

TEC-ZCHPC1

计算机组成原理与体系结构实验系统电脑版

实验指导书



清华大学 科教仪器厂

Tsinghua University Science & Technology Equipment Factory

目 录

第一章 实验系统简介-----	1
第二章 软件使用说明-----	6
第三章 基础汇编语言程序设计-----	19
第四章 运算器实验-----	35
第五章 存储器实验-----	52
第六章 输入输出 IO 实验-----	64
第七章 微程序控制器实验-----	71
第八章 组合逻辑控制器实验-----	86
第九章 中断实验-----	95
第十章 综合设计-----	105
第十一章 BASIC 语言程序设计-----	123

作者邮箱: bluesky_blueseas@yeah.net

个人微博: <http://weibo.com/u/1403786364>

第一章 TEC-ZCHPC1

计算机组成原理与体系结构实验系统电脑版简介

1. 软件简介

TEC-ZCHPC1 计算机组成原理与体系结构实验系统电脑版是一款计算机组成原理实验教学虚拟仿真系统。随着计算机技术的不断发展,利用虚拟现实技术实现计算机组成原理实验环境,可以有效的替代传统的计算机组成原理实验室多种硬件实验设备,使学生在计算机上就可以完成计算机组成原理与体系结构的实验,不受传统实验室硬件限制。有利于学生对计算机理论知识的巩固提高了教学时效及学生自主学习和创新能力。

TEC-ZCHPC1 计算机组成原理与体系结构实验系统电脑版由清华大学科教仪器厂研制。适用于本科、硕士研究生的计算机组成原理和计算机系统结构课程的教学实验。该系统有自己的指令系统和监控程序,采用模块化的结构(运算器、控制器、主存储器、I/O接口和中断)构成一台完整的模型计算机,支持组合逻辑控制器和微程序控制器两种控制器方案基本指令系统支持多种基本寻址方式。本系统用直观的方式表现系统内部状态,便于理解和教学,可减少操作失误,提高设计能力。

TEC-ZCHPC1支持WinXP/Win7/Win8等操作系统,是16位组成原理实验教学系统,集成了汇编器、调试器,独一无二的“动画显示”能力。拥有完善的控制功能,包括单节拍调试、单指令调试、连续运行。可以精确观察计算机在每个节拍的运行过程,对复杂结构计算机的调试非常必要。数据在动画界面上的移动和变化对理解原理和设计指令非常便利。上述实验方式及功能使得实验过程更为形象直观,具有非常良好的教学效果。

2. 实验系统的软硬件层次

软件运行	解释 BASIC 语言 扩展汇编语言支持 监控程序(指令)
硬件模拟	运算器、控制器(两种方式)模拟 存储器模拟;总线模拟;各译码器模拟 I/O接口模拟;中断响应;开关、按键、指示灯模拟
运行环境	PC机, Windows系统

3. 模拟硬件子系统

运算器：4 片 AMD 公司的 AM2901 并联，共 16 位。内部有 16 个寄存器。

控制器：第一种是微程序方式，使用 AMD 公司的 AM2910 微程序定位器。

第二种是使用硬布线方式。

存储器：地址空间 0~FFFFH，其中 0~27FFH 是 RAM，4000H~5FFFH 是 EEPROM，其它地址分别映射到这两段。

I/O 接口：两个，地址分别是 80H/81H 和 A0H/A1H，独立使用。

手动开关：32 个控制开关，16 个数据开关，4 个功能选择开关。

按键：RESET 键，START 键，单节拍，单指令按键，3 个中断按键。

4. 模拟软件的文件组成

由下列文件构成：

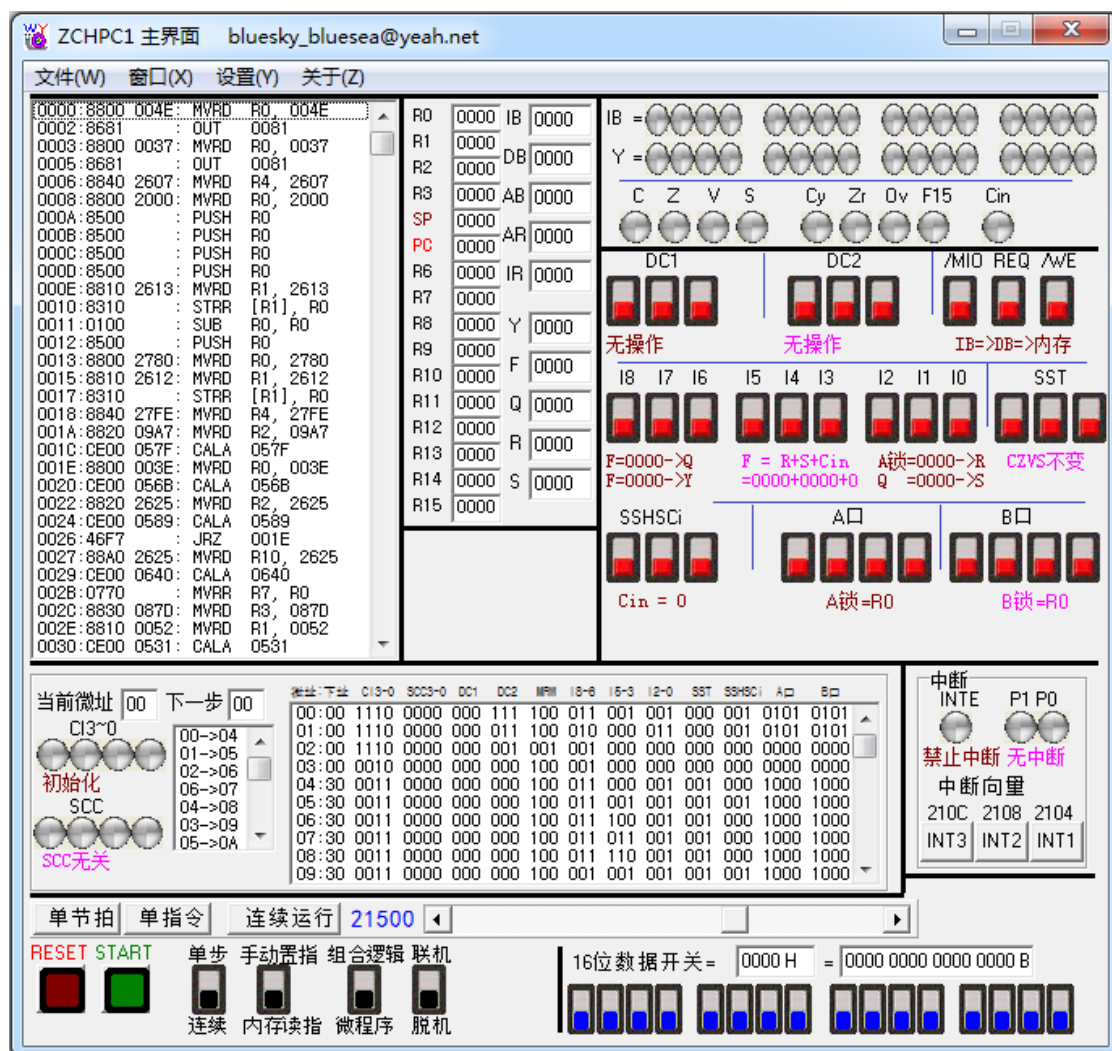
- (1) ZCHPC1.exe: 实验系统程序，运行时自动读取其它文件。
- (2) MicroIns.txt: 微程序保存文件。
- (3) InsMap.txt: 指令到微程序的入口地址映射文件。
- (4) HardWiredLogic.txt: 硬布线逻辑保存文件。
- (5) EEPROM.txt: EEPROM 的记录文件。
- (6) InsExt.txt: 扩展指令格式记录文件，供汇编和反汇编使用。

5. 主要技术指标

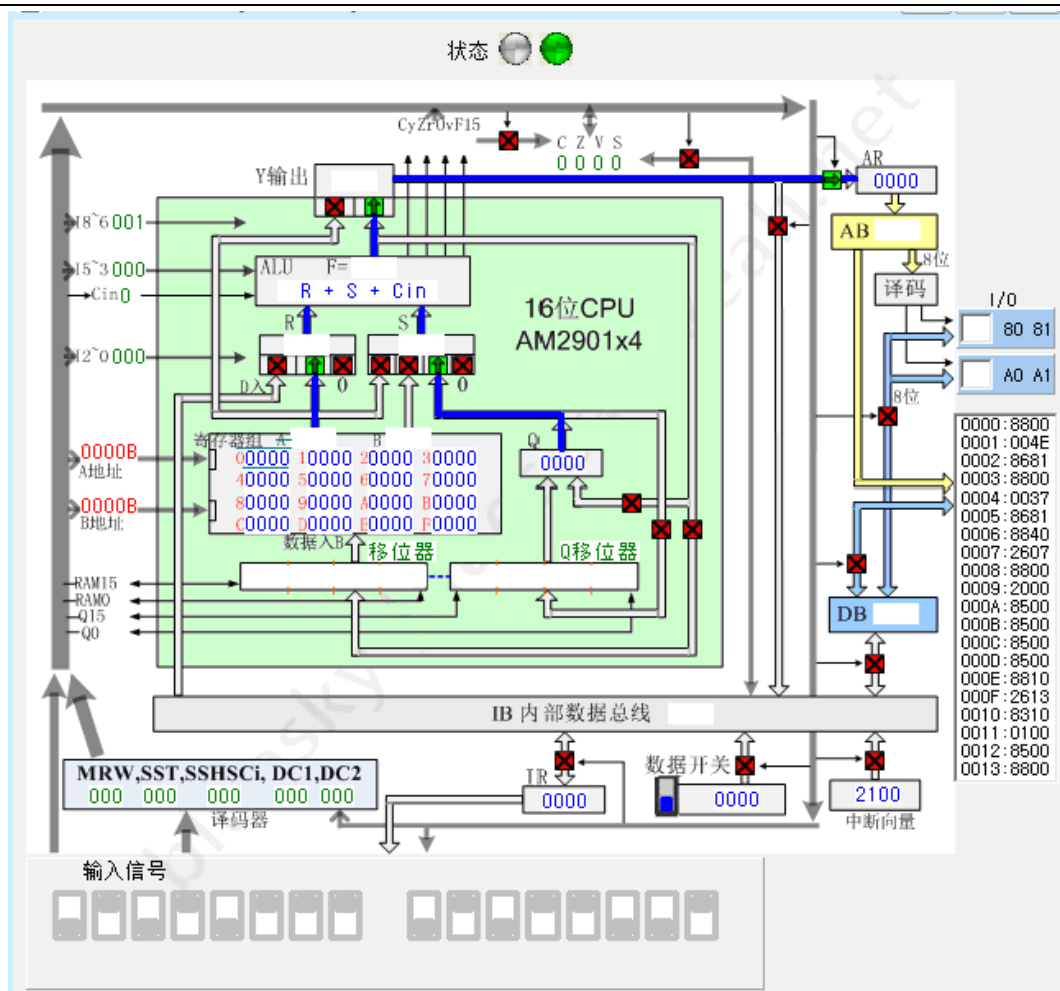
- (1) 机器字长 16 位，即运算器、主存、数据总线、地址总线均是 16 位。
- (2) 指令系统支持多种基本寻址方式。可供实验者自己设计新指令。
- (3) 存储器空间 18K 字，由容量 10K 的 RAM 和 8K 的 EEPROM 组成。
- (4) 主频可在 1Hz~500KHz 之间选择，可以手动暂停。
- (5) 运算器由 4 片位片结构器件级联而成。其内部的 ALU 可实现 8 种算术逻辑运算功能，拥有 16 个双端口读出、单端口写入的通用寄存器，和一个能自行移位的乘商寄存器 Q。设置 C（进位）、Z（结果为 0）、V（溢出）和 S（符号位）四个状态标志位。
- (6) 控制器采用微程序和硬布线两种控制方案实现，可自由选择。实验人员可方便地修改现有设计，添加若干条自己设计的新指令。
- (7) 在主界面的右下角，设置了可完成中断实验的功能区。

6. 模拟软件各界面

主要界面如下：



主界面，用来进行实验操作，控制实验运行，观察实验现象等。所有其它界面都由本窗口菜单调出。



动画界面，能以动画形式表现数据的运算过程，数据在各部件之间的移动。

```
终端1 80H 81H
TEC-2000 CRT MONITOR
Version 2.0 2001.10
Computer Architectur Lab., Tsinghua University
Copyright Jason He
>d 0
0000 8800 004E 8681 8800 0037 8681 8840 2607 ...N.....7...@&.
0008 8800 2000 8500 8500 8500 8500 8810 2613 .. .....&.
0010 8310 0100 8500 8800 2780 8810 2612 8310 .....&...
0018 8840 27FE 8820 09A7 CE00 057F 8800 003E .@'. . . . .>
0020 CE00 056B 8820 2625 CE00 0589 46F7 88A0 ...k. &%...F...
0028 2625 CE00 0640 0770 8830 087D 8810 0052 &%...@.p.0.}...R
0030 CE00 0531 0301 470B 0820 4703 CE00 012D ...1..G.. G...-
0038 41E5 CE00 06CA 4632 CE00 0186 41DF 0930 A....F2....A..0
0040 8113 0511 462B 0301 47FA 8501 8810 0008 ....F+..G.....
0048 0013 8131 8710 CE00 0696 4603 CE00 062B ...1.....F....+
0050 471D 0820 8400 8500 8501 8502 8810 27FE G.. .....
0058 8820 8000 8312 0910 8313 8720 8710 8700 .....
0060 8C00 8507 CE00 27FE 8770 8800 0047 0307 .....p...G..
0068 47B5 CE00 0563 CE00 0847 41B0 8820 0A1D G....c...GA.. ..
0070 CE00 057F 41AB 4705 8501 8810 2606 81F1 ... A.G.....&...
>
```

终端界面，进行程序编写，监控操作，调试程序，观察实验现象等。

第二章 TEC-ZCHPC1

计算机组成原理与体系结构实验系统电脑版使用说明

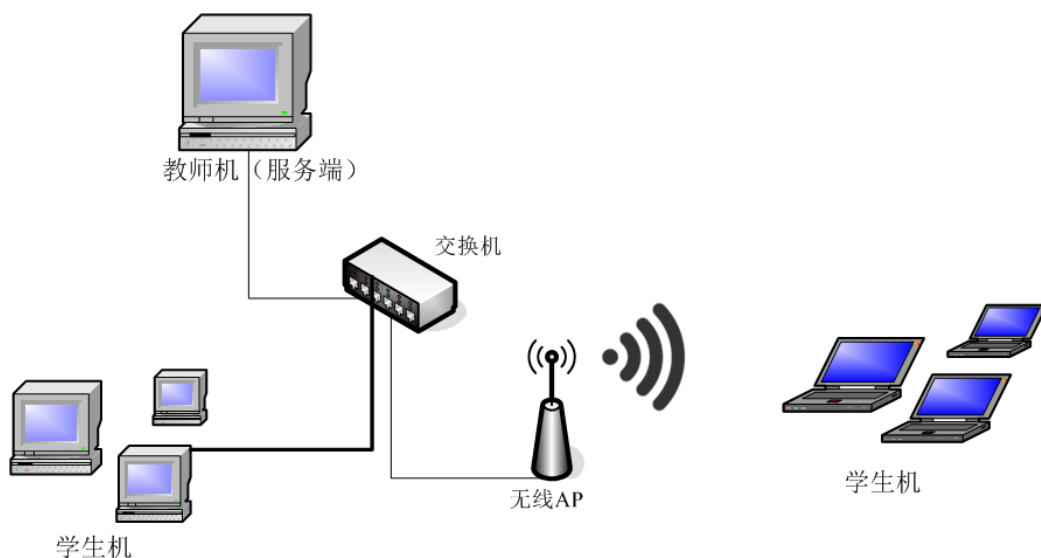
1、服务器端与学生端

TEC-ZCHPC1 计算机组成原理与体系结构实验系统电脑版包含学生端和服务端软件，其中学生端软件在学生用电脑运行；服务端软件在教师机或者服务器运行。学生端与服务端电脑都可使用台式机或者笔记本。

运行服务端软件的电脑，必须插入指定的加密狗，否则无法启动服务端软件。

学生端软件运行时，会自动连接服务端，如果连接成功，可以做实验。

教师机和学生机可以支持的操作系统有 WinXP/ Win7/Win8 等，32 位与 64 位均可。



运行环境示意图

2、实验系统总体概述

- ◆模拟软件的 CPU 是模仿的 4 片 AM2901，字长 16 位，CPU 内部有 16 个寄存器。
- ◆内存地址空间 0~FFFFH。其中 0~27FFH 是 RAM，4000~5FFFH 是 EEPROM，其它地址分别映射到这两段。
- ◆能方便的进行微指令和组合逻辑修改，设计新指令。
- ◆能大范围调节运行速度。能单步微指令和单步指令执行。
- ◆运行过程中，可观察到执行到哪条指令、哪条微指令、组合逻辑当前状态。
- ◆手动操作能完成所有功能，可以用来设计新指令。
- ◆能实时显示系统内部状态和数据。包括所有寄存器、各总线、内存所有单元等。
- ◆能随时设置各寄存器和内存的数值，方便调试。

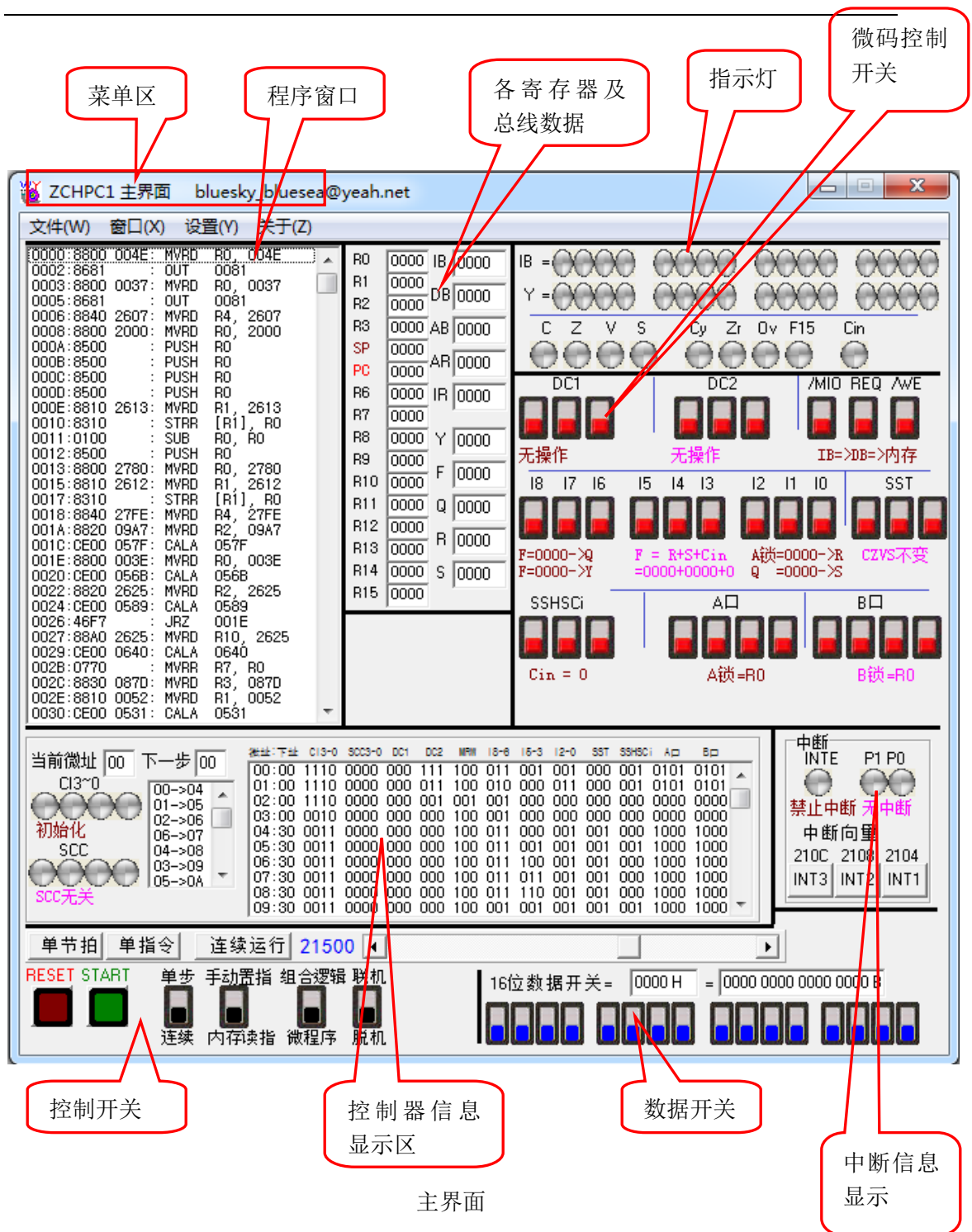
- ◆拥有 2 个终端，使编程更灵活。
- ◆能动画显示系统内部的数据流动，能直观的表现现象，便于理解和调试。
- ◆能编译汇编语言，并可自己设计新指令并编译。
- ◆能模拟 EEPROM 功能，能长期保存数据。并且便于查看和修改。
- ◆可以运行 BASIC 语言。

3、主界面各部分介绍

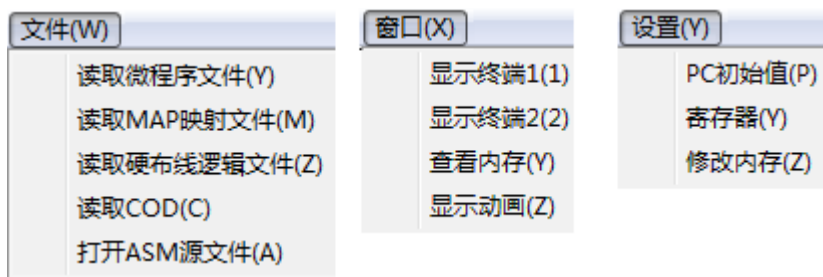
运行 ZCHPC1.exe 文件即可显示主界面。

主界面的信息非常多，不同的实验使用的区域也不同。有些实验要观察的信息可以通过二级窗口显示。

现将主界面功能分别介绍。



(1) 主界面 - 菜单区



(2) 主界面 - 程序窗口

0571:8281	:	IN	0081
0572:0800	:	SHR	R0
0573:45FD	:	JRNC	0571
0574:8700	:	POP	R0
0575:8680	:	OUT	0080
0576:8F00	:	RET	
0577:CE00	065D:	CALA	065D
0579:0680	:	OR	R8, R0
057A:CE00	06F5:	CALA	06F5
057C:8000	0335:	JMPA	0335
057E:8F00	:	RET	
057F:8500	:	PUSH	R0
0580:8102	:	LDRR	R0, [R2]
0581:0500	:	TEST	R0, R0
0582:4604	:	JRZ	0587
0583:CE00	056B:	CALA	056B
0585:0920	:	INC	R2
0586:41F9	:	JR	0580
0587:8700	:	POP	R0
0588:8F00	:	RET	

显示当前内存的所有程序，从左到右 3 列，分别是内存地址、机器码、汇编码，用冒号“:”间隔。蓝色条显示当前执行的语句。本例中，当前语句的地址是 0576H，指令的机器码是 8F00H，汇编程序的指令是 RET。

本窗口初始就有程序数据，是操作系统和 BASIC 语言解释器。如果读入机器码 cod 文件，会在本窗口相应地址显示。

(3) 主界面 - 各寄存器及总线数据

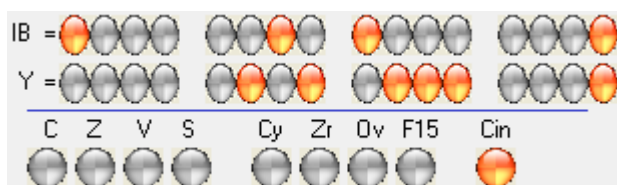
R0	7F80	IB	0000
R1	2612		
R2	09BD	DB	0B00
R3	0000	AB	0572
SP	27FA		
PC	0573	AR	0572
R6	0000	IR	0B00
R7	0000		
R8	0000	Y	FF01
R9	0000		
R10	0000	F	FF01
R11	0000	R	0000
R12	0000		
R13	0000	S	FF01
R14	0000	Q	0000
R15	0000		

本图中，左边是实验系统运算器 AM2901 中 16 个寄存器，其中 R4 和 R5 是专用寄存器，R4 用作 SP 堆栈指针，R5 用做 PC 程序计数器。

右边从上到下依次是：内部数据总线 IB 数值，外部数据总线 DB 数值，地址总线 AB 数值，地址寄存器 AR，指令寄存器 IR，ALU 的输出 Y、运算结果 F、ALU 数据来源 R 端和 S 端，移位寄存器 Q。

本窗口只用来显示数据，不能更改数据。有单独的修改界面修改寄存器数值，将在后面介绍。

(4) 主界面 - 指示灯



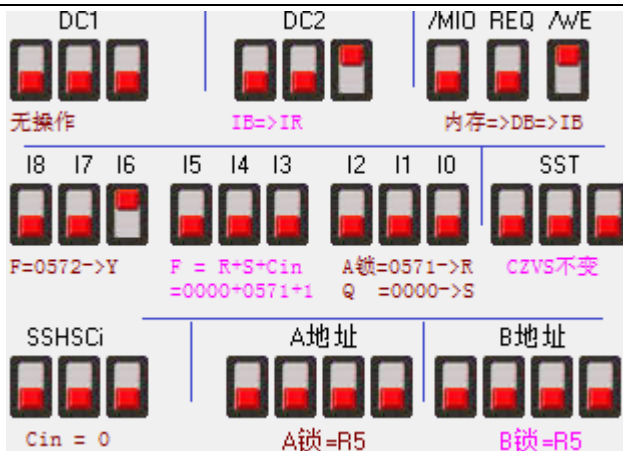
指示灯显示模拟软件的重要数据。

IB（内部数据总线）和 Y（ALU 输出）已经在“主界面 - 各寄存器及总线数据”用十六进制数值显示了，这里再用指示灯显示出二进制。IB 与 Y 是系统重要的两个数据，变化最频繁。

C、Z、V、S 是模拟软件的状态位寄存器值。C（进位）、Z（结果为 0）、V（溢出）和 S（符号位）。

Cy、Zr、Ov、F15 是 ALU 的输出标志位，Cin 是 ALU 的进位输入。

(5) 主界面 - 微码控制开关



在主界面的左下的“控制开关”部分，“联机/脱机”选择开关的“脱机”状态时，通过拨动此区域开关来设置系统微码控制信号。在“联机”状态时，开关不能拨动。

(6) 主界面 - 控制开关和数据开关



“单节拍”按钮：运行一个节拍，相当于发出一个单脉冲。

“单指令”按钮：运行一条指令。连续运行多个节拍，当前指令完成后自行暂停。

“连续运行”按钮：按下将连续运行，再次按下可暂停。右边控制条可调节“连续运行”的速度。

RESET 键：复位键。可停止系统运行，停止发脉冲，进行系统复位等设置。

START 键：系统运行，产生脉冲。根据“单步/连续”选择，可能是单脉冲，也可能是连续脉冲。

“单步/连续”开关：拨动开关置向上，为单步；置向下，为连续（置上为1，置下为0）。

“单步”时，每按一次“START”按键，产生一个单脉冲（相当于上面的“单节拍”）；“连续”时，在复位后，按一次“START”按键后，将产生连续脉冲（相当于上面的“连续运行”）。

“手动置指/内存读指”：确定指令寄存器 IR 的数据来源。拨动开关置向上，IR 来源为“数据开关”；置向下，IR 来源为内存。通常情况下的执行程序，应将其置为“内存读指”（置下）；只有在单独执行某一条指定的指令时，才置为“手动置指”（置上），此时可将右边 16 位数据开关设置为某条指令的机器码，该数值将会送到指令寄存器 IR。

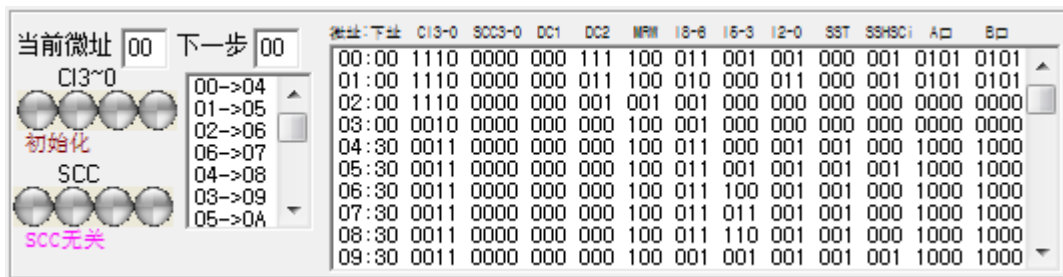
“组合逻辑/微程序”：控制器选择。开关置上，为组合逻辑控制器；置下，为微程序控制器。

“联机/脱机”：控制信号来源选择。拨动开关置向上，为联机，控制信号来自“组合逻辑”。

辑/微程序”；置向下，为脱机，控制信号来自几十个红色“微码控制开关”。执行程序时，应将其置为“联机”；做手动实验时，置为“脱机”。

16 位数据开关：数据开关。蓝色的，共 16 个。上方显示输入的数值，分别是十六进制和二进制显示。在实验中，无论“联机”还是“脱机”，本组开关都能拨动。

(7) 主界面 - 控制器信息显示区



控制器信息区

本界面是控制器信息显示区。在“联机”状态时，本区域显示各个控制信号是如何产生的：如果是“联机+微程序”状态，则显示微程序控制器当前状态；如果是“联机+组合逻辑”状态，则显示组合逻辑控制器当前状态。在“脱机”状态时，由于控制信号来自“微码控制开关”，本界面信息无效。本界面详细介绍参见相关章节。

(8) 主界面 - 中断信息显示



本界面三个指示灯显示当前中断状态，下面 3 个按键分别作为 3 级中断的中断源，按键上方数字表示按下时中断服务程序所在的内存地址。

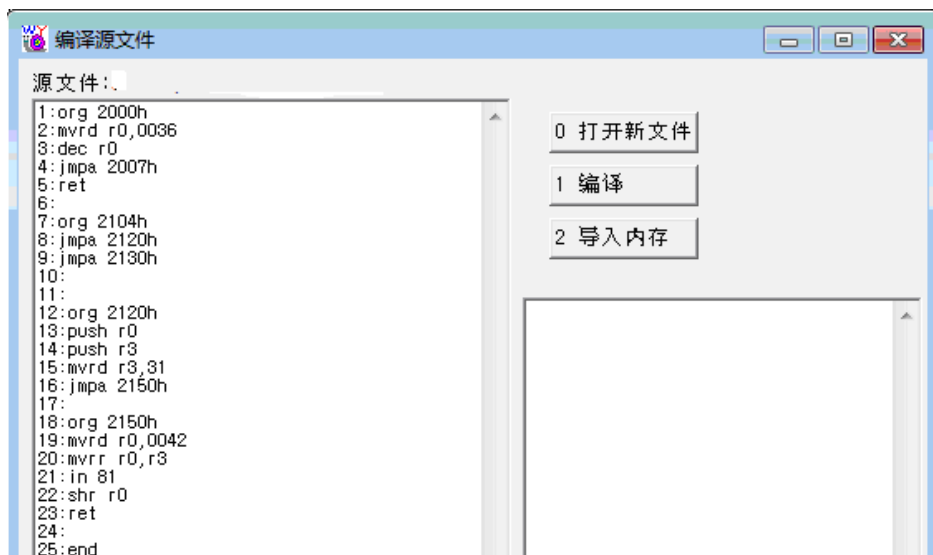
4、 菜单区的二级窗口简介

一、“文件”菜单

(1) 文件菜单中的“读取微程序文件”、“读取 MAP 映射文件”、“读取硬布线逻辑文件”等选项，在系统打开时已经自动执行过，如果没有修改相应的文件，不需要再次执行。

(2) 文件 -> 打开 ASM 源文件

通过菜单“文件”->“打开 ASM 源文件”即可打开。编译 ASM 汇编文件并导入内存。



编译文件界面

界面上有 3 个按钮，如果已经打开了源程序，则按“编译”后，就能生成 COD 文件，如果想继续导入到模拟软件的模拟内存，则按“导入内存”即可。

本软件只能编译 ASM 文件，无法编辑和修改。若想修改源程序请使用编辑软件，比如 Edit，记事本，FlexEdit 等软件。

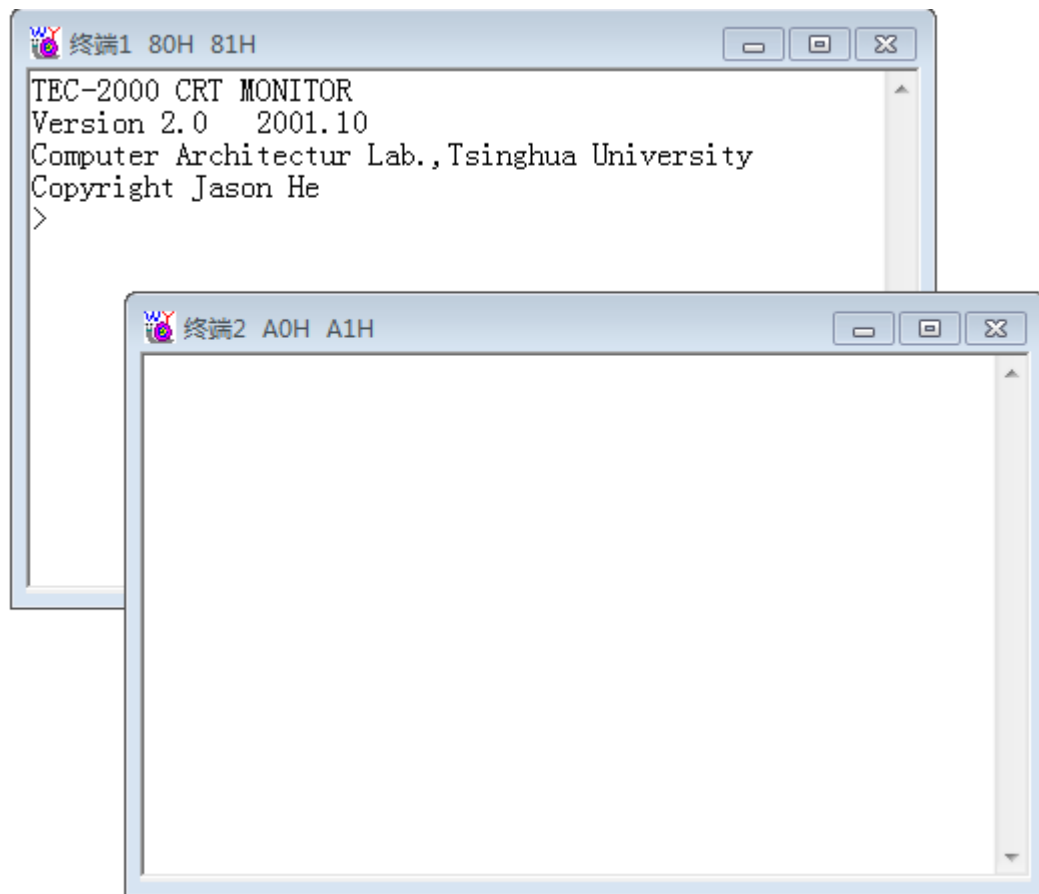
(3) 文件 -> 读取 COD

如果是只导入 COD 文件，则打开菜单“文件”---“读取 COD”即可。导入 COD 文件到模拟内存。

二、“窗口”菜单

(1) 窗口-> 显示终端 x (x 是 1 或者 2)

通过菜单“窗口”->“显示终端 x”即可打开终端窗口。



终端窗口

模拟软件有 2 个终端窗口，可用于信息输入和输出。软件启动会自行开启窗口“终端 1”。两个终端窗口对应的 I/O 地址在窗口标题栏有显示，分别是“终端 1 80H、81H”和“终端 2 A0H、A1H”。终端 I/O 地址不可更改，在编程时必须按照终端相应地址设置程序的输入输出地址。

(2) 窗口-> 查看内存

通过菜单“窗口”->“查看内存”即可打开。显示内存的窗口。



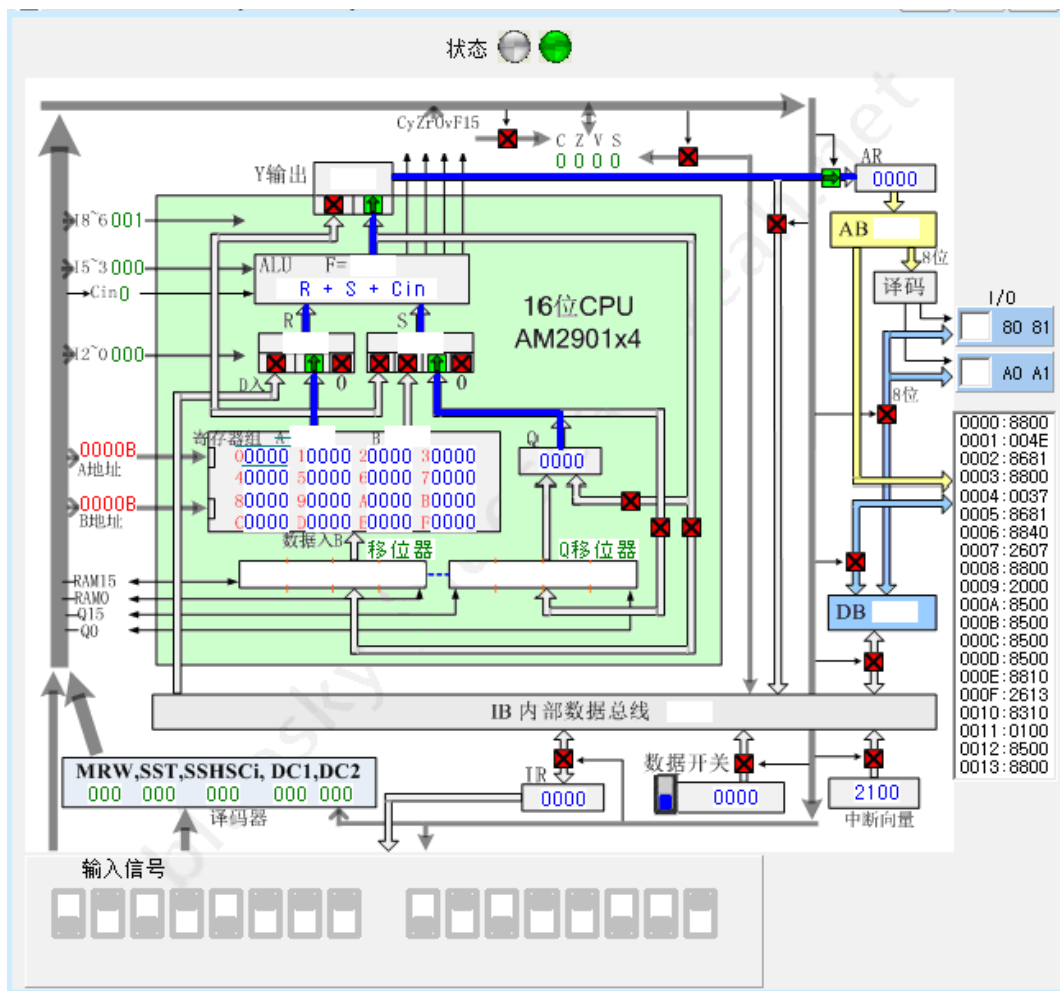
显示所有RAM内容，每行16 单元，每单元2字节。 查看内容前先点--> <input type="button" value="刷新"/>															
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000:	8800	004E	8681	8800	0037	8681	8840	2607	8800	2000	8500	8500	8500	8810	2613
0010:	8310	0100	8500	8800	2780	8810	2612	8310	8840	27FE	8820	09A7	CE00	057F	8800
0020:	CE00	0560	8820	2625	CE00	0589	46F7	88A0	2625	CE00	0640	0770	8830	087D	8810
0030:	CE00	0531	0301	4700	0820	4703	CE00	012D	41E5	CE00	06CA	4632	CE00	0186	41DF
0040:	8113	0511	462B	0301	47FA	8501	8810	0008	0013	8131	8710	CE00	0696	4603	CE00
0050:	471D	0820	8400	8500	8501	8502	8810	27FE	8820	8000	8312	0910	8313	8720	8710
0060:	8C00	8507	CE00	27FE	8770	8800	0047	0307	47B5	CE00	0563	CE00	0847	41B0	8820
0070:	CE00	057F	41AB	4705	8501	8810	2606	81F1	8710	07EF	88C0	000F	07FE	CE00	0664

