****

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：模型机的设计—7

专业班级：

姓　　名：

学　　号：

指导教师：包仲贤

成　　绩：

摘　要

计算机组成原理讲授单处理机系统的组成和工作原理，此课程的知识面非常广、难度大、更新快，内容包括中央处理器、指令系统、微指令、存储系统、总线和输入输出系统等方面，内容抽象难于理解。计算机组成原理是计算机科学与技术系的一门核心专业课程，对学生理解计算机工作原理起到了重要作用。

此次课程设计通过对一个简单模型机的设计以及模型机的调试，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，并使学生对整体计算机的工作原理有一个更加深入的认识与理解。

关键词：　计算机组成原理；指令系统；微指令；模型机；

目　　录

前　言

该设计要求我们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。通过对简单模型机的设计与调试，使我们将掌握的计算机组成基本知识应用于实践中，在实际操作中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。掌握微程序设计思想和设计方法，设计实现一个简单的模型机指令系统。该模型机由运算器、寄存器、译码电路、存储器、和存储微指令用的控制存储器组成，能够实现一些简单的机器指令，并根据设计好的指令系统设计简单的机器指令程序，实现输入、输出、存储器读写和简单的控制指令。

正　文

一. 设计目的和设计原理

1.1 设计目的

融会贯通计算机组成原理课程中各章的内容，通过知识的综合运用，加深对计算机系统各模块的工作原理及相互联系的认识，特别是对微程序控制器的认识，建立清晰的整机概念。对计算机的基本组成、部件的设计、部件间的连接、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。

在掌握计算机整体的工作原理之上，进一步根据计算机组成原理的知识构造一台基本模型计算机。

1.2 设计原理

此基本模型机的设计主要包括运算器、存储器、微程序控制器、指令系统和输入/输出设备五个模块组成。其中运算器主要由运算器（74LS181）、暂存器（74LS273）、输出缓冲器（74LS245）、移位器（74LS299）以及进位控制和判零标志控制电路等构成。存储器存储该主存储器采用一级 cache-存储器结构。控制器它主要由控制存储器、微指令寄存器和地址转移逻辑三大部分组成，其中微指令寄存器分为微地址寄存器和微命令寄存器两部分。一台计算机中所有机器指令的集合，成为这台计算机的指令系统，它是计算机系统设计的一个核心问题，它不仅与计算机的硬件结构紧密相关，而且直接关系到用户的使用需要。输入/输出设备主要有两种外部 I/O 设备，一种是键盘，它作为输入设备 INPUT；另一种是字符显示块，它作为输出设备 OUTPUT。

指令从内存中读出，首先放到 IR 指令寄存器中，再送入到微控器中进行译码，再由微程序控制器输出各种控制信号给各功能部件，执行相应的操作。指令和数据统一放在内存中，从形式上看，它们都是二进制代码，。一般来讲，取值周期从内存中读出的信息流是指令流，它指向控制器；而在执行周期中从内

存中读出的信息流是数据流，它由内存流向运算器。指令从存储器中得到，送到指令寄存器中。由指令得到相应的控制信号，从控制器输出控制信号来控制计算机的运行。

二. 模型机的逻辑结构及框图

2.1 各模块功能

a.运算器

运算器就好像是一个由电子线路构成的算盘，它的主要功能就是进行加、乘、除以及逻辑运算。电子器件的特性，计算机中通常采用二进制数。运算器采用片间串行进位 8 位移位运算器。其运算规律非常简单。例如：加法与乘法运算，0+0=0,0+1=1，1+0=1,1+1=10,0×0=0，0×1=0，1×0=0，1×1=1。

b.存储器

存储器的功能是保存或“记忆”各种数据。在存放到存储器以前，他们全部变成 0 或 1 表示的二进制代码。采用半导体器件来存放大量的 0，1.一个半导体触发器由于有 0 和 1 两种状态，可以记忆一个二进制代码。一个数据假定用 8 位二进制代码来表示，那么就需要 8 个触发器来保存这些代码。通常，在存储器中保存一个数的 8 个触发器称为一个存储单元。存储器是由许多存储单元组成的，每一个存储单元都有编号，称为地址。存储器的所有存储单元的总数称为存储器的存储容量。存储器采用一级 cache-存储器结构。

c.指令系统

指令系统控制计算机系统有条不紊的工作

d.控制器

控制器是计算机中发号施令的部件，它控制计算机的各部件有条不紊的进行工作。更具体地讲，控制器的任务就是从内存中取出指令加以分析，然后执行相应的操作。

e.输入/输出设备

此简单模型机有两种外部 I/O 设备，一种是键盘，它作为输入设备 INPUT；另一种是字符显示块，它作为输出设备 OUTPUT。

f.寄存器介绍

①指令寄存器用来保存当前正在执行的一条指令。当执行一条指令时，先把它从内存取到缓冲寄存器中，然后再传送到指令寄存器。指令划分为操作码

和地址码字段，由二进制构成，为了执行任何一条给定的指令，必须对操作码进行测试P(1),通过节拍脉冲T4的控制以便识别所要求的操作。“指令译码器”根据指令中的操作码进行译码，强置微控器单元的微地址，使下一条微指令指向相应的微程序首地址。

