Laboratory Exercise 5

Character string with SYSCALL function, and sorting

Goals

After this laboratory exercise, students known how to store a string in memory and know how to print out a string in the console. Furthermore, students know how to use some algorithms to sort elements in a string.

Preparation

Students should review the textbook Computer Organisation and Design by Patterson & Henessy (Section 2.8, 2.13, and 6.1).

About SYSCALL

The following table shows a number of system services, mainly for input and output, that are available for user mode. Contents in MIPS registers are not affected by a system call, except for some specified registers that store the returned results.

How to use SYSCALL system services

1. Load the service number in register $v0.
2. Load argument values, if any, in $a0, $a1, $a2, or $f12 as specified.
3. Issue the SYSCALL instruction.
4. Retrieve return values, if any, from result registers as specified.

Example: display the value stored in $t0 on the console

```
    li   $v0, 1            # service 1 is print integer
    li   $a0, 0x307        # the interger to be printed is 0x307
    syscall                # execute
```

Table of Frequently Available Services

Service	Code in $v0	Arguments	Result
print decimal integer	1	$a0 = integer to print	
print string	4	$a0 = address of null-terminated string to print	
read integer	5		$v0 contains integer read
read string	8	$a0 = address of input buffer $a1 = maximum number of characters to read	*See note below table*
exit	10	(terminate execution)	
print character	11	$a0 = character to print	*See note below table*
read character	12		$v0 contains character read

open file	13	\$a0 = address of null-terminated string containing filename \$a1 = flags \$a2 = mode	\$v0 contains file descriptor (negative if error). *See note below table*
read from file	14	\$a0 = file descriptor \$a1 = address of input buffer \$a2 = maximum number of characters to read	\$v0 contains number of characters read (0 if end-of-file, negative if error). *See note below table*
write to file	15	\$a0 = file descriptor \$a1 = address of output buffer \$a2 = number of characters to write	\$v0 contains number of characters written (negative if error). *See note below table*
close file	16	\$a0 = file descriptor	
exit2 (terminate with value)	17	\$a0 = termination result	*See note below table*
time (system time)	30		\$a0 = low order 32 bits of system time \$a1 = high order 32 bits of system time. *See note below table*
MIDI out	31	\$a0 = pitch (0-127) \$a1 = duration in milliseconds \$a2 = instrument (0-127) \$a3 = volume (0-127)	Generate tone and return immediately. *See note below table*
sleep	32	\$a0 = the length of time to sleep in milliseconds.	Causes the MARS Java thread to sleep for (at least) the specified number of milliseconds. This timing will not be precise, as the Java implementation will add some overhead.
MIDI out synchronous	33	\$a0 = pitch (0-127) \$a1 = duration in milliseconds \$a2 = instrument (0-127) \$a3 = volume (0-127)	Generate tone and return upon tone completion. *See note below table*
print integer in hexadecimal	34	\$a0 = integer to print	Displayed value is 8 hexadecimal digits, left-padding with zeroes if necessary.
print integer in binary	35	\$a0 = integer to print	Displayed value is 32 bits, left-padding with zeroes if necessary.
print integer as unsigned	36	\$a0 = integer to print	Displayed as unsigned decimal value.
(not used)	37-39		
set seed	40	\$a0 = i.d. of pseudorandom number generator (any int). \$a1 = seed for corresponding pseudorandom number generator.	No values are returned. Sets the seed of the corresponding underlying Java pseudorandom number generator (`java.util.Random`). *See note below table*
random int	41	\$a0 = i.d. of pseudorandom number generator (any int).	\$a0 contains the next pseudorandom, uniformly distributed int value from this random number generator's sequence. *See note below table*
random int range	42	\$a0 = i.d. of pseudorandom number generator	\$a0 contains pseudorandom, uniformly distributed int value in the range 0 = [int] [upper bound], drawn from this random

		(any int). $a1 = upper bound of range of returned values.	number generator's sequence. *See note below table*
ConfirmDialog	50	$a0 = address of null-terminated string that is the message to user	$a0 contains value of user-chosen option 0: Yes 1: No 2: Cancel
InputDialogInt	51	$a0 = address of null-terminated string that is the message to user	$a0 contains int read $a1 contains status value 0: OK status -1: input data cannot be correctly parsed -2: Cancel was chosen -3: OK was chosen but no data had been input into field
InputDialogString	54	$a0 = address of null-terminated string that is the message to user $a1 = address of input buffer $a2 = maximum number of characters to read	*See Service 8 note below table* $a1 contains status value 0: OK status. Buffer contains the input string. -2: Cancel was chosen. No change to buffer. -3: OK was chosen but no data had been input into field. No change to buffer. -4: length of the input string exceeded the specified maximum. Buffer contains the maximum allowable input string plus a terminating null.
MessageDialog	55	$a0 = address of null-terminated string that is the message to user $a1 = the type of message to be displayed: 0: error message, indicated by Error icon 1: information message, indicated by Information icon 2: warning message, indicated by Warning icon 3: question message, indicated by Question icon other: plain message (no icon displayed)	N/A
MessageDialogInt	56	$a0 = address of null-terminated string that is an information-type message to user $a1 = int value to display in string form after the first string	N/A
MessageDialogString	59	$a0 = address of null-terminated string that is an information-type message to user $a1 = address of null-terminated string to display after the first string	N/A

1. print decimal integer

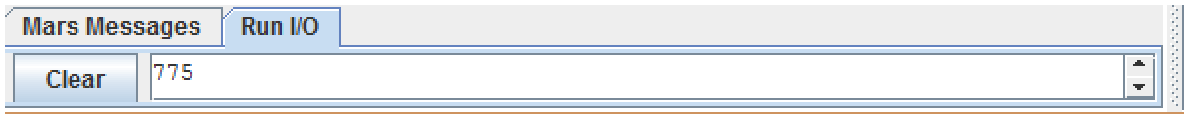
To print an integer to standard output (the console).
Argument(s):

$v0	= 1
$a0	= number to be printed

Return value:

none

Example:
```
   li   $v0, 1              # service 1 is print integer
   li   $a0, 0x307          # the interger to be printed is 0x307
   syscall                  # execute
```
and result is



```
775
```

2. MessageDialogInt

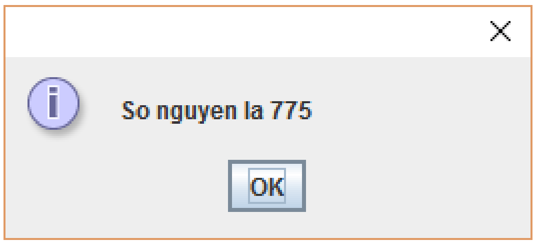
To show an integer to a information-type message dialog.
Argument(s):

$v0	= 56
$a0	= address of the null-terminated message string
$a1	= int value to display in string form after the first string

Return value:

none

Example:
```
.data
Message: .asciiz "So nguyen la "
.text
   li   $v0, 56
   la   $a0, Message
   li   $a1, 0x307          # the interger to be printed is 0x307
   syscall                  # execute
```
and result is


So nguyen la 775

3. print string

To print a string to standard output (the console).
Argument(s):

$v0	= 4
$a0	= value to be printed

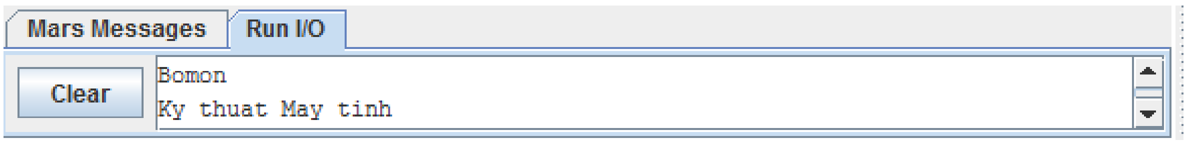
Return value:

none

Example:

```
.data
Message: .asciiz "Bomon \nKy thuat May tinh"
.text
   li   $v0, 4
   la   $a0, Message
   syscall
```

and result is



```
Bomon
Ky thuat May tinh
```

4. MessageDialogString

To show a string to a information-type message dialog

Argument(s):

