

Họ và tên: *Trịnh Minh Đức*

MSSV: *20225813*

Assignment 1:

➔ Thực hiện bài mẫu 1

- Code:

```
C:\Users\84328\OneDrive\Tài liệu\Desktop\mars\duc1.asm
2  A: .word -2, 6, -1, 3, -2
3  .text
4  main:
5      la $a0,A
5      li $a1,5
7      j mspfx
3      nop
9  continue:
0  lock:
1      j lock
2      nop
3  end_of_main:
4  mspfx:
5      addi $v0,$zero,0 #initialize length in $v0 to 0
5      addi $v1,$zero,0 #initialize max sum in $v1 to 0
7      addi $t0,$zero,0 #initialize index i in $t0 to 0
3      addi $t1,$zero,0 #initialize running sum in $t1 to 0
9  loop:
0      add $t2,$t0,$t0 #put 2i in $t2
1      add $t2,$t2,$t2 #put 4i in $t2
2      add $t3,$t2,$a0 #put 4i+A (address of A[i]) in $t3
3      lw $t4,0($t3) #load A[i] from mem(t3) into $t4
```

```

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:

```

- Sự thay đổi giá trị thanh ghi:

	\$v0	\$v1	\$t0	\$t1	\$t3
Trước khi vòng lặp	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
Sau vòng lặp đầu tiên	0x00000000	0x00000000	0x00000001	0xffffffff	0x10010000
Sau vòng lặp thứ hai	0x00000002	0x00000004	0x00000002	0x00000004	0x10010004
Sau vòng lặp thứ ba	0x00000002	0x00000004	0x00000003	0x00000003	0x10010008
Sau vòng lặp thứ tư	0x00000004	0x00000006	0x00000004	0x00000006	0x1001000c
Sau vòng lặp thứ năm	0x00000004	0x00000006	0x00000005	0x00000004	0x10010010

- Kết quả:

\$v0	reserved for assembler	2	0x00000004
\$v1		3	0x00000006
\$a0		4	0x10010000
\$a1		5	0x00000005

- Giải thích: Chương trình này dùng để tìm ra dãy con bắt đầu từ vị trí đầu tiên đến vị trí N sao cho tổng của N số là lớn nhất và kết quả trả về là độ dài của dãy và tổng dãy.

- Khởi tạo chuỗi [-2, 6, -1, 3, -2]
- Khởi tạo \$a0 = địa chỉ đầu tiên chuỗi (A[0]), \$a1 = số phần tử chuỗi (= 5)

- Khởi tạo 2 thanh ghi \$v0 để lưu độ dài chuỗi, \$v1 lưu tổng lớn nhất
- \$t0 = i, Step = \$t2 = 4*i, \$t3 = địa chỉ của A[step] = A[0 + i]
- Load word \$t4 = A[i]
- \$t1 = sum = tổng các phần tử sau mỗi bước lặp = \$t1 + A[i]
- So sánh \$t1 là tổng hiện thời với \$v1 (tổng lớn nhất, ban đầu được gán bằng 0), nếu \$t1 > \$v1 thì thực hiện cập nhật lại giá trị của \$v1 (= \$t1) và \$v0 (= i + 1), sau đó sẽ tăng i lên, nếu i < n tiếp tục thực hiện vòng lặp, ngược lại sẽ kết thúc và kết luận độ dài dãy có tổng lớn nhất và tổng lớn nhất. Như ở trên ta thu được \$v0 = 4 và \$v1 = 6

- Nhân xét: Chương trình cho ra kết quả đúng với tính toán theo lý thuyết.

➔ Thực hiện nhập mảng có N phần tử và in ra độ dài và tổng của dãy có tổng tính từ phần tử đầu tiên đến N lớn nhất.

- Code:

.data

A: .word

Message1: .ascii "Nhap so luong phan tu: "

Message2: .ascii "Nhap so: "

Message3: .ascii "Do dai cua mang co tong lon nhat la: "

Message4: .ascii "Tong lon nhat la: "

Newline: .ascii "\n"

.text

input_number:

Nhap so luong phan tu:

li \$v0, 4

la \$a0, Message1

syscall

li \$v0, 5

syscall

move \$a1, \$v0 # Gan N vao \$a1

sub \$a1, \$a1, 1 # Tru \$a1 di 1

la \$a0, A

addi \$s0, \$a0, 0 # Gan dia chi mang A vao \$s0

input_array:

Nhap cac phan tu cua mang:

bgt \$t0, \$a1, main

Neu \$t0 > \$a1 thi ket thuc nhap phan tu

li \$v0, 4

la \$a0, Message2

syscall

li \$v0, 5

syscall

move \$t1, \$v0

sw \$t1, 0(\$s0) # Gan cac gia tri vua nhap vao mang A

```
addi $s0, $s0, 4
```

```
# Tro den dia chi cua phan tiep theo trong mang
```

```
addi $t0, $t0, 1
```

```
j input_array
```

```
main:
```

```
la $a0, A
```

```
j mspfx
```

```
nop
```

```
continue:
```

```
# In ra output
```

```
addi $s0, $v0, 0
```

```
addi $s1, $v1, 0
```

```
li $v0, 4
```

```
la $a0, Message3
```

```
syscall
```

```
li $v0, 1
```

```
la $a0, 0($s0)
```

```
syscall
```

```
li $v0, 4
```

la \$a0, Newline

syscall

li \$v0, 4

la \$a0, Message4

syscall

li \$v0, 1

la \$a0, 0(\$s1)

syscall

lock:

li \$v0, 10

syscall

end_of_main:

mfpfx:

addi \$v0, \$zero, 0 # max_length = 0

addi \$v1, \$zero, 0 # max_sum = 0

addi \$t0, \$zero, 0 # i = 0

addi \$t1, \$zero, 0 # sum = 0

loop:

```
add    $t2, $t0, $t0      # 2*i
add    $t2, $t2, $t2      # 4*i
add    $t3, $t2, $a0      # $t3 = (address of A) + 4*i
lw     $t4, 0($t3)        # $t4 = A[i]
add    $t1, $t1, $t4      # sum += A[i]
slt    $t5, $v1, $t1      # if max_sum < sum
bne    $t5, $zero, mdfy   # j mdfy
j      test
```

mdfy:

```
addi   $v0, $t0, 1        # max_length = i + 1
addi   $v1, $t1, 0        # max_sum = sum
```

test:

```
addi   $t0, $t0, 1        # i++
sle    $t5, $t0, $a1      # if i <= n thi loop
bne    $t5, $zero, loop
```

done:

```
j      continue
```

mfpfx_end:

- Kết quả thu được:

```

Nhập số lượng phần tử: 5
Nhập số: -2
ear  Nhập số: 6
      Nhập số: -1
      Nhập số: 3
      Nhập số: -2
      Độ dài của mảng có tổng lớn nhất là: 4
      Tổng lớn nhất là: 6

```

➔ Kết quả thu được hoàn toàn đúng theo lý thuyết.

Assignment 2:

➔ Thực hiện bài mẫu 2

```

1  .data
2  A: .word 7, -2, 5, 1, 5,6,7,3,6,8,8,59,5
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  sort:      beq $a0,$a1,done #single element list is sorted
13             j max #call the max procedure
14  after_max: lw $t0,0($a1) #load last element into $t0
15             sw $t0,0($v0) #copy last element to max location
16             sw $v1,0($a1) #copy max value to last element
17             addi $a1,$a1,-4 #decrement pointer to last element
18             j sort #repeat sort for smaller list
19  done:      j after_sort
20  max:
21            addi $v0,$a0,0 #init max pointer to first element
22            lw $v1,0($v0) #init max value to first value
23            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

```

- Trước khi sắp xếp:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	7	-2	5	1	5
0x10010020	6	8	8	59	5
0x10010040	0	0	0	0	0
0x10010060	n	n	n	n	n

- Sau khi sắp xếp:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	-2	1	3	5	
0x10010020	7	7	8	8	
0x10010040	0	0	0	0	

- Giải thích:

- Chương trình sắp xếp chuỗi bằng cách sử dụng một biến (thanh ghi) để lưu **giá trị** và một con trỏ (thanh ghi) để lưu **vị trí** của phần tử **giá trị** lớn nhất chuỗi, sau khi tìm được giá trị lớn nhất ta hoán đổi giá trị của số lớn nhất và số cuối cùng của chuỗi
- Khởi tạo chuỗi [7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5]
- Khởi tạo \$a0 = địa chỉ đầu tiên chuỗi (A[0]), \$a1 = địa chỉ cuối chuỗi (A[n-1])
- Chạy hàm sort nếu địa chỉ của A[i] = A[n-1] thì chuỗi đã sắp xếp xong và nhảy sang hàm done và kết thúc chương trình thu được chuỗi tăng dần ngược lại ta thực hiện hàm after_sort để in ra các chuỗi sắp xếp
- Hàm max để tìm ra con trỏ và giá trị lớn nhất có \$v0 = địa chỉ của A[i] và \$v1 = A[i], khởi tạo i = 0
- Hàm loop so sánh \$t0 và \$a1 <=> so sánh địa chỉ của A[i] và A[n-1]. Nếu bằng thì ta thực hiện hàm ret nhảy đến hàm after_max để hoán đổi giá trị của số lớn nhất và số ở vị trí cuối cùng bằng cách lưu giá trị của địa chỉ A[n-1] vào \$t0, sau