②数据寄存器用来存放从内存中读出的数据或是从输入设备输入的数据，然后将数据送到运算器等功能部件，起了暂存的作用。

③PC 为程序计数寄存器，用来存放下一条将要执行指令的地址。

④SP 为堆栈指针计数寄存器，用来在堆栈寻址方式中指示栈顶指针的位置。

2.2 整体设计

（1）总体设计

此模型机的总体原理图如下

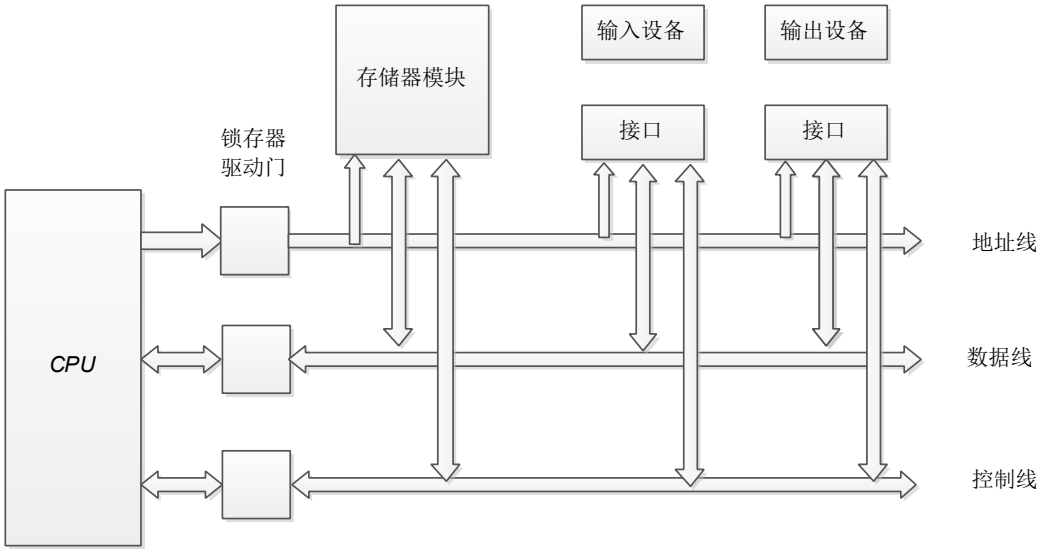

图 2—1 计算机原理图

该模型机使用总线系统来连接 CPU、主存和 I/O 设备，总线分为地址线、数据线和控制线。地址线是单向的，用了传送主存与设备的地址；数据线是双向的，用来传送数据；控制线对每一根线来说是单向的（CPU 发向接口，或接口发向 CPU），用来指明数据传送方向（存储器读写、I/O 读写）中断控制（请求、识别）和定时控制等。

（2）数据通路

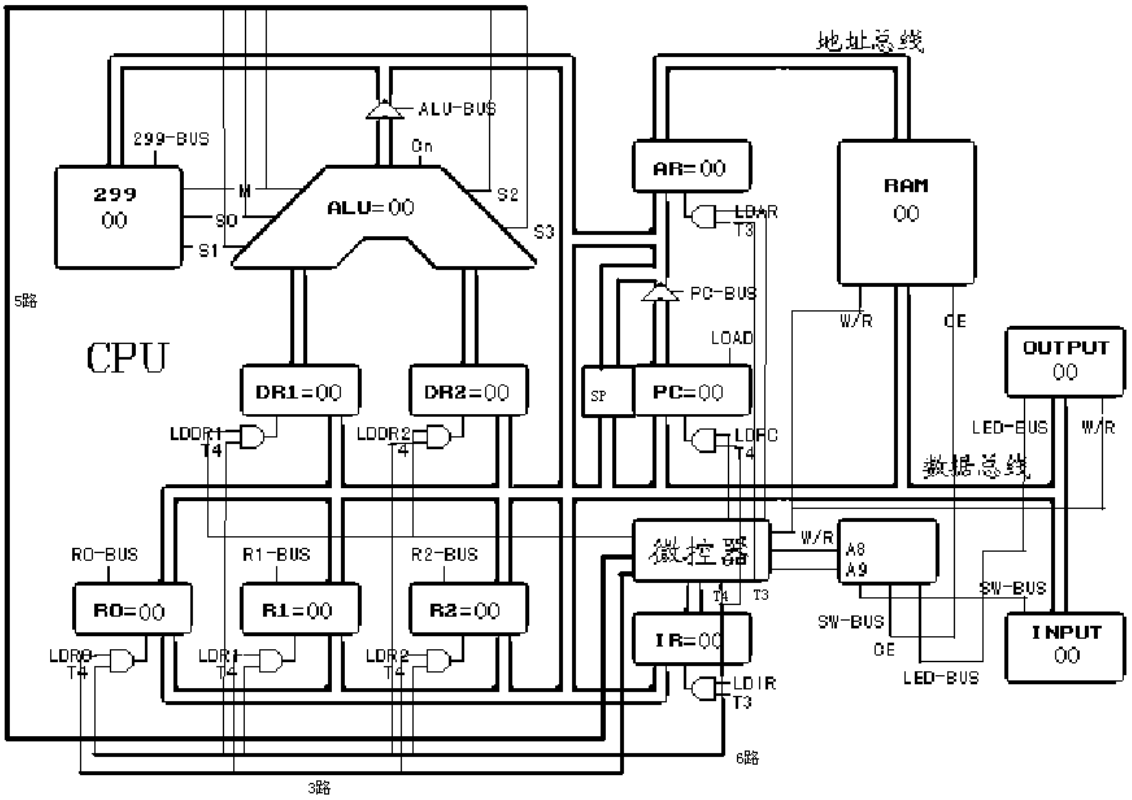

图 2—2 数据通路图

此模型机是由运算器，存储器，控制器，输入设备，输出设备五大部分组成。运算器由算术逻辑单元(ALU)、通用寄存器、数据缓冲寄存器 DR 和状态条件寄存器 PSW 组成，它是数据加工处理部件。相对控制器而言，它是执行部件。运算器有两个主要功能： (1)执行所有的算术运算；(2)执行所有的逻辑运算，并进行逻辑测试，如零值测试或两个值的比较。控制器根据指令操作码和时序信号，产生各种操作控制信号，以便正确地建立数据通路，从而完成取指令和执行指令的控制。存储器作为计算机的记忆部件，用于存放程序和数据。输入设备为键盘，计算机键盘的功能就是及时发现被按下的键，并将该按键的信息送入计算机。输出设备为显示器将输出的信息以字符的形式显示出来。

模型机运行的主要过程为首先将程序计数器 PC 的内容装入地址寄存器 AR；然后程序计数器的内容加 1，为下一条程序做准备；接着地址寄存器的内容放到地址总线上；从而使存储单元的内容传送的缓冲寄存器 DR；然后将缓冲寄存器的内容传送到指令寄存器。到这里完成了取指令。比如现在要做的操作为执行 CLA 指令，操作控制器送一控制信号给 ALU，接着 ALU 响应控制信号对 AC 清零。如果接下来执行 ADD 操作，取指令与上面相同，然后从内存中读取操作数，操作数与累加器相加后存入累加器。

三. 模型机的详细设计

3.1 运算器的物理结构

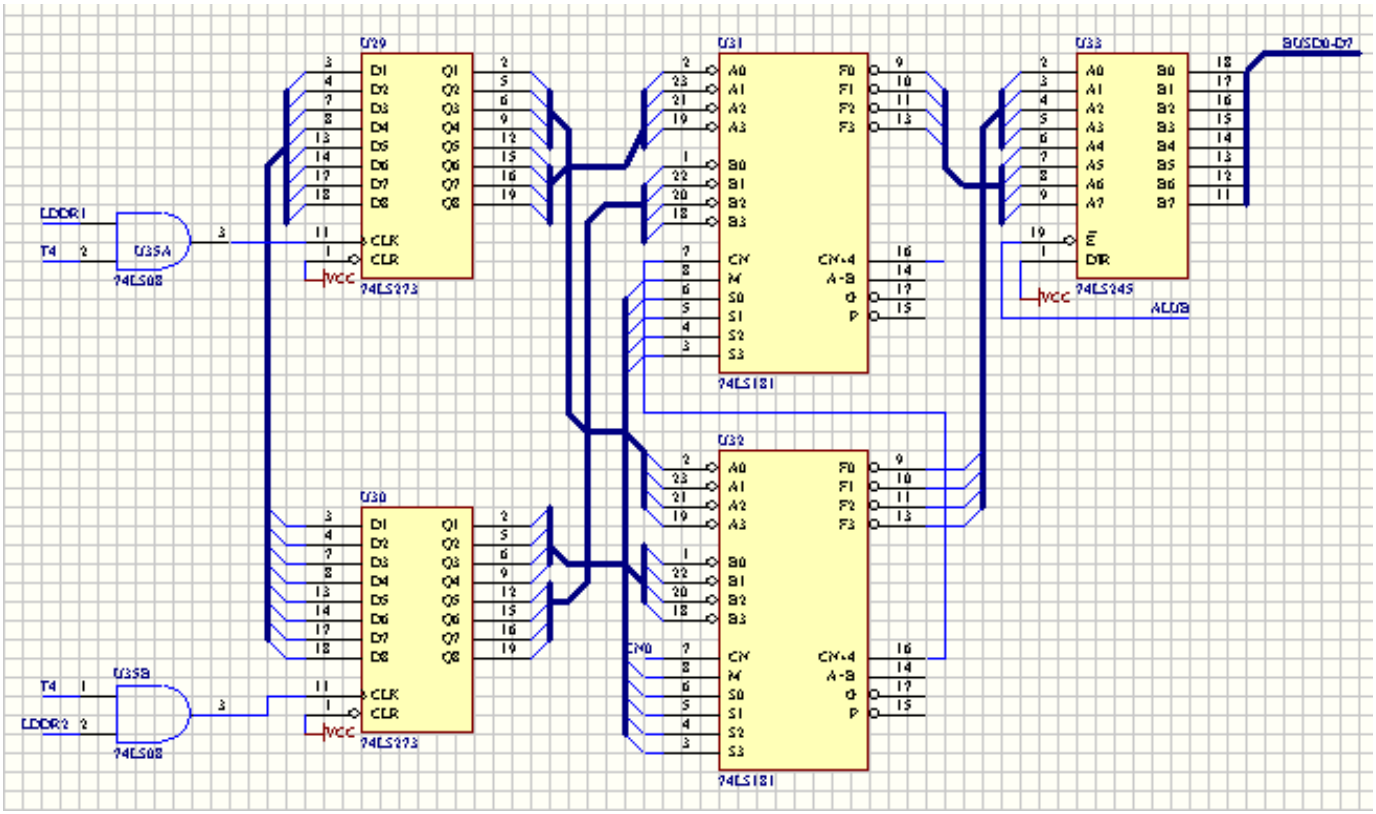

图 3-1　8 位移位运算器设计图

在该运算器中，有两片 74LS181 级联而成，构成 8 位算术和逻辑运算器。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。74LS299 用来完成移位操作，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。运算器采用片间串行进位。串行进位优点是硬件电路简单，但是运算速度较慢。

