****

实践教学

****

兰州理工大学

计算机与通信学院

2021 年春季学期

计算机组成原理课程设计

题　　目：模型机设计—9

专业班级：

姓　　名：

学　　号：

指导教师：谢鹏寿

成　　绩：

前　言

计算机组成原理课程是计算机系很重要的一门专业基础课，从这门课的内容特点看，它属于工程性、技术性和实践性都很强的一门课，因此，在进行课堂教学的同时，必须对实验教学环节给予足够的重视，要有良好的实验环境，能进行反应主要教学内容的、水平确实比较高的实验项目，在深化计算机各功能部件实验的同时，加强对计算机整机硬件系统组成与运行原理有关内容的实验；在教学实验的整个过程中，坚持以硬件知识为主的同时，加深对计算机整机系统中软硬件的联系与配合的认识。目前，有些单位和院校都研制出一些用于计算机组成原理课程教学实验的系统或装置，也各具特色但基本上都是相对孤立的功能部件的实验，整机硬件方面的实验很难胜任，更不能对计算机系统中硬软件的联系和配合的学习提供足够的帮助。而**DVCC**系列实验计算机系统就是专为计算机组成原理课的授课和教学实验而研制的。

DVCC系列计算机组成原理实验系统作为较高层次、专用于计算机原理课程教学实验的实验计算机系统具有良好的实验性能和系统的完整性以及可扩展性。

良好的实验性体现在**DVCC**系列机能很好地完成计算机硬件系统各功能部件的教学实验，它包括运算器部件、控制器部件、主存储器部件、总线和几种最重要的外设接口实验，包括中断、定时计数器、输入/输出接口等；计算机的**CPU**自行设计与实现，配有小的监控程序，有自己的汇编语言的支持。在相应软件的配合下，将各功能部件有机的结合起来，完成计算机整机的实验。

系统的完整性体现在 **DVCC** 系列机与学生常见到的简单计算机大体相同，其主要组成与运行方式和 **PC** 机差不多，该系列机是一台硬软件相对完整、配置巧妙合理的完整的计算机系统，通过它能体现出重要教学内容、能完成主要教学实验项目。

目　录

摘　要

该基本模型机是基于 DVCC 实验箱设计的一款能够实现简单算术运算的模型机。存储系统使用模型机的存储模块，运算器使用模型机的器件，带有片间串行进位 16 位算数逻辑运算功能，微程序控制器模块使用教学机的系统。指令系统包括输入输出指令 IN 、OUT，存取指令 STA、LDA，访问指令及转移指令 JMP、BZC、CLR、MOV、,算数运算指令 ADD 、SUB、DEC,逻辑运算指令 COM。

关键词：模型机；算术运算；指令设计

第1章　模型机设计概述

1.1　设计目的

1.在掌握部件单元电路实验的基础上，进一步将其组成系统构造一台复杂模型计算机。

2.为其定义几条机器指令，并编写相应的微程序，具体上机调试掌握整机概念。

3、学习设计和调试计算机的基本步骤和方法，提高使用软件仿真工具和集成电路的基本技能。

根据计算机组成原理课程所学知识，设计、开发一套简单的模型计算机。通过对一个简单计算机的设计，以达到对计算机的基本组成、部件的功能与设计、微程序控制器的设计、微指令和微程序的编制与调试等过程有更深的了解，加深对理论课程的理解。通过模型机的设计和调试，连贯运用计算机组成原理课程学到的知识，建立计算机整机概念，加深计算机时间和空间概念的理解。

部件实验过程中，各部件单元的控制信号是人为模拟产生的，如运算器实验中对74LS181芯片的控制，存储器中对存储器芯片的控制信号，以及几个实验中对输入设备的控制。这里，计算机数据通路的控制将由微程序控制器来完成，CPU 从内存中取出一条机器指令到指令执行结束的一个指令周期全部由微指令组成的序列来完成，即一条机器指令对应一段微程序。

本系统使用两种外部设备，一种是二进制代码开关(DATA UNIT)，它作为输入设备；另一种是发光二极管(BUS UNIT 上的一组发光二极管)，它作为输出设备。例如：输入时，二进制开关数据直接经过三态门送到总线上，只要开关状态不变，输入的信息也不变。输出时，将输出数据送到数据总线 BUS 上，驱动发光二极管显示。本实验在实验七基本模型机的基础上增加移位控制电路，实现移位控制运算。

1.2　设计任务

以教学实验用模型机为背景，通过调研、分析现有的模型机，建立带有带 8 位移位运算指令的整机模型。

1.3　设计原理

本系统使用两种外部设备，一种是二进制代码开关(DATA UNIT)，它作为输入设备；另一种是发光二极管(BUS UNIT 上的一组发光二极管)，它作为输出设备。例如：输入时，二进制开关数据直接经过三态门送到总线上，只要开关状态不变，输入的信息也不变。输出时，将输出数据送到数据总线 BUS 上，驱动发光二极管显示。

其中 IN 为单字长（8 位），其余为双字长指令，xxxxxxxx 为 addr 对应的二进制地址码。

微控器读取一条机器指令后，将通过如下的逻辑电路，对 SE1～SE5 中的某一位或者几位激活，从而实现机器指令与微程序的对应。当然，该逻辑电路还能接收外部控制输入 SWA、SWB，内部状态输出 FC、FZ 等信号，并对这些信号给出相应的输出。

为了向 RAM 中装入程序和数据，检查写入是否正确，并能启动程序执行，还必须设计三个控制台操作程序。

存储器读操作（KRD）：拨动总清开关 CLR 后，控制台开关 SWB、SWA 为“0 0”时，按 START 微动开关，可对 RAM 连续手动读操作。

存储器写操作（KWE）：拨动总清开关 CLR 后，控制台开关 SWB、SWA 为“0 1”时，按 START 微动开关，可对 RAM 连续手动写入。

启动程序：拨动总清开关 CLR 后，控制台开关 SWB、SWA 为“1 1”时，按 START 微动开关，即可转入到第 01 号“取指”微指令，启动程序运行。

上述三条控制台指令用两个开关 SWB、SWA 的状态来设置，得 SWB、SWA 定义表，如表 2 所示。

表 1.1 SWB、SWA 定义

SWB	SWA	控制台指令
0	0	读内存（KRD）
0	1	写内存（KWE）
1	1	启动程序（RP）

1.4 实验设备与器材

DVCC 试验箱，74lS181 四位算术逻辑单元/函数发生器，暂存器 74LS273，输出缓冲/显示驱动 74LS245，移位寄存器 74LS299，4 位二进制计数器 74LS161，E^2PROM 2816 芯片，6264。

第2章　模型机总体设计

2.1　模型机的逻辑结构

运算器模块主要由运算器 74LS181、暂存器 74LS273、输出缓冲器 74LS245 以及进位控制和判零标志控制电路等构成。移位寄存器采用 74LS299，它具有并行接数、逻辑左移、逻辑右移、保持等功能，具体由 S0、S1、M、DS0、DS7 决定。T4 是它的工作脉冲，正跳变有效。寄存器堆模块为实验计算机提供了 4 个 8 位通用寄存器。它们用来保存操作数及其中间运算结果，它对运算器的运算速度、指令系统的设计等都有密切的关系。程序计数器 PC（8 位）由二片可预置的 4 位二进制同步计数器 74LS161 构成，它具有接数、计数、清零等功能。程序计算器的输出采用三态传输器件 74LS245。地址寄存器部分由地址寄存器和地址显示灯构成。地址寄存器采用 74LS273，它的输入直接连到系统总线 BUSD0～D7 上，输出直接接到程序存储器 6264 的地址输入端 AD0～AD7，输出为三态。

指令寄存器模块中指令寄存器 74LS273 的输出部分以排针形式引出到 IJ1，部分内部已连好， 构成实验计算机机时用它作为指令译码电路的输入，实现程序跳转控制。微程序控制器模块主要由微程序编程器、核心微控制器两部分组成。

微程序编程器就是将预先定义好的机器码对应的微代码程序写入到 E^2ROM 2816 控制存储器中，并可以对控制存储器中的数据进行校验。本系统具有本机现场直接编程功能，且由于选用 E^2ROM 2816 芯片为控制存储器，因此，具有掉电保护功能。主存储器单元电路主要用于存放实验中的机器指令，它的数据总线挂在扩展数据总线 EXD0～EXD7 上；它的地址总线由地址寄存器单元电路中的地址寄存器 74LS273 给出，地址值由 8 个 LED 灯 LAD0～LAD7 显示，高电平亮，低电平灭；它的读信号直接接地；它的写信号和片选信号由写入方式确定。

2.1.1　运算器

在该运算器中，有两片 74LS181 组成算术和逻辑运算。数据的来源由 74LS273 寄存器提供，74LS273 产生 8 位数据，分别送入到 74LS181 运算器中进行相应的运算，而如何进行数据的传送是由 LDDR1 和 LDDR2 以及 T4 信号控制的，当 LDDR1 和 T4 都为高电平时，选定相应的寄存器来进行数据输入，同理，LDDR2 和 T4。然后经过相应的运算之后将产生的结果通过总线送回到寄存器中。整个数据的运送过程有相应的控制信号提供，S0、S1、 S2、S3、S4、M 都是通过控制器的相关指令来控制。让其进行某种算数运算和逻辑运算。整个数据和指令都是通过数据总线，控制总线和地址总线来进行传送。

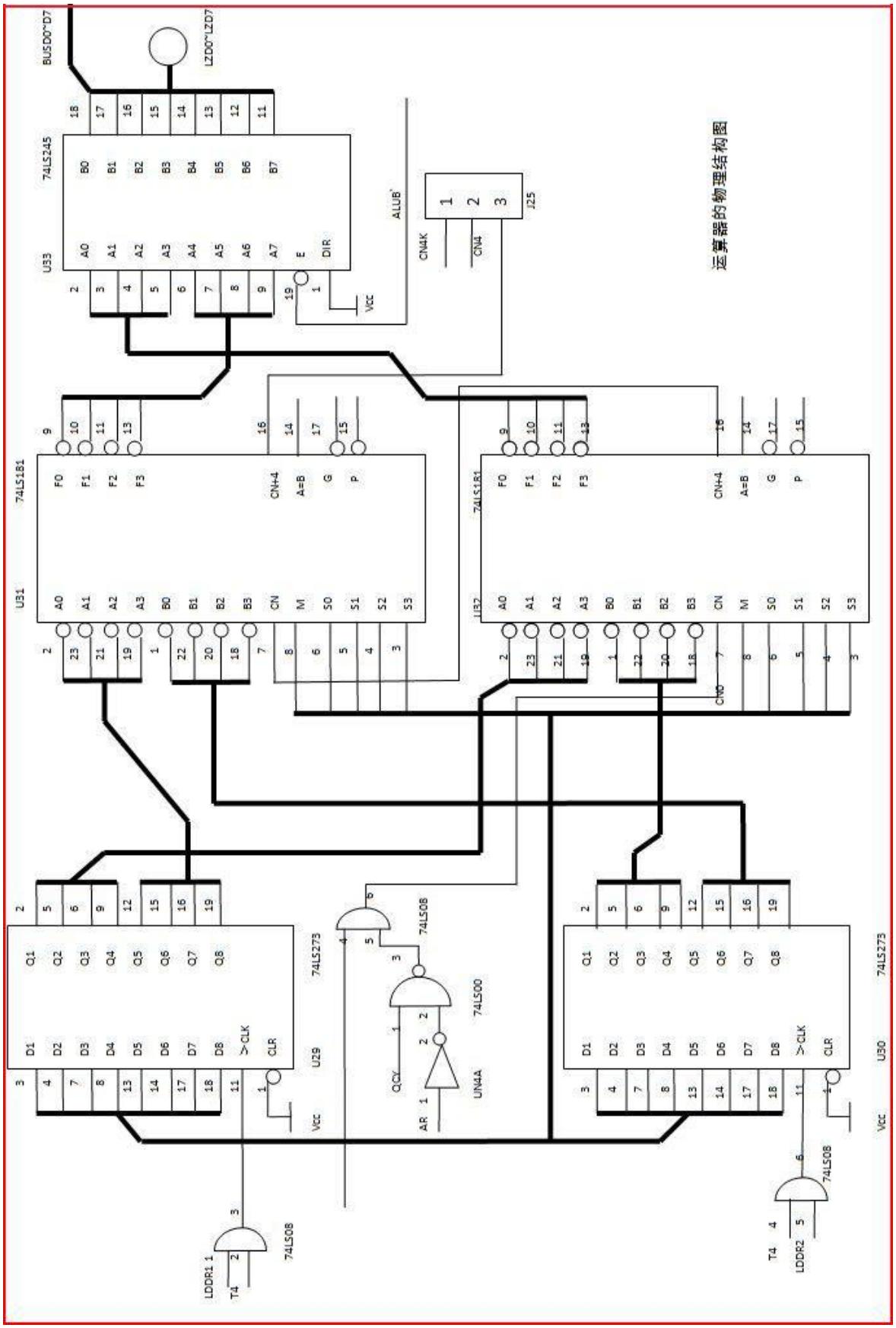

运算器的物理结构图
BUSD0~D7
LZD0~LZD7
74LS245
U33
74LS181
U31
U32
74LS273
U29
U30
74LS08
74LS00
UN4A
J25
CN4K
CN4
ALUB'
Vcc
LDDR1
LDDR2
T4
QCY
AR

图 2.1 运算器物理结构

2.1.2 存储系统

(1)存储器相关知识：组成(主存储器)见图 3.1 存储器原理图，存储体用于存放信息的实体，寻址系统用于对地址码译码，选择存储单元，读/写线路和数据寄存器完成读/写操作，暂存读/写数据，控制线路用于产生读/写时序，控制读/写操作。

(2)CPU 与存储器的连接：地址总线为地址信号，用来指明选中的存储单元地址。数据总线为数据信号，它是微处理器送往存储器的信息或存储器送往微处理器的信息。它包括指令和数据。控制总线发出存储器读写信号，以便从 ROM、RAM 中读出指令或数据，或者向 RAM 写入数据。

(3)集成 RAM 芯片 SRAM6264 芯片介绍：常用的普通集成 RAM 芯片 SRAM6264 的封装图、电路符号及内部结构分别如图所示。图中 $\overline{CS_1}$，$\overline{CS_2}$ 是片选信号，$\overline{WE}$ 是允许与入信号，$\overline{OE}$ 是允许输出(即读出信号)，A_0～A_{12} 是地址输入代码；I/O_0～I/O_7 是 8 位数据输出，V_{DD} 为电源电压，GND 接地，NC 悬空。RAM 的容量用“字数×位数”，6264 的存储容量为“8192 字×8 位”。当存储容量不够时，可以进行字位扩展。

(4)存储容量的扩展：存储器芯片种类繁多、容量不一样。当一片 RAM(或 ROM)不能满足存储容量位数(或字数)要求时，需要多片存储芯片进行扩展，形成一个容量更大、字数位数更多的存储器。扩展方法根据需要有位扩展、字扩展和字位同时扩展 3 种：位扩展，字扩展，字、位同时扩展。

2.1.3 指令系统

模型机规定采用定点补码表示法表示数据，且字长为 8 位，其格式如图 2.1 所示。

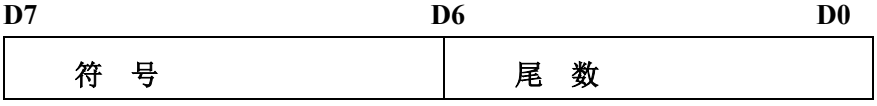


图 2.1 数据格式

其中第 7 位为符号位，数值表示范围是：－1≤X＜1。

模型机设计四大类指令共十六条，其中包括算术逻辑指令、I/O 指令、存数指令、取数指令、转移指令和停机指令。

⑴ 算术逻辑指令　设计 9 条算术逻辑指令并用单字节表示，寻址方式采用寄存器直接寻址，其格式如图 2.2 所示。

图 2.2　算术逻辑指令

其中，OP—CODE 为操作码，RS 为源寄存器，RD 为目的寄存器：

⑵ 访问指令及转移指令　模型机设计 2 条访问指令：即存数 STA、取数 LDA；2 条转移指令：即无条件转移 JMP、有进位转移指令 BZC。指令格式如图 2.3 所示。

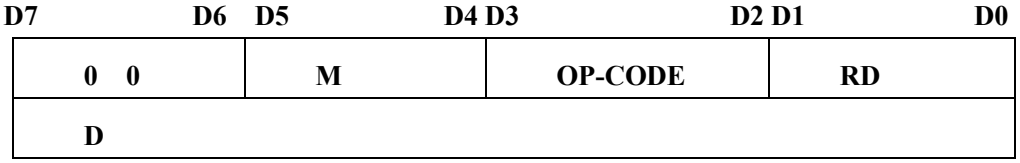


图 2.3　访问及转移指令格式

表 2.1 寄存器编号

RS　或　RD	选定的寄存器
00	R0
01	R1
10	R2

其中，OP—CODE 为操作码，RD 为目的寄存器地址（LDA、STA 指令使用）。D 为位移量（正负均可），M 为寻址模式，其定义如表 2.2 所示。

表 2.2　寻址模式

寻址模式	有效地址	说明
00	E=D	直接寻址
01	E=(D)	间接寻址
10	E=(RI)+D	RI 变址寻址
11	E=(PC)+D	相对寻址

本模型机规定变址 RI 指定为寄存器 R2。

⑶ I / O 指令　输入 IN 和输出 OUT 指令采用单字节指令，其格式如图 2.4 所示。

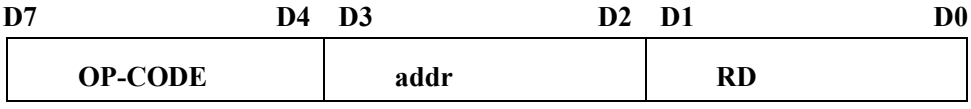


图 2.4　输入输出指令格式

其中，addr=01 时，选中输入数据开关组 KD0～KD7 作为输入设备，addr=10 时，选中

2 位数码管作为输出设备。

2.1.4 微程序控制器

1）微程序控制的基本思想

就是仿照通常解题程序的方法，把操作控制信号编制成所谓的“微指令”，存放到只读存储器里。当机器运行时一条有一条的读出这些微指令，从而产生全机所需要的各种操作控制信号，使相应部件执行所规定的操作。

2）基本组成（见图 2.5） 控制存储器，微指令寄存器，微地址寄存器，地址转移逻辑

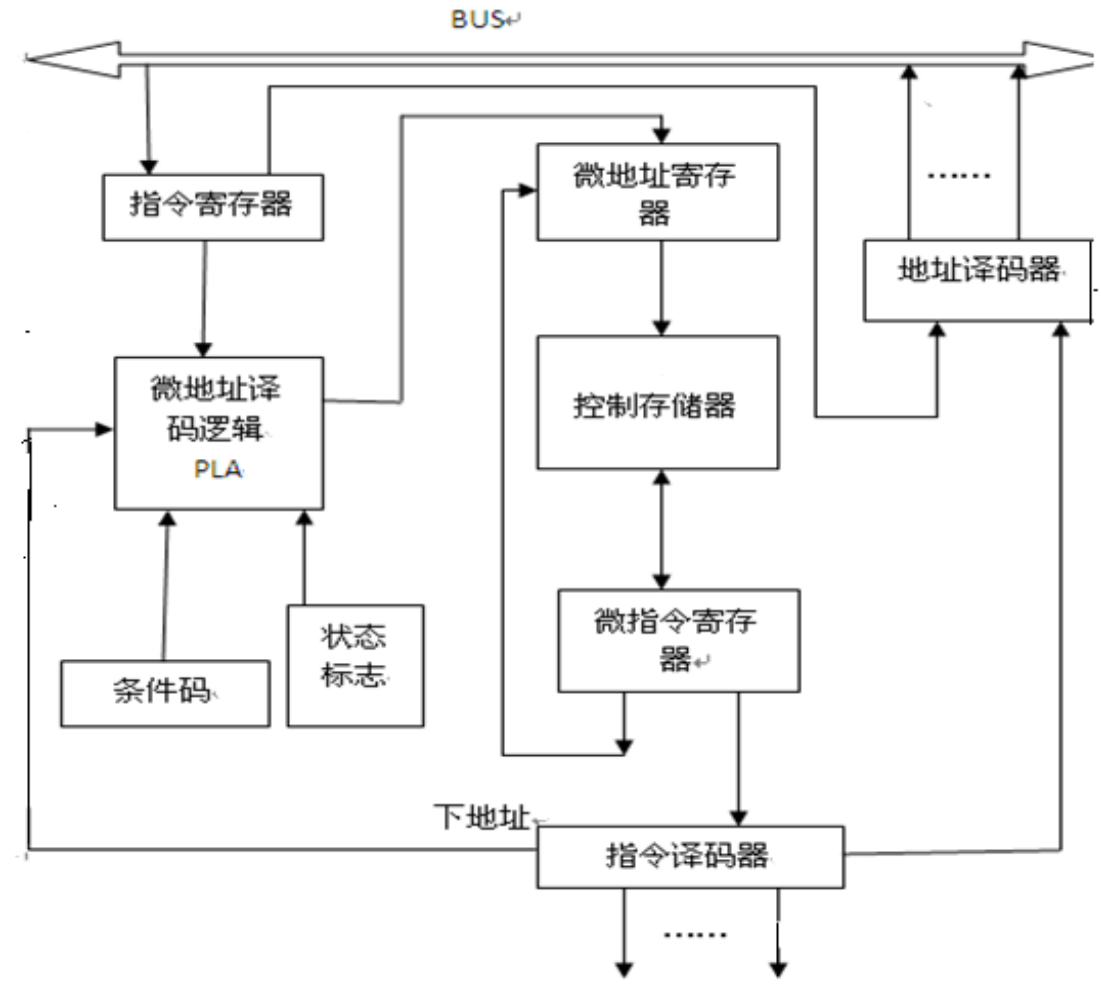


图 2.5 微程序控制器原理框图

a.控制存储器(CM)

控制存储器用来存放实现全部指令系统的微程序，它是一种只读存储器。一旦微程序固化，机器运行时则只读不写。其工作过程是：每读出一条微指令，则执行这条微指令；接着又读出下一条微指令，又执行这一条微指令……。读出一条微指令并执 行微指令的时间总

和称为一个微指令周期。通常，在串行方式的微程序控制器中，微指令周期就是只读存储器的工作周期。控制存储器的字长就是微指令字的长度，其存储容量视机器指令系统而定，即取决于微程序的数量。对控制存储器的要求是速度快，读出周期要短。微指令寄存器：存放当前由控制存储器读出的一条微指令信息，分为微地址寄存器和微命令寄存器两个部分。其中微地址寄存器决定将要访问的下一条微指令的地址，微命令寄存器则保存一条微指令的操作控制字段和判别测试字段(P)的信息

b.微指令寄存器

存放当前由控制存储器读出的一条微指令信息，分为微地址寄存器和微命令寄存器两个部分。其中微地址寄存器决定将要访问的下一条微指令的地址，微命令寄存器则保存一条微指令的操作控制字段和判别测试字段(P)的信息.

c.地址转移逻辑

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

3） 微程序控制器的工作过程： 开始运行程序时

① CPU 自动将取指令的微程序入口地址送入 uAR ，启动控制存储器进行读操作，将微指令送入 uIR 。

② 指令的操作码部分经译码器产生一组微命令，送到有关部件控制完成一组微操作。

③ 由微地址产生逻辑或微指令的下字址给出下一条微指令的地址。再按取微指令，执行微指令的过程重复。

2.1.5 输入输出模块

输入系统模块中用 8 个拨动开关作为输入设备，通过总线驱动器 74LS245（U51）输出到系统的扩展数据总线 EXD0～EXD7 上，输入的数据显示在 LD0～LD7 八个 LED 上，高电平亮，低电平灭。输出设备单元设置两个七段数码管，用于显示需要输出的数据。七段数码管的译码电路由两片 GAL16V8（U53、U54）组成。

2.2 模型机的数据通路

本次课设所能实现的功能 将由微程序控制自动产生各部件单元控制信号，实现特定

指令的功能。这里，实验计算机数据通路的控制将由微程序控制器来完成，CPU从内存中取出一条机器指令到指令执行结束的一个指令周期全部由微指令组成的序列来完成，即一条机器指令对应一个微程序。

实验系统的数据通路图，如图 2.6 所示。

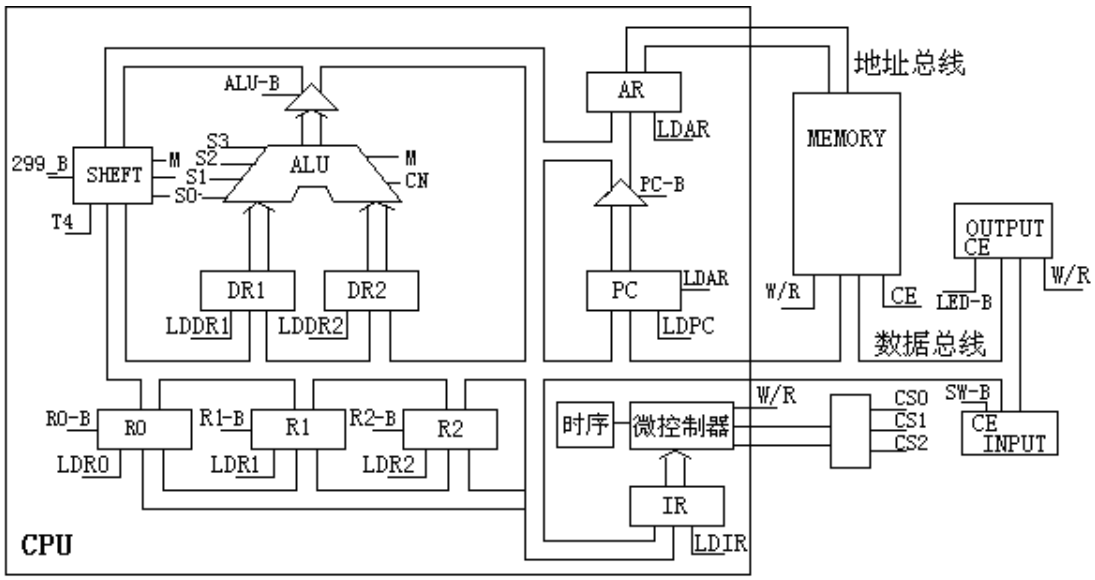


图 2.6 复杂模型机数据通路框图

注意：

① 片选信号 CE=0 为有效电平，CE=1 为无效电平。

② WE=1 为写入，WE=0 为读出。

③ LOAR 和 LDPC 同时为“1”时，可将总线上的数据装入到 PC 中，LDPC 为“1”，同时 LOAR 为“0”时，将 PC 中的内容加 1。

④ M=0 为算术运算，M=1 为逻辑运算。

⑤ CN=0 表示运算开始时低位有进位，否则低位无进位。

图 2.6 中包括运算器、存储器、微控器、输入设备、输出设备以及寄存器。这些部件的动作控制信号都有微控器根据微指令产生。需要特别说明的是由机器指令构成的程序存放在存储器中，而每条机器指令对应的微程序存储在微控器中的存储器中。

第3章　模型机详细设计

3.1　存储器原理

(1)存储器相关知识

功能：存储信息。

组成(主存储器)：见图 3.1

存储体：存放信息的实体。

寻址系统：对地址码译码，选择存储单元。

读/写线路和数据寄存器：完成读/写操作，暂存读/写数据。

控制线路：产生读/写时序，控制读/写操作。

(2)存储器原理

存储器原理如图 3.1 所示。

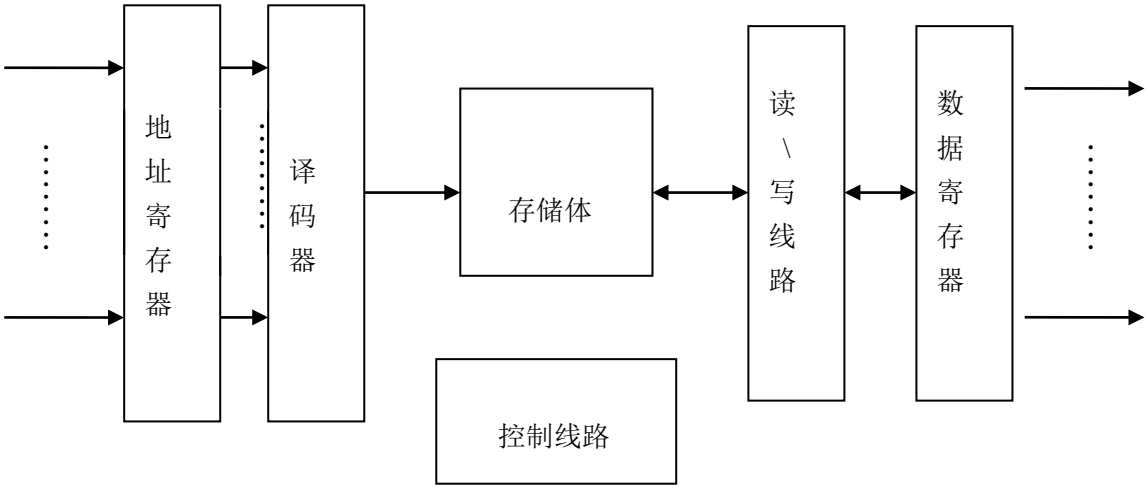


图 3.1　存储器原理图

(3)CPU 与存储器的连接

其原理图如图 3.2 所示。

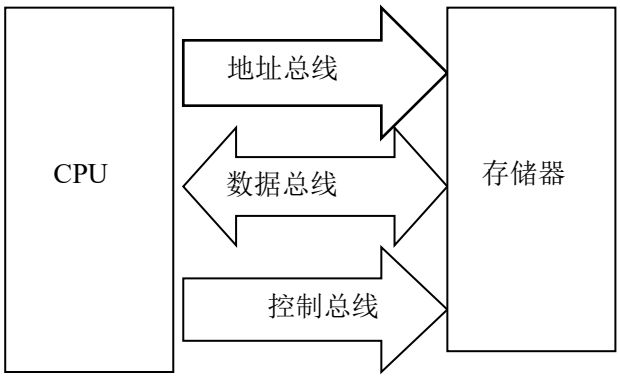


图 3.2　CPU 与存储器的连接原理图

地址总线为地址信号，用来指明选中的存储单元地址。

数据总线为数据信号，它是微处理器送往存储器的信息或存储器送往微处理器的信息。它包括指令和数据。

控制总线发出存储器读写信号，以便从 ROM、RAM 中读出指令或数据，或者向 RAM 写入数据。

①把输入信息存储到由地址信号和控制信号指定的存储单元中。

②根据控制信号的读出要求，把存储在指定存储单元中的数据读出来。

(4)集成 RAM 芯片 SRAM6264 芯片介绍

常用的普通集成 RAM 芯片 SRAM6264 的封装图、电路符号及内部结构分别如图所示。图中 $\overline{CS_1}$，$\overline{CS_2}$ 是片选信号，$\overline{WE}$ 是允许与入信号，$\overline{OE}$ 是允许输出(即读出信号)，A_0～A_{12} 是地址输入代码；I/O_0～I/O_7 是 8 位数据输出，V_{DD} 为电源电压，GND 接地，NC 悬空。

