Báo cáo tuần 7
Thực hành kiến trúc máy tính

Họ tên: Phan Minh Anh Tuấn
MSSV: 20205227

Mục lục

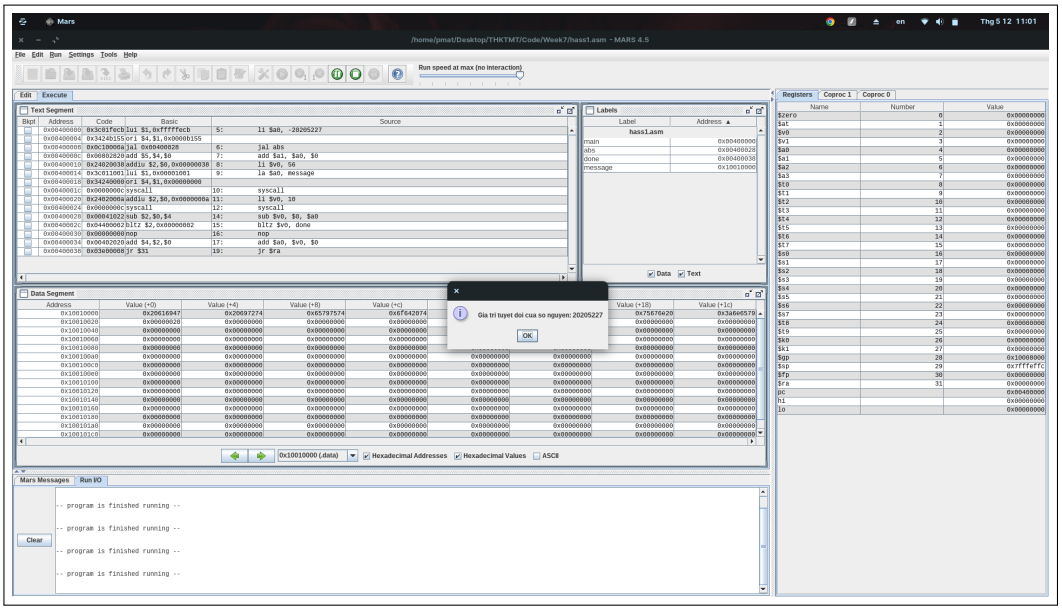
1 Assignment 1

```
.data
message: .asciiz "Gia tri tuyet doi cua so nguyen: "
.text
main:
        li $a0, -20205227
        jal abs
        add $a1, $a0, $0
        li $v0, 56
        la $a0, message
        syscall
        li $v0, 10
        syscall
abs:
        sub $v0, $0, $a0
        bltz $v0, done
        nop
        add $a0, $v0, $0
done:
        jr $ra
```

Hình 1: Code của Assignment 1

Giải thích: Dây là chương trình trả về giá trị tuyệt đối số đầu vào

- Giá trị đầu vào được lưu vào $a0.
- Nhảy đến hàm abs, lưu địa chỉ của câu lệnh ngay dưới vào $ra.
- Thực hiện lấy 0 - giá trị $a0 lưu vào $v0. So sánh nếu $v0 < 0 thì giữ nguyên giá trị $a0, ngược lại lưu giá trị $a0 là giá trị $v0 hiện thời.



Hình 2: Kết quả của Assignment 1

2 Assignment 2

```
.data
message: .asciiz "So nguyen lon nhat: "
.text
main:
        li $a0, 20
        li $a1, 40
        li $a2, 6
        jal max
        li $v0, 56
        la $a0, message
        add $a1, $s0, $0
        syscall
        li $v0, 10
        syscall
max:
        add $s0, $a0, $0
        sub $t0, $a1, $s0
        bltz $t0, continue
        nop
        add $s0, $a1, $0
continue:
        sub $t0, $a2, $s0
        bltz $t0, end
        nop
        add $s0, $a2, $0
end:
        jr $ra
```

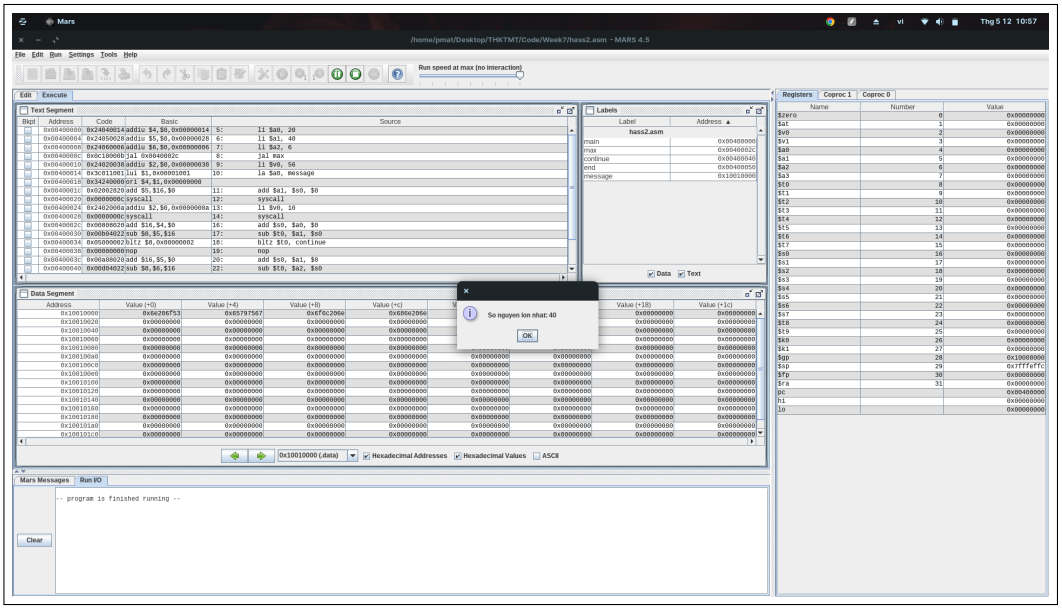
Hình 3: Code của Assignment 2

Giải thích: Đây là chương trình trả về số nguyên lớn nhất trong 3 đầu vào

- Giá trị đầu vào được lưu vào $a0, $a1, $a2.
- Nhảy đến hàm max, giá trị lớn nhất được lưu vào thanh ghi $s0. Khởi tạo giá trị đầu của $s0 = $a0.

- Tiến hành so sáng $a1 với $s0. Nếu $a1 > $s0 thì tiến hành gán $s0 = $a1, ngược lại vẫn giữ nguyên giá trị $s0. Tiếp tục nhảy đến hàm continue để so sánh $s0 và $a2.

- Sau khi chạy đến hàm end. Chương trình quay về địa chỉ ngay sau lần nhảy đầu tiên (jal max) để in ra số lớn nhất ra màn hình.



Hình 4: Kết quả của Assignment 2

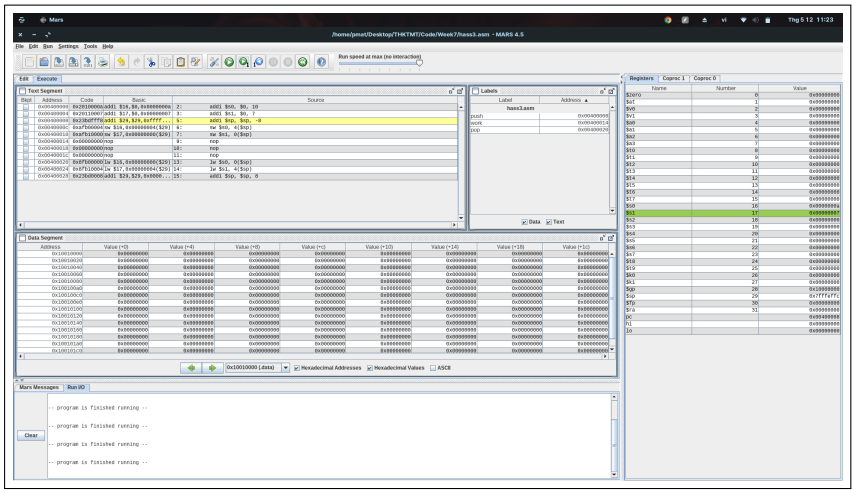
3 Assignment 3

```
.text
        addi $s0, $0, 10
        addi $s1, $0, 7
push:
        addi $sp, $sp, -8
        sw $s0, 4($sp)
        sw $s1, 0($sp)
work:
        nop
        nop
        nop
pop:
        lw $s0, 0($sp)
        lw $s1, 4($sp)
        addi $sp, $sp, 8
```

