

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



BÁO CÁO MÔN HỌC
LAB003 - IT012.P24.1

Nguyễn Quốc Khánh
MSSV: 24520793

Contents

1 Thực hành	3
1.1 Mô phỏng các lệnh cơ bản và chức năng của chúng	3
1.2 Mô phỏng cái chương trình cơ bản	4
1.2.1 Ví dụ 1	4
1.2.2 Ví dụ 2	5
1.2.3 Ví dụ 3	5
1.2.4 Ví dụ 4	6
2 Bài tập nhập xuất chuỗi	7
2.1 Bài toán:	7
2.2 Bài giải	7
2.2.1 Câu a	7
2.2.2 Câu b	7
2.2.3 Câu c	8
2.2.4 Câu d	9

LAB003 - IT012.P24.1

Nguyễn Quốc Khánh

MSSV: 24520793

Trường Đại học Công Nghệ Thông Tin

03/04/2025

ABSTRACT

Bài LAB003 này trình bày các lệnh cơ bản trong MIPS và cách mô phỏng chúng, bao gồm các phép toán số học, logic, và các lệnh load/store. Ngoài ra, bài lab cũng hướng dẫn cách viết chương trình MIPS đơn giản để nhập xuất chuỗi và thực hiện các thao tác liên quan đến chuỗi, cũng như nhập xuất và tính toán với số nguyên.

1 Thực hành

1.1 Mô phỏng các lệnh cơ bản và chức năng của chúng

Lệnh	Mô phỏng	Chức năng
add	add \$1,\$s2,\$s3	$s1 = s2 + s3$
addi	addi \$s1, \$s2, 3	$s1 = s2 + 3$
addu	addu \$s1,\$s2,\$s3	$s1 = s2 + s3$ s1 sẽ được xem là số không dấu khi tổng bị tràn
addiu	addiu \$s1,\$s2,100	$s1 = s2 + 100$ s1 sẽ được xem là số không dấu khi tổng bị tràn
sub	sub \$s1,\$s2,\$s3	$s1 = s2 - s3$
subu	subu \$s1,\$s2,\$s3	$s1 = s2 - s3$ Nếu s2 nhỏ hơn s3, tức kết quả của phép trừ là số âm thì nó sẽ trả về số không âm lớn nhất.
and	and \$s1, \$s2, \$s3	$s1 = s2 \wedge s3$
andi	andi \$s1, \$s2, 12	$s1 = s2 \wedge 12$
or	or \$s1, \$s2, \$s3	$s1 = s2 \vee s3$
nor	nor \$s1, \$s2, \$s3	$s1 = s2 \text{ NOR } s3$
lw	lw \$t0, 12(\$sp)	$\$t0 = \text{memory}[\$sp + 12]$
sw	sw \$t0, 16(\$gp)	$\text{memory}[\$gp + 16] = \$t0$
slt	slt \$t0, \$s1, \$s2	if ($\$s1 < \$s2$) $\$t0 = 1$; else $\$t0 = 0$
slti	slti \$t0, \$s1, 5	if ($\$s1 < 5$) $\$t0 = 1$; else $\$t0 = 0$
sltu	sltu \$t0, \$s1, \$s2	if ($\$s1 < \$s2$) $\$t0 = 1$; else $\$t0 = 0$ (so sánh không dấu)

Lệnh	Mô phỏng	Chức năng
sltiu	sltiu \$t0, \$s1, 10	if (\$s1 < 10) \$t0 = 1; else \$t0 = 0 (so sánh không dấu)
syscall	syscall	sử dụng để gọi các hệ thống gọi (system call) của hệ điều hành.

1.2 Mô phỏng cái chương trình cơ bản

1.2.1 Ví dụ 1

```

                .data
var1:           .word 23

                .text
__start:
    lw $t0, var1
    li $t1, 5
    sw $t1, var1

```

Giải thích code:

- Ở bốn dòng đầu của đoạn code, chúng ta khai báo biến có tên **var1** và khởi tạo nó với giá trị **23**
- Sau đó, ta:
 1. Tải giá trị của **var1** (23) vào thanh ghi \$t0
 2. Tải giá trị tức thời là **5** vào thanh ghi \$t1
 3. Lưu giá trị của thanh ghi \$t1 (có giá trị là 5) vào bộ nhớ của **var1**.

