**EECE 2323 Digital Logic Design Lab Report**

Lab #5 Adding Data Memory to the Datapath

Kaitlyn O’Flaherty Section #: 1

Emad Aboelela

Keshav

1. ***Background and Purpose***

The purpose of this lab was to add a new type of storage, data memory, to our datapath. The data memory is much larger and slower, but can hold many more locations than the register file created in lab 4. The memory holds 256 data words. The processor is a RISC, so the values are loaded from memory to register and stored to memory from register. The results of the ALU will be stored in the register before they are stored in the memory. The ALU is used to calculate the memory address but not the data itself when used in load and store instructions.

This lab is important to the scientific community because each processor needs to be able to store values in its memory. This allows the processor to hold and access much more information, and without memory a computer would not be able to hold all of the values in just the registers.

***2. Prelab***

***2.1 - Questions***

| Register | Value (binary) | Value (signed decimal) |
| --- | --- | --- |
| 0 | 00000010 | 2 |
| 1 | 11111011 | -5 |
| 2 | 00010101 | 21 |
| 3 | 00010111 | 23 |

1. Explain what these tests do functionally, (e.g. load Reg[1] from Mem[1], store value x in Mem[4] memory loc etc.)

1. reset

sets all values in the registers to 0

2. store values 2 and 4 to Mem[0] and Mem[1]

sets the value of memory address 0 to equal 2 and value of memory address 1 equal to 4

3. Load Mem[0] to Reg[0]

The value of register 0 is now equal to the value of memory address 0 (2).

4. Load ~Mem[1] to Reg[1]

the value in memory address 2 (4) is bitwise inverted (now = -5). The value of

register 1 is now equal to the new value (-5).

5. Write the value 8’h15 to Reg[2]

The value of register 2 is changed to 8’h15 (21).

6. store the value of (Reg[0] + Reg[2]) to Reg[3]

The value of register 0 (2) and register 2 (21) are added. Register 3 is now equal

to their sum (23)

7. store Reg[2] and Reg[3] to Mem[2] and Mem [3]

The values of memory addresses 2 and 3 are equal to the values of registers 2

and 3, respectively (21, 23).

8. reset

The values of the register are now all set to 0.

***2.2 - Create Your Own sequence***

1. reset
2. store to memory addresses 0 and 1 the values 1 and -21
3. load the value of mem address 0 to register 0
4. load the value of mem address 1 and use ALU inv operation to bitwise invert. store result in reg 5.
5. write the value 4 to reg 2
6. add reg 0 and reg 2, store in reg 3.
7. left shift the value of reg 2 by 2 and store the new value in reg 4
8. branch if not equal reg 5, reg 4
9. store contents of reg 5 and reg 4 in memory addresses 5 and 4 respectively.
10. clear the regfile

* expected values:

| Register | Value(binary) | Value (decimal) |
| --- | --- | --- |
| 0 | 00000001 | 1 |
| 1 | 11101011 | -21 |
| 2 | 00000100 | 4 |
| 3 | 00000101 | 5 |
| 4 | 00010100 | 20 |
| 5 | 00010100 | 20 |

* The ALU operations not tested are: bitwise OR, bitwise AND, arithmetic shift right, branch if not equal.
* functionality:

1. reset
   1. sets all values in the registers to 0
2. store values 1 and -21 to Mem[0] and Mem[1]
   1. sets the value of memory address 0 to equal 1 and value of memory address 1 equal to -21
3. Load Mem[0] to Reg[0]
   1. The value of register 0 is now equal to the value of memory address 0 (1).
4. Load ~Mem[1] to Reg[1]
   1. the value in memory address 2 (4) is bitwise inverted (now = -5). The value of register 1 is now equal to the new value (-5).
5. Write value 8’h15 to Reg[2]
   1. The value of register 2 is changed to 8’h15 (21).
6. store the value of (Reg[0] + Reg[2]) to Reg[3]
   1. The value of register 0 (1) and register 2 (4) are added. Register 3 is now equal to their sum (5)
7. Shift Reg[2] left by two, store in Reg[4]
   1. the value of register2 is shifted left and the new value (20) is stored in register 4
8. beq Reg[4] and Reg[5]
   1. bnq is true(1) if the values of Reg[4] and Reg[5] are not equal. (statement is false because values reg[4]=20=reg[5]).
9. store Reg[5] and Reg[4] to Mem[5] and Mem [4]
   1. The values of memory addresses 5 and 4 are equal to the values of registers 5 and 4, respectively (20, 20).
10. reset
    1. The values of the register are now all set to 0.

***3. Summary of Design Implementation***

***3.1 Results and Analysis***

***3.2. Conclusions and Recommendations***

A conclusion that can be made after completing this lab is that the clock is an essential part of a sequential logic circuit because it provides the register file with a sense of time to determine when the data input will be looked at. Another conclusion is that the zero register helps make the register more efficient because it can implement a “clear” instruction which could be used to initialize or reset register values.