RAM 的容量用“字数×位数”，6264 的存储容量为“8192 字×8 位”。当存储容量不够时，可以进行字位扩展。

表3.1　存储器组成

工作方式	I/O		输入		
	DI	DO	$\overline{OE}$	$\overline{WE}$	$\overline{CS}$
非选择	X	HIGH-Z	X	X	H
读出	HIGH-Z	DO	L	H	L
写入	DI	HIGH-Z	H	L	L
写入	DI	HIGH-Z	L	L	L
选择	X	HIGH-Z	H	L	L

3.2 存储容量的扩展

存储器芯片种类繁多、容量不一样。当一片 RAM(或 ROM)不能满足存储容量位数(或字数)要求时，需要多片存储芯片进行扩展，形成一个容量更大、字数位数更多的存储器。扩展方法根据需要有位扩展、字扩展和字位同时扩展 3 种。

（1）位扩展

若一个存储器的字数用一片集成芯片已经够用，而位数不够用，则用“位扩展”方式将多片该型号集成芯片连接成满足要求的存储器。扩展的方法是将多片同型号的存储器芯片的地址线、读/写控制线($R/\overline{W}$)和片选信号$\overline{CS}$相应连在一起，而将其数据线分别引出接到存储器的数据总线上。

（2）字扩展

若每一片存储器的数据位数够而字线数不够时，则需要采用“字线扩展”的方式将多片该种集成芯片连接成满足要求的存储器。扩展的方法是将各个芯片的数据线、地址线和读写($R/\overline{W}$)控制线分别接在一起，而将片选信号线($\overline{CS}$)单独连接。

（3）字、位同时扩展

在很多情况下，要组成的存储器比现有的存储芯片的字数、位数都多，需要字位同时进行扩展。扩展时可以先计算出所需芯片的总数及片内地址线、数据线的条数，再用前面介绍的方法进行扩展，先进行位扩展，再进行字扩展。

3.3 电路设计

用 8K×8 位的 RAM6264 集成芯片若干片，构成一个 32k×16 位的 RAM 需要 RAM6264 的片数=32k×16 位/(8k×8 位)=8（片）因为芯片 6264 的容量 8192×8 位，表明片内字数 8192=213，所以地址线有 13 条，即(A0～A12)，每字 8 位，数据线有 8 条(D0～D7)。而存储容量为 32K×16 位的 RAM，即字数 32K=215，所以地址线有 15 条，即(A0～A14)，每字 16 位，数据线有 16 条(D0～D15)。

32k×16 位的 SRAM 的逻辑图具体连接方法如图 3.3 所示。

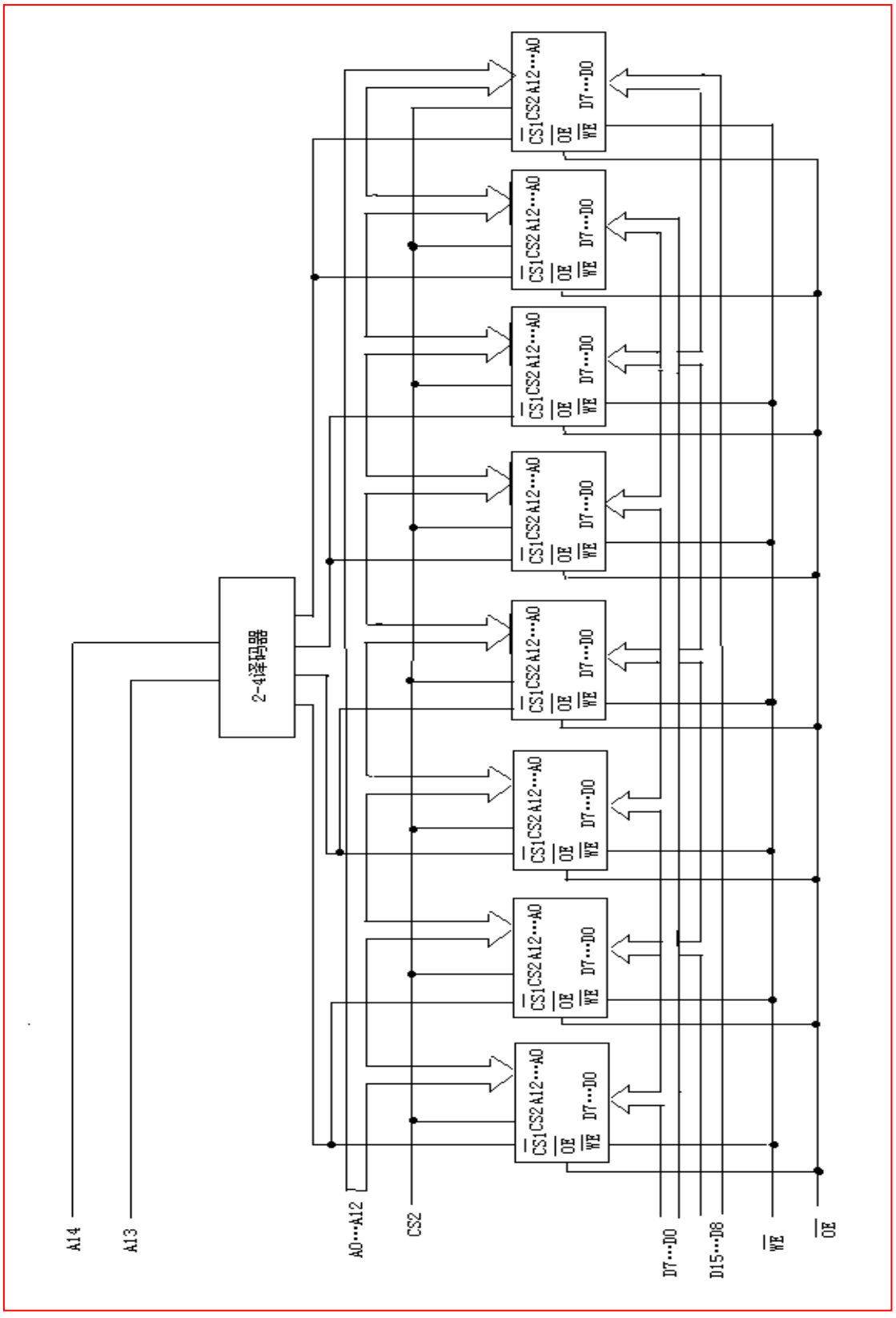
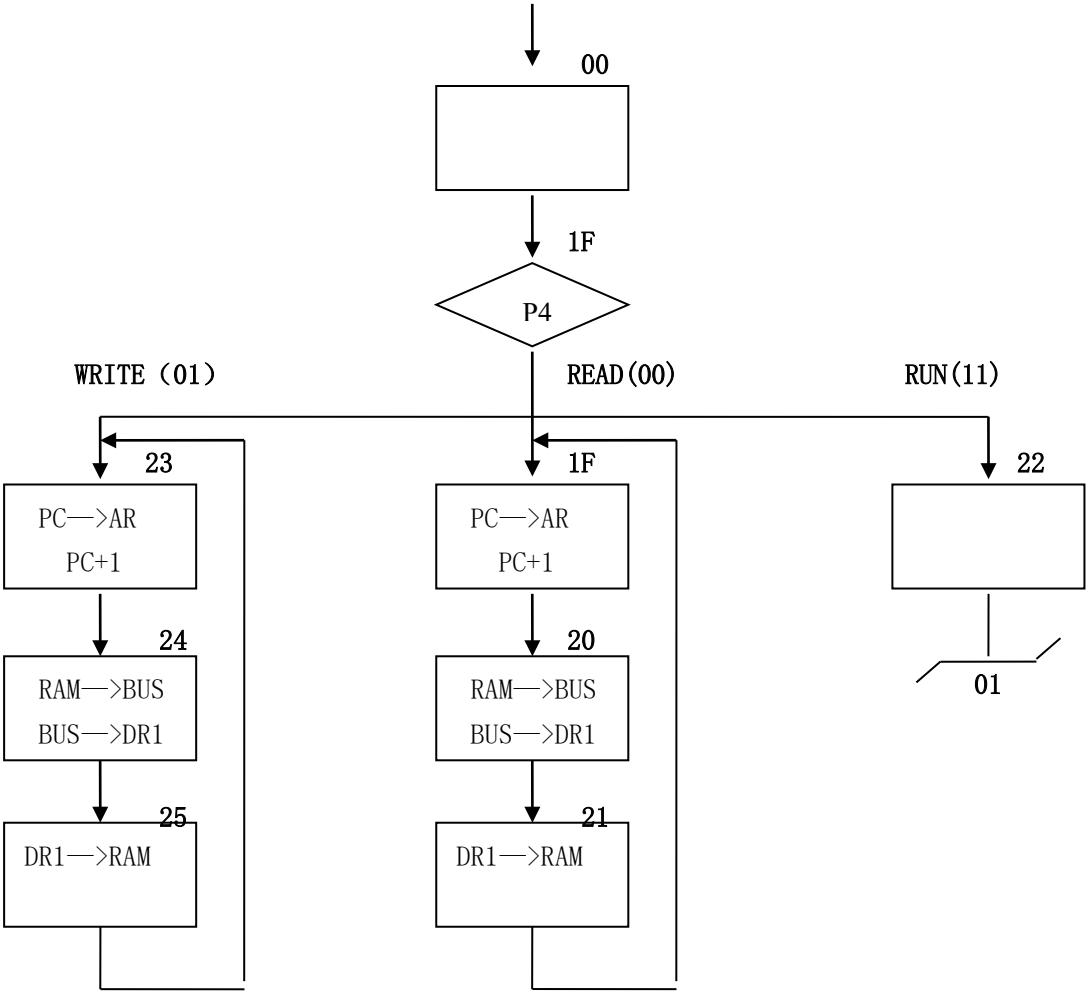


