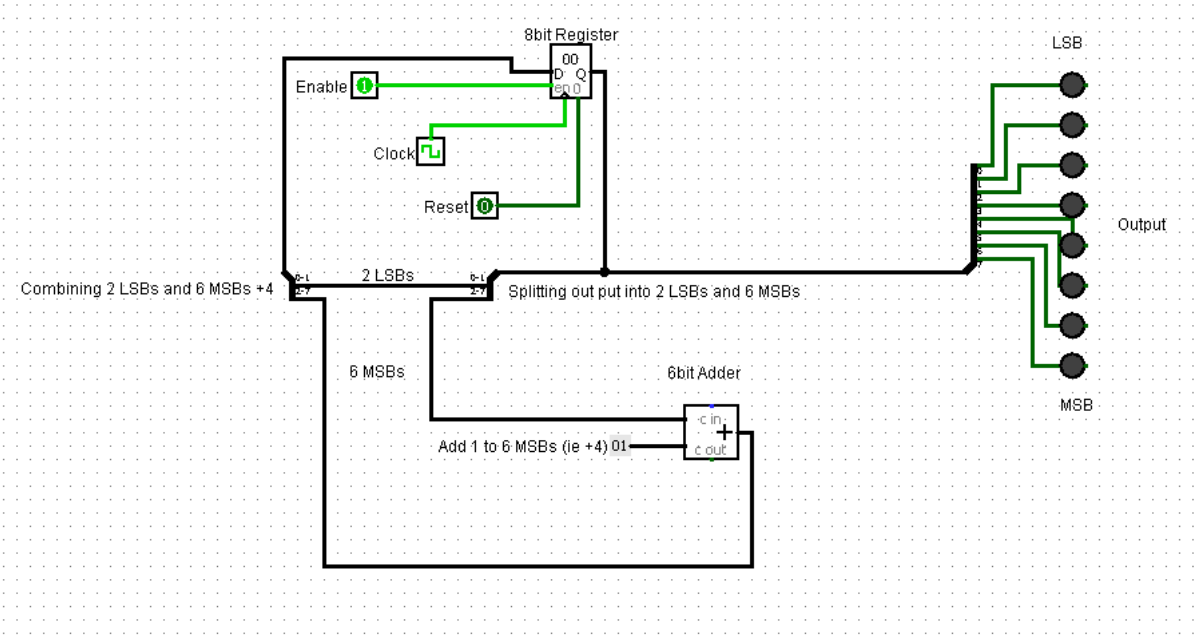
COMP30080 Processor Design

Assignment 4 Fergal Lonergan

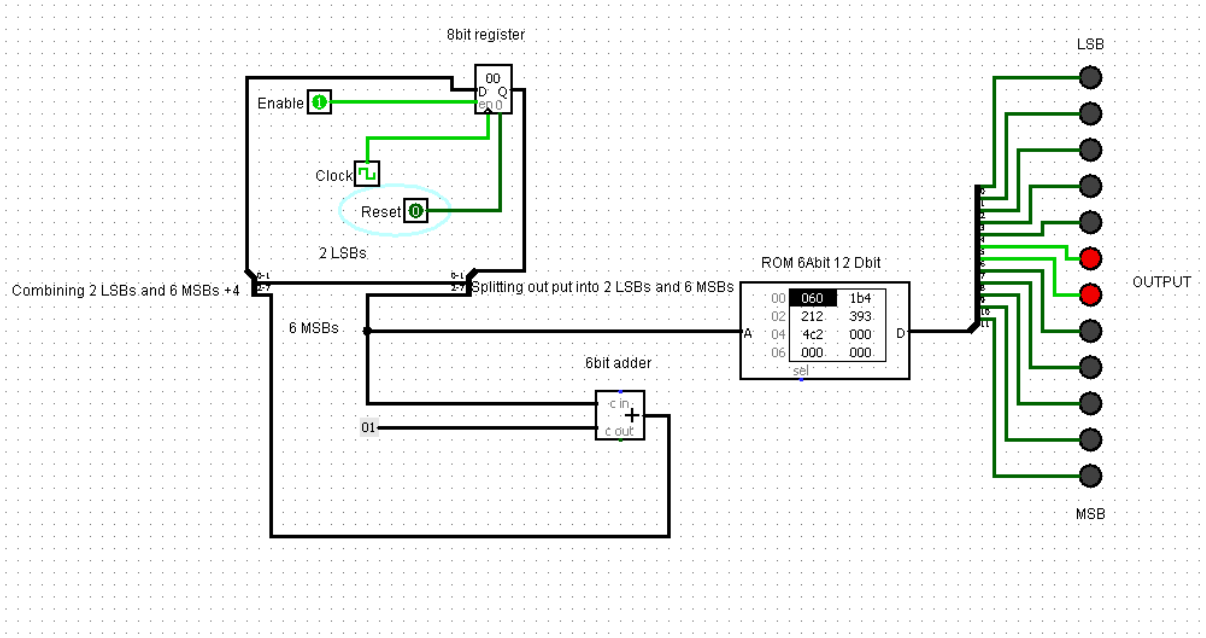
Program Counter (PC)



