



计算机组成原理课程设计

姓　　名　　 罗毅

班　　级　　 软件工程2303班

学　　号　　 302023315014

提交日期　　 2025/6/8

目录

[一、实验目的 1](#_Toc103858468)

[二、实验内容 1](#_Toc103858469)

[三、实验原理 1](#_Toc103858470)

[3.1指令系统及分析 1](#_Toc103858471)

[3.2指令框图及分析 3](#_Toc103858472)

[3.3指令系统对应微程序二进制代码实现及分析 6](#_Toc103858473)

[3.4机器程序及分析 7](#_Toc103858474)

[四、实验步骤 1](#_Toc103858475)7

[4.1微程序写入及校验 1](#_Toc103858476)7

[4.2程序运行 1](#_Toc103858477)8

[五、实验结果及分析 1](#_Toc103858478)8

[六、实验问题及思考 1](#_Toc103858481)9

[七、实验总结 2](#_Toc103858482)0

[7.1验收答辩问题回答 2](#_Toc103858483)0

[7.2实验心得体会 2](#_Toc103858484)1

# 一、实验目的

综合运用所学计算机组成原理知识，设计并实现较为完整的计算机。能够理解并掌握计算机的整机结构，掌握基本功能模块的工作原理和操作方法，掌握控制器的微程序实现并且训练逻辑分析和故障排查能力。

# 二、实验内容

1. 设计并实现一套完整的指令系统；
2. 设计并实现完整的计算机（采用上述指令系统）；
3. 利用该计算机实现 多功能计时秒表 。

# 三、实验原理

## 3.1指令系统及分析

1、指令格式设计

为了支持多功能计时秒表的需求，设计了两种指令格式：单字节指令和双字节指令。

单字节指令：共8位，4-7位为操作码OP，指定指令类型；2-3位为RS，源寄存器编号；0-1位为RD，目标寄存器编号。

|  |  |  |
| --- | --- | --- |
| 7-4 | 3-2 | 1-0 |
| OP | RS | RD |

双字节指令：共16位，第一字节的组成和单字节指令相同，第二字节的8位用于表示立即数。

第一字节：

|  |  |  |
| --- | --- | --- |
| 7-4 | 3-2 | 1-0 |
| OP | RS | RD |

第二字节：

|  |
| --- |
| 7-0 |
| 立即数 |

2、微指令格式及其分析说明

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 23 | 22 | 21 | 20 | 19 | 18～15 | 14～12 | 11～9 | 8～6 | 5～0 |
| M23 | INTA | WR | RD | IOM | S3～S0 | A字段 | B字段 | C字段 | MA5～MA0 |

以上是具体的微指令格式表格，其中23位保留，22位表示中断响应输入，21位表示写，20为表示读，19位表示IN/OUT单元，18～15位表示运算器中的运算操作，14～12位表示A字段，11～9位表示B字段，8～6位表示C字段，5～0位表示后续微地址。A、B、C字段的具体含义如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | 10 | 9 | 选择 |
| 0 | 0 | 0 | NOP |
| 0 | 0 | 1 | ALU\_B |
| 0 | 1 | 0 | RO\_B |
| 0 | 1 | 1 | PD\_B |
| 1 | 0 | 0 | RI\_B |
| 1 | 0 | 1 | SP\_B |
| 1 | 1 | 0 | PC\_B |
| 1 | 1 | 1 | 保留 |

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 7 | 6 | 选择 |
| 0 | 0 | 0 | NOP |
| 0 | 0 | 1 | P<1> |
| 0 | 1 | 0 | P<2> |
| 0 | 1 | 1 | P<3> |
| 1 | 0 | 0 | P<4> |
| 1 | 0 | 1 | LDPC |
| 1 | 1 | 0 | STI |
| 1 | 1 | 1 | CLI |

|  |  |  |  |
| --- | --- | --- | --- |
| 14 | 13 | 12 | 选择 |
| 0 | 0 | 0 | NOP |
| 0 | 0 | 1 | LDA |
| 0 | 1 | 0 | LDB |
| 0 | 1 | 1 | LDRi |
| 1 | 0 | 0 | LDSP |
| 1 | 0 | 1 | LOAD |
| 1 | 1 | 0 | LDAR |
| 1 | 1 | 1 | LDIR |

