高校名称：北京航空航天大学

课程名称：《计算机组成原理》

目录

第一章 导论 课堂笔记 3
第一章 导论 关键词汇 5
第一章 导论 FAQ 6
第一章 导论 拓展资源 7
第二章 计算机的应用 课堂笔记 8
第二章 计算机的应用 关键词汇 9
第二章 计算机的应用 FAQ 9
第二章 计算机的应用 拓展资源 10
第三章 系统总线 课堂笔记 11
第三章 系统总线 关键词汇 20
第三章 系统总线 FAQ 21
第三章 系统总线 拓展资源 22
第四章 存储系统 课堂笔记 23
第四章 存储系统 关键词汇 53
第四章 存储系统 FAQ 55
第四章 存储系统 拓展资源 57
第五章 输入输出系统 课堂笔记 58
第五章 输入输出系统 关键词汇 89
第五章 输入输出系统 FAQ 90
第五章 输入输出系统 拓展资源 92
第六章 计算机的运算方法 课堂笔记 93
第六章 计算机的运算方法 关键词汇 109
第六章 计算机的运算方法 FAQ 111
第六章 计算机的运算方法 拓展资源 114
第七章 指令系统 课堂笔记 115
第七章 指令系统 关键词汇 128
第七章 指令系统 FAQ 129
第七章 指令系统 拓展资源 130
第八章 CPU 课堂笔记 132
第八章 CPU 关键词汇 145
第八章 CPU FAQ 147
第八章 CPU 拓展资源 149
第九章 控制单元的功能 课堂笔记 150
第九章 控制单元的功能 关键词汇 155
第九章 控制单元的功能 FAQ 156
第九章 控制单元的功能 拓展资源 157
第十章 控制单元的设计 课堂笔记 159
第十章 控制单元的设计 关键词汇 166
第十章 控制单元的设计 FAQ 167
第十章 控制单元的设计 拓展资源 168

第一章 导论 课堂笔记

◆主要知识点掌握程度

重点掌握冯·诺依曼思想，计算机系统的层次结构，知道计算机由那些部分组成。明白软硬件的概念以及划分。清楚计算机各部分的特点以及性能技术指标。

◆ 知识点整理

一、计算机的发展

（一）计算机的发展

一代 电子管计算机 1946-1958，中国使用的这代计算机即为 108 机；

二代 晶体管计算机 1958-1964 由二极管 三极管 电容等分离元件组成。我国的 DJS 系列如 112 机，一直使用到差不多 80 年代，计算次数已经达到了 50 万次；

三代 集成电路计算机 1964-1971 典型的有：PDP-8 DJS-130 IBM-360

四代 大规模和超大规模集成电路计算机 比如银河多媒体计算机，广泛应用于信息处理

（二）冯·诺依曼计算机的特点

冯·诺依曼体系计算机的核心思想是“**存储程序**”的概念。它的特点如下：

1、 计算机由运算器、存储器、控制器和输入设备、输出设备五大部件组成；
2、 指令和数据都用二进制代码表示；
3、 指令和数据都以同等地位存放于存储器内，并可按地址寻访；
4、 指令是由操作码和地址码组成，操作码用来表示操作的性质，地址码用来表示操作数所在存储器中的位置；
5、 指令在存储器内是顺序存放的；
6、 机器以运算器为核心，输入输出设备与存储器的数据传送通过运算器。

（三）智能计算机 /第五代计算机

冯.诺依曼计算机的能力局限于算术运算和逻辑运算；

计算机的能力不如人类，计算机的优势在于运算速度，用最笨的方法做最快的运算（计算机和世界冠军的象棋比赛）；

二、计算机系统的基本概念

（一）计算机的软硬件概念和五大部件

典型的冯·诺依曼计算机是以运算器为中心的，其中，输入、输出设备与存储器之间的数据传送都需通过运算器。现代的计算机已转化为以存储器为中心，

五大部件的功能是：

1、运算器用来完成算术运算和逻辑运算，并将运算的中间结果暂存在运算器内；
2、存储器用来存放数据和程序；
3、控制器用来控制、指挥程序和数据的输入、运行及处理运算结果；
4、输入设备用来将人们熟悉的信息形式转换为机器能识别的信息形式，常见的有键盘、鼠标等；
5、输出设备可将机器运算结果转换为人们熟悉的信息形式如打印机输出、显示器输出等。

计算机的五大部件在控制器的统一指挥下，有条不紊地自动工作。

由于运算器和控制器在逻辑关系和电路结构上联系十分紧密，尤其在大规模集成电路制作工艺出现后，这两大部件往往制作在同一芯片上，因此，通常将他们合起来统称为中央处理器，简称 CPU。把输入设备与

输出设备简称为 I/O 设备。

因此，现代计算机可认为由三大部分组成：CPU、I/O 设备及主存储器 MM。CPU 与 MM 合起来称为主机，I/O 设备叫作外设。存储器分为主存储器 MM 和辅助存储器。主存可直接与 CPU 交换信息，辅存又叫外存。

（二）计算机系统的层次结构

1、传统计算机和现代计算机基本结构基本按照冯诺依曼思想组成。

2、计算机的层次结构—不同的计算机分为一级，多级层次。

三、计算机的工作过程

（一）工作过程

建立数学模型 确立计算方法 编写解题程序 上机运行

（二）从一个实例看计算机的工作过程 计算：ax^2+bx+c

1、把题目分解为（ax+b）x+c

2、解题步骤划分为几大步骤：

- ✓ x 送运算器；
- ✓ 乘法，ax 结果存入运算器；
- ✓ 加法，ax+b 结果存入运算器；
- ✓ 乘法，（ax+b）x 结果存入运算器
- ✓ 存数到内存；
- ✓ 打印；
- ✓ 停机

3、为这几个步骤编写操作码分别是：

0001 0100 0011 0001 0011 0010 0101 0000

4、存储器映像：

如下图所示：

十	二	指令 操作码	地址码	说明
0	0000	0001	0100	取x
1	0001	0100	1001	乘法
2	0010	0011	1010	加法
3	0011	0100	1000	乘法
4	0100	0011	1011	加c
5	0101	0010	1100	存数
6	0110	0101	1100	打印
7	0111	0110		停机
8	1000	X		数据
9	1001	a		数据
10	1010	b		数据
11	1011	c		数据
12	1100			结果

四、计算机硬件的主要技术指标

（一）机器字长

机器字长是指 CPU 一次能处理数据的位数，通常与 CPU 的寄存器位数有关，字长越长，数的表示范围也越大，精度也越高。机器的字长也会影响机器的运算速度。

机器字长对硬件的造价也有较大的影响。它将直接影响加法器、数据总线以及存储字长的位数。所以机器字长的确定不能单从精度和数的表示范围来考虑。

（二）存储容量

存储器的容量应该包括主存的容量和辅存的容量。

主存容量是指主存中存放二进制代码的总数。即存储容量=存储单元个数×存储字长

现代计算机中常以字节的个数来描述容量的大小，因为一个字节被定义为 8 位二进制代码， 故用字节数便能反映主存容量。同理，辅存容量也可用字节数来表示，如，某机辅存（如硬盘）容量为 4GB（1G=1KM=2^{30}，B 用来表示一个字节）

（三）运算速度

计算机的运算速度与许多因素有关，如机器的主频、执行什么样的操作、主存本身的速度（主存速度快，取指、取数就快）等等都有关。计算机的运算速度普遍采用单位时间内执行指令的平均条数来衡量，并用 MPIS 作为计量单位，即每秒执行百万条指令。 也有用 CPI 即执行一条指令所需的时钟周期数， 或用 FLOPS 即每秒浮点运算次数来衡量运算速度。

第一章 导论 关键词汇

1、运算器

运算器包括三个寄存器和一个算逻单元 ALU。其中 ACC 为累加器，MQ 为乘商寄存器，X 为操作数寄存器。这三个寄存器在完成不同运算时，所存放在操作数类别也各不相同。

2、存储器

主存储器包括存储体、各种逻辑部件及控制电路等。存储体由许多存储单元组成，每个存储单元又包含若干个存储元件，每个存储元件能寄存一位二进制代码“0”或“1”。可见，一个存储单元可存储一串二进制代码，称这串二进制代码为一个存储字，这串二进制代码的个数叫做存储字长。

3、控制器

控制器是计算机组成的神经中枢，由它指挥全机各部件自动、协调地工作。具体而言，它首先要命令存储器读出一条指令，这叫取指过程。接着对这条指令进行分析，指出该指令要完成什么样的操作，并按寻址特征指明操作数的地址，这叫分析指令过程。最后根据操作数所在的地址，取出操作数并完成某种操

作，这叫作执行过程。以上就是通常所说的完成一条指令操作的取指、分析和执行三阶段。

4、机器字长

机器字长是指 CPU 一次能处理数据的位数，通常与 CPU 的寄存器位数有关，字长越长，数的表示范围也越大，精度也越高。机器的字长也会影响机器的运算速度。

5、存储容量

存储器的容量应该包括主存的容量和辅存的容量。

6、运算速度

计算机的运算速度与许多因素有关，如机器的主频、执行什么样的操作、主存本身的速度（主存速度快，取指、取数就快）等等都有关。

7、CPU

控制器与运算器合起来，称为 CPU，即中央处理器。

8、主机

CPU 与主机合起来称为主机。

第一章 导论 FAQ

一、比较数字计算机和模拟计算机的特点。

模拟计算机的特点：数值由连续量来表示，运算过程是连续的；

数字计算机的特点：数值由数字量（离散量）来表示，运算按位进行。

二、字计算机如何分类？分类的依据是什么？

分类：数字计算机分为专用计算机和通用计算机。通用计算机又分为巨型机、大型机、中型机、小型机、微型机和单片机六类。

分类依据：专用和通用是根据计算机的效率、速度、价格、运行的经济性和适应性来划分的。

通用机的分类依据主要是体积、简易性、功率损耗、性能指标、数据存储容量、指令系统规模和机器价格等因素。

三、诺依曼型计算机的主要设计思想是什么？它包括哪些主要组成部分？

冯. 诺依曼型计算机的主要设计思想是：存储程序和程序控制。

存储程序：将解题的程序（指令序列）存放到存储器中；

程序控制：控制器顺序执行存储的程序，按指令功能控制全机协调地完成运算任务。

主要组成部分有：控制器、运算器、存储器、输入设备、输出设备。

四、什么是存储容量？什么是单元地址？什么是数据字？什么是指令字？

存储容量：指存储器可以容纳的二进制信息的数量，通常用单位 KB、MB、GB 来度量，存储容量越大，表示计算机所能存储的信息量越多，反映了计算机存储空间的大小。

单元地址：单元地址简称地址，在存储器中每个存储单元都有唯一的地址编号，称为单元地址。

数据字： 若某计算机字是运算操作的对象即代表要处理的数据，则称数据字。

指令字： 若某计算机字代表一条指令或指令的一部分，则称指令字。

五、什么是指令？什么是程序？

指令：计算机所执行的每一个基本的操作。

程序：解算某一问题的一串指令序列称为该问题的计算程序，简称程序。

六、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？

一般来讲，在取指周期中从存储器读出的信息即指令信息；而在执行周期中从存储器中读出的信息即为数据信息。

七、什么是内存？什么是外存？什么是 CPU？什么是适配器？简述其功能。

内存：一般由半导体存储器构成，装在底版上，可直接和 CPU 交换信息的存储器称为内存储器，简称内存。用来存放经常使用的程序和数据。

外存：为了扩大存储容量，又不使成本有很大的提高，在计算机中还配备了存储容量更大的磁盘存储器和光盘存储器，称为外存储器，简称外存。外存可存储大量的信息，计算机需要使用时，再调入内存。

CPU：包括运算器和控制器。基本功能为：指令控制、操作控制、时间控制、数据加工。

适配器：连接主机和外设的部件，起一个转换器的作用，以使主机和外设协调工作。

八、计算机的系统软件包括哪几类？说明它们的用途。

系统软件包括：（1）服务程序：诊断、排错等

（2）语言程序：汇编、编译、解释等

（3）操作系统

（4）数据库管理系统

用途：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及用途。

第一章 导论 拓展资源

穿孔卡是早期计算机的信息输入设备，通常可以储存 80 列数据。它是一种很薄的纸片， 面积为 190×84 毫米。首次使用穿孔卡技术的数据处理机器，是美国统计专家赫曼·霍列瑞斯（H.Hollerith）博士的伟大发明。

公元 1880 年，美利坚合众国举行了一次全国性人口普查，为当时 5000 余万的美国人口登记造册。当时美国经济正处于迅速发展的阶段，人口流动十分频繁；再加上普查的项目繁多，统计手段落后，从当年元月开始的这次普查，花了 7 年半的时间才把数据处理完毕。也就是说，直到快进行第二次人口普查时，美国政府才能得知第一次人口普查期间全国人口的状况。

人口普查需要大量处理的是数据，如年龄、性别等用调查表采集的项目，并且还要统计出每个社区有多少儿童和老人，有多少男性公民和女性公民等等。这些数据是否也可由机器自动进行统计？采矿工程师霍列瑞斯想到了纺织工程师杰卡德 80 年前发明的穿孔纸带。杰卡德提花机用穿孔纸带上的小孔，主要用来控制提花操作的步骤，即编写程序，霍列瑞斯则进一步设想要用它来储存和统计数据，发明一种自动制表的机器。两年后，霍列瑞斯博士离开了人口局，到专利事务所工作过一段时间，也曾任教于麻省理工学院，一边工作，一边致力于自动制表机的研制。

霍列瑞斯首先把穿孔纸带改造成穿孔卡片，以适应人口数据采集的需要。由于每个人的调查数据有若干不同的项目，如性别、籍贯、年龄等等。霍列瑞斯把每个人所有的调查项目依次排列于一张卡片，然后根据调查结果在相应项目的位置上打孔。例如，穿孔卡片“性别” 栏目下，有“男”和“女”两个选项；“年龄”栏目下有从“0 岁”到“70 岁以上”等系列选项，如此等等。统计员可以根据每个调查对象的具体情况，分别在穿孔卡片各栏目相应位置打出小孔。每张卡片都代表着一位公民的个人档案。

霍列瑞斯博士巧妙的设计在于自动统计。他在机器上安装了一组盛满水银的小杯，穿好孔的卡片就放置在这些水银杯上。卡片上方有几排精心调好的探针，探针连接在电路的一端，水银杯则连接于电路的另一端。与杰卡德提花机穿孔纸带的原理类似：只要某根探针撞到卡片上有孔的位置，便会自动跌落下去，与水银接触接通电流，启动计数装置前进一个刻度。由此可见，霍列瑞斯穿孔卡表达的也是二进制信息：有孔处能接通电路计数，代表该调查项目为“有”(“1”)，无孔处不能接通电路计数，表示该调查项目为“无”(“0”)。

直到 1888 年，霍列瑞斯博士才实际完成自动制表机设计并申报了专利。他发明的这种机电式计数装置，比传统纯机械装置更加灵敏，因而被 1890 年后历次美国人口普查选用，获得了巨大的成功。例如，1900 年进行的人口普查全部采用霍列瑞斯制表机，平均每台机器可代替 500 人工作，全国的数据统计仅用了 1 年多时间。虽然霍列瑞斯发明的并不是通用计算机，除了能统计数据表格外，它几乎没有别的什么用途，然而，制表机穿孔卡第一次把数据转变成二进制信息。在以后的计算机系统里，用穿孔卡片输入数据的方法一直沿用到 20 世纪 70 年代，数据处理也发展成为电脑的主要功能之一。

依托自己发明的制表机，霍列瑞斯博士创办了一家专业制表机公司，但不久就因资金周转不灵陷入困境，被另一家 CTR 公司兼并。1924 年，CTR 公司更名为“国际商业机器公司”，英文缩写“IBM”，专门生产打孔机、制表机一类产品。

杰卡德和霍列瑞斯分别用开创了程序设计和数据处理之先河。以历史的目光审视他们的发明，正是这种程序设计和数据处理，构成了电脑“软件”的雏形。

第二章 计算机的应用 课堂笔记

◆ **主要知识点掌握程度**

了解软件系统和用户软件的相关知识点，及现代计算机的发展。

◆ **知识点整理**

一、软件系统

系统软件又称为系统程序，主要用来管理整个计算机系统，监视服务，使系统资源得到合理调度，确保高效运行。主要包括：标准程序库、语言处理程序、操作系统、服务性程序、数据库管理系统、网络软件等等。其中，操作系统是计算机工作必须的系统软件，用来管理计算机的资源(如：处理器、存储器、外设以及各种编译、应用程序等)自动调度用户的作业程序，使多个户能有效地共用一套计算机系统：提供操作界面，提供程序运行平台，使计算机的使用效率大大提高。

如：

机器语言；

符号语言（汇编语言）；

高级语言 FORTRAN—1954 年；

PASCAL—1968 年，结构化程序设计语言；

C 面向对象程序设计；

JAVA 网络环境的面向对象；

编译程序，解释程序；

操作系统：自动调度系统的软硬件资源；

数据库 数据库管理系统；

网络软件；

装配程序 诊断程序。

二、用户软件

应用软件又称为应用程序，它是用户根据任务所编制的各种程序，如科学计算程序、数据处理程序、过程控制程序、事务处理程序等。

如：购物软件；

文字处理，排版软件；

电子表格；

数据库应用系统；

网络通讯；

CAD；

大型游戏。

现代计算机的发展：

第二章 计算机的应用 关键词汇

1、系统软件

系统软件又称为系统程序，主要用来管理整个计算机系统，监视服务，使系统资源得到合理调度，确保高效运行。

2、用户软件

是用户根据任务所编制的各种程序，如科学计算程序、数据处理程序、过程控制程序、事务处理程序等。

第二章 计算机的应用 FAQ

一、系统软件包括哪些？

系统软件主要包括：标准程序库、语言处理程序、操作系统、服务性程序、数据库管理系统、网络软件等等。其中，操作系统是计算机工作必须的系统软件，用来管理计算机的资源(如：处理器、存储器、外设以及各种编译、应用程序等)自动调度用户的作业程序，使多个户能有效地共用一套计算机系统：提供操作界面，提供程序运行平台，使计算机的使用效率大大提高。

二、用户软件包括哪些？

如购物软件；文字处理，排版软件；电子表格；数据库应用系统；网络通讯；CAD；大型游戏。

第二章 计算机的应用 拓展资源

—— 计算机的发展及应用

一、计算机的发展史

（一）计算机的产生和发展

1、一代电子管计算机

ENIAC 的出现标志着人类进入了计算机时代。

2、第二代晶体管计算机

1947 年在 Bell 实验室成功地用半导体硅片作基片，制成了第一个晶体管，它的小体积、低耗电以及载流子高速运行的特点，使真空管望尘莫及。用晶体管取代电子管以后，计算机的性能有了很大的提高。

3、第三代集成电路计算机

集成电路制作技术就是利用光刻技术把晶体管、电阻、电容等构成的单个电路制作在一块极小（如几个平方微米）的硅片上。进一步发展，实现了将成百上千个这样的门电路全部制作在一块极小的硅片上，并引出与外部连接的引线，这样，一次便能制作成成百上千相同的门电路，又一次大大地缩小了计算机的体积，大幅度下降了耗电量，极大地提高了机器的可靠性。这就是人们称作的小规模集成电路（SSI）和中等规模集成电路（MSI）的第三代计算机。

4、第四代大规模、超大规模集成电路技术制成的计算机

第三代计算机之后，人们没有达成定义新一代计算机的一致意见，如果从硬件技术上讲，可以把用大规模、超大规模集成电路技术制成的计算机称为第四代计算机。

（二）微型计算机的出现和发展

1971 年， 美国 Intel 公司研制成世界上第一个 4 位微处理器芯片 4004， 集成了 2300 个晶体管。随后，微处理器经历了 4 位、8 位、16 位、32 位和 64 位几个阶段的发展，芯片的集成度和速度都有很大的提高。

Intel 公司的典型产品有 ：

8080 、8088、8086、80286、80386、80486、Pentium(80586) 、Pentium pro（P6） 、Pentium Ⅱ 、Pentium Ⅲ 、Pentium Ⅳ

（三）软件技术的兴起和发展

软件发展有以下几个特点：

（1） 开发周期长；

（2） 制作成本昂贵；

（3） 检测软件产品质量的特殊性。

二、计算机的应用

（一）科学计算和数据处理

1、科学计算

科学计算是计算机的重要应用领域之一。其特点是计算量大和数值变化范围大。主要应用领域是天文学、量子化学、空气动力学和核物理学等领域，此外在其他学科和工程设计方面也都得到了广泛的应用。

2、数据处理

数据处理是计算机应用最广泛的领域。是利用计算机对大量的数据、文字、图像等信息进行收集、存储、分类、检索、排序、汇总、统计等处理。

它的特点是数据量大而计算并不复杂。主要应用在企业管理中的财务管理、人事管理、仓储管理、图书检索以及银行业务等方面。

（二）工业控制和实时控制

计算机实时控制过程就是计算机及时收信、检测和分析被控对象的有差数据，按照某种最佳的控制规律进行高速运算并根据计算结果发出控制信号，进行自动控制的过程。主要用于工业生产和现代化军事领域中。

（三）网络技术的应用

1、电子商务

电子商务是指任何一个组织机构可利用 Internet 网络来改变他们与客户、供应商、业务伙伴和内部员工的交流，也可以认为是消费者、销售者和结算部门之间利用 Internet 完成商品采购和支付收款的过程。

2、网络教育

基于 Internet 教育网络的建立，学生受教可以不受时间、空间和地域的限制，通过网络伸展到全球的每个角落，建立起真正意义上的开放式的虚拟学校，每个学生可以在任意时间、任意地点通过网络自由地学习

3、敏捷制造

敏捷制造由两部分组成：敏捷制造的基础结构和敏捷制造的虚拟企业。前者为形成虚拟企业提供环境和条件，后者对市场不可预期的变化作出迅速响应。

（四）虚拟现实

虚拟现实是利用计算机生成的一种模拟环境，通过多种传感设备使用户“投入”到该环境中，实现用户与环境直接进行交互的目的。

（五)办公自动化和管理信息系统

办公自动化是利用计算机及自动化的办公设备来替代“笔、墨、纸、砚”及办公人员的部分脑、体力劳动，从而提高了办公的质量和效率。

（六）CAD/CAM/CIMS

计算机辅助工程是指利用计算机来辅助人们进行各项工作。 CAD 即计算机辅助设计、CAM 计算机辅助制造、CIMS 为计算机集成制造系统。

（七）多媒体技术

多媒体技术是计算机技术和视频、音频及通信等技术集成的产物。它是用来实现人和计算机交互地对各种媒体的采集、传输、转换、编辑、存储、管理，并由计算机综合处理为文字、图形、动画、音响、影像等视听信息而有机合成的新媒体。

（八）人工智能

人工智能是专门研究如何使用计算机来模拟人的智能的技术。近年来在模式识别、语音识别、专家系统和机器人制作方面都取得了很大的成就。

三、 计算机的发展

计算机的发展方向：

1、 微型化
2、 巨型化
3、 网络化
4、智能化

第三章 系统总线 课堂笔记

◆ **主要知识点掌握程度**

知道计算机硬件系统的几个组成部分：中央处理器、存储器、I/O 以及系统总线。重点掌握系统总线常见的几个分类，结构，以及重要的控制方式。

◆ **知识点整理**

一、总线的基本概念

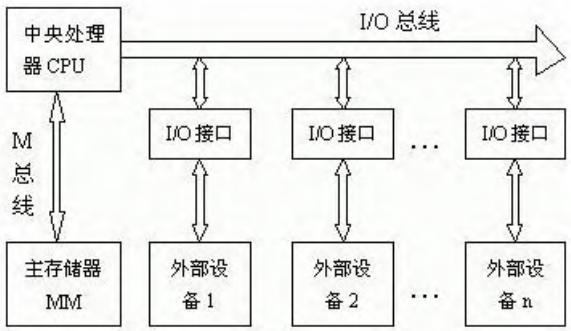
（一）总线的作用和概念

总线是连接计算机内部多个部件之间的信息传输线，是各部件共享的传输介质。多个部件和总线相连，在某一时刻，只允许有一个部件向总线发送信号，而多个部件可以同时从总线上接收相同的信息。

（二）总线的几种结构

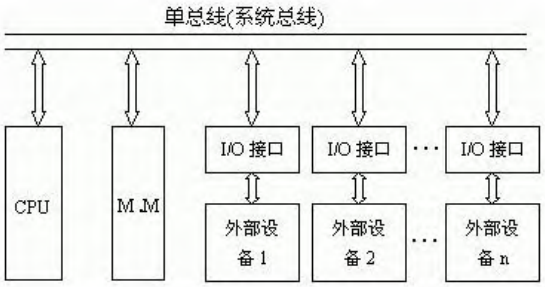
1、以 CPU 为中心的双总线结构

在这种结构中，**存储总线（M 总线）**用来连接 CPU 和主存，**输入/输出总线（I/O 总线）**用来建立 CPU 和各 I/O 之间交换信息的通道。各种 I/O 设备通过 I/O 接口挂到 I/O 总线上。这种结构在 I/O 设备与主存交换信息时仍然要占用 CPU，因此会影响 CPU 的工作效率。



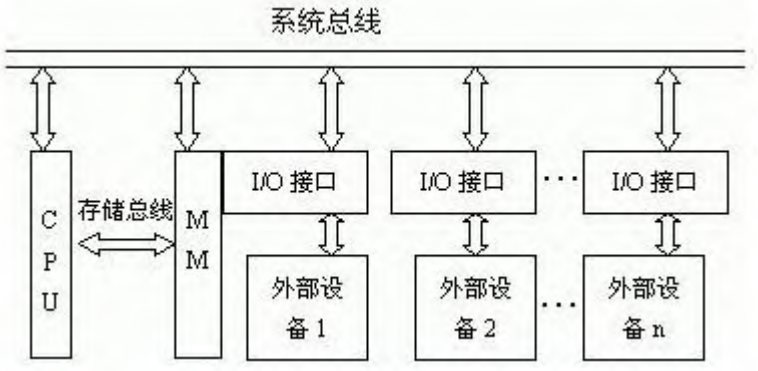
2、单总线结构

在这种结构中，**将 CPU、主存和 I/O 设备都挂到一组总线上，** 形成单总线结构的计算机。这种结构最明显的特点就是，当 I/O 与主存交换信息时，原则上不影响 CPU 的工作，CPU 仍可继续处理不访问主存或 I/O 的操作，这就使 CPU 工作效率有所提高。但是，因为只有一组总线，当某一时刻各部件都要占用时，就会出现争夺现象。



3、以存储器为中心的双总线结构

这种总线是在单总线基础上，又单独开辟一条 CPU 与主存之间的通路，叫**存储总线**。这组总线速度高，只供主存与 CPU 之间传输信息。这样既提高了传输效率，又减轻了系统总线的负担，还保留了 I/O 与存储器交换信息时不经过 CPU 的特点。



二、总线的分类

总线的应用很广泛，从不同角度可以有不同的分类方法。**按数据传送方式**可分为并行传输总线和串行传输总线。

若**按总线的使用范围划分**，则又有计算机总线、测控总线、网络通信总线等。

下面按连接部件的不同，分几类介绍总线。

（一）片内总线

片内总线是指芯片内部的总线，如在 CPU 芯片内部， 寄存器与寄存器之间、寄存器与算术逻辑单元之间都有总线连接。

（二）系统总线

系统总线是指 CPU、主存、I/O 各大部件之间的信息传输线。由于这些部件通常都安放在主板或各个插件板（插卡）上，故又称为板级总线或板间总线。

按传输信息的不同，可分为三类：数据总线、地址总线和控制总线。

1、数据总线

数据总线用来传输各功能部件之间的数据信息，它是双向传输总线，其位数与机器字长、存储字长有关。数据总线的条数称为数据总线宽度，它是衡量系统性能的一个重要参数。

2、地址总线

地址总线主要用来指出数据总线上的源数据或目的数据在主存单元的地址。它是单向传输的。地址线的位数与存储单元的个数有关，如地址线为 20 根，则对应的存储单元个数为 2^{20}。

3、控制总线

控制总线是用来发出各种控制信号的传输线。**常见的控制信号有：**

时钟	用来同步各种操作
复位	表示各模块恢复初始状态
总线请求	表示某部件需获得总线使用权
总线允许	表示需要获得总线使用权的部件已获得了控制权
中断请求	表示某部件提出中断请求
中断确认	表示中断请求已被接收
存储器写	将数据总线上的数据写至存储器的指定地址单元内
存储器读	将指定存储单元中的数据读到数据总线上
I/O 读	从指定的 I/O 端口将数据输出到指定的 I/O 端口内
I/O 写	将数据总线上的数据输出到指定的 I/O 端口内
数据确认	表示数据已被接收或已读到总线上

（三）通信总线

这类总线用于计算机系统之间或计算机系统与其他系统（如控制仪表、移动通讯等）之间的通信。

三、总线特性及性能指标

尺寸、形状、管脚数及排列顺序、传输方向和有效的电平范围、每根传输线的功能、信号的时序关系。

（一）总线特性

1、机械特性

机械特性是指总线在机械方式上的一些性能，如插头与插座使用的标准，它们的几何尺寸、形状、引脚的个数以及排列的顺序，接头处的可靠接触等等。

2、电气特性

电气特性是指总线的每一根传输线上信号的传递方向和有效的电平范围。通常规定由 CPU 发出的信号叫输出信号，送入 CPU 的信号叫输入信号。总线的电平定义与 TTL 相符。如 RS-232C（串行总线接口标准），其电气特性规定低电平表示逻辑“1”，并要求电平低于-3V；用高电平表示逻辑“0”，还要求高电平需高于+3V，额定信号电平为-10V 和+10V 左右。

3、功能特性

功能特性是指总线中每根传输线的功能，如地址总线用来指出地址号；数据总线传递数据；控制总线发出控制信号等。可见各条线其功能不一。

4、时间特性

时间特性是指总线中的任一根线在什么时间内有效。每条总线上的各种信号，互相存在着一种有效时序的关系，因此，时间特性一般可用信号时序图来描述。

（二）总线的性能指标

总线的性能指标包括：

（1）总线宽度：它是指数据总线的根数， 用 bit（位）表示，如 8 位、16 位、32 位、64 位。

（2）标准传输：即在总线上每秒能传输的最大字节量，用 MB/s（每秒多少兆字节）表示。

（3）时钟同步/异步：总线上的数据与时钟同步工作的总线称同步总线，与时钟不同步工作的总线称为异步总线。

（4）总线复用： 通常地址总线与数据总线在物理上是分开的两种总线。地址总线传输地址码，数据总线传输数据信息。为了提高总线的利用率，优化设计，特将地址总线和数据总线共用一条物理线路，只是某一时刻该总线传输地址信号，另一时刻传输数据信号或命令信号。这叫总线的多路复用。

（5）信号线数： 即地址总线、数据总线和控制总线三种总线数的总和。

（6）总线控制方式： 包括并发工作、自动配置、仲裁方式、逻辑方式、计数方式等。

（7） 其他指标：如负载能力问题等。

（三）总线标准

所谓**总线标准**，可视为系统与各模块、模块与模块之间一个互连的标准界面。这个界面对两端的模块都是透明的，即界面的任一方只需根据总线标准的要求完成自身一面接口的功能要求，而无需了解对方接口与总线的连接要求。因此，按总线标准设计的接口可视为通用接口。

目前流行的总线标准有：

（1）ISA（Industrial Standard Architecture）总线，又称 AT 总线，它采用独立于 CPU 的总线时钟，因此 CPU 可采用比总线频率更高的时钟，有利于 CPU 性能的提高。但 ISA 总线没有支持总线仲裁的硬件逻辑，因此不支持多台主设备系统， 且 ISA 上的所有数据的传送必须通过 CPU 或 DMA 接口来管理，因此使 CPU 花费了大量时间来控制与外部设备交换数据。**ISA 总线时钟频率为 8MHz，最大传输率为 16MB/s。**

（2）EISA(Extended Industrial Standard Architecture)总线，是一种在 ISA 基础上扩充开放的总线标准，它与 ISA 完全兼容，它从 CPU 中分离出了总线控制权，是一种智能化的总线，能支持多总线主控和突发方式的传输。 **EISA 总线的时钟频率为 8MHz， 最大传输率可达 33MB/s**，数据总线为 32 位，地址总线为 32 位，扩充 DMA 访问。

(3)VL-BUS 是由 VESA(Video Electronic Standard Association 视频电子标准协会)提出的局部总线标准。所谓局部总线是指在系统外，为两个以上模块提供的高速传输信息通道。 VL-Bus 是由 CPU 总线演化而来的，采用 CPU 的时钟频率达 33MHz、数据线为 32 位，配有局部控制器。通过局部控制的判断， 将高速 I/O 直接挂在 CPU 的总线上，实现 CPU 与高速外设之间的高速数据交换。

(4)PCI(Peripheral Component Interconnect 外部设备互连总线)是由 Intel 公司提供的总线标准。它与 CPU

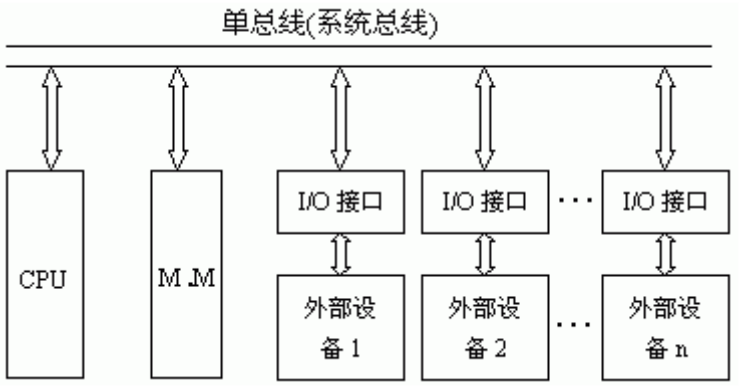
时钟频率无关，自身采用 33MHz 总线时钟，数据线为 32 位，可扩充到 64 位，数据传输率达 132～246MB/s。它具有很好的兼容性，与 ISA、EISA 总线均可兼容，可以转换为标准的 ISA、EISA。它能支持无限读写突发方式，速度比直接使用 CPU 总线的局部总线快。它可视为 CPU 与外设间的一个中间层，通过 PCI 桥路(PCI 控制器)与 CPU 相连。

EISA 和 PCI 都具有即插即用的功能。

四、总线结构

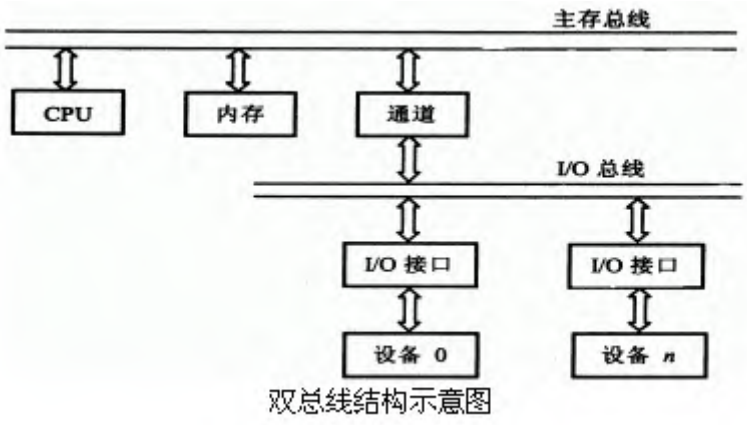
（一）单总线结构

它是一组总线连接整个计算机系统的各大功能部件，各大部件之间的所有的信息传送都通过这组总线。其结构如图所示。单总线的优点是允许 I/O 设备之间或 I/O 设备与内存之间直接交换信息，只需 CPU 分配总线使用权，不需要 CPU 干预信息的交换。所以总线资源是由各大功能部件分时共享的。单总线的缺点是由于全部系统部件都连接在一组总线上，所以总线的负载很重，可能使其吞量达到饱和甚至不能胜任的程度。故多为小型机和微型机采用。



（二）双总线结构

它有两条总线，一条是内存总线，用于 CPU，内存和通道之间进行数据传送;另一条是 I/O 总线，用于多个外围设备与通道之间进行数据传送。其结构如图所示。双总线结构中，通道是计算机系统中的一个独立部件，使 CPU 的效率大为提高，并可以实现形式多样而更为复杂的数据传送。双总线的优点是以增加通道这一设备为代价的，通道实际上是一台具有特殊功能的处理器，所以双总线通常在大，中型计算机中采用。

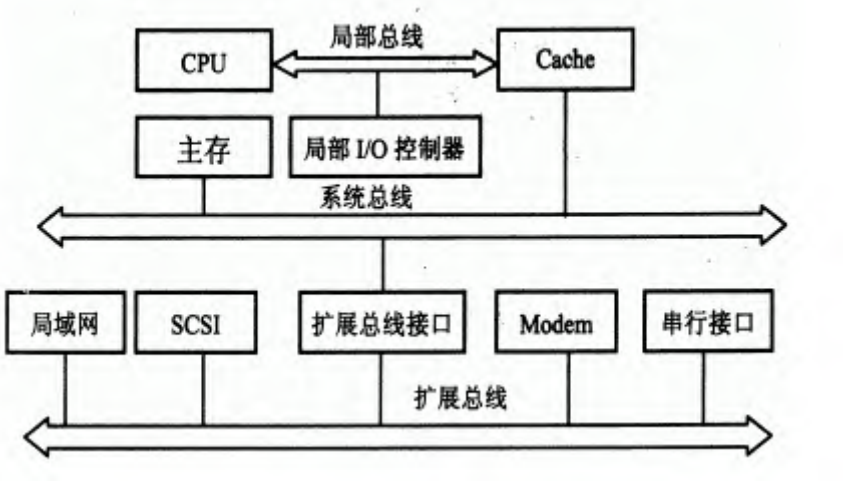


双总线结构示意图

（三）三总线结构

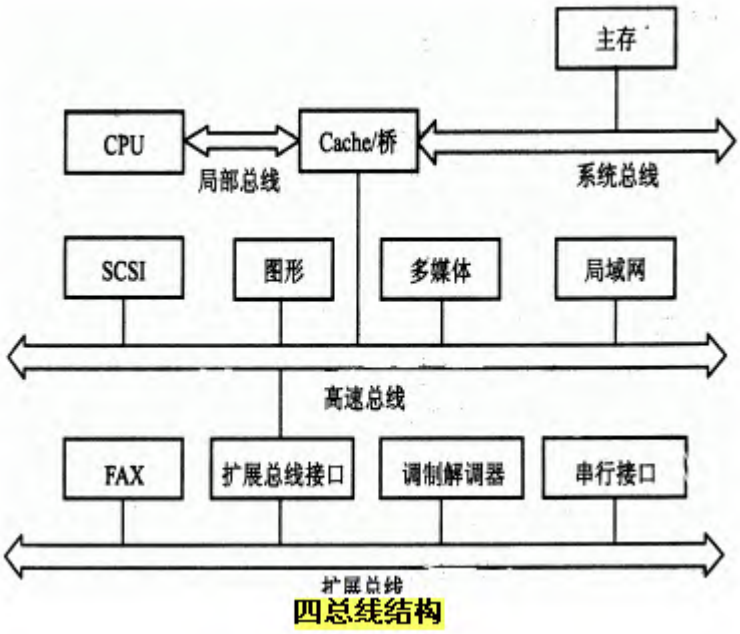
即在计算机系统各部件之间采用三条各自独立的总线来构成信息通路。这三条总线是:内存总线，输入

/输出(I/O)总线和直接内存访问(DMA)总线，如图所示。内存总线用于 CPU 和内存之间传送地址，数据的控制信息;I/O 总线供 CPU 和各类外设之间通讯用;DMA 总线使内存和高速外设之间直接传送数据。一般来说，在三总线系统中，任一时刻只使用一种总线;但若使用多入口存储器，内存总线可与 DMA 总线同时工作，此时三总线系统可以比单总线系统运行得更快。但是三总线系统中，设备到不能直接进行信息传送，而必须经过 CPU 或内存间接传送，所以三总线系统总线的工作效率较低。



（四）四总线结构

在三总线的基础上，又增加了一条与计算机系统紧密相连的高速总线。在高速总线上挂接了一些高速性能的外设，如高速局域网、图形工作站、多媒体、SCSI 等。 它们通过 Cache 控制机构中的高速总线桥或高速缓冲器与系统总线和局部总线相连，使得这些高速设备与处理器更密切。而一些较低速的设备如图文传真 FAX、 调制解调器及串行接口仍然挂在扩展总线上，并由扩展总线接口与高速总线相连。



四总线结构

五、总线控制

总线控制主要包括判优控制和通信控制。

（一）总线判优控制

多个主设备同时要使用总线时，就由总线控制器的判优、仲裁逻辑按一定的优先等级顺序，确定哪个主设备能使用总线。

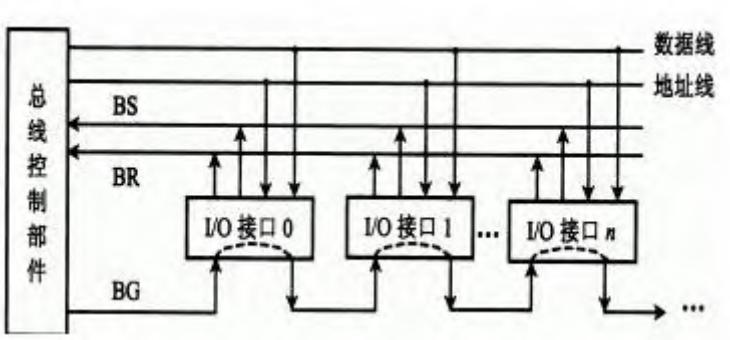
只有获得总线使用权的主设备才能开始传送数据。

总线判优控制可分集中式和分布式两种。

集中控制将控制逻辑集中在一处(如在 CPU 中)，分散控制将控制逻辑分散在与总线连接的各个部件或设备上。

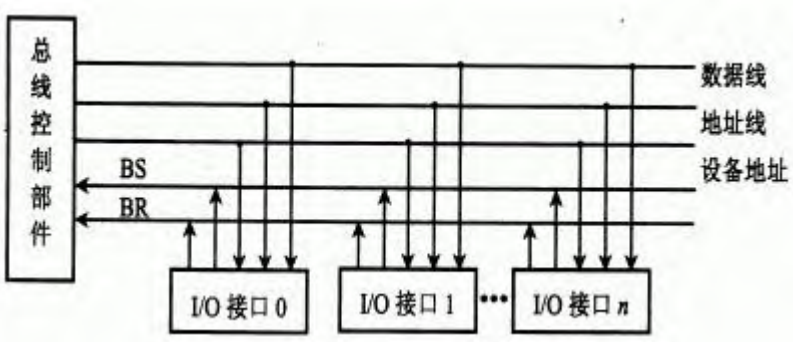
常见的集中控制有三种优先权仲裁方式：

1、链式查询



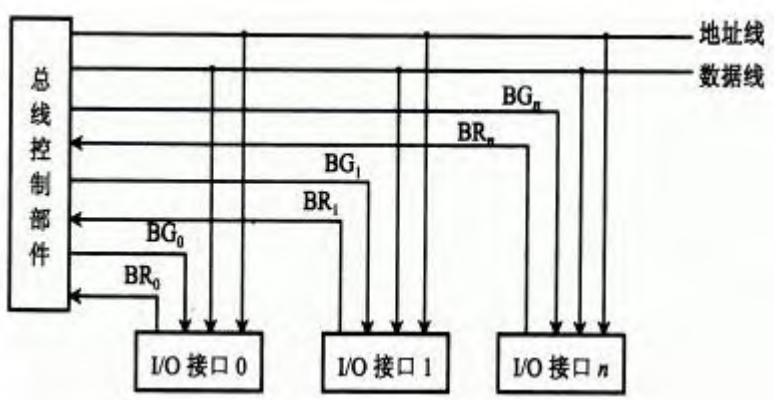
图中控制总线中有**三根线**用于总线控制（BS 总线忙；BR 总线请求、BG 总线同意），其中总线同意信号 BG 是串行地从一个 I/O 接口送到下一个 I/O 接口。如果 BG 到达的接口有总线请求，BG 信号就不再往下传。意味着该接口获得了总线使用权，并建立总线忙 BS 信号，表示它占用了总线。可见在查询链中，离总线控制部件最近的设备具有最高的优先级。这种方式的**特点**是：只需很少几根线就能按一定优先次序实现总线控制，并且很容易扩充设备，但对电路故障很敏感。

2、计数器定时查询



它与链式查询方式相比，多了一组设备地址线，少了一根总线同意线 BG。总线控制部件接到由 BR 送来的总线请求信号后，在总线未被使用(BS＝0)的情况下，由计数器开始计数，向各设备发出一组地址信号。当某个有总线请求的设备地址与计数值一致时，便获得总线使用权，此时终止计数查询。这种方式的特点是：计数可以从“0”开始，此时设备的优先次序是固定的；计数也可以从终止点开始，即是一种循环方法，此时设备使用总线的优先级相等；计数器的初始值还可由程序设置，故优先次序可以改变。此外，对电路故障不如链式查询方式敏感，但增加了主控制线(设备地址)数，控制也较复杂。

3、独立请求方式



工作原理: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。当设备要求使用总线时，便发出该设备的请求信号。总线控制器中的排队电路决定首先响应哪个设备的请求，给设备以授权信号BGi。

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。

（二）总线通信控制

总线在完成一次传输周期时，可分为四个阶段：

（1）申请分配阶段：由需要使用总线的主模块(或主设备)提出申请，经总线仲裁机构决定下一传输周期的总线使用权授于某一申请者；

（2）寻址阶段：取得了使用权的主模块，通过总线发出本次打算访问的从模块(或从设备)的存储地址或设备地址及有关命令，启动参与本次传输的从模块；

（3）传数阶段：主模块和从模块进行数据交换，数据由源模块发出经数据总线流入目的模块；

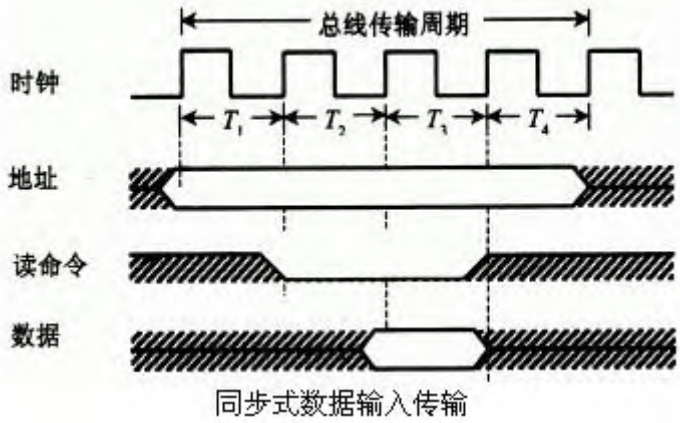
（4）结束阶段：主模块的有关信息均从系统总线上撤除，让出总线使用权。

总线通信控制主要解决通信双方如何获知传输开始和传输结束，以及通信双方如何协调如何配合。

一般常用四种方式：同步通信、异步通信、半同步通信和分离式通信。

1、同步通信

通信双方由统一时标控制数据传送称为同步通信。时标通常由CPU的总线控制部件发出，送到总线上的所有部件；也可以由每个部件各自的时序发生器发出，但必须由总线控制部件发出的时钟信号对它们进行同步。



同步式数据输入传输

上图表示某个输入设备向CPU传输数据的同步通信过程。

图中总线传输周期是总线上两个部件完成一次完整而可靠的传输时间，它包含 4 个时钟周期 T_1、T_2、T_3、T_4。

主模块在 T_1 时刻发出地址信息；T_2 时刻发出读命令；从模块按照所指定的地址和命令进行一系列内部动作，必须在 T_3 时刻前找到 CPU 所需的数据，并送到数据总线上；CPU 在 T_3 时刻开始，一直维持到 T_4 时刻，可以从数据线上获取信息并送到其内部寄存器中；T_4 时刻开始输入设备不再向数据总线上传送数据，撤消它对数据总线的驱动。如果总线采用三态驱动电路，则从 T_4 起，数据总线呈浮空状态。

同步通信在系统总线设计时，T_1、T_2、T_3、T_4 都有明确的、惟一的规定。

对于读命令，其传输周期为：

T_1 主模块发地址；

T_2 主模块发读命令；

T_3 从模块提供数据；

T_4 主模块撤消读命令。

对于写命令，其传输周期为：

T_1 主模块发地址；

$T_{1.5}$ 主模块提供数据；

T_2 主模块发出写命令，从模块接收到命令后，必须在规定时间内将数据总线上的数据写到地址总线所指明的单元中：

T_4 主模块撤消写命令和数据等信号。

优点：规定明确、统一，模块间的配合简单一致。其缺点是主从模块时间配合属强制性“同步”，必须在限定时间内完成规定的要求。并且对所有从模块都用同一限时，这就势必造成对各不相同速度的部件而言，必须按最慢速度部件来设计公共时钟，严重影响总线的工作效率，也给设计带来了局限性，缺乏灵活性。

同步通信一般用于总线长度较短，各部件存取时间比较一致的场合。

2、异步通信

异步通信克服了同步通信的缺点，允许各模块速度的不一致性，给设计者充分的灵活性和选择余地。它没有公共的时钟标准；不要求所有部件严格的统一动作时间，而是**采用应答方式(又称握手方式)**，即当主模块发出请求(Request)信号时，一直等待从模块反馈回来“响应”(Acknowledge)信号后才开始通信。当然，这就要求主从模块之间增加两条应答线(即握手交互信号线 Handshaking)。

异步通信方式可分为不互锁、半互锁和全互锁三种类型，如下图所示。

（1）不互锁方式

主模块发出请求信号后，不等待接到从模块的回答信号，而是经过一段时间。确认从模块已收到请求信号后，便撤消其请求信号；从设备接到请求信号后，在条件允许时发出回答信号，并且经过一段时间，确认主设备已收到回答信号后，自动撤消回答信号。可见通信双方并无互锁关系。

（2）半互锁方式

主模块发出请求信号，待接到从模块的回答信号后再撤消其请求信号，存在着简单的互锁关系：而从模块发出回答信号后，不等待主模块回答，在一段时间后便撤消其回答信号，无互锁关系。故称半互锁方式。

（3）全互锁方式

主模块发出请求信号，待从模块回答后再撤其请求信号；从模块发出回答信号，待主模块获知后，再

撤消其回答信号。故称全互锁方式。

3、半同步通信

半同步通信时序可为：

T_1 主模块发地址；

T_2 主模块发命令；

T_w 当$\overline{WAIT}$为低电平有效时，进入等待，其间隔与 T 统一；

T_w 当$\overline{WAIT}$为低电平有效时，进入等待，其间隔与 T 统一；

……

T_3 从模块提供数据(若属读命令)；

T_4 从模块撤消数据。

半同步通信控制方式比异步通信简单，在全系统内各模块又在统一的系统时钟控制下同步工作，可靠性较高，同步结构较方便。其缺点是对系统时钟频率不能要求太高，故从整体上来看，系统工作的速度还是不很高。

4、分离式通信

分离式通信的基本思想：将一个传输周期(或总线周期)分解为两个子周期。

在第一个子周期中，主模块 A 在获得总线使用权后将命令、地址以及其他的有关信息，包括该主模块编号(当有多个主模块时，此编号尤为重要)发到系统总线上，经总线传输后，由有关的从模块 B 接收下来。主模块 A 向系统总线发布这些信息只占用总线很短的时间，一旦发送完，立即放弃总线使用权，以便其他模块使用。

在第二个子周期中，当 B 模块收到 A 模块发来的有关命令信号后，经选择、译码、读取等一系列内部操作，将 A 模块所需的数据准备好，便由 B 模块申请总线使用权，一旦获准，B 模块便将 A 模块的编号、B 模块的地址，A 模块所需的数据令系列信息送到总线上，供 A 模块接收。很明显，上述两个传输子周期都只有单方向的信息流，每个模块都变成了主模块。

这种通信方式的**特点**是：①各模块欲占用总线使用权都必须提出申请；②在得到总线使用权后，主模块在限定的时间内向对方传送信息，采用同步方式传送，不再等待对方的回答信号；②各模块在准备数据传送的过程中都不占用总线，使总线可接受其他块的请求；④总线被占用时都在作有效工作，或者通过它发命令，或者通过它传送数据，不存在空闲等待时间，最充分地发挥了总线的有效占用。

第三章 系统总线 关键词汇

1、片内总线

片内总线是指芯片内部的总线，如在 CPU 芯片内部， 寄存器与寄存器之间、寄存器与算术逻辑单元之间都有总线连接。

2、系统总线

系统总线是指 CPU、主存、I/O 各大部件之间的信息传输线。按传输信息的不同，可分为三类：数 据总线、地址总线和控制总线。

3、半同步通信

半同步通信集同步与异步通信之优点，既保留了同步通信的基本特点，如所有的地址、命令、数据信号的发出时间，都严格参照系统时钟的某个前沿开始，而接收方都采用系统时钟后沿时刻来进行判断识别。

同时又像异步通信那样，允许不同速度的模块和谐地工作。为此增设了一条“等待”($\overline{WAIT}$)响应信号线。

4、总线判优控制

总线上所连接的各类设备，按其对总线有无控制功能可分为主设备和从设备两种。主设备对总线有控制权，从设备只能响应从主设备发来的总线命令。总线上信息的传送是由主设备启动的，如某个主设备欲与另一个设备(从设备)进行通信时，首先由主设备发出总线请求信号，若多个主设备同时要使用总线时，就由总线控制器的判优、仲裁逻辑按一定的优先等级顺序，确定哪个主设备能使用总线。只有获得总线使用权的主设备才能开始传送数据。

5、总线通信控制

众多部件共享总线，在争夺总线使用权时，只能按各部件的优先等级来解决。而在传送通信时间上，按分时方式来解决，即哪个部件获得使用，此刻就由它传送，下一部件获得使用，接着下一时刻传送。这样一个接一个轮流交替传送。

6、单，双，混合总线结构

单总线结构将 CPU、主存、I/O 设备都挂在一组总线上，允许 I/O 之间、I/O 与主存之间直接交换信息。这种结构简单，也便于扩充，但所有的传送都通过这组共享总线，因此极易形成计算机系统的瓶颈。

双总线结构的特点是将速度较低的 I/O 设备从单总线上分离出来，形成主存总线与 I/O 总线分开的结构。通道是一个具有特殊功能的处理器，CPU 将一部分功能下放给通道，使其对 I/O 设备具有统一管理的功能，以完成外部设备与主存之间的数据传送，其系统的吞吐能力可以相当大。这种结构大多用于大、中型计算机系统。

如果将速率不同的 I/O 设备进行分类， 然后将它们连接在不同的通道上，那么计算机系统的利用率将会更高，由此发展成多总线结构。

第三章 系统总线 FAQ

一、总线有哪些特性和性能指标？

特性：

机械特性 电气特性 功能特性 时间特性

性能指标：

总线宽度 标准传输 时钟同步/异步 总线复用

信号线数 总线控制方式

其他指标：如负载能力问题等。

二、同步通信和异步通信的区别和各自的优缺点？

（1） 通信双方由统一时标控制数据传送称为同步通信。时标通常由 CPU 的总线控制部件发出，送到总线上的所有部件；也可以由每个部件各自的时序发生器发出，但必须由总线控制部件发出的时钟信号对它们进行同步。

优缺点：通信双方由统一时标控制数据传送称为同步通信。时标通常由 CPU 的总线控制部件发出，送到总线上的所有部件；也可以由每个部件各自的时序发生器发出，但必须由总线控制部件发出的时钟信号对它们进行同步。

（2） 异步通信克服了同步通信的缺点，允许各模块速度的不一致性，给设计者充分的灵活性和选择余地。它没有公共的时钟标准；不要求所有部件严格的统一动作时间，而是采用应答方式(又称握手方式)，即当主模块发出请求(Request)信号时，一直等待从模块反馈回来“响应”(Acknowledge)信号后才开始通信。当然，这就要求主从模块之间增加两条应答线(即握手交互信号线 Handshaking)。

优缺点：设计灵活，但另外增添了应答线路

三、分析分离式通信方式的思想和优缺点？

分离式通信的基本思想是将一个传输周期分解为两个子周期。在第一个子周期中，主模块 A 在获得总线使用权后将命令、地址以及其他的有关信息，发到系统总线上，经总线传输后，由有关的从模块 B 接收下来。在第二个子周期中，B 模块将 A 模块发来的有关命令信号经一系列内部操作，申请总线使用权，一旦获准，B 模块便将 A 模块的编号、B 模块的地址，A 模块所需的数据令系列信息送到总线上，供 A 模块接收。

优缺点分析：

（1）各模块欲占用总线使用权都必须提出申请；

（2）在得到总线使用权后，主模块在限定的时间内向对方传送信息，采用同步方式传送，不再等待对方的回答信号；

（3）各模块在准备数据传送的过程中都不占用总线，使总线可接受其他块的请求；

（4）总线被占用时都在作有效工作，或者通过它发命令，或者通过它传送数据，不存在空闲等待时间，最充分地发挥了总线的有效占用。从而实现了总线为多个主从模块间进行信息交叉重叠并行式传送，这对大型计算机系统是极为重要的。当然，这种方式控制比较复杂，一般在普通微机系统很少采用。

四、异步通信方式分类如何？

异步通信方式可分为不互锁、半互锁和全互锁三种类型：

（1）不互锁方式。主模块发出请求信号后，不等待接到从模块的回答信号，而是经过一段时间。确认从模块已收到请求信号后，便撤消其请求信号；从设备接到请求信号后，在条件允许时发出回答信号，并且经过一段时间，确认主设备已收到回答信号后，自动撤消回答信号。可见通信双方并无互锁关系。

（2）半互锁方式。主模块发出请求信号，待接到从模块的回答信号后再撤消其请求信号，存在着简单的互锁关系：而从模块发出回答信号后，不等待主模块回答，在一段时间后便撤消其回答信号，无互锁关系。故称半互锁方式。

（3）全互锁方式。主模块发出请求信号，待从模块回答后再撤其请求信号；从模块发出回答信号，待主模块获知后，再撤消其回答信号。故称全互锁方式。

第三章 系统总线 拓展资源

在一个计算机系统中，采用哪种总线结构，往往对计算机系统的性能有很大影响。下面从三个方面来讨论这种影响。

最大存储容量

初看起来，一个计算机系统的最大存储容量似乎与总线无关，但实际上总线结构对最大存储容量也会产生一定的影响。例如在单总线系统中，对主存和外设进行存取的差别，仅仅在于出现在总线上的地址不同，为此必须为外围设备保留某些地址。由于某些地址必须用于外围设备，所以在单总线系统中，最大主存容量必须小于由计算机字长所决定的可能的地址总数。 在双总线系统中，对主存和外设进行存取的判断是利用各自的指令操作码。由于主存地址和外设地址出现于不同的总线上，所以存储容量不会受到外围设备多少的影响。

指令系统

在双总线系统中，CPU 对存储总线和系统总线必须有不同的指令系统，这是因为操作码规定了要使用哪一条总线，所以在双总线系统中，访存操作和输入/输出操作各有不同的指令。 另一方面，在单总线系统中，访问主存和 1/O 传送可使用相同的操作码，或者说使用相同的指令，但它们使用不同的地址。

吞吐量

计算机系统的吞吐量是指流入，处理和流出系统的信息的速率。它取决于信息能够多快地输入内存，

CPU 能够多快地取指令，数据能够多快地从内存取出或存入，以及所得结果能够多快地从内存送给一台外围设备。这些步骤中的每一步都关系到主存，因此，系统吞吐量主要取决于主存的存取周期。 由于上述原因，采用双端口存储器可以增加主存的有效速度。

整个总线分成如下四部分：

1 数据传送总线：由地址线，数据线，控制线组成。

2 仲裁总线：包括总线请求线和总线授权线。

3 中断和同步总线:用于处理带优先级的中断操作，包括中断请求线和中断认可线。

4 公用线：包括时钟信号线，电源线，地线，系统复位线以及加电或断电的时序信号线等。

总线结构实例

大多数计算机采用了分层次的多总线结构。在这种结构中，速度差异较大的设备模块使用不同速度的总线，而速度相近的设备模块使用同一类总线。显然，这种结构的优点不仅解决了总线负载过重的问题，而且使总线设计简单，并能充分发挥每类总线的效盲目。

PCI 总钱 用于连接高速的 1/O 设备模块，如图形显示器适配器，网络接口控制器，硬盘控制器等。通过"桥"芯片，上面与更高速的 CPU 总线相连，下面与低速的 ISA 总线相接。PCI 总线是一个 32(或 64 位)的同步总线，32 位(或 64 位)数据/地址线是同一组线，分时复用。总线时钟频率为 33。3MHZ，总线带宽是 132MB/s。 PCI 总线采用集中式仲裁方式，有专用的 PCI 总线仲裁器。主板上一般有 3 个 PCI 总线扩充槽。

ISA 总线 Pentium 机使用该总线与低速 1/O 设备连接。主板上一般留有 3-4 个 ISA 总线扩充槽，以便使用各种 16 位/8 位适配器卡。该总线支持 7 个 DMA 通道和 15 级可屏蔽硬件中断。另外，ISA 总线控制逻辑还通过主板上的片级总线与实时钟/日历，ROM，键盘和鼠标控制器(8042 微处理器)等芯片相连接。

第四章 存储系统 课堂笔记

◆ **主要知识点掌握程度**

掌握存储器的分类，分级结构以及主存储器的技术指标；重点掌握存储器的原理、随机读写存储器，明白只读存储器和闪速存储器、高速存储、Cache 存储器、虚拟存储器之前的区别与联系，了解存储保护。比较学习不同存储器和存储方式。

◆ **知识点整理**

一、概述

存储器是计算机系统中的记忆设备，用来存放程序和数据。

（一）存储器的分类

1、按存储介质分类

(1) 半导体存储器

是一种以半导体电路作为存储媒体的存储器。

按其制造工艺可分为：双极晶体管存储器和 MOS 晶体管存储器。

按其存储原理可分为：静态和动态两种。

其**优点**是：体积小、存储速度快、存储密度高、与逻辑电路接口容易。

主要用作高速缓冲存储器、主存储器、只读存储器、堆栈存储器等。

半导体存储器的两个技术指标是：存储容量和存取时间。

缺点：当电源消失时，所存信息也随即丢失，是一种易失性存储器。

(2) 磁表面存储器

所谓磁表面存储，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。磁盘

存储器、磁带存储器均属于磁表面存储器。

按载磁体形状的不同，可分为磁盘、磁带盒磁鼓。

(3) 光盘存储器

光盘存储器是应用激光在记录介质(磁光材料)上进行读写的存储器，具有非易失性的特点。光盘记录密度高、耐用性好、可靠性高和可互换性强等。

2、按存取方式分类

（1）随机存储器（RAM）

RAM 是一种可读写存储器， 其特点是存储器的任何一个存储单元的内容都可以随机存取，而且存取时间与存储单元的物理位置无关。

由于存储信息原理的不同， RAM 又分为静态 RAM (以触发器原理寄存信息)和动态 RAM(以电容充放电原理寄存信息)。

（2）只读存储器

只读存储器是能对其存储的内容读出，而不能对其重新写入的存储器。

只读存储器分为掩膜型只读存储器 MROM（Masked ROM）、可编程只读存储器 PROM(Programmable ROM)、可擦除可编程只读存储器 EPROM(Erasable Programmable ROM)、用电可擦除可编程的只读存储器 EEPROM(Electrically Erasable Programmable ROM)。

（3）串行访问存储器

如果对存储单元进行读写操作时，需按其物理位置的先后顺序寻找地址，则这种存储器叫做串行访问存储器。

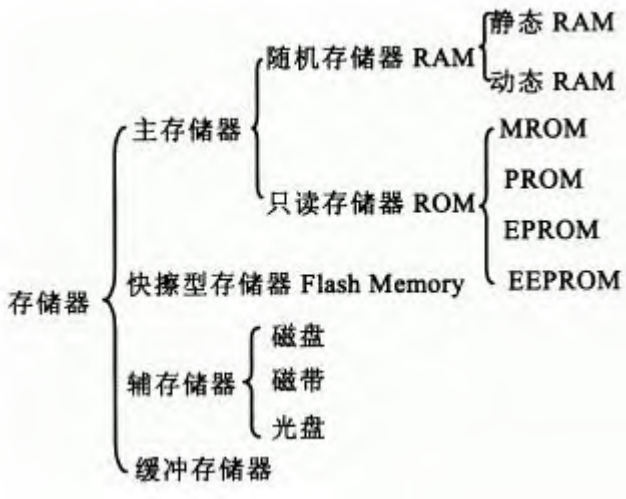
3、按在计算机中的作用分类

分为主存储器、辅助存储器、高速缓冲存储器、控制存储器等。

主存储器的主要特点是它可以和 CPU 直接交换信息。

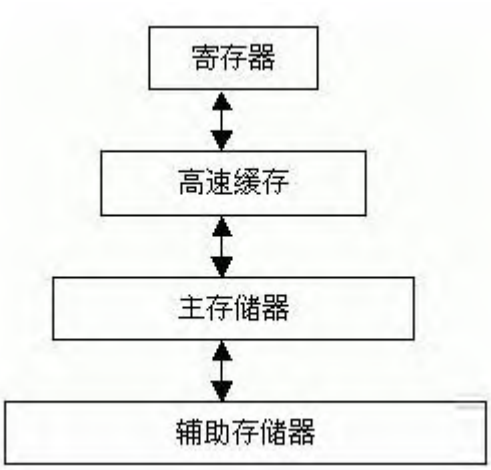
辅助存储器是主存储器的后援存储器，用来存放当前暂时不用的程序和数据，它不能与 CPU 直接交换信息。

