

Lab 2 : Pseudo Instructions and Memory Access

Name:

Sign the following statement:

On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work

1 Objective

The objective of this lab is to make you more familiar with MIPS pseudo instructions as well as using memory.

2 Pre-requisite

Before starting with this lab, you are required to know what pseudo-instructions are, as well as how MIPS accesses memory.

3 Basic Memory Access

1. Type the following program into MARS (save it as lab2a.s) and use it to answer the questions below

```

1      .data
2      msg1:  .asciiiz "A 17 byte message"
3      msg2:  .asciiiz "Another message of 27 bytes"
4      num1:  .byte 45
5      num2:  .half 654
6      num3:  .word 0xcafebabe
7      num4:  .word 0xfeedface
8      .text
9      .globl main
10     main:
11     li $v0, 4          #system call for print_str
12     la $a0, msg1       #address of string to print
13     syscall
14     la $a0, msg2       #address of string to print
15     syscall
16     lb $t0, num1       #load num1 into $t0
17     lh $t1, num2       #load num2 into $t1
18     lw $t2, num3       #load num3 into $t2
19     lw $t3, num4       #load num4 into $t3
20     li $v0, 10         # system call for exit
21     syscall           # we are out of here.

```

(a) What are the assembly instructions for the pseudo instruction **la \$a0, msg2**?

(b) What is the address for each of the following items?

Object	Address
Data Segment	
Text Segment	
msg1	
msg2	
num1	
num2	
num3	
num4	

(c) Why does msg2 start 18 bytes after msg1 when msg1 is only 17 bytes long?

- (d) Why are there unused bytes between num1, num2, and num3, but num4 start immediately after num3?

2. Go to the *Settings* on menu bar and make sure the item *Popup dialog for input syscalls (5,6,7,8,12)* is checked. Type the following program into MARS (save as lab2b.s):

```

1      .data
2      msg1:    .word 0:24
3      .text
4      .globl main
5      main:
6      li $v0, 8          #syscall for read_str
7      la $a0, msg1       #load address of msg1 to store string
8      li $a1, 100        #msg1 is 100 bytes
9      syscall
10     lb $t0, 5($a0)     #load the character into $t0
11     li $t1, 'a'        #get value of 'a'
12     blt $t0, $t1, nomodify #do nothing if letter is less than 'a'
13     li $t1, 'z'        #get value of 'z'
14     bgt $t0, $t1, nomodify #do nothing if letter is greater than 'z'
15     addi $t0, $t0, -0x20 #encap the letter
16     sb $t0, 5($a0)     #store the new letter
17     nomodify:
18     li $v0, 4          #syscall for print_str
19     syscall
20     li $v0, 10         # system call for exit
21     syscall           # we are out of here.
```

Specify the operation performed by the program:

3. Write a program that reads a string from the user and outputs the number of lower-case letters in the string. Save your program as lab2c.s, and run it to make sure it works correctly. Demonstrate your progress to the TA. .
4. Type the following program into MARS (save your program as lab2d.s):

```

1      .data
2      hextable:      .ascii "0123456789abcdef"
3      msg1:          .asciiz "Your number in Hex is: "
4      .text
5      .globl main
6      main:
7          li $v0, 5          #syscall for read_int
8          syscall
9          add $s1, $v0, $0
10         li $v0, 4          #syscall for print_str
11         la $a0, msg1
12         syscall
13         la $a1, hextable
14         srl $t0, $s1, 4     #get upper 4 bits
15         add $a2, $a1, $t0   #get address in hextable
16         lb $a0, 0($a2)     #get character
17         li $v0, 11         #syscall for print_char
18         syscall
19         andi $t0, $s1, 0xf   #get lower 4 bits
20         add $a2, $a1, $t0   #get address in hextable
21         lb $a0, 0($a2)     #get character
22         li $v0, 11         #syscall for print_str
23         syscall
24         li $v0, 10         # system call for exit
25         syscall            # we are out of here.

```

Specify the operation performed by the program:

5. Write a program that reads a number **x** from the user, and prints the first **x** letters of the alphabet (in lower case). You do not need to check whether the number is positive.

Save your program as lab2f.s, and run it to make sure it works correctly. Demonstrate your progress to the TA. .

4 Deliverables

- Submit completed copy of this lab manual.
- Include the following in a compressed file (.zip format) to your TA:
 - The source code for all .s files.
 - All log files.