方式就可以。系统中允许有多条 PCI 总线，上边的 PCI 总线使用 HOST 桥与 HOST 总线相连，下边的 PCI 总线使用 PCI/PCI 桥与上边的 PCI 总线相连，从而得以扩充整个系统的 PCI 总线负载能力。PCI 总线用于连接高速的外围设备。

【ppt p1 动画 3】LAGACY 总线：可以是 ISA，EISA，MCA 等这类性能较低的传统总线，以便充分利用市场上丰富的适配器卡，支持中、低速 I/O 设备，用于连接中、低速设备。

【ppt p2 动画 1】在 PCI 总线体系结构中的三种桥。

【ppt p2 动画 2】桥连接两条总线，使彼此间相互通信。

桥又是一个总线转换部件，可以把一条总线的地址空间映射到另一条总线的地址空间上，从而使系统中任意一个总线主设备都能看到同样的一份地址表。

PCI 总线的基本传输机制是猝发式传送，利用桥可以实现总线间的猝发式传送。写操作时，桥把上层总线的写周期先缓存起来，以后的时间再在下层总线上生成写周期， 即延迟写。读操作时，桥可早于上层总线，直接在下层总线上进行预读。无论延迟写和预读，桥的作用可使所有的存取都按 CPU 的需要出现在总线上。

由上可见，以桥连接实现的 PCI 总线结构具有很好的扩充性和兼容性，允许多条总线并行工作。它与处理器无关，不论 HOST 总线上是单 CPU 还是多 CPU，也不论 CPU 是什么型号，只要有相应的 HOST 桥芯片(组)，就可与 PCI 总线相连。即多层总线的概念。各层之间使用桥进行连接。

第7章　外设与输入/输出系统（3 学时）

通过前几章内容的学习，我们知道：除了处理器和存储器，计算机系统的第三类关键部件是 I/O 逻辑模块。一个计算机系统的综合处理能力，系统的可扩展性、兼容性和性能价格比，都和 I/O 系统有着密切关系。因此，本章首先讲授信息交换方式，然后简要介绍两种 I/O 标准接口：SCSI 和 IEEE1394。

7.1 外设概述及其信息交换方式

7.1.1 外围设备的速度分级

（PPT71 第 1 页）（动画 1-4）同学们，我们知道：外设的种类相当繁多，有机械式和电动式，也有电子式和其他形式。比如鼠标，就有早期的机械式、光电式，又有现在的蓝牙鼠标、红外鼠标等等。这些设备的输入信号，可以是数字式的电压，也可以是模拟式的电压和电流。信息传输速率相差也很悬殊。例如，当手动键盘输入时，每个字符输入的间隔可达数秒钟。又如磁盘输入时，找到对应磁道后，磁盘能以大于 30 000B/s 的速率输入数据。

（动画 5-6）那么，把高速工作的 CPU 和不同速度工作的外设相连接，首先遇到的一个问题，就是如何保证处理机与外设在时间上同步？这就是我们要讨论的外设的定时问题。

（PPT71 第 2 页）（动画 1）对于不同速度的外设，需要有不同的定时方式，总的说来，CPU 与外设之间的定时，有以下三种情况。

（动画 2）1）速度极慢或简单的外设 对这类设备，如机械开关、显示二极管等等，CPU 总是能足够快地作出响应。所以，在这种情况下，CPU 只要接收或发送数据就可以了。

（动画 3）2）慢速或中速的外设 由于这类设备的速度和 CPU 的速度并不在一个数量级，或者由于设备（如键盘)本身是在不规则时间间隔下操作的，因此，CPU 与这类设备之间的数据交换通常采用异步定时方式。

（动画 4）3）高速的外设 由于这类外设，如磁盘，是以相等的时间间隔操作的，而 CPU 也是以等同隔的速率执行输入/输出指令的，因此，这种方式叫做同步定时方式。一旦 CPU 和外设发生同步，它们之间的数据交换便靠时钟脉冲控制来进行。

7.1.2 信息交换方式

（PPT71 第 3 页）（动画 1-2）前期课程《操作系统原理》中，我们已经了解到 CPU 对外设的四种常用控制方式：程序查询、程序中断、直接存储器访问（DMA）和通道，这就是 CPU 与外设间的四种信息交换方式。

教材上通过日常生活中幼儿园阿姨给小朋友分糖的例子，类比这几种交换方式。我们可将其上升到“管理学”上的“例外原则”。美国管理学家泰勒认为：“为了提高效率和控制大局，上级只保留处理例外和非常规时间的决定权和控制权，例行和常规的权利由部下分享”。在计算机系统发展过程中，CPU 与外设之间信息交换方式的发展更新，正是体现了泰勒提出的这种“权利下放”的例外原则。

这里，我们简要回顾一下这四种方式的工作原理：

首先，我们看看程序查询方式，它是一种最简单的输入输出方式，数据在 CPU 和外设之间的传送完全靠计算机程序控制。该方式中，CPU 采用定期地由主程序转向查询设备状态的子程序进行扫描轮询，看外设是否需要进行信息交换。它的优点是 CPU 的操作和外设的操作能够同步，而且硬件结构比较简单。但问题是，外设动作很慢，程序进入查询循环时将白白浪费掉 CPU 很多时间。因此除单片机和数字信号处理机 DSP 外，大型机中不使用程序查询方式。

其次是程序中断方式。中断是外设用来“主动”通知 CPU，准备送出输入数据或接收输出数据的一种方法。通常，当一个中断发生时，CPU 暂停它的现行程序，而转向中断处理程序，从而可以输入或输出一个数据。当中断处理完毕后，CPU 又返回到它原来的任务，并从它停止的地方开始执行程序。这种方式节省了 CPU 宝贵的时间，是管理 I/O 操作的一个比较有效的方法，一般适用于随机出现的服务，并且一旦提出要求，应立即进行。同程序查询方式相比，硬件结构相对复杂些，服务开销时间较大。

第三种是 DMA 方式。我们知道，用中断方式交换数据时，每处理一次 I/O 交换，大约需要几十微秒到几百微秒。对于一些高速的外设，以及成组交换数据的情况，仍然显得速度太慢。直接内存访问方式是一种完全由硬件执行 I/O 交换的工作方式。这种方式既考虑到中断响应，同时又要节约中断开销。此时，DMA 控制器从 CPU 完全接管对总线的控制，数据交换不经过 CPU，而直接在内存和外设之间进行，以高速传送数据。这种方式中，CPU 将低速的 I/O 操作交给 DMA 控制器，体现了泰勒提出的例外原则中的“权力下放”。主要优点是数据传送速度很高，传送速率仅受到内存访问

