VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Kiến Trúc Máy Tính (CO2008)

BÀI TẬP LỚN

Advisor: Nguyễn Xuân Minh
Student: Hồ Thành Nhân
Student ID: 2212352
Student: Trần Phước Nhật
Student ID: 2212412

HO CHI MINH CITY, DECEMBER 2023

MỤC LỤC

Kiến Trúc Máy Tính (CO2008) **1**
Đề 3. Merge sort số thực chính xác đơn. **3**
1. Giải Thuật Merge Sort **3**
2. Giải pháp hiện thực: **4**
3. Thống kê số lệnh, loại lệnh (instruction type) sử dụng trong chương trình: **4**
4. Thời gian chạy của chương trình (Execution time): **4**
5. Kết quả kiểm thử **5**
6. Phụ lục **5**

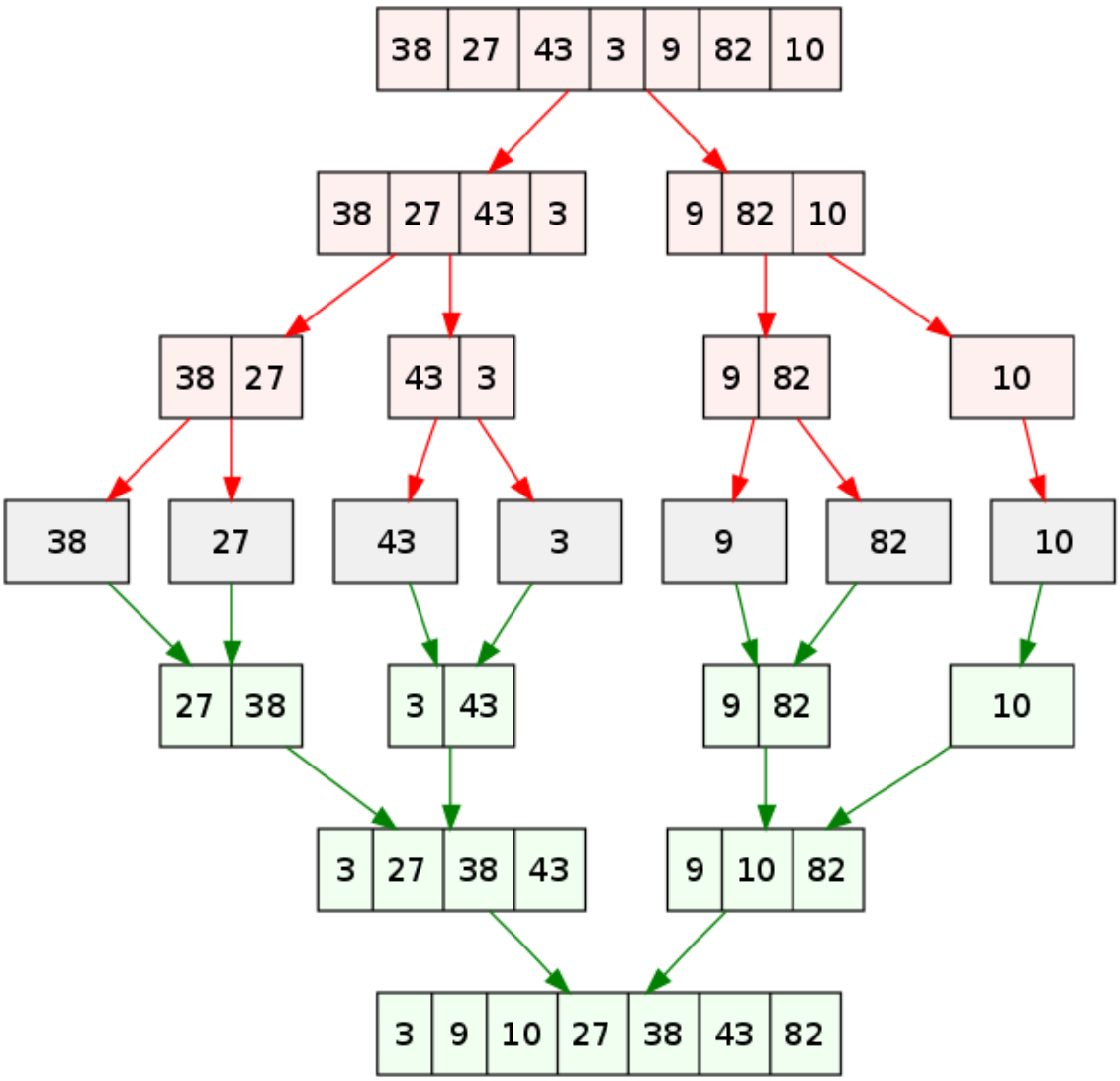
Đề 3. Merge sort số thực chính xác đơn.

Viết chương trình sắp xếp dãy số thực chính xác đơn có 15 phần tử dùng giải thuật Merge sort.

Yêu cầu xuất dãy ra màn hình mỗi bước. Dữ liệu đầu vào đọc từ file lưu trữ dạng nhị phân trên đĩa FLOAT 15.BIN (15 phần tử x 4 bytes = 60 bytes).

1. Giải Thuật Merge Sort

Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.



Minh Họa giải thuật Merge Sort

```
mergeSort(arr[], l,  r)

If r > l
   1. Tìm chỉ số nằm giữa mảng để chia mảng thành 2 nửa:
        middle m = (l+r)/2
   2. Gọi đệ quy hàm merge Sort cho nửa đầu tiên:
        mergeSort(arr, l, m)
   3. Gọi đệ quy hàm mergeSort cho nửa thứ hai:
        mergeSort(arr, m+1, r)
   4. Gộp 2 nửa mảng đã sắp xếp ở (2) và (3):
        merge(arr, l, m, r)
```

Pseudocode cho Merge sort

2. Giải pháp hiện thực:

Bước 1: Hiện thực giải thuật Merge sort trong code C++

Bước 2: Hiện thực quá trình đọc dữ liệu từ đầu vào là đĩa binary sang mảng số thực trong MIPS.

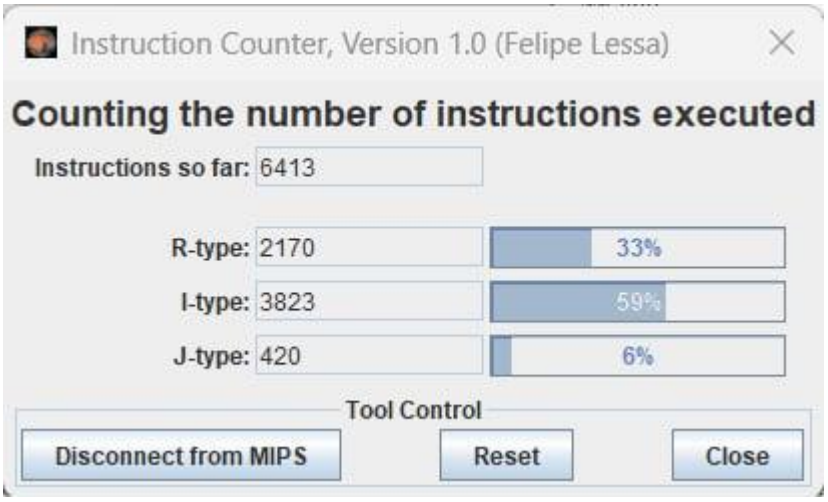
Bước 3: Từng bước chuyển giải thuật Merge Sort từ C++ sang MIPS.

3. Thống kê số lệnh, loại lệnh (instruction type) sử dụng trong chương trình:

Tổng cộng có 6413 số lệnh được thực thi

Trong đó gồm:

+ 2170 lệnh R-Type
+ 3823 lệnh I-Type
+ 420 lệnh J-Type



4. Thời gian chạy của chương trình (Execution time):

$$\text{Execution time} = \frac{IC.CPI}{CR} = \frac{3438.1}{1GHz} = 0.000003438\ (s)$$

5. Kết quả kiểm thử

Đầu vào: 15 số thực được lưu dưới dạng nhị phân trong đĩa "FLOAT 15.BIN" như sau

```
Unsorted Array:  1.1 -2.2 -3.3 -4.4 -5.5 -6.6 -7.7 -8.8 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5
```

Đầu ra: dãy các số thực đã được sắp xếp theo giải thuật Merge Sort

```
Sorted Array:  -15.5 -12.3 -10.1 -9.9 -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 11.2 13.3 14.4
```

6. Phụ lục

Quá trình sắp xếp chia mảng và hợp nhất các mảng đã sắp xếp:

```
Left: 0
Mid: 0
Right: 1
Sort:  -2.2 1.1 -3.3 -4.4 -5.5 -6.6 -7.7 -8.8 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5
```

```
Left: 2
Mid: 2
Right: 3
Sort:  -2.2 1.1 -4.4 -3.3 -5.5 -6.6 -7.7 -8.8 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5
```

```
Left: 0
Mid: 1
Right: 3
Sort:  -4.4 -3.3 -2.2 1.1 -5.5 -6.6 -7.7 -8.8 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5
```

```
Left: 4
Mid: 4
Right: 5
Sort:  -4.4 -3.3 -2.2 1.1 -6.6 -5.5 -7.7 -8.8 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5
```

```
Left: 6
Mid: 6
Right: 7
Sort:  -4.4 -3.3 -2.2 1.1 -6.6 -5.5 -8.8 -7.7 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5
```

```
Left: 4
Mid: 5
```

Right: 7
Sort: -4.4 -3.3 -2.2 1.1 -8.8 -7.7 -6.6 -5.5 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5

Left: 0
Mid: 3
Right: 7
Sort: -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 -9.9 -10.1 11.2 -12.3 13.3 14.4 -15.5

Left: 8
Mid: 8
Right: 9
Sort: -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 -10.1 -9.9 11.2 -12.3 13.3 14.4 -15.5

Left: 10
Mid: 10
Right: 11
Sort: -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 -10.1 -9.9 -12.3 11.2 13.3 14.4 -15.5

Left: 8
Mid: 9
Right: 11
Sort: -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 -12.3 -10.1 -9.9 11.2 13.3 14.4 -15.5

Left: 12
Mid: 12
Right: 13
Sort: -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 -12.3 -10.1 -9.9 11.2 13.3 14.4 -15.5

Left: 12
Mid: 13
Right: 14
Sort: -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 -12.3 -10.1 -9.9 11.2 -15.5 13.3 14.4

Left: 8
Mid: 11
Right: 14
Sort: -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 -15.5 -12.3 -10.1 -9.9 11.2 13.3 14.4

Left: 0
Mid: 7
Right: 14
Sorted Array: -15.5 -12.3 -10.1 -9.9 -8.8 -7.7 -6.6 -5.5 -4.4 -3.3 -2.2 1.1 11.2 13.3 14.4

Code C++ dùng để hiện thực MIPS:

```
#include <iostream>

using T = int;

void printArray(T *start, T *end) {
   for (T *ptr = start; ptr != end; ptr++) {
      std::cout << *ptr << " ";
   }
   std::cout << std::endl;
}
void merge(int *a, int low, int mid, int high) {
   int i = low;
   int k = 0;  // change here
   int j = mid + 1;
   int *c = new int[high - low + 1];

   while (i <= mid && j <= high) {
      if (a[i] < a[j]) {
         c[k] = a[i];
         k++;
         i++;
      } else {
         c[k] = a[j];
         k++;
         j++;
      }
   }

   while (i <= mid) {
      c[k] = a[i];
      k++;
      i++;
   }

   while (j <= high) {
      c[k] = a[j];
      k++;
      j++;
   }

   for (i = low, k = 0; i <= high; i++, k++) {  // change here
      a[i] = c[k];
   }
```

```
    delete[] c;
}
void mergeSort(int *a, int low, int high) {
    if (low < high) {
        int mid = low + (high - low) / 2;
        mergeSort(a, low, mid);
        mergeSort(a, mid + 1, high);
        merge(a, low, mid, high);
        printArray(a + low, a + high + 1);  // print array after merge
    }
}

int main() {
    T array[] = {44, 33, 27, 10, 35, 19, 14, 42, 31, 26};
    int size = sizeof(array) / sizeof(array[0]);

    std::cout << "Before sorting: ";
    printArray(array, array + size);

    mergeSort(array, 0, size - 1);

    std::cout << "After sorting: ";
    printArray(array, array + size);

    return 0;
}
```