15CSE 381 Computer Organization and Architecture

Lab 2

5th July 2019

1 Read and Understand

1.1 Program Demonstrating System Calls

```
## Program showing system calls
## Enter two integers in console window
## Sum is displayed
        .text
        .globl main

main:
        la $t0, value

        li $v0, 5           # load code for read_int call in register $v0
        syscall             # read integer
        sw $v0, 0($t0)      # store integer returned by call (from console)

        li $v0, 5           # load code for read_int call in register $v0
        syscall             # read integer
        sw $v0, 4($t0)      # store integer returned by call (from console)

        lw $t1, 0($t0)
        lw $t2, 4($t0)
        add $t3, $t1, $t2
        sw $t3, 8($t0)

        li $v0, 4           # load code for print_string call in register $v0
        la $a0, msg1        # load register $a0 with argument for print_string call
        syscall             # print string
```

```
        li $v0, 1           # load code for print_int call in register $v0
        move $a0, $t3       # load register $a0 with argument for print_int call
        syscall             # print integer

        li  $v0, 10         # load code for program exit
        syscall             # exit

        .data
value:  .word 0, 0, 0
msg1:   .asciiz "Sum = "
```

2 Practice Program

2.1 A Simple ALP for Arithmetic Operations

```
.data
    prompt1:    .asciiz     "Enter the first number: "
    prompt2:    .asciiz     "Enter the second number: "
    menu:       .asciiz     "Enter the number associated: 1 => add, 2 => subtract or 3 => multiply
    resultText:     .asciiz     "Your final result is: "
    .text
.globl main
main:
    #The following block of code is to pre-load the integer values representing the various instru
    li $t3, 1    #This is to load the immediate value of 1 into the temporary register $t3
    li $t4, 2    #This is to load the immediate value of 2 into the temporary register $t4
    li $t5, 3    #This is to load the immediate value of 3 into the temporary register $t5

#asking the user to provide the first number
    li $v0, 4     #command for printing a string
    la $a0, prompt1 #loading the string to print into the argument to enable printing
    syscall      #executing the command


    #the next block of code is for reading the first number provided by the user
    li $v0, 5    #command for reading an integer
    syscall      #executing the command for reading an integer
    move $t0, $v0     #moving the number read from the user input into the temporary register $t0
```

```
#asking the user to provide the first number
    li $v0, 4     #command for printing a string
    la $a0, prompt1 #loading the string to print into the argument to enable printing
    syscall      #executing the command


    #the next block of code is for reading the first number provided by the user
    li $v0, 5    #command for reading an integer
    syscall      #executing the command for reading an integer
    move $t0, $v0     #moving the number read from the user input into the temporary

    #asking the user to provide the second number
    li $v0, 4    #command for printing a string
    la $a0, prompt2    #loading the string into the argument to enable printing
    syscall      #executing the command


    #reading the second number to be provided to the user
    li $v0, 5    #command to read the number  provided by the user
    syscall      #executing the command for reading an integer
    move $t1, $v0    #moving the number read from the user input into the temporary r

li $v0, 4    #command for printing a string
    la $a0, menu    #loading the string into the argument to enable printing
    syscall      #executing the command
```

```
    #the next block of code is to read the number provided by the user
    li $v0, 5    #command for reading an integer
    syscall      #executing the command
    move $t2, $v0    #this command is to move the integer provided into the temporary
 beq $t2,$t3,addProcess    #Branch to 'addProcess' if $t2 = $t3
    beq $t2,$t4,subtractProcess #Branch to 'subtractProcess' if $t2 = $t4
    beq $t2,$t5,multiplyProcess #Branch to 'multiplyProcess' if $t2 = $t5
  addProcess:
    add $t6,$t0,$t1    #this adds the values stored in $t0 and $t1 and assigns them to
    j resultsection
 subtractProcess:
    sub $t6,$t0,$t1 #this adds the values stored in $t0 and $t1 and assigns them to th
    j resultsection
multiplyProcess:
    mul $t6,$t0,$t1 #this adds the values stored in $t0 and $t1 and assigns them to th
resultsection:
     li $v0,4    #this is the command for printing a string
    la $a0,resultText    #this loads the string to print into the argument $a0 for pri
    syscall      #executes the command

    #the following line of code prints out the result of the addition computation
    li $v0,1
    la $a0, ($t6)
    syscall
    li $v0,10 #This is to terminate the program
    syscall
```

3 Questions

1. Write a non-recursive MIPS program to find factorial of a number ranging 0-15. The program should read the input and check for the range.

2. Write MIPS code to implementing the following function program, without using stack.

```
  #include <stdio.h>
int function(int a);
int main()
{
  int x=5;
  int y;
  y = function(x);
  printf("y=%i\n", y);
  return 0;
```

```
}
int function(int a)
{
  return 3*a+5;
}
```