David Young
520648

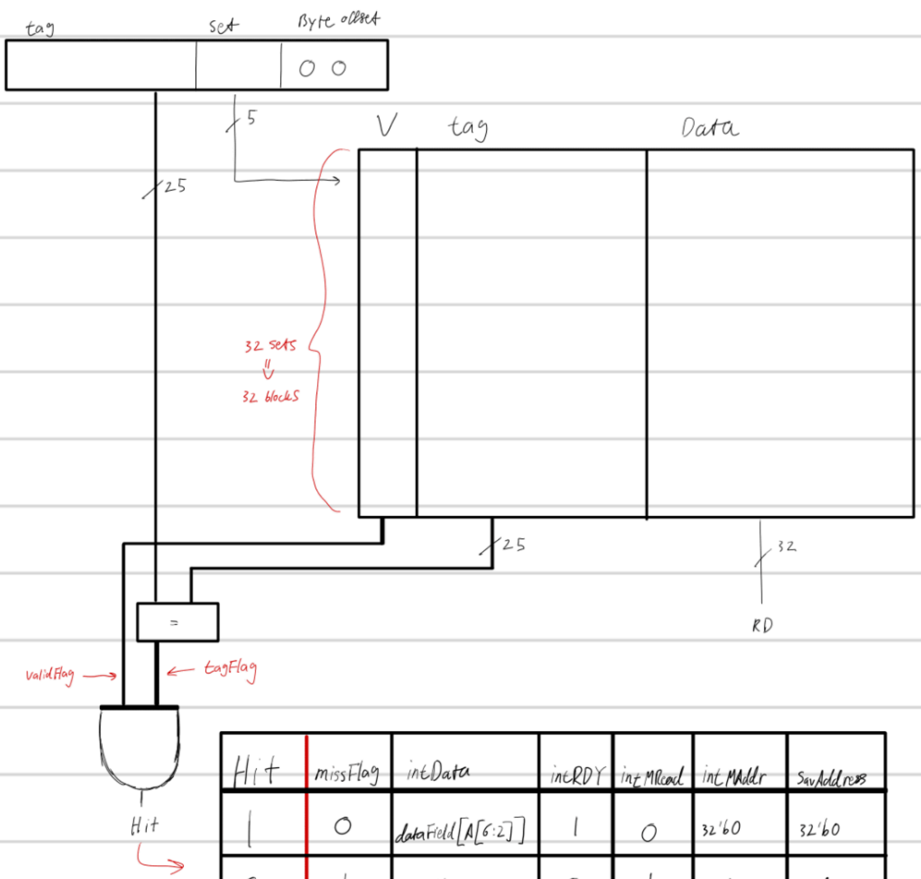
ESE362 - HW4

1. **Time Spent:** More than 12 hours.
2. **Figure 1** shows my diagram of my icache design. I have split icache into two sections: sequential logic and combinational logic.

David Young
520648

#2

Combinational Logic

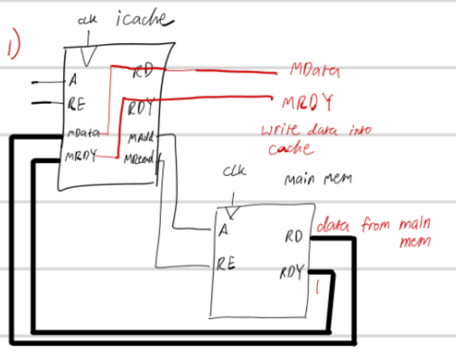


Hit	missFlag	intData	intRDY	int MRead	int MAddr	SavAddress
1	0	dataField[A[6:2]]	1	0	32'b0	32'b0
0	1	32'b0	0	1	A	A

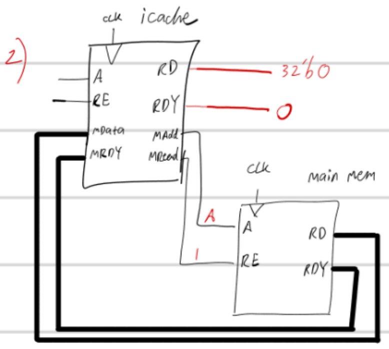
Output to the sequential block or main memory.

