DHA Suffa University
Department of Computer Science
Computer Organization & Assembly Language
Spring 2021
Lab # 3 (Strings & Integers Practices in MIPS Programming)

RECAP

To discuss data types and literal, data declarations, load/store instructions and system calls.

Data Types and Literals:

Data types:

- Instructions are all 32 bits
- byte(8 bits), halfword (2 bytes), word (4 bytes)
- a character requires 1 byte of storage
- an integer requires 1 word (4 bytes) of storage

Literals:

- numbers entered as is. e.g. 4
- characters enclosed in single quotes. e.g. 'b'
- strings enclosed in double quotes. e.g. "A string"

Data Declarations:

Format for declarations:

name: storage_type value(s)

- Create storage for variable of specified type with given name and specified value
- Value usually gives initial value(s); for storage type .space, gives number of spaces to be allocated

Note: labels always followed by colon (:)

Examples

```
var1:  .word     3                    # create a single integer variable with initial value 3
array1: .byte    'a','b'             # create a 2-element character array with elements initialized
                                     # to a  and  b
array2: .space  40        # allocate 40 consecutive bytes, with storage uninitialized
                          # could be used as a 40-element character array, or a
                              #10-element integer array; a comment should indicate which!
string1:    .asciiz         "Print this.\n"       # declaration for string variable
```

Load / Store Instructions:

RAM access only allowed with load and store instructions all other instructions use register operands

Load:

```
lw        register_destination, RAM_source

          #copy word (4 bytes) at source RAM location to destination register.

lb        register_destination, RAM_source

                  #copy byte at source RAM location to low-order byte of destination register,
            # and sign-e.g.tend to higher-order bytes
```

Store word:

```
sw        register_source, RAM_destination

          #store word in source register into RAM destination

sb        register_source, RAM_destination

          #store byte (low-order) in source register into RAM destination
```

Load immediate:

```
li      register_destination, value

        #load immediate value into destination register
```

Example:

```
.data
var1:   .word   23              # declare storage for var1; initial value is 23

.text
.globl main
main:

lw      $t0, var1               # load contents of RAM location into register $t0:  $t0 = var1
li      $t1, 5          #  $t1 = 5   ("load immediate")
sw      $t1, var1               # store contents of register $t1 into RAM:  var1 = $t1
```

Printing a String:

To print a value or a string on the console, we need to make a system call using syscall instruction. For the computer to understand which system call to initiate, we have to provide the number in $v0 register. For printing an integer, the value of $v0 should be 1 and the number to be printed should be loaded in $a0 register. For printing a string, the value of $v0 should be 4 and the address of the string to be printed should be loaded in $a0 register using instruction “la”. See the following codes for example.

```
.data
msg1: .asciiz “Hello World”
.text
.globl main
main:

# to print a string
```

```
la $a0, msg1     # load the address referred by msg1 in the register a0
li $v0, 4                        # v0 should 4 for printing string
syscall
```

Taking Input:

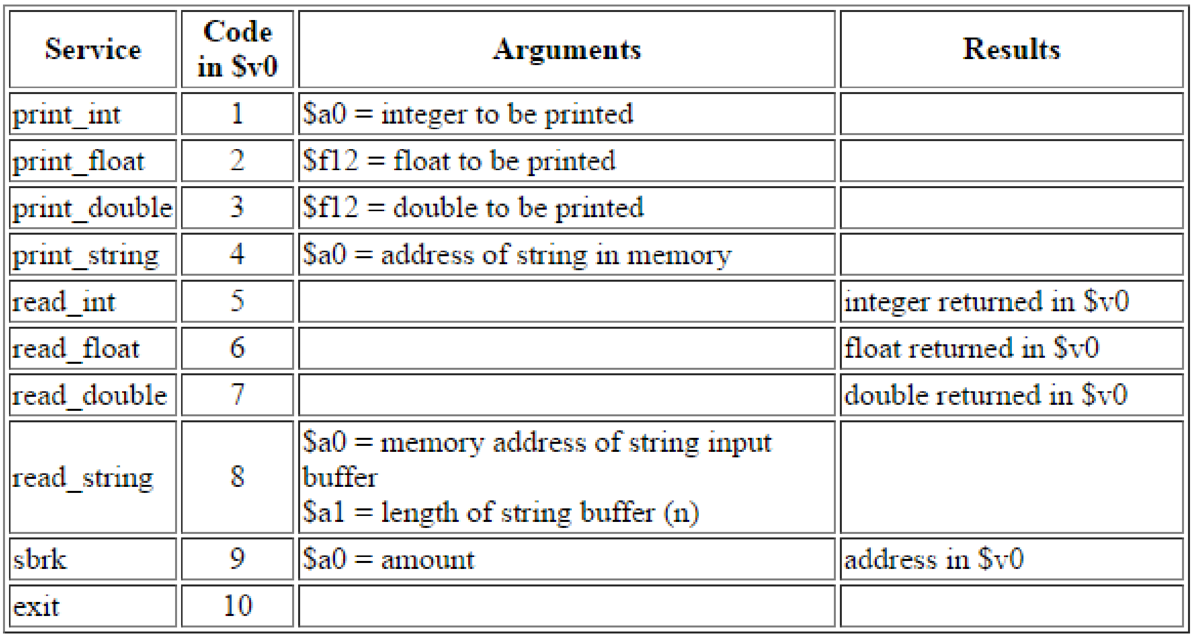
To input a value from the console, we need to make a system call using syscall instruction. For the computer to understand which system call to initiate, we have to provide the number in $v0 register. For inputting an integer, the value of $v0 should be 5. After the number is entered by the user it will be available in $v0. See the following codes for example.

```
.text
.globl main
main:

# to take input an Integer

li $v0, 5                       # $v0 should be loaded with value 5 for taking an integer as input
syscall
move $t0, $v0                       # As  user provides the integer as an input then it is stored in $v0 by default
```

System Calls and I/O (SPIM Simulator)

- Used to read or print values or strings from input/output window, and indicate program end
- Use **syscall** operating system routine call
- First supply appropriate values in registers $v0 and $a0-$a1
- Result value (if any) returned in register $v0

The following table lists the possible **syscall** services.



Service	Code in $v0	Arguments	Results
print_int	1	$a0 = integer to be printed	
print_float	2	$f12 = float to be printed	
print_double	3	$f12 = double to be printed	
print_string	4	$a0 = address of string in memory	
read_int	5		integer returned in $v0
read_float	6		float returned in $v0
read_double	7		double returned in $v0
read_string	8	$a0 = memory address of string input buffer $a1 = length of string buffer (n)	
sbrk	9	$a0 = amount	address in $v0
exit	10		

Assembly code for printing a string:

```
.data
msg1: .asciiz “Hello World”
.text
.globl main
main:
# to print a string
la $a0, msg1     # load the address referred by msg1 in the register a0
li $v0, 4         # v0 should 4 for printing string
syscall
```

Assembly code for taking integer as an input:

```
.text
.globl main
main:
 # to take input an Integer
 li $v0, 5                        # $v0 should be loaded with value 5 for taking an integer as input
syscall
move $t0, $v0                # As  user provides the integer as an input then it is stored in $v0 by default
```

Loading & Storing Byte

```
.data
var1: .byte 'a'
.text
```

```
lw $t0, var1
move $a0, $t0
li $v0, 11
syscall

li $t0, 'b'
sw $t0, var1

lw $t1, var1
move $a0, $t1
li $v0, 11
syscall


li $v0, 10
syscall
```

LAB TASK

(1) Write the MIPS code for the following C code:

```
void main()
    int testInteger;
    printf("Enter an integer: ");
    scanf("%d",&testInteger);
    testInteger =testInteger* testInteger;
    printf("Number = %d",testInteger);
}
```

(2) Write the MIPS code for the following C code:

```
  void main() {
   int length, width, area;

   printf("\nEnter the width of rectangle: ");
   scanf("%d", &width);

   printf("\nEnter the length of rectangle: ");
   scanf("%d", &length);

   area = length * width;
   printf("\nArea of Rectangle : %d ", area);
}
```

(3) Write the MIPS code for the following C code:

```
void main()
{
  int x, y, z, a1, b1, c1;
  printf("\nEnter the value of x : ");
  scanf("%d ", &x);

  printf("\nEnter the value of y : ");
  scanf("%d ", &y);

  printf("\nEnter the value of z : ");
  scanf("%d", &z);

  a1 = x * y + z;
  b1 = x + y * z;
  c1 = x*y-z;

  printf("\nValue of a1 = %d",a1);
  printf("\nValue of b1 = %d",b1);
  printf("\nValue of c1 = %d",c1);
}
```