**BÁO CÁO THỰC HÀNH**

**IT3280 – 156788 – THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

**NỘI DUNG**

**Bài 6. Mảng và con trỏ**

|  |  |
| --- | --- |
| **Họ và tên** | Nguyễn Minh Quân |
| **Mã số sinh viên** | 20235816 |

**Assignment 1**

**Tạo project thực hiện chương trình trong Home Assigment 1. Khởi tạo bộ giá trị mới cho mảng, dịch và nạp lên mô phỏng. Chạy chương trình từng bước một và quan sát sự thay đổi các thanh ghi để kiểm nghiệm chương trình hoạt động đúng với thuật toán**

**Chương trình thực hiện:**

.data

A: .word -5, -2, 10,-6

.text

main:

la a0, A

li a1, 4

j mspfx

continue:

exit:

li a7, 10

ecall

end\_of\_main:

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

# Procedure mspfx

# @brief find the maximum-sum prefix in a list of integers

# @param[in] a0 the base address of this list(A) needs to be processed

# @param[in] a1 the number of elements in list(A)

# @param[out] s0 the length of sub-array of A in which max sum reachs.

# @param[out] s1 the max sum of a certain sub-array

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

# Procedure mspfx

# Function: find the maximum-sum prefix in a list of integers

# The base address of this list(A) in a0 and the number of

# elements is stored in a1

mspfx:

li s0, 0 # initialize length of prefix-sum in s0 to 0

li s1, 0x80000000 # initialize max prefix-sum in s1 to smallest int

li t0, 0 # initialize index for loop i in t0 to 0

li t1, 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

n elements

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

blt s1, t1, mdfy # if (s1 < t1) modify results

j next

mdfy:

addi s0, t0, 1 # new max-sum prefix has length i+1

addi s1, t1, 0 # new max sum is the running sum

next:

addi t0, t0, 1 # advance the index i

blt t0, a1, loop # if (i<n) repeat

done:

j continue

mspfx\_end:

**Chạy chương trình từng bước một và quan sát sự thay đổi các thanh ghi để kiểm nghiệm chương trình hoạt động đúng với thuật toán**

1. Khởi tạo

A white background with black dots

AI-generated content may be incorrect.

A close-up of a computer screen

AI-generated content may be incorrect.

Kết quả trên các thanh ghi:













1. Vòng lặp chính

A screenshot of a computer code

AI-generated content may be incorrect.

* Step 1: i=0

loop:

add t2, t0, t0 # t2 = t0 + t0 = 0 + 0 = 0

add t2, t2, t2 # t2 = t2 + t2 = 0 + 0 = 0

add t3, t2, a0 # t3 = t2 + a0 = 0 + 0x1000 = 0x1000

lw t4, 0(t3) # t4 = A[0] = -5

add t1, t1, t4 # t1 = t1 + t4 = 0 + (-5) = -5

blt s1, t1, mdfy # So sánh s1 (0x80000000) < t1 (-5) → Nhảy tới mdfy

mdfy:

addi s0, t0, 1 # length s0 = t0 + 1 = 0+1 = 1

addi s1, t1, 0 # s1 = t1 = -5

next:

addi t0, t0, 1 # t0 = t0 + 1 = 0 + 1 = 1

blt t0, a1, loop # So sánh t0 (1) < a1 (6) → Nhảy đến loop

Kết quả:









* Step 2: i =1

loop:

add t2, t0, t0 # t2 = t0 + t0 = 1 + 1 = 2

add t2, t2, t2 # t2 = t2 + t2 = 2 + 2 = 4

add t3, t2, a0 # t3 = t2 + a0 = 4 + 0x1000 = 0x1004

lw t4, 0(t3) # t4 = A[1] = -2

add t1, t1, t4 # t1 = t1 + t4 = -5 - 2 = -7

blt s1, t1, mdfy # So sánh s1 (-5) > t1 (-7) → Không nhảy

j next # Nhảy đến nhãn next

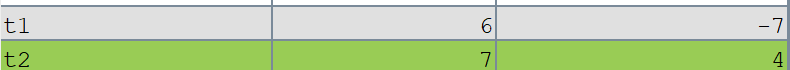
next:

addi t0, t0, 1 # t0 = t0 + 1 = 1 + 1 = 2

blt t0, a1, loop # So sánh t0 (2) < a1 (6) → Nhảy đến loop

Kết quả:









* Step 3: i = 2

loop:

add t2, t0, t0 # t2 = t0 + t0 = 2 + 2 = 4

add t2, t2, t2 # t2 = t2 + t2 = 4 + 4 = 8

add t3, t2, a0 # t3 = t2 + a0 = 8 + 0x1000 = 0x1008

lw t4, 0(t3) # t4 = A[2] = 10

add t1, t1, t4 # t1 = t1 + t4 = -7 + 10 = 3

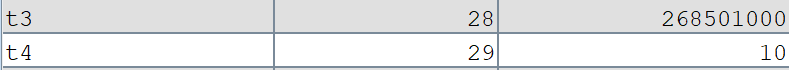
blt s1, t1, mdfy # So sánh s1 =-5 < t1 (3) → Nhảy tới mdfy

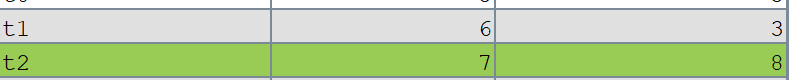
mdfy:

addi s0, t0, 1 # s0 = t0 + 1 = 2+1 = 3

addi s1, t1, 0 # s1 = t1 = 3

Kết quả:









* Step 4: i=3

loop:

add t2, t0, t0 # t2 = t0 + t0 = 3 + 3 = 6

add t2, t2, t2 # t2 = t2 + t2 = 6 + 6 = 12

add t3, t2, a0 # t3 = t2 + a0 = 12 + 0x1000 = 0x100C

lw t4, 0(t3) # t4 = A[3] = -6

add t1, t1, t4 # t1 = t1 + t4 = 3 - 6 = -3

blt s1, t1, mdfy # So sánh s1 (3) > t1 (-3) → Không nhảy

j next # Nhảy đến nhãn next

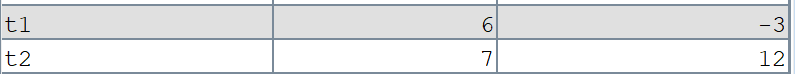
next:

addi t0, t0, 1 # t0 = 3 + 1 =4

blt t0, a1, loop # t0 =4 = a1 -> Stop

Kết quả:









**Assignment 2**

**Tạo mới một project thực hiện chương trình trong Home Assigment 2. Khởi tạo bộ giá trị mới cho mảng, dịch và nạp lên mô phỏng. Chạy chương trình từng bước một và quan sát sự thay đổi các thanh ghi để kiểm nghiệm chương trình hoạt động đúng với thuật toán. Viết thêm chương trình con để in ra mảng sau mỗi lượt sắp xếp**

Home Assignment 2

Thuật toán sắp xếp lựa chọn (selection sort). Một mảng số nguyên gồm n phần tử có thể được sắp xếp theo thứ tự tăng dần như sau. Tìm phần tử có giá trị lớn nhất trong danh sách và đổi chỗ nó với phần tử cuối cùng trong dãy. Phần tử cuối cùng đã được đặt đúng vị trí. Tiếp tục thực hiện các bước trên với n – 1 phần tử chưa được sắp xếp cho đến khi chỉ còn lại 1 phần tử. Khi đó thuật toán kết thúc, mảng được sắp xếp theo thứ tự tăng dần. Chương trình dưới đây minh họa việc thực hiện thuật toán sắp xếp lựa chọn bằng phương pháp truy nhập kiểu con trỏ. Hãy đọc kỹ và hiểu cách thực hiện của chương trình.

