# BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 3

**Họ và tên: Hoàng Văn Thắng**

**MSSV: 20235828**

## Assignment 1

Tạo project để thực hiện đoạn mã trong Home Assignment 1. Khởi tạo các biến cần thiết. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, kiểm tra sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy.

**Nhập chương trình**

**A screenshot of a computer program

AI-generated content may be incorrect.**

|  |
| --- |
| # Laboratory Exercise 3, Home Assignment 1  .data  i: .word 5 # Khai báo biến i với giá trị 5  j: .word 3 # Khai báo biến j với giá trị 3  x: .word 0 # Khai báo biến x với giá trị 0  y: .word 0 # Khai báo biến y với giá trị 0  z: .word 0 # Khai báo biến z với giá trị 0  .text  start:  # TODO:  # Khởi tạo giá trị i vào thanh ghi s1  # Khởi tạo giá trị j vào thanh ghi s2  la t0, i # Load address i in t0  lw s1, 0(t0) # load value of i in s1  lw s2, 4(t0) # load value of j in s2  # Khởi tạo x, y, z (t1, t2, t3)  li t1, 0 # x = 0  li t2, 0 # y = 0  li t3, 0 # z = 0  # Cạch 1:  # blt s2, s1, else # if j < i them jump else  # Cách 2:  slt t0, s2, s1 # set t0 = 1 if j < i else clear t0 = 0  bne t0, zero, else # t0 != 0 means t0 = 1, jump else    then:  addi t1, t1, 1 # then part: x = x + 1  addi t3, zero, 1 # z = 1  j endif # skip else part  else:  addi t2, t2, -1 # begin else part: y = y - 1  add t3, t3, t3 # z = 2 \* z  endif: |

**Nội dung các thanh ghi ở mỗi bước chạy**

**Ở trạng thái ban đầu:**

* t0: Giá trị không xác định
* s1: Giá trị không xác định
* s2: Giá trị không xác định
* t1: Giá trị không xác định
* t2: Giá trị không xác định
* t3: Giá trị không xác định
* pc: Giá trị không xác định (trỏ đến lệnh la t0, i)
* Bộ nhớ:
  + Địa chỉ i: 5
  + Địa chỉ j: 3
  + Địa chỉ x, y, z: 0

**Sau lệnh *la t0, i***

****

* t0: Chứa địa chỉ của biến i (ví dụ: 0x10010000).
* s1: Giá trị không thay đổi (vẫn không xác định).
* s2: Giá trị không thay đổi (vẫn không xác định).
* t1: Giá trị không thay đổi
* t2: Giá trị không thay đổi.
* t3: Giá trị không thay đổi.
* pc: Tăng lên (trỏ đến lệnh lw s1, 0(t0)).

**Sau lệnh *lw s1, 0(t0)***



* t0: Giá trị không thay đổi (vẫn là địa chỉ của i).
* s1: 5 (giá trị của i).
* s2: Giá trị không thay đổi (vẫn không xác định).
* t1: Giá trị không thay đổi
* t2: Giá trị không thay đổi.
* t3: Giá trị không thay đổi.
* pc: Tăng lên (trỏ đến lệnh lw s2, 4(t0)).

**Sau lệnh *lw s2, 4(t0)***

******

* t0: Giá trị không thay đổi (vẫn là địa chỉ của i).
* s1: Giá trị không thay đổi (vẫn là 5).
* s2: 3 (giá trị của j).
* t1: Giá trị không thay đổi
* t2: Giá trị không thay đổi.
* t3: Giá trị không thay đổi.
* pc: Tăng lên (trỏ đến lệnh li t1, 0).

**Sau lệnh *li t1, 0***

****

* t0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* t1: 0 (giá trị của x).
* t2: Giá trị không thay đổi.
* t3: Giá trị không thay đổi.
* pc: Tăng lên.

**Sau lệnh *li t2, 0***



* t0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: 0 (giá trị của y).
* t3: Giá trị không thay đổi.
* pc: Tăng lên.

**Sau lệnh *li t3, 0***



* t0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* t3: 0 (giá trị của z).
* pc: Tăng lên.

**Sau lệnh *slt t0, s2, s1***



* t0: 1 (vì s2 (3) < s1 (5)).
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* t3: Giá trị không thay đổi.
* pc: Tăng lên.

