Thực hành Kiến trúc máy tính

Week 7

Vũ Ngọc Đức – 20225816

Assignment 1

.text

main:

li $a0, -45 #gán giá trị a0 = -45

jal abs #jump tới nhãn abs và gán giá trị của địa chỉ vào $ra

nop

add $s0, $zero, $v0 #s0 = v0

li $v0, 10 #exit

syscall

abs:

sub $v0,$zero,$a0 #v0 = -a0

bltz $a0,done #nếu (a0 < 0) => done

nop

add $v0,$a0,$zero #else v0 = a0

done:

jr $ra #nhảy tới địa chỉ $ra

=> Kết quả lưu tại s0 = 45

Assignment 2

.text

main:

li $a0,2 #gán giá trị a0 = 2

li $a1,6 #gán giá trị a1 = 6

li $a2,9 #gán giá trị a1 = 9

jal max #jump tới nhãn max và gán giá trị địa chỉ vào $ra

nop

li $v0, 10 #exit

syscall

max:

add $v0,$a0,$zero #v0 = a0. Cập nhật max = a0

sub $t0,$a1,$v0 #t0 = a1 – v0

bltz $t0,okay #if (t0 < 0) => okay

nop

add $v0,$a1,$zero #else v0 = a1. Cập nhật max = a1

okay:

sub $t0,$a2,$v0 #t0 = a2 – v0

bltz $t0,done #if (t0 < 0) => done

nop

add $v0,$a2,$zero #else v0 = a2. Cập nhật max = a2

done:

jr $ra #nhảy tới địa chỉ $ra

=> Kết quả là giá trị max của 3 biến lưu tại v0 trước khi chương trình exit

Assignment 3

.text

.main

li $s0, 5 #gán giá trị s0 = 5

li $s1, 7 #gán giá trị s1 = 7

push:

addi $sp,$sp,-8 #điều chỉnh con trỏ stack

sw $s0,4($sp) #cho giá trị s0 vào stack

sw $s1,0($sp) #cho giá trị s1 vào stack

work:

nop

nop

nop

pop:

lw $s0,0($sp) #lấy giá trị ra khỏi stack vào s0

lw $s1,4($sp) #lấy giá trị ra khỏi stack vào s1

addi $sp,$sp,8 #điều chỉnh con trỏ stack

=> stack có cơ chế LIFO nền giá trị của s0 và s1 hoán đổi nhau

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

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 $a0,6 #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

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:

Đây là chương trình tính giai thừa bằng phương pháp đệ quy.

Ở phần .data, biến Message được khởi tạo là một chuỗi ký tự "Ket qua tinh giai thua la: ".

Ở phần .text, địa chỉ của chương trình bắt đầu từ main. Tại đây, hàm WARP được gọi bằng lệnh jal. Sau đó, kết quả tính toán được trả về vào thanh ghi $v0 và được lưu trữ trong thanh ghi $a1 bằng lệnh add.

Hàm print được gọi và in kết quả ra màn hình bằng cách sử dụng lệnh syscall và thanh ghi $v0 có giá trị là 56.

Trong hàm WARP, các giá trị của các thanh ghi được lưu trữ trong stack. Frame pointer (fp) là con trỏ trỏ đến vị trí đầu tiên của frame hiện tại. Lệnh sw $fp,-4($sp) được sử dụng để lưu giá trị của fp. Sau đó, fp được thay đổi để trỏ đến đỉnh stack mới được cấp phát, và stack pointer (sp) được điều chỉnh để trỏ đến đỉnh stack mới. Giá trị của return address (ra) được lưu trữ trên stack bằng lệnh sw $ra,0($sp).

Hàm FACT được gọi bằng lệnh jal. Nó tính toán giai thừa của một số nguyên bằng phương pháp đệ quy và lưu trữ kết quả trên thanh ghi $v0. Sau đó, các giá trị trên stack được phục hồi bằng cách sử dụng lệnh lw và lệnh addi.

Cuối cùng, các thanh ghi và stack được phục hồi và chương trình kết thúc bằng lệnh syscall và giá trị $v0 được đặt thành 10.

Assignment 5

.data

Message1: .asciiz "Largest: "

Message2: .asciiz "Smallest: "

Comma: .asciiz ","

Endline: .asciiz "\n"

.text

main:

jal warp

print:

add $a1, $v0, $zero # $a1 = result from max(list)

add $a2, $v1, $zero # $a1 = result from min(list)

li $v0, 4

la $a0, Message1

syscall

li $v0, 1

addi $a0, $a1, 0

syscall

li $v0, 4

la $a0, Comma

syscall

li $v0, 1

addi $a0, $t0, 0

syscall

li $v0, 4

la $a0, Endline

syscall

li $v0, 4

la $a0, Message2

syscall

li $v0, 1

addi $a0, $a2, 0

syscall

li $v0, 4

la $a0, Comma

syscall

li $v0, 1

addi $a0, $t1, 0

syscall

quit:

li $v0, 10 #terminate

syscall

endmain:

warp:

addi $fp, $sp, 0

addi $sp, $sp, -32

addi $s0, $zero, 20

sw $s0, 28($sp)

addi $s1, $zero, 30

sw $s1, 24($sp)

addi $s2, $zero, 20

sw $s2, 20($sp)

addi $s3, $zero, -99

sw $s3, 16($sp)

addi $s4, $zero, 40

sw $s4, 12($sp)

addi $s5, $zero, 22

sw $s5, 8($sp)

addi $s6, $zero, 35

sw $s6, 4($sp)

addi $s7, $zero, 21

sw $s7, 0($sp)

addi $v0, $zero, 0x80000000 # value of max element

addi $v1, $zero, 0x7fffffff # value of min element

addi $t0, $zero, 7 # index of max element

addi $t1, $zero, 7 # index of min element

addi $t7, $zero, 7 # index

loop:

lw $t2, 0($sp)

check\_max:

slt $t3, $v0, $t2 #check: max < current

beq $t3, $zero, check\_min # if max > current then next check

addi $v0, $t2, 0 #if max < current then update: max = current

addi $t0, $t7, 0 # update index

check\_min:

slt $t3, $t2, $v1#check: current < min

beq $t3, $zero, continue# if current > min then continue

addi $v1, $t2, 0 #if min > current then update: min = current

addi $t1, $t7, 0 #update idex

continue:

addi $sp, $sp, 4

addi $t7, $t7, -1

bne $sp, $fp, loop

li $fp, 0

jr $ra