đó truyền giá trị \$t0 và địa chỉ ptr, cuối cùng truyền giá trị max và địa chỉ của A[n-1] và giảm n đi 1 để chuyển vị trí cuối cùng sang n-2

- Ngược lại thì lưu giá trị của A[i + 1] vào \$t1 và so sánh \$t1 và \$v1. Nếu \$t1 nhỏ hơn \$v1 (giá trị tiếp theo nhỏ hơn giá trị ban đầu) thì thực hiện lại vòng loop còn ngược lại sẽ cập nhật lại \$v0 = địa chỉ của A[i + 1] và \$v1 = A[i + 1] sau đó cứ loop lại cho đến khi i = n

- Nhận xét: Chương trình đã cho ra kết quả đúng là một dãy đã được sắp xếp theo thứ tự từ bé đến lớn.

➔ Thực hiện nhập mảng có N phần tử và sắp xếp mảng đó theo thứ tự tăng dần bằng sắp xếp lựa chọn:

- Code:

.data

A: .word

Message: .ascii "Nhap so luong phan tu cua mang: "

Message0: .ascii "Nhap so: "

Message1: .ascii "Mang sau khi thuc hien selection sort la:"

Message2: .ascii ", "

Newline: .ascii "\n"

.text

input_number:

Nhap N la so luong phan tu:

li \$v0, 4

la \$a0, Message

syscall

li \$v0, 5

syscall

move \$a1, \$v0

sub \$a1, \$a1, 1

addi \$s0, \$s0, 0

la \$a0, A

addi \$s0, \$a0, 0

input_array:

Nhap cac gia tri cua mang:

bgt \$t0, \$a1, end_input

li \$v0, 4

la \$a0, Message0

syscall

li \$v0, 5

syscall

move \$t1, \$v0

sw \$t1, 0(\$s0)

addi \$s0, \$s0, 4

addi \$t0, \$t0, 1

j input_array

end_input:

main:

```
la    $a0, A        # address of A[0]

mul   $s3, $a1, 4    # $s3 = (n - 1) * 4

add   $t6, $a0, $s3  # $t6 = address of A[0] + (n - 1) * 4 = address of A[n - 1]

addi  $t6, $t6, 4    # $t6 = address of A[n]

la    $a1, 0($t6)

sub   $a1, $a1, 4    # $a1 = address of A[n - 1]

j     max
```

end_main:

```
li    $v0, 10

syscall
```

sort:

```
bgt   $a0, $a1, done    # if i = n => done

j     after_sort
```

after_max:

```
lw    $t0, 0($a1)      # $t0 = value of address A[n-1]

sw    $t0, 0($v0)      # value of address ptr = $t0
```

```

        sw    $v1, 0($a1)        # value of A[n-1] = max

        addi  $a1, $a1, -4        # n -= 4

        j     sort

done:   j     end_main

max:

        la    $a0, A

        addi  $v0, $a0, 0        # ptr = address of A[0]

        lw    $v1, 0($v0)        # max = A[0]

        addi  $t0, $a0, 0        # i = 0

loop:

        beq   $t0, $a1, ret       # if i = n ret

        addi  $t0, $t0, 4        # i += 4

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

        slt   $t2, $t1, $v1      # temp < max

        bne   $t2, $zero, loop    # if temp < max loop

        addi  $v0, $t0, 0        # ptr = address of A[i]

        addi  $v1, $t1, 0        # max = A[i]

        j     loop

after_sort:

```

Print message1

li \$v0, 4

la \$a0, Message1

syscall

Print new line

la \$a0, Newline

syscall

print number of array

la \$s0, A

la \$s1, 0(\$t6)

lw \$s2, 0(\$s0)

li \$v0, 1

la \$a0, 0(\$s2)

syscall

addi \$t3, \$zero, 0 # i = 0

print_array:

Print Message2

addi \$t3, \$t3, 4 # i += 4

add \$t4, \$s0, \$t3 # \$t1 = address of A[0] + 4*i

```

lw    $t5, 0($t4)      # x = A[i]

beq   $t4, $s1, end     # if i > (n-1) end

li    $v0, 4

la    $a0, Message2

syscall

```

```

# Print A[i]

li    $v0, 1

la    $a0, 0($t5)

syscall

j     print_array

```

end:

```

li    $v0, 4

la    $a0, Newline

syscall

j max

```

ret:

```

j     after_max

```

- Kết quả:

```

Nhap so luong phan tu cua mang: 7
Nhap so: -13
Nhap so: 5
Nhap so: 89
Nhap so: -29
Nhap so: 10
Nhap so: 47
Nhap so: 79
Mang sau khi thuc hien selection sort la:
-13, 5, 79, -29, 10, 47, 89

Mang sau khi thuc hien selection sort la:
-13, 5, 10, -29, 47, 79, 89
Mang sau khi thuc hien selection sort la:
-13, 5, -29, 10, 47, 79, 89
Mang sau khi thuc hien selection sort la:
-13, -29, 5, 10, 47, 79, 89
Mang sau khi thuc hien selection sort la:
-29, -13, 5, 10, 47, 79, 89

-- program is finished running --

```

➔ Chương trình cho ra kết quả hoàn toàn chính xác.