**Sau lệnh *bne t0, zero, else***

* t0: Giá trị không thay đổi (vẫn là 1).
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* t3: Giá trị không thay đổi.
* pc: Thay đổi, trỏ đến lệnh tại nhãn else (vì t0 khác 0).

**Sau lệnh *addi t2, t2, -1* (trong nhánh else)**

****

* t0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* t1: Giá trị không thay đổi
* t2: -1(y = y -1, y=0).
* t3: Giá trị không thay đổi.
* pc: Tăng lên.

**Sau lệnh *add t3, t3, t3* (trong nhánh else)**

****

* t0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* t3: 0 (vì 0 \* 2 = 0).
* pc: Tăng lên (trỏ đến endif:).

**Sau *endif*:**

Kết thúc chương trình

**Sự thay đổi của bộ nhớ**: Bộ nhớ không thay đổi trong quá trình chạy chương trình, do chỉ thao tác với các thanh ghi và đã nạp dữ liệu từ bộ nhớ của biến i, j vào thanh ghi s1, s2; ở cuối chương trình cũng không có các thao tác lưu giá tị biến vào bộ nhớ => bộ nhớ chỉ thay đổi ở bước khởi tạo biến, sau đó giữ nguyên đến cuối.

## Assignment 2

Tạo project để thực hiện đoạn mã trong Home Assignment 2. Khởi tạo các biến cần thiết và mảng A. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, quan sát sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy. Thay bộ giá trị khác để kiểm tra sự đúng đắn của chương trình.

**Nhập chương trình:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

|  |
| --- |
| # Laboratory 3, Home Assignment 2  .data  A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6  .text  # TODO: Khởi tạo giá trị các thanh ghi s2, s3, s4  li s1, 0 # i = 0  la s2, A # load address of array A in s2  li s3, 9 # s3 = 9 (number of element in array A)  li s4, 1 # s4 = 1 (step = 1)  li s5, 0 # sum = 0  loop:  bge s1, s3, endloop # if i >= n then end loop  add t1, s1, s1 # t1 = 2 \* s1  add t1, t1, t1 # t1 = 4 \* s1 => t1 = 4\*i  add t1, t1, s2 # t1 store the address of A[i]  lw t0, 0(t1) # load value of A[i] in t0  add s5, s5, t0 # sum = sum + A[i]  add s1, s1, s4 # i = i + step  j loop # go to loop  endloop: |

**Sự thay đổi của bộ nhớ và nội dung các thanh ghi ở mỗi bước chạy**

* **Giá trị ban đầu:**

Thanh ghi:

* s1 (i): Giá trị không xác định.
* s2 (base address of A): Giá trị không xác định.
* s3 (n): Giá trị không xác định.
* s4 (step): Giá trị không xác định.
* s5 (sum): Giá trị không xác định.
* t0: Giá trị không xác định.
* t1: Giá trị không xác định.
* pc: Giá trị không xác định (trỏ đến lệnh la s2, A).

Bộ nhớ:

* A[0]: 1
* A[1]: 3
* A[2]: 2
* A[3]: 5
* A[4]: 4
* A[5]: 7
* A[6]: 8
* A[7]: 9
* A[8]: 6
* **Sau lệnh *li s1, 0***



**Thanh ghi:**

* s1: 0.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ**: Không thay đổi.

* **Sau lệnh *la s2, A***



**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Địa chỉ của mảng A (0x10010000).
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

* **Sau lệnh *li s3, 9***



**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: 9.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

* **Sau lệnh *li s4, 1***



**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: 1.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

* **Sau lệnh *li s5, 0***



**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: 0.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* pc: Tăng lên (trỏ đến nhãn loop).

**Bộ nhớ:** Không thay đổi.

* **Bắt đầu vòng lặp loop (s1 = 0):**

**Sau lệnh *bge s1, s3, endloop* (lần đầu tiên):**

**Thanh ghi:**

* s1: Giá trị không thay đổi (vẫn là 0).
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* pc: Không nhảy (vì 0 < 9), trỏ đến lệnh add t1, s1, s1.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *add t1, s1, s1* (lần 1):**

****

**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: 0 (vì 0 \* 2 = 0).
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *add t1, t1, t1* (lần 1):**



**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: 0 (vì 0 \* 2 = 0).
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *add t1, t1, s2* (lần 1):**

****

**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Địa chỉ của A[0] (0x10010000).
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lw t0, 0(t1) (lần 1):**

****

**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: 1 (giá trị của A[0]).
* t1: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau add s5, s5, t0 (lần 1):**

****

**Thanh ghi:**

* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: 1 (vì 0 + 1 = 1).
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau add s1, s1, s4 (lần 1):**

****

**Thanh ghi:**

* s1: 1 (vì 0 + 1 = 1).
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* s4: Giá trị không thay đổi.
* s5: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* PC: Tăng lên.
* **Bộ nhớ:** Không thay đổi.

**Sau j loop (lần 1):**

**Thanh ghi:** Không có thanh ghi nào thay đổi giá trị *trực tiếp*.

**Bộ nhớ:** Không thay đổi.

* pc: Thay đổi, trỏ trở lại nhãn loop.
* Vòng lặp tiếp tục lần 2

Vòng lặp sẽ tiếp tục thực hiện, với s1 (i) tăng dần lên 1, s5 (sum) cộng dồn giá trị của các phần tử trong mảng A, và t1 lần lượt chứa địa chỉ của A[1], A[2], ... cho đến khi s1 (i) bằng s3 (n = 9).

**Vòng lặp cuối cùng (khi s1 = 8):**

**Sau add s5, s5, t0 (lần cuối, s1 = 8):**

* s5 sẽ chứa tổng của các phần tử từ A[0] đến A[7] cộng thêm A[8]
* Tổng từ A[0] đến A[7] = 30, s5 = 30 + A[8] = 30 + 6 = 36

**Sau add s1, s1, s4 (lần cuối, s1 = 8):**

* s1: 9 (vì 8+1=9).

**Sau j loop (lần cuối):**

* pc: Trỏ đến loop.

**Tại loop (lần tiếp theo, s1 = 9)**

**Sau bge s1, s3, endloop (khi s1 = 9):**

**A table with numbers and a green line

AI-generated content may be incorrect.**

**Thanh ghi:**

* s1: 9.
* s2: Địa chỉ của mảng A.
* s3: 9.
* s4: 1.
* s5: 45 (Tổng của tất cả các phần tử trong mảng A).
* t0: Giá trị của A[8] (từ vòng lặp trước).
* t1: Địa chỉ của A[8] (từ vòng lặp trước).
* pc: Nhảy đến endloop (vì 9 >= 9).

**Bộ nhớ:** Không thay đổi.

**Sau khi đến endloop:**

* Các thanh ghi, bộ nhớ giữ nguyên giá trị (Bộ nhớ giữ nguyên giá trị do chỉ thao tác với thanh ghi, không có các lệnh tác động vào bộ nhớ trực tiếp)

**Như vậy, sau khi vòng lặp kết thúc, thanh ghi s5 sẽ chứa tổng của tất cả các phần tử trong mảng A (1 + 3 + 2 + 5 + 4 + 7 + 8 + 9 + 6 = 45)**.



**Thay bộ giá trị khác để kiểm tra sự đúng đắn của chương trình**

**A screenshot of a computer

AI-generated content may be incorrect.**

Sau khi chạy chương trình ta được tổng s5 = 5c16 = 9210 (kết quả đúng)



## Assignment 3

Tạo project để thực hiện đoạn mã trong Home Assignment 3. Dịch và mô phỏng với RARS. Chạy chương trình ở chế độ từng dòng lệnh, quan sát sự thay đổi của bộ nhớ và nội dung các thanh ghi ở từng bước chạy. Thay đổi bộ giá trị và chạy lại chương trình một vài lần để kiểm tra tất cả các trường hợp

**Nhập chương trình:**

A screenshot of a computer

AI-generated content may be incorrect.

|  |
| --- |
| # Laboratory Exercise 3, Home Assignment 3  .data:  test: .word 1 # Khai báo biến test với giá trị 1  .text  la s0, test # Nạp địa chỉ biến test vào s0  lw s1, 0(s0) # Nạp giá trị của biến test vào s1  li s2, 10 # a = 10  li s3, 5 # b = 5  li t0, 0 # Nạp giá trị cần kiểm tra  li t1, 1 # Nạp giá trị cần kiểm tra  li t2, 2 # Nạp giá trị cần kiểm tra  beq s1, t0, case\_0  beq s1, t1, case\_1  beq s1, t2, case\_2  j default  case\_0:  addi s2, s2, 1 # a = a + 1  j continue  case\_1:  sub s2, s2, t1 # a = a - 1  j continue  case\_2:  add s3, s3, s3 # b = 2 \* b  j continue  default:  continue: |

**Sự thay đổi của bộ nhớ và nội dung các thanh ghi ở từng bước chạy**

**Giá trị ban đầu:**

**Thanh ghi:**

* s0: Giá trị không xác định.
* s1: Giá trị không xác định.
* s2: Giá trị không xác định.
* s3: Giá trị không xác định.
* t0: Giá trị không xác định.
* t1: Giá trị không xác định.
* t2: Giá trị không xác định.
* pc: Giá trị không xác định (trỏ đến lệnh la s0, test).

**Bộ nhớ:**

* Địa chỉ test: 1 (giá trị khởi tạo).

**Sau lệnh *la s0, test***

****

**Thanh ghi:**

* s0: Địa chỉ của biến test (0x10010000).
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh lw s1, 0(s0):**



**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: 1 (giá trị của biến test).
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *li s2, 10***



**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: 10 (giá trị khởi tạo của a).
* s3: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *li s3, 5***



**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: 5 (giá trị khởi tạo của b).
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *li t0, 0***



**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* t0: 0.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *li t1, 1***

****

**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: 1.
* t2: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau lệnh *li t2, 2***

****

**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: 2.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau beq s1, t0, case\_0:** Không thay đổi bất kì thứ gì

**Sau beq s1, t1, case\_1:**

**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: Giá trị không thay đổi.
* s3: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* pc: Nhảy đến case\_1 (bỏ qua case\_0).

**Bộ nhớ:** Không thay đổi.

**Sau sub s2, s2, t1 (trong case\_1):**

****

**Thanh ghi:**

* s0: Giá trị không thay đổi.
* s1: Giá trị không thay đổi.
* s2: 9 (vì 10 - 1 = 9).
* s3: Giá trị không thay đổi.
* t0: Giá trị không thay đổi.
* t1: Giá trị không thay đổi.
* t2: Giá trị không thay đổi.
* pc: Tăng lên.

**Bộ nhớ:** Không thay đổi.

**Sau j continue (trong case\_1):**

**Thanh ghi:** Giá trị các thanh ghi không thay đổi

**Bộ nhớ:** Không thay đổi.

* pc: Nhảy đến continue.

**Sau continue:**

**Thanh ghi**: Các giá trị giữ nguyên

**Bộ nhớ:** Không thay đổi.

* Chương trình kết thúc.

**Như vậy, sau khi thực thi, thanh ghi s2 (biến a) sẽ có giá trị 9, và các thanh ghi/biến khác giữ nguyên giá trị như đã mô tả.**

**Thay đổi bộ giá trị và chạy lại chương trình một vài lần để kiểm tra tất cả các trường hợp.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Ta thấy kết quả đều đúng trong những lần thay đổi bộ giá trị**

## Assignment 4

Lần lượt thay thế điều kiện rẽ nhánh trong Home Assignment 1 bằng các điều kiện sau đây:

1. **i < j**

**Nhập chương trình:**

**A screenshot of a white paper with text

AI-generated content may be incorrect.**

|  |
| --- |
| # Laboratory Exercise 3, Assignment 4a  .text  start:  # Khởi tạo i, j, x, y, z  li s1, 5 # i = 5  li s2, 5 # j = 5  li t1, 1 # t1 = 1  li t2, 2 # t2 = 2  li t3, 3 # t3 = 3  # Điều kiện rẽ nhánh i < j  bge s1, s2, else  then:  addi t1, t1, 1 # then part: x = x + 1  addi t3, zero, 1 # z = 1  j endif # skip else part  else:  addi t2, t2, -1 # begin else part: y = y - 1  add t3, t3, t3 # z = 2 \* z  endif: |

**Với i = j = 5**

**A table with numbers and letters

AI-generated content may be incorrect.**

Ta có kết quả sau:

y = y – 1 = 2 – 1 = 1

z = 2 \* z = 2 \* 3 = 6

**Với i = 4, j = 5 (i < j)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A table with numbers and letters

AI-generated content may be incorrect.**

Ta có kết quả sau:

x = x + 1 = 2

z = 1

**Với i > j (tương tự với khi i = j)**

1. i >= j

**Nhập chương trình:**

**A screenshot of a computer

AI-generated content may be incorrect.**

|  |
| --- |
| # Laboratory Exercise 3, Assignment 4b  .text  start:  # Khởi tạo i, j, x, y, z  li s1, 10 # i = 10  li s2, 5 # j = 5  li t1, 1 # x = 1  li t2, 2 # y = 2  li t3, 3 # z = 3  # Điều kiện rẽ nhánh i >= j  blt s1, s2, else  then:  addi t1, t1, 1 # then part: x = x + 1  addi t3, zero, 1 # z = 1  j endif # skip else part  else:  addi t2, t2, -1 # begin else part: y = y - 1  add t3, t3, t3 # z = 2 \* z  endif: |

**Với i >= j**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A table with numbers and letters

AI-generated content may be incorrect.**

Ta có kết quả sau:

x = x + 1 = 2

z = 1

**Với i < j**

**A screenshot of a white paper with numbers and symbols

AI-generated content may be incorrect.**

**A table of numbers and letters

AI-generated content may be incorrect.**

Ta có kết quả sau:

y = y – 1 = 2 – 1 = 1

z = 2 \* z = 3 \* 2 = 6

1. i + j <= 0

**Nhập chương trình:**

**A screenshot of a computer

AI-generated content may be incorrect.**

|  |
| --- |
| # Laboratory Exercise 3, Assignment 4c  .text  start:  # Khởi tạo i, j, x, y, z  li s1, 10 # i = 10  li s2, 5 # j = 5  li t1, 1 # x = 1  li t2, 2 # y = 2  li t3, 3 # z = 3  # Điều kiện rẽ nhánh i + j <= 0  add t0, s1, s2 # sum = i + j  bgtz t0, else # if i + j <= 0, nhảy đến else  then:  addi t1, t1, 1 # then part: x = x + 1  addi t3, zero, 1 # z = 1  j endif # skip else part  else:  addi t2, t2, -1 # begin else part: y = y - 1  add t3, t3, t3 # z = 2 \* z  endif: |

**Với i + j > 0**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A table with numbers and letters

AI-generated content may be incorrect.**

y = y – 1 = 1

z = 2 \* z = 6

**Với i + j <= 0**

A screenshot of a computer

AI-generated content may be incorrect.

A table with numbers and letters

AI-generated content may be incorrect.

x = x + 1 = 2

z = 1

1. **i + j > m + n**

**Nhập chương trình:**

A screenshot of a computer program

AI-generated content may be incorrect.

|  |
| --- |
| # Laboratory Exercise 3, Assignment 4d  .text  start:  # Khởi tạo i, j, x, y, z, m, n  li s1, 10 # i = 10  li s2, 5 # j = 5  li t1, 1 # x = 1  li t2, 2 # y = 2  li t3, 3 # z = 3  li s4, 7 # m = 7  li s5, 3 # n = 3  # Điều kiện rẽ nhánh i + j > m + n  add t0, s1, s2 # sum = i + j  add t4, s4, s5 # sum2 = m + n  bge t4, t0, else # if i + j <= m + n, nhảy đến else  then:  addi t1, t1, 1 # then part: x = x + 1  addi t3, zero, 1 # z = 1  j endif # skip else part  else:  addi t2, t2, -1 # begin else part: y = y - 1  add t3, t3, t3 # z = 2 \* z  endif: |

**Với i + j > m + n**

A table with numbers and symbols

AI-generated content may be incorrect.

x = x + 1 = 2

z = 1

**Với i + j <= m + n**

**A table of numbers and letters

AI-generated content may be incorrect.**

y = y – 1 = 1

z = 2 \* z = 6

## Assignment 5

Lần lượt thay thế điều kiện nhảy (thoát khỏi vòng lặp) trong Home Assignment 2 bằng các điều kiện sau đây: (Cần thiết lập giá trị các phần tử của mảng để điều kiện có thể được thỏa mãn.**)**

1. i > n

**Nhập chương trình:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

|  |
| --- |
| # Laboratory 3, Assignment 5a  .data  A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6  .text  # TODO: Khởi tạo giá trị các thanh ghi  li s1, 0 # i = 0  la s2, A # load address of array A in s2  li s3, 9 # s3 = 9 (number of element in array A)  li s4, 1 # s4 = 1 (step = 1)  li s5, 0 # sum = 0  loop:  blt s3, s1, endloop # if i > n then end loop  add t1, s1, s1 # t1 = 2 \* s1  add t1, t1, t1 # t1 = 4 \* s1 => t1 = 4\*i  add t1, t1, s2 # t1 store the address of A[i]  lw t0, 0(t1) # load value of A[i] in t0  add s5, s5, t0 # sum = sum + A[i]  add s1, s1, s4 # i = i + step  j loop # go to loop  endloop: |

**A table of numbers and symbols

AI-generated content may be incorrect.**

**Giải thích kết quả chạy:**

* Thay lệnh bge ban đầu ở bài 2 thành blt s3, s1, endloop.
* Nếu n < i, kết thúc vòng lặp; ngược lại thực hiện tiếp chương trình.

1. sum < 0

**Nhập chương trình:**

A screenshot of a computer program

AI-generated content may be incorrect.

|  |
| --- |
| # Laboratory 3, Assignment 5b  .data  A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6  .text  # TODO: Khởi tạo giá trị các thanh ghi  li s1, 0 # i = 0  la s2, A # load address of array A in s2  li s3, 9 # s3 = 9 (number of element in array A)  li s4, 1 # s4 = 1 (step = 1)  li s5, 0 # sum = 0  loop:  blt s5, zero, endloop # if sum < 0 then end loop  add t1, s1, s1 # t1 = 2 \* s1  add t1, t1, t1 # t1 = 4 \* s1 => t1 = 4\*i  add t1, t1, s2 # t1 store the address of A[i]  lw t0, 0(t1) # load value of A[i] in t0  add s5, s5, t0 # sum = sum + A[i]  add s1, s1, s4 # i = i + step  j loop # go to loop  endloop: |

A table with numbers and symbols

AI-generated content may be incorrect.

**Giải thích kết quả chạy:**

* Thay lệnh bge ban đầu ở bài 2 thành blt s5, zero, endloop.
* Nếu s5 (sum) <0, kết thúc vòng lặp; ngược lại thực hiện tiếp chương trình.

1. A[i] == 0

**Nhập chương trình:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

|  |
| --- |
| # Laboratory 3, Assignment 5c  .data  A: .word 1, 3, 2, 5, 4, 7, 8, 9, 6  .text  # TODO: Khởi tạo giá trị các thanh ghi  li s1, 0 # i = 0  la s2, A # load address of array A in s2  li s3, 9 # s3 = 9 (number of element in array A)  li s4, 1 # s4 = 1 (step = 1)  li s5, 0 # sum = 0  loop:  beq t0, zero, endloop # if A[i] == 0 then end loop  add t1, s1, s1 # t1 = 2 \* s1  add t1, t1, t1 # t1 = 4 \* s1 => t1 = 4\*i  add t1, t1, s2 # t1 store the address of A[i]  lw t0, 0(t1) # load value of A[i] in t0  add s5, s5, t0 # sum = sum + A[i]  add s1, s1, s4 # i = i + step  j loop # go to loop  endloop: |

A table of numbers and letters

AI-generated content may be incorrect.

**Giải thích kết quả chạy:**

**add t1, s1, s1 # t1 = 2 \* s1**

**add t1, t1, t1 # t1 = 4 \* s1 => t1 = 4\*i**

**add t1, t1, s2 # t1 store the address of A[i]**

**lw t0, 0(t1) # load value of A[i] in t0**

* 4 câu lệnh để load giá trị của A[i] vào t0; hai câu lệnh add ban đầu là để cập nhật địa chỉ của A[i] sau mỗi lần lặp, vì 1 word có giá trị (4 byte).
* Nếu A[i] = 0, nhảy đến nhãn endloop, ngược lại thực hiện tiếp chương trình.

## Assignment 6

Tạo project để thực hiện chương trình sau:

Tìm phần tử có giá trị tuyệt đối lớn nhất từ một danh sách các số nguyên 32-bit. Giả sử danh sách số nguyên được lưu trong một mảng biết trước số phần tử.

A screenshot of a computer program

AI-generated content may be incorrect.

|  |
| --- |
| # Laboratory 3, Assignment 6  .data  array: .word 1, -3, 2, -5, 4, 7, -8, -9, -21  .text  start:  la t0, array # t0: &A  li t1, 9 # t1: n (number element of Array)  li t2, 0 # t2: i = 0  lw t3, 0(t0) # t3: max\_abs = A[0]  li t6, -2147483648 # t6: -2^31  mv a1, t3 # a1: max\_val = A[0] (giữ nguyên dấu)  bltz t3, neg\_change # Nếu A[0] < 0, đảo dấu  j loop  neg\_change:  sub t3, zero, t3 # max\_abs = -A[0]  loop:  bge t2, t1, endloop # Thoát nếu i >= n  add t4, t2, t2  add t4, t4, t4 # t4 = i \* 4  add t4, t4, t0 # t4 = &A[i]  lw t5, 0(t4) # t5 = A[i]    beq t5, t6, set\_max # Nếu A[i] == -2^31, dừng ngay lập tức  bltz t5, neg\_val # Nếu A[i] < 0, đảo dấu  j compare  neg\_val:  sub t5, zero, t5 # t5 = 0 - t5  compare:  blt t5, t3, increment # Nếu max\_abs >= abs(A[i]), bỏ qua  addi t3, t5, 0 # max\_abs = abs(A[i])  mv a1, t5 # max\_val = A[i] (giữ nguyên dấu)  increment:  addi t2, t2, 1 # i++  j loop  set\_max:  mv a1, t6 # max\_val = -2^31  j endloop  endloop:  # a1 chứa phần tử có giá trị tuyệt đối lớn nhất, giữ nguyên dấu |

**Quan sát kết quả chạy:**

**Với trường hợp mảng số không có phần tử -231**

**A table with numbers and letters

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Với trường hợp mảng có chứa phần tử -231**

**A table of numbers and letters

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Giải thích kết quả chạy:**

* t3 chứa giá trị tuyệt đối lớn nhất ban đầu (tạm thời là A[0]).
* a1 chứa **giá trị của phần tử có giá trị tuyệt đối lớn nhất** theo yêu cầu của bài toán. (mv a1, t3# a1: max\_val = A[0] (giữ nguyên dấu))
* Nếu A[0] âm (bltz t3), thì t3 = |A[0]|.
* Loop: Duyệt mảng bằng cách tính địa chỉ A[i] và tải giá trị vào t5.
* Nếu A[i](t5) == -2^31, ta chọn luôn làm phần tử có giá trị tuyệt đối lớn nhất.
* Nếu A[i] (t5) âm, đổi dấu để lấy giá trị tuyệt đối.
* Nếu |A[i]| > max\_abs, cập nhật max\_abs (t3) và gán a1 = A[i].
* Tăng chỉ số i (i++)
* Kết thúc vòng lặp

**KẾT LUẬN:**

**Lệnh Branch (Rẽ nhánh):**

* **Chức năng:** Thay đổi luồng thực thi *có điều kiện*, dựa trên kết quả so sánh. Nếu điều kiện *không* đúng, chương trình tiếp tục lệnh kế tiếp.
* **Địa chỉ đích:** Tính *tương đối* so với PC hiện tại (dùng offset).
* **Khuôn dạng:** B-type/SB-type (sử dụng offset 12-bit).
* **Mục đích:** Dùng cho các cấu trúc điều khiển trong phạm vi gần (vòng lặp, if-else).

**Lệnh Jump (Nhảy):**

* **Chức năng:** Thay đổi luồng thực thi *vô điều kiện*. Luôn luôn nhảy đến địa chỉ mới.
* **Địa chỉ đích:** Có thể *tương đối* (offset so với PC) hoặc *tuyệt đối* (tính từ giá trị thanh ghi + offset).
* **Khuôn dạng:** J-type (jal), I-type (jalr), hoặc pseudo-instruction (j).
* **Mục đích:** Dùng cho lời gọi hàm, returns, và nhảy đến các vị trí xa.

**Tại sao khác khuôn dạng?**

* **Chức năng khác:** Branch *có điều kiện*, jump *vô điều kiện*.
* **Địa chỉ đích khác:** Branch thường dùng *tương đối*, jump có thể dùng cả *tương đối* và *tuyệt đối*.
* **Tối ưu hóa:** Khuôn dạng khác nhau giúp RISC-V:
* Dùng ít bit cho branch (offset nhỏ).
* Hỗ trợ cả địa chỉ tương đối và tuyệt đối.
* Thực hiện branch và jump hiệu quả.

**Tóm lại: Branch rẽ nhánh *có điều kiện* và thường dùng địa chỉ *tương đối*. Jump nhảy *vô điều kiện* và có thể dùng cả địa chỉ *tương đối* và *tuyệt đối*. Khuôn dạng khác nhau để tối ưu hóa cho các mục đích sử dụng khác nhau.**