My Program Counter (PC) was designed using an 8 bit register and a 6 bit adder. The PC increments by 4 each time, so it was unnecessary to change the 2 Least Significant Bits (LSBs) so I spilt my output and only the 6 Most Significant Bits (MSBs) go into my adder. They are then added with a 6 bit constant of value 0x000001. This new number is then combined back with the two LSBs and entered back into our register. $100_2 = 4_{10}$. I also have an enable which when set to run will allow the value of my register to change, a clock signal that triggers my register on the positive edge to change to the new added value, and a reset button which wen set to 1 sets the value of my register to 0.

I then display the output of my register using the LEDs marked output.

PC&ROM



The next task was to add a 6bit address, 12bit data output ROM into our PC circuit. This involved taking our PC circuit from before and taking the 6 MSB from our 8bit register and inputting them as the address to our ROM. I also had to increase our output to 12 as our instructions are 12 bits long.

To come up with the instructions that were to be saved in my ROM I used the following information given to us in the Assignment sheet. I assumed that the binary values for the registers were as follows:

$zero	00
$t0	01
$t1	10
$t2	11

Register Set
$zero, $t0, $t1, $t2

Instruction Set Architecture

```
lw   rt, offset(rs)      # load word from M[rs+offset] to R[rt]
sw   rt, offset(rs)      # store word from R[rt] to M[rs+offset]
addi rt, rs, const       # add immediate R[rt] = R[rs] + const
add  rd, rs, rt          # add R[rd]=R[rs]+R[rt]
sub  rd, rs, rt          # add R[rd]=R[rs]-R[rt]
```

Instruction Formats
{opcode; rs; rt; rd/offset/const}
opcode = 4 bits; rs = 2 bits; rt = 2 bits; rd/offset/const = 4 bits

