REPORT LAB 2

ITITIU22240 – ĐÀM NGUYỄN TRỌNG LỄ

**1. MIPS assembler directives**

**2. Pseudo-instructions (20pts)**

2.1 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.

li      $t0,     -5 => pseudo code : -5 = 0xFFFFFFFB

lui $t), 0xFFFF ( Load Upper Immediate the super half of $at )

$at = 0xFFFF0000

ori $t0, $at, 0xFFFB ( OR Immediate with the lower half of $at)

$at = 0xFFFF0000

OR 0x0000FFFB

* $at = 0xFFFFFFFB

Move $at to $8

=> $t0 = $8

lui $8, 0xFFFF | super half => -1

ori $8, $at, 0xFFFB | lower half => -5

then we OR immediate to link two halves to get the 32-bit value

So the li consist of 2 line in MIPS

[00400024] 3c01ffff  lui $1, -1               ; 11: li $t0, -5

[00400028] 3428fffb  ori $8, $1, -5

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

2.2 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.

The 32-bit immediate value 0xaabbccdd is split into two 16-bit parts

Upper 16 bits: 0xaabb

Lower 16 bits: 0xccdd

li $t0, 0xaabbccdd => pseudo code : 0xaabbccdd

lui $at, 0xaabb

ori $t0, $at, 0xccdd

=> $t0 = $8

lui $at, 0xaabb | super half => 0xaabb

ori $8, $at, 0xccdd | lower half => 0xccdd

then we OR immediate to link two halves to get the 32-bit value

**3. Branching (40pts)**

3.1 Load the assembly file Lab2\_3.s into qtSpim and run. Try to win the game. What is the secret number?

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

the secret number is on the line

[00400024] 340801fa  ori $8, $0, 506          ; 11: li $t0, 0x1fa

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

Nội dung do AI tạo ra có thể không chính xác.

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

Even we guess the correct nums we still have the lose line, the reason is the after we go into the WIN branch there no jump to the end but we continue to go into LOSE branch thus print the "You lose!!"

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

Implement 2 branch approach:

1 for win, 1 for lose , with two auto return to start

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, phần mềm, số, Trang web

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

Apply while loop with the exit condition is user get right number (506)

Use branch if equal (beq)

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

Ảnh có chứa ảnh chụp màn hình, văn bản, máy tính, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Implememt FLAG feature after print stop\_promt user can input 1111 => stop the game and jump to exit branch

Orther than that stay in the while loop until guessing the correct number

**4. String (20pts)**

*.data*                                                           # the data segment

prompt\_input:       *.asciiz* "Input a string (limit 100 char): "

prompt\_output:      *.asciiz* "Output: "

newline:            *.asciiz* "\n"

is\_space:           *.word*   1

user\_input:         *.space*  100

user\_input\_length:  *.word*   100

*.text*                                                           # the code segment

                    .globl  main

main:

    # Print prompt for input

    li      $v0,            4

    la      $a0,            prompt\_input

    syscall

    # Read user input string

    li      $v0,            8

    la      $a0,            user\_input

    lw      $a1,            user\_input\_length

    syscall

    # Initialize registers

    la      $t0,            user\_input                          # $t0 = address of the string

    li      $t1,            1                                   # $t1 = is\_space flag (start with 1 to capitalize first letter)

process\_string:

    lb      $t2,            0($t0)                              # Load current character 1 byte = 255 > 'z' which is 122

    beqz    $t2,            finish                              # If null terminator character, finish

    # Check if character is a space

    li      $t3,            32                                  # ASCII for space

    beq     $t2,            $t3,                found\_space

    li      $t3,            9                                   # ASCII for tab

    beq     $t2,            $t3,                found\_space

    li      $t3,            10                                  # ASCII for newline

    beq     $t2,            $t3,                found\_space

    # If not a space, check if we need to capitalize

    beqz    $t1,            not\_first\_letter

    # Check if lowercase letter (97-122 in ASCII)

    slti    $t4,            $t2,                123             # $t4 = 1 if character < 123

    slti    $t5,            $t2,                97              # $t5 = 1 if character < 97

    sub     $t4,            $t4,                $t5             # $t4 = 1 if 97 <= character < 123

    beqz    $t4,            not\_lowercase                       # If not lowercase, skip

    # Convert lowercase to uppercase (subtract 32)

    sub     $t2,            $t2,                32

    sb      $t2,            0($t0)                              # Store back the capitalized letter

not\_lowercase:

    li      $t1,            0                                   # Reset is\_space flag

    j       continue

found\_space:

    li      $t1,            1                                   # Set is\_space flag

    j       continue

not\_first\_letter:

    # Not a first letter, do nothing

continue:

    addi    $t0,            $t0,                1               # Move to next character

    j       process\_string

finish:

    # Print output prompt

    li      $v0,            4

    la      $a0,            prompt\_output

    syscall

    # Print modified string

    li      $v0,            4

    la      $a0,            user\_input

    syscall

    # Print newline

    li      $v0,            4

    la      $a0,            newline

    syscall

exit:

    # terminate the program

    li      $v0,            10

    syscall

the key idea is set up FLAG feature (is\_space before the letter, default is 1 for the first letter in the string is auto uppercase) and treat the string of user text input like an array

the space have 3 case: normal case: 1 blank space ( ascii = 32)

tab, \n is some edge case ( ascii = 9, 10)

we compare the value then we decide to branching to or not into the found\_space branch   
the found\_space branch is have oly one perpose is set the FLAG on ( is\_space = 1)

when the FLAG is on we check the first letter we catch whether it is legit or not (‘a’ to ‘z’) using the ascii code value (97 to 122) then sub by 32 to get the ascii code for the uppercase, repeat the process with the $t0 is the iterator

then print the array affter modified to a window.

Also we assume that the value of 1 character that user input is base on the ascii list => 1byte only => load byte

Ảnh có chứa ảnh chụp màn hình, văn bản, máy tính, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, màn hình

Nội dung do AI tạo ra có thể không chính xác. Ảnh có chứa văn bản, màn hình, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.