Joshua Pollock

EE 310 – Lab 6 Report

NAU, 12 April 2020

Problem Description

In this lab, we have been asked to add the sub instruction to the supplemental files. This is detailed in section 6.3 of Zybooks but a simplified run-down/guide was provided to us in the lab report. In this lab the use of offsets is allowed but loops and subroutines are not. Once adding the sub instruction to the instruction set, we were tasked with creating a MIPS assembly file that completed the operations seen in Figure 1. Once we verify our code is fully functional, we were tasked with editing lines 31 and 32 to have random numbers.

Before the program run:

```
DM[5000] = x
DM[5004] = y
```

After the program run:

```
DM[5000] = y-x
DM[5004] = 2y-x
DM[5008] = 3y-x
DM[5012] = 3y-2x
DM[5016] = 4y-x
DM[5020] = 4y-3x
DM[5024] = 5y-x
DM[5028] = 5y-3x
```

Figure 1. Expected behavior of the circuit

Solution Plan

In order to solve the problem explained above, I used the supplmental code provided to set up the project. Once the project was set up, following the instructions within the lab I modified the mipzy_control, adder_32, and MIPSzy files to all include the sub instruction. This was straight forward and just required following along with the pictures provided.

Next, we were tasked with creating a program that would perform simple addition and subtraction. These operations were given to us in the lab report and while they do contain multiplication operations, we can easily just use the add instruction to produce the same effect.

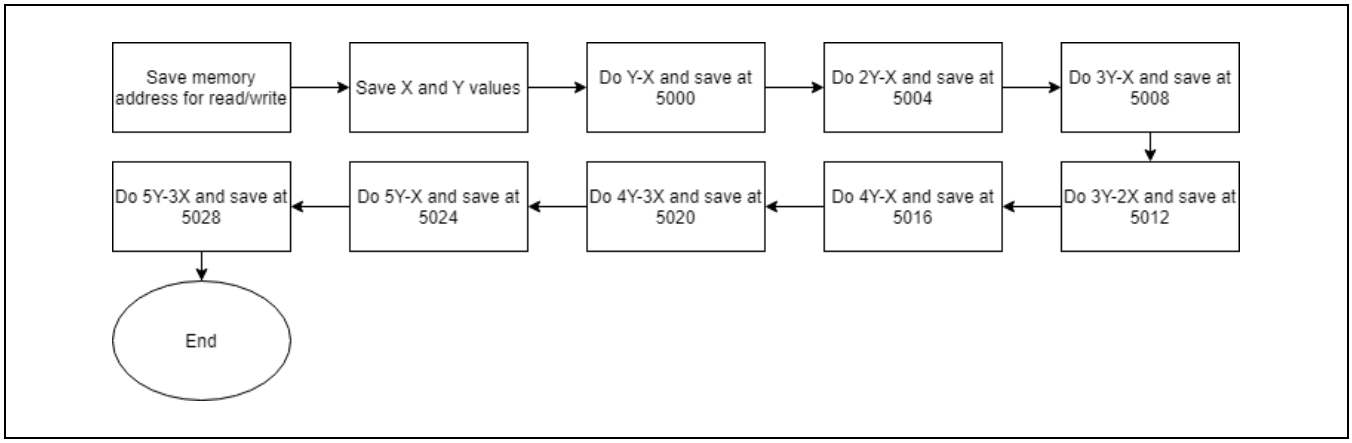

Figure 2. State diagram for the proposed solution

Implementation and Test Plan

I have implemented the solution plan explained above, by first adding the sub instruction to the MIPSzy instruction set. This was easy and straightforward as we were given picture guides and told how to do it. Next, I simply created a basic MIPS file that did the operations requested in Figure 1. Register t3 acted as my running Y variable. This allowed me to keep register t2 set as the original Y value and add that to $t3 when a multiplication operation took place (y -> 2y -> 3y etc…). Register t5 became my result register. This is where I would store the result of the operations. Lastly, t4 was my running X register. It operated in a similar way to register t3.

Lab7.txt

```
addi $t0,$zero,5000          # Memory Pointer
lw      $t1,0($t0)                  # X
lw      $t2,4($t0)                  # Y
#y-x
sub $t3, $t2,$t1             #y-x
sw      $t3,0($t0)
#2y-x
add $t3,$t2,$t2                     # y+y
sub $t5,$t3,$t1              # 2y - x
sw  $t5,4($t0)
```

```
#3y-x
add $t3,$t3,$t2                 # 2y + y
sub $t5,$t3,$t1             # 3y - x
sw  $t5,8($t0)
#3y-2x
add $t4,$t1,$t1                 # 2x
sub $t5,$t3,$t4             # 3y-2x
sw  $t5,12($t0)
#4y-x
add $t3,$t3,$t2                 # 3y + y
sub $t5,$t3,$t1             # 4y-x
sw  $t5,16($t0)
#4y-3x
add $t4,$t4,$t1                 # 2x+x
sub $t5,$t3,$t4             # 4y-3x
sw  $t5,20($t0)
#5y-x
add $t3,$t3,$t2                 # 4y + y
sub $t5,$t3,$t1             # 5y-x
sw  $t5,24($t0)
#5y-3x
sub $t5,$t3,$t4             # 5y-3x
sw  $t5,28($t0)
```

Figure 3. Verilog code for the proposed solution

```
test_numbers[0] = 3 ;  // Number to be stored in DM[5000]
test_numbers[1] = 1 ;  // Number to be stored in DM[5004]
```

```
# Data at address 5000 is          -2
# Data at address 5004 is          -1
# Data at address 5008 is           0
# Data at address 5012 is          -3
# Data at address 5016 is           1
# Data at address 5020 is          -5
# Data at address 5024 is           2
# Data at address 5028 is          -4
# ** Note: $stop    : I:/#EE/EE 310/Lab 7/MIPSzy_TB.v(85)
#    Time: 2180 ns  Iteration: 0  Instance: /MIPSzy_TB
# Break in Module MIPSzy_TB at I:/#EE/EE 310/Lab 7/MIPSzy_TB.v line 85
```

First run with the default numbers set

```
test_numbers[0] = 6 ;  // Number to be stored in DM[5000]
test_numbers[1] = 5 ;  // Number to be stored in DM[5004]
```

```
# Data at address 5000 is          -1
# Data at address 5004 is           4
# Data at address 5008 is           9
# Data at address 5012 is           3
# Data at address 5016 is          14
# Data at address 5020 is           2
# Data at address 5024 is          19
# Data at address 5028 is           7
# ** Note: $stop    : I:/#EE/EE 310/Lab 7/MIPSzy_TB.v(85)
#    Time: 2180 ns  Iteration: 0  Instance: /MIPSzy_TB
# Break in Module MIPSzy_TB at I:/#EE/EE 310/Lab 7/MIPSzy_TB.v line 85
```

Second run with random numbers set. X=6 Y=5

Figure 4. Lab pictures of the running solution