Opcodes

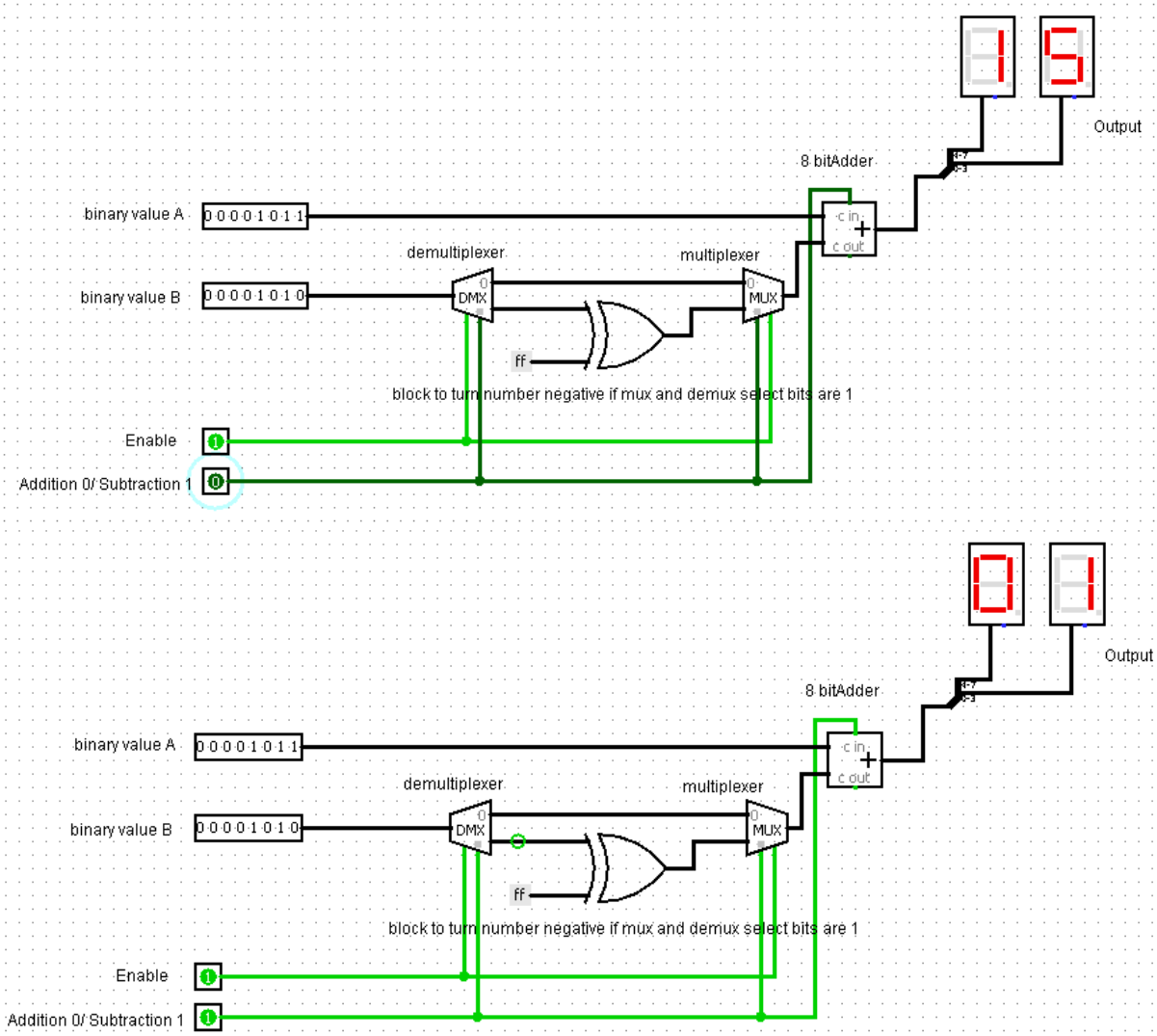
```
lw   : opcode = 0000
sw   : opcode = 0001
addi : opcode = 0010
add  : opcode = 0011
sub  : opcode = 0100
```

I then made 5 instructions using all five operations given to us in the assignment and saved them into my ROM in the first 5 data slots. They are saved in Hexadecimal format which is why I have converted my values below to hex.

- lw $t1, 0($t0) , $0000\ 01\ 10\ 0000_2$ $0x060_{16}$
- sw $t2,4($t1), $0001\ 10\ 11\ 0100_2$ $0x1B4_{16}$
- addi $t0, $zero, 2 $0010\ 00\ 01\ 0010_2$ $0x212_{16}$
- add $t2, $t1, $t0 $0011\ 10\ 01\ 0011_2$ $0x393_{16}$
- sub $t0, $t2, $zer0 $0100\ 11\ 00\ 0010_2$ $0x4C2_{16}$

With each positive edge of our clock our ROM moves to a new instruction and outputs it on the LEDs.

ALU



The two images above are or the circuit diagram of my ALU. We were asked to design an ALU that could do 8bit addition and subtraction and then to show the corresponding output in a hex display. I have 2 8bit numbers labelled binary value A and B. The enable must be on for the circuit to work. If the addition/subtraction pin is set to 0 then the select pin of my de-multiplexer and multiplexer is 0 and so my binary value B is selected by both. This is then passed into an adder with number A where

my carry in is zero so they just add the two numbers together and display the answer on the hex display.

If the addition/subtraction pin is set to 1 then both my de-multiplexer and multiplexer's select bits are 1. This then passes the number B into our XOR with -1 so we get the twos compliment of the number. (ie the number plus 1 all multiplied by -1. i.e. if number was 6 it would now be -7). This is then passed into our added with number A as well as our carry bit of one (which returns the 2s compliment number to be just itself multiplied by negative 1) and gives us the answer of A minus B displayed on the hex display.