EECS 112L Lab2 Report

Jay Patel
ID: 77742251

Professor Pooria M. Yaghini

March 18, 2018

1 Block Diagram of the design Implementation

In this Section, I aim to describe my design of a single cycle processor.In addition, I present a brief description on how each Instruction type is implemented and supported by the processor.

The original processor only allowed execution of R-type and I-type(Load,ADDI) instructions.

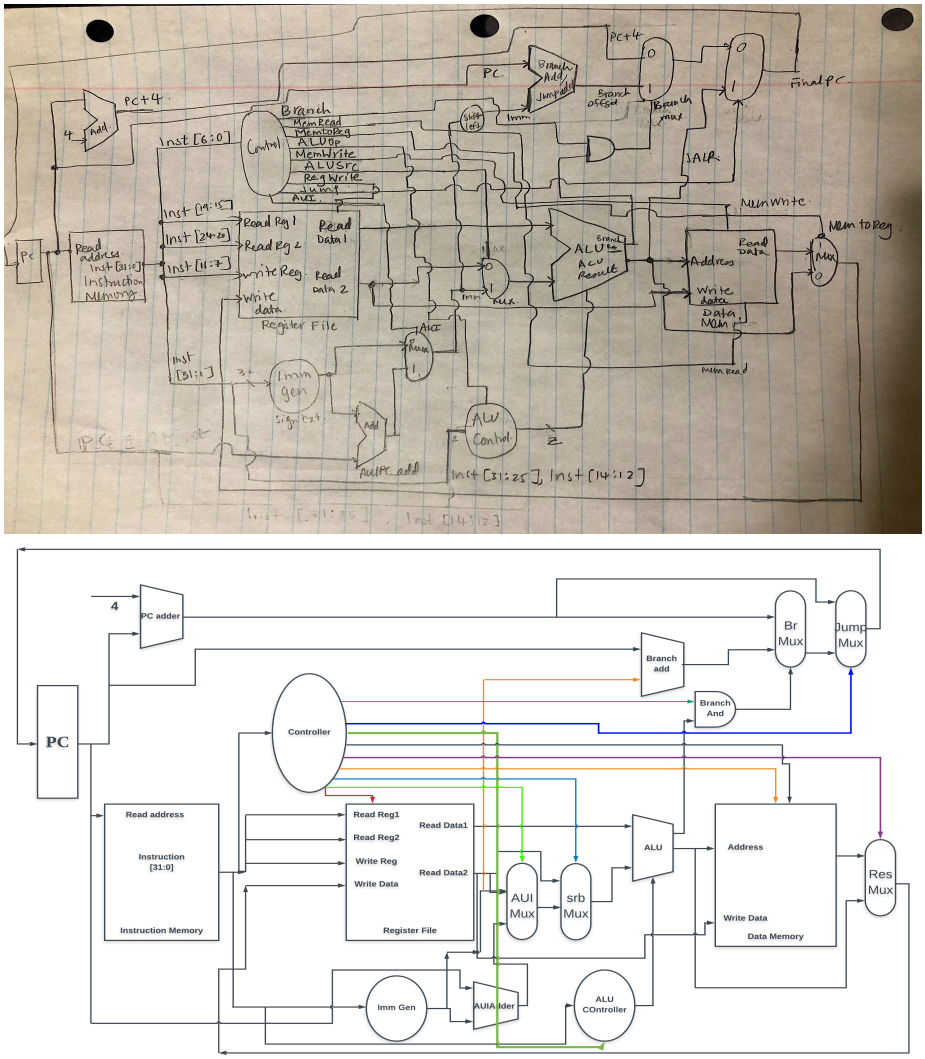

Figure 1: Complete Design of Single Cycle processor

In addition to the original instructions supported in Lab 1, this processor is

aimed to execute the following instruction types:

