Assignment 1

Mã nguồn:

```
#Laboratory Exercise 7 Home Assignment 1
.text
main: li $a0,-20194484 #load input parameter
 jal abs #jump and link to abs procedure
 nop
 add $s0, $zero, $v0
 li $v0,10 #terminate
 syscall
endmain:
#---------------------------------------------------------------
# function abs
# param[in] $a1 the interger need to be gained the absolute
value:
# return $v0 absolute value
#---------------------------------------------------------------
abs:
 add $a1, $zero, $a0
 sub $v0,$zero,$a1 #put -(a0) in v0; in case (a0)<0

 bltz $a1,done #if (a0)<0 then done
 nop
 add $v0,$a1,$zero #else put (a0) in v0
done:
 jr $ra
```

Giải thích:

- Câu lệnh `li $a0, -20194484` thực hiện load giá trị vào thanh ghi $a0.

$a0	4	0xfecbdb4c

- Sau đó, lệnh `jal abs` nhảy đến thủ tục abs. Lúc này, thanh ghi $pc và lệnh tiếp ngay sau câu lệnh $pc có giá trị

0x00400008	0x0c100007	jal 0x0040001c
0x0040000c	0x00000000	nop

.

- Sau khi thực hiện `jal`, thanh ghi $pc sẽ nhảy đến địa chỉ của thủ tục abs, thanh ghi $ra lưu giá trị của lệnh ngay bên dưới lệnh `jal`

$ra	31	0x0040000c
pc		0x0040001c

- *(Ở thủ tục abs, để code hoạt động đúng thì chúng ta phải sao chép giá trị thanh ghi $a0 vào thanh ghi $a1 bằng lệnh `add $a1, $zero, $a0`).*

```
add $a1, $zero, $a0
sub $v0,$zero,$a1 #put -(a0) in v0; in case (a0)<0

bltz $a1,done #if (a0)<0 then done
nop
add $v0,$a1,$zero #else put (a0) in v0
```

- Ở trong thủ tục abs:
  - Câu lệnh `sub $v0, $zero, $a0` thực hiện gán giá trị -a0 cho thanh ghi $v0.

  - Lệnh `bltz $a1, done` thực hiện so sánh giá trị thanh ghi $a1 với 0. Nếu $a1 < 0 thì thực hiện nhảy đến nhãn *done. (Thanh ghi $a1 = -20194484 < 0 nên chương trình sẽ nhảy đến nhãn done, thoát khỏi thủ tục abs)*

- Nhãn done chứa câu lệnh `jr $ra`. Khi thực hiện lệnh này, $pc sẽ nhận giá trị của thanh ghi $ra.

- Thực hiện sao chép giá trị thanh ghi $v0 vào thanh ghi $s0 bằng câu lệnh `add $s0, $zero, $v0`.

- Chương trình kết thúc bằng lời gọi *syscall* với $v0 = 10$.

Assignment 2

Mã nguồn:

```
#Laboratory Exercise 7, Home Assignment 2
.text
main: li $a0,4 #load test input
 li $a1,8
 li $a2,4
 jal max #call max procedure
 nop

 add $s0, $zero, $v0
 li $v0,10 #terminate
 syscall
endmain:
#------------------------------------------------------------------

#Procedure max: find the largest of three integers
#param[in] $a0 integers
#param[in] $a1 integers
#param[in] $a2 integers
#return $v0 the largest value
#------------------------------------------------------------------

max: add $v0,$a0,$zero #copy (a0) in v0; largest so far
 sub $t0,$a1,$v0 #compute (a1)-(v0)
 bltz $t0,okay #if (a1)-(v0)<0 then no change
 nop
 add $v0,$a1,$zero #else (a1) is largest thus far
okay: sub $t0,$a2,$v0 #compute (a2)-(v0)
 bltz $t0,done #if (a2)-(v0)<0 then no change
 nop
 add $v0,$a2,$zero #else (a2) is largest overall
done: jr $ra #return to calling program
```

Giải thích:

- 3 câu lệnh đầu tiên thực hiện load 3 giá trị input. $a0 = 4, $a1 = 8, $a2 = 4.

$a0	4	0x00000004
$a1	5	0x00000008
$a2	6	0x00000004

- Lệnh `jal max` nhảy đến thủ tục max. Lúc này thanh ghi $ra lưu giá trị của lệnh kế tiếp lệnh *jal*

0x0040000c	0x0c100008	jal 0x00400020
0x00400010	0x00000000	nop

.
Thanh ghi $pc nhảy đến địa chỉ của nhãn *max*.