Sequential logic : 3 possibilities : 1) Data from main mem, 2) Miss , and 3) cache has the data.

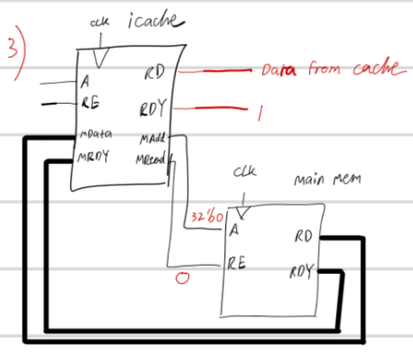
1)



2)



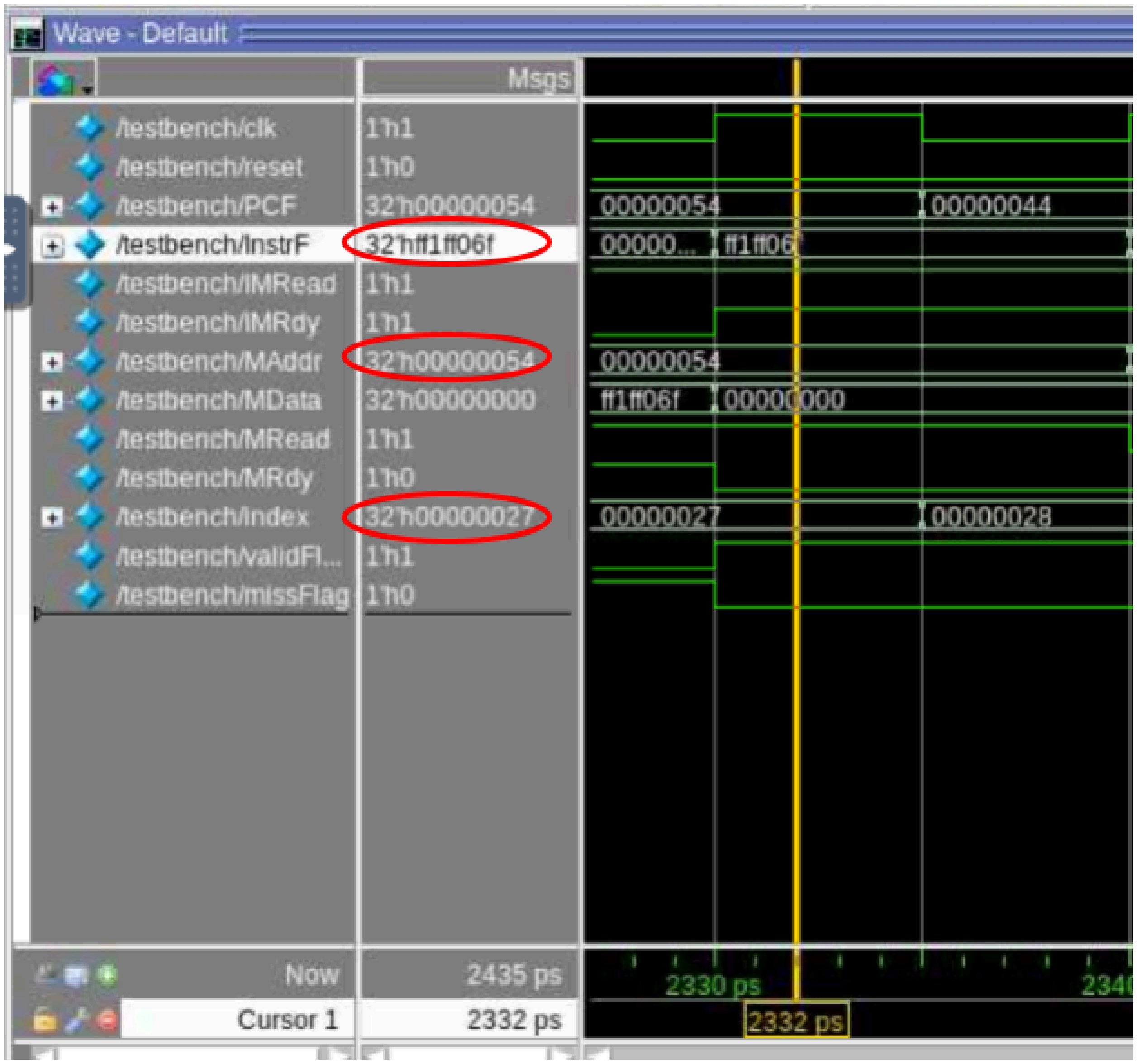
3)



David Young
520648

Figure 1 - Rough Diagram of icache Design

3. **Figure 2** shows the waveform screenshot of a time in the simulation.

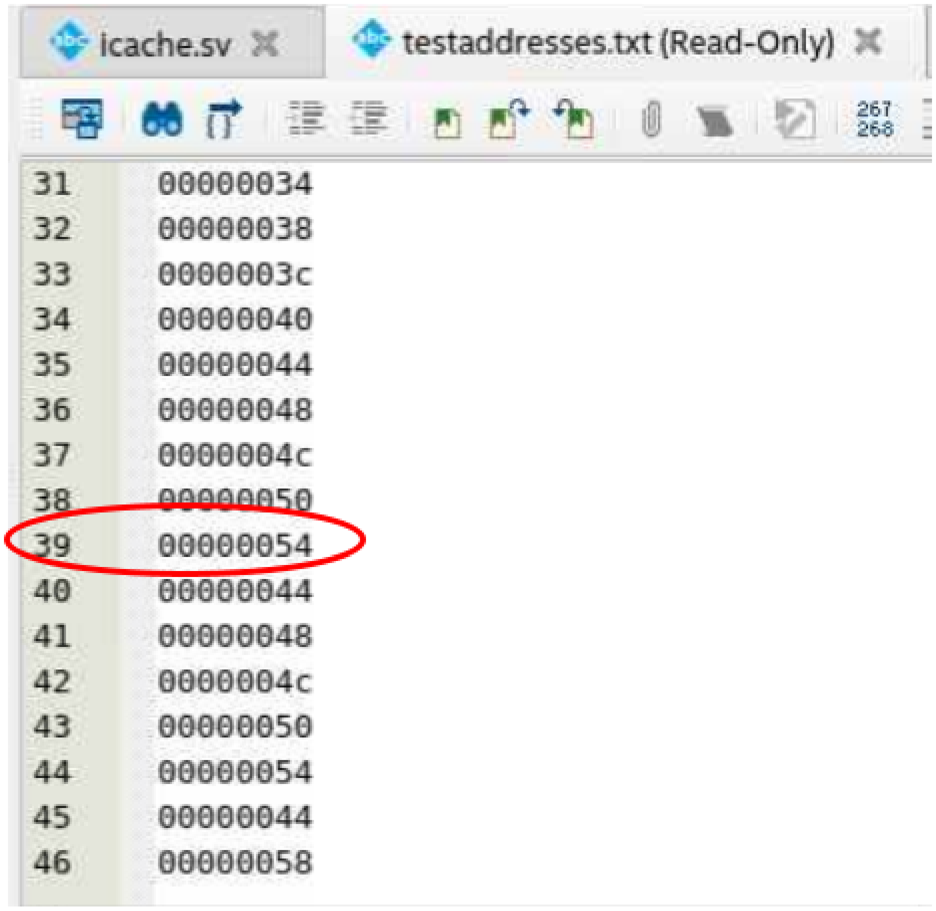


```
Wave - Default
                          Msgs
/testbench/clk            1'h1
/testbench/reset          1'h0
/testbench/PCF            32'h00000054    00000054          00000044
/testbench/InstrF         32'hff1ff06f    00000...  ff1ff06f
/testbench/IMRead         1'h1
/testbench/IMRdy          1'h1
/testbench/MAddr          32'h00000054    00000054
/testbench/MData          32'h00000000    ff1ff06f  00000000
/testbench/MRead          1'h1
/testbench/MRdy           1'h0
/testbench/index          32'h00000027    00000027          00000028
/testbench/validFl...     1'h1
/testbench/missFlag       1'h0

Now                       2435 ps         2330 ps           2340
Cursor 1                  2332 ps         2332 ps
```

Figure 2 - A Random Time in the Simulation

In **Figure 2**, I have three numbers circled in red: InstrF (The Instruction output from icache corresponding to an address input into icache), MAddr (The address sent to main memory because cache did not have the instruction), and index (equivalent to program counter). By checking a random moment in time, I hope to show that if the output of my instruction memory is correct, then the rest are correct. To verify the correctness of the output of the instruction memory (InstrF), I first looked at the index field. 0X27 = 39 in decimal. If I look at **Figure 3,** I see that at an index of 39, the corresponding address that should be sent to the main memory to retrieve an instruction should be 0X54 = 84 in decimal, which I do see in the MAddr field in **Figure 2**.

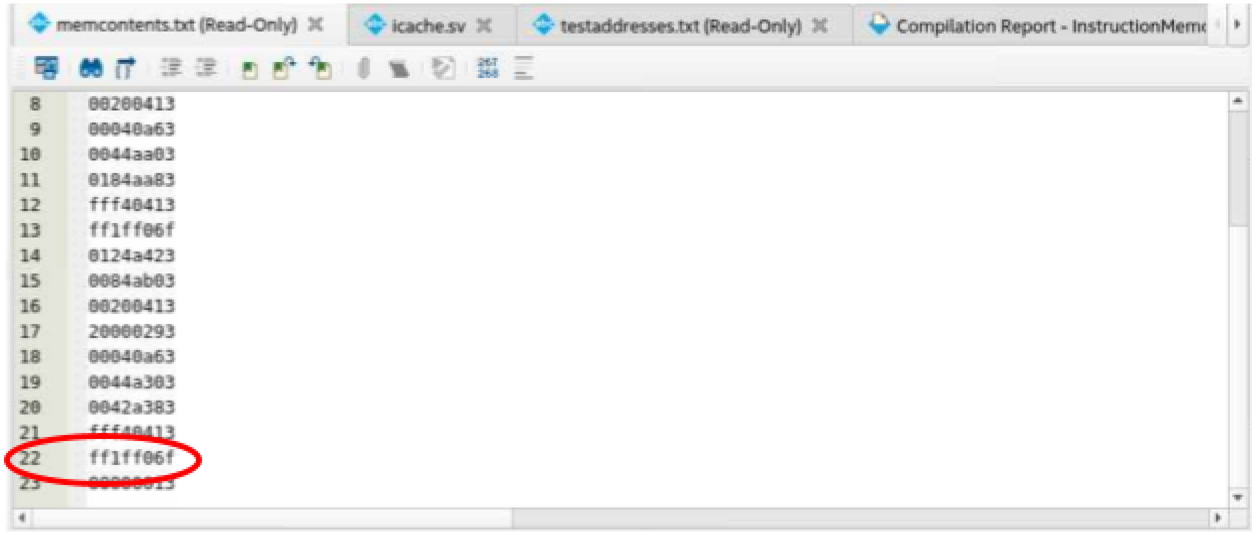


```
31      00000034
32      00000038
33      0000003c
34      00000040
35      00000044
36      00000048
37      0000004c
38      00000050
39      00000054
40      00000044
41      00000048
42      0000004c
43      00000050
44      00000054
45      00000044
46      00000058
```

Figure 3 - The corresponding address to index 39

If I map address 0x54 to an instruction in the main memory, I get 21 in decimal. The process is 0x54 = 84 but because each instruction is mapped to addresses that are multiples of 4; i.e. address 0 is mapped to the first instruction, address 4 is mapped to the second instruction, address 8 is mapped to the third instruction, and so on, 84/4 = 21. Thus, I just need to find the instruction that corresponds to the 21st location in the instruction array, which corresponds to an instruction of ff1ff06f as seen in **Figure 4**.

