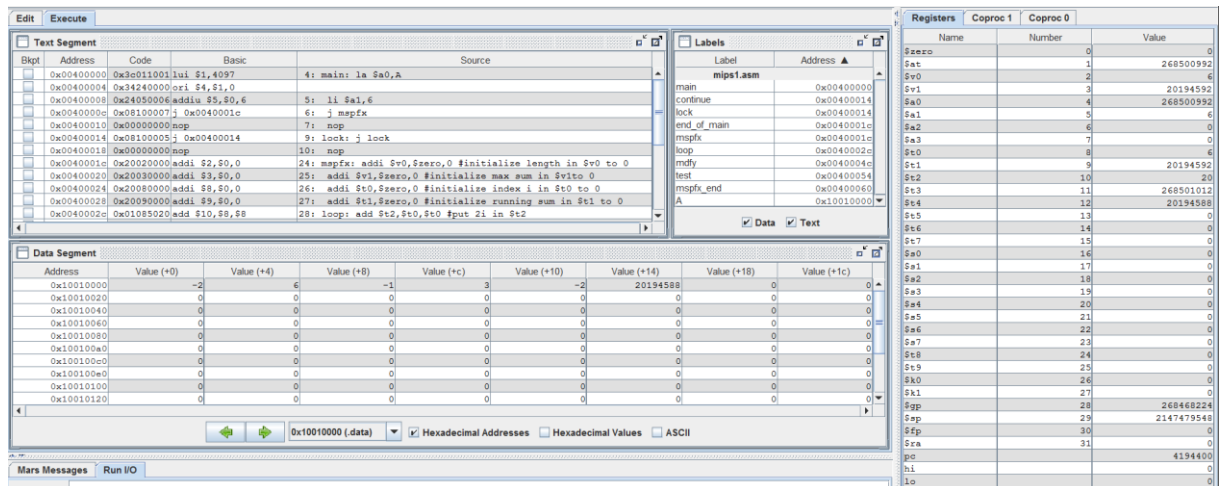


Assignment 1:

- Mã nguồn:

```
mips1.asm
1  .data
2  A: .word -2, 6, -1, 3, -2, 20194588
3  .text
4  main: la $a0,A
5        li $a1,6
6        j mspfx
7        nop
8  continue:
9  lock: j lock
10       nop
11  end_of_main:
12  #-----
13  #Procedure mspfx
14  # @brief find the maximum-sum prefix in a list of integers
15  # @param[in] a0 the base address of this list(A) need to be processed
16  # @param[in] a1 the number of elements in list(A)
17  # @param[out] v0 the length of sub-array of A in which max sum reaches.
18  # @param[out] v1 the max sum of a certain sub-array
19  #-----
20  #Procedure mspfx
21  #function: find the maximum-sum prefix in a list of integers
22
23  #elements is stored in a1
24  mspfx: addi $v0,$zero,0 #initialize length in $v0 to 0
25        addi $v1,$zero,0 #initialize max sum in $v1 to 0
26        addi $t0,$zero,0 #initialize index i in $t0 to 0
27        addi $t1,$zero,0 #initialize running sum in $t1 to 0
28  loop: add $t2,$t0,$t0 #put 2i in $t2
29        add $t2,$t2,$t2 #put 4i in $t2
30        add $t3,$t2,$a0 #put 4i+A (address of A[i]) in $t3
31        lw $t4,0($t3) #load A[i] from mem(t3) into $t4
32        add $t1,$t1,$t4 #add A[i] to running sum in $t1
33        slt $t5,$v1,$t1 #set $t5 to 1 if max sum < new sum
34        bne $t5,$zero,mdfy #if max sum is less, modify results
35        j test #done?
36  mdfy: addi $v0,$t0,1 #new max-sum prefix has length i+1
37        addi $v1,$t1,0 #new max sum is the running sum
38  test: addi $t0,$t0,1 #advance the index i
39        slt $t5,$t0,$a1 #set $t5 to 1 if i<n
40        bne $t5,$zero,loop #repeat if i<n
41  done: j continue
42  mspfx_end:
```

- Màn hình chạy:



- Kết quả = 20194592 => Chính xác
- Giải thích:
 - Dòng 1,2: Khai báo mảng
 - Dòng 4: load địa chỉ A vào a0
 - Dòng 5: Gán a1 = 6 (= số phần tử)
 - Dòng 6: Chạy mspfx
 - Mspfx (dòng 24 – 35):
 - Dòng 24-27: Khởi tạo các giá trị
 - Dòng 28,30: Lấy từng phần tử của A
 - Dòng 31: load phần tử hiện tại (t3) vào t4
 - Dòng 32: This running sum
 - Dòng 33-34: So sánh running sum với max sum, nếu max sum nhỏ hơn => thực hiện modify (Gán lại max sum dòng 36,37)
 - Dòng 35: kiểm tra điều kiện lặp với hàm test
 - Dòng 41: Sau khi xong thì lặp vô hạn để chờ tín hiệu tiếp

Assignment 2:

- Mã nguồn:

```

1  .data
2  A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5, 20194588
3  Aend: .word
4  .text
5  main: la $a0,A #$a0 = Address(A[0])
6        la $a1,Aend
7        addi $a1,$a1,-4 #$a1 = Address(A[n-1])
8        j sort #sort
9  after_sort: li $v0, 10 #exit
10       syscall
11  end_main:
12  #-----
13  #procedure sort (ascending selection sort using pointer)
14  #register usage in sort program
15  #$a0 pointer to the first element in unsorted part
16  #$a1 pointer to the last element in unsorted part
17  #$t0 temporary place for value of last element
18  #$v0 pointer to max element in unsorted part
19  #$v1 value of max element in unsorted part
20  #-----
21  sort: beq $a0,$a1,done #single element list is sorted
22
23  j max #call the max procedure
24  after_max: lw $t0,0($a1) #load last element into $t0
25            sw $t0,0($v0) #copy last element to max location
26            sw $v1,0($a1) #copy max value to last element
27            addi $a1,$a1,-4 #decrement pointer to last element
28            j sort #repeat sort for smaller list
29  done: j after_sort
30  #-----
31  #Procedure max
32  #function: fax the value and address of max element in the list
33  #$a0 pointer to first element
34  #$a1 pointer to last element
35  #-----
36  max:
37  addi $v0,$a0,0 #init max pointer to first element
38  lw $v1,0($v0) #init max value to first value
39  addi $t0,$a0,0 #init next pointer to first
40  loop:
41  beq $t0,$a1,ret #if next=last, return
42  addi $t0,$t0,4 #advance to next element
43  lw $t1,0($t0) #load next element into $t1

```

Edit Execute				Registers Coproc 1 Coproc 0																																																																																																				
Text Segment <table border="1"> <thead> <tr> <th>Bkpt</th> <th>Address</th> <th>Code</th> <th>Basic</th> <th>Source</th> </tr> </thead> <tbody> <tr><td></td><td>0x00400000</td><td>0x3e010001</td><td>lui \$t, 4097</td><td></td></tr> <tr><td>0x00400004</td><td>0x34240000</td><td>ori \$t, \$t, 0</td><td></td><td>5: main: la \$a0, A # \$a0 = Address(A[0])</td></tr> <tr><td>0x00400008</td><td>0x3e010001</td><td>lui \$t, 4097</td><td></td><td>6: la \$a1, Aend</td></tr> <tr><td>0x0040000c</td><td>0x34250038</td><td>ori \$t, \$t, 156</td><td></td><td></td></tr> <tr><td>0x00400010</td><td>0x20a5fffc</td><td>addi \$t, \$t, -4</td><td></td><td>7: addi \$a1, \$a1, -4 # \$a1 = Address(A[n-1])</td></tr> <tr><td>0x00400014</td><td>0x08100008</td><td>j 0x00400020</td><td></td><td>8: j sort</td></tr> <tr><td>0x00400018</td><td>0x2402000a</td><td>addiu \$t, \$t, 10</td><td></td><td>9: after sort: li \$r0, 10 # exit</td></tr> <tr><td>0x0040001c</td><td>0x0000000b</td><td>syscall</td><td></td><td>10: syscall</td></tr> <tr><td>0x00400020</td><td>0x10850006</td><td>bqz \$t, \$t, 56</td><td></td><td>21: sort: bqz \$a0, \$a1, done # single element list is sorted</td></tr> <tr><td>0x00400024</td><td>0x08100010</td><td>j 0x00400040</td><td></td><td>22: j max find the max procedure</td></tr> <tr><td>0x00400028</td><td>0x0ac00000</td><td>lw \$r0, 0(\$t)</td><td></td><td>23: after max: lw \$r0, 0(\$a1) # load last element into \$r0</td></tr> <tr><td>0x0040002c</td><td>0x0ac00000</td><td>sw \$r0, 0(\$t)</td><td></td><td>24: sw \$r0, 0(\$r0) # copy last element to max location</td></tr> </tbody> </table>				Bkpt	Address	Code	Basic	Source		0x00400000	0x3e010001	lui \$t, 4097		0x00400004	0x34240000	ori \$t, \$t, 0		5: main: la \$a0, A # \$a0 = Address(A[0])	0x00400008	0x3e010001	lui \$t, 4097		6: la \$a1, Aend	0x0040000c	0x34250038	ori \$t, \$t, 156			0x00400010	0x20a5fffc	addi \$t, \$t, -4		7: addi \$a1, \$a1, -4 # \$a1 = Address(A[n-1])	0x00400014	0x08100008	j 0x00400020		8: j sort	0x00400018	0x2402000a	addiu \$t, \$t, 10		9: after sort: li \$r0, 10 # exit	0x0040001c	0x0000000b	syscall		10: syscall	0x00400020	0x10850006	bqz \$t, \$t, 56		21: sort: bqz \$a0, \$a1, done # single element list is sorted	0x00400024	0x08100010	j 0x00400040		22: j max find the max procedure	0x00400028	0x0ac00000	lw \$r0, 0(\$t)		23: after max: lw \$r0, 0(\$a1) # load last element into \$r0	0x0040002c	0x0ac00000	sw \$r0, 0(\$t)		24: sw \$r0, 0(\$r0) # copy last element to max location	Labels <table border="1"> <thead> <tr> <th>Label</th> <th>Address</th> </tr> </thead> <tbody> <tr><td>mips2.asm</td><td>0x00400000</td></tr> <tr><td>main</td><td>0x00400008</td></tr> <tr><td>after sort</td><td>0x00400018</td></tr> <tr><td>end_main</td><td>0x00400020</td></tr> <tr><td>sort</td><td>0x00400020</td></tr> <tr><td>after_max</td><td>0x00400028</td></tr> <tr><td>done</td><td>0x00400038</td></tr> <tr><td>max</td><td>0x00400040</td></tr> <tr><td>loop</td><td>0x0040004c</td></tr> <tr><td>ret</td><td>0x0040006c</td></tr> <tr><td>A</td><td>0x10010000</td></tr> </tbody> </table>		Label	Address	mips2.asm	0x00400000	main	0x00400008	after sort	0x00400018	end_main	0x00400020	sort	0x00400020	after_max	0x00400028	done	0x00400038	max	0x00400040	loop	0x0040004c	ret	0x0040006c	A	0x10010000										
Bkpt	Address	Code	Basic	Source																																																																																																				
	0x00400000	0x3e010001	lui \$t, 4097																																																																																																					
0x00400004	0x34240000	ori \$t, \$t, 0		5: main: la \$a0, A # \$a0 = Address(A[0])																																																																																																				
0x00400008	0x3e010001	lui \$t, 4097		6: la \$a1, Aend																																																																																																				
0x0040000c	0x34250038	ori \$t, \$t, 156																																																																																																						
0x00400010	0x20a5fffc	addi \$t, \$t, -4		7: addi \$a1, \$a1, -4 # \$a1 = Address(A[n-1])																																																																																																				
0x00400014	0x08100008	j 0x00400020		8: j sort																																																																																																				
0x00400018	0x2402000a	addiu \$t, \$t, 10		9: after sort: li \$r0, 10 # exit																																																																																																				
0x0040001c	0x0000000b	syscall		10: syscall																																																																																																				
0x00400020	0x10850006	bqz \$t, \$t, 56		21: sort: bqz \$a0, \$a1, done # single element list is sorted																																																																																																				
0x00400024	0x08100010	j 0x00400040		22: j max find the max procedure																																																																																																				
0x00400028	0x0ac00000	lw \$r0, 0(\$t)		23: after max: lw \$r0, 0(\$a1) # load last element into \$r0																																																																																																				
0x0040002c	0x0ac00000	sw \$r0, 0(\$t)		24: sw \$r0, 0(\$r0) # copy last element to max location																																																																																																				
Label	Address																																																																																																							
mips2.asm	0x00400000																																																																																																							
main	0x00400008																																																																																																							
after sort	0x00400018																																																																																																							
end_main	0x00400020																																																																																																							
sort	0x00400020																																																																																																							
after_max	0x00400028																																																																																																							
done	0x00400038																																																																																																							
max	0x00400040																																																																																																							
loop	0x0040004c																																																																																																							
ret	0x0040006c																																																																																																							
A	0x10010000																																																																																																							
Data Segment <table border="1"> <thead> <tr> <th>Address</th> <th>Value (+0)</th> <th>Value (+4)</th> <th>Value (+8)</th> <th>Value (+c)</th> <th>Value (+10)</th> <th>Value (+14)</th> <th>Value (+18)</th> <th>Value (+1c)</th> </tr> </thead> <tbody> <tr><td>0x10010000</td><td>-2</td><td>1</td><td>3</td><td>5</td><td>5</td><td>5</td><td>6</td><td>6</td></tr> <tr><td>0x10010020</td><td>7</td><td>7</td><td>8</td><td>8</td><td>59</td><td>2019458</td><td>0</td><td>0</td></tr> <tr><td>0x10010040</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0x10010060</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0x10010080</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0x100100a0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0x100100c0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0x100100e0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0x10010100</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0x10010120</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </tbody> </table>				Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	0x10010000	-2	1	3	5	5	5	6	6	0x10010020	7	7	8	8	59	2019458	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	Name Number Value \$zero 0 \$at 268 \$v0 2 \$v1 3 \$a0 4 \$a1 5 \$a2 6 \$a3 7 \$t0 8 \$t1 9 \$t2 10 \$t3 11 \$t4 12 \$t5 13 \$t6 14 \$t7 15 \$a0 16 \$a1 17 \$a2 18 \$a3 19 \$a4 20 \$a5 21 \$a6 22 \$a7 23 \$t8 24 \$t9 25 \$t0 26 \$t1 27 \$fp 28 \$ap 2147 \$fp 30 \$ra 31 no 4 hi lo	
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)																																																																																																
0x10010000	-2	1	3	5	5	5	6	6																																																																																																
0x10010020	7	7	8	8	59	2019458	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																																																																																																
<input checked="" type="checkbox"/> Data <input checked="" type="checkbox"/> Text																																																																																																								
<input checked="" type="checkbox"/> Hexadecimal Addresses <input type="checkbox"/> Hexadecimal Values <input type="checkbox"/> ASCII																																																																																																								

- Giải thích:
 - Dòng 1-3: Khai báo mảng
 - Dòng 5-7: load địa chỉ của phần tử đầu, cuối của A vào a0,a1
 - Dòng 8: Thực hiện hàm sort
 - Dòng 9-11: Sau khi sort xong thì end chương trình
 - Hàm sort (Dòng 21-22): Nếu còn 1 phần tử thôi thì nhảy đến hàm max
 - Hàm done (Dòng 28): Nhảy đến after_sort
 - Hàm after_sort: Thoát khỏi chương trình
 - Hàm loop (Dòng 39-47):
 - Dòng 40: So sánh next = last thì thoát
 - Dòng 41,42: Lấy phần tử tiếp theo
 - Dòng 42: So sánh next và max
 - Dòng 43: Nếu $<=>$ tiếp tục loop
 - Dòng 45,46: Cập nhật next và max mới
 - Dòng 47: Tiếp tục loop

Assignment 3:

- Mã nguồn:

```
mips1.asm  mips2.asm  mips3.asm
1  # Bubble sort algorithm
2  .data
3      A: .word 20, 19, -5, 1, 8, 8, 20194588
4
5  .text
6  main:
7      la $a0, A          #$a0 = Address(A[0])
8      li $a1, 7          # length of array A: n
9      j  sort            #sort
10
11  after_sort:
12      li $v0, 10          #exit
13      syscall
14  end_main:
15
16  sort:
17      li $t0, 0           # index i của vòng lặp thu nhất
18
19  loop_1:
20      li $t1, 0           # index i của vòng lặp thu 2
21      add $t0, $t0, 1     # i++
22      sub $t2, $a1, $t0   # n - i
23
24  loop_2:
25      add $t3, $t1, $t1   #put 2i in $t1
26      add $t4, $t3, $t3   #put 4i in $t2
27      add $t4, $t4, $a0
28      lw $a2, 0($t4)      # load A[i]
29      lw $a3, 4($t4)      # Load A[i+1]
30  if:
31      slt $t7, $a2, $a3   # If A[i] < A[i+1]
32      bne $t7, $zero, end_if
33
34      # Swap
35      sw $a3, 0($t4)
36      sw $a2, 4($t4)
37
38  end_if:
39
40      add $t1, $t1, 1
41
42      slt $s1, $t1, $t2   # neu t1 < t2 return 1; else return 0
```

- Màn hình chạy:

- Giải thích:

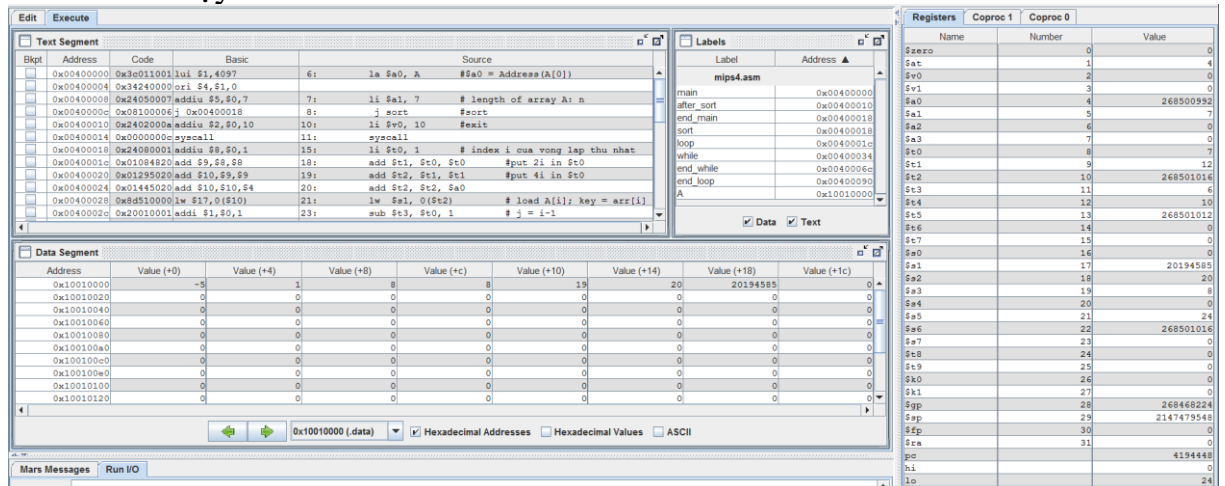
- Dòng 1-3: Khai báo mảng
- Dòng 7-8: load mảng vào a0 và gán a1 = số lượng phần tử
- Dòng 9: nhảy đến hàm sort
- Hàm sort(Dòng 16): Khởi tạo t0 (index i) = 0
- Chạy loop_1 (Dòng 19): Khởi tạo index j của vòng lặp thứ 2, j++, t2 = n - j
- loop_2 (Dòng 24): Lấy ra A[j], A[j+1]
- if (Dòng 30): Tiến hành so sánh A[j] và A[j+1], nếu bé hơn thì end_if, còn > thì phải swap
- end_if: Tăng j = i+j, so sánh với n-j để kiểm tra điều kiện lặp
- end_loop_2: So sánh i với n để kiểm tra điều kiện lặp

Assignment 4:

- Mã nguồn:

```
mips1.asm  mips2.asm  mips3.asm  mips4.asm
1  # Insertion sort algorithm
2  .data
3      A: .word 20, 19, -5, 1, 8, 8, 20194585
4  .text
5  main:
6      la $a0, A          #$a0 = Address(A[0])
7      li $a1, 7          # length of array A: n
8      j sort #sort
9  after_sort:
10     li $v0, 10          #exit
11     syscall
12 end_main:
13 sort:
14
15     li $t0, 1           # index i của vòng lặp thứ nhất
16
17 loop: # for i in range(1, len(arr))
18     add $t1, $t0, $t0    #put 2i in $t0
19     add $t2, $t1, $t1    #put 4i in $t0
20     add $t2, $t2, $a0
21     lw  $s1, 0($t2)      # load A[i]; key = arr[i]
22
23     sub $t3, $t0, 1      # j = i-1
24
25 while:
26
27     slt $t8, $t3, $zero   # if j < 0
28     bne $t8, $zero, end_while
29
30     add $t4, $t3, $t3     #put 2i in $t3
31     add $t5, $t4, $t4     #put 4i in $t3
32     add $t5, $t5, $a0
33     lw  $s2, 0($t5)      # arr[j]
34
35     slt $t9, $s1, $s2     # key < arr[j]
36     beq $t9, $zero, end_while
37
38     # Swap
39     lw  $s3, 4($t5)
40     sw  $s3, 0($t5)      # arr[j + 1] = arr[j]
41     sw  $s2, 4($t5)
42
```

- Màn hình chạy:



- Giải thích:

- Dòng 2,3: Khởi tạo mảng
- Dòng 6,7: load địa chỉ của A vào `a0` và khởi tạo độ dài của A vào `a1`
- Dòng 8: Nhảy đến hàm `sort`
- Hàm `sort` (Dòng 13-15): Khởi tạo `i = 0`
- Hàm `loop` (Dòng 17-23) : Lấy ra `A[i]` và cho `j = i-1`
- Hàm `while` (Dòng 25-44):
 - Dòng 27,28: So sánh `j` với 0, nếu `<0` thì `end_while`
 - Dòng 30-33: Lấy ra `A[j+1]`
 - Dòng 35-42: So sánh `A[j]` và `A[j+1]`, nếu `>` thì tiến hành `end_while`
 - đổi chỗ, giảm `j`
- `End_while` (dòng 45):
 - Tăng `j = j+1`
 - Lấy ra `A[j+1]` và `A[j]` để đổi chỗ
 - Kiểm tra điều kiện `I < n` để `end_loop`