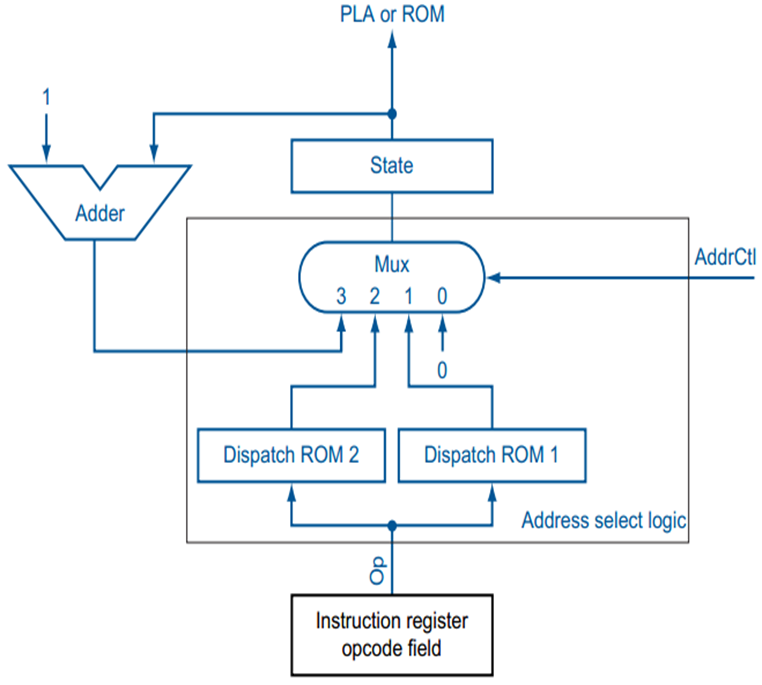
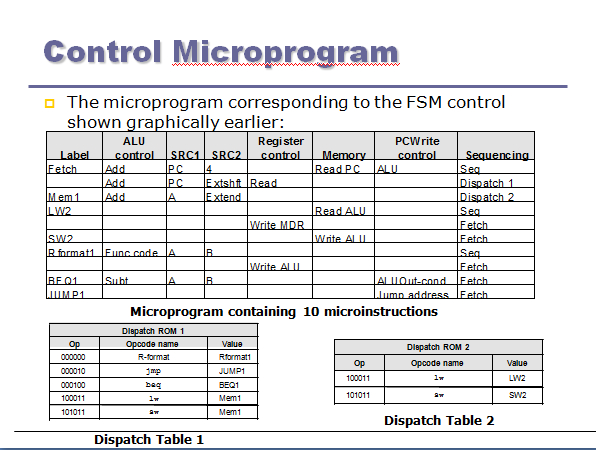
1. **主要部件的说明及测试**
2. 主控制器

说明：控制器中，我采用微程序来实现有限状态机的功能。

控制器的参考模型为：



控制器的微程序流程的参考模型为：



根据上述方法，我对控制信号进行了扩展，从而满足要求的指令需求。

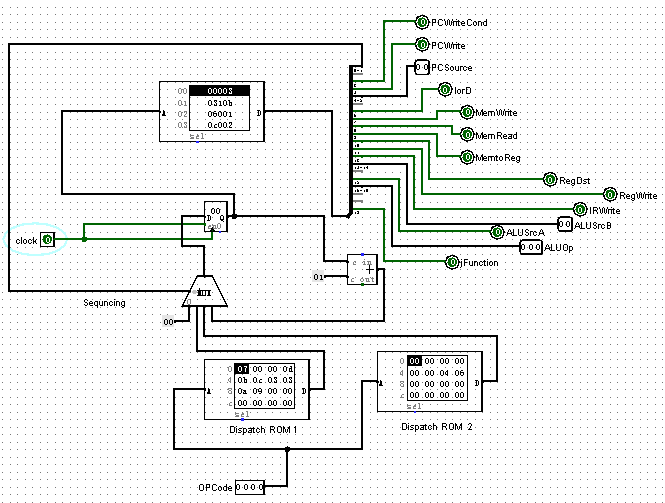
微指令格式如下：（微指令为20位）

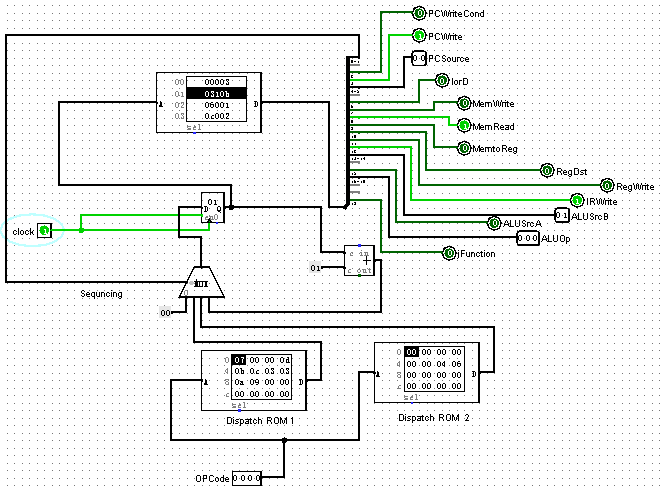
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 与指令B  EQ和BNQ的控制信号 | ALU control | | | SRC1 | SRC2 | | Register  control | | | Memory | | | | PCWrite  control | | | | Sequencing  control | |
| JFunction | ALUOp2 | ALUOp1 | ALUOp0 | ALUSrcA | ALUSrcB1 | ALUSrcB0 | IRWrite | RegWrite | RegDst | MemtoReg | MemRead | MemWrite | IorD | PCSource1 | PCSource0 | PCWrite | PCWriteCond | Addrctl1 | Addrctl0 |
| 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

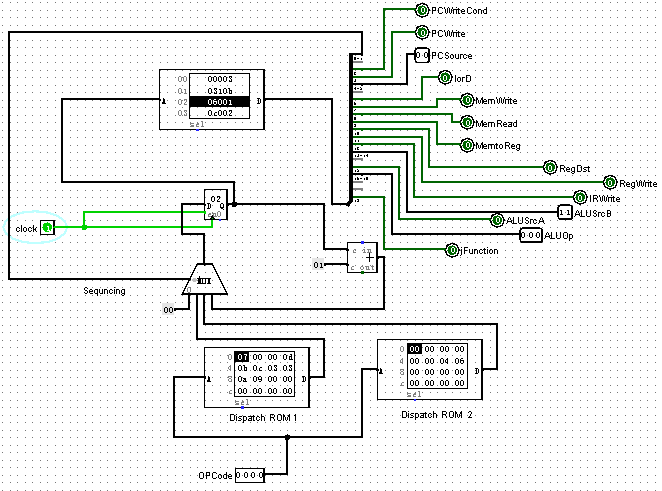
我将微指令存于ROM中，如下表所示（数据和地址都为16进制）：

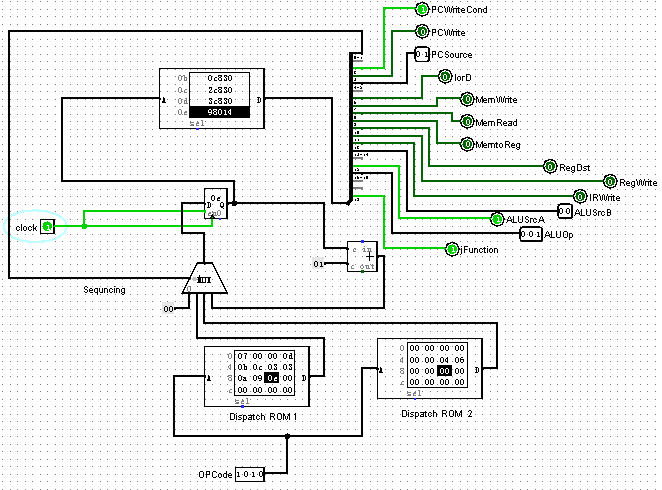
|  |  |  |  |
| --- | --- | --- | --- |
| 标签 | 地址 | 微指令 | 功能 |
| Fetch0 | 00 | 00003 | NOP，微指令地址加1 |
| Fetch | 01 | 0310b | 取指令，PC加1送入PC，微指令地址加1 |
|  | 02 | 06001 | PC加立即数等待送入PC，进入对应指令的微程序的入口地址 |
| Mem | 03 | 0c002 | A加立即数送入ALUOut, 进入SW或LW指令的微程序的入口地址 |
| LW | 04 | 00343 | ALUOut的值作为内存地址，微指令地址加1 |
|  | 05 | 00a00 | 从内存中将数据读到目的寄存器，返回Fetch0 |
| SW | 06 | 000c0 | 将B中的数据写入目前地址的内存空间，返回Fetch0 |
| R-format | 07 | 48403 | 由指令的后两位决定对A和B采取何种运算，再送入ALUOut，微指令地址加1 |
|  | 08 | 00c00 | 将ALUOut的值送目的寄存器，返回Fetch0 |
| BEQ | 09 | 18014 | A和B相减相等则将要跳转的地址送PC，返回Fetch0 |
| JUMP | 0A | 00028 | 将要跳转的地址送PC，返回Fetch，返回Fetch0 |
| ORI | 0B | 3c033 | A与立即数或运算，微指令地址加1 |
|  | 0C | 00c00 | 送目的寄存器，返回Fetch0 |
| ADDI | 0D | 0c033 | A+立即数，微指令地址加1 |
|  | 0E | 00c00 | 送目的寄存器，返回Fetch0 |
| ANDI | OF | 2c033 | A与立即数和运算，微指令地址加1 |
|  | 10 | 00c00 | 送目的寄存器，返回Fetch0 |
| BNE | 11 | 98014 | A和B相减不相等则将要跳转的地址送PC，返回Fetch0 |

测试：现在的操作码为0AH（10102），其微指令入口为0EH，微指令码为98014H









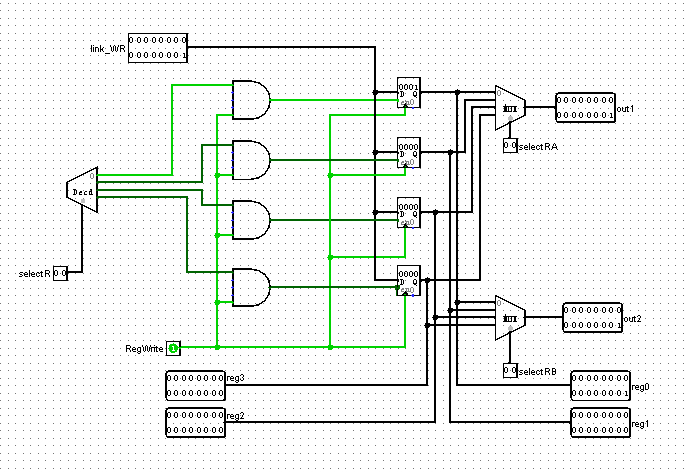
经多次验证主控制器的逻辑和电路正确。

1. 寄存器堆

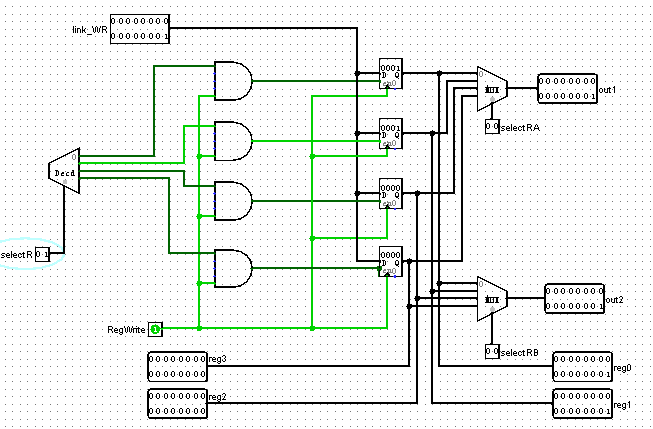
说明信号selectR决定将数据存入哪个寄存器（目的寄存器），selectRA决定哪个寄存器的数据送入A，selectRB决定哪个寄存器的数据送入B。

这里以0001H为例存入。

1、存到Reg0



存到Reg1



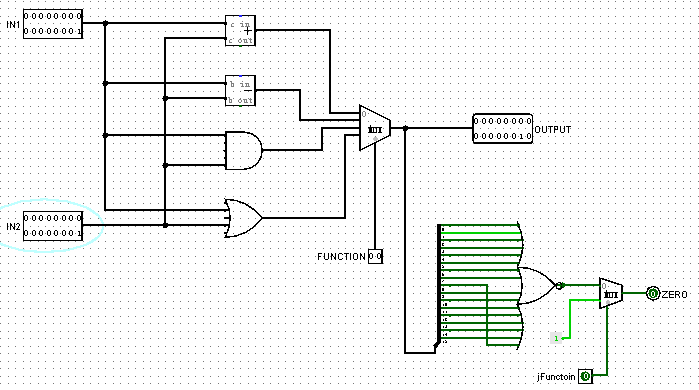
测试结果已经可以证明，寄存器堆功能和电路的正确性。

1. ALU

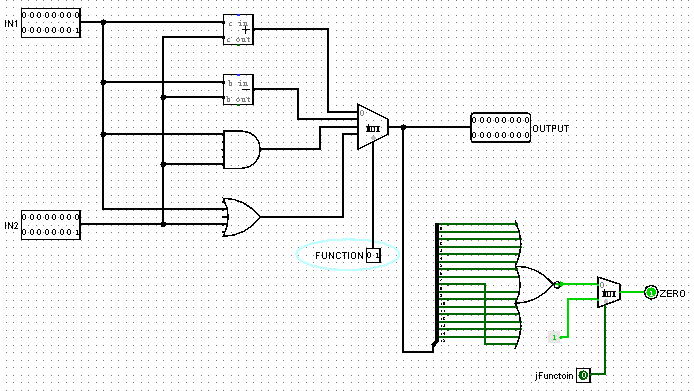
测试说明：信号FUNCTION决定输出何种运算的结果。

这里我以0001H和0001H作为两个输入进行测试：

加运算：



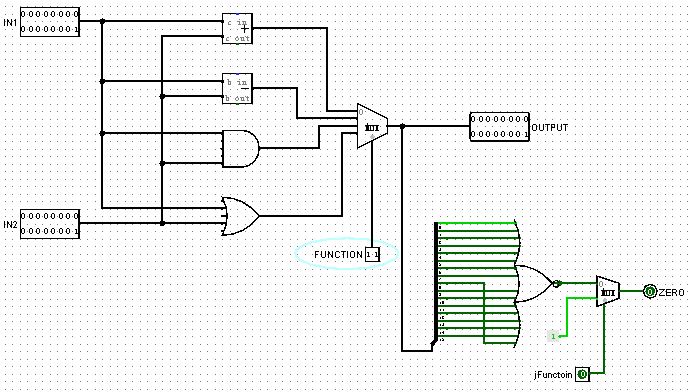
减运算：



与运算：



或运算：



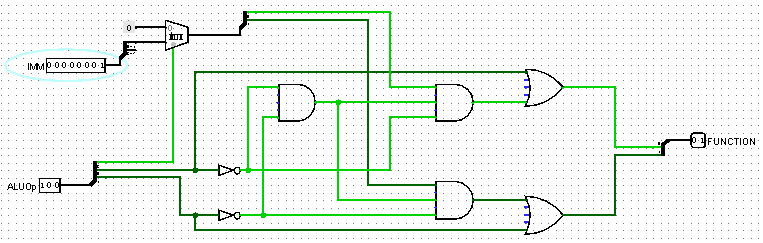
由上可知：运算器功能正确，电路逻辑没有问题。

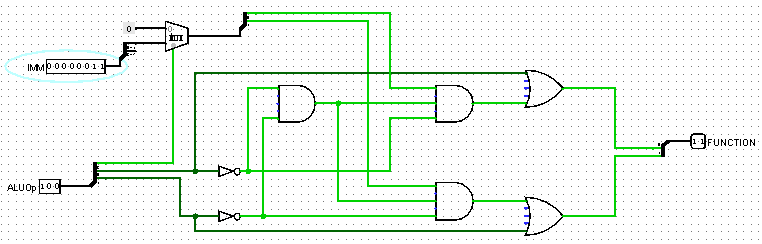
1. ALU 控制器

测试说明：当信号ALUOp为100时，输出与指令的最后两位一致；当信号ALUOp为0\*\*时，则输出与ALUOp信号的后两位一致。

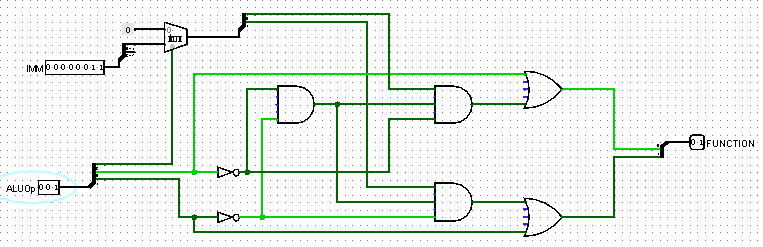
测试结果：

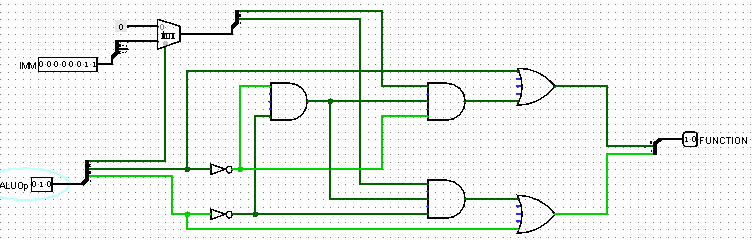
1、ALUOp为100时





2、0\*\*时：





从测试结果看来：运算控制器满足功能需求，没有出现错误。所以运算控制器设计成功。

**二、多周期处理机的功能实现**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **15-12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | | **3** | **2** | **1** | **0** |
| 0 | rs | | rt | | rd | | party bits! | | | | funct | | | 见R-type Instructions |
| \* | \* | | \* | | \* | | | | | | | | | \* |
| \* | \* | | \* | | \* | | | | | | | | | \* |
| 3 | rs | | rt | | immediate | | | | | | | | | ori: $rt = $rs | imm |
| 4 | rs | | rt | | immediate | | | | | | | | | addi: $rt = $rs + imm |
| 5 | rs | | rt | | immediate | | | | | | | | | andi: $rt = $rs & imm |
| 6 | rs | | rt | | immediate | | | | | | | | | lw: $rt = MEM[$rs + imm] |
| 7 | rs | | rt | | immediate | | | | | | | | | sw: MEM[$rs+imm] = $rt |
| 8 | jump address | | | | | | | | | | | | | jump |
| 9 | rs | | rt | | offset | | | | | | | | | beq |
| 10 | rs | | rt | | offset | | | | | | | | | bne |
| **funct** | **meaning** | | | | | | | |
| 0 | add: $rd = $rs + $rt | | | | | | | |
| 1 | sub: $rd = $rs - $rt | | | | | | | |
| 2 | or: $rd = $rs | $rt | | | | | | | |
| 3 | and: $rd = $rs & $rt | | | | | | | |

实现了如上表所示功能及指令格式。

测试代码：(以下数值均为16进制)

地址 机器码 汇编指令 结果

00 6020 lw r0,r0,20 r0=2

01 6521 lw r1,r1,21 r1=1

02 0180 add r0,r1,r2 r0=2 ,r1=1 ,r2=3

03 0181 sub r0,r1,r2 r0=2 ,r1=1 ,r2=1

04 0182 and r0,r1,r2 r0=2 ,r1=1 ,r2=0

05 0183 or r0,r1,r2 r0=2 ,r1=1 ,r2=3

06 7220 sw r0, r2,20 r0=2 ,r1=1 ,r2=3

07 9110 beq r0, r1,20 r0=2 ,r1=1 ,r2=3

08 a107 bne r0, r1,20 r0=2 ,r1=1 ,r2=3

\*

\*

\*

10 3301 ori r0,r3,1 r0=2,r1=1,r2=3,r3=3

11 4302 addi r0,r3,2 r0=2,r1=1,r2=3,r3=4

12 5301 andi r0,r3,1 r0=2,r1=1,r2=3,r3=0

13 8000 jump 0

**三、总结**

通过本次project，使自己更加充分地理解计算机的组成原理，而且对计算机底层逻辑有了较好的理解。

在本次project中，还存在一些没弄明白的地方，比如display指令，我不知道怎样去实现。

但不可否认，通过本次试验，自己学会很多计算硬件及工作原理等相关知识。

作为初学者，花了大量时间去琢磨，这是个一个很艰巨但同样很快乐的过程。