实验目标

1.1 文字描述

实验目的： 设计一个8位模型计算机，通过模型机理解计算机整机的结构和功能，特别是CPU、存储器、中断控制器和接口的结构、实现逻辑及各部件之间的接口关系。

实验原理：

微操作和微命令：

微操作：一条机器指令可以分解成一系列的微操作，这些微操作是计算机中最基本、不可再细分的操作。

微命令：在微程序控制的计算机中，控制部件向执行部件发出的各种控制命令称为微命令。它们是控制序列的最小单位，与微操作一一对应。微命令是执行微操作的控制信号，控制微操作的执行过程。

微指令、微地址和微周期：微指令：是若干微命令的集合，分为操作控制字段和顺序控制字段。操作控制字段（微操作码字段）：用于产生某一步操作所需的控制信号。

顺序控制字段（微地址码字段）：用于控制下一条要执行的微指令地址的产生。

微地址：存放微指令的控制存储器单元的地址。

微周期：从控制存储器中读取一条微指令并执行相应的微操作所需的时间。

程序与微程序：程序：指令的有序集合，用于完成特定的功能。

微程序：微指令的有序集合，一条指令的功能由一段微程序来实现。

主存储器与控制存储器：主存储器：用于存放程序和数据，位于CPU外部，用RAM实现。

控制存储器：用于存放微程序，位于CPU内部，用ROM实现。

地址寄存器(MAR)与微地址寄存器(CMAR)：地址寄存器(MAR)：存放要访问的存储器地址，位于CPU内部，CU外部，对应指令。

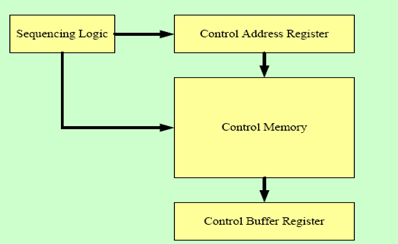
微地址寄存器(CMAR)：存放要访问的控制存储器地址，位于CPU内部，CU内部，对应微指令。

指令寄存器(IR)与微指令寄存器(CMDR)：指令寄存器(IR)：存放机器指令，位于CPU内部，CU外部，对应指令。

微指令寄存器(CMDR)：存放微指令，位于CPU内部，CU内部，对应微指令。

微指令控制单元： 操作控制信号预先存储在专门的控制存储器（控制存储器）中，每条机器指令编写成一个微程序，这些微程序存储在控制存储器中，通过类似于寻址用户程序机器指令的方式来寻址每个微程序中的微指令。

草图



指令总控制信号如下：

Bits in Control Memory Micro-operation Meaning

C0~C7 / Branch Addresses

C8 PC←0 Clear PC

C9 PC←PC+1 Increment PC

C10 PC←MBR[7..0] MBR[7..0] to PC

C11 ACC←0 Clear ACC

C12--C15 ALU CONTROL Control operations of ALU

C16 R Read data from Memory to MBR

C17 W Write data to Memory

C18 MAR←MBR[7..0] MBR[7..0] to MAR as address

C19 MAR←PC PC value to MAR

C20 MBR←ACC ACC value to MBR

C21 IR←MBR[15..8] MBR[15..8] to IR as opcode

C22 BR←MBR Copy MBR to BR

C23 CAR←CAR+1 Increment CAR

C24 CAR←C0~C7 C7~C0 to CAR

C25 CAR←OPCODE+CAR Add OP to CAR

C26 CAR←0 Reset CAR

C27--C31 Not use -----------

1.2实验设计

1.总体设计

单地址指令格式用于我们简单的CPU 设计。指令字包括操作码和地址码俩个部分，其中操作码用于定义指令的功能，比如指令是加法运算还是减法运算等，地址码在本次设计中又分为直接寻址和立即寻址两种，其中直接寻址的地址码存放的是操作数在内存中的地址，立即寻址的地址码即为操作数。

2.微指令设计

0 : 00810000;        R←1,         CAR←CAR+1

1 : 00A00000;       OP←MBR[15..8],CAR←CAR+1

2 : 02000000;       CAR←CAR+OP

3 : 01000014;       CAR←14H

4 : 01000019;       CAR←19H

5 : 0100001E;       CAR←1EH

6 : 01000023;       CAR←23H

7 : 01000041;       CAR←41H

8 : 01000028;       CAR←28H

9 : 0100002D;       CAR←2DH

a : 01000032;       CAR←32H

b : 01000037;       CAR←37H

c : 0100003C;       CAR←3CH

d : 01000046;       CAR←46H

e : 0100004B;       CAR←4BH

f : 00000000;

   …       ……

14 : 00840000;    MAR←MBR[7..0], CAR←CAR+1     ------STORE

15 : 00920200;    MBR←ACC, PC←PC+1,W←1,CAR←CAR+1

16 : 04080000;    CAR←0, MAR←PC

17 : 00000000;

18 : 00000000;

19 : 00840000;    MAR←MBR[7..0], CAR←CAR+1      ------LOAD

1a : 00810A00;    PC←PC+1,R←1,ACC←0,CAR←CAR+1

1b : 00C03000;    BR←MBR,ACC←ACC+BR, CAR←CAR+1

1c : 04080000;    CAR←0

1d : 00000000;

1e : 00840000;    MAR←MBR[7..0], CAR←CAR+1  ----------ADD

1f : 00810200;    PC←PC+1,R←1,CAR←CAR+1

20 : 00C03000;    BR←MBR,ACC←ACC+BR, CAR←CAR+1

21 : 04080000;    CAR←0

22 : 00000000;

23 : 00840000;    MAR←MBR[7..0], CAR←CAR+1  ----------SUB

24 : 00810200;    PC←PC+1,R←1,CAR←CAR+1

25 : 00C04000;    BR←MBR,ACC←ACC-BR, CAR←CAR+1

26 : 04080000;    CAR←0

27 : 00000000;

28 : 00840000;    MAR←MBR[7..0], CAR←CAR+1   ---------AND

29 : 00810200;    PC←PC+1,R←1,CAR←CAR+1

2a : 00C06000;    BR←MBR,ACC←ACC AND BR,CAR←CAR+1

2b : 04080000;     CAR←0

2c : 00000000;

2d : 00840000;     MAR←MBR[7..0], CAR←CAR+1  ---------OR

2e : 00810200;     PC←PC+1,R←1,CAR←CAR+1

2f : 00C07000;     BR←MBR,ACC←ACC OR BR, CAR←CAR+1

30 : 04080000;     CAR←0

31 : 00000000;

