# 目 录

# 第一章 计算机系统概述

## 第一节 计算机的发展和应用

### 一、计算机的发展

1946 年，世界上第一台计算机（ENIAC）诞生于美国宾夕法尼亚大学；此后，计算机硬件技术获得了飞速地发展，主要可分为四个阶段，如表 1-1 所示：

**表 1-1 计算机的发展阶段**

| 阶段 | 时间 | 元器件 |
|---|---|---|
| 第一代 | 1946-1957 年 | 电子管计算机 |
| 第二代 | 1958-1964 年 | 晶体管计算机 |
| 第三代 | 1965-1971 年 | 中小规模集成电路计算机 |
| 第四代 | 1972-至今 | 大规模和超大规模集成电路计算机 |

### 二、计算机的应用

#### （一）科学计算

这是计算机应用最早的领域，利用计算机来完成科学研究和工程技术中提出的数学问题的计算。

#### （二）过程控制

将计算机应用于机械、冶金、纺织、水电、航天等各个领域生产过程的自动控制，减轻劳动强度，提高产品质量和生产效率。

### （三）人工智能

人工智能是计算机模拟人类的智能活动，近年来主要应用于图像识别、语音识别、语言翻译、专家系统、机器人等领域。

### （四）数据处理

用计算机对各种数据进行分析、加工、变换或综合处理。据统计，80%以上的计算机主要用于数据处理。

### （五）计算机辅助应用

1.计算机辅助设计（CAD）

2.计算机辅助制造（CAM）

3.计算机辅助教学（CAI）

4.计算机辅助测试（CAT）

5.计算机辅助工程（CAE）

### （六）网络应用

计算机技术与通信技术的结合构成了计算机网络。目前，网络应用已经涉及人类生活的方方面面，正在改变着整个世界。

### （七）多媒体技术

最初的计算机只能处理文字，由于新技术的运用，计算机可以处理文字、图像、动画、声音等各种数据，这种技术被称为“多媒体技术”。

### （八）电子商务

电子商务通常是指在全球各地广泛的商业贸易活动中，在因特网开放的网络环境下，基于浏览器/服务器应用方式进行各种商贸活动。电子商务主要的分类有： B2B、B2C、C2C、B2G、C2G、O2O 等。

## 三、计算机的发展趋势

### （一）巨型化

巨型化是指计算机的运算速度更高、存储容量更大、功能更强。目前正在研制的巨型计算机其运算速度可达每秒亿亿次级别。

### （二）微型化

微型计算机已进入仪器、仪表、家用电器等小型仪器设备中，同时也作为工业控制过程的心脏，使仪器设备实现“智能化”。随着微电子技术的进一步发展，笔记本型、掌上型等微型计算机必将以更优的性能价格比受到人们的欢迎。

### （三）智能化

计算机人工智能化是未来发展的必然趋势。现代计算机具有强大的功能和运行速度，但与人脑相比，其智能化和逻辑能力仍有待提高。新一代计算机，将可以模拟人的感觉行为和思维过程的机理，进行“看”、“听”、“说”、“想”、“做”，具有逻辑推理、学习与证明的能力。

### （四）网络化

互联网将世界各地的计算机连接在一起，从此进入了互联网时代。

### （五）多媒体化

传统的计算机处理的信息主要是字符和数字。事实上，人们更习惯的是图片、文字、声音等多种形式的多媒体信息。多媒体技术可以集图形、图像、音频、视频、文字为一体，使信息处理的对象和内容更加接近于真实世界。

# 第二节 计算机的特点和分类

## 一、计算机的特点

### （一）运算速度快

计算机可以高速准确地完成各种算术运算，使大量复杂的科学计算问题得以解决。

### （二）运算精度高

科学技术的发展特别是尖端科学技术的发展，需要高度精确的计算。比如计算机控制的导弹之所以能准确地击中预定的目标，是与计算机的精确计算分不开的。一般计算机的计算精度可由千分之几到百万分之几，是任何计算工具所望尘莫及的。

### （三）逻辑运算能力强

计算机不仅能进行精确计算，还具有逻辑运算功能，能对信息进行比较和判断。

### （四）存储容量大

计算机内部的存储器具有记忆特性，可以存储大量的信息，不仅包括各类数据信息，还包括加工这些数据的程序。

### （五）自动化程度高

由于计算机具有存储记忆能力和逻辑判断能力，所以人们可以将预先编好的程序组纳入计算机内存，在程序控制下，计算机可以连续、自动地工作，不需要人的干预。

### （六）性价比高

计算机发展非常迅速，越来越普遍化、大众化。

## 二、计算机的分类

### （一）按数据处理方式

**1.数字式计算机**

数字式电子计算机是当今世界电子计算机行业中的主流，其内部处理的是一种称为符号信号或数字信号的电信号。

**2.模拟式计算机**

模拟式电子计算机问世较早，内部所使用的电信号模拟自然界的实际信号，因而称为模拟电信号。

**3.数字模拟混合式计算机**

混合计算机是取数字、模拟计算机之长，既能高速运算，又便于存储信息。

### （二）按计算机用途

**1.专用计算机**

专用计算机是指为适应某种特殊应用而设计的计算机，一般用在过程控制中，如智能仪表、飞机的自动控制、导弹的导航系统等。

**2.通用计算机**

通用计算机是指为解决各种问题，具有较强的通用性而设计的计算机。该机适用于一般的科学计算、学术研究、工程设计和数据处理等用途，具有较大的适用范围。

### （三）按计算机性能

**1.巨型计算机**

运算速度快、存储容量大，价格相当昂贵，主要用于复杂、尖端的科学研究领域，特别是军事科学计算。

**2.大/中型计算机**

通用性能好、外部设备负载能力强、处理速度快。它有完善的指令系统，丰富的外部设备和功能齐全的软件系统，并允许多个用户同时使用。主要用于科学计算、数据处理或做网络服务器。

3.小型计算机

具有规模较小、结构简单、成本较低、操作简单、易于维护、与外部设备连接容易等特点。

4.微型计算机

以运算器和控制器为核心，加上由大规模集成电路制作的存储器、输入/输出接口和系统总线，构成了体积小、结构紧凑、价格低但又具有一定功能的计算机。

5.单片机

单片机把 CPU、一定容量的存储器和输入/输出接口电路集成到一个芯片上，它是一片特殊的、具有计算机功能的集成电路芯片，一般用作专用机或用来控制高级仪器、家用电器等。

## （四）按计算机使用方式

1.桌面型计算机

桌上型计算机包括 PC 机、工作站和笔记本型计算机，为用户提供良好的计算性能和较低成本的工作环境。桌上型计算机是成本低、应用广的计算机类型。

2.服务器型计算机

服务器型计算机是指在网络环境或具有客户-服务器结构的分布式计算环境中，为客户请求提供服务的节点计算机。

3.嵌入式计算机

嵌入式计算机是将计算机作为一个部件，成为某个设备的一部分，嵌入式计算机成本更低，用途更广。它一般面向特定应用。不同的嵌入式应用有不同的要求，需要根据不同的应用进行专门的开发设计。

# 第三节 计算机的性能指标

## 一、字长

字长以二进制为单位，是 CPU 一次能够处理的二进制数据的位数，它直接关系到计算机的计算精度和运算能力。微机字长一般都以 2 的 n 次方为单位，如 8 位、16 位、32 位、64 位等。

## 二、运算速度

由于计算机执行不同类型指令所需时间不同，所以通常用各类指令的平均执行时间和相应指令的运行比例进行综合计算，作为衡量计算机运行速度的标准。用来衡量运算速度的指标有 MIPS（百万条指令/秒）、MFLOPS（百万次浮点运算/秒）以及 CPI（执行一条指令所需的时钟周期）。

## 三、时钟频率

时钟频率，也称主频，是指 CPU 在单位时间（秒）内发出的脉冲数。通常，时钟频率以兆赫（MHz）和吉赫（GHz）为单位。一般时钟频率越高，其运算速度就越快。

## 四、内存容量

### （一）位

计算机中最小的数据单位是二进制的一个数位，简称为位（bit，比特）。一个二进制位可表示两种状态（0 或 1）。

### （二）字节

为了表示数据中的字符（包括字母、数字以及各种专用符号等），需要用 7 位或 8 位二进制数，人们选定 8 位为一个字节（Byte），通常用 B 表示。字节是计算机中用来表示存储空间的最基本的容量单位。

除用字节为单位表示存储容量外，还可以用千字节（KB）、兆字节（MB）以及吉兆字节（GB）等表示存储容量，它们之间存在下列换算关系：

1B = 8 bit

1KB = $2^{10}$ B =1024 B

1MB = $2^{20}$ B =1024 KB

1GB = $2^{30}$ B = 1024 MB

1TB = $2^{40}$ B = 1024 GB

内存一般以 KB、MB 或 GB 为单位。内存容量反映了内存储器存储数据的能力。存储容量越大，其主机处理数据的范围就越广，运算速度一般也就越快。

# 第四节 计算机系统的组成和结构

## 一、冯·诺依曼计算机

存储程序的概念是由冯·诺依曼于1945年提出的，它奠定了现代计算机的结构基础。尽管过去了几十年，但目前广泛应用的计算机仍然是依据冯·诺依曼提出的结构体系和工作原理来设计制造的，故又统称为“冯·诺依曼计算机”。

### （一）冯·诺依曼思想

①计算机的五大组成

计算机由运算器、控制器、存储器、输入设备和输出设备五大部件组成。

②采用二进制

采用二进制形式存储所有的信息，即计算机内不管是程序还是待处理的数据或是其他信息均为二进制编码形式。

③存储程序

这是冯·诺依曼思想的核心。将事先编好的程序存入计算机中，计算机按照这些程序自动运行，这是计算机自动连续工作的基础。

### （二）冯·诺依曼计算机的工作原理

①存储程序：将要执行的程序和数据事先编成二进制形式的编码存入主存储器中。

②程序控制：程序自动地、连续地从主存储器中依次取出指令并执行，直到获得所要求的结果为止。

典型的冯·诺依曼计算机是以运算器为中心的，而现代的计算机已经转为以存储器为中心。

## 二、计算机系统的构成

一个计算机系统包括硬件系统和软件系统两大部分。硬件是由各种电介质、磁介质及机械的器件组成的物理实体，包括运算器、控制器、存储器、输入设备和输出设

备等五个基本部分。软件则是程序和有关文档的总称，通常存放在计算机的主存或辅存内。

在一个具体的计算机系统中，硬件、软件是紧密相关、缺一不可的，但对某一具体功能来说，既可以用硬件实现，也可以用软件实现，这是硬件、软件在逻辑功能上的等效。硬件、软件在逻辑功能上的等效是指任何由硬件实现的操作，在原理上均可用软件模拟来实现；同样，任何由软件实现的操作，在原理上都可由硬件来实现。因此在设计一个计算机系统时，须根据设计要求、现有技术与器件条件，首先确定哪些功能直接由硬件实现，哪些功能通过软件实现，这是硬件和软件的功能分配。

## （一）硬件系统

计算机的硬件系统由主机系统和外部设备两部分组成，把运算器、控制器与内存储器统称为计算机主机，而把外存储器、输入设备、输出设备称为计算机的外部设备，如图 1-1 所示。

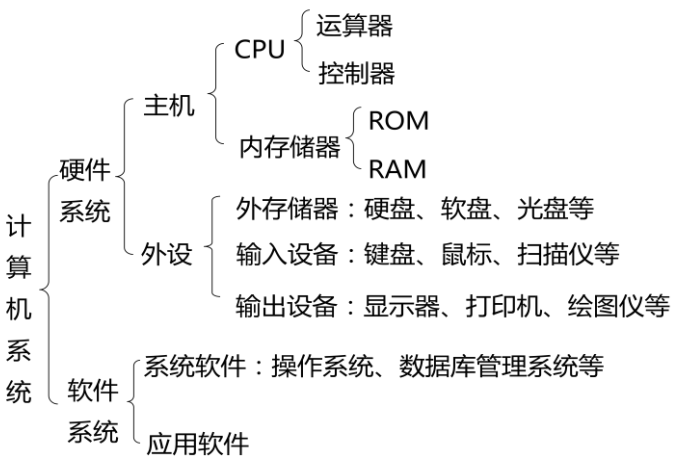



**图 1-1 计算机系统的组成**

### 1.主机

运算器用于对数据的加工处理，完成算术运算和逻辑运算。算术运算是指按照算术运算规则进行的运算，如加、减、乘、除及其复合运算。逻辑运算为非算术性的运算，如与、或、非、异或等。运算器的核心是算术逻辑部件(ALU，Arithmetic Logical Unit)。运算器中还设有若干寄存器，用于暂存操作数据和中间结果。由于这些寄存器

往往兼备多种用途，如用作累加器、变址寄存器、基址寄存器等，所以通常称为通用寄存器。

控制器是整个计算机的指挥中心，它按照事先安排好的步骤，控制计算机各个部件有条不紊地自动工作。通常把运算器和控制器统称为 CPU。

计算机具有超强的记忆能力，是因为计算机中有存储器部件。CPU 能够直接访问的是内存储器（也称内存，主存）。内存主要由两部分组成：随机存储器和只读存储器。

**（1）随机存储器**

随机存储器（RAM）可以随时读写，而且速度很快，通常作为操作系统或其他正在运行中的程序的临时数据存储媒介。

**（2）只读存储器**

只读存储器（ROM）所存数据，一般是装入整机前事先写好的，计算机工作过程中只能读出，而不像随机存储器那样能快速、方便地加以改写。ROM 所存数据稳定，断电后也不会改变，常用于存储各种固定程序和数据。

**2.外部设备**

外存储器又称为外存、辅存，是计算机系统中除内存储器外，以计算机能接受的形式存储信息的媒体，如硬盘、软盘、光盘、U 盘等。其特点是能长期保存数据，而且价格便宜，存储量大。

输入设备的主要功能是将程序和数据以机器所能识别和接受的信息形式输入到计算机内。最常用也是最基本的输入设备是键盘、鼠标、扫描仪、摄像机等。

输出设备的任务是将计算机处理的结果以人们所能接受的信息形式或其他系统所要求的信息形式输出。最常用且最基本的输出设备是显示器、打印机、绘图仪等。

## （二）软件系统

计算机软件系统包括系统软件和应用软件两大类。

系统软件是一组保证计算机系统高效、正确运行的基础软件，通常作为系统资源提供给用户使用。主要有以下几类：操作系统，语言处理程序如编译器和解释器，数据库管理系统，各种服务程序如装入程序、调试程序、诊断程序等。

应用软件是指用户为解决某个应用领域中的各类问题而编制的程序，如各种科学计算类程序、数据统计与处理程序、情报检索程序、生产过程控制程序等。

### （三）计算机系统的层次结构

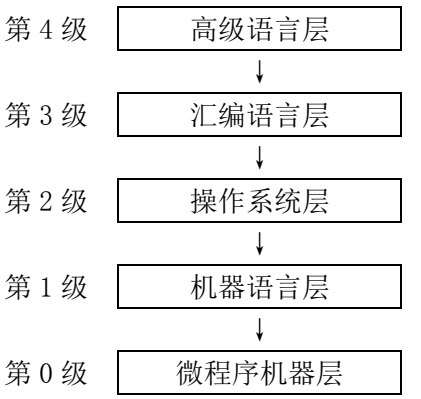



**图 1-2 计算机系统的层次结构**

计算机系统的层次结构如图 1-2 所示，一共分为五级：

第 0 级是微程序机器层，这是一个实在的硬件层，它由机器硬件直接执行微指令。

第 1 级是机器语言层，这一层由微程序解释机器指令系统。

第 2 级是操作系统层，它由操作系统程序实现。操作系统由机器指令和广义指令组成的，这些广义指令是为扩展机器功能而设置的，是由操作系统定义和解释的软件指令。

第 3 级是汇编语言层，它为用户提供一种符号形式语言，借此可编写汇编语言源程序，这一层由汇编程序支持和执行。

第 4 级是高级语言层，它是面向用户的，为方便用户编写应用程序而设置的。该层由各种高级语言编译程序支持和执行。

从计算机系统的多级层次结构来看，可以将硬件研究的主要对象归结为机器语言层和微程序机器层。软件的研究对象主要是操作系统级以上的各级虚拟机。软硬件交界界面的划分并不是一成不变的，随着超大规模集成电路技术的不断发展，一部分软件功能将由硬件来实现，如目前操作系统已实现了部分固化（把软件永久地存于只读

存储器中），称为固件。可以认为固件是一种介于传统的软件和硬件之间的实体，就它的功能来说，类似软件，但从形态来说，又类似硬件。

# 第二章 数据的表示和运算

## 第一节 进制转换

### 一、非十进制数转换成十进制

#### （一）方法

每位上的数码×基的位次方，然后求和。

#### （二）例题

1.二进制转化成十进制

（1101.101）$_2$=1×$2^3$+1×$2^2$+0×$2^1$+1×$2^0$+1×$2^{-1}$+0×$2^{-2}$+1×$2^{-3}$=8+4+0+1+0.5+0+0.125=（13.625）$_{10}$

2.八进制转化成十进制

（456.124）$_8$=4×$8^2$+5×$8^1$+6×$8^0$+1×$8^{-1}$+2×$8^{-2}$+4×$8^{-3}$=256+40+6+0.125+0.03125+0.0078125=（302.1640625）$_{10}$

3.十六进制转换成十进制

（32CF.48）$_{16}$=3×$16^3$+2×$16^2$+C×$16^1$+F×$16^0$+4×$16^{-1}$+8×$16^{-2}$=12288+512+192+15+0.25+0.03125=（13007.28125）$_{10}$

### 二、十进制数转换成非十进制数

#### （一）方法

整数部分：除 N 取余数，直至商为 0，余数倒输出。

小数部分：乘 N 取整数，直至积为 0（或满足精度），整数正输出。

（N 为进制数，N 可取 2、8、16。）

## （二）例题

### 1.十进制数转换成二进制数

将十进制数（37.75）$_{10}$转换为二进制数。

整数部分 37 转换如下：

| | | 余数 |
|---|---|---|
| 2 | 37 | |
| 2 | 18 | 1 |
| 2 | 9 | 0 |
| 2 | 4 | 1 |
| 2 | 2 | 0 |
| 2 | 1 | 0 |
| | 0 | 1 |

小数部分 0.75 转换如下：

| | 整数 |
|---|---|
| 0.75×2＝1.5 | 1 |
| 0.5×2＝1.0 | 1 |

即（37.75）$_{10}$=（100101.11）$_{2}$。

### 2.十进制数转换成八进制数

将十进制数（1725.6875）$_{10}$ 转换成八进制数。

整数部分 1725 转换如下：

| | | 余数 |
|---|---|---|
| 8 | 1725 | |
| 8 | 215 | 5 |
| 8 | 26 | 7 |
| 8 | 3 | 2 |
| | 0 | 3 |

小数部分 0.6875 转换如下：

整数

0.6875×8＝5.5　　5

0.5×8＝4.0　　4

即（1725.6875）$_{10}$=（3275.54）$_{8}$

③十进制数转换成十六进制数

将十进制数（12347.671875）$_{10}$ 转换为十六进制数。

整数部分 12347 转换过程如下：

| | | 余数 |
|---|---|---|
| 16 | 12347 | |
| 16 | 771 | ~~11~~ B |
| 16 | 48 | 3 |
| 16 | 3 | 0 |
| | 0 | 3 |

