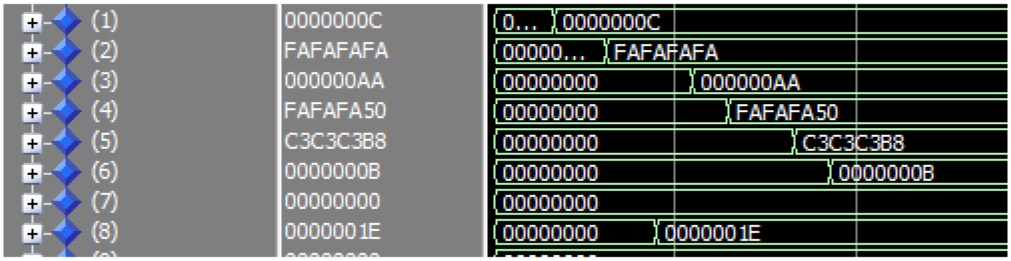
Project Report

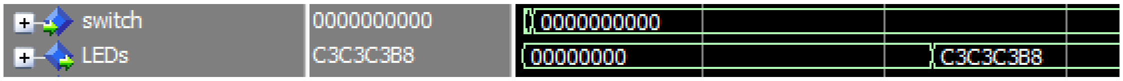
Week 2 Deliverables:

Week 2 Test Case:



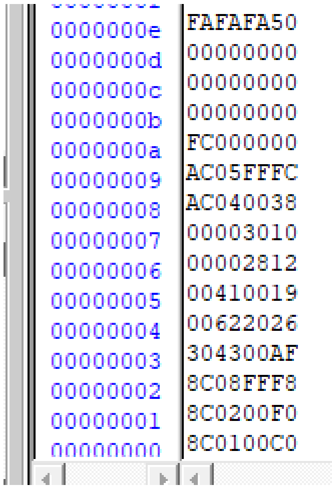
```
(1)  0000000C
(2)  FAFAFAFA
(3)  000000AA
(4)  FAFAFA50
(5)  C3C3C3B8
(6)  0000000B
(7)  00000000
(8)  0000001E
```

The registers in the file register^^



```
switch  0000000000
LEDs    C3C3C3B8
```

Outport^^

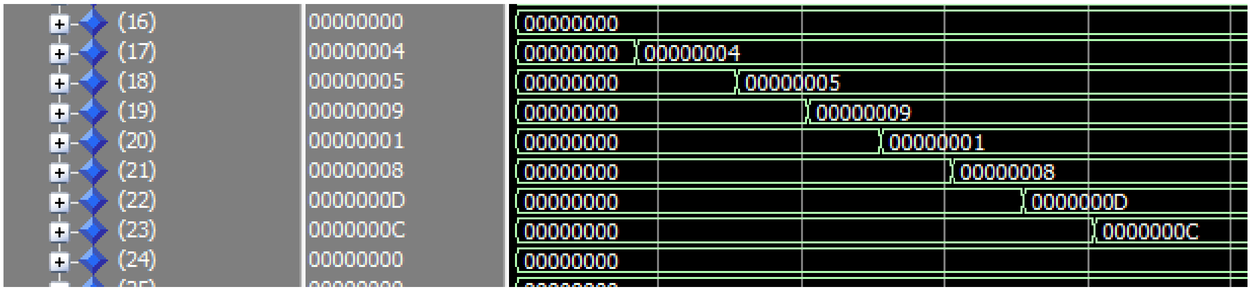


```
0000000e  FAFAFA50
0000000d  00000000
0000000c  00000000
0000000b  00000000
0000000a  FC000000
00000009  AC05FFFC
00000008  AC040038
00000007  00003010
00000006  00002812
00000005  00410019
00000004  00622026
00000003  304300AF
00000002  8C08FFF8
00000001  8C0200F0
00000000  8C0100C0
```

Data in the ram

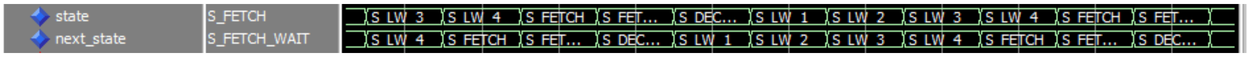
Week 3 Deliverables:

Test Case 1:



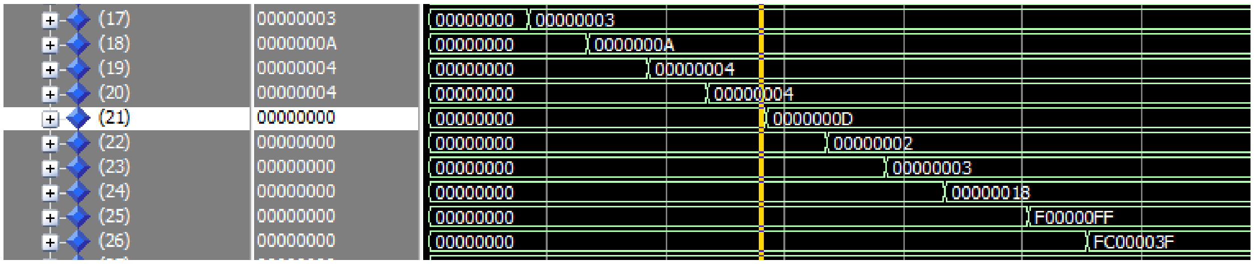
```
(16)  00000000
(17)  00000004
(18)  00000005
(19)  00000009
(20)  00000001
(21)  00000008
(22)  0000000D
(23)  0000000C
(24)  00000000
```

Easiest to look at the register in ^^ this gray column.

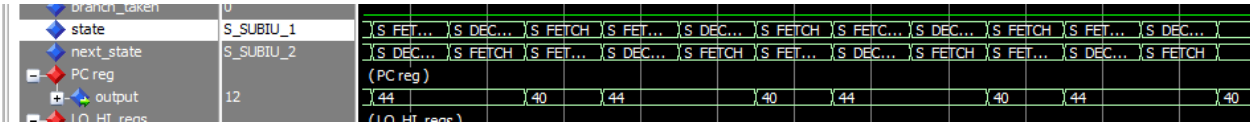

state S_FETCH
next_state S_FETCH_WAIT
S LW 3 | S LW 4 | S FETCH | S FET... | S DEC... | S LW 1 | S LW 2 | S LW 3 | S LW 4 | S FETCH | S FET...
S LW 4 | S FETCH | S FET... | S DEC... | S LW 1 | S LW 2 | S LW 3 | S LW 4 | S FETCH | S FET... | S DEC...

Example of some of the states. Load word starts with fetch, fetch wait, lw_1, lw_2, lw_3, lw_4, then goes onto the fetch stage for the next instruction.

Test Case 2:

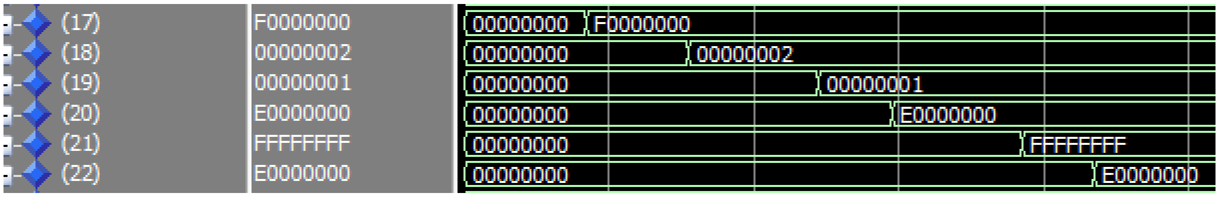


Registers for test case 2.


branch_taken 0
state S_SUBIU_1
S FET... | S DEC... | S FETCH | S FET... | S DEC... | S FETCH | S FETC... | S DEC... | S FETCH | S FET... | S DEC...
next_state S_SUBIU_2
S DEC... | S FETCH | S FET... | S DEC... | S FETCH | S FET... | S DEC... | S FETCH | S FET... | S DEC... | S FETCH
PC reg
(PC reg)
output 12
44 | 40 | 44 | 40 | 44 | 40 | 44 | 40
LO_HI_regs
(LO_HI_regs)

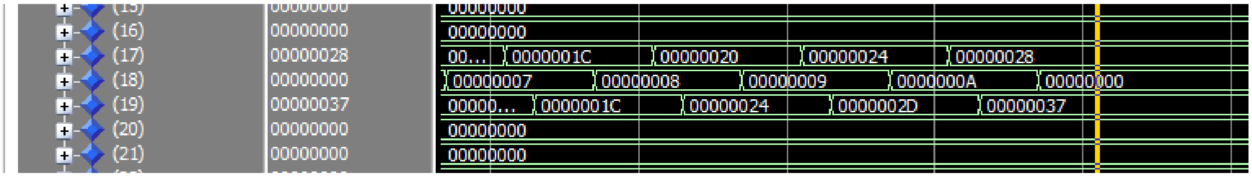
Example of some states, this is for the end jump that keeps jumping to itself at the end of the assembly program. You can see the output of the PC register staying at 40, which translates

Test Case 4:

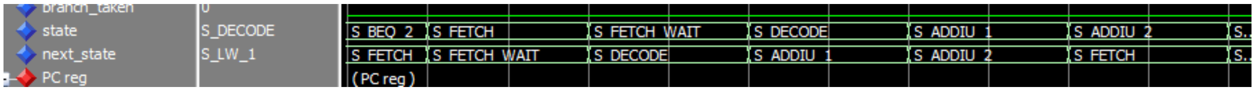

(17) F0000000 | 00000000 | F0000000
(18) 00000002 | 00000000 | 00000002
(19) 00000001 | 00000000 | 00000001
(20) E0000000 | 00000000 | E0000000
(21) FFFFFFFF | 00000000 | FFFFFFFF
(22) E0000000 | 00000000 | E0000000

Registahs

Test Case 7:

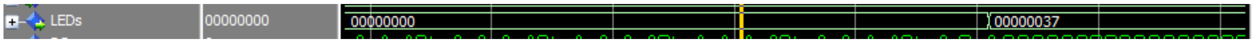

(15) 00000000 | 00000000
(16) 00000000 | 00000000
(17) 00000028 | 00... | 0000001C | 00000020 | 00000024 | 00000028
(18) 00000000 | 00000007 | 00000008 | 00000009 | 0000000A | 00000000
(19) 00000037 | 0000... | 0000001C | 00000024 | 0000002D | 00000037
(20) 00000000 | 00000000
(21) 00000000 | 00000000

Register 18 loops through 10, returning to 0, summing up to 0x37 in register 19


branch_taken 0
state S_DECODE
S BEQ 2 | S FETCH | S FETCH WAIT | S DECODE | S ADDIU 1 | S ADDIU 2 | S..
next_state S_LW_1
S FETCH | S FETCH WAIT | S DECODE | S ADDIU 1 | S ADDIU 2 | S FETCH | S..
PC reg
(PC reg)

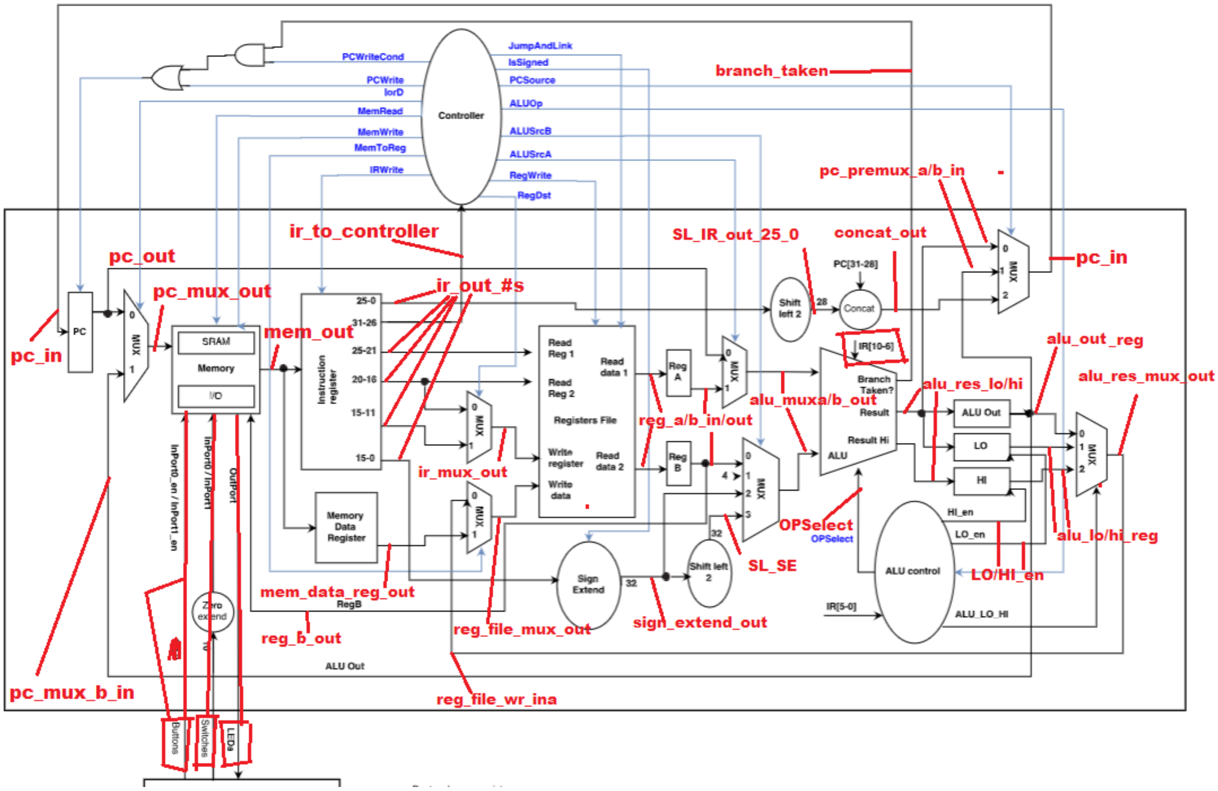
Example of states changing. In this example, for the addiu command, it starts by going through the fetch state, the fetch delay, then the decode stage, which also acts as a delay in this example.

Then the machine passes through states S_ADDIU_1 and S_ADDIU_2 before going back to the S_FETCH stage.

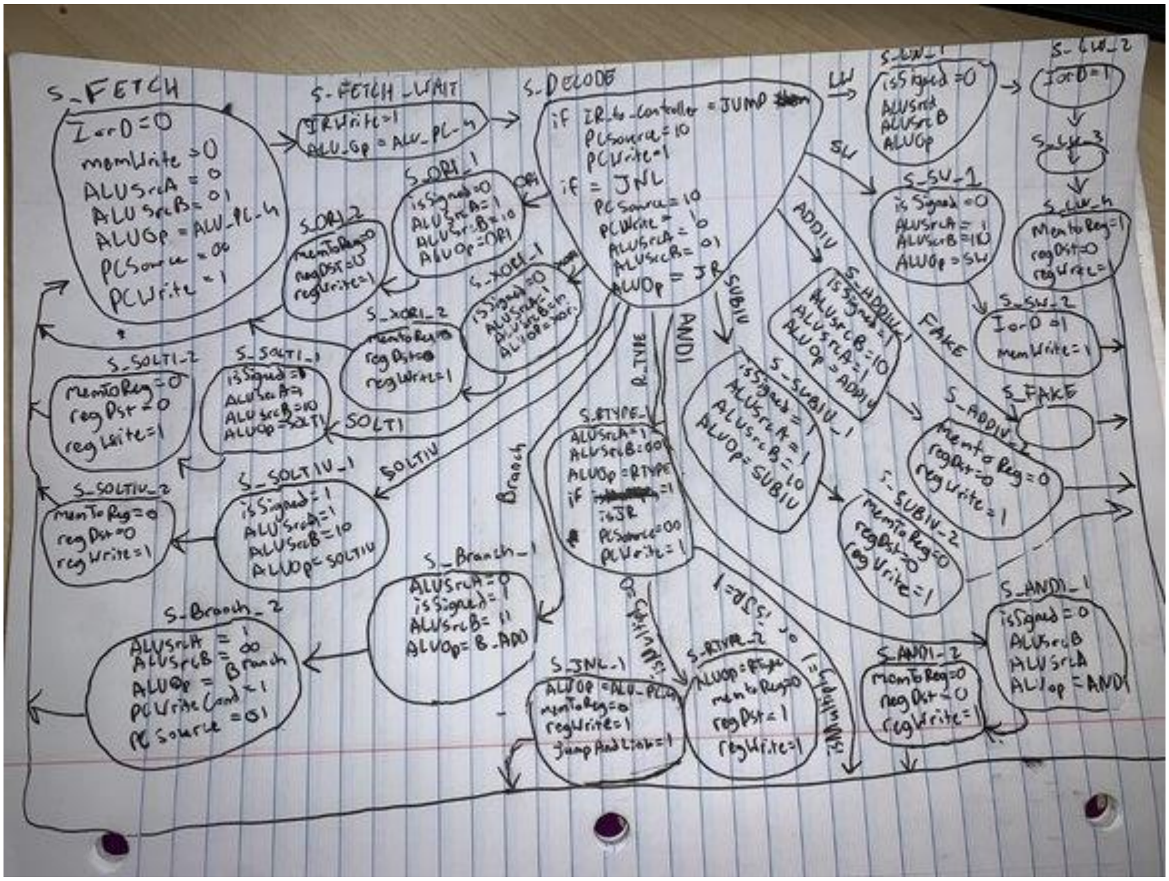


Output register comes from register 19.

Datapath:


JumpAndLink
PCWriteCond
IsSigned
PCWrite
IorD
PCSource
MemRead
ALUOp
Controller
MemWrite
ALUSrcB
MemToReg
ALUSrcA
IRWrite
RegWrite
RegDst
branch_taken
pc_premux_a/b_in
ir_to_controller
SL_IR_out_25_0
concat_out
pc_out
PC[31-28]
pc_in
pc_mux_out
ir_out_#s
25-0
Shift left 2
28
Concat
PC
MUX
0
1
SRAM
Memory
I/O
mem_out
31-26
25-21
20-16
15-11
15-0
Instruction register
IR[10-6]
alu_out_reg
pc_in
Read Reg 1
Read data 1
Read Reg 2
Registers File
Write register
Read data 2
Write data
Reg A
Reg B
Branch Taken?
Result
Result Hi
ALU
alu_res_lo/hi
alu_res_mux_out
alu_muxa/b_out
reg_a/b_in/out
ALU Out
LO
HI
ir_mux_out
4
2
3
MUX
InPort0_en / InPort1_en
InPort0 / InPort1
OutPort
Memory Data Register
Sign Extend
32
Shift left 2
OPSelect
OPSelect
SL_SE
HI_en
LO_en
ALU control
alu_lo/hi_reg
LO/HI_en
mem_data_reg_out
RegB
Zero extend
reg_b_out
reg_file_mux_out
sign_extend_out
IR[5-0]
ALU_LO_HI
ALU Out
pc_mux_b_in
reg_file_wr_ina
Buttons
Switches
LEDs

FSM:


S_FETCH
IorD=0
memWrite=0
ALUSrcA=0
ALUSrcB=01
ALUOp=ALU_PC_4
PCSource=00
PCWrite=1
S_FETCH_WAIT
IRWrite=1
ALU_Op=ALU_PC_4
S_DECODE
if IR_to_Controller=JUMP
PCSource=10
PCWrite=1
if = JNL
PCSource=10
PCWrite=10
ALUSrcA=0
ALUSrcB=01
ALUOp=JR
ORI
S_ORI_1
isSigned=0
ALUSrcA=1
ALUSrcB=10
ALUOp=ORI
S_ORI_2
memToReg=0
regDst=0
regWrite=1
S_XORI_1
isSigned=0
ALUSrcA=1
ALUSrcB=10
ALUOp=XORI
S_XORI_2
memToReg=0
regDst=0
regWrite=1
SOLTI
S_SOLTI_1
isSigned=1
ALUSrcA=1
ALUSrcB=10
ALUOp=SOLTI
S_SOLTI_2
memToReg=0
regDst=0
regWrite=1
SOLTIU
S_SOLTIU_1
isSigned=1
ALUSrcA=1
ALUSrcB=10
ALUOp=SOLTIU
S_SOLTIU_2
memToReg=0
regDst=0
regWrite=1
Branch
S_Branch_1
ALUSrcA=0
isSigned=1
ALUSrcB=11
ALUOp=B_ADD
S_Branch_2
ALUSrcA=1
ALUSrcB=00
ALUOp=Branch
PCWriteCond=1
PCSource=01
R_TYPE
S_RTYPE_1
ALUSrcA=1
ALUSrcB=00
ALUOp=RTYPE
if =1
isJR
PCSource=00
PCWrite=1
S_JNL_1
ALUOp=ALU_PC_4
memToReg=0
regWrite=1
JumpAndLink=1
S_RTYPE_2
ALUOp=RType
memToReg=0
regDst=1
regWrite=1
ANDI
S_ANDI_1
isSigned=0
ALUSrcB
ALUSrcA
ALUOp=ANDI
S_ANDI_2
memToReg=0
regDst=0
regWrite=1
SUBIU
S_SUBIU_1
isSigned=1
ALUSrcA=1
ALUSrcB=10
ALUOp=SUBIU
S_SUBIU_2
memToReg=0
regDst=0
regWrite=1
ADDIU
S_ADDIU_1
isSigned=1
ALUSrcB=10
ALUSrcA=1
ALUOp=ADDIU
S_ADDIU_2
memToReg=0
regDst=0
regWrite=1
FAKE
S_FAKE
LW
S_LW_1
isSigned=0
ALUSrcA
ALUSrcB
ALUOp
S_LW_2
IorD=1
S_LW_3
S_LW_4
MemToReg=1
regDst=0
regWrite=
SW
S_SW_1
isSigned=0
ALUSrcA=1
ALUSrcB=10
ALUOp=SW
S_SW_2
IorD=1
memWrite=1