清華大學 计算机系列教材

计算机组成与结构（第5版）习题详解、实验和CPU设计指导

王爱英　杨蔚明　编著

清华大学出版社

清華大學 计算机系列教材

王爱英 杨蔚明 编著

计算机组成与结构（第5版）习题详解、实验和CPU设计指导

清华大学出版社
北京

内 容 简 介

本书是配合清华大学系列教材《计算机组成与结构(第5版)》而编写的习题详解、实验和CPU设计指导。全书共分两部分:第一部分是为《计算机组成与结构(第5版)》各章编写的习题集及其参考解答;第二部分是计算机的硬件实验与测试方法,是根据作者多年来的教学和科研工作经验编写的。

本书可作为计算机专业和其他相关专业的计算机原理课程的补充教材或课外阅读资料,也可供计算机硬件的设计、制造和维修工程技术人员阅读和参考。

图书在版编目(CIP)数据

计算机组成与结构(第5版)习题详解、实验和CPU设计指导/王爱英等编著.—北京:清华大学出版社,2014

清华大学计算机系列教材

ISBN 978-7-302-32976-3

Ⅰ.①计…　Ⅱ.②王…　Ⅲ.①计算机体系结构—高等学校—题解　Ⅳ.①TP303-44

中国版本图书馆CIP数据核字(2013)第147764号

责任编辑:白立军　战晓雷
封面设计:常雪影
责任校对:梁　毅
责任印制:杨　艳

出版发行:清华大学出版社
网　　址:http://www.tup.com.cn, http://www.wqbook.com
地　　址:北京清华大学学研大厦A座　　**邮　　编:**100084
社 总 机:010-62770175　　**邮　　购:**010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载:http://www.tup.com.cn,010-62795954
印 装 者:北京嘉实印刷有限公司
经　　销:全国新华书店
开　　本:185mm×260mm　**印　张:**14.25　**字　　数:**356千字
版　　次:2014年1月第1版　**印　　次:**2014年1月第1次印刷
印　　数:1～2000
定　　价:29.00元

产品编号:052403-01

前　言

随着计算机技术的飞速发展，计算机的应用也日益广泛，从航天航空领域到家用电器领域都是如此。

计算机由硬件和软件组成，随着软件（操作系统、系统软件和应用软件）的发展，计算机的使用变得非常容易，从而能在各个行业中推广，而这一切都必须建立在硬件的基础上。

清华大学出版社出版的教材《计算机组成与结构（第5版）》（在本书中简称为教材）对计算机硬件进行了全面论述，本书是为配合该教材的使用而编写的，全书由两部分组成。

第一部分（第1章至第12章）是与教材中每一章相配合的习题及其解答，在内容方面涉及计算机的各个组成部件和整机（从工作原理到具体结构），尤其对教材中各章的重点和难点，通过习题形式予以解答，虽然教材中也有习题和答案，因限于篇幅，只是简略提供，而本书中的习题基本上都给出了详细的解题过程，希望读者注意解题思路。另外有些题（尤其是具有设计性质的题）可以有多种解答方法，读者如能得出不同的题解，甚至得出更便捷的答案则将收获更大。

第二部分（第13章至第15章）涉及计算机实验、设计、测试与应用等内容。要学好“计算机组成与结构”这门课程，除了学好书本知识之外，实践也是很重要的。由于计算机涉及的知识面广，结构又比较复杂，因此通过多种途径加强实践是很重要的，做实验、搞设计，阅读并分析计算机的逻辑设计图，参加计算机的测试与调试等都是很好的实践环节。唯有理论、实践与应用三者结合，才能为创新准备实现条件。

实验是学生在教师和实验室工作人员帮助下进行的，安排适当，可以很快掌握实验技巧，学会仪器的使用方法，增强实际动手能力，为今后参加设计工作、调试工作，进入计算机的核心技术领域打好基础。

计算机是很复杂的设备，在设计阶段就要考虑怎样测试的问题。本部分提出了实验与测试的内容及其对设备的要求。考虑到各校实验设备与实验条件的差异，本部分不与具体的实验设备挂钩。

在这里还要强调一下，实验与测试的方法可以多种多样，在实践中没有最好的方法，但肯定有更好的方法。

本书在清华大学出版社的鼓励和支持下出版了，由于作者的知识水平和工作经验有限，而计算技术又在不断发展之中，所以给出的解答只是一种参考，也有可能存在不妥甚至错误之处，希望得到有识之士的指教。

作　者

2013年11月

目　录

第一部分

习题和解答

第1章　计算机系统概论

1. 什么是硬件？什么是软件？

答：计算机系统是由硬件和软件组成的。硬件是构成计算机系统的实际装置。软件是为便于计算机应用和对硬件进行管理而编写的程序。软件和硬件的交界面为指令系统。

在计算机中能由硬件执行的程序是由指令组成的，因此计算机执行程序的过程，实际上就是按照给定次序执行一组指令的过程。在一条指令中规定了要求计算机进行的操作及其相关信息，例如操作数的来源等。所有指令的集合称为指令系统。在机器硬件内部，指令是用二进制代码表示的。

软件可分为系统软件和应用软件。在系统软件中，操作系统是为管理硬件和人机交换信息而设计的；各种语言（高级语言、汇编语言等）的编译程序是为系统程序员编制系统软件和用户编制应用软件服务的。

2. 说明高级语言、汇编语言和机器语言三者的差别和联系。

答：用二进制代码表示的指令系统称为机器语言。用一串二进制代码（0 和 1）来编制程序是很困难的，容易出错，也不便于检查，于是产生了汇编语言，它利用符号来表示指令。例如，一条加法运算指令可用符号 ADD A，B 来表示，其中 ADD 表示加法操作，A、B 为两个操作数（通常为操作数的地址）。同样可用 SUB 表示减法运算，MUL 表示乘法运算等。汇编语言表示的一条语句基本上与机器语言表示的一条指令相对应，但计算机的硬件只能执行用机器语言编写的程序，因此需要由编译程序将汇编语言程序转换成机器语言程序。

无论用汇编语言还是用机器语言编写程序，都需要对计算机硬件结构和指令系统有深入的理解，这对编程人员的要求是很高的，不易推广。

高级语言程序是由英文字母、数字、符号和计算公式等按照一定的语法规则组成的，编写程序时不需要了解硬件结构和指令系统，高级语言的一条语句的功能比汇编语言（或机器语言）的一条指令的功能强得多，相对来说，编程也比较容易。但是在执行该程序时要先将它翻译成机器语言，翻译工作是由系统程序员预先设计好的编译程序在计算机上运行而自动完成的。

3. 计算机硬件由哪几部分组成？各部分的作用是什么？各部分之间是怎样联系的？

答：组成计算机的基本部件有中央处理部件（CPU）、存储器和输入输出设备。

CPU 主要用于处理数据和控制程序（指令流）的执行，发出执行每条指令所需要的控制信号。

存储器用于存储信息（程序和数据）。存储器芯片中的最小存储单元是 1 位二进制代码（0 或 1）。若干个二进制代码组成 1 个字，字长可以是 8 位、16 位、32 位或 64 位等，由各个机器自行定义。8 位代码组成 1 个字节。存储器的每个字（或字节）都有一个唯一的地址。一般存储器本身没有处理数据的能力，仅能根据 CPU 送来的地址和读写命令取出或写入信息（指令或数据）。

输入设备用来输入原始数据和处理这些数据的程序。输入的内容存放在存储器中，可以通过 CPU 传送，也可以直接从输入设备传送到存储器中，后者由输入设备给出存储器地址和写命令。

输出设备用来输出计算机的处理结果，同样可以由 CPU 或存储器直接送来输出的内容，后者由 CPU 或输出设备给出存储器地址和读命令。

当前的计算机一般都能连接网络，联网设备可视为输入输出设备的一种。

目前，计算机软件极为丰富，人机对话极为简便，计算机与通信设备、移动工具紧密融合。

计算机内各部件之间是通过总线联系的，根据作用的不同，分成地址总线、数据总线和控制总线 3 种，如图 1.1 所示。其中数据总线能双向传送数据。控制总线传送命令和应答信号，各部件之间的控制或应答信号也要相互传送。唯有地址线有所不同，CPU 仅送出地址，存储器仅接收地址，输入输出设备可接收从 CPU 送来的地址或向存储器送出地址。

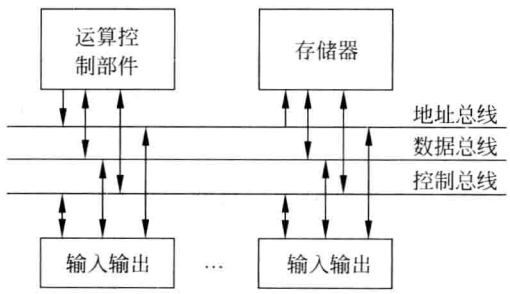


图 1.1　以总线连接的计算机框图

4. 计算机系统可分为哪几个层次？

答：如果用户用高级语言编写程序，该系统的层次结构如图 1.2 所示。但是应用程序并不一定要使用高级语言，则实际运行时的层次结构将有所变化。

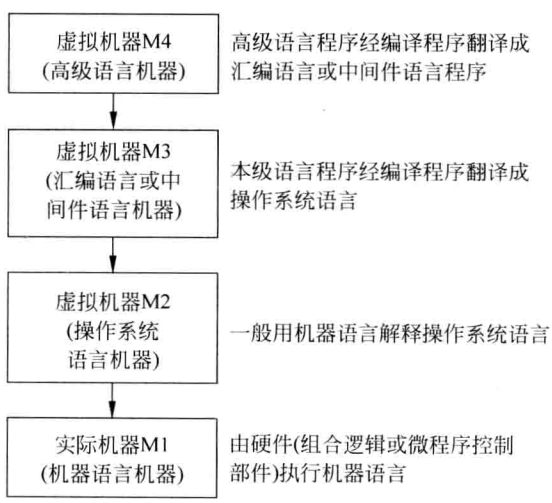


图 1.2　计算机系统的多级层次结构

5. 如何划分计算机发展的 5 个阶段（第一代到第五代）？当前广泛应用的计算机主要采用哪种电路？

答：可以根据计算机中使用的物理器件划分其发展的5个阶段。第一代主要使用电子管器件，第二代主要使用晶体管器件，第三代主要使用集成电路器件，第四代主要使用大规模集成电路器件。根据摩尔定律，集成电路的集成度每隔18个月翻一番。当前广泛应用的计算机主要使用超大规模集成电路，被人们认为是第五代计算机。

6. 试用汇编语言编写求5个数的平均值程序（自定指令系统）。

答：假设计算机至少有3个寄存器（R_1、R_2 和 R_3）。R_1 存放被加数的个数，其初始值为5。R_2 用作变址寄存器，5个数据（a、b、c、d、e）顺序存放在存储器中以DATA为起始地址的存储单元中。R_3 为累加器，存放运算的中间结果和最后的累加和。

程序的流程图如图1.3所示。图中用 R_1 表示寄存器地址，(R_1) 为 R_1 中的内容，DATA为存储器地址，(DATA)为DATA地址中的内容，其余类推。程序循环执行，第1次的加数a从DATA地址中取出，第2次的加数b从DATA+1地址中取出……每执行一次加法运算，变址寄存器的内容+1，循环次数 $(R_1)-1$，当 $(R_1)=0$ 时，5个数相加完毕，除以5即得平均值。

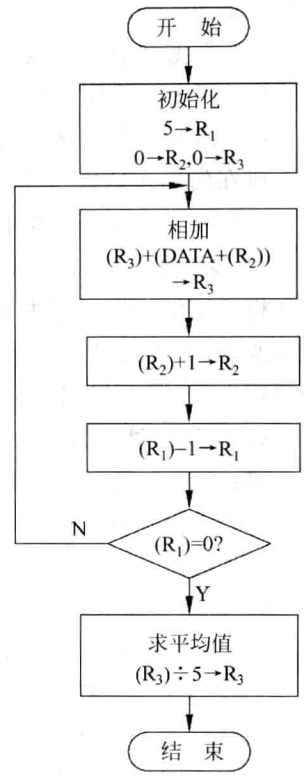


图1.3 求平均值的程序流程

用汇编语言编写的程序如下：

```
START:  MOV   R1,5              ;将常数5传送到R1
        CLR   R2                ;R2清0(0→R2)
        CLR   R3                ;R3清0(0→R3)
LOOP:   ADD   R3,DATA+(R2)      ;R3中为累加和
        INC   R2                ;(R2)+1→R2
        DEC   R1                ;(R1)-1→R1,根据结果置状态位Z,如果结果(R1)为0,Z=1,
                                ;否则Z=0
        JMPZ̄  LOOP              ;(R1)≠0,转移到LOOP;(R1)=0,顺序执行下一条指令
        DIV   R3,5              ;(R3)/5→R3,R3中为平均值
        HLT                     ;停机
DATA:   a                       ;数据
        b
        c
        d
        e
```

START、LOOP和DATA是用符号表示的存储器地址。符号“;”后面的内容为注释。

7. 冯·诺依曼机结构的特点是什么？近年来称为冯·诺依曼型的计算机有哪些主要变化？

答：一般认为冯·诺依曼机具有以下基本特点：

(1) 计算机由运算器、控制器、存储器、输入设备和输出设备5部分组成。

(2) 采用存储程序的方式，程序和数据放在同一个存储器中，指令和数据一样可以送到运算器运算，即由指令组成的程序是可以修改的。

(3) 数据和指令都以二进制码表示。

(4) 指令由操作码和地址码组成。

(5) 指令在存储器中按执行顺序存放，由指令计数器(即程序计数器 PC)指明要执行的指令所在的单元地址，一般按顺序递增，但可按运算结果或外界条件而改变。

(6) 机器以运算器为中心，输入输出设备与存储器间的数据传送都通过运算器。

60 多年来，随着技术的发展和新应用领域的开拓，对冯·诺依曼型机做了很多改革，使计算机系统结构有了很大发展，例如程序与数据分开存放在不同的存储器区域中，程序不允许修改，机器不再以运算器为中心，而是以存储器为中心等。虽然有以上这些突破，但原则变化不大，习惯上仍称之为冯·诺依曼型机。

8. 科学计算与数据处理有何差别？

答：科学计算的特点是数据值的变化范围大和计算工作量大，科学计算用的计算机一般设置有浮点运算指令。

数据处理的特点是输入输出的数据量大，但计算比较简单。

9. 通用计算机和工业控制计算机有何不同？

答：通用计算机一般用于科学计算、数据处理和计算机辅助设计等领域。

工业控制计算机一般用于工厂中的生产过程自动控制，通常都是实时控制，它们对计算机的速度要求不高，但可靠性要求很高。

用于控制的计算机，其输入信息往往是电压、温度、机械位置等模拟量，要先将它们转换成数字量，称为模数转换，然后计算机才能进行处理或计算。当从被控制对象测量到的信息是温度、位置等非电量时，要先将它们转换成电量，然后再转换成数字量。如何测量，用什么仪表测量也是一个很重要的问题。计算机的处理结果是数字量，一般要将它们转换成模拟量去控制对象，称为数模转换。如有需要，可将结果打印输出或显示在屏幕上，以供观察。提供计算机控制系统的厂家往往已将控制程序(称为应用程序包)编制好，可提供给用户。

10. Internet(互联网)采用什么协议？

答：互联网使用 TCP/IP 协议。TCP 为传输控制协议。IP 提供网间地址，在网络中心的统一管理下，对网络上的主机分配地址，保证一个地址对应一台网际主机，叫做 IP 地址。

11. 请解释网络中传输信息的两种方式：基带传输和宽带传输。

答：计算机中的信息以 0、1 两种方式出现，用以表示的电脉冲信号呈现方波形式，称为基带信号。在传输距离较近时，计算机网络系统都采用基带传输方式(例如以太网等局域网)。而在远程传输中，特别是通过无线信道或光信道进行的数据传输中，数字基带信号必须经过高频调制后才能在信道中进行传输，称之为数字基带信号的载波传输或宽带传输，在数字通信系统中大多采用载波调幅传输方式，有载波输出表示 1，无载波输出表示 0，如图 1.4 所示。

经过调制的数字信号在传输到接收端之后要通过滤波器滤掉相应的载波信号、噪声与干扰，从而恢复原来的基带信号，这一过程称为解调，其相应的设备称为解调器。将基带信号变换为调制信号的设备称为调制器。由于计算机通信时，既要接收信号，也要发送信号，

因此需要具有调制和解调功能的设备，即调制解调器。

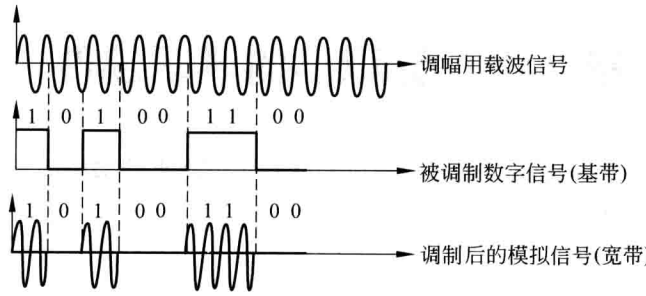


图 1.4　载波调幅信号

第 2 章　计算机的逻辑部件

1. 将下列逻辑表达式化简。化简方法可参阅数字逻辑电路书中的基础部分。

(1) $F=(A+B)(A+\overline{B})$

(2) $F=AB+\overline{A}C+BC$

(3) $F=AB+\overline{A}C+BCD$

(4) $F=AB+\overline{B}D+(A+B)(A+\overline{B})(B+E)$

答：(1) 公式化简：$F=(A+B)(A+\overline{B})=A\cdot A+A\overline{B}+AB+B\overline{B}=A(1+\overline{B}+B)=A$

或用卡诺图化简：2 变量的卡诺图如图 2.1 所示。同时满足 $A+B$ 和 $A+\overline{B}$ 的是图中的重叠部分 A，所以 $(A+B)(A+\overline{B})=A$。

(2) 公式化简：$F=AB+\overline{A}C+BC=AB+\overline{A}C+BC(A+\overline{A})=AB(1+C)+\overline{A}C(1+B)$
$=AB+\overline{A}C$

或用卡诺图化简：3 变量的卡诺图如图 2.2 所示，BC 包含在 AB 和 $\overline{A}C$ 中，可消去，所以 $AB+\overline{A}C+BC=AB+\overline{A}C$。

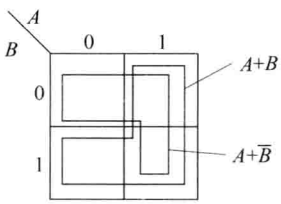


图 2.1　卡诺图(2 变量)

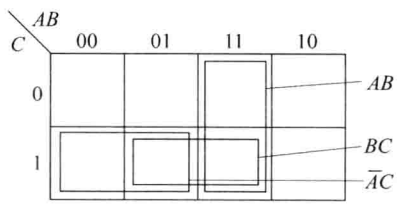


图 2.2　卡诺图(3 变量)

(3) $F=AB+\overline{A}C+BCD=AB+\overline{A}C+BCD(A+\overline{A})$
$=(AB+ABCD)+(\overline{A}C+\overline{A}CBD)=AB+\overline{A}C$

(4) $F=AB+\overline{B}D+(A+B)(A+\overline{B})(B+E)=AB+\overline{B}D+(A+A\overline{B}+AB)(B+E)$
$=AB+\overline{B}D+A(B+E)=AB+\overline{B}D+AE$

2. 将下列逻辑表达式化简，并画出(1)的逻辑图。

(1) $F=A\overline{B}+A\overline{C}D+\overline{A}C+B\overline{C}$

(2) $F=AC+\overline{A}\overline{B}+\overline{B}\overline{C}\overline{D}+B\overline{C}E+\overline{C}DE$

答：(1) $F=A\overline{B}+A\overline{C}D+\overline{A}C+B\overline{C}=A\overline{B}+A\overline{C}D(B+\overline{B})+\overline{A}C+B\overline{C}$
$=A\overline{B}+A\overline{B}\overline{C}D+AB\overline{C}D+\overline{A}C+B\overline{C}=A\overline{B}(1+\overline{C}D)+B\overline{C}(AD+1)+\overline{A}C$
$=A\overline{B}+B\overline{C}+\overline{A}C$

或用卡诺图化简：4 变量卡诺图如图 2.3 所示。其中 $A\overline{C}D$ 包含在 $A\overline{B}$ 和 $B\overline{C}$ 中，可消去。所以 $A\overline{B}+A\overline{C}D+\overline{A}C+B\overline{C}=A\overline{B}+\overline{A}C+B\overline{C}$。

其逻辑图如图 2.4 所示。

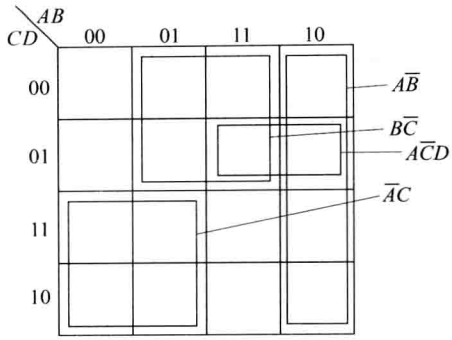


图 2.3　卡诺图(4 变量)

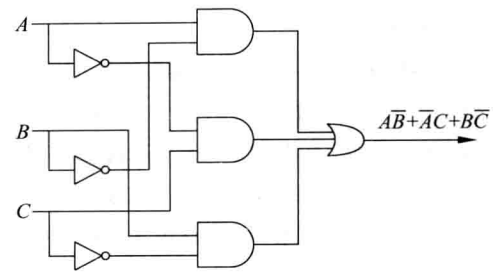


图 2.4　逻辑图

也可以将公式变换如下：

$$
\begin{aligned}
A\overline{B}+B\overline{C}+\overline{A}C &= \overline{\overline{A\overline{B}+B\overline{C}+\overline{A}C}}=\overline{(\overline{A\overline{B}})(\overline{B\overline{C}})(\overline{\overline{A}C})} \\
&= \overline{(\overline{A}+B)(\overline{B}+C)(A+\overline{C})} \\
&= \overline{(\overline{A}\overline{B}+\overline{A}C+BC)(A+\overline{C})} \\
&= \overline{\overline{A}\overline{B}\overline{C}+ABC}
\end{aligned}
$$

其逻辑图如图 2.5 所示。

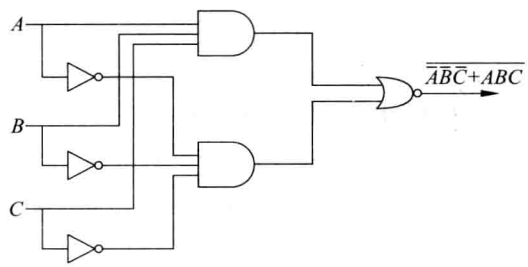


图 2.5　逻辑图

(2) 用卡诺图化简公式：$F=AC+\overline{A}\overline{B}+\overline{B}\overline{C}\overline{D}+B\overline{C}E+\overline{C}DE$

5 变量卡诺图如图 2.6 所示，图中 AC、$\overline{A}\overline{B}$ 和 $B\overline{C}E$ 已用文字标出，$\overline{B}\overline{C}\overline{D}$ 由 00，10，04，14 小块构成，$\overline{C}DE$ 由 30，32，36，34 小块构成。

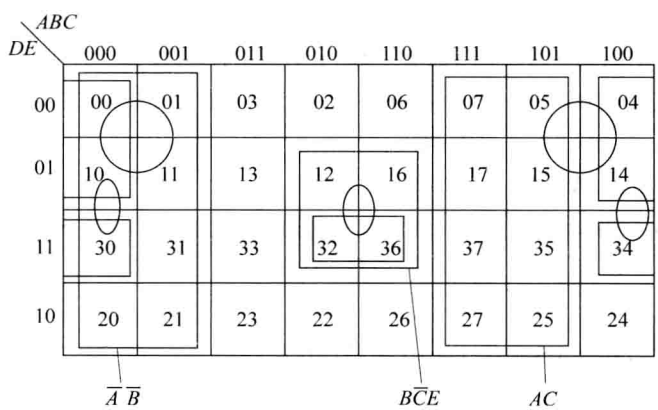


图 2.6　卡诺图(5 变量)

化简后，得 $F=AC+\overline{A}\overline{B}+\overline{B}\overline{D}+\overline{C}E$，其中 AC 和 $\overline{A}\overline{B}$ 从原公式中保留下来，$\overline{B}\overline{D}$ 由卡诺图中的 00，01，10，11 和 05，04，15，14 组成（用圆圈表示），$\overline{C}E$ 由 10，30，12，16，32，36，14，34 组成（用椭圆表示）。

注：在图 2.6 中，每一小块的编号由 2 位构成，高位由 DE 构成，低位由 ABC 构成。例如，第 25 小块的 DE 为 10，以 2 表示；ABC 为 101，以 5 表示。

3. 用布尔代数的基本公式和规则证明下列等式成立。

(1) $A\overline{B}+BD+\overline{A}D+CD=A\overline{B}+D$

(2) $AB\overline{D}+A\overline{B}\overline{D}+AB\overline{C}=A\overline{D}+AB\overline{C}$

(3) $BC+D+\overline{D}(\overline{B}+\overline{C})(DA+B)=B+D$

答：(1)
$$\begin{aligned}A\overline{B}+BD+\overline{A}D+CD&=A\overline{B}(D+\overline{D})+BD(A+\overline{A})+\overline{A}D(B+\overline{B})+CD\\&=(A\overline{B}D+A\overline{B}\overline{D})+(ABD+\overline{A}BD)\\&\quad+(\overline{A}BD+\overline{A}\overline{B}D)+CD\\&=(A\overline{B}D+ABD+\overline{A}BD+\overline{A}\overline{B}D)+A\overline{B}\overline{D}+\overline{A}BD+CD\\&=1\cdot D+A\overline{B}\overline{D}+\overline{A}BD+CD\\&=D(1+\overline{A}B+C)+A\overline{B}\overline{D}\\&=D+A\overline{B}\overline{D}\\&=D+A\overline{B}\quad(\text{根据公式 } X+\overline{X}Y=X+Y)\\&=A\overline{B}+D\end{aligned}$$

(2) $AB\overline{D}+A\overline{B}\overline{D}+AB\overline{C}=A\overline{D}(B+\overline{B})+AB\overline{C}=A\overline{D}+AB\overline{C}$

(3)
$$\begin{aligned}BC+D+\overline{D}(\overline{B}+\overline{C})(DA+B)&=BC+D+(\overline{B}+\overline{C})(DA+B)\\&=BC+D+\overline{B}DA+\overline{C}DA+\overline{C}B\\&=(C+\overline{C})B+(\overline{B}A+\overline{C}A)D+D\\&=B+D\end{aligned}$$

4. 今有一全加器，其输入为 A、B、C，其输出为 S（和）、C'（进位），今输入为反码 $\overline{A}$、$\overline{B}$、$\overline{C}$，问此全加器是否能用？该输出是什么？

答：全加器能用，但输出是 S 的反码 $\overline{S}$ 和 C' 的反码 $\overline{C'}$。证明如下：

根据全加器功能 $S=A\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}\overline{B}C+ABC$。将其变换，

$$\begin{aligned}\overline{S}&=\overline{A\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}\overline{B}C+ABC}\\&=(\overline{A\overline{B}\overline{C}+\overline{A}B\overline{C}})(\overline{\overline{A}\overline{B}C+ABC})\\&=((\overline{A}+B+C)(A+\overline{B}+C))((A+B+\overline{C})(\overline{A}+\overline{B}+\overline{C}))\\&=(\overline{A}\overline{B}+\overline{A}C+AB+BC+AC+\overline{B}C+C)(A\overline{B}+A\overline{C}+\overline{A}B+B\overline{C}+\overline{A}\overline{C}+\overline{B}\overline{C}+\overline{C})\\&=(\overline{A}\overline{B}+AB+C)(A\overline{B}+\overline{A}B+\overline{C})\\&=\overline{A}\overline{B}\overline{C}+AB\overline{C}+A\overline{B}C+\overline{A}BC\end{aligned}$$

说明：当 A、B、C 用反码 $\overline{A}$、$\overline{B}$、$\overline{C}$ 输入时，通过全加器得到的是 $\overline{S}$。

$$C'=AB+BC+AC$$

将其变换为

$$\begin{aligned}\overline{C'}&=\overline{AB+BC+AC}=(\overline{A}+\overline{B})(\overline{B}+\overline{C})(\overline{A}+\overline{C})\\&=(\overline{A}\overline{B}+\overline{A}\overline{C}+\overline{B})(\overline{A}+\overline{C})\\&=\overline{A}\overline{B}+\overline{A}\overline{B}\overline{C}+\overline{A}\overline{C}+\overline{A}\overline{C}+\overline{A}\overline{B}+\overline{B}\overline{C}\end{aligned}$$

$=\overline{A}\overline{B}+\overline{A}\overline{C}+\overline{B}\overline{C}$

说明：当 A、B、C 用反码输入时，通过全加器得到的进位是$\overline{C'}$。

也可用表 2.1 的真值表来验证。

表 2.1　和(S)与进位(C')的真值表

输入				输出			
A	B	C	$\overline{A}$ $\overline{B}$ $\overline{C}$	S	C'	$\overline{S}$	$\overline{C'}$
0	0	0	1 1 1	0	0	1	1
0	0	1	1 1 0	1	0	0	1
0	1	0	1 0 1	1	0	0	1
0	1	1	1 0 0	0	1	1	0
1	0	0	0 1 1	1	0	0	1
1	0	1	0 1 0	0	1	1	0
1	1	0	0 0 1	0	1	1	0
1	1	1	0 0 0	1	1	0	0

5. 今有 8 位带符号整数的加法器，实现输入 $A=A_7A_6\cdots A_0$ 与 $B=B_7B_6\cdots B_0$ 之间的加法运算，结果为 $F=F_7F_6\cdots F_0$，其中 A_7、B_7 和 F_7 为符号位。试写出判断溢出的逻辑表达式。

答：根据计算机中数值的表示方法，符号位为 0，表示正数，符号位为 1，表示负数，符号位参与运算。

(1) 加法运算：当相加两数皆为正数或皆为负数时有可能溢出(数值超出计算机所能表示的范围称为溢出)。如果结果的符号位与加数/被加数不同为溢出。即溢出条件为 $\overline{A}_7\overline{B}_7F_7+A_7B_7\overline{F}_7$。符号“1”表示“或”。

(2) 减法运算($A-B$)：减数与被减数符号不同才可能溢出。如果 A 为正数，B 为负数，结果 F 为正数时不溢出；如果 A 为负数，B 为正数，结果 F 为负数时不溢出，否则为溢出。所以溢出条件为 $\overline{A}_7B_7F_7+A_7\overline{B}_7\overline{F}_7$。

判断是否溢出还可以有其他方法。选择的原则是：①电路数量少；②产生结果的延迟时间短。在不能兼顾时，根据实际情况折中选择之。

6. 设计用若干个全加器和若干个与门实现的 8421 码十进制加法器单元电路，运算结果保存在 D 触发器(寄存器)中。如果全加器的输出不用触发器保存是否可以？

答：采用 8421 码的十进制数，当运算结果≥10(十进制)时应产生进位信号，设 $A_3\sim A_0$ 和 $B_3\sim B_0$ 是参加运算的两个数，其中 A_3 和 B_3 是高位，$F_3\sim F_0$ 是全加器的加法运算结果，$C_4\sim C_1$ 是全加器的进位输出(见图 2.7)，那么 8421 码向高位进位的条件是 $C_4+F_3(F_2+F_1)$，其中 C_4 是运算结果≥16 时的进位信号，$F_3(F_2+F_1)$是运算结果在 10～15 之间时的进位信号。有关 8421 码的十进制运算规则参见教材，本题也可以在学习了有关内容时再做。当产生进位信号时，本位数据应该加 6，即二进制的 0110。所以十进制加法分两步进行，步骤如下：

① 执行 $A+B$，将 F 保存在寄存器中，其内容在图 2.7 中为 $F_3'\sim F_0'$。

② 如果上次加法结果有进位，即 $C_4+F_3(F_2+F_1)=1$，再进行一次+6 运算，将相加结果(最后的结果)再次保存到寄存器，其内容即为 8421 码的十进制数的运算结果。

如果上次加法结果无进位，则不再进行＋6 操作。

十进制加法器单元电路如图 2.7 所示。图 2.8 是外部送来的控制信号时序图。如果时钟 CP 直接加在 D 触发器的脉冲输入端，则触发器的电位输入端逻辑图将按图 2.18（见第 12 题的(1)）修改。

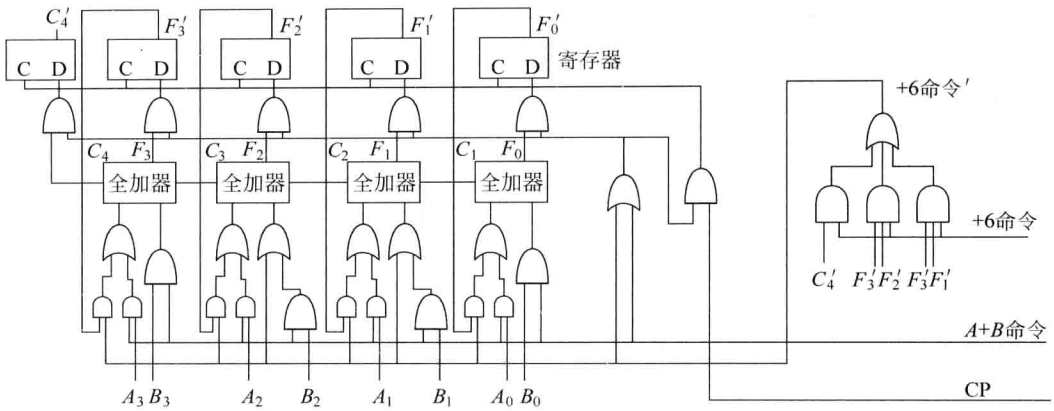


图 2.7　十进制加法器

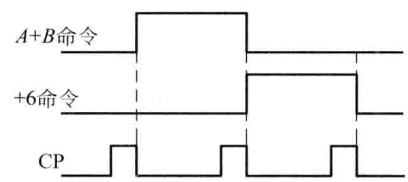


图 2.8　加法器的控制信号

第 1 次执行 $A+B$ 运算的结果与进位保存在寄存器中，(其输出用 $F_3'F_2'F_1'F_0'$ 和 C_4' 来表示)，以保证执行＋6 运算时，＋6 命令′的完整性，否则由于全加器无记忆功能，其输出随输入而变化，而造成＋6 命令′信号的不稳定。

7. 在 4 位超前进位加法器中，得到进位公式如下：

$$C_1=G_1+P_1C_0$$

$$\vdots$$

$$C_4=G_4+P_4G_3+P_4P_3G_2+P_4P_3P_2G_1+P_4P_3P_2P_1G_0$$

其中 $G_i=A_iB_i$，$P_i=A_i+B_i$，$i=1\sim4$，A、B 为相加两数。

请回答：

(1) 产生进位信号需要几个门的延迟时间？

(2) 如果改为 8 位超前进位加法器，按照上述思路，写出 C_5 的公式。

(3) 如果与门和或门的输入端不得超过 5 个，其允许连接的负载也不得超过 5 个，实现 C_5 时将产生什么问题？如何解决问题？

答：(1) 按照本题给出的公式，产生 G_i 和 P_i 需 1 个门的延迟时间，再经过一个与门和 1 个或门产生 C_i，所以总共为 3 个门延迟时间。

(2) C_5 的进位公式如下：

$$C_5=G_5+P_5G_4+P_5P_4G_3+P_5P_4P_3G_2+P_5P_4P_3P_2G_1+P_5P_4P_3P_2P_1G_0$$

(3) 在 C_5 公式中，最后一项组成的与门需要 6 个输入端，而整个等式的右边由 6 项组成，即或门的输入端也需要 6 个，这显然不满足要求。

解决问题的最简单方法是增加 1 级与门和 1 级或门。令 $P_5P_4P_3P_2P_1G_0=(P_5P_4P_3)(P_2P_1G_0)$，第 1 级为 3 输入与门，第 2 级为 2 输入与门，如图 2.9(a)所示。将 C_5 公式的前 3 项和后 3 项分别组成 3 输入端的或门，随后再送入第 2 级或门，如图 2.9(b)所示。形成 C_5 共需 5 个门延迟时间。

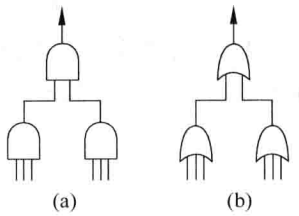


图 2.9　2 级与门和 2 级或门

在 C_5 公式中，负载最重的信号是 P_5，但也没有超过 5 个，所以符合要求。但如果列出 C_6 的公式，P_6 的负载为 6 个，将超过 5 个，为分散负载，也可采取增加 1 级门的办法，这样又会增加延时。

上述解决办法还有一个缺点，就是产生各个进位信号(C_i)的电路各不相同，不利于集成电路的实现，而且位数越高，所需的进位电路也越多。如果将数据分组，小组内采用超前进位的方法，组间采用快速进位的方法，这样各个小组的电路就相同了。

8. 某计算机的 ALU 部件字长 32 位，采用下述几种结构。请分别计算 ALU 部件完成一次加法运算所需要的时间。

(1) 32 位加法器，位间串行进位，假设加法器完成加法时间和进位时间均为 20ns。

(2) 32 位加法器，分成 8 组，每组 4 位，组内超前进位，组间串行进位。加法时间为 30ns，进位时间为 20ns。

(3) 32 位加法器，分成 8 组，每组 4 位，组内组间均为超前进位，加法时间为 30ns，进位时间(小组进位和组间进位)各为 20ns。

答：完成加法运算所需的时间除了与硬件有关外，还与参加运算的数据有关，一般考虑实现 111…11＋000…01 运算的时间为最长的加法运算时间。

(1) 31 个进位时间＋1 次加法时间＝31×20＋20＝640ns。

(2) 7 个小组串行进位时间＋1 次加法时间＝7×20＋30＝170ns。如果详细分析，请参考教材第 2 章中的四位 ALU 逻辑图。当 A 与 B 进入全加器后，首先要形成 P、G 信号，所以由最低位组成的小组(第 1 小组)，其进位信号的形成时间，要比其后面小组(高位)所需的进位延时要长，因为当第 1 小组的进位输出传送到后面小组时，后面小组的相应的 P、G 信号早已形成，所以第 1 小组的进位时间会＞20ns。同理，当最高位进行加法运算时，由于 P、G 信号早已形成，所以加法时间会＜30ns，相互补偿，对计算结果影响不大。

(3) 加法器的逻辑图如图 2.10 所示。

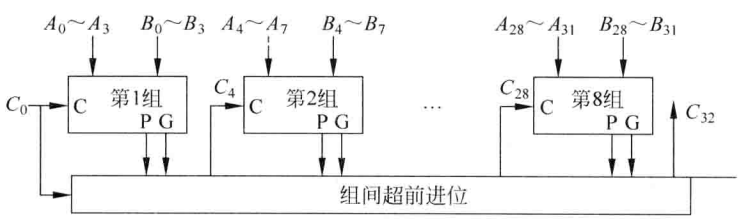


图 2.10　32 位超前进位加法器

加法运算时间＝形成 P、G 信号时间＋组间进位时间＋加法时间≤20ns＋20ns＋30ns≤70ns。(形成 P、G 的时间≤20ns。)

9. 今有两个逻辑电路，如图 2.11(a)和图 2.11(b)所示，其连接是否正确？完成什么功能？是否有允许将门电路直接连接的电路？

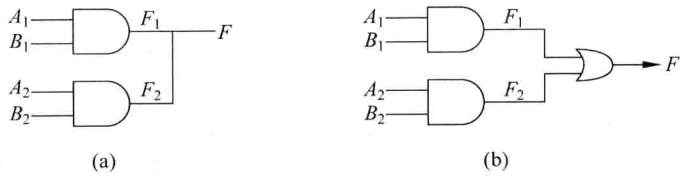


图 2.11　判断是否正确的逻辑电路

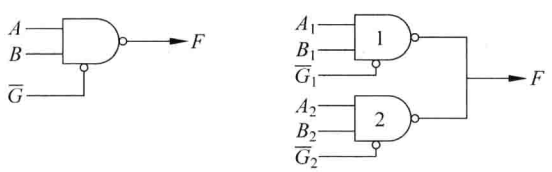
答：图 2.11(a)的连接图是错误的，图 2.11(b) 的连接图是正确的。$F=A_1B_1+A_2B_2$。

一般门电路的输出是不允许直接连接的。在图 2.11(a)中，当两门不连接时，F_1 输出为 1，F_2 输出为0 时(或 $F_1=0$，$F_2=1$)，连接后 F 的状态是不稳定的，而且有可能会因为输出级电路的电流过大，而损坏器件。

采用三态门或集极开路(OC)门，可将多个门电路的输出直接连接在一起，但两者不能混用。

三态门电路有 3 种输出状态："1"、"0"和高阻抗。例如，图 2.12(a)为三态与非门，当 $G=1(\overline{G}=0)$时为正常状态，$F=\overline{AB}$；当 $G=0(\overline{G}=1)$时 F 为高阻抗。图 2.12(b)是连接两个三态门的逻辑电路，当 $\overline{G}_1$ 为 0、$\overline{G}_2$ 为 1 时 $F=\overline{A_1B_1}$，门 2 输出为高阻抗，反之，$F=\overline{A_2B_2}$，门 1 输出为高阻抗，当 $\overline{G}_1$、$\overline{G}_2$ 同时为 1 时，F 为高阻抗，$\overline{G}_1$ 和 $\overline{G}_2$ 同时为 0 的情况是不允许出现的。当有 n 个($n>2$)三态门连接在一起时，只允许其中一个门的 $\overline{G}$ 处于 0 状态(其余门的 $\overline{G}$ 均为 1)。所以当 $\overline{G}$ 的状态变化时，要保证电路中各个 $\overline{G}$ 的正跳变比负跳变先到达。

集极开路(OC)电路的输出通过一个公用电阻(在芯片外)连接到电源上。当连接两个 OC 与非门时，其布尔表达式为 $F=\overline{A_1B_1}\cdot\overline{A_2B_2}=\overline{A_1B_1+A_2B_2}$，其逻辑图如图 2.13 所示，其连接点的逻辑状态是两个门输出状态的"与"("线与"连接)。



(a) 三态与非门　　(b) 连接两个三态门的逻辑电路

图 2.12　三态门

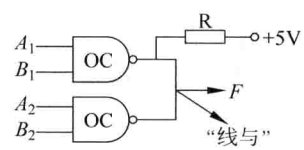


图 2.13　集极开路(OC)门的连接

三态门的信号传输速度高于集极开路门。

注：上述三种门功能的差异主要是由芯片内的输出级电路决定的。

10. 把 4 个信号(A、B、C、D)中的 1 个送到某一输出线上，可使用 4 选 1 多路选择器(与-或门)，也可使用三态门，请画出逻辑图，它们对开门信号的要求有什么差别？

答：4 选 1 多路选择器的逻辑图如图 2.14 所示。选择信号由译码器输出，译码器输入

S_0S_1 的值分别表示选择 4 路数据中的某 1 路。如果输入 S_0 和 S_1 之间的信号有延迟，或者译码器内部电路的不一致性，译码器的输出端会产生一些毛刺(尖脉冲)，但时间很短，很快会稳定，一般不影响使用。如果需要去除毛刺，可以在译码器上加入译码允许信号 $\overline{E}$，当 $\overline{E}$ 为高电位时，译码器的 4 个输出均为低电位。

三态门电路的逻辑图如图 2.15 所示。译码器的某一输出为低电位时选中三态门。要求三态门的开门信号不能同时有 2 个或 2 个以上处于低电平。

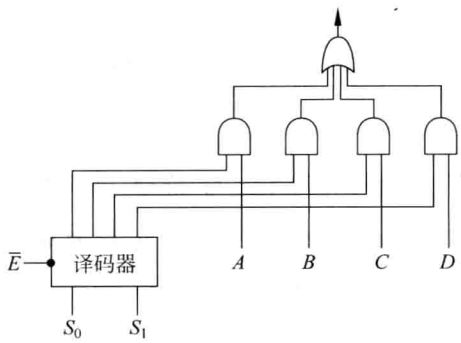


图 2.14　与或门多路选择器

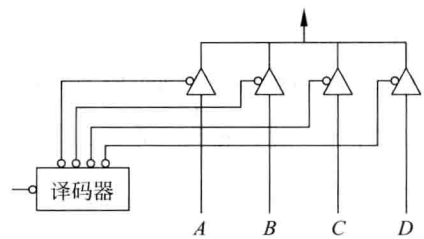


图 2.15　三态门多路选择器

11. 设 A 为锁存器，B 为 D 触发器，输入信号和触发信号的关系如图 2.16 所示，画出输出端波形(设 A、B 的初始状态均为 0)。

答：输出波形如图 2.17 所示。

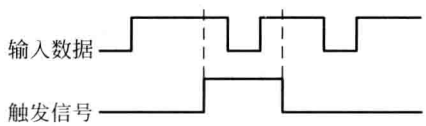


图 2.16　输入数据和触发信号的时序

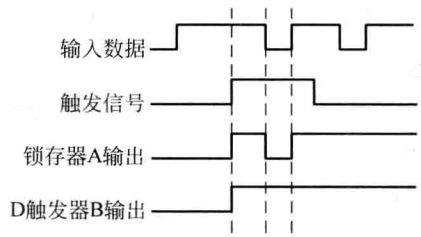


图 2.17　锁存器和 D 触发器的输出波形

锁存器的特点：当触发信号为高电位时，输出随输入数据而变；当触发信号为低电平时，输出保持不变。

D 触发器的特点：触发器在触发信号的上升沿时接收输入数据。

12. 假设触发器的脉冲输入端为时钟，请画出满足下列要求的逻辑图。

(1) D 触发器在不接收数据时保持原状态。

(2) JK 触发器处于计数状态。

答：(1) D 触发器的逻辑图如图 2.18 所示。当控制输入信号为高电位时，在 CLK 作用下 D 触发器接收输入数据；当控制输入信号为低电位时，D 触发器保持状态不变。

(2) JK 触发器的 J 和 K 输入端都接高电位时，在 CLK 脉冲作用下计数。其逻辑图如图 2.19 所示。

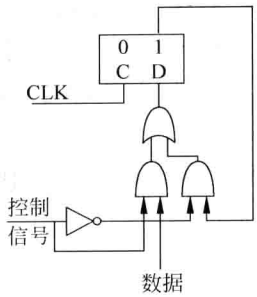


图 2.18　D 触发器(接收与保持)

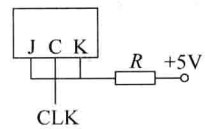


图 2.19　JK 触发器(计数)

13. 填充题。

由“与”阵列和“或”阵列组成的阵列逻辑电路有[A]。其第 1 级是“与”阵列,第 2 级是“或”阵列。其中“与”阵列和“或”阵列均可由用户编程的阵列逻辑电路是[B]。“与”阵列不可由用户编程而“或”阵列可由用户编程的阵列逻辑电路是[C]。“与”阵列可编程而“或”阵列不可编程的是[D]。在上述电路中,除了[E]可多次擦写、重复编程外,其余的均仅能编程一次。

答:A——只读存储器 ROM、可编程逻辑阵列 PLA、可编程阵列逻辑 PAL 和通用阵列逻辑 GAL;B——PLA;C——ROM;D——PAL 和 GAL;E——GAL。

14. 填充题。

在门阵列(GA)、宏单元阵列(MA)和标准单元阵列(SCA)3 种阵列逻辑电路中,集成度由高到低排列次序是[A]、[B]和[C]。其中[D]是半定制器件,[E]是全定制器件。

答:A——标准单元阵列(SCA);B——宏单元阵列(MA);C——门阵列(GA);D——宏单元阵列和门阵列;E——标准单元阵列。

15. 用 PLA 和一个 4D 寄存器组成 4 位双向移位寄存器。要求:当控制信号 $M=0$ 时左移,$M=1$ 时右移。左移时最低位输入为 D_L,右移时最高位输入为 D_R。

答:4 位双向移位寄存器如图 2.20 所示。

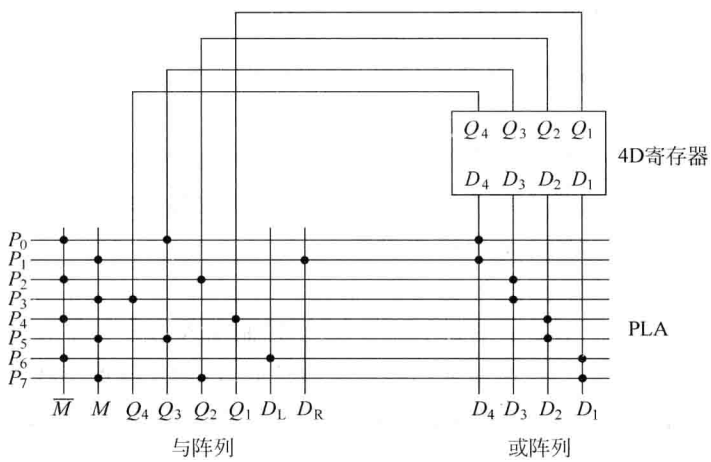


图 2.20　双向移位寄存器

$M=0$，左移，即 $Q_4 \leftarrow Q_3 \leftarrow Q_2 \leftarrow Q_1 \leftarrow D_L$。

$M=1$，右移，即 $D_R \rightarrow Q_4 \rightarrow Q_3 \rightarrow Q_2 \rightarrow Q_1$。

因此，

$$D_4=\overline{M}Q_3+MD_R=P_0+P_1$$

$$D_3=\overline{M}Q_2+MQ_4=P_2+P_3$$

$$D_2=\overline{M}Q_1+MQ_3=P_4+P_5$$

$$D_1=\overline{M}D_L+MQ_2=P_6+P_7$$

16. 用 PLA 和 4 个 D 触发器组成 1 位十进制计数器。

答：十进制计数器的真值表如表 2.2 所示。

表 2.2　十进制计数器真值表

数　值	原　值 $Q_D\ Q_C\ Q_B\ Q_A$	+1 后的值 $D_D\ D_C\ D_B\ D_A$	数　值	原　值 $Q_D\ Q_C\ Q_B\ Q_A$	+1 后的值 $D_D\ D_C\ D_B\ D_A$
0	0 0 0 0	0 0 0 1	5	0 1 0 1	0 1 1 0
1	0 0 0 1	0 0 1 0	6	0 1 1 0	0 1 1 1
2	0 0 1 0	0 0 1 1	7	0 1 1 1	1 0 0 0
3	0 0 1 1	0 1 0 0	8	1 0 0 0	1 0 0 1
4	0 1 0 0	0 1 0 1	9	1 0 0 1	0 0 0 0
			10～15	不会出现	

由真值表可得出表达式如下：

$$D_A=\overline{Q}_A=P_0$$

$$\begin{aligned}D_B&=\overline{Q}_D\overline{Q}_C\overline{Q}_BQ_A+\overline{Q}_DQ_C\overline{Q}_BQ_A+\overline{Q}_DQ_C\overline{Q}_BQ_A+\overline{Q}_DQ_CQ_B\overline{Q}_A\\&=\overline{Q}_D\overline{Q}_BQ_A+\overline{Q}_DQ_B\overline{Q}_A\end{aligned}$$

由于 $Q_DQ_B\overline{Q}_A$ 不会出现，因此有

$$\begin{aligned}D_B&=\overline{Q}_D\overline{Q}_BQ_A+\overline{Q}_DQ_B\overline{Q}_A+Q_DQ_B\overline{Q}_A\\&=\overline{Q}_D\overline{Q}_BQ_A+Q_B\overline{Q}_A\\&=P_1+P_2\end{aligned}$$

$$\begin{aligned}D_C&=\overline{Q}_D\overline{Q}_CQ_BQ_A+\overline{Q}_DQ_C\overline{Q}_B\overline{Q}_A+\overline{Q}_DQ_C\overline{Q}_BQ_A+\overline{Q}_DQ_CQ_B\overline{Q}_A\\&=\overline{Q}_D\overline{Q}_CQ_BQ_A+(\overline{Q}_DQ_C\overline{Q}_B\overline{Q}_A+\overline{Q}_DQ_C\overline{Q}_BQ_A)+(\overline{Q}_DQ_C\overline{Q}_B\overline{Q}_A+\overline{Q}_DQ_CQ_B\overline{Q}_A)\\&=\overline{Q}_D\overline{Q}_CQ_BQ_A+\overline{Q}_DQ_C\overline{Q}_B+\overline{Q}_DQ_C\overline{Q}_A\end{aligned}$$

由于 $Q_D\overline{Q}_CQ_BQ_A$、$Q_DQ_C\overline{Q}_B$ 和 $Q_DQ_C\overline{Q}_A$ 不会出现，因此

$$\begin{aligned}D_C&=(\overline{Q}_D\overline{Q}_CQ_BQ_A+Q_D\overline{Q}_CQ_BQ_A)+(\overline{Q}_DQ_C\overline{Q}_B+Q_DQ_C\overline{Q}_B)+(\overline{Q}_DQ_C\overline{Q}_A+Q_DQ_C\overline{Q}_A)\\&=\overline{Q}_CQ_BQ_A+Q_C\overline{Q}_B+Q_C\overline{Q}_A\\&=P_3+P_4+P_5\end{aligned}$$

$$D_D=\overline{Q}_DQ_CQ_BQ_A+Q_D\overline{Q}_C\overline{Q}_B\overline{Q}_A$$

加上不会出现项，得

$$\begin{aligned}D_D&=\overline{Q}_DQ_CQ_BQ_A+Q_D\overline{Q}_A(\overline{Q}_C\overline{Q}_B+Q_C\overline{Q}_B+\overline{Q}_CQ_B+Q_CQ_B)\\&=\overline{Q}_DQ_CQ_BQ_A+Q_D\overline{Q}_A\\&=P_6+P_7\end{aligned}$$

十进制计数器的逻辑图如图 2.21 所示，加电时将计数器置成 0。

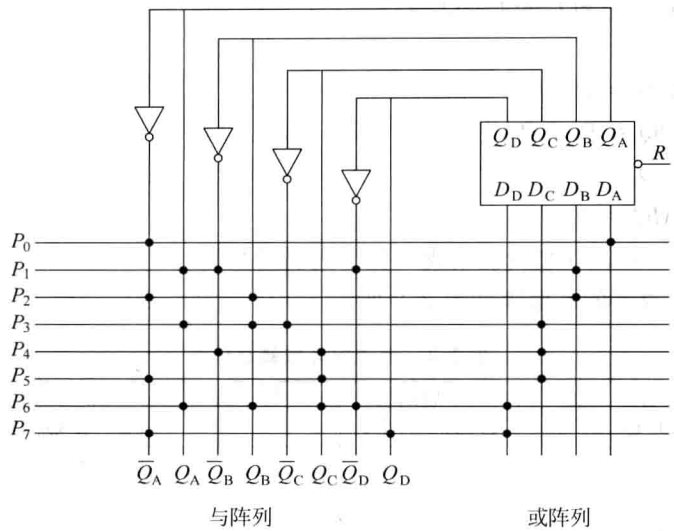


图 2.21　十进制计数器

17. 从有关 GAL16V8 的叙述中，选出正确答案填入□中。

(1) GAL16V8 最多可有 A 个输入，但此时只能有 B 个输出；输出最多 C 个。一般来说，每个输出(F)可以是 D 个乘积项的“或”的函数，这些乘积项的输入可以是输入码 I_0～I_7 或其反码，也可以是 E 函数或其反码。

(2) 假设选择从 F_4 的 OLMC 的触发器输出，此时触发器的时钟 CLK 从芯片第 G 个引出端输入。输出有效控制信号$\overline{OE}$从芯片第 H 个引出端输入。

(3) 当某个输入输出端(I/O)用作输入端时，其 OLMC 是将内部的输出置为 I，且将其引入与阵列。

供选择的答案：

A,B,C,D,G,H：①1；②2；③7；④8；⑤11；⑥16。

E：①输出；②其他。

I：①恒高；②恒低；③高阻态。

答：A——⑥；B——②；C——④；D——④；E——①；G——①；H——⑤；I——③。

18. 今要生产 10 台计算机样机，用于试验定型，预计需根据试验结果修改设计，然后正式生产。今有 4 种芯片(门阵列 GA，宏单元阵列 MCA、标准单元阵列 SCA 和可编程序门阵列 FPGA)可供选择，你认为应该选择哪一种芯片来生产样机和正式产品比较合理？

答：生产样机使用 FPGA 较好，因为 FPGA 可通过编程修改设计，而其他 3 种芯片是半定制或定制芯片，修改设计需要重新生产，费用高，时间长。

正式生产，且批量大时，采用 GA、MCA 或 SCA 比较合理，根据集成度、生产条件、生产周期和价格等因素综合考虑后决定采用其中哪一种芯片。

19. 如果某设备规模较小，对集成度要求不高，在试验阶段有 PAL、PLA 和 GAL 芯片可供使用，你认为选择哪种芯片比较合理？如果试验结束正式投产，选择哪种芯片比较合理？

答：在试验阶段选用 GAL 比较合理，因为可通过编程修改设计，PAL 和 PLA 采用熔丝工艺，一次编程后不能修改。正式生产可考虑用 PAL 或 PLA。

20. 今有 4 门组成的电路如图 2.22 所示，其输入信号有 2 个：电位 D 和时钟 CP，请分析该电路的输出（门 3 和门 4 的输出）。

答：该电路可保证输出脉冲的完整性，在 CP 脉冲上升时，如果 D 为高电平，则在门 4 输出一个负脉冲；如果 D 为低电平，则在门 3 输出一个负脉冲，输出脉冲的宽度与 CP 脉冲的宽度相等。在 CP 脉冲作用期间，如果 D 信号改变，也不会影响输出。

以上结论可通过图 2.23 的波形分析得到。图中输入电位 D 和 3 个脉冲 CP，结果在门 4 输出 2 个负脉冲，门 3 输出 1 个负脉冲。在初始状态，由于 CP 为负，所以门 3 和门 4 输出为高电位，输入 D 为高电位，门 1 的输出为低电位，门 2 的输出为高电位，然后由于 CP 与 D 的变化而引起门 1～门 4 输出的变化如图 2.23 所示，图中的箭头表示波形变化的因果关系。

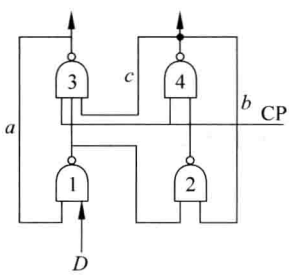


图 2.22　产生完整脉冲的电路

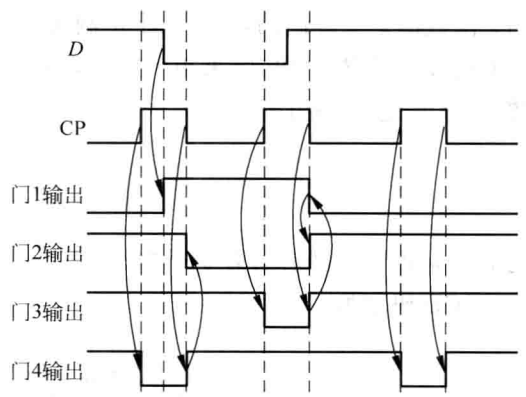


图 2.23　各个门的输出波形

图 2.22 中的 a、b 两根连线是维持线，维持门 3 或门 4 的输出脉冲在 CP 作用期间保持不变（完整）而与 D 的变化无关。连线 c 是阻塞线，保证在门 4 有负脉冲输出时，门 3 不会输出负脉冲。假如没有 a、b、c 这 3 根线，则可将图 2.22 简化成图 2.24，其输出波形如图 2.25 所示，出现不完整的脉冲输出，这种情况在计算机中是不允许产生的。

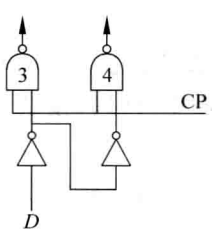


图 2.24　输出窄脉冲的电路

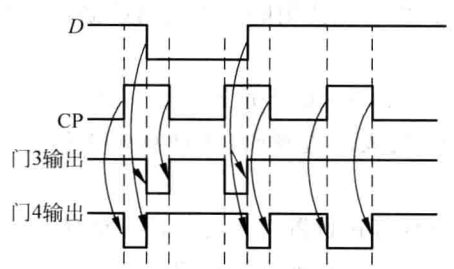


图 2.25　图 2.24 的输出波形

第 3 章　运算方法和运算部件

1. 在讨论计算机中用到的数值时，经常遇到哪些记数制？

答：经常遇到的记数制有二进制、八进制、十六进制和十进制。1 位八进制数用 3 位二进制数来表示，1 位十六进制数用 4 位二进制数来表示，因此在计算机内部，这 3 种记数制并没有差别，仅是在书写时，同一数据当用八进制或十六进制表示时，比二进制短得多，也容易识别。

1 位十进制数在计算机中也是用 4 位二进制数来表示的，称为 BCD 码。同一代码表示多位数据时，在不同的记数制中，其值是不同的。例如代码 00010001，当其为 BCD 码（8421 码）时，等于十进制数 11；当其为十六进制（或二进制）时，等于十进制数 17。

2. 今有两数 00100110 和 01000111，求两数之和。

（1）两数都是二进制码，结果用十进制表示。

（2）两数都是 BCD 码，结果用十进制表示。

答：（1）二进制码相加。

$$\begin{array}{r} 0010\ 0110 \\ +\ 0100\ 0111 \\ \hline 0110\ 1101 \end{array}$$

运算结果用十进制表示为 64＋32＋8＋4＋1＝109。两数相加为 38＋71＝109。

（2）BCD 码相加。

$$\begin{array}{rl} 0010\ 0110 & \\ +\ 0100\ 0111 & \\ \hline 0110\ 1101 & \\ +\ \quad 1\ 0110 & \\ \hline 0111\ 0011 & \text{低位加 6，进位为 1} \end{array}$$

用十进制表示的结果为 73。两数相加为 26＋47＝73。

本题的目的是了解二进制数和二-十进制数（BCD 码）的运算方法的差别，相同代码表示的运算结果是不同的。

3. 人们习惯使用的是十进制数据，而计算机内部运算的是二进制数据，通常是怎样处理的？

答：人们输入的十进制数据一般是用 ASCII 码表示的，先按位转换成 BCD 码，然后将 BCD 码转换成二进制码，在计算机中运算，运算结果再转换成 BCD 码，按位用十进制码输出。

某些计算机设置有十进制运算指令，可以用硬件直接对两个 BCD 码数据进行运算。

4. 将$(73)_{10}$转换成二进制码。

答：（1）用十进制数转换。

	余数	结果
2⌊73		
2⌊36	1	低位
2⌊18	0	↓
2⌊9	0	
2⌊4	1	
2⌊2	0	
2⌊1	0	
0	1	高位

$(73)_{10}=(1001001)_2$

(2) 用 BCD 码转换。

	余数	结果
2⌊ 0111 0011		
2⌊ 0011 0110*	1	低位
2⌊ 0001 1000**	0	↓
2⌊ 0000 1001	0	
2⌊ 0000 0100	1	
2⌊ 0000 0010	0	
2⌊ 0000 0001	0	
0000 0000	1	高位

$(01110011)_{BCD}=(1001001)_2$

注：* 高位除 2 得高位商 0011，高位余数 0001，将该余数加到低位，所以低位为 (0011+1010)/2=1101/2，得低位商 0110 和余数 1。

** 低位(0110+1010)/2=10000/2，得低位商 1000 和余数 0。下同。

如果高位无余数，低位直接除 2。

(3) 采用乘加方法转换。

$(73)_{10}=(7\times10+3)_{10}=(0111\times1010+0011)_2$

$$\begin{array}{r} 0111 \\ \times\ 1010 \\ \hline 0111 \\ 0111 \\ \hline 1000110 \end{array} \qquad \begin{array}{r} 1000110 \\ +\quad 0011 \\ \hline 1001001 \end{array}$$

$(73)_{10}=(1001001)_2$

5. 采用乘加方法，将$(254)_{10}$转换成二进制码。

答： $(254)_{10}=2\times10^2+5\times10+4=((2\times10)+5)\times10+4$

$=((0010\times1010)+0101)\times1010+0100$

$=(10100+0101)\times1010+0100$

$=11001\times1010+0100$

$=11111010+0100$

$=(11111110)_2$

注：在计算机中一般采用乘加方法，所以将公式变换一下，以利于编程和节省运算时间。

6. 将$(0.75)_{10}$转换成二进制码。

答：(1) 用十进制转换。

结果	0.75×2
高位 1	.50×2
低位 1	.00

$$(0.75)_{10}=(0.11)_2$$

(2) 用 BCD 码转换。

$$(0.75)_{10}=(0.01110101)_{BCD}$$

		0111 0101	×2
		1110 1010	
结果		0111 0110	高位 +6+1(低位来的进位)，低位 +6
高位	1	0101 0000	×2
↓		1010 0000	
		0110	高位 +6
低位	1	0000 0000	

$$(0.75)_{10}=(0.11)_2$$

7. 将$(0.73)_{10}$转换成二进制码(小数点后取 4 位有效值)。

答：(1) 用十进制转换。

结果	0.73×2
高位 1	.46×2
↓ 0	.92×2
↓ 1	.84×2
低位 1	.68

$$(0.73)_{10}=(0.1011)_2$$

(2) 用 BCD 码转换。

$$(0.73)_{10}=(0.01110011)_{BCD}$$

		0111 0011	×2
		1110 0110	
结果		0110	高位+6
高位	1	0100 0110	×2
↓		1000 1100	
		0001 0110	高位+1(低位来的进位)，低位+6
	0	1001 0010	×2
	1	0010 0100	
		0110	高位+6
低位		1000 0100	×2
	1	0000 1000	
		0110	高位+6
		0110 1000	

$$(0.73)_{10}=(0.1011)_2$$

8. 将$(73.75)_{10}$转换成二进制码。

答：由于整数部分和小数部分的转换方法不同，所以要分别计算。由第 4 题得$(73)_{10}=$

$(1001001)_2$，由第 6 题得$(0.75)_{10}=(0.11)_2$，所以$(73.75)_{10}=(1001001.11)_2$。

9. 将下列二进制数转换成十进制数。

(1) 1001001　　(2) 11000111

答：将二进制数除以$(10)_{10}$，余数即为结果的最低位，如果商$\geqslant(10)_{10}$，将商再除以$(10)_{10}$，其余数为次低位……当商$<(10)_{(10)}$时，运算结束。$(10)_{10}=(1010)_2$。

(1)

```
                 1 1 1 ···· 十位   商<(1010)2
1 0 1 0 ) 1 0 0 1 0 0 1
          1 0 1 0
          ---------
          1 0 0 0 0 1
            1 0 1 0
          ---------
            0 1 1 0 1
              1 0 1 0
          ---------
              0 0 1 1 ···· 个位
```

$(1001001)_2=(01110011)_{BCD}=(73)_{10}$

(2)

```
                 1 0 0 1 1 ···· 商>(1010)2
1 0 1 0 ) 1 1 0 0 0 1 1 1
          1 0 1 0
          -----------
            0 1 0 0 0 1 1 1
              1 0 1 0
          -----------
              1 0 0 1 1
                1 0 1 0
          -----------
                1 0 0 1 ···· 个位
```

```
                  1 ···· 百位
1 0 1 0 ) 1 0 0 1 1
          1 0 1 0
          -------
          1 0 0 1 ···· 十位
```

$(11000111)_2=(000110011001)_{BCD}=(199)_{10}$

10. 将二进制码 0.1110 转换成十进制码。

答：

结果	$1110\times1010=10001100$
高位 1000	$1100\times1010=01111000$
↓ 0111	$1000\times1010=01010000$
低位 0101	0000

$(0.1110)_2=(0.1000\ 0111\ 0101)_{BCD}=(0.875)_{10}$

11. 将二进制码 11001001.0101 转换成八进制数和十六进制数。

答：$(11001001.0101)_2=(\underline{011}\ \underline{001}\ \underline{001}.\ \underline{010}\ \underline{100})_2=(311.24)_8$

$(11001001.0101)_2=(\underline{1100}\ \ \underline{1001}.\underline{0101})_2=(C9.5)_{16}$

12. 将八进制数$(57.43)_8$转换成二进制数和十六进制数。

答：$(57.43)_8=(\underline{101}\ \underline{111}.\ \underline{100}\ \underline{011})_2=(101111.100011)_2$。八进制转换成十六进制过程：先转换成二进制，再转换成十六进制。在计算机内部，八进制、十六进制和二进制的表现形式是相同的。

$(57.43)_8=(101111.100011)_2=(\underline{0010}\ \underline{1111}.\ \underline{1000}\ \underline{1100})_2=(2F.8C)_{16}$

13. 将十进制数 $7\frac{3}{4}$和$\frac{3}{64}$转换成二进制数。

答：$\frac{1}{4}=2^{-2}$，$\frac{3}{4}=11_2\times2^{-2}=(0.11)_2$，所以 $7\frac{3}{4}=(111.11)_2$，$\frac{1}{64}=2^{-6}$，$\frac{3}{64}=11_2\times2^{-6}=(0.000011)_2$。

14. 完成下列二进制运算(手工运算)。

(1) 101.111+11.011　　(2) 1001.10−110.01

(3) 1001.10×11.01　　(4) 101110111÷1101

答：(1) 101.111+11.011=1001.010

```
   1 0 1.1 1 1
 + 0 1 1.0 1 1
 -------------
 1 0 0 1.0 1 0
```

(2) 1001.10−110.01=11.01

```
 1 0 0 1.1 0
 -   1 1 0.0 1
 -------------
       1 1.0 1
```

(3) 1001.10×11.01=11110.1110

```
       1 0 0 1.1 0
     ×       1 1.0 1
     -----------------
       1 0 0 1 1 0
     1 0 0 1 1 0
   1 0 0 1 1 0
   -----------------
   1 1 1 1 0.1 1 1 0
```

(4) 101110111÷1101=11100

```
                 1 1 1 0 0
1 1 0 1 ) 1 0 1 1 1 0 1 1 1
          1 1 0 1
          -------------
          1 0 1 0 0
            1 1 0 1
          -------------
              1 1 1 1
              1 1 0 1
          -------------
                  1 0 1 1
```

15. 请写出 X 为小数和整数时的补码公式。

答：当 X 为小数时，定义如下：

$$[X]_{补}=\begin{cases}X, & 0\leqslant X<1\\ 2+X=2-|X|, & -1\leqslant X<0\end{cases}$$

当 X 为整数时，定义如下：

$$[X]_{补}=\begin{cases}X, & 0\leqslant X<2^n\\ 2^{n+1}+X=2^{n+1}-|X|, & -2^n\leqslant X<0\end{cases}$$

其中 n 为数值部分的长度(不包括符号位)。

16. 当 X 为负值时，$[X]_{补}$ 与 $[X]_{反}$ 有什么关系？

答：当 X 为负值时，$[X]_{补}=2+X$，$[X]_{反}=2-2^{-n}+X$，所以 $[X]_{补}=[X]_{反}+2^{-n}$。

本题假设 X 为小数，n 为小数点后的有效位数。

在计算机中，从负数的原码转换成补码时，通常采用取数的反码，并在最低位+1 的方法。

17. 为什么在计算机中进行加减法运算时都采用补码运算的方法？

答：如果采用原码运算，由于加减法运算的逻辑表达式是不同的，所以需要采用两套电路来分别执行加法运算和减法运算。而采用补码运算时，由于下列两公式的存在：

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$$

所以加减法运算都可以用加法来实现，节省了一套电路。而从 $[Y]_{补}$ 转变到 $[-Y]_{补}$ 是很方便的，因为在运算部件中，数据都是存放在寄存器中的，寄存器的每一位都有两个电位不同的输出端，如果 $[Y]_{补}$ 从一个输出端输出，则 $[-Y]_{补}$ 从另一个输出端输出，并在最低位+1 即可。+1 信号送到加法器最低位的进位输入端。

18. 当 X 为负数时，为什么 $[X]_{补}$ 和 $[X]_{反}$ 能表示的数值范围不同(假设 X 为小数，举例

说明)?

