微机原理与应用实验报告

简单计算机系统的设计与实现

第二部分

班级：电 92

姓名：曲昊源

学号：2009010939

实验 3 简单计算机系统的设计与实现

一、 实验目的

1. 了解计算机的组成与各部件的功能；
2. 熟悉简单计算机的指令集，学习编写汇编语言程序和机器码程序；
3. 熟悉各类型指令执行的数据通道；
4. 设计一个 8 位单周期简单计算机系统。

二、 实验任务

1. 编程练习

先采用汇编语言格式编写程序，检查程序的思路、流程，在无误情况下，转换成机器码。

程序 1：

完成将两个固定数据（如 0x95, 0x35E）进行加、减、与、或、比较运算，将结果顺序存放在地址分别为 0x20~0x21、0x22~0x23、0x24~0x25、0x26~0x27、0x28~0x29 的 10 个 RAM 单元中。

表 1 程序 1 指令编码

编号	汇编语言格式命令	二进制机器码指令	十六进制机器码指令	功能
0	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
1	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
2	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
3	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
4	ADDI R0,R0,0x95	1010 0000 1001 0101	A095	将 0x95 的低 8 位写入 R0
5	ADDI R1,R1,0x5E	1010 0101 0101 1110	A55E	将 0x35E 的低 8 位写入 R1
6	ADDI R2,R2,0	1010 1010 0000 0000	AA00	将 0x95 的高 8 位写入 R2
7	ADDI R3,R3,0x03	1010 1111 0000 0011	AF03	将 0x35E 的高 8 位写入 R3
8	ADD R0,R0,R1	0010 0001 0000 0000	2100	R0=R0+R1（低 8 位加）
9	ADDC R2,R2,R3	0110 1011 1000 0000	6B80	R2=R2+R3（高 8 位带进位加）
10	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
11	SW R0,R1,0x20	1100 0100 0010 0000	C420	将低八位加法运算结果写入 RAM 单元 0x20
12	SW R2,R1,0x21	1100 0110 0010 0001	C621	将高八位加法运算结果写入 RAM 单元 0x21
13	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
14	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零

15	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
16	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
17	ADDI R0,R0,0x95	1010 0000 1001 0101	A095	将 0x95 的低 8 位写入 R0
18	ADDI R1,R1,0x5E	1010 0101 0101 1110	A55E	将 0x35E 的低 8 位写入 R1
19	ADDI R2,R2,0	1010 1010 0000 0000	AA00	将 0x95 的高 8 位写入 R2
20	ADDI R3,R3,0x03	1010 1111 0000 0011	AF03	将 0x35E 的高 8 位写入 R3
21	SUB R0,R0,R1	0011 0001 0000 0000	3100	R0=R0-R1（低 8 位减）
22	SUBC R2,R2,R3	0101 1011 1000 0000	5B80	R2=R2-R3（高 8 位带借位减）
23	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
24	SW R0,R1,0x22	1100 0100 0010 0010	C422	将低八位减法运算结果写入 RAM 单元 0x22
25	SW R2,R1,0x23	1100 0110 0010 0011	C623	将低八位减法运算结果写入 RAM 单元 0x23
26	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
27	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
28	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
29	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
30	ADDI R0,R0,0x95	1010 0000 1001 0101	A095	将 0x95 的低 8 位写入 R0
31	ADDI R1,R1,0x5E	1010 0101 0101 1110	A55E	将 0x35E 的低 8 位写入 R1
32	ADDI R2,R2,0	1010 1010 0000 0000	AA00	将 0x95 的高 8 位写入 R2
33	ADDI R3,R3,0x03	1010 1111 0000 0011	AF03	将 0x35E 的高 8 位写入 R3
34	AND R0,R0,R1	0000 0001 0000 0000	0100	R0=R0&R1（低八位与）
35	AND R2,R2,R3	0000 1011 1000 0000	0B80	R2=R2&R3（高八位与）
36	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
37	SW R0,R1,0x24	1100 0100 0010 0100	C424	将低八位与运算结果写入 RAM 单元 0x24
38	SW R2,R1,0x25	1100 0110 0010 0101	C625	将高八位与运算结果写入 RAM 单元 0x25
39	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
40	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
41	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
42	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
43	ADDI R0,R0,0x95	1010 0000 1001 0101	A095	将 0x95 的低 8 位写入 R0

44	ADDI R1,R1,0x5E	1010 0101 0101 1110	A55E	将0x35E的低8位写入R1
45	ADDI R2,R2,0	1010 1010 0000 0000	AA00	将 0x95 的高 8 位写入R2
46	ADDI R3,R3,0x03	1010 1111 0000 0011	AF03	将0x35E的高8位写入R3
47	OR R0,R0,R1	0001 0001 0000 0000	1100	R0=R0|R1（低八位或）
48	OR R2,R2,R3	0001 1011 1000 0000	1B80	R2=R2|R3（高八位或）
49	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
50	SW R0,R1,0x26	1100 0100 0010 0110	C426	将低八位或运算结果写入 RAM 单元 0x26
51	SW R2,R1,0x27	1100 0110 0010 0111	C627	将高八位或运算结果写入 RAM 单元 0x27
52	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
53	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
54	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
55	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
56	ADDI R0,R0,0x95	1010 0000 1001 0101	A095	将 0x95 的低 8 位写入R0
57	ADDI R1,R1,0x5E	1010 0101 0101 1110	A55E	将0x35E的低8位写入R1
58	ADDI R2,R2,0	1010 1010 0000 0000	AA00	将 0x95 的高 8 位写入R2
59	ADDI R3,R3,0x03	1010 1111 0000 0011	AF03	将0x35E的高8位写入R3
60	SLT R2,R2,R3	0100 1011 1000 0000	4B80	进行高八位比较，结果写入 R2
61	SLT R0,R0,R1	0100 0001 0000 0000	4100	进行低八位比较，结果写入 R0
62	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
63	SW R0,R1,0x28	1100 0100 0010 1000	C428	将低八位比较结果写入 RAM 单元 0x28
64	SW R2,R1,0x29	1100 0110 0010 1001	C629	将高八位比较结果写入 RAM 单元 0x29
65	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
66	ADDI R3,R3,0x20	1010 1111 0010 0000	AF20	将 0x20 写入 R3
67	LW R0,R3,0	1011 1100 0000 0000	BC00	将 RAM 单元 0x20 的数据读出给 R0
68	LW R1,R3,1	1011 1101 0000 0001	BD01	将 RAM 单元 0x21 的数据读出给 R1
69	LW R0,R3,2	1011 1100 0000 0010	BC02	将 RAM 单元 0x22 的数据读出给 R0
70	LW R1,R3,3	1011 1101 0000 0011	BD03	将 RAM 单元 0x23 的数

				据读出给 R1
71	LW R0,R3,4	1011 1100 0000 0100	BC04	将 RAM 单元 0x24 的数据读出给 R0
72	LW R1,R3,5	1011 1101 0000 0101	BD05	将 RAM 单元 0x25 的数据读出给 R1
73	LW R0,R3,6	1011 1100 0000 0110	BC06	将 RAM 单元 0x26 的数据读出给 R0
74	LW R1,R3,7	1011 1101 0000 0111	BD07	将 RAM 单元 0x27 的数据读出给 R1
75	LW R0,R3,8	1011 1100 0000 1000	BC08	将 RAM 单元 0x28 的数据读出给 R0
76	LW R1,R3,9	1011 1101 0000 1001	BD09	将 RAM 单元 0x29 的数据读出给 R1
77	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行

程序 2：

完成将两个固定数据（如 0x95, 0x35E）进行加、减运算，将运算结果顺序显示在数码管上。两个结果显示之间需加一定延时（软件延时，可以通过给一个寄存器赋初值，对这个寄存器进行减运算，直至结果为 0）。

表 2 程序 2 指令编码

编号	汇编语言格式命令	二进制机器码指令	十六进制机器码指令	功能
0	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
1	ORI R0,R0,0x95	1001 0000 1001 0101	9095	将 0x95 的低 8 位写入 R0
2	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
3	ORI R1,R1,0x5E	1001 0101 0101 1110	955E	将 0x35E 的低 8 位写入 R1
4	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
5	ADD R2,R0,R1	0010 0001 1000 0000	2180	R2=R0+R1（低 8 位加）
6	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
7	SW R2,R3,0	1100 1110 0000 0000	CE00	将低 8 位运算结果写入 I/O0 端口
8	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
9	ORI R0,R0,0	1001 0000 0000 0000	9000	将 0x95 的高 8 位写入 R0
10	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
11	ORI R1,R1,0x03	1001 0101 0000 0011	9503	将 0x35E 的高 8 位写入 R1
12	ADDC R2,R0,R1	0110 0001 1000 0000	6180	R2=R0+R1（高 8 位带进位加）
13	SW R2,R3,1	1100 1110 0000 0001	CE01	将高 8 位运算结果写入

				I/O1 端口
14	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
15	ORI R3,R3,9	1001 1111 0000 1001	9F09	给 R3 赋初值 9
16	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
17	ORI R1,R1,1	1001 0101 0000 0001	9501	给 R1 赋值 1
18	SUB R2,R3,R1	0011 1101 1000 0000	3D80	R2=R3-1
19	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
20	BEQ R2,R0,3	1101 1000 0000 0011	D803	比较 R2 是否为 0,若为 0 跳转至下 4 条处开始执行
21	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
22	ORI R3,R2,R3	1001 1011 0000 0000	9B00	将 R2 的值赋给 R3
23	JMP 18	0111 0000 0001 0010	7012	跳至程序 18 处，继续对 R3 进行减法运算
24	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
25	ORI R0,R0,0x95	1001 0000 1001 0101	9095	将 0x95 的低 8 位写入 R0
26	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
27	ORI R1,R1,0x5E	1001 0101 0101 1110	955E	将 0x35E 的低 8 位写入 R1
28	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
29	SUB R2,R0,R1	0011 0001 1000 0000	3180	R2=R0-R1（低 8 位减）
30	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
31	SW R2,R3,0	1100 1110 0000 0000	CE00	将低 8 位运算结果写入 I/O0 端口
32	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
33	ORI R0,R0,0	1001 0000 0000 0000	9000	将 0x95 的高 8 位写入 R0
34	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
35	ORI R1,R1,0x03	1001 0101 0000 0011	9503	将 0x35E 的高 8 位写入 R1
36	SUBC R2,R0,R1	0101 0001 1000 0000	5180	R2=R0-R1(高 8 位带借位加)
37	SW R2,R3,1	1100 1110 0000 0001	CE01	将高 8 位运算结果写入 I/O1 端口
38	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行

程序 3：

从键盘输入一个表达式，如：149 - 862= ，通过执行 ROM 中的程序代码，将运算结果显示在数码管上。

表 3 程序 3 指令编码

编号	汇编语言格式命令	二进制机器码指令	十六进制机器码指令	功能
0	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
1	LW R0,R0,7	1011 0000 0000 0111	B007	由 I/O7 读取 finish 信号
2	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
3	ORI R1,R1,1	1001 0101 0000 0001	9501	将 R1 赋值为 1
4	BEQ R0,R1,1	1101 0001 0000 0001	D101	判断 finish 是否为 1，相等后跳转到下 2 条处开始执行
5	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行
6	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
7	LW R0,R0,2	1011 0000 0000 0010	B002	将 srcL 由 I/O2 读入 R0
8	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
9	SW R0,R3,8	1100 1100 0000 1000	CC08	将 srcL 写入 RAM 单元 8
10	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
11	LW R1,R1,3	1011 0101 0000 0011	B503	将 srcH 由 I/O3 读入 R1
12	SW R1,R3,9	1100 1101 0000 1001	CD09	将 srcH 写入 RAM 单元 9
13	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
14	LW R0,R0,4	1011 0000 0000 0100	B004	将 dstL 由 I/O4 读入 R0
15	SW R0,R3,10	1100 1100 0000 1010	CC0A	将 dstL 写入 RAM 单元 10
16	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
17	LW R1,R1,5	1011 0101 0000 0101	B505	将 dstH 由 I/O5 读入 R1
18	SW R1,R3,11	1100 1101 0000 1011	CD0B	将 dstH 写入 RAM 单元 11
19	ANDI R2,R2,0	1000 1010 0000 0000	8A00	将 R2 清零
20	LW R2,R2,6	1011 1010 0000 0110	BA06	将 aluop 由 I/O6 读入 R2
21	SW R2,R3,12	1100 1110 0000 1100	CE0C	将 aluop 写入 RAM 单元 12
22	ORI R3,R3,1	1001 1111 0000 0001	9F01	将 R3 赋值为 1
23	BNE R2,R3,**10**	1110 1011 0000 1010	EB0A	判断是否进行加法运算，如不是跳转到下 **11** 条开始执行
24	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
25	LW R0,R3,8	1011 1100 0000 1000	BC08	将 srcL 读入 R0
26	LW R1,R3,10	1011 1101 0000 1010	BD0A	将 dstL 读入 R1
27	ADD R2,R0,R1	0010 0001 1000 0000	2180	R2=R0+R1（低 8 位加）
28	SW R2,R3,0	1100 1110 0000 0000	CE00	将低 8 位加法运算结果

				送到 I/O0 端口
29	LW R0,R3,9	1011 1100 0000 1001	BC09	将 srcH 读入 R0
30	LW R1,R3,11	1011 1101 0000 1011	BD0B	将 dstH 读入 R1
31	ADDC R2,R0,R1	0110 0001 1000 0000	6180	R2=R0+R1（高 8 位带进位加）
32	SW R2,R3,1	1100 1110 0000 0001	CE01	将高 8 位加法运算结果送到 I/O1 端口
33	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行
34	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
35	LW R2,R3,12	1011 1110 0000 1100	BE0C	将运算符读入 R2
36	ORI R3,R3,2	1001 1111 0000 0010	9F02	将 R3 赋值为 2
37	BNE R2,R3,**10**	1110 1011 0000 1010	EB0A	判断是否进行减法运算，如不是跳转到下 **11** 条开始执行
38	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
39	LW R0,R3,8	1011 1100 0000 1000	BC08	将 srcL 读入 R0
40	LW R1,R3,10	1011 1101 0000 1010	BD0A	将 dstL 读入 R1
41	SUB R2,R0,R1	0011 0001 1000 0000	3180	R2=R0-R1（低 8 位减）
42	SW R2,R3,0	1100 1110 0000 0000	CE00	将低 8 位减法运算结果送到 I/O0 端口
43	LW R0,R3,9	1011 1100 0000 1001	BC09	将 srcH 读入 R0
44	LW R1,R3,11	1011 1101 0000 1011	BD0B	将 dstH 读入 R1
45	SUBC R2,R0,R1	0101 0001 1000 0000	5180	R2=R0+R1（高 8 位带进位减）
46	SW R2,R3,1	1100 1110 0000 0001	CE01	将高 8 位减法运算结果送到 I/O1 端口
47	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行
48	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
49	LW R2,R3,12	1011 1110 0000 1100	BE0C	将运算符读入 R2
50	ORI R3,R3,4	1001 1111 0000 0100	9F04	将 R3 赋值为 4
51	BNE R2,R3,**10**	1110 1011 0000 1010	EB0A	判断是否进行与运算，如不是跳转到下 **11** 条开始执行
52	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
53	LW R0,R3,8	1011 1100 0000 1000	BC08	将 srcL 读入 R0
54	LW R1,R3,10	1011 1101 0000 1010	BD0A	将 dstL 读入 R1
55	AND R2,R0,R1	0000 0001 1000 0000	0180	R2=R0&R1（低 8 位与）
56	SW R2,R3,0	1100 1110 0000 0000	CE00	将低 8 位与运算结果送到 I/O0 端口
57	LW R0,R3,9	1011 1100 0000 1001	BC09	将 srcH 读入 R0
58	LW R1,R3,11	1011 1101 0000 1011	BD0B	将 dstH 读入 R1

59	AND R2,R0,R1	0000 0001 1000 0000	0180	R2=R0&R1（高 8 位与）
60	SW R2,R3,1	1100 1110 0000 0001	CE01	将高 8 位与运算结果送到 I/O1 端口
61	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行
62	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
63	LW R2,R3,12	1011 1110 0000 1100	BE0C	将运算符读入 R2
64	ORI R3,R3,8	1001 1111 0000 1000	9F08	将 R3 赋值为 8
65	BNE R2,R3,17	1110 1011 0001 0001	EB11	判断是否进行比较运算，如不是跳转到下 18 条开始执行
66	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
67	LW R0,R3,9	1011 1100 0000 1001	BC09	将 srcH 读入 R0
68	LW R1,R3,11	1011 1101 0000 1011	BD0B	将 dstH 读入 R1
69	BEQ R0,R1,5	1101 0001 0000 0101	D105	比较高 8 位是否相等，若相等跳转至下 6 条处开始执行
70	SLT R2,R0,R1	0100 0001 1000 0000	4180	如果不等，得到比较结果 R2
71	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
72	SW R3,R3,0	1100 1111 0000 0000	CF00	将 0 送到 I/O0 端口
73	SW R2,R3,1	1100 1110 0000 0001	CE01	将 R2 的比较结果送到 I/O1 端口
74	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行
75	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
76	LW R0,R3,8	1011 1100 0000 1000	BC08	将 srcL 读入 R0
77	LW R1,R3,10	1011 1101 0000 1010	BD0A	将 dstL 读入 R1
78	SLT R2,R0,R1	0100 0001 1000 0000	4180	比较低 8 位，结果写入 R2
79	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
80	SW R3,R3,0	1100 1111 0000 0000	CF00	将 0 送到 I/O0 端口
81	SW R2,R3,1	1100 1110 0000 0001	CE01	将 R2 的比较结果送到 I/O1 端口
82	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行
83	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
84	LW R2,R3,12	1011 1110 0000 1100	BE0C	将运算符读入 R2
85	ORI R3,R3,16	1001 1111 0001 0000	9F10	将 R3 赋值为 16
86	BNE R2,R3,9	1110 1011 0000 1001	EB09	判断是否进行或运算，如不是跳转到下 10 条开始执行
87	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零

88	LW R0,R3,8	1011 1100 0000 1000	BC08	将 srcL 读入 R0
89	LW R1,R3,10	1011 1101 0000 1010	BD0A	将 dstL 读入 R1
90	OR R2,R0,R1	0001 0001 1000 0000	1180	R2=R0|R1（低 8 位或）
91	SW R2,R3,0	1100 1110 0000 0000	CE00	将低 8 位或运算结果送到 I/O0 端口
92	LW R0,R3,9	1011 1100 0000 1001	BC09	将 srcH 读入 R0
93	LW R1,R3,11	1011 1101 0000 1011	BD0B	将 dstH 读入 R1
94	OR R2,R0,R1	0001 0001 1000 0000	1180	R2=R0|R1（高 8 位或）
95	SW R2,R3,1	1100 1110 0000 0001	CE01	将高 8 位或运算结果送到 I/O1 端口
96	JMP 0	0111 0000 0000 0000	7000	跳到程序开始处 0，重新执行

2. 简单计算机系统设计 A

利用设计的 ROM、RAM、ALU、控制器、PC 程序指针计数器模块，构成简单计算机系统 A，在 ROM 中存放编程练习中的程序 1，并进行系统的仿真和调试，下载到实验板上进行测试、运行。

注意在各模块的连接中，需根据各指令数据通路的要求增加多路选择器等部件，避免信号相连时的冲突。

- **顶层原理图**

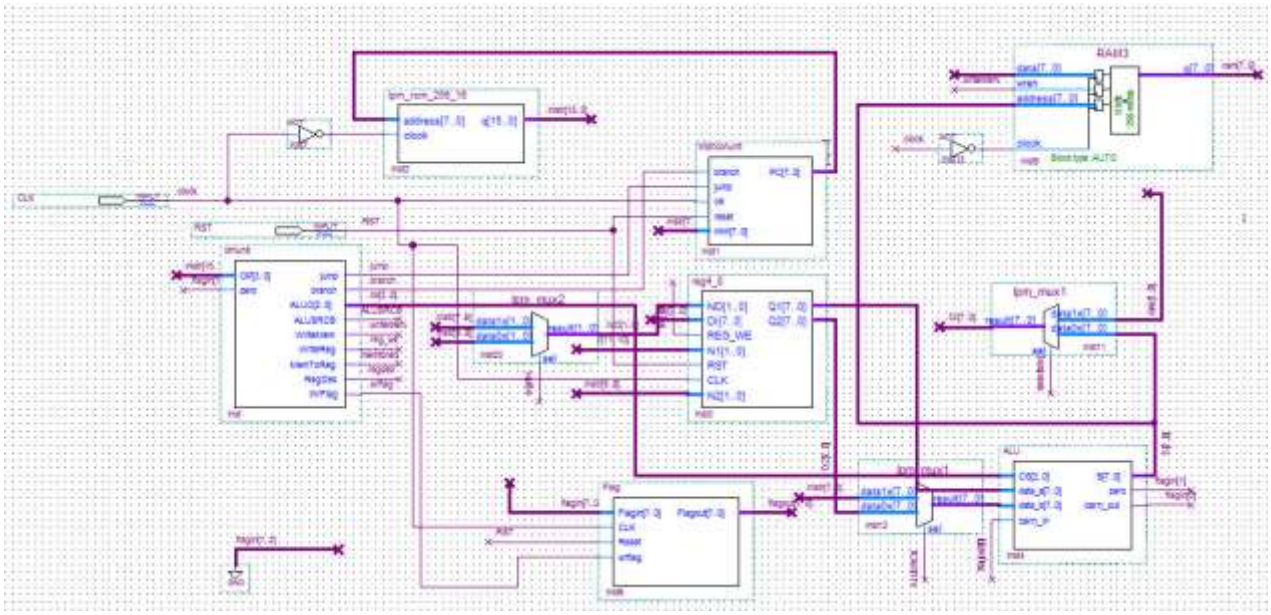

图1 系统A顶层原理图

➢ **系统设计思路**

对于系统 A 的要求主要是将控制器单元、寄存器组、标志寄存器、ALU、ROM 和 RAM 几个核心模块组装在一起，尚未与输入输出接口相连，因此应该重点关注各模块之间信号的连接和配合。

时钟信号的配合：

由于各模块存在不同程度的延迟时间，因此，应该合理设计各模块的时钟工作边沿。在本设计中，为了使各模块之间的配合较为稳妥，分别对程序指针计数器、ROM、寄存器组、RAM依次进行时钟边沿触发，其中，程序指针计数器和寄存器组由上升沿触发，ROM和RAM由下降沿触发。这样的时钟沿错开处理，有效地避免了各信号之间的冲突。但美中不足的是，需要两个完整的时钟周期才能处理完一条指令。

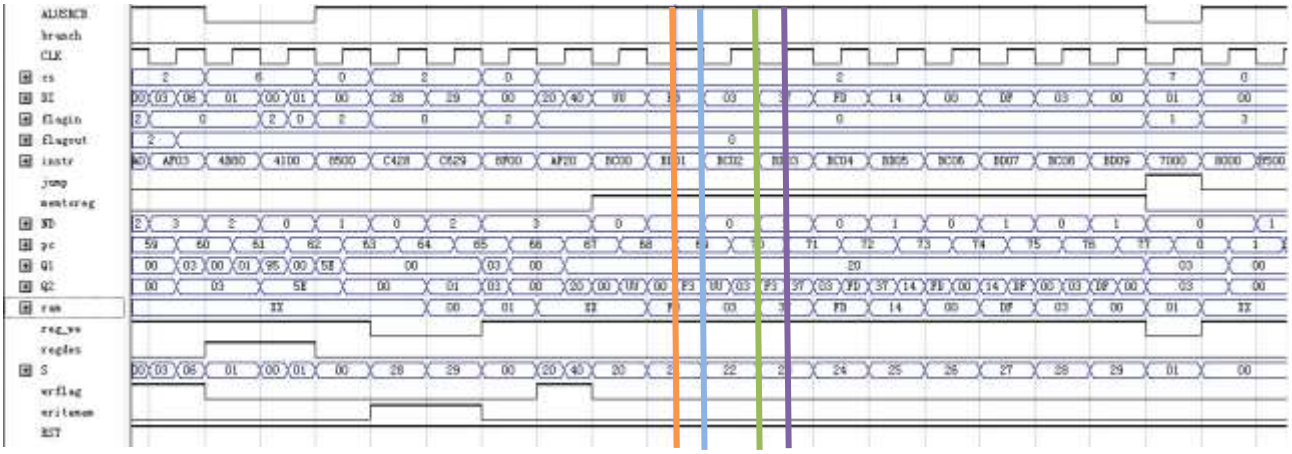

图2 系统A波形仿真图

下面分析系统A的波形仿真图，以LW指令为例。当时钟上升沿到来时（如橙色线所示），PC完成+1的操作，变为69；下降沿到来时（如蓝色线所示），69号指令BC02被读出。由于寄存器R1始终为20，此时，控制器给出ALUSRCB的高电平信号将需要运算的立即数2送到ALU端，ALU立即计算出需要读取的地址0x22，并在RAM端等待。在下一个时钟下降沿到来时（如绿色线所示），RAM单元0x22存储的数据37被读出（可以从ram端口的数据看出）。然后再下一个上升沿到来时（如紫色线所示），这个数据被写入寄存器组中（可以从Q2的数据看出）。由以上波形仿真分析可以看出，一个RAM读取指令需要两个时钟周期才能完成。

控制信号及数据的匹配：

在将各模块组合在一起时，有如下模块和信号需要添加数据选择器进行选通：

(1) 目的寄存器的编号ND

由于R型指令和I型指令的目的寄存器不同，因此应根据控制器单元输出的目的寄存器信号regdes选择目的寄存器编号在16位指令中的位置：当regdes=1时，由7,6两位读出目的寄存器编号；当regdes=0时，由9,8两位读出目的寄存器编号。

(2) ALU的运算数data_b

R型指令是对两个寄存器里的数据进行运算，而I型指令是对寄存器中的数据和立即数进行运算。因此应根据控制器单元输出的立即数控制信号ALUSRCB决定data_b的数据来源：当ALUSRCB=1时，data_b为指令后8位的立即数imm；当ALUSRCB=0时，data_b为由寄存器组

读出的第二个数据Q2。

(3) 写入寄存器的数据DI

写入寄存器的数据可能是RAM中读取的数据q，也可能是ALU的运算结果S。因此应根据控制器单元输出的memtoreg信号来控制写入寄存器数据的来源：当memtoreg=1时，DI由RAM中读取的数据决定；当memtoreg=0时，DI由ALU的运算结果S决定。

- **遇到的问题及解决方案**

减法借位信号的处理：

在对系统进行仿真后，发现高位减法运算结果是FC，而不是FD，而其他所有运算结果正确。因此判断问题应该出在标志寄存器Flag上。

查找仿真波形，我发现这与ROM程序的编写思路有关。一开始，我的想法是先存低位，然后进行低位运算，写入RAM，然后再存高位。由于赋值运算ADDI进行运算时，wr_flag信号也是高电平，所以之前低位减法的借位信号就被覆盖了。解决这个问题，可以有两种思路：一个是将赋值运算改为ORI，另一种是低位和高位运算相邻进行。我采取的是第二种方法，因此汇编语言程序的流程图大致如下：

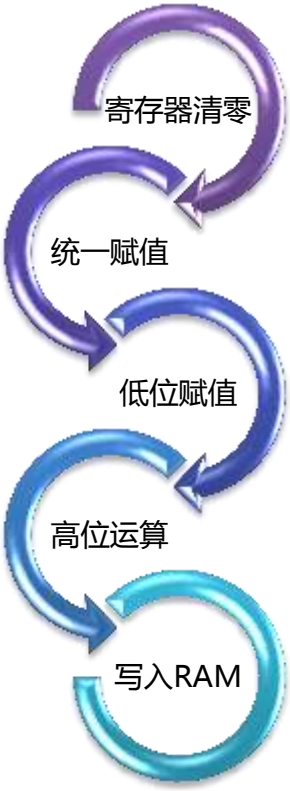

图3 系统A汇编语言流程图

RAM读取时钟错开的问题：

将系统的时钟配合按照上述的进行组装好以后，发现仿真波形与预想的时钟周期不一样，ram端的读取比预计的延迟了1个周期，造成整体数据向后错位3-4个周期。后来与同学们交流，发现RAM端后面q加入了一个寄存器，使数据延迟了一个周期。取消了q端寄存器后，

仿真波形就像上面分析的一样了。

3. 简单计算机系统设计 B

在简单计算机系统 A 的基础上，增加 I/O 端口及其映射模块、数码管输出接口，将 I/O 端口及其映射模块中的 IO0[7..0]、IO1[7..0]与数码管输出接口的 datainL[7..0]、datainH[7..0]相连，构成简单计算机系统 B。在 ROM 中存放编程练习中的程序 2，并进行系统的仿真和调试，下载到实验板上进行测试、运行。

- **顶层原理图**

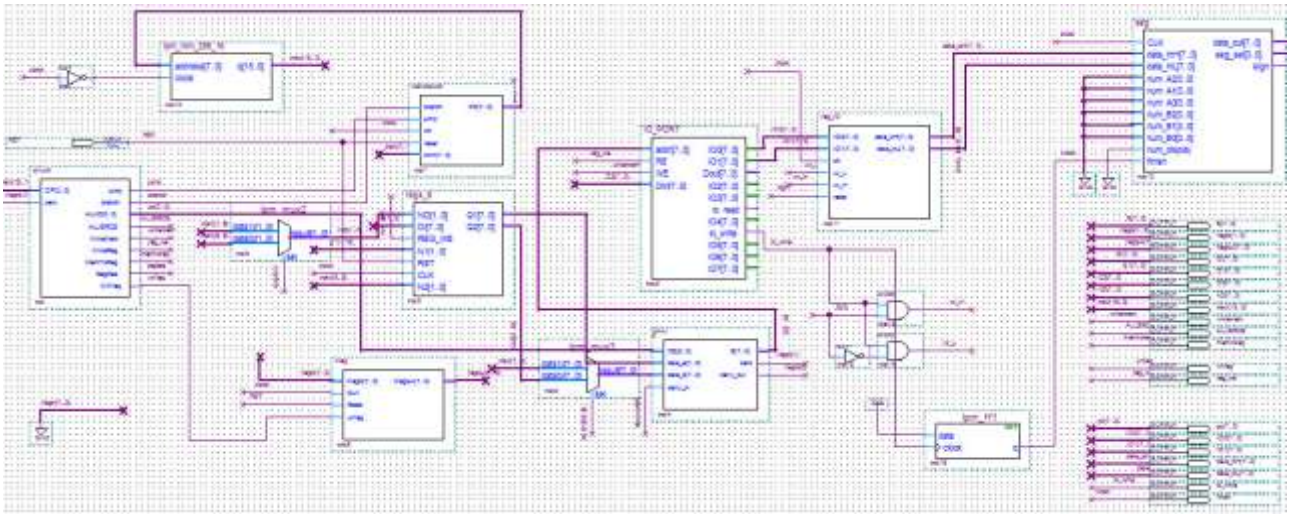

图4 系统B顶层原理图

- **系统设计思路**

设计系统B时，由于运算数是在ROM程序中预存好的，而输出是直接送到IO端口，因此没有使用RAM模块。在时钟配合方面，几乎是完全沿用了系统A的时钟体系。而在系统B中主要添加的是IO端口及显示模块。

IO端口模块的连接：

取消了RAM模块后，IO端口模块几乎是完全替代了RAM的位置：地址信号由ALU运算结果端给出，读取信号RE为写寄存器reg_we信号，写入信号WE为写入mem信号writemem，写入的数据由寄存器组的Q2给出。输出IO0作为最终数据的低8位data_inL，输出IO1作为最终数据的高8位data_inH。

负数运算结果的处理：

根据目前的运算，当ALU对高位数据进行减法运算后，若第一个运算数小于第二个运算数，那么高位的运算结果就是补码输出。为了将补码转换成最终的数据，在小学期seg模块的基础上，我在解码器Decoder中加入了判断结果是否负数，并转化为绝对值原码输出的语句。具体的算法如下所示：

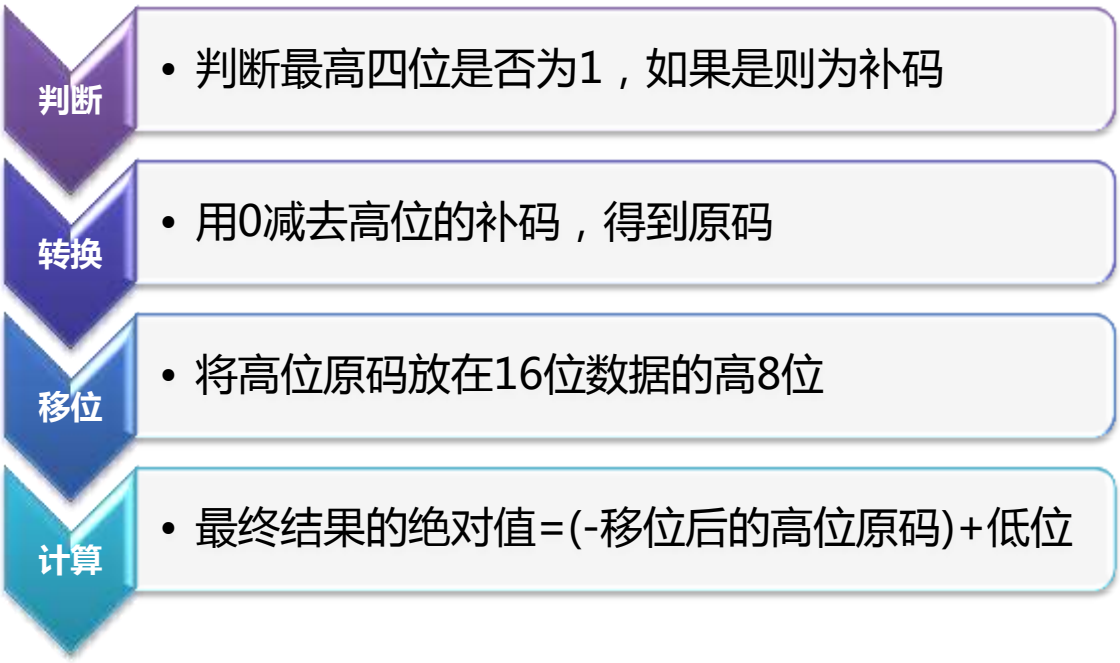

图5 减法运算补码转换算法图

减法运算后的仿真波形图如下：

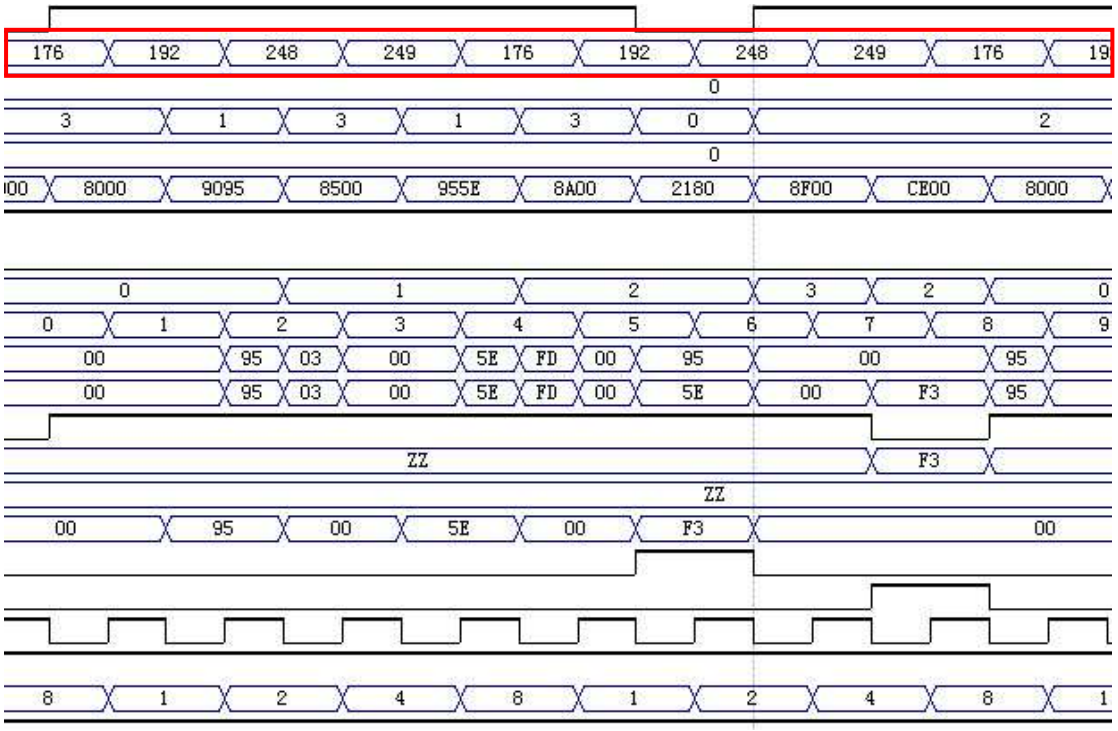

图 6 系统 B 仿真波形图（减法结果）

观察仿真波形图，其中第二行为数码管的 8 段显示数据：192 代表千位的 0，248 代表百位的 7，249 代表十位的 1，176 代表各位的 3。运算结果恰好是-713。

延时的处理：

根据题目要求，在加法结果和减法结果显示之间要有一定的延时，使其通过数码管充分

得显示出来。为了实现这一延时，在ROM程序中加入了延时环节：

14	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
15	ORI R3,R3,9	1001 1111 0000 1001	9F09	给 R3 赋初值 9
16	ANDI R1,R1,0	1000 0101 0000 0000	8500	将 R1 清零
17	ORI R1,R1,1	1001 0101 0000 0001	9501	给 R1 赋值 1
18	SUB R2,R3,R1	0011 1101 1000 0000	3D80	R2=R3-1
19	ANDI R0,R0,0	1000 0000 0000 0000	8000	将 R0 清零
20	BEQ R2,R0,3	1101 1000 0000 0011	D803	比较 R2 是否为 0,若为 0 跳转至下 4 条处开始执行
21	ANDI R3,R3,0	1000 1111 0000 0000	8F00	将 R3 清零
22	ORI R3,R2,R3	1001 1011 0000 0000	9B00	将 R2 的值赋给 R3
23	JMP 18	0111 0000 0001 0010	7012	跳至程序 18 处，继续对 R3 进行减法运算

如此一来，在系统分频后，再加上考虑到各模块的延时可能对减法运算结果的延迟，可以很好地实现加法运算后的延迟环节。仿真波形如下：

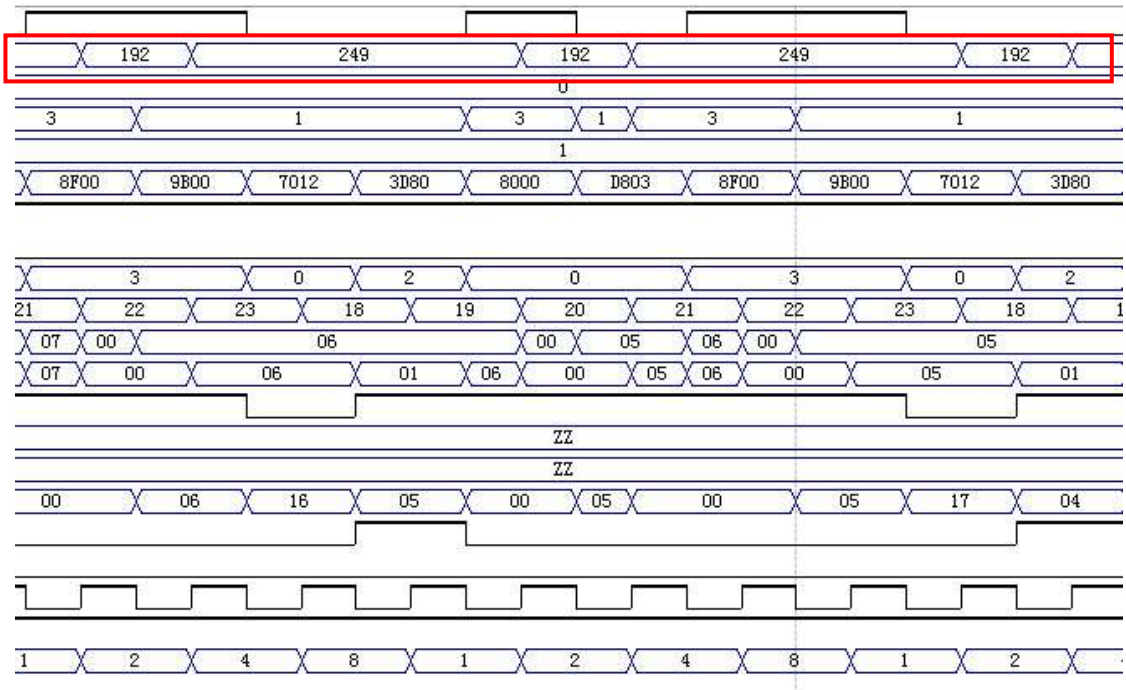

图7 系统B仿真波形图（延迟时）

其中波形的第二行为数码管的输出（折算成十进制），249 代表 1，192 代表 0，因此延时期间显示的数据应该是加法运算的结果 1011，在最终数码管上的现象就是 1011 和 713 交替显示，且频率取决于 ROM 中延时环节初值的设定。

- **遇到的问题及解决方案**

IO 端口模块数据的保持：

按上述方法连接好电路后，通过仿真波形发现 IO 端口的输出数据不能保持，因此应该

加入寄存器环节。我利用 Verilog 编写了 IO 寄存器模块 reg_IO：

```
module reg_IO(IO0,IO1,clk,wr_L,wr_H,reset,data_inH,data_inL);
input [7:0]IO0;
input [7:0]IO1;
input clk;
input wr_L;
input wr_H;
input reset;
output reg [7:0]data_inH;
output reg [7:0]data_inL;

always @(posedge clk) begin
    if (reset) begin
        if (wr_L==1) begin
            data_inL = IO0;
        end
        if (wr_H==1) begin
            data_inH = IO1;
        end
        else begin
            data_inH = data_inH;
            data_inL = data_inL;
        end
    end
    else begin
        data_inH = 0;
        data_inL = 0;
    end
end
endmodule
```

图 8 reg_IO 模块

其中，wr_L 是低位写入信号，wr_H 是高位写入信号，二者是由 io_write 和 S 的最低位或最低位取反相与得到的，也就是说，当写入 IO 并且地址为 0 时读入低位数据 data_inL，当写入 IO 并且地址为 1 时读入高位数据 data_inH。这样就可以有效地将计算结果经过 IO 模块送到数码管显示模块。

4. 简单计算机系统设计 C

在简单计算机系统 B 的基础上，增加 4x4 键盘输入接口模块，将 I/O 端口及其映射模块中 IO3[7..0]~IO6[7..0]分别与键盘输入接口模块的 srcL[7..0]、srcH[7..0]、dstL[7..0]、dstH[7..0]、, aluop[7..0]相连，构成简单计算机系统 C，在 ROM 中存放编程练习中的程序 3，并进行系统的仿真和调试，下载到实验板上进行测试、运行。

➢ **顶层原理图**

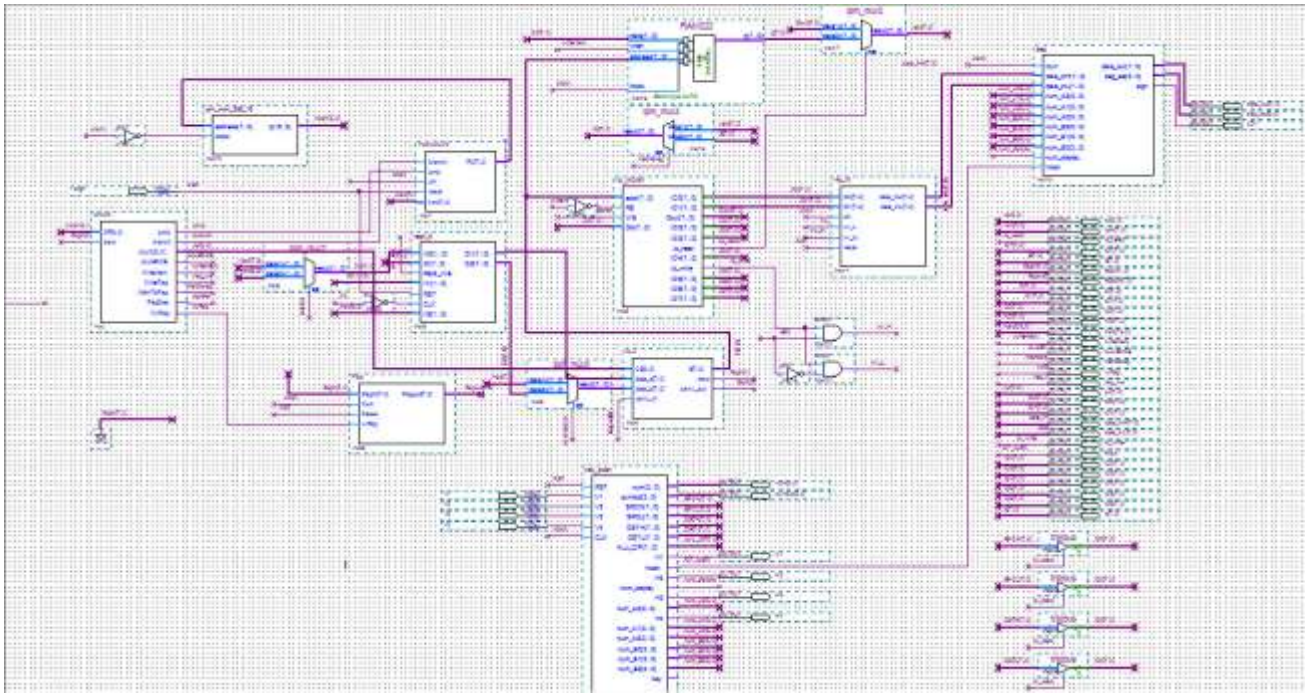

图9 系统C顶层原理图

➢ **系统设计思路**

系统 C 是在之前系统的基础上加入了键盘模块，从而实现了由键盘输入数字进行运算，然后由数码管输出的效果。

ROM 程序设计思路：

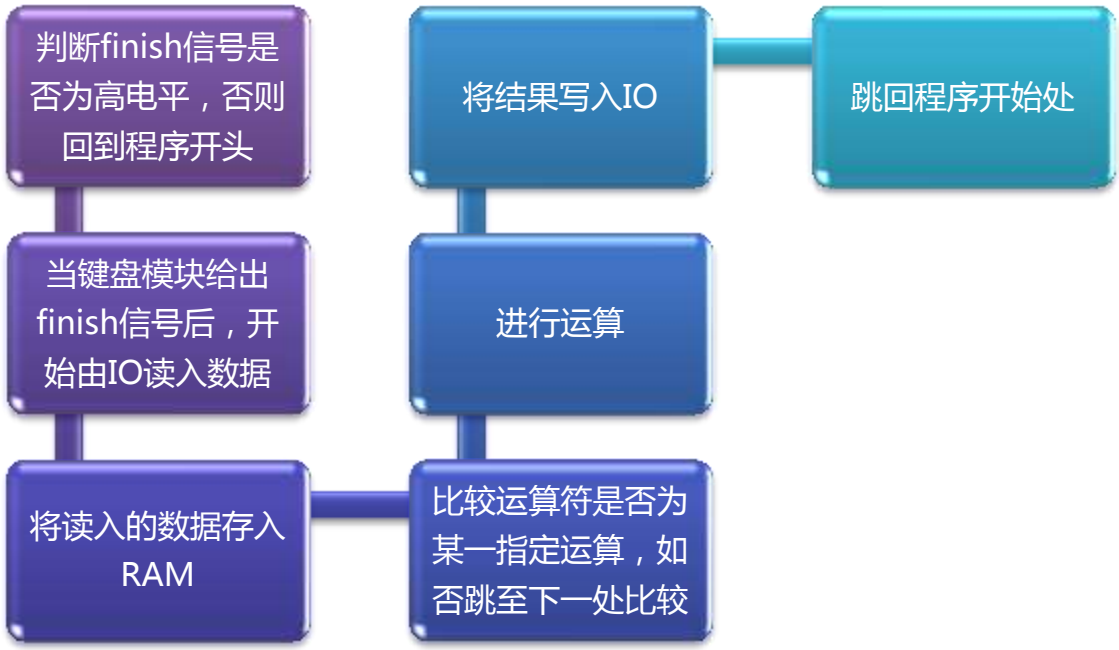

图 10 系统 C 汇编语言程序流程图

RAM 和 IO 地址的对应关系：

要特别注意的是 RAM 地址和 IO 地址的对应关系。根据原理图，RAM 和 IO 的地址端连的是同样的地址。为了避免混淆，规定当 addr=0~7 时，对 IO 端口进行操作，而 RAM 使用的存储单元为 8~255。

- **遇到的问题及解决方案**

时钟信号配合问题：

在沿用了之前的系统后，在系统 C 仿真时一直出现不定态，然后在某一 PC 指令后就会停止读取（主要是 PC 跳转出现了问题）。因此猜想可能是之前的时钟配合方案每条指令的执行周期太长，导致在特定情况下模块不知道该如何跳转。

为了解决这个问题，应该将指令的周期缩短，于是将寄存器组改成下降沿触发，RAM 利用上升沿触发，有效地缩短了周期，最终得到了正确的结果。

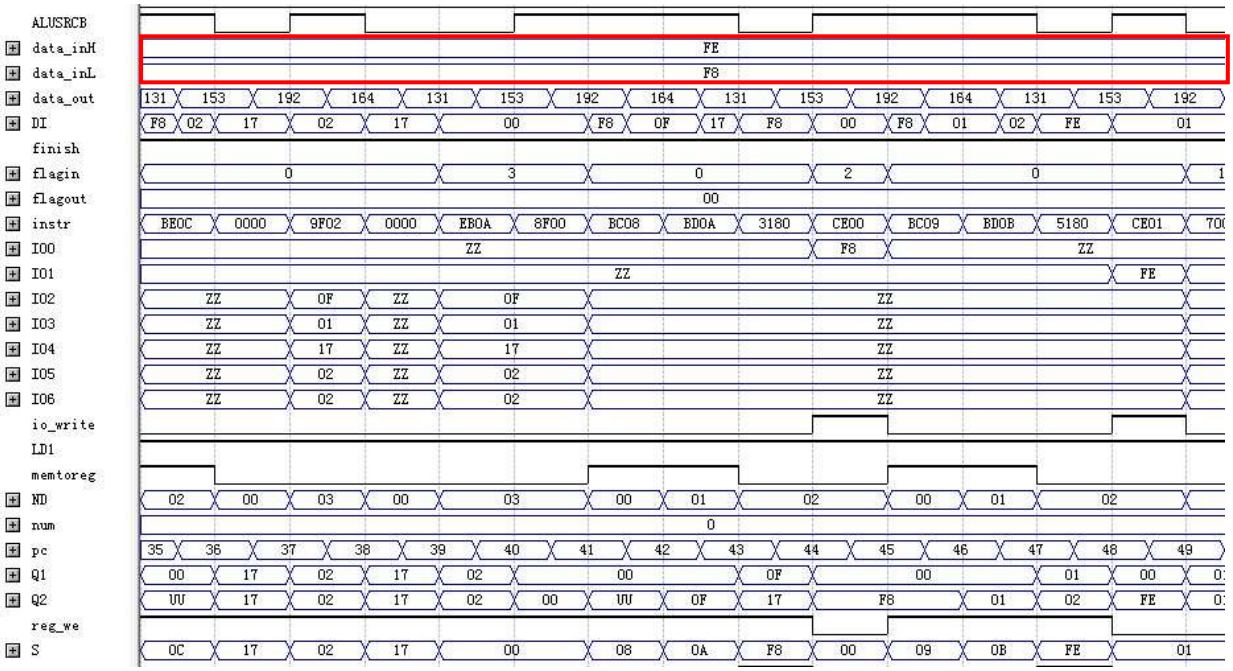

图 11 系统 C 仿真波形图

IO 端口输入问题：

将键盘模块的输出 SRCL、SRCH、DSTH、DSTL 等接至 IO 时，如果直接相接在编译时会报错。因此必须先通过一个三态门然后再送到 IO 端口。

接线问题：

后来遇到的问题一直是仿真没有问题，但是下载后没有运算结果。仔细检查后也没有发现电路的原理性问题。后来检查下载的工程中的原理图发现，在分频后寄存器组的时钟信号接成原始的了。改接线后就解决问题了。类似的问题还出现在之前的输入数字显示的模块上。可见，在面对复杂系统时，还需要培养细致认真的精神。

实验收获与体会

在两周的实验中，我充分体会到了程序调试的困难和最终成功后的喜悦。在完成实验任务的过程中，我从一开始的不熟练到后来上手后的娴熟，中间也历经了很多波折。仅仅是一个简单的 ROM 程序，我就改之又改。我深深地体会到，软件和硬件程序设计单独看来都不难，最痛苦的便是软硬件结合，一边看着仿真波形发现电路图的漏洞，一边改写着 ROM 的程序，有时还要绞尽脑汁去想有没有其他的实现思路。

在时间安排方面，这次也吃了不少亏。之前一直不是很抓紧，直到最后一周才拼命调试，最终是很紧张地恰好在课上完成了，不然很有可能完不成。所以，吃一堑，长一智，以后的实验一定要尽早完成，然后利用开放时间进行调试，避免上课的窘态。

另外，在调试程序的时候一定要有耐心，仔细地梳理写程序及构建电路的思路，及时地发现问题，然后集中精力解决问题。如果一出现问题就烦躁不堪，是很难高效率地解决问题的。因此，在今后的调试程序中，还应该时刻调整自己的心态。

总体来说，完成这次实验对我是个不小的挑战。希望自己能够在后续的单片机实验中更给力些。谢谢老师和助教们的帮助和指导！