两者相比主存速度快、容量小、每位价格高；辅存速度慢、容量大、每位价格低。缓冲存储器用在两个速度不同的部件之中， 如 CPU 与主存之间可设置一个快速缓冲存储器，起到缓冲作用。



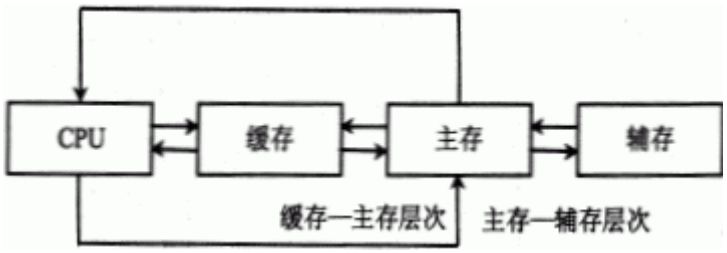
（二）存储器的层次结构

存储器有三个主要特性：速率、容量和价格/位（简称位价）。



图中由上至下，每位的价格越来越低，速度越来越慢，容量越来越大，CPU 访问的频度也越来越少。

实际上，存储器的层次结构主要体现在缓存—主存、主存—辅存这两个存储层次上，如下图所示。



从 CPU 角度来看，缓存—主存这一层次的速度接近于缓存， 高于主存；其容量和位价却接近于主存。

主存—辅存这一层次，从整体分析，其速度接近于主存，容量接近于辅存，平均位价也接近于低速、廉价的辅存位价，这又解决了速度、容量、成本这三者矛盾。

在主存—辅存这一层次的不断发展中，形成了虚拟存储系统。

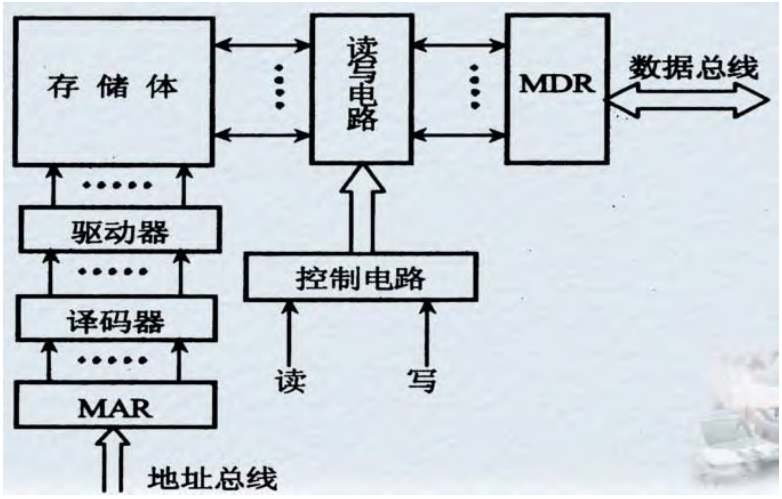
在这个系统中，程序员编程的地址范围与虚拟存储器的地址空间相对应。虚存的地址称为虚地址或逻辑地址，而主存的实际地址称作物理地址或实地址。

具有虚拟存储器的计算机系统，编程时可用的地址空间远远大于主存空间，使程序员以为自己占有一个容量极大的主存，其实这个主存并不存在，因此称其为虚拟存储器。

对虚拟存储器而言，其逻辑地址变换为物理地址的工作，是由计算机系统的硬设备和操作系统自动完成的，对程序员是透明的。

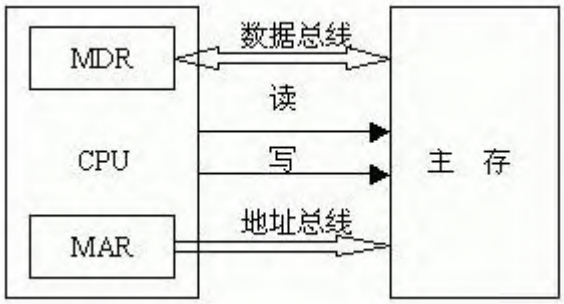
二、主存储器

（一）概述



主存和 CPU 的联系：

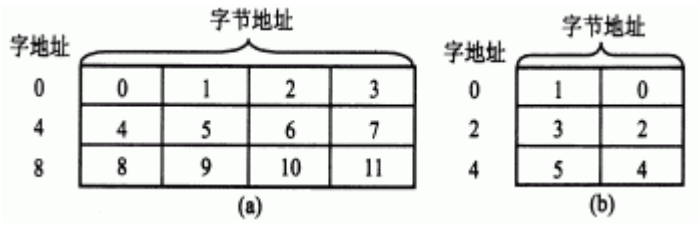
驱动器、译码器和读写电路均制作在存储芯片中，而 MAR 和 MDR 制作在 CPU 芯片内。存储芯片和 CPU 芯片可通过总线连接，如下图所示。



当要从存储器读出某一信息字时，首先由 CPU 将该字的地址送到 MAR，经地址总线送至主存，然后发读命令。主存接到读命令后，得知需将该地址单元的内容读出，便完成读操作，将该单元的内容读至数据总线上，至于该信息由 MDR 送至什么地方，远已不是主存的任务，而是由 CPU 决定的。若要向主存存入一个信息字时，首先 CPU 将该字所在主存单元的地址经 MAR 送到地址总线，并将信息字送入 MDR，然后向主存发写命令，主存按到写命令后，便将数据线上的信息写入到对应地址线指出的主存单元中。

1、主存中存储单元地址的分配

主存各存储单元的空间位置是由单元地址号来表示的，而地址总线是用来指出存储单元地址号的，根据该地址可读出一个存储字。不同的机器存储字长也不同，为了满足字符处理的需要，常用 8 位二进制数表示一个字节，因此存储字长都取 8 的倍数。通常计算机系统既可按字寻址，也可按字节寻址。例如 IBM370 机其字长为 32 位，它可按字节寻址，即它的每一个存储字包含 4 个可独立寻址的字节，其地址分配如下图(a)所示。字地址是用该字高位字节的地址来表示，故其字地址是 4 的整数倍，正好用地址码的末两位来区分同一字的 4 个字节的位置。但对 PDP－11 机而言，其字地址是 2 的整数倍，它用低位字节的地址来表示字地址，如下图(b)所示。



如上图(a)所示，对 24 位地址线的主存而言，按字节寻址的范围是 16MB，按字寻址的范围为 4MB。如上图(b)所示，对 24 位地址线而言，按字节寻址的范围仍为 16MB，但按字寻址的范围为 8MB。

2、主存的技术指标

主存的主要技术指标是存储容量和存储速度。

(1) 存储容量

存储容量是指主存能存放二进制代码的总数，即：

存储容量＝存储单元个数×存储字长

它的容量也可用字节总数来表示，即：

存储容量＝存储单元个数×存储字长 / 8

(2) 存储速度

存储速度是由存取时间和存取周期来表示的。

存取时间又叫存储器的访问时间(Memory Access Time)，它是指启动一次存储器操作(读或写)到完成该操作所需的全部时间。**存取时间分读出时间和写入时间两种**。读出时间是从存储器接收到有效地址开始，到产生有效输出所需的全部时间。写入时间是从存储器接收到有效地址开始，到数据写入被选中单元为止所需的全部时间。

存取周期(Memory Cycle Time)是指存储器进行连续两次独立的存储器操作(如连续两次读操作)所需的最小间隔时间，通常存取周期大于存取时间。

与存取周期密切相关的指标叫**存储器的带宽**，它表示每秒从存储器进出信息的最大数量，单位可用字/秒或字节/秒或位/秒表示。如存取周期为 500ns，每个存取周期可访问 16 位，则它的带宽为 32M 位 / 秒。

存储器的带宽决定了以存储器为中心的机器可以获得的信息传输速度，它是改善机器瓶颈的一个关键因素。

为了提高存储器的带宽，可以采用以下措施：

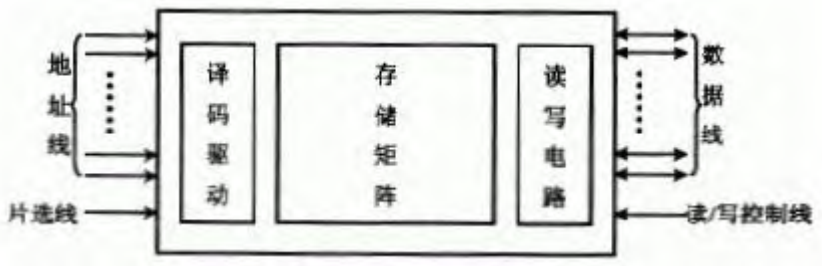
①缩短存取周期；

②增加存储字长，使每个周期访问更多的二进制位；

③增加存储体。

（二）半导体存储芯片简介

1、半导体存储芯片的基本结构



译码驱动能把地址总线送来的地址信号翻译成对应存储单元的选择信号，该信号在读写电路的配合下完成对被选中单元的读写操作。

读写电路包括读出放大器和写入电路，用来完成读写操作。

存储芯片通过地址总线、数据总线和控制总线与外部连接。

地址线是单向输入的，其位数与芯片容量有关。

数据线是双向的(有的芯片可用成对出现的数据线分别作输入或输出)，其位数与芯片可读出或写入的数据位数有关。

地址线和数据线的位数共同反映存储芯片的容量。如地址线为 10 根，数据线为 4 根，则芯片容量为

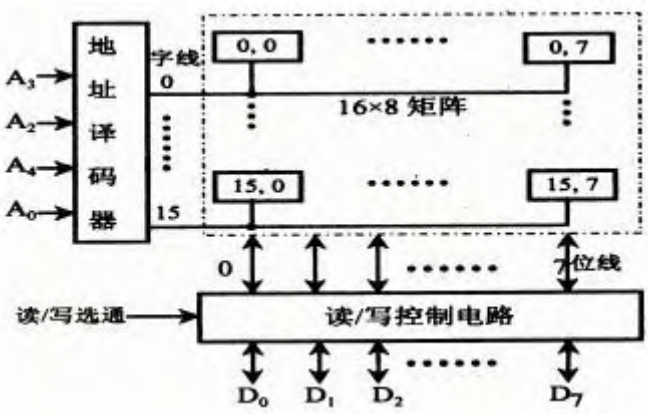
2^{10}×4B＝4KB

控制线主要有读/写控制线与片选线两种。读/写控制线决定芯片进行读/写操作，片选线用来选择存储芯片。由于存储器是由许多芯片组成，需用片选信号来确定哪个芯片被选中。

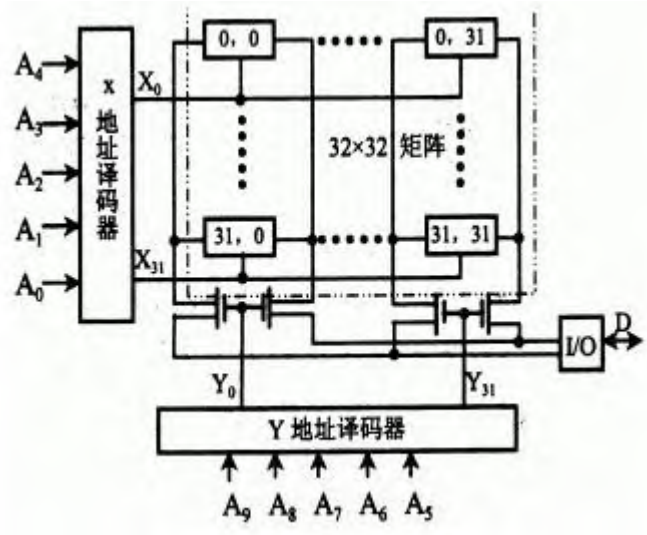
2、半导体存储芯片的译码驱动方式

半导体存储芯片的译码方式有两种：线选法和重合法

(1) 线选法（又称单译码方式）



（2）重合法



（三）随机存取存储器（RAM）

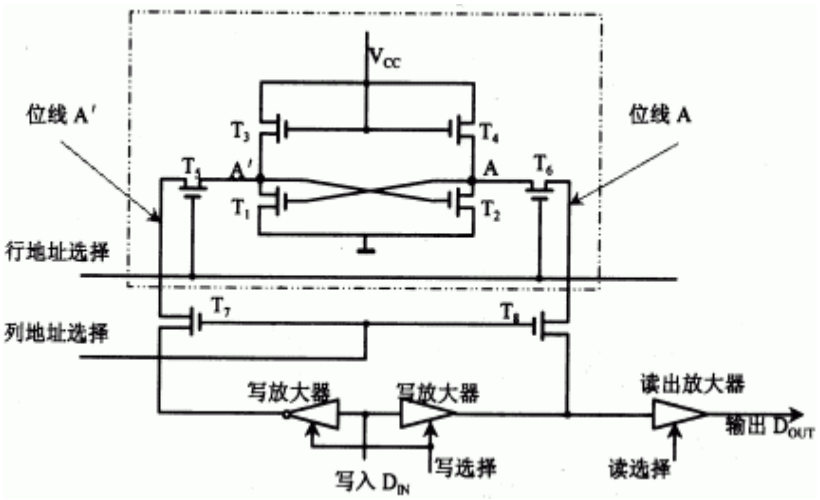
随机存取存储器按其存储信息的原理不同，可分为静态 RAM 和动态 RAM 两大类。

静态 RAM（SRAM） 在不掉电的情况下能够保持住原存信息，不需要再生。但电源掉电时原存信息丢失，故属于易失性存储器。

动态 RAM（DRAM）靠电容存储电荷的原理来寄存信息，在不掉电的情况下原存信息也会丢失，故在原存信息丢失前需要再生。

1、静态 RAM（SRAM）

(1)静态 RAM 基本单元电路。存储器中用于寄存“0”和“1”代码的电路叫做存储器的基本单元电路，下图所示一个 6 个 MOS 管组成的基本单元电路。



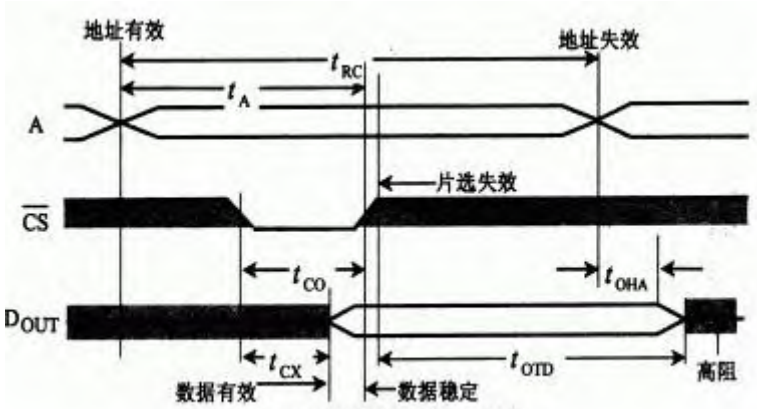
图中 T_1～T_4是一个由 MOS 管组成的触发器基本电路，T_5、T_6尤如一个开关，受行地址选择信号控制。由 T_1～T_6共同构成一个六管 MOS 基本单元电路。T_7、T_8受列地址选择控制，分别与位线 A 和 A′ 相连，它们并不包含在基本单元电路内，而是芯片内同一列的各个基本单元电路所共有的。

假设触发器已存有“1”信号，即 A 点为高电平。当需读出时，只要使行、列地址选择信号均为有效，则使 T_5、T_6，T_7、T_8均导通，A 点高电平通过 T_6后，再由位线 A 通过 T_8作为读出放大器的输入信号，在读选择有效时，将“1”信号读出。

由于静态 RAM 是触发器存储信息，因此即使信息读出后，它仍保持其原状态，不需要再生。但电源掉电时，原存信息丢失，故它属易失性半导体存储器。

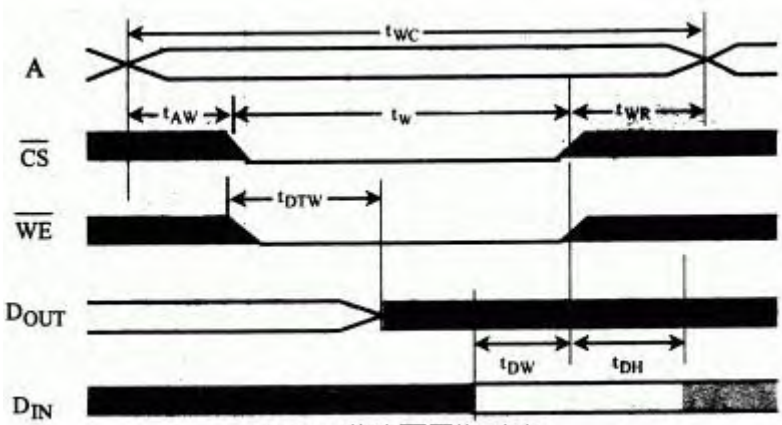
写入时可以不管触发器原状态如何，只要将写入代码送至 D_{IN}端，在写选择有效时，经两个写放大器，使两端输出为相反电平。当行、列地址选择有效时，使 T_5、T_6、T_7、T_8导通，并使 A 与 A′ 点置成完全相反的电平。这样，就把欲写入的信号写入到该单元电路中。如欲写入“1”，即 D_{IN}=1，经两个写放大器使位线 A 为高电平，位线 A′ 为低电平，结果使 A 点为高，A′ 点为低，即写入了“1”信息。

(2)静态 RAM 读写时序



2114RAM芯片读周期时序

上图是 2114RAM 芯片读周期时序，在整个读周期中 $\overline{WE}$始终为高电平(故图中省略)。读周期 t_{RC}是指对芯片进行两次连续读操作的最小间隔时间。读时间 t_A表示从地址有效到数据稳定所需的时间。图中 t_{CO}是从片选有效到输出稳定的时间。可见只有当地址有效经 t_A后，且当片选有效经 t_{CO}后，数据才能稳定输出，这两者必须同时具备。根据 t_A和 t_{CO}的值，便可知当地址有效后，经 t_A—t_{CO}时间必须给出片选有效信号，否则信号不能出现在数据线上。



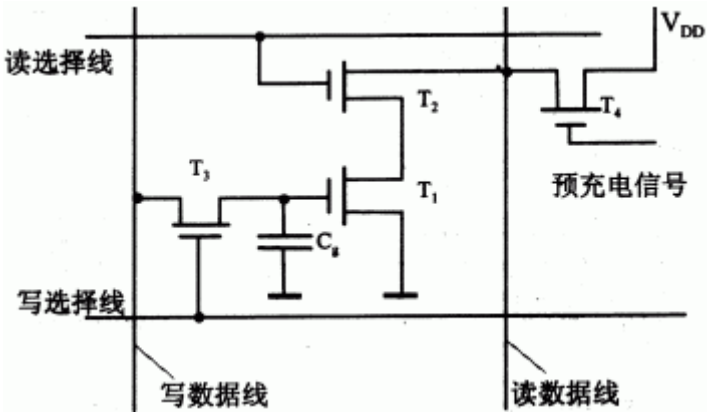
2114RAM芯片写周期时序

上图是 2114RAM 芯片写周期时序。写周期 t_{WC} 是对芯片进行连续两次写操作的最小间隔时间。写周期包括滞后时间 t_{AW}、写入时间 t_W 和写恢复时间 t_{WR}。在有效数据出现前，RAM 的数据线上存在着前一时刻的数据 D_{out}，故在地址线发生变化后，$\overline{CS}$ 、$\overline{WE}$ 均需滞后 t_{AW} 再有效，以避免将无效数据写入到 RAM 的错误。但写允许 失效后，地址必须保持一段时间，叫做写恢复时间。此外，RAM 数据线上的有效数据(即 CPU 送至 RAM 的写入数据 D_{IN})必须在 、 失效前的 t_{DW} 时刻出现，并延续一段时间 t_{DH}(此刻地址线仍有效，t_{WR} ＞t_{DH})， 以保证数据可靠写入。

2、动态 RAM(DRAM)

(1)动态 RAM 的基本单元电路

常见的动态 RAM 基本单元电路有三管式和单管式两种，它们的共同特点都是靠电容存储电荷的原理来寄存信息的。若电容上存有足够多的电荷表示存“1”，电容上无电荷表示存“0”。电容上的电荷一般只能维持 1～2ms，因此即使电源不掉电信息也会自动消失。为此，必须在 2ms 内对其所有存储单元恢复一次原状态，这个过程叫再生或刷新。由于它与静态 RAM 相比，具有集成度更高、功耗更低等特点，因此目前被各类计算机广泛应用。



上图示意了由 T_1、T_2，T_3 三个 MOS 管组成的三管 MOS 动态 RAM 基本单元电路。

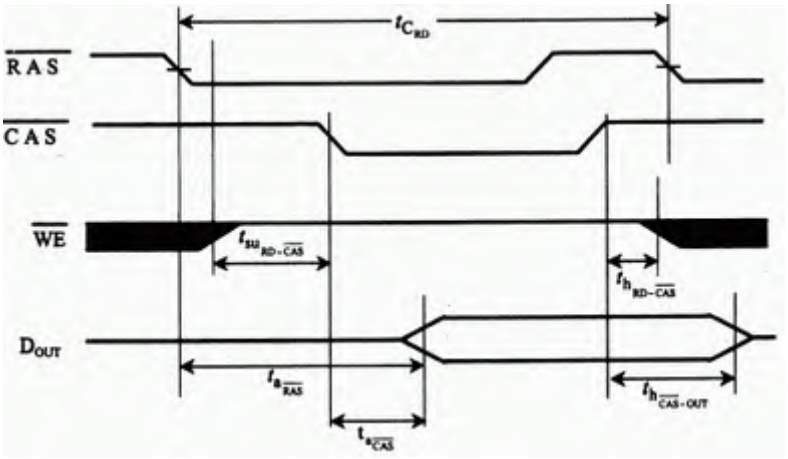
读出时，先对预充电管 T_4 置一预充电信号(在存储矩阵中，每一列共用一个 T_4 管)，使读数据线达高电平 V_{DD}，然后由读选择线打开 T_2，若 T_1 的极间电荷 Cg 存有足够多的电荷(被认为原存“1”)，使 T_1 导通，则因 T_2、T_1 导通接地，使读数据线降为零电平，读出“0”信息。若 C_g 没足够电荷(原存“0”)，则 T_1 截止，读数据线为高电平不变，读出“1”信息。可见，由读出线的高低电平可区分其是读“1”，还是读“0”，只是它与原存信息反相。

写入时，将写入信号加到写数据线上，然后由写选择线打开 T_3，这样，C_g便能随输入信息充电(写“1”)或放电(写“0”)。

(2) 动态 RAM 时序

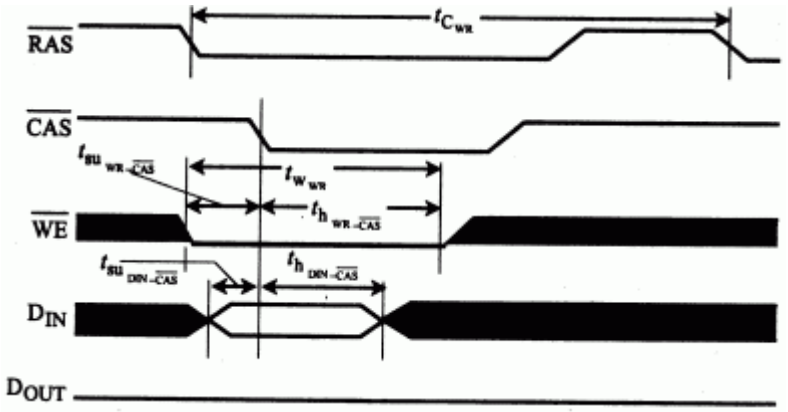
- ✓ 由于动态 RAM 的行、列地址是分别传送的，因此分析其时序时，应特别注意 $\overline{RAS}$ 、$\overline{CAS}$ 与地址的关系。即：
- ✓ 先由将行地址送入行地址缓存器，再由 将列地址送入列地址缓存器，因此， $\overline{CAS}$ 滞后于 $\overline{RAS}$ 的时间必须要超过其规定值。
- ✓ $\overline{RAS}$ 和 $\overline{CAS}$ 正、负电平的宽度应大于规定值，以保证芯片内部正常工作。
- ✓ 行、列地址 $\overline{RAS}$ 和 $\overline{CAS}$ 的下沿(负跳变)应满足有足够的地址建立时间和地址保持时间，以确定行、列地址均能准确写入芯片。

读时序：



在读工作方式时(写允许 $\overline{WE}$ =1)，读工作周期是指动态 RAM 完成一次“读”所需的最短时间 t_{CRD}，也是 $\overline{RAS}$ 一个周期。为了确保读出数据无误，必须要求写允许 $\overline{WE}$ =1 在列地址送入前(即 $\overline{CAS}$ 下沿到来前)建立，而 $\overline{WE}$ =1 的撤除应在 $\overline{CAS}$ 失效后(即 $\overline{CAS}$ 上升沿后)；还要求读出数据应在 $\overline{RAS}$ 有效后一段时间 $t_{k\overline{CAS}-out}$ 且 $\overline{CAS}$ 有效后一段时间 $t_{k\overline{CAS}-out}$ 时出现，而数据有效的撤除时间，应在 $\overline{CAS}$ 失效后一段时间 $t_{k\overline{CAS}-out}$。

写时序：



在写工作方式时(写允许 $\overline{WE}$=0)， $\overline{RAS}$ 的一个周期 t_{CWR} 即为写工作周期，如上图所示。

为了确保写入数据准确无误， $\overline{WE}$=0 应先于 $\overline{RAS}$ =0，而且数据的有效存在时间应与 $\overline{CAS}$ 及 $\overline{WE}$ 的有效相对应。即写入数据应在 $\overline{WE}$ 有效前的一段时间 $t_{k\overline{CAS}-out}$ 出现，它的保持时间应为 $\overline{CAS}$ 有效后的一段时间 $t_{k\overline{CAS}-out}$，这是因为数据的写入实际上是由 $\overline{CAS}$ 的下沿激发而成的。可见，为了保证正常写入， $\overline{WE}$ 、$\overline{CAS}$ 有效均要大于数据 D_{IN} 有效的时间。

(3) 动态 RAM 的刷新

刷新的过程实质上是先将原存信息读出，再由刷新放大器形成原信息并重新写入的再生过程。

由于存储单元被访问是随机的，有可能某些存储单元长期得不到访问，无读出也就无重写，其原信息必然消失。为此，必须采用定时刷新的方法，它规定在一定的时间内，对动态 RAM 的全部基本单元电路必作一次刷新，一般取 2ms，这个时间叫做刷新周期，或叫再生周期。在刷新周期内，由专用的刷新电路对基本单元电路进行刷新。通常有两种方式刷新：

① 集中刷新

集中刷新是在规定的一个刷新周期内，对全部存储单元集中一段时间进行刷新，此刻必须停止读 / 写操作。如 Intel ll03 动态 RAM 芯片内排列成 32×32 矩阵，读/写周期为 0.5ms 连续刷新 32 行需 16μs(占 32 个读 / 写周期)在刷新周期 2ms 内含 4000 个读写周期，实际分配是前 3968 个周期用于读/写操作或维持，后 32 个周期用于刷新，

② 分散刷新

分散刷新是指对每行存储单元的刷新分散到每个读 / 写周期内完成。把存取周期分成两段，前半段用来读写或维持，后半段用来刷新

3、静态 RAM 与动态 ROM 的比较

目前，动态 RAM 的应用比静态 RAM 要**广泛**得多。

其原因是：①在同样大小的芯片中，动态 RAM 的集成度远高于静态 RAM，如动态 RAM 的基本单元电路为一个 MOS 管，静态 RAM 的基本单元电路为 6 个 MOS 管；

②动态 RAM 行、列地址按先后顺序输送，减少了芯片引脚，封装尺寸也减少；

③动态 RAM 的功耗仅为静态 RAM 的 1/6；

④动态 RAM 的价格仅为静态 RAM 的 1/4。因此，随着动态 RAM 容量不断扩大，速度不断提高，它被广泛应用于计算机的主存。

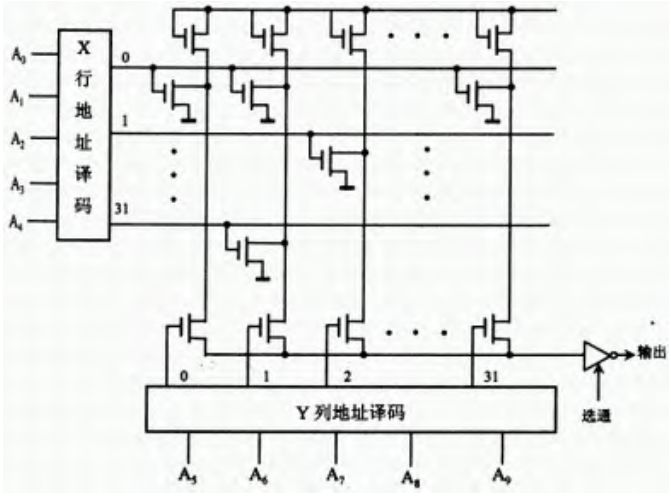
动态 RAM 也有**缺点**：①由于使用动态元件(电容)，因此它的速度比静态 RAM 低；

②动态 RAM 需要再生，故需配置再生电路，也需要消耗一部分功率。通常容量不大的高速存储器大多用静态 RAM 实现。

（四）只读存储器

只读存储器分为掩膜 ROM、PROM、EPROM 和 EEPROM 等多种。对于半导体 ROM，基本器件为两种：MOS 型和 TTL 型。

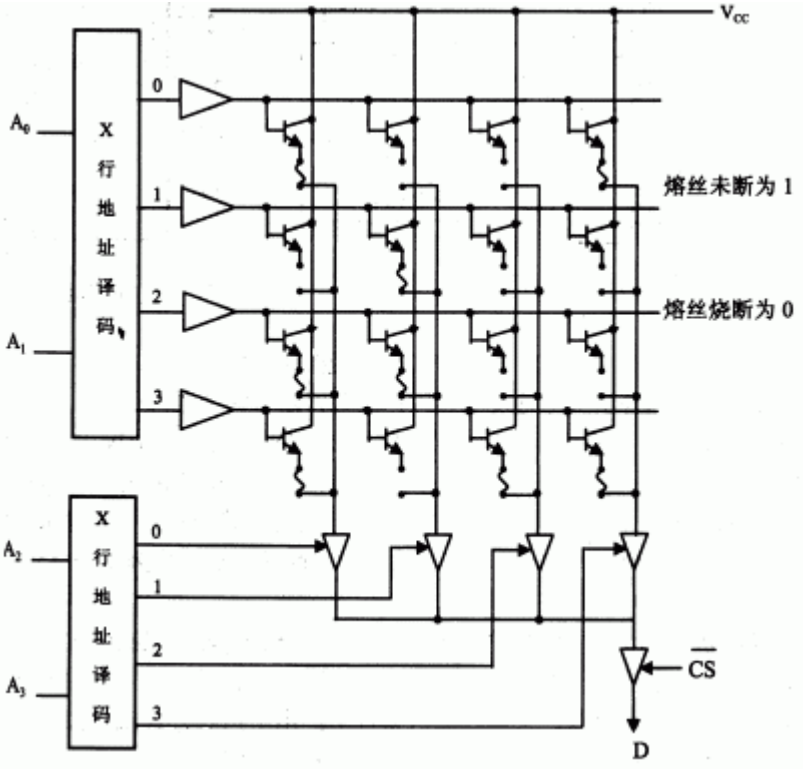
1、掩膜 ROM



上图为 MOS 型掩膜 ROM，其容量为 1K×1 位，采用重合法驱动，行、列地址线分别经行、列译码器，各得 32 根行、列选择线。行选择线与列选择线交叉处既可有耦合元件 MOS 管，也可没有。列选择线各控制一

个列控制管，32 个列控制管的输出端共连一个读放大器；当地址为全“0”时；第 0 行、0 列被选中，若其交叉处有耦合元件 MOS 管，因其导通而使列线输出为低电平，经读放大器反相为高电平，输出“1”。当地址 A_4～A_0 为 11111，A_9～A_5 为 00000 时，即第 31 行、第 0 列被选中，但此刻行、列的交叉处无 MOS 管，故 0 列线输出为高电平，经读放大器反相为“0”输出。可见，用行、列交叉处是否有耦合元件 MOS 管，便可区分原存放“1”还是存“0”。当然，此 ROM 制成后不可能改变原行、列交叉处的是否存在 MOS 管，所以，用户是无法改变原始状态的。

2、PROM



PROM 是可以实现**一次性编程的只读存储器**，上图是 16K×1 位双极型镍铬熔丝式 PROM 芯片，其基本单元电路是由双极型电路和熔丝构成的。

在这个电路中，基极由行选择线控制，发射极与列线之间形成一条镍铬合金薄膜制成的熔丝(可用光刻技术实现)，集电极接电源 V_{CC}。用户在使用前可按需要将信息存入行、列交叉的耦合元件内。若欲存“0”，则置耦合元件一个大电流，将熔丝烧掉。若欲存“1”，则耦合处不置大电流，熔丝不断。当被选中时，熔丝断掉处将读得“0”，熔丝未断处将读得“1”。当然，已断的熔丝是无法再恢复的，故这种 ROM 往往只能实现一次编程，不得再修改。

3、EPROM

EPROM 是一种**可擦洗可编程的只读存储器**。它可以由用户对其所存信息作任意次的改写。目前用得较多的 EPROM 是用浮动栅雪崩注入型 MOS 管构成，又称 FAMOS 型 EPROM

EPROM 的改写可用两种方法，一种用紫外线照射，但擦洗时间比较长，而且不能对个别需改写的单元进行单独擦洗或重写。另一种方法用电气方法将存储内容擦除，再重写。甚至在联机条件下，用字擦除方式或页擦除方式，既可局部擦写，又可全部擦写，这种 EPROM 叫 EEPROM。

还有一种**闪速存储器(Flash Memory)，又叫快擦型存储器**，它是在 EPROM 和 EEPROM 工艺基础上产生的一种新型的、具有性能价格比更好、可靠性更高的可擦写非易失性存储器。它既有 EPROM 的价格便宜、集成度高的优点，又有 EEPOM 电可擦洗重写的特性。

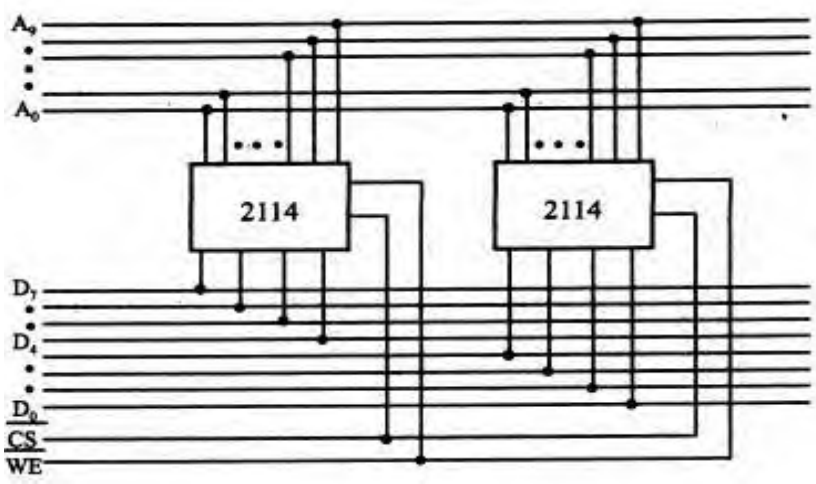
（五）存储器与 CPU 的连接

1、存储容量的扩展

由于单片存储芯片的容量总是有限的，很难满足实际的需要，因此，必须将若干存储芯片连在一起才能组成足够容量的存储器，这就叫存储容量的扩展通常有位扩展和字扩展。

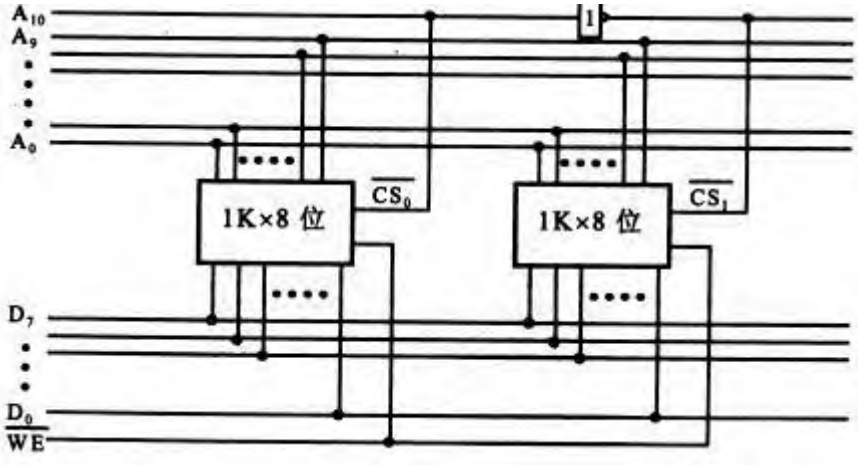
(1)位扩展

位扩展是指增加存储字长，如 2 片 1K×4 位的芯片，可组成 1K×8 位的存储器，如下图所示。图中两片 2114 的地址线 A_9～A_0、$\overline{CS}$、$\overline{WE}$ 分别连在一起，其中一片的数据线作为高 4 位 D_7～D_4，另一片的数据线作为低四位 D_3～D_0。这样，它便构成了一个 1K×8 位的存储器。



(2)字扩展

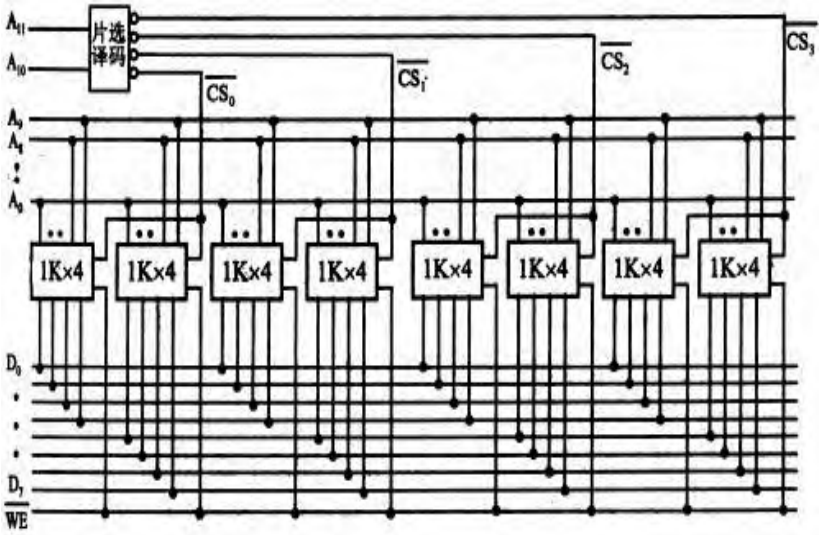
字扩展是指增加存储器字的数量。如用 2 片 1K×8 位的存储芯片，可组成一个 2K×8 位的存储器，即存储字增加了一倍，如下图所示。



在此，将 A_{10} 用作片选信号。由于存储芯片的片选输入端要求低电平有效，故当 A_{10} 为低时，$\overline{CS_0}$ 有效，选中左边的 1K×8 位芯片；当 A_{10} 为高时，反相后 $\overline{CS_1}$ 有效，选中右边的 1K×8 位芯片。

(3)字、位扩展

字、位扩展是指既增加存储字的数量，又增加存储字长。下图示意的是用 8 片 1K×4 位的芯片组成 4K×8 位的存储器。



由图可见，每两片构成 1K×8 位的存储器，4 组两片便构成 4K×8 位的存储器。地址线 A_{11}、A_{10}经片选译码获得 4 个片选信号 $\overline{CS_0}$、$\overline{CS_1}$、$\overline{CS_2}$、$\overline{CS_3}$ 分别选择其中 1K×8 位的存储芯片。$\overline{WE}$ 为读/写控制信号。

2、存储器与 CPU 的连接

存储芯片与 CPU 芯片相连时，特别要注意它们片与片之间的地址线、数据线和控制线的连接。

(1)地址线的连接

存储芯片容量不同，其地址线数也不同，而 CPU 的地址线数往往比存储芯片的地址线数要多。通常总是将 CPU 地址线的低位与存储芯片的地址线相连。CPU 地址线的高位或作存储芯片扩充时用，或作其他用法，如作片选信号等。例如，设 CPU 地址线为 16 位 A_{15}～A_0，1K×4 位的存储芯片仅有 10 根地址线 A_9～A_0，此时，可将 CPU 的低位地址 A_9～A_0与存储芯片地址线 A_9～A_0相连。

(2)数据线的连接

CPU 的数据线数与存储芯片的数据线数也不一定相等。此时，必须对存储芯片扩位，使其数据位数与 CPU 的数据线数相等。

(3)读/写命令线的连接

CPU 读/写命令线一般可直接与存储芯片的读/写控制端相连，通常高电平为读，低电平为写。

(4)片选线的连接

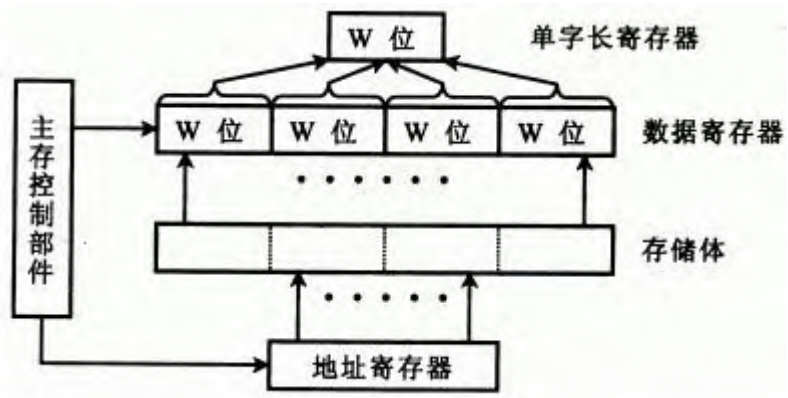
片选信号的连接是 CPU 与存储芯片正确工作的关键。由于存储器是由许多存储芯片叠加组成的，哪一片被选中完全取决于该存储芯片的片选控制端 $\overline{CS}$ 是否能接收到来自 CPU 的片选有效信号。

片选有效信号与 CPU 的访存控制信号 $\overline{MREQ}$ (低电平有效)有关，因为只有当 CPU 要求访存时，才要求选择存储芯片。若 CPU 访问 I/O，则 $\overline{MREQ}$ 为高，表示不要求存储器工作。此外，片选有效信号还和地址有关，因为 CPU 给出的存储单元地址的位数往往大于存储芯片的地址线数，故那些未与存储芯片连上的高位地址必须和访存控制信号共同作用，产生存储器的片选信号。

（六）提高访存速度的措施

1、单体多字系统

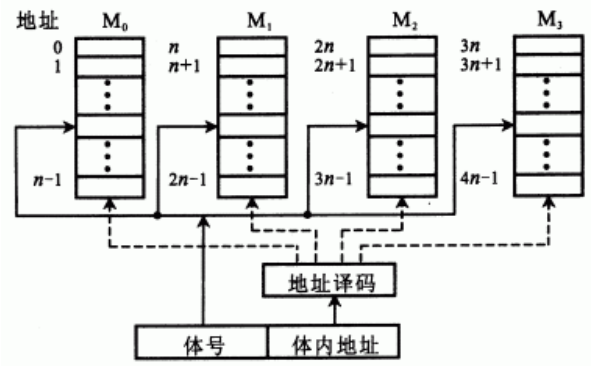
由于程序和数据在存储体内是连续存放的，因此 CPU 访存取出的信息也是连续的，如果可以在一个存取周期内，从同一地址取出 4 条指令，然后再逐条将指令送至 CPU 执行，也即每隔四分之一存取周期，主存向 CPU 送一条指令，这样显然增大了存储器的带宽，提高了单体存储器的工作速度。



上图所示的是一个单体四字结构的存储器，每字 W 位。按地址在一个存取周期内可读出 4×W 位的指令或数据，使主存带宽提高到 4 倍。显然，采用这种办法的前提是：指令和数据在主存内必须是连续存放的，一旦遇到转移指令，或者操作数不能连续存放，这种方法的效果就不明显。

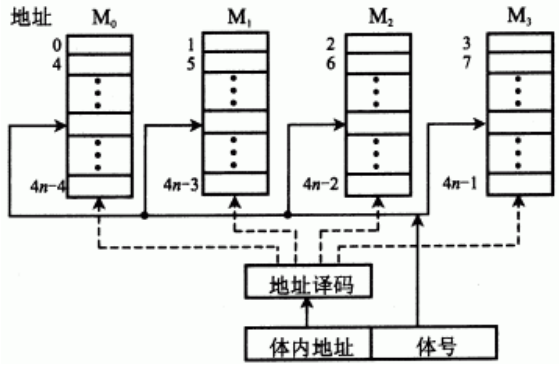
2、多体并行系统

多体并行系统就是采用多体模块组成的存储器。每个模块有相同的容量和存取速度，各模块各自都有独立的地址寄存器、地址译码器、驱动电路和读写电路，它们能并行工作，又能交叉工作。



高位交叉编址的多体存储器

并行工作即同时访问 N 个模块，同时启动， 同时读出，完全并行地工作，上图是适合于并行工作的高位交叉编址的多体存储器结构示意图，图中程序按体内地址存放，一个体存满后，再存入下一个体。显然，高位地址可表示体号。按这种编址方式，只要合理调动，便可提高存储器的带宽。



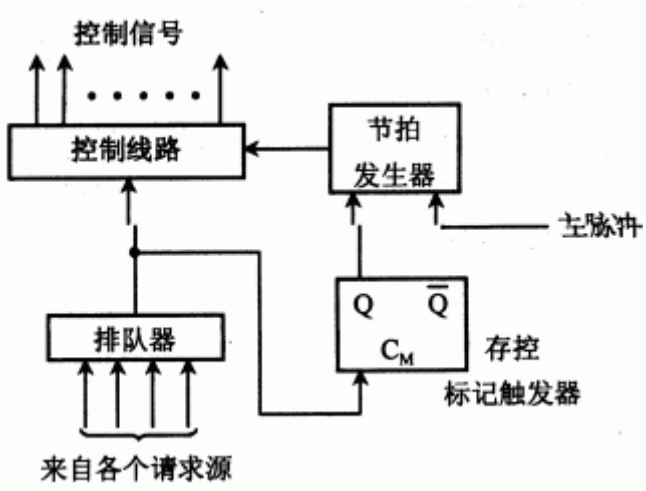
低位交叉编址的多体存储器

在按低位交叉编址的多体模块结构示意中，程序连续存放在相邻体中，显然低位地址用来表示体号，

高位地址为体内地址。这种编址方法又叫作模 M 编址(M 等于模块数)，一般模块数 M 取 2 的方幂，使硬件电路比较简单。有的机器为了减少存储器冲突，采用质数个模块，如我国银河机的 M 为 31，其硬件实现比较复杂。

在某一时刻，决定主存究竟与哪个部件交换信息，必须由存储器控制部件(简称存控)来承担。存控具有合理安排各部件请求访问的顺序以及控制主存读写操作的功能。

下图是一个**存控基本结构框图**，它**由排队器、控制线路、节拍发生器及标记触发器**等组成。



（1）排队器

由于要求访存的请求源很多，而且访问都是随机的，这样有可能在同一时刻出现多个请求源请求访问同一个存储体。为了防止发生两个以上的请求源同时占用同一存储体，并防止将代码错送到另一个请求源等各种错误的发生，在存控内需设置一个排队器，由它来确定请求源的优先级别。其确定原则为：

①对易发生代码丢失的请求源，应列为最高优先级，如外设信息最易丢失，故它的级别最高；

②对严重影响 CPU 工作的请求源，给予次高优先级，否则会导致 CPU 工作失常。

（2）存控标记触发器 CM

它用来接受排队器的输出信号，一旦响应某请求源的请求，CM 被置“1”，以便启动节拍发生器工作

（3）节拍发生器

它用来产生固定节拍，与机器主脉冲同步，使控制线路按一定时序发出信号。

（4）控制线路

由它将排队器给出的信号与节拍发生器提供的节拍信号配合，向存储器各部件发出各种控制信号，用以实现对总线控制及完成存储器读写操作，并向请求源发出回答信号，表示存储器已响应了请求等等。

三、高速缓冲存储器

（一）概述

1、为什么要使用 Cache

原因 1：若 I/O 访存优先级别高于 CPU 访存，I/O 访存时 CPU 空等，如果加上 Cache,可以使 CPU 在空等时访问 Cache。

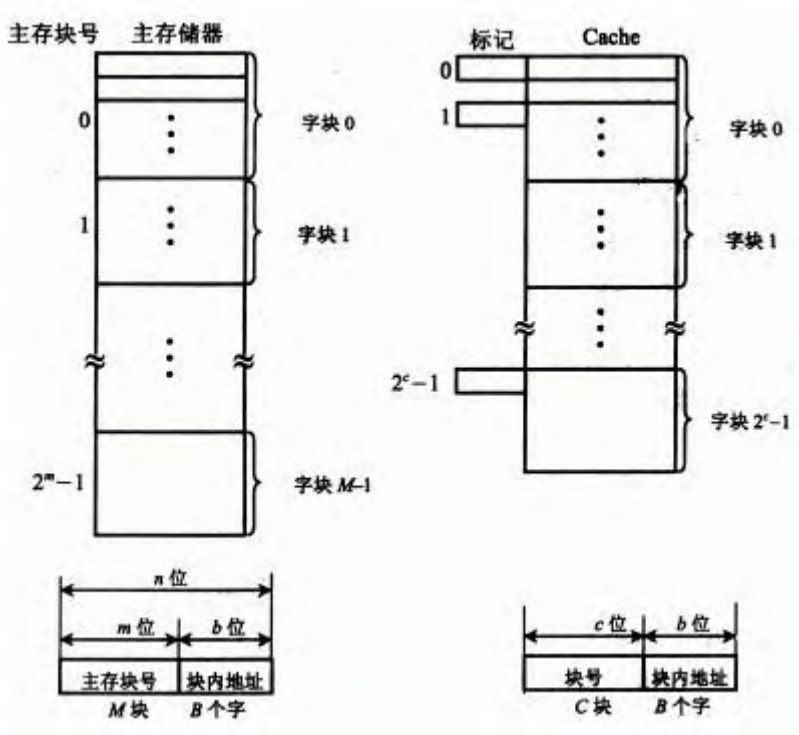
原因 2：主存的速度的提高始终跟不上 CPU 的发展，需要用高速缓存来解决主存与 CPU 的速度不匹配问题。

2、程序访问的局部性原理

指令和数据在主存内都是连续存放的，并且有些指令和数据往往会被多次调用(如子程序循环程序和一些常数)，也即指令和数据在主存的地址分布不是随机的，而是相对的簇聚，使得 CPU 在执行程序时，访存具有相对的局部性，这就叫程序访问的局部性原理。

根据这一原理，很容易设想，只要将 CPU 近期要用到的程序和数据， 提前从主存送到 Cache， 那么就可以做到 CPU 在一定时间内只访问 Cache。

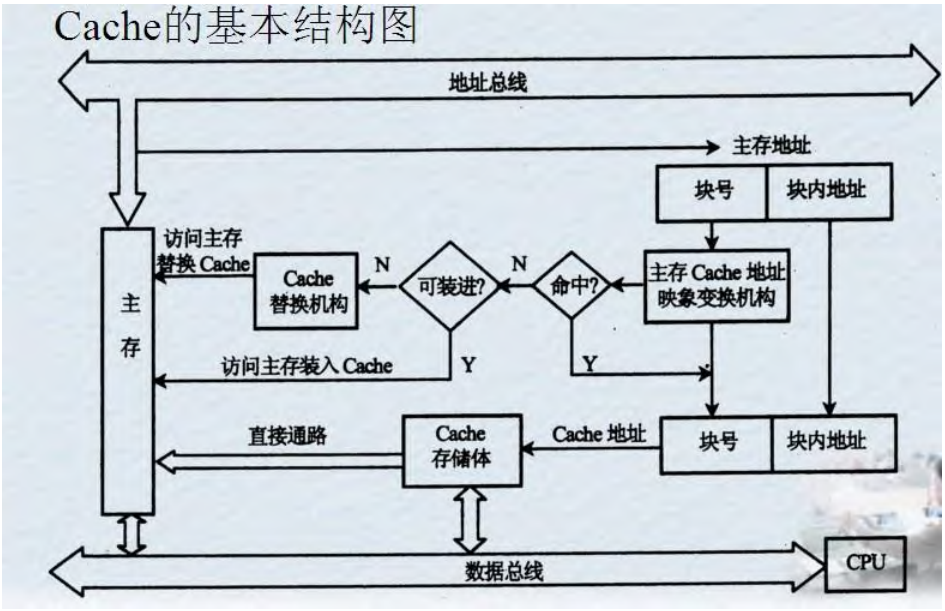
3、Cache 的工作原理



上图是 Cache/主存存储空间的基本结构示意。

- ✓ 主存和 Cache 都分为大小相同的若干块
- ✓ 主存的地址：m 位主存块号+b 位块内地址
- ✓ Cache 的地址：c 位 Cache 块号+b 位块内地址
- ✓ 主存块数 2m=M ， Cache 块数 2c=C , M>C
- ✓ CPU 命中 Cache
- ✓ CPU 不命中 Cache
- ✓ Cache 的容量与块长是影响 Cache 命中率的重要因素

4、Cache 的基本结构



它由 Cache 存储体、地址映象变换机构、Cache 替换机构几大模块组成。

(1)Cache 存储体

Cache 存储体以块为单位与主存交换信息，为加速 Cache 与主存之间的调动，主存大多采用多体结构，且 Cache 访存的优先级最高。

(2)地址映象变换机构

它是将 CPU 送来的主存地址转换为 Cache 地址。由于主存和 Cache 的块大小相同，块内地址都是相对于块的起始地址的偏移量(即低位地址相同)，因此地址变换主要是主存的块号(高位地址)与 Cache 块号间的转换。

如果转换后的 Cache 块已与 CPU 欲访问的主存块建立了对应关系，即命中，则 CPU 可直接访问 Cache 存储体。如果转换后的 Cache 块与 CPU 欲访问的主存块未建立对应关系，即不命中。此刻 CPU 在访问主存时，不仅将该字从主存取出，同时将它所在的主存块一并调入 Cache，供 CPU 使用。当然，此刻能将主存块调入 Cache 内，也是由于 cache 原来处于未被装满的状态。反之，倘若 Cache 原来已被装满，即已无法将主存块调入 Cache 内时，就得采用替换策略。

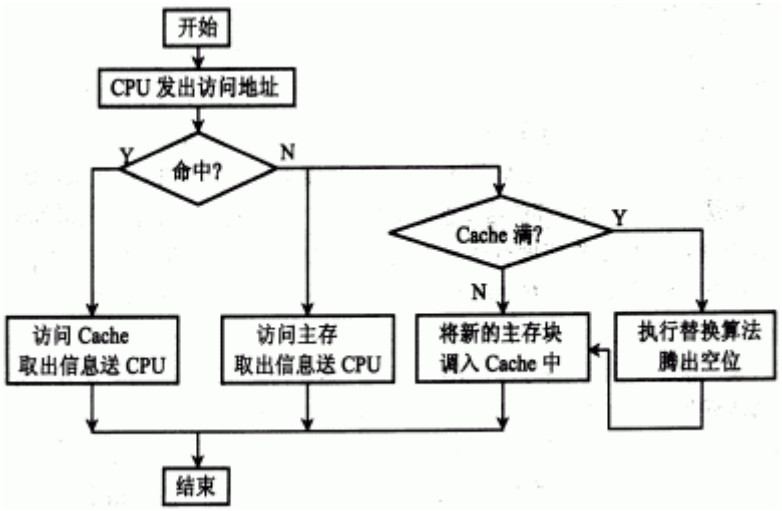
(3)替换机构

当 Cache 内容已满，无法接受来自主存块的信息时，就由 Cache 内的替换机构由按一定的替换算法来确定应从 Cache 内移出哪个块返回主存，而把新的主存块调入 Cache。

特别需指出的是，Cache 对用户是透明的，即用户编程时所用到的地址是主存地址，用户根本不知道这些主存块是否已调入 Cache 内。因为，将主存块调入 Cache 的任务全由机器硬件自动完成。

(4)Cache 的读/写操作

Cache 读操作的过程可用下述流程图来描述。



Cache 写操作比较复杂，目前主要采用以下几种方法：

- ✓ 写直达法，又叫通过式写(Write-through)或叫通过式存(Store-through)，它能随时保证主存与 Cache 的数据始终一致。但有可能会增加访存次数，因每向 Cache 写入时，都需向主存写入。
- ✓ 写回法（Write-back），数据每次只是暂时写入 Cache，并用标志将该块加以注明，直至该块从 Cache 替换出时，才写入主存。这种方法又称标志交换式，其速度快，但因主存中的字块未经随时修改，可能失效。
- ✓ 信息只写入主存，同时将相应的 Cache 块有效位置“0”，表明此 Cache 块已失效，需要时从主存调入。还有一种可能，被修改的单元根本不在 Cache 内，因此写操作只对主存进行。

5、Cache 的改进

(1) 单一缓存和两级缓存

单一缓存即在 CPU 和主存之间只设一个缓存。

由片外缓存和片内缓存组成的 Cache，叫做两级缓存，并称片内缓存为第一级，片外缓存为第二级。

(2) 统一缓存和分开缓存

统一缓存是指指令和数据都存放在同一缓存内的 Cache；分开缓存是指指令和数据分别存放在两个缓存中，一个叫指令 Cache，一个叫数据 Cache。两种缓存的选用主要考虑如下两个因素。

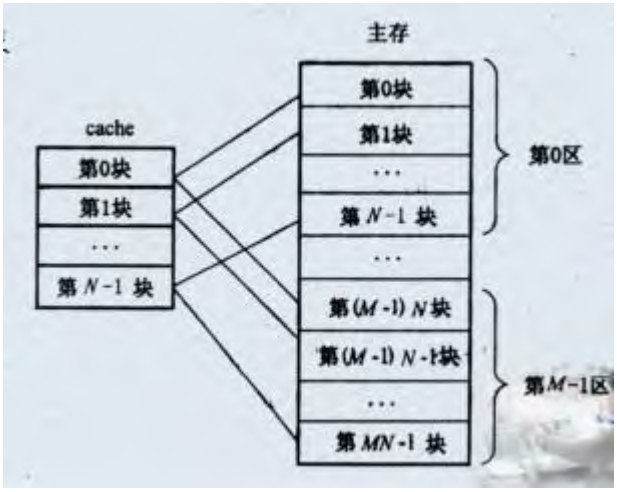
其一，它与主存结构有关，如果计算机的主存是统一的(指令、数据存在同一主存内)，则相应的 Cache 就采用统一缓存；如果主存采用指令、数据分开存放的方案：则相应的 Cache 就采用分开缓存。

其二，它与机器对指令执行的控制方式有关。当采用超前控制或流水线控制方式时，一般都采用分开缓存。

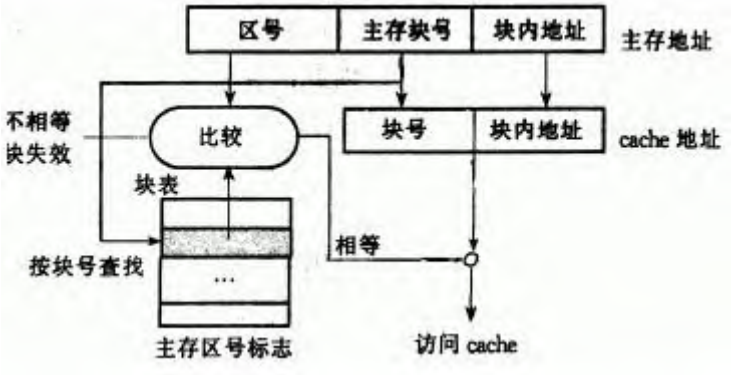
所谓**超前控制**是指在当前指令执行过程尚未结束时，就提前将下一条准备执行的指令取出，即超前取指或叫指令预取。所谓**流水线控制**实质上是多条指令同时执行，又可视为指令流水。

（二）Cache——主存地址映象

1、直接映像



一个主存块只能映象到 cache 中的惟一一个指定块的地址映象方式称为**直接映象**。

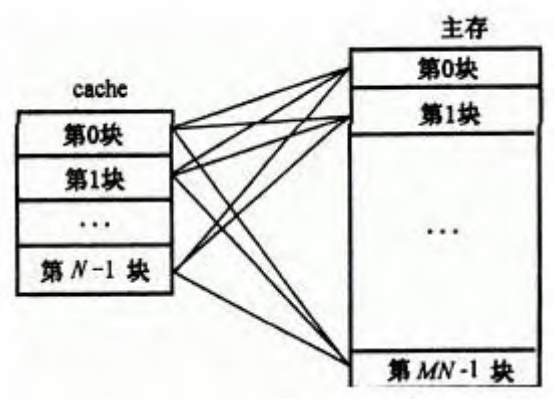


实现地址转换的过程如上图所示，其中地址映象用的块表中包含 Cache 存储器各块的区号，Cache 地址的块内地址与主存地址的块内地址部分相同，块号也相同。在访存操作时，根据地址中的块号读出表中的区号并与当前地址的区号段进行比较，比较结果相同表示 Cahe 命中，访问可对 Cache 进行；比较结果不相同则表示不命中，访问需要对主存进行。这时在对主存进行访问并将主存中的块调人 Cache 中的同时将区号段写入块表中，这就完成了地址映象关系的改变。在新的数据块调入时，Cache 中的原数据块被替换。从主存读人的数据可以先替换原数据然后再从 Cache 送到 CPU，也可以在替换 Cache 原数据块时直接送到 CPU。

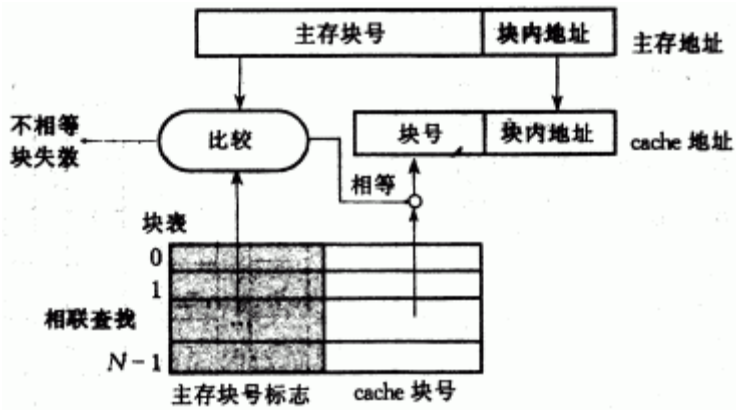
直接映象是一种最简单的地址映象方式，它的地址变换速度快，而且不涉及其他两种映象方法中的替换策略问题。**缺点**是不够灵活，因每个主存块只能固定地对应某个缓存块，即使缓存内还空着许多位置也不能占用，使缓存的存储空间得不到充分的利用。此外，如果程序恰好要重复访问对应同一缓存位置的不同主存块，就要不停地进行替换，从而降低了命中率。

2、全相联映像

每个主存块都可映象到任何 Cache 块的地址映象方式称为**全相联映象**，如下图所示。



在全相联映象方式下，主存中存储块的数据可调入 Cache 中的任意块框架，如果 Cache 中能容纳程序所需的绝大部分指令和数据，则可达到很高的 Cache 命中率。但全相联映象 Cache 的实现比较复杂。当访问一个块中的数据时，块地址要同时与块表中的所有地址标志进行比较以确定是否命中。在数据块调入时，还存在着一个比较复杂的替换策略问题，即决定将数据块调入 Cache 中什么位置，将 Cache 中哪一块数据调出。



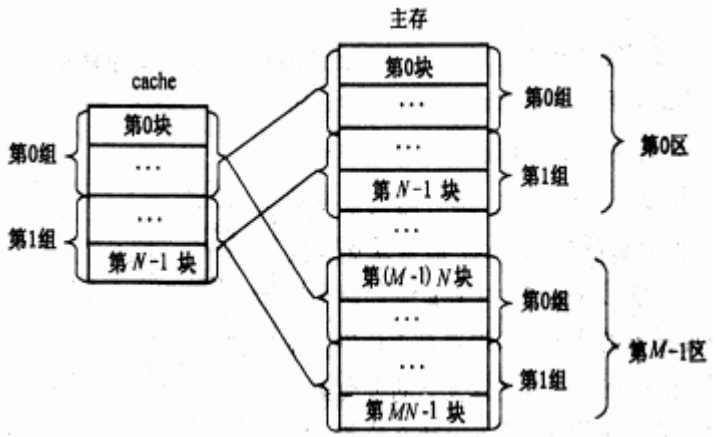
采用全相联映象方式后，地址变换方式如上图所示。Cache 地址中，块内地址部分直接取自主存地址的块内地址段，Cache 块号则根据主存从块表中查到。块表中包含 Cache 存储器各块的主存块号以及对应的 Cache 块号，在访存操作时，根据地址中的块号在块表中查找是否有相同的主存块号。如果有相同的，则表示 Cache 命中，将对应的 Cache 块号取出以对 Cache 进行访问，没有相同的则表示不命中，在对主存进行访问并将主存中的块调入 Cache 中的同时将主存块号和 Cache 块号写入块表中，以改变地址映象关系。查找地址映象表时需要查找表中的每个项，全部查完后才能确定 Cache 不命中。在新的数据块调入时，还需确定将 Cache 中的哪个数据块替换出去。图中块表中的阴影区域表示块表查找的范围。

特点：

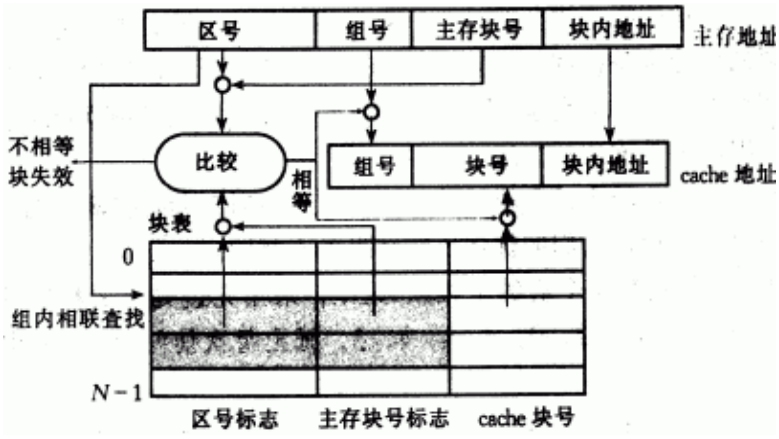
- ✓ 全相联方法在 Cache 中的块全部装满后才会出现块冲突，而且可以灵活地进行块的分配，所以块冲突的概率低，Cache 的利用率高。
- ✓ 全相联 Cache 中块表查找的速度慢，控制复杂，需要一个用硬件实现的替换策略，实现起来比较困难。

3、组相联映象

组相联映象指的是将存储空间分成若干组，**各组之间是直接映象，而组内各块之间则是全相联映象。**如下图所示，在组相联映象方式下，主存也按 Cache 的容量分区，每个分区又分成若干个组，每个组包含若干个块，Cache 也进行同样的分组。主存中存储块的数据块可调入 Cache 中一个指定组内的任意块框架中，但主存中一个组的地址空间只能映象到 Cache 中相同的组中，也就是说组内是全相联映象，组间则是直接映象。组相联映象可以看做是上述两种地址映象方式的一般形式，如果组的容量为 1 个块时就变成了直接映象；如果组的容量变成了整个 Cache 的容量(也就是一个区的容量)时就变成了全相联映象。



在组相联映象中，组的个数一般为 2 的幂次数，组内块的个数也是 2 的幂次。主存地址分成四段，高字段是区号；然后是组标志，用于确定组号；第三段是组中的块地址，用于确定组中的块；低字段是块内寻址段。Cache 地址分三段：组号、组内块号和块内地址。组相联方法在判断块命中以及替换算法上都要比全相联方法简单，块冲突的概率比直接映象的低，其命中率也介于直接映象和全相联映象方法之间。



组相联映象的地址变换方式如上图所示，其中 cache 地址中的块内地址部分直接取自主存地址的块内地址段，组号部分也直接取自主存地址(因为组间是直接映象)，组内的块号部分则是查找块表的结果。块表中包含 Cache 存储器各块的主存区号；组内块号以及对应的 cache 组内块号。在访存操作时，根据地址中的组号和块号在块表中的该组对应的若干项中查找是否有相同的主存区号和组内块号。如果有相同的，则表示 Cache 命中，将对应的 Cache 组内块号取出以对 Cache 进行访问，没有相同的则表示不命中，在对主存进行访问并将主存中的块调入 Cache 中的同时将主存区号和组内块号和 Cache 的组内块号写入块表中，以改变地址映象关系。在新的数据块调入时，还需确定将组内的哪一个数据块替换出去。为了提高查块表和比较的速度，可以将一组的表项同时读出，分别与主存地址进行比较。

一般在容量小的 Cache 中可采用组相联映象或全相联映象方法，而在容量大的 Cache 中则可以采用直接映象的 Cache。在速度要求较高的场合采用直接映象，而在速度要求较低的场合采用组相联或全相联映象。

4、段相联映象

段相联映象是直接映象和全相联映象两者结合的又一种方式。它是将主存和 Cache 都分成若干段，且使它们每段包含的块数都相等，段之间采用全相联象，段内块之间采用直接映象。当段数与 Cache 块数相等(即每段只包含一块)时，便为全相联映象，当段数为 1 时，便为直接映象。

（三）替换算法

当新的主存块需要调入 Cache 并且它的可用空间位置又被占满时，就产生了一个替换算法(策略)问题。目前，常用的两种算法是：**先进先出(FIFO)算法和近期最少使用(LRU)算法。**

1、先进先出(FIFO)算法

FIFO 算法的原则总是将最先调入 Cache 的字块替换出来，它不需要随时记录各字块的使用情况，所以容易实现、开销小。但其缺点是可能把一些需要经常使用的程序(如循环程序)块也作为最早进入 Cache 的块而被替换出去。

2、近期最少使用(LRU)算法

LRU 算法是将近期最少使用的块替换出来。它需要随时记录 Cache 中各个字块的使用情况，以便确定哪个字块是近期最少使用的字块。LRU 算法的平均命中率比 FIFO 高，尤其是当分组容量加大时(组相联映象)更能提高 LRU 算法的命中率。

四、虚拟存储器

虚拟存储器主要用于解决计算机中主存储器的容量不足的问题，要求在不明显降低平均访存速度的前提下增加程序的访存空间。

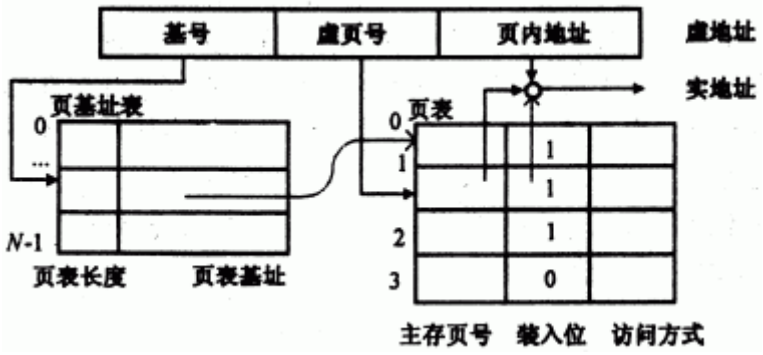
（一）页式虚拟存储器

页式虚拟存储器是把虚拟存储空间和实际存储空间等分成固定容量的页，各虚拟页可装入主存中不同的实际页面位置。

在页式虚拟存储器中，程序中的**逻辑地址**由**基号、虚页号和页内地址三部分**组成；**实际地址**分为页号和页内地址两部分，地址映象机构将虚页号转换成主存的实页号。

基号是操作系统给每个程序产生的地址附加的地址字段，以便于区分不同程序的地址空间。在任一时刻，每个虚拟地址都对应一个实际地址，这个实际地址可能在内存中，也可能在外存中。这种把存储空间按页分配的存储管理方式称为**页式管理**。页式管理用一个页表，其中包括每个页的主存页号、表示该页是否已装入主存的装入位等。**虚页号**一般对应于该页在页表中的行号。页的长度是固定的，因此不需要在页表中记录。**页表**是虚拟页号(或称逻辑负号)与物理页号的映象表。

在页式地址转换过程中，**在进行地址映象时**，首先根据基号查找页基址表， 页基址表一般是 CPU 中的专门寄存器组，其中每一行代表一个运行的程序的页表信息，包括页表起始地址和页表长度。从页基址表中查出页表的起始地址，然后用虚页号从页表中查找实页号，同时判断该页是否装人内存。如果该页已装入内存，则从页表中取出实页号，与页内地址一起构成物理地址。其地址映象方法如下图所示。

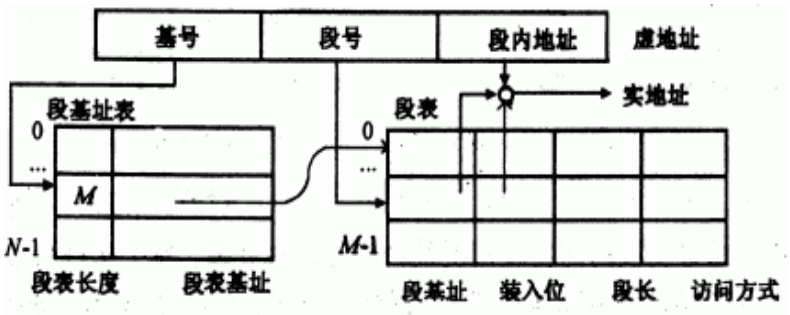


页式管理在存储空间较大时，由于页表过大，工作效率将降低。当页面数量很多时，页表本身占用的存储空间将很大，对这样的页表可能又要分页管理了；为了解决这个问题，人们提出了段式虚拟存储器的概念。

（二）段式虚拟存储器

把主存按段分配的存储管理方式称为段式管理，采用段式管理的虚拟存储器称为段式虚拟存储器。段的长度可以任意设定，并可以放大和缩小。段式管理是一种模块化的存储管理方式，在段式管理的系统中，操作系统给每一个运行的用户程序分配一个或几个段，每个运行的程序只能访问分配给该程序的段对应的主存空间，每个程序都以段内地址访问存储器，即每个程序都按各自的虚拟地址访存。系统运行时，每个程序都有一个段标识符，不同的程序中的地址被映象到不同的段中，因此也可以将段标识符作为虚地址的最高位段，即基号。在段式虚拟存储器中，程序中的逻辑地址由基号、段号和段内地址三部分组成。

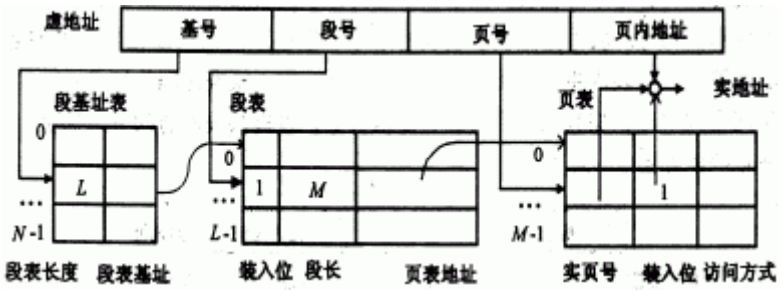
段式虚拟存储器的地址映象如下图所示：



在进行地址映象时，首先根据基号查找段基址表，段基址表一般也是 CPU 中的专门寄存器组。 从表中查出段表的起始地址，然后用段号从段表中查找该段在内存中的起始地址，向时判断该段是否装入内存。如果该段已装入内存，则从段表中取出段起始地址，与段内地址相加构成被访问数据的物理地址。段表本身也存放在一个段中，一般常驻主存。因为段的长度是可变的，所以必须将段长信息存储在段表中，一般段长都有一个上限。分段方法能使大程序分模块编制，独立运行，容易以段为单位实现存储保护和数据共享。

（三）段页式虚拟存储器

段式管理和页式管理各有其优点和缺点，段页式管理是两者的结合。它将存储空间按逻辑模块分成段，每段又分成若干个页。这种访存通过一个段表和若干个页表进行。段的长度必须是页长的整数倍，段的起点必须是某一页的起点。在段页式虚拟存储器中，虚拟地址被分为基号、段号、页号和页内地址四个字段。目前大多数计算机系统都采用段页式管理。在进行地址映象时，首先根据基号查找段基址表，从表中查出段表的起始地址，接着用段号从段表中查找该段的页表的起始地址，然后根据段内页号在页表中查找该页在内存中的起始地址，即实页号，同时判断该段是否装入内存。如果该段已装人内存，则从段表中取出实页号，与页内地址字段拼接构成被访问数据的物理地址。段页式虚拟存储器地址映象方法如下图所示。



段页式管理在地址变换时需要查两次表，即段表和页表。每个运行的程序通过一个段表和相应的一组页表建立虚拟地址与物理地址的映象关系。段表中的每一项对应一个段，其中的装入位表示该段的页表是否已装入主存。若已装入主存，则地址项指出该段的页表在主存中的起始地址，段长项指示该段页表的行数。页表中还包含装入位、主存页号等信息。

五、辅助存储器

（一）概述

1、特点：

容量大、速度慢、价格低、可脱机保存信息 ，属“非易失性”存储器 。

计算机系统的辅助存储器有硬磁盘、软磁盘、磁带、光盘等。前三种均属磁表面存储器。

磁表面存储器是在不同形状(如盘状、带状等)的载体上，涂有磁性材料层，工作时，靠载磁体高速运动，由磁头在磁层上进行读写操作，信息被记录在磁层上。

2、磁表面存储器的主要技术指标

(1)记录密度

记录密度通常是指单位长度内所存储的二进制信息量。磁盘存储器用道密度和位密度表示；磁带存储器则用位密度表示。磁盘沿半径方向单位长度的磁道数为**道密度**，单位是道/英寸(TPI)或道/毫米(TPM)。为了避免干扰，磁道与磁道之间需保持一定距离，相邻两条磁道中心线之间的距离叫道距，因此道密度 D_t 等于道距 P 的倒数。即

$$D_t = \frac{1}{P}$$

单位长度磁道能记录二进制信息的位数，称为**位密度或线密度**，单位是 bpi (bits per inch)或 bpm(位/毫米)。磁带存储器主要用位密度来衡量，常用的磁带有 800bpi，1600bpi，6250bpi 等。对于磁盘，位密度 D_b 可按下式计算：

$$D_b = \frac{f_t}{\pi \cdot d_{\min}}$$

f_t 为每道总位数，$d_{\min}$ 为同心圆中最小直径，各磁道上所记录的信息量是相同的而位密度不同，一般的磁盘位密度是指最内圈磁道的位密度，即最大位密度。

(2)存储容量

存储容量是指外存所能存储的二进制信息总数量，一般以位或字节为单位。以磁盘存储器为例，存储容量可按下式计算：

$$C = n \times k \times s$$

其中 C 为存储总容量，n 为存放信息的盘面数，k 为每个盘面的磁道数，s 为每条磁道上记录的二进制代码数。

磁盘有格式化容量和非格式化容量两个指标，非格式化容量是磁表面可以利用的磁化单元总数。格式化容量是指按某种特定的记录格式所能存储信息的总量，即用户可以使用的容量，它一般为非格式化容量的 60%～70%。

(3)平均寻址时间

由存取方式分类可知，磁盘采取直接存取方式，寻址时间分为两个部分，其一是磁头寻找目标磁道的找道时间 t_s，其二是找到磁道后，磁头等待欲读写的磁道区段旋转到磁头下方所需要的等待时间 t_w。由于从最外圈磁道找到最里圈磁道和寻找相邻磁道所需时间是不等的，而且磁头等待不同区段所花的时间也不等。因此，取其平均值，称作**平均寻址时间** T_a，它是平均找道时间 t_{sa} 和平均等待时间 t_{wa} 之和：

$$T_a = t_{sa} + t_{wa} = \frac{t_{s\max} + t_{s\min}}{2} + \frac{t_{w\max} + t_{w\min}}{2}$$

平均寻址时间是磁盘存储器的一个重要指标。硬磁盘的平均寻址时间比软磁盘的平均寻址时间短，所以硬磁盘存储器比软磁盘存储器速度快。

磁带存储器采取顺序存取方式，磁头不动，磁带移动，不需要寻找磁道，但要考虑磁头寻找记录区段的等待时间，所以磁带寻址时间是指磁带空转到磁头应访问的记录区段所在位置的时间。

(4)数据传输率

数据传输率 D_r 是指单位时间内磁表面存储器向主机传送数据的位数或字节数，它与记录密度 D 和记录介质的运动速度 V 有关：

$$D_r = D \times V$$

此外，辅存和主机的接口逻辑应有足够快的传送速度，用来完成接收/发送信息，以利主机与辅存之间的传送正确无误。

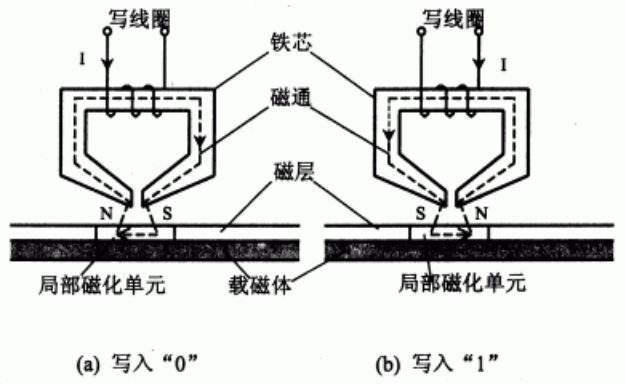
(5)误码率

误码率是衡量磁表面存储器出错概率的参数，它等于从辅存读出时，出错信息位数和读出的总信息位数之比。为了减少出错串，磁表面存储器通常采用循环冗余码来发现并纠正错误。

（二）磁记录的基本原理和记录方式

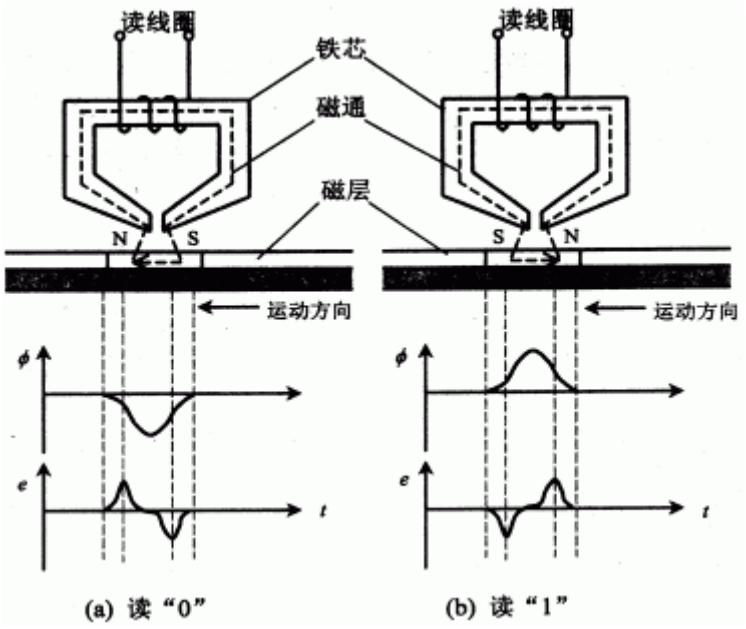
1、磁记录原理

磁表面存储器通过磁头和记录介质的相对运动完成读写操作。写入时，记录介质在磁头下方匀速通过，根据写入代码的要求，对写入线圈输入一定方向和大小的电流，使磁头导磁体磁化，产生一定方向和强度的磁场。由于磁头与磁层表面间距非常小，磁力线直接穿透到磁层表面，将对应磁头下方的微小区域磁化(叫作磁化单元)。可以根据写入驱动电流的不同方向，使磁层表面被磁化的极性方向不同，以区别记录“0”或“1”。



磁表面存储器写入原理

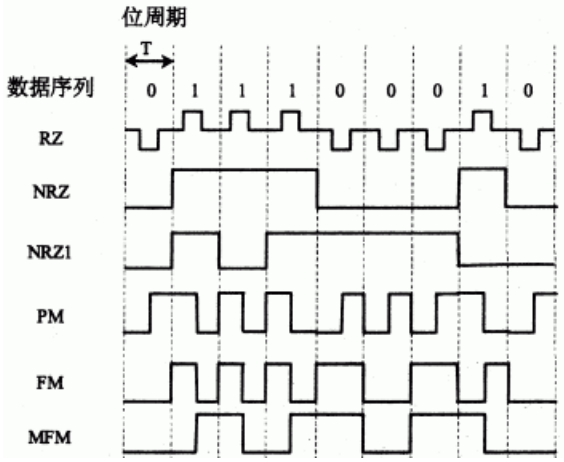
读出时，记录介质在磁头下方匀速通过，磁头相对于一个个被读出的磁化单元作切割磁力线的运动，从而在磁头读线圈中产生感应电势 e，且 $e=-n\frac{d\phi}{dt}$（n 为读出线圈匝数），其方向正好和磁通的变化方向相反。由于原来磁化单元的剩磁通的方向 ϕ 不同，感应电势方向也不同，便可读出“1”或“0”两种不同信息。如下图所示。



2、磁表面存储器的记录方式

磁记录方式又称为编码方式，它是按某种规律，将一串二进制数字信息变换成磁表面相应的磁化状态。磁记录方式对记录密度和可靠性都有很大影响，

常用的记录方式有 6 种，如下图所示。



图中波形既代表了磁头线圈中的写入电流波形，也代表磁层上相应位置所记录的理想的磁通变化状态。

(1)归零制(RZ)

归零制记录“1”时，通以正向脉冲电流，记录“0”时，通以反向脉冲电流，使其在磁表面形成两个不同极性的磁饱和状态，分别表示“1”和“0”。由于两位信息之间驱动电流归零，故叫归零制记录方式。这种方式在写入信息时很难覆盖原来的磁化区域，所以为了重新写入信息，在写入前，必须先抹去原存信息。这种记录方式原理简单，实施方便，但由于两个脉冲之间有一段间隔没有电流，相应的该段磁介质未被磁化，即该段空白，故记录密度不高，目前很少使用。

(2)不归零制(NRZ)

不归零制记录信息时，磁头线圈始终有驱动电流，不是正向，便是反向，不存在无电流状态。这样，磁表面层不是正向被磁化，就是反向被磁化。当连续记录“1”或“0”时，其写电流方向不变，只有当相邻两信息代码不同时，写电流才改变方向，故称为“见变就翻”的不归零制。

(3)见“1”就翻的不归零制(NRZ1)

见“1”就翻的不归零制在记录信息时，磁头线圈也始终有电流。但只有在记录“0”时电流改变方向，使磁层磁化方向发生翻转；记录“1”时，电流方向保持不变，使磁层的磁化方向也维持原来状态，这就叫见“1”就翻的不归零制。

(4)调相制(PM)

调相制又称为相位编码(PE)。其记录规则是：记录“1”时，写电流由负变正：记录“0”时，写电流由正变负，而且电流变化出现在一位信息记录时间的中间时刻，它以相位差为180°的磁化翻转方向来表示“1”和“0”。因此，当连续记录相同信息时，在每两个相同信息的交界处，电流方向都要变化一次，若相邻信息不同，则两个信息位的交界处电流方向维持不变。调相制在磁带存储器中用得较多。

(5)调频制(FM)

调频制的记录规则是：以驱动电流变化的频率不同来区别记录“1”还是“0”。当记录“0”时，在一位信息的记录时间内电流保持不变；当记录“1”时，在一位信息记录时间的中间时刻，使电流改变一次方向。而且无论记录“0”还是“1”，在相邻信息的交界处，线圈电流均变化一次。因此，写“1”时，在位单元的起始和中间位置，都有磁通翻转；在写“0”时，仅在位单元起始位置有翻转。显然，记录“1”的磁翻转频率为记录“0”的两倍，故又称为倍频制。调频制记录方式被广泛应用在硬磁盘和软磁盘中。

(6)改进调频制(MFM)

这种记录方式基本上同调频制，即记录“0”时，在位记录时间内电流不变；记录“1”时，在位记录时间的中间时刻电流发生一次变化。两者不同之处在于，改进调频制只有当连续记录两个或两个以上的“0”时，才在每位的起始处电流改变一次，不必在每个位起始处都改变电流方向。由于这一特点，在写入同样数据序列时，MFM 比 FM 磁翻转次数少，在相同长度的磁层上可记录的信息量将会增加，从而提高了磁记录密度。FM 制记录一位二进制代码最多是两次磁翻转，MFM 制最多只要一次翻转，记录密度提高了一倍，故又称之为倍密度记录方式。倍密度软磁盘即采用 MFM 记录方式。

3．评价记录方式的主要指标

评价一种记录方式的优劣标准，主要反映在编码效率和自同步能力等方面。

（1）编码效率

编码效率是指位密度与磁化翻转密度的比值，可用记录一位信息的最大磁化翻转次数来表示。如 FM、PM 记录方式中，记录一位信息最大磁化翻转次数为 2，因此编码效率为 50%；而 MFM、NRZ、NRZ1 三种记录方式的编码效率为 100%，因为它们记录一位信息磁化翻转最多一次。

(2) 自同步能力

自同步能力是指从单个磁道读出的脉冲序列中所提取同步时钟脉冲的难易程度。从磁表面存储器的读出可知，为了将数据信息分离出来，必须有时间基准信号，称为同步信号。同步信号可以从专门设置用来记录同步信号的磁道中取得，这种方法叫做外同步，如 NRZl 制就是采用外同步的。

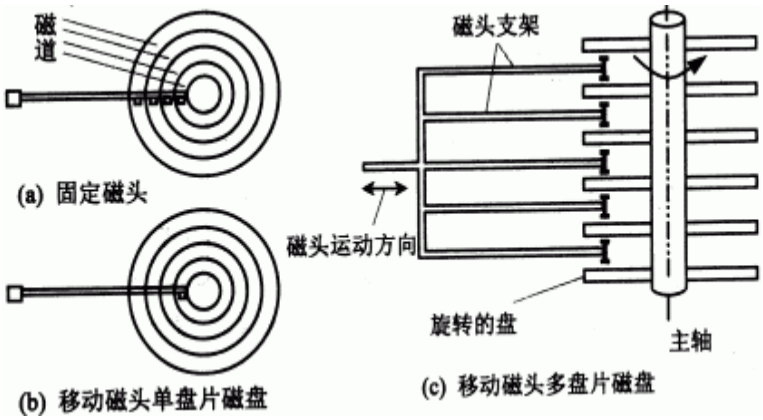
（三）硬磁盘存储器

1、硬磁盘存储器类型

硬磁盘存储器的盘片由硬质铝合金材料制成，表面涂有一层可被磁化的硬磁特性材料。它可以按以下几种方式分类：

（1）按磁头的工作方式分

硬磁盘存储器按磁头的工作方式分为固定磁头磁盘存储器和移动磁头磁盘存储器；如下图所示：



固定磁头的磁盘存储器，其磁头位置固定不动，磁盘上的每一个磁道都对应一个磁头，盘片也不可更换。其特点是省去了磁头沿盘片径向运动所需寻找磁道的时间，存取速度快，只要磁头进入工作状态即可进行读写操作。

移动磁头的磁盘存储器在存取数据时，磁头在盘面上作径向运动，这类存储器可以由一个盘片组成，也可由多个盘片装在一个同心主轴上，每个记录面各有一个磁头，除上下两外侧为保护面外，其余的盘面可作为记录面，并对应多个磁头。这类结构的硬磁盘存储器，目前应用最广泛。最典型的就是温彻斯特磁盘。

（2）按磁盘是否具有可换性分

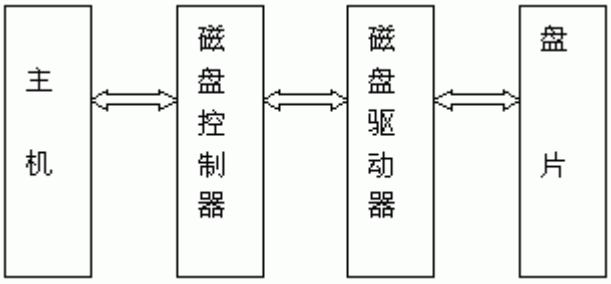
硬磁盘存储器按磁盘是否具有可换性可分为可换盘磁盘存储器和固定盘磁盘存储器。

可换盘磁盘存储器是指盘片可以脱机保存。这种磁盘可以在互为兼容的磁盘存储器之间交换数据，便于扩大存储容量。盘片可以只换单片，如在 4 片盒式磁盘存储器中，3 片磁盘固定，只有 1 片可换。也可以将整个磁盘组(如 6 片、11 片、12 片等)换下。

固定盘磁盘存储器是指磁盘不能从驱动器中取下，更换时要把整个“头盘组合体”一起更换。

2、硬磁盘存储器的结构

硬磁盘存储器是由磁盘驱动器、磁盘控制器和盘片组成，如下图所示。



(1)磁盘驱动器

磁盘驱动器是主机外的一个独立装置，又称磁盘机。大型磁盘驱动器要占用一个或几个机柜，温盘只是一个比砖还小的小匣子。驱动器主要包括主轴、定位驱动及数据控制等

(2) 磁盘控制器

磁盘控制器通常制作成一块电路板，插在主机总线插槽中。其作用是接受由主机发来的命令，将它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和数据传送，并控制驱动器的读写。

(3)盘片

盘片是存储信息的载体，随着计算机系统的不断小型化，硬盘也在朝着小体积和大容量的方向发展。

3、硬磁盘存储器的发展动向

(1) 半导体盘

(2) 提高磁盘记录密度。为提高磁盘记录密度，通常可采用以下技术：

- ✓ 采用高密度记录磁头；
- ✓ 采用先进的信息处理技术，克服由高密度带来的读出信号减弱和信号干扰比下降的缺点；
- ✓ 降低磁头浮动高度和采用高性能磁头浮动块；
- ✓ 改进磁头伺服跟踪技术；
- ✓ 采用高性能介质和基板的磁盘；
- ✓ 改进编码方式

(3) 提高传输率和缩短存取时间

(4) 采用磁盘阵列 RAID

4、硬磁盘的磁道记录格式

盘面的信息串行排列在磁道上，以字节为单位，若干相关的字节组成记录块，一系列的记录块又构成一个“记录”，一批相关的“记录”组成了文件。为了便于寻址，数据块在盘面上的分布遵循一定规律，称为磁道记录格式。**常见的有定长记录格式和不定长记录格式两种。**

(1) 定长记录格式

一个具有 n 个盘片的磁盘组，可将其 n 个面上同一半径的磁道看成一个圆柱面，这些磁道存储的信息叫做柱面信息。在移动磁头组合盘中，磁头定位机构一次定位的磁道集合正好是一个柱面。信息的交换通常在圆柱面上进行，柱面个数正好等于磁道数，故柱面号就是磁道号，而磁头号则是盘面号。

(2) 不定长记录格式

在实际应用中，信息常以文件形式存入磁盘。若文件长度不是定长记录的整数倍时，往往造成记录块的浪费。不定长记录格式可根据需要来决定记录块的长度

（四）软磁盘存储器

1、概述

（1）软磁盘存储器和硬磁盘存储器的比较

- ✓ 存储原理和记录方式相同
- ✓ 结构上有较大差别

（2）软磁盘存储器的种类

- ✓ 按其盘片尺寸不同区分，有 8 英寸、5.25 英寸、3.5 英寸和 2.5 英寸几种。
- ✓ 按使用的磁记录面不同和记录密度不同，又可分为单面单密度、单面双密度、双面双密度等。

2、软磁盘片

软磁盘片的盘基是由厚约为 76μm 的聚脂薄膜制成，其两面涂有厚为 2.3～3μm 的磁层。盘片装在塑料封套内，套内有一层无纺布，用来防尘，保护盘面不受碰撞，还起到消除静电的作用。盘片连封套一起插入软盘机中，盘片在塑料套内旋转，无纺布消除因盘片转动而产生的静电，保证信息可以正常读写。

3、软磁盘的记录格式

软磁盘存储器采用软分段格式，软分段格式有 IBM 格式和非 IBM 格式两种。格式被国际标准化组织(ISO)确定为国际标准。

4、软磁盘驱动器和控制器

软磁盘存储器也由软磁盘驱动器、软磁盘控制器和软磁盘片三部分组成。

软磁盘驱动器是一个相对独立的装置，又称软磁盘机，主要由驱动机构、磁头及定位机构和读写电路组成。

软磁盘控制器的功能是解释来自主机的命令，并向软盘驱动器发出各种控制信号，同时还要检测驱动器的状态，按规定的数据格式向驱动器发出读写数据命令等。

磁盘控制器的具体操作：

（1）寻道操作
(2) 地址检测操作
(3) 读数据操作
(4) 写数据操作
(5) 初始化

软磁盘控制器发给驱动器的信号有：驱动器选择信号(表示某台驱动器与控制器接通)；马达允许信号(表示驱动器的主轴电机旋转或停止)；步进信号（使所选驱动器的磁头按指定方向移动，一次移一道）：步进方向(指磁头移动的方向)；写数据与写允许信号；选头信号(选择“0”面还是“1”面的磁头)。

驱动器提供给控制器的信号有：读出数据信号；写保护信号(表示盘片套上是否贴有写保护标志，如果贴有标记，则发写保护信号)；索引信号(表示盘片旋转到索引孔位置，表明一个磁道的开始)；0 磁道信号(表示磁头正停在 0 号磁道上)。

（五）磁带存储器

1、概述

磁带存储器按顺序进行存取。

磁带存储器是由磁带和磁带机两部分组成。

磁带的分类：

- ✓ 磁带按长度分有 2400 英尺、1200 英尺、600 英尺几种；
- ✓ 按宽度分有 1 件英寸、1 / 2 英寸、1 英寸、3 英寸几种；
- ✓ 按记录密度分有 800bpi、1600bpi、6250bpi 等几种；
- ✓ 按磁带表面并行记录信息的道数分有 7 道、9 道、16 道等；
- ✓ 按磁带外形分有开盘式磁带和盒式磁带两种。

磁带机的分类：

- ✓ 按磁带机规模分布标准半英寸磁带机，海量宽带磁带机(Mass storage)和盒式磁带机三种。
- ✓ 按磁带机走带速度分，有高速磁带机(4～5m/s)、中速磁带机(2～3m/s)和低速磁带机(2m/s 以下)。

磁带机正朝着提高传输率、提高记录密度、改善机械结构、提高可索性等方向发展。

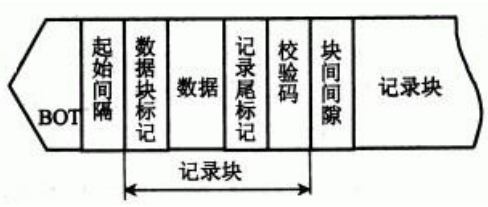
2、数据流磁带机

数据流磁带机是将数据连续地写到磁带上，每个数据块后有一个记录间隙，使磁带机在数据块间不启停，简化了磁带机的结构，用电子控制替代了机械启停式控制，降低了成本，提高了可靠性。

数据流磁带机采用类似磁盘的串行读写方式，它的记录格式与软盘类似。

3、磁带的记录格式

磁带上的信息可以以文件形式存储，也可以按数据块存储。磁带可以在数据块之间启停，进行数据传输。按数据块存储的磁带互换性更好。



（六）光盘存储器

1、概述

光盘(Optical Disk)是利用光学方式进行读写信息的圆盘。根据光存储性能和用途的不同，光盘存储器可

分为三类：

(1)只读型光盘(CD—ROM)。

(2)只写一次型光盘(WORM)

(3)可擦写型光盘

2、光盘的存取原理

光盘存储器利用激光束在记录表面上存储信息，根据激光束和反射光的强弱不同可以实现信息的读写。由于光学读写头和介质保持较大的距离，因此，它是非接触型读写的存储器。

3、光盘存储器的组成

由盘片、驱动器和控制器组成。驱动器同样有读/写头、寻道定位机构、主轴驱动机构等。

4、光盘存储器与其他辅助存储器的比较

光盘、硬磁盘、软磁盘、磁带在记录原理上很相似，都属于表面介质存储器。它们都包括头、精密机械、马达及电子线路等。

光盘是非接触式读/写信息；

光盘可靠性高，对使用环境要求不高，机械振动的问题甚少，不需要采取特殊的防震和防尘措施；

光盘记录头分量重，体积大，使寻道时间长 30～100ms；

光盘的介质互换性好，存储容量大；

硬盘存储器容量大，数据传输率比光盘高(采用磁盘阵列，数据传输率可达 100Mb/s)，等待时间短，它作为主存的后备存储器，用以存放程序的中间和最后结果。

软盘存储器容量小，数据传输率低，平均寻道时间长。

第四章 存储系统 关键词汇

1、交叉方式存储器

交叉方式的存储器可以实现多模块流水式并行存取，大大提高存储器的带宽。由于 CPU 的速度比存储器快，假如我们能同时从存储器取出 M 条指令，这必然会提高机器的运行速度。多体交叉存储器就是基于这种思想提出来的。

存储器地址交叉的方式：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。 在二模块交叉存储器实例中，主存储器按字节编址，而不是按字编址。 注意四个字节允许信号。

2、相联存储器应用

计算机系统中，相联存储器主要用于虚拟存储器中存放分段表、页表和快表；在高速缓冲存储器中，相联存储器作为存放从主存调入快存的页面单元地址之用。这是因为，在这两种应用中，都需要快速查找。

3、替换策略

当使用全相联和组相联方式时，就会使用到替换策略或是替换算法。

较为简单的替换算法有 FIFO，但其效果不是很好，不符合程序的局部性原则，经常出现所谓的“颠簸”现象。常用的替换算法有如下三种：

最不经常使用（LFU）算法：这种算法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。没有考虑对新调入行的处理，因为新调入行的计数值小，容易被替换出去。

近期最少使用（LRU）算法：目前使用较多的一种策略，能够有效的提高命中率。

随机替换：硬件上容易实现并且速度快，虽然表面看起来算法比较随意，但实际模拟显示，其性能还是不错的。

4、段式、页式管理

段是利用程序的模块化性质，按照程序的逻辑结构划分成的多个相对独立部分。把段作为基本信息单位在主存－外存之间传送和定位是比较合理的。把主存按段分配的存储管理方式称为段式管理。

页式管理系统的基本信息传送单位是定长的页。主存的物理空间被划分为等长的固定区域，称为页面。

5、层次页表

当一个页表的大小超过一个页面的大小时，页表就可能分成几个页，分存于几个不连续的主存页面中，然后，将这些页表的起始地址又放入一个新页表中。这样，就形成了二级页表层次。一个大的程序可能需要多级页表层次。

6、数据传输率

数据传输率 Dr 是指单位时间内磁表面存储器向主机传送数据的位数或字节数，它与记录密度 D 和记录介质的运动速度 V 有关：$D_r = D \times V$. 此外，辅存和主机的接口逻辑应有足够快的传送速度，用来完成接收/发送信息，以利主机与辅存之间的传送正确无误。

7、记录密度

记录密度通常是指单位长度内所存储的二进制信息量。磁盘存储器用道密度和位密度表示；磁带存储器则用位密度表示。磁盘沿半径方向单位长度的磁道数为道密度，单位是道/英寸(TPI)或道/毫米(TPM)。为了避免干扰，磁道与磁道之间需保持一定距离，相邻两条磁道中心线之间的距离叫道距，因此道密度 Dt 等于道距 P 的倒数。单位长度磁道能记录二进制信息的位数，称为位密度或线密度，单位是 bpi (bits per inch)或 bpm(位/毫米)。磁带存储器主要用位密度来衡量，常用的磁带有 800bpi，1600bpi，6250bpi 等。

对于磁盘，位密度 Db 可按下式计算：$D_b = \dfrac{f_t}{\pi \cdot d_{\min}}$ f_t 为每道总位数，$d_{\min}$为同心圆中最小直径，各磁道上所记录的信息量是相同的而位密度不同，一般的磁盘位密度是指最内圈磁道的位密度，即最大位密度。

8、存储容量

存储容量是指外存所能存储的二进制信息总数量，一般以位或字节为单位。以磁盘存储器为例，存储容量可按下式计算：$C = n \times k \times s$ 其中 C 为存储总容量，n 为存放信息的盘面数，k 为每个盘面的磁道数，s 为每条磁道上记录的二进制代码数。

磁盘有格式化容量和非格式化容量两个指标，非格式化容量是磁表面可以利用的磁化单元总数。格式化容量是指按某种特定的记录格式所能存储信息的总量，即用户可以使用的容量，它一般为非格式化容量的 60%～70%。

9、平均寻址时间

寻址时间分为两个部分，其一是磁头寻找目标磁道的找道时间 ts，其二是找到磁道后，磁头等待欲读写的磁道区段旋转到磁头下方所需要的等待时间 tw。由于从最外圈磁道找到最里圈磁道和寻找相邻磁道所需时间是不等的，而且磁头等待不同区段所花的时间也不等。因此，取其平均值，称作平均寻址时间 Ta，它是平均找道时间 tsa 和平均等待时间 twa 之和：

$$T_a = t_{sa} + t_{wa} = \frac{t_{s\max} + t_{s\min}}{2} + \frac{t_{w\max} + t_{w\min}}{2}$$

10、误码率

误码率是衡量磁表面存储器出错概率的参数，它等于从辅存读出时，出错信息位数和读出的总信息位

数之比。为了减少出错串，磁表面存储器采用循环冗余码来发现并纠正错误。

第四章 存储系统 FAQ

一、存储保护方式分类如何？

存储区域保护　不是虚拟存储器的主存系统中，可采用界限寄存器方式。

访问方式保护

二、半导体存储器的主要优缺点有哪些？

半导体存储器的主要优点是存取速度快，存储体积小，可靠性高，价格低廉；

主要缺点是断电时读写存储器不能保存信息。

三、ROM 的分类如何？

掩模式只读存储器：优点：可靠性高，集成度高，价格便宜。缺点：不能重写。

次编程只读存储器：分为 PN 结击穿型和熔丝烧断型两种。

第一种写入原理属于结破坏型，即在行列线交点处制作一对彼此反向的二级管，它们由于反向而不能导通，称为 0。若该位需要写入 1，则在相应行列线之间加较高电压，将反偏的一只二极管永久性击穿，留下正向可导通的一只二极管，称为写入 1。显然这是不可逆转的。

更常用的一种写入原理属于熔丝型，制造时在行列交点处连接一段熔丝，即易熔材料称为存入 0。若该位需写入 1，则让它通过较大电流，使熔丝熔断。显然这也是不可逆转的。

多次编程只读存储器：分为 EPROM、EEPROM、FLASH ROM

四、传统存储器的问题有哪些？

速度慢，和 CPU 的速度不匹配，原因如下：

CPU 和主存储器是用不同的材料制成的

一个 CPU 周期可能需要几个存储器字

高速计算的解决方法有：

存储器采用更高速的技术来缩短存储器的读出时间，或加长存储器的字长

采用并行操作的双端口存储器

在 CPU 和主存储器之间插入一个高速缓冲存储器(cache)，以缩短读出时间

在每个存储器周期中存取几个字

五、双端口存储器当两个端口的冲突时如何解决？

当两个端口同时存取存储器同一存储单元时，便发生读写冲突。此时，由判断逻辑来决定哪一个端口进行读写操作。判断的方法有两种：

(1) 如果地址匹配且在 之前有效，片上的控制逻辑在 和 之间进行判断来选择端口（ 判断）。

(2) 如果 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(地址有效判断)。

六、Cache 基本原理是什么？

Cache 是为了解决 CPU 和主存之间速度匹配问题而采用的一项重要技术。它的存取速度要比主存快，由高速的 SRAM 组成，全部功能由硬件实现，保证了其高速度。 Cache 除了有 SRAM 外，还要有控制逻辑。Cache

与主存之间数据交换的单位是“块”。

Cache 的基本操作方式：CPU 首先在 Cache 中进行比较（可使用相联存储器），若 Cache 中有要访问的数据，则无需访存，若没有在进行主存读写，同时，把该数据所在的块复制到 Cache 中。Cache 的这种操作是基于程序执行的局部性原理，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。具体内容可参考操作系统中的存储管理。

七、主存与 Cache 的地址映射方式如何？

由于 Cache 比主存小的多，因此必须使用一种机制将主存地址定位到 Cache 中，即地址映射。这个映射过程全部由硬件实现，对程序员透明。映射的三种方式：

全相联：灵活，不易产生冲突。其缺点是比较电路难于实现，且效率低，速度慢。

直接映射：硬件简单，成本低，但容易产生冲突，不能有效利用 Cache 空间。

组相联：结合上面两种的优点。

八、列举 Cache 的写操作策略？

即 Cache 的数据一致性维护策略。主要有三种方式：

写回法：优点是速度快，缺点是存在数据不一致隐患。

全写法：优点是数据不会出现不一致，缺点是对写操作没有高速缓存的作用。

写一次法：上述两种方法的结合，主要用于多个 Cache 数据不一致的维护，具体策略可参考体系结构的相关内容。

九、虚拟存储器和 Cache 异同？

1）把程序中最近常用的部分驻留在高速度的存储器中。

2）一旦这部分变的不常用了，把它们送回到低速的存储器中。

3）这种换入、换出操作是由硬件或是 OS 完成，对用户透明。

4）力图使存储系统的性能接近高速存储器，价格接近低速存储器。

两者的不同点在于：

1） Cache 用硬件实现，对操作系统透明，而虚拟存储器是用软件、硬件相结合组成。

2） 虚拟存储器对未命中更加明感。

十、段式，页式管理的优缺点？

优点：段的分界与程序的自然分界相对应；段的逻辑独立性使它易于编译、管理、修改和保护。也便于多进程共享；某些类型的段（堆栈、队列）具有动态可变长度，允许自由调度以便有效利用主存空间。

缺点：因为段的长度各不相同，段的起点和终点不定．给主存空间分配带来麻烦。而且容易在段间留下许多空余的零碎存储空间不好利用，造成浪费。

优点：页面的起点相终点地址是固定的，给造页表带来了方便。新页调入主存也很容易掌握，只要有空白页面就可容纳。比段式管理系统的段外空间浪费要小得多。

缺点：由于页不是逻辑上独立的实体，所以处理、保护和共享都不及段式来得方便。

段式存储和页式存储管理各有其优缺点，可以采用分段和分页结合的段页式存储管理系统。程序按模块分段，段内再分页，进入主存仍以页为基本信息传送单位。用段表和页表（每段一个页表）进行两级定位管理。

十一、软磁盘控制器工作步骤：

（1）寻道操作：将磁头定位在目标磁道上；

(2)地址检测操作：主机将目标地址送往磁盘控制器，控制器从驱动器上按记录格式读取地址信息，并与

目标地址进行比较，找到欲读(写)信息的磁盘地址；

(3)读数据操作：首先检测数据标志是否正确，然后将数据字段的内容送入主存、最后用 CRC 校验；

(4)写数据操作：写数据时，不仅要将原始信息经编码后写入磁盘，同时要写上数据区标志和 CRC 校验码以及间隙；

(5)初始化：在盘片上写格式化信息，对每个磁道划分区段。

十二、光盘存储器分类

(1)只读型光盘(CD—ROM)

(2)只写一次型光盘(WORM)

(3)可擦写型光盘。

十三、光盘存储器与其他辅助存储器的比较

光盘、硬磁盘、软磁盘、磁带在记录原理上很相似，都属于表面介质存储器。

光盘是非接触式读/写信息； 光盘可靠性高，对使用环境要求不高，机械振动的问题甚少;光盘记录密度高约为磁盘的 10～100 倍;光盘记录头分量重，体积大，使寻道时间长 30～100ms。写入速度低，约为 0.2 秒，平均存取时间为 100～500ms，与主机交换信息速度不匹配。因此，它不能代替硬盘只能作为硬盘的后备存储器。光盘的介质互换性好，存储容量大;

硬盘存储器容量大，数据传输率比光盘高,等待时间短，它作为主存的后备存储器，用以存放程序的中间和最后结果。

软盘存储器容量小，数据传输率低，平均寻道时间长，而且是接触式存取，易造成盘面磨损或出现误码，不易提高位密度。但软盘盘片灵活装卸，便于携带，互换性好，价格便宜。因此，用它存储操作系统和应用软件极为方便。

磁带存储器数据传输率更低，采用接触式记录，容量也很大，每兆字节价格较低，记录介质也容易装卸、互换和携带，可用作硬盘的后备存储器。

第四章 存储系统 拓展资源

如果没有高速内存的配合，CPU 或 GPU 运算能力再强也无济于事。在计算工业中，芯片性能的提升往往体现为两个方面：其一是芯片自身运算能力的提升，其二就是获取数据能力的增强，也就是拥有更高的内存带宽。那么，接下来就必须得到内存技术的配合，倘若内存工业未理睬 CPU/GPU 的技术跃进，那么 CPU 和图形厂商只能在那里干瞪眼！

基于应用需求的不同，涉及大量图形数据处理的 GPU 需要更快的内存支持，GPU 自身也因此具有惊人的内存位宽——譬如目前 GeForce 8 和 Radeon HD 3000 所流行的 512bit 位宽，这相当于 PC 双通道内存的 4 倍！另一方面，GPU 必须与频率更高的显存芯片配合，由于高速显存总是颇为昂贵，顶级显卡也因此价格不菲。而与 CPU 交互的主内存对容量敏感，在高昂的价格面前，主内存不得不牺牲速度如你所见，PC 主内存的速度总是比显存慢得多，但容量却高出数倍。

在不考虑成本的前提下，无论是显存还是主内存，速度都是越快越好、容量越大越好，内存工业朝着这个共同的方向前进——现在，我们将看到 DDR3 技术进入主流，而 JEDEC（美国的电子工业会 EIA 旗下组织，掌握 DRAM 标准的制定工作）也开始了下一代 DDR4 及其后续技术的制定，而尘封多时的 Rambus XDR 在 PS3 游戏机中重获新生，差分信号技术与串行内存重新出现在日程上——在 CPU 和图形工业开始朝着多核迈进的同时，内存工业也在酝酿同样的技术革命。

摩尔定律最初所指的只是存储器件而非 CPU，直到今天，摩尔定律依然在 DRAM 存储器领域发挥效用。我们可以看过，DRAM 存储器每隔 2-3 年，容量就会增加一倍，DRAM 的带宽大约每隔三年增加一倍，这个步伐也与 CPU 和 GPU 的发展水平相适应。

在主内存领域，虽然 AMD 仍然坚守 DDR2 阵营，AMD 希望延长 DDR2 的生命，为此向 JEDEC 推荐 DDR2-1066 规格，虽然未获得积极的回应，但已经有不少内存厂商推出 DDR2-1066 规格的内存产品。而英特尔则继续扮演技术领导者的角色，义无反顾投入 DDR3 体系的推广，现行的“3”系列芯片组开始提供 DDR3 规格的支持，而新一代“4”系列芯片组则一步到位支持到顶级的 DDR3-1333，这意味着 DDR3 内存将在 2008 年进入主流。与 AMD 类似，英特尔现在计划将 DDR3 规范提高到 2133MHz 的速度，以进一步增强自身平台的性能优势，但现在同样未获得 JEDEC 的认可。JEDEC 现在的重点在于下一代 DDR4 的开发以及未来内存技术的发展方向——按照 JEDEC 的规划，DDR4 将在 2010 年后进入市场，而届时四核心处理器将进入桌面主流，超多核架构的产品也开始入市，更强大的运算力无疑意味着更高的数据渴求，高带宽的 DDR4 可以充分满足这一需求。

第五章 输入输出系统 课堂笔记

◆ **主要知识点掌握程度**

除了 CPU 和存储器两大模块外，计算机硬件系统的第三个关键部分即是输入输出模块，又称输入输出系统。本章重点分析 I/O 与主机交换信息的三种控制方式(程序查询、中断和 DMA)及其相应的接口功能和组成，并对几种常用的 I/O 设备也作简单介绍。

◆ **知识点整理**

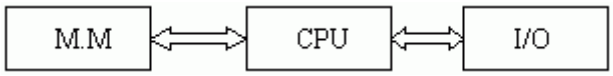
一、概述

（一）输入输出系统的发展概况

输入输出系统的发展大致可分为四个阶段。

1、早期阶段

早期的 I/O 设备种类较少， I/O 设备与主机交换信息都必须通过 CPU，如下图所示。



这种 I/O 设备具有以下几个特点：

（1）每个 I/O 设备都必须配有一套独立的逻辑电路与 CPU 相连，用来实现与主机交换信息，因此线路十分零散、庞杂。

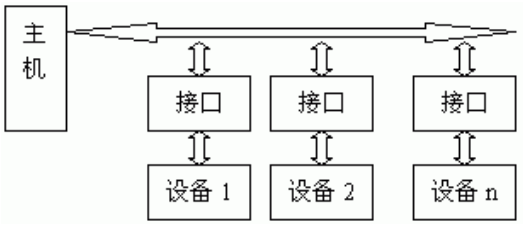
（2）输入输出过程是穿插在 CPU 执行程序之中进行的，当 I/O 与主机交换信息时，CPU 不得不停止其各种运算，因此，I/O 与 CPU 是按串行方式工作的，极浪费时间。

（3）每个 I/O 设备的逻辑控制电路与 CPU 的控制器紧密构成一个不可分割的整体，它们彼此依赖，相互牵连，因此，欲增添或撤减或更换 I/O 设备是非常困难的。

在这个阶段中，计算机系统硬件价格十分昂贵，机器速度不高，配置的 I/O 设备不多，主机与 I/O 交换的信息量也不大，计算机应用极不普及。

2、接口模块和 DMA 阶段

这个阶段 I/O 设备通过接口模块与主机连接，计算机系统采用了总线结构，如下图所示。



通常在接口中都设有数据通路和控制通路。数据经过接口既起到缓冲作用，又可完成串—并变换或并—串变换。控制通路用以传送 CPU 向 I/O 设备发出的各种控制命令，或使 CPU 接受来自 I/O 设备的反馈信号。许多接口还能满足中断请求处理的要求，使 I/O 设备与 CPU 可按并行方式工作，大大地提高了 CPU 的工作效率。采用接口技术还可以使多台 I/O 设备分时占用总线， 使多台 I/O 设备互相之间也可实现并行工作方式，有利于整机工作效率提高。

虽然这个阶段实现了 CPU 和 I/O 并行工作，但是在主机与 I/O 交换信息时，CPU 要中断现行程序，也即 CPU 与 I/O 还不能做到绝对的并行工作。

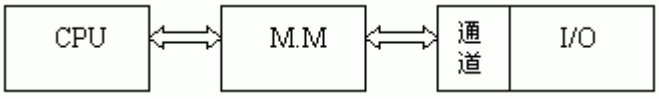
为了进一步提高 CPU 的工作效率，又出现了 DMA(Direct Memory Access)技术，其特点是 I/O 与主存之间有一条直接数据通路，I/O 设备可以与主存直接交换信息，使 CPU 在 I/O 与主存交换信息时，能继续完成自身的工作，故其资源利用率得到了进一步的提高。

3、具有通道结构的阶段

在小型和微型计算机中， 采用 DMA 方式可实现高速外设与主机成组数据的交换，但在大、中型计算机中，外设配置繁多，数据传送频繁，若仍采用 DMA 方式会出现一系列问题：

（1）如果每台外设都配置专用的 DMA 接口，不仅增加了硬件成本，而且为了解决众多 DMA 同时访问主存的冲突问题，会使控制变得十分复杂。

（2）CPU 需要对众多的 DMA 进行管理，同样会占用 CPU 的工作时间，而且因频繁地进入周期挪用阶段，也会直接影响 CPU 的整体工作效率，因此在大、中型计算机系统中，采用了 I/O 通道的方式来进行数据交换。



通道是用来负责管理 I/O 设备以及实现主存与 I/O 设备之间交换信息的部件，它可视为一种具有特殊功能的处理器。通道有专用的通道指令，它能独立地执行用通道指令所编写的输入输出程序，但它不是一个完全独立的处理器，它受 CPU 的 I/O 指令启动、停止或改变其工作状态，是从属于 CPU 的一个专用处理器。依赖通道管理的 I/O 设备在与主机交换信息时，CPU 不直接参与管理，故 CPU 的资源利用率更高。

4、具有 I/O 处理机的阶段

输入输出系统发展到第四阶段是具有 I/O 处理机的阶段。

I/O 处理机又叫做外围处理机(Peripheral Processor Unit 或 PPU)，它基本独立于主机工作，既可完成 I/O 通道要完成的 I/O 控制，还可完成码制变换、格式处理、数据块检错、纠错等操作。 具有 I/O 处理机的输入输出系统与 CPU 工作的并行性更高，这说明 I/O 系统对主机来说，具有更大的独立性。

（二）输入输出系统的组成

输入输出系统应该由 I/O 软件和 I/O 硬件两部分组成。

1、I/O 软件

输入输出系统软件的主要任务是：①如何将用户编制的程序(或数据)输入至主机内；②如何将运算结果输送给用户；③如何实现 I/O 系统与主机工作的协调等。

不同结构的 I/O 系统所采用的软件技术差异很大。一般而言，当采用接口模块方式时，应用机器指令系统中的 I/O 指令及系统软件中的管理程序，便可使 I/O 与主机协调工作。当采用通道管理方式时，除 I/O 指令外，还必须有通道指令及相应的操作系统。即使都采用操作系统，不同的机器其操作系统的复杂程度差异也是很大的。

(1)I/O 指令

I/O 指令是机器指令的一类，它能反映 CPU 与 I/O 设备交换信息的各种特点。

指令的一般格式：

操作码	命令码	设备码

操作码字段可作为 I/O 指令与其他类指令(如访存指令、算逻指令、控制指令等)的判别代码；命令码用来体现 I/O 的具体操作；设备码是作为对多台 I/O 设备的选择码。

I/O 指令的命令码，一般可表述如下几种情况：

• 将数据从 I/O 设备输入至主机。 例如将某台设备接口电路中的数据缓冲寄存器中的数据读至 CPU 的某个寄存器(如累加器 ACC)中；

• 将数据从主机输出至 I/O 设备，例如将 CPU 中的某个寄存器(如 ACC)中的数据写入到某台设备接口电路中的数据缓冲寄存器内；

• 状态测试。利用命令码检测各个 I/O 设备所处伪状态是“忙”(Busy) 还是“准备就绪”(Ready)，以便决定下一步是否可进入主机与 I/O 交换信息的阶段；

• 形成某些操作命令。不同 I/O 设备与主机交换信息时，需完成不同的操作。如磁带机需要正转、反转、读、写、写文件结束等等。又如对于磁盘驱动器，需要读扇区、写扇区、找磁道、扫描记录标识符等。

I/O 指令的设备码相当于设备的地址。只有对繁多的 I/O 设备赋以不同的编号，才能准确选择某台设备与主机交换信息。

(2)通道指令

通道指令是对具有通道的 I/O 系统专门设置的指令， 这类指令一般用以指明参与传送(写入或读出)的数据组在主存中的首地址；指明需要传送的字数或所传送数据组的末地址；指明所选设备的设备码及完成某种操作的命令码。这类指令的位数一般较长， 如 IBM 370 机的通道指令为 64 位。

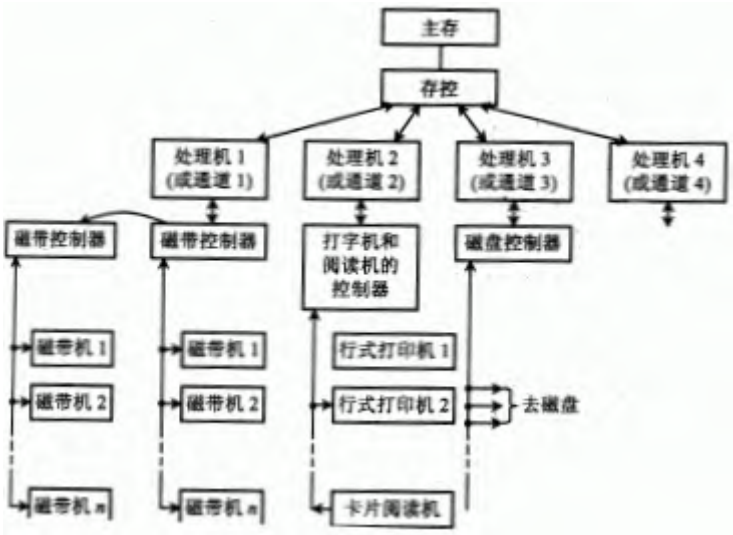
通道指令又叫通道控制字(CCW)。它是通道用于执行 I/O 操作的指令，它可以由管理程序存放在主存的任何地方，由通道从主存中取出并执行。通道程序即由通道指令组成，它完成某种外围设备与主存传送信息的操作。例如，将磁带记录区的部分内容送到指定的主存缓冲区内。

通道指令是通道自身的指令，用来执行 I/O 操作，如读、写、反读、磁带走带及磁盘找道等。而 I/O 指令是 CPU 指令系统的一部分，是 CPU 用来控制输入输出操作的指令，由 CPU 译码后执行。在具有通道结构的机器中，I/O 指令不实现 I/O 数据传送，主要完成启、停 I/O 设备，查询通道和 I/O 设备的状态及控制通道所作的其他一些操作。具有通道指令的计算机，一旦 CPU 执行了启动 I/O 的指令后，就由通道来代替 CPU 对 I/O 的管理。

2、I/O 硬件

输入输出系统的硬件组成是多种多样的，在带有接口的 I/O 系统中，一般包括接口模块及 I/O 设备两大部分。

下图是具有通道的 I/O 系统示意图。



一个通道可以和一个以上的设备控制器相连，一个设备控制器又可以控制若干台同一类型的设备。如 IBM 360 系统的一个通道可以连接 8 个设备控制器，一个设备控制器又与 8 台设备相连，因此，一个通道可以管理 64 台设备。如果一台机器有 6 个通道，便可带动 384 台设备。当然，实际上由于设备利用率和通道的频带影响，主机不可能带动这么多的设备。

（三）I/O 设备与主机的联系方式

I/O 设备与主机交换信息和 CPU 与主存交换信息相比，有许多不同点。

例如，CPU 如何对 I/O 编址；如何寻找 I/O 设备号；信息传送是逐位串行还是多位并行；I/O 与主机以什么方式进行联络，使它们之间彼此都知道双方处于何种状态；以及 I/O 与主机是怎么连接的等等。这一系列问题统称为 I/O 与主机的联系方式。

1、编址方式

通常将 I/O 设备码视为地址码，对 I/O 地址码的编址可采用两种方式：**统一编址或不统一编址**。

统一编址就是将 I/O 地址看作是存储器地址的一部分。如在 64K 地址的存储空间中，划出 8K 地址作为 I/O 的地址，凡是在这 8K 地址范围内的访问，就是对 I/O 的访问， 所用的指令与访存指令相似。

不统一编址就是指 I/O 地址和存储器地址是分开的，所有对 I/O 的访问必须有专用的 I/O 指令。显然统一编址占用了存储空间，减少了主存容量，但无需专用的 I/O 指令。 不统一编址由于不占用主存空间，故不影响主存容量，但需设 I/O 专用指令 。因此，设计机器时，需根据实际情况权衡考虑选取何种编址方式。

当设备通过接口与主机相连时，CPU 可以通过接口地址来访问 I/O 设备。

2、设备寻址

由于每台设备都赋予一个设备号，因此，当要启动某一设备时，可由 I/O 指令的设备码字段直接指出该设备的设备号。通过接口电路中的设备选择电路，便可选中要交换信息的设备。

3、传送方式

在同一瞬间，n 位信息同时从 CPU 输送至 I/O 设备，或由 I/O 设备输入到 CPU， 这种传送方式叫做并行传送。其特点是传送速度较快，但要求数据线多，如 16 位信息并行传送，需 16 根数据线。

若在同一瞬间只传送一位信息，在不同时刻连续逐位传送一串信息，这种传送方式叫做串行传送。其特点是传送速度较慢，但它只需一根数据线和一根地线。当 I/O 设备与主机距离很远时，采用串行传送较为合理，例如远距离数据通讯。

不同的传送方式需配置不同的接口电路，如并行传送接口、串行传送接口或串并联用的传送接口等。用户可按需要选择合适的接口电路。

4、联络方式

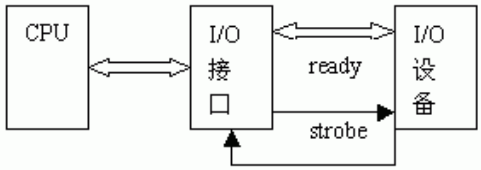
不论是串行传送还是并行传送， I/O 设备与主机之间必须互相了解彼此当时所处的状态，如相互是否可以传送，传送是否已结束等等。这就是 I/O 设备与主机之间的联络问题。按 I/O 设备工作速度的不同，可分为三种联络方式。

(1)立即响应方式

对于一些工作速度十分缓慢的 I/O 设备， 如指示灯的亮与灭；开关的通与断；A/D 转换器缓变信号的输入等等。当它们与 CPU 发生联系时，通常都已使其处于某种等待状态，因此，只要 CPU 的 I/O 指令一到，它们使立即响应，故这种设备无需特殊联络信号，称作立即响应方式。

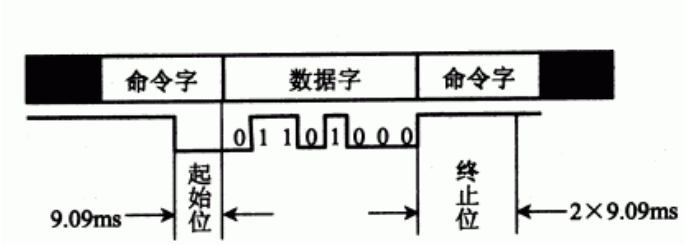
(2)异步工作采用应答信号联络

当 I/O 设备与主机工作速度不匹配时， 通常采用异步工作方式。这种方式在交换信息前，I/O 与 CPU 各自完成自身的任务，一旦出现联络信号时，彼此才准备交换信息。下图示意并行传送的异步联络方式。



当 CPU 将数据输出到 I/O 接口后，接口立即向 I/O 设备发出一个“Ready”(准备就续)信号，告诉 I/O 设备可以从接口内取数据。I/O 设备收到“Ready”后，通常便立即从接口中取出数据，接着便向接口回发一个“Strobe”信号，并让接口转告 CPU，接口中的数据已被取走，CPU 还可继续向此接口送数。同理，倘若 I/O 设备需向 CPU 传送数据，则先由 I/O 向接口送数据，并向接口发“Strobe”信号，表明数据已送出。接口接到联络信号后便通知 CPU 可以来取数，一旦 CPU 取走时，接口便向 I/O 设备发“Ready”信号，告诉 I/O 设备，数据已被取走，尚可继续送数。这种一应一答的联络方式称作异步联络。

下图示意了串行传送的异步联络方式。



I/O 设备与 CPU 双方设定一组特殊标记，用“起始”和“终了”来建立联系。图中 9．09ms 的低电平表示“起始”，又用 2×9.09ms 的高电平表示“终了”。

(3)同步工作采用同步时标联络

同步工作要求 I/O 设备与 CPU 的工作速度完全同步，例如在数据采集过程中，若外部数据以 2400 位/秒速率传送至接口，则 CPU 也必须以 1/2400 秒的速率接收每一位数。这种联络互相之间还得配有专用电路，用以产生同步时标来控制同步工作。

5、I/O 与主机的连接方式

I/O 设备与主机的连接方式通常有两种：辐射式和总线式。

采用辐射式连接方式时， 要求每台 I/O 设备都有一套控制线路和一组信号线，因此所 用的器件和连线较多，对 I/O 设备的增删都比较困难。这种连接方式大多出现在计算机发展的初期阶段。

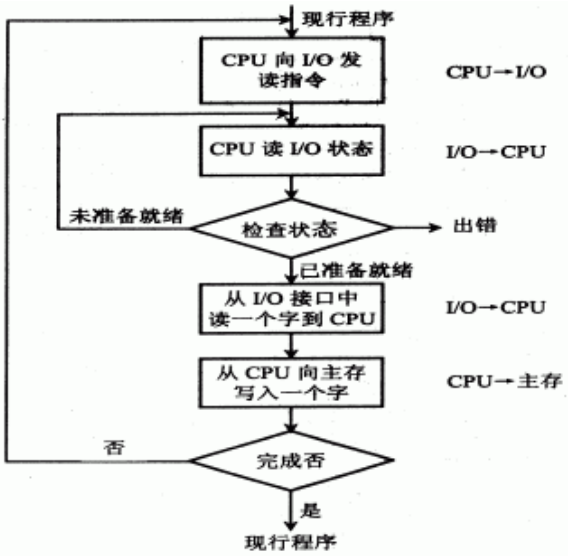
在总线连接方式，通过一组总线(包括地址线、数据线、控制线等)，将所有的 I/O 设备与主机连接。这种连接方式是现代大多数计算机系统所采用的方式。

（四）I/O 与主机信息传送的控制方式

I/O 设备与主机交换信息时，共有五种控制方式：程序查询方式、程序中断方式、直接存储器存取方式(DMA)、I/O 通道方式、I/O 处理机方式。

1、程序查询方式

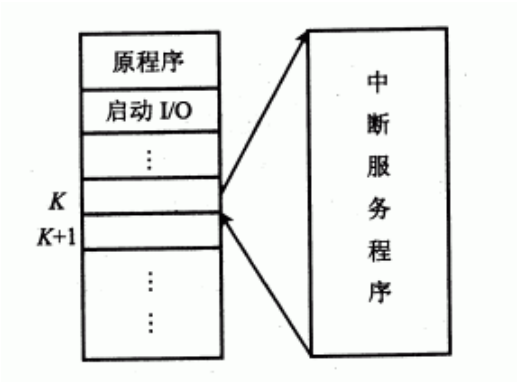
程序查询方式是由 CPU 通过程序不断查询 I/O 设备是否已做好难备，从而控制 I/O 与主机交换信息。采用这种方式实现主机和 I/O 交换信息，要求 I/O 接口内设置一个能反映设备是否准备就绪的状态标记，CPU 通过对此标记的检测，可得知设备的准备情况。下图示意了 CPU 欲从某一外设读数据块(例如从磁带上读一记录块)至主存的查询方式流程。当现行程序需启动某设备工作时，即将此程序流程插入到运行的程序中。由图可知，CPU 启动 I/O 后便开始对 I/O 的状态进行查询。若查得 I/O 未准备就绪，就继续查询；若查得 I/O 准备就绪，就将数据从 I/O 接口送至 CPU，再由 CPU 送至主存。这样一个字一个字地传送，直至这个数据块的数据全部传送结束，CPU 又重新回到原现行程序。



由这个查询过程可见，只要 CPU-启动 I/O 设备，CPU 便不断查询 I/O 的准备情况，从而终止了原程序的执行。CPU 在反复查询过程中，犹如就地“踏步”。另一方面，I/O 准备就绪后，CPU 要一个字一个字地从 I/O 设备取出，经 CPU 送至主存，此刻 CPU 也不能执行原程序，可见这种方式使 CPU 和 I/O 处于串行工作状态，CPU 的工作效率不高。

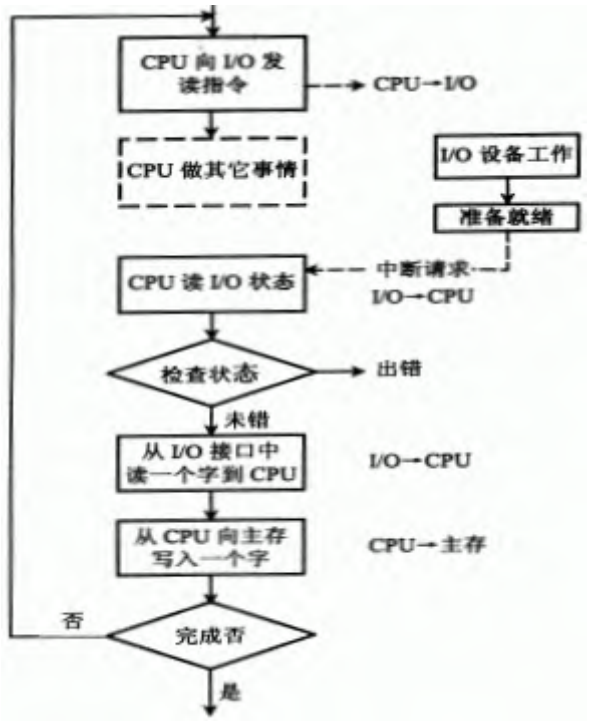
2、程序中断方式

倘若 CPU 在启动 I/O 设备后，对设备是否已准备就绪不加过问，继续执行自身程序，只是当 I/O 设备准备就绪并向 CPU 发出中断请求后才予理睬。这将大大提高 CPU 的工作效率。



由图可见；CPU 启动 I/O 后仍继续执行原程序， 在第 K 条指令执行结束后，CPU 响应了 I/O 的请求，中断了现行程序，转至中断服务程序，等处理完后又返回到原程序断点处， 继续从第 K+1 条指令往下执行。由于这种方式使原程序中断了运行，故叫程序中断方式。

下图是采用程序中断方式从外设读数据块到主存的程序流程示意图。



由图可见，CPU 向 I/O 设备发出读命令后，仍在处理其他一些事情(如继续在算题)。当设备向 CPU 发出请求后，CPU 才从 I/O 接口读一个字经 CPU 送至主存(这是通过执行中断服务程序完成的)。如果 I/O 设备的一批数据(一个数据块的全部数据)尚未传送结束时，CPU 再次启动 I/O 设备，命令 I/O 设备再作准备，一旦又接收到 I/O 设备中断请求时，CPU 又重复上述中断服务过程，这样周而复始，直至一批数据传送完毕。

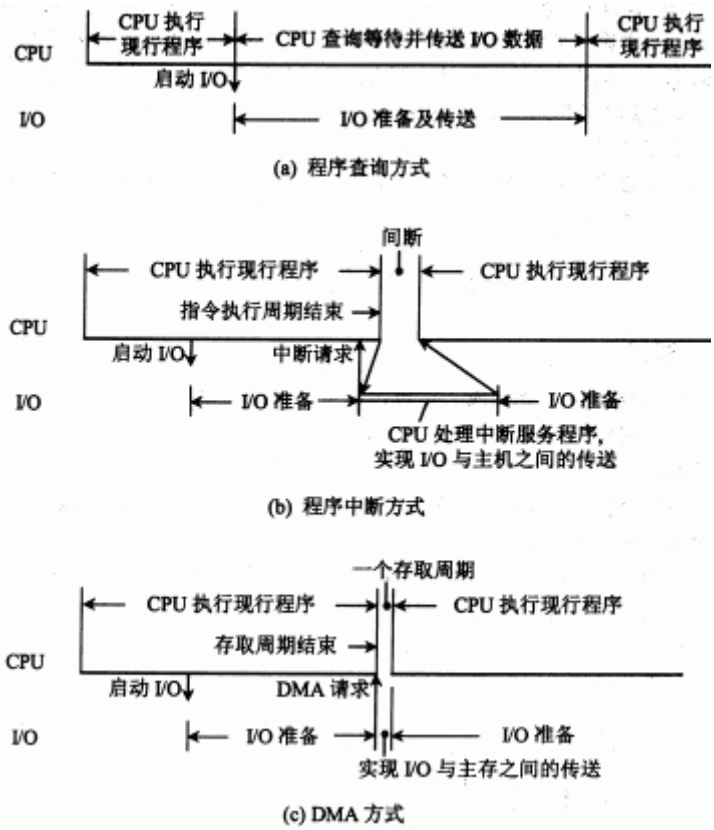
显然，程序中断方式在 I/O 进行准备时， CPU 不必时刻查询 I/O 的准备情况，不出现“踏步”现象，即 CPU 执行程序与 I/O 设备作准备是同时进行的，这种方式和 CPU 与 I/O 是串行工作的程序查询方式相比，其 CPU 的资源得到了充分的利用。

3、DMA 方式

虽然程序中断方式消除了程序查询方式的“踏步”现象，提高了 CPU 资源的利用率，但是 CPU 在响应中断请求后，必须停止现行程序而转入中断服务程序，并且为了完成 I/O 与主存交换信息，还不得不占用 CPU 内部的一些寄存器，这同样是对 CPU 资源的消耗。如果 I/O 设备能直接与主存交换信息而不占用 CPU，那么，CPU 的资源利用率显然又可进一步提高，这就出现了直接存储器存取的 DMA 方式。

在 DMA 方式中，主存与 I/O 设备之间有一条数据通路，主存与 I/O 设备交换信息时，无需处理中断服务程序。若出现 DMA 和 CPU 同时访问主存，CPU 总是将总线占有权让给 DMA，通常把 DMA 的这种占有叫做“窃取”或“挪用”。窃取的时间一般为一个存储周期，故又把 DMA 占用的存取周期叫做“窃取周期”或“挪用周期”。而且，在 DMA 窃取存取周期时，CPU 尚能继续作内部操作(如乘法运算)。可见，DMA 方式与程序查询和程序中断方式相比，又进一步提高了 CPU 的资源利用率。

下图 (c)示意了 DMA 方式的 CPU 效率。当然，采用 DMA 方式时，也需增加必要的 DMA 接口电路。



二、外部设备

（一）概述

中央处理器和主存构成了主机，除主机外的大部分硬件设备都可称作外部设备，或叫外围设备，简称外设。计算机系统没有输入输出设备，就如计算机系统没有软件一样，是毫无意义的。

外部设备大致可分为三类：

1、人机交互设备

它是用来实现操作者与计算机之间互相交流信息的设备。它能将人体五官可识别的信息媒体转换成机器可识别的信息。如键盘、鼠标、手写板、图形扫描仪、摄像机、语言识别器等等。反之，另一类是将计算机的处理结果信息转换为人的可识别的信息媒体，如打印机，显示器、绘图仪、语音合成器等等。

2、计算机信息的驻留设备

它是作为大批信息的驻留设备。例如系统软件和各种计算机的有用信息，其信息量极大，需存储保留起来。这类设备多数可作为计算机系统的辅助存储器，如磁盘、光盘、磁带等等。

3、机-机通信设备

它是用来实现一台计算机与其他计算机或与别的系统之间完成通信任务的设备。例如，两台计算机之间可利用电话线进行通信，它们可以通过调制解调器（MODEM）完成。 用计算机对各种工业控制实行即时操作，可通过 D/A、A/D 转换设备来完成。计算机与计算机及其他系统还可通过各种设备实现远距离的信息交换。

（二）输入设备

输入设备完成输入程序、数据和操作命令等功能。当实现人工输入时，往往与显示器联用，以便检查和修正输入时的错误。也可以利用软盘、磁带等脱机录入的介质进行输入。现在已经可以实现语音直接输入。

1、键盘

键盘是应用最普遍的输入设备，它可通过键盘上的各个键，按某种规范向主机输入各种信息，如汉字、外文、数字等。

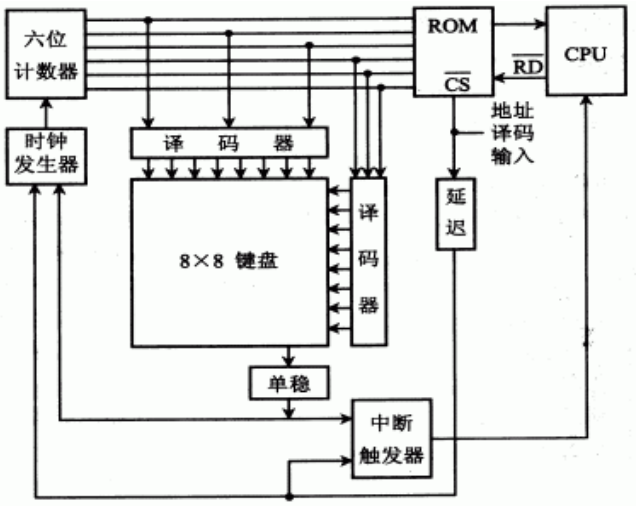
键盘由一组排列成阵列形式的按键开关组成。键盘上的按键分字符键和控制功能键两类。字符键包括字母、数字和一些特殊符号键；控制功能键是产生控制字符的键(由软件系统定义功能)，还有控制光标移动的光标控制键，用于插入或消除字符的编辑键等。

键盘输入信息分三个步骤：

- 按下一个键；
- 查出按下的是哪个键；
- 将此键翻译成 ASCII 码，由计算机接收。

按键是由人工操作的，确认按下的是哪一个键，可由硬件或软件的办法来实现。

采用硬件确认哪个键被按下的方法叫作编码键盘法，它由硬件电路形成对应被按键的惟一的编码信息。带只读存储器的编码键盘原理如下图所示。



8×8 的键盘，由一个六位计数器经两个八选一的译码器对键盘扫描，若键未按下，则扫描将随着计数器的循环计数而反复进行。一旦扫描发现某键被按下，则键盘通过一个单稳电路产生一个脉冲信号。该信号一方面使计数器停止计数，用以终止扫描，此刻计数器的值便与所按键的位置相对应， 该值可作为只读存储器 ROM 的输入地址，而该地址中的内容即为所按键的 ASCII 码。 可见只读存储器存储的信息便是对应各个键的 ASCII 码。 另一方面此脉冲经中断请求触发器向 CPU 发中断请求，CPU 响应请求后便转入中断服务程序，在中断服务程序的执行过程中， CPU 通过执行读入指令，将计数器所对应的 ROM 地址中的内容，即所按键对应的 ASCII 码送入 CPU 中。CPU 的读入指令既可用作读出 ROM 内容的片选信号， 而且经一段延迟后，又可用作清除中断请求触发器，并重新启动六位计数器开始新的扫描。

采用软件判断键是否按下的方法叫作非编码键盘法，它是利用简单的硬件和一套专用键盘编码程序来判断按键的位置， 然后由 CPU 将位置码经查表程序转换成相应的编码信息。这种方法结构简单，但速度比较慢。

在按键时往往会出现键的**机械抖动**，容易造成误动。为了防止形成误判，在键盘控制电路中专门设有硬件消抖电路，或采取软件技术，使可有效地消除因键的抖动而出现的错误。

此外，为了提高传输的可靠性，可采用奇偶校验码。

2、鼠标器

鼠标器(Mouse)是一种手持式的坐标定位部件， 由于它拖着一根长线与接口相连，外形有点像老鼠，故取名为鼠标器。鼠标器有两种：一种是机械式的，它的底座装有一个金属球，球在光滑表面上摩擦使球

转动，球与 4 个方向的电位器接触，可测得上下左右 4 个方向的相对位移量，通过显示器便可确定欲寻求的方位；另一种是光电式鼠标器，它需与一块画满小方格的长方形金属板配合使用。安装在鼠标器底部的光电转换器可以确定坐标点的位宣，同样由显示器显示所寻找的方位。光电式鼠标器比机械式鼠标器可靠性高，但需增加一块金属板。机械式鼠标器可以直接在光滑的桌面上摩擦，但往往因桌面上的灰尘随金属球转动带入鼠标器内，致使金属球转动不灵。

3、触摸屏

触摸屏是一种对物体的接触或靠近能产生反映的定位设备。按触摸原理的不同，大致可分为五类：电阻式、电容式、表面超声波式、扫描红外线式和压感式。

电阻式触摸屏是由显示屏上加二个两层高透明度的、并涂有导电物质的薄膜组成。在两层薄膜之间绝缘隔开，其间隙为 0.0001 英寸。

当用户触摸塑料薄膜片时，涂有金属导电物质的第一层塑料片与挨着玻璃罩上的第二层塑料片(也涂有金属导电物)接触，这样根据其接触电阻的大小求得触摸点所在的 x 和 y 坐标位置。

电容式触摸屏是在显示屏幕上加一个内部涂有金属层的玻璃罩。当用户触摸此罩表面时，与电场建立了电容耦合，在触摸点产生小电流到屏幕四个角，由四个电流大小计算出触摸点的位置。

表面超声波式触摸屏是由一个透明的玻璃罩组成。在罩的 x 和 y 轴方向都有一个发射和接收压电转换器和一组反射器条，触摸屏还有一个控制器发送 5MHz 的触发信号，给发射、接收转换器，让它转换成表面超声波；此超声被在屏幕表面传播；当用手指触摸屏幕时，在确定的位置上超声波被吸收，使接收信号发生变化，经控制分析和数字转换为 x 和 y 的坐标值。

4、其他输入设备

(1)光笔

光笔(Light pen)的外形与钢笔相似， 头部装有一个透镜系统，能把进入的光会聚成一个光点。光笔的后端用导线连到计算机输入电路上，光笔头部附有开关，当按下开关时，进行光检测，光笔便可拾取显示屏上的绝对坐标。光笔与屏幕的光标配合，可使光标跟踪光笔移动，在屏幕上画出图形或修改图形，类似人们用钢笔画图的过程。

(2)画笔与图形板

画笔(Stylus)为笔状，但不是光笔。 它不是用于 CRT 屏幕，而是用于图形板（Tablet)。 当画笔接触到图形板上的某一位置时，画笔在图形板上的位置坐标就会自动传送到计算机中，随着画笔在板上的移动可以画出图形。图形板和画笔构成二维坐标的输入设备，主要用于输入工程图等。将图纸贴在图形板上画笔沿着图纸上的图形移动，即可输入工程图。

图形板是一种二维的 A/D 变换器， 又称作数字化扳。坐标的测量方法有电阻式、电容式、电磁感应式和超声波式几种。

画笔与光笔都是输入绝对坐标，而鼠标器只能输入相对坐标。

(3)图像输入设备

最直接的图像输入设备是摄像机(Camera)，它能摄取任何地点、任何环境下的自然景物和各类物体，经数字量化后变成数字图像存入磁带或磁盘。

如果图像已记录在某种介质上，则可用读出装置来读出图像。例如记录在录像带上的图像可用录放机读出，再将视频信号经图像板量化输入到计算机中。记录在数字磁带上的遥感图像，可直接从磁带输入到计算机中。如果把纸上的图像输入到计算机内，则可用摄像机直接摄入，或用装有 CCD(电荷耦合器件)的图文扫描仪(Scanner)或图文传真机送入计算机，还有一种专用的光机扫描鼓，也可把纸上的图像直接转换成数字图像存入计算机

（三）输出设备

1、显示设备

(1)概述

以可见光的形式传递和处理信息的设备叫显示设备。它是应用最广的人机通信设备。显示设备种类繁多：

按显示器件划分，有阴极射线管(CRT)显示器、液晶显示器(LCD)、等离子显示器(PD)等等；按显示内容分有字符显示器、图形显示器和图像显示器；

按显示器功能分有普通显示器和显示终端（终端是由显示器和键盘组成的一套独立完整的输入/输出设备， 它可以通过标准接口接到远离主机的地方。终端的结构比显示器复杂得多)两类。

在 CRT 显示器中，按扫描方式不同可分为光栅扫描和随机扫描两种；按分辨率不同又可分为高分辨率和低分辨率的显示器。

CRT 是目前应用最广泛的显示器件， 它既可作为字符显示器，又可作为图像、图形显示器。

CRT 的两个重要技术指标：

分辨率：是指显示屏面能表示的像素点数，分辨率越高，图像越清晰。

灰度等级：是指显示像素点相对于亮暗的级差，在彩色显示器中它还表现为色彩的差别。

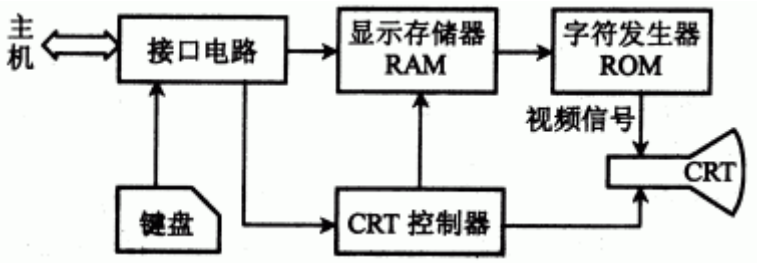
CRT 荧光屏发光是由电子束轰击荧光粉产生的， 其发光亮度一般只能维持几十毫秒。为了使人眼能看到稳定的图像，电子束必须在图像变化前不断地进行整个屏幕的重复扫描，这个过程叫做**刷新**。每秒刷新的次数叫做刷新频率，一般刷新频率大于 30 次 / 秒时，人眼就不会感到闪烁。在显示设备中，通常都采用电视标准，每秒刷新 50 帧(Frame)图像。

为了不断地刷新，必须把瞬时图像保存在存储图像的存储器中，这种存储器叫做**刷新存储器**，又叫做“帧存储器”或“视频存储器”(VRAM)。**刷新存储器的容量由图像分辨率和灰度等级决定**。分辨率越高，灰度等级越多，刷新存储器容量就越大。

计算机的显示器大多采用光栅扫描方式。所谓**扫描**是指电子束在荧光屏上按某种轨迹运动，光栅扫描是从上至下顺序扫描，可分为逐行扫描和隔行扫描两种。

(2)字符显示器

字符显示器是计算机系统中最基本的输出设备，它通常**由显示控制器和显示器(CRT)组成**。



显示存储器(刷新存储器)RAM：

显示存储器存放欲显示字符的 ASCII 码，其容量与显示屏能显示的字符个数有关。每个字符所在存储单元的地址与字符在荧光屏上的位置一一对应，即显示存储器单元的地址顺序与屏面上每行从左到右，按行从上到下的显示器位置对应。

字符发生器：

由于荧光屏上的字符是由光点组成，而显示 RAM 中存放的是 ASCII 码，因此，必须有一个部件能将每个 ASCII 字符码转变为一组 5×7 或 7×9 的光点矩阵信息。具有这种变换功能的部件叫做**字符发生器**，它实质是一个 ROM。

CRT 控制器：

CRT 控制器通常都做成专用芯片，它可接收来自 CPU 的数据和控制信号， 并给出访问显示 RAM 的地址和访问字符发生器的光栅地址，还能给出 CRT 所需的水平同步和垂直同步信号等。

芯片中需配置点计数器、字计数器(水平地址计数器)、行计数器(光栅地址计数器)和排计数器(垂直地址计数器)，这些计数器用来控制显示器的逐点、逐行、逐排、逐幕的刷新显示，还可以控制对显示 RAM 的访问和屏幕间扫描的同步。

计数器的工作情况：

点计数器记录每个字的横向光点，因每个字符占 7 个光点，字符间留一个光点作间隙，共占 8 个光点，故点计数器为模 8 计数器，计满 8 个点向字计数器进位。字计数器用来记录屏幕上每排的字数，字计数器计满 110 就归零，并向行计数器进位；排计数器用来记录每屏字符的排数。

字计数器反映了光栅扫描的水平方向，排计数器反映了光栅扫描的垂直方向，将这两个方向的同步信号输至 CRT 的 x 和 y 偏转线圈，使可达到按指定位置进行显示的要求。

(3)图形显示器

图形显示器是用点、线(直线和曲线)、面(平面和曲面)组合而成的平面或立体图形。并可作平移、比例变化、旋转、坐标变换、投影变换(把三雄图形变为二维图形)、透视变换(由一个三维空间向另一个三维空间变换)、透视投影(把透视变换和投影变换结合在一起)、轴侧投影(三面图)、单点透视、两点或三点远视以及隐线处理(观察物体时把看不见的部分去掉)等操作。

利用 CRT 显示器产生图形有两种方法：一种是随机扫描法，另一种是光栅扫描法。

光栅扫描显示器的通用性强，灰度层次多，色调丰富，显示复杂图形时无闪烁，所形成的图形可以有消除隐藏面、阴影效应和涂色等功能。

(4)图像显示器

图形显示器所显示的图形是由计算机用一定的算法形成的点、线、面、阴影路，它来自主观世界，故又称主观图像或叫做计算机图像。

图像显示器是把由计算机处理后的图像(称为数字图像)，以点阵列的形式显示出来。

图像显示器除了能存储从计算机输入的图像并在显示屏幕上显示外，还具有灰度变换、窗口技术、真彩色和伪彩色显示等图像增强技术功能。

(5)IBM PC 系列微型机的显示标准

这里仅介绍几种显示标准。

①MDA(Monochrome Display Adapter)标准

MDA 是单色字符显示标准

②CGA(Color Graphics Adapter)标准

CGA 是彩色图形/字符显示标准，它可兼容字符和图形两种显示方式。

③EGA(Enhanced Graphics Adapter)标准

EGA 标准集中了 MDA 和 CGA 两个显示标准的优点，并有所增强。

④VGA(Video Graphics Adapter)标准

在图形方式下分辨率为 640×480、16 种颜色，或 320×200、256 种颜色。改进型 VGA 显示控制板(如 VGA+、Super VGA 或 TVGA)的图形分辨率可达 800×600、960×720 和 1024×768、256 种颜色。

习惯上将 MDA，CGA 称作 IBM PC 机的第一代显示标准，EGA 为第二代标准，VGA 为第三代标准。

2、打印设备

(1)打印设备的分类

按印字原理分，有击打式和非击打式两大类。**击打式打印机**是利用机械动作使印字机构与色带和纸相接击而打印字符，其特点是设备成本低，印字质量较好，但噪音大，速度慢。它又分为活字打印机和点阵针式打印机两种。活字打印机是将字符刻在印字机构的表面上，印字机构的形状有圆柱形、球形、菊花瓣形、鼓轮形、链形等，现在用得越来越少。点阵打印机的字符是点阵结构，它利用钢针撞击的原理印字，目前仍用得较普遍。**非击打式打印机**是采用电、磁、光、喷墨等物理、化学方法来印刷字符。如激光打印

机、静电打印机、喷墨打印机等，它们速度快，噪音低，印字质量比击打式好，但价格比较高，有的设备需用专用纸张印刷。

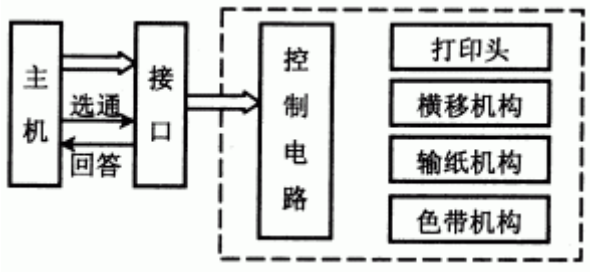
按工作方式分，有串行打印机和行式打印机两种。前者是逐字打印，后者是逐行打印，故行式打印机比串行打印机速度快。

(2)点阵针式打印机

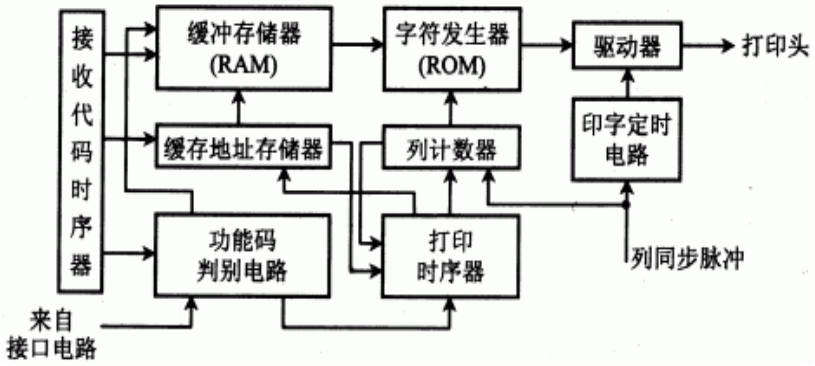
点阵针式打印机结构简单、体积小、重量轻、价格低、字符种类不受限制、较易实现汉字打印，还可打印图形和图像，是目前应用最广泛的一种打印设备。

点阵针式打印机的印字原理是由打印针(钢针)印出 n×m 个点阵来组成字符或图形。点越多越密，其字形质量越高。西文字符点阵通常采用 5×7、7×7、7×9、9×9 几种，汉字的点阵采用 16×16、24×24、32×32 和 48×48 多种。

针式打印机由打印头、横移机构、输纸机构、色带机构和相应的控制电路组成，如下图所示。



控制电路的细化图如下图所示。



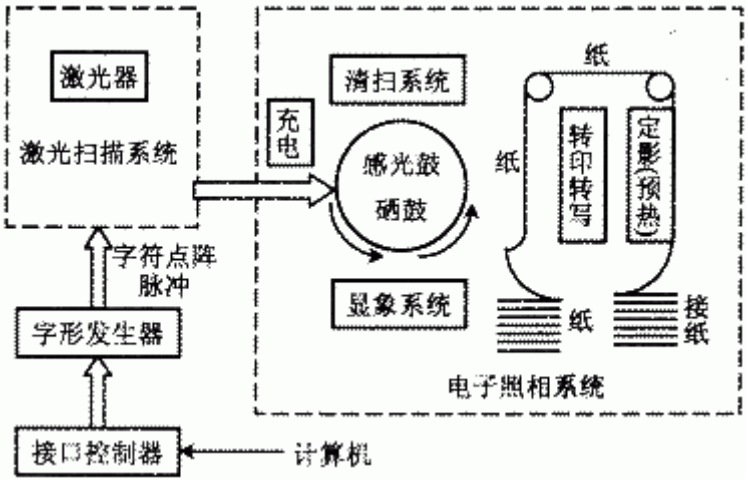
打印机被 CPU 启动后， 在接收代码的时序控制下，功能码判别电路开始接收从主机送来的欲打印字符的字符代码(ASCII 码)。 首先判断该字符是打印字符码还是控制功能码(如回车、换行、换页等)，若是打印字符码，则送至缓冲存储器， 直到把缓存 RAM 装满为止；若是控制功能码，则打印控制器停止接收代码并转入打印状态。打印时首先启动打印时序器，并在它控制下，从缓存中逐个读出打印字符码，再以该字符码作为字符发生器 ROM 的地址码，从中选出对应的字符点阵信息(字符发生器可将 ASCII 码转换成打印字符的点阵信息)。然后在列同步脉冲计数器控制下，将一列列读出的字符点阵信息送至打印驱动电路，驱动电磁铁带动相应的钢针进行打印。每打印一列，固定钢针的托架就要横移一列距离，直到打印完最后一列，形成 n×m 点阵字符。当一行字符打印结束或换行打印、或缓存内容已全部打印完毕时，托架就返回到起始位置，并向主机报告，请求打印新的数据。

输纸机构受步进电机驱动，每打印完一行字符，按给定要求走纸。色带的作用是供给色源，如同复写纸的作用一样。钢针撞击在色带上，就可将色印在纸上，色带机构可使色带不断移动，以改变受击打的位置，避免色带的破损。

有的点阵针式打印机内部配有一个独立的微处理器，用来产生各种控制信号，完成复杂的打印任务。

(3)激光打印机

激光打印机采用了激光技术和照相技术，由于它的印字质量好，因此在各种计算机系统中广泛被采用。下图示意了激光打印机的工作原理。

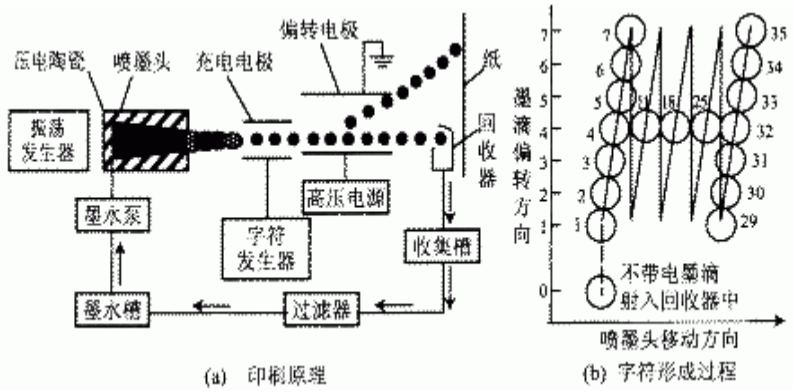


激光打印机由激光扫描系统、电子照相系统、字形发生器和接口控制器几部分组成。接口控制器接收由计算机输出的二进制字符编码及其他控制信号；字形发生器可将二进制字符编码转换成字符点阵脉冲信号；激光扫描系统的光源是激光器，该系统受字符点阵脉冲信号的控制，能输出很细的激光束，该激光束对作圆周运动的感光鼓进行轴向(垂直于纸面)扫描。感光鼓是电子照相系统的核心部件，鼓面上涂有一层具有光敏特性的感光材料，通常用硒，故又有硒鼓之称。感光鼓在未被激光扫描之前，先在黑暗中充电，使鼓表面均匀地沉积一层电荷，扫描时激光束对鼓表面有选择地曝光，被曝光的部分产生放电现象，未被曝光的部分仍侠留充电时的电荷，这就形成了“潜像”。随着鼓的圆周运动，“潜像”部分通过装有碳粉盒的显像系统，使“潜像”部分(实际上是具有字符信息的区域)吸附上碳粉，达到“显影”的目的。当鼓上的字符信息区和打印纸接触时，由于纸的背面施以反向的静电电荷，则鼓面上的碳粉就会被吸附到纸面上，这就是“转印”或“转写”过程。最后经过定影系统就将碳粉永久性地粘在纸上。转印后的鼓面还留有残余的碳粉，故先要除去鼓面上的电荷，经清扫系统将残余碳粉全部清除，然后再重复上述充电、曝光、显形、转印、定影等一系列过程。

(4)喷墨打印机

喷墨打印机是串行非击打式打印机，印字原理是将墨水喷射到普通打印纸上。若采用红、绿、蓝三色喷墨头，便可实现彩色打印。随着喷墨打印技术的不断提高，使其输出效果接近于激光打印机，而价格又与点阵针式打印机相当，因此，在计算机系统中被广泛应用。

下图是一种电荷控制式喷墨打印机的原理框图。主要由喷头、充电电极、墨水供应、过滤回收系统及相应的控制电路组成。



喷墨头后部的压电陶瓷受振荡脉冲激励，使喷墨头喷出具有一定速度的一串不连续、不带电的墨水滴。墨水滴通过充电电极时被充上电荷，其电荷量的大小由字符发生器的输出控制。字符发生器可将字符编码转换成字符点阵信息。由于各点的位置不同，因此充电电极所加的电压也不同。电压越高，充电电荷越多，墨滴经偏转电极后偏移的距离也越大，最后墨滴落在印字纸上。图中只有一对垂直方向的偏转电极，因此

墨滴只能在垂直方向偏移。若垂直线段上某处不需喷点(对应字符在此处无点阵信息)，则相应墨滴不充电，在偏转电场中不发生偏转，而射入回收器中。横向没有偏转电极，靠喷头相对于记录纸作横向移动来完成横向偏转。上图(b)示意了 H 字符由 5×7 点阵组成。墨滴的运动轨迹如图中所示的数字顺序移动，可见字符中的每个点都要一个个地进行控制，故字符发生器的输出必须是一个点一个点的信息。这与点阵针式打印机的字符发生器一次输出一列的上七个点信息，分 5 次打印一个字符是完全不同的。

(5)几种打印机的比较。以上介绍的三种打印机都配有一个字符发生器，它们的共同点是都能将字符编码信息变为点阵信息，不同的是这些点阵信息的控制对象不同。点阵针式打印机的字符点阵用于控制打印针的驱动电路：激光打印机的字符点阵脉冲信号用于控制激光束；喷墨打印机的字符点阵信息控制墨滴的运动轨迹。

此外，点阵针式打印机是属击打式的打印机，可以逐字打印也可以逐行打印，喷墨打印机只能逐字打印，激光打印机属页式输出设备。后两种都属非击打式打印机。

不同种类的打印机其性能和价格差别很大，用户可根据不同需要合理选用。要求印字质量高的场合可选用激光打印机；要求价格便宜的或只需具有文字处理功能的个人用计算机，可配置串行点阵针式打印机：要求处理的信息量很大，速度又要快，应该配行式打印机或高速激光打印机。

（四）其他外部设备

计算机的外部设备中有一类属于既是输入设备，又是输出设备。如磁盘、终端、A/D、D/A 转换器，以及汉字处理设备等等。

1、终端设备

终端是由显示器和键盘组成的一套独立完整的输入/输出设备,它可以通过标准接口接到远离主机的地方使用。终端与显示器是两个不同的概念，终端的结构比显示器复杂，它能完成显示控制与存储、键盘管理及通信控制等，还可完成简单的编辑操作。

2、A/D、D/A 转换器

当计算机用于过程控制时，其控制信号是模拟量，而计算机仅能处理数字量，这就要用 A/D、D/A 转换器来完成模拟量与数字量之间的相互转换任务。

A/D 转换器是模拟/数字转换器，它能将模拟量转换成数字量，是计算机的输入设备。A/D 转换器均已制成各种规格的芯片。

D/A 转换器是数字/模拟转换器，它能将计算机输出的数字量转换成控制所需的模拟量，以便控制被控对象或直接输出模拟信号，它是计算机的输出设备。D/A 转换器现在也均已制成规格化的各类芯片。

A/D、D/A 转换器均属于过程控制设备，往往还需要配置其他设备，如传感器、放大电路、执行机构以及开关量输入/输出设备等， 与计算机共同完成对象的过程控制。

3、汉字处理设备

汉字编码可分为输入码、内码和字形码三大类。输入码是解决汉字的输入和识别问题的；内码是由输入码转换而成的，只有内码才能在计算机内进行加工处理；字形码能显示或打印输出。

汉字处理设备包括汉字输入、汉字存储和汉字输出三个部分。

(1)汉字的输入

采用西文标准键盘输入汉字时，必须对汉字进行编码，以使用字母、数字串替代汉字输入。

汉字编码方法主要有三类：数字编码、拼音编码和字形编码。

(2)汉字的存储

汉字的存储包括汉字内码存储和字形码的存储。

汉字内码是汉字信息在机内存储、交换、检索等过程中所使用的机内代码，通常用两个字节表示。使用汉字内码字符时，应注意和英文字符区别开。 英文字符的机内代码是七位 ASCII 码，字节的最高位为“0”，而汉字内码的两个字节最高位均为“1”。以汉字操作系统 CCDOS 中的汉字内码为例，汉字国标码

“兵”用十六进制表示为“3224H”，每个字节最高位加“1”后，使得汉字内码为“B2A4H”。当使用编辑程序输入汉字时，存储到磁盘上的文件就是用机内码表示汉字的。有些机器把字节的最高位用作奇偶校验这位，这时汉字内码需用三个字节表示。

(3)汉字的输出

汉字输出有打印输出和显示输出两种形式。针式汉字打印机有 24 针和 16 针两种，前者印字质量高。也可采用 9 针的西文打印机，当用 9 针打印机打印汉字时，需用软件控制把一行汉字分成两次打印，即每次打印 8 个点，第一次打印一行汉字的上半部，第二次打印一行汉字的下半部，拼在一起构成 16×16 的点阵汉字。

（五）多媒体技术

1、什么是多媒体

研究多媒体计算机技术，就是要强调计算机与声音、活动图像和文字相结合。例如将录像内容输到计算机盘上(如果需要可进行处理)，在播放时，可与多种其他媒体信息(如文字、声音)混合在一起，形成一个多媒体的演示系统。又如，将计算机产生的图形或动画与摄像机摄得的图像叠加在一起等等。

此外，采用人机对话方式，对计算机存储的各种信息进行查找、编辑以及实现同时播放，使多媒体系统成为一个交互式的系统。可见，多媒体计算机可作为研制高度智能计算机系统的一个平台。

2、多媒体计算机的关键技术

(1)视频和音频数据的压缩和解压缩技术

(2)多媒体专用芯片

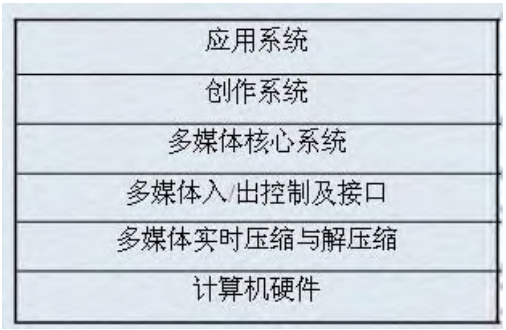
由于多媒体计算机承担大量与数据信号处理、图像处理、压缩与解压缩以及解决多媒体之间关系等有关的问题，而且要求处理速度快，因此需研制专用芯片。一般多媒体专用芯片有两种类型：固定功能的和可编程的。

(3)大容量存储器

多媒体计算机需要存储的信息量极大，因此研制大容量的存储器仍是多媒体计算机系统的关键技术。

(4)适用于多媒体技术的软件。

下图示意了多媒体系统的层次结构：



三、I/O 接口

（一）概述

I/O 接口通常是指主机与外部设备之间设置的一个硬件电路及其相应的软件控制。不同的设备都有其相应的设备控制器，而它们往往都是通过 I/O 接口与主机取得联系的。

主机与外设之间设置接口的理由是：

(1)一台机器通常配有多台外设， 他们各自有其设备号(地址)，通过接口可实现设备的选择。

(2)外部设备种类繁多，速度不一，与 CPU 速度相差可能很大，通过接口可实现数据缓冲达到速度匹配。

(3)有些外部设备可能串行传送数据，而 CPU 一般为并行传送，通过接口可实现数据串一并格式的转换。

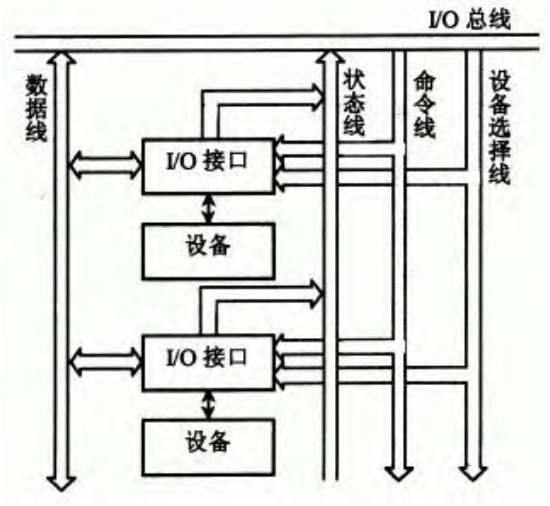
(4) 外部设备的入/出电平可能与 CPU 的入/出电平不同，通过接口可实现电平转换。

(5)CPU 启动外部设备工作，要向外设发各种控制信号，通过接口可传送控制命令。

(6)外部设备需将其工作状态(如“忙”、“就绪”、“错误”、“中断请求”等)及时向 CPU 报告，通过接口可监视设备的工作状态，并可保存状态信息，供 CPU 查询。

（二）接口的功能和组成

1、总线连接方式的 I/O 接口电路



上图所示为总线结构的计算机，每一台设备都是通过 I/O 接口挂到系统总线上的。图中的 I/O 总线包括数据线、设备选择线、命令线和状态线。

(1)数据线

数据线是 I/O 与主机之间数据代码的传送线， 其根数一般等于存储字长的位数或字符的位数，它通常是双向的，也可以是单向的。若采用单向数据总线，则必须用两组才能实现数据的输入和输出两种功能，而双向数据总线只需一组即可。

(2)设备选择线

设备选择线是用来传送设备码的， 它的根数取决于 I/O 指令中设备码的位数。如果把设备码看作是地址号，那么设备选择线又可称为地址线。设备选择线可以有一组也可以有两组，其中一组用于主机向 I/O 发送设备码，另一组用于 I/O 向主机回送设备码。当然设备选择线也可采用一组双向总线代替两组单向总线。

(3)命令线

命令线主要用以传输 CPU 向设备发出的各种命令信号， 如启动、清除、屏蔽、读、写等等。它是一组单向总线，其根数与命令信号多少有关。

(4)状态线

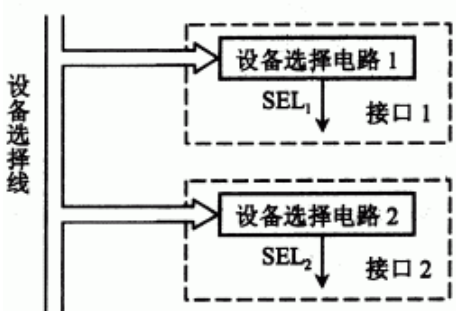
状态线是将 I/O 设备的状态向主机报告的信号线， 如设备是否准备就绪，是否向 CPU 发出中断请求等等。它也是一组单向总线。

现代计算机中大多采用三态逻辑电路来构成总线。

2、接口的功能和组成

(1) 选址功能

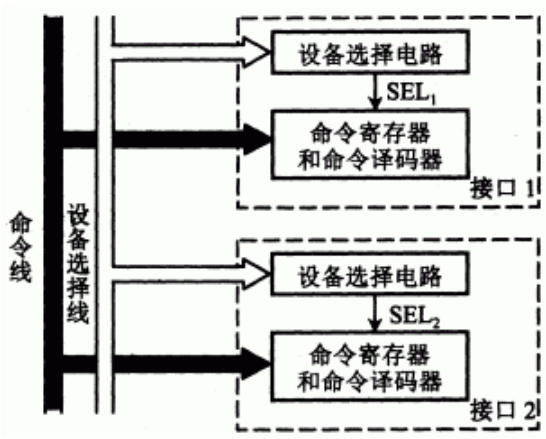
由于 I/O 总线与所有设备的接口电路相连，但 CPU 究竟选择哪台 I/O，还得通过设备选择线上的设备码来确定。该设备码将送至所有设备的接口，因此，要求每个接口都必须具有选址功能，即当设备选择线上的设备码与本设备码相符时，应发出设备选中信号 SEL，这种功能可通过接口内的设备选择电路来实现。



上图示意了接口 1 和接口 2 的设备选择电路。这两个电路具体线路可以不同，它们分别能识别出自身的设备码，一旦某接口设备选择电路有输出时，它便可控制这个设备通过命令线、状态线和数据线与主机交换信息。

(2)传送命令的功能

当 CPU 向 I/O 发出命令时，要求 I/O 设备能作出响应，如果 I/O 接口不具备传送命令信息的功能，那么设备将无法响应， 故通常在 I/O 接口中设有存放命令的命令寄存器以及命令译码器，如下图所示。



(3)传送数据的功能

既然接口处于主机与 I/O 设备之间， 因此数据必须通过接口才能实现主机与 I/O 设备之间的传送， 这就要求接口中具有数据通路，完成数据传送。这种数据通路还应具有缓冲能力，即将数据能暂存在接口内。接口中通常设有数据缓冲寄存器 DBR(Data Buffer Register)，它用来暂存 I/O 设备与主机准备交换的信息，它与 I/O 总线中的数据线是相连的。

(4)反映 I/O 设备工作状态的功能

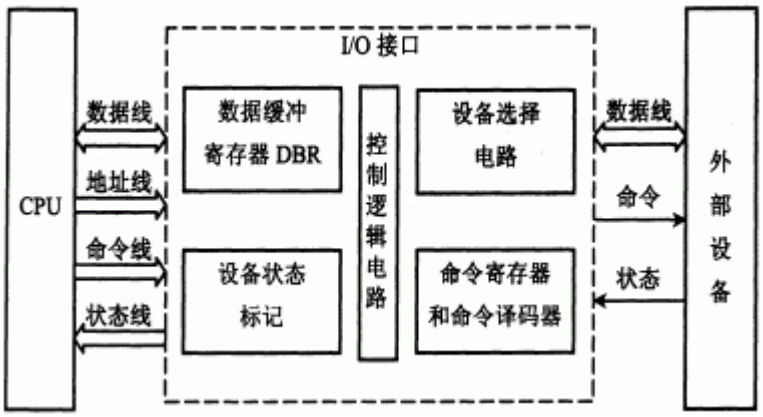
为了使 CPU 能及时了解各 I/O 设备的工作状态，接口内必须设置一些反映设备工作状态的触发器。例如用完成触发器 D 和工作触发器 B 来标志设备所处的状态。

当 D=0，B=0 时，表示 I/O 设备处于暂停状态；

当 D=1，B=0 时，表示 I/O 设备已经准备就绪；

当 D=0，B=1 时，表示 I/O 设备正处于准备状态。

下图示意了 I/O 接口的基本组成：



（三）接口类型

I/O 接口按不同方式分类有以下几种。

1、按数据传送方式分类：

有并行接口和串行接口两类。

并行接口是将一个字节(或一个字)的所有位同时传送(如 Intel 8255)；

串行接口是在设备与接口间一位一位传送(如 Intel8251)。 由于接口与主机之间是按字节或字并行传送，因此对串行接口而言，其内部还必须没有串－并转换装置。

2、按功能选择的灵活性分类

有可编程接口和不可编程接口两种。可编程接口的功能及操作方式，可用程序来改变或选择(如 Intel 8255、8251)；不可编程接口不能由程序来改变其功能，但可通过硬连线逻辑来实现不同的功能(如 Intel 8212)。

3、按通用性分类

有通用接口和专用接口。 通用接口可供多种外设使用，如 Intel8255、8212；专用接口是为某类外设或某种用途专门设计的，如 Intel 8279 可编程键盘/显示器接口；Intel 8275 可编程 CRT 控制器接口等。

4、按数据传送的控制方式分类

有程序型式接口和 DMA 式接口。

程序型式接口用于连接速度较馒的 I/O 设备，如显示终端、键盘、打印机等。现代计算机一般都可采用程序中断方式实现主机与 I/O 设备交换信息，故都配有这类接口，如 Intel 8259。

DMA 型接口是用于连接高速 I/O 设备，如磁盘、磁带等，常用 Inte1 8257。

四、程序查询方式

（一）程序查询流程

程序查询方式的核心问题在于每时每刻需不断查询 I/O 设备是否准备就绪。当 I/O 设备较多时，CPU 需按各个 I/O 设备在系统中的优先级别进行逐级查询，

1、为了正确完成这种查询，通常要执行的指令

(1)测试指令，用来查询设备是否准备就绪；

(2)传送指令，当设备已准各就绪时，执行传送指令；

(3)转移指令，若设备未准备就绪，执行转移指令，转至测试指令，继续测试设备的状态。

2、当 CPU 需启动外设前，必须准备工作

(1)由于这种方式传送数据时要占用 CPU 中的寄存器，故首先需将寄存器原内容保护起来(若该寄存器中存有有用信息)。

(2)由于传送往往是一批数据，因此需先设置设备与主机交换数据的计数值。

(3)设置欲传送数据在内存缓冲区的首地址。

然后 CPU 启动设备，接着以下步骤操作。

(4)启动外部设备。

(5)将 I/O 接口中的设备状态标记取至 CPU 并测试 I/O 是否准备就绪。如果未准备就绪，则踏步等待，直到准备就绪为止。当准备就绪时，接着可实现传送。对输入而言，准备就绪意味着接口电路中的数据缓冲寄存器已装满欲传送的数据，叫做“输入缓冲满”，CPU 即可取走数据；对输出而言准备就结绪意味着接口电路中的数据已被设备取走，故叫“输出缓冲空”，这样 CPU 可再次将数据送到接口，设备可再次从接口接收数据。

(6)CPU 执行 I/O 指令，或从 I/O 接口的数据缓冲寄存器中读出一个数据，或把一个数据写入到 I/O 接口中的数据缓冲寄存器内，同时把接口中的状态标记复位。

(7)修改内存地址。

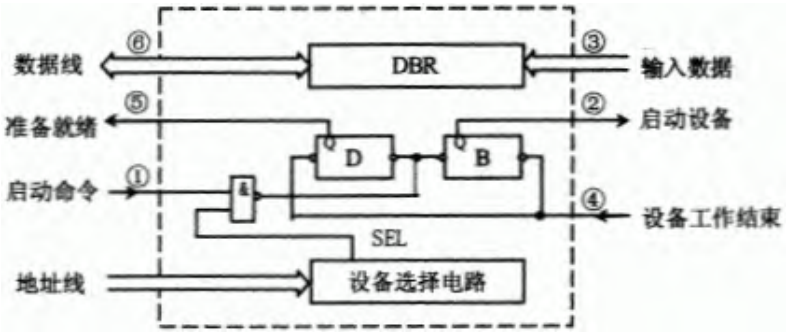
(8)修改计数值，若原设置计数值为原码，则依次减 1；若原设置计数值为负数补码，则依次加 1。

(9)判断计数值。若计数值不为 0，表示一批数据尚未传送完，重新启动外设继续传送；若计数值为 0，则表示一批数据已传送完毕。

(10)结束 I/O 传送，继续执行其他程序。

（二）程序查询方式的接口电路

程序查询方式接口电路的基本组成如下图所示。



图中设备选择电路用以识别本设备地址，当地址线上的设备号与本设备号相符时，SEL 有效，可以接收命令。

DBR 是数据缓冲寄存器，用以存放欲传送的数据；

D、B 是反映设备工作状态的标记触发器。

以输入设备为例，本接口的工作过程如下：

当设备选中后，（1）由 CPU 发出启动外设命令，将工作触发器 B 置“1”，完成触发器 D 置“0”；

（2）启动外设开始工作；

（3）输入设备将数据送入 DBR；

（4）外设工作完成，向接口发“设备工作结束”信号，将 D 置“1”，B 置“0”；

（5）D 触发器以“准备就绪”状态通知 CPU，表示“数据缓冲满”；

（6）CPU 执行输入指令，将输入数据送至 CPU 的通用寄存器，再存入主存相关单元。

五、程序中断方式

（一）中断的概念

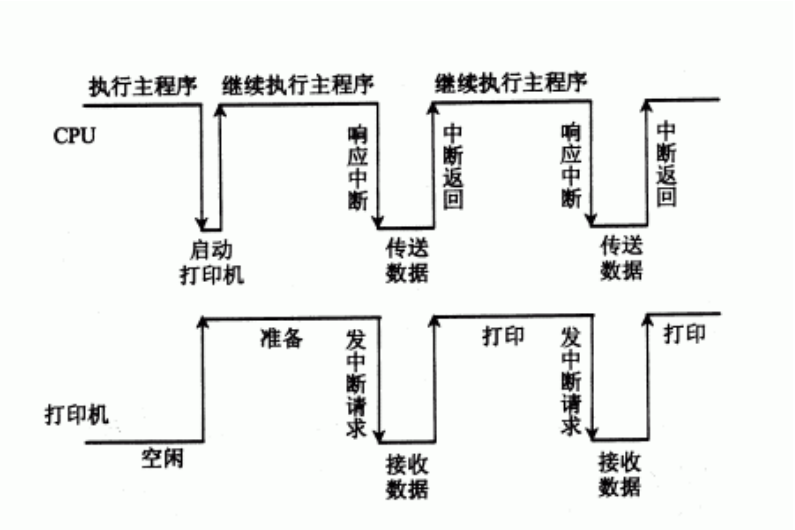
计算机在执行程序的过程中，当出现异常情况或特殊请求时，计算机停止现行程序的运行，转向对这些异常情况或特殊请求的处理，处理结束后再返回到现行程序的间断处，这就是“中断”。

通常又把实现这种功能所需的软硬件技术，统称为中断技术。

（二）I/O 中断的产生

在 I/O 与主机交换信息时，由于设备本身机电特性的影响，其工作速度较低，与 CPU 无法匹配，因此，CPU 启动设备后，往往需要等待一段时间才能实现主机与设备交换信息。如果在设备准备的同时，CPU 不作无谓的等待，而继续执行现行程序，只有当 I/O 准备就绪向 CPU 提出请求后，再暂时中断 CPU 现行程序转入 I/O 服务程序，这便产生了 I/O 中断。

下图示意了由打印机引起的 I/O 中断时，CPU 与打印机的并行工作时间示意。



（三）程序中断方式的接口电路

为处理 I/O 中断，在 I/O 接口电路中必须配置相关的硬件线路。

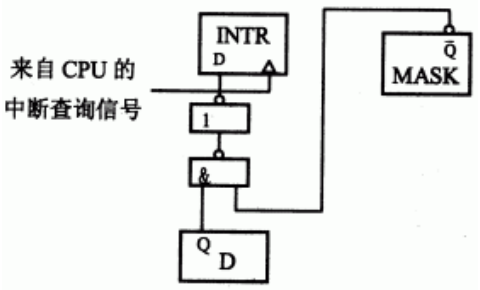
1、中断请求触发器和中断屏蔽触发器

每台外部设备都必须配置一个中断请求触发器 INTR，当其为“1”时，表示该设备向 CPU 提出中断请求。但是设备欲提出中断请求时，其设备本身必须准备就绪，也即接口内的完成触发器 D 必为 “1”状态。

把凡能向 CPU 提出中断请求的各种因素统称为**中断源**。当多个中断源向 CPU 提出中断请求时，CPU 必须坚持一个原则，即在任何瞬间只能接受一个中断源的请求。

CPU 总是在统一的时间，即执行每条指令的最后时刻，查询所有的设备是否有中断请求。

综合上述各因素，可得出接口电路中的完成触发器 D、中断请求触发器 INTR、中断屏蔽触发器 MASK 和中断查询信号的关系如下图所示。



2、排队器

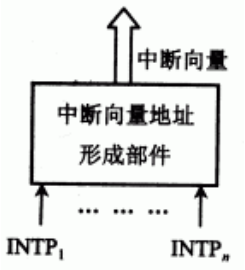
当多个中断源同时向 CPU 提出请求时，CPU 只能按中断源的不同性质对其排队，给于不同等级的优先权，并按优先等级的高低予以响应。就 I/O 中断而言，速度越高的 I/O 设备其优先级越高，因为若 CPU 不及时响应高速 I/O 的请求，其信息可能立即会丢失。

设备优先权的处理可以采用硬件办法，也可采用软件办法。硬件排队器的实现方法很多，即可在 CPU 内部设置一个统一的排队器， 对所有中断源进行排队，也可在接口电路内分别设置各个设备的排队器，下图是设在各个接口电路中的排队器电路，又叫链式排队电路。

3、中断向量地址形成部件(设备编码器)

CPU 一旦响应了 I/O 中断，就要暂停现行程序，转去执行该设备的中断服务程序。不同的设备有不同的中断服务程序，每个服务程序都有一个入口地址，CPU 必须找到这个入口地址。

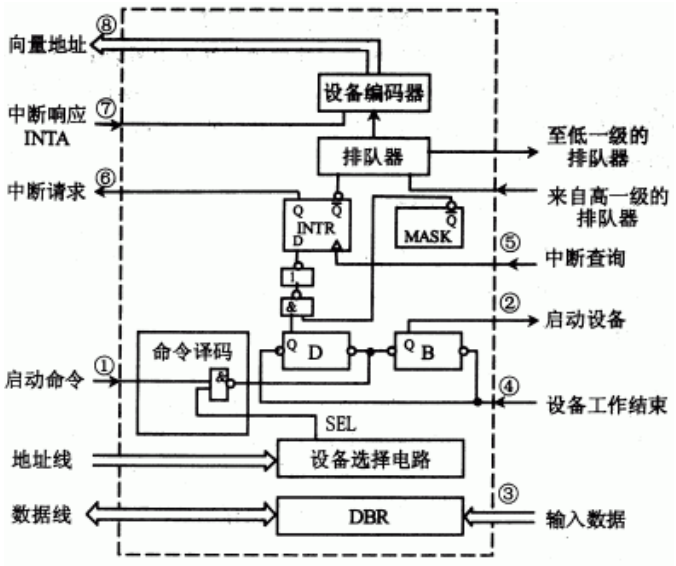
入口地址的寻找也可用硬件或软件的方法来完成，这里只介绍硬件向量法。所谓硬件向量法就是通过向量地址来寻找设备的中断服务程序入口地址，而且向量地址是由硬件电路产生的，如下图所示。



中断向量地址形成部件的输入是来自排队器的输出 $INTP_1$……$INTP_n$，输出是用二进制表示的中断向量，其位数与计算机可以处理中断源的个数有关，即一个中断源对应一个向量地址。可见，该部件实质是一个编码器，在 I/O 接口中的编码器又叫设备编码器。

4、程序中断方式接口电路的基本组成

程序中断方式接口电路的基本组成如下图所示。



（四）I/O 中断处理过程

1、CPU 响应中断的条件和时间

CPU 响应 I/O 提出中断请求的条件是必须满足 CPU 中的允许中断触发器 EINT 为“1”。该触发器可用开中断指令置位(称作开中断)；也可用关中断指令或硬件自动使其复位(称作关中断)。

由前述分析可知，I/O 设备准备就绪的时间(即 D＝1)是随机的，而 CPU 是在统一的时刻(每条指令执行阶段结束前)向接口发中断查询信号，以获取 I/O 的中断请求。因此，CPU 响应中断的时间一定是在每条指

令执行阶段的结束时刻。

2、I/O 中断处理过程

中断处理的全过程。当通过 I/O 指令的地址码选中某设备后，则

①由 CPU 发启动外设命令，将接口中 B 置“1”，D 置“0”；

②接口启动输入设备开始工作；

③输入设备将数据送入 DBR；

④输入设备向接口发“设备工作结束”信号，使 D 置“1”，B 置“0”，标志设备准备就绪；

⑤当设备准备就绪(D＝1)，且本设备未被屏蔽(MASK=0)时，在指令执行阶段的结束时刻，由 CPU 发中断查询信号；

⑥设备中断请求触发器 INTR 被置“1”，标志设备向 CPU 提出中断谓求。与此同时，INTR 送至排队器，进行中断判优；

⑦若 CPU 允许中断（ENIT＝1），设备又被排队选中，即进入中断响应阶段，由中断响应信号 INTA 将排队器输出送至编码器形成向量地址；

⑧向量地址送至 PC，作为下—条指令的地址；

⑨由于向量地址中存放的是一条无条件转移指令(见图 5．40)，故这条指令执行结束后，即无条件转至该设备的服务程序入口地址，开始执行中断服务程序，进入中断服务阶段；

⑩中断服务程序的最后一条指令执行结束，即中断返回至原程序的断点处。至此，一个完整的程序中断处理过程即告结束。

综上所述，**可将一次中断处理过程简单地归纳为中断请求、中断判优、中断响应、中断服务和中断返回五个阶段。**

（五）中断服务程序流程

一般中断服务程序的流程分四大部分：保护现场、中断服务、恢复现场和中断返回。

1、保护现场

保护现场有两个含意，其一是保存程序的断点；其二是保存通用寄存器和状态寄存器的内容。前者由中断隐指令完成，后者由中断服务程序完成。具体而言，可在中断服务程序的起始部分安排若干条存数指令，将寄存器的内容存至存储器中保存，或用进栈指令(PUSH)将各寄存器的内容推入堆栈保存，即将程序中断时的“现场”保存起来。

2、中断服务(设备服务)

这是中断服务程序的主体部分，不同的中断请求源其中断服务操作内容是不同的， 如打印机要求 CPU 将需打印的一行字符代码，通过接口送入打印机的缓冲存储器中，以供打印机打印；又如显示设备要求 CPU 将需显示的一屏字符代码， 通过接口送入显示器的显示存储器中。

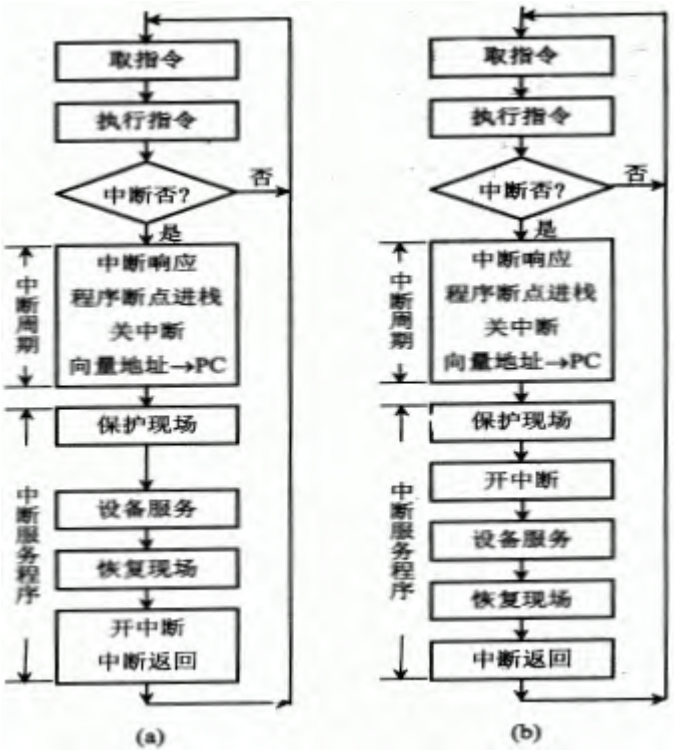
3、恢复现场

这是中断服务程序的结尾部分，要求在退出服务程序前，将源程序中断时的“现场”恢复到原来的寄存器中。通常可用取数指令或出栈指令(POP)， 将保存在存储器(或堆栈)中的信息送回到原来的寄存器中。

4、中断返回

中断服务程序的最后一条指令通常是一条中断返回指令，使其返回到原程序的断点处，以便继续执行原程序。计算机在处理中断的过程中，有可能出现新的中断请求， 此时如果 CPU 暂停现行的中断服务程序，转去处理新的中断请求，这种现象叫做中断嵌套，或叫做多重中断。 倘若 CPU 在执行中断服务程序时，对新的中断请求不予理睬，这种中断叫做单重中断。这两种处理方式的中断服务程序略有区别。

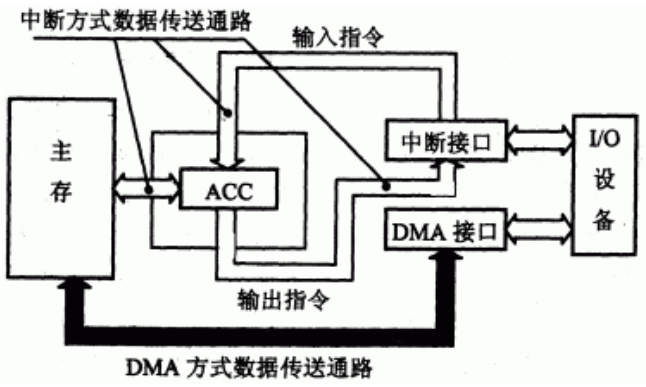
下图(a)和(b)分别为单重中断和多重中断服务程序流程。比较(a)和(b)发现其区别在于“开中断”的设置时间不同。



六、DMA 方式

（一）DMA 方式的特点

下图是 DMA 方式与程序中断方式两者数据通路的比较。

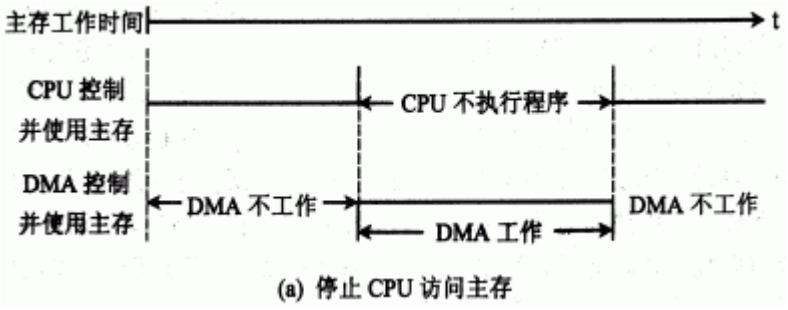


若出现高速 I/O(通过 DMA 接口)和 CPU 同时访问主存，CPU 必须将总线(如地址线、数据线)占有权让给 DMA 接口使用，即 DMA 采用周期窃取的方式占用一个存取周期。

在 DMA 方式中，由于 DMA 接口与 CPU 共享主存， 这就有可能会出现两者争用主存的冲突，为了有效地分时使用主存， DMA 通常与主存交换数据时可采用如下三种方法。

1、停止 CPU 访问主存

当外设要求传送一批数据时，由 DMA 接口向 CPU 发一个停止信号，要求 CPU 放弃地址线、数据线和有关控制线的使用权。DMA 接口获得总线控制权后，开始进行数据传送，在数据传送结束后，DMA 接口通知 CPU 可以使用主存，并把总线控制权交回给 CPU，下图(a)是该方式的时间示意图。



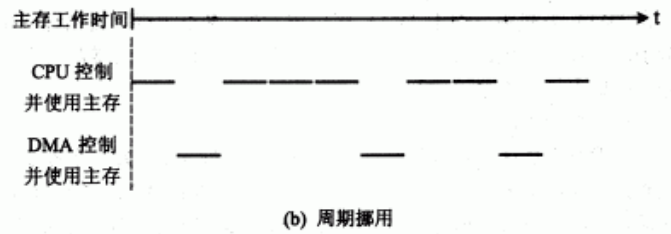
(a) 停止 CPU 访问主存

这种方式的**优点**是控制简单，适用于数据传输率很高的 I/O 设备实现成组数据的传送。**缺点**是 DMA 接口在访存时，CPU 基本上处于不工作状态或保持原状态。而且即使 I/O 设备高速运行，但其两个数据之间的准备间隔时间也总大于一个存取周期，因此，CPU 对主存的利用率并没得到充分的发挥。如软盘读一个 8 位二进制数大约需要 32μs，而半导体存储器的存取周期大大小于 1μs。为此在 DMA 接口中，一般设有一个小容量存储器(这种存储器是半导体芯片制作的)，使 I/O 设备首先与小容量存储器交换数据，然后由小容量存储器与主存交换数据，这便可减少 DMA 传送数据时占用总线的时间，即可减少 CPU 的暂停工作时间。

2、周期挪用(或周期窃取)

在这种方法中，每当 I/O 设备发出 DMA 请求时，I/O 设备便挪用或窃取总线占用权一个或几个主存周期，而 DMA 不请求时，CPU 仍继续访问主存。

I/O 设备要求 DMA 传送会遇到三种情况，一种是此时 CPU 不需访问主存(如 CPU 正在执行乘法指令，由于乘法指令执行时间较长，此时 CPU 不需访问主存)，故 I/O 设备访存与 CPU 不发生冲突。第二种情况是 I/O 设备要求 DMA 传送时，CPU 正在访存，此时必须待存取周期结束时刻，CPU 才能将总线占有权让出。第三种情况是 I/O 设备要求访存时，CPU 也要求访存，这就出现了访存冲突。此刻，I/O 访存优先于 CPU 访存，因为 I/O 不立即访存就可能丢失数据，这时 I/O 要窃取一二个存取周期，意味着 CPU 在执行访存指令过程中插入了 DMA 请求，并挪用了一二个存取周期，使 CPU 延缓了一二个存取周期再访存。图 (b)示意了 DMA 周期挪用的时间对应关系。



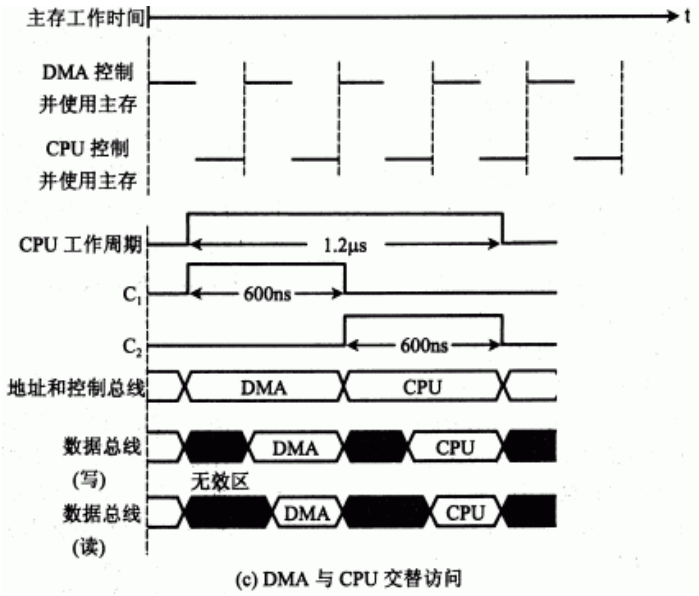
(b) 周期挪用

与 CPU 暂停访存的方式相比，它既实现了 I/O 传送，又较好地发挥了主存与 CPU 的效率，是一种广泛采用的方法。

3、DMA 与 CPU 交替访问

这种方法适合于 CPU 的工作周期比主存存取周期长的情况。例如 CPU 的工作周期为 1.2μs，主存的存取周期小于 0.6μs，那么可将一个 CPU 周期分为 C1 和 C2 两个分周期，其中 C1 专供 DMA 访存，C2 专供 CPU 访存，如图(c)所示。

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 分别控制的。CPU 与 DMA 接口各自有独立访存地址寄存器、数据寄存器和读写信号。在这种工作方式下，CPU 既不停止主程序的运行也不进入等待状态，在 CPU 不知不觉中完成了 DMA 的数据传送，故又有“透明的 DMA”方式之称，当然其相应的硬件逻辑变得更为复杂。



(c) DMA 与 CPU 交替访问

（二）DMA 接口的功能和组成

1、DMA 接口的功能

利用 DMA 方式传送数据时，数据的传输过程完全由 DMA 接口电路控制，故 DMA 接口又有 DMA 控制器之称。DMA 接口应具有如下几个功能：

(1) 向 CPU 申请 DMA 传送；

(2) 在 CPU 允许 DMA 工作时，处理总线控制权的转交，避免因进入 DMA 工作而影响 CPU 正常活动或引起总线竞争；

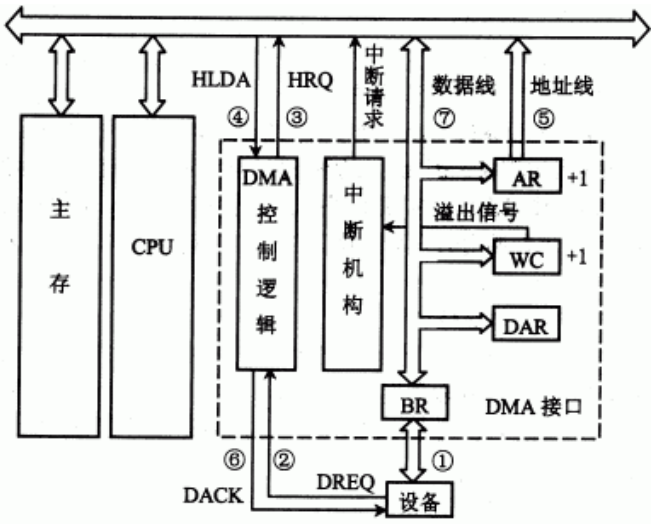
(3)在 DMA 期间管理系统总线，控制数据传送；

(4)确定数据传送的起始地址和数据长度，修正数据传送过程中的数据地址和数据长度。

(5)在数据块传送结束时，给出 DMA 操作完成的信号。

2. DMA 接口基本组成

最简单的 DMA 接口组成原理如下图所示。



它由以下几个逻辑部件所组成。

(1)主存地址寄存器 AR

AR 用于存放主存中需要交换数据的地址。在 DMA 传送前，须通过程序将数据在主存中的首地址送到主存地址寄存器。在 DMA 传送过程中，每交换一次数据，将地址寄存器内容加 1，直到一批数据传送完毕为止。

(2)字计数器 WC

WC 用于记录传送数据的总字数，通常以交换字数的补码值预置。在 DMA 传送过程中，每传送一个字，字计数器加 1，直到计数器为 0，即最高位产生进位时，表示该批数据传送完毕。于是 DMA 接口向 CPU 发中断请求信号。

(3)数据缓冲寄存器 BR

BR 用于暂存每次传送的数据。通常 DMA 接口与主存之间采用字传送，而 DMA 与设备之间可能是字节或位传送。因此 DMA 接口中还可能包括有装配或拆卸字信息的硬件逻辑，如数据移位缓冲寄存器、字节计数器等。

(4)DMA 控制逻辑

它用于负责管理 DMA 的传送过程，由控制电路、时序电路及命令状态控制寄存器等组成。每当设备准备好一个数据字(或一个字传送结束)，就向 DMA 接口提出申请(DREQ)，DMA 控制逻辑便向 CPU 请求 DMA 服务，发出总线使用权的请求信号(HRQ)。待收到 CPU 发出的响应信号 HLDA 后，DMA 控制逻辑便开始负责管理 DMA 传送的全过程，包括对主存地址寄存器和字计数器的修改、识别总线地址、指定传送类型(输入或输出)以及通知设备已经被授予一个 DMA 周期(DACK)等。

(5)中断机构

当字计数器溢出(全“0”)时，表示一批数据交换完毕，由“溢出信号”通过中断机构向 CPU 提出中断请求，请求 CPU 作 DMA 操作的后处理。必须注意，这里的中断与上一节介绍的 I/O 中断的技术相同，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批数据传送结束。它们是 I/O 系统中不同的中断事件。

(6)设备地址寄存器 DAR

DAR 存放 I/O 设备的设备码或表示设备信息存储区的寻址信息，如磁盘数据所在的区号、盘面号和柱面号。具体内容取决于设备的数据格式和地址的编址方式。

（三）DMA 的工作过程

1、DMA 传送过程

DMA 的数据传送过程分预处理、数据传送和后处理三个阶段。

(1)预处理

在 DMA 接口开始工作之前，CPU 必须给它预置如下信息：

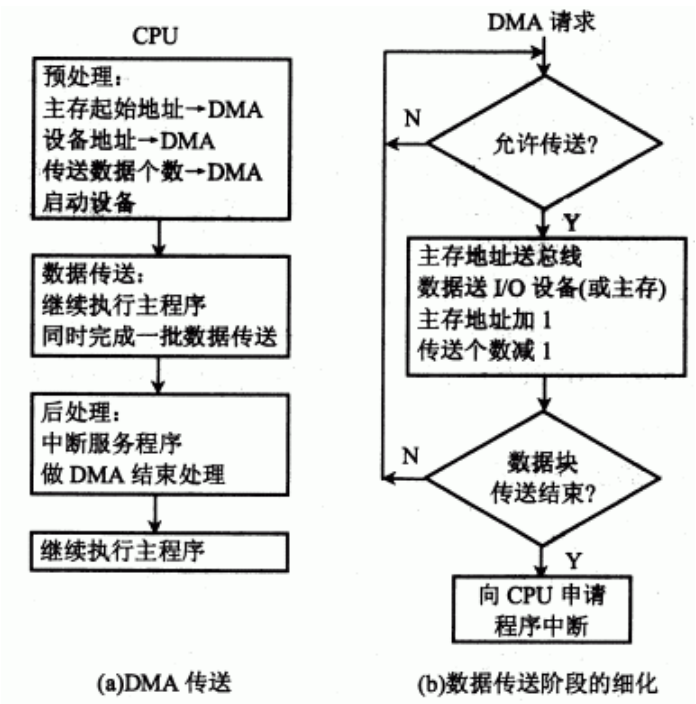
- 给 DMA 控制逻辑指明数据传送方向是输入(主存写)还是输出(主存读)；
- 向 DMA 设备地址寄存器送入设备号，并启动设备；
- 向 DMA 主存地址寄存器送入交换数据的主存起始地址；
- 对字计数器赋以交换数据的个数。

上述工作由 CPU 执行几条输入输出指令完成，即程序的初始化阶段。这些工作完成后，CPU 继续执行原来的程序。

当外部设备准备好发送的数据(输入)或上次接受的数据已经处理完毕(输出)时，它便通过 DMA 接口向 CPU 提出占用总线的申请，若有多个 DMA 同时申请，则按轻重缓急由硬件排队判优逻辑决定优先等级。待设备得到主存总线的控制权后，数据的传送便由该 DMA 接口进行管理。

(2)数据传送

DMA 方式是以数据块为单位传送的，以周期挪用的 DMA 方式为例，其数据传送的流程可用下图示意。



(a)DMA 传送　　(b)数据传送阶段的细化

以数据输入为例，具体操作如下：

①从设备读入一个字到 DMA 的数据缓冲寄存器 BR 中，表示数据缓冲寄存器“满”(如果 I/O 设备是面向字符的，则一次读入一个字节，组装成一个字)；

②设备向 DMA 接口发请求(DREQ)；

③DMA 接口向 CPU 申请总线控制权(HRQ)；

④CPU 发回 HLD 信号，表示允许将总线控制权交给 DMA 接口使用；

⑤将 DMA 主存地址寄存器中的主存地址送地址总线；

⑥通知设备已被授予一个 DMA 周期（DACK），并为交换下一个字做准备；

⑦将 DMA 数据缓冲寄存器的内容送数据总线；

⑧命令存储器作写操作；

⑨修改主存地址和字计数值；

⑩判断数据块是否传送结束，若未结束，则继续传送；若已结束，(字计数器溢出)，则向 CPU 申请程序中断，标志数据块传送结束。

若为输出数据，则应完成以下操作：

①当 DMA 数据缓冲寄存器已将输出数据送至加设备后，表示数据缓冲寄存器已“空”；

②设备向 DMA 接口发请求(DREQ)：

③DMA 接口向 CPU 申请总线控制权(HRQ)；

④CPU 发回 HLDA 信号，表示允许将总线控制权交给 DMA 接口使用；

⑤将 DMA 主存地址寄存器中的主存地址送地址总线，并命令存储器读；

⑥通知设备已授予一个 DMA 周期(DACK)，并为交换下一个字做准备；

⑦主存将相应地址单元的内容通过数据总线读入到 DMA 的数据缓冲寄存器中；

⑧将 DMA 数据缓冲寄存器的内容送到输出设备，若为字符设备，则需将其拆成字符输出；

⑨修改主存地址和字计数值；

⑩判断数据块是否已传送完毕，若未完，继续传送；若已送完，则向 CPU 申请程序中断。

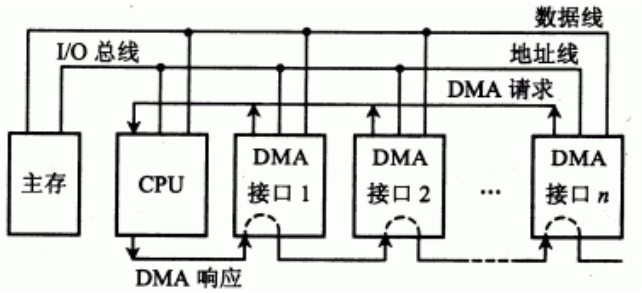
(3)后处理

当 DMA 的中断请求得到响应后，CPU 停止源程序的执行，转去执行中断服务程序，做一些 DMA 的结束工作。它包括校验送入主存的数据是否正确；决定是否继续用 DMA 传送其他数据块，若继续传送，则又

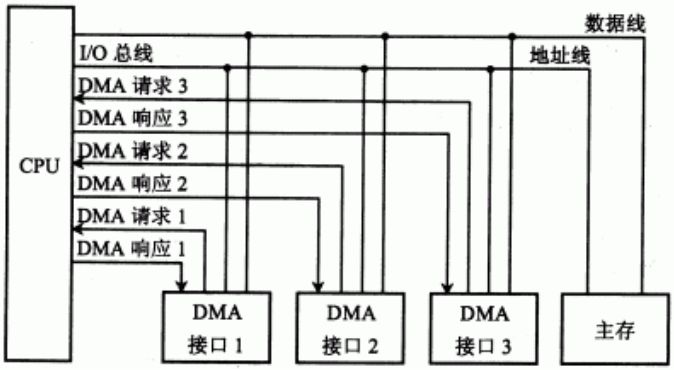
要对 DMA 接口进行初始化，若不需要传送，则停止外设；测试在传送过程中是否发生错误，若出错，则转错误诊断及处理错误程序。

2、DMA 接口与系统的连接方式

DMA 接口与系统的连接方式有两种。



上图为具有**公共请求线的 DMA 请求方式**，若干个 DMA 接口通过一条公用的 DMA 请求线向 CPU 申请总线控制权。CPU 发出响应信号用链式查询方式通过 DMA 接口，首先选中的设备获得总线控制权，即可占用总线与主存传送信息。



上图是**独立的 DMA 请求方式**，每一个 DMA 接口各有一对独立的 DMA 请求线和 DMA 响应线，它由 CPU 的优先级判别机构裁决首先响应哪个请求，并在响应线上发出响应信号，被获得响应信号的 DMA 接口使可控制总线与主存传送数据。

3、DMA 小结

与程序中断方式相比，DMA 方式有如下特点：

(1)从数据传送看，程序中断方式靠程序传送，DMA 方式靠硬件传送。

(2)从 CPU 响应时间看，程序中断方式是在一条指令执行结束时响应，而 DMA 方式可在指令周期内的任一存取周期结束时响应。

(3)程序中断方式有处理异常事件的能力，DMA 方式没有这种能力，它主要用于大批数据的传送，如硬盘存取、图像处理、高速数据采集系统等，可提高数据吞吐量。

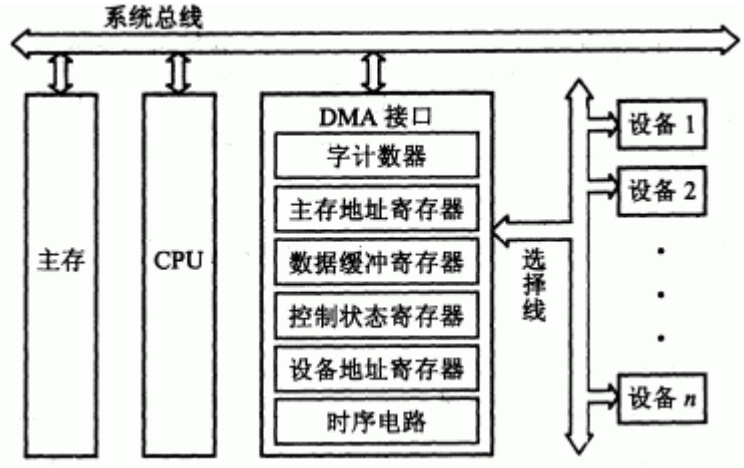
(4)程序中断方式要中断现行程序，故需保护现场，DMA 方式不中断现行程序，无需保护现场。

(5)DMA 的优先级比程序中断高。

（四）DMA 接口的类型

1、选择型 DMA 接口

这种类型的 DMA 接口的主要特点是在物理上可连接多个设备，在逻辑上允许连接一个设备，即在某一个时间内，DMA 接口只能为一个设备服务，关键是在预处理时将所选设备的设备号送入设备地址寄存器，下图是选择型 DMA 接口的逻辑框图，选择型 DMA 接口特别适用于数据传输率很高的设备。



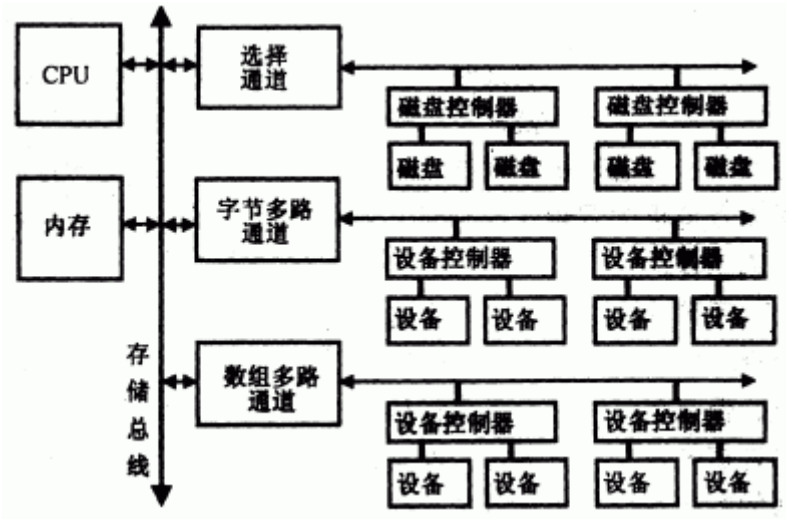
2、多路型 DMA 接口

多路型 DMA 接口不仅在物理上可以连接多个设备，而且在逻辑上也允许多个设备同时工作，各个设备采用字节交叉的方式通过 DMA 接口进行数据传送。在多路型 DMA 接口中，为每个与它连接的设备部设置了一套寄存器，分别存放各自的传送参数。

七、通道方式

（一）通道的类型

下图是通道结构的例子。采用这种结构的计算机中有两种总线：一种是存储器总线， 它承担 CPU 与内存、通道之间的数据传输任务；另一种是承担输入输出操作的总线，即通道总线。



1、选择通道

对于高速的设备，如磁盘等，要求较高的数据传输速度。对于这种高速传输，通道难以同时对多个这样的设备进行操作，只能一次对一个设备进行操作。这种通道称为选择通道，它与设备之间的传输一直维持到设备请求的传输完成为止，然后为其他外围设备传输数据。选择通道的数据宽度是可变的，通道中包含一个保存输入输出数据传输所需的参数寄存器。参数寄存器包括存放下一个主存传输数据存放位置的地址和对传输数据计数的寄存器。选择通道的输入输出操作启动之后，该通道就专门用于该设备的数据传输直到操作完成。选择通道的缺点是设备申请使用通道的等待时间较长。

2、数组多路通道

数组多路通道以数组(数据块)为单位在若干高速传输操作之间进行交叉复用。这样可减少外设申请使用通道时的等待时间。数组多路通道适用于高速外围设备，这些设备的数据传输以块为单位。通道用块交叉的方法，轮流为多个外设服务。当同时为多台外设传送数据时，每传送完一块数据后选择下一个外设进行数据传送，使多路传输并行进行。数组多路通道既保留了选择通道高速传输的优点，又充分利用了控制性操作的时间间隔为其它设备服务，使通道的功能得到有效发挥，因此数组多路通道在实际系统中得到较多的的应用。特别是对于磁盘和磁带等一些块设备，它们的数据传输本来就是按块进行的。而在传输操作之前又需要寻找记录的位置，在寻找的期间让通道等待是不合理的。数组多路通道可以先向一个设备发出一个寻找的命令，然后在这个设备寻找期间为其他设备服务。在设备寻找完成后才真正建立数据连接，并一直维持到数据传输完毕。因此采用数组多路通道可提高通道的数据传输的吞吐率。

3．字节多路通道

字节多路通道用于连接多个慢速的和中速的设备，这些设备的数据传送以字节为单位。每传送一个字节要等待较长时间，如终端设备等。因此，通道可以以字节交叉方式轮流为多个外设服务，以提高通道的利用率。这种通道的数据宽度一般为单字节。它的操作模式有两种：字节交叉模式和猝发模式。在字节交叉模式中，通道操作分成较短的段。通道向准备就绪的设备进行数据段的传输操作。传输的信息可由一个字节的数据以及控制和状态信息构成。通道与设备的连接时间是很短的。如果需要传输的数据量比较大，则通道转换成猝发的工作模式。在猝发模式下，通道与设备之间的传输一直维持到设备请求的传输完成为止。 通道使用一种超时机制判断设备的操作时间(即逻辑连接时间)， 并决定采用哪一种模式。如果设备请求的逻辑连接时间大于某个额定的值，通道就转换成猝发模式，否则就以字节交叉模式工作。

（二）通道的功能

衡量通道性能的指标是**通道的流量**， 它指通道在传送数据时，1 秒钟时间内传送的位数(b/s)。 通道所能达到的最大流量称为通道的极限流量。

对于采用字节多路通道，通道的极限流量应大于所接外设的字节传送速率之和，因为字节多路通道同时为多个外设传输数据；

对于采用其他两种方式的通道，通道的极限流量应大于所接外设中字节传送速率最大的设备，因为数组多路通道和选择通道是轮流为外设传输数据的

1、通道的功能

（1）接受 CPU 的输入输出操作指令，按指令要求控制外围设备。

（2）从内存中读取通道程序，并执行， 即向设备控制器发送各种命令。

（3）组织和控制数据在内存与外设之间的传送操作。 根据需要提供数据中间绥存空间以及提供数据存入内存的地址和传送的数据量。

（4）读取外设的状态信息，形成整个通道的状态信息，提供给 CPU 或保存在内存中。

（5）向 CPU 发出输入输出操作中断请求，将外围设备的中断请求和通道本身的中断请求按次序报告 CPU。

CPU 通过执行输入输出指令以及处理来自通道的中断， 实现对通道的管理。来自通道的中断有两种：一种是数据传输结束中断；另一种是故障中断。通道的管理是操作系统的任务。

2、设备控制器具体任务

（1）从通道接受通道指，控制外围设备完成指定的操作；

（2）向通道提供外围设备的状态；

（3）将各种外围设备的不同信号转换成通道能够识别的标准信号。

在具有通道的计算机中，实现数据输入输出操作的是**通道指令**。CPU 的输入输出指令不直接实现输入输出的数据传送， 而是由通道指令实现这种传送，CPU 用输入输出指令启动通道执行通道指令。CPU 的通

道输入输出指令的基本功能主要是启动、停止输入输出过程，了解通道和设备的状态以及控制通道的其他一些操作。

通道指令也叫通道控制字(CCW)， 它是通道用于放行输入输出操作的指令，可以由 CPU 存放在内存中 ，由通道处理器从内存中取出并执行。通道执行通道指令以完成输入输出传输。通道程序由一条或几条通道指令组成，也称通道指令链。

（三）通道工作过程

通道中包括通道控制器、状态寄存器、中断机构、通道地址寄存器、通道指令寄存器等。这里， 通道地址寄存器相当于一般 CPU 中的程序计数器。

CPU 在进行一个输入输出操作之前， 首先准备好通道程序，然后安排好数据缓冲区，再给通道和设备发启动命令。 CPU 准备好的通道程序存放在内存中，由通道控制器读取并执行。

通道接到启动信号后，首先到指定的内存单元中取通道地址字，放在通道地址寄存器中。这个存放通道地址字的内存单元的地址可以是固定的，然后根据通道地址寄存器中的值到内存中去取第一条通道指令，并放在通道指令寄存器中。

通道程序执行时通过在通道指令寄存器中的相应位进行设置来告诉通道指令执行机构在执行完成当前指令后，自动转入下一条指令或者结束数据传送过程。通道程序的最后一条指令是一条结束指令，通道在执行到这条结束指令时就不再取下一条指令，而是通知设备结束操作。在通道程序执行完毕后，由通道向 CPU 发中断信号， 并将通道状态字写入内存专用单元， CPU 根据通道状态字分析这次输入输出操作的执行情况。

第五章 输入输出系统 关键词汇

1、程序查询方式

程序查询方式是主机与外设间进行信息交换的最简单方式，程序查询方式的核心问题在于需要不断地查询 I/O 设备是否准备就绪。由 CPU 执行一段输入输出程序来实现主机与外设之间数据传送的方式叫做程序直接控制方式。根据外设的不同性质，这种传送方式又可分为无条件传送和程序查询方式两种。

2、中断方式

在程序查询方式使 CPU 循环等待，造成了 CPU 资源的浪费。中断传送方式很好地解决了这个问题，在外设没有做好数据传送准备时，CPU 可以运行与传送数据无关的其他指令，外设做好传送准备后，主动向 CPU 提出申请，若 CPU 响应这一申请，则暂停正在运行的程序，转去执行数据输入/输出操作的指令，数据传送完毕后返回，CPU 继续执行原来运行的程序，这样使得外设与 CPU 可以并行工作，提高了系统的效率。

3、多重中断

多重中断是指在处理某一中断过程中，又有比该中断优先级高的中断请求，于是 CPU 中断原中断服务程序的执行，而又转去执行新的中断服务程序。这种多重中断的执行形成了中断嵌套。

4、中断屏蔽

当多个中断源发出中断请求时，CPU 只能响应一个中断，中断屏蔽可实现部分中断的封锁。每个中断源在配有一个中断请求触发器的同时，配有一个中断屏蔽触发器(MASK)，它们成对出现，当 MASK 置 1 时，该中断源的请求被屏蔽，中断请求不能进入中断排队逻辑，进行判优：当 MASK 置 0 时，该中断请求被允许，经过中断排队判优后，最终送往 CPU。

5、DMA 方式

DMA 方式即直接存储器访问(Direct Memory Access)方式，是为了在主存与外设之间实现高速、批量数据交换而设置的。DMA 方式的数据传送直接依靠硬件(DMA 控制器)来实现，不需要执行任何程序。

6、主存地址计数器

用来存放待交换数据的主存地址。该计数器的初始值为主存缓冲区的首地址，当 DMA 传送时，每传送一个数据，将地址计数器加“1”，从而以增量方式给出主存中要交换的一批数据的地址，直至这批数据传送完毕为止。

7、I/O 设备

I/O 设备按功能分类，可分为两类：输入设备，输出设备。

输入设备是指向主机输入程序、原始数据和操作命令等信息的设备。这些记录在载体上的信息，可以是数字、符号，甚至是图形、图像及声音。输入设备将其变换成主机能识别的二进制代码，并负责传送到主机。

输出设备将计算机处理过的二进制代码信息，转换成用户能识别的形式，如数字、符号、文字、图形、图像或声音等输出出来。

8、I/O 系统

把外围设备、接口部件及相应的管理软件，定义为计算机的输入输出系统，简称 I/O 系统。

9、接口

是各设备之间连接的部分。I/O 接口是指主机与外设之间或计算机之间通过总线相连的转接器或适配器，是信息交换窗口，起缓冲作用，能对传送的信息进行差错控制，能控制信息流程，对保证 I/O 正确操作起重要作用。

10、统一编址

统一编址就是将 I/O 地址看作是存储器地址的一部分。如在 64K 地址的存储空间中，划出 8K 地址作为 I/O 的地址，凡是在这 8K 地址范围内的访问，就是对 I/O 的访问， 所用的指令与访存指令相似。

11、不统一编址

不统一编址就是指 I/O 地址和存储器地址是分开的，所有对 I/O 的访问必须有专用的 I/O 指令。

第五章 输入输出系统 FAQ

一、计算机 I/O 数据传送控制方式通常可分为哪几种？

（1） 程序查询方式：CPU 的操作和外围设备的操作能够同步，而且硬件结构比较简单

（2） 程序中断方式：一般适用于随机出现的服务，且一旦提出要求应立即进行，节省了 CPU 的时间，但硬件结构相对复杂一些。

（3） 直接内存访问（DMA）方式：数据传输速度很高，传输速率仅受内存访问时间的限制。需更多硬件，适用于内存和高速外设之间大批交换数据的场合。

（4） 通道方式：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，大大提高了 CPU 的工作效率。

（5） 外围处理机方式：通道方式的进一步发展，基本上独立于主机工作，结果更接近一般处理机。

二、在输入输出系统中，ＤＭＡ方式是否可以替代中断方式？

DMA 方式不可以替代中断方式，因为在 DMA 方式的传送结束阶段，还需要向 CPU 发出中断请求。

三、DMA 方式和中断控制方式的不同点是什么？

（1） 中断方式通过程序实现数据传送，而 DMA 方式不使用程序，直接靠硬件来实现。

（2） CPU 对中断的响应是在执行完一条指令之后，而对 DMA 的响应则可以在指令执行过程中的任何两个存储周期之间。

（3） 中断方式不仅具有数据传送能力，而且还能处理异常事件；DMA 只能进行数据传送。

（4） 中断方式必须切换程序，要进行 CPU 现场的保护和恢复操作；DMA 仅挪用了一个存储周期，不改变 CPU 现场。

（5） DMA 请求的优先权比中断请求高。CPU 优先响应 DMA 请求，是为了避免 DMA 所连接的高速外设丢失数据。

（6）

四、什么是输入输出操作？

主存与外围设备之间的信息传送操作称为输入输出操作。

五、什么是查询输入输出方式，它有什么特点？

查询输入输出方式是指 CPU 在与外设交换前先检测外设的状态线，如果外设准备好与 CPU 交换数据则通过状态线通知 CPU，CPU 在检测到外设准备好后再与外设交换数据。查询输入输出方式除了接口中有数据线外，还有状态线。

六、中断方式的原理是怎样的？

在程序中安排一条指令，发出 START 信号启动外围设备，然后机器继续执行程序。当外围设备完成数据传送的准备后，便向 CPU 发“中断请求”信号。CPU 接到请求后若可以停止正在运行的程序，则在一条指令执行完后，转去执行中断服务程序，完成传送数据工作。传送完毕后仍返回原来的程序。

七、向量中断和非向量中断有何区别？

向量中断是指在提出中断请求的同时，能通过硬件向 CPU 提供中断服务程序入口地址；而非向量中断，中断源不提供中断服务程序入口地址，而由 CPU 自行预先确定，并用程序查询中断源。

八、I/O 软件的主要任务是什么？

①如何将用户编制的程序(或数据)输入至主机内；

②如何将运算结果输送给用户；

③如何实现 I/O 系统与主机工作的协调等。

九、I/O 设备寻址的意义何在？

由于每台 I/O 设备都赋予一个设备号，因此，当要启动某一设备时，可由 I/O 指令的设备码字段直接指出该设备的设备号。通过接口电路中的设备选择电路，便可选中要交换信息的设备。

十、程序查询方式 I/O 流程如何？

程序查询方式是由 CPU 通过程序不断查询 I/O 设备是否已做好难备，从而控制 I/O 与主机交换信息。采用这种方式实现主机和 I/O 交换信息，要求 I/O 接口内设置一个能反映设备是否准备就绪的状态标记，CPU 通过对此标记的检测，可得知设备的准备情况。如若 CPU 欲从某一外设读数据块(例如从磁带上读一记录块)至主存，当现行程序需启动某设备工作时，即将此程序流程插入到运行的程序中。CPU 启动 I/O 后便开始对 I/O 的状态进行查询。若查得 I/O 未准备就绪，就继续查询；若查得 I/O 准备就绪，就将数据从 I/O 接口送至 CPU，再由 CPU 送至主存。这样一个字一个字地传送，直至这个数据块的数据全部传送结束，CPU 又重新回到原现行程序。

十一、程序中断方式 I/O 流程如何？

CPU 在启动 I/O 设备后，对设备是否已准备就绪不加过问，继续执行自身程序，只是当 I/O 设备准备就绪并向 CPU 发出中断请求后才予理睬，这将大大提高 CPU 的工作效率。CPU 启动 I/O 后仍继续执行原程序，在第 K 条指令执行结束后，CPU 响应了 I/O 的请求，中断了现行程序，转至中断服务程序，等处理完后又返回到原程序断点处，继续从第 K+1 条指令往下执行。由于这种方式使原程序中断了运行，故叫程序中断方式。

十二、DMA 方式 I/O 流程的提出。

在 DMA 方式中，主存与 I/O 设备之间有一条数据通路，主存与 I/O 设备交换信息时，无需处理中断服务程序。若出现 DMA 和 CPU 同时访问主存，CPU 总是将总线占有权让给 DMA，通常把 DMA 的这种占有叫做“窃取”或“挪用”。窃取的时间一般为一个存储周期，故又把 DMA 占用的存取周期叫做“窃取周期”或“挪用周期”。而且，在 DMA 窃取存取周期时，CPU 尚能继续作内部操作(如乘法运算)。可见，DMA 方式与程序查询和程序中断方式相比，又进一步提高了 CPU 的资源利用率。

十三、通道方式与 DMA 方式不同点？

（1）在 DMA 方式中，数据的传送方向、存放数据的内存始址以及传送的数据块长度等都由 CPU 控制，而在通道方式中，这些都由通道来进行控制。

（2）DMA 方式每台设备至少需要一个 DMA 控制器，一个通道控制器可以控制多台设备，显然，通道方式进一步减轻了 CPU 的工作负担和增加了计算机系统的并行工作程度。

十四、通道有哪些分类？解决通道不足的问题？

按信息交换方式和连接的外设类型不同，通道可分为三种类型：

（1）字节多路通道：以字节为单位传送数据，它主要用来连接大量的低速设备，如终端、打印机等。当一台设备传送一个字节后，立即转去为另一设备传送一个字节。

（2）选择通道：它用于连接磁带、磁鼓和磁盘等设备，以块为单位成批传送数据，但一次只能执行一道通道程序，控制一台设备进行 I/O 操作，当一个 I/O 请求操作完成后，再选择与通道相连的另一设备。

（3）数组多路通道：它先为一台设备执行一条通道命令，然后自动转换，为另一台设备执行一条通道命令。数组多路通道的实质是：对通道程序采用多道程序设计技术的硬件实现。

为了解决通道不足的矛盾，使设备能得到充分利用，较经济的方法是：

①减少使用通道的时间；

②增加通路，提高通道的灵活性。

第五章 输入输出系统 拓展资源

—— 多媒体技术

视频和音频数据的压缩和解压缩技术。多媒体计算机的关键问题是如何实时综合处理声、图和文字信息，需要将每幅图像从模拟量转换成数字量，然后进行图像处理，与图形、文字复合后存放在机器中。数字化图像和声音的信息量是非常大的。以一般彩色电视信号为例，设代表光强、色彩和色饱和度的 YIQ 色空间中各分量的带宽分别为 4.2MHz、1.5MHz 和 0.5MHz。根据采样原理，仅当采样频率＞2 倍的原始信号的频率时，才能保证采样后信号可被保真地恢复为原始信号。再设各分量均被数字化为 8 个比特，从而 1 秒钟的电视信号的数据总量应为： (4.2+1.5+0.5)×2×8＝99.2 (Mbits)

也就是说，彩电信号的数据量约每秒为 100Mbits，因而一个容量为 1GB 的 CD-ROM 仅能存放约一分钟的原始电视数据(每字节后面附有 2 位校验位)，很显然电视信号数字化后直接保存的方法是令人难以接受的。

对于语音的数据也一样，一般人类语音的带宽为 4KHz，同样依据采样定理，并没数字化精度为 8 比特，则一秒钟的数据量为：4K×2×8＝64Kbits，因此在上述采样条件下，讲一分钟话的数据量约为 480KB。

由此可见，电视图像、彩色图像、彩色静图像、文件图像以及语音等数据量是相当大的。特别是电视图像的数据量，在相同条件下要比语音数据量大 1000 倍。再加上计算机总线的传输率也跟不上，因此，必须对信息进行压缩和解压缩。所谓图像压缩是指图像从像素存储的方式，经过图像变换、量化和高效编码等处理，转换成特殊形式的编码，从而大大降低计算机所需存储和实时传送的数据量。 例如 Intel 公司的交互式数字视频系统 DVI 能将动态图像数据压缩到 135KB/S 的传送速度。

信息编码方式很多，应选用符合国际标准的，并能用计算机或 VLSI 芯片快速实现的编码方法。

多媒体专用芯片。由于多媒体计算机承担大量与数据信号处理、图像处理、压缩与解压缩以及解决多媒体之间关系等有关的问题，而且要求处理速度快，因此需研制专用芯片。一般多媒体专用芯片有两种类型：

固定功能的和可编程的。

大容量存储器。多媒体计算机需要存储的信息量极大，因此研制大容量的存储器仍是多媒体计算机系统的关键技术。

适用于多媒体技术的软件。下图示意了多媒体系统的层次结构。

应用系统
创作系统
多媒体核心系统
多媒体入/出控制及接口
多媒体实时压缩与解压缩
计算机硬件

最底层为计算机硬件，还可配置电视机、录像机及音像设备等。其上层是多媒体实时压缩和解压缩层，它将视频和音频信号压缩后存储在盘上，播放时要解压缩，而且要求处理速度快，通常采用以专用芯片为基础的电路卡。

多媒体入/出控制及接口层与多媒体设备打交道， 驱动控制这些硬件设备，并提供与高层软件的接口。

多媒体核心系统层是多媒体操作系统， Intel、IBM、Microsoft 和 Apple 等公司都开发了这层软件。

创作系统层是为方便用户开发应用系统而设置的，具有编辑和播放等功能。

应用系统层包括厂家或用户开发的应用软件。

以上除最底层的硬件层外，其他层次都包含适用于多媒体技术的软件。

第六章 计算机的运算方法 课堂笔记

◆ **主要知识点掌握程度**

本章主要介绍参与运算的各类数据（包括无符号数和有符号数；定点数和浮点数等），以及它们在计算机中的计算方法。

◆ **知识点整理**

一、无符号数

在计算机中参与运算的数有两大类：无符号数和有符号数。

计算机中的数均放在寄存器中，通常称寄存器的位数为机器字长。所谓无符号数即没有符号的数，在寄存器中的每一位均可用来存放数值。当存放有符号数时，则需留出位置存放“符号”。因此，在机器字长相同时，无符号数与有符号数所对应的数值范围不同的。以机器字长为 16 位为例，无符号数的表示范围为 0～65535，而有符号数的表示范围为-32768～+32767(此数值对应原码表示)。

二、有符号数

（一）机器数与真值

用“0”表示“正”，用“1”表示“负”，这样符号也被数字化了，并且规定将它放在有效数字的前面，这样就组成了有符号数。

把符号“数字化”的数叫做**机器数**，而把带“十”或“—”符号的数叫做**真值**。一旦符号数字化后，符号和数值就形成了一种新的编码。在运算过程中，符号位能否和数值部分一起参加运算？如果参加运算，符号位又需作哪些处理？这些问题都与符号位和数值位所构成的编码有关，这些编码就是原码、补码、反码和移码。

（二）原码表示法

原码是机器数中最简单的一种表示形式，其符号位为 0 表示正数，符号位为 1 表示负数，数值位即真值的绝对值，故原码表示又称作带符号的绝对值表示。

1、整数原码的定义为

$$[x]_{原}=\begin{cases}0,x\\2^n-x\end{cases}$$

式中 x 为真值，n 为整数的位数。

例如，当 x=+1110 时，$[x]_{原}$=0,1110

当 x=-1110 时，$[x]_{原}=2^4-(-1110)=1,1110$

小数原码的定义为:

$$[x]_{原}=\begin{cases}x & 1>x\geq 0\\1-x & 0\geq x>-1\end{cases}$$

例如，当 x=+0.1101 时，$[x]_{原}$=0.1101

当 x=-0.1101 时，$[x]_{原}$=1-(-0.1101)＝1.1101

根据定义，已知真值可求原码，反之已知原码也可求真值。如：当$[x]_{原}$=1.0011 时，由定义得 x=1-$[x]_{原}$=1-1.0011=-0.0011

当$[x]_{原}$=1,1100 时，由定义得 $x=2^n-[x]_{原}=2^4-1,1100=10000-11100=-1100$

当$[x]_{原}$=0.1101 时，x=0.1101

2、“0”的原码表示法

当 x=0 时,$[+0.0000]_{原}$=0.0000

$[-0.0000]_{原}$=1-(0.0000)=1.0000

可见$[+0]_{原}$不等于$[-0]_{原}$，即**原码中的“零”有两种表示形式**。

3、原码的表数范围

对于定点整数：

一个 n+1 位原码能表示的最大正数为 01…11，即 2^n-1；能表示的最小数为绝对值最大的负数 111…1，即$-(2^n-1)$。所以原码能表示的数值范围为： $-(2^n-1)\leqslant x\leqslant 2^n-1$。

对于定点小数：

一个 n+1 位定点小数原码能表示的最大正数为 0.1…11,即 $1-2^{-n}$；能表示的最小数为绝对值最大的负数为 1.11…1，即$-(1-2^{-n})$。定点小数原码的数值范围为： $-(1-2^{-n})\leqslant x\leqslant 1-2^{-n}$。

（三）补码表示法

1、补数的概念

在日常生活中，常会遇到“补数”的概念。如时钟指示 6 点，欲使它指示 3 点，既可按顺时针方向将分针转 9 圈，又可按逆时针方向将分针转 3 圈，结果是一致的。

- 一个负数可用它的正补数来代替，而这个正补数可以用模加上负数本身求得。
- 两个互为补数的数，它们绝对值之和即为模数。
- 正数的补数即该正数本身。

将补数的概念用到计算机中，使出现了补码这种机器数。

2、补码的定义

整数补码的定义为:

$$[x]_{补}=\begin{cases}0,x & 2^n>x\geq 0\\ 2^{n+1}+x & 0\geq x>-2^n(\mathrm{mod}\ 2^{n+1})\end{cases}$$

式中 x 为真值，n 为整数的位数。

例如，当 x=+1010 时，

$[x]_{补}$=0, 1010

↑

用逗号将符号位和数值部分隔开

当 x=-1101 时，

$[x]_{补}=2^{n+1}+x$=100000-1101＝1, 0011

↑

用逗号将符号位和数值部分隔开

小数补码的定义为:

$$[x]_{补}=\begin{cases}x & 1>x\geq 0\\ 2+x & 0\geq x>-1(\mathrm{mod}\ 2)\end{cases}$$

式中 x 为真值。

例如，当 x=0.1001 时，$[x]_{补}$＝0.1001

当 x=-0.0110 时，

$[x]_{补}$=2+x=10.0000-0.0110＝1.1010

3、“0”的补码表示法

当 x=0 时, $[+0.0000]_{补}$=0.0000

$[-0.0000]_{补}$=2+(-0.0000)=10.0000-0.0000=0.0000

显然$[+0]_{补}=[-0]_{补}$=0.0000，即**补码中的“零”只有一种表示表示形式**。

4、补码的表数范围

一个 n+1 位整数补码能表示的最大数是 011…1，即 2^n-1；能表示的最小数为 100…0，即-2^n。所以它能表示的数值范围是：$-2^n\leqslant x\leqslant 2^n-1$

一个 n+1 位小数补码能表示的最大数是 0.11…1，即 $1-2^{-n}$；能表示的最小数为 1.00… 0，即-1。所以它能表示的数值范围是：$-1\leqslant x\leqslant 1-2^{-n}$

对于小数，若 x=-1，则根据小数补码定义，有$[x]_{补}$=2+x=10.0000-1.0000＝1.0000。可见，-1 本不属于小数范围，但却有$[-1]_{补}$存在（其实在小数补码定义中已指明），这是由于补码中的零只有一种表示形式，故它比原码能多表示一个“-1”。此外，根据补码定义，已知补码还可以求真值，如

若$[x]_{补}$=1.0101

则 x=$[x]_{补}$-2=1.0101-10.0000=-0.1011

由于负数$-x_1x_2x_3x_4$的原码为 1，$x_1x_2x_3x_4$，因此对这个负数求补，可以看作对它的原码除符号位外，每位求反，末位加 1，简称“求反加 1”。这样，由真值通过原码求补码就可避免减法运算。同理，对于小数也有同样结论。

“由原码除符号位外，每位求反，末位加 1 求补码”这一规则，同样适用于由$[x]_{补}$求$[x]_{原}$。而对于一个负数，若对其原码除符号位外，每位求反(简称“每位求反”)，或是对其补码减去末位的 1，即得机器数的反码。

（四）反码表示法

反码通常用来作为由原码求补码或者由补码求原码的中间过渡。反码的定义如下：

1、整数反码的定义

$$[x]_{反}=\begin{cases}0,x & 2^n>x\geq 0\\ (2^{n+1}-1)+x & 0\geq x>-2^n(\mathrm{mod}(2^{n+1}-1))\end{cases}$$

式中 x 为真值，n 为整数的位数。

例如，当 x=+1101 时，

$[x]_{反}$=0, 1101

↑

用逗号将符号位和数值部分隔开

当 x=−1101 时，

$[x]_{反}=(x^{4+1}-1)+x=11111-1101=1,0010$

↑

用逗号将符号位和数值部分隔开

小数反码的定义为：

$$[x]_{反}=\begin{cases}x & 1>x\geq 0\\ (2-2^{-n})+x & 0\geq x>-1(\mathrm{mod}(2-2^{-n}))\end{cases}$$

式中 x 为真值，n 为小数的位数。

例如，当 x=+0.0110 时，$[x]_{反}$=0.0110

当 x=−0.0110 时，

$[x]_{反}=(2-2^{-4})+x=1.1111-0.0110=1.1001$

2、“0”的反码表示法

当 x=0 时，$[+0.0000]_{反}$=0.0000

$[-0.0000]_{反}=(10.0000-0.0001)-0.0000=1.1111$

可见$[+0]_{反}$不等于$[-0]_{反}$，即反码中的“零”也有两种表示形式。

3、反码的表数范围

定点整数反码的数值范围为：$-(2^n-1)\leqslant x\leqslant 2^n-1$。

定点小数原码的数值范围为：$-(1-2^{-n})\leqslant x\leqslant 1-2^{-n}$。

综上所述，三种机器数的特点可归纳如下：

- 三种机器数的最高位均为符号位。符号位和数值部分之间可用“.”(对于小数)或“，”(对于整数)隔开。
- 当真值为正时，原码、补码和反码的表示形式均相同，即符号位用“0”表示，数值部分与其值相同。
- 当真值为负时，原码、补码和反码的表示形式不同，但其符号位都用“1”表示，而数值部分有如下关系，即补码是原码的“求反加 1”，反码是原码的“每位求反”。

（五）移码表示法

1、移码的定义

当真值用补码表示时，由于符号位和数值部分一起编码，与习惯上的表示法不同，因此人们很难从补码的形式上直接判断其真值的大小。

由此可得移码的定义：

$[x]_{移}=2^n+x\quad(2^n>x\geqslant -2^n)$

式中 x 为真值，n 为整数的位数。

其实移码就是在真值上加一个常数 2^n。在数轴上移码所表示的范围恰好对应与真值在数轴上的范围向轴的正方向移动 2^n 个单元。如下图所示，由此而得移码之称。

例如，x=10100

$[x]_{移}=2^5+10100=1,10100$

↑

用逗号将符号位和数值部分隔开

x=−10100

$[x]_{移}=2^5-10100=0,01100$

↑

用逗号将符号位和数值部分隔开

2、“0”的移码表示

当 x=0 时，$[+0]_{移}=2^5+0=1,00000$

$[-0]_{移}=2^5-0=1,00000$

可见[+0]移等于[-0]，即移码表示中零也是惟一的。

此外，由移码的定义可见，当 n=5 时，其最小的真值为 $x=-2^5=-100000$，则$[-100000]_{移}=2^5+x=100000-100000=0,00000$，即最小值的移码为全 0，这符合人们的习惯。利用移码的这一特点，当浮点数的阶码用移码表示时，就能很方便地判断阶码的大小。

3、移码的表数范围

进一步观察发现，同一个真值的移码和补码仅差一个符号位，若将补码的符号位由“0”改为“1”，或从“1”改为“0”，即可得该真值的移码。

整数移码的表数范围和整数补码的表数范围相同，即为$-2^n \leqslant x \leqslant 2^n-1$

三、数的定点表示和浮点表示

（一）定点表示

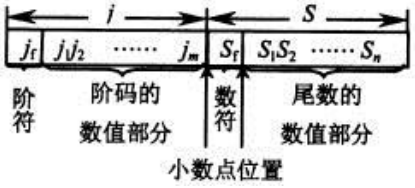
小数点固定在某一位置的数为定点数，有以下两种格式。

当小数点位于数符和第一数值位之间时，机器内的数为纯小数；当小数点位于数值位之后时，机器内的数为纯整数。采用定点数的机器叫做**定点机**。数值部分的位数 n 决定了定点机中数的表示范围。若机器数采用原码，小数定点机中数的表示范围是$-(1-2^{-n})\sim(1-2^{-n})$，整数定点机中数的表示范围是$-(2^n-1)\sim(2^n-1)$。

（二）浮点表示

1、浮点数的表示形式

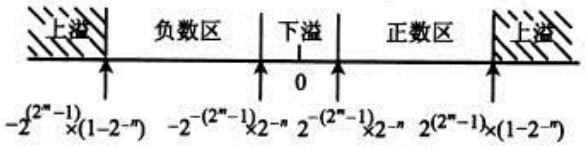
浮点数在机器中的形式如下所示。采用这种数据格式的机器叫做**浮点机**。



可见浮点数由阶码 j 和尾数 S 两部分组成。阶码是整数，阶符和阶码的位数 m 合起来反映浮点数的表示范围及小数点的实际位置；尾数是小数，其位数 n 反映了浮点数的精度；尾数的符号 S_f 代表浮点数的正负。

2、浮点数的表示范围

以通式 $N= S\times r^j$ 为例，设浮点数阶码的数值位取 m 位，尾数的数值位取 n 位，当浮点数为非规格化数时，它在数轴上的表示范围如下所示。



由图可见，其**最大正数**为 $2^{(2^m-1)}\times(1-2^{-n})$；**最小正数**为 $2^{-(2^m-1)}\times2^{-n}$；**最大负数**为 $-2^{-(2^m-1)}\times2^{-n}$；**最小负数**为 $-2^{(2^m-1)}\times(1-2^{-n})$。当浮点数阶码大于最大阶码时，称为“**上溢**”，此时机器停止运算，进行中断溢出处理；当浮点数阶码小于最小阶码时，称为“**下溢**”，此时“溢出”的数绝对值很小，通常将尾数各位强置为零，按机器零处理，此时机器可以继续运行。

3、浮点数的规格化

为了提高浮点数的精度，其尾数必须为规格化数。如果不是规格化数，就要通过修改阶码并同时左右移尾数的办法，使其变成规格化数。将非规格化数转换成规格化数的过程叫做规格化。

当基数为 2 时，尾数最高位为 1 的数为规格化数。规格化时，尾数左移一位，阶码减 1，（这种规格化叫做向左规格化，简称左规）；尾数右移一位，阶码加 1（这种规格化叫做向右规格化，简称右规）。浮点数规格化后，其最大正数为 $2^{(2^m-1)}\times(1-2^{-n})$；最小正数为 $2^{-(2^m-1)}\times2^{-1}$；最大负数为 $-2^{-(2^m-1)}\times2^{-1}$；最小负数为 $-2^{(2^m-1)}\times(1-2^{-n})$。

当基数为 4 时，尾数的最高两位不全为零的数为规格化数。规格化时，尾数左移两位，阶码减 1；尾数右移两位，阶码加 1。

当基数为 8 时，尾数的最高三位不全为零的数为规格化数。规格化时，尾数左移三位，阶码减 1；尾数右移三位，阶码加 1。

（三）定点数和浮点数的比较

定点数和浮点数可从如下几个方面进行比较：

（1）当浮点机和定点机中的数其位数相同时，浮点数的表示范围比定点数大得多。

（2）当浮点数为规格化数财，其精度远比定点数高。

（3）浮点数运算要分阶码部分和尾数部分，而且运算结果都要求规格化，故浮点运算步骤比定点运算步骤多，运算速度比定点低，运算线路比定点复杂。

（4）在溢出的判断方法上，浮点数是对规格化数的阶码进行判断，而定点数是对数值本身进行判断。如小数定点机中的数其绝对值必须小于 1，否则即“溢出”，此时要求机器停止运算，进行处理。为了防止溢出，上机前必须选择比例因子，这个工作比较麻烦，给编程带来不便。而浮点数的表示范围远比定点数大，仅当“上溢”时机器才停止运算，故一般不必考虑比例因子的选择。

（四）IEEE754 标准

在 IEE754 浮点数标准中，定义的浮点数的格式如下表所示。其中浮点数编码有 32 位、64 位和 80 位三种格式，分别称为短实数、长实数和临时实数，短实数和长实数又分别称为单精度数和双精度数。

浮点数	符号位	阶码	尾数	总位数
短实数	1	8	23	32
长实数	1	11	52	64
临时实数	1	25	64	80

在 IEE754 浮点数标准中，符号位 s 仍然用 0 表示正数，用 1 表示负数。正常数的阶码取值 e 的范围为 1～254，尾数部分可以取任意的二进制数值 f。这样，单精度数所表示的数值为：

$$(-1)^s \times 1.f \times 2^{e-127}$$

四、定点数运算

（一）移位运算

1、移位的意义

移位运算又叫移位操作。

计算机中机器数的字长往往是固定的，当机器数左移 n 位或右移 n 位时，必然会使其 n 位低位或 n 位高位出现空位。那么，对空出的空位应该添补 0 还是 1 呢？这与机器数采用有符号数还是无符号数有关，对有符号的移位叫算术移位。

2、算术移位规则

对于正数，由于$[x]_{原}=[x]_{补}=[x]_{反}$=真值，故移位后出现的空位均以 0 添之。

对于负数，由于原码、补码和反码的表示形式不同，故当机器数移位时，对其空位的添补规则也不同。下表列出了三种不同码制的机器数（整数或小数均可），分别对应正数或负数，移位后的添补规则。必须注意的是：不论是正数还是负数，移位后其符号位均不变，这是算术移位的重要特点。

不同码制机器数移位后的空位添补规则：

	码　　制	添补代码
正数	原码、补码、反码	0
	原码	0
负数	补码	左移添 0
		右移添 1
	反　　码	1

由上表可得出如下结论：

（1）机器数为正时，不论左移或右移，添补代码均为 0。

（2）由于负数的原码其数值部分与真值相同，故在移位时只要使符号位不变，其空位均添 0。

（3）由于负数的反码其各位除符号位外与负数的原码正好相反，故移位后所添的代码应与原码相反，即全部添 1。

（4）分析任意负数的补码可发现，当对其由低位向高位找到第一个“1”时，在此“1”左边的各位均与对应的反码相同，而在此“1”右边的各位（包括此“1”在内）均与对应的原码相同，即添 0；右移时困空位出现在高位，则添补的代码应与反码相同，即添 1。

对于负数，三种机器数移位后符号位均不变。负数的原码左移时，高位丢 1，结果出错；低位丢 1，影响精度。负数的补码左移时，高位丢 0，结果出错；低位丢 1，影响精度。负数的反码左移时，高位丢 0，结果出错；低位丢 0，影响精度。

3、算术移位和逻辑移位的区别

有符号数的移位称为**算术移位**，无符号数的移位称为**逻辑移位**。

逻辑移位的规则是：逻辑左移时，高位移出，低位添 0；逻辑右移时，低位移出，高位添 0。

（二）加法与减法运算

1、补码加减运算的基本公式

补码加法的基本公式为：

整数 $[A]_{补}+[B]_{补}=[A+B]_{补} \pmod{2^{n+1}}$

小数 $[A]_{补}+[B]_{补}=[A+B]_{补} \pmod{2}$

即补码表示肋两个数在进行加法运算时，可以把符号位与数位同等处理，只要结果不超出机器能表示的数值范围，运算后的结果按 2^{n+1} 取模(对于整数)；或按 2 取模(对于小数)，就能得到本次加法的运算结果。

对于减法因 A-B=A+(-B)，则[A-B]补=[A+(-B)]补

由补妈加法基本公式可得：

整数 $[A-B]_{补}=[A]_{补}+[-B]_{补} \pmod{2^{n+1}}$

小数 $[A-B]_{补}=[A]_{补}+[-B]_{补} \pmod{2}$

因此，若机器数采用补码， 当求 A-B 时， 只需先求$[-B]_{补}$（称$[-B]_{补}$为“求补”后的减数），就可按补码加法规则进行运算。而$[-B]_{补}$由$[B]_{补}$连同符号位在内，每位取反，末位加 1 而得。

2、溢出判断

补码定点加减运算判断溢出有三种方法。

(1) 用一位符号位判断溢出

对于加法，只有在正数加正数和负数加负数两种情况下才可能出现溢出，符号不同的两个数相加是不会出现溢出的。**对于减法**，只有在正数减负数或负数减正数两种情况下才可能出现溢出，符号相同的两个数相减是不会出现溢出的。因此在判断溢出时可以根据参加运算的两个数据和结果的符号位进行；两个符号位相同的补码相加，如果和的符号位与加数的符号相反，则表明运算结果溢出；两个符号位相反的补码相减，如果差的符号位与被减数的符号位相反，则表明运算结果溢出。这种方法需要判断操作是加法还是减法，以及运算结果与操作数的符号关系。

判断溢出的逻辑表达式：$V = x_0 y_0 \overline{z_0} + \overline{x_0}\,\overline{y_0} z_0$

这种溢出判断方法不仅需要判断加法运算的结果，而且需要保持原操作数。

（2）利用数据编码的最高位（符号位）和次高位（数值部分的最高位）的进位状况来判断运算结果是否发生了溢出

$$V = c_0\overline{c_1} + \overline{c_0}c_1 = c_1 \oplus c_0$$

两个补码数实现加减运算时，若最高数值位向符号位的进位值与符号位产生的进位输出值不相同，则表明加减运算产生了溢出。

（3）采用双符号位补码进行判断

正常时两个符号位的值相同，在运算结果中当两个符号位不同时则表明发生了溢出。运算结果的符号位为 01 表明两个正数相加，结果大于机器所能表示的最大正数，称为**上溢**；运算结果的符号位为 10 表明两个负数相加，结果小于机器所能表示的最小负数，称为**下溢**。也就是说，两个正数相加，数值位不应向符号位同时产生进位，使得结果数的符号位和操作数的一样，为 00：

00+00+00（进位）=00 （mod 4）

两个负数相加，数值位应向符号位产生进位，使得两个负数的双符号位的运算为 11；

11+11+01（进位）=11 （mod 4）

当运算结果的两个符号位不相同时，表明出现了溢出。判断溢出的逻辑表达式：

$$V = Z_0'\overline{Z_0} + \overline{Z_0'}Z_0 = Z_0 \oplus Z_0'$$，其中 Z′ 为增加的符号位。

当最高数据位有进位而符号位无进位时产生上溢出；当最高数据位无进位而符号位有进位时，表示下溢出。

3、基本的二进制加/减法器

设加法器的输入端为 x_i 和 y_i，进位输入端为 c_i，结果输出端为 z_i，进位输出端为 c_{i+1}，则一位加法器的真值表如下表所示。

输入			输出	
x_0	y_0	c_i	c_{i+1}	z_i
0	0	0	0	0
0	0	1	0	1
1	0	0	0	1
1	0	1	1	0
0	1	0	0	1
0	1	1	1	0
1	1	0	1	0
1	1	1	1	1

第 i 位加减法电路的输入输出关系可表示为

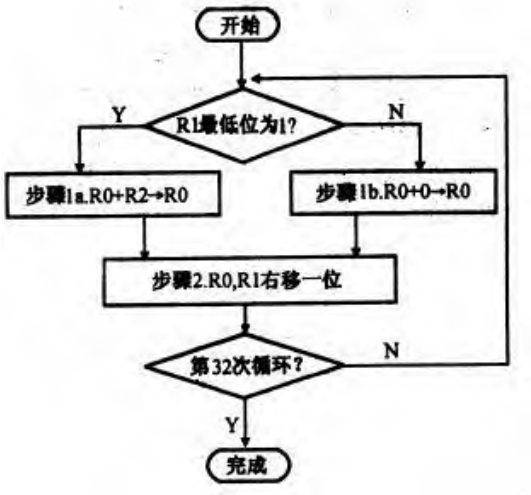
$$z_i = x_i \oplus y_i \oplus c_i$$
$$c_{i+1} = (x_i + c_i)y_i + x_i c_i$$

（三）乘法运算

1、原码一位乘法

完成这个定点原码一位乘法的运算规则可以用如下图所示的逻辑流程图表示。

在该乘法过程中，每次操作是根据乘数的一位进行操作，对于 32 位数的乘法，需要循环 32 次完成一个乘法操作，因此称为一位乘法。



原码一位乘法计算 2*3 的过程如下图所示：

循环	步骤	乘积（R0，R1）
0	初始值	0000 0011
1	1a：加0010	0010 0011
	2：右移1位	0001 0001
2	1a：加0010	0011 0001
	2：右移1位	0001 1000
3	1b：加0	0001 1000
	2：右移1位	0000 1100
4	1b：加0	0000 1100
	2：右移1位	0000 0110

2、原码两位乘法

原码两位乘与原码一位乘一样，符号位的运算和数值部分是分开进行的，但原码两位乘是用两位乘数的状态来决定新的部分积如何形成，因此可提高运算速度。

两位乘数共有 4 种状态，对应这 4 种状态可得下表。

乘数 $y_{n-1}y_n$	新的部分积
00	等于原部分积右移两位
01	等于原部分积加被乘数后右移两位
10	等于原部分积加 2 倍被乘数后右移两位
11	等于原部分积加 3 倍被乘数后右移两位

2 倍被乘数可通过将被乘数左移一位实现，而 3 倍被乘数的获得可以分两步来完成，利用 3=4-1，第一步先完成减 1 倍被乘数的操作，第二步完成加 4 倍被乘数的操作。而加 4 倍被乘数的操作实际上是由比“11”高的两位乘数代替完成的，可以看作是在高两位乘数上加“1”。这个“1”可暂时存在 C_j 触发器中。机器完成置“1” C_j 即意味着对高两位乘数加 1，也即要求高两位乘数代替本两位乘数“11”来完成加 4 倍被乘数的操作。

原码两位乘法规则：

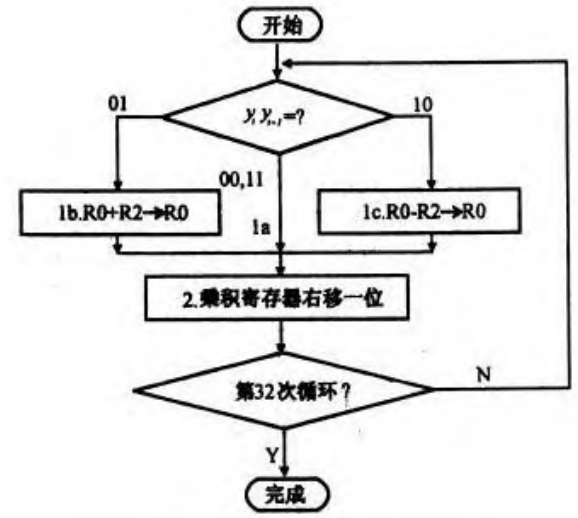
乘数判断位 $y_{n-1}y_n$	标志位 C_j	操 作 内 容
00	0	$z\rightarrow 2, y^*\rightarrow 2, C_j$ 保持“0”
01	0	$z+x^*\rightarrow 2$， $y^*\rightarrow 2, C_j$ 保持“0”
10	0	$z+2x^*\rightarrow 2$， $y^*\rightarrow 2, C_j$ 保持“0”
11	0	$z-x^*\rightarrow 2$， $y^*\rightarrow 2$, 置“1” C_j
00	1	$z+x^*\rightarrow 2$， $y^*\rightarrow 2$, 置“0” C_j
01	1	$z+2x^*\rightarrow 2$， $y^*\rightarrow 2$, 置“0” C_j
10	1	$z-x^*\rightarrow 2$， $y^*\rightarrow 2$， C_j 保持“1”
11	1	$z\rightarrow 2, y^*\rightarrow 2, C_j$ 保持“1”

3、补码一位乘法

Booth 算法：对乘数从低位开始判断，根据两个数据位的情况决定进行加法、减法还是仅仅移位操作。判断的两个数据位为当前位及其右边的位(初始时需要增加一个辅助位 0)，移位操作是向右移动。设 $y=y_0,y_1y_2\ldots y_n$ 为被乘数，x 为乘数，y_i 是 a 中的第 i 位(当前位)。

y_i	y_{i+1}	操作	说明
0	0	无	处于 0 串中，不需要操作

0	1	加 x	1 串的结尾
1	0	减 x	1 串的开始
1	1	无	处于 1 串中，不需要操作



实现32位Booth乘法算法的流程图

4、补码两位乘法

补码两位乘运算规则是根据补码一位乘的规则，把比较 $y_i y_{i+1}$ 的状态应执行的操作和比较 $y_{i-1} y_i$ 的状态应执行的操作合并成一步，便可得出补码两位乘的运算方法。

补码两位乘法运算规则如下

判断位 $y_{i-1} y_i y_{i+1}$	操作内容
000	$[z_{i+1}]_{补}=2^{-2}[z_i]_{补}$
001	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+[x]_{补}\}$
010	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+[x]_{补}\}$
011	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+2[x]_{补}\}$
100	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+2[-x]_{补}\}$
101	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+[-x]_{补}\}$
110	$[z_{i+1}]_{补}=2^{-2}\{[z_i]_{补}+-x\}_{补}\}$
111	$[z_{i+1}]_{补}=2^{-2}[z_i]_{补}$

（四）除法运算

1、分析笔算除法

手工进行二进制除法的规则是判断除数与被除数的大小，若被除数小，则上商 0，并把被除数的下一位移下来或者补 0,再用余数和右移一位的除数比；若被除数大则上商 1 并做减法。

在除法过程有一个比较被除数（或部分余数）与除数的大小的问题。计算机中数据大小的比较是通过减法实现的，在发现该被除数或部分余数不够减时减法的结果已经形成，这时再用加法恢复原来的部分余数。从而形成了恢复余数的除法运算方法。

2、原码除法

原码除法中由于对余数的处理不同，又可分为恢复余数法和不恢复余数法(加减交替法)两种。

（1）原码恢复余数法

小数定点除法对被除数和除数有一定的约束，即必须满足下列条件：

$$0 < |\text{被除数}| \leq |\text{除数}|$$

恢复余数法的**特点**是：当余数为负时，需加上除数，将其恢复成原来的余数。商值的确定是通过比较被除数和除数的绝对值大小，即 x*-y*实现的， 而计算机内只设加法器， 故需将 x*-y*操作变为$[x^*]_{补}+[-y^*]_{补}$的操作。

在恢复余数法中，每当余数为负时，都需恢复余数，这变延长了机器除法的时间，操作也很不规则，对线路结构不利。加减交替法可克服这些缺点。

（2）加减交替法

加减交替法又称不恢复余数法，可以认为它是恢复余数法的一种改进算法。

分析原码恢复余数法得知：

当余数 Ri>0 时，可上商“1”，再对 Ri 左移一位后减除数，即 $2R_i-y^*$。

当余数 Ri<0 时，可上商“0”，然后再做 R_i+y^*，即完成恢复余数的运算，再做 $2(R_i+y^*)-y^*$，也即 $2R_i+y^*$。

3、补码除法

补码除法也分恢复余数法和加减交替法，后者用得较多，在此只讨论加减交替法。

补码加减交替法运算规则：补码除法其符号位和数值部分是一起参加运算的，因此在算法上不像原码除法那样直观，主要需解决三个问题：第一，如何确定商值；第二，如何形成商符；第三，如何获得新的余数。

① 商值的确定。欲确定商值，必须先比较被除数和除数的大小，然后才能求得商值。

② 商符的形成。在补码除法中，商符是在求商的过程中自动形成的。

③ 新余数$[R_{i+1}]_{补}$的获得。

新余数$[R_{i+1}]_{补}$的获得方法与原码加减交替法极相似，其算法规则为：

当$[R_0]_{补}$与$[y]_{补}$同号时，商上“l”，新余数

$[R_{i+1}]_{补}=2[R_i]_{补}-[y]_{补}=2[R_i]_{补}+[-y]_{补}$

当$[R_0]_{补}$与$[y]_{补}$异号时，商上“0”，新余数

$[R_{i+1}]_{补}=2[R_i]_{补}+[y]_{补}$

五、浮点数四则运算

（一）浮点数加法和减法

浮点数加减运算的步骤：

1、对阶

对阶的目的是使两操作数的小数点位置对齐，即使两数的阶码相等。

首先要求出阶差，再按小阶向大阶看齐的原则，使阶小的尾数向右移位，每右移一位，阶码加 1，直到两数的阶码相等为止。右移的次数正好等于阶差。

尾数右移时可能会发生数码丢失，影响精度。

2、尾数运算

将对阶后的两个尾数按定点加(减)运算规则进行运算。

3、规格化

规格化又分左规和右规两种。

（1）左规

当尾数出现 00.0××…×或 11.1××…×时，需左规。左规时尾数左移一位，阶码减 1，直到符合补码规格化表示式为止。

（2）右规

在尾数加减运算时，一般采用双符号位的补码。尾数运算的结果的两个符号位如果为 01 或是 10，即两个符号位不相等，称为定点加减运算的溢出，这是不允许的，但在浮点运算中，它表明尾数求和结果的绝对值大于 1，可将其右移实现规格化表示。这种规格化的过程称为向右规格化。

其操作是：尾数右移 1 位，阶码加 1。

4、舍入

在对阶和右规的过程中，可能会将尾数的低位丢失，引起误差，影响了精度，为此可用舍入法来提高尾数的精度。常用的舍入方法有三种。

（1）截去法

将多余的位截去，剩下的位不变。其最大误差接近于数据最低位上的 1。

特点：有舍无入，具有误差积累。

（2）“0 舍 1 入”法

“0 舍 1 入”法类似于十进制运算中的“四舍五入”法，即在尾数右移时，被移去的最高数值位为 0，则舍去；被移去的最高数值位为 1，则在尾数的末位加 1。这样做可能使尾数又溢出，此时需再做一次右规。

其最大误差是最低位上的-1/2 到接近于 1/2 之间，正误差可以和负误差抵消。是比较理想的方法，但实现起来比较复杂。

（3）“恒置 1”法

尾数右移时，不论丢掉的最高数值位是“1”或“0”，都使右移后的尾数末位恒置“1”。这种方法同样有使尾数变大和变小的两种可能。

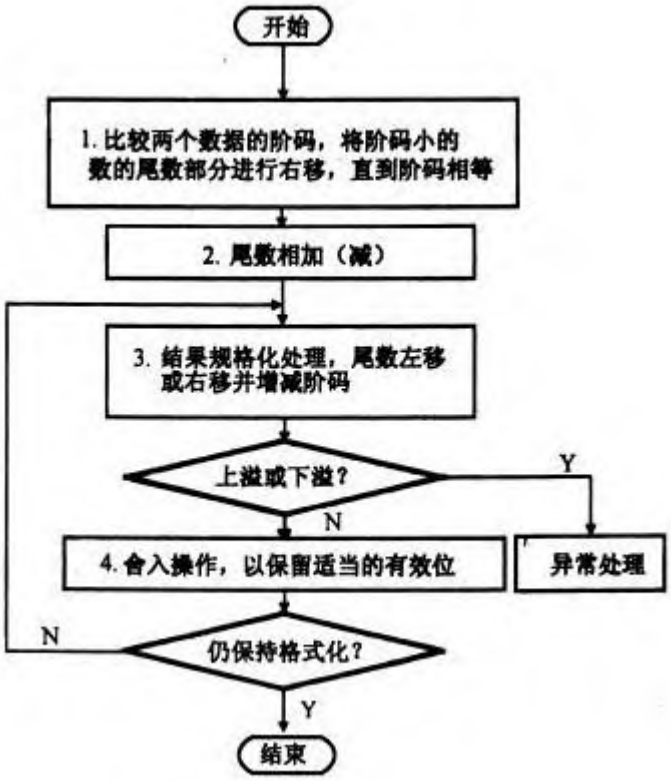
特点：尽管误差范围扩大了，但正负误差可以相互抵消，从统计角度，平均误差为 0。因此最后运算结果的准确性提高了。

5、溢出判断

检查阶码是否溢出。

浮点数的溢出表现为**阶码的溢出**。如果阶码正常，运算正常完成；若阶码下溢，要置结果为浮点形式的机器零；若阶码上溢，则置溢出标志。

6、浮点加减运算流程



（二）浮点乘除法运算

两个浮点数相乘，其乘积的阶码应为相乘两数的阶码之和，其乘积的尾数应为相乘两数的尾数之积。

两个浮点数相除，商的阶码为被除数的阶码减去除数的阶码，其尾数为被除数的尾数除以除数的尾数所得的商。

1、阶码运算

若阶码用补码运算，乘积的阶码为$[j_x]_{补}+[j_y]_{补}$，商的阶码为$[j_x]_{补}-[j_y]_{补}$，两个同号的阶码相加或异号的阶码相减可能产生溢出，此时应作溢出判断。

若阶码用移码运算，则

因为$[j_x]_{移}=2^n+ j_x -2^n \leqslant j_x <2^n$（n 为整数的位数）

$[j_y]_{移}=2^n+ j_y -2^n \leqslant j_y<2^n$（n 为整数的位数）

所以$[j_x]_{移}+[j_y]_{移}= 2^n+ j_x +2^n+ j_y = 2^n+ (2^n +(j_x + j_y))$

$= 2^n+[j_x + j_y]_{移}$

可见，直接用移码求阶码和时，其最高位多加了一个 2^n，要得到移码形式的结果，必须减去 2^n。

求阶码和可用下式完成：

$[j_x]_{移}+[j_y]_{补}= 2^n+ j_x +2^{n+1}+ j_y$

$= 2^n+ [2^n +(j_x + j_y)]$

$= [j_x + j_y]_{移} (\text{mod } 2^{n+1})$

求阶码的差可用下式完成：

[jx-jy]移=[jx]移+[-jy]补

阶码采用移码表示后的溢出判断方法：

如果在原有移码符号位的前面(即高位)再增加位符号位，并规定该位恒用“0”表示，便能方便地进行溢出判断。

溢出的条件是运算结果移码的最高符号位为 1；此时若低位符号位为 0，表示上溢；低位符号位为 1，表示下溢。如果运算结果移码的最高符号位为 0，即表明没溢出。此时若低位符号位为 1，表明结果为正：低位符号位为 0，表示结果为负。

2、尾数运算

（1）浮点乘法尾数运算

可以采用定点小数的任何一种乘法运算来完成。相乘结果可能要进行左规，左规时调整阶码后如果发生阶下溢，则作机器零处理；如果发生阶上溢，则作溢出处理。

此外，尾数相乘会得到一个双倍字长的结果，若限定只取 1 倍字长，则乘积的若干低位将会丢失。如何处理丢失的各位值，通常有两种办法：

其一，无条件的丢掉正常尾数最低位之后的全部数值，这种办法被称为截断处理，其优点是处理简单，但影响精度。

其二，舍入法，对于原码，不论其值是正数还是负数，都采用 0 舍 1 入法。对于正数的补码，也采用 0 舍 1 入法。对于**负数的补码的处理规则**：

① 当丢失的各位均为 0 时，不必舍入；

② 当丢失的各位数中的最高位为 0，且以下各位不全为 0；或者丢失的各位数的最高位为 1，且以下各位均为 0 时，则舍去被丢失的各位；

③ 当丢失的各位数中的最高位为 1，且以下各位不全为 0 时，则在保留尾数的最末位加 1 修正。

（2）浮点除法尾数运算

①检测被除数是否为 0，若为 0，则商为 0；再检测除数是否为 0，若为 0，则商为无穷大，另作处理。若两数均不为 0，则可进行除法运算。

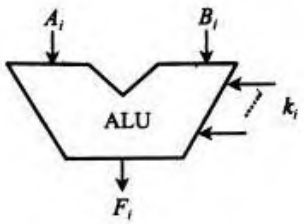
②两浮点数尾数相除同样可采取定点小数的任何—种除法运算来完成。对已规格化的尾数，为了防止除法结果溢出，可先比较被除数和除数的绝对值，如果被除数的绝对值大于除数的绝对值，则先将被除数右移一位，其阶码加 1，再作尾数相除。此时所得结果必然是规格化的定点小数。

（三）浮点运算所需的硬件配置

浮点运算器主要由两个定点运算部件组成，一个是阶码运算部件，用来完成阶码加、减，以及控制对阶码、阶的尾数右移次数和规格化时对阶码的调整；另一个是尾数运算部件，用来完成尾数的四则运算以及判断尾数是否已规格化，此外，还需有判断运算结果是否溢出的电路等。

六、算术逻辑单元

（一）ALU 电路

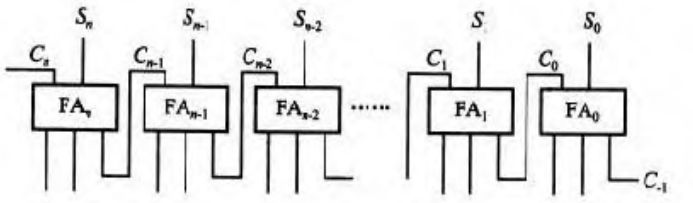


图中 A_i 和 B_i 为输入变量；K_i 为控制信号，K_i 的不同取值可决定该电路作哪一种算术运算或哪一种逻辑运算；F_i 是输出函数。

（二）快速进位链

1、并行加法器

并行加法器由若干个全加器组成，如下图所示。*n*+1 个全加器级联，就组成了一个 *n*+1 位的并行加法器。



由全加器的逻辑表达式可知：

和 $S_i = \overline{A_i}\,\overline{B_i}C_{i-1} + \overline{A_i}B_i\overline{C_{i-1}} + A_i\overline{B_i}\,\overline{C_{i-1}} + A_iB_iC_{i-1}$

$$C_i = \overline{A_i}B_iC_{i-1} + A_i\overline{B_i}\,\overline{C_{i-1}} + A_iB_i\overline{C_{i-1}} + A_iB_iC_{i-1}$$

进位 $= A_iB_i + (A_i + B_i)C_{i-1}$

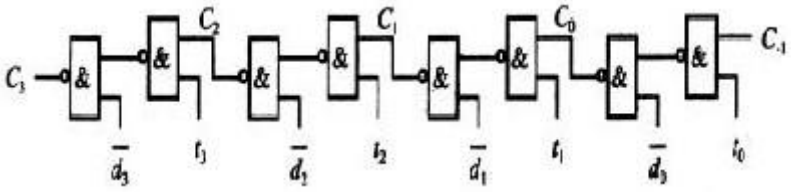
2、串行进位链

串行进位链是指并行加法器中的进位信号采用串行传递。

以四位并行加法器为例，每一位的进位表达式可示为：

$$C_0 = d_0 + t_0C_{-1}$$
$$C_1 = d_1 + t_1C_0$$
$$C_2 = d_2 + t_2C_1$$
$$C_3 = d_3 + t_3C_2$$

由上式可见，采用与非逻辑电路可方便地实现进位传递。



3、并行进位链

并行进位链是指并行加法器中的进位信号是同时产生的，又称先行进位、跳跃进位等。理想的并行进位链是 n 位全加器的 n 位进位同时产生，但实际实现有困难；通常并行进位链有单重分组和双重分组两种实现方案。

(1)单重分组跳跃进位

单重分组跳跃进位就是将 M 位全加器分成若干小组，小组内的进位同时产生，小组与小组之间采用串行进位，这种进位又有组内并行、组间串行之称。

以四位并行加法器为例，对其进位表示式稍作变换，便可获得并行进位表达式：

$$C_0 = d_0 + t_0 C_{-1}$$
$$C_1 = d_1 + t_1 C_0 = d_1 + t_1 d_0 + t_1 t_0 C_{-1}$$
$$C_2 = d_2 + t_2 C_1 = d_2 + t_2 d_1 + t_2 t_1 d_0 + t_2 t_1 t_0 C_{-1}$$
$$C_3 = d_3 + t_3 C_2 = d_3 + t_3 d_2 + t_3 t_2 d_1 + t_3 t_2 t_1 d_0 + t_3 t_2 t_1 t_0 C_{-1}$$

（2）双重分组跳跃进位

将 n 位全加器分成几个大组，每个大组又包含几个小组，而每个大组内所包含的各个小组的最高位进位是同时形成的，大组与大组间采用串行进位。因各小组最高位进位是同时形成的，小组内的其他进位也是同时形成的（注意两小组内的其他进位与小组的最高位进位并不是同时产生的），**故又有组(小组)内并行、组(小组)间并行之称**。

第六章 计算机的运算方法 关键词汇

1、数字化编码

计算机内部处理的所有数据都是“数字化编码”的二进制数据。计算机的输入设备（或接口芯片）实现将现实世界中的媒体信息（模拟信号），如声音、文字、图画、活动图像等转化为二进制数据（数字信号）。在计算机中进行处理、存储和传输的信息采用二进制进行编码的原因有以下几点：

（1）二进制只有两种基本状态，使用有两个稳定状态的物理器件（如三极管）就可以表示二进制数的每一位，而制造有两个稳定状态的物理器件要比制造有多个稳定状态的物理器件容易得多。例如用高、低两个电位，或用脉冲的有无，或脉冲的正、负极性等都可以方便、可靠地表示“0”和“1”；

（2）二进制的编码、计数和运算规则都很简单。可用开关电路实现，简便易行；

（3）两个符号“1”和“0”正好与逻辑命题的两个值“真”和“假”相对应，为计算机中实现逻辑运算和程序中的逻辑判断提供了便利的条件。

2、定点数

计算机处理的数据不仅有符号，而且大量的数据带有小数，小数点不占有二进制一位而是隐含在机器数里某个固定位置上。通常采取两种简单的约定：一种是约定所有机器数的小数的小数点位置隐含在机器数的最低位之后，叫定点整数。

3、浮点数

当要处理的数是既有整数又有小数的混合小数时，采用定点数格式很不方便。为此，人们一般都采用浮点数进行运算。浮点数与科学计数法相似，把一个二进制数通过移动小数点位置表示成阶码和尾数两部分。

4、原码、反码、补码

对于无符号数，原码是一种用数值本身表示的二进制编码。

对于无符号数，反码是一种用对数值按位取反表示的二进制编码。对于有符号数，反码是一种用符号位和对数值按位取反表示的二进制编码。

对于无符号数，补码是一种用对数值按位取反并加 1 表示的二进制编码。对于有符号数，补码是一种用符号和对数值按位取反并加 1 表示的二进制编码。

5、原码的定义和表示方法

（1）原码是机器数中最简单的一种表示形式，其符号位为 0 表示正数，符号位为 1 表示负数，数值位即真值的绝对值，故原码表示又称作带符号的绝对值表示。

（2）整数原码的定义为 $[x]_{原} = \begin{cases} 0, x \\ 2^n - x \end{cases}$

小数原码的定义为 $[x]_{原}=\begin{cases}x & 1>x\geq 0\\ 1-x & 0\geq x>-1\end{cases}$

“0”的原码表示法　　当 x=0 时　　$[+0.0000]_{原}=0.0000$

$[-0.0000]_{原}=1-(0.0000)=1.0000$

6、补码的定义和表示方法

(1) • 一个负数可用它的正补数来代替，而这个正补数可以用模加上负数本身求得。

• 两个互为补数的数，它们绝对值之和即为模数。• 正数的补数即该正数本身。

补数的概念用到计算机中，使出现了补码这种机器数。

(2)补码的定义。

整数补码的定义为 $[x]_{补}=\begin{cases}0,x & 2^n>x\geq 0\\ 2^{n+1}+x & 0\geq x>-2^n(\mathrm{mod}\,2^{n+1})\end{cases}$

式中 x 为真值，n 为整数的位数。

小数补码的定义为 $[x]_{补}=\begin{cases}x & 1>x\geq 0\\ 2+x & 0\geq x>-1(\mathrm{mod}\,2)\end{cases}$

“0”的补码表示法　　当 x=0 时，　　$[+0.0000]_{补}=0.0000$

$[-0.0000]_{补}=2+(-0.0000)=10.0000-0.0000=0.0000$

7、反码的定义和表示方法

反码通常用来作为由原码求补码或者由补码求原码的中间过渡。反码的定义如下：

（1）整数反码的定义 $[x]_{反}=\begin{cases}0,x & 2^n>x\geq 0\\ (2^{n+1}-1)+x & 0\geq x>-2^n(\mathrm{mod}(2^{n+1}-1))\end{cases}$

小数反码的定义为： $[x]_{反}=\begin{cases}x & 1>x\geq 0\\ (2-2^{-n})+x & 0\geq x>-1(\mathrm{mod}(2-2^{-n}))\end{cases}$

“0”的反码表示法　　当 x=0 时　　$[+0.0000]_{反}=0.0000$

$[-0.0000]_{反}=(10.0000-0.0001)-0.0000=1.1111$

8、移码的定义和表示方法

（1）当真值用补码表示时，由于符号位和数值部分一起编码，与习惯上的表示法不同，因此很难从补码的形式上直接判断其真值的大小。如果我们对每个真值加上一个 2^n(n 为整数的位数)，情况就发生了变化。

（2）$[x]_{移}=2^n+x$　$(2^n>x\geq -2^n)$　　式中 x 为真值，n 为整数的位数。

其实移码就是在真值上加一个常数 2^n。在数轴上移码所表示的范围恰好对应与真值在数轴上的范围向轴的正方向移动 2^n 个单元。

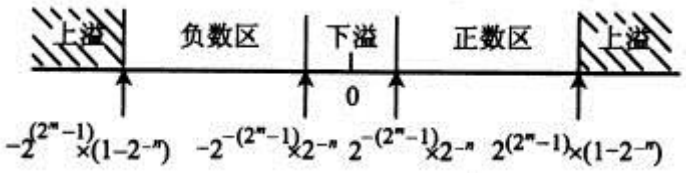
“0”的移码表示。　　当 x=0 时，　　$[+0]_{移}=2^5+0=1,00000$

$[-0]_{移}=2^5-0=1,00000$

9、浮点数的表示形式和范围

（1）浮点数由阶码 j 和尾数 S 两部分组成。阶码是整数，阶符和阶码的位数 m 合起来反映浮点数的表示范围及小数点的实际位置；尾数是小数，其位数 n 反映了浮点数的精度；尾数的符号 S_f 代表浮点数的正负。

（2）浮点数的表示范围　　以通式 $N= S\times r^j$ 为例，设浮点数阶码的数值位取 m 位，尾数的数值位取 n 位，当浮点数为非规格化数时，它在数轴上的表示范围如下所示。



第六章 计算机的运算方法　FAQ

一、比较原码，补码和反码的概念。

• 三种机器数的最高位均为符号位。符号位和数值部分之间可用“.”(对于小数)或“,”(对于整数)隔开。

• 当真值为正时，原码、补码和反码的表示形式均相同，即符号位用“0”表示，数值部分与其值相同。

• 当真值为负时，原码、补码和反码的表示形式不同，但其符号位都用“1”表示，而数值部分有如下关系，即补码是原码的“求反加 1”，反码是原码的“每位求反”。

二、机器数的表示范围如何？

（1）对于定点整数：一个 n+1 位原码能表示的最大正数为 01…11，即 2^n-1；能表示的最小数为绝对值最大的负数 111…1,即$-(2^n-1)$。所以原码能表示的数值范围为：$-(2^n-1) \leqslant x \leqslant 2^n-1$。

对于定点小数：　一个 n+1 位定点小数原码能表示的最大正数为 0.1…11,即 $1-2^{-n}$；能表示的最小数为绝对值最大的负数为 1.11…1，即$-(1-2^{-n})$。定点小数原码的数值范围为：$-(1-2^{-n}) \leqslant x \leqslant 1-2^{-n}$。

（2）一个 n+1 位整数补码能表示的最大数是 011…1，即 2^n-1；能表示的最小数为 100…0，即-2^n。所以它能表示的数值范围是：$-2^n \leqslant x \leqslant 2^n-1$

一个 n+1 位小数补码能表示的最大数是 0.11…1，即 $1-2^{-n}$；能表示的最小数为 1.00… 0，即-1。所以它能表示的数值范围是：$-1 \leqslant x \leqslant 1-2^{-n}$

（3）定点整数反码的数值范围为：$-(2^n-1) \leqslant x \leqslant 2^n-1$。

定点小数原码的数值范围为：$-(1-2^{-n}) \leqslant x \leqslant 1-2^{-n}$。

（4）进一步观察发现，同一个真值的移码和补码仅差一个符号位，若将补码的符号位由“0”改为“1”，或从“1”改为“0”，即可得该真值的移码。 整数移码的表数范围和整数补码的表数范围相同，即为$-2^n \leqslant x \leqslant 2^n-1$

三、定点数和浮点数的比较。

（1）当浮点机和定点机中的数其位数相同时，浮点数的表示范围比定点数大得多。

（2）当浮点数为规格化数财，其精度远比定点数高。

（3）浮点数运算要分阶码部分和尾数部分，而且运算结果都要求规格化，故浮点运算步骤比定点运算步骤多，运算速度比定点低，运算线路比定点复杂。

（4）在溢出的判断方法上，浮点数是对规格化数的阶码进行判断，而定点数是对数值本身进行判断。如小数定点机中的数其绝对值必须小于 1，否则即“溢出”，此时要求机器停止运算，进行处理。为了防止

溢出，上机前必须选择比例因子，这个工作比较麻烦，给编程带来不便。而浮点数的表示范围远比定点数大，仅当“上溢”时机器才停止运算，故一般不必考虑比例因子的选择。

四、不同码制机器数移位后的空位添补规则。

	码　　制	添补代码
正数	原码、补码、反码	0
	原码	0
负数	补码	左移添 0
		右移添 1
	反　　码	1

（1）机器数为正时，不论左移或右移，添补代码均为 0。

（2）由于负数的原码其数值部分与真值相同，故在移位时只要使符号位不变，其空位均添 0。

（3）由于负数的反码其各位除符号位外与负数的原码正好相反，故移位后所添的代码应与原码相反，即全部添 1。

（4）分析任意负数的补码可发现，当对其由低位向高位找到第一个“1”时，在此“1”左边的各位均与对应的反码相同，而在此“1”右边的各位（包括此“1”在内）均与对应的原码相同，即添 0；右移时困空位出现在高位，则添补的代码应与反码相同，即添 1。

五、算术移位和逻辑移位的区别？

有符号数的移位称为算术移位，无符号数的移位称为逻辑移位。逻辑移位的规则是：逻辑左移时，高位移出，低位添 0；逻辑右移时，低位移出，高位添 0。例如，寄存器内容为 01010011，逻辑左移为 1010010，算术左移为 00100110（最高数位“1”移丢）。又如寄存器内容为 10110010，逻辑右移为 01011001。若将其视为补码，算术右移为 11011001。显然，两种移位的结果是不同的。上例中为了避免算术左移时最高数位丢 1，可采用带进位(C_y)的移位，其示意图如下图所示。算术左移时，符号位移至 C_y，最高数位就可避免移出。

六、补码加减法的运算补码加减运算的基本公式和溢出判断。

（1）补码加法的基本公式为：

整数　　$[A]_{补}+[B]_{补}=[A+B]_{补}$　　$(\mathrm{mod}\ 2^{n+1})$

小数　　$[A]_{补}+[B]_{补}=[A+B]_{补}$　　$(\mathrm{mod}\ 2)$

减法因 A-B=A+(-B)　　则$[A-B]_{补}=[A+(-B)]_{补}$

由补妈加法基本公式可得：

整数　　$[A-B]_{补}=[A]_{补}+[-B]_{补}$　　$(\mathrm{mod}\ 2^{n+1})$

小数　　$[A-B]_{补}=[A]_{补}+[-B]_{补}$　　$(\mathrm{mod}\ 2)$

因此，若机器数采用补码，　当求 A-B 时，　只需先求$[-B]_{补}$（称$[-B]_{补}$为“求补”后的减数），就可按补码加法规则进行运算。而$[-B]_{补}$由$[B]_{补}$连同符号位在内，每位取反，末位加 1 而得。

（2）溢出判断

一位符号位判溢出　参加操作的 两个数（减法时即为被减数和“求补”以后的减数）符号相同，其结果的符号与原操作数的符号不同，即为溢出

二位符号位判溢出

$$[x]_{补'}=\begin{cases} x & 1>x\geqslant 0 \\ 4+x & 0>x\geqslant -1\ (\mathrm{mod} \end{cases}$$

[x]补’ + [y]补’ = [x + y]补’ mod 4[x －y]补’ = [x]补’ + [－ y]补’（mod 4）
结果的双符号位 相同 未溢出；结果的双符号位 不同 溢出；
最高符号位 代表其真正的符号

七、原码一位乘的缺点和改进。

缺点 第一、 将多个数一次相加，机器难以实现。一般的加法器，只能把两个输入数相加，多个位积的同时输入是无法实现的。第二、 乘积位数增长了一倍，即 2n，而机器字长只有 n 位。
改进：(a)把一次求和的操作，变成逐步累加求部分积的操作 (b)将求积过程中逐位按权左移位积的操作，改为位积不动，而是上次部分积右移的操作

B.机器算法： 若用 Zi 表示第 i 次部分积，则

Z0 = 0 Z1 = 2-1 (BnA + Z0) Z2 = 2-1 (Bn-1A + Z1)
… Zi = 2-1 (Bn-i+1A + Zi-1)
… Zn = 2-1 (B1A + Zn-1) Zn 即为 A 和 B 的乘积，即 A•B=Zn

八、浮点加减法运算步骤？

（1）对阶
（2） 尾数求和（差）
（3） 规格化
（4）舍入
（5） 浮点数的溢出判断

设两个浮点数 x 和 y 分别为：

x=Sx•2Ex y=Sy•2 Ey

其中，Ex、Ey 分别是 x 和 y 的阶码，Sx 和 Sy 是 x、y 的尾数。
假定它们都是规则化的数，即其尾数绝对值总小于 1(用补码表示，允许为 1)

九、浮点数的对阶运算方法？

原则：小阶向大阶看齐

对阶的第一步是求阶差：△E =Ex-Ey
若△E =0，表示两数阶码相等，即 Ex＝Ey，不需要对阶
若△E >0，表明 Ex>Ey 若△E <0，表明 Ex<Ey
对于 Ex≠Ey 的这种情况，需要对阶。

采用 “小阶向大阶看齐”的方法，即小阶的尾数右移△E 位，小阶的阶码增加△E 与大阶相等。

十、规格化方法？

（1）对于定点小数，其规格化数为：

00.1xx…x
11.0xx…x （原码表示法）

（2）对于负数的补码表示法，规格化定义有所不同：
根据规格化浮点数的定义可知，规格化的尾数应满足：

S>0 时 1/2≤S<1
对于 S<0, 用补码表示时 -1/2>S≥-1

理论上，S 可等于-1/2，但[-1/2]补=11.100…0，为了便于判别是否是规格化数，不把-1/2 列为规格化数，而把-1 列入规格化数。

∵ [-1]补=11.00…0
∴ 补码规格化的浮点数应有两种形式：

00.1xx…x 11.0xx…x

由此可知补码规格化的条件是：
（A）若和或差的尾数两符号位相等且与尾数第一位相等，则需向左规格化。即将和或差的尾数左移，每移

一位，和或差的阶码减一，直至尾数第一位与尾符不等时为止。

（B）若和或差的尾数两符号位不等，即 01.xx…x 或 10.xx…x 形式,表示尾数求和(差)结果绝对值大于 1，向左破坏了规格化。此时应该将和(差)的尾数右移 1 位，阶码加 1，即进行向右规格化。

十一、如何舍入？

（1） “0 舍 1 入”法，即右移时丢掉的最高位为 0，则舍去；是 1，则将尾数的末位加 1(相当于进入)。

（2）“恒置 1”法，即不管移掉的是 0 还是 1，都把尾数的末位置 1。

十二、运算器与控制器的关系

（1） 运算器接收控制器发来的各种运算控制命令：

A、 控制接收数据命令

B、 控制运算操作命令

C、 控制输出传送命令

D、 控制通用寄存器读/写命令等

（2） 把运算过程中的反馈信号送回控制器：状态

A、 溢出否？

B、 结果是否为“0”？

C、 是否非法数？

D、 是正还是负？

（3） 进行地址运算，由地址总线互连

例如：（PC）+1=》PC，寻址方式计算

十三、运算器与存储器的关系？

存储器是计算机中保存程序和数据的功能部件。它的基本功能是：读、写。

（1） 运算器与存储器的联系，也是在控制器的控制下进行的。

（2） 控制器对主存是异步工作控制。异步控制方式也称可变时序控制方式

第六章 计算机的运算方法 拓展资源

—— 计算机原码、补码和反码

数值在计算机中表示形式为机器数,计算机只能识别 0 和 1,使用的是二进制,而在日常生活中人们使用的是十进制,″正如亚里士多德早就指出的那样,今天十进制的广泛采用,只不过我们绝大多数人生来具有 10 个手指头这个解剖学事实的结果.尽管在历史上手指计数(5,10 进制)的实践要比二或三进制计数出现的晚.″(摘自〈〈数学发展史〉〉有空大家可以看看哦~,很有意思的).为了能方便的与二进制转换,就使用了十六进制(2 4)和八进制(23).下面进入正题.

数值有正负之分,计算机就用一个数的最高位存放符号(0 为正,1 为负).这就是机器数的原码了.假设机器能处理的位数为 8.即字长为 1byte,原码能表示数值的范围为

(-127~-0 +0~127)共 256 个.

有了数值的表示方法就可以对数进行算术运算.但是很快就发现用带符号位的原码进行乘除运算时结果正确,而在加减运算的时候就出现了问题,如下: 假设字长为 8bits

(1) 10- (1)10 = (1)10 + (-1)10 = (0)10

(00000001)原 + (10000001)原 = (10000010)原 = (-2) 显然不正确.

因为在两个整数的加法运算中是没有问题的,于是就发现问题出现在带符号位的负数身上,对除符号位外的其余各位逐位取反就产生了反码.反码的取值空间和原码相同且一一对应. 下面是反码的减法运算:

(1)10 - (1) 10= (1) 10+ (-1) 10= (0)10

(00000001) 反+ (11111110)反 = (11111111)反 = (-0) 有问题.

(1)10 - (2)10 = (1)10 + (-2)10 = (-1)10
(00000001) 反+ (11111101)反 = (11111110)反 = (-1) 正确

问题出现在(+0)和(-0)上,在人们的计算概念中零是没有正负之分的.(印度人首先将零作为标记并放入运算之中,包含有零号的印度数学和十进制计数对人类文明的贡献极大).

于是就引入了补码概念. 负数的补码就是对反码加一,而正数不变,正数的原码反码补码是一样的.在补码中用(-128)代替了(-0),所以补码的表示范围为:

(-128~0~127)共 256 个.

注意:(-128)没有相对应的原码和反码, (-128) = (10000000) 补码的加减运算如下:

(1) 10- (1) 10= (1)10 + (-1)10 = (0)10

(00000001)补 + (11111111)补 = (00000000)补 = (0) 正确

(1) 10- (2) 10= (1)10 + (-2)10 = (-1)10

(00000001) 补+ (11111110) 补= (11111111)补 = (-1) 正确

所以补码的设计目的是:

⑴使符号位能与有效值部分一起参加运算,从而简化运算规则.

⑵使减法运算转换为加法运算,进一步简化计算机中运算器的线路设计

所有这些转换都是在计算机的最底层进行的,而在我们使用的汇编、C 等其他高级语言中使用的都是原码。看了上面这些大家应该对原码、反码、补码有了新的认识了吧!

第七章 指令系统 课堂笔记

◆ **主要知识点掌握程度**

重点掌握机器指令、操作数类型和操作类型、常见寻址方式及 RISC 技术。本章内容比较重要,重点是指令的格式及寻址方式,难点是对寻址方式的理解。

◆ **知识点整理**

一、机器指令

(一)指令的一般格式

指令是由操作码和地址码两部分组成的,其基本格式如下图所示。

操作码字段	地址码字段

1、操作码

操作码是用来指明该指令所要完成的操作,如加法、减法、传送、移位、转移等等。通常,其位数反映了机器的操作种类,也即机器允许的指令条数,如操作码占 7 位,则该机器最多包含 27=128 条指令。

操作码的长度可以是固定的,也可以是变化的。

操作码的长度固定时,将操作码集中放在指令字的一个字段内;操作码长度不固定的指令,其操作码分散在指令字的不同字段中。扩展操作码的安排示意。

OP	A_1	A_2	A_3

四位操作码 0000 A_1 A_2 A_3

0001 A_1 A_2 A_3 15 条三地址指令

	：	：	：	：
	1110	A_1	A_2	A_3
八位操作码	1111	0000	A_2	A_3
	1111	0001	A_2	A_3 15 条二地址指令
	：	：	：	：
	1111	1110	A_2	A_3
十二位操作码	1111	1111	0000	A_3
	1111	1111	0001	A_3 15 条一地址指令
	：	：	：	：
	1111	1111	1110	A_3
十六位操作码	1111	1111	1111	0000
	1111	1111	1111	0001 16 条零地址指令
	：	：	：	：
	1111	1111	1111	1111

2、地址码

地址码用来指出该指令的源操作数的地址（一个或两个）、结果的地址以及下一条指令的地址。这里的地址可以是主存的地址，也可以是寄存器的地址，甚至可以是 I/O 设备的地址。

(1) 四地址指令

这种指令的地址字段有 4 个，其格式为：

OP	A_1	A_2	A_3	A_4

其中，OP 为操作码；

A_1为第一操作数地址；

A_2为第二操作数地址；

A_3为结果地址；

A_4为下一条指令的地址。

如果指令字长为 32 位，操作码占 8 位，4 个地址字段各占 6 位，则指令的直接寻址范围为 26=64。如果地址字段均指示主存的地址，刚完成一条四地址指令，共需访问四次存储器（取指令一次，取两个操作数两次，存入结果一次）。

（2）三地址指令

三地址指令中只有三个地址，其格式为：

OP	A_1	A_2	A_3

它可完成(A_1)OP$(A_2)\rightarrow A_3$的操作，这种指令的地址隐含在程序计数器 PC 之中。如果指令字长不变，设 OP 仍为 8 位，则三个地址字段各占 8 位，故三地址指令直接寻址范围可达 $2^8=256$。

若地址字段均为主存地址，则完成一条三地址指令也需访问四次存储器。

（3）二地址指令

二地址指令中只含两个地址字段，其格式为：

OP	A_1	A_2

它可完成(A_1)OP（A_2）→A 1 的操作，即 A_1字段既代表源操作数的地址，又代表存放本次运算结果的地址。在不改变指令字长和操作码的位数前提下，二地址指令可直接寻访的主存地址数为 $2^{12}=4K$。

（4）一地址指令

一地址指令的地址码字段只有一个，其格式为：

OP	A_1

它可完成(ACC)OP(A_i) →ACC 的操作，ACC 既存放参与运算的操作数，又存放运算的中间结果，这样，完成一条一地址指令只需两次访存。在指令字长仍为 32 位、操作码位数仍固定位 8 位时， 一地址指令可直接寻址的范围达 2^{24}，即 16M。

(5)零地址指令

零地址指令在指令字中无地址码，例如进栈(PUSH)、出栈(POP)这类指令，其操作数的地址隐含在堆栈指针 SP 中。地址字段也可用来表示寄存器。地址字段表示寄存器时，也可有三地址、二地址、一地址之分。

（二）指令字长

指令字长取决于操作码的长度、操作数地址的长度和操作数地址的个数。不同机器的指令字长是不相同的。

一台机器的指令系统可以采用位数不相同的指令，即指令字长是可变的，如单字长指令、多字长指令。控制这类指令的电路比较复杂，而且多字长指令要多次访问存储器才能取出一条完整的指令，因此使 CPU 速度下降。

二、操作数类型和操作类型

（一）操作数类型

机器中常见的操作数类型有：地址、数字、字符、逻辑数据等。

1、地址

地址可被认为是一个无符号的整数。

2、数字

计算机中常见的数字有：定点数、浮点数和十进制数。

3、字符

普遍采用 ASCII 码

4、逻辑数据

计算机除了作算术运算外，有时还需作逻辑运算，此时 n 个 0 和 1 的组合不是被看作算术数字，而是看作逻辑数。

（二）数据在存储器中的存放方式

通常计算机中的数据存放在存储器或寄存器中，而寄存器的位数便可反映机器字长。一般机器字长可取字节的 1、2、4、8 倍，这样便于字符处理。在大、中型机器中字长为 32 位和 64 位，在微型机中字长从 4 位、8 位逐渐发展到目前的 16 位和 32 位。

由于不同的机器数据字长不同，每台机器处理的数据字长也不统一，例如奔腾处理器可处理 8(字节)、16(字)、32(双字)、64(四字)； PowerPC 可处理 8(字节)、16(半字)、32(字)、64(双字)。因此，为了便于硬件实现，通常要求多字节的数据在存储器的存放方式能满足“边界对准”的要求，如下图所示。

存储器 地址 （十进制）

<table>
<tr><td colspan="4">字（地址 0）</td><td>0</td></tr>
<tr><td colspan="4">字（地址 4）</td><td>4</td></tr>
<tr><td>字节（地址 11）</td><td>字节（地址 10）</td><td>字节（地址 9）</td><td>字节（地址 8）</td><td>8</td></tr>
<tr><td>字节（地址 15）</td><td>字节（地址 14）</td><td>字节（地址 13）</td><td>字节（地址 12）</td><td>12</td></tr>
<tr><td colspan="2">半字（地址 18）</td><td colspan="2">半字（地址 16）</td><td>16</td></tr>
<tr><td colspan="2">半字（地址 22）</td><td colspan="2">半字（地址 20）</td><td>20</td></tr>
</table>

双字（地址 24）	24
双字	28
双字（地址 32）	32
双字	36

图中所示的存储器其存储字长为 32 位，可按字节、半字、字、双字访问。在对准边界的 32 位字长的计算机中(如上图(a)所示)，半字地址是 2 的整数倍，字地址是 4 的整数倍，双字地址是 8 的整数倍。当所存数据不能满足此要求时，可填充一个至多个空白字节。而**字节的次序有两种**，如下图所示，

字地址

0	3	2	1	0
4	7	6	5	4

（a）

0	0	1	2	8
4	4	5	6	7

(b)

其中(a)表示低字节为低地址，(b)表示高字节为低地址。

在数据不对准边界的计算机中，数据(例如一个字)可能在两个存储单元中，此时需要访问两次存储器，并对高低字节的位置进行调整后，才能取得一字，下图的阴影部分即属于这种情况。

存储器		地址（十进制）
字（地址 2）		半字（地址 0）	0
字节（地址 7）	字节（地址 6）	字（地址 4）	4
半字（地址 10）		半字（地址 8）	8

（三）操作类型

不同的机器操作类型也是不同的，但几乎所有的机器都有以下几类通用的操作。

1、数据传送

数据传送包括寄存器与寄存器、寄存器与存储单元、存储单元与存储单元之间的传送。

2、算术逻辑操作

实现算术运算（加、减、乘、除、增 1、减 1、取负数即求补）和逻辑运算(与、或、非、异或);

3、移位

移位可分为算术移位、逻辑移位和循环移位三种。算术移位和逻辑移位分别可实现对有符号数和无符号数乘以 2n(左移)或整除 2n (右移)的运算。

4、转移

转移指令可改变程序的执行顺序。

转移指令按其转移特征又可分为无条件转移、条件转移、跳转、过程调用与返回、陷阱(Trap)等几种。

（1）无条件转移

无条件转移不受任何条件约束，可直接把程序转移到下一条需执行指令的地址。

（2）条件转移

条件转移是根据当前指令的执行结果，来决定是否需转移。若条件满足，则转移；若条件不满足，则继续按顺序执行。

（3）调用与返回

调用指令包括过程调用、系统调用和子程序调用。它可实现从一个程序转移到另一个程序的操作。调用指令(CALL)一般与返回指令(RETURN)配合使用。CALL 用于从当前的程序位置转至于程序的入口；RETURN 用于子程序执行完后重新返回到原程序的断点。

需注意几点：

- 子程序可在多处被调用；
- 子程序调用可出现在子程序中，即允许子程序嵌套；
- 每个 CALL 指令都对应一条 RETURN 指令。

（4）陷阱(Trap)与陷阱指令

陷阱其实是一种意外事故的中断。一旦出现意外故障，计算机就发出陷阱信号，暂停当前程序的执行，转入故障处理程序进行相应的故障处理。

计算机的陷阱指令一般不提供给用户直接使用，而作为隐指令(即指令系统中不提供的指令)，在出现意外故障时，由 CPU 自动产生并执行。

5、输入输出

对于 I/O 单独编址的计算机而言，通常设有输入输出指令，它完成从外设中的寄存器读入一个数据到 CPU 的寄存器内，或将数据从 CPU 的寄存器输出至某外设的寄存器中。

6、其他

其他包括等待指令、停机指令、空操作指令、开中断指令、关中断指令、置条件码指令等等。

三、寻址方式

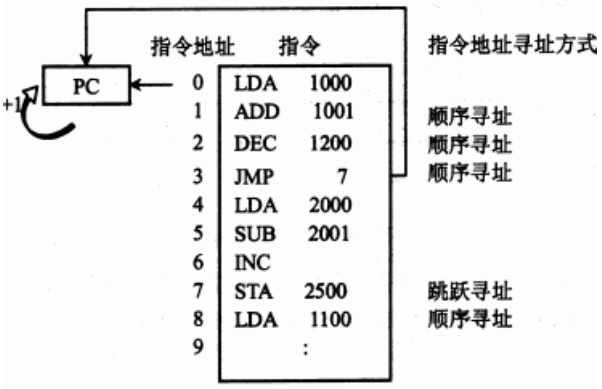
寻址方式是指确定本条指令的数据地址，以及下一条将要执行的指令地址的方法，它与硬件结构紧密相关，而且也直接影响指令格式和指令功能。

寻址方式分为指令寻址和数据寻址两大类。

（一）指令寻址

指令寻址比较简单，它分为顺序寻址和跳跃寻址两种。

顺序寻址可通过程序计数器 PC 加 1，自动形成下一条指令的地址；跳跃寻址则通过转移类指令实现。指令寻址过程如下图所示。



（二）数据寻址

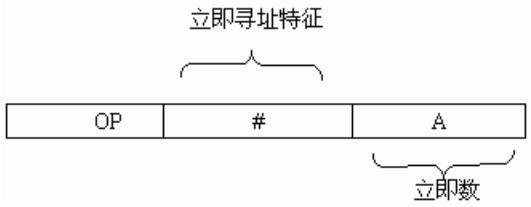
数据寻址方式种类较多，在指令字中必须设一字段来指明属哪一种寻址方式。指令的地址码字段，通常都不代表操作数的真实地址，把它称作**形式地址**，记作 A。 **操作数的真实地址叫做有效地址，记作 EA，**它是由寻址方式和形式地址共同来确定的。由此可得**指令的格式**应如下图所示。

操作码	寻址特征	形式地址 A

为了便于分析研究各类寻址方式，假设指令字长、存储字长、机器字长均相同。

1、立即寻址

立即寻址的特点是操作数本身设在指令字内， 即形式地址 A 不是操作数的地址，而是操作数本身，又称之为立即数。数据是采用补码形式存放的，如下图所示，图中#表示立即寻址特征标记。

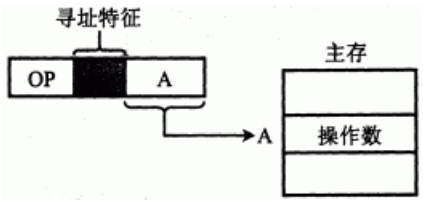


可见，它的优点在于只要取出指令，便可立即获得操作数，这样指令在执行阶段不必再访问存储器。显然，A 的位数限制了这类指令所能表述的立即数的范围。

2、直接寻址

直接寻址的特点是，指令字中的形式地址 A 就是操作数的真实地址 EA，即 EA＝A。

下图所示为直接寻址方式。

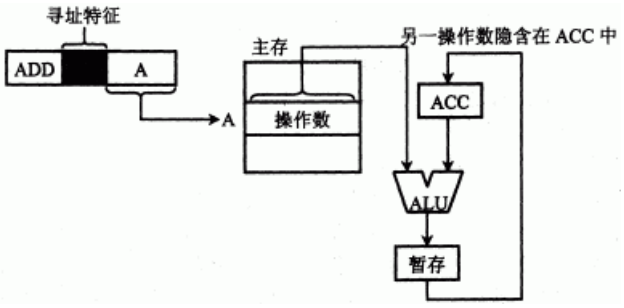


它的**优点**是寻找操作数比较简单；也不需要专门计算操作数的地址，在指令执行阶段对主存只访问一次。 它的缺点在于 A 的位数限制了指令的寻址范围，而且必须修改 A 的值，才能修改操作数的地址。

3、隐含寻址

隐含寻址是指指令字中不明显地给出操作数的地址，其操作数的地址隐含在操作码或某个寄存器中。

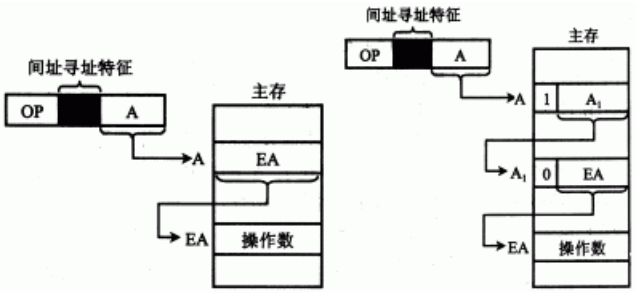
下图所示为隐含寻址方式：



由于隐含寻址在指令字中少了一个地址，由此，这种寻址方式的指令有利于缩短指令字长。

4、间接寻址

倘若指令字中的形式地址不直接指出操作数的地址，而是指出操作数有效地址所在的存储单元地址，也就是说，有效地址是由形式地址间接提供的，故为间接寻址。即 EA＝（A），如下图所示。



图(a)一次间址　　图(b)两次间址

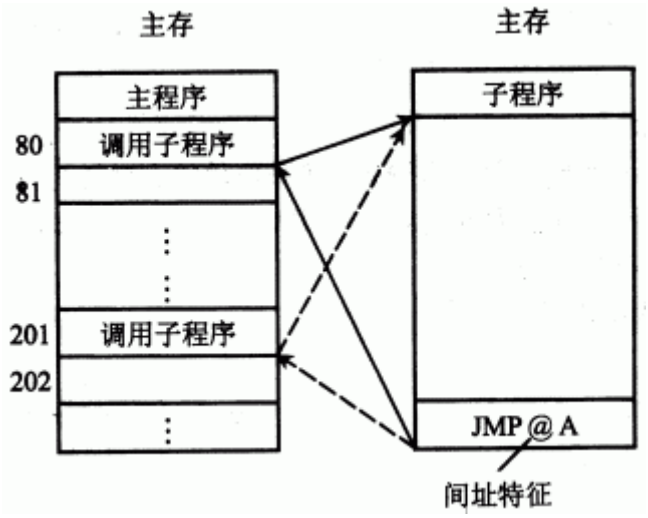
图中(a)为一次间址，即 A 地址单元的内容 EA 是操作数的有效地址；(b)为两次间址，即 A 地址单元的内容 A1 还不是有效地址，而由 A1 所指单元的内容 EA 才是有效地址。

间接寻址的特点：

（1） 扩大了操作数的寻址范围

（2） 便于编制程序

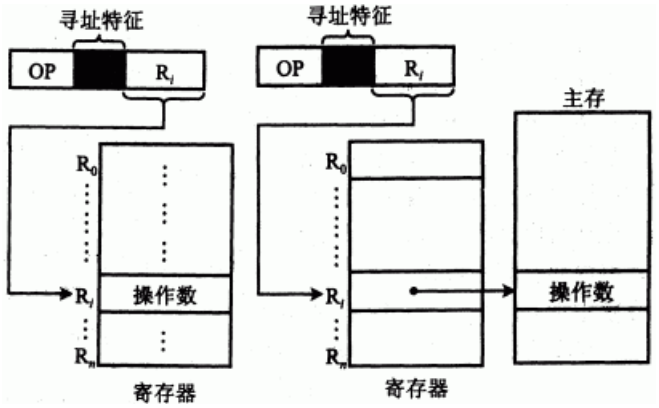
间接寻址的第**二个优点**在于它便于编制程序。例如，用间接寻址可以很方便地完成子程序返回，下图所示为用于子程序返回的间址过程。



图中表示两次调用于程序，只要在调用前先将返回地址存入子程序最末条指令的形式地址 A 的存储单元内， 便可准确返回到原程序断点。如第一次调用前，使[A]＝81， 第二次调用前；使[A]＝202。这样，当第一次子程序执行到最末条指令 JMP @A(@为间址特征位)，便可无条件转至 81 号单元。同理，第二次执行完子程序后，便可返回到 202 号单元。

5、寄存器寻址

在寄存器寻址的指令字中，地址码字段直接指出了寄存器的编号，即 EA＝Ri，如下图所示。其操作数在由 Ri 所指的寄存器内。由于操作数不在主存中，故寄存器寻址在指令执行阶段无须访存，减少了执行时间。 由于地址字段只需指明寄存器编号(计算机中寄存器数有限)，指令字较短，节省了存储空间，因此寄存器寻址在计算机中得到广泛应用。



图(a)寄存器寻址　　图(b)寄存器间接寻址

6、寄存器间接寻址

上图(b)示意了寄存器间接寻址过程。

图中 Ri 中的内容不是操作数，而是操作数所在主存单元的地址号，即有效地址 EA＝（Ri）。与寄存器寻址相比，指令的执行阶段还需访问主存。与图(a)相比， 因有效地址不是存放在存储单元中，而是存放在寄存器电故称其为寄存器间接寻址，它比间接寻址少一次访存。

7、基址寻址

基址寻址需设有基址寄存器 BR，其操作数的有效地址 EA 等于指令字中的形式地址与基址寄存器中的内容(称作基地址)相加。即

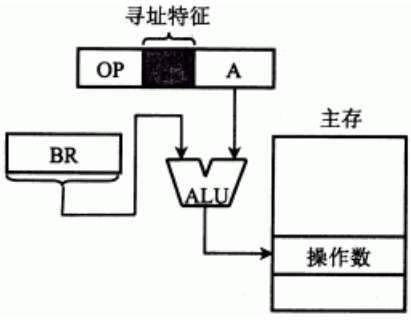
EA＝A+(BR)

基址寻址的特点：

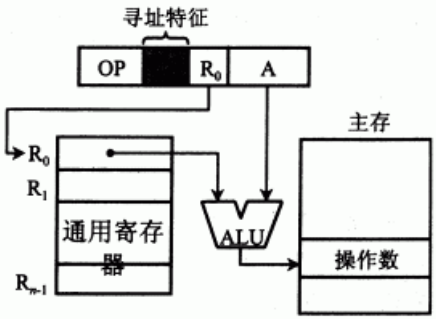
（1） 可以扩大指令对主存的寻址范围

（2） 在多道程序和浮动程序编制时极为有用。

下图示意了基址寻址过程：



图(a)专用基址寄存器 BR



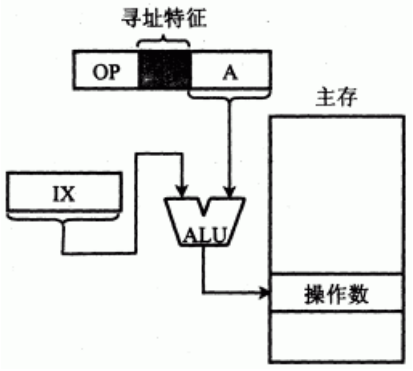
图(b)通用寄存器作基址寄存器

8、变址寻址

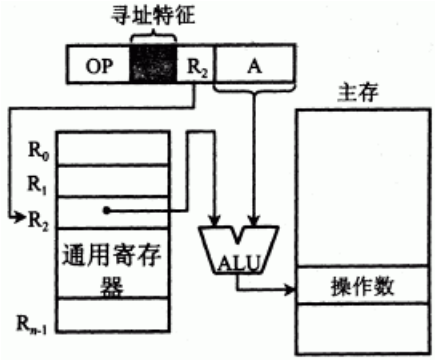
变址寻址与基址寻址极为相似；其有效地址，EA 等于指令字中的形式地址 A 与变址寄存器 IX 的内容相加之和。即

EA＝A+(IX)

显然只要变址寄存器位数足够，也可扩大指令的寻址范围，其寻址过程如下图所示。



图(a)专用变址寄存器

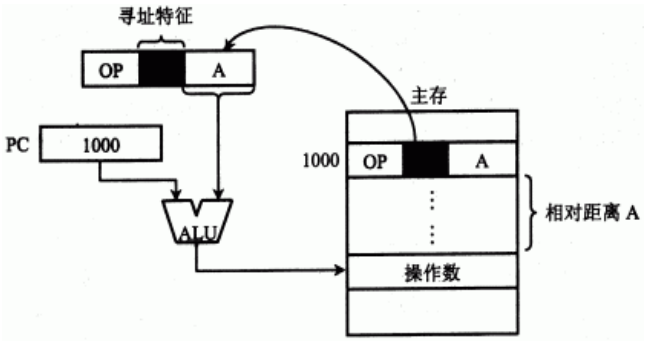


图(b)通用寄存器作变址寄存器

9、相对寻址

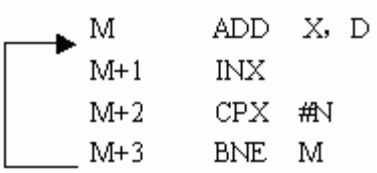
相对寻址的有效地址是将程序计数器 PC 的内容（即当初前指令的地址）与指令字中的形式地址 A 相加而成。即

EA＝(PC)+A



上图示意了相对寻址的过程，由图可见，操作数的位置与当前指令的位置有一段距离 A。

相对寻址的最大特点是转移地址不固定，它可随 PC 值的变化而变，因此，无论程序在主存的哪段区域，都可正确运行，对于编写浮动程序特别有利。例如上表中有一条转移指令 BNE M，它存于 M+3 单元内，也即



显然，随程序首地址改变，M 也改变。如果采用相对寻址，将 BNE M 改写为 BNE *-3(*为相对寻址特征)，就可使该程序浮动至任一地址空间都能正常运行。因为从第 M+3 条指令转至第 M 条指令，其相对位移量为-3，故当执行第 M+3 条指令 BNE *-3 时，其有效地址为

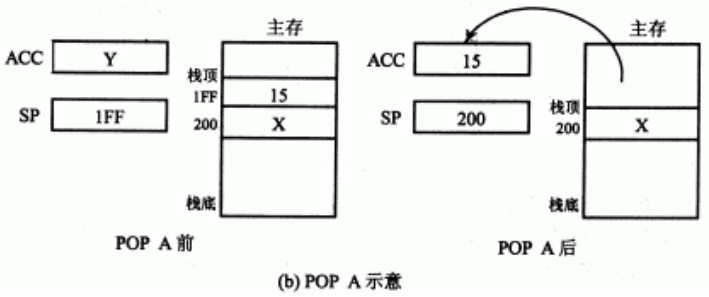
EA=(PC)+(-3)=M+3-3=M

直接指向了转移后的目标地址。

相对寻址也可与间址寻址配合使用。

10、堆栈寻址

堆栈寻址就其本质也可视为寄存器间址，因 SP 可视为寄存器；它存放着操作数的有效地址。下图示意了堆栈寻址过程。



(b) POP A 示意

图中(a)、(b)分别表示进栈 PUSH A 和出栈 POP A 的过程。

由于 SP 始终指示着栈顶地址，因此不论是执行进栈(PUSH)，还是出栈(POP)，SP 的内容都需发生变化。若栈底地址大于栈顶地址，则每次进栈(SP)－Δ→SP；每次出栈(SP)＋Δ→SP。Δ 取值与内存编址方式有关。若按字编址，则 Δ 取 1；若按字节编址，则需根据存储字长是几个字节构成才能确定 Δ，例如字长为 16 位，则 Δ=2，字长为 32 位，Δ=4。

四、指令格式举例

指令格式不仅体现了指令系统的各种功能，而且也突出地反映了机器的硬件结构特点。设计指令格式时必须从诸多方面综合考虑，并经一段模拟运行后，最后确定。

（一）设计指令格式应考虑的各种因素

指令系统集中反映了机器的性能，又是程序员编程的依据。用户在编程时高档机必须能兼容低档机的程序运行，称之为“向上兼容”。

指令格式集中体现了指令系统的功能，为此，在确定指令格式时，必须从以下几个方面综合考虑。

- 操作类型：包括指令数及操作的难易程度；
- 数据类型：确定哪些数据类型可以参与操作；
- 指令格式：包括指令字长、操作码位数、地址码位数、地址个数、寻址方式以及指令字长和操作码位数是否可变等等；
- 寻址方式：寻找操作数真实地址的方式；
- 寄存器个数：寄存器的多少直接影响指令的执行时间。

（二）指令格式举例

不同机器其指令格式可以有很大的差别，下面列举几种较为典型的指令格式。

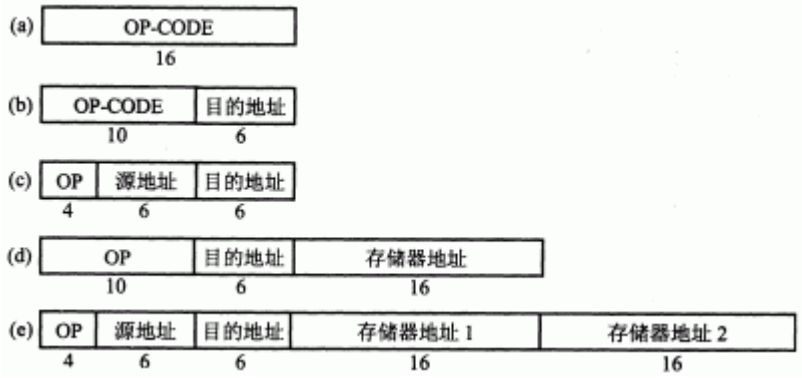
1、PDP-8

PDP-8 的指令字长统一为 12 位，CPU 内只设一个通用寄存器，即累加器 ACC，其主存被划分为若干个容量相等的存储空间(每个相同的空间被称为一页)。该机的指令格式可分为三大类，如下图所示。

2、PDP-11

PDP-11 机器字长为 16 位，CPU 内设 8 个 16 位通用寄存器，其中两个通用寄存器有特殊作用，一个用作堆栈指针 SP，一个用作程序计数器 PC。

PDP-11 指令字长有 16 位、32 位和 48 位三种，采用操作码扩展技术，使操作码位数不固定，指令字的地址格式有零地址、一地址、二地址等共有 13 类指令格式，下图所示的是其中五种。



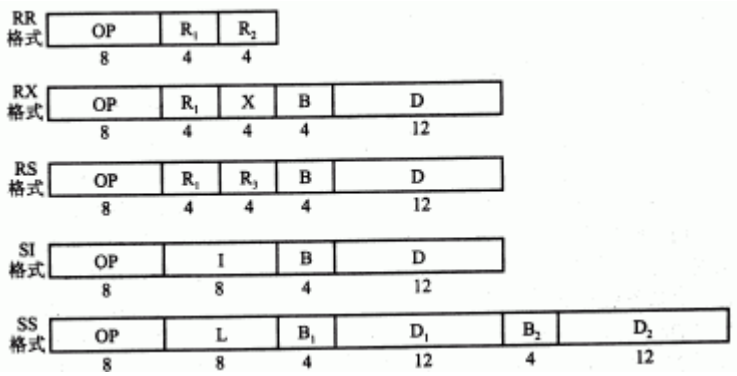
图中(a)为零地址格式；图(b)为一地址格式，其中 6 位目的地址码中的 3 位为寻址特征位，另 3 位表示 8 个寄存器中的任一个；图(c)、图(d)、图(e)均为二地址格式指令，但操作数来源不同，有寄存器—寄存器型、寄存器—存储器型和存储器—存储器型。

PDP-11 指令系统和寻址方式比较复杂，这既增加了硬件的价格，又增加了编程的复杂度，但好处是能编出非常高效的程序。

3、IBM 360

IBM 360 属系列机，所谓系列机是指其基本指令系统相同，基本体系结构相同的一系列计算机。IBM370 对 IBM360 是完全向上兼容的。所以 IBM 370 可看作 IBM360 的扩展或延伸或改进。

IBM360 是 32 位机器，按字节寻址，并可支持多种数据类型，如字节、半字、字、双字(双精度实数)、压缩十进制数、字符串等等。在 CPU 中有 16 个 32 位通用寄存器(用户可选定任一个寄存器作为基址寄存器 BR 或变址寄存器 IX)，4 个双精度(64 位)浮点寄存器。指令字长有 16 位、32 位、48 位三种，如下图所示。



4、Inte18086/80486 系列机

Inte18086/80486 系列微机的指令字长为 1～6 个字节，即不定长。如零地址格式的空操作指令 NOP 只占一个字节；一地址格式的 CALL 指令可以是 3 字节(段内调用)或 5 字节(段间调用)；二地址格式指令中的两个操作数，既可以是寄存器一寄存器型、寄存器—存储器型，也可以是寄存器—立即数型或存储器一立即数型，它们所占的字节数分别为 2、2～4、2～3、3～6 个字节。

五、RISC 技术

RISC 是精简指令系统计算机的英文缩写，即 Reduced Instruction Set Computer，与其对应的是 CISC，即复杂指令系统计算机(Complex Instruction Set Computer)。

（一）RISC 的产生和发展

计算机发展至今，机器的功能越来越强，硬件结构越来越复杂。人们开始进一步分析研究 CISC，发现一个 80－20 规律，即典型程序中 80%的语句仅仅使用处理机中 20%的指令，而且这些指令都是属于简单指令，如取数、加、转移等等。

（二）RISC 的主要特征

计算机执行程序所需的时间 P 可用下式表述：

$$P=I\times C\times T$$

其中 I 是高级语言程序编译后在机器上运行的机器指令数，C 为执行机器指令所需的平均机器周期，T 是每个机器周期的执行时间。

下表列出了第二代 RISC 机与 CISC 机的 I、C、T 统计，其中 I、T 为比值，C 为实际周期数。

RISC/CISC 的 I、C、T 统计比较

	I	C	T
RISC	1.2～1.4	1.3～1.7	<7
CISC	1	4～10	1

由于 RISC 指令比较简单，用这些简单指令编制出的子程序来代替 CISC 机中比较复杂的指令，因此 RISC 中的 I 比 CISC 多 20%～40%。但 RISC 的大多数指令仅用一个机器周期完成，C 的值比 CISC 小得多。而且 RISC 结构简单，完成一个操作所经过的数据通路较短，使 T 值也大大下降。因此总折算结果，RISC 的性能仍优于 CISC 2～5 倍。

1、RISC 的主要特点

通过对 RISC 各种产品的分析，可归纳出 RISC 机应具有如下一些特点：

• 选取使用频率较高的一些简单指令以及一些很有用但又不复杂的指令，让复杂指令的功能由频度高的简单指令的组合来实现。

• 指令长度固定，指令格式种类少，寻址方式种类少。

• 只有取数/存数(LOAD/STORE)指令访问存储器：其余指令的操作都在寄存器内完成。

• 采用流水线技术，大部分指令在一个时钟周期内完成。采用超标量和超流水线技术，可使每条指令的平均执行时间小于一个时钟周期。

• 控制器采用组合逻辑控制，不用微程序控制。

• CPU 中有多个通用寄存器。

• 采用优化的编译程序。

值得注意的是，商品化的 RISC 机通常不会是纯 RISC 机，故上述这些特点不是所有 RISC 机全部具备的。

下面以 RISC Ⅱ为例，着重分析其指令种类和指令格式。

2、RISCⅡ指令系统举例

(1)指令种类

RISC Ⅱ共有 39 条指令，分为 4 类：

• 寄存器—寄存器操作：移位、逻辑、算术(整数)运算等 12 条：

• 取 / 存数指令：取存字节、半字、字等 16 条：

• 控制转移指令：条件转移、调用/返回等 6 条；

• 其他：存取程序状态字 PSW 和程序计数器等 5 条。

(2)指令格式

RISC 机的指令格式比较简单，寻址方式也比较少，如 RISC Ⅱ其指令格式有两种：短立即数格式和长立即数格式。指令字长固定为 32 位，指令字中每个字段都有固定位置，如下图所示。

(a) 第二源操作数在寄存器中的短立即数格式

(b) 第二源操作数为 imm_{13} 的短立即数格式

(c) 长立即数格式

3、RISC 指令系统的扩充

从实用角度出发，商品化的 RISC 机，因用途不同还可扩充一些指令，如：

• 浮点指令，用于科学计算的 RISC 机，为提高机器速度，增设浮点指令；

• 特权指令，为便于操作系统管理机器，为防止用户破坏机器的运行环境，特设置特权指令。

• 读后置数指令，完成读－修改－写，用于寄存器与存储单元交换数据等。

• 一些简单的专用指令。如某些指令用得较多，实现起来又比较复杂，若用子程序来实现，占用较多的时间，则可考虑设置一条指令来缩短子程序执行时间。有些机器用乘法步指令来加快乘法运算的执行速度。

（三）RISC 和 CISC 的比较

与 CISC 机相比，RISC 机的主要优点可归纳如下：

1、充分利用 VLSI 芯片的面积

CISC 机的控制器大多采用微程序控制，其控制存储器在 CPU 芯片内所占的面积为 50%以上，而 RISC 机控制器采用组合逻辑控制，其硬布线逻辑只占 CPU 芯片面积的 10%左右。可见它可将空出的面积供其他功能部件用。

2、提高计算机运算速度

RISC 机能提高运算速度，主要反映在以下 5 个方面。

• RISC 机的指令数、寻址方式和指令格式种类较少，而且指令的编码很有规律，因此 RISC 的指令译码比 CISC 快。

• RISC 机内通用寄存器多，减少了访存次数，可加快运行速度。

• RISC 机采用寄存器窗口重叠技术，程序嵌套时不必将寄存器内容保存到存储器中，故又提高了执行速度。

• RISC 机采用组合逻辑控制，比采用微程序控制的 CISC 机的延迟小，缩短了 CPU 的周期。

• RISC 机选用精简指令系统，适合于流水线工作，大多数指令在一个时钟周期内完成。

3、便于设计，可降低成本，提高可靠性

RISC 机指令系统简单，故机器设计周期短。

RISC 机逻辑简单，设计出错可能性小，有错时也容易发现，可靠性高。

4、有效支持高级语言程序

RISC 机靠优化编译来更有效地支持高级语言程序。由于 RISC 指令少，寻址方式少，使编译程序容易选择更有效的指令和寻址方式。而且由于 RISC 机的通用寄存器多，可尽量安排寄存器的操作，使编译程序的代码优化效率提高。

第七章 指令系统 关键词汇

1、操作码

操作码是用来指明该指令所要完成的操作，如加法、减法、传送、移位、转移等等。通常，其位数反映了机器的操作种类，也即机器允许的指令条数，如操作码占 7 位，则该机器最多包含 27＝128 条指令。

2、地址码

地址码用来指出该指令的源操作数的地址（一个或两个）、结果的地址以及下一条指令的地址。这里的地址可以是主存的地址，也可以是寄存器的地址，甚至可以是 I/O 设备的地址。

3、数据传送

包括寄存器与寄存器、寄存器与存储单元、存储单元与存储单元之间的传送。如从源到目的之间的传送、对存储器读(LOAD)和写(STORE)、交换源和目的的内容、置 1、清 0、进栈、出栈等。

4、算术逻辑操作

这类操作可实现算术运算（加、减、乘、除、增 1、减 1、取负数即求补）和逻辑运算(与、或、非、异或)；对于低档机而言，一般算术运算只支持最基本的二进制加减、比较、求补等，高档机还能支持浮点运算和十进制运算。

5、移位

移位可分为算术移位、逻辑移位和循环移位三种。算术移位和逻辑移位分别可实现对有符号数和无符号数乘以 2n(左移)或整除 2n (右移)的运算。并且移位操作所需时向远比乘除操作执行时间短，因此，移位操作经常被用来代替简单的乘法和除法运算。

6、转移

在多数情况下，计算机是按顺序执行程序的每条指令的，但有时需要改变这种顺序，此刻可采用转移类指令来完成。转移指令按其转移特征又可分为无条件转移、条件转移、跳转、过程调用与返回、陷阱等几种。

7、立即寻址

立即寻址的特点是操作数本身设在指令字内， 即形式地址 A 不是操作数的地址，而是操作数本身，又称之为立即数。

8、直接寻址

直接寻址的特点是，指令字中的形式地址 A 就是操作数的真实地址 EA，即

EA＝A

9、隐含寻址

隐含寻址是指指令字中不明显地给出操作数的地址，其操作数的地址隐含在操作码或某个寄存器中。

10、间接寻址

倘若指令字中的形式地址不直接指出操作数的地址，而是指出操作数有效地址所在的存储单元地址，也就是说，有效地址是由形式地址间接提供的，故为间接寻址。

11、寄存器寻址

在寄存器寻址的指令字中，地址码字段直接指出了寄存器的编号，即 EA＝Ri，如下图所示。其操作数在由 Ri 所指的寄存器内。

12、基址寻址

基址寻址需设有基址寄存器 BR，其操作数的有效地址 EA 等于指令字中的形式地址与基址寄存器中的内容(称作基地址)相加。

13、变址寻址

变址寻址与基址寻址极为相似；其有效地址，EA 等于指令字中的形式地址 A 与变址寄存器 IX 的内容相加之和。

14、相对寻址

相对寻址的有效地址是将程序计数器 PC 的内容（即当初前指令的地址）与指令字中的形式地址 A 相加而成。

第七章 指令系统 FAQ

一、说明程序与微程序、指令与微指令的异同？

程序和微程序都可以用程序设计的方法进行设计。其区别是前者由机器指令组成，存于存储器；而后者由微指令组成，存于控制存储器，一个微程序对应一条机器指令。

指令和微指令都是计算机的操作命令。但前者由操作码和地址码两部分组成。操作码经译码后与时序、状态条件等组合产生微操作，其地址码部分是用来给出操作数地址的。

微指令中包含一组微命令，经组合后可产生一组微操作，它还包含地址字段，但这个地址是用来确定下一条要执行的微指令的地址。

二、指令和数据均存放在内存中，CPU 如何从时间和空间上区分它们是指令还是数据？

时间上讲，取指令事件发生在取指周期，取数据事件发生在执行周期。

空间上讲，从内存读出的指令流向控制器；从内存中读出的数据流一定流向运算器。

三、微程序控制器的工作原理是什么？

微程序控制器的基本原理是，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，使相应部件执行所规定的操作。

四、举出 CPU 中 6 个主要寄存器并说明其名称及功能？

（1） 指令寄存器（IR）：用来保存当前正在执行的一条指令。

（2） 程序计数器（PC）：用来确定下一条指令的地址。

（3） 地址寄存器（AR）：用来保存当前 CPU 所访问的内存单元的地址。

（4） 缓冲寄存器（DR）：<1>作为 CPU 和内存、外部设备之间信息传送的中转站。
<2>补偿 CPU 和内存、外围设备之间在操作速度上的差别。
<3>在单累加器结构的运算器中，缓冲寄存器还可兼作为操作数寄存器。

（5） 通用寄存器（AC）：当运算器的算术逻辑单元（ALU）执行全部算术和逻辑运算时，为 ALU 提供一个工作区。

（6） 状态条件寄存器：保存由算术指令和逻辑指令运行或测试的结果建立的各种条 件码内容。除此之外，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。

五、一个较完善的指令系统应具备哪些要求？

一个完善的指令系统应满足如下四方面的要求：

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。

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。

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。

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；

匀齐性是指：一种操作性质的指令可以支持各种数据类型；

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。

4） 兼容性：至少要能做到“向上兼容”，即低档机上运行的软件可以在高档机上运行。

六、什么是 RISC？RISC 指令系统的特点是什么？

简称为精简指令系统计算机（简称 RISC）， 起源于 80 年代的 MIPS 主机（即 RISC 机），RISC 机中采用的微处理器统称 RISC 处理器。 RISC 典型范例如：MIPS R3000、HP—PA8000 系列，Motorola M88000 等均属于 RISC 微处理器。

RISC 主要特点： RISC 微处理器不仅精简了指令系统，采用超标量和朝流水线结构；它们的指令数目只有几十条，却大大增强了并行处理能力。

第七章 指令系统 拓展资源

一台计算机所能执行的各种不同类型指令的总和。即一台计算机所能执行的全部操作。不同计算机的指令系统包含的指令种类和数目也不同。一般均包含算术运算型、逻辑运算型、数据传送型、判定和控制型、输入和输出型等指令。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，而且也直接影响到系统软件，影响到机器的适用范围。

计算机各种指令的集合称为指令系统，或指令集。

指令的发展历程：

50 年代：指令系统只有定点加减、逻辑运算、数据传送、转移等十几至几十条指令。

60 年代后期：增加了乘除运算、浮点运算、十进制运算、字符串处理等指令，指令数目多达一二百条，寻址方式也趋多样化。

60 年代后期开始出现系列计算机、复杂指令系统计算机、精简指令系统计算机。

系列计算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。其必要条件是同一系列的各机种有共同的指令集．而且新推出的机种指令系统一定包含所有旧机种的全部指令，即实现一个“向上兼容”。因此旧机种上运行的各种软件可以不加任何修改便可在新机种上运行，大大减少了软件开发费用。

复杂指令系统计算机(CISC)计算机的指令系统多达几百条。但是如此庞大的指令系统难以保证正确性，不易调试维护，造成硬件资源浪费。为此人们又提出了便于 VLSI 技术实现的

精简指令系统计算机（RISC）RISC 是一种计算机系统结构的设计思想，至今还没有一个确切的定义。

指令系统的性能决定了计算机的基本功能，它的设计直接关系到计算机的硬件结构和用户的需要。**一个完善的指令系统应满足如下四方面的要求：**

完备性：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。

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。

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。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；匀齐性是指：一种操作性质的指令可以支持各种数据类型；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。

兼容性：至少要能做到“向上兼容”，即低档机上运行的软件可以在高档机上运行。

概念和定义

指令系统：计算机所能执行的全部指令的集合，它描述了计算机内全部的控制信息和“逻辑判断”能力。

指令：微机完成规定操作的命令，一条指令通常由操作码和地址码组成。

操作数：计算机在运行过程需要的数据称为操作数。

寻址方式：寻找指令中所需要的操作数或操作数地址的方式。寻址方式可分为三大类：关于操作数的寻址

方式、对程序转移地址的寻址方式、关于 I/O 端口的寻址方式，其中关于操作数的寻址又可分为：立即寻址、直接寻址、寄存器寻址、寄存器间接寻址、寄存器相对寻址、基址加贬址寻址、相对基址加变址寻址。

CISC（复杂指定系统计算机）和 RISC（精简指令系统计算机）是当前设计和制造微处理器的两种典型技术。桌面计算机最流行的 x86 使用 RISC，而 ARM、MIPS 等体系结构使用 CISC。指令：引起计算机执行某种操作的最小的功能单。

指令系统：一台计算机中所有机器指令的集合。

CISC：复杂指令系统计算机（Complex Instruction Set Computer)

增强指令功能，设置功能复杂的指令

面向目标代码，面向高级语言、面向操作系统

用一条指令代替一串指令

RISC：简单指令系统计算机（Reduced Instruction Set Computer)

只保留功能简单的指令

功能较复杂的指令用子程序来实现

指令格式：一条指令由操作码和操作数地址码两部分组成。

操作码：指明本条指令的操作功能。如算术运算、逻辑运算、存数、取数、转移等。每条指令分配一个确定的操作码。

操作数地址码：指出该条指令涉及的操作数的地址。

指令字长：一个指令字中包含二进制的位数

机器字长：指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。

机器字长通常与主存单元的位数一致

单字长指令：指令字长=机器字长

半字长指令：指令字长=1/2 机器字长

双字长指令：指令字长=2 倍机器字长

从计算机组成的层次结构分，指令有以下几种：

微指令：微程序级的命令 硬件

机器指令：介于二者之间

宏指令：若干条机器指令组成的软件 软件

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

从组成的角度讲，指令是软件与硬件的接口、交界面。

计算机语言有：

机器语言：0、1 代码，机器可直接识别；

汇编语言：符号化、需汇编程序翻译；

高级语言：B、F、C、P…需翻译（编译或解释）

机器语言是以机器指令的形式书写的语言，其它类型的语言，只有变成机器指令的形式，机器才能直接执行。

高级语言与计算机的硬件结构及指令系统无关，汇编语言依赖于计算机的硬件结构和指令系统。不同的机器有不同的指令，所以用汇编语言编写的程序不能在其他类型的机器上运行。

指令系统：一台机器所包含的全部指令不同的计算机，其用途不同，系统结构不同，采用的硬软件技术不同，其指令系统的功能也不同，有的强大，有的弱小，但其指令不外乎以下几类：

1 算逻运算类

2 数据传送类

3 指令控制类

4 I/O 类

5 其它：停机

第八章 CPU 课堂笔记

◆ **主要知识点掌握程度**

本章从分析 CPU 的功能和内部结构入手， 详细讨论机器完成一条指令的全过程，以及为了进一步提高数据的处理能力，开发系统的并行性所采取的流水技术。此外，本章还进一步概括了中断技术在提高整机系统效能方面的作用。本章要求掌握、识记 CPU 的概念、功能；掌握、识记 CPU 组成；掌握、运用指令周期基本概念；理解典型指令周期；了解时序信号产生器；

◆ **知识点整理**

一、CPU 的功能和结构

（一）CPU 的功能

CPU 是由运算器和控制器组成的， 在这里我们**重点介绍控制器的功能**。

对于冯·诺依曼结构的计算机而言，一旦程序进入存储器后，就可由计算机自动完成取指令和执行指令的任务，控制器就是专用于完成此项任务的，它负责协调并控制计算机各部件执行程序的指令序列，其基本功能是取指令、分析指令和执行指令。

1．取指令

控制器必须具备能自动地从存储器中取出指令的功能。为此，要求控制器能自动形成指令的地址，并能发出取指令的命令，将对应此地址的指令取到控制器中。第一条指令的地址可以人为指定，也可由系统设定。

2．分析指令

分析指令包括两部分内容，其一，分析此指令要完成什么操作，即控制器需发出什么操作命令；其二，分析参与这次操作的操作数地址，即操作数的有效地址。

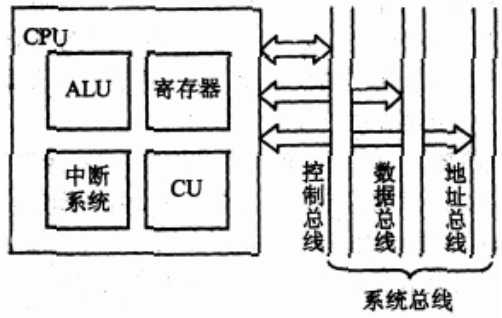
3．执行指令

执行指令就是根据分析指令产生的“操作命令”和“操作数地址”的要求，开成操作控制信号序列（不同的指令有不同的操作控制信号序列），通过对运算器、存储器以及 I/O 设备的操作，执行每条指令。

此外，控制器还必须能控制程序的输入和运算结果的输出（即控制主机与 I/O 交换信息）以及对总线的管理， 甚至能处理机器运行过程中出现的异常情况和特殊请求，即处理中断的能力。

（二）CPU 结构框图

CPU 由四大部分组成：**地址寄存器**， 用于存放当前指令的地址；**指令寄存器**，用来存放当前指令和对指令的操作码进行译码；**控制部件**，能发出各种操作命令序列；**ALU**， 用来完成算术运算和逻辑运算；为了处理异常情况和特殊请求，还必须有中断系统。其结构如下图所示。



（三）CPU 的寄存器

1、用户可见寄存器

通常 CPU 执行机器语言访问的寄存器为用户可见寄存器， 按其特征又可分为以下几类：

(1)通用寄存器

通用寄存器可由程序设计者指定许多功能，**可用于存放操作数，也可作为满足某种寻址方式所需的寄存器**。基址寻址所需的基址寄存器、变址寻址所需的变址寄存器和堆栈寻址所需的栈指针，都可用通用寄存器代替。寄存器间接寻址时还可用通用寄存器存放有效地址的地址。

(2)数据寄存器

数据寄存器用于存放操作数，其位数应满足多数数据类型的数值范围，有些机器允许使用两个连读的寄存器存放双倍字长的值。还有些机器的数据寄存器只能用于保存数据，不能用于操作数地址的计算。

(3)地址寄存器

地址寄存器用于存放地址，其本身可以具有通用性，也可用于特殊的寻址方式，如用于基址寻址的段指针(存放基地址)、用于变址寻址的变址寄存器和用于堆栈寻址的栈指针。地址寄存器的位数必须足够长，以满足最大的地址范围。

(4)条件代码寄存器

这类寄存器中存放条件码，它们对用户来说是部分透明的。条件码是 CPU 根据运算结果由硬件设置的位(bits)， 如算术运算会产生正、负、零或溢出等结果。条件码可被测试，作为分支运算的依据。此外，有些条件码也可被设置，如最高位进位标志 C，可用指令对它置位和复位。将条件码放到一个或多个寄存器中，就构成了条件码寄存器。

2．控制和状态寄存器

CPU 中还有一类寄存器用于控制 CPU 的操作或运算。在一些机器里，大部分这类寄存器对用户是透明的。如以下四种寄存器在指令执行过程中起重要作用。

MAR 存储器地址寄存器，用于存放将被访问的存储单元的地址；

MDR 存储器数据寄存器， 用于存放欲存入存储器中的数据或最近从存储器中读出的数据；

PC 程序计数器，存放现行指令的地址，通常具有计数功能。当遇到转移类指令时，PC 的值可被修改；

IR 指令寄存器，存放当前欲执行的指令。

通过这四个寄存器，CPU 和主存可交换信息。

在 CPU 内部必须给 ALU 提供数据，因此 ALU 必须可直接访问 MDR 和用户可见寄存器，ALU 的外围还可以有另一些寄存器，这些寄存器用于 ALU 的输入/输出以及用于和 MDR 及用户可见寄存器交换数据。

在 CPU 的控制和状态寄存器中，还有用来存放程序状态字 PSW 的寄存器，该寄存器用来存放条件码和其他状态信息。在具有中断系统的机器中还有中断标记寄存器。

（四）控制单元 CU 和中断系统

控制单元 CU 是提供完成机器全部指令操作的微操作命令序列部件。现代计算机中微操作命令序列的形成方法有两种，一种是组合逻辑设计方法，为硬联线逻辑；另一种是微程序设计方法，为存储逻辑。

中断系统主要用于处理计算机的各种中断。

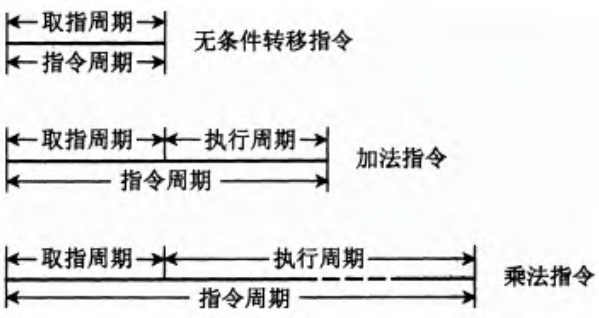
二、指令周期

（一）指令周期的基本概念

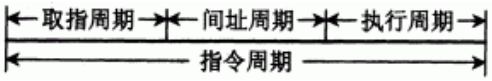
CPU 每取出并执行一条指令所需的全部时间，也即 CPU 完成一条指令的时间叫**指令周期**。如下图所示。

由于各种指令操作功能不同，因此各种指令的指令周期是不相同的。

不同指令周期的比较如下图所示：



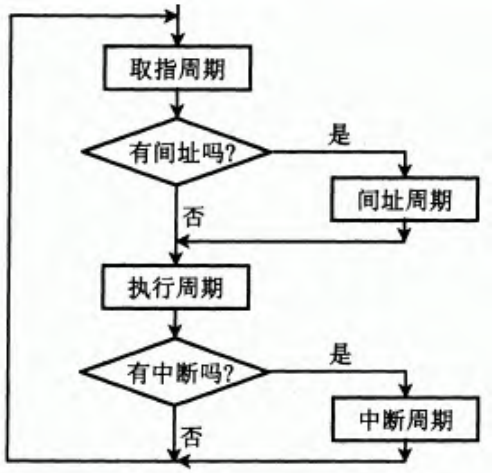
间址寻址的指令周期：



其中间址周期用于取操作数的有效地址，因此间址周期介于取指周期和执行周期之间。

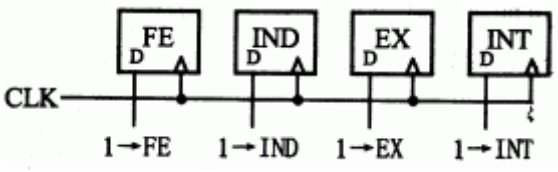
当 CPU 采用中断方式实现主机与 I/O 交换信息时，CPU 在每条指令执行阶段结束前， 都要发中断查询信号，以检测是否有某个 I/O 提出中断请求。如果有请求，CPU 则要进入中断响应阶段，又称**中断周期**。在这阶段， CPU 必须将程序断点保存到存储器中。这样，一个完整的指令周期应包括**取指、间址、执行和中断 4 个子周期**。

指令周期流程如下图所示：



上述 4 个周期都有 CPU 访存操作， 只是访存的目的不同。取指周期是为了取指令，间址周期是为了取有效地址，执行周期是为了取操作数(当指令为访存指令时)，中断周期是为了保存程序断点。这 4 个周期又可叫 CPU 的工作周期，为了区别它们，在 CPU 内可设置 4 个标志触发器。

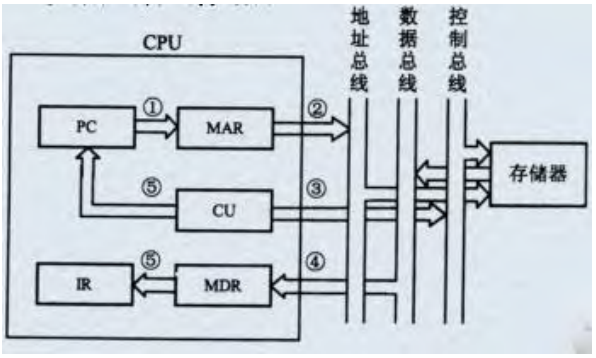
CPU 工作周期的标志：



图中的 FE、IND，EX 和 INT 分别对应取指、间址、执行和中断 4 个周期，并以“1”状态表示有效；它们分别由 1→FE、1→IND、1→EX 和 1→INT4 个信号控制。

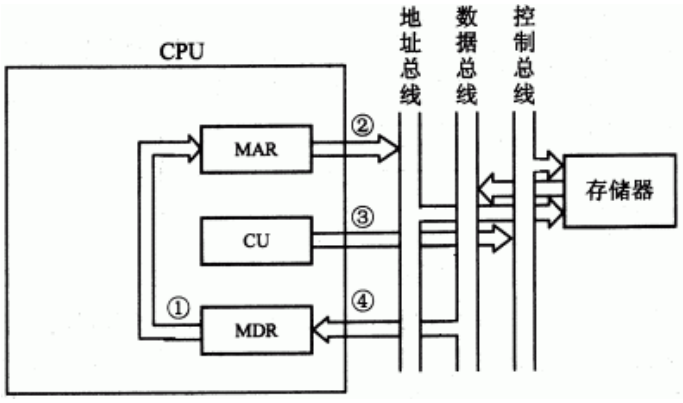
（二）指令周期的数据流

1、取指周期数据流



上图是取指周期的数据流。PC 中存放现行指令的地址，该地址送到 MAR 并送至地址总线， 然后由控制部件 CU 向存储器发出读命令，使对应 MAR 所指单元的内容(指令)经数据总线送至 MDR，再送至 IR，与此同时 CU 控制 PC 内容加 1，形成下一条指令的地址。

2、间址周期数据流



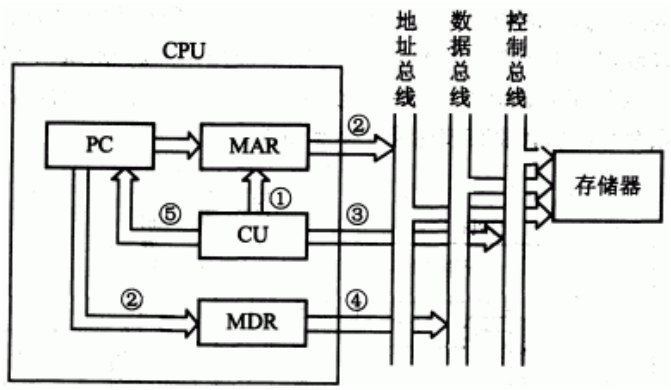
间址周期的数据流如上图所示。一旦取指周期结束，CU 便检查 IR 中的内容，以确定其是否有间址操作，如果需间址操作，则 MDR 中指示形式地址的右 N 位将被送至 MAR，又送至地址总线，此后 CU 向存储器发读命令，并获取有效地址并存至 MDR。

3、执行周期的数据流

由于不同的指令在执行周期的操作不同，因此执行周期的数据流是多种多样的， 可能涉及到 CPU 内部寄存器间的数据传送、或对存储器(或 I/O)进行读写操作、或对 ALU 的操作，因此，无法用统一的数据流图表示。

4、中断周期的数据流

CPU 进入中断周期要完成一系列操作， 其中 PC 当前的内容必须保存起来，以待执行完中断服务程序后可准确返回到该程序的间断处，这一操作的数据流如下图所示。



图中由 CU 把用于保存程序断点的存储器特殊地址（如堆栈指针的内容）送往 MAR，并送到地址总线上，同时将 PC 的内容（程序断点）送到 MDR，并命令存储器写，最终使程序断点经数据总线存入存储器。此外，还需将中断服务程序的入口地址送至 PC，为下一个指令周期的取指周期作好准备。

三、指令流水

为了进一步提高整机的处理能力，通常可从两方面入手：

(1)提高器件的性能

提高器件的性能一直是提高整机性能的重要途径，计算机的发展史就是按器件把计算机分为电子管、晶体管、集成电路和大规模集成电路这四代的。器件的每一次更新换代都使计算机的软硬件技术和计算机性能获得突破性进展。特别是大规模集成电路的发展，由于其集成度高、体积小、功耗低、可靠性高、价格便宜等特点，使人们可采用更复杂的系统结构造出性能更高、工作更可靠、价格更低的计算机。但是由于半导体器件的集成度越来越接近物理极限，机器速度的提高越来越慢。

(2)改进系统的结构，开发系统的并行性

所谓**并行**包含同时性和并发性两个方面。前者是指两个或多个事件在同一时刻发生，后者是指两个或多个事件在同一时间段发生。也就是说，在同一时刻或同一时间段内完成两种或两种以上性质相同或不同的功能，只要在时间上互相重叠，就存在并行性。

（一）指令流水原理

完成一条指令实际上也可分为许多阶段。为简单起见，**把指令的处理过程分为取指令和执行指令两个阶段**，在不采用流水技术的计算机里，一旦取指令和执行指令是周而复始地重复出现，各条指令是按顺序串行执行的，如下图所示。

取指令 1	执行指令 1	取指令 2	执行指令 2	取指令 3	执行指令 3	……

图中取指令的操作可由指令部件完成，执行指令的操作可由执行部件完成。

两条指令的重叠，也即指令的**二级流水**。

由指令部件取出一条指令，并将它暂存起来，如果执行部件空闲，就将暂存的指令传给执行部件执行。与此同时，指令部件又可取出下一条指令并暂存起来，这就叫**指令预取**。

影响指令流水效率加倍的原因有：

（1）指令的执行时间—般大于取指时间，因此，取指阶段可能要等待一段时间，也即存放在指令部件缓冲区的指令还不能立即传给执行部件，缓冲区不能空出。

（2）当遇到条件转移指令时，下一条指令是不可知的，因为必须等到执行阶段结束后，才能获知条件是否成立，从而决定下条指令的地址，造成时间损失。

为了进一步提高处理速度，需将指令的处理过程分解为更细的几个阶段：

- ✓ 取指（FI）：从存储器取出一条指令并暂时存入指令部件的缓冲区。
- ✓ 指令译码（DI）：确定操作性质和操作数地址的形成方式。
- ✓ 计算操作数地址（CO）：计算操作数的有效地址，涉及到寄存器间址、间址、变址、基址、相对寻址等各种地址计算方式。
- ✓ 取操作数（FO）： 从存储器中取操作数(若操作数在寄存器中，则无需此阶段)。
- ✓ 执行指令（EI）：执行指令所需的操作，并将结果存于目的位置(寄存器中)。
- ✓ 写操作数（WO）：将结果存入存储器。

（二）影响流水线性能的因素

1、访存冲突

因为取指令、取操作数和存结果都要访问存储器。

为了避免冲突，可采用如下一些方法：

①设置两个独立的存储器分别存放操作数和指令，以免取指令和取操数同时进行时互相冲突，使取某条指令和取另一条指令的操作数实现时间上的重叠。

②采用指令预取技术，如在 CPU(8086)中设置指令队列， 将指令预先取到指令队列中排队。指令预取技术的实现基于访存周期很短的情况，如在执行指令阶段，取数时间很短，因此在执行指令时，主存会有空闲，此时，只要指令队列空出，就可取下一条指令，并放至空出的指令队列中，从而保证在执行第 K 条指令的同时对第 K+1 条指令进行译码，实现“执行 K”与“分析 K+1”的重叠。

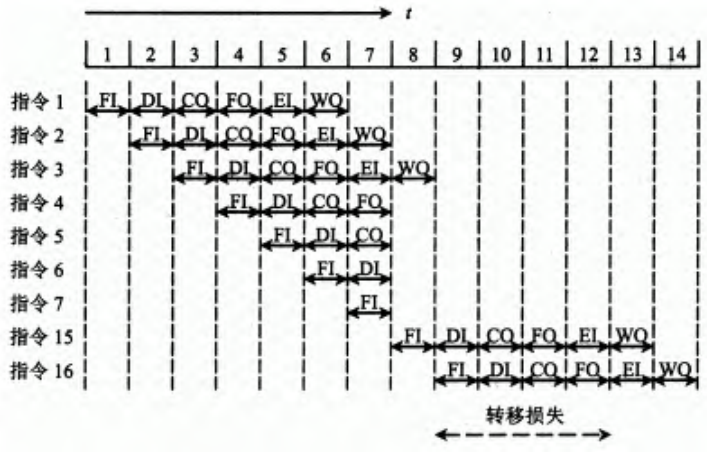
2、相关问题

所谓**相关问题**是指程序的相近指令之间出现某种关联，使指令流水线出现停顿，影响了指令流水线的效率。指令间的相关大体可分控制相关和数据相关两类。

(1) 控制相关

如果一条指令要等前一条(或几条)指令作出转移方向的决定后，才能进入流水线，便发生了**控制相关**，**最典型的情况就是条件转移指令**。

条件转移指令对指令流水操作的影响：



（2）数据相关

数据相关是发生在几条相近的指令间共用同一个存储单元或寄存器时发生的。例如某条指令为了计算操作数的地址需用到某寄存器的内容。但是产生这“内容”的指令还未执行结束，也即还没有产生结果，这时

流水线也只能暂停等待。

为了解决此问题，可采用**旁路技术**，即在执行部件与指令部件之间设置直接传送数据的通路，在执行部件产生结果向寄存器送数的同时，把此数直接送至指令部件，计算操作数的有效地址。

（三）流水线中的多发技术

流水线中的多发技术，即设法在一个时钟周期(机器主频的倒数)内，产生更多条指令的结果。

常见的多发技术有超标量技术、超流水线技术和超长指令字技术。

1、超标量技术

超标量(Super scalar)技术是指在每个时钟周期内可同时并发多条独立指令，即以并行操作方式将两条或两条以上指令编译并执行。

要实现超标量技术，要求处理机中配置多个功能部件和指令译码电路，以及多个寄存器端口和总线，以便能实现同时执行多个操作，此外还要编译程序决定哪几条相邻指令可并行执行。

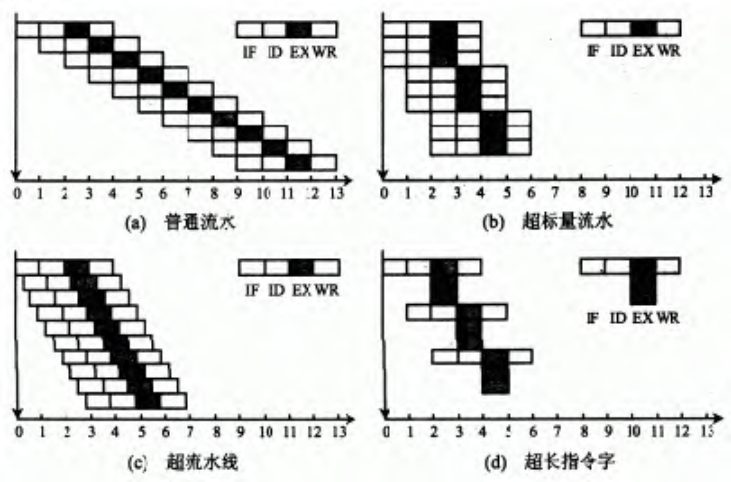
2、超流水线技术

超流水线(Super pipe lining) 技术是将一些流水线寄存器插入到流水线段中，好比将流水线再分道，它将原来的一个时钟周期又分成三段，使超级流水线的处理器周期比一般流水线的处理器周期短，这样，在原来的时钟周期内，功能部件被使用三次，使流水线以 3 倍于原来时钟频率的速度运行。

3、超长指令字(VLIW)技术

超长指令字(VLIW)技术和超标量技术都是**采用多条指令在多个处理部件中并行处理的体系结构**，在一个时钟周期内能流出多条指令。但超标量的指令来自同一标准的指令流，VLIW 则是由编译程序在编译时挖掘出指令间潜在的并行性后，把多条能并行操作的指令组合成一条具有多个操作码字段的超长指令(指令字长可达几百位)，由这条超长指令控制 VLIW 机中多个独立工作的功能部件，由每一个操作码字段控制一个功能部件，相当于同时执行多条指令

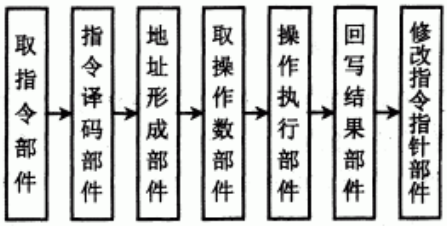
4、四种流水技术的比较



（四）流水线结构

1、指令流水线结构

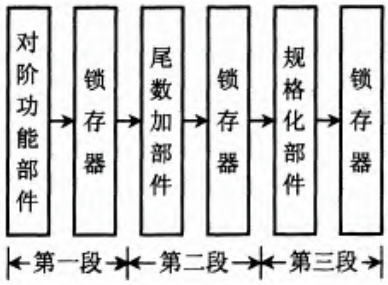
指令流水线是将指令的整个执行过程用流水线进行分段处理，典型的指令执行过程分为取指令－指令译码－形成地址－取操作数－执行指令－回写结果－修改指令指针这几阶段，与此相对应的指令流水线结构由下图所示的几个部件组成。



指令流水线对机器性能的改善程度取决于把处理过程分解成多少个相等的时间段数。

2、运算流水线

将流水技术用于部件级。如浮点加法运算，可以分成“对阶”“尾数加”及“结果规格化”三段，每一段都有一个专门的逻辑电路完成操作，并将其结果保存在锁存器中，作为下一段的输入。如下图所示，当对阶完成后，将结果存入锁存器，便又可进入下一条指令的对阶运算。



四、中断系统

（一）概述

1、引起中断的各种因素

（1）人为设置的中断

这种中断一般叫做自愿中断，因为它是在程序中人为设置的，故一旦机器执行这种人为中断时，便自愿停止现行程序而转入中断处理。

（2）程序性事故

如定点溢出、浮点溢出、操作码不能识别、除法中出现“非法”等，这些都属于由程序设计不周而引起的中断。

（3）硬件故障

硬件故障类型很多，如插件接触不良、通风不良、磁表面损坏、电源掉电等都属硬设备故障。

（4）I/O 设备

I/O 设备被启动以后，一旦其准备就绪，便向 CPU 发出中断请求。每个 I/O 设备都能发中断请求，因此这种中断与计算机所配置的 I/O 设备多少有关。

（5）外部事件

用户通过键盘来中断现行程序属于外部事件中断。

能引起中断的各个因素称作**中断源**，**中断源可分两大类**：一类为不可屏蔽中断，这类中断 CPU 不能禁止响应，如电源掉电；另一类属可屏蔽中断，对可屏蔽中断源的请求，CPU 可根据该中断源是否被屏蔽来确定是否给予响应，若未屏蔽则能响应；若已被屏蔽，则 CPU 不能响应。

2、中断系统需解决的问题

(1) 各中断源如何向 CPU 提出中断请求；

(2) 当多个中断源同时提出中断请求时，中断系统如何确定优先响应哪个中断源的请求；

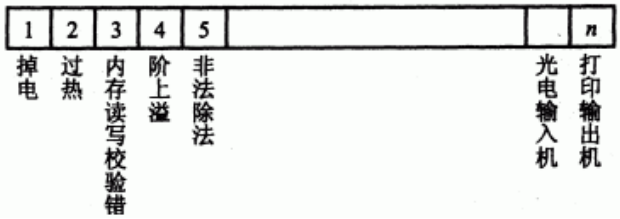
(3) CPU 在什么条件、什么时候、以什么方式来响应中断；

(4) CPU 响应中断后如何保护现场；
(5) CPU 响应中断后，如何停止原程序的执行而转入中断服务程序的入口地址；
(6) 中断处理结束后，CPU 如何恢复现场，如何返回到原程序的间断处；
(7) 在中断处理过程中又出现了新的中断请求，CPU 该如何处理。

（二）中断请求标记和中断判优逻辑

1、中断请求标记

为了判断是哪个中断源提出请求，在中断系统中必须设置中断请求标记触发器，简称中断请求触发器，记作 INTR，当其状态为“1”时，表示中断源有请求。如下图所示。



2、中断判优逻辑

当某一时刻有多个中断源提出中断请求时，中断系统必须按其优先顺序予以响应，这就叫中断判优。各中断源的优先顺序是根据该中断源若得不到及时响应，致使机器工作出错的严重程度而定的。

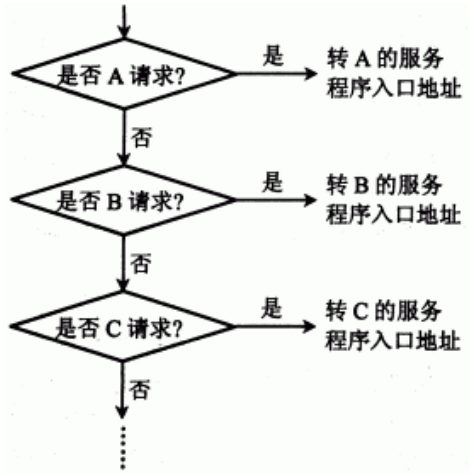
中断判优可用硬件实现，也可用软件实现。

（1）硬件排队

硬件排队又分两种，一种为链式排队器，它对应中断请求触发器分散在各个接口电路中的情况，每一个接口电路中都没有一个非门和一个与非门，它们犹如链条一样串接起来；另一种排队器设在 CPU 内。

（2）软件排队

软件排队是通过编写查询程序实现的，其程序框图如下图所示。程序按中断源的优先等级，从高至低逐级查询各中断源是否有中断请求，这样就可保证 CPU 首先响应级别高的中断源的请求。



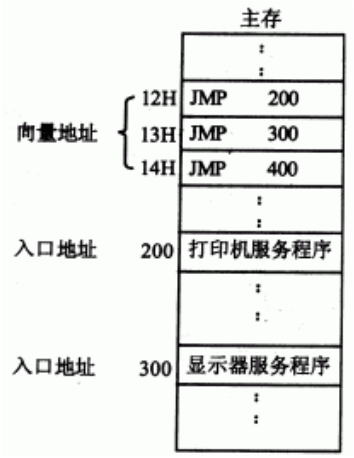
（三）中断服务程序入口地址的寻找

1、硬件向量法

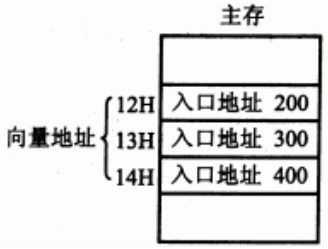
硬件向量法就是利用硬件产生向量地址，再由向量地址找到中断服务程序的入口地址。向量地址由中断向量地址形成部件产生，这个电路可分散设置在各个接口电路中，也可设置在 CPU 内。

由向量地址寻找中断服务程序的入口地址通常采用两种办法。一种如下图所示，在向量地址内存放一

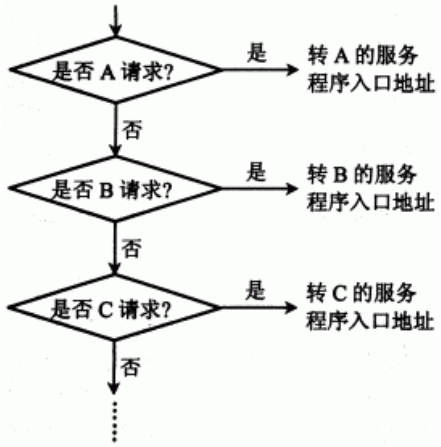
条无条件转移指令，CPU 响应中断时，只要将向量地址(如 12H)送至 PC，执行这条指令，便可无条件转向打印机服务程序的入口地址 200。



另一种是设置向量地址表，如下图所示。该表设在存储器内，存储单元的地址为向量地址，存储单元的内容为入口地址，如图中 12H、13H、14H 为向量地址，200、300、400 为入口地址，只要访问向量地址所指示的存储单元，便可获得入口地址。



2、软件查询法



用软件寻找中断服务程序入口地址的方法叫**软件查询法**，其框图如上图。由图可见，当查到某一中断源有中断请求时，接着安排一条转移指令，直接指向此中断源的中断服务程序入口地址，机器便能自动进入中断处理。至于各中断源对应的入口地址，则由程序员(或系统)事先确定。

（四）中断响应

1、响应中断的条件

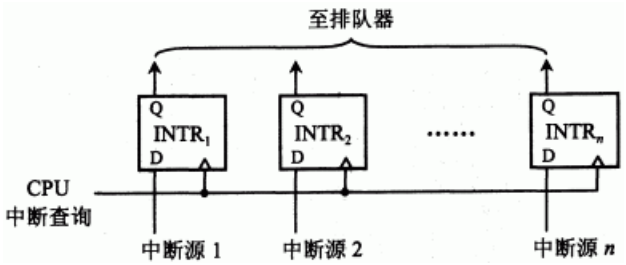
CPU 响应 I/O 中断的条件是允许中断触发器必须为“1”。

2、响应中断的时间

CPU 总是在指令执行周期结束后，响应任何中断源的请求。在指令执行周期结束后，若有中断，CPU

则进入中断周期；若无中断，则进入下一条指令的取指周期。

因为 CPU 在执行周期的结束时刻统一向所有中断源发中断查询信号，只有此时，CPU 才能获知哪个中断源有请求。如下图所示，图中 $INTR_i$(i=1,2……)是各个中断源的中断请求触发器，触发器的数据端来自各中断源，当它们有请求时，数据端为“1”，而且只有当 CPU 查询信号到触发器的时钟端时，才能置“1” $INTR_i$。



3、中断隐指令

中断隐指令即在机器指令系统中没有的指令，它是 CPU 在中断周期内由硬件自动完成的一条指令。CPU 执行中断隐指令主要完成以下操作：

（1）保护程序断点

保护程序断点就是要将当前程序计数器 PC 的内容(程序断点)保存到存储器中。它可以存在存储器的特定单元(如 0 号地址)内，也可以存入堆栈。

(2)寻找中断服务程序的入口地址

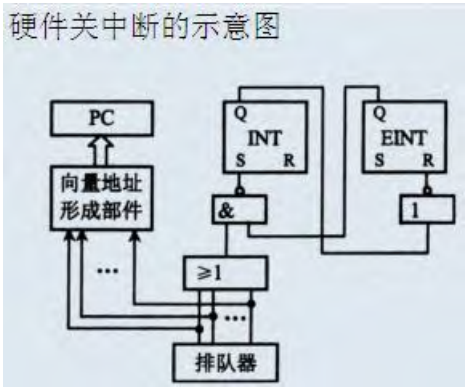
由于中断周期结束后进入下条指令(即中断服务程序的第一条指令)的取指周期，因此在中断周期内必须设法找到中断服务程序的入口地址。由于入口地址有两种方法获得，**因此在中断周期内也有两种方法寻找入口地址**：

其一，在中断周期内，将向量地址送至 PC(对应硬件向量法)，使 CPU 下一条执行无条件转移指令，转至中断服务程序的入口地址。

其二，在中断周期内，将软件查询入口地址的程序(又叫中断识别程序)其首地址送至 PC，使 CPU 执行中断识别程序，找到入口地址(对应软件查询法)。

(3)关中断

CPU 进入中断周期，意味着 CPU 响应了某个中断源的请求，为了确保 CPU 响应后所需作的一系列操作不至于又受到新的中断请求的干扰，在中断周期内必须自动关中断，以禁止 CPU 再次响应新的中断请求。



（五）保护现场和恢复现场

保护现场应该包括保护程序断点和保护 CPU 内部各寄存器内容的现场两个方面。程序断点的现场由中断隐指令完成，各寄存器的内容可在中断服务程序中由用户(或系统)用机器指令编程实现。

恢复现场是指在中断返回前，必须将寄存器的内容恢复到中断处理前的状态，这部分工作也由中断服

务程序完成。

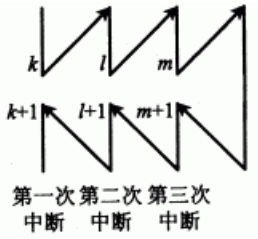
（六）中断屏蔽技术

中断屏蔽技术主要用于多重中断。

1、多重中断的概念

当 CPU 正在执行某个中断服分程序时，另一个中断源又提出了新的中断请求，而 CPU 又响应了这新的请求，暂时停止了正在运行的服务程序，转去执行新的中断服务程序，这就叫**多重中断**，或叫**中断嵌套**。

如果 CPU 对新的请求不予响应，待执行完当前的服务程序后再响应，即为**单重中断**。中断系统若要具有处理多重中断的功能，必须具备各项条件。多重中断示意如下图所示：

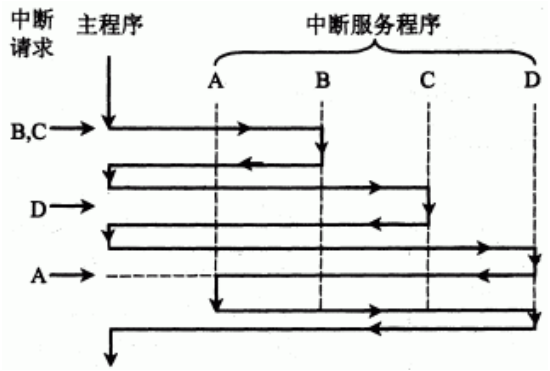


2、实现多重中断的条件

（1）提前设置“开中断”指令。

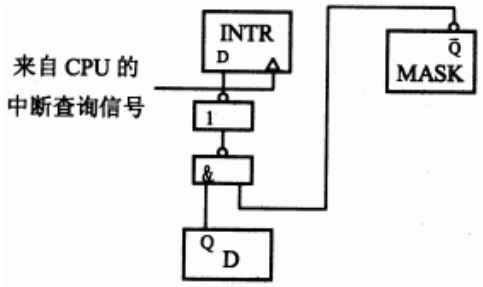
（2）优先级别高的中断源有权中断优先级别低的中断源。

中断处理示意图如下图所示：



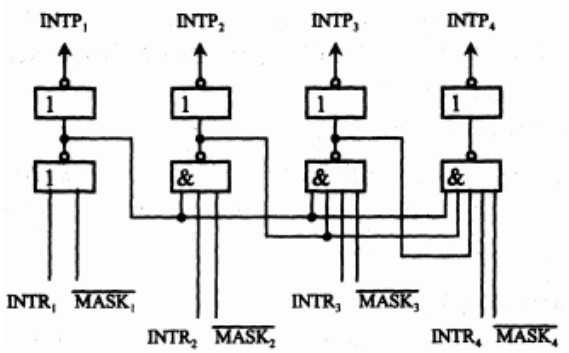
3、屏蔽技术

（1）屏蔽触发器与屏蔽字



上图示出了程序中断接口电路中完成触发器 D、中断请求触发器 INTR 和屏蔽触发器 MASK 三者之间的关系。

具有屏蔽功能的排队器，如下图所示。



对应每个中断请求触发器就有一个屏蔽触发器，将所有屏蔽触发器组合在一起，便构成一个屏蔽寄存器，屏蔽寄存器的内容称作**屏蔽字**。屏蔽字与中断源的优先级别是一一对应的，如下表所示：

优先级	屏蔽字
1	1111111111111111
2	0111111111111111
3	0011111111111111
4	0001111111111111
5	0000111111111111
6	0000011111111111
⋮	⋮
15	0000000000000011
16	0000000000000001

（2）屏蔽技术可改变优先等级

利用屏蔽技术可以任意改变中断源的优先等级。例如 5 级中断源级别高于 6 级中断源，但若在中断服务程序中先设置一个屏蔽字 0000101111111111，这样，当 5、6 级中断源同时有请求时，由于 5 级被屏蔽，6 级未被屏蔽，因此 CPU 优先响应 6 级中断源的请求。只有当处理完 6 级中断源的请求后，再设一屏蔽字 0000011111111111，CPU 才能响应 5 级中断源的请求。

(3)屏蔽技术的其他作用

屏蔽技术还能给程序控制带来更大的灵活性。例如，在浮点运算中，当程序员估计到执行某段程序时可能出现“阶上溢”，但又不希望因“阶上溢”而使机器停机，为此可设一屏蔽字，使对应“阶上溢”的屏蔽位为“1”，这样，即使出现“阶上溢”，机器也不停机。

4、多重中断的断点保护

多重中断时，每次中断出现的断点都必须保存起来。中断系统对断点的保存都是在中断周期内由中断隐指令实现的，对用户是透明的。

断点可以保存在堆栈中，由于堆栈先进后出的特点，依次将程序的断点压入堆栈中。出栈时，按相反顺序便可准确返回到程序间断处。

断点也可保存在特定的存储单元内。

第八章 CPU 关键词汇

1、分析指令

分析指令包括两部分内容，其一，分析此指令要完成什么操作，即控制器需发出什么操作命令；其二，分析参与这次操作的操作数地址，即操作数的有效地址。

2、执行指令

执行指令就是根据分析指令产生的“操作命令”和“操作数地址”的要求，开成操作控制信号序列（不同的指令有不同的操作控制信号序列），通过对运算器、存储器以及 I/O 设备的操作，执行每条指令。

3、控制单元 CU

控制单元 CU 是提供完成机器全部指令操作的微操作命令序列部件。现代计算机中微操作命令序列的形成方法有两种，一种是组合逻辑设计方法，为硬联线逻辑；另一种是微程序设计方法，为存储逻辑。

4、中断系统

中断系统主要用于处理计算机的各种中断。

5、指令周期

CPU 每取出并执行一条指令所需的全部时间，也即 CPU 完成一条指令的时间叫指令周期。

6、中断周期

当 CPU 采用中断方式实现主机与 I/O 交换信息时，CPU 在每条指令执行阶段结束前， 都要发中断查询信号，以检测是否有某个 I/O 提出中断请求。如果有请求，CPU 则要进入中断响应阶段，又称中断周期。

7、指令预取

由指令部件取出一条指令，并将它暂存起来，如果执行部件空闲，就将暂存的指令传给执行部件执行。与此同时，指令部件又可取出下一条指令并暂存起来，这就叫指令预取。

8、相关问题

所谓相关问题是指程序的相近指令之间出现某种关联，使指令流水线出现停顿，影响了指令流水线的效率。指令间的相关大体可分控制相关和数据相关两类。

（1）控制相关

如果一条指令要等前一条(或几条)指令作出转移方向的决定后，才能进入流水线，便发生了**控制相关**

（2）数据相关

数据相关是发生在几条相近的指令间共用同一个存储单元或寄存器时发生的。例如某条指令为了计算操作数的地址需用到某寄存器的内容。但是产生这“内容”的指令还未执行结束，也即还没有产生结果，这时流水线也只能暂停等待。

9、超标量(Super scalar)技术

超标量(Super scalar)技术是指在每个时钟周期内可同时并发多条独立指令，即以并行操作方式将两条或两条以上指令编译并执行。

10、超流水线(Super pipe lining) 技术

超流水线(Super pipe lining) 技术是将一些流水线寄存器插入到流水线段中，好比将流水线再分道，它将原来的一个时钟周期又分成三段，使超级流水线的处理器周期比一般流水线的处理器周期短，这样，在原来的时钟周期内，功能部件被使用三次，使流水线以 3 倍于原来时钟频率的速度运行。

11、指令流水线

指令流水线是将指令的整个执行过程用流水线进行分段处理。

12、中断源

能引起中断的各个因素称作中断源，中断源可分两大类：一类为不可屏蔽中断，这类中断 CPU 不能禁止响应，如电源掉电；另一类属可屏蔽中断，对可屏蔽中断源的请求，CPU 可根据该中断源是否被屏蔽来确定是否给予响应，若未屏蔽则能响应；若已被屏蔽，则 CPU 不能响应。

13、硬件向量法

硬件向量法就是利用硬件产生向量地址，再由向量地址找到中断服务程序的入口地址。向量地址由中断向量地址形成部件产生，这个电路可分散设置在各个接口电路中，也可设置在 CPU 内。

14、多重中断

当 CPU 正在执行某个中断服分程序时，另一个中断源又提出了新的中断请求，而 CPU 又响应了这新的请求，暂时停止了正在运行的服务程序，转去执行新的中断服务程序，这就叫**多重中断**，或叫**中断嵌套**。

15、单重中断

如果 CPU 对新的请求不予响应，待执行完当前的服务程序后再响应，即为**单重中断**。

16、屏蔽字

对应每个中断请求触发器就有一个屏蔽触发器，将所有屏蔽触发器组合在一起，便构成一个屏蔽寄存器，屏蔽寄存器的内容称作**屏蔽字**。屏蔽字与中断源的优先级别是一一对应的。

第八章 CPU FAQ

一、指令和数据均存放在内存中，CPU 如何从时间和空间上区分它们是指令还是数据？

时间上讲，取指令事件发生在取指周期，取数据事件发生在执行周期。

空间上讲，从内存读出的指令流向控制器；从内存中读出的数据流一定流向运算器。

二、举出 CPU 中 6 个主要寄存器并说明其名称及功能？

（1） 指令寄存器（IR）：用来保存当前正在执行的一条指令。

（2） 程序计数器（PC）：用来确定下一条指令的地址。

（3） 地址寄存器（AR）：用来保存当前 CPU 所访问的内存单元的地址。

（4） 缓冲寄存器（DR）：<1>作为 CPU 和内存、外部设备之间信息传送的中转站。
<2>补偿 CPU 和内存、外围设备之间在操作速度上的差别。
<3>在单累加器结构的运算器中，缓冲寄存器还可兼作为操作数寄存器。

（5） 通用寄存器（AC）：当运算器的算术逻辑单元（ALU）执行全部算术和逻辑运算时，为 ALU 提供一个工作区。

（6） 状态条件寄存器:保存由算术指令和逻辑指令运行或测试的结果建立的各种条 件码内容。除此之外，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。

三、简述 CPU 的基本功能，CPU 有哪些部分组成？

(1)指令控制：程序的顺序控制

(2)操作控制：CPU 管理并产生由内存取出的每条指令的操作信号，把各种操作信号送往相应的部件，从而控制这些部件按指令的要求进行动作。

(3)时间控制：对各种操作实施时间上的定时。

(4)数据加工：对数据进行算术运算和逻辑运算处理。完成数据的加工处理，是 CPU 的根本任务

1. 运算器
2. 控制器
3. Cache(指令 cache 和数据 cache)

四、CPU 的功能？

CPU 是由运算器和控制器组成的， 在这里我们**重点介绍控制器的功能**。

其基本功能是取指令、分析指令和执行指令。

1．取指令

控制器必须具备能自动地从存储器中取出指令的功能。为此，要求控制器能自动形成指令的地址，并能发出取指令的命令，将对应此地址的指令取到控制器中。第一条指令的地址可以人为指定，也可由系统设定。

2．分析指令

分析指令包括两部分内容，其一，分析此指令要完成什么操作，即控制器需发出什么操作命令；其二，分析参与这次操作的操作数地址，即操作数的有效地址。

3．执行指令

执行指令就是根据分析指令产生的“操作命令”和“操作数地址”的要求，开成操作控制信号序列（不同的指令有不同的操作控制信号序列），通过对运算器、存储器以及 I/O 设备的操作，执行每条指令。

此外，控制器还必须能控制程序的输入和运算结果的输出（即控制主机与 I/O 交换信息）以及对总线的管理， 甚至能处理机器运行过程中出现的异常情况和特殊请求，即处理中断的能力。

五、为了避免访存冲突，可采用哪些方法？

为了避免冲突，可采用如下一些方法：

①设置两个独立的存储器分别存放操作数和指令，以免取指令和取操数同时进行时互相冲突，使取某条指令和取另一条指令的操作数实现时间上的重叠。

②采用指令预取技术，如在 CPU(8086)中设置指令队列， 将指令预先取到指令队列中排队。指令预取技术的实现基于访存周期很短的情况，如在执行指令阶段，取数时间很短，因此在执行指令时，主存会有空闲，此时，只要指令队列空出，就可取下一条指令，并放至空出的指令队列中，从而保证在执行第 K 条指令的同时对第 K+1 条指令进行译码，实现“执行 K”与“分析 K+1”的重叠。