**Chương trình thực hiện**

.data

A: .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5

Aend: .word

.text

main:

la a0, A # a0 = address(A[0])

la a1, Aend # a1 = address(Aend)

mv s2, a0 # Lưu địa chỉ ban đầu của mảng vào s2

addi a1, a1, -4 # a1 = address(A[n-1])

li s6, 13 # Số phần tử của mảng (13)

j sort # Nhảy đến thủ tục sort để sắp xếp

after\_sort:

li a7, 10 # Chuẩn bị gọi hệ thống để kết thúc chương trình

ecall # Gọi hệ thống để kết thúc chương trình

end\_main:

sort:

beq a0, a1, done # Nếu a0 == a1 (mảng chỉ có 1 phần tử), nhảy đến nhãn done

j max # Gọi thủ tục max để tìm phần tử lớn nhất trong phần chưa sắp xếp

after\_max:

lw t0, 0(a1) # Load giá trị của phần tử cuối cùng vào t0

sw t0, 0(s0) # Ghi giá trị của phần tử cuối cùng vào vị trí của phần tử lớn nhất

sw s1, 0(a1) # Ghi giá trị lớn nhất vào vị trí của phần tử cuối cùng

addi a1, a1, -4 # Di chuyển con trỏ a1 sang trái (giảm kích thước phần chưa sắp xếp)

jal print\_array # Gọi thủ tục in mảng

j sort # Lặp lại thủ tục sort

done:

j after\_sort

max:

addi s0, a0, 0 # Khởi tạo con trỏ max (s0) trỏ đến phần tử đầu tiên

lw s1, 0(s0) # Khởi tạo giá trị max (s1) bằng giá trị của phần tử đầu tiên

addi t0, a0, 0 # Khởi tạo con trỏ next (t0) trỏ đến phần tử đầu tiên

loops:

beq t0, a1, ret # Nếu t0 == a1 (đã duyệt hết mảng), nhảy đến nhãn ret

addi t0, t0, 4 # Di chuyển con trỏ t0 sang phải (phần tử tiếp theo)

lw t1, 0(t0) # Load giá trị của phần tử tiếp theo vào t1

blt t1, s1, loops # Nếu t1 < s1 (phần tử tiếp theo nhỏ hơn max), lặp lại

addi s0, t0, 0 # Cập nhật con trỏ max (s0) trỏ đến phần tử tiếp theo

addi s1, t1, 0 # Cập nhật giá trị max (s1) bằng giá trị của phần tử tiếp theo

j loops # Lặp lại vòng lặp

ret:

j after\_max

print\_array:

addi sp, sp, -8 # Cấp phát không gian trên stack

sw ra, 0(sp) # Lưu giá trị của ra vào stack

sw a0, 4(sp) # Lưu giá trị của a0 vào stack

li s5, 0 # Khởi tạo biến đếm s5 = 0

print\_loop:

beq s5, s6, print\_done # Nếu s5 == s6 (đã duyệt hết mảng), kết thúc in

slli s7, s5, 2 # s7 = s5 \* 4 (offset của phần tử thứ s5)

add s7, s7, s2 # s7 = địa chỉ của phần tử thứ s5 (sử dụng s2 thay vì a0)

lw s8, 0(s7) # Load giá trị của phần tử thứ s5 vào s8

# In giá trị

li a7, 1 # Chuẩn bị in số nguyên

mv a0, s8 # Đặt giá trị cần in vào a0

ecall # Gọi hệ thống để in giá trị

# In khoảng trắng

li a7, 11 # Chuẩn bị in ký tự

li a0, 32 # Đặt ký tự khoảng trắng (ASCII 32) vào a0

ecall # Gọi hệ thống để in ký tự

addi s5, s5, 1 # Tăng biến đếm s5 lên 1

j print\_loop # Lặp lại vòng lặp

print\_done:

# In ký tự xuống dòng

li a7, 11 # Chuẩn bị in ký tự

li a0, 10 # Đặt ký tự xuống dòng (ASCII 10) vào a0

ecall # Gọi hệ thống để in ký tự

lw ra, 0(sp) # Khôi phục giá trị của ra từ stack

lw a0, 4(sp) # Khôi phục giá trị của a0 từ stack

addi sp, sp, 8 # Giải phóng không gian trên stack

jr ra # Trở về thủ tục gọi

Kết quả thu được:

**A number grid with black and white numbers

AI-generated content may be incorrect.**

**A number grid with black and blue lines

Description automatically generated with medium confidence**

**Khởi tạo giá trị**

la a0, A # Load địa chỉ A[0] vào a0

la a1, Aend

addi a1, a1, -4 # a1 trỏ vào phần tử cuối A[n-1]

j sort # Nhảy vào phần sắp xếp







**Vòng lặp sắp xếp (Selection Sort)**

sort:

beq a0, a1, done # Nếu chỉ còn một phần tử, dừng lại

j max # Gọi max để tìm phần tử lớn nhất

**Tìm phần tử lớn nhất (Hàm max)**

max:

addi s0, a0, 0 # s0 trỏ vào phần tử đầu (vị trí max ban đầu)

lw s1, 0(s0) # s1 = giá trị max ban đầu

addi t0, a0, 0 # t0 trỏ vào phần tử đầu (vị trí đang xét)







**Lặp qua mảng để tìm max**

loop:

beq t0, a1, ret # Nếu đã xét đến cuối mảng, quay về hàm gọi

addi t0, t0, 4 # Tiến đến phần tử tiếp theo

lw t1, 0(t0) # Lấy giá trị phần tử tiếp theo vào t1

blt t1, s1, loop # Nếu t1 < max hiện tại, tiếp tục vòng lặp

addi s0, t0, 0 # Nếu t1 > max hiện tại, cập nhật vị trí max

addi s1, t1, 0 # Cập nhật giá trị max

j loop







Lặp qua mảng, nếu tìm thấy phần tử **lớn hơn giá trị max hiện tại**, cập nhật s0 và s1.

**Ví dụ**





Khi hết vòng lặp, s0 chứa địa chỉ phần tử lớn nhất.

**Hoán đổi phần tử lớn nhất với phần tử cuối**

after\_max:

lw t0, 0(a1) # Lấy giá trị cuối mảng vào t0

sw t0, 0(s0) # Gán giá trị cuối vào vị trí max

sw s1, 0(a1) # Gán giá trị max vào cuối mảng

addi a1, a1, -4 # Giảm kích thước mảng (co ngắn lại)

j sort # Tiếp tục sắp xếp phần còn lại





* **Hoán đổi** phần tử lớn nhất với phần tử cuối cùng của đoạn hiện tại.
* **Thu hẹp kích thước mảng** bằng cách giảm a1.
* **Lặp lại quá trình** cho đến khi mảng được sắp xếp hoàn toàn.

**Kết thúc chương trình**

done:

j after\_sort

after\_sort:

li a7, 10

ecall # Thoát chương trình

**Kết quả trong ví dụ trên:**

|  |  |
| --- | --- |
| **Bước** | **Mảng A sau khi sắp xếp từng phần** |
| **0** | 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 5, 59 |
| **1** | |  | | --- | | 7, -2, 5, 1, 5, 6, 7, 3, 6, 5, 8, 8, 59 |  |  | | --- | |  | |
| **2** | |  |  |  | | --- | --- | --- | | |  | | --- | | 7, -2, 5, 1, 5, 6, 7, 3, 6, 5, 8, 8, 59 | |  | |  |  | | --- | |  | |
| **3** | |  |  |  | | --- | --- | --- | | |  | | --- | | 5, -2, 5, 1, 5, 6, 7, 3, 6, 7, 8, 8, 59 | |  | |  |  | | --- | |  | |
| **4** | |  |  |  | | --- | --- | --- | | |  | | --- | | 5, -2, 5, 1, 5, 6, 6, 3, 7, 7, 8, 8, 59 | |  | | |
| **5** | |  |  |  | | --- | --- | --- | | |  | | --- | | 5, -2, 5, 1, 5, 3, 6, 6, 7, 7, 8, 8, 59 | |  | | |
| **6** | |  | | --- | | 5, -2, 5, 1, 5, 3, 6, 6, 7, 7, 8, 8, 59 | |  | |
| **7** | |  | | --- | | 3, -2, 5, 1, 5, 5, 6, 6, 7, 7, 8, 8, 59 | |  | |
| **8** | |  | | --- | | 3, -2, 5, 1, 5, 5, 6, 6, 7, 7, 8, 8, 59 | |  | |
| **9** | |  | | --- | | 3, -2, 1, 5, 5, 5, 6, 6, 7, 7, 8, 8, 59 | |  | |
| **10** | |  | | --- | | 1, -2, 3, 5, 5, 5, 6, 6, 7, 7, 8, 8, 59 | |  | |
| **11** | |  | | --- | | -2, 1, 3, 5, 5, 5, 6, 6, 7, 7, 8, 8, 59 | |

