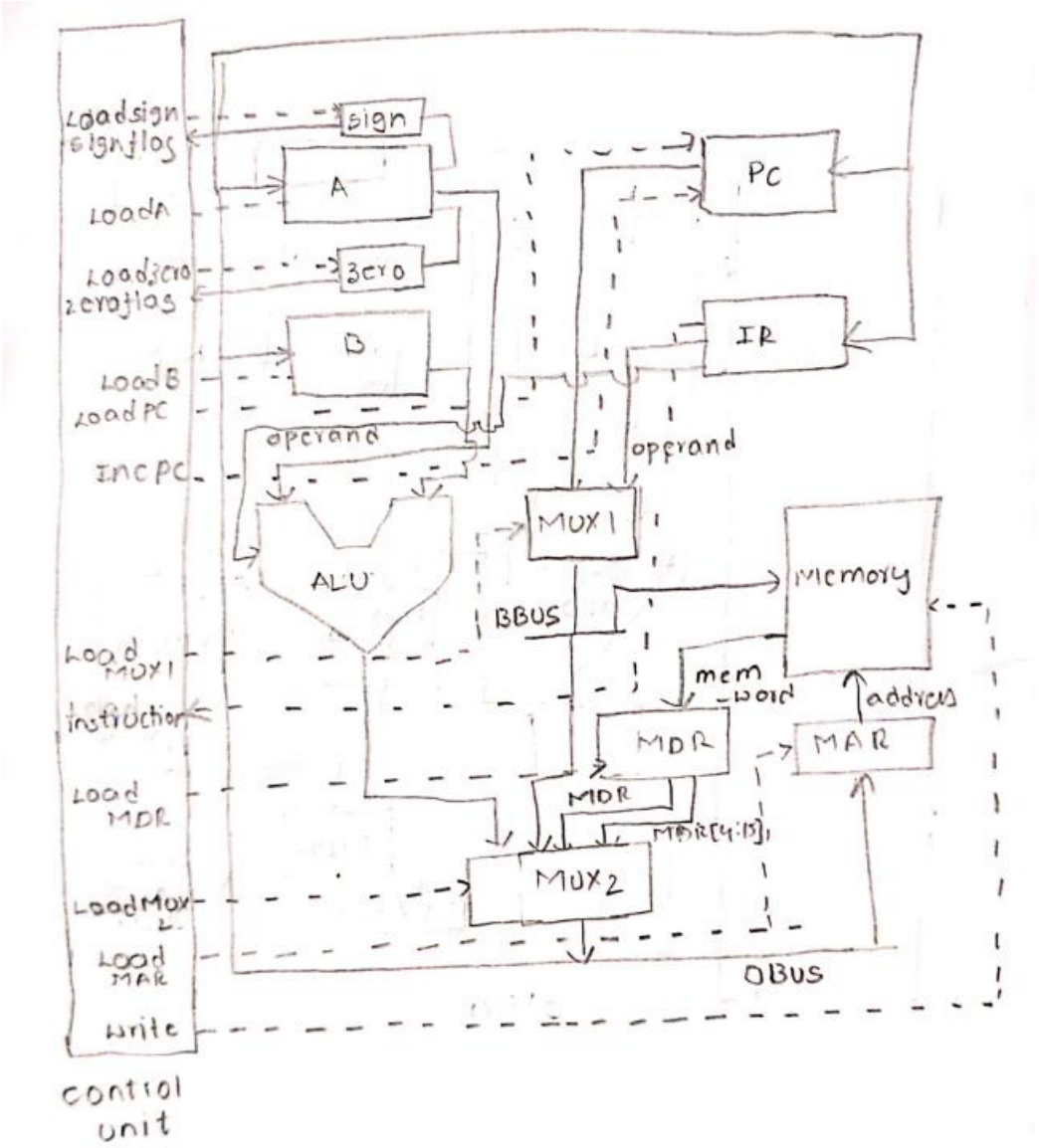
Vesp1.0:

Architecture 1:



The above architecture contains A and B registers. The zero and signed registers are used for raising the flags when necessary for jump instructions. The two mux are used for the selection according to the instruction data path.
Mux1: It is used for the selection of ALU, MDR, MDR[4:15], MAR
Mux2: It is used for the selection of PC, IR[4:15]
I have used the 16 bit buses for both 16 bit and 12 bit data transfers.