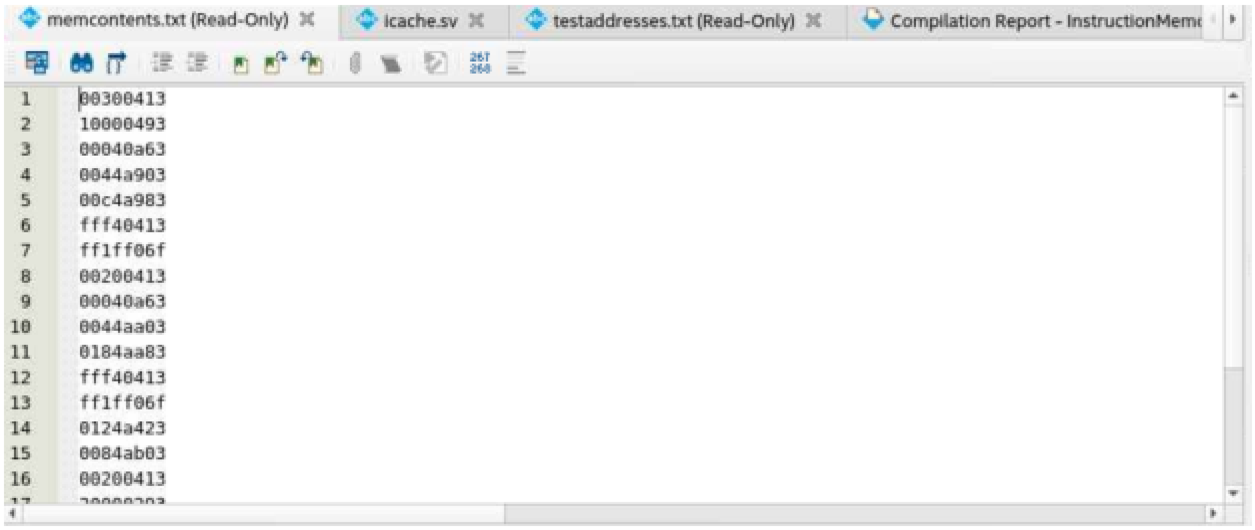
David Young
520648


memcontents.txt (Read-Only)
icache.sv
testaddresses.txt (Read-Only)
Compilation Report - InstructionMem

```
8   00200413
9   00040a63
10  0044aa03
11  0184aa83
12  fff40413
13  ff1ff06f
14  0124a423
15  0084ab03
16  00200413
17  20000293
18  00040a63
19  0044a303
20  0042a383
21  fff40413
22  ff1ff06f
```

Figure 4 - Instruction corresponding to the 21 elements of the array

Figure 4 shows the 21st element of the instruction array circled in red. It may seems strange that the row number next to it is 22 and not 21 but **Figure 5** illustrates why the 22nd row corresponds to the 21 element in the array.


memcontents.txt (Read-Only)
icache.sv
testaddresses.txt (Read-Only)
Compilation Report - InstructionMem

```
1   00300413
2   10000493
3   00040a63
4   0044a903
5   00c4a983
6   fff40413
7   ff1ff06f
8   00200413
9   00040a63
10  0044aa03
11  0184aa83
12  fff40413
13  ff1ff06f
14  0124a423
15  0084ab03
16  00200413
```

Figure 5 - Shows that the array begins at 1 and not 0.

Figure 5 shows us that row 1 corresponds to the first element of the instruction array, however, that first element in the array should correspond to an index of 0, meaning that if I want to find which element in the instruction array corresponds to an index I just need to subtract 1 from all of the row numbers seen in **Figure 5**. As a result, the instruction in row 22 of **Figure 4** corresponds to the 21st element. Long story short, if I check the simulation, it indeed is ff1ff06f which is what I expected.