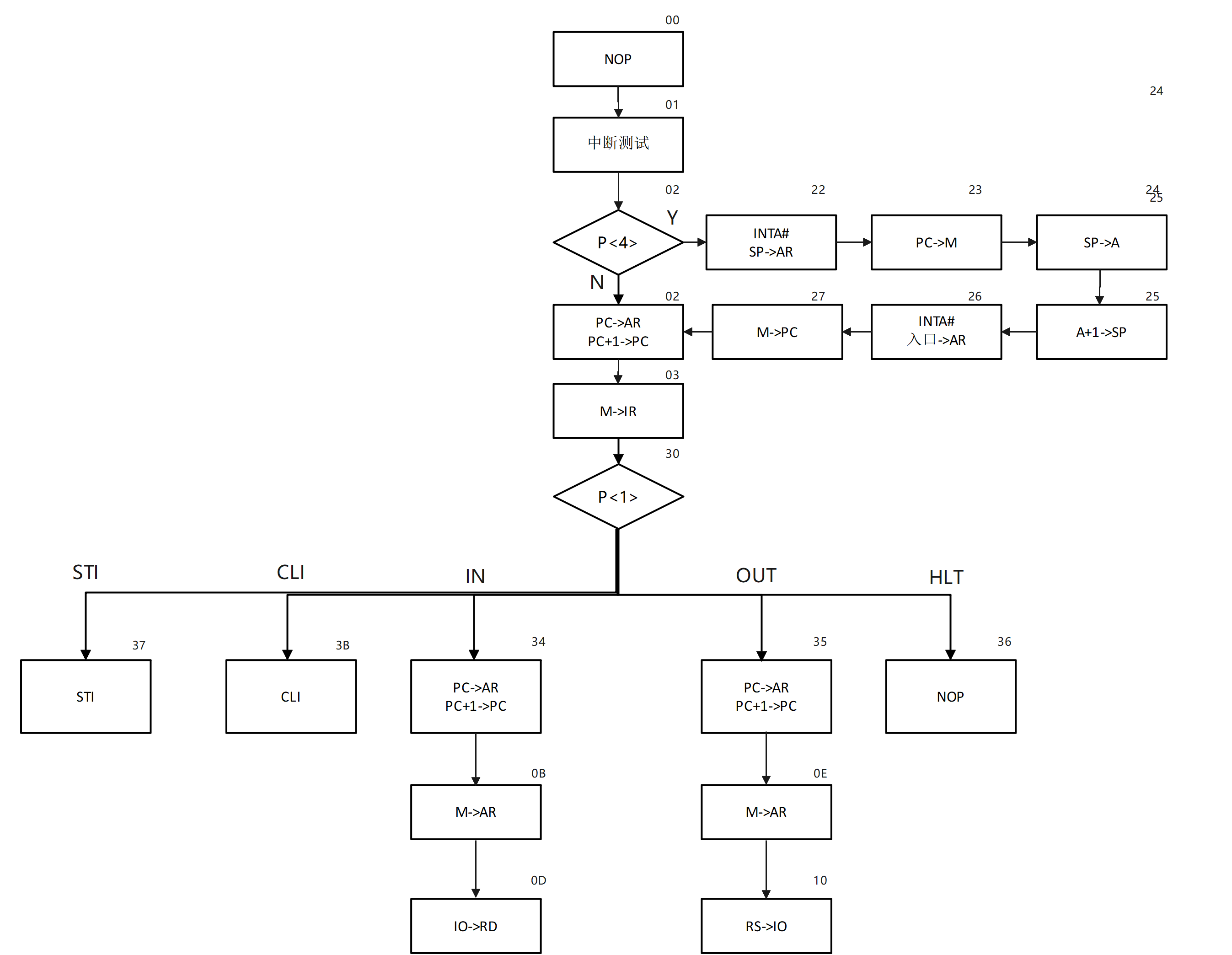
3、完整指令集设计与功能说明

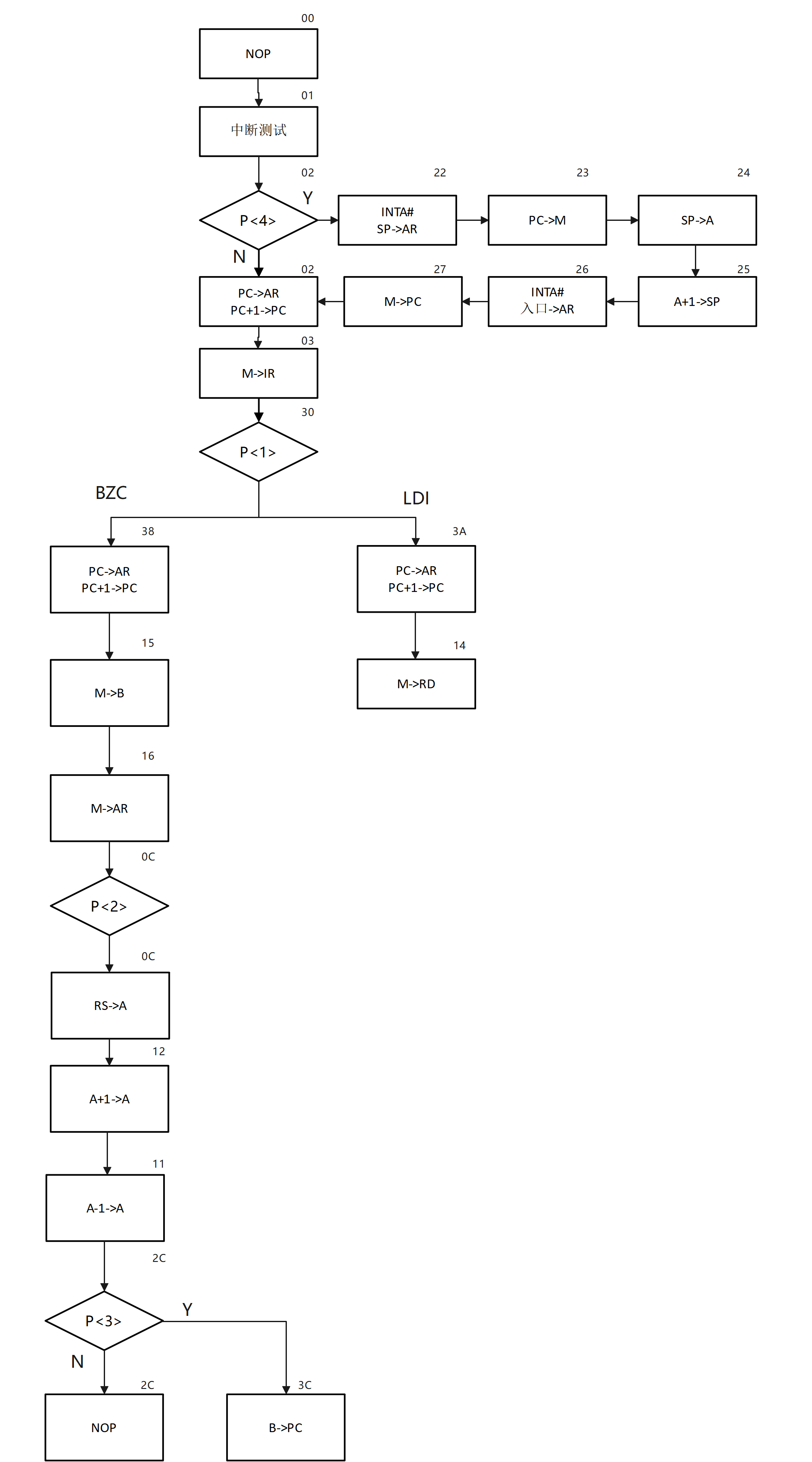
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 指令 | 功能说明 | 操作码 | 格式 | | | |
| MOV | RS->RD | 0 | 0000 | RS | | RD |
| ADD | RD+1->RD | 1 | 0001 | RD | | RD |
| SUB | RD-1->RD | 2 | 0010 | RD | | RD |
| JMP | D->PC | 3 | 0011 | \*\* | \*\* | D |
| IN | 从I/O端口读取数据 | 4 | 0100 | \*\* | RD | P |
| OUT | 向I/O端口写入数据 | 5 | 0101 | RS | \*\* | P |
| HLT | 程序停止 | 6 | 0110 | \*\* | | \*\* |
| STI | CPU开中断 | 7 | 0111 | \*\* | | \*\* |
| BZC | 当RS=0时,D->PC | 8 | 1000 | RS | \*\* | D |
| IRET | 中断返回 | 9 | 1001 | \*\* | | \*\* |
| LDI | 加载数据到寄存器 | A | 1010 | \*\* | RD | D |
| CLI | CPU关中断 | B | 1011 | \*\* | | \*\* |

