

Laboratory Exercise 5

Character string with SYSCALL function, and sorting

Trần Khánh Quỳnh-20225762

1. Assignment 1:

```
1 #Lab 5, Assignment 1, Tran Khanh Quynh
2 .data
3 test: .asciiz "Hello World"
4 .text
5     li $v0, 4
6     la $a0, test
7     syscall
```

Kết quả chạy chương trình:

The screenshot shows the Mars MIPS simulator interface. The 'Text Segment' window displays the assembly code: `li $v0, 4` and `la $a0, test` followed by `syscall`. The 'Registers' window shows the state of the registers, with `$v0` highlighted at 4 and `$a0` highlighted at 0x10010000. The 'Data Segment' window shows the memory layout, and the 'Mars Messages' window shows the output 'Hello World'.

-Nhận xét:

+Thanh ghi \$v0 được nạp vào giá trị 4, tiếp theo thanh ghi \$a0 chuyển thành 0x10010000. Cuối cùng thanh ghi \$a0 chuyển thành 0x10010000

+Thanh ghi \$v0 nạp vào giá trị là 4, còn \$a0 nạp địa chỉ của chuỗi ký tự (0x10010000 là địa chỉ của chuỗi ký tự biến test). Khi nạp 4 vào \$v0 thì ghi gọi lệnh syscall → thực hiện in ra màn hình giá trị trong địa chỉ được lưu ở \$a0

Lưu trữ trong địa chỉ 0x10010000:

Chuỗi “Hello World” được tách thành cụm 4 ký tự, rồi được lưu vào theo thứ tự đảo ngược lại với 4 byte địa chỉ

→ Tách thành “lleH” “oW o” và “\0ldr”

“lleH”: chữ l có mã ascii là 108, khi chuyển sang hệ hexa là 0x6c

Chữ e có mã ascii là 101, chuyển sang hệ hexa là 0x65

Chữ H có mã ascii là 72, chuyển sang hệ hexa là 0x48

Lưu “lleH” sẽ là 0x6c6c6548

“oW o”: Chữ o có mã ascii là 111 → hexa là 0x6F

Chữ W có mã ascii là 87 → chuyển sang hexa là 0x57

Dấu cách → 0x20

Khi lưu “roW o” sẽ là 0x6F57206F

“\0ldr” : \0 khi chuyển sang hệ hexa là 0x00

Chữ l có mã ascii 108 → 0x6c

Chữ d có mã ascii 100 → 0x64

Chữ r có mã ascii 114 → 0x72

Khi lưu “\0ldr” sẽ là 0x00646c72

2. Assignment 2

-Mã nguồn

```
Lab5*
1  .data:
2  Message: .ascii "The sum of "
3  Message1: .ascii " and "
4  Message2: .ascii " is "
5  .text
6      li $s0, 1      #value of $s0
7      li $s1, 2      #Value of $s1
8      add $s2, $s0, $s1 #Sum
9
10     li $v0, 4        #Print Message
11     la $a0, Message
12     syscall
13     li $v0, 1        #Print value of $s0
14     la $a0, 0($s0)
15     syscall
16     li $v0, 4        #Print Message1
17     la $a0, Message1
18     syscall
19     li $v0, 1        #Print value of $s1
20     la $a0, 0($s1)
21     syscall
22     li $v0, 4        #Print Message2
23     la $a0, Message2
24     syscall
25     li $v0, 1        #Print sum
26     la $a0, 0($s2)
27     syscall
```

-Kết quả chạy chương trình

The screenshot displays the Mars MIPS simulator interface. The main window shows assembly code with columns for Address, Code, Basic, and Source. The code includes instructions like `addiu $2,$0,0x00000000`, `ori $1,$0,0x00000000`, `add $4,$1,$1`, `syscall`, `addiu $2,$0,0x00000000`, `li $v0,4`, `la $a0,message2`, `ori $1,$0,0x00000000`, `addiu $2,$0,0x00000000`, `li $v0,1`, `ori $1,$0,0x00000000`, `add $4,$1,$1`, and `syscall`. The Data Segment window shows memory addresses from `0x10010000` to `0x10010120` with values. The Registers window shows the state of registers, with `$a0` highlighted. The Messages window shows the output: "The sum of 1 and 2 is 3" and "-- program is finished running (dropped off bottom) --".

Lần lượt lưu giá trị 1 vào \$s0, 2 vào \$s1 và sau đó tính tổng rồi lưu vào \$s2.

In lần lượt đầu tiên là đoạn “The sum of”, sau đó in ra giá trị của \$s0; đoạn “and” với giá trị của \$s1. Tiếp theo đó in “is” và giá trị của \$s2 in cuối cùng.

