

# Báo cáo Thực hành KTMT Tuần 5

Họ và tên: Nguyễn Đức Phú

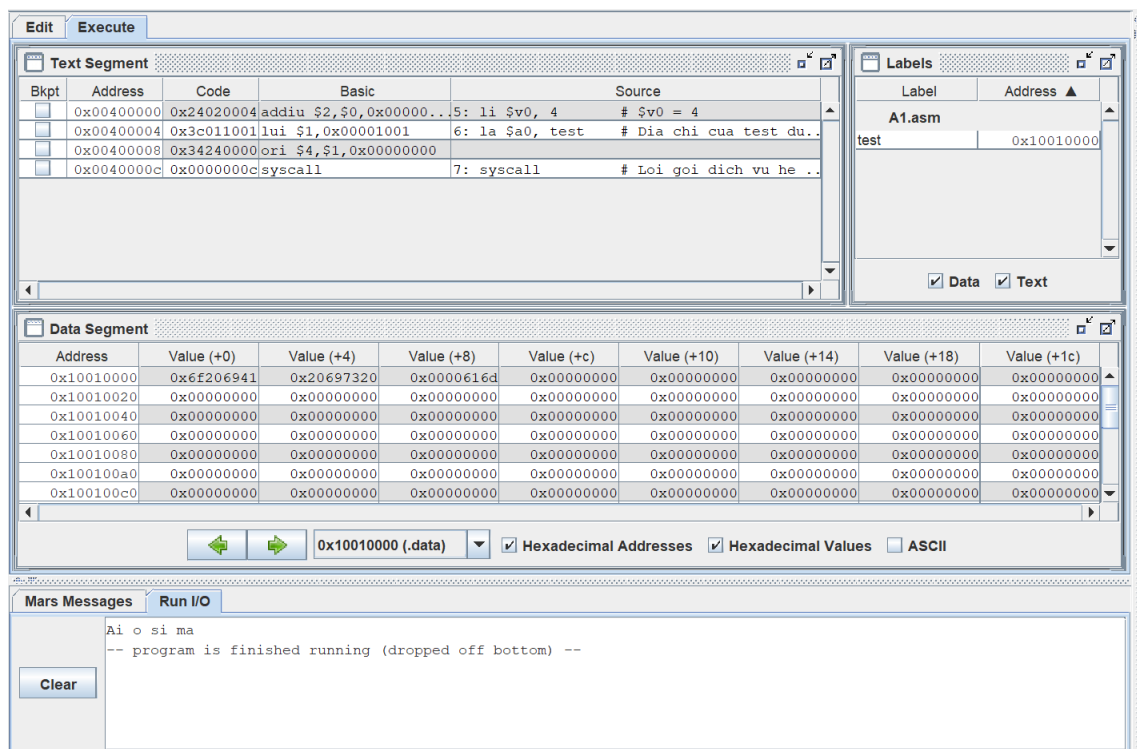
MSSV: 20215116

## Assignment 1:

- Code:

```
#Laboratory Exercise 5, Assignment 1
.data
    test: .asciiz "Ai o si ma"
.text
    li $v0, 4                # $v0 = 4
    la $a0, test              # Địa chỉ của test được ghi vào $a0
    syscall                  # Lời gọi dịch vụ hệ thống
```

- Kết quả chạy thử:



## Assignment 2:

- Code:

```
.data
    str1: .asciiz "The sum of "
    str2: .asciiz " and "
    str3: .asciiz " is "
```

```

.text
    li $s0, 6           # number1 = 6
    li $s1, 9           # number2 = 9
    add $t0, $s0, $s1   # $t0 = Sum of 6 and 9
    li $v0, 4           # Print string "str1"
    la $a0, str1
    syscall
    li $v0, 1           # Print $s0
    move $a0, $s0
    syscall
    li $v0, 4           # Print string "str2"
    la $a0, str2
    syscall
    li $v0, 1           # Print $s1
    move $a0, $s1
    syscall
    li $v0, 4           # Print string "str3"
    la $a0, str3
    syscall
    li $v0, 1           # Print $t0
    move $a0, $t0
    syscall
Exit: li $v0, 10
      syscall

```

- Kết quả chạy thử:

The screenshot shows the Mars MIPS simulator interface. The **Text Segment** window displays the assembly code with addresses and comments. The **Data Segment** window shows memory addresses and their corresponding values in hexadecimal. The **Run I/O** window shows the output of the program.

**Text Segment:**

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24110006	addiu \$16,\$0,0x0000...	6: li \$s0, 6 # number1 = 6
	0x00400004	0x24110009	addiu \$17,\$0,0x0000...	7: li \$s1, 9 # number2 = 9
	0x00400008	0x02114020	add \$8,\$16,\$17	8: add \$t0, \$s0, \$s1 # \$t0 = Sum ..
	0x0040000c	0x24020004	addiu \$2,\$0,0x0000...	9: li \$v0, 4 # Print string "str1"
	0x00400010	0x3c011001	lui \$1,0x00001001	10: la \$a0, str1
	0x00400014	0x34240000	ori \$4,\$1,0x00000000	
	0x00400018	0x0000000c	syscall	11: syscall
	0x0040001c	0x24020001	addiu \$2,\$0,0x0000...	12: li \$v0, 1 # Print \$s0
	0x00400020	0x00102021	addu \$4,\$0,\$16	13: move \$a0, \$s0

**Data Segment:**

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x20656854	0x206d7573	0x0020666f	0x646e6120	0x69200020	0x00002073	0x00000000	0x00000000
0x10010020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010060	0x00000000	0x00000000	0x00000000	0x00000000	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

**Run I/O:**

```

Reset: reset completed.
The sum of 6 and 9 is 15
-- program is finished running --

```

## Assignment 3:

- Code:

```
.data
x: .space 32          # destination string x, empty
y: .asciiz "Random string" # source string y

.text
strcpy:
    add $s0, $zero, $zero    # $s0 = i = 0
    la $a1, y                # Load address of y to $a1 - src
    la $a0, x                # Load address of x to $a0 - des
L1: add $t1, $s0, $a1        # $t1 = $s0 + $a1 = i + y[0]
                               # = address of y[i]
    lb $t2, 0($t1)           # $t2 = value at $t1 = y[i]
    add $t3, $s0, $a0        # $t3 = $s0 + $a0 = i + x[0]
                               # = address of x[i]
    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:
```

- Kết quả chạy thử:

Text Segment					Labels				
Bkpt	Address	Code	Basic	Source	Label	Address	Name	Number	Value
	0x00400000	0x00008020	add \$16,\$0,\$0	6: add \$s0, \$zero...			\$zero	0	0x00000000
	0x00400004	0x3c011001	lui \$1,0x00001...	7: la \$a1, y # Load...			\$a1	1	0x10010000
	0x00400008	0x34250020	ori \$5,\$1,0x00...		strcpy	0x00400000	\$v0	2	0x00000000
	0x0040000c	0x3c011001	lui \$1,0x00001...	8: la \$a0, x # Load...	L1	0x00400014	\$v1	3	0x00000000
	0x00400010	0x34240000	ori \$4,\$1,0x00...		end_of_strcpy	0x00400038	\$a0	4	0x10010000
	0x00400014	0x02054820	add \$9,\$16,\$5	9: L1: add \$t1,\$s0,\$a1 ...	x	0x10010000	\$a1	5	0x10010020
	0x00400018	0x812a0000	lb \$10,0x00000...	11: lb \$t2,0(\$t1) ...	y	0x10010020	\$a2	6	0x00000000
	0x0040001c	0x02045820	add \$11,\$16,\$4	12: add \$t3,\$s0,\$a0 ...			\$a3	7	0x00000000
	0x00400020	0xa16a0000	sb \$10,0x00000...	14: sb \$t2,0(\$t3) ...			\$t0	8	0x00000000
							\$t1	9	0x1001002d
							\$t2	10	0x00000000
							\$t3	11	0x1001000d
							\$t4	12	0x00000000
							\$t5	13	0x00000000
							\$t6	14	0x00000000
							\$t7	15	0x00000000
							\$s0	16	0x0000000d
							\$s1	17	0x00000000
							\$s2	18	0x00000000
							\$s3	19	0x00000000
							\$s4	20	0x00000000
							\$s5	21	0x00000000
							\$s6	22	0x00000000
							\$s7	23	0x00000000
							\$t8	24	0x00000000
							\$t9	25	0x00000000

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+16)
0x10010000	d n a R	s m o	n i r t	\0 \0 \0 g	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	d n a R	s m o	n i r t	\0 \0 \0 g	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

⇒ Nội dung xâu từ \$a1 có địa chỉ 0x10010020 đã được copy sang \$a0 có địa chỉ 0x10010000 (hiển thị trên cửa sổ Data Segment)

## Assignment 4:

- Code:

```
#Laboratory Exercise 5, Assignment 4
.data
    string: .space 50
    Message1: .asciiz "Nhap xau: "
    Message2: .asciiz "Do dai xau la: "
.text
main:
get_string:
    li $v0, 54          # Get a string from dialog
    la $a0, Message1    # Load address of the Message1 to $a0
    la $a1, string       # Load address of input buffer "string" to $a1
    la $a2, 50           # Maximum number of characters to read
    syscall

get_length:
    la $a0, string       # $a0 = address(string[0])
    add $t0, $zero, $zero # $t0 = i = 0

check_char:
    add $t1, $a0, $t0     # $t1 = $a0 + $t0
                        # = address(string[i])
    lb $t2, 0($t1)        # $t2 = string[i]
    beq $t2, $zero, end_of_str # is null char?
    addi $t0, $t0, 1      # $t0 = $t0 + 1 -> i = i + 1
    j check_char

end_of_str:
end_of_get_length:
print_length:
    addi $t0, $t0, -1     # Do dai xau = $t0-(null_char)
    li $v0, 56            # Show the length to message dialog
    la $a0, Message2      # Load address of the Message1 to $a0
    move $a1, $t0          # Set $a1 to contents of $t0
    syscall
```

- ⇒ Ta thấy ở vòng lặp cuối với lý tự kết thúc xâu thì giá trị `$t0` vẫn được tăng thêm 1
- ⇒ Khi xử lý in độ dài xâu cần thêm lệnh `addi $t0, $t0, -1` để in được đúng giá trị chính xác

- Kết quả chạy thử:

The screenshot shows a debugger interface with the following components:

- Text Segment:** A table of assembly instructions.
 

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020036	addiu \$2,\$0,0x...	9: li \$v0, 54 ...
	0x00400004	0x3c011001	lui \$1,0x00001...	10: la \$a0, Message1 ...
	0x00400008	0x34240032	ori \$4,\$1,0x00...	
	0x0040000c	0x3c011001	lui \$1,0x00001...	11: la \$a1, string # Load...
	0x00400010	0x34250000	ori \$5,\$1,0x00...	
	0x00400014	0x24060032	addiu \$6,\$0,0x...	12: la \$a2, 50 # Maxi...
	0x00400018	0x0000000c	syscall	13: syscall
	0x0040001c	0x3c011001	lui \$1,0x00001...	15: la \$a0, stri...
	0x00400020	0x34240000	ori \$4,\$1,0x00...	
- Data Segment:** A table of memory addresses and values.
 

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	i a d	u a x	: a l	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
- Labels:** A table of labels and addresses.
 

Label	Address
A4.asm	
main	0x00400000
get_string	0x00400000
get_length	0x0040001c
check_char	0x00400028
end_of_str	0x0040003c
end_of_get_l...	0x0040003c
- Registers:** A table of registers and their values.
 

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$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

The screenshot shows the same debugger interface as above, but with a different dialog box:

- Dialog Box:** A message box with the text "Do dai xau la: 10" and an "OK" button.

⇒ Kết quả chạy thử trả về đúng: Độ dài 10 đối với xâu “Xau nao do”

## Assignment 5:

- Code:

```
#Laboratory Exercise 5, Assignment 5
.data
    get_char: .space 20
    message1: .ascii "Nhap ky tu thu "
    message2: .ascii ": "
    message3: .ascii "\n"
    message4: .ascii "Chuoai ky tu vua nhap la: "

.text
    li $s0, 20                # N = 20
    li $s1, 0                 # i = 0
    la $s2, get_char          # Load address of get_char[0]
    li $s3, 10                # Char \n in ASCII

read_char:
    beq $s1, $s0, end_read_char # i = N branch to exit
# Show message "Nhap ky tu thu i: "
    li $v0, 4
    la $a0, message1
    syscall
    addi $t1, $s1, 1
    li $v0, 1
    move $a0, $t1
    syscall
    li $v0, 4
    la $a0, message2
    syscall

    li $v0, 12                # Read character
    syscall
    move $t0, $v0
    beq $t0, $s3, end_read_char # Press "Enter" branch to exit
    li $v0, 4
    la $a0, message3
    syscall
    add $s5, $s2, $s1          # $s5=Address of get_char[i]=get_char[0]+i
    sb $t0, 0($s5)             # Store character to get_char[i]
    addi $s1, $s1, 1           # i++
    j read_char

end_read_char:
    li $v0, 4                  # Show message4
    la $a0, message4
    syscall

print_string:
    li $v0, 11                 # Show ky tu tai dia chi trong $s5
    lb $a0, 0($s5)
```

```

syscall
beq $s5, $s2, exit      # $s5 = address của ký tự cuối cùng
addi $s5, $s5, -1       # Tiến dần đến ký tự đầu tiên
j print_string

exit:
li $v0, 10
syscall

```

- Kết quả chạy thử:

The screenshot shows the Mars MIPS simulator interface. The top window, titled "Data Segment", displays a memory table with columns for Address, Value (+0), Value (+4), Value (+8), Value (+c), Value (+10), Value (+14), Value (+18), and Value (+1c). The memory contains the string "kientrucmaytinh" stored in reverse order, with each character followed by a null terminator (\0). Below the table are navigation buttons and checkboxes for "Hexadecimal Addresses", "Hexadecimal Values", and "ASCII".

The bottom window, titled "Mars Messages" and "Run I/O", shows the program's output. It lists 16 input characters: k, i, e, n, t, r, u, c, m, a, y, t, i, n, h. The final output line is "Chuoi ky tu vua nhap la: hnityamcurtneik", which is the reverse of the input string. The program concludes with "-- program is finished running --". A "Clear" button is visible on the left side of the output window.

- ⇒ Kết quả khi nhập 'kientrucmaytinh' đã được đảo ngược thành 'hnityamcurtneik'
- ⇒ Dừng nhập thành công khi nhấn Enter

Edit
Execute

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	d c b a	h g f e	l k j i	p o n m	t l s q	p a h N	y k	t u t
0x10010020	\0 u h	\n \0 :	u h C \0	k i o	u t y	a u v	a h n	a l p
0x10010040	\0 \0 :	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

Hexadecimal Addresses
Hexadecimal Values
ASCII

Mars Messages
Run I/O

Clear

Nhập ký tự thứ 4: a  
Nhập ký tự thứ 5: e  
Nhập ký tự thứ 6: f  
Nhập ký tự thứ 7: g  
Nhập ký tự thứ 8: h  
Nhập ký tự thứ 9: i  
Nhập ký tự thứ 10: j  
Nhập ký tự thứ 11: k  
Nhập ký tự thứ 12: l  
Nhập ký tự thứ 13: m  
Nhập ký tự thứ 14: n  
Nhập ký tự thứ 15: o  
Nhập ký tự thứ 16: p  
Nhập ký tự thứ 17: q  
Nhập ký tự thứ 18: s  
Nhập ký tự thứ 19: l  
Nhập ký tự thứ 20: t  
Chuỗi ký tự vừa nhập là: t l s q p o n m l k j i h g f e d c b a  
-- program is finished running --

- ⇒ Trong trường hợp nhập tới 20 ký tự chương trình tự động dừng nhập
- ⇒ Đưa ra kết quả đảo ngược tương ứng