COMP2611: Computer Organization

MIPS syscall services

Overview

- You will learn the following in this lab:
  - how to perform a system service using the instruction *syscall* in a MIPS program.

MIPS syscall services

- A MIPS instruction syscall is defined to perform a system service, e.g., Console Input/Output.
- Run the example program printString.s which uses the syscall to print the string "Hello World" to the console.
- Before executing the syscall instruction, you need to:
  - store the *system call code* (an integer) in the register v0, and the service performed by the syscall is determined by this register value (at the moment of executing the syscall instruction) .
  - pass any argument(s) for the syscall service via some particular register(s), e.g., passing the output value in the register a0 for printing an integer to the console.

Common syscall services

- Some common syscall services (you must know the yellow ones):

Service	System Call Code ($v0)	Arguments	Result	Example
print_int	**1**	**$a0=integer**		**li $v0, 1 li $a0, 100 syscall**
print_float	2	$f12=float		
print_double	3	$f12=double		
print_string	**4**	**$a0=start address of the string**		
read_int	**5**		**integer (in $v0)**	**li $v0, 5 syscall # $v0 = input value**
read_float	6		float (in $f0)	
read_double	7		double (in $f0)	
read_string	8	$a0=buffer, $a1=length		
sbrk	9	$a0=amount	address (in $v0)	
exit	**10**			**li $v0, 10 syscall**

Printing a string to console

In C++	In MIPS

// C++ version
// declare the string mesg
char mesg[] =
{'H', 'e', 'l', 'l', 'o', ' ',
'W', 'o', 'r', 'l', 'd', '\n', '\0' };
 	
// main is the default
//starting point of the program
void main() {
 	
cout << mesg;
 	
}
	#------ Data Segment ---------
	.data
	# declare the string mesg
	mesg: .asciiz "Hello World\n"
	
	#------ Text Segment ---------
	.text
	
	.globl main
	main:
	
	# Execute the "print_str" system call
	li $v0, 4
	la $a0, mesg
	syscall

Address	
Mesg	'H'
mesg+1	'e'
mesg+2	'l'
mesg+3	'l'
mesg+4	'o'
mesg+5	' '
mesg+6	'W'
mesg+7	'o'
mesg+8	'r'
mesg+9	'l'
mesg+10	'd'
mesg+11	'\n'
mesg+12	'\0'

Printing a string to console

In C++	In MIPS

// C++ version
// declare the string mesg
char mesg[] =
{'H', 'e', 'l', 'l', 'o', ' ',
'W', 'o', 'r', 'l', 'd', '\n', '\0' };
 	
// main is the default
//starting point of the program
void main() {
 	
cout << mesg;
 	
}
	#------ Data Segment ---------
	.data
	# declare the string mesg
	mesg: .asciiz "Hello World\n"
	
	#------ Text Segment
	.text
	
	.globl main
	main:
	
	# Execute the "print_str" system call
	li $v0, 4
	la $a0, mesg
	syscall

Setting v0 to 4, the processor knows we need to print a string to the console when executing a syscall.

Address	
Mesg	'H'
mesg+1	'e'
mesg+2	'l'
mesg+3	'l'
mesg+4	'o'
mesg+5	' '
mesg+6	'W'
mesg+7	'o'
mesg+8	'r'
mesg+9	'l'
mesg+10	'd'
mesg+11	'\n'
mesg+12	'\0'

Printing a string to console

In C++	In MIPS

// C++ version
// declare the string mesg
char mesg[] =
{'H', 'e', 'l', 'l', 'o', ' ',
'W', 'o', 'r', 'l', 'd', '\n', '\0' };
 	
// main is the default
//starting point of the program
v
}
 	
When **la $a0, mesg** is executed, the starting address of the string will be assigned to the register a0.
	#------ Data Segment ---------
	.data
	# declare the string mesg
	mesg: .asciiz "Hello World\n"
	
	#------ Text Segment
	.text
	
	.globl mair
	main:
	
	# Execute the "print_str" system call
	li $v0, 4
	la $a0, mesg
	syscall
	
	Setting v0 to 4, the processor knows we need to print a string to the console when executing a syscall.

Address	
Mesg	'H'
mesg+1	'e'
mesg+2	'l'
mesg+3	'l'
mesg+4	'o'
mesg+5	' '
mesg+6	'W'
mesg+7	'o'
mesg+8	'r'
mesg+9	'l'
mesg+10	'd'
mesg+11	'\n'
mesg+12	'\0'

Printing a string to console

In C++

```
// C++ version
// declare the string mesg
char mesg[] =
   {'H', 'e', 'l', 'l', 'o', ' ',
    'W', 'o', 'r', 'l', 'd', '\n', '\0' };

// main is the default
//starting point of the program
v
}
```

In MIPS

```
#------ D
       .da
# declare the string mesg
mesg:   .asciiz "Hello World\n"

#------ Text Segment
       .text

       .globl mair
main:

# Execute the "print_str" system call
       li $v0, 4
       la $a0, mesg
       syscall
```

e.g., if mesg (character 'H') is located at the 1001-th byte of memory, then a0 = 1001.

When **la $a0, mesg** is executed, the starting address of the string will be assigned to the register a0.

Setting v0 to 4, the processor knows we need to print a string to the console when executing a syscall.

Address	
Mesg	'H'
mesg+1	'e'
mesg+2	'l'
mesg+3	'l'
mesg+4	'o'
mesg+5	' '
mesg+6	'W'
mesg+7	'o'
mesg+8	'r'
mesg+9	'l'
mesg+10	'd'
mesg+11	'\n'
mesg+12	'\0'

Printing a string to console

In C++

```
// C++ version
// declare the string mesg
char mesg[] =
   {'H', 'e', 'l', 'l', 'o', ' ',
    'W', 'o', 'r', 'l', 'd', '\n', '\0' };

// main is the default
//starting point of the program
v
}
```

In MIPS

```
#------ D
       .da
# declare the string mesg
mesg:   .asciiz "Hello World\n"

#------ Text Segment
       .text

       .globl mair
main:

# Execute the "p
       li $v0, 4
       la $a0, me
       syscall
```

e.g., if mesg (character 'H') is located at the 1001-th byte of memory, then a0 = 1001.

Address	
Mesg	'H'
mesg+1	'e'
mesg+2	'l'
mesg+3	'l'
mesg+4	'o'
mesg+5	' '
mesg+6	'W'
mesg+7	'o'
mesg+8	'r'
mesg+9	'l'
mesg+10	'd'
mesg+11	'\n'
mesg+12	'\0'

When **la $a0, mesg** is executed, the starting address of the string will be assigned to the register a0.

Setting v0 to 4, the processor knows we need to print a string to the console when executing a syscall.

After executing syscall, the processor **reads the memory byte by byte** from the address in a0 (e.g. 1001--> 1002 --> 1003 ... and so on). The corresponding **character** will be **displayed one by one until the end of string character ('\0') is read.**

Example programs

- Try the following example programs:
  - printString.s (for Printing a string to the console).
  - printInt.s (for Printing an integer to the console).
  - readInt.s (for Reading an integer from the console).

Syscall service "exit"

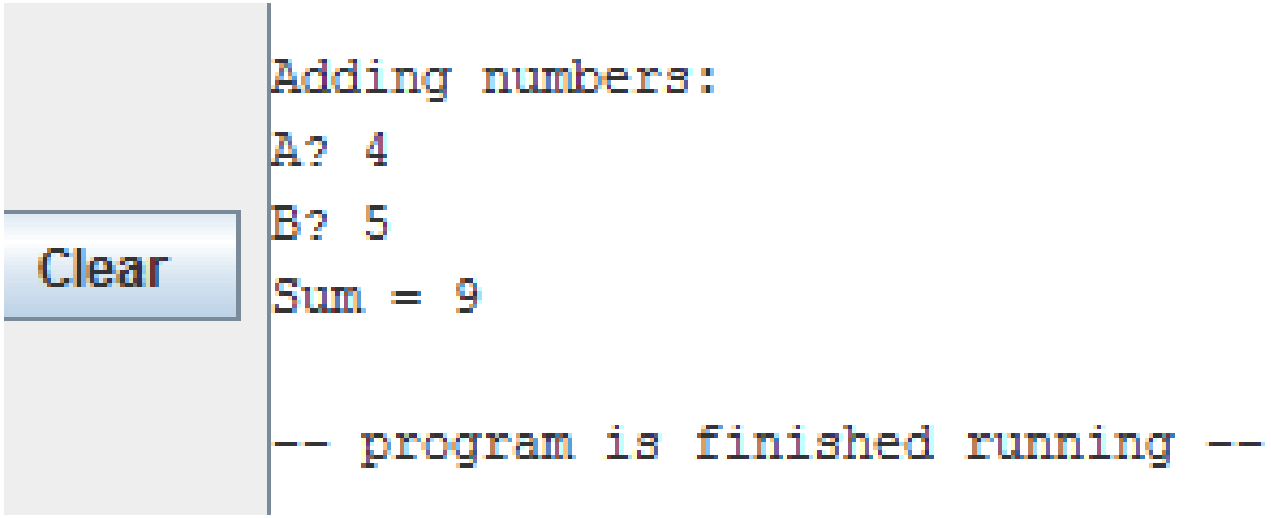
- The syscall service "exit" terminates the program immediately after this syscall instruction is executed.

```
# starting main program
.text
.globl __start
__start:

addi $t0, $zero, 5
addi $t1, $t0, -2

li $v0, 10
syscall     # the program is terminated after executing this syscall

# the codes below will never be executed
addi $t1, $t1, 1
add $t1, $t0, $t1
```

- Try the example programs exitExample1.s and exitExample2.s.

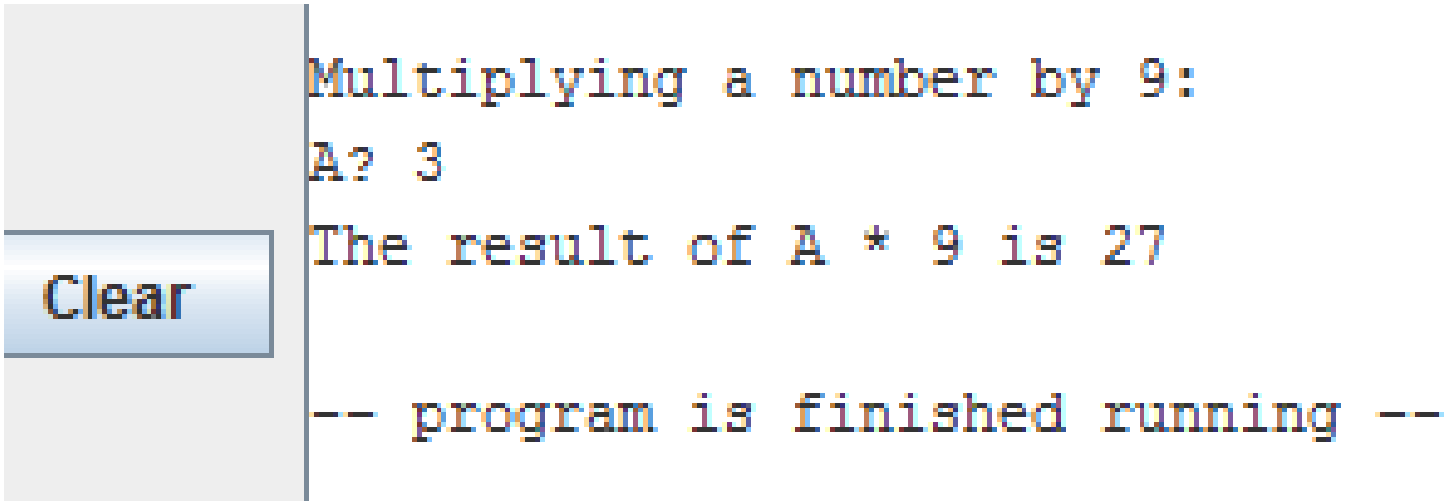
Exercise

- Question 1: Adding two number and output the result
  - write a MIPS program that prompts the user for two integer inputs,
  - and displays the sum of the two integers,
  - the program should be able to exit using the syscall service after displaying the sum,
  - you do not need to verify the correctness of the input integers.



Exercise

- Question 2: Multiplying a number by 9
  - write a MIPS program that prompts the user for one integer input,
  - and displays the result of the number * 9
  - the program should be able to exit using the syscall service after displaying the result,
  - you do not need to verify the correctness of the input integer
  - you are not allowed to use mult or multu instructions.



Conclusion

- You have learnt:
  - how to perform a system service using the instruction syscall in a MIPS program.