

BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 7

Họ và tên: Nguyễn Quý Đức MSSV: 20235682

Mã HP: IT3280

Mã lớp: 156788

1. Assignment 1

Trường hợp $a0 = -45$

```
w7h1.asm
1  # Laboratory Exercise 7 Home Assignment 1
2  .text
3  main:
4      li a0, -45  # load input parameter
5      jal abs    # jump and link to abs procedure
6      li a7, 10   # terminate
7      ecall
8  end_main:
9  # -----
10 # function abs
11 # param[in] a0 the interger need to be gained the absolute value
12 # return s0 absolute value
13 # -----
14 abs:
15     sub s0, zero, a0    # put -a0 in s0; in case a0 < 0
16     blt a0, zero, done  # if a0<0 then done
17     add s0, a0, zero    # else put a0 in s0
18 done:
19 jr ra
```

Trạng thái	a0	s0	ra	pc
li a0, -45	-45	0	0x00000000	0x00400004
jal abs #mark	-	-	0x00400008	0x00400010
sub s0, a0, zero	-	45	-	0x00400014
blt a0, zero, done	-	-	-	0x0040001c
add s0, a0, zero	-	-	-	-
jr ra #jump to mark.next	-	-	-	0x00400008
li a7, a0	-	-	-	0x0040000c
ecall	-	-	-	0x00400010

Thanh ghi pc (program counter): Chứa địa chỉ lệnh tiếp theo sẽ thực thi. Khi thực hiện jal abs, pc nhảy đến địa chỉ của hàm abs, và khi jr ra được thực thi, pc quay lại địa chỉ đã lưu trong ra.

Thanh ghi ra (return address): Lưu địa chỉ lệnh ngay sau jal, để khi kết thúc hàm con abs, chương trình có thể quay lại đúng vị trí tiếp theo để tiếp tục thực thi. Trong trường hợp này, jal abs lưu pc = 0x00400008 vào ra, và jr ra đưa pc trở về địa chỉ này để tiếp tục chạy lệnh li a7, 10.

Trường hợp a0 = 12

```

w7h1.asm
1  # Laboratory Exercise 7 Home Assignment 1
2  .text
3  main:
4      li a0, 12    # load input parameter
5      jal abs     # jump and link to abs procedure
6      li a7, 10   # terminate
7      ecall
8  end_main:
9  # -----
10 # function abs
11 # param[in] a0 the interger need to be gained the absolute value
12 # return s0 absolute value
13 # -----
14 abs:
15     sub s0, zero, a0    # put -a0 in s0; in case a0 < 0
16     blt a0, zero, done  # if a0<0 then done
17     add s0, a0, zero    # else put a0 in s0
18 done:
19     jr ra

```

Trạng thái	a0	s0	ra	pc
li a0, -45	12	0	0x00000000	0x00400004
jal abs #mark	-	-	0x00400008	0x00400010
sub s0, a0, zero	-	-12	-	0x00400014
blt a0, zero, done	-	-	-	0x00400018
add s0, a0, zero	-	12	-	0x0040001c
jr ra #jump to mark.next	-	-	-	0x00400008
li a7, a0	-	-	-	0x0040000c
ecall	-	-	-	0x00400010

Truyền tham số a0 = 12, s0 chưa khởi tạo. Vì a0 = 12 > 0, nên không nhảy xuống done mà gán s0 = a0 = 12, sau đó dùng lệnh jr để nhảy đến địa chỉ được lưu trong \$ra. Cuối cùng, chương trình kết thúc.

2. Assignment 2

```

w7h1.asm  w7a2.asm*
1  .text
2  main:
3      li a0, 2 # load test input
4      li a1, 6
5      li a2, 9
6      jal max # call max procedure
7      li a7, 10 # terminate
8      ecall
9  end_main:
10 max:
11     add s0, a0, zero # copy a0 in s0; Largest so far
12     sub t0, a1, s0 # compute a1 - s0
13     blt t0, zero, okay # if a1 - s0 < 0 then no change
14     add s0, a1, zero # else a1 is largest thus far
15 okay:
16     sub t0, a2, s0 # compute a2 - v0
17     blt t0, zero, done # if a2 - v0 < 0 then no change
18     add s0, a2, zero # else a2 is largest overall
19 done:
20     jr ra # return to calling program
21

```

Trạng thái	a0	a1	a2	s0	t0	ra	pc
li a0, 2							0x00400004
li a1, 6	2	6	9	0	0	0x00000000	0x00400008
li a2, 9							0x0040000c
jal max	-	-	-	-	-	0x40000010	0x40000018
add s0, a0, zero				2	-		0x4000001c
sub t0, a1, s0				-	4		0x40000020
blt t0, zero, okay	-	-	-	-	-	-	0x40000024
add s0, a1, zero				6	-		0x40000028
sub t0, a2, s0				-	3		0x4000002c
blt t0, zero, done	-	-	-	-	-	-	0x40000030
add s0, a2, zero				9	-		0x40000034
jr ra	-	-	-	-	-	-	0x40000010
li a7, 10							0x40000014
ecall	-	-	-	-	-	-	0x40000018

Ban đầu, pc thực thi jal max, lưu địa chỉ tiếp theo (0x0040000c) vào ra, rồi nhảy đến max. Sau khi tìm số lớn nhất trong 3 số, jr ra đưa pc trở về 0x0040000c để tiếp tục chương trình.

Thay bộ số (a0, a1, a2) = (42, 12, 94):

```

w7h1.asm  w7a2.asm
1  .text
2  main:
3      li a0, 42 # Load test input
4      li a1, 12
5      li a2, 94
6      jal max # call max procedure
7      li a7, 10 # terminate
8      ecall
9  end_main:
10 max:
11     add s0, a0, zero # copy a0 in s0; Largest so far
12     sub t0, a1, s0 # compute a1 - s0
13     blt t0, zero, okay # if a1 - s0 < 0 then no change
14     add s0, a1, zero # else a1 is largest thus far
15 okay:
16     sub t0, a2, s0 # compute a2 - v0
17     blt t0, zero, done # if a2 - v0 < 0 then no change
18     add s0, a2, zero # else a2 is largest overall
19 done:
20     jr ra # return to calling program
21

```

Trạng thái	a0	a1	a2	s0	t0	ra	pc
li a0, 2							0x00400004
li a1, 6	42	12	94	0	0	0x00000000	0x00400008
li a2, 9							0x0040000c
jal max	-	-	-	-	-	0x40000010	0x40000018
add s0, a0, zero				42	-		0x4000001c
sub t0, a1, s0	-	-	-	-	-30	-	0x40000020
blt t0, zero, okay				-	-		0x40000028
sub t0, a2, s0				-	-52		0x4000002c
blt t0, zero, done	-	-	-	-	-	-	0x40000030
add s0, a2, zero				94	-		0x40000034
jr ra	-	-	-	-	-	-	0x40000010
li a7, 10							0x40000014
ecall							0x40000018

3. Assignment 3

```

w7h1.asm  w7a2.asm  w7h3.asm
1  # Laboratory Exercise 7, Home Assignment 3
2  .text
3  init:
4      li s0, 4
5      li s1, 3
6
7  push:
8      addi sp, sp, -8 # adjust the stack pointer
9      sw s0, 4(sp) # push s0 to stack
10     sw s1, 0(sp) # push s1 to stack
11 work:
12     nop
13     nop
14     nop
15 pop:
16     lw s0, 0(sp) # pop from stack to s0
17     lw s1, 4(sp) # pop from stack to s1
18     addi sp, sp, 8 # adjust the stack pointer

```

Giá trị ban đầu của $(s0, s1) = (4, 3)$

s0	8	4
s1	9	3

Thanh ghi sp ban đầu ở giá trị mặc định

sp	2	0x7ffefffc
----	---	------------

Cần đẩy 2 số s0, s1 vào stack nên gán $sp = sp - 8$.

sp	2	0x7ffefff4
----	---	------------

Khối push: Gán giá trị s0 cho thanh ghi tại địa chỉ $sp+4$ và giá trị s1 cho thanh ghi tại địa chỉ sp, tức là đẩy $s0 = 4$ vào trước, $s1 = 3$ vào sau.

Value (+14)	Value (+18)
3	4

Khối pop: Gán giá trị tại thanh ghi $0(sp)$ vào thanh ghi s0, tức là lấy phần tử ở đỉnh stack là -3 gán vào s0, sau đó gán giá trị còn lại là 2

trong stack vào thanh ghi s1. Như vậy, dùng stack ta đã hoán đổi được giá trị của 2 thanh ghi s0 và s1. Sau khi kết thúc, ta lấy $sp + 8$ để trả lại giá trị ban đầu cho thanh ghi sp.



4.Assignment 4

Trạng thái	pc	ra	sp	a0	s0
ban đầu	0x00400000	0x00000000	0x7ffffeffc	0x00000000	0x00000000
jal WARP	0x00400020	0x00400004			
addi sp, sp, -4	0x00400024		0x7ffffeff8		
sw ra, 0(sp)	0x00400028				
li a0, 3	0x0040002c			0x00000003	
jal FACT	0x0040003c	0x00400030			
addi sp, sp, -8	0x00400040		0x7ffffeff0		
sw ra, 4(sp)	0x00400044				
sw a0, 0(sp)	0x00400048				
li t0, 2	0x0040004c				
bge a0, t0, recursive	0x00400058				
addi a0, a0, -1	0x0040005c			0x00000002	
jal FACT	0x0040003c	0x00400060			
addi sp, sp, -8	0x00400040		0x7ffffefe8		
sw ra, 4(sp)	0x00400044				
sw a0, 0(sp)	0x00400048				
li t0, 2	0x0040004c				
bge a0, t0, recursive	0x00400058				
addi a0, a0, -1	0x0040005c			0x00000001	
jal FACT	0x0040003c	0x00400060			
addi sp, sp, -8	0x00400040		0x7ffffefe0		
sw ra, 4(sp)	0x00400044				
sw a0, 0(sp)	0x00400048				
li t0, 2	0x0040004c				
bge a0, t0, recursive	0x00400050				
li s0, 1	0x00400054				0x00000001
j done	0x00400068				
lw ra, 4(sp)	0x0040006c	0x00400060			
lw a0, 0(sp)	0x00400070			0x00000001	
addi sp, sp, 8	0x00400074		0x7ffffefe8		
jr ra	0x00400060				
lw s1, 0(sp)	0x00400064				
mul s0, s0, s1	0x00400068				0x00000002
lw ra, 4(sp)	0x0040006c	0x00400060			
lw a0, 0(sp)	0x00400070			0x00000002	
addi sp, sp, 8	0x00400074		0x7ffffeff0		

jr ra	0x00400060				
lw s1, 0(sp)	0x00400064				
mul s0, s0, s1	0x00400068				0x00000006
lw ra, 4(sp)	0x0040006c	0x00400030			
lw a0, 0(sp)	0x00400070			0x00000003	
addi sp,sp,8	0x00400074		0x7ffffeff8		
jr ra	0x00400030				
lw ra, 0(sp)	0x00400034	0x00400004			
addi sp,sp,4	0x00400038		0x7ffffeffc		
jr ra	0x00400004				
add a1, s0, zero	0x00400008				
li a7, 56	0x0040000c				
la a0, message	0x00400010			0x10010000	
ecall	0x00400014				
li a7, 10	0x00400018				
ecall	0x0040001c				

Giá trị của vùng nhớ stack: (n = a0 = 3)

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)
0x7ffffefe0	0x00000001	0x00400060	0x00000002	0x00400060	0x00000003	0x00400030	0x00400004

5. Assignment

```

.data
min_msg: .string "min = "
at_msg:  .string ", at "
max_msg: .string "max = "
newline: .string "\n"

.text
main:
    li a0, 1
    addi sp, sp, -4
    sw a0, 0(sp)

    li a1, 2
    addi sp, sp, -4
    sw a1, 0(sp)

    li a2, 3
    addi sp, sp, -4
    sw a2, 0(sp)

    li a3, 4
    addi sp, sp, -4
    sw a3, 0(sp)

    li a4, 5
    addi sp, sp, -4
    sw a4, 0(sp)

    li a5, 8
    addi sp, sp, -4
    sw a5, 0(sp)

    li a6, 6
    addi sp, sp, -4
    sw a6, 0(sp)

    li a7, 7
    addi sp, sp, -4
    sw a7, 0(sp)

    addi sp, sp, -24
    mv s0, a0      # min.value
    mv s1, a0      # max.value
    li s2, 0 # min.index
    li s3, 0 # max.index
    li s4, 1      # index
    bge a1, s0, not_min_1
    mv s0, a1

```

```

    mv s2, s4
not_min_1:
    ble a1, s1, not_max_1
    mv s1, a1
    mv s3, s4
not_max_1:
    addi s4, s4, 1
    bge a2, s0, not_min_2
    mv s0, a2
    mv s2, s4
not_min_2:
    ble a2, s1, not_max_2
    mv s1, a2
    mv s3, s4
not_max_2:
    addi s4, s4, 1
    bge a3, s0, not_min_3
    mv s0, a3
    mv s2, s4
not_min_3:
    ble a3, s1, not_max_3
    mv s1, a3
    mv s3, s4
not_max_3:
    addi s4, s4, 1
    bge a4, s0, not_min_4
    mv s0, a4
    mv s2, s4
not_min_4:
    ble a4, s1, not_max_4
    mv s1, a4
    mv s3, s4
not_max_4:
    addi s4, s4, 1
    bge a5, s0, not_min_5
    mv s0, a5
    mv s2, s4
not_min_5:
    ble a5, s1, not_max_5
    mv s1, a5
    mv s3, s4
not_max_5:
    addi s4, s4, 1
    bge a6, s0, not_min_6
    mv s0, a6
    mv s2, s4
not_min_6:

```

```

    ble a6, s1, not_max_6
    mv s1, a6
    mv s3, s4
not_max_6:
    addi s4, s4, 1
    bge a7, s0, not_min_7
    mv s0, a7
    mv s2, s4
not_min_7:
    ble a7, s1, not_max_7
    mv s1, a7
    mv s3, s4
not_max_7:
    addi s2, s2, 1
    addi s3, s3, 1
    la a0, min_msg
    li a7, 4
    ecall
    mv a0, s0
    li a7, 1
    ecall
    la a0, at_msg
    li a7, 4
    ecall
    mv a0, s2
    li a7, 1
    ecall
    la a0, newline
    li a7, 4
    ecall
    la a0, max_msg
    li a7, 4
    ecall
    mv a0, s1
    li a7, 1
    ecall
    la a0, at_msg
    li a7, 4
    ecall
    mv a0, s3
    li a7, 1
    ecall
    lw ra, 0(sp)
    li a7, 10
    ecall

```


Kết quả thực hiện lệnh:

Messages	Run I/O
<div>Clear</div>	-- program is finished running (0) --
	min = 1, at 1
	max = 8, at 6
	-- program is finished running (0) --