

目 录

第 1 章 JUC2 十六位微程序控制计算机概述.....	1
1.1 JUC2 微程序控制计算机硬件系统的数据通路.....	1
1.2 指令系统.....	2
1.3 微程序控制器.....	6
1.4 微程序设计.....	8
1.5 输入输出和中断系统	16
第 2 章 十六位微程序控制计算机的设计.....	21
2.1 熟悉微程序的设计和调试方法.....	21
2.2 双操作数指令的设计与调试.....	23
2.3 条件转移指令的设计与调试.....	24
2.4 移位指令的设计与调试.....	27
2.5 堆栈相关指令的设计与调试.....	30
2.6 中断系统的设计与调试.....	30
2.7 答辩及验收.....	32

图表目录

表 1-1 模型机地址空间分配	2
表 1-2 寻址方式及编码表	3
表 1-3 指令编码表	5
表 1-4 JUC2 模型机微指令格式	7
表 1-5 JUC2 模型机微转移方式字段 BM	7
表 1-6 控存地址分配	8
表 1-7 取指令微程序的微命令编码	10
表 1-8 取指令微程序	11
表 1-9 MOV 指令的微程序	16
表 1-10 LED 输出接口地址分配表	16
表 1-11 开关输入接口地址分配表	17
表 1-12 中断向量地址	18
图 1-1 JUC2 模型计算机硬件系统的数据通路	1
图 1-2 指令格式	2
图 1-3 微程序控制器的基本组成	6
图 1-4 微程序控制方式的时序	6
图 1-5 指令执行过程	9
图 1-6 取指令微流程	10
图 1-7 微指令图例	11
图 1-8 以流程图表示的取指令微程序	12
图 1-9 以流程图表示的取源操作数微程序	13
图 1-10 以流程图表示的取目的操作数微程序	14
图 1-11 指令执行阶段微程序入口地址的形成	15
图 1-12 单操作数运算指令的执行阶段微程序	16
图 1-13 输入接口逻辑框图	17
图 1-14 中断控制器组成框图	18
图 2-1 转移指令微流程和微程序	21
图 2-2 BM3 微地址修改逻辑	22
图 2-3 移位操作	23
图 2-4 左移操作实现原理	24
图 2-5 左移移入位 lsb 选择逻辑	24
图 2-6 右移操作实现原理	24
图 2-7 右移移入位 hsb 选择逻辑	25
图 2-8 CF 选择逻辑	25

第 1 章 JUC2 十六位微程序控制计算机概述

1.1 JUC2 微程序控制计算机硬件系统的数据通路

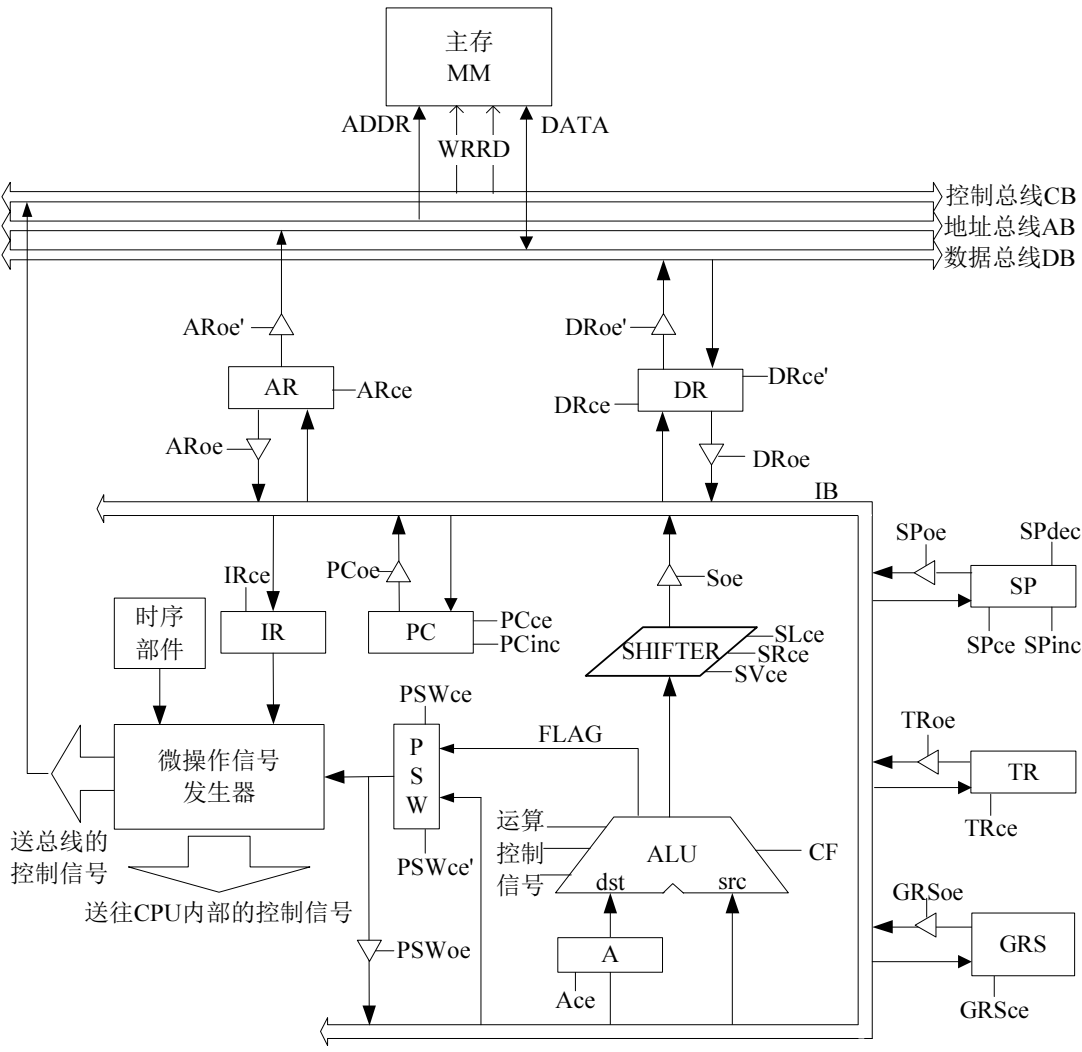


图 1-1 JUC2 模型计算机硬件系统的数据通路

JUC2 模型计算机硬件系统的数据通路如图 1-1。CPU 的字长为 16 位，包括运算器和控制器两个部分，各个部件通过 16 位内部总线 IB 相连；图中还包括了系统总线和存储器，系统总线采用单总线结构，包括 16 位的数据总线 DB、16 位的地址总线 AB 和控制总线 CB。主存、外设通过系统总线与 CPU 连接；CPU 内部总线 IB 与系统总线之间通过 DR、

AR 相联。主存储器的字长也是 16 位，并且按字编址，不能按字节访问。

复位时，PC 的初始值为 0030H，即主程序的第一条指令放在 0030H 单元。SP 的初始值也为 0030H，堆栈的存储空间为 002FH~0008H。中断向量表的入口地址为 0000H，一共预留了 8 个中断向量的存储空间；IO 接口与主存统一编址，占用 FF00H~FFFFH 的地址空间。整个 64K 地址空间分配如下：

表 1-1 模型机地址空间分配

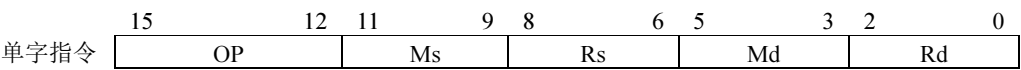
0000H~0007H	中断向量表
0008H~002FH	堆栈
0030H~FEFFH	程序及数据
FF00H~FFFFH	IO 接口

1.2 指令系统

JUC2 模型机的指令系统包括各类传送类指令、算术逻辑运算类指令、移位类指令、转移类指令、子程序调用返回指令、输入输出类指令等。在寻址方式上采用最典型的寻址方式，分别是立即寻址、直接寻址、间接寻址、寄存器寻址、寄存器间接寻址、寄存器自增间接寻址、寄存器变址寻址、相对寻址 8 种。

1.2.1 指令格式

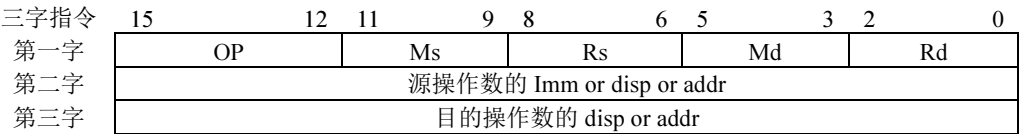
JUC2 模型机指令格式规整，以单字指令为基础，根据不同的寻址方式可扩展为双字指令和三字指令，如图 1-2 所示。指令的第二字和第三字是一些常数，如立即数、直接地址、间接地址、偏移量等。



(a) 单字指令的指令格式



(b) 双字指令的指令格式



(c) 三字指令的指令格式

图 1-2 指令格式

图 1-2 中，Ms 表示源操作数的寻址方式，Md 表示目的操作数的寻址方式，Rs 和 Rd 分别表示的是源操作数和目的操作数的寄存器号。

1.2.2 寻址方式及编码

在图 1-2中可以看出，寻址方式 Ms、Md 分别由 IR 的 11~9 和 5~3 位表示。各位含义见表 1-2。

表 1-2 寻址方式及编码表

寻址方式	助记符	编码 M
寄存器寻址	Rn	000
寄存器间接寻址	(Rn)	001
寄存器自增间接寻址	(Rn)+	010
立即寻址	#imm	011
直接寻址	addr	100
间接寻址	(addr)	101
变址寻址	disp(Rn)	110
相对寻址	`disp	111

1.2.3 双操作数指令

本模型机设计了 9 条双操作数指令：

MOV，

ADD、ADC，SUB、SUBB，CMP

AND、OR、XOR

指令编码格式如下：

15	12	11	9	8	6	5	3	2	0
OP				Ms		Rs		Md	Rd

OP 表示操作码。

1.2.4 单操作数指令

模型机设计有 20 多条单操作数指令，由于只用到一个操作数，所以基本操作码字段（IR_{15~12}）用全 0 表示扩展，将双操作数指令的源操作数字段（IR_{11~6}）用于表示单操作数指令的操作码，其中用 IR_{11~IR₉}来区别不同的单操作数指令的类型，其指令格式如下。

15	12	11	6	5	3	2	0
0000				OP		Md	Rd

1. 移位类指令

SHL、SHR：逻辑左移、右移

SAR：算术右移

ROL、ROR：循环左移、右移

RCL、RCR：带进位的循环左移、右移

指令编码格式如下：

15	12	11	9	8	7	6	5	3	2	0
----	----	----	---	---	---	---	---	---	---	---

0000	0 0 0	OP	Md	Rd
------	-------	----	----	----

2. 条件转移指令

JC、JNC、JO、JNO、JS、JNS、JZ、JNZ

指令编码格式如下：

15	12	11	9	8	7	6	5	3	2	0
0000		0 0 1		OP			Md		Rd	

3. 单操作数运算指令和无条件转移指令

INC、DEC、NOT、JMP

指令编码：

15	12	11	9	8	7	6	5	3	2	0
0000		0 1 0		OP			Md		Rd	

4. 堆栈指令和子程序调用指令

PUSH、POP、CALL

指令编码：

15	12	11	9	8	7	6	5	3	2	0
0000		0 1 1		OP			Md		Rd	

1.2.5 无操作数指令

模型机设计有 6 条无操作数指令。由于没有操作数，(IR₁₅₋₆)用全 0 表示扩展，(IR₅₋₀)用于表示无操作数指令的操作码，其指令格式如下。

15	6	5	0
0 0 0 0 0 0 0 0 0 0	OP		

1.2.6 指令编码表

表 1-3 指令编码表

指令助记符		指 令 编 码																影响 PSW			
		F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	S	Z	O	C
MOV	src, dst	0	0	0	1	源地址码						目的地址码						—	—	—	—
ADD	src, dst	0	0	1	0	源地址码						目的地址码						√	√	√	√
ADDC	src, dst	0	0	1	1	源地址码						目的地址码						√	√	√	√
SUB	src, dst	0	1	0	0	源地址码						目的地址码						√	√	√	√
SUBB	src, dst	0	1	0	1	源地址码						目的地址码						√	√	√	√
AND	src, dst	0	1	1	0	源地址码						目的地址码						√	√	×	×
OR	src, dst	0	1	1	1	源地址码						目的地址码						√	√	×	×
XOR	src, dst	1	0	0	0	源地址码						目的地址码						√	√	×	×
CMP	src, dst	1	0	0	1	源地址码						目的地址码						√	√	√	√
TEST	src, dst	1	0	1	0	源地址码						目的地址码						√	√	×	×
SAR	dst	0	0	0	0	0	0	0	0	0	1	目的地址码						×	×	×	√
SHL	dst	0	0	0	0	0	0	0	0	1	0	目的地址码						×	×	×	√
SHR	dst	0	0	0	0	0	0	0	0	1	1	目的地址码						×	×	×	√
ROL	dst	0	0	0	0	0	0	0	1	0	0	目的地址码						×	×	×	√
ROR	dst	0	0	0	0	0	0	0	1	0	1	目的地址码						×	×	×	√
RCL	dst	0	0	0	0	0	0	0	1	1	0	目的地址码						×	×	×	√
RCR	dst	0	0	0	0	0	0	0	1	1	1	目的地址码						×	×	×	√
JC	dst	0	0	0	0	0	0	1	0	0	0	目的地址码						—	—	—	—
JNC	dst	0	0	0	0	0	0	1	0	0	1	目的地址码						—	—	—	—
JO	dst	0	0	0	0	0	0	1	0	1	0	目的地址码						—	—	—	—
JNO	dst	0	0	0	0	0	0	1	0	1	1	目的地址码						—	—	—	—
JZ	dst	0	0	0	0	0	0	1	1	0	0	目的地址码						—	—	—	—
JNZ	dst	0	0	0	0	0	0	1	1	0	1	目的地址码						—	—	—	—
JS	dst	0	0	0	0	0	0	1	1	1	0	目的地址码						—	—	—	—
JNS	dst	0	0	0	0	0	0	1	1	1	1	目的地址码						—	—	—	—
JMP	dst	0	0	0	0	0	1	0	0	0	0	目的地址码						—	—	—	—
INC	dst	0	0	0	0	0	1	0	0	0	1	目的地址码						√	√	√	√
DEC	dst	0	0	0	0	0	1	0	0	1	0	目的地址码						√	√	√	√
NOT	dst	0	0	0	0	0	1	0	0	1	1	目的地址码						√	√	×	×
PUSH	dst	0	0	0	0	0	1	1	0	0	0	目的地址码						—	—	—	—
POP	dst	0	0	0	0	0	1	1	0	0	1	目的地址码						—	—	—	—
CALL	dst	0	0	0	0	0	1	1	0	1	0	目的地址码						—	—	—	—
HALT		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—
NOP		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	—	—	—	—	
RET		0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	—	—	—	—	
RETI		0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	—	—	—	—	
EI		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	—	—	—	—	
DI		0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	—	—	—	—	

注：√表示指令设置 PSW 的该标志位；—表示不影响；×表示会影响、但没有意义

1.3 微程序控制器

1.3.1 微程序控制器的基本构成

微程序控制器由五部分组成，基本组成框图如图 1-3。

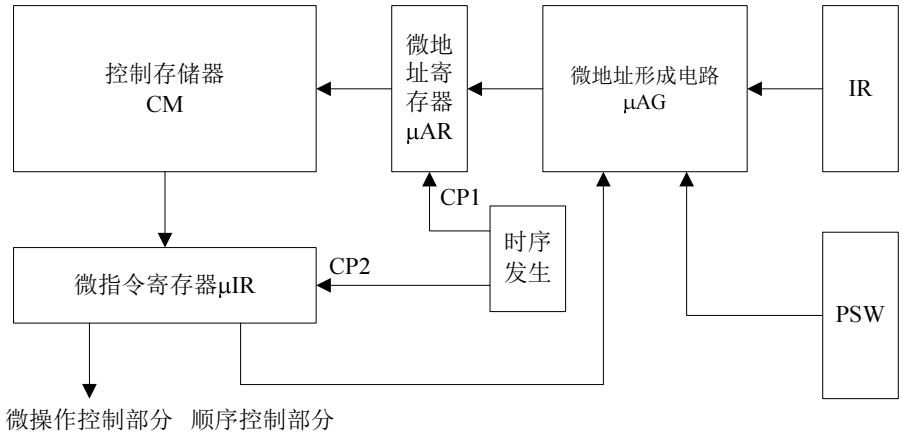


图 1-3 微程序控制器的基本组成

- (1) 控制存储器 CM，存放微程序。
- (2) 微地址寄存器 μAR ，存放 CM 地址。
- (3) 微指令寄存器 μIR ，存放由 CM 中取出的微指令。
- (4) 微地址形成线路 μAG ，形成微地址，送给 μAR 。

该电路有三个输入，除了 μIR 的顺序控制部分之外，还有 IR 和 PSW。IR 主要用于产生微程序的入口地址，比如依据指令的操作码形成对应各指令执行阶段的微程序入口地址。PSW 中的状态标志，在某些需要判定是否符合条件的场合，决定分支转移的微地址。

- (5) 时序部件，产生微程序控制器的时钟信号。

微程序控制器的基本时序单位是微周期，微周期是一条微指令执行所需的时间，一条微指令的执行时间包括两部分：一部分是从 CM 中读取微指令所需要的时间，这个时间便是 ROM 的读出时间，另一部分是微指令执行所需要的时间，这个时间包括微命令的译码时间 CPU 内部数据通路的传输时间。

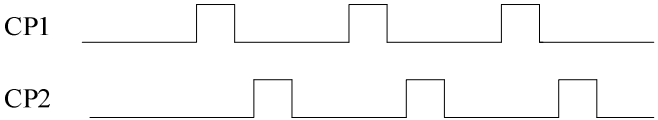


图 1-4 微程序控制方式的时序

本设计中微程序的时序由 CP1 和 CP2 两个等周期信号组成，如图 1-4。CP1 信号上升沿的作用是将微地址打入控存微地址寄存器，启动一次读操作。CP2 的上升沿的作用是将

从 CM 中读取的微指令打入微指令寄存器，这标志着取微指令的结束和执行微指令的开始。CP1 同时作为 CPU 内部寄存器的打入脉冲。显然，CP1 的上升沿到 CP2 的上升沿为取微指令时间，而从 CP2 的上升沿至下一个 CP1 的上升沿为执行微指令时间。

1.3.2 微指令格式设计

在本设计中，微指令的编码方式采用字段直接编码方式。微指令格式如表 1-4 所示，其中下址字段占 9 位，微转移方式字段占 3 位，微转移方式见表 1-5，微指令的总宽度为 32 位。

表 1-4 JUC2 模型机微指令格式

F0:XXoe (3 位)	F1:XXce (3 位)	F2:ALU (4 位)	F3:Shifter (2 位)	F4:AR (2 位)	F5:DR (2 位)	F6:PC (1 位)	F7:MEM (3 位)	F8:BM (3 位)	F9: NA (9 位)
0:NOP 1:PCoe 2:GRSoe 3:Soe 4:TRoe 5:ARoe 6:DRoe 7:SPoe	0:NOP 1:PCce 2:GRSce 3:IRce 4:TRce 5:Ace 6:PSWce 7:SPce	0:NOP 1:ADD 2:ADDC 3:SUB 4:SUBB 5:AND 6:OR 7:NOT 8:XOR 9:INC A:DEC B:SPinc C:SPdec	0:NOP 1:SRce 2:SLce 3:SVce	0:NOP 1:ARoe' 2:ARce	0:NOP 1:DRoe' 2:DRce' 3:DRce	0:NOP 1:PCinc	0:NOP 1:RD 2:WR 3:PSWoe 4:PSWce' 5:STI 6:CLI 7:INTA	表 1-5	

