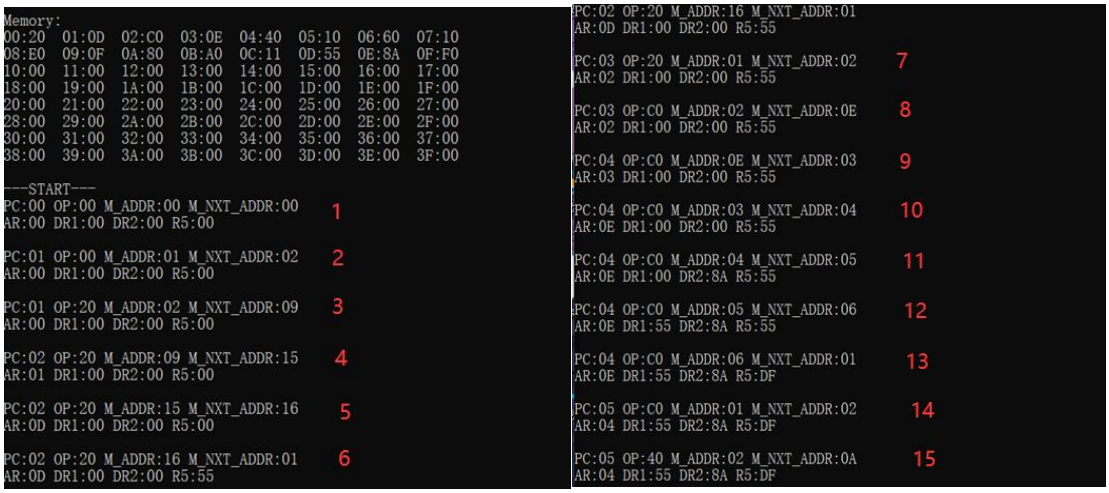
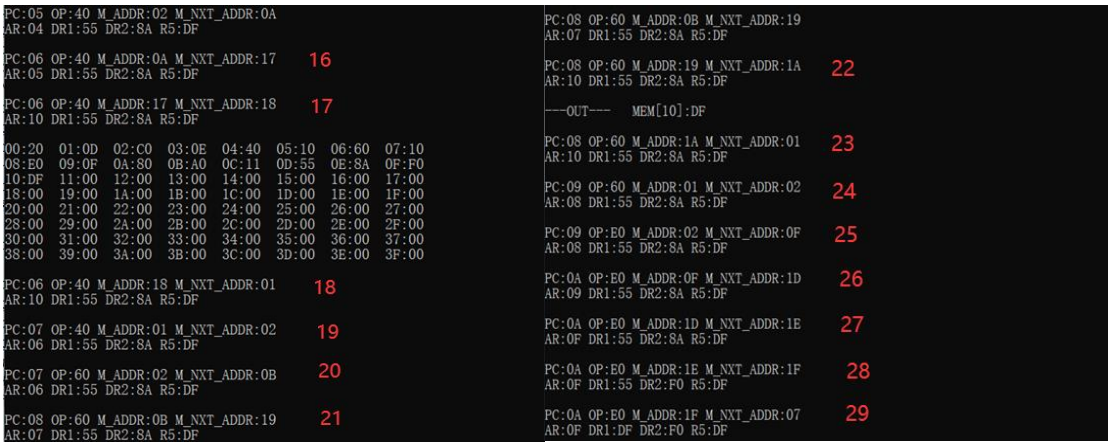
微控 1

- Memory 为用户程序，其内每一个内存单元字长为 8 位，单元内数据用 16 进制表示,用来存放指令、数据、地址。所有数据都用 16 进制表示 M_ADDR——当前地址、NEXT_ADDRT——下一地址、OP——操作码，对应 IR，其余为寄存器和暂存器(16 位，高位 0 不显示)。当前地址与下一地址的有效位为低 5 位，以下说明均用低 5 位。

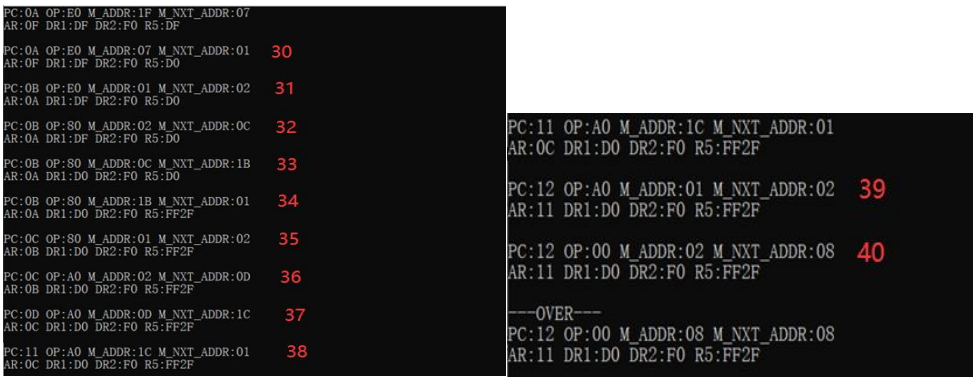


1. 初始状态，当前微地址为 00000，操作码、PC、寄存器、暂存器初始化为 00，执行操作 SW->PC
2. 当前微地址为 00001，SW->PC 后，PC=00，执行操作 PC->AR，PC+1，这里先将 PC 打入 AR，所以 AR 为 00H，在执行 PC+1，所以 PC=01H，下一微地址由 ROM 读出为 02H 即 00010
3. 当前微地址为 00010，执行操作 RAM->IR 译码器，00H 地址的 RAM 单元根据用户程序看到是 20H（LDA 指令），所以 OP(IR)=IR=20H=00100000，P=1，将 IR 的高三位 001 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01001
4. 当前微地址为 01001，此时 PC=01H，执行 PC->AR，PC+1，先执行 PC->AR，所以 AR=01H，随后 PC+1=02H，下一微地址由 ROM 读出为 10101
5. 当前微地址为 10101，AR=01H，执行操作 RAM->AR，将 01H 地址 RAM 内存单元即用户程序 01H 地址单元内容打入 AR，AR 变为 0DH，下一微地址由 ROM 读出为 10110
6. 当前微地址为 10110，执行操作 RAM->R5，此时 AR=0DH，将 0DH 地址对应 RAM 单元内容传给 R5，根据用户程序，0DH 对应单元内容为 55H，R5=55H，下一微地址由 ROM 读出为 00001
7. 当前微地址为 00001，此时 PC=02H，执行操作 PC->AR，AR=02H，PC+1，PC=03H，下一微地址由 ROM 读出为 00010
8. 当前微地址为 00010，执行操作为 RAM->IR 译码器，02H 地址的 RAM 单元根据用户程序看到是 C0H（ADD 指令），所以 OP(IR)=IR=C0H=11000000，P=1，将 IR 的高三位 110 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01110
9. 当前微地址为 01110，此时 PC=03H，执行操作 PC->AR，AR=03H，PC+1，PC=04H，下一微地址由 ROM 读出为 00011
10. 当前微地址为 00011，AR=03H，执行操作 RAM->AR，将 AR 对应 RAM 单元的内容传给 AR，03H 对应的用户程序为 0EH，AR=OEH，下一地址由 ROM 读出为 00100
11. 当前微地址为 00100，AR=0EH，执行操作 RAM->DR2，将 AR=0EH 对应的 RAM 内存单元内容传给 DR2，0EH 单元内容为 8A，所以 DR2=8AH，下一地址由 ROM 读出为 00101
12. 当前微地址为 00101，执行 R5->DR1，故 DR1=R5=55H，下一微地址由 ROM 读出为 00110
13. 当前微地址为 00110，执行 DR1+DR2->R5，DR1=55H,DR2=8AH，故 R5=55H+8AH=DFH，下一微地址由 ROM 读出为 00001
14. 当前微地址为 00001，此时 PC=04H，执行操作 PC->AR，AR=04H，PC+1，PC=05H，下一微地址由 ROM 读出为 00010
15. 当前地址为 00010，执行操作为 RAM->IR 译码器，04H 地址的 RAM 单元根据用户程序看到是 40H（STA 指令），所以 OP(IR)=IR=40H=01000000，P=1，将 IR 的高三位 010 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01010