Assignment 3

Thuật toán bubble sort

```

void BubbleSort(int a[], int n){
    int temp; // biến tạm temp
    for (int i = 0; i < n; i++){
        for (int j = i + 1; j < n; j++){
            if (a[j] > a[j+1]){
                temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    }
}

```

Code :

.data

A: .word

Message: .asciiz "Nhap so luong phan tu N: "

Message0: .asciiz "Nhap so: "

Message1: .asciiz "Mang sau khi thuc hien bubble sort la: "

Message2: .asciiz ", "

Newline: .asciiz "\n"

.text

input_number:

Nhap N la so luong phan tu:

li \$v0, 4

la \$a0, Message

syscall

li \$v0, 5

syscall

move \$a1, \$v0

sub \$a1, \$a1, 1

addi \$s0, \$s0, 0

la \$a0, A

addi \$s0, \$a0, 0

input_array:

Nhap cac gia tri cua mang:

bgt \$t0, \$a1, end_input

li \$v0, 4

la \$a0, Message0

syscall

```

li    $v0, 5

syscall

move  $t1, $v0

sw    $t1, 0($s0)

addi  $s0, $s0, 4

addi  $t0, $t0, 1

j     input_array

```

end_input:

main:

```

# Khoi tao cac gia tri

la    $a0, A      # address of A[0]

addi  $a2, $a0, 4  # address of A[1]

mul   $s3, $a1, 4  # $s3 = (n - 1) * 4

add   $a1, $s3, $a0 # $a1 = (n - 1) * 4 + A[0] = address of A[n - 1]

addi  $t6, $a1, 4  # $t6 = address of A[n]

addi  $t0, $zero, 0 # $t0 to count the loop

j     bubble_sort

```

swap:

```

lw    $t2, 0($a2)  # $t2 = A[i]

sw    $t2, 0($t1)  # A[j] = $t2 = A[i]

sw    $v0, 0($a2)  # A[i] = *($v0) = A[j]

```

j continue

reset:

la \$a0, A

addi \$t0, \$zero, 0 # j = 0

addi \$a2, \$a2, 4 # i += 4

j bubble_sort

bubble_sort:

bgt \$a2, \$a1, print_sort # if i > (n-1) end

add \$t1, \$a0, \$t0 # \$t1 = address of A[0] + 4*j = A[j]

beq \$t1, \$a2, reset # if j = i print

lw \$v0, 0(\$t1) # \$v0 = A[j]

lw \$v1, 0(\$a2) # \$v1 = A[i]

blt \$v1, \$v0, swap # if A[i] > A[j] swap

continue:

addi \$t0, \$t0, 4 # j += 4

j bubble_sort

print_sort:

Print message1

li \$v0, 4

la \$a0, Message1

syscall

Print new line

la \$a0, Newline

syscall

Print number of array

la \$s0, A

la \$s1, 0(\$t6)

lw \$s2, 0(\$s0)

li \$v0, 1

la \$a0, 0(\$s2)

syscall

addi \$t3, \$zero, 0 # i = 0

print_array:

addi \$t3, \$t3, 4 # i += 4

add \$t4, \$s0, \$t3 # \$t4 = address of A[0] + 4*i = A[i]

lw \$t5, 0(\$t4) # x = A[i]

beq \$t4, \$s1, end # if i > (n-1) end

li \$v0, 4

la \$a0, Message2

```
syscall                                # Print Message2
```

```
li      $v0, 1
```

```
la      $a0, 0($t5)
```

```
syscall                                # Print A[i]
```

```
j      print_array
```

end:

```
li      $v0, 4
```

```
la      $a0, Newline
```

```
syscall
```

end_main:

```
# Exit
```

```
li      $v0, 10
```

```
syscall
```

➔ Kết quả là

Messages | Run

Clear

Nhap so luong phan tu N: 6

Nhap so: -13

Nhap so: 34

Nhap so: 25


Nhap so: 45

Nhap so: 89

Nhap so: -45

Mang sau khi thuc hien bubble sort la:

-45, -13, 25, 34, 45, 89

 75°F