本窗口能显示 RAM 所有数据，打开窗口后，必须按“刷新”才能显示。每行显示

16 个单元，每单元 4 位十六进制显示。地址空间是 0000H-27FFH。本窗口只能显示，不能更改内容。

(3) 窗口-> 显示动画

通过菜单“窗口”->“显示动画”即可打开动画显示窗口。



动画界面

本窗口通过动画展现数据在系统内部的流动过程。当动画正在运行时，上面的红灯

会闪烁，动画结束时绿灯会亮起。

动画窗口并不是打开就能显示动画，需要满足以下三条之一才能显示：

◆当“单步”时，只要打开动画窗口，就可以显示动画。

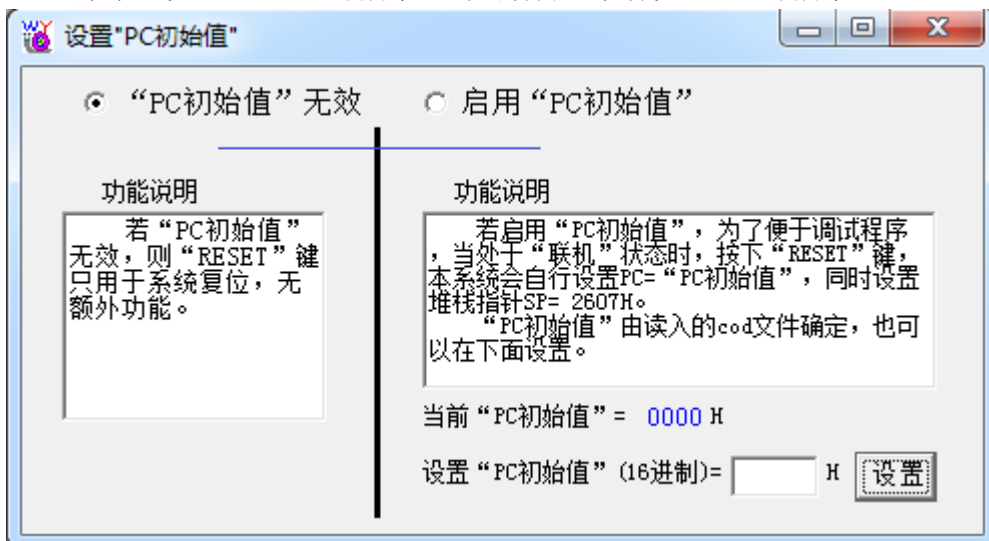
◆当“连续”时，速度设置成低于 20，如图 ，显示。

◆当“连续”时，无论速度设置如何，按“单节拍”也显示动画。

三、“设置”菜单

(1) 设置 -> PC 初始值

通过菜单“设置”->“PC 初始值”即可打开。手动设置“PC 初始值”。



设置 PC 初始值

本窗口的默认选择是“PC 初始值”无效。

如果为了调试程序方便，可以选择“启用 PC 初始值”，此时的 RESET 键增加了一些功能，即 PC 指针复位到需要调试程序的地址，便于反复执行自己编写的程序，观察现象，不受操作系统的约束。

(2) 设置 -> 寄存器

通过菜单“设置”->“寄存器”即可打开。手动设置寄存器数值。

设置寄存器界面

寄存器列表 (R0-R15):

- R0:
- R1:
- R2:
- R3:
- R4=SP:
- R5=PC:
- R6:
- R7:
- R8:
- R9:
- R10:
- R11:
- R12:
- R13:
- R14:
- R15:

状态位:

- C: ☐
- Z: ☐
- V: ☐
- S: ☐

中断向量初始:

AR:

IR:

Q:

全清空:

保存: 关闭:

设置寄存器界面

在相应框中直接填写所需十六进制数值并按“保存”，不需要修改的寄存器，保持空白。

为便于调试程序，在本界面也可以直接设置“中断”向量及状态位寄存器的数值。

(3) 设置 -> 修改内存

通过菜单“设置”->“修改内存”即可打开。

RAM内存修改

第一步：读取内存数值 第二步：表中填写新数值 第三步：写回内存

起始地址: 读取 写回

(16进制 0~3FFF)

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0000	8800	004E	8681	8800	0037	8681	8840	2607	8800	2000	8500	8500	8500	8500	8810	2613
0010	8310	0100	8500	8800	2780	8810	2612	8310	8840	27FE	8820	09A7	CE00	057F	8800	003E
0020	CE00	056B	8820	2625	CE00	0589	46F7	88A0	2625	CE00	0640	0770	8830	087D	8810	0052
0030	CE00	0531	0301	470B	0820	4703	CE00	012D	41E5	CE00	06CA	4632	CE00	0186	41DF	0930
0040	8113	0511	462B	0301	47FA	8501	8810	0008	0013	8131	8710	CE00	0696	4603	CE00	062B
0050	4710	0820	8400	8500	8501	8502	8810	27FE	8820	8000	8312	0910	8313	8720	8710	8700
0060	8C00	8507	CE00	27FE	8770	8800	0047	0307	47B5	CE00	0563	CE00	0847	41B0	8820	0A1D
0070	CE00	057F	41AB	4705	8501	8810	2606	81F1	8710	07EF	88C0	000F	07FE	CE00	0664	8800
0080	3A20	CE00	0565	81FE	CE00	0664	8800	0020	CE00	056B	810E	09E0	0700	CE00	0535	CE00
0090	07E7	470E	CE00	055C	8800	4457	CE00	0565	8800	2009	CE00	0565	07FD	CE00	0664	411B
00A0	8800	2009	CE00	0565	0522	4615	0700	CE00	0531	0820	4621	0820	4623	0820	4628	0820
00B0	4636	0820	4637	0820	463C	0820	4646	0820	4659	0820	416C	8500	CE00	0563	8700	CE00
00C0	075C	4604	08C0	4602	8000	007C	8501	8810	2606	831E	8710	8F00	0700	CE00	0786	41EB
00D0	0700	CE00	054F	0700	CE00	0786	41E4	0700	000E	CE00	0531	850E	8500	070E	CE00	0535
00E0	87E0	000E	87E0	07F0	CE00	0664	41D4	CE00	0662	41D1	0700	CE00	0531	07F0	CE00	0664
00F0	41CA	0700	CE00	0531	CE00	0656	CE00	07B6	CE00	0553	CE00	0662	41BE	CE00	0556	8500

信息: 2000H~27FFH, 2800H~2FFFFH, 3000H~37FFFH, 3800H~3FFFFH 的内存地址, 都映射到 2000H~27FFH

分 3 步操作，“读取”→“修改”→“写回”。

这里修改的是 RAM 内容，并且实际地址只有 0000~27FFH。

如果需要直接修改 EEPROM 内容，先编辑修改文件 EEPROM.txt 中相应单元内容，再运行模拟软件。

5、 监控使用简介

系统内存 0~2000H 单元已经存储了监控程序和 BASIC 语言解释程序。其中的监控相当于一个小操作系统，包括 A、U、G、T、R、D、E 共 7 条命令，其格式为“命令+空格+参数”。

A 单条汇编命令

格式：A adr。这里的参数“adr”是十六进制数字（不要带 H），表示要写入机器码的地址。不带参数 adr 则从系统默认地址开始。功能：完成单条指令的汇编操作，把产生出来的执行代码放入对应的内存单元中。输入完一句后，按回车，可继续输入下一句汇编程序。

U 反汇编命令

格式：U adr。功能：每次从指定的（或默认的）地址反汇编 15 条指令，并将结果显示在终端屏幕上。执行完后，接下来键入不带参数的 U 命令时，将从上一次反汇编的最后一条语句之后接着继续反汇编。

G 执行程序命令

格式：G adr。功能：从指定的（或默认的）地址连续运行一个用户程序。为了使程序执行后能返回监控，要求每个用户程序的最后一条指令一定为 RET 指令。

T 单指令执行程序命令

格式：T adr。功能：从指定的（或 PC 中的当前地址）开始单条执行程序指令。每运行一次 T 将执行一条指令。每次执行后均显示所有通用寄存器及状态寄存器的内容，并反汇编出下一条将要执行的指令。

R 显示/修改寄存器内容命令

格式：R 寄存器名。功能：不带参数时，是显示全部寄存器及状态寄存器的值。带参数时，是修改相应寄存器的值，如 R R0✓，再输入 36，表示把 16 进制的值 36 赋给 R0。

D 显示存储器内容命令

格式：D adr。功能：从指定（或默认的）地址开始显示内存 128 个存储字的内容。再次单独输入 D 命令可继续显示。

E 修改存储器内容命令

格式：E adr。功能：从指定（或默认的）地址逐字节修改每个内存单元的内容，要求用户输入一个新的值。如果要连续修改下面的单元的内容，则应在输入新值后按空格键。若用户敲了回车键，则会结束 E 命令的执行过程。

第三章 基础汇编语言程序设计

3.1 指令系统综述

3.1.1 指令分类

本系统的指令按不同的分类标准有多种划分方式：

- 1、从指令长度区分，有单字指令和双字指令。
- 2、从操作数的个数区分，有三操作数指令、双操作数指令、单操作数指令和无操作数指令。
- 3、从使用的寻址方式区分，有寄存器寻址、寄存器间址、立即数寻址、直接地址、相对寻址等多种基本寻址方式。
- 4、从指令功能区分，给出了算术和逻辑运算类指令、读写内存类指令、输入/输出类指令、转移指令、子程序调用和返回类指令，还有传送、移位、置进位标志和清进位标志等指令。
- 5、按照指令的功能和它们的执行步骤，可以把系统的指令划分为如下 4 组。

A 组： 基本指令 ADD、SUB、AND、OR、XOR、CMP、TEST、MVRR、DEC、INC、SHL、SHR、JR、JRC、JRNC、JRZ、JRNZ

扩展指令 ADC、SBB、RCL、RCR、ASR、NOT、CLC、STC、EI、DI、JRS、JRNS、JMPR

B 组： 基本指令 JMPA、LDRR、STRR、PUSH、POP、PUSHF、POPF、MVRD、IN、OUT、RET

C 组： 扩展指令 CALR、LDRA、STRA、LDRX、STRX

D 组： 基本指令 CALA 扩展指令 IRET

- 1) A 组指令完成的是通用寄存器之间的数据运算或传送，在取指之后可一步完成。
- 2) B 组指令完成的是一次内存或 I/O 读、写操作，在取指之后可两步完成，第一步把要使用的地址传送到地址寄存器 ARH、ARL 中，第二步执行内存或 I/O 读、写操作。
- 3) C 组指令在取指之后可三步完成，其中 CALR 指令在用两步读写内存之后，第三步执行寄存器之间的数据传送；而其它指令在第一步置地址寄存器 ARH、ARL，第二步读内存（即取地址操作数）、计算内存地址、置地址寄存器 ARH、ARL，第三步读、写内存。
- 4) D 组指令完成的是两次读、写内存操作，在取指之后可四步完成。

3.1.2 指令格式

TEC-ZCHPC1 是 16 位机，实现 29 条基本指令，用于编写监控程序和支持简单的汇编语言程序设计。同时保留了 19 条扩展指令，供学生在实验中完成对这些指令的设计与调试。

指令格式，支持单字和双字指令，第一个指令字的高 8 位是指令操作码字段，低 8 位和双字指令的第二个指令字是操作数、地址字段，分别有 3 种用法，如下图所示。

操作码	DR	SR
	IO 端口地址/相对偏移量	

立即数/直接内存地址/变址偏移量

这 8 位指令操作码（记作“IR15~IR8”），含义如下：

- 1) IR15、IR14 用于区分指令组：0X 表示 A 组，10 表示 B 组，11 表示 C、D 组；C、D 组的区分还要用 IR11，IR11=0 为 C 组，IR11=1 为 D 组。
- 2) IR13 用于区分基本指令和扩展指令：基本指令该位为 0，扩展指令该位为 1；
- 3) IR12 用于简化控制器实现，其值恒为 0；
- 4) IR11~IR8 用于区分同一指令组中的不同指令。

16 位机根据指令字长、操作数不同可划分为如下 5 种指令格式：

- 1) 单字、无操作数指令：

格式：

操作码	0000	0000
-----	------	------

基本指令：

PSHF; 状态标志（C、Z、V、S、P1、P0）入栈
POPF; 弹出栈顶数据送状态标志寄存器
RET 子程序返回

扩展指令：

CLC; 清进位标志位 C=0
STC; 置进位标志位 C=1
EI; 开中断，置中断允许位 INTE=1
DI; 关中断，置中断允许位 INTE=0
IRET; 中断返回

- 2) 单字、单操作数指令：

格式：

操作码	DR 0000
	0000 SR
	OFFSET
	I/O PORT

基本指令：

DEC DR; $DR \leftarrow DR - 1$
INC DR; $DR \leftarrow DR + 1$

SHL DR; DR 逻辑左移, 最低位补 0, 最高位移入 C
 SHR DR; DR 逻辑右移, 最高位补 0, 最低位移入 C
 JR OFFSET; 无条件跳转到 ADR, ADR=原 PC 值+OFFSET
 JRC OFFSET; 当 C=1 时, 跳转到 ADR, ADR=原 PC 值+OFFSET
 JRNC OFFSET; 当 C=0 时, 跳转到 ADR, ADR=原 PC 值+OFFSET
 JRZ OFFSET; 当 Z=1 时, 跳转到 ADR, ADR=原 PC 值+OFFSET
 JRNZ OFFSET; 当 Z=0 时, 跳转到 ADR, ADR=原 PC 值+OFFSET
 IN I/O PORT; $R0 \leftarrow [I/O\ PORT]$, 从外设 I/O PORT 端口读入数据到 R0
 OUT I/O PORT; $[I/O\ PORT] \leftarrow R0$, 将 R0 中的数据写入外设 I/O PORT 端口
 PUSH SR; SR 入栈
 POP DR; 弹出栈顶数据送 DR

扩展指令:

RCL DR; DR 与 C 循环左移, C 移入最低位, 最高位移入 C
 RCR DR; DR 与 C 循环右移, C 移入最高位, 最低位移入 C
 ASR DR; DR 算术右移, 最高位保持不变, 最低位移入 C
 NOT DR; DR 求反, 即 $DR \leftarrow \neg DR$
 JMPR SR; 无条件跳转到 SR 指向的地址
 CALR SR; 调用 SR 指向的子程序
 JRS OFFSET; 当 S=1 时, 跳转到 ADR, ADR=原 PC 值+OFFSET
 JRNS OFFSET; 当 S=0 时, 跳转到 ADR, ADR=原 PC 值+OFFSET

3) 单字、双操作数指令:

格式:

操作码	DR	SR
-----	----	----

基本指令:

ADD DR, SR; $DR \leftarrow DR + SR$
 SUB DR, SR; $DR \leftarrow DR - SR$
 AND DR, SR; $DR \leftarrow DR \text{ and } SR$
 CMP DR, SR; $DR - SR$
 XOR DR, SR; $DR \leftarrow DR \text{ xor } SR$
 TEST DR, SR; DR and SR
 OR DR, SR; $DR \leftarrow DR \text{ or } SR$
 MVRR DR, SR; $DR \leftarrow SR$
 LDRR DR, [SR]; $DR \leftarrow [SR]$
 STRR [DR], SR; $[DR] \leftarrow SR$

扩展指令:

ADC DR, SR; $DR \leftarrow DR + SR + C$

SBB DR, SR; $DR \leftarrow DR - SR - C$

4) 双字、单操作数指令:

格式:

操作码	0000 0000
ADR	

基本指令:

JMPA ADR; 无条件跳转到地址 ADR

CALA ADR; 调用首地址在 ADR 的子程序

5) 双字、双操作数指令:

格式 1:

操作码	DR 0000
	0000 SR
DATA	

基本指令:

MVRD DR, DATA; $DR \leftarrow DATA$

扩展指令:

LDRA DR, [ADR]; $DR \leftarrow [ADR]$

STRA [ADR], SR; $[ADR] \leftarrow SR$

格式 2:

操作码	DR	SR
ADR		

扩展指令:

LDRX DR, OFFSET[SR]; $DR \leftarrow [OFFSET + SR]$

STRX DR, OFFSET[SR]; $[OFFSET + SR] \leftarrow DR$

3.1.3 指令汇总表

基本指令

基本指令表

指令格式	汇编语句	操作 数个 数	CZVS	指令 类型	功能说明
00000000 DRSR	ADD DR, SR	2	****	A 组 指 令	$DR \leftarrow DR + SR$
00000001 DRSR	SUB DR, SR	2	****		$DR \leftarrow DR - SR$
00000010 DRSR	AND DR, SR	2	****		$DR \leftarrow DR \text{ and } SR$
00000011 DRSR	CMP DR, SR	2	****		$DR - SR$
00000100 DRSR	XOR DR, SR	2	****		$DR \leftarrow DR \text{ xor } SR$
00000101 DRSR	TEST DR, SR	2	****		$DR \text{ and } SR$
00000110 DRSR	OR DR, SR	2	****		$DR \leftarrow DR \text{ or } SR$
00000111 DRSR	MVRR DR, SR	2		$DR \leftarrow SR$
00001000 DR0000	DEC DR	1	****		$DR \leftarrow DR - 1$
00001001 DR0000	INC DR	1	****		$DR \leftarrow DR + 1$
00001010 DR0000	SHL DR	1	* ...		$DR, C \leftarrow DR * 2$
00001011 DR0000	SHR DR	1	* ...		$DR, C \leftarrow DR / 2$
01000001 OFFSET	JR OFFSET	1		无条件跳转
01000100 OFFSET	JRC OFFSET	1		$C=1$ 时跳转
01000101 OFFSET	JRNC OFFSET	1		$C=0$ 时跳转
01000110 OFFSET	JRZ OFFSET	1		$Z=1$ 时跳转
01000111 OFFSET	JRNZ OFFSET	1		$Z=0$ 时跳转
10000000 00000000 ADR (16 位)	JMPA ADR	1	B 组 指 令	无条件跳到 ADR
10000001 DRSR	LD RR DR, [SR]	2		$DR \leftarrow [SR]$
10000010 I/O PORT	IN I/O PORT	1		$R0 \leftarrow [I/O \text{ PORT}]$
10000011 DRSR	STRR [DR], SR	2		$[DR] \leftarrow SR$
10000100 00000000	PSHF	0		FLAG 入栈
10000101 0000SR	PUSH SR	1		SR 入栈
10000110 I/O PORT	OUT I/O PORT	1		$[I/O \text{ PORT}] \leftarrow R0$
10000111 DR0000	POP DR	1		$DR \leftarrow$ 出栈
10001000 DR0000	MVRD DR, DATA	2		$DR \leftarrow DATA$
10001100 00000000	POPF	0	D 组	FLAG \leftarrow 出栈
10001111 00000000	RET	0		子程序返回
11001110 00000000 ADR (16 位)	CALA ADR	1		调用首地址在 ADR 的子程序

注：1、表中 CZVS 一列，“*”表示对应状态位在该指令执行后会被重置；“.”表示对应状态位在该指令执行后不会被修改。

2、运算器芯片中有 16 个通用寄存器（累加器）R0~R15，其中：

R4 用作 16 位的堆栈指针 SP； R5 用作 16 位的程序计数器 PC；

其余寄存器用作通用寄存器，即多数双操作数指令和单操作数指令中的 DR、SR；

扩展指令

扩展指令表

指令格式	汇编语句	操作数个数	CZVS	指令类型	功能说明
00100000 DRSR	ADC DR, SR	2	****	A 组指令	$DR \leftarrow DR + SR + C$
00100001 DRSR	SBB DR, SR	2	****		$DR \leftarrow DR - SR - C$
00101010 DR0000	RCL DR	1	* ...		$DR \leftarrow DR$ 带进位 C 循环左移
00101011 DR0000	RCR DR	1	* ...		$DR \leftarrow DR$ 带进位 C 循环右移
00101100 DR0000	ASR DR	1	* ...		$DR \leftarrow DR$ 算术右移
00101101 DR0000	NOT DR	1	****		$DR \leftarrow \neg DR$
01100000 0000SR	JMPR SR	1		跳转到 SR 指明的地址
01100100 OFFSET	JRS OFFSET	1		S=1 时跳转
01100101 OFFSET	JRNS OFFSET	1		S=0 时跳转
01101100 00000000	CLC	0	0 ...		C=0
01101101 00000000	STC	0	1 ...		C=1
01101110 00000000	EI	0		开中断
01101111 00000000	DI	0		关中断
11100000 0000SR	CALR SR	1	C 组指令	调用 SR 指明的子程序
11100100 DR0000	LDRA DR, [ADR]	2		$DR \leftarrow [ADR]$
11100101 DRSR ADR (16 位)	LDRX DR, OFFSET[SR]	2		$DR \leftarrow [OFFSET + SR]$
11100110 DRSR ADR (16 位)	STRX DR, OFFSET[SR]	2		$[OFFSET + SR] \leftarrow DR$
11100111 0000SR	STRA [ADR], SR	1		$[ADR] \leftarrow SR$
11101111 00000000	IRET	0	D 组	中断返回

3.1.4 伪指令及其它

编译汇编程序 asm 文件时，需要伪指令辅助才能编译成功。用法如下：

ORG 定义程序首地址。如 “org 2000h” 表示本段程序将来调入地址为 2000h 的内存处。

一段程序中可以有多个 org，用于指定各个程序段的起始地址。

END 标志程序结束。放置在程序末尾。

EQU 定义变量。如 “CR EQU 0DH”，编译时 CR 代表数值 0DH。

DW 定义多个变量，如 “lab DW 3,5,'string','china' ”

字串：用于标志程序位置，便于从别处跳转到这里。如 “LOOP: DEC R2 ”

系统默认只能编译 29 条基本指令，所有的扩展指令，必须在文件 “InsExt.txt” 中有该指令的类型代码，否则无法编译。该文件中已经有 19 条扩展指令的代码，如果要编译更多的新指令，需要在文件中添加新指令类型代码。

3.2 模拟软件的基础汇编语言程序设计实验

3.2.1 实验目的

- 1、使用监控命令方法、单指令方式、动画显示方式运行编写的程序；
- 2、学习和掌握模拟软件监控命令的用法；
- 3、学习和掌握模拟软件的指令系统；
- 4、学习掌握简单汇编程序的设计；

3.2.2 实验要求

- 1、在使用该模拟软件之前，应先熟悉软件的各个组成部分及其使用方法。参见 “软件使用说明”
- 2、掌握模拟软件的指令系统，并且能够编写简单的汇编程序并调试成功。

3.2.3 实验内容

- 1、使用监控程序的 R 命令显示/修改寄存器内容、D 命令显示存储器内容、E 命令修改存储器内容。用 A 命令写一小段汇编程序，U 命令反汇编刚输入的程序，用 G 命令连续运行该程序，用 T、P 命令单步运行并观察程序单步执行情况。
- 2、文本编辑环境编写汇编程序文件，使用监控命令运行。
- 3、在模拟软件中，不通过监控命令直接运行程序。
- 4、以动画方式显示程序运行过程。

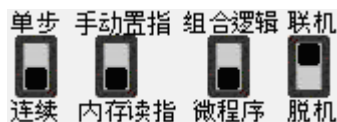
3.2.4 实验步骤

3.2.4.1 监控命令的使用

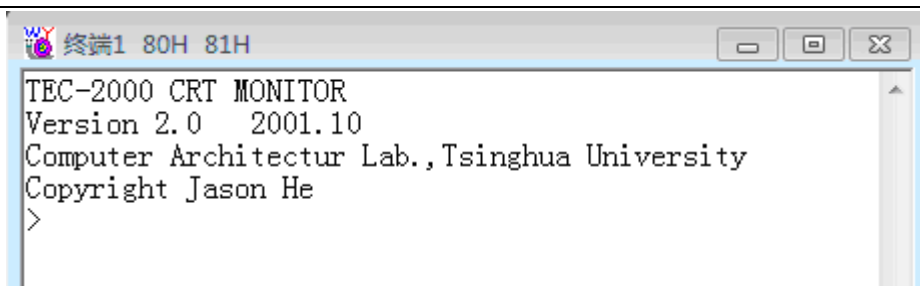
- 1、练习监控命令 R、D、E 等：

运行程序 “ZCHPC1.exe”

置控制开关为 **0001**（连续、内存读指、微程序、联机）。



按一下 **【RESET】** 按键，再按一下 **【START】** 按键，终端 1 上显示：



监控运行

使用监控命令 R, D, E 命令练习:

```
>R
R0=0000 R1=0000 R2=0000 R3=0000 SP=2780 PC=2000 R6=0000 R7=0000 R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=0000 R15=0000 F=00000000
2000: 0000      ADD    R0,    R0
>D
2000  0000 0000 0000 0000 0000 0000 0000 0000 .....
2008  0000 0000 0000 0000 0000 0000 0000 0000 .....
2010  0000 0000 0000 0000 0000 0000 0000 0000 .....
2018  0000 0000 0000 0000 0000 0000 0000 0000 .....
2020  0000 0000 0000 0000 0000 0000 0000 0000 .....
2028  0000 0000 0000 0000 0000 0000 0000 0000 .....
2030  0000 0000 0000 0000 0000 0000 0000 0000 .....
2038  0000 0000 0000 0000 0000 0000 0000 0000 .....
2040  0000 0000 0000 0000 0000 0000 0000 0000 .....
2048  0000 0000 0000 0000 0000 0000 0000 0000 .....
2050  0000 0000 0000 0000 0000 0000 0000 0000 .....
2058  0000 0000 0000 0000 0000 0000 0000 0000 .....
2060  0000 0000 0000 0000 0000 0000 0000 0000 .....
2068  0000 0000 0000 0000 0000 0000 0000 0000 .....
2070  0000 0000 0000 0000 0000 0000 0000 0000 .....
>E2000
2000  0000:
```

监控的 R、D、E 命令

2、监控命令 A 输入一段程序，反汇编，调试运行。

```

终端1 80H 81H
Version 2.0 2001.10
Computer Architectur Lab., Tsinghua University
Copyright Jason He
>a
2000: MVRD R0,AAAA
2002: MVRD R1,5555
2004: ADD R0,R1
2005: AND R0,R1
2006: RET
2007:
>U2000
2000: 8800 AAAA MVRD R0, AAAA
2002: 8810 5555 MVRD R1, 5555
2004: 0001 ADD R0, R1
2005: 0201 AND R0, R1
2006: 8F00 RET
2007: 0000 ADD R0, R0
2008: 0000 ADD R0, R0
2009: 0000 ADD R0, R0
200A: 0000 ADD R0, R0
200B: 0000 ADD R0, R0
200C: 0000 ADD R0, R0
200D: 0000 ADD R0, R0
200E: 0000 ADD R0, R0
200F: 0000 ADD R0, R0
2010: 0000 ADD R0, R0
>

```

A 命令输入，U 反汇编

```

2010: 0000 ADD R0, R0
>t2000

R0=AAAA R1=0000 R2=0000 R3=0000 SP=2780 PC=2002 R6=0000 R7=0000 R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=260D R15=0000 F=00000000
2002: 8810 5555 MVRD R1, 5555
>t

R0=AAAA R1=5555 R2=0000 R3=0000 SP=2780 PC=2004 R6=0000 R7=0000 R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=260D R15=0000 F=00000000
2004: 0001 ADD R0, R1
>t

R0=FFFF R1=5555 R2=0000 R3=0000 SP=2780 PC=2005 R6=0000 R7=0000 R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=260D R15=0000 F=00010000
2005: 0201 AND R0, R1
>

```

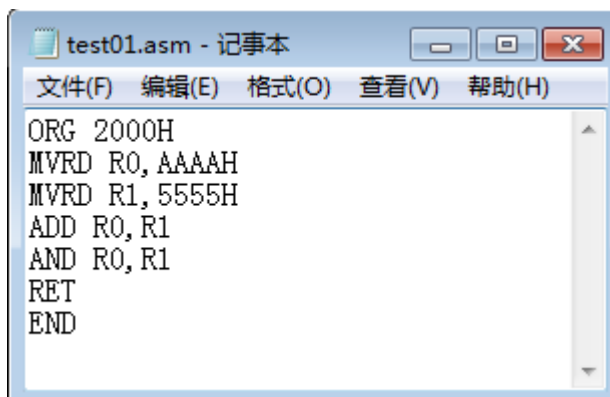
单步调试程序

单步调试，每执行一条指令，查看结果是否与预期值一致，如有错误再进行修改。

3.2.4.2 监控命令运行程序

文本编辑环境编写汇编程序文件，使用监控命令运行。

- 1、用记事本编写汇编程序“test01.asm”并保存（或者用其它软件编写），内容如下：

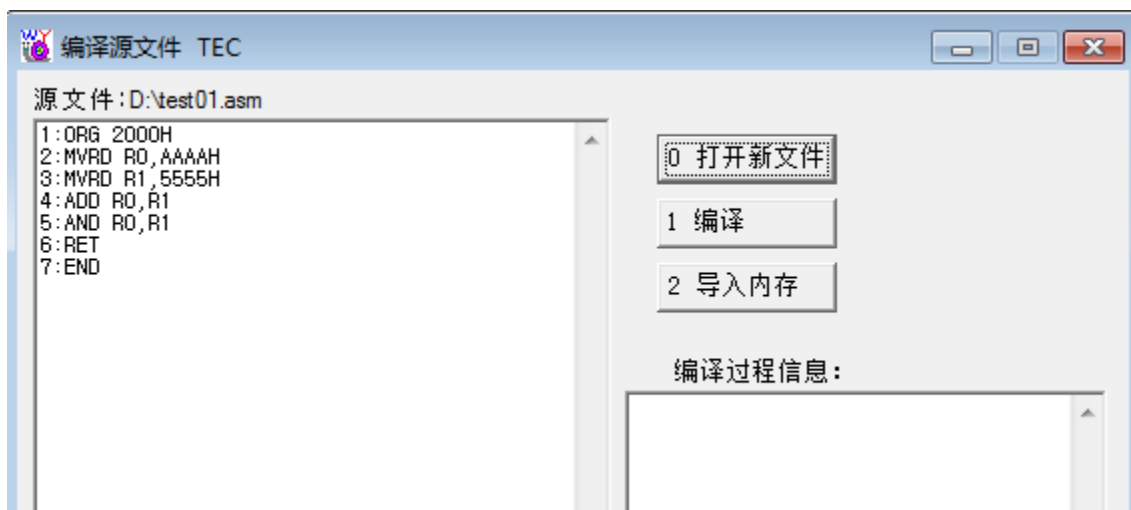


记事本输入源程序

- 2、运行程序 “ZCHPC1.exe”，显示主界面。
- 3、主界面菜单“设置”->“PC 初始值”，确保窗口内选择如下图：

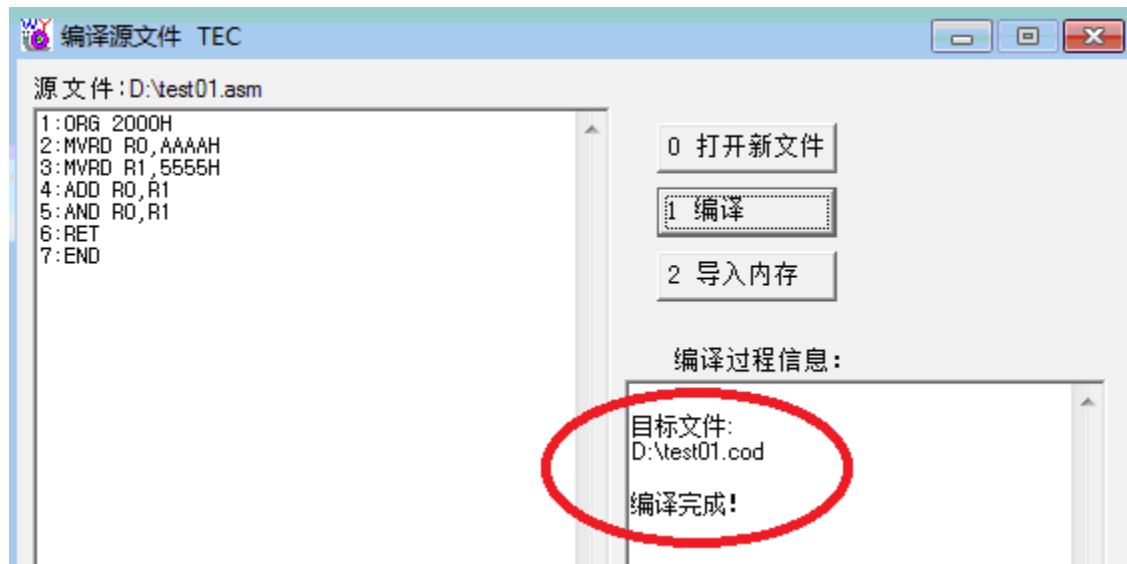


- 4、主界面菜单“文件”->“打开 ASM 源文件”，打开“test01.asm”，



编译界面

界面上有 3 个按钮，按【编译】后，生成.cod 文件。

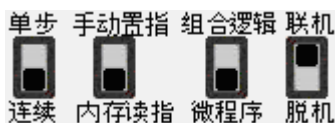


编译成功

编译如果无误，再按【导入内存】即可。如果程序导入内存完成，本界面自行消失。

- 5、置控制开关为 **0011**（连续、内存读指、组合逻辑、联机）

或者置开关为 **0001**（连续、内存读指、微程序、联机）。



- 6、设置速度条到最高

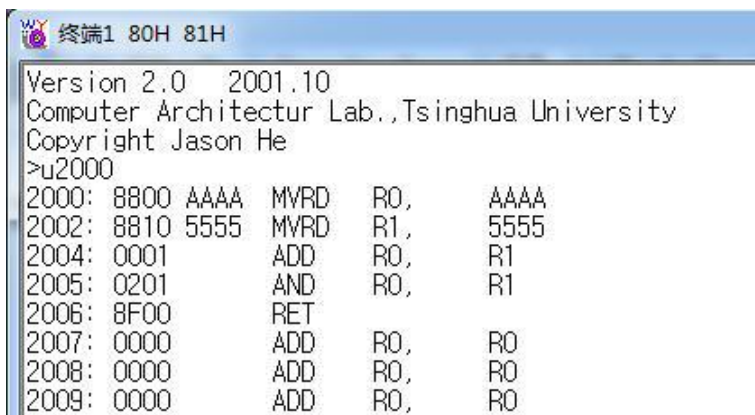


按一下【RESET】按键，再按一下【START】按键，终端 1 上显示大于号“>”。

此时在终端 1 窗口内即可进行监控命令操作。

若终端 1 窗口没有显示，点击“窗口”菜单 -> 显示终端 1，即可打开。

- 7、使用相关的监控命令调试程序。U 反汇编查看下是否是自己编写的程序，然后单步调试 T 或 P 查找错误进行修改，最后全速运行 G 查看结果。



```
终端1 80H 81H
>g2000
R0=5555 R1=5555 R2=0000 R3=0000 SP=2780 PC=2000 R6=0000 R7=0000 R8=0000
R9=0000 R10=0000 R11=0000 R12=0000 R13=0000 R14=2612 R15=0000 F=00000000
>
```

U 反汇编、G 运行

3.2.4.3 直接运行程序

在模拟软件中，不通过监控命令直接运行程序。

- 1、用记事本编写汇编程序“test01.asm”并保存，内容同上。
- 2、运行程序“ZCHPC1.exe”，显示主界面。
- 3、主界面菜单“设置”->“PC 初始值”，确保窗口内选择如下图



- 4、主界面菜单“文件”->“打开 ASM 源文件”，打开“test01.asm”。

按【编译】后，生成.cod 文件。

如果编译信息处显示有错误，请修改代码直至编译无误。

编译正确完成后，再按【导入内存】。会显示初始信息：



注：若有已经编译好的 cod 文件的话，直接导入 cod 文件即可。

首先还是前面的第 3 步，主界面菜单“设置”->“PC 初始值”，确保窗口内选择为“启用“PC 初始值””

然后主菜单“文件->读取 COD 文件”，选择文件，打开，导入内存。

1FFF:0000	:	ADD	R0, R0	R0	FFFF	IB	0201
2000:8800	AAAA:	MVRD	R0, AAAA	R1	5555	DB	0201
2002:8810	5555:	MVRD	R1, 5555	R2	0000	AB	2005
2004:0001	:	ADD	R0, R1	R3	0000	AR	2005
2005:0201	:	AND	R0, R1	SP	2607	IR	0201
2006:8F00	:	RET		PC	2006		
2007:0000	:	ADD	R0, R0	R6	0000		
2008:0000	:	ADD	R0, R0				
2009:0000	:	ADD	R0, R0				
200A:0000	:	ADD	R0, R0				
200B:0000	:	ADD	R0, R0				

执行完第3条指令

再按一下“单指令”：此时第四条 AND R0,R1 执行完成，可以看到R0=5555H

1FFF:0000	:	ADD	R0, R0	R0	5555	IB	8F00
2000:8800	AAAA:	MVRD	R0, AAAA	R1	5555	DB	8F00
2002:8810	5555:	MVRD	R1, 5555	R2	0000	AB	2006
2004:0001	:	ADD	R0, R1	R3	0000	AR	2006
2005:0201	:	AND	R0, R1	SP	2607	IR	8F00
2006:8F00	:	RET		PC	2007		
2007:0000	:	ADD	R0, R0	R6	0000		
2008:0000	:	ADD	R0, R0				
2009:0000	:	ADD	R0, R0				
200A:0000	:	ADD	R0, R0				
200B:0000	:	ADD	R0, R0				

执行完第4条指令

再按一下“单指令”，此时第五条 RET 执行，可以看到返回到 0000 地址。

0000:8800	004E:	MVRD	R0, 004E	R0	5555	IB	8800
0002:8681	:	OUT	0081	R1	5555	DB	8800
0003:8800	0037:	MVRD	R0, 0037	R2	0000	AB	0000
0005:8681	:	OUT	0081	R3	0000	AR	0000
0006:8840	2607:	MVRD	R4, 2607	SP	2608	IR	8800
0008:8800	2000:	MVRD	R0, 2000	PC	0001		
000A:8500	:	PUSH	R0	R6	0000		
000B:8500	:	PUSH	R0				
000C:8500	:	PUSH	R0				
000D:8500	:	PUSH	R0				
000E:8810	2613:	MVRD	R1, 2613				

执行完第5条指令

8、按【RESET】键，观察到 PC=2000H。重复按“单指令”，可再次执行本段程序。

这段代码的起始地址是“2000H”，在导入到内存时，2000H 即保存在“PC 初始值”处。在任何时刻，按下【RESET】键后，在进行系统初始化后，还将 PC（寄存器 R5）强置为“PC 初始值”，此时再按 单指令 按钮，就是从地址“2000H”开始执行。如此可以反复执行自己的程序，方便调试。

3.2.4.4 动画方式运行程序

以动画方式显示程序运行过程。动画界面以动画形式体现了实验系统内部的数据移动和变化情况，无论是在“脱机”还是“联机”状态，“单步”还是“连续”运行，只要条件设置合适，都能使动画启动。在这里我们以“联机”状态下为例讲解如何设置操作。

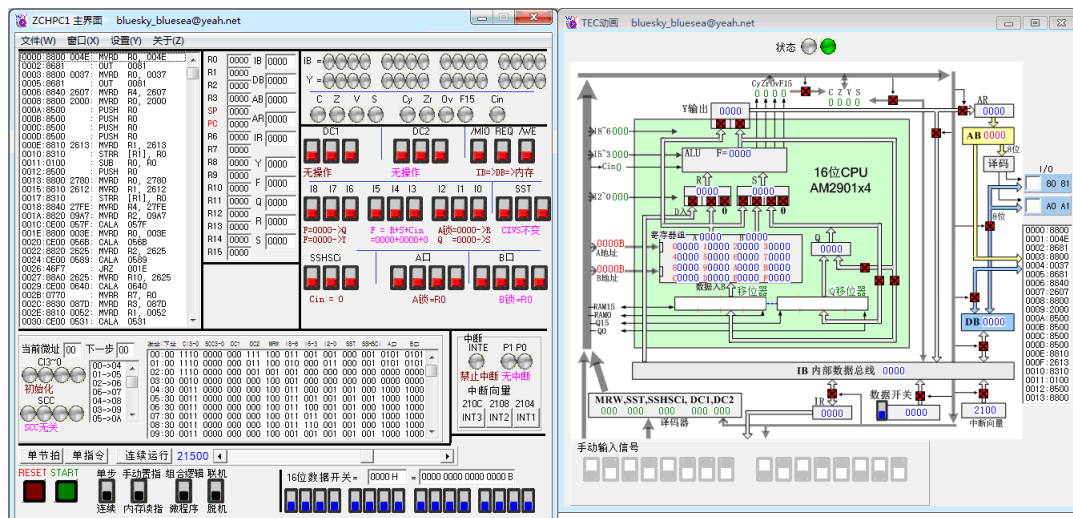
- 1、运行程序“ZCHPC1.exe”，显示主界面。
- 2、置控制开关为 0011（连续、内存读指、组合逻辑、联机）