32 : 00840000;     MAR←MBR[7..0], CAR←CAR+1  ----------NOT

33 : 00808200;     PC←PC+1, ACC←NOT ACC,CAR←CAR+1

34 : 04080000;     CAR←0

35 : 00000000;

36 : 00000000;

37 : 00840000;     MAR←MBR[7..0], CAR←CAR+1  ----------SHIFTR

38 : 08092000;     PC←PC+1, ACC←SHIFT ACC to Right 1 bit,CAR←CAR+1

39 : 04080000;     CAR←0

3a : 00000000;

3b : 00000000;

3c : 00840000;    MAR←MBR[7..0], CAR←CAR+1  -----------SHIFTL

3d : 0080A200;    PC←PC+1, ACC←SHIFT ACC to Left 1 bit,CAR←CAR+1

3e : 04080000;    CAR←0

3f : 00000000;

40 : 00000000;

41 : 00840000;    MAR←MBR[7..0], CAR←CAR+1  -----------JMPGEZ

42 : 00805000;    CAR←CAR+1,(if ACC>0 then PCjmp<='1', PCout<=MBR\_PC)

43 : 04080000;    CAR←0

44 : 00000000;

45 : 00000000;

46 : 00840000;    MAR←MBR[7..0], CAR←CAR+1  ------------MPY

47 : 00810200;    PC←PC+1,R←1,CAR←CAR+1

48 : 00C0B000;   BR←MBR,ACC←ACC\*BR, CAR←CAR

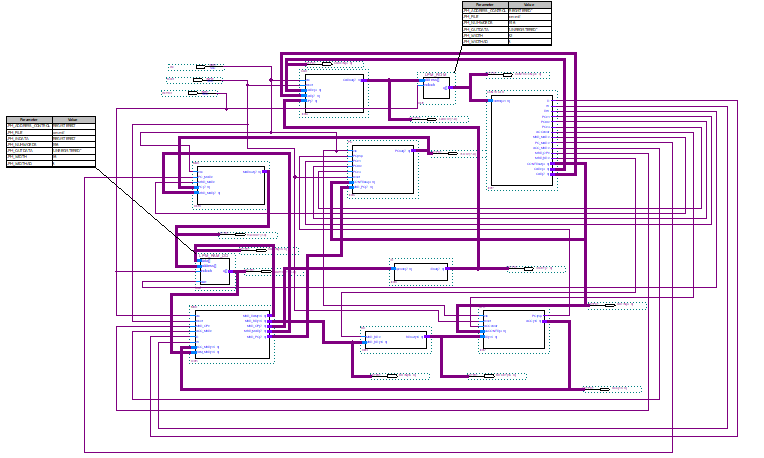
49 : 04080000;    CAR←0

4a : 00000000;

4b : 0100004B;    CAR←4BH  ------------------------------HALT

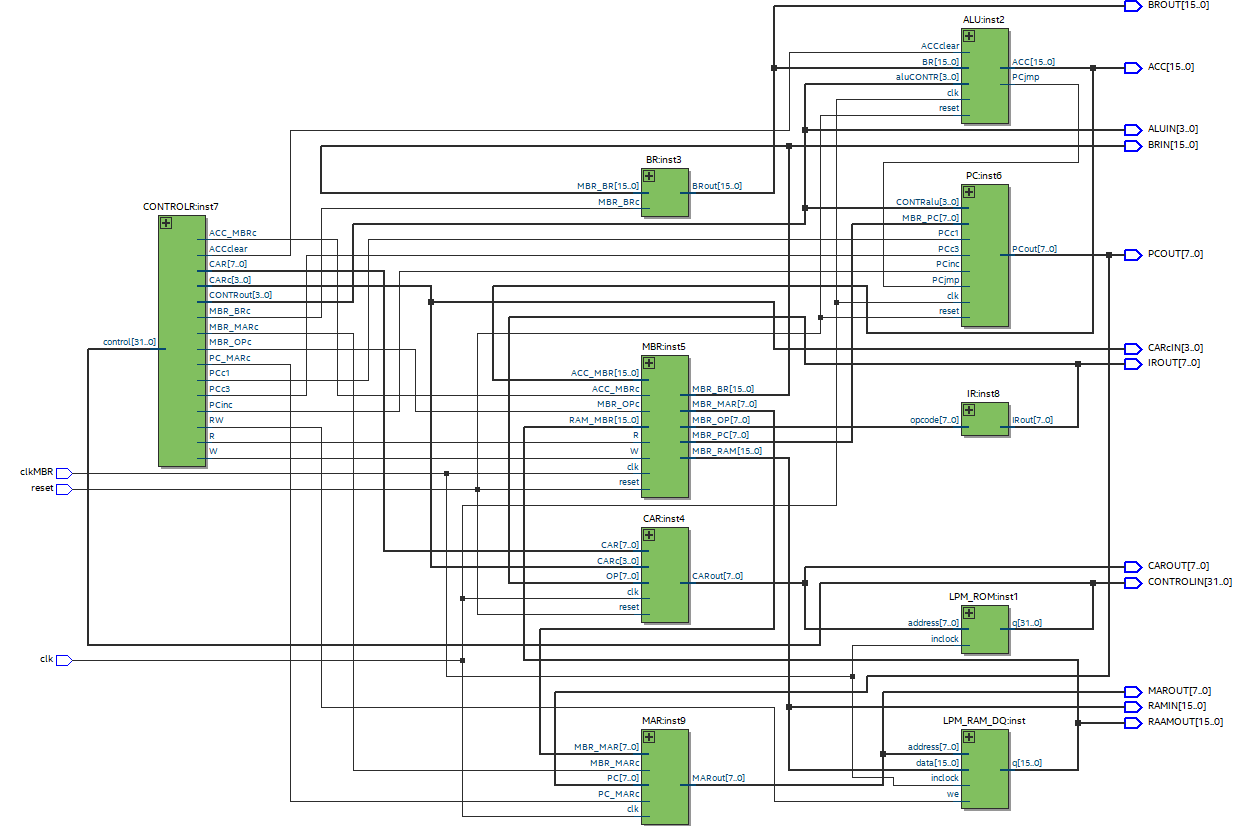
4c : 00000000;

顶层设计原理图：



二．实验结果分析说明

CPU电路图

****

ram中测试程序说明(50+08)