时间的限制。与中断方式相比，它需要更多的硬件。DMA 方式适用于内存和高速外设之间大批数据交换的场合。

最后是通道方式。DMA 方式的出现已经减轻了 CPU 对 I/O 操作的控制，使得 CPU 的效率有了显著提高，而通道的出现则进一步提高了 CPU 的效率。这是因为，CPU 将“数据传输”的权力下放给通道，它只专注于高速的“数据处理”。通道是一个具有特殊功能的处理器，它可以实现对外设的统一管理和外设与内存之间的数据传送。这种方式大大提高了 CPU 的工作效率。然而这种提高 CPU 效率的办法是以花费更多硬件为代价的。

程序查询方式和程序中断方式适用于数据传输率比较低的外设，而 DMA 方式、通道方式适用于数据传输率比较高的设备。目前，程序中断方式和 DMA 方式多用于微型机中。通道方式用在大型计算机中。从下一节起，我们以信息交换方式为纲，介绍这四种典型的输入/输出系统结构、工作原理及操作过程。

7.2 程序中断方式

程序查询方式和程序中断方式，我们在《汇编语言与微机原理》、《操作系统原理》课程中已学过，因此，请同学们自己复习程序查询方式相关知识，本节我们主要回顾并学习程序中断方式。

7.2.1 中断的基本概念

（PPT721 第 1 页）（动画 1）中断概念的出现，是计算机系统结构设计中的一个重大变革。（动画 2）中断系统是计算机实现中断功能的软、硬件总称。（动画 3）一般在 CPU 中设置中断机构，（动画 4）在外设接口中设置中断控制器，（动画 5）在软件上设置相应的中断服务程序。

（PPT721 第 2 页）（动画 1）上一节曾经提到，在程序中断方式中，某一外设的数据准备就绪后，它“主动”向 CPU 发出请求中断的信号，请求 CPU 暂时中断目前正在执行的程序而进行数据交换。当 CPU 响应这个中断时，便暂停运行主程序，并自动转移到该设备的中断服务程序。当中断服务程序结束以后，CPU 又回到原来的主程序。

这种原理和子程序调用相仿，不过，这里要求转移到中断服务程序的请求是由外设发出的。中断方式特别适合于随机出现的服务。

（动画 2）下图给出了中断处理示意图。主程序只是在设备 A，B，C 数据准备就绪时，才去处理 A，B，C，进行数据交换。在速度较慢的外设准备自己的数据时，

CPU 照常执行自己的主程序。在这个意义上说，CPU 和外设的一些操作是并行地进行的，因而同串行进行的程序查询方式相比，计算机系统的效率大大提高了。

（PPT721 第 3 页）（动画 1）实际的中断过程还要复杂些，图中给出了中断处理过程的详细流程图。（动画 2）当 CPU 执行完一条现行指令时，如果外设向 CPU 发出中断请求，那么 CPU 在满足响应条件的情况下，将发出中断响应信号，与此同时关闭中断（“中断屏蔽触发器”置“1”），表示 CPU 不再受理另外一个设备的中断请求。这时，CPU 将寻找中断源是哪一个设备，并保存 CPU 自己的程序计数器 PC 的内容。然后，它将转移到处理该中断源的中断服务程序。CPU 在保存现场信息，为设备服务（如交换数据）后，将恢复现场信息。在这些动作完成以后，开放中断(“中断屏蔽触发器”置“0”)，并返回到原来被中断的主程序的下一条指令。

（PPT721 第 4 页）（动画 1）以上是中断处理的大致过程，但是有一些问题需要进一步加以说明。

（动画 2）第一个问题，尽管外界中断请求是随机的，但 CPU 只有在当前一条指令执行完毕后，即转入公操作时才受理设备的中断请求，这样才不至于使当前指令的执行受到干扰。外界中断请求信号通常存放在外设接口中的中断源锁存器里，并通过中断请求线连至 CPU，每当一条指令执行到末尾，CPU 便检查中断请求信号。若中断请求信号为“1”，则 CPU 转入“中断周期”，受理外界中断。

（动画 3）第二个问题，为了在中断服务程序执行完毕后，能正确返回到主程序被中断的断点而继续执行，必须把程序计数器 PC 的内容，以及当前指令执行结束后 CPU 的状态(包括寄存器的内容和一些状态标志位)都保存到堆栈中去。这些操作叫做保存现场。

（动画 4）第三个问题，当 CPU 响应中断后，正要去执行中断服务程序时，可能有另一个新的中断源向它发出中断请求。为了不致造成混乱，在 CPU 的中断管理部件中必须有一个“中断屏蔽”触发器，它可在程序的控制下置“1”或置“0”。只有在“中断屏蔽”标志为“0”时，CPU 才可以受理中断。当一条指令执行完毕 CPU 接受中断请求并作出响应时，它一方面发出中断响应信号 INTA，另一方面把“中断屏蔽”标志置“1”，即关闭中断。这样，CPU 不能再受理另外的新的中断源发来的请求。只有在 CPU 把中断服务程序执行完毕后，它才重新使“中断屏蔽”标志置“0”，即开放中断，并返回主程序。因此，中断服务程序的最后必须有两条指令，即开中断指令和返主（或回）指令，同时在硬件上要保证返主指令执行以后，才受理新的中断请求。

（动画 5）第四个问题，中断处理过程是由硬件和软件结合来完成的。从中断处理流程图可以看出，“中断周期”由硬件实现，而中断服务程序由机器指令序列实现。后者除执行保存现场、恢复现场、开放中断并返回主程序任务外，对要求中断的设备进行服务，使其同 CPU 交换一个字的数据，或作其他服务。至于在中断周期中如何转移到各个设备的中断服务程序，将留在后面介绍。

（动画 6）第五个问题，中断分为内中断和外中断。机器内部原因导致出错引起的中断叫内中断，也叫异常，如除数为零。外部设备请求服务的中断叫外中断。

7.2.2 程序中断方式的基本 I/O 接口

（PPT722 第 1 页）（动画 1）下图为程序中断方式的基本接口示意图。接口电路中有一个（动画 2）工作标志触发器 BS，（动画 3）就绪标志触发器 RD，还有一个控制触发器，（动画 4）它叫允许中断触发器(EI)。

程序中断由外设接口的状态和 CPU 两方面来控制。

（动画 5）在接口方面，有决定是否向 CPU 发出中断请求的机构，主要是接口中的“准备就绪”标志 (RD)和“允许中断”标志(EI)两个触发器。

（动画 6）在 CPU 方面，有决定是否受理中断请求的机构，主要是“中断请求”标志(IR)和“中断屏蔽”标志(IM)两个触发器。

上述四个标志触发器的具体功能如下:

准备就绪的标志(RD) ： 一旦设备做好一次数据的接收或发送，便发出一个设备动作完毕信号，使 RD 标志置“1”。 在中断方式中，该标志用作中断源触发器，简称中断触发器。

允许中断触发器(EI) ： 可以用程序指令来置位。 EI 为“1”时，某设备可以向 CPU 发出中断请求；EI 为“0”时，不能向 CPU 发出中断请求，这意味着某中断源的中断请求被禁止。设置 EI 标志的目的，就是通过软件来控制是否允许某设备发出中断请求。

中断请求触发器(IR): 它暂存中断请求线上由设备发出的中断请求信号。当 IR 标志为“1”时，表示设备发出了中断请求。

中断屏蔽触发器(IM): 是 CPU 是否受理中断或批准中断的标志。IM 标志为“0”时，CPU 可以受理外界的中断请求，反之，IM 标志为“1”时，CPU 不受理外界的中断。

图中需要注意的是：CPU 中有两个 IR，（动画 7）上面的是中断请求触发器 IR，与 IM 配合使用；（动画 8）下面的是指令寄存器 IR，与程序计数器 PC 配合使用。

（PPT722 第 2 页）（动画 1）图中，标号①~⑧表示由某一外设输入数据的控制过程。

（动画 2）①表示由程序启动外设，将该外设接口的“忙”标志 BS 置“1”，“准备就绪”RD 清“0”；（动画 3）②表示接口向外设发出启动信号；（动画 4）③表示数据由外设传送到接口的缓冲寄存器；（动画 5）④表示当设备动作结束或缓冲寄存器数据填满时，设备向接口送出一个控制信号，将数据“准备就绪”标志 RD 置“1”；（动画 6）⑤表示允许中断标志 EI 为“1” 时，接口向 CPU 发出中断请求信号；（动画 7）⑥表示在一条指令执行末尾 CPU 检查中断请求线，将中断请求线的请求信号接收到“中断请求”标志 IR；（动画 8）⑦表示如果“中断屏蔽”标志 IM 为“0”时，CPU 在一条指令执行结束后受理外设的中断请求，向外设发出中断响应信号并关闭中断；（动画 9）⑧表示转向该设备的中断服务程序入口；（动画 10）⑨表示在中断服务程序中通过输入指令把接口中数据缓冲寄存器的数据读至 CPU 中的寄存器；（动画 11）⑩表示 CPU 发出控制命令 C 将接口中的 BS 和 RD 标志复位。

7.2.3 单级中断

根据计算机系统对中断处理的策略不同，可分为单级中断系统和多级中断系统。单级中断系统是中断结构中最基本的形式。本节我们首先学习单级中断，详细讨论如何识别单级中断源？中断向量如何产生？

（PPT723 第 1 页）（动画 1）我们首先来看单级中断的概念。（动画 2）在单级中断系统中，所有的中断源都属于同一级。（动画 3）图中给出了单级中断示意图(a)和单级中断系统结构图(b)。从中我们可以看出：

所有中断源触发器排成一行，其优先次序是离 CPU 近的优先权高。

当响应某一中断请求时，执行该中断源的中断服务程序。在此过程中，不允许其他中断源再打断中断服务程序，即使优先权比它高的中断源也不能再打断。只有该中断服务程序执行完毕后，才能响应其他中断。

所有的 I/O 设备通过一条线向 CPU 发出中断请求信号。CPU 响应中断请求后，发出中断响应信号 INTA，以链式查询方式识别中断源。这种中断结构与第六章讲的链式总线仲裁相对应，中断请求信号 IR 相当于总线请求信号 BR。

（PPT723 第 2 页）（动画 1）其次，我们来学习单级中断源的识别。

如何确定中断源，并转入被响应的中断服务程序入口地址，是中断处理首先要解决的问题。

（动画 2）在单级中断中，采用串行排队链路来实现具有公共请求线的中断源判优识别。（动画 3）其逻辑电路如图所示。

（动画 4）图中下面的虚线部分是一个串行的优先链，称作中断优先级排队链。（动画 5）IR_i 是从各中断源设备来的中断请求信号，优先顺序从高到低是 IR_1，IR_2，IR_3。（动画 6）而 IS_1，IS_2，IS_3 是与 IR_1，IR_2，IR_3 相对应的中断排队选中信号，若 IS_i=1，即表示该中断源被选中。（动画 7）INTI 为中断排队输入，（动画 8）INTO 为中断排队输出。

（PPT723 第 3 页）（动画 1-2）若没有更高优先级的中断请求时，（动画 3-4）INTI =0，门 1 输出高电平，即 IS_1=1，（动画 5-9）若此时中断请求 IR_1=1(有中断请求)，当 CPU 发来中断响应信号 INTA=1 时，发出 IR_1 请求的中断源被选中，选中信号经门 7 送入编码电路，产生一个唯一对应的设备地址 001010，并经数据总线送往 CPU 的主存地址寄存器，然后执行该中断源的中断服务程序。

（动画 10-13）另一方面，由于此时 IR_1 非为 0，封锁门 2，使 IS_2，IS_3 全为低电平，即排队识别工作不再向下进行。

（PPT723 第 4 页）（动画 1-2）若 IR_1 无请求，（动画 3-7）则 IR_1=0，门 7 被封锁，不会向编码电路送入选中信号。（动画 8-22）与此同时，因 IR_1 非=1，经门 2 和门 3，使 IS_2=1。如果 IR_2=1，则被选中，数据总线上送出 001011。否则查询链继续向下查询，直至找到发出中断请求信号 IR_i 的中断源设备为止。

（PPT723 第 5 页）（动画 1）最后，我们来看看中断向量是如何产生的？

由于存储器的地址码是一串布尔量的序列，因此常常把地址码称为向量地址。（动画 2）当 CPU 响应中断时，由硬件直接产生一个固定的地址(即向量地址)，（动画 3-6）图中给出了各部分的功能。

（PPT723 第 6 页）（动画 1-2）由向量地址指出每个中断源设备的中断服务程序入口，这种方法通常称为向量中断。

显然，每个中断源分别有一个中断服务程序，而每个中断服务程序又有自己的向量地址。当 CPU 识别出某中断源时，由硬件直接产生一个与该中断源对应的向量地址。

向量中断要求在硬件设计时考虑所有中断源的向量地址，而实际中断时只能产生一个向量地址。（动画 3-4）如图中左边，当排队器输出信息最高位为 1，其余为 0 时，即最高级 IR_1 有请求，经编码器产生 IR_1 的中断向量地址为 00010010（对应的十六进制数为 12H），动画 5）按向量地址 12H 访问主存，找到无条件转移指令“JMP 200”，其中的 200 就是“打印机中断服务程序”入口地址。动画 6-8）同样方式，当排队器输出次高位为 1，其余为 0 时，IR_2 有请求，也能按中断向量地址 13H，找到“显示器服务程序”入口地址 300。

这里需要注意一下：12H，13H，14H，...，所对应的的内存区就是中断向量表。

有些计算机中由硬件产生的向量地址不是直接地址，而是一个“位移量”，这个位移量加上 CPU 某寄存器中存放的基地址，最后得到中断处理程序的入口地址。

还有一种采用向量地址转移的方法。假设有 8 个中断源，由优先级编码电路产生 8 个对应的固定地址码(如 0, 1, 2， ...， 7)，这 8 个单元中存放的是转移指令，通过转移指令可转入设备各自的中断服务程序入口。这种方法允许中断处理程序放在内存中任何地方，非常灵活。

7.2.4 多级中断

（PPT724 第 1 页）（动画 1）同学们，我们这一节来看看多级中断。

（动画 2）多级中断系统是指计算机系统中有相当多的中断源，根据各中断事件的轻重缓急程度不同而分成若干级别，（动画 3）每一中断级分配给一个优先权。一般来说，优先权高的中断级可以打断优先权低的中断服务程序，以程序嵌套方式进行工作。（动画 3）如图所示，三级中断优先权高于二级，而二级中断优先权又高于一级。

（PPT724 第 2 页）（动画 1）多级中断的结构。（动画 2）根据系统的配置不同，多级中断又可以分为一维多级中断和二维多级中断，（动画 3）如图所示。一维多级中断是指每一级中断中只有一个中断源，而二维多级中断是指每一级中断中有多个中断源。（动画 4）图中虚线左边结构为一维多级中断，如果去掉虚线则成为二维多级中断结构。

（PPT724 第 3 页）（动画 1）对多级中断，着重说明如下几点：

（动画 2）1)一个系统若有 n 级中断，在 CPU 中就有 n 个中断请求触发器 IR，总称为中断请求寄存器; 与之对应的有 n 个中断屏蔽触发器 IM，总称为中断屏蔽寄存器。与单级中断不同，在多级中断中，中断屏蔽寄存器的内容是一个很重要的程序现场，

因此在响应中断时，需要把中断屏蔽寄存器的内容保存起来，并设置新的中断屏蔽状态。一般在某一级中断被响应后，要置“1”(关闭)本级和优先权低于本级的中断屏蔽触发器，置“0”(开放)更高级的中断屏蔽触发器，以此来实现正常的中断嵌套。

（动画 3）2)多级中断中的每一级可以只有一个中断源，也可以有多个中断源。在多级中断之间可以实现中断嵌套，但是同一级内有不同中断源的中断是不能嵌套的，必须是处理完一个中断后再响应和处理同一级内其他中断源。

（动画 4）3)设置多级中断的系统一般都希望有较快的中断响应时间，因此首先响应哪一级中断和哪一个中断源，都是由硬件逻辑实现，而不是用程序实现。上一节我们学习的中断优先级排队电路，就是用于决定优先响应中断级的硬件逻辑。另外，在二维中断结构中，除了有中断优先级排队电路确定优先响应中断级外，还要确定优先响应的中断源，一般通过链式查询的硬件逻辑来实现。显然，这里采用了独立请求方式与链式查询方式相结合的方法决定首先响应哪个中断源。

（动画 5）4)和单级中断情况类似，在多级中断中也使用中断堆栈保存现场信息。使用堆栈保存现场的好处是:①控制逻辑简单，保存和恢复现场的过程按先进后出顺序进行。②每级中断不必单独设置现场保护区，各级中断现场可按其顺序放在同一个栈里。

（PPT724 第 4 页）（动画 1）接下来，我们看看多级中断源的识别问题。

在多级中断中，每一级均有一根中断请求线送往 CPU 的中断优先级排队电路，对每一级赋予了不同的优先级。显然，这种结构就是独立请求方式的逻辑结构。

（动画 2）图中给出了独立请求方式的中断优先级排队与中断向量产生的逻辑结构。每个中断请求信号保存在“中断请求”触发器中，经“中断屏蔽”触发器控制后，可能有若干个中断请求信号 IR_i 进如虚线框所示的排队电路。排队电路在若干中断源中决定首先响应哪个中断源，并在其对应的输出线 IR_i 上给出“1”信号，而其他各线为“0”信号(IR_1~IR_4 中只有一个信号有效)。之后，编码电路根据排上队的中断源输出信号 IR_i，产生一个预定的地址码，转向中断服务程序入口地址。

例如，假设图中请求源 1 的优先级最高，请求源 4 的优先级最低。又假定中断请求寄存器的内容为（动画 3）1111，中断屏蔽寄存器的内容为（动画 4）0010，从中断屏蔽触发器的 Q 非端输出结果为（动画 5-6）1101，每一位跟中断请求寄存器中对应位同时经过“与门”，那么，（动画 7-8）进入排队器的中断请求是 1101。根据优先次序，（动画 9-10）排队器输出为 1000。然后由编码器产生中断源 1 所对应的向量地址。

在多级中断中，如果每一级请求线上还连接有多个中断源设备，那么在识别中断源时，还需要进一步用串行链式方式查询。这意味着要用二维方式来设计中断排队逻辑。

本节课后两个例题，请同学们自己研究，互相讨论。

7.2.5 中断控制器

（PPT725 第 1 页）（动画 1）8259 中断控制器是一个集成电路芯片，它将中断接口与优先级判断等功能汇集于一身，常用于微型机系统。（动画 2）其内部结构如图所示。（动画 3-4）8 位中断请求寄存器(IR)接受 8 个外部设备送来的中断请求，每一位对应一个设备。中断请求寄存器的各位送入优先权判断器，（动画 5-6）根据中断屏蔽寄存器(IM)各位的状态来决定最高优先级的中断请求，并将各位的状态送入中断状态寄存器(IS)。IS 保存着判优结果。由（动画 7）控制逻辑向 CPU 发出中断请求信号 INT，并接受 CPU 的中断响应信号 INTA。

（动画 8）数据缓冲器用于保存 CPU 内部总线与系统数据总线之间进行传送的数据。（动画 9）读/写逻辑决定数据传送的方向，其中 $\overline{RD}$ 为读控制，$\overline{WR}$ 为写控制，$\overline{CS}$ 为设备选择，AO 为 I/O 端口识别。

每个 8259 中断控制器最多能控制 8 个外部中断信号，（动画 10）但是可以将多个 8259 进行级联以处理多达 64 个中断请求。在这种情况下允许有一个主中断控制器和多个从中断控制器，称为主从系统。主从控制器的级联是由级联总线 C_0，C_1, C_2 实现的，并将从控制器的中断请求 INT 连入主控制器的某个 IR 端。当有从控制器的中断请求得到响应时，主控制器将得到响应的从控制器编码经级联总线送往从控制器的级联缓冲器，并和从控制器自己的编码相比较，比较一致的从控制器立即向数据总线发送被其选中的 I/O 设备的中断向量。

（PPT725 第 2 页）（动画 1）8259 的中断优先级选择方式有四种：（动画 2-5）①完全嵌套方式；　②轮换优先级方式 A；③轮换优先级方式 B④查询方式

（动画 6）8259 提供了两种屏蔽方式：（动画 7-8）①简单屏蔽方式②特殊屏蔽方式。8259 中断控制器的不同工作方式是通过编程来实现的。CPU 送出一系列的初始化控制字和操作控制字来执行选定的操作。关于这些知识的详细描述，请同学们参看“微机原理”课程相关教材，此处不再赘述。

7.3 DMA 方式

7.3.1 DMA 的基本概念

同学们，前面我们学习了两种主要由程序控制的 I/O 控制方式，接下来我们学习两种主要由硬件实现的 I/O 控制方式。

首先，我们学习 DMA 方式。（PPT731 第 1 页）（动画 1）直接内存访问，是一种完全由硬件执行 I/O 交换的工作方式。（动画 2）在这种方式中，DMA 控制器从 CPU 完全接管对总线的控制，数据交换不经过 CPU，而直接在内存和 I/O 设备之间进行。（动画 3）DMA 方式一般用于高速传送成组数据。（动画 4）DMA 控制器将向内存发出地址和控制信号，修改地址，对传送的字的个数计数，并且以中断方式向 CPU 报告传送操作的结束。

（动画 5）DMA 方式的主要优点是速度快。由于 CPU 根本不参加传送操作，因此就省去了 CPU 取指令、取数、送数等操作。在数据传送过程中，没有保存现场、恢复现场之类的工作。内存地址修改、传送字个数的计数等等，也不是由软件实现，而是用硬件线路直接实现的。所以 DMA 方式能满足高速 I/O 设备的要求，也有利于 CPU 效率的发挥。正因为如此，包括微型机在内，DMA 方式在计算机中被广泛采用。

7.3.2 DMA 传送方式

DMA 技术的出现，使得外围设备可以通过 DMA 控制器直接访问内存，与此同时，CPU 可以继续执行程序。那么 DMA 控制器与 CPU 怎样分时使用内存呢?

通常采用三种方法:①停止 CPU 访内；②周期挪用；③DMA 与 CPU 交替访内。接下来我们详细讨论这三种方法的工作原理和优缺点。

（PPT732 第 1 页）（动画 1）1.停止 CPU 访问内存

（动画 2）图中给出了这种传送方式的时间图，从中我们可以看出:

当外设要求传送一批数据时，由 DMA 控制器发一个停止信号给 CPU，要求 CPU 放弃对地址总线、数据总线和有关控制总线的使用权。DMA 控制器获得总线控制权以后开始进行数据传送。在一批数据传送完毕后，DMA 控制器通知 CPU 可以使用内存，并把总线控制权交还给 CPU。很显然，在这种传送过程中，CPU 基本处于不工作状态或者说保持状态。

（动画 3）它的优点是控制简单，适用于数据传输率很高的设备进行成组传送。（动画 4）缺点是在 DMA 控制器访内阶段，内存的效能没有充分发挥，相当一部分内存

工作周期是空闲的。这是因为，外设传送两个数据之间的间隔一般总是大于内存存储周期，即使高速外设也是如此。例如，软盘读出一个 8 位二进制数大约需要 32μs，而半导体存储器的存取周期小于 0.2μs，因此许多空闲的存储周期不能被 CPU 利用。

（PPT732 第 2 页）（动画 1）2.周期挪用

（动画 2）下图是周期挪用的 DMA 方式示意图，（动画 3）在这种 DMA 传送方法中，当外设没有 DMA 请求时，CPU 按程序要求访问内存；一旦外设有 DMA 请求，则由外设挪用一个或几个内存周期。

外设要求 DMA 传送时可能遇到两种情况:一种是此时 CPU 不需要访内，如 CPU 正在执行乘法指令。由于乘法指令执行时间较长，此时 I/O 访内与 CPU 访内没有冲突，即外设挪用一两个内存周期对 CPU 执行程序没有任何影响。另一种情况是，外设要求访内时 CPU 也要求访内，这就产生了访内冲突，在这种情况下外设访内优先，因为 I/O 访内有时间，要求前一个 I/O 数据必须在下一个访内请求到来之前存取完毕。显然，在这种情况下外设挪用一两个内存周期，意味着 CPU 延缓了对指令的执行。

与前一种方法比较，周期挪用既实现了 I/O 传送，又较好地发挥了内存和 CPU 的效率，是一种广泛采用的方法。但是，外设每一次周期挪用都有申请总线控制权、建立总线控制权和归还总线控制权的过程，所以传送一个字对内存来说要占用一个周期，但对 DMA 控制器来说一般要 2~5 个内存周期(视逻辑线路的延迟而定)。因此，（动画 4）该方法适用于外设读写周期大于内存存储周期的情况

（PPT732 第 3 页）（动画 1）3.DMA 与 CPU 交替访内

（动画 2）其原理示意图如图所示，如果 CPU 的工作周期比内存存取周期长很多，此时采用交替访内的方法可以使 DMA 传送和 CPU 同时发挥最高的效率。假设 CPU 工作周期为 1.2μs，内存取周期小于 0.6μs，那么一个 CPU 周期可分为 C1 和 C2 两个分周期，其中 C1 专供 DMA 控制器访内，C2 专供 CPU 访内。

（动画 3）这种方式不需要总线使用权的申请、建立和归还过程，总线使用权是通过 C1 和 C2 分时控制的。CPU 和 DMA 控制器各自有自己的访内地址寄存器、数据寄存器和读/写信号等控制器。在 C1 周期中，如果 DMA 控制器有访内请求，可将地址、数据等信号送到总线上。在 C2 周期中如果 CPU 有访内请求，同样传送地址、数据等信号。事实上，对于总线，这是用 C1，C2 控制的一个多路转换器，这种总线控制权的转移几乎不需要什么时间，所以对 DMA 传送来讲效率是很高的。

（动画 4）这种传送方式又称为“透明的 DMA”方式，其来由是这种 DMA 传送对 CPU 来说，如同透明玻璃一般，没有任何感觉或影响。在透明的 DMA 方式下工作，CPU 既不停止主程序的运行，也不进入等待状态，是一种高效率的工作方式。当然，相应的硬件逻辑也就更加复杂。

7.3.3 基本的 DMA 控制器

同学们，从上一节的学习中，我们知道，使用 DMA 控制器“接替”CPU，负责完成外设和内存之间的直接数据传送。哪么，DMA 控制器的结构是怎样的呢？DMA 数据传送的过程又如何呢？下面我们一起来看看。

（PPT733 第 1 页）（动画 1）首先，介绍 DMA 控制器的基本组成。

（动画 2）一个 DMA 控制器，实际上是采用 DMA 方式的外设与系统总线之间的接口电路，就是在中断接口的基础上再加 DMA 机构组成。

（动画 3）下图给出了一个最简单的 DMA 控制器组成示意图，它由以下逻辑部件组成:

（动画 4-5）**1）内存地址计数器** 用于存放要交换的数据在内存中的地址。在 DMA 传送前，须通过程序将数据在内存中的起始位置(首地址)送到内存地址计数器。而当 DMA 传送时，每交换一次数据，将址计数器加“1”，从而以增量方式给出内存中要交换的一批数据的地址。

（动画 6-7）**2）字计数器** 用于记录传送数据块的长度。其值也是在数据传送之前由程序预置，交换的字数通常以补码形式表示。在 DMA 传送时，每传送一个字，字计数器就加“1”，当计数器溢出即最高位产生进位时，表示这批数据传送完毕，于是引起 DMA 控制器向 CPU 发中断信号。

（动画 8-9）**3）数据缓冲寄存器** 用于暂存每次传送的数据(一个字)。当输入时，由设备(如磁盘)送往数据缓冲寄存器，再由缓冲寄存器通过数据总线送到内存。反之，输出时，由内存通过数据总线送到数据缓冲寄存器，然后再送到设备。

（动画 10-11）**4）DMA 请求标志** 每当设备准备好一个数据字后给出一个控制信号，使“DMA 请求”标志置“1”。该标志置位后向“控制状态”逻辑发出 DMA 请求，后者又向 CPU 发出总线使用权的请求(HOLD)，CPU 响应此请求后发回响应信号 HLDA，“控制/状态”逻辑接收此信号后发出 DMA 响应信号，使“DMA 请求”标志复位，为交换下一个字做好准备。

（动画 12-13）**5）控制/状态逻辑** 由控制和时序电路以及状态标志等组成，用于修改内存地址计数器和字计数器，指定传送类型(输入或输出)，并对“DMA 请求”信号 HOLD 和 CPU 响应信号 HLDA 进行协调和同步。

（动画 14-15） **6）中断机构** 当字计数器溢出时(全 0)，意味着一组数据交换完毕，由溢出信号触发中断机构，向 CPU 提出中断报告。此处的中断与上一节介绍的 I/O 中断所采用的技术相同，但中断的目的不同，前面是为了数据的输入或输出，而这里是为了报告一组数据传送结束。因此，它们是 I/O 系统中不同的中断事件。

（PPT733 第 1 页）（动画 1）其次，我们来看看 DMA 数据传送过程。

（动画 2）DMA 的数据块传送过程可分为三个阶段:传送前预处理; 正式传送; 传送后处理。

预处理阶段由 CPU 执行几条输入/输出指令，测试设备状态，向 DMA 控制器的设备地址寄存器中送入设备号并启动设备，向内存地址计数器中送入起始地址，向字计数器中送入交换的数据字个数。在这些工作完成后，CPU 继续执行原来的主程序。

当外设准备好发送数据或接受数据时，它发出 DMA 请求，由 DMA 控制器向 CPU 发出总线使用权的请求信号 HOLD。下图给出了停止 CPU 访内方式的 DMA 传送数据的流程图。当外设发出 DMA 请求时，CPU 在指令周期执行结束后响应该请求，并使 CPU 的总线驱动器处于高阻状态。之后，CPU 与系统总线相脱离，而 DMA 控制器接管数据总线与地址总线的控制，并向内存提供地址。于是，在内存和外设之间进行数据交换。每交换一个字，则地址计数器和字计数器加“1”，当计数值到达零时，DMA 操作结束，DMA 控制器向 CPU 提出中断报告。

DMA 的数据传送是以数据块为基本单位进行的，因此，每次 DMA 控制器占用总线后，无论是数据输入操作，还是输出操作，都是通过循环来实现的。当进行输入操作时，外设的数据(一次一个字或一个字节)传向内存; 当进行输出操作时，内存的数据传向外设。

DMA 的后处理进行的工作是，一旦 DMA 的中断请求得到响应，CPU 停止主程序的执行，转去执行中断服务程序做一些 DMA 的结束处理工作。这些工作包括校验送入内存的数据是否正确; 决定继续用 DMA 方式传送下去，还是结束传送; 测试在传送过程中是否发生了错误等等。

7.3.4 选择型和多路型 DMA 控制器

同学们，上一节我们了解了最简单的 DMA 控制器，一个控制器只控制一个 I/O 设备。实际中经常采用的是选择型 DMA 控制器和多路型 DMA 控制器，它们已经被做成集成电路芯片。

（PPT734 第 1 页）（动画 1）我们首先看看选择型 DMA 控制器。（动画 2-3）它在物理上可以连接多个设备，而在逻辑上只许连接一个设备。换句话说，在某一段时间内只能为一个设备服务。（动画 4）下图是选择型 DMA 控制器的逻辑框图。

它的工作原理与前面的简单 DMA 控制器基本相同。除了前面讲到的基本逻辑部件外，（动画 5）还有一个设备号寄存器。数据传送是以数据块为单位进行，每个数据块传送之前的预置阶段，除了用程序中 I/O 指令给出数据块的传送个数、起始地址、操作命令外，还要给出所选择的设备号。从预置开始，一直到这个数据块传送结束，DMA 控制器只为所选设备服务。下一次预置再根据 I/O 指令指出的设备号，选择另一个设备进行服务。显然，选择型 DMA 控制器相当于一个逻辑开关，根据 I/O 指令来控制此开关与某个设备连接。

选择型 DMA 控制器只增加少量硬件达到了为多个外设服务的目的，它特别适合数据传输率很高以至接近内存存取速度的设备。

（PPT734 第 2 页）（动画 1）接下来我们看看多路型 DMA 控制器。

选择型 DMA 控制器不适用于慢速设备。但是多路型 DMA 控制器却适合于同时为多个慢速外设服务。（动画 2）下图给出了独立请求方式的多路型 DMA 控制器的原理图。

（动画 3-4）从图中可以看出，多路型 DMA 控制器不仅在物理上可以连接多个外设，而且在逻辑上也允许这些外设同时工作，各设备以字节交叉方式通过 DMA 控制器进行数据传送。

由于多路型 DMA 同时要为多个设备服务，因此对应多少个设备，在 DMA 控制器内部就有多少组寄存器用于存放各自的传送参数。

（PPT734 第 3 页）（动画 1）下图是一个多路型 DMA 控制器的芯片内部逻辑结构，通过配合使用 I/O 通用接口芯片，它可以对 8 个独立的 DMA 通路(CH)进行控制，使外设以周期挪用方式对内存进行存取。

（动画 2）8 条独立的 DMA 请求线或响应线能在外设与 DMA 控制器之间进行双

向通信。一条线上进行双向通信是通过分时和脉冲编码技术实现的。也可以分别设立 DMA 请求线和响应线实现双向通信。（动画 3）每条 DMA 线在优先权结构中具有固定位置，一般 DMA_0 线具有最高优先权，DMA_7 线具有最低优先权。

（动画 4）控制器中有 8 个 8 位的控制传送长度的寄存器，（动画 5）8 个 16 位的内存地址寄存器。每个长度寄存器和地址寄存器对应一个设备。每个寄存器都可以用程序中的 I/O 指令从 CPU 送入控制数据。每一寄存器组各有一个计数器，用于修改内存地址和传送长度。

（PPT734 第 4 页）（动画 1-6）下面我们通过【例 4】来看看 DMA 控制器是如何为设备提供服务的。

（PPT734 第 5 页）（动画 1-14）解：由图看出，T1 间隔中控制器首先为打印机服务，因为此时只有打印机有请求。T2 间隔前沿磁盘、磁带同时有请求，首先为优先权高的磁盘服务，然后 T3 间隔为磁带服务，之后 T4、T5、T6 间隔分别为磁盘、磁带、磁盘服务，这三个间隔只有一个设备有请求，所以无需判断优先级。在 T7 间隔前沿磁盘、磁带同时有请求，先为优先权高的磁盘服务，然后 T8 间隔为磁带服务。每次服务传送一个字节。在 120us 时间阶段中，为打印机服务只有一次(T1)，从图中看到，这种情况下 DMA 控制器尚有空闲时间，说明它还可以容纳更多设备。

7.4 通道方式

通道是大型计算机中使用的技术。随着时代进步，通道的设计理念有新的发展，并应用到大型服务器甚至微型计算机中。

7.4.1 通道的功能

（PPT741 第 1 页）1、通道的功能 学习了上一讲内容之后，我们知道，DMA 控制器的出现已经减轻了 CPU 对数据输入/输出的控制，使得 CPU 的效率有了显著的提高。而通道的出现则进一步提高了 CPU 的效率。这是因为（动画 1）通道是一个特殊功能的处理器，它有自己的指令和程序，专门负责数据输入输出的传输控制，（动画 2）而 CPU 将“传输控制”的功能下放给通道后只负责“数据处理”功能。这样，（动画 3）通道与 CPU 分时使用存储器，实现了 CPU 内部运算与 I/O 设备的并行工作。

（PPT741 第 2 页）（动画 1）2、通道的结构 （动画 2）下图是典型的具有通道的计算机系统结构图。它具有两种类型的总线，一种是系统总线，它承担通道与存储器、CPU 与存储器之间的数据传输任务。另一种是通道总线，即 I/O 总线，它承担外

设与通道之间的数据传送任务。这两类总线可以分别按照各自的时序同时进行工作。

从图中可以看出，通道总线可以接若干个 I/O 模块（或设备控制器），一个 I/O 模块可以接一个或多个设备。因此，从逻辑结构上讲，I/O 系统一般具有四级连接：CPU 与存储器→通道→I/O 模块→外设。为了便于通道统一管理各设备，通道与 I/O 模块之间用统一的标准接口（如串行接口、并行接口、IDE 接口、SCSI 接口、USB 接口等），I/O 模块与设备之间则根据设备要求不同而采用专用接口。

具有通道的机器一般是大型计算机和服务器，数据流量很大。如果所有的外设都接在一个通道上，那么通道将成为限制系统效能的瓶颈。因此，大型计算机的 I/O 系统一般接有多个通道。显然，设立多个通道的另一好处是，对不同类型的外设可以进行分类管理。

图中的存储管理部件(MMU)，是存储器的控制部件，它的主要任务是根据事先确定的优先次序，决定下一周期由哪个部件使用系统总线访问存储器。由于大多数 I/O 设备是旋转性的设备，如磁盘，读写信号具有实时性，不及时处理会丢失数据，所以通道与 CPU 同时要求访问存储器时，通道优先权高于 CPU。在多个通道有访存请求时，选择通道的优先权高于多路通道，因为前者一般连接磁盘等高速设备。

（PPT741 第 3 页）（动画 1）3、通道的基本功能 （动画 2-5）是执行通道指令，组织外设和内存进行数据传输，按 I/O 指令要求启动外设，向 CPU 报告中断等，具体有以下五项任务:

1）接受 CPU 的 I/O 指令，按指令要求与指定的外设进行通信。

2）从存储器选取属于该通道程序的通道指令，经译码后向 I/O 控制器模块发送各种命令。

3）组织外设和存储器之间进行数据传送，并根据需要提供数据缓存的空间，以及提供数据存入存储器的地址和传送的数据量。

4）从外设得到设备的状态信息，形成并保存通道本身的状态信息，根据要求将这些状态信息送到存储器的指定单元，供 CPU 使用。

5）将外设的中断请求和通道本身的中断请求，按次序及时报告 CPU。

（PPT741 第 4 页）（动画 1）4、CPU 对通道的管理 （动画 2） CPU 是通过执行 I/O 指令以及处理来自通道的中断，实现对通道的管理。（动画 3）来自通道的中断有两种，一种是数据传送结束中断，另一种是故障中断。

通常把 CPU 运行操作系统管理程序的状态称为管态，而把 CPU 执行目的程序或

用户程序时的状态称为目态。大型计算机的 I/O 指令都是管态指令，只有当 CPU 处于管态时，才能运行 I/O 指令，目态时不能运行 I/O 指令。这是因为大型计算机的软、硬件资源为多个用户所共享而不是分给某个用户专用。

（动画 4）5、通道对设备控制器的管理　（动画 5）通道通过使用通道指令来控制 I/O 模块进行数据传送操作，并以通道状态字接收 I/O 模块反映的外设的状态。因此，（动画 6）I/O 模块是通道对 I/O 设备实现传输控制的执行机构。I/O 模块的具体任务如下：

1）从通道接受通道指令，控制外设完成所要求的操作；

2）向通道反映外设的状态；

3）将各种外设的不同信号转换成通道能够识别的标准信号。

7.4.2 通道的类型

（PPT742 第 1 页）（动画 1）根据通道的工作方式，通道分为选择通道、多路通道两种类型。一个系统可以兼有两种类型的通道，也可以只有其中一种。

（动画 2）1)选择通道　（动画 3）选择通道又称高速通道，在物理上它可以连接多个设备，但是这些设备不能同时工作，在某一段时间内通道只能选择一个设备进行工作。选择通道很像一个单道程序的处理器，在一段时间内只允许执行一个设备的通道程序，只有当这个设备的通道程序全部执行完毕后，才能执行其他设备的通道程序。

（动画 4）选择通道主要用于连接高速外围设备，如磁盘、磁带等，信息以数据块方式高速传输。由于数据传输率很高，所以在数据传送期间只为一台设备服务是合理的。但是这类设备的辅助操作时间很长，如磁盘机平均找道时间是 10ms，磁带机走带时间可以长达几分钟。在这样长的时间里通道处于等待状态，因此整个通道的利用率不是很高。

多路通道又称多路转换通道，在同一时间能处理多个 I/O 设备的数据传输。它又分为数组多路通道和字节多路通道。

（PPT742 第 2 页）（动画 1）2)数组多路通道　它对选择通道的一种改进，它的基本思想是：（动画 2）当某设备进行数据传送时，通道只为该设备服务；（动画 3）当设备在执行寻址等控制性动作时，通道暂时断开与这个设备的连接，挂起该设备的通道程序，去为其他设备服务，即执行其他设备的通道程序。所以数组多路通道很像一个多道程序的处理器。

（动画 4）数组多路通道不仅在物理上可以连接多个设备，而且在一段时间内能交替执行多个设备的通道程序，换句话说在逻辑上可以连接多个设备，这些设备应是高速设备。

由于数组多路通道既保留了选择通道高速传送数据的优点，又充分利用了控制性操作的时间间隔为其他设备服务，使通道效率充分得到发挥，因此数组多路通道在大型系统中得到较多应用

（PPT742 第 3 页）（动画 1）3）字节多路通道 它主要用于连接大量的低速设备，如键盘、打印机等等，这些设备的数据传输率很低。例如数据传输率是 1000B/s，即传送一个字节的时间是 1ms，而通道从设备接收或发送一个字节只需要几百纳秒，因此通道在传送两个字节之间有很多空闲时间，字节多路通道正是“见缝插针”地利用这个空闲时间为其他设备服务。

（动画 2）字节多路通道和数组多路通道共同之处是：它们都是多路通道，在一段时间内能交替执行多个设备的通道程序，使这些设备同时工作。（动画 3-4）它们的不同之处，主要是：

一、数组多路通道允许多个设备同时工作，但只允许一个设备进行传输型操作，其他设备进行控制型操作。而字节多路通道不仅允许多个设备同时操作，而且也允许它们同时进行传输型操作。

二、数组多路通道与设备之间数据传送的基本单位是数据块，通道必须为一个设备传送完一个数据块以后，才能为别的设备传送数据块。而字节多路通道与设备之间数据传送的基本单位是字节，通道为一个设备传送一个字节后，又可以为另一个设备传送一个字节，因此各设备与通道之间的数据传送是以字节为单位交替进行。

通道结构的进一步发展，就出现了两种计算机 I/O 系统结构，一种是通道结构的 I/O 处理器，通常称为输入输出处理器（IOP），IOP 可以和 CPU 并行工作，提供高速的 DMA 处理能力，实现数据的高速传送。但是它不是独立于 CPU 工作的，而是主机的一个部件。有些 IOP，例如 Intel 8089 IOP，还提供数据的变换、搜索以及字装配/拆卸能力。这类 IOP 可应用于服务器及微型计算机中。

另一种是外围处理机(PPU)。PPU 基本上是独立于主机工作的，它有自己的指令系统，完成算术/逻辑运算，读/写主存储器，与外设交换信息等。有的外围处理机干脆就选用已有的通用机。外围处理机 I/O 方式一般应用于大型高性能计算机系统中。