Introduction:

(I am submitting one day late and using the grace day provided to us (which I have not used earlier))

There are 9 files namely alu.vhd, control_fsm.vhd, data_memory.vhd, flags.vhd, register_file.vhd, stage5.vhd, shifter.vhd, PMconnect.vhd and testbench.vhd.

alu.vhd: handles all the arithmetic operations and also incrementes pc.

data_memory.vhd: reads and write in the program memory and the data memory. The size of data_memory is 128x32 out of which first half stores program memory and second half stores data memory.

flags.vhd: it sets the C, N, V and Z flags.

register_file.vhd: it reads and write in the 16 registers of the program.

control_fsm.vhd: it sets all the control signals depending on the current state and update the state also.

stage6.vhd: this is glue code for this stage.

shifter.vhd: this handles all the shift operations. data, amount and type of shift are its input and it outputs final data and carry.

Pmconnect.vhd Connector the register file and data memory.

testbench.vhd to test the code.

- To run new code paste instruction set in data memory.
- I have added 1 new state to fsm, now it has 13 states.
- The new state(13) added write back into the register for ldr/str instruction if needed.
- Pmconnect for store instruction is done in last state of store while writing in data memory (i.e. 10).
- Pmconnect for load instruction is done in last state of load while writing in register file (i.e. 12).
- New control signals are
    - pmIns: stores instruction to be passed to Pmconnect.
    - Wsrc: tells whether to write back or not.

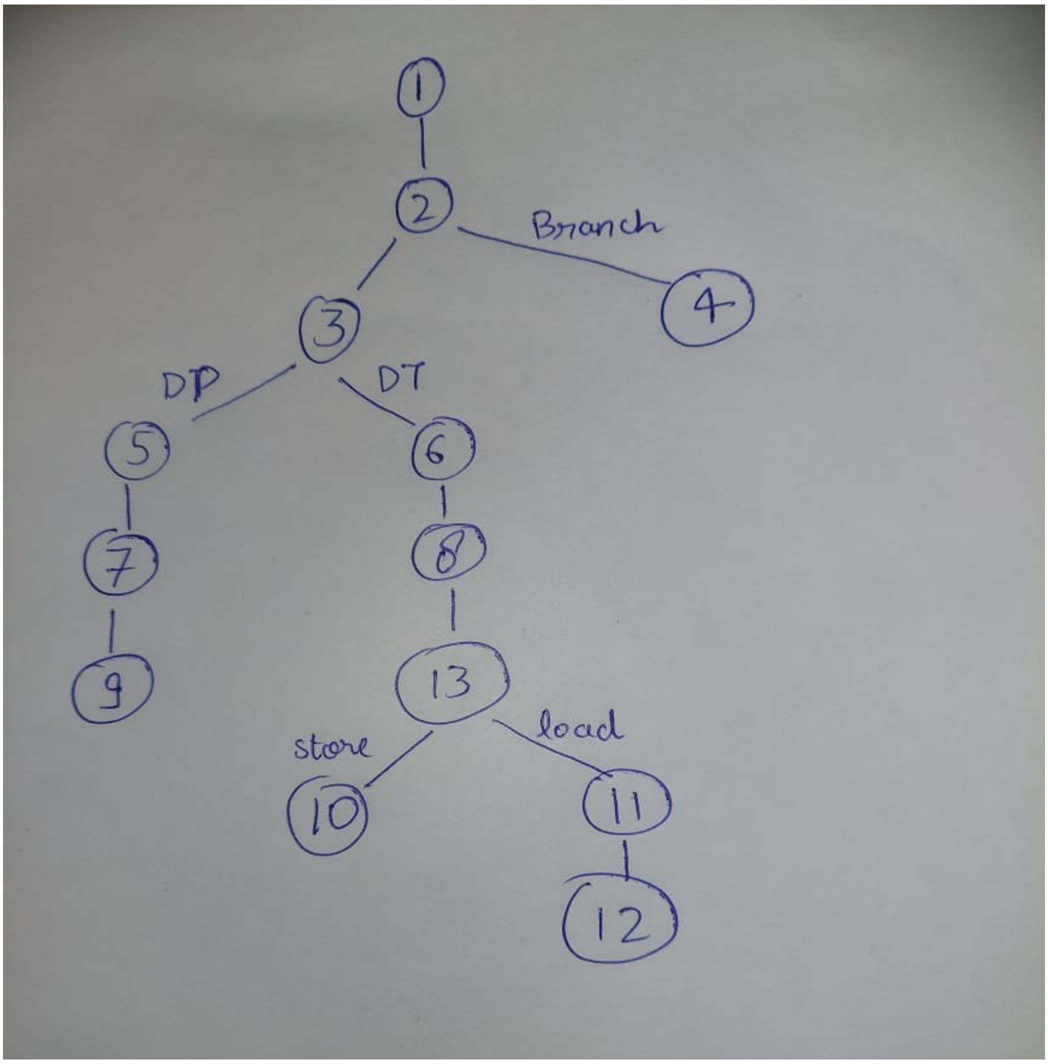


Image of my FSM

States:

1: IR = Mem[PC], PC = PC+4
2: A = RF[IR[19-16]], B = RF[IR[3-0]]
3: Read Rm (IR[3-0]) or Rs (IR[11-8]) and store in C.
4: PC = PC + S2(IR[23-0])+4 depending on predicate.
5: Shift for DP instruction.
6: Shift for DT instruction.
7: Res,Flags = ALU(A,B,C,IR[24-21])
8: Res = A ± ex(IR[11-0])
9: RF[IR[15-12]] = Res
10: Mem[Res] = B depending on predicate and Pmconnect for store.
11: DR = Mem[Res]
12: RF[IR[15-12]] = DR depending on predicate and Pmconnect for store.
13: Write back in Rn if needed.

Test done:

I have majorly tested the commands related to this stage only as my code for previous stages was working fine.

ARMSim code:

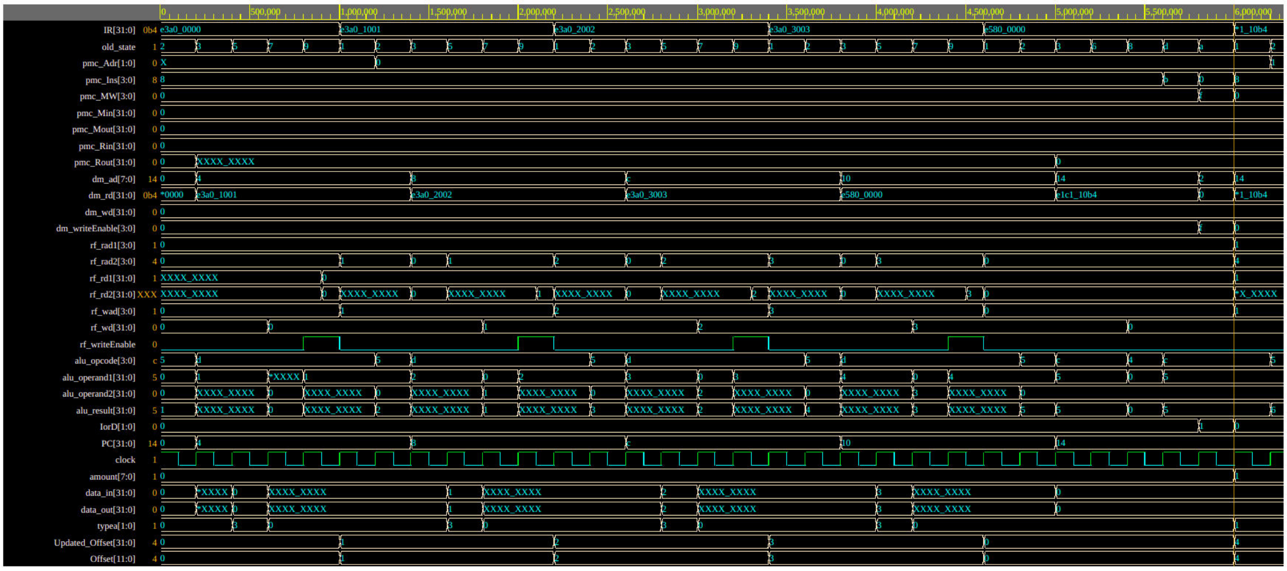
```
mov r0, #0
mov r1, #1
mov r2, #2
mov r3, #3
str r0, [r0]
strh r1, [r1, #4]
strb r2, [r2, r4]
str r1, [r1]
str r2, [r2]
ldr r4, [r0]
ldrh r5, [r1]
ldrsh r6, [r2]
ldrb r7, [r3]
ldrsb r8, [r0]
ldrh r9, [r0, #5]
ldrh r9, [r0, r1]
```

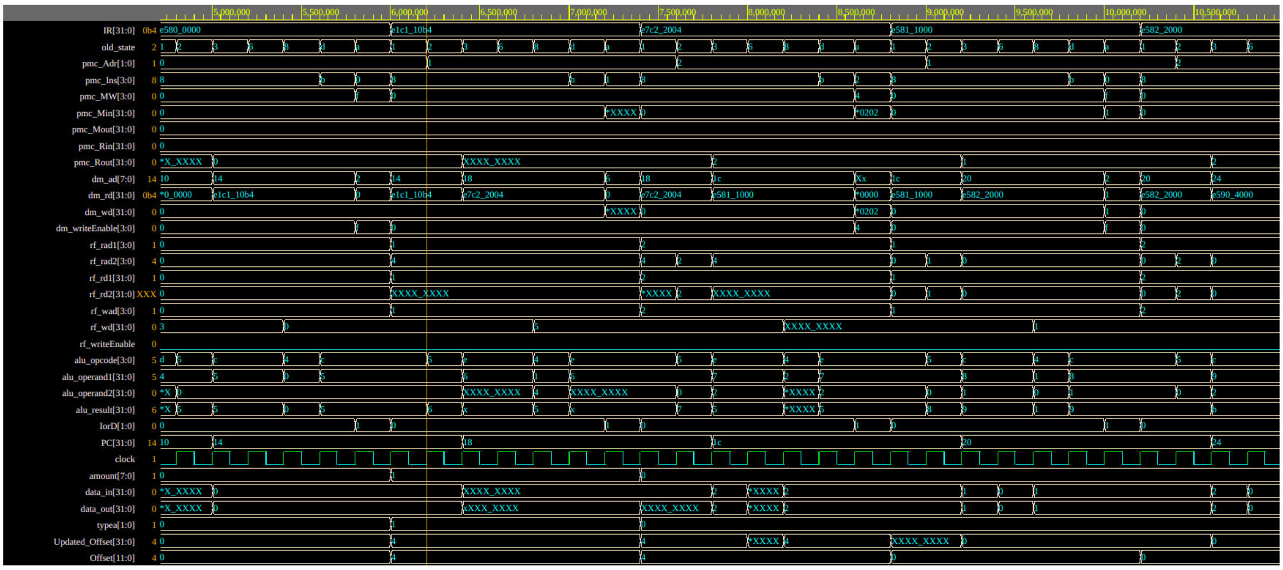
Instruction set for Data Memory:

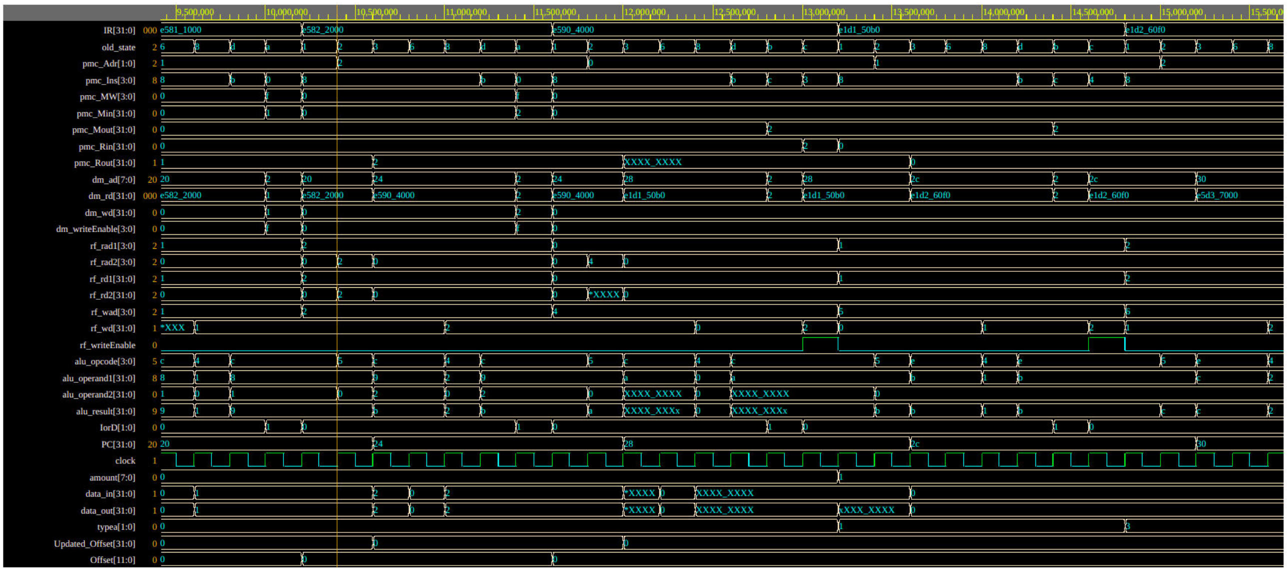
```
(       0 => X"E3A00000",
        1 => X"E3A01001",
        2 => X"E3A02002",
        3 => X"E3A03003",
        4 => X"E5800000",
        5 => X"E1C110B4",
        6 => X"E7C22004",
        7 => X"E5811000",
        8 => X"E5822000",
        9 => X"E5904000",
        10 => X"E1D150B0",
        11 => X"E1D260F0",
        12 => X"E5D37000",
        13 => X"E1D080D0",
        14 => X"E1D090B5",
        15 => X"E19090B1",
        others => X"00000000" );
```

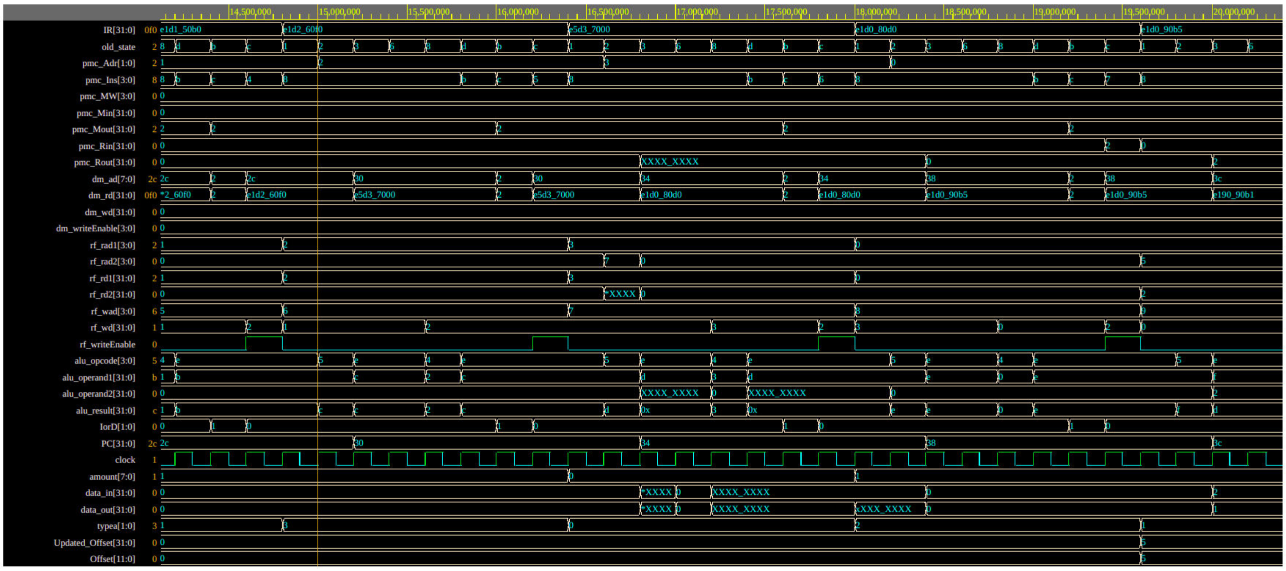
EPWave:

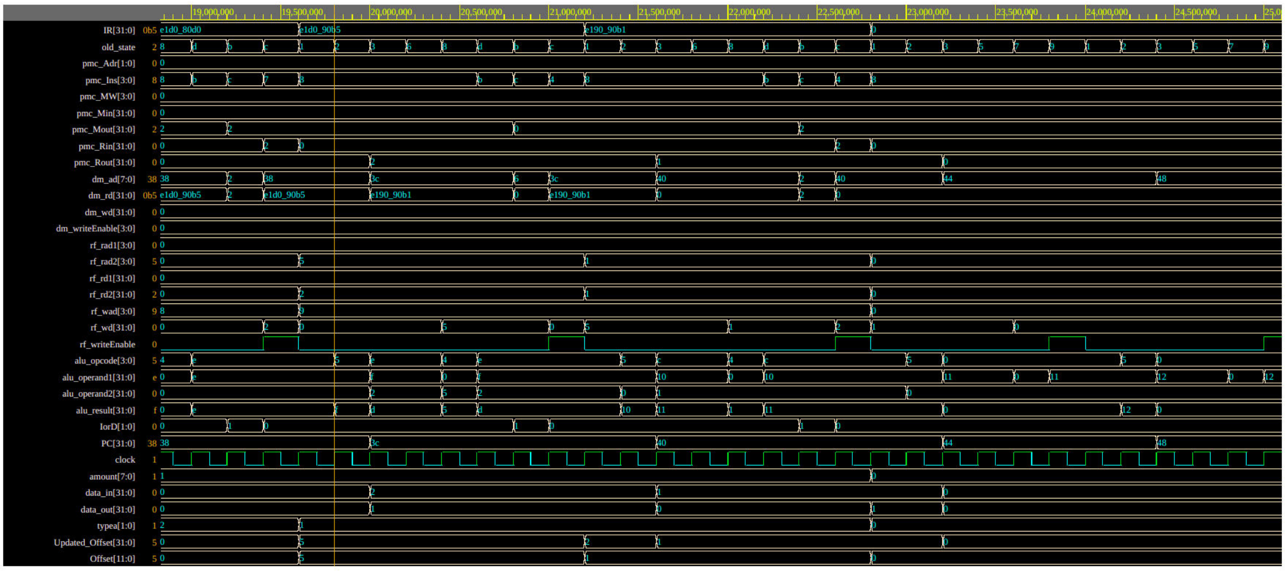
- The images are on the right side of the previous image
- Only important signals are shown.








14,500,000
15,000,000
15,500,000
16,000,000
16,500,000
17,000,000
17,500,000
18,000,000
18,500,000
19,000,000
19,500,000
20,000,000
IR[31:0]
old_state
pmc_Adr[1:0]
pmc_Ins[3:0]
pmc_MW[3:0]
pmc_Min[31:0]
pmc_Mout[31:0]
pmc_Rin[31:0]
pmc_Rout[31:0]
dm_ad[7:0]
dm_rd[31:0]
dm_wd[31:0]
dm_writeEnable[3:0]
rf_rad1[3:0]
rf_rad2[3:0]
rf_rd1[31:0]
rf_rd2[31:0]
rf_wad[3:0]
rf_wd[31:0]
rf_writeEnable
alu_opcode[3:0]
alu_operand1[31:0]
alu_operand2[31:0]
alu_result[31:0]
IorD[1:0]
PC[31:0]
clock
amount[7:0]
data_in[31:0]
data_out[31:0]
typeal[1:0]
Updated_Offset[31:0]
Offset[11:0]
e1d1_50b0
e1d2_60f0
e5d3_7000
e1d0_80d0
e1d0_90b5
e1d0_90b5
e190_90b1
XXXX_XXXX


19,000,000
19,500,000
20,000,000
20,500,000
21,000,000
21,500,000
22,000,000
22,500,000
23,000,000
23,500,000
24,000,000
24,500,000
IR[31:0]
old_state
pmc_Adr[1:0]
pmc_Ins[3:0]
pmc_MW[3:0]
pmc_Min[31:0]
pmc_Mout[31:0]
pmc_Rin[31:0]
pmc_Rout[31:0]
dm_ad[7:0]
dm_rd[31:0]
dm_wd[31:0]
dm_writeEnable[3:0]
rf_rad1[3:0]
rf_rad2[3:0]
rf_rd1[31:0]
rf_rd2[31:0]
rf_wad[3:0]
rf_wd[31:0]
rf_writeEnable
alu_opcode[3:0]
alu_operand1[31:0]
alu_operand2[31:0]
alu_result[31:0]
IorD[1:0]
PC[31:0]
clock
amount[7:0]
data_in[31:0]
data_out[31:0]
typeal[1:0]
Updated_Offset[31:0]
Offset[11:0]
e1d0_80d0
e1d0_90b5
e190_90b1