COMP 273 Assignment 4

School of Computer Science
McGill University
Available On: Friday, March 22, 2019.
Due Date: Friday, April 5, 2019 11:59pm.

Submit your solution in electronic form using MyCourses
Read the submission instructions at the end of the document
(Late policy: 10 marks off per day late, up to 2 days late. 0 After that.)
Sharing code is strictly prohibited.
YOU WILL BE GRADED ON WHAT YOU SUBMIT IN YOUR ASSIGNMENT FOLDER AND IT IS YOUR RESPONSIBILITY TO CHECK THAT THE FILES YOU INTENDED TO SUBMIT ARE ACTUALLY IN THAT FOLDER!
Make sure to follow instructions carefully!

1 Word Count (40 Marks)

Using the template *wordcount.asm* as a starting point, you are to write a MIPS program that counts the number of occurrences of a word in a text block entered using the keyboard. The user enters a text block (no more than 600 characters) in the console and then asks for a word to search for. Note that a word means any continuous set of characters before the next space. The program should search the text block and display the number of occurrences of the word entered. The program should print the text and the word entered by the keyboard. The end of the text block is reached when the user presses the <Enter> key. After the program finishes it should provide an option of redoing the task or exiting the program.
You should implement the program using **memory-mapped I/O**, i.e., you are not allowed to use the syscall. A sample output would be like this:

```
Word count
Enter the text segment:
    Roses are red, the sky is blue and firetrucks are red.
Enter the search word:
    red
The word 'red' occurred 2 time(s).
press 'e' to enter another segment of text or 'q' to quit.
```

Your code should handle cases where the search word is not in the text block. For example:

```
Word count
Enter the text segment:
    I love assembly language. Language is my thing!
Enter the search word:
    computer
The word 'computer' occurred 0 time(s).
press 'e' to enter another segment of text or 'q' to quit.
```

Note that for this assignment you need to use the "Keyboard and Display MMIO Simulator" in MARS by connecting it to MIPS. This can be found under Tools->Keyboard and Display MMIO Simulator. Click Connect to MIPS. In the main MARS window, after assembling your code, select the 0xffff0000 (MMIO) portion of the data segment. This is the memory mapped IO area. Click Reset in the MMIO simulator window.

2 QuickSort using Memory-Mapped I/O (60 marks)

You are required to implement the quicksort algorithm using the MIPS assembly language by completing the *quicksort.asm* template file provided to you. Your program must use **memory-mapped I/O** for both the inputs and outputs.

1. As inputs, consider only numbers that have up to two digits.
2. The numbers will be separated by a single blank space.
3. Your program should work correctly on any valid inputs. Also, it should ignore any unknown keys (see the table below for known keys). You may also assume that the user will not enter three consecutive digits.
4. Your program must echo (print) the digits entered by the user.
5. Note that entries such as 'c', 's' and 'q' are not to be echoed on the screen.
6. You may assume that the program will not be tested on negative numbers.
7. You may also assume a fixed size array (containing at-least 10 elements).
8. In the event that the user presses **s** with less than 10 numbers having being entered, the results displayed should be only for the numbers entered.
9. If the array has not been cleared (i.e 'c' has not been pressed), the elements of the array entered previously should be combined with the new entries, before the sorting algorithm is applied.

10. You may assume that no attempt will be made to check for the overflow of the array, i.e., that the count of numbers entered will never exceed the size of the array.

key	Meaning	Echo
0-9	The digits 0 to 9	yes
SPACE	The blank space	yes
c	Clear/reinitialize the array	no
s	Display the sorted array	no
q	Quit the program	no

Guidelines

In order to give you an idea how the program should work, here is a sample output

```
Welcome to QuickSort
12 3 4 67 89 <s>
The sorted array is: 3 4 12 67 89
99 78 <c>
The array is re-initialized
20 13 56 99 <s>
The sorted array is: 13 20 56 99
3 40 99 78 0 10 <s>
The sorted array is: 0 3 10 13 20 40 56 78 99 99
<q>
<program ends>
```

The commands in brackets <s,c,q> are not echoed on-screen

1. In order to understand the working of the program, a schematic diagram is provided with this assignment. This provides one way to approach the problem. This is by NO MEANS the only way to do this assignment. Please feel free to implement your own design.

2. There are some subtle points that have to be carefully worked out. One important point is how to deal with sorting an incomplete array of elements. One way is to maintain a variable which will contain the the maximum index value of the filled array. Initially it will be set to 0, but it will be updated whenever new numbers are entered. If the user presses c, the index value will be reset to 0.

3. Take special care when dealing with a two digit number. For example if the user inputs 23, you will receive the input in the following order : 2 3. In order to generate the numerical value of 23, you will have to multiply the 2 by 10 and then add 3.

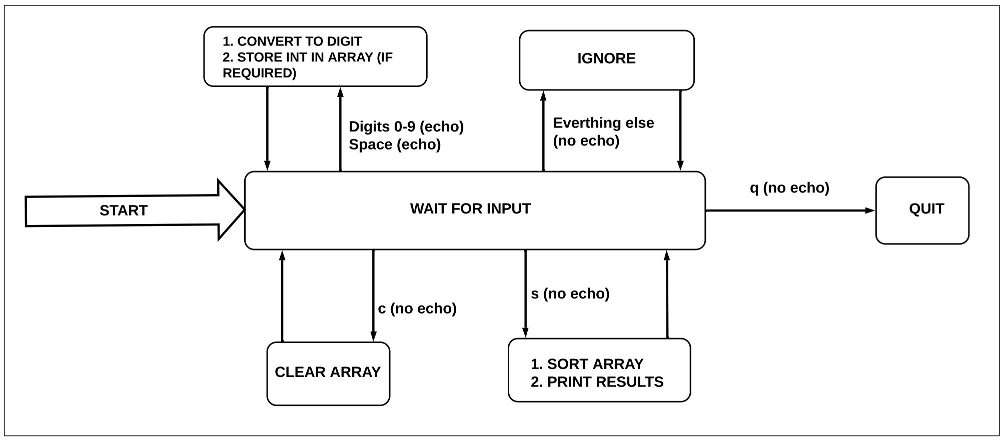

Figure 1: Flowchart for quicksort program flow.

QuickSort

QuickSort is an in-place sort algorithm that uses the divide and conquer paradigm. It has two phases: the partition phase and the sort phase. It picks an element from the array (the pivot), partitions the remaining elements into those greater than and less than this pivot, and recursively sorts the partitions. In the most general case, we don't know anything about the items to be sorted, so any choice of pivot element works. (The first element is a convenient choice.) You can choose to implement any strategy for the choice of pivot.

There are some excellent websites with java applets showing quicksort in action. One such website is:

`https://visualgo.net/bn/sorting`

The C source code for quicksort is provided below.

```
#include <stdio.h>
#define LISTSIZE 5
int list[LISTSIZE];

void quicksort(int [], int, int );
int partition(int  [], int, int );
void swap(int [],int,int );

int main(int argc, char *argv[]) {
  int val;
  int i;

  // read in the data
  i = 0;
  printf("List size = %d\n",LISTSIZE);
  while(i < LISTSIZE) {
    printf("Enter value: ");
    scanf("%d",&val);
    list[i] = val;
    i++;
  }
  // sort it
  quicksort(list, 0, i-1);

  // print it out
  printf("Sorted list:\n");
  for (i = 0; i < LISTSIZE; i++){
    printf("%d\n", list[i]);
  }
  return 0;
}

void quicksort(int a[], int low, int hi) {
  int pivot;
  if (hi <= low) {
    return;
  }
  pivot = partition(a, low, hi);
  quicksort(a, low, pivot-1);
  quicksort(a, pivot+1, hi);
}

int partition(int a[], int low, int hi) {
  int pivot, p_pos,i;
  p_pos = low;
  pivot = a[p_pos];
  for(i=low+1;i<=hi;i++){
    if(a[i] < pivot) {
      p_pos++;
      swap(a,p_pos,i);
    }
  }
  swap(a,low,p_pos);
  return p_pos;
}

void swap(int a[],int i ,int j) {
  int temp;
  temp = a[i];
  a[i] = a[j];
  a[j] = temp;
}
```

3 Assignment Submission Instructions

1. Submit your solution to myCourses before or by the due date.
2. Zip the two completed files (*count.asm* and *sort.asm*). The zipped file should be named <studentID>.zip. If my student ID is 123456789, then the zip file to submit will be named *123456789.zip*.
3. If you have special comments about your code, feel free to include a *readme.txt* text file in your zip containing your specific comments. Otherwise, simply comment your code as you normally would. Partial marks are easier to give (though no certainty that any will be given) if we see that you understand where your code is going wrong.
4. Your code *must* run and assemble, even if the final solution is not quite working. Part marks will be awarded for correct high-level control flow and use of conventions. If something is not working, comment-out the broken parts of code and write a comment or two about what you expect to happen, what is happening, and what the problem may be. **You are expected to follow conventions. Doing so will save you time and a headache during debugging, so its great for everyone!**