答：设 $X=-0.1010$，$Y=-0.0110$，$X+Y=-0.1010-0.0110=-1$。

$[X]_{补}+[Y]_{补}=1.0110+1.1010=1.0000$，即为$-1$。

$[X+Y]_{反}=0.1111$(反码运算规则见下题)，两负数相加，结果的符号位为 0(表示正数)，说明该结果不正确，即溢出。由此说明$[X]_{补}$可以为-1，而$[X]_{反}$不能为-1，其最小的负数为$[X]_{反}=1.0000$，即 $X=-(1-2^{-4})$。

19. 试举例说明反码运算的规则。

答：$[X+Y]_{反}=[X]_{反}+[Y]_{反}+$最高位产生的进位信号(加到最低位)。

设 X、Y 为整数，数值部分有 4 位，符号位 1 位(为最高位)。举例如下：

(1) $X=-1$，$Y=+2$

$[X]_{反}=11110$，$[Y]_{反}=00010$

$$\begin{array}{r} 11110 \\ +\ 00010 \\ \hline 100000 \\ +\qquad 1 \\ \hline 00001 \end{array}$$

最高位进位加到最低位

$X+Y=1$

(2) $X=-7$，$Y=2$

$[X]_{反}=11000$，$[Y]_{反}=00010$

$$\begin{array}{r} 11000 \\ +\ 00010 \\ \hline 11010 \end{array}$$

最高位无进位，最低位不需要加 1

$[X+Y]_{反}=11010$，$X+Y=-0101$(即-5)

(3) 参考第 18 题，$[X]_{反}=1.0101$，$[Y]_{反}=1.1001$

$[X+Y]_{反}$：

$$\begin{array}{r} 1.0101 \\ +\ 1.1001 \\ \hline 10.1110 \\ +\qquad 1 \\ \hline 0.1111 \end{array}$$

最高位有进位，加到最低位

溢出

反码运算需要将最高位产生的进位信号加到最低位，且零值有两种编码。在计算机中一般不采用反码运算。因此本题仅供参考，暂时可以不必考虑。

20. 求下列各数的原码：

(1) $[X]_{补}=0.1010$　　(2) $[X]_{反}=0.1010$

(3) $[X]_{补}=1.1010$　　(4) $[X]_{反}=1.1010$

答：(1) $[X]_{原}=0.1010$　　(2) $[X]_{原}=0.1010$

(3) $[X]_{原}=1.0110$　　(4) $[X]_{原}=1.0101$

已知补码求原码的方法和已知原码求补码的方法类似。反码的情况相同。

21. 设 $X=(-69)_{10}$，求$[X]_{原}$、$[X]_{补}$和$[X]_{移}$。

答：$[X]_{原}=11000101$　　最高位为符号位。

$[X]_{补}=10111011$　　同上。

$[X]_{移}=00111011$　　同上。

22. 设$[X]_{补}=0.1011$，$[Y]_{补}=1.1110$，求$[X+Y]_{补}$和$[X-Y]_{补}$之值。

答：$[-Y]_{补}=0.0010$

$[X+Y]_{补}=[X]_{补}+[Y]_{补}=0.1011+1.1110=0.1001$

$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=0.1011+0.0010=0.1101$

23. 设$[X]_{补}=1.0101$，$[Y]_{补}=0.0010$，求$[X+Y]_{补}$和$[X-Y]_{补}$之值。

答：$[-Y]_{补}=1.1110$

$[X+Y]_{补}=[X]_{补}+[Y]_{补}=1.0101+0.0010=1.0111$

$[X-Y]_{补}=[X]_{补}+[-Y]_{补}=1.0101+1.1110=1.0011$

24. 设 $X=-0.1101$，$Y=-0.1001$，求$[X+Y]_{补}$之值。

答：$[X]_{补}=1.0011$，$[Y]_{补}=1.0111$

$[X+Y]_{补}=[X]_{补}+[Y]_{补}=1.0011+1.0111=10.1010$　溢出

25. 已知$[X]_{移}=10101$，$[Y]_{移}=10011$，求$[X+Y]_{移}$和$[X-Y]_{移}$之值。

答：$[Y]_{补}=00011$，$[-Y]_{补}=11101$

$[X+Y]_{移}=[X]_{移}+[Y]_{补}=10101+00011=11000$

$[X-Y]_{移}=[X]_{移}+[-Y]_{补}=10101+11101=10010$

26. 写出下列各二进制数的原码、补码和反码：

0.1010，0，−0，−0.1010，0.1111，−0.0100

答：

X	$[X]_{原}$	$[X]_{补}$	$[X]_{反}$
0.1010	0.1010	0.1010	0.1010
0	0.0000	0.0000	0.0000
−0	1.0000	0.0000	1.1111
−0.1010	1.1010	1.0110	1.0101
0.1111	0.1111	0.1111	0.1111
−0.0100	1.0100	1.1100	1.1011

27. 已知$[X]_{补}$为下列各值：(1)0.1110，(2)1.1100，(3)0.0001，(4)1.1111，(5)1.0000。求 X(真值)。

答：X 的真值如下：

(1)0.1110　(2)−0.0100　(3)0.0001　(4)−0.0001　(5)−1.0000

28. 已知 $X=0.1011$，$Y=-0.0101$。求$[X]_{补}$、$[X/2]_{补}$、$[X/4]_{补}$、$[2X]_{补}$、$[Y]_{补}$、$[Y/2]_{补}$、$[Y/4]_{补}$和$[2Y]_{补}$。

答：$[X]_{补}=0.1011$

$[X/2]_{补}=0.0101$　舍去最低位

$[X/4]_{补}=0.0010$　再舍去最低位

$[2X]_{补}$为溢出

$[Y]_{补}=1.1011$

$[Y/2]_{补}=1.1101$　舍去最低位

$[Y/4]_{补}=1.1110$　再舍去最低位

$[2Y]_{补}=1.0110$

二进制数除以 2，可将数右移 1 位实现之，但要保留符号位不变；乘以 2，将数左移 1 位，最低位补充 0，要判别数据是否溢出。

29. 用压缩十进制数串表示下列十进制数：+66，−78，+254，−396，+1980，−1992。

答：用一个字节存放两个十进制数位，其值用 BCD 码或 ASCII 码的低 4 位表示，符号位也占半个字节并放在最低数字位之后，其值可从 4 位二进制码中的 6 种冗余状态中选用，例如用 B(1011)表示正号，D(1101)表示负号，并规定数字和符号位个数之和必须为偶数，否则在最高位之前补一个 0。用压缩十进制串表示的数据如下：

数字	+66	−78	+254	−396	+1980	−1992
压缩十进制串	066B	078D	254B	396D	01980B	01992D

例如，066B 在计算机内表示则为 0000 0110 0110 1011。

30. 9 位补码(内含 1 个符号位)定点整数和定点小数所能表示的绝对值最大的负数是多少？

答：绝对值最大的负数即是最小的负数。定点整数为−256，定点小数为−1。

31. 说出 $n+1$ 位(含符号位)定点整数(原码、补码、反码和移码)的范围。说出 $n+1$ 位(含符号位)定点小数(原码、补码、反码和移码)的范围。

答：(1) 定点整数：

原码　$-(2^n-1)\sim(2^n-1)$

补码　$-2^n\sim(2^n-1)$

反码　$-(2^n-1)\sim(2^n-1)$

移码　$-2^n\sim(2^n-1)$

(2) 定点小数：

原码　$-(1-2^{-n})\sim(1-2^{-n})$

补码　$-1\sim(1-2^{-n})$

反码　$-(1-2^{-n})\sim(1-2^{-n})$

移码　在计算机中移码仅可能用于阶码，所以无小数。

32. 数据 1.0000，当其为原码、补码和反码时表示的真值是多少？

答：原码表示真值−0，补码表示真值−1，反码表示真值$-(1-2^{-4})$。

33. 补码(含符号位)010101、101010、0.10101 和 1.01010 扩充成 8 位后等于什么？

答：扩充后应保持数值不变，所以整数扩充的位数应安排在左面，小数扩充的位数应安排在右面。整数按符号扩充，小数补以 0，所以扩充后的值为 00010101、11101010、0.1010100和 1.0101000。

34. 设有 5 位二进制数据，当其为无符号整数时，写出数据范围；若为有符号数，最高位为符号，写出数据范围。上述各种情况各能表示多少个数据？

答：无符号整数范围为$(0\sim31)_{10}$，相邻数据的间隔为 1，所以可表示 32 个数据。

有符号原码表示的数据范围为$(-15\sim+15)_{10}$，由于有两种状态 00000 和 10000 均表示数据 0_{10}(+0 和−0)，所以能表示的数据为 31 个。

反码表示的数据范围也为$(-15\sim+15)_{10}$，由于有两种状态 00000 和 11111 均表示数据 0_{10}(+0 和−0)，所以能表示的数据为 31 个。

补码表示的数据范围为$(-16\sim+15)_{10}$，数据 0 只有一种表示形式 00000。如果$[X]_{补}=00000$，则$[-X]_{补}=11111+1$(反码+1)$=00000$，即正 0($[X]_{补}$)和负 0($[-X]_{补}$)均为 00000。$[X]_{补}$ 10000 表示的数据为$(-16)_{10}$，假如将$[-X]_{补}$转换成原码则为 11111+1=

100000，应为+16，但此时的原码已溢出。5 位补码能表示 32 个数据。

35. 设 $X=0.1001$，$Y=-0.1011$，试用原码一位乘法、补码一位乘法（布斯算法）和原码两位乘法计算 $X \cdot Y$ 的乘积，并说明求部分积的次数（即加减运算次数）和移位次数。

答： $[X]_{补}=0.1001$，$[-X]_{补}=1.0111$，$2[-X]_{补}=110.1110$，$[Y]_{补}=1.0101$

（1）原码一位乘法。

部分积	乘数	说明
0 0.0 0 0 0	1 0 1 1	初始值
0 0.1 0 0 1		$+X$
0 0.1 0 0 1		$\xrightarrow{1}$（右移 1 位）
0 0.0 1 0 0	1 1 0 1	
0 0.1 0 0 1		$+X$
0 0.1 1 0 1		$\xrightarrow{1}$
0 0.0 1 1 0	1 1 1 0	
0 0.0 0 0 0		$+0$
0 0.0 1 1 0		$\xrightarrow{1}$
0 0.0 0 1 1	0 1 1 1	
0 0.1 0 0 1		$+X$
0 0.1 1 0 0		$\xrightarrow{1}$
0 0.0 1 1 0	0 0 1 1	

$|X \cdot Y|=0.1001 \times 0.1011=0.01100011$

$X \cdot Y=-0.01100011$

$[X \cdot Y]_{原}=1.01100011$

（2）补码一位乘法。

部分积	乘数	说明
0 0.0 0 0 0	1 0 1 0 1 0	初始值，乘数最后位补一个 0
1 1.0 1 1 1		$+[-X]_{补}$
1 1.0 1 1 1		$\xrightarrow{1}$（右移 1位）
1 1.1 0 1 1	1 1 0 1 0 1	
0 0.1 0 0 1		$+[X]_{补}$
0 0.0 1 0 0		$\xrightarrow{1}$
0 0.0 0 1 0	0 1 1 0 1 0	
1 1.0 1 1 1		$+[-X]_{补}$
1 1.1 0 0 1		$\xrightarrow{1}$
1 1.1 1 0 0	1 0 1 1 0 1	
0 0.1 0 0 1		$+[X]_{补}$
0 0.0 1 0 1		$\xrightarrow{1}$
0 0.0 0 1 0	1 1 0 1 1 0	
1 1.0 1 1 1		$+[-X]_{补}$
1 1.1 0 0 1	1 1 0 1	

$[X \cdot Y]_{补}=11.10011101 \quad X \cdot Y=-0.01100011$

(3) 原码两位乘法。

部分积	乘数	C	说明
0 0 0.0 0 0 0	1 0 1 1	0	初始值
1 1 1.0 1 1 1			$+[-X]_{补}$
1 1 1.0 1 1 1		1	2 →
1 1 1.1 1 0 1	1 1 1 0		
1 1 1.0 1 1 1			$+[-X]_{补}$
1 1 1.0 1 0 0		1	2 →
1 1 1.1 1 0 1	0 0 1 1		
0 0 0.1 0 0 1			$+[X]_{补}$
0 0 0.0 1 1 0	0 0 1 1		

$|X \cdot Y|=0.01100011 \quad X \cdot Y=-0.01100011$

$[X \cdot Y]_{原}=1.01100011$

(4) 加减法运算和移位次数。

设数值部分有 n 位，令 $n=4$，则

原码一位乘法：加法 4 次(n 次)，移位 4 次(n 次)。

补码一位乘法：加减法 5 次($n+1$ 次)，移位 4 次(n 次)。

原码两位乘法：加减法 3 次($n/2+1$ 次)，移位 2 次($n/2$ 次)。

36. 请说明乘法运算部分积的符号位取多少位？

答：一位乘法，部分积符号位取 2 位，因为加减时，数值部分有可能会产生进位到符号位，此时两符号位中的高位是真正的符号位。同理，两位乘法要再增加 1 位符号位，即成为 3 符号位，因为加减时，数值部分有可能产生进位到第 2 符号位，此时最高位是真正的符号位。乘法运算过程中的移位为算术移位，移位时要保持符号位不变。举例如下：设 $X=0.1111$，$Y=0.1010$。执行原码两位乘法。

部分积	乘数	说明	
0 0 0.0 0 0 0	1 0 1 0	初始值	
0 0 1.1 1 1 0		$+2X$	
0 0 1.1 1 1 0		2 →	
0 0 0.0 1 1 1	1 0 1 0		
0 0 1.1 1 1 0		$+2X$	部分积进位到第 2 符号位
0 1 0.0 1 0 1		2 →	
0 0 0.1 0 0 1	0 1 1 0		

$X \cdot Y=0.1111 \times 0.1010=0.10010110$

37. 试画出阵列乘法器的逻辑框图，并说明之。

答：为了进一步提高乘法运算速度，可采用类似于人工计算的方法，用图 3.1 所示的一个阵列乘法器完成 $X \cdot Y$ 乘法运算($X=X_1X_2X_3X_4$，$Y=Y_1Y_2Y_3Y_4$)。

人工计算方法如下：

$$
\begin{array}{rrrrrrrrl}
 & & & & X_1 & X_2 & X_3 & X_4 & \\
 & & & \times) & Y_1 & Y_2 & Y_3 & Y_4 & \\
\hline
 & & & & X_1Y_4 & X_2Y_4 & X_3Y_4 & X_4Y_4 & \text{---该行+初始值得 } P_1 \\
 & & & X_1Y_3 & X_2Y_3 & X_3Y_3 & X_4Y_3 & & \text{--------该行+}P_1\text{ 得 }P_2 \\
 & & X_1Y_2 & X_2Y_2 & X_3Y_2 & X_4Y_2 & & & \text{----------该行+}P_2\text{ 得 }P_3 \\
+) & X_1Y_1 & X_2Y_1 & X_3Y_1 & X_4Y_1 & & & & \text{----------该行+}P_3\text{ 得 }P_4\text{(结果)} \\
\hline
P_{4-1} & P_{4-2} & P_{4-3} & P_{4-4} & P_{4-5} & P_{4-6} & P_{4-7} & P_{4-8} &
\end{array}
$$

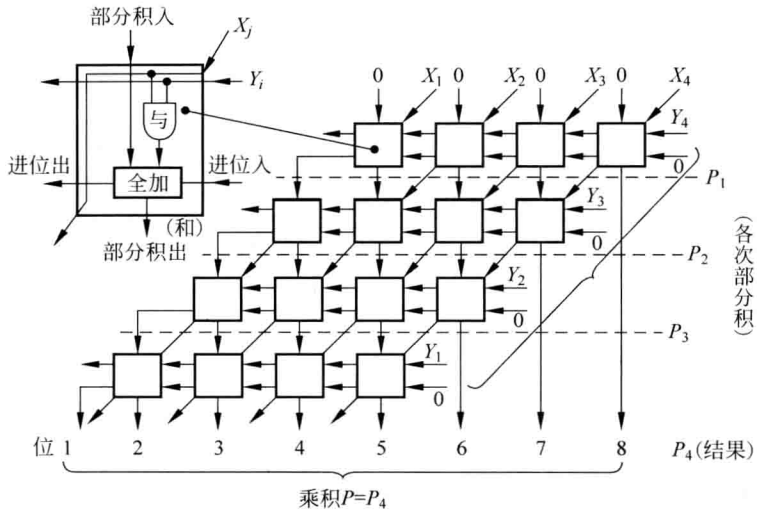


图 3.1　阵列乘法器

图中的每一个方框包含一个与门和一位全加器。阵列的每一行送入乘数 Y 的同一数位，例如第一行的各方框都送入乘数的 Y_4，第二行各方框都送入乘数的 Y_3……而各行错开形成的每一斜列则送入被乘数的每一数位。由于部分积的初始值为 0，所以第一行的部分积输入均为 0；由于最右边的斜列无进位输入，故进位输入也为 0。乘积(P_4)得 8 位，最高位为 P_{4-1}，依次为 P_{4-2}，P_{4-3}，…，P_{4-8}。

38. 试证明采用加减交替法的原码除法运算的正确性。

答：加减交替法是对恢复余数法的一种修正，当某一次求得的余数 R_i 为负时，不是恢复它，而是继续求下一位商，但用加上除数($+Y$)的办法来取代($-Y$)操作，其他操作依然不变，证明如下。

在恢复余数除法中，若第 $i-1$ 次求商的余数为 R_{i-1}，下一次求商的余数为 R_i，则

$R_i=2R_{i-1}-Y$　(R_{i-1}左移 1 位，再减去除数 Y)

如果 $R_i<0$，商的第 i 位上 0，并执行以下操作：恢复余数($+Y$)，将余数左移 1 位，再减 Y，得 R_{i+1}。其过程可用公式表示如下：

$R_{i+1}=2(R_i+Y)-Y=2R_i+Y$。由此得到证明。

39. 定点除法运算为什么先要判断一下被除数是否小于除数？

答：如果被除数不小于除数，商将大于或等于 1，溢出，不能得到正确结果。

40. 叙述定点原码一位除法(不恢复余数法)的运算规则。假设 $X=-0.10110$，$Y=0.11111$，用不恢复余数法求 X/Y 之商。

答：两个原码数相除，其商的符号为两数符号的异或值，数值为两数绝对值相除的商，通常采用加减交替法，即为不恢复余数法。

除法运算规则如下（被除数和除数为绝对值）：

(1) 被除数减除数，余数为正，则溢出；余数为负，上商 0，余数左移 1 位后加除数。

(2) 以后的操作遵循以下规则：

- 当余数为正时，上商 1，求下一位商的办法是余数左移 1 位，再减去除数。
- 当余数为负时，上商 0，求下一位商的办法是余数左移 1 位，再加上除数。
- 重复此步操作，当达到商的有效位长度时，运算结束。

(3) 如果最后一次商 0，而又要求得出正确余数，则再加上除数，即得正确余数；如果最后一次商 1，即为正确余数。

设 $X=-0.10110$，$Y=0.11111$　则

$|X|=0.10110$　$|Y|=0.11111$　$[-Y]_{补}=1.00001$

被除数或余数	说明
0 0.1 0 1 1 0	被除数
1 1.0 0 0 0 1	$+[-Y]_{补}$
1 1.1 0 1 1 1	商 0，$\xleftarrow{1}$（左移 1 位）
1 1.0 1 1 1 0	
0 0.1 1 1 1 1	$+[Y]_{补}$
0 0.0 1 1 0 1	商 1，$\xleftarrow{1}$
0 0.1 1 0 1 0	
1 1.0 0 0 0 1	$+[-Y]_{补}$
1 1.1 1 0 1 1	商 0，$\xleftarrow{1}$
1 1.1 0 1 1 0	
0 0.1 1 1 1 1	$+[Y]_{补}$
0 0.1 0 1 0 1	商 1，$\xleftarrow{1}$
0 1.0 1 0 1 0	
1 1.0 0 0 0 1	$+[-Y]_{补}$
0 0.0 1 0 1 1	商 1，$\xleftarrow{1}$
0 0.1 0 1 1 0	
1 1.0 0 0 0 1	$+[-Y]_{补}$
1 1.1 0 1 1 1	商 0
0 0.1 1 1 1 1	$+[Y]_{补}$（恢复余数）
0 0.1 0 1 1 0	商的符号＝1＋0＝1

$|商|=0.10110$，商＝1.10110（原码）

$|余数|=0.10110\times2^{-5}$，余数＝1.10110×2^{-5}（原码）

41. 叙述定点补码一位除法的运算规则。设 $X=0.10110$，$Y=-0.11111$，用补码一位除法求 X/Y 之商。

答：在被除数的绝对值小于除数的绝对值（即商不溢出）的情况下，补码一位除法的运算规则如下：

(1) 如果被除数与除数同号，用被除数减去除数；如果两数异号，用被除数加上除数。如果所得余数与除数同号，上商 1；如果余数与除数异号，上商 0。该商即为结果的符号位。

(2) 求商的数值部分。

如果上次上商 1，将余数左移一位后减去除数；如果上次上商 0，将余数左移一位后加上除数。然后判断本次操作后的余数，如果余数与除数同号，上商 1；如果余数与除数异号，上商 0。设数值部分有 n 位，重复执行 $n-1$ 次。

(3) 商的最后一位一般采用恒置 1 的办法，此时最大误差为 $\pm 2^{-n}$。如果对商的精度要求较高，则可按(2)的规则再进行一次操作，以求得商的第 n 位。当除不尽时，若商为负，要在商的最低位加 1，使商从反码值转变为补码值；若商为正，最低位不需要加 1。

设 $X=0.10110$，$Y=-0.11111$　用补码一位除法求 X/Y 之商的方法如下：

$$[Y]_{补}=1.00001 \quad [-Y]_{补}=0.11111$$

被除数或余数	说明
0 0.1 0 1 1 0	被除数
1 1.0 0 0 0 1	$+[Y]_{补}$
1 1.1 0 1 1 1	商 1，$\leftarrow 1$
1 1.0 1 1 1 0	
0 0.1 1 1 1 1	$+[-Y]_{补}$
0 0.0 1 1 0 1	商 0，$\leftarrow 1$
0 0.1 1 0 1 0	
1 1.0 0 0 0 1	$+[Y]_{补}$
1 1.1 1 0 1 1	商 1，$\leftarrow 1$
1 1.1 0 1 1 0	
0 0.1 1 1 1 1	$+[-Y]_{补}$
0 0.1 0 1 0 1	商 0，$\leftarrow 1$
0 1.0 1 0 1 0	
1 1.0 0 0 0 1	$+[Y]_{补}$
0 0.0 1 0 1 1	商 0，$\leftarrow 1$
0 0.1 0 1 1 0	
1 1.0 0 0 0 1	$+[Y]_{补}$
1 1.1 0 1 1 1	商 1

商(反码)=1.01001　商(补码)=1.01010　商=−0.10110

另有一种方法，最后一次 $+[Y]_{补}$ 不进行，而采取商的最低位恒置“1”的方法，此时商(补码)=1.01001。

42. 定点数的加减法运算如何判断溢出？

答：定点数加减法运算一律转换成补码加法运算处理：

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$$

$$[X+Y]_{补}=[X]_{补}+[Y]_{补}$$

补码加法运算判断溢出的条件如下：

(1) 符号相同两数相加，结果符号与加数(或被加数)的符号不同，为溢出。

(2) 任意符号两数相加，如果数值部分最高位的进位与符号位的进位不同，为溢出。

(3) 采用双符号位 $f_{s2}f_{s1}$，即正数的符号位为 00，负数的符号位为 11，符号位参与运算，相加结果的两个符号位不同 $f_{s2}\neq f_{s1}$，为溢出，f_{s2} 为第 2 符号位，f_{s1} 为第 1 符号位。

例：$X=0.1011$，$Y=0.0110$，$X+Y$ 的 3 种判断溢出的方法如下：

```
(1)   0.1011        (2)   0.1011        (3)   00.1011
     +0.0110              +0.0110              +00.0110
     -------              -------              --------
      1.0001              01.0001              01.0001
  结果的符号            ←←                    双符号位不同
  与加数不同            无有                  (fs2=0,fs1=1)
                        进进
                        位位
```

上例说明：满足 3 个条件中的任意 1 条，均可判断为溢出。由双符号位表示的补码又称变形补码。

43. 如何判断定点数乘/除法运算的结果是否溢出？

答：定点原码乘法运算的结果不会溢出，因为两个绝对值<1 的定点小数相乘不会≥1，两个 n 位整数相乘其结果不会超过 $2n$ 位(结果取双倍字长)。补码运算仅当 $X=-1$ 且 $Y=-1$ 时(小数)或 $X=-2^n$ 且 $Y=-2^n$ 时(整数)才会产生溢出。

除法运算先要比较两数的大小，要求被除数的绝对值小于除数，否则为溢出。

补码乘法运算举例如下。

例 1：$[X]_{补}=1.0000$，$[Y]_{补}=1.0000$，求 $X\cdot Y$。

答：$[-X]_{补}=01.0000$

```
部分积        乘数        说明
00.0000      |0000
00.0000      0|000       +0,1→
00.0000      00|00       +0,1→
00.0000      000|0       +0,1→
00.0000      0000|       +0,1→
01.0000                  +[-X]补
-------------------
01.0000      0000|
```

$X\cdot Y=01.0000\ 0000=(+1)_{10}$ 溢出

例 2：$[X]_{补}=1.0000$，$[Y]_{补}=1.0001$，求 $X\cdot Y$。

答：$[X]_{补}=11.0000$　$[-X]_{补}=01.0000$

```
部分积        乘数        说明
00.0000      |0001
11.0000                  +[X]补
-------------------
11.0000      |           1→
11.1000      0|000
11.1100      00|00       +0,1→
11.1110      000|0       +0,1→
11.1111      0000|       +0,1→
01.0000                  +[-X]补
-------------------
00.1111      0000|
```

$X\cdot Y=0.11110000$　不溢出

44. 将数据$(123)_{10}$转换成规格化浮点数。已知阶码 5 位(含符号位),基数为 2 和 8 两种情况,采用移码;尾数 9 位(含符号位),采用补码。

答:$(123)_{10}=(1111011)_2$

当基数为 2 时,有

$(123)_{10}=0.1111011\times 2^7$

阶码$=(7)_{10}=(10111)_2$(移码),尾数$=0.11110110$

当基数为 8 时,有

$(123)_{10}=(0.001111011)_2\times 2^9=(0.001111011)_2\times 8^3$

阶码$=(3)_{10}=(10011)_2$(移码),尾数只能保留 9 位,要将最低位舍去,今假设采用 0 舍 1 入法,则得尾数$(0.00111110)_2$。

45. 浮点数阶码 4 位(含符号位),尾数 9 位(含符号位)均采用补码,求数值范围。基数为 2 和 8 两种情况。哪种情况能表示的数值范围大,精度高?

答:基数为 2 时,最大阶码为 0111,最大正数为 $2^7(1-2^{-8})$,最小负数为 $2^7(-1)$,数值范围为$-2^7\sim 2^7(1-2^{-8})$。

基数为 8 时,最大阶码为 0111,最大正数为 $8^7(1-2^{-8})$,最小负数为 $8^7(-1)$,数值范围为$-8^7\sim 8^7(1-2^{-8})$。

基数为 8 时,表示的数值范围大。精度取决于尾数的长度,两者是相同的。

46. 今有以下两浮点数(阶码和尾数都用补码表示)

X:阶码 0000　尾数 0.10000111

Y:阶码 1111　尾数 0.11100100

求 $X-Y$ 之值。

答:(1) 对阶:

$[-Y_{阶}]_{补}=0001$　$[X_{阶}-Y_{阶}]_{补}=0000+0001=0001$(阶差)

取大阶 0000,$Y_{尾}$ 右移 1 位,得 0.01110010。

(2) 尾数相减,有

$[-Y_{尾}]_{补}=1.10001110$

$[X_{尾}-Y_{尾}]_{补}=0.10000111+1.10001110=0.00010101$

(3) 向左规格化:尾数左移 3 位,得 0.10101000,阶码-3,$[阶码]_{补}=0000+1101=1101$(-3 的二进制补码为 1101)。

(4) 舍入:不需要。

结果:阶码为 1101,尾数为 0.10101000,均为补码。

47. 今有以下两浮点数(阶码用移码表示,尾数用补码表示):

X:阶码 1111　尾数 0.10000111

Y:阶码 1111　尾数 0.11100100

求 $X+Y$ 之值。

答:$[-Y_{阶}]_{补}=1001$

(1) 求阶差:

$[X_{阶}-Y_{阶}]_{移}=[X_{阶}]_{移}+[-Y_{阶}]_{补}=1111+1001=1000$

阶差为 0,取阶值 1111。

(2) 尾数相加,有

$0.10000111+0.11100100=1.01101011$

(3) 向右规格化：尾数右移 1 位，得 0.101101011，阶码＋1，$[阶码]_{移}$＝1111＋0001，阶码上溢，运算中止，发溢出中断。

48. 今有以下两浮点数(阶码用移码表示，尾数用补码表示)：

X：阶码 0010　尾数 0.10000111

Y：阶码 0001　尾数 0.11100100

求 $X-Y$ 之值。

答：$[-Y_{阶}]_{补}=0111$

(1) 求阶差：

$[X_{阶}]_{移}+[-Y_{阶}]_{补}=0010+0111=1001$

取大阶 0010，$Y_{尾}$ 右移 1 位得 0.01110010。

(2) 尾数相减：

$[-Y_{尾}]_{补}=1.10001110$

$[X_{尾}-Y_{尾}]_{补}=0.10000111+1.10001110=0.00010101$

(3) 规格化：尾数左移 3 位，得 0.10101000，阶码－3，$[阶码]_{移}$＝0010＋1101＝1111，溢出(下溢)。

(4) 将阶码和尾数置全 0(机器 0)。

49. 一个 IEEE 754 标准单精度浮点数如下：

0000　1111　1110　1111　1100　0000　0000　1111

请说明阶码和尾数各为多少？

答：单精度浮点数字长 32 位，阶码用增码(即移码)表示，尾数用原码表示。设最高位为第 31 位，最低位为第 0 位，则：

第 31 位为尾数符号位 0。

第 30～23 位为阶码 00011111。

第 22～0 位为尾数，在尾数前还有 1 位隐藏位，所以实际的尾数在最高位前面补 1，为 1110 1111 1100 0000 0000 1111，共 24 位。

50. 设有两浮点数 X 和 Y，阶和尾数均以补码表示，已知：X 的阶码为 0010，尾数为 0.1001；Y 的阶码为 1101，尾数为 1.0111。求 $X\cdot Y$ 和 X/Y 之值。

答：(1) 求 $X\cdot Y$ 之值，步骤如下。

① 阶码相加：0010＋1101＝1111。

② 尾数相乘：$X_{尾}\cdot Y_{尾}=-0.01010001$，或$[X_{尾}\cdot Y_{尾}]_{补}=1.10101111$(乘法过程略)。

③ 向左规格化：左移 1 位，阶码－1。

乘积的阶码＝阶码－1＝1111＋1111＝1110(补码)。

乘积的尾数＝1.01011110(补码)。

④ 舍入(取 4 位结果)：1.0101＋0.0001＝1.0110(补码)。

结果：阶码 1110，尾数 1.0110，均以补码表示。

(2) 求 X/Y 之值，步骤如下。

① 阶码相减：0010＋0011＝0101($[-Y_{阶}]_{补}=0011$)。

② 尾数相除：采用原码 1 位除法(加减交替法)，$X_{尾}=0.1001$，$|Y_{尾}|=0.1001$，$[-|Y_{尾}|]_{补}=1.0111$，被除数尾数取两符号位。

被除数或余数	说明
0 0.1 0 0 1	初始值
1 1.0 1 1 1	$-Y$(即$+[-\lvert Y\rvert_{尾}]_{补}$)
0 0.0 0 0 0	商 1
0 0.0 0 0 0	$\xleftarrow{1}$(左移 1 位)
1 1.0 1 1 1	$-Y$
1 1.0 1 1 1	商 0
1 0.1 1 1 0	$\xleftarrow{1}$
0 0.1 0 0 1	$+Y$
1 1.0 1 1 1	商 0
1 0.1 1 1 0	$\xleftarrow{1}$
0 0.1 0 0 1	$+Y$
1 1.0 1 1 1	商 0
1 0.1 1 1 0	$\xleftarrow{1}$
0 0.1 0 0 1	$+Y$
1 1.0 1 1 1	商 0

$$|商_{尾}|=1.0000$$

$$尾数符号=0+1=1$$

$$[尾数]_{补}=[-|商_{尾}|]_{补}=1.0000$$

不需要进行规格化和舍入操作。

结果：阶码=0101，$[尾数]_{补}=1.0000$。

51. 浮点数的阶码为什么经常采用移码？

答：因为移码便于比较(正数符号位为 1，负数的符号位为 0，编码大的数其值也大)，并便于实现加减运算。移码的最小值正好是机器 0。

52. 规格化浮点数阶码有 P 位，尾数有 M 位，各自包含 1 符号位，且都用补码表示，该数的基数 r 为 2，请说出用这种格式能表示的全部不同的规格化数的总个数。当 $r=8$ 时，又能表示多少个规格化数(不考虑机器 0)。

答：对于同一阶码，尾数能表示的个数为 2^M 个，当 $r=2$ 时，能表示的规格化数为 $2^M\times\frac{1}{2}=2^{M-1}$。阶码的个数为 2^P，所以能表示的全部规格化浮点数为 $2^P\cdot 2^{M-1}=2^{P+M-1}$个。

当 $r=8$ 时，能表示的浮点数为 $2^P\cdot 2^M\cdot\frac{7}{8}=\frac{7}{8}\times 2^{P+M}$个。

53. 在计算机内部，浮点数相加是否满足加法结合律，即 $x+(y+z)=(x+y)+z$？请说明理由。

答：在计算机内部，由于浮点运算存在精度问题(由规格化及舍入引起的)，所以不满足加法的结合律，运算结果为非精确值，存在误差。今举例如下：

设 $x=2^{-4}\times 0.1000$，$y=2^{-4}\times 0.1001$，$z=2^0\times 0.1000$，尾数保留 4 位。

(1) 采用截断舍入法。

$$x+(y+z)=2^{-4}\times 0.1000+(2^{-4}\times 0.1001+2^0\times 0.1000)$$

$$=2^{-4}\times 0.1000+2^{0}\times 0.1000$$
$$=2^{0}\times 0.1000$$
$$(x+y)+z=(2^{-4}\times 0.1000+2^{-4}\times 0.1001)+2^{0}\times 0.1000$$
$$=2^{-3}\times 0.1000+2^{0}\times 0.1000$$
$$=2^{0}\times 0.1001$$

两者不相等，相差 2^{-4}。

(2) 采用 0 舍 1 入法。

$$x+(y+z)=2^{-4}\times 0.1000+(2^{-4}\times 0.1001+2^{0}\times 0.1000)$$
$$=2^{-4}\times 0.1000+2^{0}\times 0.1001$$
$$=2^{0}\times 0.1010$$
$$(x+y)+z=(2^{-4}\times 0.1000+2^{-4}\times 0.1001)+2^{0}\times 0.1000$$
$$=2^{-3}\times 0.1001+2^{0}\times 0.1000$$
$$=2^{0}\times 0.1001$$

两者不相等，相差 2^{-4}。

54. 在计算机中，经常采用的数据校验码是什么？一般使用在什么场合？

答：经常采用的数据校验码是奇偶校验码、海明校验码和循环冗余校验码(CRC)。一般在传送数据时为每一字节数据配以一个奇(或偶)校验位。主存储器为每一字配以海明校验码。CRC 一般用于磁盘、磁带等存储器中。

55. 今有 4 个数：00001111、11110000、00000000 和 11111111，问：

(1) 其码距为多少？能纠正和(或)发现多少位错，如果出现数据 00011111 应纠正成什么数？如何纠正？

(2) 如果再加上两个数 00010000、11001111(共 6 个数)，其码距为多少？能纠正和(或)发现多少位错？

(3) 如果为上述各数加上奇校验位，求奇校验位之值。

答：(1) 4 个数之间有 4 位不同，故码距为 4，能纠正 1 位错和发现两位错。如果出现数据 00011111，应纠正成 00001111，纠正的办法是取出错位的反码。

(2) 码距是根据数据中任意两个合法码之间最少有几位二进制位不同而确定的。由于 00000000 和 00010000 仅有 1 位不同，故码距为 1，假如数据 00010000 出错而成为 00000000 时，由于后者仍为合法码，所以不能发现和纠正该错误。

(3) 除了 00010000 的奇校验位为 0 外，其余数据的奇校验位均为 1。

56. 设待校验的数据 $D_8\sim D_1=10101011$，其最高位为 D_8，若采用能纠正 1 位错的海明校验码，求出其校验码。如何纠错？

答：根据公式 $2^r\geqslant k+r+1$，今数据位 $k=8$，所以校验位 $r=4$，设 4 位校验位为 $P_1\sim P_4$ 并采用偶校验。8 位数据位为：$\begin{matrix}D_8 & D_7 & D_6 & D_5 & D_4 & D_3 & D_2 & D_1\\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1\end{matrix}$，则：

$$P_1=D_1\oplus D_2\oplus D_4\oplus D_5\oplus D_7=1+1+1+0+0=1$$
$$P_2=D_1\oplus D_3\oplus D_4\oplus D_6\oplus D_7=1+0+1+1+0=1$$
$$P_3=D_2\oplus D_3\oplus D_4\oplus D_8=1+0+1+1=1$$
$$P_4=D_5\oplus D_6\oplus D_7\oplus D_8=0+1+0+1=0$$

纠错时，将 8 位数据码和 4 位校验码组成的汉明码按下列顺序排列：

$$H_{12}\,H_{11}\,H_{10}\,H_9\,H_8\,H_7\,H_6\,H_5\,H_4\,H_3\,H_2\,H_1$$

$$D_8\;\;D_7\;\;D_6\;\;D_5\;P_4\;D_4\;D_3\;D_2\;P_3\;D_1\;P_2\;P_1$$

P_1、P_2、P_3 和 P_4 各处于 H_1、H_2、H_4 和 H_8 位置（2 的幂次）。检查 $P_1 \sim P_4$ 之值，如果上述公式均成立，说明无错。如果仅 P_1 公式有误，说明检验位 P_1 出错；同样，如果仅是 P_2、P_3 或 P_4 公式之一有误，说明是检验位 P_2、P_3 或 P_4 之一有误。假如有一个以上公式有误，说明数据位有误，出现错误的位置由 H 的下标相加而得到。例如，当 P_1 和 P_3 公式有误时，其对应位置的 H 下标为 1 和 4，所以出错的位置在 $H_5(1+4=5)$，即 D_2。检查 P 的公式，在 P_1 和 P_3 的公式中有 D_2，而在 P_2 和 P_4 公式中无 D_2，因此当 D_2 出错时，反映出公式 P_1 和 P_3 有误；同理，如果 P_1、P_2 和 P_4 有错，其对应位置的 H 下标为 1、2 和 8，所以出错的位置在 $H_{11}(1+2+8=11)$，即 D_7。

查出错误码后，取其反码，即得正确码。

57. 计算机准备传送的数据是 $(100110)_2$，采用 CRC 校验码，生成多项式选用 $X^3+X^1+X^0$。

(1) 要求计算校验位。

(2) 如何判断数据是否出错？

答：(1) 计算校验位的步骤如下：

① 将数据表示成多项式 $M(x)$。

$$M(x)=x^5+x^2+x$$

② 将 $M(x)$ 乘以 x^3，目的是空出 3 位，以便拼装 3 位校验码。

$$M(x)\cdot x^3=x^8+x^5+x^4$$

③ 用 $G(x)$ 对 $M(x)\cdot x^3$ 进行模 2 除。$G(x)$ 为生成多项式。

$$\frac{M(x)x^3}{G(x)}=\frac{x^8+x^5+x^4}{x^3+x^1+x^0}=x^5+x^3+1+\frac{x+1}{x^3+x+1}$$

余数多项式 $R(x)=x+1$。

④ 将 $M(x)x^3$ 与 $R(x)$ 相加得 CRC 多项式 $x^8+x^5+x^4+x+1$。

CRC 码为 100110011，其中信息码为 100110，校验码为 011。

(2) 判断数据是否出错。

将 CRC 多项式除以 $G(x)$（模 2 除），如能除尽，表示无错误；如余数不为 0，说明有错，而且错误位不同，余数也不相同，所以不是任意一个多项式都能用做生成多项式。

$$\frac{\text{CRC 多项式}}{G(x)\text{多项式}}=\frac{x^8+x^5+x^4+x+1}{x^3+x+1}=\frac{100110011}{1011}=101001\text{，余数为 }0\text{。}$$

运算过程如下：

```
                 1 0 1 0 0 1
        ┌──────────────────
1 0 1 1 │ 1 0 0 1 1 0 0 1 1
          1 0 1 1
          ─────────────────
            0 1 0 1 0
              1 0 1 1
          ─────────────────
              0 0 0 1 0 1 1
                    1 0 1 1
          ─────────────────
                    0 0 0 0
```

除尽，无错误。

58. 在运算器中，完成 n 位加减法运算一般需要哪些逻辑电路？

答：一般遵循补码运算规则进行加减法运算。需要下列逻辑电路。

(1) 两个寄存器。用于存放两个源操作数，其中一个还用于保存运算结果。寄存器用 n 位触发器组成。

(2) n 位加法器。如果要判别带符号的数的运算是否发生溢出情况，则需要增加判别溢出的电路或增加 1 位加法器；如果数的长度超过 n 位(例如 $2n$ 位)，则需进行 2 次运算，其中低位运算可能会向高位产生进位或借位信号。为此在运算器中一般设置有溢出触发器 V 和进位/借位触发器 C，称为条件码或状态码。

(3) 传送数据的门电路。包括寄存器向加法器传送原码或反码的门电路以及运算结果从加法器传送到寄存器的门电路。如有必要(例如将反码变为补码，或双字长运算低 n 位产生进位信号)则形成 $+1$ 信号送往 n 位加法器的最低位。

(4) 其他。一般在运算器中还设置 N 和 Z 两个条件码，当运算结果为负数时 $N=1$，否则为 0；当运算结果为零时，$Z=1$，否则 $Z=0$。

另外，在运算器中一般设置有通用寄存器，用于暂存参与运算的数据和运算结果。

59. 接上题，如果运算器增加乘除法运算功能，主要增加哪些逻辑电路？

答：增加一个寄存器，因为运算过程中需要保留部分乘积或余数；增加实现移位功能的电路；设置一个计数器控制运算过程中相加(减)和移位的次数。其他还有一些完成某些功能的电路，例如用于求乘积或商的符号(正数/负数)、上商以及与加/减运算不同的判溢出逻辑电路等。

在设计运算器时，对所有的实现细节都要进行详细考虑。

60. 在以下的空格中，填上有关浮点数运算的内容。

完成浮点数运算的部件分成阶码运算和 $\boxed{A}$ 两部分。加法运算一般按以下顺序执行：①对阶，② $\boxed{B}$，③ $\boxed{C}$，④舍入。在上述各阶段中，凡是涉及阶码运算内容时，都要判断中间结果或最终结果是否溢出，溢出分为 $\boxed{D}$ 和下溢两种情况，当发生下溢时，将运算结果置为 $\boxed{E}$；而发生 $\boxed{D}$ 情况时，则按出错处理。

阶码一般用补码或 $\boxed{F}$ 表示，尾数要维持规格化表示形式，以达到尽量保持数据的 $\boxed{G}$。

答：A——尾数加、减、乘、除运算；B——尾数运算；C——规格化；D——上溢；E——机器零；F——移码；G—精度。

第4章 主存储器

1. 组成计算机的主存储器有哪些类型？

答：(1) 随机存储器(RAM)又称为读写存储器。是可随机对其各个存储单元进行读出或写入操作的存储器。

(2) 只读存储器(ROM)是一种对其内容只能读出而不能写入的存储器，它的内容是在芯片制造时写入的，写入后不能再修改。

(3) 可编程只读存储器(PROM)芯片制造出来以后，可由用户一次性写入的存储器，写入后，可多次读出，但不能再修改。

(4) 可擦除的可编程只读存储器(EPROM)是可用紫外线擦除其内容的可编程只读存储器，擦除后可再次写入。

(5) 可用电擦除的可编程只读存储器(EEPROM 或 E^2PROM)是可用电擦除后再写入内容的可编程只读存储器。近年来快速发展起来的闪存(flash memory)具有 E^2PROM 的特点，但工艺有差别，它用一个管子来存储一位(bit)数据，而 E^2PROM 用 2 个管子。

上述各种存储器除 RAM 在停电后会丢失数据外，其余的均能保持原有数据，为非易失性存储器(又称为非挥发性存储器)，RAM 则被称为易失性存储器(又称为挥发性存储器)。

在写入方面，除了 RAM 能直接写入以外，其余的存储器(EPROM、E^2PROM 和 flash memory)都需要先擦除再写入。

一般将可由用户写入内容或改变性能的芯片都称为可编程芯片。

2. 存储器的读写时间和存取周期有何不同？

答：读写时间指的是从读出或写入操作开始到数据读出或写入操作完成的时间。

存取周期指的是两次读出或写入操作之间的时间间隔。

存取周期大于读写时间。

3. 静态随机存储器(SRAM)和动态随机存储器(DRAM)有何差别？

答：SRAM 用六管电路存储 1 位数据，只要不停止供电，数据不会丢失；DRAM 用一个管子和一个电容器存储 1 位数据，电容上存储的电荷(数据)由于漏电而会消失，因此每隔一定时间(例如 2ms)需要再充一次电，这一过程称为再生或刷新。

SRAM 的集成度比 DRAM 低，这是因为存储 1 位信息所需的芯片面积大，所以价格也比较贵，但读写时间短，速度快。

从芯片的引出端来看，地址线、数据线和读写命令两者都是需要的，片选信号不同，SRAM 由片选允许($\overline{CE}$)或片选选择($\overline{CS}$)信号来控制是否能进行读写，DRAM 则由行选信号($\overline{RAS}$)来控制是否能进行读写。

计算机的主存储器一般使用 DRAM，而 cache 存储器一般使用 SRAM。

4. 为什么 DRAM 芯片的地址一般要分两次接收？今有容量为 1GB 的芯片，当其内部结构各为 1G×1 位或 256M×4 位时，其引出端有何不同？

答：当芯片容量增大时，其地址线数量相应增加，分两次接收地址可将地址线的数量减少一半。引出端多，芯片面积大。

1G×1 位的 DRAM，其地址为 30 位，地址线为 30/2＝15 根，即芯片地址的引出端为 15 个。数据引出端 1 个，当读出时，数据从该端输出；当写入时，数据从该端输入。

256M×4 位的 DRAM，其地址为 28 位，芯片地址引出端为 14 个，数据引出端为 4 个。

5. 对于 SRAM 芯片，如果片选信号始终是有效的。问：

(1) 若读控制信号有效后，地址仍在变化，或数据线上有其他电路送来的信号，问对读出有什么影响？有什么其他问题？

(2) 若写控制信号有效后，地址仍在变化，或写入数据仍不稳定，对写入有什么影响？有什么其他问题？

答：(1) 若读控制信号有效后，地址仍在变化，则输出的数据将受地址变化的影响而变化，而且输出可能会延迟，因为数据要在地址稳定一定时间后才能正常输出。但不影响芯片中存储的数据。如果数据端有其他电路送来的信号，则可发生信号冲突现象，会影响输出信号甚至损坏芯片。

(2) 写控制信号有效后，地址仍在变化，则可能会误写入（地址变化前后的相关单元都可能写入数据），也有可能因地址或数据的稳定时间不够而不能对要求的地址写入正确的数据。

6. 图 4.1 是某 SRAM 的写入时序图，其中 R/$\overline{W}$ 是读写命令控制线，当 R/$\overline{W}$ 线为低电平时，存储器按给定地址 24A8H 把数据线上的数据写入存储器。请指出该图写入时序中的错误，并画出正确的写入时序图。

答：R/$\overline{W}$ 命令（低电平）应往后延，写时地址不允许变化，数据可以在 R/$\overline{W}$ 命令下降边之前或之后给出，但要注意写入时间是否足够。图 4.2 为正确的写入时序图。

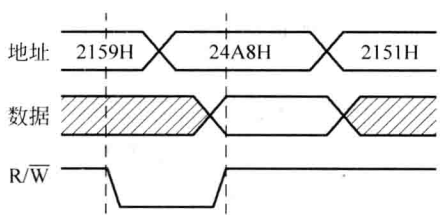


图 4.1　有错误的 SRAM 写入时序图

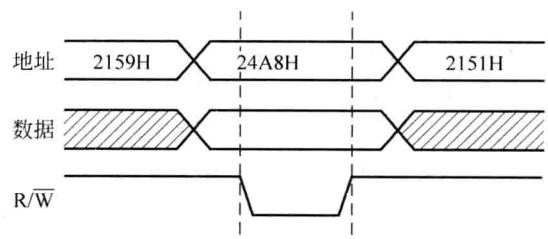


图 4.2　正确的 SRAM 写入时序图

7. 图 4.3 是某 DRAM 写入时序图，请指出图中的错误，并画出正确的时序图。

答：请先复习教材第 4 章中的动态存储器框图。

图 4.3 中行选通$\overline{RAS}$与列选通$\overline{CAS}$的次序应互换，有了$\overline{RAS}$后，才能产生列选通信号。

写信号（R/$\overline{W}$＝0）一般早于$\overline{CAS}$到达，此时写入操作由$\overline{CAS}$下降沿启动。如果写信号迟于$\overline{CAS}$到达，则写入操作由写信号（R/$\overline{W}$＝0）的下降沿启动。

D_{IN}信号应早于$\overline{CAS}$到达。写入数据 D_{IN}是由写时钟来锁存的，而写时钟又是列时钟（它是由 $\overline{CAS}$ 下沿激发的）和 R/$\overline{W}$＝0 共同作用产生的，如果 R/$\overline{W}$＝0 先于$\overline{CAS}$＝0 到达，那么输入数据锁存的是$\overline{CAS}$下降沿作用时的 D_{IN}数据，所以 D_{IN}要先于$\overline{CAS}$＝0 到达。

图 4.4 是修改后的时序图。

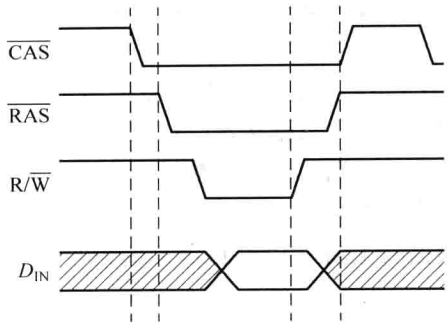


图 4.3　有错误的 DRAM 写入时序图

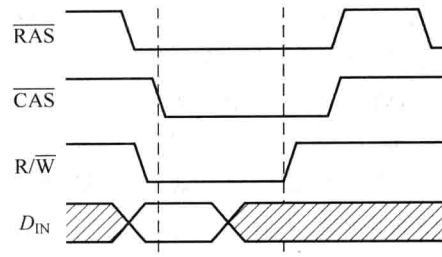


图 4.4　正确的 DRAM 写入时序图

8. 有一个 512K×16 位的存储器，由 64K×1 位的 2164RAM 芯片构成（芯片内是 4 个 128×128结构），问：

（1）总共需要多少个 RAM 芯片？

（2）采用分散刷新方式，如单元刷新间隔不超过 2ms，则刷新信号的周期是多少？

（3）如采用集中刷新方式，设读/写周期 $T=0.1\mu s$，存储器刷新一遍最少用多少时间？

答：（1）总共需要(512/64)×16＝128 片。

（2）由于芯片内部是 4 个 128×128 结构，所以访问 128 个行地址可刷新一遍。

分散刷新的刷新信号周期＝2ms/128＝15.6μs。

（3）集中刷新一遍的时间＝0.1μs×128＝12.8μs。

9. SRAM 和 DRAM 分别进行位扩展和字扩展时应如何进行连接？

答：SRAM 的位扩展：将各芯片的$\overline{CE}$（或$\overline{CS}$）端连接在一起，相对应的地址线连接在一起（即所有芯片的 A_0 连接在一起，所有芯片的 A_1 连接在一起，……），数据线各自连接到数据总线上。

SRAM 的字扩展：相对应的低位地址端连接在一起（低位地址的长度由芯片的容量决定），高位地址经译码后分别连到各芯片的$\overline{CE}$（或$\overline{CS}$）端，相应位的数据端连接在一起。

DRAM 的连接方法基本上与 SRAM 相同，但因 DRAM 没有$\overline{CE}$端，而用$\overline{RAS}$代替。各芯片的$\overline{CAS}$可连接在一起，因为不来$\overline{RAS}$时，$\overline{CAS}$不起作用（参见 DRAM 的逻辑框图）。

10. 有些 CPU 向存储器分别传送读、写控制信号（$\overline{RD}$和$\overline{WR}$），而有些 CPU 仅有一个读写信号 R/$\overline{W}$，两者有何区别？

答：在一般情况下，$\overline{RD}$为低电平读选通信号，当进行读操作时$\overline{RD}$＝0，平时处于高电平；$\overline{WR}$为低电平写选通信号，也是在写入时$\overline{WR}$才为 0。因此在不读不写时这两个信号都处于高电位，存储器不工作。

R/$\overline{W}$ 信号是让存储器处于读或写的状态，如果不希望它工作，CPU 还应该有其他信号。例如，用于控制 SRAM 的片选$\overline{CS}$、控制 DRAM 的$\overline{RAS}$信号等。

11. 某机有一个地址空间为 0000H～1FFFH 的 ROM 区域，现在再用 SRAM 芯片（8K×4）组成一个 16K×8 位的 SRAM 区域，起始地址为 2000H，假设 SRAM 芯片有$\overline{CS}$和 R/$\overline{W}$ 信号控制端。CPU 地址总线为 $A15\sim A0$，数据总线为 $D7\sim D0$，控制信号为 R/$\overline{W}$（读/写），$\overline{MREQ}$（当存储器进行读或写操作时，该信号指示地址总线上的地址是有效的）。要求画出逻辑图。

答：地址空间为 0000H～1FFFH 的 ROM 容量为 8KB(字长为 1B)，ROM 和 SRAM 的地址为 13 位(A_0～A_{12})，A_{13}～A_{15}经译码后控制 ROM 和 SRAM 的$\overline{CS}$端。$\overline{CS_0}=\overline{A}_{15}\overline{A}_{14}\overline{A}_{13}$，$\overline{CS_1}=\overline{A}_{15}\overline{A}_{14}A_{13}$，$\overline{CS_2}=\overline{A}_{15}A_{14}\overline{A}_{13}$。图 4.5 为逻辑图。

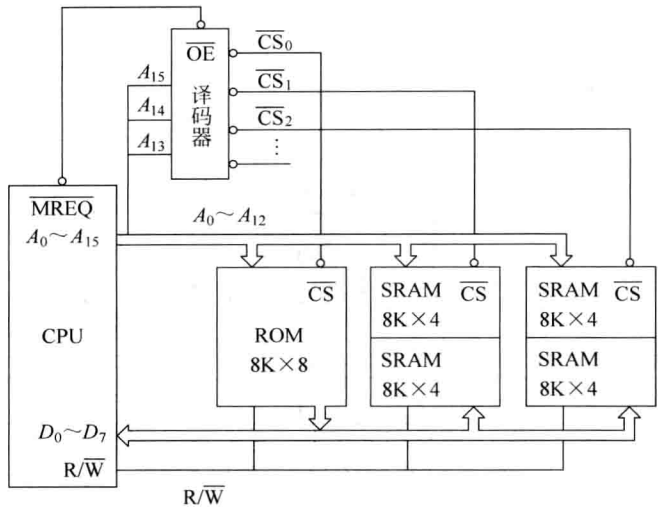


图 4.5　ROM 和 SRAM 存储器逻辑图

12. 某 CPU 提供 16 条地址线(A_0～A_{15})、8 条数据线(D_0～D_7)及 R/$\overline{W}$ 控制信号，目前使用的存储器容量为 8KB，其中 4KB 为 ROM，采用 2K×8 位的芯片，其地址范围为 0000H～0FFFH；4KB 为 DRAM，采用 4K×2 位芯片，其地址范围为 4000H～4FFFH。问：

(1) 需 ROM 和 DRAM 芯片各多少片？

(2) 画出 CPU 与存储器之间的连接图(译码器自定)。

答：(1) ROM 2 片和 DRAM 4 片。

(2) 最大存储空间为 64KB，设 A_0 为地址线的最低位，则访问 ROM 的条件为 A_{15}～A_{12}=0000，访问 RAM 的条件为 A_{15}～A_{12}=0100。译码器可以有多种考虑，如考虑到今后有可能扩展到 64KB，则选择全译码方案(4 输入 A_{12}～A_{15}，16 输出 $\overline{Y}_0$～$\overline{Y}_{15}$)，并将$\overline{Y_0\cdot\overline{A_{11}}}$和$\overline{Y_0\cdot A_{11}}$分别接到 2 个 ROM 的$\overline{CE}$端。

今将$\overline{Y_0\overline{A_{11}}}$和$\overline{Y_0Y_{11}}$变换如下：

$$\overline{Y_0\overline{A_{11}}}=\overline{Y}_0+\overline{\overline{A}_{11}}=\overline{Y}_0+A_{11}$$

$$\overline{Y_0A_{11}}=\overline{Y}_0+\overline{A}_{11}$$

并将变换后的信号送 ROM 的$\overline{CE}$端。

译码器的输出受 CPU 的访存请求信号$\overline{MREQ}$控制，在不访问时，译码的所有输出端均为高电位。

为了控制 DRAM 的读写，在 CPU 和存储器芯片之间还要增加一些控制电路，在$\overline{MREQ}$信号的作用下，生成$\overline{RAS}'$和$\overline{CAS}'$信号，并将 A_0～A_{11}信号分两次(A_0～A_5，A_6～A_{11})送往存储器芯片。芯片的$\overline{RAS}$端接收的是$\overline{Y_4\cdot\overline{RAS}'}=\overline{Y}_4+\overline{RAS}'$，芯片的$\overline{CAS}$端接收的是$\overline{CAS}'$信号。

图 4.6 是 CPU 与存储器之间的连接图。

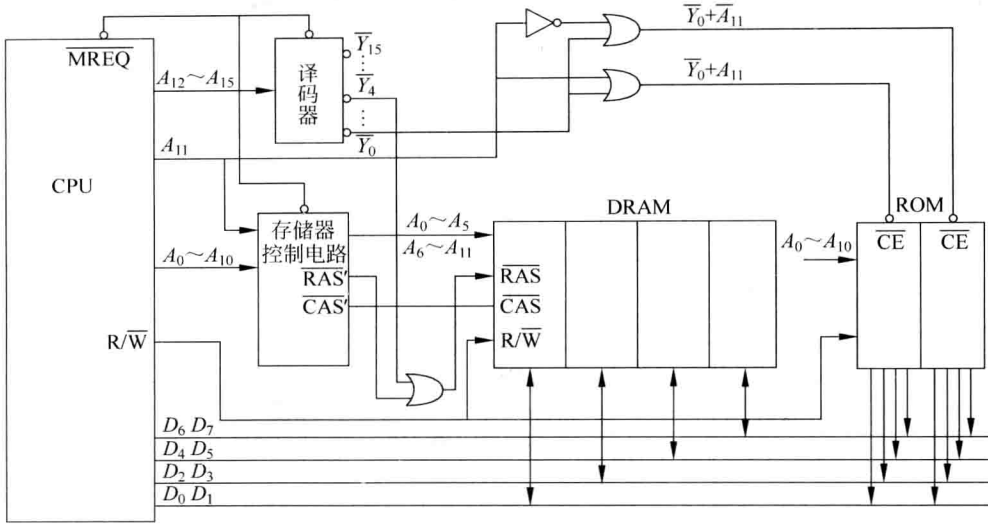


图 4.6　CPU 与存储器之间的连接图

13. 某 CPU 有 16 条地址线和 8 条数据线。从 0 地址开始已有 40KB 内存，现要在 40KB 地址空间之后再增加 8KB SRAM(地址连续)。如用 4K×8 位 SRAM 芯片来扩容，试设计 CPU 与 8KB SRAM 的连接图。

答：存储容量 1KB 实际值为 $2^{10}=1024$B。设 16 条地址线为 A_{15}～A_0(A_{15}为最高位)。

增加的 8KB RAM 其地址范围为 1010 $\underbrace{00\cdots0}_{12位}$～1011 $\underbrace{11\cdots1}_{12位}$。需 2 片 SRAM 芯片，其片选为 $A_{15}\overline{A}_{14}A_{13}\overline{A}_{12}$ 和 $A_{15}\overline{A}_{14}A_{13}A_{12}$。CPU 与新增的存储器连接图如图 4.7 所示。

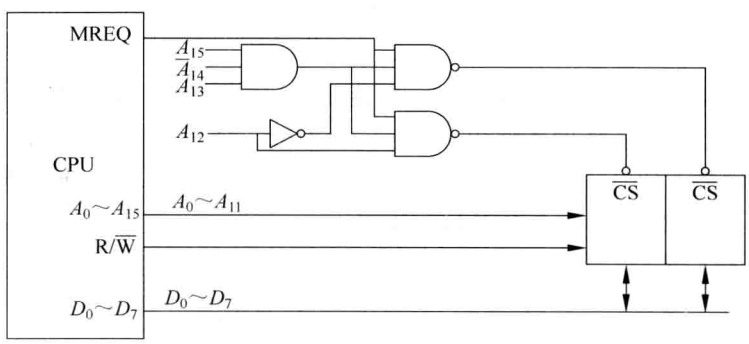


图 4.7　存储器扩充容量

14. 某微机的寻址范围为 64KB，由 8 片 8KB 的芯片组成，每个芯片由 8K×8 位组成。

(1) 请写出每个芯片的寻址范围。

(2) 如果运行时发现不论往哪个芯片存放 8KB 数据，寻址地址大于等于 A000H 起始地址的存储芯片都有相同的数据，请分析故障原因。

(3) 若发现译码器中的地址线 A_{13} 与 CPU 断线，并搭接到高电平的故障，问后果如何？

答：(1) 8 个芯片的寻址范围如下(16 条地址线 A_{15}～A_0)。

芯片 0	0000H～1FFFH
芯片 1	2000H～3FFFH
芯片 2	4000H～5FFFH
芯片 3	6000H～7FFFH
芯片 4	8000H～9FFFH
芯片 5	A000H～BFFFH
芯片 6	C000H～DFFFH
芯片 7	E000H～FFFFH

(2) 芯片 5、6、7 的$\overline{CE}$端接地(或接低电平),所以无论向哪个芯片写入内容,同时向芯片 5、6、7 写入同样内容。

(3) A_{15}、A_{14}、A_{13}是控制芯片的片选信号的,其中 A_{13}为 0 时,可控制芯片 0、2、4、6 中的一片进行读写操作;$A_{13}=1$ 可控制芯片 1、3、5、7 中的一片进行读写操作,这是正常工作时的情况。当发生 A_{13}搭接到高电位的故障时,芯片 0、2、4、6 就不可能再工作,无论 CPU 要将数据送到芯片 0 或芯片 1,芯片 1 都会接收数据,其他芯片的情况类似。同样,无论 CPU 要从芯片 0 还是芯片 1 取出数据,芯片 1 都会读出数据。其结果不仅是存储容量减半,而且数据会出错。

15. 在计算机中以 DRAM 组成的存储器,要求其进行读/写操作的来源有哪些?如果它们同时发出访问请求该如何处理?

答:要求 DRAM 读写的请求来自 CPU、外部设备及自身的刷新请求。刷新是以"读操作"的形式进行的,对 DRAM 单管存储单元进行读操作是一种破坏存储内容的操作,在存储器内部每一列上有一个读放,在读出的同时,读放又使该单元的存储信息自动恢复,因此刷新时,只要依次改变行地址,轮流对存储矩阵的每一行读出,该行的所有单元同时得以再生,当把所有行轮流读一遍后,就完成了整个存储器的刷新。外部设备(例如磁盘机、磁带机)与主存之间直接传送信息,对存储器的读写请求是由外部设备提出的。

在存储器控制器中,设置有排队线路,对多个读写控制请求信号进行排队,其优先次序的排队原则是:

(1) 不丢失信息,即不出错,这是首先要考虑的。

(2) 速度要求高的来源先处理。

(3) 存储器正在执行读写操作的过程时,处理完后再接受新的读写请求。

刷新和外部设备的访存要求,如果不及时处理,信息会丢失,因此在排队时,其优先级应高于 CPU 的请求,但有时会有一些折中方案,刷新请求可以有条件地让 CPU 请求先执行,例如可以采取下述方案:如果存储器不工作时,同时有访存请求和刷新请求,可以先保证访存;如果存储器在工作时又有新的访存请求和刷新请求,则刷新请求优先。

16. 现有一台笔记本电脑,存储器容量为 256MB,请问访存地址的最小长度为多少位?请写出访存的最小地址值和最大地址值。

答:访存地址的最小长度为 28 位。

访存的最小地址值为 0000000H,最大地址值为 FFFFFFFH(即为 $2^{28}-1=16^7-1$)。H 表示十六进制数值。

17. 动态随机存储器(DRAM)和同步动态随机存储器(SDRAM)有何主要差别?

答：处理器与 DRAM 之间的读/写操作是异步进行的，当处理器发出读写命令后，等待存储器完成读/写操作的回答信号后才能进行下一步操作，而且等待时间不是固定的。

处理器与 SDRAM 之间的读/写命令到 SDRAM 并锁存在 SDRAM 中，在经过一定数量的时钟周期（其值由芯片生产厂家提供）后，SDRAM 完成读/写操作，在此期间，处理器可以进行其他工作，不必处于空闲等待状态。

18. 目前计算机主要采用何种 DRAM？

答：计算机主要采用 DDR3 存储器芯片，是从 SDRAM 发展来的。其发展过程为 SDRAM→DDR→DDR2→DDR3。

DDR 是双数据传送速率的 SDRAM，与 SDRAM 不同的是时钟的上升沿和下降沿都能读/写数据（由 1 位提高到 2 位）。

DDR2 在时钟的上升沿和下降沿各能读写 2 位，DDR3 在时钟的上升沿和下降沿各能读写 4 位（即每个时钟能读/写 8 位）。

19. 假设芯片内部工作的时钟频率为 200MHz，如何计算 DDR 存储器和 DDR3 存储器的数据传输率？又 CL 值表示什么概念？

答：若 DDR 内部时钟频率为 200MHz，则用 DDR400 表示。由它组成的 64 位数据传输率＝200M×2×64b＝400M×64b＝3200MB/s＝3.2GB/s，以 PC3200 表示。

若 DDR3 内部时钟频率也为 200MHz，则用 DDR3-1600 表示，由它组成的 64 位数据传输率＝200M×8×64b＝1600M×64b＝12800MB/s＝12.8GB/s，以 PC3-12800 表示。

上述芯片是以同步动态随机存储方式工作的，当按收到读/写命令后，等待一定数量的时钟周期后开始进行读/写操作，该数量由生产厂家提供，称为 CL 值，目前 DDR 的 CL 为 2 或 2.5，DDR3 的 CL 为 5～11。以 DDR3-1600 为例说明芯片的读写时间：DDR3 的内部时钟频率为 1600MHz/8＝200MHz，时钟周期为 1/200MHz＝5ns，如果 CL 值为 6，读写操作时间为 7ns，则总的读写延迟时间为 5ns×6＋7ns＝37ns，由于其时间是固定的，所以可采用同步工作方式。

20. 假如 CPU 按以下地址相继访问存储器（以十进制表示）：

$$0000_{10}, 0005_{10}, 0010_{10}, \cdots, 0285_{10}$$

存储器为 4 体交错存储器，求该存储器比单体存储器平均访问速率提高多少？

答：按上述次序访问存储器，访问地址的间隔为 5，在按地址低位选择存储体的模 4 交叉存储器中，相继访问的地址在不同存储体中，每隔 1/4 存储周期启动一次访问操作（轮流启动各存储体，对每个存储体而言，仍是一个周期启动一次访问操作），所以该存储器比单体存储器平均访问速率提高到 4 倍。但第 1 次访存仍需 1 个存取周期，以后每隔 1/4 周期可访问一次。

实现时，还可能存在问题，参见本章第 25 题。

21. 为什么在多体交叉存储器中，按地址低位来选体而不用地址高位来选体？

答：存储器存放的程序一般是按地址顺序存放的，数据虽不如程序集中，但一般也不会分布在很大的范围内，这就是存储的局部性原理，所以按地址低位选体可以提高轮流访问各个体的机率，在上一次访存未完成时就可启动下一次访存操作。如按高位选体，则各存储器重叠工作的可能性很小，不能发挥交叉存储器的作用。

22. 如果从 4 体交叉存储器取出 20 个地址连续编号的数据，一个体的存取周期为 t，总

共需要多少时间？

答：第 1 个数据取出的时间为 t，以后每隔 $t/4$ 即可取出 1 个数据，总共需要时间为

$$t+\frac{1}{4}t\times 19=5.75t$$

23. (1) 今有一个 4 体交叉存储器，如何根据主存给出的地址选择存储单元？

(2) 假设一个存储体的存取周期为 T，字长为 32 位，数据总线的宽度也为 32 位，CPU 访问存储器的地址是连续的，问如何分配各存储体的工作时间？

答：(1) 设地址长度为 n 位，由第 0 位、第 1 位译码输出控制存储体的读写，由第 2～$n-1$ 位选择体内地址，如图 4.8 所示。图中 AR 为地址寄存器，DR 为数据寄存器。

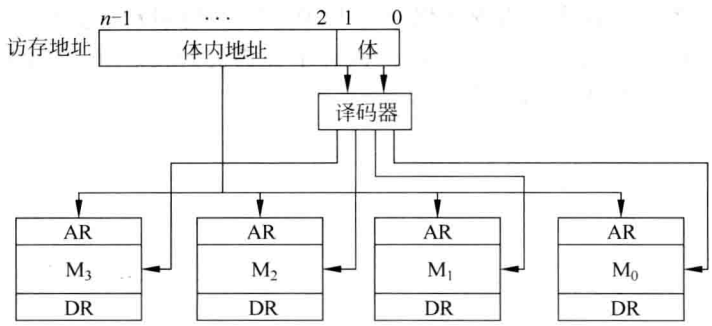


图 4.8　多体交叉存储的编址

(2) 存储体的存取周期为 T，每隔 $T/4$ 时间轮流启动各存储体的读写，如图 4.9 所示。

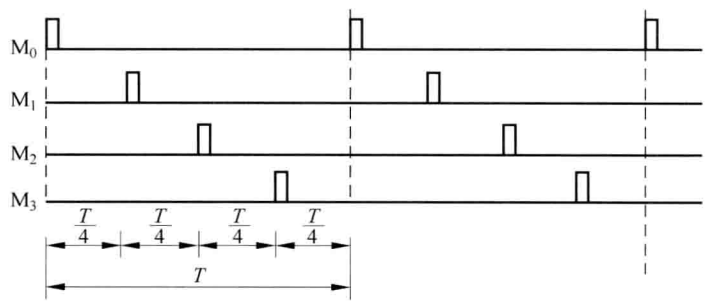


图 4.9　交叉访问存储体的工作时间

24. 在交叉存储器中，影响相关访存速度的因素有哪些？

答：在交叉存储器中，当不能轮流访问各存储体时，将影响访存速度。影响的因素有：

(1) 程序是按顺序存放在存储器中，为轮流访问各存储体提供了条件，但当频繁执行转移指令或程序中断时会打破这种局面。

(2) 执行程序时插入数据地址会影响地址的连续性。

25. 按本章第 20 题所述的次序访问交叉存储器是否一定能使平均访问速率提高到单体存储器的 4 倍？

答：在一般交叉存储器中，如果访问地址为 0000 的存储单元，那么相隔 1/4 周期后存储器自动访问地址为 0001 的单元，再隔 1/4 周期后访问地址为 0002 的单元……此时 CPU 还未发出下一个取指(或取数)的命令，属于预取性质。因此实际上在访问 0000 单元后，隔

1/4 周期只能取到 0001 单元的内容而取不到 0005 单元的内容，除非在一定条件下采取一些特殊措施才能将速率提高到 4 倍。

26. 设有一个 4 体交叉存储器，但使用时经常遇到连续访问同一存储体的情况，会产生怎样的结果？

答：4 体交叉存储器连续访问同一存储体，其作用与单体存储器相似，相邻两次访问的时间间隔等于存储器的存取周期。

27. 在计算机的主存中，常常设置一定的 ROM 区，试说明设置 ROM 区的目的。

答：计算机加电时，需要执行一段程序和设置一些初始数据，如果这些程序和数据放在 RAM 区，那么断电后会丢失，所以把它放在 ROM 区。至于 ROM 区的大小则由操作系统决定，可以将操作系统中很小一段程序保存在 ROM 中，加电启动计算机后依靠这段程序将操作系统由磁盘调入主存。由于 ROM 中的内容不会丢失，所以 ROM 区大些较为有利，但 ROM 的每位价格比磁盘大得多。

第5章 指令系统

1. 什么是指令？什么是指令系统？

答：由操作码和地址码组成的，能由计算机硬件执行其规定操作的一条命令，称为指令。指令和数据在计算机中都是以二进制代码形式表示的，不同的操作码代表不同的指令。通常在一台计算机中有几十条乃至几百条指令。计算机中所有指令的集合称为指令系统。

2. 一条指令通常包含哪些信息？

答：(1) 操作码。具体指定操作内容，例如执行加法运算或控制程序流的转移等。

(2) 操作数来源。指定操作数或操作数的地址。

(3) 操作结果地址。保存处理结果的地址。

(4) 下一条执行的指令地址。在大多数情况下，程序是顺序执行的，此时不需要指出下一条指令的地址，仅当执行改变程序流的转移类指令时，需要给出下一条执行的指令地址。

上述(2)、(3)和(4)的内容均由地址码给出，或者由操作码隐含，详见第3题答案。

3. 根据指令的地址码部分所给出的地址个数，如何划分指令？

答：有零地址指令、一地址指令、二地址指令、三地址指令和多(三个以上)地址指令等。

零地址指令：仅有操作码，无地址码。这类指令不需要操作数，或操作数地址是默认的。

一地址指令：指令中有操作码和1个地址，这类指令只需要1个操作数，或其他操作数的地址是默认的。

二地址指令：指令中有操作码和2个地址，其中1个地址还用来保存处理结果。这是最常见的指令格式。

三地址指令：指令中有操作码和3个地址。分别指出两个操作数和存放处理结果的地址。

多地址指令：某些大型机中往往设置一些功能很强的用于处理成批数据的指令，此时有可能需要3个以上的地址或控制信息。

一台计算机可能具有上述指令格式中的一部分。

4. 操作码有哪几种表现形式？

答：操作码有以下两种形式。

(1) 固定长度的操作码：如果操作码的长度为 k 位，则可以表示 2^k 条不同的指令。

(2) 可变长度的操作码：计算机采用几种长度的操作码，通常将使用频率较高的指令设计为操作码较短的指令，以减少程序的存储量和CPU与存储器之间的信息流通量。

5. 请解释什么是霍夫曼编码？

答：采用霍夫曼编码的操作码是一种根据程序中指令出现频率高低而赋以不同长度的操作码。

设一台机器有7条指令($I_1 \sim I_7$)，每条指令的使用频率如表5.1所示，差别很大，若用定长操作码，则需3位。霍夫曼编码的操作码可按表5.2安排，A方案全按霍夫曼编码，最长

需 6 位；B 方案基本按霍夫曼编码，但后 4 种状态应用一般的二进制编码，最长的操作码从 6 位缩短成 5 位。

表 5.1　指令使用频率

指令	使用频率	指令	使用频率
I_1	0.45	I_5	0.03
I_2	0.30	I_6	0.01
I_3	0.15	I_7	0.01
I_4	0.05		

A 方案操作码的平均长度为 $0.45\times1+0.30\times2+0.15\times3+0.05\times4+0.03\times5+0.01\times6+0.01\times6=1.97$ 位。

表 5.2　两种霍夫曼编码的操作码方案比较

指令	使用频率	A 方案		B 方案	
		操作码	长度	操作码	长度
I_1	0.45	0	1 位	0	1 位
I_2	0.30	10	2 位	10	2 位
I_3	0.15	110	3 位	110	3 位
I_4	0.05	1110	4 位	11100	5 位
I_5	0.03	11110	5 位	11101	5 位
I_6	0.01	111110	6 位	11110	5 位
I_7	0.01	111111	6 位	11111	5 位

同理可得 B 方案操作码的平均长度为 2.00 位。

上述两种方案，其平均长度都比定长操作码的 3 位要短得多。但是每个操作码的长度都不同，不便于译码，也不便于实际使用，所以只能采用它的变形。参照 B 方案，如果前 3 位状态也用二进制译码表示，则可得如表 5.3 所示的操作码编码，具有两种长度，其平均长度为 2.20 位。这种表示方法就是扩展操作码法。

表 5.3　扩展操作码

指令	使用频度	操作码	长度
I_1	0.45	00	2 位
I_2	0.30	01	2 位
I_3	0.15	10	2 位
I_4	0.05	1100	4 位
I_5	0.03	1101	4 位
I_6	0.01	1110	4 位
I_7	0.01	1111	4 位

一般设计计算机时，指令的长度选择字节的整数倍（可包含单字节指令），以便于存储和读取，否则控制起来太复杂。

6. 某计算机指令字长为 16 位，采用扩展操作码，操作数地址需 4 位，该指令已有三地址指令 M 条，二地址指令 N 条，没有零地址指令，问最多还可有多少条一地址指令？

答： 三地址指令操作码长度为 4 位，二地址指令的操作码可扩展到 8 位，一地址指令的

操作码可扩展到 12 位。

一地址指令最多为$((2^4-M)\times 2^4-N)\times 2^4$条。

式中2^4-M是减去三地址指令后允许使用的 4 位操作码，对应每个操作码最多可以有2^4条二地址指令，再减去 N 后是一地址允许使用的 8 位操作码，对应每个操作码可以有2^4条一地址指令。

计算过程中不应出现负值，否则会因为 M 或 N 值过大而无法实现。

7. 某指令系统指令长 16 位，每个操作数的地址码长 6 位，指令分无操作数、单操作数和双操作数 3 类。若双操作数指令有 K 条，无操作数指令有 L 条，问单操作数指令最多可能有多少条？

答：本指令系统的操作码位数＝指令长度－地址码长度＝16－2×6＝4 位。设单操作数指令最多有 N 条，则

$$L=((2^4-K)\times 2^6-N)\times 2^6$$

$$N=(2^4-K)\times 2^6-L/2^6$$

如果 $L/2^6$ 不是整数，则取大于 $L/2^6$ 的最小整数。

8. 各计算机的指令系统差别很大，但一般计算机中至少具有哪几种指令类型？

答：一般计算机中都具有算术逻辑运算指令、转移类指令和停机（或动态停机）指令。

9. 哪些指令根据运算结果置状态位？一般设置哪些状态位？其用处何在？

答：一般算术运算指令根据运算结果置状态位，一般设置的状态位有 N（结果为负数时置 $N=1$，否则 $N=0$）、Z（结果为 0 时，置 $Z=1$，否则为 0）、V（结果溢出时，置 $V=1$，否则为 0）和 C（运算有进位或借位时 $C=1$，否则为 0）。状态位主要用于控制程序转移与否。其中状态位 C 还用于双字长算术运算，在无直接执行双字长运算指令的机器中，如果需要执行双字长运算，用单字长运算指令分别对高位和低位进行运算，当低位有进位或借位时，高位要作相应的处理。

下面以双字长加法运算为例来说明 C 的作用。

假设在寄存器 R_1、R_2 和 R_3、R_4 分别存放两个双字长操作数，其中 R_1、R_3 为高位。加法运算可分为两种情况讨论：

（1）假设机器设置有 ADD（加法指令）和 ADC（加进位的加法指令），则执行下列两条指令后在 R_3、R_4 中得到运算结果：

```
K:     ADD  R2,R4    ;低位相加，R4←(R2)+(R4)，并根据运算结果置进位位 C
K+1:   ADC  R1,R3    ;高位相加，并加进位位 C，R3←(R1)+(R3)+C
```

（2）假设机器仅设置 ADD 指令，而没有 ADC 指令，则应执行下列程序：

```
K:     ADD  R2,R4    ;低位相加，R4←(R2)+(R4)，并根据运算结果置进位位 C
K+1:   BCC  K+3      ;如 C=0，程序转移到 K+3；如 C=1，顺序执行下一条指令
K+2:   ADD  #1,R1    ;R1←(R1)+1，#1 表示立即数 1
K+3:   ADD  R1,R3    ;高位相加，R3←(R1)+(R3)
```

在 R_3、R_4 中得到运算结果。

溢出时（$V=1$），发溢出中断，进行溢出故障处理。

10. 某计算机设置有单字长算术运算指令，请编写实现双字长减法运算的子程序。

答：假设在 R_1、R_2 和 R_3、R_4 中分别存放双字长被减数和减数，其中 R_1、R_3 为高位，R_2、R_4 为低位。双字长减法运算的步骤如下：

(1) 低位相减：若不够减，置 $C=1$；够减，置 $C=0$。

双定长数据的低位部分是不带符号位的数值，如果不够减($C=1$)，则向被减数的高位借位。

(2) 高位相减：首先判断 C 值，如果 $C=1$，则将被减数的高位部分减 1，或者将减数的高位部分加 1；如果 $C=0$，则不作此项处理。然后进行高位相减的运算。

子程序设计如下：

```
K:     SUB  R2,R4      ;R4←(R2)-(R4),并根据运算结果置借位 C
K+1:   BCC  K+3        ;若 C=0,转移到 K+3
K+2:   ADD  #1,R3      ;R3←(R3)+1,减数+1
K+3:   SUB  R1,R3      ;R3←(R1)-(R3)
```

在 R_3、R_4 中得到两数之差。

11. 算术移位和逻辑移位有何不同？

答：算术移位实现右移时，最高位(符号位)保持不变。当数据用补码表示时，如果右移 n 位，相当于除以 2^n；左移 n 位，相当于乘以 2^n。设有补码 11100，右移 1 位后为 11110，移位前的数值是 −0100，移位后的数值为 −0010，相当于除以 2。

逻辑移位实现右移时，一般最高位补以 0。

算术左移和逻辑左移的结果一般是相同的。

但是逻辑移位还可以有进位状态位 C 参与的循环移位。

12. 计算机中常用的寻址方式有哪些？

答：有存储器直接寻址、寄存器直接寻址、基址寻址、变址寻址、间接寻址、相对寻址和立即数寻址。有的计算机设计有堆栈寻址方式。

13. 基址寄存器的内容为 2000H(H 表示十六进制)，变址寄存器内容为 03A0H，指令的地址码部分是 3FH，当前正在执行的指令所在地址为 2B00H，请求出变址编址(考虑基址)和相对编址两种情况的访存有效地址(即实际地址)。

答：变址编址的有效地址＝(基址寄存器)＋(变址寄存器)＋3F＝2000＋03A0＋3F＝23DFH。

相对编址＝(PC)＋3F＝2B00＋3F＝2B3FH。

注：(基址寄存器)表示基址寄存器的内容。

14. 接上题。

(1) 设变址编址用于取数指令，相对编址用于转移指令，存储器内存放的内容如下：

地　址	内　容
003FH	2300H
2000H	2400H
203FH	2500H
233FH	2600H
23A0H	2700H
23DFH	2800H

2B00H　　063FH

请写出从存储器中所取的数据以及转移地址。

(2) 若采取直接编址，请写出从存储器取出的数据。

答：(1) 从存储器所取的数据＝(23DF)＝2800H。

转移地址为 2B3FH。

(2) 若机内没有基址寄存器，所取数据＝(3F)＝2300H；若机内设有基址寄存器，所取数据的地址＝(基址寄存器)＋3F＝203FH，所取数据＝(203F)＝2500H。

15. 假设寄存器 R 中的数值为 1000，地址为 1000 的存储器中存储的数据为 2000，地址为 2000 的存储单元中存储的数据是 3000，PC 的值是 4000，说出下述各寻址方式取得的数据是什么。

(1) 寄存器寻址：R　　(2) 寄存器间接寻址：(R)

(3) 直接寻址：1000　　(4) 存储器间接寻址：(1000)

(5) 相对寻址：－2000(PC)　　(6) 立即数寻址：＃2000

答：各寻址方式取得的数据如下：

(1) (R)＝1000

(2) (1000)＝2000

(3) (1000)＝2000

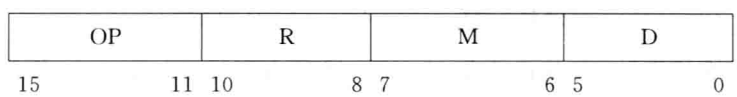
(4) (2000)＝3000

(5) (PC)＋(－2000)＝4000－2000＝2000　(地址)

地址 2000 中的内容＝3000。如果执行转移指令，2000 即为转移地址。

(6) 符号＃表示紧跟它的是数据，其值为 2000。

16. 某计算机指令长度为 16 位，指令格式如下：



其中，OP 为操作码(5 位)；R 为寄存器地址(3 位)，用来指定一个源操作数地址和目的地址；另一个源操作数根据寻址方式得到，M 为寻址方式特征位(2 位)，规定如下：

M＝00　为立即数寻址，D 为立即数；

M＝01　为相对寻址，D 为位移量；

M＝10　为变址寻址，D 为位移量。

今执行加法指令，PC＝001000，R＝R_0，其内容为 000015，变址寄存器的内容为 001002，D＝01 都用八进制表示。

存储器内容如下：

地址	内容
001000	xxxx01
001001	001050
001002	001150
001003	001250
⋮	⋮

002001	002006
002002	002016

当该加法指令的寻址方式为立即寻址、相对寻址和变址寻址时，分别写出指令执行之后 R_0 和 PC 的内容。状态寄存器的内容是否相同？

答：字长 16 位，假设存储器按字编址，因此执行加法指令之后 PC＝001000＋1＝001001（上述各种寻址方式，执行加法指令之后 PC 值都相等）。

立即寻址：(R_0)＝000015＋01＝000016（01 为立即数）。

相对寻址：操作数有效地址＝001001＋01＝001002（因执行时，PC 已加 1），(R_0)＝000015＋001150＝001165。

变址寻址：操作数有效地址＝001002＋01＝001003，(R_0)＝000015＋001250＝001265。

根据运算结果置状态位：$N=0$，$Z=0$，$V=0$，$C=0$。本题的 3 种寻址方式状态位相同。

17. 设运算器某寄存器的内容为 11110001，进位 $C=0$，今分别执行逻辑左移、算术左移、逻辑右移、算术右移、循环左移和循环右移（C 参与循环移位）指令，移 1 位，请写出执行移位指令后 C 与寄存器的内容。

答：

	C	寄存器
移位前	0	11110001
逻辑左移	0	11100010
算术左移	0	11100010
逻辑右移	0	01111000
算术右移	0	11111000
循环左移	1	11100010
循环右移	1	01111000

18. 存储器堆栈指针 SP 的内容是$(100)_{16}$，栈顶内容是$(1000)_{16}$，一条双字长子程序调用指令（直接寻址）位于存储器地址$(2000)_{16}$和$(2001)_{16}$处，$(2001)_{16}$的内容是转移地址字段，内容为$(3000)_{16}$，求出以下情况下 PC、SP 和栈顶内容。

(1) 子程序调用指令读取之前。

(2) 子程序调用指令执行以后。

(3) 从子程序返回之后。

注：本题中的$(100)_{16}$表示 100 是十六进制数（或地址）。其余相同。

答：(1) (PC)＝$(2000)_{16}$；(SP)＝$(100)_{16}$；栈顶内容＝$(1000)_{16}$。

(2) (PC)＝$(3000)_{16}$；(SP)＝$(100)_{16}-(2)_{16}$＝0FEH（子程序返回地址压入堆栈，堆栈指针－2）；子程序返回地址＝$(2000)_{16}+(2)_{16}=(2002)_{16}$，即为栈顶内容。

(3) (PC)＝$(2002)_{16}$；(SP)＝$(100)_{16}$；栈顶内容＝$(1000)_{16}$。

19. 说明不操作指令 NOP、等待指令 WAIT 和停机指令的功能。

答：NOP 指令除了执行 PC＋1→PC（如果 NOP 为单字节指令）操作外，不进行其他操作，然后继续执行下一条指令。

WAIT 指令也不进行实质性操作，但也不停机，而是不断重复执行该指令，当激活条件满足（例如中断请求）后机器继续执行程序。

HALT 指令使机器停机(例如封锁主频脉冲),恢复工作需重新启动。

20. (1) 属于转移类的指令有哪些? (2) 无条件转移指令和条件转移指令有何不同? (3) 转移指令和转子指令又有何不同? (4) 执行转子指令和响应中断有何区别?

答:(1) 属于转移类的指令有无条件转移指令、条件转移指令、调用指令(即转子程序指令)、返回指令和软中断指令(陷阱指令)。

(2) 无条件转移指令是必转指令,条件转移指令根据是否满足条件(通常由 N、Z、V 或 C 提供条件)而决定是否转移,如果条件不满足,则仍顺序执行下一条指令。

(3) 转移指令不保留本条指令的地址,转子指令则保留本条指令地址(实际上是 PC+1,见第 6 章指令执行流程),当子程序执行完毕后返回原程序继续运行。

(4) 转子指令是在程序中安排的,而响应中断是当计算机产生异常情况或有 I/O 中断请求时 CPU 作出的反应。响应中断后执行中断处理程序,也要保留返回地址,中断程序执行完毕后返回原程序继续运行。

21. 条件转移指令通常用算术指令建立的条件码来控制程序的执行方向。今假设已由算术指令执行了 $A-B$ 操作而建立了状态,问在下列各种情况下如何利用条件码 N、Z、V、C 建立转移条件?

(1) $A=B$ 时转移。

(2) A 和 B 为无符号数,当 $A>B$ 时转移。

(3) A 和 B 为带符号数,当 $A>B$ 时转移。

答:(1) $Z=1$ 时转移。

(2) $\overline{C+Z}=1$ 时转移。

如果 $A>B$,即 $A-B>0$,不产生借位,所以 $C=0$,而且 $A-B$ 的结果不为 0,所以 $Z=0$,也就是说,$C+Z=0$,故$\overline{C+Z}=1$ 时转移。

(3) $\overline{(N\oplus V)+Z}=1$ 时转移。

如果 $A>B$,则 $A-B$ 的结果有两种可能:

① 如果不溢出(即 $V=0$),差值为正(即 $N=0$)且结果不为 0(即 $Z=0$)。

② 如果溢出(即 $V=1$),$A-B$ 的符号位为负(即 $N=1$),且结果不为 0(即 $Z=0$)。

综合上述两种情况得出$(N\oplus V)+Z=0$ 时 $A>B$,即$\overline{(N\oplus V)+Z}=1$ 时转移。

22. 某机主存为 64KB,CPU 内有 8 个 16 位通用寄存器(其中 4 个又可以当成 8 个 8 位通用寄存器)。该机指令系统有 64 条指令,全部为双地址指令,且必有一个操作数 R_1 在寄存器中,另一操作数采用下列寻址方式:

(1) 寄存器直接寻址。

(2) 寄存器间接寻址(用 16 位寄存器)。

(3) 存储器直接寻址。

(4) 变址寻址(用任意一个 16 位寄存器做变址寄存器,位移量 16 位)。

要求:

(1) 设计该计算机的指令格式,并列出各种类型的指令格式。

(2) 写出各种寻址方式的有效地址。

答:该计算机有 64 条指令,操作码 6 位。寄存器地址用 4 位表示,其中 1 位用于表示字或字节,3 位表示 8 个寄存器。寻址方式 2 位,用以表示 4 种寻址方式。存储器容量为

64KB，其地址为 16 位。

设计两种指令长度：16 位和 32 位。

(1) 指令格式：

- 寄存器直接寻址

6	2	4	4
操作码	00	R_1	R_2

00 表示寄存器直接寻址的寻址方式。

- 寄存器间接寻址

6	2	4	4
操作码	01	R_1	R_2

01 表示寄存器间接寻址的寻址方式。

- 存储器直接寻址

6	2	4	4
操作码	10	R_1	(空)
存储器地址			

10 表示存储器直接寻址的寻址方式。

- 变址寻址

6	2	4	4
操作码	11	R_1	R_2
位移量			

R_2 为变址寄存器地址。11 表示变址寻址的寻址方式。

(2) 有效地址：

- 寄存器直接寻址　R_1，R_2。
- 寄存器间接寻址　R_1，(R_2)。
- 存储器直接寻址　R_1，存储器地址(指令的第 2 个字)。
- 变址寻址　R_1，(R_2)＋位移量(指令的第 2 个字)。

23. 一台模型机共有 7 条指令，主频 25MHz，各指令的使用频率与 CPI 如表 5.4 所示，该机有 8 位和 16 位两种指令字长，采用 2-4 扩展操作码。8 位字长指令为寄存器-寄存器(R-R)二地址类型，16 位字长指令为寄存器-存储器(R-M)二地址变址寻址类型(地址码范围是－128～127)。

表 5.4　指令的使用频率和 CPI

指令(字长)	使用频率/%	执行一条指令的周期数(CPI)
I_1(8 位)	35	1
I_2(8 位)	25	2
I_3(8 位)	20	2
I_4(16 位)	10	2
I_5(16 位)	5	1
I_6(16 位)	3	2
I_7(16 位)	2	2

(1) 计算该机的 MIPS 速率(即计算每秒能运行多少个百万条指令)。

(2) 计算操作码的平均码长。

(3) 设计该机的两种指令格式，标出各字段位数并给出操作码编码。

(4) 该机允许使用多少个可编址的通用寄存器和多少个变址寄存器？

(5) 如何计算存储器有效地址？

答：(1) 平均 CPI＝0.35×1＋0.25×2＋0.20×2＋0.10×2＋0.05×1＋0.03×2

$+0.02\times2=1.6$

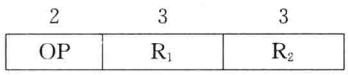
$$速率=\frac{25\text{MHz}}{1.6}=15.6\text{MIPS}$$

(2) 操作码的平均长度$=2$ 位$\times(0.35+0.25+0.2)$

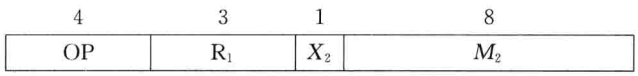
$+4$ 位$\times(0.10+0.05+0.03+0.02)$

$=2.4$ 位

(3) 指令格式(由操作码来区分 R-R 型指令和 R-M 型指令)



R-R 型($OP=\overline{11}$)



R-M 型($OP=11\times\times$)

操作码 I_1:00 I_2:01 I_3:10 I_4:1100 I_5:1101 I_6:1110 I_7:1111

(4) 允许 8 个通用寄存器和 2 个变址寄存器。

(5) 存储器有效地址$=(X_2)+M_2$,M_2 的最高位为符号位。如果(X_2)的字长>8 位,则在 M_2 的前面按符号位补充若干位,使其长度与变址寄存器的字长相等。

24. 指令中地址码的位数与直接访问的存储器空间有什么关系?字寻址计算机和字节寻址计算机在地址码安排上有何区别?如果指令系统支持对字节、字、双字运算而又不要求对准边界,则会产生什么影响?如果对准边界,其相应的地址应符合什么条件?

答:设地址码有 n 位,则可直接访问的存储器空间为 2^n。字寻址计算机的存储器空间为 2^n 个字,假如字长为 16 位(2B),则可访问的存储器空间在同样长度的地址码情况下与字节寻址相比可增加一倍。

如果访问一次存储器可取出一个字(16 位),且按字节寻址,那么在不对准边界的情况下,取一个字有可能要访问 2 次存储器。

取双字有可能要访问 3 次存储器,增加了控制的复杂性,还影响到速度。假如一个字在存储器中安排的位置如图 5.1 的阴影部分所示,那么取一个字要访问 2 次存储器。如果对准边界,存取字的地址最低位应为 0,存取双字的地址最低 2 位应为 0。

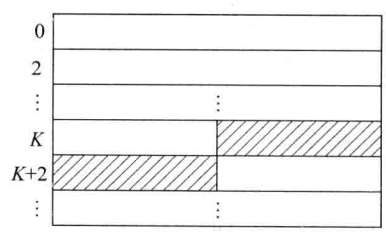


图 5.1 字边界不对准的情况

25. 加法指令和逻辑加指令的区别何在?

答:加法指令执行算术运算,并根据运算结果置状态位;逻辑加指令执行逻辑运算,不改变状态位的状态。

假设计算机的状态位 N、Z、V、C 依次为 0001,执行一条算术指令后得运算结果为 10000000,不溢出,有进位,则其状态位应是 1001;如果执行的是逻辑指令,则状态位仍是 0001。

26. 某计算机指令长度在 1~4 字节内变化,CPU 与存储器之间数据传送宽度为 32 位,每次取出 1 字(32 位),请问如何知道该字包含多少条指令呢?

答:在设计指令系统时,一般在指令(1 字节指令)或指令的第一个字节(多字节指令)中安排 1 至 2 位来区分是几字节指令。当安排 1 位时,用于区分是 1 字节指令或多字节指

令，然后在指令的第 2 字节中再区分多字节指令的字节数；若安排 2 位，则有 4 种状态，可直接区分 1～4 字节。另外也可将指令长度隐含在操作码中。

27. 在下列有关计算机指令系统的描述中，选择出正确的描述。对于不正确的描述请解释之。

(1) 浮点运算指令对用于科学计算的计算机是很必要的，可以提高机器的运算速度。

(2) 不设置浮点运算指令的计算机，就不能用于科学计算。

(3) 处理大量输入输出数据的计算机，一定要设置十进制运算指令。

(4) 兼容机之间指令系统是相同的，但硬件的实现方法可以不同。

(5) 同一系列中的不同型号计算机，保持软件向上兼容的特点。

(6) 在计算机的指令系统中，真正必需的指令数是不多的，其余的指令都是为了提高机器速度和/或便于编程而引入的。

答：(1)、(4)、(5)和(6)是正确的，(2)和(3)不正确。对(2)的解释如下：不设置浮点运算的计算机可以预先编制好浮点运算的子程序(使用定点指令)，运算时调用它即可。对(3)的解释如下：有大量输入输出数据而处理又比较简单的计算机，设置十进制运算指令后就不必进行十进制与二进制数据之间的转换操作，可提高数据的处理速度；但是如果不设置十进制运算指令，也可通过程序来解决。

28. 假设某 RISC 机寄存器 R_0 的内容恒为 0，要求用加法指令和减法指令将寄存器 R_2 的内容清除($(R_2)=0$)，如何实现？

答：假设加法/减法指令为三地址指令，可用下面任意一条指令将 R_2 的内容清除：

(1) ADD R_0, R_0, R_2 ；$R_2 \leftarrow (R_0)+(R_0)$

(2) SUB R_2, R_2, R_2 ；$R_2 \leftarrow (R_2)-(R_2)$

(3) ADD $R_0, \#0, R_2$ ；#0 为立即数 0，$R_2 \leftarrow (R_0)+0$

29. 讨论 RISC 和 CISC 在指令系统方面的主要差别。

答：RISC 的指令系统具有下述一些特点(CISC 不具备)：

(1) 优先选取使用频率最高的一些简单指令，以及一些很有用但不复杂的指令。避免复杂指令。

(2) 指令长度固定，指令格式种类少，寻址方式种类少。指令之间各字段的划分比较一致，各字段的功能也比较规整。

(3) 只有取数/存数指令(load/store)访问存储器，其余指令的操作都在寄存器之间进行。

(4) CPU 中通用寄存器数量相当多。算术逻辑运算指令的操作数都在通用寄存器中存取。

30. 什么是指令系统兼容和指令系统向上兼容？并讨论兼容的优缺点。

答：两种计算机的指令系统完全相同，称之为指令系统兼容。

在研制系列计算机的新型号或高档产品时，指令系统可以有较大的扩充，但仍保留原有机种的全部指令，称之为指令系统向上兼容。

兼容机的软件可以互相交换使用，因此软件也是兼容的。指令系统向上兼容则原有软件不作修改就可在新型号或高档产品上继续运行，称之为软件向上兼容。其优点是可以保护用户在软件上的投资，缺点是指令系统越来越庞大，新机种要保留原有机种的全部指令，

即使是这些指令在新编写的程序中根本不会使用。

31. 已知某机字长 16 位，主存按字编址，其一地址指令格式如下：

0　　5	6　7	8　　15
操作码	R	D

其中 R 为变址寄存器地址，寄存器字长为 16 位。试问在下列寻址方式下能访问的最大主存区为多少机器字？

(1)D 为立即数；(2)D 为主存直接地址；(3)D 为主存间接地址；(4)D 为变址寻址下的位移量。

答：(1) D 为立即数，与主存储器容量无关。

(2) D 为 8 位，能访问的最大存储区为 2^8 字。

(3) D 为间接寻址，由于主存字长为 16 位，所以能访问的最大存储区为 2^{16} 字。

(4) 变址寻址的有效地址＝(R)＋D，一般能访问的最大存储区为 2^{16} 字。但也可以是 $2^{16}+2^8$ 字。

32. 在字边界不对准的情况下，一个字分开存放在 $K+1$ 和 $K+2$ 两个存储单元中(参见第 23 题图 5.1)，其中字的高字节在 $K+1$ 中，低字节在 $K+2$ 中，今要将该字取到寄存器中，并调整其位置(字的高字节在左，低字节在右)，请写出实现方法及所需的指令。

答：字长为 16 位(2B)。假设 $K+1$ 和 $K+2$ 中的数据已取到 R_1 和 R_2 中。实现的方法有多种。

(1) 方法 1 的步骤如下：

① 将 R_1 中的数据左移 8 位(右 8 位补以 0)。

② 将 R_2 中的数据逻辑右移 8 位(左 8 位补以 0)。

③ R_1 和 R_2 中的数据相加(或逻辑加、按位加)，结果在 R_1 中。

需要的指令：左移(逻辑或算术左移)、逻辑右移和加(或逻辑加、按位加)。

(2) 方法 2 的步骤如下：

① 将 R_1 中的数据与常数 0000000011111111 进行逻辑乘，送 R_1，R_1 的左 8 位为 0，右 8 位保留。

② 将 R_2 中的数据与常数 1111111100000000 进行逻辑乘，送 R_2，R_2 的右 8 位为 0，左 8 位保留。

③ R_1 和 R_2 中的数据相加(或逻辑加、按位加)，结果在 R_1 中。

④ R_1 中的数据执行半字交换指令(左 8 位与右 8 位交换)，结果在 R_1 中。

需要的指令：逻辑乘、加(或逻辑加、按位加)和半字交换。

(3) 方法 3 的步骤如下：

① R_2 中的数据循环左移 1 位(C 参与移位)，将 R_2 中的最高位左移到 C。

② R_1 中的数据左移 1 位，将 C 移到 R_1 的最低位。

③ 重复①与②，共 7 次(包括第 1 次的操作则为 8 次)，结果在 R_1 中。

需要的指令：循环左移。

33. 设某机有加(＋)、减(－)两条算术运算指令，另外还有两条逻辑运算指令，如下所述，求这 3 种逻辑运算结果：

(1) 有逻辑加($\vee$)、逻辑乘($\wedge$)指令,求按位加($\oplus$)结果;

(2) 有逻辑加($\vee$)、按位加($\oplus$)指令,求逻辑乘($\wedge$)结果;

(3) 有逻辑乘($\wedge$)、按位加($\oplus$)指令,求逻辑加($\vee$)结果。

答:设有 A、B 两数。

(1) 按位加结果:

① $A\oplus B=(A\vee B)\wedge(\overline{A\wedge B})$

或

② $A\oplus B=(A\vee B)-(A\wedge B)$

例:设 $A=0101$,$B=1001$,此例已包括 A、B 的所有 4 种组合,从左到右,A、B 为 0、1、1、0、0、0、1、1。

验算如下:

① $A\vee B=0101\vee 1001=1101$,$A\wedge B=0101\wedge 1001=0001$,$\overline{A\wedge B}=1110$

$(A\vee B)\wedge(\overline{A\wedge B})=1101\wedge 1110=1100$,即为 $A\oplus B$ 的结果。

② $(A\vee B)-(A\wedge B)=1101-0001=1100$,也为 $A\oplus B$ 的结果。

(2) 逻辑乘的结果:

① $A\wedge B=(A\vee B)\oplus(A\oplus B)$

或

② $A\wedge B=(A\vee B)-(A\oplus B)$

同(1),验算如下:

① $(A\vee B)\oplus(A\oplus B)=1101\oplus 1100=0001$,即等于 $A\wedge B$。

② $(A\vee B)-(A\oplus B)=1101-1100=0001$,也等于 $A\wedge B$。

(3) 逻辑加的结果:

① $A\vee B=(A\wedge B)\oplus(A\oplus B)$

或

② $A\vee B=(A\wedge B)+(A\oplus B)$

同(1),验算如下:

$(A\wedge B)\oplus(A\oplus B)=0001\oplus 1100=1101$,即等于 $A\vee B$。

$(A\wedge B)+(A\oplus B)=0001+1100=1101$,也等于 $A\vee B$。

34. 请为一台具有简单指令的计算机编写定点数乘法运算的子程序,并讲述子程序中所用到的指令的功能。

答:设定定点数的字长为 n 位,其最高位为符号位,数值部分为 $n-1$ 位,数据以原码形式相乘。运算结果 $X\cdot Y$ 保存在通用寄存器 R_1、R_2。

运算器中有 n 位加法器和 4 个以上的通用寄存器,执行算术运算指令时,根据运算结果置状态位 Z 和 C。

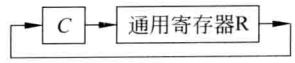
在乘法运算子程序中用到以下指令。

① LD:从存储器取数据到指定的通用寄存器中。

② ADD:对通用寄存器中的数进行加法运算,并置状态位 Z 和 C。

③ DEC:将通用寄存器的数值 -1,并置状态位 Z 和 C。

④ ROR:循环右移 1 位,进位状态位 C 参与循环右移如下:



如果连续执行两条 ROR 指令(ROR R_1 和 ROR R_2),可将第 1 条指令中指定的 R_1 的最低位移到第 2 条指令中指定的 R_2 的最高位,并将 R_2 的最低位移入 C。

⑤ ROL:循环左移 1 位。

⑥ BRC:条件转移指令,状态位 $C=0$ 转移,$C=1$ 顺序执行下一条指令。

⑦ BRZ:条件转移指令,状态位 $Z=0$ 转移,$Z=1$ 顺序执行下一条指令。

⑧ EOR:逻辑按位加指令(异或)。

⑨ RTS:从子程序返回。

乘法运算子程序如下:

```
START:  LD    R1, DATA        ;R1 清零,R1 存放部分乘积和乘积高位
        ADD   R1, R1          ;置状态位 C=0
        LD    R2, DATA+1      ;乘数送入 R2(Y→R2)
        LD    R3, DATA+2      ;被乘数取到 R3(X→R3)
        LD    R4, DATA+3      ;相加或移位次数 n-1→R4
LOOP:   ROR   R1              ;两条 ROR 指令,将 R1 和 R2 连接成双字
        ROR   R2              ;长数据,并将 R2 的最低位移入状态位 C
        BRC   +2              ;若 C=0,转移到 PC+2;若 C=1,顺序执行下一条指令
        ADD   R1, R3          ;R1←(R1)+(R3),置状态位 C
        DEC   R4              ;R4←(R4)-1,置 Z 和 C,且 C=0
        BRZ   -5              ;若 Z=0,转移到 PC-5,若 Z=1,顺序执行下一条指令
END:    EOR   R3, DATA+2      ;X⊕Y,最高位为乘积的符号(正或负)
        ROL   R3              ;乘积符号位→C
        ROR   R1              ;两条 ROR 指令,乘积高位在 R1,低位在 R2
        ROR   R2
        RTS                   ;乘法运算结束,从子程序返回原程序
DATA:   0                     ;常数 0
DATA+1:X                      ;被乘数
DATA+2:Y                      ;乘数
DATA+3:n-1                    ;考虑相加和移位的次数
```

第 6 章的第 24、30、31 和 32 题都与乘法运算的实现方法有关,可相互参考。

第 6 章　中央处理器

1. 中央处理器(CPU)的功能是什么?

答：计算机是由 CPU、存储器和输入输出系统组成的。CPU 负责程序的执行。程序是由指令按一定顺序编成的，所以 CPU 的责任是实现每条指令的功能，并及时从存储器取出接下来要执行的指令，包括需要由它自身完成的数据运算和数据处理工作，以及对存储器和输入输出设备发出控制信号，完成数据的存取和输入输出操作。

在 CPU 内部，可分成运算部件和控制部件两部分。运算部件主要完成数据的运算和处理；控制部件的作用是控制程序的执行，完成取指令、指令译码和发出执行指令时所需要的控制信号，还要进行故障处理和中断处理。

由于芯片集成度的提高，CPU 芯片内除了有运算部件和控制部件以外，还将存储器的一部分(例如 cache)集成到芯片中。

2. 简述控制器的基本组成。

答：控制器由以下 4 个部件组成。

(1) 程序计数器(PC)。PC 内存放的是当前正在执行的指令的地址或下一条将执行的指令地址，由于指令基本上是顺序执行的(除转移指令外)，由(PC)+1→PC 得到下一条要执行的指令的地址，所以称之为程序计数器。当执行转移指令时，由该指令指出下一条要执行的指令的地址，并将它送入 PC。

(2) 指令寄存器(IR)和指令译码器。IR 存放当前正在执行的指令，直到本条指令的完成。控制器根据 IR 的译码输出发出完成本条指令的全部命令。

(3) 脉冲源及启停线路。脉冲源即时钟脉冲，是机器周期和工作脉冲的基准信号。在机器刚加电时，还应产生一个总清信号(reset)，使计算机中的某些寄存器处于初始状态，例如将 PC 内置以启动机器后执行的第一条指令的地址等。

启停线路保证可靠地送出或封锁时钟脉冲，并保证时钟脉冲的完整(脉冲的幅度和宽度都符合要求)。

(4) 时序控制信号的形成部件。在时钟、机器周期和指令译码信号等的控制下，产生全机工作所需的控制信号，有硬连线(组合逻辑)和微程序两种控制方式。

3. CPU 结构如图 6.1 所示，其中有一个累加寄存器 AC、1 个状态寄存器和其他 4 个寄存器，各部分之间的连线表示数据通路，箭头表示信息传送方向。要求：

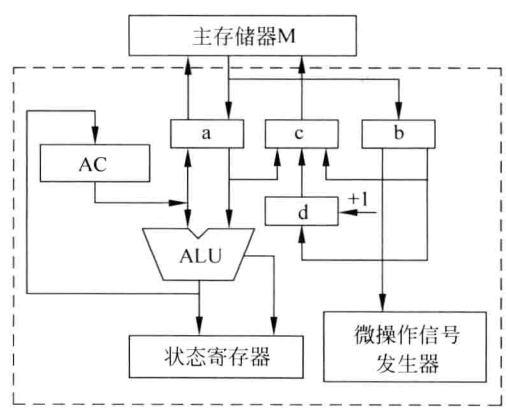


图 6.1　CPU 结构

(1) 标明图中 a、b、c、d 这 4 个寄存器的名称。

(2) 简述指令从主存取到控制器的数据

通路。

(3) 简述数据在运算器和主存之间进行存取/访问的数据通路。

答：(1) 已知 AC 为累加器，ALU 为算逻运算部件。其输入的 2 个数来源为(AC)和(a)。运算结果送入 AC。主存 M 与 CPU 之间的连线有地址线与数据线，其中地址线的信息仅从 CPU 传送到主存，数据则允许双方向传送，由此可见 a 与 c 分别是主存的数据寄存器 MDR 和地址寄存器 MAR。MDR 和 MAR 一般设置在主存中。在 CPU 中必须有程序计数器 PC 和指令寄存器 IR。PC 送地址到地址寄存器，并有自动加 1 的功能，因此 d 寄存器为 PC。IR 接收从主存来的指令，并控制微操作信号发生器，因此 b 寄存器是 IR。IR 向 PC 传送的是转移地址(当转移时，IR 的地址码字段存放的是转移地址)。

(2) 指令地址从 d(PC)送到 c(地址寄存器 MAR)，数据从主存送到 b(指令寄存器 IR)。

(3) 读写地址由 b(IR 的地址段)→c(MAR)，读时数据从主存→a(MDR)→ALU→AC，写时数据从 AC→a(MDR)→主存。

4. 如图 6.2 所示，在单总线结构中，有以下 4 类寄存器的传输操作，如果在同一类中的操作同时执行，存在什么问题？

(1) $R_1 \leftarrow R_2$，$R_1 \leftarrow R_3$；

(2) $R_1 \leftarrow R_2$，$R_2 \leftarrow 0$；

(3) $R_1 \leftarrow \overline{R}_1$；

(4) $R_1 \leftarrow PC$，$PC \leftarrow PC+1$。

答：假设寄存器由 CP 上升沿触发的触发器组成，输入与输出均有门控制。由于是总线结构，所以在同一时间，仅允许一个寄存器输出，输出采用三态门。门的控制信号与触发脉冲 CP 的时间关系如图 6.3 所示。

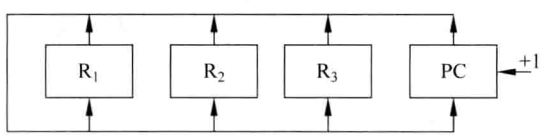


图 6.2　寄存器的传输操作

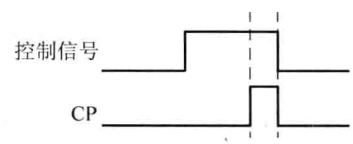


图 6.3　控制信号与脉冲波形图

(1) R_3 和 R_2 同时输出是不允许的，如果同时输出，则三态门输出不确定，且有可能损坏。

(2) $R_2 \leftarrow 0$，在图上未画出，可有两种方法实现，其一是将 0(低电位)送到总线上，此时也需要通过门输出，在这种情况下，同样不允许 0 和 R_2 同时送总线。其二是利用触发器置 0 端，将 R_2 置 0，置 0 的动作不需要 CP 作用。如果在 CP 上升沿来到之前已将 R_2 置 0，其结果是将 R_1 也置成 0；如果在 CP 上升沿来到之后，将 R_2 置 0，则将 R_2 中原来存放的内容传送至 R_1。

(3) 要实现 $\overline{R}_1$ 送到 R_1，需要从 $\overline{R}_1$ 端输出，是可实现的。

(4) 如果 PC 本身是一个计数器，即不需要与总线相连就能完成计数功能，那么送往总线的仅有一个输出，没有问题。如果完成计数的时间在 CP 上升沿之后，则将计数前的 PC 值送往 R_1。

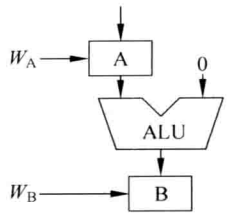
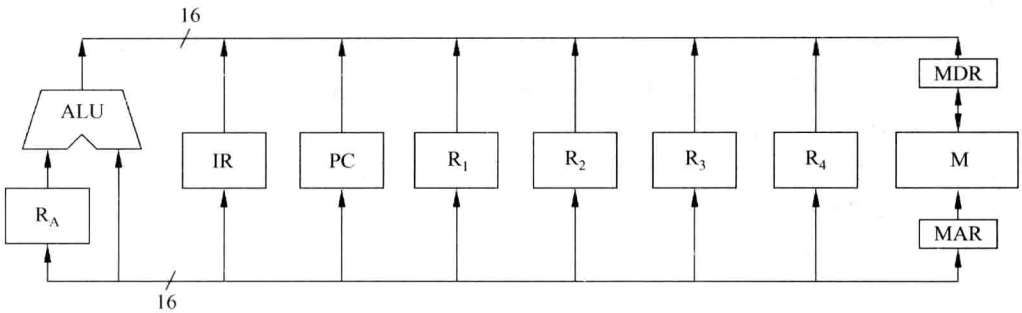


图 6.4　通过 ALU 接收数据

5. CPU 的部分数据通路如图 6.4 所示，W_A、W_B 是分别写入寄存器 A、B 的控制信号，如果要把寄存器 A 的内容写入 B，A 再接收新内容，问 W_A、W_B 能否包含在一条微指令中？如要将 W_A 和 W_B 包含在一条微指令中要采取什么措施？

答：如果寄存器 A、B 是采用电位触发方式的触发器，那么在 W_A、W_B 的控制下，A、B 的状态将随着各自的输入改变，最后 B 接收的是 A 的新内容（ALU 的输出是跟着输入变化的）。如果将 A、B 改成采用边沿触发方式的触发器，W_A 和 W_B 可包含在一条微指令中。

6. 某 CPU 数据通路为双总线结构，如图 6.5 所示。



图中　ALU——运算器　　R_A——ALU 的输入寄存器
IR——指令寄存器　　PC——程序计数器
R_1~R_4——通用寄存器　　MAR——存储器地址寄存器
MDR——存储器数据寄存器　　M——存储体

图 6.5　有错误连线的双总线结构

该图连线有误，请画出修正错误后的连线图（不改变原有的双总线结构），再写出实现加法指令 ADD addr，R_1 的过程，该指令实现（R_1）+（addr）→addr 功能，addr 为存储器某一单元的地址。

答：修改内容是：（1）改线，将 ALU 的输入输出线互换。

（2）添线，从 ALU 输出连一根线到 MDR 的输入。

（3）+1 信号送 ALU 的最低位进位输入端。

修改后的双总线结构见图 6.6。

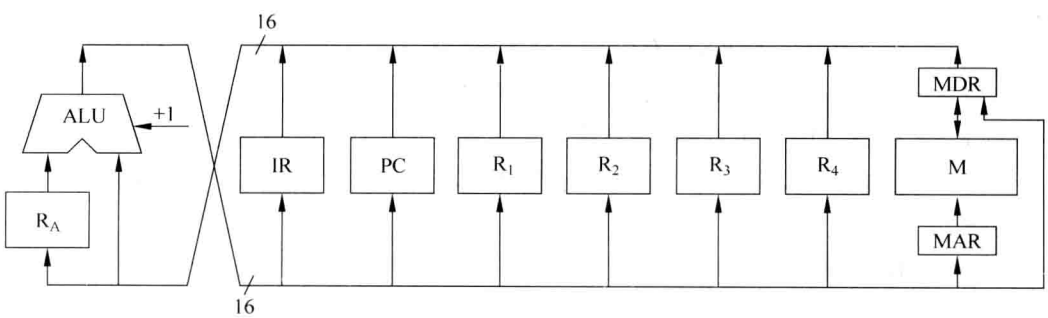


图 6.6　改正错误后的双总线结构

加法指令的执行过程如下(寄存器间传送都通过 ALU 进行)。

(1) 取指令：PC→ALU,ALU→MAR,读(读出数据到 MDR,该数据即为指令),(MDR)→ALU,ALU→IR,PC+1→PC(PC→ALU,+1→ALU,加)；

(2) 取操作数：IR 的地址码→MAR(IR 的地址码送入 ALU,ALU→MAR),读,(MDR)→R_A；

(3) 加：(R_1)→ALU,(R_A)→ALU,加,ALU→MDR(结果送入 MDR)；

(4) 写回：IR 的地址码→ALU,ALU→MAR,写。

以上操作中,凡是一个数据通过 ALU 传送到另一总线时,ALU 的另一输入为 0,ALU 执行加法运算。

7. 接上题,如果执行相对转移指令,写出其执行过程。

答：(1) 取指令(同上题)。

(2) 计算转移地址。由于取指令后 PC 已加 1,所以指令中给出的偏移值应等于实际值减 1,例如,要转移到本条指令后面的第 5 条指令,指令中的偏移值应等于 4。

执行过程如下：

① PC→R_A

② IR 地址码(偏移量)→ALU,R_A→ALU,加,ALU→PC

8. 写出转子程序指令的执行过程以及实现子程序需要考虑的问题。

答：执行过程如下：

(1) 取指令到 IR,PC+1→PC；

(2) 指令译码；

(3) 执行操作。

如果为直接寻址,则 IR 的地址码字段设置的就是被调用的子程序的入口地址。在这种情况下,先将 PC(顺序执行的指令地址)送入主存储器的固定单元(设计机器时就已确定的)或者送入堆栈,这个操作称为保留返回地址,然后将 IR 的地址码送入 PC。本条指令执行完毕。

当取出下一条执行的指令时,即进入子程序。需考虑问题如(4)、(5)所述。

(4) 进入子程序后有些操作是需要共同执行的(即无论是何种子程序都需要执行的),现将要点叙述如下：

① 将返回地址从固定单元保留到本子程序设定的位置,因为在执行子程序时可能会再次执行一条转子程序(嵌套)指令或者要进行中断处理,如果不将返回地址预先从固定单元移走,此时即会丢失。

② 将有关通用寄存器等的内容保留起来,避免执行子程序用到这些寄存器时破坏了其原有的内容,从而在返回调用程序后造成错误。

因此在真正进入子程序的具体操作之前需要进行保留返回地址、保留寄存器等操作(称为保留现场)。这些操作可以用程序来实现,也可以用硬件来实现,若用硬件来实现,由于执行时间很长,RISC 计算机是不允许的。保留的数据可保留在子程序设定的存储区,也可以保留在堆栈中。

(5) 从子程序返回原调用程序时要执行一条返回指令,恢复 PC 和寄存器内容,后者可以用软件(程序)实现或硬件(返回指令)实现。

9. 假设某计算机执行一条指令需要若干个机器周期，每个机器周期只有一个节拍（一个时钟脉冲），在这种情况下到存储器去取指令到 IR 需要几个节拍？如何改进？

答：完成取指操作需要相继执行两个操作，PC→MAR；读，MDR→IR。因此需要 2 个节拍（此处假设存储器的读取时间为一个节拍，在第 2 节拍的周期开始后送读命令，在时钟脉冲到来之前，数据自动送到 MDR，时钟脉冲将 MDR 的内容送入 IR）。

计算机通常在上一条指令结束时执行 PC→MAR 操作，这样可以节省一个节拍时间。

10. 某计算机指令长度固定，为二地址指令，请分析下列 3 种类型指令：

（1）算术逻辑运算指令。有加、减、逻辑加（∨）和逻辑乘（∧）4 条指令，寄存器寻址、寄存器间接寻址和存储器直接寻址 3 种寻址方式。例如，加法指令表示为

$$\text{ADD } R_1, R_2; \quad \text{ADD } (R_1), (R_2); \quad \text{ADD addr}_1, \text{addr}_2。$$

（2）转移指令。有无条件转移、条件转移和转子程序 3 条指令，例如，无条件转移指令表示为 JMP addr，一种寻址方式，指令中地址字段给出转移地址。

（3）堆栈指令。有进栈 PUSH 和退栈 POP 两条指令。堆栈建在主存中，堆栈指针为寄存器，本身有＋1 与－1 功能。进出堆栈的数据在 PC 或寄存器 R_1 中。

要求画出指令执行的流程图，并指出每条指令的周期数。

答：指令执行流程图如图 6.7 所示。

本流程图假设 PC 本身有＋1 功能；算术逻辑运算的结果保存在第 1 地址所指示的地址中，R_1、R_2 为寄存器地址，addr_1、addr_2 为存储器地址；ALU 的输入端各有一个寄存器 A、B。又假设本条指令执行前，PC 值已在 MAR 中，所以在本条指令结束前，需将下一条要执行的指令地址送到 MAR 中。转移指令和堆栈指令只需一个地址。

条件转移的条件码由前面执行的算术运算设置。转子程序指令将返回地址（断点）压入堆栈。实际上具有转子程序指令的计算机还应该有返回指令，在本题中不进行分析。

堆栈指令将 PC 或 R_1 的内容压入或弹出堆栈。

执行一条机器指令最多需要 7 个机器周期，列在同一周期内的操作是可以并行操作的，每条指令从 T_1 开始，T_2 指令译码所需的时间比较短，但为了简化设计，保持与其他机器周期一致。

算术逻辑运算指令（简称算逻指令）中寄存器寻址方式需 5 个机器周期（T_1，T_2，T_3，T_4，T_6），其他寻址方式为 7 个机器周期。转移指令中条件转移与无条件转移需 3 个机器周期（$T_1 \sim T_3$）。转子程序为 4 个周期（$T_1 \sim T_4$），堆栈指令为 4 个机器周期（$T_1 \sim T_4$）。

11. 接上题，完成下述任务：

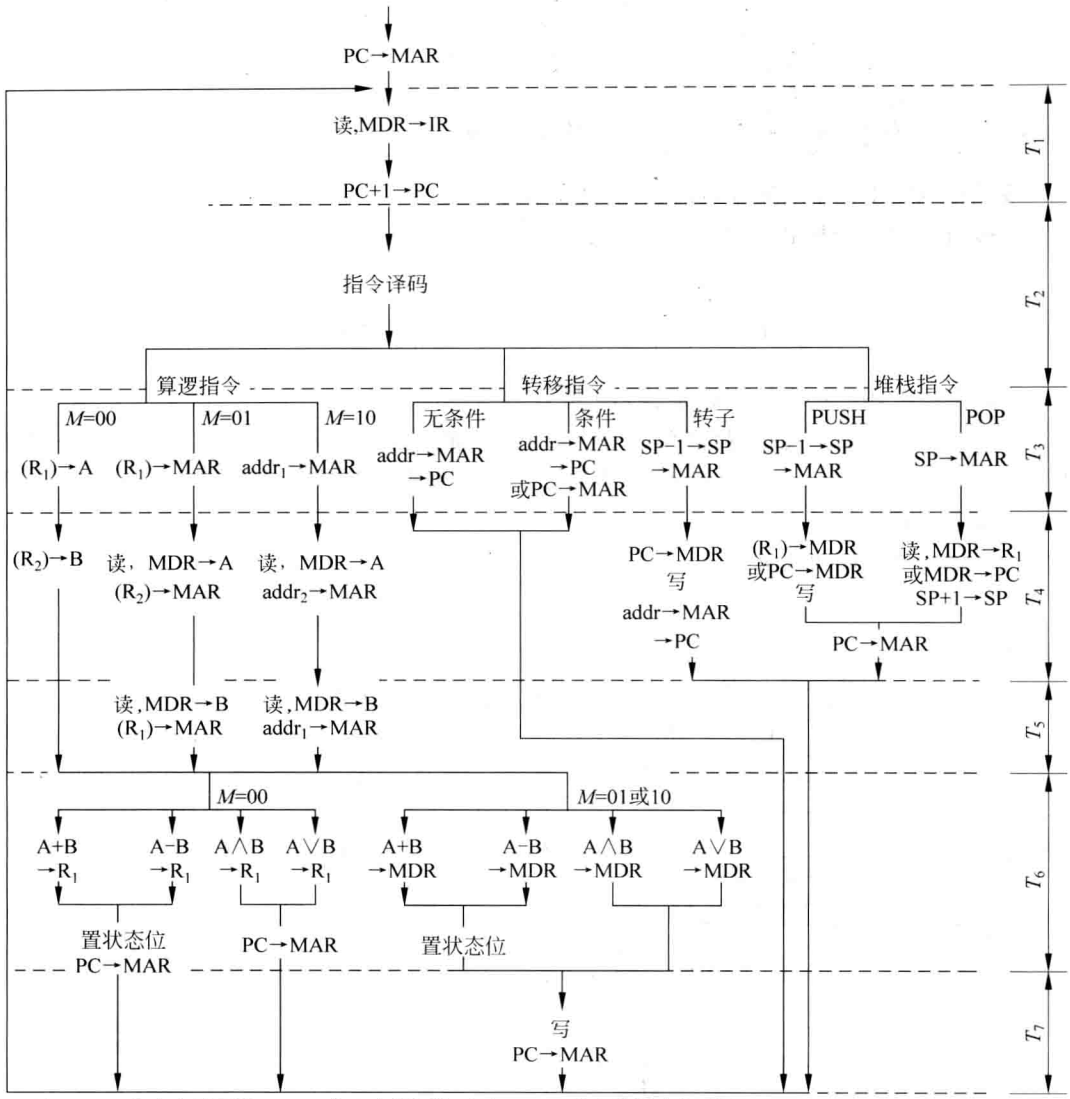
（1）设计指令格式。假设指令字长不超过 32 位，存储器容量为 4MB，按字编址。通用寄存器有 32 个，字长 32 位。

（2）本题给出的条件有何缺点？请提出一种改进的方法。

（3）根据第 10 题图 6.7 的流程图画出逻辑框图。

答：（1）指令格式。操作码 OP 为 4 位，有第 10 题指出有 9 条指令，还可扩充 7 条指令。9 条指令的操作码（$OP_3 \sim OP_0$）设定如下：0000 加法，0001 减法，0010 逻辑加，0011 逻辑乘，0100 无条件转移，0101 条件转移，0110 转子程序，1000 进栈操作，1001 退栈操作。

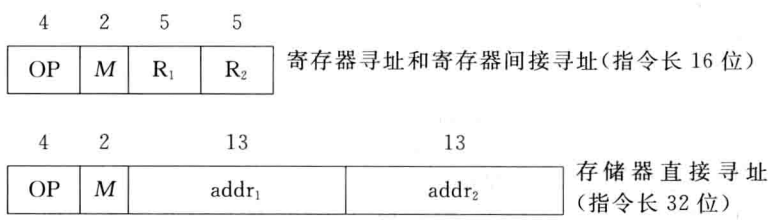
地址码。总共可有 28 位。内存容量为 4M 字，所需地址为 22 位。寄存器地址码为 5 位。



注：M=00为寄存器寻址，M=01为寄存器间接寻址，M=10为存储器直接寻址

图 6.7　指令执行流程图

算术逻辑运算指令的指令格式如下。



$M(M_1,M_0)$为寻址方式位，$M=00$ 为寄存器寻址，$M=01$ 为寄存器间接寻址，$M=10$ 为存储器直接寻址。

转移指令的指令格式如下。

4	6	22	
OP	空	转移地址 addr	无条件转移和转子指令
OP	转移条件	转移地址 addr	条件转移

转子指令将 PC 存入堆栈，堆栈设在主存中。

堆栈指令的指令格式如下。

4	2	5	5	
OP	C	R_1	空	堆栈指令

C 为控制位，01 是 PC 进栈/退栈，10 是 R_1 进栈/退栈。

(2) 缺点。存储器直接寻址字段只有 13 位，不能对整个存储器寻址，如果增加一种变址寻址方式就可对整个存储器寻址(原来的方案可用寄存器间接寻址对整个存储器寻址，但有时使用不方便)。

(3) 逻辑框图见图 6.8。

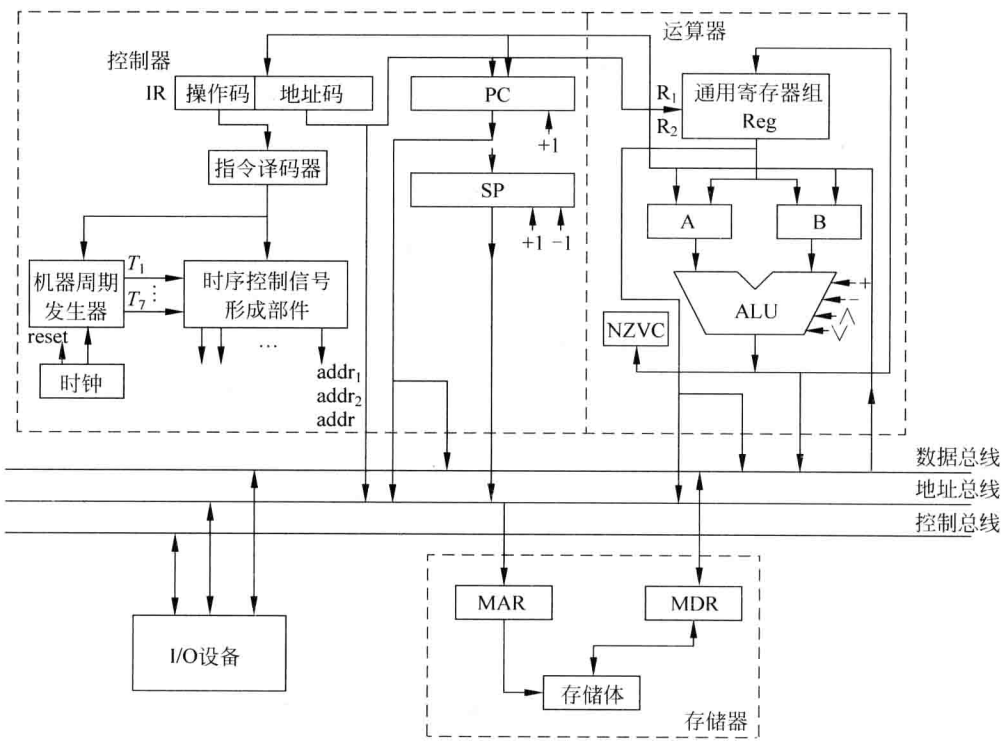


图 6.8 实现图 6.7 流程的逻辑框图

本图分 3 部分：运算器、控制器和存储器。

运算器由通用寄存器组 Reg、ALU、寄存器 A、寄存器 B 和状态寄存器 NZVC 组成，条件转移指令根据此状态决定是否转移。寄存器 A、B 的输出直接送入 ALU(不受其他信号控制)。

控制器由程序计数器 PC、指令寄存器 IR、指令译码器、堆栈指针 SP、时钟、机器周期发生器和时序控制信号形成部件构成；存储器由地址寄存器 MAR、数据寄存器 MDR 和存储器芯片(存储体)组成。

机器启动时，由时钟部件发出总清信号 reset，将 PC 和 MAR 置以启动后第 1 条要执行的指令地址，并根据需要将某些触发器或寄存器置于初始状态，SP 的内容置为栈底地址，机器周期发生器处于 T_1 状态。启动后从 T_1 到 T_7 顺序产生信号，任意时间只能有一个处于工作状态。为节省时间，应跳过无操作的周期，例如，执行转移指令时，从 T_3 直接跳到下一条指令的周期 T_1，执行寄存器寻址的算术逻辑运算指令跳过 T_5 和 T_7。

流程图上每一个操作都受时序控制信号形成部件发出的控制信号控制，每一个输入端都有一个受控制的门，当控制信号来时(为 1 时)才能将门打开，输入数据或地址。例如，图 6.7 流程图中的(R_1)→A(从地址为 R_1 的通用寄存器中取数到 A 寄存器)需要 2 个控制命令：①将地址 R_1 送入通用寄存器；②R_1 的内容送入 A 寄存器。而 PC+1 只需要一个控制信号。

图 6.8 逻辑框图中的运算器和控制器组成 CPU，从 CPU 送到地址总线有 4 组地址，CPU 和存储器之间是通过总线连接的。还有 I/O 设备也连到总线上。CPU 连接到总线的输出电路应是三态电路，不输出时处于高阻抗状态。

12. 结合第 10 题的流程图(图 6.7)和第 11 题的逻辑框图(图 6.8)回答下列问题：

(1) 列出各条指令所需要的控制信号。

(2) 如果采用硬连线控制方案，怎样形成控制信号？(任意举出两个以上控制信号的逻辑表达式。)

(3) 如果采用微程序控制方案(水平型微指令)，怎样形成控制信号。

答：(1) 列出算术逻辑运算指令的控制信号：

PC→MAR，读，写，MDR→IR，PC+1→PC，R_1→Reg(地址)，R_2→Reg(地址)，Reg(内容)→A，Reg(内容)→B，Reg(内容)→MAR，$addr_1$→MAR，$addr_2$→MAR，MDR→A，MDR→B，+，−，∨，∧，置 NZVC，ALU→R_1，ALU→MDR。

流程图上(R_1)→A 是将 R_1 作为通用寄存器的地址，从其中取出数据送 A 寄存器，实现时需要两个信号 R_1→Reg(地址)，Reg(内容)→A。读、写信号是送往存储器的。

列出转移指令的控制信号：

PC→MAR，读，写，MDR→IR，PC+1→PC，addr→MAR，addr→PC，SP−1→SP，SP−1→MAR，PC→MAR。

列出堆栈指令的控制信号：

PC→MAR，读，写，MDR→IR，PC+1→PC，SP−1→SP，SP−1→MAR，SP→MAR，R_1→Reg(地址)，Reg(内容)→MDR，PC→MDR，MDR→R_1，MDR→PC，SP+1→SP。

(2) 硬连线控制方案

由操作码、寻址方式及周期 T_1～T_7 构成控制信号，列出上述每一个控制信号的逻辑表达式，然后进行化简。

举例：

① PC→MAR

在算术逻辑运算指令(寄存器寻址)的 T_6 以及算术逻辑运算指令(寄存器间接寻址或存储器直接寻址)的 T_7 产生 PC→MAR 控制信号。

在条件转移指令的 T_3(转移条件不满足时)，产生 PC→MAR 控制信号。在下面条件转移满足用 J 表示，不满足用 $\bar{J}$ 表示。

在堆栈指令的 T_4 产生 PC→MAR 信号，所以：

$$
\begin{aligned}
\text{PC}\rightarrow\text{MAR}=&\text{算术逻辑运算指令}\times\text{寄存器寻址}\times T_6\\
&+\text{算术逻辑运算指令(寄存器间接寻址}+\text{存储器直接寻址)}\times T_7\\
&+\text{条件转移指令}\times\text{转移条件不满足}\times T_3+\text{堆栈指令}\times T_4\\
=&\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\overline{M_1}\cdot\overline{M_0}\cdot T_6+\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot(M_0+M_1)\cdot T_7\\
&+\overline{\text{OP}_3}\cdot\text{OP}_2\cdot\overline{\text{OP}_1}\cdot\text{OP}_0\cdot\bar{\text{J}}\cdot T_3+\text{OP}_3\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot T_4
\end{aligned}
$$

式中 $\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}$ 是算术逻辑运算指令操作码，是由 $\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot\overline{\text{OP}_0}+\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot\text{OP}_0+\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\text{OP}_1\cdot\overline{\text{OP}_0}+\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\text{OP}_1\cdot\text{OP}_0=\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}$ 化简而来的。

$\text{OP}_3\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}$ 为堆栈指令操作码，是由 $\text{OP}_3\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot\overline{\text{OP}_0}+\text{OP}_3\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot\text{OP}_0=\text{OP}_3\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}$ 化简而来的。

② $\text{R}_1\rightarrow$Reg(地址)

$$
\begin{aligned}
\text{R}_1\rightarrow\text{Reg(地址)}=&\text{算逻指令}((\overline{M_1}\cdot\overline{M_0}+\overline{M_1}\cdot M_0)T_3+\overline{M_1}\cdot M_0\cdot T_5\\
&+\overline{M_1}\cdot\overline{M_0}\cdot T_6)+\text{堆栈指令 }T_4\cdot(C=01)\\
=&\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}((\overline{M_1}\cdot\overline{M_0}+\overline{M_1}\cdot M_0)\cdot T_3+\overline{M_1}\cdot M_0\cdot T_5+\overline{M_1}\cdot\overline{M_0}\cdot T_6)\\
&+\text{OP}_3\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot T_4
\end{aligned}
$$

式中 C 是堆栈指令中的 2 位控制位，$C=01$ 可消去，因为执行堆栈指令时即使 $C\neq01$，产生 $\text{R}_1\rightarrow$Reg(地址)信号，也不会出错。C 的定义见第 11 题堆栈指令的指令格式。

③ Reg(内容)→A＝算逻指令$\times\overline{M_1}\cdot\overline{M_0}\cdot T_3=\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\overline{M_1}\cdot\overline{M_0}\cdot T_3$

④ 完成 A＋B→R_1，A＋B→MDR 操作。

流程图上的 A＋B→R_1 操作实际上需要 3 个控制信号：＋，$\text{R}_1\rightarrow$Reg(地址)，ALU→R_1。A＋B→MDR 操作需要 2 个控制信号：＋，ALU→MDR。下面讨论＋、ALU→R_1 和 ALU→MDR 的逻辑表达式($\text{R}_1\rightarrow$Reg(地址)已在前面讨论过)。

$$+=\text{加法指令}\times T_6=\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot\overline{\text{OP}_0}\cdot T_6$$

$$\text{ALU}\rightarrow\text{R}_1=\text{算逻指令}\times\overline{M_1}\cdot\overline{M_0}\cdot T_6=\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\overline{M_1}\cdot\overline{M_0}\cdot T_6$$

$$\text{ALU}\rightarrow\text{MDR}=\text{算逻指令}\times(M_1+M_2)T_6=\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot(M_1+M_2)\cdot T_6$$

其他控制信号的逻辑表达式不再详细介绍。

上述表达式虽已经过初步简化，但仍可进一步简化。举例如下：

① 某些控制信号的组成部分多处用到，可先进行组合，例如 $\overline{\text{OP}_3}\cdot\overline{\text{OP}_2}\cdot\overline{\text{OP}_1}\cdot\overline{\text{OP}_0}$ 可先用一个“与门”(逻辑乘)实现之，当其他公式中遇到它时，利用该“与门”的输出即可。

② 在 $\text{R}_1\rightarrow$Reg(地址)的公式中，当执行算逻指令到 T_3 时，即使在 $M=10$ 的条件下产

生 $R_1 \rightarrow Reg$(地址)控制信号，也不会出错。该信号在 T_5 与 T_6 时的情况类似，于是可将 $R_1 \rightarrow Reg$(地址)简化如下：

$$R_1 \rightarrow Reg(\text{地址}) = \overline{OP_3} \cdot \overline{OP_2}(T_3 + T_5 + T_6) + OP_3 \cdot \overline{OP_2} \cdot \overline{OP_1} \cdot T_4$$

注：算逻指令($\overline{M_1} \cdot \overline{M_0}$)不存在 T_5。

(3) 微程序控制方案

完全水平型微指令每个控制信号都要在控制字段设置一个控制位，从(1)得出执行算术逻辑运算指令的控制信号为 21 个，执行转移指令的控制信号再加上 5 个，执行堆栈指令的控制信号再增加 5 个，因此总共有 31 个控制信号。当需要产生某个控制信号时，就将该位置 1，否则置 0，一个机器周期内的操作是可以同时执行的，控制这些操作的控制位应为 1。将互斥的信号分在一个组内，采用编码方式可以减少控制位，例如，控制 ALU 操作的＋、－、∧、∨为互斥信号，为了保留一个不操作状态，所以该组需要 3 个控制位。

流程图上每个机器周期内完成的操作就是一条微指令完成的操作，假如采用断定方法来确定下址，那么下址就在每条微指令的下址字段中。当一条指令取到 IR 中以后，就由译码器给出下条微指令在控制存储器中的位置。如果遇到微程序有分支，就由指定条件实现多路转移。

13. 在以下的空格中填上有关微程序的内容。

(1) 完成各条指令操作的一系列微指令的集合称为[A]。

(2) 微指令分两个字段，它们是[B]和[C]。微指令的基本格式有[D]和[E]两种。

(3) 如果微指令的操作控制字段为 60 位，采用水平型格式，而且所有微操作都不是互斥的，则原则上允许同时启动[F]个微操作，实际上是不可能的。

对于垂直型微指令，如果微操作码为 6 位，则可定义[G]种微操作，通常一条微指令启动[H]个微操作。

(4) 由此得出结论，垂直型微指令字的长度比水平型微指令[I]，但其无并行操作能力，所以执行一条指令的时间比水平型微指令[J]。

答：A——微程序；B——控制字段；C——下址字段；D——水平型；E——垂直型；F——60；G——2^6；H——1；I——短；J——长

14. 从供选择的答案中，选出正确答案填入[ ]中。

(1) 在实现微程序时，取下一条微指令和执行本条微指令一般是[A]进行的，而微指令之间是[B]执行的。

(2) 实现机器指令的微程序一般是存放在[C]中的，而用户可写的控制存储器则由[D]组成。

供选择的答案：

A、B：①顺序，②重叠。

C、D：①随机存储器 RAM，②只读存储器 ROM。

答：A——②；B——①；C——②；D——①

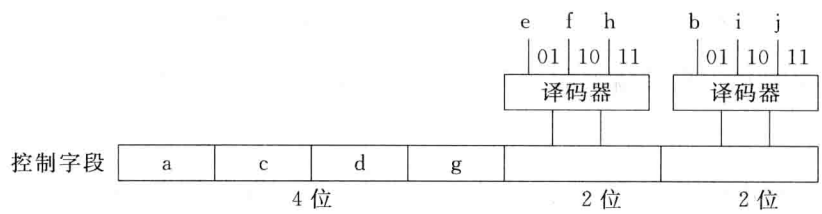
15. 某机有 8 条微指令 I1～I8，每条微指令所包含的微命令控制信号如下表所示。

微指令	微命令控制信号									
	a	b	c	d	e	f	g	h	i	j
I1	√	√	√	√	√					
I2	√			√		√	√			
I3		√						√		
I4			√							
I5			√		√		√		√	
I6	√							√		√
I7			√	√				√		
I8	√	√						√		

a～j 分别对应 10 种不同性质的微命令信号，假设一条微指令的控制字段为 8 位，请安排微指令的控制字段格式。

答：由于微命令有 10 个，而控制字段只有 8 位，可以将互斥的命令编成一组，成为微指令字的一个字段，用二进制编码来表示。实现的方案可以有多种。下面举出两种方案。

方案 1：



a、c、d、g 为直接控制字段。

e、f、h 编成一组，b、i、j 编成一组。当编码为 00 时，为不操作。

方案 2：

a、c、d、i 为直接控制字段。

b、g、j 编成一组，e、f、h 编成一组。

16. 某机的微指令格式中有 10 个独立的控制字段 C_0～C_9，每个字段可激活 N_i 条控制线中的某一条，且是互斥的。设各字段的 N_i 数量如下：

字段 C	0	1	2	3	4	5	6	7	8	9
N_i	4	6	3	11	9	5	7	1	8	15

问：

(1) 为表示这 10 个控制字段，需要多少控制位(采用编码表示法)？

(2) 如果采用完全水平型编码格式，需要多少控制位？

答：下述公式中的每一项按字段顺序给出。

(1) 控制位数＝3＋3＋2＋4＋4＋3＋3＋1＋4＋4＝31 位(按字段译码)

(2) 控制位数＝4＋6＋3＋11＋9＋5＋7＋1＋8＋15＝69 位(完全水平型)

17. 在微程序控制器中，微程序计数器 μPC 可用 μMAR 来代替，试问 PC 是否可用具有计数功能的存储器地址寄存器 MAR 来代替？

答：不能。因为存储器中除了存放程序外还存放数据，遇到取数据的指令时其地址是任意的(与 PC 值无关)，此时如果用有计数功能的 MAR 来取代 PC，将使 PC 值丢失。而控制存储器中存放的是微程序，其 μMAR 不会用做数据的地址寄存器，所以可以取代 μPC。

18. 指令译码器采用何种方法实现？哪种方法的速度相对慢些？取指令操作受操作码控制吗？

答：在硬连线控制器中，译码器用逻辑电路(门电路)实现；在微程序控制器中，译码器可用只读存储器实现，其输入地址为指令操作码，其输出(即 ROM 的内容)是相应指令的微程序入口地址，每条指令的功能是通过执行一段微程序而实现的。注意译码器的 ROM 和控制存储器的 ROM 是两个相互独立的 ROM。

无论在硬连线控制器或微程序控制器中，取指令周期内的操作是不受指令操作码控制的，称之为公共操作。在指令取到指令寄存器 IR 以后，操作码才起作用。在微程序控制器中，取指令操作使用微子程序或一条微指令实现，然后根据操作码转移到各条指令的微程序入口。图 6.9 为微程序执行流程的举例，图中每一方框表示一条微指令，方框上方表示的是该条微指令的地址，方框内为执行的操作，在其右下角为下一条要执行的微指令的地址，表示在微指令的下址字段中。取指微指令的操作对所有的指令都是相同的。所以是一条公用的微指令，其下址由操作码译码产生。

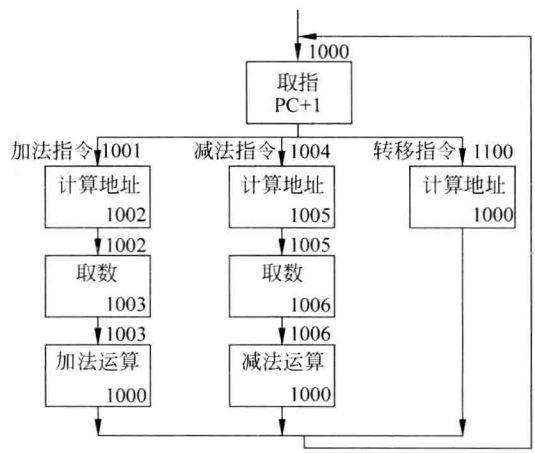


图 6.9　微程序执行流程图

由于 ROM 的操作比逻辑电路慢，所以采用 ROM 作为译码器时，其速度慢些。

19. 某机采用微程序控制方式，其控制存储器容量为 1024×48 位。微程序可在整个控制存储器中实现转移，可控制微程序转移的条件共有 4 个(直接控制)，微程序采用水平型格式，如下所示：

微指令	操作控制字段	判别测试字段	下地址字段

(1) 微指令中 3 个字段分别为多少位？

(2) 画出围绕这种微指令格式的微程序控制器逻辑框图。

答：(1) 控制存储器容量为 1024 字，所以下地址字段为 10 位；判别测试字段控制微程序转移，为 4 位；还有 48－10－4＝34 位为操作控制字段长度。

(2) 图 6.10 为微程序控制器逻辑框图。图中译码器可用 ROM 构成，μPC 同时也为 μMAR。μPC 输入有 4 种情况，顺序执行时 μPC+1，取指令后从译码器输入，符合转移条件时从下址字段输入，执行微转子程序指令返回时从微返回寄存器 μRR 输入(微程序中也可设计微子程序)。

微子程序的转移情况与子程序相同，其微返回地址一般保留在寄存器中(图中的 μRR)或寄存器堆中。

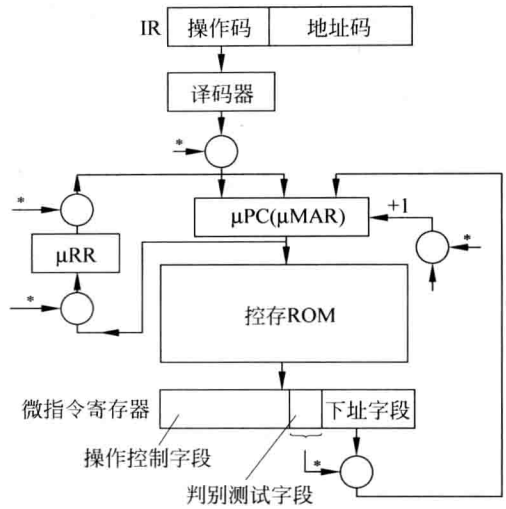


图 6.10 微程序控制器逻辑框图

注：图中圆圈为与门，带“*”号的为判别测试字段来的控制信号。

20. 某机采用微程序控制方式，微指令字长 24 位，采用水平型编码控制的微指令格式，共有微命令 29 个，构成 4 个相斥类，各包含 7 个、8 个、14 个和 2 个微命令。控制下址的外部条件共 3 个。

(1) 控制存储器容量最多为多少字？

(2) 设计出微指令的具体格式。

答：(1) 4 个相斥类的控制位分别为 3 位、4 位、4 位和 2 位，共 13 位；控制下址的外部条件为 2 位，其中 00 表示不转移。余下 9 位作为下址，所以控制存储器容量最多为$2^9=512$ 字。

(2) 微指令格式如图 6.11 所示。

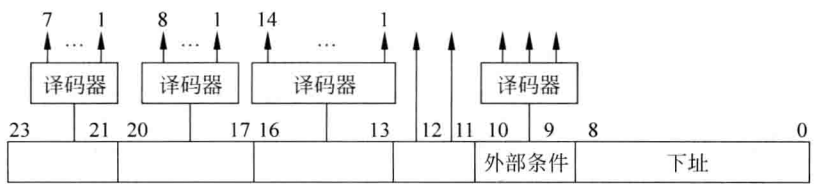


图 6.11 微指令格式

每个相斥类上接一个译码器，当相应的代码为 0…0 时，不进行译码，不控制操作(无操作)，产生 2 个微命令的相斥类设计成直接控制操作。

21. 某机采用微程序控制方式，已知每一条机器指令的执行过程(不包括取指令的微指令)均可分解为 8 条微指令组成的微程序，该机指令系统采用 6 位定长操作码格式，问：

(1) 控制存储器至少应能容纳多少条微指令？

(2) 如何确定指令操作码与该指令微程序的起始地址的对应关系？

(3) 如果每条指令执行所需的微指令数各不相等，如何确定每条指令的微程序起始地址？

答：(1) 控制存储器至少能容纳 $2^6\times8+1=513$ 条微指令，其中一条是取指令的微指令，是一条公用的微指令。实际上还要考虑微中断等，所需的控存容量还要更大些。

(2) 执行每条指令均为 8 条微指令时，可将指令的操作码直接送到 μPC 的高 6 位，低 3 位置 0(μPC 共 9 位)，这就是每条指令微程序的起始地址(其他微指令另作处理)。

(3) 如果每条指令执行的微指令数各不相同，则在指令操作码输出接上一个由 ROM 构成的译码器(指令操作码作为 ROM 的地址)，然后顺序将执行每一条指令所需的微指令数累加，作为下一个操作码所对应的微程序的起始地址。例如，假设 00_8 操作码的指令执行时需 8 条微指令，01 操作码的指令执行时需 10 条微指令，02 操作码的指令执行时需 13 条微指令，……。假设从控存的 000_8 位置开始存放微程序，则 00、01、02、03、…操作码所对应的微程序起始地址为 000_8、010_8、022_8、037_8、…，将其存入 ROM(译码器)的 0 号、1 号、2 号、3 号、…地址中即可，这样每当取一条指令到指令寄存器中，就可从 ROM 的输出得到相应的微程序起始地址。

22. 设某计算机运算控制器逻辑图和控制信号如教材中的图 6.6 和表 6.1 所示(见本章附录)。

已知指令格式和微指令格式如下所示。

其中 1～23 位代表的 1～23 号控制信号见教材中的表 6.1。

试写出下述 3 条指令的微程序编码(下址字段略)：

(1) JMP(无条件转移到(rsl)＋disp)

(2) Load(从(rsl)＋disp 指示的内存单元取数，送 rd 保存)

(3) Store(把 rs 内容送到(rsl)＋disp 指示的内存单元)

提示：先列出各指令执行步骤和每步所需控制信号，最后再写出编码。

答：每条指令都要执行取指令到指令寄存器的微指令，这是一条公用的微指令。所需的控制信号如下：

- PC→AB(该信号由上一条指令结束前发出)。
- 指令→IR_0：ADS^{21}，$M/\overline{IO}^{22}$，读$(W/\overline{R}=0)^{23}$，主存自动将读出数据送数据总线 DB，$DB \rightarrow IR^5$。以上信号控制完成取指令操作。控制信号右上角的编号即为其在教材表 6.1 中的序号。
- $PC+1^3$：为顺序执行的下一条指令地址作准备。

所以第一条取指微指令的编码如下：

<1> 0010,1000,0000,0000,0000,110

然后由指令译码器(ROM)根据操作码产生下一条微指令的地址。

下面讨论执行各条指令的微程序。

(1) JMP 指令所需控制信号如下。

- 计算转移地址((rsl)＋disp)送入 AR：
  $rsl \rightarrow GR^8$，$(rsl) \rightarrow ALU^{10}$，$disp \rightarrow ALU^4$，$+^{13}$，$ALU \rightarrow AR^{19}$
- 下一条指令地址送入 PC：
  $ALU \rightarrow PC^2$

所以 JMP 的微指令为

<2> 0101,0001,0100,1000,0010,0××

(2) LOAD 指令。

- 计算主存地址((rsl)+disp)并送入 AR：
  rsl→GR^{8},(rsl)→ALU^{10},disp→ALU^{4},$+^{13}$,ALU→AR^{19}

所以第 2 条微指令为

<2> 0001,0001,0100,1000,0010,0××

- 从主存取数：AR→AB^{20}(地址送地址总线),ADS^{21},$M/\overline{IO}^{22}$读($W/\overline{R}=0$),DB→DR^{6}(读出数据送数据寄存器)。

所以第 3 条微指令为

<3> 0000,0100,0000,0000,0001,110

- 存储器取出的数据经过 ALU 送 r_d：
  DR→ALU^{12},r_d→GR(地址)^{9},$+^{13}$,ALU→GR^{17}(数据)
- 下一条指令地址送地址总线：
  PC→AB^{1}

所以第 4 条微指令为

<4> 1000,0000,1001,1000,1000,0××

(3) store 指令。

- 计算主存地址,同 Load 指令。

所以第 2 条微指令为

<2> 0001,0001,0100,1000,0010,0××

- 从 GR 取数到 DR：
  rs→GR^{9}(地址),(rs)→ALU^{11},$+^{13}$(ALU 输入端仅有 1 个数据,另一数据为 0),ALU→DR^{18}

所以第 3 条微指令为：

<3> 0000,0000,1010,1000,0100,0××

- 将数据存入主存：
  AR→AB^{20},DR→DB^{7},ADS^{21},$M/\overline{IO}^{22}$,$W/\overline{R}^{23}$
- PC→AB^{1}

所以第 4 条微指令为

<4> 1000,0010,0000,0000,0001,111

注：×表示可为任意值,当 ADS=0 时,微指令最后两位不起作用。

现将 3 条指令的微程序编码总结如下(未列出下址部分)：

取指 <1> 0010, 1000, 0000, 0000, 0000, 110
JMP <2> 0101, 0001, 0100, 1000, 0010, 0××
Load <2> 0001, 0001, 0100, 1000, 0010, 0××

<3> 0000，0100，0000，0000，0001，110

<4> 1000，0000，1001，1000，1000，0××

Store <2> 0001，0001，0100，1000，0010，0××

<3> 0000，0000，1010，1000，0100，0××

<4> 1000，0010，0000，0000，0001，111

JMP 指令执行 2 条微指令，Load 和 Store 指令各执行 4 条微指令(包括取指微指令)。本题没有考虑指令译码微指令。

23. 参照教材中的图 6.6 和表 6.1(见本章附录)画出下述 3 条指令的微程序流程图：

(1) JMP disp(相对寻址)；

(2) Load rs，@rsl(间接寻址)；

(3) ADD rs，rsl(寄存器寻址)。

答：3 条指令的流程图如图 6.12 所示。

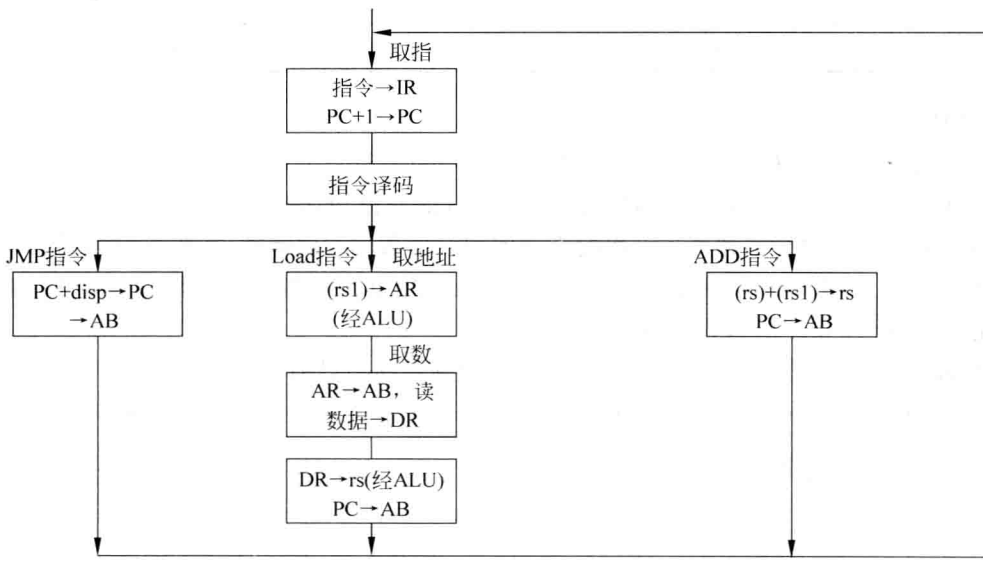


图 6.12 微程序流程图

24. 某机 CPU 结构如图 6.13 所示。A、B、C 均为 8 位寄存器，它们的输入和输出数据的控制信号分别为 IN_A、IN_B、IN_C 和 OUT_A、OUT_B、OUT_C；A、C 还可以串联右移，其移位控制信号为 SHT_{AC}；A 的清空(置 0)控制信号为 CLR_A；D 为计数器，其置数控制信号为 SET_D，减 1 计数器控制信号为 DEC_D；Z 和 S 为状态信号，当 D=0 时置 $Z=1$，S 为 C 寄存器的最低位，“+”为 ALU 的加法控制信号。

该计算机采用微程序控制，微指令格式为

15 ～ 5	4 ～ 3	2 ～ 0
CF	BCF	BAF

其中，CF 为控制字段，采用直接控制法，控制信号从高位到低位顺序为 OUT_A、OUT_B、OUT_C、IN_A、IN_B、IN_C、+、CLR_A、SET_D、DEC_D、SHT_{AC}。

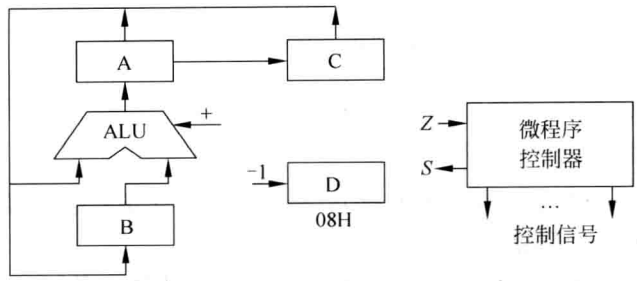


图 6.13　某机 CPU 结构

BCF 为微程序流控制字段，其含义如下表所示：

BCF	含　义	BCF	含　义
00	顺序	10	测试 S，$S=0$ 转移
01	测试 Z，$Z=0$ 转移	11	未定义

BAF 为转移字段地址。

设 B、C 分别存放被乘数和乘数（已存入），且均为无符号定点小数。

请编写实现 B×C→BC（B 存放高位积，C 存放低位积）的微程序。

答：采用循环执行加和移位 8 次的方法实现 8 位数的乘法运算，微程序开始执行后，先在 D 中置入 8，每进行一次移位操作后，D－1，测试 D=0 否？当 D=0 时，表示乘法完成，否则转移到 BAF 所指示的微程序地址。

如果 $S=1$ 时顺序执行，$Z=1$ 时顺序执行，否则转移到 BAF。

地址（二进制）	控制信号	微指令编码			说　明
		CF	BCF	BAF	
000	CLR_A　SETD	000 000 01100	00	000	清 A，置 D
001	测试 S　BAF(011)	000 000 00000	10	011	测试 C 的最低位S
010	OUT_A，OUT_B，+，IN_A	110 100 10000	00	000	A+B→A
011	SHT_{AC}	000 000 00001	00	000	AC 级联右移
100	DEC_D	000 000 00010	00	000	计数器 D－1
101	测试 Z　BAF(001)	000 000 00000	01	001	测试 D=0 否
110	OUT_A　IN_B	100 010 00000	00	000	乘法完成，AC 中为乘积，A→B

当顺序执行时，BAF 无用，今填入 000。

结果在 AC 或 BC 中。

25. 在微程序控制计算机中，下一条要执行的微指令地址都有哪些可能的来源？各发生在什么场合？

答：(1) 由指令操作码译码器（实际上由 ROM 组成）产生后继微指令地址，用于根据指令码转入相应微程序入口处。

(2) 以增量方式产生后继微指令地址。μPC+1（增量通常为 1）作为后继微指令地址。

(3) 由微指令的下址字段指出后继微指令地址。

(4) 以增量和下址字段结合产生后继微地址。当顺序执行时，采用增量方式；当微程序转移时，由下址字段指出后继微指令地址。

(5) 多路转移方式。根据硬件状态来决定后继微地址。属于这种状态的可以是根据运算结果所置的标志位(N、Z、V、C)、计数器状态等。根据 1 种状态(1 或 0)来决定微地址可以有两种情况，实现两路转移；根据 2 种状态来决定微地址可以有 4 种情况，实现 4 路转移。

(6) 由硬件设置微程序入口地址。

当机器启动时，由硬件设置第 1 条要执行的微指令地址。

当产生程序中断请求时，由硬件设置微中断程序的入口地址。

(7) 从微中断或微子程序返回的下一条要执行的微指令地址。该地址一般保存在微返回寄存器 μRR 或微堆栈中。

对于具体的微程序控制器，不要求实现上述全部功能。

26. 在图 6.14 所示的电路中，CLK_2 为输入脉冲，请分析 CLK_2、CLK 和 T_1 是什么关系？

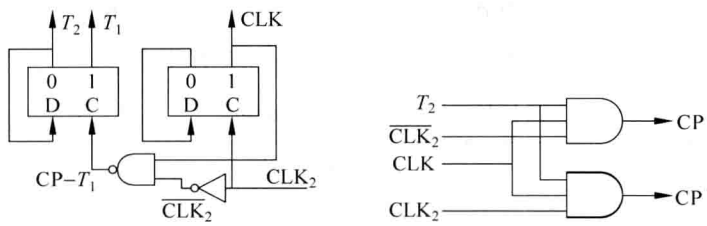


图 6.14　分频电路

已知一级门的延迟时间 a 略少于触发器的翻转时间 b，请画出 CLK_2、$\overline{CLK_2}$、CLK、CP$-T_1$、T_1 和 CP 的时间关系图，其中 $CP=T_2 \cdot CLK \cdot \overline{CLK_2}$，$T_2=\overline{T}_1$。如果用一级与门实现 $CP'=T_2 \cdot CLK \cdot CLK_2$，是否能产生超前于 CP 的工作脉冲？

答：CLK_2 二分频得到 CLK，CLK 二分频得到 T_1 和 T_2。图中的两个 D 型触发器，其“0”输出端连到 D 输入端，当 C 端输入脉冲时，在上升沿触发下，进行计数。

$\overline{CLK_2}$、CLK_2、CLK、CP$-T_1$、T_1(或 T_2)、CP 与 CP′ 的时间关系如图 6.15 所示。图 6.15(a)表示波形发生跳变的因果关系。图 6.15(b)相邻两虚线之间的时间为 a，并假设 $b=2a$，按照本题给出的图与延迟时间而画出的波形图。

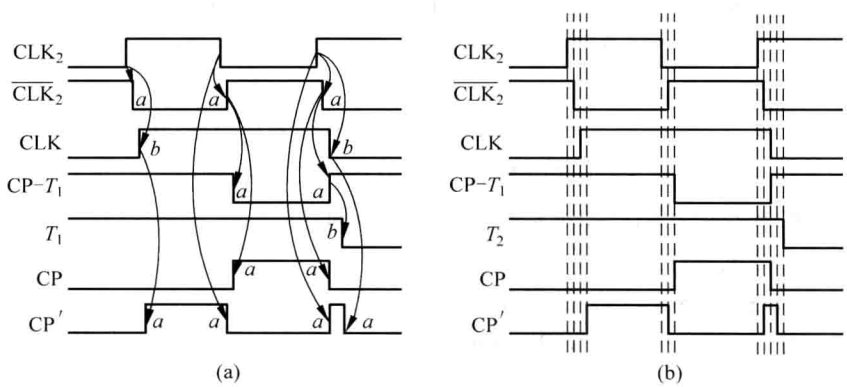


图 6.15　波形图

CP′除了产生超前于 CP 的脉冲以外，还产生毛刺，这是由于 T_2 的下降延迟于 CLK 和 CLK_2 的上升沿和 CLK 的下降沿延迟于 CLK_2 的上升沿而产生的，所以 CP′不能用作工作脉冲。