图 3.3　32k×16 位的 SRAM 的逻辑图

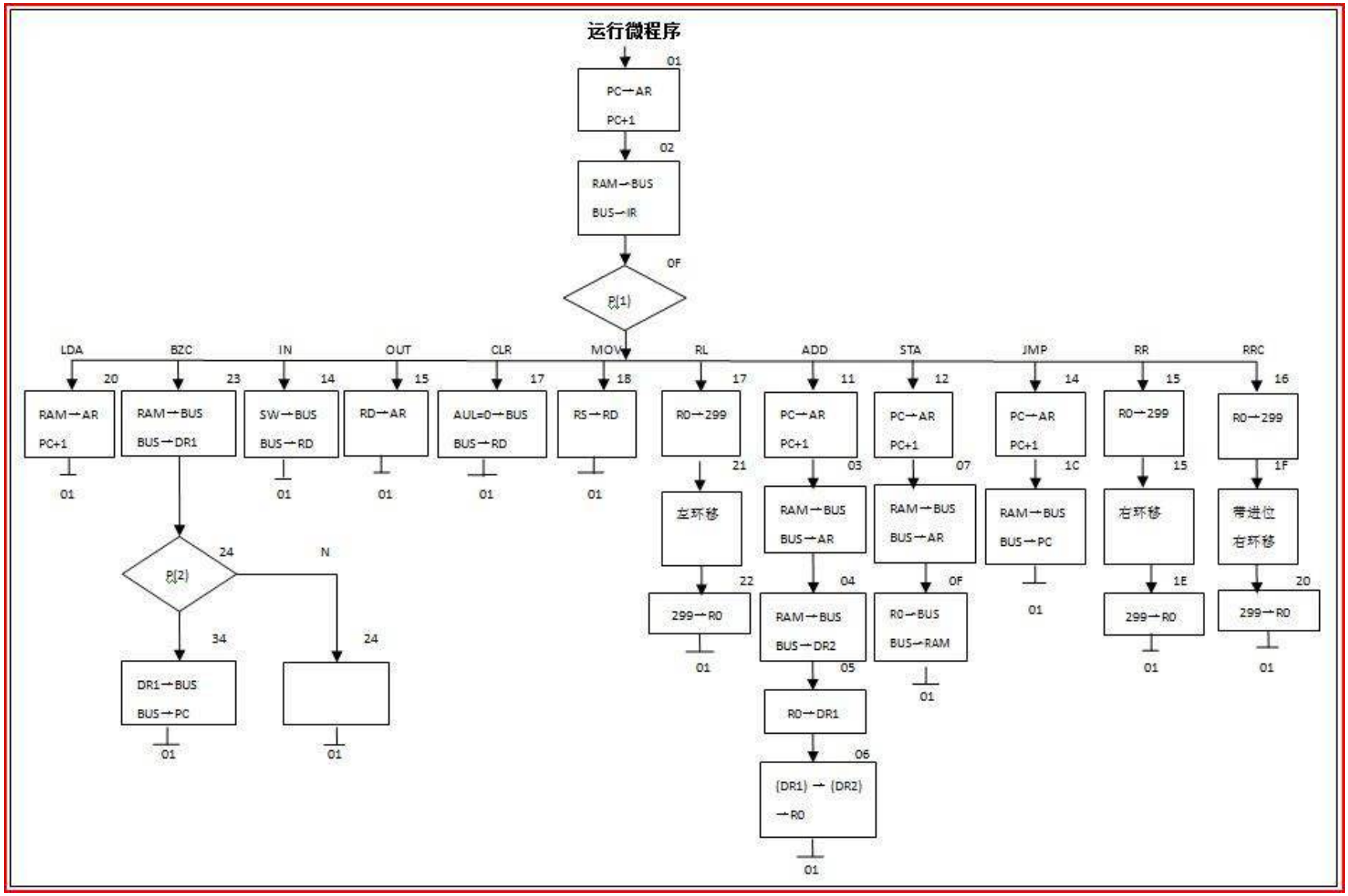
第4章　微程序的设计与实现

4.1　微程序设计流程

微程序流程图见下：



(a) 控制台流程图



(b) 微程序流程图

图 4.1 流程图

4.2 微指令格式设计

微指令长共 24 位，其控制位顺序如表 4.1 所示。

表 4.1 微指令格式表

微程序	24	23	22	21	20	19	18	17	16	15	14	13
控制信号	3	2	1	0	M	CN	WE	B1	B0	A		
微程序	12	11	10	9	8	7	6	5	4	3	2	1
控制信号	B			C			uA5	uA4	uA3	uA2	uA1	uA0

A 字段

15	14	13	选择
0	0	0	
0	0	1	LDRi
0	1	0	LDDR1
0	1	1	LDDR2
1	0	0	LDIR
1	0	1	LOAD
1	1	0	LDAR

B 字段

12	11	10	选择
0	0	0	
0	0	1	RS-B
0	1	0	RD-B
0	1	1	RI-B
1	0	0	299-B
1	0	1	ALU-B
1	1	0	PC-B

C 字段

9	8	7	选择
0	0	0	
0	0	1	P(1)
0	1	0	P(2)
0	1	1	P(3)
1	0	0	P(4)
1	0	1	AR
1	1	0	LDPC

对表 4.1 解释：

S3 S2 S1 S0 M：微运算器 74LS181 芯片的控制信号，详见表 74LS181 功能表。

WE：为 W\R 信号对 RAM 和 OUT 进行写操作，高电平为写有效。

B1，B0：为对外部设备（RAM， OUTPUT， INPUT）地址进行译码，B0B1=00 时，INPUT 选中；B0B1 =01 时，RAM（CE）选中；B0B1=10 的，OUTPUT 选中; B0B1=11 时，外部设备不选中。

A 字段：

LDRi：寄存器输入选中，具体选择同指令寄存器（IR）的最低 2 位（I1，I0）配合，当 I1，I0=00 时为输入到 R0 寄存器；I1，I0=01 时为 R1；I1，I0=10 时为 R2。

LDDR1：暂存器 DR1 选中。

LDDR2：暂存器 DR2 选中。

LDIR：指令寄存器 IR 选中。

LOAD：总线数据直接装载到 PC 计数器。

LDAR：地址寄存器 AR 选。

B 字段：

RS-B：为源寄存器输出选中。具体选择同指令寄存器（IR）的 3，4 位（I3，I2）配合，

当 I3，I2=00 时为输入到 R0 寄存器；I3，I2=01 时为 R1；I3，I2=10 时为 R2。

RD-B：为目的寄存器输出选中。具体选择同指令寄存器（IR）的最低 2 位（I1，I0）配合，当 I1，I0=00 时为输入到 R0 寄存器；I1，I0=01 时为 R1；I1，I0=10 时为 R2。

RI-B：为变址寄存器选中。本机定固定为 R2 。

299-B：移位寄存器输出选中。

ALU-B：逻辑运算单元结果输出。

PC-B ：PC 计数器输出。

C 字段：

P（1）：分支判断 1，和指令寄存器（IR）的高四位（IR7-IR4）作为测试条件。可分 16 个分支。

P（2）：分支判断 2，和指令寄存器（IR）的三四位（IR3，IR2）作为测试条件，有 4 个分支。

P（3）：分支判断 3，和 CY 或 ZI 作为测试条件，有两个分支。

P（4）：分支判断 4，和开关 SWB，SBA 作为测试条件，有 4 个分支。用于控制台控制区 （读程序，写程序，和运行程序）。

AR：进行算术运算时是否影响进位和判零标志的控制位。 选中时进行带进位运算。

LDPC：为 PC 计数信号选中。

UA5……UA0：为下一步微地址。

指令的后续地址的产生方法是：在没有跳转的指令中后六位就是下一条微指令的入口地址。在有跳转的指令根据跳转的条件微控制器根据相应的条件和地址将下地址直接送到为控制器的地址强制端得到下一条指令的地址。

微程序是按顺序在在为控存中存放在系统初始化的是时候指令是从 00H 地址开始的 00H 地址中存放的是一条跳转指令直接可以跳转到 01H 的中存放的就是真正在控制程序功能的指令。机器就根据指令一条的执行。在微控制器的控制下让机器根据指令的来进行有条不紊的工作。

为指令的入口地址的形成是根据机器指令的高四位进行判断后得出的。每一条微指令都对应相应的一个地址。地址的编制和每一微指令是一一对应。不存在冲突。

4.3 二进制微代码表设计

微指令的格式表 4.2 所示。

表 4.2 微指令格式表

微地址	S3 S2 S1 S0	M CN WE R1 R0	A	B	C	UA5…UA0
00	0 0 0 0	0 1 0 1 1	0 0 0	0 0 0	1 0 0	0 0 1 0 0 0
01	0 0 0 0	0 1 0 1 1	1 1 0	1 1 0	1 1 0	0 0 0 0 1 0
02	0 0 0 0	0 1 0 0 1	1 0 0	0 0 0	0 0 1	0 1 0 0 0 0
03	0 0 0 0	0 1 0 0 1	0 1 0	0 0 0	0 0 0	0 0 0 1 0 0
04	0 0 0 0	0 1 0 0 1	1 1 0	0 0 0	0 1 0	1 0 0 0 0 0
05	0 0 0 0	0 1 0 0 1	1 1 0	0 0 0	0 0 0	0 0 0 1 1 0
06	0 0 0 0	0 1 0 0 1	0 1 0	0 0 0	0 0 0	0 0 0 1 1 1
07	0 0 0 0	0 1 0 0 1	1 1 0	0 0 0	0 1 0	1 0 0 0 0 0
08	0 0 0 0	0 1 0 1 1	1 1 0	1 1 0	1 1 0	0 0 1 0 1 0
09	0 0 0 0	0 1 0 1 1	1 1 0	1 1 0	1 1 0	0 0 1 1 0 0
0A	0 0 0 0	0 1 0 0 1	0 1 0	0 0 0	0 0 0	0 1 1 1 0 1
0B	0 0 0 0	0 1 0 1 1	0 0 0	0 0 0	0 0 0	0 0 0 0 0 1
0C	0 0 0 0	0 1 0 0 0	0 1 0	0 0 0	0 0 0	0 1 1 1 1 0
0D	0 0 0 0	0 1 0 0 1	0 1 0	0 0 0	0 0 0	0 0 1 1 1 0
0E	0 0 0 0	0 1 0 1 1	0 1 1	0 1 1	0 0 0	0 0 1 1 1 1
0F	1 0 0 1	0 1 0 1 1	1 1 0	1 0 1	0 0 0	1 0 0 1 0 1
10	0 0 0 0	0 1 0 1 1	1 1 0	1 1 0	1 1 0	0 0 0 0 1 1
11	0 0 0 0	0 1 0 1 1	1 1 0	1 1 0	1 1 0	0 0 0 1 0 1
12	0 0 0 0	0 1 0 1 1	1 1 0	1 1 0	1 1 0	0 0 1 1 0 1
13	0 0 0 0	0 1 0 1 1	1 1 0	1 1 0	1 1 0	1 0 0 1 1 0
14	0 0 0 0	0 1 0 0 0	0 0 1	0 0 0	0 0 0	0 0 0 0 0 1
15	0 0 0 0	0 1 1 1 0	0 0 0	0 1 0	0 0 0	0 0 0 0 0 1
16	0 0 1 1	0 1 0 1 1	0 0 0	0 0 1	0 0 0	0 1 1 0 1 1
17	0 0 1 1	1 1 0 1 1	0 0 1	1 0 1	0 0 0	0 0 0 0 0 1
18	0 0 0 0	0 1 0 1 1	0 0 1	0 0 1	0 0 0	0 0 0 0 0 1
19	0 0 0 0	0 1 0 1 1	0 1 0	0 0 1	0 0 0	1 0 1 0 1 0
1A	0 0 0 0	0 1 0 1 1	0 1 1	0 0 1	0 0 0	1 0 1 1 0 0
1B	0 0 0 1	1 1 0 1 1	0 0 0	1 0 0	0 0 0	0 1 1 1 0 0
1C	0 0 0 0	0 1 0 1 1	0 0 1	1 0 0	0 0 0	0 0 0 0 0 1
1D	0 0 0 0	0 1 1 1 0	0 0 0	1 0 1	0 0 0	0 0 1 0 0 0
1E	0 0 0 0	0 1 1 0 1	0 0 0	1 0 1	0 0 0	0 0 1 0 0 1
1F	0 0 0 0	0 1 0 1 1	1 0 1	1 0 1	1 1 0	0 0 0 0 0 1
20	0 0 0 0	0 1 0 0 1	0 0 1	0 0 0	0 0 0	0 0 0 0 0 1
21	0 0 0 0	0 1 1 0 1	0 0 0	0 1 0	0 0 0	0 0 0 0 0 1
22	0 0 0 0	0 1 0 1 1	1 0 1	1 0 1	1 1 0	0 0 0 0 0 1
23	0 0 0 0	0 1 0 1 1	0 0 0	0 0 0	0 1 1	1 0 0 1 0 0
24	0 0 0 0	0 1 0 1 1	0 0 0	0 0 0	0 0 0	0 0 0 0 0 1
25	1 0 0 1	0 1 0 1 1	0 1 0	1 0 1	0 1 0	1 0 0 0 0 0
26	0 0 0 0	0 1 0 0 1	0 1 0	0 0 0	0 0 0	1 0 0 1 1 1
27	0 0 0 0	0 1 0 1 1	0 1 1	1 1 0	0 0 0	1 0 1 0 0 0

