桂林電子科技大學
GUILIN UNIVERSITY OF ELECTRONIC TECHNOLOGY
正德厚學 篤行致新

计算机组成原理课程设计说明书

题　　目：求最大负数的绝对值的 CISC 模型计算机

院（系）：计算机科学与工程学院

专　　业：计算机科学与技术

学生姓名：罗盛明

学　　号：1100310218

指导教师：陈智勇

前言： 这个 CISC 模型机我是在陈老师设计的基础上加以修改完成的；大部分电路图和 VHDL 程序的设计我沿用陈老师的设计，而剩下的一小部分我是加以修改完成本课设所需要的模型机。布线和 VHDL 修改的地方我会在后面的图中指出。报告框架和部分内容是从 PPT 指导书上截取的。

目　录

1、课程设计的题目和内容.....1
1.1 课程设计题目.....1
1.2 课程设计完成的内容.....1
1.3 课程设计的基本要求.....1
2、完成系统的整体设计.....2
3、设计时序产生器电路.....3
4、设计机器指令格式和指令系统.....3
5、微程序流程图的设计.....6
5.1 微程序流程图.....6
5.2 微程序控制器.....7
5.3 设计地址转移逻辑电路.....8
6、设计电路.....8
7、编写汇编语言源程序.....10
8、编写机器语言源程序.....11
9、机器语言源程序的功能仿真波形图及结果分析.....12
10、故障现象和故障分析.....14
11、心得体会.....14
12、附录代码.....14

1、课程设计的题目和内容

1.1 课程设计题目

设计一台嵌入式CISC模型计算机

采用定长CPU周期、联合控制方式，并运行能完成一定功能的机器语言源程序进行验证，机器语言源程序功能如下：

1、输入包含5个整数（有符号数）的数组M，输出最大负数的绝对值。

说明：

① 5个有符号数（补码格式）从外部输入；

② 将从外部输入的5个数存到RAM的数组中；

③ 统一输入完后，依次从ARM中读入并进行处理；

1.2 课程设计完成的内容

1.完成系统的总体设计，画出模型机数据通路框图；
2.设计微程序控制器（CISC模型计算机）的逻辑结构框图；
3.设计机器指令格式和指令系统；
4.设计时序产生器电路；
5.设计所有机器指令的微程序流程图；
6.设计操作控制器单元；

在CISC模型计算机中，设计的内容包括微指令格式（建议采用全水平型微指令）、微指令代码表（根据微程序流程图和微指令格式来设计）和微程序控制器硬件电路（包括地址转移逻辑电路、微地址寄存器、微命令寄存器和控制存储器等。具体电路根据微程序控制器的逻辑结构框图、微指令格式和微指令代码来设计）。

7.设计模型机的所有单元电路，并用VHDL语言（也可使用GDF文件----图形描述文件）对模型机中的各个部件进行编程，并使之成为一个统一的整体，即形成顶层电路或顶层文件；
8.由给出的题目和设计的指令系统编写相应的汇编语言源程序；
9.根据设计的指令格式，将汇编语言源程序手工转换成机器语言源程序，并将其设计到模型机中的ROM中去；
10.使用EDA软件进行功能仿真，要保证其结果满足题目的要求；（其中要利用EDA软件提供的波形编辑器，选择合适的输入输出信号及中间信号进行调试。）

1.3 课程设计的基本要求

该课程设计作为一门独立的课程，要求学生掌握CISC模型机的组成和工作

原理（CISC 模型机可以是任何计算机组成原理教材上的结构图），学会 Altera MAX+plus Ⅱ或 Quartus Ⅱ EDA 软件的使用，能用 EDA 软件设计一个能完成一定功能的模型计算机，并通过功能仿真和在 EDA 实验平台上运行一个程序来验证模型机设计的正确性。

2、完成系统的整体设计

设计一台嵌入式 CISC 模型机

模型机是按照机组课本图 5.1 的 CPU 结构图来建立的，此结构是哈佛-三数据总线结构：

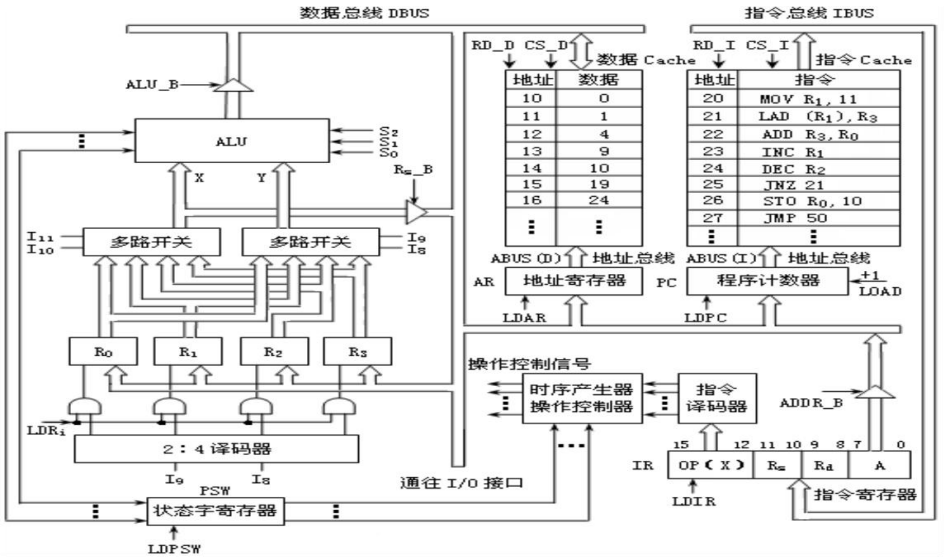


图 1

完成系统的整体设计就是要画出模型机数据通路框图，如图 1 所示。微程序控制器主要由控制存储器、微指令寄存器和地址转移逻辑电路三大部分组成，如图 2 所示。

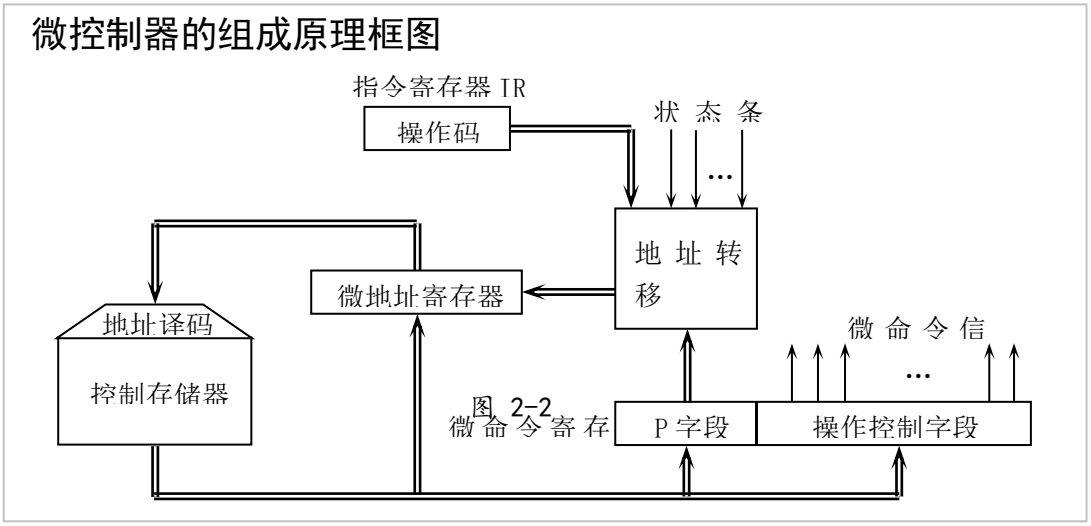


图 2

3、设计时序产生器电路

CISC 的微处理器的时钟信号 Q 和清除信号 CLR 由外部输入，节拍脉冲 Ti 由时序产生器产生。图 3 描述了节拍脉冲信号与外部时钟信号、清除信号的时序关系。

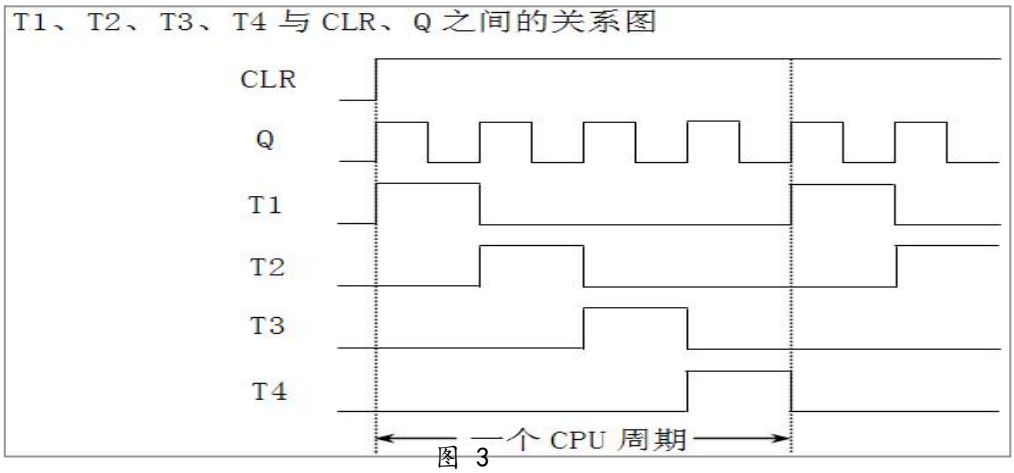


图 3

4、设计机器指令格式和指令系统

为了完成实验要求的功能，系统中设计了 9 条指令：IN1(输入)、MOV(将一个数送入寄存器)、DEC（自减 1）、TEST（测试）、JS（为负数则跳转）、ADD（两数相加）、MUL（两数相乘）、JMP（无条件跳转）、OUT1（输出）。下面分别介绍这些指令的指令格式。

（1）I/O 指令

输入（INP）指令，格式：

15-12	11 10	9 8	7-0
OP	× ×	Rd	

说明：此指令是将外部输入数据存放到 Rd 中。

输出（OUTP）指令，格式：

15-12	11 10	9 8	7-0
OP	Rs	× ×	

说明：此指令是将 Rs 中内容输出到外部显示设备。

（2）数据传输类指令

MRTR(move register to register)指令，格式：

15-12	11 10	9 8	7-0
OP	Rs	Rd	

说明：将 Rs 中内容放到 Rd

MITR(move immediate to register)指令，格式：

15-12	11 10	9 8	7-0
OP	× ×	Rd	dada

说明：将立即数 data 存放到 Rd

MMTR(move RAM to register)指令，格式：

15-12	11 10	9 8	7-0
OP	Rs	Rd	

说明：将 Rs 里对应的地址的内容存到 Rd,即从 RAM 中读取数据到 Rd。

MRTM(move register to RAM)指令，格式：

15-12	11 10	9 8	7-0
OP	Rs	Rd	

说明：将 Rs 的内容存放到 Rd 对应的地址，即将 Rs 内容存放到 RAM 中。

（3）自加指令

自加 1（INC）指令，格式：

15-12	11 10	9 8	7-0
OP	× ×	Rd	

说明：将 Rd 的内容自加 1，然后冷存放到 Rd 中。

（4）测试 TEST 指令

测试指令（TEST)采用，格式:

15-12	11 10	9 8	7-0
OP	× ×	Rd	

说明：测试 Rd 的状态，如是否为零，是正是负。将 Rd 送到 ALU 中只影响 PSW。

（5）转移指令

条件转移指令 JZ 和 JB 和无条件转移指令（JMP）指令，格式：

15-12	11 10	9 8	7-0
OP	× ×	× ×	addr

说明：JZ 若 zf=1，则程序跳转到 addr

JB 若 sf=1，则程序跳转到 addr

JMP 直接跳转到 addr。

（6）减法和比较指令

比较指令 CMP 和减法指令 SUB，格式：

15-12	11 10	9 8	7-0
OP	Rs	Rd	

说明：CMP Rd-Rs ,只影响 PSW，不存结果。
SUB Rd-Rs,影响 PSW，保存结果到 Rd。

“Rs”为源寄存器，存放的是源操作数。

“Rd”为目的寄存器，存放的是目的操作数所在的地址。

以下是对 Rs 和 Rd 的选择编码规定：

Rs 或 Rd	选定的寄存器
0 0	R0
0 1	R1
1 0	R2

模型机规定数据的表示采用定点整数补码表示，单字长为 8 位，其格式如下：

7	6 5 4 3 2 1 0
符号位	尾数

表 1 列出了每一条指令的助记符号、指令格式和功能。

13 条指令的助记符号、指令格式和功能

指令助记符	指令格式				功能
	15 - 12	11 10	9 8	7----------0	
MRTR	0001	Rs	Rd	××××××××	Rs —> Rd
MITR	0010	××	Rd	imm	立即数 imm —> Rd
MMTR	0011	Rs	Rd	××××××××	(Rs) —> Rd
MRTM	0100	Rs	Rd	××××××××	Rs —> (Rd)
INP	0101	××	Rd	××××××××	输入设备 —> Rd
OUTP	0110	Rs	××	××××××××	Rs —> 输出设备
INC	0111	××	Rd	××××××××	Rd+1 —> Rd
CMP	1000	Rs	Rd	××××××××	Rd - Rs
SUB	1001	Rs	Rd	××××××××	Rd - Rs —> Rd
JZ	1010	××	××	addr	若 zf=1 则跳转到 addr
JB	1011	××	××	addr	若 sf=1 则跳转到 addr
JMP	1100	××	××	addr	跳转到 addr
TEST	1101	××	Rd	××××××××	测试 Rd 的状态

