San Francisco Bay University

EE488 - Computer Architecture
Homework Assignment #5

Due day: 8/7/2024

Md. Maniruzzaman
ID: 20099

Instruction:

1. Push the answer sheet to GitHub in word file
2. Overdue homework submission could not be accepted.
3. Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)

1. Implement a subprogram that prompt the user for *3* numbers, finds the median (middle value) of the *3*, and returns that value to the calling program.

Answer:

```
.data
prompt1: .asciiz "Enter the first number: "
prompt2: .asciiz "Enter the second number: "
prompt3: .asciiz "Enter the third number: "
median_msg: .asciiz "\nThe median number is: "

.text
.globl main

main:
  li $v0, 4
  la $a0, prompt1
  syscall

  li $v0, 5
  syscall
  move $t0, $v0

  li $v0, 4
  la $a0, prompt2
  syscall

  li $v0, 5
  syscall
  move $t1, $v0

  li $v0, 4
  la $a0, prompt3
  syscall

  li $v0, 5
  syscall
  move $t2, $v0

  jal find_median
  move $t3, $v0
```

```
    li $v0, 4
    la $a0, median_msg
    syscall

    li $v0, 1
    move $a0, $t3
    syscall

    li $v0, 10
    syscall

find_median:
    ble $t0, $t1, check_t0_t2
    move $a0, $t0
    move $t0, $t1
    move $t1, $a0

check_t0_t2:
    ble $t0, $t2, check_t1_t2
    move $a0, $t0
    move $t0, $t2
    move $t2, $a0

check_t1_t2:
    ble $t1, $t2, return_t1
    move $a0, $t1
    move $t1, $t2
    move $t2, $a0

return_t1:
    move $v0, $t1
    jr $ra
```

======================

2. Implement a recursive program that takes in a number and finds the square of that number through addition. For example if the number *3* is entered, you would add *3+3+3=9*. If *4* is entered, you would add *4+4+4+4=16*. This program must be implemented using recursion to add the numbers together.

   Answer:

```
.data
  user_input: .asciiz "Input your number : "
  addition: .asciiz "Square the user input value : "

.text


  li $v0,4
  la $a0,user_input
  syscall

  li $v0,5
   syscall
  move $s0,$v0
   move $t0,$v0
  li $s1,0

iteration:
    blez $t0,exit
    add $s1,$s1,$s0
    addi $t0,$t0,-1
    j iteration

exit:
   li $v0,4
   la $a0,addition
   syscall

  li $v0,1
  move $a0,$s1
  syscall

  li $v0,10
  syscall
```

==========================

3. Write a recursive program to calculate factorial numbers. Use the definition of factorial as *F(n) = n * F(n-1)*

Answer:

```
.data
prompt: .asciiz "Enter a number: "
result_msg: .asciiz "The factorial of the number is: "

.text
.globl factorial_program

factorial_program:
    li $v0, 4
    la $a0, prompt
    syscall

    li $v0, 5
    syscall
    move $a0, $v0

    jal factorial
    move $t0, $v0

    li $v0, 4
    la $a0, result_msg
    syscall

    li $v0, 1
    move $a0, $t0
    syscall

    li $v0, 10
    syscall

factorial:
    li $t1, 1
    beq $a0, $t1, base_case
    li $t1, 0
    beq $a0, $t1, base_case

    addi $sp, $sp, -8
    sw $ra, 4($sp)
    sw $a0, 0($sp)

    addi $a0, $a0, -1
    jal factorial

    lw $a0, 0($sp)
    lw $ra, 4($sp)
    addi $sp, $sp, 8

    mul $v0, $v0, $a0
    jr $ra

base_case:
    li $v0, 1
    jr $ra
```

