San Jose State University
Department of Computer Engineering

CMPE 140 Lab Report

Lab 5 Report

Title Processor Design (1): Design Code Review and Functional Verification

Semester Fall **Date** 03/10/2020

by

Name Thien N Hoang **SID** 012555673
(typed) *(typed)*

Name Phat Le **SID** 012067666
(typed) *(typed)*

Lab Checkup Record

Week	Performed By (signature)	Checked By (signature)	Tasks Successfully Completed*	Tasks Partially Completed*	Tasks Failed or Not Performed*
1					

*** Detailed descriptions must be given in the report.**

Authors

	Thien Hoang I was born and raised in Ho Chi Minh City, Viet Name. I came to the United States when I was 16 years old. Before transferring to SJSU, I lived in Houston Texas with my family. I am currently a 5th-year Computer Engineering major at San Jose State University. I have an interest in building the circuit and aspire to be a circuit designer in the future. I enjoy riding a motorcycle on the weekend. It helps me release my stress
	Phat Le I am an international student from Vietnam, also a transfer student from Seattle, and currently a senior Computer Engineering student at San Jose State University. My primary goal is to invent a device that will have huge improvements in humanity. By understanding the way computer software and hardware works, I believe my dream can become true one day. I also enjoy playing RPG console video games and singing alone with my guitar. Sounds very introvert right?!

Introduction

The purpose of this lab is to gain hands-on processor design experience by observing the given RTL Verilog design code of the single-cycle MIPS processor. Another task is to draw block diagrams of the processor as well as build the DUT and functionally verify its testbench based on the given sample program.

Design methodology

One of the first tasks is to download the Verilog archived files called *"lab5_processor_design_1"* from Canvas. Then, the next task is to draw block diagrams of the control unit microarchitecture, datapath microarchitecture, instruction memory, data memory, processor core, and complete processor code by observing their design codes as shown in *Figure 1, 2, 3* and *4*. Finally, the simulation testbench is built to functionally verify the design using the given assembly code as shown in *Table 1*.

