**LAB REPORT NO 03**

****

**Computer organization and Architecture (CSE 304L)**

**Fall 2024**

**Submitted By: Naveed Ahmad**

**Registration No: 22PWCSE2165**

**Section: B**

Date: 02/12/2024

**Submitted To:**

**Dr. Ammad Khalil**

**Department of Computer Systems Engineering**

**University of Engineering and Technology**

**TASK 1:**

1. Create a directory to hold the files for this lab.
2. Launch your favorite editor and type the following program. Note that assembly language is free-form but it is a good idea to align the four fields (label, instruction, operand, comment) in order to enhance the program's readability:

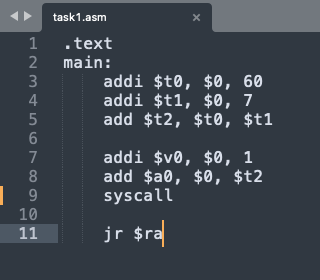
.text  
main: #---------------------------------------

addi $t0, $0, 60  
# t0 = 60 addi $t1, $0, 7  
# t1 = 7 add $t2, $t0, $t1  
# t2 = t0+t1 #-----------------------------------------

addi $v0, $0, 1  
# service #1 add $a0, $0, $t2  
# printInt syscall  
# do print #----------------------------------------- jr $ra # return

Save the program under the name: LabA1.asm.

**CODE PIC:**

****

**OUTPUT PIC:**



**TASK 2:**

1. Modify the program by adding a directive before the first statement and a label for the last statement:

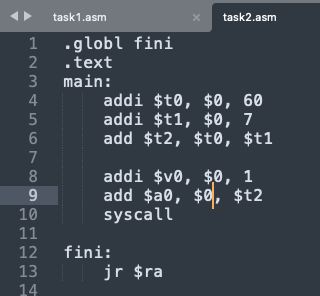
.globl fini

.text  
main: #---------------------

...  
fini: jr $ra # return

Save the modified program as: LabA2.asm.

**CODE PIC:**

****

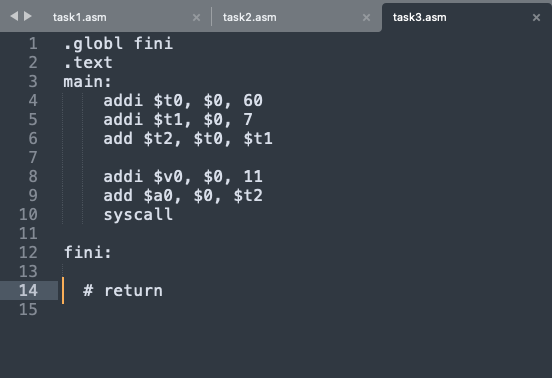
**OUTPUT PIC:**



**TASK 3:**

1. Our program printed the output using service #1, which prints integers. Modify the program so it prints using service #11, which interprets and prints the contents of $a0 as a character. Save the modified program as: LabA3.asm. Run it and explain why the output became C.

**CODE PIC:**

****

**OUTPUT PIC:**

****

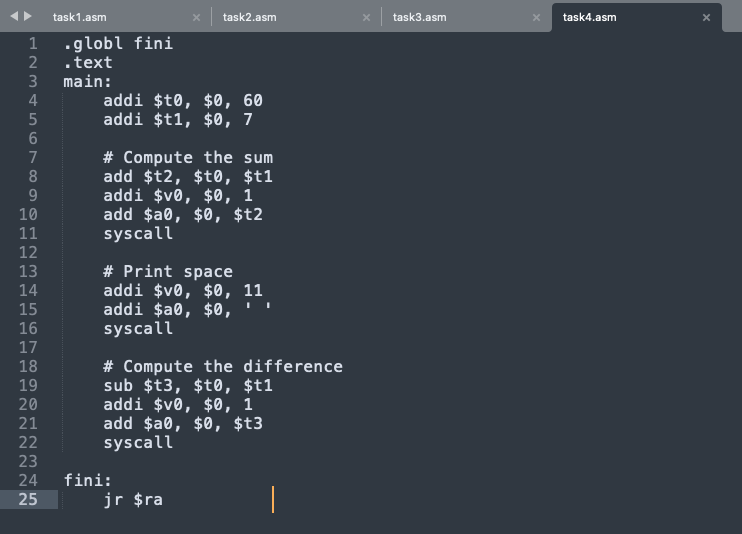
**TASK 4:**

1. Revert back to LabA2.asm and save it as LabA4.asm. Modify it so that it prints the sum and the difference of $t0 and $t1 separated by a space. Use the sub instruction to subtract registers. In order to print a space delimiter, use service #11 with $a0 being the space character:

addi $a0, $0, ' '  
(Alternatively, you can store 32 in $a0.) Run and verify that the output becomes:

67 53

**CODE PIC:**

****

**OUTPUT PIC:**

****

**TASK 5:**

1. Revert back to LabA2.asm and save it as LabA5.asm. Rather than hard-coding the numbers in our program, let us read one of them from the user by using service #5, readInt. Replace the statement:

addi $t0, $0, 60 # t0 = 60 with:

addi $v0, $0, 5 # v0 = readInt syscall  
add $t0, $0, $v0

Notice that you must set $v0 to 5 prior to issuing syscall. Afterwards, the entered integer is returned to you in $v0. We copied the return to $t0 so that the rest of the program can remain unchanged. Run the program and enter 10. Do you obtain the expected output?