=========================

4. The following pseudo code converts an input value of a single decimal number from $1 \leq n \geq 15$ into a single hexadecimal digit. Translate this pseudo code into MIPS assembly.

```
main{
  String a[16]
  a[0] = "0x0"
  a[1] = "0x1"
  a[2] = "0x2"
  a[3] = "0x3"
  a[4] = "0x4"
  a[5] = "0x5"
  a[6] = "0x6"
  a[7] = "0x7"
  a[8] = "0x8"
  a[9] = "0x9"
  a[10] = "0xa"
  a[11] = "0xb"
  a[12] = "0xc"
  a[13] = "0xd"
  a[14] = "0xe"
  a[15] = "0xf"

  int i = prompt("Enter a number from 0 to 15 ")
  print("your number is " + a[i])
}
```

Answer:

```
.data
prompt: .asciiz "Enter a number from 0 to 15: "
result: .asciiz "Your number is: "
newline: .asciiz "\n"

# Array of pointers to hexadecimal
```

```
array: .word a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14,
a15

a0: .asciiz "0x0"
a1: .asciiz "0x1"
a2: .asciiz "0x2"
a3: .asciiz "0x3"
a4: .asciiz "0x4"
a5: .asciiz "0x5"
a6: .asciiz "0x6"
a7: .asciiz "0x7"
a8: .asciiz "0x8"
a9: .asciiz "0x9"
a10: .asciiz "0xa"
a11: .asciiz "0xb"
a12: .asciiz "0xc"
a13: .asciiz "0xd"
a14: .asciiz "0xe"
a15: .asciiz "0xf"

.text
.globl main

main:
    li $v0, 4
    la $a0, prompt
    syscall

    li $v0, 5
    syscall
    move $t0, $v0

    li $t1, 0
    li $t2, 15
    blt $t0, $t1, invalid_input
    bgt $t0, $t2, invalid_input

    la $t3, array
    sll $t5, $t0, 2
    add $t3, $t3, $t5
    lw $a1, 0($t3)

    li $v0, 4
    la $a0, result
    syscall

    li $v0, 4
    move $a0, $a1
    syscall
    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 10
    syscall

invalid_input:
    li $v0, 4
    la $a0, newline
    syscall
    li $v0, 10
    syscall
```

============================

5. The following pseudo code program calculates the Fibonacci numbers from *1…n*, and stores them in an array. Translate this pseudo code into MIPS assembly, and use the PrintIntArray subprogram to print the results.

```
main{
  int size = PromptInt("Enter a max Fibonacci number to calc: ")
  int Fibonacci[size]

  Fibonacci[0] = 0
  Fibonacci[1] = 1

  for (int i = 2; i < size; i++){
    Fibonacci[i] = Fibonacci[i-1] + Fibonacci[i-2]
  }
  PrintIntArray(Fibonacci, size)
}
```

Answer:

```
.data
   msg: .asciiz "Please input a max Fibonacci number to calculate: "
   newline: .asciiz "\n"

.text
   main:
      la $a0, msg
      li $v0, 4
      syscall

      li $v0, 5
      syscall
      move $t0, $v0

      li $a0, 0
      li $v0, 1
      syscall
      li $a0, 32
```

```
        li $v0, 11
        syscall
        li $a0, 1
        li $v0, 1
        syscall
        li $a0, 32
        li $v0, 11
        syscall

        jal printarray_line

        li $v0, 10
        syscall

printarray_line:
    addi $sp, $sp, -4
    sw $t0, 0($sp)

    li $t1, 2
    blt $t1, $t0, iteration

    lw $t0, 0($sp)
    addi $sp, $sp, 4
    jr $ra

iteration:
    li $t2, 0
    li $t3, 1

fibonacci_number:
    add $t4, $t3, $t2

    move $a0, $t4
    li $v0, 1
    syscall
    li $a0, 32
    li $v0, 11
    syscall

    move $t2, $t3
    move $t3, $t4

    addi $t1, $t1, 1

    blt $t1, $t0, fibonacci_number

    lw $t0, 0($sp)
    addi $sp, $sp, 4
    jr $ra
```