或者置开关为 0001（连续、内存读指、微程序、联机）。

3、设置速度条为小于 20。

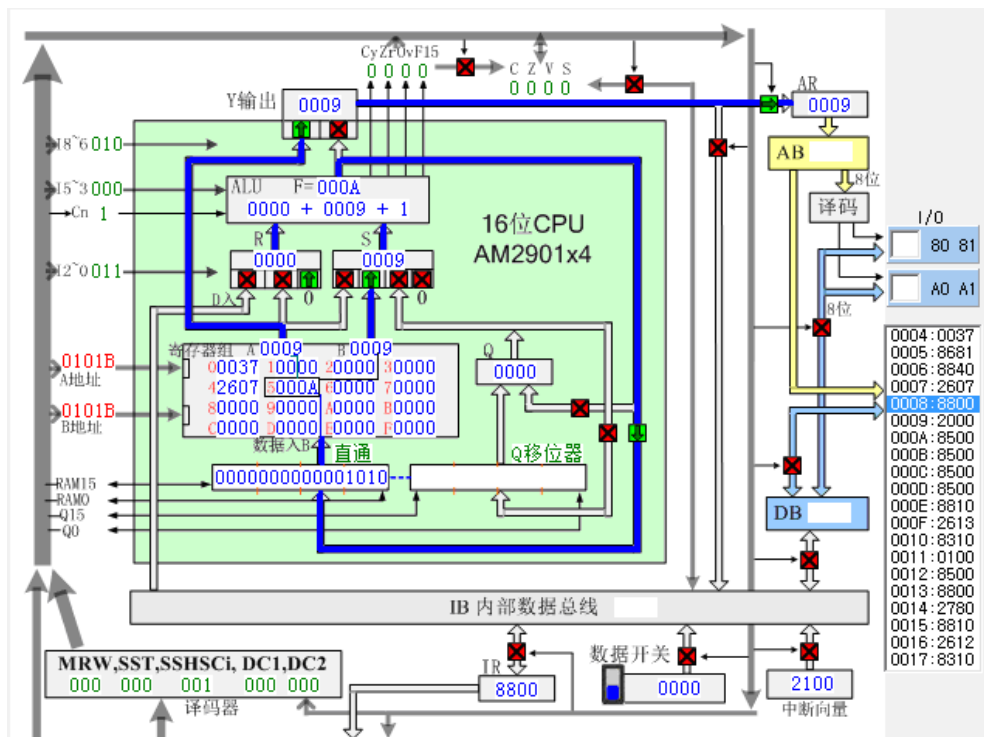
15

4、主界面菜单“窗口”->“显示动画”，打开动画窗口。





左边是主界面，右边是动画窗口。

5、按一下【RESET】按键，再按一下【START】按键，即可运行程序，同时启动动画。



动画界面

动画进行时上方的红灯不停的闪烁，动画结束后绿灯亮。本例子中是“连续”运行，所以动画会一直运行，不会自行停止。

动画窗口的淡绿色部分就是运算器部分，4片 AM2901 芯片。包含了寄存器、运算器和移位器等。右边是 I/O 和内存区域。正下方是控制信息显示区域，本图的“微程序信号”显示当前运行的微指令，右边的“注释”是该微指令的功能说明。

6、主界面的【RESET】键按下，动画和程序都会停止，再按【START】键可再运行。

如果运行时没有选择按【START】键，而是按 **单节拍**，则动画运行一个节拍的内容后停止，再次按 **单节拍** 会进行下一个节拍的动画，一步步运行。

动画运行速度可以调整，也可以暂停和单步。软件界面上有控制键，直接操作即可。

如果要以动画方式显示自己编写的程序，可按照先前的步骤，通过主界面菜单“文件”->“打开 ASM 源文件”编译和导入自己的程序，自行设置好 PC 后按【START】键或者 **单节拍**，即可在动画窗口观察程序的运行。也能通过按【RESET】键重置 PC，重新运行程序。

7、第 2 小步中的控制开关可以设置为 1001，此时按【START】键，即可启动动画，只运行一步的动画，运行结束动画会停止，等待再次按【START】键。功能与上步中的 **单节拍** 按键相同。

注：1、“连续”状态，运行速度设置的小于 20 时，按【START】键和“连续运行”键都可以启动动画，但“单指令”键不会有动画。

2、“连续”状态，无论速度设置如何，按“单节拍”键显示动画。

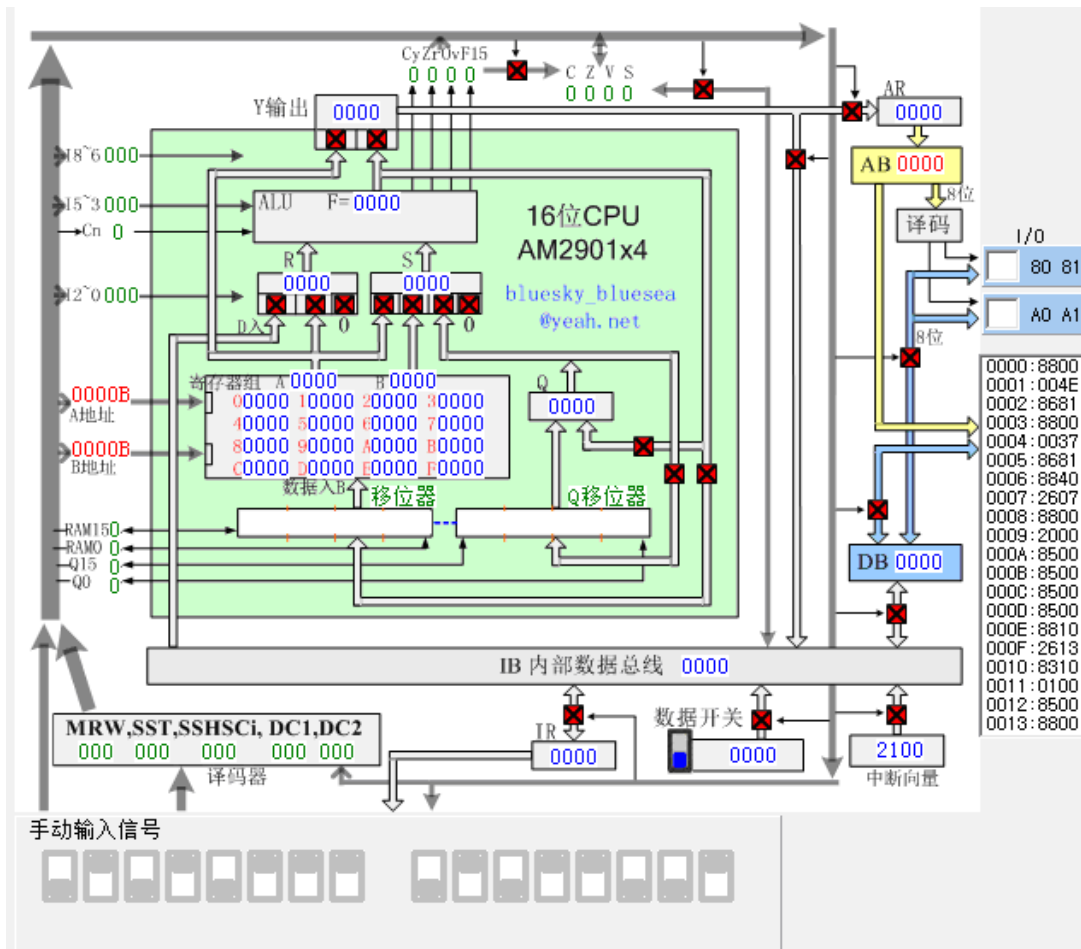
3、“单步”状态时，按【START】键即可启动动画。

第四章 运算器实验

4.1 模拟软件的运算器实验

4.1.1 实验系统模拟硬件基本组成

实验系统的模拟硬件包括运算器、控制器、主存储器、启停控制部件、串行 I/O 接口、中断线路、译码电路以及各类控制按键、开关及指示灯，其系统总体框图如下：



图中红色■受信号控制，确定本线路是断还是通。

图中的空心线→是 16 位的数据通道，方向箭头→和灰色箭头→表示控制信号通道。所有线路终端都有箭头，一端带箭头表示仅单向传输，两端都有箭头表示可双向传输。

图中数字有多种颜色，绿色的表示控制信号，蓝色表示数据，红色表示地址。

图的中心部位淡绿色方形大框内为 4 片 AM2901 芯片并联，左边多个控制信号输入，16 位输入数据 D 从下方进入（连在 IB 上），16 位输出数据向上送出（Y），4 个标志信号 Cy、Zr、Ov、F15 也向上送出。

图中最下方为控制信号产生处，在选择“脱机”时，信号来自主界面红色的微码控制开关；在“联机+微程序”方式时，控制信号来自微程序控制器；在“联机+组合逻辑”方式时，控制信号来自硬布线。

图左下方的 MRW, SST, SSHSCi, DC1, DC2 是译码器。

图下方的 IR 是指令寄存器；图右上方的 AR 是地址寄存器。

图中的“数据开关”对应主界面右下方的 16 位蓝色数据开关。

图中右下角的“中断向量”对应 3 个中断按钮的响应地址。

图中右边是 I/O 和内存。I/O 是 8 位的，标记“80 81”对应终端 1 窗口；标记“A0 A1”对应终端 2 窗口。内存栏两列分别是地址和内容，每个单元都是 16 位。

AM2901 中的寄存器 R5 是程序计数器 PC，R4 是堆栈指针 SP。

总线：使用两组总线，即地址总线、数据总线。

1) 地址总线 AB:

地址总线宽度为 16 位，地址总线 AB 的数值只来自地址寄存器 AR，而 AR 只接收运算器的结果 Y 输出。地址总线 AB 的输出送往存储器和 I/O 接口。由于系统只使用 8 位的 I/O 端口地址，因此在 I/O 读写指令中，只用到地址总线的低 8 位。

2) 数据总线 IB、DB:

数据总线分内部数据总线 IB（在 CPU 一方）与外部数据总线 DB（在主存与外设接口一方）。内、外部数据总线均为 16 位。

总线、中断、内存和 I/O 相关的控制信号如下：

DC1: 向 IB 总线写数据的控制信号。

DC1 编码	功能说明
000	无操作
001	ALU 的 Y 输出=>IB
010	IRL=>IB
011	C、Z、V、S、P1、P0=>IB
100	未使用，待扩展
101	中断向量=>IB
110	未使用，待扩展
111	16 位数据开关=>IB

DC2: 与寄存器 IR 和 AR 接收、中断有关的控制信号。

DC2 编码	功能说明
000	无操作
001	IB=>IR
010	未使用，待扩展

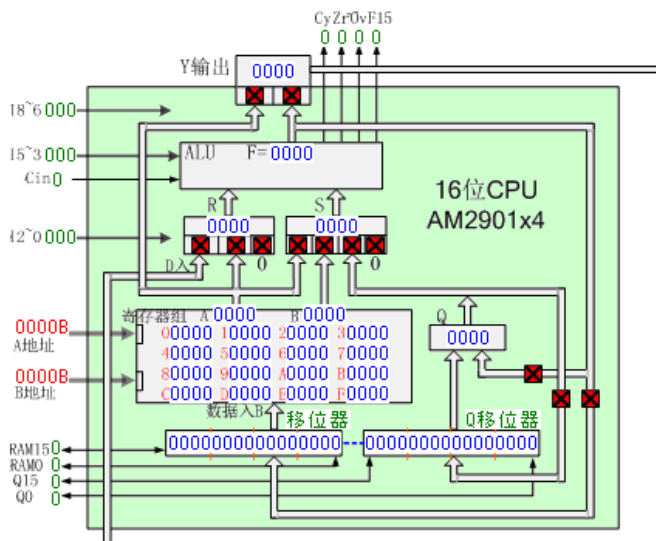
011	Y 输出=>AR
100	IB=>P1、P0
101	中断新等级=>P1、P0
110	允许中断
111	禁止中断

MRW:与内存、I/O 读写有关的控制信号：/MIO、REQ、/WE。（缩写为 MRW）

/MIO REQ /WE	功能说明
000	写内存。IB=>DB=>MEM
001	读内存。MEM=>DB=>IB
010	写 I/O。IBL=>DBL=>I/O
011	读 I/O。I/O=>DBL=>IBL
1XX	无上述读写操作

4.1.1.1 模拟硬件芯片的结构和功能

AM2901 芯片字长 4 位，模拟软件将 4 片并联成 16 位运算器。内部结构如下：



AM2901 由以下几部分组成：

(1) 算数逻辑运算部件 ALU，运算结果为 F。3 个运算输入是数据 R、S 和进位信号 Cin。它能够实现 $R+S$ 、 $S-R$ 、 $R-S$ 三种算术运算功能和 $R \vee S$ 、 $R \wedge S$ 、 $R \wedge \bar{S}$ 、 $R \oplus S$ 、 $\bar{(R \oplus S)}$ 五种逻辑运算功能。

(2) CyZrOvF15：产生运算结果 F 的同时，还送出 4 个标志位。分别是向高位的进位输出信号 Cy，运算结果为零的标志信号 Zr，溢出标志信号 Ov，最高位的状态信号 F15

(可用作符号位)。

(3) 16 个通用寄存器组成的寄存器组与 2 个锁存器。可以同时读双寄存器 (A 地址和 B 地址确定), 写单寄存器 (B 地址确定)。读寄存器时数据分别送到 A、B 锁存器 (图中寄存器组上方的 A、B)。写数据时由 B 地址确定写入哪个寄存器。

(4) Q 寄存器: 主要用于实现硬件的乘法、除法指令, 下方带移位器可左右移位。Q 能接收 Q 移位器的结果也能接收 ALU 的结果 F 值。Q 输出可以送到 ALU 的 S 输入端也可以送到下面的 Q 移位器。

(5) 移位器: 可以左右移动或者直通, 数据来自 ALU 的输出 F, 输出到寄存器组 (B 地址确定)。本移位器和 Q 移位器都是独立移位器, 但可通过外部电路连接后进行联合移位。

(6) 该芯片的其余部分是 5 组多路选通门, 通过它们, 实现芯片内上述 ALU、寄存器组、Q 寄存器三部分之间的联系, 实现该芯片和外界信息的输入与输出操作。

4.1.1.2 模拟硬件芯片控制信息

为了控制 Am2901 运算器按要求完成功能, 就必须向其提供相应的控制信号和数据。控制信号包括:

I2~I0: 确定送入 ALU 的两个操作数据 R 和 S 的组合关系 (实际来源)。R 从 D、A 和 0 中三选一, S 从 A、B、Q 和 0 中四选一, AM2901 芯片确定了 8 种组合。

I5~I3: 确定 ALU 的八种运算 (三种算术、五种逻辑运算) 功能中执行哪个。

I8~I6: ① 确定 Y 输出的来源 ② 寄存器组和 Q 寄存器是否接收数据 ③ 寄存器组和 Q 寄存器如何接收数据: 运算结果 F 是否送到寄存器组和 Q 寄存器, 是否中途进行移位运算; Q 移位器是否移位。

I2~0: 确定参加运算的数据 R、S 来源。表中 D 实际上来自内部数据总线 IB。

A、B 表示 A、B 锁存器, 来自“A 地址”和“B 地址”确定的寄存器值, Q 表示 Q 寄存器。

R、S 来源规则由 AM2901 芯片确定, 不能改变。

I2~0 编码	R、S 取值
000	R=A, S=Q
001	R=A, S=B
010	R=0, S=Q
011	R=0, S=B
100	R=0, S=A
101	R=D, S=A
110	R=D, S=Q
111	R=D, S=0

I5~3: 确定运算功能，一共 8 种运算，由 AM2901 芯片确定，不能改变。

表中 R、S 为 I2~0 确定的运算数；Cin 是最低位进位，由控制信号 SSHSCi 确定。

I5~3 编码	运算功能 $F = R ? S$
000	$F = R + S + Cin$
001	$F = S - R - Cin$
010	$F = R - S - Cin$
011	$F = R \vee S$
100	$F = R \wedge S$
101	$F = /R \wedge S$
110	$F = R \oplus S$
111	$F = /(R \oplus S)$

I8~6: 通用寄存器组和 Q 寄存器的结果选择以及运算结果 Y 输出的选择。AM2901 芯片固有规则，不能改变。A 表示 A 锁存器。B 表示“B 地址”确定的寄存器。

I8~6 编码	操作	说明
000	$F \Rightarrow Y, F \Rightarrow Q$	
001	$F \Rightarrow Y$	
010	$A \Rightarrow Y, F \Rightarrow B$	
011	$F \Rightarrow Y, F \Rightarrow B$	
100	$F \Rightarrow Y, F/2 \Rightarrow B, Q/2 \Rightarrow Q$	F、Q 分别右移
101	$F \Rightarrow Y, F/2 \Rightarrow B$	F 右移
110	$F \Rightarrow Y, 2F \Rightarrow B, 2Q \Rightarrow Q$	F、Q 分别左移
111	$F \Rightarrow Y, 2F \Rightarrow B$	F 左移

左移操作时 RAM15 与 Q15 分别为输出，RAM0 和 Q0 分别为输入；

右移操作时 RAM0 和 Q0 分别为输出，RAM15 和 Q15 分别为输入。

作为移位输入时，RAM15、RAM0、Q15 和 Q0 由 SSH 确定数值。

SSHSCi 功能 1: 确定最低位进位 Cin 的取值。16 位运算器的最低位进位输入，仅参与 3 种算数运算。

SSHSCi	Cin 取值	说明
000 或 1xx	0	1xx 时 SSHSCi 用于移位
001	1	
010	C	C 是状态位寄存器
011	Cin 保持	

SSHSCi 功能 2: 有左右移位运算时，确定各移位器的最高、最低位的初始值。

移位操作运算中 RAM15、RAM0、Q15 和 Q0 作为输入数的数值由 SSHSCi 确定。一般情况下 SSHSCi 要与 I8~6 配合使用。

左移运算 I8~6=110、111		
SSHSCi	RAM0	Q0
100 或 0xx	0	0
101	C	0
110	Q15	/F15
111	0	0

右移运算 I8~6=100、101		
SSHSCi	RAM15	Q15
100 或 0xx	0	0
101	C	0
110	Cy	RAM0
111	F15 \oplus Ov	RAM0

表中 Cy、F15、Ov 是 AM2901 的标志位输出，C 是状态位寄存器。

SST: 状态位寄存器 C、Z、V、S 数值确定。

SST	C	Z	V	S	说明
000	C	Z	V	S	C、Z、V、S 保持不变
001	Cy	Zr	Ov	F15	接收 ALU 的标志位输出的值
010	IB15	IB14	IB13	IB12	IB \Rightarrow C、Z、V、S
011	0	Z	V	S	C=0, Z、V、S 保持不变
100	1	Z	V	S	C=1, Z、V、S 保持不变
101	RAM0	Z	V	S	C=RAM0, Z、V、S 保持不变
110	RAM15	Z	V	S	C=RAM15, Z、V、S 保持不变
111	Q0	Z	V	S	C=Q0, Z、V、S 保持不变

A 口、A 地址、A 锁；B 口、B 地址、B 锁

AM2901 用来确定要读写寄存器的信号是“A 地址”、“B 地址”，即实际起作用的控制信号。

读寄存器时，A、B 地址分别确定 A 锁存器（A）、B 锁存器（B）的来源寄存器。写寄存器时，B 地址确定运算结果 F 存入哪个寄存器。

主界面微码控制开关和微程序中确定 A、B 地址的对应字段处叫“A 口”、“B 口”，可以手动操作和编写微程序。指令寄存器 IR 的低 8 位 IRL 也可以确定 A、B 地址。

A、B 口数值决定 A 地址、B 地址是来自“A 口”、“B 口”还是来自 IRL。规则如下：

当 A 口不是“1000”时，A 地址=A 口；当 A 口=1000 时，A 地址=IR[3]~[0]。

当 B 口不是“1000”时，B 地址=B 口；当 B 口=1000 时，B 地址=IR[7]~[4]。

举例：

A 口=0101。现象：读寄存器时，A 地址=A 口=0101，A 锁存器值来自寄存器 R5。

A 口=0101；B 口=1000，IR=“xxxx xxxx 0011 xxxx”。现象：读寄存器时，A 地址=A 口=0101，A 锁存器值=R5；B 地址=IR[7]~[4]=0011，B 锁存器值=R3。写寄存器时，运算结果 F 存入 R3。

A 口=1000，B 口=1000；IRL=“0111 0110”。现象：读寄存器时，B 地址=IR[7]~[4]=0111，

B锁=R7; A地址=IR[3]~[0]=0110, A锁=R6。写寄存器时, 运算结果F存入R7。

4.1.2 实验目的

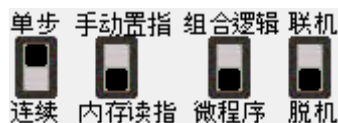
学习 AM2901 运算器的功能与具体用法, 掌握运算器部件的组成、控制与使用等知识。

4.1.3 实验内容

- 1、学习使用 AM2901 的控制信号 I8~6、I5~3、I2~0 的功能及其多种组合的应用。
- 2、使用 SSHSCi 控制 Cin 并与 I8...I0 组合使用。
- 3、操作状态寄存器 C, 参与移位运算。
- 4、使用 A、B 地址控制寄存器读写。

4.1.4 实验步骤

- 1、运行程序 “ZCHPC1.exe”。
- 2、将左下方的控制开关置为 1xx0 (单步、X、X、脱机)



- 3、主界面菜单 “窗口” -> “显示动画”, 打开动画窗口。
 - 4、按 **【RESET】** 键, 进行初始化。
- 按照以下 5、6、7.....顺序进行一系列的运算器实验。

注: 为方便操作, “单步” 状态时, 主界面右侧会显示另外一个 **【START】** 按键, 与左边的 START 功能相同。

5、验证 I2~0 所有组合的动能。

(1) 准备工作 1。设置数值, 寄存器 R0、R3、Q 置为不同的数值, 通过 I2~0 的不同编码确定参加运算的数据来源。这里置 R0=1, R3=2, Q=3。

(1.1) 设置 R0=1

微码控制开关为: “x” 与 “无关” 表示任意设置, 可以是 “0” 或 “1”。

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	1xx	011	000	111	00x	000	无关	0000

16 位数据开关为: 0000 0000 0000 0001

按 **【START】** 键, 观察动画窗口数据流动。

注: 为缩短实验操作时间, 可直接设置相关寄存器数值。方法是: 主界面菜单 “设置” -> “寄存器”, 在相应位置直接填入需要的数值保存即可, 注意是十六进制。

(1.2) 设置 R3=2

微码控制开关为:

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	1xx	011	000	111	00x	000	无关	0011

16 位数据开关为: 0000 0000 0000 0010

按【START】键, 观察动画窗口数据流动。

(1.3) 设置 Q=3

微码控制开关为:

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	1xx	000	000	111	00x	000	无关	无关

16 位数据开关为: 0000 0000 0000 0011

按【START】键, 观察动画窗口数据流动。

(2) 准备工作 2。设置外部数据 D=7。

为了验证开关 I2~0 的功能, 需要将其它功能开关设置为无效或辅助状态。

微码控制开关为:

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	011	1xx	001	000	000	000	000	0000	0011

16 位数据开关为: 0000 0000 0000 0111

(3) 经过 (1) (2) 步骤, R0、R3、Q 和 D 都有数值, 下面尝试 I2~0 的所有编码设置。

观察对象: 参与运算的 R、S 来源, 数据流向, AR 寄存器的结果。记录下结果并且分析。

(3.1) 置 I2~0=000, R=A, S=Q。按【START】键。

(3.2) 置 I2~0=001, R=A, S=B。按【START】键。

(3.3) 置 I2~0=010, R=0, S=Q。按【START】键。

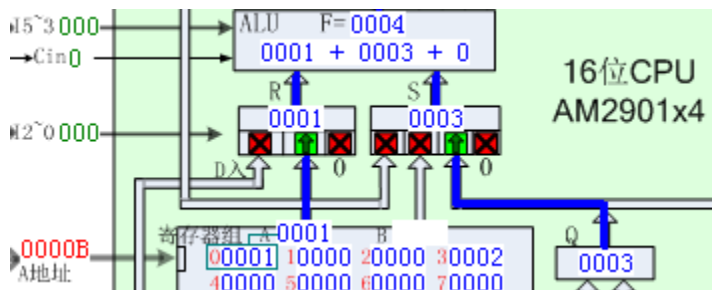
(3.4) 置 I2~0=011, R=0, S=B。按【START】键。

(3.5) 置 I2~0=100, R=0, S=A。按【START】键。

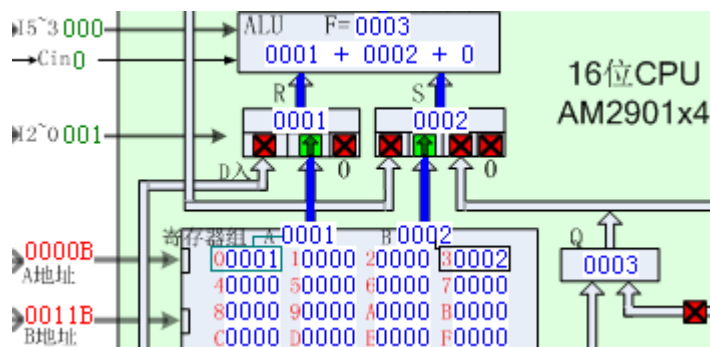
(3.6) 置 I2~0=101, R=D, S=A。按【START】键。

(3.7) 置 I2~0=110, R=D, S=Q。按【START】键。

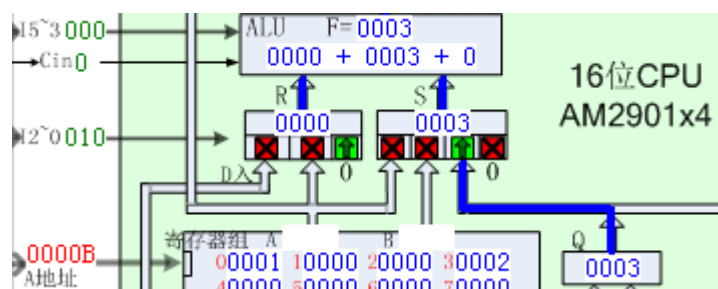
(3.8) 置 I2~0=111, R=D, S=0。按【START】键。



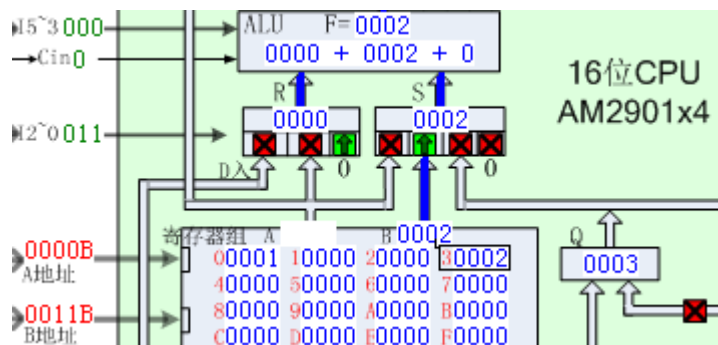
I2~0=000: R=A, S=Q



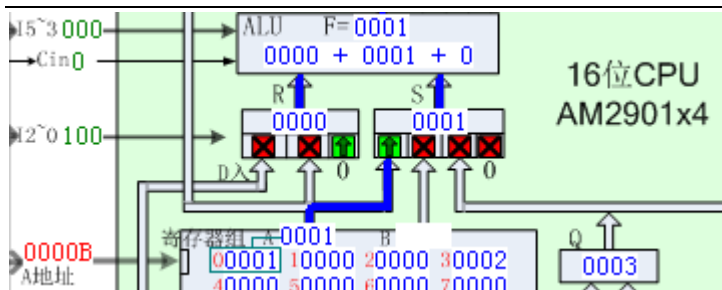
I2~0=001: R=A, S=B



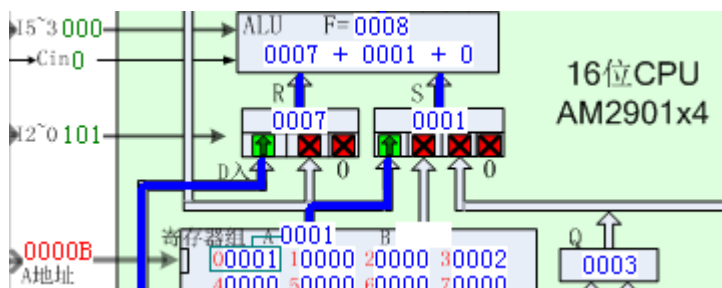
I2~0=010: R=0, S=Q



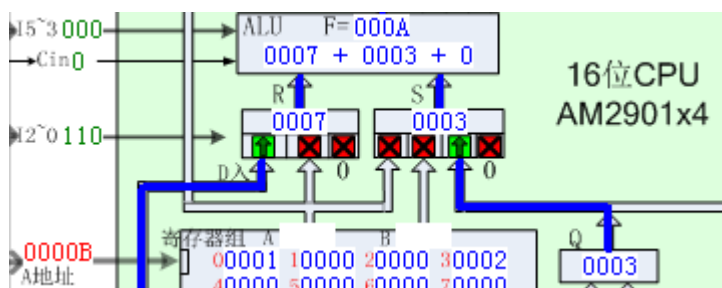
I2~0=011: R=0, S=B



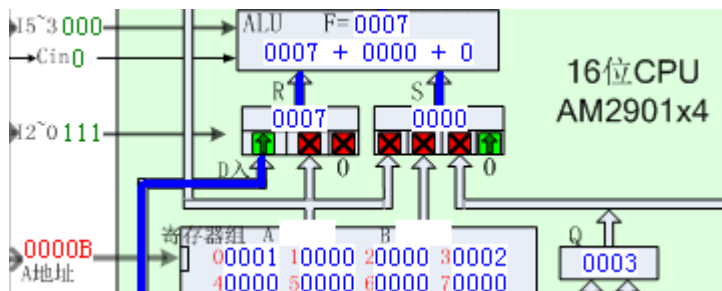
I2~0=100: R=0, S=A



I2~0=101: R=D, S=A



I2~0=110: R=D, S=Q



I2~0=111: R=D, S=0

6、验证 I5~3 所有组合的功能。

(1) 准备工作 1。设置参加运算的数，寄存器 R0、R3 置为不同的数值，通过 I5~3 的不同编码确定 ALU 的运算功能。这里置 R0=1，R3=2。

(1.1) 设置 R0=1

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	1xx	011	000	111	00x	000	无关	0000

16 位数据开关为：0000 0000 0000 0001

按【START】键，观察动画窗口数据流动。

(1.2) 设置 R3=2

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	1xx	011	000	111	00x	000	无关	0011

16 位数据开关为：0000 0000 0000 0010

按【START】键，观察动画窗口数据流动。

(2) 准备工作 2。

为了验证开关 I5~3 的功能，需要将其它功能开关设置为无效或辅助状态。

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	011	1xx	001	000	001	000	000	0000	0011

16 位数据开关无关，可以任意值。

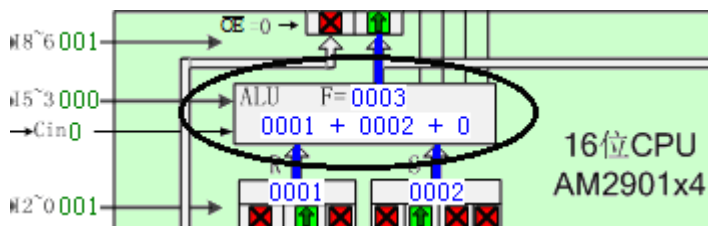
(3) 经过 (1) (2) 步骤后，下面要尝试 I5~3 的所有编码设置。此处 I2~0=001。

进行下列各步操作，观察对象：数据流向，ALU 的运算结果 F，标志位 Cy、Zr、Ov、F15。记录下结果并且分析。

(3.1) 置 I5~3=000，功能 $F = R+S+Cin$ 。

置 SSHSCi=000。(SSHSCi 的值确定 Cin 取值。)

按【START】键。

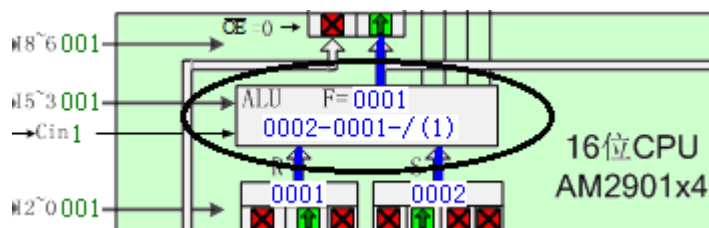


(3.2) 置 I5~3=001，功能 $F = S-R-/Cin$ 。

置 A 口=0000，B 口=0011，

置 SSHSCi=001，

按【START】键。

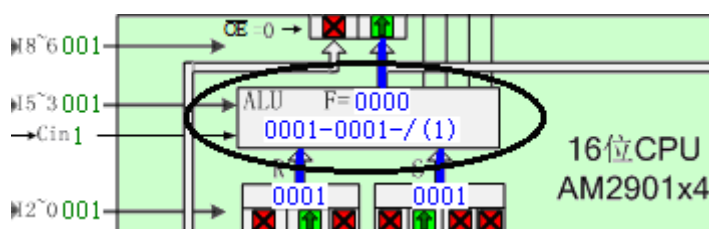


置 I5~3=001,

置 A 口=0000, B 口=0000,

置 SSHSCi=001,

按【START】键。

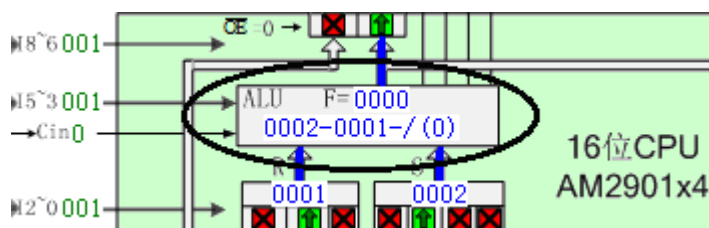


置 I5~3=001,

置 A 口=0000, B 口=0011,

置 SSHSCi=000,

按【START】键。

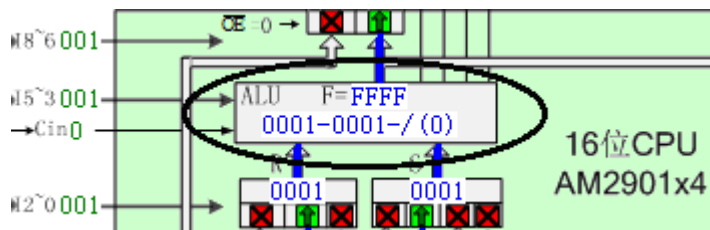


置 I5~3=001,

置 A 口=0000, B 口=0000,

置 SSHSCi=000,

按【START】键。



(3.3) 置 $I5\sim3=010$ ，功能 $F = R - S - Cin$ 。

置 $I5\sim3=010$ ，置 A 口=0000，B 口=0011，置 SSHSCi=001，按【START】键。

置 $I5\sim3=010$ ，置 A 口=0000，B 口=0000，置 SSHSCi=001，按【START】键。

置 $I5\sim3=010$ ，置 A 口=0000，B 口=0011，置 SSHSCi=000，按【START】键。

置 $I5\sim3=010$ ，置 A 口=0000，B 口=0000，置 SSHSCi=000，按【START】键。

(3.4) 置 $I5\sim3=011$ ，功能 $F = R \vee S$ 。置 A 口=0000，B 口=0011，按【START】键。

(3.5) 置 $I5\sim3=100$ ，功能 $F = R \wedge S$ 。置 A 口=0000，B 口=0011，按【START】键。

(3.6) 置 $I5\sim3=101$ ，功能 $F = R \wedge S$ 。置 A 口=0000，B 口=0011，按【START】键。

(3.7) 置 $I5\sim3=110$ ，功能 $F = R \oplus S$ 。置 A 口=0000，B 口=0011，按【START】键。

(3.8) 置 $I5\sim3=111$ ，功能 $F = R \oplus S$ 。置 A 口=0000，B 口=0011，按【START】键。

用同样的方法，进行初始设置后，可以实验如下功能，这里不再一一列举： *

验证 I8~6 所有组合的功能。

设置 SSHSCi 确定 Cin 值并参与算术运算。

设置 SST 直接确定标志寄存器 C 的值。

设置 SST 确定标志位寄存器 CZVS 来源，如 CyZrOvF15 => CZVS（接收 ALU 的标志位输出）、IB（IB15~12）=> CZVS，并验证。

多个字段（I8~6、SSHSCi、SST）联合控制移位运算。

7、A、B 地址，A、B 口

A 地址确定的寄存器只能读；B 地址确定的寄存器既能读又能写。

(1) A、B 口不是“1000”时，A、B 口确定 A、B 地址。

(1.1) 菜单“设置”->“寄存器”，设置 R0=0，R1=1，R2=2，R3=3，R4=4，保存。

(1.2) 从 A 地址确定的寄存器读数据。

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	011	1xx	001	000	100	000	000	0000	无关

16 位数据开关无关，可以任意值。

(1.2.1) 置 A 口=0000，按【START】键，观察动画窗口数据流动。

(1.2.2) 置 A 口=0001，按【START】键，观察动画窗口数据流动。

(1.2.3) 置 A 口=0010，按【START】键，观察动画窗口数据流动。

(1.2.4) 置 A 口=0011, 按【START】键, 观察动画窗口数据流动。

(1.2.5) 置 A 口=0100, 按【START】键, 观察动画窗口数据流动。

(1.3) 向 B 地址确定的寄存器写数据。

微码控制开关为:

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	1xx	011	000	111	000	000	无关	0000

16 位数据开关为: 0000 0000 0000 0101

(1.3.1) 置 B 口=0000, 按【START】键, 观察动画窗口数据流动。

(1.3.2) 置 B 口=0001, 按【START】键, 观察动画窗口数据流动。

(1.3.3) 置 B 口=0010, 按【START】键, 观察动画窗口数据流动。

(1.3.4) 置 B 口=0011, 按【START】键, 观察动画窗口数据流动。

(1.3.5) 置 B 口=0100, 按【START】键, 观察动画窗口数据流动。

(2) A、B 口是“1000”时, 指令寄存器 IRL 确定 A、B 地址。

(2.1) 菜单“设置”->“寄存器”, 设置 R1=1, R2=2, R8=5555, IR=0021。保存。

(2.2)

微码控制开关为:

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	000	1xx	011	000	001	000	000	1000	1000

16 位数据开关无关, 可以任意值。

前面曾经介绍过: 当 AB 口是“1000”时, A、B 锁存器由指令寄存器 IR 的低 8 位 IRL 确定来源寄存器。若 A 口=1000, 则 A 锁存器由 IRL 的低 4 位确定, 即指令格式中的源寄存器 SR。若 B 口=1000, 则 B 锁存器由 IRL 的高 4 位确定, 即指令格式中的目的寄存器 DR。

此时, A、B 锁存器的值不是 A、B 口确定的 R8, 而是 IR 低位 IRL (21) 确定的, 即 DR=R2, SR=R1。本功能用在指令中, 根据指令确定要操作的寄存器。

按【START】键, 观察动画窗口数据流动。

8、综合实验

(1) 完成在程序运行中常用的操作: PC->AR, PC+1->PC。

菜单“设置”->“寄存器”, 置 R5 (PC) = 5。

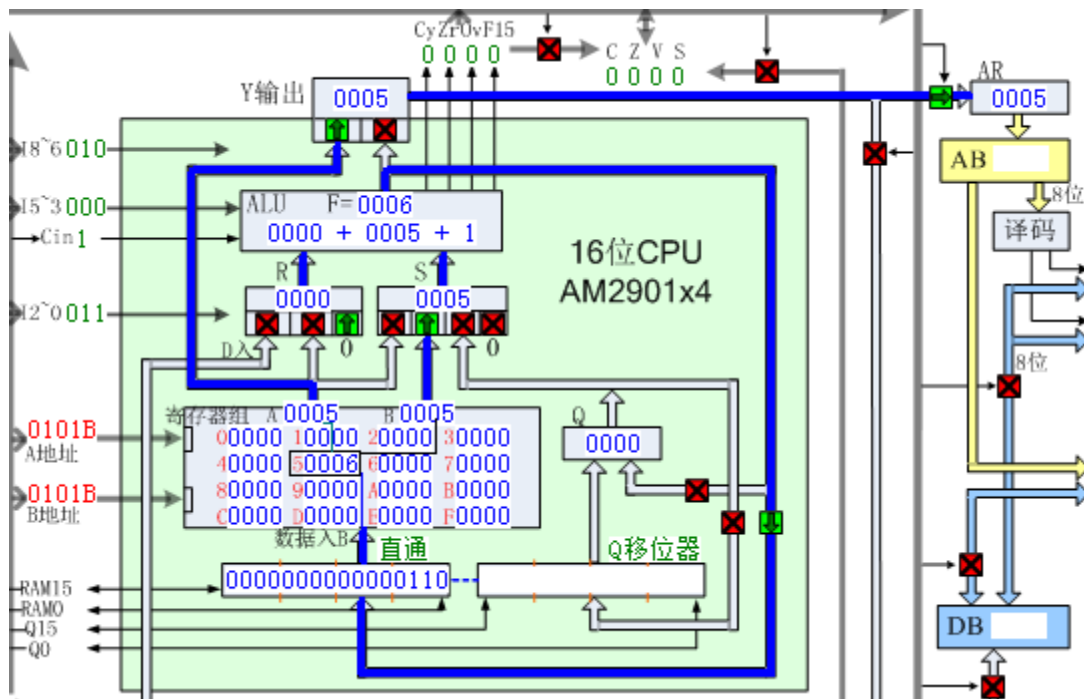
微码控制开关为:

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	011	1xx	010	000	011	000	001	0101	0101

16 位数据开关无关, 可以任意值。

按【START】键, 观察 R5, AR

重复按【START】键，观察 R5，AR.....



