計算機結構 HW2

姓名：莊育權
學號：B03901142
系級：電機三

I、作業報告

一、 Bubblesort

1. 設計架構

根據 C++的 code

Function in assembly code

```
int main() {

    int n;
    cin >> n ;
    for (int i = 0 ; i < n ; i++)
        cin >> a[i];

    bubblesort(a, 0, n);

    cout << "Sorting result" << endl;
    for (int i = 0 ; i < n ; i++)
        cout << a[i] << " ";
}

void bubblesort(int arr[], int len) {
    int i, j, temp;
    for (i = 0; i < len; i++)
        for (j = i - 1; j >= 0; j--)
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
}
```

Input

Output

for1tst

for2tst

swap

2. Function

(1) Input

根據使用者輸入的 size 創建出一個 sized array，並把使用者輸入的數字存到相對應的矩陣裡。

(2) Output

將已經排列過的數字，一個一個印出到 Console。

(3) for1tst + for2tst + swap

選取一個 array 上的位置(pivot)，將此位置的數字與上一個位置的數字做比較，若比較小就做交換，再繼續比下一組。等整組比完，之後將 pivot 往上移一個位置，繼續做相同的事情。於是 sorting 是將數字由小排到大。

二、 Quicksort

1. 設計架構

根據 C++的 code

Function in assembly code

```
int main() {
    int n;
    cin >> n ;
    for (int i = 0 ; i < n ; i++)
        cin >> a[i];

    quickSort(a, 0, n);

    cout << "Sorting result" << endl;
    for (int i = 0 ; i < n ; i++)
        cout << a[i] << " ";
}
void quickSort(int* a, int left, int right) {

    if (left < right) {
        int pivot = a[(left+right)/2];
        int i = left , j = right ;
        while (i < j)
        {
            while (a[i] < pivot); ++i;
            while (a[j] > pivot); --j;
            if (i < j) {
              if(a[i] != a[j]){
                int temp;
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
              }
              if(i<pivot) i++
              if(j>pivot) j--
            }
        }
        quickSort(a, left, i-1);
        quickSort(a, j+1, right);
    }
}
```

Input

Output

quicksort

while1

while2

while3

pre_swap

makesure

swap

quicksort_left

quicksort_right

2. Function

(1) Input

根據使用者輸入的 size 創建出一個 sized array，並把使用者輸入的數字存到相對應的矩陣裡。

(2) Output

將已經排列過的數字，一個一個印出到 Console。

(3) while1 + while2 +while3

在 pivot 左邊，找出比 pivot 還大的數字。在 pivot 右邊，找出比 pivot 還小的數字。

(4) pre_swap + makesure

pre_swap 是用來確保 i<j，避免 i 和 j 都選到 pivot，而造成不斷交換，形成無限迴圈。而 makesure 是用來確保，如果其中 i 或 j 選到 pivot，而另外一個選到跟 pivot 一樣的數字時，不要交換，直接進行下一個數字，以免造成無限迴圈。

(5) quicksort_left & quicksort_right

quicksort_left 為將分成兩邊的左邊再去做 quicksort。

quicksort_right 為將分成兩地的右邊再去做 quicksort。

三、 模擬結果

1. Bubblesort

```
How many numbers of intergers:
10
Input integers:
-1
3
-5
7
-9
2
-4
6
-8
10
Sorting Result:
-9 -8 -5 -4 -1 2 3 6 7 10
```

2. Quicksort

```
How many numbers of intergers:
10
Input integers:
-1
3
-5
7
-9
2
-4
6
-8
10
Sorting Result:
-9 -8 -5 -4 -1 2 3 6 7 10
```

四、 問題與討論

1.

一開始我先測試輸入一個數字，之後會印出一樣的數字當作練習，但我每次不管輸入甚麼數字，它都只會印出 0。最後才發現我在輸入數字的時候是使用中文輸入鍵盤，而必須使用英文輸入鍵盤才會是正確的。

2.

一開始對於矩陣的 size 要如何根據 input 的數量給定想很久。後來助教說，可以先讓使用者輸入矩陣的 size。而 syscall code = 9 就可以根據你給的 size，給出一個相對應空間的矩陣。

3.

第一次在寫 quicksort 的 recursive 時，一直再想如何讓這一層迴圈做完之後，可以準確地跳回上一層，因為 ra 又只會存一個數值，難保在做這層迴圈的時候 ra 不會被蓋掉。最後想到其實在做每一層迴圈的時候可以用 stack 把上一層的迴圈的數值都存起來，等做完這層迴圈的時候，再把 stack 裡面的數值 load 回去，這樣就可以準確地跳回上一層，而到了上一層，屬於那層的 parameter 也會跟當時是一樣的。

五、 心得

這是第一次打 assembly code，一開始光要去建立一個可以改變 size 的矩陣，然後把 input 輸入進去，就花了很多時間。Bubblesort 還算好打，只需要把 c++的 code 一行一行翻譯成 assembly code 就好。Quicksort 就很麻煩，雖然也是一行一行打，只是多了一個 recursive 的迴圈就更複雜，這裡我想了很久，要想說，要怎麼做才能使得 recursive 順利做下去，在這裡真的消耗我很大的腦量。而且我覺得 assembly 最困難的是 debug，assembly 不像是 C++可以有自己定義的 variable，assembly 都是將數值或是 address 存到 reg 裡面，這樣對於自己如何找到 variable 其實有點麻煩。而用 Qtspim 的視窗去做追蹤，看的也會眼花撩亂，最後都是用筆寫在紙上，慢慢去做 debug。

做之前，一直以為反正老師有教過，當時也有聽懂，寫起來應該不困難。等到真的寫起來，卻發覺困難重重，有很多用法都亂用。真的要等實際操作過後才能更體會和更了解裡面的運作與用法，因此也讓我更了解老師上課在什麼，跟 assembly 的用法。