表 1-5 JUC2 模型机微转移方式字段 BM

BM \leftarrow	操作 \leftarrow	意义 \leftarrow
0 \leftarrow	$\mu AR = NA \leftarrow$	固定转移 \leftarrow
1 \leftarrow	$\mu AR_{8,6-0} = NA_{8,6-0}, \mu AR_7 = INTR \cdot IE \leftarrow$	根据是否有中断请求且是否允许中断产生两分支 \leftarrow
2 \leftarrow	$\mu AR_{8-2} = NA_{8-2}, \leftarrow$ $\mu AR_1 = \overline{IR_{15} + IR_{14} + IR_{13} + IR_{12}} \leftarrow$ $\mu AR_0 = \overline{IR_{11} + IR_{10} + IR_9 + IR_8 + IR_7 + IR_6} \leftarrow$	依据操作数类型的三分支微转移。如果是双操作数指令，则 $\mu AR_1=0$ ；如果是单操作数指令，则 $\mu AR_1=1$ 、 $\mu AR_0=0$ ；如果是无操作数指令，则 $\mu AR_1=1$ 、 $\mu AR_0=1$ 。 \leftarrow
3 \leftarrow	$\mu AR_{8-1} = NA_{8-1}, \leftarrow$ $\mu AR_0 = f_{\{OP, PSW(Z, O, S, C)\}} \leftarrow$	根据条件转移指令操作码和 PSW 的 ZF、OF、SF、CF 状态标志决定微地址，若满足条件 $\mu AR_0=1$ ，否则 $\mu AR_0=0$ 。 \leftarrow
4 \leftarrow	按操作码 OP 多路转移 \leftarrow	按操作码 OP 形成多路微转移地址 \leftarrow
5 \leftarrow	$\mu AR_{8-3} = NA_{8-3}, \mu AR_{2-0} = M \leftarrow$	按寻址方式 M 形成多路微转移地址 \leftarrow
6 \leftarrow	保留 \leftarrow	\leftarrow
7 \leftarrow	$\mu AR_{8-1} = NA_{8-1}, \leftarrow$ $\mu AR_0 = IR_5 + IR_4 + IR_3 \leftarrow$	根据目的操作数寻址方式产生两分支：若 Md=000（寄存器寻址），则 $\mu AR_0=0$ ；否则 $\mu AR_0=1$ 。 \leftarrow

1.3.3 控存地址分配

复位时， μAR 的初始值为 000H。控存地址分配如表 1-6。

表 1-6 控存地址分配

微程序	微地址	微地址形成方法
取指令	000H~003H	复位 或 BM=1
取源操作数的入口	004H	BM=2
	005H	
	006H	
取目标操作数的入口	007H	
执行阶段的入口	007H	
取源操作数微程序	008H~020H	BM=5
（可用）	021H~027H	
取目标操作数微程序	028H~03FH	BM=5
（可用）	040H	
双操作数指令的入口（最多 15 条，实际 10 条）	041H~04FH	{00100, IR[15:12]} $\rightarrow \mu\text{AR}$
执行结果存入目的操作数的微程序	050H~052H	BM=7
（可用）	053H~057H	
无操作数指令的入口（最多 8 条，实际 6 条）	058H~05FH	{001011, IR[2:0]} $\rightarrow \mu\text{AR}$
单操作数指令的入口（最多 31 条，实际 22 条）	061H~07FH	{0011, IR[10:6]} $\rightarrow \mu\text{AR}$
中断响应隐指令的入口	080H	BM=1
（可用）	081H~1FFH	

1.4 微程序设计

1.4.1 微程序的设计方法

在微程序控制计算机的指令系统、数据通路及微指令的格式设计完成后，为了实现 CPU 指令系统的功能，必须编写每一条指令的微程序。由于指令的运行包括取指令、取操作数和执行三个阶段，因此，写微程序也包括三个方面，即取指令微程序、取操作数微程序和指令执行的微程序。

微程序设计的一般方法如下：

- （1）画出微流程。根据数据通路（见图 1-1）画出取指令微流程、取操作数微流程；根据每条指令的功能画出指令的执行微流程。
- （2）将微流程翻译成微命令编码。按微指令格式及微命令的编码（见表 1-4），根据微流程的每一步填写微指令 F0—F7 字段的内容。
- （3）分配微地址。根据微流程的每一步之间是否有转移以及何种转移，参照表 1-5 填写微指令 F8 字段（转移方式 BM）和 F9 字段（下地址 NA），确定每条微指令的地址。

1.4.2 指令执行过程

一条指令的执行包括四个阶段，即取指令阶段 IF、取源操作数阶段 SOF、取目的操作

数阶段 DOF 和执行阶段 EXE。JUC2 模型机取指令微程序的入口地址为 000H，机器复位时微地址寄存器 uAR 初始化为 0，即取指令的入口。取指令结束后，根据指令的类型三分支转移，如是双操作数指令，则先取源操作数，再取目的操作数，之后进入执行阶段；如是单操作数指令，直接取目的操作数，之后进入执行阶段；如是无操作数指令，则直接进入执行阶段。执行结束后，判断是否有中断请求，有中断请求且允许中断时执行中断隐指令，否则返回到取指令阶段，取下一条指令，如图 1-5。

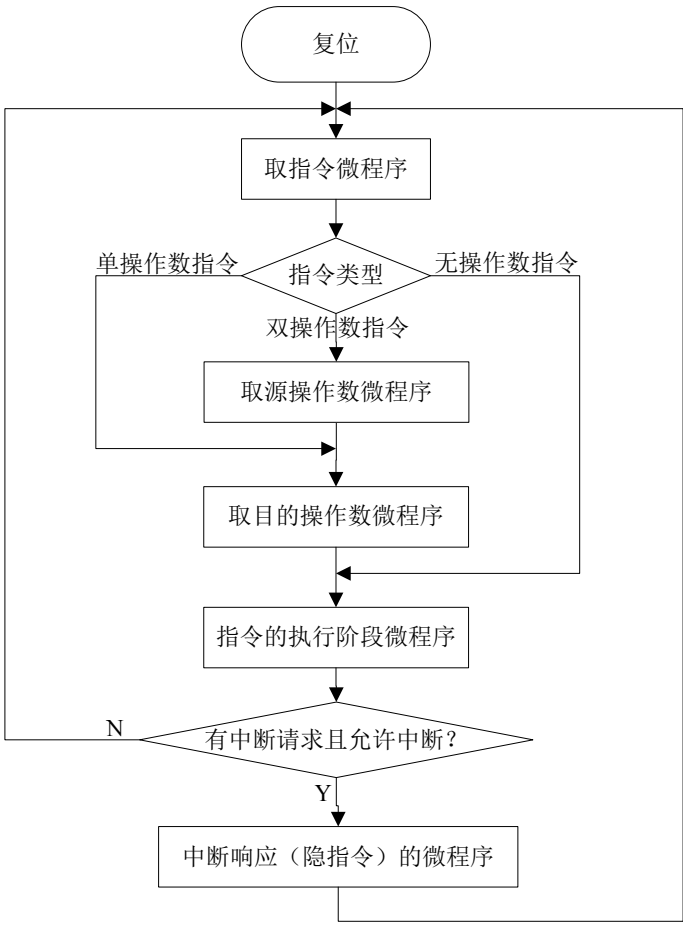


图 1-5 指令执行过程

1.4.3 取指令的微程序设计

按照1.4.1节介绍的微程序设计一般方法，分为三个步骤。

(1) 画出取指令微流程

取指令的目标是将存储器中的指令取到指令寄存器 IR 中。首先 PC 的内容送给 AR；然后 AR 的内容送到地址总线，给出读信号，将读出的指令送给 DR，同时 PC 加 1；最后 DR 的内容即指令送给 IR，完成取指令。取指阶段微流程如图 1-6。

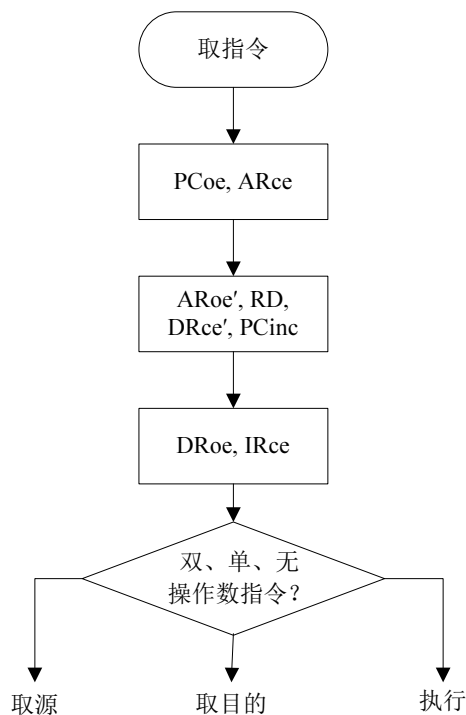


图 1-6 取指令微流程

(2) 将微流程翻译成微命令编码

参照微指令编码表 1-4，根据图 1-6填写微指令 F0~F7 字段，无操作字段填 ‘0’，见表 1-7。

表 1-7 取指令微程序的微命令编码

微地址(H)	微指令(H)	微指令字段(H)										微命令
		F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	
		1	0	0	0	2	0	0	0			PCoe, ARce
		0	0	0	0	1	2	1	1			ARoe', RD, DRce', PCinc
		6	3	0	0	0	0	0	0			DRoe, IRce

(3) 分配微地址

表 1-7中每条微指令的地址尚未确定，接下来就要根据微流程及控存的使用情况填写微指令的 F8、F9 字段。

JUC2 模型机取指令的微程序入口地址为 000H，在 CPU 复位时 $\mu AR=000H$ ，所以第一条微指令地址是 000H。从图 1-6可以看出，取指令的过程中微程序没有分支，因此接下来的几条微指令地址可以依次为 001H、002H、003H，采用固定转移 BM=0（见表 1-5）即 F8=0，转向下一地址，下地址 001H、002H、003H 由 F9 字段指定。

指令取到 IR 以后，根据指令包含操作数的个数要进行三支转移。根据 IR 的内容即可获知操作数的个数，若 $IR_{15} \sim IR_6=0$ ，表明这是一个无操作数的指令，直接进入执行阶段

EXE; 若 $IR_{11} \sim IR_6 \neq 0$ 且 $IR_{15} \sim IR_{12} = 0$, 则是单操作数指令, 应当进入取目的操作数阶段 DOF; 若 $IR_{15} \sim IR_{12} \neq 0$, 是双操作数指令进入取源操作数阶段 SOF。查表 1-5 可知 $BM=2$ 用于完成依据操作数个数的多分支转移, 由硬件实现如下逻辑操作:

$$NA \rightarrow \mu AR,$$

$$\overline{IR_{15}+IR_{14}+IR_{13}+IR_{12}} \rightarrow \mu AR_1$$

$$\overline{IR_{11}+IR_{10}+IR_9+IR_8+IR_7+IR_6} \rightarrow \mu AR_0$$

根据表 1-6 分配的控存地址空间, 设置 $NA=004$, 可依次求得取源、取目和执行的入口分别是: 004 或 005, 006, 007。

最后将微指令以 16 进制表示, 完整的微程序如表 1-8。

表 1-8 取指令微程序

微地址(H)	微指令(H)	微指令字段(H)										微命令
		F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	
000	20080001	1	0	0	0	2	0	0	0	0	001	PCoe, ARce
001	00069002	0	0	0	0	1	2	1	1	0	002	ARoe', RD, DRce', PCinc
002	CC000003	6	3	0	0	0	0	0	0	0	003	DRoe, IRce
003	00000404	0	0	0	0	0	0	0	0	2	004	BM2

表中的第一列对应每条微指令在控存中的微地址; 第二列为每条微指令的 10 个字段拼接后形成的 16 进制编码; 微指令字段所包含的 10 列为每条微指令 10 个字段的微命令编码。‘0’ 代表空操作; 最后一列对应每条微指令中有效的微命令。

对于像取指令这样以顺序执行为主的微程序, 采用表格的形式比较简洁; 但是在分支较多、转移频繁时, 用表格不容易看清微指令的执行顺序。本书设计一种图符的形式表达微程序, 能够清晰地反映微程序的执行过程, 如在图 1-7, 每条微指令用一个块 (block) 表示, 块的结构如图 1-7, 块的左上角是以十六进制表示的该微指令的微地址, 右上角是微指令代码, 中部是该微指令包含的微命令, 右下角是下址字段的微地址 (十六进制), 左下角是微转移方式字段的值及简要说明。

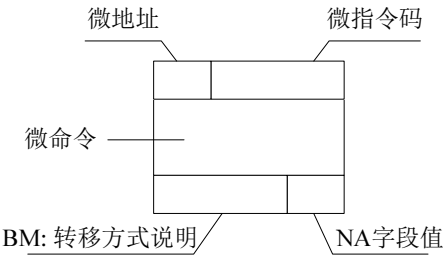


图 1-7 微指令图例

以流程图表示的取指令微程序如图 1-8。

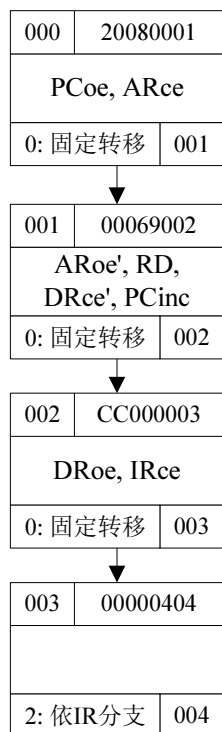


图 1-8 以流程图表示的取指令微程序

1.4.4 取源操作数的微程序设计

如果是双操作数指令，取指令结束后进入取源操作数阶段。取源操作数的微流程如图 1-9，入口地址为 004H 或 005H。

在微地址为 004 和 005 的微指令中，根据不同的寻址方式 M 多路转移，见表 1-5，该转移方式由 BM=5 控制，此时微地址的形成逻辑为：

$$NA \rightarrow \mu AR, M \rightarrow \mu AR_{2-0}$$

如果设置 NA=008H，可求得源操作数为寄存器寻址、寄存器间接、寄存器自增间接、立即、直接、间接、变址和相对寻址的微程序入口地址分别为 008H、009H、00AH、00BH、00CH、00DH、00EH 和 00FH。取完源操作数后固定转移到取目的操作数的入口 006H。其它微指令的微地址理论上可任意安排，但为了合理利用控存空间，事先已经安排好。图 1-10 中大部分微指令是空白，留待同学们完成。

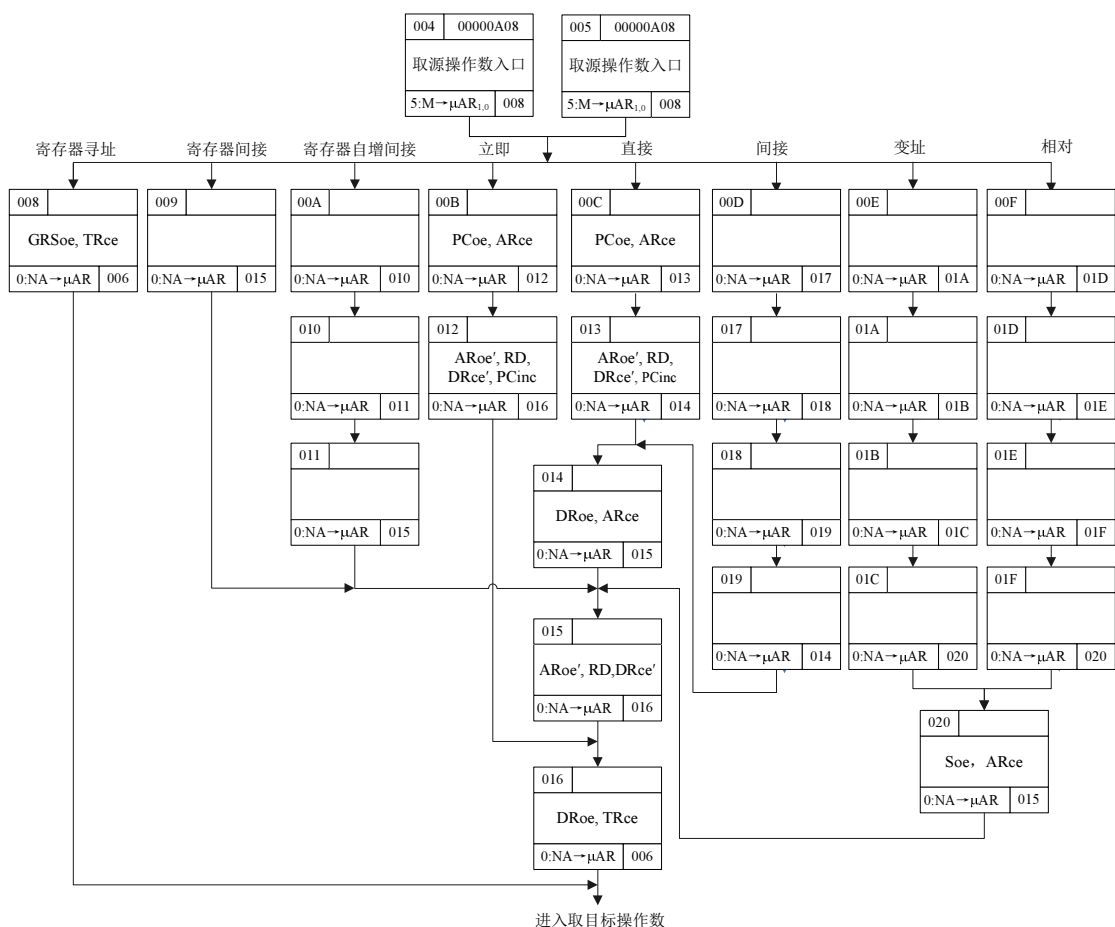


图 1-9 以流程图表示的取源操作数微程序

1.4.5 取目的操作数的微程序设计

取目的操作数微流程与取源操作数微流程基本一致，只是每条微指令的下址字段 NA 设置有所不同，并且取出的目的操作数存放在 A 中，取完目的操作数后微地址固定转移到执行阶段的入口 007。

图 1-10是以流程图表示的取目的操作数微程序，其入口地址为 006，取到的目的操作数放在 A 寄存器中。由 BM=5，NA=028 控制转向目的操作数为寄存器寻址、寄存器间接、寄存器自增间接、直接、间接、变址和相对寻址的微程序入口地址分别为 028、029、02A、02C、02D、02E 和 02F。立即寻址不应作为目的操作数的寻址方式，所以 02B 微指令什么都不做，仅仅是转到执行阶段的入口。

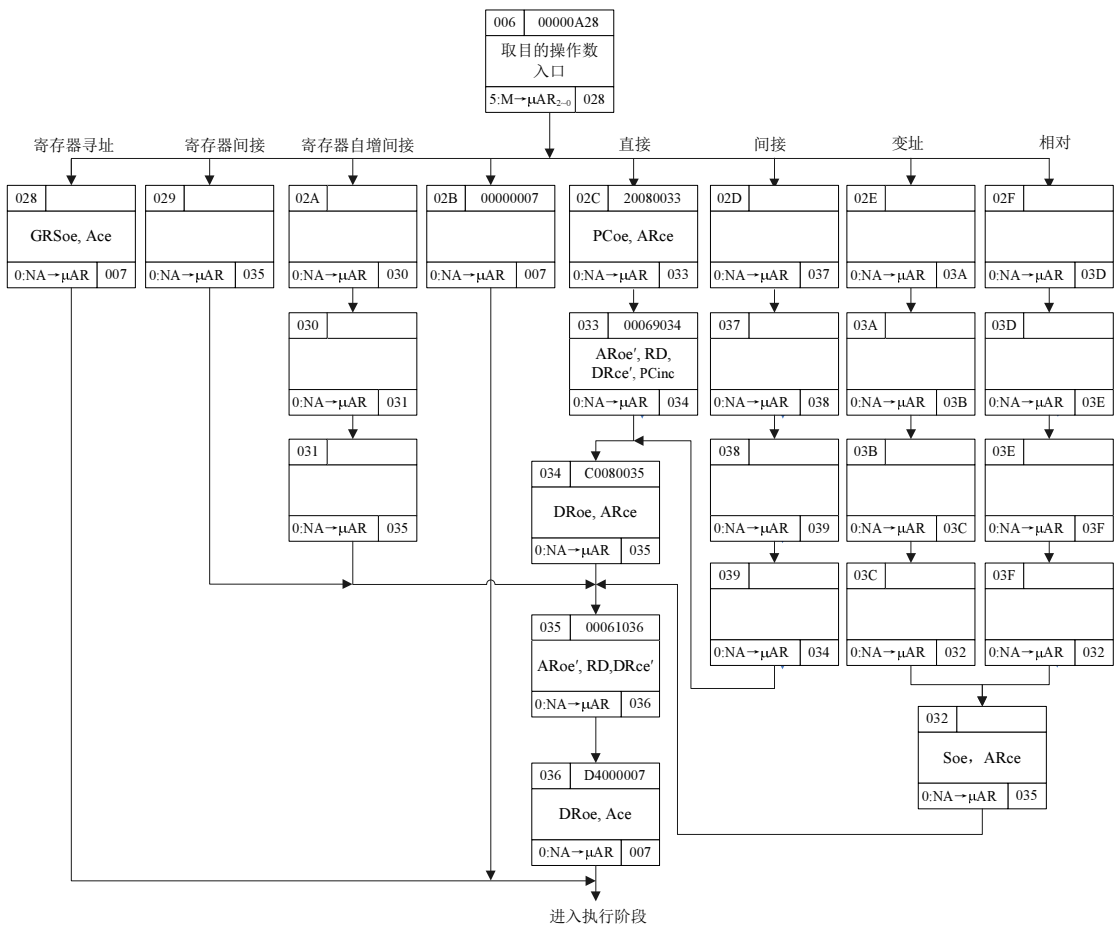


图 1-10 以流程图表示的取目的操作数微程序

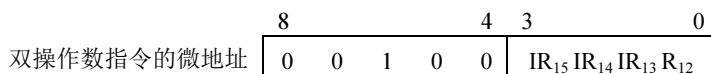
取出的目的操作数存放在 A 中，取目的操作数结束后转移到执行指令 EXE 的入口 007。

1.4.6 执行阶段的微程序设计

1. 执行阶段的微程序入口地址的形成

在进入执行阶段时，应根据机器指令的操作码产生该指令的微程序入口地址。通常一条机器指令对应一段微程序，所以转移的分支将非常多，称为宽转移。这条宽转移的微指令被安排在 007H 微地址，是所有指令执行阶段的总入口；微转移方式 BM 编码为 4。

依据指令操作码生成微转移地址，可以直接用操作码作为微地址的低位；微地址的高位是预先设定的常数，决定了微程序的地址范围。根据表 1-6 分配的微地址范围，双操作数指令、单操作数指令、无操作数指令微程序入口地址形成方法如图 1-11。



	8	5	4	0					
单操作数指令的微地址	0	0	1	1	IR ₁₀	IR ₉	IR ₈	IR ₇	IR ₆
	8	3	2	0					
无操作数指令的微地址	0	0	1	0	1	1	IR ₂	IR ₁	IR ₀

图 1-11 指令执行阶段微程序入口地址的形成

2. 单操作数运算的微程序设计

JUC2 模型机的单操作数运算指令有 3 条：INC、DEC、NOT，从图 1-11 可知它们的微程序入口分别设计在 071H、072H、073H。单操作数指令进入执行阶段时，目的操作数已经在 A 暂存器中，执行阶段只要控制运算器执行相应的运算功能，最后再将运算结果保存到目的操作数。由于很多指令都需要保存运算结果，可以将其设计为公用的微程序，表 1-6 分配给它的微地址范围是 050H~052H。微程序见图 1-12。

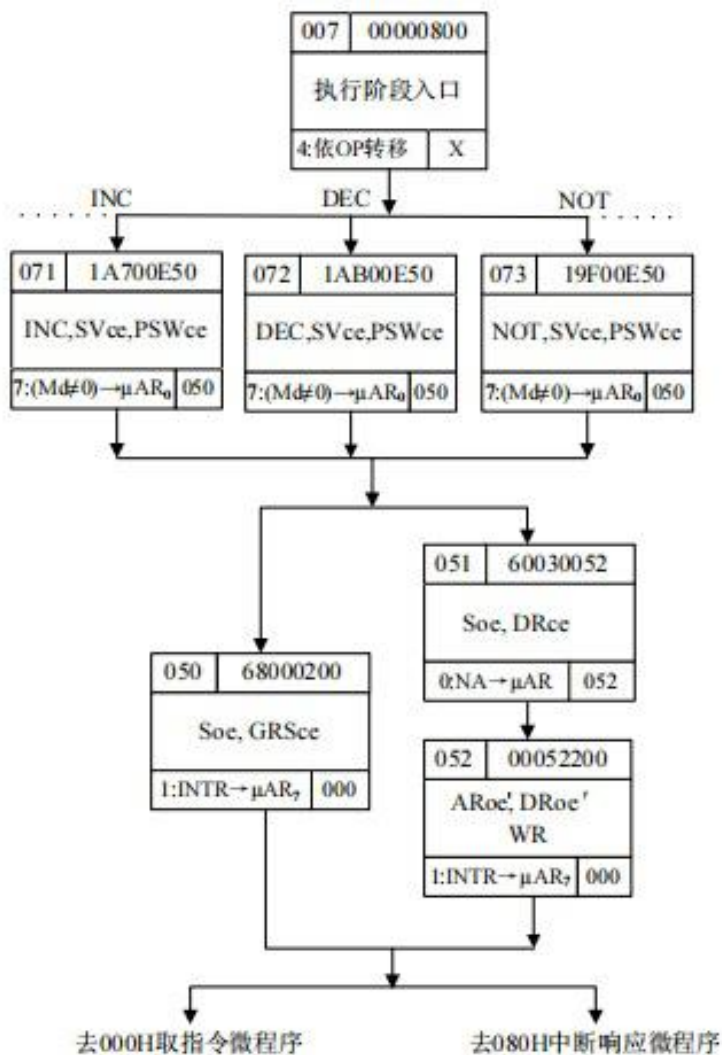


图 1-12 单操作数运算指令的执行阶段微程序

3. MOV 指令的微程序设计

MOV 指令是将源操作数送给目的寄存器或内存单元，上面单操作数指令运算之后也是要将运算结果送给目的寄存器或内存单元（见图 1-12），为了能够和运算指令共用保存结果的微程序，MOV 指令的微程序需要将 TR 中的源操作数送给 SHIFTER 寄存器，见表 1-9。

表 1-9 MOV 指令的微程序

微地址(H)	微指令(H)	微指令字段(H)										微命令
		F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	
041	94000040	4	5	0	0	0	0	0	0	0	040	TRoe, Acc
040	00300E50	0	0	0	3	0	0	0	0	7	050	SVce

041 微指令将 TR 送到 ALU 的 A 输入端，但是不给任何运算控制信号，使 ALU 的 A 输入端与 0 相加；下一条微指令将结果保存在 SHIFTER 中，并根据目的操作数的寻址方式决定转向 050H 还是 051H，将 SHIFTER 的内容保存到目的寄存器或内存单元。

MOV 指令的微程序入口地址 041H 是根据图 1-11 计算得到，下一条微指令的地址原则上可以是任何空闲的控存单元，查表 1-6 可知 040H 控存单元是空闲的，故本例采用 040H。

4. HALT 指令

HALT 是停机指令，主要用于调试时避免程序跑飞。该指令的微程序只需一条微指令，实现原地踏步，如下。

微地址(H)	微指令(H)	微指令字段(H)										微命令
		F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	
058	00000058	0	0	0	0	0	0	0	0	0	058	

注意，停机指令执行后，只有复位才能使 CPU 重新工作。

1.5 输入输出和中断系统

1.5.1 输出接口

基本输出接口可以用来连接实验板上的 LED 指示灯。如 DE2-115 实验板有 9 个绿色 LED 和 18 个红色 LED，模型机系统设计了 2 个基本输出接口连接到这些 LED，接口地址见表 1-10。

表 1-10 LED 输出接口地址分配表

接口地址	有效位数	输出接口寄存器
FF01H	8	绿色 LEDG[7:0] 数据寄存器
FF02H	16	红色 LEDR[15:0] 数据寄存器

注：红色 LED[17:16]和绿色 LED[8]没有使用

DE2-115 实验板的外设比较丰富，同学们还可以自己设计一些输出接口连接到这些外设，如数码管、字符液晶显示器、VGA 显示器接口等。

1.5.2 输入接口

基本输入接口可以用来连接实验板上的拨动开关。如 DE2-115 实验板有 18 个拨动开关和 4 个按键，模型机系统设计了 4 个输入接口连接到这些拨动开关和按键。每个输入接口含有 1 个 16 位的数据寄存器和 1 个 1 位的状态寄存器，如图 1-13 所示。4 个输入接口的 READY 分别连接到 DE2-115 实验板的 KEY0~KEY3 四个按键，而 DATA 则共用 16 个拨动开关（最高两位拨动开关 SW17~16 没有使用）。