以上为完整的指令集设计，对ADD和SUB重新设计，修改为自增1和自减1。对指令BZC同样在实验教材的基础上进行了修改，从对FZ判0修改为对RS判0,即可以判断寄存器中的数字是否为0。指令JMP和指令BZC均采用直接寻址。

## 1(3)3.2指令框图及分析

指令周期图由如下三张图所示：





分析说明如下：每一条指令都从NOP进入，之后进入中断测试。如果中断测试成功，则会跳转到中断向量处，执行指令，直到执行IRET指令后返回执行原来的指令。译码器根据P<1>字段译码，分析出每一条指令对应的第一条执行的微指令，并且根据每一条微指令的后续微地址找到下一条微指令并执行。若其中一条微指令的后续微地址为中断测试，则不会再继续执行微指令，指令执行结束。BZC指令的框图中还有P<3>字段，用于判断FC或者FZ是否为1，根据FC和FZ决定后续微地址。

## 3.3指令系统对应微程序二进制代码实现及分析

微程序流程图所对应的完整微程序二进制代码表如下所示：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 功能 | 十六进制地址 | 二进制地址 | 十六进制表示 | 高五位 | S3～S0 | A字段 | B字段 | C字段 | MA5～MA0 |
| NOP | 00 | 000000 | 000001 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| 中断测试 | 01 | 000001 | 000102 | 00000 | 0000 | 000 | 000 | 100 | 000010 |
| PC->AR  PC+1->PC | 02 | 000010 | 006D43 | 00000 | 0000 | 110 | 110 | 101 | 000011 |
| M->IR | 03 | 000011 | 107070 | 00010 | 0000 | 111 | 000 | 001 | 110000 |
| A-1->S | 04 | 000100 | 064205 | 00000 | 1100 | 100 | 001 | 000 | 000101 |
| SP->AR | 05 | 000101 | 006A06 | 00000 | 0000 | 110 | 101 | 000 | 000110 |
| M->PC | 06 | 000110 | 105141 | 00010 | 0000 | 101 | 000 | 101 | 000001 |
| A+1->RD | 07 | 000111 | 06B201 | 00000 | 1101 | 011 | 001 | 000 | 000001 |
| A-1->RD | 08 | 001000 | 063201 | 00000 | 1100 | 011 | 001 | 000 | 000001 |
| M->A | 09 | 001001 | 10100A | 00010 | 0000 | 001 | 000 | 000 | 001010 |
| M->AR | 0A | 001010 | 10608F | 00010 | 0000 | 110 | 000 | 010 | 001111 |
| M->AR | 0B | 001011 | 10608D | 00010 | 0000 | 110 | 000 | 010 | 001101 |
| RS->A | 0C | 001100 | 001412 | 00000 | 0000 | 001 | 010 | 000 | 010010 |
| IO->RD | 0D | 001101 | 183001 | 00011 | 0000 | 011 | 000 | 000 | 000001 |
| M->AR | 0E | 001110 | 106010 | 00010 | 0000 | 110 | 000 | 000 | 010000 |
| A->PC | 0F | 001111 | 005341 | 00000 | 0000 | 101 | 001 | 101 | 000001 |
| RS->IO | 10 | 010000 | 280401 | 00101 | 0000 | 000 | 010 | 000 | 000001 |
| A-1->A | 11 | 010001 | 0612EC | 00000 | 1100 | 001 | 001 | 011 | 101100 |
| A+1->A | 12 | 010010 | 069211 | 00000 | 1101 | 001 | 001 | 000 | 010001 |
| NOP | 2C | 101100 | 000001 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| B->PC | 3C | 111100 | 00D341 | 00000 | 0001 | 101 | 001 | 101 | 000001 |
| M->RD | 14 | 010100 | 103001 | 00010 | 0000 | 011 | 000 | 000 | 000001 |
| M->B | 15 | 010101 | 102016 | 00010 | 0000 | 010 | 000 | 000 | 010110 |
| M->AR | 16 | 010110 | 10608C | 00010 | 0000 | 110 | 000 | 010 | 001100 |
| INTA#  SP->AR | 22 | 100010 | 406A23 | 01000 | 0000 | 110 | 101 | 000 | 100011 |
| PC->M | 23 | 100011 | 200C24 | 00100 | 0000 | 000 | 110 | 000 | 100100 |
| SP->A | 24 | 100100 | 001A25 | 00000 | 0000 | 001 | 101 | 000 | 100101 |
| A+1->SP | 25 | 100101 | 06C226 | 00000 | 1101 | 100 | 001 | 000 | 100110 |
| INTA#  入口->AR | 26 | 100110 | 406027 | 01000 | 0000 | 110 | 000 | 000 | 100111 |
| M->PC | 27 | 100111 | 105142 | 00010 | 0000 | 101 | 000 | 101 | 000010 |
| RS->RD | 30 | 110000 | 003401 | 00000 | 0000 | 011 | 010 | 000 | 000001 |
| RD->A | 31 | 110001 | 001407 | 00000 | 0000 | 001 | 010 | 000 | 000111 |
| RD->A | 32 | 110010 | 001408 | 00000 | 0000 | 001 | 010 | 000 | 001000 |
| PC->AR  PC+1->PC | 33 | 110011 | 006D49 | 00000 | 0000 | 110 | 110 | 101 | 001001 |
| PC->AR  PC+1->PC | 34 | 110100 | 006D4B | 00000 | 0000 | 110 | 110 | 101 | 001011 |
| PC->AR  PC+1->PC | 35 | 110101 | 006D4E | 00000 | 0000 | 110 | 110 | 101 | 001110 |
| NOP | 36 | 110110 | 000001 | 00000 | 0000 | 000 | 000 | 000 | 000001 |
| STI | 37 | 110111 | 000181 | 00000 | 0000 | 000 | 000 | 110 | 000001 |
| PC->AR  PC+1->PC | 38 | 111000 | 006D55 | 00000 | 0000 | 110 | 110 | 101 | 010101 |
| SP->A | 39 | 111001 | 001A04 | 00000 | 0000 | 001 | 101 | 000 | 000100 |
| PC->AR  PC+1->PC | 3A | 111010 | 006D54 | 00000 | 0000 | 110 | 110 | 101 | 010100 |
| CLI | 3B | 111011 | 0001C1 | 00000 | 0000 | 000 | 000 | 111 | 000001 |

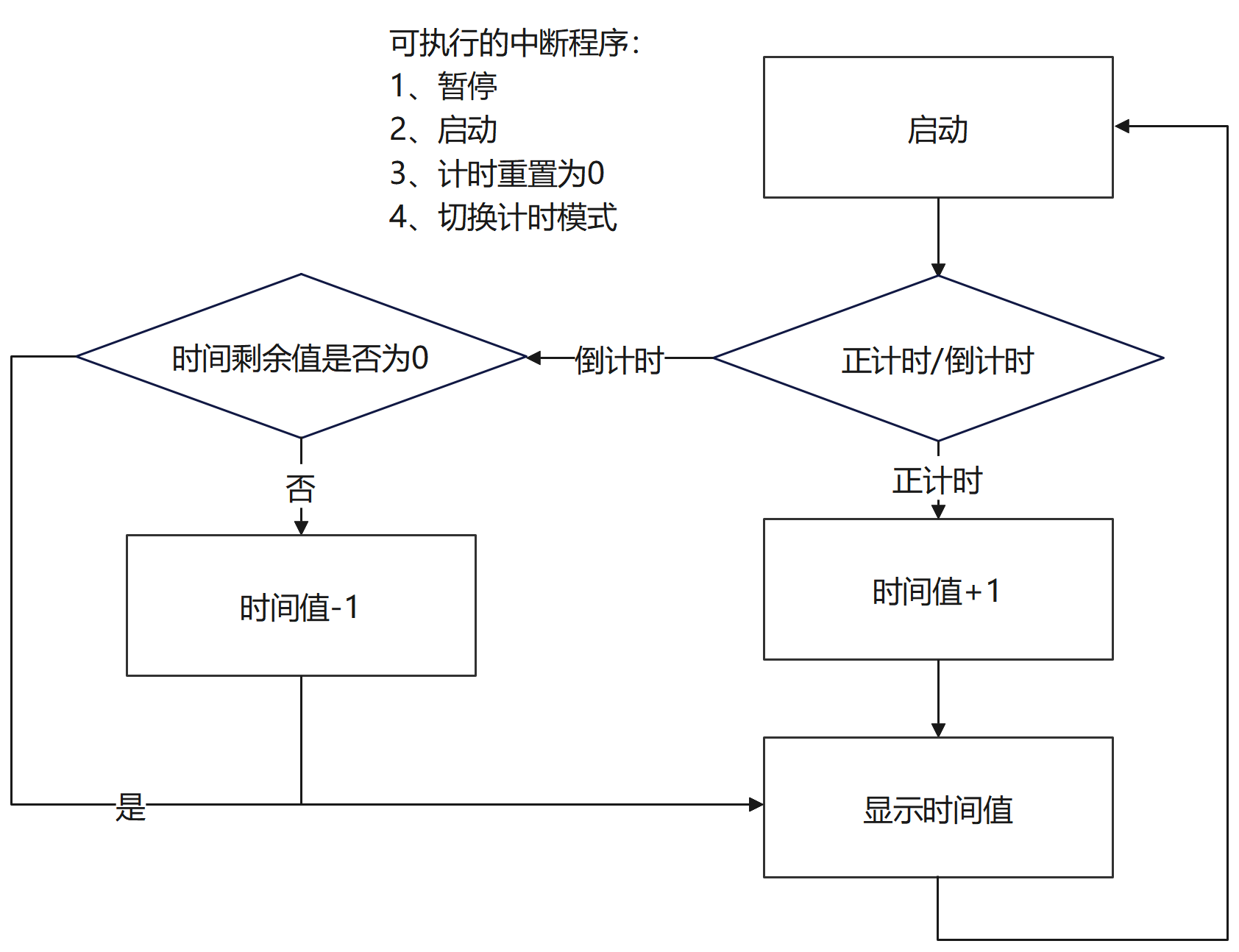
新指令ADD和SUB的源操作数和目的操作数均存放在同一个寄存器中，所以寄存器均记为RD，这样两个指令自增自减的概念更加清晰。将寄存器中的数字送入A后，ADD进行A+1运算，SUB进行A-1运算，最后将运算好的结果送入RD。

新指令BZC为了判断寄存器RS中的数字是否为0，并且如果为0则PC需要跳到第二字节所对应的地址，首先将PC送AR并且PC自增1。内存通过AR中的内容，找到了第二字节对应的地址，并且将地址送B。然后将RS中的数字送A用于判断是否为0，经过A+1送A，再经过A-1送A之后就能通过FZ标志判断。最后通过P<3>字段的规则生成散转下址，如果不为0则不操作，如果为0则B送PC，程序就会执行第二字节对应地址的指令。

## 3.4机器程序及分析

1、程序功能设计及说明

实现的功能有启动/暂停计时、复位归零、实时显示当前计时值、计时模式切换。其中，程序启动时，时间值为0，并且默认为正计时模式开始运行，时间值不断变大。外界可以分别发出启动、暂停、复位归零、计时模式切换这四种功能的中断请求，从而实现多功能计时器。

2、程序流程图

以上为程序流程图。程序启动后，判断当前为正计时模式还是倒计时模式。若为正计时模式，则时间值+1后输出时间值并且返回循环起点继续判断；若为倒计时模式，则需判断时间剩余值是否为0，为0则直接显示，不为0则时间值-1后显示，最终也返回循环起点继续判断。

3、程序源代码说明

程序源代码如下：

; //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

; // 计时器程序 - 带中断处理能力的模型机设计实验 //

; //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

; //\*\*\*\*\* Start of Main Memory Data \*\*\*\*\*//

$P 00 A0 ; LDI R0 ; 初始化当前时间值为0

$P 01 00 ; 存0

$P 02 A1 ; LDI R1 ; 初始化为正计时模式(1=正计时,0=倒计时)

$P 03 01 ; 存1，默认正计时

$P 04 A2 ; LDI R2 ; 初始化暂停标志为1(1=运行,0=暂停)

$P 05 01 ; 存1，默认运行

$P 06 50 ; PRINT R0 ; 打印当前时间值

$P 07 40 ;输出到OUT单元即40端口

;8259初始化配置

$P 08 A3 ; LDI R3 13H

$P 09 13

$P 0A 5C ;OUT R3至端口C0H

$P 0B C0

$P 0C A3 ;LDI R3 30H

$P 0D 30

$P 0E 5C ;OUT R3 至端口C1H

$P 0F C1

$P 10 A3

$P 11 03

$P 12 5C

$P 13 C1 ;82599初始化配置

$P 14 A3

$P 15 00

$P 16 5C

$P 17 C1 ;开放全部中断请求

$P 18 A3 ;此时重新利用R3为栈顶指针

$P 19 A0 ;初始化为A0

$P 1A 70 ;开中断

; 主循环

$P 1B 8A ; BZC R2; 检查暂停标志

$P 1C 1B ; 实现死循环造成暂停

$P 1D 85 ; BZC R1 ;检查计时模式

$P 1E C1 ; 若为0，跳转倒计时模式

$P 1F 30 ; JMP 正计时 ;否则跳转正计时模式

$P 20 40

;存储中断向量信息

$P 30 60

$P 31 70

$P 32 80

$P 33 90

; 正计时逻辑

$P 40 10 ; ADD R0变为R0+1 ; 时间值加1

$P 41 50 ; 输出时间值到OUT单元

$P 42 40

$P 43 30 ; 跳转到主循环

$P 44 1B

; 倒计时逻辑

$P C1 50 ; 输出R0到显示设备

$P C2 40

$P C3 85 ;检查R1是否为0，如果为1可以跳出下方的死循环

$P C4 C7 ;R1若为0，则应继续倒计时

$P C5 30 ;R1若为1，则应跳到主循环

$P C6 1B

$P C7 80 ;判断R0是否为0

$P C8 C1 ;为0跳到C1实现死循环

$P C9 20 ;不为0则减一

$P CA 30 ;不为0跳到主循环

$P CB 1B

;暂停

$P 60 8A ;检查暂停标志，如果已经暂停则退出中断

$P 61 63

$P 62 2A; 若标志为1,改为0,暂停

$P 63 90 ;中断返回

;启动

$P 70 8A ;检查暂停标志，如果已经暂停才修改暂停标志

$P 71 73 ;如果不是0就执行72，中断返回

$P 72 90 ;中断返回

$P 73 1A ;若标志为0，改为1，启动

$P 74 90 ;中断返回

;复位归零

$P 80 A0 ;

$P 81 00 ;

$P 82 90 ;中断返回

;切换计时模式

$P 90 85 ;判断当前计时模式是否为倒计时

$P 91 94 ;倒计时+1变为正计时

$P 92 25 ;正计时-1变为倒计时

$P 93 90 ;中断返回

$P 94 15 ;

$P 95 90

; //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

; // 微控制器程序 - 处理中断和计时器基础操作 //

; //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

$M 00 000001

$M 01 000102

$M 02 006D43

$M 03 107070

$M 04 064205

$M 05 006A06

$M 06 105141

$M 07 06B201

$M 08 063201

$M 09 10100A

$M 0A 10608F

$M 0B 10608D

$M 0C 001412

$M 0D 183001

$M 0E 106010

$M 0F 005341

$M 10 280401

$M 11 0612EC

$M 12 069211

$M 2C 000001

$M 3C 00D341

$M 14 103001

$M 15 102016

$M 16 10608C

$M 22 406A23

$M 23 200C24

$M 24 001A25

$M 25 06C226

$M 26 406027

$M 27 105142

;3x为指令微程序入口

$M 30 003401

$M 31 001407

$M 32 001408

$M 33 006D49

$M 34 006D4B

$M 35 006D4E

$M 36 000001

$M 37 000181

$M 38 006D55

$M 39 001A04

$M 3A 006D54

$M 3B 0001C1

其中$P后存储的是指令信息，$M后存储的是微指令信息，每一条指令的执行均由若干条微指令的执行组成。在源代码的关键部分中加上了注释信息，便于理解代码含义和检查。

4、机器码说明

主程序的机器码如下所示：

00000000 10100000 ; LDI R0 ; 初始化当前时间值为0

00000001 00000000 ; 存0

00000010 10100001 ; LDI R1 ; 初始化为正计时模式(1=正计时,0=倒计时)

00000011 00000001 ; 存1，默认正计时

00000100 10100010 ; LDI R2 ; 初始化暂停标志为1(1=运行,0=暂停)

00000101 00000001 ; 存1，默认运行

00000110 01010000 ; PRINT R0 ; 打印当前时间值

00000111 01000000 ;输出到OUT单元即40端口

;8259初始化配置

00001000 10100011 ; LDI R3 13H

00001001 00010011

00001010 01011100 ;OUT R3至端口C0H

00001011 11000000

00001100 10100011 ;LDI R3 30H

00001101 00110000

00001110 01011100 ;OUT R3 至端口C1H

00001111 11000001

00010000 10100011

00010001 00000011

00010010 01011100

00010011 11000001 ;82599初始化配置

00010100 10100011

00010101 00000000

00010110 01011100

00010111 11000001 ;开放全部中断请求

00011000 10100011 ;此时重新利用R3为栈顶指针

00011001 10100000 ;初始化为A0

00011010 01110000 ;开中断

; 主循环

00011011 10001010 ; BZC R2; 检查暂停标志

00011100 00011011 ; 实现死循环造成暂停

00011101 10000101 ; BZC R1 ;检查计时模式

00011110 11000001 ; 若为0，跳转倒计时模式

00011111 00110000 ; JMP 正计时 ;否则跳转正计时模式

00100000 01000000

;存储中断向量信息

00110000 01100000

00110001 01110000

00110010 10000000

00110011 10010000

; 正计时逻辑

01000000 00010000 ; ADD R0变为R0+1 ; 时间值加1

01000001 01010000 ; 输出时间值到OUT单元

01000010 01000000

01000011 00110000 ; 跳转到主循环

01000100 00011011

; 倒计时逻辑

11000001 01010000 ; 输出R0到显示设备

11000010 01000000

11000011 10000101 ;检查R1是否为0，如果为1可以跳出下方的死循环

11000100 11000111 ;R1若为0，则应继续倒计时

11000101 00110000 ;R1若为1，则应跳到主循环

11000110 00011011

11000111 10000000 ;判断R0是否为0

11001000 11000001 ;为0跳到C1实现死循环

11001001 00100000 ;不为0则减一

11001010 00110000 ;不为0跳到主循环

11001011 00011011

以下为四个中断服务程序：

;暂停

01100000 10001010 ;检查暂停标志，如果已经暂停则退出中断

01100001 01100011

62 2A; 若标志为1,改为0,暂停

01100011 10010000 ;中断返回

;启动

01110000 10001010 ;检查暂停标志，如果已经暂停才修改暂停标志

01110001 01110011 ;如果不是0就执行72，中断返回

01110010 10010000 ;中断返回

01110011 00011010 ;若标志为0，改为1，启动

01110100 10010000 ;中断返回

;复位归零

10000000 10100000 ;

10000001 00000000 ;

10000010 10010000 ;中断返回

;切换计时模式

10010000 10000101 ;判断当前计时模式是否为倒计时

10010001 10010100 ;倒计时+1变为正计时

10010010 00100101 ;正计时-1变为倒计时

10010011 10010000 ;中断返回

10010100 00010101 ;

10010101 10010000

# 四、实验步骤

## 4.1微程序写入及校验

（1）写入微程序：选择联机软件的“转储”→“装载”命令，将该格式的文件（\*.txt）装载入TD-CMA实验系统。装载过程中，在软件的输出区的“结果”栏会显示装载信息，如当前正在装载的是机器指令，还是微指令，还剩多少条指令等。

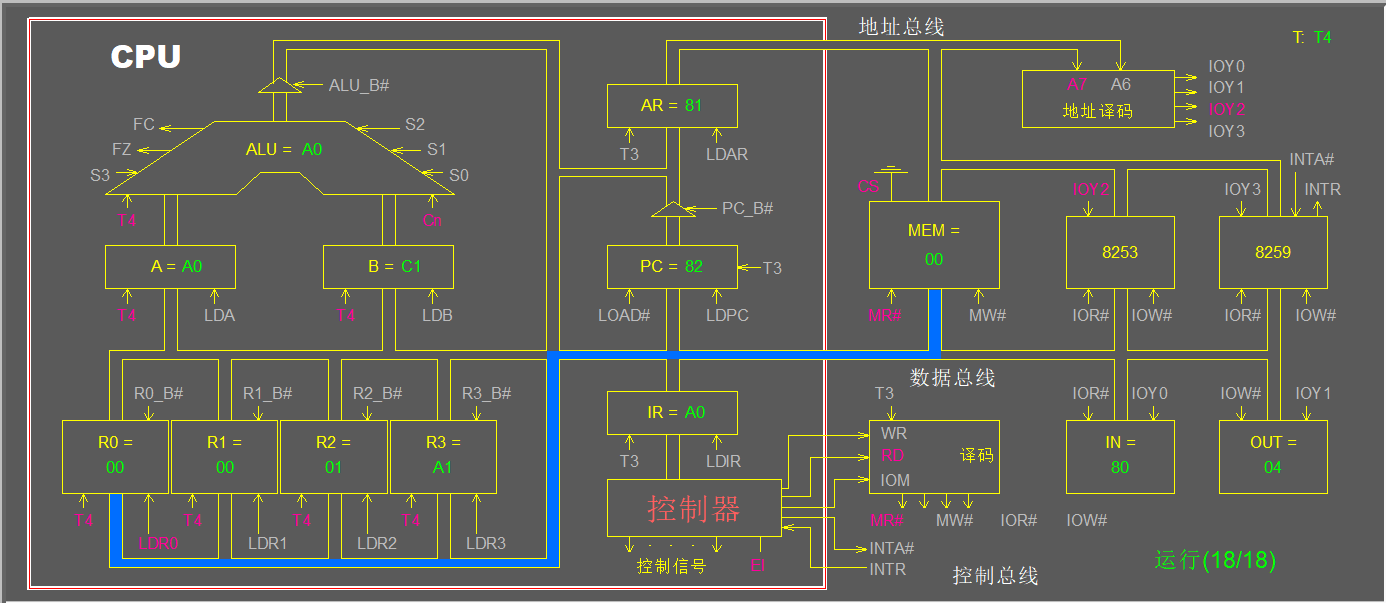
（2）校验微程序：选择联机软件的“转储”→“刷新指令区”命令可以读出下位机所有的机器指令和微指令，并在指令区显示。检查微程序控制器相应地址单元的数据是否和以下十六进制数据相同，如果不同，说明写入操作失败，应重新写入，可以通过联机软件单独修改某个单元的微指令，先单击指令区的微存TAB按钮，然后单击需要修改的单元的数据，此时该单元变为编辑框，输入6位数据并按回车键，编辑框消失，写入的数据显示红色。

## 4.2程序运行

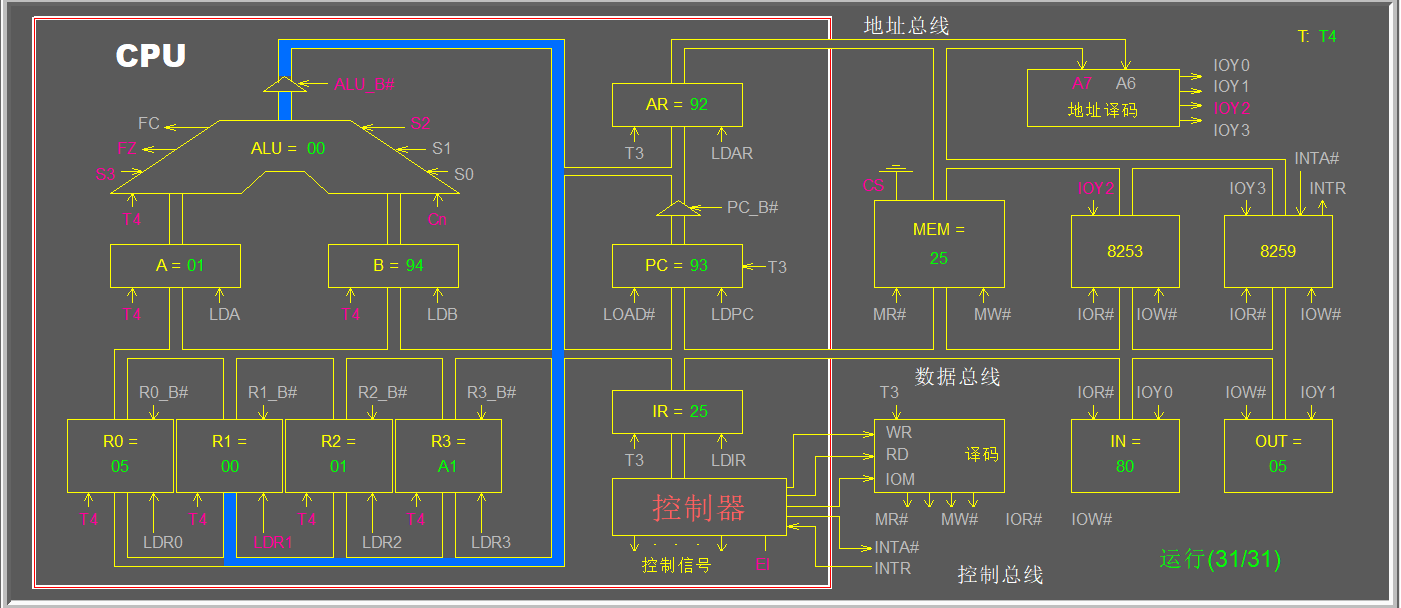
在软件界面中，点击右键，点击“连续运行”后，计时器即可启动。若要发出中断请求，需要对不同的端口发出中断请求信号。