每一个循环，寄存器 R5 的值通过 Y 输出到 AR 寄存器，同时 R5 值加 1 运算结果返回 R5。

本例使用了多个微码，组合起来实现一个复杂功能。说明如下：

① 由 I2~0=011 确定运算数分别是“0”与 R[B]，根据 B 地址值 (0101B) 从 R5 寄存器取得数值 0005H 送至 S 端，同时数值“0”送至 R 端。根据 SSHSCi=001 得到 Cin=1。三者准备参与运算。

② 由 I5~3=000 确定运算公式是 $F=R+S+Cin$ ，则 $0000+0005+1$ 得到 $F=0006$ 。

③ 由 I8~6=010 确定 A 锁→Y 输出、F→B 口。根据 A 地址值 (0101B) 从 R5 寄存器取得数值 0005H 送至 A 锁，不经过 ALU 运算，从左边绕过直接送到 Y 输出。为④做准备。

由②得到的运算结果 $F=0006$ 沿着路线直接通过移位器，送到 B 地址 (0101B) 确定的寄存器 R5。完成 $PC+1 \rightarrow PC$ 的功能。

④ 由 DC2=0011 确定 $Y \rightarrow AR$ 。Y=0005 沿着路线送至地址寄存器 AR。完成 $PC \rightarrow AR$ 功能。

总结：上述①②③实现 $PC+1 \rightarrow PC$ 的功能。③④实现 $PC \rightarrow AR$ 功能。

重复按【START】键，则重复上述步骤，R5 和 AR 的值一直递增。

(2) 完成常用的“出栈”操作的一小步：SP-1→SP、AR。

菜单“设置”->“寄存器”，置 R4 (SP) = 5。

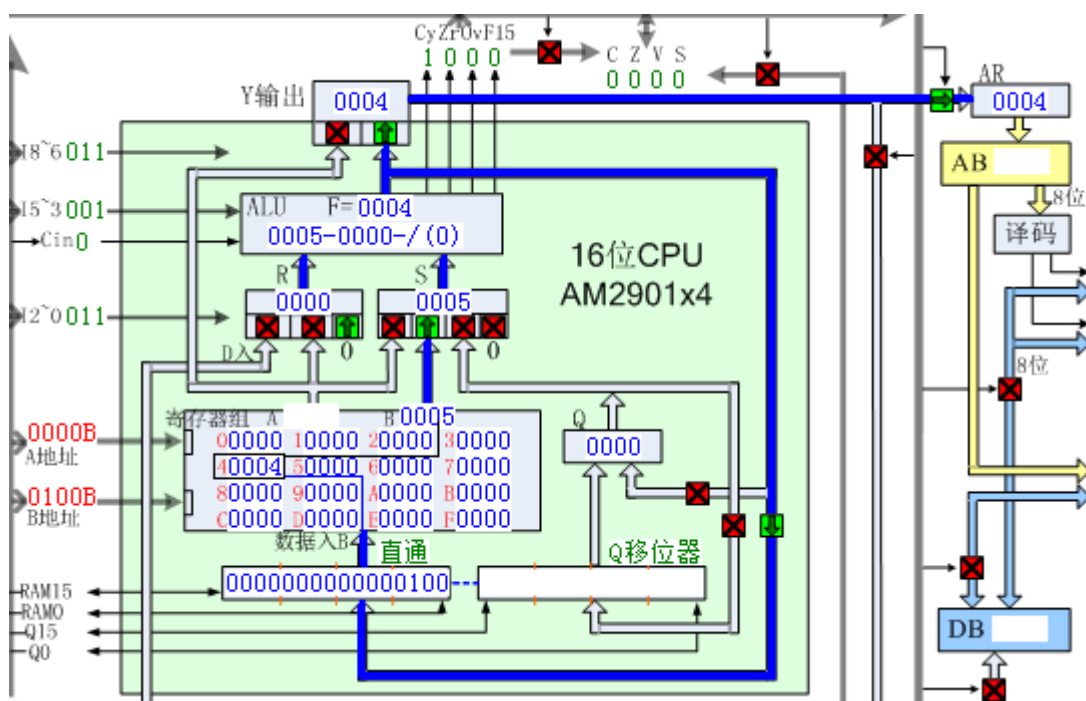
微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	011	1xx	011	001	011	000	000	无关	0100

16 位数据开关无关，可以任意值。

按【START】键，观察 R4，AR

重复按【START】键，观察 R4，AR.....



本例使用多个微码，组合起来实现一个复杂功能。说明如下：

① 由 I2~0=011 确定运算数是“0”与 R[B]，根据 B 地址值 (0100B) 从 R4 寄存器取得数值 0005H 送至 S 端，同时数值“0”送至 R 端。根据 SSHSCi=000 得到 Cin=0。三者准备参与运算。

② 由 I5~3=001 确定运算公式是 $F = S - R - Cin$ ，则 $0005 - 0000 - 1$ 得到 $F = 0004$ 。

③ 由 I8~6=011 确定 $F \rightarrow Y$ 输出、 $F \rightarrow B$ 口。运算结果 $F = 0004$ 向上送到 Y 输出；同时沿着路线直接通过移位器，送到 B 地址 (0100B) 确定的寄存器 R4。完成 $SP-1 \rightarrow SP$ 的功能。

④ 由 DC2=0011 确定 $Y \rightarrow AR$ 。Y=0004 沿着路线送至地址寄存器 AR。完成 $SP-1 \rightarrow AR$ 功能。

总结：上述①②③实现 $SP-1 \rightarrow SP$ 的功能。①②③④实现 $SP-1 \rightarrow AR$ 功能。

重复按【START】键，则重复上述步骤，R4 和 AR 的值一直递减。

【思考题】:

- 1、完成简单算术运算：1 加 2（通过数据开关手动置数）。将操作步骤写出，并截图说明。
- 2、完成以下步骤：
 - (1) R3 和 R9 置数，R3=64H，R9=64H
 - (2) $R3+1 \rightarrow R3$
 - (3) R9 左移一位
 - (4) $(R3+R9)$ 与 Q 联合右移一位，结果分别送至 R3、Q。

4.1.5 实验报告要求

总结模拟硬件 AM2901 运算器相关控制信号的应用。

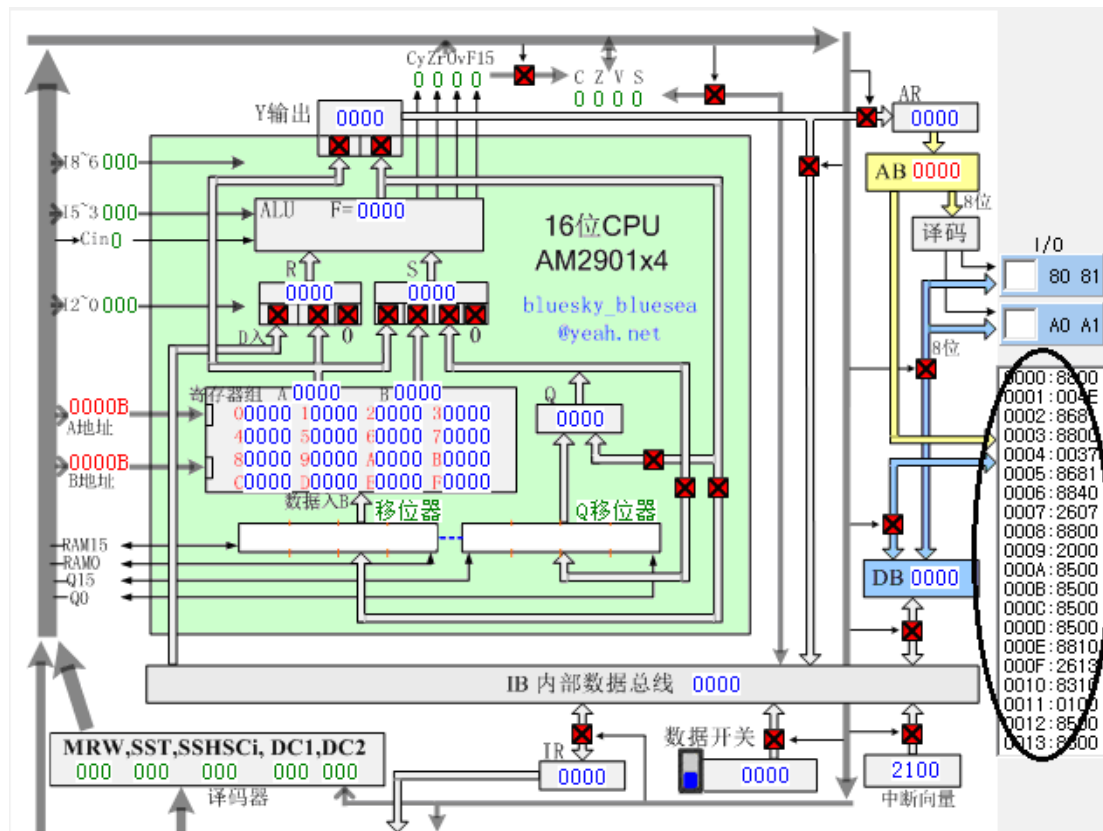
给出思考题操作步骤并截图说明，观察动画数据流动。

第五章 存储器实验

5.1 模拟软件的存储系统硬件介绍

5.1.1 硬件总体框图

模拟软件的存储系统在总体框图的右侧，如图所示：



存储系统

5.1.2 存储器系统

模拟软件是 16 位系统，存储器每单元 16 位；地址 16 位，对应的内存地址空间 0~FFFFH，64K。虽然地址范围是 0~64K，但实现时并没有全部分配存储空间。实际存储空间是 10K 的 RAM 和 8K 的 EEPROM：

(1) 地址 0~3FFFFH 是 RAM，16K，又分为下面两部分：

I、0~2000H 这 8K 的地址是独享的，每个存储单元只对应一个地址。此空间存放着操作系统和 BASIC 语言解释程序，内容比较重要，写入其它数据会导致运行不正常，每次重新运行软件可自行恢复。操作系统还占用 2600H~27FFH 用来存储一些数据，如堆栈数据等。

II、2000H~3FFFFH 分成下面 4 段，每段长 800H (2K)：

2000H~27FFH、2800H~2FFFH、3000H~37FFH、3800H~3FFFH。

这 4 段，都映射到第一段 2000H~27FFH。

RAM 内容的读写方式：可以通过操作系统、自己编写程序，也可以通过菜单直接读取和修改。

(2) 地址 4000H~FFFFH 是 EEPROM，48K。

4000H~FFFFH 分成下面 6 段，每段 2000H（8K）长度：

4000H~5FFFH、6000H~7FFFH、8000H~9FFFH、A000H~BFFFH、C000H~DFFFH、E000H~FFFFH。

这 6 段，都映射到第一段 4000H~5FFFH。

模拟软件用文件保存 EEPROM 内容，可以通过操作系统读写，也可以直接打开文件进行编辑

与内存和输入输出 I/O 接口相关的控制信号是/MIO、REQ、/WE，缩写为“MRW”，见下表：

MRW：与内存、I/O 接口读写有关的控制信号：/MIO、REQ、/WE（MRW）。

/MIO REQ /WE	功能说明
000	写内存。IB=>DB=>MEM
001	读内存。MEM=>DB=>IB
010	写 I/O。IBL=>DBL=>I/O
011	读 I/O。I/O=>DBL=>IBL
1XX	无上述读写操作

本实验中，需要用到其它如总线、中断相关的控制信号如下：

DC1：向 IB 总线写数据的控制信号。

DC1 编码	功能说明
000	无操作
001	ALU 的 Y 输出=>IB
010	IRL=>IB
011	C、Z、V、S、P1、P0=>IB
100	未使用，待扩展
101	中断向量=>IB
110	未使用，待扩展
111	16 位数据开关=>IB

DC2：与寄存器 IR 和 AR 接收、中断有关的控制信号。

DC2 编码	功能说明
000	无操作
001	IB=>IR
010	未使用，待扩展
011	Y 输出=>AR
100	IB=>P1、P0
101	中断新等级=>P1、P0
110	允许中断
111	禁止中断

5.2 模拟软件的存储器实验

5.2.1 实验目的

- 1、深入理解存储器的工作过程。
- 2、写入和读出存储器数据的操作过程。

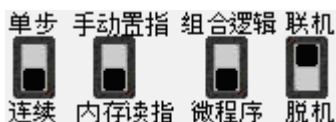
5.2.2 实验内容

- 1、使用操作系统进行存储器读写
- 2、通过菜单对内存直接读写
- 3、手动存储器读写
- 4、编写程序对存储器进行读写

5.2.3 实验步骤

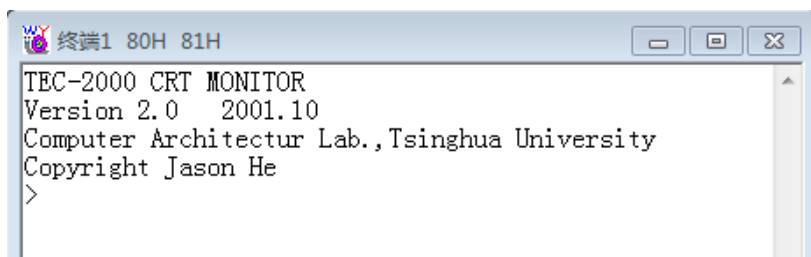
5.2.3.1 操作监控程序对存储器读写

- 1、运行程序 “ZCHPC1.exe”
- 2、置控制开关为 **0011**（连续、内存读指、组合逻辑、联机），
或者置开关为 **0001**（连续、内存读指、微程序、联机）



- 3、设置速度条到最高 

按【RESET】键，再按【START】按键，终端 1 显示：



终端 1 窗口

4、用 E 命令修改存储器内容。

(1) 在命令行提示符状态下输入：

E 2020

屏幕将显示： 2020 内存单元原值：

按如下形式键入：

2020 原值：2222 （空格）原值：3333（空格）原值：4444（空格）原值：5555 回车

(2) 在命令行提示符状态下输入：

D 2020

屏幕将显示从 2020 内存单元开始的值，其中 2020H~2023H 的值为：

2222 3333 4444 5555

注意：用 E 命令连续修改内存单元的值时，每修改完一个，按一下空格键，系统会自动给出下一个内存单元的值，等待修改；按回车键则退出 E 命令。

5.2.3.2 通过菜单读写内存

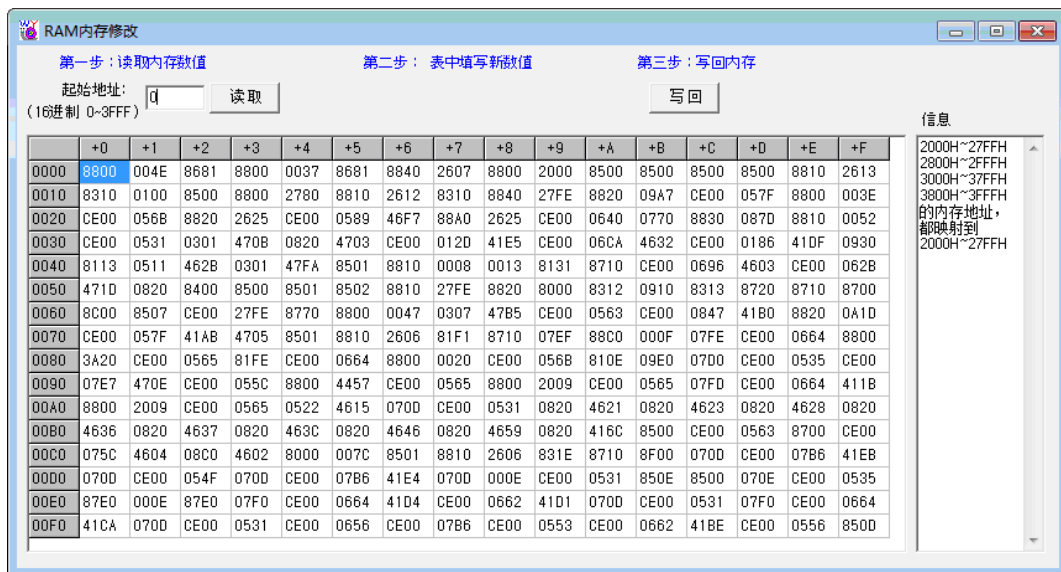
模拟软件可以通过运行监控程序对内存进行读写，也可以通过菜单直接查看和修改内存。

1、菜单“窗口”->“查看内存”，可直接查看内存所有内容。



本窗口能显示 RAM 所有信息，打开窗口后，按“刷新”即可显示。每行显示 16 个单元，每单元 4 位十六进制数。地址空间是 0000H-27FFH，便于观察所有信息。本窗口只能显示，不能更改内容。

2、菜单“设置”->“修改内存”，可修改指定内存单元。



分3步操作，“读取”→“修改”→“写回”。

这里修改的是RAM内容，有效地址0000~27FFH。

本界面一次能读取256个单元内容，修改后再写回去，如果需要修改更多内容，须进行多次读出修改再写入的过程。

5.2.3.3 手动存储器读写

1、运行程序“ZCHPC1.exe”

2、将左下方的控制开关置为1XX0（单步、X、X、脱机）；



3、主界面菜单“窗口”->“显示动画”，打开动画窗口。

4、按【RESET】键，进行初始化。

5、写存储器：向地址2010H的内存写入数值5555H。

(1) 前期准备：通过菜单“设置”->“修改内存”，将地址2010H单元置0，避免先前数据干扰。

(2) 地址寄存器AR置数2010H。

微码控制开关为：（“x”表示任意，可以是“0”或“1”）

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A口	B口
111	011	1xx	001	000	111	00x	000	无关	无关

16位数据开关为：0010 0000 0001 0000

按【START】键，观察数据流动。

注：为缩短实验操作时间，可直接设置相关寄存器数值。方法是：主界面菜单“设置”->“寄存器”，在相应位置直接填入需要的数值保存即可，注意是十六进制。

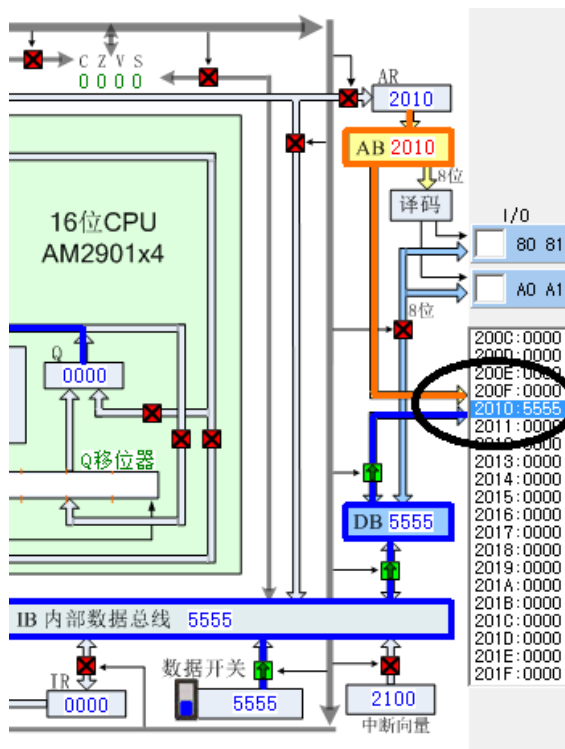
(3) 内存 2010H 写入数值 5555H

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	000	001	无关	无关	000	无关	无关	无关

16 位数据开关为：0101 0101 0101 0101

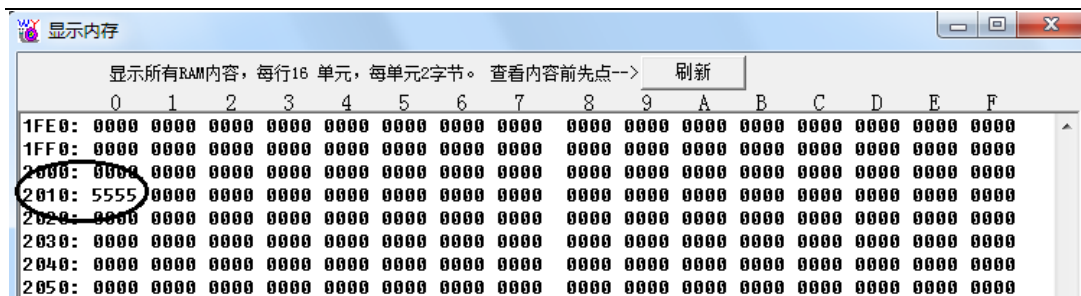
按【START】键，观察数据流动。



数据 5555H 存入内存单元 2010H

可以观察到数据 5555 通过 IB 和 DB 到达内存，送入地址为 2010H 单元内。

(4) 通过菜单“窗口”->“查看内存”，查找地址 2010H，验证是否为 5555H。



读存储器内容的实验请自行完成。

【思考题】

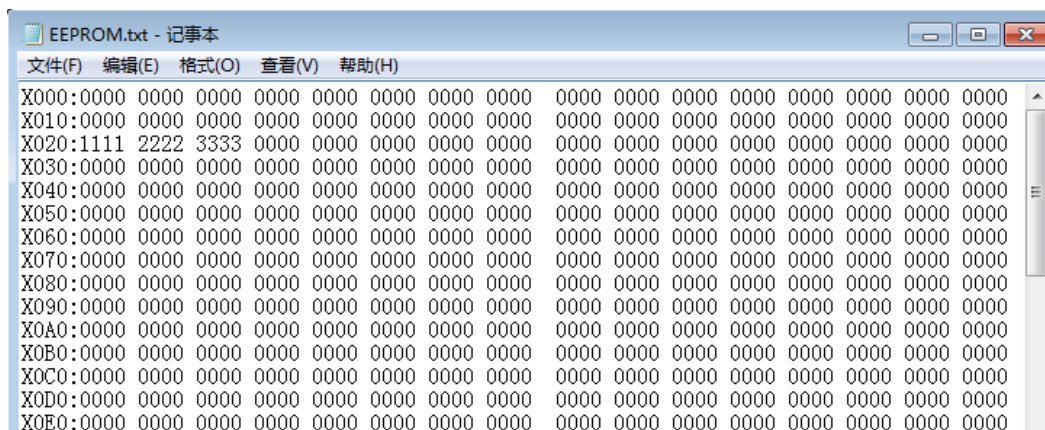
- 1、数据从内存到寄存器走的路径很复杂，还经过了运算，有没有更简洁的途径？
- 2、如何实现将寄存器 R5 的值送至内存 2020H 单元？

EEPROM 存储文件介绍：

模拟软件将 EEPROM 内容存储在文件“EEPROM.txt”中。直接修改文件即可改变内容，运行程序“ZCHPC1.exe”时会自动读取该文件。实验时可以通过程序读写，也可以通过操作系统读写。

EEPROM 的地址范围是 4000H~FFFFH，共 48K，每个单元都是 16 位。在模拟软件中，EEPROM.txt 文件只实现了 8K 的空间，地址是“X000~XFFF”与“Y000~YFFF”。则地址 4000H~5FFFH、6000H~7FFFH、8000H~9FFFH、A000H~BFFFH、C000H~DFFFH、E000H~FFFFH，自动映射到这两段。例如地址 4000H~5FFFH，其 4000H~4FFFH 映射到 X000~XFFF，5000H~5FFFH 映射到 Y000~YFFF。在进行文件修改时，需要自己换算对应的地址。

文件内部格式如下：



文件第一列是地址，每行 16 个单元，每单元 4 位十六进制数。修改时直接编辑保存即可。

7、EEPROM 单元写入数值：向地址为 4010H 的 EEPROM 写入数值 5555H。

(1) 前期准备：用软件“记事本”打开文件“EEPROM.txt”，将地址 X010 内容置“0000”，保存。

(2) 地址寄存器 AR 置数 4010H。

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	011	1xx	001	000	111	00x	000	无关	无关

16 位数据开关为：0100 0000 0001 0000

按【START】键，观察数据流动。

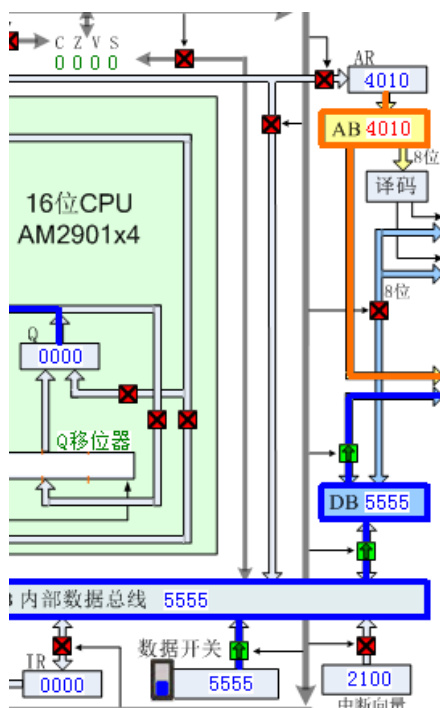
(3) EEPROM 的 4010H 单元内写入数值 5555H

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	000	001	无关	无关	000	无关	无关	无关

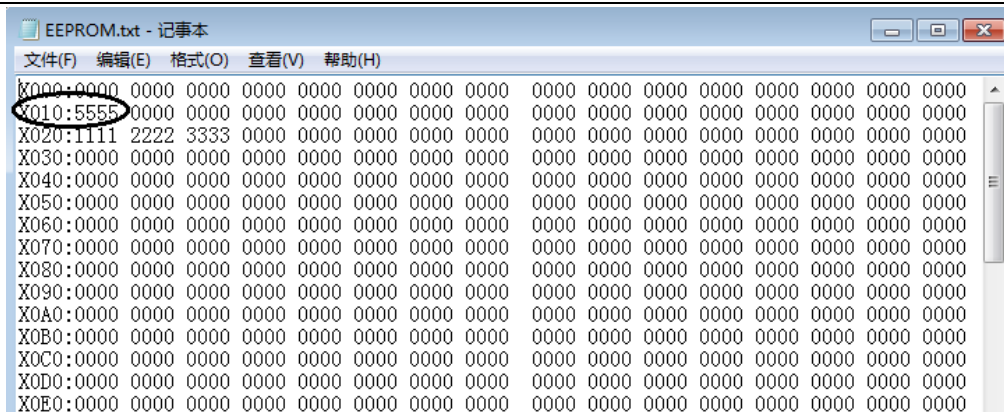
16 位数据开关为：0101 0101 0101 0101

按【START】键，观察数据流动。



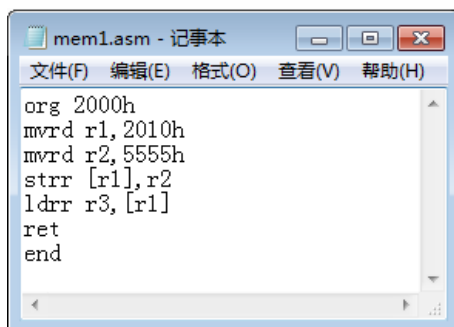
数值写入 EEPROM 单元

(4) 用软件“记事本”打开文件“EEPROM.txt”，观察地址 X010H 单元内容。



5.2.3.4 编程对存储器读写

- 1、用记事本编写汇编程序“mem1.asm”并保存，内容如下：



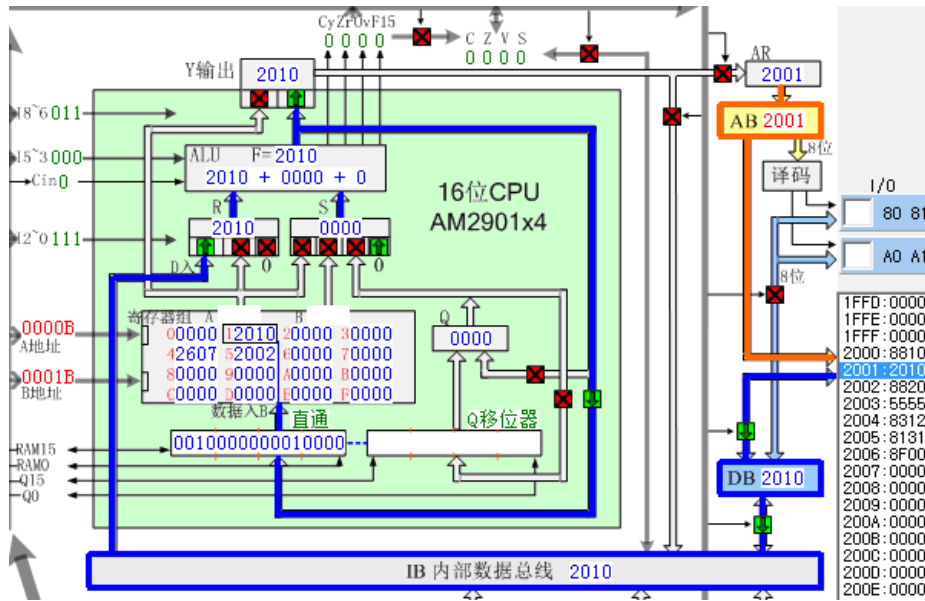
功能是将数据 5555H 存入到内存 2010H 单元，再读出写入寄存器 R3。

- 2、运行程序“ZCHPC1.exe”
- 3、主界面菜单“设置”->“PC 初始值”，确保窗口内选择如下图

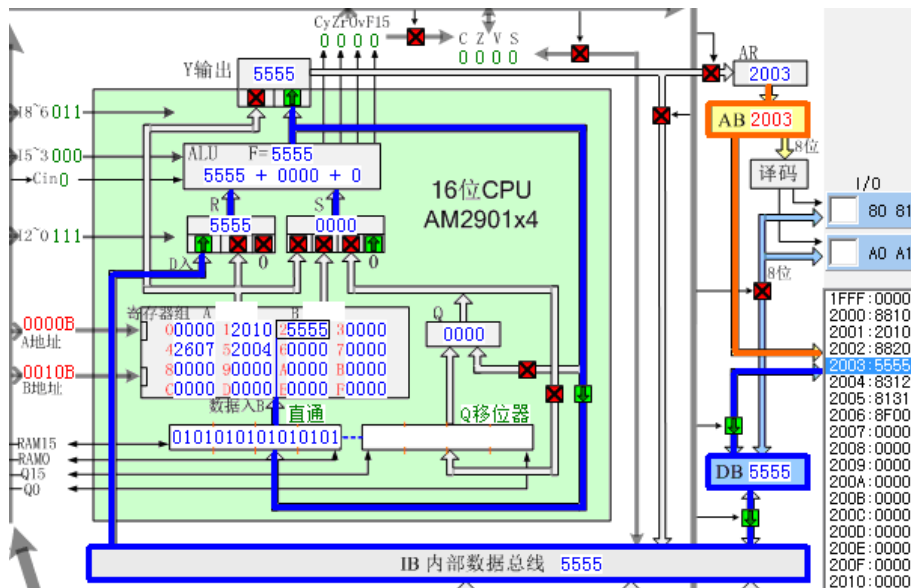


- 4、置控制开关为 0011（连续、内存读指、组合逻辑、联机），
或者置开关为 0001（连续、内存读指、微程序、联机）
- 5、通过“设置”菜单，将内存地址 2010H 单元内容置 0；将寄存器 R1、R2、R3 置 0。
- 6、主界面菜单“文件”->“打开 ASM 源文件”，打开“mem1.asm”
按“编译”后，再按“导入内存”即可。
- 7、按【RESET】键，进行初始化。
- 8、按“单节拍”或者“单指令”按钮，可以看到程序一步一步运行。
运行到指令“RET”时。查看 R3 的数值，通过“窗口”菜单观察内存 2010 数值。
运行过程中若打开动画窗口，可看到数据 5555H 从寄存器 R2 流动到内存，再流回到 R3。
- 9、若想再次运行程序，按下【RESET】键后再按“单节拍”或者“单指令”即可重新运行。

注意：按“单指令”键是不会有动画的，按“单节拍”可以启动动画，可以观察每个节拍的数据流向。

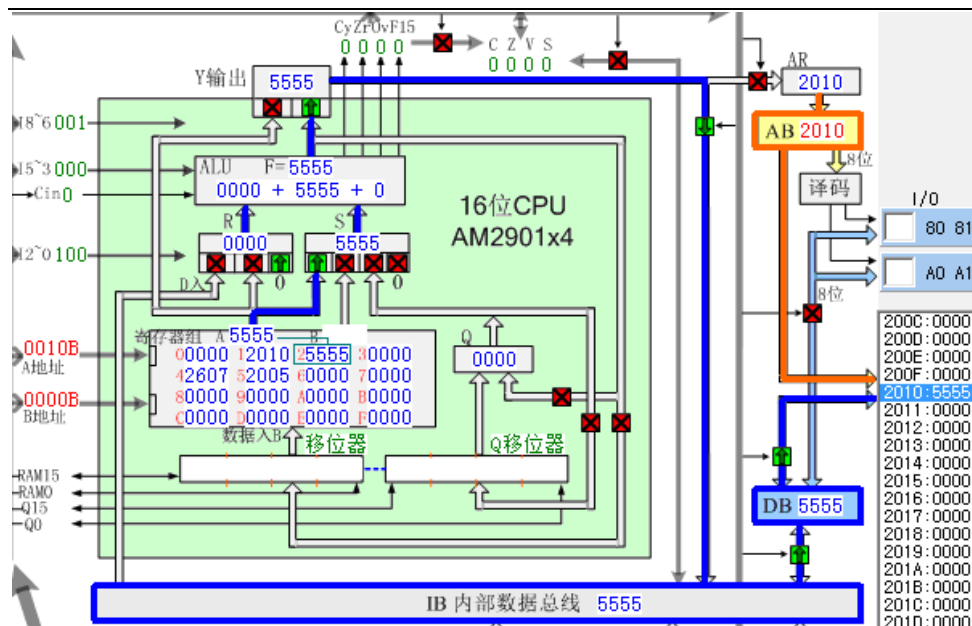


“mvr r1,2010h”实现时动画图



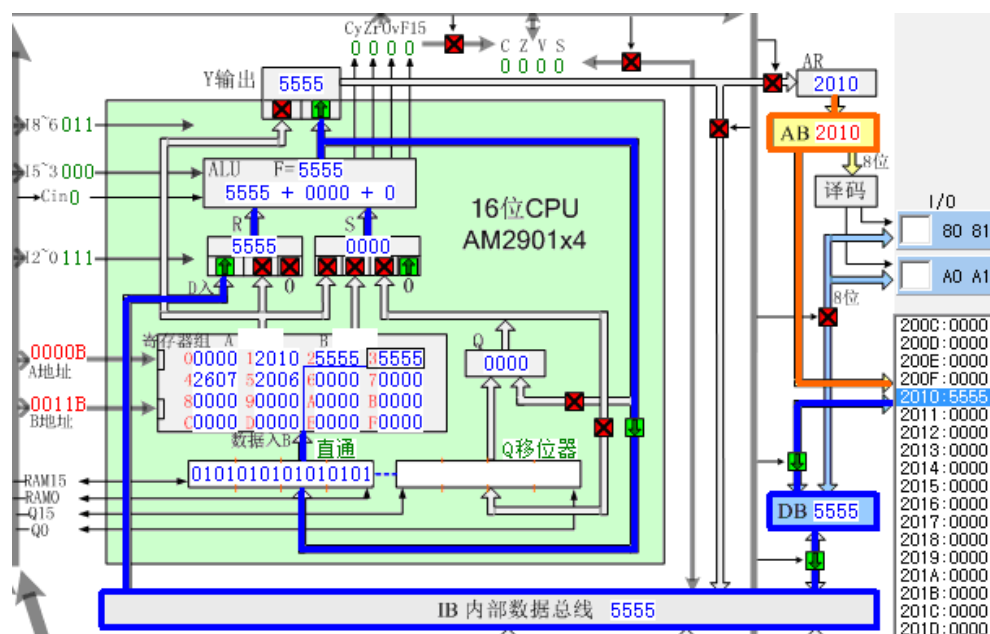
“mvr r2,5555h”实现时动画图

内存 2003H 单元数值 5555H 送入寄存器 R2。



“str [r1], r2” 实现时动画图

寄存器 R2 值送入内存 2010H 单元。



“ldrr r3, [r1]” 实现时动画图

内存 2010H 单元值 5555H 送入寄存器 R3。

【思考题】:

- 1、将数值 000FH 写入 5555H 单元中，应该进行怎样的操作才能实现？
- 2、编写程序：在 5000H~500FH（EEPROM）单元中依次写入数据 0000H、0001H、……、000FH。如何利用模拟软件完成，观察结果的正确性。

5.2.4 实验报告要求

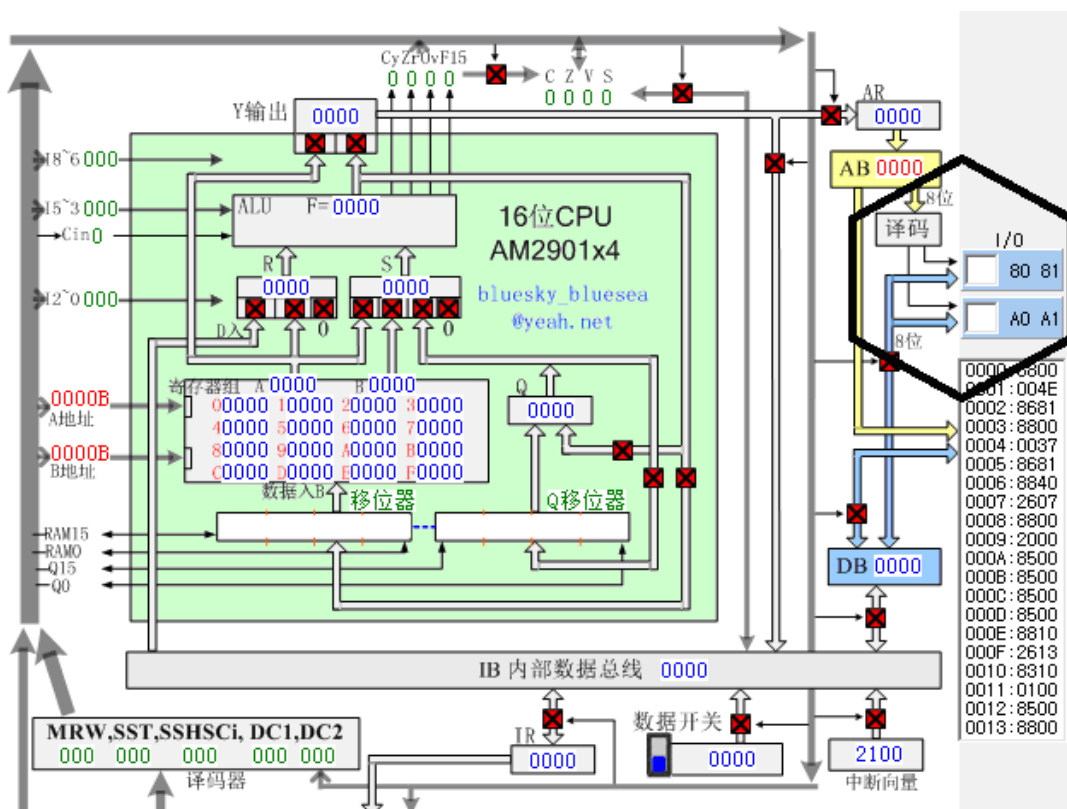
- 1、记录操作过程和数据以及遇到的问题及解决方法。
- 2、解答本教材中的**【思考题】**，写出操作步骤，记录数据，书写完整的实验过程。