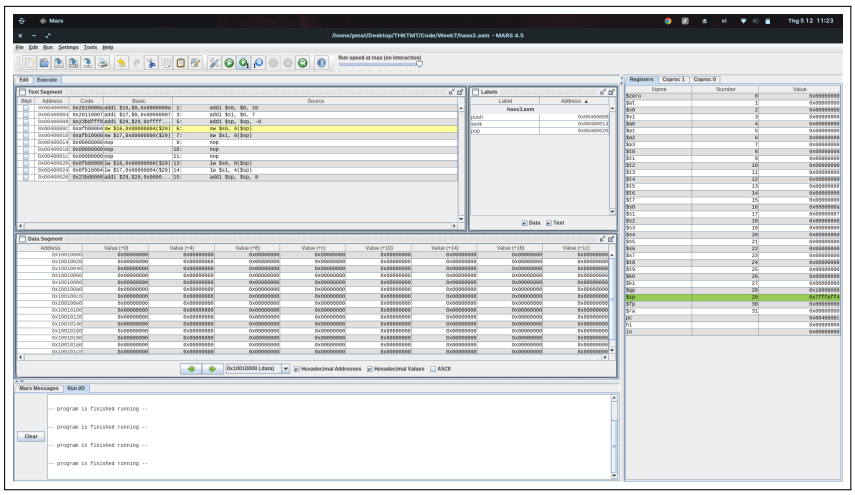
Hình 5: Code của Assignment 3

Giải thích: Chương trình swap giá trị của $s0 $s1 bằng stack.

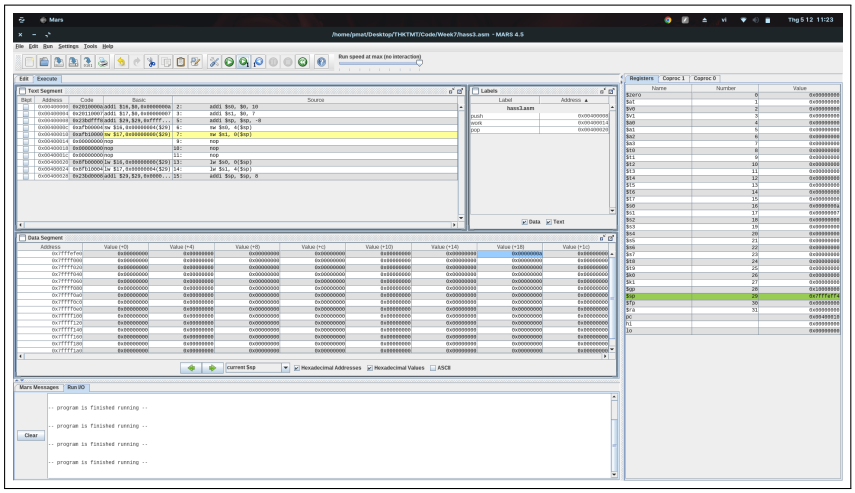
- Giảm nội dung của con trỏ ngăn xếp xuống 8 để lưu thêm 2 phần tử. Sau đó lưu giá trị $s0 và $s1 vào ngăn xếp.
- Load lại giá trị của $s0 và $s1 từ ngăn xếp.



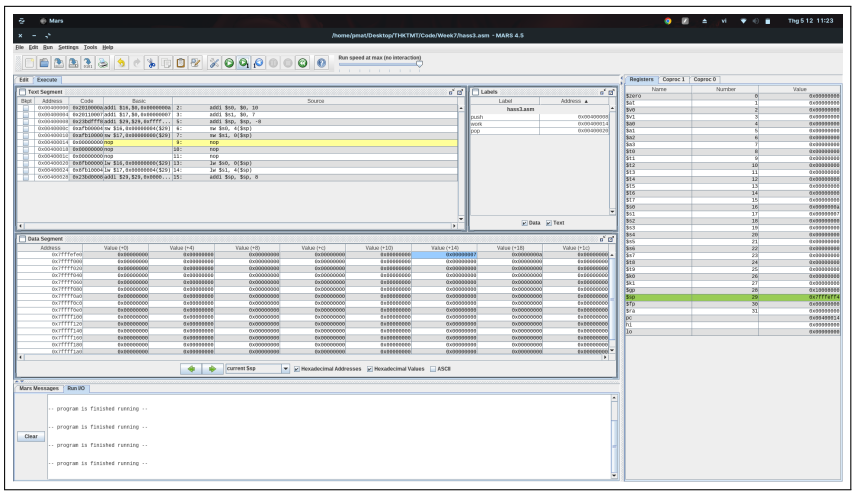
Step 1: Con trỏ $sp đang có giá trị 0x7fffeffc



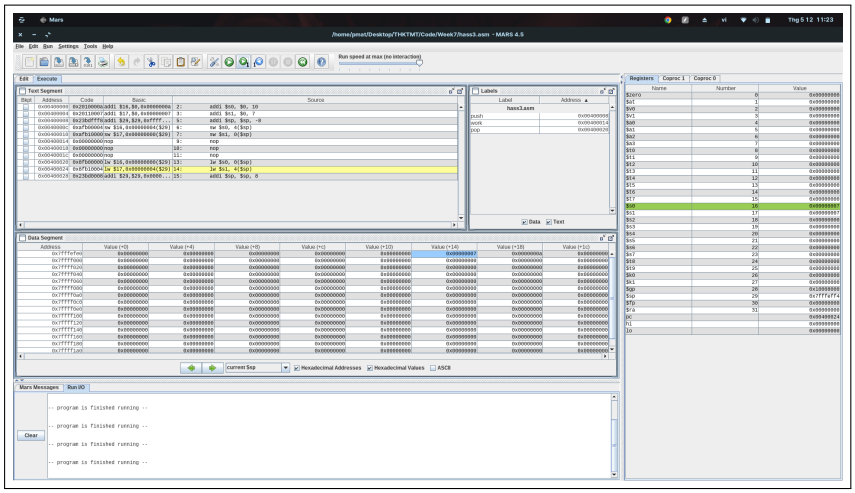
Step 2: Con trỏ $sp chuyển thành giá trị 0x7fffeff4



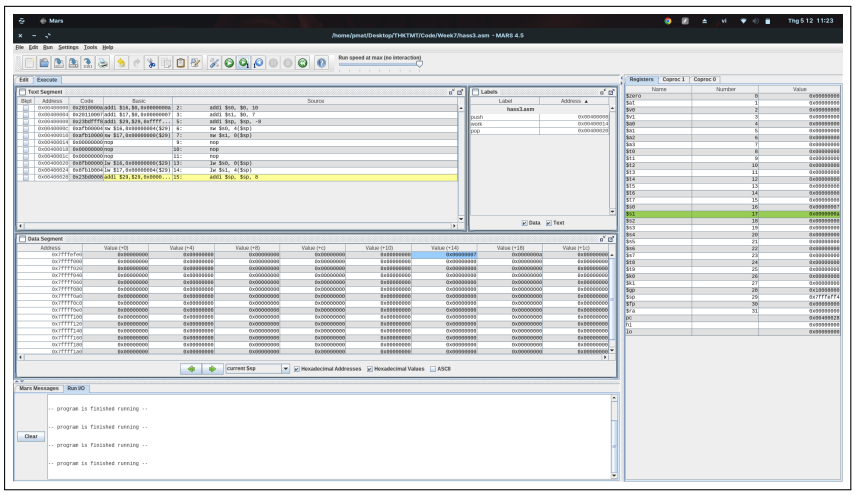
Step 3: Lưu giá trị $s0 vào ngăn xếp



Step 4: Lưu giá trị $s1 vào ngăn xếp



Step 5: Load giá trị $s0 từ ngăn xếp



Step 6: Load giá trị $s1 từ ngăn xếp

4 Assignment 4

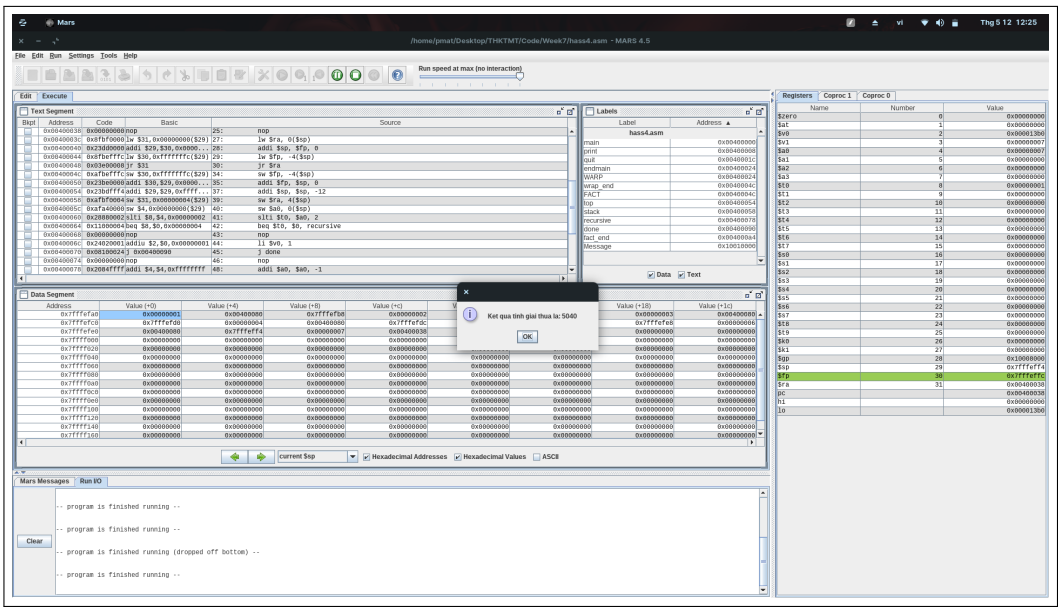
```
.data
Message: .asciiz "Ket qua tinh giai thua la: "

.text
main:   li $a0, 7
        jal WARP

print:
        add $a1, $v0, $0
        li $v0, 56
        la $a0, Message
        syscall
quit:
        li $v0, 10
        syscall
endmain:

WARP:
        sw $fp, -4($sp)
        addi $fp, $sp, 0
        addi $sp, $sp, -8
        sw $ra, 0($sp)

        jal FACT
        nop

        lw $ra, 0($sp)
        addi $sp, $fp, 0
        lw $fp, -4($sp)
        jr $ra
wrap_end:
```

```
FACT:
        sw $fp, -4($sp)
        addi $fp, $sp, 0
top:
        addi $sp, $sp, -12
stack:
        sw $ra, 4($sp)
        sw $a0, 0($sp)
        slti $t0, $a0, 2
        beq $t0, $0, recursive
        nop
        li $v0, 1
        j done
        nop
recursive:
        addi $a0, $a0, -1
        jal FACT
        nop
        lw $v1, 0($sp)
        mult $v1, $v0
        mflo $v0
done:
        lw $ra, 4($sp)
        lw $a0, 0($sp)
        addi $sp, $fp, 0
        lw $fp, -4($sp)
        jr $ra
fact_end:
```

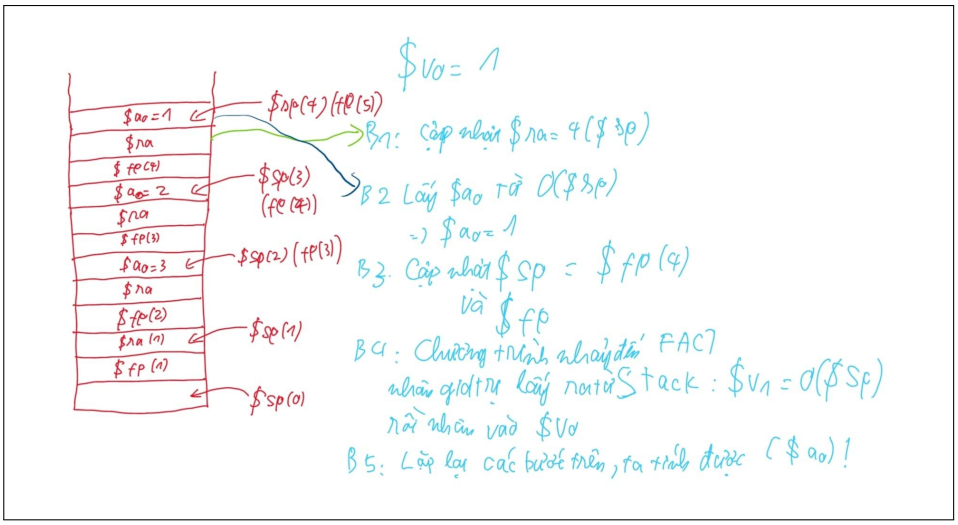
Hình 6: Code của Assignment 4

Giải thích: Chương trình trả về kết quả của ($a0)! Đoạn code dưới đây tương ứng với $a0 = 7.

Ý tưởng: Lưu $a0, $fp vào stack sau đó giảm giá trị $a0 = $a0 - 1. Khi $a0 < 2 thì dừng quá trình lặp, chuyển sang bước lấy giá trị từ stack để nhân vào $v0.



Hình 7: Kết quả của Assignment 4



Hình 8: Minh họa với trường hợp n = 3

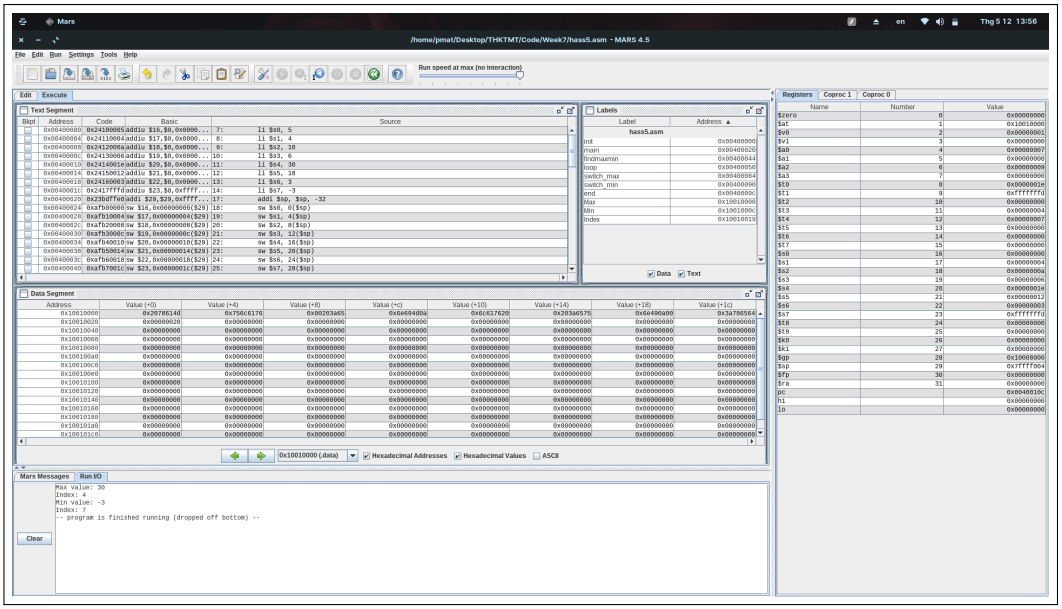
5 Assignment 5

```
.data
Max: .asciiz "Max value: "
Min: .asciiz "\nMin value: "
Index: .asciiz "\nIndex: "
.text
init:
        li $s0, 5
        li $s1, 4
        li $s2, 10
        li $s3, 6
        li $s4, 30
        li $s5, 18
        li $s6, 3
        li $s7, -3

main:
        addi $sp, $sp, -32
        sw $s0, 0($sp)
        sw $s1, 4($sp)
        sw $s2, 8($sp)
        sw $s3, 12($sp)
        sw $s4, 16($sp)
        sw $s5, 20($sp)
        sw $s6, 24($sp)
        sw $s7, 28($sp)
findmaxmin:
        #a0: max index
        #a1: min index
        #a2: index
        #t0: max value
        #t1: min value
        add $t0, $s0, $0
        add $t1, $s0, $0
        addi $a2, $a2, -1
loop:
        lw $s0, 0($sp)
        addi $sp, $sp, 4
        addi $a2, $a2, 1
        bgt $a2, 8, end
        blt $t0, $s0, switch_max
        nop
        blt $s0, $t1, switch_min
        nop
```

Ý tưởng: Dem các số vào trong stack, tạo 4 biến để lưu trữ giá trị lớn nhất, giá trị nhỏ nhất và index của chúng. Khi push ra từ stack ta sẽ tiến hành so sánh với các giá trị nhỏ nhất và lớn nhất hiện thời để cập nhật.

```
switch_max:
        add $t0, $s0, $0
        add $t3, $a2, $0
        j loop
switch_min:
        add $t1, $s0, $0
        add $t4, $a2, $0
        j loop
end:
        li $v0, 4
        la $a0, Max
        syscall
        li $v0, 1
        add $a0, $0, $t0
        syscall
        li $v0, 4
        la $a0, Index
        syscall
        li $v0, 1
        add $a0, $0, $t3
        syscall
        li $v0, 4
        la $a0, Min
        syscall
        li $v0, 1
        add $a0, $0, $t1
        syscall
        li $v0, 4
        la $a0, Index
        syscall
        li $v0, 1
        add $a0, $0, $t4
        syscall
```

Hình 9: Code của Assignment 5



Hình 10: Kết quả của Assignment 5