

计算机组成原理

课程设计报告

题 目： 设计一台嵌入式CISC模型计算机

学号、姓名：2100301709陈国健

2100301711郭宇

2100301725陶万启

专 业： 计算机科学与技术d

指 导 教 师：管军霖

所 属 学 院：计算机与信息安全学院

目录

[一 实验题目 3](#_Toc146232172)

[二 微程序控制器逻辑结构框图 3](#_Toc146232173)

[三 机器指令格式和指令系统 4](#_Toc146232174)

[四 微指令流程图和微指令代码表 4](#_Toc146232175)

[五 逻辑转移表达式 5](#_Toc146232176)

[六 汇编语言及其机器语言 5](#_Toc146232177)

[七 机器语言源程序的功能仿真波形图及结果分析 7](#_Toc146232178)

[八 故障现象和故障分析 11](#_Toc146232179)

[九 学习体会 12](#_Toc146232180)

[十 软件清单 13](#_Toc146232181)

[10.1 ALU单元 13](#_Toc146232182)

[10.2 状态条件寄存器PSW 15](#_Toc146232183)

[10.3 DECODER 16](#_Toc146232184)

[10.4 通用寄存器 17](#_Toc146232185)

[10.5 1:2分配器FEN2 17](#_Toc146232186)

[10.6 3选1数据选择器MUX3\_1 18](#_Toc146232187)

[10.7 4选1数据选择器MUX4\_1 19](#_Toc146232188)

[10.8 程序计算器PC 20](#_Toc146232189)

[10.9 地址寄存器AR 20](#_Toc146232190)

[10.10 主存储器ROM 21](#_Toc146232191)

[10.11 RAM 22](#_Toc146232192)

[10.12 指令寄存器IR 23](#_Toc146232193)

[10.13 时序产生器COUNTER 24](#_Toc146232194)

[10.14 微程序控制器CROM 25](#_Toc146232194)

[10.15 地址转移逻辑电路ADDR 25](#_Toc146232194)

[10.16 微地址寄存器aa 26](#_Toc146232194)

[10.17 微地址转换器F1 26](#_Toc146232194)

[10.18 控制存储器COUNTROM 27](#_Toc146232194)

[10.19 微命令寄存器MCOMMAN 28](#_Toc146232194)

[10.20 微地址转换器F2 29](#_Toc146232194)

[10.21 指令代码转换器F3 30](#_Toc146232194)

[附录 31](#_Toc146232195)

[附录A 顶层电路图 31](#_Toc146232194)

[附录B crom内部电路图 32](#_Toc146232194)

[附录C aa内部电路图 33](#_Toc146232194)

# 一 实验题目

设计一台嵌入式CISC模型计算机

采用定长CPU周期、联合控制方式，并运行能完成一定功能的机器语言源程序进行验证，机器语言源程序功能：输入包含10个整数(无符号数)的数组M，按从大到小的顺序输出这10个整数。

**CICS模型机数据通路框图**

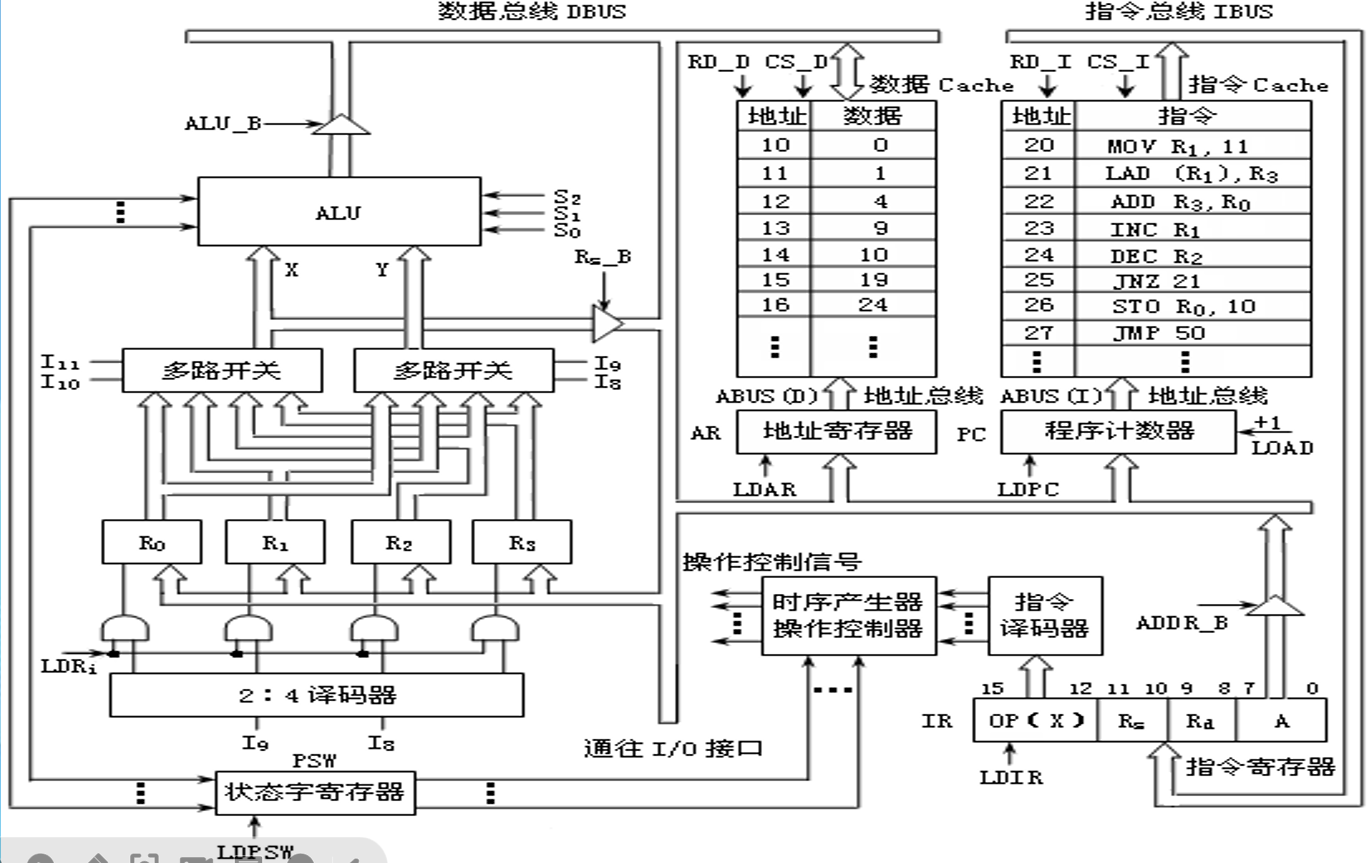


图1

# 二 微程序控制器逻辑结构框图

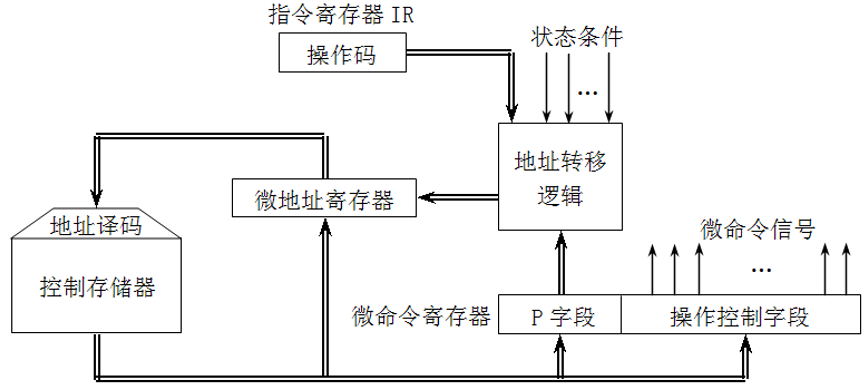


图2

# 三 机器指令格式和指令系统

本模型机中的指令系统中共有 12条汇编指令，表1列出了每条指令的助记符号、指令格式以及相应功能。

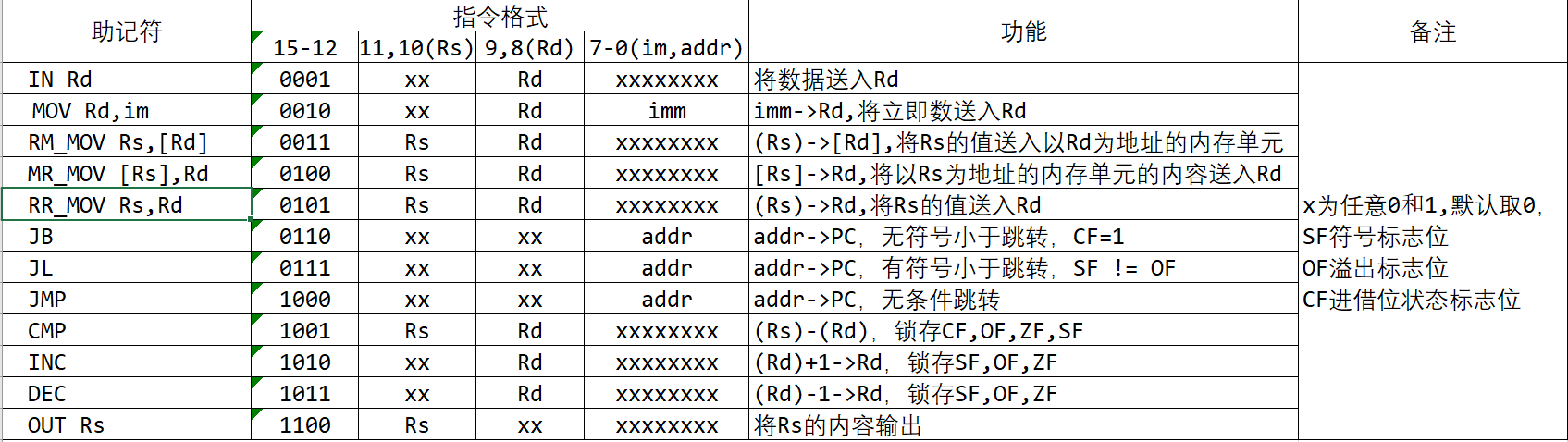


表1 12条汇编指令的助记符号、指令格式和功能

以下是对Rs和 Rd的规定：

|  |  |
| --- | --- |
| Rs或 Rd | 选定的寄存器 |
| 0 0 | R0 |
| 0 1 | R1 |
| 1 0 | R2 |
| 1 1 | R3 |