3. Assignment 3

```
#Assignment 3, Lab 05, Tran Khanh Quynh

.data
x: .space 100
y: .asciiz "HelloWorld"

.text

la $a0, x
la $a1, y

strcpy:
    add $s0, $zero, $zero    #s0 = i = 0
L1:
    add $t1, $s0, $a1        #t1 = s0 + a1 = i + y[0]
    lb $t2, 0($t1)           #t2 = value at t1 = y[i]
    add $t3, $s0, $a0        #t3 = i + x[0]
    sb $t2, 0($t3)           #x[i] = t2 = y[i]
    beq $t2, $zero, end_of_strcpy #if y[i]==0, exit
    nop
    addi $s0, $s0, 1         #s0=s0 + 1 <-> i=i+1
    j L1                    #next character
    nop
end_of_strcpy:
    li $v0, 4
    syscall
```

Kết quả chạy

The screenshot displays the Mars MIPS simulator interface. The top section shows the assembly code with columns for Op, Address, Code, Basic, and Source. The registers section on the right lists registers \$zero through \$lo with their current values. The data segment section shows memory addresses and their corresponding values. The messages section at the bottom shows the output of the program, including "Hello World" and "program is finished running".

Op	Address	Code	Basic	Source
lui	0x3e011001	lui \$1, 0x00001001	7:	la \$a0, x
ori	0x34240000	ori \$4, \$1, 0x00000000		
lui	0x3e011001	lui \$1, 0x00001001	8:	la \$a1, y
ori	0x34250004	ori \$5, \$1, 0x00000004		
add	0x00008020	add \$16, \$0, \$0	11:	add \$s0, \$zero, \$zero #s0 = i = 0
add	0x02054820	add \$9, \$16, \$5	13:	add \$t1, \$s0, \$a1 #t1 = s0 + a1 = i + y[0]
lb	0x812a0000	lb \$t2, 0(\$t1)	14:	lb \$t2, 0(\$t1) #t2 = value at t1 = y[i]
add	0x02045820	add \$11, \$16, \$4	15:	add \$t3, \$s0, \$a0 #t3 = i + x[0]
sb	0xa16a0000	sb \$t2, 0(\$t3)	16:	sb \$t2, 0(\$t3) #x[i] = t2 = y[i]
beq	0x11400004	beq \$t2, \$zero, end_of_strcpy	17:	beq \$t2, \$zero, end_of_strcpy #if y[i] == 0, exit
nop	0x00000000	nop	18:	nop
addi	0x22100001	addi \$16, \$16, 0x0000...19:		addi \$s0, \$s0, 1 #s0 = s0 + 1 <-> i = i + 1

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000004
\$v1	3	0x00000000
\$a0	4	0x10010000
\$a1	5	0x10010064
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x1001006f
\$t2	10	0x00000000
\$t3	11	0x10010000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x0000000b
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400040
\$hi		0x00000000
\$lo		0x00000000

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x6c6c6548	0x6f57206f	0x00464c72	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x6c6c6548	0x6f57206f	0x00464c72	0x00000000	0x00000000	0x00000000	0x00000000
0x10010080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Mars Messages: Run I/O

```
hello world
-- program is finished running (dropped off bottom) --
Hello World
-- program is finished running (dropped off bottom) --
```

Giải thích:

+Đầu tiên sẽ lưu địa chỉ của biến x vào \$a0; lưu địa chỉ của biến y vào \$a1. Do ta quy định x có không gian là 100 (64 trong hệ hexa), vì vậy nên giá trị lưu trong \$a0 (tức địa chỉ của x) các giá trị của \$a1 (tức địa chỉ của y) 64 đơn vị(0x10010000 và 0x10010064)

+Thực hiện lệnh add để gán i = 0 vào thanh ghi \$s0

+Đi vào lần đầu của vòng lặp L1:

Câu lệnh add để cộng i với địa chỉ của i, tức i sẽ đi lần lượt từng kí tự một trong chuỗi. Với lần đầu tiên thì i = 0

Thanh ghi \$t1 sẽ lưu địa chỉ của y[i]. Thanh ghi \$t1 lúc này sẽ lưu địa chỉ của y[0], nên \$t1 có giá trị là 0x10010064

Câu lệnh lb dùng để lấy giá trị từ địa chỉ lưu trong thanh ghi \$t1 lưu vào thanh \$t2. Ta có y[0] = 'H'. H có mã ascii là 72, khi chuyển sang hexa sẽ là 48. Vì vậy nên thanh \$t2 sẽ chuyển thành giá trị là 0x00000048

Thanh ghi \$t3 sẽ lưu địa chỉ của x[i]. Với i = 0 lúc này nên thanh ghi \$t3 lưu địa chỉ của x[0], vì thế nên \$t3 có giá trị là 0x10010000.

Lệnh sb \$t2, 0(\$t3) lưu giá trị của \$t2 vào địa chỉ mà \$t3 trỏ đến. Tức trong thanh nhớ có địa chỉ 0x10010000 thì đã lưu lại kí tự 'H'; hay đã lưu được kí tự đầu tiên của chuỗi y vào chuỗi x.

Câu lệnh beq để kiểm tra nếu \$t2 có giá trị bằng 0, tức gặp kí tự '\0' thì sẽ kết thúc chuỗi nên dừng copy.

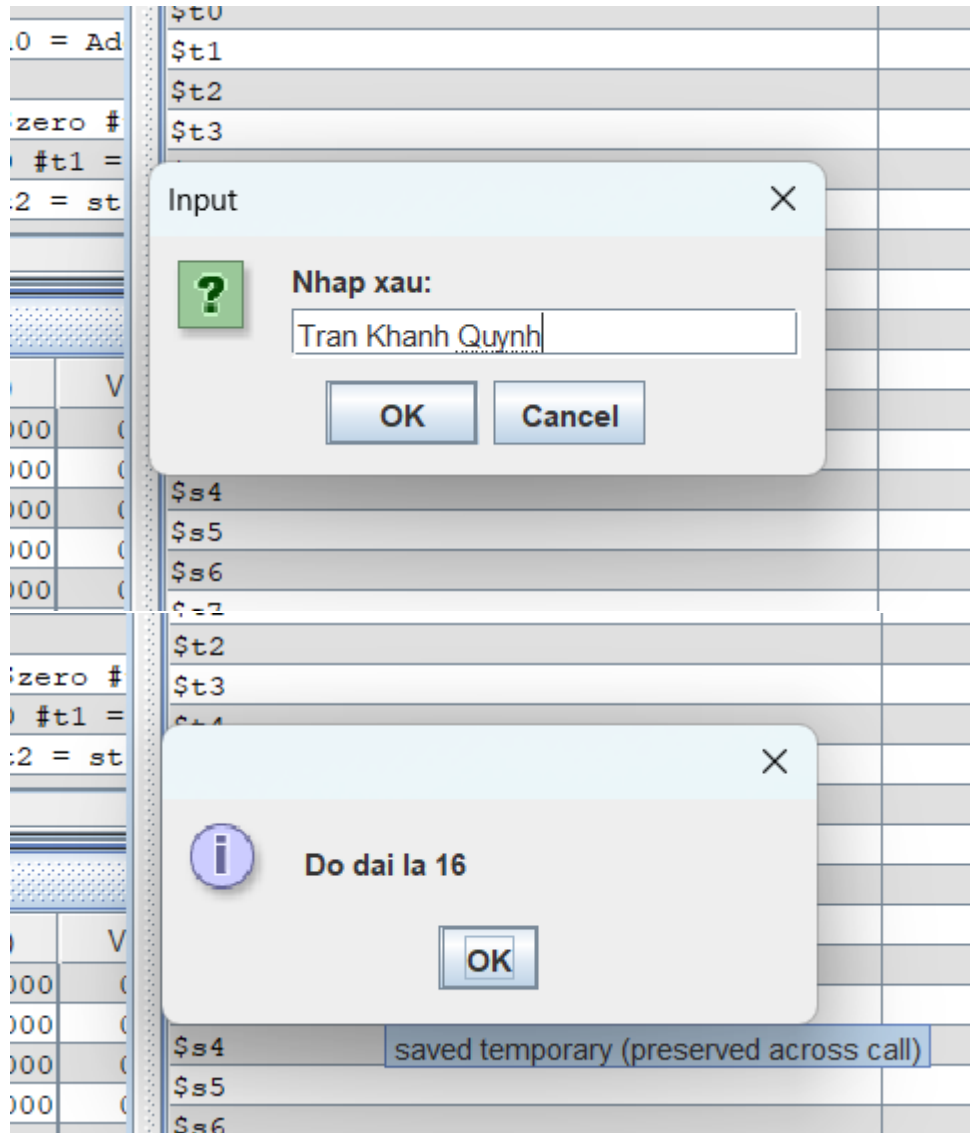
+Tiếp tục thực hiện vòng lặp cho đến khi gặp kí tự kết thúc, ta lần lượt lưu được nội dung của chuỗi y vào chuỗi x.

4. Assignment 4

-Mã nguồn:

```
1  #Assignment 4, Lab 05, Tran Khanh Quynh
2
3  .data
4  string: .space 50
5  Message1: .asciiz "Nhap xau:"
6  Message2: .asciiz "Do dai la "
7
8  .text
9  main:
10 get_string:
11     li $v0, 54
12     la $a0, Message1
13     la $a1, string
14     la $a2, 50
15     syscall
16 get_length:
17     la $a0, string #a0 = Address(string[0])
18     xor $t0, $zero, $zero #t0 = i = 0
19 check_char:
20     add $t1, $a0, $t0 #t1 = a0 + t0 = Address(string[0] + i)
21     lb $t2, 0($t1) #t2 = string[i]
22     beq $t2, $zero, end_of_str
23     addi $t0, $t0, 1 #i = i+1
24     j check_char
25 end_of_str:
26 end_of_get_length:
27 print_length:
28     li $v0, 56
29     la $a0, Message2
30     sub $a1, $t0, 1
31     syscall
```

-Kết quả chạy:



-Giải thích:

+get_string: Đặt \$v0 bằng 54 để có thể InputDialogString. Sau đó đưa các tham số: \$a0 lưu địa chỉ của chuỗi Message1, \$a1 lưu địa chỉ của chuỗi nhập vào, \$a2 lưu lại giới hạn số kí tự sẽ đọc

+print_length: Nạp 56 vào \$v0 để gọi đến dịch vụ MatDialogInt. Sau đó đưa các tham số: \$a0 lưu địa chỉ chuỗi Message2, \$a1 lưu giá trị phép tính trừ giá trị thành \$t0 với 1 (không tính kí tự kết thúc chuỗi '\0'; \$t0 lưu độ dài của chuỗi)

5. Assignment 5

```

1  #Assignment 5, Lab 05, Tran Khanh Quynh
2  .data
3  string: .space 21
4  newline: .asciiz "\n"
5  .text
6      li $t0, 0      # i = 0
7      la $s0, string # load the address of string into $s0
8  read_char:
9      li $v0, 12     # read character
10     syscall
11
12  check_char:
13      beq $v0, 10, reverse # If the input char is enter then reverse
14      add $t1, $s0, $t0    # t1 = address(string[i])
15      sb $v0, 0($t1)      # store input char into string[i]
16      addi $t0, $t0, 1     # i = i+1
17      slti $t2, $t0, 20   # t2 = (i < 20) ? 1 : 0
18      beq $t2, $0, reverse # if i = 20 --> reverse
19      j read_char
20
21      addi $t0, $t0, -1    # move to the last character
22
23      li $v0, 4
24      la $a0, newline
25      syscall
26  reverse:
27      add $t1, $s0, $t0
28      lb $t2, 0($t1)      # load char at string[i]
29      li $v0, 11          # print character
30      add $a0, $t2, $0
31      syscall
32      beq $t0, $0, exit   # if i = 0, exit
33      addi $t0, $t0, -1   # decrement i
34      j reverse
35  exit:

```

Kết quả chạy:

TH1: gõ enter

The screenshot shows the MARS MIPS simulator interface. The main window displays the assembly code from the previous block. The 'Text Segment' window is active, showing the code with addresses and sources. The 'Data Segment' window shows memory addresses from 0x10010000 to 0x10010120. The 'Registers' window on the right shows the state of registers, with \$v0 highlighted. The 'MARS Messages' window at the bottom shows the output 'abc' and a message 'program is finished running (dropped off bottom) --'.

Lần lượt chạy chương trình, mỗi lần gọi syscall nhập vào các kí tự a, b, c và sau đó enter; chương trình sẽ in ra lần lượt c, b, a rồi kết thúc

TH2: 20 kí tự

Đầu vào:

The screenshot shows the MARS MIPS simulator interface. The Text Segment window displays assembly code for a program that reads a string into register \$t1. The Data Segment window shows memory addresses from 0x10010000 to 0x10010120, with the value 0x74737271 (ASCII '1234567890') at address 0x10010000. The Registers window shows the state of registers \$zero to \$t0, \$a0 to \$a3, and \$t1 to \$t7. The Run I/O window shows the input string 'abcdefghijklmnopqrst'.

Kết quả chạy:

The screenshot shows the MARS MIPS simulator interface after the program has executed. The Text Segment window displays assembly code for a program that prints the string in reverse. The Data Segment window shows memory addresses from 0x10010000 to 0x10010120, with the value 0x74737271 (ASCII '1234567890') at address 0x10010000. The Registers window shows the state of registers \$zero to \$t0, \$a0 to \$a3, and \$t1 to \$t7. The Run I/O window shows the output string 'abcdefghijklmnopqrst' and the message 'program is finished running (dropped off bottom)'.

Lần lượt chạy chương trình, mỗi lần gọi syscall nhập vào 20 kí tự; đến khi đủ 20 kí tự thì chương trình sẽ in ra lần lượt in ra các kí tự theo thứ tự ngược lại khi nhập vào.