计算机组成原理与系统结构
实验指导书

徐展翼 程林滨 编

中国计量学院信息工程分院

2009

目录

第一章　　系统概述

1.1　DJ-CPTH 简介

DJ-CPTH 型计算机组成原理实验系统<以下简称系统>，是由江苏启东市东疆计算机有限公司结合国内同类产品的优点，最新研制开发的超强型实验计算机装置<以下简称模型机>。该系统采用单片机管理和 EDA 控制技术，自带键盘和液晶显示器，支持脱机和联 PC 机两种工作模式，运用系统监控和数码管等实时监视，全面动态管理模型机的运行和内部资源。模型机软硬件配置完整，支持 8 位字长的多种寻址方式，指令丰富，系统支持 RS-232C 串行通讯，并配有以 win98/2000/XP 为操作平台的动态跟踪集成调试软件，示教效果极佳，特别适用于计算机组成原理课程的教学与实验。

1.2　DJ-CPTH 特点

1、采用总线结构

总线结构的计算机具有结构清晰，扩展方便等优点。DJ-CPTH 实验系统使用三组总线即地址总线 ABUS、数据总线 DBUS、指令总线 IBUS 和控制信号，CPU、主存、外设和管理单片机等部件之间通过外部数据总线传输， CPU 内部则通过内部数据总线传输信息。各部件之间，通过三态缓冲器作接口连接，这样一方面增强总线驱动能力，另一方面在模型机停机时，三态门输出浮空，能保证不管模型机的 CPU 工作是否正常，管理单片机总能读/写主存或控存。

2、计算机功能模块化设计

DJ-CPTH 为实验者提供运算器模块 ALU，众多寄存器模块（A，W，IA ，ST，MAR，R0…R3 等），程序计数器模块 PC，指令部件模块 IR，主存模块 EM，微程序控制模块〈控存〉uM，微地址计数器模块 UPC，组合逻辑控制模块及 I/O 等控制模块。各模块间的电源线、地线、地址总线和数据总线等已分别连通，模块内各芯片间数据通路也已连好，各模块的控制信号及必要的输出信号已被引出到主板插孔，供实验者按自己的设计进行连接。

3、智能化控制

系统在单片机监控下，管理模型机运行和读写，当模型机停机时，实验者可通过系统键盘，读写主存或控存指定单元的内容，使模型机实现在线开发。模型机运行时，系统提供单步一条微指令（微单步）、单步一条机器指令（程单步），连续运行程序及无限止暂停等调试手段，能动态跟踪数据，流向、捕捉各种控制信息，实时反映模型机现场，使实验者及时了解程序和微程序设计的正确性，便以修改。

4、提供两种实验模式

①手动运行“Hand……”：通过拨动开关和发光二极管二进制电平显示，支持最底层的手动操作方式的输入/输出和机器调试。

②自动运行：通过系统键盘及液晶显示器或 PC 机，直接接输入或编译装载用户程序<机器码程序和微程序>，实现微程序控制运行，运用多种调试手段运行用户程序，使实验者对计算机组成原理一目了然。

5、开放性设计

运算器采用了 EDA 技术设计，随机出厂时，已提供一套已装载的方案，能进行加、减、与、或、带进位加、带进位减、取反、直通八种运算方式，若用户不满意该套方案，可自行重新设计并通过 JTAG 口下载。逻辑控制器由 CPLD 实现，也可进行重新设计并通过 JTAG 口下载。用户还可以设计自己的指令/微指令系统。系统中已带三套指令/微程序系统，用户可参照来设计新的指令/微程序系统。

系统的数据线、地址线、控制线均在总线接口区引出，并设计了 40 芯锁进插座，供用户进行 RAM、8251、8255、8253、8259 等接口器件的扩展实验。

6、支持中断实验

采用最底层的器件设计，让学生可以从微程序层面上学习中断请求、中断响应、中断处理、中断入口地址的产生、中断服务程序及中断返回（RETI）整个过程。

7、支持两种控制器实验

系统提供两种控制器方式，即微程序控制器和组合逻辑控制器。在微程序控制器中，系统能提供在线编程，实时修改程序，显示程序并进行调试的操作环境。组合逻辑控制器，已下载有一套完整的实验方案，用户也可使用 CPLD 工具在 PC 机上进行自动化设计。

8、支持子程序调用、返回、指令流水线和 RISC 精简指令系统实验。

9、配备以 Win98/2000/XP 为操作平台的集成调试软件包

系统支持 RS-232C 串行通讯，借助 PC 资源形成了强大的在线文档与图形的动态管理系统，自带编译器，支持汇编语言的编辑、编译、调试，一次点击即可完成程序和与其对应微程序的链接装载并自动弹出调试窗口，在主界面中开辟了程序和与其对应微程序的调试、模型机结构示意图（点击各模块即可修改双向模块参数）、微程序等跟踪显示窗口，供用户选择，可动态显示数据流向、实时捕捉数据、地址、控制总线的各种信息，使调试过程极为生动形象。

1.3 实验系统组成

CPTH 计算机组成原理实验系统由实验平台、开关电源、软件三大部分组成。

实验平台上有寄存器组 R0-R3、运算单元、累加器 A、暂存器 W、直通/左移/右移单元、地址寄存器、程序计数器、堆栈、中断源、输入/输出单元、存储器单元、微地址寄存器、指令寄存器、微程序控制器、组合逻辑控制器、扩展单元、总线接口区、微动开关/指示灯、逻辑笔、脉冲源、管理单片机、24 个按键、字符式 LCD、RS232。

第二章　模型机模块实验

对于硬件的描述可以有多种方法：如原理图，真值表，高级语言（本手册使用ABEL/VHDL），时序图等，在本手册中可以使用以上的四种方式来综合描述硬件。

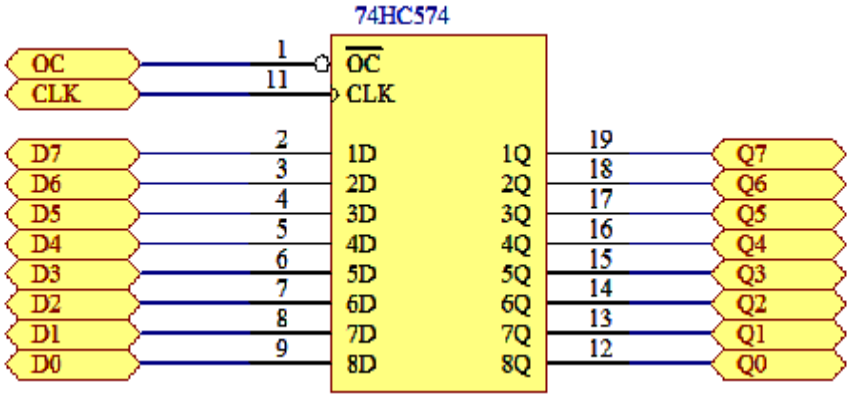
2.1 寄存器实验

实验要求：利用 CPTH 实验仪上的 K16..K23 开关做为 DBUS 的数据，其它开关做为控制信号，将数据写入寄存器，这些寄存器包括累加器 A，工作寄存器 W，数据寄存器组 R0..R3，地址寄存器 MAR，堆栈寄存器 ST，输出寄存器 OUT。

实验目的：了解模型机中各种寄存器结构、工作原理及其控制方法。

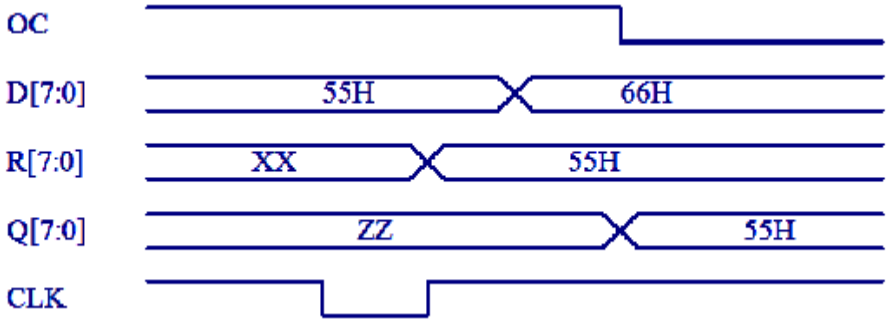
实验电路：寄存器的作用是用于保存数据的，因为我们的模型机是 8 位的，因此在本模型机中大部寄存器是 8 位的，标志位寄存器(Cy, Z)是二位的。

CPTH 用 74HC574 来构成寄存器。74HC574 的功能如下：



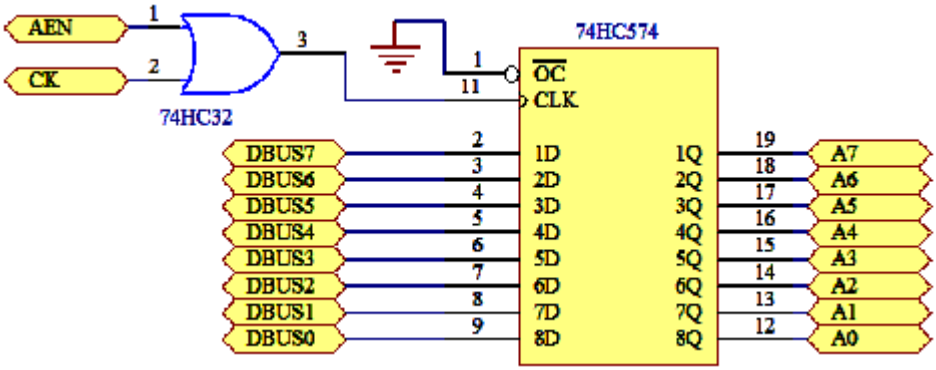
1. 在 CLK 的上升沿将输入端的数据打入到 8 个触发器中
2. 当 OC = 1 时触发器的输出被关闭，当 OC=0 时触发器的输出数据

OC	CLK	Q7..Q0	注释
1	X	ZZZZZZZZ	OC为1时触发器的输出被关闭
0	0	Q7..Q0	当OC=0时触发器的数据输出
0	1	Q7..Q0	当时钟为高时，触发器保持数据不变
X	↑	D7..D0	在CLK的上升沿将输入端的数据打入到触发器中

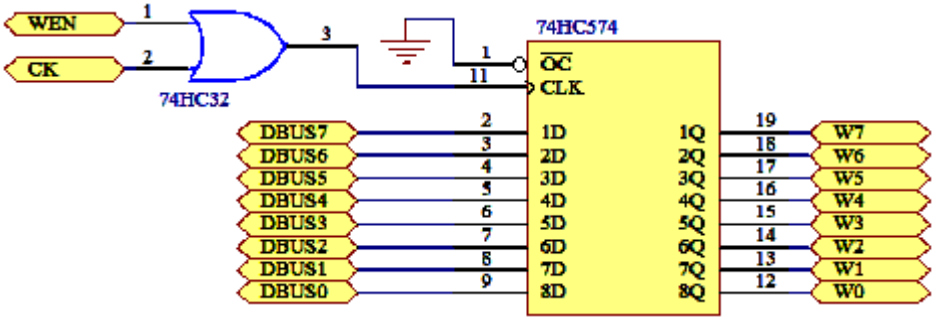


74HC574 工作波形图

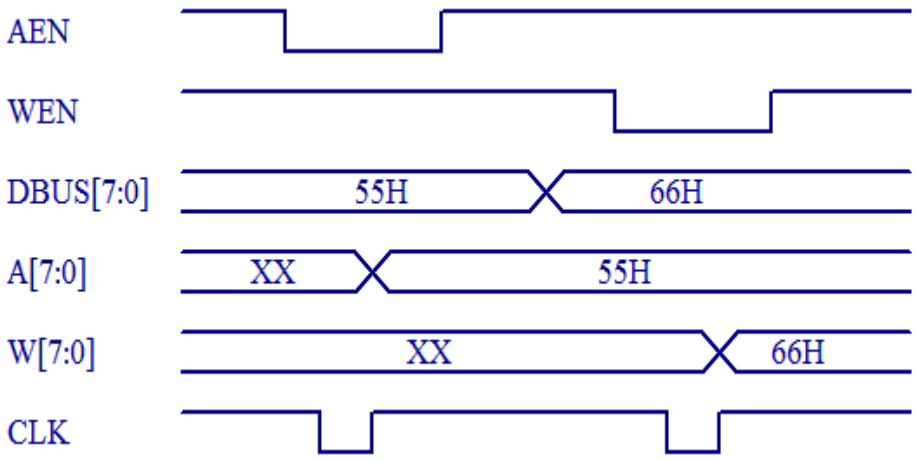
实验 1：A，W 寄存器实验



寄存器 A 原理图



寄存器 W 原理图



寄存器 A，W 写工作波形图

连接线表：

连接	信号孔	接入孔	作用	状态说明
1	J1座	J3座	将K23-K16接入DBUS[7:0]	实验模式：手动
2	AEN	K3	选通A	低电平有效
3	WEN	K4	选通W	低电平有效
4	CK	已连	ALU工作脉冲	上升沿打入

系统清零和手动状态设定：**K23-K16** 开关置零，按**[RST]**钮，按**[TV/ME]**键三次，进入**"Hand....."**手动状态。

在后面实验中实验模式为手动的操作方法不再详述.

将 55H 写入 A 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 55H

K23	K22	K21	K20	K19	K18	K17	K16
0	1	0	1	0	1	0	1

置控制信号为：

K4(WEN)	K3(AEN)
1	0

按住 STEP 脉冲键，CK 由高变低，这时寄存器 A 的黄色选择指示灯亮，表明选择 A

寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 55H 被写入 A 寄存器。

将 66H 写入 W 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 66H

K23	K22	K21	K20	K19	K18	K17	K16
0	1	1	0	0	1	1	0

置控制信号为：

K4(WEN)	K3(AEN)
0	1

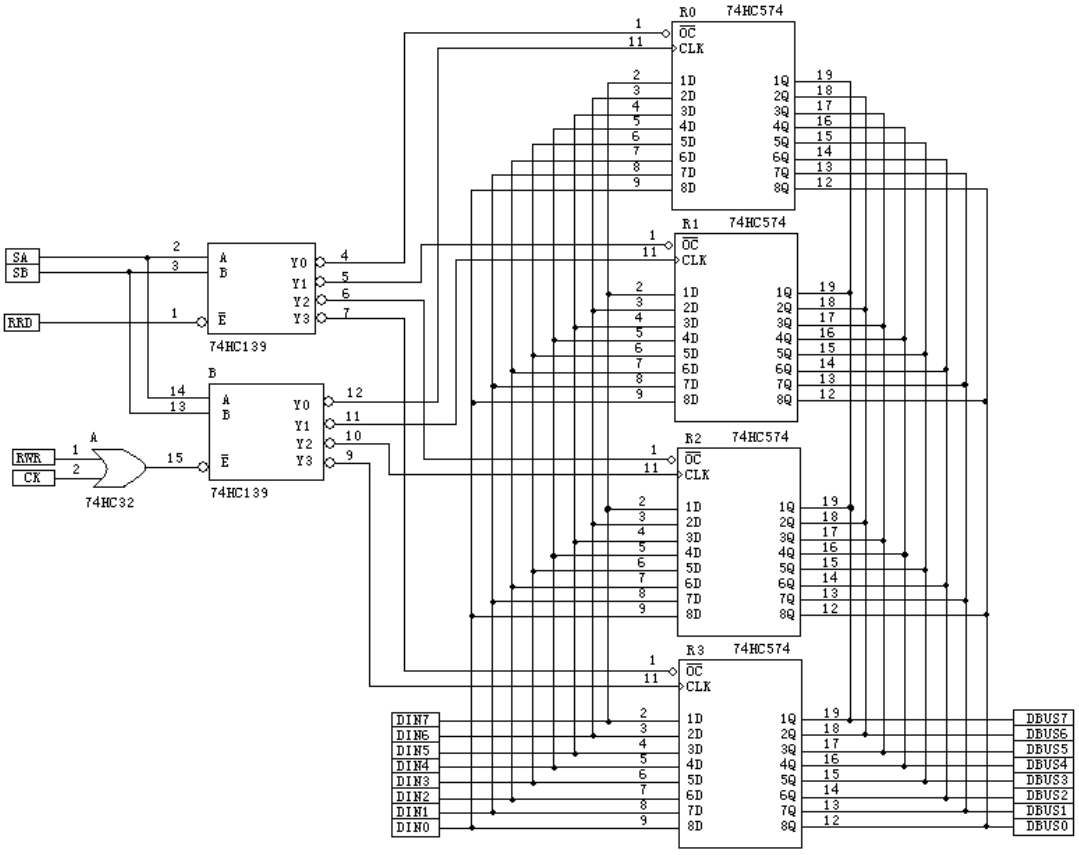
按住 STEP 脉冲键，CK 由高变低，这时寄存器 W 的黄色选择指示灯亮，表明选择 W 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 66H 被写入 W 寄存器。

注意观察：

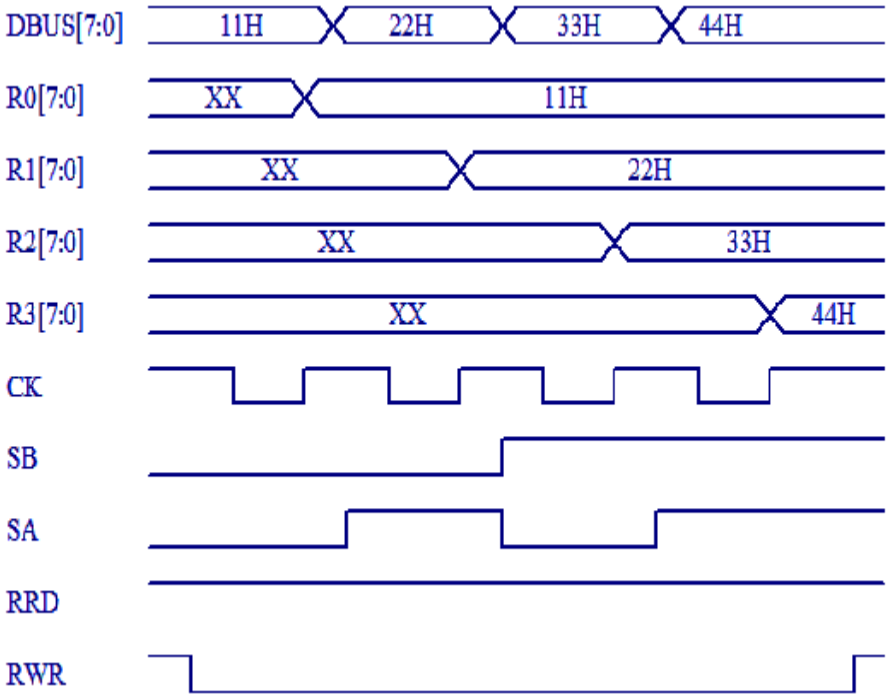
1．数据是在放开 STEP 键后改变的，也就是 CK 的上升沿数据被打入。

2．WEN，AEN 为高时，即使 CK 有上升沿，寄存器的数据也不会改变。

实验 2：R0，R1，R2，R3 寄存器实验



寄存器 R 原理图



寄存器 R 写工作波形图

连接线表

连接	信号孔	接入孔	作用	状态说明
1	J1座	J3座	将K23-K16接入DBUS[7:0]	实验模式：手动
2	RRD	K11	寄存器组读使能	低电平有效
3	RWR	K10	寄存器组写使能	低电平有效
4	SB	K1	寄存器选择B	
5	SA	K0	寄存器选择A	
6	CK	已连	寄存器工作脉冲	上升沿打入
7	D7..D0	L7..L0	观察寄存器数据输出	

将 11H 写入 R0 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 11H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	1	0	0	0	1

置控制信号为：

K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
1	0	0	0

按住 STEP 脉冲键，CK 由高变低，这时寄存器 R0 的黄色选择指示灯亮，表明选择 R0 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 11H 被写入 R0 寄存器。

将 22H 写入 R1 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 22H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	1	0	0	0	1	0

置控制信号为：

K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
1	0	0	1

按住 STEP 脉冲键，CK 由高变低，这时寄存器 R1 的黄色选择指示灯亮，表明选择 R1 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 22H 被写入 R1 寄存器。

将 33H 写入 R2 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 33H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	1	1	0	0	1	1

置控制信号为：

K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
1	0	1	0

按住 STEP 脉冲键，CK 由高变低，这时寄存器 R2 的黄色选择指示灯亮，表明选择 R2 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 33H 被写入 R2 寄存器。

将 44H 写入 R3 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 44H

K23	K22	K21	K20	K19	K18	K17	K16
0	1	0	0	0	1	0	0

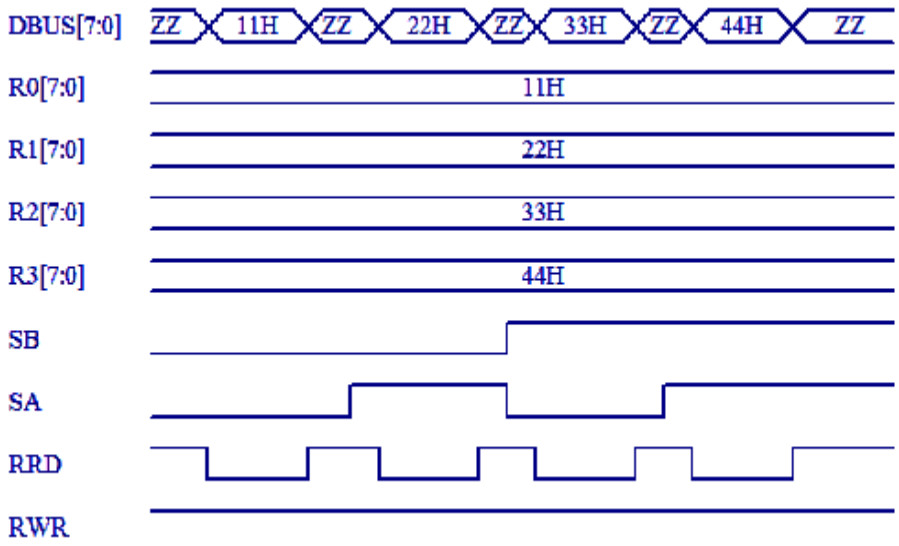
置控制信号为：

K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
1	0	1	1

按住 STEP 脉冲键，CK 由高变低，这时寄存器 R3 的黄色选择指示灯亮，表明选择 R3 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 44H 被写入 R3 寄存器。注意观察：

1. 数据是在放开 STEP 键后改变的，也就是 CK 的上升沿数据被打入。
2. K1(SB)， K0(SA) 用于选择寄存器。

K1（SB）	K0（SA）	选择
0	0	R0
0	1	R1
1	0	R2
1	1	R3



寄存器 R 读工作波形图

读 R0 寄存器

置控制信号为：

K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
0	1	0	0

这时寄存器 R0 的红色输出指示灯亮，R0 寄存器的数据送上数据总线。此时数据总线指示灯 L7... L0 为：00010001. 将 K11(RRD)置为 1，关闭 R0 寄存器输出。

读 R1 寄存器

置控制信号为：

K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
0	1	0	1

这时寄存器 R1 的红色输出指示灯亮，R1 寄存器的数据送上数据总线。此时数据总线指示灯 L7... L0 为： 00100010. 将 K11(RRD)置为 1， 关闭 R1 寄存器输出。

读 R2 寄存器

置控制信号为：

K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
0	1	1	0

这时寄存器 R2 的红色输出指示灯亮，R2 寄存器的数据送上数据总线。此时数据总线指示灯 L7... L0 为： 00110011. 将 K11(RRD)置为 1， 关闭 R2 寄存器输出。

读 R3 寄存器

置控制信号为：

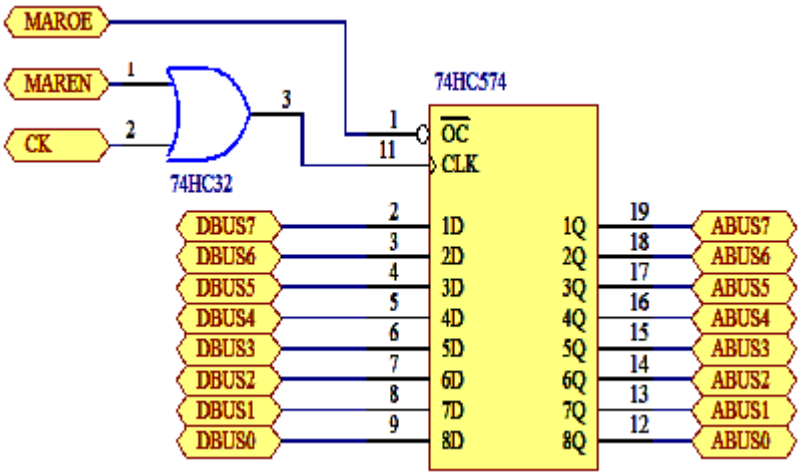
K11(RRD)	K10(RWR)	K1(SB)	K0(SA)
0	1	1	1

这时寄存器 R3 的红色输出指示灯亮，R3 寄存器的数据送上数据总线。此时数据总线指示灯 L7... L0 为： 01000100. 将 K11(RRD)置为 1， 关闭 R3 寄存器输出。

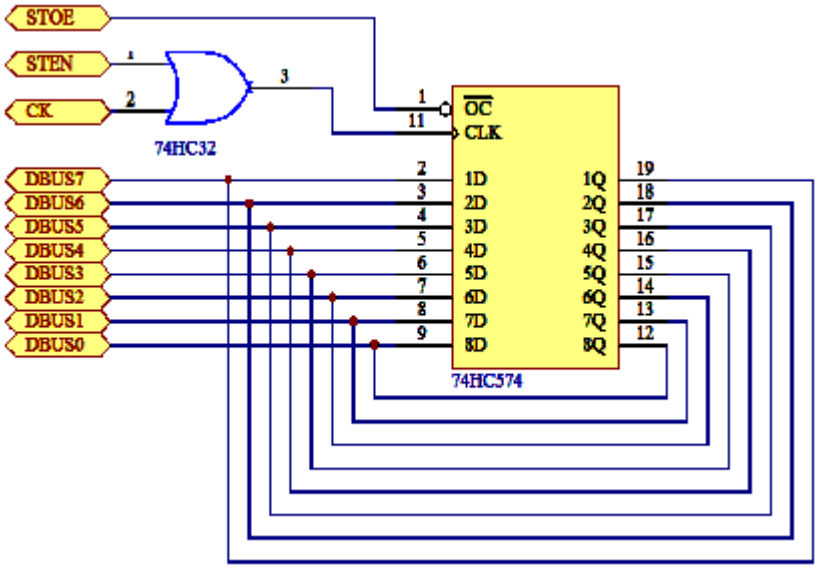
注意观察：

1.数据在 K11(RRD)为 0 时输出，不是沿触发，与数据打入不同。

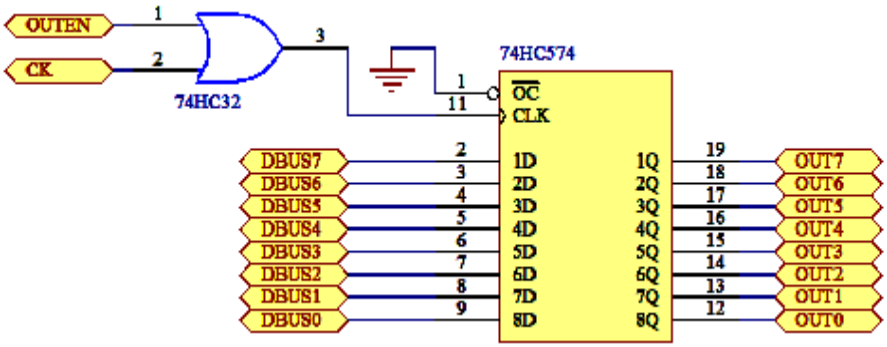
实验 3：MAR 地址寄存器，ST 堆栈寄存器，OUT 输出寄存器



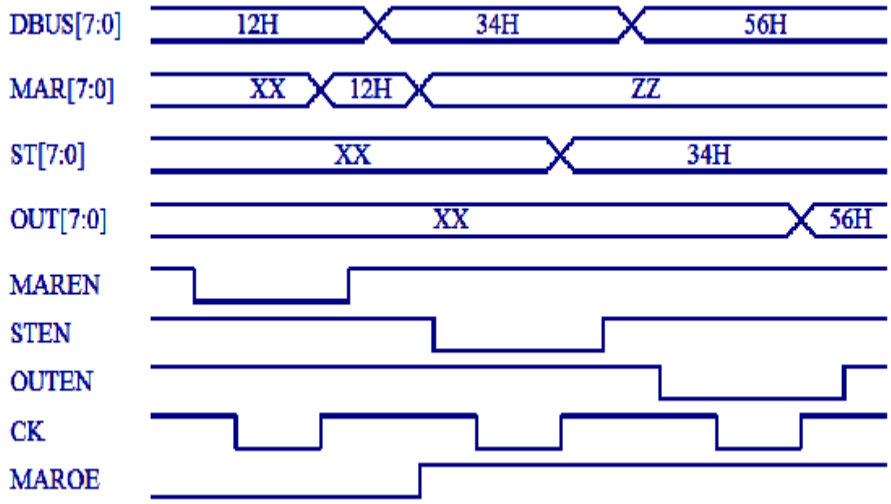
寄存器 MAR 原理图



寄存器 ST 原理图



寄存器 OUT 原理图



寄存器 MAR，ST，OUT 写工作波形图

连接线表

连接	信号孔	接入孔	作用	状态说明
1	J2座	J3座	将K23-16接入DBU[7:0]	实验模式：手动
2	MAROE	K14	MAR地址输出使能	低电平有效
3	MAREN	K15	MAR寄存器写使能	低电平有效
4	STEN	K12	ST寄存器写使能	低电平有效
5	OUTEN	K13	OUT寄存器写使能	低电平有效
6	CK	已连	寄存器工作脉冲	上升沿打入

将 12H 写入 MAR 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 12H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	1	0	0	1	0

置控制信号为：

K15(MAREN)	K14(MAROE)	K13(OUTEN)	K12(STEN)
0	0	1	1

按住 STEP 脉冲键，CK 由高变低，这时寄存器 MAR 的黄色选择指示灯亮，表明选择 MAR 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 12H 被写入 MAR 寄存器。

K14(MAROE)为 0， MAR 寄存器中的地址输出，MAR 红色输出指示灯亮。
将 K14(MAROE)置为 1，关闭 MAR 输出。

将 34H 写入 ST 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 34H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	1	1	0	1	0	0

置控制信号为：

K15(MAREN)	K14(MAROE)	K13(OUTEN)	K12(STEN)
1	1	1	0

按住 STEP 脉冲键，CK 由高变低，这时寄存器 ST 的黄色选择指示灯亮，表明选择 ST 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 34H 被写入 ST 寄存器。

将 56H 写入 OUT 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 56H

K23	K22	K21	K20	K19	K18	K17	K16
0	1	0	1	0	1	1	0

置控制信号为：

K15(MAREN)	K14(MAROE)	K13(OUTEN)	K12(STEN)
1	1	0	1

按住 STEP 脉冲键，CK 由高变低，这时寄存器 OUT 的黄色选择指示灯亮，表明选择 OUT 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 56H 被写入 OUT 寄存器。

2.2 运算器实验

实验要求：利用 CPTH 实验仪的 K16..K23 开关做为 DBUS 数据，其它开关做为控制信号，将数据写累加器 A 和工作寄存器 W，并用开关控制 ALU 的运算方式，实现运算器的功能。

实验目的：了解模型机中算术、逻辑运算单元的控制方法。

实验电路：CPTH 中的运算器由一片 CPLD 实现，有 8 种运算，通过 S2，S1，S0 来选择，运算数据由寄存器 A 及寄存器 W 给出，运算结果输出到直通门 D。

S2	S1	S0	功能	
0	0	0	A+W	加
0	0	1	A-W	减
0	1	0	A|W	或
0	1	1	A & W	与
1	0	0	A+W+C	带进位加
1	0	1	A-W-C	带进位 减
1	1	0	~A	A 取反
1	1	1	A	输出 A

连接线表

连接	信号孔	接入孔	作用	状态说明
1	J1 座	J3	将 K23-K16 接入 DBUS[7:0]	实验模式：手动
2	S0	K0	运算器功能选择	
3	S1	K1	运算器功能选择	
4	S2	K2	运算器功能选择	
5	AEN	K3	选通 A	低电平有效
6	WEN	K4	选 W	低电平有效
7	Cy IN	K5	运算器进位输入	
8	CK	已连	ALU 工作脉冲	上升沿打入

将 55H 写入 A 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 55H

K23	K22	K21	K20	K19	K18	K17	K16
0	1	0	1	0	1	0	1

置控制信号为：

K5(Cy IN)	K4(WEN)	K3(AEN)	K2(S2)	K1(S1)	K0(S0)
0	1	0	0	0	0

按住 STEP 脉冲键，CK 由高变低，这时寄存器 A 的黄色选择指示灯亮，表明选择 A 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 55H 被写入 A 寄存器。

将 33H 写入 W 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 33H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	1	1	0	0	1	1

置控制信号为：

K5(Cy IN)	K4(WEN)	K3(AEN)	K2(S2)	K1(S1)	K0(S0)
0	0	1	0	0	0

按住 STEP 脉冲键，CK 由高变低，这时寄存器 W 的黄色选择指示灯亮，表明选择 W 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 33H 被写入 W 寄存器。

置下表的控制信号，检验运算器的运算结果

K5(Cy IN)	K2(S2)	K1(S1)	K0(S0)	结果（直通门 D)	注释
X	0	0	0	88H	加运算
X	0	0	1	22H	减运算
X	0	1	0	77H	或运算
X	0	1	1	11H	与运算
0	1	0	0	88H	带进位加运算
1	1	0	0	89H	带进位加运算
0	1	0	1	22H	带进位减运算
1	1	0	1	21H	带进位减运算
X	1	1	0	AAH	取反运算
X	1	1	1	55H	输出 A

注意观察：

运算器在加上控制信号及数据(A,W)后，立刻给出结果，不须时钟。

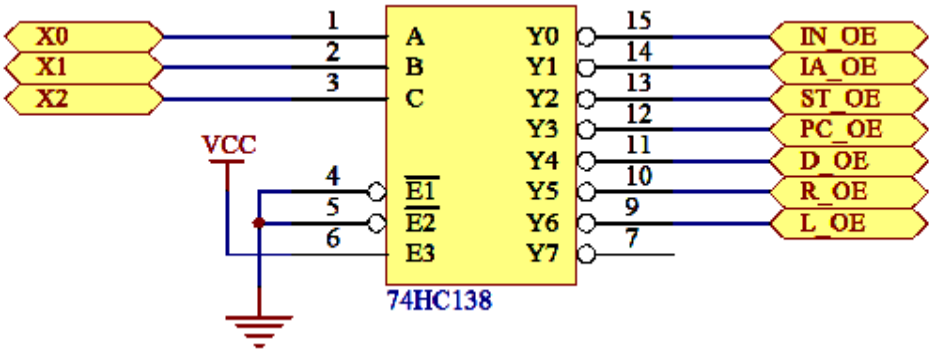
2.3 数据输出实验/移位门实验

实验要求：利用 CPTH 实验仪的开关做为控制信号，将指定寄存器的内容读到数据总线 DBUS 上。

实验目的：1、了解模型机中多寄存器接数据总线的实现原理。

2、了解运算器中移位功能的实现方法。

实验电路：CPTH 中有 7 个寄存器可以向数据总线输出数据，但在某一特定时刻只能有一个寄存器输出数据，由 X2，X1，X0 决定那一个寄存器输出数据。



数据输出选择器原理图

X0	X1	X0	输出寄存存器
0	0	0	IN_OE 外部输入门
0	0	1	IA_OE 中断向量
0	1	0	ST_OE 堆栈寄存器
0	1	1	PC_OE PC寄存器
1	0	0	D_OE 直通门
1	0	1	R_OE 右移门
1	1	0	L_OE 左移门
1	1	1	没有输出

连接线表

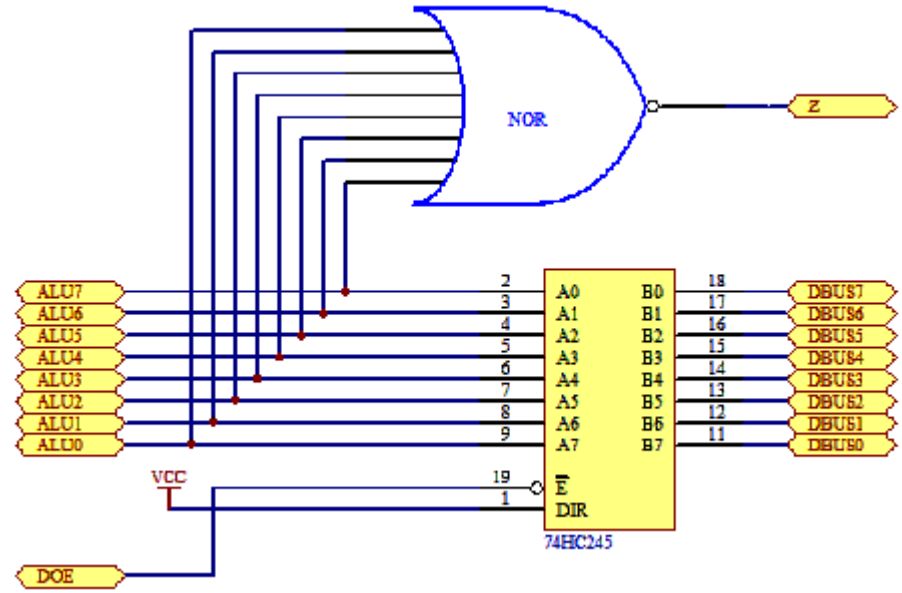
连接	信号孔	接入孔	作用	状态说明
1	J1座	J3座	将K23-K16接入DBUS[7:0]	实验模式：手动
2	X0	K5	寄存器输出选择	
3	X1	K6	寄存器输出选择	
4	X2	K7	寄存器输出选择	
5	AEN	K3	选通A	低电平有效
6	CN	K9	移位是否带进位	0:不带进位　1:带进位
7	Cy IN	K8	移位进位输入	
8	S2	K2	运算器功能选择	
9	S1	K1	运算器功能选择	
10	S0	K0	运算器功能选择	
11	CK	已连	ALU工作脉冲	上升沿打入

实验 1：数据输出实验

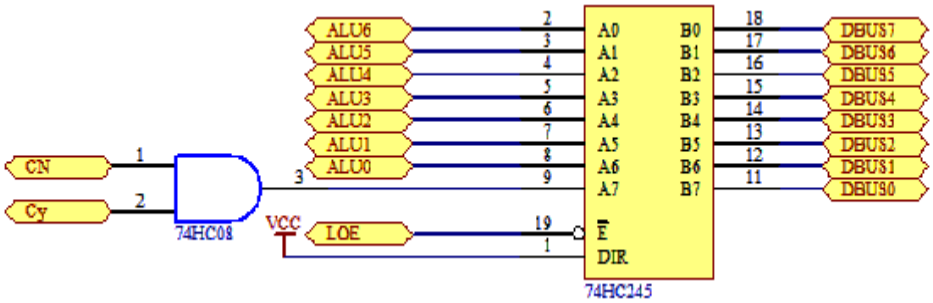
置下表的控制信号，检验输出结果

X2	X1	X0	指示灯（红色）	选通数据总线
0	0	0	IN指示	输入门（K23-16）
0	0	1	IA指示	中断向量（由拨动开关给出）
0	1	0	ST指示	堆栈寄存器
0	1	1	PC指示	PC寄存器
1	0	0	D直通门指示	D直通门
1	0	1	R右移门指示	R右移门
1	1	0	L左移门指示	L左移门
1	1	1		没有输出

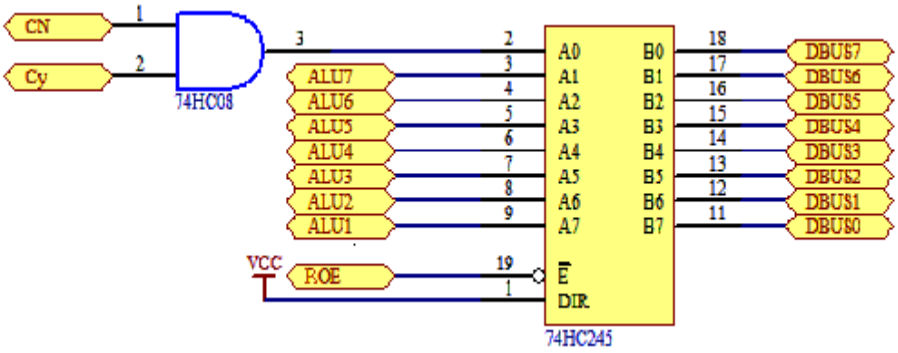
实验 2：移位实验



ALU 直接输出和零标志位产生原理图



ALU 左移输出原理图



ALU 右移输出原理图

直通门将运算器的结果不移位送总线。当 X2X1X0=100 时运算器结果通过直通门送到数据总线。同时，直通门上还有判 0 电路，当运算器的结果为全 0 时，Z=1，右移门将运算器的结果右移一位送总线。当 X2X1X0=101 时运算器结果通过右通门送到数据总线。具体内部连接是：

Cy 与 CN →DBUS7
ALU7→DBUS6
ALU6→DBUS5
ALU5→DBUS4
ALU4 → DBUS3
ALU3 → DBUS2
ALU2 → DBUS1
ALU1 → DBUS0
Cy 与 CN → DBUS7
当不带进位移位时(CN=0)：
0 →DBUS7
当带进位移位时(CN=1)：
Cy →DBUS7

左移门将运算器的结果左移一位送总线。当 X2X1X0=110 时运算器结果通过左通门送到数据总线。具体连线是：

ALU6 →DBUS7
ALU5→ DBUS6
ALU4→ DBUS5
ALU3→ DBUS4
ALU2→ DBUS3
ALU1→ DBUS2
ALU0→ DBUS1
当不带进位移位时(CN=0)：
0 → DBUS0
当带进位移位时(CN=1)：
Cy→ DBUS0

将 55H 写入 A 寄存器

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 55H

K23	K22	K21	K20	K19	K18	K17	K16
0	1	0	1	0	1	0	1

置控制信号为：

K3(AEN)	K2(S2)	K1(S1)	K0(S0)
0	1	1	1

按住 STEP 脉冲键，CK 由高变低，这时寄存器 A 的黄色选择指示灯亮，表明选择 A 寄存器。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 55H 被写入 A 寄存器。

S2S1S0=111 时运算器结果为寄存器 A 内容

CN	Cy IN	L	D	R
0	X	AA 1010 1010	55 0101 0101	2A 0010 1010
1	0	AA 1010 1010	55 0101 0101	2A 0010 1010
1	1	AB 1010 1011	55 0101 0101	AA 1010 1010

注意观察：

移位与输出门是否打开无关，无论运算器结果如何，移位门都会给出移位结果。但究竟把那一个结果送数据总线由 X2X1X0 输出选择决定。

2.4 微程序计数器 uPC 实验

实验要求：利用 CPTH 实验仪上的 K16..K23 开关做为 DBUS 的数据，其它开关做为控制信号，实现微程序计数器 uPC 的写入和加 1 功能。

实验目的：1、了解模型机中微程序的基本概念。

2、了解 uPC 的结构、工作原理及其控制方法。

实验电路：

74HC161 是一片带预置的 4 位二进制记数器。功能如下：

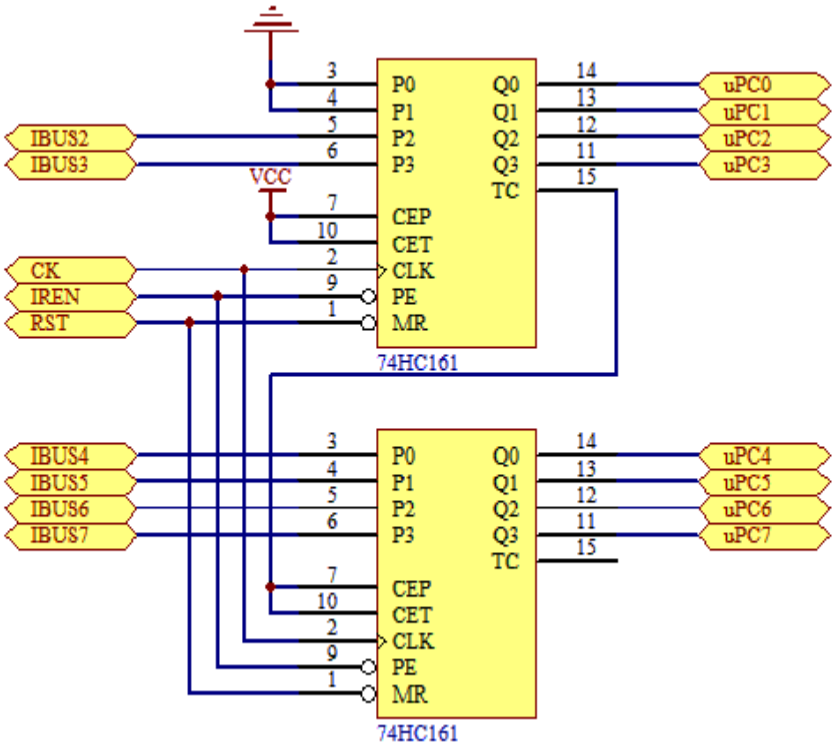
当 RST = 0 时，记数器被清 0

当 IREN = 0 时，在 CK 的上升沿，预置数据被打入记数器

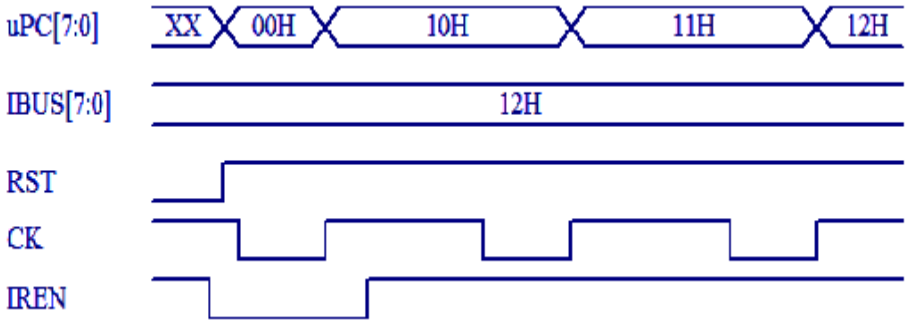
当 IREN = 1 时，在 CK 的上升沿，记数器加一

TC 为进位，当记数到 F（1111）时，TC=1

CEP，CET 为记数使能，当 CEP，CET=1 时，记数器工作，CEP，CET=0 时，记数器保持原记数值



uPC 原理图



uPC 工作波形图

在 CPTH 中，指令 IBUS[7:0]的高六位被接到 uPC 预置的高六位，uPC 预置的低两位被置为 0。一条指令最多可有四条微指令。

微程序初始地址为复位地址 00，微程序入口地址由指令码产生，微程序下一地址有计数器产生。

连接线表

连接	信号孔	接入孔	作用	状态说明
1	J2座	J3座	将K23-K16接入DBU[7:0]	实验模式：手动
2	IREN	K0	预置uPC	低电平有效
3	EMEN	K1	EM存储器工作使能	低电平有效
4	EMWR	K2	EM存储器写使能	低电平有效
5	EMRD	K3	EM存储器读使能	低电平有效
6	CK	已连	uPC工作脉冲	上升沿打入

实验 1：uPC 加一实验

置控制信号为：

K3(EMRD)	K2(EMWR)	K1(EMEN)	K0(IREN)
1	1	1	1

按一次 STEP 脉冲键，CK 产生一个上升沿，数据 uPC 被加一。

实验 2：uPC 打入实验

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 12H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	1	0	0	1	0

置控制信号为：

K3(EMRD)	K2(EMWR)	K1(EMEN)	K0(IREN)
1	0	0	0

当 EMWR，EMEN=0 时，数据总线（DBUS）上的数据被送到指令总线（IBUS）上。

按住 STEP 脉冲键，CK 由高变低，这时寄存器 uPC 的黄色预置指示灯亮，表明 uPC 被预置。放开 STEP 键，CK 由低变高，产生一个上升沿，数据 10H 被写入 uPC 寄存器。

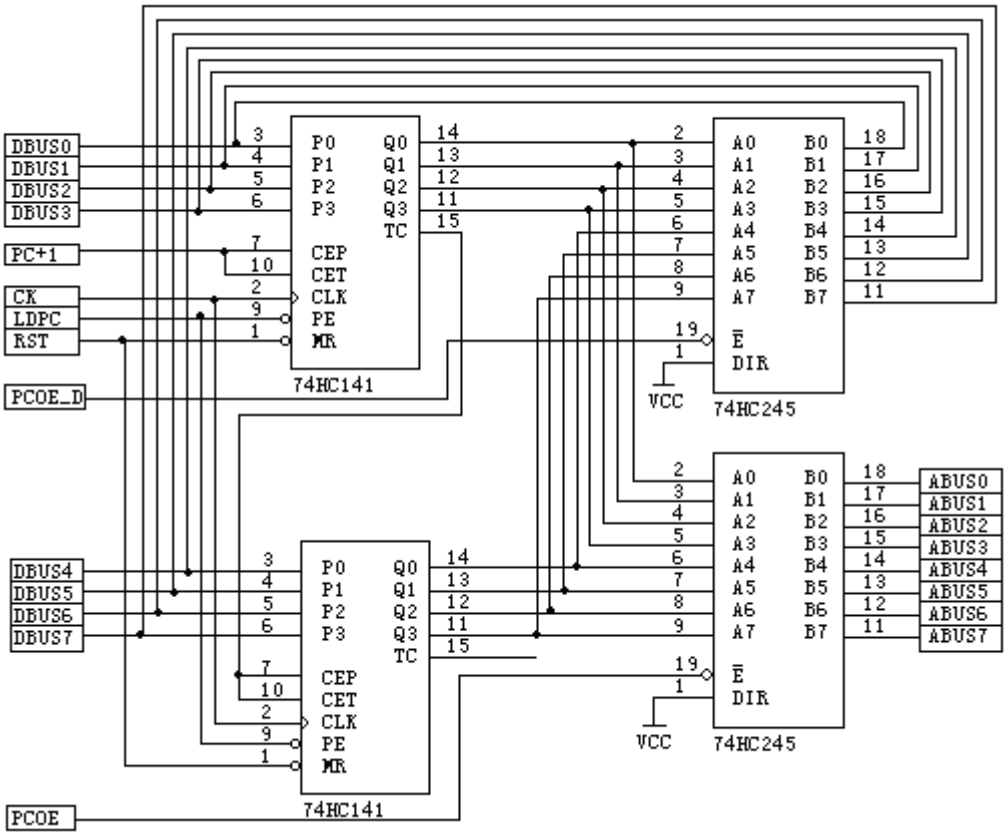
2.5 PC 实验

实验要求：利用 CPTH 实验仪上的 K16..K23 开关做为 DBUS 的数据，其它开关做为控制信号，实现程序计数器 PC 的写入及加 1 功能。

实验目的：1、了解模型机中程序计数器 PC 的工作原理及其控制方法。2、了解程序执行过程中顺序和跳转指令的实现方法。

实验电路：

PC 是由两片 74HC161 构成的八位带预置记数器，预置数据来自数据总线。记数器的输出通过 74HC245（PCOE）送到地址总线。PC 值还可以通过 74HC245（PCOE_D）送回数据总线。



PC 原理图

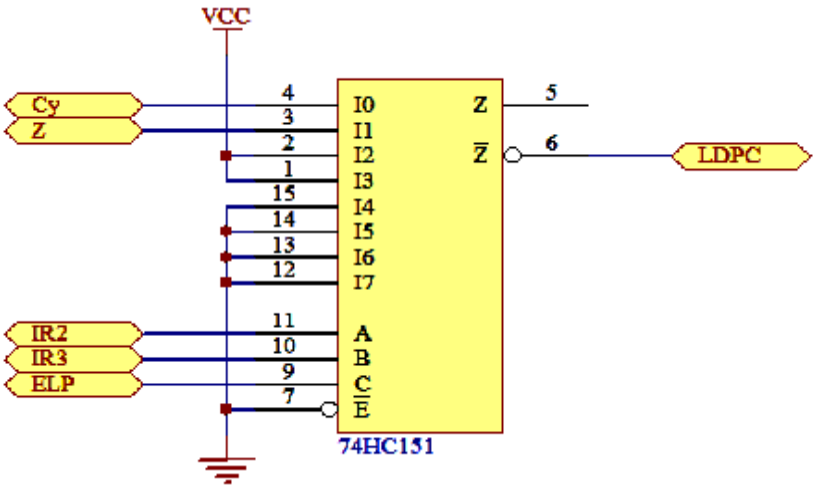
在 CPTH 中，PC+1 由 PCOE 取反产生。

当 RST = 0 时，PC 记数器被清 0

当 LDPC = 0 时，在 CK 的上升沿，预置数据被打入 PC 记数器

当 PC+1 = 1 时，在 CK 的上升沿，PC 记数器加一

当 PCOE = 0 时，PC 值送地址总线



PC 打入控制原理图

PC 打入控制电路由一片 74HC151 八选一构成(isp1016 实现)。

ELP	IR3	IR2	Cy	Z	LDPC
1	X	X	X	X	1
0	0	0	1	X	0
0	0	0	0	X	1
0	0	1	X	1	0
0	0	1	X	0	1
0	1	X	X	X	0

当 ELP=1 时，LDPC=1，不允许 PC 被预置
当 ELP=0 时，LDPC 由 IR3，IR2，Cy，Z 确定
当 IR3 IR2 = 1 X 时，LDPC=0，PC 被预置
当 IR3 IR2 = 0 0 时，LDPC=非 Cy，当 Cy=1 时，PC 被预置
当 IR3 IR2 = 0 1 时，LDPC=非 Z，当 Z=1 时，PC 被预置
连接线表

连接	信号孔	接入孔	作用	状态说明
1	J2座	J3座	将K23-K16接入DBU[7:0]	实验模式：手动
2	PCOE	K5	PC输出到地址总线	低电平有效
3	JIR3	K4	预置选择1	
4	JIR2	K3	预置选择0	
5	JRZ	K2	Z标志输入	
6	JRC	K1	C标志输入	
7	ELP	K0	预置允许	低电平有效
8	CK	已连	PC工作脉冲	上升沿打入

实验 1：PC 加一实验

置控制信号为：

K5（PCOE）	K0（ELP）
0	1

按一次 STEP 脉冲键，CK 产生一个上升沿，数据 PC 被加一。

实验 2：PC 打入实验

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 12H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	1	0	0	1	0

置控制信号为：

IR3 (K4)	IR2 (K3)	JRZ (K2)	JRC (K1)	ELP (K0)	LDPC	黄色PC预置指示灯
X	X	X	X	1	1	灭
0	0	X	1	0	0	亮
0	0	X	0	0	1	灭
0	1	1	X	0	0	亮
0	1	0	X	0	1	灭
1	X	X	X	0	0	亮

每置控制信号后，按一下 STEP 键，观察 PC 的变化。

2.6 存储器 EM 实验

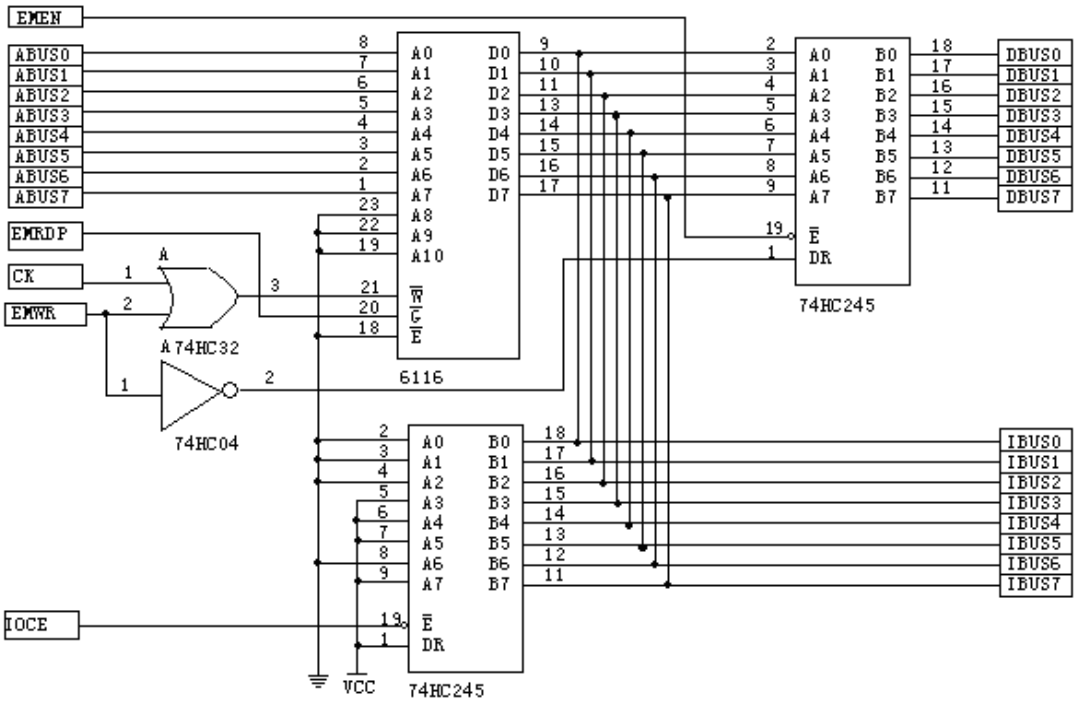
实验要求：利用 CPTH 实验仪上的 K16..K23 开关做为 DBUS 的数据，其它开关做为控制信号，实现程序存储器 EM 的读写操作。

实验目的：了解模型机中程序存储器 EM 的工作原理及控制方法。

实验电路：

存储器 EM 由一片 6116RAM 构成，是用户存放程序和数据的地方。存储器 EM 通过一片 74HC245 与数据总线相连。存储器 EM 的地址可选择由 PC 或 MAR 提供。

存储器 EM 的数据输出直接接到指令总线 IBUS，指令总线 IBUS 的数据还可以来自一片 74HC245。当 ICOE 为 0 时，这片 74HC245 输出中断指令 B8。



EM 原理图

连接线表

连接	信号孔	接入孔	作用	状态说明
1	J2座	J3座	将K23-K16接入DBUS[7:0]	实验模式：手动
2	IREN	K6	IR,uPC写允许	低电平有效
3	PCOE	K5	PC输出地址	低电平有效
4	MAROE	K4	MAR输出地址	低电平有效
5	MAREN	K3	MAR写允许	低电平有效
6	EMEN	K2	存储器与数据总线相连	低电平有效
7	EMRD	K1	存储器读允许	低电平有效
8	EMWR	K0	存储器写允许	低电平有效
9	CK	已连	PC工作脉冲	上升沿打入
10	CK	已连	MAR工作脉冲	上升沿打入
11	CK	已连	存储器写脉冲	上升沿打入
12	CK	已连	IR,uPC工作脉冲	上升沿打入

实验 1：PC/MAR 输出地址选择

置控制信号为：

K5 (PCOE)	K4 (MAROE)	地址总线	红色地址输出指示灯
0	1	PC输出地址	PC地址输出指示灯亮
1	0	MAR输出地址	MAR地址输出指示灯亮
1	1	地址总线浮空	
0	0	错误,PC及MAR同时输出	PC及MAR地址输出指示灯亮

以下存贮器 EM 实验均由 MAR 提供地址

实验 2：存储器 EM 写实验

将地址 0 写入 MAR

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 00H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	0	0	0	0	0

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	1	0	1	1	1

按 STEP 键，将地址 0 写入 MAR

将数据 11H 写入 EM[0]

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 11H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	1	0	0	0	1

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	0	1	0	1	0

按 STEP 键，将数据 11H 写入 EM[0]

将地址 1 写入 MAR

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 01H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	0	0	0	0	1

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	1	0	1	1	1

按 STEP 键，将地址 1 写入 MAR

将数据 22H 写入 EM[1]

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 22H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	1	0	0	0	1	0

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	0	1	0	1	0

按 STEP 键，将数据 22H 写入 EM[1]

实验 3：存储器 EM 读实验

将地址 0 写入 MAR

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 00H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	0	0	0	0	0

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	1	0	1	1	1

按 STEP 键, 将地址 0 写入 MAR

读 EM[0]

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	0	1	1	0	1

EM[0]被读出：11H

将地址 1 写入 MAR

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 01H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	0	0	0	0	1

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	1	0	1	1	1

按 STEP 键，将地址 0 写入 MAR

读 EM[1]

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	0	1	1	0	1

EM[1]被读出：22H

实验 4：存储器打入 IR 指令寄存器/uPC 实验
将地址 0 写入 MAR

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 00H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	0	0	0	0	0

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	1	0	1	1	1

按 STEP 键，将地址 0 写入 MAR

读 EM[0]，写入 IR 及 uPC

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
0	1	0	1	1	0	1

EM[0]被读出：11H
按 STEP 键，将 EM[0]写入 IR 及 uPC，IR=11H，uPC=10H

将地址 1 写入 MAR

二进制开关 K23-K16 用于 DBUS[7:0]的数据输入，置数据 01H

K23	K22	K21	K20	K19	K18	K17	K16
0	0	0	0	0	0	0	1

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
1	1	1	0	1	1	1

按 STEP 键，将地址 1 写入 MAR

读 EM[1]，写入 IR 及 uPC

置控制信号为：

K6 (IREN)	K5 (PCOE)	K4 (MAROE)	K3 (MAREN)	K2 (EMEN)	K1 (EMRD)	K0 (EMWR)
0	1	0	1	1	0	1

EM[1]被读出：22H
按 STEP 键，将地址 EM[1]写入 IR 及 uPC，IR=22H，uPC=20H

实验 5：使用实验仪小键盘输入 EM

1．连接 J1，J2
2．打开电源
3．按 TV/ME 键，选择 EM
4．输入两位地址，00
5．按 NEXT，进入程序修改
6．按两位程序数据
7．按 NEXT 选择下个地址/按 LAST 选择上个地址
8．重复 6，7 步输入程序
9．按 RST 结束

2.7 微程序存储器 uM 实验

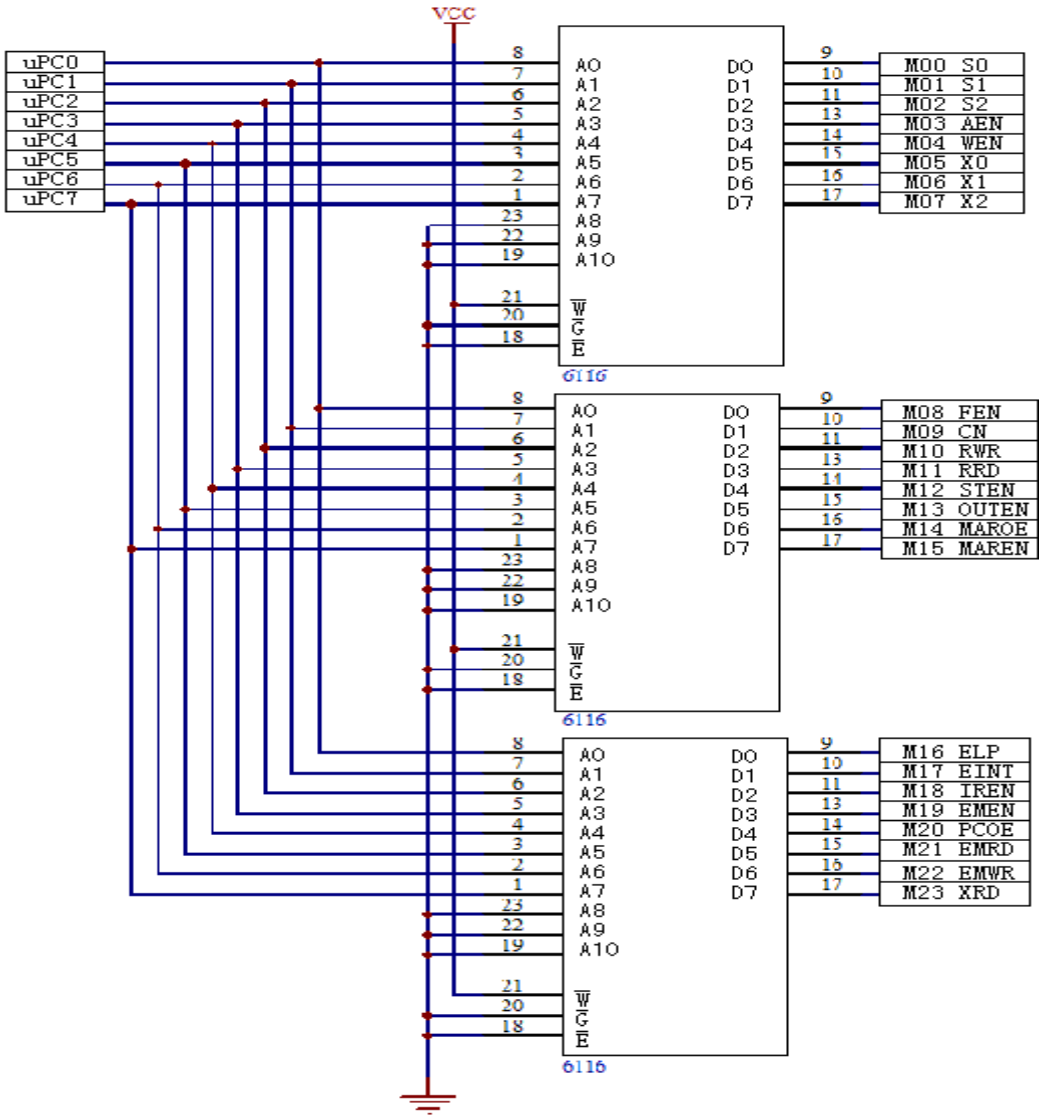
实验要求：利用 CPTH 实验仪上的开关做为控制信号，实现微程序存储器 uM 的输出功能。

实验目的：1、了解微程序控制方式模型机的基本工作原理。

2、了解微程序存储器 uM 的控制方法。

实验电路：

存储器 uM 由三片 6116RAM 构成，共 24 位微指令，采用水平型微指令格式。存储器的地址由 uPC 提供，片选及读信号恒为低，写信号恒为高. 存储器 uM 始终输出 uPC 指定地址单元的数据。



uM 原理图

连接线表

连接	信号孔	接入孔	作用	状态说明
1	J2座	J3座	将K23-K16接入DBUS[7:0]	实验模式：手动
2	IREN	K0	IR，uPC写使能	低电平有效
6	CK	已连	uPC工作脉冲	上升沿打入

