[redacted]

COEN 122 Computer Architecture

[redacted]June 2021

COEN 122 Project Final Report

Abstract

This project was created to design a 32-bit pipelined CPU for a given SCU Instruction Set Architecture (SCU ISA). This SCU ISA uses a register file size of 64 registers where each register has 32 bits. The PC is 32 bits and each instruction is 32-bit wide, and consists of five fields: opcode, rd, rs, rt, and unused. With this CPU design, instructions in the SCU ISA should be able to write two versions of the assembly program to compute and find the maximum of n numbers where $A = \max\{a_1,a_2,...,a_n\}$. The first version will not use the MAX instruction while the second one will use the MAX instruction.

Process

Our datapath and verilog had to go through many iterations before reaching the final product. In our first iteration, we spent a good chunk of time making sure all the wires were correctly connected with the appropriate amount of bits assigned correctly. We then soon realized that our phase 3 branch mux needed to be passed through the Ex-Mem-Wb buffer and soon changed it to work with a 3to1mux to function as expected. The next biggest change we made from our original datapath was making sure our phases were correctly synced since we were troubled with our CPU being a cycle off. In our final iterations, we decided to add buffers and NOPs. This helped tremendously since it synchronized our PC and memory instruction. By focusing on a

single instruction at a time, we were able to add NOPs and buffers to ensure that all the connections lead to the desired results on the waveform. Our final version of the datapath was achieved by altering our branching technicalities such that the 3to1mux uses the branch condition from ID/EX and ALU values in EX-MEM-WB.

Description of CPU

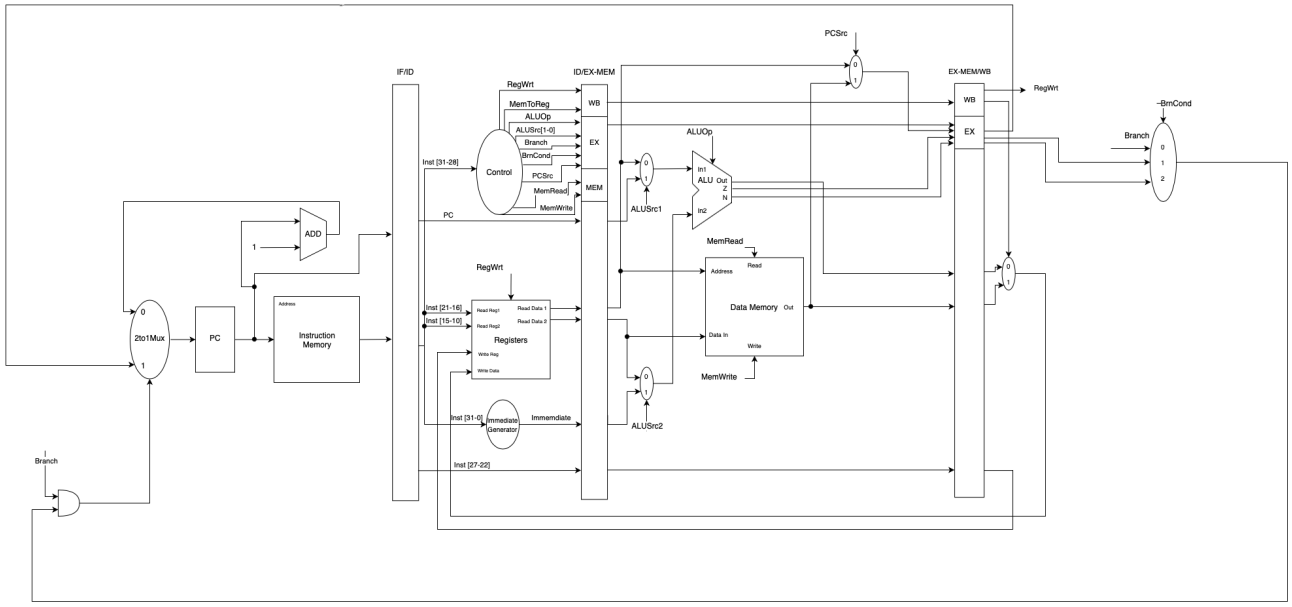


Figure 1: Datapath

Control Truth Table

Instruct -ion	Symbol	Opcode	WriteBack		Memory		Execution				
			Register Write	MemTo-Reg	Read Memory	Write Memory	ALUOp	ALUSrc	Branch	Branch condition	PCSrc
No Op	NOP	0000	0	0	0	0	0000	00	0	0	X
SavePC	SVPC	1111	1	0	0	0	0100	11	0	0	X
Load	LD	1110	1	1	1	0	XXXX	XX	0	0	X
Store	ST	0011	0	0	0	1	XXXX	XX	0	0	X
Add	ADD	0100	1	0	0	0	0100	00	0	0	X
Increme nt	INC	0101	1	0	0	0	0100	01	0	0	X
Negate	NEG	0110	1	0	0	0	0010	0X	0	0	X