$v0	= 59
$a0	= address of the null-terminated message string
$a1	= address of null-terminated string to display

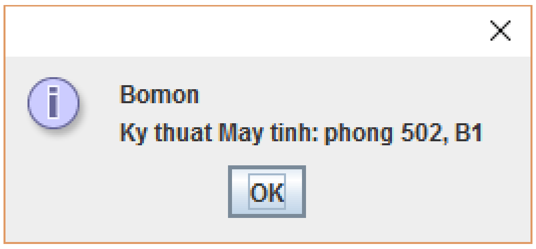
Return value:

none

Example:

```
.data
Message: .asciiz "Bomon \nKy thuat May tinh:"
Address: .asciiz " phong 502, B1"
.text
   li   $v0, 59
   la   $a0, Message
   la   $a1, Address
   syscall
```

and result is



Bomon
Ky thuat May tinh: phong 502, B1

5. read integer

To get a integer from standard input (the keyboard).

Argument(s):

$v0 = 5

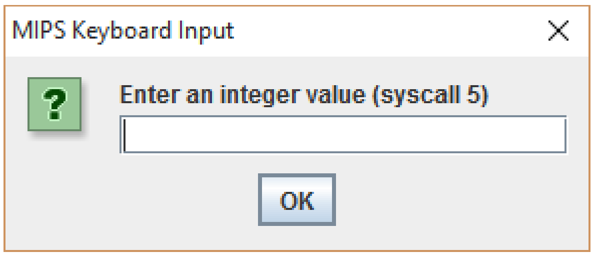
Return value:

$v0 = contains integer read

Example:

```
     li      $v0, 5
     syscall
```

and result is


MIPS Keyboard Input

Enter an integer value (syscall 5)

6. InputDialogInt

To show a message dialog to read an integer with content parser

Argument(s):

$v0 = 51
$a0 = address of the null-terminated message string

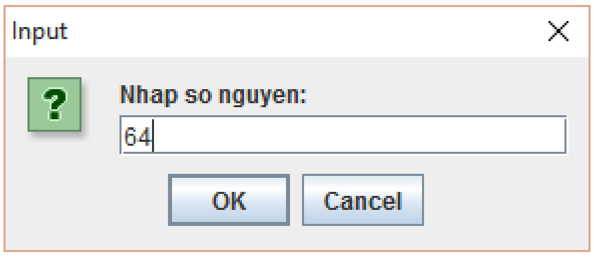
Return value:

$a0 = contains int read
$a1 contains status value
0: OK status
-1: input data cannot be correctly parsed
-2: Cancel was chosen
-3: OK was chosen but no data had been input into field

Example:

```
.data
Message: .asciiz "Nhap so nguyen:”
.text
   li    $v0, 51
   la    $a0, Message
   syscall
```

and result is


Input

Nhap so nguyen:

64

7. read string

To get a string from standard input (the keyboard).

Argument(s):

$v0 = 8
$a0 = address of input buffer
$a1 = maximum number of characters to read

Return value:

none

Remarks:

For specified length n, string length cannot be longer than n-1.

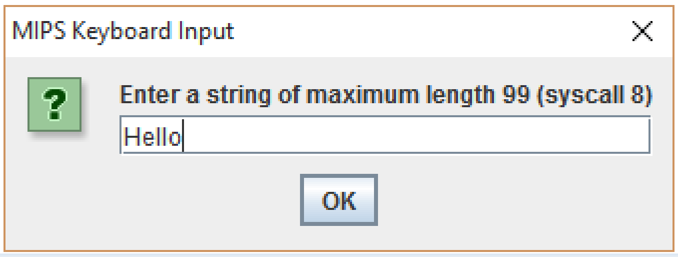
- If less than that, add newline to end.
- In either case, then pad with null byte

Just in special cases:
If n = 1, input is ignored and null byte is placed at buffer address.
If n < 1, input is ignored and nothing is written to the buffer.

Example:

```
.data
Message: .space 100    # Buffer 100 byte chua chuoi ki tu can
.text
   li  $v0, 8
   la  $a0, Message
   li  $a1, 100
   syscall
```

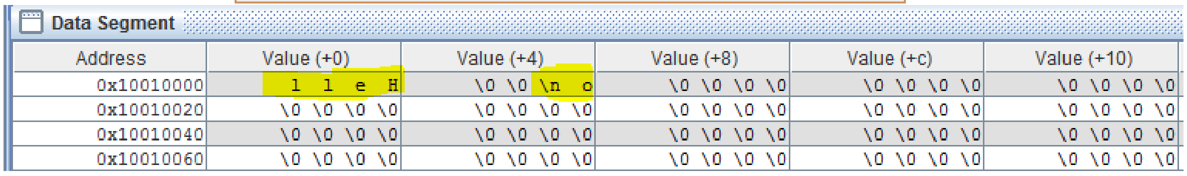
and result is


MIPS Keyboard Input

Enter a string of maximum length 99 (syscall 8)

Hello

OK


Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	l l e H	\0 \0 \n o	\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
0x10010040	\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

8. InputDialogString

To show a message dialog to read a string with content parser

Argument(s):

$v0	= 54
$a0	= address of the null-terminated message string
$a1	= address of input buffer
$a2	= maximum number of characters to read

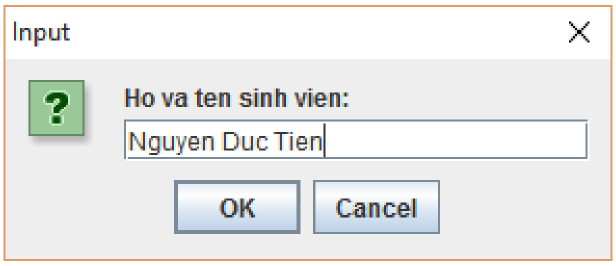
Return value:

$a1 contains status value
0: OK status
-2: OKstatus is chosen but no data has been input into field. No change to buffer.
-3: OK status is chosen but no data has been input into field.
-4: length of the input string exceeds the specified maximum. Buffer contains the maximum allowable input string plus a terminating null.

Example:

```
.data
Message: .asciiz "Ho va ten sinh vien:"
string:  .space   100
.text
   li    $v0, 54
   la    $a0, Message
   la    $a1, string
   la    $a2, 100
   syscall
```

and result is



Ho va ten sinh vien:

Nguyen Duc Tien

9. print character

To print a character to standard output (the console).

Argument(s):

$v0	= 11
$a0	= character to print (at the lowest significant byte)

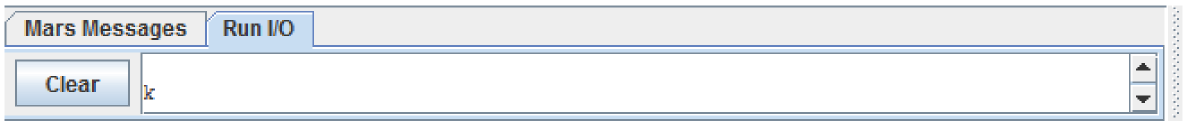
Return value:

none

Example:

```
   li   $v0, 11
   li   $a0, 'k'
   syscall
```

and result is



```
k
```

10. read character

To get a character from standard output (the keyboard).

Argument(s):

$v0	= 12

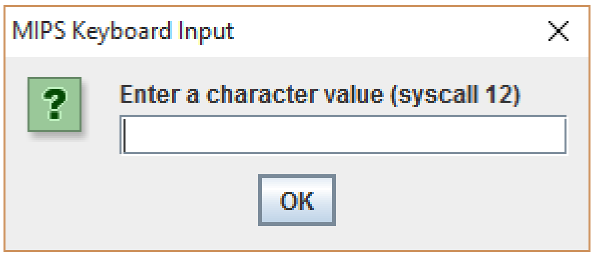
Return value:

$v0	contains character read

Example:

```
   li   $v0, 12
   syscall
```

and result is



Enter a character value (syscall 12)

11. ConfirmDialog

To show a message box with 3 buttons: Yes | No | Cancel

Argument(s):

$v0 = 50
$a0 = address of the null-terminated message string

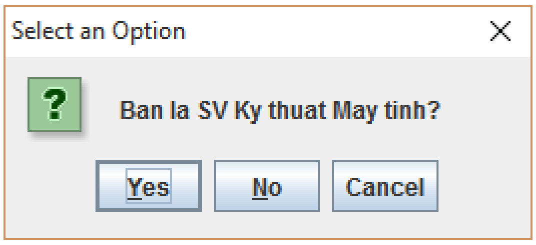
Return value:

$a0 = contains value of user-chosen option
0: Yes
1: No
2: Cancel

Example:

```
.data
Message: .asciiz "Ban la SV Ky thuat May tinh?"
.text
   li   $v0, 50
   la   $a0, Message
   syscall
```

and result is


Select an Option

Ban la SV Ky thuat May tinh?

12. MessageDialog

To show a message box with icon and button OK only

Argument(s):

$v0 = 55
$a0 = address of the null-terminated message string
$a1 = the type of message to be displayed:
0: error message, indicated by Error icon
1: information message, indicated by Information
icon
2: warning message, indicated by Warning icon
3: question message, indicated by Question icon
other: plain message (no icon displayed)

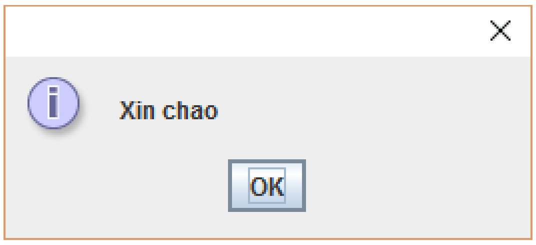
Return value:

none

Example:

```
.data
Message: .asciiz "Xin chao"
.text
   li   $v0, 55
   la   $a0, Message
   syscall
```

and result is


Xin chao

13. MIDI out

To make a sound

Argument(s):

$v0	= 31
$a0	= pitch (0-127)
$a1	= duration in milliseconds
$a2	= instrument (0-127)
$a3	= volume (0-127)

Return value:

Generate tone and return immediately

Example:

```
    li $v0, 33
    li $a0, 42     #pitch
    li $a1, 2000   #time
    li $a2, 0      #musical instrusment
    li $a3, 212    #volume
```

14. MIDI out synchronous

To make a sound

Argument(s):

$v0	= 33
$a0	= pitch (0-127)
$a1	= duration in milliseconds
$a2	= instrument (0-127)
$a3	= volume (0-127)

Return value:

Generate tone and return upon tone completion

Example:

```
    li $v0, 33
    li $a0, 42     #pitch
    li $a1, 2000   #time
    li $a2, 0      #musical instrusment
    li $a3, 212    #volume
    syscall
```

15. Exit

Terminate the software. Make sure that there is no EXIT instruction in the Instruction Set of any processors. EXIT is a service belongs to Operating System.

Argument(s):

$v0 = 10

Return value:

none

Example:

```
    li         $v0, 10     #exit
    syscall
```

16. Exit with code

Terminate the software. Make sure that there is no EXIT instruction in the Instruction Set of any processors. EXIT is a service belongs to Operating System.

Argument(s):

$v0	= 17
$a0	= termination result

Return value:

none

Example:

```
    li       $v0, 17       # exit
    li       $a0, 3        # with error code = 3
    syscall
```

Sample Code and Assignments

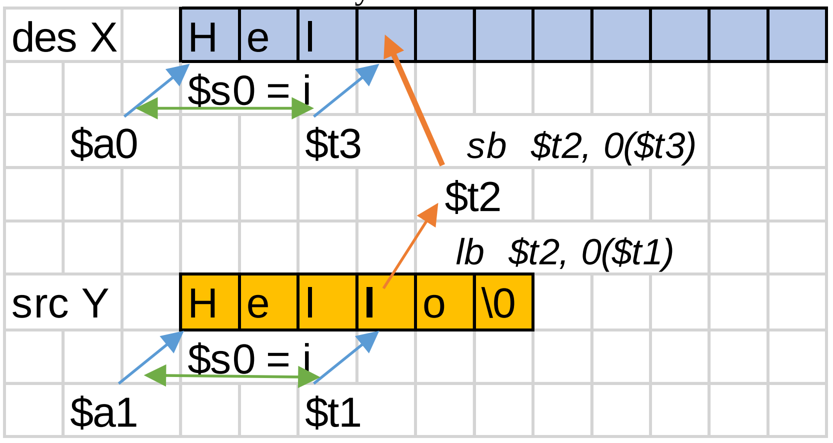
Sample Code 1

The following simple assembly program displays the string “Hello World” on the console by using the *printf* function. Read this example carefully and see how to pass parameters to the *printf* function.

```
#Laboratory Exercise 5, Sample Code 1
.data
test: .asciiz "Hello World"
.text
   li  $v0, 4
   la  $a0, test
   syscall
```

Sample Code 2

The *strcpy* procedure copies string *Y* into string *X*. Both strings have a null terminator (like the termination convention of C). Read this example carefully and try to understand the code line by line.



```
#Laboratory Exercise 5, Sample Code 2
.data
x: .space 1000                     # destination string x, empty
y: .asciiz "Hello"                 # source string y

.text
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  bb$s0,$s0,1             #s0=s0 + 1 <-> i=i+1
      j     L1                      #next character
      nop
end_of_strcpy:
```

Sample Code 3

The following program measures the length of a null-terminated string. Read this example carefully and try to understand the code line by line.

```
#Laboratory Exercise 5, Sample Code 3
.data
string:       .space   50
Message1:     .asciiz "Nhap xau:”
Message2:     .asciiz "Do dai la "
.text
main:
get_string:    # TODO

get_length:    la   $a0, string          # a0 = Address(string[0])
               xor  $v0, $zero, $zero    # v0 = length = 0
               xor  $t0, $zero, $zero    # t0 = i = 0
check_char:    add  $t1, $a0, $t0        # t1 = a0 + t0
                                         #= Address(string[0]+i)
               lb   $t2, 0($t1)          # t2 = string[i]
               beq  $t2,$zero,end_of_str # Is null char?
               addi $v0, $v0, 1          # v0=v0+1->length=length+1
               addi $t0, $t0, 1          # t0=t0+1->i = i + 1
               j    check_char
end_of_str:
end_of_get_length:
print_length: # TODO
```

Assignment 1

Create a new project in the Mars simulator to implement the program in Sample Code 1. Run the code and observe the result. Go to the data memory section and check how the string (i.e. test) is stored in memory.

Assignment 2

Create a new project in the Mars simulator to print out the sum of two register $s0 and $s1 according to the following format:

“The sum of (*s0*) and (*s1*) is (*result*)”.

Assignment 3

Create a new project in the Mars simulator to implement the program in Sample Code 2. Assign a different string to variable *y* and implement *strcpy* function agian. Run the code and observe the result.

Assignment 4

Reference the Sample Code 3 and use the syscall function to get a string from the input dialog, and output the string length to the message dialog.

Assignment 5

Write a program that let user input a string. The input process will be terminated when user press the *Enter* key or the length of the string exceeds 20 characters. Print out the reverse string on the console.

Assignment 6

Write a program that multiplies two 32-bit integers and then prints out the 64-bit product on the console. The sample ouput should follow this format:

“The multiplication of **X** base 10 (or **X’** base 16) and **Y** base 10 (or **Y’** base 16) is **Z** base 10 (or **Z’** base 16).”

Questions

- What the difference between the string in C and Java?
- In C, how many characters can be stored within 8 bytes?
- In Java, how many characters can be stored within 8 bytes?