- Integer Register-Register Instructions(R-type)
- Integer Register-Immediate Instructions(RI-type)
- Load Instructions
- Store Instructions
- Conditional Branch Instructions
- Jump(JAL, JALR) Instructions
- Load Upper Immediate(LUI), Add Upper Immediate to PC(AUIPC)

1.1 Integer Register-Register Instructions(R-type)

31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	src2	src1	ADD/SLT/SLTU	dest	OP
0000000	src2	src1	AND/OR/XOR	dest	OP
0000000	src2	src1	SLL/SRL	dest	OP
0100000	src2	src1	SUB/SRA	dest	OP

Figure 2: Snippet of R-type instructions

For R-type, the register file module provides the two register source operands.The first operand, RS1 is wired directly to Input 1 of ALU.RS2 is connected to a 2-1 multiplexer. The control signal ALUsrc will be 0 for this mux to pass RS2 operand as the output for Input 2 of ALU. Meanwhile, using funct3 operand (Inst[14:12]) and funct7 operand (Inst[31:25]),ALUcontroller will set ALU operation bits according to various instructions like ADD, AND, OR, SLL, etc. Then the ALU will execute the instruction based on operation and turn on necessary flags to send the result to the destination register.

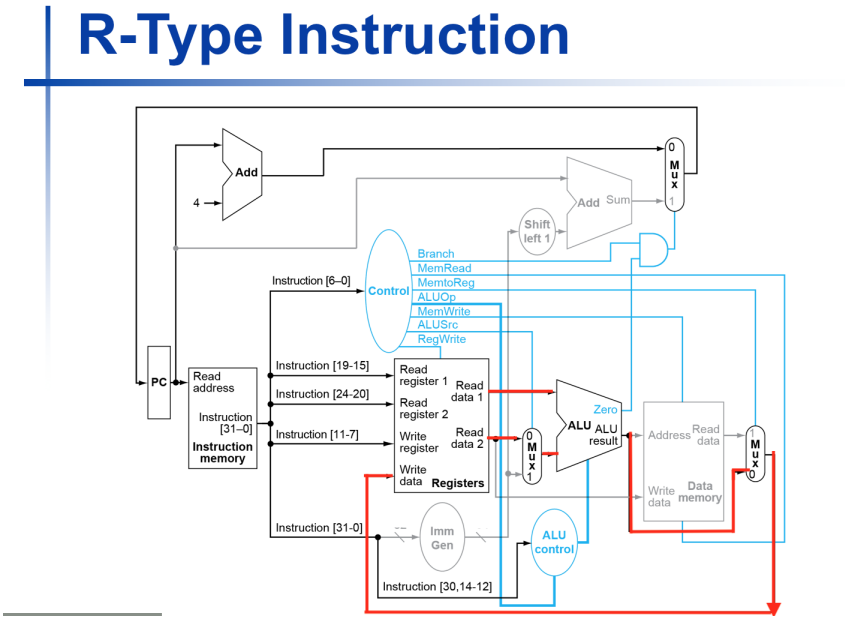

Figure 3: R-type datapath

1.2 Integer Register-Immediate Instructions(RI-type)

31 — 20	19 — 15	14 — 12	11 — 7	6 — 0
imm[11:0]	rs1	funct3	rd	opcode
12	5	3	5	7
I-immediate[11:0]	src	ADDI/SLTI[U]	dest	OP-IMM
I-immediate[11:0]	src	ANDI/ORI/XORI	dest	OP-IMM

31 — 25	24 — 20	19 — 15	14 — 12	11 — 7	6 — 0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM

Figure 4: Snippet of RI-type instructions

For RI-type, the register file module provides the a register source operand and an immediate field of 12 bits.The first operand, RS1 is wired directly to Input 1 of ALU. The immediate value is connected to a module called ImmGen, where the immediate value is sign extended to 32 bit value. The output of ImmGen is connected to a 2-1 multiplexer. The control signal ALUsrc will be 1 for this mux to pass the immediate value as the second source operand for Input 2 of ALU. Meanwhile, using funct3 operand (Inst[14:12]) and funct7 operand (Inst[31:25]),ALUcontroller will set ALU operation bits according to various instructions like ADDI, ANDI, SLLI, etc. Then the ALU will execute the instruction based on operation and turn on necessary flags to send the result to the destination register.

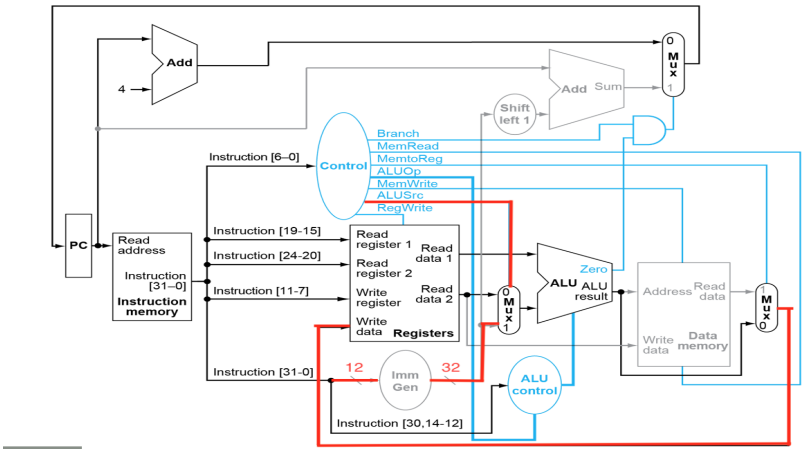

Figure 5: Snippet of RI-type instruction datapath

1.3 Load Instructions

imm[11:0]	rs1	000	rd	0000011	LB
imm[11:0]	rs1	001	rd	0000011	LH
imm[11:0]	rs1	010	rd	0000011	LW
imm[11:0]	rs1	100	rd	0000011	LBU
imm[11:0]	rs1	101	rd	0000011	LHU

Figure 6: Snippet of Load instructions

For Load, the immediate gen module will sign extend the immediate value and then a mux will select, based on the ALUSrc flag, second source operand as the immediate.Then the ALU will execute the instruction by adding immediate value to source operand 1. The Alu result is an address for the data memory, where the value at the address corresponding to load-type(LB, LH, LW, LBU,or, LHU) is written into the destination register.

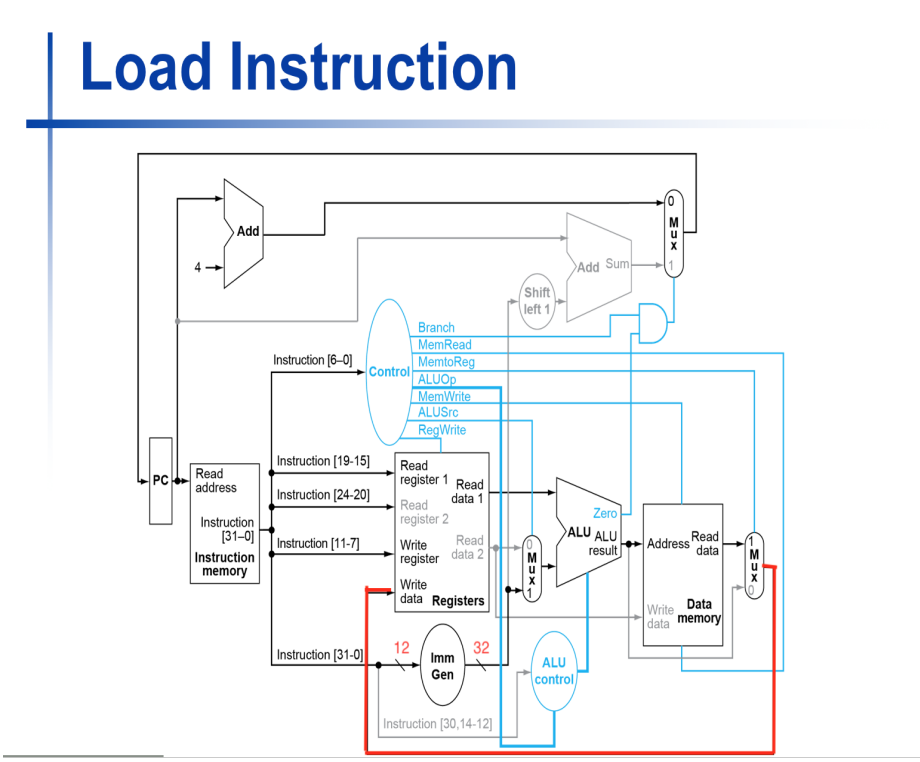

Figure 7: Load Datapath

1.4 Store Instructions

imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW

Figure 8: Snippet of Store instructions

For store, the immediate gen module will sign extend the immediate value and then a mux will select, based on the ALUSrc flag, second source operand as the immediate.Then the ALU will execute the instruction by adding immediate value to source operand 1. The Alu result is an address for the data memory, where the value at the address in the data memory will be overwritten with values corresponding to store-type(SB, SH, SW).

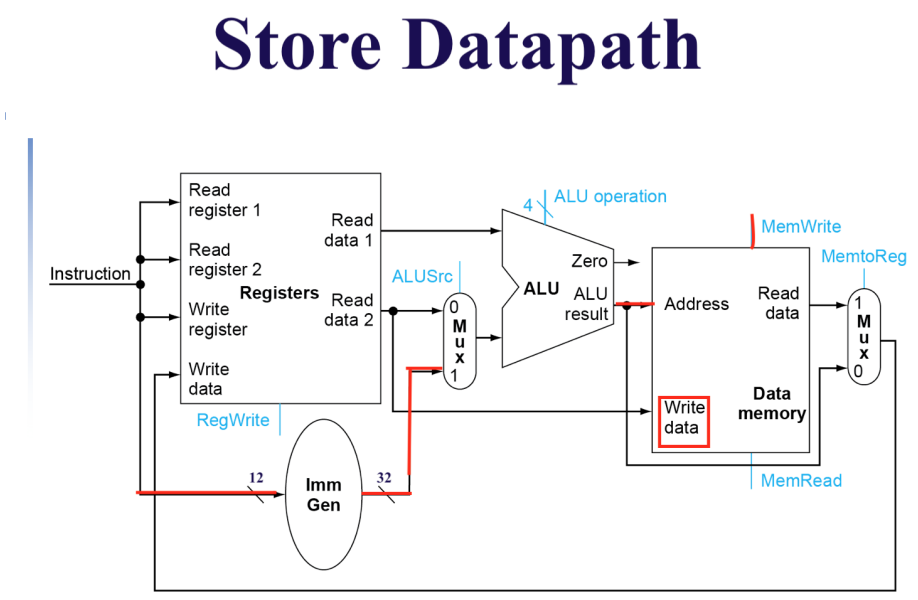

Figure 9: Store instructions Datapath

1.5 Conditional Branch Instructions

31	30 25	24 20	19 15	14 12	11 8	7	6 0
imm[12]	imm[10:5]	rs2	rs1	funct3	imm[4:1]	imm[11]	opcode
1	6	5	5	3	4	1	7
offset[12,10:5]		src2	src1	BEQ/BNE	offset[11,4:1]		BRANCH
offset[12,10:5]		src2	src1	BLT[U]	offset[11,4:1]		BRANCH
offset[12,10:5]		src2	src1	BGE[U]	offset[11,4:1]		BRANCH

Figure 10: Snippet of Branch instructions

For branch, the immediate gen module will sign extend the immediate and then a mux will select, based on the ALUSrc flag, second source operand as the immediate. Then the ALU will execute the instruction based on operation using Function 3 of the instruction. and turn on necessary flags to send the result a branch andgate, which will set the branch signal to 1 if both the branch control signal and the signal based on ALU result. If the branch signal is 1, the new Pc will be the branch offset + PC and if branch signal is 0, then the New Pc will be PC+ 4.

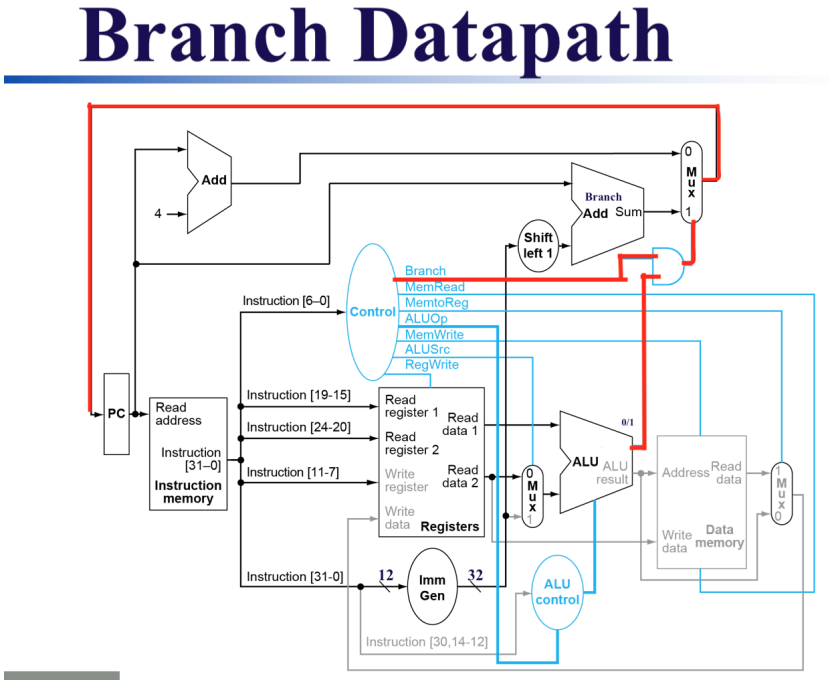

Figure 11: Branch instructions Datapath

1.6 Load Upper Immediate(LUI), Add Upper Immediate to PC(AUIPC)

31 — 12	11 — 7	6 — 0
imm[31:12]	rd	opcode
20	5	7
U-immediate[31:12]	dest	LUI
U-immediate[31:12]	dest	AUIPC

Figure 12: LUI/AUIPC instructions

For LUI, the immediate gen module will sign extend the immediate and then a mux will select the immediate value, based on the ALUSrc flag .Then, the ALU will execute the instruction based on operation for LUI, and so the upper 20 bits of the immediate value will be written back to the destination register.

For AUIPC, the immediate gen module will sign extend the immediate and then a mux will select the immediate value, based on the ALUSrc flag. Then, the immediate value will be added to the PC to get the new value. Then, the ALU will execute the instruction based on operation for AUIPC, and so the upper 20 bits of the value will be written back to the destination register.

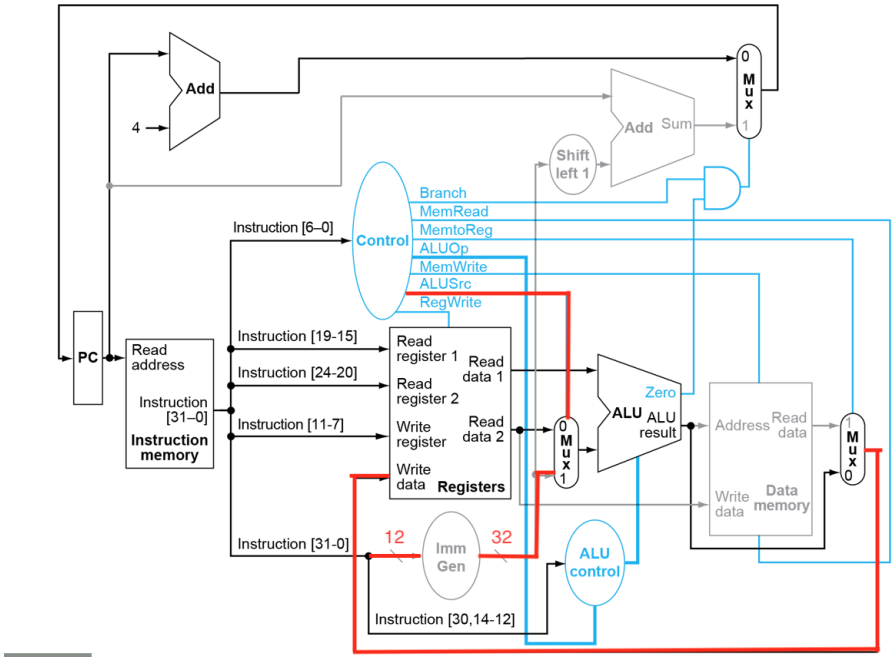

Figure 13: Utype instructions datapath

1.7 Jump(JAL, JALR) Instructions

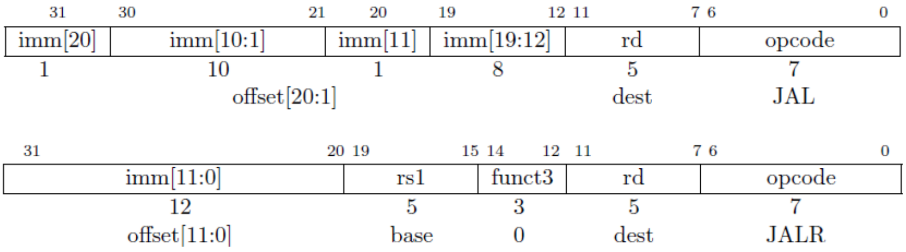

Figure 14: Snippet of Jump instructions

For JAL, the immediate gen module will sign extend the immediate value. Then, the immediate value is added to PC to get a new PC address. The PC will jump,based on the jump signal, to that address and will execute the instruction at that taget address. Meanwhile, for jump instruction, the ALU result will be the target address plus 4, which will be stored into the destination register.

For JALR, the immediate gen module will sign extend the immediate value. Then, the immediate value is added to Register one operand to get a new PC address. The PC will jump,based on the jump signal, to that taget address and will execute the instruction at that taget address. Meanwhile, for jump instruction, the ALU result will be the target address plus 4, which will be stored into the destination register.

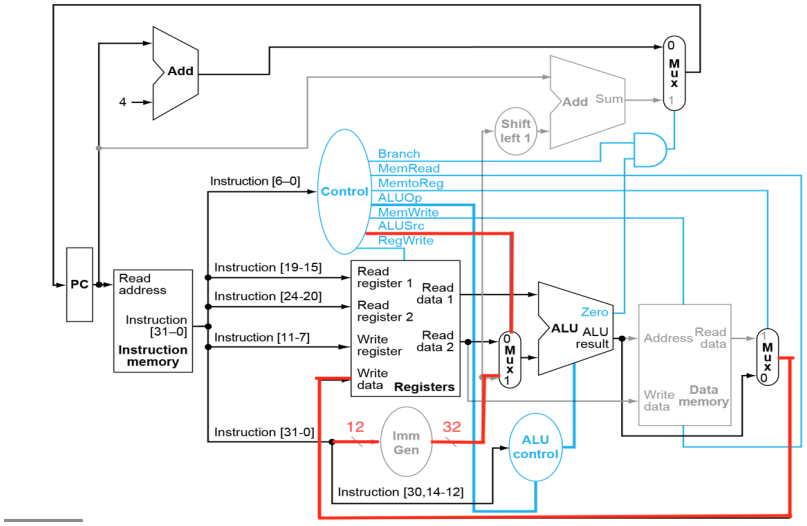

Figure 15: Jump instructions datapath

2 Simulation Waveforms

Here are the snippets simulation waveforms that include Clk, reset, and ALU-Result: The simulations ran without any warnings or errors.