S_idle: State entered after reset is asserted. No action

Fetch stage:

S_fet1: This cycle is used for loading the PC contents into the MAR register.
S_fet2: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register.
S_fet3: This cycle is used for Loading the Instruction register with the contents of MDR register and Incrementing the PC value.

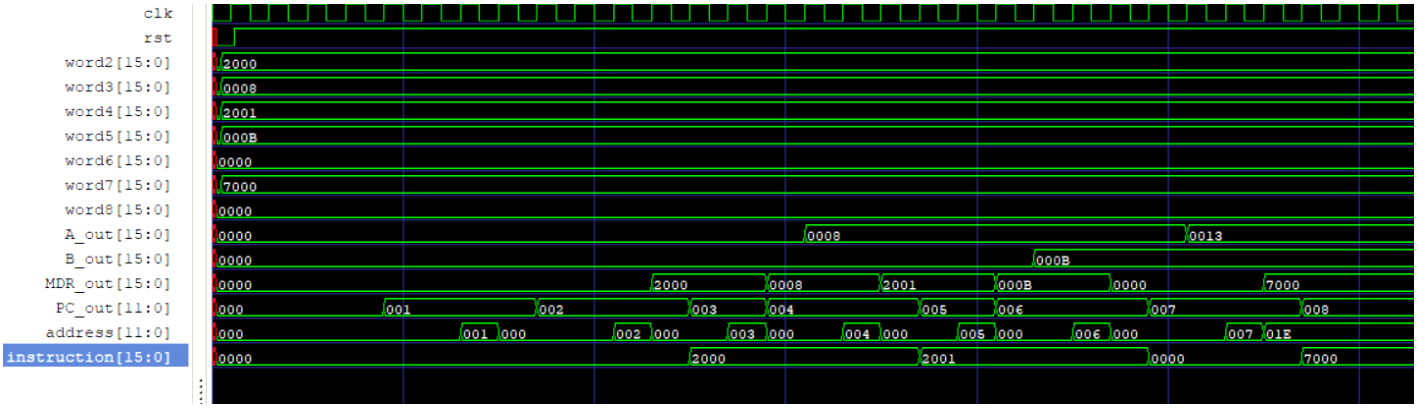
Decode:

S_dec: This cycle operates based on the opcode and decodes what type of instruction is to be executed. If the instruction contains ALU operations then appropriate register is selected according to the instruction execution. These ALU operations do not require another execution cycle since the execution can be completed in decode stage. If it is LDA, MOV instruction then the already incremented PC value is selected and loaded in MAR. JMP instructions do not require another execution cycle as these instructions can be implemented in the decode stage itself.

Execute:

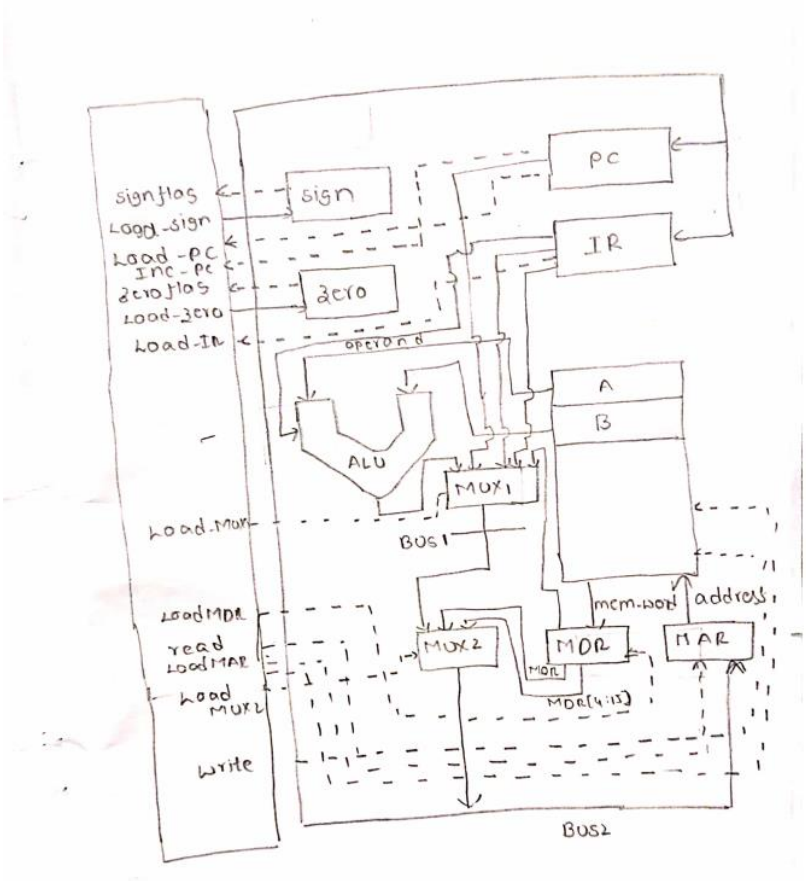
S_rd1: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register and incrementing the PC value.
S_rd2: This cycle is used for loading the MDR value in the appropriate registers.
S_mv1: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register and incrementing the PC value.
S_mv2: This cycle is used for loading the MDR[4:15] into the MAR register.
S_mv3: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register.
S_mv4: This cycle is used for loading the MDR value to the appropriate operand value{M[IR[4:15]] = M[M[MAR+1][4:15]]}.

S_halt: Default state to trap failure to decode a valid instruction.



The above test bench shows the vcd file of the implemented test code. These are labels of the six registers shown. The memory values are provided from the memory location 2.
A: A_out, B: B_out, MDR_out: MDR, PC_out: PC, address: MAR, instruction: IR

Architecture 2:



In this architecture. I have considered both A and B in the memory locations itself which has taken few more cycles for fetch and execution than the previous architecture.

MUX1 has 5 inputs in total: ALU, PC, IR, IR[4:15], MDR,
MUX2 has 3 inputs in total: Bus1, MDR, MDR[4:15].

I have selected 16-bit buses for both 16 bit and 12-bit data transfer.

S_idle: State entered after reset is asserted. No action

Fetch Stage:

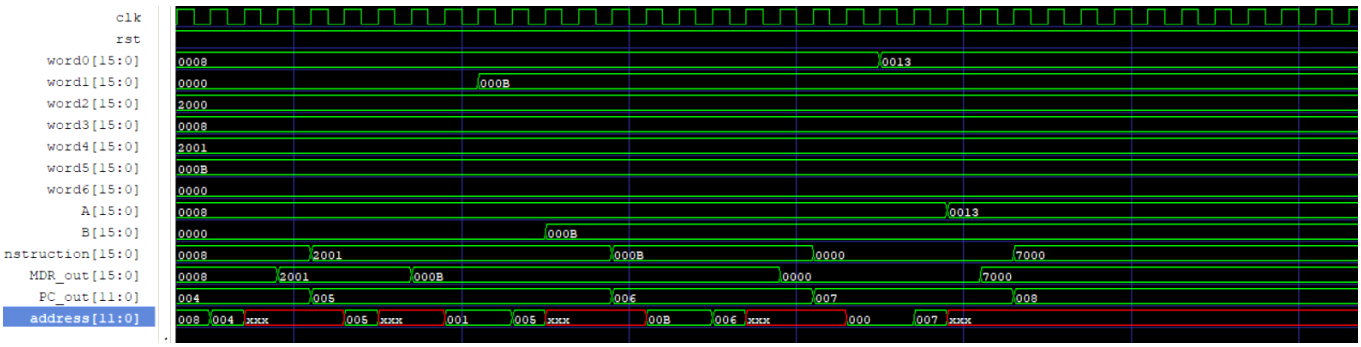
S_fet1: This cycle is used for loading the PC contents into the MAR register.
S_fet2: This cycle is used for reading the contents of the memory location from the MAR register.
S_fet3: This cycle is used for loading the contents into the MDR register from the memory location.
S_fet4: This cycle is used for loading the IR register with the contents of MDR register and incrementing the PC value.

Decode Stage:
This cycle operates based on the opcode and decodes what type of instruction is to be executed. If it contains the ALU instructions, then we need to load the appropriate register address into the MAR register. If it is LDA, MOV instruction then the already incremented PC value is selected and loaded in MAR. JMP instructions do not require another execution cycle as these instructions can be implemented in the decode stage itself.

Execution Stage:

S_exc1: This cycle is used for loading the ALU output by raising the write signal.
S_rd1: This cycle is used is used for reading the memory location value addressed by MAR register.
S_rd2: This cycle is used for loading the contents of the memory value into the MDR register.
S_rd3: This cycle is loading the operand value IR[4:15] into the MAR register.
S_rd4: This cycle is used for writing mem[MAR+1] into the IR[4:15] memory location.
S_mv1: This cycle is used is used for reading the memory location value addressed by MAR register.
S_mv2: This cycle is used for loading the contents of the memory value into the MDR register.
S_mv3: This cycle is used for loading MDR[4:15] into MAR register.
S_mv4: This cycle is used for reading MDR[4:15] address data.
S_mv5: This cycle is used for loading the mem[mem[MAR+1][4:15]] into MDR register.
S_mv6: This cycle is used for loading IR[4:15] value into MAR register.
S_mv7: This cycle is used for writing the mem[mem[MAR+1][4:15]] value into mem[IR[4:15]] .

S_halt: Default state to trap failure to decode a valid instruction.

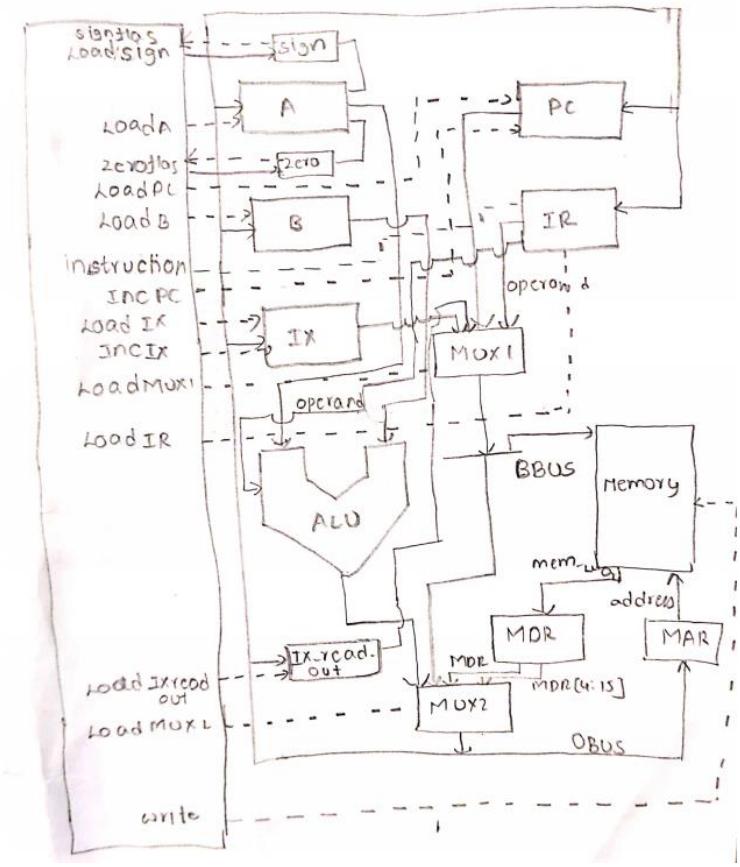


The above VCD file shows the Vesp1.0 architecture 2. The code is written from memory location 2 which is word 2 in here. Word0 and word1 are A, B registers.

Instruction: IR register. MDR-out: MDR register, A: A register, B: B register, PC_out – PC register, address: MAR register.

Vesp2.0:

Architecture 1:



The above architecture contains A and B registers for ALU operations. IX register is an index register. I have used another register IX_read out in order to reduce the number of cycles by storing the index value in a specified location.

I have used 16-bit buses through which you can transfer 12 bit data as well.

Two Mux have been used:
MUX1: ALU, PC, IR, IR[4:15], MDR
MUX2: Bus1, MDR, MDR[4:15]

S_idle: State entered after reset is asserted. No action

Fetch Stage:

S_fet1: This cycle is used for loading the PC contents into the MAR register.
S_fet2: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register.
S_fet3: This cycle is used for Loading the Instruction register with the contents of MDR register and Incrementing the PC value.

Decode Stage:

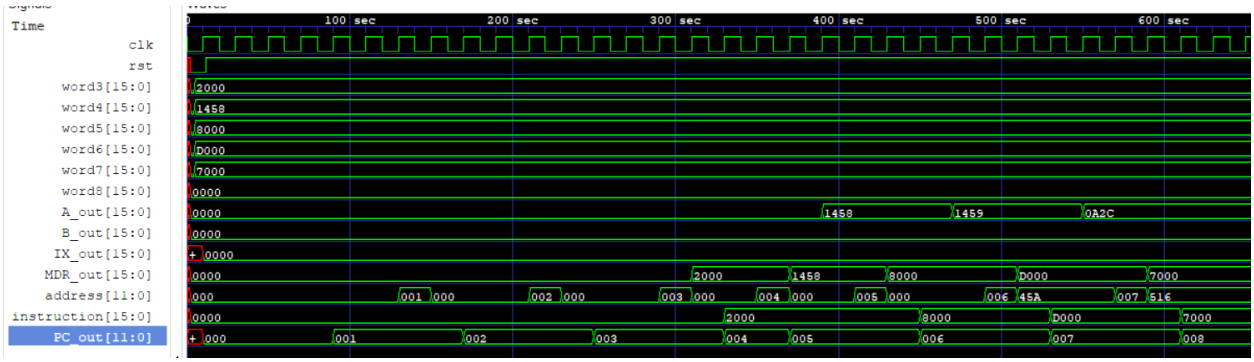
S_dec: This cycle operates based on the opcode and decodes what type of instruction is to be executed. If the instruction contains ALU operations then the ALU is selected with the A register. These ALU operations do not require another execution cycle since the execution can be completed in decode stage. If it is LDA, MOV instruction then the already incremented PC value is selected and loaded in MAR. JMP instructions do not require another execution cycle as these instructions can be implemented in the decode stage itself. For MXF operation the index register value is loaded in MAR and for MXT register IR[4:15] value is loaded in MAR.

Execute Stage:

S_rd1: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register and incrementing the PC value.
S_rd2: This cycle is used for loading the MDR value in the appropriate registers.
S_mv1: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register and incrementing the PC value.
S_mv2: This cycle is used for loading the MDR[4:15] into the MAR register.
S_mv3: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register.
S_mv4: This cycle is used for loading the MDR value to the appropriate registers.
S_mxf1: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register.
S_mxf2: This cycle is used for loading the contents of MDR register to IX_read register.
S_mxf3: This cycle is used for loading the IR[4:15] to MAR register.

S_mxf4: This cycle is used for incrementing the IX register as well writing the M[IX] to M[IR[4:15]
S_mxt1: This cycle is used for loading the contents of the memory addressed by MAR register to MDR register.
S_mxt2: This cycle is used for incrementing the IX register as well as writing M[IR[4:15]] to M[IX].

