ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA

Computer Architecture (CO2008)

Bài báo cáo Chapter 2

Họ và tên Nguyễn Phúc Tiến
MSSV 2014725
Lớp L03

Ngày 14 tháng 3 năm 2022

Mục lục

1 Chapter 2.1 MIPS_ISA_Arithmetic

1.1 Bài 1: Syscall

a. Viết chương trình nhập vào 3 số nguyên a, b, c rồi xuất ra màn hình giá trị của hàm. f(a,b,c) = (a - b) + c.
- Đầu tiên ta sẽ tạo các string trong .data để giao tiếp với người dùng như sau (ứng với 3 câu a, b và c)

```
.data
    message1: .asciiz "question a)\nenter a: "
    message2: .asciiz "enter b: "
    message3: .asciiz "enter c: "
    message4: .asciiz"f(a,b,c) = (a - b) + c = "
```

- Tiếp đến phần .text ta dùng thanh ghi $v0$ và load giá trị 4 để print string.

```
li $v0, 4
la $a0, message1
syscall
```

- Sau đó ta tiếp tục dùng thanh ghi $v0$ và load cho nó giá trị 5 để nhập số nguyên a, b, c vào và lần lượt gán vào các thanh ghi $t0$, $t1$ và $t2$.

```
li $v0, 5
syscall
move $t0, $v0
```

- Dựa vào f(a, b, c) đã cho, ta sử dụng lệnh sub và add và gán kết quả vào thanh ghi $t3$.

```
sub $t3, $t0, $t1
add $t3, $t3, $t2
```

- Tiếp tục dùng đến thanh ghi $v0$ và load giá trị 1 để xuất ra kết quả là số nguyên.

```
li $v0, 1
move $a0, $t3
syscall
```

b. Viết chương trình xuất ra chuỗi "Kien Truc May Tinh 2020". (giống ví dụ HelloWorld!)
- Đầu tiên ta tạo string đã cho trong .data.

```
.data
    messageb: .asciiz "\nquestion b)\nKien Truc May Tinh 2020\n"
```

- Dùng thanh ghi $v0$ và load giá trị 4 để print ra string trên.

```
li $v0, 4
la $a0, messageb
syscall
```

c. Viết chương trình đọc vào một chuỗi 10 ký tự sau đó xuất ra màn hình chuỗi ký tự đó.
- Tương tự các câu trên, ta dùng thanh ghi $v0$ và load giá trị 8 để đọc chuỗi string từ người dùng và load giá trị 4 để in ra chuỗi string đó.

```
.data
    chuoi: .asciiz
```

Dưới phần text ta sẽ dùng thanh ghi $v0$ với các giá trị load đã nêu trên. Tuy nhiên khi load giá trị 8 thì ta cần thêm 1 dòng argument $a1$ để thể hiện độ dài của chuỗi, ở đây là 11 vì kết thúc chuỗi sẽ có kí tự \n.

```
li $v0, 8
la $a0, chuoi
la $a1, 11
syscall


li $v0, 4
la $a0, chuoi
syscall
```

* Kết quả bài 1

```
question a)
enter a: 3
enter b: 4
enter c: 5
f(a,b,c) = (a - b) + c = 4
question b)
Kien Truc May Tinh 2020
question c)
Nhap chuoi 10 ki tu: KT MayTinh
Chuoi da nhap la: KT MayTinh
```

1.2 Bài 2: Các lệnh số học luận lý

(a) Viết chương trình dùng các lệnh add, addi, sub, subi, or, ori . . . để thực hiện phép tính. Kết quả chứa vào thanh ghi $s0 và xuất kết quả ra màn hình (console).
- Ta lưu các số vào các thanh ghi $t0$, $t1$, $t2$ và sau đó sử dụng lệnh phù hợp để giải quyết bài toán.

```
addi $t0, $zero, 100000
addi $t1, $zero, 5000
addi $t2, $zero, 400

add $t3, $t0, $t1
sub $s0, $t2, $t3

li $v0, 1
move $a0, $s0
syscall
```

* Kết quả bài 2

```
-104600
```

1.3 Bài 3: Các lệnh về số học, phép nhân

Viết chương trình tính giá trị biểu thức f(x) bên dưới. Kết quả lưu vào thanh ghi $s0 và xuất ra màn hình.

$$f(x) = a.x^3 - b.x^2 + c.x - d$$

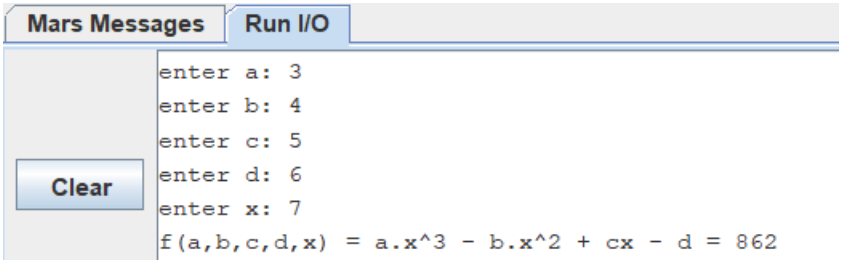
- Tương tự các câu trên, ta lưu giá trị a, b, c, d, x lần lượt vào các thanh ghi $t0$, $t1$, $t2$, $t3$ và sử dụng các phép toán phù hợp để giải quyết bài toán.

```
mul $s0, $t0, $t4 # a.x
sub $s0, $s0, $t1 # a.x - b
mul $s0, $s0, $t4 # (ax-b)x
add $s0, $s0, $t2 # (ax-b)x + c
mul $s0, $s0, $t4 # ((ax-b)x + c)x
sub $s0, $s0, $t3 # ((ax-b)x + c)x - d
```

- Tiếp theo ta đã có kết quả của f(x) được lưu vào thanh ghi $s0$, ta sử dụng thanh ghi $v0$ và load giá trị 1 để in kết quả ra console.

```
li $v0, 1
move $a0, $s0
syscall
```

* Kết quả bài 3



1.4 Bài 4: Lệnh load/store

a. Cho dãy số nguyên 10 phần tử, xuất ra kết quả là HIỆU của phần tử thứ 2 và 5. Mảng bắt đầu từ phần tử thứ 0.
Vì thanh ghi chỉ lưu được tối đa 32 bit mà số tối đa của 10 phần tử đã vượt ngoài tầm, nên bài này ta sẽ đọc dữ liệu theo string.
- Đầu tiên ta sử dụng thanh ghi $v0$ và load giá trị 8 để đọc dữ liệu từ người dùng và gán vào thanh ghi $t0$.

```
li $v0, 8
la $a1, 11
syscall
move $t0, $a0
```

- Sau đó ta sử dụng lệnh lb (load byte) để load địa chỉ của thanh ghi từ index 2 và 5 và lần lượt lưu giá trị vào thanh ghi $t2$ và $t5$.

```
lb $t2, 2($t0)
lb $t5, 5($t0)
```

- Khi có được 2 giá trị index 2 và 5 theo mã ASCII, ta sử dụng lệnh sub để giải quyết theo yêu cầu bài toán và lưu kết quả cuối cùng vào thanh ghi $t3$.

```
sub $t3, $t2, $t5
```

- Xuất kết quả ra console.

```
li $v0, 1
move $a0, $t3
syscall
```

∗ Kết quả bài 4a

```
Nhap so co 10 phan tu: 9573585886
Hieu phan tu 2 va 5 la: -1
```

b. Chuyển đổi vị trí cuối và đầu của chuỗi "MSSV Ho-Ten". Ví dụ chuỗi "123456 - Nguyen Van A" sẽ chuyển thành "A23456 - Nguyen Van 1". Sinh viên thay tên và mã số sinh viên của mình vào chuỗi trên.
- Ý tưởng bài này sẽ là dùng lệnh lb lấy giá trị phần tử cuối và ghi vào một array mới, sau đó ghi lần lượt từ index 1 đến index thứ size(string) - 1 rồi ghi phần tử đầu vào cuối của array.

```
.data
    input: .asciiz "2014725 - Nguyen Phuc Tien"
    output: .space 256
```

- Ta tạo một array là output với độ rộng là 256 bits.

```
.text
.globl main
main:
```

- Đầu tiên ta lưu input string vào thanh ghi $a0$.

```
la $a0, input
```

- Ta tạo một jal strlen (Jump and Link) để tính toán độ dài của chuỗi (input).

Ý tưởng sẽ là ta tạo một hàm lặp và đếm cho đến khi gặp byte null tức kí tự 0x0a. Biến index sẽ là giá trị của thanh ghi $t0$. Sau khi tính được chiều dài của string thì ta sẽ trừ đi 1 bởi vì index bắt đầu từ 0.

```
strlen:
        li $t0, 0
        li $t2, 0
        strlen_loop:
                add $t2, $a0, $t0
                lb $t1, 0($t2)
                beqz $t1, strlen_exit
                addiu $t0, $t0, 1
                j strlen_loop


        strlen_exit:
                subi $t0, $t0, 1
                add $v0, $zero, $t0
                add $t0, $zero, $zero
                jr $ra
```

- Ta gán giá trị thanh ghi $v0$ là $t0$ và trả giá trị thanh ghi $t0$ về 0.

∗ Như ý tưởng ta đã nêu ở trên, để giải quyết bài này sau khi tìm được chiều dài của string thì dưới đây sẽ là phần code để mô tả ý tưởng đó.
- Chiều dài string ta lưu ở thanh ghi $t1$, string lưu ở thanh ghi $t2$. Sau đó ta lưu giá trị 1 vào thanh ghi $t5$, đây cũng là giá trị index của mình khi thực hiện lưu các giá trị của string vào output.

```
add $t1, $zero, $v0
add $t2, $zero, $a0
li $t5, 1
```

- Sử dụng lb để lưu giá trị đầu tiên và cuối cùng của string. Kí tự đầu sẽ được lưu vào index 0 của output. Thay thế string bằng thanh ghi $t3$ và $t6$. Với thanh ghi $t6$ ta dùng để load kí tự cuối nên sẽ cộng thêm giá trị thanh ghi $t1$ tức size(string) - 1. Sử dụng sb (store byte) để ghi giá trị vào thanh ghi mới.

```
add $t3, $t2, $t0 # t0 = 0
add $t6, $t2, $t1


lb $t7, 0($t6) # lấy kí tự cuối cùng của input
sb $t7, output($zero) # lưu kí tự cuối cùng vào index 0 của output
add $t6, $t2, $zero # trả $t6 về input ban đầu
lb $s0, 0($t6) # lấy kí tụ đầu tiên của input và lưu vào thanh ghi s0
```

- Sử dụng vòng lặp để ghi các kí tự từ index 1 đến index cuối cùng của input vào output

```
myLoop:
        bgt $t5, $a0, myLoop_exit # nếu index t5 bằng a0 tức độ dài của string - 1 thì exit
        add $t6, $t2, $t5 # dịch trái 1
        lb $t7, 0($t6) # lấy giá trị đầu tiên và lưu vào thanh ghi t7
        sb $t7, output($t5) # lưu giá trị đầu tiên vào index t5
        addi $t5, $t5, 1
        j myLoop
myLoop_exit:
```

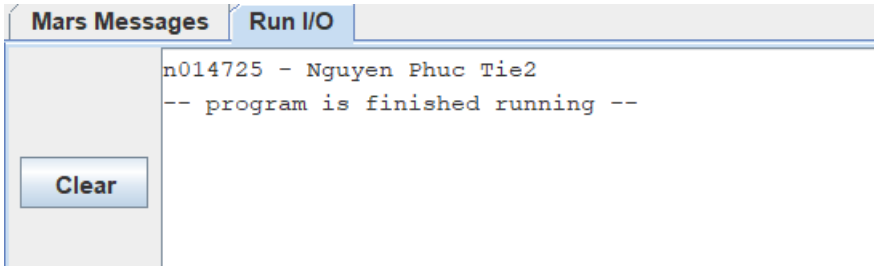
- Cuối cùng ta còn kí tự đầu tiên của input phải lưu vào kí tự cuối cùng của output.

```
sb $s0, output( $a0) # Lưu vào index a0 = strlen - 1
```

- Xuất kết quả ra console.

```
li $v0, 4
la $a0, output
syscall
```

* Kết quả bài 4b



2 Chapter 2.2 MIPS_ISA_Control

2.1 Bài 1: Phát biểu IF - ELSE (1)

```
1 if( a % 2 == 1) { Print string: "Computer Science and Engineering, HCMUT"}
2 else            { Print string: "Computer Architecture 2020"}
```

- Đầu tiên ta khai báo 2 string cho mệnh đề IF - ELSE trong .data

.data

ifMessage: .asciiz "Computer Science and Engineering, HCMUT"

elseMessage: .asciiz "Computer Architecture 2020"

- Tiếp theo ta sử dụng thanh ghi *v0* và load giá trị 5 để nhập dữ liệu int vào và lưu giá trị vào thanh ghi *t0*.

li $v0, 5

syscall

move $t0, $v0

- Ta sử dụng lệnh *rem* để lấy số dư của phép chia và sau đó giải quyết yêu cầu bài toán.

rem $t1, $t0, 2 # giá trị dư của t0 chia 2 được lưu vào thanh ghi t1

- Vì phép chia 2 chỉ có dư 0 và dư 1 nên ta sẽ so sánh giá trị thanh ghi t0 với 0, nếu bằng nhau thì xuất ra elseMessage, ngược lại sẽ xuất ra ifMessage.

beqz $t1, else # rẽ nhánh tới lable "else" nếu giá trị thanh ghi t1 = 0

#print ra ifMessage nếu rơi vào điều kiện của if tức giá trị thanh ghi t1 = 1

li $v0, 4

la $a0, ifMessage

syscall

j Endif # jump đến lable Endif để kết thúc mệnh đề If - Else

else: # print ra elseMessage

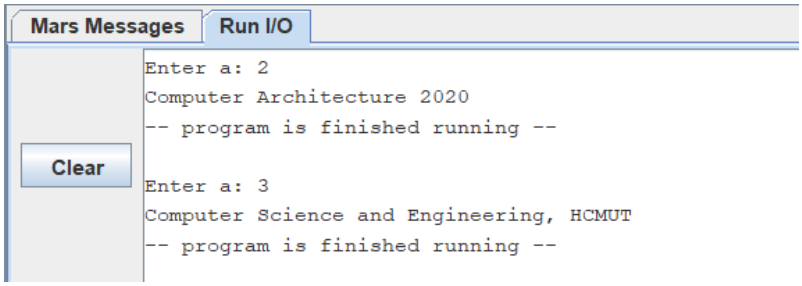
li $0̆, 4

la $a0, elseMessage

syscall

Endif:

* Kết quả bài 1



2.2 Bài 2: Phát biểu IF - ELSE (2)

```
1 if( a < -5 or a >= 3 ) { a = b * c;}
2 else                   { a = b + c;}
```

- Đầu tiên ta nhập các giá trị a, b, c và lưu vào các thanh ghi $t0$, $t1$ và $t2$.

slti $s0, $t0, -5 # Nếu t0 < -5 thì thanh ghi s0 sẽ được lưu giá trị là 1, ngược lại sẽ là giá trị 0
sge $s1, $t0, 3 # Nếu t0 >= 5 thì thanh ghi s1 sẽ được lưu giá trị là 1, ngược lại sẽ là giá trị
or $s0, $s0, $s1 # sử dụng or và lưu giá trị vào thanh ghi s0
beqz $s0, else # xét giá trị s0, nếu = 0 thì sẽ rẽ nhánh đến label "else"

- Tùy vào điều kiện If - Else ta sẽ sử dụng các lệnh tính toán phù hợp để giải quyết yêu cầu bài toán.

mul $t0, $t1, $t2 # a = b * c
# in ra message cho mệnh đề if
li $v0, 4
la $a0, ifMessage
syscall
j Endif

else:

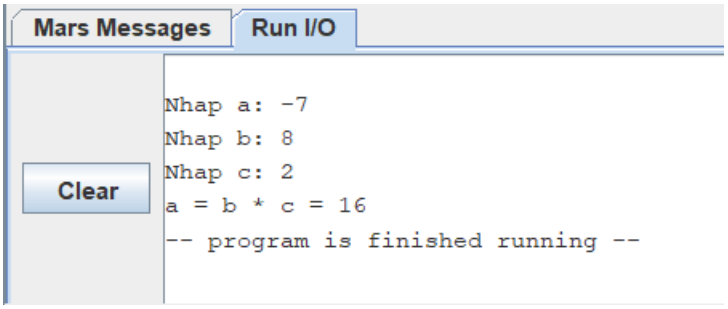
add $t0, $t1, $t2 # a = b + c
# in ra message cho mệnh đề else
li $v0, 4
la $a0, elseMessage
syscall

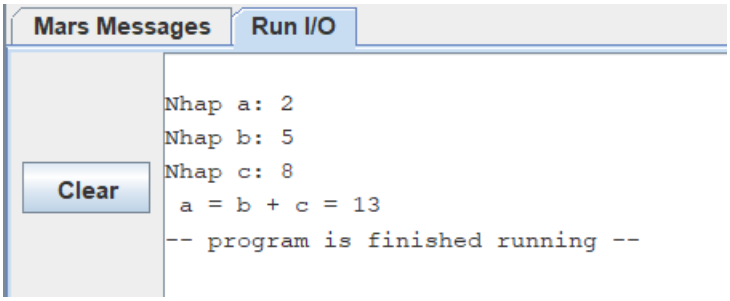
Endif:

- Xuất kết quả ra sau khi đã tính toán ở 2 mệnh đề If - Else. Kết quả cùng được lưu vào thanh ghi t0.

li $v0, 1
move $a0, $t0
syscall

∗ Kết quả bài 2





2.3 Bài 3: Phát biểu SWITCH - CASE

* Cho biết b = 200, c = 4

```
switch (input)
{
    case 1: a = b + c; break;
    case 2: a = b - c; break;
    case 3: a = b x c; break;
    case 4: a = b / c; break;
    default: NOP; // No-Operation; a = 0
       break;
}
```

Như 2 bài IF - ELSE trên thì SWITCH - CASE cũng chỉ là nhiều mệnh đề IF lồng với nhau.
- Đầu tiên ta cần người dùng nhập vào giá trị input và ta sẽ lưu vào thanh ghi s0.

```
li $t1, 200 # lưu giá trị b là 200 vào thanh ghi t1
li $t2, 4 # lưu giá trị c là 4 vào thanh ghi t2
```

- Tiếp theo ta sẽ tạo 4 nhánh tương ứng với 4 case là 1, 2, 3 và 4.

```
beq $s0, 1, A1_CON # s0 = 1 sẽ rẽ nhánh đến label A1_CON
beq $s0, 2, A2_CON
beq $s0, 3, A3_CON
beq $s0, 4, A4_CON
```

- Nếu cả 4 trường hợp này không thoải thì sẽ rơi vào case default.

```
j default_CON # jump đến label default
```

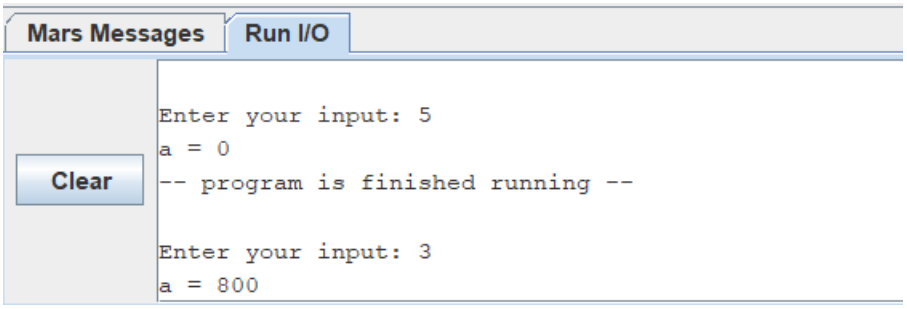
- Sau khi đã thiết lập đầy đủ các điều kiện của SWITCH - CASE thì ta sẽ thiết lập nên các label của nó.

```
A1_CON:
    add $t0, $t1, $t2 # a = b + c
    j done # jump đến label done để kết thúc SWITCH - CASE (break)
A2_CON:
    sub $t0, $t1, $t2 # a = b - c
    j done
A3_CON:
    mul $t0, $t1, $t2 # a = b * c
    j done
A4_CON:
    div $t0, $t1, $t2 # a = b / c
    j done
default_CON:
    li $t0, 0 # case default
done: # Kết thúc SWITCH - CASE
```

- Cuối cùng thì ta xuất kết quả ra console bằng cách sử dụng thanh ghi $v0$ và load cho nó giá trị 1 để in ra số nguyên.

```
li $v0, 1
move $a0, $t0
syscall
```

* Kết quả bài 3



2.4 Bài 4: Vòng lặp FOR xác định số Fibonacci thứ n

```
if      (n == 0) {return 0;}
else if(n == 1) {return 1;}
else{
    f0= 0; f1 = 1;
    for ( i = 2; i <= n; i++){
        fn = fn-1 + fn-2;
    }
}
return fn;
```

- Đầu tiên ta cần sử dụng thanh ghi $v0$ và load giá trị 5 để cho người dùng nhập input n vào và lưu giá trị vào thanh ghi t0.

```
li $v0, 5
syscall
move $t0, $v0
```

- Sau đó ta thiết lập cho 2 trường hợp n = 0, n = 1.

```
beq $t0, 0, Case_0 # t0 = 0 sẽ rẽ nhánh đến label case_0
beq $t0, 1, Case_1
```

- Thiết lập 2 giá trị ban đầu của F(0) = 0 và F(1) = 1 và index i bắt đầu từ 2.

```
li $t1, 0 # khoi tao f0 = 0
li $t2, 1 # khoi tao f1 = 1
li $t3, 0 # f(n)
li $t5, 2 # khoi tao index
```

- Tạo một vòng lặp FOR để tính số n của dãy Fibonacci. Sau khi kết thúc một vòng lặp thì ta sẽ dời các số Fibonacci khởi tạo của mình sang phải để tính toán cho vòng lặp kế tiếp.

```
for:
        bgt $t5, $t0, exit_loop # nếu index t5 > n thì exit
        add $t3, $t2, $t1 # fn = f(n-1) + f(n-2)
        addi $t5, $t5, $t1 # increase index by 1
        add $t1, $t2, $zero # f(n-2) = f(n-1)
        add $t3, $t2, $zero # f(n-1) = f(n)
        j for
exit_loop:
```

- Sau khi kết thúc vòng lặp ta sẽ xuất ra ngoài console và kết thúc chương trình.

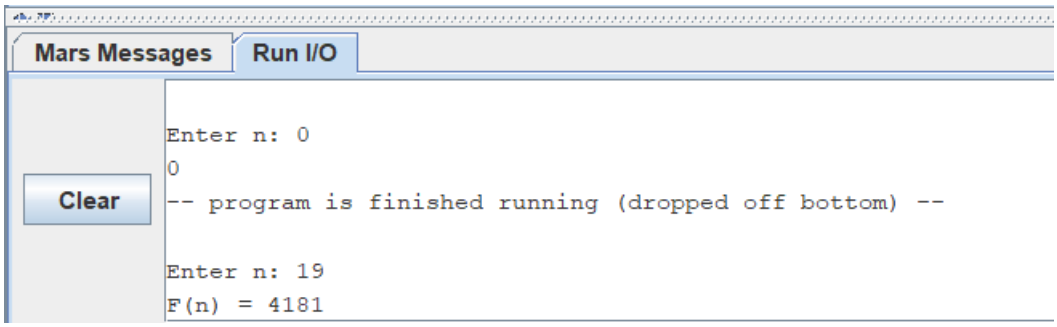
```
li $v0, 1
move $a0, $t3
syscall


li $v0, 10 # exit()
```

```
        syscall
```
- Nếu như ban đầu n đã rơi vào 2 trường hợp n = 0 và n = 1 thì sẽ rẽ nhánh xuống 2 label sau.

