Lab 3 :: CPU alARM

Konsing Ham Lopez, 1037, ECS 154A 001

Table of Contents

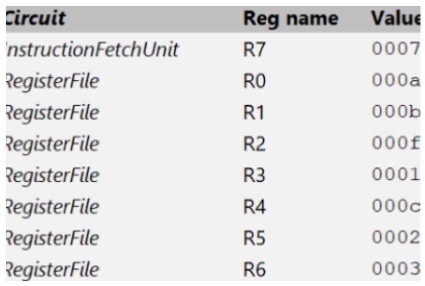
Introduction

Lab 3 is essentially a more complicated version of the FISC (five or four instruction set computer) that we completed in question 6 of lab 2. This particular circuit consists of a much more complex logic due to it being 16 bit.

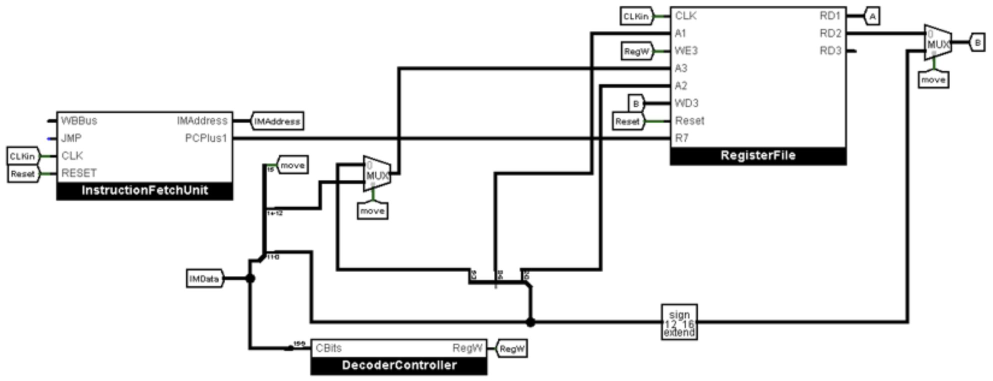
Move

The first instruction I worked on is move. I pass 12 either the immediate value (which is 12 bits) or register Rm. Since move instructions contain a 1 on the most significant bit, this makes it easier to identify. This means that the Rd is bits 14-12 while 11-0 are the immediate value. This immediate value goes from B to the ALU and writes it to the Rd register.

```
                    v2.0 raw
MOV r0, 0xA         800A
MOV r1, 0x00B       900B
MOV r2, 0x00F       A00F
MOV r3, 0x001       B001
MOV r4, 0xC         C00C
MOV r5, 0x2         D002
MOV r6, 0x003       E003
```

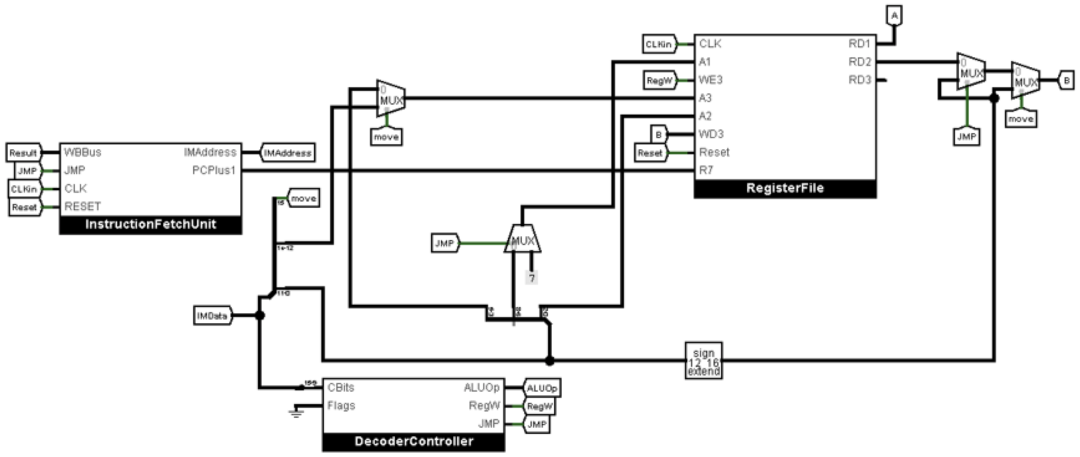


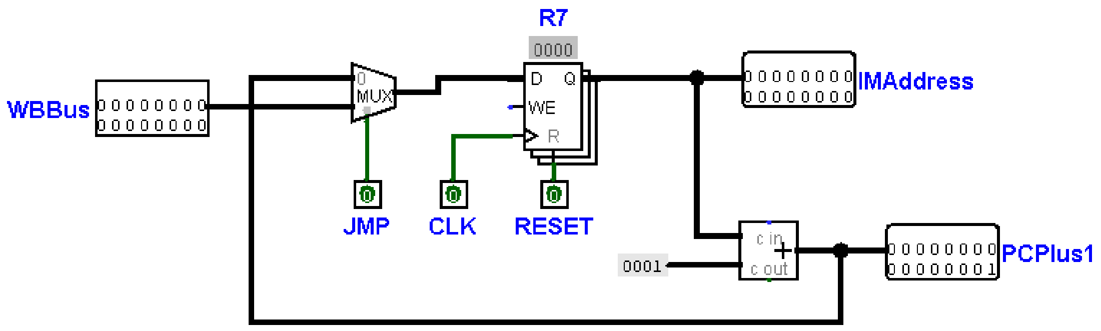
Circuit	Reg name	Value
nstructionFetchUnit	R7	0007
RegisterFile	R0	000a
RegisterFile	R1	000b
RegisterFile	R2	000f
RegisterFile	R3	0001
RegisterFile	R4	000c
RegisterFile	R5	0002
RegisterFile	R6	0003

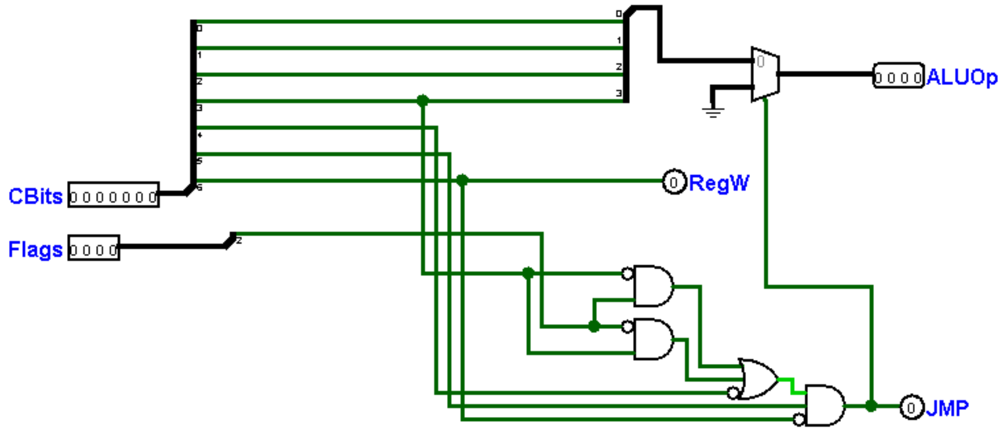


Branch Operations

The next operation is branch. This operation reads the 12 bit immediate value and adds it to the pc plus 1 or R7. In other words, if the instruction is a mov or B, then B is sign extended immediate value. Otherwise, the value Rm is passed through. Furthermore, the constant seven is passed to A1 if we have a branch operation to pass the updated pc + 1 to A. Additionally, B's values are added through the arithmetic logic unit. This is why I set the ALU operations to 0 (adding op code). The result of this operation is routed from the arithmetic logic unit into the WBBus and ultimately to either R7 or the PC register.

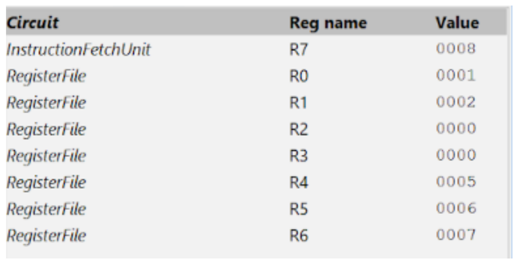






- Branch operation (B): Branch (to pc+1 + the immediate values) no matter what our flags are

```
MOV r0, 0x001          v2.0 raw
MOV r1, 0x002          8001
B 0x002                9002
MOV r2, 0x003          4002
MOV r3, 0x004          A003
MOV r4, 0x005          B004
MOV r5, 0x006          C005
MOV r6, 0x007          D006
                       E007
```

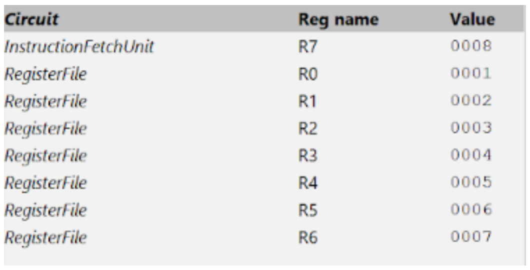


Circuit	Reg name	Value
InstructionFetchUnit	R7	0008
RegisterFile	R0	0001
RegisterFile	R1	0002
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0005
RegisterFile	R5	0006
RegisterFile	R6	0007

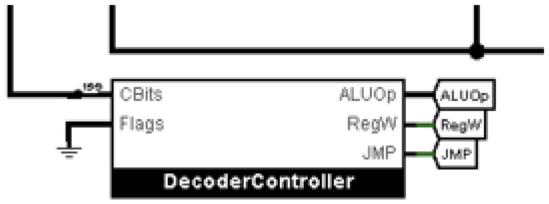
- BEQ branches to pc + 1 + immediate value depending on z flag

(BEQ) when Z is 0, pass 0 due to the bit sequence 0000

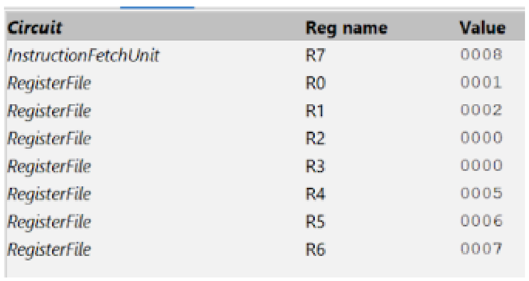
MOV r0, 0x001	v2.0 raw
MOV r1, 0x002	8001
BEQ 0x002	9002
MOV r2, 0x003	6002
MOV r3, 0x004	A003
MOV r4, 0x005	B004
MOV r5, 0x006	C005
MOV r6, 0x007	D006
	E007



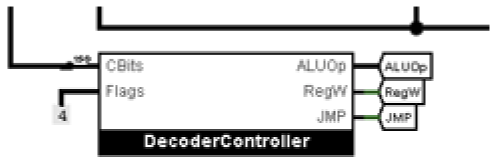
Circuit	Reg name	Value
InstructionFetchUnit	R7	0008
RegisterFile	R0	0001
RegisterFile	R1	0002
RegisterFile	R2	0003
RegisterFile	R3	0004
RegisterFile	R4	0005
RegisterFile	R5	0006
RegisterFile	R6	0007



(BEQ) when Z is 1, pass 4 due to the bit sequence 0100



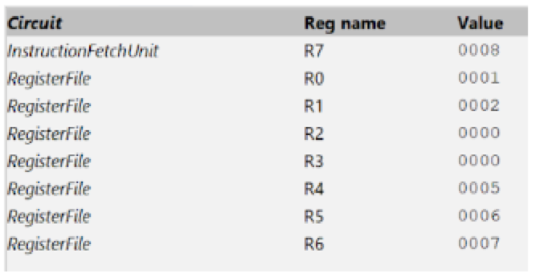
Circuit	Reg name	Value
InstructionFetchUnit	R7	0008
RegisterFile	R0	0001
RegisterFile	R1	0002
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0005
RegisterFile	R5	0006
RegisterFile	R6	0007



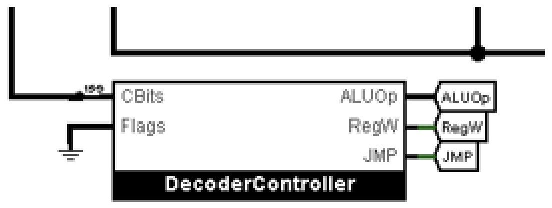
- BNE

(BNE) when Z is 0, pass 0 due to the bit sequence 0000

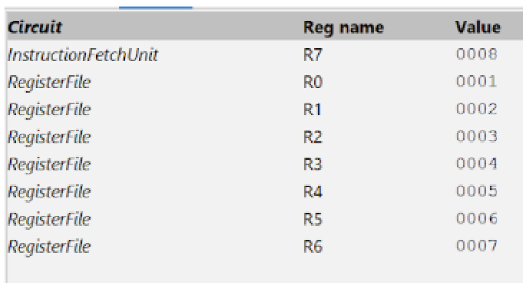
```
MOV r0, 0x001          v2.0 raw
MOV r1, 0x002          8001
BNE 0x002              9002
MOV r2, 0x003          7002
MOV r3, 0x004          A003
MOV r4, 0x005          B004
MOV r5, 0x006          C005
MOV r6, 0x007          D006
                       E007
```



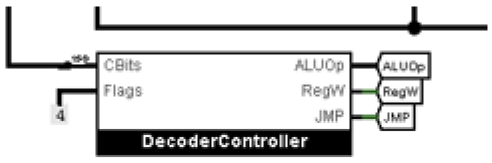
Circuit	Reg name	Value
InstructionFetchUnit	R7	0008
RegisterFile	R0	0001
RegisterFile	R1	0002
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0005
RegisterFile	R5	0006
RegisterFile	R6	0007



(BNE) when Z is 1, pass 4 due to the bit sequence 0100



Circuit	Reg name	Value
InstructionFetchUnit	R7	0008
RegisterFile	R0	0001
RegisterFile	R1	0002
RegisterFile	R2	0003
RegisterFile	R3	0004
RegisterFile	R4	0005
RegisterFile	R5	0006
RegisterFile	R6	0007



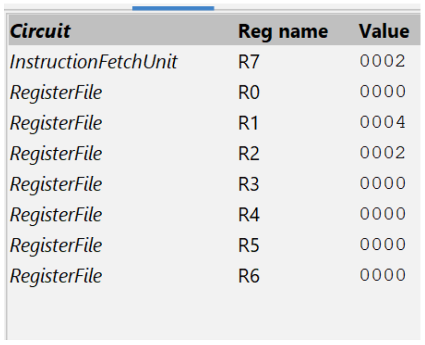
ALU Operations

The ALU has a 001 OP code at the beginning of every ALU operation. This means that every time we use the arithmetic logic unit, we know that the instruction that was passed in started with 001 in the most significant bits. ALU instructions take in the A and B inputs and perform the instructions indicated by the 4 least significant bits. This stuff is then passed into the 0 input of the mux. The mux then passes either the register or imm val B or the arithmetic logic unit's result.

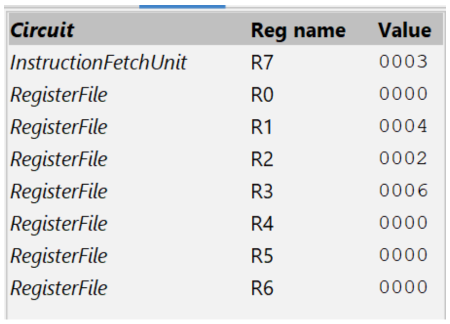
Testing MOV, ADD, SUB, MUL, MULU, DIV, MOD, AND, OR, EOR, NOT, LSL, LSR, ASR, ROL, ROR, CMP ALU instructions:

```
  MOV r1, 0x004
  MOV r2, 0x002
  ADD r3, r1, r2
  SUB r4, r2, r1
  MUL r5, r1, r2
  MULU r6, r1, r2
  DIV r1, r2, r1
  MOD r3, r2, r1
  AND r4, r1, r2
  OR r5, r1, r2
  EOR r6, r1, r2
  NOT r1, r3
  LSL r3, r1, r2
  LSR r4, r1, r2
  ASR r5, r1, r2
  ROL r6, r1, r2
  ROR r1, r1, r2
  CMP r1, r2
```

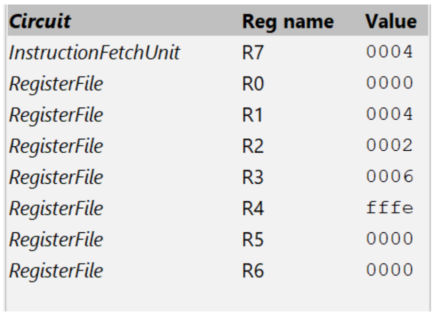
```
v2.0 raw
9004
A002
205A
22A1
246A
2672
2889
2A99
2C62
2E6A
3072
32C8
345A
3662
386A
3A72
3C4A
3E42
```



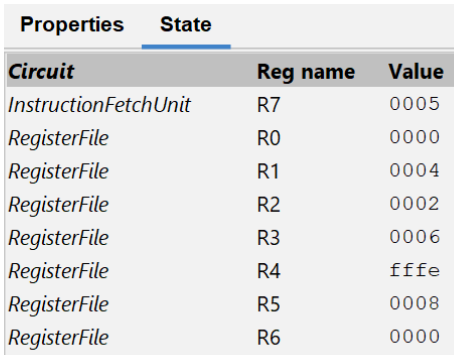
Circuit	Reg name	Value
InstructionFetchUnit	R7	0002
RegisterFile	R0	0000
RegisterFile	R1	0004
RegisterFile	R2	0002
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0000
RegisterFile	R6	0000



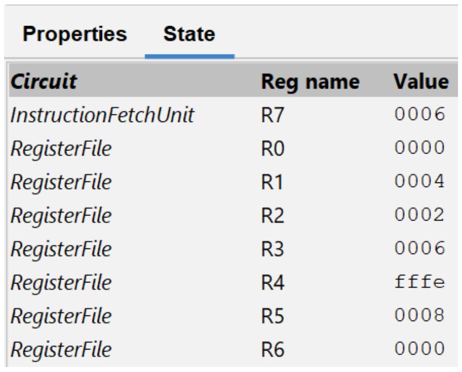
Circuit	Reg name	Value
InstructionFetchUnit	R7	0003
RegisterFile	R0	0000
RegisterFile	R1	0004
RegisterFile	R2	0002
RegisterFile	R3	0006
RegisterFile	R4	0000
RegisterFile	R5	0000
RegisterFile	R6	0000



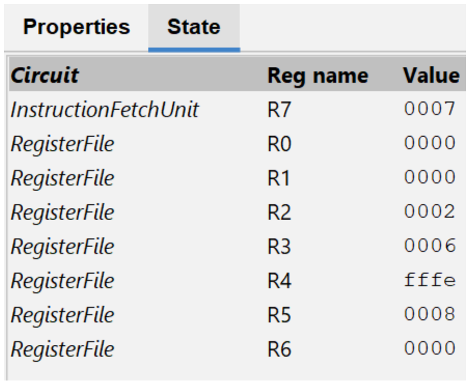
Circuit	Reg name	Value
InstructionFetchUnit	R7	0004
RegisterFile	R0	0000
RegisterFile	R1	0004
RegisterFile	R2	0002
RegisterFile	R3	0006
RegisterFile	R4	fffe
RegisterFile	R5	0000
RegisterFile	R6	0000



