

MỤC LỤC

1.	<i>Mô phỏng thực thi các lệnh và chức năng của các lệnh trong MIPS.....</i>	2
2.	<i>Mô phỏng chương trình và giải thích ý nghĩa:</i>	3
3.	<i>Thực hiện bài tập</i>	4
a.	Khai báo và xuất ra chuỗi I/O 2 chuỗi:	4
b.	Biểu diễn nhị phân của hai chuỗi trên dưới bộ nhớ:	5
c.	Xuất chuỗi đã nhập:	5
d.	Nhập vào hai số nguyên và tính tổng:	6

BÁO CÁO THỰC HÀNH

Tổ chức cấu trúc máy tính – IT012

Sinh viên: Nguyễn Công Hậu – MSSV: 23520453

Giảng viên hướng dẫn thực hành: Nguyễn Thành Nhân

1. Mô phỏng thực thi các lệnh và chức năng của các lệnh trong MIPS

STT	Tên	Chức năng	Mô phỏng
1	add	$R[rd] = R[rs] + R[rt]$ Thực hiện cộng giá trị thanh ghi rs với giá trị thanh ghi rt, tổng đưa vào thanh ghi rd	add \$s1, \$s2, \$s3
2	addi	$R[rt] = R[rs] + \text{SignExtImm}$ Thực hiện cộng giá trị thanh ghi rs với số tức thời, kết quả đưa vào thanh ghi rt.	addi \$s1, \$s2, 10
3	addu	<i>Addu</i> có cú pháp và thực hiện chức năng giống <i>add</i>	addu \$s1, \$s2, \$s3
4	addiu	<i>Addiu</i> có cú pháp và thực hiện chức năng giống <i>addi</i>	addiu \$s1, \$s2, 10
5	sub	Lệnh <i>sub</i> có cú pháp tương tự như lệnh <i>add</i> , nhưng: <ul style="list-style-type: none">• <i>add</i> thực hiện phép toán cộng 2 thanh ghi, kết quả lưu vào thanh ghi thứ 3• <i>sub</i> thực hiện phép toán trừ 2 thanh ghi, kết quả lưu vào thanh ghi thứ 3	sub \$s1, \$s2, \$s3
6	subu	Lệnh <i>subu</i> có cú pháp và chức năng giống như <i>sub</i> , nhưng: <i>subu</i> không xét đến kết quả có bị overflow hay không <i>sub</i> có xét đến kết quả có bị overflow hay không; nếu bị overflow, sẽ có thông báo	subu \$s1, \$s2, 10
7	and	$R[rd] = R[rs] \& R[rt]$ Thực hiện <i>and</i> từng bit giá trị của thanh ghi rs và rt với nhau, kết quả lưu vào thanh ghi rd	and \$s1, \$s2, \$s3
8	andi	Ý nghĩa: $R[rt] = R[rs] \& \text{ZeroExtImm}$ Lệnh này thực hiện <i>and</i> từng bit giá trị thanh ghi rs và một số tức thời. Số tức thời đang là số 16 bits, mở rộng thành số 32 bits theo kiểu ZeroExtImm, tức 16 bits nữa cao còn thiếu sẽ điền 0 vào. Sau đó thực hiện <i>and</i> từng bit giá trị của thanh ghi rs và số tức thời đã được mở rộng thành 32 bits với nhau, kết quả lưu vào thanh ghi rd	andi \$s1, \$s2, 10
9	or	<i>or</i> và <i>nor</i> cách viết tương tự như <i>and</i> , nhưng thay vì thực hiện phép toán <i>and</i> , 2 lệnh này sẽ thực hiện	or \$s1, \$s2, \$s3

10	nor	phép toán <i>or</i> hoặc <i>nor</i> cho từng bit trong 2 thanh ghi, kết quả lưu vào thanh ghi thứ 3	nor \$s1, \$s2, \$s3
11	lw	Ý nghĩa: $R[rt] = M[R[rs] + \text{SignExtImm}]$ □ Lấy giá trị trong thanh ghi rs cộng với số tức thời đang lưu trong offset (số tức thời này này được mở rộng có dấu thành 32 bits) ta được địa chỉ của từ nhớ cần lấy dữ liệu. Dữ liệu của từ nhớ này sẽ được lấy để lưu vào thanh ghi rt Lưu ý: $M[X]$: là lấy giá trị của từ nhớ tại địa chỉ X	lw \$s1, 0(\$s2)
12	sw	Ý nghĩa: $M[R[rs] + \text{SignExtImm}] = R[rt]$ □ Lưu giá trị thanh ghi rt vào từ nhớ có địa chỉ được tính bằng giá trị thanh ghi rs cộng với offset (offset được mở rộng có dấu thành số 32 bits trước khi cộng)	sw \$s1, 0(\$s2)
13	slt	slt: $R[rd] = (R[rs] < R[rt]) ? 1 : 0$ Kiểm tra xem giá trị trong thanh ghi rs có nhỏ hơn thanh ghi rt hay không, nếu nhỏ hơn thì thanh ghi rd nhận giá trị là 1; ngược lại thanh ghi rd sẽ nhận giá trị 0	slt \$s1, \$s2, \$s3
14	sltu	sltu: Ý nghĩa thực hiện giống như slt. Nhưng việc kiểm tra giá trị thanh ghi rs có nhỏ hơn thanh ghi rt hay không trong lệnh slt thực hiện trên số có dấu, còn trong sltu thực hiện trên số không dấu	sltu \$s1, \$s2, \$s3
15	slti	slti/sltiu: $R[rd] = (R[rs] < \text{SignExtImm}) ? 1 : 0$ Ý nghĩa 2 lệnh này giống nhau là so sánh giá trị một thanh ghi với một số tức thời, nếu giá trị trong thanh ghi rs	slti \$s1, \$s2, 10
16	sltiu	nhỏ hơn số tức thời thì thanh ghi rd nhận giá trị là 1; ngược lại thanh ghi rd sẽ nhận giá trị 0 Số tức thời cho phép trong lệnh này là số 16 bits. Trước khi so sánh với thanh ghi rs, số tức thời được mở rộng có dấu (SignExtImm) thành số 32 bits <i>slti</i> khác <i>sltiu</i> là <i>slti</i> so sánh 2 giá trị theo kiểu có dấu dạng bù 2, trong khi đó <i>sltiu</i> so sánh theo kiểu số không dấu	sltiu \$s1, \$s2, 10
17	syscall	Làm treo sự thực thi của chương trình và chuyển quyền điều khiển cho hệ điều hành (được gửi giả lập bởi MARS). Sau đó, hệ điều hành sẽ xem giá trị thanh ghi \$v0 để xác định xem chương trình muốn nó làm việc gì	syscall

2. Mô phỏng chương trình và giải thích ý nghĩa:

Ảnh chụp màn hình	Code	Ý nghĩa từng câu lệnh
<pre> 1 .data 2 var1: .word 23 3 4 .text 5 start: 6 lw \$t0, var1 7 li \$t1, 5 8 sw \$t1, var1 9 </pre> <p>Hình 1</p>	<pre> .data var1: .word 23 .text start: lw \$t0, var1 li \$t1, 5 sw \$t1, var1 </pre>	<p>Khai báo vùng nhớ data</p> <p>Khai báo biến kiểu word: var1 = 23</p> <p>Khai báo vùng nhớ text</p> <p>\$t0 lưu giá trị var 1</p> <p>\$t1 = 5</p> <p>Var1 lưu giá trị \$t1</p>
<pre> 1 .data 2 array1: .space 12 3 4 .text 5 start: 6 la \$t0, array1 7 li \$t1, 5 8 sw \$t1, (\$t0) 9 li \$t1, 13 10 sw \$t1, 4(\$t0) 11 li \$t1, -7 12 sw \$t1, 8(\$t0) 13 </pre> <p>Hình 2</p>	<pre> .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) </pre>	<p>Khai báo vùng nhớ data cấp 12-byte bộ nhớ, chưa được khởi tạo</p> <p>Khai báo vùng nhớ text</p> <p>\$t0 = địa chỉ array1</p> <p>\$t1 = 5</p> <p>array1[0] = \$t1</p> <p>\$t1 = 13</p> <p>array[1] = \$t1</p> <p>\$t1 = -7</p> <p>array1[2] = \$t1</p>
<pre> 1 li \$v0, 5 2 syscall </pre> <p>Hình 3</p>	<pre> li \$v0, 5 syscall </pre>	<p>Truyền tham số 5 vào thanh ghi \$v0</p> <p>Thực hiện chức năng (đọc số nguyên)</p>
<pre> 1 .data 2 string1: .ascii "Print this. \n" 3 4 .text 5 main: 6 li \$v0, 4 7 la \$a0, string1 8 syscall 9 </pre> <p>Hình 4</p>	<pre> .data string1: .ascii "Print this. \n" .text main: li \$v0, 4 la \$a0, string1 syscall </pre>	<p>Khai báo vùng nhớ data</p> <p>Khai báo mảng string1 = “Print this.\n”</p> <p>Khai báo vùng nhớ text</p> <p>Truyền tham số 4 vào \$v0 (thực hiện chức năng in chuỗi kí tự mà địa chỉ được lưu trong \$a0)</p> <p>\$a0 = địa chỉ string1</p> <p>Thực hiện chức năng</p>

3. Thực hiện bài tập

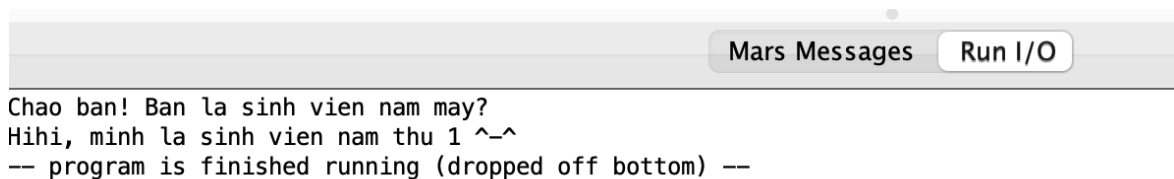
a. Khai báo và xuất ra chuỗi I/O 2 chuỗi:

```

1      .data
2  string1: .ascii "Chao ban! Ban la sinh vien nam may?"
3  string2: .ascii "\nHihi, minh la sinh vien nam thu 1 ^-^"
4
5      .text
6  main:
7      li $v0, 4
8      la $a0, string1
9      syscall
10
11     li $v0, 4
12     la $a0, string2
13     syscall

```

Hình 5: Code khai báo và xuất ra cửa sổ I/O hai chuỗi



Chao ban! Ban la sinh vien nam may?
 Hihi, minh la sinh vien nam thu 1 ^-^
 — program is finished running (dropped off bottom) —

Hình 6: Kết quả từ cửa sổ I/O

b. Biểu diễn nhị phân của hai chuỗi trên dưới bộ nhớ:

```

001001000000001000000000000000100
001111000000000100010000000000001
001101000010010000000000000000000
0000000000000000000000000000001100
001001000000001000000000000000100
001111000000000100010000000000001
001101000010010000000000000100100
0000000000000000000000000000001100

```

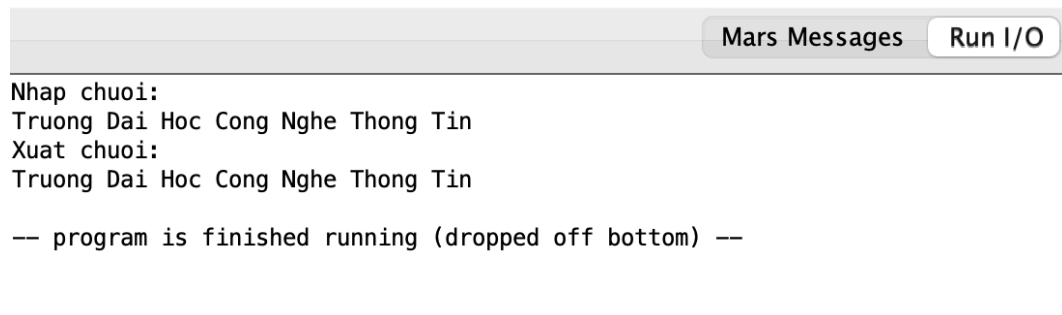
c. Xuất chuỗi đã nhập:

```

1      .data
2  msg_input: .asciiz "Nhap chuoi: \n"
3  msg_output: .asciiz "Xuat chuoi: \n"
4  buffer: .space 64
5
6      .text
7  main:
8  #thong bao nhap chuoi
9  li $v0, 4
10 la $a0, msg_input
11 syscall
12
13 #nhap chuoi
14 li $v0, 8
15 la $a0, buffer
16 li $a1, 64
17 syscall
18
19 #thong bao xuat chuoi
20 li $v0, 4
21 la $a0, msg_output
22 syscall
23
24 #xuat chuoi
25 li $v0, 4
26 la $a0, buffer
27 syscall

```

Hình 6: Nhập chuỗi và xuất chuỗi đã nhập



Hình 7: Kết quả từ cửa sổ I/O

d. Nhập vào hai số nguyên và tính tổng:

```

1      .data
2      msg_input1: .asciiz "Nhap a: \n"
3      msg_input2: .asciiz "Nhap b:\n"
4      msg_output: .asciiz "Tong hai so nguyen la: \n"
5      buffer: .space 64
6      .text
7      main:
8      #thong bao nhap so nguyen a
9      li $v0, 4
10     la $a0, msg_input1
11     syscall
12
13     #nhap so nguyen a
14     li $v0, 5
15     syscall
16     move $t0, $v0
17
18     #thong bao nhap so nguyen b
19     li $v0, 4
20     la $a0, msg_input2
21     syscall
22
23     #nhap so nguyen 2
24     li $v0, 5
25     syscall
26     move $t1, $v0
27
28     #tinh tong so nguyen
29     move $t2, $v0
30     addu $t0, $t0, $t1
31
32     #thong bao xuat tong
33     li $v0, 4
34     la $a0, msg_output
35     syscall
36
37     #in ket qua tong 2 so nguyen
38     move $a0, $t0
39     li $v0, 1
40     syscall

```

Hình 8: Code nhập và tính tổng hai số nguyên



Hình 9: Kết quả chương trình từ cửa sổ I/O