Giá trị ở địa chỉ 0x10010000 thay đổi từ 23 (là 0x00000017 ở hệ HEX) sang 5:

Address	Value (+0)
0x10010000	0x00000017

Address	Value (+0)
0x10010000	0x00000005

1.2.2 Ví dụ 2

```
.data
array1: .space 12
.text
__start: la      $t0, array1
        li      $t1, 5
        sw $t1, ($t0)
        li $t1, 13
        sw $t1, 4($t0)
        li $t1, -7
        sw $t1, 8($t0)
```

Giải thích code:

- Ở phần đầu của đoạn code, ta cho **array1** có thể lưu được 12 bytes.
- `la $t0, array1` : Ta cho load address **array1** vào thanh ghi **t0**
- `li $t1, 5` : Load immediate giá trị 5 vào thanh ghi **t1** (\$t1 sẽ hiển thị 0x00000005)
- `sw $t1, ($t0)` : Store word từ **t1** (có giá trị 5) vào 4 bytes đầu tiên (0-3) của **t0** (đang chứa **array1**)
- `li $t1, 13` : Load immediate giá trị 13 vào thanh ghi **t1** (\$t1 sẽ hiển thị 0x0000000d)
- `sw $t1, 4($t0)` : Store word từ **t1** (có giá trị 13) vào 4 bytes tiếp theo (4-7) của **t0** (đang chứa **array1**)
- `li $t1, -7` : Load immediate giá trị -7 vào thanh ghi **t1** (\$t1 sẽ hiển thị 0xffffffff9 là dạng bù 2 của -7)
- `sw $t1, 8($t0)` : Store word từ **t1** (có giá trị -7) vào 4 bytes cuối cùng (8-11) của **t0** (đang chứa **array1**)

Kết quả:

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x00000005	0x0000000d	0xffffffff9

1.2.3 Ví dụ 3

```
li $v0, 5

syscall
```

Giải thích code:

Ta Load Immediate 5 vào \$v0 và khi **v0** nhận giá trị là 5, theo bảng service của syscall, giá trị 5 trong **v0** sẽ nhận giá trị do người dùng nhập vào ở tab Run I/O là lưu giá trị do người dùng nhập vào và lưu nó vào thanh ghi **v0**.

Ví dụ:

Khi chạy code, ta nhập vào giá trị -7 vào tab Run I/O

```
Pages Run I/O
-7
-- program is finished running (dropped off bottom) --
```

Giá trị ở **v0** sẽ là **0xffffffff9** tức là -7 ở dạng bù 2.

\$v0	2	0xffffffff9
------	---	-------------

1.2.4 Ví dụ 4

```
                .data
string1:        .ascii "Print this. \n"

                .text

main:           li      $v0, 4

                la      $a0, string1
                syscall
```

Giải thích code:

```
                .data
string1:        .ascii "Print this. \n"
                .text
```

- Ở phần code trên, ta đặt tên string1 cho một vị trí nhớ và lưu ASCII vào.

```
li $v0, 4
la $a0, string1
syscall
```

- Ở phần code này, ta:

- Load Immediate giá trị 4 vào \$v0, và nó sẽ chỉ dẫn cho syscall để in chuỗi ra.
- Load Address của chuỗi "Print this. \n" vào thanh ghi \$a0
- Thực hiện lệnh syscall và in chuỗi ra Run I/O

```
Pages Run I/O
Print this.
-- program is finished running (dropped off bottom) --
```

2 Bài tập nhập xuất chuỗi

2.1 Bài toán:

Nhập vào một chuỗi, xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau:

a) Khai báo và xuất ra cửa sổ I/O 2 chuỗi có giá trị như sau:

Chuỗi 1: Chao ban! Ban la sinh vien nam thu may?

