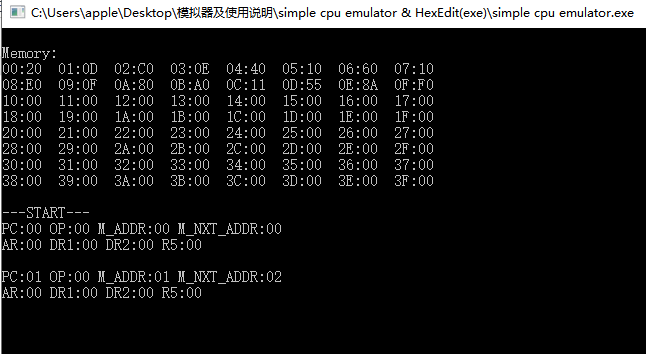
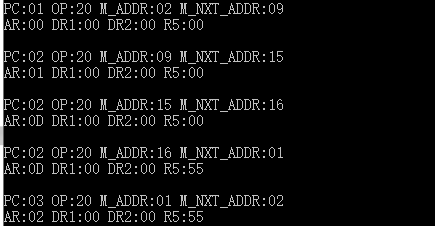
# 实验日志-控制器

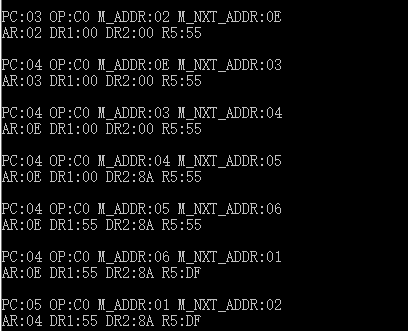
1. 模拟器中指令的执行情况载图
2. 开机



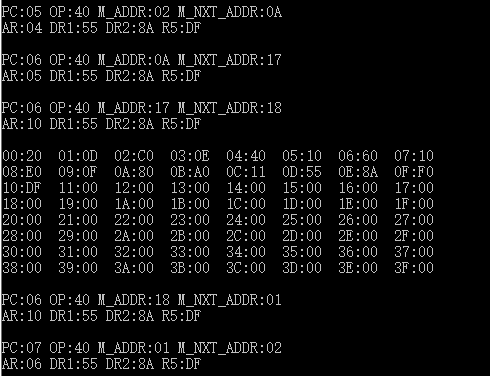
1. 开始为初始状态，所有数值均为0
2. 执行当前地址M\_ADDR==01H下的指令：PC🡪AR; PC+1；
3. 经过此步骤PC == 01H; M\_NXT\_ADDR == 02H
4. 读取LDA指令



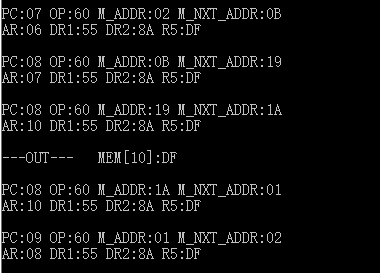
1. 执行当前地址M\_ADDR==02H下的指令：RAM🡪IR;功能跳转到OP==02H时的操作LDA；经过此步骤，OP == 20H; M\_NXT\_ADDR == 09H
2. 执行当前地址M\_ADDR==09H下的指令：PC🡪AR; PC+1;
3. 经过此步骤，PC == 02H; M\_NXT\_ADDR == 15H; AR == 01H
4. 执行当前地址M\_ADDR==15H下的指令：RAM🡪AR; 此时01H地址的值为0DH，因此AR被赋值为0DH；经过此步骤，M\_NXT\_ADDR == 16H; AR == 0DH
5. 执行当前地址M\_ADDR==16H下的指令：RAM🡪R5; 此时 0DH地址的值为55H，因此R5被赋值为55H；经过此步骤，M\_NXT\_ADDR == 01H; R5 == 55H
6. 执行当前地址M\_ADDR==01H下的指令：PC🡪AR; PC+1；
7. 经过此步骤，PC == 03H; M\_NXT\_ADDR == 02H; AR == 02H
8. 读取并执行指令ADD
9. 执行当前地址M\_ADDR==02H下的指令：RAM🡪IR;功能跳转到OP==C0H时的操作ADD；经过此步骤，OP == C0H; M\_NXT\_ADDR == 0E
10. 执行当前地址M\_MXT\_ADDR == 0E下的指令：PC🡪AR; PC+1;
11. 经过此步骤，PC == 04H; M\_NXT\_ADDR == 03H; AR == 03H
12. 执行当前地址M\_ADDR==03H下的指令：RAM🡪AR; 此时03H地址的值为0EH，因此AR被赋值为0EH



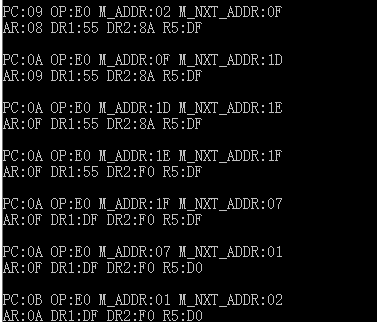
1. 经过此步骤，M\_NXT\_ADDR == 04H; AR == 0EH
2. 执行当前地址M\_ADDR==04H下的指令：RAM🡪DR2; 此时 0EH地址的值为8AH，因此DR2被赋值为8AH；经过此步骤，M\_NXT\_ADDR == 05H; DR2 == 8AH
3. 执行当前地址M\_ADDR==05H下的指令：R5🡪DR1; 此时 R5的值为55H，因此DR1被赋值为55；经过此步骤，M\_NXT\_ADDR == 06H; DR1 == 55H
4. 执行当前地址M\_ADDR==06H下的指令：ALU🡪R5; 此时 ALU进行加法运算后的的值为DFH，因此R5被赋值为DFH；经过此步骤，M\_NXT\_ADDR == 01H; R5 == DFH
5. 执行当前地址M\_ADDR==01H下的指令：PC🡪AR; PC+1;
6. 经过此步骤，PC == 05H; M\_NXT\_ADDR == 02H; AR == 04H
7. 读取并执行指令STA



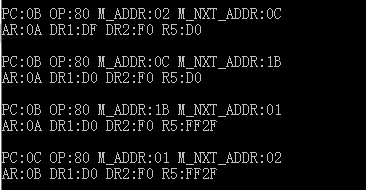
1. 执行当前地址M\_ADDR==02H下的指令：RAM🡪IR;功能跳转到OP==40H时的操作STA；经过此步骤，OP == 40H; M\_NXT\_ADDR == 0AH
2. 执行当前地址M\_ADDR==0AH下的指令：PC🡪AR; PC+1;
3. 经过此步骤，PC == 06H; M\_NXT\_ADDR == 17H; AR == 05H
4. 执行当前地址M\_ADDR==17H下的指令：RAM🡪AR; 此时05H地址的值为10H，因此AR被赋值为10H；经过此步骤，M\_NXT\_ADDR == 18H; AR == 10H
5. 执行当前地址M\_ADDR==18H下的指令：R5🡪RAM; 此时 R5的值为DFH，因此RAM的10E位置被赋值为DFH；经过此步骤，M\_NXT\_ADDR == 01H; RAM中 地址为10E的值变为DFH
6. 执行当前地址M\_ADDR==01H下的指令：PC🡪AR; PC+1;
7. 经过此步骤，PC == 07H; M\_NXT\_ADDR == 02H; AR == 06H
8. 读取并执行指令OUT
9. 执行当前地址M\_ADDR==02H下的指令：RAM🡪IR;功能跳转到OP==60H时的操作OUT；经过此步骤，OP == 60H; M\_NXT\_ADDR == 0BH
10. 执行当前地址M\_ADDR==0BH下的指令：PC🡪AR; PC+1;
11. 经过此步骤，PC == 08H; M\_NXT\_ADDR == 19H; AR == 07H
12. 执行当前步骤M\_ADDR==19H下的指令：RAM🡪AR;此时07H地址的值为10H，因此AR被赋值为10H；经过此步骤，M\_NXT\_ADDR == 1AH; AR == 10H
13. 执行当前步骤M\_ADDR == 1AH下的指令：RAM🡪BUS;此时10H地址的值为DFH，因此向总线输出DF；经过此步骤，M\_NXT\_ADDR == 01H
14. 执行当前步骤M\_ADDR == 01H下的指令：PC🡪AR; PC+1;
15. 经过此步骤，PC = 09; M\_NXT\_ADDR == 02H; AR == 08H



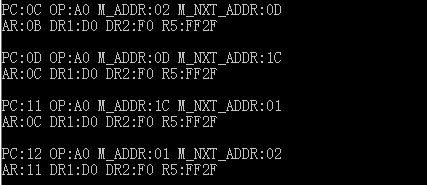
1. 读取并执行指令AND



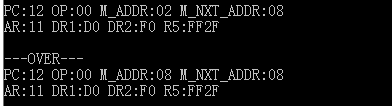
1. 执行当前地址M\_ADDR==02H下的指令：RAM🡪IR;功能跳转到OP==E0H时的操作AND；经过此步骤，OP == E0H; M\_NXT\_ADDR == 0F
2. 执行当前地址M\_MXT\_ADDR == 0F下的指令：PC🡪AR; PC+1;
3. 经过此步骤，PC == 0AH; M\_NXT\_ADDR == 1DH; AR == 09H
4. 执行当前地址M\_ADDR==1DH下的指令：RAM🡪AR; 此时09H地址的值为0FH，因此AR被赋值为0FH；经过此步骤，M\_NXT\_ADDR == 1EH; AR == 0FH
5. 执行当前地址M\_ADDR==1EH下的指令：RAM🡪DR2; 此时 0FH地址的值为F0H，因此DR2被赋值为F0H；经过此步骤，M\_NXT\_ADDR == 1FH; DR2 == F0H
6. 执行当前地址M\_ADDR==1FH下的指令：R5🡪DR1; 此时 R5的值为DFH，因此DR1被赋值为DFH；经过此步骤，M\_NXT\_ADDR == 07H; DR1 == DFH
7. 执行当前地址M\_ADDR==07H下的指令：ALU🡪R5; 此时 ALU进行AND运算后的的值为D0H，因此R5被赋值为D0H；经过此步骤，M\_NXT\_ADDR == 01H; R5 == D0H
8. 执行当前地址M\_ADDR==01H下的指令：PC🡪AR; PC+1;
9. 经过此步骤，PC == 0BH; M\_NXT\_ADDR == 02H; AR == 0AH
10. 读取并执行指令COM



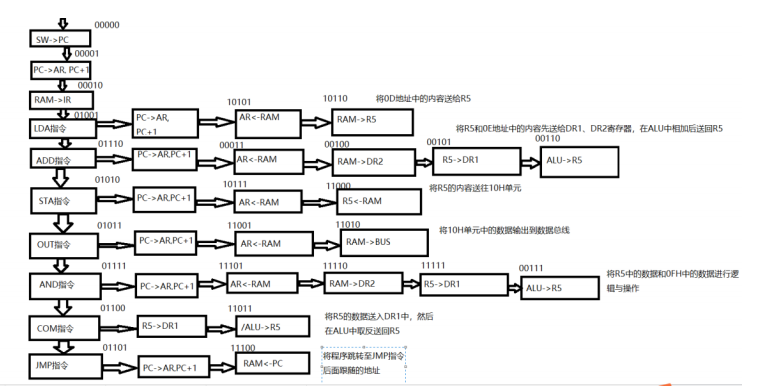
1. 执行当前地址M\_ADDR==02H下的指令：RAM🡪IR;功能跳转到OP==80H时的操作COM；经过此步骤，OP == 80H; M\_NXT\_ADDR == 0C
2. 执行当前地址M\_ADDR==0CH下的指令：R5🡪DR1,R5的值为D0H，因此DR1被赋值为D0H；经过此步骤，M\_MXT\_ADDR == 1BH; DR1 == D0H
3. 执行当前地址M\_ADDR==1BH下的指令：/ALUR5🡪R5,R5的值变为FF2FH
4. 经过此步骤，M\_MXT\_ADDR == 01H; R5 == FF2FH
5. 执行当前地址M\_ADDR==01H下的指令：PC🡪AR; PC+1;
6. 经过此步骤，PC == 0CH; M\_NXT\_ADDR == 02H; AR == 0BH
7. 读取并执行指令JMP



1. 执行当前地址M\_ADDR==02H下的指令：RAM🡪IR;功能跳转到OP==A0H时的操作JMP；经过此步骤，OP == A0H; M\_NXT\_ADDR == 0D
2. 执行当前地址M\_ADDR==0DH下的指令：PC🡪AR; PC+1;
3. 经过此步骤，PC == 0DH; M\_NXT\_ADDR == 1CH; AR == 0CH
4. 执行当前地址M\_ADDR==1CH下的指令：RAM🡪PC; 此时0CH地址的值为11H，因此PC被赋值为11H；经过此步骤，PC == 11; M\_NXT\_ADDR == 01H
5. 执行当前地址M\_ADDR==01H下的指令：PC🡪AR; PC+1;
6. 经过此步骤，PC == 12H; M\_NXTADDR == 02H; AR == 11H
7. 结束程序



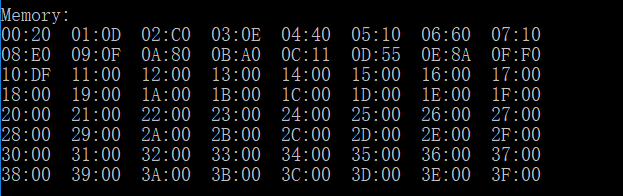
1. 执行当前地址M\_ADDR==02H下的指令：：RAM🡪IR; 已无对应功能，程序结束
2. 流程图



模拟器执行的指令是由上往下的执行顺序，横向顺序是每一条指令对应的具体机器操作，由于指令是在地址中连续存储的，所以模拟器也是连续执行的，每一条指令后面都

有文字描述具体的操作

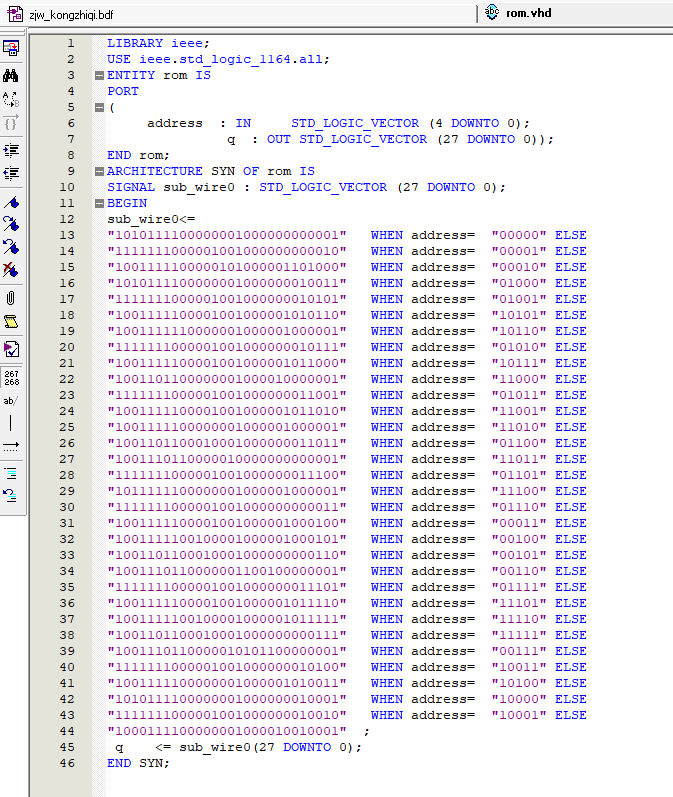
1. 模拟器执行复合运算的最后结果载图



最后运行结果，将ADD指令计算出的值写入10地址，故10:00->10:DF

1. ROM模块的设计

说明：根据不同的指令需求来设置信号，如在address为“00001”时，要执行PC🡪AR;PC+1;的指令，按照该需求分别设置28位信号，同时其接下来跳变为“00010”根据跳变来一步步设计信号，直到包括了所有的信号。



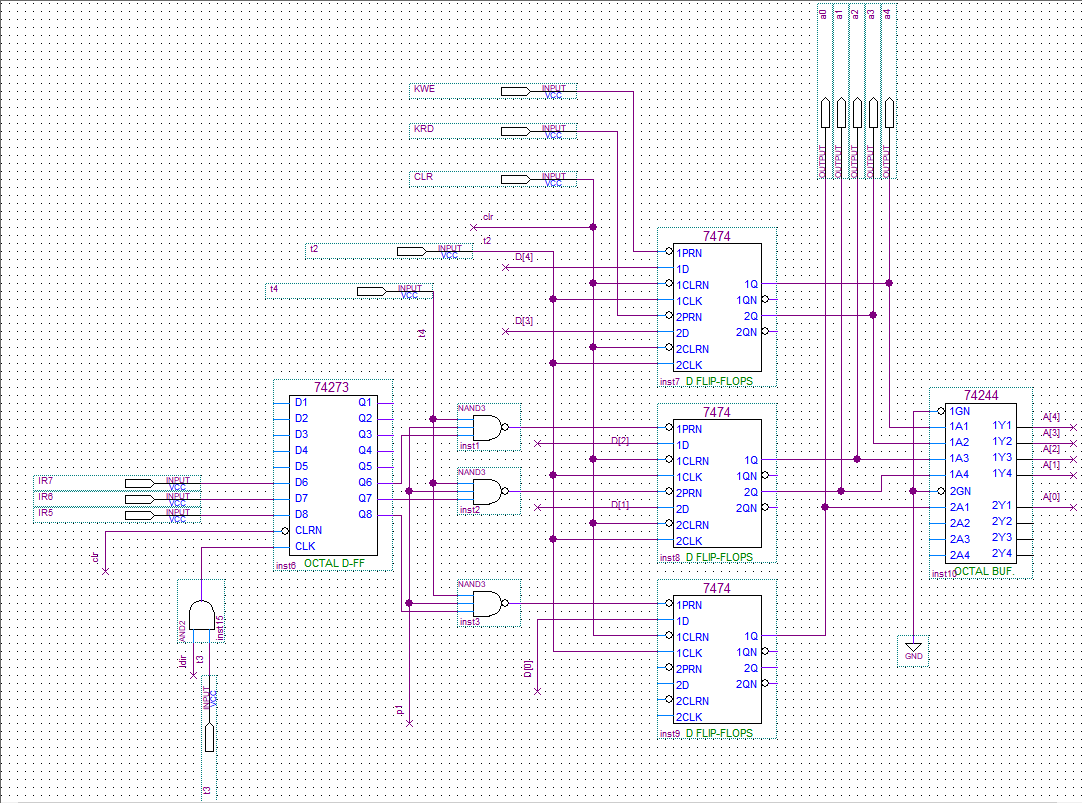
1. 微程序控制器电路图截图

1.ROM输出以下

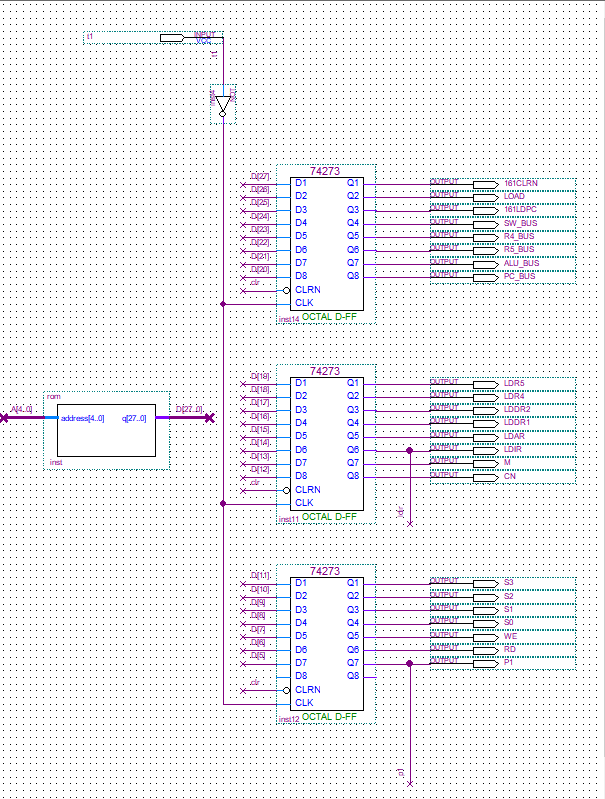
最左边的74273模块用于保存输入的IR7,IR6,IR5，这三个输入在001~111个输入情况下分别表示LDA,STA,OUT,COM,JMP,ADD,AND等命令。74273的时钟信号由节拍电路的T3和LDIR控制，即取指令。

74273模块和右边的三个三输入与非门（这三个与非门为地址转移逻辑）将IR7,IR6,IR5送给中间的三个7474模块，这三个模块为微地址寄存器，会根据输入的指令以及输入的ROM输出的4-0为（即下址），将下址将下址送给后面的74244模块。微地址寄存器的时钟信号由节拍电路的T2控制，即给出下址。

ROM左边的74244模块8位缓冲器，其作用是将左边三个7474送过来的下址转变为当前地址并送给ROM。



2.ROM输出以上



ROM右边的三个74273模块为微命令寄存器，根据ROM输出的微指令来控制信号。

上面的74273控制161CLRN,LOAD,161LDPC,PC\_BUS,LDADR,WE,RD,SW\_BUS。

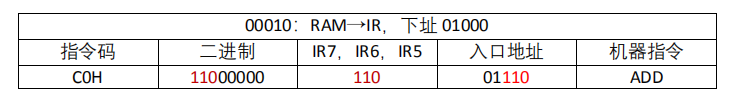
中间的74273控制LDRS,LDDR1,R5\_BUS,M,S3,S2,S1,S0。

下面的74273控制P1,LDIR,LDDR2,R4\_BUS,LDR4,ALU\_BUS,CN。

微命令寄存器的时钟部分为节拍电路的T1，即给出当前微地址的微命令。

1. 分析机器指令的执行过程

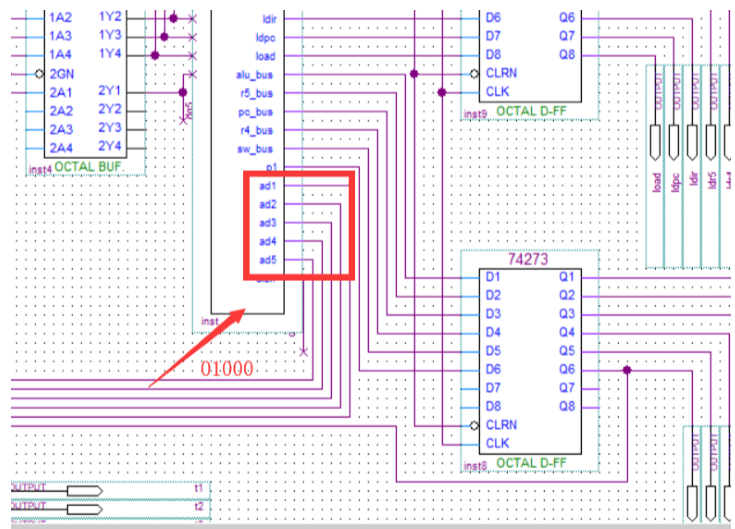
以ADD指令为例



1. T1/T2

微指令寄存器即3个34273 送出微指令P1=1

绝对地址01000



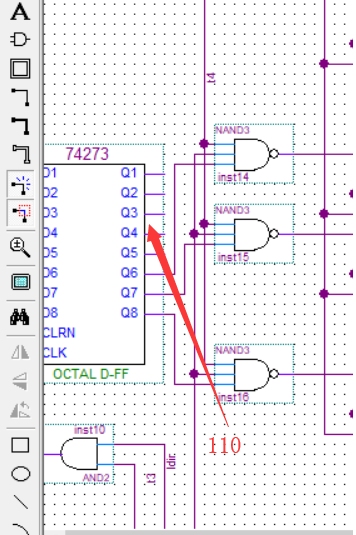
1. T2

微地址寄存器读取

绝对地址01000

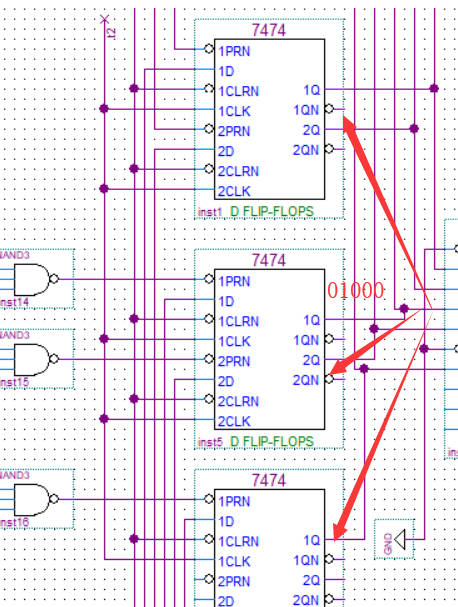
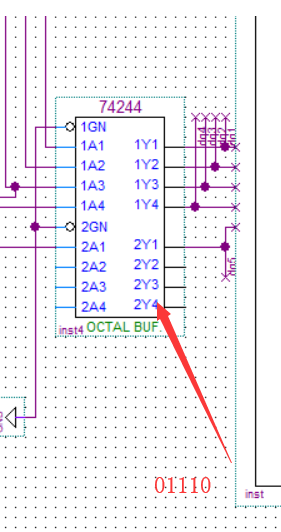
1. T3

由数据通路中的RAM读取 IR【7..5】到IR (74273)



1. T4

产生新的下址到74244中 即从01000变为01110 进入ADD操作的流程



具体分析

1. 构造初始化状态，使CLR == 0
2. 启动时序电路，使qd == 0
3. 执行00000对应的操作：SW🡪PC,并产生下址00001此时SW\_BUS==0
4. 执行00001对应的操作：PC🡪AR,PC+1,并产生下址00010，此时pc\_BUS==0，LDAR == 1；PC\_SEL == 111
5. 执行00010对应的操作：RAM🡪IR 跳转到了LDA指令，产生下址01001
6. 执行01001对应的操作：PC🡪AR,PC+1,产生下址10101
7. 执行01001对应的操作：AR🡪RAM,产生下址10110