$ra	31	0x00400010
pc		0x00400020

- Trong thủ tục max:

```
max: add $v0,$a0,$zero #copy (a0) in v0; largest so far
 sub $t0,$a1,$v0 #compute (a1)-(v0)
 bltz $t0,okay #if (a1)-(v0)<0 then no change
 nop
 add $v0,$a1,$zero #else (a1) is largest thus far
okay: sub $t0,$a2,$v0 #compute (a2)-(v0)
 bltz $t0,done #if (a2)-(v0)<0 then no change
 nop
 add $v0,$a2,$zero #else (a2) is largest overall
```

- Đầu tiên, coi $a0 là giá trị lớn nhất tạm thời bằng cách dùng lệnh `add $v0, $a0, $zero`.

$v0	2	0x00000004

- Sau đó, dùng lệnh
- Sau đó, so sánh $v0 và $a1 bằng câu lệnh `sub $t0, $a1, $v0`. Nếu $v0 > $a1 thì rẽ sang nhãn *okay*. Nếu không thì gán giá trị $a1 là giá trị lớn nhấn tạm thời. *(Ở đây, do $a0 > $a1 nên chương trình rẽ sang nhãn okay)*
- Nhãn *okay* thực hiện các bước tương tự, so sánh giá trị lớn nhất tạm thời $v0 với giá trị của $a2. Do $v0 < $a2 nên thực hiện gán giá trị $a2 cho thanh ghi $v0.

$v0	2	0x00000008

- Đến nhãn *done*, thủ tục kết thúc. Lệnh `jr $ra` được thực hiện. Thanh ghi $pc lúc này có giá trị bằng với $ra.

$ra	31	0x00400010
pc		0x00400010

- *Ta phải thêm 3 câu lệnh sau vào dưới lệnh mà `jr` nhảy đến để chương trình có thể kết thúc được. Giá trị trả về được lưu vào thanh ghi $s0.*

```
add $s0, $zero, $v0
li $v0,10 #terminate
syscall
```

Assignment 3

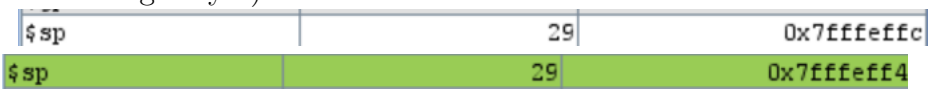
Mã nguồn:

```
#Laboratory Exercise 7, Home Assignment 3
.text
addi $s0, $s0, 2019 # $s0 = 4 so dau mssv
addi $s1, $s1, 4484 # $s1 = 4 so cuoi mssv
push: addi $sp,$sp,-8 #adjust the stack pointer
 sw $s0,4($sp) #push $s0 to stack
 sw $s1,0($sp) #push $s1 to stack
work:
add $s0, $zero, $zero # $s0 = 0
add $s1, $zero, $zero # $s1 = 0
pop: lw $s0,0($sp) #pop from stack to $s0
 lw $s1,4($sp) #pop from stack to $s1
 addi $sp,$sp,8 #adjust the stack pointer
```

Giải thích:

- 2 câu lệnh đầu tiên thực hiện khởi tạo giá trị cho 2 thanh ghi $s0 = 2019, $s1 = 4484.

- Đến nhãn *push*, ta thực hiện nạp nội dung 2 thanh ghi $s0, $s1 vào ngăn xếp. Câu lệnh `addi $s1, $sp, -8` thực hiện giảm giá trị của thanh ghi $sp 8 byte. Tạo ra 8 byte trống dùng để lưu 2 biến. (Mỗi biến dùng 4 byte)
  Giá trị ban đầu của $sp:
  Giá trị sau đó của $sp:



- 2 câu lệnh `sw` tiếp theo dùng để lưu 2 giá trị $s0, $s1 vào 2 vị trí chúng ta đã tạo ra trong ngăn xếp.

- Đến nhãn *work*, chúng ta thực hiện thay đổi giá trị 2 thanh ghi $s0, $s1 thành 0 bằng cách dùng lệnh `add $s0, $zero, $zero` và `add $s1, $zero, $zero`.

- Ở nhãn *pop*, sau khi thay đổi giá trị 2 thanh ghi $s0, $s1 về 0, chúng ta tiến hành khôi phục lại giá trị ban đầu bằng cách sử dụng 2 lệnh `lw`, load lại dữ liệu đã được lưu trong ngăn xếp vào trong 2 thanh ghi $s0, $s1.

- Cuối cùng, thực hiện giải phóng ngăn xếp bằng cách sử dụng lệnh `addi $sp, $sp, 8`, cộng 8 byte lại cho ngăn xếp.
  Giá trị ban đầu của $sp:
  Giá trị sau đó của $sp: