

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO MÔN HỌC
LAB004 - IT012.P24.1

Nguyễn Quốc Khánh
MSSV: 24520793

Contents

1 Thực hành	3
1.1 Bài thực hành số 1	3
1.1.1 Đề bài	3
1.1.2 Đoạn code MIPS	3
1.1.3 Kiểm tra bằng MARS	4
1.2 Bài thực hành số 2	4
1.2.1 Đề bài	4
1.2.2 Đoạn code MIPS	4
1.2.3 Kiểm tra bằng MARS	5
2 Bài tập	6
2.1 Bài tập a	6
2.1.1 Đề bài 1	6
2.1.2 Đoạn code MIPS của bài 1	6
2.1.3 Output bài 1	8
2.1.4 Đề bài 2	8
2.1.5 Đoạn code MIPS của bài 2	8
2.1.6 Output bài 2	9
2.2 Bài tập b	9
2.2.1 Đề bài	9
2.2.2 Đoạn code MIPS	9
2.2.3 Output	10

LAB004 - IT012.P24.1

Nguyễn Quốc Khánh

MSSV: 24520793

Trường Đại học Công Nghệ Thông Tin

18/04/2025

ABSTRACT

Bài LAB004 này trình bày các lệnh if-else và while trong MIPS và cách mô phỏng chúng và các phép so sánh, phép nhảy và các phép toán cơ bản.

1 Thực hành

1.1 Bài thực hành số 1

1.1.1 Đề bài

Chuyển đoạn code sau sang MIPS và kiểm tra lại kết quả sử dụng MARS:

```
if (i==j)
    f = g + h;
else
    f = g - h;
```

(Với giá trị i, j, f, g, h lần lượt chứa trong thanh ghi \$s0, \$s1, \$s2, \$t0, \$t1)

1.1.2 Đoạn code MIPS

```
beq $s0, $s1, equal
sub $s2, $t0, $t1
j end_if

equal:
add $s2, $t0, $t1
```

Giải thích:

- Ta sử dụng “Branch on Equal”, kiểm tra giá trị trong 2 thanh ghi (\$s0, \$s1) và chuyển sang Branch được liệt kê (ở đây là Branch “equal”) nếu 2 giá trị ban đầu bằng nhau.
- Nếu như hai giá trị ở \$s0, \$s1 không bằng nhau thì chúng không nhảy tới Branch equal mà sẽ thực hiện lệnh sub ($f = g - h$) thay vì thực hiện lệnh add ở Branch equal ($f = g + h$). Giá trị của f sẽ được ghi ở \$s2

1.1.3 Kiểm tra bằng MARS

Ta sẽ gán giá trị $i = 1, j = 1, g = 4, h = 5$. Vì $i == j$ nên $f = 9$.

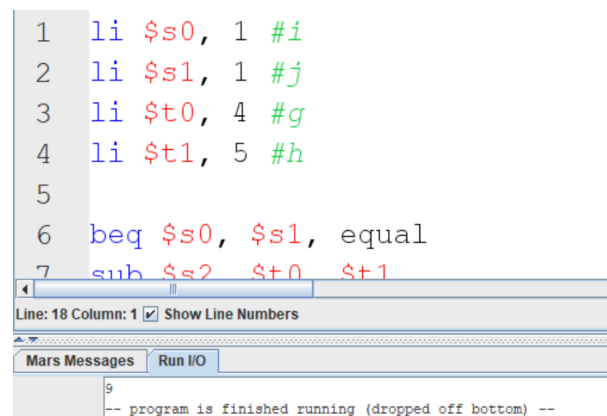
```
li $s0, 1 #i
li $s1, 1 #j
li $t0, 4 #g
li $t1, 5 #h

beq $s0, $s1, equal
sub $s2, $t0, $t1
j end_if

equal:
add $s2, $t0, $t1

end_if:
li $v0, 1
move $a0, $s2
syscall
```

Và để chắc chắn giá trị $f = 9$, ta sẽ xuất giá trị f ra màn hình. Kết quả như sau:



```
1 li $s0, 1 #i
2 li $s1, 1 #j
3 li $t0, 4 #g
4 li $t1, 5 #h
5
6 beq $s0, $s1, equal
7 sub $s2, $t0, $t1
```

Line: 18 Column: 1 ☒ Show Line Numbers

Mars Messages Run I/O

9
-- program is finished running (dropped off bottom) --

1.2 Bài thực hành số 2

1.2.1 Đề bài

Chuyển đoạn code sau sang MIPS và sử dụng MARS để kiểm tra lại kết quả

```
int Sum = 0
for (int i = 1; i<=N; ++i){
    Sum = Sum + 1;
}
```

(Với giá trị của i , N , Sum lần lượt chứa trong $\$s0$, $\$s1$, $\$s2$)

1.2.2 Đoạn code MIPS

Đoạn code MIPS:

```

5  start:
6  ble $s0, $s1, loop
7  j  end_loop
8
9  loop:
10 addi $s2, $s2, 1
11 addi $s0, $s0, 1
12 j  start

```

Giải thích code:

- Để thực hiện lệnh $i \leq N$, ta sử dụng “Branch Less Than or Equal” hoặc Mnemonic là ble để so sánh giữa i và N.
- Nếu chúng thoả điều kiện thì chuyển sang phần loop để thực hiện, nếu không thì nhảy tới end_loop để dừng.

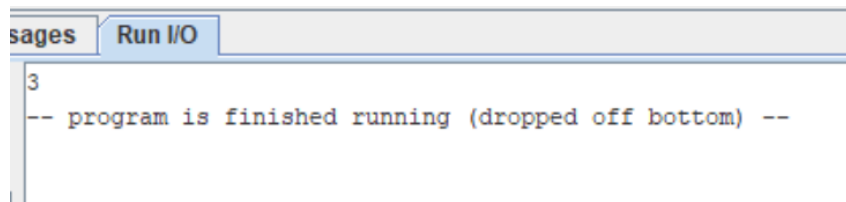
1.2.3 Kiểm tra bằng MARS

```

1  li $s0, 1 #i
2  li $s2, 0 #Sum
3  li $s1, 3 #N
4
5  start:
6  ble $s0, $s1, loop
7  j  end_loop
8
9  loop:
10 addi $s2, $s2, 1
11 addi $s0, $s0, 1
12 j  start
13
14 end_loop:
15 li    $v0, 1
16 move $a0, $s2
17 syscall

```

Ở đây, ta gán giá trị $N = 3$, tức là i sẽ chạy 3 vòng gồm $i = 1, 2, 3$. Và ở end_loop, ta sẽ print giá trị của Sum ra màn hình. Vì vòng lặp được chạy ba lần nên $\text{Sum} = 3$ và kết quả xuất ra màn hình sẽ là 3:



2 Bài tập

2.1 Bài tập a

2.1.1 Đề bài 1

Input: Nhập vào một ký tự

Output: Xuất ra cửa sổ I/O ký tự liền trước và liền sau của ký tự nhập vào

Ví dụ:

- Nhập ký tự: b
- Ký tự trước: a
- Ký tự sau: c

2.1.2 Đoạn code MIPS của bài 1

```
.data
nhap:      .asciiz "Nhap mot ky tu : "
before:    .asciiz "Ky tu truoc: "
after:     .asciiz "Ky tu sau: "
none_msg:  .asciiz "Khong co"
newline:   .asciiz "\n"

.text
.globl main

main:
    li $v0, 4
    la $a0, nhap
    syscall

    li $v0, 12
    syscall
    move $t0, $v0

    li $t1, 'a'
    blt $t0, $t1, check_uppercase

    li $t1, 'z'
    bgt $t0, $t1, check_uppercase

    li $t1, 'a'
    beq $t0, $t1, handle_a_case
```

```
    li $t1, 'z'
    beq $t0, $t1, handle_z_case

    addi $t1, $t0, -1
    addi $t2, $t0, 1
    j print_both

check_uppercase:
    li $t1, 'A'
    blt $t0, $t1, invalid_input

    li $t1, 'Z'
    bgt $t0, $t1, invalid_input

    li $t1, 'A'
    beq $t0, $t1, handle_A_case

    li $t1, 'Z'
    beq $t0, $t1, handle_Z_case

    addi $t1, $t0, -1
    addi $t2, $t0, 1
    j print_both
```

```

handle_a_case:
    addi $t2, $t0, 1
    j print_after_only

handle_z_case:
    addi $t1, $t0, -1
    j print_before_only

handle_A_case:
    addi $t2, $t0, 1
    j print_after_only

handle_Z_case:
    addi $t1, $t0, -1
    j print_before_only

print_both:
    li $v0, 4
    la $a0, before
    syscall
    li $v0, 11
    move $a0, $t1
    syscall
    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, after
    syscall
    li $v0, 11
    move $a0, $t2
    syscall
    li $v0, 4
    la $a0, newline
    syscall
    j exit

print_before_only:
    li $v0, 4
    la $a0, before
    syscall
    li $v0, 11
    move $a0, $t1
    syscall
    li $v0, 4
    la $a0, newline
    syscall

```

```

    li $v0, 4
    la $a0, after
    syscall
    li $v0, 4
    la $a0, none_msg
    syscall

    li $v0, 4
    la $a0, newline
    syscall
    j exit

print_after_only:
    li $v0, 4
    la $a0, before
    syscall
    li $v0, 4
    la $a0, none_msg
    syscall
    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, after
    syscall
    li $v0, 11
    move $a0, $t2
    syscall
    li $v0, 4
    la $a0, newline
    syscall
    j exit

invalid_input:
    j exit

exit:
    li $v0, 10
    syscall

```

2.1.3 Output bài 1

```
Nhap mot ky tu chu cai: ZKy tu truoc: Y
Ky tu sau: Khong co
```

```
Nhap mot ky tu chu cai: gKy tu truoc: f
Ky tu sau: h
```

2.1.4 Đề bài 2

Input: Ký tự nhập vào chỉ được phép là ba loại: số, chữ thường và chữ hoa.

Output: Xuất ra cửa sổ đó là loại nào, nếu như không rơi vào một trong ba loại trên thì xuất ra thông báo “invalid type”.

2.1.5 Đoạn code MIPS của bài 2

```
.text
.globl main
main:
    li $v0, 4
    la $a0, prompt
    syscall

    li $v0, 12
    syscall
    move $t0, $v0

    li $v0, 11
    li $a0, 10
    syscall

    li $t1, 97
    li $t2, 122

    blt $t0, $t1, check_uppercase
    bgt $t0, $t2, check_uppercase

    li $v0, 4
    la $a0, lowercase_msg
    syscall
    j exit

check_uppercase:
    li $t1, 65
    li $t2, 90
```

```
    blt $t0, $t1, check_number
    bgt $t0, $t2, check_number

    li $v0, 4
    la $a0, uppercase_msg
    syscall
    j exit

check_number:
    li $t1, 48
    li $t2, 57

    blt $t0, $t1, invalid_input
    bgt $t0, $t2, invalid_input

    li $v0, 4
    la $a0, number_msg
    syscall
    j exit

invalid_input:
    li $v0, 4
    la $a0, invalid_msg
    syscall

exit:
    li $v0, 10
    syscall
```


2.1.6 Output bài 2

```
Nhap ky tu
2
so
```

```
Nhap ky tu
a
chu thuong
```

```
Nhap ky tu
A
chu hoa
```

2.2 Bài tập b

2.2.1 Đề bài

Input: Nhập từ bàn phím 2 số nguyên

Output: Số lớn hơn, tổng hiệu, tích, thương của hai số.

2.2.2 Đoạn code MIPS

```
.data
prompt:      .ascii "Nhap 2 so nguyen a va b\n"
larger_msg:  .ascii "So lon hon: "
sum_msg:     .ascii "a+b = "
sub_msg:     .ascii "a-b = "
mul_msg:     .ascii "a x b = "
div_msg:     .ascii "a : b = "
div_zero:    .ascii "Cannot divide by zero"
newline:     .ascii "\n"

.text
.globl main
main:
    li $v0, 4
    la $a0, prompt
    syscall

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

    li $v0, 5
    syscall
    move $t1, $v0

    li $v0, 4
    la $a0, larger_msg
    syscall

    bge $t0, $t1, a_is_larger
    move $a0, $t1
    j print_larger

a_is_larger:
    move $a0, $t0
```

```
print_larger:
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, sum_msg
    syscall

    add $a0, $t0, $t1
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, sub_msg
    syscall

    sub $a0, $t0, $t1
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, newline
    syscall

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

```
    mul $a0, $t0, $t1
    li $v0, 1
    syscall

    li $v0, 4
    la $a0, newline
    syscall

    li $v0, 4
    la $a0, div_msg
    syscall

    beqz $t1,
div_by_zero

    div $t0, $t1
    mflo $a0
    li $v0, 1
    syscall
    j exit

div_by_zero:
    li $v0, 4
    la $a0, div_zero
    syscall

exit:
    li $v0, 10
    syscall
```

2.2.3 Output

```
Nhap 2 so nguyen a va b
10
5
So lon hon: 10
a+b = 15
a-b = 5
a x b = 50
a : b = 2
```