**Bài 6. Mảng và con trỏ**

**Họ và tên:** Nguyễn Thành Duy

**MSSV:** 20235696

Assignment 1

Code:

|  |
| --- |
| .data  A: .word -1, 6, 1, -2, 0  .text  main:  la a0, A  li a1, 5  j mspfx  continue:  exit:  li a7, 10  ecall  end\_of\_main:  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: |

Sau lần lặp thứ nhất: Ứng với độ dài 1 thì s1 = -1



Sau lần lặp thứ hai: Ứng với độ dài 2 thì s1 = -1 + 6 = 5



Cứ lặp tiếp thì kết thúc chương trình: Thanh ghi s1 = 6 (tổng max của các phần tử liên tiếp)



Assignment 2

Code:

|  |
| --- |
| .data  A: .word 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5  Aend: .word  line: .asciz "\n"  space: .ascii " "  .text  la a3, A  la a1, Aend  addi a1, a1, -4  li t2, 13 # length of A  j sort  done:  li a7, 10  ecall  sort:  beq a3, a1, done  j max  after\_max:  lw t0, 0(a1) # load last element into t0  sw s1, 0(a1) # copy max value to last element  sw t0, 0(s0) # copy last element to max location  addi a1, a1, -4  print:  la s2, A  li t0, 0  # print array A after\_max  print\_loop:  add t3, t0, t0  add t3, t3, t3  add s3, t3, s2  li a7, 1  lw a0, 0(s3)  ecall  addi t0, t0 , 1  bge t0, t2, end\_print\_loop  # print space  li a7, 4  la a0, space  ecall  j print\_loop  end\_print\_loop:  # print new line  li a7, 4  la a0, line  ecall  j sort  max:  addi s0, a3 , 0  lw s1, 0(s0)  addi t0, a3, 0  loop:  beq t0, a1, ret  addi t0, t0, 4  lw t1, 0(t0)  blt t1, s1, loop  addi s0, t0, 0  addi s1, t1, 0  j loop  ret:  j after\_max |

Giải thích: a0, a1 là con trỏ, trỏ đến đầu và cuối mảng A. Ở mỗi lần lặp lần lượt tìm phần tử lớn nhất trong mảng chèn vào cuối rồi dịch chuyển con trỏ a1 về trước cứ như vậy tới khi nào a0 = a1 thì chương trình kết thúc.

Kết quả sau khi chạy:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Mảng ban đầu: 7, -2, 5, 1, 5, 6, 7, 3, 6, 8, 8, 59, 5

Mảng sau mỗi lượt sắp xếp: -2 1 3 5 5 5 6 6 7 7 8 8 59

Assignment 3

Code:

|  |
| --- |
| .data  A: .word 5, 1, 4, -2, 7, 5, 6, 9, 6  Aend: .word  newline: .asciz "\n"  .text  main:  la t6, A # a0 = địa chỉ của A[0]  la a1, Aend  addi a1, a1, -4 # a1 = địa chỉ của A[n-1]  j bubble\_sort # gọi thủ tục sắp xếp  after\_sort:  li a7, 10  ecall  end\_main:  # --------------------------------------------------------------  # Thủ tục Bubble Sort (Sắp xếp nổi bọt)  # Cách sử dụng thanh ghi:  # t6 - con trỏ đến mảng A  # a1 - con trỏ đến phần tử cuối cùng  # t1 - con trỏ đến phần tử đang xét  # t2 - giá trị A[i]  # t3 - giá trị A[i+1]  # --------------------------------------------------------------  bubble\_sort:  outer\_loop:  beq t6, a1, done # nếu chỉ còn 1 phần tử thì done  la t1, A  inner\_loop:  addi t2, t1, 4 # t2 chứa địa chỉ của A[i+1]  bge t1, a1, next\_outer # nếu đến cuối của mảng thì kết thúc inner\_loop  lw t3, 0(t1) # A[i]  lw t4, 0(t2) # A[i+1]  ble t3, t4, no\_swap # kiểm tra xem cần đổi chỗ không  sw t3, 0(t2)  sw t4, 0(t1) # thực hiện đổi chỗ  no\_swap:  addi t1, t1, 4 # trỏ đến phần tử tiếp theo  j inner\_loop # tiếp tục vòng lặp trong  next\_outer:  addi a1, a1, -4 # cố định phần tử cuối dãy hiện tại  print:  la a5, A # lấy lại địa chỉ đầu mảng A  la a6, Aend  addi a6, a6, -4 # lấy địa chỉ cuối mảng A  j print\_array  continue:  j outer\_loop # bắt đầu sắp xếp như cũ nhưng với mảng nhỏ hơn 1 phần tử  done:  j after\_sort  # chương trình in mảng  print\_array:  bgt a5, a6, print\_end # nếu con trỏ đến phần tử cuối cùng dừng việc in  li a7, 1  lw a0, 0(a5) # lấy giá trị A[i] hiện tại  ecall # in ra  li a7, 11  li a0, 32 # Mã ASCII của dấu cách  ecall # in khoảng trắng giữa 2 số liên tiếp  addi a5, a5, 4  j print\_array  print\_end:  li a7, 4  la a0, newline  ecall # in dấu xuống dòng  j continue |

Kết quả:

Mảng ban đầu: 5, 1, 4, -2, 7, 5, 6, 9, 6

Mảng sau khi sắp xếp: -2, 1, 4, 5, 5, 6, 6, 7, 9

A screenshot of a computer

AI-generated content may be incorrect.

Giải thích: Thuật toán lặp lại quá trình so sánh a[i] và a[i+1] nếu như a[i] > a[i+1] thì swap còn không dịch chuyển con trỏ để so sánh 2 phần tử tiếp theo. Kết thúc 1 vòng lặp thì ta sẽ chuyển được giá trị lớn nhất xuống cuối mảng và cứ tiếp tục như vậy khi con trỏ đầu = con trỏ cuối.

Assignment 4

Code:

|  |
| --- |
| .data    A: .word   5, 1, 4, -2, 7, 5, 6, 9, 6    Aend: .word     dau\_cach: .asciz " "    newline:  .asciz "\n" .text main:    la    t6, A       # t6 = địa chỉ đầu của A    la    a1, Aend     addi  a1, a1, -4  # a1 = địa chỉ của A[n-1]    j     insertion\_sort   # Gọi thủ tục sắp xếp after\_sort:    li    a7, 10     ecall  end\_main: # -------------------------------------------------------------- # Thủ tục Insertion Sort (Sắp xếp chèn) # -------------------------------------------------------------- insertion\_sort:    la    t0, A    addi  t0, t0, 4   # lấy địa chỉ phần tử thứ 2 của mảng outer\_loop:                # Duyệt từ A[1] đến A[n-1] mỗi lần chọn A[i] làm key    bgt   t0, a1, done      # nếu t1 vượt quá địa chỉ cuối mảng thì sắp xếp xong     lw    t1, 0(t0)         # A[i]    addi  t2, t0, -4        # lấy địa chỉ phần tử trước đó t2 = j = i-1  inner\_loop:                # tìm vị trí thích hợp để chèn key    blt   t2, t6, insert\_key    # nếu j < 0, thì chèn key     lw    t3, 0(t2)         # Lấy A[i-1]    bgt   t3, t1, shift\_right   # nếu A[j] > key (A[i-1] > A[i]) thì dịch A[j] sang phải     j     insert\_key        # nếu không thì chèn key shift\_right:    sw    t3, 4(t2)         # A[j+1] = A[j], dịch A[j] sang phải    addi  t2, t2, -4        # j = j - 1, dịch sang trái để lấy phần tử tiếp theo cần so sánh    j     inner\_loop        # Tiếp tục vòng lặp insert\_key:    sw    t1, 4(t2)           # A[i] = A[j+1] = key (chèn key vào vị trí đúng)    addi  t0, t0, 4           # Lấy địa chỉ phần tử tiếp theo của mảng print:    la    a5, A             # lấy lại địa chỉ đầu mảng A    la    a6, Aend    addi  a6, a6, -4        # lấy địa chỉ cuối mảng A    j     print\_array       # in mảng sau mỗi vòng lặp continue:    j     outer\_loop          # sau khi in thì xét vòng lặp mới done:    j     after\_sort # chương trình in mảng print\_array:    bgt   a5, a6, print\_end    # nếu con trỏ đến phần tử cuối cùng dừng việc in     li    a7, 1    lw    a0, 0(a5)            # lấy giá trị A[i] hiện tại    ecall                      # in ra     li    a7, 11    li    a0, 32               # Mã ASCII của dấu cách    ecall                      # in khoảng trắng giữa 2 số liên tiếp     addi  a5, a5, 4    j     print\_array print\_end:    li    a7, 4    la    a0, newline    ecall                      # in dấu xuống dòng    j     continue |

Kết quả:

Mảng ban đầu: 5, 1, 4, -2, 7, 5, 6, 9, 6

Mảng sau khi sắp xếp: -2, 1, 4, 5, 5, 6, 6, 7, 9

A screenshot of a computer

AI-generated content may be incorrect.

Giải thích:

Vòng lặp 1: Phần tử thứ 2 là 1 được lấy ra, so sánh với 5 và được xếp vào bên trái 5. (Các phần tử khác không được lấy ra sắp xếp nên vị trí không đổi)

Vòng lặp 2: Phần tử thứ 3 là 4 được lấy ra, so sánh 1 < 4 < 5 nên được xếp vào giữa 1 và 5.

Tương tự như vậy đối với các phần tử khác của mảng.