28	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**
29	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
2A	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**
2B	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
2C	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**
2D	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**0**
2E	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**
2F	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
30	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**1**
31	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**

第5章　系统调试及运行报告

5.1　调试环境搭建

5.1.1 DVCC 试验箱的连线

a、跳线器 J1～J12 全部拨在右边（自动工作方式）；

b、跳线器 J16、J18、J23、J24 全部拨在左边；

c、跳线器 J15、J19、J25 全部拨在右边，跳线器 J13、J14 拨在左边；

d、跳线器 J20～J22、J26、J27 连上短路片；

e、UJ1 连 UJ2，JSE1 连 JSE2，SJ1 连 SJ2；

f、MBUS 连 BUS2；

g、REGBUS 连 BUS5；

h、PCBUS 连 EXJ2；

i、ALUBUS 连 EXJ3；

j、ALUO1 连 BUS1；

k、EXJ1 连 BUS3；

l、IJ1 连 IJ2。

5.1.2 联机读写

用 DVCC 联机软件的装载功能将 16 进制格式文件（文件名为 test）装入实验机即可。

5.2　运行程序

（1） 单步运行程序

A.“编程开关”置“运行”状态，“运行方式”开关置为“单步”状态，“运行控制”开关置为“运行”状态。

B. 拨动总清开关(0→1)，微地址清零，PC 计数器清零，程序首地址为 OOH。

C. 按动“启动运行”开关，即单步运行一条微指令。对照微程序流程图，观察微地址显示灯是否和流程一致。

（2） 连续运行程序

A.“编程开关”置“运行”状态，“运行方式”开关置为“连续”状态，“运行控制”开关置为“运行”状态。

B. 拨动总清开关，清微地址及 **PC** 计数器，按动“启动运行”开关，系统连续运行程序。如果要停止程序的运行，只需将“运行控制”开关置为“停止”状态，系统就停机。

5.3 解决调试中的问题

（1）微程序准确录入，校验。

（2）排线的线路出问题，更换排线后得以解决。

（3）准确记录程序执行过程。

5.4 指令执行过程

通过上机实验，记录指令执行过程如下，这里我们设置输入值为 **10**。

INPUT(10)→BUS→R0(10)

PC(01)→AR(00)→RAM(44)

RAM(44)→IR(44)→微控制器

PC(02)→AR(01)→RAM(46)

RAM(46)→IR(46)→微控制器

INPUT(10)→BUS→R2(10)

PC(03)→AR(02)→RAM(98)

RAM(98)→IR(98)→微控制器

R2(10)→DR1(10)→ALU(10)

R0(10)→DR2(10)→ALU(20)

ALU(20)→R0(20)

PC(04)→AR(03)→RAM(81)

RAM(81)→IR(81)→微控制器

R0(20)→R1(20)

PC(05)→AR(04)→RAM(F5)

RAM(F5)→IR(F5)→微控制器

R1(20)→299(20)→R1(00)

PC(06)→AR(05)→RAM(0C)

RAM(0C)→IR(0C)→微控制器

PC(07)→AR(06)→RAM(00)

RAM(00)→DR1(00)

RAM(00)→AR(00)→RAM(00)

设计总结

这是一个复杂模型机设计与实现的课程设计。总体上是要加深对计算机各部件的组成和工作原理的理解，掌握微程序计算机中指令和微指令的编码方法，深入理解机器指令在计算机中的运行过程。

在两周的实验中，我设计了一个复杂的模型机，该模型机包含若干条简单的计算机指令，其中包括输入、输出指令，存储器读写指令，寄存器访问指令，运算指令，程序控制指令。自行设计出了这些机器指令对应的微指令代码，并将其存放于控制存储器，并利用机器指令设计了一段简单机器指令程序。将实验设备通过串口连接计算机，通过联机软件将机器指令程序和编写的微指令程序存入主存中，并运行此段程序，通过联机软件显示和观察该段程序的运行，验证编写的指令和微指令的执行情况是否符合设计要求，并对程序运行结果的正、误分析了原因。

在实验的开始阶段问题是不少的。首先对机器指令及微指令的的编码方法不了解，对计算机各部件的组成和工作原理也不是很理解，通过大家对实验指导书上的例程的研究学习和相互探讨，逐渐理解了计算机各部件的组成和工作原理，掌握的机器指令和微指令的编码方法，最终正确的实现了课程设计要求的内容。

通过将近两个星期的努力，在老师的指导下和我自己的不懈努力下，我终于成功完成了此次复杂模型机的设计，在此次课程设计中，我受益颇多，从刚刚接触到次此类设计，在这其中可以说其乐无穷啊，这也许是只有亲历亲为，当见到自己的努力有所收获时，那种感受不言而喻，总之，此次课程设计使我学到了好多了以前不知道的有关指令编码的知识，使我对微机有了进一步的认识。

参考文献

[1] 白中英. 计算机组成原理. 北京：科学技术出版社，2006.8

[2] 白中英. 计算机组成原理题解、题库、实验. 北京：科学技术出版社，2006.8

[3] 王爱英. 计算机组成与结构. 北京：清华大学出版社，1999

[4] 王诚. 计算机组成与结构. 北京：清华大学出版社，1999

[5] 唐朔飞. 计算机组成原理. 北京：高等教育出版社，1993

致　谢

首先感谢我的指导老师谢鹏寿老师，他在我的课程设计过程中提出了指导性的方案和架构，并指引我阅读相关的资料和书籍，使我在不熟悉的领域中仍能迅速掌握新的技术。

谢老师的严谨的工作作风、亲切的待人方式，渊博的专业知识都给我留下了深刻的印象，老师细心的讲解和指点使我从课程设计之中受益匪浅，使我从实际的算法实现中更好地掌握了理论知识。

感谢我的《计算机组成原理》老师包仲贤老师以及与课设有关的各任课教师。一个学期以来老师们严肃的教学态度使我从对该课程的一无所知到收获颇丰，也为我今后的求学生涯树立了榜样。你们一直以来默默无闻的扮演着传道、授业、解惑的角色，您就是那将我们送到对岸的摆渡人。还要感谢所有参加评阅设计说明书以及参加答辩的老师，是你们让我及时的发现错误，改正错误，很快的取得了进步。

最后还要感谢我的同学们，感谢他们耐心的讲解和细心的指导，也感谢他们给我提出那么多的解决方案和指导性意见，帮我顺利完成我的课程设计。