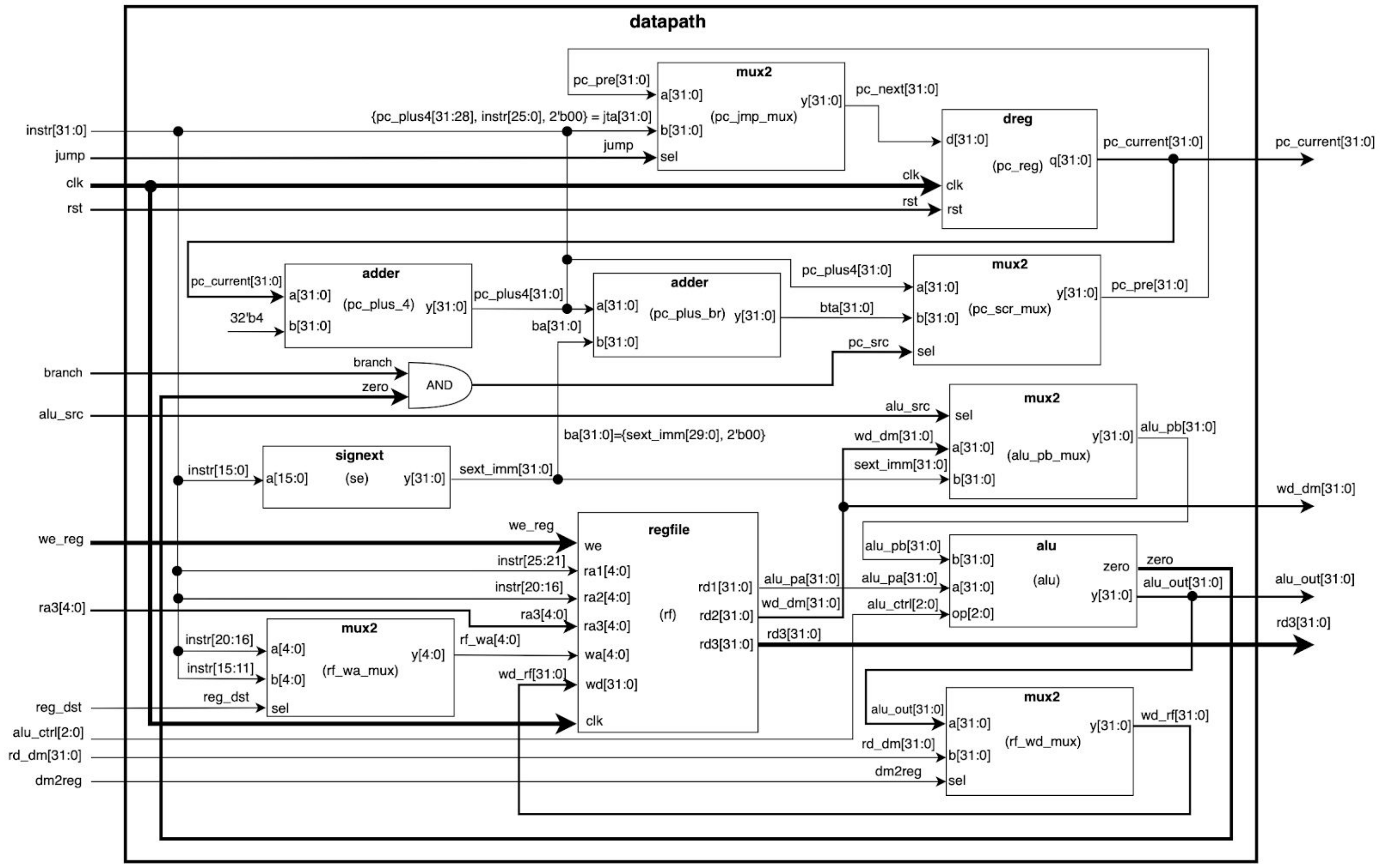


Figure 1: Datapath with microarchitecture details (without memories)