六、引发中断有哪些因素？

引起中断的各种因素

（1）人为设置的中断

这种中断一般叫做自愿中断，因为它是在程序中人为设置的，故一旦机器执行这种人为中断时，便自愿停止现行程序而转入中断处理。

（2）程序性事故

如定点溢出、浮点溢出、操作码不能识别、除法中出现“非法”等，这些都属于由程序设计不周而引起的中断。

（3）硬件故障

硬件故障类型很多，如插件接触不良、通风不良、磁表面损坏、电源掉电等都属硬设备故障。

（4）I/O 设备

I/O 设备被启动以后，一旦其准备就绪，便向 CPU 发出中断请求。每个 I/O 设备都能发中断请求，因此这种中断与计算机所配置的 I/O 设备多少有关。

（5）外部事件

用户通过键盘来中断现行程序属于外部事件中断。

七、中断系统需要解决哪些问题？

中断系统需解决的问题

(1) 各中断源如何向 CPU 提出中断请求；

(2) 当多个中断源同时提出中断请求时，中断系统如何确定优先响应哪个中断源的请求；

(3) CPU 在什么条件、什么时候、以什么方式来响应中断；

(4) CPU 响应中断后如何保护现场；

(5) CPU 响应中断后，如何停止原程序的执行而转入中断服务程序的入口地址；
(6) 中断处理结束后，CPU 如何恢复现场，如何返回到原程序的间断处；
(7) 在中断处理过程中又出现了新的中断请求，CPU 该如何处理。

第八章 CPU 拓展资源

——— 从 Larrabee 看英特尔未来 CPU 开发趋势

“Larrabee（阿拉伯人）”的重要性并不在于英特尔名为“Larrabee”的图形芯片。Larrabee 反映了英特尔未来 CPU 架构的发展方向。Larrabee 的核心理念是最新的 x86 指令 LNI （Larrabee New Instruction，Larrabee 新指令）以及 CPU 核心网络。而且，Larrabee 还引导着英特尔主流 CPU 技术的发展趋势。

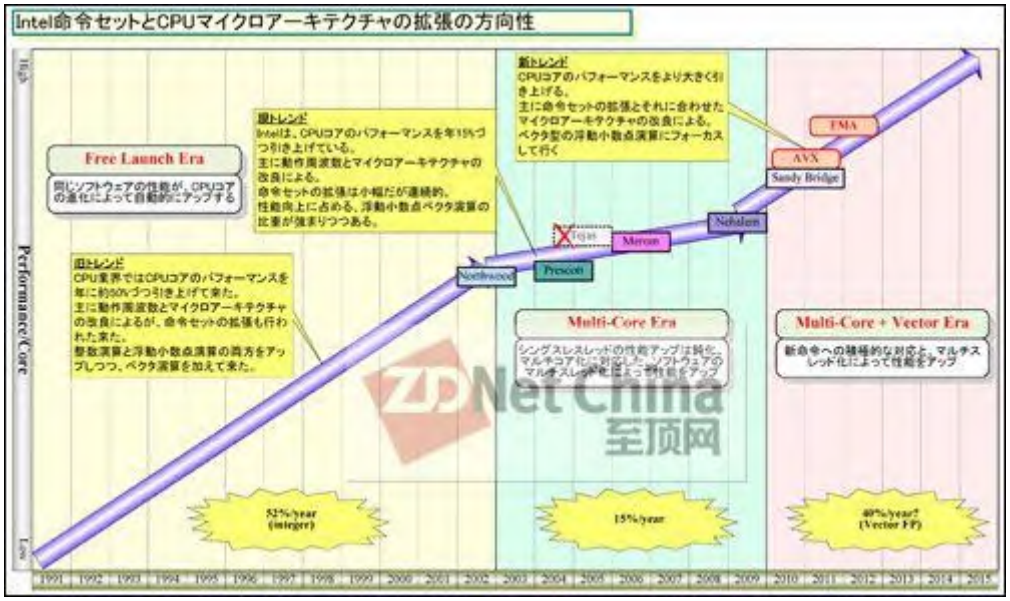
因为 PC 服务器 CPU 的主流趋势要么是加强数据级并行性（Data-Level Parallelism，DLP），要么是加强线程级并行性（Thread-Level Parallelism，TLP），所以有人会认为在开发这种架构的时候 Larrabee 起到了帮助作用，而且 Larrabee 还将应用于检测工具中。

单从一款产品来说，Larrabee 的成功与否并不重要。重要的不是 Larrabee 本身，不管 Larrabee 是否能够取得成功，我们都可以从 Larrabee 看到未来英特尔 CPU 发展的趋势，从 Larrabee 我们能推测出英特尔新一代架构的成功与否。

准确地说，未来英特尔将把 1/3 的精力放在 CPU 架构上。因为目前英特尔公司内部有三个开发 x86 架构 CPU 的团队，其中一个团队主要负责 Larrabee 和指令集。

这三个团队中有一个团队是隶属于英特尔数字企业部门的俄勒冈州 Hilzboro 开发中心，主要负责 Pentium III/4 和 Core i7 的研发工作。第二个团队是隶属于英特尔移动部门的以色列 Haifa 开发中心，主要负责 Pentium M 和 Core 2 的研发工作。最后一个是负责 Atom 系统研发工作的德克萨斯州 Austin 开发中心。Hilzboro 和 Haifa 团队开发主流 x86 CPU 架构已经有两年时间了。

Larrabee 是由三个团队中的 Hilzboro 团队研究人员开发的。有趣的是，不仅这款产品，包括升级的指令集架构在内都是属于每个团队的。



图中是英特尔指令集和 CPU 架构扩展的方向（点击放大），从图中可以看出来 Intel 的 CPU 发展可以划分成三个阶段，第一个阶段是从 1991 年到 2002 年的单核时期，这期间主要以提升 CPU 主频来增强 CPU

的性能，此间年均性能增长幅度在 52%左右，而从 2003 年开始至 2009 年，也就是我们现在所处的多核（线程）处理器阶段，通过多核与多线程技术来进一步提升性能，并加强了浮点运算能力，年均增长幅度在 15%左右，从 2009 年开始，随着 AVX 等新一代向量指令集的加入，CPU 将进化入新的多核心＋向量处理的阶段，浮点运算能力也将达到新的水平，预计年均性能增长幅度可达 40%

英特尔将在 2010 年推出装载 AVX（Advanced Vector Extensions，高级向量扩展指令集）新型 SIMD 指令的 CPU 产品。而且据说 Haifa 团队研发的更新一代处理器“Sandy Bridge”将采用这个新的指令扩展。实际上，在英特尔 IDF 大会上登台讲解 AVX 的演讲者中大多数是 Haifa 团队成员。总之，出现在英特尔指令集升级路线图中的 AVX 不仅是 Haifa 团队开发的，而且被加载到了 Haifa 团队开发的 CPU 中。

另一个是由 Hilzboro 团队开发的 LNI（Larrabee New Instruction，Larrabee 新指令）。如果开发了这个指令集的团队实现了每个指令集的扩展，那么 LNI 或者完善后的指令就将被主流 CPU 所采用，当然这也将改变 Hilzboro 开发主流 CPU 的路线，也许英特尔计划在 2012 年推出代号为 Haswell 的 CPU 将成为成功的新一代产品。总之，Larrabee 将为英特尔 2012 年之后 CPU 架构奠定下坚实的基础。

从这点来看，可以推测出明年（2009 年）以后 Intel 处理器的指令集架构将出现一种分裂的态势。一种是面向主流 x86 处理器的 256bit 向量的 AVX 扩展指令，一种则是高吞吐量处理器所采用的 512bit 向量的 LNI 指令集，这种将向量指令集扩展成两个并行发展的系统，给人一种“走向分裂”的异样感觉。也因此，Intel 内部两个研发团队之间相互竞争的状态也明显体现出来。

如果 Intel 把 Larrabee 产品作为未来的一个发展走向，并因此新设立了视觉计算业务群组(VCG,Visual Computing Group），那么 LNI 向 x86 处理器指令架构的扩张将会容易很多，指令集部分也将具备了一贯性。如果 Intel 真的采取了这一战略，那么作为产品而开发的 Larrabee 所被赋予的地位和战略重要性，将在很长时期内不可动摇。然而，以现有的产品计划，Larrabee 更像是 x86 处理器的一个“鬼影”，或者说是一个潜在的“破坏者”，从这些策略上的差异上也可以看出 Intel 内部各研发部门之间的分歧，或者说是一种“隔阂”。

而且，Hilzboro 团队的负责人 Patrick Gelsinger（英特尔数字企业部门高级副总裁兼总经理）暗示说，AVX 不仅将继续保留在 SSE 计划中，而且还将长期作为指令集扩展路线图的一部分。也许在关注完 AVX 之后，我们又要继续关注 LNI 了。

第九章 控制单元的功能 课堂笔记

◆ **主要知识点掌握程度**

了解控制单元的外特性，控制信号在执行一条指令中的流程；了解多级时序系统；掌握概念：指令周期，机器周期，时钟周期；

◆ **知识点整理**

一、微操作命令的分析

控制单元具有发出各种微操作命令(即控制信号)序列的功能。

计算机的功能就是执行程序。在执行程序的过程中，控制单元要写出各种微操作命令，而且不同的指令对应不同的命令。

下面按指令周期的四个阶段进一步分析其对应的微操作命令。

（一） 取指周期

取指令的过程可归纳为以下几个操作：

(1)现行指令地址送至存储器地址寄存器，记作 PC→MAR；

(2)向主存发读命令，启动主存作读操作，记作 1→R；

(3)将 MAR(通过地址总线)所指的主存单元中的内容(指令)经数据总线读至 MDR 内，记作 M(MAR)→MDR；

(4)将 MDR 的内容送至 IR，记作 MDR→IR：

(5)形成下一条指令的地址，记作(PC)+1→PC。

（二）间指周期

间址周期完成取操作数有效地址的任务，具体操作如下：

(1)将指令的地址码部分(形式地址)送至存储器地址寄存器，记作 Ad(IR) →MAR；

(2)向主存发读命令，启动主存作读操作，记作 1→R；

(3)将 MAR(通过地址总线)所指的主存单元中的内容(有效地址)经数据总线读至 MDR 内，记作 M(MAR)→MDR；

(4)将有效地址送至指令寄存器的地址字段，记作 MDR→Ad(IR)。此操在有些机器中可省略。

（三）执行周期

不同指令执行周期的微操作是不同的，下面分别讨论非访存指令、访存指令和转移类指令的微操作。

1、非访存指令

这类指令在执行周期不访问存储器。

(1)清除累加器指令 CLA。该指令在执行阶段只完成清除累加器操作，记作 0→ACC。

(2)累加器取反指令 COM。该指令在执行阶段只完成累加器内容取反，结果送果加器的操作，记作 $\overline{ACC} \rightarrow ACC$

(3)算术右移一位指令 SHR。该指令在执行阶段只完成累加器内容算术右移一位的操作，记作 L(ACC)→R(ACC)，$ACC_0 \rightarrow ACC_0$。(ACC 的符号位不变)

(4)循环左移—位指令 CSL。该指令在执行阶段只完成累加器内容循环左移一位的操作，记作 R(ACC)→L(ACC)，$ACC_0 \rightarrow ACC_n$(或 $p^{-1}(ACC)$)

(5)停机指令 STP。计算机中有一运行标志触发器 G，当 G＝1 时，表示机器运行：当 G＝0 时，表示停机。STP 指令在执行阶段只需将运行标志触发器置“0”，记作 0→G。

2、访存指令

(1)加法指令 ADD X

该指令在执行阶段需完成累加器内容与对应于主存 X 地址单元的内容相加，结果送累加器的操作，具体为：

①将指令的地址码部分送至存储器地址寄存器，记作 Ad(IR) →MAR；

②向主存发读命令，启动主存作读操作，记作 1→R；

③将 MAR(通过地址总线)所指的主存单元中的内容(操作数)经数据总线读至 MDR 内，记作 M(MAR)→MDR；

④给 ALU 发加命令，将 ACC 的内容和 MDR 的内容相加，结果存于 ACC，记作(ACC)+(MDR) →ACC。

也有的加法指令指定两个寄存器的内容相加，如 ADD AX，BX，如 ADD AX，BX，该指令在执行阶段无需访存，只需完成(AX)+(BX) →AX 的操作。

（2）存数指令

该指令在执行阶段需将累加器 ACC 的内容存于主存的 X 地址单元中，具体操作如下：

①将指令的地址码部分送至存储器地址寄存器，记作 Ad(IR) →MAR；

②向主存发写命令，启动主存作写操作，记作 1→W；

③将累加器内容送至加 R，记作 ACC→MDR

④将 MDR 的内容(通过数据总线)写入到 MAR(通过地址总线)所指的主存单元中，记作 MDR→M（MAR）。

(3)取数指令 LDA X

该指令在执行阶段需将主存 X 地址单元的内容取至累加器 ACC 中，具体操作如下：

①由将指令的地址码部分送至存储器地址寄存器，记作 Ad(IR) →MAR；

②向主存发读命令，启动主存作读操作，记作 1→W

③将 MAR(通过地址总线)所指的主存单元中的内容(操作数)经数据总线读至 MDR 内，记作 M(MAR)→MDR

④将 MDR 的内容送至 ACC，记作 MDR→ACC。

3、转移类指令

这类指令在执行阶段也不访问存储器。

(1) 无条件转移指令 JMP X

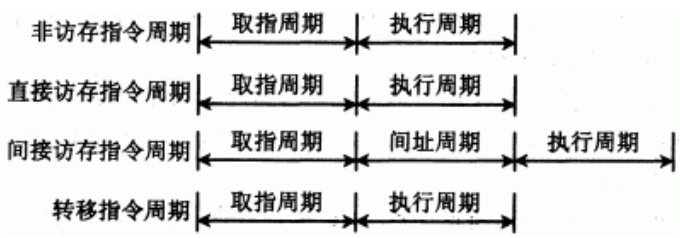
该指令在执行阶段完成将指令的地址码部分 X 送至 PC 的操作，记作 Ad(IR) →PC。

(2) 条件转移(负则转)指令 BAN X

该指令根据上一条指令运行的结果决定下一条指令的地址，若结果为负(累加器最高位为 1，即 A_0=1)，则指令的地址码送至 PC，否则程序按原顺序执行。由于在取指阶段已完成了(PC)+1→PC，所以当累加器结果不为负(即 A_0=0)时，就按取指阶段形成的 PC 执行，记作 A_0 ·Ad(IR)十 A_0(PC) →PC。

4、三类指令周期的比较

三类指令的指令周期如下图所示。



（四）中断周期

在执行周期结束时刻，CPU 要查询是否有允许中断的中断事件发生，如果有则进入中断周期。在中断周期，由中断隐指令自动完成保护断点、寻找中断服务程序入口地址以及硬件关中断的操作。

假设程序断点存至主存的 0 地址单元，且采用硬件向量法寻找入口地址；则**在中断周期需完成**如下操作：

(1)将特定地址“0”送至存储器地址寄存器，记作 0→MAR；

(2)向主存发写命令，启动存储器作写操作，记作 1→W；

(3)将 PC 的内容(程序断点)送至 MDR，记作 PC→MDR；

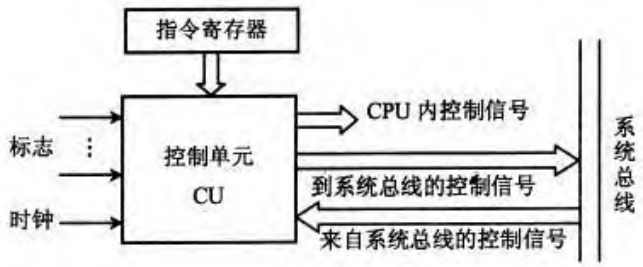
(4)将 MDR 的内容(程序断点)通过数据总线写入到 MAR(通过地址总线)所指示的主存单元(0 地址单元)中，记作 MDR→M(MAR)；

(5)向量地址形成部件的输出送至 PC，记作向量地址→PC，为下一条指令周期作准备；

(6)关中断，将允许中断触发器清 0，记作 0→EINT(该操作可直接由硬件线路完成)。

二、控制单元的功能

（一）控制单元外特性



1、输入信号

(1)时钟

上述各种操作有两点应特别注意：

• 完成每个操作都需占用一定时间；

• 各个操作是有先后顺序的。例如存储器读操作要用到 MAR 中的地址，故 PC→MAR 应先于 M(MAR)→MDR。

(2)指令寄存器

现行指令的操作码决定了不同指令在执行周期所需完成的不同操作，故指令的操作码字段是 CU 的输入信号，它与时钟配合可产生不同的控制信号。

(3)标志

控制单元有时需依赖 CPU 当前所处的状态(如此 U 操作的结果)产生控制信号， 如 BAN 指令，控制单元要根据上条指令的结果是否为负而产生不同的控制信号。因此“标志”也是 CU 的输入信号。

(4)来自控制总线的控制信号，如中断请求、DMA 请求。

2、输出信号

(1)CPU 内的控制信号

主要用于 CPU 内的寄存器之间的传送和控制 ALU 实现不同的操作。

(2)送至控制总线的信号

如命令主存或 I/O 读/写，中断响应等。

（二）多级时序系统

1、指令周期

指令周期是提指从取指令、分析指令和执行指令所经历的全部时间。

由于各种指令的功能不同，因此，相应的指令长度也不尽相同。

2、机器周期

通常把一个指令周期分为若干个机器周期，执行一条指令可按功能分成若干个确定的任务，如取指令、地址计算和取操作数，执行指令等，因此，一般机器周期有取数周期、执行周期、中断周期等。

确定机器周期时，通常要分析机器指令的执行步骤及每一步骤所需的时间。

通过对机器指令执行步骤的分析，会找到一个基准时间，在这个基准时间内，所有指令的操作都能结束。若以这个基准时间定为机器周期，显然不是最合理的。因为只有以完成复杂指令功能所需的时间(最长时间)作为基准，才能保证所有指令在这时间内完成全部操作，这对简单指令来说，显然是一种浪费。

进一步分析发现， 机器内的各种操作大致可归属为对 CPU 内部的操作和对主存的操作两大类，由于 CPU 内部的操作速度较快，CPU 访存的操作时间较长，因此通常以访问一次存储器的时间定为基准时间较为合理，这基准时间就是机器周期。

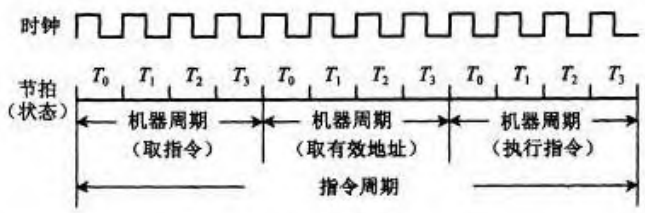
2、时钟周期(节拍、状态)

在一个机器周期里可完成若干个微操作，每个微操作都需一定的时间，可用时钟信号来控制产生每一个微操作命令。这样，一个机器周期内就包含了若干个时钟周期，又称节拍或状态。每个节拍的宽度正好对应一个**时钟周期**。在每个节拍内机器可完成一个或几个需同时执行的操作。

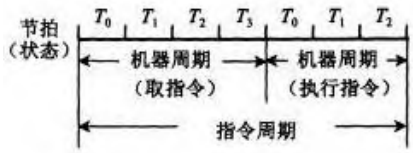
3、多级时序系统

一个指令周期包含若干个机器周期，一个机器周期又包含若干个时钟周期(节拍)，每个指令周期内的机器周期数可以不等，每个机器周期内的节拍数也可以不等。

机器周期、节拍(状态)组成了多级时序系统。



(a) 定长的机器周期



(b) 不定长的机器周期

(四) 控制方式

控制单元控制一条指令执行的过程，实质上是依次执行一个确定的微操作序列的过程。由于不同指令所对应的微操作数及其复杂程度不同，因此每条指和每个微操作所需的执行时间也不同。通常将如何形成控制不同微操作序列所采用的时序控制方式称作 CU 的控制方式。**常见的控制方式有同步控制、异步控制、联合控制和人工控制四种。**

1、同步控制方式

任何一条指令或指令中的任何一个微操作的执行，都由事先确定且有统一基准时标的时序信号所控制的方式，叫做**同步控制方式**。

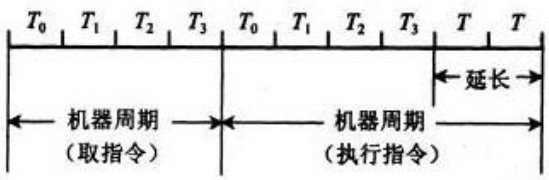
为了提高 CPU 的效率，在同步控制中又有三种方案：

(1)采用完全统一的机器周期和节拍

这种方案的**特点**是：不论指令所对应的微操作序列有多长，也不管微操作的简繁，一律以最长的微操作序列和最繁的微操作作为标准，采取完全统一的、具有相同时间间隔和相同数目的节拍作为机器周期来运行各种不同的指令。显然，这种方案对于微操作序列较短的指令来说，会造成时间上的浪费。

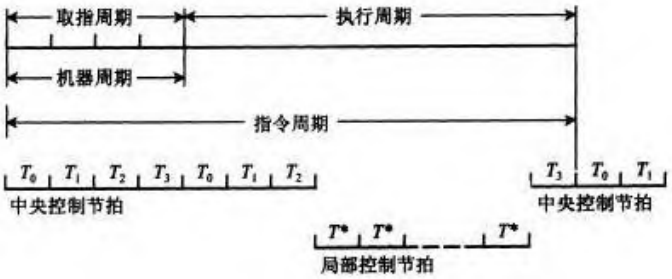
(2)采用不同节拍的机器周期

这种方案每个机器周期内的节拍数可以不等。这种控制方式可解决微操作执行时间不统一的问题。通常把大多数微操作安排在一个较短的机器周期内完成，而对某些复杂的微操作，采用延长机器周期或增加节拍的办法来解决。



(3)采用中央控制和局部控制相结合的方法

这种方案将机器的大部分指令安排在统一的、较短的机器周期内完成，称为**中央控制**，而将少数操作复杂的指令中的某些操作采用局部控制方式来完成，如乘除法和浮点运算等。



2、异步控制方式

异步控制方式不存在基准时标信号，没有固定的周期节拍和严格的时钟同步，执行每条指令和每个操作需要多少时间就占用多少时间。

这种方式微操作的时序由专门的应答线路控制，即当 CU 发出执行某一微操作的控制信号后，等待执行部件完成了该操作后发回“回答”(或“结束”)信号，**再开始新的微操作， 使 CPU 没有空闲状态，但因需要采用各种应答电路，故**其结构比同步控制方式复杂。

3、联合控制方式

同步控制和异步控制相结合就是联合控制方式。这种方式对各种不同指令的微操作实行大部分统一、小部分区别对待的办法。例如，对每条指令都有的取指令操作，采用同步方式控制；对那些时间难以确定的微操作，如 I/O 操作， 则采用异步控制，以执行部件送回的“回答”信号作为本次微操作的结束。

4、人工控制方式

人工控制是为了调机和软件开发的需要，在机器面板或内部设置一些开关或按键，来达到人工控制的目的。

(1)Reset(复位)键

按下 Reset 键，使计算机处于初始状态。 当机器出现死锁状态或无法继续运行时，可按此键。若在机器运行时按此键，将会破坏机器内某些状态而引起错误，因此要慎用。有些微机未设此键，当机器死锁时，可采用停电后再加电的办法重新启动计算机。

(2)连续或单条执行转换开关

为调机的需要，有时需观察执行完一条指令后的机器状态，有时又需要观察连续运行程序后的结果，设置连续或单条执行转换开关，能为用户提供这两种选择。

(3)符合停机开关

有些计算机还配有符合停机开关，这组开关指示存储器的位置，当程序运行到与开关指示的地址相符时，机器便停止运行，称为**符合停机**。

第九章 控制单元的功能 关键词汇

1、控制单元

CU,具有发出各种微操作命令(即控制信号)序列的功能，完成操作步骤的控制和协调。

2、指令格式

是计算机指令编码的格式，指定指令中编码字段的个数、各个字段的位数以及各个字段的编码方式。

3、指令周期

从一条指令的启动到下一条指令的启动的间隔时间。

4、机器周期

指令执行中每一步操作所需的时间。

5、同步控制方式

任何一条指令或指令中的任何一个微操作的执行，都由事先确定且有统一基准时标的时序信号所控制的方式，叫做**同步控制方式。**

6、时钟周期

在一个机器周期里可完成若干个微操作，每个微操作都需一定的时间，可用时钟信号来控制产生每一个微操作命令。这样，一个机器周期内就包含了若干个时钟周期，又称节拍或状态。每个节拍的宽度正好对应一个**时钟周期。**

7、微指令

控制器存储的控制代码，分为操作控制部分和顺序控制部分。

8、微地址

微指令在控制存储器中的存储地址。

第九章 控制单元的功能 FAQ

一、微指令编码有哪三种方式？微指令格式有哪几种？微程序控制有哪些特点？

微指令编码方式有三种：直接表示法、编码表示法、混合表示法。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。水平型微指令又分为三种：全水平型微指令、字段编码的水平型微指令、直接和编码相混合的水平型微指令。微程序的控制器具有规整性、可维护性和灵活性的优点，可实现复杂指令的操作控制，使得在计算机中可以较方便地增加和修改指令，甚至可以实现其他计算机的指令。

二、控制单元有哪些作用？

计算机能自动协调地工作是在控制单元 CU 的统一指挥下进行的，为完成不同指令所发出的各种操作命令，这些命令(又称控制信号)控制着计算机的所有部件有次序地完成相应的操作，以达到执行程序的目的。

三、概述取指令的过程。

(1)现行指令地址送至存储器地址寄存器，记作 PC→MAR；

(2)向主存发读命令，启动主存作读操作，记作 1→R；

(3)将 MAR(通过地址总线)所指的主存单元中的内容(指令)经数据总线读至 MDR 内，记作 M(MAR)→MDR；

(4)将 MDR 的内容送至 IR，记作 MDR→IR：

(5)形成下一条指令的地址，记作(PC)+1→PC。

四、简述间址周期有哪些操作？

间址周期完成取操作数有效地址的任务，具体操作如下：

(1)将指令的地址码部分(形式地址)送至存储器地址寄存器，记作 Ad(IR) →MAR；

(2)向主存发读命令，启动主存作读操作，记作 1→R；

(3)将 MAR(通过地址总线)所指的主存单元中的内容(有效地址)经数据总线读至 MDR 内，记作 M(MAR)→MDR；

(4)将有效地址送至指令寄存器的地址字段，记作 MDR→Ad(IR)。此操在有些机器中可省略。

五、在中断周期有哪些操作？

(1)将特定地址“0”送至存储器地址寄存器，记作 0→MAR；

(2)向主存发写命令，启动存储器作写操作，记作 1→W；

(3)将 PC 的内容(程序断点)送至 MDR，记作 PC→MDR；

(4)将 MDR 的内容(程序断点)通过数据总线写入到 MAR(通过地址总线)所指示的主存单元(0 地址单元)中，记作 MDR→M(MAR)；

(5)向量地址形成部件的输出送至 PC，记作向量地址→PC，为下一条指令周期作准备；

(6)关中断，将允许中断触发器清 0，记作 0→EINT(该操作可直接由硬件线路完成)。

第九章 控制单元的功能 拓展资源

——ATM 交换机中心控制单元的设计与实现

1. ATM 交换软件的核心部分

信令软件包括用户到网络信令和网络到网络信令（即 UNI/NNI 信令）都将放在中心控制单元的硬件上运行。本文主要探讨中心控制单元的硬件设计与实现。

中心控制单元为网管代理提供符合 IEEE802.3 的协议的 10BASE-T 以太网接口，为维护终端提供符合 RS232C 标准的串行接口，与各个业务单元之间采用 HDLC 接口，与交换单元采用交换矩阵接口。由于信令软件、操作系统等都要加载到中心控制单元的硬件上，并且信令信息要进行 AAL5 适配，所以要求硬件提供足够大的程序存储空间与数据存储空间，并且 CPU 的处理能力足够高。为此，我们选择了美国 Motorola 公司生产的 MPC860SAR 芯片。

MPC860SAR 芯片是一种功能强大的通信处理芯片，不仅可以实现以太网协议、HDLC 协议、UART 等多种协议的处理，还可以实现 SAR 的功能。MPC860SAR 产生的 ATM 信元先到达路由及业务管理模块，在路由及业务管理模块确定交换路由并排队缓存，之后才到达交换矩阵进行交换。只是由于 MPC860SAR 与路由及业务管理模块之间的接口不匹配，需要设计接口转换电路。

2 中心控制单元完成的功能

*接收网管代理的信息，发送至业务单元，并且将业务单元的数据上报网管代理；

*接收维护终端的信息，发送至各个业务单元，并且将各业务单元的状态等信息汇总、上报给维护终端；*完成 SAAL（Singling ATM adapt layer，信令 ATM 适配层）的 CP（公共部分），即对高层产生的信令信息进行 AAL5 适配，经 UTOPIA 接口将 ATM 信元发送到路由及业务管理模块，然后再送至交换单元。

3 中心控制单元的设计与实现

3.1 工作原理及模块间功能描述

中心控制单元由 CPU 模块、通信处理模块和路由及业务管理模块组成。正常工作状态下，CPU 模块只是控制是否允许某项通信协议的接收、发送，并不直接参与具体通信协议的处理。具体通信协议的处理、协议数据的接收、发送都是由通信处理模块独立完成的。

CPU 模块对通信处理模块的控制是通过一个命令寄存器进行的，它们之间的数据交换是通过双端口 RAM 进行的。

通信处理模块中的 ATM 业务模块即 SAR 模块的信息通过路由及业务管理模块后到达交换矩阵进行交换。通信处理模块提供 ATM 侧的收发复用的 8bit 宽的 Utopia 接口，而路由及业务管理模块提供 ATM 侧的 16bit 宽的 Utopia 接口，两者不能直接相连，两个模块之间需要进行 Utopia 接口转换，转换电路采用 CPLD 实现。

路由及业务管理模块作为 CPU 的一个外部设备，与 CPU 模块之间通过 CPU 模块之间通过 CPU 的外部数据总线、地址总线及控制总线相连。由于路由及业务管理模块的 CPU 模块提供分开的外部数据总线和地址总线，两模块之间需要进行总线接口转换，转换电路采用 EPLD 实现。

3.2 模块内部的功能描述

（1）CPU 模块

CPU 模块包括 CPU 核、32bit 宽数据存储器（SDRAM）、32bit 宽程序存储器（FLASH）和 CPU 的硬件复位配置电路。其中 CPU 核采用 32-bit PowerPC 结构，内含指令单元和指令执行单元，是用户程序的执行；SDRAM 用于存储在程序执行过程中产生或需要的数据；FLASH 用于存储用户所编制的程序，与 EPROM 器件相比突出的优点是使系统具有在线编程能力有灵活的块锁存而起到保护作用；硬件复位配置电路用于在硬件复位时对 CPU 的某些参数及复用管脚进行设置。

（2）通信处理模块

通信处理模块包含异步串行通信（UART）处理模块、以太网（Ethernet）处理模块、HDLC 通信处理模块和 AAL5 的 SAR（ATM 信元的分段与重组）功能模块等。其中 UART 处理模块用于处理维护终端的信息；Ethernet 处理模块用于处理满足 IEEE802.3 协议的网管代理的信息；HDLC 通信处理模块有两种：一种用于中心控制单元和各业务板之间的通信，另一种用于主备中心控制单元之间的通信；SAR 功能模块用来实现支持 AAL5 协议的 SAR 功能及部分 ATM 层功能。下面将分别对各功能模块进行简要介绍：

通信处理模块中各功能模块的工作原理大致相同，只是 SAR 模块稍有不同。工作原理如下：通过 CPU 模块对一些寄存器进行设置，初始化为某种特定的通信协议，然后在双端口 RAM 中的参数 RAM 设置成针对该通信协议的参数。当然，不同通信协议的接口控制信号是不同的。

SAR 功能模块用来实现支持 AAL5 协议的 SAR 功能及部分 ATM 层功能，即对高层产生的信息进行 AAL5 适配，将其分割为等长的 48 字节的 CS-PDU 再加上 5 字节的 ATM 信元头，形成 53 字节的 ATM 信元，发送至路由及业务管理模块；相反，对从路由及业务管理模块接收的 ATM 信元进行重组而发送到高层。

（3）路由及业务管理模块

***功能概述**

ATM 路由及业务管理模块作为一种先进的通信器件，它能够支持非常强大、高性能的 ATM 交换系统。RTM 丰富的性能可为系统设计提供灵活的网络业务。与 ATM 交换矩阵单元（SE）组成的交换矩阵组合应用，能够构建 622Mbps 到 160Gbps 的交换容量，该模块能提供 622Mbps UTOPIA 访问。该模块单独也能构建一个 622Mbps 交换。

RTM 利用每个 VC 接收队列，64 个接收业务类，31 个虚输出能够进行灵活的多优先级的排序运算。该排序器能被用来作为 CBR,VBR,UBR 的虚通道连接的 QoS 的计算。RTM 也提供五个独立的阻塞门限，每个以滞后而有选择的控制 AAL5 的早包丢弃(EPD)与 UBR 基于信元丢失优先级的信元丢弃，RTM 还支持完全的 VPI/VCI 头翻译，64K 的输入、输出信元缓冲，以及 VP/VC 交换。在接收侧和发送侧分别支持 16K 个 VCS。

*信元流概述

（1）在接收侧，从 UTOPIA 接口接收到的信元完成通道号的查找。接收到的信元要么丢弃要么发送到接收信元缓冲 DRAM 中，这依据六个阻塞管理检查机制（即最大门限、阻塞门限业务类组，业务类以及连接等）进行排队。

（2）当一个可用信元时间发生时间发生时，由接收侧排序器选择四个单元，并从接收信元缓冲 DRAM 读信元并且发送到交换矩阵。

（3）在发送侧，一旦从交换矩阵接收到一个信元，或者丢弃或者发送缓冲 DRAM 并且在发送队列中进行排队。这依靠十个阻塞管理检查机制（即最大门限，阻塞门限，VO,SC,SCG,SCQ 以及连接等）；

（4）信元由发送侧排序器选择发送时，从信元缓冲 DRAM 中移走，并且由相应的头翻译与分配的多点或头匹配器进行处理，然后，信元被发送到 UTOPIA 接口，并且在发送侧离开该模块。

*业务管理模块

应用 RTM 的业务管理性进行监控与 RTM 资源的控制。该模块根据分配的情况与队列深度的需要，利用 CAC 完成呼叫允许与拒绝。

（1）负责接收连接与拒绝连接：根据当前阻塞情况来接收连接或拒绝连接；

（2）负责分配业务类队列连接：对每个连接，根据不同业务的 QoS，在接收与发送方向分别设置该连接相应的队列深度；

（3）调整 SCQs 轮循业务顺序权值：保证高优先级的信元先发送；

（4）更新连接与业务类队列的深度：根据业务需要灵活地在线更改参数设置。

第十章 控制单元的设计 课堂笔记

◆ **主要知识点掌握程度**

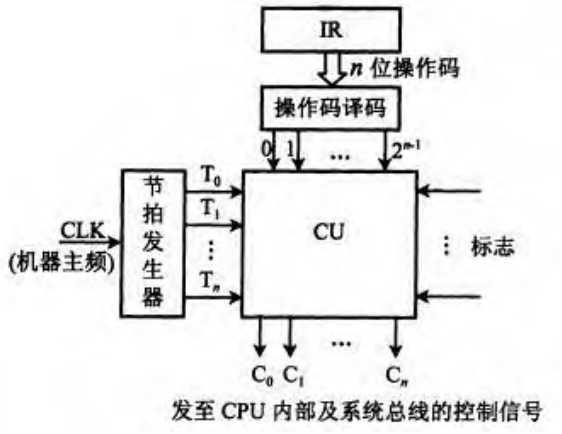
掌握学习系统的初步设计，控制单元的组合逻辑和微程序的简单设计。

◆ **知识点整理**

一 、组合逻辑设计

（一）控制单元的结构框图和预设计的结构图

以时钟为计数脉冲，通过一个计数器，又称**节拍发生器**，便可产生一个与时钟周期等宽的节拍序列。如果将指令译码和节拍发生器从 CU 中分离出来，便可得简化的控制单元框图，如下图所示。



（二）微操作的节拍安排（基本指标）

假设机器采用同步控制，每个机器周期包含 3 个节拍，而且 CPU 内部结构为非总线结构，其中 MAR 和 MDR 分别直接和地址总线和数据总线相连，并假设 IR 的地址码部分与 MAR 之间有通路。

安排微操作节拍时应注意三点：

第一，有些微操作的次序是不容改变的，故安排微操作节拍时必须注意微操作的先后顺序。

第二，凡是被控制对象不同的微操作，若能在一个节拍内执行，应尽可能安排在同一个节拍内，以节

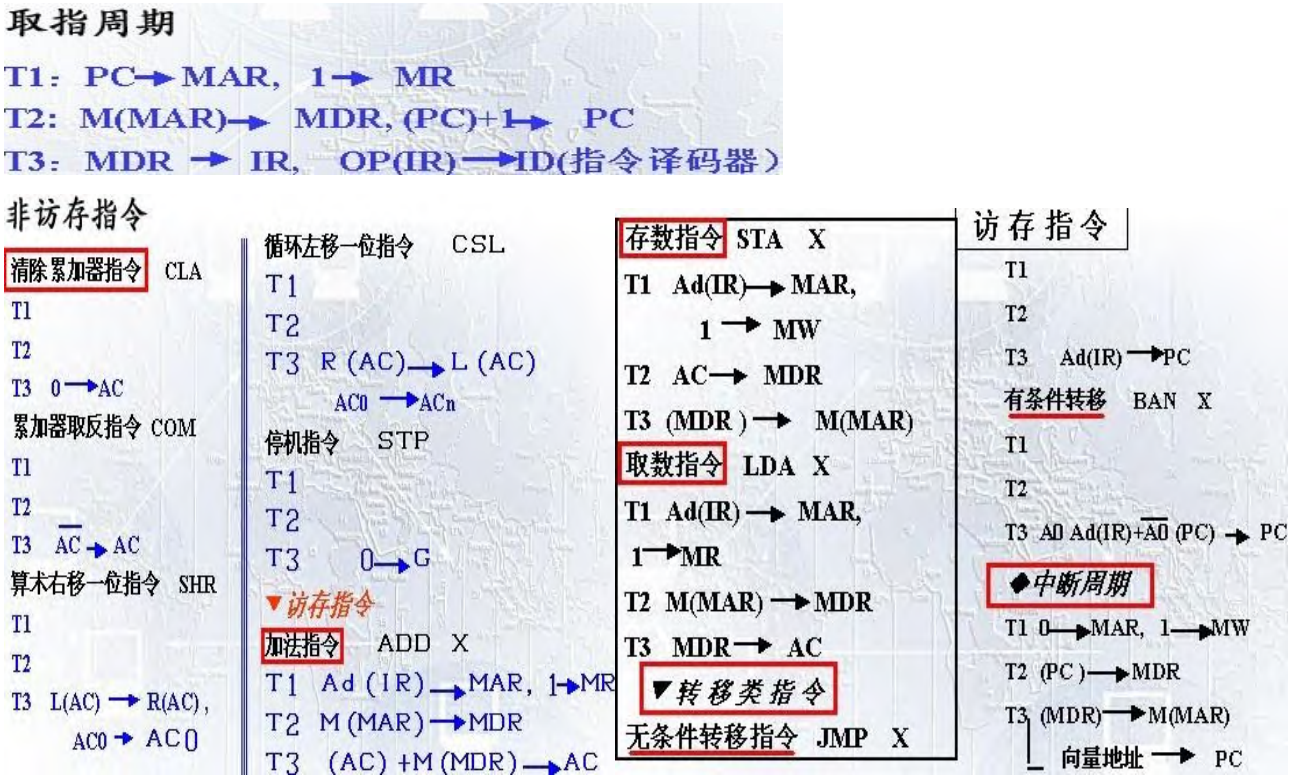
省时间。

第三，如果有些微操作所占的时间不长，应该将它们安排在一个节拍内完成，并且允许这些微操作有先后次序。

（三）组合逻辑设计步骤

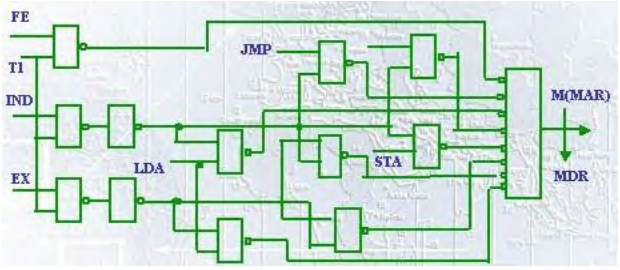
组合逻辑设计控制单元时，首先根据上述微操作的节拍安排，列出微操作命令的操作时间表，然后写出每一个微操作命令(控制信号)的逻辑表达式，最后根据逻辑表达式画出相应的组合逻辑电路图。

二、设计举例



工作周期	节拍	状态条件	微操作信号	CAL	COM	SHR	CSL	STP	ADD	SAT	LDA	JMP	BAN
取指 FE	T1		PC→MAR	1	1	1	1	1	1	1	1	1	1
			1→MR	1	1	1	1	1	1	1	1	1	1
	T2		M(MAR)→MDR	1	1	1	1	1	1	1	1	1	1
			(PC)+1→PC	1	1	1	1	1	1	1	1	1	1
	T3		MDR→IR	1	1	1	1	1	1	1	1	1	1
			OP(IR)▶ID	1	1	1	1	1	1	1	1	1	1
		I	1→IND						1	1	1	1	1
		$\bar{I}$	1→EX	1	1	1	1	1	1	1	1	1	1

工作周期	节拍	状态条件	微操作信号	CAL	COM	SHR	CSL	STP	ADD	SAT	LDA	JMP	BAN
EX 执行	T1		AD(IR)→MAR						1	1	1		
			1→R						1		1		
			1→W							1			
	T2		M(MAR)→MDR						1		1		
			AC→MDR							1			
	T3		(AC)+(MDR)→AC						1				
			MDR→M(MAR)							1			
			MDR→AC								1		
			0→AC	1									
			$\overline{AC}$→AC		1								
			R(AC)→L(AC),AC0不变			1							
			ρ^{-1}(AC)				1						
			AD(IR)→PC									1	
		A0	AD(IR)→PC										1
			0→G					1					



三、微程序设计

（一）微程序设计思想的产生

为了克服组合逻辑控制单元线路庞杂的缺点，他大胆设想来用与存储程序相类似的办法，来解决微操作命令序列的形成。将一条机器指令编写成一个微程序，每一个微程序包含若干条微指令，每一条微指令对应一个或几个微操作命令。然后把这些微程序存到一个控制存储器中，用寻找用户程序机器指令的办法来寻找每个微程序中的微指令。

由于这些微指令是以二进制代码形式表示的，每位代表一个控制信号(若该位为 1，表示该控制信号有效；若该位为 0， 表示此控制信号无效)，因此，逐条执行每一条微指令， 也就相应地完成了一条机器指令的全部操作。

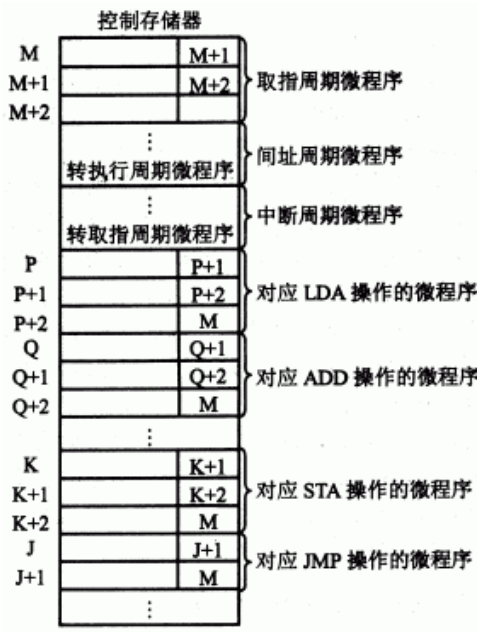
微程序控制单元的核心部件是一个控制存储器，简称控存。由于执行一条机器指令必须多次访问控制存储器，以取出多条微指令来控制执行各个微操作，因此要求控存的速度较高。

微程序设计的特点：

微程序设计省去了组合逻辑设计过程中对逻辑表达式的化简步骤，无需考虑逻辑门级数和门的扇入系数，使设计更简便。而且由于控制信号是以二进制代码的形式出现的，因此只要修改微指令的代码，就可改变操作内容，便于调试、修改，甚至增删机器指令，有利于计算机仿真。

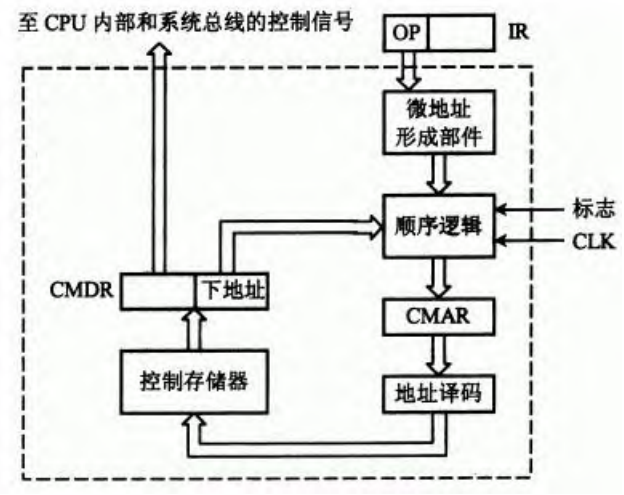
（二）微程序控制单元框图及工作原理

1、机器指令对应的微程序



微程序设计控制单元的过程就是编写每一条机器指令的微程序，它是按执行每条机器指令所需的微操作命令的先后顺序而编写的，因此，一条机器指令对应一个微程序，如上图所示。图中每一条机器指令都与一个以操作性质命名的微程序对应。

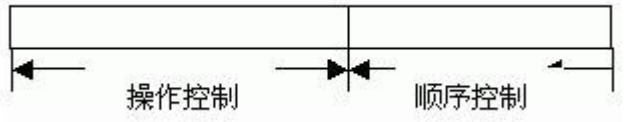
2、微程序控制单元的基本框图



图中虚线框内为微程序控制单元，它的输入有指令寄存器、各种标志和时钟，输出是输至 CPU 内部或系统总线的控制信号。

虚框内的**控制存储器（简称控存）**是微程序控制单元的核心部件，用来存放全部微程序；**CMAR** 是控存地址寄存器，用来存放欲读出的微指令地址；**CMDR** 是控存数据寄存器，用来存放从控存读出的微指令；**顺序逻辑**是用来控制微指令序列的，具体就是控制形成下一条微指令(即后继微指令)的地址， 其输入与微地址形成部件(与指令寄存器相连)、微指令的下地址字段以及外来的标志有关。

微指令的基本格式如下图所示：



3、工作原理

(1) 取指阶段

①将取指周期微程序首地址 M→CMAR ；

②取微指令，将对应控存 M 地址单元中的第一条微指令读到控存数据寄存器中，记作 CM(CMAR)→CMDR；

③产生微操作命令；

第一条微指令的操作控制字段中为“1”的各位发出控制信号，如 PC→MAR，1→R，命令主存接受程序首地址并进行读操作。

④形成下一条微指令的地址；

此微指令的顺序控制字段指出了下一条微指令的地址为 M+1，将 M+1 送至 CMAR。

⑤取下一条微指令；

将对应控存 M+1 地址单元中的第二条微指令读到 CMDR 中，即 CM(CMAR) →CMDR

⑥产生微操作指令；

由第二条微指令的操作控制字段中对应“1”的各位发出控制信号，如 M(MAR) →MDR 使对应主存 2000H 地址单元中的第一条机器指令从主存中读出送至 MDR 中。

⑦形成下一条微指令的地址：

将第二条微指令下地址字段指出的地址 M+2 送至 CMAR，即 Ad(CMDR) →CMAR

以此类推，直到取出取指周期最后一条微指令，并发出微命令为止，此时第一条机器指令 LDA X 已存至指令寄存器 IR 中。

(2)执行阶段

①取数指令微程序首地址的形成；

当取数指令存入 IR 后，其操作码 OP(IR)直接送到微地址形成部件，该部件的输出即为取数指令微程序的首地址 P，且将 P 送至 CMAR，记作 OP(IR) →CMAR

②取微指令；

将对应控存 P 地址单元中的微指令读到 CMDR 中，即 CM(CMAR) →CMDR

③产生微操作命令；

由微指令操作控制字段中对应“1”的各位发出控制信号，如 Ad(IR) →MAR，1→R，命令主存读操作数。

④形成下一条微指令的地址；

将此条微指令下地址字段指出的 P+1 送至 CMAR，即 Ad(CMDR) →CMAR

⑤取微指令，即 CM(CMAR) →CMDR；

⑥产生微操作命令；

⋮

以此类推， 直到取出取数指令微程序的最后一条微指令 P+2，并发出微命令,至此即完成了将主存 X 地址单元中的操作数取至累加器 AC 的操作。

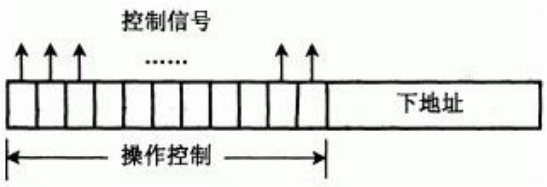
P+2 这条微指令的顺序控制字段为 M， 即表明 CPU 又开始进入下一条机器指令的取指周期，控存又要依次读出取指周期微程序的逐条微指令，发出微命令， 完成将第二条机器指令 ADD Y 从主存取至指令寄存器 IR 中……

（三）微指令的编码方式

微指令的编码方式又叫微指令的控制方式，它是指如何对微指令的控制字段进行编码，以形成控制信号。

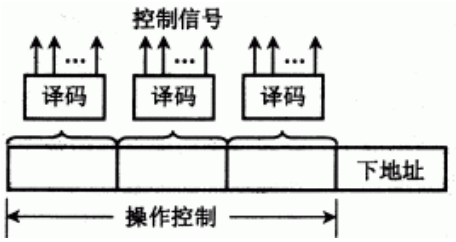
1、直接编码（直接控制）方式

在微指令的的操作控制字段中，每一位代表一个微命令，这种编码方式即为直接编码方式。



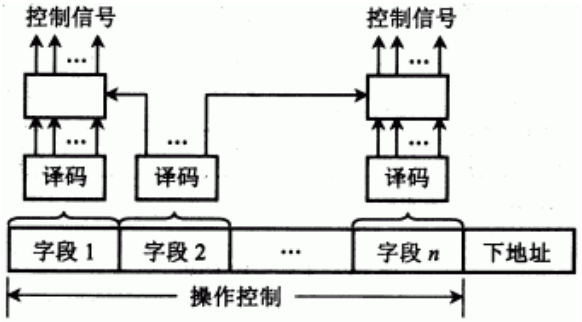
2、字段直接编码方式

这种方式就是将微指令的操作控制字段分成若干段，将一组互斥的微命令放在一个字段内，通过对这个字段译码，便可对应每一个微命令，如下图所示。这种方式因靠字段直接译码发出微命令，故又有显式编码之称。



3、字段间接编码方式

这种方式一个字段的某些微命令还需由另一个字段中的某些微命令来解释，如下图所示。图中字段 1 译码的某些输出受字段 2 译码输出的控制，由于不是靠字段直接译码发出微命令，故称为字段间接编码，又称隐式编码。



4、混合编码

这种方法是把直接编码和字段编码(直接或间接)混合使用，以便能综合考虑微指令的字长、灵活性和执行微程序的速度等方面的要求。

5、其他

微指令中还可设置常数字段，用来提供常数、计数器初值等。常数字段还可以和某些解释位配合，如解释位为 0， 表示该字段提供常数；解释位为 1，表示该字段提供某种命令，使微指令更灵活。

（四）微指令序列地址的形成

1、直接由微指令的下地址字段指出

在微指令的下地址字段直接指出后继微指令的地址。

2、根据机器指令的操作码形成

当机器指令取至指令寄存器后，微指令的地址由操作码经微地址形成部件形成。微地址形成部件实际是一个编码器，其输入为指令操作码，输出就是对应该机器指令微程序的首地址。它可采用 PROM 实现，以指令的操作码作为 PROM 的地址，而相应的存储单元内容就是对应该指令微程序的首地址。

3、增量计数器法

仔细分析发现，在很多情况下，后继微指令的地址是连续的，因此对于顺序地址，微指令可采用增量计数方法，即(CMAR)+1→CMAR 来形成后继微指令的地址。

4、分支转移

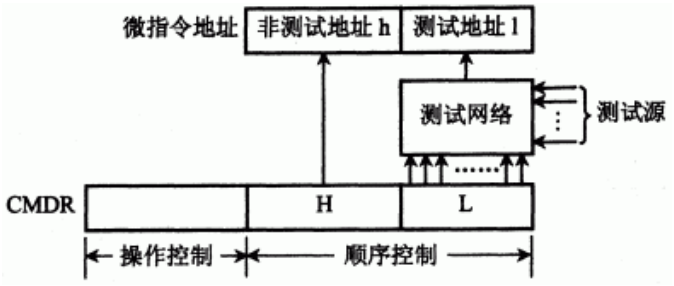
当遇到条件转移指令时，微指令出现了分支，必须根据各种标志来决定下一条微指令的地址。微指令的格式为：

操作控制字段	转移方式	转移地址

其中转移方式是指明判别条件，转移地址是指明转移成功后的去向，若不成功则顺序执行。也有的转移微指令中设两个转移地址，条件满足时选择其中一个转移地址；条件不满足时选择另一个转移地址。

5、通过测试网络形成

微指令的地址还可通过测试网络形成，如下图所示。图中微指令的地址分两部分，高段 h 为非测试地址，由微指令的 H 段地址码直接形成：低段 l 为测试地址，由微指令的 L 段地址码通过测试网络形成。



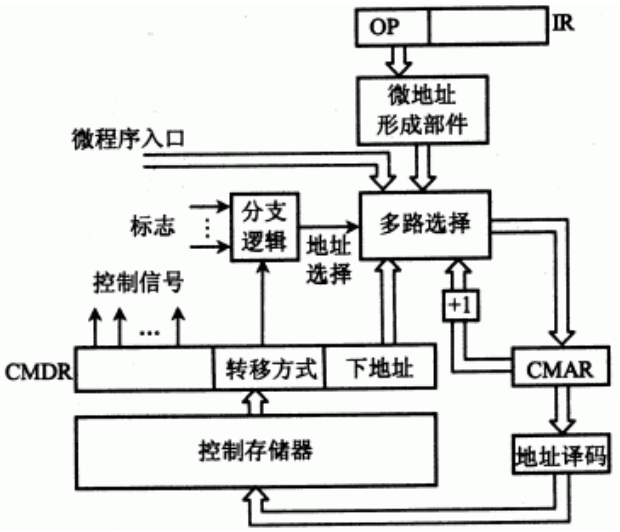
6、微程序入口地址

当电源加电后，第一条微指令的地址可由专门的硬件电路产生，也可由外部直接向 CMAR 输入微指令的地址，这个地址即为**取指周期微程序的入口地址**。

当有中断请求时，若条件满足， CPU 响应中断进入中断周期，此时需中断现行程序，转至对应中断周期的微程序。由于设计控制单元时已安排好中断周期微程序的入口地址，故响应中断时，可由硬件产生中断周期微程序的入口地址。

同理当出现间接寻址时，也可由硬件产生间址周期微程序的入口地址。

综合上述各种方法，可得出形成后继微指令地址的原理图如下图所示。



（五）微指令格式

微指令格式与微指令的编码方式有关，通常分为水平型微指令和垂直型微指令两种。

1、水平型微指令

水平型微指令的特点是一次能定义并执行多个并行操作的微命令。从编码方式看，直接编码、字段直接编码、字段间接编码以及直接和字段混合编码都属水平型微指令。其中直接编码速度最快，字段编码要经过译码，故速度受影响。

2、垂直型微指令

垂直型微指令的特点是采用类似机器指令操作码的方式，在微指令字中，设置微操作码字段，由微操作码规定微指令的功能。通常一条微指令有 1～2 个微命令，控制 1～2 种操作。这种微指令不强调其并行控制功能。

3、两种微指令格式的比较

(1)水平型微指令比垂直型微指令并行操作能力强，效率高，灵活性强。

(2)水平型微指令执行一条机器指令所需的微指令数目少，因此速度比垂直型微指令快。

(3)水平型微指令用较短的微程序结构换取较长的微指令结构，垂直型微指令正相反，它以较长的微程序结构换取较短的微指令结构。

(4)水平型微指令与机器指令差别较大，垂直型微指令与机器指令相似。

（六）静态微程序设计和动态微程序设计

通常指令系统是固定的，对应每一条机器指令的微程序是计算机设计者事先编好的，因此一般微程序无需改变，这种微程序设计技术即称为静态微程序设计，其控存采用 ROM。 前面讲述的内容基本上属于这一类。

如果采用 EPROM 作为控制存储器， 人们可以通过改变微指令和微程序来改变机器的指令系统，这种微程序设计技术称为动态微程序设计。动态微程序设计由于可以根据需要改变微指令和微程序，因此可以在一台机器上实现不同类型的指令系统，有利于仿真。但是这种设计对用户的要求很高，目前难以推广。

（七）毫微程序设计

微程序可看作是解释机器指令的，毫微程序可看作是解释微程序的，而组成毫微程序的毫微指令则是用来解释微指令的。

毫微程序设计采用两级微程序的设计方法。第一级微程序为垂直型微指令，并行功能不强，但有严格的顺序结构，由它确定后继微指令的地址，当需要时可调用第二级。第二级微程序为水平型微指令，具有很强的并行操作能力，但不包含后继微指令的地址。第二级微程序执行完毕后又返回到第一级微程序。两级微程序分别放在两级控制存储器内。

（八）串行微程序控制和并行微程序控制

完成一条微指令也分两个阶段：取微指令和执行微指令。

（1）串行微程序控制

(a) 串行操作

（2）并行微程序控制

第十章 控制单元的设计 关键词汇

1、微程序

将一条机器指令编写成一个微程序，每一个微程序包含若干条微指令，每一条微指令对应一个或几个

微操作命令。然后把这些微程序存到一个控制存储器中，用寻找用户程序机器指令的办法来寻找每个微程序中的微指令。

2、直接编码（直接控制）方式

在微指令的的操作控制字段中，每一位代表一个微命令，这种编码方式即为直接编码方式。

3、字段直接编码方式

这种方式就是将微指令的操作控制字段分成若干段，将一组互斥的微命令放在一个字段内，通过对这个字段译码，便可对应每一个微命令。这种方式因靠字段直接译码发出微命令，故又有显式编码之称。

4、微程序入口地址

当电源加电后，第一条微指令的地址可由专门的硬件电路产生，也可由外部直接向 CMAR 输入微指令的地址，这个地址即为取指周期微程序的入口地址。

第十章 控制单元的设计 FAQ

1、组合逻辑设计步骤

组合逻辑设计控制单元时，首先根据上述微操作的节拍安排，列出微操作命令的操作时间表，然后写出每一个微操作命令(控制信号)的逻辑表达式，最后根据逻辑表达式画出相应的组合逻辑电路图。

2、微程序设计的特点：

微程序设计省去了组合逻辑设计过程中对逻辑表达式的化简步骤，无需考虑逻辑门级数和门的扇入系数，使设计更简便。而且由于控制信号是以二进制代码的形式出现的，因此只要修改微指令的代码，就可改变操作内容，便于调试、修改，甚至增删机器指令，有利于计算机仿真。

3、微指令格式的分类：

微指令格式与微指令的编码方式有关，通常分为水平型微指令和垂直型微指令两种。

1、水平型微指令

水平型微指令的特点是一次能定义并执行多个并行操作的微命令。从编码方式看，直接编码、字段直接编码、字段间接编码以及直接和字段混合编码都属水平型微指令。其中直接编码速度最快，字段编码要经过译码，故速度受影响。

2、垂直型微指令

垂直型微指令的特点是采用类似机器指令操作码的方式，在微指令字中，设置微操作码字段，由微操作码规定微指令的功能。通常一条微指令有 1～2 个微命令，控制 1～2 种操作。这种微指令不强调其并行控制功能。

4、两种微指令格式的比较。

(1)水平型微指令比垂直型微指令并行操作能力强，效率高，灵活性强。

(2)水平型微指令执行一条机器指令所需的微指令数目少，因此速度比垂直型微指令快。

(3)水平型微指令用较短的微程序结构换取较长的微指令结构，垂直型微指令正相反，它以较长的微程序结构换取较短的微指令结构。

(4)水平型微指令与机器指令差别较大，垂直型微指令与机器指令相似。

第十章 控制单元的设计 拓展资源

——微程序设计举例

1．写出对应机器指令的微操作及节拍安排

(1)取指阶段的微操作及节拍安排。取指阶段的微操作基本与组合逻辑控制相同，不同的是指令取至IR后，微程序控制需由操作码形成执行阶段微程序的入口地址。即

T_0 PC→MAR，1→R

T_1 M(MAR) →MDR，(PC)+1→PC

T_2 MDR→IR，OP(IR) →微地址形成部件(编码器)

(2)执行阶段的微操作及节拍安排。执行阶段的微操作由操作码性质而定，同时也需考虑后继微指令地址的形成问题。

·非访存指令

①CLA指令

与组合逻辑控制一样， 该指令在执行阶段只有一个微操作0→AC，只需一个时钟周期T， 故对应一条微指令。该微指令的下地址字段应直接给出取指微程序的入口地址， 而且由下一个T的上升沿将地址打入到CMAR内。这样，对应CLA指令执行阶段的微指令有两条：

T_0 0→AC

T_1 Ad(CMDR) →CMAR，取指微程序入口地址→CMAR

同理可得其余4条非访存指令对应的微操作。

②COM指令

T_0 $\overline{AC}\rightarrow AC$

T_1 Ad(CMDR) →CMAR，取指微程序入口地址→CMAR

③SHR指令

T_0 L(AC)→R(AC)，$AC_0\rightarrow AC_0$

T_1 Ad(CMDR) →CMAR，取指微程序入口地址→CMAR

④CSL指令

T_0 R(AC)→L(AC)，$AC_0\rightarrow AC_0$ (即 $\rho^{-1}(AC)$)

T_1 Ad(CMDR) →CMAR，取指微程序入口地址→CMAR

⑤STP指令

T_0 0→G

T_1 Ad(CMDR) →CMAR，取指微程序入口地址→CMAR

这里由于安排了Ad(CMDR) →CMAR，使再次启动机器时，可直接用已存入CMAR中的取指微程序的入口地址。

·访存指令

①ADD指令

T_0 Ad(IR)→MAR，1→R

T_1 Ad(CMDR) →CMAR

T_2 M(MAR)→MDR

T_3 Ad(CMDR) →CMAR

T_4 (AC)+(MDR) →AC

T_5 Ad(CMDR) →CMAR,取指微程序入口地址→CMAR

②STA指令

T_0 Ad(IR)→MAR，1→W

T_1 Ad(CMDR) →CMAR

T_2 AC→MDR

T_3 Ad(CMDR) →CMAR

T_4 MDR →M（MAR）

T_5 Ad(CMDR) →CMAR,取指微程序入口地址→CMAR

③LDA 指令

T_0 Ad(IR)→MAR，1→R

T_1 Ad(CMDR) →CMAR

T_2 M(MAR)→MDR

T_3 Ad(CMDR) →CMAR

T_4 MDR →AC

T_5 Ad(CMDR) →CMAR,取指微程序入口地址→CMAR

·转移类指令

①JMP 指令

T_0 Ad(IR)→PC

T_1 Ad(CMDR) →CMAR，取指微程序入口地址→CMAR

②BAN 指令

T_0 $A_0 \cdot Ad(IR)+\overline{A_0} \cdot (PC) \rightarrow PC$

T_1 Ad(CMDR) →CMAR，取指微程序入口地址→CMAR

上述全部微操作共 20 个，微指令共 38 条。

2．确定微指令格式

微指令的格式包括微指令的编码方式、后继微指令的地址形成方式和微指令字长等三个方面。

(1)微指令的编码方式。上述微操作数不多，可采用直接编码方式，由微指令控制字段的某一位直接控制一个微操作。

(2)后继微指令的地址形成方式。根据上述分析，可采用由指令的操作码和微指令的下地址字段两种方式形成后继微指令的地址。

(3)微指令字长。微指令由操作控制字段和下地址字段两部分组成。根据直接编码方式，20 个微操作对应 20 位操作控制字段；根据 38 条微指令， 对应 6 位下地址字段。这样，微指令字长至少 26 位。

仔细分析发现,在 38 条微指令中有 19 条微指令是为了控制将后继微指令的地址打入到 CMAR 的操作(其中 18 条微指令地址字段 Ad(CMDR) →CMAR 和 1 条指令操作码 OP(IR) →CMAR，因此实际上是每两个时钟周期才能取出并执行一条微指令。如果能做到每一个时钟周期取出并执行一条微指令，将大大提高微程序控制的速度。