表 1

5、微程序流程图的设计

现在,开始微程序控制器的设计,它包括以下几部分工作:

（1）根据指令格式和指令系统设计所有机器指令的微程序流程图，并确定每条微指令的微地址和后继微地址；

（2）设计微指令格式和微指令代码表；

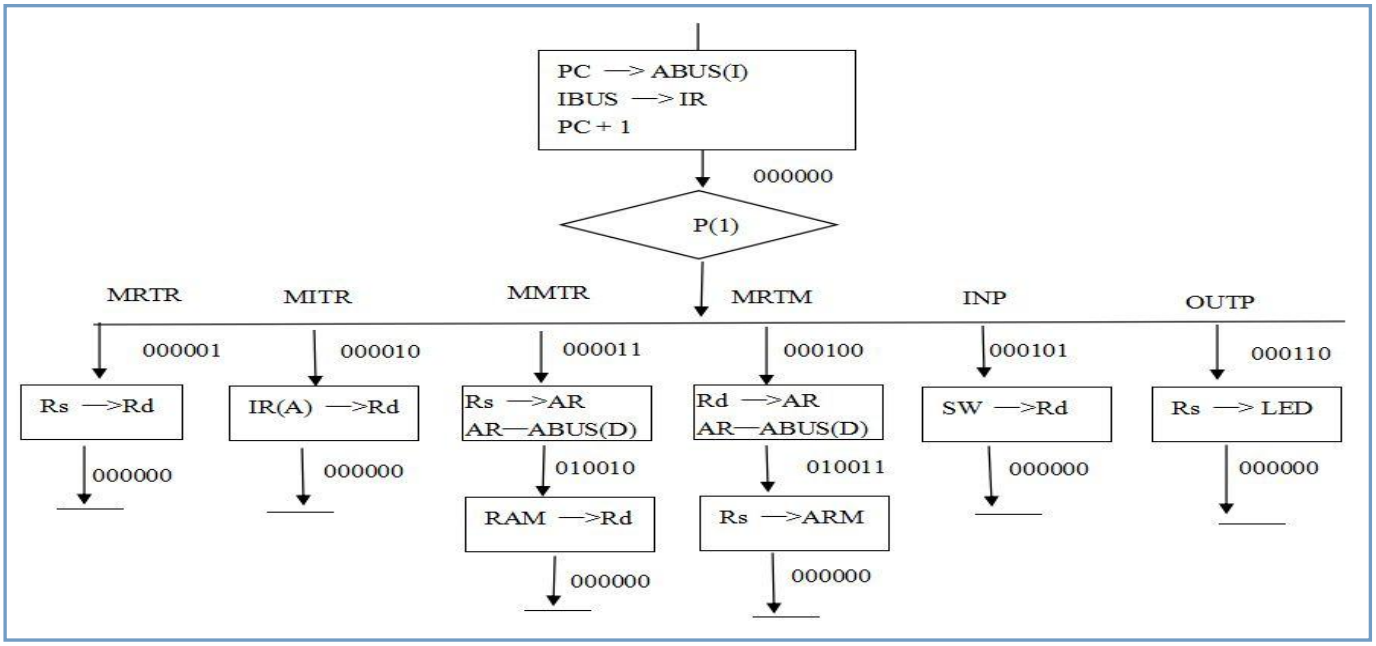
（3）设计地址转移逻辑电路；

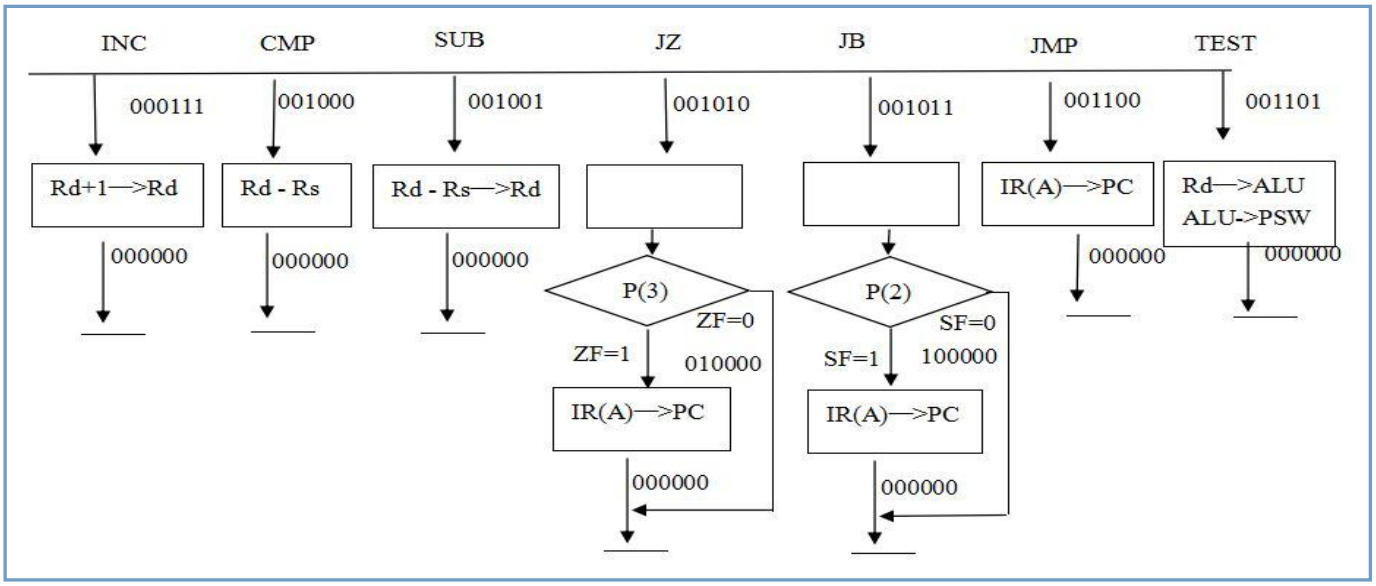
（4）设计微程序控制器中的其它逻辑单元电路，包括微地址寄存器、微命令寄存器和控制存储器；

（5）设计微程序控制器的顶层电路（由多个模块组成）。

5.1 微程序流程图

首先，根据指令格式和指令系统设计所有机器指令的微程序流程图。对于我设计的模型机，对应的指令流程图 4 所示：





5.2 微程序控制器

指令流程图设计完成后，开始设计微指令格式和微指令代码表，按照要求，CISC 模型机系统使用的微指令采用全水平型微指令，字长为 25 位，其中微命令字段为 17 位，P 字段为 2 位，后继微地址为 6 位，其格式如下：

