Thực hành kiến trúc máy tính

Báo cáo thực hành

Bài 6. Mảng và con trỏ

|  |  |
| --- | --- |
| Họ Tên | Lê Thành An |
| MSSV | 20235631 |

**ASSIGNMENT 1**

ĐOẠN MÃ :

.data

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

.text

main:

la a0, A

li a1, 8

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

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:

Kết quả:











**ASSIGNMENT 2 :**

CODE :

.data

A: .asciz "The sum of "

B: .asciz " and "

C: .asciz " is "

.text

li s0, 18 #s0=18

li s1, 20 #s1=20

add t1, s0, s1 #t1=s1+s0

li a7, 4 #a7=4 in ra chuỗi

la a0, A #a0 = địa chỉ của chuỗi A

ecall

li a7, 1 #a7=1 in ra số nguyên

mv a0, s0 #a0=s0

ecall

li a7, 4 #a7=4 in ra chuỗi

la a0, B #a0 = địa chỉ của chuỗi B

ecall

li a7, 1 #a7=1 in ra số nguyên

mv a0, s1 #a0=s1

ecall

li a7, 4 #a7=4 in ra chuỗi

la a0, C #a0 = địa chỉ của chuỗi C

ecall

li a7, 1 #a7=1 in ra số nguyên

mv a0, t1 #a0=t1

ecall

Kết quả:

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, đại số

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, số, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

**ASSIGNMENT 3 :**

# Laboratory Exercise 5, Home Assignment 2

.data

x: .space 32 # Chuỗi đích x, khởi tạo là buffer rỗng

y: .asciz "Hello" # Chuỗi nguồn y

.text

strcpy:

add s0, zero, zero # s0 = i = 0

L1:

add t1, s0, a1 # t1 = s0 + a1 = i + y[0] = address of y[i]

lb t2, 0(t1) # t2 = value at t1 = y[i]

add t3, s0, a0 # t3 = s0 + a0 = i + x[0] = address of x[i]

sb t2, 0(t3) # x[i]= t2 = y[i]

beq t2,zero,end\_of\_strcpy # if y[i] = 0, then exit

addi s0, s0, 1 # s0 = s0 + 1 <-> i = i + 1

j L1 # Ký tự tiếp theo

end\_of\_strcpy:

Giá trị các thanh ghi:

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, số, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

**ASSIGNMENT 4:**

**Code:**

.data

A:

.word 50, 12, -3, 0, 7, 98, -12, 45, 3, 27 # Bộ dữ liệu mảng mới (10 phần tử)

prompt: .asciz " "

space: .asciz " "

newline: .asciz "\n"

.text

.globl main

main:

# Khởi tạo:

# Chuỗi in đầu dòng cho mảng

# Khoảng trắng giữa các phần tử

# Xuống dòng

# s0 = địa chỉ đầu mảng, s1 = số phần tử (10)

la s0, A

li s1, 10

# s3 = chỉ số outer loop, bắt đầu từ 1 (vì phần tử A[0] được coi là đã sắp xếp)

li s3, 1

outer\_loop:

bge s3, s1, sort\_done # Nếu i >= n thì mảng đã được sắp xếp

# Lấy key = A[i]

slli t1, s3, 2 # t1 = i \* 4 (để tính địa chỉ của A[i])

add t2, s0, t1 # t2 = địa chỉ của A[i]

lw t3, 0(t2) # t3 = key (giá trị cần chèn)

addi s4, s3, -1 # s4 = j = i - 1

inner\_loop:

blt s4, zero, inner\_done # Nếu j < 0, thoát vòng lặp bên trong

slli t1, s4, 2 # t1 = j \* 4

add t2, s0, t1 # t2 = địa chỉ của A[j]

lw s2, 0(t2) # s2 = A[j]

ble s2, t3, inner\_done # Nếu A[j] <= key, kết thúc vòng lặp bên trong

# Dời A[j] sang phải: A[j+1] = A[j]

addi t1, s4, 1 # t1 = j + 1

slli t1, t1, 2 # t1 = (j+1) \* 4

add t2, s0, t1 # t2 = địa chỉ của A[j+1]

sw s2, 0(t2) # di chuyển giá trị A[j] sang phải

addi s4, s4, -1 # j = j - 1

j inner\_loop

inner\_done:

addi s4, s4, 1 # Vị trí chèn: j+1

slli t1, s4, 2 # t1 = (j+1) \* 4

add t2, s0, t1 # t2 = địa chỉ của A[j+1]

sw t3, 0(t2) # Chèn key vào A[j+1]

