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

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

**🙞🕮🙜**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**MÔN HỌC: THỰC HÀNH CẤU TRÚC MÁY TÍNH**

**Giảng viên hướng dẫn: NGUYỄN NHƯ HOÀNG**

**Nhóm sinh viên thực hiện: NHÓM 9**

**Thành phố Hồ Chí Minh, ngày 15 tháng 06 năm 2023**

**MỤC LỤC**

[**DANH SÁCH THÀNH VIÊN** 4](#_Toc139183365)

[**PHÂN CÔNG CÔNG VIỆC, ĐÁNH GIÁ THÀNH VIÊN** 5](#_Toc139183366)

[**NỘI DUNG BÁO CÁO** 6](#_Toc139183367)

[**Bài tập 1:** 6](#_Toc139183368)

[1. Thuật toán Cycle Sort: 6](#_Toc139183372)

[2. Mã giả/lưu đồ thuật toán: 6](#_Toc139183373)

[3. Mô phỏng thuật toán: 9](#_Toc139183374)

[4. Mã nguồn assembly cho CPU Nios II trên hệ thống DE10-lite: 11](#_Toc139183375)

[5. Kết quả thực hiện: 14](#_Toc139183376)

[**Bài tập 2:** 15](#_Toc139183377)

[1. Vẽ lưu đồ thuật toán. 17](#_Toc139183388)

[2. Vẽ sơ đồ khối. 18](#_Toc139183389)

[3. Phân tích và giải thích nguyên lý hoạt động ở bước 2. 19](#_Toc139183390)

[4. Ưu và nhược điểm của thiết kế. 19](#_Toc139183392)

# **DANH SÁCH THÀNH VIÊN**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Ghi chú** |
| 1 | Lê Hoàng Việt Quốc | 20200323 | Nhóm trưởng |
| 2 | Đinh Văn Đạt | 20200158 |  |
| 3 | Bùi Quốc Hưng | 20200211 |  |
| 4 | Hoàng Công Nhiên | 20200297 |  |
| 5 | Phan Vũ Phong | 20200304 |  |
| 6 | Nguyễn Hồng Phương | 20200318 |  |
| 7 | Nguyễn Trung Quân | 20200319 |  |
| 8 | Bùi Minh Quang | 20200320 |  |
| 9 | Nguyễn Tiến Trung | 20200382 |  |
| 10 | Huỳnh Phước Xuyên | 20200426 |  |

# **PHÂN CÔNG CÔNG VIỆC, ĐÁNH GIÁ THÀNH VIÊN**

**Phân công nhiệm vụ các thành viên:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thành viên** | **Nhiệm vụ** |  | **Đánh giá** |
| Lê Hoàng Việt Quốc | Mã nguồn ASM, mô phỏng | Bài tập số 1 | 10/10 |
| Đinh Văn Đạt | Lưu đồ thuật toán | 10/10 |
| Bùi Quốc Hưng | Lưu đồ thuật toán | 10/10 |
| Hoàng Công Nhiên | Mã giả | 10/10 |
| Phan Vũ Phong | Mã nguồn assembly, mô phỏng | 10/10 |
| Nguyễn Hồng Phương | Lưu đồ thuật toán | Bài tập số 2 | 10/10 |
| Nguyễn Trung Quân | Lưu đồ thuật toán | 10/10 |
| Bùi Minh Quang | Sơ đồ khối | 10/10 |
| Nguyễn Tiến Trung | Sơ đồ khối | 10/10 |
| Huỳnh Phước Xuyên | Nguyên lý hoạt động, ưu điểm nhược điểm | 10/10 |

# **NỘI DUNG BÁO CÁO**

|  |
| --- |
| **Bài tập 1:** Viết chương trình assembly cho CPU NIOS II trên hệ thống DE10-lite thực hiện sắp xếp mảng bằng thuật toán Cycle Sort với các yêu cầu:Không dùng các lệnh liên quan đến phép nhân và chia.Đầu vào là N số ngẫu nhiên được lưu trước trong bộ nhớ.Đầu ra là mảng N số đã được sắp xếp được lưu trong bộ nhớ. |

### **1. Thuật toán Cycle Sort:**

**Thuật toán Cycle Sort:** Thuật toán Cycle Sort là một thuật toán sắp xếp sử dụng để sắp xếp mảng số nguyên. Nó hoạt động bằng cách tìm kiếm các chu kỳ trong mảng và di chuyển các phần tử đúng vào vị trí đúng của chúng.

Dưới đây là mô tả cụ thể của thuật toán Cycle Sort:

* **Bước 1:** Duyệt qua từng phần tử trong mảng, gọi phần tử hiện tại là “item”.
* **Bước 2:** Tìm vị trí đúng của “item” trong mảng sắp xếp bằng cách đếm số phần tử nhỏ hơn nó (từ vị trí 0 đến vị trí hiện tại) và gọi vị trí này là “pos”.
* **Bước 3:** Nếu “pos” bằng vị trí hiện tại của “item”, tiếp tục sang phần tử tiếp theo.
* **Bước 4:** Nếu “pos” khác với vị trí hiện tại của “item”, di chuyển “item” vào vị trí “pos” bằng cách hoán đổi giá trị của “item” và giá trị tại vị trí “pos”.
* **Bước 5:** Tiếp tục quá trình tìm kiếm chu kỳ và di chuyển các phần tử cho đến khi tất cả các phần tử đều đã được đặt đúng vị trí.

### **2. Mã giả/lưu đồ thuật toán:**

Theo thuật toán đã có, ta tạo được lưu đồ thuật toán như sau:

A picture containing screenshot, black and white, text, design

Description automatically generated

Từ lưu đồ thuật toán, ta có được mã giả như sau:

CycleSort(arr[], n)

for cycle\_start = 0 to n - 2

item = arr[cycle\_start]

pos = cycle\_start

for i = cycle\_start + 1 to n - 1

if arr[i] < item

pos = pos + 1

if pos = cycle\_start

continue

while item = arr[pos]

pos = pos + 1

if pos ≠ cycle\_start

swap item and arr[pos]

while pos ≠ cycle\_start

pos = cycle\_start

for i = cycle\_start + 1 to n - 1

if arr[i] < item

pos = pos + 1

while item = arr[pos]

pos = pos + 1

if item ≠ arr[pos]

swap item and arr[pos]

End for

**Ví dụ:** Cho mảng cần sắp xếp gồm 6 phần tử lần lượt theo thứ tự là: 5, 3, 2, 1, 4.

### **3. Mô phỏng thuật toán:**

Mô phỏng thuật toán Cycle Sort với mảng đã cho:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 3 | 2 | 1 | 4 |

Xét mảng từ vị trí **1** đến vị trí **n - 1**.

Bắt đầu với **i = 1**, **item = arr[i] = 5**, **pos = i = 1.**

Xét mảng từ vị trí **i + 1** đến **n – 1**, có 4 phần tử nhỏ hơn **5** → **pos** **= 1 + 4 = 5.**

**pos ≠ i** → hoán đổi phần tử ở vị trí **i = 1** và **pos = 5** → **item = arr[i] = 4.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 3 | 2 | 1 | 4 |

Ta được mảng:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 1 | 5 |

Vì **pos ≠ i** → **pos = i = 1**, xét mảng từ vị trí **i + 1** đến **n – 1**, có 3 phần tử nhỏ hơn **4** → **pos = 1 + 3 = 4.**

Vì **item ≠ arr[pos] = 4** → hoán đổi phần tử ở vị trí **pos** và phần tử **item.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 1 | 5 |

Ta được mảng:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 2 | 4 | 5 |

Vì **pos ≠ i** → xét mảng từ vị trí **i + 1** đến **n – 1**, có 0 phần tử nhỏ hơn **1** → **pos = 1 + 0 = 1**, lúc này **pos = i** → tiến đến phần tử **i** tiếp theo.

Với **i = 2**, **item = arr[i] = 3**, **pos = i = 2.**

Xét mảng từ vị trí **i + 1** đến **n – 1**, có 1 phần tử nhỏ hơn **3** → **pos = 2 + 1 = 3.**

**pos ≠ i** → hoán đổi phần tử ở vị trí **i = 2** và **pos = 3** → **item = arr[i] = 2.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 2 | 4 | 5 |

Ta được mảng:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

Vì **pos ≠ i** → **pos = i = 2**, xét mảng từ vị trí **i + 1** đến **n – 1**, có 0 phần tử nhỏ hơn **2** → **pos = 1 + 0 = 2**, lúc này **pos = i** → tiến đến phần tử **i** tiếp theo.

Với **i = 3**, **item = arr[i] = 3**, **pos = i = 3.**

Xét mảng từ vị trí **i + 1** đến **n – 1**, có 0 phần tử nhỏ hơn **3** → **pos = 3 + 1 = 3.**

Vì **pos = i** → tiến đến phần tử **i** tiếp theo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

Với **i = 4**, **item = arr[i] = 4**, **pos = i = 4.**

Xét mảng từ vị trí **i + 1** đến **n – 1**, có 0 phần tử nhỏ hơn **4** → **pos = 4 + 0 = 4.**

Vì **pos = i** → tiến đến phần tử **i** tiếp theo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

Lúc này **i = n – 1**, kết thúc thuật toán và ta có được mảng đã được sắp xếp như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

### **4. Mã nguồn assembly cho CPU Nios II trên hệ thống DE10-lite:**

**.text**

**.global** \_start

**\_start:**

movia r4, ARRAY

movia r5, SIZE

subi r5, r5, 5

mov r6, r4 #cycle

**outer\_loop:**

mov r7, r4 #pos

mov r12, r7 #i

ldw r8, 0(r6)

**inner\_loop:**

bge r7, r5, check

addi r7, r7, 4

ldw r10, 0(r6)

ldw r11, 0(r7)

bgt r11, r10, inner\_loop

addi r12, r12, 4

ldw r9, 0(r12)

br inner\_loop

check:

beq r12, r6, out

stw r9, 0(r6)

stw r10, 0(r12)

**while:**

beq r12, r6, out

**outer\_loop1:**

mov r7, r4

mov r12, r7

ldw r8, 0(r6)

**inner\_loop\_1:**

bge r7, r5, check\_1

addi r7, r7, 4

ldw r10, 0(r6)

ldw r11, 0(r7)

bgt r11, r10, inner\_loop\_1

addi r12, r12, 4

ldw r9, 0(r12)

br inner\_loop\_1

**check\_1:**

stw r9, 0(r6)

stw r10, 0(r12)

br while

**out:**

blt r5, r6, out2

**increment:**

addi r6, r6, 4

br outer\_loop

**out2:**

br out2

**.data**

**ARRAY:**

**.word** 5, 3, 2, 1, 4

**SIZE:**

**.word** 5

**.end**

### **5. Kết quả thực hiện:**

Chạy thử mã nguồn với đầu vào là mảng gồm **5** phần tử: **5, 3, 2, 1, 4.**

A screenshot of a computer

Description automatically generated

**Hình 1:** Đầu vào là N số ngẫu nhiên được lưu trước trong bộ nhớ.

Sau khi chạy chương trình ta thu được đầu ra là mảng gồm **5** phần tử với thứ tự đã được sắp xếp:

A screenshot of a computer

Description automatically generated

**Hình 2:** Đầu ra là mảng N số đã được sắp xếp được lưu trong bộ nhớ.

|  |
| --- |
| **Bài tập 2:** Thiết kế bộ cộng trừ dấu chấm cộng IEEE 754 32-bit:Vẽ lưu đồ thuật toán.Vẽ sơ đồ khối (vẽ chi tiết).Phân tích và giải thích nguyên lý hoạt động ở bước 2.Ưu và nhược điểm của thiết kế. |

### Tiêu chuẩn IEEE 754 32-bit là một tiêu chuẩn quy định định dạng và các quy định định dạng và các quy tắc cho biểu diễn số dấu chấm động trong máy tính. Đây là tiêu chuẩn phổ biến được sử dụng rộng rãi trong các hệ thống máy tính hiện đại.

Trong tiêu chuẩn IEEE 754 32-bit, một số dấu chấm động được biểu diễn bằng 32 bit và được chia thành các phần như sau:

1. Dấu (sign): Sử dụng một bit để biểu diễn dấu của số. Bit này có giá trị 0 nếu số là dương và 1 nếu số là âm.
2. Mũ (exponent): Sử dụng tám bit để biểu diễn mũ của số. Mũ này biểu thị phần mũ của số dấu phẩy động và được biểu diễn dưới dạng dạng bù hai (two's complement). Với 8 bit, phạm vi biểu diễn của mũ là từ -127 đến 128. Tuy nhiên, một số giá trị đặc biệt được dùng để biểu diễn các trạng thái đặc biệt như vô cùng (infinity) và không-a-số (NaN).
3. Phần định trị (fraction): Sử dụng 23 bit để biểu diễn phần định trị của số. Phần định trị này biểu diễn các chữ số nhị phân sau dấu phẩy của số dấu phẩy động và có giá trị từ 0 đến (1 - 2-23).

Dưới đây là ví dụ biểu diễn số 0.15625 dưới dạng dấu chấm động:

A picture containing screenshot, text, line

Description automatically generated

Tổng quát, số dấu phẩy động có dạng:

### **Ví dụ:** Thực hiện cộng hai số và sử dụng thuật toán trên:

Ta có dạng thập phân của 2 số với 4 chữ số phần định rõ là:

### 

### 

Theo thuật toán:

### Bước 1: Phần định rõ của số có phần mũ nhỏ hơn () được dịch phải cho đến khi phần mũ của nó bằng với số lớn hơn:

Bước 2: Cộng phần định rõ:

### Bước 3: Chuẩn hoá tổng, kiểm tra dữ liệu:

Vì , nằm trong phạm vi dữ liệu. Số mũ biased là nằm giữa và

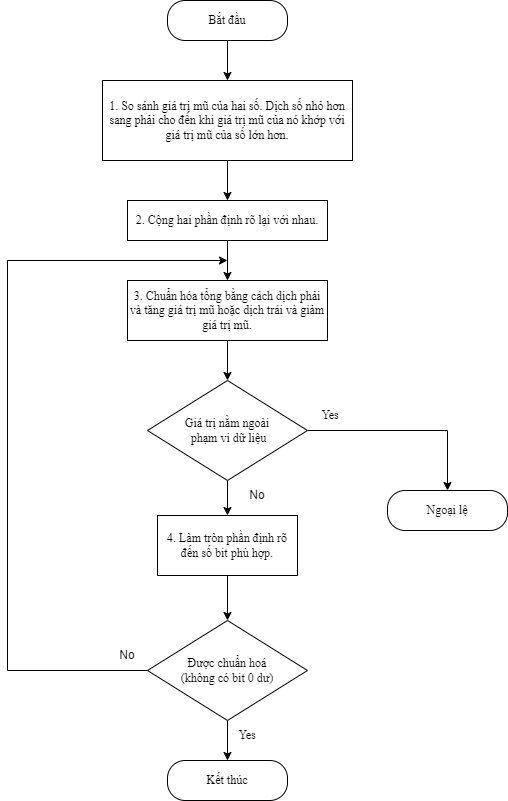
Bước 4: Làm tròn tổng:

Tổng đã chính xác 4 bits, do đó không thay đổi với các bits do làm tròn.

Đây là tổng cần tìm.

### **1. Vẽ lưu đồ thuật toán.**

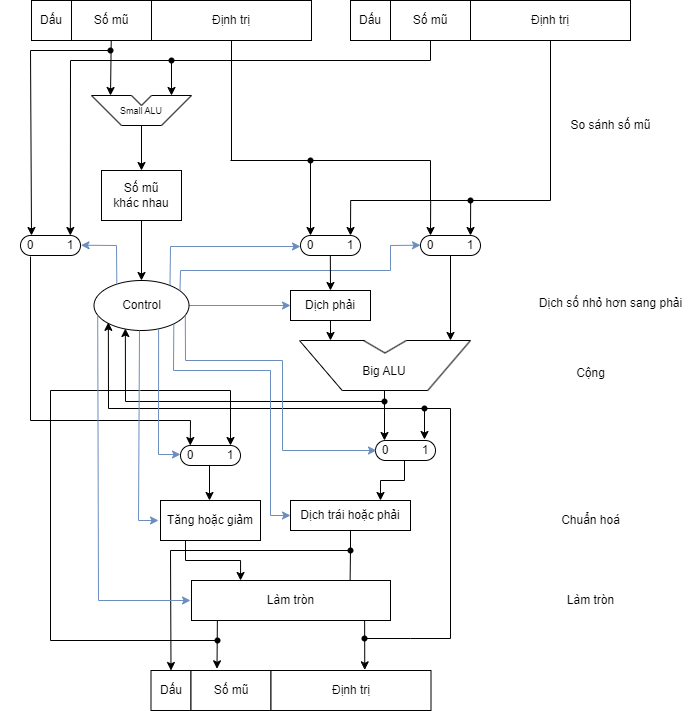
Từ ví dụ, ta có được lưu đồ thuật toán như sau:



**Hình 3:** Lưu đồ thuật toán bộ cộng trừ dấu chấm động IEEE 754 32-bit.

### **2. Vẽ sơ đồ khối.**

Với từng bước của lưu đồ thuật toán, ta có sơ đồ khối với từng khối tương ứng từ trên xuống dưới như sau:



**Hình 4:** Sơ đồ khối chi tiết bộ cộng trừ dấu chấm động IEEE 754 32-bit.

### **3. Phân tích và giải thích nguyên lý hoạt động ở bước 2.**

### Trước tiên, sử dụng một bộ trừ nhỏ ALU, giá trị mũ của một toán hạng được trừ đi từ toán hạng còn lại để xác định toán hạng nào lớn hơn và chênh lệch là bao nhiêu. Sự khác biệt này điều khiển ba bộ chọn đa phương tiện; từ trái qua phải, chúng chọn giá trị mũ lớn hơn, phần định rõ của số nhỏ hơn và phần định rõ của số lớn hơn. Phần định rõ nhỏ hơn được dịch sang phải và các phần định rõ được cộng lại với nhau bằng cách sử dụng bộ tính toán ALU lớn. Sau đó, bước chuẩn hóa sẽ dịch tổng sang trái hoặc sang phải, tăng hoặc giảm giá trị mũ. Bước làm tròn sau đó tạo ra kết quả cuối cùng, có thể yêu cầu chuẩn hóa lại để tạo ra kết quả cuối cùng.

### **4. Ưu và nhược điểm của thiết kế.**

**Ưu điểm:**

* Thiết kế sử dụng chuẩn IEEE 754 32-bit, nên nó tuân thủ các quy tắc và tiêu chuẩn đã được chấp nhận rộng rãi trong tính toán số dấu chấm động.
* Các bước được thực hiện theo trình tự logic, đảm bảo tính chính xác và đúng đắn trong việc thực hiện các phép toán số dấu chấm động.
* Sơ đồ khối chi tiết cung cấp một cái nhìn tổng quan về quá trình thực hiện các phép toán, giúp dễ dàng hiểu và sửa lỗi trong trường hợp cần thiết.

**Nhược điểm:**

* Thiết kế này chỉ áp dụng cho phép cộng và trừ số dấu chấm động. Các phép toán khác như nhân, chia, căn bậc hai, v.v. không được bao gồm.
* Mặc dù sơ đồ khối chi tiết cung cấp một cái nhìn tổng quan, nó không cung cấp chi tiết về triển khai các khối trong thiết kế, điều này có thể làm cho việc hiểu và triển khai thực tế phức tạp hơn.

-----🙞🕮🙞-----