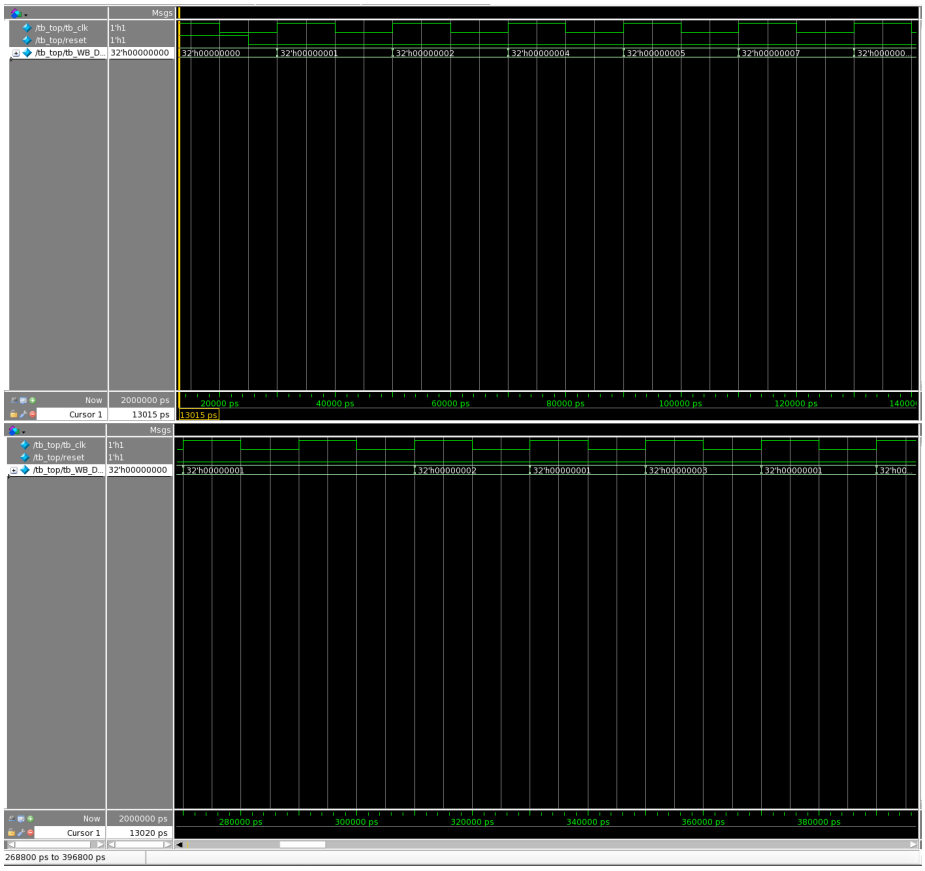

Figure 16: Simulation snippet

3 Synthesis Results

Report: The synthesis completed without any issues. Here is the synthesis report for the design implementation.

3.1 Timing:

```
Timing Path Group 'clk'
-----------------------------------
Levels of Logic:                5.00
Critical Path Length:           0.34
Critical Path Slack:            3.65
Critical Path Clk Period:       4.00
Total Negative Slack:           0.00
No. of Violating Paths:         0.00
Worst Hold Violation:           0.00
Total Hold Violation:           0.00
No. of Hold Violations:         0.00
-----------------------------------
```

Figure 17: Timing analysis

3.2 Cell Count:

```
Cell Count
-------------------------------------
Hierarchical Cell Count:          7
Hierarchical Port Count:        452
Leaf Cell Count:                574
Buf/Inv Cell Count:             115
Buf Cell Count:                   4
Inv Cell Count:                 111
CT Buf/Inv Cell Count:            0
Combinational Cell Count:       567
Sequential Cell Count:            7
Macro Count:                      0
-------------------------------------
```

Figure 18: Cell count analysis

3.3 Execution time:

```
Compile CPU Statistics
----------------------------------------
Resource Sharing:                  2.34
Logic Optimization:                0.32
Mapping Optimization:              2.06
----------------------------------------
Overall Compile Time:             18.37
Overall Compile Wall Clock Time:  99.80
```

Figure 19: Execution time analysis

3.4 Area:

```
Area
-----------------------------------
Combinational Area:     1210.233744
Noncombinational Area:    49.812225
Buf/Inv Area:            152.486401
Total Buffer Area:            10.17
Total Inverter Area:         142.32
Macro/Black Box Area:       0.000000
Net Area:                357.455119
-----------------------------------
Cell Area:              1260.045969
Design Area:            1617.501088

Number of ports:                          34
Number of nets:                          111
Number of cells:                          62
Number of combinational cells:            61
Number of sequential cells:                0
Number of macros/black boxes:              0
Number of buf/inv:                        15
Number of references:                     18

Combinational area:              1210.233744
Buf/Inv area:                     152.486401
Noncombinational area:             49.812225
Macro/Black Box area:               0.000000
Net Interconnect area:            357.455119

Total cell area:                 1260.045969
Total area:                      1617.501088
```

Figure 20: Area analysis

3.5 Power:

```
Global Operating Voltage = 1.05
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1pW
```

```
   dp/U14                          default                          0.0000
dp/U14  (Cell Total)                                  0.0000    14761.1621
-----------------------------------------------------------------------------
Totals (574 cells)                                   25.179uW    552.818uW
```

```
Operating Conditions: tt1p05vn40c   Library: saed32lvt_tt1p05vn40c
Wire Load Model Mode: enclosed

Design        Wire Load Model            Library
------------------------------------------------
riscv                 8000               saed32lvt_tt1p05vn40c
Datapath              8000               saed32lvt_tt1p05vn40c
flopr_WIDTH9          ForQA              saed32lvt_tt1p05vn40c
instructionmemory     8000               saed32lvt_tt1p05vn40c
mux2_WIDTH32_0        ForQA              saed32lvt_tt1p05vn40c
imm_Gen               ForQA              saed32lvt_tt1p05vn40c
_seo_mux_C8           8000               saed32lvt_tt1p05vn40c
mux2_WIDTH32_1        ForQA              saed32lvt_tt1p05vn40c


Global Operating Voltage = 1.05
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1pW


--------------------------------------------------------------------------------
                                  Switch   Int      Leak     Total
Hierarchy                         Power    Power    Power    Power    %
--------------------------------------------------------------------------------
riscv                               11.704   25.179 5.53e+08  589.701 100.0
  dp (Datapath)                     11.377   23.094 4.97e+08  531.080  90.1
    srcbmux (mux2_WIDTH32_1)         1.756    2.101 4.02e+07   44.037   7.5
    Ext_Imm (imm_Gen)                1.017    1.720 7.27e+07   75.431  12.8
    resmux (mux2_WIDTH32_0)          0.734    1.703 5.45e+07   56.939   9.7
    instr_mem (instructionmemory)    3.555    5.880 1.93e+08  202.660  34.4
      _seo_mux_C8 (_seo_mux_C8)      3.555    5.880 1.93e+08  202.660  34.4
    pcreg (flopr_WIDTH9)             0.480    9.825 5.40e+07   64.262  10.9
1
```

Figure 21: Area analysis

3.6 Conclusion

This Lab consisted of designing a single cycle processor. It serves well as an exposure to the real world applications. In concluding this lab, I have gained immense knowledge about the design process of processor and that will assist in future endeavors. Also, this lab sets up the groundwork for an optimized pipelined design for Lab3.