Lab Assignment #4

Teo Patrosio

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

Professor: Mark Hempstead TA:Parnian Mokri

Tufts University, Fall 2021

Introduction + Purpose

In this lab assignment the single cycle CPU will be modified to allow for pipelining without hazard detection.

Components built

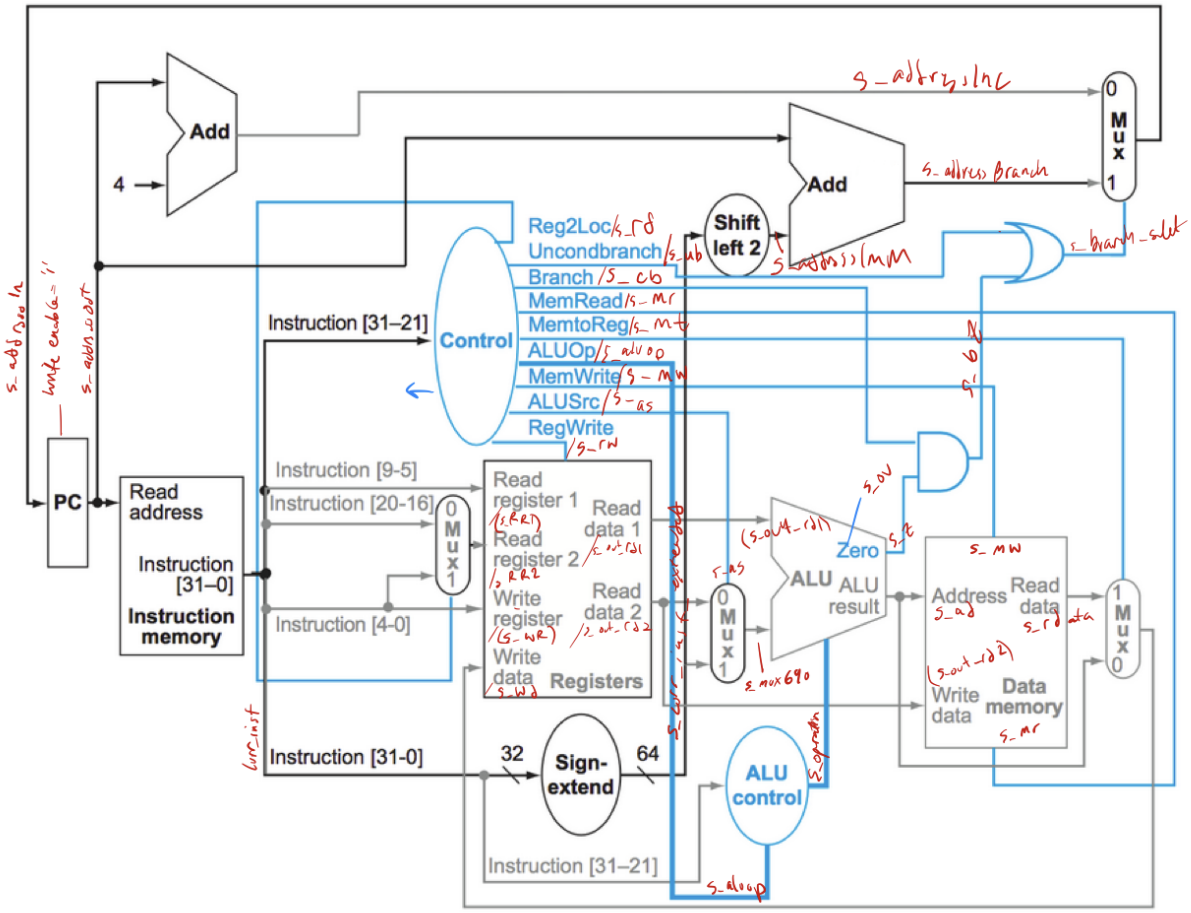

Figure 1: Figure 4-23 from Computer Organization and Design ARM Edition annotated with signal names that were used to connect the various components from the single cycle cpu.

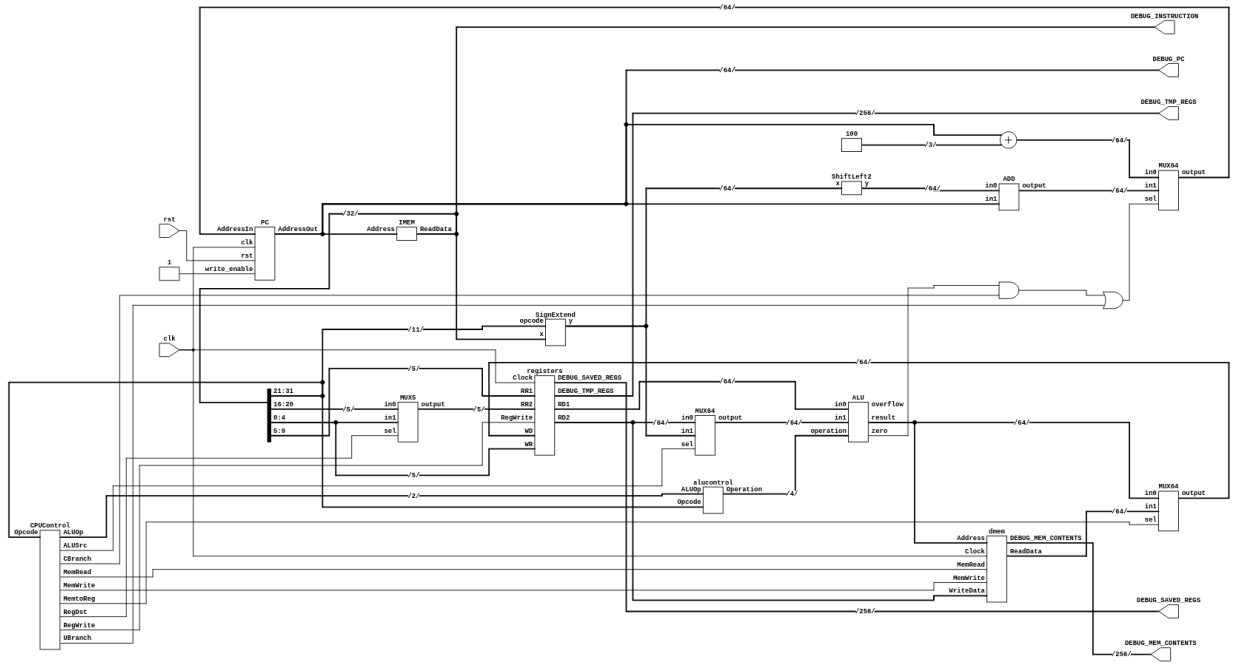

Figure 2: Block diagram generated from the single cycle cpu.

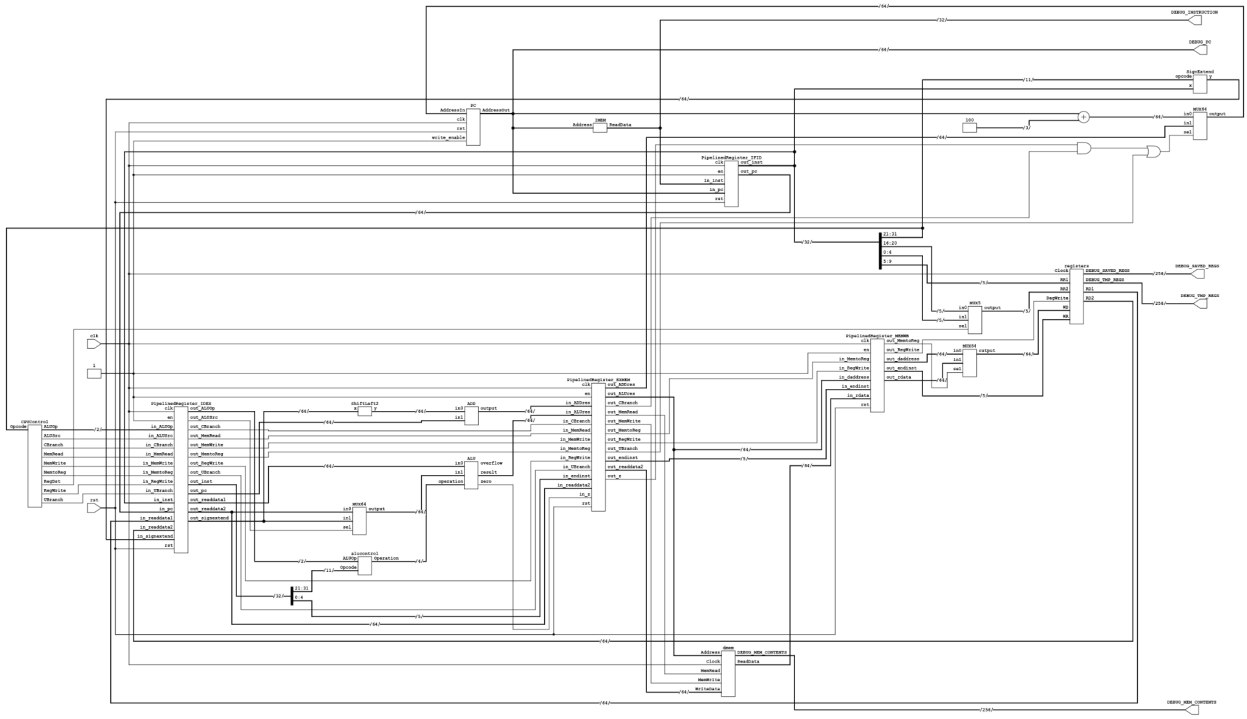

Figure 3: Block diagram generated from the pipelined cpu.

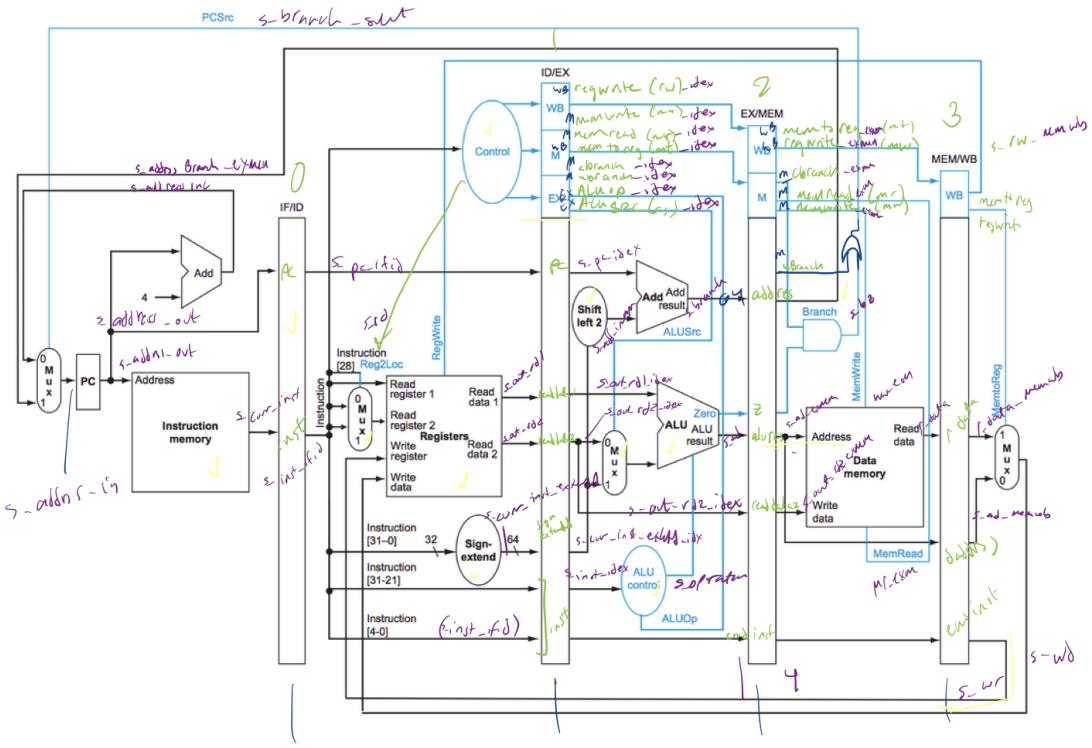

Figure 4: Figure 4-50 from Computer Organization and Design ARM Edition annotated with signal names that were used to connect the various components from the pipelined cpu.

To run compile and simulate the pipelinedcpu0:

```
vcom -quiet -2008 *.vhd tb/*.vhd imem/*.vhd
vsim -quiet -c work.pipelinedcpu0_tb -do "log -r /*; run 500ns; exit" -wlf
pipelinedcpu0.wlf &&  wlf2vcd pipelinedcpu0.wlf -o pipelinedcpu0.vcd
```

These commands will specifically compile the pipe0_imem.vhd kept in the /imem directory.

Annotated test program waveforms

PC (dec)			
0	ADD X11, X9, X10	10001011000010100000000100101011	8b 0a 01 2b
4	STUR X11, XZR,0	11111000000000000000001111101011	f8 00 03 eb
8	SUB X12, X9, X10	11001011000010100000000100101100	cb 0a 01 2c
12	STUR X11, [XZR,0]	11111000000000000000001111101011	f8 00 03 eb
16	STUR X12, [X12,8]	11111000000000001000000110001100	f8 00 81 8c
20	STUR X12, [X12,8]	11111000000000001000000110001100	f8 00 81 8c
24	ORR X21, X19, X20	10101010000101000000001001110101	aa 14 02 75
28	NOP		
32	NOP		
36	STUR X21, [XZR,0]	11111000000000000000001111110101	f8 00 03 f5
40	NOP		
44	NOP		
48	NOP		
52	NOP		

Figure 5: The test program provided with associated binary and values in hex.

	Clock Cycle														
PC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	IF	ID	EX	MEM	WB										
4		IF	ID	EX	MEM	WB									
8			IF	ID	EX	MEM	WB								
12				IF	ID	EX	MEM	WB							
16					IF	ID	EX	MEM	WB						
20						IF	ID	EX	MEM	WB					
24							IF	ID	EX	MEM	WB				
36										IF	ID	EX	MEM	WB	

Figure 6: Instruction Pipeline timeline.

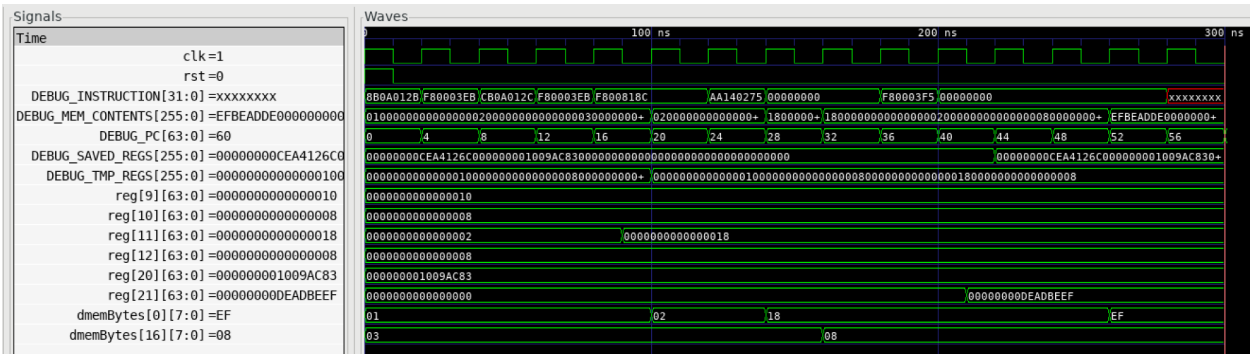

Figure 7: Output waveform of the test program

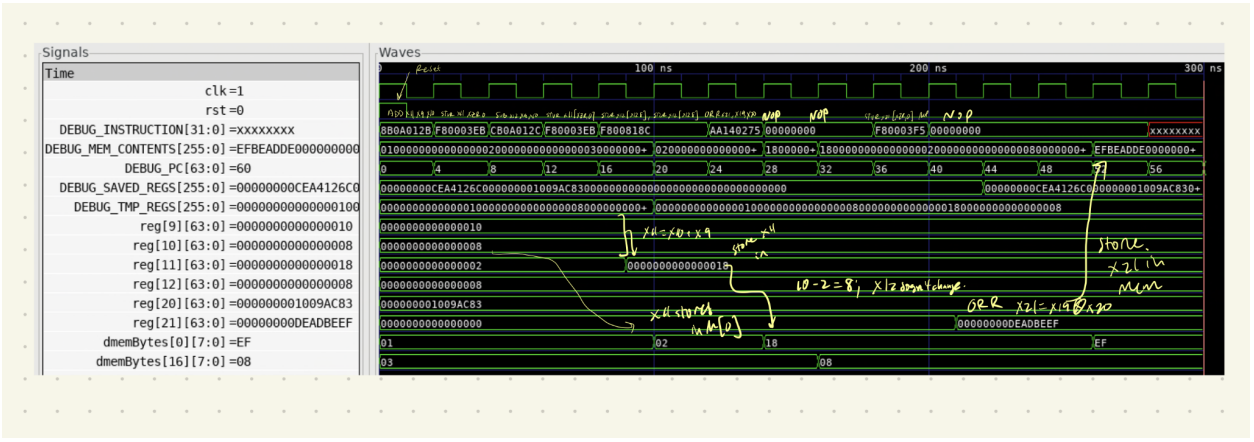

Figure 8: Annotated output waveforms of the test program

The instructions can be clearly seen in DEBUG_INSTRUCTION that matches the instructions laid out in Figure 5 and 6 in accordance with their clock signals. Each instruction is fetched the cycle after the last, with no regard to data hazards (or any hazards for that matter). A clear example of this is that the first instruction computes the value in X11, and the instruction right after requires the value in X11 before it can be updated as per the first instruction. This is apparent as the value fetched from X11 at cycle 3 by the second instruction (STUR X11, XZR,0) is still 2 before it's updated from the ADD instruction in its WB phase , and then the dmem value at 0 is accordingly updated to 2 in its MEM phase. The third instruction does not appear to update any signals shown on the output as the value in X9 during the ID phase is 10 and X10 is 2. X12 is originally 8 so it appears as if nothing changes.

The fourth instruction (STUR X11, [XZR,0]), stores X11, as it updates to be 18 from the first ADD instruction to Mem[0]. Calling the fifth instruction STUR X12, [X12,8], cases dmem[16] to update 08, three cycles later in its MEM phase. The result of the the ORR instruction updates X21 to 0xDEADBEEF during the 11th clock cycle. Two NOP instructions are inserted presumably so X21 has time to update. Finally a final STUR is called and saves the 0xDEADBEEF string to memory.