第六章 输入输出 I/O 接口实验

6.1 模拟软件的输入输出 I/O 接口硬件介绍

6.1.1 硬件总体框图

模拟软件的输入输出 I/O 接口以及控制部件在总体框图的右侧，如图所示：

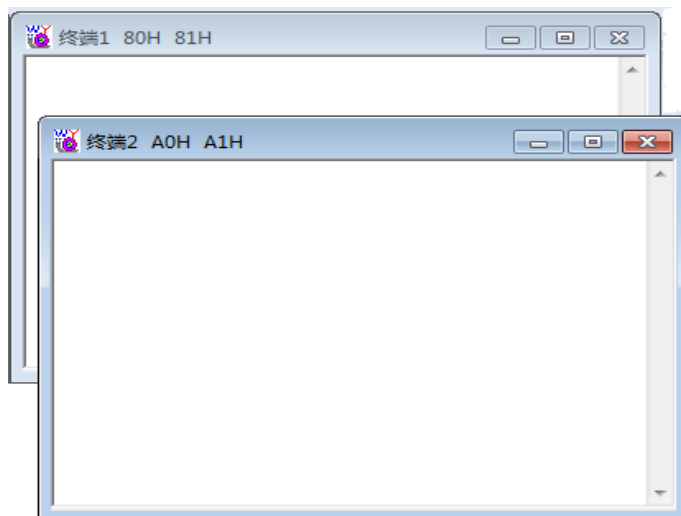


输入输出 I/O

6.1.2 输入输出 I/O 接口系统

模拟软件中的 I/O 接口地址和数据都是 8 位。地址范围 0~FFH，但只有 4 个地址有效，分别是 80H、81H、A0H、A1H，不能更改。在总体框图上可看到分成 2 组：80H、

81H 一组 ，A0H、A1H 一组 ，系统将它们接在两个终端界面，可通过菜单“窗口”->“显示终端”打开，如下图：



模拟软件通过菜单“窗口”->“显示终端”打开窗口即可直接使用，两个终端窗口可同时使用。

模拟软件对终端的通信控制，模拟了 8251 芯片的部分功能。对端口进行操作时，要进行端口检测，对返回的数据进行判断，以确定下一步操作。

“终端 1”：地址 80H 为数据寄存器，地址 81H 为控制与状态寄存器。

“终端 2”：地址 A0H 为数据寄存器，地址 A1H 为控制与状态寄存器。

以终端 1 为例，对 81H 口的数据读写是用来控制 I/O 接口以及返回端口状态信息。写入 80H 的数据会直接显示在终端 1 窗口，读取 80H 得到的数据来自在终端 1 窗口的按键。在读 80H 口前，要先读 81H 口状态值，如果返回“00000001”，则表示 80H 口没有输入；如果返回“00000010”，则表示 80H 口有输入，此时可以读取 80H 口。写 80H 口则不需要判断，可以直接写。

与内存和输入输出 I/O 接口相关的控制信号是 /MIO、REQ、/WE，缩写为“MRW”。

MRW：与内存、I/O 接口读写有关的控制信号：

/MIO REQ /WE	功能说明
000	写内存。IB=>DB=>MEM
001	读内存。MEM=>DB=>IB
010	写 I/O。IBL=>DBL=>I/O
011	读 I/O。I/O=>DBL=>IBL
1XX	无上述读写操作

本实验中，需要用到其它如总线、中断相关的控制信号如下：

DC1：向 IB 总线写数据的控制信号。

DC1 编码	功能说明
000	无操作
001	ALU 的 Y 输出=>IB
010	IRL=>IB
011	C、Z、V、S、P1、P0=>IB
100	未使用，待扩展
101	中断向量=>IB
110	未使用，待扩展
111	16 位数据开关=>IB

DC2: 与寄存器 IR 和 AR 接收、中断有关的控制信号。

DC2 编码	功能说明
000	无操作
001	IB=>IR
010	未使用，待扩展
011	Y 输出=>AR
100	IB=>P1、P0
101	中断新等级=>P1、P0
110	允许中断
111	禁止中断

6.2 模拟软件的输入输出 I/O 接口实验

6.2.1 实验目的

- 1、了解输入输出 I/O 的工作过程
- 2、对输入输出 I/O 进行写入和读出数据

6.2.2 实验内容

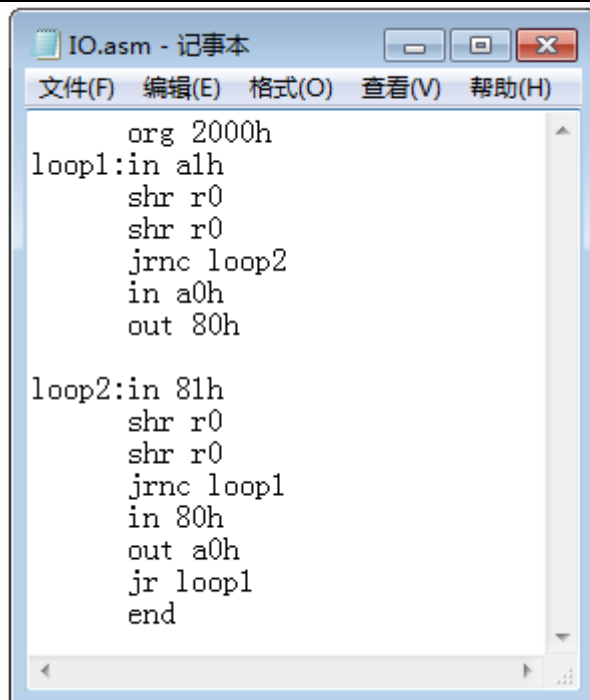
- 1、编程进行双端口通讯
- 2、手动进行输入输出 I/O 实验

6.2.3 实验步骤

6.2.3.1 编程进行双端口通讯

- 1、用记事本编写汇编程序“IO.asm”并保存。

功能：终端 1（80H）和终端 2（A0H）互相显示对方键入的字符。



```

org 2000h
loop1: in alh
      shr r0
      shr r0
      jrncl loop2
      in a0h
      out 80h

loop2: in 81h
      shr r0
      shr r0
      jrncl loop1
      in 80h
      out a0h
      jr loop1
end

```

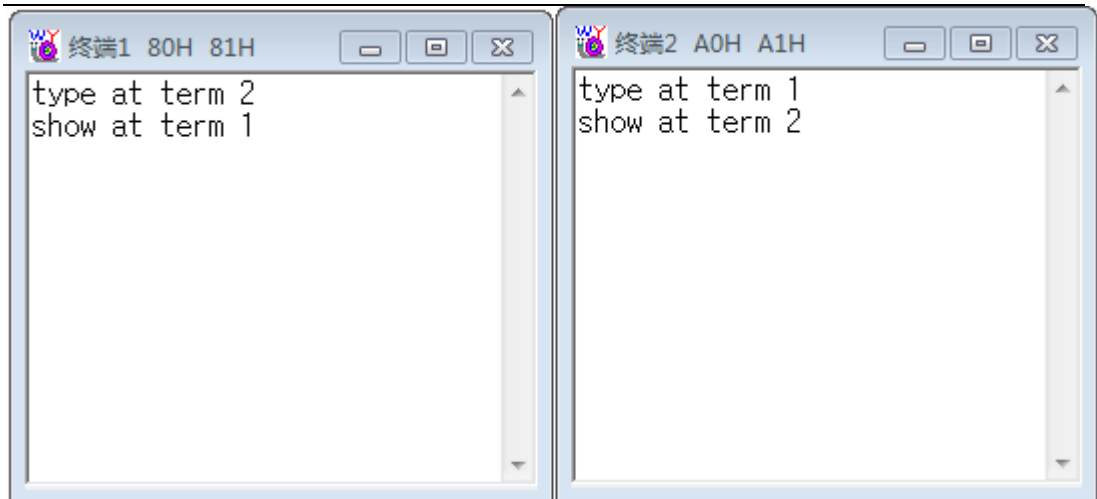
- 2、运行程序 “ZCHPC1.exe”。通过“窗口”菜单显示两个终端窗口。
- 3、主界面菜单“设置”->“PC 初始值”，选择“启用 PC 初始值”。



- 4、置控制开关为 **0011**（连续、内存读指、组合逻辑、联机），
或者置开关为 **0001**（连续、内存读指、微程序、联机）



- 5、主界面菜单“文件”->“打开 ASM 源文件”，打开“IO.asm”。按“编译”后，再按“导入内存”。
- 6、按【RESET】键。
- 7、按【START】键，此时在终端 1 窗口键入字符，在终端 2 窗口显示；而在终端 2 窗口键入字符，在终端 1 窗口显示。

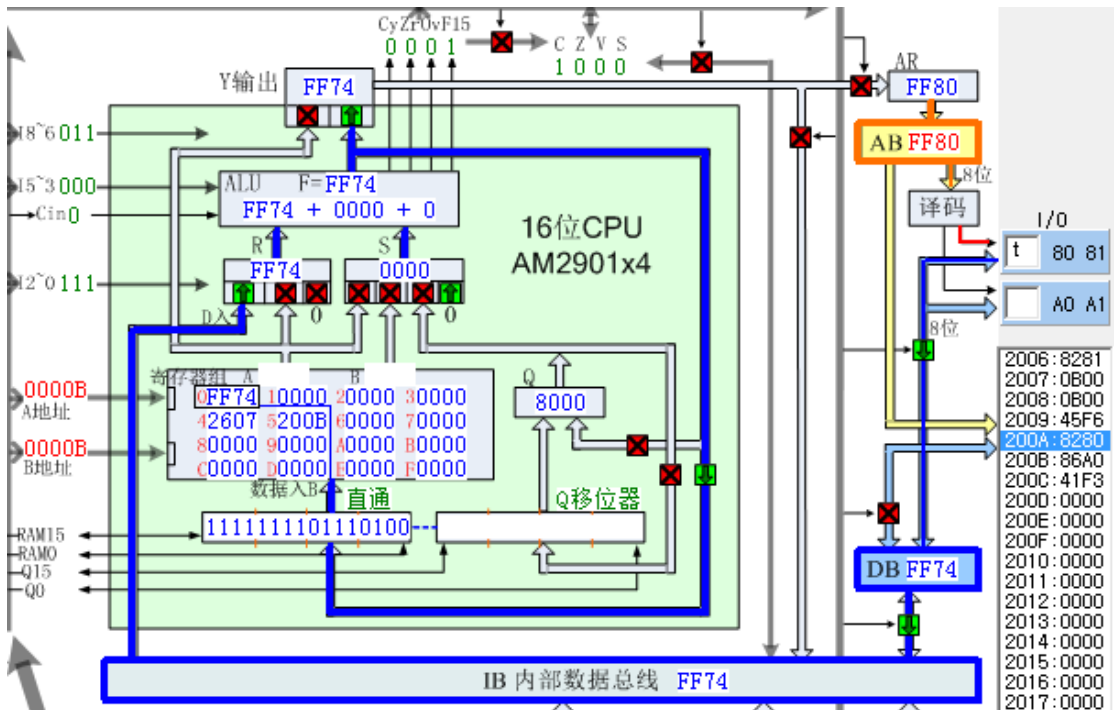


双终端通讯窗口显示

如果将主界面速度调至低于 20,

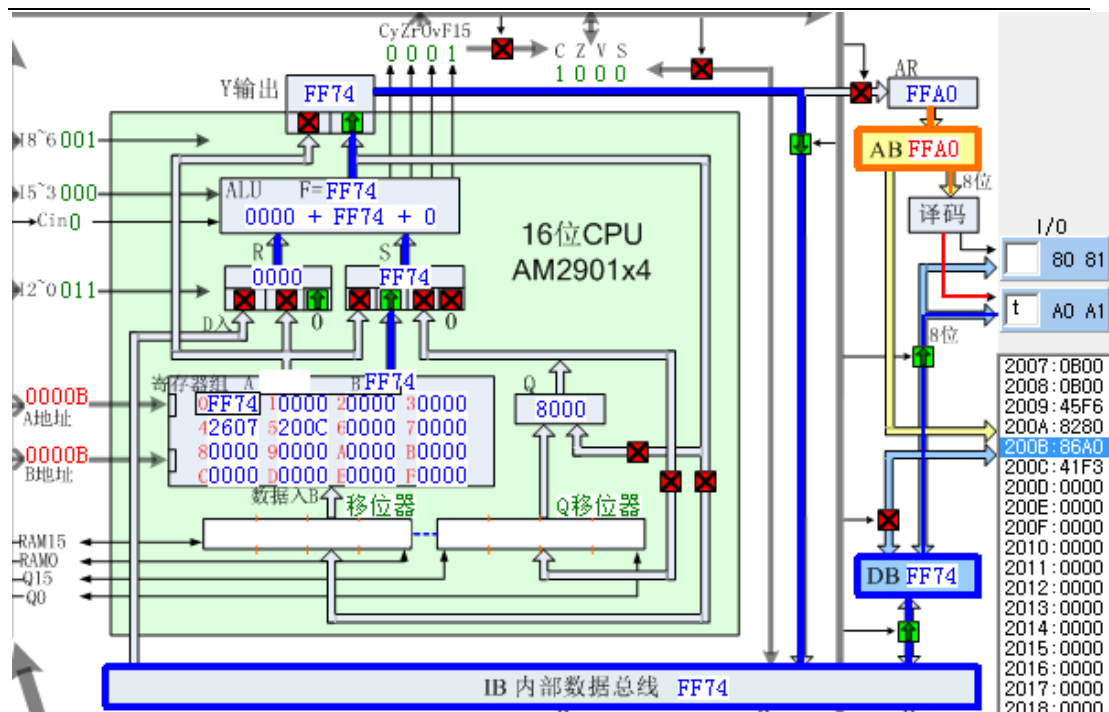


再打开动画界面，可观察到两个 I/O 口是如何交换数据的。（先调速度再打开动画界面）



字符“t”存入 R0

终端 1 窗口键入“t”，则从 I/O 80 地址读取字符“t”，存储在寄存器 R0。



字符“t”输出端口 A0

从寄存器 R0 读取“t”的数值，送往 I/O A0，显示在终端 2 窗口。

6.2.3.2 手动输入输出 I/O 实验

- 1、运行程序 “ZCHPC1.exe”
- 2、将左下方的控制开关置为 1XX0（单步、X、X、脱机）；



- 3、主界面菜单“窗口”->“显示动画”，打开动画窗口。
- 4、按【RESET】键，进行初始化。

- (1) 地址寄存器 AR 置数 xx80H。“x”可以任意。

微码控制开关为：（“x”表示任意，可以是“0”或“1”）

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A □	B □
111	011	1xx	001	000	111	00x	000	无关	无关

16 位数据开关为:	XXXX XXXX 1000 0000
------------	---------------------

按【START】键，观察数据流动。

本步也可通过菜单“设置”->“寄存器”直接设置AR值。

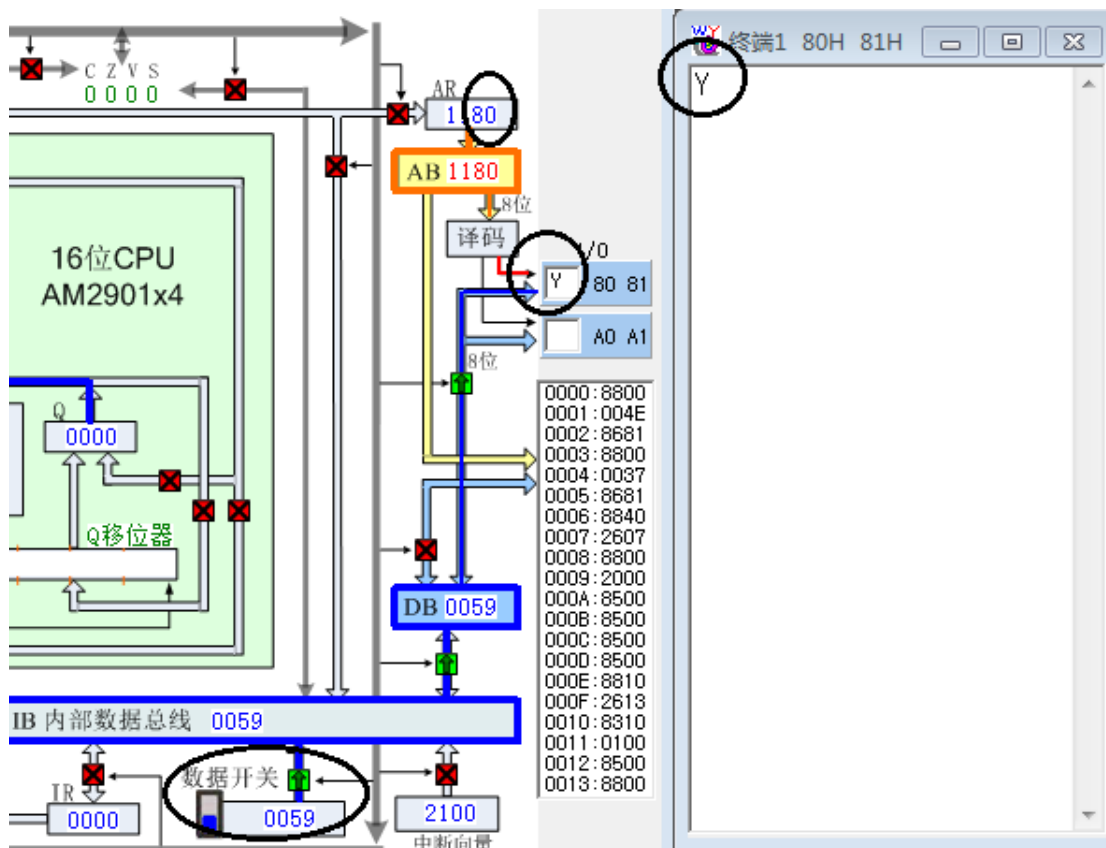
(2) 字符“Y”给 80H

微码控制开关为:

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
111	000	010	001	无关	无关	000	无关	无关	无关

16 位数据开关为: `xxxx xxxx 0101 1001`, 字符“Y”的 ASCII 码。

按【START】键, 观察数据流动, 观察终端 1 窗口显示。



字符“Y”输出到终端 1

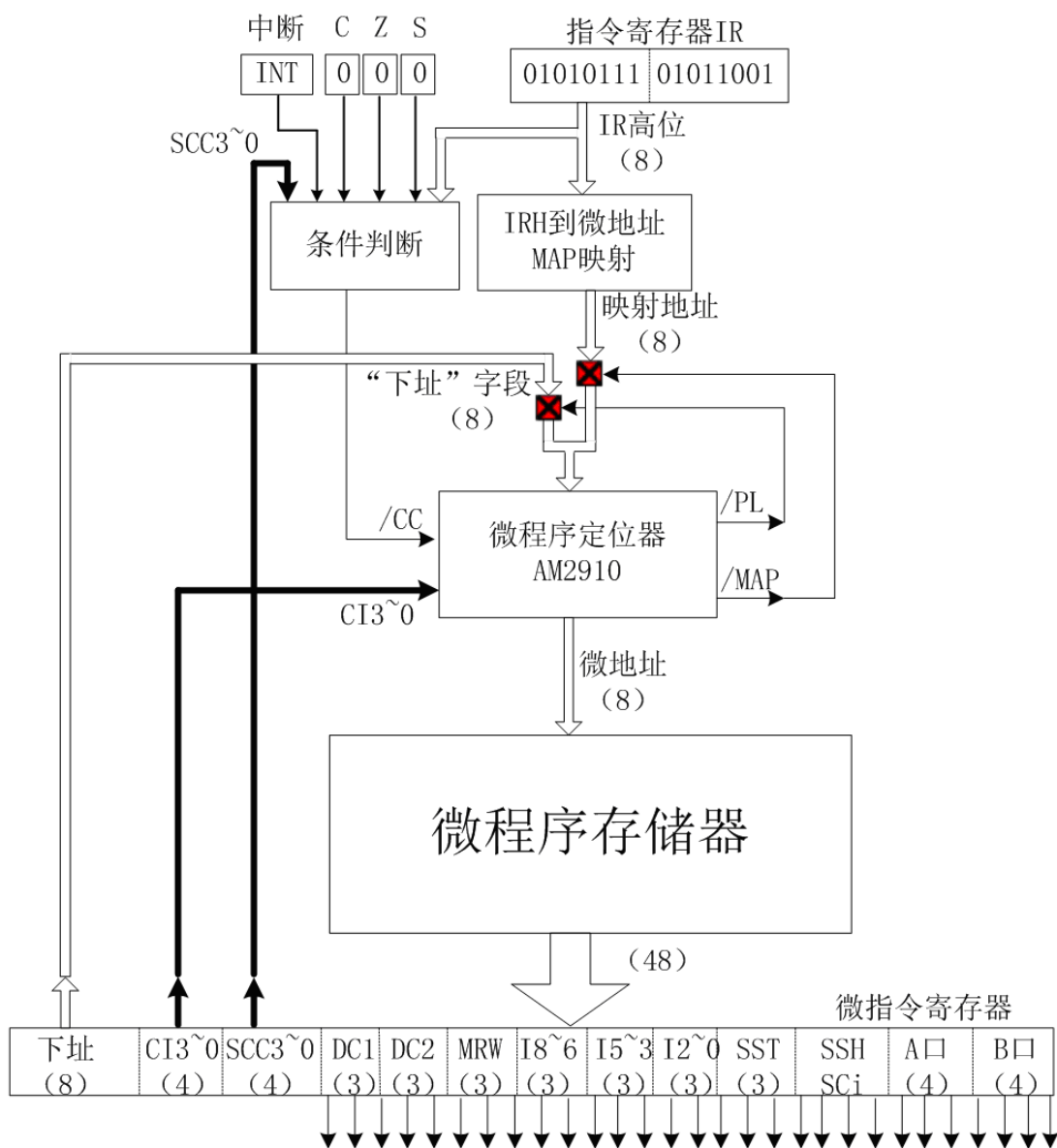
读取终端窗口数据的实验请自行完成。

第七章 微程序控制器实验

7.1 模拟软件的微程序控制器实验

7.1.1 微程序控制器工作原理

微程序控制器的工作流程如图，主要由条件判断线路、微指令地址映射部件、微程序定位器 AM2910、微程序存储器、微指令寄存器组成。



微程序控制器的工作流程

1、微指令寄存器

微指令寄存器用于存放当前微指令的内容。将各段微码送至相应控制部件。

2、IRH 到微地址 MAP 映射部件

其输入是指令寄存器 IR 给出的 8 位指令操作码 IRH，输出是对应的微程序的入口地址。每次 IR 得到新指令后，需要映射得到微程序入口地址才能实现其功能。

3、条件判断线路

产生微程序定序器 AM2910 的 /CC 信号。

当 AM2910 执行微程序转移的功能时，需要判断转移条件是否成立，由“条件判断部件”完成。根据控制信号 SCC3~0 指定的判断条件，结合指令操作码 IRH、中断信息、标志位寄存器 CZS 等，得出“/CC”并送至 AM2910。如果符合转移条件，就转移到微指令的“下址”字段；不符合转移条件，就顺序执行，即“当前微地址+1”。

注意不是每步都需要进行“转移条件判断”，只有当 CI3~0=0011 时 AM2910 芯片才接收判断结果。

4、下一步微地址的形成

由 AM2910、MAPROM、条件判断部件共同实现。AM2910 最主要的功能是产生当前微指令的下一步微地址，这是由 AM2910 内部的 1 个 4 输入多路地址选择器实现的。

总结：当前微指令的下一步微地址的来源有三个：微指令中的“下址”字段、映射部件 MAPROM、AM2910 内部的微程序计数器。

7.1.1.1 AM2910 相关知识简介

AM2910 是一片能提供 12 位微指令地址的器件，即它的输入输出的地址位数和器件内的部件位数均为 12 位，能直接寻址 4096 条微指令字的空间范围，模拟软件只用低 8 位，地址空间 256。

AM2910 包括一个四输入的多路地址选择器，用来从内部寄存器/计数器（R/C），外部直接输入（D），内部微程序计数器（μPC）或内部微堆栈（F）四个输入中，选择其一作为下一条微指令的地址。本模拟软件没有使用“内部微堆栈”，实际是“三选一”。

AM2910 输出使能信号：/PL、/MAP，用以决定直接输入 D 的来源。

当 /PL 有效时（即 /PL=0），D 来源于微指令的下地址字段，用于实现微程序转移；/PL 受输入信号 /CC 控制。

当 /MAP 有效时（即 /MAP=0），D 来源于 MAPROM，用于实现从机器指令的操作码到相应的微程序段首地址的转移；/MAP 受输入信号 CI3~0 控制。

AM2910 提供了 16 条命令，用来控制 AM2910 内部的操作和选择下一条将要执行的微指令的地址。其中有些命令（如 0、2、14 号命令）的执行结果仅由命令码本身决定，其它命令还都要受到测试条件（例如 /CC）为真还是为假的控制，现将设计中要用到的 0、2、3、14 命令的功能和我们的具体用法说明如下：

0 号命令（CI3~0=0000）：用于初始化，即无条件使微地址为“0”，用于系统加电时，

确保此时系统从 0 号微地址开始执行微程序。

2 号命令 (CI3~0=0010)：用于指令功能分支，即输出信号/MAP 为有效，映射结果作为输出微地址值，实现指令操作码找到对应微程序段的入口地址，从而开始该条指令的特定的执行过程。

3 号指令 (CI3~0=0011)：用于条件微转移控制，当条件成立，即/CC 为低时，用/PL 把微指令字中的“下址”字段的内容(转移地址)作为下一步微地址，实现微程序转移。当/CC 为高时，微程序顺序执行，即把已增 1 后的微指令地址作为下地址。若设置 SCC3~0 使得送入的/CC 的状态肯定为低，本指令可以用作实现微程序无条件跳至指定微地址。

14 号命令 (CI3~0=1110)：顺序执行，即执行紧跟在本条微指令后面的那条微指令。

微程序专用的控制信号简要说明：

1) 微程序下址(8 位)：当前微程序的“下址”字段。是当前微指令的“下一步”的选择之一，是否选它作为真正的“下一步”要根据当前微程序的字段 CI3~0 和 SCC3~0 值，还要判断当前系统状态是否满足要求，详细内容见下面。

2) CI3~0：为 AM2910 提供命令码。CI3~0 功能表

CI3~0	功能
0000 (0 号命令)	AM2910 初始化，下一步微地址=00。
0010 (2 号命令)	MAP 映射，/MAP 信号有效。下一步的微地址来自 MAP 映射地址。
0011 (3 号命令)	条件微转移，使输入的/CC 起作用，下一步微地址由 SCC 确定。见下表。
1110 (14 号命令)	顺序执行，下一步微地址是当前微地址加 1。
CI3~0 其它值	错误。下一步微地址=00。

3) SCC3~0：给出形成 AM2910 的/CC 信号的条件码。当上表中 CI3~0=0011 时本表有效

SCC3~0	<p>下面的条件满足时，使/CC=0，即/PL 有效，即下一步微地址来自当前微指令的“下址”字段。相当于“跳转执行”。</p> <p>下面的条件不满足时，使/CC=1，则下一步微地址是当前微地址加 1，即“顺序执行”。</p>
0000	必转，无条件跳转“下址”
0010	当有新中断请求即/INT=0 时，转“下址”
0100	JRC、JRNC、JRZ、JRNZ 条件不成立时，转“下址”
0101	JRS、JRNS 条件不成立时，转“下址”
0110	IR10=0 时，转“下址” (为 IN/OUT 指令服务)
0111	IR8=1 时，转“下址” (为 PSH/F, POP/F 指令服务)
其它值	判定为不满足条件，不转，顺序执行。

7.1.1.2 微程序入口地址映射表

参见软件文件夹的“InsMap.txt”。



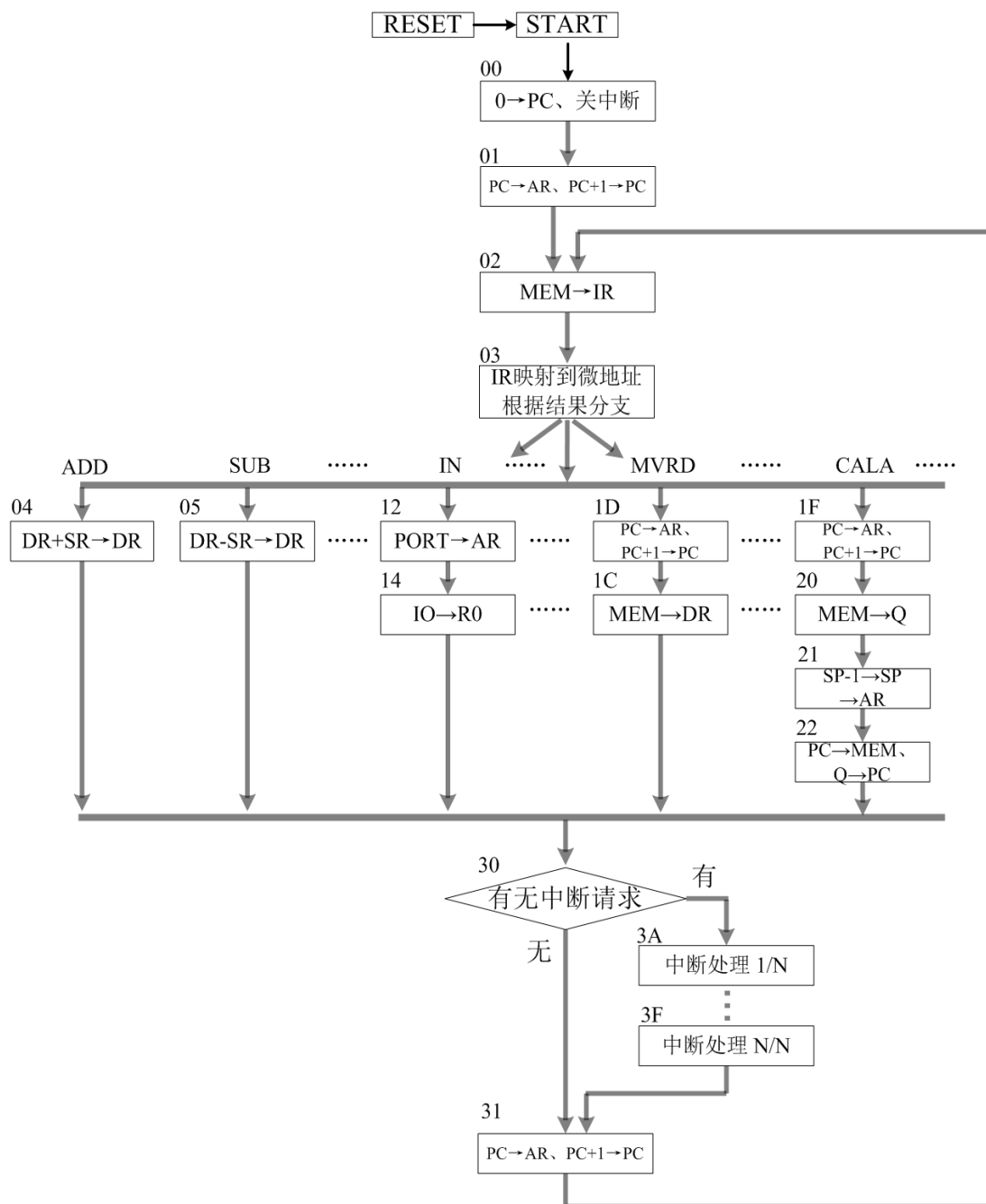
编码	入口	指令
0000 0000	04	; ADD DR, SR
0000 0001	05	; SUB DR, SR
0000 0010	06	; AND DR, SR
0000 0110	07	; OR DR, SR
0000 0100	08	; XOR DR, SR
0000 0011	09	; CMP DR, SR
0000 0101	0A	; TEST DR, SR
0000 0111	0B	; MVRR DR, SR

模拟软件运行时会自动读取本文件，如果对本文件进行了修改，可以重新读入。

本文件左侧的“编码”是已经实现指令的 8 位操作码 IRH，采用二进制表示；中间的“入口”是该操作码对应的微程序的入口地址，采用十六进制表示。例如图中选中的“0000 0001 05”表示操作码“0000 0001”的微程序入口地址是 05H。分号后面是注释。

如果用微程序方式设计了新指令，需要在本文中添新指令与微程序入口地址的映射关系，否则当执行到新指令时，无法找到对应微程序的入口。

7.1.1.3 微程序流程图



微程序流程图

7.1.1.4 微程序存储文件

微程序存储文件是“MicroIns.txt”。

MicroIns.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

地址	CI3~0	SCC3~0	DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A□	B□
00:00	1110	0000	000	111	100	011	001	001	000	001	0101	0101
01:00	1110	0000	000	011	100	010	000	011	000	001	0101	0101
02:00	1110	0000	000	001	001	001	000	000	000	000	0000	0000
03:00	0010	0000	000	000	100	001	000	000	000	000	0000	0000
04:30	0011	0000	000	000	100	011	000	001	001	000	1000	1000
05:30	0011	0000	000	000	100	011	001	001	001	001	1000	1000

软件运行时会自动读取本文件，如果对本文件进行了修改，可以重新读入。

图中第一列是本条微指令在“微程序存储器”的位置，叫“微址”，十六进制，范围是 00~FF。冒号“:”后面是本条微指令内容。


图中选中行是实现指令“ADD”的微指令。冒号后的第一个字段“30”是当前微指令的下址字段，简称“下址”，是十六进制。“下址”后面的所有数字都是二进制。“下址”、CI3~0 和 SCC3~0 用于确定“下一步微指令”的存储单元；DC1、DC2、MRW、I8~6、I5~3、I2~0、SST、SSHSCi 是本条微指令执行时送到各个部件的控制信号；A 口、B 口以及指令寄存器低 8 位 IRL 综合确定读写寄存器用的“A 地址”和“B 地址”。

如果用微程序方式设计了新指令，需要在本文中添加工实现新指令的各条微指令，并重新读入到模拟系统。

7.1.1.5 微程序控制信息

当前微址 04 下一步 30

CI3~0



00->04

01->05

02->06


06->07

04->08


03->09

05->0A

SCC



无条件转下址



微址	下址	CI3~0	SCC3~0	DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A□	B□
00:00	1110	0000	000	111	100	011	001	001	000	001	0101	0101	
01:00	1110	0000	000	011	100	010	000	011	000	001	0101	0101	
02:00	1110	0000	000	001	001	001	000	000	000	000	0000	0000	
03:00	0010	0000	000	000	100	001	000	000	000	000	0000	0000	
04:30	0011	0000	000	000	100	011	000	001	001	000	1000	1000	
05:30	0011	0000	000	000	100	011	001	001	001	001	1000	1000	
06:30	0011	0000	000	000	100	011	100	001	001	000	1000	1000	
07:30	0011	0000	000	000	100	011	011	001	001	000	1000	1000	
08:30	0011	0000	000	000	100	011	110	001	001	000	1000	1000	
09:30	0011	0000	000	000	100	001	001	001	001	001	1000	1000	

微程序控制信息

图中界面是微程序控制器状态：“当前微址”显示当前执行的微指令存储地址；“下一步”是将要执行的下一条微指令地址；CI 和 SCC 指示灯显示如何确定“下一步”地址；中间框显示指令码到微程序入口的 MAP 映射，内容来自文件“InsMap.txt”，蓝色选中行表示指令码 00H 映射到微程序入口地址 04H；右边大框内容来自微程序存储文件“MicroIns.txt”，蓝色选中行表示当前正执行微地址是 04H 的微指令。

7.1.2 实验目的

理解系统中几条典型指令（例如，ADD、SHR、OUT、MVRD、JRC、RET、CALA 等指令）的功能、格式和执行流程。

- 1、深入理解微程序控制器的功能、组成知识；
- 2、深入地学习各类典型指令的执行流程；

- 3、对指令格式、寻址方式、指令系统、指令分类等建立具体的概念;
- 4、学习微程序控制器的工作过程和相关技术。


7.1.3 实验内容

- 1、微程序方式运行监控程序
- 2、微程序方式运行手置指令

7.1.4 实验步骤

7.1.4.1 微程序方式运行监控程序



- 1、运行程序 “ZCHPC1.exe”
- 2、将左下方的控制开关置为 **1001** (单步、内存读指、微程序、联机);
- 3、按 **【RESET】** 键, 进行初始化。
- 4、重复按 **【START】** 键, 观察微程序信息窗口、微信号、IR 值、程序窗口变化。


本次实验观察的是操作系统的运行过程。能观察到微地址  的变化依次是 “00” -> “01” -> “02” -> “03” ……。00、01、02、03、30 微指令不属于任何指令, 是公共操作, 完成 “取指” “映射” 等功能。


第一次按 **【START】**, 微地址是 “00”。微指令的 $CI3\sim0=1110$, 强制顺序执行, 下一步微地址必然是 “01”。该本步完成 “0->PC” 功能, 同时 “禁止中断”。

再按 **【START】**, 微地址是 “01”, 微指令的 $CI3\sim0=1110$, 下一步微地址必然是 “02”。本步功能是完成 “PC→AR、PC+1→PC”, 地址寄存器 AR=0, 为读取第一条指令做准备。

在初始两步 “00” 和 “01” 时, 只是微程序信息窗口变化, 程序窗口是没有选中的,

如图 , IR 的初始值 IR 。

再按 **【START】**, 微地址是 “02”。微指令的 $CI3\sim0=1110$, 下一步微地址必然是 “03”。本步功能是 “MEM→IR”, 从内存读取数据送到指令寄存器 IR。现象是 IR 显示 IR ,

程序窗口选中第 1 行 。本指令 “8800 004E” 总长 2 字, 存储在内存的 “0000” 和 “0001” 单元, 现在只读取了第 1 字 “8800”, 在指令具体实现时完成读取第 2 字 “004E”。本指令的功能是 “MVRD R0, 004E”。下面就是具体实现指令 IR=8800 的功能。

再按 **【START】**, 微地址是 “03”。微指令的 $CI3\sim0=0010$, 功能是 IR 到微地址的 MAP 映射。如图



可观察到“88->1D”，指令码“88”的微程序入口地址“1D”，下一步微地址是“1D”。本步只完成“映射”功能，即该微指令的后半部分没有实际功能。

再按【**START**】，微程序窗口运行到“1DH”

```
1C:30 0011 0000 000 000 001 011 000 111 000 000 0000 1000
1D:1C 0011 0000 000 011 100 010 000 011 000 001 0101 0101
```

“1D”微指令的“下址”=1C，CI3~0=0011，SCC3~0=0000，根据查“表 7-2-2”可知“无条件转下址”，则“下一步”微地址必然是“1C”。本步实现了“PC→AR、PC+1→PC”，为读取指令第2字“004E”做准备。

再按【**START**】，微程序信息窗口运行到“1CH”

```
1C:30 0011 0000 000 000 001 011 000 111 000 000 0000 1000
1D:1C 0011 0000 000 011 100 010 000 011 000 001 0101 0101
```

“1C”微指令的“下址”=30，CI3~0=0011，SCC3~0=0000，根据查“表 7-2-2”可知“无条件转下址”，则“下一步”微地址必然是“30”。本步实现了“MEM→DR”，即读取内存数据“004E”（指令第2字）送DR，由 Sb=1 与 IRL=00H 确定 B 地址=0000B，即 R0 就是 DR，则“004E”送 R0。指令 IR=8800 的功能完成。下面进入公共操作。

再按【**START**】，微程序窗口运行到“30H”，进行中断检测。

```
30:3A 0011 0010 011 000 100 000 000 111 000 000 0000 0000
31:02 0011 0000 000 011 100 010 000 011 000 001 0101 0101
```

“30”是所有指令的微程序结束后的“最终归宿”，属于公共操作，功能是执行“中断检测”，根据结果要么跳转至“3A”要么顺序执行“31”。在微地址“00”已经设置“禁止中断”，所以必然是“顺序执行”。

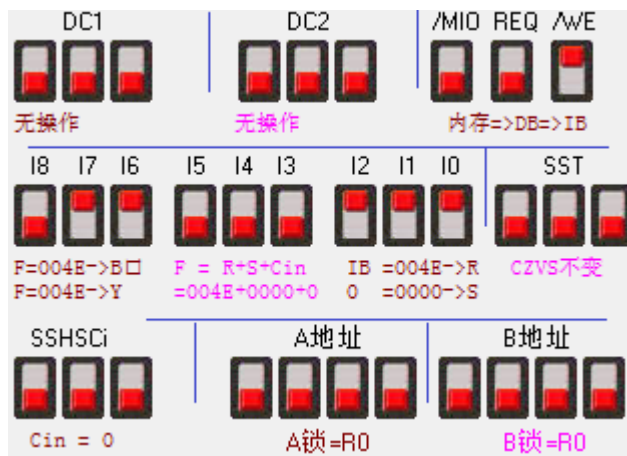
再按【**START**】，微程序信息窗口运行到“31H”，也属于公共操作，功能“PC->AR，PC+1->PC”，为取“下一条指令”做准备。

“31”微指令的“下址”=02，CI3~0=0011，SCC3~0=0000，“无条件转下址”，下一步微指令“02”。

再按【**START**】，微程序信息窗口运行到“02H”，读取新指令到 IR 寄存器，开始下一轮指令功能实现……。

主界面菜单“窗口”->“显示动画”，打开动画窗口，可观察动画数据流动。需要注意的是，每次按下【**START**】动画运行，要等到动画结束灯亮（绿灯），才能再按【**START**】。

在“单步+微程序+联机”状态运行过程中，主界面的“微码控制开关”是无法拨动的，但依然有变化，显示的是当前微程序产生的控制信息，如图：

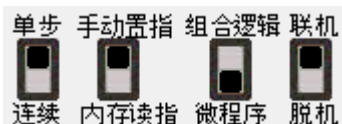


微程序控制信号

本图是指令“MVRD R0, 004E”的第二步的状态，每个部件得到的控制信息都用开关显示，开关下面的注释有助于理解部件功能。需要注意的是：最后显示的是实际起作用的“A地址”、“B地址”而不是“A口”、“B口”。

7.1.4.2 微程序方式运行手置指令

- 1、运行程序“ZCHPC1.exe”
- 2、将左下方的控制开关置为 1101（单步、手动置指、微程序、联机）：



- 3、拨动蓝色的 16 位数据开关，设置指令操作码。
- 4、按【RESET】键，进行初始化。
- 5、按【START】键，观察微程序信息窗口、微信号、IR 值、程序窗口变化。也可以观察动画数据流动。
- 6、再次按【START】键……

注意：这里步骤“3”设置的指令操作码，必须是微程序已经实现的指令，如基本指令。

一、指令“ADD R0, R3”，A 组，1 步完成。

1.1 16 位数据开关 0000 0000 0000 0011，表示指令“ADD R0, R3”。

1.2 菜单“设置”->“寄存器”，设置 R0=5555H, R3=AAAAH。ADD 指令要操作的数。

1.3 按【RESET】键，系统初始化。

1.4 按【START】键，显示当前微址 00，进行“0->PC”操作。

1.5 按【START】键，当前微址 01，进行“PC->AR, PC+1->PC”操作。

1.6 按【START】键，当前微址 02， “MEM->IR”，IR 有数值 0003。

1.7 按【START】键，当前微址 03， “/MAP”操作，指令到微程序入口，此时 MAP 窗口信息 00->04
01->05，IRH = “00”映射到微程序地址“04”，则下一步微地址必然是 04。

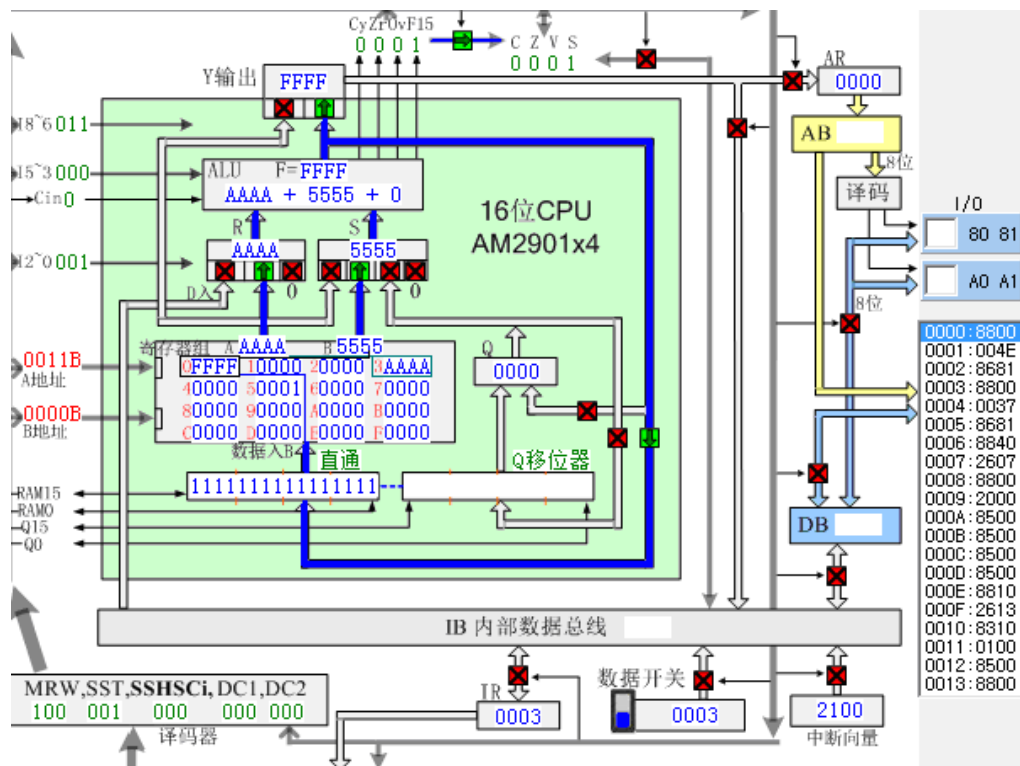
1.8 按【START】键，当前微址 04，1步完成 ADD 功能。结果R0 FFFF。

1.9 按【START】键，当前微址 30，进行中断判断。

1.10 按【START】键，当前微址 31，进行“PC->AR, PC+1->PC”操作，同 1.5

1.11 按【START】键，当前微址 02，回到了 1.6……

主界面菜单“窗口”->“显示动画”，打开动画窗口，可观察动画数据流动。需要注意的是，每次按下【START】动画运行，要等到动画结束灯亮（绿灯），才能再按【START】。



R0+R3->R0

二、指令“PUSH R0”，B组，2步完成。

2.1 16位数据开关 1000 0101 0000 0000，表示指令“PUSH R0”

2.2 菜单“设置”->“寄存器”，设置 R0=5555H，R4(SP)=2607H。

2.3 按【RESET】键，系统初始化

2.4 按【START】键，显示当前微址 00，进行“0->PC”操作

2.5 按【START】键，当前微址 01，进行“PC->AR，PC+1->PC”操作

2.6 按【START】键，当前微址 02，“MEM->IR”，IR有数值IR 8500

2.7 按【START】键，当前微址 03，“/MAP”操作，指令到微程序入口，此时 MAP

窗口信息 85->15，IRH = “85”映射到微程序地址“15”，则下一步微地址必然是15。

2.8 按【START】键，当前微址 15，PUSH操作第1步。

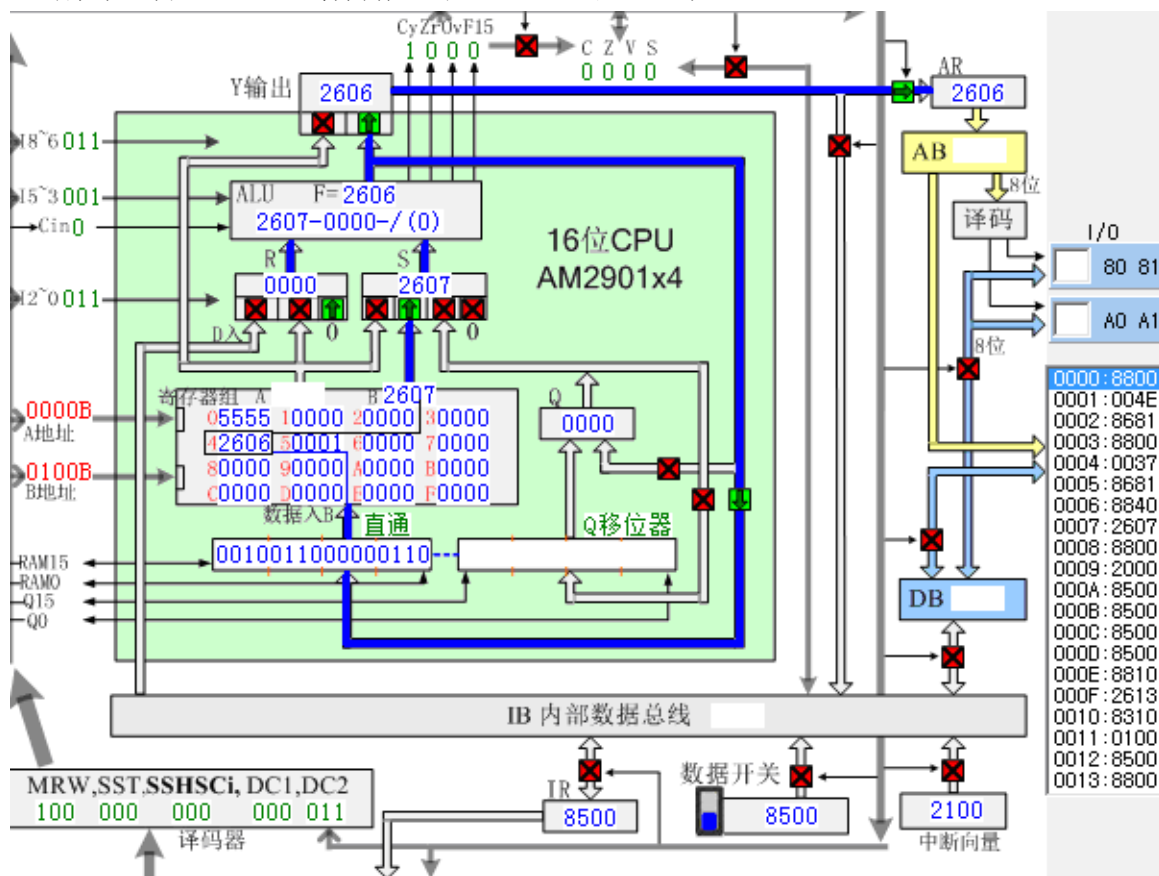
2.9 按【START】键，当前微址 1A，PUSH 操作第 2 步。PUSH 操作结束。

2.10 按【START】键，当前微址 30，进行中断判断。

2.11 按【START】键，当前微址 31，进行“PC->AR, PC+1->PC”操作，同 2.5

2.12 按【START】键，当前微址 02，回到了 2.6……

此时菜单“窗口”->“查看内存”，在地址 2606 处，显示“5555”。



PUSH 操作第 1 步，AR 寄存器设置堆栈指针

④ JRC、JRZ 等操作，一定要在按【RESET】键后再设置 C、Z 标志位寄存器数值。如果提前设置了，按下【RESET】键会将 C、Z、V、S 清零。

模拟软件中 R5 寄存器是 PC，R4 寄存器是堆栈指针 SP。

A 组基本指令举例

指令	16 位数据开关	说明
ADD	0000 0000 0000 0001	R0+R1，结果存入 R0
SUB	0000 0001 0000 0001	R0-R1，结果存入 R0
AND	0000 0010 0000 0001	R0,R1 与运算，结果存入 R0
CMP	0000 0011 0000 0001	R0-R1，不保存结果，只影响 CZVS 标志
XOR	0000 0100 0000 0001	R0,R1 异或运算，结果存入 R0
TEST	0000 0101 0000 0001	R0,R1 与运算，不保存结果，只影响 CZVS 标志
OR	0000 0110 0000 0001	R0,R1 或运算，结果存入 R0
MVRR	0000 0111 0000 0001	R0 数值给 R1
DEC	0000 1000 0000 ****	R0-1，结果存回 R0
INC	0000 1001 0000 ****	R0+1，结果存回 R0
SHL	0000 1010 0000 ****	R0 值左移，结果存回 R0
SHR	0000 1011 0000 ****	R0 值右移，结果存回 R0
JR	0100 0001 0000 1000	无条件跳转，PC 值+00001000B->PC
JRC	0100 0100 0000 1000	C=1 时跳转，PC 值+00001000B->PC
JRNC	0100 0101 0000 1000	C=0 时跳转，PC 值+00001000B->PC
JRZ	0100 0110 0000 1000	Z=1 时跳转，PC 值+00001000B->PC
JRNZ	0100 0111 0000 1000	Z=0 时跳转，PC 值+00001000B->PC

注明：“*”表示无关，可以是“0”也可以是“1”

B 组基本指令举例

指令	16 位数据开关	说明
JMPA	1000 0000 **** **	无条件跳转，内存地址（PC+1）的内容给 PC
LDRR	1000 0001 0000 0001	内存地址是 R1 值的内存内容存入 R0
IN	1000 0010 1000 0001	从端口 81H 读数值，给 R0
STRR	1000 0011 0000 0001	R1 的值，给内存地址是 R0 值的内存单元
PSHF	1000 0100 **** **	CZVS 标志入栈，存到内存地址（R4-1）单元
PUSH	1000 0101 **** 0001	R1 值入栈，R1 值存到内存地址（R4-1）单元
OUT	1000 0110 1000 0000	将 R0 的值给端口 80H，提前设置 R0=0037H
POP	1000 0111 0000 ****	出栈。内存地址（R4）的内容存入 R0
MVRD	1000 1000 0000 ****	立即数给 R0，内存地址（PC+1）的内容给 R0

POPF	1000 1100 **** *	出栈，结果存入 CZVS
RET	1000 1111 **** *	出栈。内存地址（R4）的内容存入 R5，就是 PC

7.1.5 实验报告要求

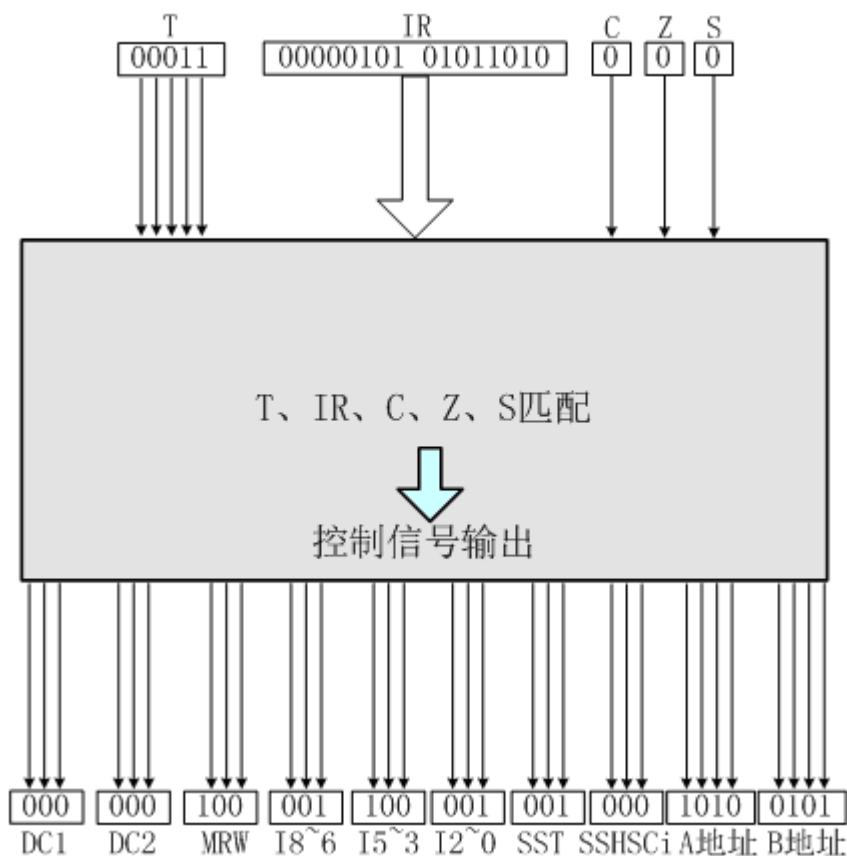
按照教材中的方法，将基本指令和扩展指令用微程序方式运行一遍。观察主界面及动画界面的相关数据并记录。（通过动画界面，能看到数据的流动以及变化，可以显示各个指令的操作细节。）

第八章 组合逻辑控制器实验

8.1 模拟软件的组合逻辑控制器实验

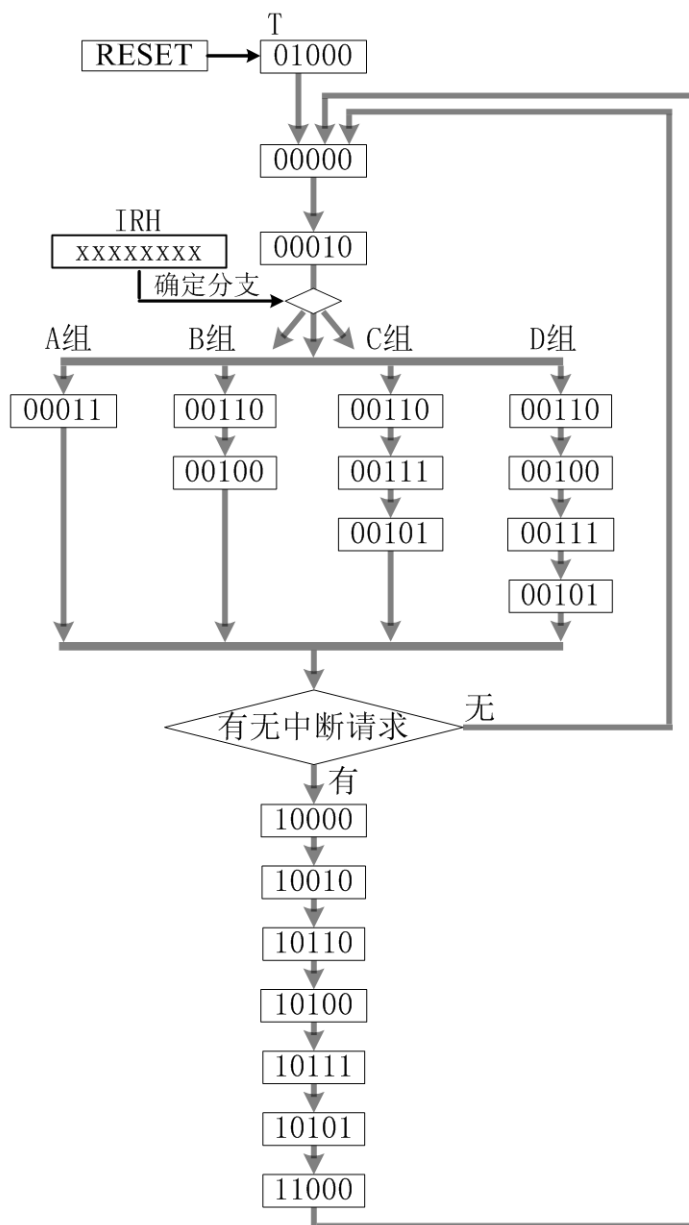
8.1.1 组合逻辑控制器工作原理

模拟软件的组合逻辑，以节拍 T （5 位）、 IR 、 C 、 Z 、 S 作为输入，各部件的控制信号作为输出。其中 T 是必须的，而 IR 和 C 、 Z 、 S 在某些情况下不需要。



组合逻辑控制器的工作流程

节拍 T 有自己的跳变规则，见下图：



节拍 T 跳变规则

图中，按【RESET】键后，会强制设置 $T=01000$ ； T 下一跳必然是“00000”；再下一跳必然是“00010”。再下一跳， T 有 4 个选择分别是：A 组指令、B 组指令、C 组指令、D 组指令，由当前 IR 的值确定，此时 IR 是将要执行的指令。因此前 3 个 T 要完成“取指令”操作。

- ① $T=01000$ 时，实现 $PC=0$ ，同时禁止中断。
- ② $T=00000$ 时，实现 $PC \rightarrow AR$ ， $PC+1 \rightarrow PC$ 。

③ $T=00010$ 时，内存数据送 IR 。 IR 即为要执行的指令，并据此确定 A、B、C、D 分支路线，即下一个 T 的值。

不同的指令，实现不同的功能，需要的步数不同，有 1 步就能完成的，有 2 步能完成的，也有 3 步 4 步完成的，分别是 A、B、C、D 组指令（超过 4 步才能完成的指令，在这里不能用组合逻辑方式，需用微程序方式）。A 组指令经过一个 $T=00011$ 即可执行完；B 组指令需要 2 步， $T=00110$ 、 00100 ；C 组 D 组分别是 3 步和 4 步完成。A、B、C、D 分组由 IR 值确定，规则如下：

- (1) IR_{15} 、 IR_{14} 为 00 和 01 都是 A 组；10 是 B 组；11 是 C、D 组。
- (2) IR_{15} 、 IR_{14} 、 IR_{11} 为 110 是 C 组；111 是 D 组。
- (3) IR_{13} 用于供使用者区分基本指令和扩展指令： $IR_{13}=0$ 是基本指令， $IR_{13}=1$ 是扩展指令。
- (4) $IR_{11}\sim IR_8$ 用于区分同一指令组中的不同指令。

分组规则表

	基本指令， IR_H 值	扩展指令， IR_H 值
A 组指令	0*0* ****	0*1* ****
B 组指令	100* ****	101* ****
C 组指令	110* 0***	111* 0***
D 组指令	110* 1***	111* 1***

注：“*”表示可以是“0”或“1”。

如果设计实现新的扩展指令时只使用微程序控制器方式而不用组合逻辑控制器方式，指令编码可完全不遵循这里的 A、B、C、D 分组规则。

A、B、C、D 组结束时检测是否有中断请求，如果没有，则回到“ $T=00000$: $PC \rightarrow AR$, $PC+1 \rightarrow PC$ ”和“ $T=00010$, 读内存（下一条将要执行的指令）到 IR ”，然后执行指令，循环往复。如果 A、B、C、D 组完成时检测到有中断请求，则 T 会依次变化为 $10000 \rightarrow 10010 \rightarrow 10110 \rightarrow 10100 \rightarrow 10111 \rightarrow 10101 \rightarrow 11000$ ，并再次回到“ $T=00000$ ”读取下一条指令……。

当 $T=01000$ 、 00000 、 00010 以及 10000 、 10010 、 10110 、 10100 、 10111 、 10101 、 11000 ，即处于“A、B、C、D”组以外时（参照“图 节拍 T 的规则”），不需要 IR 及 C、Z、S 即可生成控制信号，此时的 T 不特定为某指令服务（即与 IR 寄存器无关），处于这些 T 的功能可以称之为“公共指令”，每个“公共 T ”的功能在设计实验系统时基本就确定了，数量也不再变，根据需要一共设计了 10 条，各个 T 的前后顺序也已经固定，不可更改。

当 T 处于“A、B、C、D”组内时，根据组别不同， T 严格按照各自的路线变化。组别是 IR 确定的，各个组内部每个 T 也必须与 IR 结合，必要时以 C、Z、S 辅助，才能准确生成控制信号。不同的 IR 与相同的 T 结合，生成的控制信号也不同，非常灵活。这里体现了实现每条指令的细则，占据了组合逻辑存储的大部分空间，还能根据需要进行扩展。如果需要改变原有指令功能或者设计新的扩展指令，主要在这里实现。

模拟软件已经实现的基本指令和扩展指令，编码都符合组合逻辑控制器的格式，分成了4组：

A 组：基本指令 ADD、SUB、AND、OR、XOR、CMP、TEST、MVRR、DEC、INC、SHL、SHR、JR、JRC、JRNZ、JRZ、JRNZ

扩展指令 ADC、SBB、RCL、RCR、ASR、NOT、CLC、STC、EI、DI、JRS、JRNS、JMPR

B 组：基本指令 JMPA、LDRR、STRR、PUSH、POP、PSHF、POPF、MVRD、IN、OUT、RET

C 组：扩展指令 CALR、LDRA、STRA、LDRX、STRX

D 组：基本指令 CALA 扩展指令 IRET

8.1.2 组合逻辑存储文件

组合逻辑存储文件是“HardWiredLogic.txt”。



;输入匹配			输出信号									
;T	IRH	CZS	DC1	DC2	MRW	I86	I53	I20	SST	SSHSCi	A	B
01000	--	---	0	7	4	3	1	1	0	1	5	5
00000	--	---	0	3	4	2	0	3	0	1	5	5
00010	--	---	0	1	1	1	0	0	0	0	0	0
00011	00	---	0	0	4	3	0	1	1	0	R	R
00011	20	---	0	0	4	3	0	1	1	2	R	R
00011	01	---	0	0	4	3	1	1	1	1	R	R

软件运行时会自动读取本文件，如果对本文件进行了修改，可以重新读入。

本文件左三列分别是 T、IRH、CZS 的匹配条件，如果是“--”则表示不需要匹配，T 和 CZS 是二进制，IRH 是十六进制。右边是“输出信号”。

图中选中行是实现指令“ADD”的现象。本行的匹配条件即为“T=00011 并且 IRH=00，CZS 不考虑”。DC1、DC2、MRW、I86、I53、I20、SST、SSHSCi 是选中本行时送到各个部件的控制信号，都是十六进制。最后的“AB”如果是数字，则表示“A 地址”和“B 地址”由该数字确定；如果是字母“R R”，则表示“A 地址”和“B 地址”由指令寄存器低 8 位 IRL 确定。

如果用组合逻辑方式设计了新指令，需要在本文件中添加实现新指令的匹配条件和控制信号，并重新读入到模拟系统。

8.1.3 实验目的与内容

掌握已经设计好并正常运行的几条典型指令（例如，ADD、SHR、JRC、MVRD、OUT、RET、CALA 等指令）的功能、格式和执行流程：

- 1、深入理解控制器的功能、组成知识；
- 2、深入地学习各类典型指令的执行流程；
- 3、对指令格式、寻址方式、指令系统、指令分类等相关知识熟悉掌握；

实验内容：

- 1、理解掌握模拟软件的功能部件组成原理。分析系统中的基本指令（例如 ADD、SHR、JRC）和扩展指令（例如 ADC、JRS、JRNS、LDRA）的功能、格式和执行流程。在模拟软件中、观察指令的运行流程，包括 T 以及控制信号值。
- 2、组合逻辑方式运行监控程序
- 3、组合逻辑方式运行手置指令


8.1.4 实验步骤

8.1.4.1 组合逻辑方式运行监控程序

- 1、运行程序 “ZCHPC1.exe”
- 2、将左下方的控制开关置为 **1011**（单步、内存读指、组合逻辑、联机）；




- 3、按 **【RESET】** 键，进行初始化，同时执行第一步 T=01000。观察控制及各个部件状态值。
- 4、按 **【START】** 键，观察组合逻辑控制器信息窗口的变化。包括微控制信号、T4~0、IR 值、C、Z、S 值、程序窗口。
- 5、再次按 **【START】** 键，观察后续现象。
- 6、再次按 **【START】** ……

本次实验观察的是操作系统的运行过程。能观察到 T 灯  的变化依次是 “01000” -> “00000” -> “00010” ……（每按一下 **【START】**）。

按 **【RESET】** 键即执行了组合逻辑的第一步，此时 T=01000，属于公共指令，不需要 IR 和 CZS 匹配。本步完成 “0->PC” 功能，同时 “禁止中断”。根据 T 的跳变规则，下一步 T=00000。

再按 **【START】**，T=00000，公共指令，本步功能是 “PC→AR、PC+1→PC”，地址寄存器 AR=0，为读取第一条指令做准备。根据 T 的跳变规则，下一步 T=00010。

T 在初始两步时（01000,00000），程序窗口是没有选中的，如图



，IR 是初始值 IR 0000。

再按 **【START】**，T=00010，公共指令，本步功能是 “MEM→IR”，从内存读取数据送到指令寄存器 IR。现象是 IR 显示 IR 8800，程序窗口选中第 1 行

```
0000:8800 004E: MVRD R0, 004E
0002:8681 : OUT 0081
```

。本指令“8800 004E”总长 2 字，存储在内存的“0000”和“0001”单元，现在只读取了第 1 字“8800”，在指令具体实现时完成读取第 2 字“004E”。由 IR=8800 确定当前指令属于“B 组”指令，需要 2 步完成，其 T 的跳变已经确定，依次是 00110、00100。这 2 步不但需要匹配 T，还需要匹配 IR 甚至 CZS。

再按【START】，同时匹配 T=00110 和 IRH=88，现象是

```
00110 88 --- => 0 3 4 2 0 3 0 1 5 5
00100 88 --- => 0 0 1 3 0 7 0 0 0 R
```

。本步实现了“PC→AR、PC+1→PC”，为读取指令第 2 字“004E”做准备。

再按【START】，同时匹配 T=00100 和 IRH=88，现象是

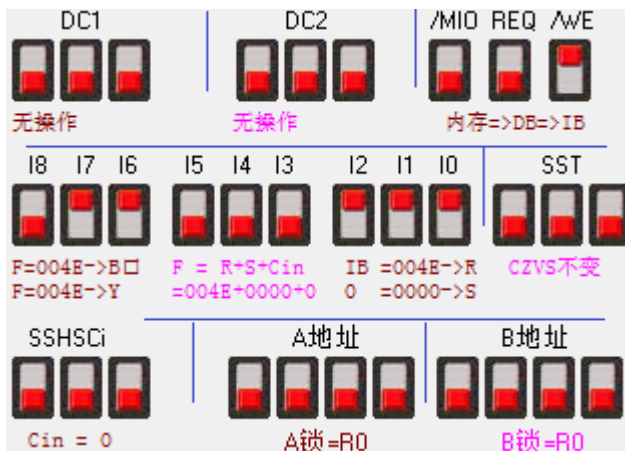
```
00110 88 --- => 0 3 4 2 0 3 0 1 5 5
00100 88 --- => 0 0 1 3 0 7 0 0 0 R
```

。本步实现了“MEM→DR”，即读取内存数据“004E”（指令第 2 字）送 DR，B=R 与 IRL=00H 确定 B 地址=0000B，即 R0 就是 DR，则“004E”送 R0。指令 IR=8800 的功能完成。完成指令的同时进行中断判定，在 T=01000 已经设置“禁止中断”，所以下一步是 T=00000 的公共操作。

再按【START】，T=00000，公共指令，功能是“PC→AR、PC+1→PC”，又回到了循环原点，准备读取下一条指令，开始新一轮循环……。

主界面菜单“窗口”→“显示动画”，打开动画窗口，可观察动画数据流动。需要注意的是，每次按下【START】动画运行，要等到动画结束灯亮（绿灯），才能再按【START】。

在“单步+组合逻辑+联机”状态运行过程中，主界面的“微码控制开关”是无法拨动的，但依然有变化，显示的是当前组合逻辑产生的控制信息，如图：

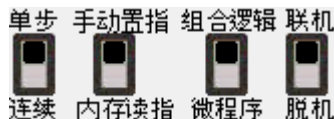


组合逻辑控制信号

本图执行的是指令“MVRD R0, 004E”的第二步的状态。每个部件得到的控制信息都用开关显示，开关下面的注释有助于理解部件功能。最后显示的是实际起作用的“A 地址”、“B 地址”。

8.2.4.2 组合逻辑方式运行手置指令

- 1、运行程序 “ZCHPC1.exe”
- 2、将左下方的控制开关置为 **1111**（单步、手动置指、组合逻辑、联机）；




- 3、拨动蓝色的 16 位数据开关，置指令操作码。
- 4、按一下【RESET】按键，进行初始化，同时执行第一步 T=01000。观察控制及各个部件状态值。
- 5、按【START】键，观察组合逻辑控制器信息窗口的变化。包括微信号、IR 值、T 值、程序窗口等。
- 6、再次按【START】……

这里步骤“3、置指令操作码”，设置的指令操作码，必须是组合逻辑控制器已经实现的指令，如基本指令或扩展指令。

注明：模拟软件中 R5 寄存器就是 PC，R4 寄存器就是堆栈指针 SP。

- 1、指令“ADD R0, R1”，A 组，1 步完成。
 - 1.1 16 位数据开关 0000 0000 0000 0001，表示指令“ADD R0, R1”。
 - 1.2 菜单“设置”->“寄存器”，设置 R0=5555，R1=AAAA。设置 ADD 指令参加操作的数。

1.3 按一下【RESET】按键。T 灯显示 ，进行“0->PC”操作

1.4 按【START】，T 灯显示 ，进行“PC->AR, PC+1->PC”操作

1.5 按【START】，T 灯显示 ，IR 有数值 IR 0001

1.6 按【START】，T 灯显示 ，1 步完成 ADD 功能。

1.7 按【START】，T 灯显示 ，回到了 1.4……

此时 R0 数值即为计算结果。





主界面菜单“窗口”->“显示动画”，打开动画窗口，可观察动画数据流动。需要注意的是，每次按下【START】动画运行，要等到动画结束灯亮（绿灯），才能再按【START】。

2、指令“PUSH R0”，B 组，2 步完成。

2.1 16 位数据开关 ，表示指令“PUSH R0”。

2.2 菜单“设置”→“寄存器”，设置 R0=5555，R4(SP)=2607H。

2.3 按一下【RESET】按键。T 灯显示 ，进行“0->PC”操作

2.4 按【START】，T 灯显示 ，进行“PC->AR, PC+1->PC”操作

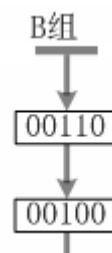
2.5 按【START】，T 灯显示 ，IR 有数值 IR

2.6 按【START】，T 灯显示 ，PUSH 操作第 1 步。

2.7 按【START】，T 灯显示 ，PUSH 操作第 2 步。

2.8 按【START】，T 灯显示 ，指令完成后，回到了 2.4……

此时菜单“窗口”→“查看内存”，在地址 2606 处，显示“5555”。





3、指令“CALA ADR”，D 组，4 步完成。


3.1 16 位数据开关 ，表示指令“CALA”。

3.2 菜单“设置”→“修改内存”，设置 0001H 地址内容 5555。

菜单“设置”→“寄存器”，设置 R4(SP)=2607。

3.3 按一下【RESET】按键。T 灯显示 ，进行“0->PC”操作

3.4 按【START】，T 灯显示 ，进行“PC->AR, PC+1->PC”操作

3.5 按【START】，T 灯显示 ，IR 有数值 IR

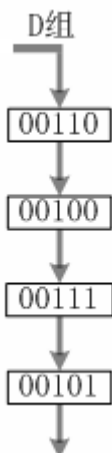
3.6 按【START】，T 灯显示 ，CALA 操作第 1 步。

3.7 按【START】，T 灯显示 ，CALA 操作第 2 步。

3.8 按【START】，T 灯显示 ，CALA 操作第 3 步。

3.9 按【START】，T 灯显示 ，CALA 操作第 4 步。

3.10 按【START】，T 灯显示 ，回到了 3.4……



此时 AR 的值变为子程序的地址 5555, CALA 的下一条指令的地址值送到了内存 2606 单元。查看内存时, 注意要点“刷新”。

4、按照上述方法, 将基本指令和扩展指令用组合逻辑控制器方式运行一遍。实验前置条件参看第七章。

8.1.5 实验报告要求

按照教材中的方法, 将基本指令和扩展指令用组合逻辑方式运行一遍。观察主界面及动画界面的相关数据并记录。(通过动画界面, 能看到数据的流动以及变化, 可以显示各个指令的操作细节。)

第九章 中断实验

9.1 模拟软件的中断实验

9.1.1 中断相关知识

模拟软件的中断信息在主界面右下方。包括 INTE、P1、P0、中断向量值、中断源按键。



中断信息

◆ INTE：中断允许标志。

INTE=1（灯亮）表示“允许中断”，由 DC2=110 设置“允许”。

INTE=0（灯灭）表示“禁止中断”，由 DC2=111 设置“禁止”。

◆ P1P0：记录当前中断等级，其编码含义如表。

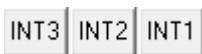
P1P0	含义
00	当前没有中断请求或没有处理中断服务
01	正在处理一级中断请求
10	正在处理二级中断请求
11	正在处理三级中断请求

◆ 中断向量：

图“中断信息”中的 210C、2108、2104 即为 INT3、INT2、INT1 默认中断向量。

可通过主界面的“设置”->“寄存器”设置“中断向量初始”，保存后三个中断向量分别是刚设置的“中断向量初始”加 4、加 8、加 C。

◆ 三个中断源按键：



本系统支持三级中断，这三个按键作为中断源，从左到右依次为三级、二级、一级中断，对应的中断等级编码 P1P0 分别为 11B、10B、01B，依次降低。

◆ /INT：新中断请求标志位，上图没有显示本信号。

标志位/INT 初始为“1”。系统据此可判断“没有中断请求”。

当 INTE=0（禁止中断）时，或者没有新中断到来时，或者新中断等级不如当前正处

理的中断等级高时，/INT 保持为“1”。

当 INTE=1（允许中断）时，如果到来的新中断等级比当前高，则标记/INT=0，系统检测到会进行中断处理。系统进行中断处理时会将/INT 恢复为“1”。

◆ 有关中断的操作见表：

DC2 编码表

DC2 编码	功能说明
000	无操作
001	IB=>IR
010	未使用，待扩展
011	Y 输出=>AR
100	恢复中断等级，IB=>P1P0
101	中断新等级（01、10 或 11）=> P1P0
110	允许中断，INTE=1
111	禁止中断，INTE=0