# In mảng sau mỗi lượt chèn

jal ra, printArray

addi s3, s3, 1

j outer\_loop

sort\_done:

# i = i + 1

# In mảng cuối cùng đã được sắp xếp

jal ra, printArray

li a7, 10

ecall

# Kết thúc chương trình

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

# Procedure printArray:

# In ra mảng theo định dạng "Array: <A[0]> <A[1]> ... <A[9]>\n"

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

printArray:

la a0, prompt

li a7, 4

ecall

la t0, A

li t1, 10

print\_loop:

# In chuỗi "Array: "

# t0 = địa chỉ đầu mảng A

# t1 = số phần tử cần in (10)

beq t1, zero, print\_done # Nếu đã in hết các phần tử

lw t2, 0(t0)

mv a0, t2

li a7, 1

ecall

# Tải phần tử hiện tại vào t2

# In số nguyên (ECALL với a7 = 1)

la a0, space # In khoảng trắng giữa các phần tử

li a7, 4

ecall

addi t0, t0, 4

addi t1, t1, -1

j print\_loop

print\_done:

# Chuyển đến phần tử kế tiếp

la a0, newline # Xuống dòng sau khi in xong

li a7, 4

ecall

jr ra Kết quả chương trình:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màn hình

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác. Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, số, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

**ASSIGNMENT 5:**

**Code:**

.data

msg\_input: .asciz "Enter a string: "

buffer: .space 21

.text

\_start:

# In thông báo yêu cầu nhập chuỗi

la a0, msg\_input

li a7, 4

ecall

# Khởi tạo các thanh ghi

li t0, 0 # t0 = 0 (biến đếm số kí tự nhập vào)

la t1, buffer # t1 = Địa chỉ bắt đầu của buffer

input\_loop:

# Đọc một kí tự từ bàn phím

li a7, 12 # Syscall number for read\_char

ecall

mv t2, a0 # Lưu kí tự vào t2

# Kiểm tra xem kí tự nhập vào có phải là Enter (mã ASCII 10) hay không

li t3, 10 # ASCII code của kí tự Enter

beq t2, t3, end\_input # Nếu kí tự là Enter thì kết thúc nhập

# Kiểm tra độ dài chuỗi có vượt quá 20 kí tự không

li t3, 20

beq t0, t3, end\_input # Nếu độ dài chuỗi đạt 20 thì kết thúc nhập

# Lưu kí tự vào bộ nhớ và tăng biến đếm

sb t2, 0(t1) # Lưu kí tự vào buffer tại vị trí t1

addi t1, t1, 1 # Di chuyển con trỏ t1 đến vị trí tiếp theo trong buffer

addi t0, t0, 1 # Tăng biến đếm t0

j input\_loop # Quay lại vòng lặp nhập tiếp kí tự

end\_input:

# In chuỗi ngược lại

addi t1, t1, -1 # Quay con trỏ t1 về vị trí cuối của chuỗi đã nhập

reverse\_print\_loop:

lb a0, 0(t1) # Đọc kí tự từ vị trí hiện tại của buffer

beq a0, zero, done # Nếu gặp kí tự null thì kết thúc

# In kí tự

li a7, 11

ecall

addi t1, t1, -1 # Di chuyển con trỏ ngược lại (lùi 1 byte)

j reverse\_print\_loop # Tiếp tục in kí tự ngược lại

done:

# Dừng chương trình

li a7, 10 # exit

ecall

Kết quả chương trình:

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

…

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, Phông chữ, số, hàng

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biên lai

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, Phông chữ, số, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.