16. 当前地址为 01010，此时 PC=05H，执行操作 PC->AR，AR=05H，PC+1，PC=06H，下一微地址由 ROM 读出为 10111
17. 当前微地址为 10111，此时 AR=05H，执行 RAM->AR 将 05H 对应 RAM 内存单元内容打入 AR，根据用户程序可以看到 05H 单元为 10H，所以 AR=10H，下一微地址由 ROM 读出为 11000
18. 当前微地址为 11000，此时 AR=10H，执行 R5->RAM，将 R5 的值打入 AR 对应的 RAM 内存单元，根据内存单元情况可以看到 10H 单元内容已经变为 R5=DFH，下一微地址由 ROM 读出为 00001
19. 当前地址为 00001，此时 PC=06H，执行操作 PC->AR，AR=06H，PC+1，PC=07H，下一微地址由 ROM 读出为 00010
20. 当前微地址为 00010，执行操作为 RAM->IR 译码器，06H 地址的 RAM 单元根据用户程序看到是 60H（OUT 指令），所以 OP(IR)=IR=C0H=01100000，P=1，将 IR 的高三位 011 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01011
21. 当前微地址为 01011，此时 PC=07H，执行操作 PC->AR，AR=07H，PC+1，PC=08H，下一微地址由 ROM 读出为 11001

22. 当前微地址为 11001，此时 AR=07H，执行操作 RAM->AR，将 07H 对应 RAM 内存单元内容 10H 打入 AR，AR=10H，下一微地址由 ROM 读出为 11010
23. 当前微地址为 11010，此时 AR=10H，执行操作 RAM->BUS，将 10H 对应 RAM 内存单元 DFH 输出到总线 BUS，可以看到输出了 DF，下一微地址由 ROM 读出为 00001
24. 当前地址为 00001，此时 PC=08H，执行操作 PC->AR，AR=08H，PC+1，PC=09H，下一微地址由 ROM 读出为 00010
25. 当前微地址为 00010，执行操作为 RAM->IR 译码器，08H 地址的 RAM 单元根据用户程序看到是 E0H（AND 指令），所以 OP(IR)=IR=E0H=11100000，P=1，将 IR 的高三位 111 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01111
26. 当前微地址为 01111，此时 PC=09H，执行操作 PC->AR，AR=09H，PC+1，PC=0AH，下一微地址由 ROM 读出为 11101
27. 当前微地址为 11101，此时 AR=09H，执行操作 RAM->AR，将 09H 对应 RAM 内存单元 0FH 打入 AR，AR=0FH，下一微地址由 ROM 读出为 11110
28. 当前微地址为 11110，此时 AR=0FH，执行操作 RAM->DR2，将 0FH 对应 RAM 内存单元内容 F0H 打入 DR2，DR2=0FH，下一微地址由 ROM 读出为 11111
29. 当前微地址为 11111，执行操作 R5->DR1，所以 DR1=R5=DFH，下一微地址为 00111

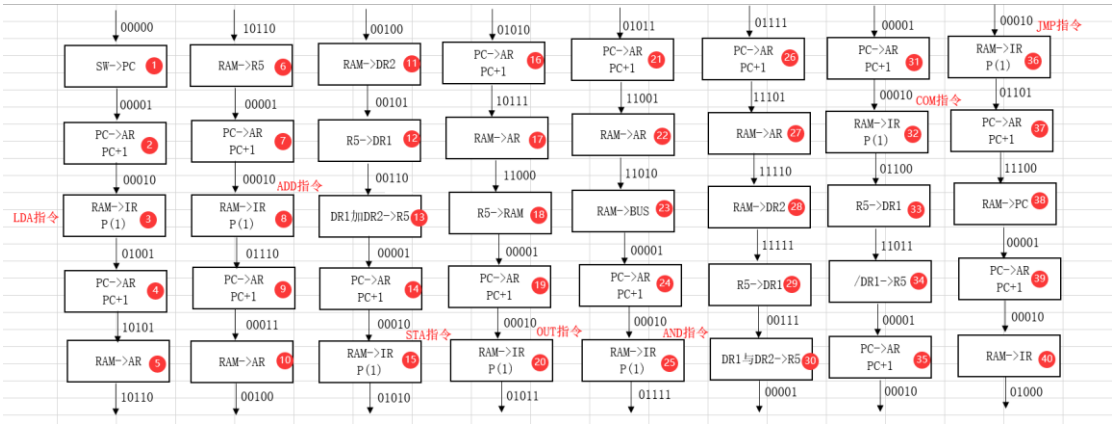


30. 当前微地址 00111，执行操作 DR1 与 DR2->R5，将运算结果 D0H 打入 R5，R5=D0H，下一微地址由 ROM 读出为 00001

31. 当前地址为 00001，此时 PC=0AH，执行操作 PC->AR，AR=0AH，PC+1，PC=0AH，下一微地址由 ROM 读出为 00010
32. 当前微地址为 00010，执行操作为 RAM->IR 译码器，0AH 地址的 RAM 单元根据用户程序看到是 80H（COM 指令），所以 OP(IR)=IR=80H=10000000，P=1，将 IR 的高三位 100 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01100
33. 当前微地址为 01100，执行操作 R5->DR1，DR1=R5=D0H，下一微地址为 11011
34. 当前微地址为 11011，执行操作/DR1->R5，将运算结果 FF2FH 打入 R5，R5=FF2FH，下微一地址由 ROM 读出为 00001
35. 当前地址为 00001，此时 PC=0BH，执行操作 PC->AR，AR=0BH，PC+1，PC=0CH，下一微地址由 ROM 读出为 00010
36. 当前微地址为 00010，执行操作为 RAM->IR 译码器，0BH 地址的 RAM 单元根据用户程序看到是 A0H（JMP 指令），所以 OP(IR)=IR=A0H=10100000，P=1，将 IR 的高三位 101 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01101
37. 当前微地址为 01101，此时 PC=0CH，执行操作 PC->AR，AR=0CH，PC+1，PC=0DH，下一微地址由 ROM 读出为 11100
38. 当前微地址为 11100，AR=0CH，执行操作 RAM->PC，将 0CH 对应 RAM 单元内容 11H 打入 PC，PC=11H，下一微地址由 ROM 读出为 00001

39. 当前地址为 00001，此时 PC=11H，执行操作 PC->AR，AR=11H，PC+1，PC=12H，下一微地址由 ROM 读出为 00010
40. 当前微地址为 00010，执行操作为 RAM->IR 译码器，11H 地址的 RAM 单元根据用户程序看到是 00H（无指令），所以 OP(IR)=IR=00H=00000000，进入强读，程序结束。

- 流程图



文字描述

1. 初始状态，当前微地址为 00000，操作码、PC、寄存器、暂存器初始化为 00，执行操作 SW->PC
2. 当前微地址为 00001，SW->PC 后，PC=00，执行操作 PC->AR，PC+1，这里先将 PC 打入 AR，所以 AR 为 00H，在执行 PC+1，所以 PC=01H，下一微地址由 ROM 读出为 02H 即 00010
3. 当前微地址为 00010，执行操作 RAM->IR 译码器，00H 地址的 RAM 单元根据用户程序看到是 20H（LDA 指令），所以 OP(IR)=IR=20H=00100000，P=1，将 IR 的高三位 001 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01001
4. 当前微地址为 01001，此时 PC=01H，执行 PC->AR，PC+1，先执行 PC->AR，所以 AR=01H，随后 PC+1=02H，下一微地址由 ROM 读出为 10101
5. 当前微地址为 10101，AR=01H，执行操作 RAM->AR，将 01H 地址 RAM 内存单元即用户程序 01H 地址单元内容打入 AR，AR 变为 0DH，下一微地址由 ROM 读出为 10110
6. 当前微地址为 10110，执行操作 RAM->R5，此时 AR=0DH，将 0DH 地址对应 RAM 单元内容传给 R5，根据用户程序，0DH 对应单元内容为 55H，R5=55H，下一微地址由 ROM 读出为 00001
7. 当前微地址为 00001，此时 PC=02H，执行操作 PC->AR，AR=02H，PC+1，PC=03H，下一微地址由 ROM 读出为 00010
8. 当前微地址为 00010，执行操作为 RAM->IR 译码器，02H 地址的 RAM 单元根据用户程序看到是 C0H（ADD 指令），所以 OP(IR)=IR=C0H=11000000，P=1，将 IR 的高三位 110 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01110
9. 当前微地址为 01110，此时 PC=03H，执行操作 PC->AR，AR=03H，PC+1，PC=04H，下一微地址由 ROM 读出为 00011
10. 当前微地址为 00011，AR=03H，执行操作 RAM->AR，将 AR 对应 RAM 单元的内容传给 AR，03H 对应的用户程序为 0EH，AR=OEH，下一地址由 ROM 读出为 00100

11. 当前微地址为 00100，AR=0EH，执行操作 RAM->DR2，将 AR=0EH 对应的 RAM 内存单元内容传给 DR2，0EH 单元内容为 8A，所以 DR2=8AH，下一地址由 ROM 读出为 00101
12. 当前微地址为 00101，执行 R5->DR1，故 DR1=R5=55H，下一微地址由 ROM 读出为 00110
13. 当前微地址为 00110，执行 DR1+DR2->R5，DR1=55H,DR2=8AH，故 R5=55H+8AH=DFH，下一微地址由 ROM 读出为 00001
14. 当前微地址为 00001，此时 PC=04H，执行操作 PC->AR，AR=04H，PC+1，PC=05H，下一微地址由 ROM 读出为 00010
15. 当前地址为 00010，执行操作为 RAM->IR 译码器，04H 地址的 RAM 单元根据用户程序看到是 40H（STA 指令），所以 OP(IR)=IR=40H=01000000，P=1，将 IR 的高三位 010 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01010
16. 当前地址为 01010，此时 PC=05H，执行操作 PC->AR，AR=05H，PC+1，PC=06H，下一微地址由 ROM 读出为 10111
17. 当前微地址为 10111，此时 AR=05H，执行 RAM->AR 将 05H 对应 RAM 内存单元内容打入 AR，根据用户程序可以看到 05H 单元为 10H，所以 AR=10H，下一微地址由 ROM 读出为 11000
18. 当前微地址为 11000，此时 AR=10H，执行 R5->RAM，将 R5 的值打入 AR 对应的 RAM 内存单元，根据内存单元情况可以看到 10H 单元内容已经变为 R5=DFH，下一微地址由 ROM 读出为 00001
19. 当前地址为 00001，此时 PC=06H，执行操作 PC->AR，AR=06H，PC+1，PC=07H，下一微地址由 ROM 读出为 00010
20. 当前微地址为 00010，执行操作为 RAM->IR 译码器，06H 地址的 RAM 单元根据用户程序看到是 60H（OUT 指令），所以 OP(IR)=IR=C0H=01100000，P=1，将 IR 的高三位 011 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01011
21. 当前微地址为 01011，此时 PC=07H，执行操作 PC->AR，AR=07H，PC+1，PC=08H，下一微地址由 ROM 读出为 11001
22. 当前微地址为 11001，此时 AR=07H，执行操作 RAM->AR，将 07H 对应 RAM 内存单元内容 10H 打入 AR，AR=10H，下一微地址由 ROM 读出为 11010
23. 当前微地址为 11010，此时 AR=10H，执行操作 RAM->BUS，将 10H 对应 RAM 内存单元 DFH 输出到总线 BUS，可以看到输出了 DF，下一微地址由 ROM 读出为 00001
24. 当前地址为 00001，此时 PC=08H，执行操作 PC->AR，AR=08H，PC+1，PC=09H，下一微地址由 ROM 读出为 00010
25. 当前微地址为 00010，执行操作为 RAM->IR 译码器，08H 地址的 RAM 单元根据用户程序看到是 E0H（AND 指令），所以 OP(IR)=IR=E0H=11100000，P=1，将 IR 的高三位 111 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01111
26. 当前微地址为 01111，此时 PC=09H，执行操作 PC->AR，AR=09H，PC+1，PC=0AH，下一微地址由 ROM 读出为 11101
27. 当前微地址为 11101，此时 AR=09H，执行操作 RAM->AR，将 09H 对应 RAM 内存单元 0FH 打入 AR，AR=0FH，下一微地址由 ROM 读出为 11110
28. 当前微地址为 11110，此时 AR=0FH，执行操作 RAM->DR2，将 0FH 对应 RAM 内存单元内容 F0H 打入 DR2，DR2=0FH，下一微地址由 ROM 读出为 11111
29. 当前微地址为 11111，执行操作 R5->DR1，所以 DR1=R5=DFH，下一微地址为 00111
30. 当前微地址 00111，执行操作 DR1 与 DR2->R5，将运算结果 D0H 打入 R5，R5=D0H，下一微地址由 ROM 读出为 00001
31. 当前地址为 00001，此时 PC=0AH，执行操作 PC->AR，AR=0AH，PC+1，PC=0AH，下一微地址由 ROM 读出为 00010
32. 当前微地址为 00010，执行操作为 RAM->IR 译码器，0AH 地址的 RAM 单元根据用户程序看到是 80H（COM 指令），所以 OP(IR)=IR=80H=10000000，P=1，将 IR 的高三位 100 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01100
33. 当前微地址为 01100，执行操作 R5->DR1，DR1=R5=D0H，下一微地址为 11011
34. 当前微地址为 11011，执行操作/DR1->R5，将运算结果 FF2FH 打入 R5，R5=FF2FH，下微一地址由 ROM 读出为 00001
35. 当前地址为 00001，此时 PC=0BH，执行操作 PC->AR，AR=0BH，PC+1，PC=0CH，下一微地址由 ROM 读出为 00010
36. 当前微地址为 00010，执行操作为 RAM->IR 译码器，0BH 地址的 RAM 单元根据用户程序看到是 A0H（JMP 指令），所以 OP(IR)=IR=A0H=10100000，P=1，将 IR 的高三位 101 传到原 ROM 读出的下一微地址 01000 的低三位即下一微地址变为 01101
37. 当前微地址为 01101，此时 PC=0CH，执行操作 PC->AR，AR=0CH，PC+1，PC=0DH，下一微地址由 ROM 读出为 11100
38. 当前微地址为 11100，AR=0CH，执行操作 RAM->PC，将 0CH 对应 RAM 单元内容 11H 打入 PC，PC=11H，下一微地址由 ROM 读出为 00001
39. 当前地址为 00001，此时 PC=11H，执行操作 PC->AR，AR=11H，PC+1，PC=12H，下一微地址由 ROM 读出为 00010
40. 当前微地址为 00010，执行操作为 RAM->IR 译码器，11H 地址的 RAM 单元根据用户程序看到是 00H（无指令），所以 OP(IR)=IR=00H=00000000，进入强读，程序结束。

模拟器执行复合运算的最后结果

```
---OVER---
PC:12 OP:00 M_ADDR:08 M_NXT_ADDR:08
AR:11 DR1:D0 DR2:F0 R5:FF2F
```

数据 A=55H(Mem[0D]),B=8AH(Mem[0E]),C=0F(Mem[0F])H 实现 NOT(A 加 B AND C)

最后结果为 FF2FH。模拟器进行的复合运算为/((Mem[0D]+Mem[0E]) and Mem[0F])，过程为先计算(Mem[0D]+Mem[0E]，并将结果写入 Mem[10]，并将其输出，再计算(Mem[0D]+Mem[0E]) and Mem[0F]，最后计算/((Mem[0D]+Mem[0E]) and Mem[0F])并将结果 FF2FH 保存在 R5 中。

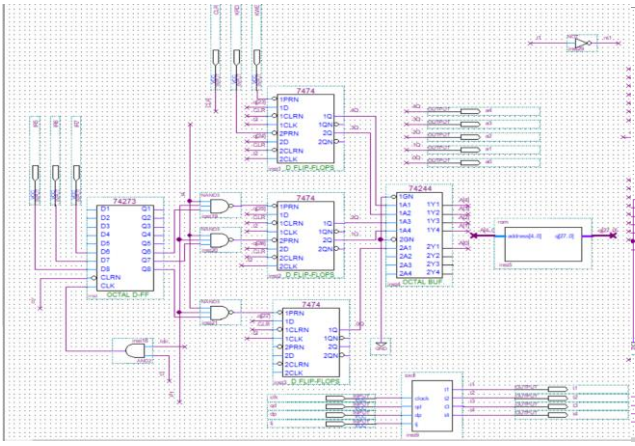
微控 2

- ROM 模块 VHDL 语言描述

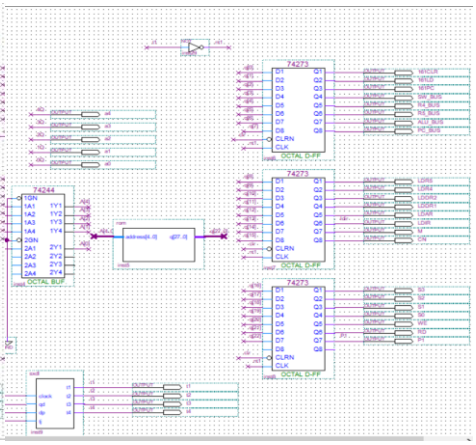
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY rom IS
PORT
(
    address : IN    STD_LOGIC_VECTOR (4 DOWNTO 0);
            q : OUT STD_LOGIC_VECTOR (27 DOWNTO 0));
END rom;
ARCHITECTURE SYN OF rom IS
SIGNAL sub_wire0 : STD_LOGIC_VECTOR (27 DOWNTO 0);
BEGIN
sub_wire0<=
"101011110000000100000000001"    WHEN address=   "00000"  ELSE
"111111100000100100000000010"    WHEN address=   "00001"  ELSE
"100111110000010100000101000"    WHEN address=   "00010"  ELSE
"101011110000000100000010011"    WHEN address=   "01000"  ELSE
"111111100000100100000010101"    WHEN address=   "01001"  ELSE
"100111110000100100001010110"    WHEN address=   "10101"  ELSE
"100111111000000100000100001"    WHEN address=   "10110"  ELSE
"111111100000100100000010111"    WHEN address=   "01010"  ELSE
"100111110000100100001011000"    WHEN address=   "10111"  ELSE
"100110110000000100010000001"    WHEN address=   "11000"  ELSE
"111111100000100100000011001"    WHEN address=   "01011"  ELSE
"100111110000100100001011010"    WHEN address=   "11001"  ELSE
"100111110000000100000100001"    WHEN address=   "11010"  ELSE
"100110110001000100000011011"    WHEN address=   "01100"  ELSE
"100111011000001000000000001"    WHEN address=   "11011"  ELSE
"111111100000100100000011100"    WHEN address=   "01101"  ELSE
"101111110000000100000100001"    WHEN address=   "11100"  ELSE
"111111100000100100000000011"    WHEN address=   "01110"  ELSE
"100111110000100100001000100"    WHEN address=   "00011"  ELSE
"100111110010000100000100101"    WHEN address=   "00100"  ELSE
"100110110001000100000000110"    WHEN address=   "00101"  ELSE
"100111011000001100100000001"    WHEN address=   "00110"  ELSE
"111111100000100100000011101"    WHEN address=   "01111"  ELSE
"100111110000100100001011110"    WHEN address=   "11101"  ELSE
"100111110010000100000101111"    WHEN address=   "11110"  ELSE
"100110110001000100000000111"    WHEN address=   "11111"  ELSE
"100111011000000010110000001"    WHEN address=   "00111"  ELSE
"100011110000000100000100001"    WHEN address=   "10010"  ELSE
"111111100000100100000010100"    WHEN address=   "10011"  ELSE
"100111110000000000000010011"    WHEN address=   "10100"  ELSE
"101011110000000100000000001"    WHEN address=   "10000"  ELSE
"111111100000100100000010010"    ;
 q    <= sub_wire0(27 DOWNTO 0);
END SYN;
```

其中，address 为当前微地址，q 为下一微指令，都有 logic 向量定义。输入为微地址 address，输出为下一微指令 q，其中低 5 为下一微地址，其余为微指令发出的控制信号的值，共 28 位。采用 when——else 语句来控制当前微地址并编写对应的下一微指令，从而完成映射关系，映射关系根据流程图编写微码表完成映射，最终最终完成 ROM 设计。

微程序控制器电路图截图



ROM 的输出以下部分



ROM 的输出以上部分

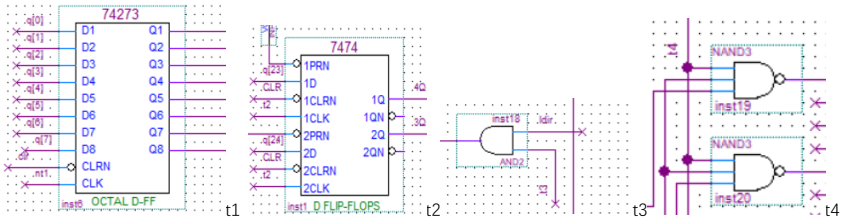
- 控制器电路图说明——控制器由以下几个部件构成

控制存储器：对应电路图中中间的 ROM 模块　微命令寄存器：对应电路中最右边的三个 74273 芯片　微地址寄存器：对应电路图中靠左部分的三个 7474 芯片　指令寄存器：对应电路图中最左边的一个 74273　地址转移逻辑：对应电路图靠左的 3 个 3 输入与非门　时序电路：对应电路图中最下方的 sxdl 模块　输入——时钟信号 clk，清零信号 CLR，强读信号 KRD，强写信号 KWE，指令的高三位 IR7 IR6 IR5，以及时序电路控制信号 qddb、tj

输出——28 位微指令、时序 t1-t4

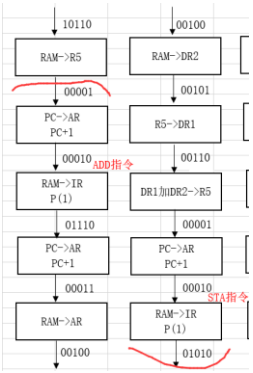
- 分析机器指令 ADD 的执行过程

首先根据电路图分析每条微指令的执行过程



T1 连接着微命令寄存器（最右边三块 74273 芯片）的时钟端，T1 时刻，将当前微命令的控制信号输出控制电路。T2 连接着微地址寄存器的时钟端，T2 时刻，将当前微指令的低五位（后继微地址）输入 74244 数据缓冲器再经其输出到 ROM，由 ROM 读出下一微指令。T3 与 LDIR 控制信号的与连接到指令寄存器（最左边的 74273）的时钟端，T3 时刻，若 LDIR=1，则指令寄存器将指令 IR 的高三位输出到地址转移逻辑(3 个 3 输入与非门)，否则保持输出不变。T4、P1 以及 IR 的高三位连接着 3 个 3 输入与非门的输入端，当 T4 时刻到来时，若 P=1，则将 IR 的高三位取代 T2 时刻读出的后继微地址的低三位，重新映射出后继微地址，由 ROM 重新读出新的微命令，实现重新映射地址跳转，否则后继微地址不变不跳转（顺序执行、绝对地址跳转）。对应流程图中每一条微指令都是如此进行。

ADD 指令为双字节指令，需要两个内存单元来存放，第一个内存单元为指令码，第二个内存单元为数据存放在内存的地址。



根据流程图可以看到，微地址 00001 对应操作 PC->AR，PC+1，将 PC 打入 AR，使 AR 指向内存中存放 ADD 指令的地址，随后 PC+1，指向下一条指令或者数据地址。执行完 00001 后，进入微地址 00010 对应操作，ldir=1，将 AR 对应的内存单元 ADD 指令码 C0H 读出来并打入 IR 寄存器，并且 P=1，微地址跳转重新映射，将 IR 的高三位 110 传入由 ROM 读出的原下一微指令 01000 的低三位，下一微地址变为 01110。01110 微地址对应操作将 AR 指向 ADD 指令吗的下一单元即数据存放的地址，PC+1，下一微地址为 00100，00100 执行操作 RAM->AR，将 AR 指向数据存放单元的地址，下一微地址为 00100，将 AR 对应单元数据打入 DR2，下一微地址为 00101，00101 执行操作将 R5 数据打入 DR1，下一微地址为 00110，00110 执行操作将 DR1、DR2 送入 ALU 并计算出 DR1 加 DR2 并将结果送入 R5，下一微地址重新回到公共微指令 00001，随后再执行公共微指令 00010 继续读出下一条机器指令并执行直至执行完所有指令。

- 控制器实验总结

在之前的数电实验 CPU 以及数据通路实验中，模型机以及数据通路中的指令、控制信号都由我们自己手动输入，并通过波形仿真验证。且内存中中存放的都为数据并没有指令，与真正意义上自动的模型机相差甚远，其关键部分就在与这次实验设计的控制器模块上。控制器是模型机中的核心，其发出的控制信号控制模型机各部件相互配合、有条理完成各个指令，让各部件有机的形成模型机整体。

而设计控制器的关键在于理解微指令、理解控制器电路图工作原理、ROM 的正确设计（一一对应）以及其与模型机整体的联系。

通过控制器模块，模型机能真正实现自动执行指令，而用户要做的只是将指令以及数据输入内存中，不再需要手动输入指令以及控制信号便模型机便可以自动的逐一执行指令，让模型机在时序脉冲下自动运行，从而真正实现设计一台模型机。

通过这次实验，拉近了我与模型机底层的距离，真正看到了模型机运行时的时序控制逻辑，大大加深了我对模型机工作的原理的理解。