实验 1：微程序存储器 uM 读出

置控制信号为：K0 为 1

uM 输出 uM[0]的数据

按一次 STEP 脉冲键，CK 产生一个上升沿，数据 uPC 被加一。
uM 输出 uM[1]的数据

按一次 STEP 脉冲键，CK 产生一个上升沿，数据 uPC 被加一。
uM 输出 uM[2]的数据

实验 2：使用实验仪小键盘输入 uM

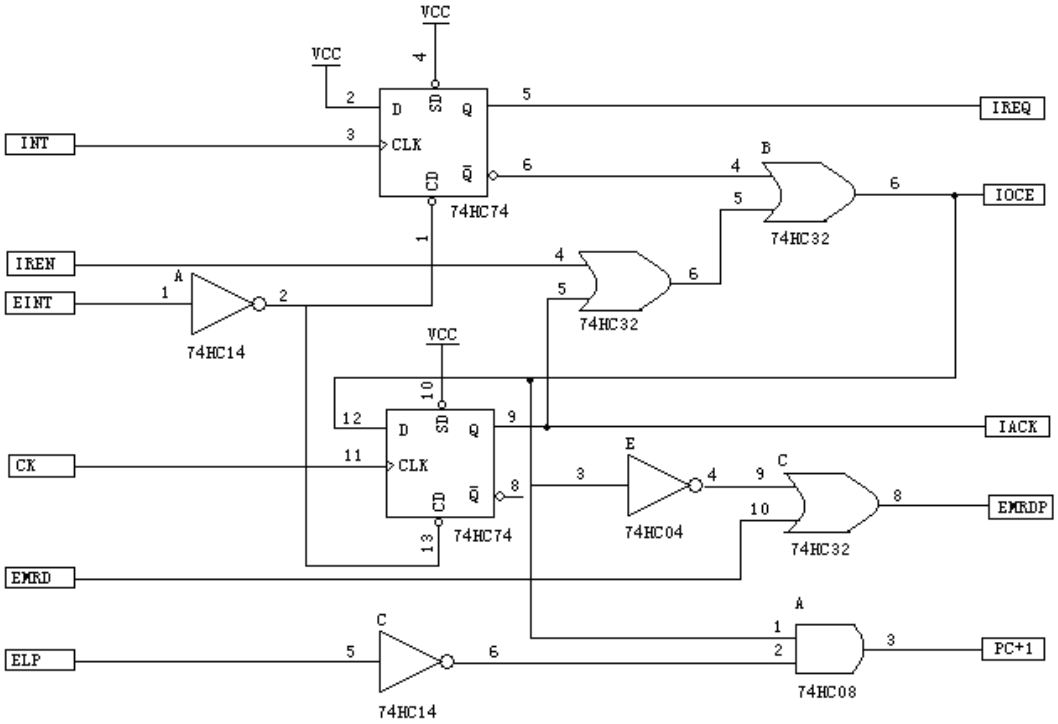
1．连接 J1, J2
2．打开电源
3．按 TV/ME 键, 选择 uM
4．输入两位地址, 00
5．按 NEXT, 进入微程序修改
6．按六位微程序数据
7．按 NEXT 选择下个地址/按 LAST 选择上个地址
8．重复 6,7 步输入微程序
9．按 RST 结束

2.8 中断实验

实验要求：利用 CPTH 实验仪上的开关做控制信号，实现中断功能。

实验目的：了解模型机的中断功能的工作原理及中断过程中，申请、响应、处理、返回各阶段时序。

实验电路：中断电路有两个 D 触发器，分别用于保存中断请求信号(IREQ)及中断响应信号(IACK)。INT 有上升沿时，IREQ 触发器被置为 1。当下一条指令取指时(IREN=0), 存贮器 EM 的读信号(EMRDP)被关闭，同时产生读中断指令(ICEN)信号, 程序的执行被打断转而去执行 B8 指令响应中断。在取 B8 的同时置 IACK 触发器被置为 1，禁止新的中断响应。EINT 信号置 0，IACK，IREQ 触发器为 0，中断电路可以响应新的中断。



中断控制器原理图

连接线表

连接	信号孔	接入孔	作用	状态说明（手动模式）
1	IREN	K0	IR,uPC写允许	低电平有效
2	EINT	K1	清中断寄存器	低电平有效
3	INT	已连	中断输入	上升沿有效
4	CK	已连	时钟输入	上升沿有效

置控制信号为：

K1(EINT)	K0(IREN)
1	0

短路块选择端 JINT 指向 RG 侧，按 RG 脉冲键，产生中断请求，此时黄色 REQ 指示灯亮，同时 B8 输出红色指示灯。

按 STEP 脉冲键，产生取指脉冲，黄色 ACK 指示灯亮。

置控制信号为：

K1(EINT)	K0(IREN)
0	1

REQ，ACK 灯灭。

第三章　　CPTH 模型机

3.1 模型机总体结构

CPTH 模型机包括了一个标准 CPU 所具备所有部件，这些部件包括：运算器 ALU、累加器 A、工作寄存器 W、左移门 L、直通门 D、右移门 R、寄存器组 R0-R3、程序计数器 PC、地址寄存器 MAR、堆栈寄存器 ST、中断向量寄存器 IA、输入端口 IN、输出端口寄存器 OUT、程序存储器 EM、指令寄存器 IR、微程序计数器 uPC、微程序存储器 uM，以及中断控制电路、跳转控制电路。其中运算器和中断控制电路以及跳转控制电路用 CPLD 来实现，其它电路都是用离散的数字电路组成。微程序控制部分也可以用组合逻辑控制来代替。

模型机为 8 位机，数据总线、地址总线都为 8 位，但其工作原理与 16 位机相同。相比而言 8 位机实验减少了烦琐的连线，但其原理却更容易被学生理解、吸收。

模型机的指令码为 8 位，根据指令类型的不同，可以有 0 到 2 个操作数。指令码的最低两位用来选择 R0-R3 寄存器，在微程序控制方式中，用指令码做为微地址来寻址微程序存储器，找到执行该指令的微程序。而在组合逻辑控制方式中，按时序用指令码产生相应的控制位。在本模型机中，一条指令最多分四个状态周期，一个状态周期为一个时钟脉冲，每个状态周期产生不同的控制逻辑，实现模型机的各种功能。模型机有 24 位控制位以控制寄存器的输入、输出，选择运算器的运算功能，存储器的读写。24 位控制位分别介绍如下：

XRD ：外部设备读信号，当给出了外设的地址后，输出此信号，从指定外设读数据。

EMWR：程序存储器 EM 写信号。

EMRD：程序存储器 EM 读信号。

PCOE：将程序计数器 PC 的值送到地址总线 ABUS 上。

EMEN：将程序存储器 EM 与数据总线 DBUS 接通，由 EMWR 和 EMRD 决定是将 DBUS 数据写到 EM 中，还是从 EM 读出数据送到 DBUS。

IREN：将程序存储器 EM 读出的数据打入指令寄存器 IR 和微指令计数器 uPC。

EINT：中断返回时清除中断响应和中断请求标志，便于下次中断。

ELP：PC 打入允许，与指令寄存器的 IR3、IR2 位结合，控制程序跳转。

MAREN：将数据总线 DBUS 上数据打入地址寄存器 MAR。

MAROE：将地址寄存器 MAR 的值送到地址总线 ABUS 上。

OUTEN：将数据总线 DBUS 上数据送到输出端口寄存器 OUT 里。

STEN：将数据总线 DBUS 上数据存入堆栈寄存器 ST 中。

RRD：读寄存器组 R0-R3，寄存器 R?的选择由指令的最低两位决定。

RWR：写寄存器组 R0-R3，寄存器 R?的选择由指令的最低两位决定。

CN：决定运算器是否带进位移位，CN=1 带进位，CN=0 不带进位。

FEN：将标志位存入 ALU 内部的标志寄存器。

X2：X2、X1、X0 三位组合来译码选择将数据送到 DBUS 上的寄存器。
X1：见 18 页表。
X0：
WEN：将数据总线 DBUS 的值打入工作寄存器 W 中。
AEN： 将数据总线 DBUS 的值打入累加器 A 中。
S2： S2、S1、S0 三位组合决定 ALU 做何种运算。
S1： 见 16 页表。
S0：

3.2 模型机寻址方式

模型机的寻址方式分五种：

累加器寻址：操作数为累加器 A，例如“CPL A”是将累加器 A 值取反，还有些指令是隐含寻址累加器 A，例如“OUT”是将累加器 A 的值输出到输出端口寄存器 OUT。

寄存器寻址：参与运算的数据在 R0-R3 的寄存器中，例如 “ADD A，R0”指令是将寄存器 R0 的值加上累加器 A 的值，再存入累加器 A 中。

寄存器间接寻址：参与运算的数据在存储器 EM 中，数据的地址在寄存器 R0-R3 中，如 “MOV A，@R1”指令是将寄存器 R1 的值做为地址，把存储器 EM 中该地址的内容送入累加器 A 中。

存储器直接寻址：参与运算的数据在存储器 EM 中，数据的地址为指令的操作数。例如“AND A，40H”指令是将存储器 EM 中 40H 单元的数据与累加器 A 的值做逻辑与运算，结果存入累加器 A。

立即数寻址：参与运算的数据为指令的操作数。例如 “SUB A，#10H”是从累加器 A 中减去立即数 10H，结果存入累加器 A。

3.3 模型机指令集

模型机的缺省的指令集分几大类： 算术运算指令、逻辑运算指令、移位指令、数据传输指令、跳转指令、中断返回指令、输入/输出指令。

助记符	机器码 1	机器码 2	注释
FATCH	000000xx		实验机占用，不可修改，复位后，所有寄存器清0(IR 除外)，首先执行 _FATCH_指令取指
	000001xx		未使用
	000010xx		未使用
	000011xx		未使用
ADD A,R?	000100xx		将寄存器 R?的值加入累加器 A 中
ADD A,@R?	000101xx		将间址存储器的值加入累加器 A 中
ADD A,MM	000110xx	MM	将存储器 MM 地址的值加入累加器 A 中
ADD A,#I I	000111xx	I I	将立即数 I I 加入累加器 A 中
ADDC A,R?	001000xx		将寄存器 R?的值加入累加器 A 中，带进位
ADDC A,@R?	001001xx		将间址存储器的值加入累加器 A 中，带进位
ADDC A,MM	001010xx	MM	将存储器 MM 地址的值加入累加器 A 中，带进位
ADDC A,#I I	001011xx	I I	将立即数 I I 加入累加器 A 中，带进位
SUB A,R?	001100xx		从累加器 A 中 减去寄存器 R?的值
SUB A,@R?	001101xx		从累加器 A 中 减去间址存储器的值
SUB A,MM	001110xx	MM	从累加器 A 中 减去存储器 MM 地址的值
SUB A,#I I	001111xx	I I	从累加器 A 中 减去立即数 I I 加入累加器 A 中
SUBC A,R?	010000xx		从累加器 A 中 减去寄存器 R?值，减进位
SUBC A,@R?	010010xx		从累加器 A 中 减去间址存储器的值，减进位
SUBC A,MM	010010xx	MM	从累加器 A 中 减去存储器 MM 地址的值，减进位
SUBC A,#I I	010011xx	I I	从累加器 A 中 减去立即数 I I，减进位
AND A,R?	010100xx		累加器 A“与”寄存器 R?的值
AND A,@R?	010101xx		累加器 A“与”间址存储器的值
AND A,MM	010110xx	MM	累加器 A“与”存储器 MM 地址的值
AND A,#I I	010111xx	I I	累加器 A“与”立即数 I I
OR A,R?	011000xx		累加器 A“或 ”寄存器 R?的值
OR A,@R?	011001xx		累加器 A“或”间址存储器的值
OR A,MM	011010xx	MM	累加器 A“或”存储器 MM 地址的值
OR A,#I I	011011x	II	累加器 A“或”立即数 I I
MOV A,R?	011100xx		将寄存器 R?的值送到累加器 A 中
MOV A,@R?	011101xx		将间址存储器的值送到累加器 A 中

MOV A,MM	011110xx	MM	将存储器 MM 地址的值送到累加器 A 中
MOV A,#I I	011111xx	I I	将立即数 I I 送到累加器 A 中
MOV R?,A	100000xx		将累加器 A 的值送到寄存器 R?中
MOV @R?,A	100001xx		将累加器 A 的值送到间址存储器中
MOV MM,A	100010xx	MM	将累加器 A 的值送到存储器 MM 地址中
MOV R?,#I I	100011xx	I I	将立即数 I I 送到寄存器 R?中
READ MM	100100xx	MM	从外部地址 MM 读入数据，存入累加器 A 中
WRITE MM	100101xx	MM	将累加器 A 中数据写到外部地址 MM 中
	100110xx		未使用
	100111xx		未使用
JC MM	101000xx	MM	若进位标志置 1，跳转到 MM 地址
JZ MM	101001xx	MM	若零标志位置 1，跳转到 MM 地址
	101010xx		未使用
JMP MM	101011xx	MM	跳转到 MM 地址
	101100xx		未使用
	101101xx		未使用
INT	101110xx		实验机占用，不可修改。进入中断时，实验机硬件产生_INT_指令
CALL MM	101111xx	MM	调用 MM 地址的子程序
IN	110000xx		从输入端口读入数据到累加器 A 中
OUT	110001xx		将累加器 A 中数据输出到输出端口
	110010xx		未使用
RET	110011xx		子程序返回
RR A	110100xx		累加器 A 右移
RL A	110101xx		累加器 A 左移
RRC A	110110xx		累加器 A 带进位右移
RLC A	110111xx		累加器 A 带进位左移
NOP	111000xx		空指令
CPL A	111001xx		累加器 A 取反，再存入累加器 A 中
	111010xx		未使用
RETI	111011xx		中断返回
	111100xx		未使用
	111101xx		未使用
	111110xx		未使用
	111111xx		未使用

3.4 模型机微指令集

指令/微指令表(insfile1.mic)

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
FATCH	T0	00	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		01	FFFFFF				A 输出		1	
		02	FFFFFF				A 输出		1	
		03	FFFFFF				A 输出		1	
UNDEF	T0	04	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		05	FFFFFF				A 输出		1	
		06	FFFFFF				A 输出		1	
		07	FFFFFF				A 输出		1	
UNDEF	T0	08	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		09	FFFFFF				A 输出		1	
		0A	FFFFFF				A 输出		1	
		0B	FFFFFF				A 输出		1	
UNDEF	T0	0C	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		0D	FFFFFF				A 输出		1	
		0E	FFFFFF				A 输出		1	
		0F	FFFFFF				A 输出		1	
ADD A, R?	T2	10	FFF7EF	寄存器值 R?	寄存器 W		A 输出		1	
	T1	11	FFFE90	ALU 直通	寄存器 A 标志位 C,Z		加运算		1	
	T0	12	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		13	FFFFFF				A 输出		1	
ADD A, @R?	T3	14	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
	T2	15	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	16	FFFE90	ALU 直通	寄存器 A 标志位 C,Z		加运算		1	
	T0	17	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1

ADD A,MM	T3	18	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
	T2	19	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	1A	FFFE90	ALU 直通	寄存器 A 标志位 C，Z		加运算		1	
	T0	1B	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
ADD A,#I I	T2	1C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出		1	1
	T1	1D	FFFE90	ALU 直通	寄存器 A 标志位 C，Z		加运算		1	
	T0	1E	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		1F	FFFFFF				A 输出		1	
助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
ADDC A,R?	T2	20	FFF7EF	寄存器值 R?	寄存器 W		A 输出		1	
	T1	21	FFFE94	ALU 直通	寄存器 A 标志位 C，Z		带进位加运算		1	
	T0	22	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		23	FFFFFF				A 输出		1	
ADDC A,@R?	T3	24	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
	T2	25	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	26	FFFE94	ALU 直通	寄存器 A 标志位 C，Z		带进位加运算		1	
	T0	27	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
ADDC A,MM	T3	28	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
	T2	29	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	2A	FFFE94	ALU 直通	寄存器 A 标志位 C，Z		带进位加运算		1	
	T0	2B	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
ADDC A,#I I	T2	2C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出		1	1
	T1	2D	FFFE94	ALU 直通	寄存器 A 标志位 C，Z		带进位加运算		1	
	T0	2E	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		2F	FFFFFF				A 输出		1	

SUB A,R?	T2	30	FFF7EF	寄存器值 R?	寄存器 W		A 输出		1	
	T1	31	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算		1	
	T0	32	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		33	FFFFFF				A 输出		1	
SUB A,@R?	T3	34	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
	T2	35	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	36	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算		1	
	T0	37	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
SUB A,MM	T3	38	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
	T2	39	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	3A	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算		1	
	T0	3B	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
SUB A,#I I	T2	3C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出		1	1
	T1	3D	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算		1	
	T0	3E	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		3F	FFFFFF				A 输出		1	
助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
SUBC A,R?	T2	40	FFF7EF	寄存器值 R?	寄存器 W		A 输出		1	
	T1	41	FFFE95	ALU 直通	寄存器 A 标志位 C,Z		带进位减运算		1	
	T0	42	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		43	FFFFFF				A 输出		1	
SUB A,@R?	T3	44	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
	T2	45	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	46	FFFE95	ALU 直通	寄存器 A 标志位 C,Z		带进位减运算		1	
	T0	47	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1

SUBC A,MM	T3	48	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
	T2	49	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	4A	FFFE95	ALU 直通	寄存器 A 标志位 C,Z		带进位减运算		1	
	T0	4B	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
SUBC A, # I I	T2	4C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出		1	1
	T1	4D	FFFE95	ALU 直通	寄存器 A 标志位 C,Z		带进位减运算		1	
	T0	4E	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		4F	FFFFFF				A 输出		1	
AND A, R?	T2	50	FFF7EF	寄存器值 R?	寄存器 W		A 输出		1	
	T1	51	FFFE93	ALU 直通	寄存器 A 标志位 C,Z		与运算		1	
	T0	52	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		53	FFFFFF				A 输出		1	
AND A, @R?	T3	54	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
	T2	55	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	56	FFFE93	ALU 直通	寄存器 A 标志位 C,Z		与运算		1	
	T0	57	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
AND A,MM	T3	58	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
	T2	59	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
	T1	5A	FFFE93	ALU 直通	寄存器 A 标志位 C,Z		与运算		1	
	T0	5B	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
AND A, # I I	T2	5C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出		1	1
	T1	5D	FFFE93	ALU 直通	寄存器 A 标志位 C,Z		与运算		1	
	T0	5E	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		5F	FFFFFF				A 输出		1	
助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC

OR	A, R?	T2	60	FFF7EF	寄存器值 R?	寄存器 W		A 输出		1	
		T1	61	FFFE92	ALU 直通	寄存器 A　标志位 C,Z		或运算		1	
		T0	62	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
			63	FFFFFF				A 输出		1	
OR	A, @R?	T3	64	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
		T2	65	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
		T1	66	FFFE92	ALU 直通	寄存器 A　标志位 C,Z		或运算		1	
		T0	67	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
OR	A, MM	T3	68	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
		T2	69	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出		1	
		T1	6A	FFFE92	ALU 直通	寄存器 A　标志位 C,Z		或运算		1	
		T0	6B	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
OR	A,# I I	T2	6C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出		1	1
		T1	6D	FFFE92	ALU 直通	寄存器 A　标志位 C,Z		或运算		1	
		T0	6E	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
			6F	FFFFFF				A 输出		1	
MOV	A, R?	T1	70	FFF7F7	寄存器值 R?	寄存器 A		A 输出		1	
		T0	71	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
			72	FFFFFF				A 输出		1	
			73	FFFFFF				A 输出		1	
MOV	A,@R?	T2	74	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
		T1	75	D7BFF7	存贮器值 EM	寄存器 A	MAR 输出	A 输出		1	
		T0	76	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
			77	FFFFFF				A 输出		1	
MOV	A, MM	T2	78	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1

	T1	79	D7BFF7	存贮器值 EM	寄存器 A	MAR 输出	A 输出		1	
	T0	7A	CBFFFF		指令寄存 IR	PC 输出	A 输出		写入	1
		7B	FFFFFF				A 输出		1	
MOV A,# I I	T1	7C	C7FFF7	存贮器值 EM	寄存器 A	PC 输出	A 输出		1	1
	T0	7D	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		7E	FFFFFF				A 输出		1	
		7F	FFFFFF				A 输出		1	
助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
MOV R?, A	T1	80	FFFB9F	ALU 直通	寄存器 R?		A 输出		1	
	T0	81	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		82	FFFFFF				A 输出		1	
		83	FFFFFF				A 输出		1	
MOV @R?, A	T2	84	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出		1	
	T1	85	B7BF9F	ALU 直通	存贮器 EM	MAR 输出	A 输出		1	
	T0	86	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		87	FFFFFF				A 输出		1	
MOV MM, A	T2	88	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
	T1	89	B7BF9F	ALU 直通	存贮器 EM	MAR 输出	A 输出		1	
	T0	8A	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		8B	FFFFFF				A 输出		1	
MOV R?,# I I	T1	8C	C7FBFF	存贮器值 EM	寄存器 R?	PC 输出	A 输出		1	1
	T0	8D	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		8E	FFFFFF				A 输出		1	
		8F	FFFFFF				A 输出		1	
READ A, MM	T2	90	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1

	T1	91	7FBFF7		寄存器 A	MAR 输出	A 输出		1	
	T0	92	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		93	FFFFFF				A 输出		1	
WRITE MM, A	T2	94	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出		1	1
	T1	95	FF9F9F	ALU 直通	用户 OUT	MAR 输出	A 输出		1	
	T0	96	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		97	FFFFFF				A 输出		1	
UNDEF	T0	98	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		99	FFFFFF				A 输出		1	
		9A	FFFFFF				A 输出		1	
		9B	FFFFFF				A 输出		1	
UNDEF	T0	9C	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		9D	FFFFFF				A 输出		1	
		9E	FFFFFF				A 输出		1	
		9F	FFFFFF				A 输出		1	
助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
JC MM	T1	A0	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出		1	写入
	T0	A1	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		A2	FFFFFF				A 输出		1	
		A3	FFFFFF				A 输出		1	
JZ MM	T1	A4	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出		1	写入
	T0	A5	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		A6	FFFFFF				A 输出		1	
		A7	FFFFFF				A 输出		1	

UNDEF	T0	A8	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		A9	FFFFFF				A 输出		1	
		AA	FFFFFF				A 输出		1	
		AB	FFFFFF				A 输出		1	
JMP MM	T1	AC	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出		1	写入
	T0	AD	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		AE	FFFFFF				A 输出		1	
		AF	FFFFFF				A 输出		1	
UNDEF	T0	B0	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		B1	FFFFFF				A 输出		1	
		B2	FFFFFF				A 输出		1	
		B3	FFFFFF				A 输出		1	
UNDEF	T0	B4	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		B5	FFFFFF				A 输出		1	
		B6	FFFFFF				A 输出		1	
		B7	FFFFFF				A 输出		1	
INT	T2	B8	FFEF7F	PC 值	堆栈寄存器 ST		A 输出		1	
	T1	B9	FEFF3F	中断地址 IA	寄存器 PC		A 输出		1	写入
	T0	BA	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		BB	FFFFFF				A 输出		1	
CALL MM	T3	BC	EF7F7F	PC 值	地址寄存器 MAR	PC 输出	A 输出		1	1
	T2	BD	FFEF7F	PC 值	堆栈寄存器 ST		A 输出		1	
	T1	BE	D6BFFF	存贮器值 EM	寄存器 PC	MAR 输出	A 输出		1	写入
	T0	BF	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
IN	T1	C0	FFFF17	用户 IN	寄存器 A		A 输出		1	
	T0	C1	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		C2	FFFFFF				A 输出		1	
		C3	FFFFFF				A 输出		1	
OUT	T1	C4	FFDF9F	ALU 直通	用户 OUT		A 输出		1	
	T0	C5	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		C6	FFFFFF				A 输出		1	
		C7	FFFFFF				A 输出		1	
UNDEF	T0	C8	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		C9	FFFFFF				A 输出		1	
		CA	FFFFFF				A 输出		1	
		CB	FFFFFF				A 输出		1	
RET	T1	CC	FEFF5F	堆栈寄存器 ST	寄存器 PC		A 输出		1	写入
	T0	CD	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		CE	FFFFFF				A 输出		1	
		CF	FFFFFF				A 输出		1	
RR A	T1	D0	FFFCB7	ALU 右移	寄存器 A 标志位 C,Z		A 输出	右移	1	
	T0	D1	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		D2	FFFFFF				A 输出		1	
		D3	FFFFFF				A 输出		1	
RL A	T1	D4	FFFCD7	ALU 左移	寄存器 A 标志位 C,Z		A 输出	左移	1	
	T0	D5	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		D6	FFFFFF				A 输出		1	

		D7	FFFFFF				A 输出		1	
RRC A	T1	D8	FFFEB7	ALU 右移	寄存器 A 标志位 C,Z		A 输出	带进位右移	1	
	T0	D9	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		DA	FFFFFF				A 输出		1	
		DB	FFFFFF				A 输出		1	
RLC A	T1	DC	FFFED7	ALU 左移	寄存器 A 标志位 C,Z		A 输出	带进位左移	1	
	T0	DD	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		DE	FFFFFF				A 输出		1	
		DF	FFFFFF				A 输出		1	
助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
NOP	T0	E0	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		E1	FFFFFF				A 输出		1	
		E2	FFFFFF				A 输出		1	
		E3	FFFFFF				A 输出		1	
CPL A	T1	E4	FFFE96	ALU 直通	寄存器 A 标志位 C,Z		A 取反		1	
	T0	E5	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		E6	FFFFFF				A 输出		1	
		E7	FFFFFF				A 输出		1	
UNDEF	T0	E8	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		E9	FFFFFF				A 输出		1	
		EA	FFFFFF				A 输出		1	
		EB	FFFFFF				A 输出		1	
RETI	T1	EC	FCFF5F	堆栈寄存器 ST	寄存器 PC		A 输出		1	写入
	T0	ED	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		EE	FFFFFF				A 输出		1	

		EF	FFFFFF				A 输出		1	
UNDEF	T0	F0	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		F1	FFFFFF				A 输出		1	
		F2	FFFFFF				A 输出		1	
		F3	FFFFFF				A 输出		1	
UNDEF	T1	F4	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		F5	FFFFFF				A 输出		1	
		F6	FFFFFF				A 输出		1	
		F7	FFFFFF				A 输出		1	
UNDEF	T0	F8	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		F9	FFFFFF				A 输出		1	
		FA	FFFFFF				A 输出		1	
		FB	FFFFFF				A 输出		1	
UNDEF	T0	FC	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
		FD	FFFFFF				A 输出		1	
		FE	FFFFFF				A 输出		1	
		FF	FFFFFF				A 输出		1	

第四章　　模型机综合实验（微程序控制器）

在综合实验中，模型机作为一个整体来工作的，所有微程序的控制信号由微程序存储器 uM 输出，而不是由开关输出。在做综合实验之前，先用 8 芯电缆连接 J1 和 J2，使系统处于非手动状态，这样实验仪的监控系统会自动打开 uM 的输出允许，微程序的各控制信号就会接到各寄存器、运算器的控制端口。此综合实验(1~7)使用的指令是模型机的缺省指令系统，系统自动默认装入缺省指令系统／非流水微指令系统文件：insfile1.mic。

在做综合实验时，可以用 CPTH 计算机组成原理实验软件输入、修改程序，汇编成机器码并下载到实验仪上，由软件控制程序实现单指令执行、单微指令执行、全速执行，并可以在软件上观察指令或微指令执行过程中数据的走向、各控制信号的状态、各寄存器的值。CPTH 软件的使用方法见第九章“CPTH 集成开发环境使用”。也可以用实验仪自带的小键盘和显示屏来输入、修改程序，用键盘控制单指令或单微指令执行，用 LED 或用显示屏观察各寄存器的值。实验仪上的键盘使用方法见第八章“实验仪键盘使用”。

在用微程序控制方式做综合实验时，在给实验仪通电前，拔掉实验仪上所有的手工连接的接线，再用 8 芯电缆连接 J1 和 J2，控制方式开关 KC 拨到“微程序控制”方向。若想用 CPTH 软件控制组成原理实验仪，就要启动软件，并用快捷图标的“连接通信口”功能打开设置窗口，选择实验仪连接的串行口，然后再点击“OK”按钮接通到实验仪。

实验 1：数据传送实验/输入输出实验

1．在 CPTH 软件中的源程序窗口输入下列程序

```
MOV   A, #12h
MOV   A, R0
MOV   a,@R0
MOV   A, 01H
IN
OUT
END
```

2．将程序另存为 EX1.ASM，将程序汇编成机器码，调试窗口会显示出程序地址、机器码、反汇编指令。

程序地址	机器码	反汇编指令	指令说明
00	7C 12	MOV A， #12	立即数12H送到累加器A
02	70	MOV A， R0	寄存器R0送到累加器A
03	74	MOV A， @R0	R0间址的存储器内容送到累加器A
04	78 01	MOV A, 01	存储器01单元内容送到累加器A
06	C0	IN	端口IN内容输入到累加器A
07	C4	OUT	累加器A内容输出到端口OUT

3．按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。（见 EX1.ASM 程序跟踪结果）

EX1.ASM 程序跟踪结果

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	uPC	PC
	T0	00	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
00 MOV A,#12	T1	7C	C7FFF7	存贮器值 EM	寄存器 A	PC 输出	A 输出	1	1
	T0	7D	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
02 MOV A,R0	T1	70	FFF7F7	寄存器值 R?	寄存器 A		A 输出	1	
	T0	71	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
03 MOV A,@R0	T2	74	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出	1	
	T1	75	D7BFF7	存贮器值 EM	寄存器 A	MAR 输出	A 输出	1	
	T0	76	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
04 MOV A,01	T2	78	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出	1	1
	T1	79	D7BFF7	存贮器值 EM	寄存器 A	MAR 输出	A 输出	1	
	T0	7A	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
06 IN	T1	C0	FFFF17	用户 IN	寄存器 A		A 输出	1	
	T0	C1	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
07 OUT	T1	C4	FFDF9F	ALU 直通	用户 OUT		A 输出	1	
	T0	C5	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1

每个程序的一开始的第一条微指令一定是取指令，此微指令的值为 0CBFFFFH，对应到各个控制位就是 EMRD、PCOE、及 IREN 为低，此三位有效，其它所有位都处于无效状态。在程序第一次运行时或复位后，uPC 和 PC 的值都为 0，PCOE 有效将 PC 值送到 ABUS，做为程序存储器 EM 的地址，EMRD 信号有效就是从程序存储器 EM 中读出程序指令，IREN 将读出的指令送到 IR 寄存器和 uPC，此微指令的作用就是：从程序存储器 EM 的 0 地址读出程序指令机器码 7CH，并存入 uPC 中做为微程序存储器 uM 的地址，从微程序存储器 uM 的 7CH 单元中读出微控制码 0C7FFF7H，同时 PC 加 1 为读下一条指令或数据做准备。

MOV A，#12： 本指令为两个状态周期。在 T1 状态时，上次读出的指令机器码为 7CH，存入 uPC 中做为微程序存储器 uM 的地址，读出微指令的值为 0C7FFF7H，对应到各个控制位就是 EMRDPCOE、EMEN 及 AEN 为低，处于有效状态，其它控制位为无效状态。由于上条微指令（取指操作）已将 PC 加 1，此时 PCOE 是将加 1 后的 PC 输出到 ABUS 做为程序存储器 EM 的地址，EMRD 就是从程序存储 EM 中读出数据，本指令中读出的数据应为 12H，EMEN 将读出的数据送到 DBUS 总线上，AEN 是将 DBUS 总线上的值存入累加器 A 中。同时 uPC 加 1 为执行下条微指令做准备，PC 加 1 为读取下一条指令做准备。每条指令的最后一条微指令一定是取指令操作，本指令的 T0 状态周期即为取指令，执行上一条微指令时 uPC 已经加 1，按照此 uPC 为地址从微程序存储器 uM 读出的微指令的值为 0CBFFFFH，参照第步的说明，此微指令从程序存储器 EM 中读取指令。

MOV A，R0： 本指令为两个状态周期。在 T1 状态时，由上条取指操作取出的指令机器码为 70H，存入 uPC 后做为微程序地址访问微程序存储器 uM 的 70H 单元，读出微指令的值为 0FFF7F7，各控制位的状态为 RRD、AEN 为低电平为有效状态，RRD 有效表示从寄存器组 R0-R3 中读数送到 DBUS 上，在上条取指令操作时，IREN 将取出的指令机器码 70H 送入 IR 寄存器，而 IR 寄存器的最低两位是用来选择寄存器 R?的，此时 IR 寄存器最低两位为 00，被读出的寄存器为 R0。AEN 有效表示将 DBUS 的数据写到累加器 A 中。同时 uPC 加 1，为执行下条微指令做准备。 本指令的 T0 状态也是取指令，完成的功能是取出下一条要执行的指令机器码，并存入 uPC 和 IR 寄存器中。

