**Thực hành kiến trúc máy tính tuần 31**

**Full name: Lê Quang Khải**

**Student ID: 20225638**

Assignment 1

.text

main: li $a0, 10 #load input parameter

      jal abs #jump and link to abs procedure

      nop

      add $s0, $zero, $v0

      li $v0, 10 #terminate

      syscall

endmain:

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

# function abs

# param[in] $a0 the interger need to be gained the absolute value

# return $v0 absolute value

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

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | $a0 | $s0 | $v0 | $ra | $pc |
| li $a0,-45 | 0xffffffd3 | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400004 |
| jal abs nop | 0xffffffd3 | 0x00000000 | 0x00000000 | 0x00400008 | 0x00400018 |
| sub $v0,$0,$a0 | 0xffffffd3 | 0x00000000 | 0x0000002d | 0x00400008 | 0x0040001c |
| bltz $a0,done | 0xffffffd3 | 0x00000000 | 0x0000002d | 0x00400008 | 0x00400028 |
| jr $ra | 0xffffffd3 | 0x00000000 | 0x0000002d | 0x00400008 | 0x00400008 |
| nop | 0xffffffd3 | 0x00000000 | 0x0000002d | 0x00400008 | 0x0040000c |
| add $s0,$zero,$v0 | 0xffffffd3 | 0x00000000 | 0x0000002d | 0x00400008 | 0x00400010 |

TH1: Truyền tham số a0 =-45 , v0 = 45 vì a0 < 0 nên nhảy xuống câu lệnh done luôn và nhảy lại lên vị trí nop. Gán $s0 = $v0 =45 và in ra s0 là giá trị tuyệt đối của tham số a0.

Chú ý: khi thực hiện lệnh jal thì $ra lưu giá trị địa chỉ tại câu lệnh ngay sau nó là lệnh nop

TH2: Truyền tham số a0 =10 , v0 =-10 Vì a0 = 10 > 0 nên ko nhảy xuống done mà gán v0 = a0 = 10 sau đó dùng lệnh jr nhảy đến địa chỉ được lưu ở $ra . Sau đó gán s0 = v0 và kết thúc chương trình.

Assignment 2

.text

main: li $a0, 5

      li $a1, -10

      li $a2, 16

      jal max

      nop

      li $v0, 10

      syscall

end\_main:

max:

      add $s0, $a0, $zero # gán a0 là max

      sub $t0, $a1, $s0 # t0 = a1 - s0

      bltz $t0, then # if a1 < s0 thì nhảy tới then

      nop

      add $s0, $a1, $zero # gán a1 làm max mới

then:

      sub $t0, $a2, $s0 # t0 = a2 - s0

      bltz $t0, done

      nop

      add $s0, $a2, $zero

done: jr $ra

Địa chỉ của các label: A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Sau câu lệnh jal max thanh ghi pc trỏ đến địa chỉ của câu lệnh add $s0, $a0, $zero để bắt đầu thực thi khối max và thanh ghi $ra lưu lại giá trị thanh ghi nop sau sau câu lệnh jal. Sau khi chạy xong hàm max nhảy xuống done thực hiện lệnh jr $ra làm cho $pc được gán bằng $ra và quay lại nop ở đoạn trên để in ra max ra màn hình.

Assignment 3

.text

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

      li $s0, 2

      li $s1, -3

      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

Giá trị $sp ban đầu mặc định:



Cần đẩy 2 số $s0, $s1 vào stack nên gán $sp = $sp - 8

Lệnh push: Gán giá trị s0 cho thanh ghi tại địa chỉ sp+4 và giá trị s1 cho thanh ghi tại địa chỉ sp, tức là đẩy $s0 = 2 vào trước $s1 = -3 vào sau.

Lệnh pop: Gán giá trị tại thanh ghi 0($sp) vào thanh ghi $s0 tức là lấy phần tử ở đỉnh stack là -3 gán vào $s0, sau đó gán giá trị còn lại là 2 trong stack vào thanh ghi $s1. Như vậy dùng stack ta đã hoán đổi được giá trị của 2 thanh ghi $s0 và $s1. Sau khi kết thúc ta lấy sp +8 để trả lại giá trị ban đầu cho thanh ghi sp

Kết quả:

A white rectangular object with blue lines

Description automatically generated

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:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $pc | $ra | $sp | $fp |
| jal WARP | 0x00400020 | 0x00400004 | 0x7fffeffc | 0x00000000 |
| sw $fp, -4($sp) | 0x00400024 | 0x00400004 | 0x7fffeffc | 0x00000000 |
| addi $fp, $sp, 0 | 0x00400028 | 0x00400004 | 0x7fffeffc | 0x7fffeffc |
| addi $sp, $sp, -8 | 0x0040002c | 0x00400004 | 0x7fffeff4 | 0x7fffeffc |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| $ra = 0x00400004 |  |
| $fp = 0x7fffeffc |  |
|  |  |

- FACT với $a0 = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $pc | $ra | $sp | $fp |
| jal FACT | 0x0040004c | 0x00400038 | 0x7fffeff4 | 0x7fffeffc |
| sw $fp, -4($sp) | 0x00400050 | 0x00400038 | 0x7fffeff4 | 0x7fffeffc |
| addi $fp, $sp, 0 | 0x00400054 | 0x00400038 | 0x7fffeff4 | 0x7fffeff4 |
| addi $sp, $sp, -12 | 0x00400058 | 0x00400038 | 0x7fffefe8 | 0x7fffeff4 |
| sw $ra, 4($sp) | 0x0040005c | 0x00400038 | 0x7fffefe8 | 0x7fffeff4 |
| sw $a0, ($sp) | 0x00400060 | 0x00400038 | 0x7fffefe8 | 0x7fffeff4 |
| slti $t0, $a0, 2 | 0x00400064 | 0x00400038 | 0x7fffefe8 | 0x7fffeff4 |
| beq $t0, $zero, recursive | 0x00400078 | 0x00400038 | 0x7fffefe8 | 0x7fffeff4 |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| $a0 = 3 |  |
| $ra = 0x00400038 |  |
| $fp = 0x7fffeff4 |  |
| $ra = 0x00400004 |  |
| $fp = 0x7fffeffc |  |
|  |  |

- FACT với $a0 = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $pc | $ra | $sp | $fp |
| addi $a0, $a0, -1 | 0x0040007c | 0x00400038 | 0x7fffefe8 | 0x7fffeff4 |
| jal FACT | 0x0040004c | 0x00400080 | 0x7fffefe8 | 0x7fffeff4 |
| sw $fp, -4($sp) | 0x00400050 | 0x00400080 | 0x7fffefe8 | 0x7fffefe8 |
| addi $fp, $sp, 0 | 0x00400054 | 0x00400080 | 0x7fffefe8 | 0x7fffefe8 |
| addi $sp, $sp, -12 | 0x00400058 | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |
| sw $ra, 4($sp) | 0x0040005c | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |
| sw $a0, ($sp) | 0x00400060 | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |
| slti $t0, $a0, 2 | 0x00400064 | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |
| beq $t0, $zero, recursive | 0x00400078 | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |

|  |  |
| --- | --- |
|  |  |
| $a0 = 2 |  |
| $ra = 0x00400080 |  |
| $fp = 0x7fffeff8 |  |
| $a0 = 3 |  |
| $ra = 0x00400038 |  |
| $fp = 0x7fffeff4 |  |
| $ra = 0x00400004 |  |
| $fp = 0x7fffeffc |  |
|  |  |

- FACT với $a0 = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $pc | $ra | $sp | $fp |
| addi $a0, $a0, -1 | 0x0040007c | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |
| jal FACT | 0x0040004c | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |
| sw $fp, -4($sp) | 0x00400050 | 0x00400080 | 0x7fffefdc | 0x7fffefe8 |
| addi $fp, $sp, 0 | 0x00400054 | 0x00400080 | 0x7fffefdc | 0x7fffefdc |
| addi $sp, $sp, -12 | 0x00400058 | 0x00400080 | 0x7fffefd0 | 0x7fffefdc |
| sw $ra, 4($sp) | 0x0040005c | 0x00400080 | 0x7fffefd0 | 0x7fffefdc |
| sw $a0, ($sp) | 0x00400060 | 0x00400080 | 0x7fffefd0 | 0x7fffefdc |
| slti $t0, $a0, 2 | 0x00400064 | 0x00400080 | 0x7fffefd0 | 0x7fffefdc |
| beq $t0, $zero, recursive | 0x00400078 | 0x00400080 | 0x7fffefd0 | 0x7fffefdc |

|  |  |
| --- | --- |
| $a0 = 1 |  |
| $ra = 0x00400080 |  |
| $fp = 0x7fffefdc |  |
| $a0 = 2 |  |
| $ra = 0x00400080 |  |
| $fp = 0x7fffeff8 |  |
| $a0 = 3 |  |
| $ra = 0x00400038 |  |
| $fp = 0x7fffeff4 |  |
| $ra = 0x00400004 |  |
| $fp = 0x7fffeffc |  |
|  |  |

A screenshot of a computer

Description automatically generatedKết quả:

Assignment 5

.data

Messenger1a: .asciiz "The largest element is stored in $s"

Messenger1b: .asciiz " ,largest value is "

Messenger2a: .asciiz "The smallest element is stored in $s"

Messenger2b: .asciiz " ,smallest value is "

newline: .asciiz "\n"

.text

main:

    li $s0, 4

    li $s1, 3

    li $s2, 99

    li $s3, 3

    li $s4, 54

    li $s5, -5

    li $s6, 5

    li $s7, 123

    jal find\_maxmin

    nop

quit:

    li $v0, 10 #terminate

    syscall

endmain:

find\_maxmin:

    add $fp, $sp, $0 # fp points to the bottom of the stack

    addi $sp, $sp, -48 # 8 thanh ghi từ $s0-$s7 và 4 giá trị max và min địa chỉ của max và min

    sw $s0, 44($sp)

    sw $s1, 40($sp)

    sw $s2, 36($sp)

    sw $s3, 32($sp)

    sw $s4, 28($sp)

    sw $s5, 24($sp)

    sw $s6, 20($sp)

    sw $s7, 16($sp) # push 8 thanh ghi vào stack

    li $t0, 0 # gán $t0 = i = 0

    li $t1, 7 # n = 7

    lw $a0, 44($sp) # $a0 = $s0 (= A[0])

    sw $a0, 12($sp) # push max = A[0]

    sw $t0, 8($sp) # push maxid = 0

    sw $a0, 4($sp) # push min = A[0]

    sw $t0, 0($sp) # push minid = 0

    addi $t0, $t0, 1 # chạy từ i = 1

loop:

    bgt $t0, $t1, print

    addi $t2, $t0 , 1 # gán $t2 = i + 1

    add $t2, $t2, $t2 # $t2 = 2\*i + 2

    add $t2, $t2, $t2 # $t2 = 4\*i + 4

    sub $t2, $fp, $t2 # $t2 = address(A[i])

    lw $a0, 0($t2) # $a0 = A[i]

    lw $a1, 12($sp) # $a1 = max\_hientai

    lw $a2, 4($sp) # $a2 = min\_hientai

max:

    ble $a0, $a1, min # $a0 <= $a1 ? (A[i] < max) nếu đúng thì tìm min, nếu sai thì gán max = A[i]

    sw $a0, 12($sp) # max = A[i]

    sw $t0, 8($sp) # maxid = i

min:

    bge $a0, $a2, then # $a0 >= $a2 ? (A[i] > min) nếu đúng thì nhảy tới then, nếu sai thì gán min = A[i]

    sw $a0, 4($sp) # min = A[i]

    sw $t0, 0($sp) # minid = i

then:

    addi $t0, $t0, 1 # i++

    j loop

print:

    li $v0, 4

    la $a0, Messenger1a

    syscall

    li $v0, 1

    lw $a0, 8($sp) # $a0 = maxid

    syscall

    li $v0, 4

    la $a0, Messenger1b

    syscall

    li $v0, 1

    lw $a0, 12($sp) # $a0 = max

    syscall

    li $v0, 4

    la $a0, newline

    syscall

    li $v0, 4

    la $a0, Messenger2a

    syscall

    li $v0, 1

    lw $a0, 0($sp) # $a0 = min\_id

    syscall

    li $v0, 4

    la $a0, Messenger2b

    syscall

    li $v0, 1

    lw $a0, 4($sp) # $a0 = min

    syscall

done:

    addi $sp, $sp, 48

    jr $ra

Kết quả:

A black text on a white background

Description automatically generated