


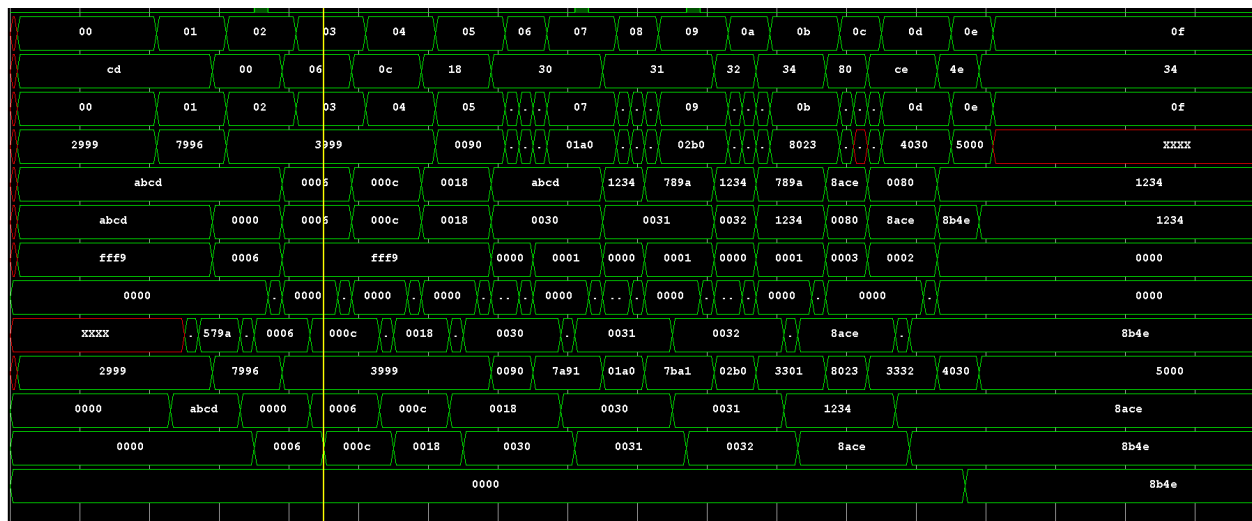
Jack Landers & Marissa Kuo
ELEN 122 Lab 5

mem.txt:

```
2000 // sub r0, r0, r0 // TODO #5 write your own program here
7006 // addi r0, r0, 6
3100 // add r1, r0, r0
3211 // add r2, r1, r1
3222 // add r2, r2, r2
0320 // load r3, r2(=48)
4030 // disp r3
8022 // store r2, r2
5000 // halt
```

>  [130][15:0]	XXXX	Array
>  [129][15:0]	XXXX	Array
>  [128][15:0]	8ace	Array
>  [127][15:0]	XXXX	Array
>  [126][15:0]	XXXX	Array

The value 8ace is stored at address 128 after program execution.



Waveform for the entire program. Displays sum of val1, val2, and val3.

1. What problems did you encounter while implementing and testing your system?

- a. We used the wrong opcode for addi so the program was adding values from a register instead of adding 1. In the data mux, we used the wrong wire in the conditional so our program didn't load external immediates.
2. Did any problems arise when demonstrating for the TA? In other words, did the TA ask you to demonstrate something that you did not think of yourself? What was the scenario that you were asked to demonstrate? Provide some thoughts about why you did not think of this yourself.
 - a. Other than our problems while testing our system, we did not encounter any issues when demonstrating.
3. This design uses memory addresses directly from registers, rather than allowing for an offset to be added to a register value. How would you modify the datapath to allow for an offset, particularly for a store operation? How would this impact your state machine design?
 - a. We would have to be able to take a memory address stored in a register and add an immediate to it which gives us a new memory location. We would then have to access the memory location and write data there.
 - b. In order to do this, we need an adder that would take mem_out and imm as inputs and output the memory address which would be the memory input for the data mux.