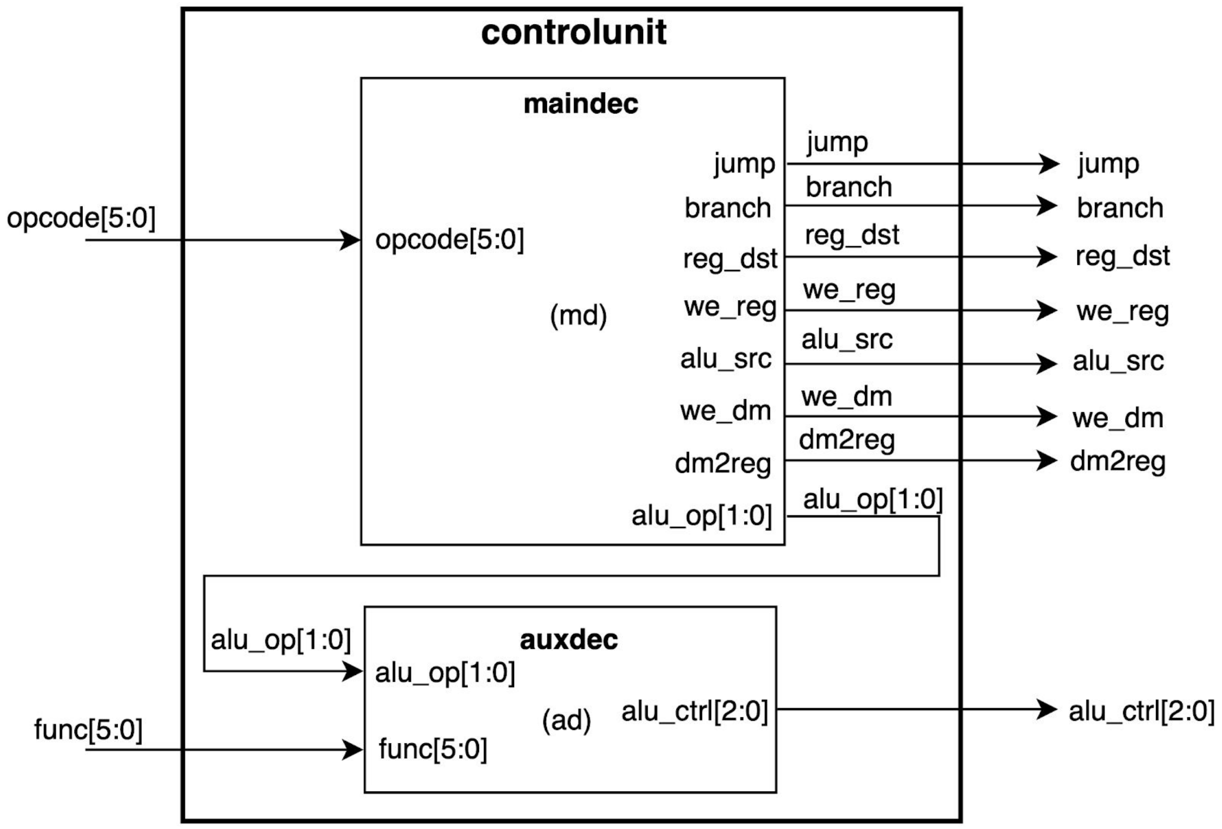


Figure 2: Control unit with microarchitecture details

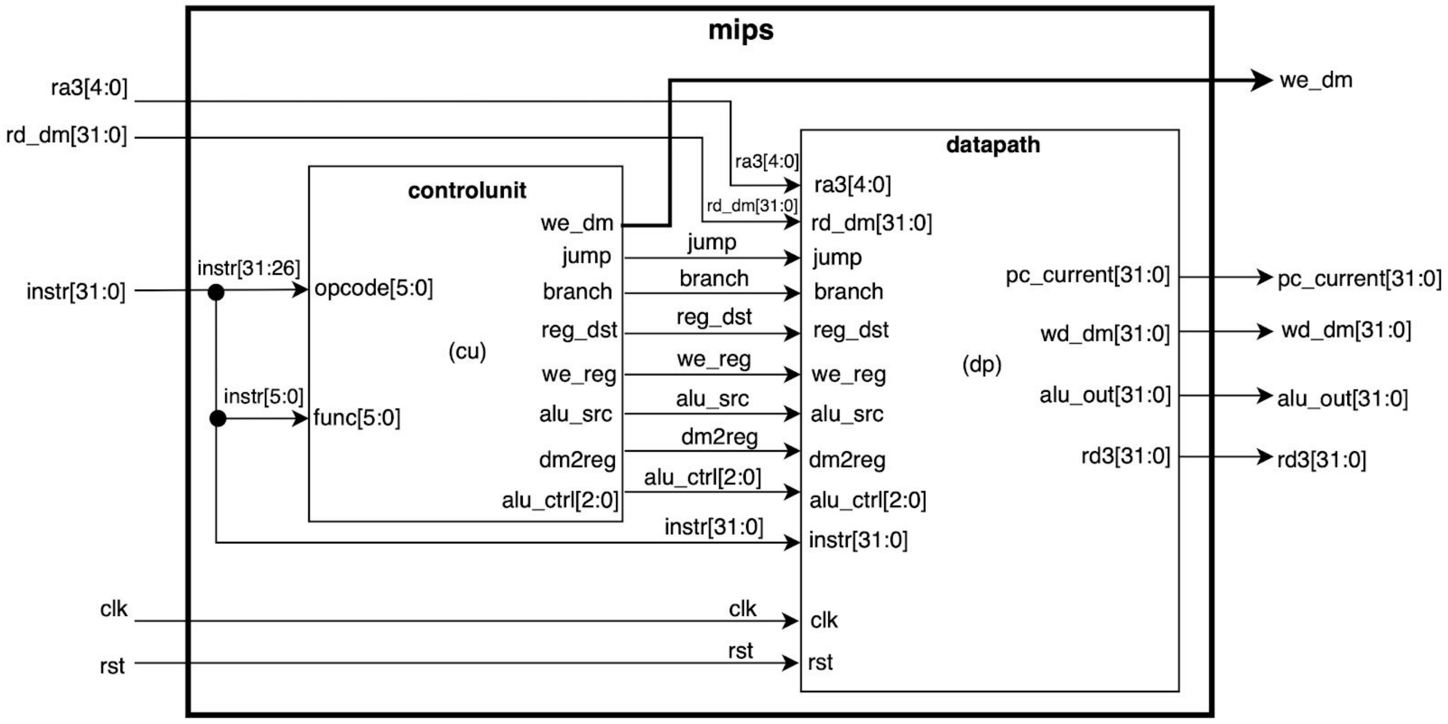


Figure 3: Processor core (including