**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**



**BÁO CÁO MÔN HỌC**

**THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

**CE119.21.1.MTCL**

|  |  |
| --- | --- |
| **GIẢNG VIÊN HƯỚNG DẪN:** | **TRẦN ĐẠI DƯƠNG** |
| **SINH VIÊN THỰC HIỆN:** | **NGUYỄN ANH KHOA** |
| **MSSV:** | **20520903** |

**TP. HỒ CHÍ MINH, 6/2021**

MỤC LỤC

[**DANH MỤC HÌNH ẢNH** I](#_Toc75203108)

[**DANH MỤC BẢNG** II](#_Toc75203109)

[**I.** **Thực hành buổi 1 – Lab01** 1](#_Toc75203110)

[**I.1.** **Thực hành** 1](#_Toc75203111)

[**I.1.1.** **Một số lệnh assembly MIPS cơ bản** 1](#_Toc75203112)

[**I.1.2.** **Mô phỏng các chương trình bên dưới và cho biết ý nghĩa của chương trình:** 1](#_Toc75203113)

[**I.2.** **Bài tập** 1](#_Toc75203114)

[**I.2.1.** **Bài tập 1a** 1](#_Toc75203115)

[**I.2.2.** **Bài tập 1b** 1](#_Toc75203116)

[**I.2.3.** **Bài tập 1c** 1](#_Toc75203117)

[**I.2.4.** **Bài tập 1d** 1](#_Toc75203118)

[**II.** **Thực hành buổi 2 – Lab02** 1](#_Toc75203119)

[**II.1.** **Thực hành** 1](#_Toc75203120)

[**II.2.** **Bài tập** 1](#_Toc75203121)

[**II.2.1.** **Bài tập 2a** 1](#_Toc75203122)

[**II.2.2.** **Bài tập 2b** 1](#_Toc75203123)

[**III.** **Thực hành buổi 3 – Lab03** 1](#_Toc75203124)

[**III.1.** **Thực hành con trỏ** 1](#_Toc75203125)

[**III.2.** **Bài tập** 1](#_Toc75203126)

[**III.2.1.** **Bài tập 3a** 1](#_Toc75203127)

[**III.2.2.** **Bài tập 3b** 1](#_Toc75203128)

[**IV.** **Thực hành buổi 4 – Lab04** 1](#_Toc75203129)

[**IV.1.** **Thực hành** 1](#_Toc75203130)

[**IV.1.1.** **Thực hành với thủ tục** 1](#_Toc75203131)

[**IV.1.2.** **Thực hành với đệ quy** 1](#_Toc75203132)

[**IV.2.** **Bài tập** 1](#_Toc75203133)

# **DANH MỤC HÌNH ẢNH**

[Hình 1: Lưu đồ thuật toán Lab01-1a 3](#_Toc75201887)

[Hình 2: Hình biểu diễn chuỗi dưới bộ nhớ (Lab01) 4](#_Toc75201888)

[Hình 3: Lưu đồ thuật toán Lab01-3c 5](#_Toc75201889)

[Hình 4: Lưu đồ thuật toán Lab1-3d 6](#_Toc75201890)

[Hình 5: Lưu đồ thuật toán Lab02-3a 8](#_Toc75201891)

[Hình 6: Lưu đồ thuật toán Lab02-3b 9](#_Toc75201892)

[Hình 7: Lưu đồ thuật toán Lab02-3c 10](#_Toc75201893)

[Hình 8: Lưu đồ thuật toán Lab03-th\_con trỏ 11](#_Toc75201894)

[Hình 9: Lưu đồ thuật toán Lab03-3a 12](#_Toc75201895)

[Hình 10: Lưu đồ thuật toán Lab03-3b 12](#_Toc75201896)

[Hình 11: Lưu đồ thuật toán Lab03-3c 13](#_Toc75201897)

[Hình 12: Lưu đồ thuật toán tính 10 giai thừa (Lab04) 18](#_Toc75201898)

[Hình 13: Lưu đồ thuật toán xuất chuỗi Fibonacci (Lab04) 19](#_Toc75201899)

# **DANH MỤC BẢNG**

[Bảng 1 - Bảng mô tả ý nghĩa một số lệnh assembly MIPS cơ bản (Lab01) 1](#_Toc75201900)

[Bảng 2 - Ý nghĩa của các chương trình (Lab01) 2](#_Toc75201901)

[Bảng 3 - Thực hành Lab02\_1 6](#_Toc75201902)

[Bảng 4 - Thực hành Lab02\_2 7](#_Toc75201903)

[Bảng 5 - Chương trình và kết quả Lab04-1h1 14](#_Toc75201904)

[Bảng 6 - Chương trình và kết quả Lab04 - 2h2 16](#_Toc75201905)

1. **Thực hành buổi 1 – Lab01**
   1. **Thực hành**
      1. **Một số lệnh assembly MIPS cơ bản**

Sinh viên tìm hiểu tài liệu “Một số lệnh assembly MIPS cơ bản” và mô phỏng việc thực thi các lệnh và cho biết chức năng của các lệnh cơ bản sau:

Bảng 1 - Bảng mô tả ý nghĩa một số lệnh assembly MIPS cơ bản (Lab01)

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Lệnh** | **Chức năng** | **Ví dụ** |
| 1 | add | Cộng giá trị trong 2 thanh ghi và lưu vào 1 thanh ghi | add $4, $5, $9 |
| 2 | addi | Cộng giá trị thanh với 1 hằng và lưu vào 1 thanh ghi | addi $4, $5, 10 |
| 3 | addu | Cộng giá trị trong 2 thanh ghi và lưu vào 1 thanh ghi, không báo lỗi khi tràn số | addu $4, $5, $9 |
| 4 | addiu | Cộng giá trị thanh với 1 hằng và lưu vào một thanh ghi, không báo lỗi khi tràn số | addiu $4, $5, 10 |
| 5 | sub | Thanh ghi thứ 2 trừ thanh ghi thứ 3 lưu vào thanh ghi thứ 1 | sub $4, $5, $6 |
| 6 | subu | Thanh ghi thứ 2 trừ hằng số lưu vào thanh ghi thứ 1, không báo lỗi khi tràn số | sub $4, $5, 10 |
| 7 | and | Thực hiện phép and thanh ghi thứ 2 và thanh ghi thứ 3, lưu vào thanh ghi thứ 1 | and $4, $5, $6 |
| 8 | andi | Thực hiện phép and thanh ghi thứ 2 và một hằng số, lưu vào thanh ghi thứ 1 | and $4, $5, 10 |
| 9 | or | Thực hiện phép or thanh ghi thứ hai và thanh ghi thứ 3, lưu vào thanh ghi thứ 1 | or $4, $5, $6 |
| 10 | nor | Thực hiện phép nor thanh ghi thứ 2 và thanh ghi thứ 3, lưu vào thanh ghi thứ 1 | nor $4, $5, $6 |
| 11 | lw | Tải một từ từ bộ nhớ vào thanh ghi thứ 1 | lw $4, -100($5) |
| 12 | sw | Lưu một từ từ địa chỉ vào bộ nhớ RAM | sw s$ -100($5) |
| 13 | slt | Nếu thanh ghi thứ 2 nhỏ hơn thanh ghi thứ 3, thanh ghi thứ nhất là 1, ngược lại là 0 | slt $4, $5, $6 |
| 14 | slti | Nếu thanh ghi thứ 2 nhỏ hơn hằng số, thanh ghi thứ nhất là 1, ngược lại là 0 | slti $4, $5, 10 |
| 15 | sltu | Thực hiện lệnh slt nhưng so sánh không dấu, kết quả trả về giống slt | sltu $4, $5, $6 |
| 16 | sltiu | Thực hiện lệnh slti nhưng so sánh không dấu, kết quar trả về giống slti | sltiu $4, $5, 10 |
| 17 | syscall | syscall | Gọi hệ thống thực hiện lệnh được chỉ định bởi giá trị được lưu trong thanh ghi $v0 |

* + 1. **Mô phỏng các chương trình bên dưới và cho biết ý nghĩa của chương trình:**

Bảng 2 - Ý nghĩa của các chương trình (Lab01)

|  |  |  |
| --- | --- | --- |
| **STT** | **Chương trình** | **Ý nghĩa** |
| 1 | .data  var1: .word 23  .text  \_\_start:  lw $t0, var1  li $t1, 5  sw $t1, var1 | Khai báo số nguyên var1 có giá trị là 23  Đặt thanh ghi $t0 có giá trị là var1  Khởi tạo thanh ghi $t1 có giá trị là 5  Lưu giá trị thanh ghi $t1 vào var1. |
| 2 | .data  array1: .space 12  .text  \_\_start: la $t0, array1    li $t1, 5  sw $t1, ($t0)    li $t1, 13  li $t1, -7  sw $t1, 8($t0) | Cấp 12 bytes bộ nhớ vào array1  Khởi tạo thanh ghi $t0 địa chỉ của array1 (0x100100000)  Khởi tạo giá trị 5 cho thanh ghi $t1  Lưu nội dung $t1 vào word đầu tiên(0x100100000) của $t0  Khởi tạo giá trị 13 cho thanh ghi $t1  Khởi tạo giá trị -7 cho thanh ghi $t1  Lưu nội dung thanh ghi của thanh ghi $t1 vào word thứ 3 (0x10010008) của $t0 |
| 3 | li $v0, 5  syscall | Khởi tạo giá trị 5 cho thanh ghi $v0  (trong syscall 5 có nghĩa là đọc số nguyên)  Thực hiện lệnh được ghi trong $v0 |
| 4 | .data  string1: .asciiz “Print this.\n”  .text  main:  li $v0, 4    la $a0, string1  syscall | Khởi tạo chuỗi “Print this .\n” cho string1  Khởi tạo giá trị 4 cho thanh ghi $v0  (trong syscall 4 có nghĩa là in giá trị được lưu trong $a0)  Load địa chỉ string1 vào thanh ghi $a0  Thực hiện lệnh được ghi trong $v0 |

* 1. **Bài tập**
     1. **Bài tập 1a**

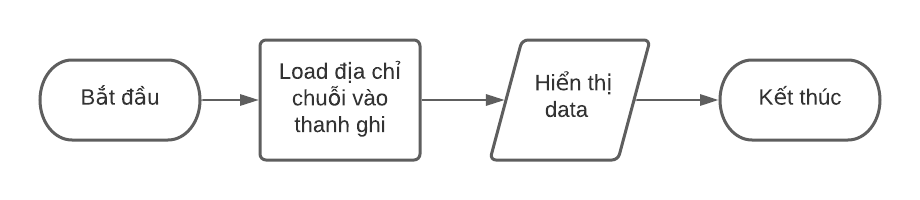
Khai báo và xuất ra cửa sổ I/O 2 chuỗi có giá trị như sau:

- Chuỗi 1: Chao ban! Ban la sinh vien nam thu may?

- Chuỗi 2: Hihi, minh la sinh vien nam thu 1 ^-^

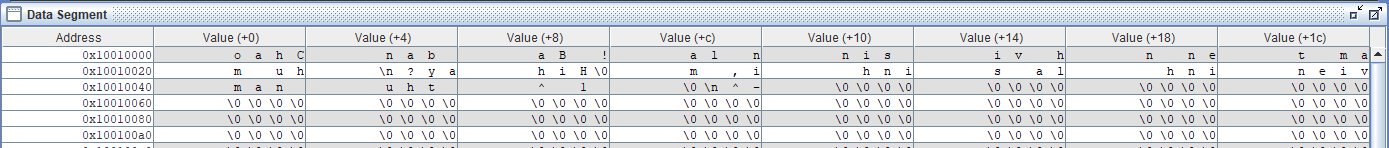
Bảng 3 – Code Bài tập 1a (Lab01)

|  |  |
| --- | --- |
| .data  string1: .asciiz "Chao ban! Ban la sinh vien nam thu may?\n"  string2: .asciiz "Hihi, minh la sinh vien nam thu 1 ^-^"  .text  main:  li $v0, 4    la $a0, string1  syscall  la $a0, string2  syscall | Khởi tạo chuỗi cho string1  Khởi tạo chuỗi cho string2  Khởi tạo giá trị 4 cho thanh ghi $v0  (trong syscall 4 có nghĩa là in giá trị được lưu trong $a0)  Load địa chỉ string1 vào thanh ghi $a0  Thực hiện lệnh được ghi trong $a0  Load địa chỉ string2 vào thanh ghi $a0  Thực hiện lệnh được ghi trong $a0 |



Hình 1: Lưu đồ thuật toán Lab01-1a

* + 1. **Bài tập 1b**



Hình 2: Hình biểu diễn chuỗi dưới bộ nhớ (Lab01)

* + 1. **Bài tập 1c**

Xuất ra lại đúng chuỗi đã nhập.

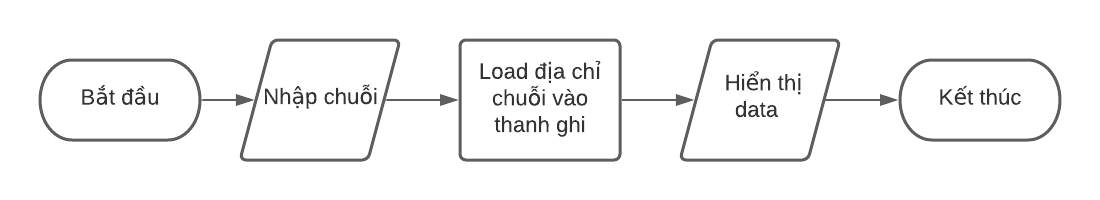
Ví dụ:

Nhap: Truong Dai hoc Cong nghe Thong tin

Xuất: Truong Dai hoc Cong nghe Thong tin

Bảng 4 – Code Bài tập 1b (Lab01)

|  |  |
| --- | --- |
| .data  str1: .space 40  strInput: .asciiz "Nhap chuoi: "  strOutput: .asciiz "Xuat chuoi: "  .text  main:  li $v0, 4    la $a0, strInput  syscall  li $v0, 8    la $a0, str1  li $a1, 40  syscall  addi $s0, $a0, 0    la $a0, strOutput    li $v0, 4  syscall    addi $a0, $s0, 0      li $v0, 4      syscall | Cấp 40 bytes bộ nhớ vào str1  Khởi tạo chuỗi cho strInput  Khởi tạo chuỗi cho strOutput  Khởi tạo giá trị 4 cho thanh ghi $v0  (trong syscall 4 có nghĩa là in giá trị được lưu trong $a0)  Load địa chỉ của strInput vào thanh ghi $a0  Thực hiện lệnh được lưu trong $a0  Khởi tạo giá trị 8 cho thanh ghi $v0  (trong syscall 8 có nghĩa là đọc giá trị được lưu trong thanh ghi $a0, thanh ghi $a1 biểu thị số lượng chữ cái tối đa để đọc)  Load địa chỉ của str1 vào thanh ghi $a0  Khởi tạo giá trị 40 cho thanh ghi $a1  Thực hiện lệnh được lưu trong $a0  Cộng 0 với thanh ghi $a0 và lưu vào thanh ghi $s0 (lưu giá trị thanh ghi $a0 vào thanh ghi $s0)  Load địa chỉ của strOutput vào thanh ghi $a0  Khởi tạo giá trị 4 vào thanh ghi $v0  (trong syscall 4 có nghĩa là in giá trị được lưu trong $a0)  Cộng 0 với thanh ghi $s0 và lưu vào thanh ghi $a0 (lưu giá trị thanh ghi $s0 vào thanh ghi $a0)  Khởi tạo giá trị 4 cho thanh ghi $v0  (trong syscall 4 có nghĩa là in giá trị được lưu trong $a0)  Thực hiện lệnh được lưu trong $v0 |



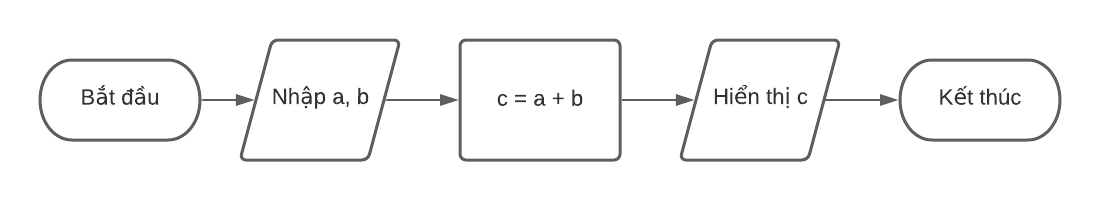
Hình 3: Lưu đồ thuật toán Lab01-3c

* + 1. **Bài tập 1d**

Nhập vào 2 số nguyên sau đó xuất tổng của 2 số nguyên này.

Bảng 5 – Code Bài tập 1d (Lab01)

|  |  |
| --- | --- |
| li $v0, 5        syscall  add $s0, $v0, $0      li $v0, 5      syscall  add $s1, $v0, $0      add $s2, $s0, $s1  li $v0, 1      add $a0, $0, $s2    syscall | Khởi tạo giá trị 5 cho than ghi $v0  (5 trong syscall có nghĩa là đọc số nguyên được nhập vào, $v0 chứa số ghi được nhập)  Thực hiện lệnh được lưu trong $v0  Cộng thanh ghi $v0 với $0 lưu vào thanh ghi $s0 (lưu giá trị thanh ghi &v0 vào thanh ghi $s0)  Khởi tạo giá trị 5 cho than ghi $v0  (5 trong syscall có nghĩa là đọc số nguyên được nhập vào, $v0 chứa số ghi được nhập)  Thực hiện lệnh được lưu trong $v0  Cộng thanh ghi $v0 với $0 lưu vào thanh ghi $s1 (lưu giá trị thanh ghi $v0 vào thanh ghi $s1)  Cộng $s0 và $s1 lưu vào thanh ghi $s2  Khởi tạo giá trị 1 cho thanh ghi $v0  (1 trong syscall có nghĩa là in số nguyên, $a0 chứa số nguyên cần in)  Cộng $s2 và $0 lưu vào thanh ghi $a0 (lưu giá trị thanh ghi $s2 lưu vào thanh ghi $a0)  Thực hiện lệnh được lưu trong $v0 |



Hình 4: Lưu đồ thuật toán Lab1-3d

1. **Thực hành buổi 2 – Lab02**
   1. **Thực hành**

Chuyển đoạn code trong bảng theo sau sang MIPS và sử dụng MARS để kiểm tra lại kết quả:

Bảng 3 - Thực hành Lab02\_1

|  |  |
| --- | --- |
| **C** | **MIPS** |
| if (i == j)  f = g + h;  else  f = g – h; | beq $s0, $s1, Else2 #kiểm tra i = j, nếu bằng thì nhảy tới Else2  j Else #lúc này i != j, nhảy tới Else  Else:  sub $s2, $t0, $t1 #f = g - h  j end  Else2:  add $s2, $t0, $t1 #f = g + h  j end  end: |

(Với giá trị của i, j, f, g, h lần lượt chứa trong các thanh ghi $s0, $s1, $s2, $t0, $t1)

Bảng 4 - Thực hành Lab02\_2

|  |  |
| --- | --- |
| **C** | **MIPS** |
| int Sum = 0;  for (int i = 1; i <=N; ++i){  Sum = Sum + i;  } | addi $s0, $0, 1 #i = 1  addi $s2, $0, 0 #Sum = 0  LOOP:  addi $t0, $s0, -1 #$t0 = i - 1  slt $t2, $t0, $s1 #so sánh $t0 và $s1(N) lưu vào $t2  beq $t2, $0, Exit #$t2 = 0, nhảy tới Exit  add $s2, $s2, $s0 #Sum = Sum + i  addi $s0, $s0, 1 #i = i + 1  j LOOP #nhảy vào LOOP  Exit: |

(Với giá trị của i, N, Sum lần lượt chứa trong các thanh ghi $s0, $s1, $s2)

* 1. **Bài tập**
     1. **Bài tập 2a**

Nhập vào một ký tự, xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau:

✓ Ký tự liền trước và liền sau của ký tự nhập vào

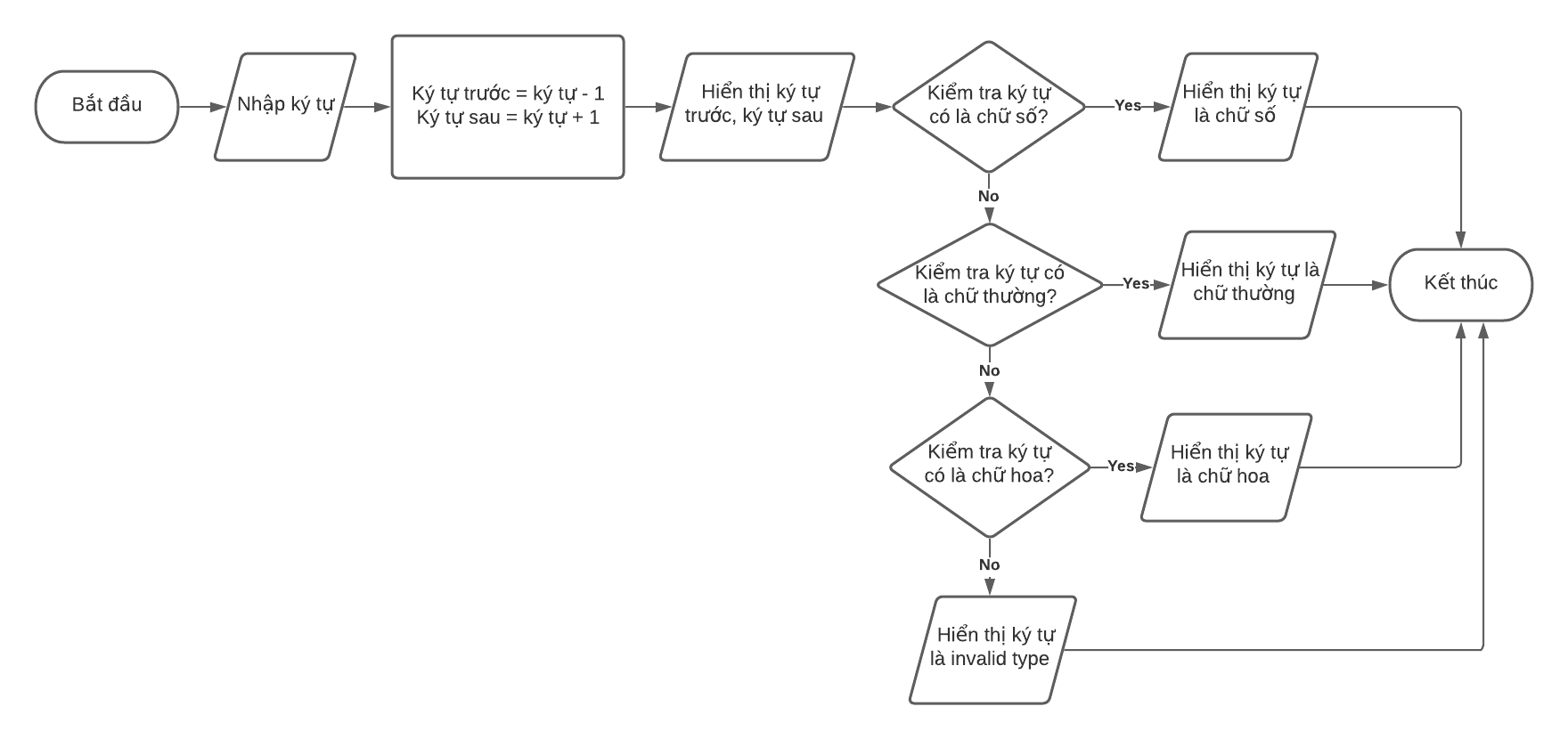
Ví dụ:

Nhap ky tu (chỉ một ký tự): b

Ky tu truoc: a

Ky tu sau: c

✓ Ký tự nhập vào chỉ được phép là ba loại: số, chữ thường và chữ hoa. Nếu ký tự nhập vào rơi vào một trong ba loại, xuất ra cửa sổ đó là loại nào; nếu ký tự nhập không rơi vào một trong ba loại trên, xuất ra thông báo “invalid type”



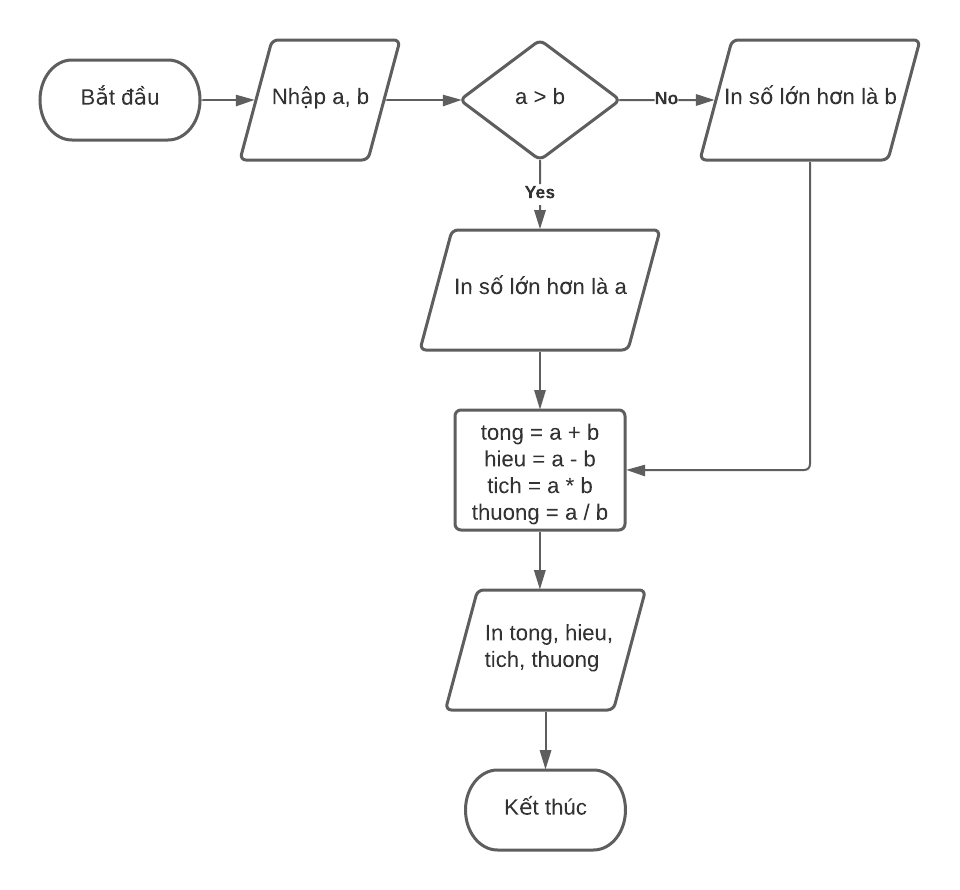
Hình 5: Lưu đồ thuật toán Lab02-3a

* + 1. **Bài tập 2b**

Nhập từ bàn phím 2 số nguyên, in ra cửa sổ I/O của MARS theo từng yêu cầu sau:

✓ Số lớn hơn

✓ Tổng, hiệu, tích và thương của hai số



Hình 7: Lưu đồ thuật toán Lab02-3c

1. **Thực hành buổi 3 – Lab03**
   1. **Thực hành con trỏ**

Cho ba mảng với cấp phát dữ liệu trong bộ nhớ như sau:

|  |
| --- |
| .data  array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4  size1: .word 10  array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16  size2: .word 16  array3: .space 8  size3: .word 8 |

Mảng array1 có 10 word, kích thước được lưu trong size1; Mảng array2 có 16 byte, kích thước được lưu trong size2; Mảng array3 có 8 byte, kích thước được lưu trong size3.

Viết code trong phần “.text” thực hiện riêng từng phần việc:

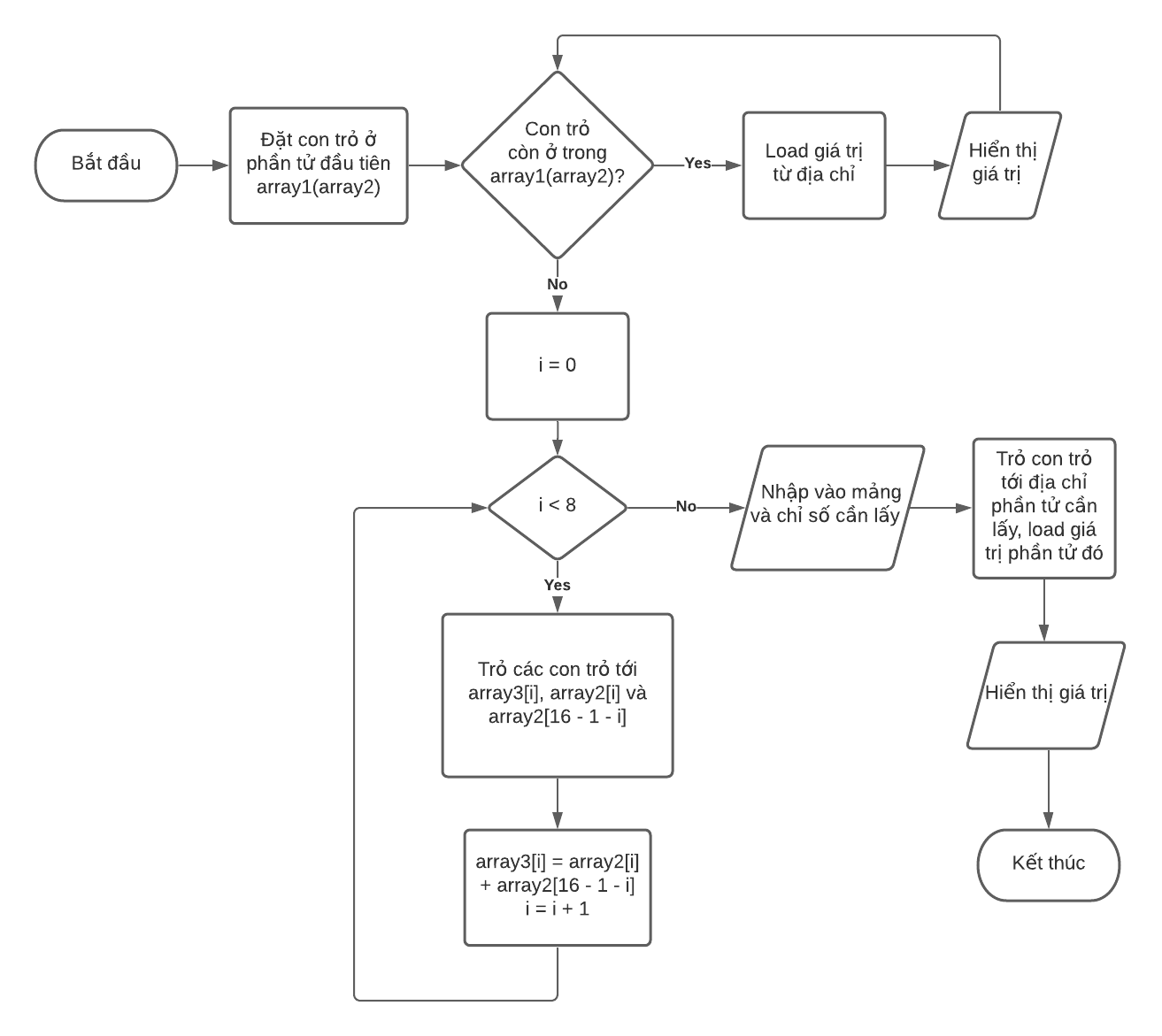
✓ In ra cửa sổ I/O của MARS tất cả các phần tử của mảng array1 và array2

✓ Gán các giá trị cho mảng array3 sao cho array3[i] = array2[i] + array2[size2 - 1 - i]

✓ Người sử dụng nhập vào mảng thứ mấy và chỉ số phần tử cần lấy trong mảng đó, chương trình xuất ra phần tử tương ứng.

Bảng 4 - Thực hành Lab02\_2

|  |
| --- |
| .data  array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4  size1: .word 10  array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16  size2: .word 16  array3: .space 8  size3: .word 8  arr1: .asciiz "Chuoi 1: "  arr2: .asciiz "Chuoi 2: "  arr3: .asciiz "Chuoi 3: "  space: .asciiz " "  enter: .asciiz "\n"  str1: .asciiz "Nhap mang: "  str2: .asciiz "Nhap chi so: "  str3: .asciiz "Phan tu co gia tri: "  .text  main:  #IN CHUOI THU 1    li $v0, 4 #in arr1  la $a0, arr1  syscall    #in chuoi    la $t0, array1 #lay dia chi array1  li $a1, 10 #$a1 = 10 (so phan tu cua array1)  sll $t1, $a1, 2 #$t1 = $a1 \* 4 = 40  add $t2, $1, $t1 #$t2 = $1 + $t1 = dia chi phan tu tiep theo cua phan tu cuoi (array1)  LOOP1:  lw $t3, 0($t0) #$t3 = array1[i]  li $v0, 1 #in array1[i]  add $a0, $0, $t3  syscall    li $v0, 4 #in space  la $a0, space  syscall    addi $t0, $t0, 4 #$t0 = $t0 + 4  slt $t4, $t0, $t2 #$t4 = ($t0 < $t2)?  bne $t4, $0, LOOP1 #$t4 != 0 -> LOOP1    li $v0, 4 #in enter  la $a0, enter  syscall    #IN CHUOI THU 2  #in arr2  li $v0, 4 #in chuoi arr2  la $a0, arr2  syscall    la $t0, array2 #lay dia chia array2  li $a1, 16 #$a1 = 16 (so phan tu cua array2)  add $t2, $t0, $a1 #$t2 = $t0 + $a1 = dia chi phan tu tiep theo cua phan tu cuoi (array2)  LOOP2:  lb $t3, 0($t0) #$t3 = array2[i]  li $v0, 1 #in array2[i]  add $a0, $0, $t3  syscall    li $v0, 4 #in space  la $a0, space  syscall    addi $t0, $t0, 1 #$t0 = $t0 + 1  slt $t4, $t0, $t2 #$t4 = ($t0 < $t2)?  bne $t4, $0, LOOP2 #$t4 != 0 -> LOOP2    li $v0, 4 #in enter  la $a0, enter  syscall  #GAN GIA TRI CHO CHUOI 3  la $t0, array2 #lay dia chi array2 luu vao $t0  li $a1, 15 #$a1 = 15  add $t1, $t0, $a1 #$t1 = $t0 + $a1 = dia chi phan tu cuoi cung array2  la $t2, array3 #lay dia chi array3 luu vao $t2  li $a1, 8 #$a1 = 8 (so phan tu cua array3)  add $t3, $t2, $a1 #$t2 = $t2 + $a1 = dia chi phan tu tiep theo cua phan tu cuoi (array3)  LOOP3.1:  lb $t4, 0($t0) #$t4 = array2[i]  lb $t5, 0($t1) #$t5 = array2[16 - 1 - i]  add $t6, $t4, $t5 #$t6 = $t5 + $t6  sb $t6, 0($t2) #array3[i] = $t6  addi $t0, $t0, 1 #$t0 = $t0 + 1  addi $t1, $t1, -1 #$t1 = $t1 - 1  addi $t2, $t2, 1 #$t2 = $t2 + 1  slt $t7, $t2, $t3 #$t7 = ($t2 < $t3)?  bne $t7, $0, LOOP3.1  #NHAP XUAT  #NHAP PHAN TU  #Nhap mang  li $v0, 4 #in str1  la $a0, str1  syscall    li $v0, 5  syscall #nhap gia tri  add $s0, $v0, $0 #lay gia tri vua nhap luu vao $s0    #Nhap chi so  li $v0, 4 #in str2  la $a0, str2  syscall    li $v0, 5  syscall #nhap gia tri  add $s1, $v0, $0 #lay gia tri vua nhap luu vao $s1    #IN PHAN TU  li $v0, 4 #in str3  la $a0, str3  syscall    #In phan tu tu array3  slti $t0, $s0, 3 #$t0 = ($s0 < 3)?  bne $t0, $0, print1 #$t0 != 0 -> print1    la $a0, array3 #lay dia chi array3  add $t2, $a0, $s1 #$t2 = dia chi array3[i]  lb $t3, 0($t2) #$t3 = array3[i]  li $v0, 1 #in array3[i]  add $a0, $0, $t3  syscall  j end  #In phan tu trong array2  print1:  slti $t0, $s0, 2 #$t0 = ($s0 < 2)?  bne $t0, $0, print2 #$t0 != 0 -> print2    la $a0, array2 #lay dia chi array2  add $t2, $a0, $s1 #$t2 = dia chi array2[i]  lb $t3, 0($t2) #$t3 = array2[i]  li $v0, 1 #in array2[i]  add $a0, $0, $t3  syscall  j end  #In phan tu trong array1  print2:  la $a0, array1 #lay dia chi array1  sll $t1, $s1, 2 #$t1 = i \* 4  add $t2, $a0, $t1 #$t2 = dia chi array1[i]  lw $t3, 0($t2) #$t3 = array1[i]  li $v0, 1 #in array1[i]  add $a0, $0, $t3  syscall  j end  end: |