Chuỗi 2: Hihi, minh la sinh vien nam thu 1 ^-^

b) Biểu diễn nhị phân của 2 chuỗi trên dưới bộ nhớ là gì?

c) Xuất ra lại đúng chuỗi đã nhập.

d) Nhập vào 2 số nguyên sau đó xuất tổng của 2 số nguyên này

2.2 Bài giải

2.2.1 Câu a

```
.data
chuoil: .asciiz "Chao ban! Ban la sinh vien nam thu may? \n"
chuoil2: .asciiz "Hihi, minh la sinh vien nam thu 1 ^-^ \n"
.text

main:   li      $v0, 4

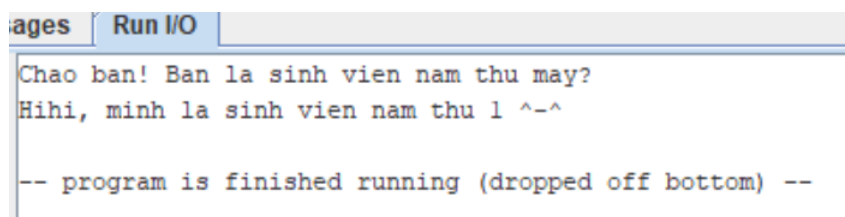
        la      $a0, chuoil
        syscall

        li      $v0, 4

        la      $a0, chuoil2
        syscall
```

Tương tự như ví dụ 4, ta khai báo hai chuỗi và lưu dữ liệu ASCII vào chúng như đề bài, Load Immediate giá trị 4 vào \$v0 để syscall in chuỗi ra Run I/O.

Kết quả



```
ages  Run I/O
Chao ban! Ban la sinh vien nam thu may?
Hihi, minh la sinh vien nam thu 1 ^-^
-- program is finished running (dropped off bottom) --
```

2.2.2 Câu b

Để thực hiện bài tập này, trước tiên ta cần nhấn nút “Dump machine code or data in a available format”, sau đó chọn Binary Text. Sau đây là kết quả:

```

00100100000000100000000000000100
00111100000000010001000000000001
00110100001001000000000000000000
000000000000000000000000000001100
001001000000000100000000000000100
00111100000000010001000000000001
001101000010010000000000000101010
000000000000000000000000000001100

```

Hoặc dưới đây là dạng HEX và ASCII lưu theo thứ tự little-edian:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x6f616843	0x6e616220	0x61422021	0x616c206e	0x6e697320	0x69762068	0x6e206e65	0x74206d61
0x10010020	0x6d207568	0x203f7961	0x6948000a	0x202c6968	0x686e696d	0x20616c20	0x686e6973	0x65697620
0x10010040	0x616e206e	0x6874206d	0x20312075	0x205e2d5e	0x0000000a	0x00000000	0x00000000	0x00000000

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	o a h C	n a b	a B !	a l n	n i s	i v h	n n e	t m a
0x10010020	m u h	? y a	i H \0 \n	, i h	h n i m	a l	h n i s	e i v
0x10010040	a n n	h t m	l u	^ - ^	\0 \0 \0 \n	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

2.2.3 Câu c

```

.data
string: .asciiz
.text

main:   li $v0, 8
        la $a0, string
        li $a1, 20
        syscall

        li $v0, 4
        la $a0, string
        syscall

```

Giải thích code:

1. Ta Load Immediate giá trị 8 vào **v0** để thực hiện đọc Chuỗi
2. Ta Load Immediate giá trị 20 vào **a1** thì người dùng có nhập vào số lượng bytes tối đa mà syscall có thể đọc được từ input, ở đây thì người dùng có thể nhập vào tối đa 19 chữ số vì byte cuối là null terminator.
3. Sau đó ta Load Immediate giá trị 4 vào **v0** để syscall xuất chuỗi vừa nhập.

2.2.4 Câu d

```
li $v0, 5
syscall
move $t0, $v0
li $v0, 5
syscall
move $t1, $v0
add $t2, $t0, $t1
li $v0, 1
move $a0, $t2
syscall
```

Giải thích code:

1. Ta Load Immediate giá trị 5 vào **v0** để syscall đọc integer thứ nhất
2. Ta move giá trị 5 ở **v0** vào **t0** để nhớ giá trị
3. Ta Load Immediate giá trị 5 vào **v0** để syscall đọc integer thứ hai và lưu ở **t1**
4. Thực hiện lệnh $t2 = t0 + t1$, tức là cộng hai số vừa nhập vào và cho giá trị đó vào **t2**
5. Load Immediate giá trị 1 để syscall xuất integer ở **t2**.