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:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA1

.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  # Print integer
    syscall            # Do print
    jr $ra             # Return
```

Output:

```
PC         = 4194336
EPC        = 0
Cause      = 0
BadVAddr   = 0
Status     = 805371664

HI         = 0
LO         = 0

R0   [r0]  = 0
R1   [at]  = 0
R2   [v0]  = 10
R3   [v1]  = 0
R4   [a0]  = 67
```

Console

67

Task 2:

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

.globl fini

.text

main:

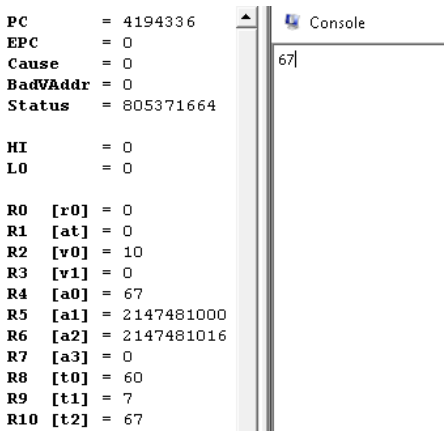
fini: jr $ra

- Save the modified program as: LabA2.asm.

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# Task 2

.globl fini
.text
main:
    addi $t0, $0, 60     # Load value 60 into $t0
    addi $t1, $0, 7      # Load value 7 into $t1
    add  $t2, $t0, $t1   # Add $t0 and $t1, store result in $t2
    addi $v0, $0, 1      # Set $v0 to 1 for print integer syscall
    add  $a0, $0, $t2    # Move value of $t2 to $a0 for printing
    syscall              # Make the syscall to print integer
fini:
    jr $ra               # Return to caller
```

Output:



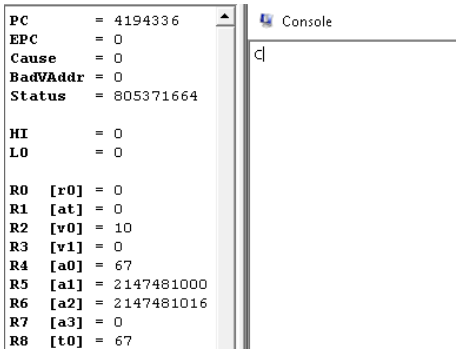
Task 3:

- 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:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA3

.text
main:
    addi $t0, $0, 67     # Load ASCII value 67 ('C') into $t0
    addi $v0, $0, 11     # Set $v0 to 11 for print character syscall
    add  $a0, $0, $t0    # Move value of $t0 to $a0 for printing
    syscall              # Make the syscall to print character
    jr $ra               # Return to caller
```

Output:



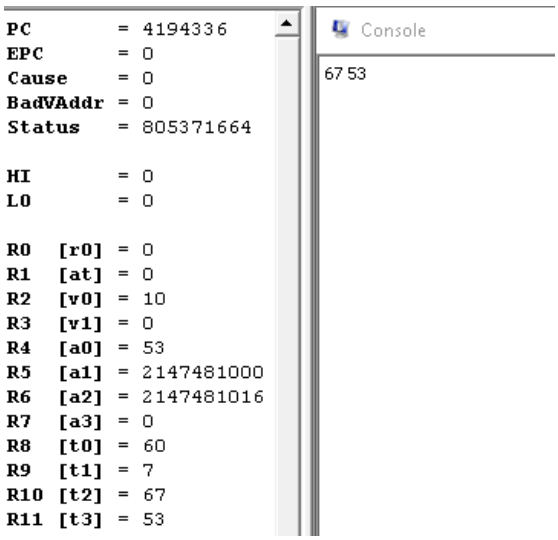
Task 4: Revert back to LabA2.asm and save it as LabA4.asm. Modify it so that it prints the sum and the difference of $t0 and $t1 separated by a space. Use the sub instruction to subtract registers. In order to print a space delimiter, use service #11 with $a0 being the space character:

addi $a0, $0, ' '

Code:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA4

.text
main:
    addi $t0, $0, 60      # Load value 60 into $t0
    addi $t1, $0, 7       # Load value 7 into $t1
    add  $t2, $t0, $t1    # Add $t0 and $t1, store result in $t2
    sub  $t3, $t0, $t1    # Subtract $t1 from $t0, store result in $t3
    addi $v0, $0, 1       # Set $v0 to 1 for print integer syscall

    add  $a0, $0, $t2     # Move value of $t2 to $a0 for printing
    syscall               # Print sum

    addi $a0, $0, ' '     # Set $a0 to space character
    addi $v0, $0, 11      # Set $v0 to 11 for print character syscall
    syscall               # Print space

    add  $a0, $0, $t3     # Move value of $t3 to $a0 for printing
    addi $v0, $0, 1       # Set $v0 to 1 for print integer syscall
    syscall               # Print difference

    jr $ra                # Return to caller
```

