Laboratory Exercise 6

Array and Pointer

Bài 1:

Code

```
.data
A: .word -2, 6, -1, 3, 20200440,-20200446
.text
main:   la $a0,A
        li $a1,6
        j mspfx
        nop
continue:
lock:   j lock
        nop
end_of_main:
mspfx:  addi $v0,$zero,0 #initialize length in $v0 to 0
        addi $v1,$zero,0 #initialize max sum in $v1to 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,$a0 #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,$a1 #set $t5 to 1 if i<n
        bne $t5,$zero,loop #repeat if i<n
done: j continue
mspfx_end:
```

Sau khi chạy, v1 = 20200446

Kết quả đó = -2 + 6 -1 + 3 + 20200440 = 20200446

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	5
$v1	3	20200446
$a0	4	268500992
$a1	5	6
$a2	6	0

Các giá trị ở thanh ghi khác

$a3	7	0
$t0	8	6
$t1	9	0
$t2	10	20
$t3	11	268501012
$t4	12	-20200446
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194324
hi		0

Giải thích các bước

Khai báo dữ liệu thuộc kiểu data

Sau đó ta khai báo mảng A có thuộc tính số nguyên

Vào mảng chính

Đầu tiên ta gán địa chỉ của mảng A vào a0, gán a1 = 6 vì mảng ở đây có 6 phần tử

Nhảy tới mspfx

Tạo ra chương trình con continue (tạo vòng lặp)

Tạo ra vòng lặp vô hạn look để dừng chương trình

Khởi tạo v0 = 0

V1=0, t0 = 0, t1 = 0

V1 = 0 khởi tạo ra maxprefix sum ở thnah ghi v1

T0 để tạo ra chỉ số I (ở đây ban đầu bằng 0)

Khởi tạo tổng chạy t1 = 0

Vòng lặp lôp gán t2 = 2i

Gán t2 = 2t2 = 4i

Gán địa chỉ của A[i] vào thanh ghi t3

Load giá trị của A[i] vào thanh ghi t4

Them A[i] vào tổng chạy của t1

Sau đó ta mang đi so sánh v1 tổng cần tìm với t1 tổng chạy, nếu mà v1<t1 thì t5 = 1

ở dòng số 22

nếu như t5 != 0 thì nhảy tới nhãn mdfy còn nếu không thì nhảy tới test

ở mdfy thì dãy maxprefix có độ dài i+1

sau đó ta gán v1 = t1

test tang chỉ số của i lên

sau đó ta so sánh nếu như t0 < a1 thì t5 sẽ bằng 1

và nếu như t5 mà khác 0 thì nhảy lại vòng lặp loop

sau khi thoả mãn hết yêu cầu thì nhảy tới continue và kết thúc chương trình

Bài 2 :

Code

```
.data
A: .word 7, -2, 28, 1, 20200440, 5
Aend: .word
.text
main: la $a0,A #$a0 = Address(A[0])
la $a1, Aend
 addi $a1,$a1,-4 #$a1 = Address(A[n-1])
 j sort #sort
after_sort: li $v0, 10 #exit
 syscall
end_main:

sort: beq $a0,$a1,done #single element list is sorted
 j max #call the max procedure
after_max: lw $t0,0($a1) #load last element into $t0
 sw $t0,0($v0) #copy last element to max location
 sw $v1,0($a1) #copy max value to last element
 addi $a1,$a1,-4 #decrement pointer to last element
 j sort #repeat sort for smaller list
done: j after_sort

max:
addi $v0,$a0,0 #init max pointer to first element
lw $v1,0($v0) #init max value to first value
addi $t0,$a0,0 #init next pointer to first
loop:
beq $t0,$a1,ret #if next=last, return
addi $t0,$t0,4 #advance to next element
lw $t1,0($t0) #load next element into $t1
```

```
max:
addi $v0,$a0,0 #init max pointer to first element
lw $v1,0($v0) #init max value to first value
addi $t0,$a0,0 #init next pointer to first
loop:
beq $t0,$a1,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
```

Run

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,4097	5: main: la $a0,A #$a0 = Address(A[0])
	0x00400004	0x34240000	ori $4,$1,0	
	0x00400008	0x3c011001	lui $1,4097	6: la $a1, Aend
	0x0040000c	0x34250018	ori $5,$1,24	
	0x00400010	0x20a5fffc	addi $5,$5,-4	7: addi $a1,$a1,-4 #$a1 = Address(A[...
	0x00400014	0x08100008	j 0x00400020	8: j sort #sort
	0x00400018	0x2402000a	addiu $2,$0,10	9: after_sort: li $v0, 10 #exit
	0x0040001c	0x0000000c	syscall	10: syscall
	0x00400020	0x10850006	beq $4,$5,6	13: sort: beq $a0,$a1,done #single ele...
	0x00400024	0x08100010	j 0x00400040	14: j max #call the max procedure
	0x00400028	0x8ca80000	lw $8,0($5)	15: after_max: lw $t0,0($a1) #load las...
	0x0040002c	0xac480000	sw $8,0($2)	16: sw $t0,0($v0) #copy last element ...
	0x00400030	0xaca30000	sw $3,0($5)	17: sw $v1,0($a1) #copy max value to ...
	0x00400034	0x20a5fffc	addi $5,$5,-4	18: addi $a1,$a1,-4 #decrement pointe...
	0x00400038	0x08100008	j 0x00400020	19: j sort #repeat sort for smaller list
	0x0040003c	0x08100006	j 0x00400018	20: done: j after_sort
	0x00400040	0x20820000	addi $2,$4,0	23: addi $v0,$a0,0 #init max pointer t...
	0x00400044	0x8c430000	lw $3,0($2)	24: lw $v1,0($v0) #init max value to f...
	0x00400048	0x20880000	addi $8,$4,0	25: addi $t0,$a0,0 #init next pointer ...
	0x0040004c	0x11050007	beq $8,$5,7	27: beq $t0,$a1,ret #if next=last, return
	0x00400050	0x21080004	addi $8,$8,4	28: addi $t0,$t0,4 #advance to next el...
	0x00400054	0x8d090000	lw $9,0($8)	29: lw $t1,0($t0) #load next element i...
	0x00400058	0x0123502a	slt $10,$9,$3	30: slt $t2,$t1,$v1 #(next)<(max) ?
	0x0040005c	0x1540fffb	bne $10,$0,-5	31: bne $t2,$zero,loop #if (next)<(max...
	0x00400060	0x21020000	addi $2,$8,0	32: addi $v0,$t0,0 #next element is ne...
	0x00400064	0x21230000	addi $3,$9,0	33: addi $v1,$t1,0 #next value is new ...
	0x00400068	0x08100013	j 0x0040004c	34: j loop #change completed; now repeat
	0x0040006c	0x0810000a	j 0x00400028	36: j after_max

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-2	1	5	7	28	20200440	0	0
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0
0x100100c0	0	0	0	0	0	0	0	0
0x100100e0	0	0	0	0	0	0	0	0
0x10010100	0	0	0	0	0	0	0	0
0x10010120	0	0	0	0	0	0	0	0

0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☐ ASCII

Giải thích code

Đầu tiên ta khai báo chuỗi

Aend kiểu word

Ta gán địa chỉ của A vào a0

Gán địa chỉ của Aend vào a1

Giảm a1 đi 4 vì 4 bit thành A[n-1] sau đó nhảy tới sort

After_sort: thực hiện gán v0 = 10, 10 là exit thoát

Sau đó ta gọi lệnh syscall

Nhãn sort: thực hiện so sánh nếu a0 = a1 thì nhảy tới done

Nếu như ko bằng thì nhảy tới max

Nhãn addter_max để lưu giá trị của phần tử cuối vào t0

Copy giá trị cuỷa phần tử cuối của vùng lớp nhất, copy giá trị lớn nhất vào phần tử cuối

Giảm địa chỉ đi 4

Nhảy quay lại sort cho mảng bé hơn (trừ phần tử đã thực hiện đưa về cuối)

Nhãn done nhảy tới after_sort để thoát

Gán giá trị v0 = a0, v0 là con trỏ lớn nhất

Load địa chỉ của v0 vào v1 sau đó ta gán t0 về địa chỉ đầu tiên của mảng

ở dòng 27 ta mang đi so sánh nếu như phần tử tiếp theo là cuối cùng thì nhảy tới ret nếu không ta thực hiện tiếp chạy chương trình ta t0 lên 4 đơn vị load địa chỉ của t0 phần tử tiếp theo vào t1

nếu như t2 != 0 tức là t1<v1 thì quay lại vòng lặp loop

gán phần tử tiếp theo thành phần tử lớn nhất mới

gán giá trị t1 vào v1 vậy thì phần tử kế tiếp sẽ là giá trị max

ở dòng 34 ta nhảy tới loop

dòng 36 ta nhảy tới after_max

Bài 3 :

Code

```
.data
arr: .word 440, 2020, 0, -89, 5, 100,-1008, -9, 10
space: .asciiz " "
.text
  la $s0, arr                    #Luu dia chi cua arr vao s0
  li $t0, 0                      #i = 0
  li $s1, 8                      #n = 9
  li $s2, 9
  add $t2, $zero, $s0
outer_loop:
  li  $t1, 0                      #j = 0
  addi $s2, $s2, -1
  add $t3, $zero, $s0
inner_loop:
lw $s3, 0($t3)
addi $t3, $t3, 4
lw $s4, 0($t3)
addi $t1, $t1, 1
slt $t4, $s3, $s4
bne $t4, $zero, cond
swap: sw $s3, 0($t3)
sw $s4, -4($t3)
lw $s4, 0($t3)
cond: bne $t1, $s2, inner_loop          #j != n-i
addi $t0, $t0, 1                        #i++
```

```
addi $t0, $t0, 1                        #i++
bne $t0, $s1, outer_loop                #i != n
li $t0, 0
addi $s1, $s1, 1
print: li $v0, 1
lw $a0, 0($t2)
syscall
li $v0, 4
la $a0, space
syscall
addi $t2, $t2, 4
addi $t0, $t0, 1
bne $t0, $s1, print                     #i != n
```

Run

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,4097	5: la $s0, arr #Luu dia
	0x00400004	0x34300000	ori $16,$1,0	
	0x00400008	0x24080000	addiu $8,$0,0	6: li $t0, 0 #i = 0
	0x0040000c	0x24110008	addiu $17,$0,8	7: li $s1, 8 #n = 9
	0x00400010	0x24120009	addiu $18,$0,9	8: li $s2, 9
	0x00400014	0x00105020	add $10,$0,$16	9: add $t2, $zero, $s0
	0x00400018	0x24090000	addiu $9,$0,0	11: li $t1, 0 #j = 0
	0x0040001c	0x2252ffff	addi $18,$18,-1	12: addi $s2, $s2, -1
	0x00400020	0x00105820	add $11,$0,$16	13: add $t3, $zero, $s0
	0x00400024	0x8d730000	lw $19,0($11)	15: lw $s3, 0($t3)
	0x00400028	0x216b0004	addi $11,$11,4	16: addi $t3, $t3, 4
	0x0040002c	0x8d740000	lw $20,0($11)	17: lw $s4, 0($t3)
	0x00400030	0x21290001	addi $9,$9,1	18: addi $t1, $t1, 1
	0x00400034	0x0274602a	slt $12,$19,$20	19: slt $t4, $s3, $s4
	0x00400038	0x15800003	bne $12,$0,3	20: bne $t4, $zero, cond
	0x0040003c	0xad730000	sw $19,0($11)	21: swap: sw $s3, 0($t3)
	0x00400040	0xad74fffc	sw $20,-4($11)	22: sw $s4, -4($t3)
	0x00400044	0x8d740000	lw $20,0($11)	23: lw $s4, 0($t3)
	0x00400048	0x1532fff6	bne $9,$18,-10	24: cond: bne $t1, $s2, inner_loop #
	0x0040004c	0x21080001	addi $8,$8,1	25: addi $t0, $t0, 1 #
	0x00400050	0x1511fff1	bne $8,$17,-15	26: bne $t0, $s1, outer_loop #
	0x00400054	0x24080000	addiu $8,$0,0	27: li $t0, 0
	0x00400058	0x22310001	addi $17,$17,1	28: addi $s1, $s1, 1
	0x0040005c	0x24020001	addiu $2,$0,1	29: print: li $v0, 1
	0x00400060	0x8d440000	lw $4,0($10)	30: lw $a0, 0($t2)
	0x00400064	0x0000000c	syscall	31: syscall
	0x00400068	0x24020004	addiu $2,$0,4	32: li $v0, 4
	0x0040006c	0x3c011001	lui $1,4097	33: la $a0, space
	0x00400070	0x34240024	ori $4,$1,36	
	0x00400074	0x0000000c	syscall	34: syscall
	0x00400078	0x214a0004	addi $10,$10,4	35: addi $t2, $t2, 4

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-1008	-89	-9	0	5	10	100	440
0x10010020	2020	32	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0
0x100100a0	0	0	0	0	0	0	0	0

Giải thích

Khởi tạo các dữ liệu mảng, xâu khoảng trắng

Ta thực hiện gán địa chỉ của mảng ar vào thanh ghi s0

Gán biến chạy i = 0, gán s1 = 8 vào phần tử mảng (có 9 phần tử vào chạy từ 0 tới 8)

Gán s2 = 9 để giảm dần

Tán t2 = địa chỉ của mảng

Outer_lôp gán t1 = 0, giảm s2 đi 1 sau đó gán t3 = địa chỉ của mảng Arr[0]

Sau đó nhãn inner_lôp: load giá trị của Arr[0]

Nhãn inner_loop giá trị của Arr[j] vào s3

Tang t3 lên 4 load giá trị của mảng arr[j+1] sau đó ta tang t1 lên 1 đơn vị

So sánh nếu như mà s3<s4 thì t4=1

Nếu như t4 != 0 thì nhảy tới cond

Nhãu swap ghi địa chỉ của s3 vào t3

Ghi địa chỉ của t3 -4 vào s4 sau đó gi lại giá trị của t3 vào s4

Nhãu cond: so sánh nếu như mà t1!=s2 tức là j chưa tới cuối mảng thì nhảy tới inner_loop

Tăng s1 lên

Nhãn print (in mảng ra màn hình bằng lệnh mời hệ thống syscall)

Gán v0 = 1 vì 1 là lệnh in số nguyên

Load tới địa chỉ của Arr[i] vào a0

Gọi lệnh in

Gán v0 = 4, 4 là lệnh in chuỗi ký tự

Gán space vào a0, gọi lệnh in chuỗi kí tự, tang t2 lên 4 để tìm tới phần tử tiếp theo

Tang t0 lên 1

Nếu như mà t0 != s1 thì nhảy lại print để in các phần tử còn lại chưa được in

Bài 4

Code

```
.data
        A: .word 22, 7, 20200440, -26, 1, -100,10
        space: .asciiz " "
.text
la $a0, A       #$Gan a0 la dia chi cua mang A
li $a1, 7       # Do dai mang A: n
li $t0, 1       # chi so i cua vong lap thu nhat

loop:
add $t1, $t0, $t0
add $t2, $t1, $t1
add $t2, $t2, $a0
lw  $s1, 0($t2)         # load A[i]; key = arr[i]
addi $t3, $t0, -1               # j = i-1

while:slt $t8, $t3, $zero       # if j < 0
bne $t8, $zero, end_while
add $t4, $t3, $t3
add $t5, $t4, $t4
add $t5, $t5, $a0
lw  $s2, 0($t5)         #  arr[j]

slt $t9, $s1, $s2       # key < arr[j]
beq $t9, $zero, end_while
        # Swap
lw $s3, 4($t5)
sw $s3, 0($t5)          # arr[j + 1] = arr[j]
sw $s2, 4($t5)
```

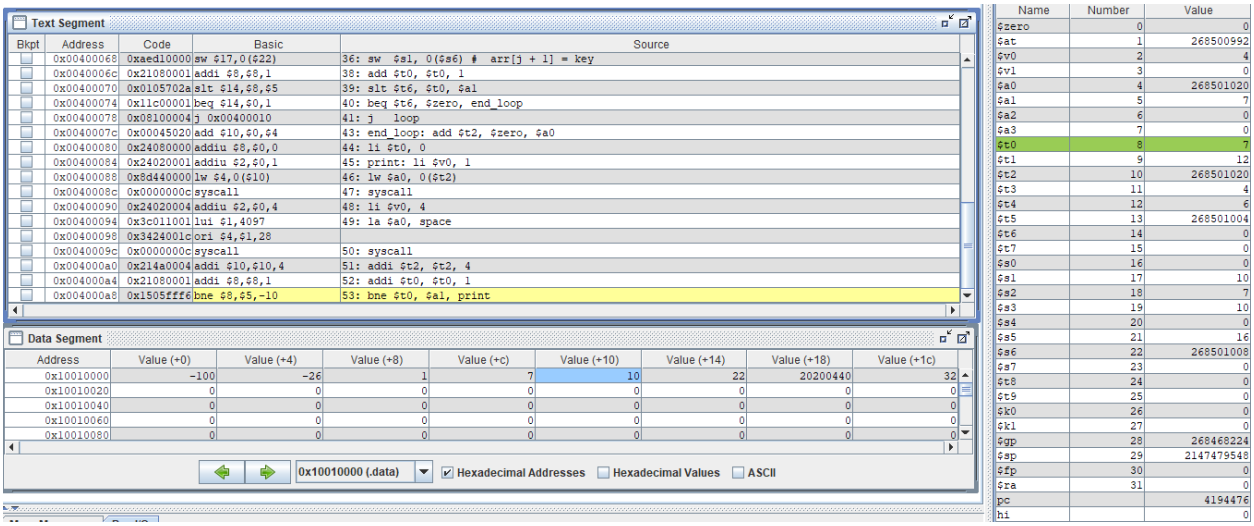
```
sw $s3, 0($t5)          # arr[j + 1] = arr[j]
sw $s2, 4($t5)

addi $t3, $t3, -1               # j -= 1
j   while
end_while:
add $t3, $t3, 1
mul $s5, $t3, 4
add $s6, $s5, $a0
sw  $s1, 0($s6) #  arr[j + 1] = key

add $t0, $t0, 1
slt $t6, $t0, $a1
beq $t6, $zero, end_loop
j   loop

end_loop: add $t2, $zero, $a0
li $t0, 0
print: li $v0, 1
lw $a0, 0($t2)
syscall
li $v0, 4
la $a0, space
syscall
addi $t2, $t2, 4
addi $t0, $t0, 1
bne $t0, $a1, print
```

Run



Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400068	0xaed10000	sw $17,0($22)	36: sw $s1, 0($s6) # arr[j + 1] = key
	0x0040006c	0x21080001	addi $8,$8,1	38: add $t0, $t0, 1
	0x00400070	0x0105702a	slt $14,$8,$5	39: slt $t6, $t0, $a1
	0x00400074	0x11c00001	beq $14,$0,1	40: beq $t6, $zero, end_loop
	0x00400078	0x08100004	j 0x00400010	41: j loop
	0x0040007c	0x00045020	add $10,$0,$4	43: end_loop: add $t2, $zero, $a0
	0x00400080	0x24080000	addiu $8,$0,0	44: li $t0, 0
	0x00400084	0x24020001	addiu $2,$0,1	45: print: li $v0, 1
	0x00400088	0x8d440000	lw $4,0($10)	46: lw $a0, 0($t2)
	0x0040008c	0x0000000c	syscall	47: syscall
	0x00400090	0x24020004	addiu $2,$0,4	48: li $v0, 4
	0x00400094	0x3c011001	lui $1,4097	49: la $a0, space
	0x00400098	0x3424001c	ori $4,$1,28	
	0x0040009c	0x0000000c	syscall	50: syscall
	0x004000a0	0x214a0004	addi $10,$10,4	51: addi $t2, $t2, 4
	0x004000a4	0x21080001	addi $8,$8,1	52: addi $t0, $t0, 1
	0x004000a8	0x1505fff6	bne $8,$5,-10	53: bne $t0, $a1, print

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	-100	-26	1	7	10	22	20200440	32
0x10010020	0	0	0	0	0	0	0	0
0x10010040	0	0	0	0	0	0	0	0
0x10010060	0	0	0	0	0	0	0	0
0x10010080	0	0	0	0	0	0	0	0

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	4
$v1	3	0
$a0	4	268501020
$a1	5	7
$a2	6	0
$a3	7	0
$t0	8	7
$t1	9	12
$t2	10	268501020
$t3	11	4
$t4	12	6
$t5	13	268501004
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	10
$s2	18	7
$s3	19	10
$s4	20	0
$s5	21	16
$s6	22	268501008
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194476
hi		0

+ Dòng 1: vùng dữ liệu, khai báo các biến
+ Dòng 2: Khai báo mảng
+ dòng 3: khai báo xâu space (dấu cách để in mảng)
+ Dòng 4: Vùng lệnh, chứa các lệnh hợp ngữ
+ Dòng 5: gán địa chỉ của mảng A vào thanh ghi a0
+ Dòng 6: gán độ dài của mảng a1 = n = 7
+ Dòng 7: khai báo t0 = 1
+ Dòng 10: nhãn loop: gán t1 = 2t0
+ Dòng 11: gán t2 = 4t0
+ Dòng 12: lấy địa chỉ của a0 + 4t0 gán vào t2
+ Dòng 13: lấy giá trị của $t2 gán vào s1
+ Dòng 14: giảm biến t0 đi 1 gán vào t3
+ Dòng 16: nhãn while: so sánh nếu t3 < 0 thì t8 = 1
+ Dòng 17: nếu t8 != 0 (tức t3 < 0) thì nhảy đến end_while
+ Dòng 18: t4 = 2t3
+ Dòng 19: t5 = 4t3
+ Dòng 20: lấy địa chỉ a0 + 4t3 gán vào t5
+ Dòng 21: lấy giá trị từ t5 vào s2
+ Dòng 23: so sánh nếu s1 < s2 thì t9 = 1
+ Dòng 24: nếu t9 = 0 (tức s1 >= s2) thì nhảy đến end_while
+ Dòng 26: ghi giá trị tiếp sau phần tử hiện tại vào s3
+ Dòng 27: lưu địa chỉ của s3 vào t5

+ Dòng 28: lưu địa chỉ của s2 vào t5 + 4
+ Dòng 30:giảm t3 đi 1
+ Dòng 31: nhảy lại đến while
+ Dòng 33: nhãn end_while: tăng t3 lên 1
+ Dòng 34:
+ Dòng 35: gán s6 = s5 + địa chỉ mảng
+ Dòng 36: lưu địa chỉ s1 vào s6
+ Dòng 38: tăng t0 lên 1
+ Dòng 39: so sánh xem t0 < a1 thì gán t6 = 1
+ Dòng 40: Nếu t6 = 0 (tức t0 >= a1) thì nhảy đến end_loop
+ Dòng 41: còn không thì quay lại vòng lặp loop
+ Dòng 43: nhãn end_loop: gán lại t2 là địa chỉ của a[0]
+ Dòng 44: gán t0 = 0
+ Dòng 45: nhãn print(in mảng): gán v0 = 1. 1 là lệnh in số nguyên
+ Dòng 46: load địa chỉ của A[i] vào a0
+ Dòng 47: gọi lệnh in
+ Dòng 48: gán v0 = 4. 4 là lệnh in chuỗi ký tự
+ Dòng 49: gán địa chỉ của space vào a0
+ Dòng 50: gọi lệnh in chuỗi ký tự
+ Dòng 51: tăng t2 lên 4 (Để đến phần tử tiếp theo của mảng)
+ Dòng 52: tăng t0 lên 1
+ Dòng 53: Nếu t0 != s1 (chưa đến cuối mảng) thì nhảy lại đến print để in các phần tử còn lại.