27. 请设计一个能实现本章第 10 题图 6.7 指令执行流程图的机器周期时序电路。

答：从流程图可见，执行一条时间最长的指令需要 7 个机器周期。今采用 7 个 D 型触发器 $T_1 \sim T_7$ 来实现 7 个机器周期 $T_1 \sim T_7$。(如图 6.16 所示)在任意时候，有一个触发器处于 1 状态，其余均处于 0 状态。在机器启动时，将触发器 T_1 置成 1 状态，其余触发器均置为 0。时钟信号加到触发器的脉冲输入端，一个机器周期有一个时钟脉冲。根据图 6.7 的流程图，得出触发器 D 输入端信号的表达式，当该信号为 1 时，在时钟脉冲作用下，将相应的触发器置成 1，否则为 0。代表各机器周期触发器的输入(0 端)条件如表 6.1 所示。

从流程图可见，T_1 的下一周期必为 T_2，T_2 的下一周期必为 T_3。所以其逻辑表达式为：$T_2 = T_1$，$T_3 = T_2$。在逻辑图上将 T_1 的输出端连接到 T_2 的 D 输入端，将 T_2 的输出端连到 T_3 的 D 输入端。

$$\begin{aligned} T_4 &= T_3 \cdot (\text{算逻指令} + \text{转子程序指令} + \text{堆栈指令}) \\ &= T_3 \cdot \overline{(\text{无条件转移指令} + \text{条件转移指令})} \end{aligned}$$

表 6.1　机器周期转换条件

机器周期	条　　件	机器周期	条　　件
T_2	T_1	T_6	$T_5 + T_4 \cdot (\text{算逻指令} \cdot (M=00))$
T_3	T_2	T_7	$T_6 \cdot (\text{算逻指令} \cdot (M=01+M=10))$
T_4	$T_3 \cdot \overline{(\text{无条件转移指令} + \text{条件转移指令})}$	T_1	$T_7 + T_6 \cdot (\text{算逻指令} \cdot (M=00)) + T_4 \cdot (\text{转子指令} + \text{堆栈指令}) + T_3 \cdot (\text{无条件转移指令} + \text{条件转移指令})$
T_5	$T_4 \cdot (\text{算逻指令} \cdot (M=01+M=10))$		

$T_5 = T_4 \cdot (\text{算逻指令} \cdot (M=01+M=10))$，在不考虑扩充指令的情况下(下同)，可简化为

$$T_5 = T_4 \cdot \text{算逻指令} \cdot \overline{M=00}$$

T_6：进入 T_5 周期的指令，下一步必进入 T_6，此外算逻指令 $\cdot (M=00)$ 从 T_4 直接进入 T_6，所以

$$T_6 = T_5 + T_4 \cdot \text{算逻指令} \cdot (M=00)$$

$T_7 = T_6 \cdot \text{算逻指令} \cdot (M=01+M=10)$，由于进入 T_6 的仅有算逻指令，所以可简化为

$$T_7 = T_6 \cdot (M=01+M=10) = T_6 \cdot \overline{M=00}$$

同理 T_1 简化为

$$\begin{aligned} T_1 &= T_7 + T_6 \cdot (M=00) + T_4 \cdot (\text{转子指令} + \text{堆栈指令}) \\ &\quad + T_3 \cdot (\text{无条件转移指令} + \text{条件转移指令}) \end{aligned}$$

假设指令的操作码和编码如本章第 11 题所示，则：

式中算逻指令的操作码为 $\overline{OP_3} \cdot \overline{OP_2} = F_1$

(无条件转移＋条件转移)的操作码

$= \overline{OP_3} \cdot OP_2 \cdot \overline{OP_1} \cdot \overline{OP_0} + \overline{OP_3} \cdot OP_2 \cdot \overline{OP_1} \cdot OP_0$

$= \overline{OP_3} \cdot OP_2 \cdot \overline{OP_1}$

$= F_2$

（转子程序＋堆栈指令）的操作码

$=\overline{OP_3}\cdot OP_2\cdot OP_1\cdot\overline{OP_0}+OP_3$

$=\overline{OP_3}\cdot OP_2\cdot OP_1\cdot\overline{OP_0}+\overline{OP_3}\cdot OP_2\cdot OP_1\cdot OP_0+OP_3$

$=\overline{OP_3}\cdot OP_2\cdot OP_1+OP_3$

$=OP_2\cdot OP_1+OP_3=F_3$

（因为$\overline{OP_3}\cdot OP_2\cdot OP_1\cdot OP_0$ 指令不存在，所以不会出现，加上该项不会出错）

$$\overline{M}_1\overline{M}_0=F_4$$

则可得图 6.16 所示的逻辑图。

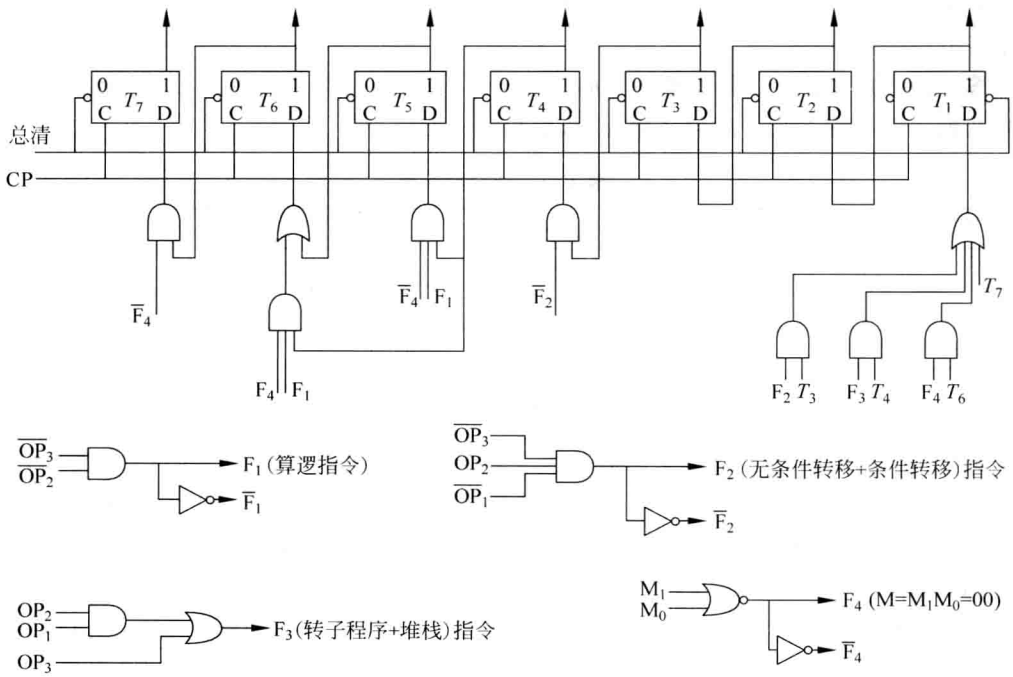


图 6.16　机器周期逻辑图

上述的实现方法不是唯一的，例如，可用 3 个触发器（$C_3C_2C_1$），利用其中的 7 种状态（001、010、…、111）通过一个译码器输出（000 不用），如图 6.17 所示。

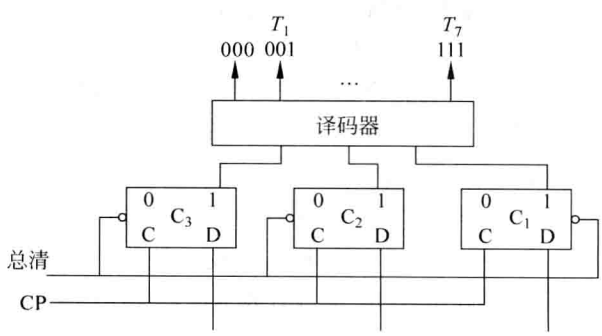


图 6.17　机器周期流程图

其 D 端的输入信号可考虑如下：

触发器 C_1 的 D 端信号应是图 6.16 中的 T_1、T_3、T_5 和 T_7 的 D 端信号的总和(逻辑或)，因为产生 T_1、T_3、T_5 和 T_7 信号时，触发器 C_1 应处于 1 状态。同理 C_2 的 D 端信号应是图 6.16 中的 T_2、T_4 和 T_6 的 D 端信号的总和(逻辑或)，C_3 的 D 端信号应是图 6.16 中的 T_4、T_5、T_6 和 T_7 的 D 端信号的总和(逻辑或)。

由于触发器翻转时间有差异，所以译码器的输出有毛刺，这一般不影响计算机正常运行。时钟脉冲则不允许有毛刺。

28. 今有 3 位计数器，其 8 个译码输出不允许有毛刺，应该如何设计编码？写出最低位 D 型触发器的输入逻辑表达式。如果该计数器仅允许有 7 种状态，是否有可能保证译码器的输出不产生毛刺？

答：如果加 1 后的计数器编码与加 1 前相比，只改变 1 个触发器的值，可保证译码输出没有毛刺。例如按表 6.2 两种次序编码，输出无毛刺。

表 6.2　计数器编码

次序	编码(一)	编码(二)	次序	编码(一)	编码(二)
0	000	000	4	101	110
1	001	001	5	100	111
2	011	011	6	110	101
3	111	010	7	010	100

今以 C_3、C_2 和 C_1 表示计数器的触发器“1”端的输出，按第 1 种次序编码，最低位 D 型触发器的输入 $=\overline{C_3}\,\overline{C_2}\,\overline{C_1}+\overline{C_3}\,\overline{C_2}C_1+\overline{C_3}C_2C_1+C_3C_2C_1=\overline{C_3}\,\overline{C_2}(\overline{C_1}+C_1)+C_2C_1(\overline{C_3}+C_3)=\overline{C_3}\,\overline{C_2}+C_2C_1$。按第 2 种次序编码，最低位 D 型触发器的输入 $=\overline{C_3}\,\overline{C_2}\,\overline{C_1}+\overline{C_3}\,\overline{C_2}C_1+C_3C_2\overline{C_1}+C_3C_2C_1=\overline{C_3}\,\overline{C_2}+C_3C_2$。

如果计数器仅允许有 7 种状态，则不可能保证加 1 后的计数器编码与加 1 前相比都能满足只改变 1 个触发器的值的要求。假如按正常的次序(0～6)计数 $000\xrightarrow{+1}001\xrightarrow{+1}010\xrightarrow{+1}011\xrightarrow{+1}100\xrightarrow{+1}101\xrightarrow{+1}110\xrightarrow{+1}000\longrightarrow\cdots$，则从 001→010 有两个触发器改变状态，从 011→100 时有 3 个触发器改变状态，……，今假设各触发器翻转延迟时间不同，且 $C_3>C_2>C_1$，那么从 001→010 时，由于 C_1 翻转得快，因此在变码的瞬间有可能会产生 000 编码，虽然时间极短，但有可能在译码的 000 输出线上出现一个小脉冲(毛刺)。同样从 011→100 时有可能在译码的 010 和 000 输出线上出现小脉冲。

29. 今有一嵌入式计算机，所有指令的操作都在 4 个机器周期中完成，每个机器周期含 1 个时钟脉冲。4 个机器周期信号由两位计数器经译码器输出，计数器的变化状态为 00→01→10→11→00→…(不考虑毛刺)，CP 为正脉冲。

(1) 设计实现上述 4 个机器周期的逻辑图，并画出波形图。

(2) 如果第 1 周期和第 4 周期访问存储器(各完成取指和存/取数据的操作)，而存储器完成操作的时间不固定，当存储器发出 ready 信号时表示存取操作已完成，试完成该逻辑图。

答：(1) 根据设计要求，两位计数器的输出 T_2、T_1 和 4 个机器周期 C_4、C_3、C_2、C_1 与时钟的关系如图 6.18 所示。

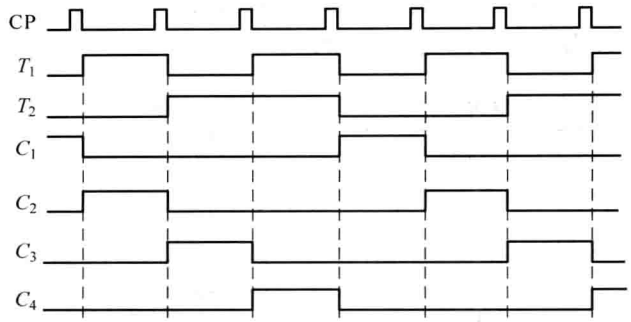


图 6.18　计数器与机器周期时序图

选择 D 型触发器利用 CP 的下降沿(即$\overline{CP}$的上升沿)触发。按正常的计数器工作,当 T_1 为 1,来$\overline{CP}$脉冲时,T_1 将翻转为"0",并向高位进位(T_2+1);而当 T_1 为 0,来$\overline{CP}$脉冲时,T_2 保持不变。根据 D 型触发器的性能,将其输出端"1"连接到 D 端时,在$\overline{CP}$作用下,触发器状态不变;将其输出端 0 连接到 D 端时,在$\overline{CP}$作用下,触发器将计数。于是得到图 6.19 所示的逻辑图。

另外也可根据表 6.3 给出的真值表列出逻辑表达式。

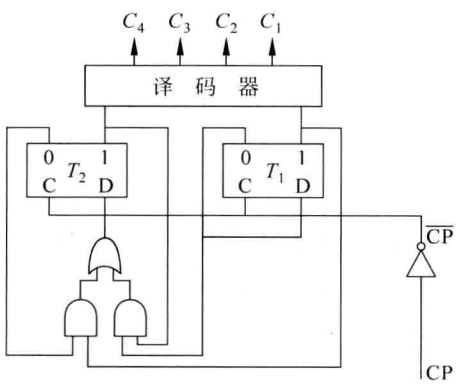


图 6.19　产生 4 个机器周期的逻辑图

表 6.3　计数器的真值表

原态		次态	
T_2	T_1	T_2	T_1
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

从表中可见,T_1 为计数状态,$T_1=\overline{T}_2\cdot\overline{T}_1+T_2\cdot\overline{T}_1=\overline{T}_1$,$T_2=\overline{T}_2\cdot T_1+T_2\cdot\overline{T}_1$。可同样用图 6.19 所示的逻辑图表示。

(2) 可用两种方法实现:

① 用$\overline{\text{ready}}$封锁 CP 时钟。$CP'=CP\cdot ready$,要求 ready 信号提前于 CP 到达,且迟后于 CP 的撤销,以保证 CP′信号的完整性,然后用 CP′取代图 6.19 中的 CP。

② 用 ready 控制计数器数据输入端。根据表 6.4 所示的真值表列出逻辑表达式。

$$T_1=\overline{T}_2\cdot\overline{T}_1\cdot ready+T_2\cdot\overline{T}_1+T_2\cdot T_1\cdot\overline{ready}$$

$$T_2=\overline{T}_2\cdot T_1+T_2\cdot\overline{T}_1+T_2\cdot T_1\cdot\overline{ready}$$

根据逻辑表达式画出图 6.20 所示的逻辑图。触发器、译码器及 CP 脉冲部分同图 6.19。

比较①与②两种方法,①的方法似乎比较简单,但由于 ready 信号与 CP 不同步,因此有

可能得到不完整(变窄)的 CP′而引起不可靠;而采用第②种方法,即使 ready 信号延后于 CP,仅是延迟 1 个时钟周期而已,不会引起错误。

表 6.4　考虑 ready 后的计数器真值表

原态			次态	
T_2	T_1	ready	T_2	T_1
0	0	0	0	0
0	0	1	0	1
0	1	×	1	0
1	0	×	1	1
1	1	0	1	1
1	1	1	0	0

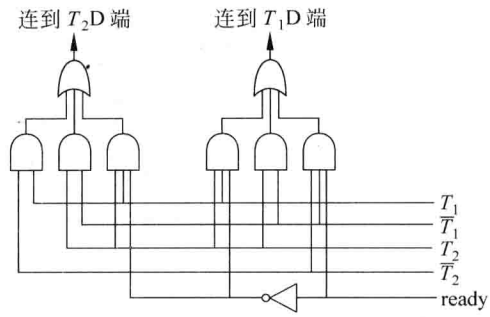


图 6.20　D 输入端逻辑图

本题利用 CP 的下降沿触发计数器,如果机器其他部分(例如 PC、IR 等)都利用 CP 的上升沿将信号打入寄存器,此时机器周期 C_1、C_2、C_3、C_4 都处于稳定状态,即使这些信号有毛刺,也不会影响操作的正确性。

30. 在计算机中实现乘法运算一般可用软件、硬件(组合逻辑)和微码控制 3 种方法。请简述:

(1) 实现上述 3 种方法的基本原理。

(2) 各种方法实现时所需配备的硬件设备比较。

(3) 各种方法速度比较。

答:(1) 下面简述这 3 种方法。

① 软件实现乘法运算:

在早期的小型机或微型机中,没有专设的乘法指令,而是用一段程序,采用重复循环执行多次乘加和移位指令的方法来实现乘法运算,其目的是简化 CPU,节省硬件,降低成本。

② 硬件(组合逻辑)实现乘法运算:

机器设置乘法指令,其控制信号由组合逻辑电路产生,可用逻辑表达式来表示。

③ 微码控制实现乘法运算:

机器设置乘法指令,一条指令由若干条微指令完成,微指令存放在控制存储器中。微码控制也是一种硬件实现,只是其控制信号的形成方式与组合逻辑不同。

(2) 软件实现时所需的硬件设备最省,运算器比较简单,控制信号较少,设计与调试也比较容易。组合逻辑与微码控制的运算器基本相同,控制信号的形成方法不同,组合逻辑为了节省电路采取多种方法化简逻辑表达式,设计与调试都比较复杂,当出现错误或需修改设计时处理起来比较麻烦;微码设计比较规整,调试和修改相对容易一些,因此 CISC 计算机一般采用微码控制。

(3) 组合逻辑实现乘法最快,微码控制因为要到控制存储器中取微指令,稍慢一些,软件要执行多条指令,速度要慢很多。

31. 如果用软件实现二进制定点乘法运算,请为该计算机设计实现乘法运算的指令系统(已知相乘两数均为正数,数值部分的长度为 n 位),设计指令操作码时可以不考虑执行其他操

作的指令。

请编写实现乘法运算的程序。

答：执行乘法运算需要 3 个寄存器，假设 A 寄存器存放部分积高位，B 寄存器存放被乘数，D 寄存器存放乘数和部分积低位，乘法操作是通过 n 次加法运算（A＋B 或 A＋0）和移位操作实现的。根据 D 寄存器的最低位来决定是否要进行 A＋B 操作，A 和 D 串联在一起进行移位操作。今用循环移位（进位 C 参与移位）指令实现右移操作。

D 循环右移 1 位，使最低位移入 C，然后用 C 来判断是否进行 A＋B（当 C＝1 时）或不操作（当 C＝0 时）。

A 和 D 串联移位是通过以下两步实现的：①A 循环右移 1 位，将最低位移入 C；②D 循环右移 1 位，将第①步移入 C 的内容送入 D 的最高位，并将 D 的最低位移入 C。

运算部件的设置如下：

A 寄存器（n＋1 位），B 寄存器（n 位），D 寄存器（n 位），加法器（n 位），状态位 C（进位）和 Z（结果为 0）。另外还有一个记录加法运算次数的计数器 N，其初始值为$[-n]_{补}$，每进行一次加法运算（A＋B，或不操作）后加 1，当（N）＝0 时，表示加法运算的次数已有 n 次。

满足本题的指令系统如下：

```
(1) CLR  A        ;清除 A 寄存器
(2) CLR  C        ;清除进位位 C
(3) SET  N        ;N 置以[-n]补值
(4) shfR          ;循环右移 1 位
(5) Branch        ;C=0 或 Z=0 转移，否则顺序执行
(6) ADD           ;A+B→A，根据运算结果置 C 和 Z
(7) INC  N        ;计数器 N+1，根据运算结果置 C 和 Z
```

二进制定点乘法运算的子程序如下：

```
k: CLR   A          ；清 A
   CLR   C          ；清 C
   SET   N          ；N 置以初始值[-n]补
   shfR  A          ；A 右移 1 位，第 1 次执行移位操作时，因为A=0，C=0，实际不起作用
   shfR  D          ；D 右移 1 位，并将最低位移入 C
   Branch  C̄,disp   ；条件转移，C 为转移条件，disp 为相对 PC 的位移量，disp=2
   ADD   A,B        ；(A)+(B)→A
   INC   N          ；(N)+1→N。置 Z 和 C
   Branch  Z̄,disp   ；条件转移，(N)≠0(即 Z=0)转移，disp=-5
   CLR   C          ；清 C
   shfR  A          ；A 右移 1 位
   shfR  D          ；D 右移 1 位
   结束             ；返回原程序
```

如果不考虑其他指令，那么指令长度可设定为 1 字节。

今将指令长度设计为 1 字节，假设寄存器有 8 个（包括 A、B、D、N 及未用到的 4 个寄存器），指令的格式及操作码的编码如下：

指令格式

(1)

2 位	3 位	3 位
操作码	地址码 R_1	地址码 R_2

(2)

2 位	2 位	4 位
操作码	转移条件	位移量 disp

(3)

5 位	3 位
操作码	地址码

指令	操作码编码
ADD R_1 R_2	00
Branch	01
CLR A	10000
CLR C	10001
shfR	10010
SET	10011
INC	10100

最后需要指出的是：凡是设计题，其答案可以（而且肯定）有多种，所以重要的是掌握方法。例如第 5 章第 34 题编写的乘法子程序所采用的指令及其符号与本题有差异，进一步说明了计算机的指令系统没有达到标准化程度。

32. 接上题，如果乘数与被乘数可以是负数，如何得到乘积的符号位？

答：假设乘数与被乘数的最高位为符号位，对这两个数据进行按位加运算（要增加一条按位加指令），其结果的符号位即为乘积的符号位（数值部分无用）。也可以再进行一次循环左移操作，结果的符号即移到进位触发器（状态位 C）中。

数值部分相乘仍按上题进行。

33. 假设某计算机采用 4 级流水线（取指、译码、执行、送结果），其中译码级可同时完成从寄存器取数的操作，并假设存储器的读写操作（允许同时取指和取数）可在一个机器周期内完成，问顺序执行下列的 3 条指令总共需要多少周期？

(1) JMP disp（相对寻址）　　JMP 为无条件转移指令

(2) Load rs，@rsl（间接寻址）　　Load 从存储器取数到寄存器

(3) ADD rs，rsl（寄存器寻址）　　ADD 为加法指令

答：3 条指令的流水线如图 6.21 所示。

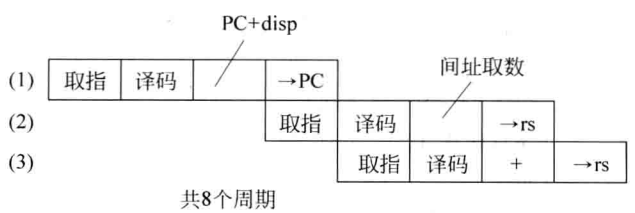


图 6.21　3 条指令的流水线

假设 PC+disp 的结果直接作为下一条指令的地址去取指，总共需要 8 个周期完成 3 条指令。假如下一条指令地址要从 PC 中取，则还需增加 1 个周期，即取第(2)条指令要在第

(1)条指令的“→PC”操作完成后才能进行。

34. 接上题，如果分别用硬布线和微程序两种方法实现，是否会影响所需的周期数？

答：一般影响不大，但与具体设计有关。

35. 在教材图 6.30 的 MB86901 逻辑框图中，为什么要有 3 个指令寄存器 D、E、W 和 3 个程序计数器 DPC、EPC、WPC？

答：因为这是一个 4 级流水线的 CPU 逻辑图，该 CPU 有可能同时执行 4 条指令（如图 6.22 所示），每条指令（取指除外）都需要有一个指令寄存器来控制其操作，取指操作对每条指令都是一样的，不受指令寄存器控制。D、E、W 分别表示处于译码、执行和写阶段的指令寄存器，当一条指令完成“译码”操作后就将该指令从 D 寄存器送到 E，D 接收下一条指令；同样当 E 寄存器中的指令完成“执行”操作后，将该指令从 E 送到 W。由于在每一级流水线中都完成了一定的操作，所以下一级的指令寄存器所需提供的信息会少些，例如 W 寄存器仅需提供“写入”的信息即可。这说明一条指令在各个寄存器中传送时可以有一些变化。

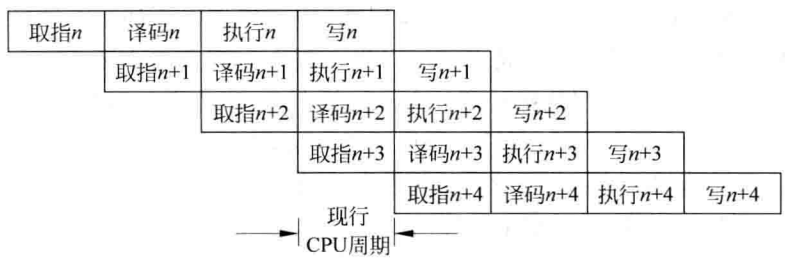


图 6.22　流水线

DPC、EPC、WPC 各表示 D、E、W 中的指令地址，由于转移类指令的存在，DPC、EPC、WPC 表示的地址可能不是按顺序的，因此需要设置 3 个程序计数器，当程序由于某些原因而中断时（例如运算溢出），需要知道确切的断点，在程序能继续执行时可保证正确运行下去。

各级流水线间的联系是通过寄存器（例如 D、E、W、DPC、EPC、FPC 均为寄存器）实现的，而不能依靠逻辑电路联系，因为在同一机器周期（图 6.22 中的现行 CPU 周期）中，各流水线在执行不同的指令。

36. 在图 6.22 中，如果第 $n+1$ 条指令在“执行”段运算结果溢出，流水线该如何处理？

答：当发生溢出时，运算结果不该写入，并转移到相应的中断程序运行，已取入 CPU 控制器的第 $n+2$ 和 $n+3$ 条指令不再执行，而予以取消。并将 EPC 保留起来，以保证在处理完中断程序后能从 EPC 继续运行下去。

37. 接上题，如果第 $n+1$ 条指令在“执行”段遇到外部设备发来的中断请求信号，该如何处理？在非流水线计算机中，将在何时响应中断请求？

答：在非流水线计算机中，一般在本条指令执行完后响应外部设备中断请求，并将下一条要执行的指令地址（PC）保存起来，以备处理完中断程序返回原程序时继续运行下去。

在流水线计算机中，遇到外部设备中断请求一般有两种处理方法。

(1) 程序继续执行，但新的指令将按中断的要求从存储器中取出，并将已进入流水线的指令执行完。

(2) 立即中止原有程序的执行，按上题处理溢出中断的方法进行处理。

38. 从供选择的答案中选出正确答案。

某机采用两级流水线组织，第一级为取指、译码，需要 200ns 完成操作；第二级为执行周期，大部分指令能在 180ns 内完成，但有两条指令要 360ns 才能完成，在程序运行时，这类指令所占比例为 5%～10%。

根据上述情况，机器周期(即一级流水线操作时间)应选为 A 。两条执行周期长的指令采用 B 的方法解决。

供选择的答案

A：①180ns；②190ns；③200ns；④360ns。

B：①机器周期选为 360ns；②两个机器周期完成。

答：A——③；B——②。

在流水线计算机中，在流水线的各个机器周期中，相邻的指令处于不同的流水线，所以机器周期应选择两级中的较大的时间。虽然在指令集中有两条指令要 360ns 才能完成，但所占比例仅为 5%～10%，采用特殊处理的方法比较合适，因此机器周期选为 200ns。

两条执行时间长的指令，采用延长机器周期的方法，比较简单的方法是采用两个机器周期完成，其流水线情况如图 6.23 所示。该图执行 6 条指令，其中第 $K+2$ 条指令的执行周期为 2 个机器周期。显然这比将整个机器周期选为 360ns 要快得多。

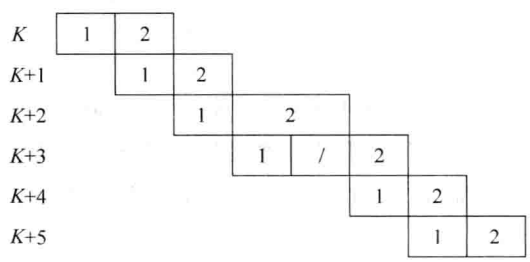


图 6.23　延长机器周期的指令流水线

39. 造成流水线阻塞的因素有多个。试列举 3 个造成流水线阻塞的因素，并给出其中两个的化解措施。

答：数据相关、转移指令和中断请求都会造成流水线阻塞。

数据相关的处理见本章第 43 题。

转移指令的处理见本章第 40 题。

中断请求的处理见本章第 36 题。

40. 请设计满足 RISC 机 SPARC 的 JMPL 指令功能的指令流水线。

答：RISC 计算机在转移指令的下一条安排的是必须执行的指令，JMPL 指令在流水线的执行段计算转移地址，图 6.24 是可供选择的一种流水线设计，假设第 n 条指令是转移指令，t 是转移地址，指令执行的次序是 $n, n+1, t, \cdots$，其中 $n+2$ 条指令被取消。

41. 在采用 MB86901 的寄存器重叠窗口的 SPARC 结构中，执行 A、B 两段程序，其中

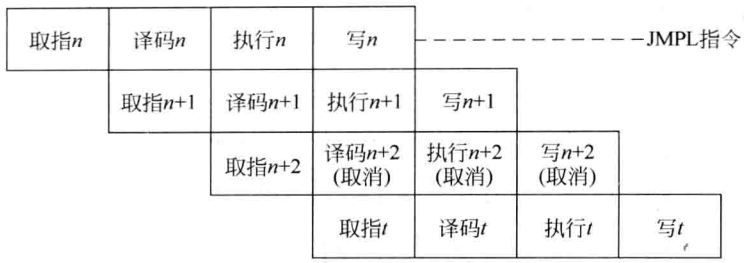


图 6.24　执行转移指令的流水线

$K,K+1,\cdots,K+n$ 为指令地址，已知 CALL 指令执行延迟转移功能，转移到 $K+n$ 入口，SAVE 指令执行 CWP－1 操作，NOP 为不操作指令。请问当分别执行完程序 A 和程序 B 的 $K+n$ 指令后，在寄存器中保存的内容是否相等(分 ADD 执行 r[19]＋r[20]→r[21]和 r[2]＋r[3]→r[4]两种操作讨论)？

程序 A		程序 B	
K	CALL	K	NOP
$K+1$	SAVE	$K+1$	NOP
⋮	NOP	⋮	⋮
	NOP		NOP
$K+n$	ADD	$K+n$	ADD
⋮	⋮	⋮	⋮

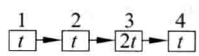
(假设在执行程序段 A 或 B 之前，执行同一程序)

又在程序 B 执行一串 NOP 指令时，遇到 I/O 设备申请中断，与无中断请求相比较，在执行 $K+n$ 指令后，在寄存器中保存的结果是否相同？

答：r[19]、r[20]和 r[31]是局部寄存器，当窗口不相同时，其实际表示的物理寄存器是不同的，程序 A 执行调用指令 CALL 和 SAVE 指令，当 $K+n$ 条指令执行 r[19]＋r[20]→r[21]操作时，程序 A 和程序 B 的寄存器窗口不同，所以物理寄存器中的内容是不同的；当执行 r[2]＋r[3]→r[4]操作时，用的是公用的全局寄存器，所以程序 A 和程序 B 执行 $K+n$ 指令后，在寄存器中保存的内容是相等的。

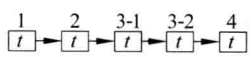
执行中断程序不会影响运算结果，因为中断处理是在下一个窗口进行的。

42. 假设某计算机执行一条指令分 4 段完成，每段所需时间如下所示。

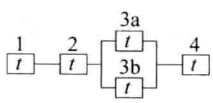


如果要按流水线设计，该如何安排比较合理？

答：如果第 3 段能分成 2 段(3-1 和 3-2)，则可按以下方法分段，实际上成了 5 级流水线。



如果第 3 段无法分成两个独立的段，则安排 2 套执行第 3 段所需的设备(3a 和 3b)，如下所示。



其流水线时空图如图 6.25 所示。

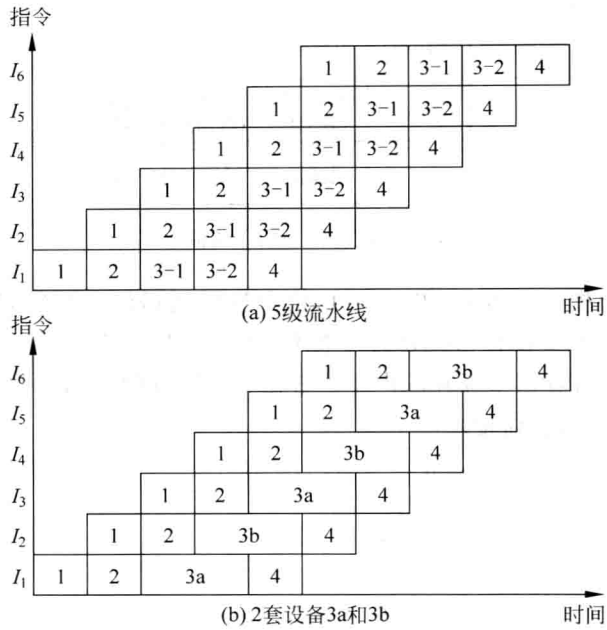


图 6.25　流水线执行时间(时空图)

43. 某一台流水线处理机,包含有取指令、译码、执行 3 个功能段。取指令、译码各需 1 个时钟周期。在执行段,MOV 操作需要 2 个时钟周期(取数、写入),ADD 操作需要 3 个时钟周期(取数、加、写入),MUL 操作需要 4 个时钟周期(取数、MUL、MUL、写入)。在该机上顺序执行下列程序:

K: MOV	R_2, R_3	; $R_2 \leftarrow (R_3)$
$K+1$: MOV	R_1, R_0	; $R_1 \leftarrow (R_0)$
$K+2$: MUL	R_0, R_2, R_1	; $R_0 \leftarrow (R_1) \cdot (R_2)$
$K+3$: ADD	R_0, R_2, R_3	; $R_0 \leftarrow (R_2) + (R_3)$

(1) 画出流水线功能段的结构图。

(2) 画出指令执行过程流水线的时空图。

(3) 为加快速度是否可改进? 如果可以,怎样改进?

答:(1) 功能段结构如图 6.26 所示。

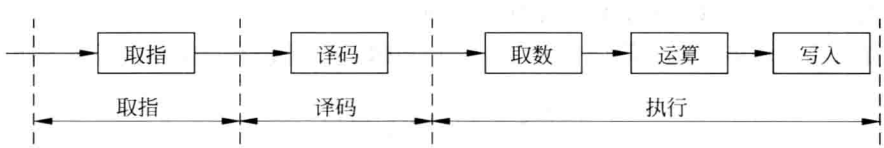


图 6.26　流水线功能段结构图

(2) 流水线的时空图如图 6.27 所示。

$K+2$ 指令取数(R_1)与 $K+1$ 指令数据相关,需延迟 1 个时钟周期;$K+3$ 指令取数不能与 $K+2$ 指令取数同时进行,需延迟 1 个时钟周期;$K+3$ 指令写数(R_0)与 $K+2$ 指令(写

图 6.27　流水线时空图

R_0)数据相关，需延迟 1 个时钟周期。如果运算器不能同时执行加和 MUL 操作，则将 $K+3$ 指令的加操作延迟 1 个时钟周期。

(3) 可以改进，改进措施如下。

- 软件处理：编译优化时，取消 $K+2$ 指令，因为该条指令的结果被下一条指令覆盖，所以该指令无用。
- 硬件处理：在 $K+2$ 指令取数相关时将上一条指令在写入前的数据直接取来，这时需要增加一条通路。在写数相关时，封锁上一条指令的写入操作。

实际情况复杂得多，因为在多级流水线的情况下，本条指令的操作不仅与上一条指令有关，还可能与更前面的指令有关。

经过软硬件处理后的流水线时空图如图 6.28 所示。

图 6.28　改进后的流水线时空图

44. 判断是非并简要说明原因。

(1) 如果两种计算机的主频相同，则它们的运算速度也是相同的。

(2) 控制存储器中的微指令是由用户设计的。

(3) 采用微程序控制器的处理器称为微处理器。

(4) 确定下一条微指令地址时，就是用微程序计数器 μPC 增量(例如 μPC+1)来给出该地址的。

(5) RISC 计算机采用多种寻址方式和多种指令格式。

(6) RISC 机采用循环重叠窗口技术后，使用窗口的子程序嵌套深度可不受限制。

(7) 每条指令的第 1 个机器周期不一定执行取指令操作。

(8) 堆栈计算机的指令都是执行压入和弹出操作。

(9) CPU 的指令都是以同步控制方式实现的。

(10) 可将多个三态门和 OC 门的输出连到一根导线上。

答：以上叙述都有错误，或不全面。

(1) 计算机的运算速度不仅与主频有关，还与 CPU 的结构、存储器的速度、输入输出系统的效率以及操作系统等有关，所以两种不同的计算机，即使主频相同，它们的运算速度也不一定相同，有时差别还很大。

(2) 计算机的指令系统以及实现指令的微程序都是由硬件设计人员和系统软件设计人员共同完成的，对于微程序(微指令)主要是由硬件设计人员实现的。

(3) 微处理器是相对于大型机或小型机的处理器而言的，以上这些处理器都可以采用微程序控制器。

(4) 还可能通过微指令顺序控制的下址字段或由设计者指定的判断字段控制产生后继微指令地址。

(5) RISC 计算机以减少寻址方式和指令格式的方法而使设计简单化，并减少 CPI(每条指令的周期数)来获得运算速度的提高。

(6) 某些 RISC 机采用循环重叠窗口(例如 SUN 公司的 SPARC 机)技术，它的窗口数量与寄存器数量和每个窗口的大小有关，是有限的。采用循环重叠窗口技术后，当所有窗口被嵌套子程序用完后，如果再有子程序调用，它可以使用最早使用的窗口，但是如果不对其原来存放的内容进行处理(一般将它保存到主存储器中)，那么当程序返回时将丢失这一层嵌套内容，造成错误。

(7) 每条指令的第 1 个机器周期应执行取指令操作，因为取指令后才能知道本条指令执行什么操作。某些计算机在执行指令的最后 1 个机器周期将下一条指令的地址送到存储器。

(8) 一般计算机的堆栈指令仅执行进栈(压入)和退栈(弹出)指令，这种计算机不叫堆栈计算机。堆栈计算机除了执行压入和弹出操作外，还有算术逻辑运算等指令，对堆栈中的数据进行运算。

(9) CPU 的操作不是都以同步方式控制的，有的操作时间事先很难确定，因而需要采用异步方式控制，例如到存储器取数，如果所取的数据在高速缓冲存储器中，很快能取到，否则到主存中去取，其时间长得多。

(10) 多个三态门的输出可连接到一起。多个 OC 门的输出也可以连接在一起。但 OC 门与三态门的输出不能连接在一起，这是由电路特性决定的。

45. 设有主频为 16MHz 的微处理器，平均每条指令的执行时间为两个机器周期，每个机器周期由两个时钟脉冲组成。

(1) 存储器为“0 等待”，求出机器速度。

(2) 假如每两个机器周期中有一个是访存周期，需插入 1 个时钟周期的等待时间，求机器速度。(“0 等待”表示存储器可在一个机器周期完成读/写操作，因此不需要插入等待时间。)

答：(1) 机器速度$=\dfrac{\text{主频}}{\text{机器周期}\times\text{时钟脉冲}}=\dfrac{16}{2\times2}=4\text{MIPS}$

(2) 机器速度$=\dfrac{16}{2+(2+1)}=3.2\text{MIPS}$

46. 从供选择的答案中选出正确答案。

微机 A 和 B 是采用不同主频的 CPU 芯片，片内逻辑电路完全相同。若 A 机的 CPU 主频为 8MHz，B 机为 12MHz。则 A 机的 CPU 主振周期为 $\boxed{A}$ μs。如 A 机的平均指令执行速度为 0.4MIPS，那么 A 机的平均指令周期为 $\boxed{B}$ μs，B 机的平均指令执行速度为 $\boxed{C}$ MIPS。

供选择的答案

A～C：①0.125；②0.25；③0.5；④0.6；⑤1.25；⑥1.6；⑦2.5。

答：A——①，B——⑦，C——④。

A 机主振周期＝1/8MHz＝0.125μs

A 机的平均指令周期＝1/0.4MIPS＝2.5μs

B 机的平均指令执行速度＝$0.4\text{MIPS}\times\frac{12}{8}=0.6\text{MIPS}$

附　　录

1. 本章第 22 题和第 23 题要用到教材中的图 6.6 和表 6.1，今摘录于后面。

假设 ALU 可以进行加（＋）、减（－）、逻辑加（∨）和逻辑乘（∧）4 种运算，图中的控制信号用符号 1，2，3，…表示，其意义见教材表 6.1，表中的序号与图中控制信号的符号保持一致。

教材表 6.1　控制信号一览表

序号	控制信号	功　　能
1	PC→AB	指令地址送地址总线
2	ALU→PC	转移地址送 PC
3	PC＋1	程序计数器加 1
4	imm(disp)→ALU	立即数或位移量送 ALU
5	DB→IR	取指到指令寄存器
6	DB→DR	数据总线上的数据送数据寄存器
7	DR→DB	数据寄存器中的数据送数据总线
8	rsl→GR	寄存器地址送通用寄存器
9	rs，rd→GR	寄存器地址送通用寄存器
10	(rsl)→ALU	寄存器内容送 ALU
11	(rs)→ALU	寄存器内容送 ALU
12	DR→ALU	数据寄存器内容送 ALU
13	＋	ALU 进行加法运算
14	－	ALU 进行减法运算

续表

序号	控制信号	功　　能
15	$\wedge$	ALU 进行逻辑乘运算
16	$\vee$	ALU 进行逻辑加运算
17	ALU→GR	ALU 运算结果送通用寄存器
18	ALU→DR	ALU 运算结果送数据寄存器
19	ALU→AR	ALU 计算得到的有效地址送地址寄存器
20	AR→AB	地址寄存器内容送地址总线
21	ADS	地址总线上地址有效
22	$M/\overline{IO}$	访问存储器或 I/O
23	$W/\overline{R}$	写或读

2. 本章的第 10 题、第 11 题、第 12 题和第 27 题是一组有关计算机设计的题，通过这些题目可以了解在给定指令系统的情况下如何设计 CPU。设计内容如下：

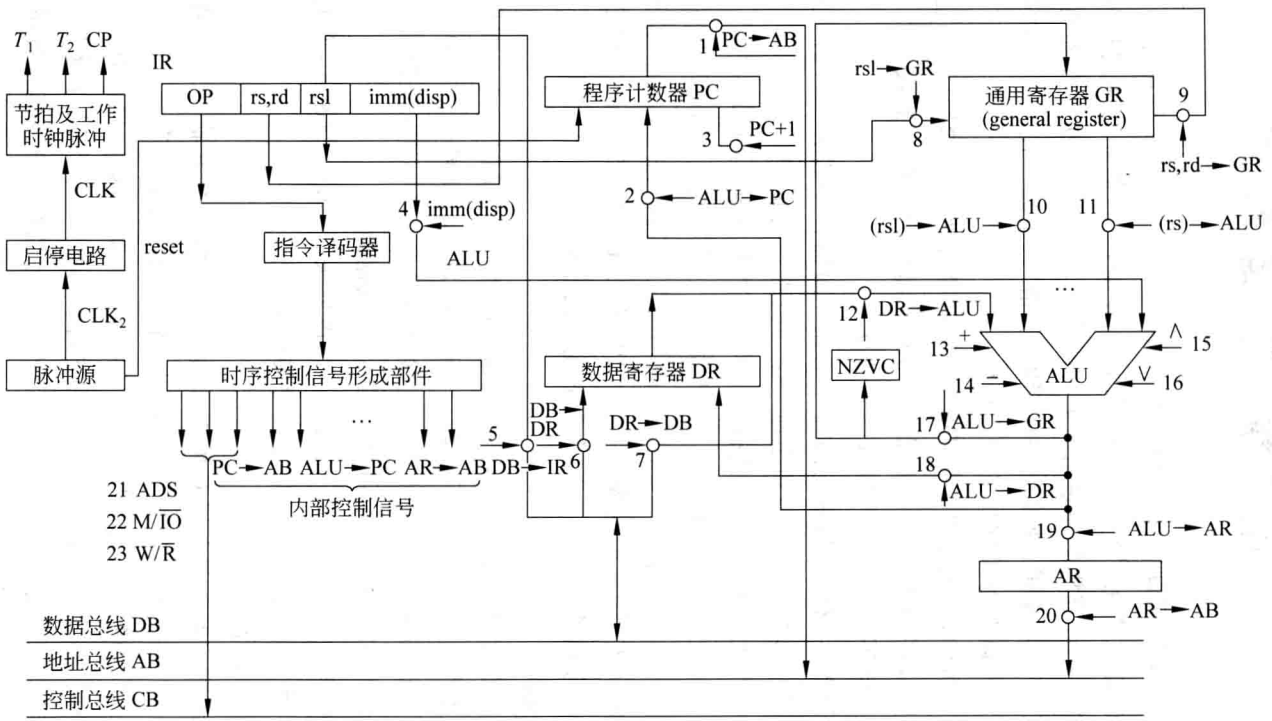
(1) 根据指令系统设计指令执行的流程图(第 10 题)。

(2) 设计指令格式(第 11 题)。

(3) 设计 CPU 总框图(第 11 题)。

(4) 设计组合逻辑控制器/微程序控制器(第 12 题)。

(5) 设计符合指令执行流程图要求的机器周期逻辑图(第 27 题)。



教材图 6.6　CPU(运算控制器)逻辑框图

第7章 存储系统

1. 在计算机中为何设置多层次的存储系统？经常可见的三层次存储系统的情况如何？

答：随着计算机应用、操作系统和 CPU 的发展，主存储器的速度和容量已跟不上客观的要求，因此希望在计算机的体系结构上有所突破。衡量计算机存储器有 3 个指标：速度、容量和价格/位。一般来讲，速度高的存储器，价格/位也高。在计算机中安装有多种类型存储器，相互配合，可以取得较理想的综合指标。

目前的计算机中，一般用半导体 DRAM 存储器作为主存储器（简称主存或内存），存放当前正在执行的程序和数据；而用磁盘、磁带和光盘作为外存储器或辅助存储器（简称外存或辅存），存放当前不在运行的大量程序和数据。

半导体存储器（主存）可随机访问任一单元。而辅存一般为串行访问存储器，不能随机访问，读写该存储器内容时，需要顺序地一位一位地进行成组传送，访问指定信息所需时间与信息所在位置有关。所以 CPU 执行程序时，不能直接从辅存读取指令或存取数据，辅存只能与主存成批地交换（输入或输出）一组数据。CPU 取指令和存取数据都是直接与主存打交道的。然而计算机的主存和 CPU 之间的速度一直保持了大约一个数量级的差距（主存慢），因此限制了 CPU 速度的提高。在 CPU 和以 DRAM 组成的主存之间设置以 SRAM 组成的高速缓冲存储器（cache）是解决存取速度的重要方法。于是构成了三层次存储系统：cache—主存—辅存。

在上述三层次存储系统中，在 CPU 取指或取数时，先访问 cache，如果指令或数据不在 cache 中，再到主存中去取，取来后同时送 CPU 和 cache；如果指令或数据在 cache 中，则从 cache 中取，不再访问主存；如果指令或数据也不在主存中，则由操作系统将包含所需指令或数据的信息块调到主存，然后再执行取指或取数据操作。

三层次存储系统在理想情况下，其最高速度取决于 cache 的速度，而其存储容量则由主存和辅存决定，系统的价格/位接近于辅存的价格/位。

2. 什么是虚拟存储器？

答：操作系统的形成和发展有可能让程序员摆脱主存和辅存之间数据调度问题的困扰，在“辅助硬件”的配合下，通过软件和硬件结合，把主存和辅存统一成了一个整体，在系统中，应用程序员可用机器指令地址码对整个程序统一编址，如同他具有对应这个地址码宽度的全部空间一样。该空间可以比主存实际空间大得多，以致可以存得下整个应用程序。这种指令地址码称为虚拟地址或逻辑地址，其对应的存储容量称为虚存容量或虚存空间；而把实际主存的地址称为物理地址或实（存）地址，其对应的存储容量称为主存容量、实存容量或实（主）存空间。

当应用程序用虚拟地址访问主存时，机器自动地把它经过硬件或辅助软硬件处理变换成主存实地址。查看这个地址所对应的单元内容是否已经装入主存，如果在主存就进行访问，如果不在主存内就经辅助软硬件把它所在的那块程序和数据由辅存调入主存，而后进行访问。这些操作都不必由程序员来安排，也就是说，对应用程序员是透明的。

上述的系统即是虚拟存储器。

主存—辅存层次的基本信息传送单位由操作系统的存储管理采用的方案所决定，有段式、页式和段页式 3 种。

把主存按段分配的存储管理方式称为段式管理。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（模块），例如应用程序段、过程、子程序、数据表、阵列等。段作为独立的逻辑单位一般可以被其他程序段调用。段式管理系统的优点是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，也便于多道程序共享，但是段的长度各不相同，段的起点和终点不能固定，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。

页式管理系统的基本信息传送单位是定长的页。主存的物理空间也被划分为等长的固定区域，称为页面。页面的起点和终点地址是固定的，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。唯一可造成浪费的是程序最后一页的零头（不能利用的页内空间），它比段式管理系统的段外空间浪费要小得多。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方便。

段页式存储管理系统采用的是将段和页结合起来的管理系统，程序按模块分段，段内再分页，主存和辅存之间的基本信息传送单位仍是页面。

3. 什么叫命中率？

答：在 cache—主存—辅存的三级存储系统中，当 CPU 访问存储器时，如果被访问的对象在 cache 中，叫做 cache 命中，cache 命中的次数与访问存储器总次数的比例称为 cache 命中率。

当 CPU 访问存储器时，如果被访问的对象在主存中，叫做存储器命中，否则需启动操作系统，将有关的页面（页式管理或段页式管理）或段（段式管理）从辅存调入主存。

由于存储器访问的局部性原理的存在，cache 的命中率和存储器的命中率都是很高的。

4. 请解释局部性原理。

答：对大量典型程序运行情况的分析结果表明，在一个较短时间间隔内，访存地址（指令或数据）往往集中在逻辑地址空间很小的范围内，称为程序访问的局部性。

在执行程序时程序地址的分布基本上是连续的，再加上循环程序段和子程序段要重复执行多次，因此，对程序地址的访问就自然地具有相对集中的倾向。数据分布的这种集中倾向不如指令明显，但对数组的存储和访问以及工作单元的选择都可以使存储器地址相对集中。这种对局部范围内的存储器地址频繁访问，而对此范围以外的地址则访问甚少的现象就称为程序访问的局部性。

时间局部性：如果一个存储单元被访问，那么这个存储单元很快会再次被访问。

空间局部性：如果一个存储单元被访问，那么这个存储单元及其邻近的存储单元很快会再次被访问。

由于程序访问的局部性，使得三级存储系统的工作很有效。

5. (1) 由容量为 C 的 cache 和容量为 M 的 DRAM 构成的存储器容量是多少？

(2) 由容量为 N 的 ROM 和容量为 M 的 DRAM 构成的存储器容量是多少？

答：(1) cache 中的内容是 DRAM 主存中内容的一部分（复制品），所以存储器的容量为 M。

(2) ROM 和 DRAM 是两个独立的存储部件，存储器容量是 $N+M$。

6. 叙述 cache 的基本结构和地址映像。

答：先介绍地址映像的概念。

为了把信息放到 cache 存储器中，必须应用某种函数把主存地址映像到 cache，称作地地映像。在信息按照这种映像关系装入 cache 后，执行程序时，为了访问 cache，应将主存地址变换成 cache 地址，这个变换过程叫做地址变换。所以地址映像反映了主存地址和 cache 地址的对应关系，也就是说明主存地址中的内容可存放在 cache 的哪些地址中。有 3 种映像：直接映像、全相联映像和组相联映像。

图 7.1 表示 cache 的基本结构。

设主存有 2^n 个单元，地址码为 n 位，将主存分块(block)，每块有 B 个字节，则共分成 $M=2^n/B$ 块。cache 也由同样大小的块组成，由于 cache 容量小，所以块的数目小得多，主存中只有一小部分块的内容可存放在 cache 中。

在 cache 中，每一块外加有一个标记，指明它是主存的哪一块的副本，所以该标记的内容包含主存中块的编号，即图中的主存块号，设主存地址为 n 位，且 $n=m+b$，其中，主存的块数 $M=2^m$，块内字节数 $B=2^b$。cache 地址码为 $(c+b)$ 位。cache 的块数为 2^c。块内字节数与主存相同。

当 CPU 发出读请求时，将主存地址 m 位(或 m 位中的一部分，与地址映像方式有关)与 cache 某块的标记相比较，根据其比较结果是否相等而区分出两种情况：当比较结果相等时，说明需要的数已在 cache 中，可直接从 cache 中读取；如果比较结果不相等，说明数据不在 cache 中，需要把该数据所在的整个字块从主存调入 cache。标记内还有一个“有效位”，表示该标记是否有效。

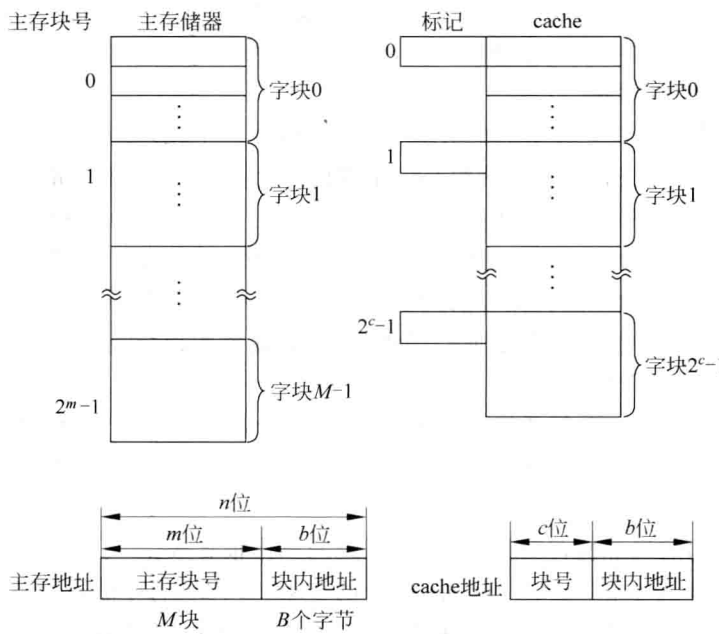


图 7.1　cache 的基本结构

CPU 与 cache 之间，通常一次传送一个字。

字块的大小一般等于在一个主存周期中由主存所能访问到的字数，例如 IBM 370/168 计算机的主存是 4 体交叉存储器，每个存储体是 8B 宽，所以 cache 块的大小为 4×8=32B。

为了便于讨论 3 种地址映像，将图 7.1 中主存的 m 位块号分成区号（t 位）、组号（d 位）和组内块号（r 位），cache 中的 c 位块号分成组号（d 位）和组内块号（r 位）。下面将组内块号简称为块号，于是得主存地址和 cache 地址的对应关系如下：

区号	组号	块号	块内地址	主存地址
t 位	d 位	r 位	b 位	

组号	块号	块内地址	cache 地址
d 位	r 位	b 位	

cache 的容量等于主存一个区的容量，主存地址中的组号、块号和块内地址的位数各与 cache 地址中的组号、块号和块内地址的位数相等。与图 7.1 中的符号关系如下：

$$n = t + d + r + b, \quad m = t + d + r, \quad c = d + r$$

下面讨论 3 种映像方式的 cache 组织。

(1) 直接映像：一组内只有一块，所以块号的 r 位=0。

直接映像的 cache 组织如图 7.2 所示。以组号为地址访问 cache，该图假设组号为 00…01，取出该cache 地址中的标记与主存地址中的区号相比较，如果相等，且标记内的有效位为 1，则为命中，如果 CPU 要取数，则该地址存放的数据即可送往 CPU；如果不等，或有效位为 0，则为不命中。

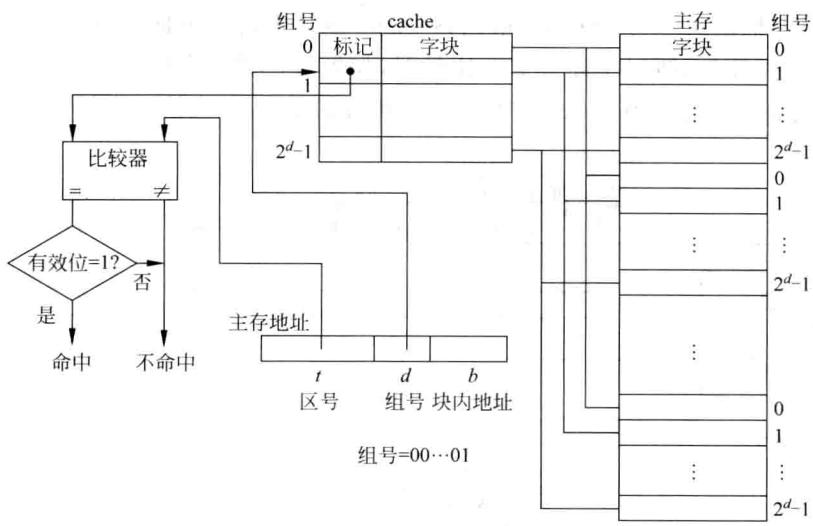


图 7.2　直接映像 cache

(2) 全相联映像：主存中的任一块内容可存放到 cache 的任一块中，所以组号的 d 位=0。

全相联映像的 cache 组织如图 7.3 所示。读出时，主存地址的（区号＋块号）要与 cache 所有字块的标记比较，如有相等的，则为命中。由于比较所需的硬件太多，因此全相联 cache 的方案不被采纳，但可应用于虚拟存储器中。

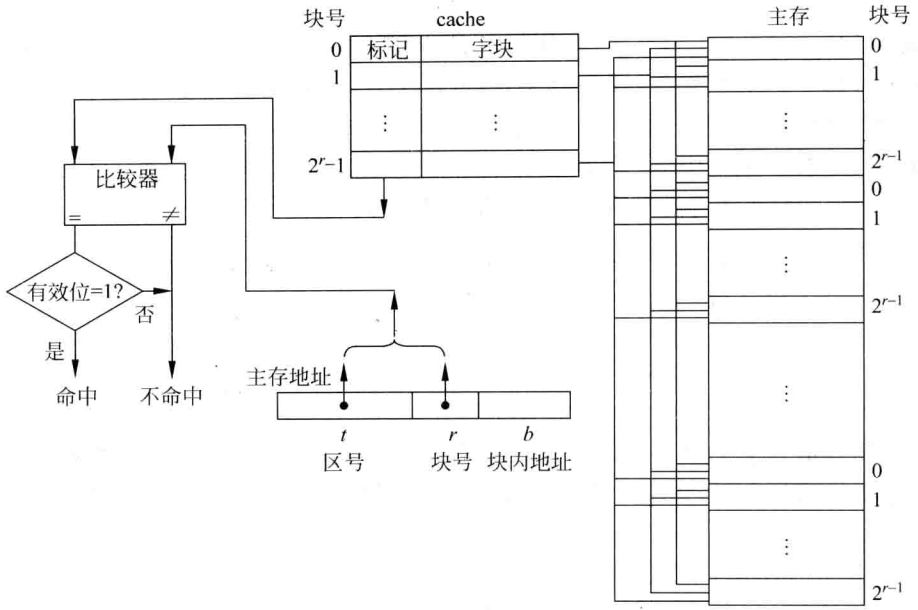


图 7.3　全相联映像 cache

(3) 组相联映像：根据 1 个组内有几块数据，称之为几路组相联，例如有 2 块，则为 2 路组相联，有 4 块则为 4 路组相联。

组相联映像的 cache 组织如图 7.4 所示。以组号为地址访问 cache，该图采用 2 路组相联映像，假设地址的组号为 00…01，取出该 cache 地址中的 2 个标记分别与主存地址中的(区号＋块号)相比较，所以实际有 2 个比较器，如果有任意一个标记与之相等，则为命中。这也说明主存的一个字块在 cache 中有两个位置可供存放，与直接映像相比提高了命中率。

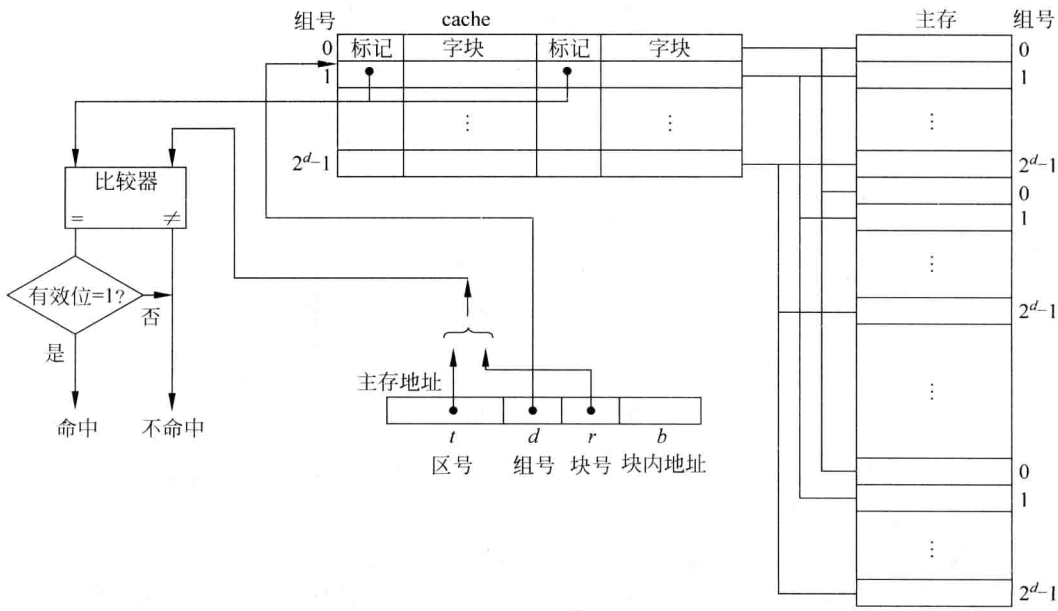


图 7.4　组相联映像 cache

由此可见，在直接映像和组相联映像 cache 中，由组号作为 cache 的地址，其内容只能是主存同一组号中的内容。在组相联映像 cache 中，主存的字块可存放在 cache 同一组内的任一字块中，因此在组相联映像 cache 中，组间为直接映像，组内为全相联映像。

7. 请画出图 7.4 中的比较器和判断命中的逻辑图。

答：根据图 7.4，得出 t 为若干位，r 为 1 位。今假设 t 为 7 位，则 cache 标记的长度为 7 位(t)+1 位(r)+1 位(有效位)，共 9 位。图 7.5 给出本题的逻辑图。

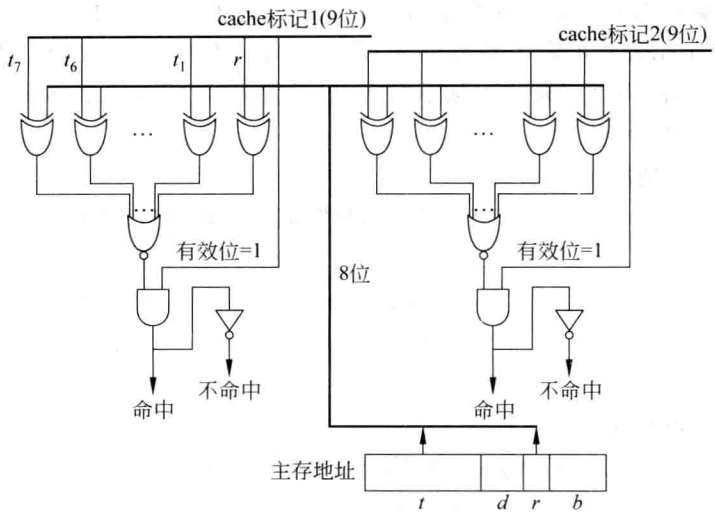


图 7.5　比较器和判断命中的逻辑图

在图 7.5 中，由于 r 为 1 位，所以画出 2 个比较器；如果 r 为 3 位，则有 8 个比较器，实现的是 8 路组相联映像。

8. 设有一个 4 体交叉存储器，单体字长为 32 位，采用直接映像 cache，主存与 cache(或 CPU)之间的数据传送宽度为 32 位，问当 CPU 访存地址为 0001H，又 cache 不命中时，主存与 cache 传送数据的过程是怎样的？

答：在一般情况，单体字块为 4B 的 4 体交叉存储器块的大小选为 4×4B=16B。cache 不命中时从存储器一次可读出 16B，但由于主存与 cache(或 CPU)之间的数据传送宽度为 4B，所以从主存读出后要分 4 次传送到 cache，设主存的存取周期为 T，数据传送的时间间隔为$\frac{T}{4}$，则读取和传送 16B 数据的时间$=T+3\times\frac{T}{4}=\frac{7}{4}T$。这与每隔$\frac{T}{4}$时间轮流从各存储体读出数据的时间相等，一般采用轮流读出的方法。

如果 CPU 访问主存储器地址为 0001H 时不命中，一般先从 1 体取出数据(0001H 地址)，然后依次从 2 体、3 体和 0 体取出地址为 0002H、0003H 和 0000H 的数据送到 cache，CPU 接收到 0001H 的数据就可继续运行程序，如果下次访问 0002H、…，就可命中，直接从 cache 中读出。

9. 设 cache 的命中率为 H_c，cache 的访问时间为 T_c，存储器的访问时间为 T_m，求平均访存时间。

答：cache 的命中率为 H_c，不命中率为 $1-H_c$，不命中时需访问主存，有以下两种计算方法：

(1) CPU 访问 cache 时，同时访问主存。

$$平均访存时间 = H_c T_c + (1 - H_c) T_m$$

(2) CPU 访问 cache，发现不命中时，才开始访问主存。

$$\begin{aligned} 平均访存时间 &= H_c T_c + (1 - H_c)(T_c + T_m) \\ &= T_c + (1 - H_c) T_m \end{aligned}$$

本章以下的习题，如未加说明，都按第 2 种方法计算。

10. 设某计算机有一个指令和数据合一的 cache，已知 cache 的读写时间为 10ns，主存的读写时间为 100ns，取指的命中率为 98%，数据的命中率为 95%，在执行程序时，约有 1/5 指令需要取一个数据，并假设流水线从不阻塞，问设置 cache 后，与无 cache 比较，计算机的运算速度可提高到多少倍？

答：有 cache 的情况如下：

$$\begin{aligned} 平均访存时间 &= 平均取指时间 + 平均取数时间 \\ &= (10ns + (1 - 98\%) \times 100ns) \\ &\quad + (10ns + (1 - 95\%) \times 100ns)/5 \\ &= 12ns + 3ns \\ &= 15ns \end{aligned}$$

无 cache 的情况如下：

$$\begin{aligned} 平均访存时间 &= 平均取指时间 + 平均取数时间 \\ &= 100 \times 1 + 100 \times 1/5 = 120ns \end{aligned}$$

速度提高倍数＝120ns/15ns＝8 倍。

11. 接上题，如果采用哈佛结构（分开的指令 cache 和数据 cache），运算速度可提高到多少倍？

答：采用哈佛结构的流水线计算机，取指与取数可同时进行。如果取数不命中时，正好取指也不命中，此时可仅考虑平均取指时间，等于 12ns。运算速度提高到 120ns/12ns＝10 倍。

如果每次取数不命中时，取指却命中，则

$$\begin{aligned} 平均访存时间 &= 平均取指时间 + 平均取数不命中时间 \\ &= 12ns + (1 - 95\%) \times 100ns \times 1/5 \\ &= 12ns + 1ns \\ &= 13ns \end{aligned}$$

运算速度可提高到 120ns/13ns≈9.2 倍。

实际的平均访存时间应在 12～13ns 之间。

12. 设某计算机的 cache 采用 4 路组相联 cache，已知 cache 容量为 16KB，主存容量为 2MB，每个字块有 8 个字，每个字有 32 位。请回答：

(1) 主存地址多少位（按字节编址），各段如何划分（各需多少位）？

(2) 设 cache 开始为空，CPU 从主存单元 0，1，…，100 依次读出 101 个字（主存为单体，即非交叉存储器，每次读出一个字），并重复按此次序总共读 11 次，问命中率是多少？若 cache 的速度是其不命中时从主存读数速度的 5 倍，问采用 cache 与无 cache 比较速度可提高到多少倍？

答：(1) 主存与 cache 的容量都不考虑扩充。2MB 容量的主存地址为 21 位。16KB 容

量的 cache 地址为 14 位，所以主存地址的区号位数＝21－14＝7 位。cache 和主存地址各段的分配如下：

主存地址：	区号	组号	块号	块内地址	字节
	7	7	2	3	2

cache 地址：	组号	块号	块内地址	字节
	7	2	3	2

(2) 第一轮全部不命中，以后 10 次全部命中，所以命中率＝10/11＝91%。设 cache 的读出时间为 1，主存的读数时间为 5，则速度可提高到

$$\frac{\text{无 cache 时的访问时间}}{\text{有 cache 时的访问时间}}=11\times5/(10\times1+1\times5)=55/15\approx3.67\text{ 倍}$$

略小于此数，因为不设置 cache 时，访问主存的速度会稍快些，无须询问 cache 是否命中。

13. 设某计算机采用直接映像 cache，已知 cache 容量为 4096 字。

(1) 若 CPU 依次从主存单元 0，1，…，99 和 4096，4097，…，4195 交替取指令，循环执行 10 次，问命中率为多少？

(2) 如 cache 存取时间为 10ns，主存存取时间为 100ns，cache 命中率由本题(1)得出，求平均存取时间。

答：(1) cache 容量为 4096 字，其地址为 0～4095。从主存 0～99 和 4096～4195 轮流取指，每次都不命中，所以命中率为 0。

(2) 平均存取时间＝10ns＋(1－0)×100ns＝10ns＋100ns＝110ns。

14. 一个组相联 cache 由 64 个存储块组成，每组包含 4 个存储块，主存包含 8192 个存储块，每块由 32 字组成，4 字节组成 1 字。

(1) 求主存地址有多少位？cache 地址有多少位？其地址映像是几路组相联？

(2) 计算主存地址格式中区号、组号、块号和块内地址字段的位数。

答：(1) 主存容量＝$8192\times32\times4=2^{13}\times2^{5}\times2^{2}=2^{20}$B，主存地址为 20 位。

cache 容量＝$64\times32\times4=2^{6}\times2^{5}\times2^{2}=2^{13}$B，cache 地址为 13 位。

地址映像为 4 路组相联。

(2) 主存地址：块号 2 位；组数＝$\frac{64}{4}$＝16 组，组号 4 位；组内地址 5 位；字节号 2 位；所以区号＝20－4－2－5－2＝7 位。

主存地址：	区号	组号	块号	块内地址	字节	
	7	4	2	5	2	位

cache 地址：	组号	块号	块内地址	字节	
	4	2	5	2	位

15. 某机主存容量为 16MB，字长 32 位，cache 容量为 2K 字，确定下述情况下的主存和 cache 的地址格式。

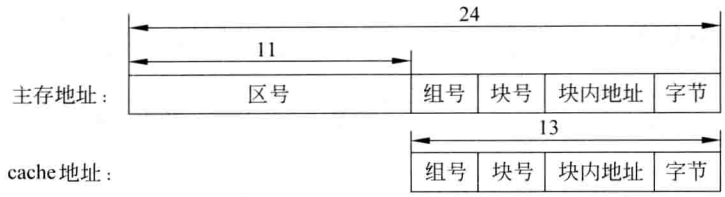
(1) 直接映像 cache，块长为 1 字。

(2) 直接映像 cache，块长为 8 字。

(3) 组相联映像 cache，块长为 1 字，组内 4 块。

(4) 组相联映像 cache，块长为 8 字，组内 4 块。

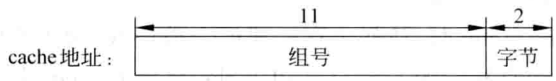
答：主存容量 16MB，地址 24 位；cache 容量 2K×4B，地址为 11+2=13 位。



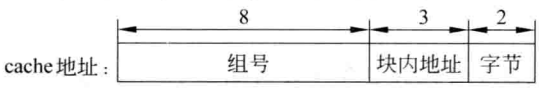
区号=24−13=11 位，对 4 种情况都适合。

组号+块号+块内地址总共 11 位，其分配随情况而变。

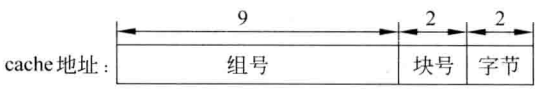
(1) 块长为 1 字，块内地址 0 位；直接映像，块号 0 位；组号=11−0−0=11 位。



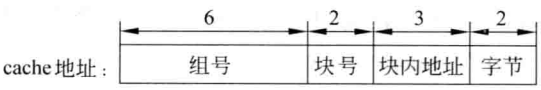
(2) 块长 8 字，块内地址 3 位；直接映像，块号 0 位；组号=11−0−3=8 位。



(3) 块长 1 字，块内地址 0 位；组相联映像，组内 4 块，块号 2 位；组号=11−2−0=9 位。



(4) 块长 8 字，块内地址 3 位；组相联映像，组内 4 块，块号 2 位；组号=11−2−3=6 位。



主存地址：在各个 cache 地址前面加上 11 位区号。不再一一画出。

16. 设有一台 RISC 机，CPU 采用流水线结构，只有 Load 和 Store 指令访问内存，它们占指令总数的 40%，不命中时的损失是 25 个时钟周期，存取数据的不命中率为 2%。取指令全部命中。问：

(1) cache 采用哈佛结构，如果取指令和存取数据全部命中时的 CPI 是 2.0，那么这台机器实际运行时的 CPI 是多少？(CPI 为运行一条指令的周期数。)

(2) 采用指令和数据合一的 cache，如果不存在 Load/Store 指令时的 CPI 为 2.0，那么当存在 Load/Store 指令时，机器实际运行时的 CPI 是多少？

答：假设采用的写入策略是：命中时写入 cache(不写入主存)，不命中时写入 cache 和主存。

(1) 存取数据不命中时增加的 CPI 为 40%×2%×25=0.2 时钟周期。实际运行时的 CPI=全部命中时的 CPI+增加的 CPI=2+0.2=2.2。

(2) 取指与取数不能同时进行。

$$
\begin{aligned}
\text{实际运行的 CPI} &= \text{取指的 CPI} + \text{取数的 CPI} \\
&= 2 + (2 + 2\% \times 25) \times 40\% \\
&= 3
\end{aligned}
$$

17. 假设机器周期为 10ns，cache 访问时间为 1 个周期，主存访问时间是 20 个周期。

(1) 设命中率为 95%，求平均访存时间。

(2) 如果 cache 容量增加一倍而使命中率提高到 97%，然而因此而使机器周期延长到 12ns，这样的改动方案是否值得采取？

答：(1) 机器的平均访存时间＝10＋(1－95%)×20×10＝20ns。

(2) 改动后的平均访存时间＝12＋(1－97%)×20×12＝19.2ns，减少了 0.8ns，但 cache 的成本上升了，是否值得还得综合考虑。

18. 假设有 3 种指令和数据合一的 cache：

(1) cache1：直接映像，块长为 1 字，指令命中率为 96%，数据命中率为 92%。

(2) cache2：直接映像，块长为 4 字，指令命中率为 98%，数据命中率为 95%。

(3) cache3：两路组相联映像，块长为 4 字，指令命中率为 98%，数据命中率为 96%。

命中时访问一次 cache 的时间为一个时钟周期。

运行时 50%指令存取一个数据。假设主存访问时间为“6 加上块的字数”个时钟周期。

求出上述 3 种情况执行一条指令的平均访存时间，并估计主存可能是一种什么样的结构。

答：(1) 平均访存时间＝平均取指时间＋平均取数时间

＝(1＋(1－96%)×(6＋1))＋(1＋(1－92%)×(6＋1)×50%)

＝1.28＋1.28＝2.56　时钟周期

(2) 平均访存时间＝(1＋(1－98%)×(6＋4))＋(1＋(1－95%)×(6＋4)×50%)

＝1.2＋1.25＝2.45　时钟周期

(3) 平均访存时间＝(1＋(1－98%)×(6＋4))＋(1＋(1－96%)×(6＋4)×50%)

＝1.2＋1.2＝2.4　时钟周期

符合第 1 种情况的主存为单体存储器，CPU 与存储器之间的数据传送宽度为 1 个字。

属于第 2 种和第 3 种情况的存储器可能是 4 体交叉存储器，所以在 6 个时钟周期内可从主存取出 4 个字，但由于 CPU(或 cache)与主存之间的数据传送宽度为 1 个字，因此传送 1 个字需要 1 个时钟周期。访问主存的时间为“6 加上块的字数”。

19. 计算机主存地址长度为 32 位，其 cache 容量为 16KB，字块长度为 4 个 32 位的字，分别考虑直接地址映像和 4 路组相联地址映像两种情况。要求：

(1) 画出 cache 结构逻辑图，给出主存各个字段的长度。

(2) 给出访问主存地址为 9A3CE5F8 的单元(按字节编址)时，该单元在 cache 的位置。

答：(1) cache 容量为 16KB，地址长度为 14 位，块内地址 2 位，字节地址 2 位。

直接映像：组号＝14－2－2＝10 位，区号＝32－14＝18 位。

4 路组相联映像：块号 2 位，组号＝14－2－2－2＝8 位，区号＝18 位。

cache 结构逻辑图分别如图 7.6 和图 7.7 所示。

(2) 直接映像：主存地址 9A3CE5F8＝$\underbrace{1001\ 1010\ 0011\ 1100\ 11}_{\text{区号(18位)}}\underbrace{10\ 0101\ 1111}_{\text{组号(10位)}}\ 1000$

该单元在 cache 中的位置为图 7.6 中的阴影部分，组号＝25FH。

4 路组相联映像：

主存地址 9A3CE5F8＝$\underbrace{1001\ 1010\ 0011\ 1100\ 11}_{\text{区号(18位)}}\underbrace{10\ 0101\ 11}_{\text{组号(8位)}}\underbrace{11}_{\substack{\text{块号}\\(2\text{位})}}\ 1000$