**CODE PIC:**

****

**OUTPUT PIC:**

****

**TASK 6:**

1. Save LabA5.asm as LabA6.asm then modify it so that it processes the two numbers as follows:

if ($t0 = = $t1) {

Print ($t0 + $t1); }

else {

Print ($t0 - $t1); }

We can express this using a conditional transfer of control (a branch) and an unconditional one (a jump):

if ($t0 = = $t1) branch to XX; print ($t0 - $t1);  
jump to YY;

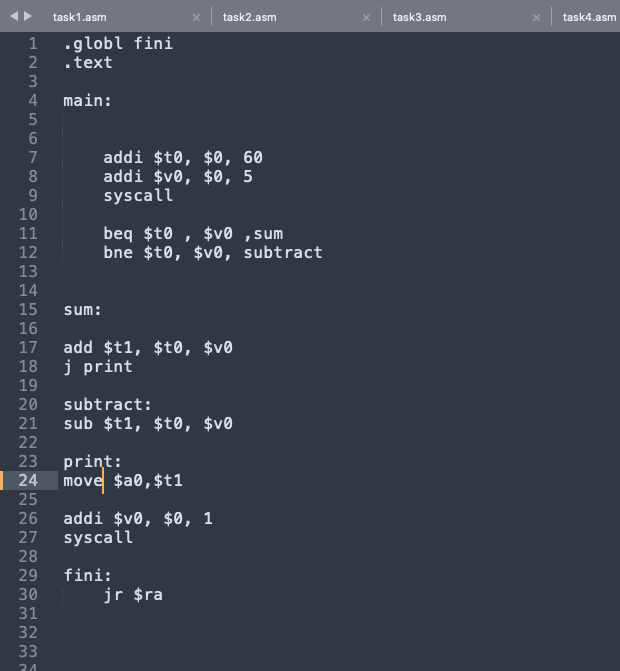
XX: print ($t0 + $t1); YY: rest of program

The first instruction in the above pseudo-code can be realized using: beq $t0, $t1, XX

The third instruction is realized using: j YY

Run the program and verify that it works as expected. Note that beq (branch-on- equal) is used for = = and bne (branch-on-not-equal) for !=.

**CODE PIC:**

****

**OUTPUT PIC:**

If $t0 and $v0 are equal.



If $t0 and $v0 are not equal.



**TASK 7:**

1. Save LabA6.asm as LabA7.asm then modify it so that it processes the two numbers as follows:

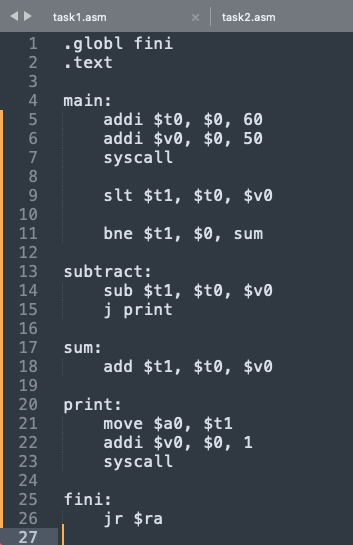
if ($t0 < $t1) {

Print ($t0 + $t1); }

else {

Print ($t0 - $t1); }

**CODE PIC:**

****

**OUTPUT PIC:**

****

**TASK 8:**

1. Start fresh and create the program LabA8.asm with the following body (between main and fini):

loop:

addi $v0, $0, 1

add $a0, $0, $0 slti $t9, $a0, 5

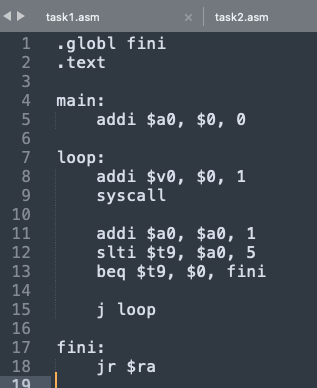
beq $t9, $0, fini syscall  
addi $a0, $a0, 1 j loop

It is important that you understand the program and attempt to predict its output before running. Save the program then run it and confirm or correct your prediction.

2. Replace the statement before last in the above program with: addi $a0, $0, 1

What will the output be in this case? Show that single stepping is ideally suited to debug such an error.

**CODE PIC:**

****

**OUTPUT PIC:**

****

**TASK 9:**

1. Start fresh and create the program LabA9.asm that operates as follows: int $s0 = 0; int $t0 = readInt (); for (int $t5 = 0; $t5 < $t0; $t5++) { $s0 = $s0 + $t5; } Print ($s0);

Note that there is no looping construct; you use branches and jumps. You can, for example, re-think the above loop as follows: loop:

$t5 = 0; if (! $t5 < $t0) branch to done; $s0 = $s0 + $t5; $t5++; jump to loop; print($s0); done:

Save the program then run it with 10 as input. Do you get 45 as output?

**CODE PIC:**

****

**OUTPUT PIC:**

****