

University of California, Santa Barbara  
Dept. of Electrical and Computer Engineering  
ECE 15B – Computer Organization  
Homework #2 – SPIM + Basic Assembly

**SPIM**

The following questions are based on FIR\_Filter.txt file  
([http://www.ece.ucsb.edu/~kastner/ece15b/homework/FIR\\_Filter.s](http://www.ece.ucsb.edu/~kastner/ece15b/homework/FIR_Filter.s)). Start by loading this file then answer the following questions:

1. Which of the windows is the register? Memory? Instruction?
2. What's the address of the first instruction in your code (addi \$sp, \$sp, -4)?  
0x00400024
3. What's the content of registers a0, a1, a2 when the instruction "jal FIR" is executed? (For extra credit, explain why)

0x10010000 ; 0x10010054 ; 0x10010064  
These are the parameters passed in to the function

4. What is the beginning address of in\_array? csnt\_array? (Hint: the instruction *la* takes two arguments; the first is the destination register for the address from the second argument.

0x10010000 ; 0x10010054

5. What's the value of ra AFTER the jal FIR instruction is executed? (For extra credit, explain why)

00400044 ; because that's the address we're jumping back to after the function is returned

Set a breakpoint at instruction 0x00400088 and run the program until you hit that breakpoint:

6. To which instruction does this breakpoint correspond?

7. What is the value of t0? 0x50

8. What is the value of t4? 0x10010004

9. Why is t3 significantly larger than t0?

Because t3 holds an address and t0 holds a value

10. What's the content of the address that t3 points to? 1

Set another breakpoint at instruction 0x004000b0 and run the program until you hit this breakpoint:

11. What's the value of t3? t5? 1001005c, 1

12. What's the value that t3 is pointing to? 3

Run through 5 breakpoints (NOTE: first encounter after you resume is the same one as you set before so do not count it!)

13. Which breakpoint did you encounter first? 0x00400088 or 0x004000B0? why?  
0x004000b0, because it's still inside the miniloop

14. How many breakpoints will it hit before it hits the 0x00400088 breakpoint? why?  
2 breakpoints and will hit 0x00400088 on the 3rd, because the program runs through the miniloop 3 times (3 values in cnst\_array)

15. After 5 breakpoints, what's the value of t3? t5? 10010064 ; 1

### **Simple MIPS Assembly**

16. Translate the following code into MIPS assembly:

$$A = B - C - 30 + D - E + F$$

Assume that variables A, B, C, D, E, F are in registers \$1, \$2, \$3, \$4, \$5, \$6.  
Also, assume that you cannot overwrite variable, A, B, C, D, E, F since they will all be used later in the program. Finally, assume that any ADD or SUBTRACT operation takes 1 cycle.

- a. Write the code such that it uses the minimal number of registers. How many additional registers (other than \$1-6) does your code require? How many cycles does your code need?

```

sub $1, $2, $3      # A = B - C
addi $1, $1, -30   # A = B - C - 30
add $1, $4, $1     # A = B - C - 30 + D
sub $1, $1, $5     # A = B - C - 30 + D - E
add $1, $1, $6     # A = B - C - 30 + D - E + F

```

No additional registers are needed. You can store the intermediate results in \$1.

The code requires 5 cycles

- b. Assume that the processor can perform one ADD operation and one SUBTRACT operation during every cycle. Rewrite the code to take advantage of this. How many cycles does your code need? How many additional registers (other than \$1-6) does your code require?

One solution is to rewrite the code as such:  $A = (B - (C + 30)) + ((D - E) + F)$

```

addi $1, $3, 30    # A = C + 30
sub $7, $4, $5     # $7 = D - E
sub $1, $2, $1     # A = B - (C + 30)
add $7, $7, $6     # $7 = D - E + F
add $1, $1, $7     # A = B - C - 30 + D - E + F

```

The code takes 3 cycles to execute (the first two sets of add/sub can execute in parallel in the same cycle).

The code requires one additional register (\$7)

17. Translate the following code into MIPS assembly:

$$A[i] = B[i-1] + B[i] + B[i+1]$$

Assume that  $A$  and  $B$  are byte arrays. Both arrays are stored in memory. Register \$1 contains the initial address for array  $A$ ; register \$2 contains the initial address for array  $B$ ; and register \$3 contains the value of  $i$ .

```

add $6, $2, $3     # $6 = &B[i]
lb $4, -1($6)     # $4 = B[i-1]
lb $5, 0($6)      # $5 = B[i]
add $5, $5, $4    # $5 = B[i-1] + B[i]
lb $4, 1($6)      # $4 = B[i+1]
add $5, $5, $4    # $5 = B[i-1] + B[i] + B[i+1]

```

```
add $6, $1, $3          # $6 = &A[i]
sb $5, 0($6)           # A[i] = B[i-1] + B[i] + B[i+1]
```

Assume that load (lb) and store (sb) take 3 cycles each.

How many cycles does your code take? How many additional registers does your code require?

$4 * 3 + 4 * 1 = 16$  cycles and 3 additional registers.