**LAB REPORT**

**IT3280E– 152049– Assembly Language and Computer Architecture Lab**

**Lab 07: Subroutine Call and Passing Parameters**

**Using the Stack**

**Assignment 1:**

*Create a project to implement Home Assignment 1. Compile and simulate it. Change the program parameters (register a0) and observe the execution results. Run the program in the single-step mode and pay attention to the changes in registers, especially the pc and ra registers.*

* Simulate:

A white rectangular object with a blue border

Description automatically generated with medium confidence

* Change parameters and observe:

1. a0 <0:

+ Program:

A screenshot of a computer

Description automatically generated

+ Run:

* li a0, -348:  
* jal abs: jump to *abs* procedure A screenshot of a computer

  Description automatically generated



* ra:



* sub s0, zero, a0: 



* blt a0, zero, done: since a0 < 0, the program is done



* jr ra: -> return to return address (after jal abs)



* li a7, 10: terminate program





* **Result: A white rectangular object with a blue border

  Description automatically generated**

1. a0 > 0:

+ Program:

A screenshot of a computer program

Description automatically generated

+ Run:

* li a0, -348:





* jal abs: jump to *abs* procedure A screenshot of a computer

  Description automatically generated



* ra:



* sub s0, zero, a0:





* blt a0, zero, done: compare a0 with 0, since a0 > 0, we continue



* add s0, a0, zero: 



* jr ra: -> return to return address (after jal abs)



* li a7, 10: terminate program





* Result: A white rectangular object with a white stripe

  Description automatically generated

**Assignment 2:**

*Create a project to implement Home Assignment 2. Compile and simulate it. Change the program parameters (registers a0, a1, a2) and observe the execution results. Run the program in the single-step mode and pay attention to the changes in registers, especially the pc and ra registers.*

* Simulate:

A screenshot of a computer

Description automatically generated

* s0 is the largest element among all three integers (stored in a0, a1, a2)
* Change parameters and observe:

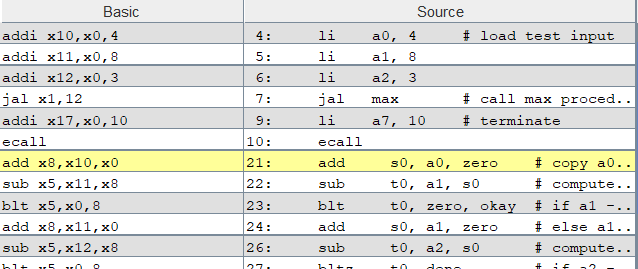
+ Program (w/ changed parameters):

A screenshot of a computer

Description automatically generated

+ Run:

* load input: A green and white rectangle

  Description automatically generated
* jal max: jump to *max* procedure 
* ra:



* max: copy a0 in s0; largest so far -> compute a1 - s0 -> if a1 - v0 < 0 then no change -> a1 is the largest so far





* sub t0, a2, s0: compute a2 - v0



* bltz t0, done: since t0 < 0 -> done
* li a7, 10: terminate program
* **Result: A screenshot of a computer

  Description automatically generated**
* the pc registers changed depending on which instruction it is running (normally increase by 4 or more if jump)

**Assignment 3:**

*Create a project to implement Home Assignment 3. Compile and simulate it. Change the program parameters (registers s0, s1), observe the process and results. Pay attention to changes in the sp register. Observe the memory pointed to by sp in the Data Segment window.*

* Simulate:



swapped



* Change parameters and observe:

+ Program:

A screenshot of a computer

Description automatically generated

+ Run:

* load text input:  
* addi sp, sp, -8: adjust stack pointer
* sw s0, 4(sp)

sw s1, 0(sp)

* push into stack



* lw s0, 0(sp)

lw s1, 4(sp)

* pop from stack



* addi sp, sp, 8: adjust stack pointer



**Assignment 4:**

*Create a project to implement Home Assignment 4. Compile and simulate it. Change the parameter in the a0 register and check the result in the s0 register. Run the program in the single-step mode and observe the changes in the registers pc, ra, sp, a0, s0. List the values in the stack memory when executing the program with n = 3.*

* Compile and Simulate:

*A screenshot of a computer

Description automatically generated*

* Change parameters and observe:

+ Program (w/changed parameters):

A screenshot of a computer code

Description automatically generated

+ Result in s0 register: 

+ Observe changes (with n = 3):

* initial value of pc, ra, sp, a0, s0: A white rectangular object with a blue border

  Description automatically generated
* jal WARP: 



* li a0, 3:



* jal FACT -> addi sp, sp, -8:



* addi a0, a0, -1: adjust input argument



* jal FACT -> addi sp, sp, -8:





* addi a0, a0, -1: adjust input argument



* jal FACT -> addi sp, sp, -8:





* li s0, 1:



* lw ra, 4(sp) # restore ra register

lw a0, 0(sp) # restore a0 register

addi sp,sp,8 # restore stack pointer



* lw s1, 0(sp): load a0

mul s0, s0, s1:



* lw ra, 4(sp) # restore ra register

lw a0, 0(sp) # restore a0 register

addi sp,sp,8 # restore stack pointer





* lw s1, 0(sp): load a0

mul s0, s0, s1:



* lw ra, 4(sp) # restore ra register

lw a0, 0(sp) # restore a0 register

addi sp,sp,8 # restore stack pointer





* lw ra, 0(sp) # restore return address

addi sp, sp, 4 # return stack pointer



* la a0, message: 



* **Result:**

*A screenshot of a computer

Description automatically generated*

+ Values in the stack memory when executing the program:



# Assignment 5:

*Write a subroutine to find the largest value, the smallest value, and their respective*

*positions in a list of 8 integers stored in the registers from a0 to a7. For example:*

*▪ Largest: 9, 3 → The largest value is 9, stored in a3.*

*▪ Smallest: -3, 6 → The smallest value is -3, stored in a6.*

***Hint****: Use the stack memory to pass parameters.*

**Subroutine:**

.text

.global main

main:

# load 8 integers into registers a0 to a7

li a0, 5

li a1, -2

li a2, 7

li a3, 9

li a4, 1

li a5, 3

li a6, -3

li a7, 4

jal find\_max\_min

# cxit program

li a7, 10

ecall

# --------------------------------------------------------------------

# Syntax:

# s0: largest value

# s1: position of the largest value

# s2: smallest value

# s3: position of the smallest value

# --------------------------------------------------------------------

find\_max\_min:

addi s0, a0, 0 # s0 = largest value

li s1, 0 # s1 = largest value position (index 0)

addi s2, a0, 0 # s2 = smallest value

li s3, 0 # s3 = smallest value position (index 0)

# compare each register from a1 to a7

li t0, 1 # Index counter

# compare a1

bgt a1, s0, update\_max\_1

blt a1, s2, update\_min\_1

j compare\_a2

update\_max\_1:

addi s0, a1, 0

addi s1, t0, 0

j compare\_a2

update\_min\_1:

addi s2, a1, 0

addi s3, t0, 0

j compare\_a2

compare\_a2:

addi t0, t0, 1

bgt a2, s0, update\_max\_2

blt a2, s2, update\_min\_2

j compare\_a3

update\_max\_2:

addi s0, a2, 0

addi s1, t0, 0

j compare\_a3

update\_min\_2:

addi s2, a2, 0

addi s3, t0, 0

j compare\_a3

compare\_a3:

addi t0, t0, 1

bgt a3, s0, update\_max\_3

blt a3, s2, update\_min\_3

j compare\_a4

update\_max\_3:

addi s0, a3, 0

addi s1, t0, 0

j compare\_a4

update\_min\_3:

addi s2, a3, 0

addi s3, t0, 0

j compare\_a4

compare\_a4:

addi t0, t0, 1

bgt a4, s0, update\_max\_4

blt a4, s2, update\_min\_4

j compare\_a5

update\_max\_4:

addi s0, a4, 0

addi s1, t0, 0

j compare\_a5

update\_min\_4:

addi s2, a4, 0

addi s3, t0, 0

j compare\_a5

compare\_a5:

addi t0, t0, 1

bgt a5, s0, update\_max\_5

blt a5, s2, update\_min\_5

j compare\_a6

update\_max\_5:

addi s0, a5, 0

addi s1, t0, 0

j compare\_a6

update\_min\_5:

addi s2, a5, 0

addi s3, t0, 0

j compare\_a6

compare\_a6:

addi t0, t0, 1

bgt a6, s0, update\_max\_6

blt a6, s2, update\_min\_6

j compare\_a7

update\_max\_6:

addi s0, a6, 0

addi s1, t0, 0

j compare\_a7

update\_min\_6:

addi s2, a6, 0

addi s3, t0, 0

j compare\_a7

compare\_a7:

addi t0, t0, 1

bgt a7, s0, update\_max\_7

blt a7, s2, update\_min\_7

j ret\_from\_sub

update\_max\_7:

addi s0, a7, 0

addi s1, t0, 0

j ret\_from\_sub

update\_min\_7:

addi s2, a7, 0

addi s3, t0, 0

ret\_from\_sub:

ret # Return from subroutine