DHA Suffa University
Department of Computer Science
Computer Organization & Assembly Language
Fall 2020

Lab # 2(Data types & Load/Store Instructions & System

Calls Objective:

To discuss data types and literal, data declarations, load/store instructions and system calls. **MIPS Datatypes for Variables:**

- .ascii
- .asciiz
- .byte
- .halfword
- .word
- .float
- .double

.ASCII:

.ASCII is used to store **string** type data. An ASCII data type don't have a NULL terminator. i.e **'\0'**, called NULL in ASCII).

```
.data

msg1 : .ascii "HELLO ASSEMBLY PROGRAMMERS"
```

.ASCIIZ:

.ASCIIZ is used to store string data but it has a **Z** at end which means it has a null terminator.

```
.data
```

msg1: .asciiz "WELCOME TO ASSEMBLY LANGUAGETUST"

.BYTE:

Byte (8 Bits) data type is used for small integers **without any decimal places**. It can also be used to **store character**.

.data

value1: .byte 736

value2: .byte 'a'

value3: .byte 'b'

value4: .byte 2,3,4,5,5

.WORD AND .HALFWORD:

In .**word**, data is stored in the form of 32 Bits. It's double of half word. **.data**

msg1: .word -20

msg2: .word 30

Array1: .word 1,2,3,4,5,6,6,7,7

A .**halfword** consists of two bytes. It is used to store half of a word. **.data**

msg1: .halfword 'A'

.FLOAT AND .DOUBLE:

Float data type is used to store floating point value with single decimal value. **.data**

float_value: .float 1.1

float_value1: .float 123.5

Double data type is used to store double value with more than one decimal places. **.data**

double_value: .double 1.10

double_value1: .double 123.55

Load/Store Instructions

These instructions are used to access the variables stored in RAM. **lw register_destination, RAM_source**

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

sw register_source, RAM_destination

#store word in source register into RAM destination

MIPS REGISTERS

MIPS assembly language employs a convention for use of registers. This convention is not enforced by the assembler or the hardware, but it must be followed by all MIPS assembly language programmers in order to avoid unexpected behavior of modules written by different people.

Table 5.3. MIPS Registers

5.3. The MIPS Register Files

Register Number	Conventional Name	Usage
$0	$zero	Hard-wired to 0
$1	$at	Reserved for pseudo-instructions
$2 - $3	$v0, $v1	Return values from functions
$4 - $7	$a0 - $a3	Arguments to functions - not preserved by subprograms
$8 - $15	$t0 - $t7	Temporary data, not preserved by subprograms
$16 - $23	$s0 - $s7	Saved registers, preserved by subprograms
$24 - $25	$t8 - $t9	More temporary registers, not preserved by subprograms
$26 - $27	$k0 - $k1	Reserved for kernel. Do not use.
$28	$gp	Global Area Pointer (base of global data segment)
$29	$sp	Stack Pointer
$30	$fp	Frame Pointer
$31	$ra	Return Address
$f0 - $f3	-	Floating point return values
$f4 - $f10	-	Temporary registers, not preserved by subprograms
$f12 - $f14	-	First two arguments to subprograms, not preserved by subprograms
$f16 - $f18	-	More temporary registers, not preserved by subprograms
$f20 - $f31	-	Saved registers, preserved by subprograms

SYSTEM CALLS AND INPUT/OUTPUT

- 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**

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		

SYSTEM CALLS AND INPUT/OUTPUT

LOAD WORDS

.data

marks: .word 123

msg: .asciiz "\nHello\n"

.text

lw $t1,marks

move $a0,$t1

li $v0,1

syscall

li $v0,4

la $a0,msg

syscall

li $v0,10

syscall

STORE WORDS

```
.data

number: .word

.text

li $t0, 1

li $t1, 4

add $t2, $t1, $t0

sw $t2, number

lw $t0, number

move $a0,$t0

li $v0,1

syscall

li $v0, 10

syscall
```

LAB TASK 02

1. Prompt the user to input the marks of Physics, Chemistry, Maths & English out of hundred. Double the marks of each subject and then print new marks on screen. Also show the sum of all the new marks out of 800 and then the old marks out of 400.

LAB ASSIGNMENT 02

1. Create four variables for subjects of your choice, with some initial values out of 100. Calculate the sum and print marks out of 400 in a good format.

2. Draw a similar shape in your mips program.