**BÀI THỰC HÀNH TUẦN 6**

**KIẾN TRÚC MÁY TÍNH**

Họ và tên: Đinh Huy Dương

MSSV: 20215020

**Bài 1:**

#Assignment 1

.data

in: .asciiz "Input the number of elements 'n': "

in2: .asciiz "Input the element of array: "

A: .word

.text

main:

la $s0, A

j input

next:

j mspfx

nop

continue:

j end

end\_of\_main:

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

mspfx:

addi $v0,$zero,0 # initialize length in $v0 to 0

addi $v1,$zero,0 # initialize max sum in $v1 to 0

addi $t0,$zero,0 # initialize index i in $t0 to 0

addi $t1,$zero,0 # initialize running sum in $t1 to 0

loop:

add $t2,$t0,$t0 # put 2i in $t2

add $t2,$t2,$t2 # put 4i in $t2

add $t3,$t2,$s0 # put 4i+A (address of A[i]) in $t3

lw $t4,0($t3) # load A[i] from mem(t3) into $t4

add $t1,$t1,$t4 # add A[i] to running sum in $t1

slt $t5,$v1,$t1 # set $t5 to 1 if max sum < new sum

bne $t5,$zero,mdfy # if max sum is less, modify results

j test # done?

mdfy:

addi $v0,$t0,1 # new max-sum prefix has length i+1

addi $v1,$t1,0 # new max sum is the running sum

test:

addi $t0,$t0,1 # advance the index i

slt $t5,$t0,$s1 # set $t5 to 1 if i<n

bne $t5,$zero,loop # repeat if i<n

done:

j continue

mspfx\_end:

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

input:

li $v0,4

la $a0,in

syscall

li $v0, 5

syscall

add $s1, $zero, $v0

li $s2,0 # i = 0

li $t0,0 # init $t0, store the current address

while:

li $v0,4

la $a0,in2

syscall

li $v0, 5

syscall

sll $s3,$s2,2 #j=i\*4

add $t0,$s0,$s3 # current A[i]

sw $v0,0($t0)

addi $s2,$s2,1 # i++

blt $s2,$s1,while

j next

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

end:

- Mục đích chương trình: Tìm ra giá trị lớn nhất của các tổng tiền tố (prefix sum)

- Cấu trúc chương trình: Viết tương tự với một chương trình C, có hàm “main” và hàm “mspfx” để xác định yêu cầu đề bài. Trong hàm “main” rẽ nhánh sáng “mspfx” để thực hiện hàm này. Trong hàm “mspfx” duyệt qua từng phần tử trong mảng để cộng dần và so sánh với giá trị max prefix, nếu max prefix bé hơn giá trị tổng mới cộng được, thay thế max prefix.

- Các giá trị của mảng được khai báo với kiểu “word”, 4 byte, nên các địa chỉ ô nhớ của các phần tử cách nhau 4 byte. Sử dụng một biến chạy index i, nhân với 4 và cộng vào địa chỉ phần tử ban đầu, ta sẽ xác định được địa chỉ của phần tử A[i]. (Indexing method).



Giá trị của i được lưu trong thanh ghi $t0, tại $t2 là kết quả nhân 4\*i trong lần lặp, và $t3 = Địa chỉ ban đầu của mảng (0x10010000) cộng với $t2 để ra được địa chỉ của A[i].

Kết quả:

Table

Description automatically generated

Ta có thể thấy tại $t0, sau khi vòng lặp kết thúc, giá trị của i =5 = số lượng phần tử của mảng, thỏa mãn với điều kiện kết thúc vòng lặp. Tại thanh ghi $t3 mang giá trị 0x10010010 là địa chỉ ô nhớ của phần tử cuối cùng của mảng. Kết quả của max của tổng prefix được lưu ở thanh $v1 với giá trị 0xd = 13 = 4+6+3 là tổng của 3 phần tử đầu, $v0 = 3 là độ dài của tổng prefix max.

**Bài 2:**

# Assignment 2

.data

ArMsg: .asciiz "\nThe array of round "

colon: .asciiz " is: "

comma: .asciiz " "

in: .asciiz "\nInput the number of elements 'n': "

in2: .asciiz "Input the element of array: "

temp: .asciiz ""

A: .word

.text

main:

la $s0,A # $a0 = Address(A[0])

j input

next:

sll $s5,$s5,2

add $s1,$s0,$s5

addi $s1,$s1,-4 # $s1 = Address(A[n])

li $s2,0 # init the number of rounds = $s2

add $s3,$s1,$zero # $s3 = Address(A[n]) (Permanent)

j print # sort

after\_sort:

li $v0, 10 # exit

syscall

end\_main:

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

#procedure sort (ascending selection sort using pointer)

print:

j printarr

sort:

beq $s0,$s1,done #single element list is sorted

j max #call the max procedure

after\_max:

lw $t0,0($s1) #load last element into $t0

sw $t0,0($v0) #copy last element to max location

sw $v1,0($s1) #copy max value to last element

addi $s1,$s1,-4 #decrement pointer to last element

addi $s2,$s2,1 #inc the number of rounds

j print #repeat sort for smaller list

done: j after\_sort

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

#Procedure max

#function: fax the value and address of max element in the list

#$s0 pointer to first element

#$s1 pointer to last element

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

max:

addi $v0,$s0,0 # init max pointer to first element

lw $v1,0($v0) # init max value to first value

addi $t0,$s0,0 # init next pointer to first

loop:

beq $t0,$s1,ret # if next=last, return

addi $t0,$t0,4 # advance to next element

lw $t1,0($t0) # load next element into $t1

slt $t2,$t1,$v1 # (next)<(max) ?

bne $t2,$zero,loop # if (next)<(max), repeat

addi $v0,$t0,0 # next element is new max element

addi $v1,$t1,0 # next value is new max value

j loop # change completed; now repeat

ret:

j after\_max

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

#Procedure printarr

#Print the array using pointer method travese

printarr:

li $v0,4

la $a0,ArMsg

syscall

li $v0,1

add $a0,$zero,$s2 # Print the current element

syscall

li $v0,4

la $a0,colon

syscall

add $t0,$zero,$s0

traverse:

li $v0,1

lw $a0,0($t0)

syscall

beq $t0,$s3,endprn

addi $t0,$t0,4 # Travese to the next element

li $v0,4

la $a0,comma

syscall

j traverse

endprn:

j sort

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

input:

li $v0,4

la $a0,in

syscall

li $v0, 5

syscall

add $s5, $zero, $v0

li $s2,0 # i = 0

li $t0,0 # init $t0, store the current address

while:

li $v0,4

la $a0,in2

syscall

li $v0, 5

syscall

sll $s3,$s2,2 #j=i\*4

add $t0,$s0,$s3 # current A[i]

sw $v0,0($t0)

addi $s2,$s2,1 # i++

blt $s2,$s5,while

j next

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

Chương trình thực hiện thuật toán Insertion sort để sắp xếp mảng. Điểm đáng lưu ý là phương thức duyệt mảng trong chương trình này khác với chương trình ở bài 1. Trong chương trình này, các phần tử không được duyệt thông qua một giá trị index, mà di chuyển con trỏ (địa chỉ) của các phần tử lần lượt tăng lên 4 để duyệt đến hết mảng, giá trị con trỏ này được lưu ở $t0.

Về thuật toán Insertion Sort: Duyệt qua các phần tử trong mảng, tìm phần tử lớn nhất và đẩy nó về vị trí cuối cùng. Từ đó thuật toán lặp lại với các mảng có độ dài trừ dần đi 1 do các phần tử ở dưới đã được coi là đã sắp xếp. Thủ tục “max” được sử dụng để tìm phần tử lớn nhất trong mảng với phần tử cuối cùng có địa chỉ $s1. Sau khi tìm ra phần tử lớn nhất trong mảng con, “max” sẽ quay về “after\_max” trong “sort” để đảo phần tử lớn nhất xuống cuối. $s1 sẽ bị trừ dần đi 4 (đúng với địa chỉ phần tử các mảng) cho đến khi gặp được $s0 là địa chỉ của phần tử đầu tiên và mảng đã được sắp xếp thành công.

Chương trình bao gồm thủ tục “printarr” để in ra sự thay đổi của mảng A sau mỗi lần lặp

Kết quả:

Calendar

Description automatically generated with low confidence

Ta đặt $s1 là giá trị địa chỉ của phần tử cuối cùng

**Bài 3:**

# Assignment 3

.data

ArMsg: .asciiz "\nThe array of round "

colon: .asciiz " is: "

comma: .asciiz " "

in: .asciiz "\nInput the number of elements 'n': "

in2: .asciiz "Input the element of array: "

temp: .asciiz ""

A: .word

.text

main:

la $s0,A # $s0 = Address(A[0])

j input

continue:

addi $s5,$s1,-1

li $s4,0 # number of rounds

j print

end\_sort:

li $v0, 10 # exit

syscall

end\_main:

#--------- Procedure Sort -----------------------------

print:

j printarr

sort:

li $s2,-1 # init index =-1

li $v0,0 # swapping status

add $t0,$zero,$s0 # $t0 = address (A[0])

load:

lw $t1, 0($t0) # $t1 = A[i]

lw $t2, 4($t0) # $t2 = A[i+1]

blt $t2, $t1, swap # if A[i+1]<A[i] then swap

next:

addi $s2, $s2, 1 # i++

add $s3, $s2, $s2 # $s3 = 2.$s2

add $s3, $s3, $s3 # $s3 = 4.$s2

add $t0, $s0, $s3 # next element

blt $s2, $s5, load # if i<n, jump back to load

bne $v0, $zero, print # if swapping status =1, return to sort from A[0]

j end\_sort

swap:

sw $t1, 4($t0) # A[i] = \*($t0+4) = A[i+1]

sw $t2, 0($t0) # A[i+1] = \*(t0) = A[i]

addi $v0, $v0, 1 # swapping status =1

j next # return to loop

#------------------Procedure Printarr -----------------

printarr:

li $v0,4

la $a0,ArMsg

syscall

li $v0,1

add $a0,$zero,$s4 # Print the current element

syscall

addi $s4,$s4,1 #number of rounds++

li $v0,4

la $a0,colon

syscall

add $t0,$zero,$s0

li $t3,0 # j = 0

traverse:

li $v0,1

lw $a0,0($t0)

syscall

addi $t3,$t3,1 # j++

sll $t4,$t3,2 # j = j\*4

add $t0,$s0,$t4 # Travese to the next element

bge $t3,$s1,endprn

li $v0,4

la $a0,comma

syscall

j traverse

endprn:

j sort

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

input:

li $v0,4

la $a0,in

syscall

li $v0, 5

syscall

add $s1, $zero, $v0

li $s2,0 # i = 0

li $t0,0 # init $t0, store the current address

while:

li $v0,4

la $a0,in2

syscall

li $v0, 5

syscall

sll $s3,$s2,2 #j=i\*4

add $t0,$s0,$s3 # current A[i]

sw $v0,0($t0)

addi $s2,$s2,1 # i++

blt $s2,$s1,while

j continue