27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD	LDPC	LDAR	LDIR	LDRi	LDPSW	RS_B	S2	S1	S0	ALU_B	SW_B	LED_B	RD_D	CS_D	arm_b	CS_I	LDDR_B	P1	P2	P3	后继微地址						

按照这个格式，再根据我设计的指令流程图，可以得到：

由微指令格式和微程序流程图编写的微指令代码表如下所示，在微指令的代码表中微命令字段从左边到右代表的微命令信号依次为：

	微地址	LOAD	LDPC	LDAR	LDIR	LDRi	LDPSW	Rs_B	S2	S1	S0	ALU_B	SW_B	LED_B	RD_D	CS_D	arm_b	CS_I	ADDR_B	P1	P2 (sf)	P3(zf)	后继微地址	
取指	000000	1	1	0	1	0	0	1	0	0	0	1	1	1	1	1	1	0	1	1	0	0	000000	IR <- ROM
MRTR Rs Rd	000001	1	0	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	000000	Rd <- Rs
MITR Data Rd	000010	1	0	0	0	1	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	000000	Rd <- Data
MMTR (Rs) Rd	000011	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	010010	AR <- (Rs)
MRTM Rs (Rd)	000100	1	0	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1	1	0	0	0	010011	AR <- (Rd)
IN Rd	000101	1	0	0	0	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0	0	000000	Rd <- BUS
OUT Rs	000110	1	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	000000	LED <- Rs
INC Rd	000111	1	0	0	0	1	1	1	0	1	0	0	1	1	1	1	1	1	1	0	0	0	000000	ALU <- Rd ALU+1
CMP Rs Rd	001000	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	0	000000	Rs-Rd, X-Y
SUB Rs Rd	001001	1	0	0	0	1	1	1	0	0	1	0	1	1	1	1	1	1	1	0	0	0	000000	Rd<-Rs-Rd, X-Y
JZ Addr	001010	1	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	0	1	000000	测ZF
JB Addr	001011	1	0	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	0	000000	测SF
JMP Addr	001100	0	1	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	000000	PC <- IR(A)
TEST Rd	001101	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	000000	
zf=1	010000	0	1	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	000000	PC <- IR(A)
mmtr	010010	1	0	0	0	1	0	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	000000	Rd <- RAM
mrtm	010011	1	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	000000	RAM <- Rs
sf=1	100000	0	1	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	000000	PC <- IR(A)

5.3 设计地址转移逻辑电路

设计好了微指令代码表之后,就可以开始设计地址转移逻辑电路。地址转移逻辑电路是根据微程序流程图中的棱形框部分及多个分支微地址,利用微地址寄存器的异步置“1”端，实现微地址的多路转移。

由于微地址寄存器中的触发器异步置“1”端低电平有效，与μA5～μA0 对应的异步置“1”控制信号 SE5～SE0 的逻辑表达式为:

$SE5=\overline{SF \cdot P(2) \cdot T4}$

$SE4=\overline{ZF \cdot P(3) \cdot T4}$

$SE3=\overline{I15 \cdot P(1) \cdot T4}$

$SE2=\overline{I14 \cdot P(1) \cdot T4}$

$SE1=\overline{I13 \cdot P(1) \cdot T4}$

$SE0=\overline{I12 \cdot P(1) \cdot T4}$

6、设计电路----- 沿用陈老师的设计图，下面只列出修改的部分

设计模型机中的所有单元电路，并用 VHDL 语言对模型机中的各个部件进行编程，并使之成为一个统一的整体，即形成顶层电路。所需要生成的器件有：操作控制器 CROM、算数逻辑运算单元 ALU、状态条件寄存器单元 LS74、1:2 分配器单元 FEN2、3 选 1 数据选择器 MUX3、4 选 1 数据选择器 MUX4、程序计数器单元 PC、主存储器 ROM16、 LS273、 时序产生器 COUNTER。

