**BÀI THỰC HÀNH TUẦN 3**

**KIẾN TRÚC MÁY TÍNH**

Họ và tên: Đinh Huy Dương

MSSV: 20215020

**Bài 1:**

A picture containing text

Description automatically generated

Gán giá trị của

* i tại thanh ghi $s1
* j tại thanh ghi $s2
* x tại thanh ghi $t1
* y tại thanh ghi $t2
* z tại thanh ghi $t3

A picture containing text

Description automatically generated

TRƯỜNG HỢP 1:

$s1 = i = 2

$s2 = j = 1

Tại trường hợp này i>j nên sẽ chuyển sẽ có được kết quả của phần else.

Máy sẽ đầu tiên sẽ so sánh $s2 và $s1 qua lệnh “slt” (nếu $2 nhỏ hơn $1 thì gán biến bằng 1) và gán vào thanh ghi $t0. Lệnh tiếp theo là lệnh rẽ nhánh qua nhãn “else” nếu $t0 khác 0, trong trường hợp này $t1 =1 nên sẽ rẽ nhánh tới nhãn “else”.

Graphical user interface, application, Word

Description automatically generatedQuan sát thanh ghi $pc mang giá trị lần lượt theo trình tự của các lệnh trong file, chỉ tới khi phần rẽ nhánh thì $pc mang giá trị của địa chỉ của phần “else” (tại 0x004001c) và tiếp tục đến địa chỉ 0x0040020 và kết thúc ở “endif” tại 0x0040024

Trong phần lệnh “else” giá trị của y (tại $t2) sẽ bằng 0-1 = -1 =0xffffffff và giá trị của z (tại $t3) bằng 0 = 2\*0 = 0x00000000Table

Description automatically generated

TRƯỜNG HỢP 2:

$s1 = i = 1

$s2 = j = 2

Tại trường hợp này máy sẽ không nhảy đến phần “else” mà tiếp tục cho đến khi cuối cùng gặp lệnh “j endif” để nhảy đến hết lệnh điều kiện. Đến cuối thanh ghi $pc cũng nhảy đến địa chỉ 0x004000024

Graphical user interface, text, application, Word

Description automatically generated

Khi đó, sau lệnh “bne”, chương trình sẽ không rẽ nhánh qua “else” tiếp tục đến 2 lệnh addi, thay đổi giá trị: x (tại $t1) = 0+1 = 1 và z =1 (tại $t3)

Table

Description automatically generated

**Bài 2:**

**Text, letter

Description automatically generated**

Gán: Giá trị biến chạy, i: $s1

Địa chỉ phần tử đầu của mảng A: $s2

Số các phần tử của mảng, n: $s3

Bước của biến chạy i, step: $s4

Tổng của các phần tử trong mảng A, sum: $s5

Đặt các giá trị của các tham số quan trọng vừa được nêu trên. Đặc biệt, lệnh “la” sử dụng để gán địa chỉ của mảng A trong Data Segment

Text

Description automatically generated with low confidence

Khởi tạo giá trị của mảng A với 5 phần tử kiểu word:{2,5,1,8,0} tại Data Segment. Ta có thể thấy ở hàng đầu tại địa chỉ 0x10010000, 5 giá trị đã được khởi tạo. Mỗi ô nhớ cách nhau 4 byte (+4)

Table

Description automatically generated

Sau đó, chương trình sẽ thực hiện nhãn “loop”. Bước đầu tiên là thực hiện lệnh “slt” để so sánh xem $s1 (biến chạy i) có bé hơn $s3 (số phần tử của mảng n). Lệnh rẽ nhánh beq để kết thúc vòng lặp khi i>n -> $t2 =0 -> nhảy đến endloop. Trong trường hợp này, i<n, gán giá trị cho thanh $t2 =1 và tiếp tục chạy.

2 lệnh add tiếp theo dùng để gán $t1 = 4\*i =4\*$s1 với mục đích dùng để sang các ô tiếp theo trong vùng Data Segment (mỗi ô cách nhau 4 byte). Tiếp tục cộng thêm địa chỉ phần tử đầu của mảng A ($s2) vào $t1 để có thể xác định được địa chỉ ô nhớ chính xác của phần tử trong mảng với vị trí i tương ứng.

Khi địa chỉ của phần từ thứ i đã được lưu vào $t1, sử dụng thanh ghi $t0 để lưu GIÁ TRỊ của phần tử i qua địa chỉ tại $t1 bằng lệnh “lw”. Từ đó có thể tính tổng sum ($s5) cộng với thanh $t0, và tăng bước chạy của i (step - $s4), trong trường hợp này, step =1, giá trị i sẽ tăng lần lượt từ 0-1-2-3-4.

Kết thúc lệnh lặp, sử dụng lệnh “j” để nhảy về đầu hàm loop cho lần lặp tiếp theo. Lệnh “j” làm cho giá trị $pc thay đổi từ địa chỉ của “loop” tới kết thúc vòng lặp tại “endloop”.

