BÁO CÁO THỰC HÀNH
IT3280 – 156788 – THỰC HÀNH KIẾN TRÚC MÁY TÍNH

NỘI DUNG
Bài 7. Lệnh gọi chương trình con, truyền tham số sử dụng ngăn xếp

Họ và tên	Nguyễn Minh Quân
Mã số sinh viên	20235816

Assignment 1

Tạo project để thực hiện Home Assignment 1. Dịch và chạy mô phỏng. Thay đổi các tham số chương trình (thanh ghi a0) và quan sát kết quả thực hiện. Chạy chương trình ở chế độ từng dòng lệnh và chú ý sự thay đổi của các thanh ghi, đặc biệt là thanh ghi pc và ra.

Home Assignment 1

Chương trình dưới đây minh họa việc khai báo và sử dụng hàm abs để tính giá trị tuyệt đối của một số nguyên. Hàm sử dụng 2 thanh ghi, a0 chứa tham số vào và s0 chứa kết quả. Đọc kỹ chương trình và hiểu cách khai báo và gọi chương trình con.

Chương trình thực hiện:

riscv1.asm*

```
1  # Laboratory Exercise 7 Home Assignment 1
2  .text
3  main:
4  li a0, -20 # load input parameter
5  jal abs # jump and link to abs procedure
6  li a7, 10 # terminate
7  ecall
8  end_main:
9  abs:
10 sub s0, zero, a0 # put -a0 in s0; in case a0 < 0
11 blt a0, zero, done # if a0<0 then done
12 add s0, a0, zero # else put a0 in s0
13 done:
14 jr ra
```

Chạy chương trình ở chế độ từng dòng lệnh và chú ý sự thay đổi của các thanh ghi, đặc biệt là thanh ghi pc và ra.

Lệnh 1: li a0, -20 #load -20 vào a0, a0 = -20

a0	10	-20
pc		4194308
ra	1	0

Lệnh 2: jal abs nhảy đến label abs và lưu địa chỉ của lệnh tiếp theo (lệnh li a7, 10) vào thanh ghi ra

pc		4194320
ra	1	4194312

Lệnh 3:sub s0, zero, a0 # s0 = -a0 = 20

s0	8	20
pc		4194324
ra	1	4194312

Lệnh 4: blt a0, zero, done # a0 = -20 < 0 → Nhảy tới label done

s0	8	20
pc		4194332
ra	1	4194312

Lệnh 5: jr ra # Nhảy tới địa chỉ lưu trong thanh ghi ra(địa chỉ của li a7,10)

pc		4194312
ra	1	4194312

Kết quả:

a0	10	-20
s0	8	20

Assignment 2

Tạo project để thực hiện Home Assignment 2. Dịch và chạy mô phỏng. Thay đổi các tham số chương trình (thanh ghi a0, a1, a2) và quan sát kết quả thực hiện. Chạy chương trình ở chế độ từng dòng lệnh và chú ý sự thay đổi của các thanh ghi, đặc biệt là thanh ghi pc và ra.

Home Assignment 2

Trong ví dụ này, chương trình con max được khai báo và sử dụng để tìm phần tử lớn nhất trong 3 số nguyên. Các tham số này được truyền vào chương trình con qua các thanh ghi a0, a1, a2. Kết quả được lưu vào thanh ghi s0. Đọc kỹ chương trình và hiểu cách khai báo và gọi chương trình con.

Chương trình thực hiện:

```
1  # Laboratory Exercise 7, Home Assignment 2
2  .text
3  main:
4  li a0, 4 # load test input
5  li a1, 8
6  li a2, 3
7  jal max # call max procedure
8  li a7, 10 # terminate
9  ecall
10 end_main:
11 max:
12 add s0, a0, zero # copy a0 in s0; largest so far
13 sub t0, a1, s0 # compute a1 - s0
14 blt t0, zero, okay # if a1 - v0 < 0 then no change
15 add s0, a1, zero # else a1 is largest thus far
16 okay:
17 sub t0, a2, s0 # compute a2 - v0
18 bltz t0, zero, done # if a2 - v0 <0 then no change
19 add s0, a2, zero # else a2 is largest overall
20 done:
21 jr ra # return to calling program
```

Chạy chương trình ở chế độ từng dòng lệnh và chú ý sự thay đổi của các thanh ghi, đặc biệt là thanh ghi pc và ra.