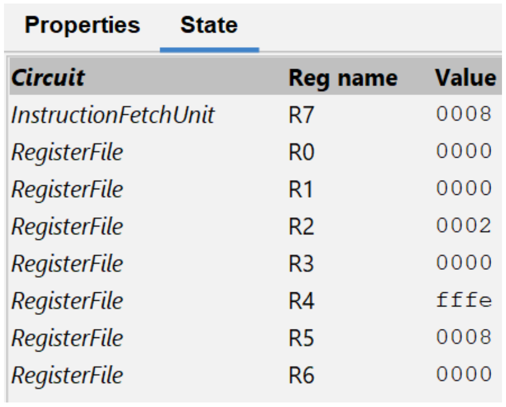
Circuit	Reg name	Value
InstructionFetchUnit	R7	0005
RegisterFile	R0	0000
RegisterFile	R1	0004
RegisterFile	R2	0002
RegisterFile	R3	0006
RegisterFile	R4	fffe
RegisterFile	R5	0008
RegisterFile	R6	0000


Properties | State

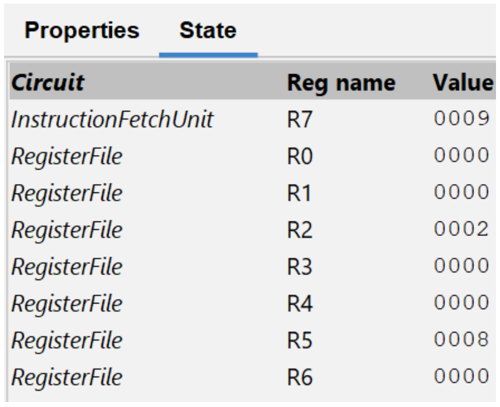
Circuit	Reg name	Value
InstructionFetchUnit	R7	0006
RegisterFile	R0	0000
RegisterFile	R1	0004
RegisterFile	R2	0002
RegisterFile	R3	0006
RegisterFile	R4	fffe
RegisterFile	R5	0008
RegisterFile	R6	0000


Properties | State

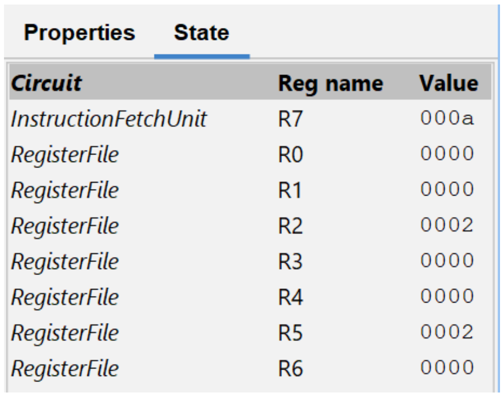
Circuit	Reg name	Value
InstructionFetchUnit	R7	0007
RegisterFile	R0	0000
RegisterFile	R1	0000
RegisterFile	R2	0002
RegisterFile	R3	0006
RegisterFile	R4	fffe
RegisterFile	R5	0008
RegisterFile	R6	0000


Properties | State

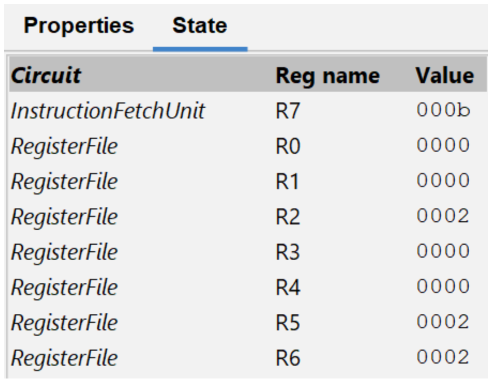
Circuit	Reg name	Value
InstructionFetchUnit	R7	0008
RegisterFile	R0	0000
RegisterFile	R1	0000
RegisterFile	R2	0002
RegisterFile	R3	0000
RegisterFile	R4	fffe
RegisterFile	R5	0008
RegisterFile	R6	0000


Properties | State

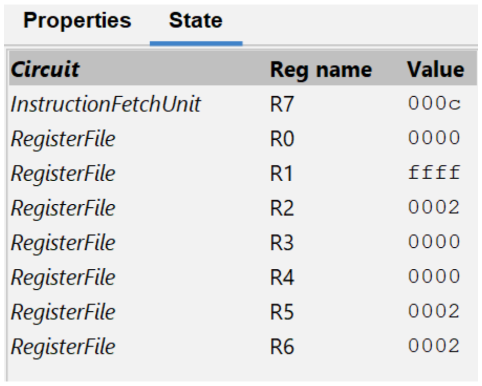
Circuit	Reg name	Value
InstructionFetchUnit	R7	0009
RegisterFile	R0	0000
RegisterFile	R1	0000
RegisterFile	R2	0002
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0008
RegisterFile	R6	0000



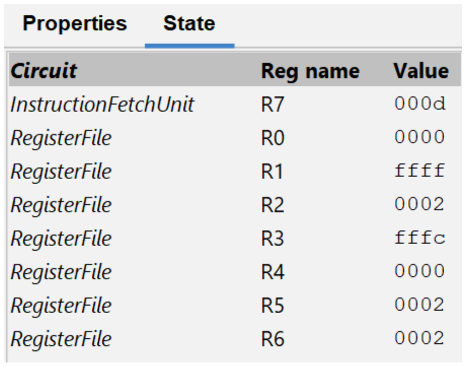
Circuit	Reg name	Value
InstructionFetchUnit	R7	000a
RegisterFile	R0	0000
RegisterFile	R1	0000
RegisterFile	R2	0002
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0002
RegisterFile	R6	0000



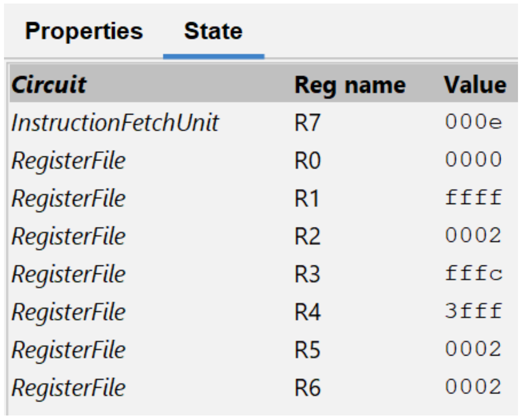
Circuit	Reg name	Value
InstructionFetchUnit	R7	000b
RegisterFile	R0	0000
RegisterFile	R1	0000
RegisterFile	R2	0002
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0002
RegisterFile	R6	0002



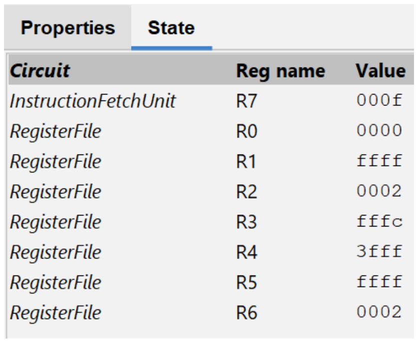
Circuit	Reg name	Value
InstructionFetchUnit	R7	000c
RegisterFile	R0	0000
RegisterFile	R1	ffff
RegisterFile	R2	0002
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0002
RegisterFile	R6	0002



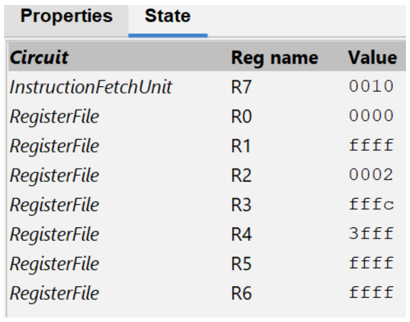
Circuit	Reg name	Value
InstructionFetchUnit	R7	000d
RegisterFile	R0	0000
RegisterFile	R1	ffff
RegisterFile	R2	0002
RegisterFile	R3	fffc
RegisterFile	R4	0000
RegisterFile	R5	0002
RegisterFile	R6	0002



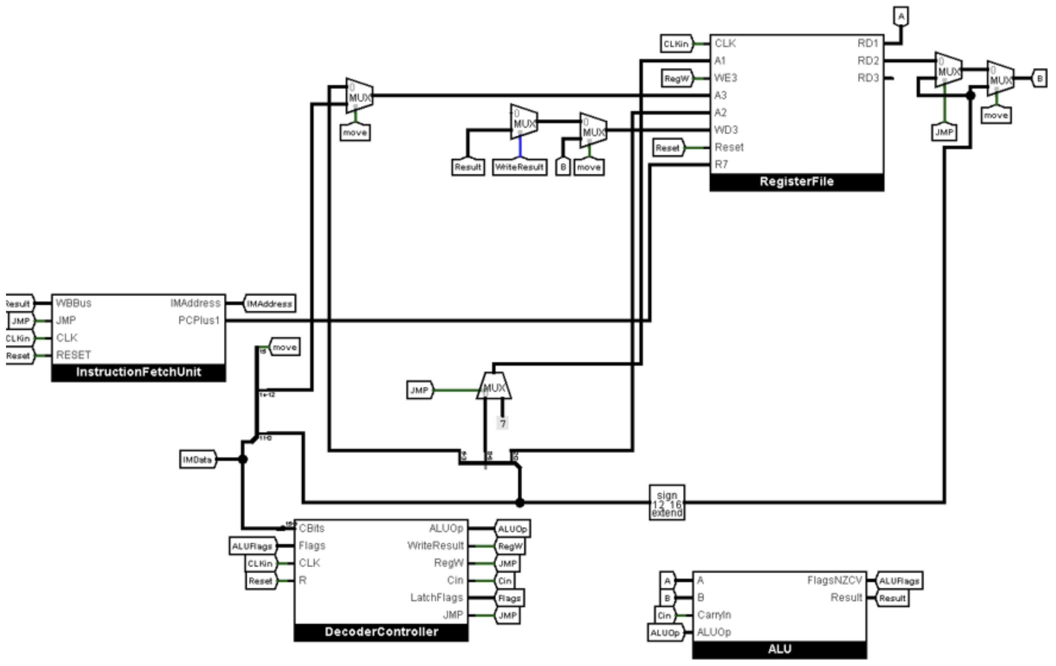
Circuit	Reg name	Value
InstructionFetchUnit	R7	`000e`
RegisterFile	R0	`0000`
RegisterFile	R1	`ffff`
RegisterFile	R2	`0002`
RegisterFile	R3	`fffc`
RegisterFile	R4	`3fff`
RegisterFile	R5	`0002`
RegisterFile	R6	`0002`