connection between control unit and datapath)

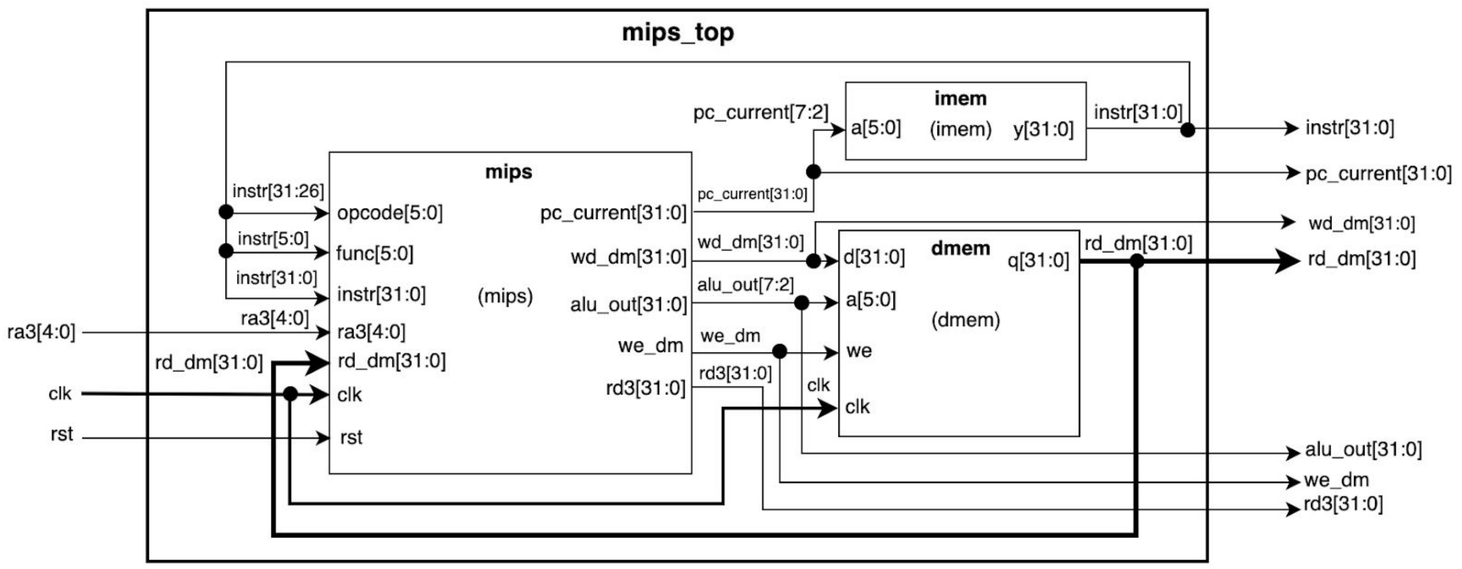


Figure 4: Complete processor (include the connection between processor core and memories)

Table 1: Sample MIPS instruction to functionally verify the DUT testbench design

Label	Assembly	Address (hex)	Machine Code (hex)
main:	*addi $2, $0, 5*	*0*	*20020005*
	addi $3, $0, 12	*4*	*2003000c*
	addi $7, $3, -9	*8*	*2067fff7*
	or $4, $7, $2	*c*	*00e22025*
	and $5, $3, $4	*10*	*00642824*
	add $5, $5, $4	*14*	*00a42820*
	beq $5, $7, end	*18*	*10a7000a*
	slt $4, $3, $4	*1c*	*0064202a*
	beq $4, $0, around	*20*	*10800001*
around:	*addi $5, $0, 0*	*24*	*20050000*
	slt $4, $7, $2	*28*	*00e2202a*
	add $7, $4, $5	*2c*	*00853820*
	sub $7, $7, $2	*30*	*00e23822*
	sw $7, 68($3)	*34*	*ac670044*
end:	*lw $2, 80($0)*	*38*	*8c020050*
	j end	*3c*	*08000011*
	addi $2, $0, 1	*40*	*20020001*
	sw $2, 84($0)	*44*	*ac020054*
	j main	*48*	*08000000*

Simulation Result

The simulation waveforms produced by the testbench file were divided in half as shown in *Figure 5* and *6.* According to *Figure 5,* the instruction read from the instruction memory matched the provided sample instruction shown in *Table 1* above. Furthermore, the arithmetic computation control signal was shown in alu_ctrl to be correct. For instance, the first half of the sample instructions were performing the following operation 'addi', 'or', 'and', and 'add' which matches the encoded alu_ctrl signals in *Figure 5* below. *Figure 6* also shows the expected behavior since the new value was stored (sw) at address 34 to the data memory when we_dm was enabled. Moreover, it is easy to observe the value was correct when it was loaded (lw) at address 38 from the data memory as shown in the rd_dm waveform. In addition, the value(7) stored in data memory produced from the simulation matched the value(7) in data memory from the Mars compiler as shown in *Figure 7*. As a result, the function verification task was successful.

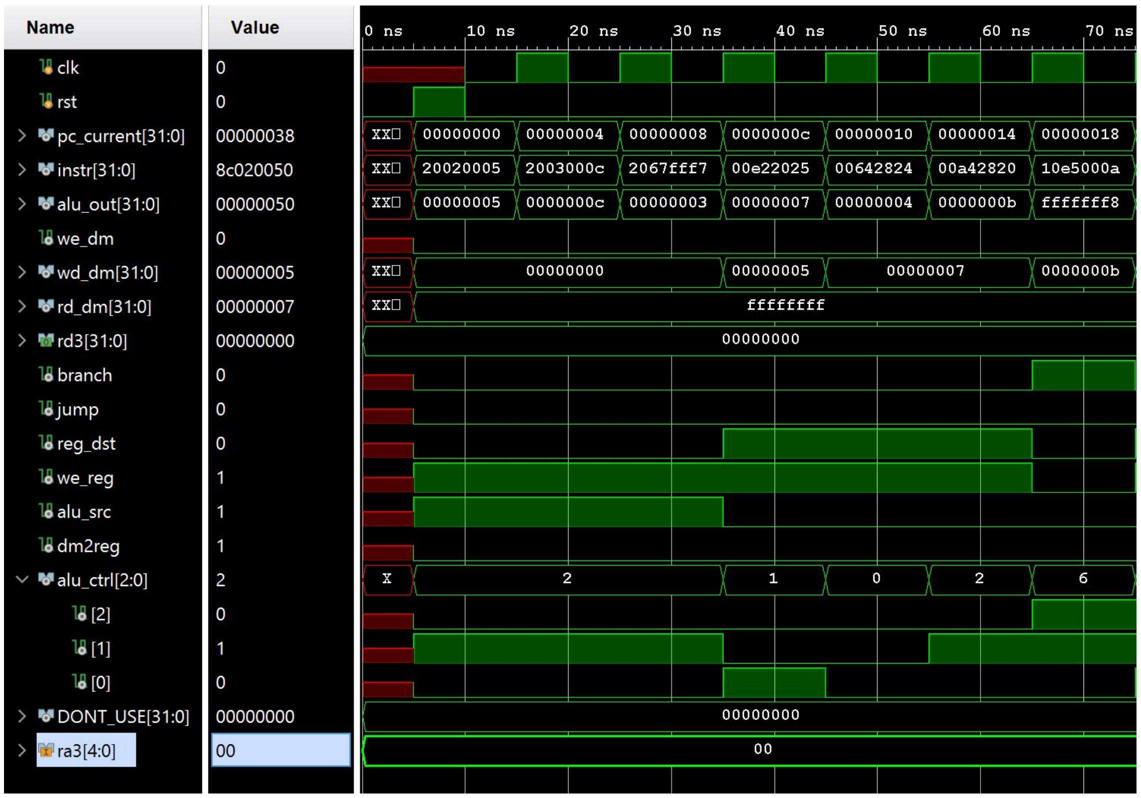


Figure 5: Simulation waveforms of the first half of the given instructions (from 2002005 to 10e5000a)

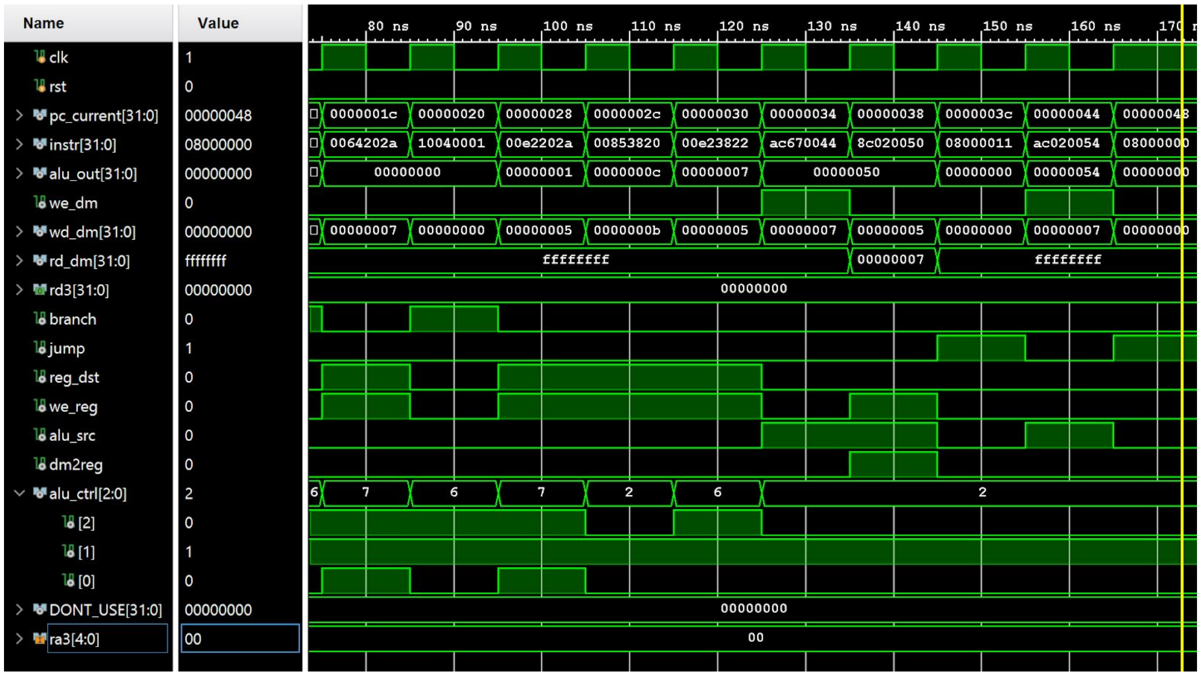
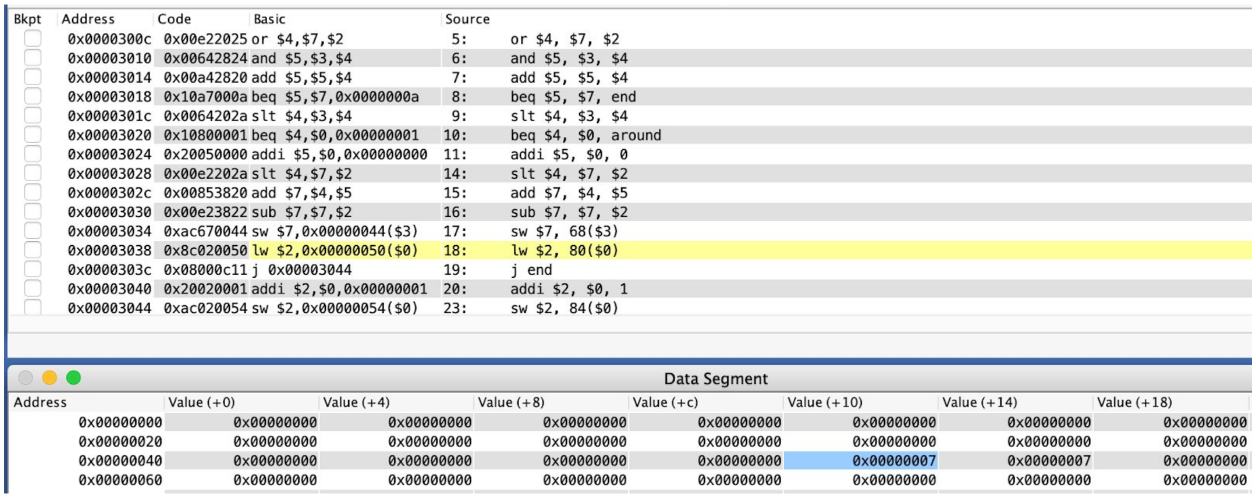