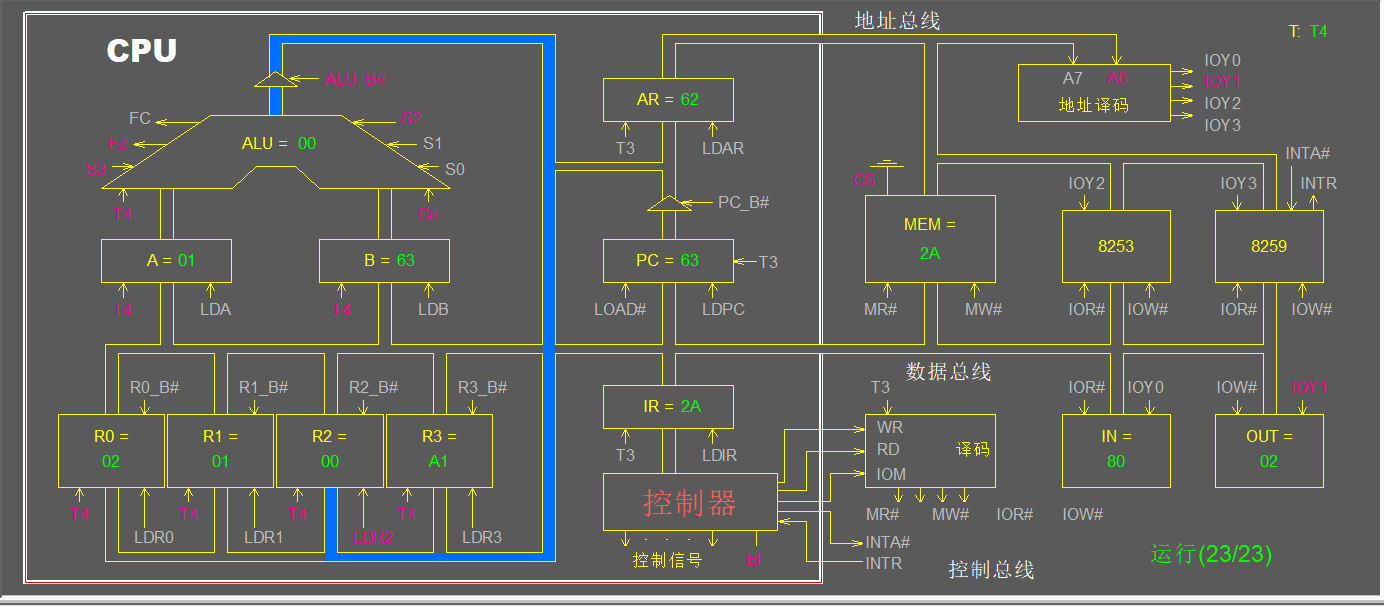
# 五、实验结果及分析

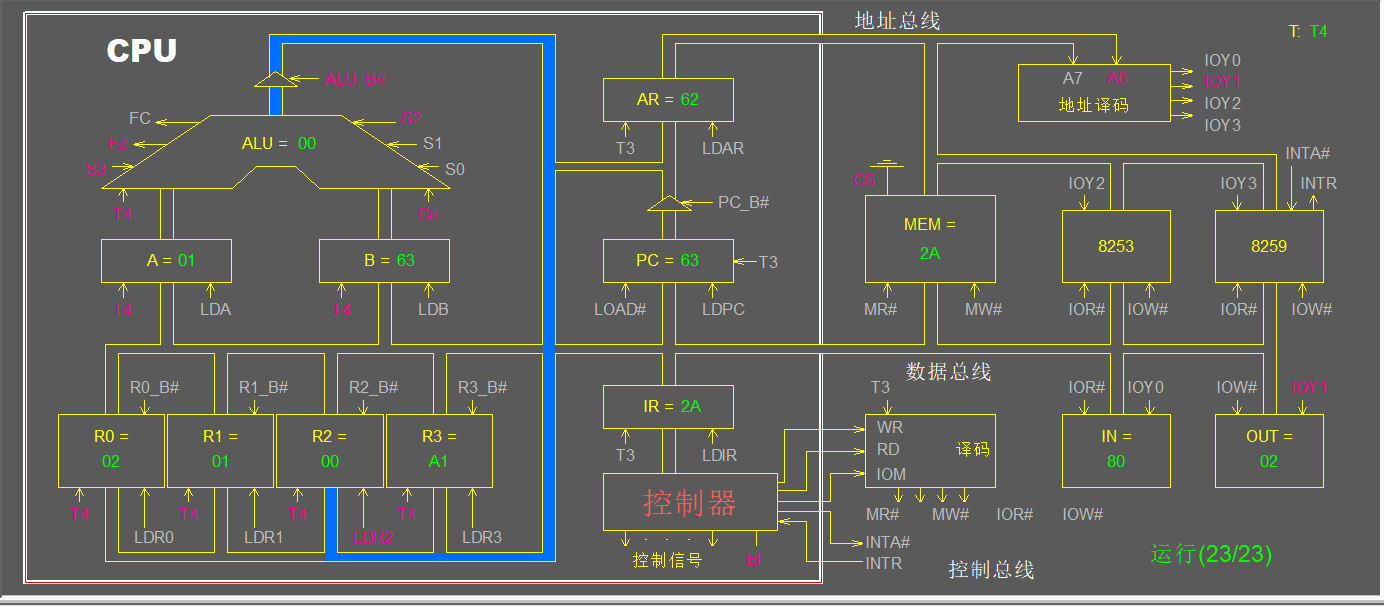
具体的演示视频在压缩包中。通过演示视频可以确认，多功能计时器的各功能可以正常运行。



上图为发出计时清零的中断请求的软件界面，可以看到R0被重置为0。



上图为发出切换计时模式的中断请求的软件界面，可以看到R1变为0，此时从正计时模式变为了倒计时。



上图为发出暂停的中断请求的软件界面，可以看到R2变成了0，此时运行中断。

# 启动

上图为发出启动的中断请求的软件界面，可以看到R2变成了1，此时运行继续。

# 六、实验问题及思考

1、当前所实现计算机，是否完整？如果不完整，还缺少哪些部件？

答：已经完整。

2、当前所实现计算机，是否能实现除法运算？如果能，可通过哪些指令实现除法运算？

答：无法实现除法运算。

3、当前所实现计算机，还能实现哪些更复杂的计算？请举例说明。

答：当前计算机能够实现一个多功能计时器，无法实现复杂运算。

多功能计时器还可调整时间变化速度，可以使时间变化得更慢。具体操作为，用一个寄存器存储循环次数，每一次循环对这个寄存器中的数字减一，当该数字为0时结束循环，之后再显示时间。这样就可以做到使时间变化得更慢。由于本实验中四个寄存器均有其作用，所以需要复用其中一个寄存器（比如用于判断正计时还是倒计时的寄存器），循环结束后恢复寄存器中内容为原来的数字。

4、当前所实现计算机，指令系统的双字长指令是如何实现的？

答：双字长指令中，第一字长部分的读入和单字长指令相同，PC在送AR之后自增1，内存根据AR找到这一地址的指令，并且送给IR。根据第一字长中的P<1>字段，可以找到第一条应该执行的微指令，在微指令中，PC会再次送AR并且自增1，内存再一次根据AR找到这一地址中的信息，此时这一地址中的信息即为第二字长的内容，可以被程序利用。

# 七、实验总结

## 7.1验收答辩问题回答

问题1：实验做了什么？实现了哪些功能？

答：实现了一个多功能计时器，可以实现启动计时、暂停计时、复位归零、实时显示当前计时值、正计时倒计时切换这些功能。

问题2：如何实现不同的中断请求？

答：首先需要在内存中特定的位置（在本次实验中为30-33）存入中断向量，其中30号位置代表着IR0端口对应的中断向量，31号位置代表着IR1端口对应的中断向量，以此类推。在发出中断请求时，根据自己想要实现的功能，选择对应端口后发出中断请求，这样就能实现多个不同的中断请求。

问题3：是否有自己设计的指令？

答：在实验教材的基础上，修改了ADD和SUB，使其效果为自增1和自减1，同时还修改了BZC，使其能够判断一个寄存器中的数字是否为0。

## 7.2实验心得体会

通过本次计算机组成原理课程设计，我深刻体会到了理论与实践相结合的重要性。从指令系统设计到微程序编写，再到最终实现多功能计时器，整个过程让我对计算机的整体架构有了更加直观和深入的理解。特别是在设计微指令格式和处理中断机制时，让我真正领悟到了计算机系统各个部件协调工作的精妙之处。

在实验过程中，我的问题解决能力得到了显著提升。面对众多技术难题，我学会了系统性地分析和调试。同时，对ADD、SUB和BZC指令的重新设计也培养了我的创新思维，让我认识到工程实践中需要根据具体应用需求来优化设计方案。

本次课程设计不仅巩固了我的专业理论知识，更重要的是培养了我的系统设计能力和工程素养。通过这次课程设计，我对计算机组成原理有了更加深刻的认识，这为我今后的专业学习和职业发展奠定了坚实的基础。