该单元在 cache 中的位置为图 7.7 中的一个阴影部分，组号＝$10\ 0101\ 11_2$＝97H。

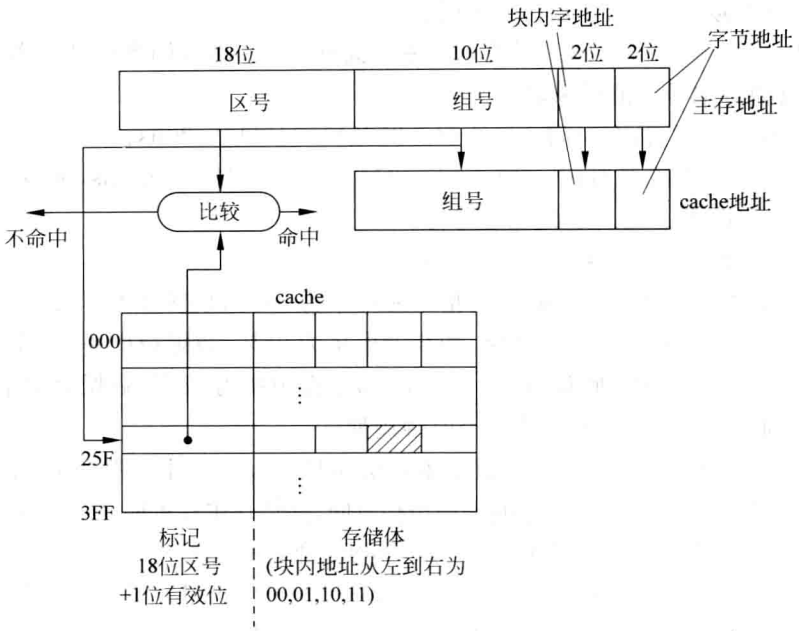


图 7.6　直接映像结构

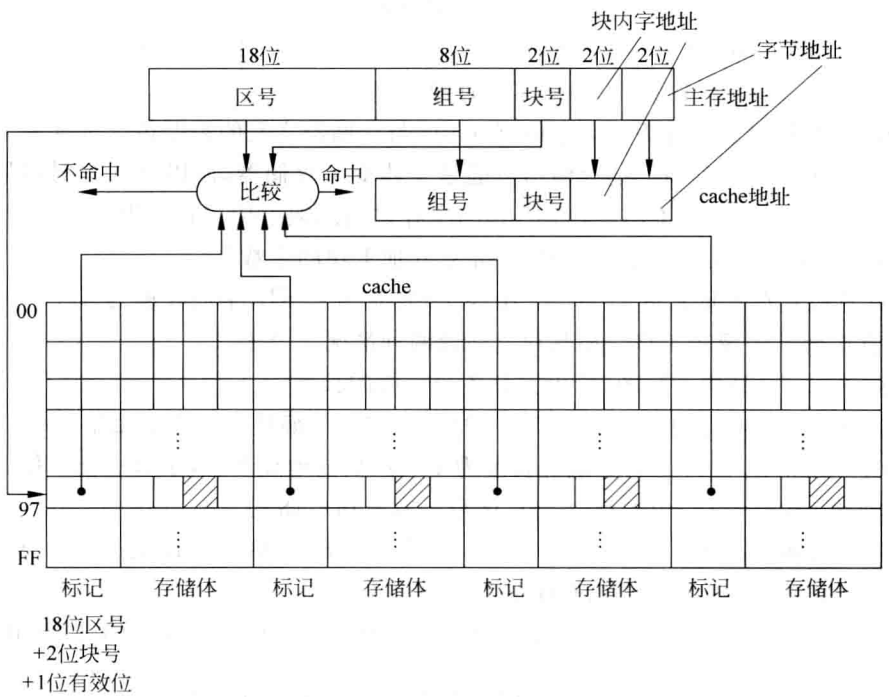


图 7.7　4 路组相联映像结构

有的计算机在标记中为每一个字设置 1 个有效位。

20. 比较 cache 三种地址映像(直接映像、两路组相联映像、全相联映像)的命中率。假设主存储器的容量为 8 块,cache 的容量为 4 块,按下列(1)、(2)次序循环访问主存各块,求

命中率(假设循环次数很多,忽略第 1 次循环对命中率的影响)。并讨论按下列(3)访存次序的两路组相联映像 cache 的命中率。

(1) 访存次序:0,4,0,4,0,4,…

(2) 访存次序:0,2,4,0,2,4,0,2,4,…

(3) 访存次序:0,1,2,3,4,0,1,2,3,4,0,1,2,3,4,…

答:在大多数情况下,全相联映像的命中率最高,组相联映像次之,直接映像最低。因为全相联映像允许主存的块映像到 cache 任一块中,只要 cache 不写满,访问总是命中的,而在组相联和直接映像中,即使 cache 未写满,也可能不命中,直接映像中主存的每一块只能映像到 cache 的特定块中,因此增加了不命中的可能性。然而具体实现时,由于访存次序及替换算法的限制有可能得到不同的结果。

cache 与主存储器之间的映像关系如图 7.8 所示。图中标以块号。全相联映像未画出。

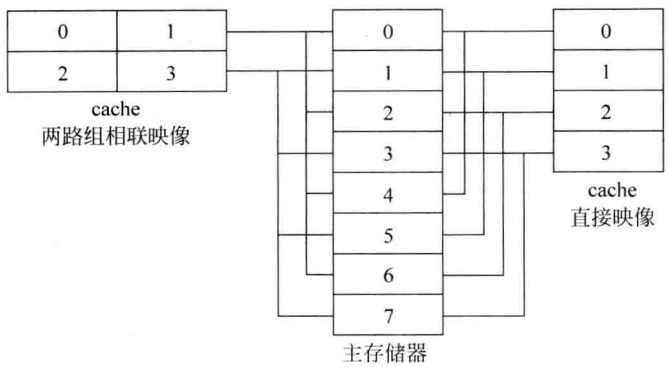


图 7.8 cache 与主存储器的映像

(1) 按 0,4,0,4,0,4,…次序访问,全相联和两路组相联的命中率各为 100%,直接映像由于 0 与 4 只能映像到 cache 的 0 块,所以命中率为 0。

(2) 按 0,2,4,0,2,4,0,2,4,…次序访问。

全相联映像:由于总共访问 3 块主存,而 cache 有 4 块,所以命中率为 100%。表 7.1 和表 7.2 列出直接映像和两路组相联映像 cache 中 4 个块的分配情况。

表 7.1 直接映像 cache 的 4 个块分配情况

访存次序	1	2	3	4	5	6	7	8	9	10	11	12	…
主存地址序列	0	2	4	0	2	4	0	2	4	0	2	4	
cache 第 0 块	0	0	4	0	0	4	0	0	4	0	0	4	
cache 第 1 块													
cache 第 2 块		2	2	2	2	2	2	2	2	2	2	2	
cache 第 3 块													
cache 命中情况	m	m	m	m	h	m	m	h	m	m	h	m	

注:表中 h 为命中,m 为不命中(或称为失效)。

命中率=1/3≈33%。

表 7.2　两路组相联映像 cache 的 4 个块分配情况(FIFO 替换算法)

访存次序	1	2	3	4	5	6	7	8	9	10	11	12	…
主存地址序列	0	2	4	0	2	4	0	2	4	0	2	4	
cache 第 0 块	0	0	4	4	2	2	0	0	4	4	2	2	
cache 第 1 块		2	2	0	0	4	4	2	2	0	0	4	
cache 第 2 块													
cache 第 3 块													
cache 命中情况	m	m	m	m	m	m	m	m	m	m	m	m	

注：全部不命中，命中率＝0。

表 7.2 得出 cache 命中率＝0 的结论，如果采用不同替换算法，如表 7.3 所示，则可提高命中率，这说明访存次序和替换算法都会影响命中率。

表 7.3　两路组相联映像 cache 的 4 个块分配情况(不同于表 7.2 的替换算法)

访存次序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
主存地址序列	0	2	4	0	2	4	0	2	4	0	2	4	0	2	4	
cach 第 0 块	0	0	0	0	2	2	2	2	4	4	4	4	0	0	0	
cach 第 1 块		2	4	4	4	4	0	0	0	0	2	2	2	2	4	
cach 第 2 块																
cach 第 3 块																
cach 命中情况	m	m	m	h	m	h	m	h	m	h	m	h	m	h	m	

注：命中率＝50%。

(3) 按 0,1,2,3,4,0,1,2,3,4,0,1,2,3,4…次序访问。

表 7.4 和表 7.5 是采用不同替换算法的两路组相联映像 cache 的命中率。

在表 7.4 中两路组相联映像从第 4 次开始到第 13 次为一循环，访问 10 次，命中 4 次，命中率＝4/10＝40%。

表 7.4　两路组相联映像 cache 的 4 个块分配情况(FIFO 替换算法)

访存次序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	…
主存地址序列	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	
cache 第 0 块	0	0	0	0	4	4	4	2	2	2	0	0	0	0	4	4	
cache 第 1 块			2	2	2	0	0	0	0	4	4	4	2	2	2	0	
cache 第 2 块		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
cache 第 3 块				3	3	3	3	3	3	3	3	3	3	3	3	3	
cache 命中情况	m	m	m	m	m	m	h	m	h	m	m	h	m	h	m	m	

表 7.5　两路组相联映像 cache 的 4 个块分配情况(不同于表 7.4 的替换算法)

访存次序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	…
主存地址序列	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	
cach 第 0 块	0	0	0	0	0	0	0	2	2	2	2	2	2	2	4	4	4	4	4	4	0	0	0	0	0	
cach 第 1 块			2	2	4	4	4	4	4	4	0	0	0	0	0	0	0	2	2	2	2	2	2	2	4	
cach 第 2 块		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
cach 第 3 块				3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	
cach 命中情况	m	m	m	m	m	h	h	m	h	h	m	h	h	h	m	h	h	m	h	h	m	h	h	h	m	

在表 7.5 中,两路组相联映像从第 5 次访问开始到第 24 次为一循环,访问 20 次,命中 14 次,命中率=14/20=70%。

表 7.3 和表 7.5 的替换原则是:在不命中时,尽量不替换最近将被访问的 cache 块,在一般情况下,这是难以预测的。

21. 某机主存容量为 8 个块,cache 容量为 4 个块,采用直接地址映像,如果开始时,cache 内容为空。今执行以下主存块地址序列:0、2、1、5、4、6、4、7、1、2、4、1、3、3、4。请列出每次访存后 cache 中各块分配情况、cache 命中情况及其命中率。

答:主存与 cache 块号的对应关系如表 7.6 所示。

表 7.6　主存所对应的 cache 块号

主存块号	0	2	1	5	4	6	4	7	1	2	4	1	3	3	4
cache 块号	0	2	1	1	0	2	0	3	1	2	0	1	3	3	0

每次访存后,cache 中各块分配情况如表 7.7 所示。

表 7.7　cache 中 4 个块的分配情况

访存次序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
主存地址序列	0	2	1	5	4	6	4	7	1	2	4	1	3	3	4
cache 第 0 块	0	0	0	0	4	4	4	4	4	4	4	4	4	4	4
cache 第 1 块			1	5	5	5	5	5	1	1	1	1	1	1	1
cache 第 2 块		2	2	2	2	6	6	6	6	2	2	2	2	2	2
cache 第 3 块								7	7	7	7	7	3	3	3
cache 命中情况	m	m	m	m	m	m	h	m	m	m	h	h	m	h	h

注:在访问次序为 7、11、12、14、15 时命中,命中率=5/15≈33%。

22. 今有直接映像 cache,容量为 16 字,由 4 字组成 1 块,开始时 cache 为空,每次访问主存取出 4 字,请回答每次访问后 cache 的内容以及命中情况。给出的访问主存的字地址序列如下:

1,4,8,11,20,17,18,19,9,16,4,43,19,60,61,62,17,63

答：cache 容量为 16 字，每块 4 字，所以共有 4 块。每次访存后 cache 块中的内容及命中情况如表 7.8 所示。

表 7.8　每次访存后 cache 块中的内容

访存次序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
主存地址序列	1	4	8	11	20	17	18	19	9	16	4	43	19	60	61	62	17	63
cache 第 0 块	0	0	0	0	0	16	16	16	16	16	16	16	16	16	16	16	16	16
	1	1	1	1	1	17	17	17	17	17	17	17	17	17	17	17	17	17
	2	2	2	2	2	18	18	18	18	18	18	18	18	18	18	18	18	18
	3	3	3	3	3	19	19	19	19	19	19	19	19	19	19	19	19	19
cache 第 1 块		4	4	4	20	20	20	20	20	20	4	4	4	4	4	4	4	4
		5	5	5	21	21	21	21	21	21	5	5	5	5	5	5	5	5
		6	6	6	22	22	22	22	22	22	6	6	6	6	6	6	6	6
		7	7	7	23	23	23	23	23	23	7	7	7	7	7	7	7	7
cache 第 2 块			8	8	8	8	8	8	8	8	8	40	40	40	40	40	40	40
			9	9	9	9	9	9	9	9	9	41	41	41	41	41	41	41
			10	10	10	10	10	10	10	10	10	42	42	42	42	42	42	42
			11	11	11	11	11	11	11	11	11	43	43	43	43	43	43	43
cache 第 3 块														60	60	60	60	60
														61	61	61	61	61
														62	62	62	62	62
														63	63	63	63	63
cache 命中情况	m	m	m	h	m	m	h	h	h	h	m	m	h	m	h	h	h	h

注：访问 18 次，命中 10 次，命中率＝10/18≈56%。

23. 设某机 cache 容量为 4K 字，采用直接地址映像，开始时 cache 为空，处理器访问地址序列为 0,1,2,…,4223(按字编址)，连续执行 10 遍，cache 的速度为主存的 10 倍。要求：

(1) 计算 cache 命中率(数据宽度为 1 字)。有 cache 与无 cache 的访问速度比值是多少？

(2) 计算 cache 命中率(数据宽度为双字)。与数据宽度为单字的情况相比，速度提高多少倍？

(3) 计算 cache 的命中率，已知主存储器改为两体交叉存储器，每个体的数据宽度为 1 个字，主存与 CPU 之间的数据传送宽度也为 1 个字。又问总访问时间有无变化？

答：(1) 处理器访问地址 0～4223，共 4224 字，超过 cache 容量 4224－4096＝128 字。第 1 遍访存时全部失效，在以后 9 遍中，每遍有 4096－128＝3968 字命中，128 字×2 次＝256 字不命中。

命中率＝命中次数/总访问次数＝(9×3968)/(10×4224)＝0.85＝85%

cache 的速度是主存的 10 倍，应该是指两者都取 1 字时的速度比。

假设得出 cache 不命中的信号后再访问主存，则有 cache 与无 cache 性能相比的倍数为

$$\frac{\text{主存存取时间}}{\text{有 cache 后平均存取时间}}=\frac{T_m}{T_c+(1-H)T_m}=\frac{1}{\frac{T_c}{T_m}+(1-H)}=\frac{1}{0.1+(1-H)}$$

$$=\frac{1}{0.1+(1-0.85)}=4$$

式中 T_c 为访问 cache 时间，T_m 为访问主存时间。

(2) 当数据宽度由单字变为双字时，访问 cache 与访问主存的次数均减半，所以命中率不变，仍为 85%，但总的访问时间减少了，是原访问时间的一半，即速度提高 1 倍。

(3) 如果 cache 块的字长为单字，则命中率不变，仍为 85%。但主存为两体交叉存储器，在连续顺序访问时，每隔 1/2 存取周期即可存取 1 个数据。假设 cache 的访问时间为 t，不命中时的访问时间 $=t+10t$。对单体存储器而言，从第 2 遍开始，1 遍的访存时间 $=(3968t+256(t+10t))=6784t$；对两体交叉存储器而言，1 遍访存时间 $=3968t+(t+10t)+255(t+5t)=5509t$，其中 $t+10t$ 是第一次访主存时间，$255(t+5t)$ 是随后访问 255 次主存时间。所以单体存储器的总访问时间 $=4224\times(t+10t)+6784t\times9=107520t$，两体交叉存储器的总访问时间 $=(t+10t)+4223\times(t+5t)+5509t\times9=74930t$。

24. 一个全相联 cache 有 6 块，每块有 8 个字，主存容量为 2^{16} 个字，cache 开始为空，cache 存取时间为 40ns，主存与 cache 间传送 8 个字需 1μs。

(1) 计算 cache 的标记域和块内地址域的大小。

(2) 一程序首先访问主存单元 20，21，…，45（十进制），然后重复访问主存单元 18，19，…，45 四次。假如 cache 没有命中时，将主存对应块一次全部读入 cache 中。计算 cache 的命中率。

(3) 计算上述程序总的存取时间。

答：(1) 块内地址域 3 位。cache 的标记域 $=16-3=13$ 位，再加上有效位等。

(2) 主存单元 20，21，…，45，按八进制表示时为 24，25，…，55，分布在 4 块中，共 26 个字，依次访问时，不命中 4 次，命中 22 次。然后访问 18，19，…，45，按八进制表示时为 22，23，…，55，也分布在上述的 4 个块中，共 28 个字，全部命中。

总共访问主存单元数 $=26+4\times28=138$ 次，不命中 4 次，命中 $138-4=134$ 次，命中率 $=134/138\approx97\%$。

(3) 总的存取时间 $=4\times1\mu s+134\times40ns\approx9.3\mu s$。

25. 设可供用户使用的主存容量为 100KB，而某用户的程序和数据所占的主存容量超过 100KB，但小于逻辑地址所表示的范围。问具有虚存与不具有虚存对用户有何影响？

答：对于非虚拟存储器，由用户负责分配存储器。对于虚拟存储器，用户可按逻辑地址编程，不必考虑实际的主存容量，所以编程方便得多，但是主存容量小会增加主存与辅存之间交换数据的次数，所以还是会影响解题速度。

26. 主存储器容量为 4MB，虚存容量为 1GB（1×10^9B），虚拟地址和物理地址各为多少位？根据寻址方式计算出来的有效地址是虚拟地址还是物理地址？如果页面大小为 4KB，页表长度是多少？

答：主存容量 $=4MB=2^{22}B$，虚存容量 $=1GB=2^{30}B$，所以虚拟地址 30 位，物理地址 22 位。

根据寻址方式计算出来的有效地址是虚拟地址。

如果页面大小为 4KB，采用页式管理时，页面的数量 $=\frac{1GB}{4KB}=\frac{2^{30}}{2^{12}}=2^{18}$，即页表长度为 2^{18}。

27. 设某虚存有如表 7.9 所示的快表放在相联存储器中，其容量为 8 个存储单元。问：按如下 3 个虚拟地址访问主存，主存的实际地址码各是多少（设地址码均为十六进制）？

	页号	页内地址
(1)	15	0324
(2)	7	0128
(3)	48	0516

表 7.9　快表

页号	本页在主存起始地址
33	42000
25	38000
7	96000
6	60000
4	40000
15	80000
5	50000
30	70000

答：先查快表（TLB），查得该逻辑页号在主存的起始位置后，与页内地址相加即得主存的实际地址。

(1) 主存地址＝80000＋0324＝80324H

(2) 主存地址＝96000＋0128＝96128H

(3) 逻辑页号为 48 的页面不在快表中，要到主存去查，如果在主存中查到，将该页的内容调入快表（若快表已满，则按替换算法撤销一行后调入），并计算主存地址。如果在主存中查不到，则需启动操作系统从辅存中调入相应的页表到主存中，然后再到主存去查。

28. 某程序对逻辑页面访问的序列为 $P_3P_4P_2P_6P_4P_3P_7P_4P_3P_6P_3P_4P_8P_4P_6$。(1)设主存容量为 3 个页面，求执行 FIFO 和 LRU 页面替换算法时各自的命中率（假设开始时主存为空）。(2)当主存容量增加到 4 个页面时，两种替换算法各自的命中率又是多少？

答：表 7.10 和表 7.11 分别表示主存为 3 个和 4 个页面时的调页情况，表中每一列表示页面淘汰次序，编号最大者（表 7.10 中的③和表 7.11 中的④）表示将被调出的页面。

(1) 主存页面为 3 时的调页情况（表 7.10）。

表 7.10　主存页面为 3 时的调页情况

页面请求		3	4	2	6	4	3	7	4	3	6	3	4	8	4	6
LRU	③	3	3	3	4	2	6	4	3	7	4	4	6	3	3	8
	②	/	4	4	2	6	4	3	7	4	3	6	3	4	8	4
	①	/	/	2	6	4	3	7	4	3	6	3	4	8	4	6
	命中	m	m	m	m	h	m	m	h	h	m	h	h	m	h	m
FIFO	③	3	3	3	4	4	2	6	3	3	7	4	4	6	3	8
	②	/	4	4	2	2	6	3	7	7	4	6	6	3	8	4
	①	/	/	2	6	6	3	7	4	4	6	3	3	8	4	6
	命中	m	m	m	m	h	m	m	m	h	m	m	h	m	m	m

注：采用 LRU 算法的命中率为 6÷15＝40%，采用 FIFO 算法的命中率为 3÷15＝20%。

(2) 主存页面为 4 时的调页情况(表 7.11)。

表 7.11　主存页面为 4 时的调页情况

页面请求		3	4	2	6	4	3	7	4	3	6	3	4	8	4	6
LRU	④	3	3	3	3	3	2	6	6	6	7	7	7	6	6	3
	③	/	4	4	4	2	6	4	3	7	4	4	6	3	3	8
	②	/	/	2	2	6	4	3	7	4	3	6	3	4	8	4
	①	/	/	/	6	4	3	7	4	3	6	3	4	8	4	6
	命中	m	m	m	m	h	h	m	h	h	h	h	h	m	h	h
FIFO	④	3	3	3	3	3	3	4	4	2	2	2	6	7	7	3
	③	/	4	4	4	4	4	2	2	6	6	6	7	3	3	4
	②	/	/	2	2	2	2	6	6	7	7	7	3	4	4	8
	①	/	/	/	6	6	6	7	7	3	3	3	4	8	8	6
	命中	m	m	m	m	h	h	m	h	m	h	h	m	m	h	m

注：采用 LRU 算法的命中率为 9÷15＝60%，采用 FIFO 算法的命中率为 6÷15＝40%。

(3) 补充其他列表方式：若按主存页面对应的逻辑页面列表，则当主存为 3 个页面时的变化情况如表 7.12 所示。

表 7.12　主存页面和对应的逻辑页面分配情况

页面请求			3	4	2	6	4	3	7	4	3	6	3	4	8	4	6
LRU	主存页面	①	3	3	3*	6	6	6*	7	7	7*	6	6	6*	8	8	8*
		②		4	4	4*	4	4	4*	4	4	4*	4*	4	4	4	4
		③			2	2	2*	3	3	3*	3	3	3	3	3*	3*	6
	命中		m	m	m	m	h	m	m	h	h	m	h	h	m	h	m
FIFO	主存页面	①	3*	3	3	6	6	6	6*	4	4	4	4*	4	8	8	8*
		②		4	4	4*	4	3	3	3*	3	6	6	6	6*	4	4
		③			2	2	2	2*	7	7	7	7*	3	3	3	3*	6
	命中		m	m	m	m	h	m	m	m	h	m	m	h	m	m	m

注：表中带 * 号的页面为即将被淘汰的页面。在采用 FIFO 算法的存储器 3 个页面中，最早调入的页面，也就是即将被淘汰的页面；在采用 LRU 算法时，如果下次访存不命中，则为即将被淘汰的页面。

在表中，LRU 的命中率为 6/15＝40%，最后在主存中的逻辑页面为 8、4、6。FIFO 的命中率为 3/15＝20%，最后在主存中的逻辑页面为 8、4、6。

其结果与表 7.10 的结论相同。

主存页面为 4 时的情况类似。

29. 上题的命中率很低，与实际情况相比，主要的差别是什么？

答：根据局部性原理，当某页调入主存后，一般将被访问很多次，而上题仅访问一次，所

以实际命中率是很高的。

30. 某计算机采用段页式虚拟存储器，已知虚拟地址有 32 位，按字编址：每个段最多可以有 1K 页，每页 16K 字，主存容量 64M 字。

(1) 求虚拟存储器容量。

(2) 说出逻辑地址和物理地址的格式。

(3) 求段表和页表长度。

答：(1) 虚存容量$=2^{32}$字$=4$G 字。

(2) $1K=2^{10}$，所以逻辑页号有 10 位。

$16K=2^{14}$，所以页内地址有 14 位。

虚拟地址的段号$=32-10-14=8$ 位。

逻辑地址由 8 位段号、10 位页号和 14 位页内地址构成。

逻辑地址：

段号	页号	页内地址
8	10	14

主存容量$=64M=2^{26}$，所以物理地址 26 位，其中页号 12 位，页内地址 14 位。

物理地址：

页号	页内地址
12	14

(3) 段号 8 位，段表长度$=2^8$ 行，每行指出页表的起始地址(物理地址 26 位)和有效位等。取 4B 以便于管理。

页号 10 位，每段页表长度$=2^{10}$行，每行指出物理页号(12 位)和有效位等，取 2B 以便于管理。所以段表长度$=2^8\times4B$。

页表总长度$\leqslant2^8\times2^{10}\times2B=2^{19}B$。

31. 某机采用页式虚拟存储器，每页存储 64 字，页表内容如表 7.13 所示，求下列虚拟地址对应的物理地址：

(1) 0000111100_2

(2) 0001001110_2

(3) 0101010101_2

表 7.13　页表

虚页号	实页号	装入位	虚页号	实页号	装入位
0000	01	1	0100	00	1
0001	11	1	0101		0
0010		0	⋮	⋮	⋮
0011	10	1			

答：每页存储 64 字，所以页内地址 6 位。根据页表可知虚页号 4 位，实页号 2 位，所以虚拟地址为 10 位，物理地址为 8 位。

(1) 虚页号为 0000，查表得物理页号 01，故物理地址为 01 111100。

(2) 虚页号为 0001，查表得物理页号 11，故物理地址为 11 001110。

(3) 虚页号为 0101，查表知尚未装入，此时应选择虚页号 0000、0001、0011 或 0100 中的

一页从页表中调出，而将相应的实页号 01、11、10 或 00 分配给虚页号 0101。在调出时，如果该页内容修改过，则应先将该页内容写回辅存。

32. 假设某计算机有 8 个虚页面要映射到 4 个实页面。某程序访问页面序列如下：

1,0,2,2,1,7,6,7,0,1,2,0,3,2

求每个虚页面访问请求之后存放在主存中的位置，假设初始时主存为空，采用 LRU 算法和 FIFO 算法，计算主存的命中率。

答：LRU 算法每次访问分配的实页号如表 7.14 所示。命中 6 次，命中率$=\frac{6}{14}\approx 0.43$。

表 7.14　每次访存后虚页面分配的实页面(LRU)

地址序列		1	0	2	2	1	7	6	7	0	1	2	0	3	2
物理页面	0	1	1*	1	1	1	1	1	1	1*	1	1	1	1*	1
	1		0	0	0	0*	0	6	6	6	6*	2	2	2	2
	2			2	2	2	2	2*	2	0	0	0	0	0	0
	3						7	7	7	7	7	7*	7	3	3
命中情况		m	m	m	h	h	m	m	h	m	h	m	h	m	h

* 为最近最少使用的页面，当有变化时标出。

FIFO 算法每次访问后分配的实页号如表 7.15 所示。命中 5 次，命中率$=\frac{5}{14}\approx 0.36$。

表 7.15　每次访存后虚页面分配的实页面(FIFO)

地址序列		1	0	2	2	1	7	6	7	0	1	2	0	3	2
物理页面	0	1	1*	1	1	1	1	6	6	6	6	6	6	6*	2
	1		0	0	0	0	0	0*	0	0	1	1	1	1	1*
	2			2	2	2	2	2	2	2	2*	2	0	0	0
	3						7	7	7	7	7	7	7*	3	3
命中情况		m	m	m	h	h	m	m	h	h	m	h	m	m	m

33. 某机有两级 cache，假设 CPU 访问存储器 1000 次，第 1 级 cache 有 50 次不命中，通过第 2 级 cache 后，仍有 10 次不命中，求第 1 级 cache 和第 2 级 cache 的命中率以及总命中率。

答：第 1 级 cache 命中率$=(1000-50)/1000=95\%$，第 2 级 cache 的命中率$=\frac{50-10}{50}=80\%$。

$$总命中率=(1000-10)/1000=99\%$$

或

$$\begin{aligned}总命中率&=1-总失效率=1-(1-95\%)(1-80\%)\\&=1-0.01=99\%\end{aligned}$$

34. 判断以下有关 cache 的叙述的正确性，如果错误，请纠正。

(1) cache 命中时，由硬件完成存/取操作，不命中时由操作系统处理。

(2) cache 与主存统一编址，即根据地址不同，访问主存或 cache。

(3) 在主机中，cache 可由 SRAM 或 DRAM 组成。

(4) cache 的容量一般与主存相差不多。

(5) cache 主要是为了扩充存储器容量。

(6) 在大多数程序执行过程中，指令的命中率小于数据的命中率。

(7) cache 中的内容是主存的一个小副本，因此在大多数计算机中，在任何时候 cache 与主存都能保持一致性。

答：每条叙述都不正确，分别纠正如下。

(1) cache 无论是否命中，都由硬件完成存取操作，除非被访问的内容(或页面)不在主存中，这时要发中断，由操作系统完成主存与辅助存储器之间数据的调动。由于 cache 的作用是弥补 CPU 与主存之间的速度差异，要求快速，所以在 CPU、cache 和主存之间的操作都是由硬件完成的。

(2) cache 是主存的一个小副本，CPU 访存时给出的地址是主存地址，如果该地址的副本在 cache 中，就访问 cache。

(3) cache 由 SRAM 组成。要求 cache 的存取速度比主存快，一般主存由 DRAM 组成，DRAM 除了存取周期比 SRAM 大以外，还要刷新，因此不适宜用于 cache。

(4) cache 的容量一般比主存小得多，因为 SRAM 的集成度比 DRAM 低，且价格贵。

(5) cache 主要用于提高 CPU 访存速度。

(6) 程序的局部性比数据的局部性强，通常指令的命中率大于数据的命中率。

(7) 由于写入 cache 的方案有多种，有的方案不同时写入 cache 和主存，因而造成了 cache 与其所映像的主存内容不一致。

35. cache 的写入方法有哪几种？

答：cache 中保存的字块是主存中相应字块的一个副本，如果要对该字块进行写操作，就会遇到如何保持 cache 与主存一致性问题。写入方法有以下 3 种：

(1) 暂时只向 cache 写入，并用修改标志加以注明，直到经过修改的字块从 cache 中被替换出来时才修改(写入)主存，这种方法称为“写回”法。其优点是某些中间结果可暂时保存在 cache 中，减少访问主存的次数，例如执行 a+b+c+d+…的运算，CPU 每次只能对两个数据进行运算，运算过程的中间结果可以保存在 cache 中，而不写入主存。其缺点是 cache 中的数据可能与其相应主存中的数据不一致，这样在取数命中时要检查 cache 的修改标志(当 DMA 外部设备访问存储器时)，以确定是从 cache 取数或者是从存储器取数。

(2) 同时写入 cache 和主存，这种方法称为“写通”法，或“通过式写”法。其优缺点正好与第(1)种方法相反。由于每次写入时无论是否命中，都要向主存写入，其结果相当于降低了数据的命中率。

(3) 只写入主存，不写入 cache。当外部设备向存储器写入时往往采取这种方法。当前计算机的外部设备与主存直接传送数据，当写入主存时，只写入主存，并置相应的 cache 有效位为“0”，以保证其他设备(例如 CPU)从主存取数时的正确性。

当有多个来源向主存写入数据时，在 cache 的控制器中往往设置有监听线路，监听地址线上的地址，如果在 cache 中有该地址的副本，就置相应的有效位为“0”，这种方法一般应用

于外部设备传送数据到主存时。

36. 判断以下有关主存储器的叙述的正确性，如有错误，请纠正。

(1) 现代计算机的主存储器速度由于半导体技术的进步而可以与 CPU 匹配。

(2) 主存储器中的 ROM 与微程序控制存储器的 ROM，由于它们都是 ROM，所以可以统一使用。

(3) CPU 执行的程序本身，必须先由应用程序从辅助存储器调入主存储器，然后才能执行。

(4) CPU 执行的程序本身，必须先从主存储器调入 cache，然后才能执行。

(5) 在 cache—主存层次中，为了提高 cache 命中率，可以采取的措施有：(a)增大 cache 容量，(b)增大主存容量，(c)程序编译优化，(d)选择适当的地址映像方式。

(6) CPU 加电产生的 reset(总清)信号，应将 cache 的全部有效位置“0”，并把主存储器中存储的所有内容都置“0”，以防程序运行出错。

答：以上叙述的每一条都有错误(或不全面)。说明如下：

(1) 半导体技术的进步，使 CPU 和主存的速度大幅度提高，但 CPU 对主存的速度要求仍超过了主存的能力，设置 cache 的目的就是为了缓冲 CPU 与主存之间的速度矛盾。

(2) 主存储器的 ROM 中存放的是操作系统的一部分，是由软件工作人员编写的。控制存储器中的 ROM 存放的是微程序，是由硬件工作人员编写的。两者的寻址方法、字长和工作性质都不相同，不能统一使用。

(3) CPU 执行的程序本身必须先从辅助存储器调入主存储器，然后才能执行。但调入的情况有两种：

① 虚拟存储器。由系统软件(操作系统)在硬件配合下自动调入，其过程对用户程序是透明的。

② 非虚拟存储器。由应用程序调入。

(4) CPU 执行的程序本身存放在主存储器中，是在执行过程中逐步由主存调入 cache 的，同时由于经常被调进调出，所以保存在 cache 中的程序也是不完整的。

(5) 增大主存容量不能提高 cache 的命中率，其他 3 种措施可以考虑。但 cache 容量和地址映像方式直接与硬件成本有关，需综合考虑。

(6) reset 信号将 cache 的全部有效位都置“0”，这样一来 cache 中的所有内容都作废了，可保证以后的操作不会再用到 cache 中的无用数据。reset 信号不将主存中的内容清除，程序员已注意到加电后主存内容的不确定性，因此启动机器后对每一个要使用的存储单元都是先写入，然后在使用时再读出。

37. 判断以下有关主存储器—辅助存储器层次的叙述的正确性，如有错误，请纠正。

(1) 辅助存储器的引入是为了解决主存速度的不足。

(2) 如果机内无辅助存储器，那么大于主存容量的程序就一定不能在该计算机中运行。

(3) 如果指令的地址码为 24 位(按字节编址)，那么立即可确定该机器的主存容量为 $2^{24}=16\text{MB}$。

(4) 页表一定存放在主存储器中。

(5) 页表的快表也可以存放在主存中。

(6) 页表的快表可采用直接、组相联或全相联查找。

(7) 虚拟存储器在每次访存时要将逻辑地址转换成物理地址，该操作是由软件完成的。

(8) 从辅助存储器调页表到主存的操作是由硬件完成的。

答：以上的叙述均有错误，纠正如下。

(1) 辅助存储器的引入主要是解决主存容量的不足。

(2) 如果机内无辅助存储器，那么大于主存容量的程序要分成若干块，每块的容量不大于主存容量，这样，程序就可运行了。

(3) 指令的地址码长度与主存的容量没有绝对的关系。一般主存的容量不超过地址码所能表示的范围，对于虚拟存储器地址码更长些。早期的一些小型机或微型机，其主存的容量已超过指令地址码所能表示的范围，这可通过字段寄存器或变址寄存器等将地址码增长。

(4) 当虚拟存储器容量很大时，其页表也很长，如果将其全部放在主存中，将占据主存很大的存储量，这是不合理的，为此可考虑将大部分页表存放在辅助存储器(例如磁盘)中，近期用到的页表才调入主存。

(5) 页表的快表存放在由逻辑电路(寄存器和门电路)组成的硬件中，因为设置快表的目的就是减少一次访存操作。

(6) 由于程序符合局部性原理，所以快表的容量可以很小，因此采用全相联查找。

(7) 虚拟存储器每次访存时，是通过查表将逻辑地址转换成物理地址的，由于每条指令执行时，取指和存取数据都要访存，因此必须由硬件完成该操作。除非查不到页表才让操作系统协助调页，并更新页表。

(8) 从辅助存储器调页表或调一页的内容到主存储器都是在硬件和软件(操作系统)配合下完成的。

38. 假设某高校学生入学考试总成绩已存入相联存储器中，每个学生有一个记录，有以下内容：准考证号、姓名、性别、年龄、志愿和总分。现要将总成绩大于 559 分的男生查找出来，请写出查找的方法和过程。

答：相联存储器的结构如图 7.9 所示，内有存储体、比较寄存器 CR、屏蔽寄存器 MR、字选择寄存器 WSR 和查找结果寄存器 SRR。根据本题的要求，存储体的每个记录还有一个比较线路，可与 CR 中的内容进行比较。

下面分两种情况进行讨论，图中虚框中的内容分别适用于情况 A 和情况 B，两者取 1。

情况 A：用 1 位区分性别，假设 1 代表男生，0 代表女生。在 MR 中，与性别和总分对应的位写上 1，其他为 0。在 CR 中，与性别对应的位写上 1，与总分对应的位写上 559。然后进行查找，将记录>××…×1×…×559 的学生表示在 SRR 中，即成绩大于 559 分的男生在 SRR 的相应位置 1。

情况 B：用 1 位表示性别，假设 1 代表男生，0 代表女生，分两次查找，第 1 次先找出男生，第 2 次从男生中找出成绩大于 559 分的学生。

第 1 次查找时，在 MR 的性别位写入 1，MR 的其他位都为 0。在 CR 中写上全 0。然后进行查找，查找的结果(性别为 1 的男生)表示在 SRR 中，然后将 SRR 的内容传送到 WSR。

第 2 次查找，在 MR 的分数段写入 11…1，其余各位为 0。在 CR 的分数段写入 559，其他位均为 0，然后进行查找，成绩大于 559 分的男生表示在 SRR 中。

第 1 次查找进行的是比较(相等)操作，第 2 次查找进行的是比较(大于)操作。

在一般计算机中完成这项工作是用程序逐条记录进行比较，当考生有 N 人时，重复进

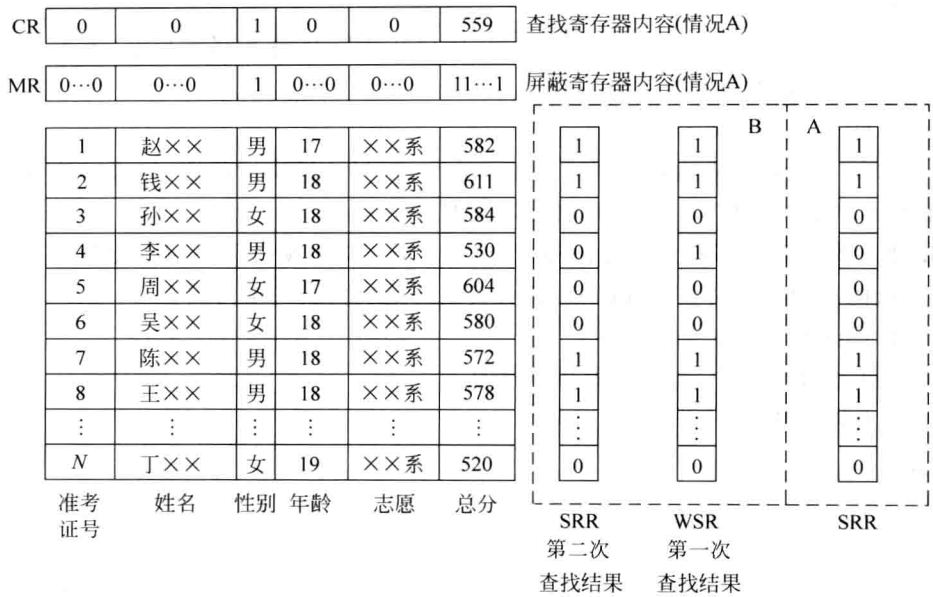


图 7.9 相联存储器

行 N 次，而在相联存储器只要进行 1 次(情况 A)或 2 次(情况 B)。但是相应于每条记录都要有一个比较器(硬件)才能满足本题的要求，而且比较(大于)操作要由多位加法器实现，很费器材。

39. 下面是有关存储保护的描述。请从本题列出的供选择答案中选择应填入□处的正确答案。

为了保护系统软件不被破坏以及在多道程序环境下防止一个用户破坏另一用户的程序而采取下列措施：

(1) 不准在用户程序中使用“设置系统状态”等指令。此类指令是[A]指令。

(2) 在段式管理存储器中设置[B]寄存器，可以防止用户访问不是分配给这个用户的存储区域。

(3) 在环保护的主存中，把系统程序和用户程序按其允许访问存储区的范围分层；假如规定内层级别高，那么系统程序应在[C]，用户程序应在[D]。内层[E]访问外层的存储区。

(4) 为了保护数据及程序不被破坏，在页式管理存储器中，可在页表内设置 R(读)、W(写)及[F]位。[F]位为 1，表示该页内存放的是程序代码。

供选择的答案：

A，B：①特权；②特殊；③上、下界；④系统。

C，D：①内层；②外层；③内层或外层。

E：①允许；②不允许。

F：①M(标志)；②P(保护)；③E(执行)。

答：A——①，在指令系统中设置特权指令，只供操作系统等系统软件使用，以防用户不慎使用这些指令而破坏机器的正常运行。设置状态寄存器，设置上、下界寄存器的内容等

指令均为特权指令。

B——③，某些机器设置上、下界寄存器，用户在访问存储器都要与上、下界进行比较，不允许超出上、下界所划定的范围。

C——①，D——②，E——①。

F——③，当代计算机的程序都是不允许修改的，当处于执行状态时，表示为程序，只准读，不准写。R(读)位和 W(写)位一般是指数据存储区。

40. 假设某计算机采用虚拟存储器，在执行一条指令时从存储器取指令与数据，请画出取指与取数的流程图以及使用 TLB 表和页表实现地址变换的结构图。

(1) 如果页表都设在主存储器。

(2) 页表设在辅存中，部分页表在主存中。

答：流程图如图 7.10 所示。

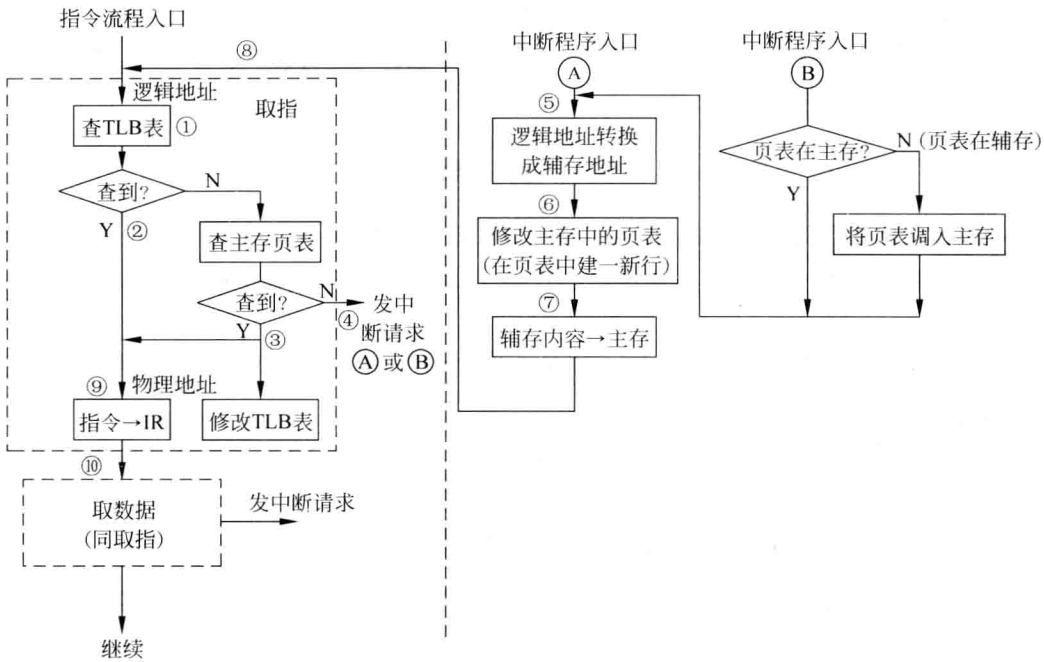


图 7.10　实现地址变换的流程图

执行指令时，从指令流程入口开始，首先将指令地址(PC)所表示的逻辑地址转换成物理地址，取指令到 IR。然后取数，接着继续完成该指令所需完成的其他操作(在流程图上不再表示)。

流程图的左边部分是用硬件实现的，右边是发中断后执行中断程序，这是软硬件配合实现的。

(1) 执行的流程如下(如果页表在主存中)。

① 根据指令逻辑地址中的逻辑页号查 TLB 表(快表)。

② 如果在 TLB 表中查到，则取出物理页号，与页内地址拼接构成物理地址访问主存。

③ 如果在 TLB 表中查不到，则查主存中的页表，如果查到，则取出物理页号，与页内地址拼接构成物理地址访问主存。并修改 TLB 表，添入 1 行，如果 TLB 表已满，则根据所采

用的替换算法淘汰 1 行后再添入。

④ 如果在主存的页表中也查不到，即本指令所在的页面尚未从辅存调入主存，于是发中断请求，由中断程序将该页的内容全部调入主存。

⑤ 进入中断程序入口Ⓐ，将逻辑地址转换成辅存地址（如果是磁盘机则转换成盘号、道号、扇区号）。

⑥ 修改主存页表（添入 1 行），如主存已满，则根据替换算法淘汰 1 行后再添入。

⑦ 辅存内容传送到主存（1 页）。

⑧ 中断程序返回原程序指令流程入口，重新开始执行原来的那条指令。重新执行①、②、③，此时一定可以从主存页表中查到所要的物理页号。

⑨ 指令→指令寄存器 IR。

⑩ 取数据。其过程与取指的过程相同，重复执行①～⑧，从中断程序返回时，也是返回到指令流程入口，因为进入中断程序后，本条指令在此之前所进行的操作已全部被取消，指令寄存器中已接受过新的指令。

（2）由于页表所在的页面可能不在主存中，所以进入中断程序入口Ⓑ，先判别一下页表是否在主存中，如不在，则将该页表从辅存调入主存，其余操作与（1）相同。

从流程图上看，（1）发中断后进入中断程序入口Ⓐ；（2）发中断后进入中断程序入口Ⓑ，进行一些处理后再进入Ⓐ。

使用 TLB 表和页表实现地址变换的结构图如图 7.11 所示。

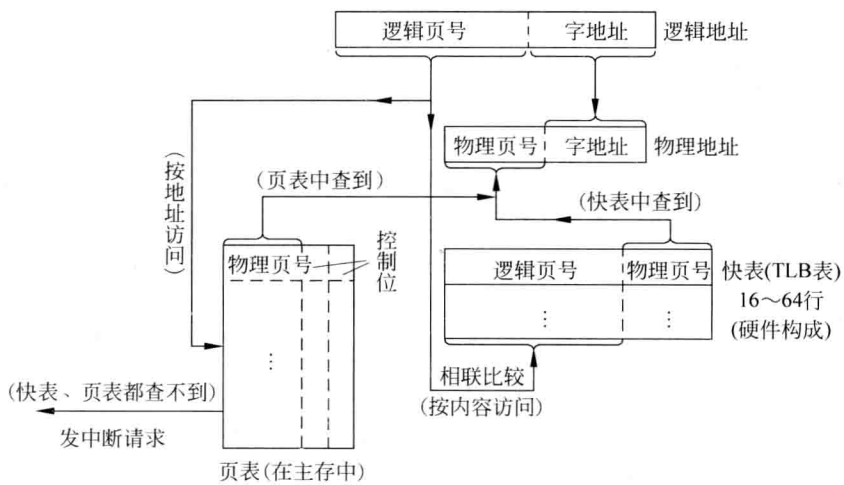


图 7.11　实现地址变换的结构图

逻辑地址中的逻辑页号要与快表（TLB 表）中的每行逻辑页号用硬件同时进行比较，其工作原理属于相联存储器范畴。

第 8 章　辅助存储器

1. 什么是辅助存储器？

答：辅助存储器又叫外存储器，简称辅存或外存。

计算机中的存储器分为主存储器和辅助存储器两大类。主存储器用来存放需立即使用的程序和数据，要求存取速度快，通常由半导体存储器构成。辅助存储器用于存放当前不需立即使用的信息，一旦需要，再和主存成批地交换数据。它作为主存的后备和补充，是主机的外部设备，常用的有磁盘机、磁带机和光盘机。其特点是容量大，存取时间长（与主存比较），而且不适宜于传送单个数据，通常在断电后仍能保持信息，是非易失性存储器。目前闪存得到广泛应用，其存储容量与上述辅存相差甚远，但便于携带，适用在某些设备间传递信息。

2. 磁表面存储器的主要技术指标有哪些？

答：磁表面存储器的主要技术指标有存储密度、存储容量、寻址时间和数据传输率等。

（1）存储密度。对于磁表面存储器，存储密度是指单位长度或单位面积磁层表面所存储的二进制信息量，对于磁盘存储器用道密度和位密度表示。

磁道指的是存储在介质表面上的信息的磁化轨迹，对于磁盘，磁道是磁盘表面上的许多同心圆，沿磁道半径方向单位长度的磁道数称为道密度。单位长度磁道所能记录的二进制信息的位数叫做位密度。

（2）存储容量。存储容量指磁表面存储器所能存储的二进制信息总量。一般以字节为单位。

对于磁盘存储器，如果盘片的直径大，道密度和位密度高，则存储容量亦大。目前发展的趋势是减小盘片尺寸，增大道密度和位密度，单片的容量不断提高。

磁盘存储器有格式化容量和非格式化容量两个指标。格式化容量指按照某种特定的记录格式所能存储信息的总量，也就是用户真正可以使用的容量。非格式化容量是磁记录表面可以利用的磁化单元总数。将磁盘存储器用于计算机系统中，必须首先进行格式化操作，然后才能供用户记录信息，格式化容量一般约为非格式化容量的 60%～70%。

（3）寻址时间。磁盘存储器寻址时间包括两部分：一是移动磁头寻找目标磁道所需的找道时间 t_s；二是找到磁道以后，磁头等待所需要读写的区段旋转到它的下方所需要的等待时间 t_w。由于寻找相邻磁道和从最外面磁道找到最里面磁道所需的时间不同，磁头等待不同区段所花的时间也不同，因此，取它们的平均值，称作平均寻址时间 T_a，它由平均找道时间 t_{s_a} 和平均等待时间 t_{w_a} 组成：

$$T_a = t_{s_a} + t_{w_a} = \frac{t_{smax} + t_{smin}}{2} + \frac{t_{wmax} + t_{wmin}}{2}$$

t_{w_a} 等于盘片转半圈的时间，与磁盘的转速有关。

磁带存储器采取顺序存取方式，不需要寻找磁道，但需要考虑磁头寻找记录区的等待时间，实际上磁头不动，磁带移动，所以寻址时间指的是磁带空转到磁头应访问记录区所在位置的时间。

(4) 数据传输率。磁表面存储器在单位时间内与主机之间传送数据的位数或字节数叫数据传输率 D_r。为确保主机与磁表面存储器之间传输信息不丢失，传输率与存储设备和主机接口逻辑两者有关。从设备方面考虑，传输率等于记录密度 D 和记录介质的运动速度 V 的乘积。从主机接口逻辑考虑，应有足够快的传送速度接收/发送信息，以便主机与辅存之间的传输正确无误。

(5) 其他还有磁盘旋转速度、磁带移动速度和误码率等。磁盘、磁带等的运动速度会影响寻址时间(等待时间)和数据传输率。

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

3. 填充题

主存储器是随机存储器，可以根据给定的地址读出其内容，当执行一条指令时，从中取出[A]和存取[B]。CPU 不能对[C]存储器随机访问一个单元。当 CPU 执行程序时，若其所要取出的指令或数据不在主存储器，则要发出中断请求信号，当操作系统对存储器采用[D]管理时，将上述指令或数据所在页面整个从辅存调入主存。

答：A——指令；B——数据；C——辅助；D——页式。

4. 填充题

磁表面存储器有直接存取存储器和顺序存取存储器两种。磁盘存储器是[A]，磁带存储器是[B]。

磁表面存储器是依靠[C]和[D]的相对运动而写入和读出信息的。要求存储介质的剩磁大，以保存信息，所以由[E]材料组成；要求磁头的剩磁小，有利于区分信息的两种状态(1 和 0)，所以由[F]材料组成。

答：A——直接存取存储器；B——顺序存取存储器；C——磁头；D——存储介质；E——硬磁；F——软磁。

说明：磁盘机是直接存取存储器，指的是磁头可直接移动到所要求的磁道位置进行存取。磁带机是顺序存储器，指的是磁头要等待磁带顺序移动到读写位置才能进行存取。开始读写后，两者都是顺序按位进行存取的。

5. 磁表面存储器读出线上的噪音是如何产生的？

答：在磁带存储器中，磁带走带时可能有抖动、扭动与偏移，因此写入与读出的磁道轨迹可能会产生变化。采用多道磁带时，为了使各道磁头同时写入或读出，将多个磁头组装在一起(称为组合磁头)，磁头间的信号会相互干扰，而且各道磁头的电磁特性不完全一致，使得读出信号的波形和读出时间也不完全相同，从而影响到读出信息的检测。磁盘存储器由于磁头移动位置的误差，相邻磁道间的相互干扰以及相邻位磁通翻转变化区的重叠都会产生噪音。

由于磁头读出的信号很小，而逻辑电路的信号(由 0→1，或由 1→0)变化很大，因此会产生线间干扰，再加上电源线与地线上的电压瞬间变化，影响了读出线上的信号噪音比。经常采取以下措施：磁头的读出线尽早连到读出放大器；读出线的布线要合理，距离大信号(电压或电流)线远些；本身应采取一些屏蔽措施；读出放大器用选通信号控制，滤掉非读出时的干扰信号，而这些干扰信号的幅度可能大大地超过读出信号。

6. 什么是脉冲拥挤效应？

答：当磁表面存储器存储介质的位密度提高时，由于磁层的连续性，两个相邻存储单元，甚至更远的存储单元间存储的信号相互重叠，从而使读出的脉冲波形受前后两位、甚至更远位的信息的影响，读出信号畸变，产生幅度衰减、峰值偏移和基线偏移现象，这就是脉冲拥挤效应。

7. 什么是磁记录方式？

答：磁记录方式是一种编码方法，指的是按照某种规则将一串二进制数字信息记录到存储介质的磁层上。这是通过在磁头的写入线圈中通以正负方向的电流实现的，在理想情况下，磁层表面剩磁的磁化方向与线圈中所通过的电流是一致的。读出时，依靠记录信息的磁层与磁头的相对运动，在磁头的读出线圈中产生感应电压。图 8.1 是不归零制(NRZ)、见 1 就翻的不归零制(NRZ1)和改进调频制(MFM)3 种记录方式的磁化强度和读出信号波形。实际上读出信号的波形极差，读出信号的边沿变化慢，干扰信号又很多，因此需要用选通信号将有用的读出信号选择出来，该选通信号就是同步信号。图 8.1 上的读出信号是经过选择和整形后产生的波形。

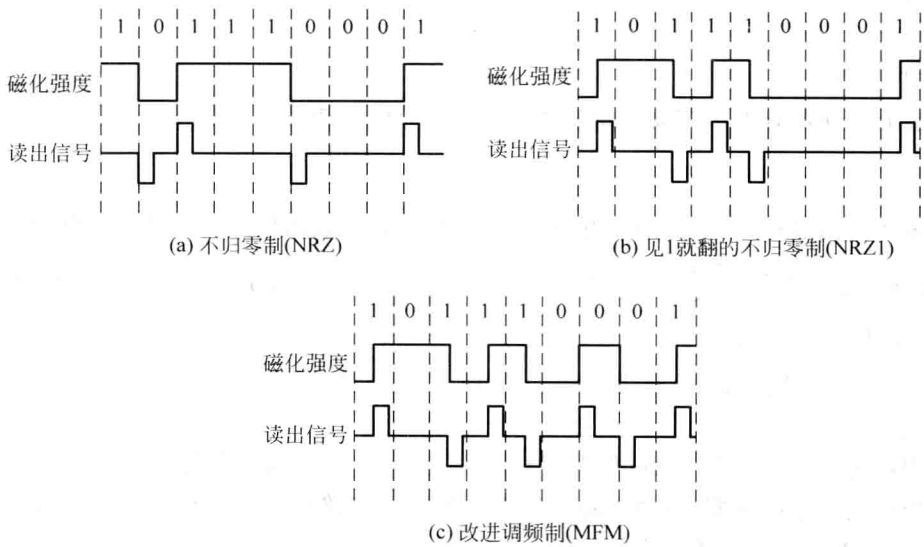


图 8.1　3 种记录方式的磁化强度和读出信号波形

通常用到的记录方式还有调相制(PM)、调频制(FM)、改进调频制(MFM)和游程长度受限码(RLL)等。

8. 今从磁盘存储器读出一串数据 10111001，假设采用不归零制和调频制，如何将该数据送到寄存器中？

答：(1) 不归零制。将读出信号的正脉冲与负脉冲放大，并分离成 2 个信号 e_1 和 e_0，其输出电平符合集成电路要求，且为负脉冲，分别送到触发器的置“1”端和置“0”端。波形图及逻辑图见图 8.2，假设读出触发器的初始状态为“1”。读出触发器输出送到具有左移功能的移位寄存器。CP 为移位时钟。

(2) 调频制。波形图如图 8.2 所示。分析读出信号，在前半周期总有一个读出脉冲(正

脉冲或负脉冲)。在后半周期,当读出"1"时,有一脉冲(正脉冲或负脉冲);在读出"0"时,则没有脉冲。今将读出信号经差分放大器将所有脉冲都转换成负脉冲,分别取出其前半周期的脉冲 e_0 和后半周期的脉冲 e_1,分别送读出放大器的置"0"端和置"1"端,读出放大器的输出送到有左移功能的移位寄存器,CP 为移位时钟,位于各周期结束前,上升沿起作用。逻辑图见图 8.3。

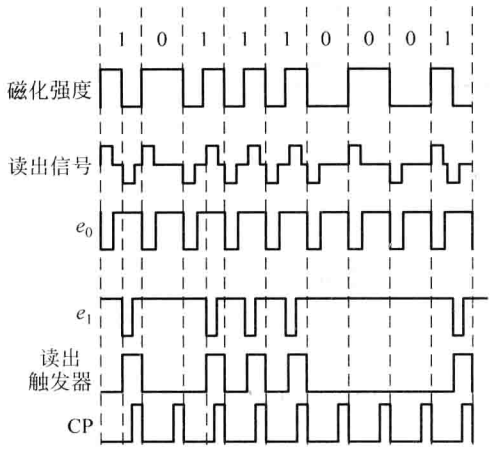


图 8.2　调频制读出时序

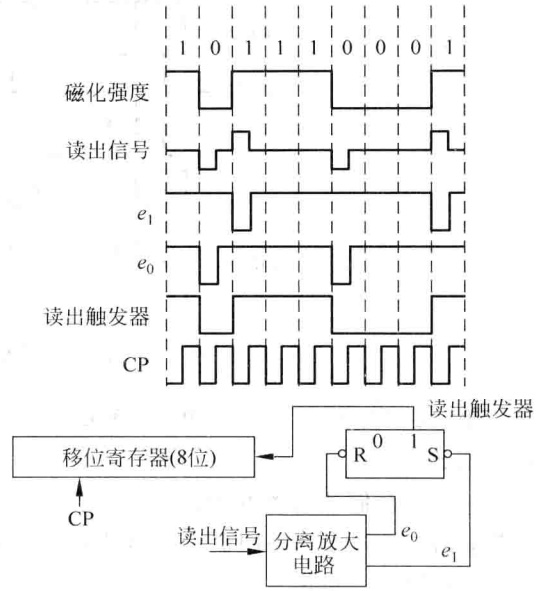


图 8.3　不归零制读出时序与逻辑图

9. 请解释自同步能力和编码效率。并说出下列 6 种磁记录方式的自同步能力和编码效率(RZ,NRZ,NRZ1,PM,FM,MFM)。

答:自同步能力是指从单个磁道读出的脉冲序列中提取同步时钟脉冲的难易程度。

$$\text{自同步能力 } R = \frac{\text{最小磁化翻转间隔}}{\text{最大磁化翻转间隔}}$$

编码效率是指每次磁层状态翻转所存储的数据信息位的多少。

$$\text{编码效率 } \eta = \frac{\text{位密度}}{\text{最大磁化翻转密度}}$$

表 8.1 列出 6 种磁记录方式的自同步能力和编码效率。

RZ 的自同步能力和编码效率都比较高,但是要实现未磁化状态困难,影响使用。

表 8.1　自同步能力和编码效率　　%

记录方式	RZ	NRZ	NRZ1	PM	FM	MFM
自同步能力	100	0	0	50	50	50
编码效率	100	100	100	50	50	100

根据定义,PM 的自同步能力为 50%,然而该记录方式在每个时钟周期的中间都产生磁化翻转,提取同步信号还是比较容易的。

10. 设写入代码为 110101001，试画出 RZ、NRZ、NRZ1、PM、FM 和 MFM 的写电流波形。

答：各记录方式写电流波形如图 8.4 所示。

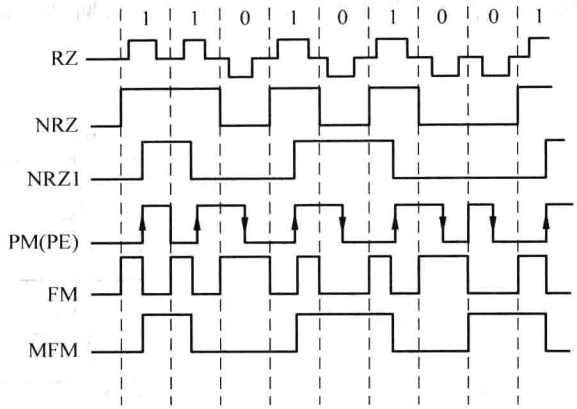


图 8.4　各种记录方式的写电流波形

注意：在 MFM 记录方式中，顺序记录单个 0 和连续多个 0 时，其波形不相同，可提高自同步能力。

11. 磁盘存储器有哪几种同步方式？

答：(1) 外同步。采用盘组的存储器可专用一个盘面，用以写入同步信号。

(2) 自同步。见第 9 题。

12. 填充题

(1) 在磁盘存储器的速度指标中，[A]和[B]与磁盘的转速有关。

(2) 在磁盘存储器中采用浮动磁头工作方式的是[C]存储器，采用磁头与盘片接触方式进行读写的是[D]存储器。

(3) 通常盘片上每一磁道的容量都相等，所以半径大的磁道比半径小的磁道的[E]小。

(4) 磁盘存储器是按[F]串行读写的，一次读写过程至少读写一个[G]。

答：(1) A——等待时间，B——传输率。

因为平均等待时间是磁盘转半圈的时间。在位密度相等的情况下，转速和传输率成正比。

(2) C——硬盘，D——软盘。

软盘的工作环境比硬盘差(硬盘中常用的温盘是一个密封装置)，所以软盘的转速、存储密度等都比硬盘低很多。也容易损坏。当前软盘实际上已被淘汰。

(3) E——位密度。

由于内圈磁道的位密度大，所以内圈磁道的半径不能太小。否则虽然增加了磁道数，而盘片的容量却可能下降。

(4) F——位，G——扇区。

通常一个盘片上只有一个磁头(固定头磁盘除外)，所以是按位串行顺序读写的。由于磁盘工作时每次寻道和等待的时间较长，而且存在程序访问存储器的局部性原理，所以每次

传送一批数据，其最小单位为一个扇区。

13. 填充题

(1) 在磁盘阵列 RAID 中，适合于传送大量数据的是 $\boxed{A}$ 和 $\boxed{B}$。适合于传送大量或少量数据的是 $\boxed{C}$ 和 $\boxed{D}$。

(2) 为了减少主存与磁盘存储器之间的速度差异，而在磁盘存储器中设置 $\boxed{E}$，它是由 $\boxed{F}$ 或 $\boxed{G}$ 组成的。

答：(1) A——RAID2，B——RAID3，C——RAID4，D——RAID5

RAID2 和 RAID3 是采取位交错存储的磁盘阵列。RAID4 和 RAID5 是采取块交错存储的磁盘阵列。

在 RAID 中，位交错就是将一个数据字中的各位分别存储在不同的磁盘上，位交错以同步方式进行读写，最小访问数据单位是每个磁盘的最小读写单位(例如扇区)乘以磁盘数，适合于传送大批量数据。块交错是以数据块为单位，将连续的数据块分别存储在不同的磁盘上，最小访问数据单位是一个磁盘的最小访问单位(例如扇区)，所以也适合于传送少量数据。

(2) E——磁盘 cache，F——SRAM，G——DRAM

磁盘 cache 使用在主存储器—辅助存储器层次上，由于磁盘存储器速度慢，所以可用 DRAM 组成磁盘 cache(当然使用 SRAM 也是可以的)，而且主存—辅存间传送数据还有操作系统参与，其速度比 CPU 与主存之间的 cache 更慢很多。CPU 与主存之间的 cache 是用 SRAM 组成的，传送数据是完全用硬件实现的。

14. 列举三个以上提高磁盘存储器记录密度的途径。

答：(1) 提高磁记录介质的质量。

介质的磁层尽量薄，具有高的矫顽力和剩磁，磁层缺陷少。

(2) 改进磁头。

传统的感应式磁头采用电磁感应原理读出记录信号，而记录密度的提高使得相邻磁化单元之间产生干扰，导致磁头读出线圈中通过的磁通量减少。而且驱动器的小型化使磁盘转动的线速度下降，这会导致磁头读出线圈中磁通量变化率的降低。这两个因素作用的结果是磁头读出信号幅值的急剧下降，从而不能正确读出记录信号。

利用磁致电阻效应(Magneto Resistive，MR)磁头能在高密度记录的情况下读出信号。MR 磁头是专用于读出的磁头，即它不能完成写入工作，但它具有高的输出灵敏度和与磁盘转速无关的输出特性，所以需要与专用的写入磁头配合使用。

(3) 提高磁头定位精度。

在可移动磁头的磁盘驱动器中，驱动磁头沿盘面径向位置移动以寻找目标磁道位置的机构叫磁头定位驱动机构。它是一个带有调节磁头移动速度和反馈磁头位置的闭环自动控制系统，前者是为了缩短找道时间，后者是为了提高道密度。可采用光栅位置检测方法以提高磁头定位精度。

(4) 采用温彻斯特磁盘(简称温盘)。

温盘是一种密封组合式的硬磁盘，将磁头、盘片、电机的驱动电路甚至读写电路等制成一个不可随意拆卸的整体，它的防尘性能好、可靠性高，从而能提高存储密度和容量。

(5) 减少浮动磁头的浮动高度。

(6) 采用编码效率高的磁记录方式。

(7) 采用垂直磁记录技术。

(8) 采用 PRML 技术。

任何数字磁记录都可看作由一串等距或不等距的磁化翻转所构成，如果两次相邻的极性相反的磁化翻转相隔很近，那么读出信号脉冲将产生部分重叠，而影响彼此读出的信号幅度，并使峰点产生偏移，这种现象称为脉冲拥挤效应或符号间干扰(码间串扰)，这会影响记录密度。PRML 技术的基本思想是一个符号的读出信号并不局限在一个符号的持续期间内，而延伸到前后若干个符号的持续期间内，但符号间的干扰不是随机的，而是有规律的，可通过计算机观察进行控制和消除，因此可以提高记录密度。实现时当然要解决很多技术问题。

15. 光磁软盘与磁光盘有何本质不同？

答：光磁软盘是采用激光伺服定位的软磁盘。

磁光盘是利用激光在磁性薄膜上产生热磁效应而存储信息的可擦写光盘。

根据磁记录原理可以知道，在一定温度下如果在磁记录介质的表面上加一个强度低于该介质矫顽力的磁场，则不会发生磁通翻转，也就不能记录信息。但介质的矫顽力可随温度而变，假如我们能设法控制温度，降低介质的矫顽力，使其低于外加弱磁场强度，则将发生磁通翻转。磁光存储就是根据这一原理存储信息的。它利用激光照射磁性薄膜，被照射处温度上升，矫顽力下降，在外加磁场 HR 的作用下发生磁通翻转，使该处的磁化方向与外加磁场 HR 一致。

16. 光盘存储器有哪几种类型？其依据的读写原理是什么？

答：光盘存储器有以下 3 种：

(1) 只读型 CD-ROM，盘中内容由生产厂家写入。

(2) 写一次型 WORM，用户可写入一次，读出多次。

(3) 可擦写型，用户可多次写入读出。

写入的设备称为刻录机，CD-R 刻录机是一次性写入的刻录机，CD-RW 刻录机用于多次擦写。

光盘所依据的读写原理有形变、相变和磁光(M-O)存储。

17. 设某磁盘存储器的平均找道时间为 t_s 秒，转速为每分钟 r 转，每磁道容量为 N 个字，每信息块为 n 个字。试推导读写一个信息块所需总时间 t_B 的计算公式。

答：转速为每分钟 r 转，转一圈的时间 $=\frac{60}{r}$s

$$
\begin{aligned}
t_B &= \text{平均找道时间} + \text{平均等待时间} + \text{传输 } n \text{ 个字的时间} \\
&= t_s + \frac{1}{2}\left(\frac{60}{r}\right) + \frac{n}{N}\left(\frac{60}{r}\right) \\
&= t_s + \left(\frac{1}{2} + \frac{n}{N}\right)\frac{60}{r}\text{s}
\end{aligned}
$$

18. 设磁盘组有 11 个盘片，每片有两个记录面；存储区域内直径 2.36 英寸，外直径 5.00 英寸；道密度为 1250tpi，内层位密度 52 400bpi，转速为 2400r/min。问：(tpi 表示每英寸磁道数，bpi 表示每英寸位数)

(1) 共有多少个存储面可用？

(2) 共有多少柱面?

(3) 每道存储多少字节? 盘组总存储容量是多少?

(4) 数据传输率是多少?

(5) 每扇区存储 2KB 数据,在寻址命令中如何表示磁盘地址?

(6) 如果某文件长度超过了一个磁道的容量,应将它记录在同一个存储面上,还是记录在同一个柱面上?

答:(1) 11 个盘片,共 22 面,最外面的 2 个面不用,则可用的存储面为 20。

(2) 外半径=5/2=2.5 英寸,内半径=2.36/2=1.18 英寸,柱面数=道数=(2.5−1.18)×1250=1650。

(3) 每道存储量$=2\pi r\times$位密度$=2\pi\times1.18\times52\,400$b

≈48.56KB

盘组总容量=20×面存储量=20×48.56KB×1650≈1.6GB

(4) 数据传输率=道存储量×转数/秒$=48.56\text{KB}\times\dfrac{2400}{60}$

≈1.94MB/s

(5) 20 个盘面,用 5 位表示。1650 道,用 11 位表示。扇区数=48.56KB/2KB≈24 个扇区,用 5 位表示。磁盘地址如下(共 21 位):

盘面号	道号	扇区号	
5	11	5	位

(6) 应该记录在同一柱面上。与记录在同一盘面上相比较,可以减少一次寻道时间。

19. 某软盘盘面有 80 个磁道,每磁道 15 扇区,每扇区存储 512B。已知磁盘的转速为 360r/min,今在一个磁道上写入 4096B 数据,平均需要多少时间? 最长时间是多少?(假设找道时间为 10~40ms)。

答:磁盘转速$=360\text{r/min}=\dfrac{360}{60}\text{r/s}=6\text{r/s}$,转一圈的时间=1/6s。4096B 数据所占的扇区数$=\dfrac{4096}{512}=8$。

写入的平均时间=平均找道时间+平均等待时间+传输数据时间

$$=\frac{10+40}{2}\text{ms}+\frac{1}{2}\times\frac{1}{6}\text{s}+\frac{8}{15}\times\frac{1}{6}\text{s}$$

$$=25\text{ms}+83\text{ms}+89\text{ms}=197\text{ms}$$

写入的最长时间=40ms+1/6s+89ms

$\approx40+167+89=296$ms

20. 一个磁盘组,其最外磁道半径为 R,最内磁道半径为 r,可用盘面数为 N,磁道密度为 d,位密度为 S,每个扇区的记录格式总长为 L 位,其中数据区长度为 K 位,每个磁道可划分 M 个扇区。问:

(1) 最内磁道半径为多少时,磁盘组容量达到最大值?

(2) 最大圆柱面数为多少?

(3) 该磁盘组总容量是多少字节?

答：(1) 内磁道半径为 r，当 $2\pi rS=ML$ 时，磁盘组容量达到最大值，因为当 r 更小时，为了保证最内磁道容量为 ML，存储密度将超过 S，这是不允许的。如果增大 r 值，而保持 S 不变，将减少盘组存储容量。因此

$$r = ML/2\pi S$$

(2) 最大圆柱面数 C =(外磁道半径－内磁道半径)×磁道密度

$$=(R-r)d=(R-ML/2\pi S)d$$

(3) 总容量＝盘面数×圆柱面数×扇区数×扇区数据

$$=N\times C\times M\times K\text{b}=N\times C\times M\times K/8\text{B}$$

实际在磁盘组上存储的容量(包括格式化内容)$=N\times C\times M\times L$b。

21. 某磁盘组的性能参数如下：转速 6000r/min，每道容量 80KB，平均寻道时间 10ms，假定操作系统采用两种方式访问该磁盘组，第 1 种方式每次读取 4KB，第 2 种方式每次读取 8KB。已知操作系统访问磁盘的额外开销为每次 1ms。

(1) 计算该磁盘组的数据传输率和平均等待时间。

(2) 如果读取一个 16KB 的磁盘文件，操作系统按第 1 种方式或第 2 种方式工作，所需的时间各是多少？

答：(1) 6000r/min＝100r/s(每秒钟转速)

传输率＝80KB×100/s＝8KB/ms

$$\text{平均等待时间}=\text{转半圈的时间}=\frac{1}{2}\times\frac{1}{100\text{r/s}}=5\text{ms}$$

(2) 假设 16KB 在一个磁道上。

第一种方式所需的时间

$$=\text{平均寻道时间}+\text{平均等待时间}+(\text{操作系统开销}+\text{读取 4KB 时间})\times\frac{16\text{KB}}{4\text{KB}}$$

$$=10\text{ms}+5\text{ms}+\left(1\text{ms}+\frac{4\text{KB}}{8\text{KB}}\text{ms}\right)\times 4$$

$$=21\text{ms}$$

第二种方式所需的时间

$$=\text{平均寻道时间}+\text{平均等待时间}+(\text{操作系统开销}+\text{读取 8KB 时间})\times\frac{16\text{KB}}{8\text{KB}}$$

$$=10\text{ms}+5\text{ms}+\left(1\text{ms}+\frac{8\text{KB}}{8\text{KB}}\text{ms}\right)\times 2$$

$$=19\text{ms}$$

如果 16KB 不在一个磁道上，则需要增加寻道时间和等待时间。

22. 设磁盘的平均寻道时间为 20ms，转速为 5400r/min，每个磁道的容量为 64KB，采用磁盘 cache，容量足够大，并忽略 cache 的访问时间。求：

(1) 在磁盘 cache 不命中时，访问 64KB 数据的时间。

(2) 在磁盘 cache 命中时，访问 64KB 数据的时间。

(3) 当磁盘 cache 的命中率为 80%时，访问 64KB 数据的平均时间。

(4) 如果磁盘 cache 访问时间为 50ns，不能忽略。每次存取 2 字节，命中率为 80%，又假设磁盘与主存之间的数据传输率为 20MB/s 和 80MB/s 两种情况。请考虑上述因素，写

出访问 64KB 数据的时间。

答：假设 64KB 数据在一个磁道上。

转速 5400r/min，转 1 圈时间$=\frac{60}{5400}=\frac{1}{90}$s

(1) 不命中时的访问时间＝平均寻道时间＋平均等待时间＋读取数据时间

$$=20\text{ms}+\frac{1}{2}\cdot\frac{1}{90}\text{s}+\frac{1}{90}\text{s}\approx 36.7\text{ms}$$

(2) 命中时访问时间≈0ms。

因为命中时不需要寻道时间和等待时间，又假设 cache 访问时间忽略不计。实际上此时的传输率等于磁盘与主存接口之间的传输率。

(3) 当命中率为 80%时，平均访问一个磁道数据时间＝0＋(1－0.8)×36.7ms＝7.34ms，命中时实际上要考虑磁盘与主存接口之间的传输率。

(4) 50ns cache，访问 cache 的频率为$\frac{1}{50\text{ns}}\times 2\text{B}=40\text{MB/s}$。数据在磁盘存储器与主存之间传输。

① 磁盘与主存之间的数据传输率为 20MB/s，低于 cache 的传输率，cache 命中时，按 20MB/s 传输。

$$\text{访问 64KB 的平均时间}=0.8\times\frac{64\text{KB}}{20\text{MB/s}}+(1-0.8)\times 36.7\text{ms}=0.8\times 3.2\text{ms}+7.34\text{ms}$$
$$=9.9\text{ms}$$

② 磁盘与主存之间的数据传输率为 80MB/s，高于 cache 的传输率，cache 命中时，按 40MB/s 传输。

$$\text{访问 64KB 的平均时间}=0.8\times\frac{64\text{KB}}{40\text{MB/s}}+(1-0.8)\times 36.7\text{ms}=8.62\text{ms}$$

23. 一磁带机有 9 个磁道，带长 700m，带速 2m/s，每个数据块 1KB，块间间隔 14mm。若磁带机的数据传输率为 128KB/s，试求：

(1) 记录位密度。

(2) 若带首尾各空 2m，求此带最大有效存储容量。

答：(1) 带速 2m/s，传输率为 128KB/s，位密度$=\frac{128\text{KB}}{2\text{m}}=64\text{KB/m}=64\text{B/mm}$，1 个磁道的记录位密度$=\frac{64\text{B/mm}}{8}=64\text{b/mm}$。

(2) 有效存储长度＝700－2×2＝696m。

$$\text{每个数据块长度}=\text{数据长度}+\text{块间间隔}=\frac{1\text{KB}}{64\text{B/mm}}+14\text{mm}$$
$$=16\text{mm}+14\text{mm}=30\text{mm}$$

$$\text{磁带最大存储容量}=\text{块数}\times\text{每块容量}=\frac{696\text{m}}{30\text{mm}}\times 1\text{KB}$$
$$=23.2\text{MB}$$

说明：9 个磁道中有 8 个磁道存储 1B 数据，1 个磁道为该字节的奇偶校验位，9 个磁道并行读出。每个数据块 1KB，所以在每一磁道上存储的是 1Kb。数据传输率是指整个磁带

的传输率，对每一磁道而言，其传输率是 128Kb/s。

24. 填充题(题中(1)～(4)中的 A、B、C 和 D 是各不相关的)

(1) 一个完整的磁盘存储器由[A]三部分组成。其中[B]又称磁盘机，是独立于主机的一个完整设备，[C]通常是插在主机总线插槽中的一块电路板，[D]是存储信息的介质。

(2) 磁盘驱动器的定位驱动系统实现[A]；主轴系统的作用是[B]；数据控制系统的作用是控制数据的写入和读出，包括[C]等。

(3) 磁盘控制器有两个方向的接口，一是与[A]的接口，与主机总线打交道，控制辅存与主机之间交换数据；另一个方向是与[B]的接口，根据主机命令控制磁盘驱动器操作。

(4) 光盘的读写头即光学头比硬盘的磁头[A]，光盘的定位速度[B]，即找道时间[C]，光盘写入时盘片需旋转 3 圈，以分别实现[D]，故光盘的速度[E]硬盘。

答：(1) A——磁盘驱动器、磁盘控制器和盘片，B——磁盘驱动器，C——磁盘控制器，D——盘片。

如果驱动器中包括盘片，则磁盘存储器由磁盘驱动器和磁盘控制器两部分组成。

(2) A——快速精确的磁头定位，B——带动盘片按额定转速稳定地旋转，C——寻址操作(寻道和旋转等待)、磁头选择(当有多个盘片时选择盘面)、写电流控制(根据写入数据和记录方式产生写入电流)、读出放大(磁头读出线圈中读出的电压甚小，必须进行放大)、数据与同步信号的分离(读出的信号中包括数据与同步信号时)。

(3) A——主机，B——磁盘驱动器。

与主机的接口中包括程序中断控制和直接存储器存取(DMA)控制部分，主机与磁盘之间的数据传送是由 DMA 控制部分直接进行控制的。

(4) A——大且重，B——低，C——长，D——擦除、写入和检验，E——低于。

25. 启停式磁带机和数据流磁带机有何区别？

答：数据是以数据块形式存储在磁带上的，数据块之间有间隙。

启停式磁带机在读写完一个数据块后，可以在间隙段停止磁带移动和再启动磁带移动，接着读/写下一数据块。为此设置了摆杆等磁带缓冲机构、真空积带箱以及机械张力控制部件等。

数据流磁带机在数据块之间不快速停止磁带移动和不再启动磁带移动，从而简化了磁带的启停机构，用电子控制来部分代替机械控制，不仅降低了成本，而且提高了可靠性。

26. 判断以下叙述的正确性，如有错误，请予纠正。

(1) 硬盘、软盘、磁带和光盘都与读写头接触读写。

(2) 磁盘盘片的旋转速度和磁带盘的转速都是恒定的。

(3) 在采用多道磁带结构的磁带机中，所有磁道之间都是并行读出的。

(4) 在磁表面存储器中主要采取奇偶校验或海明校验。

(5) 在多层次存储系统中，磁带机主要应用于主存—辅存层次中。

(6) 磁盘存储器依靠磁盘的旋转定位磁道，依靠磁臂的移动定位扇区。外圈磁道与内圈磁道的容量不同。扇区是最小访问单元。

答：每条叙述都有错误，纠正如下：

(1) 在软盘与磁带机中，读写头与存储介质接触读写。

硬盘是依靠磁盘旋转产生的气流将磁头浮动起来进行读写的，在磁盘不旋转时，如果磁头在盘片区是会与盘片相碰的，这是不允许的，因此在磁盘停止旋转前，磁臂将磁头移出盘片区。

光盘的读写头与存储介质保持一段小距离。

(2) 磁盘盘片的旋转速度是恒定的，因此内圈磁道的线速度低于外圈磁道。

磁带读写时要保持带与头之间的相对速度恒定，所以带盘的旋转速度根据带盘上绕的磁带多少而予以调整。

(3) 磁带机除了并行读出外，还有串行逐道记录的，其读写顺序类似于磁盘。以 4 道磁带机为例，4 个磁道的排列次序如图 8.5 所示，记录信息时先在第 0 道上从磁带首端 BOT 记到磁带末端 EOT；然后在第 1 道上反向记录，从 EOT 到 BOT；第 2 道又从 BOT 到 EOT，最后第 3 道从 EOT 回到 BOT。读出信息时也是这个顺序。这种方式称作蛇形(serpentine)记录。9 道1/4 英寸数据流磁带的记录方式相同，偶数磁道是从 BOT 到 EOT，奇数磁道从 EOT 到 BOT，依次首尾相接。

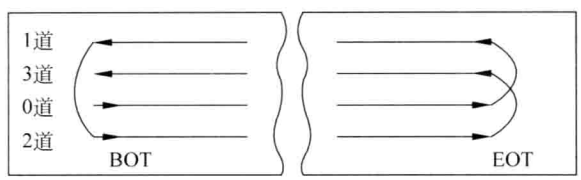


图 8.5　4 道 1/4 英寸磁带蛇形串行记录方式

(4) 磁表面存储器中主要采用 CRC 校验码，或再以奇偶校验码辅助校验。

(5) 磁带机存取速度太慢，与主存不匹配，因此在主存—辅存层次中一般用磁盘存储器。磁带机主要用作磁盘的后备存储器。

(6) 磁盘存储器依靠磁臂移动而带动磁头定位磁道，依靠盘片的旋转定位扇区。

一般外圈磁道的容量与内圈磁道的容量相等，因此内圈的位密度大。少数磁盘保持位密度相等而使内、外圈磁道的容量不同。

扇区是最小的访问单元。

27. 什么是固态盘？

答：采用半导体存储介质和磁盘接口的存储器称为固态盘(Solid State Disk，SSD)，SSD 的存储介质可以是 DRAM 或 NVRAM(非易失性随机存储器，例如闪存)。

DRAM 的缺点是断电时会丢失所存信息，因此需要在固态盘中安装电池，而且数据丢失后不能再恢复。磁盘上的数据虽然也可能丢失，但通过专业的数据恢复公司大多能挽救回来。

当采用闪存作为存储介质时，无须电源即可保存信息，提高了可靠性和可用性，一般通过 USB 接口与计算机连接。

28. 磁带机在大型计算机系统中起什么作用？

答：在大型计算机系统中采用磁带库存储大量后备数据。例如 Orcale(甲骨文)公司可提供模块化磁带库系统，具有高度的机械化和自动化能力来调用所选择的磁带。该系统可

答：每条叙述都有错误，纠正如下：

(1) 在软盘与磁带机中，读写头与存储介质接触读写。

硬盘是依靠磁盘旋转产生的气流将磁头浮动起来进行读写的，在磁盘不旋转时，如果磁头在盘片区是会与盘片相碰的，这是不允许的，因此在磁盘停止旋转前，磁臂将磁头移出盘片区。

光盘的读写头与存储介质保持一段小距离。

(2) 磁盘盘片的旋转速度是恒定的，因此内圈磁道的线速度低于外圈磁道。

磁带读写时要保持带与头之间的相对速度恒定，所以带盘的旋转速度根据带盘上绕的磁带多少而予以调整。

(3) 磁带机除了并行读出外，还有串行逐道记录的，其读写顺序类似于磁盘。以 4 道磁带机为例，4 个磁道的排列次序如图 8.5 所示，记录信息时先在第 0 道上从磁带首端 BOT 记到磁带末端 EOT；然后在第 1 道上反向记录，从 EOT 到 BOT；第 2 道又从 BOT 到 EOT，最后第 3 道从 EOT 回到 BOT。读出信息时也是这个顺序。这种方式称作蛇形(serpentine)记录。9 道1/4 英寸数据流磁带的记录方式相同，偶数磁道是从 BOT 到 EOT，奇数磁道从 EOT 到 BOT，依次首尾相接。

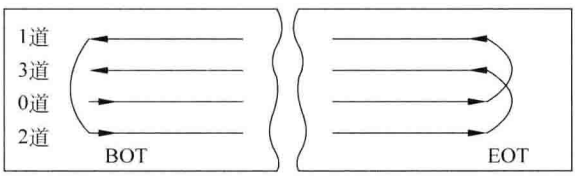


图 8.5　4 道 1/4 英寸磁带蛇形串行记录方式

(4) 磁表面存储器中主要采用 CRC 校验码，或再以奇偶校验码辅助校验。

(5) 磁带机存取速度太慢，与主存不匹配，因此在主存—辅存层次中一般用磁盘存储器。磁带机主要用作磁盘的后备存储器。

(6) 磁盘存储器依靠磁臂移动而带动磁头定位磁道，依靠盘片的旋转定位扇区。

一般外圈磁道的容量与内圈磁道的容量相等，因此内圈的位密度大。少数磁盘保持位密度相等而使内、外圈磁道的容量不同。

扇区是最小的访问单元。

27. 什么是固态盘？

答：采用半导体存储介质和磁盘接口的存储器称为固态盘(Solid State Disk，SSD)，SSD 的存储介质可以是 DRAM 或 NVRAM(非易失性随机存储器，例如闪存)。

DRAM 的缺点是断电时会丢失所存信息，因此需要在固态盘中安装电池，而且数据丢失后不能再恢复。磁盘上的数据虽然也可能丢失，但通过专业的数据恢复公司大多能挽救回来。

当采用闪存作为存储介质时，无须电源即可保存信息，提高了可靠性和可用性，一般通过 USB 接口与计算机连接。

28. 磁带机在大型计算机系统中起什么作用？

答：在大型计算机系统中采用磁带库存储大量后备数据。例如 Orcale(甲骨文)公司可提供模块化磁带库系统，具有高度的机械化和自动化能力来调用所选择的磁带。该系统可

第9章　输入输出(I/O)设备

1. 什么是键盘？它是怎样工作的？

答：键盘是目前应用最普遍的一种输入设备。

键盘是由一组排列成阵列形式的按键开关组成的，每按下一个键，产生一个相应的字符代码(每个按键的位置码)，并发中断，在中断程序的控制下，将它转换成 ASCII 码或其他码，送主存储器，为了观察输入代码，键盘往往与显示器联合工作，将输入的字符显示在屏幕上，显示器为输出设备。

目前常用的标准键盘有 101 个键，它除了提供通常的 ASCII 字符以外，还有多个功能键(由软件系统定义功能)、光标控制键(上、下、左、右移动等)与编辑键(插入或消去字符)等。

ASCII 是 128 个字符组成的字符集。其中编码值 0～31 不对应任何可印刷(或称有字形)字符，通常称它们为控制字符，用于通信中的通信控制或对计算机设备的功能控制。编码值为 20H 的是空格(或间隔)字符 SP。编码值为 7FH 的是删除控制 DEL 码。其余的 94 个字符称为可印刷字符，有人把空格也计入可印刷字符，则称有 95 个可印刷字符。

2. 哪些设备输入位置坐标？哪些设备输入相对坐标？

答：输入位置坐标的有光笔、图形板和画笔(或游动标)、触摸屏等。

光笔用于显示屏，光笔头部装有一个透镜系统，可拾取显示器屏幕上的坐标，与屏幕上的光标配合，可使光标跟踪光笔移动，在屏幕上画出图形或修改图形，经操作员确认后可存入计算机。

图形板和画笔的作用相当于上述的显示屏和光笔，主要用于输入工程图。用游动标取代画笔，可取得更为精确的坐标。

触摸屏是透明的，可安装在任何一种显示器屏幕的外面(表面)。使用时，显示器屏幕上根据实际应用的需要显示出用户所需控制的项目或查询的内容(标题)，供用户选择。用户只要用手指(或其他东西)点一下所选择的项目(或标题)，即可由触摸屏将此信息送到计算机中，用户通过与触摸屏的直接接触(点击或移动)，向计算机输入的是接触点的坐标位置，以后的工作就由程序去执行了。

输入相对坐标的有鼠标器、跟踪球和操作杆。它们必须和显示器的光标配合。计算机先要给定光标的初始位置，然后用读取的相对位移移动光标。鼠标器、跟踪球和操作杆操作容易、制作简单而且造价低，但定位精度都比较差。

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，样子像老鼠，由此得名。跟踪球(trackball)是用手指或掌心推动的金属球体，操作杆(joystick，也叫游戏棒)是能用于前后左右移动的金属杆，它们读取坐标的方式与鼠标器类似，都是得到相对位移量，用相对坐标定位。

3. 请解释光学字符识别(OCR)。

答：在美国，由于普遍使用打字机，因此识别的对象多是印刷体英文、数字和某些符号。

大部分扫描仪都配备了英文 OCR 软件，用户可方便地将英文文本送入计算机。

在我国，OCR 的识别对象有印刷体汉字和手写体汉字，其识别技术所面临的问题是西方文字无法比拟的。

OCR 系统是多项技术结合的产物，识别技术是其核心内容，还包括图形文本的扫描输入、光电信号变换、电信号的数字化处理、版面分析与理解、字的切分处理以及输入信息载体（页）的自动传送技术等。

4. 阴极射线管（CRT）显示器是怎样工作的？它与平板显示器有何不同？

答： CRT 最早用于电视接收机，后来应用于计算机系统中，用来显示字符、图形和图像等。现今已基本被淘汰。

CRT 是一个电真空器件，由电子枪、偏转装置和荧光屏构成。

电子枪是阴极射线管的主要组成部分，包括灯丝、阴极、栅极、加速阳极和聚焦极。CRT 在加电以后，灯丝发热，热量辐射到阴极，阴极受热便发射电子，电子束射到荧光屏上形成光点，由光点组成图像。

为了获得高质量的图像，电子束应满足以下要求：

（1）电子束应有足够的强度和速度，而且强度可以控制。电子束的强度由控制栅极（简称栅极）来控制，而栅极由外加的亮度信号控制，根据信号的强弱，控制每个光点的亮暗，即电子束的强弱。阳极对电子束加速，以保证电子束有足够的能量轰击荧光屏，使荧光屏发出相当强的光。

（2）射到荧光屏上的电子束要足够细，以保证图像清晰可辨。从阴极发出的电子，初速不同，方向不同，并且电子之间存在排斥力，所以在运动过程中将逐渐扩散，射到荧光屏是模糊暗淡的斑点。聚焦装置将电子束聚集得很细。

偏转装置的作用就是控制电子束的运动方向，它是由套在 CRT 尾部的偏转线圈实现的。偏转线圈接扫描电路，在偏转线圈产生的磁场的作用下，电子束能运动到荧光屏的任何位置。

在 CRT 玻璃屏的内壁涂有荧光粉，它将电子束的动能转换成光能，从而显示出光点。

以上介绍的是单色（黑白）显示器。

彩色 CRT 与单色 CRT 的原理基本相似，但是彩色 CRT 有 3 个电子枪，分别对应红、绿、蓝三基色的信号强弱。在荧光屏内壁涂有彩色荧光粉，按三基色迭加原理形成彩色图像。

CRT 荧光屏的尺寸用对角线的长度表示。

平板显示器一般是指显示器的厚度小于显示屏幕对角线 1/4 长度的显示器件，有以下几种：

（1）液晶显示器（LCD）。液晶是液态晶体的简称，液晶显示器具有工作电压低、功耗小、体轻薄，适于 LSI 驱动，显示色彩优良等特点，它本身不发光，但能够调制外照光实现信息显示，因此使用时需要背光源。现广泛应用于笔记本电脑、PDA、手机、摄像机和数码相机等。

（2）彩色等离子体显示器（PDP）。利用惰性气体在一定电压作用下产生气体放电现象的特性，气体放电产生紫外线，紫外线激发荧光粉，荧光粉发射可见光，使用三基色荧光粉产生红、绿、蓝三色，迭加形成彩色图像。其结构好比把数十万至数百万个气体微型荧光灯按

一定排列方式设置在两块平板玻璃之间。

PDP 的应用市场主要集中在 20～30 英寸的显示器市场和 37 英寸以上的大屏幕电视机市场。

(3) 其他还有场发射显示(FED)、电致发光显示(ELD)等。

以上叙述的显示器除了 LCD 以外，自身都能发光。

5. 请简介液晶显示器(LCD)。

答：液晶是液态晶体的简称，具有液体的流动性和分子排列有序的晶体特性，但本身不发光，使用时需要背光源，液晶分子通过外照光的调制形成像素。液晶显示器具有低电压、微功耗、体轻薄、色彩优良等特点，易于实现大画面显示。

背光源有多种可供使用。冷阴极荧光灯曾是使用最广泛的背光源，它发出的光在由导光板等组成的背光模组作用下，通过漫射和反射而成为亮度均匀并垂直射出的面光源，背光模组的设计和制造都很复杂，价格相对来说较贵，LCD 显示器出现问题时，可能出自背光模组。

近年来广泛应用 LED 作为背光源，LED 背光源分为直射式和侧射式，直射式是指 LED 灯均匀分布在整块液晶屏幕背面，侧射式将 LED 分布在屏幕的外侧线处。

LED 灯与冷阴极荧光灯比较，具有工作电压低、安全性强、环保性好、使用寿命长等特点。使用 LED 的显示器具有色彩鲜艳、亮度调节范围大、画面稳定和视觉效果好等特点。

6. 从供选择的答案中选出正确答案填入□中。

显示器的控制逻辑和存储逻辑一般都在[A]中。终端是由[B]组成的一套独立完整的[C]设备。它能够完成显示控制与存储，键盘管理及通信控制等功能，还可完成简单的编辑操作。供选择的答案：

A：①主机内部；②显示器内部；③主机与显示器的接口板中。

B：①显示器和控制逻辑；②键盘和显示器；③键盘和控制逻辑。

C：①输入；②输出；③输入输出。

答：A——③，B——②，C——③

终端是由键盘和显示器组成的输入输出设备，其中键盘是输入设备，显示器是输出设备。

7. 解释下述与计算机的显示器有关的概念：图形，图像，分辨率，灰度级，刷新，刷新频率，帧存储器，视频随机存储器，显示存储器，光栅扫描，随机扫描。

答：(1) 图形。图形是指用计算机表示现实世界的各种事物，并且形象逼真地加以显示。如动画设计、花布图案设计、地图的显示等平面图，飞机、汽车、建筑物的造型设计等立体图。需要有深浅和颜色。它所用的技术包括点、线、面、体等平面和立体图的表示和生成。同时，由于要在平面上显示立体图，还要研究阴影的产生，隐藏线、隐藏面的消除技术以及光照方向与颜色的模拟等技术。

(2) 图像。数字图像处理的对象多半来自客观世界，例如由摄像机摄取下来存入计算机的数字图像(遥感图像、医用图像等)。可能充满噪音，图像很不清晰。由于摄取的位置随机，图像可能发生畸变。图像处理的任务是去除噪音，恢复原形，使图像清晰。

在显示屏幕上，图形和图像都是由称作像素的光点组成的。数字图像是逐点存储的，所

以需要占用的主存空间很大。在计算机中表示图形，只需存储绘图命令和坐标点，没有必要存储每个像素点。例如，用两点式表示一条直线 $L(x_0, y_0; x_1, y_1)$，计算机中只存储这两个坐标，坐标点到图形像素点的转换由计算机或显示设备自动完成。

(3) 分辨率和灰度级。分辨率(resolution)指的是显示设备所能表示的像素个数。像素越密，分辨率越高，图像越清晰。分辨率取决于荧光粉的粒度、屏的尺寸和电子束的聚焦能力。

例如，16 英寸分辨率为 1024×768 的显示器表示其水平方向有 1024 个像素点，垂直方向有 768 个像素点。

灰度级(gray level)指的是所显示像素点的亮暗差别，在彩色显示器中则表现为颜色的不同。灰度级越多，图像层次越清楚逼真。如果用 8 位表示一个像素，则可表示 $2^8=256$ 级灰度或颜色，存储 1024×768 分辨率显示器 1 幅图像的容量为 1024×768×8 位。字符显示器只用“0”，“1”两级灰度就可以表示字符的有无，这种只有两级灰度的显示器称为单色显示器或黑白显示器。具有多种颜色的显示器称为彩色显示器，具有多种灰度级的黑白显示器称为多灰度级黑白显示器。

(4) 刷新和刷新频率。CRT 器件的发光是由电子束射在荧光粉上引起的。电子束扫过之后，其发光亮度只能维持短暂一瞬(大约几十毫秒)便消失。为了使人眼能看到稳定的图像，就必须在图像消失之前使电子束不断地重复扫描整个屏幕。这个过程叫做刷新(refresh)。每秒刷新的次数称刷新频率或扫描频率。结合人的视觉生理，刷新频率应大于 30 次/秒，人眼才不会感到闪烁。在平板显示器中的刷新概念类似。显示设备中通常选用电视中的标准，每秒刷新 50 帧(frame)图像。

(5) 帧存储器和视频随机存储器(VRAM)。为了不断提供刷新图像的信号，必须把图像存储起来，存储图像的存储器叫“帧存储器”或“视频存储器”(VRAM)(电视不用帧存储器也可以看到图像，是因为电视接收机不断接收从天线来的信号)。帧存储器的容量由图像分辨率和灰度级决定，分辨率为 1024×1024，256 级灰度的图像，存储容量为 1024×1024×8b=1MB。帧存储器的存取周期必须满足刷新频率的要求。

VRAM 的设计考虑了图形数据存取的特点，在 VRAM 芯片中实现了移位功能，双端口 VRAM 能同时读、写图形数据，性能比一般 DRAM 好，但价格贵。

(6) 显示存储器。早期的显示图形适配器只起到 CPU 与显示器之间的接口作用，如今的适配器还起到处理图形/图像数据、加速显示等作用，在适配器中的存储器称为显示存储器。

(7) 光栅扫描和随机扫描。电子束在荧光屏上按某种轨迹运动称为扫描。随机扫描是控制电子束在需要作图或显示字符的地方扫描，而不必扫描整个屏幕，现在已经很少见。光栅扫描是电视中采用的方法，要求电子束扫过整个屏幕，采用从上至下顺序扫描的方法，有逐行扫描和隔行扫描两种方式。在计算机的显示器中广泛采用光栅扫描。

8. 设某光栅扫描显示器的分辨率为 1024×768，帧频为 50(逐行扫描或隔行扫描)，回扫和水平回扫时间忽略不计，则此显示器的行频是多少？每一像素允许读出时间是多少？

答：(1) 逐行扫描

行频＝行数×帧频＝768×50＝38400s^{-1}

每一像素从帧存储器读出的时间间隔$=\frac{1}{1024\times768\times50}\text{s}\approx0.0254\mu\text{s}\approx25\text{ns}$

(2) 隔行扫描

$$行频=\frac{768}{2}\times50=19200\text{s}^{-1}$$

$$读出时间=\frac{1}{1024\times(768/2)\times50}\approx12.5\text{ns}$$

9. 比较光栅扫描的图形图像显示器与光栅扫描的字符显示器的主要异同点。

答：相同点：按构成一帧显示内容的像素点逐行扫描，显示一帧内容。

不同点：图形和图像显示器需将每个像素的信息都存放在 VRAM 中；而字符显示器只需将要显示字符的 ASCII 码存放在 VRAM 中，字符的点阵来自字形发生器的 ROM。

10. 填充题

对于单色字符显示器，若每屏可显示 80 列×25 行=2000 个字符，字符窗口为 9×14 点阵，字符为 7×9 点阵。其 VRAM 中存放的是[A]，容量为[B]字节。如果是图像显示器，帧存储器的容量是[C]字节。

答：A——2000 个字符的 ASCII 码，B——2000（每个 ASCII 码用 1 字节表示），C——31.5K（容量=2000×9×14/8=31.5KB）

11. 从供选择的答案中，选出正确答案填入[ ]中。

几种字符打印机（串行点阵针式打印机、行式点阵打印机、激光打印机和喷墨打印机）的特点可归纳如下：串行点阵针式打印机是按[A]打印的，打印速度[B]；喷墨打印机是按[C]打印的，速度[D]；激光打印机是按[E]打印的，速度[F]；行式点阵针式打印机是按[G]打印的，速度[H]。

所有打印机的打印都受到打印字符的点阵的控制。打印字符的点阵信息在点阵针式打印中控制打印针[I]，在激光打印机中控制激光束的[J]。

供选择的答案：

A，C，E，G：①字符；②行；③页。

B，D，F，H：①最快；②最慢；③较快；④中等。

I，J：①运动方向；②有无；③是否动作。

答：A——①；B——②；C——①；D——④；E——③；F——③；G——②；H——③；I——③；J——②

12. 从供选择的答案中，选出正确答案填入[ ]中。

一级汉字有 3755 个，如每个汉字字模采用 16×16 点阵，并存放在主存中，约占用[A]字节。假如将汉字显示在荧光屏上，共 24 行，每行 80 个字，为保存一帧信息，约需[B]字节的存储空间。

汉字在输入时采用[C]，在存储时采用[D]，打印或显示时用[E]。存储一个汉字的机内码一般可用[F]字节，有时也用[G]字节。

供选择的答案：

A,B：①30K；②60K；③90K；④120K。

C,D,E：①ASCII 码；②字形码；③机内码；④点阵；⑤拼音码；⑥区位码。

F,G：①1 个；②2 个；③3 个；④32 个；⑤16 个。

答：A——④(3755×16×16/8＝120.2KB)；B——②(24×80×16×16/8＝61.4KB)；C——②或⑤或⑥；D——③；E——④；F——②；G——③

13. 针式打印机由哪几部分组成？简述打印字符的过程。

答：针式打印机由打印头和字车、输纸机构、色带机构和控制器 4 部分组成。

针式打印机的印字方法是由打印针印出 n(横)×m(纵)个点阵组成的字符图形。为减少打印头制造的难度，串行点阵打印机的打印头中一般只装有一列 m 根打印针，每根针可以单独驱动，印完一列后打印头沿水平方向移动一步微小距离，n 步以后，形成一个 $n\times m$ 点阵，照此逐个字符打印。

针式打印机有单向打印和双向打印两种。当打印完一行字符以后，打印纸在输纸机构控制下前进一行，同时打印头(字车)回到一行的起始位置，重新由左至右打印，这个过程为单向打印。双向打印指的是自左至右一行字符打印完毕后，字车无须回车，在输纸的同时，字车走到反向打印的起始位置，再从右至左打印一行。反向打印结束，字车又回到正向打印起始位置。由于双向打印省去了空回车时间，所以打印速度大大提高。

14. 填充题

目前使用的打印机有击打式和[A]两种，属于击打式打印机的是[B]，属于非击打式打印机的有[C]、[D]、[E]等。

激光打印机的墨粉、感光鼓等都安装在[F]内，打印的图像(墨粉)形成于墨盒中的[G]上，再传送到[H]上，并熔融到[H]上。

答：A——非击打式；B——针式打印机；C——激光打印机；D——喷墨打印机；E——热转印打印机；F——墨盒；G——感光鼓；H——打印纸

15. 填充题

激光打印机、喷墨打印机和热转印打印机的印字质量比针式打印机[A]。

喷墨打印机的喷墨技术有连续式和[B]两种，当前大多采用[C]喷墨技术。随机式喷墨打印机的喷墨机构可采用多种技术，目前流行的是[D]和[E]喷墨技术。

答：A——好；B——随机式；C——随机式；D——压电式；E——热电式

第 10 章　输入输出(I/O)系统

1. 填充题

输入输出系统包括外部设备(输入输出设备和 A)及其与主机(CPU 和 B)之间的控制部件。

辅助存储器与主机(CPU 和主存储器)之间的数据传送控制方式是和 C 相同的。

答：A——辅助存储器；B——主存储器；C——输入输出设备

2. CPU 通过什么方法完成输入输出操作？

答：(1) 通过输入输出指令完成输入输出操作。

为了 CPU 便于对 I/O 设备进行寻址和选择，必须给每一台设备规定一些地址码(称为设备号或设备代码)，例如，某台设备有 2 个数据寄存器、1 个控制寄存器和 1 个状态寄存器，则赋以 4 个设备地址码。CPU 使用 I/O 指令向设备控制器发送命令，例如向控制寄存器发送启动设备命令，或从设备数据寄存器取数据，或从状态寄存器获取设备状态，了解工作是否正常……。I/O 指令通常包含操作码和设备地址码等。

某些计算机没有设置专门的 I/O 指令，而是利用取数存数指令完成输入输出功能，使用这种方法时，从主存的地址空间中分出一部分地址码作为 I/O 设备的代码，当访问到这些地址时，表示被访的不是主存，而是 I/O 设备。

(2) 直接存储器存取(DMA)。

利用 I/O 指令或存数取数指令启动 I/O 设备后，CPU 不再管理 I/O 设备，而由 I/O 设备的设备控制器控制 I/O 设备与主存储器之间的数据传送，称之为直接存储器存取(DMA)。CPU 要用到这些数据时再到主存储器去取，CPU 要输出数据也是先送到主存。

3. I/O 设备的数据传送方式有哪几种？

答：有 5 种。

(1) 程序直接控制方式；

(2) 程序中断传送方式；

(3) 直接存储器存取方式；

(4) I/O 通道控制方式；

(5) 外围处理机方式。

一般小型机和微机采用前 3 种方式，大型机还采用 I/O 通道控制方式或外围处理机方式。

4. 从供选择的答案中，选出正确答案填入□中。

计算机系统的输入输出接口是 A 之间的交接界面。主机一侧通常是标准的 B 。一般这个接口就是各种 C 。

供选择的答案：

A：① 存储器与 CPU；②主机与外围设备；③存储器与外围设备。

B：①内部总线；②外部总线；③系统总线。

C：①设备控制器；②总线适配器。

答：A——②和③；B——③；C——①。

5. 从供选择答案中，选出正确答案填入□中。

中断处理过程中保存现场的工作是[A]的。保存现场中最基本的工作是保存断点和当前状态，其他工作是保存当前寄存器的内容等。后者与具体的中断处理有关，常在[B]用[C]实现，前者常在[D]用[E]完成。

设 CPU 中有 16 个通用寄存器，某中断处理程序运行时仅用到其中的 2 个，则进入该处理程序前要把这[F]个寄存器内容保存到内存中去。

若某机器在响应中断时，由硬件将 PC 保存到主存 00001 单元中，而该机允许多重中断，则进入中断程序后，[G]将此单元的内容转存到其他单元中。

供选择的答案：

A：①必需的；②可有可无的。

B，D：①中断发生前；②响应中断前；③具体的中断服务程序执行时；④响应中断时。

C，E：①硬件；②软件。

F：①16；②2。

G：①不必；②必须。

答：A——①；B——③；C——②；D——④；E——①；F——②；G——②。

6. 从供选择的答案中，选出正确答案，填入□中。

设置中断触发器保存外设提出的中断请求，是因为[A]或[B]。后者也是中断分级、中断排队、中断屏蔽、中断禁止与允许、多重中断等概念提出的缘由。

供选择的答案：

A，B：①中断不需要立即处理；②中断设备与 CPU 不同步；③CPU 无法对发生的中断请求立即进行处理；④可能有多个中断同时发生。

答：A——③；B——④。

7. 从供选择的答案中，选出正确答案，填入□中。

一般 CPU 在一条指令执行结束前判断是否有中断请求，若无，则执行下条指令；若有，则按如下步骤进行中断处理。步骤 a，关中断，然后将断点（PC 内容）和程序状态字等现场保存，并转入中断处理程序。步骤 b，判断中断源，根据中断源进入处理相应中断的处理程序。步骤 c，先做好交换新旧中断屏蔽码等准备工作后即执行开中断，然后进入具体的中断服务程序执行中断服务。步骤 d，先关中断，然后恢复现场。步骤 e，先执行开中断，然后立即执行中断返回。

步骤 a 由[A]实现。若采用向量中断方式，则不必执行步骤 b。步骤 c 中开中断的目的是[B]。由于设置了[C]，故可在多重中断发生时改变中断响应顺序。步骤 e 的开中断是由[D]实现的。

供选择的答案：

A：①程序；②中断隐指令（硬件）。

B：①使原来的屏蔽码不起作用；②便于高级的中断请求得以及时处理。

C：①新的屏蔽码；②开中断。

D：①程序；②硬件。

答：A——②；B——②；C——①；D——①。

8. 程序中断设备接口由哪些逻辑电路组成？各逻辑电路的作用是什么？

答：一般由下列逻辑电路组成：

(1) 设备码寄存器和设备选中电路。当 CPU 需使用某外设时，通过 I/O 指令或其他访问 I/O 设备地址的指令，将设备码通过地址线送往所有外围设备接口，但仅仅具有该设备号的设备才产生选中信号。于是，该外围设备及其接口才能响应主机的控制并进行数据传送。

(2) 中断请求触发器。设备向 CPU 申请中断时，将该触发器置“1”，CPU 响应中断后将其置“0”。每个设备有一个中断请求触发器。

(3) 中断排队电路。当有多个中断源时，根据各个设备的优先级，选出请求中断的各个中断源中优先级最高的设备。

(4) 中断屏蔽触发器(或寄存器)。每个设备分配有一个屏蔽触发器，当该触发器为“1”时，屏蔽该设备的中断请求，中断屏蔽触发器的状态是由程序设置的，因此可以利用程序改变设备的优先级。所有设备的中断屏蔽触发器集中在一起，组成中断屏蔽寄存器。

(5) 中断类型码寄存器(中断向量)。设备接收到 CPU 发出的“响应中断”信号后，由设备送出中断类型码(中断向量)，从而得到中断服务程序的入口地址。也可以用设备码来代替中断类型码。

(6) 数据寄存器。存放设备与 CPU 之间传送的数据。

9. 在单重中断方式下(即进入中断处理后不再响应其他中断请求)，CPU 响应中断的步骤是哪些？在多重中断方式下 CPU 响应中断的步骤又是哪些？

答：单重中断方式 CPU 响应中断的步骤如下：

关中断，保存现场，判断中断源，执行与设备有关的中断服务程序，恢复现场，开中断，返回原程序。

多重中断方式 CPU 响应中断的步骤如下：

关中断，保存现场(包括保存原中断屏蔽字并取新屏蔽字)，判断中断源，开中断(如果有多个中断源发出中断请求，则要增加处理步骤，举例见第 14、15 题)，执行与设备有关的中断服务程序，关中断，恢复现场(包括恢复中断屏蔽字)，开中断，返回原程序。

上述步骤中的开中断和关中断是通过 CPU 中设置的一个开中断触发器实现的，当该触发器为“1”时，开中断，此时 CPU 可响应中断请求；当它为“0”时，关中断，此时 CPU 不响应任一中断请求(不可屏蔽中断除外)。

在执行上述单重中断的步骤中，一开始就“关中断”，直到处理完毕才“开中断”，在执行过程中，如果有不可屏蔽(例如停电)中断产生，还是应该及时处理的。

10. 填充题。

CPU 在执行正常程序(未响应中断)时，CPU 的级别应 $\boxed{A}$ 所有中断源的中断级别，这时 CPU 在开中断的情况下，可响应任一中断源的中断请求，当有多个中断源同时提出中断请求时，CPU 响应其中级别 $\boxed{B}$ 的中断源的中断请求。

当 CPU 响应中断后，CPU 的级别(即执行主程序的级别)应 $\boxed{C}$ 它正在处理的中断源级

别，此时，若有新的中断请求，则新中断源的级别必须 D 当前正在执行的程序的级别，才能得到 CPU 的响应。

答：A——低于；B——最高；C——等于；D——高于。

11. 填充题。

在多重中断系统中，在开中断的情况下，CPU 按 A 和 B 来确定响应哪一个中断源的中断请求和是否响应新的中断请求。

一般 A 是由硬件连接而实现的，B 是由软件设定的。

答：A——中断排队电路优先级别；B——中断屏蔽字。

12. 根据中断产生的原因，可将中断分为哪几种？

答：可分为内中断（故障中断）、外中断（外部设备提出的中断）和软中断（中断指令或陷阱指令）。

一般情况下，故障中断（掉电、算术运算结果溢出和除数为 0 等）是非屏蔽中断，外部设备中断是可屏蔽的。软中断是由于在程序中安排了一条中断指令而转移到指令所指定的程序入口继续运行。

13. 怎样进入中断服务程序入口？

答：可以采用不同的转向中断服务程序入口地址的方法。一种是在中断程序中设一条专门接收中断设备码的指令 INTA，取到申请中断且优先级最高的设备的设备号后，再通过主存的中断跳转表产生中断服务程序入口地址。

跳转表中存放的就是每个设备的中断服务程序入口地址。跳转表通常安排在主存的 0 页，此时跳转表地址高位部分为 0，低位部分就是设备号（跳转表也可安排在主存的其他页中，此时跳转表高位地址即是页号，低位部分仍是设备号），这样中断服务程序就可安排在主存的任意地方。

另一种目前应用更广泛的方法叫做向量中断。

向量中断方式是为每一个中断源设置一个中断向量，由被选中的设备直接产生中断向量，再通过主存的中断跳转表（或称为中断向量表）转到中断服务程序的入口。所以中断向量是中断服务程序入口地址的地址。

14. 有 5 个中断源 A1、A2、A3、A4、A5，它们的中断排队电路优先级从高到低分别是 1 级、2 级、3 级、4 级和 5 级，各级中断源在响应中断后正常设置的中断屏蔽码和改变后的中断屏蔽码如表 10.1 所示，其中“1”表示该中断源被屏蔽，“0”表示该中断源不被屏蔽。

表 10.1　中断源排队电路的优先级和屏蔽码

中断源	中断排队电路优先级	正常的中断屏蔽码 A1 A2 A3 A4 A5	改变后的中断屏蔽码 A1 A2 A3 A4 A5
A1	1	1 1 1 1 1	1 0 0 0 0
A2	2	0 1 1 1 1	1 1 0 0 0
A3	3	0 0 1 1 1	1 1 1 0 0
A4	4	0 0 0 1 1	1 1 1 1 1
A5	5	0 0 0 0 1	1 1 1 0 1

如果 5 个中断源同时提出中断请求，请回答：

(1) 在中断源提出中断申请之前，中断屏蔽码应该是何值？

(2) 仅使用正常的中断屏蔽码，处理机响应各个中断请求的先后次序是什么？完成各个中断请求的先后次序是什么？

(3) 当使用改变后的中断屏蔽码时，处理机响应各个中断请求的先后次序是什么？完成各个中断请求的先后次序是什么？请画出程序运行过程的示意图。

答：(1) 在所有中断源提出中断请求之前，A1、A2、A3、A4、A5 的中断屏蔽码应是 0、0、0、0、0。

(2) 处理机响应各个中断请求的先后次序为 A1→A2→A3→A4→A5，完成一个再响应下一个，因此完成的次序也是 A1→A2→A3→A4→A5。

(3) 改变中断屏蔽码后，处理机响应中断和处理中断过程如图 10.1 所示。

图 10.1　处理机响应中断和处理中断过程

因为中断响应硬件只能按照由第 1 级到第 5 级的次序响应，所以先进入第 1 级中断处理程序。由于第 1 级的中断屏蔽码 10000 对第 2 级到第 5 级中断是开放的，所以就经中断响应进入第 2 级中断程序，同理进入第 3 级中断和第 4 级中断程序。由于第 4 级中断对所有中断级屏蔽，所以先处理。处理完后返回第 3 级中断，由于第 3 级中断对第 4 级和第 5 级中断是开放的，所以响应第 5 级中断，并进行处理。然后逐次返回第 3 级中断进行处理、第 2 级中断进行处理、第 1 级中断进行处理。

所以在该例中处理机硬件响应中断的次序是 A1→A2→A3→A4→A5，而完成中断处理的次序是A4→A5→A3→A2→A1。

15. 如果 5 级中断源(1,2,3,4,5)的优先次序为 1→4→5→2→3(从高到低)，请求出各级中断处理程序的中断屏蔽码(即改变后的中断屏蔽码)。

答：给出的中断屏蔽码如表 10.2 所示。

16. 在一个 5 级中断系统中，硬件中断响应从高到低的优先顺序是 1→2→3→4→5。

(1) 请设计中断屏蔽码，使完成中断处理的顺序改为 1→2→4→3→5。

(2) 当 CPU 执行程序时有 2、3、4 和 5 级的中断请求同时到达，CPU 在按优先顺序处理第 3 个中断过程中又有 1 个 1 级中断请求来到，请画出 CPU 处理这些中断的过程示意图。

答：(1) 给出的中断屏蔽码如表 10.3 所示。

表 10.2　各级中断处理程序设置的屏蔽码

中断源	中断屏蔽码				
	1	2	3	4	5
1	1	1	1	1	1
2	0	1	1	0	0
3	0	0	1	0	0
4	0	1	1	1	1
5	0	1	1	0	1

表 10.3　中断屏蔽码

中断源	中断屏蔽码				
	1	2	3	4	5
1	1	1	1	1	1
2	0	1	1	1	1
3	0	0	1	0	1
4	0	0	1	1	1
5	0	0	0	0	1

(2) CPU 响应中断和处理中断的过程如图 10.2 所示(按设计的中断屏蔽码处理)。

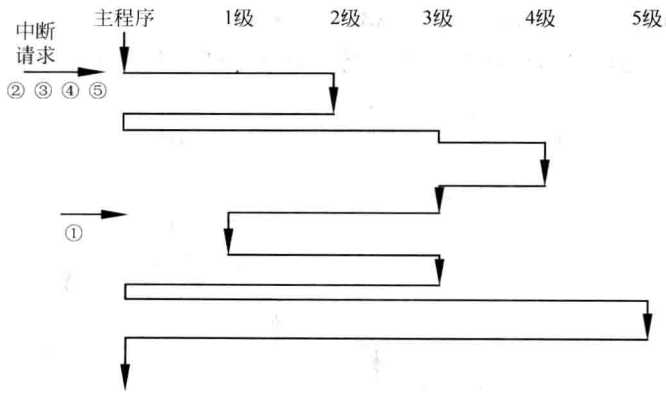


图 10.2　CPU 响应中断和处理中断过程

17. 已知多重中断处理过程如图 10.3 所示,能求出中断优先级的顺序和设计出中断屏蔽码吗?

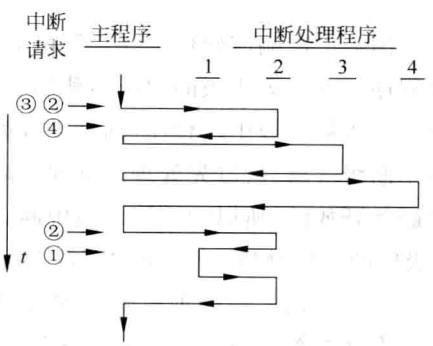


图 10.3　多重中断处理示意图

答:分析图 10.3 可知:

- ③与②同时发中断,响应②,说明②的优先级>③,其优先顺序为 2→3。
- 在②执行期间,来中断请求④时,不予响应,所以优先次序为 2→4。
- ②中断处理结束后,同时存在③与④,但响应③的请求,所以优先次序为 3→4。
- ②能被①中断,说明①优先于②,1→2。

综合以上所述,优先次序为 1→2→3→4。

根据优先级,列出中断屏蔽码如表 10.4 所示。

如果硬件(中断排队电路)的排队次序为 1→2→3→4。那么表 10.5 设置的中断屏蔽码也能完成图 10.3 的处理过程。

表 10.4 和表 10.5 在实现图 10.3 时,其处理过程是相同的。但在其他情况下则可能有差异,例如采用表 10.4 的屏蔽码时,当执行中断源③的中断处理程序时,可被中断源②的中断请求打断;而采用表 10.5 的屏蔽码时,则不能打断,从软件角度来看,中断源②与中断源③属于同一级别。

表 10.4　中断屏蔽码(一)

中断源	中断屏蔽码			
	1	2	3	4
1	1	1	1	1
2	0	1	1	1
3	0	0	1	1
4	0	0	0	1

表 10.5　中断屏蔽码(二)

中断源	中断屏蔽码			
	1	2	3	4
1	1	1	1	1
2	0	1	1	1
3	0	1	1	1
4	0	0	0	1

18. 设某计算机有 3 个中断源,其中断请求信号为 INTR1、INTR2 和 INTR3,而 CPU 只有 1 个中断请求输入线 INTR,1 个中断响应输出线 INTA。中断请求级别从高到低依次为 INTR1、INTR2 和 INTR3。请设计该系统的中断判优电路和中断向量生成电路。

为完成输入或输出操作,还应该有哪些功能电路?

答:中断判优采用串行链接电路(如图 10.4 所示),由 2 个与门(1,2)和 2 个反相器构成,串行链接电路的输入端通过电阻 R 接+5V 电源,当有 INTR1 中断请求信号时,门 1 的输出为低电位,从而封锁了门 2、门 4 和门 5。

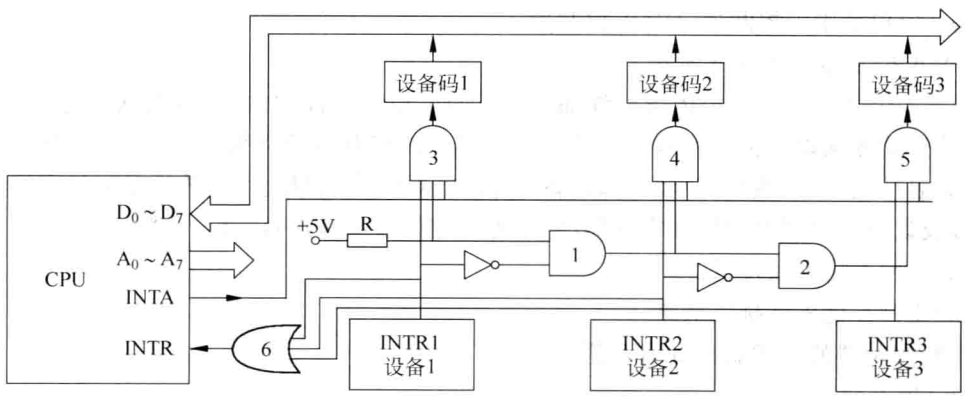


图 10.4　中断判优和中断向量生成图

INTR1、INTR2 和 INTR3 连接到门 6,门 6 的输出接到 CPU 的 INTR(也可以利用 OC 电路或三态门代替门 6),CPU 接收到 INTR 请求后,发出 INTA 信号,将串行链接电路中申请中断且优先级最高的设备码(或中断向量)送往数据总线 $D_0 \sim D_7$,这是通过门 3、门 4、门 5 及其控制的设备码寄存器实现的。CPU 根据设备码或中断向量访问主存中的中断跳转表,设备码或中断向量是相应的中断处理程序入口地址的地址。然后将相应的中断请求清除。

为完成 I/O 操作,还应该有数据缓冲器,CPU 在执行中断处理程序时,从地址线 $A_0 \sim A_7$ 送出数据缓冲器的地址,将数据缓冲器的数据送到 $D_0 \sim D_7$(当输入时),或通过 $D_0 \sim D_7$ 将数据送数据缓冲器(当输出时)。

如要用软件改变中断优先级,还应该设置中断屏蔽寄存器。

19. 直接存储器存取(DMA)有哪 3 种工作方式?

答:DMA 是 I/O 设备与主存储器之间由硬件组成的直接数据通路,用于高速 I/O 设备与主存之间的成组数据传送。数据传送是在 DMA 控制器控制下进行的。

DMA 的 3 种工作方式如下：

(1) CPU 暂停方式。主机响应 DMA 请求后，让出存储总线，直到一组数据传送完毕后，DMA 控制器才把总线控制权交还给 CPU，采用这种工作方式的 I/O 设备，在其接口中一般设置有存取速度较快的小容量存储器，I/O 设备与小容量存储器交换数据，小容量存储器与主机交换数据，这样可减少 DMA 传送占用存储总线的时间，也即减少 CPU 暂停工作时间。

(2) CPU 周期窃取方式。DMA 控制器与主存储器之间传送一个数据，占用(窃取)一个 CPU 周期，即 CPU 暂停工作一个周期，然后继续执行程序。

(3) 直接存储器访问工作方式。这是标准的 DMA 工作方式。如果传送数据时 CPU 正好不占用存储总线，则对 CPU 不产生任何影响。如果 DMA 和 CPU 同时需要访问存储总线，则 DMA 的优先级高于 CPU。

在 DMA 传送数据过程中，不能占用或破坏 CPU 硬件资源或工作状态，否则将影响 CPU 的程序执行。

20. 某外部设备采用 DMA 方式传送数据到主存储器，该设备的控制器应包括哪些内容？

答：应包括中断控制器、DMA 控制器。

DMA 控制器包括以下内容：

(1) 多个寄存器。整个传送过程都是由 DMA 控制器控制的，因此每次传送时主存的地址、传送的数据以及传送字数的记录都要有相应的寄存器予以保存。另外还需要保存命令(例如启动设备、读或写等)和外部设备工作状态(例如数据校验错、传送是否结束等)的寄存器以及设备自身读写的地址寄存器(例如磁盘存储器的盘片号、道号和扇区号等)。

(2) DMA 请求触发器。

(3) DMA 优先级排队电路。

中断控制器的组成见本章第 8 题。

21. 填充题

采用 DMA 方式传送一组数据分 3 个阶段：[A]，[B]和[C]。其中[A]和[C]是用程序实现的，[B]是用硬件实现的。

DMA 数据传送完毕后，要发出[D]，DMA 设备中有两个优先级排队电路：[E]和[F]。

答：A——预处理；B——数据传送；C——后处理；D——中断请求；E——中断优先级排队电路；F——DMA 优先级排队电路。

22. 判断以下有关 DMA 控制的叙述的正确性，如有错误，请纠正。

(1) DMA 设备的中断优先级一定比所有的程序中断设备的中断优先级高。

(2) 在各个 DMA 设备中，其 DMA 优先级排队电路的优先次序与中断优先级排队电路的次序是相同的。因此可以使用同一个排队电路。

(3) DMA 设备传送一组数据结束后的工作可以由中断程序完成，也可以在主程序中安排完成。

答：以上叙述都有错误。有关(1)和(2)的解释如下：

DMA 和程序中断优先级考虑原则是：首先要考虑数据不会丢失，不至造成错误，其次

传输率高的设备一般排在前面。但是 DMA 设备在程序中断处理阶段和数据传输阶段的情况是不同的，例如在数据传送阶段，磁盘存储器的传输率一般高于其他设备，因为相邻两个字的传送间隔时间较短，在 DMA 优先级排队电路中，它的优先级较高，然而其中断优先级并不一定比其他设备的优先级（包括程序中断设备）高，因为当 DMA 设备（例如磁盘）发出中断请求时，即使稍晚一些处理，也不会造成错误，当然它的优先级也不会是最低的。

在中断优先级排队电路中，DMA 设备和程序中断设备是混合排队的。计算机的 DMA 优先级排队电路和程序中断优先级电路基本上是两个独立的排队电路。

（3）由于 DMA 设备一组数据的传送时间难以精确估计，所以数据传送结束后发中断，由中断程序处理，而不是在主程序中安排的。另外假如数据传送过程中发现错误，也会发中断请求。

23. 填充题

一般数据传输率低的设备采用 A 工作方式，例如键盘、触摸屏等人工操纵的设备，其两次操作间的间隔时间较长，因此都采用 B 工作方式。磁盘、磁带等设备的数据传输率高，采用 C 工作方式来传送数据。

DMA 控制器访问主存储器有 3 种方式：(1)CPU 暂停方式，(2) D ，(3) E 。其中最常用的是 F 。

答：A——程序中断；B——程序中断；C——DMA；D——CPU 周期窃取方式；E——直接访问存储器方式；F——直接访问存储器方式。

24. 判断下列叙述的正确性，如有错误，请纠正。

（1）DMA 设备在传送数据时，可以利用 DMA 屏蔽寄存器改变设备的数据传送次序，从而实现由软件改变数据传送的优先级。

（2）在采用“CPU 周期窃取”的 DMA 工作方式时，由于数据传送时，CPU 程序暂停一个时钟周期，因此允许 DMA 设备将数据传送到 CPU 的通用寄存器中。

（3）DMA 设备向主存储器传送数据时，为了保证 cache 和主存储器的一致性，该数据同时也传送到 cache 中。

（4）中断服务程序一般是由用户根据所应用的外部设备特点而编写的。

（5）由程序查询方式确定中断源比采用中断向量方法确定中断源的速度相差不多。

（6）CPU 响应中断后，首先执行关中断和保存程序断点操作，这些操作可以由硬件完成，也可用软件实现（执行程序中的指令）。

（7）在多重中断情况下，CPU 的现场信息可保存到通用寄存器或堆栈中，堆栈是在主存储器中安排的先进后出存储区。

答：上述各条均有错误，说明如下：

（1）DMA 设备不设 DMA 屏蔽寄存器，当开始传输数据后，传输过程不再受程序控制，相邻两个数据相隔的时间很短，不允许插入程序。

（2）在执行 DMA 传送时，CPU 在同时执行程序，不允许 DMA 破坏 CPU 运行现场，不允许改变通用寄存器中的内容。

（3）DMA 设备向主存储器传送数据时，一般不传送到 cache。为了保证 cache 和主存

的一致性，一般在 cache 控制器内设有监听线路，当被访的主存储器单元在 cache 中有副本时，将相应的 cache 行的“有效位”置为“0”，使该 cache 行作废。

(4) 中断服务程序一般包含在操作系统内，由操作系统的设计人员(系统程序员)编写。

(5) 程序查询方式确定中断源时要逐个询问设备是否有中断请求，直到找到中断源为止，中断向量则由硬件直接给出应该得到响应的中断源的间接地址，所以快得多。

(6) 关中断和保存程序断点操作一般是由硬件完成的，尤其是关中断，必须由硬件完成，否则在关中断之前有可能被更高级的中断所打断，从而丢失一个程序断点。

(7) 现场信息不能保存到通用寄存器，其理由同(2)。一般保存在堆栈，堆栈是在主存储器中开辟的一个供程序嵌套使用的专用存储区。

25. 填充题

在大型计算机中，通常用 A 或 B 来管理外部设备。它们都有自己的指令系统，并能实现指令所规定的操作。其中 A 不是一个完全独立的处理机，它配合 CPU 工作；B 可以是一个能独立工作的处理机，它在计算机系统中被分配来完成对 I/O 设备或辅助存储器的控制，还可完成码制变换、格式处理、数据块的检错和纠错等操作。

答：A——I/O 通道；B——输入输出处理机。

26. I/O 通道分成哪几类？简述其区别和最大数据传输率。

答：根据数据传送方式，可将 I/O 通道分成字节多路通道、选择通道和数组多路通道。

字节多路通道通过多条子通道连接多台慢速 I/O 设备轮流(交叉)传送数据(字节)。当通道传送某一设备的数据时，其他设备可以并行工作：准备需要传送的数据字节，或处理收到的字节，以准备接收新字节。字节多路通道的最大数据传输率为该通道连接的所有 I/O 设备的传输率之和。

选择通道可以连接多台 I/O 设备，但每次只能选择一台设备传送数据，适应快速 I/O 设备连续传送数据的要求。各 I/O 设备之间不能并行工作，当一个设备完成所要求的数据传送之后，才能选择另一个设备传送数据。选择通道的传输率等于其中 1 个速度最快的 I/O 设备的传输率。

数组多路通道可以连接多台快速 I/O 设备以成组交叉方式传送数据。当通道传送某一设备的数据块时，其他设备可以并行地工作，例如进行数据块传送的准备工作等。只有当一个设备数据块传送完毕后，通道才能为另一设备传送数据。因此数组多路通道综合了字节多路通道和选择通道的优点，既允许各子通道间并行工作，又以数据块为单位成组地连续传送，具有很高的数据传输率。数组多路通道的数据传输率应该等于其中 1 个速度最快的 I/O 设备的传输率。

27. 填充题。

(1) 在采用 DMA 设备的计算机中，主存储器接收到的访问要求可以来自 A、B 和 C。因此在主存储器的控制器中有 D 电路，用以决定为哪个访问源服务。

(2) 在 DMA 的 3 种工作方式中，E 方式速度可能最快，因为它一般是在外设中的小存储器与主存之间传送数据，但是这可能会影响 CPU 与其他设备的正常工作，因此比较理想的还是采用 F 方式。

答：A——CPU；B——主存的刷新请求；C——DMA 设备；D——优先排队；E——CPU 暂停；F——直接存储器访问。

28. 填充题。

CPU 与外部设备之间的联系是通过[A]实现的。

在单总线结构中，连接 CPU、主存和外设的总线称为[B]总线。在双总线结构中，连接 CPU(或 cache)和主存的总线称为[C]总线，系统总线通过扩展总线接口连接[D]。在多总线结构中将速度较快的设备(例如磁带、磁盘)与速度较慢的字符设备(例如键盘)分别连到不同的总线，在这两种总线之间是通过[E]相连接的。

答：A——设备控制器；B——系统；C——系统；D——外部设备；E——扩展总线接口/扩充总线桥。

29. 填充题。

从两个或两个以上源部件传送信息到一个或多个部件的一组传输线称为[A]。可能存在多个设备或部件同时申请对总线的使用权，为保证在同一时间内只能有一个申请者使用总线，需要设置[B]机构，该机构按照申请者的优先权选择可以控制总线的设备或部件。可以控制总线的设备称为[C]，受其控制的设备称为[D]。在计算机系统中，CPU 是主设备，[E]是从设备；当外部设备与 CPU 联系时，CPU 是主设备；当外部设备与主存储器交换数据时，[F]是主设备。

答：A——总线；B——总线判优控制；C——主设备；D——从设备；E——主存储器；F——外部设备。

30. 填充题。

(1) PCI 是一种并行总线，是微机中用得较多的系统总线，其传送字的宽度为[A]位。能实现[B]功能，即在加电时，操作系统可自动检测机器配置，而给各个外设分配中断请求优先级和存储器缓冲区等。

(2) 在当前流行的微机或小型机中，用得较多的外部设备接口有[C]和[D]等。

(3) 连接外设的接口分为串行接口和[E]接口两类。一般串行接口按位传送，连接速度较慢或离主机较远的外部设备，并行接口连接传输率[F]的外部设备。但是当前传输率进一步提高，由于并行传输线之间的同步和线间干扰等问题而影响了可靠性，串行高速传送的接口又应运而生。

答：A——32/64；B——即插即用；C——SCSI；D——PCI Express；E——并行；F——较高。

31. 将数据从主存的一个区移到另一区通常是怎样实现的？将磁盘上的数据移到磁带上是怎样实现的？

答：当数据在主存内部传送时，一般是通过传送指令实现的，有的机器需要经过 CPU 中的寄存器来转送；有的计算机中设置有成批传送数据指令。成批传送数据指令的执行时

间比较长，通常在执行这条指令时，允许中途被中断请求暂时中止，处理完中断程序后再继续传送数据。

磁盘与磁带之间不能直接传送数据，需要通过 DMA 方式将磁盘上的数据送到主存，再用 DMA 方式将主存中的数据传送到磁带上。

32. 从供选择的答案中选出正确答案填入□ 中。

某行式打印机打印速度为每分钟 760 行，每行 132 字节。打印机经异步串行口与主机相连。设传送效率为 0.8，但在传输前将数据进行压缩处理，使得传送效率提高一倍。串口的波特率应选择 A 。若打印机以中断方式与 CPU 传输数据，则在接口应有 B 逻辑电路。

供选择的答案：

A：①2400b/s，②4800b/s，③9600b/s。

B：①串-并转换，②并-串转换。

答：A——③，串行传输率$=\frac{760}{60}\times 132B\div 0.8\div 2=1045B/s=8360b/s$，所以选择③。

B——②，CPU 将数据并行传送到打印机控制器（接口），转换成串行数据送打印机，所以接口中有并-串转换逻辑电路。

33. 某计算机的 I/O 系统有 2 个选择通道和 1 个字节多路通道，每个选择通道上连接 2 台磁盘机和 1 台磁带机，字节多路通道上连接了 3 台行式打印机和 5 台终端。已知各设备的传输率如下：磁盘机为 1MB/s，磁带机为 250KB/s，行式打印机为 7KB/s，终端为 1KB/s。

计算该计算机系统的最大 I/O 数据传输率。

答：1 个选择通道的数据传输率为 1MB/s，字节多路通道的数据传输率$=3\times 7+5\times 1=26KB/s$，总的 I/O 数据传输率$=2\times 1MB/s+26KB/s=2.026MB/s$。

34. 某计算机系统有 3 个 I/O 通道：

（1）字节多路通道，连接传输率为 1.5KB/s 的终端 2 台，传输率为 7KB/s 的打印机 3 台。

（2）选择通道，连接传输率为 1MB/s 的光盘 1 台，传输率为 2MB/s 的温盘 1 台。

（3）数组多路通道，连接传输率为 1MB/s 的温盘和传输率为 300KB/s 的磁带机各 1 台。

计算该计算机的最大 I/O 数据传输率。

答：字节多路通道的最大传输率$=1.5KB/s\times 2+7KB/s\times 3=24KB/s$。

选择通道的最大传输率$=2MB/s$。

数组多路通道的最大传输率$=1MB/s$。

该计算机的最大 I/O 数据传输率$=24KB/s+2MB/s+1MB/s=3.024MB/s$。

35. 假设硬盘的数据传输率为 2MB/s，数据以字为单位传送，字的长度为 32 位，CPU 的时钟频率为 50MHz，试计算在采用下述 3 种方式进行输入操作时所占用的 CPU 时间比率。

（1）程序查询方式。一个轮流查询操作需要 100 个时钟周期（包括传送一个字的时间）。

（2）中断方式。每次中断处理（包括传送一个字）需要 100 个时钟周期。

（3）DMA 方式。DMA 的启动操作需要 1000 个时钟周期，传送完成后处理中断需要

500 个时钟周期，一次传送数据的长度为 4KB，假设磁盘和主存直接传送数据，传送时不占用 CPU 时间。

答：磁盘数据传输率＝2MB/s＝0.5M 字/s。

(1) 程序查询方式：

磁盘每秒传送 0.5M 字的时间＝100×0.5M＝50M 时钟周期

CPU 主频为 50MHz，所以每秒有 50M 时钟周期。

磁盘工作时占用 CPU 的时间/CPU 总时间$=\frac{50M}{50M}=100\%$

此时 CPU 处于临界正常工作状态，此方式不可取。

(2) 中断方式：占用 CPU 时间比率与程序查询方式相同，也为 100%，所以此方式也不可取。

(3) DMA 方式：根据传输率 2MB/s，传送 4KB 需要的时间$=\frac{4KB}{2MB/s}\approx 2ms$。DMA 操作(启动和中断处理)时间＝1000＋500＝1500 时钟周期，占用 CPU 的时间$=\frac{1500}{50M}=30\mu s=0.03ms$，1 次 DMA 操作的总时间＝2ms＋0.03ms＝2.03ms。2.03ms 时间内 CPU 的时钟周期＝50M×2.03ms＝101.5K 时钟周期。DMA 方式磁盘工作时占用 CPU 的时间比率$=\frac{1500}{101.5K}=1.48\%$。

因此磁盘存储器应采用 DMA 工作方式。

36. 某计算机系统 I/O 与存储器之间的总线带宽为 100MB/s。CPU 的性能为 150MIPS(每秒运算 1 百万条指令称为 1MIPS)。今用于事务处理，每个事务处理需要进行 2 次写磁盘和 2 次读磁盘操作，每次读写磁盘时，操作系统要执行 10 000 条指令，而每进行一次事务处理，数据库软件要执行 5 000 条指令，每次传送数据块的大小为 1000 字节。每台磁盘的容量为 200MB，每秒能进行 30 次写磁盘或读磁盘操作。假定对所有磁盘的读写请求不会发生需要等待的情况，而且传输数据时不发生总线冲突。今需要的数据文件存储容量为 10GB，问该系统每秒钟能处理多少个事务？

答：完成本任务需要的磁盘数为 10GB/200MB＝50 台。

仅从磁盘角度考虑，50 台磁盘每秒能进行的读写次数为 50×30＝1500 次，能执行的事务数为 1500/(2＋2)＝375 个。

仅从 CPU 角度考虑，每个事务执行的指令数为(2＋2)×10000＋5000＝45000 条，每秒可进行的事务数为 150M/45000≈3333 个。

仅从总线传输角度考虑，每秒可进行的事务数＝100MB/[(2＋2)×1000B]＝25000 个。

如果上述三者能重叠工作，那么可执行的事务数为 min(375,3333,25000)＝375 个。

如果磁盘读写可与总线传送重叠工作，而与 CPU 操作串行进行，设实际可执行的事务数为 x，则$\frac{x}{375}+\frac{x}{3333}=1$，$x\approx 337$ 个。

37. 某 32 位机的一台外设通过 32 位总线与系统主存相连。CPU 每秒执行 10^6 条指令，平均每条指令需 5 个机器周期，其中 3 个周期必须访问主存，主存读写需 1 个机器周期。假定 CPU 在同时执行 2 个程序。在 95%时间内执行 A 程序且这段时间内不执行 I/O 指

令；在 5%时间内执行 B 程序。现该外设需要把一个非常大的数据块传送到内存。

(1) 如果采用程序 I/O 方式，每传送一个 32 位字的数据需要 CPU 执行 2 条指令。请计算(在上述运行程序情况下)该外设的最大数据传输率(单位：字/s)。

(2) 如果采用 DMA 方式，当 DMA 与 CPU 出现总线冲突时，CPU 优先。请计算(在上述运行程序情况下)该外设的最大数据传输率(单位：字/s)。

答：(1) 程序 I/O 方式，最大数据传输率为：

每秒执行的 I/O 指令数/传送 1 个字的指令数$=(10^6\times5\%)/2=25$K 字/s

(2) DMA 方式，最大数据传输率为(等于系统总线可供 DMA 传送数据的机器周期数)：

$$10^6\times95\%\times(5-3)+10^6\times5\%\times5=2150\text{K 字/s}=2.15\text{M 字/s}。$$

其中第(1)项为 CPU 执行 A 程序时可达到的 DMA 最大数据传输率，第(2)项为 CPU 执行 B 程序时可达到的 DMA 最大数据传输率。

38. 填充题。

(1) 在外设接口中，[A]寄存器的作用是存放 CPU 给外设的操作命令，[B]寄存器的作用是存放外设的工作状态(例如操作完成、校验错等)，[C]寄存器存放输入或输出的数据。

(2) 接口与外设之间有两种传输线的连接方式：[D]连接方式和[E]连接方式，其中[E]连接方式允许传输线的传输距离较长。

(3) 计算机各部件之间的数据传送可采用异步方式或同步方式，当传送的时间能确定时，经常采用[F]，当时间不能确定时，采用[G]。DMA 设备与主存之间的数据传送采用[H]。CPU 与主存之间的数据传送也采用[H]。

答：A——控制；B——状态；C——数据；D——单端(或单线)；E——差分；F——同步方式；G——异步方式；H——异步方式。

39. 程序中断工作方式有哪些作用？

答：(1) CPU 和 I/O 设备并行工作。

(2) 硬件故障处理，出错处理等。

(3) 实现实时处理，在计算机过程控制系统中广泛应用。

(4) 实现多道程序和分时操作。

(5) 实现人机联系。

(6) 实现应用程序和操作系统之间的联系。

(7) 多处理机系统中各处理机之间的联系。

40. 设磁盘盘面有 200 磁道。存储总容量为 1.6MB，磁盘转一圈的时间为 25ms，每磁道分 4 扇区，扇区之间有间隙，磁头通过扇区间的间隙时间为 1.25ms，问：(1)磁盘接口的最大传输率是多少？(2)今有人为上述磁盘机设计了一个接口，磁盘读出数据串行进入移位寄存器，当移满 16 位后，向 CPU 发一个请求接收数据的信号，CPU 响应请求信号，并接收数据。磁盘机不断地串行送下一个 16 位字到移位寄存器，如果 CPU 接收到信号后，最长响应时间是 3μs，这样的接口能否正确工作？应如何改进？

答：(1) 该盘接口的最大传输率$=\dfrac{\text{磁道存储量}}{\text{盘转一圈时间}-\text{磁头通过 4 个扇区间的间隙时间}}$

$$=\frac{1.6\text{MB}/200}{25\text{ms}-4\times1.25\text{ms}}=\frac{8\text{KB}}{20\text{ms}}=400\text{KB/s}$$

(2) 从盘上读出 1 位的时间$=\dfrac{1}{\text{传输率}}=\dfrac{1}{400\text{K}\times8/\text{s}}\approx0.31\mu\text{s}$，即移位寄存器的移位间隔时间。

本题设计的磁盘接口不能正确工作，因为移位寄存器保持一个字不变的时间仅 0.31μs，而 CPU 的最大响应时间为 3μs，所以会丢失数据。

改进方法：再设置一个发送寄存器，每当移位寄存器内移满 1 个字(16 位)时就将其内容送入发送寄存器(移位寄存器可接收下一个字)，再由发送寄存器将数据送 CPU。发送寄存器保持 1 个字的时间$=0.31\times16=5\mu$s，大于 3μs。

41. 今有一磁盘存储器，磁盘转速为 3000r/min，分 8 个扇区，每扇区存储 1KB，主存与磁盘之间每次传送 1 个字，字长 16 位。

(1) 描述从磁盘处于静止状态开始，到将主存中的 2KB 数据传送到磁盘的全过程。

(2) 假如 CPU 执行一条指令的最长时间为 30μs，是否可采用 CPU 在指令结束时响应 DMA 请求的方案？如果不行，应采用怎样的方案？

答：(1) 数据传送过程如下：

① 主程序应先启动磁盘驱动器，然后继续执行主程序。

② 磁盘转速正常后，接口向 CPU 发中断请求。CPU 响应后，由中断服务程序向接口发送设备地址、主存初始地址和传送字数(2KB=1K 字)等，然后返回主程序继续工作。

③ 磁头找道，并等待磁盘转到访问的扇区后，通过接口向 CPU 发 DMA 请求，传送数据，因为传送 1K 字，所以总共发出 1K 个 DMA 请求。

④ 传送数据结束，接口向 CPU 发中断请求，CPU 响应中断后，由中断服务程序判断是继续传送还是停止传送，在这里应该是停止磁盘工作。

(2) 转速$=3000\text{r/min}=50\text{r/s}$，转一圈的时间$=\dfrac{1}{50}\text{s}=20\text{ms}$，每一磁道的存储容量$=8\text{KB}=4\text{K}$ 字。

磁盘写入 1 个字的时间$=\dfrac{\text{转 1 圈的时间}}{\text{磁道存储量}}=\dfrac{20\text{ms}}{4\text{K}}=5\mu\text{s}$，由于 CPU 执行一条指令的最长时间为 30μs，如果要待指令结束后响应 DMA 请求，则有可能产生错误。解决办法是让 CPU 在每个机器周期都能响应 DMA 请求，或者增加磁盘机内的数据寄存器数量。

42. 影响并行接口传输率进一步提高的原因是什么？

答：一般认为，并行通信的传输率高于串行通信，但当频率提高到一定范围后，并行通信就会显露以下缺点。

① 信号时滞。当信号通过一根电线或一个集成电路时会产生一些延迟。在并行方式中，从发送端同时发出的多位信息可能不会同时到达接收端，其中一些会迟于其他位，这被称为时滞。电线数量越多，传输距离越大，时滞现象越严重，从而限制了实际使用的时钟频率以及并行线缆的长度和数目。

② 串扰。并行线缆在物理上是紧靠在一起的，相邻的电线会相互产生干扰，而且信号

强度会随距离衰减(尤其在较高频率时),从而无法识别 1 和 0,造成误码率的增加。

③ 影响机箱内部的散热。并行方式使用的大连接器和很宽的带状电缆会堵塞机箱内部的气流,影响散热。

由于上述原因,影响了并行通信的时钟频率和传输率,不能满足高速网络和视频信号等对传输率的要求,促使 PCI 等并行总线和 SCSI 等并行接口向串行方向发展,产生了新的规范(原有的并行传输规范仍在继续使用),提高了时钟频率和传输率。

43. 列举出几种外部设备串行接口和并行接口的名称。

答:串行接口有 USB、IEEE 1394、SATA、iSCSI、PCI-Express、FC、infiniband 等。

并行接口有 ATA(IDE)、SCSI、PCI 等。

44. PCI、PCI-X 和 PCI-Express 是串行总线还是并行总线?

答:PCI 和 PCI-X 为并行总线。PCI-Express 为串行总线。

45. 简单介绍 DAS、SAN 和 NAS 三种存储体系结构。

答:(1) DAS(Direct Attached Storage,直接连接存储器)是存储设备通过 IDE、SCSI 等 I/O 总线直接连接到计算机总线上的一种存储结构,计算机可以直接访问存储设备。

由于网络的发展,促成了网络存储结构(SAN 和 NAS)的出现。

(2) SAN(Storage Area Network,存储区域网络)是以数据块为传输单位的存储网络,是建立在 FC(Fibre Channel)技术上,使用专用的通道交换机通过光纤将存储设备连接在一起,建立一个能够进行高速数据存储的专用存储网络,如图 10.5 所示。由于 SAN 的存储设备是用专用网络相连的,是与 LAN 分开的,与共享带宽的网络相比,具有更高的数据传输率。

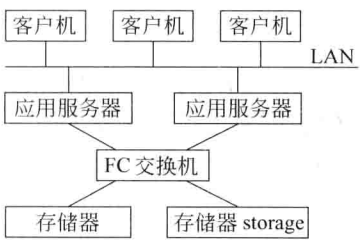


图 10.5 一个典型的 SAN 连接结构

根据存储网络所采用的传输协议以及传输介质的不同,SAN 有很多种实现方式,目前比较流行的是 FC-SAN 和 IP-SAN。

FC-SAN 通过 FCP(Fibre Channel Protocol)协议传输 SCSI 命令和数据,由于光纤通道的远距连接能力以及高速的传输能力,使得 FC-SAN 获得极好的性能,但是由于光纤网络的价格较高以及需要培训专业人员等因素影响,使得 FC-SAN 局限在大型计算机系统中使用。

IP-SAN 是采用 iSCSI 协议构架在 IP 网络上的 SAN,iSCSI 协议通过 IP 协议来封装 SCSI 命令,并在 IP 网络上传输 SCSI 命令和数据。与 FC-SAN 不同的是,在 FC-SAN 结构中,服务器的消息传送使用的是前端局域网,数据传输在后端的存储网络中进行;而 IP-SAN 存储系统使用 IP 网络进行消息传递和数据传输,因此 IP-SAN 可充分利用目前普遍使用的 IP 网络基础设施,建设费用便宜。另外 IP-SAN 的数据传输路径和 FC-SAN 的传输路径相比,只是光纤通道卡和 FCP 协议变为 iSCSI 卡和 iSCSI 协议,其他完全相同。

(3) NAS(Network Attached Storage,网络附加存储)是以文件为传输单位的存储网络,按照 TCP/IP 协议进行数据传输,一个典型的 NAS 结构如图 10.6 所示。其中 NAS 服

务器包括处理器、文件服务管理模块和多个存储设备(例如硬盘驱动器等),NAS 服务器可以看成专门用来在网络上提供文件服务的服务器。

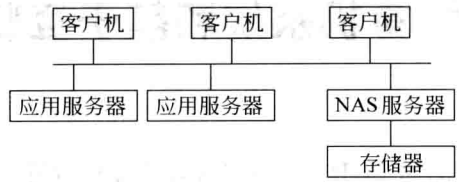


图 10.6　一个典型的 NAS 连接结构

第 11 章　计算机系统和基于互联网的应用

1. 选择题

操作系统和高级语言(例如 FORTRAN、Java)的编译程序是[A]。计算机加电后在显示屏上出现的图标和菜单是由[B]实现的。实现银行卡、交通卡功能的程序属于[C]的范围。实现某个游戏的程序属于[D]的范围。

存放用户的程序和数据的主存区域为[E],操作系统能访问的主存区域为[F],用户编写的程序不能直接访问的主存区域为[G]。

应用软件通过[H]管理计算机硬件各个组成部件的操作、控制人机对话和随机产生的中断事件。早期计算机尚无操作系统,用户应深切理解[I]和硬件,直接使用指令编写应用程序。

供选择的答案:

A、B、C、D、H、I:①系统软件;②应用软件;③操作系统;④指令系统

E、F、G:①系统管理区;②用户区;③系统管理区和用户区。

答:A——①;B——③;C——②;D——②;E——②;F——③;G——①;H——③;I——④

2. 请简述操作系统的作用。

答:应用程序通过操作系统来管理和控制计算机完成指定的任务。操作系统的重要功能如下:

(1) 管理计算机系统中的资源,包括所有硬件(处理器、存储器和外部设备等)和相应的软件,在硬件和操作系统的配合下顺利完成任务,并提高系统工作效率。

(2) 在多道程序运行环境下,保证资源调配合理、正确、安全。

(3) 向应用软件(或中间件平台)提供良好的接口,便于推广应用。

(4) 向使用者提供友好人机界面,以便于可靠、简便和有效使用计算机系统。

3. 简述操作系统管理处理机采用的"进程调度"概念及实现方法。

答:早期的计算机是单用户、单任务系统,处理机完全提供给一个任务使用,执行程序时所有的操作都是串行进行的,操作系统比较简单,但是处理机经常处于空闲状态,影响使用效率。

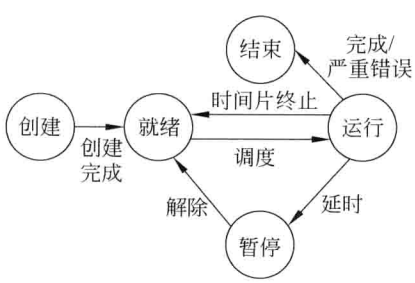


图 11.1　进程运行过程

现代的计算机支持单用户多任务或多用户多任务并行操作,在硬件配合下,操作系统负责将处理机轮流分配给一个任务使用,其执行过程用"进程"来描述。计算机系统宏观上可并行执行多个"进程"(多道程序),包括操作系统进程和多个任务进程,一个进程的运行过程如图 11.1 所示。操作系统在允许并行执行的多道程序数量范围内根据用户需求创建新的进

程，然后进入就绪阶段。处于就绪阶段的多个进程排队申请处理机的使用权，在硬件或软件配合下，实现进程调度，进入运行阶段。在就绪阶段的进程调度可根据优先级别、轻重缓急或时间片限制来实现，其目的是不遗漏一个进程的执行，并提高处理机的使用效率。

4. 在三层次存储器(cache—主存—辅存)的管理工作中，硬件和操作系统的分工原则是什么？虚拟存储器有何特点？

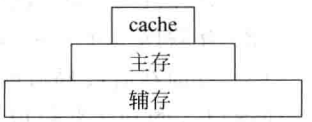


图 11.2　三层次存储系统

答：如图 11.2 所示，在三层次存储器中，其下层的存储容量大，实际上在一般计算机中，下层与上层容量的比例还要大得多。而且在上层的数据都从下层取来(复制)，例如在主存中的内容都包容在辅存中。在存储器系统中最早实现的是主存，为了提高存取速度和扩大存储容量分别采用了 cache 和辅存，所以 cache—主存层次一定由硬件实现，而主存和辅存之间数据的传送量(每次)较大，因此，传送时间也较大，允许操作系统在硬件配合下通过中断处理和 DMA 方式传送数据，上述内容是硬件和操作系统分工的基本原则。在设计时还与用户操作方便和成本等因素密切相关。

在虚拟存储器中，用户编写程序访问存储器可采用逻辑地址，逻辑地址比存储器实际的物理地址大很多，当应用程序访存时，由操作系统直接将逻辑地址转换成物理地址，如果访问内容不在主存中，由操作系统根据页式管理或段页式管理方法自动实现主存与辅存之间的数据传送，完成传送后，再返回应用程序继续执行，上述操作是在软硬件配合的情况下实现的。在非虚拟存储器系统中，用户程序中的访存地址应在物理地址涉及的范围内，如果超出此范围，则在用户程序中包含实现主存和辅存之间数据传送的程序。

5. 判断题(判断正确/错误，并指出错误的原因)。

(1) 在计算机的指令系统中，为了可靠、有效或快速实现操作系统，往往设置有一些特权指令。

(2) 硬件设计时，应尽量考虑操作系统的需求，以利于应用程序的快速、正确实现。

(3) 操作系统与硬件之间的连接已实现标准化，因此在不同型号处理机操作系统上运行的应用程序具有“可移植性”和“互操作性”。

(4) 输入输出(I/O)设备种类繁多，通过设备控制器(或称为设备接口)与 CPU(处理机)传输信号。设备控制器与两端(CPU 和设备)的连接线功能与布置的位置已实现标准化，因此操作系统中管理 I/O 设备的程序可应用于不同型号的处理机上。

(5) I/O 设备通过中断方式或 DMA 方式与 CPU 交换数据的全部过程都由操作系统独立实现。

答：(1)和(2)是正确的。

(3)、(4)和(5)有错误，分别说明如下。

(3)“可移植性”是指将操作系统或应用程序可移植到不同厂家的各种不同型号的计算机上使用。“互操作性”是指不同厂家在不同平台上开发的多种应用软件具有共享数据和协同工作的能力。由于芯片集成度不断提高和计算机应用范围不断扩大，不同型号处理机的指令系统无法统一，而且指令系统本身也在不断发展，因此建于指令系统上的操作系统很难移植。目前操作系统和应用程序之间的接口尚未统一，期望在上述两者之间增设一个平台，给应用软件提供一个标准化信息传送的平台，提高可移植和互操作能力。

(4) I/O 设备控制器一般设置在主机中，其与处理机之间尚未达成传送数据的协议，因

此操作系统中管理 I/O 设备的程序(BIOS)不能应用于不同的处理机上。

(5) I/O 设备控制器中为实现中断或 DMA 控制都设置相关的逻辑电路,而且 DMA 从传送数据开始直到一组数据传送完毕的操作全由硬件实现。

6. 判断题(判断正确/错误,并指出错误的原因)。

(1) 计算机网络系统的出现增加了硬件和软件系统的复杂性,但增强了计算机系统的功能。

(2) 互联网(Internet)有 5 个层次,每个层次功能都由网络操作系统实现。

(3) 目前在有线局域网中广泛采用的是以太网,根据信号传输速率的不同有传统以太网、快速以太网和千兆位以太网等,传输速率分别为 10MB/s、10MB/s 和 1000MB/s 等。

(4) 以太网中,各计算机结点是通过总线相互传输数据的。

答:(1)和(4)是正确的。(2)和(3)有错误,分别说明如下。

(2)在 5 个层次中,最高层为应用层,由应用程序实现,最低层为物理层,由硬件实现。中间 3 层为传输层(TCP)、网络层(IP)和数据链路层,是由网络操作系统实现的,其完成的功能总结为:将应用层送来的报文逐层转换成分组、数据报和帧,并填入数据报头;对传送来的数据进行检查,如果有错误,要求上层重发;确定接收数据的计算机结点地址,直接传送还是通过路由器传输数据。

(3)传输速率为 10Mb/s,1000Mb/s、1000Mb/s(1GMb/s)。

7. 选择题。

以太网卡是应用于[A]网络的产品,WiFi 是参加 WiFi 联盟的制造商生产的用于[B]网络的产品商标,移动通信是通过[C]网络传送数据的。

网络互连设备工作在 ISO/OSI 协议物理层,对一个局域网络内传送的信息起整形、放大作用的设备为[D]。工作在 ISO/OSI 协议数据链路层,连接 2 个或多个同类局域网的设备为[E]。工作在 ISO/OSI 网络层,可实现相同或不同类型网络互联并进行路由选择的设备为[F]。工作在 ISO/OSI 协议第 4～7 层,可实现不同类型网络互联的设备为[G],又称为协议转换器。工作在 ISO/OSI 协议第 2～7 层,联接在该设备各端口上的计算机(或其他)相互之间可直接独立传输信息,该设备称为[H]。

在一个通信信道上,可广义看作并行传输多个数据源的数据,这种功能称为[I],实现该功能的设备常用的有[J]。

供选择的答案

A、B、C:①有线;②无线

D、E、F、G、H:①网关;②网桥;③路由器;④交换机;⑤中继器

I、J:①多路复用;②频分多路复用器和时分多路复用器

答:A——①;B——②;C——②;D——⑤;E——②;F——③;G——①;H——④;I——①;J——②

8. 什么是客户机/服务器和浏览器/服务器?

答:客户机和服务器都是连接到网络上的计算机。

在客户机/服务器(C/S)系统中,如果一台连入网络的计算机向其他计算机提供服务,例如文件和数据的共享、数据的处理、应用目标的实施等,该计算机就被叫做服务器。客户机是与用户交互的计算机,提供用户界面,负责完成用户命令和数据的输入,具有一定的数

据处理能力，与服务器分工完成用户的要求。服务器接受从客户机送来的服务请求，进行处理后，将结果返回给客户机，再经过客户机的综合分析，将最后的结果提交给用户。

在浏览器/服务器(B/S)系统中，在客户端与用户进行交互的是一台能上网的非常简单的计算机或设备，在其上安装的应用软件为浏览器，用户的需求通过浏览器提交给服务器，经服务器处理后，将结果返回给客户端的浏览器，呈现给用户，所以完成用户需求的应用软件、数据库等都安装在服务器，由服务器实现数据库管理、数据处理和计算等功能。当系统功能升级时，主要的改进工作都在服务器端进行。

9. 请简述 C/S 系统和 B/S 系统的主要区别。

答：(1) 应用环境

C/S 系统一般是小型专用网络，各网之间可通过专用服务器提供连接和数据交换服务，客户机和服务器分工完成用户需求。B/S 系统一般建立在广域网上，客户端一般只有操作系统和浏览器软件，用户需求主要由服务器完成。客户端与服务器之间的通信比 C/S 系统频繁。

(2) 安全保障性

C/S 系统一般面向相对固定的用户群，对信息安全的保障性较强。B/S 系统面向众多用户，对信息安全的保障性相对较弱。

(3) 系统升级和维护

C/S 系统必须对客户机和服务器整体考虑和处理出现的问题以及提供整体的系统升级方案。C/S 系统一般有多个客户机，各个客户机可能使用不同的操作系统，加大了系统升级的难度。B/S 系统客户端的维护和升级工作极为简单。

10. 数据中心起什么作用？

答：数据中心是指能实现信息的集中存储、传输、交换、处理和管理的场所，它可以是单位内部设置的一个专门部门，为本单位的其他部门服务；也可以是为社会上众多用户服务的公用企业。从发展情况来说，数据中心的规模与服务内容在不断扩大。

对于大型的数据中心，为了保证其对用户的服务质量，需要对以下内容特别关注。

(1) 计算机系统。服务器、存储器、网络设备和电信设备等硬件配备齐全、性能良好，其规模满足众用户的服务需求。为了保证在系统运行发生硬件故障时能及时修复，配备充分的冗余设备对相关出错硬件实时进行切换。

同时，应配置满足用户需求的系统软件和应用软件。

(2) 数据。支持数据的海量存储能力，实现本地和异地的数据备份。当出现错误时，能快速地对数据纠错和修复。关注数据的检索、传输与处理时间。

(3) 可用性。能提供每年 365 天、每天 24 小时不间断应用服务。为此还要保证电力供应和冷却设备运行和不中断。

(4) 安全性。保证数据中心中的用户数据不泄露给他人，不被他人篡改。

(5) 灾难及恢复。配置有效的监测装置，以预防和抗拒人为的和自然的灾害威胁。考虑万一发生灾害后的恢复措施和备用场所。

上述内容表明，由大型数据中心提供服务的用户，仅需拥有极为简单的计算机设备(甚至仅是一台浏览器)，不必培养资深的应用人员和系统维护人员。

11. 选择题。

实现物联网的核心和基础是 $\boxed{\text{A}}$。通过各种信息传感设备，实时采集需要监控、连接和移

动的物品位置或活动情况，在互联网上传送采集到的信息并进行分析处理，从而组成了[B]。

物联网最早采用的信息传感设备是[C]及其读写器。

从技术架构上分析，物联网从底层向上可分为3层：[D]、[E]和[F]。[D]的主要功能是识别物体和采集信息。[E]主要负责感知层获取信息的传递和处理。[F]根据用户需求实现物联网应用。

供选择的答案：

A、B：①互联网；②计算机系统；③物联网

C：①全球卫星定位系统(GPS)；②射频识别标签(RFID)；③激光扫描器

D、E、F：①感知层；②网络层；③应用层；④物理层

答：A——①；B——③；C——②；D——①；E——②；F——③

12. 选择题。

云计算实现了一种基于[A]的新的IT服务模式，用户可通过软件[B]向云计算提供商提出服务请求。云计算提供商有能力根据用户提出的需求进行各种服务。云计算包括3个层次的服务，从基层向上分别称为[C]、[D]和[E]。云计算有两种基本类型，它们是[F]和[G]，并在此基础上组成[H]模式。

供选择的答案：

A：①大型计算机；②多指令流多数据流(NUND)计算机；③互联网

B：①浏览器；②基本输入输出系统(BIOS)

C、D、E：①软件即服务(SaaS)；②平台即服务(PaaS)；③基础设施即服务(IaaS)

F、G、H：①公有云；②私有云；③混合云

答：A——③；B——①，C——③，D——②，E——①；F——①或②；G——②或①；H——③

13. 请简述云计算3个层次服务的内容。

答：(1) 软件即服务(SaaS)。提供云计算的企业将应用软件统一部署在自己的服务器上，客户通过互联网向提供云计算服务的企业订购所需的应用软件服务或租用软件而不必购买软件，消除了客户购买、构建和维护基础设施和应用软件的需要。

(2) 平台即服务(PaaS)。提供云计算的企业将软件研发平台租给客户的一种服务，可加快客户的软件开发速度。

(3) 基础设施即服务(IaaS)。通过互联网向客户提供基础架构硬件和软件资源的模式，客户可租用服务器、操作系统、磁盘存储器和数据库等资源。

14. 简述公有云和私有云的主要区别。

答：私有云是企业(用户)创建云计算所需的基础设施和软硬件资源，以供企业内各部门共享，并提供服务，对数据和安全性提供有效控制。

公有云借助互联网，由提供商向社会上众多用户出租，收取租费。用户无须建立和购买应用所需的基础设施和软硬件资源，并能减少或免去人才培训和设备维修的费用。与私有云比较，公有云数据的保密和安全性较差。在我国，公有云尚在筹建中。

第 12 章　计算机系统硬件技术的发展及其实施基础

1. 什么是单指令流单数据流(SISD)计算机?

答:单指令流单数据流(SISD)计算机通常由一个指令控制部件、一个处理器和一个主存储器(包括交叉存储器)组成,每次执行一条指令,每次访存从存储器存取一个数据,但执行一条指令时可多次从存储器存取数据。前面各章讨论的计算机各个部件的依据就是单指令流单数据流计算机。当然它也是单指令流多数据流计算机和多指令流多数据流计算机的基础。

2. 什么是单指令流多数据流(SIMD)计算机?

答:单指令流多数据流(SIMD)计算机通常由一个指令控制部件、多个处理器和多个存储器组成。各处理器和存储器之间通过计算机系统内部的互联网络进行通信。在程序运行时由指令控制部件向各个处理器"播送"同一条指令,所有"活动的"处理器在同一时刻执行同样的指令,这就是单指令流(各处理器可由程序分别设置成"活动的"或"不活动的","不活动的"处理器不执行指令规定的操作)。各"活动的"处理器执行指令时所需的数据可通过内部网络到存储器取,也可从它本身专用的存储器中取得,各处理器所处理的数据是各不相同的,这就是多数据流。为各个处理器所专用的存储器又叫局部存储器。

阵列处理机系统就是一种典型的 SIMD 并行处理机系统。由在同一控制器(CU)控制下的多个处理单元(PE)组成(见图 12.1),在 CU 控制下,各 PE 对各自的数据并行地执行同一条指令所规定的操作。

阵列处理机的指令也是顺序执行的,指令分成两类,一类为标量指令(单数据流指令),由 CU 执行;另一类是并行处理指令,CU 把它同时送到所有 PE,那些处于"活动"状态的 PE 并行地执行指令。

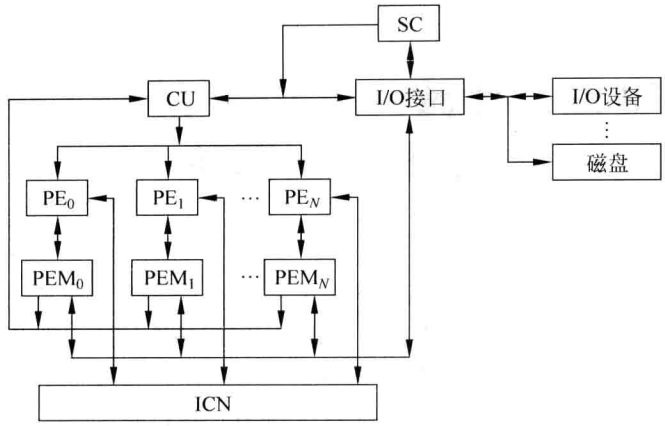
互连网络 ICN 是计算机系统内部传送数据的通路。

根据存储器的不同组成形式,有局部存储器和共享存储器两种阵列处理机结构,分别如图 12.1(a)和图 12.1(b)所示。在图 12.1(a)中,存储器分布在各个处理单元,成为处理单元的局部存储器 PEM,在图 12.1(b)中,共享的多体并行存储器 M_0,M_1,…,M_M 通过互连网络 ICN 与处理单元 PE_0,PE_1,…,PE_N 相连,通常 $M>N$。通过在存储器之间的合理分配数据,使各处理单元所需的数据能来自不同的存储器,而少受存储器冲突的影响。ICN 能提供多种连接方式,使每个 PE 与不同的存储器相连,用以传送数据。

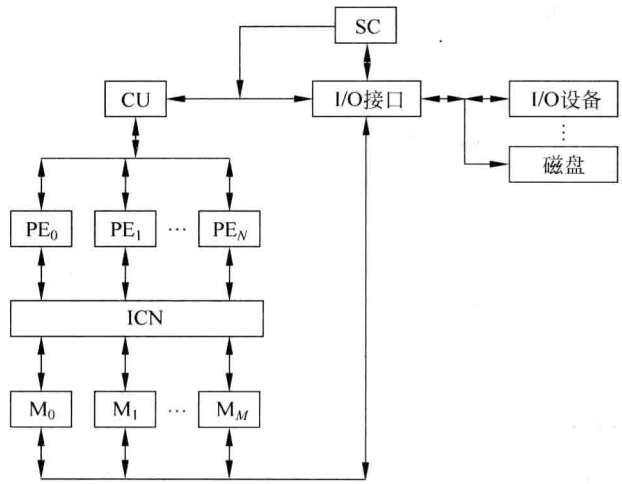
在系统中还设置有一个管理机 SC(内有存储器,存放常驻主存的操作系统),完成系统控制以及 I/O 操作等。

3. 什么是多指令流多数据流(MIMD)计算机?

答:典型的 MIMD 计算机系统由多台处理器(包括指令控制部件和处理器)和多个存储器组成,并有一个互连网络实现各处理器和各存储器之间的通信。每台处理器执行各自



(a) 局部存储器系统



(b) 共享存储器系统

图 12.1　阵列处理机结构

的指令，存取各自的数据(各不相同)。

图 12.2 是 MIMD 计算机(多处理机)的结构框图，P_0、P_1、…、P_N 是处理机(每个处理机包含 CU 和 PE)，通过互连网络共享主存储器。I/O 设备和外存储器 SM 通过设备控制器与互连网络连接，与处理机 P 共享主存。处理机之间也可通过互连网络交换信息，例如发中断等。

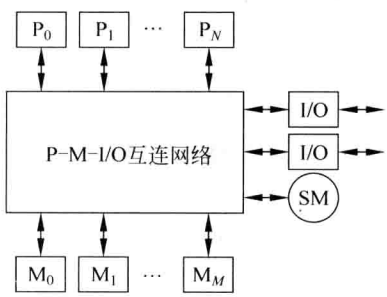


图 12.2　多处理机结构框图

另外还有一种每个处理机具有局部存储器的多处理机系统，由于处理机存取的数据主要在局部存储器中，因此减少了互连网络的工作量，比之共享主存的系统可连接更多的处理机。

目前的处理机都带有 cache 存储器。

在多处理机系统中，往往需要一台计算机(称为主机(host))作为系统管理之用。

4. 在图 12.1(b)和图 12.2 的共享存储器系统中，为什么将主存分成若干个模块而不是一个存储器？

答：因为这些多数据流计算机系统要同时存取多个数据，而一个存储器每次只能存/取一个数据，所以要将主存分为若干个模块，而且主存的模块数一般大于处理机数，以利于为处理机提供数据。

5. 填充题。

在当前各种应用计算机中发展最快的是[A]，它的核心是微处理器芯片，它沿着增加字长、提高[B]和[C]方向发展，为了加快与存储器之间的存/取操作，目前已将[D]集成在芯片内。

答：A——微型计算机；B——主频；C——集成度；D——cache。

6. 填充题。

(1) 根据微机体积的大小，可以分为[A]和[B]。

(2) 智能手机是兼有[C]和部分计算、娱乐功能的袖珍设备。它由[D]供电，没有内置磁盘，但有[E]功能。

(3) 嵌入到应用系统中的计算机称为[F]，它广泛应用于医疗仪器、工业控制和通信设备等方面，其中增长最快的市场是[G]等手持设备市场。

(4) 多媒体计算机是对多种媒体(文字、图形、图像和声音等)进行处理和传送的计算机，是将音像技术、计算技术和通信技术结合起来的计算机，为了解决存储和传送大量信息问题，需要对数据进行[H]和[I]。

答：A——台式机；B——便携机(或笔记本电脑)；C——通信；D——电池；E——存储；F——嵌入式计算机；G——手机；H——压缩；I——解压。

7. 提高计算机系统运算速度的方法(依靠硬件)有哪些？

答：提高计算机系统运算速度的方法很多，可归纳为两种：

(1) 改进器件工艺，提高集成度与工作频率。

(2) 改进计算机系统结构，并使各部件之间的速度匹配。

其中(1) 由微电子部门解决，(2)由计算机设计和实施部门解决。

改进计算机系统结构可用改进单机系统性能和采用多机系统来实现。

改进单机系统的方法如下：

(1) 增强指令系统功能，使之便于编程，并减少程序中的指令数，因而减少访存次数，提高效率。复杂指令系统计算机(CISC)为其代表。

简化指令系统功能，便于硬件实现，便于编译。精简指令系统计算机(RISC)为其代表。

综合技术、速度和价格等因素考虑指令系统，以获得更大的效益。

(2) 采用多个通用寄存器来暂存运算的中间结果，以减少访问存储器的次数。

(3) 采用多体交叉存储器和 cache，以协调 CPU 和存储器之间的速度匹配。

(4) CPU 和输入输出设备并行工作，以减少 CPU 等待和空闲时间。

(5) 操作重叠的流水线工作方式。

(6) 指令预取和乱序执行。

指令预取指的是提前将指令从主存取到 CPU，CPU 设置有指令栈以存放预取的指令，这样执行程序时可以从指令栈提取指令，减少取指时间。指令预取后还可以进行一些预处理，例如对无条件转移指令的处理或预取数据等。

乱序执行指的是对提前取出的指令进行分析，如果条件不足而不能执行时，可将后面的指令在不妨碍运算结果正确性的情况下提前执行。

(7) 采取多个功能部件，例如设置 1 个以上的 ALU 部件、乘法器等。

多计算机系统以并行处理的方法来提高运算速度。

8. 多处理机系统中的互连网络与计算机网络有什么根本不同？

答：多处理机系统中的互连网络用来将该系统中的处理器、存储器和 I/O 系统连接起来，这些部件的位置是很靠近的，一般都在一个机柜内。计算机网络是将地理位置不同，并具有独立功能的计算机系统通过通信设备和线路互相连接起来的网络。

9. 简述多处理机中使用的交叉开关。

答：交叉开关是多机系统中的一种互连网络。它使每个存储器模块都有一套总线同 p 个处理机和 n 个输入输出通道相连，如图 12.3 所示。如果存储器模块数 $m \geqslant p+n$，则在同一时刻，p 个处理机和 n 个输入输出通道都能分到一套总线，与 m 个存储器模块中的一个相连，因此能大大提高传输带宽和系统效率。

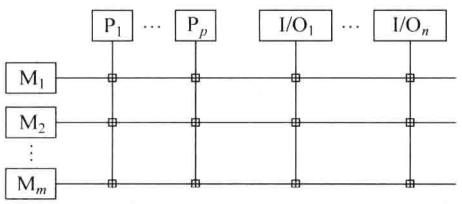


图 12.3　交叉开关

交叉开关结构中的每一个交叉接点都是一套开关，不仅要有传送数据和地址码的开关，还要有在多个处理机和 I/O 通道向同一个存储器模块发出访问请求时的排队能力，所以每一交叉接点上都有相当大的硬件设备量，当 $m=p+n$ 时设备量和 m^2 成正比。

10. 超级标量(superscalar)处理机和超长指令字(VLIW)处理机有何根本差别？

答：超级标量处理机和超长指令字处理机都是单指令流单数据流处理机。

(1) 超级标量处理机：在超级标量处理机中，配置了多个功能部件和指令译码电路，采取了多条流水线，可以同时执行多个操作。它可以同时从存储器中取出几条指令，并对这几条指令进行译码，把能够并行执行的指令同时送入不同的功能部件。超级标量处理机的硬件是不能重新安排指令的前后次序的，但可以在编译程序时采取优化的办法对指令的执行次序进行精心安排，把能并行执行的指令搭配起来。超级标量处理机能与同一系列的原有标量机器保持软件兼容。

上面讲到的标量是相对于向量而言的，向量处理机的一条向量指令可直接对一个数组进行加法或乘法等运算，向量数据是一个含有 N 个元素的有序数组，N 称为向量长度，向量中的每一元素是一个标量。

(2) 超长指令字(VLIW)处理机：VLIW 是一种单指令流多操作码多数据流的系统结构，由编译程序在编译时把多个能并行执行的操作组合在一起，成为一条有多个操作段的超长指令。CPU 每次从存储器取出一条超长指令，由这条超长指令控制 VLIW 处理机中多个互相独立工作的功能部件，每个操作段控制一个功能部件，相当于同时执行多条指令。VLIW 处理机的硬件是不能重新安排指令的前后次序的。VLIW 的指令不能与以前的指令保持兼容。

11. 什么是容错计算机？什么是热备份系统和冷备份系统？

答：容错计算机是高可靠性计算机，当其硬件在一定范围内发生故障时，该系统能自动采取一些措施，保证计算机继续运行。要使计算机具有这种能力，基本的方法是采用冗余技术，即在计算机内设置一些在正常运行时并不一定必需的设备或部件，让两个或两个以上的设备或部件完成同一功能或互相作备份。容错计算机应用于航空航天、电话交换、工业控制以及银行金融业务等联机事务处理系统中，可以采用两台计算机实现，又可分为两处理机运行不同的程序和运行相同的程序两类，分别称为“双机系统”和“双工系统”。

(1) 双机系统：两台处理机各有自身的存储器运行不同的程序，其中一台是主处理机，另一台是备份处理机。主处理机运行程序，每隔一定时间，设置一个检查点，把一些关键数据复制到备份处理机的存储器中。一旦主处理机发生故障不能继续工作，由备份处理机接替工作。主处理机在每个检查点处传送给备份处理机的数据应足以保证备份处理机可从最近的一个检查点开始执行主处理机中止的程序。备份处理机平时可执行其他程序。双机系统有时被称为“冷备份系统”，在作控制使用时，主处理机用于在线控制，备份处理机用作离线计算。

(2) 双工系统：两台处理机同时接收相同的输入数据，执行相同的程序，但只有主处理机输出数据，一般设置一套开关部件随时或定点对两处理机的运算结果进行比较，并控制主处理机输出。如主处理机发生故障，备份处理机可立即接替主处理机继续工作，其接替速度比双机系统快得多。因为不必返回到检查点执行程序，并立即由备份处理机输出数据，而原来的主处理机脱离系统进行维修。双工系统通常称为“热备份系统”。

由于互联网应用的蓬勃发展，系统的容错功能通常与网络紧密结合。

12. 什么是向量计算机？它对运算单元和存储器各有什么要求？

答：设有向量指令和向量数据表示的计算机称为向量计算机。它的运算单元一般采用流水线结构。当它处理一个数组时，对向量中的每个元素执行相同的操作，而且各元素间是互相无关的，因此流水线能以每个时钟送出一个结果的速度运行(输出第一个结果时间较长，因为需要做准备工作和通过整条流水线)。

为了存储器系统能及时提供数据，向量处理器一般有一个大容量的分成多个模块交叉工作的主存储器。

对向量的处理，要尽量避免流水线功能的频繁切换以及操作数之间的相关，例如有一个向量运算：

$$D = A \times (B + C)$$

式中 A、B、C、D 都是长度为 N 的向量。

为了不频繁改变流水线的功能，可考虑先执行所有元素的加法运算，然后再执行乘法运算。从数据相关方面考虑，也是这样做合理，因为加法与乘法运算产生数据相关，即加法得出结果后才能进行乘法运算，所以上述公式的执行顺序如下(用向量加和向量乘两条向量指令)：

$$B_i + C_i \rightarrow D_i \quad (i \text{ 为 } 1 \sim N)$$

$$D_i \times A_i \rightarrow D_i \quad (i \text{ 为 } 1 \sim N)$$

为了提高运算速度，在向量处理机的运算部件中可采用多个功能部件，并设置 cache 存储器。向量计算机内仍有一般的指令，以实现操作系统、数据库管理、程序执行流程控制和标量计算等。

13. 试写出在传统计算机上和阵列计算机上计算 $A=B+C$(其中 A、B 和 C 各由 n 个元

素组成)的过程。

答：在传统计算机上先算 $A(1)=B(1)+C(1)$，再算 $A(2)=B(2)+C(2)$，…，$A(n)=B(n)+C(n)$，这几步是顺序执行的，并通过循环程序来完成。在阵列计算机上，如有 n 个处理单元，则用一条加法指令控制 n 个处理单元同时执行加法，即第 1，2，…，n 个处理单元在同一时刻各自执行 $A(1)=B(1)+C(1)$，$A(2)=B(2)+C(2)$，…，$A(n)=B(n)+C(n)$。

14. 根据存储器组成方式的不同，多处理机可分成哪两类？哪一类可减少访问存储器冲突？

答：多处理机可分成各处理机具有局部存储器和各处理机共享存储器两类。在具有局部存储器的系统中，应将本处理机常用到的数据首先考虑安排在本机的局部存储器中，因此减少了访问其他存储器的次数，从而减少了访存冲突。

15. 台式微机、便携式计算机(笔记本电脑)有何主要差别？

答：便携式计算机相对于台式微机体积小、重量轻、省电，因此其主存、外存容量和连接的 I/O 设备一般比台式机少，但都有 LCD 显示器。在芯片方面，有专门为之设计的节能芯片，其性能可能逊于台式机芯片，一般带有电源管理功能。

16. 为微型机提供芯片组的目的是什么？请介绍微型机中的主机板的构成。

答：一台微机除了 CPU、主存和外部设备以外，还有很多电路用来连接和控制这些部件，例如中断逻辑电路、DMA 逻辑电路和接口电路等，如果不把这些电路集成在芯片内，那么将要用很多个中、小规模集成电路，这将增加微机体积，并降低整机运行速度，于是与微机配套的芯片组应运而生。主机板是 PC 硬件系统集中管理的核心载体。几乎集中了全部系统功能，能够根据系统和程序的需要，调度 PC 各个子系统配合工作，并为实现系统的管理提供充分的硬件保证。

主机板上安装有 CPU 插座、芯片组及某些外部设备的接口等，此外，还安装有若干个插座，允许插入主存条或连接设备控制器等，其中某些芯片(例如 CPU)发热量很大，芯片外壳接散热片。

17. 判断以下叙述的正确性，如有错误，请予解释。

(1) 开放系统是由 IBM 等大计算机公司首先提出并付诸实施的。

(2) 遵循同一标准，具有开放系统特点的计算机之间具有良好的“可移植性”和“互操作性”。

(3) 只要是开放系统的计算机，其操作系统和应用程序即可互相交换使用，而不必作任何修改。

答：(2) 是正确的。

(1)与(3)有问题，解释如下。

(1) 早期的计算机，其硬件和系统软件都由一家公司提供，不向外提供核心技术，例如 IBM 等公司都是这样发展起来的。随着计算机的发展，成立了许多新公司，为了争取市场和公司之间的相互配合，推出了遵循一定标准的开放系统产品，用户愿意购买基于开放系统的产品，因为可以选用市场上价格与性能合适的软硬件产品构成系统，也便于设备的更新、维修与扩充。

(3) 由于标准不是唯一的，有国际标准、国家标准和事实上的标准等，例如操作系统就有 Windows、UNIX 和 Linux 等。基于不同操作系统上的应用程序就不能相互通用。而且

同一类操作系统也有多种版本。

18. 对计算机进行性能测试的目的何在？

答：(1) 对现有的计算机进行测试可为购机者提供参考，选择价格合适而又满足要求的机器。考虑到计算机技术发展很快，应选择采用较先进技术的机器。

(2) 为计算机设计人员提供分析资料，有助于对系统结构和指令系统的确定。

(3) 为设计同一系列计算机的人员提供改进依据，找出关键问题，根据实际需要进行改进或扩充。

19. 判断以下叙述的正确性，如有错误，请予解释。

(1) 计算机的时钟频率反映了机器速度，主频越高，解题越快。

(2) 根据各类指令在程序中的执行频率计算出来的平均运算速度是一种很正确的评价机器性能的方法。

(3) 采用同一种型号 CPU 芯片的计算机，执行相同程序时，其速度是相等的。

(4) SPEC 基准测试程序全部选自实际的应用程序，因此它能完全正确反映计算机性能。

(5) 在某台机器上运行全部 SPEC 95 基准程序的速度是在 SPARC Station 10/40 工作站上运行速度的 n 倍，则这台机器的 SPEC 分数 $=n$。

(6) SPEC 分数目前被广泛用来评测计算机的速度，SPEC 分数值大的机器在运行任何程序时，都能获得比 SPEC 分数值小的机器更高的速度。

(7) 事务处理性能测试委员会(TPC)制定的测试标准用事务吞吐量(每分钟可完成多少个任务)和性能价格比作为评测指标。

(8) 用基准程序测试的结果(计算机性能)基本可信，但不一定能准确反映各方面实际使用的效果。

答：(1)～(3) 不完全正确，一起解释如下。

同一型号的 CPU 芯片，其时钟频率的高低反映了 CPU 速度，但是执行一条指令还需要与存储器配合工作，存储器的存取速度和容量都会影响 CPU 执行指令的实际时间，对系统来讲，磁盘等外部设备、操作系统和编译程序等都会影响机器的实际运行速度。

对于不同型号的 CPU 芯片，除了上述因素外，由于指令系统、数据字长和执行每条指令所需的时钟数等存在差别，因此主频不能作为计算机解题快慢的唯一衡量标准。

由于各个程序中，各条指令执行的频率不同，所以根据推荐的指令使用频率计算出来的平均运算速度并不适合所有程序，这种方法没有考虑指令系统功能、数据长度和存储器等因素，因此不能正确评价机器性能。

(4) SPEC 基准测试程序是选自某些实际的应用程序，但各应用程序差别很大，因此针对某个用户，不能仅根据 SPEC 测试结果来选用计算机，而且 SPEC 测试程序主要测试 CPU 和存储器的性能，还不够全面。

(5) 是正确的。

(6) 由于各个程序差别很大，因此 SPEC 的测试结果不适用于评价所有程序。

(7) 是正确的。

(8) 是正确的。

20. 什么是高性能计算和网格计算？

答：高性能计算(High Performance Computing，HPC)或超级计算(supercomputing)系统是指已达到千万亿次浮点运算速度的计算机系统。从1992年开始，用Linpack基准测试程序的线性代数方程包对各种超级计算机的性能进行测试，并且将测试结果按速率大小排名，形成了现在的全球TOP 500超级计算机的排名，每半年排一次，分别于当年六月在德国和十一月在美国举行的全球超级计算大会上发布。Linpack主要是解一个N元N次的线性方程组，它的计算主要集中在CPU的处理，其数据的读出写入较少，各进程之间的消息传递也不多，所以其测试结果差不多是理论峰值时间的同义词，具有局限性。虽有各种提议用更全面的测量方法，但尚未达成共识，估计在今后相当长一段时间内，Linpack仍是一个主要的超级计算机的基准测试工具。

传统的单机超级计算机系统在扩展性上都不太好，而且造价极高，基本上已不再采用，为便于扩展，以求得更高的性能，一般采用分布式多处理器的结构，以普通服务器或者双路SMP服务器来组成的集群系统，由于极高的性价比而受用户欢迎。在超级计算机TOP排行榜上已有超过半数的系统使用了集群系统，而且其所占的份额越来越大。

网格计算(grid computing)是利用互联网把分散在不同地理位置上的多个计算资源通过逻辑关系组成一台“虚拟的超级计算机”。这台机器把每一台参与其中的、包括个人计算机在内的计算机都作为自己的一个“结点”，成千上万个这样的结点连接起来，就组成了一张“有超级计算能力的网格”。而每一位将自己的计算机连接到网格上的用户，也就“拥有了”这台超级计算机，可以随时调用其中的计算和信息资源，在应用层面上实现所有资源的全面连通，包括计算、存储、软件、数据、信息和仪器设备等。连接到网格上的计算机除了能获得其他计算机的服务外，也允许其他计算机调用自己的资源。打个比方，网格环境下的计算机好比小电站，把这些小电站拼网使用而构成一个公用电网，因此网格最终应该是一种“公用设施”，由网格应用服务商提供服务，其作用与电力公司类似。

21. 请简单介绍片上系统。

答：片上系统(System on Chip，SoC)是根据应用需求将标准化的电子功能模块和新设计的功能模块在单一的集成电路芯片上制作的完整系统。这种标准化的电子功能模块称为知识产权核或IP核(intellectual property core)，它不是为当前制作的芯片专门设计的，具有可在许多芯片中重复使用的特点。SoC可以包含许多原来只有在印制电路板(PCB)上才可以放置的器件，这样一来，原来需要很多器件、尺寸很大的PCB板就因为SoC而变得十分小巧。如今由于芯片集成度的提高，在一个芯片中已经可以集成微处理器、数字信号处理器(DSP)、逻辑电路、存储器和外围功能模块等，甚至可将数字电路、模拟电路和射频电路集成在一个芯片中。由于将原来的多个器件都放在了一个芯片中，器件之间的物理距离大大缩短了，因此可以提高性能、减少尺寸、降低功耗。与全部独立设计的芯片相比，可以降低成本，并快速推向市场。在SoC中使用最为广泛的微处理器(即IP核)为ARM。

SoC的胜出得益于微电子工艺的进步而导致的芯片集成度的提高，以及ASIC(Application Specific IC，专用集成电路)设计技术的成熟，然而SoC的发展必须不断地解决设计、集成、验证及测试等方面的难题。

22. 选择题。

(1) 指令系统设计主要由$\boxed{A}$进行。设计同一系列不同型号的计算机时，新机器应包括

原有机器的全部指令，以达到 B 的目的。

(2) 目前以下这些工作主要依靠计算机自动完成(当能提供适当的软硬件环境时)：C、D、E 和 F。

供选择的答案：

A：①硬件设计人员；②软件设计人员；③软硬件设计人员共同参加

B：①硬件兼容；②软件兼容；③软件向上兼容

C、D、E、F：①指令系统设计；②指令系统模拟；③从指令系统到硬件实现；④逻辑模拟；⑤电路模拟；⑥印制板布线

答：A——③；B——③。

指令系统设计时，软件人员根据指令在程序中的运行情况提供设计依据，硬件人员根据指令功能实现的难易程度提供设计依据，共同商量确定。

软件兼容的计算机，其指令系统是完全相同的。如果新机器包括原机器的全部指令，而且扩充了指令系统，只能做到软件向上兼容，即原机器上的全部程序可以在新机器上运行，但为新机器设计的程序则不一定能在原机器上运行。

C——②；D——④；E——⑤；F——⑥。

指令系统模拟见第 23 题。

逻辑模拟用以模拟计算机各个部件的逻辑正确性，电路模拟是将逻辑图中的各个组成部分用电路来替代后模拟其功能，以确定其正确性。印制板布线要求布线合理，且在同一层上的线不能相交(相交则短路)，目前一般都采用多层布线(印制板由多层板压制)。

23. 指令系统的模拟和指令系统的仿真有何区别？如何实现？

答：指令系统的模拟是在一台计算机上用程序来实现另一台不同指令系统计算机的每条指令功能，这通常应用于设计新的指令系统时，在现有的机器上来测试新指令的功能，这是计算机辅助设计的一个例子。其实现方法如下。

在现有的计算机上开辟一个专用的主存区，新机器中的硬件(主要是寄存器和触发器)各用主存区的一个单元来替代，假设新机器的运算部分中有 A、B、C 3 个寄存器、1 个 ALU 部件和 8 个通用寄存器，另外还有根据运算结果所置的状态位 N、Z、V、C。当然在控制部分中，程序计数器 PC 和指令寄存器 IR 也是少不了的。为讨论简化起见，其他部件在本题中就不再考虑。这样在主存区可将上述的硬件安排如图 12.4 所示。其中每个状态位仅占存储单元中的 1 位。

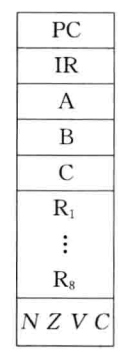


图 12.4　模拟硬件的主存区

现讨论模拟一条加法指令的过程。

加法指令：ADD　R_A，R_B。

R_A、R_B 为 8 个通用寄存器中的任意 1 个，运算结果存放在 R_A 中。并置状态位。

执行步骤如下(在现有的计算机上执行，PC、IR 等都在主存区内)：

① PC→主存，读(从主存取指令)，指令→IR。

② PC+1→PC。

③ 分析指令操作码，转加法子程序（每条指令都编好一段子程序，指令功能由子程序实现）。以下是加法子程序完成的功能。

④ $(R_A) \rightarrow A$。

⑤ $(R_B) \rightarrow B$。

⑥ $A+B \rightarrow R_A$。

⑦ 如果运算结果为负数，置 $N=1$，否则为 0；

如果运算结果为零，置 $Z=1$，否则为 0；

如果运算结果溢出，置 $V=1$，否则为 0；

如果运算结果有进位或借位，置 $C=1$，否则为 0。

上述步骤可用汇编语言或指令系统描述语言实现，后者是在现有计算机上为模拟新指令系统而专门编写的软件，新指令系统的设计人员用指令系统描述语言编程比汇编语言容易，因为它比较接近于高级语言。

有关指令系统描述语言的举例可参考第 24 题。

指令系统的仿真是在一台计算机上用微程序来实现另一台不同指令系统计算机的每条指令，这通常应用于新机器要运行不同指令系统的旧机器程序的场合，这相当于在新机器的微程序控制器中要安排实现两套指令系统的微程序，因此提高了计算机的成本与复杂性，除非有特殊需要，一般计算机都没有这种功能。

指令系统的模拟是完全用软件实现的，效率低；仿真是用硬件实现的，效率高，因而决定了它们的应用领域不同。

24. 请举例说明使用指令系统描述语言实现加法指令的软件。

答：下面用 Sun 公司设计的指令系统描述语言，对 Sun 公司的 SPARC 计算机加法指令（ADD、ADDx、ADDcc）进行描述。在设计新机器时，需要对每一条指令功能和实现过程进行详情的描述，通过模拟方法验证指令系统的正确性。

```
add instruction  (加法指令)
  if(trap=1) then                          ;trap=1,表示执行本指令以前已有故障或中断请求
    execute trap                           ;执行 trap 程序
  next;                                    ;下一步执行
    instruction←memory_read(pc)            ;从存储器读出指令
  next;
    if memory_exception then               ;如果存储器有异常
    execute memory_exception               ;执行 memory_exception 程序
  else (
    dispatch instruction                   ;发送指令到执行部件,准备执行加法操作
    pc←npc                                 ;pc 和 npc 为当前指令和下一条指令的地址
    npc←npc+4                              ;相邻指令地址相差 4 字节
    )
  next;
    operand2: if i=0 then r[rs2] else sign-extend(simml3);
              ; 如 i=0,第 2 个源操作数在 rs2 所指定的寄存器中,否则是扩充符号的立即数 simml3
    if (ADD or ADDcc) then
      result←r[rs1]+operand2;              ;执行加法操作
```

```
    else if (ADDx or ADDxcc) then
        result←r[rs1]+operand2+C;          ;执行加法操作
next;
if(rd<>0) then                             ;如果 R≠0,运算结果送 rd
    r[rd]←result;                          ;rd=0 的寄存器,内容恒为 0
if (ADDcc or ADDxcc) then(                 ;根据运算结果置 N、Z、V 和 C
    N←result<31>;                          ;<31>表示第 31 位,即最高位
    Z←if result=0 then 1 else 0
    V←(r[rs1]<31>and operand2<31>and not result<31>)or
        (not r[rs1]<31>and not operand2<31>and result<31>);
    C←(r[rs1]<31>and operand2<31>)or
        (not result<31>and (r[rs1]<31>or operand2<31>)
);
```

在本例中,前面的一些操作,诸如检查是否有中断和异常,取指令,程序计数器+4(按字节编址)等是与具体指令无关的公共操作。

25. 在实现微程序设计自动化时,怎样表示用微汇编语言编写的微指令?微汇编语言的编译程序完成什么工作?

答:用微汇编语言编写的微指令,用该指令所需要的控制信号来表示。在本书第 6 章图 6.7 的指令执行流程图中,每个机器周期 T 所需完成的操作即是一条微指令所需完成的操作,例如在 T_1 完成的微指令需要的控制信号为读(Read),MDR→IR,PC+1→PC。该微指令的微汇编语言表示形式即为 Read,MDR→IR,PC+1→PC。微汇编语言编译程序的作用是将其转换成微指令的二进制编码,这样微程序编写人员编写的微汇编语言程序比较容易检查,而编译程序的功能也比较简单。

26. 设有一全加器,其输入信号为 IN1、IN2 和 CIN(进位)。已知该全加器由两个半加器组成,其输出信号为 SUM(和)和 COUT(进位)。其逻辑图如图 12.5 所示。请用 VHDL 语言进行描述。

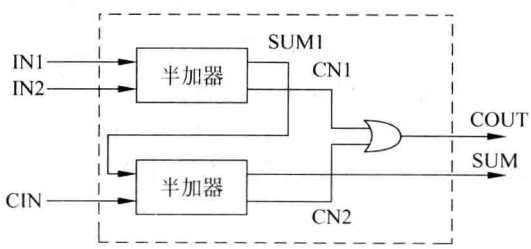


图 12.5 全加器逻辑图

如果要测试该电路的正确性,应如何确定其激励信号。

答:图中 SUM1 为半加器输出的半加和,CN1 和 CN2 为半加器输出的进位信号,该全加器用 VHDL 语言描述如下:

```
ENTITY full-adder is
  port (IN1, IN2, CIN : in bit;
        SUM, COUT   : out bit);
  end full-adder;
  ARCHITECTURE behavior of full-adder is
  begin
    process (IN1, IN2, CIN)
        variable SUM1, CN1, CN2: bit;
    begin
```

```
        SUM1<=IN1 xor IN2;
        CN1<=IN1 and IN2;
        SUM<=SUM1 xor CIN;
        CN2<=SUM1 and CIN;
        COUT<=CN1 or CN2;
    end process;
  end behavior;
```

其中 bit 表示二值(0,1)数据类型,xor 表示异或(半加)操作,and 表示逻辑乘,or 表示逻辑加,<=表示将其右边的结果赋给左边的变量。

表 12.1 全加器的激励信号与输出

激励信号			输出	
IN1	IN2	CIN	SUM	COUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

如果要测试电路的正确性,在 3 变量输入的情况下应加上 8 组激励信号,即 IN1、IN2、CIN 从 0,0,0;0,0,1;…,到 1,1,1。激励信号与输出的关系如表 12.1 所示。

如果输出不符合表中的内容,说明逻辑图设计有错误,改正后,再模拟,这样就有可能把错误消灭在设计阶段,同时也可免去对实际模型进行测试验证的工作。目前大规模集成电路非常复杂,要进行模型实验是非常困难的。

27. 某公司设计一台新指令系统计算机,但仍想运行该公司原来机器上运行的程序,一般可采用什么方法?

答:如有高级语言编程的源程序,可在新机器上重新编译一次,因为高级语言比较标准,而且与机器的具体实现(硬件)关系不大。

如要运行原机器上的汇编语言程序或原二进制代码程序,则很困难,因为这两种语言与机器的具体实现有关,将原来机器上的汇编语言程序转换成在新机器上运行的程序很难,直接进行指令(二进制代码)的转换也不可能。一般采用模拟或仿真的方法(参见本章第 23 题)。

28. 什么是 ASIC?

答:ASIC 是硬件设计人员根据实际需要而专门设计的集成电路芯片,如果没有这些芯片,那么在计算机系统中的某些逻辑部件要采用好几片中小规模集成电路,这将增加计算机的体积,增加电路间的延迟时间。ASIC 芯片除了能满足一定的使用要求外,由于其内容不公开,所以还具有保密作用。

为了使设计者能在短期内开发出高质量的 ASIC 芯片,电子设计自动化(EDA)系统已有很大发展,它能提供具有一定功能的元件库,供设计者采用,并允许在实际制作芯片前,对相当规模的系统进行认真详细的分析,以期达到一次设计投片成功的目标。

第二部分

实验、设计与测试

第 13 章　计算机硬件实验概述

计算机组成与结构课程是计算机专业学生一门很重要的专业课程和专业基础课程，通过该课程的学习，要求学生掌握一台计算机的基本结构，从而为后续课程的学习打好基础，然而计算机本身是很复杂的，它涉及的知识面很广，实践性很强，因此为它设置实验课是很必要的。

学生毕业后，无论是继续深造或参加计算机硬件、软件或应用相关行业的工作，如果通过本课程实验获得切实的硬件基础，将顺利地和有创造性地进行今后的学习和工作。

当前计算机的应用已广泛普及到各个领域，例如航天航空、军事、科研、计算、通讯、工业控制、计算机辅助设计、计算机辅助制造以及日常生活、娱乐等各个方面。因此在教材和本书中考虑相关专业（通信、微电子和自动控制等）学生的学习需求，为他们的后续课程学习和今后的工作打好基础。

本部分讨论的重点以计算机的硬件为主。但也涉及相关的软件。本部分仅从使用的角度出发对用到的软件作简单的介绍。深入的内容将根据各专业的应用与发展在后续课程中解决。

本部分讨论的内容不受具体的实验装置限制，但对实验装置提出了要求及解决方案。对开设实验课的学校来说，可以自制实验设备，也可外购，这两种办法各有利弊。自制设备是对教师的一种锻炼，尤其是对缺乏实践经验的教师来讲更是如此，通过这种锻炼，对提高教学质量有利；外购设备一般比较正规，还可能附有一些基本的管理程序等，甚至是一台安装 EDA（电子设计自动化）软件的计算机，实验内容比较丰富，但价格可能不便宜，而且有些设备对硬件电路的基本知识不够重视。各校可根据自身情况采取相应的办法。

1. 实验目的

（1）加深对计算机各个部件及整机的理解，巩固课堂上学得的知识。

（2）掌握实验方法与技巧，通过实验提高对计算机的设计、调试、诊断和运行稳定性的认识。

（3）掌握基本仪器的使用方法及测试技巧。

（4）实验内容宜少而精，但要培养学生从实际出发思考问题和解决问题的能力和举一反三的能力。计算机技术发展很快，只有掌握住本质的东西才不致迷失方向，才能正确使用各种计算机辅助设计工具。对实验中产生的问题，不仅要寻找解决办法，还要了解为什么会产生这些问题，以免在其他地方重犯这种错误。

2. 实验内容

实验安排应符合从简到繁的原则。

可先进行各部件的实验，如运算器、控制器、存储器和外部设备接口等。由于各个部件之间有地址、数据和控制信号的联系，比较复杂，可先将部件之间的联系隔离，而由实验装置提供这些信息。在实际设计和试制新计算机时，也是在各个部件的设计和调试完成后再连接在一起进行联合调试，当然各部件之间连接线的规范是预先约定的。

在进行部件实验之前，可以先进行一些更为简单的实验，从而熟悉一些基本仪器的使用方法。最基本的仪器是示波器和电压表，其他例如逻辑分析仪等也是经常使用的，但价格较贵。一旦掌握了基本的实验技术，当有更先进的数字仪器时，通过阅读仪器说明书是不难掌握其使用方法的。

整机实验考虑在实验装置（或称为实验机）上执行一段程序，因此要设计实验机的指令系统，设计取指令和执行一条指令的流程及其实现方法（微程序控制和组合逻辑控制）。指令系统可设计得比较简单，重点是了解一台最简单的计算机的整机结构和计算机执行程序的基本原理及其实现。

计算机技术（硬件）发展很快，本实验的目的是打好基础，掌握提出问题和解决问题的方法，在今后的工作中边学习边实践边创作。我国自主设计的 CPU 芯片已得到应用，今后将需要更多的人才来从事硬件设计和制造工作。

3. 第 14 章和 15 章内容

第 14 章在简单的实验装置上进行实验，配合教材内容，理解组成计算机的元器件、部件到整机的实现原理、设计方法和测试方法。

第 15 章在安装有 VHDL（硬件描述语言）和其他 EDA 软件的计算机上进行实验与设计的方法。

第 14 章　计算机的硬件实验、设计与测试技术（以硬件基础为主）

14.1　实验装置

实验装置基本上由两部分组成：控制板与线路板。控制板（或控制面板）上安装有开关、按键、指示灯和控制电路等，主要用于控制实验装置的运行状态、输入数据、观察运算（或操作）的结果等，是为配合人机交流而设置的。线路板上主要安装被测试的器件、部件和整机线路，根据需要各部件可独立工作（分调）或联机工作（联调）。

14.1.1　控制板上安装的元件

1. 开关

1）电源开关

当合上电源开关后，由于电源输出端接入大容量电容器或者还有其他原因，送到装置的直流电压（假设是＋5V）是逐渐上升的（见图 14.1）。在电源电压上升到一定数值，且还不能使实验装置中的电路工作时，产生一个总清$\overline{\text{reset}}$（低电平）信号，其持续时间等于＋5V 电压上升到稳定值的时间加上完成总清操作的时间，总清信号的目的是使装置中的触发器处于初始状态，其作用与计算机中的$\overline{\text{reset}}$信号相同。

供给装置的＋5V 电源的电流不是恒定的，当装置中的触发器及门电路状态变化时，其电流也随之变化（在状态跳变时，电流最大），当脉冲电流通过导线时，导线有一定的阻抗，所以在电源线上产生噪音（干扰），为了减少它的影响，一般采取的措施有：＋5V 电源线加粗，地线用铜皮替代，加电容（除了加强集中的大电容外，还在每一插座的电源与地之间，甚至在每一芯片的电源引入端与地之间加上电容，并选择能对高频起作用的电容）等。

2）数据开关

当运算器与存储器脱离工作联系时（计算机的部件分调即属这种情况），操作数由面板上的开关提供，开关数量与字的长度相等，例如字长 8 位，开关数也为 8 个。如果考虑到需要 2 个操作数，则安排两组开关（每组 8 个）。开关的输出有两种状态："1"和"0"，其连接的方法如图 14.2 所示。

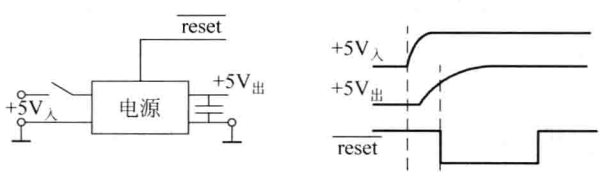


图 14.1　加电时的总清$\overline{\text{reset}}$信号

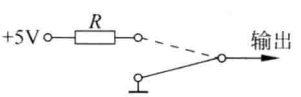


图 14.2　开关的连接方法

图 14.2 所示为输出为"0"时的情况，当开关打到上面时，输出接近于＋5V。当开关作为数据或控制信号时，一般都要串接一个几 kΩ 电阻 R 再连接到＋5V。接电阻的原因是由于＋5V

电源内阻很小，防止因开关安装或质量有问题造成电源短路。在此处，各开关可连接到一个公用电阻上，电阻的另一端接+5V 电源。

3）指令开关和断点控制开关

当运算器独立调试时，还需要根据具体情况设置一些开关作为产生控制信号之用。

当运算器与控制器联合调试时，为了逐条调试每条指令，在控制面板上设置一组指令开关，用来取代指令寄存器 IR，或将指令开关的内容送入 IR。

当测试整机性能时，往往设置断点控制开关，当正在执行的程序指令地址与断点开关所示的地址相等时，测试程序停止工作，并由指示灯显示操作结果，与人工预计的结果比较以判断操作是否正确。

4）工作方式开关（连续工作或单步工作）

当计算机或实验装置执行程序有错，而一时难以确定错误原因时，可考虑令程序逐步运行，并设定观察点，每走一步（例如发一个时钟脉冲）停下来，测量观察点的状态是否与预期的状态相同，从而找出错误的原因，是设计错误？还是器件损坏？或者连线有误？为此在控制板上设置一个工作方式开关，将实验装置设定在连续工作或单步工作状态。根据设计者的意愿，可进一步将单步工作方式再分成单时钟和单条指令两种，前者是每发出一个时钟脉冲机器就停下来，后者是每执行完一条指令机器停下来。单步操作能验证实验机逻辑设计的正确性，但信息（数据或控制命令）通过电路时会产生延迟，因此还需进一步考虑时序问题。一般还要让实验机按设计的时钟频率进行连续工作来解决时序问题。

2. 按键

与开关不同，当按下按键的手松开时，按键会恢复到原始状态。

每台计算机内都有一个时钟脉冲，在该脉冲作用下计算机才能运行。计算机的启动与停止一般是由人工控制的（执行停机指令例外），人工启停与时钟之间的配合情况（时序）是随机的，加电后机器的时钟 CLK 不断发出，当启动后，允许该时钟通行，产生 CP，发往计算机的各个部件；当停机后，则将该时钟封锁，计算机各个部件得不到时钟脉冲，也就停止工作了。启停控制电路就是用来控制时钟的通行与封锁，其关键问题是要保证发出的每一个时钟脉冲都是完整的。对于实验装置，启停更为频繁，对启停控制电路的要求（发出完整的时钟脉冲）与计算机相同。

一般在控制板上设置以下按键。

1）总清（$\overline{\text{reset}}$）按键

前面讲到，在计算机加电时，会自动产生$\overline{\text{reset}}$信号。另外，当计算机或实验装置处于一种混乱状态时，人工按一下总清按键，依靠它产生$\overline{\text{reset}}$信号，使装置处于初始状态，再重新启动工作。

$\overline{\text{reset}}$信号破坏了计算机的当前运行状态，重新启动后不能继续执行程序，而是要从头开始或从设定的断点开始执行程序，因此计算机在执行程序时不要轻易按此按键。实验装置不要求正常执行程序，因此无此限制。

2）启动按键

总清后，按启动按键，此时如果工作方式开关处于连续工作状态，实验装置将不断发出

时钟连续工作；而处于单步（单脉冲）工作状态时，启动一次只能发出一个时钟脉冲。

3）停止按键

当实验装置运行时，按下停止按键，将封锁住时钟。

启动或停止时，皆要保证发出的时钟脉冲的完整性。（参考教材第 2 章 D 触发器中的维持阻塞电路）。

需要注意的是：在做开关与按键动作时，都可能产生抖动，因而引起多次高低电平的变化，采用图 14.3 的 R-S 触发器电路，可防止抖动的影响。在图中，S 是人按下启动按键时产生的信号，R 是人按下停机按键时产生的信号，这些信号在不起作用时都处于高电平。如果按下 S 键时产生抖动，那么当 S 第一次输入低电平时将 R-S 触发器置为“1”（即门 2 输出高电平），以后的抖动不会对 R-S 触发器的状态产生影响。因此使用 R-S 触发器输出来控制 CP 时钟的产生与消失。

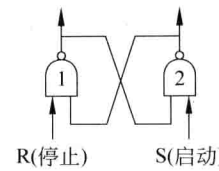


图 14.3　消除抖动影响的电路

3. 指示灯

在机器运行停下来时，为了检查操作是否正确，需要对某些观察点进行测量，例如进行加法运算，相加结果保存在寄存器中，如果要确定该结果是否和预期的结果相等，则需要将该寄存器的数据读出来，一般为该寄存器中的每一位配置一个指示灯，从灯的亮和暗就可知道每一位的数据值是 1 还是 0。

另外，在单步工作时，往往希望将当前 PC 的内容和指令的操作码显示出来。其他诸如是否已加电、运算器中的状态位（N、Z、V、C）等也往往用指示灯显示出来。

需要注意的是，指示灯的电流比较大，用一般的集成电路驱动它功率不够，所以每一个观察点需外加一级驱动电路，以保证有足够的电流流过指示灯。

如果要测量某点电平，而又未接指示灯，此时可用万用表（或电压表）测量，也可用示波器测量（直接输入）。需要注意的是接入测量仪器的瞬间对被测电路来说有可能是一种冲击，影响观察点的电压值。如果测量触发器的输出有可能会改变触发器的状态。

4. 存储器

某些实验装置具有将控制面板上设置的数据写入存储器以及将存储器的内容读出显示的功能，为此要给出存储器的地址和读写命令。这样就可以将一小段程序逐条从面板上输入到存储器，从而可以方便地调试指令功能。

14.1.2　实验装置上安装的电路以及测试仪器的使用方法

1. 实验装置上安装的电路

1）预先安装的供实验使用的信号

必备的信号有$\overline{\text{reset}}$（总清）、CLK（加电位就产生的连续不断的时钟）。CP（受控制的单脉冲和连续脉冲）以及实验所需的机器周期电位信号。

后者也可考虑通过学生实验而产生。

2）小容量存储器

小容量存储器能从控制板输入被测试的指令和数据的若干个字。

3）连接孔和观察点

安排好实验线路和控制板信号的连接孔和学生实验时的观察点。进行实验时，可能会

遇到事前未准备好的连接点和观察点，务必注意操作，不要产生短路连接情况。

2. 测试仪器的使用方法

建议实验时采用万用表(或电压表)和示波器，要求输入阻抗大，否则测试时成为被测电路的负载而影响正常工作。采用双线示波器便于观察相关观察点波形的时间关系。示波器的具体使用方法在 14.2.1 节中介绍。

14.2 实验内容的设计

实验可考虑按下述 3 方面进行：

(1) 逻辑电路的设计和调试。

(2) 根据计算机各个部件的特点设定实验内容。

(3) 指令系统执行流程及调试。

由于实验时间的限制，实验装置应预先准备好，安装好部分元器件，并具有一定的功能。实验内容不可能面面俱到，主要由教师根据具体的实验条件指导学生进行实验，掌握工作原理和实验方法，培养独立分析问题和解决问题的能力。以下介绍的内容仅供参考，希望能起到抛砖引玉的效果。

14.2.1 部件实验

1. 逻辑电路的设计和实验(测试)

1) 执行一条指令的时钟周期

实验要求：假设某计算机执行一条指令需要 8 个时钟周期(每条指令都相同)，设计产生 8 个时钟周期的电路。

电路设计：建议用 3 个 D 型触发器组成计数器，计数器输出送到译码器，译码器输出即可得到 8 个周期信号。具体设计较简单，在这里不再讨论，主要介绍一些测试方法及测试波形的分析。

测试波形：假设计数器按一般二进制方式进行，其计数顺序为 000，001，010，011，100，101，110，111，000，…。则其逻辑框图与用示波器观察的理想波形如图 14.4 所示。然而由

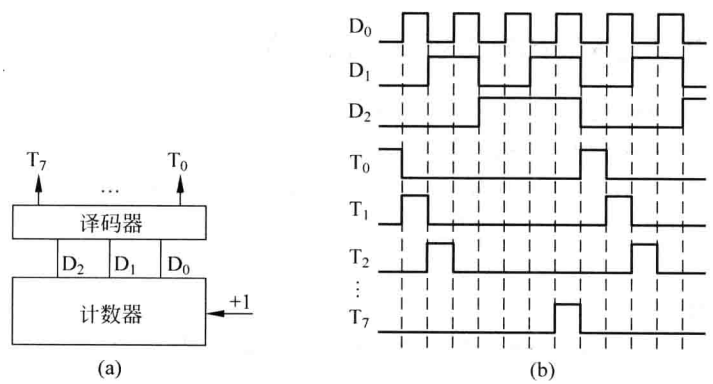


图 14.4 计数器译码器的逻辑框图与时序图

于各个触发器和门电路的上升沿、下降沿及延迟时间有差异，受其影响，有可能使译码器输出线上产生尖脉冲，如图 14.5 所示。例如，当计数器($D_2D_1D_0$)从 001 转变到 010 时，如果 D_0 的变化迟后于 D_1 的变化，则在该瞬间会出现 011，于是在 T_3 线上出现尖脉冲；如果 D_1 的变化迟后于 D_0 的变化，则在该瞬间会出现 000，于是在 T_0 线上出现尖脉冲。假如仅是这一原因，则在该瞬间不可能同时在 T_0 线和 T_3 线上产生尖脉冲，也不可能在其他线上产生尖脉冲。否则可能是其他原因，最常见的是电源线和地线上有噪音，或导线之间的感应造成的。同理，当计数器从 011 转变到 100 时，有可能在 T_0 或其他译码器输出线上产生尖脉冲(与逻辑设计和电路性能有关)。

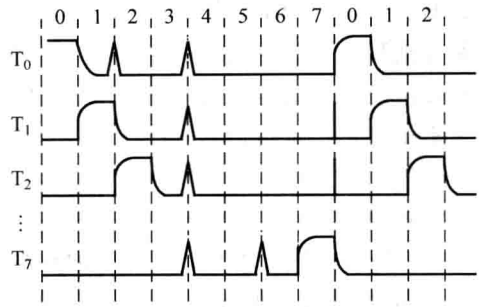


图 14.5　译码器输出产生尖脉冲的时序图

测试方法：完成线路连接并检查无误后加电，首先采用单步工作方式观察指示灯是否按正确计数顺序显示，然后用示波器观察计数脉冲波形及计数器 3 个触发器的状态是否与图 14.4 描述的相同。然后再观察译码器的输出。如果波形不对，则分析其原因，先排除触发器的问题(如有错的话)，再排除译码器的问题(如有错的话)，找出错误原因(例如设计错误、连线错误或器件有问题)。如果开始观察波形时就是正确的，教师可以人为设置一些故障，训练学生排除故障的能力。

下面介绍示波器使用方法，主要讨论如何选择示波器的同步信号。

用普通示波器观察信号，要求该信号是能不断重复的，示波器不断进行水平方向的扫描，调节示波器的同步旋钮，使波形稳定在屏幕上。同步有两种方式：内同步和外同步。内同步是利用被观察的信号对示波器水平扫描的起点进行同步，外同步是依靠外接的同步信号对水平扫描的起点进行同步。如果各观察点的波形是相对独立的，采用内、外同步的效果是基本一致的，如果要测试一些观察点波形变化的相对关系，如图 14.4 和图 14.5 所示，而示波器同时只能测试 1 个观察点(单踪示波器)或 2 个观察点(双踪示波器)，那么采用外同步比较合适，而且最好选择在一个循环周期中变化频率最低的信号作为外同步信号，在图 14.4 中，选择 D_2 或 $T_0 \sim T_7$ 中的一个信号作为外同步信号比较合适，外同步信号固定后，各观察点在屏幕上的相对时序关系也就确定了。

学生可以采用不同的同步方式(内、外同步)和不同的外同步信号(例如 D_0 和 D_2)来比较其对观察波形的同步能力(如果用 D_0 作外同步信号，观察 D_2 波形，会感到波形不稳定)。选择合适的同步方式和外同步信号对以后的实验也是很有用的。

实验是在特定环境下进行的。某些情况的出现不是恒定不变的(例如图 14.5 中的尖脉冲是否出现及其出现后的脉冲幅度和宽度)，所以对实验的结果要加以分析，求得合理的解释。

2) 执行一条指令的时钟 CP 和机器周期 T1、T2、T3(见 14.2.2 节)

3) 寄存器的接收和保持(例如 IR)(见 14.2.2 节)

2. 运算器的实验

实验要求：实现 8 位二进制数据的加法运算和乘法运算。

实验装置中有一个 8 位并行加法器、两个 8 位寄存器 A 和 B。当进行加法运算时，A 存放运算结果，B 无用，加数与被加数由控制面板的数据开关设置，A 的输出接到控制面板上

的指示灯。当进行乘法运算时，乘数与被乘数由控制面板的数据开关设置，运算前，将乘数送到 B 寄存器，A、B 寄存器有串联右移功能，A 寄存器在运算过程中保存部分积，运算结束后保存乘积的高位，乘积的低位保存在 B 寄存器中。为进行乘法运算，实验装置需要发出加和移位的控制信号并控制加和移位的次数。

实验装置中也可设置 3 个 8 位寄存器，参加运算的数先从开关送到寄存器，然后由寄存器送到加法器进行运算。

实验内容：可从以下条目中选择一些进行。

(1) 加法运算：数据开关设置加数和被加数。

① 单脉冲执行加法运算，从指示灯观察运算结果。如有错误，而又无法判别原因时，则重复(连续)执行加法运算，并用示波器观察各点波形，寻找原因。见③。

② 改变数据后再按照①执行。

③ 连续执行加法运算，测试各点波形，测试点可选择加法器的输出、进位的输入与输出等。测试结果与预先算出的结果比较，如果不相等，则寻找原因。

④ 测试最长的进位时间(与数据有关)，将实测到的进行一次加法运算的时间与计算所得的加法时间进行比较，从而估算出该加法器允许工作的最高频率(注意：由于各个门电路的延迟时间和上升沿、下降沿的差异，实测得的时间与根据器件说明书提供的参数而计算出来的时间有差异)。

⑤ 改变数据后再执行③和④。

上述①与②的实验是单脉冲执行加法运算。用指示灯观察，③、④与⑤的实验是重复执行加法运算；用示波器观察，为了能观察到波形，加法器的输入数据要有规则地进行变化。当加法器的输入数据直接取自开关时，建议将控制面板上的数据开关通过“门”连接到加法器上，如图 14.6 所示，该图仅画出了一位的电路，用控制信号来控制数据的输入，当控制信号为高电平时，将两组开关的数据送到全加器；当控制信号为低电平时，加法器的输入数据为 0，因此加法器轮流执行“加数＋被加数”和“0＋0”运算。利用该控制信号作为外同步信号，就可以进行观察波形的实验了。

图 14.7 画出在控制信号作用下，加法器某 2 位的输出波形示意图，假设某中 1 位的输出稳定后，1、0 相间变化，另一位稳定后全部为 0。由于各位相加的数据和进位信号可能不同，而且各个门的延迟时间有差别，所以在控制信号改变时，在输出点上会产生杂乱的波形，然后才能稳定下来，得到运算结果。杂乱波形的形状和幅度等一般都不相同。

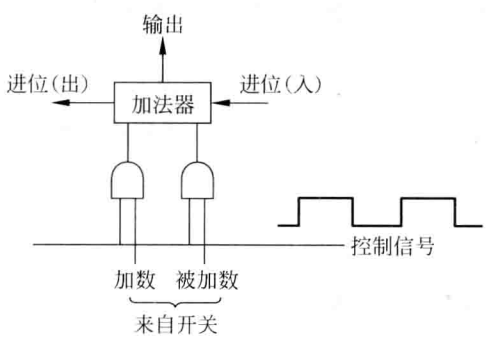


图 14.6　加法器(1 位)实验逻辑图

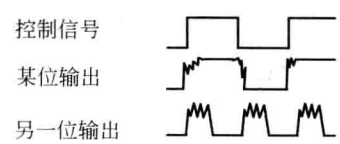


图 14.7　加法器输出波形示意图

加法器没有记忆作用，其输出结果随输入而变。

(2) 乘法运算：在加法运算已能执行的情况下，乘法运算就比较简单了。

乘法运算可以用单脉冲一步一步进行，用指示灯观察每步操作的结果。

如果发现某步操作的结果有错，那么首先考虑乘法运算与加法运算有什么不同，检查尚未测试过的部分电路，用电表或示波器测量，电表测量静态电平，示波器测量动态波形，后者要求被测信号是能够重复周期变化的。例如，如果想测试移位(右移)功能是否正确，则将A、B两寄存器串接起来，并且进行循环移位；或者在移位时，不断补充1010…数据到A寄存器的最高位。测试方法可以多样化。

(3) 其他实验：可用其他实验补充或替换上述实验，在掌握了器件的逻辑特性和物理特性以及调试方法后，进行其他部件实验也就不难了。

3. 存储器的实验

主要测试存储器的写入和读出功能。写入时可考虑在控制板的开关上设置地址和数据，并发出写入命令，将数据写入存储器的指定地址中。至于是否真正写入，可通过读出操作进行检查。读出时，也在控制板的开关上设置地址，将数据读出到A寄存器，由控制板上的指示灯显示数据。

存储器和存储控制器芯片在市场上都能购买到。如果采用动态随机存储器(DRAM)，要注意该存储控制器刷新所需的时钟是存储控制器内部自己产生的还是需要外接的，假如需外接，则要用不受控制的时钟脉冲，即用在启停控制电路中的CLK而不是CP(见14.1.1节的按键部分)，否则将因启停控制电路处于停机状态而导致存储器刷新不能按时进行，其结果是存储器中的数据丢失。在计算机的其他地方也有可能用到CLK脉冲的情况。

如果单脉冲进行存储器写入和读出操作不能正确执行，而且也不能肯定出错原因时，可考虑连续进行读写操作，用示波器根据电路各部分的输入输出关系，逐点检查其波形。先检查存储控制芯片的输入和输出($\overline{RAS}$、$\overline{CAS}$、…)信号，再检查存储器芯片的读写功能，在一般情况下应该先执行写入操作，再执行读出操作。如果有刷新操作，有可能会影响观察的波形。

在执行写入操作时，连续送入写(WR)命令，写命令的周期应参考存储器说明书给出的读写周期，从控制板上给出存储器地址和数据，并用写命令作为示波器的外同步信号。在实验前应仔细阅读存储控制器芯片和存储器芯片的说明书。

在实验时，也可使存储器的地址寄存器有计数功能，这样就可以对所有存储单元按顺序写入内容或读出数据。

执行读出的操作方法类似，将读出数据显示在控制板的指示灯上，或用示波器观察。

如果连线及各点信号都正确，但仍不能读写时，就要怀疑芯片是否是好的了。更换芯片前一定要仔细检查芯片各个封装引出端是否异常，例如在非电源引出端是否有+5V电压，电路的输出端是否有短路现象等，否则有可能会损坏新换上去的芯片。

4. 外部设备的实验

外部设备种类很多，工作原理各不相同，调试方法将因设备而异。如果中断和DMA部分要独立进行调试则比较费劲，由于外部设备的逻辑电路部分比CPU简单得多，因此不必刻意安排这方面的实验，如果教学计划中还有微机原理、操作系统、汇编语言等课程，都会涉及中断和DMA，所以以后还有机会进行这方面的实验。

如果条件具备，可考虑安排与设备有关的实验，例如磁盘存储器中存储数据的电、磁信号。

磁盘存储器盘片上的信息是依靠磁头中的电流而写入的，由剩磁的不同来区分“1”和“0”两种数据，读出时，依靠磁头与盘片的相对运动而在磁头的读出线圈上感应出电压。例如，采用不归零制(NRZ)记录方式，其写入电流和读出信号的理想时序波形如图 14.8 所示。实际上真实情况与理想情况相差甚远。首先，写入电流和盘片上磁化强度(剩磁)波形由于相邻位的影响而不同，因而读出信号与存储密度和相邻位数据有关。其次，依靠剩磁在磁头读出线上读出的感应电压是很小的，极容易受到干扰，而且干扰波形的幅度可以比读出的信号大得多，但其峰值在时间上与读出信号的峰值不重叠，可用较窄的同步信号(称之为选通信号)将干扰信号滤去，而将读出信号通过读出放大器输出。

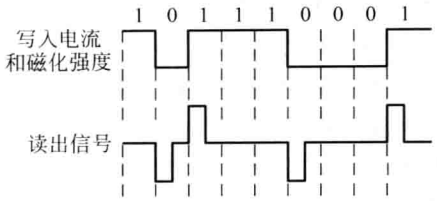


图 14.8　不归零制

本实验的目的是了解磁盘存储器的读写原理以及对小信号的处理方法。

由于磁盘的读出信号很小，极易受到干扰而破坏对读出信号的辨别，因而对读出线的布局予以特殊的处理，读出线圈与读出放大器之间的连线尽量短，注意屏蔽，并远离其他大信号线。测量仪器要用探头，并在探头处将信号放大，以减少测量线对读出信号的影响。在实验中可以用改变数据序列的方法来考察相邻单元之间的影响，改变数据序列有时就相当于改变了存储密度。

本实验可用计算机上淘汰下来的磁盘驱动器经改造后实现，由于难度大，不作为基本要求。

5. 有关实验的几点说明

(1) 实验方法和测试方法具有一致性。

前面讲到的实验方法实际上也是电路的测试方法，虽然目前设计时已有很多软件设计工具可供使用，但是有时还免不了要进行人工调试(例如连线错误或器件损坏)，同时对软件设计工具使用的某些参数可能已进行了简化，因此还需要判断其正确性，这说明掌握最基本的人工测试方法和取得实际经验是必要的。

(2) 实验设备除了要有控制板和被实验的电路外，还应该有时钟、启停电路以及产生实验所需控制信号的时序电路。另外还应该安排一些接线孔和测试观察点，接线孔是为了便于学生做实验时临时改线所用，测试观察点是为了便于观察。如果学生需要到芯片管脚上去测量(应尽量避免)，一定要小心操作，不要产生管脚之间的短路。

被实验的电路可以由学生自己设计和安装，也可以在实验装置中事先安排好一些基本电路，这样可以节省时间，也便于教师指导。

(3) 实验内容可根据具体情况(例如实验装置的情况)确定，主要是在了解电路基本原理的情况下，学会测试方法，培养分析问题和解决问题的能力。

(4) 实验所用的集成电路芯片的集成度宜适当低些，这样可提供更多的观察点。

(5) 单脉冲工作和连续工作有以下差别：单脉冲工作频率极低，连续工作则有电路允许工作的频率问题，有时两者的操作结果不相等。当发生这种情况时，要分析是设计问题还是频率问题。

(6) 在实际电路中，由于导线之间的电容与电磁感应的存在，再加上导线中的信号反射，观察点的波形有可能发生畸变，如图 14.9 所示，如果振荡幅度过大，会引起错误。

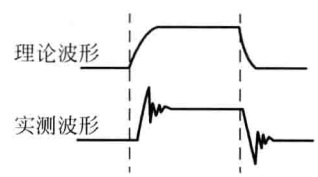


图 14.9　波形的振荡

(7) 由于测量仪器和测量线的阻抗(电阻、电容和电感)以

及仪器内部电路的影响，使得观察点观察到的波形产生变化，如果用示波器测量波形，要求示波器的允许工作频率远大于被测电路的工作频率（厂家给出的示波器频率是指正弦波频率，而被测电路的频率是脉冲频率，两者差别很大）。

（8）门电路多余输入端按以下方法处理。如果具有 5 个输入端的与门（包括与非门），仅用了 3 个输入端，那么其他 2 个输入端通过电阻接＋5V 电源；如果具有 5 个输入端的或门（包括或非门），仅用了 3 个输入端，那么其他 2 个输入端接地。这样可防止悬空的输入端（阻抗较大）受干扰影响而使门的输出不稳定。

14.2.2 整机实验和 CPU 设计

整机实验内容主要是设计并实现一台最简单的计算机（CPU）模型，从而理解计算机是怎样工作的，怎样设计和调试。

1. 整机实验的工作步骤

工作按下列步骤进行：

（1）确定计算机的字长。

（2）设计指令系统。确定每条指令的功能、指令格式和寻址方式。为方便起见，指令系统可设计得简单些。

（3）设计每条指令实现的流程，画出流程图。

（4）运算器的设计和实现。

（5）控制器的设计。

控制器有组合逻辑控制器和微程序控制器。

组合逻辑控制器：每个控制信号用逻辑表达式表示后，建议用 GAL 芯片实现，因为该芯片内电路的逻辑关系可修改，适于实验使用。控制器还应该有时钟脉冲、启停电路、机器周期电路、程序计数器和指令寄存器等。

微程序控制器：建议采用水平型微指令，因为水平型微指令比垂直型微指令的并行操作执行能力强，一般微程序控制计算机都采用水平型微指令。微程序控制器内除了有控制存储器外，与组合逻辑控制器一样，还应该有时钟脉冲、启动电路、程序计数器和指令寄存器等。

（6）CPU 与存储器的联系。

（7）实验装置的结构设计，芯片布局，跳接线孔和测试点的安排、生产、检查。

（8）调试。本方案没有考虑外部设备。调试的程序和数据考虑由控制板送入存储器，由 CPU（运算器和控制器）和存储器共同执行指令。

由于实验学时的限制，学生可根据具体情况参加一部分工作，但建议一定要参与实现指令的操作流程设计和调试工作。

一般将 CPU、存储器和 I/O 设备分开进行调试称为分调，联合在一起进行调试称为联调。

2. 中央处理器（CPU）的设计

从精简指令系统和简化硬件结构出发，设计一台完整的计算机，以增加学生对整机的理解，并予以实现。

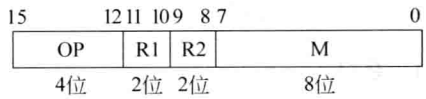
（1）指令和数据字长都是 16 位。

(2) 指令系统要求如下。

9 条指令：算术运算(加、减)2 条，逻辑移位(左移、右移)2 条，存储器(存、取数)2 条，转移(无条件转移、条件转移)2 条，动态停机 1 条。

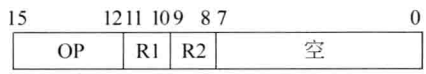
寻址方式：直接寻址和相对寻址两种，仅有转移指令采用相对寻址方式，其他指令都采用直接寻址方式。

指令格式：两地址指令。操作码 OP4 位，用 OP4～OP1 表示(指令的第 15～12 位)。



(3) 指令操作码与地址的分配

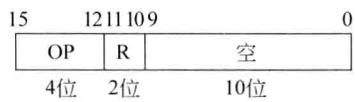
① 加法减法指令



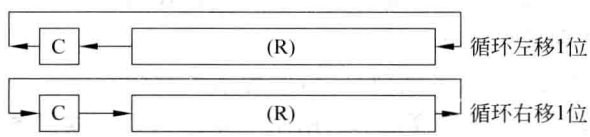
加法的操作码为 0000，减法的操作码为 0010，运算器中有 4 个通用寄存器，R1、R2 为通用寄存器地址。执行的操作为(R1)+(R2)→R1，(R1)−(R2)→R1。

根据运算结果置标志位：Z=1(结果为 0)、C=1(有进位或借位)，否则为 0。

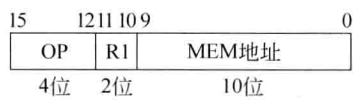
② 逻辑移位指令(带 C 的循环移位)



循环左移的操作码为 0100，循环右移的操作码为 0110。



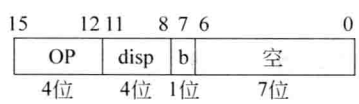
③ (存储器)存取指令——写指令、读指令



写指令的操作码为 1000，读指令的操作码为 1010。

存数(写)：MEM 地址→MAR，(R1)→MDR。取数(读)：MEM 地址→MAR，MDR→R1。

④ 转移指令



寻址方式：相对寻址，disp 为位移量，b 为转移标志位。条件转移指令的操作代码为 1100。如果 b=0，则 Z=1 转移；如果 b=1，则 C=1 转移。否则不转移。

无条件转移指令的操作码为 1110，b 无用。

转移地址是当前 PC 加 disp，但此时 PC 已经加 1，所以要调整 disp 值(减 1)。

⑤ 动态停机指令

操作码为 1111(OP4～OP1),由于其他指令的 OP1 均为 0,所以当 OP1 为 1 时,即为动态停机指令,该指令消除在机器周期 T1 时进行的 PC+1 操作的影响,在 T3 时执行 PC−1。本指令不进行其他操作,而重复执行停机指令,称为动态停机。

(4) 指令执行流程

执行一条指令需要 3 个机器周期 T1、T2、T3。时钟 CP 和 T1、T2、T3 的时间关系如图 14.10 所示。指令执行的流程如图 14.11 所示。

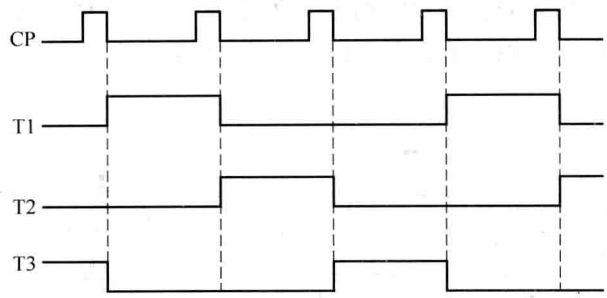
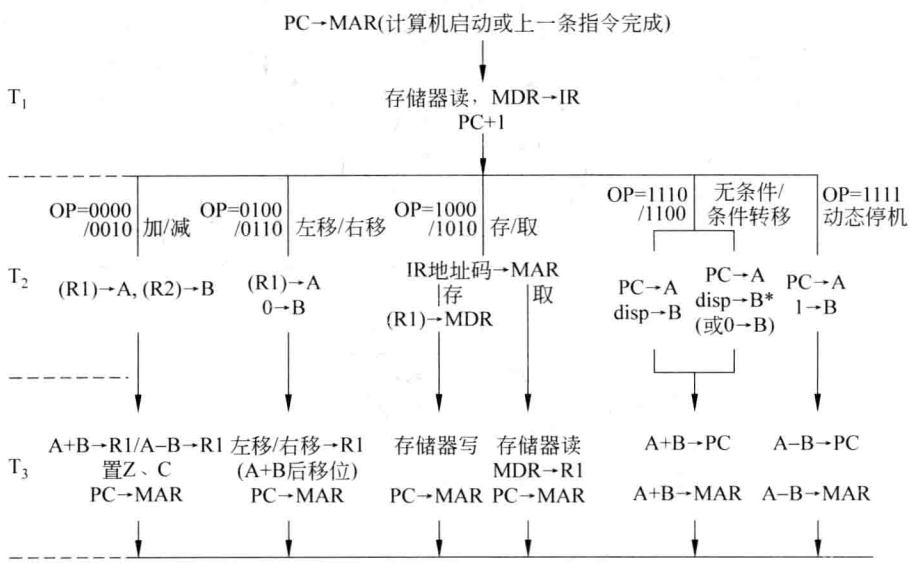


图 14.10　时钟周期和机器周期



注：*条件转移成功执行disp→B，不成功执行0→B

图 14.11　指令执行的流程图

T1、T2 和 T3 完成的操作如下：

T1　存储器读出指令送指令寄存器 IR,并执行 PC+1 操作。完成的是公共操作,即与指令操作码无关。

T2　为各条指令在 T3 的操作做准备。

T3　执行操作,并将下一条指令的地址(PC)送存储器。

阅读指令执行的流程图时,需参阅 CPU 逻辑图及相关说明。

(5) CPU 逻辑图

由实验装置控制板提供开关、按键和指示灯，并提供电源和下列信号：$\overline{\text{reset}}$、CP、T1、T2 和 T3。在图 14.10 中 T1～T3 可从 3 个 D 触发器组成的移位寄存器输出，并由 CP 的下降沿改变 D 触发器状态。在 CPU 逻辑图中的触发器或寄存器则由 CP 的上升沿起作用，CP 的负载很重，必须注意其驱动能力，而且要保证 CP 的完整性，不允许有毛刺。