表2

# 四 微指令流程图和微指令代码表

（1）微指令流程图

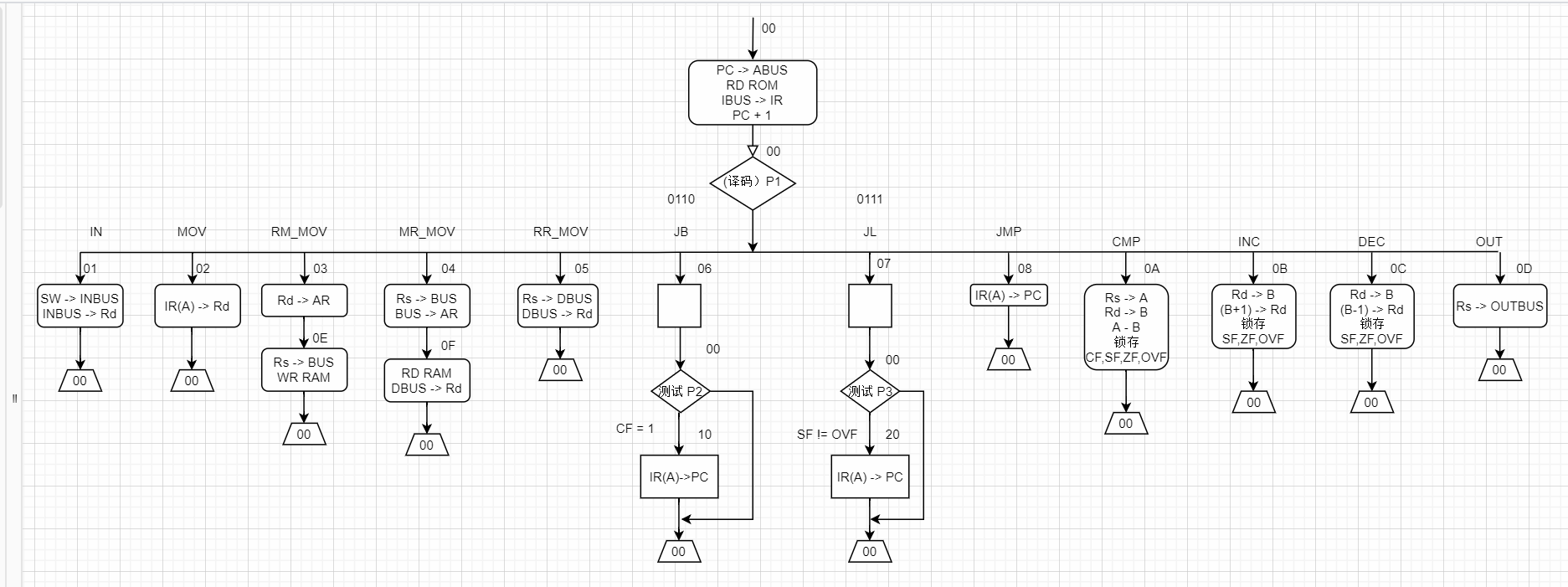


图3

（2）微指令代码表

CISC模型机系统使用的微指令采用全水平型微指令，字长为27位，其中微命令字段为18位，P字段为3位，后继微地址为6位，其格式如下:

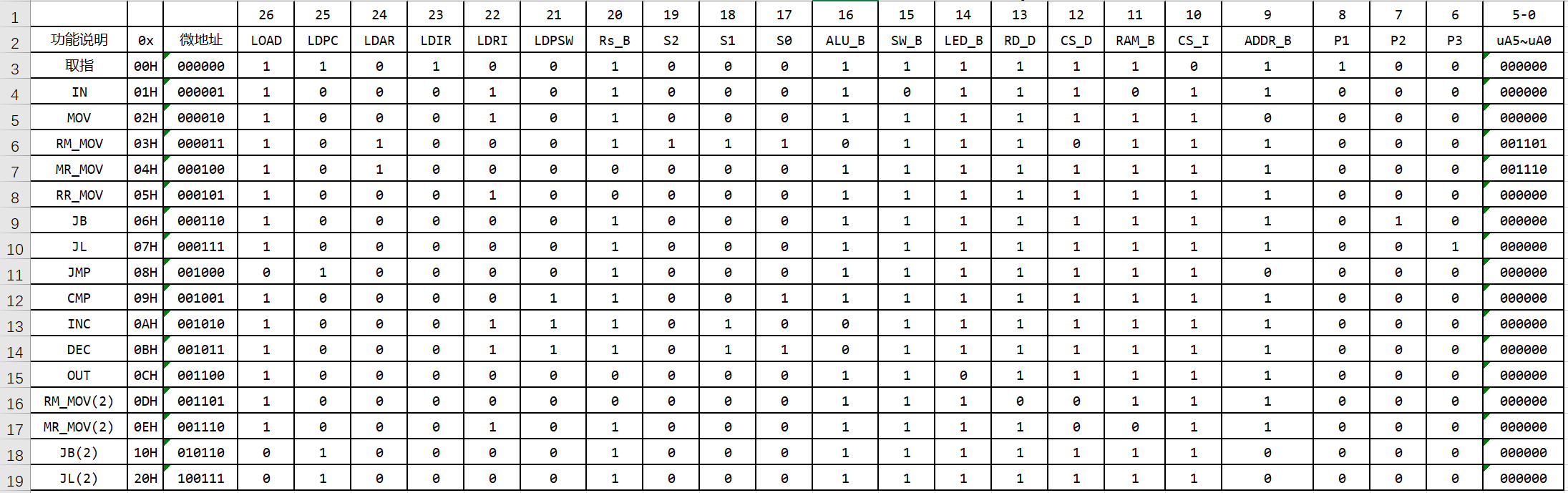


表3

# 五 逻辑转移表达式

SE5 <=NOT ((NOT(SF) AND OVF AND P3 AND T4) OR (SF AND NOT(OVF) AND P3 AND T4));

SE4 <=NOT (CF AND P2 AND T4);

SE3 <=NOT (I7 AND P1 AND T4);

SE2 <=NOT (I6 AND P1 AND T4);

SE1 <=NOT (I5 AND P1 AND T4);

SE0 <=NOT (I4 AND P1 AND T4);

其中I7,I6,I5,I4表示汇编指令的操作码。

时序产生器电路(T2,T3,T4与CLR,CLK之间的关系)：

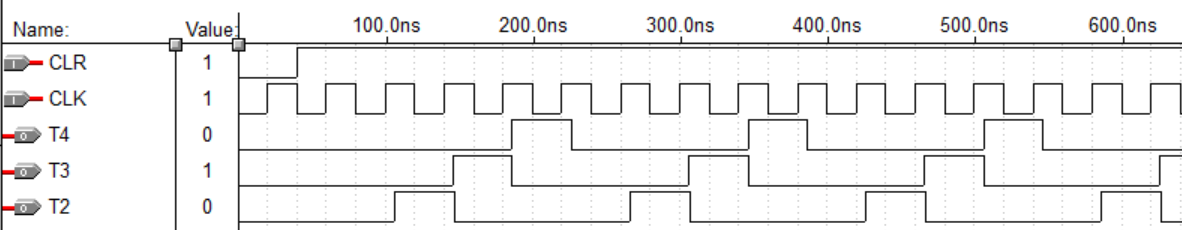


图4

# 六 汇编语言及其机器语言

算法思想：此程序采用了插入排序的思想，首先将数组元素的个数存储到寄存器R2中，然后从数组的第二个元素开始，依次将当前元素与前面已排序的元素进行比较。若当前元素小于前面已排序的元素，则将当前元素插入到合适的位置，并将后面的元素依次后移一位。继续进行下一个元素的比较和插入操作，直到所有元素都排序完成。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 汇编代码 | 地址(0x) | 机器代码(B) | 机器代码(H) | 注释 |
| MOV R0, 00H | 00 | 0010'00'00'00000000 | 1000 | R0<=0,数组指针 |
| DATA\_INPUT: IN R1 | 01 | 0001'00'01'00000000 | 0100 | R1<=INPUT |
| RM\_MOV R1, [R0] | 02 | 0011'01'00'00000000 | 2400 | 将 R1 的值传入 R0指针 指向的内存单元 |
| INC R0 | 03 | 1010'00'00'00000000 | 9000 | R0指针++ |
| MOV R2, 0AH | 04 | 0010'00'10'00001010 | 120A | R2<=0A,R2暂存数组大小10 |
| CMP R0, R2 | 05 | 1001'00'10'00000000 | 8200 | R0-R2 |
| JB DATA\_INPUT | 06 | 0110'00'00'00000001 | 6001 | 若R0<R2,继续输入 |
| SORT: MOV R0, 01H | 07 | 0010'00'00'00000001 | 1001 | R0看做 i |
| OUT\_LOOP: MR\_MOV [R0],R1 | 08 | 0100'00'01'00000000 | 3100 | R1看作TEMP,存储arr[i] |
| INNER: RR\_MOV R0, R2 | 09 | 0101'00'10'00000000 | 4200 | R2看作j 执行j = i |
| INNER\_LOOP: DEC R2 | 0A | 1011'00'10'00000000 | A200 | 执行j=j-1操作 此时R2是j-1，注意R2 0-1时的变化 |
| MOV R3, 00H | 0B | 0010'00'11'00000000 | 1300 |  |
| CMP R2, R3 | 0C | 1001'10'11'00000000 | 8B00 | 判断j-1（R2） >= 0 ？ |
| JL OUT\_LOOP\_NEXT | 0D | 0111'00'00'00010101 | 6015 | 若j-1 < 0 退出内层循环 执行有符号数的跳转操作 |
| MR\_MOV [R2], R3 | 0E | 0100'10'11'00000000 | 3B00 | R3 临时存储 arr[j-1] |
| CMP R1, R3 | 0F | 1001'01'11'00000000 | 8700 | 执行 arr[j]-arr[j-1] |
| JB OUT\_LOOP\_NEXT | 10 | 0110'00'00'00010101 | 6015 | arr[j] < arr[j-1] ,退出内层循环 |
| INC R2 | 11 | 1010'00'10'00000000 | 9200 | R2 恢复成 j |
| RM\_MOV R3, [R2] | 12 | 0011'11'10'00000000 | 2E00 | 执行 arr[j] = arr[j-1] |
| DEC R2 | 13 | 1011'00'10'00000000 | A200 | 执行内层循环中的j-- |
| JMP INNER\_LOOP | 14 | 1000'00'00'00001010 | 700A |  |
| OUT\_LOOP\_NEXT: INC R2 | 15 | 1010'00'10'00000000 | 9200 | 在内层循环跳转前将R2赋值为j-1,先恢复R2=j |
| RM\_MOV R1, [R2] | 16 | 0011'01'10'00000000 | 2600 | 执行 arr[j] = temp |
| INC R0 | 17 | 1010'00'00'00000000 | 9000 | 执行 i++ 操作 |
| MOV R3, 0AH | 18 | 0010'00'11'00001010 | 130A |  |
| CMP R0, R3 | 19 | 1001'00'11'00000000 | 8300 | 执行i - arr.length |
| JB OUT\_LOOP | 1A | 0110'00'00'00001000 | 6008 | 若i < arr.length 继续执行外层循环 |
| PRINT: MOV R0, 00H | 1B | 0010'00'00'00000000 | 1000 | R0指针 |
| PRINT\_LOOP:MR\_MOV [R0],R1 | 1C | 0100'00'01'00000000 | 3100 | 将指针所指向内存单元的内容赋值到R1中 |
| OUT R1 | 1D | 1100'01'00'00000000 | B400 | 输出R1的内容 |
| INC R0 | 1E | 1010'00'00'00000000 | 9000 | 指针指向下一个内存单元 |
| MOV R3, 0AH | 1F | 0010'00'11'00001010 | 130A |  |
| CMP R0, R3 | 20 | 1001'00'11'00000000 | 8300 |  |
| JB PRINT\_LOOP | 21 | 0110'00'00'00011100 | 601C |  |
| JMP PRINT | 22 | 1000'00'00'00011011 | 701B | 循环输出 |

表4

# 七 机器语言源程序的功能仿真波形图及结果分析

结果分析：

当外部清零控制信号CLR=1时，在外部时钟输入信号CLK的周期性变化下，输入10个数01，02，03，04，05，06，07，08，09，0A,一共耗时21.0us。

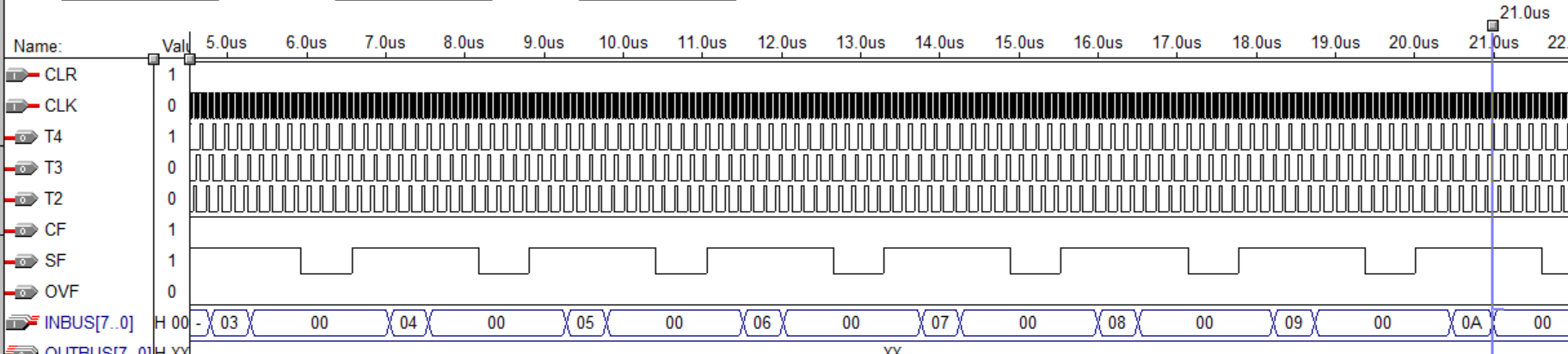


图5

可以看出到21.3us左右输入的数据全部存入到RAM中。

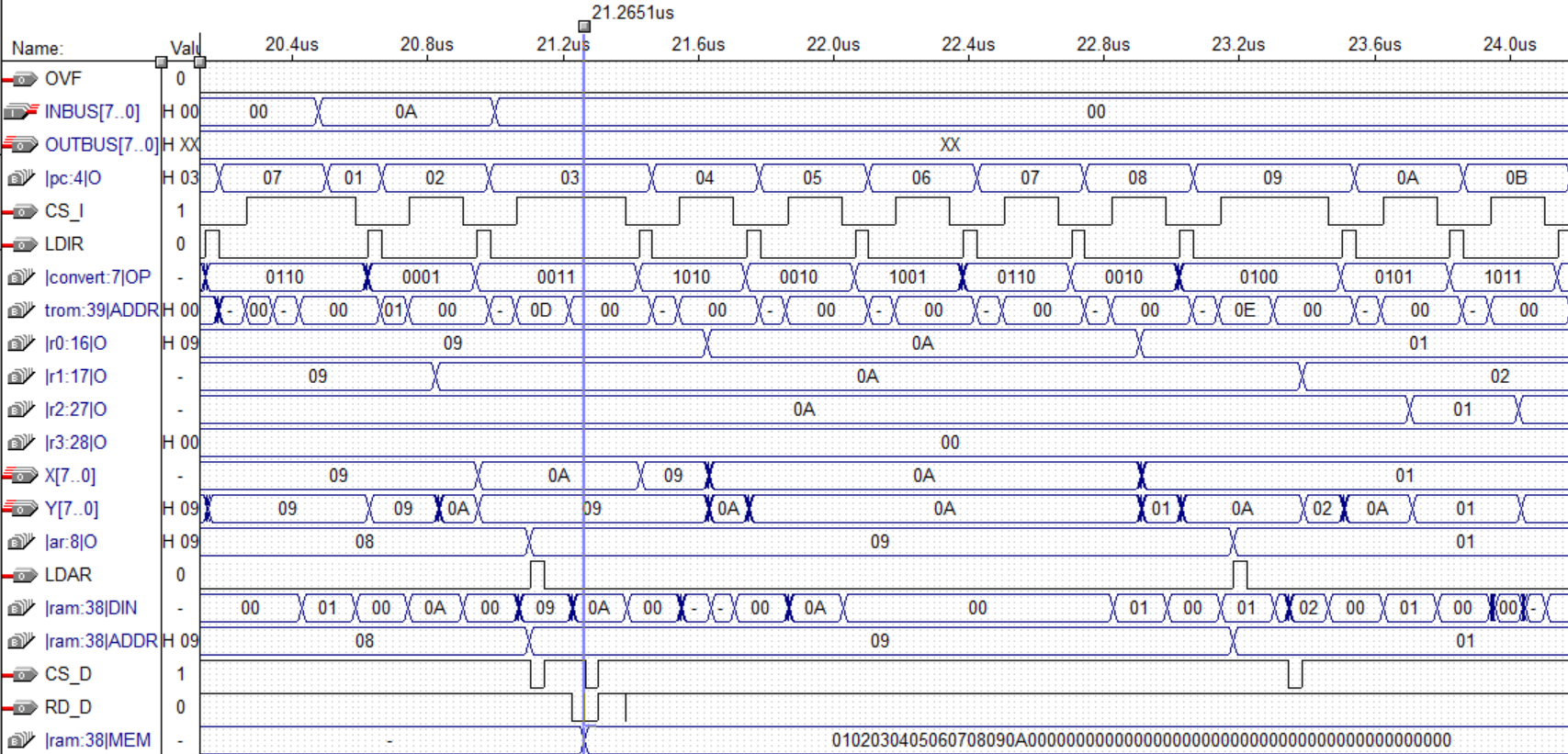
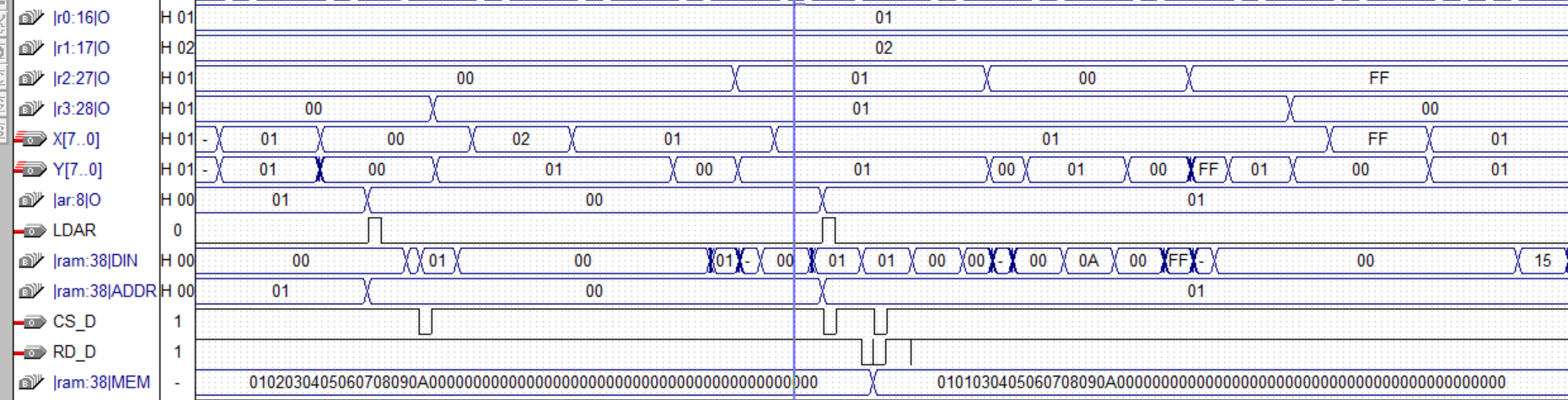


图6

输入完成后，开始执行插入排序：

第一趟排序波形图，将01与02进行比较，小的数字01放到后面，然后02替换到01的位置。



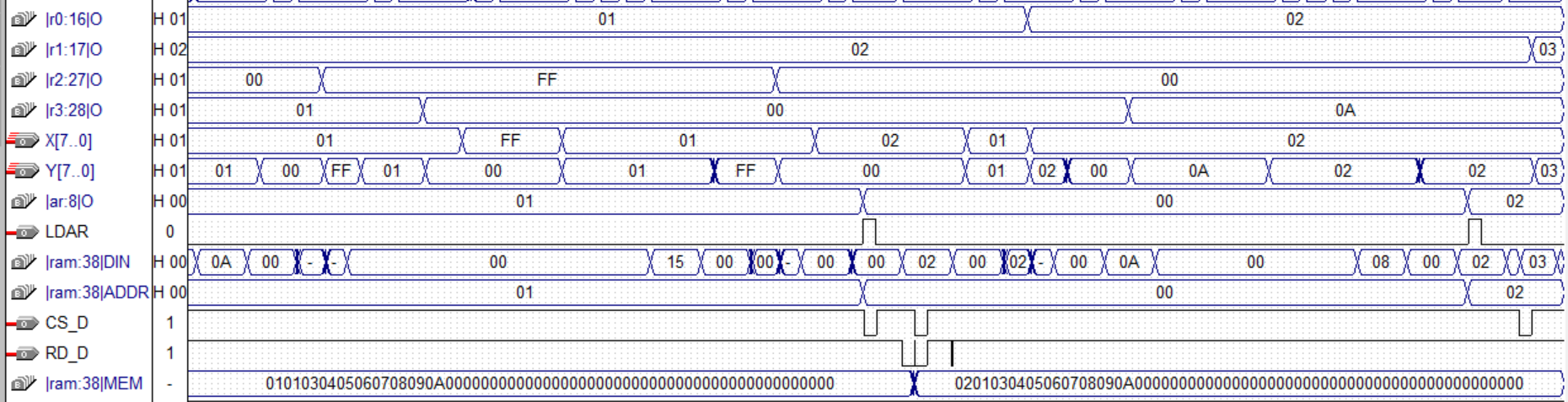


图7

第二趟

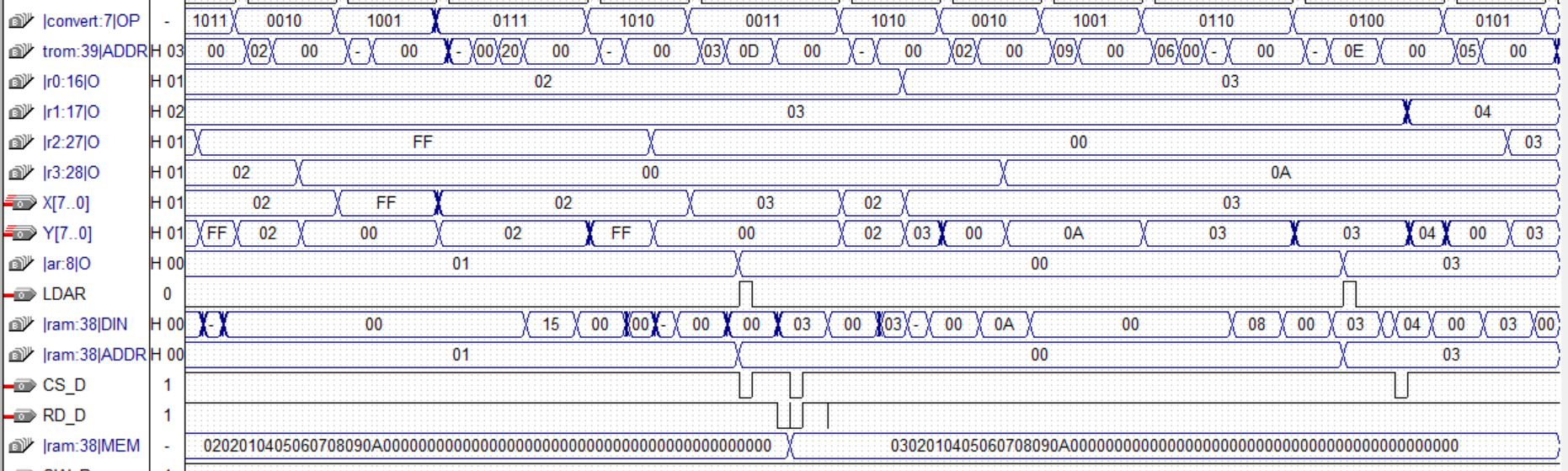


图8

第三趟

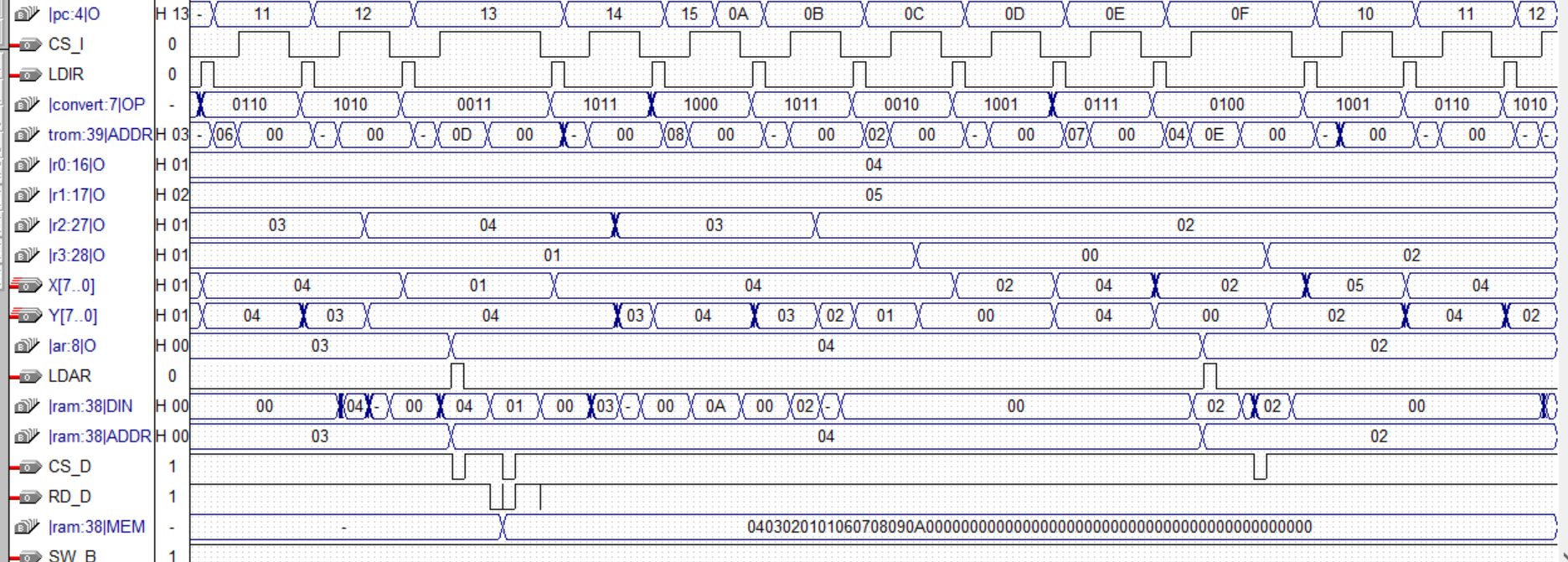


图9

第四趟



图10

第五趟

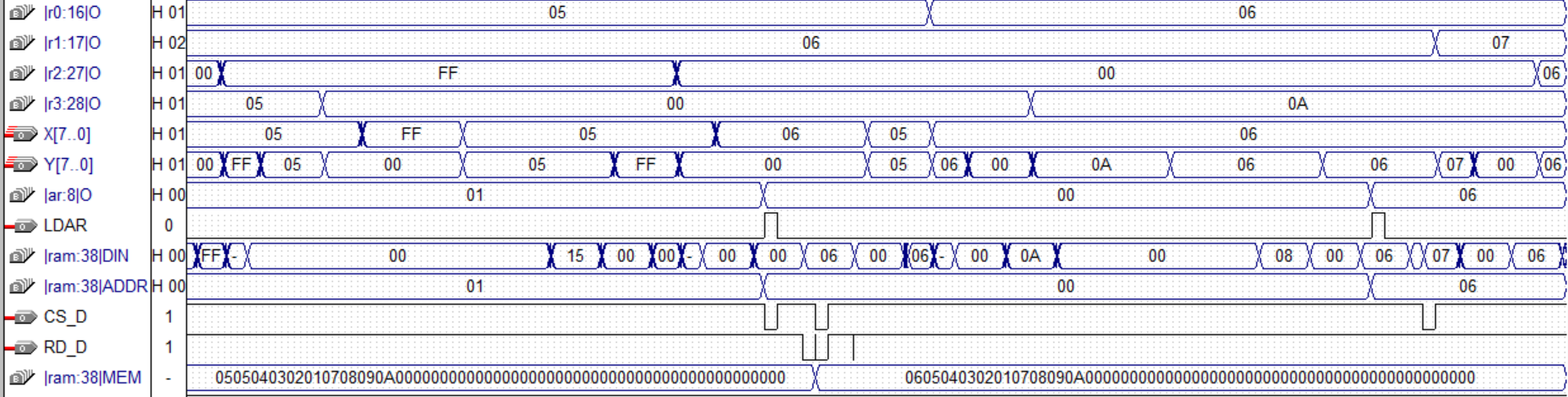


图11

第六趟

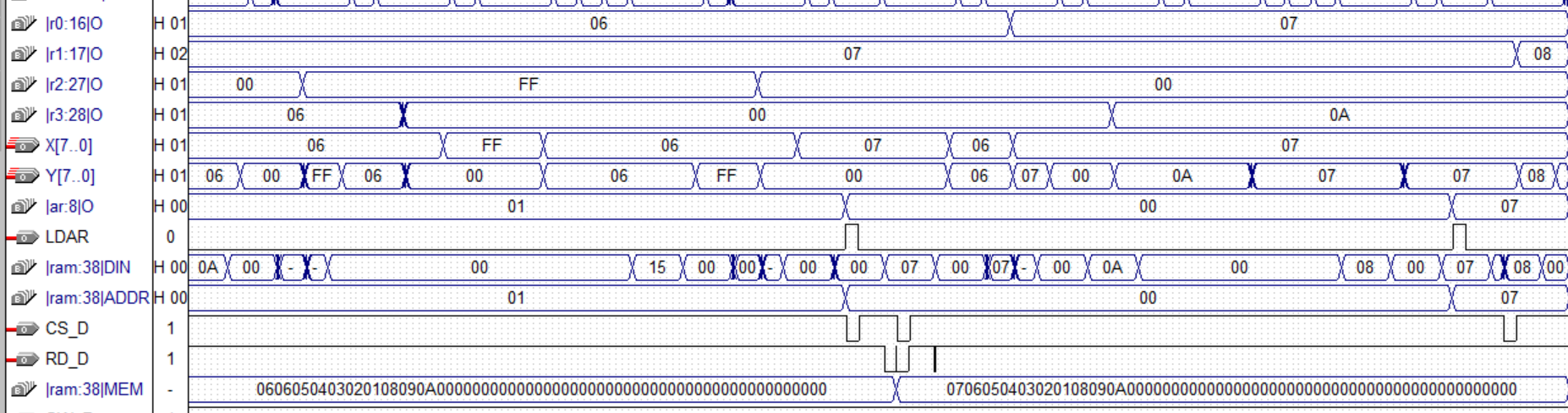


图12

第七趟

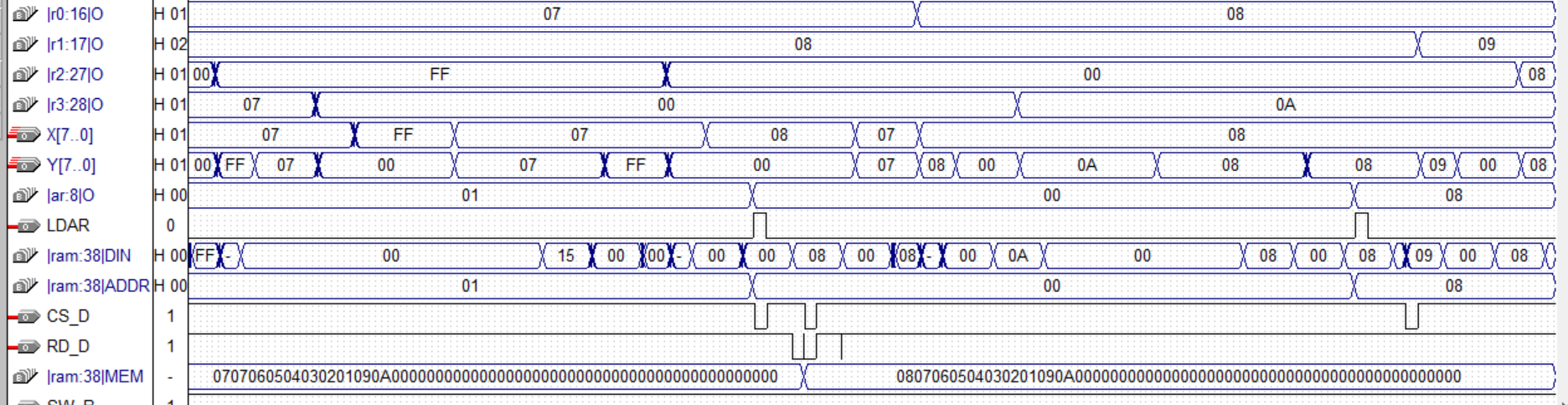


图13

第八趟

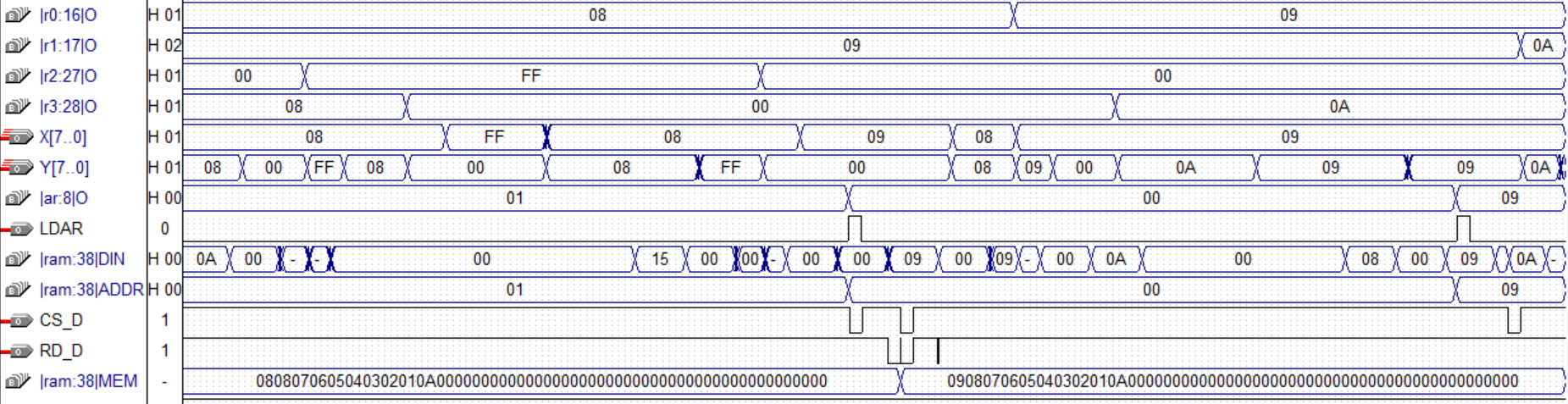


图14

第九趟

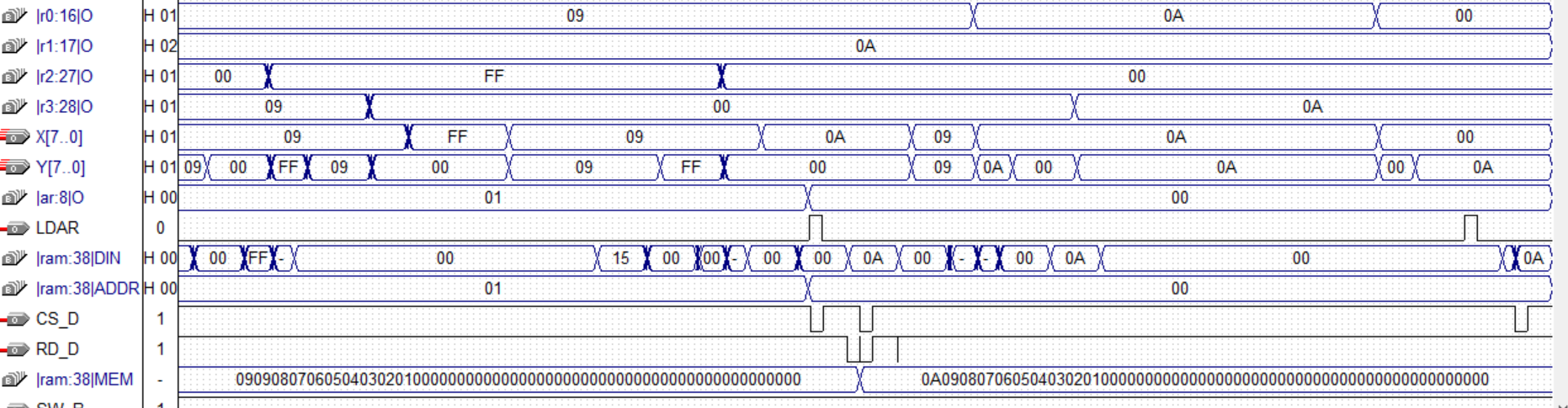


图15

最后排序的结果是0A,09,08,07,06,05,04,03,02,01，结果正确。

从图中可以得知程序在234.5479us时就已经完成排序。

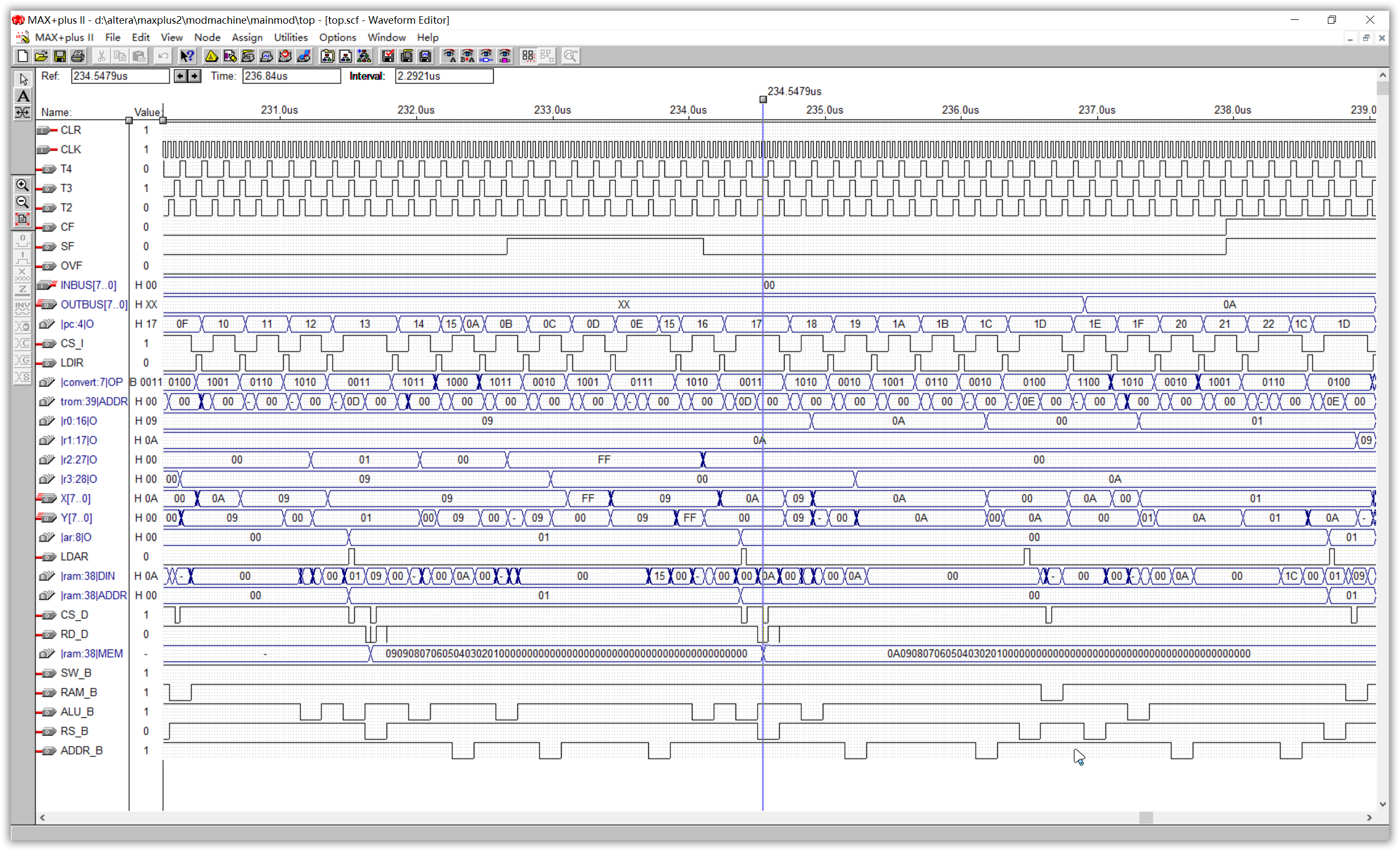


图16

# **八 故障现象和故障分析**

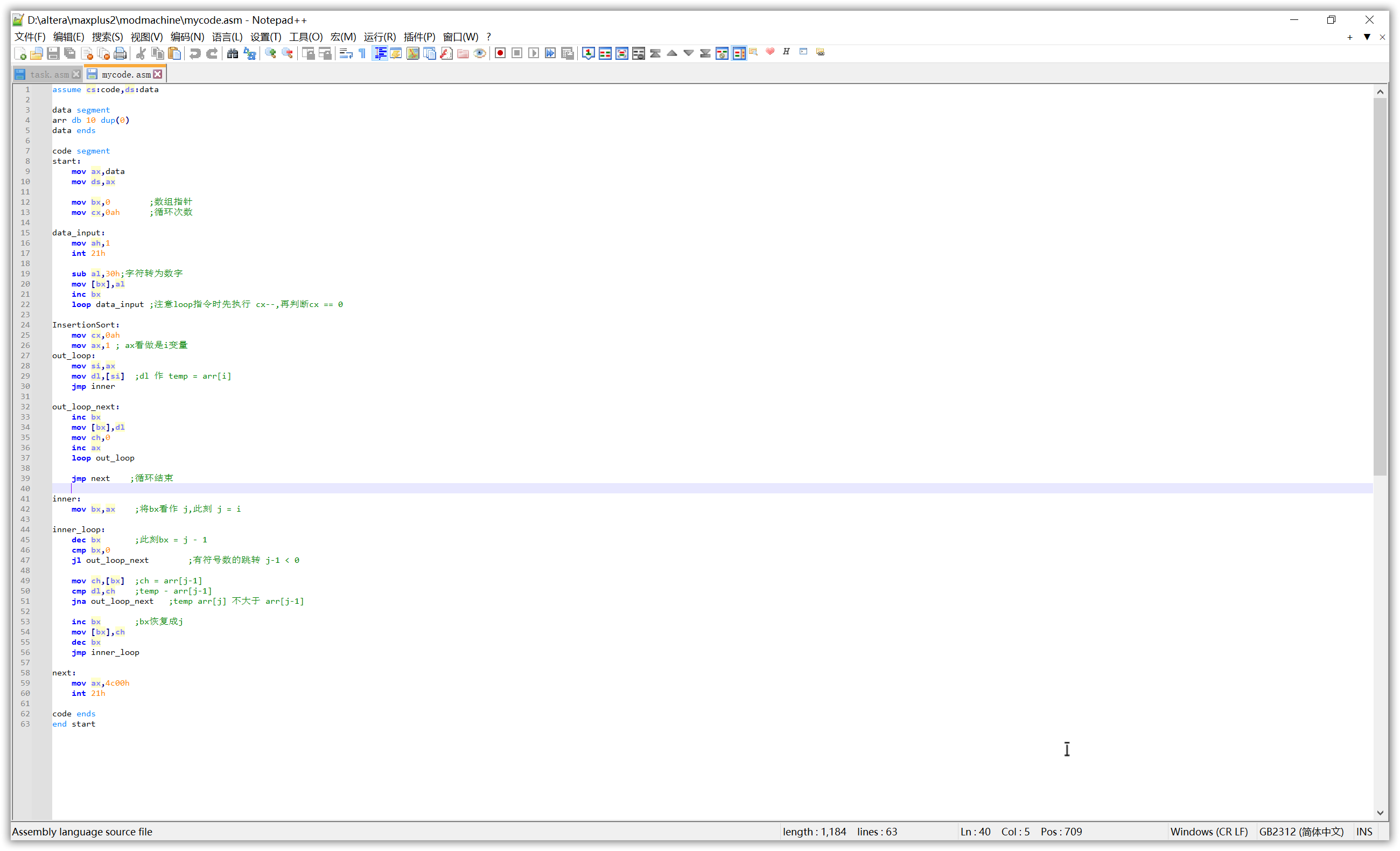
1. 刚开始编写汇编代码时，由于没有合适的运行环境，不清楚算法的正确与否，就先编写x86格式的汇编代码，然后在emu8086软件上试运行，发现结果正确。

图17

然后开始按照奥卡姆剃刀的原理，将x86汇编代码中的指令逐步精简并与本课设所需要的指令一一对应，最终完成的最精简的x86汇编代码，依然可以在emu8086软件上运行。

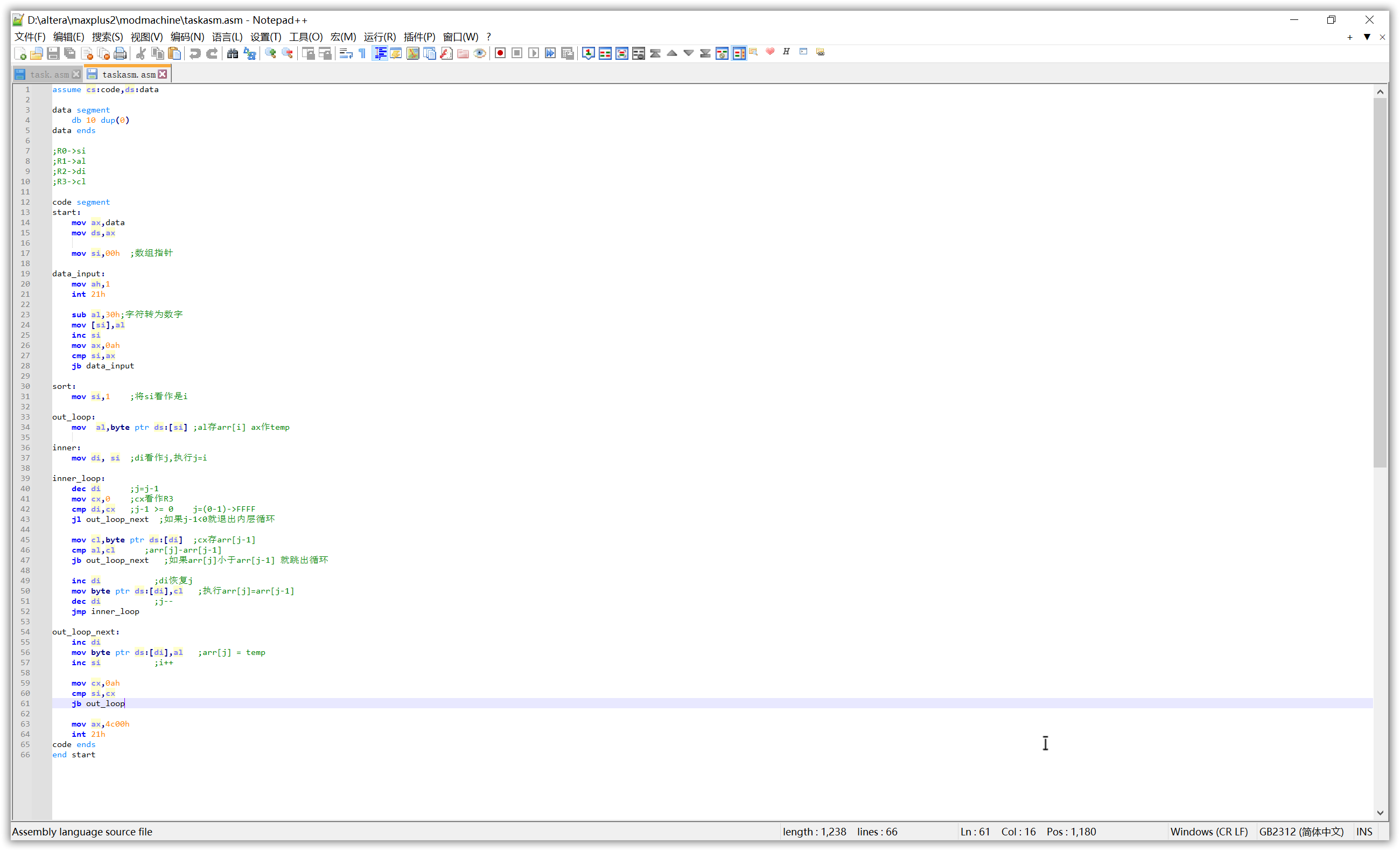


图18

最后按照本课设所设计的指令系统将它转化为最终我们需要的汇编代码。

2.可能是软件过于老旧的问题，有时修改一下某个元器件的vhdl代码后，不能成功编译得到所需要的新的symbol，这时需要找到本项目所在的文件目录下删除原先的symbol文件，再回到maxplus2软件上重新编译才得到新的symbol文件。

3. 在仿真图中，发现程序在执行jl指令时，不能跳转到指定的程序地址，此时需要再scf文件中添加影响jl指令执行的SF（符号标志位）和OVF（这个是溢出标志位，因为在vhd文件中OF是特殊符号不能使用，所以将它写成OVF），发现没问题，之后再添加CROM里面元件CONTROM的ADDR[5..0]端口信号（这个是微指令的后续微地址），发现它的值不是我们需要的JL的后续微地址的值，这时转到CROM的ADDR元件中，发现里边的地址表达式的计算结果出错，就重新设计一下逻辑表达式，按照Intel规定的x86中JL的逻辑判断来重新设计SF和OVF的使用，完成后重新编译ADDR，生成新的ADDR的symbol文件，并将CROM的ADDR元件更新，重新编译生成新的CROM的symbol文件，最后将top顶层电路图中的CROM的元件更新，在重新编译整个top顶层电路图，之后回到top.scf仿真文件重新仿真，得到正确的执行顺序结果了。

# 九 学习体会

本次实验是一次富有挑战性和收获的学术旅程。在这个实验中，我们着手设计一台嵌入式CISC模型计算机，积累了宝贵的经验和知识。以下是对这次实验的总结：

在实验过程中，我们深刻理解了计算机的内部结构和运作原理。从CPU的ALU到控制单元，从寄存器到内存，每个组成部分都在手中逐渐成形。通过亲手搭建这个计算机，我们更加清晰地认识到了硬件与软件之间的密切关系。

同时，问题解决能力在这个实验中得到了锻炼。每当遇到困难和错误时，我们都会仔细排查、分析问题，并通过实验和研究找到解决方法。这种能力在计算机领域的未来职业中将是无价之宝。

团队合作是这次实验很重要的一部分。与同学一起合作完成任务，我们互相支持，分享经验，共同完成了这项复杂的任务。这使我们的团队协作和沟通技能得到进一步的提升，让我们明白了在集体智慧的协作下，可以创造出更加出色的成果。

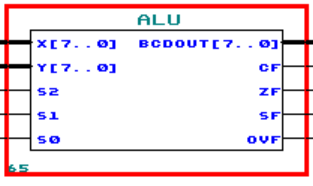
设计优化是本次实验中的另一项收获。通过不断尝试不同的设计方案，优化硬件和指令集，来提高计算机的性能和效率。这有效培养了创新思维和工程设计能力。

最后，这次实验使我们更加了解了嵌入式计算机系统的设计和应用，为今后的学术和职业生涯提供了坚实的基础。使我们明白了理论知识与实际应用的紧密联系，也增强了对计算机科学的热情。

综上所述，本次实验是一次富有挑战性和收获的学术经历。通过这个实验，我们不仅深化了对计算机原理的理解，还提高了问题解决能力、团队协作技能和设计优化能力。这将对我们的未来发展产生积极的影响，很期待能将这些经验和知识应用到未来的学术和职业生涯中。

# 十 软件清单

## 10.1 ALU单元



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_SIGNED.ALL;

ENTITY ALU IS

PORT(

X: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

Y: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

S2,S1,S0: IN STD\_LOGIC;

BCDOUT: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);

CF,ZF,SF,OVF: OUT STD\_LOGIC

);

END ALU;

ARCHITECTURE A OF ALU IS

SIGNAL AA,BB,TEMP: STD\_LOGIC\_VECTOR(8 DOWNTO 0);

SIGNAL TEMP1: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

PROCESS

BEGIN

TEMP <="000000000";

IF (S2='0' AND S1='0' AND SO='0')THEN --执行加法操作

NULL;

ELSIF (S2='0' AND S1='0' AND S0='1') THEN --CMP

AA <= '0'&X;

BB <= '0'&Y;

TEMP <= AA-BB;

--BCDOUT <= TEMP(7 DOWNTO 0);

SF <= TEMP(7);

CF <= TEMP(8);

--两数相减，同号，则不溢出;

--两数为异号，结果与减数符号相同，则溢出。

IF (AA(7) = BB(7)) THEN

OVF <= '0';

ELSE

IF(TEMP(7) = BB(7)) THEN

OVF <= '1';

ELSE

OVF <= '0';

END IF;

END IF;

ELSIF (S2='0' AND S1='1' AND S0='0') THEN --INC

AA <= '0'&Y;

TEMP <= AA+1;

BCDOUT <= TEMP(7 DOWNTO 0);

SF <= TEMP(7);

IF(TEMP(7 DOWNTO 0)="00000000") THEN

ZF <='1';

ELSE

ZF <='0';

END IF;

--正正得负则溢出，负负得正则溢出

IF(AA(7) ='1') THEN --负数+1不溢出

OVF <= '0';

ELSE

IF(TEMP(7) = '1') THEN --正数+正数 结果 为 负 溢出

OVF <= '1';

ELSE

OVF <= '0';

END IF;

END IF;

ELSIF (S2='0' AND S1='1' AND S0='1') THEN --DEC

AA <= '0'&Y;

TEMP <= AA-1;

BCDOUT <= TEMP(7 DOWNTO 0);

IF(TEMP(7 DOWNTO 0)="00000000") THEN

ZF <='1';

ELSE

ZF <='0';

END IF;

SF <= TEMP(7);

IF(AA(7)='0') THEN --两数相减，同号，则不溢出

OVF <= '0';

ELSE --两数为异号，结果与减数符号相同，则溢出。

IF(TEMP(7) = '0') THEN

OVF <= '1';

ELSE

OVF <= '0';

END IF;

END IF;

ELSIF(S2='1' AND S1='1' AND S0='1') THEN --Y 直接输出Rd

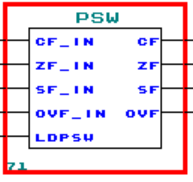
BCDOUT <= Y;

END IF;

END PROCESS;

END A;

## 10.2 状态条件寄存器PSW



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY PSW IS

PORT(

CF\_IN,ZF\_IN, SF\_IN, OVF\_IN: IN STD\_LOGIC;

CF,ZF, SF, OVF: OUT STD\_LOGIC;

LDPSW: IN STD\_LOGIC

);

END PSW;

ARCHITECTURE A OF PSW IS

BEGIN

PROCESS(LDPSW)

BEGIN

IF (LDPSW'EVENT AND LDPSW='1') THEN

CF <= CF\_IN;

ZF <= ZF\_IN;

SF <= SF\_IN;

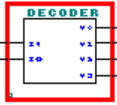
OVF <= OVF\_IN;

END IF;

END PROCESS;

END A;

## 10.3 DECODER



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY DECODER IS

PORT(

I9,I8:IN STD\_LOGIC;

Y0,Y1,Y2,Y3:OUT STD\_LOGIC

);

END DECODER;

ARCHITECTURE A OF DECODER IS

BEGIN

PROCESS

BEGIN

IF(I9='0' AND I8='0')THEN

Y0<='1';

Y1<='0';

Y2<='0';

Y3<='0';

ELSIF(I9='0' AND I8='1')THEN

Y0<='0';

Y1<='1';

Y2<='0';

Y3<='0';

ELSIF(I9='1' AND I8='0')THEN

Y0<='0';

Y1<='0';

Y2<='1';

Y3<='0';

ELSE

Y0<='0';

Y1<='0';

Y2<='0';

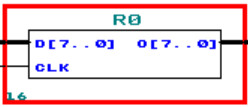
Y3<='1';

END IF;

END PROCESS;

END A;

## 10.4 通用寄存器



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY R0 IS

PORT(

D: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

CLK: IN STD\_LOGIC;

O: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)

);

END R0;

ARCHITECTURE A OF R0 IS

BEGIN

PROCESS(CLK)

BEGIN

IF(CLK'EVENT AND CLK='1') THEN

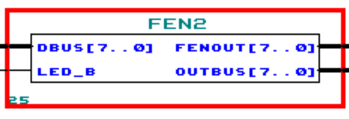
O<=D;

END IF;

END PROCESS;

END A;

## 10.5 1:2分配器FEN2



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY FEN2 IS

PORT(

DBUS: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

LED\_B: IN STD\_LOGIC;

FENOUT,OUTBUS: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)

);

END FEN2;

ARCHITECTURE A OF FEN2 IS

BEGIN

PROCESS

BEGIN

IF(LED\_B='0')THEN

OUTBUS<=DBUS;

ELSE

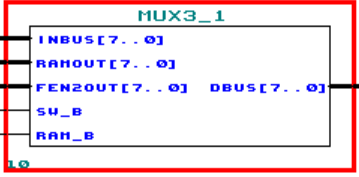
FENOUT<=DBUS;

END IF;

END PROCESS;

END A;

## 10.6 3选1数据选择器MUX3\_1



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY MUX3\_1 IS

PORT(

INBUS,RAMOUT,FEN2OUT: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

SW\_B,RAM\_B: IN STD\_LOGIC;

DBUS: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));

END MUX3\_1;

ARCHITECTURE A OF MUX3\_1 IS

BEGIN

PROCESS

BEGIN

IF(SW\_B='0') THEN

DBUS<=INBUS;

ELSE

IF(RAM\_B='0') THEN

DBUS<=RAMOUT;

ELSE

DBUS<=FEN2OUT;

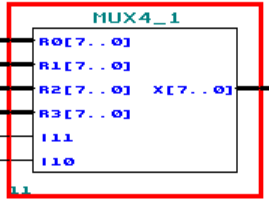
END IF;

END IF;

END PROCESS;

END A;

## 10.7 4选1数据选择器MUX4\_1



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY MUX4\_1 IS

PORT(

R0,R1,R2,R3: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

I11,I10: IN STD\_LOGIC;

X: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));

END MUX4\_1;

ARCHITECTURE A OF MUX4\_1 IS

BEGIN

PROCESS

BEGIN

IF(I11='0' AND I10='0') THEN

X<=R0;

ELSIF(I11='0' AND I10='1') THEN

X<=R1;

ELSIF(I11='1' AND I10='0') THEN

X<=R2;

ELSE

X<=R3;

END IF;

END PROCESS;

END A;

## 10.8 程序计数器PC



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_SIGNED.ALL;

ENTITY PC IS

PORT(

LOAD,LDPC,CLR:IN STD\_LOGIC;

D: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

O: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));

END PC;

ARCHITECTURE A OF PC IS

SIGNAL QOUT: STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

PROCESS(LDPC,CLR,LOAD)

BEGIN

IF(CLR='0') THEN

QOUT<="00000000";

ELSIF(LDPC'EVENT AND LDPC='1') THEN

IF(LOAD='0')THEN

QOUT<=D; --BUS->PC

ELSE

QOUT<=QOUT+1; --PC+1

END IF;

--不装入,也不计数

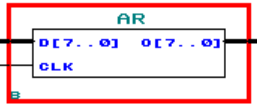
END IF;

END PROCESS;

O<=QOUT;

END A;

## 10.9 地址寄存器AR



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY AR IS

PORT(

D: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

CLK: IN STD\_LOGIC;

O: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)

);

END AR;

ARCHITECTURE A OF AR IS

BEGIN

PROCESS(CLK)

BEGIN

IF(CLK'EVENT AND CLK='1') THEN

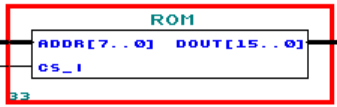
O<=D;

END IF;

END PROCESS;

END A;

## 10.10 主存储器ROM



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_SIGNED.ALL;

ENTITY ROM IS

PORT(

DOUT: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0);

ADDR: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

CS\_I: IN STD\_LOGIC

);

END ROM;

ARCHITECTURE A OF ROM IS

BEGIN

DOUT <= --CS\_I 低电平

"0010000000000000" WHEN ADDR="00000000" AND CS\_I='0' ELSE

"0001000100000000" WHEN ADDR="00000001" AND CS\_I='0' ELSE

"0011010000000000" WHEN ADDR="00000010" AND CS\_I='0' ELSE

"1010000000000000" WHEN ADDR="00000011" AND CS\_I='0' ELSE

"0010001000001010" WHEN ADDR="00000100" AND CS\_I='0' ELSE

"1001001000000000" WHEN ADDR="00000101" AND CS\_I='0' ELSE

"0110000000000001" WHEN ADDR="00000110" AND CS\_I='0' ELSE

"0010000000000001" WHEN ADDR="00000111" AND CS\_I='0' ELSE

"0100000100000000" WHEN ADDR="00001000" AND CS\_I='0' ELSE

"0101001000000000" WHEN ADDR="00001001" AND CS\_I='0' ELSE

"1011001000000000" WHEN ADDR="00001010" AND CS\_I='0' ELSE

"0010001100000000" WHEN ADDR="00001011" AND CS\_I='0' ELSE

"1001101100000000" WHEN ADDR="00001100" AND CS\_I='0' ELSE

"0111000000010101" WHEN ADDR="00001101" AND CS\_I='0' ELSE

"0100101100000000" WHEN ADDR="00001110" AND CS\_I='0' ELSE

"1001011100000000" WHEN ADDR="00001111" AND CS\_I='0' ELSE

"0110000000010101" WHEN ADDR="00010000" AND CS\_I='0' ELSE

"1010001000000000" WHEN ADDR="00010001" AND CS\_I='0' ELSE

"0011111000000000" WHEN ADDR="00010010" AND CS\_I='0' ELSE

"1011001000000000" WHEN ADDR="00010011" AND CS\_I='0' ELSE

"1000000000001010" WHEN ADDR="00010100" AND CS\_I='0' ELSE

"1010001000000000" WHEN ADDR="00010101" AND CS\_I='0' ELSE

"0011011000000000" WHEN ADDR="00010110" AND CS\_I='0' ELSE

"1010000000000000" WHEN ADDR="00010111" AND CS\_I='0' ELSE

"0010001100001010" WHEN ADDR="00011000" AND CS\_I='0' ELSE

"1001001100000000" WHEN ADDR="00011001" AND CS\_I='0' ELSE

"0110000000001000" WHEN ADDR="00011010" AND CS\_I='0' ELSE

"0010000000000000" WHEN ADDR="00011011" AND CS\_I='0' ELSE

"0100000100000000" WHEN ADDR="00011100" AND CS\_I='0' ELSE

"1100010000000000" WHEN ADDR="00011101" AND CS\_I='0' ELSE

"1010000000000000" WHEN ADDR="00011110" AND CS\_I='0' ELSE

"0010001100001010" WHEN ADDR="00011111" AND CS\_I='0' ELSE

"1001001100000000" WHEN ADDR="00100000" AND CS\_I='0' ELSE

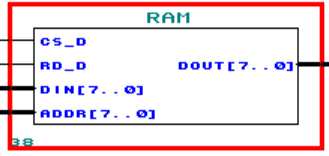
"0110000000011100" WHEN ADDR="00100001" AND CS\_I='0' ELSE

"1000000000011011" WHEN ADDR="00100010" AND CS\_I='0' ELSE

"0000000000000000";

END A;

## 10.11 RAM



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY RAM IS

PORT(

CS\_D,RD\_D: IN STD\_LOGIC;

DIN: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

ADDR: IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);

DOUT: OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0));

END RAM;

ARCHITECTURE A OF RAM IS

TYPE MEMORY IS ARRAY(0 TO 31) OF STD\_LOGIC\_VECTOR(7 DOWNTO 0);

BEGIN

PROCESS(CS\_D)

VARIABLE MEM:MEMORY;

BEGIN

IF(CS\_D'EVENT AND CS\_D='0') THEN

IF(RD\_D='0') THEN

MEM(CONV\_INTEGER(ADDR(4 DOWNTO 0))):=DIN;

ELSE

DOUT<=MEM(CONV\_INTEGER(ADDR(4 DOWNTO 0)));

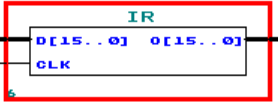
END IF;

END IF;

END PROCESS;

END A;

## 10.12 指令寄存器IR



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY IR IS

PORT(

D: IN STD\_LOGIC\_VECTOR(15 DOWNTO 0);

CLK: IN STD\_LOGIC;

O: OUT STD\_LOGIC\_VECTOR(15 DOWNTO 0)

);

END IR;

ARCHITECTURE A OF IR IS

BEGIN

PROCESS(CLK)

BEGIN

IF(CLK'EVENT AND CLK='1') THEN

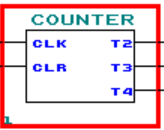
O<=D;

END IF;

END PROCESS;

END A;

## 10.13 时序产生器COUNTER



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_SIGNED.ALL;

ENTITY COUNTER IS

PORT(

CLK,CLR: IN STD\_LOGIC;

T2,T3,T4: OUT STD\_LOGIC

);

END COUNTER;

ARCHITECTURE A OF COUNTER IS

SIGNAL X:STD\_LOGIC\_VECTOR(1 DOWNTO 0);

BEGIN

PROCESS(CLK,CLR)

BEGIN

IF(CLR='0') THEN

T2<='0';

T3<='0';

T4<='0';

X<="00";

ELSIF(CLK'EVENT AND CLK='1') THEN

X<=X+1;

T2<=(NOT X(1)) AND X(0);

T3<=X(1) AND (NOT X(0));

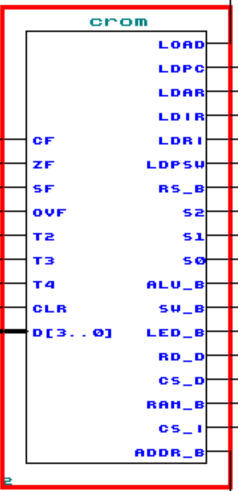
T4<=X(1) AND X(0);

END IF;

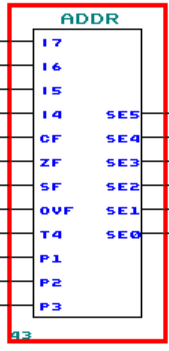
END PROCESS;

END A;

## 10.14 微程序控制器CROM



## 10.15 地址转移逻辑电路ADDR



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY ADDR IS

PORT(

I7,I6,I5,I4: IN STD\_LOGIC;

CF,ZF,SF,OVF,T4,P1,P2,P3: IN STD\_LOGIC;

SE5,SE4,SE3,SE2,SE1,SE0: OUT STD\_LOGIC

);

END ADDR;

ARCHITECTURE A OF ADDR IS

BEGIN

SE5 <=NOT ((NOT(SF) AND OVF AND P3 AND T4) OR (SF AND NOT(OVF) AND P3 AND T4));

SE4 <=NOT (CF AND P2 AND T4);

SE3 <=NOT (I7 AND P1 AND T4);

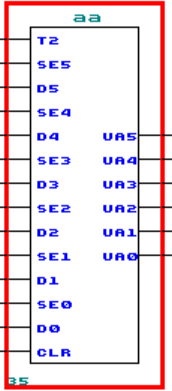
SE2 <=NOT (I6 AND P1 AND T4);

SE1 <=NOT (I5 AND P1 AND T4);

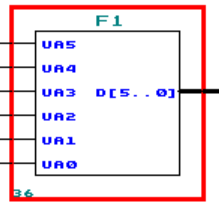
SE0 <=NOT (I4 AND P1 AND T4);

END A;

## 10.16 微地址寄存器aa



## 10.17 微地址转换器F1



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY F1 IS

PORT(

UA5,UA4,UA3,UA2,UA1,UA0:IN STD\_LOGIC;

D:OUT STD\_LOGIC\_VECTOR(5 DOWNTO 0)

);

END F1;

ARCHITECTURE A OF F1 IS

BEGIN

D(5)<=UA5;

D(4)<=UA4;

D(3)<=UA3;

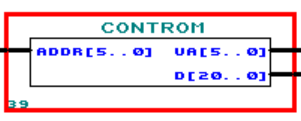
D(2)<=UA2;

D(1)<=UA1;

D(0)<=UA0;

END A;

## 10.18 控制存储器COUNTROM



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_SIGNED.ALL;

ENTITY CONTROM IS

PORT(

ADDR: IN STD\_LOGIC\_VECTOR(5 DOWNTO 0);--地址

UA: OUT STD\_LOGIC\_VECTOR(5 DOWNTO 0);--后续微地址

D: OUT STD\_LOGIC\_VECTOR(20 DOWNTO 0));--微命令

END CONTROM;

ARCHITECTURE A OF CONTROM IS

SIGNAL DATAOUT: STD\_LOGIC\_VECTOR(26 DOWNTO 0);

BEGIN

PROCESS(ADDR)

BEGIN

CASE ADDR IS

WHEN "000000"=>DATAOUT<="110100100011111101100000000"; --00H 取指

WHEN "000001"=>DATAOUT<="100010100010111011000000000"; --01H IN

WHEN "000010"=>DATAOUT<="100010100011111110000000000"; --02H MOV

WHEN "000011"=>DATAOUT<="101000111101110111000001101"; --03H RM\_MOV

WHEN "000100"=>DATAOUT<="101000000011111111000001110"; --04H MR\_MOV

WHEN "000101"=>DATAOUT<="100010000011111111000000000"; --05H RR\_MOV

WHEN "000110"=>DATAOUT<="100000100011111111010000000"; --06H JA

WHEN "000111"=>DATAOUT<="100000100011111111001000000"; --07H JL

WHEN "001000"=>DATAOUT<="010000100011111110000000000"; --08H JMP

WHEN "001001"=>DATAOUT<="100001100111111111000000000"; --09H CMP

WHEN "001010"=>DATAOUT<="100011101001111111000000000"; --0AH INC

WHEN "001011"=>DATAOUT<="100011101101111111000000000"; --0BH DEC

WHEN "001100"=>DATAOUT<="100000000011011111000000000"; --0CH OUT

WHEN "001101"=>DATAOUT<="100000000011100111000000000"; --0DH RM\_MOV(2)

WHEN "001110"=>DATAOUT<="100010100011110011000000000"; --0EH MR\_MOV(2)

WHEN "010000"=>DATAOUT<="010000100011111110000000000"; --10H JA(2)

WHEN "100000"=>DATAOUT<="010000100011111110000000000"; --20H JL(2)

WHEN OTHERS=>DATAOUT<="100000100011111111000000000";

END CASE;

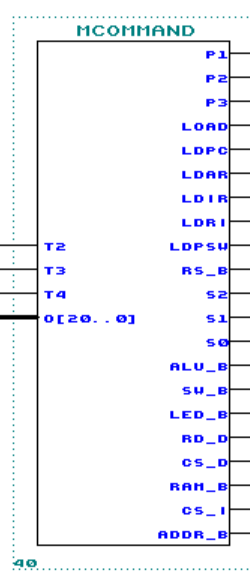
UA(5 DOWNTO 0)<=DATAOUT(5 DOWNTO 0);

D(20 DOWNTO 0)<=DATAOUT(26 DOWNTO 6);

END PROCESS;

END A;

## 10.19 微命令寄存器MCOMMAN



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY MCOMMAND IS

PORT(

T2,T3,T4: IN STD\_LOGIC;

O: IN STD\_LOGIC\_VECTOR(20 DOWNTO 0);

P1,P2,P3,LOAD,LDPC,LDAR,LDIR,LDRI,LDPSW,RS\_B,S2,S1,S0,ALU\_B,SW\_B,LED\_B,RD\_D,CS\_D,RAM\_B,CS\_I,ADDR\_B: OUT STD\_LOGIC);

END MCOMMAND;

ARCHITECTURE A OF MCOMMAND IS

SIGNAL DATAOUT: STD\_LOGIC\_VECTOR(20 DOWNTO 0);

BEGIN

PROCESS(T2)

BEGIN

IF(T2'EVENT AND T2='1') THEN

DATAOUT(20 DOWNTO 0)<=O(20 DOWNTO 0);

END IF;

P3<=DATAOUT(0);

P2<=DATAOUT(1);

P1<=DATAOUT(2);

ADDR\_B<=DATAOUT(3);

CS\_I<=DATAOUT(4);

RAM\_B<=DATAOUT(5);

CS\_D<=NOT(NOT DATAOUT(6) AND T3);

RD\_D<=NOT(NOT DATAOUT(7) AND (T2 OR T3));

LED\_B<=DATAOUT(8);

SW\_B<=DATAOUT(9);

ALU\_B<=DATAOUT(10);

S0<=DATAOUT(11);

S1<=DATAOUT(12);

S2<=DATAOUT(13);

RS\_B<=DATAOUT(14);

LDPSW<=DATAOUT(15) AND T4;

LDRI<=DATAOUT(16) AND T4;

LDIR<=DATAOUT(17) AND T3;

LDAR<=DATAOUT(18) AND T3;

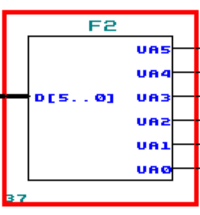
LDPC<=DATAOUT(19) AND T4;

LOAD<=DATAOUT(20);

END PROCESS;

END A;

## 10.20 微地址转换器F2



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY F2 IS

PORT(

D:IN STD\_LOGIC\_VECTOR(5 DOWNTO 0);

UA5,UA4,UA3,UA2,UA1,UA0:OUT STD\_LOGIC

);

END F2;

ARCHITECTURE A OF F2 IS

BEGIN

UA5<=D(5);

UA4<=D(4);

UA3<=D(3);

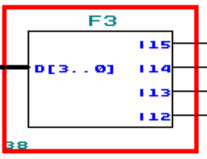
UA2<=D(2);

UA1<=D(1);

UA0<=D(0);

END A;

## 10.21 指令代码转换器F3



LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY F3 IS

PORT(

D:IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

I15,I14,I13,I12:OUT STD\_LOGIC

);

END F3;

ARCHITECTURE A OF F3 IS

BEGIN

I15<=D(3);

I14<=D(2);

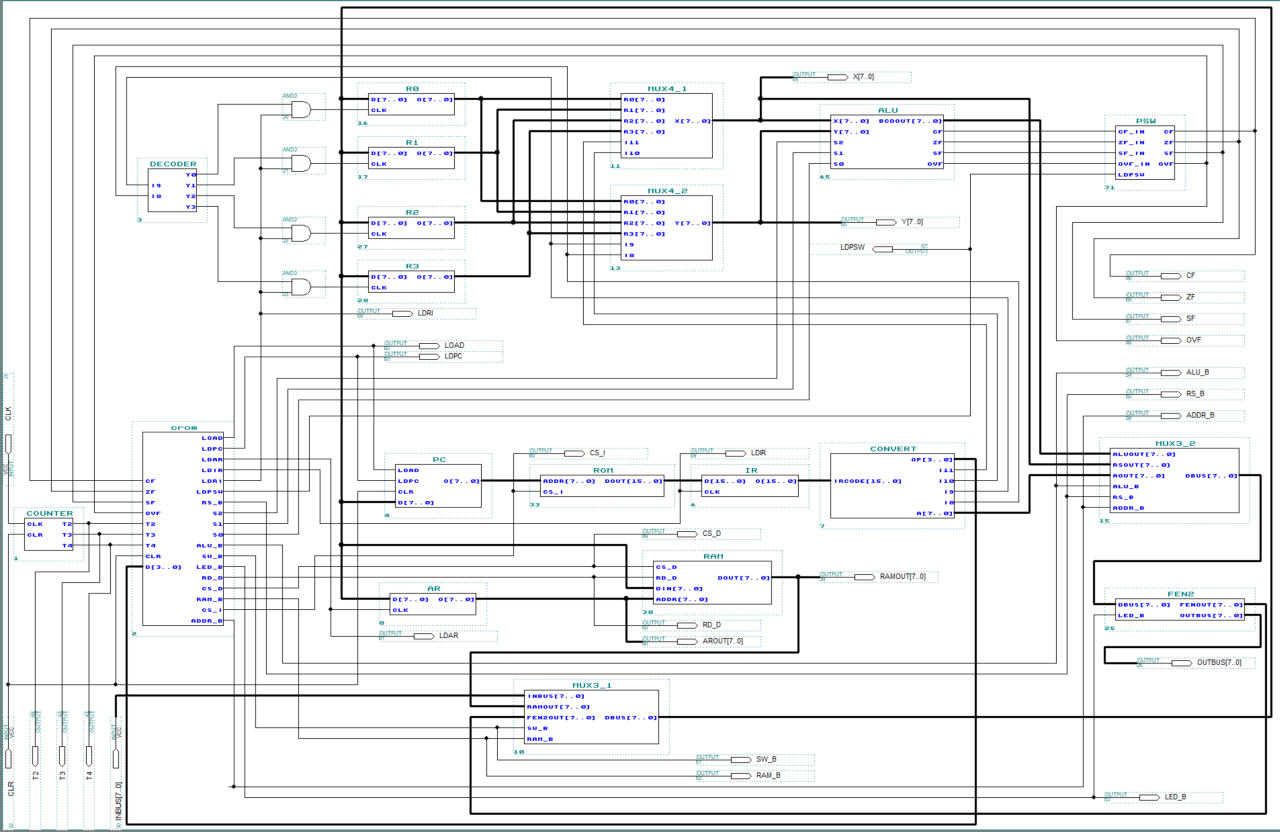
I13<=D(1);

I12<=D(0);

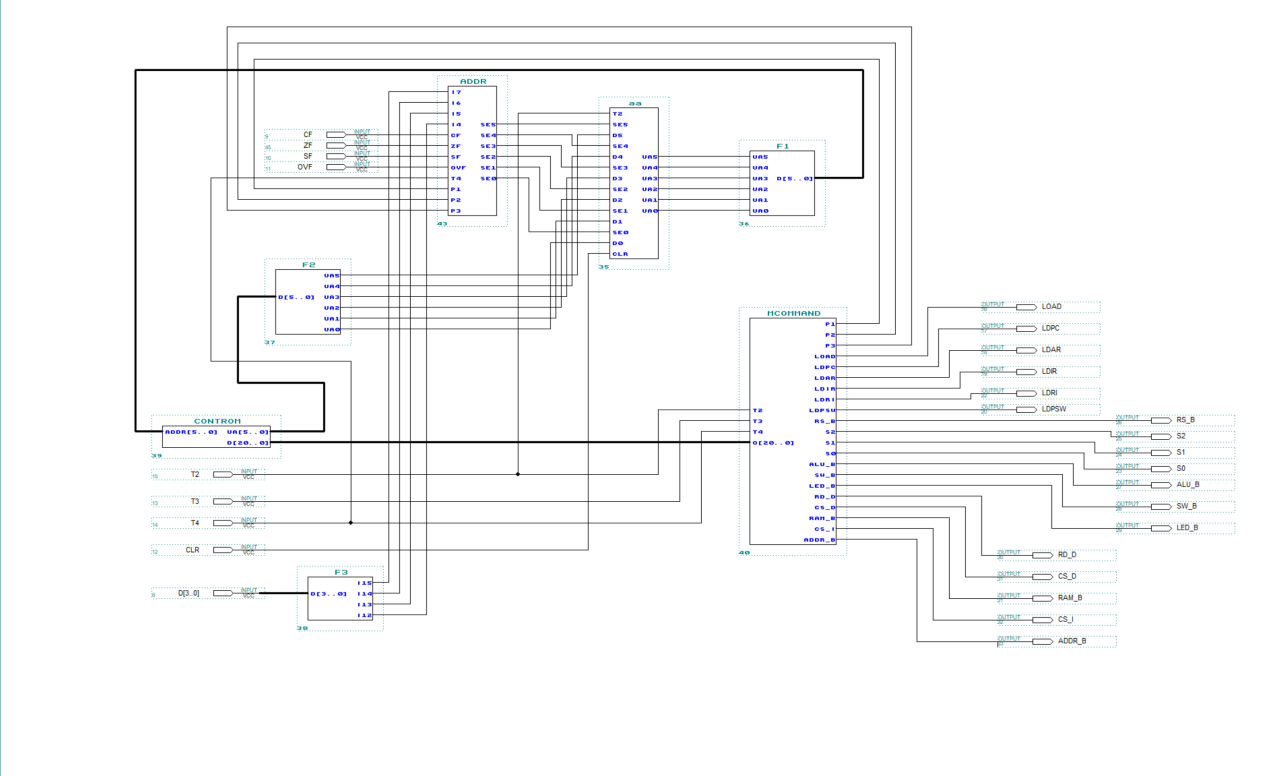
END A;

# 附录

附录A 顶层电路图



附录B crom内部电路图



附录C aa内部电路图：

