Lab Assignment #3

Teo Patrosio

EE-126/COMP-46: Computer Engineering w/lab

Professor: Mark Hempstead TA:Parnian Mokri

Tufts University, Fall 2021

Please apply 2 late day.

Introduction + Purpose

In this lab assignment the previous components will be connected and modified to support a single cycle cpu. Through this lab, a deep understanding of the data path and debugging with waveforms skills will be gained.

Components built

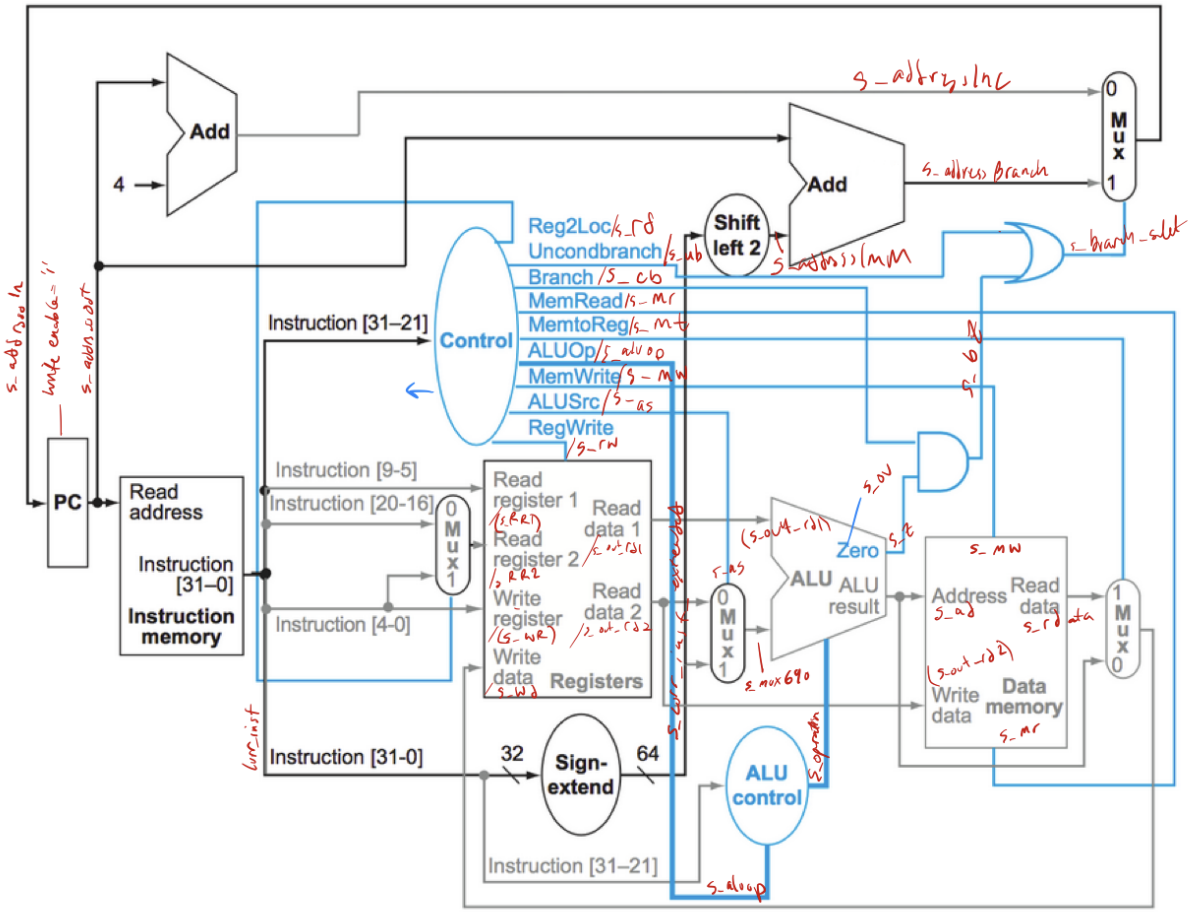

Figure 1: Figure 4-23 from Computer Organization and Design ARM Edition annotated with signal names that were used to connect the various components.