Table

Description automatically generated

Kết quả của tổng sum tại $s5 = 2+5+1+8+0=16=0x0000 0010

**Bài 3:**

Text, letter

Description automatically generated

Text

Description automatically generated

Đặt a tại $s2, b tại $s3

Khai báo biến test dưới kiểu word. Đây sẽ là biến biểu thị lựa chọn vào các case của chương trình. Khi đó ta sẽ lưu địa chỉ và giá trị của biến test vào thanh ghi qua 2 lệnh “la” và “lw” qua lần lượt thanh $s0 và $s1.

Khi đó ta gán các giá trị của các trường hợp của các giá trị test: 0,1,2 vào trong 3 thanh ghi $t0, $t1, $t2. Khi đó 3 lệnh so sánh và rẽ nhánh “beq” tiếp theo sẽ so sánh và dịch chuyển về các case đã có sẵn. Trường hợp test không bằng bất cứ case nào, chương trình sẽ chạy qua 3 lệnh “beq” và đi thẳng vào lệnh “j” nhảy đến default và kết thúc chương trình ở nhãn “continue”.

Cụ thể, trong trường hợp như trong hình, test = 0, chương trình sẽ nhảy tới case\_0, thay đổi giá trị a (tại $s2) = a+1 =1, và kết thúc ở “continue” với $pc mang địa chỉ “continue”





Table

Description automatically generated

Trong trường hợp test = 1, chương trình nhảy tới case\_1, thay đổi $s2 thành a-1 = -1 = 0xffffffff qua lệnh “sub” để trừ. Tương tự với test =2, giá trị của $s3 sẽ được nhân đôi.

Bất cứ trường hợp test nào khác 3 trường hợp trên sẽ nhảy xuống “default” và kết thúc ở “continue”.

**Bài 4: Mặc định x=y=z=0**

a, i<j

start:

addi $s2, $zero, 0x00001

addi $s1, $zero, 0x00003

slt $t0, $s1, $s2 # i<j -> $st0 ?1:0

beq $t0, $zero, else # branch to “else if j>=i

addi $t1, $t1, 1 # ”then” part: x=x+1

addi $t3, $zero, 1 # z=1

j endif # skip “else” part

else:

addi $t2, $t2, -1 #begin “else” part: y=y-1

add $t3, $t3, $t3 #z=z\*2

endif:

Table

Description automatically generated

Trường hợp này i>j nên bị rẽ nhánh sang “else”, thay đổi y=$t2= 0xffffffff, z=$t3= 0

b, i>=j

start:

addi $s2, $zero, 0x00001

addi $s1, $zero, 0x00003

slt $t0, $s1, $s2 # i<j -> $st0 ?1:0

bne $t0, $zero, else # branch to “else” if i<j

addi $t1, $t1, 1 # ”then” part: x=x+1

addi $t3, $zero, 1 # z=1

j endif # skip “else” part

else:

addi $t2, $t2, -1 # begin “else” part: y=y-1

add $t3, $t3, $t3 # z=z\*2

endif:

Table

Description automatically generated

Trong trường hợp này, i>j, nên sẽ thay đổi thanh ghi của x ($t1) và z($t3) =1

c, i+j<=0

start:

addi $s2, $zero, 0x00001

addi $s1, $zero, 0x00003

add $t0, $s1, $s2 # $t0= i+j

slt $t5, $zero, $t0 # i+j >0 -> $t5 ? 1:0

bne $t5, $zero, else # branch to “else” if i+j >0

addi $t1, $t1, 1 # ”then” part: x=x+1

addi $t3, $zero, 1 # z=1

j endif # skip “else” part

else:

addi $t2, $t2, -1 # begin “else” part: y=y-1

add $t3, $t3, $t3 # z=z\*2

endif:

Table

Description automatically generated

Trường hợp này có i+j ($t5) > 0 nên rẽ nhánh sang nhãn “else”

khiến cho thay đổi thanh ghi của y và z, y=y-1=0-1=-1=0xffffffff

z=z\*2 = 0\*2=0

d, i+j > m+n

Đặt giá trị của m và n lần lượt tại các thanh ghi $s3 và $s4

start:

addi $s1, $zero, 0x00000000 # i = ?

addi $s2, $zero, 0x00000001 # j = ?

addi $s3, $zero, 0x00000001 # m = ?

addi $s4, $zero, 0x00000000 # n = ?

add $t0,$s1,$s2 # $t0 = i+j

add $t4,$s3,$s4 # $t4 = m+n

slt $t5, $t4, $t0 # i+j >m+n -> $t5 ? 1:0

beq $t5,$zero,else # branch to "else" if i+j <= m+n

addi $t1,$t1,1 # "then" part: x=x+1

addi $t3,$zero,1 # z=1

j endif # skip “else” part

else:

addi $t2,$t2,-1 # begin "else" part: y=y-1

add $t3,$t3,$t3 # z=2\*z

endif:

Table

Description automatically generated

Trường hợp này có x+y = m+n nên thanh ghi $t2, $t3 được thay đổi thành:

y=y-1 =0-1 =-1 =0xffffffff

z=z\*2 = 0\*2 = 0x000000000

**Bài 5:**

a, i<n: Giống với Bài 2

b, i<=n

.data

A: .word 2,5,1,8,0

.text

addi $s1, $zero, 0 # i = 0

la $s2, A # Store the Address of A into $s2

addi $s3, $zero, 5 # n = 5

addi $s4, $zero, 1 # step = 1

addi $s5, $zero, 0 # sum = 0

loop:

slt $t2, $s3, $s1 # $t2 = i > n ? 0 : 1

bne $t2, $zero, endloop

add $t1, $s1, $s1 # $t1 = 2 \* $s1

add $t1, $t1, $t1 # $t1 = 4 \* $s1

add $t1, $t1, $s2 # $t1 store the address of A[i]

lw $t0, 0($t1) # load value of A[i] in $t0

add $s5, $s5, $t0 # sum = sum + A[i]

add $s1, $s1, $s4 # i = i + step

j loop # goto loop

endloop:

Table

Description automatically generated

Tương tự với Bài 2, tuy nhiên giá trị của i tại thanh $s1 =6 vì điều kiện để thoát vòng lặp là i>n

c, sum >=0

.data

A: .word 2,5,1,-9,0

.text

addi $s1, $zero, 0 # i = 0

la $s2, A # Store the Address of A into $s2

addi $s3, $zero, 5 # n = 5

addi $s4, $zero, 1 # step = 1

addi $s5, $zero, 0 # sum = 0

loop:

add $t1, $s1, $s1 # $t1 = 2 \* $s1

add $t1, $t1, $t1 # $t1 = 4 \* $s1

add $t1, $t1, $s2 # $t1 store the address of A[i]

lw $t0, 0($t1) # load value of A[i] in $t0

add $s5, $s5, $t0 # sum = sum + A[i]

bltz $s5, endloop

add $s1, $s1, $s4 # i = i + step

j loop # goto loop

endloop:

Table

Description automatically generated

Trường hợp này, tại a[4]=-9, sum =-1, nên chương trình rẽ nhánh tới endloop, thanh ghi sum tại $s5 = 0xffffffff và thanh ghi của i tại $1 = 0x3 và thanh ghi tạm thời $t0 chứa giá trị cuối cùng của mảng trước khi kết thúc lặp: -9=0xfffffff7

d, A[i] = 0

.data

A: .word 2,5,1,8,0,4

.text

addi $s1, $zero, 0 # i = 0

la $s2, A # Store the Address of A into $s2

addi $s3, $zero, 6 # n = 6

addi $s4, $zero, 1 # step = 1

addi $s5, $zero, 0 # sum = 0

loop:

add $t1, $s1, $s1 # $t1 = 2 \* $s1

add $t1, $t1, $t1 # $t1 = 4 \* $s1

add $t1, $t1, $s2 # $t1 store the address of A[i]

lw $t0, 0($t1) # load value of A[i] in $t0

beq $t0, $zero, endloop

add $s5, $s5, $t0 # sum = sum + A[i]

add $s1, $s1, $s4 # i = i + step

j loop # goto loop

endloop:

Table

Description automatically generated

Trong trường hợp này gặp a[4] =0 nên chương trình rẽ sang endloop và dừng lại không tính thêm a[5] =4 vào tổng nên giá trị sum ở thanh $s5 = 0x10 =16

**Bài 6:**

.data

arr: .word 1,2,-4,5,-7

.text

addi $s1, $zero, 0 # i = 0

la $s2, arr # Store the Address of arr into $s2

addi $s3, $zero, 5 # n = 5

addi $s4, $zero, 1 # step = 1

addi $s5, $zero, 0 # max = 0

j loop

absolute:

sub $t0, $zero, $t0 # |n| = 0 - n, n<0

j compare

loop:

slt $t2, $s1, $s3 # $t2 = i < n ? 1 : 0

beq $t2, $zero, endloop

add $t1, $s1, $s1 # $t1 = 2 \* $s1

add $t1, $t1, $t1 # $t1 = 4 \* $s1

add $t1, $t1, $s2 # $t1 store the address of arr[i]

lw $t0, 0($t1) # load value of arr[i] in $t0

add $t4, $zero, $t0 # the original value of arr[i] is stored in $t4

bltz $t0, absolute # if arr[i] <0 then branch to absolute

compare:

slt $t3, $s5, $t0 # arr[i] > max -> $t3 ? 1:0

bne $t3, $zero, maximize

add $s1, $s1, $s4 # i = i + step

j loop # goto loop

maximize:

add $s5, $zero, $t0 # max = arr[i]

add $s6, $zero, $t4 # the value of max

add $s7, $zero, $s1 # the index of max in the array

add $s1, $s1, $s4 # i = i + step

j loop # goto loop

endloop:

Giá trị của biến max để so sánh được lưu vào thanh $s5

Giá trị của phần tử có trị tuyệt đối lớn nhất được lưu vào thanh $s6

Giá trị của vị trí phần tử đấy trong mảng được lưu vào thanh $s7

Mảng arr = {1,2,-4,5,-7}

Kết quả:

Table

Description automatically generated

Tại $s5, giá trị max =7 = |-7|

$s6 = 0xfffffff9 = -7

$s7 = 4 là vị trí của -7 trong mảng arr (arr[4])