3.2 存储器系统的组成

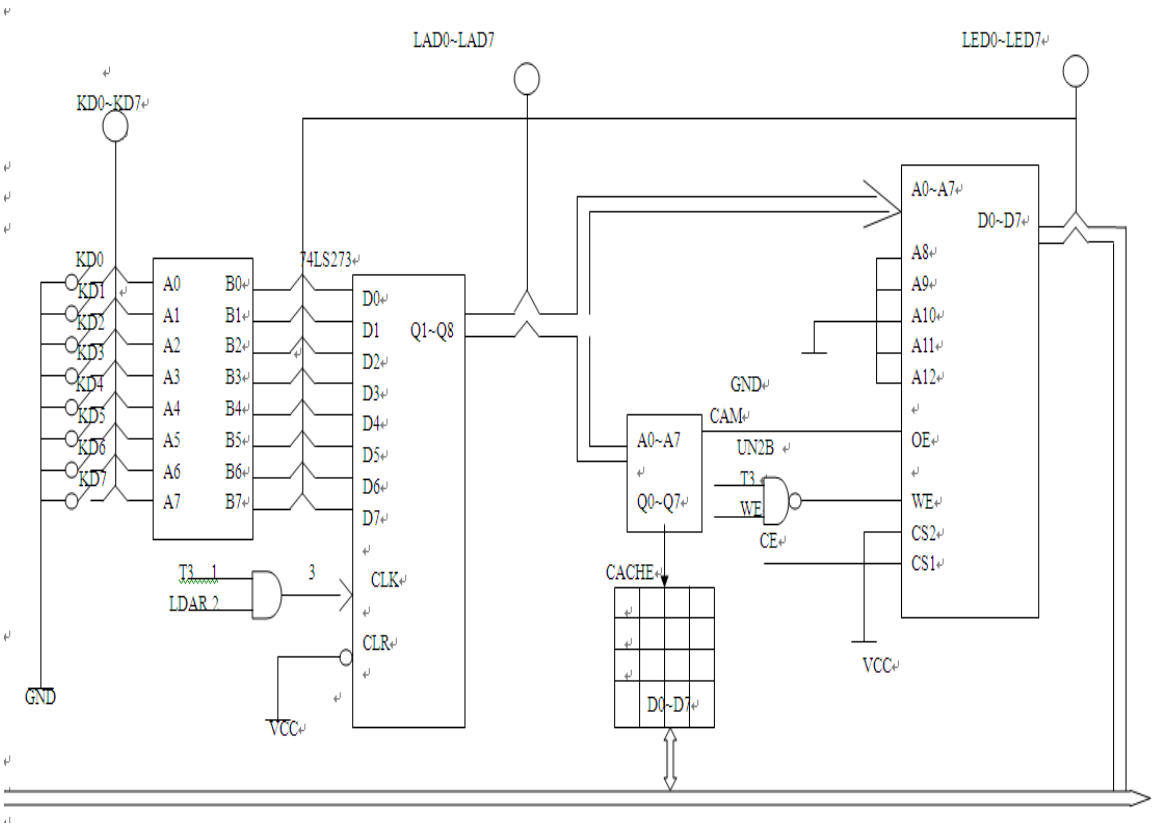

图 3—2 存储器原理图

本模型机主存储器采用一级 cache-存储器结构。一个存储单元存放一个机器字，每一个存储单元有唯一的地址。存储器存放数据与指令，当存储器数据传输到控制器时，传送的是指令；当存储器数据传输到运算器时，传送的是数据。

Cache 是一种高速缓冲存储器，是为了解决 CPU 和主存之间速度不匹配而采用的一项重要技术。主存容量配置几百 MB 的情况下，cache 的典型值是几百 KB。Cache 能高速地向 CPU 提供指令和数据，它是主存的缓冲器，由高速地 SRAM 组成。

CPU 与 cache 之间的数据交换是以字为单位的，而 cache 与主存之间的数据交换是以块为单位。一个块由若干字组成，是定长的。当 CPU 读取主存一个字时，便发出此字的内存地址到 cache 和主存。此时 cache 控制逻辑依据地址判断此字当前是否在 cache 中：若是，此字立即传送给 CPU；若非，则用主存读周期把此字从主存读出送到 CPU，与此同时，把含有这个字的整个数据块从主存读出送到 cache 中。

本模型机中 cache 分为四行，每行 4 个字（W），分配给 cache 的地址存放在一个相联存储器 CAM 中，它是按内容寻址的存储器。为了把主存块放到 cache 中，采用组相联映射方式把主存地址定位到 cache 中。组相联映射是全相联映射与直接映射方式的折中，兼顾两者的优点，存储位置的灵活性和命中率较高，而且硬件电路不复杂。

Cache 的工作原理要求它尽可能保存最新数据。当一个新的主存块需要拷贝到 cache 二而允许存放此块的行位置都被其他主存块占满是，就产生替换。此模型机采用近期最少使用（LRU）算法，cache 将近期内长久未被访问过的行换出。

每行也设置一个计数器，但它们是 cache 没命中一次，命中行计数器清零，其他各行计数器增 1。当需要替换时，比较各特定行的计数值，将计数值最大的行换出，LRU 算法保护了刚拷贝到 cache 中的新数据行，符合 cache 工作原理，因而使 cache 有较高的命中率。

由于 cache 的内容只是主存内容的拷贝，它应当与主存内容保持一致。而 CPU 对 cache 的写入更改了 cache 的内容。为了保持与主存内容一致，写操作策略采用写回法。当 CPU 写 cache 命中时，只修改 cache 的内容，而不立即写入主存；只有当此行被换出时才写回主存。 这种方法是 cache 真正在 CPU-主存之间读/写两方面都起到高速缓冲作用。

3.3 指令系统的设计与指令格式分析

1）数据格式

模型机规定采用定点补码表示法表示数据，且字长为8位，其格式如下：

D7	D6 D5 D4 D3 D2 D1 D0
符号	尾数

其中第7位为符号位，数值表示范围是：$-128 \le X \le 127$ 。

2）指令系统的设计

本模型机共有12条基本指令，其中算术运算类指令3条（ADD、SUB、CLR），

数据传送类指令（MOVE）,访问内存指令2条(STA、LDA),程序控制指令2条（JMP、BZC），I / O指令2条(IN、OUT)，位操作指令2条（OR、XOR）。

此模型机具有12条指令，ADD、SUB、MOVE、CLR、IN、OUT、OR、XOR采用单字节长指令，STA、LDA、JMP、BZC采用双字长指令。12条指令需要操作码四位，用余下的四位进行操作数地址选择。双字长指令有四种寻址方式，采用寻址方式位法决定寻址方式。指令设计如下：

(1) 算术逻辑指令

设计3条算术运算指令（ADD、SUB、CLR），单字长指令，寻址方式采用寄存器直接寻址，格式如下：

D7 D6 D5 D4	D3 D2	D1 D0
OP-CODE	RS	RD

其中，OP—CODE为操作码，RS为源寄存器，RD为目的寄存器，并规定：

RS 或 RD	选定的寄存器
00	R0
01	R1
10	R2

(2) 数据传送指令

设计一条数据传送类指令MOVE，单字长指令，寻址方式采用寄存器直接寻址，格式如下：

D7 D6 D5 D4	D3 D2	D1 D0
OP-CODE	RS	RD

(3) 访问内存指令和程序控制指令

模型机设计访问内存指令2条（存数STA、取数LDA）和程序控制指令2条 （、无条件转移JMP、有进位转移指令BZC）。寻址方式4种，采用寻址方式位法决定寻址方式。

指令格式为：

D7 D6 D5 D4	D3 D2	D1 D0
OP-CODE	M	RD
D		

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占两位，决定四种寻址方式，定义如下：

M	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。

(4) I / O指令

输入IN和输出OUT指令采用单字节指令，其格式如下：

D7 D6 D5 D4	D3 D2	D1 D0
OP-CODE	addr	RD

addr=01时，选中键盘作为输入设备；

addr=10时，选中字符显示器作为输出设备。

(5) 位操作指令

位操作指令有两条（OR、XOR）

D7 D6 D5 D4	D3 D2	D1 D0
OP-CODE	RS	RD

其中RS为源寄存器，RD为目的寄存器

由以上设计的指令系统的格式可得具体指令编码如下表：

表3-1 指令系统表

汇编符号	指令格式			功能
ADD rs,rd	0001	rs	rd	rs+rd→rd
SUB rs,rd	0010	rs	rd	rs-rd→rd
MOV rs,rd	0011	rs	rd	rs→rd
CLR rd	0100	00	rd	0→rd
STA rd	0101	M	rd	rd→E
	D			

LDA rd	0110	M	rd	E→rd
	D			
JMP M,D	0111	M	rd	E→PC
	D			
BZC M,D	1000	M	rd	当 CY=1 时 E→PC
	D			
IN addr,rd	1010	01	rd	addr→rd
OUT addr,rd	1011	10	rd	rd→addr
OR	1100	rs	rd	rs∨rd→rd
XOR	1101	rs	rd	rs⊕rd→rd

3.4 微程序控制器的逻辑结构及功能

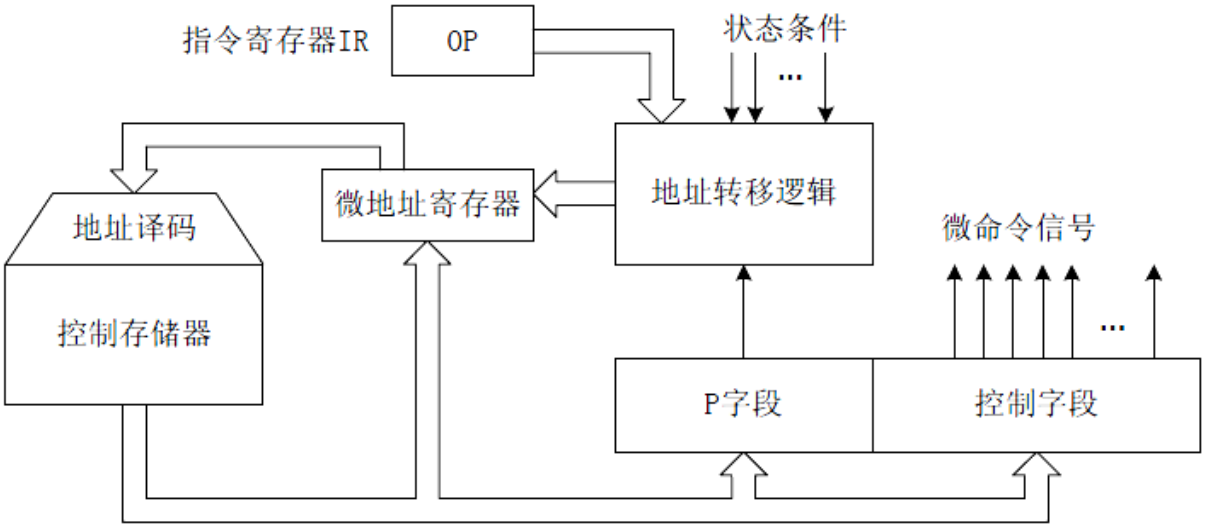

图 3-3 微程序控制器原理图

微程序控制器主要由控制存储器、微指令寄存器（包括微地址寄存器和微命令寄存器）和地址转移逻辑三大部分组成。

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。

微指令寄存器用来存放由控制存储器读出的一条微指令信息。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的信息。

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。

3.5 微程序的设计与实现

1)微指令格式设计

此模型机微指令字长 **24** 位，操作控制字段为 **24~10** 位，顺序控制字段为 **9~1** 位；顺序控制字段中判别测试字段为 **C**，下地址字段为 **6~1** 位。

微指令采用水平型微指令，微命令编码采用混合表示法，把直接表示法和地段编码法混合使用，能够满足微指令字长、灵活性、执行微程序速度等方面的要求。

微地址的形成方法为多路转移方式。当微程序不产生分支时，后继微地址直接由微指令的顺序控制字段给出；当微程序出现分支时，按顺序控制字段的“判别测试”标志和“状态条件”信息来选择其中的一个微地址。

微指令格式如下：

表 3-2 微指令格式表

24	23	22	21	20	19	18	17	16	15 14 13	12 11 10	9 8 7	6	5	4	3	2	1
S	S	S	S	M	C	W	B	B	A	B	C	UA	UA	UA	UA	UA	UA

3	2	1	0		n	E	1	0				5	4	3	2	1	0

24 位代码的含义：

S3，S2，S1，S0，M，Cn：为运算器74LS181芯片的控制信号，详见74LS181功能表，如下：

表 3-3 74LS181 功能表

	S3 S2 S1 S0	M＝0（算术运算）		M＝1（逻辑运算）
		Cn=1 无进位	Cn=0 有进位	
4位ALU	L L L L	F＝A	F＝A加1	F＝/A
	L L L H	F＝A+B	F＝（A+B）加1	F＝/(A+B)
	L L H L	F＝A+/B	F＝（A+/B）加1	F＝/A*B
	L L H H	F＝2的补	F＝0	F＝0
	L H L L	F＝A加(A*/B)	F＝A加(A*/B)加1	F＝/(A*B)
	L H L H	F＝(A+B)加(A*/B)	F＝（A+B）加(A*/B)加1	F＝/B
	L H H L	F＝A减B减1	F＝A减B	F＝$A \oplus B$
	L H H H	F＝(A*/B)减1	F＝(A*/B)	F＝(A*/B)
	H L L L	F＝A加A*B	F＝A加AB加1	F＝/A+B
	H L L H	F＝A加B	F＝A加B加1	F＝$\overline{A \oplus B}$
	H L H L	F＝（A+/B）加A*B	F＝（A+/B）加A*B加1	F＝B
	H L H H	F＝A*B减1	F＝AB	F＝AB
	H H L L	F＝A加A	F＝A加A加1	F=1

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，B1B0=00时，INPUT（即 SWB）选中；B1B0=01 时，RAM（即 CE）选中；B1B0=10时，OUTPUT（即 LEDB）选中，B1B0=11 时，外部设备都不选中。

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

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选中。

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

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：

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

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

2) 微程序设计

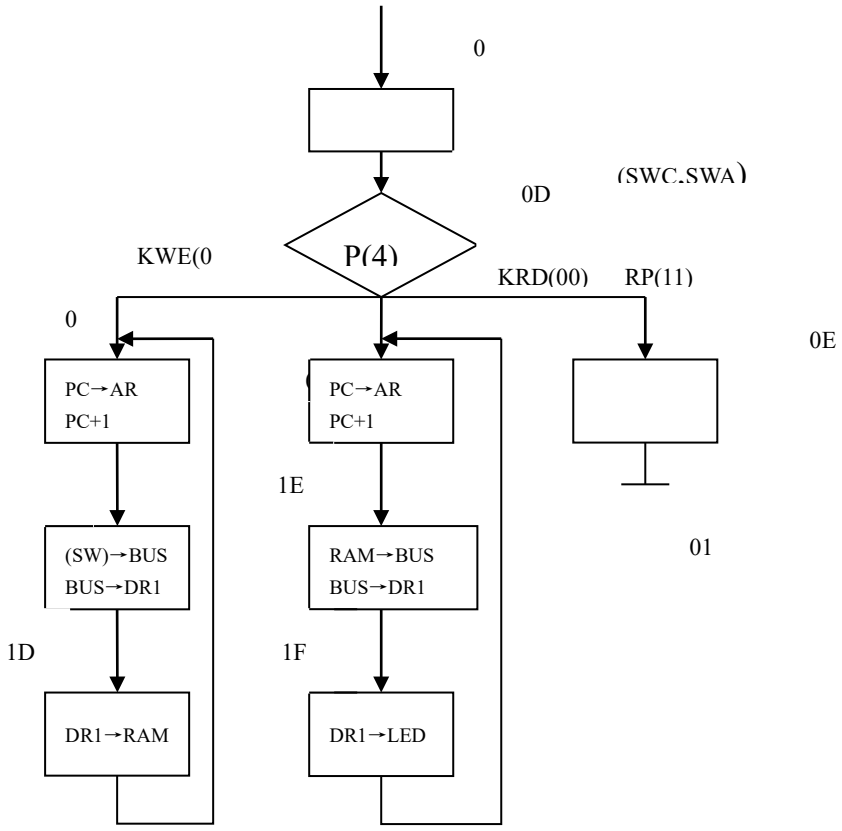
控制台
0
0D
(SWC,SWA)
P(4)
KWE(0
KRD(00)
RP(11)
0
0E
PC→AR
PC+1
PC→AR
PC+1
1E
(SW)→BUS
BUS→DR1
RAM→BUS
BUS→DR1
01
1D
1F
DR1→RAM
DR1→LED

图 3-4 微程序流程图（1）

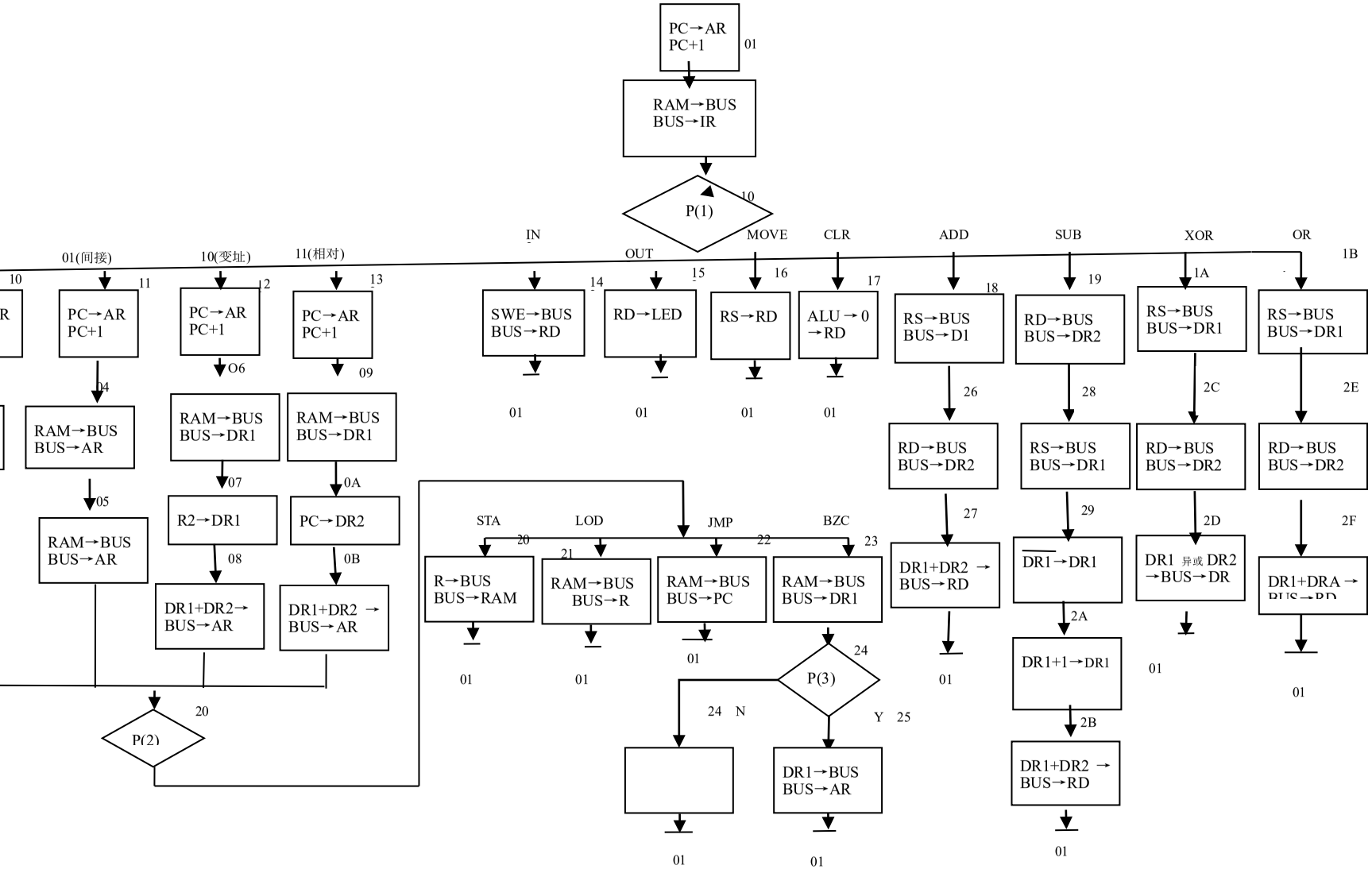
PC→AR
PC+1
01
RAM→BUS
BUS→IR
P(1)
10
IN
OUT
MOVE
CLR
ADD
SUB
XOR
OR
01(间接)
10(变址)
11(相对)
1B
PC→AR
PC+1
RAM→BUS
BUS→AR
RAM→BUS
BUS→DR1
R2→DR1
DR1+DR2→
BUS→AR
PC→DR2
DR1+DR2 →
BUS→AR
SWE→BUS
BUS→RD
RD→LED
RS→RD
ALU→0
→RD
RS→BUS
BUS→D1
RD→BUS
BUS→DR2
RS→BUS
BUS→DR1
RD→BUS
BUS→DR2
RS→BUS
BUS→DR1
RD→BUS
BUS→DR2
DR1+DR2 →
BUS→RD
DR1→DR1
DR1+1→DR1
DR1+DR2 →
BUS→RD
DR1 异或 DR2
→BUS→DR
DR1+DRA →
BUS→RD
STA
LOD
JMP
BZC
R→BUS
BUS→RAM
RAM→BUS
BUS→R
RAM→BUS
BUS→PC
RAM→BUS
BUS→DR1
P(3)
24 N
Y 25
DR1→BUS
BUS→AR
P(2)
20

图 3-5　微程序流程图（2）

表 3-4 伪指令代码表

微地址	S3	S2	S1	S0	M	CN	WE	B1	B0	A	B	C	UA5…UA0	十六进制微指令内容
00	0	0	0	0	0	1	0	1	1	000	000	100	001101	0D8105
01	0	0	0	0	0	1	0	1	1	110	110	110	000010	82ED05
02	0	0	0	0	0	1	0	0	1	100	000	001	010000	50C004
03	0	0	0	0	0	1	0	0	1	110	000	101	100000	60E104
04	0	0	0	0	0	1	0	0	1	110	000	000	000101	05E004
05	0	0	0	0	0	1	0	0	1	110	000	010	100000	A0E004
06	0	0	0	0	0	1	0	0	1	110	000	000	000111	07E004
07	0	0	0	0	0	1	0	1	1	011	001	000	001000	08B205
08	1	0	0	1	0	1	0	1	1	110	101	010	100000	A0EA95
09	0	0	0	0	0	1	0	0	1	010	000	000	001010	0AA004
0A	0	0	0	0	0	1	0	1	1	011	110	000	001011	0BBC05
0B	1	0	0	1	0	1	0	1	1	110	101	010	100000	A0EA95
0C	0	0	0	0	0	1	0	1	1	110	110	110	001111	8FED05
0D	0	0	0	0	0	1	0	1	1	110	110	110	011110	9EED05
0E	0	0	0	0	0	1	0	1	1	000	000	000	000001	018005
0F	0	0	0	0	0	1	0	0	0	010	000	000	011101	1D2004
10	0	0	0	0	0	1	0	1	1	110	110	110	000011	83ED05
11	0	0	0	0	0	1	0	1	1	110	110	110	000100	84ED05
12	0	0	0	0	0	1	0	1	1	110	110	110	000110	86ED05
13	0	0	0	0	0	1	0	1	1	110	110	110	001001	89ED05
14	0	0	0	0	0	1	0	0	0	001	000	000	000001	011004