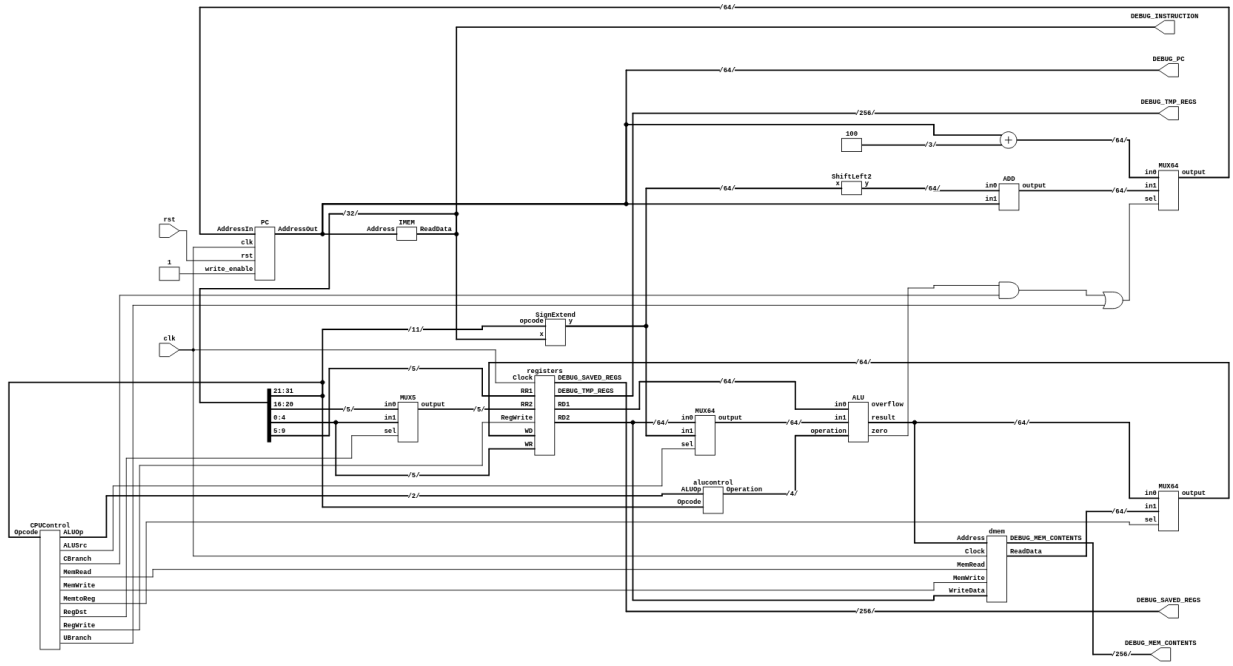

Figure 2: Block diagram generated from the single cycle cpu.

To run compile and simulate the single cycle cpu:

```
vcom -quiet -2008 *.vhd imem/imem_ldstr.vhd
vsim -quiet -c work.singlecyclecpu_tb -do "log -r /*; run 500ns; exit" -wlf
singlecyclecpu.wlf &&  wlf2vcd singlecyclecpu.wlf -o singlecyclecpu.vcd
```

These commands will specifically compile the imem_ldstr.vhd, however that can be swapped out for any test imem.vhd file. Note, the imems are kept in a folder within the directory of the singlecyclecpu and its dependents.

Annotated test program waveforms

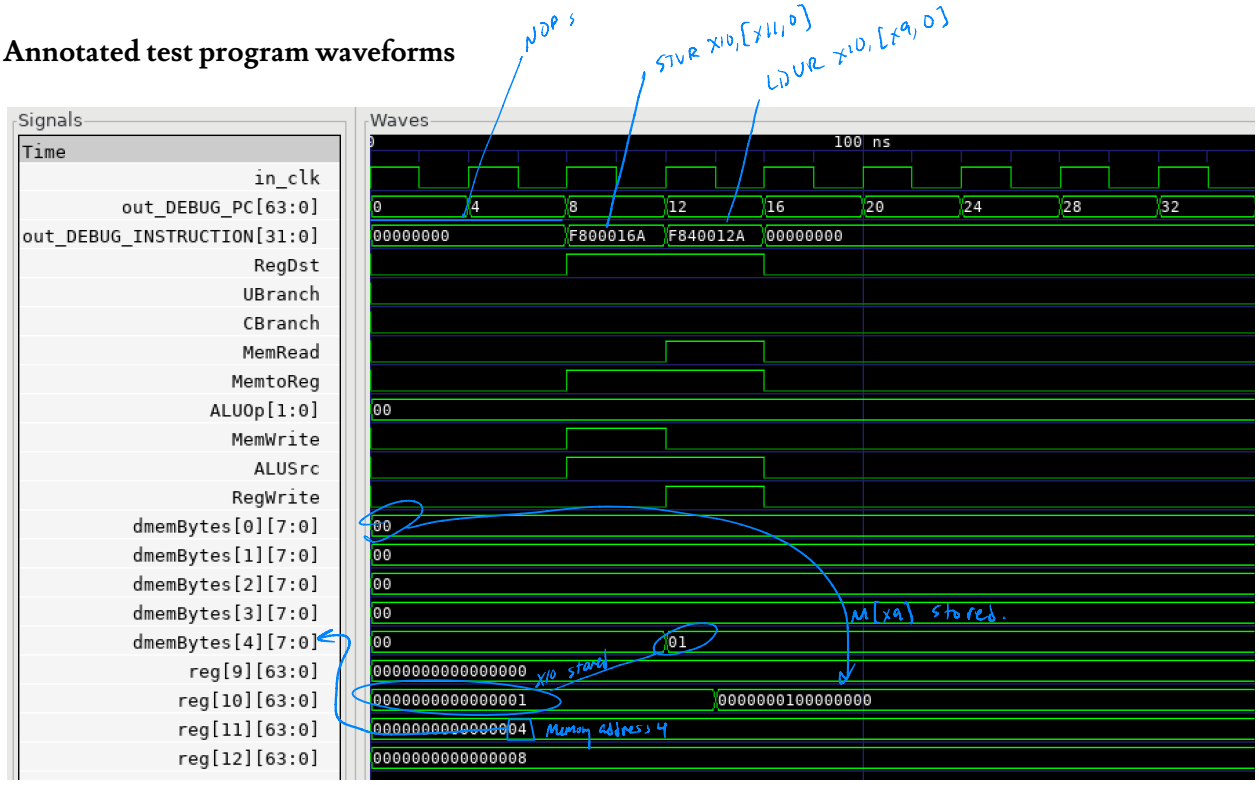

Figure 3: Output waveforms from IMEM_ldstr

```
        -- STUR X10, [X11,0]
        imemBytes(11) <= "11111000";
        imemBytes(10) <= "00000000";
        imemBytes(9)  <= "00000001";
        imemBytes(8)  <= "01101010";

        -- LDUR X10, [X9, 0]
        imemBytes(15) <= "11111000";
        imemBytes(14) <= "01000000";
        imemBytes(13) <= "00000001";
        imemBytes(12) <= "00101010";
```

Figure 4: IMEM_ldstr test program

The program first starts with two NOP instructions as seen in the DEBUG_INSTRUCTION[31:0] output in the first two clock cycles. This is due to the IMEM databytes at 0-3 and 4-7 being set to all zeros which were added to the imem_ldstr program. If they are not set, they may appear as floating signals.

At the rising edge of the third clock cycle, the first instruction (STUR X10, [X11,0]) which is seen as 0xF800016A in DEBUG_INSTRUCTION[31:0]. An important thing to note is the control bits that are set during this instruction. They update according to Figure 4.22 from the textbook (shown in Figure 5). It should also be noted that the contents of registers X9, X10, and X11 are 0x0, 0x1, and 0x4 respectively. In the STUR instruction, the data in X10 is being saved in dmem at the contents of X11 with an offset of zero. Accordingly the MemWrite control bit is set to high. At the end of the cycle it can be noted that the dmem at byte 4 (X11 + 0) is indeed updated to 1, the contents of X10.

At the rising edge of the fourth clock cycle, it can be seen that the second (non NOP: LDUR X10, [X9, 0]) instruction is pulled which is seen as 0xF8440012. The control bits are updated again in accordance with Figure 4.22 from the textbook, shown here as Figure 5. In LDUR instructions, the memory contents stored in the dmem at the address stored in X9 (with an offset of 0) is fetched and then set to X10. Since X9 is zero, and the offset is zero, the contents of dmem at 0 is pulled and then set to X10.

