Báo cáo thực hành môn Kiến trúc máy tính

Họ tên: Nguyễn Thanh Hưng

MSSV: 20225633

1. Assignment 1

Code:

#Laboratory Exercise 7 Home Assignment 1

.text

main:    li      $a0,    -3        #load input parameter

    jal     abs                     #jump and link to abs procedure

    nop

    add     $s0,    $zero,  $v0

    li      $v0,    10              #terminate

    syscall

endmain:

abs:

    sub     $v0,    $zero,  $a0     #put -(a0) in v0; in case (a0)<0

    bltz    $a0,    done            #if (a0)<0 then done

    nop

    add     $v0,    $a0,    $zero   #else put (a0) in v0

done:

    jr      $ra

Giải thích code:

* **Li** $a0, -3: load giá trị -3 vào thanh ghi $a0



* **Jal** abs: lệnh jal thực hiện việc nhảy đến chương trình con abs và sẽ lưu địa chỉ của việc tiếp theo (sau lệnh **Jal**) vào thanh ghi **$ra**, thanh ghi **$ra** được sử dụng để lưu trữ địa chỉ trả về sau khi thực hiện lệnh gọi hàm.



Lúc này ở thanh ghi **$pc** sẽ tăng lên 1 giá trị bằng với **địa chỉ tiếp theo** của lệnh **Jal**.



Giá trị tăng của **pc** được tính bằng cách:

Giá trị tăng **PC** = địa chỉ **‘abc’** – địa chỉ **‘jal’** + 4



* Sau đó thực hiện chương trình con tìm **Abs** của giá trị lưu ở thanh ghi $a0.
* Kết thúc chương trình con **Jr $ra** nhảy về trở về câu lệnh sau câu lệnh **Jal** ở trên và thực hiện những câu lệnh còn lại của chương trình.



Thanh ghi **$pc** lúc này cũng sẽ nhảy trở về bằng với giá trị của thanh ghi **$ra** và thực hiện bước nhảy cho các lệnh còn lại của chương trình



1. Assignment 2

Code:

    #Laboratory Exercise 7, Home Assignment 2

.text

main:

    li      $a0,    3               #load test input

    li      $a1,    5

    li      $a2,    7

    jal     max                     #call max procedure

    nop

endmain:

max:

    add     $v0,    $a0,    $zero   #copy (a0) in v0; largest so far

    sub     $t0,    $a1,    $v0     #compute (a1)-(v0)

    bltz    $t0,    okay            #if (a1)-(v0)<0 then no change

    nop

    add     $v0,    $a1,    $zero   #else (a1) is largest thus far

okay:

    sub     $t0,    $a2,    $v0     #compute (a2)-(v0)

    bltz    $t0,    done            #if (a2)-(v0)<0 then no change

    nop

    add     $v0,    $a2,    $zero   #else (a2) is largest overall

done:

    jr      $ra                     #return to calling program

Giải thích code:

* Ở đây sự hoạt động của **$ra** và **$pc** sẽ giống với ở bài trên.
* **Jal max**:

Thanh ghi **$ra** sẽ lưu trữ địa chỉ trả về sau khi thực hiện lệnh gọi hàm



Thanh ghi **$pc** cũng sẽ tăng thêm 1 giá trị bằng với **địa chỉ tiếp theo** của lệnh **Jal**.



Giá trị tăng của **pc** được tính bằng cách:

Giá trị tăng **PC** = địa chỉ **‘max’** – địa chỉ **‘jal’** + 4



* Kết thúc chương trình con **Jr $ra** nhảy về trở về câu lệnh sau câu lệnh **Jal** ở trên. Thanh ghi **$pc** lúc này cũng sẽ nhảy trở về bằng với giá trị của thanh ghi **$ra** và thực hiện bước nhảy cho các lệnh còn lại của chương trình



1. Assignment 3

Code:

    #Laboratory Exercise 7, Assignment 3

.text

    li      $s0,    1

    li      $s1,    2

push:    addi    $sp,    $sp,    -8 #adjust the stack pointer

    sw      $s0,    4($sp)          #push $s0 to stack

    sw      $s1,    0($sp)          #push $s1 to stack

work:    nop

    nop

    nop

pop:    lw      $s0,    0($sp)      #pop from stack to $s0

    lw      $s1,    4($sp)          #pop from stack to $s1

    addi    $sp,    $sp,    8       #adjust the stack pointer

* Kết quả chương trình:

A screenshot of a computer

Description automatically generated

* Load giá trị vào Stack:

A screenshot of a computer

Description automatically generated

1. Assignment 4

Code:

        #Laboratory Exercise 7, Home Assignment 4

    .data

    Message:    .asciiz "Ket qua tinh giai thua la: "

    .text

    main:           jal     WARP

    print:          add     $a1,    $v0,        $zero   # $a0 = result from N!

        li      $v0,    56

        la      $a0,    Message

        syscall

    quit:           li      $v0,    10                  #terminate

        syscall

    endmain:

        #---------------------------------------------------------------------

        #Procedure WARP: assign value and call FACT

        #---------------------------------------------------------------------

    WARP:           sw      $fp,    -4($sp)             #save frame pointer (1)

        addi    $fp,    $sp,        0                   #new frame pointer point to the top (2)

        addi    $sp,    $sp,        -8                  #adjust stack pointer (3)

        sw      $ra,    0($sp)                          #save return address (4)

        li      $a0,    3                               #load test input N

        jal     FACT                                    #call fact procedure

        nop

        lw      $ra,    0($sp)                          #restore return address (5)

        addi    $sp,    $fp,        0                   #return stack pointer (6)

        lw      $fp,    -4($sp)                         #return frame pointer (7)

        jr      $ra

    wrap\_end:

        #---------------------------------------------------------------------

        #Procedure FACT: compute N!

        #param[in] $a0 integer N

        #return $v0 the largest value

        #---------------------------------------------------------------------

    FACT:           sw      $fp,    -4($sp)             #save frame pointer

        addi    $fp,    $sp,        0                   #new frame pointer point to stack?s top

        addi    $sp,    $sp,        -12                 #allocate space for $fp,$ra,$a0 in stack

        sw      $ra,    4($sp)                          #save return address

        sw      $a0,    0($sp)                          #save $a0 register

        slti    $t0,    $a0,        2                   #if input argument N < 2

        beq     $t0,    $zero,      recursive           #if it is false ((a0 = N) >=2)

        nop

        li      $v0,    1                               #return the result N!=1

        j       done

        nop

    recursive:

        addi    $a0,    $a0,        -1                  #adjust input argument

        jal     FACT                                    #recursive call

        nop

        lw      $v1,    0($sp)                          #load a0

        mult    $v1,    $v0                             #compute the result

        mflo    $v0

    done:           lw      $ra,    4($sp)              #restore return address

        lw      $a0,    0($sp)                          #restore a0

        addi    $sp,    $fp,        0                   #restore stack pointer

        lw      $fp,    -4($sp)                         #restore frame pointer

        jr      $ra                                     #jump to calling

    fact\_end:

Kết quả chương trình:

A screenshot of a computer

Description automatically generated

Stack:

A screenshot of a computer

Description automatically generated

Với n = 3:

A table with numbers and symbols

Description automatically generated

1. Assignment 5

Code:

find\_largest\_smallest:

    # Save registers on the stack

    addi    $sp,    $sp,                -16

    sw      $s0,    0($sp)

    sw      $s1,    4($sp)

    sw      $s2,    8($sp)

    sw      $ra,    12($sp)

    # Initialize largest and smallest variables

    move    $s3,    $s0                         # Largest element

    move    $s4,    $zero                       # Position of largest element

    move    $s6,    $s0                         # Smallest element

    move    $s7,    $zero                       # Position of smallest element

    # Loop through the list

    li      $t0,    1                           # Counter

    li      $t1,    8                           # Number of elements in the list

loop:

    # Compare current element with largest element

    slt     $t2,    $s0,                $s3

    beqz    $t2,    check\_smallest

    # Update largest element and position

    move    $s3,    $s0

    move    $s4,    $t0

check\_smallest:

    # Compare current element with smallest element

    slt     $t2,    $s6,                $s0

    beqz    $t2,    increment\_counter

    # Update smallest element and position

    move    $s6,    $s0

    move    $s7,    $t0

increment\_counter:

    addi    $t0,    $t0,                1

    addi    $s0,    $s0,                4

    # Check if all elements have been processed

    bne     $t0,    $t1,                loop

    # Restore registers from the stack

    lw      $s0,    0($sp)

    lw      $s1,    4($sp)

    lw      $s2,    8($sp)

    lw      $ra,    12($sp)

    addi    $sp,    $sp,                16

    # Return from the procedure

    jr      $ra