（1）微地址寄存器 AA 的内部结构，如图 5 所示。

（2）微程序控制器 CROM 的内部结构，如图 6 所示。

（3）生成的顶层电路图，如图 7 所示。

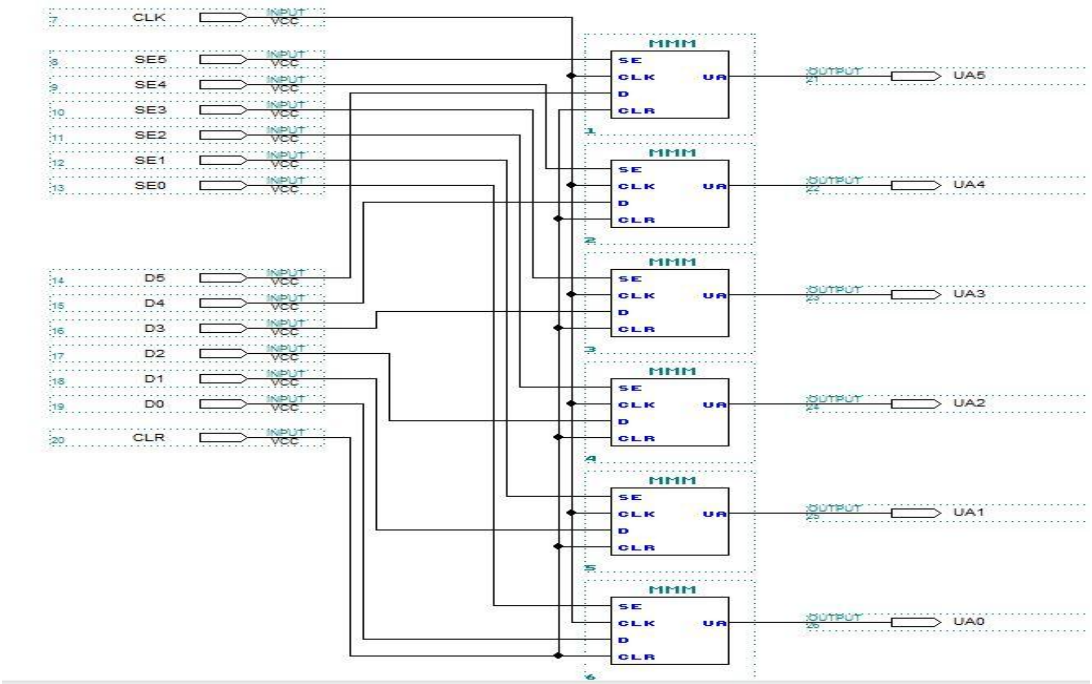


图 5 微地址寄存器 aa 的内部结构

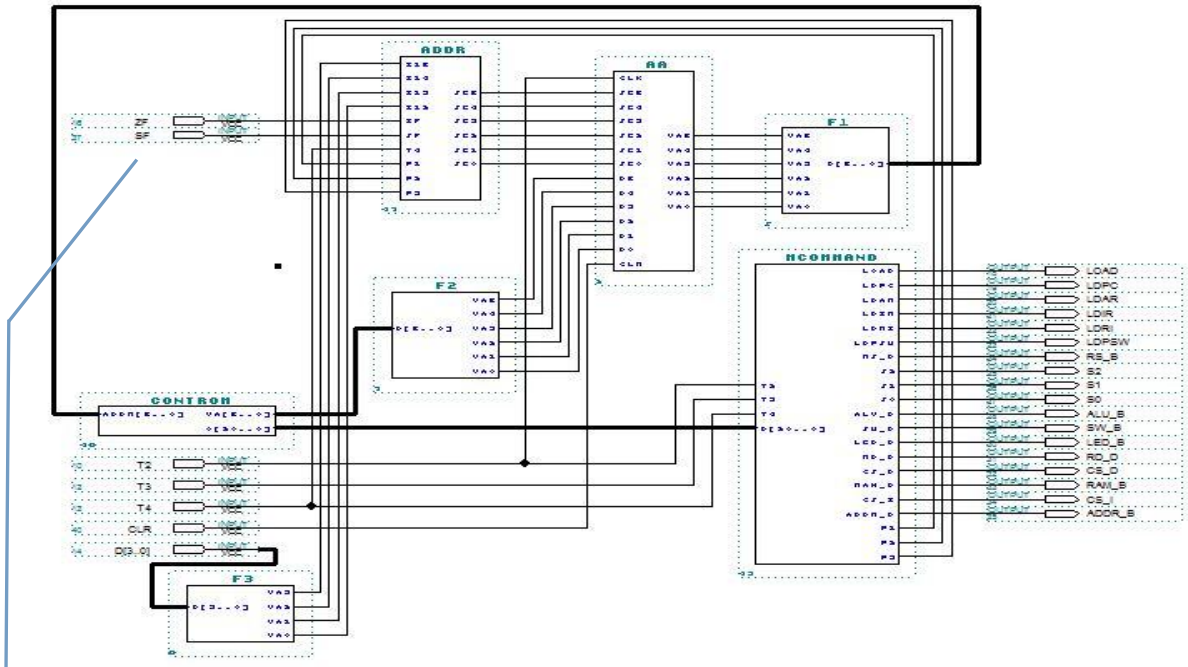
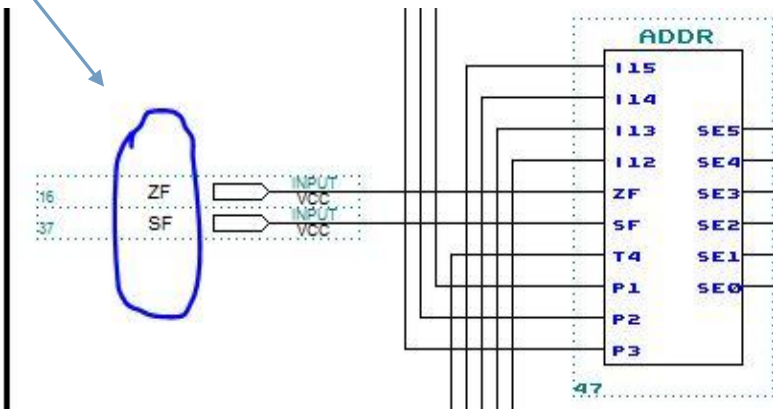


图 6　微程序控制器 CROM 的内部结构



这是我修改的其中一部分

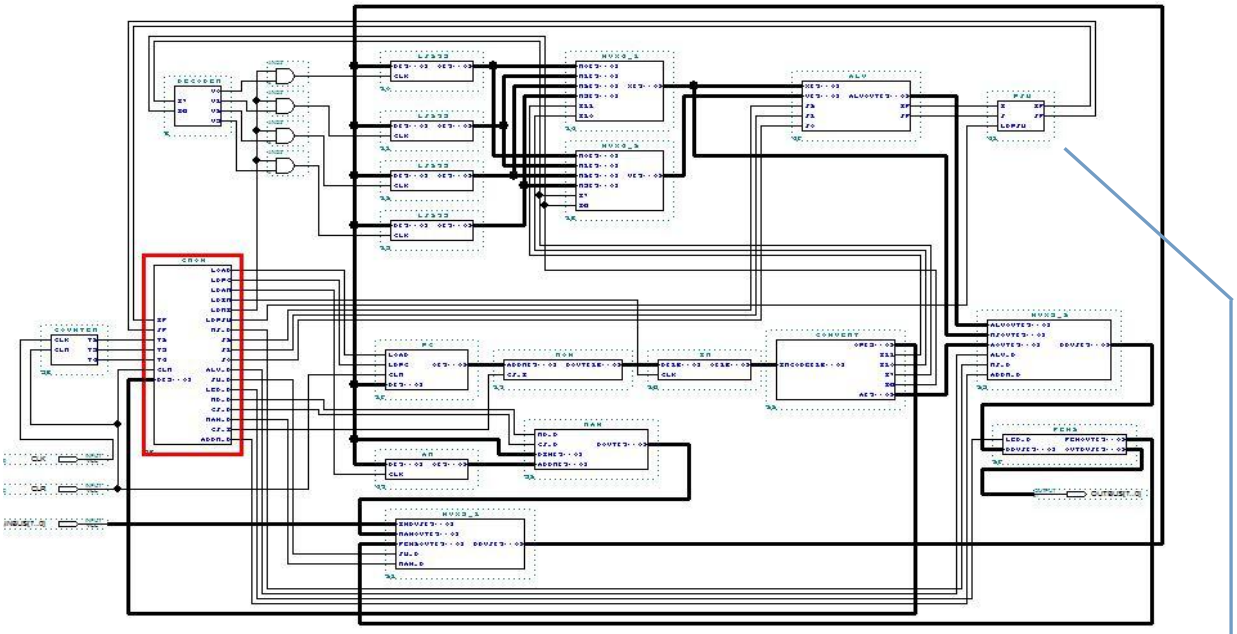
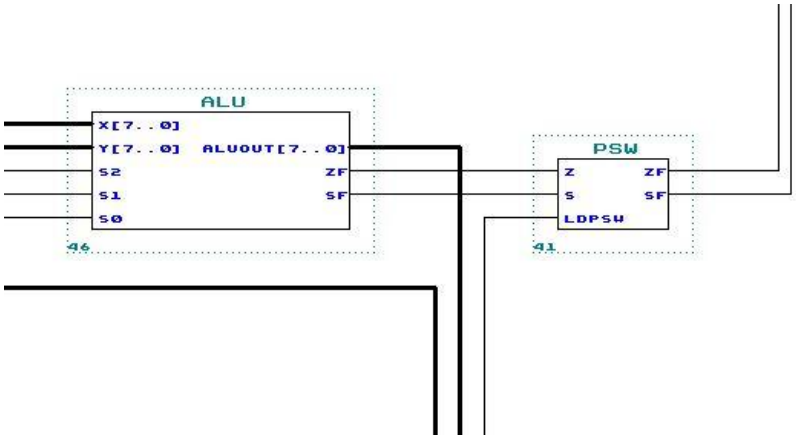


图 7 顶层电路图



这是我修改的其中一部分

7、编写汇编语言源程序

输入包含 5 个整数（有符号数）的数组 M，输出最大负数的绝对值。

思想为：通过 INP 从外部输入一个数据到 R0，然后再通过 MRTM 存放到 RAM 数组中，循环 5 次钟这样的输入，从 RAM 数组中取出一个数据，判断是否为负数，是则与 R3（发现第一个负数直接存到 R3）比较，若是最大负数则保存的到 R3，否则重新取数，依次这样循环 5 次，R1 计数为零后求 R3 中的数的绝对值和输出，

并退出程序：

汇编代码：

```
        MITR #Addr,R2
        MITR #-5,R1        ；计循环次数，
LOOP    TEST R1
        JZ START           ；检查计数是否为零
        INC R1             ；计数自加 1
        INP R0             ；输入数据
        MRTM R0,@R2        ；存放 RAM 中
        INC R2             ；数组地址加 1
        JMP LOOP           ；循环 5 次
START   MITR @Addr,R2
        MITR #-5,R1
        MITR #0,R3         ；R3 用来存最大负数，先清零
LABEL1:TEST R1
        JZ LABEL4   ;计数
        INC R1
        MMTR @R2,R0    ;R0<- ARM
        INC R2
        TEST R0
        JB LABEL2   ;判断正负
        JMP LABEL1
LABEL2: TEST R3
        JZ LABEL3          ；第一次存负数
        CMP R3,R0   ;R0-R3<0
        JB LABEL1
LABEL3: MRTR R0,R3
        JMP LABEL1
LABEL4:MITR #0,R0
        SUB R3,R0   ；求绝对值
        MMTR RO,@R2
        OUTP   R0
        END
```

8、编写机器语言源程序

根据设计的指令格式，将汇编语言源程序手工转换成机器语言源程序，并将其设计到模型机中的 ROM 中去。汇编语言源程序对应的机器语言源程序如下：

				I11 I10	I9 I8	
ROM 地址		指令	操作码	Rs	Rd	A
00000000		MITR #Addr,R2	0010	00	10	00000000
00000001		MITR #-5,R1	0010	00	01	11111011

00000010	LOOP:	TEST R1	1101	00	01	00000000
00000011		JZ START	1010	00	00	00001001
00000100		INC R1	0111	00	01	00000000
00000101		INP R0	0101	00	00	00000000
00000110		MRTM R0,@R2	0100	00	10	00000000
00000111		INC R2	0111	00	10	00000000
00001000		JMP LOOP	1100	00	00	00000010
00001001	START:	MITR @Addr,R2	0010	00	10	00000000
00001010		MITR #-5,R1	0010	00	01	11111011
00001011		MITR #0,R3	0010	00	11	00000000
00001100	LABEL1:	TEST R1	1101	00	01	00000000
00001101		JZ LABEL4	1010	00	00	00011010
00001110		INC R1	0111	00	01	00000000
00001111		MMTR @R2,R0	0011	10	00	00000000
00010000		INC R2	0111	00	10	00000000
00010001		TEST R0	1101	00	00	00000000
00010010		JB LABEL2	1011	00	00	00010100
00010011		JMP LABEL1	1100	00	00	00001100
00010100	LABEL2:	TEST R3	1101	00	11	00000000
00010101		JZ LABEL3	1010	00	00	00011000
00010110		CMP R0,R3	1000	00	11	00000000
00010111		JB LABEL1	1011	00	00	00001100
00011000	LABEL3:	MRTR R0,R3	0001	00	11	00000000
00011001		JMP LABEL1	1100	00	00	00001100
00011010	LABEL4:	MITR #0,R0	0010	00	00	00000000
00011011		SUB R0,R3	1001	00	11	00000000
00011100		MRTM R3,@R2	0100	11	10	00000000
00011101		OUT R3	0110	11	00	00000000
		END				

9、机器语言源程序的功能仿真波形图及结果分析

完成第 1 步至第 8 步的所有设计后，之前所用的 MAX+plus Ⅱ设计的模型机部分并不包含输入/输出设备，需要手动设置：Assign→Device famlily 为 ACEX1K;DevicKe 为 EP130TC144-1。在执行机器语言源程序的时候，各个寄存器随时钟的变化而变化的过程，通过分析和对比，可以检查出前面的步骤是否有错，但是并不会提醒有错误，只显示出不对应的地址或者指令。

在 CISC 模型机上的仿真波形，如图 8、图 9 和图 10 所示。图中 Q 为时钟输入信号，CLR 为外部清“0”信号，ID[7..0]是输入数据端，OD[7..0]是输出数据端，PC 是程序计数器的输出，IR 是指令寄存器的输出，DR、AC、R2、R1、R0

分别是寄存器 DR、AC、R2、R1、R0 的输出。|crom:24|controm:3|ADDR 是控制器的微地址输入端。AR 是地址的输出。

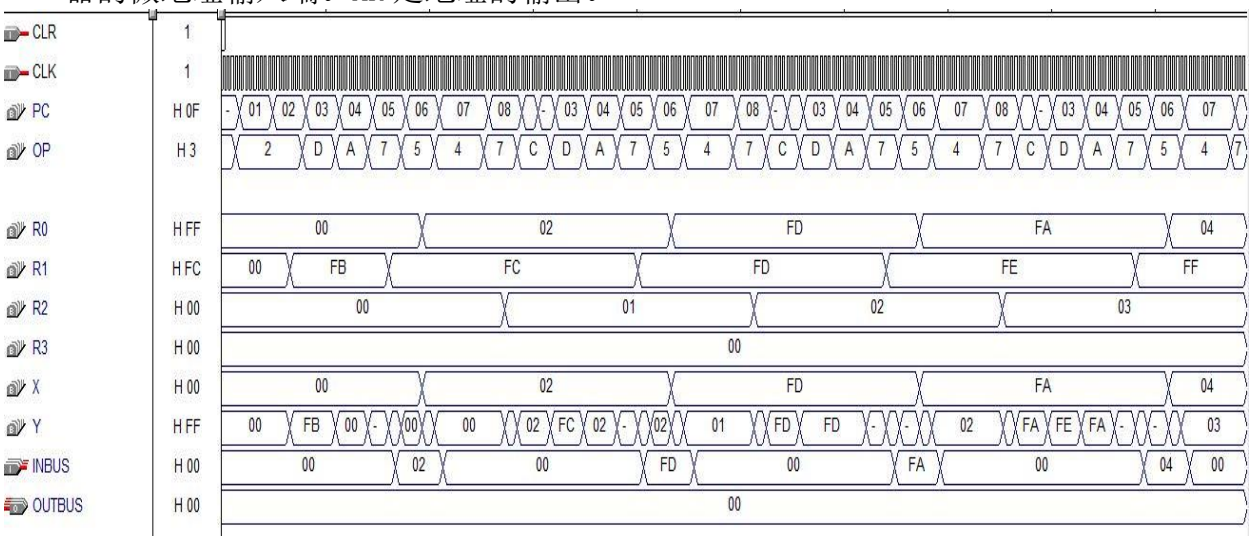


图 8　输入数据部分仿真波形

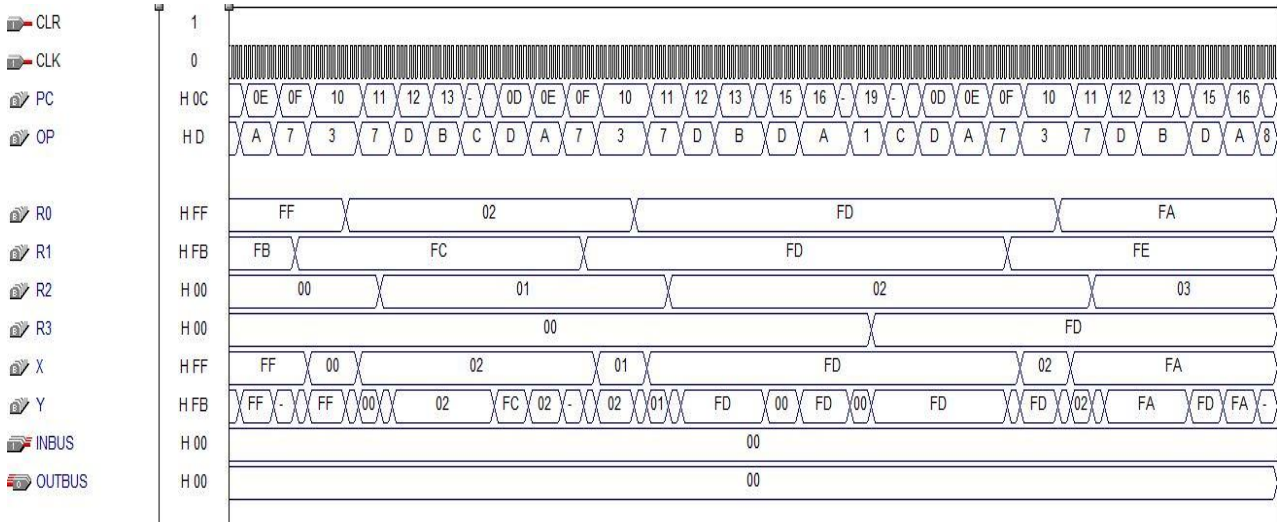


图 9 数据处理部分仿真波形

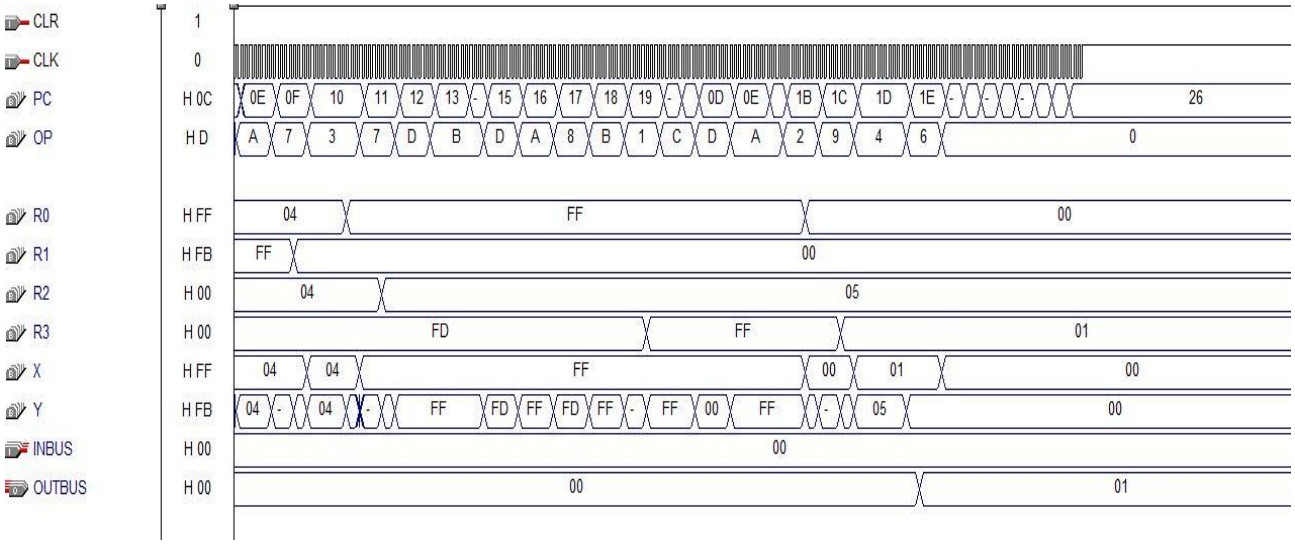


图 10 处理结果仿真波形

结果分析：

从图的 X 可以看出输入的 5 个数据分别是：02、FD、FA、04、FF ，从 R3 看输出的正确结果是 01。

结合微程序流程图和机器语言源程序，输入第一个数据 02，经过总线，送到寄存器 R0。执行 INP R0 后将数据送入寄存器 R0，MRTM R0,@R2 后将 R0 中数据送入 R2 对应的地址。每执行一条指令 R1 和 R2 都加 1，直到 R1 为零结束 5 次的循环。

从 RAM 中取出一个数如 FD（MMTR @R2,R0）后，测试这个数是否为负数（TEST R0，JB LABEL2），为 -3 ；要将 -3 存到 R3 中，经过（TEST R3，JZ LABEL3）后知道 R3 还没有任何负数可以相比较，所以第一个负数直接存入 R3，以此类推，取完 5 个数后得出结果。

10、故障现象和故障分析

测试 CISC 模型机上的仿真波形图出现的故障及其分析有：

由于我的汇编程序没有错误，所以所遇到的问题都是在微程序部分和写进 ROM 的机器码部分，有遇到跳转指令错误的情况，后来分析知道是微指令和机器码部分同时出现错误，才导致我花了好长时间去排查，还有做减法 SUB 的微指令，我将它做成两个 CPU 周期，结果是错误的，后来修改成一个 CPU 周期后就正确了。

这次做课设的进程还算顺利，是因为前期部分的工作做得比较仔细，所以在后期分析错误的时间比较容易发现问题的所在。

11、心得体会

通过这次的课设，我终于弄懂了在计算机组成原理课上不明白的一些问题，比如，指令是如何被执行的，一下子给出所有有效的微命令，它们又是如何先后有效的，即何时有效的，而通过这次实验我认识到原来时钟在 CPU 里面是多么的重要；还有知道只要控制 CPU 的 执行流程就能的控制 CPU 的行为，即 PC 的指向。

12、附录代码-----只给出我修改的部分

【程序 1】ALU 对应的 VHDL 源程序代码，如下所示：

```
PORT(
    X: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    Y: IN STD_LOGIC_VECTOR(7 DOWNTO 0);
```

```
        S2,S1,S0: IN STD_LOGIC;
        ALUOUT: OUT STD_LOGIC_VECTOR(7 DOWNTO 0) ;
        ZF,SF: OUT STD_LOGIC
      );
END ALU;

ARCHITECTURE A OF ALU IS
SIGNAL AA,BB,TEMP:STD_LOGIC_VECTOR(8 DOWNTO 0);
SIGNAL TEMP1:STD_LOGIC_VECTOR(7 DOWNTO 0);
BEGIN
      PROCESS
                        BEGIN
                        IF(S2='0' AND S1='0' AND S0='1') THEN        --CMP(SUB)
                          ALUOUT<=X-Y;
                          IF(X<Y) THEN
                                SF<='1';
                        ELSE
                          SF<='0';
                                  END IF;
            ELSIF(S2='0' AND S1='1' AND S0='0') THEN          --INC
                        AA<='0'&Y;
                        TEMP<=AA+1;
                          ALUOUT<=TEMP(7 DOWNTO 0);
                          SF<=TEMP(7);
                        IF (TEMP="100000000") THEN
                        ZF<='1';
                        ELSE
                        ZF<='0';
                        END IF;
                        ELSIF(S2='1' AND S1='1' AND S0='1') THEN          --TEST
                          TEMP1<=Y;
                          SF<=TEMP1(7);
                        IF (TEMP1="00000000") THEN
                        ZF<='1';
                        ELSE
                           ZF<='0';
                        END IF;
                        ELSE
                          ALUOUT<="00000000" ;
                        ZF<='0';
                          SF<='0';
            END IF;
      END PROCESS;
END A;
```

【程序 2】PSW 对应的 VHDL 源程序代码，如下所示：

```
ENTITY PSW IS
PORT(
        Z,S,LDPSW: IN STD_LOGIC;
        ZF,SF: OUT STD_LOGIC
      );
END PSW;
ARCHITECTURE A OF PSW IS
                        BEGIN
        PROCESS(LDPSW)
        BEGIN
              IF(LDPSW'EVENT AND LDPSW='1') THEN
                    ZF<=Z;
                    SF<=S;
              END IF;
        END PROCESS;
END A;
```

【程序 3】RAM 对应的 VHDL 源程序代码，如下所示：

```
BEGIN
      PROCESS(CS_D)
      VARIABLE MEM:MEMORY;
      BEGIN
            IF(CS_D'EVENT AND CS_D='0') THEN
                  IF(RD_D='1') THEN
                        MEM(CONV_INTEGER(ADDR(4 DOWNTO 0))):=DIN;
                  ELSE
                        DOUT<=MEM(CONV_INTEGER(ADDR(4 DOWNTO 0)));
                  END IF;
            END IF;
      END PROCESS;
END A;
```

【程序 4】ADDR 对应的 VHDL 源程序代码，如下所示：

```
PORT(
        I15,I14,I13,I12:IN STD_LOGIC;
        ZF,SF,T4,P1,P2,P3:IN STD_LOGIC;
        SE5,SE4,SE3,SE2,SE1,SE0:OUT STD_LOGIC
);
END ADDR;
ARCHITECTURE A OF ADDR IS
BEGIN
        SE5<=NOT(SF AND P2 AND T4);
        SE4<=NOT(ZF AND P3 AND T4);
```

```
        SE3<=NOT(I15 AND P1 AND T4);
        SE2<=NOT(I14 AND P1 AND T4);
        SE1<=NOT(I13 AND P1 AND T4);
        SE0<=NOT(I12 AND P1 AND T4);
END A;
```

【程序 5】指令代码转换器 F3 对应的 VHDL 源程序代码，如下所示：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY F3 IS
PORT(
                        UA7,UA6,UA5,UA4,UA3,UA2,UA1,UA0:OUT STD_LOGIC;
                        D:IN STD_LOGIC_VECTOR(7 DOWNTO 0)
                        );
END F3;
ARCHITECTURE A OF F3 IS
BEGIN
                        UA7<=D(7);
                        UA6<=D(6);
                        UA5<=D(5);
                        UA4<=D(4);
                        UA3<=D(3);
                        UA2<=D(2);
                        UA1<=D(1);
                        UA0<=D(0);
END A;
```

【程序 7】微命令寄存器 MCOMMAND 对应的 VHDL 源程序代码，如下所示：

```
ENTITY MCOMMAND IS
PORT(
       T2,T3,T4:IN STD_LOGIC;
       D:IN STD_LOGIC_VECTOR(20 DOWNTO 0);
       LOAD,LDPC,LDAR,LDIR,LDRI,LDPSW,RS_B,S2,S1,S0:OUT STD_LOGIC;
       ALU_B,SW_B,LED_B,RD_D,CS_D,RAM_B,CS_I,ADDR_B,P1,P2,P3:OUT
STD_LOGIC
      );
END    MCOMMAND;
ARCHITECTURE A OF MCOMMAND IS
SIGNAL DATAOUT:STD_LOGIC_VECTOR(20 DOWNTO 0);
BEGIN
PROCESS(T2)
      BEGIN
            IF(T2'EVENT AND T2='1') THEN
```

```
            DATAOUT(20 DOWNTO 0)<=D(20 DOWNTO 0);
        END IF;
        LOAD<=DATAOUT(20);
        LDPC<=DATAOUT(19) AND T4;
        LDAR<=DATAOUT(18) AND T3;
        LDIR<=DATAOUT(17) AND T3;
        LDRI<=DATAOUT(16) AND T4;
        LDPSW<=DATAOUT(15) AND T4;
        RS_B<=DATAOUT(14);
        S2<=DATAOUT(13);
        S1<=DATAOUT(12);
        S0<=DATAOUT(11);
        ALU_B<=DATAOUT(10);
        SW_B<=DATAOUT(9);
        LED_B<=DATAOUT(8);
        RD_D<=NOT(NOT DATAOUT(7) AND (T2 OR T3));
        CS_D<=NOT(NOT DATAOUT(6) AND T3);
        RAM_B<=DATAOUT(5);
        CS_I<=DATAOUT(4);
        ADDR_B<=DATAOUT(3);
        P1<=DATAOUT(2);
        P2<=DATAOUT(1);
        P3<=DATAOUT(0);
    END PROCESS;
END A;
```

由于我是在陈老师设计的基础上进行修改的，其他部分的 VHDL 源程序代码这这里不给出。