Hình 8: Lưu đồ thuật toán Lab03-th\_con trỏ

* 1. **Bài tập**
     1. **Bài tập 3a**

Nhập một mảng các số nguyên n phần tử (nhập vào số phần tử và giá trị của từng phần tử), xuất ra cửa sổ I/O của MARS theo từng yêu cầu sau:

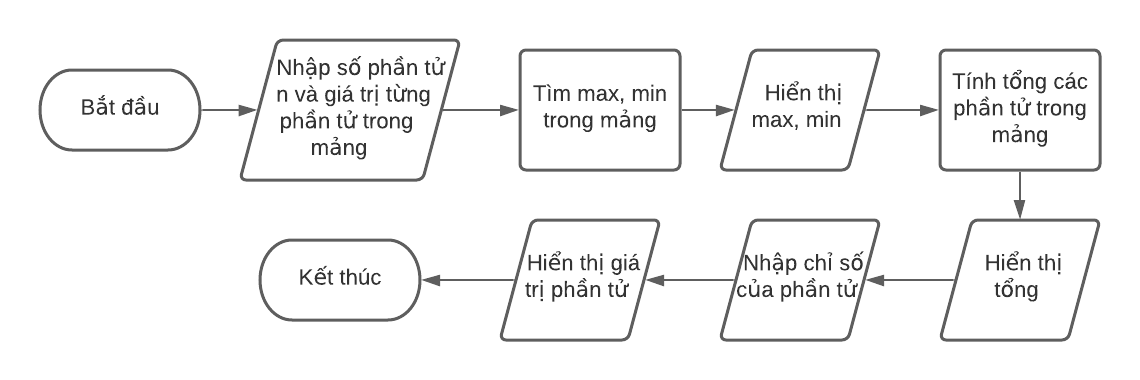
✓ Xuất ra giá trị lớn nhất và nhỏ nhất của mảng

✓ Tổng tất cả các phần tử của mảng

✓ Người sử dụng nhập vào chỉ số của một phần tử nào đó và giá trị của phần tử đó được in ra cửa sổ

Bảng 4 - Thực hành Lab02\_2

|  |
| --- |
| .data  array: .space 40  str1: .asciiz "Nhap so phan tu: "  str2: .asciiz "Nhap cac gia tri: "  str3: .asciiz "Max = "  str4: .asciiz "Min = "  str5: .asciiz "Tong = "  str6: .asciiz "Nhap vao chi so: "  str7: .asciiz "Gia tri cua phan tu: "  enter: .asciiz "\n"  .text  main:  #Nhap so phan tu cua mang  li $v0, 4  la $a0, str1 #in chuoi str1  syscall    li $v0, 5 #$v0 = 5 (read integer)  syscall  add $s0, $v0, $0 #$s0 = so phan tu cua mang (n)    #Nhap gia tri cho cac phan tu  li $v0, 4  la $a0, str2 #in chuoi str2  syscall    li $t0, 0 #$t0=0  la $t0, array #load dia chi array vao $t0  add $t1, $t0, $s0 #$t1 = $t0 + $s0 = dia chi phan tu tiep theo cua phan tu cuoi cung (array)  LOOP1:  li $v0, 5 #doc gia tri duoc nhap  syscall  sb $v0, 0($t0) #array[i] = $v0  addi $t0, $t0, 1 #$t0 = $t0 + 1 = dia chi array[i+1]  slt $t2, $t0, $t1 #$t2 = ($t0 < $t1)?  bne $t2, $0, LOOP1 #$t2 != 0 -> LOOP1    #Xuat gia tri lon nhat cua array (max)  li $v0, 4  la $a0, str3 #in chuoi ctr3  syscall    li $t7, 1 #$t7 = 1  li $t0, 0 #$t0 = i = 0  la $t0, array #load dia chi array vao $t0  lb $s1, 0($t0) #$s1 = array[0] (max tam)  add $t1, $t0, $s0 #$t1 = $t0 + $s0 = dia chi phan tu tiep theo cua phan tu cuoi cung (array)  LOOP2:  addi $t0, $t0, 1 #$t0 = $t0 + 1 = dia chi array[i+1]  slt $t4, $t0, $t1 #$t4 = ($t0 < $t1)?  beq $t4, $0, in\_max #$t4 = 0 -> continue1  lb $t3, 0($t0) #$t3 = array[i+1]  slt $t4, $s1, $t3 #t4 = ($s1 < $t3)?  beq $t4, $t7, update\_max #$t4 = 1 -> update\_max  slt $t4, $t0, $t1 #$t4 = ($t0 < $t1)?  bne $t4, $0, LOOP2  update\_max:  add $s1, $0, $t3 #$s1 = $t3  j LOOP2  in\_max:  li $v0, 1  add $a0, $0, $s1 #in $s1 (max)  syscall    li $v0, 4  la $a0, enter #in enter  syscall    #Xuat gia tri nho nhat cua array (min)  li $v0, 4  la $a0, str4 #in chuoi str4  syscall    li $t0, 0 #$t0 = i = 0  la $t0, array #load chi chi mang array  lb $s2, 0($t0) #$s2 = array[0] (min tam)  add $t1, $t0, $s0 #$t1 = $t0 + $s0 = dia chi phan tu tiep theo cua phan tu cuoi cung (array)  LOOP3:  addi $t0, $t0, 1 #$t0 = $t0 + 1 = dia chi array[i+1]  slt $t4, $t0, $t1 #$t4 = ($t0 < $t1)?  beq $t4, $0, in\_min #$t4 = 0 -> continue2  lb $t3, 0($t0) #$t3 = array[i+1]  slt $t4, $s2, $t3 #t4 = ($s1 < $t3)?  beq $t4, $0, update\_min #$t4 = 0 -> update\_min  slt $t4, $t0, $t1 #$t4 = ($t0 < $t1)?  bne $t4, $0, LOOP3 #$t4 = 0 -> LOOP3  update\_min:  add $s2, $0, $t3 #$s2 = $t3  j LOOP3  in\_min:  li $v0, 1  add $a0, $0, $s2 #in $s2 (min)  syscall    li $v0, 4  la $a0, enter #in enter  syscall    #Nhap vao chi so  li $v0, 4  la $a0, str6 #in chuoi str6  syscall  li $v0, 5 #doc gia tri duoc nhap  syscall  add $t1, $0, $v0 #$t1 = $0 + $v0    #in ra gia tri  li $v0, 4  la $a0, str7 #in chuoi str7  syscall    li $t0, 0 #$t0 = 0  la $t0, array #load dia chi mang array  add $t2, $t0, $t1 #$t2 = dia chi cua phan tu can in  LOOP4:  slt $t3, $t0, $t2 #$t3 = ($t0 < $t2)  beq $t3, $0, print #$t3 = 0 -> print  addi $t0, $t0, 1 #$t0 = $t0 + 1  j LOOP4 #jump LOOP4    print: lb $t4, 0($t0) #$t4 = array[i] can in  li $v0, 1  add $a0, $t4, $0 #in $t4  syscall  #IN SUM  li $t3, 1 #tao bien dem i  li $t5, 0 #tao bien sum  la $t0, array #load dia chi mang array  tinhsum:  slt $t4, $s0, $t3 #$t4 = ($s0 < $t3)?  beq $t4, 1, in\_sum #$t4 = 1 -> in\_sum    lb $t6,($t0) #$t6 = array[i]  add $t5, $t5, $t6 #sum = sum +$t6    addi $t0, $t0, 1 #phan tu tiep theo  addi $t3, $t3, 1 #i=i+1  j tinhsum #jump tinhsum    in\_sum:  la $a0, str5  li $v0, 4 #in str5  syscall    li $v0, 1  add $a0, $t5, $0 #in sum  syscall    li $v0, 4  la $a0, enter #in enter  syscall  #NHAP CHI SO, XUAT GIA TRI  #Nhap vao chi so  li $v0, 4  la $a0, str6 #in chuoi str6  syscall  li $v0, 5 #doc gia tri duoc nhap  syscall  add $t1, $0, $v0 #$t1 = $0 + $v0    #in ra gia tri  li $v0, 4  la $a0, str7 #in chuoi str7  syscall    li $t0, 0 #$t0 = 0  la $t0, array #load dia chi mang array  add $t2, $t0, $t1 #$t2 = dia chi cua phan tu can in  LOOP4:  slt $t3, $t0, $t2 #$t3 = ($t0 < $t2)  beq $t3, $0, print #$t3 = 0 -> print  addi $t0, $t0, 1 #$t0 = $t0 + 1  j LOOP4 #jump LOOP4    print: lb $t4, 0($t0) #$t4 = array[i] can in  li $v0, 1  add $a0, $t4, $0 #in $t4  syscall |



Hình 9: Lưu đồ thuật toán Lab03-3a

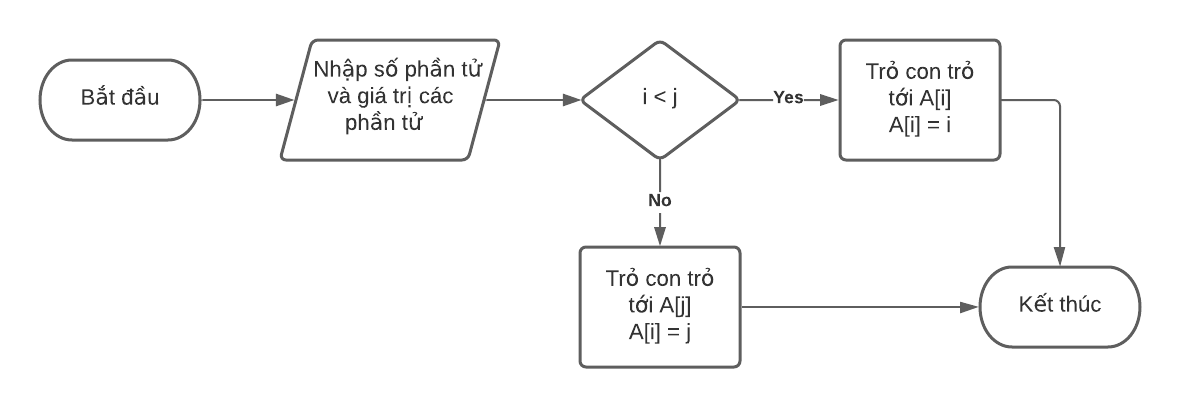
* + 1. **Bài tập 3b**

Nhập một mảng các số nguyên n phần tử (nhập vào số phần tử và giá trị của từng phần tử). Mảng này gọi là A.

Chuyển dòng lệnh C dưới đây sang mã assembly của MIPS. Với các biến nguyên i, j được gán lần lượt vào thanh ghi $s0, $s1; và địa chỉ nền của mảng số nguyên A được lưu trong thanh ghi $s3

if (i<j) A[i] = i;

else A[i] = j;



Hình 10: Lưu đồ thuật toán Lab03-3b

1. **Thực hành buổi 4 – Lab04**
   1. **Thực hành**
      1. **Thực hành với thủ tục**

Viết lại code trong Hình 1, thêm vào thủ tục tên showInt để in ra cửa sổ I/O giá trị của số int nhập vào cộng thêm 1.

Bảng 5 - Chương trình và kết quả Lab04-1h1

|  |
| --- |
| .data  prompt: .asciiz "Enter one number: "  .text  main: jal getInt  move $s0, $v0  jal showInt  j exit    getInt: li $v0, 4  la $a0, prompt  syscall    li $v0, 5  syscall  jr $ra  showInt:  addi $s1, $s0, 1 #$s1 = $s0 + 1  addi $v0, $0, 1 #$v0 = 1  add $a0, $0, $s1 #$a0 = $s1  syscall  jr $ra  exit: |
|  |

Viết lại code trong Hình 2, lúc này chương trình chính cần tính giá trị của cả hai biểu thức: (a + b) – (c + d) và (a – b) + (c – d), hàm proc\_example có hai giá trị trả về và trong thân hàm sử dụng hai biến cục bộ $s0 và $s1 ($s1 lưu kết quả của (a – b) + (c – d))

Bảng 4 - Chương trình và kết quả Lab04 -1h2

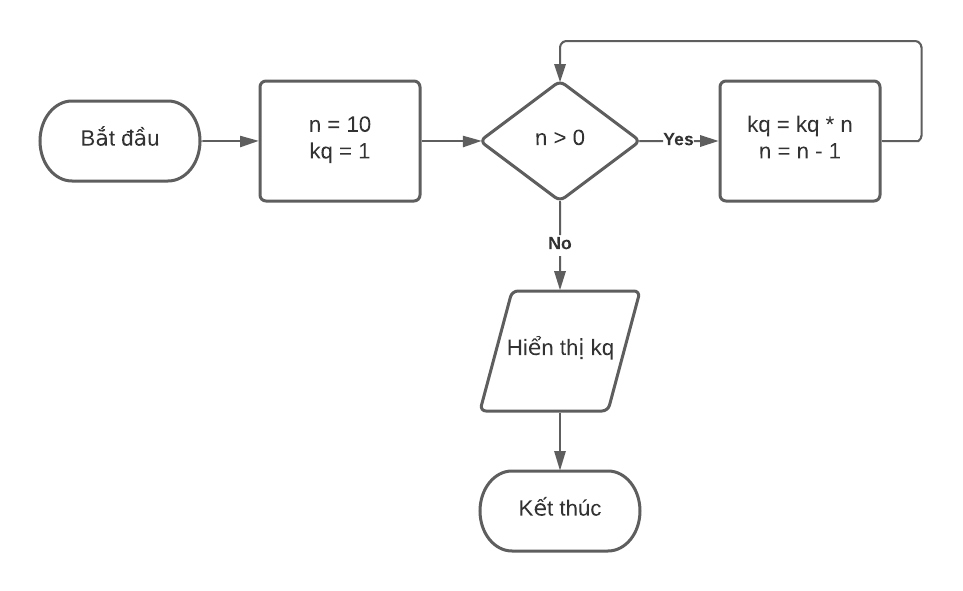
|  |
| --- |
| .data  enter: .asciiz "\n"    .text  main: move $a0, $s0 #a  move $a1, $s1 #b  move $a2, $s2 #c  move $a3, $s3 #d  jal proc\_example    #in (a + b) - (c + d)  move $a0, $v0  li $v0, 1  syscall    li $v0, 4  la $a0, enter  syscall    #in (a - b) + (c - d)  move $a0, $v1  li $v0, 1  syscall    j exit    proc\_example:  #(a + b) - (c + d) luu vao $s0  addi $sp, $sp, -4  sw $s0, 0($sp)  add $t0, $a0, $a1 #$t0 = a + b  add $t1, $a2, $a3 #$t1 = c + d  sub $s0, $t0, $t1 #$s0 = $t0 - $t1  move $v0, $s0 #$v0 = $s0  lw $s0, 0($sp)  addi $sp, $sp, 4    #(a - b) + (c - d) luu vao $s1  addi $sp, $sp, -4  sw $s1, 0($sp)  sub $t0, $a0, $a1 #$t0 = a - b  sub $t1, $a2, $a3 #$t1 = c - d  add $s1, $t0, $t1 #$s1 = $t0 + $t1  move $v1, $s1 #$v1 = $s1  lw $s1, 0($sp)  addi $sp, $sp, 4    jr $ra    exit: |
|  |

Viết lại code trong Hình 2, lúc này chương trình chính cần tính giá trị của cả hai biểu thức: (a + b) – (c + d), (e - f) hàm proc\_example có 6 input và hai giá trị trả về và trong thân hàm sử dụng hai biến cục bộ $s0 và $s1 ($s1 lưu kết quả của e - f)

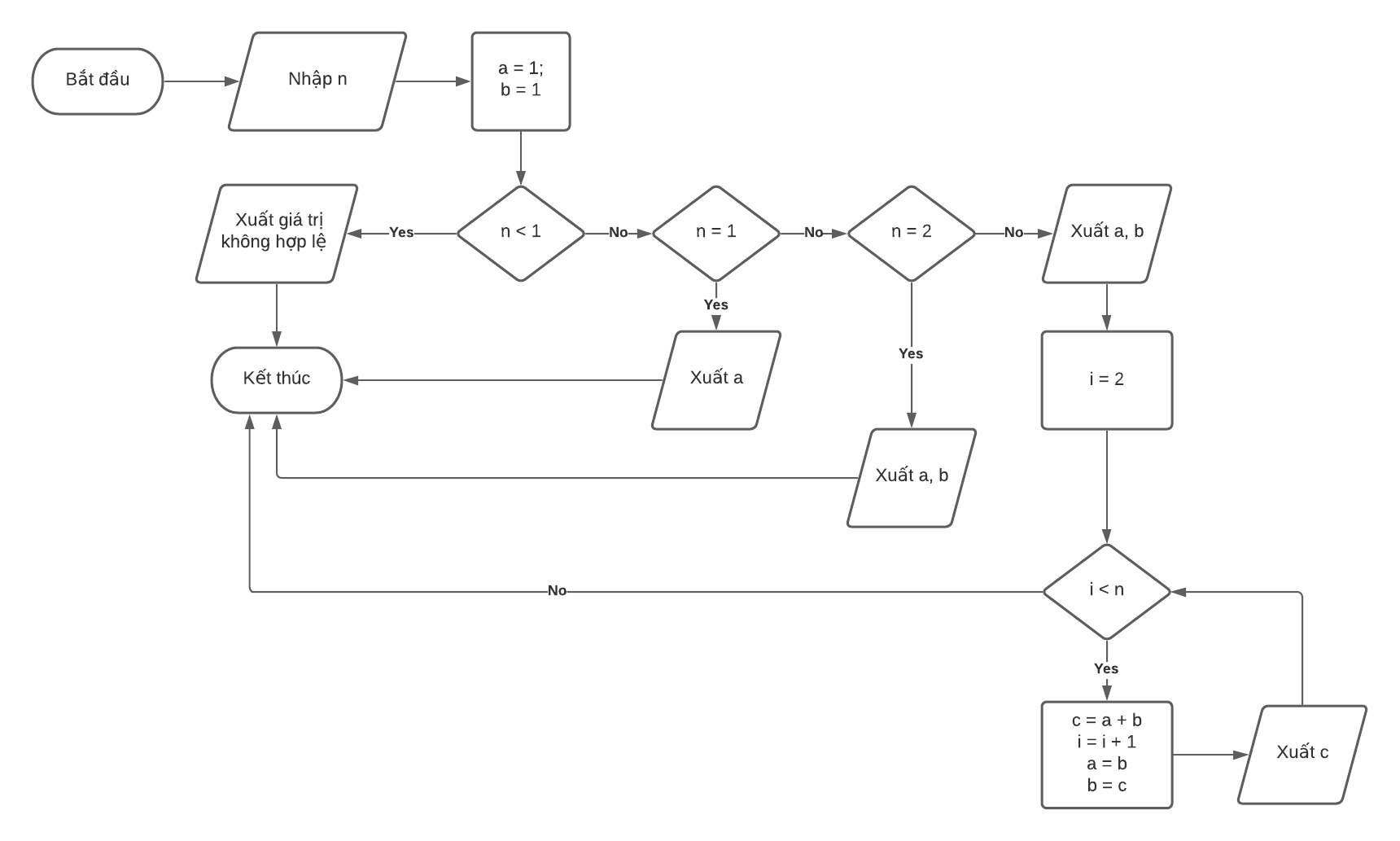
Bảng 6 - Chương trình và kết quả Lab04 - 2h2

|  |
| --- |
| .data  enter: .asciiz "\n"    .text  main: move $a0, $s0 #a  move $a1, $s1 #b  move $a2, $s2 #c  move $a3, $s3 #d  jal proc\_example    #in (a + b) - (c + d)  move $a0, $v0  li $v0, 1  syscall    li $v0, 4  la $a0, enter  syscall    #in (e - f)  move $a0, $v1  li $v0, 1  syscall    j exit    proc\_example:  #(a + b) - (c + d) luu vao $s0  addi $sp, $sp, -4  sw $s0, 0($sp)  add $t0, $a0, $a1 #$t0 = a + b  add $t1, $a2, $a3 #$t1 = c + d  sub $s0, $t0, $t1 #$s0 = $t0 - $t1  move $v0, $s0 #$v0 = $s0  lw $s0, 0($sp)  addi $sp, $sp, 4    #(e - f) luu vao $s1  add $a0, $0, $s4 #$a0 = e  add $a1, $0, $s5 #$a1 = f  addi $sp, $sp, -4  sw $s1, 0($sp)  sub $s1, $a0, $a1 #$s1 = e - f  move $v1, $s1 #$v1 = $s1  lw $s1, 0($sp)  addi $sp, $sp, 4  jr $ra  exit: |
|  |

* + 1. **Thực hành với đệ quy**



Hình 12: Lưu đồ thuật toán tính 10 giai thừa (Lab04)

* 1. **Bài tập**

Hình 13: Lưu đồ thuật toán xuất chuỗi Fibonacci (Lab04)