Case_0:
```
        # n = 0 thì in ra 0
        li $v0, 1
        move $a0, $zero
        syscall
        j exit_case
```
Case_1:
```
        # n = 1 thì in ra 1
        li $v0, 1
        li $t3, 1
        move $a0, $t3
        syscall
        j exit_case
```
exit_case:

* Kết quả bài 4



2.5 Bài 5: Vòng lặp WHILE

Xác định vị trí chữ 'e' đầu tiên trong chuỗi "Computer Architecture CSE-HCMUT".

```
i = 0;
while( charArray[i] != 'e' && charArray[i] != '\0'){
  i++;
}
```

- Đầu tiên trong phần .data ta sẽ khai báo chuỗi string đó vào myString.

myString: .asciiz "Computer Architecture CSE-HCMUT"

- Ta lưu myString vào thanh ghi s0 và tạo các biến index i trên thanh ghi t1, một flag để nhận biết tìm e chưa trên được lưu trên thanh ghi t5.
```
        la $a0, myString
        add $s0, $a0, $zero


        li $t1, 0 # index t1
        li $t5 0 # flag de biet tim duoc e chua
        add $t0, $s0, $zero # thanh ghi t0 = s0
```
- Một vòng lặp While để xác định được vị trí đầu tiên của 'e'. Trong mã ASCII thì e có giá trị là 101 nên khi ta tìm kiếm được giá trị 101 thì sẽ rẽ nhánh đến label tìm được e, gắn flag = 1 và xuất ra index của e. Ngược lại khi gặp kí tự null byte tức là 0x0a thì sẽ exit và in ra giá trị -1 ở console.

while:
```
        add $t3, $t1, $t0 # Tăng địa chỉ của string theo index t1
        lb $t4, 0($t3) # load byte tại index t1
```

```
        beqz $t4, exit # found null byte
        beq $t4, 101, found_e # found first 'e' in string
        addi $t1, $t1, 1 # tăng index 1 đơn vị
        j while
found_e:
        li $t5, 1 # flag = 1, found 'e' in string
exit:
```

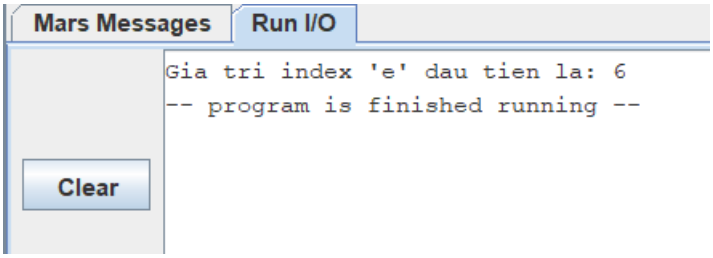
- Sau khi exit vòng lặp while thì ta sẽ check xem flag có bằng 1 không.

```
        beqz $t5, else # not found 'e'
```

- Tìm thấy 'e'(flag = 1), jump để kết thúc if - else và in kết quả ra console. Ngược lại nếu không tìm thấy 'e' thì lưu giá trị -1 vào thanh ghi chứa giá trị index.

```
        j Endif
else:
        li $t1, -1
Endif:
        li $v0, 1
        li $a0, $t1
        syscall
```

* Kết quả bài 5



* Trong tất cả các bài khi kết thúc chương trình ta sẽ sử dụng thanh ghi $v0$ và load giá trị 10 để exit(erminate execution)