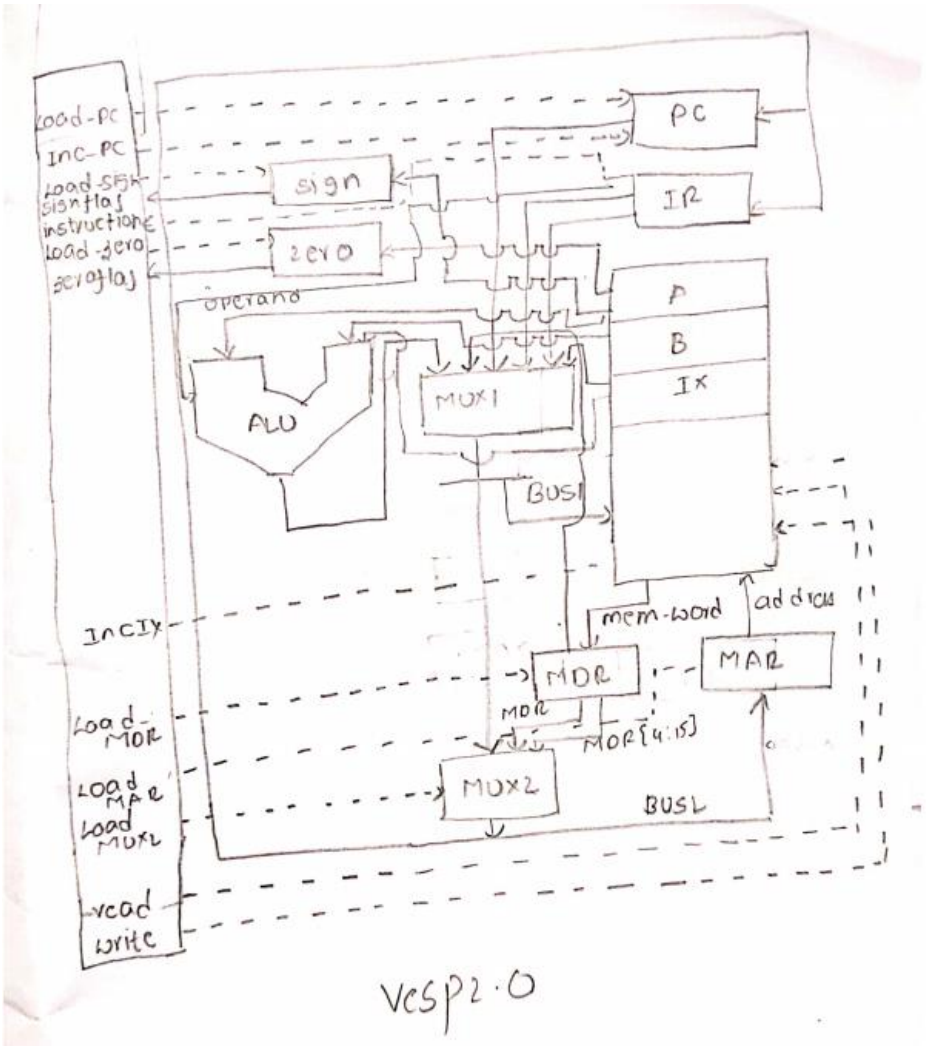
S_halt: Default state to trap failure to decode a valid instruction.



The above VCD file shows the implemented output. The program was written from memory location 3.

PC_out: PC register, A_out: A register, MDR_out: MDR register, address: MAR register, B_out: B register, instruction: IR register.

Architecture 2:



The above architecture contains two muxes.

MUX1: ALU, PC, IR, IR[4:15], MDR, IX
MUX2: Bus1, MDR, MDR[4:15]

I have used read and write control signals to control the output. Zero and signed which are used as flags for the jump instructions.

S_idle: State entered after reset is asserted. No action

Fetch Stage:

S_fet1: This cycle is used for loading the PC contents into the MAR register.
S_fet2: This cycle is used for reading the contents of the memory location from the MAR register.
S_fet3: This cycle is used for loading the contents into the MDR register from the memory location.
S_fet4: This cycle is used for loading the IR register with the contents of MDR register and incrementing the PC value.

Decode Stage:

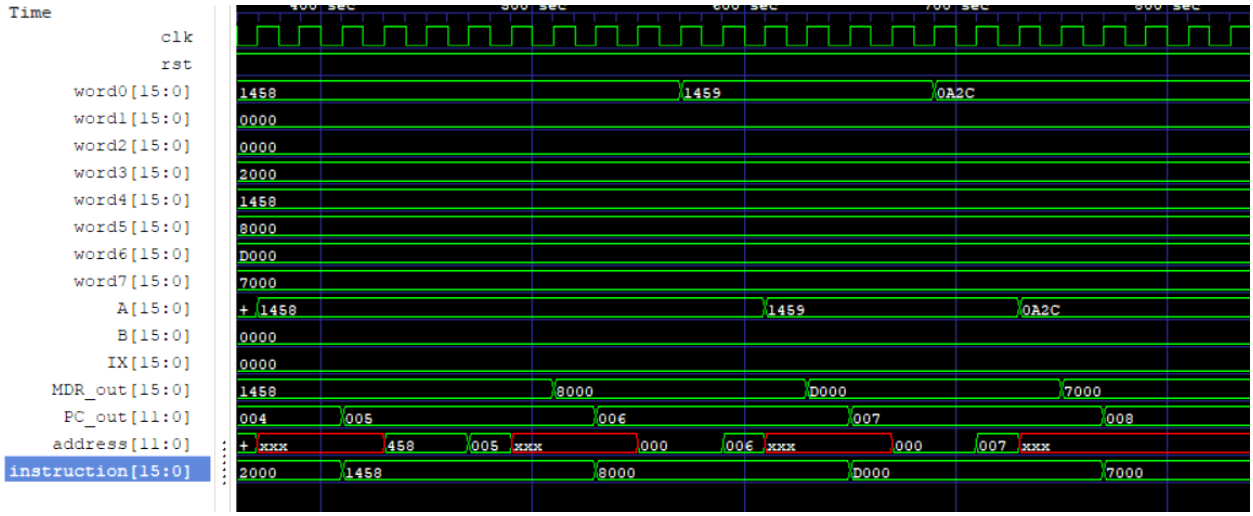
This cycle operates based on the opcode, if it contains the ALU instructions then we need to load the appropriate register address into the MAR register. If it is LDA, MOV instruction then the already incremented PC value is selected and loaded in MAR. JMP instructions do not require another execution cycle as these instructions can be implemented in the decode stage itself. In MXF instruction IX value is loaded into MAR register. In MXT instruction IR[4:15] value is loaded into MAR register.

Execute Stage:

S_exc1: This cycle is used for loading the ALU output by raising the write signal.
S_rd1: This cycle is used is used for reading the memory location value addressed by MAR register.
S_rd2: This cycle is used for loading the contents of the memory value into the MDR register.
S_rd3: This cycle is loading the operand value IR[4:15] into the MAR register.
S_rd4: This cycle is used for writing mem[MAR+1] into the IR[4:15] memory location.
S_mv1: This cycle is used is used for reading the memory location value addressed by MAR register.
S_mv2: This cycle is used for loading the contents of the memory value into the MDR register.
S_mv3: This cycle is used for loading MDR[4:15] into MAR register.
S_mv4: This cycle is used for reading MDR[4:15] address data.
S_mv5: This cycle is used for loading the mem[mem[MAR+1][4:15]] into MDR register.
S_mv6: This cycle is used for loading IR[4:15] value into MAR register.
S_mv7: This cycle is used for writing the mem[mem[MAR+1][4:15]] value into mem[IR[4:15]] .
S_mxf1: This cycle is used for reading the MAR register.
S_mxf2: This cycle is used for loading the contents of the memory value into the MDR register.
S_mxf3: This cycle is used for loading IR[4:15] into MAR register.
S_mxf4: This cycle is used for writing mem[IX] to mem[IR[4:15]]
S_mxf5: This cycle is used for incrementing the IX register.
S_mxt1: This cycle is used is used for reading the memory location value addressed by MAR register.
S_mxt2: This cycle is used for loading the contents of the memory value into the MDR register.
S_mxt3: This cycle is used for loading IX value to MAR register.

S_mxt4: This cycle is used for writing the mem[IR[4:15]] to mem[IX].
S_mxt5: This cycle is used for incrementing the IX value.

S_halt: Default state to trap failure to decode a valid instruction.



The above VCD demonstrates the implemented output.

Word0 , Word1, Word2 are same are A, B, IX registers.

MDR_out: MDR, PC_out: PC, address: MAR, IX: IX register, instruction: IR register.