HO CHI MINH UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER ARCHITECTURE

Practical session - Week 4

Lecturers: Assoc. Prof. Dr. Cuong Pham Quoc

Mr. Kieu Do Nguyen Binh

Class: CC04 Group: 8

Members: Nguyễn Minh Hùng Student ID: 1952737

Phạm Nhật Hoàng Student ID: 1952703

Nguyễn Bá Minh Hưng Student ID: 1952748

Question 1. Write a MIPS program with the following steps:
1. Request an integer number from users.
2. If the number is positive, repeat step 1. Otherwise, print sum of all integer numbers that the program has read from users.

```
.text
.globl main

main:
li          $t3,0           # sum = 0
loop:                       # repeat
li          $v0, 4          # input
la          $a0, msg
            syscall

li          $v0, 5          # integer input
            syscall
move        $s0, $v0

add         $t3,$t3,$s0     # update sum

slti        $t0,$s0,1       # t0 = (s0 < 1) ?
beq         $t0,0,loop

li          $v0, 4          # output
la          $a0, sum
            syscall
li          $v0, 1
move        $a0,$t3
            syscall

.data

msg:        .asciiz "Input integer: "
sum:        .asciiz "Sum: "
```

q1.asm

Question 2. Implement the following C code by using MIPS code. Assume that b and c are 10 and 5, respectively while input variable is read from keyboard. Print value of a to the terminal.

---

switch (input) {
case 0: a = b + c; break;
case 1: a = b - c; break;
case 2: a = c - b; break;
default: printf "please input an another integer numbers"; break;
}

```
.text
.globl main
main:
li          $s0,10      # b = 10
li          $s1,5       # c = 5

input:
li      $v0,4       # print_string syscall code = 4
la      $a0, msg    # load the address of msg
        syscall
li          $v0,5
            syscall
move        $t0,$v0     # read input


beq         $t0,0,case0
beq         $t0,1,case1
beq         $t0,2,case2
default:
li      $v0,4       # print_string syscall code = 4
la      $a0, msg1   # load the address of msg1
        syscall
j end
case0:
add         $t1,$s0,$s1
j exit
case1:
sub         $t1,$s0,$s1
j exit
case2:
sub         $t1,$s1,$s0
j exit
exit:
li      $v0,4       # print_string syscall code = 4
la      $a0, msg2   # load the address of msg2
        syscall
li          $v0,1
move        $a0,$t1
            syscall
end:
.data

msg:        .asciiz "Input: "
msg1:       .asciiz "please input an another integer numbers\n "
msg2:       .asciiz "a = "
```

q2.asm

Question 3. Write a MIPS program with the following requirements:
1. Declare an integer array with 10 synthetic data elements.
2. Read an integer number from users.
3. Find in the array if the integer read from user exists in the array or not. Print the position of the integer number in the array if found; otherwise tell users that the number does not exist in the array

```
.text
.globl main
main:

li          $v0,4               # print input
la          $a0,input
            syscall

li          $v0,5               # integer input
            syscall
move        $t0,$v0

la          $s0, array          # load array address to s0
li          $t2, 0              # position
loop:
beq         $t2,10,exit         # loop cond: i < 10
lw          $t1, 0($s0)         # load array element
beq         $t1,$t0,found        # if found
addi        $s0,$s0,4
addi        $t2,$t2,1
j           loop

exit:

li          $v0,4
la          $a0,notfound
            syscall
j           end
found:
li          $v0,4
la          $a0,print
            syscall
li          $v0,1
move        $a0,$t2
            syscall
end:

.data
array:      .word  4,2,6,1,7,9,1,2,5,6
input:      .asciiz "Input an integer: "
notfound:   .asciiz "the number does not exist in the array!"
print:      .asciiz "Position found: "
```

q3.asm

Question 4. Given the following leaf procedure in ANSI C

---

void swap(int v[], int k){
int temp;
temp = v[k]
v[k] = v[k+1];
v[k+1] = temp;
}

Assume that the $a0 register will store the base address of the v array while

the $a1 register keeps value k. The array v consists of 10 elements in integer and is pre-defined in the data section.
1. Write a main program the receive value k from user, check the value k and call the procedure swap if possible.
2. Watch the $ra register before and after the jal and jr instructions are executed.

```
.text
.globl main


main:
li          $v0,4                   # print input
la          $a0,input
            syscall
li          $v0,5                   # integer input
            syscall
move        $a1,$v0

la          $a0, array              # a0 = v[0]

slti        $t4,$a1,10               # check valid k (a1)
slti        $t5,$a1,0
nor         $t5,$t5,$t5
and         $t6,$t5,$t4
beq         $t6,0,exit
jal         swap
j end
swap:
sll         $t1, $a1, 2             # t1 = 4 * k
add         $t1, $a0, $t1           # t1 = v[k]
lw          $t0, 0($t1)             # load v[k] to t0
lw          $t2, 4($t1)             # load v[k+1] to t2
sw          $t2, 0($t1)             # store v[k+1] to v[k]
sw          $t0, 4($t1)             # store v[k] to v[k+1]
jr          $ra

exit:
li          $v0,4                   # invalid k
la          $a0,invalid
            syscall
end:


.data
input:      .asciiz "input k: "
array:      .word   1,2,3,4,5,6,7,8,9,10
invalid:    .asciiz "invalid k"
```

q4.asm

2.
- Before jal and jr instructions, %ra register value is 0x0000.
- After jal and jr instructions, %ra stores: 0x00400040 (this address is same as the address of PC after jal swap (j end).

Question 5. Given the following factorial MIPS program in a recursive form (as in the slide)

```
fact: addi $sp, $sp, -8 # adjust stack for 2 items
sw $ra, 4($sp) # save return address
sw $a0, 0($sp) # save argument
slti $t0, $a0, 1 # test for n < 1
beq $t0, $zero, L1
addi $v0, $zero, 1 # if so, result is 1
addi $sp, $sp, 8 # pop 2 items from stack
jr $ra # and return
L1: addi $a0, $a0, -1 # else decrement n
jal fact # recursive call
lw $a0, 0($sp) # restore original n
lw $ra, 4($sp) # and return address
addi $sp, $sp, 8 # pop 2 items from stack
mul $v0, $a0, $v0 # multiply to get result
jr $ra # and return
```

ex.asm

1. Type the above procedure and write a main program that call the above procedure with different n, where n is in the $a0 register. Watch the results

2. When n is 2, run the program step by step and watch the execution of instructions as well as the $ra register and values store/load to/from the stack.

```
.text
.globl main


main:
li          $v0,4                   # print input
la          $a0,input
            syscall
li          $v0,5                   # integer input
            syscall
move        $a0,$v0

jal         fact
j exit
fact: addi $sp, $sp, -8 # adjust stack for 2 items
sw $ra, 4($sp) # save return address
sw $a0, 0($sp) # save argument
slti $t0, $a0, 1 # test for n < 1
beq $t0, $zero, L1
addi $v0, $zero, 1 # if so, result is 1
addi $sp, $sp, 8 # pop 2 items from stack
jr $ra # and return
L1: addi $a0, $a0, -1 # else decrement n
jal fact # recursive call
lw $a0, 0($sp) # restore original n
lw $ra, 4($sp) # and return address
addi $sp, $sp, 8 # pop 2 items from stack
mul $v0, $a0, $v0 # multiply to get result
jr $ra # and return

exit:
```

```
32
33 move          $t0 ,$v0
34
35 li            $v0 ,1
36 move          $a0 ,$t0
37               syscall
38
39 .data
40 input:        .asciiz "Input n: "
```

q5.asm