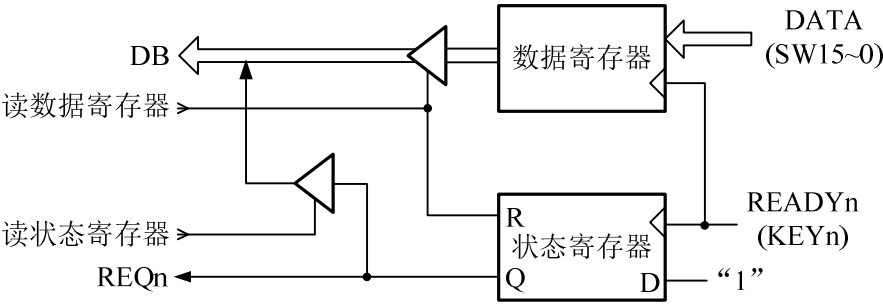


图 1-13 输入接口逻辑框图

当按下某一个 KEYn 按键时，将 16 个拨动开关作为数据打入该接口的数据寄存器，同时状态寄存器置 1；状态寄存器的输出可以由 CPU 通过数据总线读出，同时也可以作为中断请求 REQn 送给中断控制器；CPU 读该接口寄存器时，数据寄存器读信号打开三态门，将数据寄存器的内容输出到 DB 数据总线，同时将状态寄存器清零。

接口地址见表 1-11。

表 1-11 开关输入接口地址分配表

接口地址	有效位数	输入接口寄存器
FF08H	16	输入数据寄存器 0
FF09H	1	输入状态寄存器 0
FF0AH	16	输入数据寄存器 1
FF0BH	1	输入状态寄存器 1
FF0CH	16	输入数据寄存器 2
FF0DH	1	输入状态寄存器 2
FF0EH	16	输入数据寄存器 3
FF0FH	1	输入状态寄存器 3

DE2-115 实验板的外设比较丰富，同学们还可以自己设计一些输入接口连接到这些外设，如 PS/2 接口、红外接收接口、RS-232 接口等。

1.5.3 中断控制器

中断控制器的组成如图 1-14所示。中断屏蔽寄存器的地址是 FF00H，因为模型计算机的外设与主存统一编制，故可以使用 MOV 指令访问 FF00H 地址实现对中断屏蔽寄存器写入或读出。

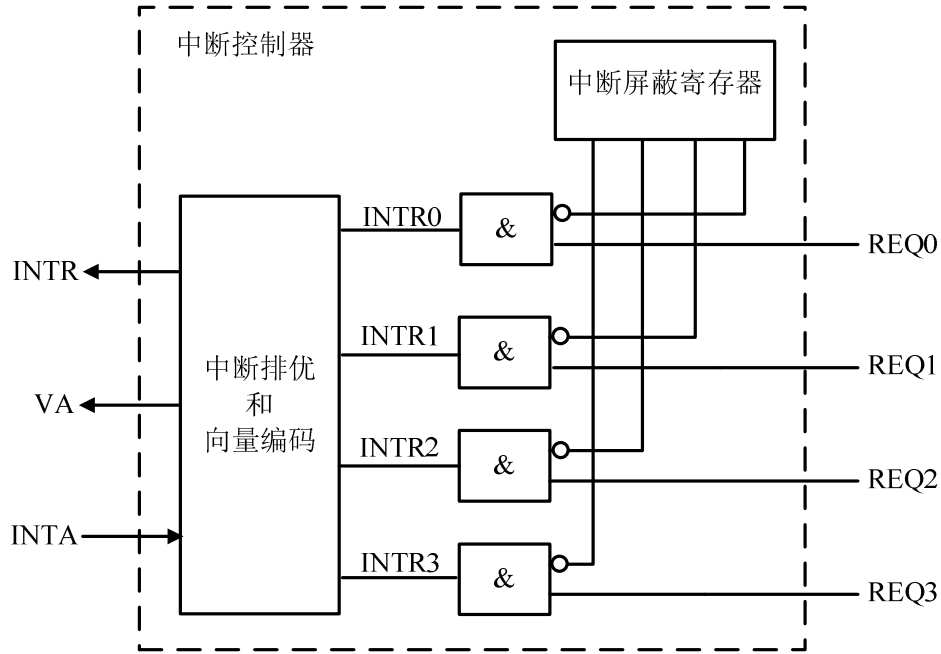


图 1-14 中断控制器组成框图

中断系统采用向量中断方式，中断向量表的首地址是 0000H，每个中断向量占用一个存储单元，存放该中断服务程序的入口地址。向量地址与中断源的对应关系见表 1-12。

表 1-12 中断向量地址

中断源	向量地址
INTR0	0000H
INTR1	0001H
INTR2	0002H
INTR3	0003H

第 2 章 十六位微程序控制计算机的设计

本章讲述课程设计的任务，主要针对前面所述的 OpenJUC-II 教学模型机，设计控制器的微程序，实现该模型机的指令系统。通过课程设计理解指令的执行过程，指令系统与硬件的关系，进而加深对计算机的结构和工作原理的理解。（具体内容参见：[JUC-II 教学模型计算机.PDF](#)）。

微程序设计的任务分解为六次，内容由简单到复杂、循序渐进地完成设计；每次任务大致可以用一天的时间完成，半天用于设计，如微程序设计、测试程序设计等，半天用于在实验设备上调试验证。

2.1 熟悉微程序的设计和调试方法

1. 设计目标

- （1）掌握微程序的设计方法
- （2）掌握将汇编指令翻译成机器指令的方法
- （3）熟悉利用调试软件运行、调试微程序的方法

2. 设计任务

（1）阅读前面模型机设计相关内容，掌握微程序的设计方法；完成以下指令所涉及的微程序设计与调试。

```
ORG 0030H
INC 0040H
JMP 0030H
```

注：*ORG 0030H* 是一条伪指令，不产生实际的机器指令，只是指明下面一条指令的地址是 0030H。

（2）将上述程序中的主存单元地址 0040H 换成红色 LED 输出寄存器的地址 FF02H，以“机器指令单步”方式运行程序，观察 LED 指示灯的变化。再以“连续运行”方式运行程序，LED 指示灯出现什么现象？思考其原因。

也可使用调试软件虚拟实验板界面观察 LED 指示灯，需启用虚拟实验板的自动刷新功能。

（3）换一种指令和寻址方式，如减 1 指令、间接寻址，编写相关的微程序并调试通过。可以采用“微指令断点”方式运行，将断点设在需要调试的微指令地址上，提高调试效率。

3. 微程序设计

- （1）取指令的微程序设计

微地址(H)	微指令(H)	微指令字段(H)										微命令
		F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	
000	20080001	1	0	0	0	2	0	0	0	0	001	PCoe, ARce
001	00069002	0	0	0	0	1	2	1	1	0	002	ARoe', RD, DRce', PCinc
002	CC000003	6	3	0	0	0	0	0	0	0	003	DRoe, IRce
003	00000404	0	0	0	0	0	0	0	0	2	004	BM2

(2) 取目的操作数的微程序设计

(1) 直接寻址的微程序

微地址(H)	微指令(H)	微指令字段(H)										微命令
		F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	

(3) 执行阶段的微程序设计

(1) INC 指令的微程序

微地址(H)	微指令(H)	微指令字段(H)										微命令
		F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	

(2) 保存运算结果的微程序

.....

(3) JMP 指令的微程序

.....

4.操作提示

主要调试步骤如下（具体操作参见实验调试软件用户手册）：

(1) 连接实验设备

注意：请在断电状态下连接调试适配器电缆。

(2) 下载 FPGA 配置数据

(3) 输入微程序

利用调试软件将微程序写入控存。任务要求中的大部分微程序在模型机设计中已经给出，包括取指令的微程序、目的操作数直接寻址的微程序和单操作数运算指令的执行阶段微程序。

说明：资料中 day1.jcm 为第一部分 INC、JMP 指令的微程序文件。

(4) 输入调机程序

首先将调机指令翻译成机器码，以 INC 0040H 为例，已知单操作数双字指令格式如下。

15	12	11	6	5	3	2	0
0	0	0	0	OP		Md	Rd
目的操作数寻址方式中包含的常数							

查指令编码表，INC 指令的操作码 OP 为 010001；查寻址方式编码表，直接寻址的编码 M 为 100；该指令不涉及寄存器，寄存器字段可取任意值，通常采用 000。指令的第二个字为直接地址 0040H。所以该指令的二进制编码如下。

15	12	11	6	5	3	2	0
0	0	0	0	0	1	1	0
0	0	0	0	0	0	0	0

由于实验系统只接受十六进制格式，故需要将上面的二进制指令码转换为十六进制，用如下形式表示：

0030: 0460; INC 0040H

0031: 0040;

冒号前面的是内存单元的地址。由于 CPU 复位时 PC=0030H，所以第一条指令的地址必须为 0030H。JMP 指令的机器码请同学们自己翻译，填入下面的横线中。

0032: ____; JMP 0030H

0033: 0030;

利用调试软件将上述 2 条指令的机器码写入主存。

(5) 运行与调试

利用调试软件以微指令单步方式运行并观察运行数据、分析运行结果是否正确。运行前首先向 0040H 单元输入一个已知的数值；INC 0040H 指令运行完成后在调试软件上刷新主存显示，观察 0040H 单元的内容是否已经加 1。调试记录中需记录 0040H 单元内容的前后变化。

关于微地址寄存器 μAR 的说明：由微程序控制时序可知，每个微指令周期结束在 CP1 上升沿之后，因此该微指令的执行结果已被保存下来，从而通过调试软件可以观察到执行结果；同时 CP1 将 μAG 形成的下一条微指令的微地址打入 μAR ，因此这时 μAR 的内容并不是当前执行的微指令的微地址，而是接下来将要执行的微指令的地址。调试系统特别设计了一个寄存器保存当前刚执行完的这条微指令的微地址，以方便通过调试软件观察。

(6) 保存微程序

将调试成功的微程序保存下来，在以后几天的设计中将不断增加微程序，最终实现整个指令系统。

2.2 双操作数指令的设计与调试

1. 设计目标

完成双操作数指令指令的微程序设计和验证；取源操作数阶段和取目的操作数阶段相关寻址方式的微程序设计和验证。

2. 设计任务

(1) 编写 MOV 指令和源操作数立即寻址的微程序，并用下面的调机程序验证。

MOV #0101H, 0040H

(2) 编写 SUB 指令的微程序，并用下面的调机程序验证。

MOV #0101H, 0040H

SUB #FFFFH, 0040H

观察 0040H 单元和 PSW 的变化。

(3) 编写取源操作数和取目的操作数的寄存器寻址、寄存器间接寻址等所有寻址方式的微程序，并设计调机程序验证。

提示：可以借助调试软件的微指令编码工具计算微指令编码。

MOV 0040H, R1

MOV R1, 0042H

JMP 0030H

(4) 编写 ADD、ADDC 指令的微程序，并设计一段程序实现双倍字长（32 位）的加法运算，用这个程序验证微程序。

2.3 条件转移指令的设计与调试

1.设计目标

为 CPU 扩充转移指令，完成转移指令的微程序设计 with 调试。

2.设计原理

(1) 微程序设计

转移指令是单操作数指令，取操作数阶段结束以后，目的数存放在寄存器 A 中，但对转移指令而言没有意义，转移指令只用 AR 的内容作为转移地址(转移指令不支持寄存器寻址)。

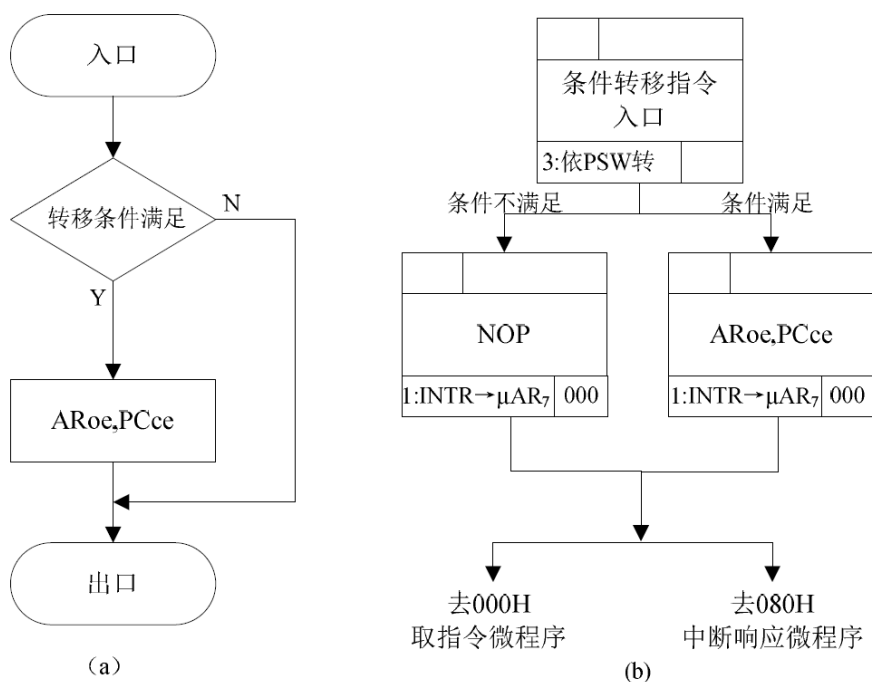


图 1 转移指令微流程和微程序

执行时根据转移条件是否满足两分支转移，条件满足时将 AR 的内容送 PC，不满足时顺序执行，由转移方式字段 BM=3 控制。图 1(a)为条件转移指令微流程，(b)为以微流程

(2) 硬件设计

为了使之前设计的初级 CPU 支持转移类指令，必须对硬件进行扩充。包括 2 个方面：

① 条件转移指令是根据条件标志位的情况来确定下一条微指令的地址的，因此要修改微地址形成电路模块 uAG，增加端口，把 PSW 中的低 4 位接进来。

② uAG 中，增加 BM=3 的微转移方式。

BM=3 时(见 [JUC-II 教学模型计算机.PDF](#) 表 2.5)，微地址的形成规则是：

NA→uAR，

$f\{OP, PSW(Z, O.S, C)\} \rightarrow uAR_0$

即先 NA→uAR，再根据条件转移指令操作码和 PSW 的 ZF、OF、SF、CF 状态标志决定 uAR₀，若满足条件 uAR₀=1，否则 uAR₀=0。F{OP, PSW(Z, O.S, C)} 所代表的微地址修改逻辑见图 2。

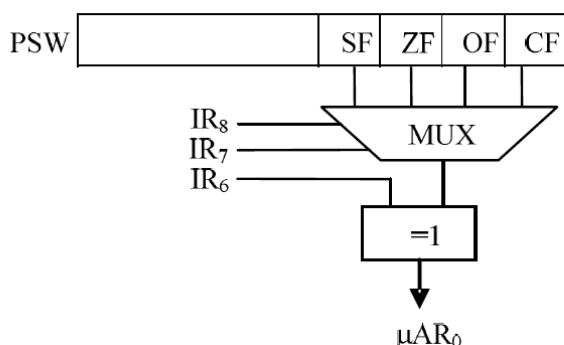


图2 BM3 微地址修改逻辑

3.设计任务

(1) 根据前面介绍的微地址形成方法，算出条件转移指令的微程序入口地址；编写 JC 指令的微程序输入到控存，并用下面的调机程序验证。

```

ORG 0030H
MOV #imm1, R1
CMP #imm2, R1
JC ADDR1
MOV #0001H, FF01H
HALT

ADDR1: MOV #0080H, FF01H
HALT

```

上面调机程序的立即数 #imm1 和 #imm2 用具体的数值代替，设计两组立即数使得转移和不转移两种情况均能出现，观察 LED 的变化和 PC 值的变化并分析原因。

如果尚未实现 CMP 指令，需编写 CMP 指令的微程序。CMP 指令与 SUB 指令类似，只是相减的结果不保存到目的操作数，但是影响 PSW 中的标志位。CMP 指令通常与条件转移指令配合使用。

提示：可以在调试软件的汇编窗口输入汇编指令，由调试软件翻译为机器指令。

(2) 编写相对寻址的微程序，将上述调机程序中条件转移指令的直接寻址方式改为相对寻址，并运行验证。

(3) 编写 JZ 指令和 TEST 指令的微程序。TEST 指令与 AND 指令类似，只是逻辑与的结果不保存到目的操作数，但是影响 PSW 中的标志位。和 CMP 指令一样，编程时 TEST 指令通常与条件转移指令配合使用。

下面的程序是一个查询输入程序。它首先测试开关输入接口 0 的状态寄存器，如果有数据则读出开关输入接口的内容并送给红色 LED 接口显示。有关输入接口的硬件设计见参考资料 [JUC-II 教学模型计算机.PDF](#)。

```

ORG 0030H

ADDR1:  TEST #0001H,FF09H    ;测试状态寄存器 0 的最低位

```

```

JZ ADDR1          ;如果为 0 继续查询，否则顺序执行
MOV FF08H,FF02H   ;将开关数据寄存器 0 的数据反映到红色 LED
JMP ADDR1

```

用这个程序验证 TEST 和 JZ 指令的微程序。运行时如果没有输入数据，则循环测试状态寄存器；然后用拨动开关设置输入数据，按下 KEY0 键将数据保存在输入寄存器中，此时测试状态寄存器为 1，执行 MOV 指令将开关数据显示在红色 LED 上。

(4) 设计一个循环程序实现软件延时，插入到下面的调机程序中，使得每次 LED 的变化能维持一段时间，以“连续运行”方式运行该程序，观察延时效果；调整延时时间，使得人眼能够分辨出指示灯的变化。

```

ORG 0030H
INC FF02H
(这里加入延时程序)
JMP 0030H

```

(5) 编写寄存器自增间接寻址的微程序，编写一段调机程序，该程序将 0100H 开始的 8 个存储单元的内容复制到 0110H 开始的 8 个存储单元。要求使用寄存器自增间接寻址，并使用条件转移指令构成循环结构。