Circuit	Reg name	Value
InstructionFetchUnit	R7	`000f`
RegisterFile	R0	`0000`
RegisterFile	R1	`ffff`
RegisterFile	R2	`0002`
RegisterFile	R3	`fffc`
RegisterFile	R4	`3fff`
RegisterFile	R5	`ffff`
RegisterFile	R6	`0002`



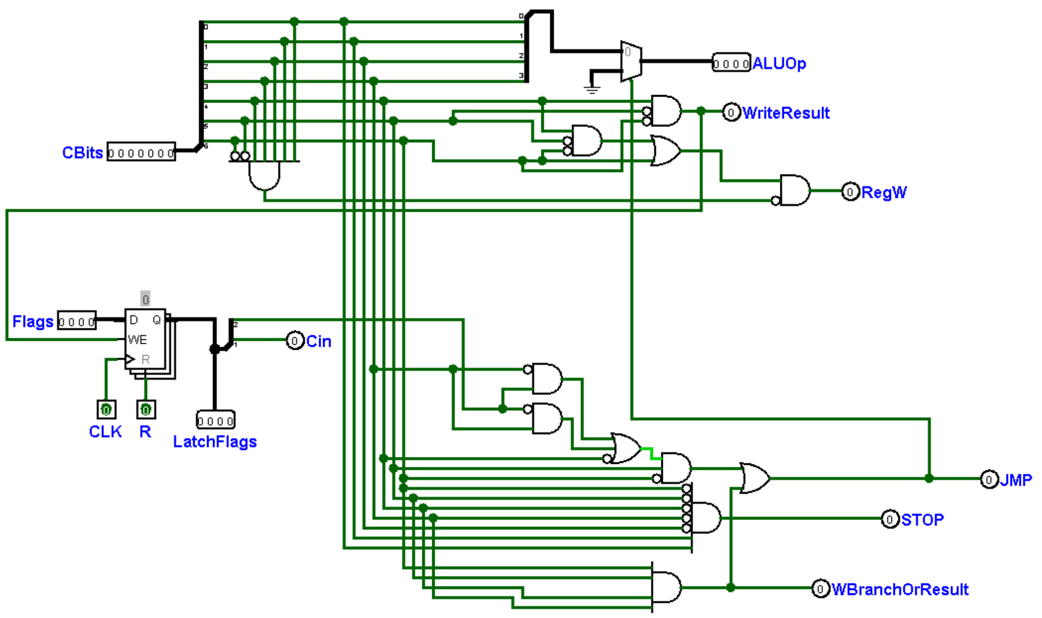
Circuit	Reg name	Value
InstructionFetchUnit	R7	`0010`
RegisterFile	R0	`0000`
RegisterFile	R1	`ffff`
RegisterFile	R2	`0002`
RegisterFile	R3	`fffc`
RegisterFile	R4	`3fff`
RegisterFile	R5	`ffff`
RegisterFile	R6	`ffff`

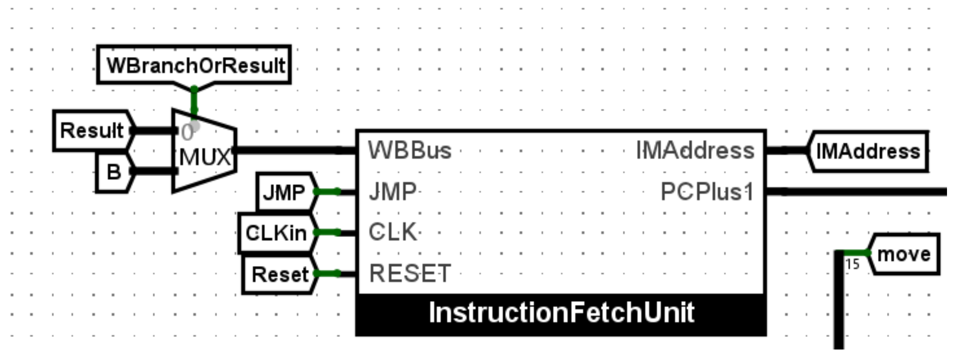


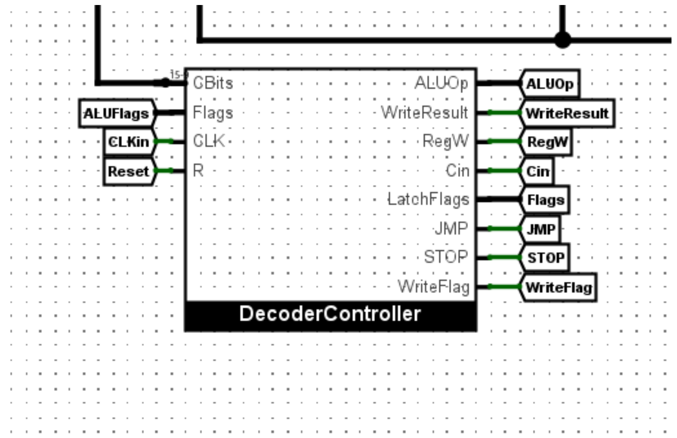
Non ALU Instructions

Move R7 immediate

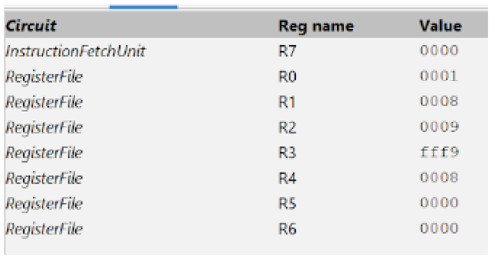
```
Added way to move immediate value into r7. Updated the decoder to
write the result of the arithmetic logic unit or the imm val to the
WBBus.
```







Move R7 immediate (Write branch or result)

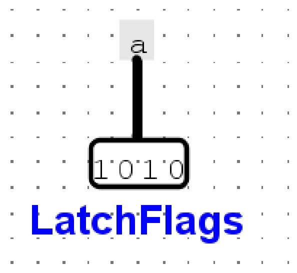


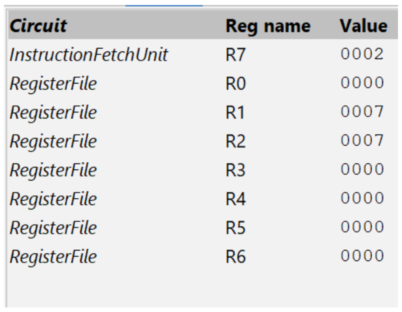
Circuit	Reg name	Value
InstructionFetchUnit	R7	0000
RegisterFile	R0	0001
RegisterFile	R1	0008
RegisterFile	R2	0009
RegisterFile	R3	fff9
RegisterFile	R4	0008
RegisterFile	R5	0000
RegisterFile	R6	0000

Move Register to Register

- Move Rd Rn

```
MOV r1, 0x002        v2.0 raw
MOV r2, r1           9002
                     0850
```



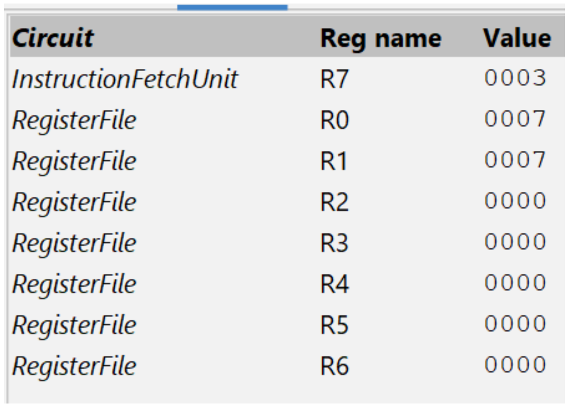


Circuit	Reg name	Value
InstructionFetchUnit	R7	0002
RegisterFile	R0	0000
RegisterFile	R1	0007
RegisterFile	R2	0007
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0000
RegisterFile	R6	0000

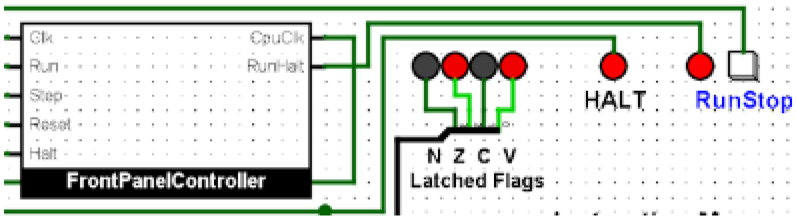
Move Flags Rn

- Moved 7 into R0 and moved it into latch flags (0101 = 5)

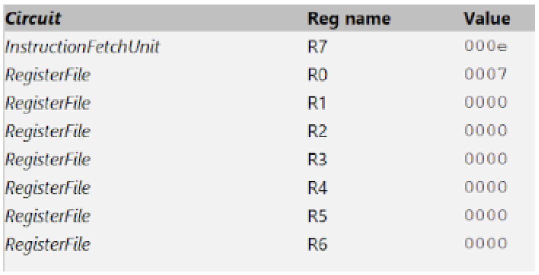
```
v2.0 raw
8007
0E00
0C08
```



Circuit	Reg name	Value
InstructionFetchUnit	R7	0003
RegisterFile	R0	0007
RegisterFile	R1	0007
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0000
RegisterFile	R6	0000



- Moved 7 into R0 and moved it into latch flags (0111 = 7)

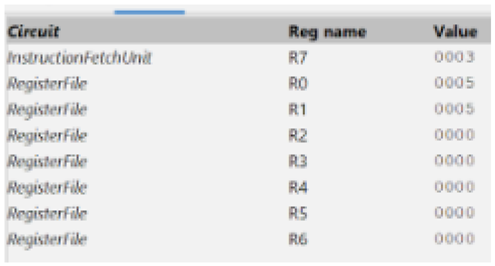


Circuit	Reg name	Value
InstructionFetchUnit	R7	000e
RegisterFile	R0	0007
RegisterFile	R1	0000
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0000
RegisterFile	R6	0000

Move Rd, Flags

- Moved 5 into R0 and moved it into latch flags (0101 = 5). Also, moved latch flags into R1 (0101 = 5).

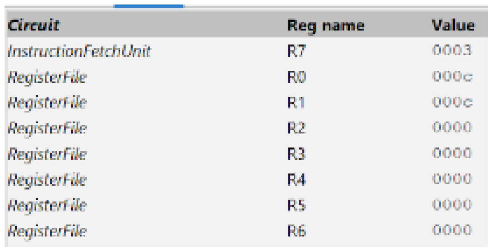
```
v2.0 raw
9005
0C10
0898

MOV R0, 0x005
MOV Flags, R0
MOV R1, Flags
```



Circuit	Reg name	Value
InstructionFetchUnit	R7	0003
RegisterFile	R0	0005
RegisterFile	R1	0005
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0000
RegisterFile	R6	0000

- Moved C into R0 and moved it into latch flags (1100 = C). Also, moved latch flags into R1 (1100 = C).

```
MOV R0, 0x00C
MOV Flags, R0
MOV R1, Flags
```



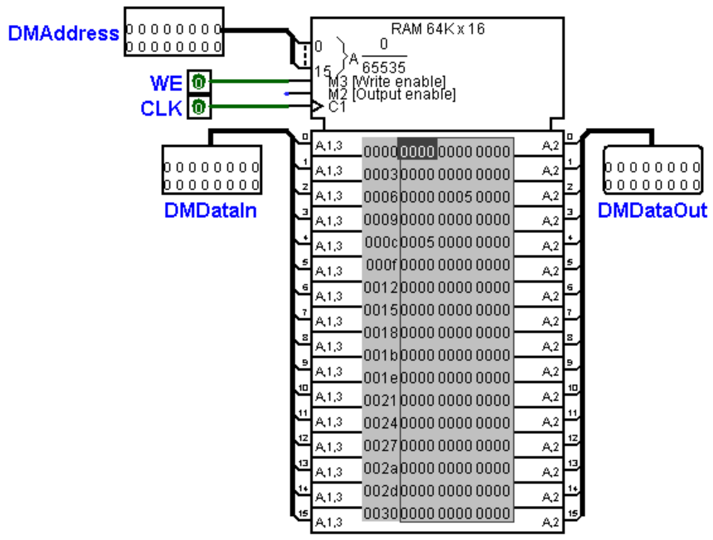
Circuit	Reg name	Value
InstructionFetchUnit	R7	0003
RegisterFile	R0	000c
RegisterFile	R1	000c
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0000
RegisterFile	R5	0000
RegisterFile	R6	0000

Update Decoder and CPU

Store # in Mem[#] and Mem [#]

- Store 5 in Mem[7] and Mem[12]

```
v2.0 raw
8005
9007
1E40
1801
1660
1029
```

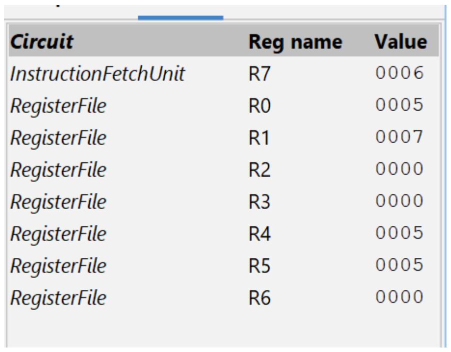


```
0000 0000 0000 0000
0003 0000 0000 0000
0006 0000 0005 0000
0009 0000 0000 0000
000c 0005 0000 0000
000f 0000 0000 0000
0012 0000 0000 0000
0015 0000 0000 0000
0018 0000 0000 0000
001b 0000 0000 0000
001e 0000 0000 0000
0021 0000 0000 0000
0024 0000 0000 0000
0027 0000 0000 0000
002a 0000 0000 0000
002d 0000 0000 0000
0030 0000 0000 0000
```

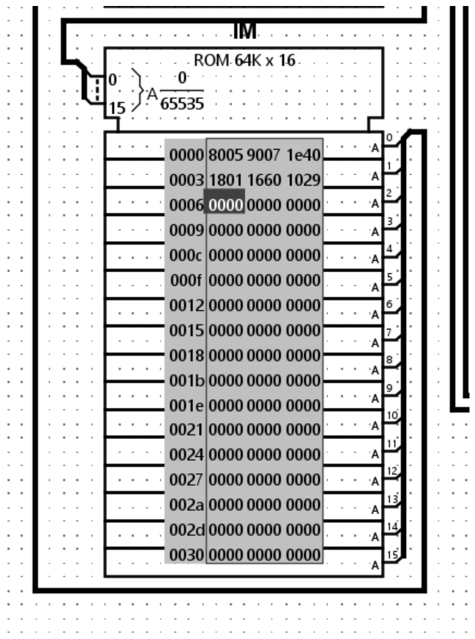
Load # in R4 and R5

- `load 5 in r4 and r5`

```
v2.0 raw
8005
9007
1E40
1801
1660
1029
```



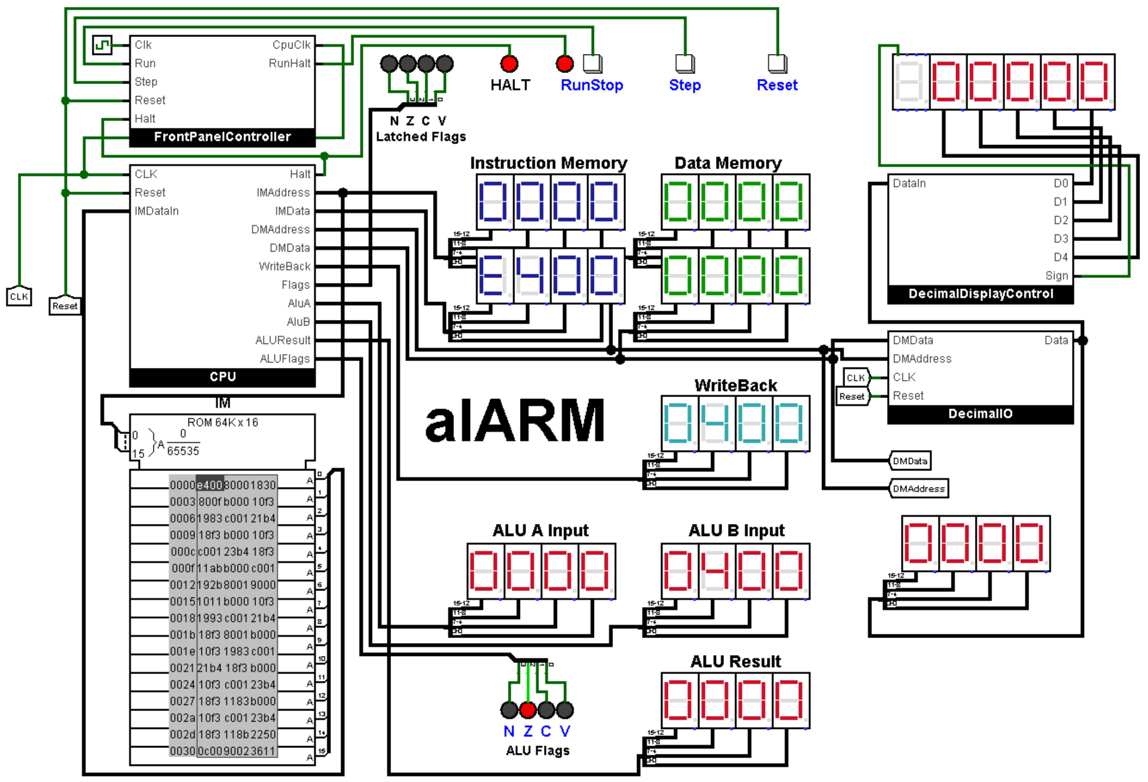
Circuit	Reg name	Value
InstructionFetchUnit	R7	0006
RegisterFile	R0	0005
RegisterFile	R1	0007
RegisterFile	R2	0000
RegisterFile	R3	0000
RegisterFile	R4	0005
RegisterFile	R5	0005
RegisterFile	R6	0000



```
0000 8005 9007 1e40
0003 1801 1660 1029
0006 0000 0000 0000
0009 0000 0000 0000
000c 0000 0000 0000
000f 0000 0000 0000
0012 0000 0000 0000
0015 0000 0000 0000
0018 0000 0000 0000
001b 0000 0000 0000
001e 0000 0000 0000
0021 0000 0000 0000
0024 0000 0000 0000
0027 0000 0000 0000
002a 0000 0000 0000
002d 0000 0000 0000
0030 0000 0000 0000
```

Appendix

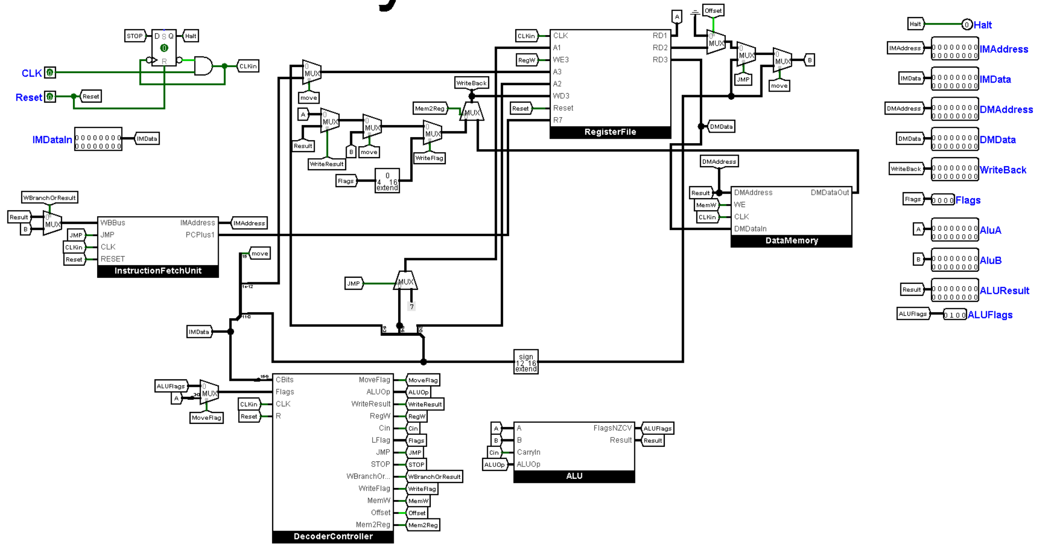
- Main



```
0000 e400 8000 1830
0003 800f b000 10f3
0006 1983 c001 21b4
0009 18f3 b000 10f3
000c c001 23b4 18f3
000f 11ab b000 c001
0012 192b 8001 9000
0015 1011 b000 10f3
0018 1993 c001 21b4
001b 18f3 8001 b000
001e 10f3 1983 c001
0021 21b4 18f3 b000
0024 10f3 c001 23b4
0027 18f3 1183 b000
002a 10f3 c001 23b4
002d 18f3 118b 2250
0030 0c00 9002 3611
```

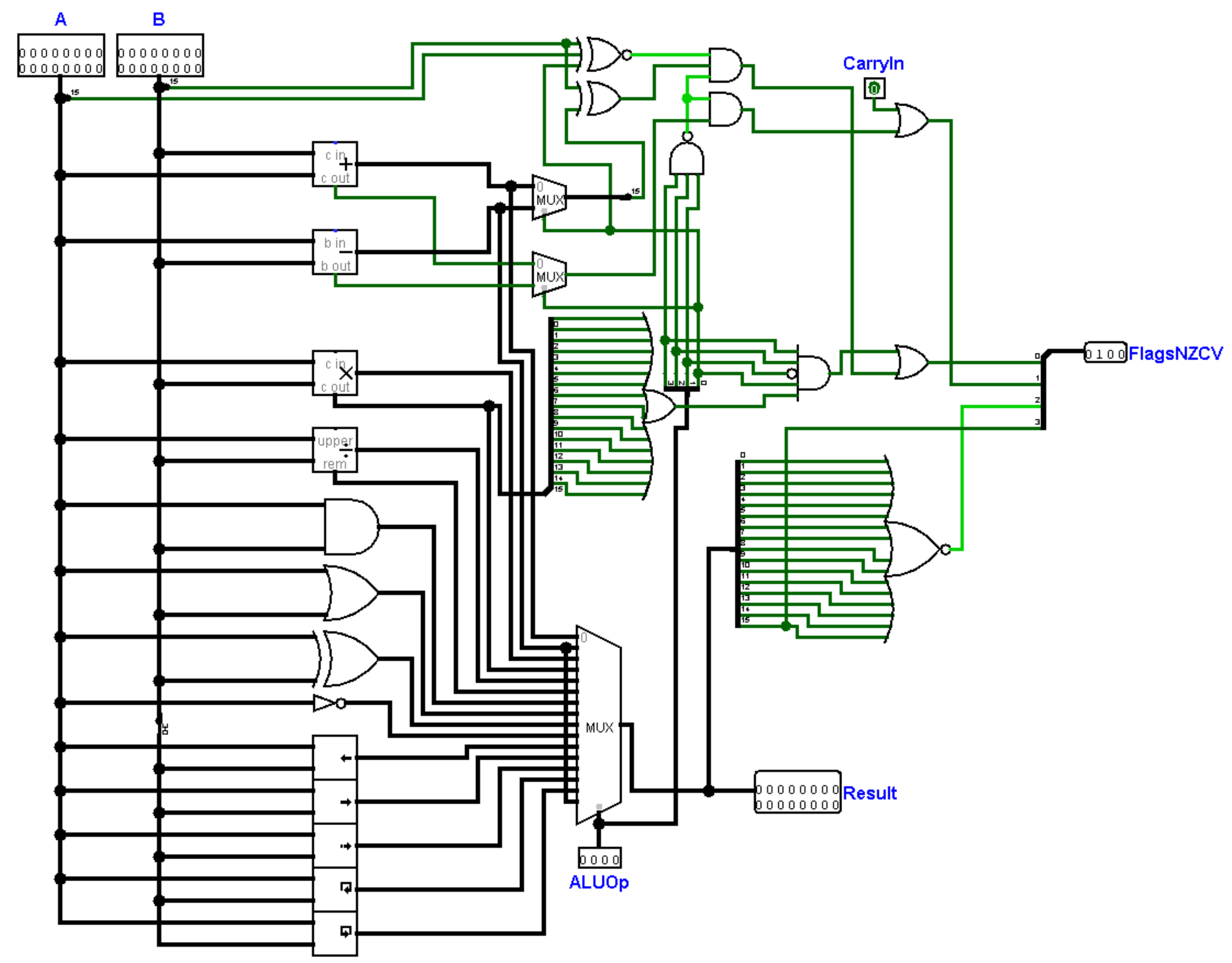
- CPU

Build your CPU here

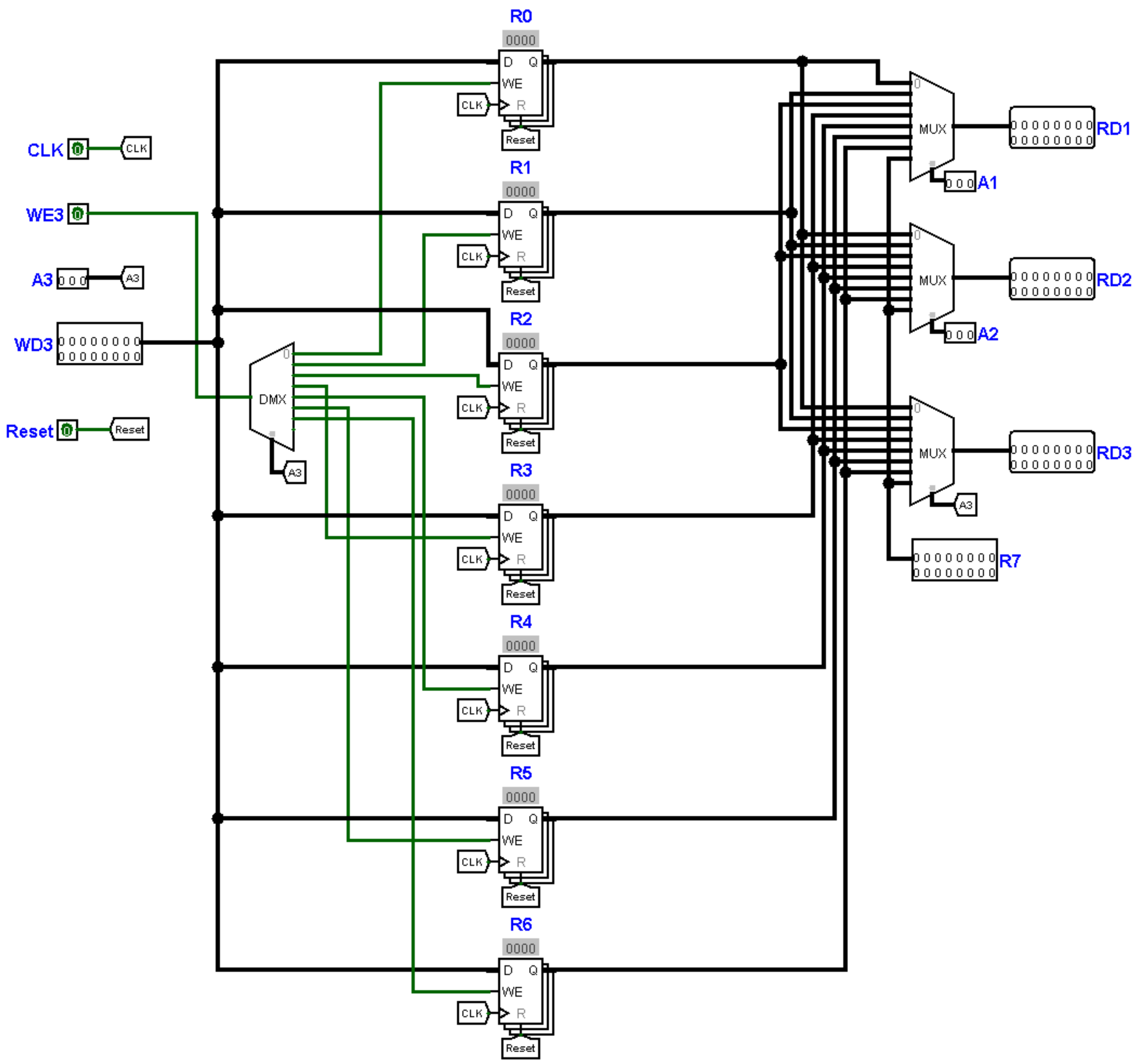


- ALU

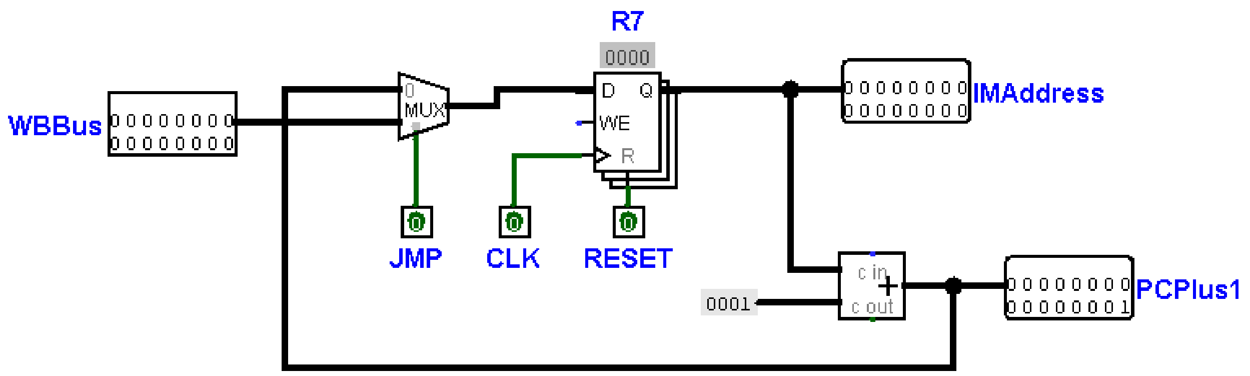
Arithmetic Logic Unit. Takes care of all the additions, subtractions, division, multiplication, AND, OR, XOR, logical left shift, logical right shift, arithmetic right shift, right rotation, and left rotation.



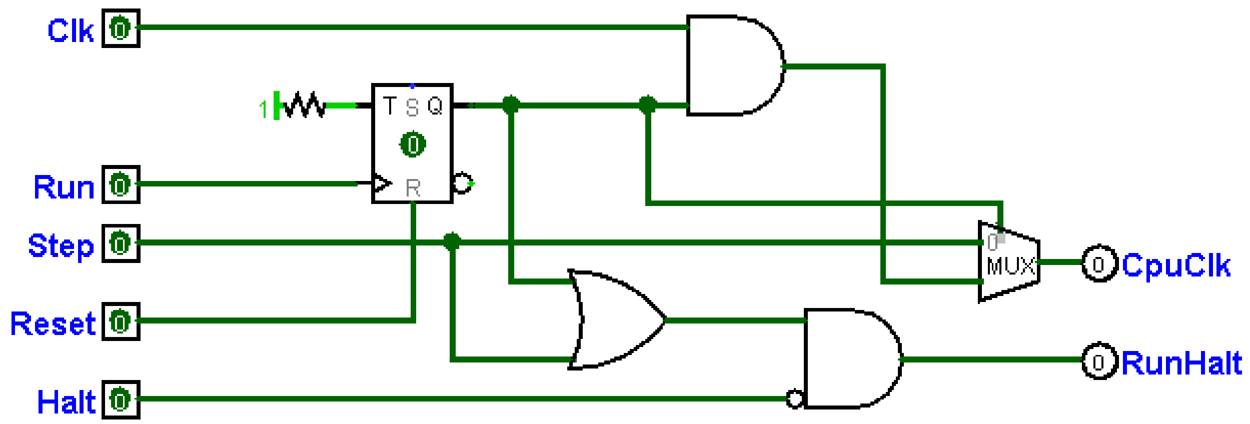
– Register File



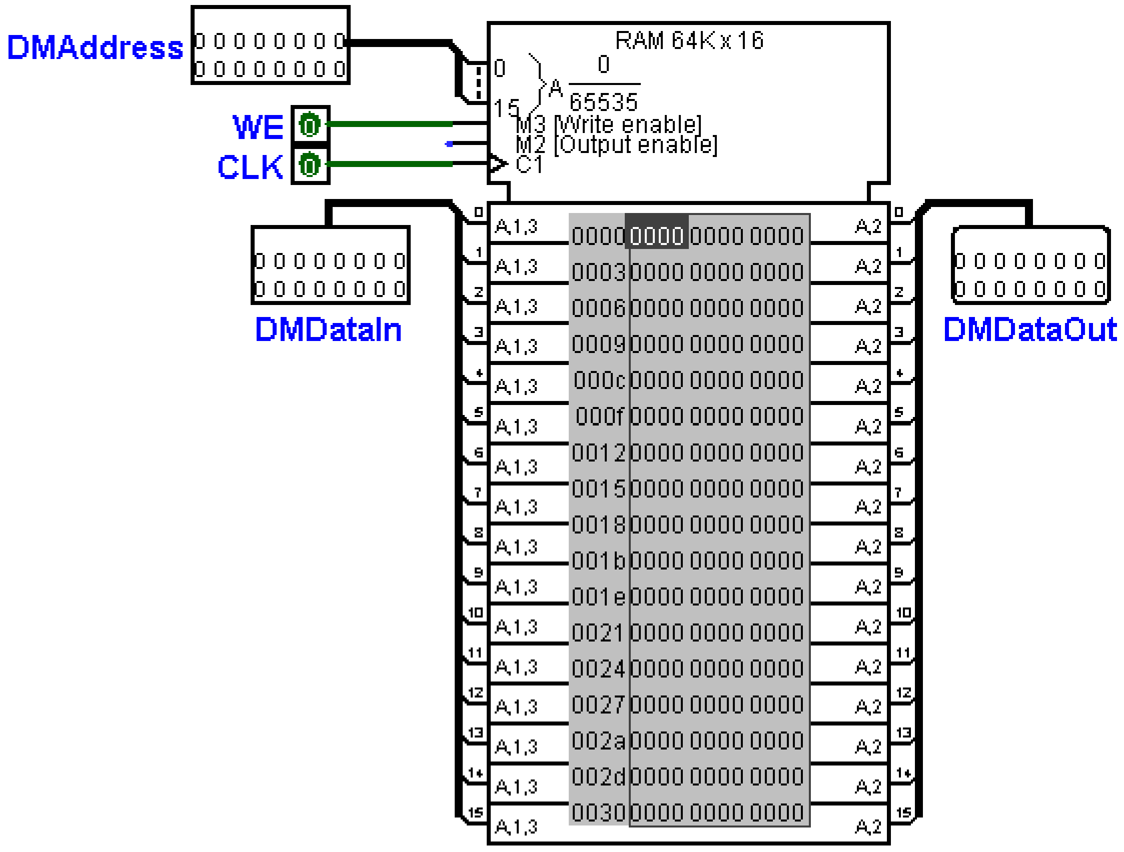
- Instruction Fetch Unit



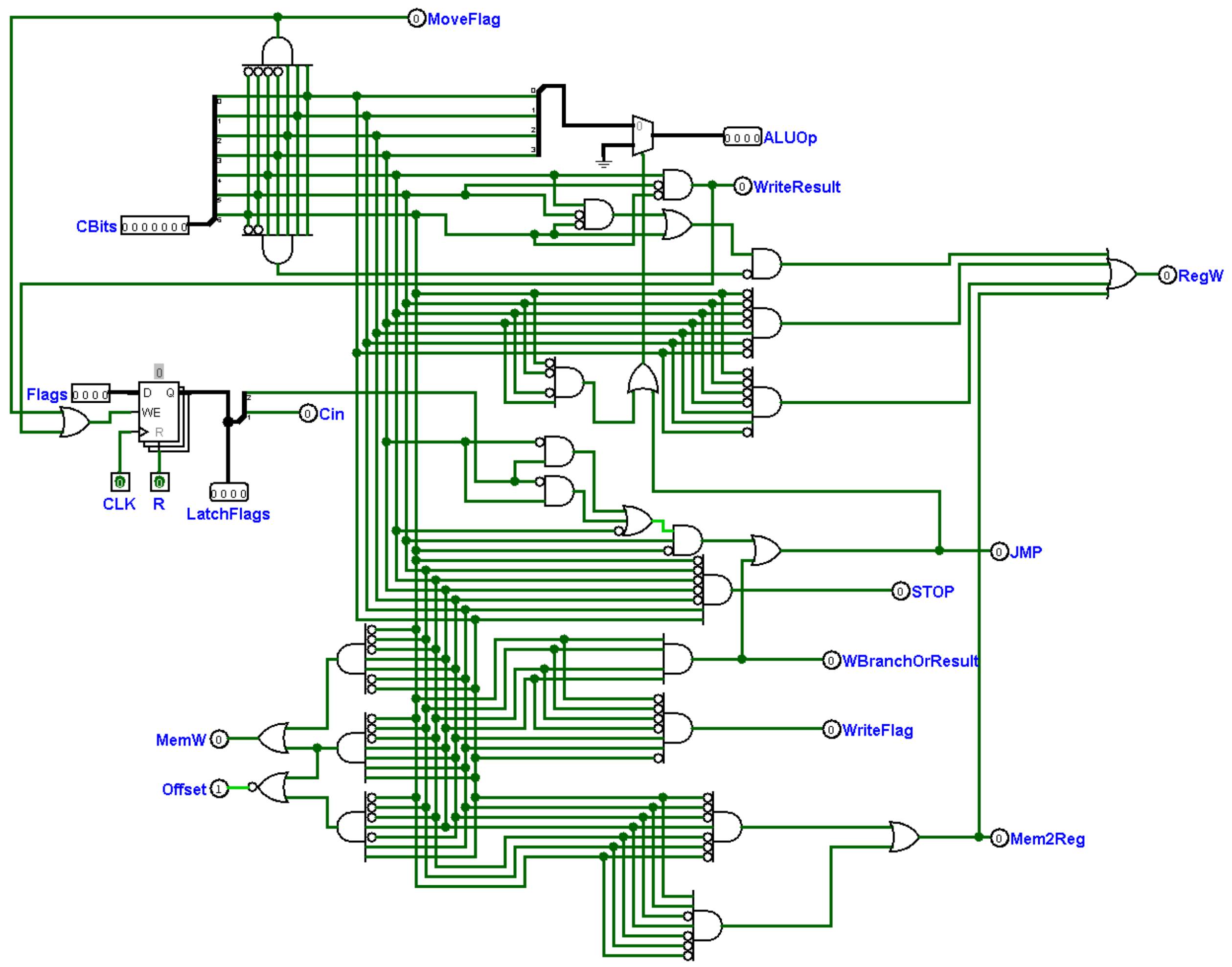
- Front Panel Controller



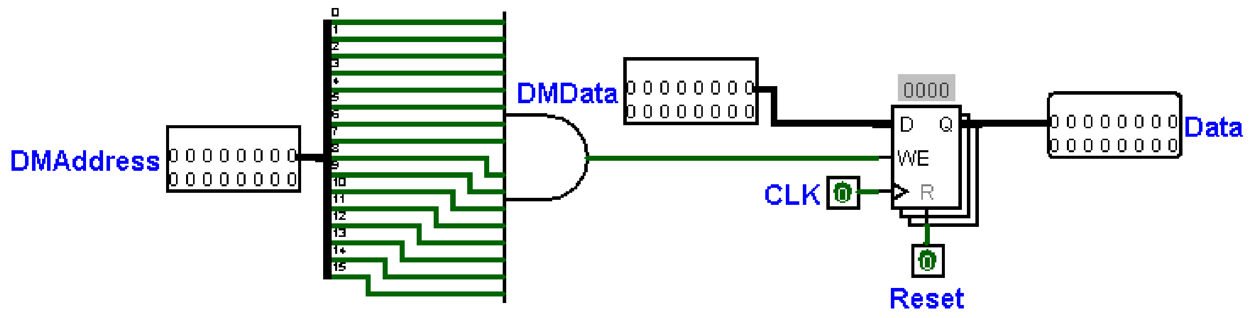
- Data Memory



- Decoder Controller

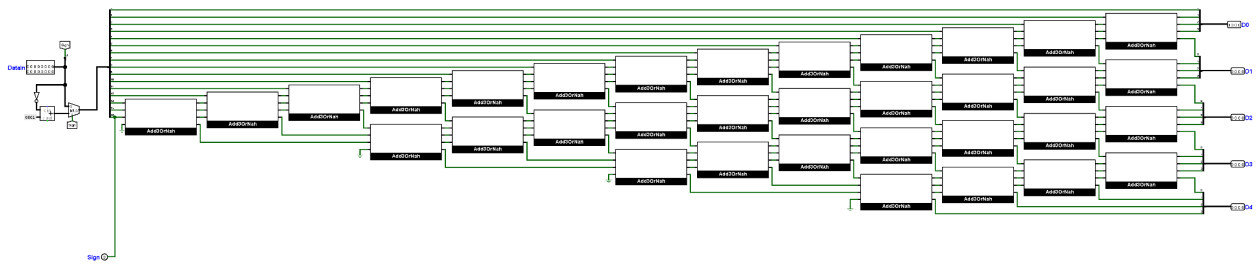


- DecimalIO



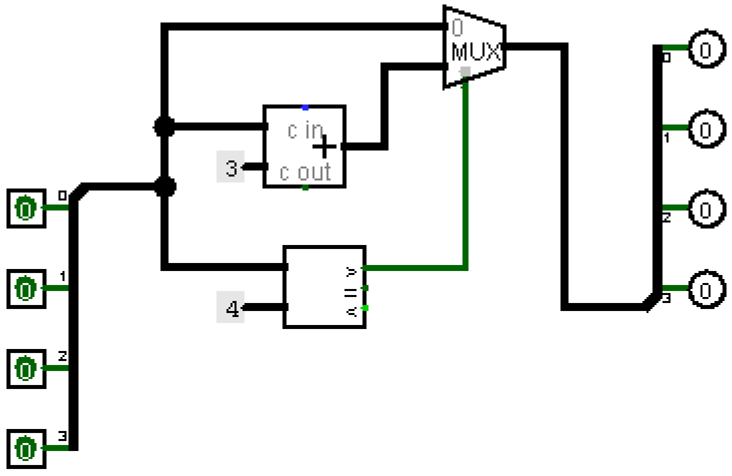
- Decimal Display Control

The Decimal Display Control takes care of performing the Double Dabble algorithm. It utilizes a sub circuit named Add3OrNot to determine if it should add 3. It basically shifts the bits.



- Add3OrNot

This is used by the Decimal Display controller and determines if it should add 3 or not to the Binary Coded Decimal that's dealt with in the Decimal Display Control. It basically adds 3 if the 4 bits in that region are greater than 5. Otherwise, it just routes the bits.



Hex code

- Display -4242

```
v2.0 raw
8000
9FFF
D12F
AFF2
256A
1829
```

- Given Code

```
v2.0 raw
E400
8000
1830
800F
B000
```

```
10F3
1983
C001
21B4
18F3
B000
10F3
C001
23B4
18F3
11AB
B000
C001
192B
8001
9000
1011
B000
10F3
1993
C001
21B4
18F3
8001
B000
10F3
1983
C001
21B4
18F3
B000
10F3
C001
23B4
18F3
1183
B000
10F3
C001
23B4
18F3
118B
2250
0C00
9002
3611
```

```
0C00
3611
B001
2C93
B000
10F3
1993
C001
21B4
18F3
B000
10F3
C001
23B4
1183
2C00
60B0
8001
9000
1011
B000
10F3
1993
C001
21B4
18F3
B000
10F3
C001
23B4
18F3
11AB
8000
9FFF
1829
8001
9000
1011
B000
10F3
1993
C001
21B4
18F3
8001
B000
```

```
10F3
1983
C001
21B4
18F3
B000
10F3
C001
23B4
18F3
1183
B000
10F3
C001
23B4
18F3
118B
2C50
B000
10F3
1993
C001
21B4
18F3
B000
10F3
C001
23B4
1183
2C00
6047
8001
9000
1011
B000
10F3
1993
C001
21B4
18F3
8003
B000
10F3
1983
C001
21B4
```

```
18F3
B000
10F3
C001
23B4
18F3
1183
B000
10F3
C001
23B4
18F3
118B
2450
B000
10F3
1993
C001
21B4
18F3
8001
B000
10F3
1983
C001
21B4
18F3
B000
10F3
C001
23B4
18F3
1183
B000
10F3
C001
23B4
18F3
118B
2050
B000
10F3
1993
C001
21B4
18F3
```

```
B000
10F3
C001
23B4
18F3
11AB
B000
C001
192B
402C
8001
9000
1011
B000
10F3
1993
C001
21B4
18F3
8001
B000
10F3
1983
C001
21B4
18F3
B000
10F3
C001
23B4
18F3
1183
B000
10F3
C001
23B4
18F3
118B
3650
B000
10F3
1993
C001
21B4
18F3
B000
```

```
10F3
C001
23B4
18F3
11AB
B000
C001
192B
4F1F
8001
9000
1011
B000
10F3
1993
C001
21B4
18F3
B000
10F3
C001
23B4
18F3
11AB
8000
9FFF
1829
0600
```

- fibo.h

```
8014  ;MOV  R0, 0x014
9080  ;MOV  R1, 0x080
1E40  ;STR  R0, [R1]
1660  ;LDR  R4, [R1]
D001  ;MOV  R5, 0x001
E000  ;MOV  R6, 0x000
8000  ;MOV  R0, 0x000
9001  ;MOV  R1, 0x001
1F80  ;STR  R0, [R6]
21B5  ;ADD  R6, R6, R5
1F88  ;STR  R1, [R6]
21B5  ;ADD  R6, R6, R5
3F06  ;CMP  R4, R6
6007  ;BEQ  0x007
2001  ;ADD  R0, R0, R1
2009  ;ADD  R1, R0, R1
1F80  ;STR  R0, [R6]
21B5  ;ADD  R6, R6, R5
1F88  ;STR  R1, [R6]
21B5  ;ADD  R6, R6, R5
4FF7  ;B    0xFF7
0600  ;HALT
```