**Kết quả cuối cùng (mảng sau khi sắp xếp tăng dần):**

|  |
| --- |
| -2, 1, 3, 5, 5, 5, 6, 6, 7, 7, 8, 8, 59 |

**Kết luận: Chương trình hoạt động đúng với thuật toán.**

**Assignment 3**

Viết chương trình thực hiện thuật toán sắp xếp nổi bọt (bubble sort).

**Chương trình thực hiện:**

.data

A: .word 1, -3, 4, 5, 2, -8, 8, 7

Aend: .word

.text

main:

la a0, A

la a1, Aend

addi a1, a1, -4

j bubble\_sort

after\_sort:

li a7, 10

ecall

bubble\_sort:

la t0, A

addi t1, a1, 4

outer\_loop:

add t2, t0, zero

addi a3, t1, -4

li t3, 0

inner\_loop:

bge t2, a3, end\_inner

lw t4, 0(t2)

lw t5, 4(t2)

ble t4, t5, no\_swap

sw t5, 0(t2)

sw t4, 4(t2)

li t3, 1

no\_swap:

addi t2, t2, 4

j inner\_loop

end\_inner:

jal print\_array

beqz t3, end\_sort

addi t1, t1, -4

b outer\_loop

end\_sort:

j after\_sort

print\_array:

addi sp, sp, -20

sw t0, 0(sp)

sw t1, 4(sp)

sw t2, 8(sp)

sw a0, 12(sp)

sw a7, 16(sp)

la t0, A

la t1, Aend

addi t1, t1, -4

print\_loop:

bgt t0, t1, print\_done

lw a0, 0(t0)

li a7, 1

ecall

li a0, 32

li a7, 11

ecall

addi t0, t0, 4

j print\_loop

print\_done:

li a0, 10

li a7, 11

ecall

lw t0, 0(sp)

lw t1, 4(sp)

lw t2, 8(sp)

lw a0, 12(sp)

lw a7, 16(sp)

addi sp, sp, 20

jr ra

**Kết quả thu được:**

**A number on a white background

Description automatically generated**

**Assignment 4**

**Viết chương trình thực hiện thuật toán sắp xếp chèn (insertion sort).**

Chương trình thực hiện:

.data

A: .word -3, 4, 1, 2, 7, 9, 10, -4, 11, 2, 3

Aend: .word

.text

main:

la a0, A

la a1, Aend

addi a1, a1, -4

j insertion\_sort

after\_sort:

li a7, 10

ecall

insertion\_sort:

la t0, A

addi t1, t0, 4

outer\_loop:

bgt t1, a1, end\_sort

lw t2, 0(t1)

add t3, t1, zero

inner\_loop:

addi t4, t3, -4

blt t4, t0, insert\_done

lw t5, 0(t4)

ble t5, t2, insert\_done

sw t5, 4(t4)

addi t3, t3, -4

j inner\_loop

insert\_done:

sw t2, 0(t3)

jal print\_array

addi t1, t1, 4

j outer\_loop

end\_sort:

j after\_sort

print\_array:

addi sp, sp, -20

sw t0, 0(sp)

sw t1, 4(sp)

sw t2, 8(sp)

sw t3, 12(sp)

sw a7, 16(sp)

la t0, A

la t1, Aend

addi t1, t1, -4

print\_loop:

bgt t0, t1, print\_done

lw a0, 0(t0)

li a7, 1

ecall

li a0, 32

li a7, 11

ecall

addi t0, t0, 4

j print\_loop

print\_done:

li a0, 10

li a7, 11

ecall

lw t0, 0(sp)

lw t1, 4(sp)

lw t2, 8(sp)

lw t3, 12(sp)

lw a7, 16(sp)

addi sp, sp, 20

jr ra

**Kết quả thu được:**

**A grid of numbers

Description automatically generated**

**A number on a white background

Description automatically generated**