**LAB NO. 03: Debugging Using Breakpoints, Single Step, Use of Loops & Conditional Statements**

**Objective:**

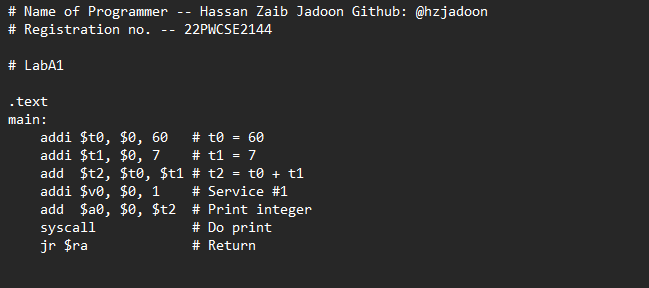
This lab assumes that you have an understanding of what assembly language is. It also assumes that you know the names of the general-purpose registers in the MIPS CPU.

**Lab Tasks:**

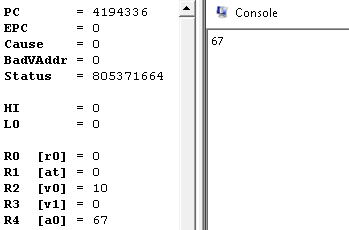
**Task 1:**

1. Create a directory to hold the files for this lab.
2. Launch your favourite 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:

**Code:**



**Output:**



**Task 2:**

* 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

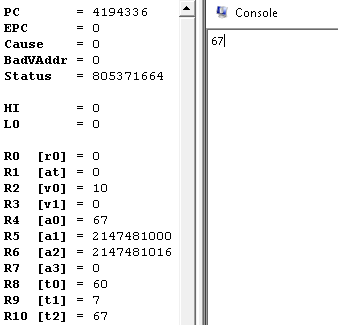
* Save the modified program as: LabA2.asm.

**Code:**

**A screenshot of a computer

Description automatically generated**

**Output:**

****

**Task 3:**

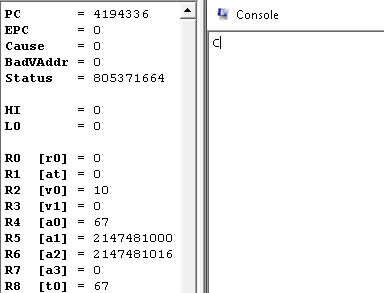
* Our program printed the output using service #1, which prints integers. Modify the program so it prints using service #11, which interprets and prints the contents of $a0 as a character. Save the modified program as: LabA3.asm. Run it and explain why the output became C.

**Code:**

**A screen shot of a computer

Description automatically generated**

**Output:**



**Task 4:** 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:

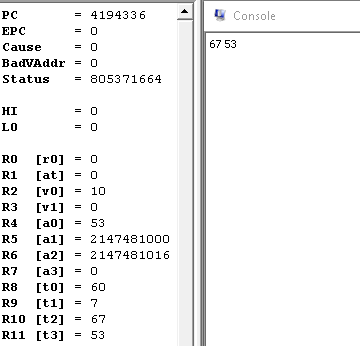
addi $a0, $0, ' '

**Code:**

**A screenshot of a computer screen

Description automatically generated**

**Output:**



**Task 5:**

* 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 # to = 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

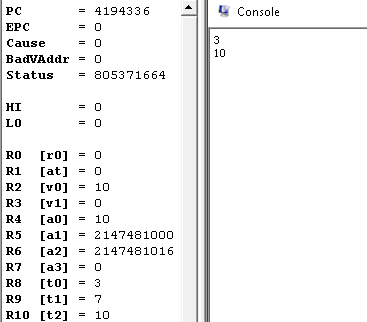
expected output?

**Code:**

**A screenshot of a computer program

Description automatically generated**

**Output:**

****

**Task 6:**

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

if ($t0 ==$t1 ) {

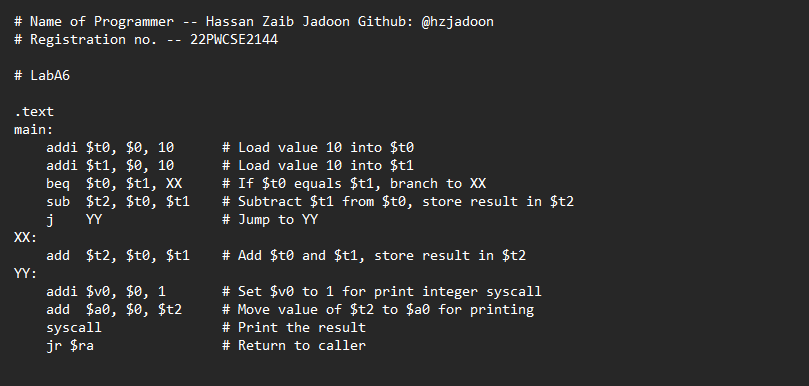
print ($t0 + $t1);

else {

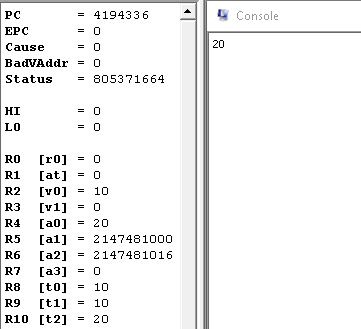
print ($t0 - $t1);

}

**Code:**

****

**Output:**



**Task 7:**

* 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)

* We can reduce a "less-than" test to a "not-equal" test by using the following instruction:

slt $x, $y, $z

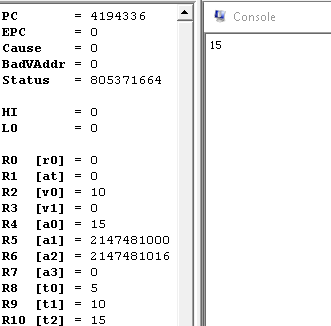
* It sets $x to 1 if $y < $z and sets it to zero otherwise. Using this "set-on-less-than" instruction, you can perform the above test using bne. Run the program and verify that it works as expected. Note that slt has an immediate variant (slti) that allows the third operand to be an immediate.

**Code:**

A screenshot of a computer program

Description automatically generated

**Output:**



**Task 8:**

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

addi $v0, $0, 1

addi $a0, $0, 0

loop: 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.
* Replace the statement before last in the above program with:

addi $a0, $0, 1

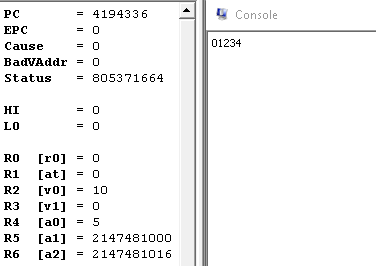
* What will the output be in this case? Show that single stepping is ideally suited to debug such an error.

**Code:**

**A screenshot of a computer

Description automatically generated**

**Output:**



**Task 9:**

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:

$t5 = 0;

loop: if (! $t5 < $t0) branch to done;

$s0 = $s0+ $t5;

$t5++;

jump to loop;

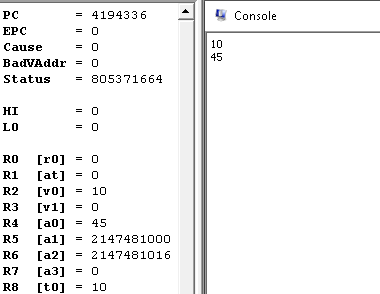
done: print($s0);

**Code:**

**A screenshot of a computer program

Description automatically generated**

**Output:**

****