MOV A，@R0: 本指令为三个状态周期。在 T2 状态时，由上个取指操作读出的指令机器码为 74H，打入 uPC 后，从微程序存储器 74H 单元读出的微指令的值为 0FF77FFH，有效的控制位为 MAREN 和 RRD，RRD 有效表示从寄存器组 R0-R3 中读出数据送到 DBUS，MAREN 有效表示将数据从 DBUS 总线上打入地址寄存器 MAR。uPC 加 1 取出下条微指令执行。在 T1 状态时，由 uPC 做为微程序存储器址，从 uM 的 75H 单元中读出微指令的值为 0D7BFF7H，其中有效的控制位为 EMRD、EMEN、MAROE 和 AEN。MAROE 表示程序存储器 EM 的地址由地址寄存器 MAR 输出，EMRD 表示从程序存储器 EM 中读出数据，EMEN 表示读出的数据送到地址总线 DBUS 上，AEN 有效表示将数据总线 DBUS 上的值存入累加器 A 中。此状态下 uPC 要加 1，为取下条微指令做准备。本指令的 T0 状态执行的是取指操作。取指操作详细描述见程序开始部分的取指令的说明。

MOV A，01： 本指令为三个状态周期。在 T2 状态时，由上条取指操作取出的指令机器码为 78H，存入 uPC 和 IR 寄存器后做为微程序存储器 uM 的地址，读出微指令的值为 0C77FFFH，相应的有效控制位为 EMRD、PCOE、EMEN 和 MAREN，PCOE 有效表示将 PC 值做为程序存储器 EM 的地址，EMRD 表示从程序存储器中读出数据，在本指令中此数据值为 01H，EMEN 表示将读出的数据送到 DBUS 总线，MAREN 表示将 DBUS 总线上的数据打入地址寄存器 MAR。uPC 同时加 1，取出下条微指令准备执行。在 T1 状态时，由 uPC 做为微程序存储器地址，从 uM 的 79H 单元中读出微指令的值为 0D7BFF7H，可以参见上条指令的 T1 状态，此微指令的所完成的功能是，以 MAR 的值做为程序存储器的地址，读出数据并送到数据总线 DBUS，同时将此数据存入累加器 A 中。uPC 加 1 取出下条微指令准备执行。在 T0 状态，微指令执行取指令操作。

IN： 本指令分两个状态周期。在 T1 状态时，由上次取指操作取出的指令机器码为 0C0H，以此做为微地址从 uM 中取出的微指令值为 0FFFF17H，有效控制位为 AEN、X2X1X0=000，因为 X2、X1、X0 为低，被选中的寄存器为输入端口 IN，也就是说，输入端口 IN 上的数据被允许送到数据总线 DBUS 上，AEN 有效表示将此数据打入累加器 A 中。同时 uPC 加 1 取出下条微指令准备执行。在 T0 状态，微指令执行的是取指令操作，取出下条指令准备执行。

OUT： 本指令分两个状态周期。 在 T1 状态，由上次取出的指令机器码为 0C4H，以此为微地址从微程序存储器 uM 中读出的微指令为 0FFDF9FH，有效控制位为 OUTEN、X2X1X0=100(二进制)，S2S1S0=111(二进制)，S2S1S0=111 表示运算器做“ALU 直通”运算，也就是累加器不做任何运算，直接输出结果，而 X2X1X0=100 表示运算器的结果不移位直接输出到数据总线 DBUS，OUTEN 有效表示将数据总线上的数据打入输出端口寄存

器 OUT 内。uPC 加 1，取出下条微指令准备执行。在 T0 状态，微指令执行的是取指操作，取出下条将要执行的指令。

实验 2：数据运算实验（加/减/与/或）

1．在 CPTH 软件中的源程序窗口输入下列程序

```
ADDC    A, R1
SUB     A, @R1
AND     A, #55H
OR      A, 02H
END
```

2．将程序另存为 EX2.ASM，将程序汇编成机器码，调试窗口会显示出程序地址、机器码、反汇编指令。

程序地址	机器码	反汇编指令	指令说明
00	21	ADDC A, R1	累加器 A 的值加上寄存器 R1 加进位
01	35	SUB A, @R1	累加器 A 减去 R1 间址的存储器内容
02	5C 55	AND A, #55	累加器 A 逻辑与立即数 55H
04	68 02	OR A, 02	累加器 A 逻辑或存储器 02 单元的内容

3．按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。(见“EX2.ASM 程序跟踪结果”详细介绍)

4．在了解数据运算的原理，可以加上一些数据传输指令给累加器 A 或寄存器 R?赋值，再运算，并观察运算结果。

EX2.ASM 程序跟踪结果

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	uPC	PC
	T0	00	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
00 ADDC A, R1	T2	20	FFF7EF	寄存器值 R?	寄存器 W		A 输出	1	
	T1	21	FFFE94	ALU 直通	寄存器 A 标志位 C,Z		带进位加运算	1	
	T0	22	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
01 SUB A,@R1	T3	34	FF77FF	寄存器值 R?	地址寄存器 MAR		A 输出	1	
	T2	35	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出	1	
	T1	36	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算	1	
	T0	37	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
02 AND A,#55	T2	5C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出	1	1
	T1	5D	FFFE93	ALU 直通	寄存器 A 标志位 C,Z		与运算	1	
	T0	5E	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
04 OR A,02	T3	68	C77FFF	存贮器值 EM	地址寄存器 MAR	PC 输出	A 输出	1	1
	T2	69	D7BFEF	存贮器值 EM	寄存器 W	MAR 输出	A 输出	1	
	T1	6A	FFFE92	ALU 直通	寄存器 A 标志位 C,Z		或运算	1	
	T0	6B	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1

程序的开始执行一条取指的微指令，读入程序第一条指令。

ADDC A，R1：本指令为三个状态周期。在 T2 状态，由上次取指操作取出的指令码为 21H，由 IREN 存入指令寄存器 IR，最低两位为 01(二进制)，选择寄存器 R1，指令码由于 IREN 打入 uPC 时，忽略掉指令的最低两位，而将 uPC 的最低两位置成 00，uPC 的值为 20H，访问微程序存储器的 20H 单元，读出微指令值为 0FFF7EFH，有效位为 RRD 及 WEN，就是将 R1 内容送到工作寄存器 W，uPC 加 1 取出下条微指令在 T1 状态，读出的微指令值为 0FFFE94H，有效位为 FEN 和 AEN，FEN 完成的操作是将标志位存入标志寄存器 F(ALU 内部)，X2X1X0 选择“ALU 直通”到数据总线 DBUS，S2S1S0 选择的运算操作为“带进位的加法运算”，AEN 将 DBUS 上的数据存入累加器 A。在 T0 状态，取出下条将要执行的指令。

SUB A，@R1：本指令有四个状态周期。在 T3 状态，上次取出的指令码为 35H，最低两位用于寻址 R1 寄存器，uPC 的最低两位置 0，来访问 uM 的 34H 单元的微指令，读出值为 0FF77FFH，将 R1 的值存入 MAR。在 T2 状态，微指令为 0D7BFEFH，表示用 MAR 做为地址从 EM 中读出数据送到 DBUS 再存到 W 中。在 T1 状态微指令为 0FFFE91H，表示 ALU 做“减运算”，其结果直通到 DBUS，再存入中，同时保存标志位。T0 状态为取指操作。

AND A，#55： 本指令为三个状态周期。在 T2 状态，微指令值为 0C7FFEFH，表示以 PC 做为地址，从 EM 中读出数据送到 DBUS，再将 DBUS 数据存 W 中。在 T1 状态，微指令为 0FFFE93H，表示 A 和 W 做“逻辑与”运算，结果直通到 DBUS，再存入 A 中，并保存标志位。

OR A，02：本指令有四个状态周期。在 T3 状态，微指令为 0C77FFFH，表示以 PC 做为地址，从 EM 中读出数据送到 DBUS，并存 MAR 中。在 T2 状态，微指令为 0D7BFEFH，

表示以 MAR 做为地址，从 EM 中读出数据送到 DBUS，并存入 W 中。在 T1 状态微指令为 0FFFE92H，表示 A 和 W 做“逻辑或”运算，结果“直通”到 DBUS 并存入 A 中。T0 状态为取指操作。

实验 3：移位/取反实验

1．在 CPTH 软件中的源程序窗口输入下列程序

```
MOV  A, #55H
RR   A
RLC  A
CPL  A
END
```

2．将程序另存为 EX3.ASM，将程序汇编成机器码，调试窗口会显示出程序地址、机器码、反汇编指令。

程序地址	机器码	反汇编指令	指令说明
00	7C 55	MOV A, #55	立即数 55H 存入累加器 A
02	D0	RR A	不带进位右移累加器 A
03	DC	RLC A	带进位左移累加器 A
04	E4	CPL A	累加器 A 内容取反

3．按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。(见“EX3.ASM 程序跟踪结果”详细介绍)

EX3.ASM 程序跟踪结果

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
	T0	00	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
00 MOV A,#55	T1	7C	C7FFF7	存贮器值 EM	寄存器 A	PC 输出	A 输出		1	1
	T0	7D	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
02 RR A	T1	D0	FFFCB7	ALU 右移	寄存器 A 标志位 C,Z		A 输出	右移	1	
	T0	D1	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
03 RLC A	T1	DC	FFFED7	ALU 左移	寄存器 A 标志位 C,Z		A 输出	带进位左移	1	
	T0	DD	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1
04 CPL A	T1	E4	FFFE96	ALU 直通	寄存器 A 标志位 C,Z		A 输出		1	
	T0	E5	CBFFFF		指令寄存器 IR	PC 输出	A 输出		写入	1

程序的开始执行一条取指的微指令，读入程序第一条指令。

MOV A, #55: 将累加器的值设为 055H，以便下面观察。

RR A：本指令为两个状态周期。在 T1 状态，由上次取指操作取出的指令码为 D0H，

访问微程序存储器的 20H 单元，读出微指令值为 0FFFCB7H，有效位为 CN、FEN 及 AEN，表示不带进位移位，运算器控制 S2S1S0=111(二进制)表示运算不运算，输出结果就为 A 的值，X2X1X0=101(二进制)表示，运算器“右移”输出到总线，FEN 将标志位保存，AEN 将 DBUS 内容存入 A 中，uPC 加 1 取出下条微令。在 T0 状态，取出下条将要执行的指令。

RLC A：本指令有两个状态周期。在 T1 状态微指令为 0FFFED7H，CN=1 表示带进位移位，S2S1S0=111 表示 ALU 不做运算，直接输出 A 内容，X2X1X0=110(二进制)表示，运算器“左移”输出到 DBUS，AEN 表示 DBUS 内容存入 A 中，FEN 表示保存标志位。T0 状态为取指操作。取出下条将要执行的指令。

CPL A： 本指令为两个状态周期。在 T1 状态，微指令为 0FFFE96H，S2S1S0=110 表示 ALU 做“取反”运算，X2X1X0=100(二进制)表示，运算器结果直通到 DBUS，再存入 A 中，并保存标志位。T0 状态为取指操作。取出下条将要执行的指令。

实验 4：转移实验

1．在 CPTH 软件中的源程序窗口输入下列程序

```
LO: MOV   A,#01
LOOP:
    SUB   A,#01
    JC    LOOP
    JZ    LOOP
    JMP   LO
    END
```

2．将程序另存为 EX4.ASM，将程序汇编成机器码，调试窗口会显示出程序地址、机器码、反汇编指令。

程序地址	机器码	反汇编指令	指令说明
00	7C 01	MOV A,#01	立即数 01H 存入累加器 A
02	3C 01	SUB A,#01	累加器 A 减 1
04	A0 02	JC 02	若有进位跳到程序 02 地址
06	A4 02	JZ 02	若 A=0 跳转到程序 02 地址
08	AC 00	JMP 00	无条件跳转到程序开始

3．按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。观察在条件满足和不满足的情况下，条件跳转是否正确执行。(见“EX4.ASM 程序跟踪结果”详细介绍)

EX4.ASM 程序跟踪结果

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	uPC	PC
	T0	00	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
00 MOV A, #01	T1	7C	C7FFF7	存贮器值 EM	寄存器 A	PC 输出	A 输出	1	1
	T0	7D	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
02 SUB A, #01	T2	3C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出	1	1
	T1	3D	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算	1	
	T0	3E	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
04 JC 02	T1	A0	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出	1	写入
	T0	A1	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
06 JZ 02	T1	A4	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出	1	写入
	T0	A5	CBFFFF		指令寄存器 IR	MAR 输出	A 输出	写入	1
02 SUB A, #01	T2	3C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出	1	1
	T1	3D	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算	1	
	T0	3E	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
04 JC 02	T1	A0	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出	1	写入
	T0	A1	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
02 SUB A, #01	T2	3C	C7FFEF	存贮器值 EM	寄存器 W	PC 输出	A 输出	1	1
	T1	3D	FFFE91	ALU 直通	寄存器 A 标志位 C,Z		减运算	1	
	T0	3E	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
04 JC 02	T1	A0	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出	1	写入
	T0	A1	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
06 JZ 02	T1	A4	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出	1	写入
	T0	A5	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
08 JMP 00	T1	AC	C6FFFF	存贮器值 EM	寄存器 PC	PC 输出	A 输出	1	写入
	T0	AD	CBFFFF		指令寄存器 IR	PC 输出	A 输出	写入	1
00 MOV A, #01	T1	7C	C7FFF7	存贮器值 EM	寄存器 A	PC 输出	A 输出	1	1

程序的开始执行一条取指的微指令，读入程序第一条指令。

MOV A, #01: 将累加器的值设为 01H，用于下面计算来产生进位标志和零标志。

SUB A，#01：A 值原为 1，将 A 值第一次减 1 后，应产生“零标志”位。

JC 02： 由上条取指读出的指令码为 0A0H，存入 IR 寄存器后，IR3、IR2 的值为 00(二进制)，表示判进位跳转功能，指令码存入 uPC 后，从 uM 读出的微指令值为 0C6FFFFH，表示以 PC 为地址从 EM 中读出数据 02H 并送到 DBUS，ELP 为低成有效状态，与 IR3、IR2 组成进位跳转控制，此时若有进位，就会产生一个控制信号，将总线 DBUS 上的值 02H 打入 PC，下条微指令取指时，就会从 EM 新的地址 02 中读指令码；此时若无进位，DBUS 上的值被忽略，PC 加 1，下条取指操作按新 PC 取出指令码执行。当前无进 位标志，顺序执行下条指令。

JZ 02： 由上条取指读出的指令码为 0A4H，存入 IR 寄存器后，IR3、IR2 的值为 01(二进制)，表示判零跳转功能，指令码存入 uPC 后，从 uM 读出的微指令值为 0C6FFFFH，表示以 PC 为地址从 EM 中读出数据 02H 并送到 DBUS，ELP 为低成有效状态，与 IR3、IR2 组成零跳转控制，与上条指令相比，尽管微指令相同，由于指令码不同，上一个为判进位跳转，这个为判零跳转。此时若零标志位为 1，即 A=0 时，就会产生一个控制信号，将总线 DBUS 上的值 02H 打入 PC，下条微指令取指时，就会从 EM 新的地址 02 中读指令码；此时若零标志位为 0，DBUS 上的值被忽略，PC 加 1，下条取指操作按新 PC 取出指令码执行。由于 A=0，零标志位为 1，产生 PC 打入信号，将 DBUS 上的值 02H 打入 PC。下一条取指操作，PC=02，以 PC 为地址从 EM 的 02 单元取出指令码执行，程序转到 02 地址。

SUB A，#01：A 值现为 0，再减 1 后，A=0FFH，并产生“进位标志”位。

JC 02： 此为判进位跳转指令，此时由于进位标志为 1，与 ELP、IR3、IR2 组成的电路产生 PC 打入信号，将数据总线上的值存入 PC，程序跳转到 02H 地址执行。

SUB A，#01：A 值现为 0FFH，再减 1 后，A=0FEH，无“零标志”，无“进位标志”位。

JC 02： 此为判进位跳转指令，此时无进位标志，程序顺序执行下条指令。

JZ 02： 此为判零跳转指令，此时无零标志位，程序顺序执行下条指令。

JMP 00： 由上条取指操作读出的指令码为 0ACH，存入 IR 寄存器后，IR3、IR2 的值为 11(二进制)，此为无条件跳转控制，指令码存入 uPC 后，从 uM 读出的微指令为 0C6FFFFH，表示以 PC 为地址从 EM 中读出数据并送到数据总线 DBUS 上，因为 ELP 有效，与 IR3、IR2 组合产生 PC 的打入信号，将 DBUS 上的数据存入 PC 中，下一条取指微指令按新的 PC 值读出程序的指令码。MOV A, #01: 程序从开头重新执行。

实验 5：调用实验

1．在 CPTH 软件中的源程序窗口,输入下列程序

```
    MOV  A,#00H
LOOP:
    CALL INCA
    JMP  LOOP
INCA:
    ADD  A,#01
    RET
    END
```

2．将程序另存为 EX5.ASM，将程序汇编成机器码，调试窗口会显示出程序地址、机器码、反汇编指令。

程序地址	机器码	反汇编指令	指令说明
00	7C 55	MOV A,#00	立即数00H存入累加器A
02	BC 06	CALL 06	调用子程序
04	AC 02	JMP 02	跳转到02地址,循环执行
06	1C 01	ADD A, #01	累加器A加1
08	CC	RET	子程序返回

3．按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。观察在调用子程序和从子程序返回时，堆栈的工作情况。(见“EX5.ASM 程序跟踪结果”详细介绍)

EX5.ASM 程序跟踪结果

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	uPC	PC
	T0	00	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
00 MOV A,#00	T1	7C	C7FFF7	存贮器值EM	寄存器A	PC输出	A输出	1	1
	T0	7D	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
02 CALL 06	T3	BC	EF7F7F	PC值	地址寄存器MAR	PC输出	A输出	1	1
	T2	BD	FFEF7F	PC值	堆栈寄存器ST		A输出	1	
	T1	BE	D6BFFF	存贮器值EM	寄存器PC	MAR输出	A输出	1	写入
	T0	BF	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
06 ADD A,#01	T2	1C	C7FFEF	存贮器值EM	寄存器W	PC输出	A输出	1	1
	T1	1D	FFFE90	ALU直通	寄存器A 标志位C,Z		加运算	1	
	T0	1E	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
08 RET	T1	CC	FEFF5F	堆栈寄存器ST	寄存器PC		A输出	1	写入
	T0	CD	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
04 JMP 02	T1	AC	C6FFFF	存贮器值EM	寄存器PC	PC输出	A输出	1	写入
	T0	AD	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
02 CALL 06	T3	BC	EF7F7F	PC值	地址寄存器MAR	PC输出	A输出	1	1

程序的开始执行一条取指的微指令，读入程序第一条指令。

MOV A, #00: 将累加器的值设为 00H，以便下面观察 A 加 1 后的结果。

CALL 06：本指令有四个状态周期。在 T3 状态，根据指令码为 0BCH，读出微指令值 0FF7F7FH，有效位为 PCOE、MAREN，X2X1X0 的值为 011(二进制)，PCOE 有效是将 PC 加 1，以便在下步将 PC 压栈时，存入堆栈的是程序下一条指令的地址，MAREN 有效及 X2X1X0 的值表示从 PC 中读出值并送到 MAR 中。在 T2 状态，读出微指令为 0FFEF7FH，有效位 STEN，X2X1X0=100(二进制)，表示从 PC 中读数据并存入堆栈寄存器 ST 中。在 T1 状态，微指令值为 0D6BFFFH，表示以 MAR 为地址从 EM 中读出数据，此数据就是子程序的地址，此时堆栈中保存的是调用子程序下条指令的地址。将此数据送到 DBUS，再存入 PC 中，实现程序跳转。在 T0 状态，按新的 PC 值，取出下条将要执行的指令。

ADD A，01：本指令将累加器加 1。

RET： 本指令有两个状态周期。在 T1 状态，上条取指操作读出的指令码为 0CCH，存入 IR 后，IR3、IR2 的值为 11(二进制)，取出的微指令的值为 0FEFF5FH，有效位为 ELP，X2X1X0=010(二进制)表示从 ST 中输出数据到总线，ELP 有效与 IR3、IR2=11 表示无条件将数据总线 DBUS 的数据打入 PC，实现子程序返回功能。在 T0 状态，按新 PC 取出指令，准备执行。

JMP 02： 程序无条件跳转到 02 地址，执行程序。

实验 6：中断实验

1．在 CPTH 软件中的源

程序窗口输入下列程序

```
  MOV  A,#00
LOOP:
  JMP  LOOP

  ORG  0E0H
  ADD  A,#01
  RETI
  END
```

2.将程序另存为 EX6.ASM，将程序汇编成机器码，反汇编窗口会显示出程序地址、机器码、反汇编指令。

程序地址	机器码	反汇编指令	指令说明
00	7C 00	MOV A,#00	立即数00H存入累加器A
02	AC 02	JMP 02	原地跳转等待中断
…	…	…	…
E0	1C 01	ADD A,#01	累加器A值加1
E2	EC	RETI	中断返回

3．在 IA 单元模块中，将拨码开关设置为“11100000”，短路块选择端 JINT 指向 RG 侧，按快捷图标的 F7，执行“单微指令运行”功能，在跟踪程序时，按下实验仪上中断请求按键[RG]，中断请求灯亮，在每个指令的最后一条微指令执行完，就会响应中断，中断响应灯高。同时，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。观察程序执行时，堆栈及中断请求，中断响应位的状态。 （见“EX6.ASM 程序跟踪结果”详细介绍）

EX6.ASM 程序跟踪结果

助记符	状态	微地址	微程序	数据输出	数据输入	地址输出	运算器	uPC	PC
	T0	00	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
00 MOV A,#00	T1	7C	C7FFF7	存贮器值EM	寄存器A	PC输出	A输出	1	1
	T0	7D	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
02 NOP	T0	E0	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
03 JMP 02	T1	AC	C6FFFF	存贮器值EM	寄存器PC	PC输出	A输出	1	写入
	T0	AD	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
INT	T2	B8	FFEF7F	PC值	堆栈寄存器ST		A输出	1	
	T1	B9	FEFF3F	中断地址IA	寄存器PC		A输出	1	写入
	T0	BA	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
E0 ADD A,#01	T2	1C	C7FFEF	存贮器值EM	寄存器W	PC输出	A输出	1	1
	T1	1D	FFFE90	ALU直通	寄存器A 标志位C,Z		加运算	1	
	T0	1E	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
E2 RETI	T1	EC	FCFF5F	堆栈寄存器ST	寄存器PC		A输出	1	写入
	T0	ED	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1

程序的开始执行一条取指的微指令，读入程序第一条指令。

MOV A, #00: 将累加器的值设为 00H，以便下面观察 A 加 1 后的结果。

NOP ：程序空操作，等待中断请求。

JMP 02：程序无条件跳转到 02 地址，执行程序。在执行此指令前，按下实验仪上的中断请求钮，中断请求的灯会亮，表示有中断请求。在本指令的 T0 状态即取指状态，IREN 有效将中断处理微程序地址 0B8H，送到指令总线 IBUS 上。

INT ：本指令为中断处理微程序，有三个状态周期。在 T2 状态，微指令的值为 0FFEF7FH，其中 X2X1X0=011(二进制)，表示从 PC 中读出数据送到 DBUS 上，STEN 有效表示将 DBUS 上数据存入堆栈寄存器 ST 中，这条微指令执行的就是将 PC 值(即下条将执行的指令的地址）存入堆栈。在 T1 状态，微指令值为 0FEFF3FH，其中 X2X1X0=001 表示将中断地址寄存器 IA 的值送到 DBUS 上，IA 的缺省值为

0E0H，ELP 有效，与指令寄存器 IR 的 IR3、IR2=10(二进制)组合，将 DBUS 值存入 PC，实现程序的跳转。在 T0 状态以中断地址 0E0H 为地址取出中断服务程序的第一条指令，准备执行。

ADD A，01：本指令将累加器加 1。

RETI：本指令有两个状态周期。在 T1 状态，取出的微指令为 0FCFF5FH，X2X1X0=010（二进制）表示从 ST 读出数据送到 DBUS 上，EINT 有效清除中断请求标志和中断响应标志，以便程序返回后，可以再次响应中断，ELP 有效与 IR3、IR2=11 表示无条件将数据总线 DBUS 的值打入 PC，实现中断服务程序返回功能。在 T0 状态，按新 PC 取出指令，准备执行。

NOP ： 上次中断是在执行完“JMP 02”指令后响应的，中断返回的地址为其下条将要执行的指令，也就是“NOP”指令。

实验 7：指令流水实验

指令流水操作，就是在微指令执行的过程中，在 T1 状态，如果 ABUS 和 IBUS 空闲，则可以利用这个空闲来进行预取指令，让 ABUS、IBUS 和 DBUS 并行工作，实现指令的流水工作。我们已经建立了一套可流水操作的指令/微指令系统。用户可调入这个指令/微指令系统进行实验。为了方便比较，我们仍用实验 1 的程序 EX1.ASM，其它指令用户可以自己做实验来比较、验证。

1．在 CPTH 软件中，用菜单的[文件|调入指令系统/微程序]功能，打开 CPTH 下的“INSFILE2.MIC”，这就是流水操作的指令/微指令系统。

2．在 CPTH 软件中，用菜单的[文件|打开文件]功能，打开 CPTH 下的“EX1.ASM”源程序。编译后产生的机器码与实验 1 相同。

3．按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。特别是在每条指令的 T0 状态周期，取指操作是否和其它指令并行执行。（见“EX1.ASM 程序流水操作跟踪结果”详细介绍）

EX1.ASM 程序流水操作跟踪结果

助记符	状态	微地址	微程序	数据输出	数据输入	地址输出	运算器	uPC	PC
	T0	00	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
00 MOV A, #12	T1	7C	C7FFF7	存贮器值EM	寄存器A	PC输出	A输出	1	1
	T0	7D	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
02 MOV A, R0	T0	70	CBF7F7	寄存器值R?	寄存器A、指令寄存器IR	PC输出	A输出	写入	1
03 MOV A,@R0	T2	74	FF77FF	寄存器值R?	地址寄存器MAR		A输出	1	
	T1	75	D7BFF7	存贮器值EM	寄存器A	MAR输出	A输出	1	
	T0	76	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
04 MOV A, 01	T2	78	C77FFF	存贮器值EM	地址寄存器MAR	PC输出	A输出	1	1
	T1	79	D7BFF7	存贮器值EM	寄存器A	MAR输出	A输出	1	
	T0	7A	CBFFFF		指令寄存器IR	PC输出	A输出	写入	1
06 IN	T0	C0	CBFF17	用户IN	寄存器A、指令寄存器IR	PC输出	A输出	写入	1
07 OUT	T0	C4	CBDF9F	ALU直通	指令寄存器IR、OUT	PC输出	A输出	写入	1

每个程序的一开始的第一条微指令一定是取指令，取出下条将要执行的指令。

MOV A，#12： 本指令为两个状态周期。在 T1 状态时，从程序存储器 EM 中读出数据送到累加器 A，ABUS 被占用，所以预指操作不能与数据总线 DBUS 上的操作并行执行。本指令的 T0 状态为正常的取指令操作。

MOV A，R0： 由于预指操作与数据总线可并行工作，本指令只有 1 个状态周期。由上条取指操作取出的指令机器码为 70H，存入 uPC 后做为微程序地址访问微程序存储器

uM 的 70H 单元，读出微指令的值为 0CBF7F7H，有效控制位为 EMRD、PCOE、IREN、RRD、AEN，由于 IR1、IR0 的值为 00，与 RRD 信号组合表示从 R0 中读出数据到 DBUS 总线，AEN 将 DBUS 上的值存入累加器 A，EMRDPCOE 和 IREN 有效表示以 PC 做为地址从 EM 中读出下条指令，并存入 IR 和 uPC 中，PC 加 1。

MOV A，@R0: 本指令为三个状态周期。在 T2 状态时，将 R0 的值存入地址寄存器 MAR。在 T1 状态时，以 MAR 为地址读出数据并送到累加器 A 中。在 T0 状态，取出下条将要执行指令。由于 ABUS 不空闲，所以取指操作不能并行工作。

MOV A，01： 本指令为三个状态周期。在 T2 状态时，以 PC 为地址从 EM 中读出数据存到 MAR 中，在 T1 状态，以 MAR 为地址从 EM 中读出数据存入累加器 A。T0 为取指操作。由于 ABUS 不空闲，取指操作不能并行执行。

IN： 本指令为 1 个状态周期。取指操作和输出操作可并行执行。由上次取指操作取出的指令机器码为 0C0H，以此做为微地址从 uM 中取出的微指令值为 0CBFF17H，有效控制位为 EMRD、PCOE、IREN 和 AEN、X2X1X0=000(二进制)表示从输入寄存器 IN 读数据送到 DBUS，AEN 表示将此数据存入 A，EMRD、PCOE 和 IREN 有效表示以 PC 为地址从 EM 中读出指令存入 IR 和 uPC 中，PC 加 1。

OUT： 本指令有 1 个状态周期。取指操作和输出操作并行完成。 由上次取出微指令值为为 0CBDF9FH，有效控制位为 EMRD、PCOE、IREN、OUTEN、X2X1X0=100(二进制)，S2S1S0=111(二进制)，S2S1S0=111 表示运算器做“ALU 直通”运算，也就是累加器不做任何运算，直接输出结果，而 X2X1X0=100 表示运算器的结果不移位直接输出到数据总线 DBUS，OUTEN 有效表示将数据总线上的数据打入输出端口寄存器 OUT 内。与此同时，EMRD、PCOE、IREN 表示以 PC 为地直址从 EM 中读出下条指令，存 IR 和 uPC 中，PC 加 1。

实验 8　　RISC 模型机

RISC 处理器设计的一般原则：

1． 只选用使用频度高的指令，减小指令系统，使每一条指令能尽快的执行

2． 减少寻址方式，并让指令具有相同的长度

3． 让大部分指令在一个时钟完成

4． 所有指令只有存（ST）、取（LD）指令可访问内存，它他指令均在寄存器间进行运算

下面我们给出一个 RISC 的指令系统

助记符	机器码1	机器码2	注释
FATCH	000000xx		实验机占用，不可修改。复位后，所有寄存器清0，首先执行 _FATCH_ 指令取指
ADDW A	000001xx		将寄存器W加入累加器A中
ADDCW A	000010xx		将寄存器W值加入累加器A中，带进位
SUBW A	000011xx		从累加器A中减去寄存器W值
SUBCW A	000100xx		从累加器A中减去寄存器W值，带进位
ANDW A	000101xx		累加器A“与”寄存器W值
ORW A	000110xx		累加器A“或”寄存器W值
LDW A	000111xx		将累加器A的值送到寄存器W中
LDW R?	001000xx		将寄存器R?的值送到寄存器W中
LDA R?	001001xx		将寄存器R?送到累加器A中
STA R?	001010xx		将累加器A的值送到寄存器R?中
RR A	001011xx		累加器A右移
RL A	001100xx		累加器A左移
RRC A	001101xx		累加器A带进位右移
RLC A	001110xx		累加器A带进位左移
CPL A	001111xx		累加器A取反，再存入累加器A中
LD A,#II	010000xx	II	将立即数送到累加器A中
LD A,MM	010001xx	MM	将存储器中MM中的值送到累加器A中
ST A,MM	010010xx	MM	将累加器A的值送到存储器MM地址中
JMP MM	010011xx		跳转到地址MM
JC MM	010100xx		若进位标志置1，跳转到地址MM
JZ MM	010101xx		若零标志位置1，跳转到地址MM

可以看出在这个指令系统中，只有访问主存 LD，ST 指令和转移指令有两个字节，其余指令均为单字节单时钟指令。

1．在 CPTH 软件中，用菜单的[文件|调入指令系统/微程序]功能，打开 CPTH 下的“RISCFILE.MIC”，这就是 RISC 指令/微指令系统。

2．在 CPTH 软件中，用菜单的[文件|打开文件]功能，打开 CPTH 下的“EX7.ASM”源程序。

3．按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，寄存器的输入/输出状态，各控制信号的状态，PC 及 uPC 如何工作。

4．比较非 RISC 指令系统，可以看出 RISC 指令系统简单很多。

第五章　组合逻辑控制

5.1　组合逻辑控制器

微程序控制器由微程序给出 24 位控制信号，而微程序的地址又是由指令码提供的，这就是说 24 位控制信号是由指令码确定的。如：MOV A，12H 及 MOV A，#34H 指令的微程序如下：

助记符	状态	微地址	微程序	相关控制位
MOV A,MM	T2	78	C77FFF	EMRD,PCOE,EMEN,MAREN
	T1	79	D7BFF7	EMRD,EMEN,MAROE,AEN
	T0	7A	CBFFFF	EMRD,PCOE,IREN
MOV A,#I	T1	7C	C7FFF7	EMRD,PCOE,EMEN,AEN
	T0	7D	CBFFFF	EMRD,PCOE,IREN

我们用组合逻辑的方法来写出相应的控制表达式

IR7..IR2 为指令的高六位（低两位用于选择寄存器 R0..R3）

T3,T2,T1,T0 为处于的周期

```
!EMRD = !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T2          // MOV A, MM        T2 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T1          // MOV A, MM        T1 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T0          // MOV A, MM        T0 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & IR2 & T1           // MOV A, #II       T1 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & IR2 & T0 ;         // MOV A, #II       T0 周期
!PCOE = !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T2          // MOV A, MM        T2 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T0          // MOV A, MM        T0 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & IR2 & T1           // MOV A, #II       T1 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & IR2 & T0 ;         // MOV A, #II       T0 周期
!EMEN = !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T2          // MOV A, MM        T2 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T1          // MOV A, MM        T1 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & IR2 & T2;          // MOV A, #II       T1 周期
!MAREN = !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T2;        // MOV A, MM        T2 周期
!MAROE = !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T1;        // MOV A, MM        T1 周期
!AEN = # !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T1         // MOV A, MM        T1 周期
     # !IR7 & IR6 & IR5 & IR4 & IR3 & IR2 & T1;           // MOV A, #II       T1 周期
!IREN = !IR7 & IR6 & IR5 & IR4 & IR3 & !IR2 & T0          // MOV A, MM        T0 周期
      # !IR7 & IR6 & IR5 & IR4 & IR3 & IR2 & T0;          // MOV A, #II       T0 周期
```

