

Lab Report 05

Assignment 1

Code:

```
.data
```

```
test:      .asciiz "Hello World"
```

```
.text
```

```
        li      $v0, 4
```

```
        la      $a0, test
```

```
        syscall
```

Comments:

- The beginning address of the string is 0x10010000, which also corresponds to letter "H".
- The content of the first 4 bytes: H e l l
- The content of the next 4 bytes: o W o
- The content of the last 4 bytes: r l d \0

Assignment 2

Code:

```
.data
```

```
string1: .asciiz "The sum of "
```

```
string2: .asciiz " and "
```

```
string3: .asciiz " is "
```

```
.text
```

```
input:
```

```
        li      $v0, 5
```

```
        syscall
```

```
        add     $s0, $zero, $v0      # Store $s0
```

```
        li      $v0, 5
```

```
        syscall
```

```
        add     $s1, $zero, $v0      # Store $s1
```

```
sum:
```

```

        add        $s2, $s0, $s1        # $s2 = $s0 + $s1
output:
        li         $v0, 4
        la         $a0, string1
        syscall                                # Print "The sum of "
        li         $v0, 1
        add        $a0, $zero, $s0        # $a0 = $s0
        syscall                                # Print (s0)
        li         $v0, 4
        la         $a0, string2
        syscall                                # Print " and "
        li         $v0, 1
        add        $a0, $zero, $s1        # $a0 = $s1
        syscall                                # Print (s1)
        li         $v0, 4
        la         $a0, string3
        syscall                                # Print " is "
        li         $v0, 1
        add        $a0, $zero, $s2        # $a0 = sum
        syscall                                # Print (sum)

```

Result:

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24020005	addiu \$2,\$0,0x00000...	7: li \$v0, 5
	0x00400004	0x0000000c	syscall	8: syscall
	0x00400008	0x00028020	add \$16,\$0,\$2	9: add \$s0, \$zero, \$v0 # Store ..
	0x0040000c	0x24020005	addiu \$2,\$0,0x00000...	10: li \$v0, 5
	0x00400010	0x0000000c	syscall	11: syscall
	0x00400014	0x00028020	add \$17,\$0,\$2	12: add \$s1, \$zero, \$v0 # Store ..
	0x00400018	0x02119020	add \$18,\$16,\$17	14: add \$s2, \$s0, \$s1 # \$s2 = ..
	0x0040001c	0x24020005	addiu \$2,\$0,0x00000...	16: li \$v0, 4
	0x00400020	0x3c011001	lui \$1,0x00001001	17: la \$a0, string1
	0x00400024	0x34240000	ori \$4,\$1,0x00000000	
	0x00400028	0x0000000c	syscall	18: syscall # Print ..
	0x0040002c	0x24020001	addiu \$2,\$0,0x00000...	19: li \$v0, 1

Labels

Label	Address
a2.asm	
input	0x00400000
sum	0x00400018
output	0x0040001c
string1	0x10010000
string2	0x1001000c
string3	0x10010012

☒ Data
 ☒ Text

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x20656854	0x206d7573	0x0020666f	0x646e6120	0x65200020	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
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

0x10010000 (.data)

☒ Hexadecimal Addresses
 ☒ Hexadecimal Values
 ☐ ASCII

Mars Messages

Run I/O

4

5

The sum of 4 and 5 is 9

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

Clear

Assignment 3

Code:

.data

x: .space 32

destination string x, empty

y: .ascii "Hello"

source string y

.text

init:

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]
= 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

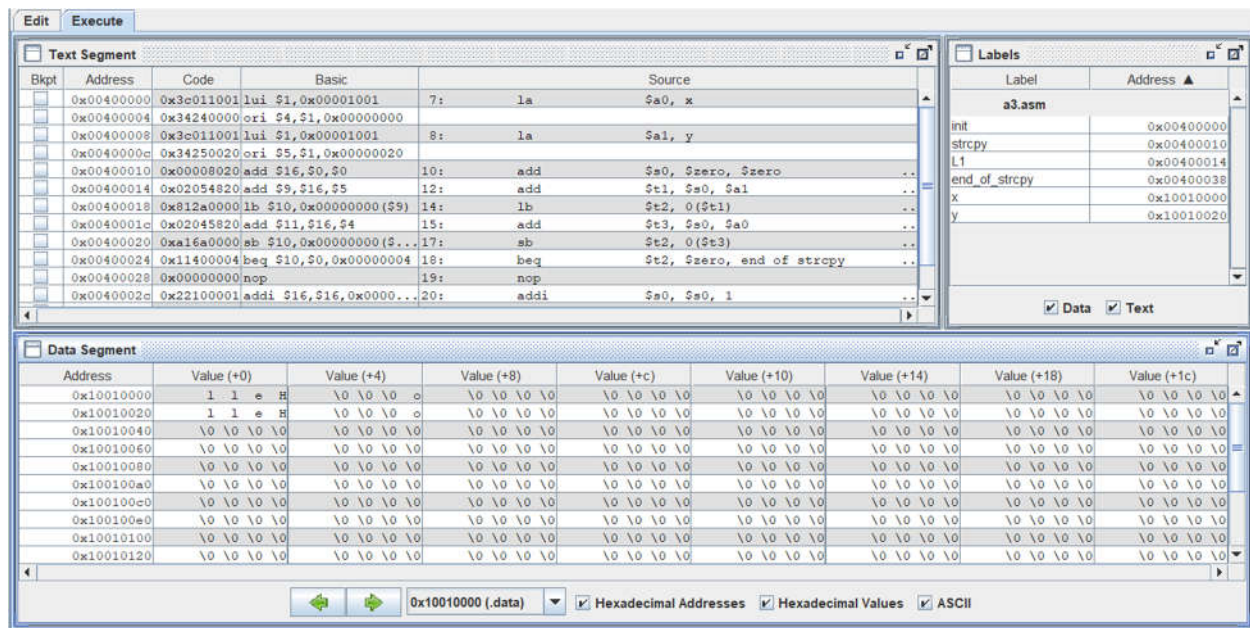
nop

end_of_strcpy:

Comments:

- The red code is where to load the address of x & y into the corresponding registers \$a0, \$a1.

Result:



Assignment 4

Code:

.data

string: .space 50

Message1: .ascii "Nhap xau: "

Message2: .ascii "Do dai xau la: "

.text

main:

get_string:

li \$v0, 54

la \$a0, Message1

la \$a1, string

la \$a2, 50

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:

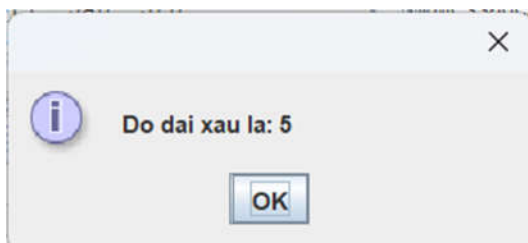
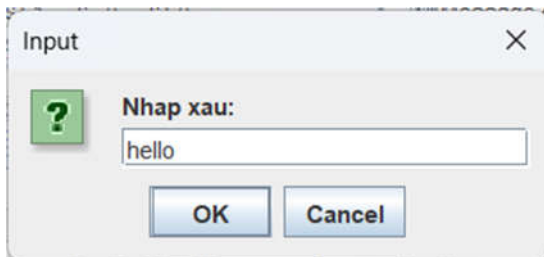
print_length:

```
li     $v0, 56
la     $a0, Message2
subi   $a1, $t0, 1
syscall
```

Comments:

- The red code is the implementation of the required parts.

Result:



Assignment 5

Code:

.data

```

string:      .space 21

.text
main:
init:
    la        $a0, string          # $a0 = address(string[0])
    add       $t0, $zero, $zero    # $t0 = i = 0
    li        $s0, 20              # Maximum size of string

scan_char:
    add       $t1, $a0, $t0        # $t1 = $a0 + $t0
                                        #   = address(string[i])

    li $v0, 12
    syscall

    sb        $v0, 0($t1)
    beq       $v0, '\n', end_of_string    # Stop if '\n' is inputted

    addi      $t0, $t0, 1
    beq       $t0, $s0, end_of_string    # Stop if maximum size is reached
    j         scan_char

end_of_string:
    add       $s1, $zero, $a0        # Store address of string in $s1
                                        # $a0 is dangerous to store in long

term!

reverse_string:

    subi      $t0, $t0, 1
    bltz      $t0, exit

```

```

add          $t1, $s1, $t0          # $t1 = $s1 + $t0
                                         # = address(string[i])

```

```
li           $v0, 11
```

```
lb           $a0, 0($t1)            # Load string[i]
```

```
syscall      # Print string[i]
```

```
j           reverse_string
```

exit:

Result:

The screenshot displays the Mars MIPS simulator interface. The top section shows the assembly code with columns for Bkpt, Address, Code, Basic, and Source. The code includes instructions like `lui $t1, 0x00001001`, `ori $t1, 0x00000000`, `add $t0, $zero, $zero`, `addiu $t1, $t0, 20`, `add $t1, $a0, $t0`, `li $v0, 12`, `syscall`, `sb $t0, 0($t1)`, `beq $v0, '\n', end_of_string`, `addi $t0, $t0, 1`, and `beq $t0, $a0, end_of_string`.

The right panel shows the Labels section with a table of labels and their addresses:

Label	Address
a5.asm	
main	0x00400000
init	0x00400000
scan_char	0x00400010
end_of_string	0x00400034
reverse_string	0x00400038
exit	0x00400058
string	0x10010000

The bottom section shows the Data Segment with a table of addresses and values:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\n c b a	\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
0x10010020	\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
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 \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
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	\0 \0 \0 \0
0x100100e0	\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
0x10010100	\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
0x10010120	\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

The bottom panel shows the Mars Messages section with a "Run I/O" button and a "Clear" button. The messages area displays the output of the program, which is "abcba".