0：022A；Load 2A

1：032B；ADD 2B

2：012C；Store 2C

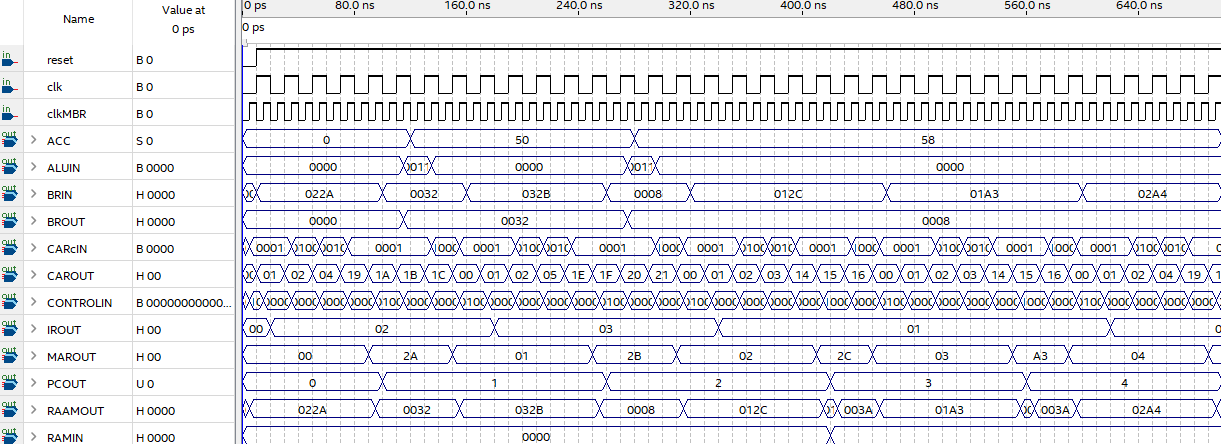
3：0C00；Halt

2a：0032；

2b：0008；

程序1波形图, 根据波形图，逐条指令数据运算过程分析，逐个地址、逐个数据流动变化的对应关系分析。（必做）

波形图



逐条指令数据运算过程分析：

由于reset置0，首先进行初始化操作：PC置0，ACC置0。

当reset置1，CPU开始执行指令：

第一个指令周期中：PC的值00传入RAM内，RAM将00号地址数据022A的操作码02传给ROM，ROM根据操作码向控制器下达LOAD指令；RAM再将地址码2A传给RAM，RAM将2A号地址数据32加载入ALU中，再输出到ACC中。

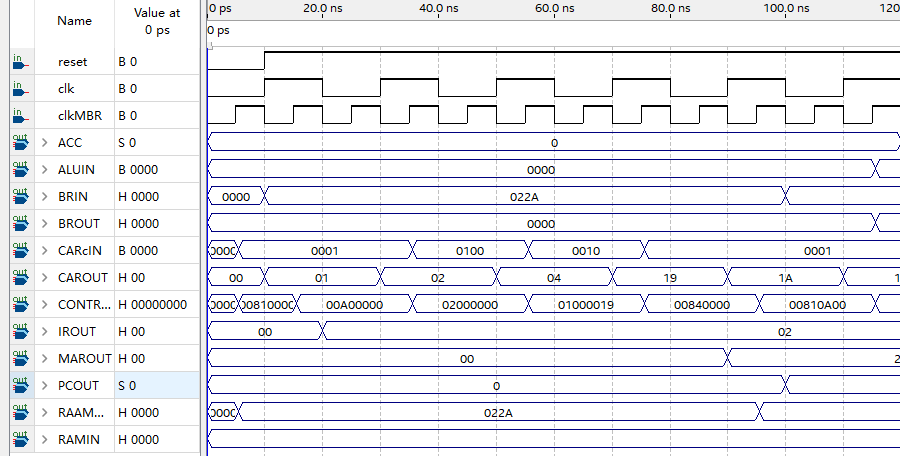
第二个指令周期中:PC的值01传入RAM内，RAM将01号地址数据032B的操作码03传给ROM，ROM根据操作码向控制器下达ADD指令；RAM再将地址码2B传给RAM，RAM将2B号地址数据08输到ALU中，ALU将08与ACC的值50作加法运算，结果58输出到ACC中。

第三个指令周期中:PC的值02传入RAM内，RAM将02号地址数据012C的操作码01传给ROM，ROM根据操作码向控制器下达STORE指令；RAM再根据地址码2C将ACC的值58存储到2C号地址中。

第四个指令周期中：PC的值03传入RAM内，RAM将03号地址数据0C00的操作码0C传给ROM，ROM根据操作码向控制器下达HALT指令，CPU进入停机状态。

任选一条指令作指令周期分析，根据波形图，结合微程序、微指令、微指令格式、代码、电路图、指令系统设计等作出具体说明，标注控制信号，地址，数据等信息的流动、变化。（必做）

波形图



指令周期分析：

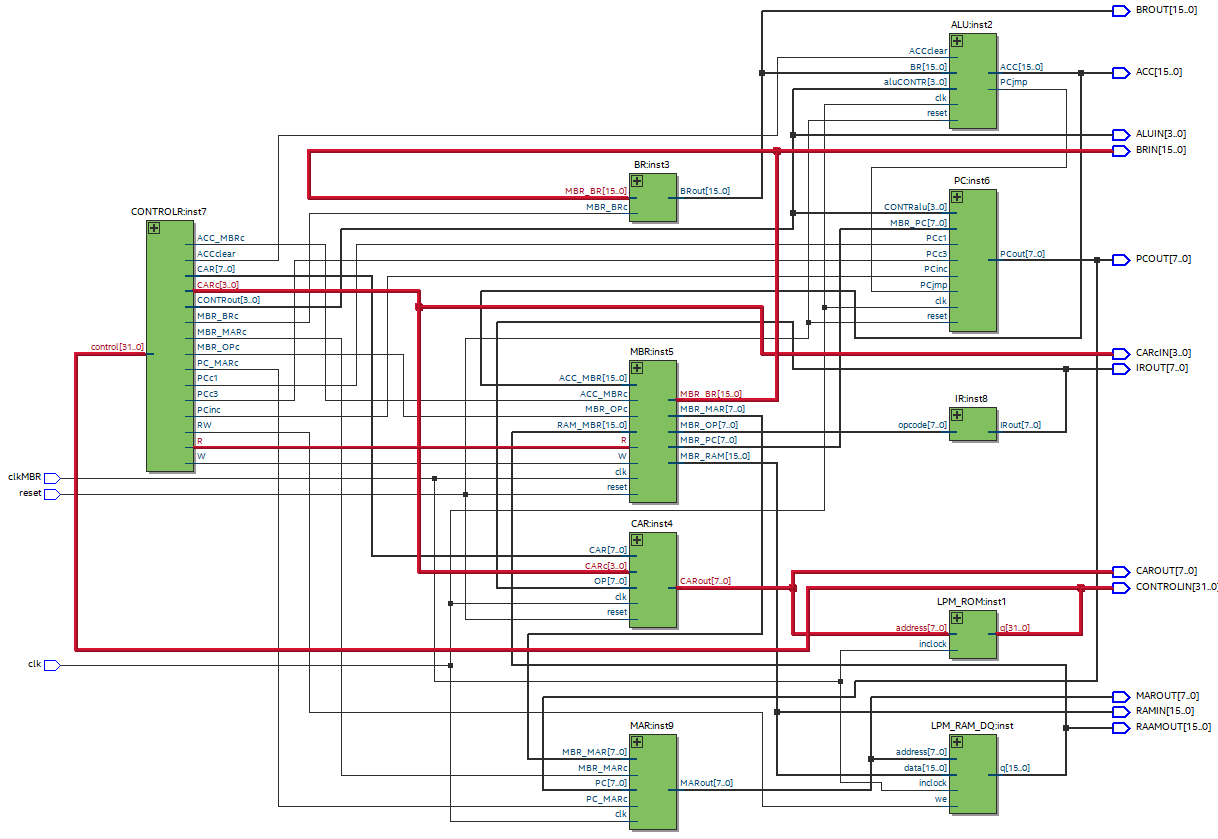
0 ：00810000 R←1, CAR←CAR+1

读主存取出指令字，CAR加1

MBR -> BR ， R有效

代码：

|  |
| --- |
| CARc<=control(26 downto 23);  if CARc="0001" then CARout<=CARout+1;  R<=control(16);  if R='1' then MBR\_BR<=RAM\_MBR; |

电路图： 

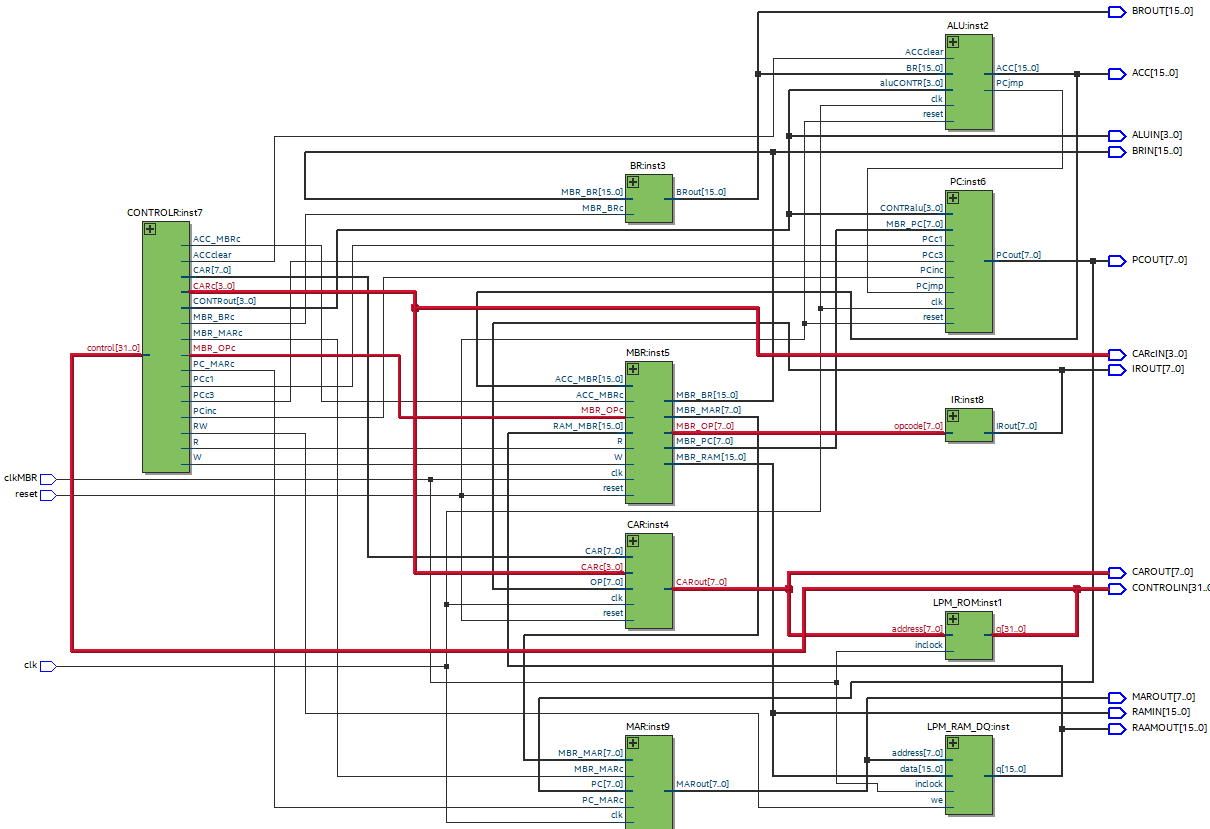
1 ：00A00000 OP←MBR[15..8],CAR←CAR+1

从MBR将指令字高8位送IR，CAR加1

MBR -> OP ， MBR\_OPc有效

代码：

|  |
| --- |
| if MBR\_OPc='1' then MBR\_OP<=RAM\_MBR(15 downto 8);  MBR\_OPc<=control(21);  CARc<=control(26 downto 23);  if CARc="0001" then CARout<=CARout+1; |

电路图：

2 ：02000000 CAR←CAR+OP

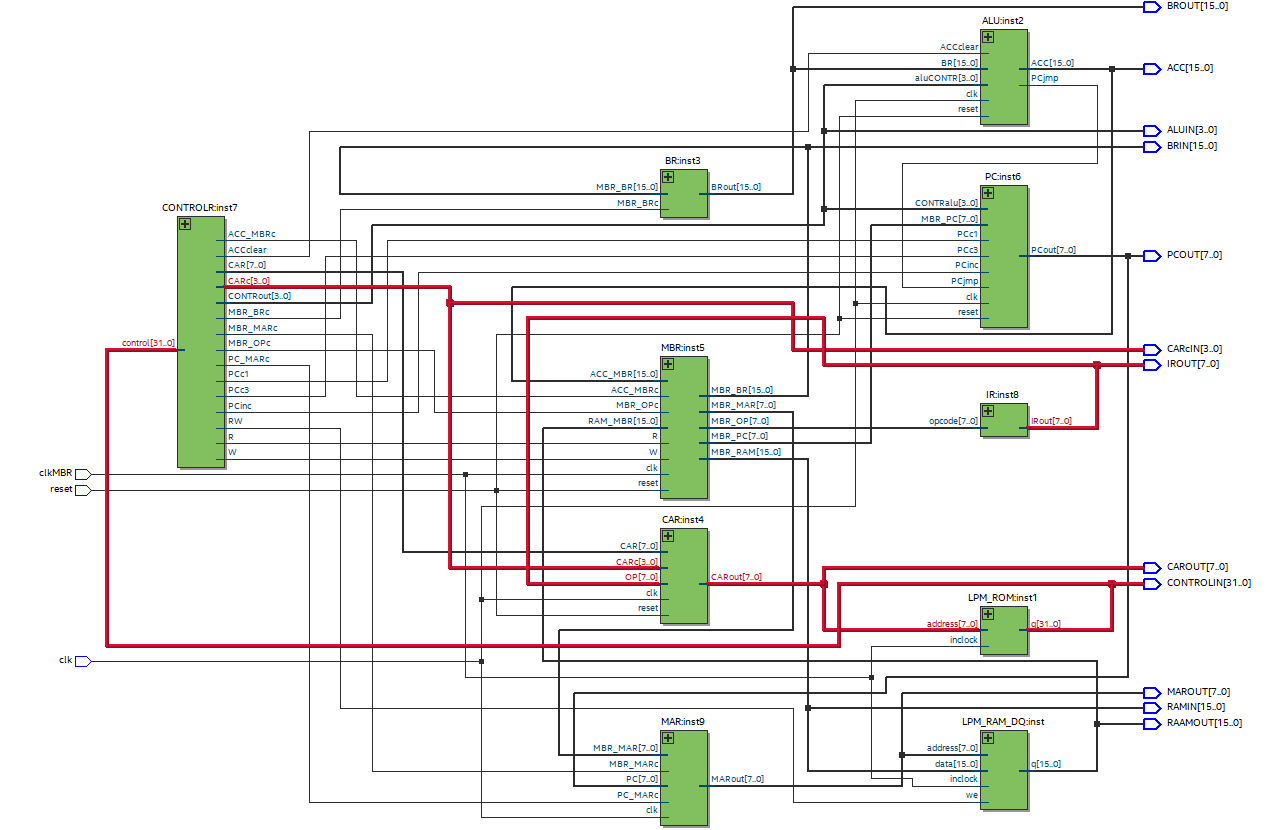
CAR加OP

IR -> CAR

代码：

|  |
| --- |
| CARc<=control(26 downto 23);  if CARc="0100" then CARout<=OP+CARout; |

电路图：



4 ：01000019 CAR←19H

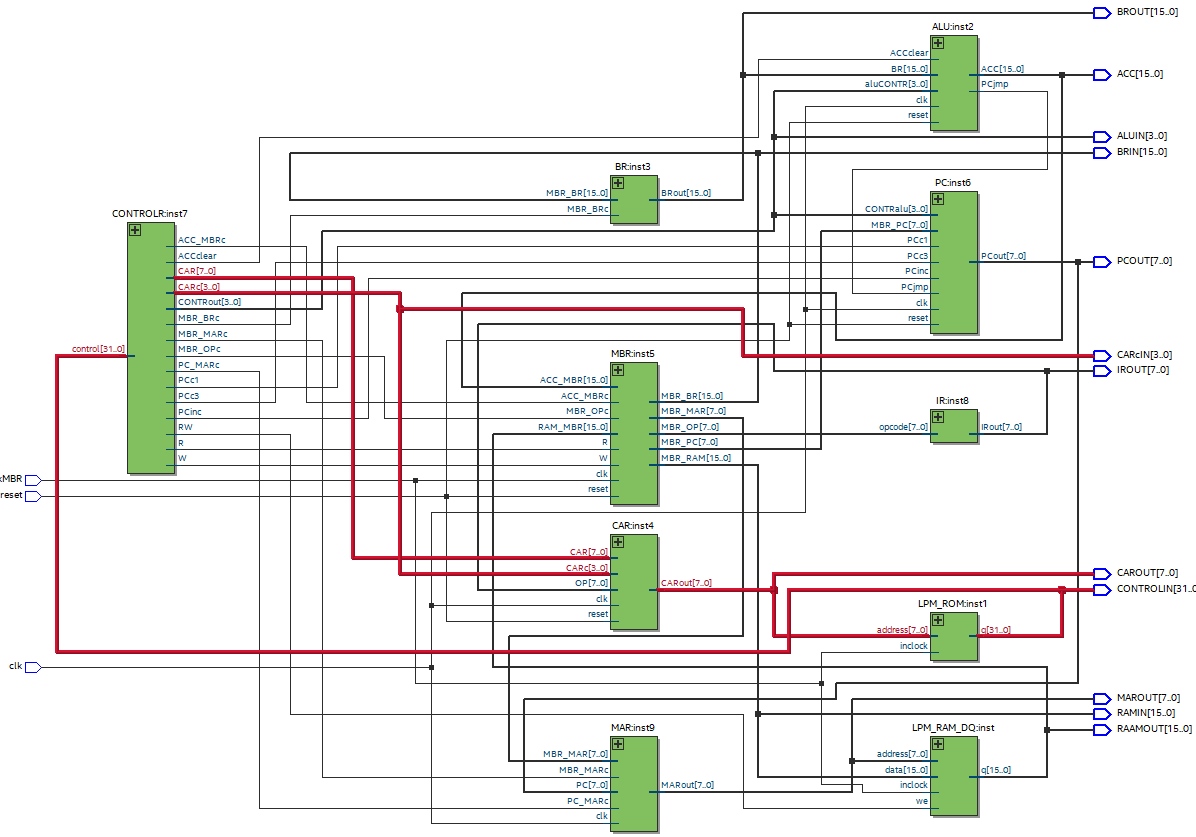
CAR赋值19H

Controlr -> CAR

代码：

|  |
| --- |
| CAR<=control(7 downto 0);  CARc<=control(26 downto 23);  if CARc="0010" then CARout<=CAR; |

电路图：



19 ：00840000 MAR←MBR[7..0], CAR←CAR+1 ------**LOAD**

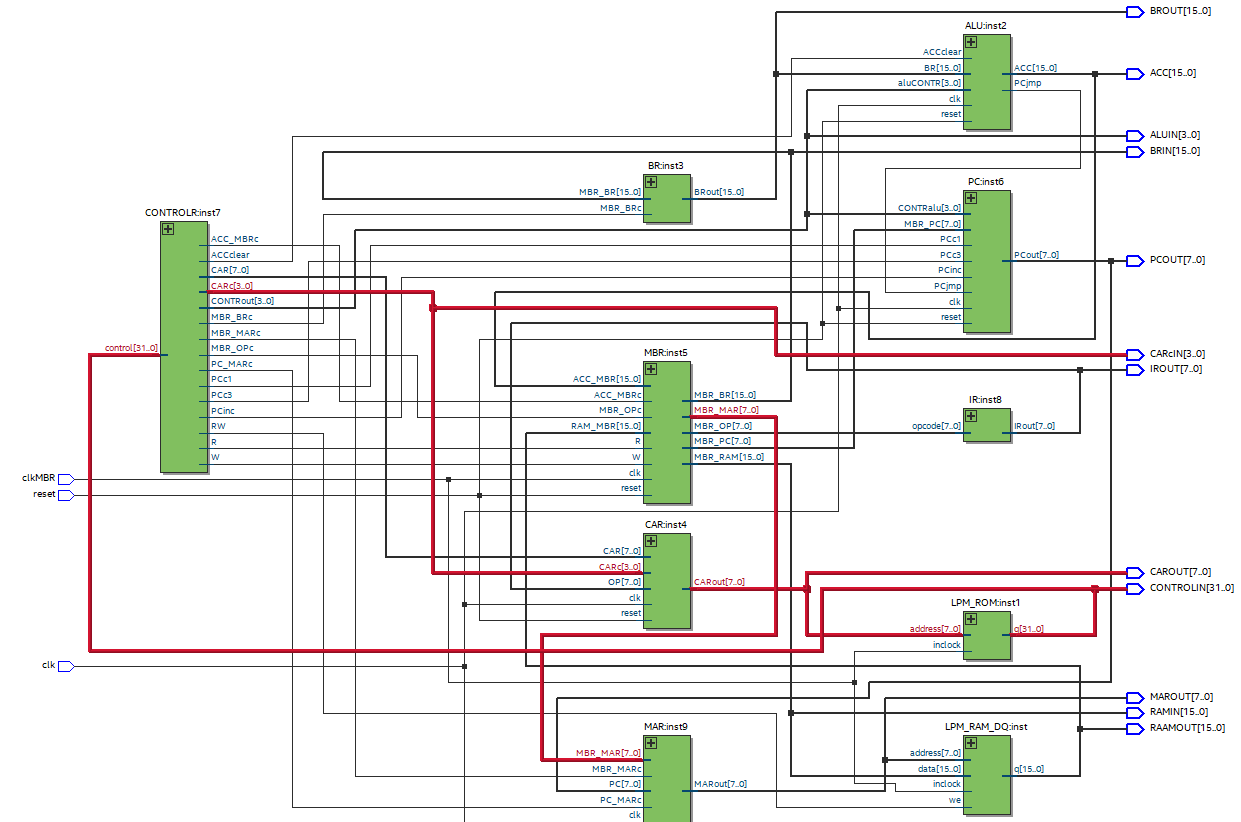
从MBR将指令字低8位送MAR，CAR加1

MBR -> MAR

代码：

|  |
| --- |
| MBR\_MAR<=RAM\_MBR(7 downto 0);  CARc<=control(26 downto 23);  if CARc="0001" then CARout<=CARout+1; |

电路图：



1A ：00810A00 PC←PC+1,R←1,ACC←0,CAR←CAR+1

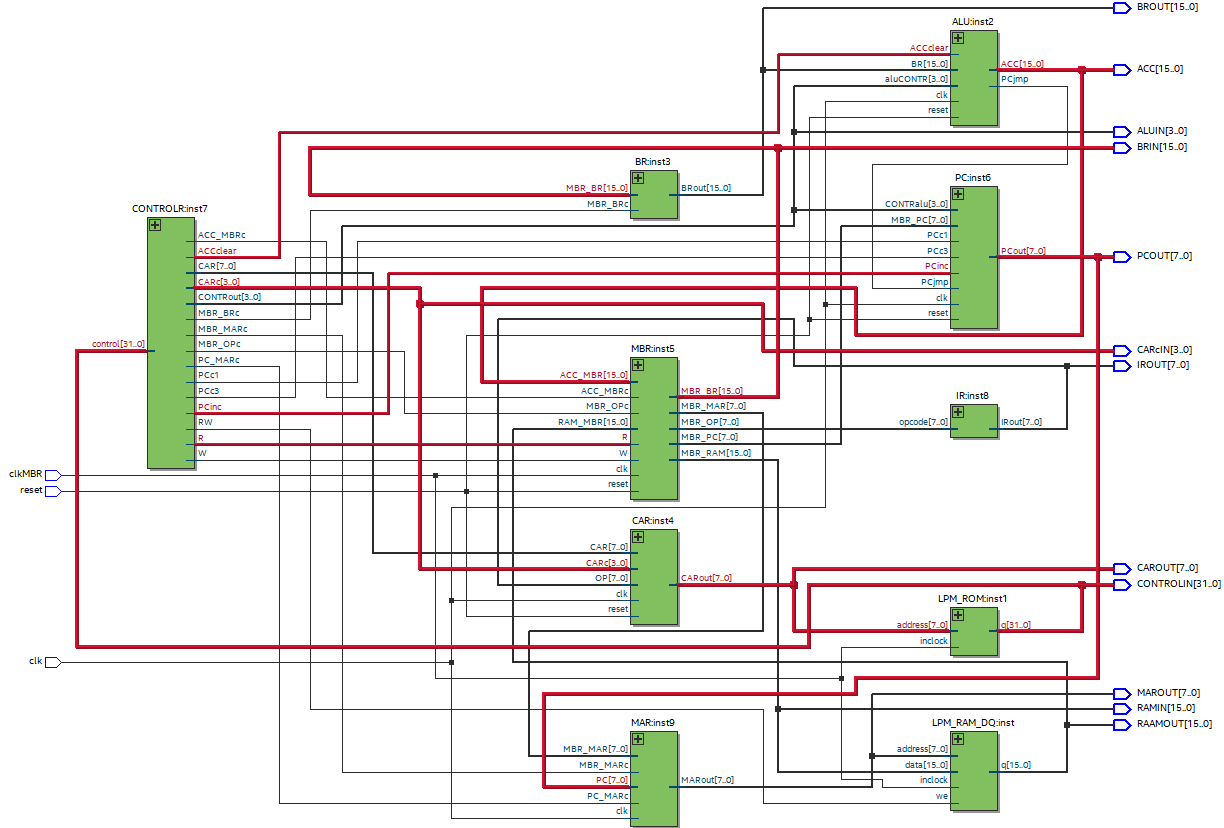
PC加1，读主存取出指令字，将ACC置0，CAR加1

MBR -> BR ， R有效

代码：

|  |
| --- |
| PCinc<=control(9);  if PCinc='1' then PCout<=PCout+1;  R<=control(16);  if R='1' then MBR\_BR<=RAM\_MBR;  ACCclear<=control(11);  if ACCclear='1' then ACC<=x"0000";  CARc<=control(26 downto 23);  if CARc="0001" then CARout<=CARout+1; |

电路图：



1B : 00C03000 BR←MBR,ACC←ACC+BR, CAR←CAR+1

从MBR送入BR，ACC加BR，CAR加1

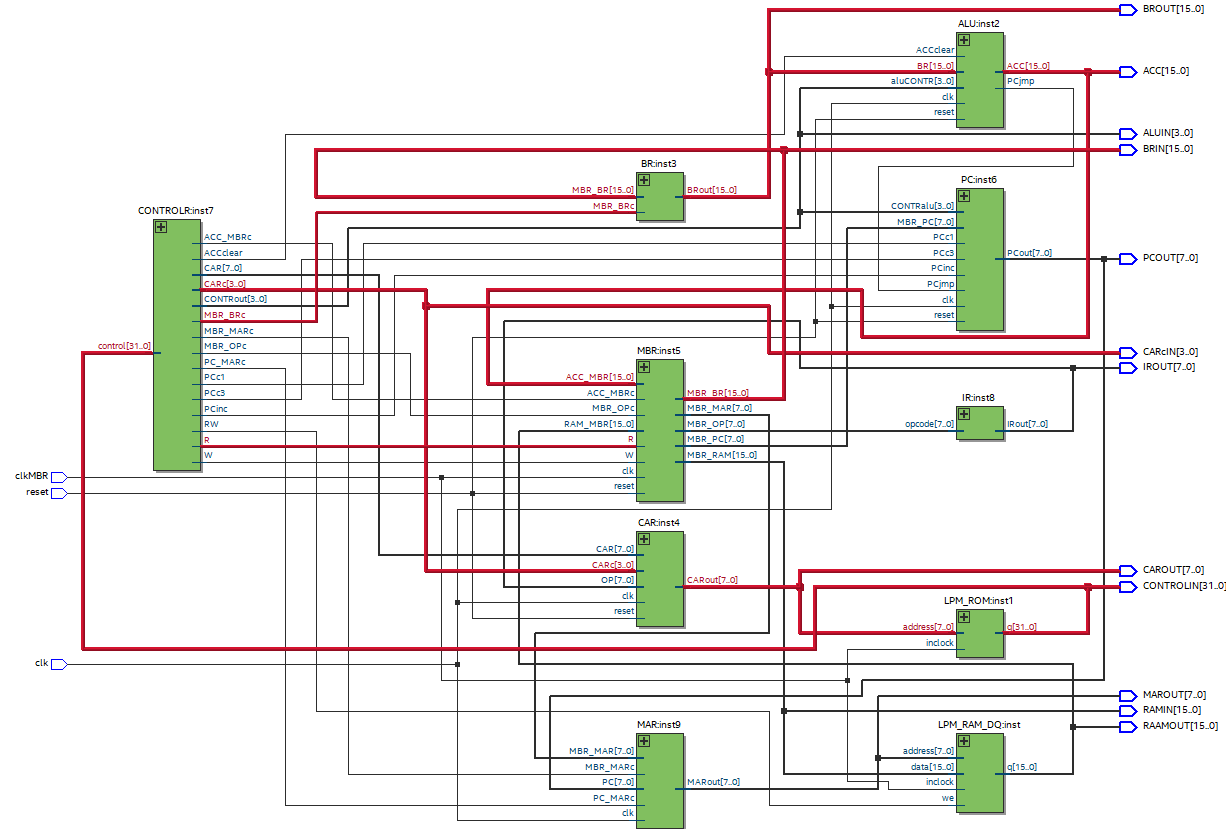
MBR -> BR ， R有效

BR -> ACC ， MBR\_BRc有效

代码：

|  |
| --- |
| R<=control(16);  if R='1' then MBR\_BR<=RAM\_MBR;  MBR\_BRc<=control(22);  if MBR\_BRc='1' then BRout<=MBR\_BR;  if aluCONTR="0011" then ACC<=BR+ACC;  CARc<=control(26 downto 23);  if CARc="0001" then CARout<=CARout+1; |

电路图：



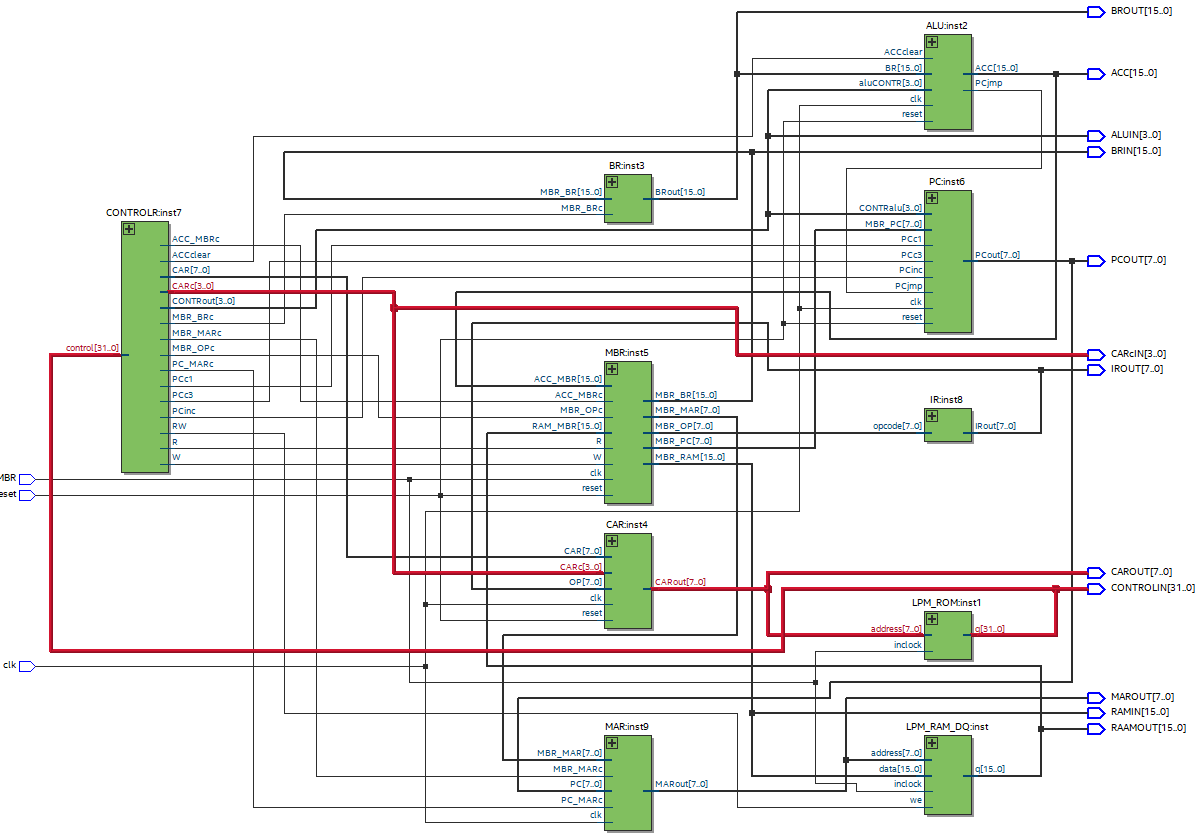
1C : 04080000 CAR←0

CAR置0

代码：

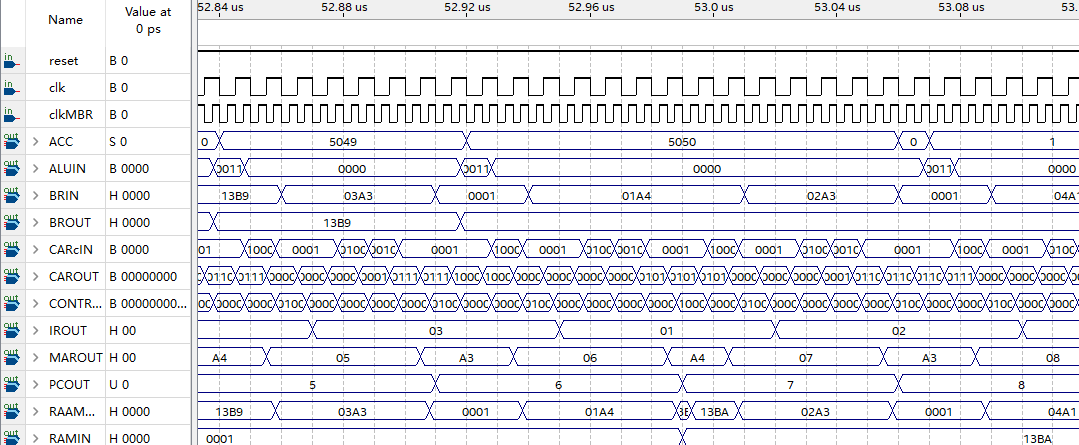
|  |
| --- |
| CARc<=control(26 downto 23);  if CARc="1000" then CARout<="00000000"; |

电路图：



Sum from 1 to 100:

波形图：



逐条指令数据运算过程分析：

00：LOAD A0

PC的值00传入RAM内，RAM将00号地址数据02A0的操作码02传给ROM，ROM根据操作码向控制器下达LOAD指令；RAM再将地址码A0传给RAM，RAM将A0号地址数据00加载入ALU中，再输出到ACC中。

01: STORE A4

PC的值01传入RAM内，RAM将01号地址数据01A4的操作码01传给ROM，ROM根据操作码向控制器下达STORE指令；RAM再将地址码A4传给RAM，RAM将ACC数据存储到A4号地址中。

02: LOAD A2

PC的值02传入RAM内，RAM将02号地址数据02A2的操作码02传给ROM，ROM根据操作码向控制器下达LOAD指令；RAM再将地址码A2传给RAM，RAM将A2号地址数据00加载入ALU中，再输出到ACC中。

03: STORE A3

PC的值03传入RAM内，RAM将03号地址数据01A3的操作码01传给ROM，ROM根据操作码向控制器下达STORE指令；RAM再将地址码A3传给RAM，RAM将ACC数据存储到A3号地址中。

04: LOOP:LOAD A4

PC的值04传入RAM内，RAM将04号地址数据02A4的操作码02传给ROM，ROM根据操作码向控制器下达LOAD指令；RAM再将地址码A4传给RAM，RAM将A4号地址数据加载入ALU中，再输出到ACC中。

05: ADD A3

PC的值05传入RAM内，RAM将05号地址数据03A3的操作码03传给ROM，ROM根据操作码向控制器下达ADD指令；RAM再将地址码A3传给RAM，RAM将A3号地址数据输到ALU中，与ACC的值作加法运算，结果输出到ACC中。

06: STORE A4

PC的值06传入RAM内，RAM将06号地址数据01A4的操作码01传给ROM，ROM根据操作码向控制器下达STORE指令；RAM再将地址码A4传给RAM，RAM将ACC数据存储到A4号地址中。

07: LOAD A3

PC的值07传入RAM内，RAM将07号地址数据02A3的操作码02传给ROM，ROM根据操作码向控制器下达LOAD指令；RAM再将地址码A3传给RAM，RAM将A3号地址数据加载入ALU中，再输出到ACC中。

08: SUB A1

PC的值08传入RAM内，RAM将08号地址数据04A1的操作码04传给ROM，ROM根据操作码向控制器下达SUB指令；RAM再将地址码A1传给RAM，RAM将A1号地址数据输到ALU中，与ACC的值作减法运算，结果输出到ACC中。

09: STORE A3

PC的值09传入RAM内，RAM将09号地址数据01A3的操作码01传给ROM，ROM根据操作码向控制器下达STORE指令；RAM再将地址码A3传给RAM，RAM将ACC数据存储到A3号地址中。

0A: JMPGZ LOOP

PC的值0A传入RAM内，RAM将0A号地址数据01A3的操作码01传给ROM，ROM根据操作码向控制器下达STORE指令；RAM再将地址码A3传给RAM，RAM将ACC数据存储到A3号地址中。

0B: HALT

PC的值0B传入RAM内，RAM将0B号地址数据0C00的操作码0C传给ROM，ROM根据操作码向控制器下达HALT指令，CPU进入停机状态。