上面给出的表达式仅是两条指令的表达式，而且没有化简的，不难看出 24 位控制信号是指令码及周期数的函数。增加一条指令，只要增加一些或项即可，如增加 ADD A，#11H

助记符	状态	微地址	微程序	相关控制位
ADD A,#II	T2	1C	C7FFEF	EMRD,PCOE,EMEN,WEN
	T1	1D	FFFE90	FEN,X1,X0,AEN,S2,S1,S0
	T0	1E	CBFFFF	EMRD,PCOE,IREN

EMRD 增加：

!IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T2 // ADD A, #II T2 周期

!IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T0 // ADD A, #II T0 周期

PCOE 增加：

!IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T2 // ADD A, #II T2 周期

!IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T0 // ADD A, #II T0 周期

EMEN 增加：

!IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T2 // ADD A, #II T2 周期

AEN 增加：

!IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T1 // ADD A, #II T1 周期

IREN 增加：

!IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T0 // ADD A, #II T0 周期

ADD A， #II 新增加的控制信号有 WEN，FEN，X2，X1，S2，S1，S0

!WEN = !IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T2 // ADD A, #II T2 周期

!FEN = !IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T1 // ADD A, #II T1 周期

!X2 = !IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T1 // ADD A, #II T1 周期

!X1 = !IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T1 // ADD A, #II T1 周期

!S2 = !IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T1 // ADD A, #II T1 周期

!S1 = !IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T1 // ADD A, #II T1 周期

!S0 = !IR7 & !IR6 & !IR5 & IR4 & IR3 & IR2 & T1 // ADD A, #II T1 周期

IR7..IR0 由指令寄存器提供，ABEL 表达式是：

[IR7..IR0] := [IBUS7..IBUS0];

[IR7..IR0].CE = !IREN;

[IR7..IR0].AR = !RST;

[IR7..IR0].CLK = CK;

CPTH 每条指令最多有 4 个周期（T3，T2，T1，T0），可由两位 D 触发器（RT1，RT0）表示。

T3 = RT1 & RT0;

T2 = RT1 & !RT0;

T1 = !RT1 & RT0;

T0 = !RT1 & !RT0;

RT1,RT0 构成一个带预置的减计数器，ABEL 表达式是：

```
WHEN !RT1 & !RT0 THEN {
    [RT1..RT0] := [CT1..CT0];
}   ELSE {
    [RT1..RT0] := [RT1..RT0] - 1;
}
[RT1..RT0].CLK = CK;
[RT1..RT0].AR = !RST;
```

当 RT1，RT0 为 0 时，表示现执行的是本指令的最后一个周期，这个周期为取指周期。在取指时将 RT1，RT0 置为下一条指令的首个周期值。

当 RT1，RT0 不为 0 时，将周期数减一。

CT1，CT0 根据指令计算出，ABEL 表达式是：

```
TRUTH_TABLE([IBUS7,IBUS6,IBUS5,IBUS4,IBUS3,IBUS2]->[CT1, CT0])
[ 0, 1, 1, 1, 1, 0]->[ 1, 0]      ; // MOV A, MM
[ 0, 1, 1, 1, 1, 1]->[ 0, 1]      ; // MOV A, #II
```

我们根据模型机提供的缺省指令系统，在 ispLEVER 软件开发环境中，参照下面 ABEL 格式的文件来设计组合逻辑控制器，来验证 4．1 实验的正确性，省略部分由学生自己完成。CPTH 实验仪上的组合逻辑控制器由一片 LC4256V-100 实现，通过开关 KC 切换．出厂时已下载的组合逻辑控制器文件 LOGIC 见随机光盘。

```
module LOGIC
Title 'LOGIC'
Declarations
XRD,FF PIN   istype 'COM';
EMWR PIN istype 'COM';
EMRD PIN istype 'COM';
PCOE PIN istype 'COM';
EMEN PIN istype 'COM';
IREN PIN istype 'COM';
EINT PIN istype 'COM';
ELP PIN   istype 'COM';
MAREN PIN istype 'COM';
MAROE PIN istype 'COM';
OUTEN PIN   istype 'COM';
STEN PIN istype 'COM';
RRD PIN istype 'COM';
RWR PIN istype 'COM';
CN PIN   istype 'COM';
FEN PIN istype 'COM';
X2 PIN   istype 'COM';
```

```
X1 PIN    istype 'COM';
X0 PIN    istype 'COM';
WEN PIN istype 'COM';
AEN PIN istype 'COM';
S2 PIN istype 'COM';
S1 PIN istype 'COM';
S0 PIN istype 'COM';
IBUS7 PIN ;
IBUS6 PIN ;
IBUS5 PIN ;
IBUS4 PIN ;
IBUS3 PIN ;
IBUS2 PIN ;
IBUS1 PIN ;
IBUS0 PIN ;
CK PIN ;
RST PIN;
RT1 PIN istype 'REG';
RT0 PIN istype 'REG';
CT1 PIN istype 'COM';
CT0 PIN istype 'COM';
MON PIN ;
IR7 NODE istype 'REG';
IR6 NODE istype 'REG';
IR5 NODE istype 'REG';
IR4 NODE istype 'REG';
IR3 NODE istype 'REG';
IR2 NODE istype 'REG';
IR1 NODE istype 'REG';
IR0 NODE istype 'REG';
T3 pin istype 'COM';
T2 pin istype 'COM';
T1 pin istype 'COM';
T0 pin istype 'COM';
Equations
// XRD
!XRD = IR7 & !IR6 & !IR5 & IR4 & !IR3 & !IR2 & T1;          // READ A, MM
// EMWR
!EMWR = IR7 & !IR6 & !IR5 & !IR4 & !IR3 & IR2 & T1 #       // MOV @R?, A
```

```
            IR7 & !IR6 & !IR5 & !IR4 & IR3 & !IR2 & T1 ;       // MOV MM, A
// EMRD
!EMRD = !IR7 & !IR6 & !IR5 & !IR4 & !IR3 & !IR2 & T0 #     // _FATCH_
        !IR7 & !IR6 & !IR5 & !IR4 & !IR3 & IR2 & T0 #          // UNDEF
        !IR7 & !IR6 & !IR5 & !IR4 & IR3 & !IR2 & T0 #          // UNDEF
        !IR7 & !IR6 & !IR5 & !IR4 & IR3 & IR2 & T0 #           // UNDEF
        !IR7 & !IR6 & !IR5 & IR4 & !IR3 & !IR2 & T0 #          // ADD A, R?
        !IR7 & !IR6 & !IR5 & IR4 & !IR3 & IR2 & T2 #           // ADD A, @R?
        !IR7 & !IR6 & !IR5 & IR4 & !IR3 & IR2 & T0 #           // ADD A, @R?
        ————————————————————以下省略部分由学生自己完成
        ……
        ……
        ……
————————————————————
// Does not output when MON is high
XRD.OE = !MON&FF;
EMWR.OE = !MON&FF;
EMRD.OE = !MON&FF;
PCOE.OE = !MON&FF;
EMEN.OE = !MON&FF;
IREN.OE = !MON&FF;
EINT.OE = !MON&FF;
ELP.OE = !MON&FF;
MAREN.OE = !MON&FF;
MAROE.OE = !MON&FF;
OUTEN.OE = !MON&FF;
STEN.OE = !MON&FF;
RRD.OE = !MON&FF;
RWR.OE = !MON&FF;
CN.OE = !MON&FF;
FEN.OE = !MON&FF;
X2.OE = !MON&FF;
X1.OE = !MON&FF;
X0.OE = !MON&FF;
WEN.OE = !MON&FF;
AEN.OE = !MON&FF;
S2.OE = !MON&FF;
S1.OE = !MON&FF;
S0.OE = !MON&FF;
```

```
// Load IR register
[IR7..IR0] := [IBUS7..IBUS0];
[IR7..IR0].CE = !IREN;
[IR7..IR0].AR = !RST;
[IR7..IR0].CLK = CK;
// T counter
WHEN !RT1 & !RT0 THEN {
[RT1..RT0] := [CT1..CT0];
} ELSE {
[RT1..RT0] := [RT1..RT0] - 1;
}
[RT1..RT0].CLK = CK;
[RT1..RT0].AR = !RST;
// set T
T3 = RT1 & RT0;
T2 = RT1 & !RT0;
T1 = !RT1 & RT0;
T0 = !RT1 & !RT0;
//    constant for CT1,0 counter
TRUTH_TABLE ([IBUS7,IBUS6,IBUS5,IBUS4,IBUS3,IBUS2] -> [CT1, CT0])
[ 0, 0, 0, 0, 0, 0] -> [ 0, 0];     // Fetch
[ 0, 0, 0, 0, 0, 1] -> [ 0, 0];       // UNDEF
[ 0, 0, 0, 0, 1, 0] -> [ 0, 0];     // UNDEF
[ 0, 0, 0, 0, 1, 1] -> [ 0, 0];     // UNDEF
[ 0, 0, 0, 1, 0, 0] -> [ 1, 0];     // ADD A, R?
[ 0, 0, 0, 1, 0, 1] -> [ 1, 1];     // ADD A, @R?
[ 0, 0, 0, 1, 1, 0] -> [ 1, 1];     // ADD A, MM
[ 0, 0, 0, 1, 1, 1] -> [ 1, 0];     // ADD A, #II
[ 0, 0, 1, 0, 0, 0] -> [ 1, 0];     // ADDC A, R?
[ 0, 0, 1, 0, 0, 1] -> [ 1, 1];     // ADDC A, @R?
[ 0, 0, 1, 0, 1, 0] -> [ 1, 1];     // ADDC A, MM
[ 0, 0, 1, 0, 1, 1] -> [ 1, 0];     // ADDC A, #II
[ 0, 0, 1, 1, 0, 0] -> [ 1, 0];     // SUB A, R?
[ 0, 0, 1, 1, 0, 1] -> [ 1, 1];     // SUB A, @R?
[ 0, 0, 1, 1, 1, 0] -> [ 1, 1];     // SUB A, MM
[ 0, 0, 1, 1, 1, 1] -> [ 1, 0];     // SUB A, #II
[ 0, 1, 0, 0, 0, 0] -> [ 1, 0];     // SUBC A, R?
[ 0, 1, 0, 0, 0, 1] -> [ 1, 1];     // SUBC A, @R?
[ 0, 1, 0, 0, 1, 0] -> [ 1, 1];     // SUBC A, MM
```

```
[ 0, 1, 0, 0, 1, 1] -> [ 1, 0];    // SUBC A, #II
[ 0, 1, 0, 1, 0, 0] -> [ 1, 0];    // AND A, R?
[ 0, 1, 0, 1, 0, 1] -> [ 1, 1];    // AND A, @R?
[ 0, 1, 0, 1, 1, 0] -> [ 1, 1];    // AND A, MM
[ 0, 1, 0, 1, 1, 1] -> [ 1, 0];      // AND A, #II
[ 0, 1, 1, 0, 0, 0] -> [ 1, 0];    // OR A, R?
[ 0, 1, 1, 0, 0, 1] -> [ 1, 1];    // OR A, @R?
[ 0, 1, 1, 0, 1, 0] -> [ 1, 1];    // OR A, MM
[ 0, 1, 1, 0, 1, 1] -> [ 1, 0];    // OR A, #II
[ 0, 1, 1, 1, 0, 0] -> [ 0, 1];    // MOV A, R?
[ 0, 1, 1, 1, 0, 1] -> [ 1, 0];    // MOV A, @R?
[ 0, 1, 1, 1, 1, 0] -> [ 1, 0];    // MOV A, MM
[ 0, 1, 1, 1, 1, 1] -> [ 0, 1];    // MOV A, #II
[ 1, 0, 0, 0, 0, 0] -> [ 0, 1];    // MOV R?, A
[ 1, 0, 0, 0, 0, 1] -> [ 1, 0];    // MOV @R?, A
[ 1, 0, 0, 0, 1, 0] -> [ 1, 0];    // MOV MM, A
[ 1, 0, 0, 0, 1, 1] -> [ 0, 1];    // MOV R?, #II
[ 1, 0, 0, 1, 0, 0] -> [ 1, 0];    // READ A, MM
[ 1, 0, 0, 1, 0, 1] -> [ 1, 0];      // WRITE MM, A
[ 1, 0, 0, 1, 1, 0] -> [ 0, 0];      // UNDEF
[ 1, 0, 0, 1, 1, 1] -> [ 0, 0];      // UNDEF
[ 1, 0, 1, 0, 0, 0] -> [ 0, 1];    // JC MM
[ 1, 0, 1, 0, 0, 1] -> [ 0, 1];    // JZ MM
[ 1, 0, 1, 0, 1, 0] -> [ 0, 0];    // UNDEF
[ 1, 0, 1, 0, 1, 1] -> [ 0, 1];    // JMP MM
[ 1, 0, 1, 1, 0, 0] -> [ 0, 0];    // UNDEF
[ 1, 0, 1, 1, 0, 1] -> [ 0, 0];    // UNDEF
[ 1, 0, 1, 1, 1, 0] -> [ 1, 0];    // _INT_
[ 1, 0, 1, 1, 1, 1] -> [ 1, 1];    // CALL MM
[ 1, 1, 0, 0, 0, 0] -> [ 0, 1];    // IN
[ 1, 1, 0, 0, 0, 1] -> [ 0, 1];    // OUT
[ 1, 1, 0, 0, 1, 0] -> [ 0, 0];    // UNDEF
[ 1, 1, 0, 0, 1, 1] -> [ 0, 1];    // RET
[ 1, 1, 0, 1, 0, 0] -> [ 0, 1];    // RR A
[ 1, 1, 0, 1, 0, 1] -> [ 0, 1];    // RL A
[ 1, 1, 0, 1, 1, 0] -> [ 0, 1];    // RRC A
[ 1, 1, 0, 1, 1, 1] -> [ 0, 1];    // RLC A
[ 1, 1, 1, 0, 0, 0] -> [ 0, 0];      // NOP
[ 1, 1, 1, 0, 0, 1] -> [ 0, 1];    // CPL A
```

```
[ 1, 1, 1, 0, 1, 0] -> [ 0, 0];     // UNDEF
[ 1, 1, 1, 0, 1, 1] -> [ 0, 1];     // RETI
[ 1, 1, 1, 1, 0, 0] -> [ 0, 0];     // UNDEF
[ 1, 1, 1, 1, 0, 1] -> [ 0, 0];     // UNDEF
[ 1, 1, 1, 1, 1, 0] -> [ 0, 0];     // UNDEF
[ 1, 1, 1, 1, 1, 1] -> [ 0, 0];     // UNDEF
end LOGIC
```

5.2 用 CPLD 实现运算器功能

在 ispLEVER 软件开发环境中，我们可以参照下面 VHDL 格式的文件来设计运算器，来验证 2．2 实验的运算器功能 ，省略部分由学生自己完成．CPTH 实验仪上的运算器由一片 LC4256V-100 实现，出厂时已下载的运算器文件 ALU 见随机光盘，用户可自行修改以实现其功能。

```
library ieee;
use ieee.std_logic_1164.all;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ALU is
   PORT (
      clk       : IN       STD_LOGIC;                                  -- 时钟输入
      rst            : IN       STD_LOGIC;                            -- 复位输入
      AEN    :IN STD_LOGIC; -- A 写允许
      WEN    :IN STD_LOGIC; -- W 写允许
      S0     :IN STD_LOGIC;
      S1     :IN STD_LOGIC;
      S2     :IN STD_LOGIC; --  运算器功能选择
      X0     :IN STD_LOGIC;
      X1     :IN STD_LOGIC;
      X2     :IN STD_LOGIC; --  寄存器输出控制
      R_CY: BUFFER STD_LOGIC; --  进位标志寄存器
      R_Z : OUT STD_LOGIC; --  零标志寄存器
      FEN    :IN STD_LOGIC; --  标志寄存器写允许
      CN     :IN STD_LOGIC; --  移位时是否带进位
      RL0,RR7,JRC,JRZ:OUT STD_LOGIC;
      D_BUS:INOUT STD_LOGIC_VECTOR(7 DOWNTO 0);
      PC7,CY_IN:IN STD_LOGIC;手动控制和手动进位
      ALU       :OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
   );
end;
```

```
architecture ALU of ALU is
-- 寄存器定义
SIGNAL A     : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL W     : STD_LOGIC_VECTOR(7 DOWNTO 0);
-- 标志定义
SIGNAL CY    : STD_LOGIC; -- 本次运算进位标志
SIGNAL Z     : STD_LOGIC; -- 本次运算零标志
-- ALU 运算器定义
SIGNAL T    : STD_LOGIC_VECTOR(8 DOWNTO 0);   -- 运算结果
begin
-- 寄存器 A
   PROCESS(clk, rst, AEN)
   BEGIN
    IF rst = '0' THEN
        A <= "00000000";
      ELSIF AEN = '1' THEN
        NULL;
      ELSIF clk'EVENT AND clk = '1' THEN
        A <= D_BUS;
      END IF;
   END PROCESS;

   -- 寄存器 W
   PROCESS(clk, rst, WEN)
   BEGIN
      IF rst = '0' THEN
        W <= "00000000";
      ELSIF WEN = '1' THEN
        NULL;
      ELSIF clk'EVENT AND clk = '1' THEN
        W <= D_BUS;
      END IF;
   END PROCESS;
------------------
   RL0<=CN AND R_CY WHEN PC7='0' ELSE
         CN AND CY_IN;
   RR7<=CN AND R_CY WHEN PC7='0' ELSE
         CN AND CY_IN;
```

```
--------------------    ＡＬＵ运算，以下省略部分由学生自己完成.
……
……
……
-------------------------
  JRZ <=R_Z       WHEN PC7='0' ELSE 'Z';
    JRC <=R_CY WHEN PC7='0' ELSE 'Z';
end ALU;
```

]

第六章　　设计指令/微指令系统

CPTH 计算机组成原理实验仪，可以由用户自己设计指令/微指令系统，前一章的“指令流水实验”就是利用另一套指令/微指令系统来实现指令的流水工作。这样用户可以在现有的指令系统上进行扩充，加上一些较常用的指令，也可重新设计一套完全不同的指令/微指令系统。CPTH 内已经内嵌了一个智能化汇编语言编译器，可以对用户设定的汇编助记符进行汇编。做为例题，我们建立一个有如下指令的系统：

指令助记符	指令意义描述
LD A，#*	将立即数装入累加器 A
ADD A，#*	累加器 A 加立即数
GOTO *	无条件跳转指令
OUT A	累加器 A 输出到端口

因为硬件系统需要指令机器码的最低两位做为 R0-R3 寄存器寻址用，所以指令机器码要忽略掉这两位。我们暂定这四条指令的机器码分别为 04H，08H，0CH，10H。其它指令的设计，用户可参考此例，做为练习来完成。

一・创建指令系统(助记符、机器码)

1．打开 CPTH 组成原理实验软件，选择[文件|新建指令系统/微程序]，清除原来的指令/微程序系统，观察软件下方的“指令系统”窗口，所有指令码都“未使用”。

2．选择第二行，即“机器码 1”为 0000 01XX 行，在下方的“助记符”栏填入数据装载功能的指令助记符“LD”，在“操作数 1”栏选择“A”，表示第一个操作数为累加器 A。在“操作数 2”栏选择“#*”，表示第二个操作数为立即数。按“修改”按钮确认。

3．选择第三行，即“机器码 1”为 0000 10XX 行，在下方的“助记符”栏填入加法功能的指令助记符“ADD”，在“操作码 1”栏选择“A”，表示第一操作数为累加器 A，在“操作数 2”栏选择“#*”，表示第二操作数为立即数。按“修改”按钮确认。

4．选择第四行，即“机器码 1”为 0000 11XX 行，在下方的“助记符”栏填入无条件跳转功能的指令助记符“GOTO”，在“操作码 1”栏选择“*”，表示跳转地址为*，此指令无第二操作数，无需选择“操作数 2”。按“修改”按钮确认。因为硬件设计时，跳转指令的跳转控制需要指令码的第 3 位和第 2 位 IR3、IR2 来决定，无条件跳转的控制要求

IR3 必需为 1，所以无条件跳转的机器码选择在此行，机器码为 000011XX。关于跳转控制的硬件设计和实验可参考前面章节。

5．选择第五行，即“机器码 1”为 0001 00XX 行，在下方的“助记符”栏填入输出数据功能的指令助记符“OUT”，由于此指令隐含指定了将累加器 A 输出到输出寄存器，所以不用选择“操作码 1”和“操作数 2”，也可在“操作码 1”栏选择“A”,按“修改”按钮确认。现在我们只是输入了四条指令(见下图)

二．下面我们根据指令的功能来设计相应的微程序。

将窗口切换到“uM 微程序”窗口，现在此窗口中所有微指令值都是 0FFFFFFH，也就是无任何操作，我们需要在此窗口输入每条指令的微程序来实现该指令的功能。

⑴每个程序开始要执行的第一条微指令应是取指操作，因为程序复位后，PC 和 uPC 的值都为 0，所以微程序的 0 地址处就是程序执行的第一条取指的微指令。取指操作要做的工作是从程序存储器 EM 中读出下条将要执行的指令，并将指令的机器码存入指令寄存器 IR 和微程序计数器 uPC 中，读出下条操作的微指令。根据此功能，观察窗口下方的各控制信号，有“勾”表示信号为高，处于无效状态，去掉“勾”信号为低，为有效状态。要从 EM 中读数，EMRD 必需有效，去掉信号下面的“勾”使其有效；读 EM 的地址要从 PC 输出，所以 PCOE 要有效，允许 PC 输出，去掉 PCOE 下面的“勾”，PCOE 有效同时还会使 PC 加 1，准备读 EM 的下一地址；IREN 是将 EM 读出的指令码存入 uPC 和 IR，所以要去掉 IREN 的“勾”使其有效。这样，取指操作的微指令就设计好了，取指操作的微指令的值为 CBFFFFH。然后把助记符 " _FATCH_ " ，状态 " T0 " ，微地址 " 0 0 " ，微程序 " CBFFFFH " . .（其余为注释可略），按照上表格式填入，其余类同.

⑵现在我们来看把立即数装入累加器 A 要做哪些工作，首先要从 EM 中读出立即数，并送到数据总线 DBUS，再从 DBUS 上将数据打入累加器 A 中，按照这个要求，从 EM 中读数据，EMRD 应该有效，EM 的地址由 PC 输出，PCOE 必需有效，读出的数据送到 DBUS，EMEN 也应有效,要求将数据存入 A 中,AEN 也要有效，根据前面描述“LD A，#*”指令有两个状态周期，T1 状态将所有有效位下面的“勾”去掉，使其有效，这条微指令的值为 C7FFF7H。为了保证程序的连续执行，每条指令的最后必需是取指令，取出下条将要执行的指令。T0 状态取指微指令的值为 CBFFFFH。（取操作描述可见第⑴ 步）。

⑶本指令为立即数加法指令，立即数加可分两步，首先从 EM 中读出立即数，送到 DBUS，并存入工作寄存器 W 中，从 EM 中读数，EMRD 应有效，读 EM 的地址由 PC 输出，PCOE 要有效，读出的数据要送到 DBUS，EMEN 应有效，数据存入 W 中，WEN 应有效，根据描述，“ADD A,#*”指令的 T1 状态微指令的值为 C7FFEFH。第二步，执行加法操作，并将结果存入 A 中。执行加法操作，S2S1S0 的值应为 000（二进制），结果无需移位直接输出到 DBUS，X2X1X0 的值就要为 100（二进制），从 DBUS 将数据再存入 A 中，AEN 应有效。与此同时，ABUS 和 IBUS 空闲，取指操作可以并行执行，也就是以 PC 为地址，从 EM 中读出下条将要执行指令的机器码，并打入 IR 和 uPC 中，根据取指操作的说明，EMRD、PCOE、IREN 要有效，根据上面描述，T0 状态时将 EMRD、PCOE、IREN、X2X1X0、AEN、S2S1S0 都置成有效和相应的工作方式，此微指令的值为 CBFF90H。

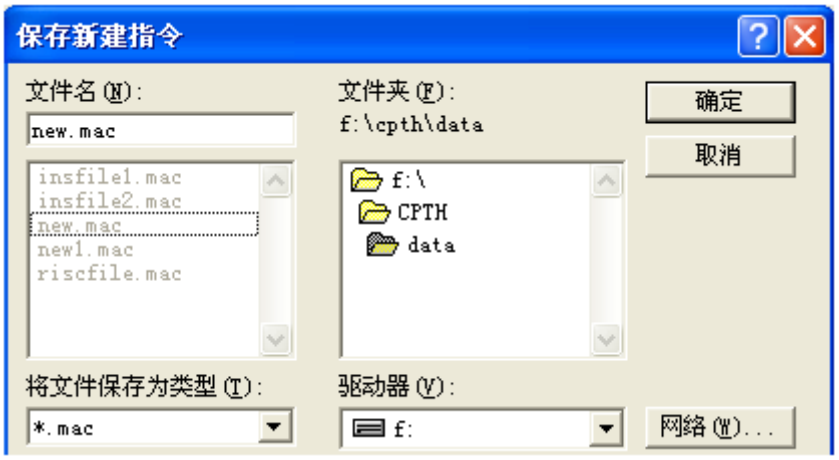
⑷“GOTO *”为无条件跳转，所要执行的操作为从 EM 中读出目标地址，送到数据总线 DBUS 上，并存入 PC 中，实现程序跳转。从 EM 中读数，EMRD 要有效，读 EM 的地址由 PC 输出，PCOE 有效，数据送到 DBUS，EMEN 要有效，将数据打入 PC 中，由两位决定，ELP 有效，指令寄存器 IR 的第三位 IR3 应为 1，由于本指令机器码为 0CH，存入 IR 后，IR3 为 1。将 EMRD、PCOE、EMEN、ELP 设成低，使其成为有效状态，结合指令的第三位，实现程序跳转，这条微指令的值为 C6FFFFH。下条微指令应为取指操作，将 EMRD、PCOE、IREN 设成有效，微指令的值为 CBFFFFH。

⑸“OUTA”，将累加器的内容输出到输出端口。其操作为累加器 A 不做运算，直通输出，ALU 结果不移位输出到 DBUS，DBUS 上的数据存入输出端口 OUT。累加器 A 直通输出结果，S2S1S0 值要为 111(二进制)，ALU 结果不移位输出到数据总线 DBUS，X2X1X0 的值要等于 100(二进制)，DBUS 数据要打入 OUT，那么 OUTEN 应有效。与此同时，ABUS 和 IBUS 空闲，取指操作可以并行执行，也就是以 PC 为地址，从 EM 中读出下条将要执行指令的机器码，并打入 IR 和 uPC 中，根据取指操作的说明，EMRD、PCOE、IREN 要有效，综上所述，将 EMRD、PCOE、IREN、OUTEN、X2X1X0、S2S1S0 置成有效状态和相应的工作方式，微指令的值为 CBDF9FH。

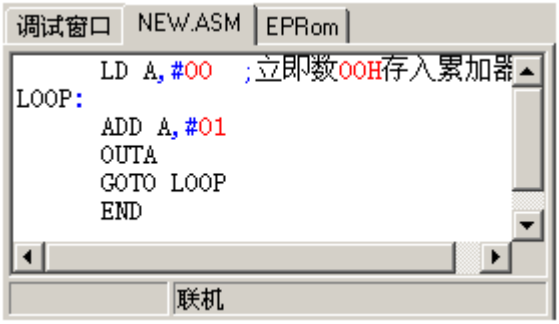
指令集 | uM微程序 | 跟踪

助记符	状态	微地址	微程序	数据输出	数据打入	地址输出	运算器	移位控制	uPC	PC
FETCH	T0	00	CBFFFF	FF 浮空		浮空	A输出		+	1
		01	FFFFFF	浮空		浮空	A输出		+1	
		02	FFFFFF	浮空		浮空	A输出		+1	
		03	FFFFFF	浮空		浮空	A输出		+1	
LD A,#*	T1	04	C7FFF7	浮空		浮空	A输出		+1	
	T0	05	CBFFFF	浮空		浮空	A输出		+1	

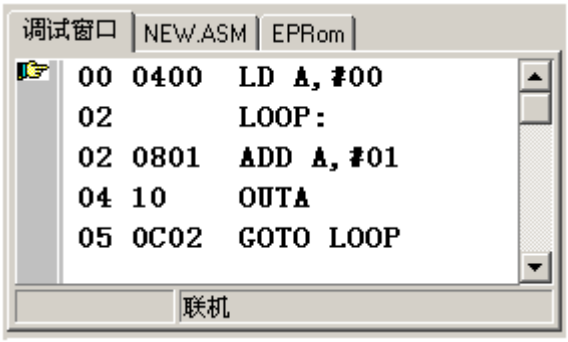
(6)选择菜单[文件|保存指令系统/微程序]功能，将新建的指令系统/微程序保存下来，以便以后调用。为不与已有的指令系统冲突，将新的指令系统/微程序保存为“new.mac”



三、在源程序窗口输入下面程序



将程序另存为 NEW.ASM，选择[文件|调入指令系统/微程序]，调入 new.mac，将程序汇编成机器码，观察反汇编窗口，会显示出程序地址、机器码、反汇编指令。



按快捷图标的 F7，执行“单微指令运行”功能，观察执行每条微指令时，数据是否按照设计要求流动，寄存器的输入/输出状态是否符合设计要求，各控制信号的状态，PC

及 uPC 如何工作是否正确。

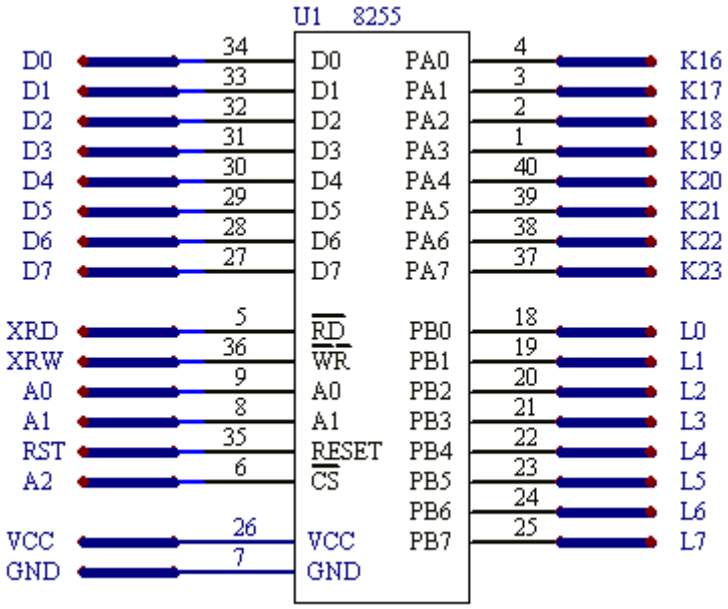
到此为止，我们利用 CPTH 软件系统已经建成了一个新的指令系统/微程序。新的指令系统从汇编助记符到指令机器码到微指令都与原来的指令系统有所不同。做为例子，我们只创建了四条指令，对于其它指令，用户可以做为练习来扩充完整，成为一个真正的指令系统。

第七章　　扩展实验

CPTH 计算机组成原理实验仪的模型机具有对外部设备操作的指令，用这两条的输入、输出指令 可以对扩展的外部设备进行操作，实现功能的扩展。例如扩展外部的 8255 来扩展输入、输出端口，扩展外部的 8253 来扩展定时器/计数器等等。

扩展实验一：用 8255 扩展 I/O 端口实验

1．将 8255 插在 40 芯紧锁座上，按图和接线表连接好 8255 各信号线。



2．打开 CPTH 计算机组成原理实验仪电源，运行 CPTH 软件，将软件连接到实验仪硬件。输入下面程序：（或从 CPTH 目录下调入 EX8255.ASM）。

调试窗口 | ex8255.asm | EPRom

```
        CONTROL EQU 03H
        PORTA   EQU 00H
        PORTB   EQU 01H
        MOV     A, #90H
        WRITE   CONTROL, A
LOOP:
        READ    A, PORTA
        CPL     A
        WRITE   PORTB, A
        JMP     LOOP
        END
```

行:11列:1　　联机

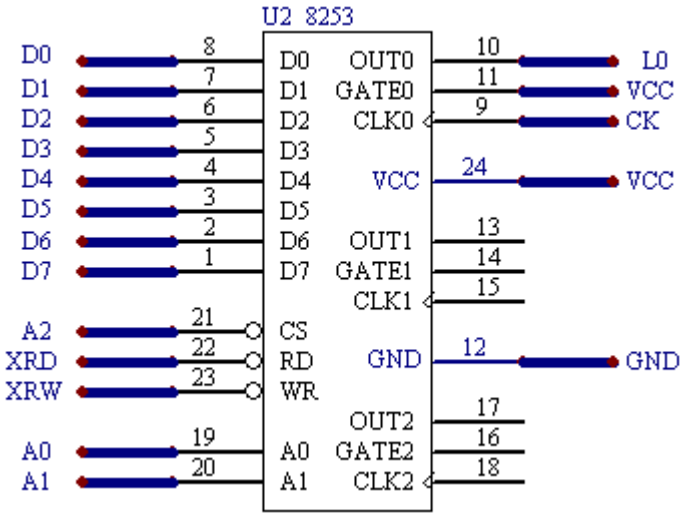
连接	8255 管脚	接入孔
1	21	L3
2	22	L4
3	23	L5
4	24	L6
5	25	L7
6	26	+5V
7	27	D7
8	28	D6
9	29	D5
10	30	D4
11	31	D3
12	32	D2
13	33	
14	34	D0
15	35	RST
16	36	XWR
17	37	K23
18	38	K22
19	39	K21
20	40	K20

连接	8255 管脚	接入孔
21	20	L2
22	19	L1
23	18	L0
24	17	不接
25	16	不接
26	15	不接
27	14	不接
28	13	不接
29	12	不接
30	11	不接
31	10	不接
32	9	A0
33	8	A1
34	7	GND
35	6	A2
36	5	XRD
37	4	K16
38	3	
39	2	K18
40	1	K19

3．将程序编译后全速执行，程序从 8255 的 PA 口读回数据，取反后输出到 PB 口，重复循环。拨动 K16-K23 开关，可以看到 L0-L7 上有相应的输出。

扩展实验二：用 8253 扩展定时器试验

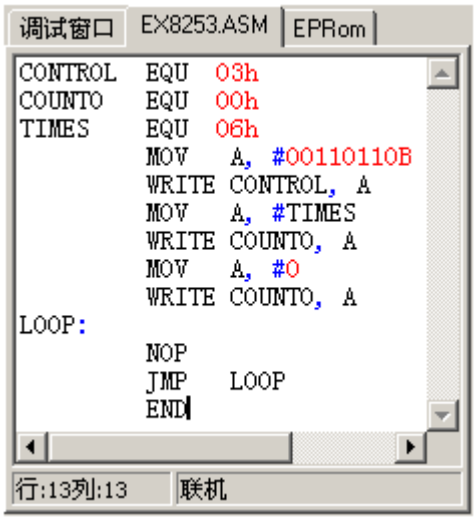
1．将 8253 插在 40 芯紧锁座上，按图和接线表接好 8253 的信号线。



连接	8253管脚	接入孔
1	13	不接
2	14	不接
3	15	不接
4	16	不接
5	17	不接
6	18	不接
7	19	A0
8	20	A1
9	21	A2
10	22	XRD
11	23	XRW
12	24	+5V

连接	8253管脚	接入孔
13	12	GND
14	11	+5V
15	10	L0
16	9	CK
17	8	D0
18	7	D1
19	6	D2
20	5	D3
21	4	D4
22	3	D5
23	2	D6
24	1	D7

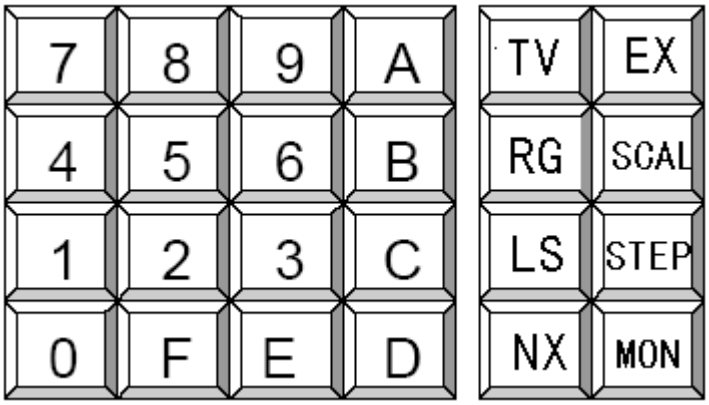
2．打开 CPTH 计算机组成原理实验仪电源，运行 CPTH 软件，将软件连接到实验仪硬件。输入下面程序：（或从 CPTH 目录下调入 EX8253.ASM）。



3．将程序汇编后单步执行到循环处，再单步观察 L0 灯的翻转情况，如果全速执行 L0 翻转过快，可对 8253 计数器高字节写入一个数字，再全速执行，观察 L0 灯。

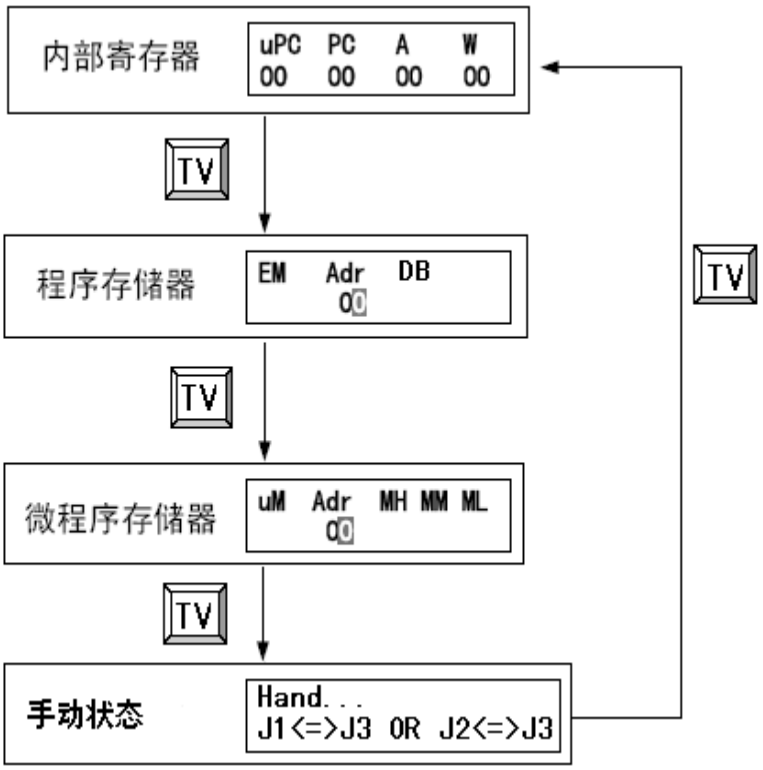
第八章　　实验仪键盘使用

DJ-CPTH 计算机组成原理实验仪除了可以连在 PC 机上调试程序，也可以用实验仪上自带的键盘输入程序及微程序，并可以单步调试程序和微程序，在显示屏上观察、修改各内部寄存器的值，编辑修改程序和微程序存储器。



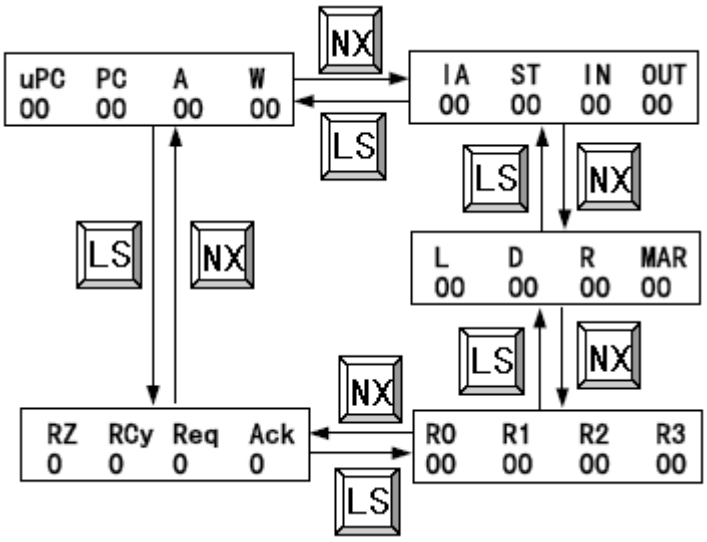
显示屏的显示内容分四个主菜单：

1、观察和修改内部寄存器；2、观察和修改程序存储器；3、观察和修改微程序存储器；4、手动状态。四个主菜单用 TV/ME 键切换。如下图：



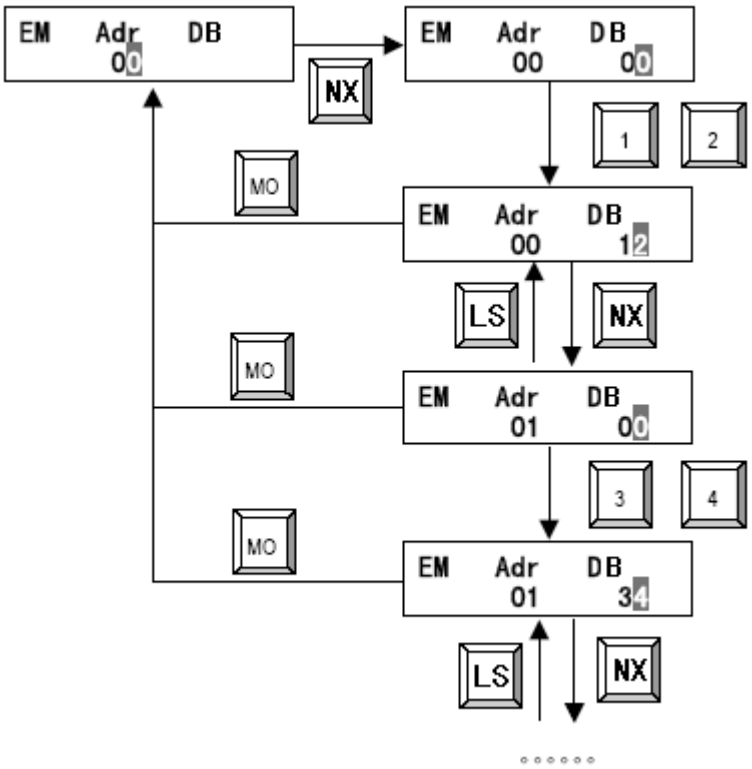
1、观察、修改内部寄存器：

内部寄存器的内容分五页显示，用 LAST 或 NEXT 键向前或向后翻页，可以显示所有内部寄存器值，输入数字可修改非只读寄存器值。见下图：



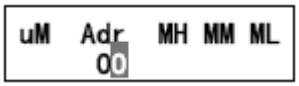
2、观察、修改程序存储器内容：

显示屏显示如下图，其中“Adr”表示程序存储器地址，“DB”表示该地址中数据。光标初始停在“Adr”处，此时可以用数字键输入想要修改的程序地址，也可以用 NEXT 和 LAST 键将光标移到“DB”处，输入或修改此地址中的数据。再次按 NEXT 或 LAST 键可自动将地址+1 或将地址-1，并可用数字键修改数据。按 MON 键可以回到输入地址 00 的状态。见下图。



3．观察、修改微程序存储器内容：

微程序存储器数据的观察、修改与上面程序存储器的观察修改方法相似，不同的是微程序要输入 3 个字节，而程序存储器的修改只要输入 1 个字节。微程序观察修改的显示屏显示如下图，其中“Adr”表示微程序地址，“MH”表示微程序的高字节，“MM”表示微程序的中字节，“ML”表示微程序的低字节。



使用实验仪键盘可以用三种方法调试程序，程序单步、微程序单步、全速执行。当用键盘调试程序时，显示屏显示寄存器第一页的内容。

[STEP] 为微程序单步执行键，每次按下此键，就执行一个微程序指令，同时显示屏显示微程序计数器、程序计数器、A 寄存器、W 寄存器的值。可以通过 NX 或 LS 键翻页

观察其它寄存器的值，观察各个寄存器的输出和输入灯的状态。

[SCAL]为程序单步执行键，每次按下此键，就执行一条程序指令，同时显示屏显示微程序计数器、程序计数器、A 寄存器、W 寄存器的值。可以通过 NX 或 LS 键翻页观察其它寄存器的值。

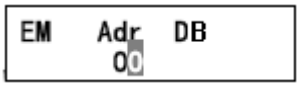
[EX]为全速执行键，按下此键时，程序就会全速执行，按键盘任一键暂停程序执行，并且显示当前 寄存器的值.

[RG]为中断请求键，按下此键时，会产生一个中断请求信号 INT.

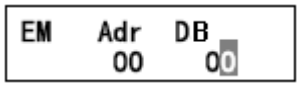
[RST]复位键，按下此键，程序中止运行，所有寄存器清零（IR 除外），程序指针回到 0 地址。

举例：用键盘输入以下程序代码：

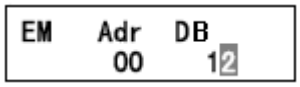
1．按 TV 键，直到显示屏显示内容为



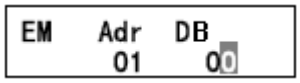
2．按 NX 键，光标移到“DB”下，显示屏为



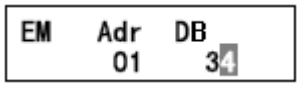
3．按 1,2 两个数字键， 显示屏为



4．按 NX 键，地址+1，显示屏为



5．按 3,4 两个数字键，显示屏为



重复 4、5 两步，直到输入所有的程序代码。

在第 1 步时，光标停在“Adr”处，可以按数字键 0 。。。F 输入要修改的程序存储器的地址，然后再按 NEXT 键输入程序代码。如果光标移到“DB”下，而此时又想改变地址，可以按 MON 键，将光标移回到“Adr”处，按数字键输入地址。输入微程序代码的方法与此相似，不同的是程序只需输入两个数字，即一个字节，而微程序要输入 6 个数字，即三个字节。如果多于 6 个数字会自动从右向左移动光标。如果输入不足 6 个数字就用 NEXT 或 LAST 翻页，则只有被改动的几个数字有效，其它数字不变。

4. 用小键盘调试实验一

程序地址	机器码	反汇编指令	指令说明
00	7C 12	MOV A，#12	立即数12H送到累加器A
02	70	MOV A，R0	寄存器R0送到累加器A
03	74	MOV A，@R0	间址的存储器内容送到累加器A
04	78 01	MOV A，01	存储器01单元内容送到累加器A
06	C0	IN	端口IN内容输入到累加器A
07	C4	OUT	累加器A内容输出到端口OUT

一：输入机器码
按 TV 键选择 EM
顺序输入机器码：7C 12 70 74 78 01 C0 C4
输完机器码后按 RST 复位
二：单步执行微程序
按 RST 复位键后，PC=0，uPC=0
uM 输出 24 位微程序：CB FF FF 此微指令为取指指令
第一条微指令
按一次 STEP 键，完成一个时钟，此时：
PC 值为 01（时钟上升沿 PC+1）
IR 值为 7C，uPC 值为 7C （指令码）
uM 输出为：C7 FF F7 （EM 值送 A）
第二条微指令
按一次 STEP 键，完成一个时钟，此时：
PC 值为 02（时钟上升沿 PC+1）
A 值为 12
uPC 值为 7D （时钟上升沿 uPC+1）
uM 输出为：CB FF FF （取指指令）
第三条微指令
按一次 STEP 键，完成一个时钟，此时：
PC 值为 03（时钟上升沿 PC+1）
IR 值为 70，uPC 值为 70 （指令码）
uM 输出为：FF F7 F7 （R？值送 A）
第四条微指令
按一次 STEP 键，完成一个时钟，此时：
PC 值为 03（时钟上升沿 PC+1）
A 值为 00
uPC 值为 71 （时钟上升沿 uPC+1）
uM 输出为：CB FF FF （取指指令）
第五条微指令
按一次 STEP 键，完成一个时钟，此时：
PC 值为 04（时钟上升沿 PC+1）
IR 值为 74，uPC 值为 74 （指令码）
uM 输出为：FF 77 FF （R？值送 MAR）
第六条微指令
按一次 STEP 键，完成一个时钟，此时：
MAR 值为 00
uPC 值为 75 （时钟上升沿 uPC+1）

uM 输出为：D7 BF F7 （EM 值送 A）

第七条微指令

按一次 STEP 键，完成一个时钟，此时：

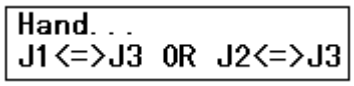
A 值为 7C

uPC 值为 76

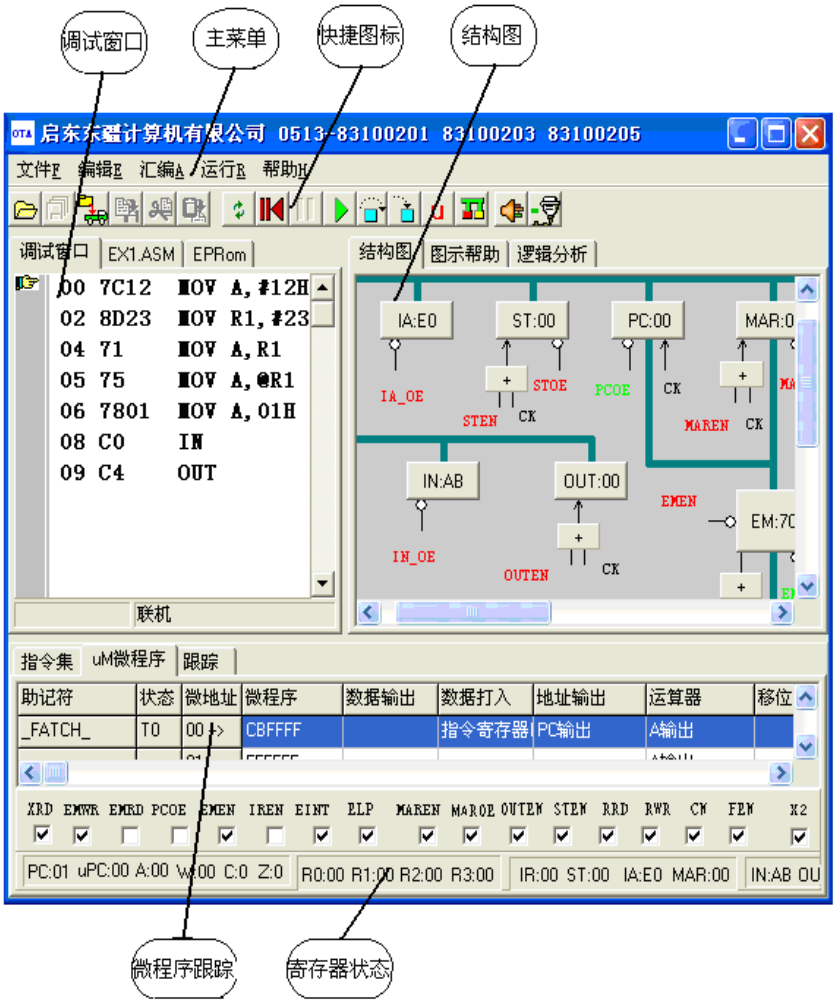
uM 输出为：CB FF FF （取指）

用同样的方法执行余下的指令。也可以用 SCAL 或 EX 键执行指令。

在做分部模块实验时，实验仪键盘 0. . . F，NX，LS 不起作用，显示屏显示内容为 8 芯电缆的连接方式。例如显示屏显示内容如下：表示手动方式，J1 通过 8 芯电缆接到 J3，或 J2 通过 8 芯电缆接到 J3。



第九章　　CPTH 集成开发环境使用



CPTH 集成调试软件界面分六部分：

1) 主菜单区　实现实验仪的各项功能的菜单，包括[文件][编辑][汇编][运行][帮助]五大项，各项下面做详细介绍。

2) 快捷图标区　快速实现各项功能按键。

3) 调试窗口区　在此区域有 调试窗口、源程序窗口、 EM 程序代码窗口。调试窗口显示程序编译后的机器码及反汇编的程序；源程序用于输入、显示、编辑汇编源程序；EM 程序代码窗口用数据方式机器码。

4) 结构图区　结构图能结构化显示模型机的各部件，以及运行时数据走向寄存器值，图示帮助进行图示化实验指导。逻辑波形图能显示模型机运行时所有信号的时序。

5) 微程序/跟踪区　微程序表格用来显示程序运行时微程序的时序，及每个时钟脉冲各控制位的状态，跟踪表用来记录显示程序及微程序执行的轨迹，指令系统显示指令集。

6) 寄存器状态区　用来显示程序执行时各内部寄存器的值。

1） 主菜单

主菜单分[文件][编辑][汇编][运行][帮助]五部分

[文件 | 打开文件] 打开已有的汇编程序或文本文件。

[文件 | 保存文件] 将修改过的文件保存。不论是汇编源程序还是其它文本文件，只要被修改过，就会被全部保存。

[文件 | 新建文件] 新建一个空的汇编源程序。

[文件 | 另存为...] 将修改过的程序换名保存。

[文件 | 打开指令系统/微程序] 打开设计好的指令系统和微程序文件，用于修改指令系统和微程序文件。

[文件 | 调入指令系统/微程序] 调入设计好的指令系统和微程序定义，用于编译汇编源程序。

[文件 | 退出] 退出集成开发环境。

[编辑 | 重做] 撤消 / 恢复上次输入的文本。

[编辑 | 剪切] 将选中的文本剪切到剪贴板上。

[编辑 | 复制] 将选中的文本复制到剪贴板上。

[编辑 | 粘贴] 从剪贴板上将文本粘贴到光标处。

[编辑 | 全选] 全部选中文本

[汇编 | 汇编] 将汇编程序汇编成机器码并下载。

[运行 | 全速执行] 全速执行程序。

[运行 | 单指令执行] 每步执行一条汇编程序指令。

[运行 | 单微指令执行] 每步执行一条微程序指令。

[运行 | 暂停] 暂停程序的全速执行。

[运行 | 复位] 将程序指针复位到程序起始处。

[帮助 | 关于] 有关 CPTH 计算机组成原理实验仪及软件的说明。

[帮助 | 帮助] 软件使用帮助。

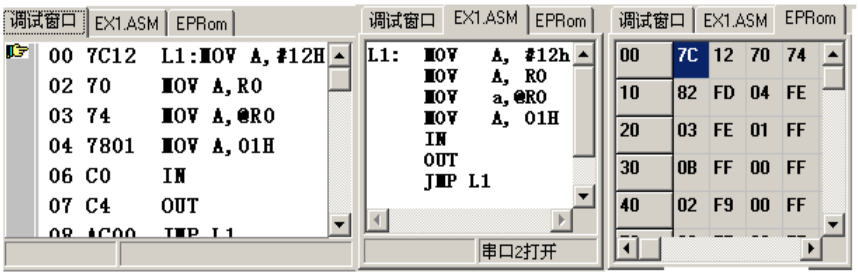
2）快捷键图标

图标的“刷新”功能就是在连机过程中刷新各寄存器，程序存储器，微程序存储器的值。以便观察实验仪的各参数内容。

文件的“打开”、“保存”功能与主菜单的相应功能一样。

文件的编辑功能，执行控制功能，其它快捷命令与主菜单也相同。

3)调试窗口区



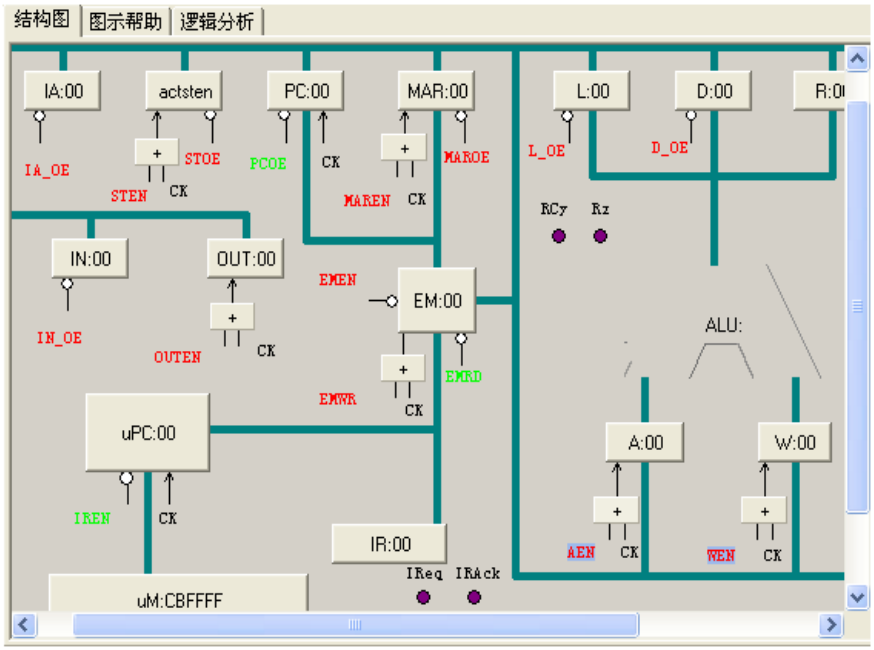
调试窗口区分三个窗口：调试窗口、源程序窗口、EM 程序窗口

源程序窗口用于输入、修改程序。在[文件]菜单中打开一个以“*.ASM”为后缀的文件时，系统认为此文件为源程序，其内容会在源程序窗口显示，并可以修改，然后编译。若再次打开以“*.ASM”后缀的文件，则新文件将旧文件覆盖，在源程序窗口只显示最新打开的汇编源程序。在[文件]菜单中，使用“新建文件”功能，会清除源程序窗口的内容，让用户重新输入新的程序。

调试窗口用于显示程序地址、机器码、反汇编后的程序。

EM 程序窗口以十六进制数据的形式显示程序编译后的机器码。可以直接输入数值来修改机器码。

4）结构图区

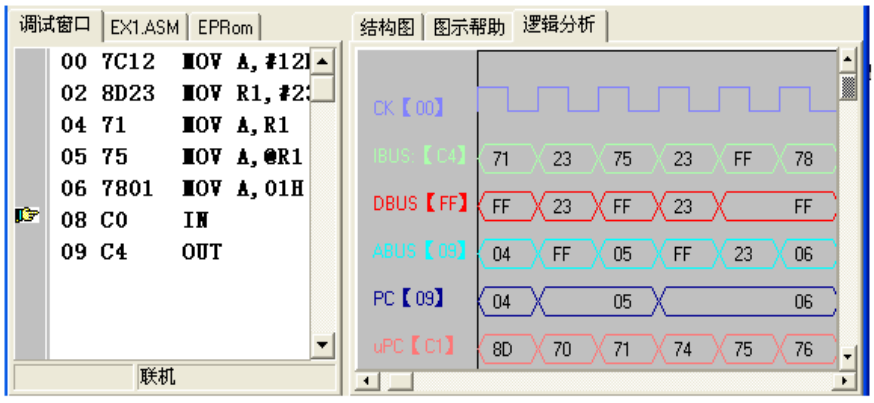


结构图区分三种窗口，结构图窗口、图示帮助窗口、逻辑分析窗口。

结构图窗口显示模型机的内部结构，包括各种寄存器（A、W、R0-R3、MAR、IR、ST、L、D、R）、运算器（ALU）、程序指针（PC）、程序存储器（EM）、微程序指针（uPC）、微程序存储器（uM）及各种状态位（RCy、Rz、IReq、IAck），在程序单步运行时，可以在结构图上看到数据的走向及寄存器的输入输出状态。其中 DBUS 为数据总线、ABUS 为地址总线、IBUS 为指令总线。RT1、RT0 显示的将要执行的指令的第几个时钟周期。本模型机最多有四个时钟周期，用 RT1、RT0 的 11、10、01、00 四个状态表示。见上图。

图示帮助进行图示化实验指导。

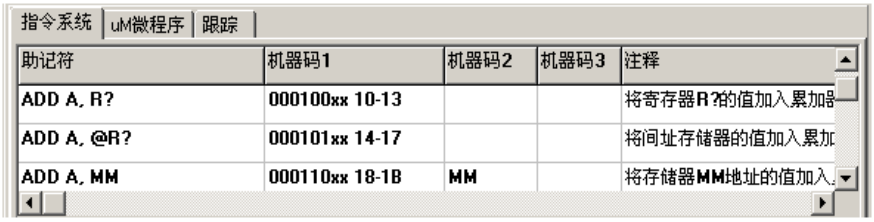
逻辑分析窗口显示的是在指令执行时，各种信号的时序波形，包括所有寄存器、所有的控制信号在不同时钟状态下的值，可以直观地看到各种信号彼此之间的先后时序关系。



5）指令/微程序/跟踪窗口

此区分三页：指令集窗口、微程序窗口、跟踪窗口。

指令集窗口用于显示指令系统和设计用户自己的指令系统。各条指令相应的微程序设计在“uM 微程序”窗口中设计（见 uM 微程序窗口）



uM 微程序窗口用于观察每条指令所对应的微程序的执行过程，以及微代码的状态。在此窗口中，可以看到数据是从何寄存器输出的、数据输入到何寄存器、地址是由 PC 输出还是由 MAR 输出、运算器在做何种运算、如何移位、uPC 及 PC 如何工作等等。可以通过改变窗口下方的微代码的各个控制位的方式来重新设计微程序，与“指令系统”窗口的指令修改相结合，可以设计自己的指令。

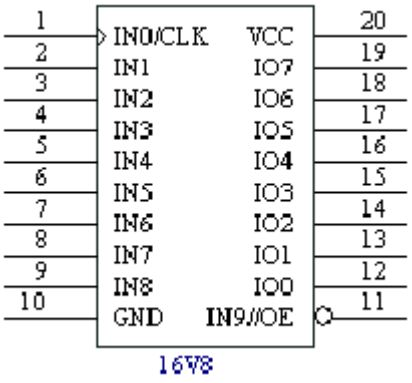
可以将鼠标移到相应的程序行或微程序行来显示执行该指令或微指令时，各寄存器、控制位的状态。

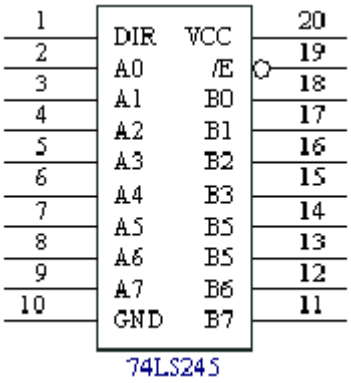
跟踪窗口显示程序执行过程的轨迹，包括每条被执行的指令、微指令，以及微指令执行时，各控制位、各个寄存器的状态。

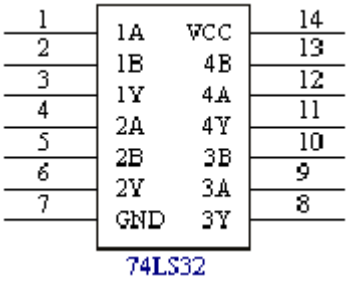
6）寄存器状态

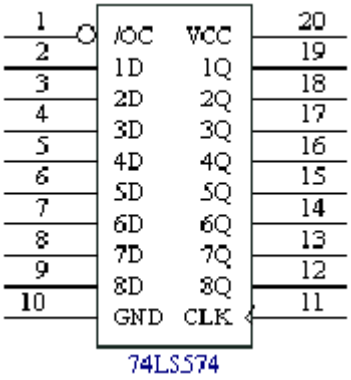
寄存器状态区显示程序执行时，各内部寄存器的值。

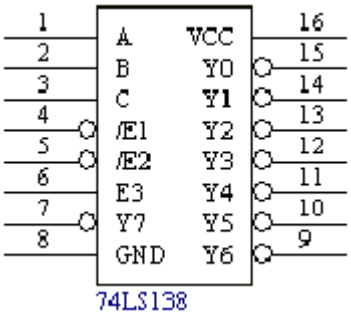
附录一　　实验用芯片介绍

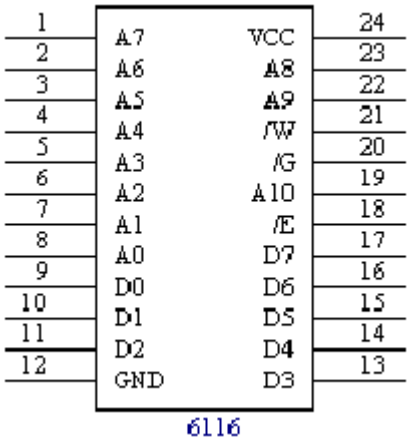

1 IN0/CLK
2 IN1
3 IN2
4 IN3
5 IN4
6 IN5
7 IN6
8 IN7
9 IN8
10 GND
VCC 20
IO7 19
IO6 18
IO5 17
IO4 16
IO3 15
IO2 14
IO1 13
IO0 12
IN9//OE 11
16V8


1 DIR
2 A0
3 A1
4 A2
5 A3
6 A4
7 A5
8 A6
9 A7
10 GND
VCC 20
/E 19
B0 18
B1 17
B2 16
B3 15
B5 14
B5 13
B6 12
B7 11
74LS245

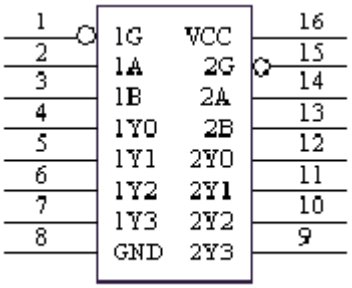

1 1A
2 1B
3 1Y
4 2A
5 2B
6 2Y
7 GND
VCC 14
4B 13
4A 12
4Y 11
3B 10
3A 9
3Y 8
74LS32


1 /OC
2 1D
3 2D
4 3D
5 4D
6 5D
7 6D
8 7D
9 8D
10 GND
VCC 20
1Q 19
2Q 18
3Q 17
4Q 16
5Q 15
6Q 14
7Q 13
8Q 12
CLK 11
74LS574

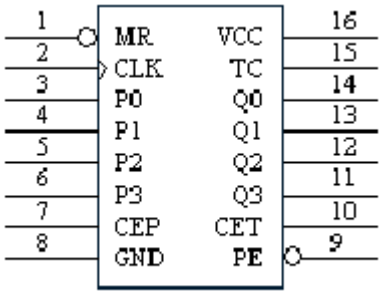

1 A
2 B
3 C
4 /E1
5 /E2
6 E3
7 Y7
8 GND
VCC 16
Y0 15
Y1 14
Y2 13
Y3 12
Y4 11
Y5 10
Y6 9
74LS138



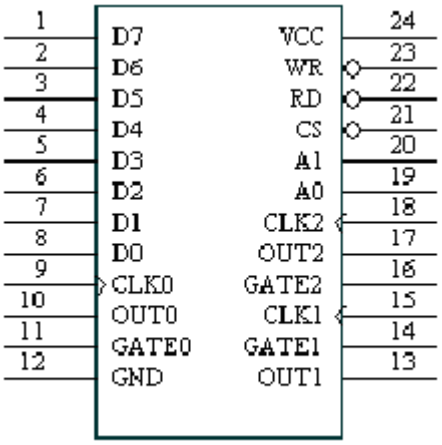
6116



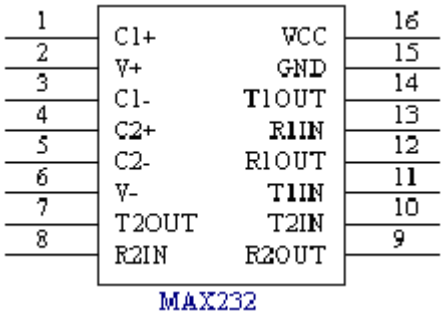
74LS139



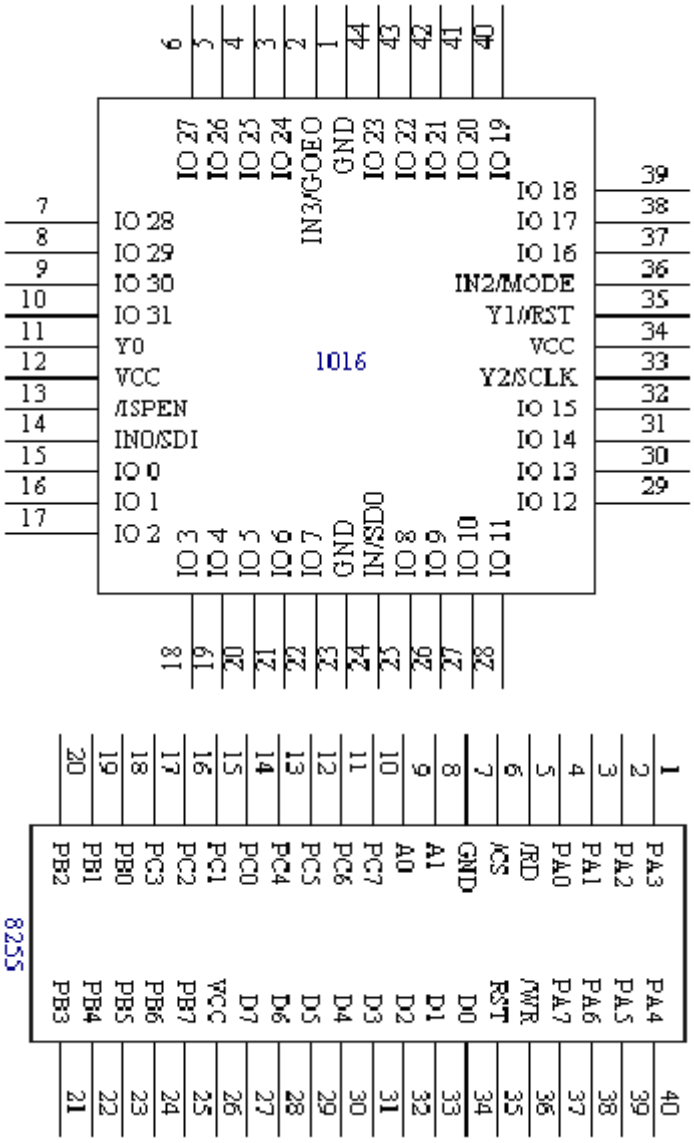
74LS161



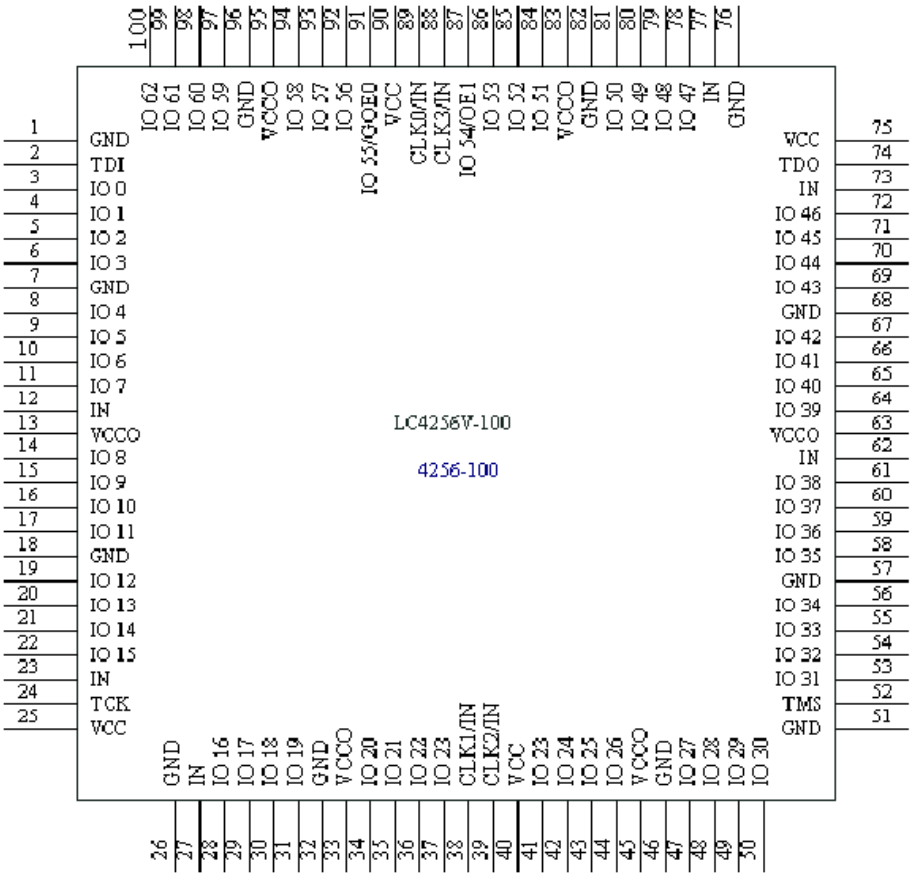
8253



MAX232


1016
IO 27
IO 26
IO 25
IO 24
IN3/GOE0
GND
IO 23
IO 22
IO 21
IO 20
IO 19
IO 18
IO 17
IO 16
IN2/MODE
Y1/RST
VCC
Y2/SCLK
IO 15
IO 14
IO 13
IO 12
IO 28
IO 29
IO 30
IO 31
Y0
VCC
/ISPEN
IN0/SDI
IO 0
IO 1
IO 2
IO 3
IO 4
IO 5
IO 6
IO 7
GND
IN1/SDO
IO 8
IO 9
IO 10
IO 11

8255
PA3
PA2
PA1
PA0
/RD
/CS
GND
A1
A0
PC7
PC6
PC5
PC4
PC0
PC1
PC2
PC3
PB0
PB1
PB2
PA4
PA5
PA6
PA7
/WR
RST
D0
D1
D2
D3
D4
D5
D6
D7
VCC
PB7
PB6
PB5
PB4
PB3


LC4256V-100
4256-100
1 GND
2 TDI
3 IO 0
4 IO 1
5 IO 2
6 IO 3
7 GND
8 IO 4
9 IO 5
10 IO 6
11 IO 7
12 IN
13 VCCO
14 IO 8
15 IO 9
16 IO 10
17 IO 11
18 GND
19 IO 12
20 IO 13
21 IO 14
22 IO 15
23 IN
24 TCK
25 VCC
26 GND
27 IN
28 IO 16
29 IO 17
30 IO 18
31 IO 19
32 GND
33 VCCO
34 IO 20
35 IO 21
36 IO 22
37 IO 23
38 CLK1/IN
39 CLK2/IN
40 VCC
41 IO 23
42 IO 24
43 IO 25
44 IO 26
45 VCCO
46 GND
47 IO 27
48 IO 28
49 IO 29
50 IO 30
51 GND
52 TMS
53 IO 31
54 IO 32
55 IO 33
56 IO 34
57 GND
58 IO 35
59 IO 36
60 IO 37
61 IO 38
62 IN
63 VCCO
64 IO 39
65 IO 40
66 IO 41
67 IO 42
68 GND
69 IO 43
70 IO 44
71 IO 45
72 IO 46
73 IN
74 TDO
75 VCC
76 GND
77 IN
78 IO 47
79 IO 48
80 IO 49
81 IO 50
82 GND
83 VCCO
84 IO 51
85 IO 52
86 IO 53
87 IO 54/OE1
88 CLK3/IN
89 CLK0/IN
90 VCC
91 IO 55/GOE0
92 IO 56
93 IO 57
94 IO 58
95 VCCO
96 GND
97 IO 59
98 IO 60
99 IO 61
100 IO 62

十六位机扩展实验板使用手册

第一章　CPTH+实验系统简介

一，CPTH+实验系统组成：

CPTH+是启东东疆公司最新推出的八位机带十六位机接口的计算机组成原理和系统结构实验仪。CPTH+在原 CPTH 主板的基础上增加十六位机接口，通过选配 CPT16 十六位机扩展实验板（也称 FPGA 实验板），完成十六位机计算机组成原理实验。

CPTH+主板以八位机模式，用 TTL74 系列器件+CPLD 构建模型机部件，让学生以可视方式观察 CPU 内各部件工作过程和模型机的实现。CPT16 实验板以十六位机模式，用 10 万门 EP1K100 FPGA 芯片构建模型机所有部件，并配置 64K×16 位存储器，在对八位机了解的基础上，让学生对十六位计算机组成原理有更深刻的理解，实现质的飞跃，为 FPGA 设计 CPU 打下基础。

二，CPTH+实验系统使用

CPTH+有两种使用方式：八位机和十六位机两种实验方式。

（1）八位机方式：

CPTH+主板是八位机模式，只需把通信选择开关“KT”拨向“CPTH”一侧，即可构成八位计算机组成原理和系统结构的实验系统。具体使用详见《DJ—CPTH 超强型计算机组成原理和系统结构实验指导书》。

（2）十六位机方式：

把选配的 CPT16 十六位机扩展实验板，对应插入“CPTH+主板”上的座，然后把通信选择开关“KT”拨向“CPT16”一侧，即可构成十六位计算机组成原理的实验系统。具体使用详见下面各章节的说明。

第二章 十六位机（FPGA）扩展实验板简介

一、FPGA 实验板主要功能

FPGA（DJ-CPT16）实验板主要是基于 EDA 设计的计算机组成原理的实验板，它的核心器件是 Altera 公司的 10 万门 EP1K100 的 FPGA 芯片。用该 FPGA 实验板，通过 VHDL 语言编程，可设计 16 位机的部件和模型机，学生将设计好的电路下载到 FPGA 芯片上，实现 16 位机的部件和模型机功能；也可完成其它设计性实验和课程设计实验。

二、FPGA 实验板组成

（1）AT89S52 单片机，主要用于接收 PC 机命令，完成 16 位程序存储器读写，管理模型机运行、暂停等功能。

（2）ISPLSI1032E 是逻辑控制芯片，负责单片机和模型机总线切换。

（3）EP1K100 是模型机主控芯片，相应管脚已连好，IDC2 是 EP1K100 芯片的下载接口，再配合 FPGA 实验板的 PC 机调试软件，可方便地进行各种实验。

（4）IDT71V016 是 64K×16 位存储器，是模型机的程序存储器，能保存大容量程序。

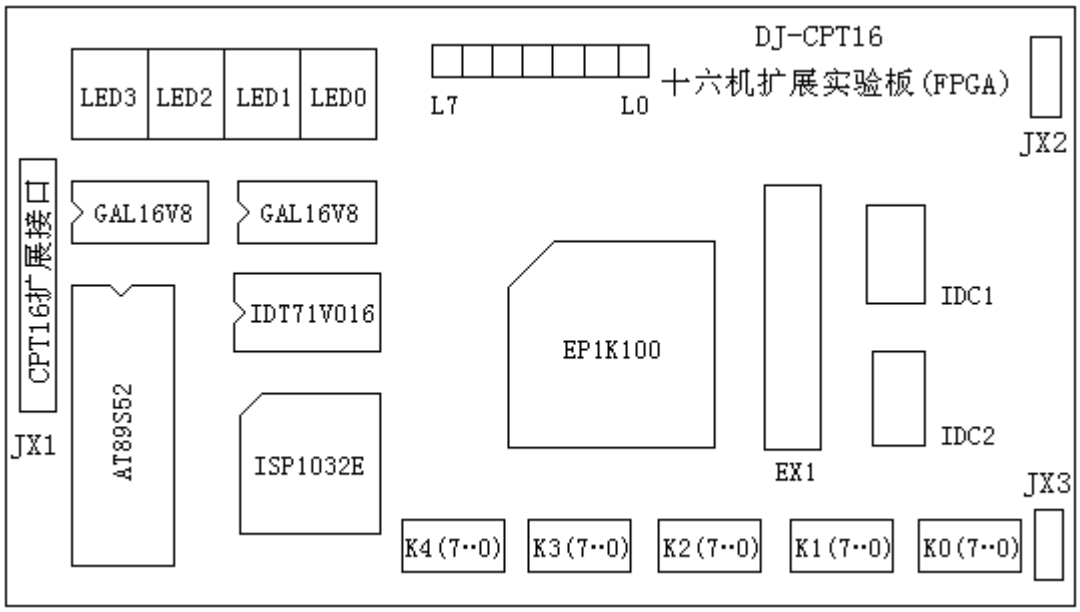
（5）四位 8 段数码管，用于显示模型机内部寄存器、总线的值，在设计时可将需要观

察的内部寄存器、总线值送 A，再通过 OUT 指令送到数码管显示。两只 GAL16V8 是四位数码管 16 进制译码。

（6）L7～L0 是 8 个发光二极管，用于显示模型机内部状态，例如：进位标志、零标志、中断申请标志等。

（7）K0（7…0）～K4（7…0）是 40 个开关，用于输入外部信号，例如，在做单步实验时，这些开关可用来输入地址总线值、数据总线值、控制信号等。

(8) EP1K100 右方的EX1座表示38 个扩展的IO 信号,当实验中需要另外的输入输出脚时可以使用这些扩展脚。



实验板框图

(9)EP1K100 的管脚连接表

K0.0	120	K3.0	86	D0	29	A8	60	OUT0	180	
K0.1	119	K3.1	85	D1	38	A9	62	OUT1	196	LDE0
K0.2	116	K3.2	83	D2	31	A10	58	OUT2	197	
K0.3	115	K3.3	80	D3	40	A11	61	OUT3	198	
K0.4	114	K3.4	78	D4	37	A12	41	OUT4	199	
K0.5	113	K3.5	75	D5	39	A13	44	OUT5	200	LDE1
K0.6	112	K3.6	74	D6	54	A14	45	OUT6	202	
K0.7	111	K3.7	73	D7	53	A15	46	OUT7	203	
K1.0	104	K4.0	71	D8	56	CS	36	OUT8	204	
K1.1	103	K4.1	70	D9	57	WR	47	OUT9	205	LED2
K1.2	102	K4.2	69	D10	55	RD	13	OUT10	206	

K1.3	101	K4.3	68	D11	24	BH	14	OUT11	207	
K1.4	100	K4.4	67	D12	19	BL	15	OUT12	208	
K1.5	99	K4.5	65	D13	18			OUT13	7	LED3
K1.6	97	K4.6	64	D14	17			OUT14	8	
K1.7	96	K4.7	63	D15	16			OUT15	9	
K2.0	95	L0	195	A0	30			RST	176	
K2.1	94	L1	193	A1	27			CK	177	
K2.2	93	L2	192	A2	25			I-REQ	179	
K2.3	92	L3	191	A3	28					
K2.4	90	L4	190	A4	26					
K2.5	89	L5	189	A5	10					
K2.6	88	L6	187	A6	11					
K2.7	87	L7	186	A7	12					

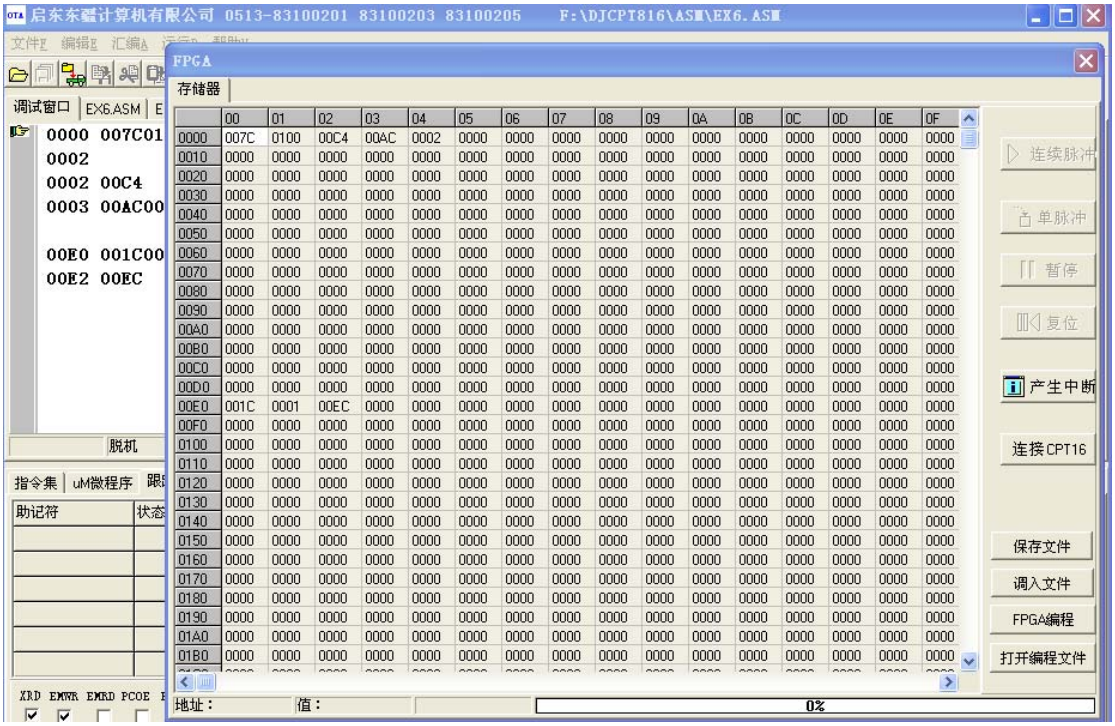
三、FPGA 实验板软件简介

双击“CPTH 计算机组成原理”快捷图标，在 CPTH 主界面上，按“打开 FPGA 扩展窗口”按钮，即可打 FPGA 扩展板界面，此界面给出了存储器窗口和功能按键，如下图。

①存储器窗口给出了 64K×16 位存储器内容，用鼠标左键单击存储单元，数据变兰色后，即可修改存储器中的数据。

②八个功能按键

- [连续脉冲]，为模型机提供连续的高频脉冲信号。
- [单脉冲]，为模型机提供单脉冲信号。
- [暂停]，当模型机连续运行时，可暂停程序的运行。
- [复位]，使模型机复位。
- [产生中断]，为模型机提供中断信号源。
- [连接 CPT16]，使 FPGA 实验板和 PC 机 CPTH 调试软件联接。
- [保存文件]，是将存储器窗口中的数据写到磁盘上。
- [调入文件]，从磁盘上读入数据文件，在此窗口中显示。
- [FPGA 编程]，是将 Quartus Ⅱ开发环境中生成的*.rbf 格式文件下载到 EP1K100 的芯片中,在下载过程中有下载进度显示。
- [打开编程文件]，从磁盘上读入*.rbf 格式文件。



四、FPGA 实验板使用

（1）用串口/USB 通信线（出产配置为串口），连接 CPTH+主板和 PC 机相应接口，把选配的 CPT16 十六位机扩展实验板，对应插入“CPTH+主板”上的座，然后把通信选择开关“KT”拨向“CPT16”一侧，接通 CPTH+实验仪电源。

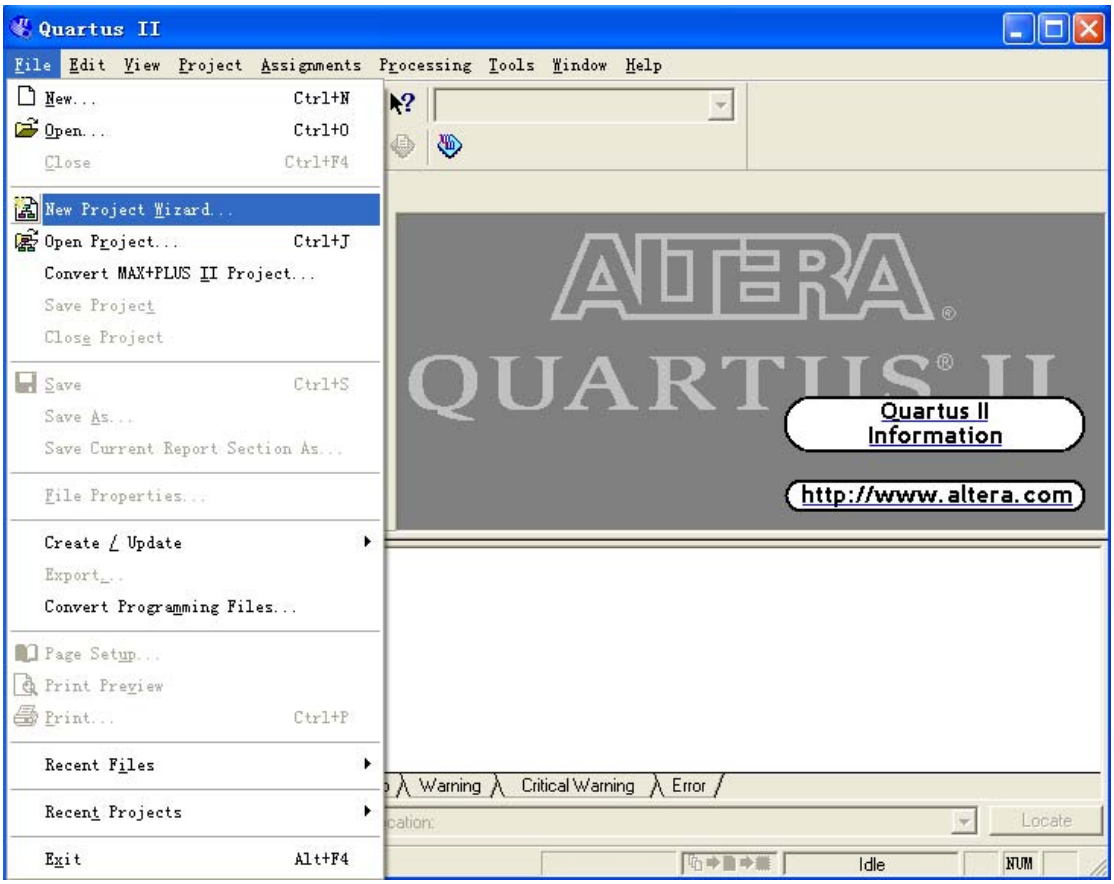
（2）在 Quartus Ⅱ软件中，用 VHDL 语言设计模型机电路，然后编译、引脚锁定，再编译生成*.sof,再转化为*.rbf 格式。如已有*.rbf,步骤（2）可省。

（3）双击“CPTH 计算机组成原理”快捷图标，在 CPTH 主界面上，按“打开 FPGA 扩展窗口”按钮，单击按键“连接 CPT16”，连机后其余按键字体变黑色可用，在 CPTH 主界面上点击打开文件图标，例如选择“EX6.asm”，点击编译下载，下载完毕后，把源文件生成的机器码自动下载到 16 位存储器中；然后点击“打开编程文件”，选择 cpt16.rbf 文件，再点击“FPGA 编程”，把模型机电路的 cpt16.rbf 下载到 EP1K100 的芯片中。也可用《第三章 Quartus Ⅱ开发环境使用入门》中器件编程方法下载。

（4）点击“连续脉冲”按键，程序连续运行，点击“产生中断”按键，看 FPGA 实验板数码管显示应加“1”变化，点击“暂停”按键，可使程序暂停运行。

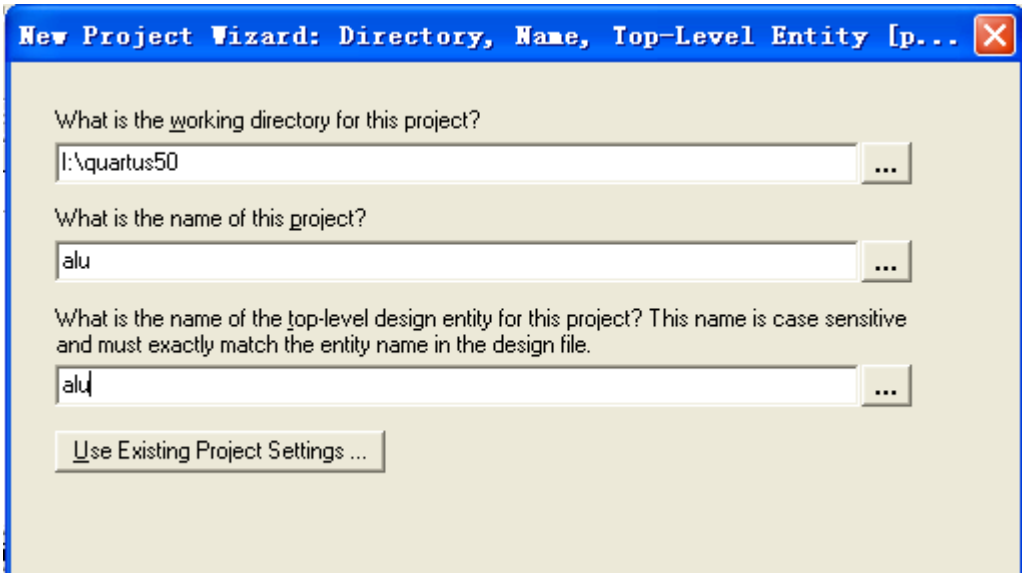
第三章 QuartusⅡ开发环境使用入门

十六位机FPGA 扩展实验板上用的10 万门大规模可编程芯片ep1k100 是altera 公司产品，altera 公司是世界上最大的FPGA/EPLD 生产商之一，其 QuartusⅡ5.0 开发环境也是电子工程师最常用的软件之一，在QuartusⅡ5.0 开发环境中可以完成设计输入(原理图、VHDL)、设计编译、设计校验、编程下载等EDA 设计的所有步骤。以QuartusⅡ5.0为例我们来了解在开发环境下EDA 设计的流程

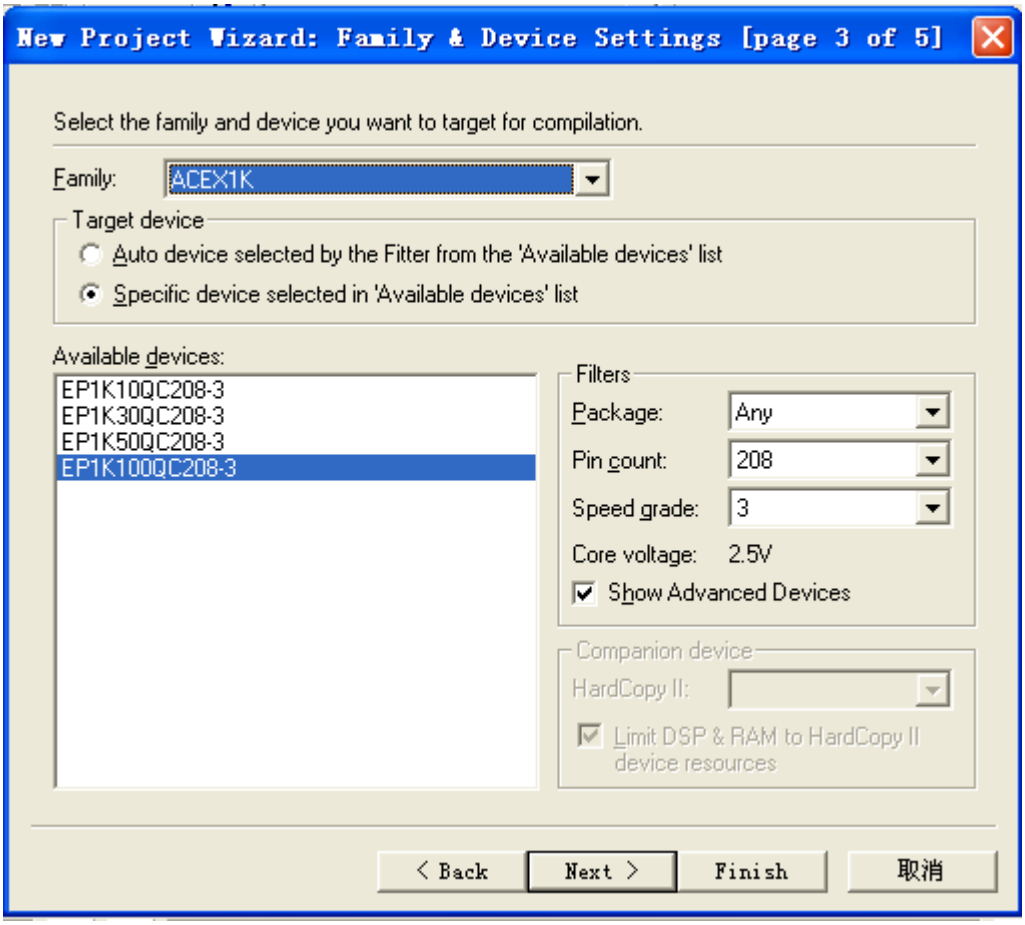


1、建立设计项目。

启动软件，打开QuartusⅡ5.0 开发环境，选择菜单[File]的[New Project…]功能，点击“NEXT”出现如图对话框，在…Name 框内填上项目名例如alu。