Output:



Task 5:

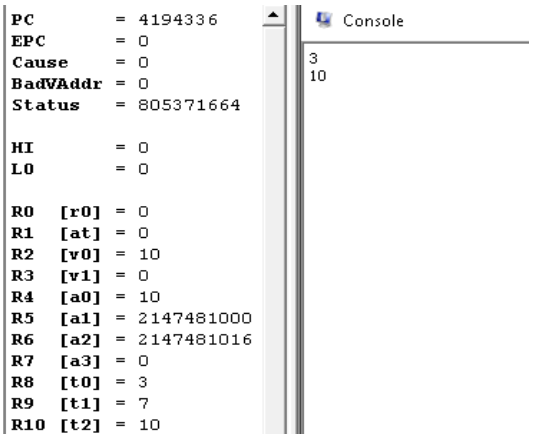
- Revert back to LabA2.asm and save it as LabA5.asm. Rather than hard-coding the numbers in our program, let us read one of them from the user by using service #5, readint.
- Replace the statement:
  addi $t0, $0, 60 # to = 60
- With:
  addi $v0, $0, 5 #v0 = readint
  syscall
  add $t0, $0,
  $v0

- 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:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA5

.text
main:
    addi $v0, $0, 5       # Set $v0 to 5 for read integer syscall
    syscall               # Make the syscall to read integer
    add  $t0, $0, $v0     # Move the input value from $v0 to $t0

    addi $t1, $0, 7       # Load value 7 into $t1
    add  $t2, $t0, $t1    # Add $t0 and $t1, store result in $t2
    addi $v0, $0, 1       # Set $v0 to 1 for print integer syscall
    add  $a0, $0, $t2     # Move value of $t2 to $a0 for printing
    syscall               # Print the result
    jr $ra                # Return to caller
```

Output:



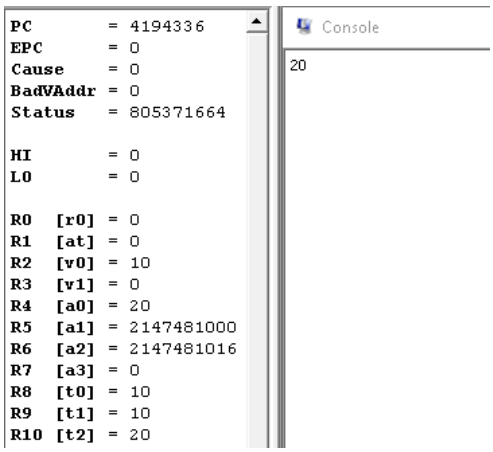
Task 6:

- 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:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA6

.text
main:
    addi $t0, $0, 10      # Load value 10 into $t0
    addi $t1, $0, 10      # Load value 10 into $t1
    beq  $t0, $t1, XX     # If $t0 equals $t1, branch to XX
    sub  $t2, $t0, $t1    # Subtract $t1 from $t0, store result in $t2
    j    YY               # Jump to YY
XX:
    add  $t2, $t0, $t1    # Add $t0 and $t1, store result in $t2
YY:
    addi $v0, $0, 1       # Set $v0 to 1 for print integer syscall
    add  $a0, $0, $t2     # Move value of $t2 to $a0 for printing
    syscall               # Print the result
    jr $ra                # Return to caller
```

Output:



Task 7:

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

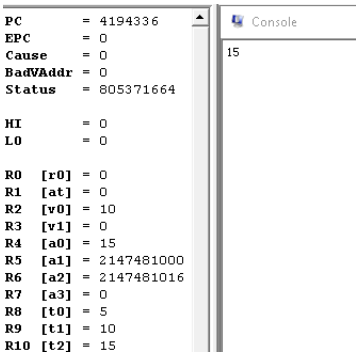
if ($t0 < $t1) {
  print ($t0 + $t1);
else {
print ($t0 - $t1)

- 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:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA7

.text
main:
    addi $t0, $0, 5       # Load value 5 into $t0
    addi $t1, $0, 10      # Load value 10 into $t1
    slt  $t3, $t0, $t1    # Set $t3 to 1 if $t0 < $t1, else 0
    bne  $t3, $0, XX      # If $t3 is not zero, branch to XX
    sub  $t2, $t0, $t1    # Subtract $t1 from $t0, store result in $t2
    j    YY               # Jump to YY
XX:
    add  $t2, $t0, $t1    # Add $t0 and $t1, store result in $t2
YY:
    addi $v0, $0, 1       # Set $v0 to 1 for print integer syscall
    add  $a0, $0, $t2     # Move value of $t2 to $a0 for printing
    syscall               # Print the result
    jr $ra                # Return to caller
```

Output:



Task 8:

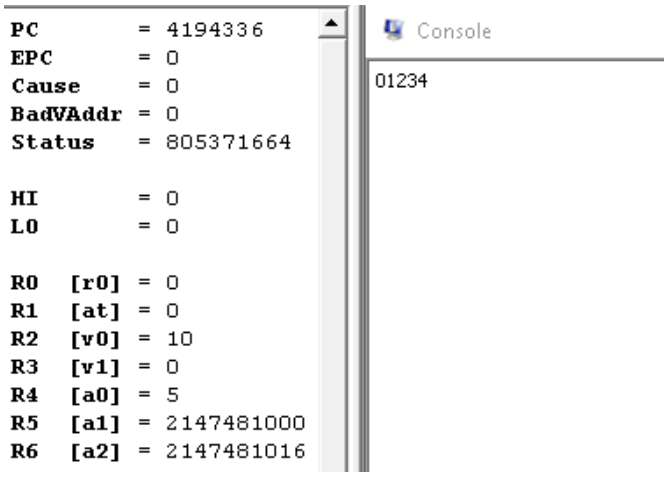
- Start fresh and create the program LabA8.asm with the following body (between main and fini):
  addi $v0, $0, 1
  addi $a0, $0, 0
  loop: slti $t9, $a0, 5
  beq $t9, $0, fini
  syscall
  addi $a0, $a0, 1
  j loop

- 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:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA8

.text
main:
    addi $v0, $0, 1       # Set $v0 to 1 for print integer syscall
    add  $a0, $0, $0      # Initialize $a0 to 0
loop:
    slti $t9, $a0, 5      # Set $t9 to 1 if $a0 < 5
    beq  $t9, $0, fini    # If $t9 is zero, exit loop
    syscall               # Print the current value of $a0
    addi $a0, $a0, 1      # Increment $a0
    j    loop             # Jump back to the beginning of the loop
fini:
    jr $ra                # Return

 caller
```

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:

```
# Name of Programmer -- Hassan Zaib Jadoon Github: @hzjadoon
# Registration no. -- 22PWCSE2144

# LabA9

.text
main:
    addi $s0, $0, 0       # Initialize $s0 to 0
    addi $v0, $0, 5       # Set $v0 to 5 for read integer syscall
    syscall               # Read integer input into $v0
    move  $t0, $v0        # Copy input value to $t0
    addi $t5, $0, 0       # Initialize loop counter $t5 to 0
loop:
    slt  $t9, $t5, $t0    # Set $t9 to 1 if $t5 < $t0
    beq  $t9, $0, done    # If $t9 is zero, exit loop
    add  $s0, $s0, $t5    # Add $t5 to $s0
    addi $t5, $t5, 1      # Increment $t5
    j    loop             # Jump back to the beginning of the loop
done:
    addi $v0, $0, 1       # Set $v0 to 1 for print integer syscall
    move  $a0, $s0        # Move sum result to $a0 for printing
    syscall               # Print the result
    jr $ra                # Return to caller
```

Output: