Lab 5. Character string with ECALL function

Goals

After this laboratory exercise, you should understand the mechanism of storing a string. You will be able to program to process string and put string to console. In addition, you should know how to sort a list of elements.

References

Preparation

About ECALL

A number of system services, mainly for input and output, are available for use by your RISC-V program. They are described in below. When using ECALL functions, RISC-V register contents are not affected by a system call, except for result registers. The list of ECAL functions can be found in the Help of RARS tool.

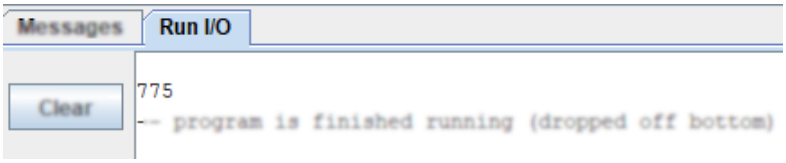
How to use ECALL system services

1. Load the service number in register a7.
2. Load argument values, if any, in a0, a1, a2, ...
3. Issue the ECALL instruction.
4. Retrieve return values, if any, from result registers as specified.

Example: display the value stored in a0 on the console

```
.text
   Li    a7, 1      # service 1 is print integer
   li    a0, 0x307  # the integer to be printed is 0x307
   ecall            # execute
```

The result is shown in Run I/O:



Common use services

1. print decimal integer

print an integer to standard output (the console).

Argument(s):

a7 = 1
a0 = number to be printed

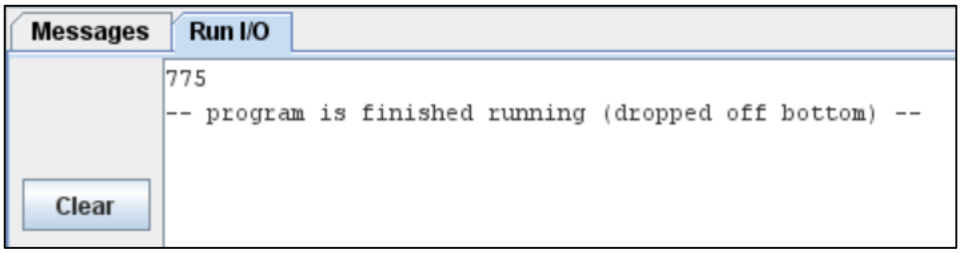
Return value:

none

Example:

```
.text
   li a7, 1
   li a0, 0x307
   ecall
```

The result is



2. MessageDialogInt

Show an integer to a information-type message dialog.

Argument(s):

a7 = 56
a0 = address of the null-terminated message string
a1 = int value to display in string form after the first string

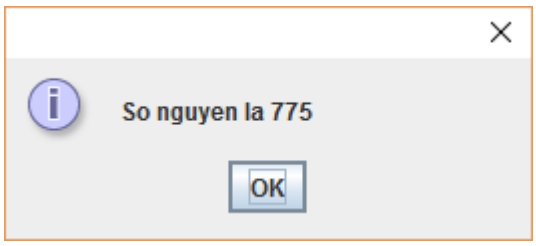
Return value:

none

Example:

```
.data
message: .asciz "So nguyen la: "
.text
   li a7, 56
   la a0, message
   li a1, 0x307
   ecall
```

And result is:



3. print string

Formatted print to standard output (the console).
Argument(s):

a7 = 4
a0 = the address of string to print

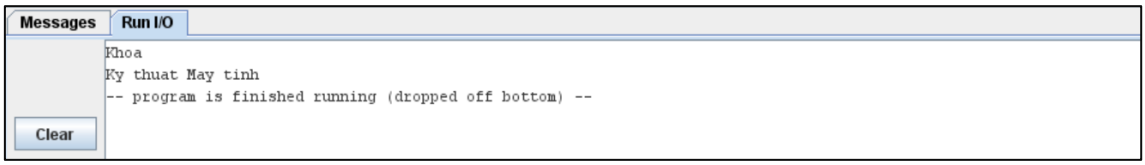
Return value:

none

Example:

```
.data
message: .asciz "Khoa \nKy thuat May tinh"
.text
   li a7, 4
   la a0, message
   ecall
```

And result is:



4. MessageDialogString

Show a string to a information-type message dialog
Argument(s):

a7 = 59
a0 = address of the null-terminated message string
a1 = address of null-terminated string to display

Return value:

none

Example:

```
.data
message: .asciz "Truong: "
address: .asciz "Cong nghe thong tin va Truyen thong"
.text
   li  a7, 59
   la  a0, message
   la  a1, address
   ecall
```

And result is:

5. read integer

Get a integer from standard input (the keyboard).
Argument(s):

a7 = 5

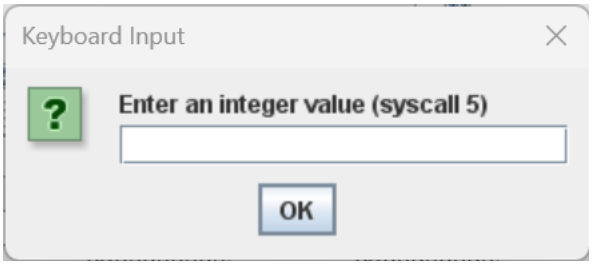
Return value:

a0 = contains integer read

Example:

```
.text
   li a7, 5
   ecall
```

And result is:



Notice: The services 5, 6, 7, 8, 12 can be inputted from Run I/O or in a popup dialog depending on the selection in menu Settings / Popup dialog for input syscalls (5, 6, 7, 8, 12)

6. read string

Get a string from standard input (the keyboard).
Argument(s):

a7 = 8
a0 = address of input buffer
a1 = maximum number of characters to read

Return value:

none

Remarks:

For specified length n, string can be no longer than n-1.

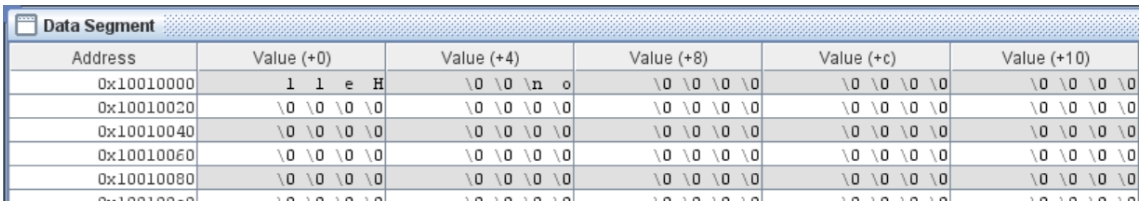
- If less than that, adds newline to end.
- In either case, then pads with null byte

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

Example:

```
.data
buffer: .space 100    # Buffer 100 byte for storing the
input string
.text
   li  a7, 8
   la  a0, buffer
   li  a1, 100
   ecall
```

And the result is:



7. InputDialogString

Show a message dialog to read a string with content parser

Argument(s):

a7 = 54
a0 = address of the null-terminated message string
a1 = address of input buffer
a2 = maximum number of characters to read

Return value:

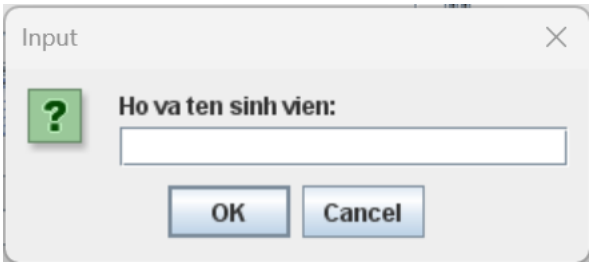
a1 contains status value

0: OK status
-2: OK was chosen but no data had been input into field. No change to buffer.
-3: OK was chosen but no data had been input into field
-4: length of the input string exceeded the specified maximum. Buffer contains the maximum allowable input string plus a terminating null.

Example:

```
.data
message: .asciz "Ho va ten sinh vien:"
buffer:  .space   100
.text
   li    a7, 54
   la    a0, message
   la    a1, buffer
   li    a2, 100
   ecall
```

And result is:



8. print character

Print a character to standard output (the console).

Argument(s):

a7 = 11
a0 = character to print (at the lowest significant byte)

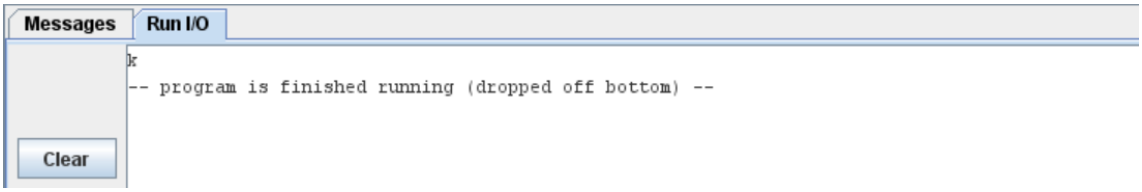
Return value:

none

Example:

```
.text
   li    a7, 11
   li    a0, 'k'
   ecall
```

And result is:



9. read character

Get a character from standard output (the keyboard).

Argument(s):

a7 = 12

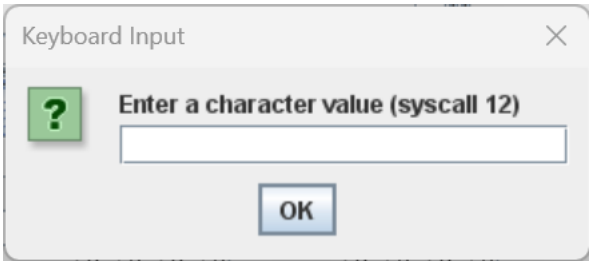
Return value:

a0 contains character read

Example:

```
.text
   li   a7, 12
   ecall
```

And result is:



10. ConfirmDialog

Show a message bog with 3 button: Yes | No | Cancel

Argument(s):

a7 = 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
question: .asciz "Ban co phai la SV Ky thuat May tinh?"
.text
   li   a7, 50
   la   a0, question
   ecall
```

And result is:

11. MessageDialog

Show a message bog with icon and button OK only

Argument(s):

	a7	= 55
	a0	= address of the null-terminated message string
	aa1	= 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: .asciz "Chuong trinh bi loi."
.text
   li   a7, 55
   la   a0, message
   li a1, 0
   ecall
```

And result is:

12. Exit

Terminated the program. Make sense that there is no EXIT instruction in the Instruction Set of any processors. Exit is a service belongs to Operating System.

Argument(s):

	a7	= 10

Return value:

none

Example:

```
li   a7, 10
ecall
```

Assignments at Home and at Lab

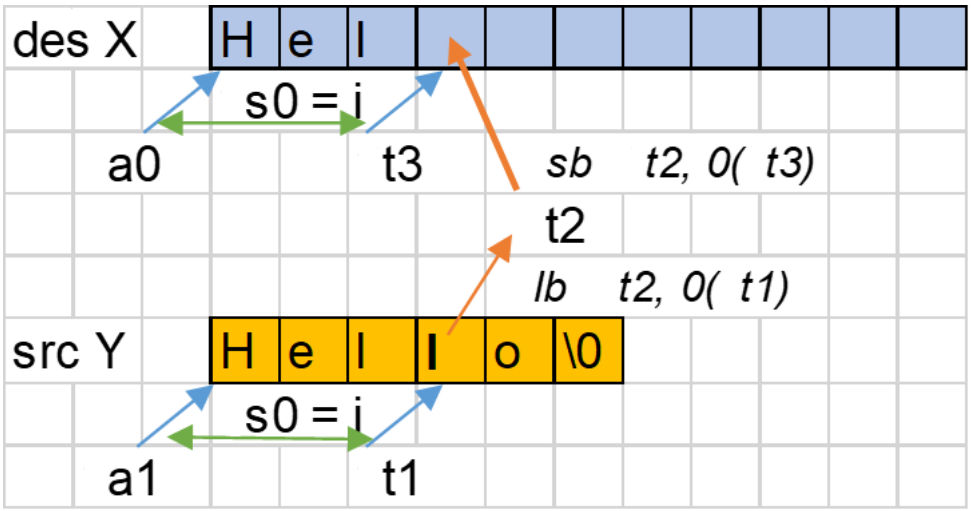
Home Assignment 1

The following simple assembly program will display a welcome string. We use **printf** function for this purpose. Read this example carefully, pay attention to the way to pass parameters for printf function.

```
# Laboratory Exercise 5, Home Assignment 1
.data
test: .asciz "Hello World"
.text
   li  a7, 4
   la  a0, test
   ecall
```

Home Assignment 2

Procedure **strcpy** copies string y to string x using the null byte termination convention of C. Read this example carefully, try to understand all of this code section.




```
# Laboratory Exercise 5, Home Assignment 2
.data
x: .space 32         # destination string x, empty
y: .asciz "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
    addi   s0, s0, 1       # s0=s0 + 1 <-> i=i+1
    j      L1              # next character
end_of_strcpy:
```

Home Assignment 3

The following program count the length of a null-terminated string. Read this example carefully, analyse each line of code

```
# Laboratory Exercise 5, Home Assignment 3
.data
string: .space 50
message1: .asciz  "Nhap xau: "
message2: .asciz  "Do dai xau la: "

.text
main:
get_string:
    # TODO Input string from keyboard
get_length:
    la     a0, string           # a0 = address(string[0])
    li     t0, 0                # 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   t0, t0, 1            # t0 = t0 + 1 -> i = i + 1
    j  check_char
end_of_str:
end_of_get_length:
print_length:
    # TODO print result to console
```

Assignment 1

Create a new project to implement the program in Home Assignment 1. Compile and upload to simulator. Run and observe the result. Go to Data Segment, check how test string are stored and packed in memory

Assignment 2

Create a new project to print the sum of two register $s0 and $s1 according to this format:

"The sum of (s0) and (s1) is (result)"

Assignment 3

Create a new project to implement the program in Home Assignment 2. Add more instructions to assign a test string for y variable, and implement *strcpy* function. Compile and upload to simulator. Run and observe the result

Assignment 4

Accomplish the Home Assignment 3 with ecall function to get a string from dialog, and show the length to message dialog

Assignment 5

Write a program that let user input a string. Input process will be terminated when user press Enter or then length of the string exceed 20 characters. Print the reverse string.

Conclusions