如果组成寄存器的 D 触发器依靠时钟的上升沿改变状态，则可用$\overline{\text{CP}}$进行触发。

① CPU 的组成

图 14.12 为 CPU 的逻辑图，CPU 的运算器由 4 字×16 位的通用寄存器 GR、16 位 ALU 及其用于输入的 A、B 两个寄存器(各为 16 位)组成。

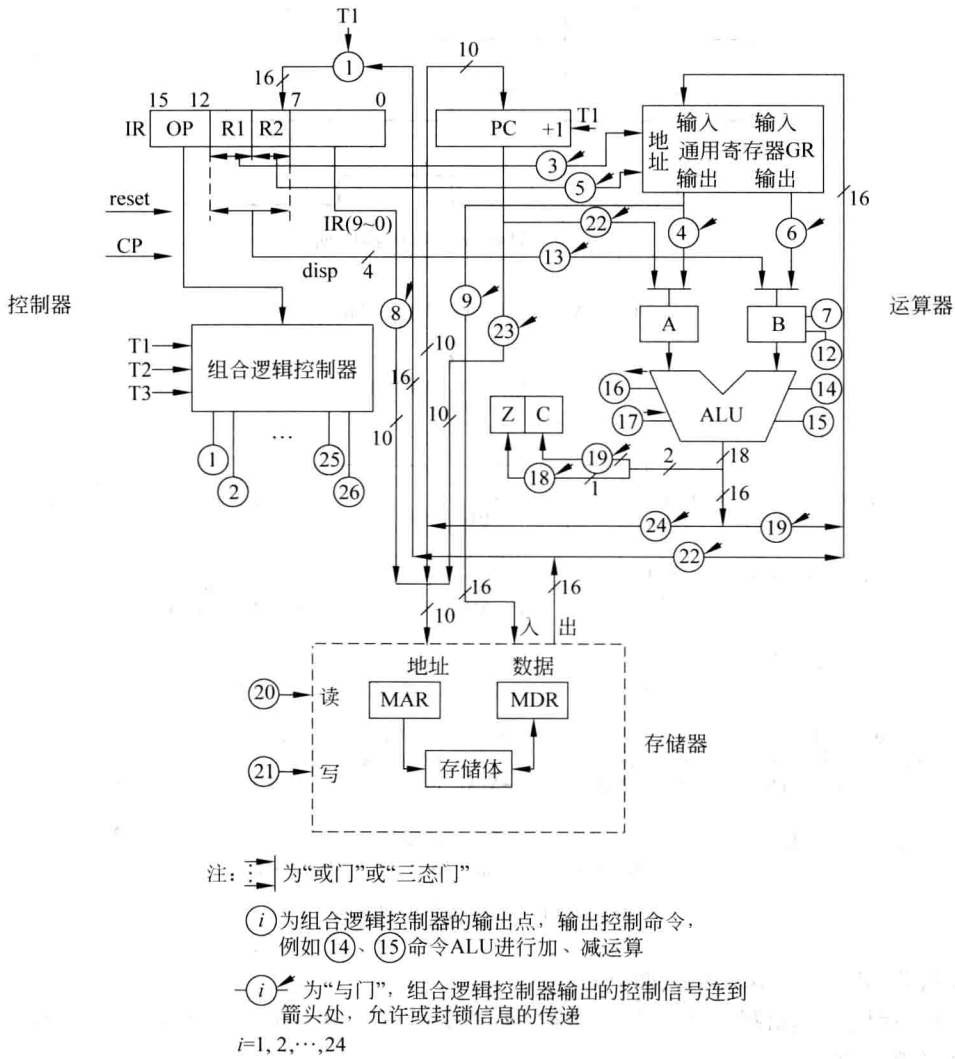


图 14.12　CPU 逻辑图

CPU 的控制器由程序计数器 PC(10 位)、指令寄存器 IR(16 位)和组合逻辑控制部件组成，后者与指令操作码和 T1、T2、T3 组合在一起，产生全机所需的全部控制信号。

存储器由存储体、地址寄存器 MAR 与数据寄存器 MDR 等组成。

下面对设计 CPU 中用到的一些基本元器件作简单介绍。

- 通用寄存器 GR 是双端口寄存器组，每个端口可以有各自的数据输入端、输出端和地址输入端，在本实验中实现了同时读取两个寄存器（输出）的功能。但仅使用了 1 个数据输入端。为全面考虑，在图上还可补上 GR 的读写控制命令，在组合逻辑控制器中以“R1 读”、“R2 读”和“R1 写”逻辑表达式表示。
- ALU 对输入的两个数据进行加法和减法运算，或对一个数据进行左移 1 位/右移 1 位操作。当进行加法/减法运算时，根据运算结果在 ALU 内部产生标志信号 Z、C，并传递到 ALU 外部的 Z、C 标志触发器，供条件转移指令使用。ALU 的输出有 16 位数据和 2 个标志位，共 18 位。
- PC 是能完成＋1 操作的计数器。
- IR 是 16 位指令寄存器，例如加法/减法指令，第 0～7 位没有使用。某些计算机考虑在存储器中 1 个字（16 位）存放 2 条 8 位指令。在本 CPU 中，指令长度固定为 16 位。
- 组合逻辑控制器在操作码和 T1、T2、T3 组合下产生完成指令执行流程图中各操作所需的控制信号，基本上由“与门”和“或门”二级逻辑电路实现，将在后面列出其表达式，学生详细阅读表达式后会增强对计算机整机的认识。
- 存储器中有地址寄存器 MAR、数据寄存器 MDR、存储体和一些控制电路。存储器和 CPU 之间的联系端有 10 位地址输入端、16 位数据输入端、16 位数据输出端和读命令端、写命令端等。执行读命令时，存储体读出的信息应在 CP 上升沿来之前已传送到 MDR 中。

② 设计重点

在指令系统确定后，主要设计指令执行的流程和实现 CPU 的逻辑图，其中 GR、ALU、PC 和 IR 可选择市场上供应的元器件组成，后面讨论的组合逻辑控制器将是进一步设计的重点。

(6) 组合逻辑控制器的逻辑表达式

综合指令系统、指令执行流程图和 CPU 逻辑图列出实现控制信号的表达式（①、②、…），并在组合逻辑控制器中用门电路构成控制信号输出，送到逻辑图相应的标号（①、②、…）处。用逻辑表达式表示如下：

① MDR→IR＝T1，PC＋1＝T1。为公共操作，与指令操作码无关。

② 备用。

③ R1→GR 表示 R1 地址送 GR，为在 GR 中读数（即下面的④等）作准备，凡是到 GR 存数或取数都要增加类似的操作。

$$
\begin{aligned}
R1 \rightarrow GR &= \text{指令操作码} \times \text{机器周期} \\
&= \text{加/减指令} \cdot T2 + \text{移位指令} \cdot T2 + \text{存数指令} \cdot T2 + \text{取数指令} \cdot T3 \\
&= \overline{OP4} \cdot \overline{OP3} \cdot T2 + \overline{OP4} \cdot OP3 \cdot T2 + OP4 \cdot \overline{OP3} \cdot \overline{OP2} \cdot T2 \\
&\quad + OP4 \cdot \overline{OP3} \cdot OP2 \cdot T3
\end{aligned}
$$

上式中的指令操作码经简化后得到，例如：

$$
\begin{aligned}
\text{加/减指令} &= \overline{OP4} \cdot \overline{OP3} \cdot \overline{OP2} \cdot \overline{OP1} + \overline{OP4} \cdot \overline{OP3} \cdot OP2 \cdot \overline{OP1} \\
&= \overline{OP4} \cdot \overline{OP3} \cdot \overline{OP2} + \overline{OP4} \cdot \overline{OP3} \cdot OP2
\end{aligned}
$$

（因为不用操作码 0001 和 0011，所以删去$\overline{OP1}$）

$=\overline{OP4}\cdot\overline{OP3}\cdot(\overline{OP2}+OP2)=\overline{OP4}\cdot\overline{OP3}$

④ (R1)→A 表示将 R1 的内容送到 A 寄存器。

(R1) → A ＝加 / 减指令 · T2 ＋ 移位指令 · T2

$=\overline{OP4}\cdot\overline{OP3}\cdot T2+\overline{OP4}\cdot OP3\cdot T2=\overline{OP4}\cdot T2$

⑤ R2→GR＝加/减指令 · T2＝$\overline{OP4}\cdot\overline{OP3}\cdot T2$

⑥ (R2)→B＝$\overline{OP4}\cdot\overline{OP3}\cdot T2$。⑥＝⑤，可省略 1 个表达式。

⑦ 0→B 表示将 16 位 B 寄存器全部置为 0。

0 → B ＝移位指令 · T2 ＋ 条件转移指令（转移不成功）· T2

$=\overline{OP4}\cdot OP3\cdot T2+OP4\cdot OP3\cdot\overline{OP2}\cdot\overline{OP1}(\bar{b}\bar{Z}+b\bar{C})\cdot T2$

式中的 b 是条件转移指令第 7 位的值。

⑧ IR 地址码→MAR＝存取指令 · T2＝$OP4\cdot\overline{OP3}\cdot T2$

⑨ (R1)→MDR＝存数指令 · T2＝$OP4\cdot\overline{OP3}\cdot\overline{OP2}\cdot T2$

⑩ 备用。

⑪ PC→A ＝转移指令 · T2＋动态停机指令 · T2

$=(OP4\cdot OP3\cdot\overline{OP2}\cdot\overline{OP1}+OP4\cdot OP3\cdot OP2\cdot\overline{OP1}$

$+OP4\cdot OP3\cdot OP2\cdot OP1)\cdot T2$

$=OP4\cdot OP3\cdot T2$

⑫ 1→B 表示将 16 位 B 寄存器的最低位置为 1，其他位均为 0。

1→B＝动态停机指令 · T2＝$OP4\cdot OP3\cdot OP2\cdot OP1\cdot T2=OP1\cdot T2$

可简化的原因：在全机的指令系统中，唯有动态停机指令的 OP1 等于 1，其他指令均等于 0）。

⑬ disp→B ＝无条件转移指令 · T2＋条件转移指令（转移成功）· T2

$=OP4\cdot OP3\cdot OP2\cdot\overline{OP1}\cdot T2$

$+OP4\cdot OP3\cdot\overline{OP2}\cdot\overline{OP1}(\bar{b}Z+bC)\cdot T2$

由于本机没有用操作码 1101，所以可进一步将公式中的条件转移指令 $OP4\cdot OP3\cdot OP2\cdot OP1$ 简化为 $OP4\cdot OP3\cdot\overline{OP2}$。下面按此规则直接简化其他表达式。不再详细说明。

⑭～⑰ 是送往 ALU 的操作命令。

⑭ ＋＝加法指令 · T3＋转移指令 · T3

$=\overline{OP4}\cdot\overline{OP3}\cdot\overline{OP2}\cdot T3+(OP4\cdot OP3\cdot\overline{OP2}\cdot\overline{OP1}$

$+OP4\cdot OP3\cdot OP2\cdot\overline{OP1})\cdot T3$

$=\overline{OP4}\cdot\overline{OP3}\cdot\overline{OP2}\cdot T3+OP4\cdot OP3\cdot\overline{OP1}\cdot T3$

⑮ －＝减法指令 · T3＋动态停机指令 · T3

$=\overline{OP4}\cdot\overline{OP3}\cdot OP2\cdot T3+OP4\cdot OP3\cdot OP2\cdot OP1\cdot T3$

$=\overline{OP4}\cdot\overline{OP3}\cdot OP2\cdot T3+OP1\cdot T3$

⑯ 左移＝左移指令 · T3＝$\overline{OP4}\cdot OP3\cdot\overline{OP2}\cdot T3$

⑰ 右移＝右移指令 · T3＝$\overline{OP4}\cdot OP3\cdot OP2\cdot T3$

⑱ ALU 输出(Z)→Z＝加/减指令・T3＝$\overline{OP4}$・$\overline{OP3}$・T3，完成置状态位 Z 操作

⑲ ALU 输出(C)→C＝(加/减指令＋移位指令)・T3＝$\overline{OP4}$・T3，完成置状态位 C 操作

ALU 输出(16 位)→R1＝(加/减指令＋移位指令)・T3＝$\overline{OP4}$・T3

⑳ 读命令(送存储器)＝取指操作・T1＋取数指令・T3

＝T1＋OP4・$\overline{OP3}$・OP2・T3

㉑ 写命令(送存储器)＝存数指令・T3＝OP4・$\overline{OP3}$・$\overline{OP2}$・T3

㉒ MDR→R1＝取数指令・T3＝OP4・$\overline{OP3}$・OP2・T3

㉓ PC→MAR ＝(加/减＋移位＋存取)指令・T3＝$\overline{OP4・OP3}$・T3

＝($\overline{OP4}$＋$\overline{OP3}$)・T3

㉔ ALU 输出→PC(即图 14.11 中的 A＋B→PC/A－B→PC)＝(转移指令＋动态停机指令)・T3＝OP4・OP3・T3

ALU 输出→MAR(表达式同上)

㉕ R1 读＝加/减指令・T2＋移位指令・T2＋存数指令＋T2

＝(加/减指令＋移位指令)・T2＋存储指令・T2

＝$\overline{OP4}$・T2＋OP4・$\overline{OP3}$・$\overline{OP2}$・T2

㉖ R2 读＝加/减指令・T2，与⑤或⑥相同，所以选择⑤

㉗ R1 写＝加/减指令・T3＋移位指令・T3＋取数指令・T3

＝⑲＋㉑

综合上述表达式，可进一步予以简化，举例如下：

- 将某些操作码先组合起来，例如令 P＝OP4・OP3，然后将“P”组合在⑪、⑬和㉔逻辑表达式中，并将“$\overline{P}$”(即为$\overline{OP4・OP3}$)组合在㉓逻辑表达式中。
- 合理设计并分配各条指令的操作码，例如用 OP4 和 OP3 对指令进行分类，(OP4OP3＝00 为加/减指令，01 为移位指令…)，同时某些操作码没有用到，因此在执行程序时不会出现，从而可简化逻辑表达式。例如在逻辑表达式③中，将加/减指令类的操作码简化为$\overline{OP4}$・$\overline{OP3}$。
- 还有其他方法，学生如有兴趣，可进一步探讨，这里不再讨论。然而从前面的逻辑式中也可发现，如果设计中有些错误或者要增加一些指令，由于表达式已进行了简化，要进行纠错或改动，有可能非常困难。

(7) 时钟周期、电路的延迟、记忆功能和负载能力

从 CPU 逻辑图中，可以得知 CPU 基本上由组合电路(门)和时序逻辑电路(触发器)组成，电路有延迟和驱动负载能力的限制，需加以分析。

① 时钟周期的确定

分析指令执行流程，明确需要对在一个时钟周期内进行算术运算和存储器读/写两种操作各自所需的时间进行计算与实验，取其较长的时间定为时钟周期的主要部分。

- 算术运算时间。时钟周期的时间长度等于下列几项之和：组合逻辑控制器形成控制命令的时间、在 ALU 部件中的运算时间、处理结果送到目的寄存器的时间和时钟脉冲宽度的时间。其倒数(1/时钟周期)即为 CPU 的主频。
- 存储器读写时间。由逻辑控制电路和存储体读写时间而定。

在复杂的计算机中如果有某些操作所需时间更长，则采取延长个别周期时间或增加周期数的办法。

在某些计算机中，为提高主频，即减少时钟周期的时间，可考虑用相同的时钟边沿（例如上升沿）对类似于前面介绍的“CPU 逻辑图中的触发器”和“形成机器周期的 T 触发器”起作用，但此时需考虑电路的延迟，确保对全机起作用的脉冲波形（尤其是上升沿）的一致性。

② 电路的延迟时间

器件的说明书中都有延迟时间的参数，但是在实验设备中，由于负载情况、环境温度和电压波动等因素的影响，同一型号的器件的延迟时间、上升沿和下降沿会有差别，需予以注意，并通过部件测试或整机测试予以验证。

③ 在 CP 的作用下，寄存器（或触发器）接收数据和保留原值方法

CPU 中的某些寄存器。例如 IR，在 T1 期间的 CP 上升沿使它接收指令，而在 T2 和 T3 期间应保持不变。为此可对触发器采取图 14.13(a)所示的方法控制其数据输入端，图中的$\overline{T1}$等于 T2＋T3。也可以采取图 14.13(b)的方法。

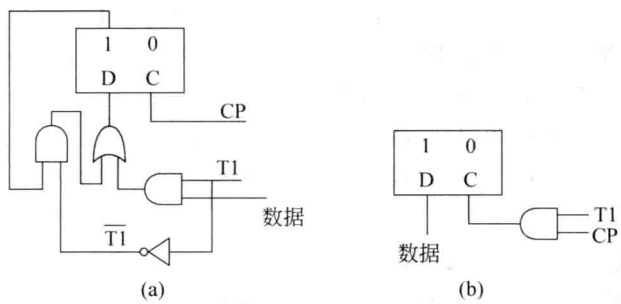


图 14.13　数据的接收和保持原值

在计算机中，要保证各触发器状态变化瞬间的一致性。

触发器能保持过去状态的功能被称为有记忆功能的电路。

仅由门组成的逻辑电路则无记忆功能。

④ 逻辑门的输出级电路分析

逻辑门一般由几级电路组成，输入级一般完成逻辑功能，输出级驱动负载。这里主要讨论输出级。

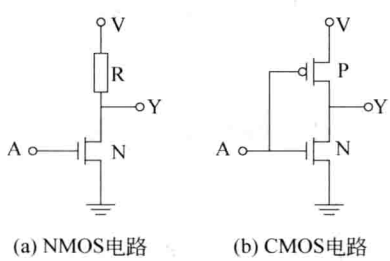


图 14.14　反相器（非门）

先讨论输出 1（高电平）和 0（低电平）两种状态的逻辑门输出级。

图 14.14(a)是 NMOS 反相器（相当于非门），当输入 A 为高电平时，NMOS 管通导，输出 Y 为低电平；当输入为低电平时，NMOS 管截止，输出 Y 为高电平。输出端的阻抗由 RC 决定，其值较大，使输出波形上升沿比下降沿大得多。在图 14.14(b)中，当输入 A 为高电平时，NMOS 管通导，PMOS 管截止，输出 Y 为低电平；当 A 为低电平时，NMOS 管截止，PMOS 管通导，输出 Y 为高电平。输出级的性能有 NMOS 和 PMOS 管互补，称为 CMOS 电路，与 NMOS 电路相比，改善了波形变化的速度，因而减少了延迟时间，同时降低了功耗。

如果门的负载重，可增加输出级在芯片中的尺寸，给出较大的驱动电流或减少延迟时间，可称它为驱动器。

上述电路的输出线不能直接连到总线上，否则相当于多个门的 R 并联起来(图 14.14(a))，减少了等效电阻值，增加了 NMOS 的电流，并降低了驱动负载的能力。市场上有集极开路(OC)的门电路，在芯片内部去掉电阻 R，多个 OC 门的输出线可直接连到总线上，在芯片外部仅接一个电阻 R 到电源上，从而避免了前述的问题。其缺点是延迟时间长，波形变化慢(与三态门比较)。

图 14.14(b)所示电路的输出更不能接到总线，否则会出现这样的情况：电源将通过一个门的通导 PMOS 管和另一个门的通导 NMOS 管与地短接，因电流过大而损毁 MOS 管。

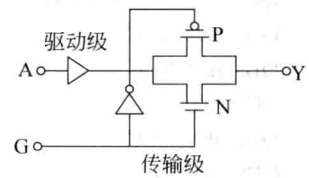


图 14.15　三态门电路

图 14.15 是经常连接到总线上的三态门电路，其输出有 1、0 和高阻 Z 三种状态。三态门电路由驱动级和传输级组成，信号的输入为 A，输出为 Y，G 为控制信号，当 G 为 1 时，NMOS 管和 PMOS 管都导通，于是 Y＝A；当 G 为 0 时，NMOS 管和 PMOS 都截止，输出线 Y 相当于一根悬空的导线(高阻)，因而可以连接到总线上。如果总线上连接多个三态门，在工作时只允许连到总线上的一个三态门的控制信号 G 为 1，其他三态门的 G 均为 0。

3. 可编程逻辑器件(PLD)

PLD 是可编程的阵列逻辑电路，有 PLA、PAL 和 GAL 等品种(参阅教材第 2 章)，其中 GAL 是可重复编程的器件，适宜于实验时发现问题或改进性能时多次修改内容。可使用 GAL 厂家提供的 GAL 专用编程软件和 GAL 编程器，向 GAL 器件写入所需的逻辑功能，由于不同厂家提供的编程软件和编程器有所不同，具体操作需阅读有关产品说明书和使用手册，这里仅作简要的说明。

GAL 芯片从输入到输出共经过三级电路，第一级为“与”逻辑电路，第二级为“或”逻辑电路，第三级为触发器(也可以不经过触发器就输出)。参阅前面设计的 CPU 逻辑图和信号表达式，说明在很多场合可使用 GAL 芯片。

下面以 GAL 16V8 为例作一说明(参见教材第 2 章图 2.26)。

GAL 16V8 的封装图如图 14.16 所示。它有 20 个引出端，其中 8 个固定为输入端(I_1～I_8)，另有 8 个可由编程选择作为输入端或输出端(I/O)，在图上用 F_0～F_7 表示。引出端1 与 11 有两种选择，也是通过编程确定的，引出端 1 可用作 I_0 输入端或时钟输入端 CLK，引出端 11 可用作 I_9 输入端或输出控制端$\overline{OE}$。各个逻辑控制单元 OLMC 的$\overline{OE}$是相互独立的，当该单元的$\overline{OE}$是低电平时，允许相应的 OLMC 输出；当$\overline{OE}$为高电平时，相应的 OLMC 输出为高阻抗(无输出)。今使用 GAL 16V8 来组成十进制计数器。在第 2 章第 16 题中用 PLA 和 4 位 D 触发器 DD、DC、DB、DA 组成计数器，而在 GAL 芯片中，可使用 OLMC 中的触发器。

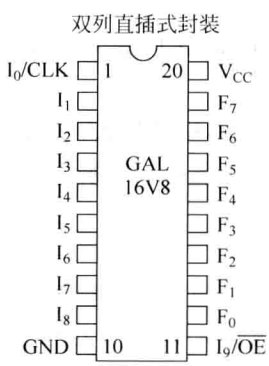


图 14.16　GAL 16V8 的封装图

十进制计数器的源程序文件格式如下：

```
GAL 16V8
×××…×
×××…×
1    2    3    4    5    6    7    8    9    10
CLK  NC   NC   NC   NC   NC   NC   NC   NC   GND
NC   QA   QB   QC   QD   NC   NC   NC   NC   VCC
QA :=/QA
QB :=/QD * /QB * QA+QB * /QA
QC :=/QC * QB * QA+QC * /QB+QC * /QA
QD :=/QD * QC * QB * QA+QD * /QA
QA. OE=VCC
QB. OE=VCC
QC. OE=VCC
QD. OE=VCC
DESCRIPTION ↵
```

对源程序文件的说明如下。

(1) 源程序文件格式：第 1 行为器件型号，必须起始于第 1 行第 1 列，例如 GAL 16V8。

第 2 行、第 3 行由设计者自行决定，例如，可以根据 GAL 的功能予以命名，写上编程日期等，GAL 编程器不对它进行处理。

第 4 行标出引出端的顺序编号。

第 5 行起为引出端名，可占用多行，从第 1 个引出端到最后一个引出端顺序给出，不用的引出端命名为 NC，电源命名为 VCC，地线命名为 GND，其余引出端可由设计者命名，引出端名之间用空格分隔。

引出端定义后，给出逻辑表达式。

表达式后，用关键字 DESCRIPTION(大写字母)结束源程序文件。

在本例中，由于计数器的输入信号是从输出反馈而来，因此不需要外加输入信号。

(2) 逻辑表达式：逻辑表达式中可含有 3 种逻辑运算符：*(与)、+(或)和/(非)，逻辑表达式必须采用“与-或”形式。编程软件没有简化逻辑表达式功能。

有两种赋值语句：

① 输出＝表达式

② 输出 :＝表达式

第①种赋值语句由等式右边的表达式直接输出，第②种赋值语句由 D 触发器输出，在时钟脉冲上升沿到来时将表达式给出的逻辑值送入 D 触发器。

(3) 输出控制：输出. OE＝表达式。当表达式的结果为高电平时，由符号名指定的输出引出端才能输出有效电平，否则就保持高阻态。这里的表达式只允许包括一个乘积项。

上例中的 QA. OE＝VCC 表示 QA 能输出。

下面再举一个例子。

用 GAL 16V8 组成 1 个 8 位移位寄存器，由 M 控制左右移，M＝0 时，左移；M＝1 时，右移。左移时，最低位的输入数据为 DL；右移时，最高位的输入数据为 DR。8 位移位寄存器的输出为 Q7Q6…Q1Q0。

移位寄存器的源程序文件如下：

```
GAL 16V8
××…×
××…×
1    2    3    4    5    6    7    8    9    10
CLK  M    DL   DR   NC   NC   NC   NC   NC   GND
NC   Q0   Q1   Q2   Q3   Q4   Q5   Q6   Q7   VCC
Q7 :=/M * Q6+M * DR
Q6 :=/M * Q5+M * Q7
Q5 :=/M * Q4+M * Q6
Q4.:=/M * Q3+M * Q5
Q3 :=/M * Q2+M * Q4
Q2 :=/M * Q1+M * Q3
Q1 :=/M * Q0+M * Q2
Q0 :=/M * DL+M * Q1
Q7. OE=VCC
Q6. OE=VCC
Q5. OE=VCC
Q4. OE=VCC
Q3. OE=VCC
Q2. OE=VCC
Q1. OE=VCC
Q0. OE=VCC
DESCRIPTION ↵
```

上面两个例子都是从触发器输出的，如果用 GAL 作为译码器或用来产生组合逻辑控制器中的控制命令，则从 OLMC 中的“与-或”电路输出。

假设译码器的输入信号为 D3，D2，D1，输出为 T0，T1，…，T6，T7，则 GAL 编程文件中的逻辑表达式为

```
T0=/D3 * /D2 * /D1
T1=/D3 * /D2 * D1
T2=/D3 * D2 * /D1
T3=/D3 * D2 * D1
T4=D3 * /D2 * /D1
T5=D3 * /D2 * D1
T6=D3 * D2 * /D1
T7=D3 * D2 * D1
```

编程文件中的其他内容略。

4. CPU 的测试方法

先进行各部分（运算器、存储器、时钟及机器周期电路）分调。如果在进行部件实验前已进行了实验装置的总体设计，各个部件是按总体要求设计的，那么分调工作已在部件实验时完成，否则在这里要按前面介绍的方法进行一次分调。

然后测试各部件间的连线是否正确，在本例中就是测试运算器、控制器与存储器连线的正确性。

最后测试每条指令的功能是否与设计要求相符。指令和数据由控制板开关送入存储

器，并从控制板设置 PC 值。如果封锁控制器中的 PC+1 控制信号，就可以不断重复执行被测试的那条指令，可以单时钟工作，也可以连续工作。在指令执行不正确时可用示波器观察，追踪错误根源，并进行更正，然后测试更正后指令的操作是否正确。

本装置测试时，先将数据和指令手动输入到存储器，调试存数指令和取数指令，然后可考虑循环执行加法和转移两条指令如下：

K	加法指令
K+1	转移到 K 的无条件转移指令

用每条被测指令逐次更换第 K 条指令，这样就可以逐次测试每一条指令的功能。

如果测试发现错误或者装置的加工工艺太差而不能更正错误，有可能要返工，严重时将从设计重新开始，所以一定要认真对待工作的每一步。

至此可以认为由 CPU 和存储器组成的计算机模型初步调试完成，通过调试学生应对计算机如何执行程序以及如何实现指令的功能有一个完整的概念。这对本课程的学习是很重要的。

为什么说是初步调试完成呢？因为还有很多工作尚未完成。大家知道计算机操作速度很快，操作量非常大，但容不得一点点错误，而计算机在对不同的数据进行处理时，计算机内各个元件的动作是不完全相同的，因此如果对某一数据进行算术运算其结果是正确的话，还不能保证在对其他数据进行运算时其结果也是正确的。另外，计算机执行不同程序时，计算机中各个元器件的实际操作频度也是不同的，有的指令执行时，还与以前执行的指令有关，因此如果个别元器件的操作频度太高或元器件的性能变坏或有逻辑上的错误，在某些情况下能表现出来，而在其他情况下却表现不出来，所以需要对不同的数据进行测试，并用多种程序来考验计算机。如果依靠控制板向存储器写入数据和程序，在工作量大时是不现实的，此时需要考虑从输入设备向存储器写入数据和程序。

实验模型的测试工作到此可以基本结束。但如果是设计一台复杂的实验机或新计算机，则需要软件配合工作，例如解决向存储器写入数据和程序问题、产生随机数问题等。并进一步进行测试。

14.3 计算机的测试、诊断和稳定性考验

14.3.1 测试、诊断和稳定性考验

在完成上一节所讨论的测试工作以后，如果是新设计的计算机，还需要进行以下测试。

1. 正确性测试

比较全面地逐条测试指令的功能，例如加法指令，要对不同的数据进行加法运算，可以进行 1111…1+1111…1，0000…0+0000…0，0101…01+0101…01，0011…0011+0011…0011，1111…1+0000…1，0010…0+0000…0，…等。每次加法运算后和预期的结果进行比较，如果有错，则停机，否则继续循环运行，如果加法结果要置状态位，需用其他指令配合测试，而该指令尚未测试，则可待有关指令测试后，再补测状态位。算术逻辑运算指令都可按这种方法进行。

上述测试加法指令的方法，实际上要执行一小段程序，举例如下：

K	ADD 指令	;被测指令
K+1	比较指令	;实际上进行减法运算,将上一条指令的运算结果与已知结果比较, ;如相等,结果为 0,状态位 Z 置 1
K+2	条件转移指令	;如果 Z=1,转移到 K,否则执行 K+3 指令
K+3	停机指令	;加法指令有错,停机检查

上述程序还未考虑数据如何变化。这说明测试一条指令需要多条指令进行配合,因此进行本处所讨论的正确性测试前实际上对每条指令的基本功能已进行过测试,否则,如果存在多处错误,则停机后检查也很困难,而且甚至可能无法知道程序转到哪里去了。循环执行程序的目的是考虑在给定的时钟频率下加法运算的稳定性。

对每条指令根据各自的特点经过上述测试后还需要进一步进行测试,仍以加法指令为例,虽然已对若干个数据进行过测试,但数量终究是有限的,为了对更多数据进行考核,在计算机内安排一段子程序,用来产生一对随机数 A 和 B,将该随机数取来进行加法运算:C=A+B,然后再执行减法运算:D=C−B,如果 D=A,则运算正确,如果不等则停机。本次运算正确后,再取下一对随机数继续运行。

最后将上述的每一段测试程序连接在一起,构成 CPU 测试程序,连续运行。连接方法如下:

K	被测指令 1	;K 到 K+n 对被测指令 1 进行测试
K+1		
⋮		
K+n	转移指令	;如果被测指令正确转移到 K+n+2,可使用相对转移指令,位移量为+2
K+n+1	停机指令	;被测指令有错,停机
K+n+2	被测指令 2	;K+n+2 到 K+n+m 对被测指令 2 进行测试
⋮		
K+n+m	转移指令	;如果被测指令正确转移到 K+n+m+2,可使用相对转移指令,位移量为+2
K+n+m+1	停机指令	;被测指令有错,停机
⋮		

CPU 测试程序是很庞大的,对每一条指令可能出现的各种情况都要测试到,所以一条指令要反复测试很多遍。每一小段程序中的被测指令有错时,停机的位置各不相同,根据停机时的 PC 地址可以知道哪条指令出了问题,缩小了寻找错误的范围。

测试程序是用汇编语言编写的。

测试存储器时要对每个存储单元存储各种数据,进行写/读操作。外部设备也是一样。这些测试都是通过程序执行的,计算机处于正常运行的环境中,控制板上除了加电、$\overline{\text{reset}}$、启动、停止按键和工作方式开关外,其他的开关均不起作用。

停机后再启动,计算机应继续运行。

为了测试计算机能正常运行的能力,一般还要进行改变电压的实验,即将直流电源电压改变一定范围(例如±5%),观察计算机是否仍能正确运行测试程序,以保证在额定电压情况下,万一电源电压产生波动或其他干扰,计算机仍能继续运行。通过变电压实验,可发现计算机内部的一些薄弱环节,及时进行处理。

2. 计算机的错误诊断

当计算机在执行程序时，发现不正常，此时需对计算机检查，其方法是让计算机运行诊断程序，诊断程序可考虑使用前面讲过的测试程序。

3. 稳定性考验

经过正确性测试后，为了考验计算机是否能稳定地长期工作，往往让计算机不间断地运行考验程序，稳定性考验程序可以由上述测试程序和一些实际的应用程序组成，运行时间约为几百小时，运行期间不允许暂时停机或人工参与。

4. 计算机正式生产

经过程序考验后，计算机可以投入生产，一般先进行小批量试生产，然后再进行大批量生产，正式生产的计算机可考虑不再采用控制板，测试时直接运行诊断程序。

在计算机进行硬件设计和调试时，应同时研制操作系统、汇编程序和高级语言程序的编译程序等，软硬件应同时交付使用。

外部设备与主机的部分联调工作可以在操作系统配合下进行。

14.3.2 为整机实验或计算机测试准备的辅助软件

1. 交叉汇编程序

在机器内部，指令字是以二进制码形式表示的，根据前面的介绍，在调试每条指令时，都要在控制板上设置二进制码的指令。如果调试一段程序，不仅麻烦，而且容易错，出错后检查也不方便。因此调试程序应该由汇编语言和二进制码同时表示。

交叉汇编程序是在已有的计算机上(例如微机)，将实验计算机模型上或新计算机上的程序(用以符号表示的指令，即汇编语言表示的指令组成的程序)，翻译成二进制码程序，由于这项翻译工作是在别的计算机上进行的，所以称之为交叉汇编程序。

2. 微汇编程序

计算机的控制器可采用组合逻辑(或称为硬连线逻辑)或微程序控制两种方案。微程序控制器内有一个用 ROM 组成的微程序存储器(称为控制存储器)，它也是用二进制码形式表示的。

微汇编程序是将用符号表示的微指令翻译成由 0 和 1 两种代码组成的微指令。这项工作也可以在其他计算机上进行。

由于模型机的指令系统与执行编译工作的计算机不同，所以汇编程序和微汇编程序不是现成的，而是需要重新编写的。

3. 其他

PLD(PAL、PLA、GAL)器件和 ROM 器件都有各自的编程软件和编程器，可将二进制码(0 和 1)写入各自的器件中，写入时，请阅读有关编程器的说明书。

小　结

本章提供的硬件实验范围从简单的逻辑电路直到计算机模型的调试，主要提供了一条如何开展实验工作的线索。在实施过程中会遇到各种预先能估计到或不能估计到的问题，解决这些问题也是实验的重要内容。另外本章比较强调因地制宜，不追求昂贵的实验装置

和测量仪器，因为消耗在这方面的成本并不与实验的收获成正比，安排得好，廉价的装置同样可以满足教学要求。

关于电子测量仪器，都在朝着数字化、智能化、网络化和微型化方向发展，例如本章所使用的普通示波器(模拟示波器)，由于要在荧光屏上显示，所以要不断重复送入被测试的信号。而采用数字化技术后，可以将一次性的输入(观察)信号经采样后存入存储器，然后可以不断地从存储器读取，显示在屏幕上，只要采样频率足够高，可以显示复杂波形和捕获干扰信号(包括前面提到的尖脉冲)。

目前某些计算机公司可为用户提供产品的远程诊断功能，即计算机有故障后，维修人员不必到现场进行检测，这是电子测量仪器与计算机、通信和网络相结合的成果。

在高等学校中，由于人数多，实验设备和测量仪器所需要的量也比较多，采购测量仪器时要选用寿命长、不易损坏、可靠性高和价格低的设备，至于功能则不用太全，只要够用就行。

科研用的设备，其标准应有所不同。

通过本章提及的实验、设计和测试后可考虑以下的问题：

(1) 在实验或批量生产时，怎样选用 PLD 器件或其他器件？

如果在实验时选用 GAL 芯片，可以多次修正设计中的错误。当实验成功而决定批量生产时，以改用 PAL 或 PLA 器件为宜，如果批量大，可考虑采用半定制或全定制器件，这主要是从器件工作的可靠性和成本来考虑。

如果实验模型规模很大，可考虑用 FPGA 器件进行实验。

采用上述可编程器件，应详细阅读说明书。

(2) 实验模型是一台简化的计算机，如果要提高机器性能，例如要实现双字长运算，引入中断处理功能，或实现子程序等，应怎样增强指令系统、修改逻辑图？这一问题有时很难解决，甚至要重新进行设计。唯有在技术人员对计算机整机的认识很清楚的情况下才能进行这项工作，本章提到的实验内容是很重要的。

第 15 章　计算机设计与测试技术(VHDL 和相关的 EDA 工具)

15.1　硬件描述语言 VHDL 和相关的电子设计自动化 EDA 工具简介

在教材的第 12.4.2 节和图 12.13 中,对集成电路的设计过程作了简介,计算机的系统设计和逻辑设计工作基本上由设计人员完成,然后将逻辑图输入拥有 EDA(电子设计自动化)工具的计算机,完成后续的设计、模拟、测试等工作。由于芯片的集成度大幅提高,逻辑图的输入工作量太大,这项工作被硬件描述语言 VHDL 替代。VHDL 语言的表达形式与其他高级语言类似,容易理解。在 VHDL 语言中(参见图 15.1),主要用实体和结构体两部分来描述电路(称为元件),而且在 EDA 工具的帮助下,还能将 VHDL 描述的元件模型转换成元件的逻辑图输出。

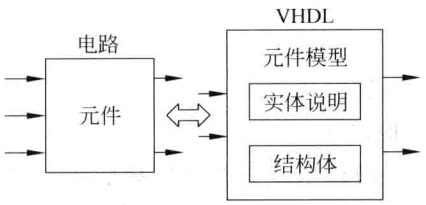


图 15.1　元件和 VHDL 元件模型的元件模型

在 1987 年 IEEE 将 VHDL 批准为标准语言后,又进行了多次修改,硬件描述语言和 EDA 工具被多家公司研究、设计和提供产品,内容有所差异,但基本上是一致的。在本书的参考文献[6]和[7]中采用的 EDA 工具是 Altera 公司的 Quartus Ⅱ,该公司为某些大学免费提供 Quartus Ⅱ软件的学生版。

本章的目标是学习 VHDL 编程方法,加强对计算机整机的认识。

1. 设计流程简介

下面简单介绍在 EDA 工具 Quartus Ⅱ和 VHDL 配合下设计计算机的过程。

首先将待设计的计算机(实验模型)由人工完成整体设计,包括指令系统、数据字长,指令执行流程和逻辑简框图等,然后才能在安装 Quartus Ⅱ和 VHDL 软件的计算机上完成后续的设计、测试和最后的实验模型。与逻辑图相关的描述由 VHDL 实现,其余的工作全部由 Quartus Ⅱ完成,设计流程如图 15.2 所示。

(1) 实验模型设计:人工完成。

(2) 建立项目、选择器件:设计者给实验模型提供一个名字,并选择 FPGA 器件(给出器件型号)。

(3) VHDL 描述:根据实验模型设计提供的逻辑简框图等编写 VHDL 程序,可采用层次化结构,由后面的举例说明层次化结构的概念。

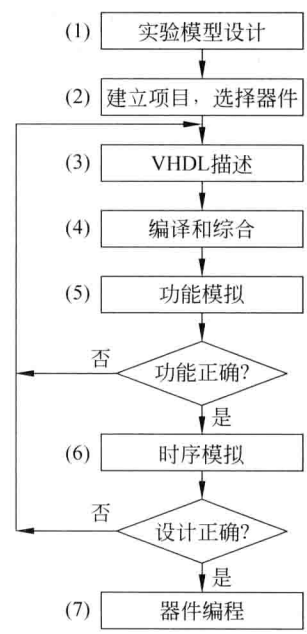


图 15.2　实验模型设计流程

(4) 编译和综合：由 Quartus Ⅱ 软件完成。根据 VHDL 描述，绘制出原理图(逻辑图)；编译 VHDL 文件，发现并纠正 VHDL 程序中的代码错误；生成功能模拟所需的文件(称为网表文件)，该文件不考虑电路延迟时间；再生成时序模拟所需的文件(也称为网表文件)，计入电路延迟时间，为可编程器件 FPGA 的编程作准备。

(5) 功能模拟。

(6) 时序模拟：如果发现错误、则返工，对改正后的电路重新编译。如果还不能解决问题，则可能返回到最开始的实验模型设计阶段。

在模拟之前除了准备好网表文件以外，还可为绘制元件的测试波形作准备，模拟时，操作人员可根据元件的输入和输出的信号标识符，在 Quartus Ⅱ 软件的引导下在计算机屏幕上给出输入点波形和显示时间，然后可自动形成完整的测试波形，显示输入输出信号的相互时间关系。

(7) 器件编程：为选择的 FPGA 芯片编程，完成实验模型。

Quartus Ⅱ 软件的内容很丰富，学习起来也不会很轻松，对本课程来讲还不属于基本要求，如有需要，可在后续课程或工作中学习。

2. VHDL 程序的层次化结构和库的概念引入

在教材第 12 章已对 VHDL 作了简单介绍，请学生务必先复习(例如实体和结构体等)。

下面以通用寄存器组的 VHDL 设计为例进行讨论。

图 15.3 中有 4 个 16 位寄存器、1 个译码器和 1 个选择器。译码器将 2 位寄存器地址 R_AD 译成控制其中 1 个寄存器写入数据的信号，通过选择器将数据 D_OUT 送出通用寄存器。

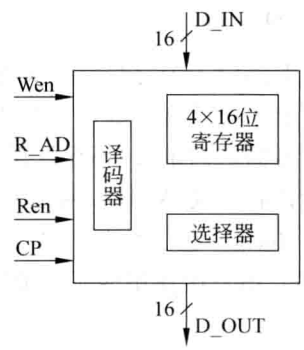


图 15.3 4×16 位通用寄存器组

上述的通用寄存器组可以被认为是一个元件，它与外界的连接信号(称为端口，**Port**)有输入数据 D_IN、输出数据 D_OUT、地址 R_AD、读允许信号 Ren 和写允许信号 Wen。也可以将它设计成层次化结构，将图中的寄存器、译码器和选择器设计为通用寄存器组(元件)的 3 个子元件，处于层次化结构中的低层。子元件的具体实现在低层描述。高层描述各子元件的连接条件和子元件之间的关系，完成通用寄存器组的设计。

VHDL 的层次结构设计对较大规模硬件的设计很有必要，而且层次数(2 个或 2 个以上)不受限制，元件的规模也不受限制，如果将设计的元件(或子元件)存放在 VHDL 的资源库中，则可被多次引用(称为例化)，其作用与程序运行时调用子程序类似。VHDL 设置有 IEEE 标准资源库，用户使用 VHDL 进行设计时，一般都需要调用库中的资源，就要用相应的 library(库)子句和 use 子句，放在实体和结构体的前面，所以用 VHDL 描述的电路模型由 3 部分组成：库、实体和结构体。具体内容在后面介绍的多个例子中予以说明。

3. VHDL 设计举例——译码器

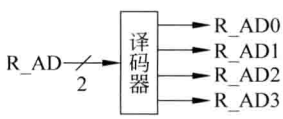


图 15.4 译码器

将图 15.3 中的译码器以独立的电路处理，所以其输入 R_AD 和输出 R_AD0、R_AD1、R_AD2 和 R_AD3 都成为端口信号(见图 15.4)，而在图 15.3 中，译码器的输出信号是通用寄存器组的内部信号，而不是端口(port)信号。

译码器的设计举例如下(设计实体 decoder 2_to_4)

```
library ieee                                    --打开 ieee 库(IEEE 资源库)
use ieee. std_logic-1164. all                   --打开 ieee. std. logic 1164 程序包,这是为了
                                                --使用 std_logic 和 std_logic_vector 数据类型
entity decoder 2_to_4 is port 1
    R_AD:    in    std_logic_vector (1 downto 0)
    R_AD0:   out   std_logic;
    R_AD1:   out   std_logic;
    R_AD2:   out   std_logic;
    R_AD3:   out   std_logic;
end decoder 2_to_4;
architecture decode of decoder 2_to_4 is
begin
    R_AD0  <=  (not R_AD(1)) and (not R_AD(0));
                                        --输出 R_AD i (i=0~3)中有 1 个为 1,其余 3 个为 0
    R_AD1  <=  (not R_AD(1)) and R_AD(0));
    R_AD2  <=  R_AD(1) and (not R_AD(0));
    R_AD3  <=  R_AD(1) and (not R_AD(0));
end decode
```

上例中各个符号的定义见 15.2.1 节。

上述设计有 3 部分:库、实体和结构体,其中的黑体字称为保留字或关键字,在 VHDL 中已有固定的定义,不允许设计者在他处作为标识符使用。符号--表示注释的开始,供读者或设计者阅读,VHDL 对注释不进行处理。标识符(上例中的实体名decoder 2_to_4、结构体名 decode 和信号 R_AD 等)由设计者命名,可用英文、数字和下划线"_"组成,但不能和 VHDL 语言中使用的所有关键字相同。关键字可用英文大写或英文小写表示。

15.2 VHDL 的基本使用规则

15.2.1 对象、数据类型和运算符

1. 对象

在 VHDL 中常用的对象有常量(constant)、信号(signal)和变量(variable)3 种。对象的一般表达形式如下:

<对象类型><标识符>:[<数据类型>][:=初始值]

需要注意的是:所有对象并不包含上述表达形式中的全部内容(方括弧中的内容是可选的)。

1) 常量

表达形式举例如下:

```
对象类型   标识符   数据类型   赋以初始值
 constant    abc:    integer      :=8;
```

设计 CPU 时，可将指令系统中的操作码作为常量，以便以后多次使用。

2）变量

表达形式举例如下。

① 对象类型　标识符　数据类型

variable　x,y,z:　integer

② 对象类型　标识符　数据类型(和数据范围)　初始值

variable　x,y,z:　integer **range** 0 **to** 7　:=0

3）信号

表达形式举例如下。

① 对象类型　标识符　数据类型

signal　sel:　std_logic

② 对象类型　标识符　数据类型(和数据范围)

signal　sel:　std_logic_vector (3 **downto** 0)

std_logic 表示信号 sel 是 1 位信号。std_logic_vector 表示 sel 信号有多位，可以使用其中的某些部分，上例 sel 有 4 位，sel(0)代表 sel 的第 0 位，sel(2)代表 sel 的第 2 位。

上述各个对象类型中的标识符可以是 1 个，也可以是多个，如果是多个，则要求它们的数据类型是相同的。

每个信号都有与之对应的电路存在，信号是硬件中各个电路之间的物理连线的抽象描述，在 VHDL 设计中，如果不使用信号，则设计不出任何一个实际电路。

在 15.1 节译码器设计中的端口(**port**)所涉及的对象都属于信号(**singal**)，其他电路的设计也是这样，因此省略了 **port** 中的 **signal** 字。并在数据类型中表明是输入(**in**)还是输出(**out**)。

2. 数据类型

在 VHDL 中已定义了多种标准数据类型。

(1) 整数 integer：表示范围为 $-(2^{31}-1)\sim(2^{31}-1)$。

(2) 字符 character：用单引号括起来，如'A'、'b'、'0'、'1'和'Z'，'Z'代表高阻态。

(3) 字符串 character string：用双引号括起来的字符串，例如"level"、"error"等。

(4) 位 bit：信号通常用位表示，用'0'或'1'表示。

(5) 位向量 bit_vector：用双引号括起来的一组位数据，例如"01100"、X"0A0B"等，X 表示后面的位向量是十六进制数据。

(6) std_logic：是多值逻辑，设计电路时经常用到的是'0'、'1'和'Z'。

(7) std_logic_vector：是由多位 std_logic 构成的向量。

进行电路设计时，用得最多的数据类型是 std_logic 和 std_logic_vector。

其他还有 boolean(布尔)、time(时间)等，不再介绍。

3. 运算符

对信号进行运算，适用的数据类型为 std_logic 和 std_logic_vector。

(1) 逻辑运算：有非(not)、与(and)、或(or)、异或(xor)、与非(nand)、或非(nor)、异或非(nxor)。

(2) 算术运算：有加、减、乘、除、指数(+、-、*、/、**)。

(3) 关系运算：等于(=)、不等(/=)、小于(<)、大于(>)、小于等于(<=)、大于等于(>=)。

(4) 赋值和连接

赋值符有两种，对常量和变量用符号“:=”进行赋值，对信号用“<=”进行赋值。

另外还有符号“>=”，对信号的作用将在后面用到时指出(见 case 语句和通用寄存器组高层设计中的 port map 语句)。

“<=”和“>=”在赋值和关系运算中都可以用到，但使用处不同，不易混淆。

连接符 & 用于位的连接，例如"1"&C，若 C 有 8 位，则"1"&C 有 9 位，"1"添加在 C 的前面。

15.2.2 VHDL 中使用的语句

VHDL 中有多种描述硬件行为的语句，本章介绍后面用到的 3 种语句：process 语句、if 语句和 case 语句。

VHDL 结构体中描述的行为之间存在并行执行关系和顺序执行关系，一般按并行执行。而 process(进程)语句内部用到的语句(例如 if 语句)则按顺序执行。process 语句代表一块功能独立的电路，一个结构体中可能描述多个电路，于是存在多个 process(进程)语句，这些进程之间存在并行执行关系。

1. if 语句

if 语句是 VHDL 提供的在 process(进程)内部执行的语句。有以下 3 种形式

```
① if 条件 1 then
      若干顺序执行语句
  end if;
② if 条件 2 then
      若干顺序执行语句
  else
      若干顺序执行语句
  end if;
③ if 条件 3 then
      若干顺序执行语句
  elsif 条件 4 then
      若干顺序执行语句
  …
  [else
      若干顺序执行语句]
  end if;
```

例：用 if 语句设计一个 D 触发器。

```
--存在 if 语句前面的描述，包括 signal d,cp,q std_logic 等
if cp'event and cp='1' then          --cp 发生跳变，然后处于高电平
    q<=d;                            --d 为触发器输入信号，q 为触发器输出
end if;
```

对于描述触发器来说，经常用到时钟 CP 的上升边和下降边，有下述 4 种表示方法。

```
cp' event and cp='1'      --上升边
cp' event and cp='0'      --下降边
rising_edge (cp)'         --上升边
falling_edge (cp          --下降边
```

当时钟信号发生跳变时，触发器按上例中的赋值语句 q<=d 接收数据，否则 q 保持原值不变。

2. process 语句

例：具有复位(reset)功能的 16 位寄存器。

```
--signal reset,cp,wen: std_logic
--signal d, q: std_logic_vector (15 downto 0)
process(reset,cp)                --reset 和 cp 为敏感信号
begin
    if reset='0' then
        q<=x "0000";             --x 表示十六进制
    elsif rising_edge(cp) then
        if wen='1' then          --wen 为允许写信号，它与 rising_edge (cp)的位置不能更换，如果更
                                 --换，VHDL 不能实现预定的功能
          q<=d;
        end if;
    end if;
end process;
```

敏感信号的状态发生变化时，启动 process 语句执行，使 process 语句代表的电路输出发生变化；如果敏感信号的状态不发生变化，则 process 内的程序不启动。敏感信号只能表示于 process 语句中。

3. case 语句

case 语句的表示格式如下：

```
case 表达式 is
    when 表达式值 1=>        --每一行中的"when"与符号"=>"同时出现
        若干顺序执行语句;    --当条件(表达式值 1)成立时，顺序执行的若干条语句
    when 表达式值 2=>
        若干顺序执行语句;
          ⋮
    when 表达式值 n=>
        若干顺序执行语句;
    when others=>
        若干顺序执行语句;
end case
```

表达式值 1～n 一般已包含表达式中的所有内容，不能遗漏，否则用 others(其他)代替。见下例。

例：运算器设计。

一个有加、减、移位(循环左移、循环右移)功能的 16 位运算器，A 和 B 是两个操作数，Q 是运算结果，C 为进位(借位)标志位，sel 是两位运算功能的选择码。

```
--Signal A,B,Q:std_logic_vector(15 downto 0);
--Signal sel:sta_logic_vector (1 downto 0);
--Signal result:std_logic_vector ( 16 downto 0);
--Signal C:std_logic;
process (A,B, sel, result, C)
begin
  case sel is
    when "00"=>                                  --加法运算
      result <='0'& A+'0'& B;                    --'0'为产生 C 而加入，result(结果)得 17 位
      Q<=result (15 downto 0);
      C<=result (16);
    when "01"=>                                  --减法运算
      result<=('0'&A)-('0'&B);
      Q<=result (15 downto 0);
      C<=result (16);
    when "10"=>                                  --循环左移，C 参加循环左移，result 得 17 位
      result<=A&'C';
      Q<=result (15 downto 0);
      C<=result (16);
    when others =>                               --循环右移，others(其他)可用"11"替代
      result<=A(0) &'C'&A(15 downto 1);
      Q<=result (15 downto 0);
      C<=result (16);
  end case;
end process;
```

case 语句有下述特点：满足某一个 when 条件的语句是顺序执行的，上述的 result、Q、C 是按先后顺序执行的；而所有的 when 是并行执行的，即上例中的 when "00"、when "01"…等是并行执行的。在一个 case 程序中，各个 when 表达式值之间不可能出现相同的表达式值，而且又不能遗漏一个表达式值。

15.2.3 设计举例——通用寄存器组的设计

1. 库和层次结构

在设计时，一般可能用到 VHDL 的库和层次结构设计方法，在此先进行解释。

1) 库(library)

VHDL 中的库用来存储可编译的设计单元(实体说明、结构体和程序包等)，可以用作其他 VHDL 设计的资源。VHDL 库分为设计库和资源库两种，std 库和 work 库是设计库。

std 库为所有设计单元所共享，是使用 VHDL 时一定用到的工具，在 std 程序包中定义了若干数据类型等。

work 库是存放用户在设计中成功的成品或半成品，为正在进行的设计服务。如果希望

该成果为以后其他项目所使用，则应将这些设计单元编译到相应的资源库中。

除了 std 库和 work 库之外，其他库均为资源库，被 IEEE 认可的资源库称为 ieee 库。VHDL 和 EDA 工具专业公司还有自己的资源库。

使用资源库中的元件或函数之前，需要使用 library 子句和 use 子句，在通用寄存器的设计中，使用了 std_logic 和 std_logic_vector 两种数据类型，这两种数据类型又放在 std_logic_1164 中，因此使用了下面的语句：

```
library ieee;
use ieee. std_logic_1164. all          --打开程序包 std_logic_1164. all 后，可使用
                                       --std_logic 和 std_logic_vector 类型
```

在设计中经常用到的语句还有(用于 std_logic_vector 数据类型的加减运算)：

```
use ieee std_logic_unsigned. all
```

work 库是 VHDL 的工作库，用户在工程中已设计好的中间件放在 work 库中，是一个临时的仓库。

2) 程序包(package)

程序包是一种使包体中的类型、常量和元件等可被调用的设计单元，见 15.3 节设计 CPU 的例子。

3) component 语句和 port map 语句

component(元件)和 port map(端口映射)配合使用，实现较高层设计实体对较低层设计实体的引用。

(1) component 语句指明引用的元件名，表示格式如下：

```
component 元件名                       --指出引用的元件名
  [generic 说明;]                      --参数传递说明，例如向量长度、延迟时间
  [port 说明;]                         --与 entity 语句中的 port 作用相同
end component
```

(2) port map 语句将较低层次设计实体(元件)的端口映射为较高层设计实体中的信号，元件的端口相当于电原理图中的输入/输出端，各元件在电原理图中就要用线连接起来，这些连接的线在 VHDL 中就是信号。

port map 的表示格式如下：

```
[标号]: port map(端口名 1=>信号名 1,端口名 2=>信号名 2,…,端口名 n=>信号名 n);
```

将端口名 1 映射到信号名 1，……，端口名 n 映射到信号名 n。例见通用寄存器组的层次结构设计。

[标号]是为了便于读懂或记忆而引入的，也可以不存在。

上述两个语句的使用方法可参阅下面的举例：通用寄存器 regfile 设计。

2. 通用寄存器组(4×16 位)的设计

图 15.5 是通用寄存器组(regfile)的逻辑简框图，输入信号有 D_IN、Wen、Ren、R_AD、CP，输出信号有 D_OUT。其中 R_AD 是寄存器地址(2 位)，CP 为时钟。

在寄存器框图的内部，译码器的输出 R_AD0、R_AD1、R_AD2 和 R_AD3 信号分别选择

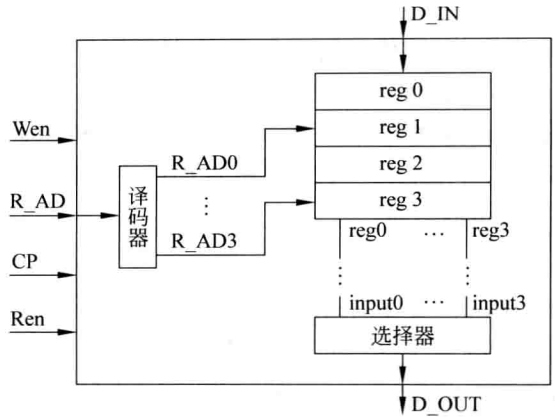


图 15.5　4×16 位通用寄存器组逻辑简框图

相应寄存器 reg0～reg3 的写入，并可通过选择器将选中的寄存器内容输出(D_OUT)。而真正完成写入和读出操作的还分别受 Wen 和 Ren 控制，如果 Ren＝'0'，则 D_OUT＝X"0000"，X 表示十六进制(数据)。

下面介绍通用寄存器组的两种设计方法。第 1 种方法不分层次结构；第 2 种方法采用双层次结构，低层次由译码器、选择器和一个寄存器读写的设计实体组成，高层设计实体 regfile 完成 3 个低层设计实体的连接，完成寄存器组的全部功能。

例 1：通用寄存器组的设计(不分层次结构)。

```
library ieee;
use ieee. std_logic_1164. all;
entity regfile is
    port (R_AD        :in  std_logic_vector(1 downto 0);
          Wen,Ren,cp :in  std_logic;
          D_IN        :in  std_logic_vector (15 downto 0);
          D_OUT       :out std_logic_vector (15 downto 0)
          );
end regfile;
architecture reg_RW of regfile is
  signal reg0,reg1,rge2,rge3:std_logic_vector (15 downto 0);
                                                    --非端口信号，但在结构体中用到，因此列出
begin
  process(cp)
  begin
    if (cp' event and cp='1') and (Wen='1') then
      case R_AD is
        when "00"=> reg0<=D_IN after 10ns;        --after 的含义见本例结束后的说明
        when "01"=> reg1<=D_IN after 10ns;
        when "10"=> reg2<=D_IN after 10ns;
        when others => reg3<=D_IN after 10ns;
      end case;
    endif;
  end process;
```

```
  process (cp)
  begin
    if (cp' event and cp=1) and (Ren='1') then
    case R_AD is
        when "00"=> D_out<=reg0;
        when "01"=> D_out<=reg1;
        when "10"=> D_out<=reg2;
        when others => D_out<=reg3;
      end case;
    end if;
  end process;
end reg_RW;
```

说明：在 VHDL 语句中，对信号进行处理都需要时间，上例中的“after 10ns”表示完成该操作需要经过 10ns 延迟时间。如果在语句中未表明，则按 VHDL 默认的延迟时间处理。

例 2：通用寄存器组 regfile 设计(双层结构)。

由 5 部分组成，其中的 1)～3)为低层设计，完成通用寄存器中全部子元件的设计。

1) 寄存器 reg

```
library ieee;
use ieee. std_logic_1164. all;

entity reg is port
    ( D_IN    :in   std_logic_vector (15 downto 0);   --数据输入
      CP      :in   std_logic;                        --时钟
      Wen     :in   std_logic;                        --允许写
      R_ADi   :in   std_logic;                        --被选中的寄存器
      reg_AD  :out  std_logic_vector (15 downto 0)    --写入被选中寄存器的数据
    );
end reg;

architecture reg_write of reg is
begin
  process(cp)
  begin
        if cp' event and cp='1' then                  --时钟上升沿触发
           if R_ADi='1' and Wen='1' then              --只有被选中且允许写时才将数据写入寄存器
           reg_AD<=D_IN;
        end if;
    end if;
  end process;
end reg_write;
```

2) 2-4 译码器 decoder_2_to_4

```
library ieee;
```

```
use ieee. std_logic_1164. all;

entity decoder_2_to_4 is
    port (
        R_AD   :in   std_logic_vector (1 downto 0);
        R_AD 0 :out  std_logic;
        R_AD 1 :out  std_logic;
        R_AD 2 :out  std_logic;
        R_AD 3 :out  std_logic;
        );
end decoder_2_to_4;

architecture behavioral of decoder_2_to_4 is
begin
    R_AD 0<= (not R_AD (1)) and (not R_AD(0));
    R_AD 1<=(not R_AD (1)) and R_AD(0);
    R_AD 2<=R_AD (1) and (not R_AD(0));
    R_AD 3<=R_AD (1) and R_AD(0);
end behavioral;
```

3）4 选 1 选择器 mux_4_to_1

```
library ieee;
use ieee. std_logic_1164. all;

entity mux_4_to_1 is
prot (
      input0, input1,input2,input3 :in  std_logic_vector(15 downto 0);  --译码器的输入
      R_AD                         :in  std_logic_vector(1 downto 0);   --寄存器地址
      D_OUT                        :out std_logic_vector(15 downto 0);  --译码器输出,也是
                                                                        --regfile 的输出
      Ren                          :in  std_logic                       --允许读
      );
end mux_4_to_1;

architecture behavioral of mux_4_to_1 is
begin
    mux:process(R_AD, input0,input1,input2,input3):                     --mux 是标号,不是必备的
    begin
      if Ren='0' then
        D_OUT<=X "0000";
      elsif Ren='1' then
        case R_AD is
            when "00"=>
                D_OUT<=input0;
            when "01"=>
```

```
            D_OUT<=input1;
        when "10"=>
            D_OUT<=input2;
        when "11"=>
            D_OUT<=input3;
    end case;
  end if;
end process max;
end behavioral;
```

4）程序包 exp_reg_components

建立程序包，存放在用户工作库（work）中，供以后的设计或其他用户调用。在这里提供给后面的高层设计使用。

```
library ieee;
use ieee.std_logic_1164.all;

package exp_reg_components is
    component reg port
        (
          reset    :    in   std_logic;
          D_IN     :    in   std_logic_vector(15 downto 0);
          cp       :    in   std_logic;
          Wen      :    in   std_logic;
          R_ADi    :    in   std_logic;
          reg_AD   :    out  std_logic_vector(15 downto 0)
        );
    end component;

    component mux_4_to_1 port
        (
          input0,input1,input2,input3 :  in   std_logic_vector(15 downto 0);
          R_AD                        :  in   std_logic_vector(1 downto 0);
          D_OUT                       :  out  std_logic_vector(15 downto 0);
          Ren                         :  in   std_logic
        );
    end component;

    component decoder_2_to_4 port
        (
          R_AD  :  in   std_logic_vector(1 downto 0);
          R_AD0 :  out  std_logic;
          R_AD1 :  out  std_logic;
          R_AD2 :  out  std_logic;
          R_AD3 :  out  std_logic;
```

```
        );
    end component;
end exp_reg_components;
```

5）通用寄存器组 regfile（高层设计）

```
library ieee;
use ieee. std_logic_1164. all;
use work. exp_reg_components. all;
                                                    --打开设计者存放在工作库中的程序包

entity regfile is
port ( R_AD     :in   std_logic_vector(1 downto 0);   --寄存器地址
       Wen,Ren :in   std_logic;                       --允许写寄存器信号,允许读寄存器信号
       cp       :in   std_logic;                      --时钟
       D_IN     :in   std_logic_vector(15 downto 0);  --写寄存器的数据
       D_OUT   :out std_logic_vector(15 downto 0);   --输出数据
    );
end regfile;

architecture struct of regfile is

signal reg0,reg1,reg2,reg3:std_logic_vector(15 downto 0);
signal R_AD0,R_AD1,R_AD2,R_AD3:std_logic;

begin
    Arge0:reg port map(                              --地址为 0 的寄存器,Areg0 为标号
        D_IN    --=> D_IN,                           --=>将其左边的元件端口名和右边的高层设计
                                                     --实体中的信号名对应起来
        cp      => cp,
        Wen     => Wen,
        R_ADi   => R_AD0,
        reg_AD  => reg0
        );

    Argel:reg port map(                              --地址为 1 的寄存器,Areg1 为标号
        D_IN    => D_IN,
        cp      => cp,
        Wen     => wen,
        R_ADi   => R_AD1,
        reg_AD  => reg1
        );

    Arge2:reg port map(                              --地址为 2 的寄存器,Areg2 为标号
        D_IN    => D_IN,
        cp      => cp,
        Wen     => wen,
        R_ADi   => R_AD2,
        reg_AD  => reg2
        );
```

```
Arge3:reg port map(                          --地址为 3 的寄存器,Areg3 为标号
        D_IN    => D_IN,
        cp      => cp,
        wen     => wen,
        R_ADi   => R_AD3,
        reg_AD  = >reg3
        );

des_decoder:decoder_2_to_4 port map(         --2-4 译码器,des_decoder 为标号
        R_AD        => R_AD,                 --=>将其左边的元件端口名和右边的高层
        R_AD0       => R_AD0,                --设计实体中的信号名对应起来
        R_AD1       => R_AD1,
        R_AD2       => R_AD2,
        R_AD3       => R_AD3
        );
Amux: mux_4_to_1 port map(                   --寄存器读出使用的 4 选 1 选择器,Amux 为标号
        input0   => reg0,
        input1   => reg1,
        input2   => reg2,
        input3   => reg3,
        R_AD     => R_AD,
        D_OUT    => D_OUT
        Ren      => Ren
        );
end struct;
```

分析上述两种设计方法,由于寄存器组 regfile 的设计内容简单,所以不值得采用双层结构的设计。选择此例是从学习的目的出发,由于设计简单,所以便于理解。假如设计复杂的电路则可将低层的元件分配给多人设计,相互进行合作 ,或多次被引用(称为例化),从而提高设计效率。

15.3 CPU 模型设计举例

1. CPU 模型总体设计

本章设计的 CPU 总体结构基本上与第 14 章的 CPU 保持一致,请参阅第 14 章的 14.2.2 节,下述内容尤其需要特别注意:

(1) 指令和数据长度都是 16 位,指令系统由 9 条指令组成,每条指令的操作码和地址码的安排以及寻址方式等都与第 14 章的 CPU 相同。

(2) 每条指令执行的机器周期都是 3 个,在第 14 章中用 T1、T2 和 T3 表示,在本章中用状态 status 表示(status 1,status 2 和 status 3)。

(3) 第 14 章的指令执行流程图(图 14.11)和 CPU 逻辑图(图 14.12)是本章用 VHDL 设计 CPU 的主要参考资料。

本设计采用 VHDL、Quartus Ⅱ 软件和 FPGA 器件,CPU 模型的全部内容都在一个 FPGA 器件中实现。指令执行流程与图 14.11 和图 14.12 比较有相应的细节变化。

2. 用 VHDL 设计的 CPU 模型

在本设计中首先定义了 op_code(操作码)程序包,用助记符代表二进制操作码,其目的是让设计者或读者容易阅读或记忆操作码。

在实体为 CPU 的结构体(struct of cpu)中有两个进程(process)。第一个进程描述机器周期的变化,总清时处于状态 0(status 0),机器启动后执行指令,都在 T1、T2 和 T3 之间循环运动。第二个进程描述指令执行流程(seq 标识符)。

EDA 工具对 VHDL 的 cpu 程序处理后,可得到相应的电路逻辑图,并进行功能模拟和时序模拟,对信号赋值时(<=)应加上延迟时间,在本设计中为了书写简化而没有标出延迟时间,考虑延迟是很重要的。

用 VHDL 设计的 CPU 模型程序如下:

```
Library ieee;                                  --打开 ieee 库
use ieee. std_logic_1164. all;                 --打开程序包 std_logic_1164. all 以后可使用 std_logic 和
                                               --std_logic_vector 类型
package op_code is                             --定义 op_code 程序包,用助记符代表二进制操作码
  constant add   : std_logic_vector(3 downto 0):="0000";      --加法
  constant sub   : std_logic_vector(3 downto 0):="0010";      --减法
  constant rol   : std_logic_vector(3 downto 0):="0100";      --循环左移
  constant ror   : std_logic_vector(3 downto 0):="0110";      --循环右移
  constant store : std_logic_vector(3 downto 0):="1000";      --存数
  constant load  : std_logic_vector(3 downto 0):="1010";      --取数
  constant brc   : std_logic_vector(3 downto 0):="1100";      --条件(c)转移
  constant brz   : std_logic_vector(3 downto 0):="1100";      --条件(z)转移
  constant jmp   : std_logic_vector(3 downto 0):="1110";      --无条件转移
  constant stop  : std_logic_vector(3 downto 0):="1111";      --停机

end op_code;

Library ieee;
use ieee. std_logic_1164. all
use ieee. std_logic_unsigned. all                --打开 std. logic_unsigned 程序包,
                                                 --以后可使用'+'、'-'运算符
use work. op_code. all                           --打开自定义程序包 op_code
entity cpu is                                    --cpu 实体
  port(                                          --端口信号
    reset     : in std_logic;                    --总清信号,低电平有效
    cp        : in std_logic;                    --时钟
    write     : out std_logic;                   --CPU 发给存储器的写信号
    read      : out std_logic;                   --发给存储器的读信号
    MAR       : out std_logic_vector(9 downto 0);  --发给存储器的地址
    MDR_IN    : in std_logic_vector(15 downto 0);  --从存储器送到 CPU 的数据或指令
    MDR_OUT : out std_logic_vector(15 downto 0));  --发送给存储器的数据
```

```
end cpu;

architecture struct of cpu is                          --cpu 结构体
    signal IR: std_logic_vector(15 downto 0);          --指令寄存器
    signal status: integer range 0 to 3;               --cpu 的状态(周期)

begin
status_change: process(reset,cp,status)                --此进程描述状态(周期)变化,status_change 为标号
  begin
    if reset='0' then statas <=0;                      --仅在总清时,进入 0 状态。
    elsif cp'event and cp='0' then
        case status is
          when 0=>
            status<=1;                                 --图 14.11 中指令执行流程 T 即为此处的 status
          when 1=>
            status<=2;
          when 2=>
            status<=3;
          when 3=>
            status<=1;
          when others=> status<=1;
        end case;
    else
      null;
  end if;
end process status_change;
seq:process(reset, cp)                                 --此进程描述指令执行过程,seq 为标号
  signal pc                  : std_logic_vector(9 downto 0);   --程序计数器,10 位
  signal reg0,reg1,reg2,rge3 : std_logic_vector(15 downto 0);  --4 个通用寄存器
  signal A,B                 : std_logic_vector(15 downto 0);  --送数据到 ALU 的寄存器 A,B
  signal temp                : std_logic_vector(16 downto 0);  --暂存数据
  signal z,c                 : std_logic;                      --标志位: z(结果为 0),c(进位/借位)

begin
  if reset='0' then                                    --置初始状态
    pc<=(others =>'0');                                --pc 置成 0,加电后 CPU 执行的
    MAR<=(others =>'0');                               --第一条指令地址 MAR 置成 0
  elsif(cp' event and cp='1') then
    z<='0';                                            --z 等于 0
    c<='0';                                            --c 等于 0
    case status is
      when '0'=>null;                                  --状态 0,无操作
      when '1'=>                                       --状态 1(status 1)
        IR<=MDR_in;                                    --取指令
```

```
    pc:=pc+1;
  when '2'=>                                    --状态 2(status 2)
    case IR(15 downto 12) is
      when add|sub|rol|ror|=>                   --加法/减法/左移/右移指令
        case IR(11 downto 10) is                --图 14.12 中的 R1 地址
          when "00"=>      A<=reg0;
          when "01"=>      A<=reg1;
          when "10"=>      A<=reg2;
          when "11"=>      A<=reg3;
        end case;
      when add|sub=>                            --加法/减法
        case IR(9 downto 8) is                  --图 14.12 中的 R2 地址
          when "00"=>      B<=reg0;
          when "01"=>      B<=reg1;
          when "10"=>      B<=reg2;
          when "11"=>      B<=reg3;
        end case;
  when load/store=>                             --取数/存数
      MAR<=IR(9 downto 0);
  when store=>                                  --存数
    case IR(11 downto 10) is
      when "00"=>      MDR_out<=reg0;
      when "01"=>      MDR_out<=reg1;
      when "10"=>      MDR_out<=reg2;
      when "11"=>      MDR_out<=reg3;
    end case;
  when jmp|brc|brz|stop=>                       --无条件转移/条件(c)转移/条件(z)转移/停机
    A<="000000"& pc;                            --PC 前面添 6 位'0',送 A
  when jmp=>                                    --无条件转移
    B<=X "000"& IR(11 downto 8);                --生成 disp
  when brc =>                                   --条件(c)转移
    if c=1 then
      B<=X "000"& IR(11 downto 8);
    elsif c=0 then
      B<=X "0000";
    end if;
  when brz =>                                   --条件(z)转移
    if z=1 then
      B<=X "000"& IR(11 downto 8);
    elsif z=0 then
      B<=X "0000";
    end if;
  when stop =>                                  --停机
      B<=X "0001";
```

```
      when other => null;
    end case;
  when '3'=>                                          --状态 3(status 3)
    case IR(15 downto 12) is
      when add| sub=>                                 --加法或减法
        if IR(13)=0 then                              --加法
          temp <=(0&A)+(0&B);                         --temp 有 17 位,最高位为 c
        elsif IR(13)=1 then                           --减法
          temp<=(0&A)-(0&B);
        end if;
        case IR(11 downto 10) is
            when "00">=      reg0<=temp(15 downto 0);
            when "01">=      reg1<=temp(15 downto 0);
            when "10">=      reg2<=temp(15 downto 0);
            when "11">=      reg3<=temp(15 downto 0);
        end case;
        c<=temp (16);
        z<=(not temp(15)) and(not temp(14)) and
             (not temp(13)) and(not temp(12)) and
             (not temp(11)) and(not temp(10)) and
             (not temp(9)) and(not temp(8)) and
             (not temp(7)) and(not temp(6)) and
             (not temp(5)) and(not temp(4)) and
             (not temp(3)) and(not temp(2)) and
             (not temp(1)) and(not temp(0));
        MAR<=pc;
        when rol|ror=>                                --左移或右移
          if IR(13)=0 then                            --左移
            temp<=A(14 downto 0)&c;
            c<=A(15);                                 --右移
          elsif IR(13)=1 then
            temp<=c&A(15 downto 1);
            c<=A(0);
          end if;
        case IR(11 downto 10) is
          when "00">=      reg0<=temp(15 downto 0);
          when "01">=      reg1<=temp(15 downto 0);
          when "10">=      reg2<=temp(15 downto 0);
          when "11">=      reg3<=temp(15 downto 0);
        end case;
        MAR<=pc;
          when store=>                                --存数
            write <='1';                              --写存储器信号
            MAR<=pc;
```

```
        when load=>                                  --取数
          read<='1'
          case IR(11 downto 10) is
            when '00'>=reg0<=MDR_in;
            when '01'>=reg1<=MDR_in;
            when '10'>=reg2<=MDR_in;
            when '11'>=reg3<=MDR_in;
          end case;
          MAR <=pc;
        when jmp|brc|brz=>                           --转移指令
          temp<=A+B;
          pc<=temp(9 downto 0);
          MAR<=temp(9 downto 0);
        when stop=>                                  --停机指令
          temp<=A-B;
          pc<=temp(9 downto 0);
          MAR<=temp(9 downto 0);
        when others=>NULL;
      end case;
    end case;
  end process seq;
end struct;
```

CPU 设计已完成，后面的工作就由 EDA 工具完成。

小　结

本章讲述的是利用 VHDL 语句和相关的 EDA 工具对设计人员提供的逻辑简框图进行分析和 VHDL 程序设计，最后完成 CPU 模型的设计。后续的 CPU 模型实现工作主要由 EDA 工具完成，在本章中简单提出，不进行讨论。

1. VHDL 程序的组成

VHDL 程序由库(library)、实体(entity)和结构体(architecture)3 部分组成，无论设计对象简繁程度如何，三者缺一不可。

2. VHDL 中使用的语句

VHDL 中使用的语句很多，在本章中仅用到 3 种语句：process、if 和 case。每一种语句中用到多个保留字(或称为关键字)，例如 if 语句中可用到 if、elsif、else 和 end if。

VHDL 中可使用的其他语句还有十几种，例如 when-else(条件型赋值)、with-select(选择型赋值)、for-loop(循环语句)和 exit(结束循环语句)等。当需要了解其用法时，可参阅讲解 VHDL 语言的书籍或公司提供的说明书。

从计算机的研究和生产的发展过程来看，计算机是从硬件发展开始的，VHDL 语言属于计算机辅助设计(CAD)领域的应用软件，协助设计人员进行计算机硬件的逻辑设计工作，CPU 和其他部件(存储器、输入输出部件的控制电路)均由逻辑电路(门电路)和触发器

组成，因此决定了 VHDL 语言中有适用于并行处理的语句和顺序执行的语句，同时引入了决定触发器状态的信号上升沿/下降沿的表示方式。

3. 第 14 章与第 15 章的关系

利用 VHDL 语言进行计算机或其他设备（工业控制机和移动设备等）的逻辑设计工作可提高设计自动化程度和效率，并减少设计的差错，同时在 EDA 工具的配合下为后续工作做好准备。但是要真正学会 VHDL 语言的使用和改进系统的逻辑结构，还需要对硬件有真实的理解，而且从本课程的培养目标与实验时间的安排来讲，首先要保证完成第 14 章提出的实验和设计要求。对不同专业（例如计算机软件和微电子专业）的学生，其学习的重点可以不同。

参考文献

[1] 王爱英. 计算机组成与结构. 5 版. 北京：清华大学出版社，2013
[2] 王爱英. 计算机组成与结构. 4 版. 北京：清华大学出版社，2007
[3] 王爱英. 计算机组成与结构(第 4 版)习题详解与实验指导. 北京：清华大学出版社，2007
[4] 王诚，等. 计算机组成与设计实验指导. 北京：清华大学出版社，2002
[5] 王尔乾，等. 数字逻辑与数字集成电路. 2 版. 北京：清华大学出版社，2002
[6] 薛宏熙，胡秀珠. 计算机组成与设计. 北京：清华大学出版社，2007
[7] 汤志忠，杨春武. 开放式实验 CPU 设计. 北京：清华大学出版社，2007

清华大学计算机系列教材

本套教材已伴随着计算机科学与技术的发展茁壮成长了三十余年，获得了中华人民共和国教育部科技进步奖、普通高等学校优秀教材全国特等奖、全国优秀畅销书金奖等三十多项部级以上奖励，被近千所高校选作教材，教学效果非常好。本套教材经过多次修订改版和增加新品种、新内容、新技术，基本涵盖了本科生和硕士研究生的主要课程。本套教材的作者全部是清华大学计算机系的教师，教材的内容、语言特点、课时安排体现了他们治学严谨的特点，概念表述严谨，逻辑推理严密，语言精练。同时，本套教材体系完整、结构严谨，理论结合实际，注重素质培养。

配套资源

本书是《计算机组成与结构（第5版）》（书号：9787302290117）的配套教材。

清华大学出版社数字出版网站
WQBook 书文局泉
www.wqbook.com

ISBN 978-7-302-32976-3
9 787302 329763 >
定价：29.00元