Lệnh 1:
li a0, 4 # a0 =4
li a1, 8 # a1 =8
li a2, 3 #a2=3

a0	10	4
a1	11	8
a2	12	3

Lệnh 2:
jal max # nhảy tới nhãn max, lưu địa chỉ trở về(câu lệnh tiếp theo)vào ra

ra	1	4194320
pc		4194328

Lệnh 3: add s0, a0, zero # max = s0 = a0

s0	8	4
pc		4194332
ra	1	4194320

Lệnh 4: sub t0, a1, s0 # So sánh xem a1 có lớn hơn s0 hiện tại hay ko, t0 = a1-s0

t0	5	4
pc		4194336
ra	1	4194320

Lệnh 5: blt t0, zero, okay # Nếu a1 – s0 < 0 hay a1 < max thì nhảy tới nhãn okay kiểm tra tiếp

pc		4194340

ra	1	4194320

Lệnh 6: add s0, a1, zero # Còn không thì update max = s0 = a1

s0	8	8
pc		4194344
ra	1	4194320

Lệnh 7: sub t0, a2, s0 # Tiếp tục check xem a2 có lớn hơn max hay không

t0	5	-5
pc		4194348
ra	1	4194320

Lệnh 8: blt t0, zero, done # Nếu a2 – s0 < 0 thì không update gì cả, nhảy tới nhãn done

t0	5	-5
pc		4194356
ra	1	4194320

Lệnh 9: jr ra # quay lại địa chỉ đã lưu trong thanh ra

pc		4194320
ra	1	4194320

Kết quả:

s0	8	8
s1	9	0
a0	10	4
a1	11	8
a2	12	3

Assignment 3

Tạo project để thực hiện Home Assignment 3. Dịch và chạy mô phỏng. Thay đổi tham số chương trình (thanh ghi s0, s1), quan sát quá trình và kết quả thực hiện. Chú ý sự thay đổi giá trị của thanh ghi sp. Quan sát vùng nhớ được trỏ bởi thanh ghi sp trong cửa sổ Data Segment.

Home assigment3

Chương trình hợp ngữ dưới đây minh họa cách sử dụng vùng nhớ ngăn xếp (stack) với hai phép toán push và pop bằng cách sử dụng lệnh lw và sw. Giá trị của hai thanh ghi s0 và s1 sẽ được hoán đổi với nhau sử dụng ngăn xếp.

Chương trình thực hiện:

```
1  # Laboratory Exercise 7, Home Assignment 3
2  .text
3  main:
4          li s0,5
5          li s1,6
6  push:
7          addi sp, sp, -8 # adjust the stack pointer
8          sw s0, 4(sp) # push s0 to stack
9          sw s1, 0(sp) # push s1 to stack
10 work:
11         nop
12         nop
13         nop
14 pop:
15         lw s0, 0(sp) # pop from stack to s0
16         lw s1, 4(sp) # pop from stack to s1
17         addi sp, sp, 8 # adjust the stack pointer
```

Quan sát quá trình và kết quả thực hiện. Chú ý sự thay đổi giá trị của thanh ghi sp. Quan sát vùng nhớ được trỏ bởi thanh ghi sp trong cửa sổ Data Segment.

Lệnh 1:
li s0,5 #s0 = 5
li s1,6 # s0 =6

sp	2	2147479548

Lệnh 2:
addi sp, sp, -8 # sp = sp -8, tạo không gian 8 byte trên stack

sp	2	2147479540

Lệnh 3:
sw s0, 4(sp) # Lưu s0=5 vào địa chỉ sp +4

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x7fffefe0	0	0	0	0	0	0	5

Lệnh 4:
sw s1, 0(sp) # Lưu s1=6 vào địa chỉ sp + 0

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x7fffefe0	0	0	0	0	0	6	5
0x7ffff000	0	0	0	0	0	0	0

Lệnh 5:
lw s0, 0(sp) # Tải giá trị từ sp + 0(6) vào thanh ghi s0 → s0 =6

s0	8	6

Lệnh 6:
lw s1, 4(sp) #Tải giá trị từ sp +4(5) vào thanh ghi s1 → s1 = 5

s1	9	5

Lệnh 7:
addi sp, sp, 8 # sp = sp +8, giải phóng bộ nhớ trên stack

sp	2	2147479548

Kết quả:

s0	8	6
s1	9	5

Assignment 4

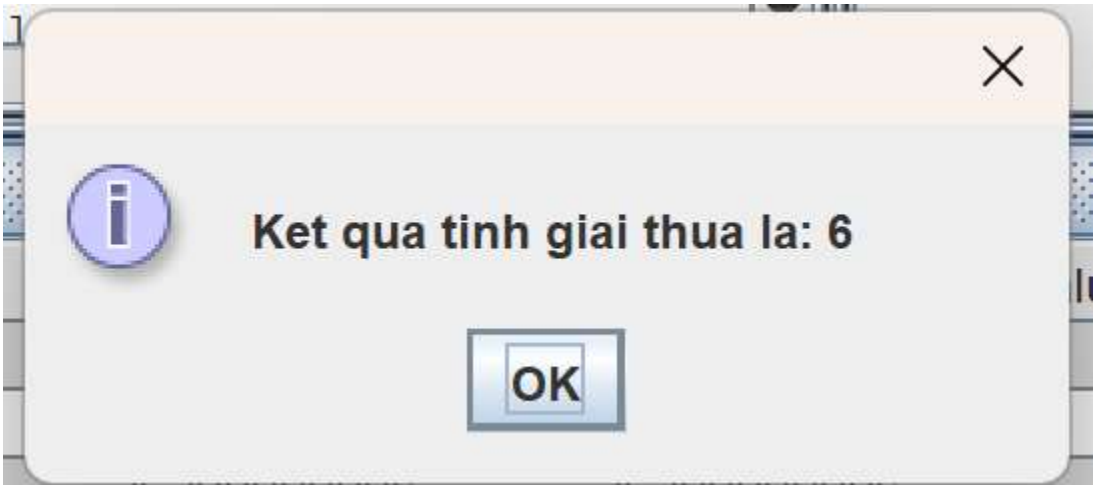
Tạo project để thực hiện Home Assignment 4. Dịch và chạy mô phỏng. Thay đổi tham số ở thanh ghi a0 và kiểm tra kết quả ở thanh ghi s0. Chạy chương trình ở chế độ từng dòng lệnh và quan sát sự thay đổi giá trị của các thanh ghi pc, ra, sp, a0, s0. Liệt kê các giá trị trong vùng nhớ ngăn xếp khi thực hiện chương trình với n = 3.

Chương trình thực hiện

```
# Laboratory Exercise 7, Home Assignment 4
.data
message: .asciz "Ket qua tinh giai thua la: "
.text
main:
jal WARP
print:
add a1, s0, zero # a0 = result from N!
li a7, 56
la a0, message
ecall
quit:
li a7, 10 # terminate
ecall
end_main:
# ---------------------------------------------------------------
# Procedure WARP: assign value and call FACT
# ---------------------------------------------------------------
WARP:
addi sp, sp, -4 # adjust stack pointer
```

```
sw ra, 0(sp) # save return address
li a0, 3 # load test input N
jal FACT # call fact procedure
lw ra, 0(sp) # restore return address
addi sp, sp, 4 # return stack pointer
jr ra
wrap_end:
# ------------------------------------------------------------------
# Procedure FACT: compute N!
# param[in] a0 integer N
# return s0 the largest value
# ------------------------------------------------------------------
FACT:
addi sp, sp, -8 # allocate space for ra, a0 in stack
sw ra, 4(sp) # save ra register
sw a0, 0(sp) # save a0 register
li t0, 2
bge a0, t0, recursive
li s0, 1 # return the result N!=1
j done
recursive:
addi a0, a0, -1 # adjust input argument
jal FACT # recursive call
lw s1, 0(sp) # load a0
mul s0, s0, s1
done:
lw ra, 4(sp) # restore ra register
lw a0, 0(sp) # restore a0 register
addi sp,sp,8 # restore stack pointer
jr ra # jump to caller
fact_end:
```

Kết quả với n=3



Giá trị trong vùng nhớ ngăn xếp khi thực hiện chương trình với n = 3:

bottom
0x00400004
0x00400030
3 (= a0)
0x00400060
2 (=a0)
0x00400060
1 (=a0)

<u>Assignment 5</u>

Viết chương trình con tìm giá trị lớn nhất, nhỏ nhất và vị trí tương ứng trong danh sách gồm 8 số nguyên được lưu trữ trong các thanh ghi từ a0 đến a7. Ví dụ: Largest: 9, 3 => Giá trị lớn nhất là 9 được lưu trữ trong a3 Smallest: -3, 6 => Giá trị nhỏ nhất là -3 được lưu trữ trong a6 Gợi ý: Sử dụng ngăn xếp để truyền tham số.

Chương trình thực hiện:

```
.data
message_largest: .asciz "largest: "
message_smallest: .asciz " \nSmallest: "

.text
main:
li a0, 56
li a1, -23
li a2, 44
li a3, 7
li a4, 3
li a5, -4
li a6, 4
li a7, -7

# Push registers a0-a7 onto the stack
addi sp, sp, -32    # Allocate 32 bytes (8 registers * 4 bytes each)
sw a0, 28(sp)     # Save a0 at top of allocated stack space
sw a1, 24(sp)     # Save a1
sw a2, 20(sp)     # Save a2
sw a3, 16(sp)     # Save a3
sw a4, 12(sp)     # Save a4
sw a5, 8(sp)     # Save a5
sw a6, 4(sp)     # Save a6
sw a7, 0(sp)     # Save a7

# Call subroutine to find min and max values
jal find_min_max

# Print the result for largest value
la a0, message_largest    # Load message address for largest
li a7, 4    # Print string syscall
ecall
```

```
ecall

mv a0, t0    # Load largest value
li a7, 1    # Print integer syscall
ecall

li a0, ','    # Print comma separator
li a7, 11    # Print character syscall
ecall

mv a0, t1    # Load largest position
li a7, 1    # Print integer syscall
ecall

# Print the result for smallest value
la a0, message_smallest    # Load message address for smallest
li a7, 4    # Print string syscall
ecall

mv a0, t2    # Load smallest value
li a7, 1    # Print integer syscall
ecall

li a0, ','    # Print comma separator
li a7, 11    # Print character sysoall
ecall

mv a0, t3    # Load smallest position
li a7, 1    # Print integer syscall
ecall

# Clean up stack and end program
addi sp, sp, 32    # Restore stack pointer
```

```
65  addi sp, sp, 32    # Restore stack pointer
66  li a7, 10    # Exit syscall
67  ecall
68
69  find_min_max:
70  # Initialize largest and smallest with the first value (equivalent to a0 at sp+28)
71  lw t0, 28(sp)    # t0 = largest value
72  li t1, 0    # t1 = position of largest (0 for a0)
73  lw t2, 28(sp)    # t2 = smallest value
74  li t3, 0    # t3 = position of smallest (0 for a0)
75
76  # Set up loop to iterate through stack values
77  li t4, 1    # Position counter (for stack values from a1 to a7)
78  li t5, 24    # Offset to next stack value (a1 stored at sp+24)
79
80  loop_check:
81  # Access current value from stack using offset (sp + t5)
82  add t5, sp, t5    # Access current value from stack using offset (sp + t5)
83  lw t6, 0(t5)    # Load value at current offset
84
85  # Check if current value is the new largest
86  bgt t6, t0, update_largest
87
88  # Check if current value is the new smallest
89  blt t6, t2, update_smallest
90
91  # If neither, continue to the next stack value
92  addi t5, t5, -4    # Move offset to the next stack value
93  addi t4, t4, 1    # Increment position counter
94  blt t4, s0, loop_check    # Continue if position < 8 (a0 to a7)
95
96  jr ra    # Return to caller
97
```

```
96  jr ra    # Return to caller
97
98  update_largest:
99  mv t0, t6    # Update largest value
100 mv t1, t4    # Update position of largest
101 j loop_check    # Continue to next value
102
103 update_smallest:
104 mv t2, t6    # Update smallest value
105 mv t3, t4    # Update position of smallest
106 j loop_check    # Continue to next value
```

Kết quả thu được: