Computer Architecture | Course: IT089IU

International University – VNU HCM | Date: March 2024

Dr. Le Hai Duong (revised by Nga Ly) | Time: 3 hours

# Laboratory Session 2

# Testing and Branching

## 1. MIPS assembler directives

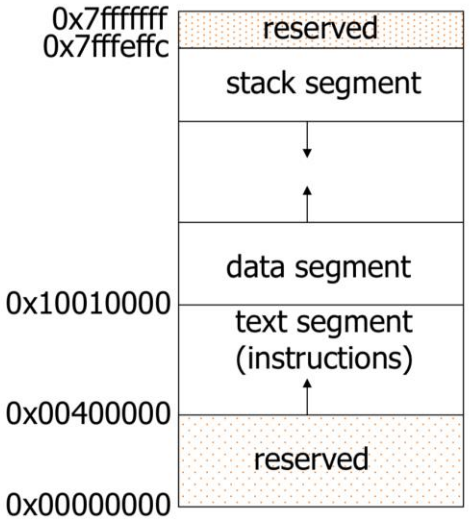


Figure 1 MIPS Memory Usage as viewed in SPIM

**.text**

indicates that following items are stored in the user text segment, typically instructions

**.data**

indicates that following data items are stored in the data segment

**.globl** sym

declare that symbol sym is global and can be referenced from other files

**Common data definitions**

**.word** w1, ..., wn

store n 32-bit quantities in successive memory words

**.half** h1, ..., hn

store n 16-bit quantities in successive memory halfword

**.byte** b1, ..., bn

store n 8-bit quantities in successive memory bytes

**.ascii** str

store the string in memory but do not null-terminate it

- strings are represented in double-quotes “str”
- special characters, eg. \n, \t, follow C convention

**.asciiz** str

store the string in memory and null-terminate it

**.float** f1, ..., fn

store n floating point single precision numbers in successive memory locations

**.double** d1, ..., dn

store n floating point double precision numbers in successive memory locations

.**space** n

reserves n successive bytes of space

.**align** n

align the next datum on a $2^n$ byte boundary.

For example, **.align 2** aligns next value on a word boundary.

.**align 0** turns off automatic alignment of **.half**, **.word**, etc. till next **.data** directive

## 2. Pseudo-instructions (20pts)

Pseudo-instructions do not correspond to real MIPS instructions. Assembler would translate pseudo-instructions to real instructions (one or more instructions). Pseudo-instructions not only make it easier to program, it can also add clarity to the program, by making the intention of the programmer clearer.

Change the pseudo-instruction “**li $t0, 5**” in **Lab2_2.s** to “**li $t0, -5**”. What are the real MIPS instructions for “**li $t0, -5**”. Explain how the real instructions work.

Change the pseudo-instruction “**li $t0, 5**” in **Lab2_2.s** to “**li $t0, 0xaabbccdd**”. What are the real instructions for “**li $t0, 0xaabbccdd**”. Explain how the real instructions work.

## 3. Branching (20pts)

3.1 Load the assembly file **Lab2_3.s** into qtSpim and run. Try to win the game. What is the **secret number**?

3.2 Result (Source Code): Why win and lose in the same time? -> BNE = Branch if NOT EQUAL

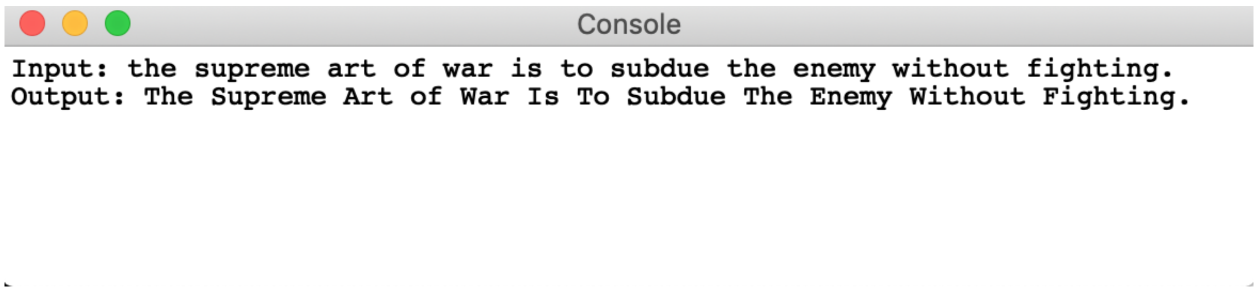
3.3 Modify the game so that it will print out as follow (no iteration) using the instructions **bgt,** or **bge,** or **blt,** or **ble**: Save your file as **Lab2_3.3.s**

3.4 Modify the game so that player can keep guessing until he finds the secret number. Save your assembly as **Lab2_3.4.s**

3.5 Modify previous version so that player can decide to stop the game by input a **flag**. Save your assembly as **Lab2_3.5.s**

**4. String (20pts)**

Write an assembly that convert an input string as follow:



```
Input: the supreme art of war is to subdue the enemy without fighting.
Output: The Supreme Art of War Is To Subdue The Enemy Without Fighting.
```

The first letter of every word is capitalized. Save your assembly as **Lab2_4.s**

**5. (10pts)**

Write a MIPS program to print out the result of F and G:

$$F = \frac{(a+b)\times(c-d)}{a^2}$$

$$G = \frac{(a+1)\times(b+2)\times(c-3)}{c-a}$$

**6. (30pts)**

a. Write a MIPS program to print a sequence of numbers “N, N*M, N*M*M, N*M*M*M, ...” X times, where N, M, X are specified by the user.

b. Write a MIPS program to print out the decimal value of a 10-bit binary number.

**Reference:**

1. https://en.wikibooks.org/wiki/MIPS_Assembly/Pseudoinstructions
2. https://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html

3. https://www.assemblylanguagetuts.com/mips-assembly-programming-tutorials/#MIPS_Data_Types
4. https://en.wikibooks.org/wiki/MIPS_Assembly/Arithmetic_Instructions
5. https://gab.wallawalla.edu/~curt.nelson/cptr280/lecture/mips%20arithmetic%20instructions.pdf