Figure 6: Simulation waveforms of the first half of the given instructions (from 0064202a to 08000000)



```
Bkpt Address     Code       Basic                        Source
     0x0000300c  0x00e22025 or $4,$7,$2                   5:    or $4, $7, $2
     0x00003010  0x00642824 and $5,$3,$4                  6:    and $5, $3, $4
     0x00003014  0x00a42820 add $5,$5,$4                  7:    add $5, $5, $4
     0x00003018  0x10a7000a beq $5,$7,0x0000000a          8:    beq $5, $7, end
     0x0000301c  0x0064202a slt $4,$3,$4                  9:    slt $4, $3, $4
     0x00003020  0x10800001 beq $4,$0,0x00000001          10:   beq $4, $0, around
     0x00003024  0x20050000 addi $5,$0,0x00000000         11:   addi $5, $0, 0
     0x00003028  0x00e2202a slt $4,$7,$2                  14:   slt $4, $7, $2
     0x0000302c  0x00853820 add $7,$4,$5                  15:   add $7, $4, $5
     0x00003030  0x00e23822 sub $7,$7,$2                  16:   sub $7, $7, $2
     0x00003034  0xac670044 sw $7,0x00000044($3)          17:   sw $7, 68($3)
     0x00003038  0x8c020050 lw $2,0x00000050($0)          18:   lw $2, 80($0)
     0x0000303c  0x08000c11 j 0x00003044                  19:   j end
     0x00003040  0x20020001 addi $2,$0,0x00000001         20:   addi $2, $0, 1
     0x00003044  0xac020054 sw $2,0x00000054($0)          23:   sw $2, 84($0)

                                   Data Segment
Address     Value (+0)  Value (+4)  Value (+8)  Value (+c)  Value (+10) Value (+14) Value (+18)
0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000
0x00000020  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000
0x00000040  0x00000000  0x00000000  0x00000000  0x00000000  0x00000007  0x00000007  0x00000000
0x00000060  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000  0x00000000
```

Figure 7: Sample code compiled by Mars compiler

Conclusion

In sum, we have gained a lot of knowledge in microprocessor design by reading through the RTL code of the initial version of the single-cycle MIPS processor. By reading the given code, we were able to visualize and sketch the block diagrams of the datapath, the control unit, microprocessor core, and the complete microprocessor. In addition, we also learned the basic techniques to functionally verify a processor by looking at the waveforms generated by the testbench.

Successful Tasks

1. We were able to draw a block diagram such as datapath, control unit, processor code, and complete processor
2. We were able to capture the waveform from the testbench
3. We were able to build the DUT and it testbench
4. We were able to functional verify base on the given sample program.