按“RESET”键复位时，会强行设置 P1=P0=0。

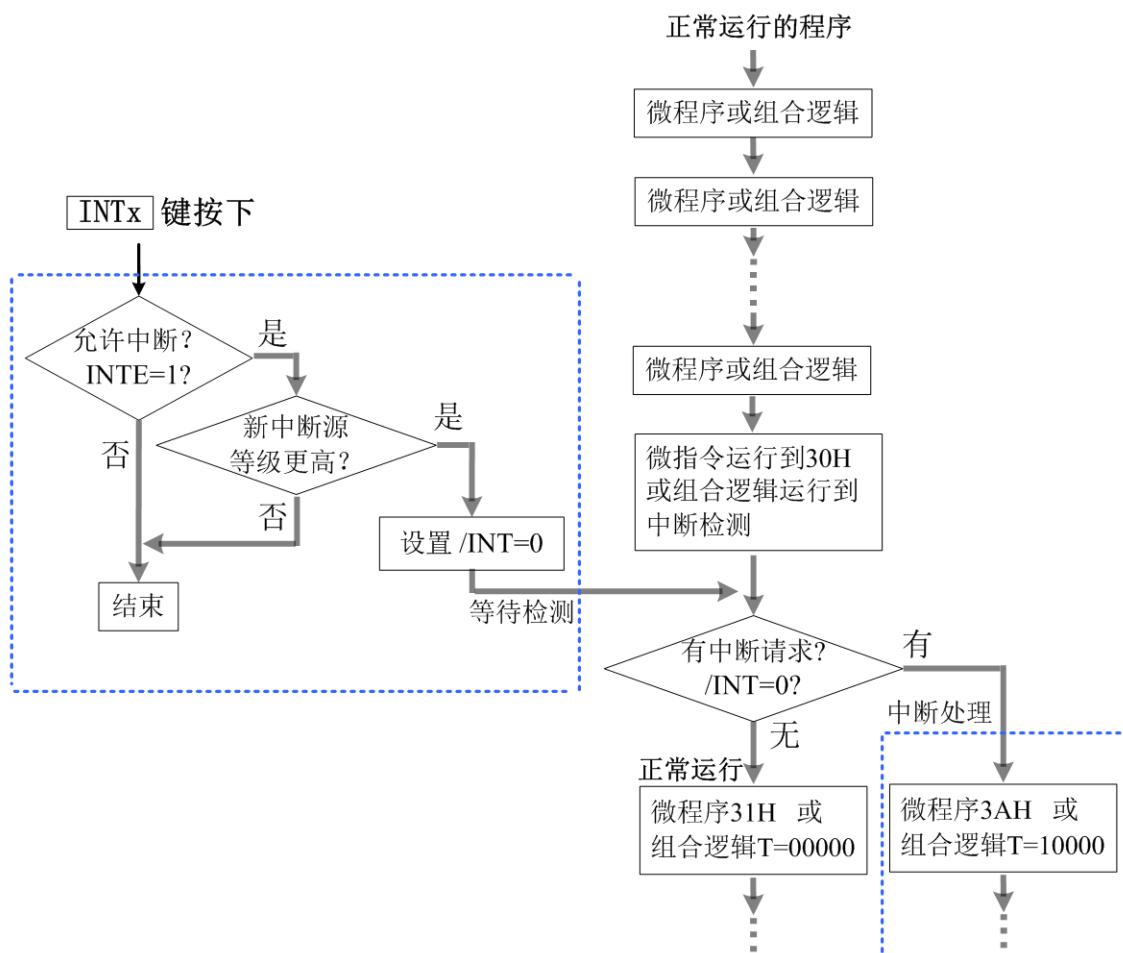
DC2=100 时，P1=IB11、P0=IB10，用作恢复旧的中断等级。

DC2=101 时，P1、P0 被置为新的中断等级。

DC1=011 时，P1、P0 随状态标志（C、Z、V、S）送至 IB，可以压入栈。

◆ 中断响应流程图

中断响应实现的流程图如下：



中断响应流程图

9.1.2 实验目的

学习和掌握中断的产生、响应和处理等技术。

9.1.3 实验内容

- 1、不通过监控直接运行程序，按中断源按键，观察现象。
- 2、通过监控运行程序，按中断源按键，观察现象。

9.1.4 实验步骤

9.1.4.1 不通过监控运行程序

- 1、编写程序 int1.asm。

```
ORG 2000H
EI
MVRD R0,0036H
CALA 2200H
MVRD R0,4000H
DEC R0
JRNZ 2007H
JR 2001H
RET
```

```
ORG 2104H
JR 2120H
```

```
ORG 2108H
JR 2130H
```

```
ORG 210CH
JR 2140H
```

```
ORG 2120H
PUSH R0
PUSH R3
MVRD R3,31H
JR 2150H
```

```
ORG 2130H
PUSH R0
PUSH R3
MVRD R3,32H
JR 2150H
```

```
ORG 2140H
PUSH R0
PUSH R3
MVRD R3,33H
JR 2150H
```

```
ORG 2150H
EI
MVRD R0,0042H
CALA 2200H
MVRD R0,0049H
CALA 2200H
MVRR R0,R3
CALA 2200H
```

```
IN 81H
SHR R0
SHR R0
JRNZ 215CH
IN 80H
```

```
MVRD R0,0045H
CALA 2200H
MVRD R0,0049H
CALA 2200H
MVRR R0,R3
CALA 2200H
POP R3
POP R0
IRET
```

```
ORG 2200H
PUSH R0
IN 81H
SHR R0
JRNZ 2201H
POP R0
OUT 80H
RET
```

```
END
```

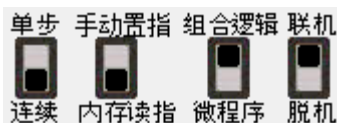
注意本程序中，在地址 2000H 处已用语句“EI”设置了“允许中断”，在地址 2150H 处再次设置，是因为进入中断服务时，“中断隐指令”首先会设置“禁止中断”，如果不修改状态，未来无法响应新中断。

2、运行程序 “ZCHPC1.exe”

3、主界面菜单“设置”->“PC 初始值”，确保窗口内选择如下图



4、置控制开关为 0011（连续、内存读指、组合逻辑、联机）






5、置速度条到最高

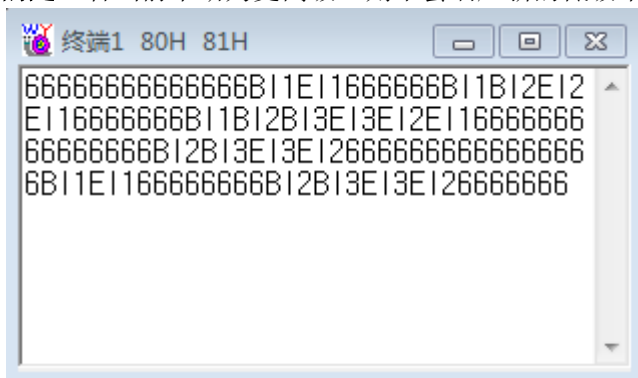


6、主界面菜单“文件”->“打开 ASM 源文件”，打开“int1.asm”，点“编译”，再点“导入内存”。

7、按【RESET】键，进行初始化。

8、按【START】键，终端 1 窗口显示“66666……”

9、按 3 个中断源按钮   ，将转向执行本级中断服务程序，显示字符“BI”和对应的中断等级“1”、“2”或“3”，等待从键盘输入一个字符。在接收键盘一个字符后，显示字符“EI”和等级，退出当前中断服务程序，恢复中断现场，继续执行断点处的程序。若在此期间，又有更高一级的中断请求，则转向执行该级的中断服务程序。**需要注意的是**，若当前中断为更高级，则不会响应新的低级中断请求。



【思考题】:

观察上图现象，需要是进行一系列什么样的操作才能与图中一样，请叙述。




如果将速度条调至低于 20，



则可以在主界面观察当前中断信息，如图



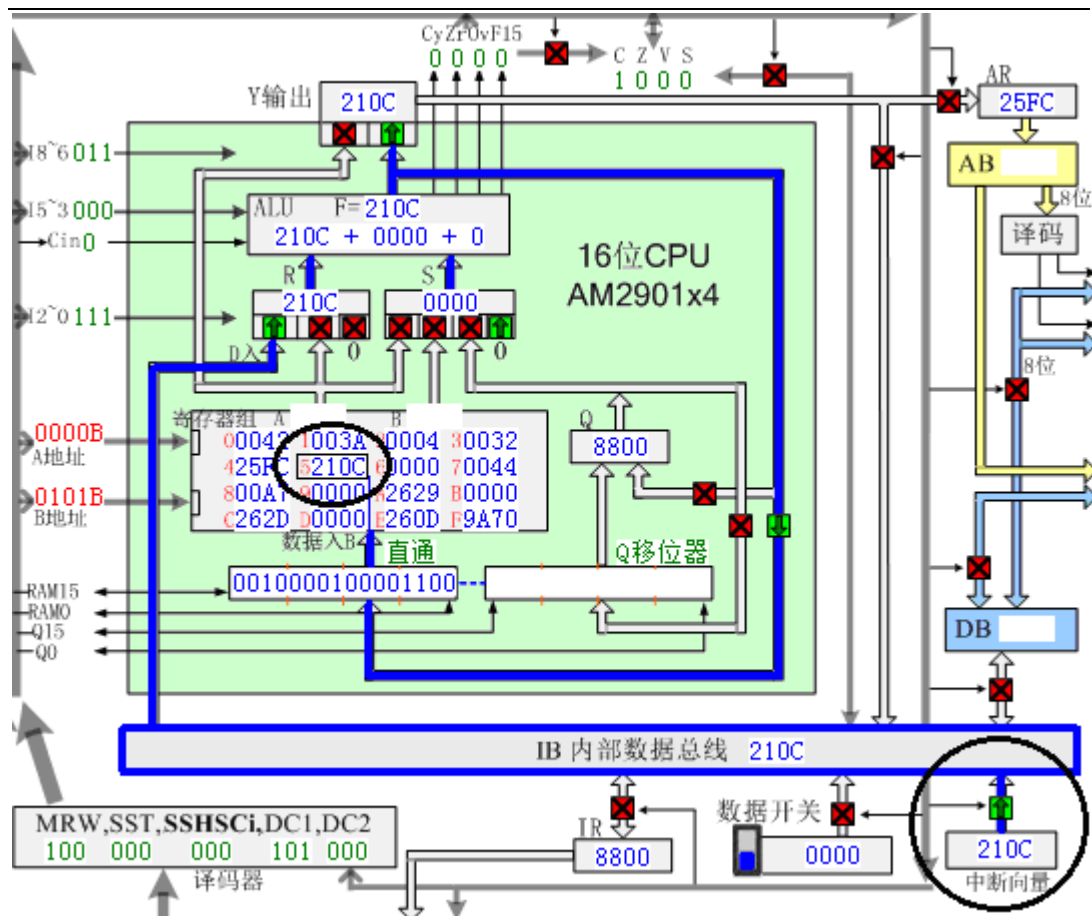
中断级别 P1P0=10 主界面现象
表示当前“允许中断”，正进行“2级中断”服务程序。

如果低速时打开“动画窗口”，按下    之中其一按键，则可以动画显示中断响应过程。

例如：按下【INT3】，则主界面信息如下表示进行3级中断。



中断级别 P1P0=11 主界面现象
动画界面的响应现象如下：



T=11000, 中断向量送 PC

INT3 的中断向量是 210CH, 响应中断时, 先保存当前状态, 再将 210CH 送到 PC, 执行中断服务程序。

9.1.4.2 通过监控运行程序

1、编写程序 int2.asm。

```

第 1 列
ORG 2000h
EI
RET

ORG 2104H
JR 2120H

ORG 2108H
JR 2130H

```

```

第 2 列
ORG 2150H
EI
MVRD R0,0042H
CALA 2200H
MVRD R0,0049H
CALA 2200H
MVRD R0,R3
CALA 2200H

```

```

ORG 210CH
JR 2140H

ORG 2120H
PUSH R0
PUSH R3
MVRD R3,31H
JR 2150H

ORG 2130H
PUSH R0
PUSH R3
MVRD R3,32H
JR 2150H

ORG 2140H
PUSH R0
PUSH R3
MVRD R3,33H
JR 2150H

```

```

PUSH R0
PUSH R2
MVRR R0,R3
MVRD R2,40H
loop1: DEC R2
CALA 2200H
JRNZ loop1
POP R2
POP R0

MVRD R0,0045H
CALA 2200H
MVRD R0,0049H
CALA 2200H
MVRR R0,R3
CALA 2200H
POP R3
POP R0
IRET

ORG 2200H
PUSH R0
IN A1H
SHR R0
JRNC 2201H
POP R0
OUT A0H
RET

END

```

本程序中，中断服务程序在终端 2 窗口显示信息，不影响终端 1 窗口。

2、运行程序“ZCHPC1.exe”

3、主界面菜单“设置”->“PC 初始值”，确保窗口内选择如下图



4、置控制开关为 0011（连续、内存读指、组合逻辑、联机）

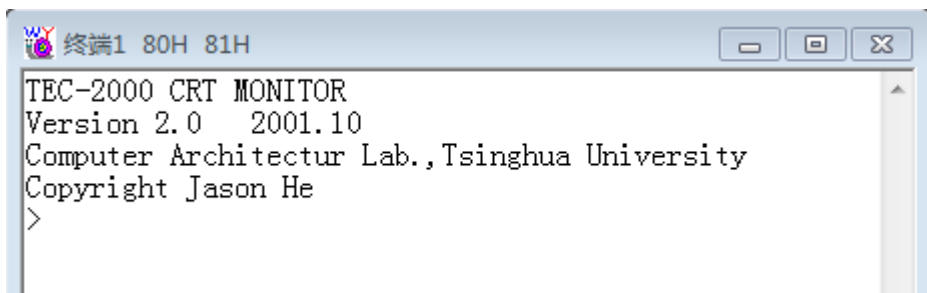
5、速度条默认即可



6、主界面菜单“文件”->“打开 ASM 源文件”，打开“int2.asm”，按“编译”，再按“导入内存”。

7、按【RESET】键，进行初始化。

8、按【START】键，终端 1 窗口显示监控信息

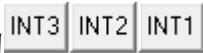


9、在“终端 1”窗口键入

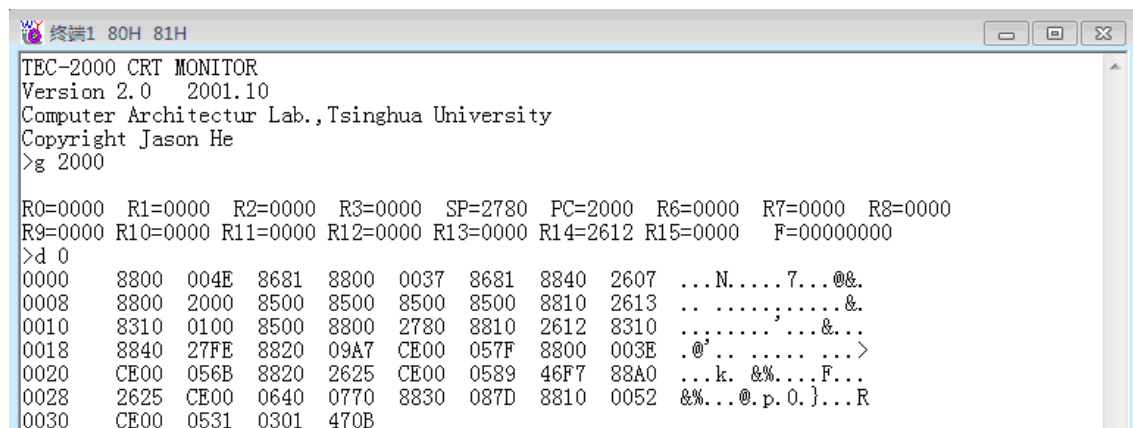
>G 2000✓

该程序只是设置系统允许中断，完成后回到监控。

10、通过菜单“窗口”->“显示终端 2”，打开终端 2 窗口。

11、此时在终端 1 窗口可以进行正常操作，任何时刻按 3 个中断源按钮  中的任意一个，将转向执行本级中断服务程序，在终端 2 窗口显示字符“BI”和对应的中断等级“1”、“2”或“3”，数字显示 40H 次后本中断服务结束，自行返回。

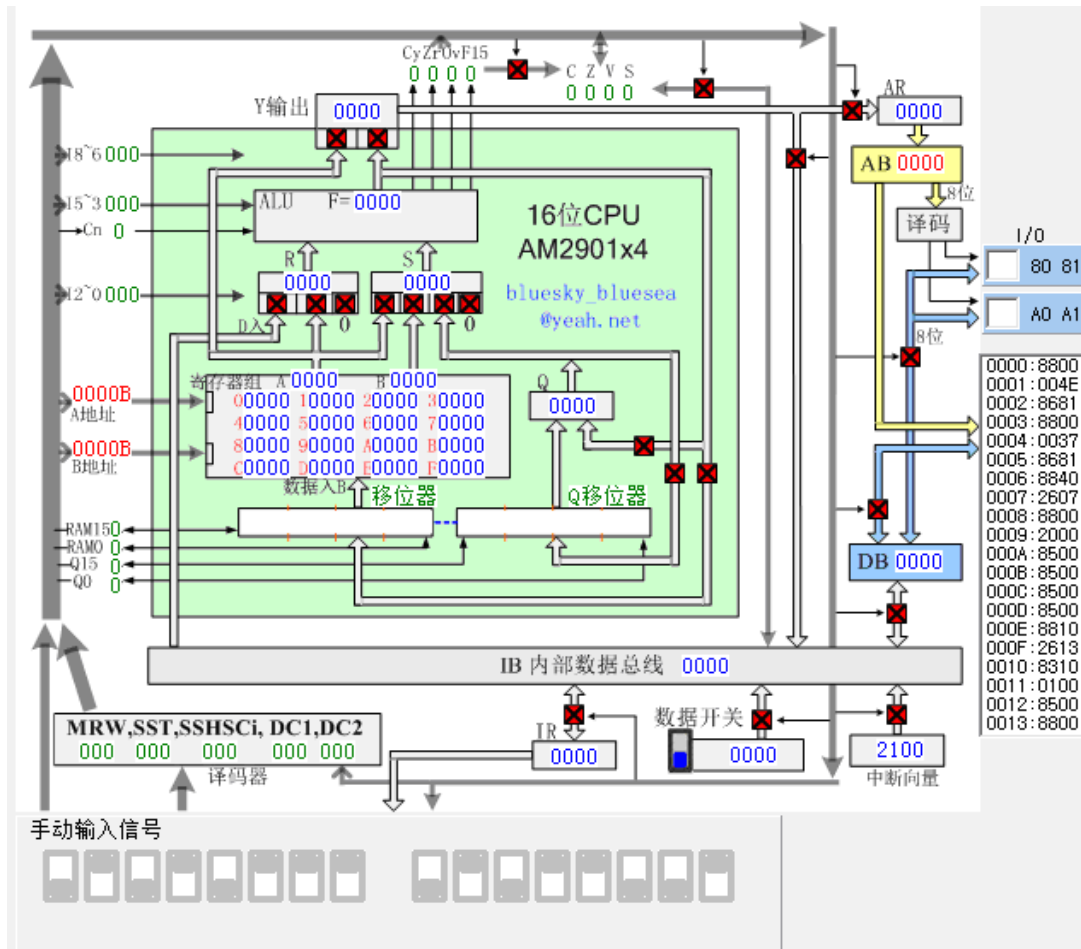
例如：在终端 1 键入“>D 0000”，显示内存信息的时间会比较长。此时按下中断源按钮，终端 1 窗口中程序运行会暂停，在终端 2 窗口中显示中断服务程序信息，完成后会继续运行终端 1 的程序。若在此期间，又有更高一级的中断请求，则转向执行该级的中断服务程序，不响应新的更低级中断请求。



第十章 综合设计

10.1 综合设计基础知识

◆ 实验系统总体框图:



实验系统总体框图

◆ 微码控制开关和数据开关图:



总线控制信号:

◆ **DC1:** 向 IB 总线写数据的控制信号

DC1 编码	功能说明
000	无操作
001	ALU 的 Y 输出=>IB
010	IRL=>IB
011	C、Z、V、S、P1、P0=>IB
100	未使用，待扩展
101	中断向量=>IB
110	未使用，待扩展
111	16 位数据开关=>IB

◆ **DC2:** 与寄存器 IR 和 AR 接收、中断有关的控制信号

DC2 编码	功能说明
000	无操作
001	IB=>IR
010	未使用，待扩展
011	Y 输出=>AR
100	IB=>P1、P0
101	中断新等级=>P1、P0
110	允许中断
111	禁止中断

- ◆ **MRW**: 与内存、I/O 读写有关的控制信号: /MIO、REQ、/WE; (缩写为 MRW)

/MIO REQ /WE	功能说明
000	写内存。IB=>DB=>MEM
001	读内存。MEM=>DB=>IB
010	写 I/O。IBL=>DBL=>I/O
011	读 I/O。I/O=>DBL=>IBL
1XX	无上述读写操作

运算器控制信号:

- ◆ **I8~6**: 通用寄存器组和 Q 寄存器的结果选择以及运算结果 Y 输出的选择。AM2901 芯片固有规则, 不能改变。**注**: A 表示 A 锁存器。B 表示“B 地址”确定的寄存器。

I8~6 编码	操作	说明
000	$F \Rightarrow Y, F \Rightarrow Q$	
001	$F \Rightarrow Y$	
010	$A \Rightarrow Y, F \Rightarrow B$	
011	$F \Rightarrow Y, F \Rightarrow B$	
100	$F \Rightarrow Y, F/2 \Rightarrow B, Q/2 \Rightarrow Q$	F、Q 分别右移
101	$F \Rightarrow Y, F/2 \Rightarrow B$	F 右移
110	$F \Rightarrow Y, 2F \Rightarrow B, 2Q \Rightarrow Q$	F、Q 分别左移
111	$F \Rightarrow Y, 2F \Rightarrow B$	F 左移

- ◆ **I5~3**: 确定运算公式, 一共 8 种运算, 由 AM2901 芯片确定, 不能改变。

I5~3 编码	运算公式 $F = R ? S$
000	$F = R + S + C_{in}$
001	$F = S - R - C_{in}$
010	$F = R - S - C_{in}$
011	$F = R \vee S$
100	$F = R \wedge S$
101	$F = /R \wedge S$
110	$F = R \oplus S$
111	$F = /(R \oplus S)$

- ◆ **I2~0**: 确定运算数据 R, S 来源。表中 D 实际上来自内部数据总线 IB。

A、B 表示 A、B 锁存器, Q 表示 Q 寄存器。

R、S 来源规则由 AM2901 芯片确定, 不能改变。

I2~0 编码	R、S 取值
000	R=A, S=Q
001	R=A, S=B
010	R=0, S=Q
011	R=0, S=B
100	R=0, S=A
101	R=D, S=A
110	R=D, S=Q
111	R=D, S=0

◆ SST: 状态位寄存器 C、Z、V、S 数值确定

SST	C	Z	V	S	说明
000	C	Z	V	S	C、Z、V、S 保持不变
001	Cy	Zr	Ov	F15	接收 ALU 的标志位输出的值
010	IB15	IB14	IB13	IB12	IB \Rightarrow C、Z、V、S
011	0	Z	V	S	C=0, Z、V、S 保持不变
100	1	Z	V	S	C=1, Z、V、S 保持不变
101	RAM0	Z	V	S	C=RAM0, Z、V、S 保持不变
110	RAM15	Z	V	S	C=RAM15, Z、V、S 保持不变
111	Q0	Z	V	S	C=Q0, Z、V、S 保持不变

◆ SSHSCi 功能 1: 确定 Cin 值, Cin 是 16 位运算器的最低位进位输入信号, 只参与 3 种算数运算。

SSHSCi	Cin 取值	说明
000 或 1xx	0	
001	1	
010	C	C 是状态位寄存器
011	无效	

◆ SSHSCi 功能 2: 移位操作时 RAM15、RAM0、Q15 和 Q0 作为输入数值的确定。

左移运算 I8~6=110、111		
SSHSCi	RAM0	Q0
100 或 0xx	0	0
101	C	0
110	Q15	/F15
111	0	0

右移运算 I8~6=100、101		
SSHSCi	RAM15	Q15
100 或 0xx	0	0
101	C	0
110	Cy	RAM0
111	F15 \oplus Ov	RAM0

◆ A 口、A 地址、A 锁；B 口、B 地址、B 锁

AM2901 用来确定要读写寄存器的信号是“A 地址”、“B 地址”，即实际起作用的控制信号。读寄存器时，A、B 地址分别确定 A 锁存器（A）、B 锁存器（B）的来源寄存器。写寄存器时，B 地址确定运算结果 F 存入哪个寄存器。

微码控制开关（脱机）或者微程序（微程序+联机）状态：由 IR、A 口、B 口相结合确定 A、B 地址。

组合逻辑（组合逻辑+联机）状态：由 IR、A、B 相结合确定 A、B 地址。

微程序下地址控制信号：

◆ CI3~0：为 AM2910 提供命令码 I3~I0。

CI3~0	功能
0000（0 号命令）	AM2910 初始化，下一步微地址=00。
0010（2 号命令）	MAP 映射，/MAP 信号有效。下一步的微地址来自 MAP 映射地址。
0011（3 号命令）	条件微转移，使输入的/CC 起作用，下一步微地址由 SCC 确定。见下表。
1110（14 号命令）	顺序执行，下一步微地址是当前微地址加 1。
CI3~0 其它值	错误。下一步微地址=00。

◆ SCC3~0：给出形成 AM2910 的/CC 信号的条件码。当上表中 CI3~0=0011 时，本表有效：

SCC3~0	<p>下面的条件满足时，使/CC=0，即/PL 有效，即下一步微地址来自当前微指令的“下址”字段。相当于“跳转执行”。</p> <p>下面的条件不满足时，使/CC=1，则下一步微地址是当前微地址加 1，即“顺序执行”。</p>
0000	必转，无条件跳转“下址”
0010	当有新中断请求即/INT=0 时，转“下址”
0100	JRC、JRNC、JRZ、JRNZ 条件不成立时，转“下址”
0101	JRS、JRNS 条件不成立时，转“下址”
0110	IR10=0 时，转“下址”（为 IN/OUT 指令服务）
0111	IR8=1 时，转“下址”（为 PSH/F，POP/F 指令服务）
其它值	判定为不满足条件，不转，顺序执行。

10.2 模拟软件中设计实现新指令

10.2.1 实验目的

深入了解系统各控制信号功能，学习设计新指令的方法。

10.2.2 实验内容

- 1、用手动方式验证新指令各步功能
- 2、以微程序和组合逻辑方式实现新指令
- 3、编写包含新指令的程序并运行验证正确性

设计新指令注意事项:

明确要实现的指令格式、功能。在执行流程中必须遵从现有约束条件。

完成设计新指令具体步骤如下:

- 1、确定指令格式和功能,遵循现有规则,尽量与已有指令的格式和分类方法保持一致。
- 2、划分指令执行步骤,设计每一步的执行功能,确定每个控制信号的取值。

本步是最重要的步骤。最好参照已有指令的处理方法和实现步骤,毕竟现有指令相对成熟,在此基础上修改会更稳妥。很多指令需要的操作如读内存和 PC+1 等,都已经实现,直接利用即可。具体实现时还要注意各方面是否有冲突,如不能多方同时写 IB、不能对寄存器 AR 同时写入与读出等。

本步在“单步 + 脱机”操作下,通过观察动画现象,既可以快速设计,也可以验证每一步的功能。

- 3、在指令流程表中填写每一步的控制信号值。用组合逻辑方式实现时,T 节拍状态应符合 A、B、C、D 分组规则。
- 4、将控制信号写入到微程序文件或硬布线文件中。并且修改相应的 MAP 映射文件和指令扩展文件。
- 5、写一个包含新指令的程序,通过运行该程序检查执行结果的正确性,来初步判断设计是否正确;如果有问题,可通过“单步 + 手动置指 + 联机”方式反复调试,直到完全正确。

10.2.3 实验步骤

10.2.3.1 设计新指令功能及分析

新指令功能要求:

“SR+[ADR]->DR”: 源寄存器与内存某单元的内容相加,结果存入目的寄存器。

指令运行时的设定: SR 寄存器用 R1, DR 寄存器用 R2, 内存地址 ADR 定为 2100H。本指令所在的内存地址是 2000H。R1 与内存 2100H 中的内容相加,结果存入 R2。

在运行新指令时,要提前在 SR 和内存单元设置数值,用于保存结果的 DR 清空,保证最终结果的正确性。

新指令实现分析:

指令格式:

该指令的参数 ADR 已经是 16 位,占用 1 个字,加上开头的指令码等,新指令需要 2 字: 第 1 字是指令码、目的寄存器 DR、源寄存器 SR; 第 2 字是内存地址 ADR。根据

“新指令功能要求”寄存器设定，机器码暂定为“xx21 2100”。其中“xx”要在微指令实现部分确定；“21”表示目的寄存器 DR 是 R2，源寄存器 SR 是 R1；“2100”表示内存地址。

指令复杂度：

计算结果可以直接存入目的寄存器，一步内完成。

进行运算的两个数值一个在寄存器，可以直接读取，一步内完成；另一个数值在内存，需要两步完成读取：第一步（1.1）地址寄存器 AR 置数（内存地址），第二步（1.2）从内存取数值并参与运算。

确定第二个数值的内存地址是指令的第 2 字，读指令时只读取了第 1 字，第 2 字需要自己设计读出步骤。这里又需要两步：第一步（2.1）地址寄存器 AR 置数（PC+1 的值），第二步（2.2）从内存取数值。

综上所述，一共需要四步完成：

第一步(2.1)：地址寄存器 AR 置数（PC+1 的值）；

第二步(2.2)：从内存取数值作为未来的内存地址，暂存一下；

第三步(1.1)：将暂存的值（内存地址）送到地址寄存器 AR；

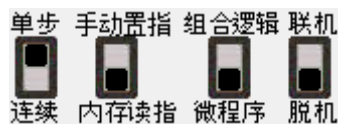
第四步(1.2)：从内存取数值并参与运算，结果直接存入目的寄存器。

新指令实现步骤：

- 1、PC->AR, PC+1->PC：访问内存前，给 AR 置数 2001H，即新指令的第 2 字地址。
- 2、MEM->Q：取出指令后跟随的数据，存入 Q，是数值存放的地址（2100H）。
- 3、Q->AR：Q 值（2100H）送 AR，为取内存数值做准备。
- 4、MEM+SR->DR：内存取数值，直接与源寄存器 R1 相加，结果送目的寄存器 R2。

10.2.3.2 手动方式实现新指令

- 1、运行程序 “ZCHPC1.exe”。
- 2、将左下方的控制开关置为 1xx0（单步、X、X、脱机）；



- 3、菜单“设置”->“寄存器”，设置 R1=2，R2=0，R5(PC)=2001H，IR=0021H。
菜单“设置”->“修改内存”，设置地址 2001H 地址内容 2100H，2100H 数值为 0003H。
- 4、主界面菜单“窗口”->“显示动画”，打开动画窗口。
- 5、按【RESET】键，进行初始化。
- 6、7……依次验证新指令各步操作。

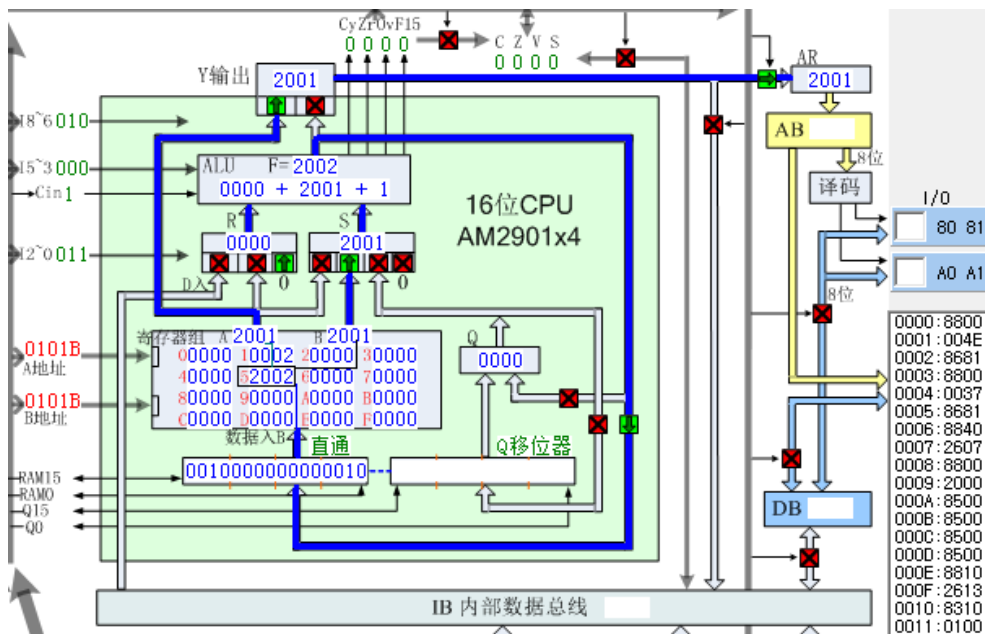
6、新指令第一步：PC->AR, PC+1->PC。

微码控制开关为：（“x”表示任意，可以是“0”或“1”）

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	011	1xx	010	000	011	000	001	0101	0101

16 位数据开关无关，可以任意值。

按【START】键，观察数据流动。



存储地址 2001 到 AR

新指令第 1 步：指令第 2 字的存储地址 2001 进入 AR，准备读取第 2 字。

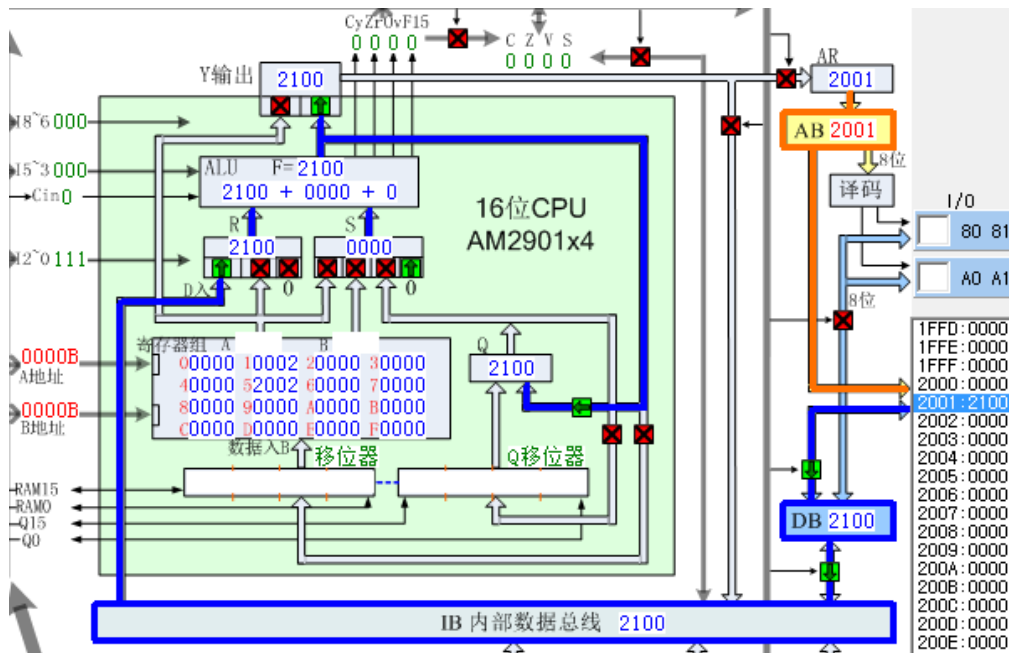
7、新指令第二步：MEM->Q

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	000	001	000	000	111	000	000	无关	无关

16 位数据开关无关，可以任意值。

按【START】键，观察数据流动。



数值 2100 送入 Q

新指令第 2 步：内存 2001 单元的数值 2100 送入 Q

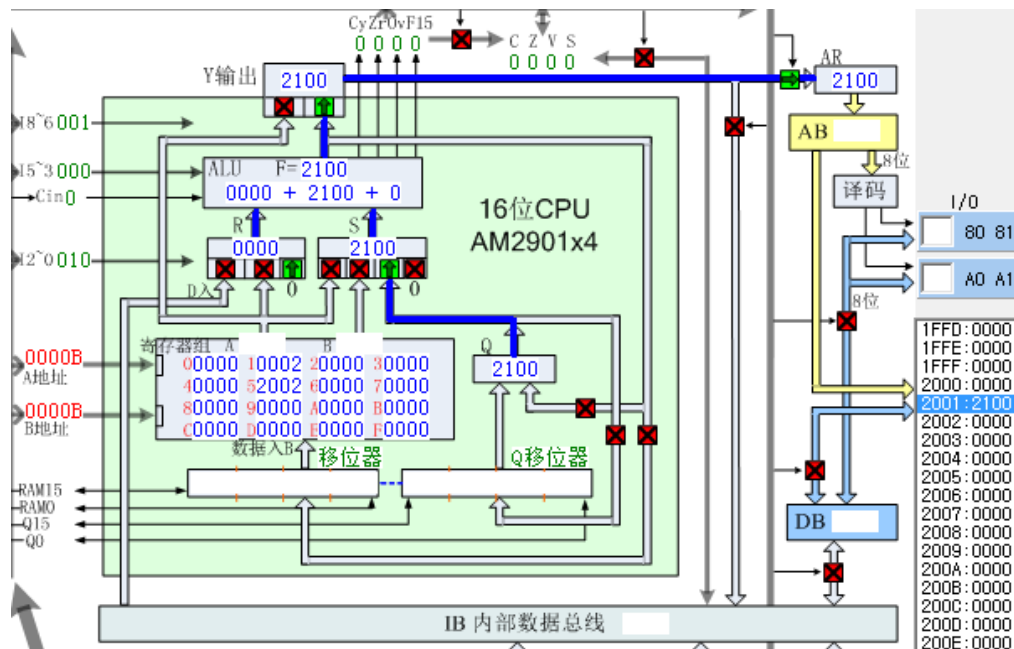
8、新指令第三步：Q->AR

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	011	1xx	001	000	010	000	000	无关	无关

16 位数据开关无关，可以任意值。

按【START】键，观察数据流动。



Q 值 2100 送入 AR

新指令第 3 步：Q 值 2100 送入 AR，为读取 2100 单元内容（0003）做准备

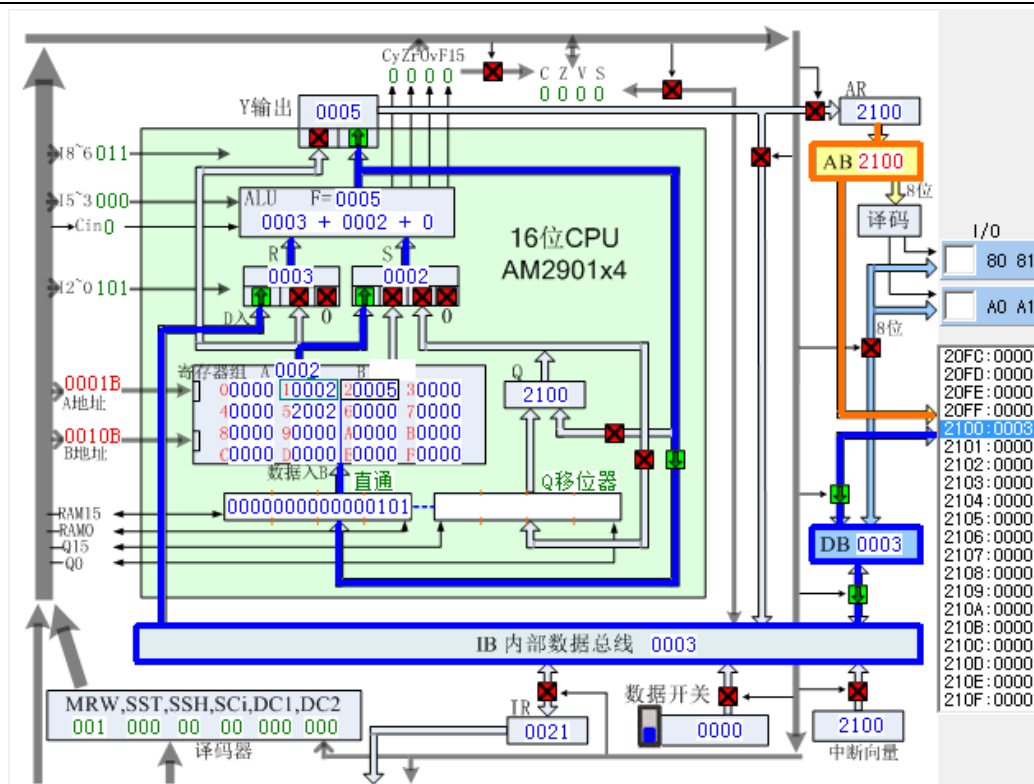
9、新指令第四步：MEM+SR->DR

微码控制开关为：

DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
000	000	001	011	000	101	000	000	1000	1000

16 位数据开关无关，可以任意值。

按【START】键，观察数据流动。



运算结果 0005 存入 R2

新指令第 4 步：观察到 0003+0002 运算结果 0005 存入 R2 寄存器。指令正确完成。

10.2.3.3 微程序方式实现新指令

在手动操作的基础上，还需要添加映射关系，确定新指令的指令符号，修改相关文件等，才能以微程序方式实现新指令。

一、指令码、机器码和类型

指令符号：为了能编译含有新指令的程序，必须给新指令命名，指令符号最长 4 个字母。这里给新指令命名为“ADRY”。

机器码设计：已知新指令 4 步能完成功能，在“组合逻辑”方式中步数范围内（组合逻辑方式最多 4 步），为了将来用组合逻辑方式实现该指令，编码要符合 ABCD 的 D 组扩展指令规则：“111* 1***”，新指令不能与现有指令相同。综上所述，新指令机器码设定为：1110 1001（参照“分组规则表”）。

为了能编译新指令，需要确定其类型。指令类型如下：

类型	格式	长度	分段
1	OP DR,SR	1 字	2 操作数
12	OP DR, [SR]	1 字	2 操作数

13	OP [DR], SR	1 字	2 操作数
21	OP BYTE(PORT)	1 字	1 操作数
22	OP SR	1 字	1 操作数
23	OP DR	1 字	1 操作数
3	OP	1 字	0 操作数
411	OP DR, DATA(16)	2 字	2 操作数
412	OP DR,[ADDR]	2 字	2 操作数
413	OP [ADDR],SR	2 字	2 操作数
42	OP DR, OFFSET[SR]	2 字	2 操作数
5	OP OFFSET	1 字	1 操作数
6	OP ADDR(16)	2 字	1 操作数

注意：本表格与实现指令的“微指令方式”和“组合逻辑方式”无关，是为指令的编译和反编译服务的。

本指令长度是 2 字，2 操作数，用到了 SR、DR、ADDR，最符合的格式是“OP DR, OFFSET[SR]”，类型定为“42”。

修改文件“InsExt.txt”，将新指令的“指令名”、“机器码”、“类型”添加如下：

ADRY	11101001	42
------	----------	----

二、实现微程序

由手动操作可知，指令“SR+[ADR]->DR”可用 4 步实现，微码控制开关为：

	DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
1	000	011	1xx	010	000	011	000	001	0101	0101
2	000	000	001	000	000	111	000	000	无关	无关
3	000	011	1xx	001	000	010	000	000	无关	无关
4	000	000	001	011	000	101	000	000	1000	1000

为了格式统一，将表中的“x”和“无关”替换成“0”，即为下表

	DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
1	000	011	100	010	000	011	000	001	0101	0101
2	000	000	001	000	000	111	000	000	0000	0000
3	000	011	100	001	000	010	000	000	0000	0000
4	000	000	001	011	000	101	000	000	1000	1000

这 4 步执行顺序是：1->2->3->4，顺序执行，中途不向其他地址跳转。

微程序下址控制部件是 AM2910，它的控制信号是“CI3~0”和“SCC3~0”。

- 如果要求是“顺序进行”，则置“CI3~0=1110”，上面的“1->2”“2->3”“3->4”都符

合。此时微指令的“下址”栏和“SCC3~0”是无关的，可以任意值，一般记作“0”。

- 根据“微程序流程图”，最后“04”执行后不是顺序执行，通常转至“30”，则置其“下址”=30。为确保转到“下址”，置“CI3~0=0011，SCC3~0=0000”。

综合上述，微程序方式这4步的“下址”“CI3~0”和“SCC3~0”值如下：

	下址	CI3~0	SCC3~0		下址	CI3~0	SCC3~0
1	无关	1110	无关	转换=>	00	1110	0000
2	无关	1110	无关		00	1110	0000
3	无关	1110	无关		00	1110	0000
4	30	0011	0000		30	0011	0000

打开微程序记录文件“MicroIns.txt”，找未占用地址。例如“40”地址开始有4个连续地址可用，则占用地址“40、41、42、43”，将上述的微码控制开关与下址控制信号添加在文件内，如下：

下址	CI3~0	SCC3~0	DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A□	B□
40:00	1110	0000	000	011	100	010	000	011	000	001	0101	0101
41:00	1110	0000	000	000	001	000	000	111	000	000	0000	0000
42:00	1110	0000	000	011	100	001	000	010	000	000	0000	0000
43:30	0011	0000	000	000	001	011	000	101	000	000	1000	1000

三、修改映射文件

指令运行中，执行到本新指令时，需要进行映射操作得到微程序的入口地址，已确定新指令的机器码是“1110 1001”，入口地址是“40”，在文件“InsMap.txt”添加如下：

1110 1001 40

四、指令验证

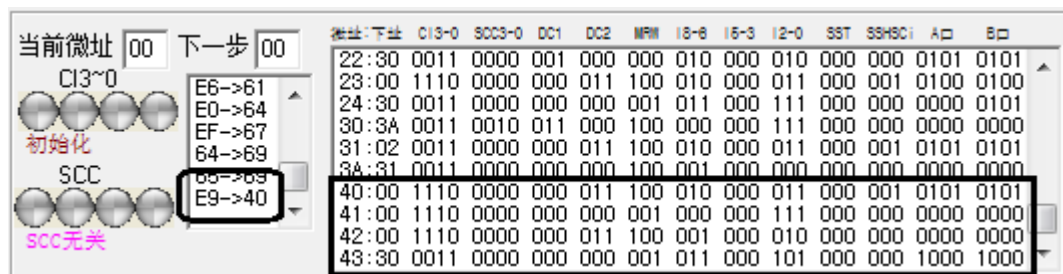
1、编写包含新指令的程序“testADRY.asm”，如下

```

org 2000h
adry r2, 2100h[r1]
ret
end
  
```

这个程序只有两条语句，一条新指令一条返回。功能是“R1 与内存 2100H 中的内容相加，结果存入 R2”，执行本程序前，需要手动在相关寄存器和内存中设置运算数值。

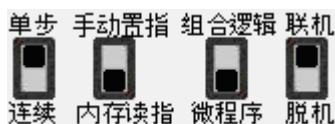
2、运行程序“ZCHPC1.exe”，观察是否正确读入了修改过的“InsMap.txt”和“MicroIns.txt”内容。



查看读入信息的正确性

图中的“E9->40”表示新指令机器码“11101001”对应的微程序入口地址是“40”。

3、置控制开关为 **1001**（单步、内存读指、微程序、联机）

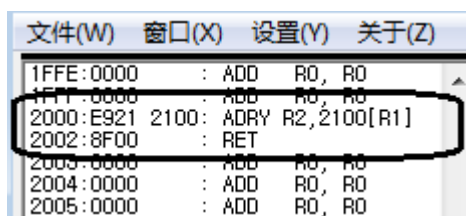


4、主界面菜单“设置”->“PC 初始值”，确保窗口内选择如下图



5、主界面菜单“文件”->“打开 ASM 源文件”，打开“testADRY.asm”按“编译”后，再按“导入内存”。在如下界面可见到程序读入后显示的机器码和反汇编语句。

如果编译有问题，需检查文件“InsExt.txt”是否修改正确。



6、菜单“设置”->“寄存器”，设置 R1=2, R2=0, IR=0。

菜单“设置”->“修改内存”，设置地址 2100H 数值为 0003H。

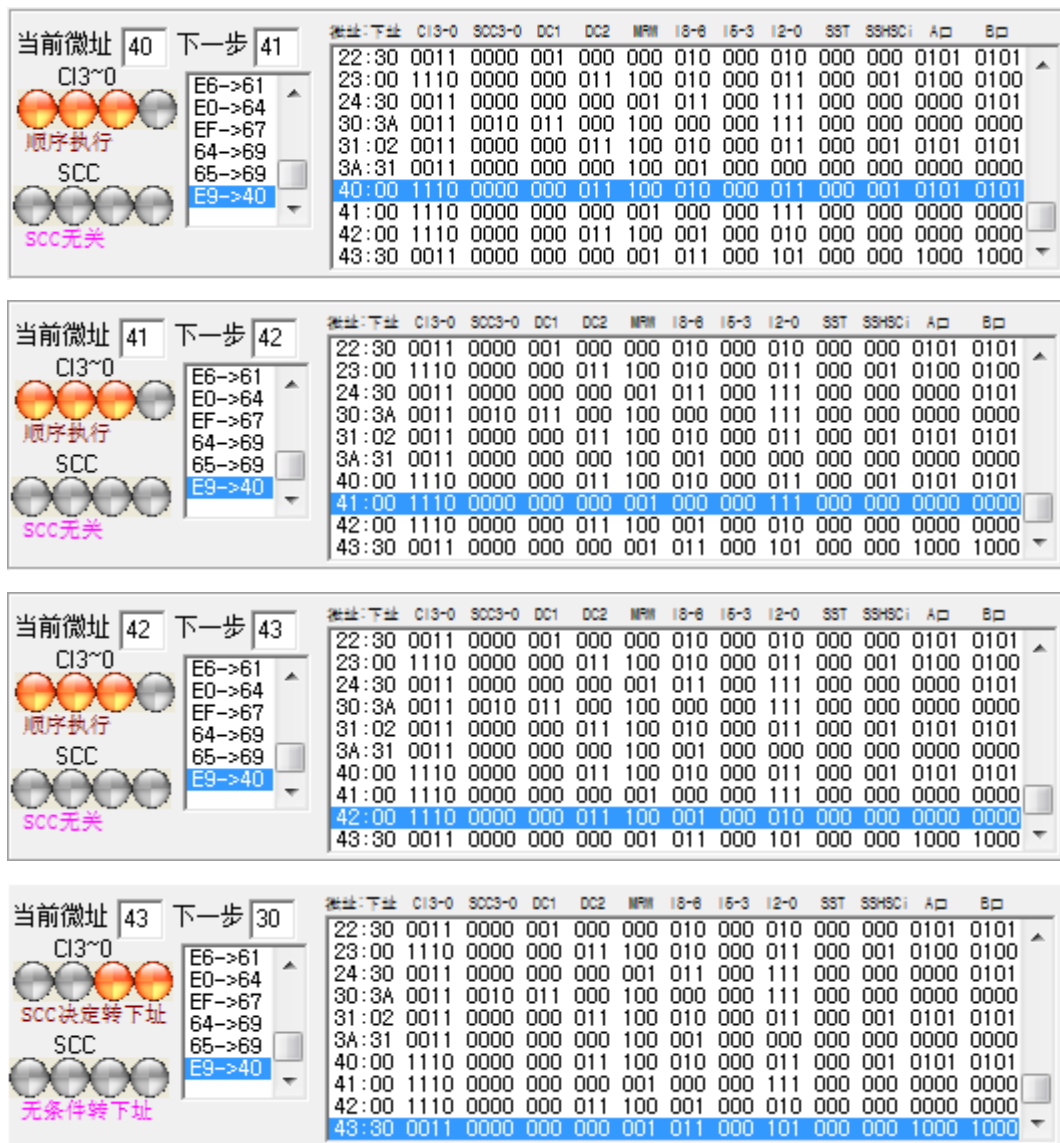
7、按 **【RESET】** 键，进行初始化。

连续按 **【START】** 键，观察微程序信息窗口、微信号、IR 值、程序窗口变化。

可观察到微程序在沿着地址 40->41->42->43 依次执行，完成新指令功能。

运行到微程序地址“43”时，新指令结束，R2 寄存器为运算结果 R2 0005。

如果打开动画窗口，可以看到类似手动操作时的动画过程。



当前微址 40 下一步 41

CI3~0

顺序执行

SCC

SCC无关

地址:下址	CI3~0	SCC3~0	DC1	DC2	MRM	I8~6	I5~3	I2~0	SST	SSHSC1	A口	B口
22:30	0011	0000	001	000	000	010	000	010	000	000	0101	0101
23:00	1110	0000	000	011	100	010	000	011	000	001	0100	0100
24:30	0011	0000	000	000	001	011	000	111	000	000	0000	0101
30:3A	0011	0010	011	000	100	000	000	111	000	000	0000	0000
31:02	0011	0000	000	011	100	010	000	011	000	001	0101	0101
3A:31	0011	0000	000	000	100	001	000	000	000	000	0000	0000
40:00	1110	0000	000	011	100	010	000	011	000	001	0101	0101
41:00	1110	0000	000	000	001	000	000	111	000	000	0000	0000
42:00	1110	0000	000	011	100	001	000	010	000	000	0000	0000
43:30	0011	0000	000	000	001	011	000	101	000	000	1000	1000

当前微址 41 下一步 42

CI3~0

顺序执行

SCC

SCC无关

地址:下址	CI3~0	SCC3~0	DC1	DC2	MRM	I8~6	I5~3	I2~0	SST	SSHSC1	A口	B口
22:30	0011	0000	001	000	000	010	000	010	000	000	0101	0101
23:00	1110	0000	000	011	100	010	000	011	000	001	0100	0100
24:30	0011	0000	000	000	001	011	000	111	000	000	0000	0101
30:3A	0011	0010	011	000	100	000	000	111	000	000	0000	0000
31:02	0011	0000	000	011	100	010	000	011	000	001	0101	0101
3A:31	0011	0000	000	000	100	001	000	000	000	000	0000	0000
40:00	1110	0000	000	011	100	010	000	011	000	001	0101	0101
41:00	1110	0000	000	000	001	000	000	111	000	000	0000	0000
42:00	1110	0000	000	011	100	001	000	010	000	000	0000	0000
43:30	0011	0000	000	000	001	011	000	101	000	000	1000	1000

当前微址 42 下一步 43

CI3~0

顺序执行

SCC

SCC无关

地址:下址	CI3~0	SCC3~0	DC1	DC2	MRM	I8~6	I5~3	I2~0	SST	SSHSC1	A口	B口
22:30	0011	0000	001	000	000	010	000	010	000	000	0101	0101
23:00	1110	0000	000	011	100	010	000	011	000	001	0100	0100
24:30	0011	0000	000	000	001	011	000	111	000	000	0000	0101
30:3A	0011	0010	011	000	100	000	000	111	000	000	0000	0000
31:02	0011	0000	000	011	100	010	000	011	000	001	0101	0101
3A:31	0011	0000	000	000	100	001	000	000	000	000	0000	0000
40:00	1110	0000	000	011	100	010	000	011	000	001	0101	0101
41:00	1110	0000	000	000	001	000	000	111	000	000	0000	0000
42:00	1110	0000	000	011	100	001	000	010	000	000	0000	0000
43:30	0011	0000	000	000	001	011	000	101	000	000	1000	1000

当前微址 43 下一步 30

CI3~0

SCC决定转下址

SCC

无条件转下址

地址:下址	CI3~0	SCC3~0	DC1	DC2	MRM	I8~6	I5~3	I2~0	SST	SSHSC1	A口	B口
22:30	0011	0000	001	000	000	010	000	010	000	000	0101	0101
23:00	1110	0000	000	011	100	010	000	011	000	001	0100	0100
24:30	0011	0000	000	000	001	011	000	111	000	000	0000	0101
30:3A	0011	0010	011	000	100	000	000	111	000	000	0000	0000
31:02	0011	0000	000	011	100	010	000	011	000	001	0101	0101
3A:31	0011	0000	000	000	100	001	000	000	000	000	0000	0000
40:00	1110	0000	000	011	100	010	000	011	000	001	0101	0101
41:00	1110	0000	000	000	001	000	000	111	000	000	0000	0000
42:00	1110	0000	000	011	100	001	000	010	000	000	0000	0000
43:30	0011	0000	000	000	001	011	000	101	000	000	1000	1000

新指令执行的4步

10.2.3.4 组合逻辑方式实现新指令

使用组合逻辑方式实现新指令与微程序方式类似，需要提前进行设计和一些设置。

一、指令码和机器码

组合逻辑方式下的新指令的指令码和机器码与微程序相同。新指令起名为“ADRY”。机器码为“1110 1001”。类型定为“42”。为了能编译新指令，修改文件“InsExt.txt”，添加如下，相同内容只需要一行。

ADRY	11101001	42
------	----------	----

二、实现组合逻辑控制信息

由手动操作可知，指令“SR+[ADR]->DR”可用4步实现，微码控制开关为：

	DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A 口	B 口
1	000	011	100	010	000	011	000	001	0101	0101
2	000	000	001	000	000	111	000	000	0000	0000
3	000	011	100	001	000	010	000	000	0000	0000
4	000	000	001	011	000	101	000	000	1000	1000

这4步执行顺序是：1->2->3->4，顺序执行，中途不向其他地址跳转。

组合逻辑文件是用十六进制记录，A、B地址的确定方法也不同，“SA、SB、A口、B口”改为“A、B”，修改记录如下表：

	DC1	DC2	MRW	I8~6	I5~3	I2~0	SST	SSHSCi	A	B
1	0	3	4	2	0	3	0	1	5	5
2	0	0	1	0	0	7	0	0	0	0
3	0	3	4	1	0	2	0	0	0	0
4	0	0	1	3	0	5	0	0	R	R

新指令是4步完成，属于D组。根据节拍T在D组的跳变规则（参见“节拍T的规则”），这4步对应的T已经确定，分别是：

	T
1	00110
2	00100
3	00111
4	00101

新指令的机器码已经确定是“1110 1001”，组合逻辑用十六进制表示，记为“E9”。新指令运行时不需要与CZS等标志位寄存器匹配，这4步对应的CZS都是“---”。打开组合逻辑记录文件“HardWiredLogic.txt”，将上述内容添加在文件中，如图：

T	IRH	CZS	DC1	DC2	MRW	I86	I53	I20	SST	SSHSCi	A	B
00110	E9	---	0	3	4	2	0	3	0	1	5	5
00100	E9	---	0	0	1	0	0	7	0	0	0	0
00111	E9	---	0	3	4	1	0	2	0	0	0	0
00101	E9	---	0	0	1	3	0	5	0	0	R	R

三、指令验证

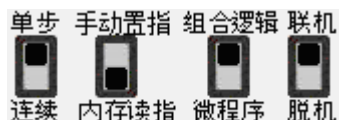
1、编写包含新指令的程序“testADRY.asm”，如下

```
org 2000h
adry r2, 2100h[r1]
ret
end
```

2、运行程序“ZCHPC1.exe”，观察是否正确读入了“HardWiredLogic.txt”

00110	E9	---	=>	0	3	4	2	0	3	0	1	5	5
00100	E9	---	=>	0	0	1	0	0	7	0	0	0	0
00111	E9	---	=>	0	3	4	1	0	2	0	0	0	0
00101	E9	---	=>	0	0	1	3	0	5	0	0	R	R

3、置控制开关为 **1011**（单步、内存读指、组合逻辑、联机）

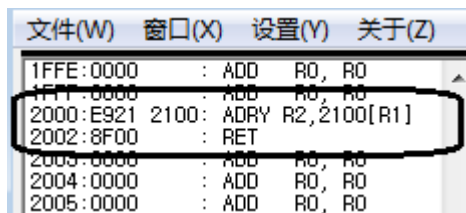


4、主界面菜单“设置”->“PC 初始值”，确保窗口内选择如下图



5、主界面菜单“文件”->“打开 ASM 源文件”，打开“testADRY.asm”按“编译”后，再按“导入内存”。在如下界面可见到程序读入后显示的机器码和反汇编语句。

如果编译有问题，需检查文件“InsExt.txt”是否修改正确。



6、菜单“设置”->“寄存器”，设置 R1=2，R2=0，IR=0。

菜单“设置”->“修改内存”，设置地址 2100H 数值为 0003H。

7、按 **【RESET】** 键，进行初始化。

连续按 **【START】** 键，观察组合逻辑信息窗口、微信号、IR 值、程序窗口变化。

可观察到 T 沿着 00110->00100->00111->00101 依次执行，完成新指令功能。

新指令结束，R2 寄存器为 R2 0005。

如果打开动画窗口，可以看到类似手动操作时的动画过程。观察数据流动并分析。

10.2.4 设计新指令

【思考题】:

实现新指令:

- 1、NXOR DR,SR 功能: $\neg(DR \oplus SR) \rightarrow DR$
- 2、SWRM DR,[SR] 功能: DR 与[SR]值互换
- 3、ADTW DR 功能: DR 增 2
- 4、ADRM DR,[SR] 功能: $DR+[SR] \rightarrow DR$
- 5、设计一条指令，其功能是把一个通用寄存器中的内容进行半字交换，即将其高低位字节的内容对换。
- 6、设计一条指令，完成对存放在低位字节的8位有符号补码数进行符号扩展，即将其变为16位的同值的补码数，仍保存在原寄存器中。

10.2.5 实验报告要求

- 1、按照“实验步骤”的内容完成实验内容，熟悉设计过程。
- 2、完成“设计新指令”，仿照教材中的设计过程设计新的指令，完成相应功能。

第十一章 BASIC 语言程序设计

11.1 BASIC 相关知识

实验系统的 BASIC 语言是用汇编程序完成的，运行时需要用到 29 条基本指令和 19 条扩展指令，是在汇编语言层次上的高级语言。本章实验对理解计算机软件系统的层次结构，特别是几个层次的语言（机器语言、汇编语言、高级语言）的功能和用法上的异同很有用，用 BASIC 语言写几个程序，可以体会一下高级语言和汇编语言在处理能力和使用的方便程度等方面的区别。

本实验的 BASIC 例子有：整数排序、求素数、汉诺塔、8 皇后、验证歌德巴赫猜想、绘制三角（正弦）函数图形。这些程序最终是通过指令系统执行的。目的是学习使用最简单的高级语言完成程序设计的过程，体会两种语言在语句格式和功能等方面的区别，感受高级语言的优越性。

编写 BASIC 程序时，在语句前要标示行号。如果只输入行号后面没有语句，则此行号无效；如果先前存在此行号的语句，则被替换。输入完 BASIC 程序之后，通过 run 命令运行这个程序。

本系统的 BASIC 程序操作命令有：

new	清除内存缓冲区已有内容，准备输入一个新的 BASIC 程序
run	运行已输入的 BASIC 程序
delete	删除 BASIC 程序中的指定行号的语句
list	显示已经输入的 BASIC 程序所有语句
system	退出 BASIC 语言，返回操作系统

本系统支持的 BASIC 语句有：

let	变量赋值，该语句名通常可以省略
dim	定义数组变量
input	从键盘向变量输入新值
print	在屏幕显示常量、变量的值
for...next	建立一段循环执行的程序段
goto	语句跳转到指定的标号之处
gosub	调用指定标号的子程序
return	子程序返回
end	程序结束

本系统支持的 BASIC 表达式有：

算术运算符：+（加），-（减），*（乘），/（除），\（整除），^（乘方），mod（取模）

逻辑运算符：not（取反），and（与），or（或），xor（异或）

关系符号 : > (大于), >= (大于等于), < (小于), <= (小于等于), <> (不等于)

数学函数 : sin (正弦), cos (余弦), tan (正切), sqr (开平方)
log (以自然常数 e 为底的对数), exp (e 为底的指数)
sgn (取数的符号位), abs (绝对值), int (取整)

11.2 模拟软件中 BASIC 语言程序设计实验

11.2.1 实验目的

- 1、学习和了解用于 BASIC 解释执行程序中的子程序;
- 2、学习和理解计算机软件系统的层次结构;
- 3、了解高级语言和汇编语言在处理能力和方便程度等方面的区别。

11.2.2 实验内容

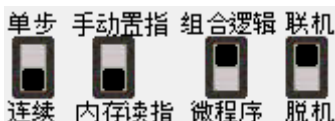
- 1、了解并练习使用 BASIC 基本命令;
- 2、了解解释程序已经实现的语句和对表达式的支持;
- 3、使用相关知识练习编写简单的典型 BASIC 程序。

11.2.3 实验要求

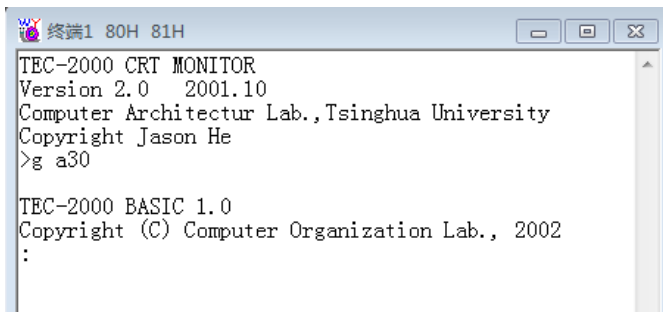
在使用模拟软件之前, 应先熟悉 BASIC 操作命令、语法和表达式。

11.2.4 实验步骤

- 1、运行程序 “ZCHPC1.exe”
- 2、置控制开关为 0011 (连续、内存读指、组合逻辑、联机)



- 3、设置速度条到最高
- 4、按【RESET】键初始化, 再按【START】键, 终端 1 显示初始信息
- 5、在“终端 1”窗口输入 “G A30” 回车, 启动 BASIC, 显示如下:



启动 BASIC

6、在屏幕提示符“:”后面分别输入下面给出的 6 个小程序例子。

例 1. 整数排序

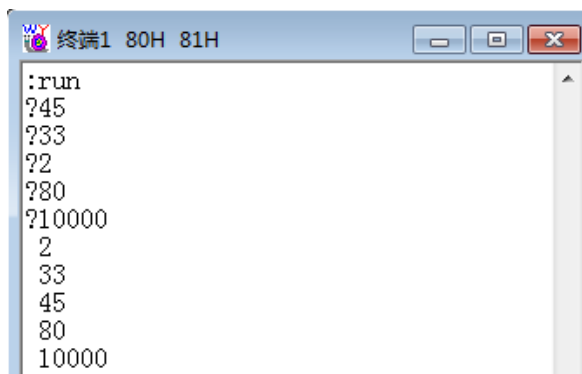
依次输入 5 个整数，进行排序操作，输出最终排序结果。

<1> 在命令行提示符“:”后面输入程序：

```
:new
:10 for i=1 to 5
:20 input a(i)
:30 next i
:40 for i=1 to 4
:50 for j=i+1 to 5
:60 if a(i)>a(j) then b=a(i):a(i)=a(j):a(j)=b
:70 next j
:80 next i
:90 for i=1 to 5
:100 print a(i)
:110 next i
:120 end
```

<2> 输入 list 命令察看输入的内容是否正确，如果不正确可以直接修改错误的语句，只要在提示符后重新输入行号及其对应的正确内容即可。

<3> 输入 run 命令运行程序，依次输入 5 个整数，自行显示排序结果并显示。可多次运行。



```
终端1 80H 81H
:run
?45
?33
?2
?80
?10000

2
33
45
80
10000
```

整数排序

<4>运行结束后如果输入 system 命令则退出 BASIC 返回到操作系统。

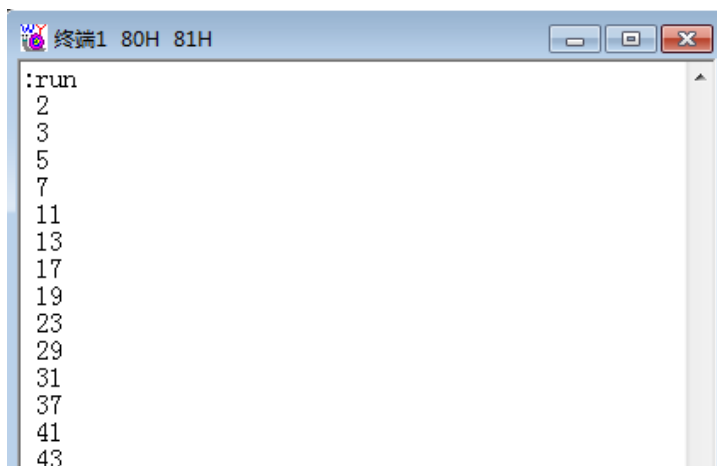
例 2. 求素数

在指定的数据（100）范围内，找出除了能被 1 和这个数本身整除之外，不会再被另外的数整除的全部正整数，并将结果依次显示。

<1> 在命令行提示符“:”后面输入程序：

```
:new
:10 dim a(100)
:20 for i=2 to 100
:30 j=i
:40 j=j+1
:50 if i*j<100 then a(i*j)=1:goto 40
:60 if i*i<100 then a(i*i)=1
:70 next i
:80 for i=2 to 99
:90 if a(i)=0 then print i
:100 next i
:110 end
```

<2> 输入 run 命令运行程序，检测出来的素数显示在屏幕上。



```
终端1 80H 81H
:run
2
3
5
7
11
13
17
19
23
29
31
37
41
43
```

求素数

例 3. 汉诺塔

把从大到小自底向上叠起来的几个盘子，从源位置 s 移动到目的位置 d，条件是必须保证在任何时刻不得出现大盘子压在小盘子上面的情形，在移动的过程中，还会用到另外一个缓冲位置 t，以便临时存放中间结果。

<1> 在命令行提示符“:”后面输入程序：

```
:new
:10 dim s(10),d(10),t(10) （注意：这里的 10 可以更大，数越大运行时可设置更多盘子）
:20 input n
:30 i=n:s(n)=1:d(n)=2:t(n)=3
:40 i=i-1:s(i)=s(i+1):d(i)=t(i+1):t(i)=d(i+1)
:50 if i>0 then 40
:60 i=i+1:if i>n then end
```



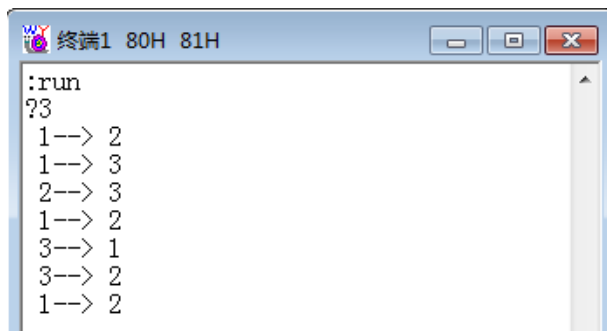
```

:70 if s(i-1)<>s(i) then 60
:80 print s(i);"- ->";d(i)
:90 i=i-1:s(i)=t(i+1):d(i)=d(i+1):t(i)=s(i+1)
:100 if i>0 then 40
:110 goto 60
:120 end

```

<2> 输入 run 命令运行程序，设置盘子个数，结果显示到屏幕上，运行时间根据设置的盘子数成指数增长。

显示的结果中，数字“1”表示源位置 s，“2”表示目的位置 d，“3”表示辅助位置 t。



```

:run
?3
1--> 2
1--> 3
2--> 3
1--> 2
3--> 1
3--> 2
1--> 2

```

汉诺塔

例 4. 8 皇后问题

在 8 行*8 列的棋盘上，以相互不能“吃子”的方式放进 8 个皇后棋子，即在任何一个横排、任何一个竖列、任何一个对角线的方向上，都不得同时出现两个皇后棋子，把全部可行结果排列出来，显示在屏幕上。

<1> 在命令行提示符“:”后面输入程序：

```

:new
:10 dim c(7),f(14),b(14),q(7)
:20 i=0:count=0
:30 q(i)=0
:40 if c(q(i))+f(i-q(i)+7)+b(i+q(i))>0 then 170
:50 c(q(i))=1:f(i-q(i)+7)=1:b(i+q(i))=1
:60 if i<7 then i=i+1:goto 30
:70 count=count+1
:80 print:print"Result:";count;":"
:90 j=0
:100 k=0
:110 if k=q(j) then print"";:goto 130
:120 print"0";
:130 k=k+1:if k<8 then 110
:140 print

```

```

:150 j=j+1:if j<8 then 100
:160 c(q(i))=0:f(i-q(i)+7)=0:b(i+q(i))=0
:170 q(i)=q(i)+1:if q(i)<8 then 40
:180 i=i-1:if i>=0 then 160
:190 print:print"Total results:";count
:200 end

```

<2> 输入 run 命令运行程序，运行时间会稍微长一些，结果显示到屏幕上，其中的“*”表示皇后位置，部分结果如下：

```

:run

Result: 1:
*0000000
0000*000
0000000*
00000*00
00*00000
000000*0
0*000000
0000000*
000*0000

Result: 2:
*0000000
00000*00
0000000*
00*00000
000000*0
000*0000
0*000000
0000*000
00000000

Result: 3:
*0000000
000000*0
000*0000
00000*00
0000000*
0*000000
0000*000
00000000
00*00000

Result: 35:
000*0000
0*000000
0000000*
0000*000
0000000*
000000*0
*0000000
00*00000
00000*00

Result: 36:
000*0000
0*000000
0000000*
00000*00
0000000*
0*000000
00*00000
0000*000
000000*0

Result: 37:
000*0000
00000*00
*0000000
000*0000
0*000000
0000000*
00*00000
00000*00
000000*0

Result: 38:
000*0000
00000*00
*0000000
000*0000
0*000000
0000000*
00*00000
0000*000
000000*0

Result: 60:
0000*000
000000*0
0*000000
00000*00
00*00000
*0000000
000*0000
0000000*
0000000*

Result: 61:
0000*000
000000*0
0*000000
00000*00
00*00000
*0000000
000*0000
0000000*
000*0000

Result: 62:
0000*000
000000*0
000*0000
*0000000
00*00000
0000000*
00000*00
0000000*
0*000000

Result: 63:
0000*000
000000*0
000*0000
*0000000
00*00000
0000000*
00000*00
0000000*
0*000000

Result: 90:
0000000*
0*000000
0000*000
00*00000
*0000000
000000*0
000*0000
000000*0
00000*00

Result: 91:
0000000*
00*00000
*0000000
00000*00
0*000000
0000*000
000000*0
000*0000
00000000

Result: 92:
0000000*
000*0000
*0000000
00*00000
00000*00
0*000000
000000*0
00000*00
00000000

Total
results: 92

```

8 皇后结果

例 5. 验证歌德巴赫猜想

在数值 100 范围内验证歌德巴赫猜想，即任何一个大于 2 的偶数都等于另外两个素数

之和，验证的结果显示在屏幕上。

<1> 在命令行提示符“:”后面输入程序：

```
:new  
:10 for i=4 to 100 step 2  
:20 for j=1 to i/2  
:30 n=j  
:40 gosub 200  
:50 if p=0 then 100  
:60 n=i-j  
:70 gosub 200  
:80 if p=0 then 100  
:90 print i;"=";j;"+";n,:goto 100  
:100 next j  
:110 next i  
:200 p=0  
:210 if n/2<2 then 250  
:220 for k=2 to n/2  
:230 if n mod k = 0 then 260  
:240 next k  
:250 p=1  
:260 return
```

<2> 输入 run 命令运行程序，结果显示到屏幕上。

```

终端1 80H 81H
:run
4= 1+ 3      4= 2+ 2      6= 1+ 5      6= 3+ 3      8= 1+ 7      8= 3+ 5
10= 3+ 7     10= 5+ 5     12= 1+ 11     12= 5+ 7     14= 1+ 13     14= 3+ 11
14= 7+ 7     16= 3+ 13     16= 5+ 11     18= 1+ 17     18= 5+ 13     18= 7+ 11
20= 1+ 19     20= 3+ 17     20= 7+ 13     22= 3+ 19     22= 5+ 17     22= 11+ 11
24= 1+ 23     24= 5+ 19     24= 7+ 17     24= 11+ 13    26= 3+ 23     26= 7+ 19
26= 13+ 13    28= 5+ 23     28= 11+ 17    30= 1+ 29     30= 7+ 23     30= 11+ 19
30= 13+ 17    32= 1+ 31     32= 3+ 29     32= 13+ 19    34= 3+ 31     34= 5+ 29
34= 11+ 23    34= 17+ 17    36= 5+ 31     36= 7+ 29     36= 13+ 23    36= 17+ 19
38= 1+ 37     38= 7+ 31     38= 19+ 19    40= 3+ 37     40= 11+ 29    40= 17+ 23
42= 1+ 41     42= 5+ 37     42= 11+ 31    42= 13+ 29    44= 1+ 43     44= 17+ 29
44= 3+ 41     44= 7+ 37     44= 13+ 31    46= 3+ 43     46= 5+ 41     46= 17+ 29
46= 23+ 23    48= 1+ 47     48= 5+ 43     48= 7+ 41     48= 11+ 37    48= 17+ 31
48= 19+ 29    50= 3+ 47     50= 7+ 43     50= 13+ 37    50= 19+ 31    52= 5+ 47
52= 11+ 41    52= 23+ 29    54= 1+ 53     54= 7+ 47     54= 11+ 43    54= 13+ 41
54= 17+ 37    54= 23+ 31    56= 3+ 53     56= 13+ 43    56= 19+ 37    58= 5+ 53
58= 11+ 47    58= 17+ 41    58= 29+ 29    60= 1+ 59     60= 7+ 53     60= 13+ 47
60= 17+ 43    60= 19+ 41    60= 23+ 37    60= 29+ 31    62= 1+ 61     62= 3+ 59
62= 19+ 43    62= 31+ 31    64= 3+ 61     64= 5+ 59     64= 11+ 53    64= 17+ 47
64= 23+ 41    66= 5+ 61     66= 7+ 59     66= 13+ 53    66= 19+ 47    66= 23+ 43
66= 29+ 37    68= 1+ 67     68= 7+ 61     68= 31+ 37    70= 3+ 67     70= 11+ 59
70= 17+ 53    70= 23+ 47    70= 29+ 41    72= 1+ 71     72= 5+ 67     72= 11+ 61
72= 13+ 59    72= 19+ 53    72= 29+ 43    72= 31+ 41    74= 1+ 73     74= 3+ 71
74= 7+ 67     74= 13+ 61    74= 31+ 43    74= 37+ 37    76= 3+ 73     76= 5+ 71
76= 17+ 59    76= 23+ 53    76= 29+ 47    78= 5+ 73     78= 7+ 71     78= 11+ 67
78= 17+ 61    78= 19+ 59    78= 31+ 47    78= 37+ 41    80= 1+ 79     80= 7+ 73
80= 13+ 67    80= 19+ 61    80= 37+ 43    82= 3+ 79     82= 11+ 71    82= 23+ 59
82= 29+ 53    82= 41+ 41    84= 1+ 83     84= 5+ 79     84= 11+ 73    84= 13+ 71
84= 17+ 67    84= 23+ 61    84= 31+ 53    84= 37+ 47    84= 41+ 43    86= 3+ 83
86= 7+ 79     86= 13+ 73    86= 19+ 67    86= 43+ 43    88= 5+ 83     88= 17+ 71
88= 29+ 59    88= 41+ 47    90= 1+ 89     90= 7+ 83     90= 11+ 79    90= 17+ 73
90= 19+ 71    90= 23+ 67    90= 29+ 61    90= 31+ 59    90= 37+ 53    90= 43+ 47
92= 3+ 89     92= 13+ 79    92= 19+ 73    92= 31+ 61    94= 5+ 89     94= 11+ 83
94= 23+ 71    94= 41+ 53    94= 47+ 47    96= 7+ 89     96= 13+ 83    96= 17+ 79
96= 23+ 73    96= 29+ 67    96= 37+ 59    96= 43+ 53    98= 1+ 97     98= 19+ 79
98= 31+ 67    98= 37+ 61    100= 3+ 97    100= 11+ 89   100= 17+ 83   100= 29+ 71
100= 41+ 59   100= 47+ 53   RETURN without GOSUB

```

哥德巴赫猜想

例 6. 绘制三角（正弦）函数图形

计算正弦三角函数的程序，将 0~360 度范围内的正弦曲线显示在计算机屏幕上。

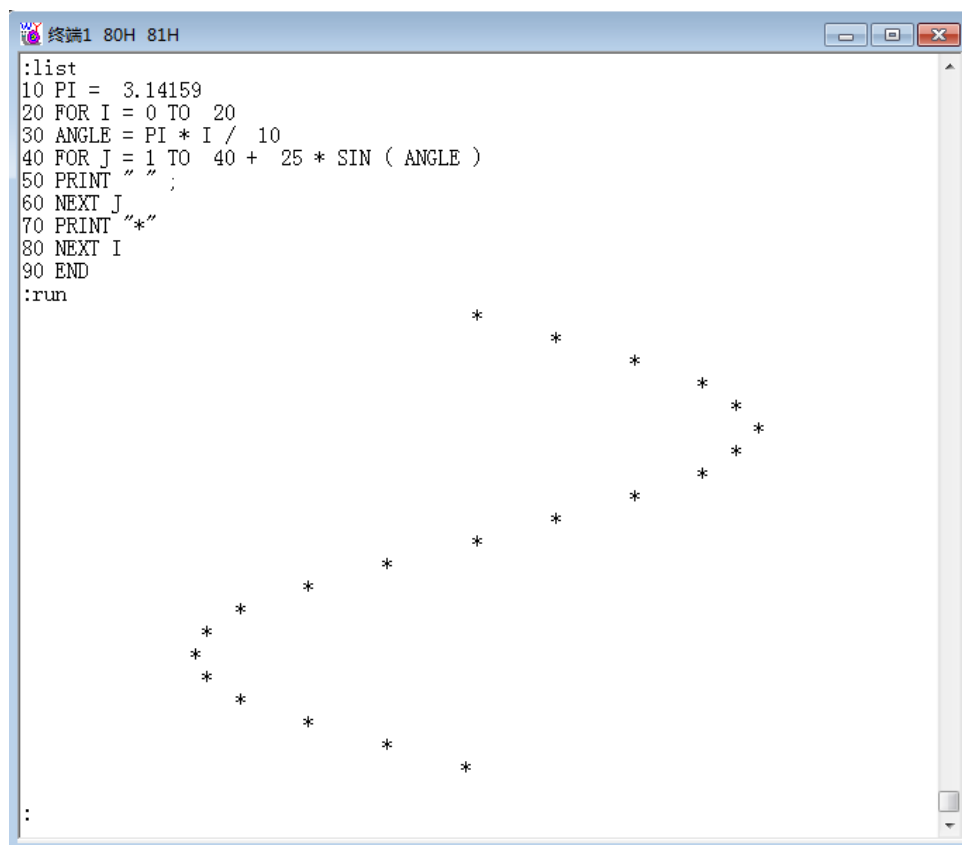
<1> 在命令行提示符“:”后面输入程序：

```

:new
:10 pi=3.14159
:20 for i=0 to 20
:30 angle=pi*i/10
:40 for j=1 to 40+25*sin(angle)
:50 print " ";
:60 next j
:70 print "*"
:80 next i
:90 end

```

<2> 输入 run 命令运行程序，结果显示到屏幕上。



绘制正弦曲线