Input or output	Signal name	R-format	LDUR	STUR	CBZ
Inputs	I[31]	1	1	1	1
	I[30]	X	1	1	0
	I[29]	X	1	1	1
	I[28]	0	1	1	1
	I[27]	1	1	1	0
	I[26]	0	0	0	1
	I[25]	1	0	0	0
	I[24]	X	0	0	0
	I[23]	0	0	0	X
	I[22]	0	1	0	X
	I[21]	0	0	0	X
Outputs	Reg2Loc	0	X	1	1
	ALUSrc	0	1	1	0
	MemtoReg	0	1	X	X
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Figure 5: Figure 4.22 from the Computer Organization and Design ARM Edition used to implement CPUControl

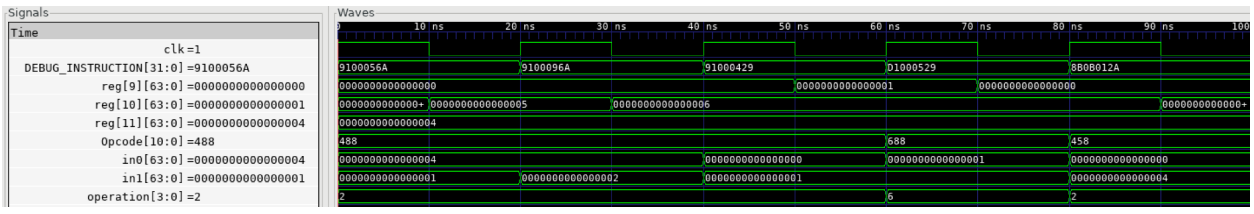

Figure 6: Overall output waveform from IMEM_comp

```
    --ADDI X10, X11, 1
imemBytes(3) <= "10010001";
imemBytes(2) <= "00000000";
imemBytes(1) <= "00000101";
imemBytes(0) <= "01101010";

--ADDI X10, X11, 2
imemBytes(7) <= "10010001";
imemBytes(6) <= "00000000";
imemBytes(5) <= "00001001";
imemBytes(4) <= "01101010";

-- ADDI X9, X9, 1
imemBytes(11) <= "10010001";
imemBytes(10) <= "00000000";
imemBytes(9)  <= "00000100";
imemBytes(8)  <= "00101001";

-- SUBI X9, X9, 1

imemBytes(15) <= "11010001";
imemBytes(14) <= "00000000";
imemBytes(13) <= "00000101";
imemBytes(12) <= "00101001";

-- ADD  X10, X9, X11
imemBytes(19) <= "10001011";
imemBytes(18) <= "00001011";
imemBytes(17) <= "00000001";
imemBytes(16) <= "00101010";
```

Figure 7: IMEM_comp test program contents

For this test program, shown in Figure 7 and the execution in Figure 6, the annotations will be shown primarily directly on the waveforms.

Shown in Figure 8:

Clock cycle 1: 0x9100056A [ADDI X10, X11, 1]

X11 + 1 = X10

4 + 1 = 5

Clock cycle 2: 0x91000964 [ADDI X10, X11, 2]

X11 + 2 = X10

4 + 2 = 6

Clock cycle 3: 0x91000429 [ADDI X9, X9, 1]

X9 + 1 = X9

0 + 1 = 0

Shown in Figure 9:

Clock cycle 4: 0xD1000529 [SUBI X9, X9, 1]

X9 - 1 = X9

1 - 1 = 0

Clock cycle 5: 0x8B0B012A [ADD X10, X9, X11]

X9 + X11 = X10

0 + 4 = 4

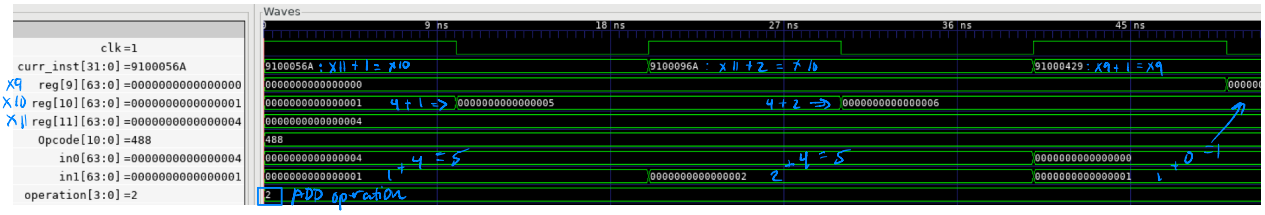

Figure 8: Close up of waveforms from IMEM_comp

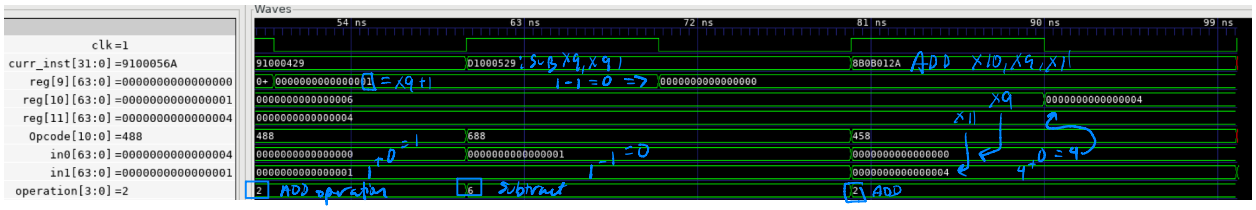

Figure 9: Close up of waveform from IMEM_comp continued

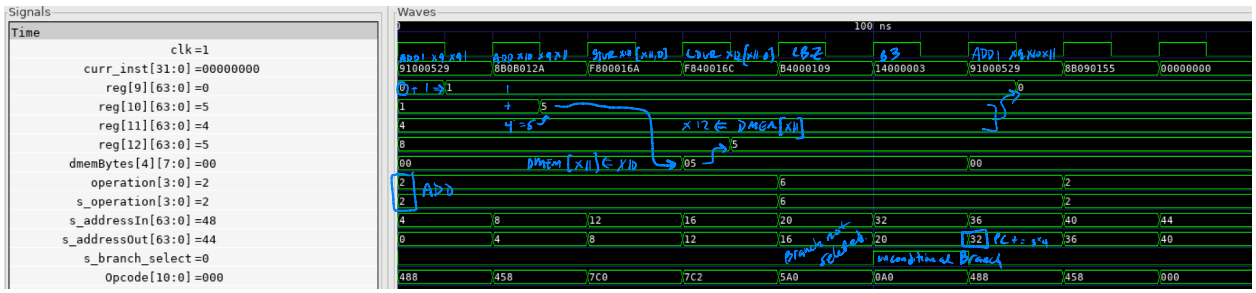

Figure 10: Partial output waveform for IMEM_p1

```
  -- ADDI X9, X9, 1 -- this we changed after the OH tonight
imemBytes(3) <= "10010001";
imemBytes(2) <= "00000000";
imemBytes(1) <= "00000101";
imemBytes(0) <= "00101001";
 --ADD  X10,X9,X11

imemBytes(7) <= "10001011";
imemBytes(6) <= "00001011";
imemBytes(5) <= "00000001";
imemBytes(4) <= "00101010";

-- STUR X10, [X11,0]
imemBytes(11) <= "11111000";
imemBytes(10) <= "00000000";
imemBytes(9)  <= "00000001";
imemBytes(8)  <= "01101010";

-- LDUR X12, [X11, 0]

imemBytes(15) <= "11111000";
imemBytes(14) <= "01000000";
imemBytes(13) <= "00000001";
imemBytes(12) <= "01101100";

-- CBZ X9, 2 --> 4
imemBytes(19) <= "10110100";
imemBytes(18) <= "00000000";
imemBytes(17) <= "00000001";
imemBytes(16) <= "00001001";

-- B 3
imemBytes(23) <= "00010100";
imemBytes(22) <= "00000000";
imemBytes(21) <= "00000000";
imemBytes(20) <= "00000011";

-- ADD X9, X10, X11
imemBytes(27) <= "10001011";
imemBytes(26) <= "00001011";
imemBytes(25) <= "00000001";
imemBytes(24) <= "01001001";

-- ADDI  X9, X9, 1
imemBytes(35) <= "10010001";
imemBytes(34) <= "00000000";
imemBytes(33) <= "00000101";
imemBytes(32) <= "00101001";

-- ADD   X21, X10, X9
imemBytes(39) <= "10001011";
imemBytes(38) <= "00001001";
imemBytes(37) <= "00000001";
imemBytes(36) <= "01010101";
```

Figure 11: IMEM_P1 test program contents

For this test program, shown in Figure 11 and the execution in Figure 10, the annotations will be shown primarily directly on the waveforms. The program flow starts at IMEM 1 which is the same thing as IMEM_comp, then the two commands from IMEM_ldstr are run. A conditional branch is tested and not taken. Then the unconditional branch [B 3] is taken. 3 * 4 is then added to the PC counter which ends in the operation of some ADDIs which are the same as IMEM_comp. The important values are highlighted directly on the waveform in Figure 10.