Subtract	**SUB**	**0111**	**1**	**0**	**0**	**0**	**0001**	**00**	**0**	**0**	**X**
Jump	**J**	**1000**	**0**	**X**	**0**	**0**	**XXXX**	**XX**	**1**	**0**	**0**
Branch Zero	**BRZ**	**1001**	**0**	**0**	**0**	**0**	**XXXX**	**XX**	**1**	**1**	**0**
Jump Memory	**JM**	**1010**	**0**	**0**	**1**	**0**	**XXXX**	**XX**	**1**	**0**	**1**
Branch Negative	**BRN**	**1011**	**0**	**0**	**0**	**0**	**XXXX**	**XX**	**1**	**2**	**0**

<u>Simulation Verification</u>: Waveforms and descriptions for the operation of at least two different instructions and a waveform and description for the max calculation

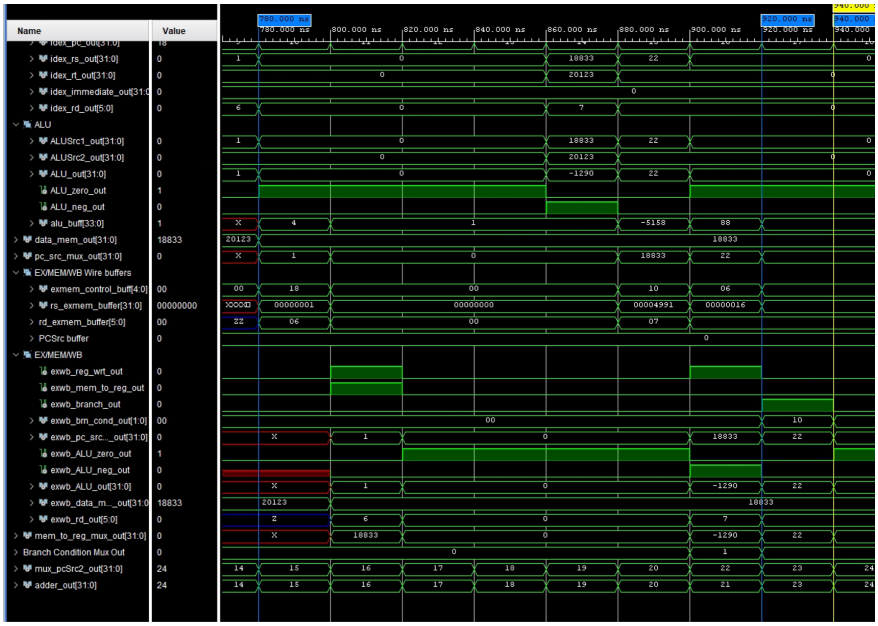
<u>Waveform for first operation</u>:

Description for first operation:

Instruction: INC x4, x0, -32 => x4 = 0 + - 32 = -32

Instruction takes 7 cycles to complete. Instruction Fetch takes 2 cycles, Instruction Decode takes another 2 cycles. The Execution stage takes 2 cycles and WriteBack takes 1 cycle.

Waveform for second operation:



Description for second operation:

SUB and BRANCH

Waveform for max calculation:

Description for max calculation:

SVPC instruction for max calculation

Assembly Code for Max Calculation

Assumptions

A = [a1, a2, ..., an]
x0 = 0
x2 = Base address of A = address(a1)
x3 = n
x4 = max(a1, a2, ..., an) = return register

Assembly Code Version 1

```
MAX     x4, x2, x3
```

Assembly Code Version 2

```
INC     x4, x0, -2^5     #Set max to smallest value
ADD   x5, x2, x3
NOP
NOP
NOP
NOP
INC     x5, x5, -1                  #x5 = Address(a1) + n-1  = Address(an)
```

```
SVPC  x9, 2
SVPC  x10, 13
LD    x6, x2          #x6 = Next element
NOP
NOP
NOP
NOP
SUB   x7, x6, x4      #x7 = ai - max
BRN   x10             #branch if max >= ai
NOP
NOP
NOP
NOP
NOP
ADD   x4, x6, x0
INC   x2, x2, 1
NOP
NOP
NOP
NOP
SUB   x8, x2, x5      #x8 = Address(ai) - Address(an)
BRN   x9              #loop if Address(an) < Address(ai)
```

Estimate of Running Time for Max Calculation - Using the component delays defined in the lab handout, we estimated the number of cycles and actual running time for the execution of the max assembly code.. This was concluded from the teacher-led assumption that the delay of memory was 2ns, register delay was 1.5ns, and delay of ALU of 2ns. Since the last time any data is updated is 940 ns, accounting for the delays and the fact that each step in the datapath is 10ns, our cycle times will be 98.5 or 99 cycles.