Steven Reeves

CST 250

4/14/18

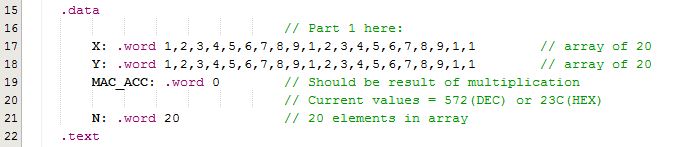
Lab 2 Report

**Introduction:**

The objective in this lab was to familiarize us with the basic syntax in the MIPS assembly language and practice designing/implementing algorithms. Some specific skills to be learned include: memory allocation, read/write syntax, indirect addressing, array processing, arithmetic manipulation, and iteration/looping.

**Part 1:**

This part of the lab had me lay down the variables to be manipulated in the main program. After some research on how to initiate these variables I ended up with the following:



X – A 20 element array with each element having the size of a “word” (32 bits in MIPS)

Y – A 20 element array with each element having the size of a “word” (32 bits in MIPS)

MAC\_ACC – A single “word” to hold the result of the multiplication, initialized to 0

N – A single “word” to hold the number of elements in the array

X, Y, and N act as the “input” to this program. They can be changed by the user before execution.

MAC\_ACC acts as the “output” of this program. It should contain the sum of the multiplication of each X[i] and Y[i] element.

**Part 2:**

This part of the lab had me design an algorithm to do the following:

* Loop n times
* Check for X[i] and Y[i] not equal to 0
* Each loop, multiply X[i] and Y[i]
* After multiply, add result to MAC\_ACC
* Terminate in infinite loop for debugging

My first step here was to design basic looping statements with the following functionality:

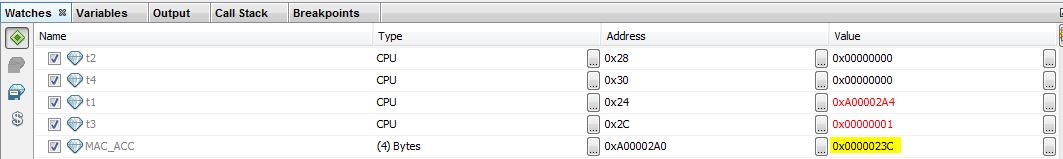
* Make sure outer loop is working
* Make sure address of head of X and Y are stored
* Make sure to add result of multiplication in MAC\_APP
* Make sure inner loop is working

I first implemented this functionality in a test driven development fashion with the following steps:

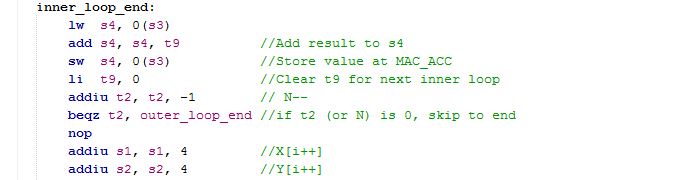
* Outer loop
  + Load X,Y, and N values
  + Loop N times and increment Y
  + Confirm number of loops is correct for different values of N by adding a watch to Y
* Inner loop
  + Load X and Y values
  + Loop Y times and add X to temp register
  + Confirm value of temp register with a watch on temp register
* Nest inner loop in outer loop
  + Confirm temp value in inner is correct and calculated N times
* Add results of temp into MAC\_APP every outer loop
  + Confirm value is correct for each loop
* Array functionality
  + Go to next element in array for each outer loop
  + Confirm elements have been moved by looking at result of multiplication
* Full functionality
  + Add 20 elements and confirm sum of multiplications
  + Confirm value of MAC\_APP with arithmetic checking

**Conclusion:**

The biggest problem I had in this process was syntax. There was a lot of looking up commands to make sure I was using the right “add” and right load commands. Practice helped out a lot here. Another problem I had was realizing the debugger’s values were in hex. For example: In my final test the result is 572(DEC). Seeing 23C(HEX) in the debugger threw me off at first.



Another problem worth mentioning is adding bytes to the address of the head of the array to step through it. I initially tried adding one byte and ended up in the middle of next element. After adding 4 instead of 1, I ended up at the correct element.



In summary, here is a list of things I learned/practiced in this lab:

* Setting and using watches on registers in IDE
* Building and running with breakpoints in IDE
* Algorithm design
* Test driven development
* Variable allocation
* Label use
* Read/write implementation
* Indirect addressing use
* Array processing
* Arithmetic use with MIPS syntax
* Iteration and loop use
* Debugging in the IDE