**BÁO CÁO THỨC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 6**

**Họ và tên: Vũ Đức Hoàng Anh**

**MSSV: 20235658**

1. **Assignment 1**

* **Nhập chương trình:**

|  |
| --- |
| .data  A: .word -2 1 5 2 -5  .text  main:  la a0, A # Lấy địa chỉ của mảng  li a1, 20 # Số byte của mảng  j mspfx # Nhảy đến chương trình tính tổng tiền tố  continue:  exit:  li a7, 10 # Lời gọi hệ thống để kết thúc chương trình  ecall  end\_of\_main:  mspfx:  li s0, 0 # Khởi tạo tổng tiền tố  li s1, 0x80000000 # Khởi tạo s1 là số nhỏ nhất để lưu giá trị max  li t0, 0 # Khởi tạo biến đếm t0 = 0  li t1, 0 # Khởi tạo giá trị ban đầu cho tổng t1 = 0  loop:  add t3, t0, a0 # Lấy địa chỉ của phần tử tiếp theo  lw t4, 0(t3) # Lấy giá trị của A[i] tại địa chỉ t3  add t1, t1, t4 # Tính tổng chuỗi tiền tố  blt s1, t1, mdfy # Kiểm tra điều kiển của tổng  # Nếu tổng lớn hơn giá trị max thì cập nhật lại  j next  mdfy:  addi s0, t0, 4 # Độ dài tổng tiền tố  addi s1, t1, 0 # Giá trị max mới  next:  addi t0, t0, 4 # Tăng giá trị biến đếm  blt t0, a1, loop # Kiểm tra điền kiện vòng lặp  done:  j continue  mspfx\_end: |

* **Các bước thực hiện của chương trình:**

+ Cụm .data để khởi tạo giá trị cho mảng

+ Cụm câu lệnh đặt trong thẻ main để lấy giá trị của mảng và khởi tạo số byte của mảng, sau đó gọi đến hàm tính tổng tiền tố.

+ Cụm exit sau khi đã thực hiện xong hàm tính tổng tiền tố chương trình quay lại đây và kết thúc chương trình

+ Cụm câu lệnh li để khởi tạo các giá trị sắp được sử dụng

+ Cụm câu lệnh trong thẻ loop để tính tổng và cập nhật giá trị của tổng max khi lớn hơn giá trị của max. Sau đó quay lại thẻ countinue và kết thúc chương trình

* **Kết quả thực hiện chương trình:**

**+** Với giá trị của mảng được khởi tạo là -2 1 5 2 -5 thì kết quả là:

|  |
| --- |
|  |

+ Giá trị được lưu ở của sổ data segment:

|  |
| --- |
|  |

1. **Assignment 2**

* **Nhập chương trình:**

|  |
| --- |
| .data  A: .word 10, 9, 8, 7, 6, 5, 4, 3, 2, 1  Aend: .word  space: .asciz " "  newline: .asciz "\n" # Xuống dòng sau mỗi lần in mảng  .text  main:  la a0, A # Lưu đỉa chỉ của mảng A  la a1, Aend  addi a1, a1, -4 # Địa chỉ của phần tử cuối cùng trong mảng  li s2 40 # Khởi tạo số byte của mảng    j sort # sort  after\_sort:  li a7, 10  ecall  end\_main:  sort:  beq a0, a1, done # Kiểm tra nếu mảng có 1 kí tự thì kết thúc  j max # Gọi đến hàm tìm max    after\_max:  lw t0, 0(a1) # Lấy giá trị của phần tử cuối cùng  sw t0, 0(s0) # Thay giá trị của phần tử cuối cùng bằng phần tử max  sw s1, 0(a1) # Đổi giá trị của phần tử lớn nhất là phần tử cuối vừa trỏ  addi a1, a1, -4 # Lùi còn trỏ đến phần tử trước đó  addi s5, a0, 0  li s3 0 # Khởi tạo biến đếm  arrLoop:  add s4, s3, s5  lw a0, 0(s4)  li a7, 1  ecall    # In khoảng trắng  la a0, space  li a7, 4  ecall    addi s3, s3, 4 # Tăng biến đếm  bge s3, s2, endArrLoop # Kiểm tra điều kiện dừng  j arrLoop # Tiếp tục vòng lặp  endArrLoop:  la a0, newline  li a7, 4  ecall    addi a0, s5, 0  j sort # Gọi đến hàm sort và tiếp tục thực hiện  done:  j after\_sort  max:  addi s0, a0, 0 # Lấy địa chỉ của phần tử đầu tiên trong mảng  lw s1, 0(s0) # Lấy giá trị của phần tử có vị trí s0  addi t0, a0, 0 # Con trỏ trỏ đến phần tử đầu tiên  loop:  beq t0, a1, ret # kiểm tra để kết thúc chương trình  addi t0, t0, 4 # Tăng biến đếm  lw t1, 0(t0) # load next element into t1  blt t1, s1, loop# kiểm tra để tiếp tục vòng lặp  addi s0, t0, 0 # Địa chỉ mới của max  addi s1, t1, 0 # Giá trị mới của max  j loop  # Sau khi đổi tiếp tục thực hiện chương trình  ret:  j after\_max |

* **Các bước thực hiện của chương trình:**

+ Cụm câu lệnh .data để khai báo giá trị của mảng và các chuỗi kí tự

+ Cụm main để khởi tạo các giá trị của chương trình và gọi đến hàm sort

+ Cụm lệnh sau thẻ after\_sort để gọi đến hệ thống kết thúc chương trình

+ Cụm lệnh sau thẻ after\_max là vòng lặp ngoài để thay đổi vị trí con trỏ, trỏ đến vị trí tiếp theo được chọn, rồi gọi đến j max

+ Cụm lệnh trong thẻ arrLoop để in giá trị của các phần tử trong mảng sau mỗi vòng lặp, sau để thực hiện xuống dòng và tiếp tục nhảy đến thẻ after\_sort

+ Cụm câu lệnh trong thẻ max để lưu vị trí con trỏ trỏ đến phần tử đầu tiên trong mảng

+ Câu lệnh trong cụm loop để tìm xem giá trị lớn nhất trong phần còn lại của mảng đang xét, sau khi duyệt hết thì quay lại after\_max và tiếp tục thực hiện chương trình.

* **Kết quả thực hiện của chương trình:**

+ Với giá trị của mảng A được khởi tạo là 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

|  |
| --- |
| Kết quả ở cửa sổ Run I/O là: |
|  |

|  |
| --- |
| Giá trị của các thanh ghi ở cửa sổ Registers là: |
|  |

|  |
| --- |
| Các dữ liệu được lưu ở cửa sổ data segment là: |
|  |

1. **Assignment 3**

* **Nhập chương trình: (sắp xếp tăng dần)**

|  |
| --- |
| .data  A: .word 10, 9, 8, 7, 6, 5, 4, 3, 2, 1  .text  main:  la a0, A # Lấy địa chỉ của mảng A  li s2 40 # Lấy số byte của mảng A  li s0 0 # Khởi tạo biến đếm Loop1  Loop1:  bge s0, s2, endLoop1# Kiểm tra điều kiện dừng của Loop1  addi s1, s0, 4 # Biến đếm loop 2  Loop2:  bge s1, s2, endLoop2  # Kiểm tra điều kiện dừng Loop2  add s4, a0, s0 # Lấy địa chỉ của A[i]  lw a2, 0(s4) # Lấy giá trị của A[i]  add s5, a0, s1 # Lấy địa chỉ của A[j]  lw a3, 0(s5) # Lấy gía trị của A[j]  blt a3, a2, swap # So sánh để tìm phần tử lớn  j continue  swap:  # Đổi chỗ A[i] và A[j]  sw a3, 0(s4)  sw a2, 0(s5)  continue:  addi s1, s1, 4 # Tăng biến đếm Loop2  j Loop2  endLoop2:  addi s0, s0, 4 # Tăng biến đếm Loop1  j Loop1  endLoop1: |

* **Các bước thực hiện của chương trình:**

+ Cụm câu lệnh .data để khởi tạo giá trị của các phần tử trong mảng

+ Cụm câu lệnh trong thẻ main để khởi tạo độ dài mảng A và biến đếm của Loop1

+ Cụm câu lệnh trong thẻ Loop1 để thực hiện loop2, khởi tạo biến đếm cho loop2

+ Cụm câu lệnh trong Loop2 để kiểm tra và so sánh A[i] và A[j] xem nếu A[i] > A[j] thì swap hay giá trị, nếu không thì tiếp tục lặp

+ Chương trình kết thúc khi đã thực hiện lặp đến hết các phần tử trong mảng với độ phức tạp 0(n^2)

* **Kết quả thực hiện chương trình:**

+ Khởi tạo giá trị cho mảng lần lượt là 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

+ Giá trị ban đầu của cửa sổ Data segment

|  |
| --- |
|  |

+ Giá trị của cửa sổ Data segment sau khi thực hiện chương trình là:

|  |
| --- |
|  |

* **Nhập chương trình: (sắp xếp giảm dần)**

|  |
| --- |
| .data  A: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  .text  main:  la a0, A # Lấy địa chỉ của mảng A  li s2 40 # Lấy số byte của mảng A  li s0 0 # Khởi tạo biến đếm Loop1  Loop1:  bge s0, s2, endLoop1# Kiểm tra điều kiện dừng của Loop1  addi s1, s0, 4 # Biến đếm loop 2  Loop2:  bge s1, s2, endLoop2  # Kiểm tra điều kiện dừng Loop2  add s4, a0, s0 # Lấy địa chỉ của A[i]  lw a2, 0(s4) # Lấy giá trị của A[i]  add s5, a0, s1 # Lấy địa chỉ của A[j]  lw a3, 0(s5) # Lấy gía trị của A[j]  blt a2, a3, swap # So sánh để tìm phần tử lớn  j continue  swap:  # Đổi chỗ A[i] và A[j]  sw a3, 0(s4)  sw a2, 0(s5)  continue:  addi s1, s1, 4 # Tăng biến đếm Loop2  j Loop2  endLoop2:  addi s0, s0, 4 # Tăng biến đếm Loop1  j Loop1  endLoop1:  li a7 10  ecall  endMain: |

* **Các bước thực hiện chương trình:**

+ Câu lệnh điều kiệm để swap A[i] và A[j] thay đổi so với chương trình sắp xếp tăng dần.

* **Kết quả của chương trình:**

+ Với giá trị của mảng A là 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

+ Giá trị ban đầu của cửa sổ Data segment là:

|  |
| --- |
|  |

+ Giá trị của của sổ Data segment sau khi thực hiện chương trình là:

|  |
| --- |
|  |

* Kết quả đúng với kết quả tính toán.

1. **Assignment 4**

* **Nhập chương trình: (sắp xếp tăng dần)**

|  |
| --- |
| .data  A: .word 1, 3, 5, 7, 9, 10, 2, 4, 6, 8  .text  main:  la a0, A # Lấy địa chỉ của mảng A  li s2, 40 # Lấy số byte của mảng A  li s0, 4 # Bắt đầu từ phần tử thứ hai (index 1)  Loop1:  bge s0, s2, endLoop1 # Kiểm tra điều kiện dừng của Loop1  add s1, a0, s0 # Lấy địa chỉ của A[i]  lw a1, 0(s1) # Lấy giá trị của A[i] (key)  addi s3, s0, -4 # j = i - 1  Loop2:  blt s3, zero, endLoop2 # Nếu j < 0 thì dừng  add s4, a0, s3 # Lấy địa chỉ của A[j]  lw a2, 0(s4) # Lấy giá trị của A[j]  bge a1, a2, shift\_right # Nếu A[j] > key thì dời A[j]  j endLoop2  shift\_right:  addi s5, s3, 4 # Lấy địa chỉ của A[j+1]  add s6, a0, s5 # Tính địa chỉ trong mảng  sw a2, 0(s6) # A[j+1] = A[j]  addi s3, s3, -4 # j--  j Loop2  endLoop2:  addi s5, s3, 4 # Lấy địa chỉ của A[j+1]  add s6, a0, s5 # Tính địa chỉ trong mảng  sw a1, 0(s6) # A[j+1] = key  addi s0, s0, 4 # i++  j Loop1  endLoop1:  # Kết thúc chương trình  li a7, 10 # syscall 10 (exit)  ecall |

* **Các bước thực hiện của chương trình:**

+ Cụm .data để khai báo giá trị của các phần tử trong mảng

+ Cụm main để khởi tạo các giá trị cho vòng lặp Loop1, biến đếm và số byte của mảng.

+ Sau thẻ Loop1 khởi tạo giá trị của biến đếm và bắt đầu Loop2

+ Trong thẻ Loop2 so sánh giá trị của của A[j] và key nếu A[j] > key thì dịch phải nếu không thì kết thúc vòng lặp.

+ Sau khi duyệt đến hết vòng lặp 1 kết thúc chương trình.

* **Kết quả của chương trình**

**+** Với giá trị của mảng A là : 1, 3, 5, 7, 9, 10, 2, 4, 6, 8

**+** Giá trị ban đầu của cửa sổ Data segment là:

|  |
| --- |
|  |

+ Giá trị của của sổ Data segment sau khi thực hiện chương trình là:

|  |
| --- |
|  |

* **Nhập chương trình: (Sắp xếp giảm dần)**

|  |
| --- |
| .data  A: .word 1, 3, 5, 7, 9, 10, 2, 4, 6, 8  .text  main:  la a0, A # Lấy địa chỉ của mảng A  li s2, 40 # Lấy số byte của mảng A  li s0, 4 # Bắt đầu từ phần tử thứ hai (index 1)  Loop1:  bge s0, s2, endLoop1 # Kiểm tra điều kiện dừng của Loop1  add s1, a0, s0 # Lấy địa chỉ của A[i]  lw a1, 0(s1) # Lấy giá trị của A[i] (key)  addi s3, s0, -4 # j = i - 1  Loop2:  blt s3, zero, endLoop2 # Nếu j < 0 thì dừng  add s4, a0, s3 # Lấy địa chỉ của A[j]  lw a2, 0(s4) # Lấy giá trị của A[j]  bge a1, a2, shift\_right # Nếu A[j] > key thì dời A[j]  j endLoop2  shift\_right:  addi s5, s3, 4 # Lấy địa chỉ của A[j+1]  add s6, a0, s5 # Tính địa chỉ trong mảng  sw a2, 0(s6) # A[j+1] = A[j]  addi s3, s3, -4 # j--  j Loop2  endLoop2:  addi s5, s3, 4 # Lấy địa chỉ của A[j+1]  add s6, a0, s5 # Tính địa chỉ trong mảng  sw a1, 0(s6) # A[j+1] = key  addi s0, s0, 4 # i++  j Loop1  endLoop1:  # Kết thúc chương trình  li a7, 10 # syscall 10 (exit)  ecall |

* **Các bước thực hiện của chương trình:**

+ Tương tự săp xếp tăng dần nhưng thay đổi điều kiện để dịch phải

* **Kết quả của chương trình:**

**+** Với giá trị của mảng A là : 1, 3, 5, 7, 9, 10, 2, 4, 6, 8

**+** Giá trị ban đầu của cửa sổ Data segment là:

|  |
| --- |
|  |

+ Giá trị của của sổ Data segment sau khi thực hiện chương trình là:

|  |
| --- |
| A screenshot of a computer  AI-generated content may be incorrect. |

* Chương trình hoạt động bình thường với độ phức tạp O(n^2)