2.4 移位指令的设计与调试

1. 设计目标

为 CPU 扩充移位指令，完成移位指令的微程序设计。与调试。

实验 CPU 设计有 7 条移位质量。移位指令是单操作数指令，目的操作数是要进行移位操作的数据，没执行一次指令对数据移位移位。移位操作如下图所示：

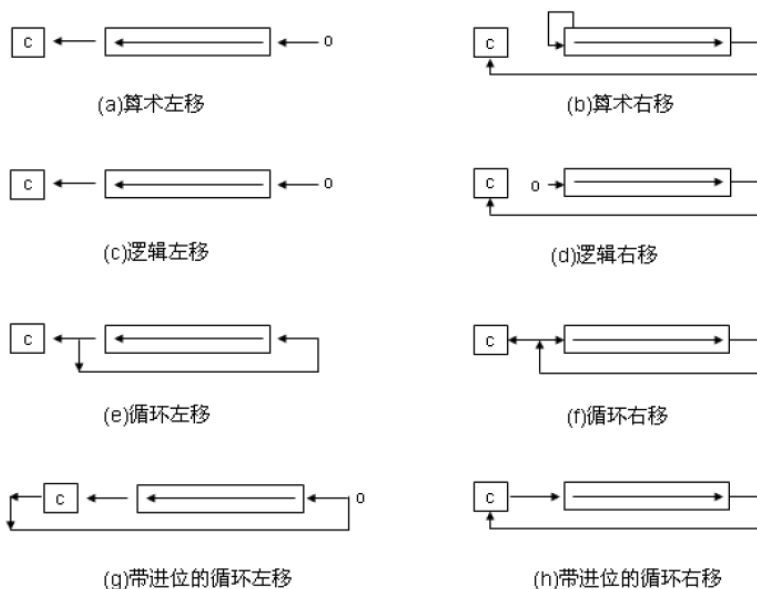


图 3 移位方式

从上图可以看出，移位指令会影响 PSW，无论是哪种移位，移出的数据位都是送到 CF 标志位；不同移位指令的区别是移入的数据来源不同。

(1) 移位寄存器移入数据的选择

如果 16 位移位寄存器的输入数据是 d ，由图 4 可以看出，左移运算所完成的操作是 $d[14:0] \rightarrow d[15:1]$ ， $d[0]$ 的取值根据不同的移位操作可取 0、 $d[15]$ 或 CF，用 lsb 表示移入的数据位。左移操作实现原理如图 4。

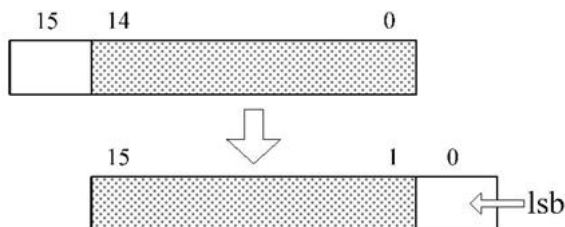


图 4 左移操作实现原理

lsb 的生成逻辑如图 5，利用多路器根据指令编码对移入数据进行选择。

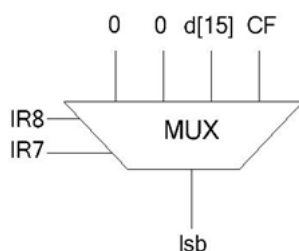


图 5 左移移入位 lsb 选择逻辑

同理，右移运算所完成的操作是 $d[15:1] \rightarrow d[14:0]$ ， $d[15]$ 的取值根据不同的移位操作可取 0、 $d[0]$ 、 $d[15]$ 或 CF，用 hsb 表示右移移入的数据位，实现原理如图 6。

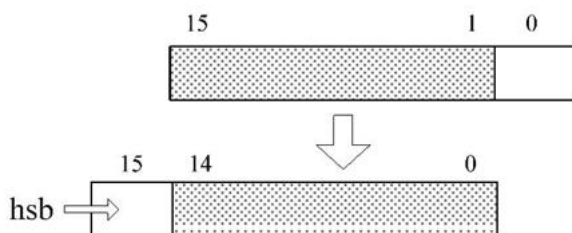


图 6 右移操作实现原理

hsb 的生成逻辑如图 7。

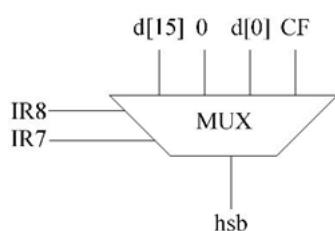


图 7 右移移入位 hsb 选择逻辑

(2) PSW 标志位 CF 输入端的选择

初级 CPU 的 CF 标志位是根据 ALU 的运算结果产生的，引入移位指令以后，影响 CF 的有两个来源，一个源于 ALU 的运算结果，一个源于移位器的移位输出。移位指令对 CF 位的影响如图 3，由图可以看出，左移时最高位送 CF，右移时最低位送 CF。CF 生成逻辑如图 8。

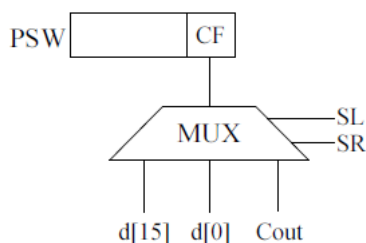


图 8 CF 选择逻辑

2.设计任务

(1) 设计右移指令的微程序，用以下调机程序验证。

```
ORG 0030H
MOV #0001H, R0
SHR R0
JC -3(PC)
HALT
```

分析上述调机程序的功能，运行之后相关寄存器和 PSW 会有怎样的变化，程序是否转移，转移的目的地址是多少。

(2) 设计左移指令的微程序，用以下调机程序验证。

```
ORG 0030H
MOV #0505H, R1
TEST #0001H, R1
JZ 3(PC)
ROL R1
JMP 0032H
HALT
```

分析上述调机程序的功能，运行之后相关寄存器和 PSW 会有怎样的变化，程序是否转移，转移的目的地址是多少。

(3) 设计自己的调机程序，使用更多的移位指令。

(4) 设计一段程序，实现 8 个绿色 LED 流水灯，要求使用移位指令。

(5) 设计一段程序，使用条件转移指令实现程序的循环，实现 $1+2+3+\dots+10$ 。

2.5 堆栈相关指令的设计与调试

1.设计目标

为 CPU 扩充 PUSH、POP、CALL、RET 指令，完成微程序设计。

2.设计任务

(1) 编写 PUSH 和 POP 指令的微程序，并用下面的调机程序验证。

```
ORG 0030H
MOV #0041H, R0
PUSH R0
PUSH 0040H
POP (R0)
POP R1
HALT
```

观察堆栈指针 SP、堆栈存储单元以及相关寄存器和内存单元的变化，理解堆栈的用法。

(2) 编写 CALL 指令的微程序，并设计调机程序验证。注意观察转向子程序后，堆栈指针及堆栈内容的变化。

(3) 编写 RET 指令的微程序，并设计调机程序验证。注意观察子程序返回是否返回到正确的地址，以及返回后堆栈指针的变化。

(4) 将第三部分的软件延时程序改写成子程序，第四部分的流水灯程序中调用该软件延时子程序。注意子程序中保护寄存器，即对子程序中用到的寄存器压入堆栈，返回前再从堆栈中恢复。运行过程中注意观察寄存器、堆栈指针及堆栈内容的变化。

(5) 设计一子程序，完成乘以 10 的运算，运算数由主程序通过堆栈传入，运算结果也通过堆栈传出。设计一主程序调用该子程序，在模型机上运行并验证结果。提示：实验 CPU 没有乘法指令，乘以 10 可以用 $(N*8+N*2)$ 实现。

2.6 中断相关指令的设计与调试

1.设计目标

完成整个中断过程各个环节的设计。

2.设计原理

模型机系统采用向量中断，有四个中断源，由四个按键 KEY0~KEY3 产生中断请求，同时将拨动开关的电平值保存到输入数据寄存器。相关硬件设计参见 **JUC-II 教学模型计算**

机.PDF2.7 输入输出。

每条指令执行结束时，检测是否有中断请求(INTR 有效) 并且 CPU 是否允许中断(IF 有效)，如果是，则转移到微地址 080H，执行中断隐指令；否则转向 000H 取下一条指令。这个检测由硬件完成，在微程序中通过 BM=1 控制硬件进行检测，例如图 2.25 执行阶段微程序保存运算结果结束时的 050 和 052 微指令。此时微地址的形成逻辑为：
 $NA \rightarrow uAR, INTR \cdot IF \rightarrow uAR_7$

设置 NA=000H，则两分支的微地址分别是 000H 和 080H，080H 是中断隐指令的微程序入口。

3.设计任务

(1) 编写中断隐指令的微程序。中断隐指令完成的操作是：

- ① 保护 PC，即 PC 入栈。
- ② 保护 PSW，即 PSW 入栈。
- ③ 中断响应信号 INTA 有效，读中断向量地址 VA。
- ④ 根据 VA，取出中断服务程序的入口地址，送 PC。
- ⑤ 关中断。

(2) 编写开中断指令(EI)的微程序，用下面的调机程序验证中断隐指令。

```
ORG 0030H
MOV #0100H, 0000H
EI
MOV.....
.....
```

假设 KEY0 中断服务程序的入口地址是 0100H，该程序中首先初始化中断向量表，由表 2.14 可知 KEY0 的向量地址是 0000H，因此用#0100H 初始化 0000H 单元；然后开中断。在开始执行 MOV 指令后，按下 KEY0 键，产生中断请求；在 MOV 指令执行结束时将检测中断请求，执行中断隐指令；之后将转向中断服务程序。

(3) 编写中断返回指令(RET)的微程序。中断返回指令 RETI 完成的操作是：

- ① 恢复 PSW；
- ② 恢复 PC；
- ③ 开中断。

(4) 将下面的中断服务程序翻译成机器指令，用它验证中断返回指令。该中断服务程序中，读取开关输入接口寄存器的值，输出到红色 LED 输出接口寄存器中。

```
ORG 0100H
MOV FF08H, FF02H
EI
RETI
```

(5) 尝试设计多重中断(中断嵌套)程序。

2.7 答辩及验收

1.验收选题

完成验收测试编程题，根据难易程度分 A（优）、B（良）、C（中和及格）三档，要求先设计微程序，然后编写汇编程序，调试运行出正确的结果，并回答三个问题。

序号	微程序完成类型（A） INC、DEC 等单操作数、双 操作数、条件转移指令、 移位指令、堆栈、子程序、 中断	微程序完成类型（B） INC、DEC 等单操作数、双操作数、 条件转移指令、移位指令	微程序完成类型（C） INC、DEC 等单操作数、双 操作数、条件转移指令
1			
2			
3			
4			

2.评分标准

评分标准包括：程序设计难易、实物验收及答辩正确性、课设报告是否规范等。