小数部分 0.671875 转换如下：

整数

0.671875×16＝10.75　　~~10~~ A

0.75×16＝12.00　　~~12~~ C

即（12347.671875）$_{10}$=（303B.AC）$_{16}$

## 三、二/八/十六进制数的互相转换

### （一）方法

由于一位八/十六进制数相当于三/四位二进制数，因此，要将八/十六进制数转换成二进制数时，只需以小数点为界，向左或向右每一位八/十六进制数用相应的三/四位二进制数取代即可。如果不足三/四位，可用零补足。反之，二进制数转换成相应的八/十六进制数，只是上述方法的逆过程，即以小数点为界，向左或向右，每三/四位二进制数用相应的一位八/十六进制数取代即可。

## （二）例题

**1.八进制数转换成二进制数**

将八进制数（624.31）$_8$ 转换成二进制数。

| 6 | 2 | 4 | . | 3 | 1 |
|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | | ↓ | ↓ |
| 110 | 010 | 100 | . | 011 | 001 |

即（624.31）$_8$=（110010100. 011001）$_2$。

**2.二进制数转换成八进制数**

将二进制数（11101.00101011）$_2$ 转换成八进制数。

| 011 | 101 | . | 001 | 010 | 110 |
|---|---|---|---|---|---|
| ↓ | ↓ | | ↓ | ↓ | ↓ |
| 3 | 5 | . | 1 | 2 | 6 |

即（11101.00101011）$_2$=（35.126）$_8$。

**3.十六进制数转换成二进制数**

将十六进制数（1AC0.6D）$_{16}$ 转换成相应的二进制数。

| 1 | A | C | 0 | . | 6 | D |
|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ | | ↓ | ↓ |
| 0001 | 1010 | 1100 | 0000 | . | 0110 | 1101 |

即（lAC0.6D）$_{16}$=（1101011000000.01101101）$_2$。

**4.二进制数转换成十六进制数**

将二进制数（10111100101.00011001101）$_2$ 转换成相应的十六进制数。

| 0101 | 1110 | 0101 | . | 0001 | 1001 | 1010 |
|---|---|---|---|---|---|---|
| ↓ | ↓ | ↓ | | ↓ | ↓ | ↓ |
| 5 | E | 5 | . | 1 | 9 | A |

即（10111100101.00011001101）$_2$=（5E5.19A）$_{16}$。

# 第二节 数值型数据的表示

## 一、机器数

无符号数，是指整个机器字长的全部二进制位均表示数值位（没有符号位），相当于数的绝对值。例如：0101 表示无符号数 5，1100 表示无符号数 12。然而，大量用到的数据还是带符号数，即正数和负数。在日常生活中用“＋”、“－”加绝对值来表示数值的大小，用这种形式表示的数值在计算机中称为“真值”。

对于数的符号 “＋”和“－”，计算机是无法识别的，需要把符号数码化。通常，约定二进制数的最高位为符号位，“0”表示正号，“1”表示负号。这种在计算机中使用的表示数的形式称为机器数，常见的机器数有原码、反码、补码等。

### （一）原码

原码表示法是机器数的一种简单的表示法。其符号位用 0 表示正号，用 1 表示负号，数值一般用二进制形式表示。设有一数用 X 表示真值，则原码表示可记作$[X]_{原}$，例如：

X1=＋1010110

X2=－1001010

其原码记作：

$[X1]_{原}$=[＋1010110]$_{原}$=01010110

$[X2]_{原}$=[－1001010]$_{原}$=11001010

在原码表示法中，对 0 有两种表示形式：

$[+0]_{原}$=00000000

$[-0]_{原}$=10000000

### （二）反码

机器数的反码可由原码得到。如果机器数是正数，则该机器数的反码与原码一样；如果机器数是负数，则该机器数的反码是对它的原码（符号位除外）各位取反而得到的。假设有一数用 X 表示真值，那么 X 的反码表示记作$[X]_{反}$，例如

X1=＋1010110

X2=－1001010

那么

$[X1]_{原}$=01010110

$[X1]_{反}$=$[X1]_{原}$=01010110

$[X2]_{原}$=11001010

$[X2]_{反}$=10110101

在反码表示法中，对 0 也有两种表示形式：

$[+0]_{反}$=00000000

$[-0]_{反}$=11111111

反码通常作为求补过程的中间形式，即在一个负数的反码的末位上加 1，就得到该负数的补码。

## （三）补码

机器数的补码可由原码得到。如果机器数是正数，则该机器数的补码与原码一样；如果机器数是负数，则该机器数的补码是对它的原码（除符号位外）各位取反，并在末位加 1 而得到的。设有一数用 X 表示真值，则 X 的补码表示记作$[X]_{补}$，例如：

X1=＋1010110

X2= –1001010

那么

$[X1]_{原}$=01010110

$[X1]_{补}$= $[X1]_{原}$=01010110

$[X2]_{原}$=11001010

$[X2]_{补}$=10110101+1=10110110

在补码表示法中，0 只有一种表示形式：

$[+0]_{补}$=00000000

$[-0]_{补}$=11111111＋1=00000000

所以有$[+0]_{补}$=$[-0]_{补}$=00000000。

## （四）补码的加减法运算

### 1.补码加法

两个补码表示的数相加，符号位参加运算，且两数和的补码等于两数补码之和，即：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

### 2.补码减法

根据补码加法公式可推出：

$$[X-Y]_{补}=[X+(-Y)]_{补}=[X]_{补}+[-Y]_{补}$$

已知$[Y]_{补}$求$[-Y]_{补}$的方法是：将$[Y]_{补}$连同符号位一起求反，末尾加“1”。

$[-Y]_{补}$被称为$[Y]_{补}$的机器负数，由$[Y]_{补}$求$[-Y]_{补}$的过程称为对$[Y]_{补}$变补，表示为：

$$[-Y]_{补}=[[Y]_{补}]_{变补}$$

要注意将“某数的补码表示”与“变补”这两个概念区分开来。一个负数由原码表示转换成补码表示时，符号位是不变的，仅对数值位的各位变反，末尾加“1”。而变补则不论这个数的真值是正是负，一律连同符号位一起变反，末尾加“1”。 $[Y]_{补}$表示的真值如果是正数，则变补后$[-Y]_{补}$所表示真值变为负数，反之亦然。例如：

Y=－0.0110，$[Y]_{补}$=1.1010，$[-Y]_{补}$=0.0110；

Y= 0.0110，$[Y]_{补}$=0.0110，$[-Y]_{补}$=1.1010。

### 3.符号扩展

在计算机算术运算中，有时必须将采用给定位数表示的数转换成具有更多位数的某种表示形式。如某个程序需要将一个 8 位数与另外一个 32 位数相加。要想得到正确的结果，在将 8 位数与 32 位数相加之前，必须将 8 位数转换成 32 位数形式，这被称为“符号扩展”。

对于补码，符号扩展方法是：原有符号位保持不变，若为正数则所有附加位都用 0 进行填充；若为负数则所有附加位都用 1 进行填充。该方法也可以理解为是用符号位来填充附加的高位。

如将用 8 位二进制补码表示的十进制数−121，扩展成 16 位二进制补码，结果用十六进制表示。

【解析】：十进制数−121 的 8 位二进制补码表示为 10000111，扩展成 16 位二进制补码，符号扩展表示为 1111111110000111= FF87H。

4.补码溢出的产生

在补码加减运算中，有时会遇到这样的情况：两个正数相加，而结果的符号位却为 1（结果为负）；两个负数相加，而结果的符号位却为 0（结果为正）。发生这种错误的原因在于两数相加之和的数值已超过了机器允许的表示范围。字长为 n+1 位的定点整数（其中一位为符号位），采用补码表示，当运算结果大于 $2^{n-1}$ 或小于 $-2^n$ 时，就产生溢出。

设参加运算的两数为 X、Y，做加法运算：

①若 X、Y 异号，不会溢出；

②若 X、Y 同号，运算结果为正且大于所能表示的最大正数或运算结果为负且小于所能表示的最小负数时，产生溢出。将两正数相加产生的溢出称为正溢；反之，两负数相加产生的溢出称为负溢。

5.补码溢出的检测

设被操作数为$[X]_{补}=X_sX_1X_2\cdots X_n$，操作数为$[Y]_{补}=Y_sY_1Y_2\cdots Y_n$，其和（差）为$[S]_{补}=S_sS_1S_2\cdots S_n$，则判断溢出的方法有以下三种：

（1）采用一个符号位

两正数相加，结果为负表明产生正溢；两负数相加，结果为正表明产生负溢。因此可得出采用一个符号位检测溢出的方法：

当 $X_s=Y_s=0$，$S_s=1$ 时，产生正溢；

当 $X_s=Y_s=1$，$S_s=0$ 时，产生负溢。

（2）采用进位位

两数运算时，产生的进位为：$C_sC_1C_2\cdots C_n$，其中：$C_s$ 为符号位产生的进位，$C_1$ 为最高数值位产生的进位。

两正数相加，当最高有效位产生进位（$C_1=1$）而符号位不产生进位（$C_s=0$）时，发生正溢。

两负数相加，当最高有效位没有进位（$C_1=0$）而符号位产生进位（$C_s=1$）时，发生负溢。

（3）采用变形补码（双符号位补码）

在双符号位的情况下，把左边的符号位 $S_{s1}$ 叫做真符，因为它代表了该数真正的符号，两个符号位都作为数的一部分参加运算。这种编码又称为变形补码。双符号位的含义如下：

$S_{s1}S_{s2}$=00，结果为正数，无溢出；

$S_{s1}S_{s2}$=01，结果正溢；

$S_{s1}S_{s2}$=10，结果负溢；

$S_{s1}S_{s2}$=11，结果为负数，无溢出。

## 二、定点数和浮点数表示

在计算机中，一般用若干个二进制位表示一个数或一条指令，把它们作为一个整体来处理、存储和传送。这种作为一个整体来处理的二进制位串，称为计算机字。表示数据的字称为数据字，表示指令的字称为指令字。

计算机是以字为单位进行处理、存储和传送的，所以运算器中的加法器、累加器以及其他一些寄存器，都选择与字长相同的位数。字长一定，则计算机数据字所能表示的数的范围也就确定了。例如，使用 8 位字长的计算机，它可以表示无符号整数的最大值是（255）$_{10}$=（11111111）$_2$。运算时，若数值超出机器数所能表示的范围，就会停止运算和处理，这种现象称为溢出。

### （一）定点数

计算机中运算的数有整数也有小数，对于小数点的位置的确定，通常有两种约定：一种是规定小数点的位置固定不变，这时的机器数称为定点数；另一种是小数点的位置可以浮动，这时的机器数称为浮点数。微型机多使用定点数。

数的定点表示是指数的小数点位置是固定不变的。小数点位置可以固定在符号位和第一数值位之间，这时表示一个纯小数；如果把小数点位置固定在数值位的最后，这时表示一个纯整数。

## （二）浮点数

浮点表示法就是小数点在数中的位置是浮动的。在以数值计算为主要任务的计算机中，由于定点表示法所能表示的数的范围太窄，不能满足计算问题的需要，因此就要采用浮点表示法。在同样字长的情况下，浮点表示法能表示的数的范围扩大了。

计算机中的浮点表示法包括两个部分：一部分是阶码（表示指数，记作 E）；另一部分是尾数（表示有效数字，记作 M）。设任意一个数 N 可以表示为

$$N=M\times 2^{E}$$

其中，2 为基数；E 为阶码；M 为尾数。浮点数在机器中的表示方法如下：

| 阶符 | E | 数符 | M |
|---|---|---|---|
| | 阶码部分 | | 尾数部分 |

由尾数部分隐含的小数点位置可知，尾数总是小于 1 的数字，它给出该浮点数的有效数字。尾数部分的符号位确定该浮点数的正负。阶码给出的总是整数，它确定小数点浮动的位数。

浮点数表示法对尾数有如下规定：

$$0.5 \leqslant M < 1$$

即要求尾数中第一位数不为零，这样的浮点数称为规格化数。

在浮点数表示和运算中，当一个数的阶码大于机器所能表示的最大阶码时，产生“上溢”，如图 2-1 所示。上溢时机器一般不再继续运算而转入“溢出”处理。当一个数的阶码小于机器所能表示的最小阶码时，产生“下溢”，此时溢出的数绝对值很小，通常将尾数各位置为 0，按机器零来处理，此时计算机可以继续运行。

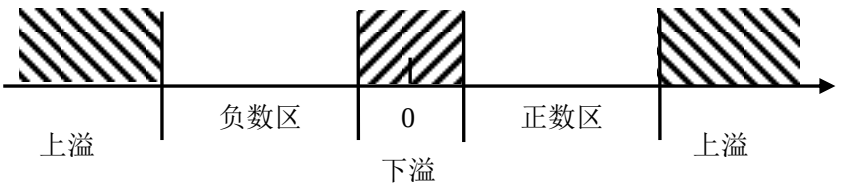



图 2-1 溢出示意图

## 三、IEEE754 标准

### （一）标准格式

IEEE754 标准中 32 位浮点数的标准格式为：

| S（1 位） | E（8 位） | M（23 位） |
|---|---|---|
| 符号位 | 阶码 | 尾数 |

其对应真值是：

$$x=(-1)^S\times(1.M)\times 2^e,\quad e=(E)_{10}-127$$

这里 $M$ 是尾数的二进制表示，$(E)_{10}$ 为阶码的十进制表示，其范围是 1～254。

当阶码 $E$ 和尾数 $M$ 取特殊值时，对应的 $x$ 值如表 2-1 所示。

表 2-1 特殊的 E 和 M 所对应的 x 值

| $(E)_{10}$ | $M$ | $x$ |
|---|---|---|
| 0 | 0 | 0 |
| 255 | 0 | 无穷大 |
| 255 | 非 0 | NaN（无效运算） |

### （二）习题举例

假定采用 IEEE754 标准中的单精度浮点数格式表示一个数为 45100000H，则该数的值为（　　）。

A. $(+1.125)_{10}\times 2^{10}$　　B. $(+1.125)_{10}\times 2^{11}$

C. $(+0.125)_{10}\times 2^{11}$　　D. $(+0.125)_{10}\times 2^{11}$

【解析】：$(45100000)_{16}=(0100\ 0101\ 0001\ 0000\ 0000\ 0000\ 0000\ 0000)_2$

易知 $S$=0，$E$=10001010，$e=(E)_{10}-127=138-127=11$，$1.M=(1.001)_2$，

故 $x=(+1.001)_2\times 2^{11}=(+1.125)_{10}\times 2^{11}$，故选 B。

# 第三节 浮点加减法运算

浮点加减运算需要经过对阶、尾数求和、规格化、舍入和溢出判断五个步骤。

## 一、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。为此，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 l，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。

## 二、尾数求和

将对阶后的两个尾数按定点加（减）运算规则进行运算。

## 三、规格化

为增加有效数字的位数，提高运算精度，必须将求和（差）后的尾数规格化。当尾数用二进制表示时，尾数 M 的规格化形式为

$$1/2 \leqslant |M| < 1$$

如果采用双符号位的补码，则显然对于正数而言，M 的补码形式为 00.1xx...x，对于负数，其补码形式为 11.0xx...x。它们的共同点为符号位与小数点后的第一位不等，这两种形式的数就是规格化的数。这里负数有两种情况需要注意：一是－1 视为规格化的数；二是规定－1/2 不是规格化的数。

若符号位与小数点后的第一位相等，则需规格化。规格化又分左规和右规两种。

### （一）左规

当尾数出现 00.0xx...x 或 11.1xx...x 时，需左规。左规时尾数左移一位，阶码减 1，直到符号位与小数点后的第一位不等为止。

### （二）右规

当尾数出现 01．xx...x 或 10．xx...x 时，表示溢出，这在定点加减运算中是不允许的，但在浮点运算中，它表明尾数求和结果的绝对值大于 1，向左破坏了规格化，此时将尾数运算结果右移以实现规格化表示，称为向右规格化。右规时尾数右移一位，阶码加 l。

## 四、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，影响精度。为提高精度，要考虑尾数右移时丢失的数值位，为此可用舍入法来提高尾数的精度。常用的舍入方法有以下两种：

### （一）“0 舍 1 入”法

“0 舍 1 入”法类似于十进制数运算中的“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；反之则在尾数的末位加 l。

### （二）“恒置 1”法

尾数右移时，只要数位被移掉，都在尾数末位恒置“1”。

## 五、溢出判断

浮点数的溢出是以其阶码溢出表现出来的。在加减运算真正结束前，要检查是否产生了溢出：若阶码正常，加减运算正常结束；若阶码下溢，要置运算结果为浮点形式的机器零；若阶码上溢，则置溢出标志。

# 第四节 逻辑运算和移位操作

运算器除了要完成数值数据的算术运算外，还要完成逻辑运算和移位运算。

## 一、逻辑运算

计算机中的逻辑运算包括与、或、非、异或等运算。由于逻辑数是非数值数据，其每一位的“0”、“l”仅用于表示逻辑上的“真”与“假”，因此逻辑运算的特点是：按位运算，运算结果的各位之间互不影响，不存在进位、借位、溢出等问题。

### （一）“与”运算

“与”运算又称逻辑乘，用符号“∧”、或“×”、或 AND 表示（一假皆假）

### （二）“或”运算

“或”运算又称逻辑加，用符号“＋”、或“∨”、或 OR 表示（一真皆真）。

### （三）“非”运算

“非”运算又称“求反”，一般在变量上加横线、或加 NOT 表示（取其反值）。

### （四）“异或”运算

“异或”运算又称“按位加”，一般用符号⊕表示（相异为真）。

## 二、移位运算

移位运算称为移位操作，对计算机来说有很大的实用价值。如当某计算机没有乘（除）法运算线路时，可以采用移位和加法相结合，以实现乘（除）运算。

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。对空出的空位应该添补 0 还是 l，与机器数采用有符号数还是无符号数有关。对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。

算术移位的规则是：

首先，不论是正数还是负数，移位后其符号位均不变；

其次，对于正数，移位后出现的空位均添 0；

对于负数，其空位的添补规则也不同：

①原码时，其空位均添 0；

②反码时，其空位均添 1；

③补码时，左移其空位添 0，右移其空位添 1。

逻辑移位的规则是：

逻辑左移时，高位移丢，低位添 0；

逻辑右移时，低位移丢，高位添 0。

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。

# 第五节 非数值型数据的表示

## 一、ASCII 码

美国信息交换标准代码（American Strandard Code for Information Interchage，ASCII）是一种西文机内码，是计算机中使用最广泛的字符编码，已经作为国际通用的信息交换标准代码。ASCII 码包括 0～9 十个数字，大小写英文字母和专用符号等 95 种可打印字符，以及 33 种控制字符（如回车、换行等），通常采用一个字节编码，由 7 位二进制编码组成，字节的最高位一般规定为 0，或用作校验码，可表示 128 个不同的字符。

## 二、汉字的编码

### （一）国标码

《信息交换用汉字编码字符集·基本集》是我国于 1980 年制定的国家标准 GB2312—80，简称国标码，是国家规定的用于汉字信息交换使用的代码的依据。GB2312—80 中规定了信息交换用的 6763 个汉字和 682 个非汉字图形符号（包括外文字母、数字和符号）的代码。国标码用两个字节表示一个汉字，每个字节只使用低 7 位，最高位为 0。

### （二）汉字机内码

汉字的机内码是供计算机系统内部进行存储、加工处理、传输统一使用的代码，又称为汉字内部码或汉字内码。汉字机内码是将国标码的两个字节的最高位分别置为 1 而得到的。其最大优点是机内码表示简单，同时也解决了中西文机内码存在的二义性的问题。

### （三）汉字输入码

汉字输入码是为了将汉字通过键盘输入计算机而设计的代码。汉字输入编码方案很多，其表示形式大多用字母、数字或符号。输入码的长度也不同，多数为四个字节。综合起来可分为流水码、音码、形码和音形码几大类。

### （四）汉字字形码

汉字字形码是汉字字库中存储的汉字字形的数字化信息，用于汉字的显示和打印。目前汉字字形的产生方式大多是数字式，即以点阵方式形成汉字。因此，汉字字形码主要是指汉字字形点阵的代码，主要有 16×16 点阵、32×32 点阵、256×256 点阵等。

## 三、十进制数的编码

人们习惯于使用十进制数，而计算机内部多采用二进制数表示和处理数值数据，因此在计算机输入和输出数据时，就要进行由十进制到二进制和从二进制到十进制的转换处理，显然，这项事务性工作如果由人工来完成，势必造成大量时间浪费。因此，必须采用一种编码的方法，由计算机自己来完成这种识别和转换工作。

人们通常采用把十进制数的每一位分别写成二进制数形式的编码，称为二-十进制编码或 BCD 编码。BCD 编码方法很多，通常采用的是 8421 编码。其方法是用四位二进制数表示一位十进制数，自左至右每一位对应的位权分别是 8，4，2，1。值得注意的是，四位二进制数有 0000～1111 十六种状态，这里只取了 0000～1001 十种状态，而 1010～1111 六种状态在这种编码中没有意义。如十进制数 864，其 BCD 进制编码为 100001100100。

| 8 | 6 | 4 |
|---|---|---|
| ↓ | ↓ | ↓ |
| 1000 | 0110 | 0100 |

# 第六节 数据校验码

## 一、奇偶校验码

为了校验编码的正确性，在被传送的 n 位有效信息代码上增加一位校验位，并使其配置后的 n+1 位代码中“1”的个数为奇数，则称其为奇校验；若配置后“1”的个数为偶数，则称其为偶校验。对于奇校验码而言，倘若传送过程中“1”的个数不为奇数，则表明传送出错，可见，奇校验码具有检错能力。同理，偶校验码也具有检错能力。

## 二、循环冗余校验码

循环冗余校验码（CRC）在磁介质存储器和计算机之间通信方面有广泛应用。CRC 码是基于模 2 运算建立的校验码，将待编码的有效信息左移若干位后，用另一个约定的多项式去除，所产生的余数就是检验位。有效信息和检验位相拼接就构成了 CRC 码。

## 三、海明校验码

海明码是目前广泛采用的一种有效的校验码，其实现原理是：在有效信息位中加入几个校验位形成海明码，当某一位出错后，就会引起有关的几个校验位的值发生变化，因此海明码不但可以发现错误，还能指出错误的位置，为自动纠错提供了依据。

# 第三章 指令和总线

## 第一节 指令系统

### 一、指令系统概述

#### （一）指令系统组成

计算机的程序是由一系列的机器指令组成的。

指令就是要计算机执行某种操作的命令。从计算机组成的层次结构来说，计算机的指令有微指令、机器指令和宏指令之分，微指令是微程序级的命令，它属于硬件；宏指令是由若干条机器指令组成的软件指令，它属于软件；而机器指令则介于微指令与宏指令之间，是硬件和软件的交界面，通常简称为指令，每条指令可完成一个独立的算术运算或逻辑运算操作。

一台计算机中所有机器指令的集合，称为这台计算机的指令系统。指令系统是表征一台计算机性能的重要因素，它的格式与功能不仅直接影响到机器的硬件结构，而且也直接影响到系统软件，影响到机器的适用范围。

指令系统一般由两种类型的指令组成：

1.非特权指令

主要供用户使用，又可分为功能性指令和非功能性指令两种。功能性指令主要包括算术逻辑指令、数据传送指令、浮点运算指令、字符串指令等；非功能性指令主要包括转移指令、控制指令等。

2.特权指令

主要供系统程序员使用，一般不允许用户使用。其中包括 I/O 指令、停机等待指令、存储管理及保护指令、控制系统状态指令、诊断指令等。若用户希望使用这类指令，必须先通过访管指令调用操作系统，再由操作系统控制和执行这些特权指令。

性能较好的指令系统一般都满足以下几个原则：

1.完备性

完备性指的是指令系统应功能齐全，给用户带来方便。用户的所有操作均可通过指令或指令串（程序或软件）实现。因此，为提高效率，指令系统应将用户常用的、简单的操作通过指令（硬件）直接实现，而复杂的操作则通过指令串（程序或软件）实现。

2.规整性

规整性指的是指令系统的正交性、均匀性、对称性。

正交性表示指令格式中各个互不相关的字段之间，在编码时应相互独立、互不相关，以方便译码和执行，如指令字可分成操作码和地址码两部分。

均匀性表示指令实现的操作与处理的数据类型无关，以方便数据类型扩展，如同一指令可支持对字节、字、双字整数的运算等。

对称性表示对不同指令中相同或相似的操作数具有相同的规定，以减少配件实现的复杂性，如对不同指令中的存储单元地址所占位数、寄存器号所占位数应具有相同的规定。

3.兼容性

兼容性指不同机种之间具有相同的基本结构和共同的基本指令集，目的是给软件资源的重复利用带来方便。指令系统的兼容性主要指向前兼容。对系列机而言，因为不同机种具有相同的基本指令集，因而系列机中各机种上的指令系统是兼容的，所有已编的软件基本可以通用。

4.可扩充性

可扩充性指指令系统中要保留一定的指令字空间，以便在需要时进行指令系统的功能扩充。因为计算机应用和硬件是不断发展的，原来不常用的、复杂的操作可能会变成常用的、基本的操作，需要添加到指令系统中。

## （二）指令分类

按照功能类型的不同，指令可分为数据传送类指令、算术运算类指令、逻辑运算类指令、程序控制类指令、输入输出类指令、字符串类指令、系统控制类指令和其他指令。

1.数据传送类指令

这类指令主要完成将数据从一个地方传送到另一个地方的功能，一般可按字节、字传送，特殊情况下可以位为单位进行传送。

2.算术运算类相令

这类指令主要实现对数据进行算术运算功能，主要包括加、减、乘、除指令，求反、求补指令，算术移位、算术比较指令等。

3.逻辑运算类相令

这类指令主要包括逻辑与、逻辑或、逻辑非、逻辑异或、逻辑移位等。逻辑非指令和算术运算中的求反指令含义相同。

4.程序控制类指令

这类指令主要实现改变指令执行方向的功能。主要包括无条件转移、条件转移、转子、返主、循环指令等。

计算机执行程序时，通常按 CPU 中程序计数器 PC 的内容取指令并执行。PC 中的内容为下一条（后续执行）指令的地址，在指令取指完成时，CPU 将自动修改 PC 的内容，以保持其总是后续执行的指令地址。

若当前指令为顺序执行指令，PC 的内容总是原 PC 的内容加上当前指令的长度(字节数）。

若当前指令为非顺序执行指令（如程序控制类指令），那么后续执行指令的地址（即 PC 的内容）在转移发生时，必须从当前指令的地址码字段中取得，而不是当前指令取指完成时的下一条指令地址。程序控制类指令的地址码给出的地址格式可以是直接给定的地址，也可以是相对于当前指令位量的偏移地址，或其他方式给出的地址。

5.输入输出类指令

这类指令主要实现 CPU 与外部设备间交换数据、传送控制命令及取得设备状态等功能。各种计算机的输入输出类指令差别较大，外部设备独立编址的计算机，其指令系统必须设置这类指令。

这类指令只有输入和输出两种指令。

6.字符串类指令

这类指令主要完成对字符串操作的功能，是一组非数值运算的指令。主要包含字符串转换（把一种编码格式的字符串转换成另一种编码格式的字符串）、字符串传送、字符串比较、字符串查找（查找字符串中某一子串）、字符串抽取（提取某一子串）、字符串替换（把某一子字符串用另一字符串替换）等指令。

7.系统控制类指令

这类指令主要完成改变系统的工作状态、实现操作系统所需要的特殊功能。大多数计算机将这类指令划为特权指令，只能被操作系统等系统软件使用，而用户一般不可以使用。主要包括停机、开中断、关中断、系统管理、存储管理指令等。

8.其他指令

除上述类型指令外，系统尚有一些必须具有特定功能的专用指令。这类指令主要包含状态寄存器置位、暂停、测试、空操作、中断返回等指令。

## （三）指令格式

1.基本格式

指令格式是指令字用二进制代码表示的结构形式，通常由操作码字段和地址码字段组成，操作码字段表征指令的操作特性与功能，如进行加法、取数等等；地址码字段通常指定参与操作的操作数的地址，特殊情况下也可能直接给出操作数本身。指令的基本格式如下：

| 操作码（OP） | 地址码（A） |
| --- | --- |

2.指令长度

指令的长度是指一条指令中所包含的二进制代码的位数，它取决于操作码字段的长度、操作数地址的个数及长度。指令长度应：①尽可能短；②等于字节的整数倍。

指令长度可以等于机器字长，也可以大于或小于机器字长。

指令长度等于机器字长的指令，称为单字长指令；

指令长度等于半个机器字长的指令，称为半字长指令；

指令长度等于两个机器字长的指令，称为双字长指令。

使用多字长指令的目的，在于提供足够的地址位来解决访问内存任何单元的寻址问题，但是使用多字长指令的主要缺点是必须多次访问内存以取出一整条指令，这降低了 CPU 的运算速度，同时又占用了更多的存储空间。

在一个指令系统中，若所有指令的长度都是相等的，称为定长指令字结构，这种指令结构简单；若各种指令的长度随指令功能而异，称为变长指令字结构，这种指令结构灵活，能充分利用指令长度，但指令的控制较为复杂。

3.操作码

指令系统中的每一条指令都有一个唯一确定的操作码，指令不同，其操作码的编码也不同。为了能表示整个指令系统中的全部指令，指令的操作码字段应当具有足够的位数。一般来说，一个包含 n 位的操作码最多能表示 $2^n$ 条指令。

指令操作码通常有两种编码格式：一种是固定格式，即操作码的长度固定，且集中放在指令字的一个字段中，这种格式对简化硬件设计非常有利，在字长较长的大中型计算机中广泛采用；另一种是可变格式，即操作码的长度可变，且分散地放在指令字的不同字段中，在字长较短的微型和小型计算机中广泛采用。

操作码长度不同会增加指令译码和分析的难度，使控制器的设计复杂。通常采用扩展操作码技术，使操作码的长度随地址数的减少而增加，既充分利用指令字的各字段，又在不增加指令长度的情况下扩展操作码的长度。

4.地址码

地址码用来指出该指令的源操作数的地址、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。

下面以主存地址为例，讨论双操作数指令的地址码结构。一条双操作数指令的除操作码之外，还应包含以下信息：第一操作数地址，用 A1 表示；第二操作数地址，用 A2 表示；操作结果存放地址，用 A3 表示；下条将要执行指令的地址，用 A4 表示。这些信息可以在指令中明显的给出，称为显地址；也可以依照某种事先的约定，用隐含的方式给出，称为隐地址。

（1）四地址指令

| OP | A1 | A2 | A3 | A4 |
|---|---|---|---|---|

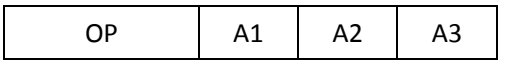
（A1）OP（A2）→A3

A4＝下条将要执行指令的地址

执行一条四地址指令需 4 次访问主存。

四地址指令可直接寻址的地址范围与地址字段的位数有关。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，则指令操作数的直接寻址范围为 $2^6$= 64。

**（2）三地址指令**



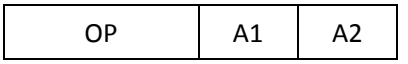

（A1）OP（A2）→A3

（PC）＋ 1＝ 下条将要执行指令的地址

执行一条三地址指令也需 4 次访问主存。

同理，若指令字长不变，设 OP 仍为 8 位，则 3 个地址字段各占 8 位，故三地址指令操作数的直接寻址范围可达 $2^8$ =256。

**（3）二地址指令**




（A1）OP（A2）→A1

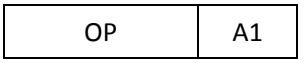
（PC）＋ 1＝下条将执行指令的地址

A1 中原存内容在指令执行后被破坏。

执行一条二地址指令需 4 次访问主存。若使其完成（A1）OP（A2）→ACC，中间结果暂存于累加器 ACC 中，则只需访存 3 次。

在不改变指令字长和操作码的位数前提下，二地址指令操作数的直接寻址范围为 $2^{12}$ =4 K。

**（4）一地址指令**




（ACC）OP（A1）→ ACC

（PC）＋ 1＝ 下条将要执行指令的地址

执行一条一地址指令需 2 次访问主存。

一地址指令短，又由于一个操作数已在 CPU 内，所以执行速度较快。

在指令字长仍为 32 位、操作码位数为 8 位的情况下，一地址指令操作数的直接寻址范围达 $2^{24}$= 16 M。

（5）零地址指令

| OP |
|---|

零地址指令的指令字中只有操作码，没有地址码。例如空操作、停机等不需要操作数的指令，或者操作数隐含在堆栈中，其地址为堆栈的栈顶或次栈顶。

指令中地址个数的选取要考虑诸多的因素。从缩短程序长度，用户使用方便等方面来看，选用三地址指令格式较好；从减少访存次数，简化硬件设计等方面来看，一地址指令格式较好。对于同一个问题，用三地址指令编写的程序最短，但指令长度最长，而用二、一地址指令来编写程序，程序的长度逐个递增，但指令的长度逐个递减。

示例：计算 x=(a×b+c-d) ÷(e+f)

①三地址指令

MUL　A，B，X

ADD　X，C，X

SUB　X，D，X

ADD　E，F，Y

DIV　X，Y，X

需要 5 条指令，每条指令 4 次访存，执行此程序共访存 20 次。

②二地址指令

MOV　X，A

MUL　X，B

ADD　X，C

SUB　X，D

MOV　Y，E

ADD　Y，F

DIV　X，Y

需要 7 条指令，其中 MOV 指令 3 次访存，算逻指令 4 次访存，执行此程序共访存 2×3+5×4=26 次。

③一地址指令

LOAD　E

ADD　F

```
STORE   X
LOAD    A
MUL     B
ADD     C
SUB     D
DIV     X
STORE   X
```

需要 9 条指令，每条指令 2 次访存，执行此程序共访存 9×2=18 次。

## 二、寻址方式

存储器既可用来存放操作数，又可用来存放指令。当某个操作数或某条指令存放在某个存储单元时，其存储单元的编号，就是该操作数或指令在存储器中的地址。因此，寻址方式可以分为数据寻址和指令寻址。寻找操作数的地址称为数据寻址，数据寻址方式较多，其最终目的都是寻找所需要的操作数。寻找下一条将要执行的指令地址称为指令寻址。

### （一）指令寻址

指令寻址比较简单，它又可以细分为顺序寻址和跳跃寻址。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；跳跃寻址是程序转移执行时的指令寻址方式，它通过转移类指令实现。

跳跃寻址的转移地址形成方式有三种：直接（绝对）、相对和间接寻址，它与数据寻址方式中的直接、相对和间接寻址是相同的，只不过寻找到的不是操作数的有效地址而是转移的有效地址而已。

### （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于哪一种寻址方式。指令的地址码字段通常都不代表操作数的真实地址，把它称为形式地址，记作 A。操

作数的真实地址称为有效地址，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所示。

| 操作码 OP | 寻址特征# | 形式地址 A |
|---|---|---|

寻址方式是根据指令中给出的地址码字段寻找真实操作数地址的方式。

指令中的形式地址 A—（寻址方式）→有效地址 EA

1.立即寻址

在取指令时，操作码和操作数被同时取出，不必再次访问存储器，从而提高了指令的执行速度，如图 3-1 所示。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，也称立即数。由于操作数是指令的一部分，故立即数的大小将受到指令长度的限制。


**图 3-1 立即寻址示意图**

2.直接寻址

指令中地址码字段给出的地址 A 就是操作数的有效地址：EA=A，如图 3-2 所示。直接寻址的缺点在于 A 的位数限制了操作数的寻址范围，且必须修改 A 的值，才能修改操作数的地址。

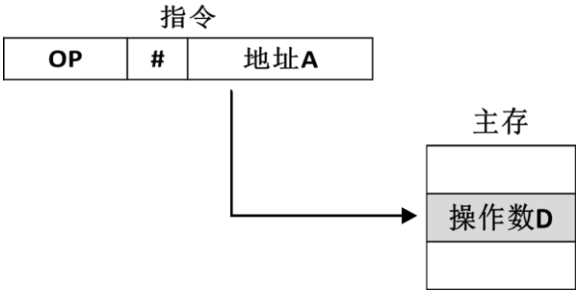
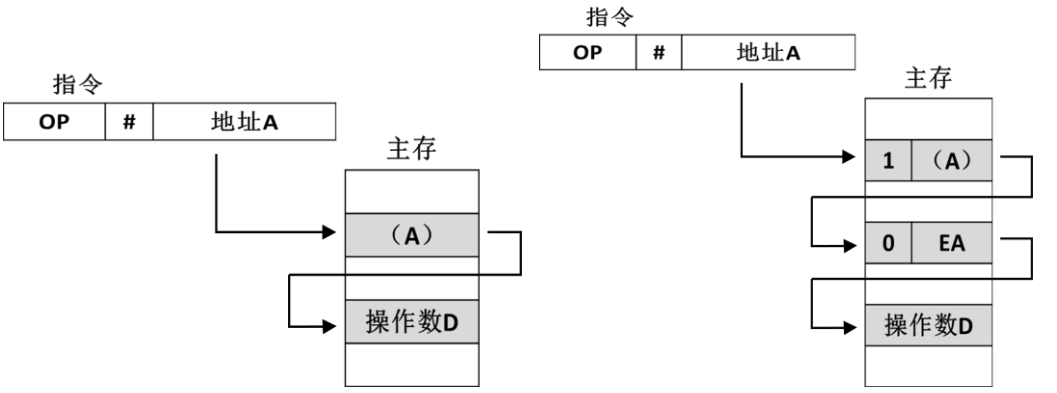



**图 3-2 直接寻址示意图**

3.间接寻址

指令中给出的地址 A 不是操作数的地址，而是存放操作数地址的地址：EA=（A），如图 3-3 所示。

间接寻址要比直接寻址灵活得多，它的主要优点为：一是扩大了寻址范围，可用指令的短地址访问大的主存空间，二是可将主存单元作为程序的地址指针，用以指示操作数在主存中的位置。当操作数的地址需要改变时，不必修改指令，只需修改存放有效地址的那个主存单元（间接地址单元）的内容就可以了。




（a）一级间接寻址　　（b）二级间接寻址

**图 3-3 间接寻址示意图**

除去一级间接寻址外，还有多级间接寻址。多级间接寻址为取得操作数需要多次访问主存，即使在找到操作数有效地址后，还需再访问一次主存才可得到真正的操作数。

若指令字长和存储字长均为 16 位，A 为 8 位，则直接寻址范围为 $2^8$，一次间接寻址的寻址范围可达 $2^{16}$。当多次间接寻址时，可用存储字的首位来标志间接寻址是否结束。如图 3-3（b）中，当存储字首位为“l”时，标明还需继续访存寻址；当存储字首位为“0”时，标明该存储字即为 EA。由此可见，存储字首位不能作为 EA 的组成部分，因此，它的寻址范围为 $2^{15}$。

4.寄存器寻址

指令中地址码部分给出某一通用寄存器的编号，所指定的寄存器中存放着操作数，如图 3-4 所示。它有两个明显的优点：一是从寄存器存取数据比主存快得多；二是由于寄存器的数量较少，其地址码字段比主存单元地址字段短得多。

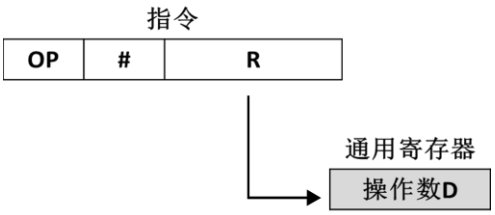



**图 3-4 寄存器寻址示意图**

5. 寄存器间接寻址

指令中的地址码给出某一通用寄存器的编号，被指定的寄存器中存放操作数的有效地址，而操作数则存放在主存单元中，如图 3-5 所示。

这种寻址方式的指令较短，并且在取指后只需一次访存便可得到操作数。

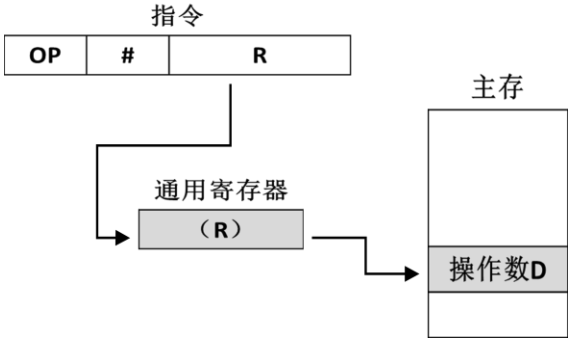



**图 3-5 寄存器间接寻址示意图**

6.变址寻址

把指令给出的形式地址 A 与变址寄存器 Rx 的内容相加，形成操作数有效地址：EA ＝ A ＋（Rx），Rx 的内容为变址值，如图 3-6 所示。

变址寻址是一种广泛采用的寻址方式，通常指令中的形式地址作为基准地址，而 Rx 的内容作为修改量。在遇到需要频繁修改地址时，无须修改指令，只要修改变址值就可以了。

如要把一组连续存放在主存单元中的数据（首地址是 A）依次传送到另一存储区（首地址为 B）中去，则只需在指令中指明两个存储区的首地址 A 和 B（形式地址），用同一变址寄存器提供修改量 K，即可实现（A＋K）→B＋K。变址寄存器的内容在每次传送之后自动地修改。

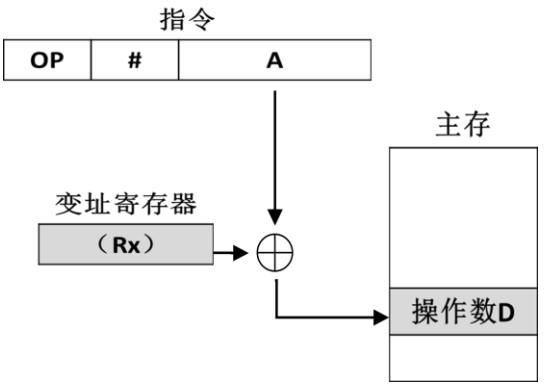



**图 3-6 变址寻址示意图**

7. **基址寻址**

将基址寄存器 Rb 的内容与形式地址 A 相加，形成操作数有效地址：EA＝（Rb）＋ A，基址寄存器的内容称为基址值，如图 3-7 所示。

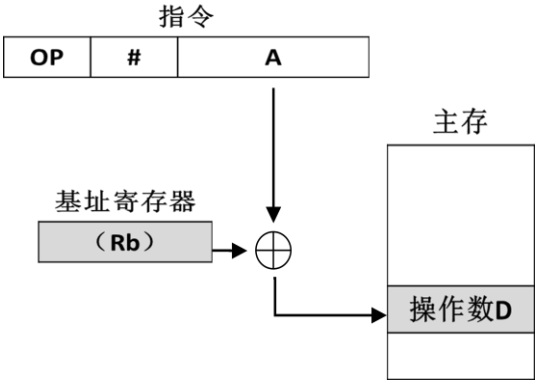



**图 3-7 基址寻址示意图**

基址寻址和变址寻址在形成有效地址时所用的算法是相同的，而且在一些计算机中，这两种寻址方式都是由同样的硬件来实现的。

但这两种寻址方式应用的场合不同，变址寻址是面向用户的，用于访问字符串、向量和数组等成批数据；而基址寻址面向系统，主要用于逻辑地址和物理地址的变换，用以解决程序在主存中的再定位和扩大寻址空间等问题。在某些大型机中，基址寄存器只能由特权指令来管理，用户指令无权操作和修改。

8. 相对寻址

相对寻址是基址寻址的一种变通，由程序计数器 PC 提供基准地址，即：

EA＝（PC）＋ A， A 是操作数和现行指令之间的相对位置，如图 3-8 所示。

相对寻址方式的特点是：操作数的地址不是固定的，它随着 PC 值的变化而变化，并且与指令地址之间总是相差一个固定值 A。当指令地址改变时，由于其位移量不变，使得操作数与指令在可用的存储区内一起移动，所以仍能保证程序的正确执行。采用 PC 相对寻址方式编写的程序可在主存中任意浮动，它放在主存的任何地方，所执行的效果都是一样的。指令中给出的位移量 A 可正、可负，通常用补码表示，故对于指令地址而言，操作数地址可能在指令地址之前或之后。

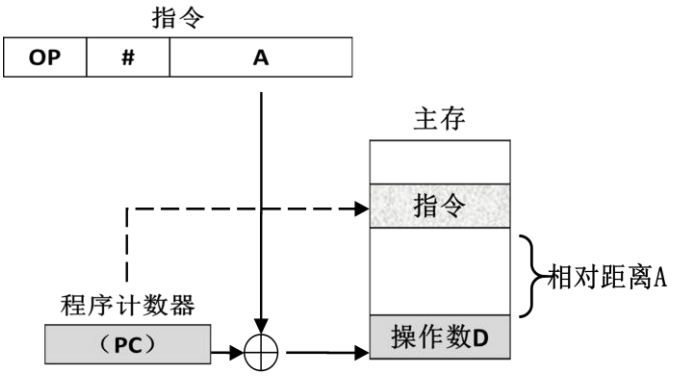



**图 3-8 相对寻址示意图**

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。如一地址指令格式，只给出一个操作数的地址，另一个操作数隐含在累加器 ACC 中，故累加器 ACC 对一地址指令格式来说是隐含地址，如图 3-9 所示。

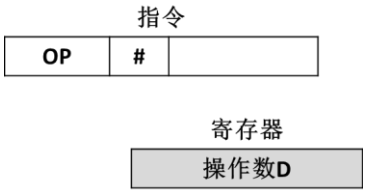



**图 3-9 隐含寻址示意图**

10.堆栈寻址

在堆栈寻址的指令字中没有形式地址码字段，它是一种零地址指令。堆栈寻址要求计算机中设有堆栈。堆栈既可用寄存器组来实现，也可利用主存的一部分空间作堆栈，前者称为硬堆栈，后者称为软堆栈。

## 三、RISC 技术

随着超大规模集成电路的飞速发展，计算机硬件结构越来越复杂，大多数计算机的指令系统多达几百条，这些计算机被称为复杂指令系统计算机，简称 CISC。CISC 的指令系统复杂庞大，指令数目多；CPU 中设有专用寄存器；指令字长不固定，指令格式多，寻址方式多；可访存指令不受限制；各种指令的执行时间相差大；采用微程序控制器；难以用优化编译生成高效的目标代码。

大量测试表明，最常使用的是一些比较简单的指令，这类指令仅占指令总数的 20％，但在各种程序中出现的频率却占 80%，其余大多数指令是功能复杂的指令，这类指令占指令总数的 80%，但其使用频率很低，仅占 20%。因此，人们把这种情况称为“20%-80%律”。

从“20%-80%律”出发，人们开始了对指令系统合理性的研究，提出了精简指令系统的想法，出现了精简指令系统计算机，简称 RISC。RISC 的主要特点是：

①选取使用频度较高的简单指令，以及很有用但不复杂的指令；

②指令长度固定，指令格式种类少，寻址方式种类少；

③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；

④CPU 中有多个通用寄存器。

⑤采用流水线技术，大部分指令在一个时钟周期内完成；

⑥控制器采用组合逻辑控制为主；

⑦采用优化编译技术。

# 第二节 总线系统

## 一、总线概述

计算机系统各部件之同的互连方式有两种，一种是各部件之同使用单独的连线，称为分散连接；另一种是将各部件连到一组公共信息传输线上，称为总线连接。

总线是连接多个部件的信息传输线，是各部件共享的传输介质。从不同角度，总线有不同分类方法：按数据传送方式，可分为串行传输总线和并行传输总线；按连接部件不同，可分为片内总线、系统总线和通信总线。

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、寄存器与算逻单元 ALU 之间都由片内总线连接。

通信总线用于计算机系统之间，或者计算机系统与其他系统如控制仪表、移动通信等之间的通信。

系统总线是微机系统中各部件之间传输信息的公共通路。信息可以从多个信息源中的任一信息源通过总线传送到多个信息接收部件中的任一部件。总线首先包括一组物理导线，这是信息传输的物理媒质。系统总线按照传输信息的不同，又可分为三类：地址总线、数据总线和控制总线。

### （一）地址总线

地址总线用来传送 CPU 发出的地址信息，是单向总线。

### （二）数据总线

数据总线用来传送数据信息，是双向总线。

### （三）控制总线

控制总线用来传送控制信号、时序信号和状态信息等，是双向总线。

## 二、总线的特性

物理特性：总线的物理连接方式；

功能特性：总线每一根线的功能；

电气特性：定义每一根线上信号的传递方向及有效电平范围；

时间特性：定义每一根线在什么时间有效。

## 三、总线性能指标

总线性能指标主要包括以下六个方面：

### （一）总线频率

总线的工作频率以 MHz 表示。它是总线工作进度的一个重要参数，工作频率越高，速度越快。

### （二）总线宽度

总线宽度是指数据总线的位数，用 bit（位）来表示，如 8 位、16 位、32 位、64 位总线宽度。

### （三）总线的数据传输率

总线的数据传输率是指在一定的时间内总线上可传送的数据总量，用每秒最大传输数据字节量来表示。总线的数据传输率的计算公式是：

总线的数据传输率＝（总线宽度/8 位）×总线频率

其单位是 MB/s，总线频率以 MHz 为单位。如 PCI 总线的总线频率为 33.3MHz，总线宽度为 32 位的情况下，其数据传输率为 133MB/s。

### （四）时钟同步/异步

总线上的数据与时钟同步工作的总线称为同步总线，与时钟不同步工作的总线称为异步总线。

### （五）总线复用

通常情况下，地址线和数据线在物理上是分开的，但为了提高总线的利用率和优化设计，在有些总线中将地址线和数据线共用同一组物理线路，然后分时传送地址或数据信号，这就是总线复用。

### （六）总线负载能力

不同的电路对总线的负载是不同的，即使是同一块电路板，在不同的工作频率下，总线的负载也不一样，所以对总线负载能力的指标并不太严格，一般情况下用可连接的扩充电路板的数目来反映总线的负载能力。

在这些性能指标中，总线频率、总线宽度和总线的数据传输率是最重要的 3 个指标。

## 四、总线结构

按照总线的排列布置及计算机内各部件连接方式的不同，可以将单机总线结构分为三种：单总线结构、双总线结构和三总线结构。

### （一）单总线结构

用一条系统总线连接计算机系统的各个功能部件，各功能部件间所有的信息传输都靠这条总线来实现，如图 3-10 所示。单总线结构多为小型机和微型机采用。

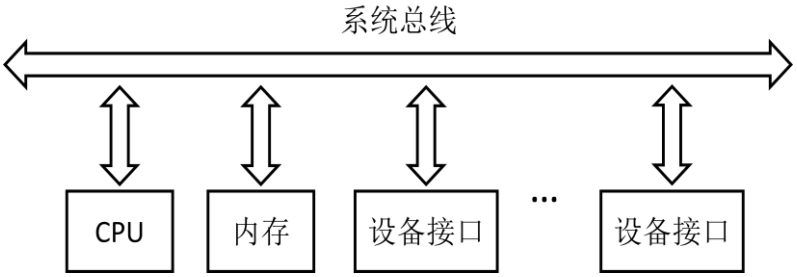



**图 3-10 单总线结构**

### （二）双总线结构

系统内增加了存储总线，存储总线专门负责 CPU 和主存之间的信息高速传输，如图 3-11 所示。双总线结构以增加硬件为代价，提高了 CPU 的效率，同时又保持了单总线结构系统简单、易于扩充的优点。

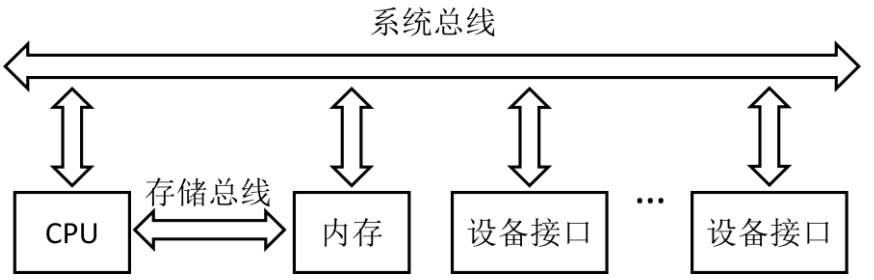



**图 3-11 双总线结构**

### （三）三总线结构

三总线结构在双总线结构的基础上增加了 I/O 总线，系统内有 3 条各自独立的总线：系统总线是 CPU、主存和通道之间进行信息传输的公共通路；I/O 总线是多个外部设备与通道之间进行数据传送的公共通路；存储总线负责 CPU 和主存之间的信息传输，如图 3-12 所示。

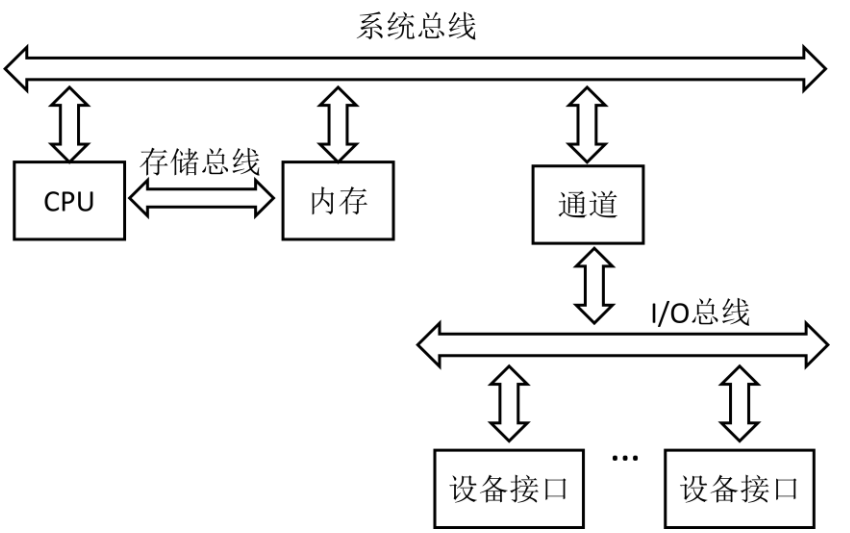



**图 3-12 三总线结构**

由于通道分担了一部分 CPU 的功能，可以实现对外设的统一管理及外设与主存之间的信息传输，所以三总线结构系统的效率有很大提高。

## 五、总线标准

总线标准是系统与各模块、模块与模块之间的一个互连标准界面，按总线标准设计的接口可视为通用接口，这为计算机接口的软硬件设计提供了方便。主要的总线标准有以下几种：

### （一）PCI

PCI（Peripheral Component Interconnect，外围部件互连）总线是现代计算机中最常用的总线之一，PCI 总线是并行的，有 32 位或 64 位，支持即插即用，支持 5V 和 3.3V 两种电压标准，具有良好的兼容性和可扩充性。

### （二）SCSI

SCSI（Small Computer System Interface，小型计算机系统接口）是一种通用的并行接口标准，用于在计算机和外部设备之间进行物理连接和传输数据。SCSI 通常用在硬盘和磁带上，也可以用于其他设备如扫描仪、CD 和 DVD 驱动器等。

### （三）RS-232C

RS-232C（ RS 即 Recommended Standard，232 为标识号，C 表示修改次数）是一种串行通信总线标准，它是应用于串行二进制交换的数据终端设备和数据通信设备之间的标准接口。

### （四） FireWire

FireWire 串行总线标准由苹果公司开发，被 IEEE 组织采用并定名为 IEEE1394 标准。FireWire 可实现即插即用，能连接多个不同的设备。

### （五）USB

USB(Universal Serial Bus，通用串行总线)是一种计算机串行接口总线标准，它可实现外设的简单快速连接，几乎所有的外设装置包括显示器、键盘、鼠标、打印机、数码相机、U 盘等可直接插入标准 USB 插口。

USB2.0 版的数据传输速度最高为 480Mbps，接口供电能力为 0.5A。

USB3.0 版的理论最高数据传输速度为 5.0Gbps，接口供电能力为 1A。USB3.0 引入了全双工数据传输，5 根线路中 2 根用来发送数据，2 根用来接收数据，1 根是地线，即 USB 3.0 可以同步全速地进行读写操作。之前的 USB 版本并不支持全双工数据传输。

## 六、总线控制

### （一）总线仲裁

系统中多个设备或模块可能同时申请对总线的使用权，为避免产生总线冲突，在多个申请者同时提出总线请求时，以一定的优先算法仲裁哪个应获得对总线的使用权。

按照总线仲裁电路的位置不同，总线仲裁方式可分为：集中式仲裁和分布式仲裁。

集中式总线仲裁的控制逻辑基本集中在一处，需要中央仲裁器，分为：链式查询方式、计数器定时查询方式、独立请求方式。

分布式仲裁不需要中央仲裁器，每个潜在的主方功能模块都有自己的仲裁号和仲裁器。当它们有总线请求时，把它们唯一的仲裁号发送到共享的仲裁总线上，每个仲裁器将仲裁总线上得到的号与自己的号进行比较。如果仲裁总线上的号大，则它的总线请求不予响应，并撤消它的仲裁号。最后，获胜者的仲裁号保留在仲裁总线上。显然，分布式仲裁是以优先级仲裁策略为基础。

以下对集中式总线仲裁的三种方式作简要介绍：

**1.链式查询**

链式查询时，总线授权信号 BG 串行地从一个 I/O 接口传送到下一个 I/O 接口，如图 3-13 所示。假如 BG 到达的接口无总线请求，则继续往下查询；假如 BG 到达的接口有总线请求，BG 信号便不再往下查询，该 I/O 接口获得了总线控制权。离中央仲裁器最近的设备具有最高优先级，通过接口的优先级排队电路来实现。

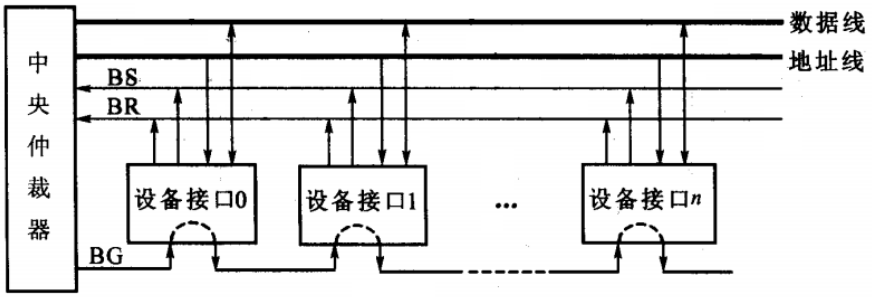



**图 3-13 链式查询方式**

链式查询的优点在于只用很少几根线就能按一定优先次序实现总线仲裁，很容易扩充设备。其主要缺点是对询问链的电路故障很敏感，如果第 i 个设备的接口中有关链的电路有故障，那么第 i 个以后的设备都不能进行工作。查询链的优先级是固定的，如果优先级高的设备出现频繁的请求时，优先级较低的设备可能长期不能使用总线。

2.计数器定时查询

如图 3-14 所示，总线上的任一设备要求使用总线时，通过总线请求线 BR 发出总线请求。中央仲裁器接到请求信号以后，在总线状态线 BS 为“0”的情况下让计数器开始计数，计数值通过一组地址线发向各设备。每个设备接口都有一个设备地址判别电路，当地址线上的计数值与请求总线的设备地址相一致时，该设备置“1”BS 线，获得了总线使用权，此时中止计数查询。

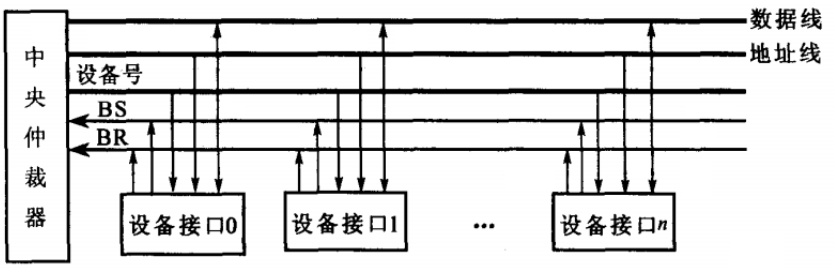



**图 3-14 计数器定时查询方式**

3.独立请求

如图 3-15 所示，独立请求的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。中央仲裁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。

这种方式的优点是响应时间快，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；还可以用屏蔽某个请求的办法，不响应来自无效设备的请求。

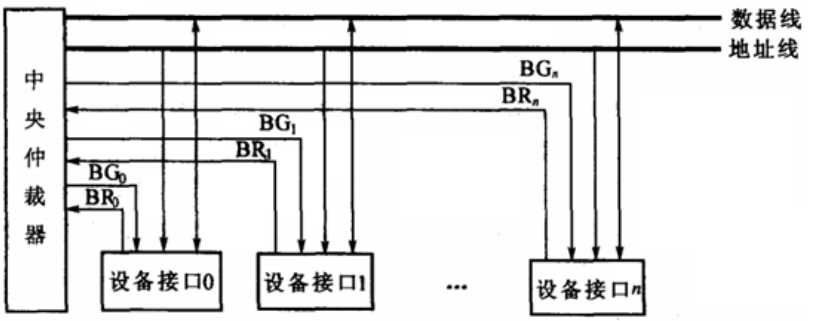



**图 3-15 独立请求方式**

## （二）总线通信控制

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。通常用四种方式：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。

这种通信的优点是规定明确、统一，模块间的配合简单一致，缺点是主、从模块时间配合属于强制性同步，必须在限定时间内完成规定的要求。

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

异步通信没有公共的时钟标准，不要求所有部件严格的统一操作时间，而是采用应答方式，即当主模块发出请求信号时，一直等待从模块反馈回来响应信号后，才开始通信。

异步通信的应答方式又可分为不互锁、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，不必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，便撤销其请求信号；从模块接到请求信号后，在条件允许时发出回答信号，并且经过一段时间确认主模块已收到回答信号后，自动撤销回答信号。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，必须待接到从模块的回答信号后再撤销其请求信号，有互锁关系；而从模块在接到请求信号后发出回答信号，但不必等待获知主模块的请求信号已经撤销，而是隔一段时间后自动撤销其回答信号，无互锁关系。由于一方存在互锁关系，一方不存在互锁关系，故称半互锁方式。

（3）全互锁方式

主模块发出请求信号，必须待从模块回答后再撤销其请求信号；从模块发出回答信号，必须待获知主模块请求信号已撤销后，再撤销其回答信号。双方存在互锁关系，故称为全互锁方式。

异步串行通信的数据传送速率用波特率来衡量。波特率是指单位时间内传送二进制数据的位数，单位用 bps(位/秒)表示，记作波特。

3. 半同步通信

半同步通信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。

4.分离式通信

为提高系统性能，人们提出了“分离式”的通信方式，其基本思想是将一个传输周期（或总线周期）分解为两个子周期。在第 1 个子周期中，主模块 A 在获得总线使用权后将命令、地址以及其他有关信息，包括该主模块编号发到系统总线上，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 2 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 模块所需的数据等一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。

## 七、总线的信息传送模式

总线的信息传送模式有 4 种：

### （一）读、写操作

读操作是从方到主方的数据传送；写操作是主方到从方的数据传送。

### （二）块传送操作

对固定块长度的数据连续读或写。

### （三）写后读、读修改写操作

写后读用于校验；读修改写用于多道程序系统中对共享存储资源的保护。

### （四）广播、广集操作

广播操作允许一个主方对多个从方进行写操作；和广播操作相反，广集将选定的多个从方数据在总线上完成 AND 或 OR 操作，用以检测多个中断源。

# 第四章 存储系统

## 第一节 存储器概述

存储器是计算机系统中的记忆设备，用来存放程序和数据。存储器中含有大量的存储单元。存储单元包含若干个存储元件（存储元），每个存储元能寄存一个“0”或“1”二进制数，故存储单元可存储一串二进制数，称为一个字，其位数称为存储字长，可为 8 位、16 位、32 位、64 位等，存储元件是存储器中最小的存储单元。

存放一个机器字的存储单元，通常称为字存储单元，相应的单元地址叫字地址。存放一个字节的单元，称为字节存储单元，相应的地址称为字节地址。若计算机中可编址的最小单位是字存储单元，则该计算机称为按字编址的计算机。若计算机中可编址的最小单位是字节，则该计算机称为按字节编址的计算机。通常计算机系统既可以按字寻址，也可按字节寻址。

### 一、数据的存储

#### （一）存储方式

通常计算机中的数据可以存储在寄存器或存储器中。

对寄存器而言，可以存放数据的长度是固定的，一般与机器字长一致。

对存储器而言，一次访问存储器单元的取得位数一般与计算机机器字长是相同的，目的是使 CPU 能够一次访问到数据的全部。

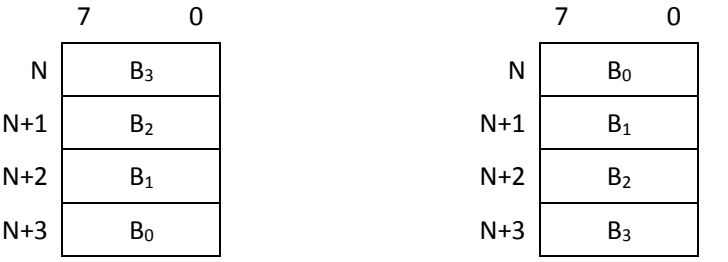
现代计算机的存储器大多以字节为单位编址，即每个存储单元地址对应一个字节的数据空间。存储器硬件特性决定了一次访问存储器所能取得的数据肯定是在连续的地址中，因此当数据长度大于一个字节时，数据就必须存放在从指定地址开始的、相邻的多个字节的存储器空间中。

按照数据的高字节和低字节在存储器中的存储次序，可分为大端（Big-endian）和小端（Little-endian）两种数据存储方式。将最低字节存储在指定存储器空间中最小地

址位置的存储方法称为小端方式；将最低字节存储在指定存储器空间中最大地址位置的存储方法称为大端方式。

假设一个数据由 4 个字节组成，需要存储在从 N 开始的存储器地址中，用 $B_3$、$B_2$、$B_1$、$B_0$ 分别表示操作数的 4 个字节，其中 $B_3$ 为数据的最高字节，$B_0$ 为数据的最低字节，该操作数的大端和小端数据存储方式结果如图 4-1 所示。

注意，大端、小端存储方式是指数据的字节间的存储次序，而字节内的数据无大、小端之分，永远是 bit7 为字节内数据的最高位，bit0 为字节内数据的最低位。




（a）大端存储方式　　（b）小端存储方式

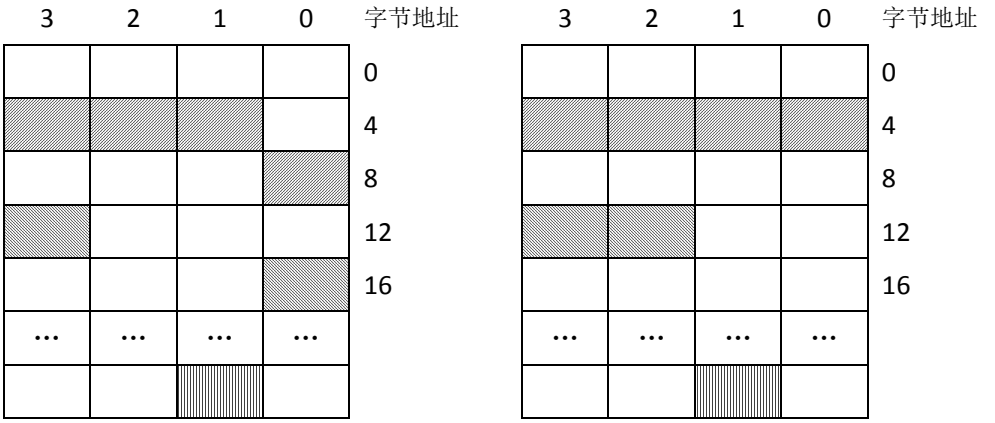
**图 4-1 数据在存储器中的两种存放方式**

## （二）数据对齐方式

对机器指令而言，操作数长度有字节、半字、字、双字等数据长度，要求存储器必须一次能够存取这些长度的操作数。因此，数据在存储器中应存储在存储空间阵列的同一行。假如操作数不能够存储在同一行，则机器指令取操作数至少需要两个访存周期，严重影响机器指令的执行速度。

假设机器字长和存储器一次访问位数均为 32 位，一个 32 位的数据存储在地址 A~A+3 中，一个 16 位的数据存储在地址 B~B+1 中，一个 8 位的数据存储在地址 C 中。。若 A、B 为任意地址，则 32 位、l6 位的数据就可能不在存储阵列的同一行，如图 4-2（a）所示，称为边界不对齐，不能够实现一次存取整个数据。而 C 为任意地址时，字节数据永远能够实现一次存取整个数据。只有当 A 为 4 的倍数、B 为 2 的倍数，即地址 A 最低两位为 00、地址 B 最低一位为 0 时，从而实现一次存取整个数据，称为边界对齐，如图 4-2（b）所示。

边界对齐存储提高了指令对操作数的访问速度，但缺点是浪费了部分存储空间，增加了硬件成本。而不按边界对齐存储，虽然节约了部分存储单元，但为达到不降低访存速度的效果，处理机或存储器的设计就变得复杂了。在早期的计算机中，有许多计算机为节约存储单元，采用信息不被整数边界存储方式。现在，随着硬件性能/价格的提高，基本上所有计算机都采用按边界对齐存储方式存储数据。




（a）边界不对齐　　（b）边界对齐

**图 4-2 数据的边界对齐**

## 二、存储器的分类

根据存储元件的性能及使用方法，存储器可有不同的分类。

### （一）按存储介质分类

目前使用的存储介质主要是半导体器件、磁性材料和光盘等。用半导体器件组成的存储器称为半导体存储器;用磁性材料做成的存储器有磁表面存储器和磁芯存储器，常见的磁表面存储器有磁盘和磁带；光盘存储器是应用激光在磁光材料上进行读/写的存储器。

### （二）按存取方式分类

若存储器中任何存储单元的内容都能被随机存取，且存取时间和存储单元的物理位置无关,这种存储器称为随机存储器。半导体存储器和磁芯存储器都是随机存储器。

如果存储器只能接某种顺序来存取，即存取时间和存储单元的物理位置有关，这种存储器称为顺序存储器，如磁带存储器就是顺序存储器。

磁盘存储器是半顺序存储器，在对磁盘读/写时，首先直接指出该存储器中的某个小区域（磁道），然后再顺序寻访，直至找到位置。即其前段是直接访同，后段是顺序访问，也称为直接存取存储器。

### （三）按存储器的读写功能分类

有些半导体存储器存储的内容是固定不变的，即只能读出而不能写入，因此这种半导体存储器称为只读存储器（ROM）。

早期采用掩模工艺，把原始信息记录在芯片中，一旦制成后无法更改，这种只读存储器称为掩模型只读存储器（Masked ROM，MROM）。之后又派生出可编程只读存储器（Programmable ROM，PROM）、可擦除可编程只读存储器（Erasable Programmable ROM，EPROM）以及用电可擦除可编程只读存储器（Electrically Erasable Programmable ROM，EEPROM）。PROM 只能写一次，而 EPROM 和 EEPROM 则可多次改写。后来出现的闪速存储器 Flash Memory，它具有 EPROM 和 EEPROM 的特点，但性价比更好、可靠性更高、擦除重写速度比 EEPROM 快得多。

既能读出又能写入的半导体存储器，称为随机存储器（RAM）。RAM 又分为 SRAM（静态随机存储器）和 DRAM（动态随机存储器）。

### （四）按信息的可保存性分类

断电后信息即消失的存储器，称为易失性存储器。断电后仍能保存信息的存储器，称为非易失性存储器，如磁性材料做成的存储器。

### （五）按在计算机系统中的作用分类

根据存储器在计算机系统中所起的作用，可分为主存储器、辅助存储器、缓冲存储器等。存储器的分类如图 4-3 所示。

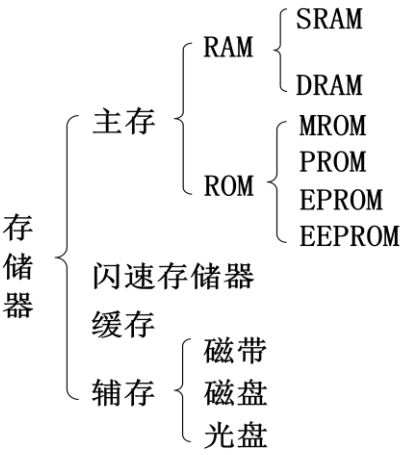



**图 4-3 存储器分类**

## 三、存储器的层次结构

### （一）性能指标

存储器的性能指标主要有三个：速度、容量和位价。

一般来说，速度越高，位价就越高；容量越大，速度就越低，价位也越低。人们追求大容量、高速度、低价位的存储器，这就需要存储系统来平衡各种存储器。图 4-4 反映了速度、容量和价位三者的关系，图中由上至下，价位越来越低，速度越来越慢，容量越来越大。

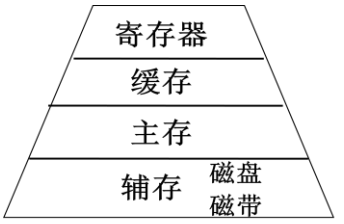



**图 4-4 存储器速度、容量和价位关系图**

## （二）层次结构

存储系统层次结构主要体现在缓存-主存和主存-辅存这两个存储层次上。缓存-主存层次主要解决 CPU 和主存速度不匹配的问题，主存-辅存层次主要解决存储系统的容量问题。现代的计算机系统几乎都具有这两个存储层次，构成了缓存、主存、辅存三级存储系统。

# 第二节 主存储器

## 一、主存的技术指标

主存储器的技术性能指标主要是存储容量、存取时间和存取周期。

在一个存储器中可以容纳的存储单元总数通常称为该存储器的存储容量。存储容量常用 B、KB、MB、GB、TB 等单位表示。存储容量反映了存储空间的大小。

存取时间又称存储器访问时间，是指从启动一次存储器操作到完成该操作所经历的时间。具体对于读操作而言，从一次读操作命令发出到该操作完成，将数据读入数据缓冲寄存器为止所经历的时间，即为存储器存取时间。

存取周期也称为存储周期，指连续启动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，存取周期略大于存取时间。存取时间和存取周期是主存的速度指标。

## 二、半导体存储芯片的基本结构

现代计算机的主存都由半导体集成电路构成，而半导体存储芯片采用超大规模集成电路制造工艺，在一个芯片内集成具有记忆功能的存储矩阵、译码驱动电路和读/写电路等。

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。

主存各存储单元的空间位置是由单元地址号来表示的，地址总线用来指出存储单元地址号，根据该地址可读入或写入一个存储字。

地址线的位数、数据线的位数均与芯片容量有关。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 $2^{10}\times4=4K$ 位；又如地址线为 14 根，数据线为 1 根，则其容量为 $2^{14}\times1=16K$ 位。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。如一个 64K×8 位的存储器可由 32 片 16K×1 位的存储芯片组成，每次读出一个存储字时，只需选中 8 片。

## 三、DRAM 的刷新

动态随机存储器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。

DRAM 是靠电容来存储信息的，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，不进行存储器的读/写操作，其存储单元内的原信息将会慢慢消失。为此，必须采用定时刷新的方法，它规定在一定的时间内，对 DRAM 的全部基本单元电路必作一次刷新，一般取 2ms（2000μs），这个时间称为刷新周期。刷新是一行行进行的，必须在刷新周期内，由专用的刷新电路来完成对基本电路的逐行刷新，才能保证 DRAM 内的信息不会丢失。通常有三种方式刷新：集中刷新、分散刷新和异步刷新。

### （一）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逐行进行刷新，此刻必须停止读/写操作。如对 128×128 矩阵的存储芯片进行刷新，刷新的时间相当于 128 个读周期。若读/写周期为 0.5μs，则对 128 行集中刷新共需 128×0.5＝64μs，其余的 1936μs（=2000－64）用来读/写或维持信息。由于在这 64μs 时间内不能进行读写操作，故称为“死时间”。

### （二）分散刷新

分散刷新把对每行存储单元的刷新分散到每个存取周期内完成。其中，把机器的存取周期分成两段，前半段用来读/写或维持信息，后半段用来刷新。若读/写周期为 0.5μs，则存取周期为 1μs，那么每隔 128μs 就可将 128 行的存储芯片全部刷新一遍，但这比允许的 2ms 间隔要短得多，且存取周期长了，整个系统的速度降低。

### （三）异步刷新

前两种方式的集合，既可缩短“死时间”，又充分利用最大刷新间隔为 2ms 的特点。如可在 2ms 内对 128 行各刷新一遍，即每隔 15.6μs（2000/128）刷新一行，每行的刷新时间仍为 0.5μs，但对每行来说，刷新间隔仍为 2ms，“死时间”缩短为 0.5μs。

## 四、存储器与 CPU 的连接

### （一）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，称为存储容量的扩展，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，只在位数方向扩展（加大字长）。位扩展的连接方式是将各存储芯片的地址线、片选线和读/写线相应地并联起来，而将各芯片的数据线单独列出。如用 64K×1 的 SRAM 芯片组成 64K×8 的存储器，需要 8 个芯片，如表 4-1 所示。

**表 4-1 位扩展组成**

| | 容量 | 地址 | 数据 |
|---|---|---|---|
| 存储器 | 64K×8 | 16 位 | 8 位 |
| 存储芯片 | 64K×1 | 16 位 | 1 位 |

当 CPU 访问该存储器时，其发出的地址和控制信号同时传给 8 个芯片，选中每个芯片的同一单元，其单元的内容被同时读至数据总线的相应位，或将数据总线上的内容分别同时写入相应单元。

2.字扩展

字扩展是指增加存储器字的数量，它仅在字数方向扩展，而位数不变。字扩展将芯片的地址线、数据线、读/写线并联，由片选信号来区分各个芯片。如用 16K×8 的 SRAM 组成 64K×8 的存储器，需要 4 个芯片，如表 4-2 所示。

表 4-2 字扩展组成

| | 容量 | 地址 | 数据 |
|---|---|---|---|
| 存储器 | 64K×8 | 16 位 | 8 位 |
| 存储芯片 | 16K×8 | 14 位 | 8 位 |

在同一时间内四个芯片中只能有一个芯片被选中。四个芯片的地址分配如图 4-5 所示：

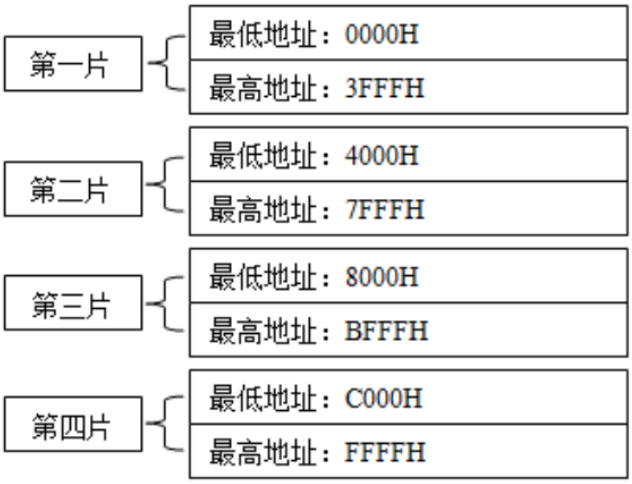



图 4-5 字扩展的地址分配

3. 字和位同时扩展

字和位同时扩展是指既增加存储字的数量，又增加存储字长。当构成一个容量较大的存储器时，往往需要在字数方向和位数方向上同时扩展，这将是前两种扩展的组合，实现起来也较为容易。如用 16K×4 的 SRAM 组成 64K×8 的存储器，需要 8 个芯片，如表 4-3 所示。

表 4-3 字和位同时扩展组成

| | 容量 | 地址 | 数据 |
|---|---|---|---|
| 存储器 | 64K×8 | 16 位 | 8 位 |
| 存储芯片 | 16K×4 | 14 位 | 4 位 |

## （二）存储器与 CPU 的连接

存储芯片与 CPU 芯片相连时，特别要注意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

存储芯片的容量不同，其地址线数也不同，CPU 的地址线数往往比存储芯片的地址线数多。通常总是将 CPU 地址线的低位与存储芯片的地址线相连。CPU 地址线的高位或在存储芯片扩充时用，或做其他用途，如片选信号等。

### 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

片选线的连接是 CPU 与存储芯片正确工作的关键。存储器由许多存储芯片组成，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。

### 5.合理选择存储芯片

合理选择存储芯片主要是指存储芯片类型（RAM 或 ROM）和数量的选择。通常选用ROM存放系统程序、标准子程序和各类常数等。RAM则是为用户编程而设置的。此外，在考虑芯片数量时，要尽量使连线简单方便。

# 第三节 高速缓冲存储器

## 一、背景

在多体并行存储系统中，由于 I/O 设备向主存请求的级别高于 CPU 访存，这就出现了 CPU 等待 I/O 设备访存的现象，降低了 CPU 的工作效率。为了避免 CPU 与 I/O 设备争抢访存，在 CPU 和主存之间加一级缓存；另一角度看，Cache 的提出也缓解了主存和 CPU 之间速度不匹配的问题。

## 二、工作原理

CPU 欲读出主存的某个字时，有两种可能：一种是所需的字已在缓存中，即可直接访问 Cache（CPU 与 Cache 之间通常一次传送一个字），称为 CPU 访问 Cache 命中；另一种是所需的字不在 Cache 中，此时需将该字所在的主存整个字块一次调入 Cache 中（Cache 与主存之间是字块传送），称为 CPU 访问 Cache 不命中。

Cache 的容量与块长是影响 Cache 效率的重要因素，通常用“命中率”来衡量 Cache 的效率。命中率是指 CPU 要访问的信息已在 Cache 内的比率。在一个程序执行期间，设 $A$ 为访问 Cache 的总命中次数，$B$ 为访问主存的总次数，则命中率 $H$ 为

$$H=\frac{A}{A+B}$$

设 $T$ 为命中时的 Cache 访问时间，$t$ 为未命中时的主存访问时间，$1-H$ 表示未命中率，则 Cache-主存系统的平均访问时间 $T_a$ 为

$$T_a=T\times H+t\times(1-H)$$

当然，以较小的硬件代价使 Cache-主存系统的平均访问时间越 $T_a$ 接近于 $T$ 越好。用 E 表示访问效率，则有

$$E=\frac{T}{T_a}\times 100\%$$

上面介绍的是读操作，而写操作则比较复杂，因为对 Cache 块内写入的信息，必须与被映射的主存块内的信息完全一致。当程序运行过程中需对某个单元进行写操作时，会出现如何使 Cache 与主存内容保持一致的问题。目前主要采用以下几种方法：

①写直达法，即写操作时数据既写入 Cache 又写入主存，能随时能保证主存和 Cache 的数据始终一致，但增加了访存次数；

②写回法，即写操作时只把数据写入 Cache 而不写入主存，但当 Cache 数据被替换出去时才写回主存，这样就会导致 Cache 中的数据会与主存中的不一致。为了识别 Cache 中的数据是否与主存一致，Cache 中的每一块要增设一个标志位，该位有两个状态：“清”（表示未修改过，与主存一致）和“浊”（表示修改过，与主存不一致）。Cache 替换时，“清”的 Cache 块不必写回主存，因为此时主存中相应块的内容与 Cache 块一致。在写 Cache 时，要将该标志位设置为“浊”，替换时此 Cache 块要写回主存，同时要使标志位为“清”。

## 三、改进

### （一）单一缓存和两级缓存

单一缓存是指在 CPU 和主存之间只设一个缓存。这个缓存直接与 CPU 制作在同一个芯片内，故又称为片内缓存。如果在主存与片内缓存之间再加一级缓存，称为片外缓存，由静态 RAM 组成。

由片外缓存和片内缓存组成的 Cache 称为两级缓存，并称片内缓存为第一级，片外缓存为第二级。

### （二）统一缓存和分立缓存

统一缓存是指指令和数据都存放在同一缓存内的 Cache；分立缓存是指指令和数据分别存放在两个缓存中，一个称为指令 Cache，一个称为数据 Cache。

## 四、地址映射

由主存地址映射到 Cache 地址称为地址映射。地址映射方式有很多，有直接映射、全相联映射、组相联映射。

### （一）直接映射

直接映射是一种固定的映射关系，每个主存块只与一个缓存块相对应，映射关系公式为：

$$i = j \bmod C \quad 或 \quad i = j \bmod 2^C$$

其中，$i$ 为缓存块号，$j$ 为主存块号，$C$ 为缓存块数或缓存地址位数。

如缓存有 0～3 号四块，主存有 0～15 号十六块，则直接映射方式主存块和缓存块的对应关系如表 4-4 所示：

**表 4-4 直接映射方式**

| 缓存块 | 主存块 |
|---|---|
| 0 | 0，4， 8，12 |
| 1 | 1，5， 9，13 |
| 2 | 2，6，10，14 |
| 3 | 3，7，11，15 |

这种方式的优点是实现简单，只需利用主存地址的某些位直接判断，即可确定所需字块是否在缓存中；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利用。

### （二）全相联映射

全相联映射允许主存中每一字块映射到 Cache 中的任何一块位置上。优点是灵活，命中率也高，缩小了块冲突率；缺点就是所需的逻辑电路较多，成本较高。

### （三）组相联映射

组相联映射是对直接映射和全相联映射的一种折中。它把 Cache 分为 Q 组，每组有 R 块，则

$$i = j \bmod Q$$

其中，i 为缓存的组号，j 为主存的块号。

组相联映射本质为分组采用的是直接映射的理念，而在每个分组内采用的是全相联映射的理念。

## 五、替换策略

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，需要替换掉 Cache 的数据，这就产生了替换策略问题。常用的替换算法有先进先出（First In First Out，FIFO）算法，近期最少使用（Least Recently Used，LRU）算法，随机法。

# 第五章 中央处理器

## 第一节 CPU 的组成和功能

### 一、CPU 的组成

计算机的工作过程就是程序的运行过程，程序一旦装入内存，就可以由计算机来自动完成取出指令和执行指令。专门用来完成此项工作的计算机部件称为中央处理器，简称 CPU。CPU 是整个计算机的核心，主要包括运算器和控制器。

运算器由算术逻辑单元（ALU）、累加寄存器、数据缓冲寄存器和状态标志寄存器组成，它是数据加工处理部件。相对控制器而言，运算器接受控制器的命令而进行动作，即运算器所进行的全部操作都是由控制器发出的控制信号来指挥的，所以它是执行部件。运算器有两个主要功能：

（1）执行所有的算术运算；

（2）执行所有的逻辑运算，并进行逻辑测试，如两个值的比较。

通常，一个算术操作产生一个运算结果，而一个逻辑操作则产生一个判决。

控制器是计算机系统的指挥中心，由程序计数器、指令寄存器、指令译码器、时序产生器和操作控制器组成。控制器的主要功能有：

（1）从内存中取出指令，并指出下一条指令在内存中的位置；

（2）对指令进行译码或测试，并产生相应的操作控制信号，以启动规定的动作；

（3）指挥并控制 CPU、内存和输入/输出设备之间数据流动的方向。

### 二、CPU 的功能

CPU 的主要功能有：

### （一）指令控制

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

### （二）操作控制

一条指令的功能往往是由若干个操作信号的组合来实现的，因此，CPU 管理并产生由内存取出的每条指令的操作信号，把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。

### （三）时间控制

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

### （四）数据加工

数据加工就是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，这是 CPU 的根本任务。

## 三、CPU 的主要寄存器

CPU 中的主要寄存器是用来暂时保存在运算和控制过程中的中间结果、最终结果以及控制、状态信息的，在 CPU 中有两类寄存器：一类是通用寄存器，另一类是专用寄存器。

## （一）通用寄存器

通用寄存器可用来存放原始数据和运算结果，有的还可以作为变址寄存器、基址寄存器、栈指针等。通用寄存器都属于用户可见的寄存器，即可以通过机器语言或汇编语言访问的寄存器。

## （二）专用寄存器

专用寄存器是 CPU 专门用来完成某一种特殊功能的寄存器。这其中有一部分属于用户可见的寄存器，而另一部分在 CPU 中起操作控制作用的寄存器在大部分机器上对用户是透明的。

### 1.指令寄存器（IR）

指令寄存器用来存放从存储器中取出的指令。当指令从主存取出暂存于指令寄存器之后，在执行指令的过程中，指令寄存器的内容不允许发生变化，以保证实现指令的全部功能。

### 2.程序计数器（PC）

程序计数器用来存放现行指令的地址或接着要执行的下条指令地址。

对于顺序执行的情况，PC 的内容修改通常是对 PC 值加 1。当遇到转移指令时，PC 的内容（即后续指令的地址）必须从指令寄存器中的地址字段取得。

### 3.存储器数据寄存器（MDR）

存储器数据寄存器用来暂时存放由主存储器读出的一条指令或一个数据字；反之，当向主存存入一条指令或一个数据字时，也暂时将它们存放在存储器数据寄存器中。

### 4.存储器地址寄存器（MAR）

存储器地址寄存器用来保存当前 CPU 所访问的存储单元的地址。由于主存和 CPU 之间存在着操作速度上的差别，所以必须使用地址寄存器来保持地址信息，直到主存的读/写操作完成为止。

5.状态标志寄存器（PSWR）

状态标志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，如进位标志，结果为零标志等。除此之外，状态标志寄存器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。

# 第二节 指令周期和指令流水

## 一、指令周期

CPU 每取出并执行一条指令所需的全部时间称为指令周期，也即 CPU 完成一条指令的时间。其中，取指阶段完成取指令和分析指令的操作，又称取指周期；执行阶段完成执行指令的操作，又称执行周期。

在大多数情况下，CPU 就是按“取指-执行-再取指-再执行……”的顺序自动工作的。由于各种指令的操作功能不同，因此各种指令的指令周期也不尽相同。

当遇到间接寻址的指令时，由于指令字中只给出操作数有效地址的地址，因此，为了取出操作数，需先访问一次存储器，取出有效地址，然后再访问存储器，取出操作数。故间接寻址的指令周期就包括取指周期、间址周期和执行周期 3 个阶段，其中间址周期用于取操作数的有效地址。

当 CPU 采用中断方式实现主机与 I/O 设备交换信息时，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 设备提出中断请求。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。此时，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。

## 二、指令流水

### （一）原理

计算机的流水处理过程同工厂中的流水装配线类似，为了实现流水，首先必须把输入的任务（或过程）分割为一系列子任务，使各子任务能在流水线的各个阶段并发地执行，将任务连续不断地输入流水线，从而实观了子任务级的并行。因此流水处理大幅度地改善了计算机的系统性能，是在计算机上实现时间并行性的一种非常经济的方法。

计算机执行程序是按顺序的方式进行的，即程序中各条机器指令是按顺序串行执行的。如按 4 个周期完成一条指令来考虑，串行执行的过程如图 5-1 所示。

| 取指1 | 计算地址1 | 取操作数1 | 运算并存结果1 | 取指2 | 计算地址2 | 取操作数2 | …… |
|---|---|---|---|---|---|---|---|

**图 5-1 指令串行执行过程**

如将一条指令分成 4 段，若每段所需时间为 T，那么一条指令的时间为 4T，但当第一条指令处理完后每隔 T 时间就能得到一条指令的处理结果，平均速度提高到 4 倍，把这种处理机称为流水线处理机，其工作过程如图 5-2 所示。

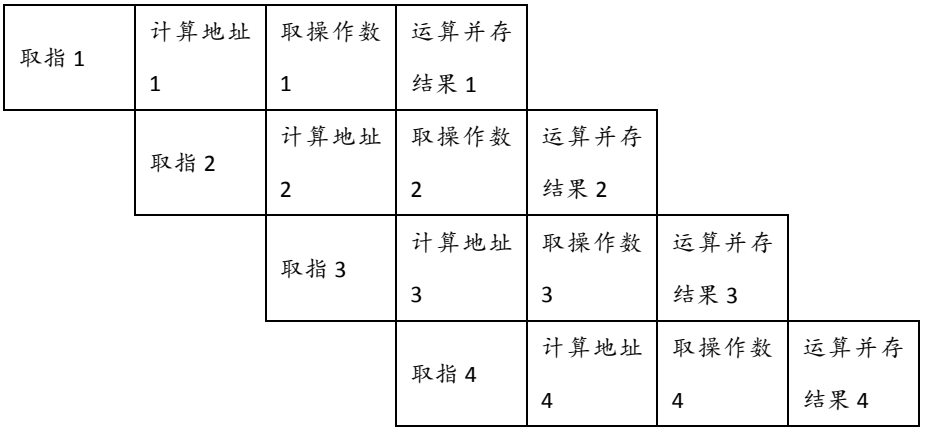



**图 5-2 流水线处理机**

## （二）影响流水性能的因素

流水过程中会出现三种相关，使得不断流实现起来很困难，这三种相关是：资源相关（结构相关）、数据相关和控制相关。

1. 资源相关

资源相关是指多条指令进入流水线后在同一机器时钟周期内争用同一个功能部件所发生的冲突，也称结构相关。解决冲突的方法可以让流水线在完成前一条指令对数据的存储器访问时，暂停取后一条指令的操作，另一种方式是采用将指令和数据分别存在两个存储器中来减少冲突。

2. 数据相关

数据相关是在一个程序中，如果必须等前一条指令执行完毕后，才能执行后一条指令，那么这两条指令就是数据相关的。其解决办法主要有两个：

一个是采用后推法，即将相关指令延迟到所需操作数被写回到寄存器后再执行；第二个是采用定向技术，即直接将执行结果送到其他指令需要的地方，而不必等某条指令的执行结果送回寄存器，再从寄存器取出该结果来作为下条指令的操作数。

3. 控制相关

控制相关冲突是由转移指令引起的。当执行转移指令时，依据转移条件的产生结果，可能是顺序取下条指令，也可能转移到新的目标地址取指令，从而使流水线发生断流。解决办法可以采用转移猜测法等来减少转移指令对流水线性能的影响。

## （三）流水线的性能

流水线的性能通常用三个指标来衡量：吞吐率、加速比和效率。

1. 吞吐率

吞吐率是指单位时间内流水线所完成指令或输出结果的数量。吞吐率有最大吞吐率和实际吞吐率之分。

最大吞吐率是指流水线在连续流动达到稳定状态后获得的吞吐率。对于 m 段指令流水而言，若各段的时间均为 Δt，则最大吞吐率为：

$$T_{p\max}=\frac{1}{\Delta t}$$

实际吞吐率是指流水线完成 n 条指令的实际吞吐率。实际吞吐率总小于最大吞吐率。对于 m 段指令流水，若各段的时间均为 Δt，则除第一条指令需 m •Δt 外，其余(n-1)条指令，每隔 Δt 就有一个输出，即共需 m • Δt +(n-1) Δt 时间，故实际吞吐率为：

$$T_p=\frac{n}{(m+n-1)\Delta t}$$

2. 加速比

加速比是指流水线的速度与等功能的非流水线的速度之比。其计算公式为：

$$S_p=\frac{\text{顺序执行时间}T_0}{\text{流水线执行时间}T_m}=\frac{\mathrm{mn}\Delta t}{(m+n-1)\Delta t}=\frac{m\mathrm{n}}{m+n-1}$$

3. 效率

效率是指流水线中各功能段的利用率，通常用流水线各段处于工作时间的时空区与流水线中各段总的时空区之比来衡量流水线的效率。图 5-3 是 4 段流水线的时空图，由图易知效率的计算公式为：

$$E=\frac{\text{流水线各段处于工作时间的时空区}}{\text{流水线中各段总的时空区}}=\frac{1\cdot T_0}{\mathrm{m}\cdot T_m}=\frac{\mathrm{mn}\Delta t}{m(m+n-1)\Delta t}=\frac{\mathrm{n}}{m+n-1}$$

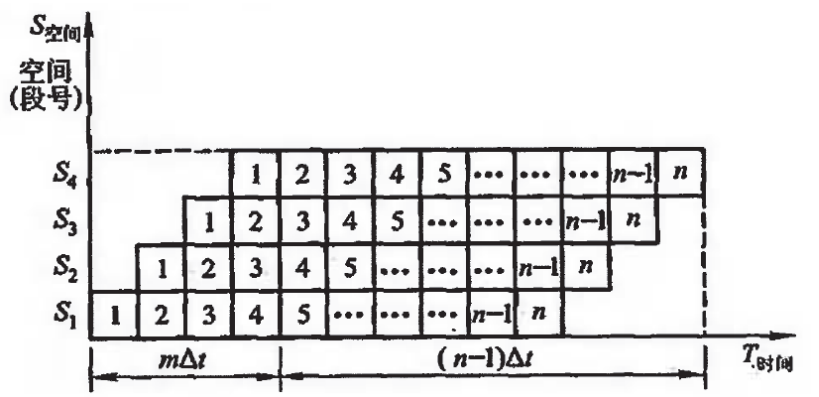



**图 5-3 各段时间相等的流水线时空图**

## （四）流水线中的多发技术

为了在一个时钟周期内，产生更多条指令的结果，可开发流水线中的多发技术，包括超标量技术、超流水线技术、超长指令字技术，如图 5-4 所示。

1.超标量技术

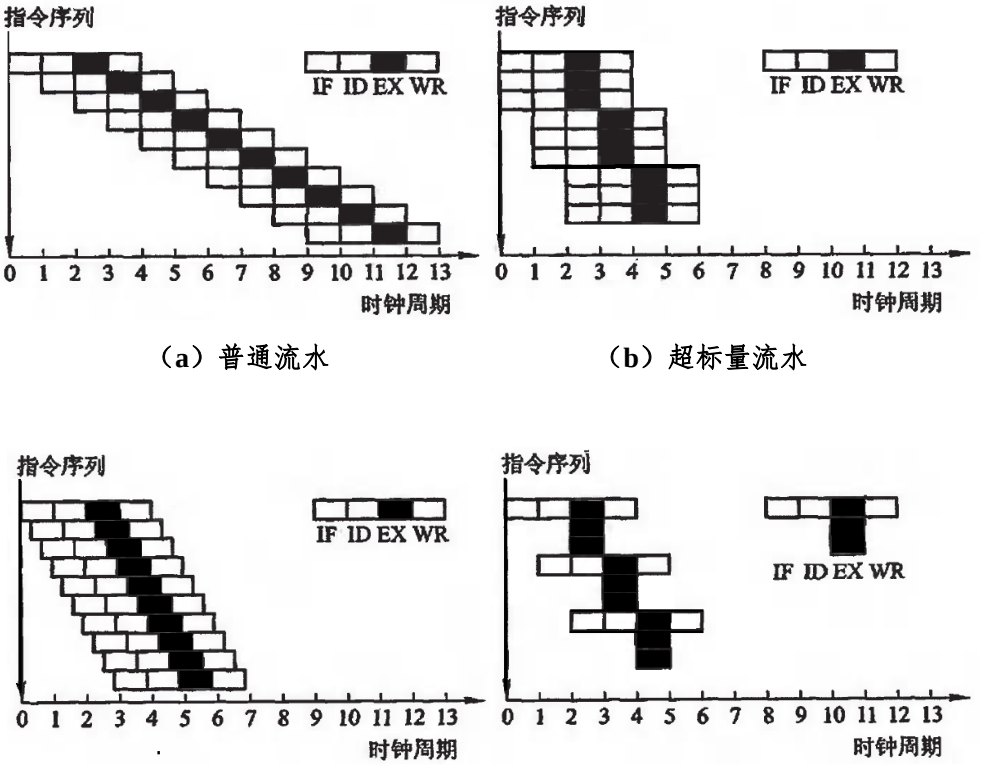
超标量技术指在每个时钟周期内可同时并发多条独立指令，即以并行操作方式将两条或两条以上指令编译并执行。

2.超流水线技术

超流水线技术是将一些流水线寄存器插入到流水线段中，相当于将流水线再分段，如将原来的一个时钟周期又分为 3 段，使超流水线的处理器周期比普通流水线的处理器周期短。

3.超长指令字（VLIW）技术

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，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。




（a）普通流水　　（b）超标量流水

（c）超流水线　　（d）超长指令字

图 5-4 流水线技术比较

# 第三节 时序系统和控制方式

## 一、时序系统

### （一）机器周期

指令周期常常用若干个 CPU 周期数来表示，CPU 周期也称为机器周期。机器周期可看做是所有指令执行过程中的一个基准时间，机器周期取决于指令的功能及器件的速度。由于 CPU 内部的操作速度较快，而 CPU 访问一次内存所花的时间较长，因此通常用内存中读取一个指令字的最短时间来规定 CPU 周期。在存储字长等于指令字长的前提下，取指周期也可看做机器周期，即一条指令的取出阶段需要一个 CPU 周期时间。

### （二）时钟周期（节拍）

一个机器周期又包含有若干个时钟周期（通常称为节拍或 T 周期）。

在一个机器周期里可完成若干个微操作，每个微操作都需要一定的时间，可用时钟信号来控制产生每一个微操作命令。用时钟信号控制节拍发生器，就可产生节拍。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，它是控制计算机操作的最小时间单位。

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、节拍组成了多级时序系统。

## 二、控制方式

控制单元（CU）是提供完成计算机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主要有两种：一种是组合逻辑设计方法，为硬连线逻辑；另一种是微程序设计方法，为存储逻辑。

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为 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。

### （一）同步控制方式

同步控制方式是指，任何一条指令或指令中任何一个微操作的执行都是事先确定的，并且都是受统一基准时标的时序信号所控制的方式。由于不同的指令，操作时间长短不一致，同步控制方式应以最复杂指令的操作时间作为统一的时间间隔标准。

这种控制方式设计简单，容易实现，但是对于许多简单指令来说会有较多的空闲时间，造成较大数量的时间浪费，从而影响了指令的执行速度。在同步控制方式中，各指令所需的时序由控制器统一发出，所有微操作都与时钟同步，所以又称为集中控制方式或中央控制方式。

### （二）异步控制方式

异步控制方式中，各项操作不采用统一的时序信号控制，而根据指令或部件的具体情况决定，需要多少时间，就占用多少时间。

异步控制采用不同时序，没有时间上的浪费，因而提高了机器的效率，但是控制比较复杂。由于这种控制方式没有统一的时钟，而是由各功能部件本身产生各自的时序信号自我控制，故又称为分散控制方式或局部控制方式。

### （三）联合控制方式

这是同步控制和异步控制相结合的方式。实际上现代计算机中几乎没有完全采用同步或完全采用异步的控制方式，大多数是采用联合控制方式。通常的设计思想是：在功能部件内部采用同步方式或以同步方式为主的控制方式，在功能部件之间采用异步方式。

### （四）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，如 Reset 键、连续或单条执行转换开关及符合停机开关，来达到人工控制的目的。

# 第四节 微程序设计

微程序设计技术的实质是将程序设计技术和存储技术相结合，即用程序设计的思想方法来组织操作控制逻辑，将操作控制信号按一定规则进行信息编码，形成微指令，存放在一个只读存储器中。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。

## 一、基本概念

一条机器指令对应于一段微程序，微程序是一系列微指令的有序集合。微程序和程序是两个不同的概念。程序由机器指令组成，它是由软件设计人员事先编制好并存放在主存或辅存中的。而微程序是由微指令组成，它实际上是机器指令的实时解释器，由计算机的设计者事先编制好并存放在控制存储器中的。对于程序员来说，计算机系统中微程序的结构和功能是透明的。

一条微指令由一组实现一定操作功能的微命令构成。微命令是控制计算机各部件完成某个基本微操作的命令。这些微操作是计算机中最基本的、不可再分解的操作。微命令和微操作是一一对应的。微操作可分为相容性和相斥性两种，所谓相容性的微操作，是指在同时或同一个 CPU 周期内可以并行执行的微操作，所谓相斥性的微操作，是指不能在同时或不能在同一个 CPU 周期内并行执行的微操作。

一条微指令通常包含两部分信息：

（1）操作控制字段，又称微操作码字段，用以产生某一步操作所需的各微操作控制信号。

（2）顺序控制字段，又称微地址码字段，用以控制产生下一条要执行的微指令地址。

微程序控制的计算机涉及到两个层次：一个是机器语言或汇编语言程序员所看到的传统机器层，包括：机器指令、程序、主存储器；另一个是机器设计者看到的微程序层，包括：微指令、微程序和控制存储器。

## 二、微指令编码

微指令编码法指的是微指令中的操作控制字段的编码方法。通常有以下三种方法：

### （一）直接编码法（直接控制法）

操作控制字段中的各位分别可以直接控制计算机，不需要进行译码。操作控制字段的每一个独立的二进制位代表一个微命令，该位为“1”表示这个微命令有效，为“0”表示这个微命令无效。

这种方法结构简单，并行性强，操作速度快，但是微指令字太长，若微命令的总数为 N 个，则微指令字的操作控制字段就要有 N 位。

### （二）字段编码法

字段编码法可进一步分为字段直接编码法和字段间接编码法。

#### 1.字段直接编码法

字段直接编码法将一组相斥性的微命令信号组成一个字段，然后通过译码器对每一个字段译码，便可对应一个微命令。各字段都可以独立地定义本字段的微命令，而和其他字段无关，这种方式靠字段直接译码发出微命令，因此又称为显式编码。这种方法缩短了微指令字，目前应用较为普遍。

#### 2.字段间接编码法

字段间接编码法是在字段直接编码法的基础上，用来进一步缩短微指令字长的方法。间接编码的含义是，一个字段的某些编码不能独立地定义某些微命令，而需要与其他字段的编码来联合定义，因此又称为隐式编码。

这种方法虽然可以进一步缩短微指令字长，但因削弱了微指令的并行控制能力，因此通常用做字段直接编码法的一种辅助手段。

字段编码法中操作控制字段的分段原则：

把互斥性的微命令分在同一段内，兼容性的微命令分在不同段内。这样不仅有助于提高信息的利用率，缩短微指令字长，而且有助于充分利用硬件所具有的并行性，加快执行的速度。

一般每个小段还要留出一个状态，表示本字段不发出任何微命令。因此当某字段的长度为三位时，最多只能表示七个互斥的微命令，通常用 000 表示不操作。

### （三）最短编译法

最短编译法的基本思想是：每一条 L 位字长的微指令只定义一个微命令。如若 L 取值为 6，就有 64 种编码状态，则可定义的微指令最多有 64 条。

按最短编译法编码的微指令比较简单，且微指令字长较短，微指令字中各位都能得到充分利用，然而，由于每条微指令只能对应一个微命令，因此并行控制能力差，微程序长度较长，执行速度慢。

假设某计算机共有 256 个微命令，如果采用直接控制法，微指令的操作控制字段就要有 256 位；而如果采用最短编译法，操作控制字段只需要 8 位就可以了；如果采用字段直接编码法，若 4 位为一个段，共需 18 段，操作控制字段只需 72 位，而且在同一时刻可以并行发出 18 个不同的微命令。

## 三、微指令格式

微指令有垂直型和水平型之分，水平型微指令则具有良好的并行性，每条微指令可以完成较多的基本操作。垂直型微指令接近于机器指令的格式，每条微指令只能完成一个基本操作。

### （一）水平型微指令

水平型微指令是指一次能定义并能并行执行多个微命令的微指令。它的并行操作能力强，效率高，灵活性强，执行一条机器指令所需微指令的数目少，执行时间短；但微指令字较长，同时微指令和机器指令的差别很大，一般用户不易掌握。按直接编码法、字段直接编码法、字段间接编码法编码的微指令都属于水平型微指令。

### （二）垂直型微指令

垂直型微指令的结构类似于机器指令的结构，它有操作码，功能简单，并行操作能力差，但由于微指令与机器指令很相似，所以容易掌握和利用，编程比较简单，且微指令字较短。按最短编译法编码的微指令属于垂直型微指令。

## 四、微地址的形成

微程序是由微指令组成的，执行当前一条微指令时，必须指出后继微指令的地址，以便当前一条微指令执行完毕后，取出下一条微指令。决定后继微指令地址的方法不只一种，主要有两大基本类型：增量方式和断定方式。

### （一）增量方式（计数器法）

这种方式和机器指令的控制方式很类似，它也有顺序执行、转移和转子之分。顺序执行时，由微程序计数器 μPC 负责生成下一个微指令的地址（μPC+1 →μPC），即后继微地址就是现行微地址加上一个增量（通常为 1）；转移或转子时，由微指令的顺序控制字段产生转移微地址。为了降低成本，一般情况下都是将微地址寄存器改为具有计数功能的寄存器，以代替 μPC。

增量方式的优点是简单、易掌握，编制微程序容易。缺点是当转移分支很多时，相应的逻辑电路也更复杂，并影内微程序的执行速度。

### （二）断定方式（下地址字段法）

μPC 为基础所建立的地址生成技术最大的缺点在于，当存在着较大量的转移微指令时，将会严重影响计算机的工作速度。作为一种替代的方法，断定方式是在微指令中设置一个专门的地址字段，称为下地址字段，用以指出下一条微指令的地址或部分地址。

断定方式的后继微地址可由微程序设计者指定，或者根据现行微指令执行所产生的状态特征或条件码的判别结果决定。

该方式的主要缺点是增加了微指令的长度，优点是消除了专门的转移微指令。

# 第六章 I/O 设备和系统

## 第一节 外存储器

CPU 和主存构成了主机，除主机外的大部分硬件设备都可称为外部设备，简称外设。计算机的外部设备主要包括外存储器、输入设备、输出设备，如图 6-1 所示。

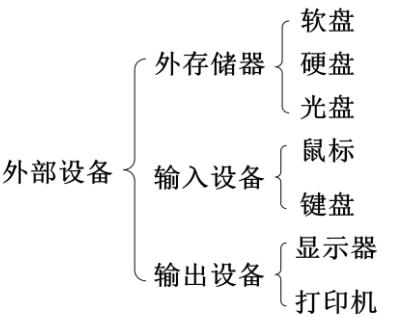



**图 6-1 外部设备组成**

外存储器简称外存或辅存，它与主存一起组成了存储器系统的主存-辅存层次。与主存相比，辅存具有容量大、速度慢、价格低、可脱机保存信息等特点，属“非易失性”存储器。常见的外存储器有软盘、硬盘、磁带、光盘等。前三种均属磁表面存储器。

### 一、磁表面存储器的主要技术指标

#### （一）记录密度

记录密度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。

道密度是磁盘沿半径方向单位长度的磁道数，单位是 tpi（道/英寸）或 tpm(道/毫米)。相邻两条磁道中心线之间的距离称为道距，道密度等于道距的倒数。

位密度是单位长度磁道能记录二进制信息的位数，也称为线密度，单位是 bpi（位/英寸）或 bpm（位/毫米）。

在磁盘各磁道上所记录的信息量是相同的，而位密度不同。一般泛指磁盘位密度时，是指最内圈磁道上的位密度（最大位密度）。

### （二）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。

### （三）平均寻址时间

磁盘的寻道时间是指磁盘的磁头移动到指定磁道所需的时间；磁盘的等待时间是指磁头已处于要访问的磁道，等待所要访问的扇区旋转至磁头下方的时间。因磁盘每次的寻道时间和等待时间并不相同，故取这些时间的平均值并求和，得到磁盘的平均寻址时间，即：平均寻址时间=平均寻道时间+平均等待时间。

平均寻址时间是磁盘存储器的一个重要指标。硬盘的平均寻址时间比软盘的平均寻址时间短，故硬盘比软盘的速度快。

### （四）数据传输率

数据传输率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。

### （五）误码率

误码率是衡量磁表面存储器出错概率的参数，它等于从外存读出时，出错信息位数和读出信息的总位数之比。为减少出错率，磁表面存储器通常采用循环冗余码（CRC）来发现并纠正错误。

## 二、磁表面存储器的记录方式

### （一）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。常用的记录方式有六种，如图 6-2 所示：

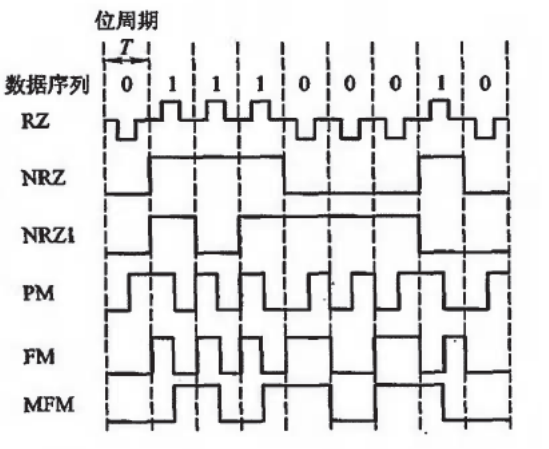



**图 6-2 六种磁记录方式**

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。由于两位信息之间驱动电流归零，故称为归零制记录方式。

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。当连续记“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 l 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“l”时电流改变方向，使磁层磁化方向发生翻转；记录“0”时，电流方向保持不变，使磁层的磁化方向也维持原来状态，因此称为“见 l 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，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 度的磁化翻转方向来表示“1”和“0”。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“l”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。则记录“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盘和软盘中。

6.改进型调频制（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“l”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，改进型调频制只有当连续记录两个或两个以上的“0”时，才在每位的起始处改变一次电流，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，故又称为倍密度记录方式。

## （二）评价指标

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。例如，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。磁表面存储器为了进行读写操作，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，这种方法称为外同步，如 NRZ1 制，但它占用了磁表面存储器的有效记录面积。

对于高密度的记录系统，可直接从磁盘读出的信号中提取同步信号，这种方法称为自同步。

自同步能力可用最小磁化翻转间隔和最大磁化翻转间隔之比值 R 来衡量。如 FM 记录方式的最大磁化翻转间隔是 T（T 为一位信息的记录时间），最小磁化翻转间隔是 T/2，所以 R=0.5。R 越大，自同步能力也越强。

NRZ 和 NRZ1 方式在连续记录“0”时，磁层都不发生磁化磁转，且 NRZ 方式在连续记录“l”时，磁层也不发生磁化翻转，因此，NRZ 和 NRZ1 都没有自同步能力。而 PM、FM、MFM 记录方式均有自同步能力。

## 三、常见的外存储器

### （一）软盘

软盘是个人计算机中最早使用的可移介质，常用的有 3.5 英寸和 2.5 英寸的软盘，容量在 1MB 以上。软盘存取速度慢，容量也小，携带方便，20 世纪八九十年代曾作为外存的主要部件。

### （二）硬盘

硬盘是计算机系统中最主要的外存设备。硬盘的存储介质材料是一种由铝合金材料制成的圆盘，盘的表面涂有一层可被磁化的硬磁特性材料。目前应用最广泛的是温彻斯特磁盘，它是一种可移动磁头固定盘片的磁盘存储器，简称温盘。

硬盘主要由磁盘、磁头及控制电路组成，信息存储在磁盘上，磁头负责读出或写入。当硬盘接到一个系统读取数据指令后磁头根据给出的地址，首先按磁道号产生驱动信号进行定位，然后再通过盘片的转动找到具体的扇区，最后由磁头读取指定位置的信息并传送到硬盘缓存中。

### （三）光盘

光盘是利用光存储技术进行读/写信息的存储设备，主要由光盘、光盘驱动器和光盘控制器组成。主要有 CD 和 DVD 两大类。

### （四）移动存储设备

U 盘是一种采用闪存为存储介质，通过 USB 接口与计算机交换数据的可移动存储设备，具有可多次擦写、体积小、即插即用等特点。

移动硬盘以硬盘为存储介质，通过 USB 接口与计算机相连，且普遍采用了热插拔技术，具有高速、大容量、即插即用、轻巧便捷的特点。

# 第二节 输入输出设备

## 一、输入设备

输入设备是计算机的外部设备之一，是向计算机输送数据的设备。其功能是将计算机的程序、文本、图形、图像、声音以及现场采集的各种数据转换为计算机能处理的数据形式并输送到计算机内部。常见的输入设备有键盘、鼠标、手写笔、扫描仪等。

### （一）鼠标

鼠标是一种输入设备，分有线和无线两种，也是计算机显示系统纵横坐标定位的指示器，因形似老鼠而得名“鼠标”。鼠标的使用是为了使计算机的操作更加简便快捷，来代替键盘繁琐的指令。

### （二）键盘

键盘是用于操作设备运行的一种指令和数据输入装置，是应用最普遍的输入设备。

## 二、输出设备

输出设备的功能是将计算机中的数据信息传送到外部媒介，并转化成某种人们所认识的表示形式。最常用的输出设备有显示器和打印机。

### （一）显示器

显示器通常也被称为监视器，它是一种将一定的电子文件通过特定的传输设备显示到屏幕的显示工具，它是目前计算机系统中应用最广泛的人-机界面设备。

按显示设备所用的显示器分类，有阴极射线管（CRT）显示器、等离子显示器（PDP）、液晶显示器（LCD）、发光二极管显示器（LED）等。

显示器的主要性能指标有：

**1.分辨率**

分辨率是指显示器屏幕能表示的像素点数，分辨率越高，图像越清晰。

2.灰度级

灰度级是指显示像素点相对亮暗的级差，在彩色显示器中它还表现为色彩的差别。

3.刷新频率

刷新频率指的是每分钟内屏幕画面更新的次数。刷新频率过低，可出现屏幕画面闪烁或抖动的情况。

4.点距

点距是指显示器屏幕上相邻像素点之间的距离。点距越小，图像越清晰。

5.屏幕尺寸

屏幕尺寸是指矩形屏幕对角线的长度。

## （二）打印机

打印机是计算机的输出设备之一，用于将计算机处理结果打印在相关介质上，是一种硬拷贝设备。

打印机有多种划分方法。按印字原理划分，有击打式打印机和非击打式打印机。击打式打印机是利用机械动作使印字机构与色带和纸相撞击而打印字符。它又分为活字打印机和点阵针式打印机两种。活字打印机现在用得越来越少。点阵针式打印机利用钢针打击色带产生打印效果，打印成本低，目前仍用得较普遍。非击打式打印机采用电、磁、光、喷墨等物理和化学方法来印刷字符，主要有激光打印机、喷墨打印机等。

按数据传输方式划分，有串行打印机和并行打印机两种，前者是逐字打印，后者是逐行打印。

打印机常见的参数指标主要有以下几个：

1.打印速度

打印速度一般用 PPM 表示，指打印机每分钟可打印的页数；

2.打印分辨率

打印分辨率指打印输出时，在横向和纵向上每英寸最多能够打印的点数，一般用 DPI（点/英寸）表示；

3.最大打印尺寸

最大打印尺寸指打印机所能打印的最大纸张尺寸，一般主要有A4和A3两种规格。

# 第三节 I/O 系统概述

在计算机的硬件系统中，输入输出系统是除了 CPU 和存储器二者之外的第三个关键部分。

## 一、主机与 I/O 设备的联系

### （一） I/O 指令

I/O 指令是机器指令的一类，其一般格式主要由三部分组成：操作码、命令码和设备码。操作码可作为 I/O 指令与其他指令（如访存指令、控制指令等）的判别代码；命令码体现 I/O 设备的具体操作；设备码是 I/O 设备的选择码。

### （二）编址方式

将 I/O 设备码看做地址码，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 地址看做是存储器地址的一部分；不统一编址（或称独立编址）指 I/O 地址和存储器地址是分开的，所有对 I/O 设备的访问必须有专用的 I/O 指令。统一编址占用存储空间，减少了主存容量，但无须专用的 I/O 指令；不统一编址不占用主存空间，不影响主存容量，但需设 I/O 专用指令。

### （三）传送方式

n 位信息同时在 CPU 和 I/O 设备之间进行传输，这种传送方式称为并行传送。其特点是传送速度较快，要求数据线多，如 32 位信息并行传送需要 32 根数据线。

若连续逐位传送信息，这种传送方式称为串行传送。当远距离数据通信时，采用串行传送较为合理。

## 二、I/O 接口

主机和外设具有不同的工作特点，它们在信息形式和工作速度上存在很大的差异，要将各种各样的外部设备与计算机连接起来，解决它们之间的差异，使之能协调地工

作，就要通过各种接口来连接。I/O 接口是指主机与 I/O 设备之间设置的硬件电路及其相应的软件控制。

接口（Interface）与端口（Port）是两个不同的概念。端口是指接口电路中的一些寄存器，这些寄存器分别存放数据信息、控制信息和状态信息，相应的称为数据端口、控制端口和状态端口。若干个端口加上相应的控制逻辑电路才组成接口。

## （一）接口功能

### 1.选址功能

CPU 对于设备的选择，通过设备选择线上的设备码来确定。将设备码送至所有设备的接口，故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，这种功能可通过接口内的设备选择电路来实现。

### 2.传送命令的功能

当 CPU 向 I/O 设备发出命令时，要求 I/O 设备能做出响应，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。

### 3.传送数据的功能

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。接口中通常设有数据缓冲寄存器（Data Buffer Register，DBR），与 I/O 总线中的数据线相连，用来暂存 I/O 设备与主机准备交换的信息。

### 4.反映 I/O 设备工作状态的功能

为使 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。

## （二）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

分为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；串行接口是在设备与接口间一位一位传送。

2.按功能选择的灵活性分类

分为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；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。

3.按通用性分类

分为通用接口和专用接口。通用接口可供多种 I/O 设备使用；专用接口是为某类外设或某种用途专门设计的。

4.按数据传送的控制方式分类

有程序型接口和 DMA 型接口。程序型接口用于连接速度较慢的 I/O 设备，如显示终端、键盘、打印机等；DMA 型接口用于连接高速 I/O 设备，如磁盘、磁带等。

## 三、主机与 I/O 设备信息传送的控制方式

主机和 I/O 设备之间的信息传送的控制方式可以分为以下四种：

### （一）程序查询方式

程序查询方式是主机与 I/O 设备之间进行信息交换的最简单方式，由 CPU 通过程序不断查询 I/O 设备是否已做好准备，从而控制 I/O 设备与主机交换信息。

这种方式控制简单，但 CPU 和 I/O 设备处于串行工作状态，CPU 的工作效率不高，主要适用于 I/O 设备少，数据传输率较低的系统。

### （二）程序中断方式

I/O 设备作好输入/输出准备时，向 CPU 发出中断请求，CPU 接到请求后就暂时中止原来执行的程序，转去执行中断服务程序，从而可以输入/输出一个数据。当中断处理完毕后，CPU 返回原来的程序继续执行。

这种方式实现了 CPU 和 I/O 设备的并行工作，提高了 CPU 的工作效率，一般适用于随机出现的服务。

### （三）直接存储器存取（DMA）方式

DMA 方式是在主存和 I/O 设备之间开辟一条直接的数据通路，可以进行基本上不需要 CPU 介入的信息传送，此时，DMA 控制器从 CPU 完全接管对总线的控制，数据交换不经过 CPU，而直接在内存和 I/O 设备之间进行。

这种方式进一步提高了 CPU 的资源利用率，能满足高速 I/O 设备的需要。

### （四）I/O 通道控制方式

通道的出现进一步提高了 CPU 的效率，这是因为 CPU 将部分权利下放给通道，通道可视为一种具有特殊功能、但不完全独立的处理器。它依据 CPU 的 I/O 指令工作，是从属于 CPU 的一个专用处理器，用来负责管理 I/O 设备、实现主存与 I/O 设备之间交换信息。

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 的工作效率大为提高，但这种提高是以花费更多硬件为代价的。

目前，小型机和微型机大都采用程序查询方式、程序中断方式和 DMA 方式，通道方式一般用在中、大型计算机中。

# 第四节 中断方式

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，继续执行原程序，这就是“中断”。通常把实现中断所需的软硬件技术统称为中断技术。

## 一、中断的提出

引起中断的因素大致可分为以下几类：

### （一）人为设置的中断

人为设置的中断一般称为自愿中断，一旦计算机执行人为中断，便停止现行程序而转入中断处理。

### （二）程序性事故

程序性事故是由程序设计不周而引起的中断，如定点溢出、操作码不能识别等。

### （三）硬件故障

硬件故障有电源掉电、插件接触不良等。

### （四）I/O 设备

I/O 设备启动后，一旦准备就绪，就向 CPU 发出中断请求。

### （五）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。

上述各种中断因素除自愿中断是人为的以外，大多都是随机的。通常将能引起中断的各个因素称为中断源。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；另一类为可屏蔽中断，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。若未屏蔽则可以响应；若已被屏蔽，则 CPU 不能响应。

## 二、中断处理过程

计算机的中断处理过程表面上看起来有点类似于调用子程序的过程，但是它们之间却是有着本质上的区别：首先，子程序的执行是由程序员事先安排好的，而中断服务程序的执行则大多是由随机的中断事件引起的；其次，子程序的执行受到主程序或上层子程序的控制，而中断服务程序一般与被中断的现行程序毫无关系；最后，不存在同时调用多个子程序的情况，而有可能发生多个 I/O 设备同时请求 CPU 为自己服务的情况。

一次中断处理过程可简单地归纳为以下几个阶段：中断请求、中断判优、中断响应、中断服务和中断返回。

### （一）中断请求

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。这种触发器可集中设在 CPU 的中断系统内，组成一个中断请求标记寄存器，也可以分散到各个中断源中，如分散在各个接口电路内。

### （二）中断判优

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，这称为中断判优。各中断源的优先顺序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。

中断判优可用硬件实现，也可用软件实现。

### （三）中断响应

#### 1.中断响应的条件和时间

个中断系统，在任一时刻，只能响应一个中断源的请求。CPU 响应中断源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”。允许中断触发

器 EINT 可用开中断指令置“1”（称为开中断），意味着 CPU 允许响应中断源的请求；也可用关中断指令或硬件自动使其置“0”（称为关中断），意味着 CPU 禁止响应中断。

CPU 响应中断的时间是在每条指令执行阶段的结束时刻。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。若有中断，CPU 进入中断周期；若无中断，则进入下一条指令的取指周期。

2.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 响应中断后，即进入中断周期。在中断周期内，CPU 要自动完成一系列操作：

（1）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 响应后所需做的一系列操作不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。

（2）寻找中断服务程序的入口地址

由于中断周期结束后，会进入下条指令（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。由于不同的中断源对应不同的中断服务程序，故准确找到服务程序的入口地址是中断处理的核心问题。通常有两种方法寻找入口地址：硬件向量法和软件查询法。

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。在中断系统中，将中断服务程序的入口地址称为中断向量。存放中断向量的单元地址称为中断向量地址。硬件向量法寻找入口地址速度快，在现代计算机中被普遍采用。

需要注意的是，某些计算机中，在中断向量地址存放一条跳转到中断服务程序入口地址的跳转指令。

②软件查询法

用软件寻找中断服务程序入口地址的方法称为软件查询法。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。这种方法不涉及硬设备，但查询时间较长。

（3）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内，也可以存入堆栈。

## （四）中断服务

一般中断服务程序的流程分为四大部分：保护现场、中断服务、恢复现场和中断返回。

1.保护现场

保护现场有两个含义，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。

2.中断服务

这是中断服务程序的主体部分，对于不同的中断请求源，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。

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。

## 三、中断屏蔽

### （一）单重中断和多重中断

计算机在处理中断的过程中，有可能出现新的中断请求，若此时 CPU 对新的中断请求不予理睬，这种中断称为单重中断；如果 CPU 暂停现行的中断服务程序，转去处理新的中断请求，这种现象称为中断嵌套，或多重中断。

### （二）实现多重中断的条件

中断系统若要具有处理多重中断的功能，必须具备以下条件：

**1.提前开中断**

CPU 一旦响应了某中断源的中断请求后，便由硬件线路自动关中断，以确保该中断服务程序的顺利执行，即中断允许触发器 EINT 被置“0”，这意味着 CPU 不能再响应其他任何一个中断源的中断请求。

对于单重中断，开中断指令设置在最后“中断返回”之前，意味着在整个中断服务处理过程中，不能再响应其他中断源的请求；对于多重中断，开中断指令提前至“保护现场”之后，意味着在保护现场后，若有级别更高的中断源提出请求，CPU 也可以响应，即再次中断现行的服务程序，转至新的中断服务程序。多重中断“开中断”指令的位置前于单重中断，这是多重中断与单重中断的主要区别。

**2.优级别高的中断源有权中断优先级别低的中断源**

在满足提前设置“开中断”指令的前提下，只有优先级别更高的中断源请求才可以中断比其级别低的中断服务程序。为了保证级别低的中断源不干扰比其级别高的中断源的中断处理过程，可采用屏蔽技术。

### （三）屏蔽技术

接口电路中设有完成触发器 D、中断请求触发器 INTR 和中断屏蔽触发器 MASK：当该中断源被屏蔽（MASK=1），即使准备就绪（D=1），中断查询信号到来时也只能将 INTR 置“0”，CPU 接收不到该中断源的中断请求，即被屏蔽；仅当设备准备就绪（D=1），且该设备未被屏蔽（MASK=0）时，CPU 的中断查询信号可将中断请

求触发器置“1”，表示该中断源向 CPU 提出中断请求，该信号送至排队器进行优先级判断，可见，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，可构成一个屏蔽寄存器，屏蔽寄存器的内容称为屏蔽字。屏蔽字与中断源的优先级别一一对应。

采用屏蔽技术后，可以改变 CPU 处理各中断源的优先等级，还能给程序控制带来更大的灵活性。

### （四）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，可保存在堆栈中，也可保存在特定的存储单元内，由中断隐指令在中断周期内实现。

# 第五节 DMA 方式

DMA 适合于内存与高速 I/O 设备或外存之间的信息交换。DMA 方式中，由于 DMA 接口与 CPU 共享内存，为有效分时使用内存，通常 DMA 与内存交换数据时采用以下三种方法：

## 一、CPU 停止访问内存法

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，直到一组数据传送完毕后，DMA 接口通知 CPU 可以使用内存，并把总线控制权交回给 CPU，如图 6-3 所示。这种方法的优点是控制简单，适用于高速 I/O 设备的数据成组传送。缺点是在 DMA 接口访问内存期间，CPU 基本处于保持或不工作状态，CPU 对主存的利用率得不到充分发挥。

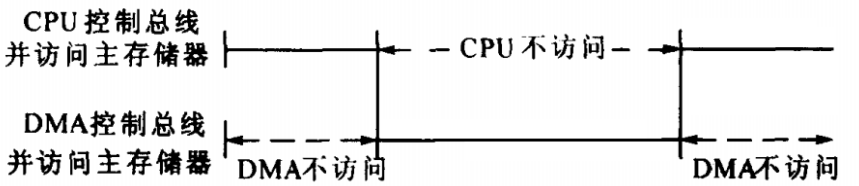



**图 6-3 CPU 停止访问内存法**

## 二、周期挪用法

周期挪用也叫周期窃取，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内存周期，而 DMA 不请求时，CPU 仍继续访问内存，如图 6-4 所示。周期窃取法每次在 DMA 控制器传送完一个数据后立即释放总线，使 I/O 设备在准备下一数据时，CPU 能插空访问主存。

若 I/O 设备请求 DMA 传送时，CPU 不需要访存，则周期挪用对 CPU 执行程序无任何影响；若 CPU 正在访问内存，此时必须待存取周期结束，CPU 才能将总线占有权让出；若同一时刻，发生 CPU 与 DMA 的访存冲突，那么优先保证 DMA 工作，因

为 I/O 设备不立即访问内存就可能丢失数据，这时 I/O 设备要窃取一、二个存取周期，CPU 则延缓一、二个存取周期再访问内存。

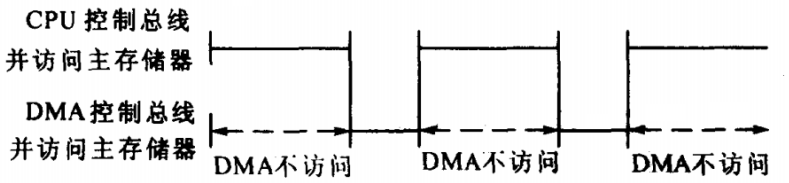



**图 6-4 周期挪用法**

与 CPU 停止访问内存法相比，周期挪用的方法既实现了 I/O 传送，又较好地发挥了内存和 CPU 的效率，是一种广泛采用的方法。

但应该指出，I/O 设备每挪用一个内存周期都要申请总线控制权、建立总线控制权和归还总线控制权。因此，尽管传送一个字对内存而言只占用一个内存周期，但对 DMA 接口而言，实质上要占 2～5 个内存周期。故周期挪用的方法比较适合于 I/O 设备的读/写周期大于内存周期的情况。

## 三、DMA 与 CPU 交替访问

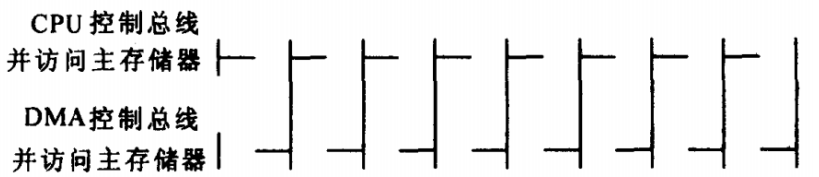



**图 6-5 DMA 与 CPU 交替访问法**

如果 CPU 的工作周期比内存存取周期长很多，此时采用交替访问的方法可以使得 DMA 传送和 CPU 同时发挥最高的效率。如图 6-5 所示，把一个 CPU 周期分成两个时间片，一个给 CPU，一个给 DMA，使 CPU 和 DMA 交替地访问内存。这种方法不需要申请和归还总线，总线控制权的转移几乎不需要什么时间，具有很高的 DMA 传送效率。而且 CPU 既不停止现行程序的运行，也不进入等待状态，不知不觉地就完成了 DMA 的数据传送，故又被称为“透明的 DMA”方式。