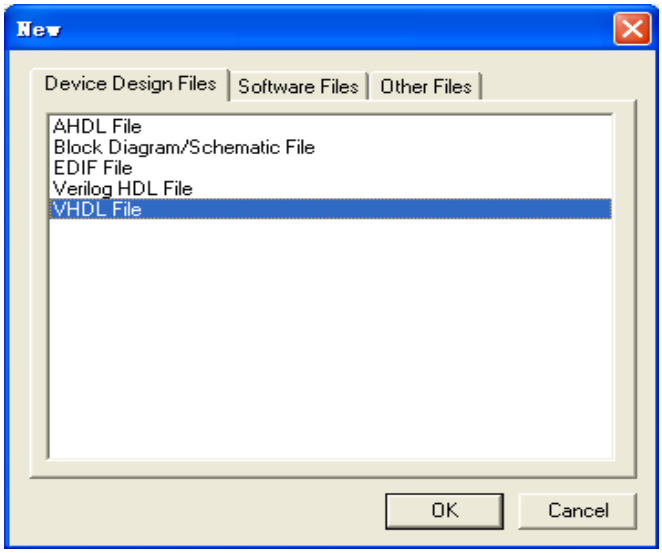


点击二次“NEXT”，选择器件，点击二次“NEXT”完成项目名创建。

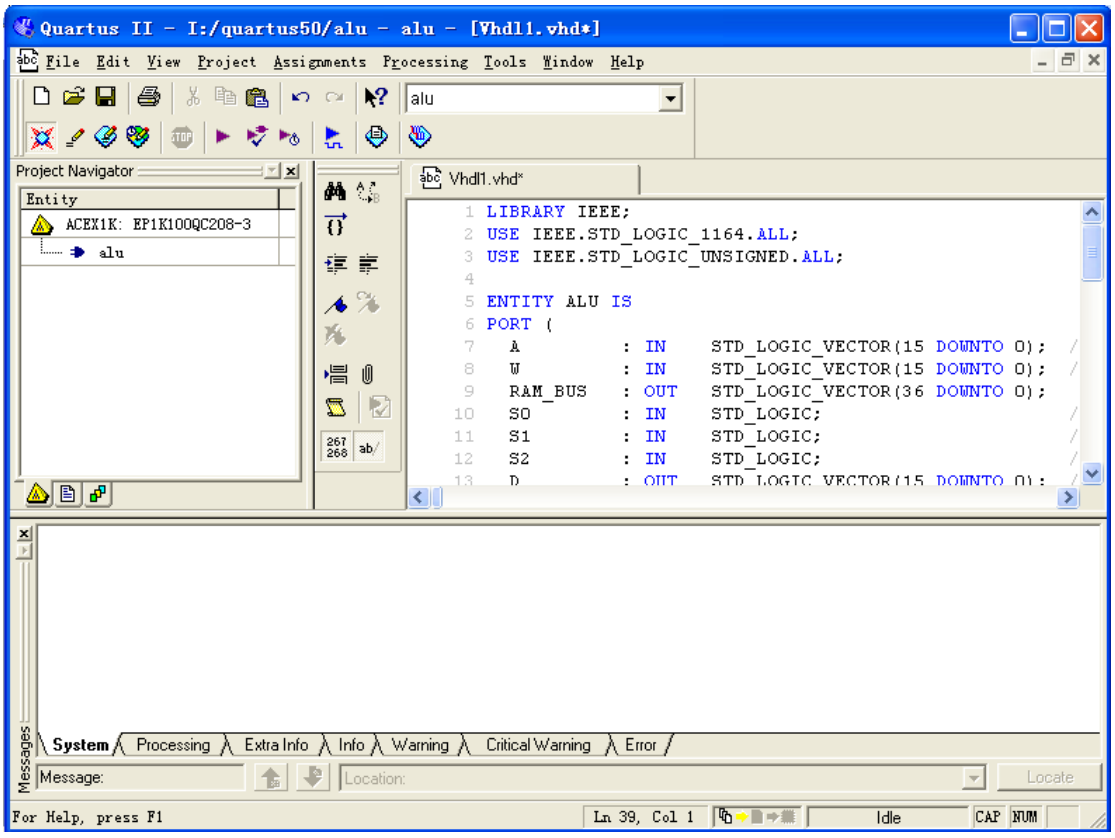


2、建立设计文件。

输入VHDL 程序。选择菜单[File]的[New]功能，选择“VHDL FILE”，点击“OK”。



在空的文本窗口中输入VHDL 语言,文件名保存为alu.vhdl。

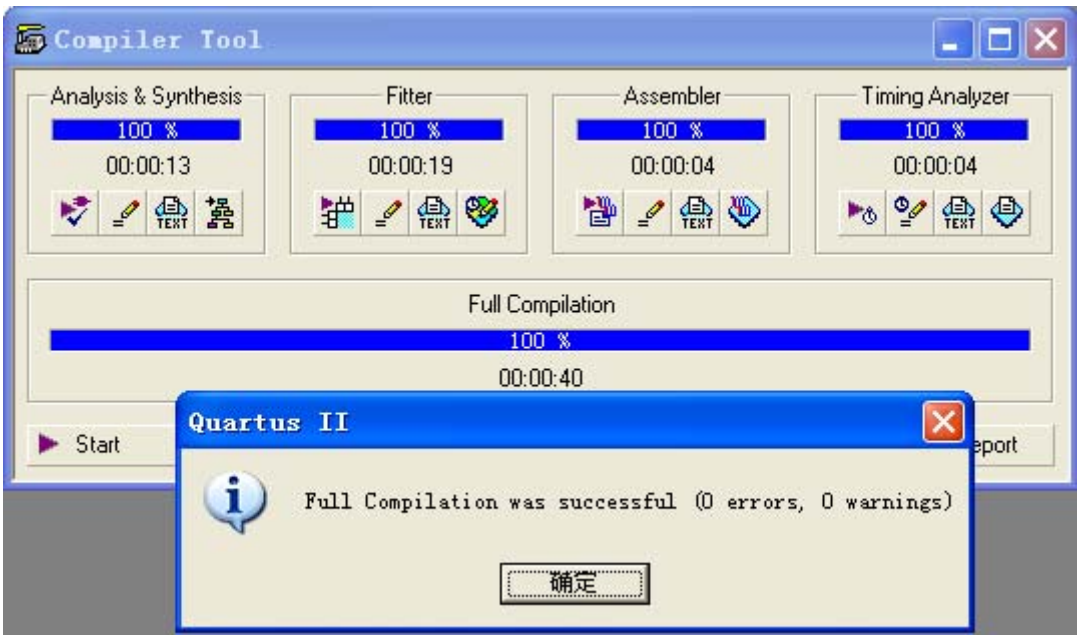


3、把文件添加到项目中

选择菜单[project]的[add/remove…]功能,选择文件“alu.vhdl”，把 alu.vhdl 文件添加到 alu 项目中。

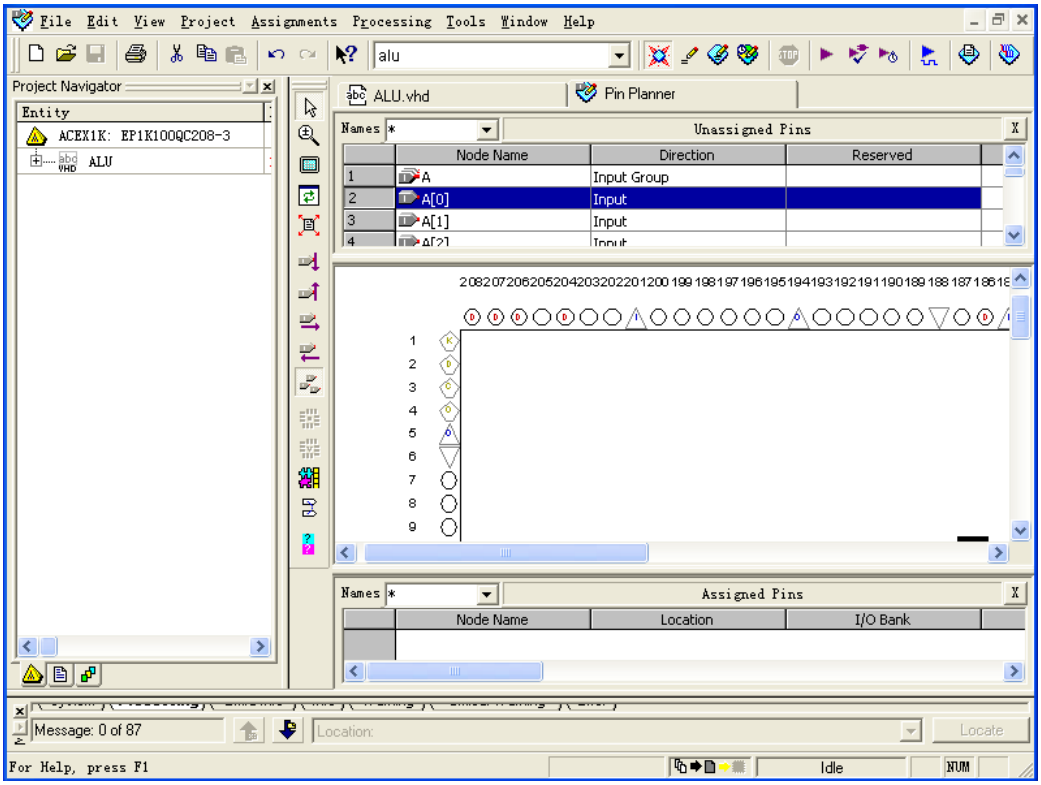
4、项目编译

选择菜单[tools]的[compiler tool]功能，点击 “start”开始编译当前 alu 项目，编译后如下图。



5、管脚锁定

选择菜单[assignments]的[pin planner]功能。例如把选中的信号名:A[0]拖拉到 ep1k100 引脚图相应引脚号上,以此类推进行锁定。锁完后重新编译。

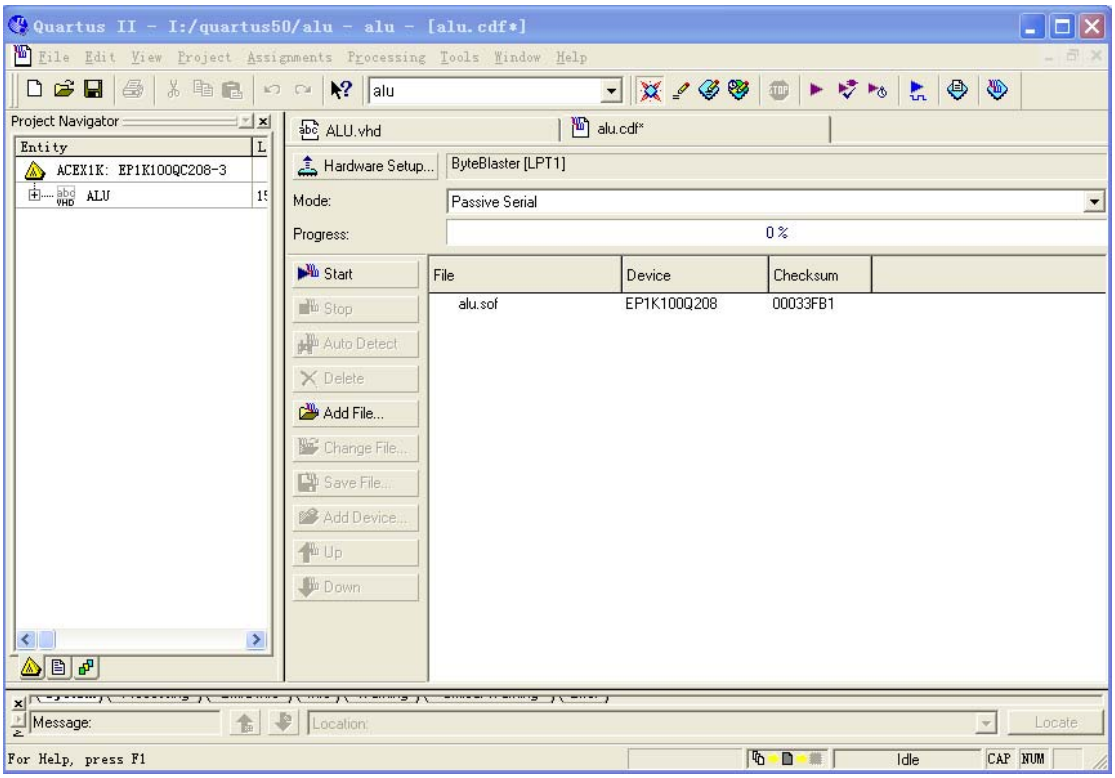


6、器件编程

通过重新项目编译后生成的*.SOF 文件用于下载。在我们例子中使用的是 EP1K100 为 ACEX1K 系列，下面我们对其进行编程：

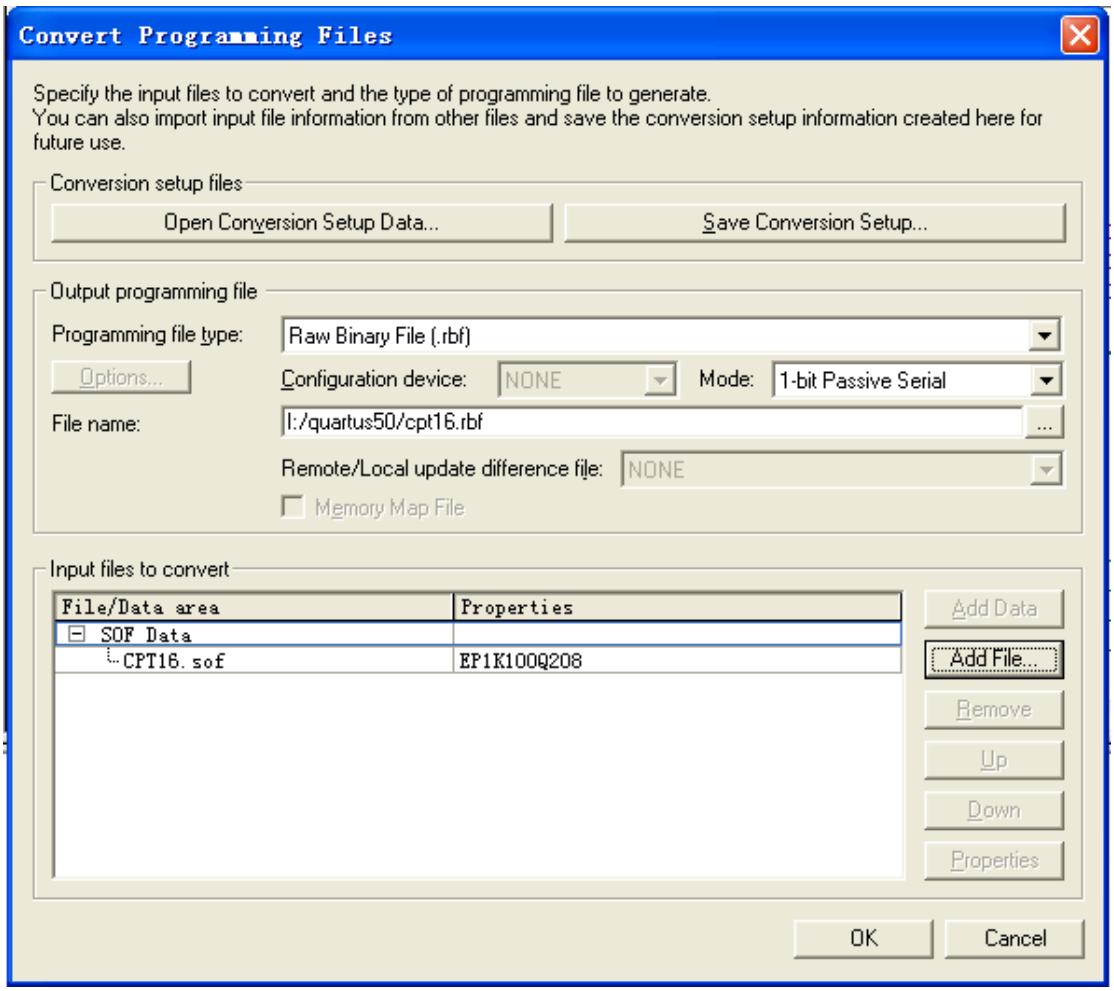
a、先关闭实验板电源，将下载电缆一端插入 LPT1（并行口，打印机口），另一端插入 CPT16 位机（FPGA）实验板的 IDC2 座，打开实验板电源；

b、从菜单“TOOLS”下选择“Programmer”，点击“Hardware serup”选择“ByteBlaster(LPT1)”，编程模式 Mode 选择“P… S…”，然后点击“Start”开始编程。到目前为止，我们已完成一个完整的设计。



7,编程文件格式转换

点击 file\convert prog...后，按下图设置，设置完后点击“ok”,把*.sof*转换.rbf。


Convert Programming Files
Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.
Conversion setup files
Open Conversion Setup Data...
Save Conversion Setup...
Output programming file
Programming file type:
Raw Binary File (.rbf)
Options...
Configuration device:
NONE
Mode:
1-bit Passive Serial
File name:
I:/quartus50/cpt16.rbf
Remote/Local update difference file:
NONE
Memory Map File
Input files to convert
File/Data area
Properties
SOF Data
CPT16.sof
EP1K100Q208
Add Data
Add File...
Remove
Up
Down
Properties
OK
Cancel

第四章 十六位机扩展实验

分部实验一、十六位ALU 实验

实验要求： 用CPT16 的扩展实验板上的开关做为输入、八段数码管做为输出，用VHDL 语言编写程序，下载到EP1K100 FPGA 中，实现十六位模型机的ALU 功能。

实验目的： 了解十六位模型机中算术、逻辑运算单元（ALU）的工作原理和实现方法学习用VHDL 语言描述硬件逻辑。学习使用EDA 开发环境。

实验说明：

在**ALU.VHD**程序中，A 为累加器，W 为工作寄存器，均为十六位寄存器，W 的值由K1、 K0 两组开关共十六位输入，A 的值由K2 、K3 两组开关输入，S0、S1、S2 为运算控制位，接在K4开关组的第0、1、2 个开关上，根据S0、S1、S2 的不同，ALU 实现的功能参见下表，D为运算结果输出显示在四位八段管LED0..LED3上，CIn 为进位输入，由K4 开关组的第3 个开关输入，COut 为进位输出，用发光二极管L0 显示其状态。

S2	S1	S0		ALU 实现的功能
0	0	0	D = A+W	运算结果为A 加W
0	0	1	D = A-W	运算结果为A 减W
0	1	0	D = A|W	运算结果为A 逻辑或W
0	1	1	D = A&W	运算结果为A 逻辑与W
1	0	0	D = A+W+CIn	运算结果为A 加W 加进位
1	0	1	D = A-W-CIn	运算结果为A 减W 减进位
1	1	0	D = ~A	运算结果为A 取反
1	1	1	D = A	运算结果为A A 直接输出

实验步骤：

1、打开**Quartus** 的**EDA** 开发环境，选择“**Open Project ...**” 打开“**EP1K100**” 下的**ALU** 项目（**Quartus** 的**EDA** 开发环境的使用可参见第三章）。

2、充分理解**ALU.VHD**（源程序见随机光盘）。了解模型机中**ALU** 的实现原理。

3、对**ALU** 项目进行综合/编译，生成**ALU.SOF** 文件，点击“programmer”，选择“ps”模式编程，点击“start”即可下载。也可用CPTH的开发环境中**FPGA** 扩展板的界面上的“FPGA编程”按钮下载**ALU.rbf**。

4、打开CPTH的开发环境，打开**FPGA** 扩展板的界面。

5、按“**连接**CPT16”键，将实验板连接到计算机上。

6、拨动**K0** 、**K1** 输入**W** 的值，拨动**K2**、 **K3** 输入**A** 的值，拨动**K4** 的**0**、**1**、**2** 位设置运算方式，拨动**K4** 的第**3** 位，设置进位，观察八段管LED3-LED0 的运算结果，观察发光二极管**L0** 是否有进位输出。

分部实验二、十六位寄存器实验

实验要求： 用CPT16 的扩展实验板上的开关做为输入、八段数码管做为输出，用VHDL语言编写程序，下载到EP1K100FPGA 中，实现十六位模型机的寄存器输入输出功能。

实验目的： 了解十六位模型机中寄存器的工作原理和实现方法。学习用VHDL 语言描述硬件逻辑。学习使用EDA 开发环境。

实验说明：

在**REG.VHD**程序中，D 为输入数据，R 为寄存器，均为十六位寄存器，D 的值由K1、K0 两组开关共十六位输入，R 为内部寄存器，用八段管LED3-LED0 显示，EN 为寄存器选通信号，接在K4开关组的第0 个开关上，RST 为复位信号，接在开关组K4 的第7 个开关上，CLK 为时钟脉冲，由FPGA界面上“**单脉冲**” 按键提供。

实验步骤：

1、打开**Quartus** 的**EDA** 开发环境选择“**Open Project ...** ”打开“**EP1K100**” 下的**REG** 项目（**Quartus** 的**EDA** 开发环境的使用可参见第三章）。

2、充分理解**REG.VHD**（源程序见随机光盘），了解寄存器的实现原理。

3、对**REG** 项目进行综合/编译生成**REG.sof** 文件，点击“programmer”，选择“ps”模式编程，点击“start”即可下载。也可用CPTH的开发环境中**FPGA** 扩展板的界面上的“FPGA编程”按钮下载**REG.rbf**。

4、打开CPTH的开发环境，打开**FPGA** 扩展板的界面。

5、按“**连接CPT16**”键将实验板连接到计算机。

6、拨动K4 的第7 位到“**1**”的位置，输出“**复位**”信号，观察八段管LED3…LED0 （R寄存器的输出显示）是否清零，再将K4 的7 位回到 “**0**” 位时，拨动K0 、K1 输入D 的值，拨动K4 的0 位，设置寄存器选通信号“**EN**”为有效状态（“**0**” 有效），按动FPGA界面上“**单脉冲**” 按键，产生一个时钟信号，观察八段管LED3 …LED0 ，看看是否将D的值存入寄存器R 中并显示出来，拨动开关组K1、 K0 改变D 的值，再将“**EN**”置于无效状态（“1” 位置），按“**单脉冲**” 产生时钟信号，观察八段管LED3 …LED0 是否会随着改变。

分部实验三、十六位寄存器组实验

实验要求： 用CPT16 的扩展实验板上的开关做为输入、八段数码管做为输出，用VHDL语言编写程序，下载到EP1K100FPGA 中，实现十六位模型机的多个寄存器输入输出功能。

实验目的： 了解十六位模型机中寄存器组的工作原理和实现方法。学习用VHDL 语言描述硬件逻辑。学习使用EDA 开发环境。

实验说明：

在REGS.VHD程序中，D 为输入数据，为十六位寄存器，D 的值由K1、 K0 两组开关

共十六位输入，R0..R3 为内部寄存器，R 用做内部寄存器显示输出，用八段管LED3..LED0显示其值，SA、SB 为寄存器选择控制信号，接在K4 开关组的第0、1 个开关上，RD 为寄存器读信号，接K4的第2 个开关上，WR 为寄存器写信号，接在K4 的第3 个开关上， RST 为复位信号，接在开关组K4 的第7 个开关上，CLK 为时钟脉冲，由FPGA界面上“**单脉冲**”按键提供。

实验步骤：

1、打开**Quartus** 的**EDA** 开发环境，选择“**Open Project ...** ”打开“**EP1K100** ”下的**REGS** 项目（**Quartus** 的**EDA** 开发环境的使用可参见第三章）。

2、充分理解**REGS.VHD**（源程序见随机光盘）。了解模型中寄存器组的实现原理。

3、对**REGS** 项目进行综合/编译，生成**REGS.SOF** 文件，点击“programmer”，选择“ps”模式编程，点击“start”即可下载。也可用CPTH的开发环境中**FPGA** 扩展板的界面上的“FPGA编程”按钮下载**REGS.rbf**。

4、打开CPTH的开发环境，打开**FPGA** 扩展板的界面。

5、按“**连接CPT16**”键将实验板连接到计算机。

6、拨动K4 的第7 位到“**1**” 的位置，输出“**复位**”信号，观察八段管LED3…LED0（R 寄存器的输出显示）是否清零，然后将K4的第2（ RD 信号）、3（WR 信号）位置成1状态，使读写信号都处于无效状态。

9、寄存器组写实验：将K4 的7 位回到时“**0**”位，拨动**K0、K1** 输入**D** 的值，拨动K4的0、1位设成“**00**”， 选择寄存器**R0**，拨动K4 的3 位，设置寄存器写信号“**WR**” 为有效状态（“**0**”有效），按动FPGA界面上“**单脉冲**” 按键，产生一个时钟信号，将**D** 写入寄存器**R0** 中。拨动**K0、K1** 开关组，改变**D** 值，再改变**K4** 的第0、1 位设成“**01**”，选择寄存器**R1** ，按“**单脉冲**”给出时钟信号，将**D** 存入寄存器**R1**中，如此将不同的十六位数据分别存**R0..R3** 寄存器中。

10、寄存器组读实验：将**K4** 的第3 位（WR 信号）设成“**1**” 使其无效，将**K4** 的0、1两位拨成“**00**”， 选择寄存器R0， 再将K4 的第2 位（**RD** 信号）设成“**0**”，读出**R0** 中的数据，并输出到八段管**LED3…LED0** 上显示。拨动**K4** 的第2 位，使**RD** 信号无效，改变**K4** 的0、1 两位选择寄存器**R1**， 再拨动**K4** 的第2 位使**RD** 信号有效，读出R1 的值并显示在八段管**LED3…LED0** 上。如此，读出寄存器**R2、R3** 的值，并观察与写入的数据是否相同。

分部实验四、十六位指令计数器PC 实验

实验要求： 用CPT16 的扩展实验板上的开关做为输入、八段数码管做为输出，用VHDL语言编写程序，下载到EP1K100FPGA 中，实现十六位模型机的指令计数器功能。

实验目的： 了解十六位模型机中指令计数器（PC）的工作原理和实现方法。学习用VHDL语言描述硬件逻辑。学习使用EDA 开发环境。

实验说明： 下为十六位模型机中PC 的VHDL 语言：

在PC.VHD程序中，D 为十六位输入数据，用于表示跳转条件满足时，跳转的目标地址。D 的值由K1、K0 两组开关输入。PC 为指令计数器（PC），用八段数码管LED3…LED0显示其值。 C、Z 接在K4 开关组的第0、1 个开关上，用于模拟模型机中的进位标志和零标志信号，ELP 接K4 的第2 个开关上，为程序跳转控制信号，为“1” 时不允许预置PC，为“0” 时，根据指令码的第3、2 位和C、Z 状态来控制程序是否跳转（见下表说明）。PC1 接在K4 的第3 个开关上，表示PC 加1 控制信号。 IR2、IR3 接在K4 的第4、5 个开关上，表示程序指令的第2 位和第3 位，在本模型机实验中，这两用于控制程序的跳转（见下表说明）。RST 为复位信号，接在开关组K4 的第7 个开关上，CLK 为时钟脉冲，由FPGA界面上“**单脉冲**” 按键提供。

ELP	IR3	IR2	C	Z	LDPC
1	X	X	X	X	1
0	0	0	1	X	0
0	0	0	0	X	1
0	0	1	X	1	0
0	0	1	X	0	1
0	1	X	X	X	0

上表中，LDPC 为内部信号，用于控制PC 是否能被预置。

当ELP = 1 时，LDPC = 1 ，不允许PC 被预置。

当ELP = 0 时，LDPC 由IR3 、IR2 、C、 Z 确定。

当IR3 、IR2 = 1X 时，LDPC = 0， D 的值在CLK 上升沿打入PC，实现程序的JMP（ 直接跳转）功能。

当IR3 、IR2 = 00 时，LDPC = C 取反，当C = 1 时，D 的值在CLK 上升沿打入PC实现程序的JC（有进位跳转）功能。

当IR3 、IR2 = 01 时，LDPC = Z 取反，当Z = 1 时，D 的值在CLK 上升沿打入PC实现程序的JZ（累加器为零跳转）功能。

本实验中，RST = 1 时，指令计数器PC 被清0， 当LDPC = 0 时，在CLK 上升沿D 的值打入PC， 当PC1 = 1 时，在CLK 上升沿PC 加1。

实验步骤：

1、打开Quartus 的EDA 开发环境，选择“**Open Project ...** ”打开“**EP1K100** ”下的PC 项目（Quartus 的EDA 开发环境的使用可参见第三章）。

2、充分理解PC.VHD（源程序见随机光盘），了解模型机中指令计数器的实现原理。

3、对PC 项目进行综合/编译，生成PC.SOF 文件，点击“programmer”，选择“ps”模式编程，点击“start”即可下载。也可用CPTH的开发环境中**FPGA** 扩展板的界面上的“FPGA编程”按钮下载**PC.rbf**。

4、打开CPTH的开发环境，打开FPGA 扩展板的界面。

5、按“**连接CPT16**”键将实验板连接到计算机。

6、拨动K4 的第7 位到“**1**” 的位置输出“**复位**”信号，使电路处于复位状态。观察八段管LED3…LED0（PC 输出显示）是否清零。

7、PC+1 实验：将K4 的第2 （ELP 信号）、3 （PC1 信号）位置成“**1**” 状态，使跳转控制信号处于PC+1 状态。再将K4 的7 位回到时“**0**” 位，退出“**复位**”状态，按动FPGA界面上“**单脉冲**” 按键，产生一个时钟信号，观察八段管LED3… LED0 的显示，看看PC 是否加1， 再给出单脉冲，观察PC 是否再次加1。

8、直接跳转实验：使拨动K0、 K1 输入D 的值，设置跳转的目标地址，拨动K4 的第2位设成“**0**”， 使ELP 信号为低，拨动K4 的5 位，将其设成“**1**” 使IR3 为高，将跳转控制设成“**直接跳转**”方式，按动FPGA界面上“**单脉冲**” 按键，产生一个时钟信号，将D 写入PC 中，观察八段管LED3…LED0， 看看PC 是否转到目标地址。

9、条件跳转实验1：拨动K4 的第4、5 位，使IR2、 IR3 置成“**00**”， 将跳转控制设置成“**判进位跳转**”方式。将K4 的0 位设置成“**1**”， 表示有进位，按动FPGA界面上“**单脉冲**” 按键，产生一个时钟信号，观察八段管LED3…LED0 ，看看PC 是否转到D 所指定的目标地址。再将K4 的0 拨成“**0**”， 表示无进位，给出一个单脉冲，观察LED3…LED0， 看看PC 是否加1。

10、条件跳转实验2： 拨动K4 的第5、4 位，使IR3、IR2 置成“**01**”， 将跳转控制设置成“**判零跳转**”方式。将K4 的1 位设置成“**1**”， 表示累加器为零，按动FPGA界面上“**单脉冲**” 按键，产生一个时钟信号，观察八段管LED3…LED0， 看看PC 是否转到D 所指定的目标地址。再将K4 的1 拨成“**0**”， 表示累加器不为零，给出一个单脉冲，观察LED3…LED0， 看看PC是否加1。

分部实验五、中断控制实验

实验要求： 用CPT16 的扩展实验板上的开关做为输入、八段数码管做为输出，用VHDL语言编写程序，下载到EP1K100FPGA 中，实现十六位模型机中断控制功能。

实验目的： 了解十六位模型机中中断控制的工作原理和实现方法。学习用VHDL 语言描述硬件逻辑。学习使用EDA 开发环境。

实验说明：

在INT.VHD程序中，IREN 接K4 的第0 位，表示程序执行过程中的取指令操作，中断请求信号只有在此信号有效时（取指令时）才会被响应。IENT 接K4 的第1 位，用于在中断返回时，清除中断请求寄存器和中断响应寄存器。ICEN 为输出信号接发光二极管L2，此信号用于控制读中断指令。ACK 接发光二极管L1， 显示中断响应信号。REQ 接发光二极管L0，显示中断请求信号。RST 为复位信号，接在开关组K4 的第7 个开关上，I_REQ 为中断申请输入信号，由FPGA界面上“**产生中断**”按键输出到FPGA 扩展板上，CLK 为时钟脉冲，由FPGA界面上“**单脉冲**” 按键提供。

实验步骤：

1、打开Quartus 的EDA 开发环境，选择“**Open Project ...** ”打开“**EP1K100**” 下

的INT 项目（Quartus 的EDA 开发环境的使用可参见第三章）。

2、充分理解INT.VHD（源程序见随机光盘），了解寄存器的实现原理。

3、对INT 项目进行综合/编译生成INT.SOF 文件，点击“programmer”，选择“ps”模式编程，点击“start”即可下载。也可用CPTH的开发环境中**FPGA** 扩展板的界面上的“FPGA编程”按钮下载**INT.rbf**。

4、打开CPTH的开发环境，打开FPGA 扩展板的界面。

5、按“**连接CPT16**”键将实验板连接到计算机。

6、拨动K4 的第7 位到“**1**” 的位置输出“**复位**”信号，将K4 的第0、1 位拨到“**1**”的位置，使IREN 和EINT 都处于无效状态，将内部的中断请求寄存器，中断响应寄存器都清零，使其能响应中断，此时只有L2发光二极管亮。

7、中断申请：将K4 的7 位回到时“**0**” 位，使电路正常工作。按下FPGA界面上“**产生中断**”按键申请中断，FPGA 扩展板上的L0 发光二极管变亮，表示有中断申请。

8、中断响应：将K4 的第0 位拨成“**0**”， 也就使IREN 有效，表示取指令操作，扩展板上的L2 发光二极管变熄灭，表示已经响应中断。

9、中断处理：按动FPGA界面上“**单脉冲**” 按键，产生一个时钟信号，扩展板上L2、L1、L0 发光二极管都亮，表示取指操作取出中断处理指令来执行。

10、中断退出：将K4 的第0 位拨成“**1**”， IREN 置成无效，将K4 的第1 位拨成“**0**”，将EINT 置成有效，发光二极管L2 变亮、L1、L0 变暗，中断申请寄存器和中断响应寄存器清零，表示可以接受下一次中断申请。

分部实验六 十六位模型机的总体实验

实验要求： 用CPT16 的扩展实验板上的开关做为输入、八段数码管做为输出，以上面介绍的分部实验为基础，用VHDL 语言编写程序，下载到EP1K100FPGA 芯片中，实现一个十六位模型机，模型机要有完整的指令系统，中断处理系统、输入输出系统。

实验目的： 了解十六位模型机整机的工作原理和实现方法。用VHDL 语言描述硬件逻辑让FPGA 能实现复杂的运算、处理功能。

实验程序：

```
    MOV   A, #00H
LOOP:
    OUT
    JMP   LOOP
    ORG   0E0H
    ADD   A, #01
    RETI
    END
```

实验说明：“CPT16.VHD ”为一个完整的十六位模型机的VHDL 程序，另见随机光盘。

在总体实验中，rst 为复位信号，接在开关组K4 的第7 个开关上。i_req 为中断申请输入信号，由FPGA界面上“**产生中断**”按键输出到FPGA 扩展板上。clk 为时钟脉冲，由FPGA界面上“**单脉冲**” 按键提供。keyin 输入端口，接在扩展板的K1、K0 开关组上，用于输入外部信号。portout 为输出端口，接到LED3…LED0 四个八段数码管上，用于显示输出信号。mem_d 为十六位存储器的数据线，mem_a 为十六位存储器的地址线，mem_rd 为存储器读控制信号，mem_wr 为存储器写控制信号，mem_cs 为存储器片选信号，mem_bh 为
存储器高8 位数据选择信号，mem_bl 为存储器低8 位数据选择信号。

实验步骤：

1、打开**Quartus** 的EDA 开发环境，选择“**Open Project ...** ”打开“**EP1K100**” 下的CPT16 项目（**Quartus** 的EDA 开发环境的使用可参见第三章）。

2、充分理解**CPT16.VHD**（源程序见随机光盘）。 了解十六们模型机的实现原理。

3、对CPT16 项目进行综合/编译，生成**CPT16.sof**文件，点击“programmer”，选择“ps”模式编程，点击“start”即可下载。也可用CPTH的开发环境中**FPGA** 扩展板的界面上的“FPGA编程”按钮下载**CPT16.rbf**。

4、打开CPTH的开发环境，再打开FPGA 扩展板的界面，按“**连接CPT16**”键将实验板连接到计算机，在开发环境的主界面中，打开“**\DJCPTH\ASM**” 目录下“**EX6.ASM**”程序，编译下载“**EX6. ASM**”，下载完毕后，把源文件生成的机器码自动下载到16位存储器中。

5、在FPGA 扩展板的界面，点击“连续脉冲”按键，程序连续运行，点击“产生中断”按键，产生中断请示信号让模型机中断，在中断处理程序中，我们将累加器A 加1中断。返回后输出累加器中的内容。重复点击“产生中断” 按键，产生多个中断请示信号，让模型机多次中断，累加器多次加1送数码管显示，观察LED3…LED0输出，是否满足设计要求。点击“暂停”按键，可使程序暂停运行，点击“复位”按键，可使模型机复位。