15	0	0	0	0	0	1	1	1	0	000	010	000	000001	010407
16	0	0	0	0	0	1	0	1	1	001	001	000	000001	019205
17	0	0	1	1	1	0	0	1	1	001	101	000	000001	019A39
18	0	0	0	0	0	1	0	1	1	010	001	000	100110	26A205
19	0	0	0	0	0	1	0	1	1	011	001	000	101000	28B205
1A	0	0	0	0	0	1	0	1	1	010	001	000	101100	2CA205
1B	0	0	0	0	0	1	0	1	1	010	001	000	101110	2EA205
1C	0	0	0	0	0	1	0	1	1	010	001	000	110000	30A205
1D	0	0	0	0	0	1	1	0	1	000	101	000	001100	0C8A06
1E	0	0	0	0	0	1	0	0	1	010	000	000	011111	1FA004
1F	0	0	0	0	0	1	0	1	0	000	101	000	001101	0D0A05
20	0	0	0	0	0	1	1	0	1	000	010	000	000001	018406
21	0	0	0	0	0	1	0	0	1	001	000	000	000001	019004
22	0	0	0	0	1	1	0	0	1	101	000	110	000001	81D10C
23	0	0	0	0	0	1	0	1	1	000	000	011	100100	E48005
24	0	0	0	0	0	1	0	1	1	000	000	000	000001	018005
25	0	0	0	0	0	1	0	0	1	101	000	110	000001	81D104
26	0	0	0	0	0	1	0	1	1	011	010	000	100111	27B405
27	1	0	0	1	0	1	0	1	1	001	101	101	000001	419B95
28	0	0	0	0	0	1	0	1	1	010	010	000	101001	29A405
29	0	0	0	0	1	0	0	1	1	101	101	000	101010	2ADA09
2A	0	0	0	0	0	0	0	1	1	010	101	000	101011	2BAA01
2B	1	0	0	1	0	1	0	1	1	001	101	000	101011	2B9A85
2C	0	0	0	0	0	1	0	1	1	011	010	000	101101	2DB405
2D	0	1	1	0	1	0	0	1	1	001	101	000	000001	019A69
2E	0	0	0	0	0	1	0	1	1	011	010	000	101111	2FB405
2F	1	0	1	1	1	0	0	1	1	001	101	000	000001	019AB9

设计此模型机的监控软件，详细如下：

$P00 44 IN 01,R0

$P01 46 IN 02,R2

$P02 82 ADD R0,R2

$P03 68 MOV R2,R0

$P04 0400 STA 00,00H,R0

$P05 A8 XOR R2,R0

$P06 00

微程序：

$ M00 0D8105

$ M01 82ED05

$ M02 50C004

$ M03 60E104

$ M04 05E004

$ M05 A0E004

$ M06 07E004

$ M07 08B205

$ M08 A0EA95

$ M09 0AA004

$ M0A 0BBC05

$ M0B A0EA95

$ M0C 8FED05

$ M0D 9EED05

$ M0E 018005

$ M0F 1D2004

$ M10 83ED05

$ M11 84ED05

$ M12 86ED05

$ M13 89ED05

$ M14 011004

$ M15 010407

$ M16 019205

$ M17 019A39

$ M18 26A205

$ M19 28B205

$ M1A 2CA205

$ M1B 2EA205

$ M1C 30A205

$ M1D 0C8A06

$ M1E 1FA004

$ M1F 0D0A05

$ M20 018406

$ M21 019004

$ M22 81D10C

$ M23 E48005

$ M24 018005

$ M25 81D104

$ M26 27B405

$ M27 419B95

$ M28 29A405

$ M29 2ADA09

$ M2A 2BAA01

$ M2B 2B9A85

$ M2C 2DB405

$ M2D 019A69

$ M2E 2FB405

$ M2F 019AB9

四.系统调试报告

1）调试结果

PC=00

PC=01->AR=00->RAM(44)

RAM(44)->IR=44->微控器

INPUT(01)->R0=01

PC=02->AR=01->RAM(46)

RAM(46)->IR=46->微控器

INPUT(02)->R2=02

PC=03->AR=02->RAM(82)

RAM(82)->IR=82->微控器

R0=01->DR1

R2=02->DR2

AUL=03->R2

PC=04->AR=03->RAM(68)

RAM(68)->IR=68->微控器

R2=03->DR1

ALU=03->R0=03

PC=05->AR=04->RAM(0400)

RAM(0400)->IR=0400->微控器

R0=03->DR1

00->AR=00

ALU=03->RAM(03)

PC=06->AR=05->RAM(A8)

RAM(A8)->IR= A8->微控器

R2=03->DR1

R0=03->DR2

ALU=00->R0

PC=07->AR=06->RAM(00)

RAM(00)->PC=00

2）调试时的问题及解决

在调试程序时，由于理论和实践没有很好的结合在一起，因而遇到了很多问题，总结起来有以下两点：

1.接线错误。 例如：没有检查排线是否正常，或者由于粗心排线的插孔没有对齐，排线接错，导致程序运行错误。

2.在写程序时的错误。输入错误，细心检查后改正。

设计总结

在本次课程设计中，在规定的时间内，基本上完成了课程设计的要求，开发出了题目所要求的系统。

通过本次课程设计，使我对计算机组成原理的理论有了更深刻的认识，对计算机中各模块功能以及各模块间的联系有了更深刻的了解，包括运算器模块，存储器模块，控制器模块，指令系统，输入输出设备等。

在两周的实验中，我设计了一个简单的模型机，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。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。

通过本次系统设计，使我深信，只要能够在实践中认真思考，就会有收获。关键在于自己能不能认真对待,能不能亲自动手做。另外与老师同学的交流使自己更好地掌握了书本中的知识。

参考文献

[1] 白中英.计算机组成原理.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.科学技术出版社，2006.8

[3] 王爱英.计算机组成与结构，清华大学出版社，1999

[4] 王诚.计算机组成与结构，清华大学出版社，1999

[5] 唐朔飞.计算机组成原理，高等教育出版社，1993

致　　谢

两周的课程设计一晃而过，在设计过程中，感触最深的便是实践联系理论的重要性，当遇到实际问题时，只要认真思考，用所学的知识再一步步探索，是完全可以解决遇到的一般问题的。但是可能由于我对这个课程设计理解上的错误，导致在做的过程中出现意思偏差，没有按照任务要求设计。经过老师的指点，我明白和理解了如何设计和如何做这样的课程设计。我得到了一次难得的锻炼机会，加深了对理论知识的理解，也让我更加深刻地体会到自学能力的重要性。课程设计让我真正做到了学有所用，在设计当中受益匪浅。

通过此次课程设计，我由实践巩固了书本知识，加深了对计算机组成的工作原理的理解。我的基础知识都是通过课堂老师的细心讲解获得，所以在这儿我要感谢我的计算机组成原理的授课老师包仲贤老师，除了给予我知识基础，还给予我解决问题的思想方法。同时也应感谢我的同学，是他们的细心帮助，才使我能够完成这次课程设计。