CSC 3210 Computer Organization and programming

LAB 6

INSTRUCTIONS FOR LAB 6

Tasks to be done in this lab:

- Use the data section for predefined values
- Load and move integer data in registers in assembler
- Perform add, subtract, NOT, NEG operations
- Perform a subtract operation with an add command

ASSIGNMENT FOR LAB 6

- Before we begin this Lab, refer to the previous lab to get a quick reminder on commands that we used.
- You should also have a clear understanding of how we print the values in Venus with the "ecall".
- This lab has five parts. Run the code for all the parts.
- Answer the questions from all the parts mentioned in bold.
- Turn in a copy of your code, and turn in your output as a text file. If iCollege gives you trouble about turning in your code, rename it to a .txt extension, and it should work. Remember to submit the lab report as it is vital for grading.

PART-1

- You may want to create a new directory, e.g.

  cd RISC_V

  mkdir lab6

  cp ../hello_world/HelloWorld.S lab6_pt1.S

- You can copy the code from the previous lab to a new file, such as "lab6_pt1.S".
- First, load integer values from the data section into two registers, add them together, and store the result in another location in the data section.
- Then, print the text "Add result " followed by the result. Here is an example of how the data section might look.

```
# Data section, initialized variables

.data
str1:   .string "Add result "   # String to use
str2:   .string "Sub result "   # String to use
str3:   .string "Not result "   # String to use
str4:   .string "Neg result "   # String to use
hexstr: .string "%x", 10        # String format to use (hex), followed by NL

int1: .word 12
int2: .word 5
int3: .word 1000
sum:  .word 0
```

PART-1

- Next, load the integer values into two registers, perform addition, print the text "Add result " followed by the result.
- Loading an integer value into a register can be done with one of the following commands.
- You will need to put the address into x7 first, as the following loads the address of "int2" into register x7.

  la x7, int2

- Then we can load the value stored at the address that x7 has, depending on the size of the value.
- The command to load from an address allows for an offset, such as address + 1 but we do not want an offset, so we use 0.

  lb x6, 0(x7) # load byte

  lh x6, 0(x7) # load halfword

  lw x6, 0(x7) # load word

- Note that x6 and x7 are placeholders; you can use other registers.

PART-1

- You can move (mv) a value from one register to another, as follows.

  mv a1, s2 # a1 = s2

- This is very useful for holding on to one register's contents and printing it out.
- Since we use a0 and a1 in printing, we need to make sure that we do not overwrite a value that we will use later.
- In other words, imagine that you have the result of an arithmetic operation in a0.
- Then you decide to print a string before you print a0's value, so you use li a0, 4 before the ecall.
- This overwrites the result that you plan to print!

PART-1

Questions:

1. **What would you need to change in the program to work with bytes instead?**
2. **What would happen if you have a large value, such as int3, and load it as a byte? Try it, then explain why the value is what it is.**
3. **What would you need to change in the program to work with halfwords instead?**
4. **Try using la x7, int1 followed by lw x6, 1(x7). What value do you get in x6? Why?**
5. **We have "sum" defined in the data section. What command(s) would we use to put the result (say, of an add command) into it?**

PART-2

- Copy your code to a new file, "lab6_pt2.S".
- Repeat what you did in part 1, only use a subtraction (sub) instead.
- Make sure to use the correct string to print the value.
- That is, perform subtraction, print the text "Sub result " followed by the result.

Question:

1. **Did you get the result that you expected?**

PART-3

- Copy your code to a new file, "lab6_pt3.S".
- Then reverse the operands of the subtraction, i.e. instead of finding "int1 - int2", find "int2 - int1".
- However, this time print the operands and operators, i.e. print

  5-12=-7

- Do not hard-code this: do not define a string as "5-12=-7". Instead, print the first operand (5), then print the "-" (as a character), then print the second operand (12), then print the "=", then print the result, and finally print a new-line character.
- You may need to find an ASCII chart to know the character codes. It's not difficult to do this, but it requires a bit of patience! Make sure that this works with the values defined in the data section, so that if we change int1's or int2's value, that it prints the correct values.
- To make sure that it works, you should change the values defined in the data section and demonstrate that it works.

PART-3

Questions:

1. **How many lines of code did it take to load the value(s)?**
2. **How many lines of code did it take to do the arithmetic?**
3. **How many lines of code did it take to print everything?**
4. **What do you observe about the code in terms of number of lines devoted to loading/arithmetic/printing, and repetition?**

PART-4

- Using the same integer value, NOT the first one and print the result.
- That is, print "Not result" then print the value.

```
# Not s0's value: Perform not operation
not   s2, s0
```

- Repeat this, but use the NEG command instead.
- Use the same integer value, but NEG the first one and show the result.
- We can use "neg" command to negate the value as follows:

```
# Neg s0's value: Perform negation
neg   s2, s0
```

PART-4

Questions:

1. **How did the computer get the result? That is, what did it do to the original value to get the value printed?**
2. **Why did we skip loading a value into s1 for these commands?**
3. **Can you tell what the difference is between a simple NOT operation and 2's complement on a data?**

PART-5

- Use "neg" on int1's value, then add it to int2's value and show the result. Here is some example code.

```
# Get the value at int1 into s0
    la    x7, int1
    lw    s0, 0(x7)
# Get the value at int2 into s1
    la    x7, int2
    lw    s1, 0(x7)
neg   s2, s0
add   s2, s2, s1
```

Question:

1. **We have used neg command to subtract a value from the other. Can we use NOT operation for the same purpose? Why or why not?**

IMPORTANT NOTE:

Remember that we will grade your lab report so it is vital to turn that in. The other files (your code, a text version of any log file, etc.) are to document your work in case we need more information.