Laboratory 5 – Arrays and Pointers

I. Array and Pointers

I. Objectives

- To build an application that use Arrays and Pointers
- To learn how to use pointers and Arrays in a project

Activity Summary

1. Launch Xilinx SDK after opening and exporting the hardware platform
2. Create the Pointer application
3. Program the FPGA (if necessary), setup Run Configuration, and run the application
4. Repeat activity 3 for changes made to the application source file

ACTIVITIES

In the past four lab sessions, we have covered some very important aspects such as the implications of an interrupt service register and how to initiate GPIO as well as perform read/write operations.

Arrays

An array is a collection of data items, all of the same type, accessed using a common name. In other words, An array in C Programing can be defined as number of memory locations, each of which can store the same data type and which can be references through the same variable name. Array variables are declared identically to variables of their data type, except that the variable name is followed by one pair of square [ ] brackets for each dimension of the array.
A one-dimensional array is a structured collection of components (often called array elements) that can be accessed individually by specifying the position of a component with a single index value. Arrays must be declared before they can be used in the program. Here is the declaration syntax of one-dimensional array:

data_type array_name[array_size];

Here "**data_type**" is the type of the array we want to define, "**array_name**" is the name given to the array and "**array_size**" is the size of the array that we want to assign to the array. The array size is always mentioned inside the "[ ]".
Initializing of array is very simple in c programming. The initializing values are enclosed within the curly braces in the declaration and placed following an equal sign after the array name. Here is an example that declares and initializes an array of five elements of type int. Array can also be initialized after declaration. Look at the following code, which demonstrate the declaration and initialization of an array.

int age[5]={2,3,4,5,6};

It is not necessary to define the size of arrays during initialization e.g.

int age[ ]={2,3,4,5,6};

In C programming, arrays can be accessed and treated like variables in C. For example:

printf("%d", age[2]);

the above statement will print the third element of an array. Arrays can be accessed and updated using its index. An array of n elements, has indices ranging from 0 to n-1

Pointers

As shown in the previous Lab sessions, variables of different types (int, char) are stored in memory. A memory address is associated to each variable, and the value of the variable is stored at that memory address. In C, a special operator "&" (address of) is used to obtain the memory address at which a variable is stored. For example, given that a variable "b" is declared, the memory address where this variable is stored can be obtained by using the expression **"&b**". Next, we discuss a new type of variable, i.e. a pointer. A pointer is a variable whose value is interpreted as a memory address. This allows a pointer to read/change the contents of the memory address to which it points to. Since at a given memory address a specific data type is stored, when a pointer is declared, it is declared such that it points to a memory address where a specific data type is stored. In C, a pointer is declared as:

<data_type> *pointer_name;

The previous line declares a pointer (pointer name) that points to a memory address where a specific

<data_type> resides. By using the declared pointer, we can access (read or write) the memory address to which it points. The following line shows how to write to a memory location where the pointer points to:

***pointer_name = 5;**

Next, we show an example of how to use pointers.

Activity Summary

The purpose of this exercise is to get a better understanding of pointers. In particular, in this example, we shall use an example of a pointer that points to char type.
Create the Pointer application and a pointer_ex.c file based on the HelloWorld template.

In pointer_ex.c, create a Char type pointer variable. Create another variable and assign a value in the main, as shown in the figure below.

```
b=0;
address = &b;
*address = 5;
while (1==1){
      }
```

Figure 1

Build the project after you have done the changes, step through the code (Debug Mode). By adding expressions in the debug mode, you can trace the changes of different expressions, or variables, as the program is running. To open the expression view, from the Window menu, select Show View→Expressions. To add a new expression, click the "**Add new expression**" button and type the expression that you would like to evaluate. To evaluate the current value of a given variable, type the name of the variable. An example to evaluate the current value of the variable "`b`" and '**address**" is provided in Figure 2.

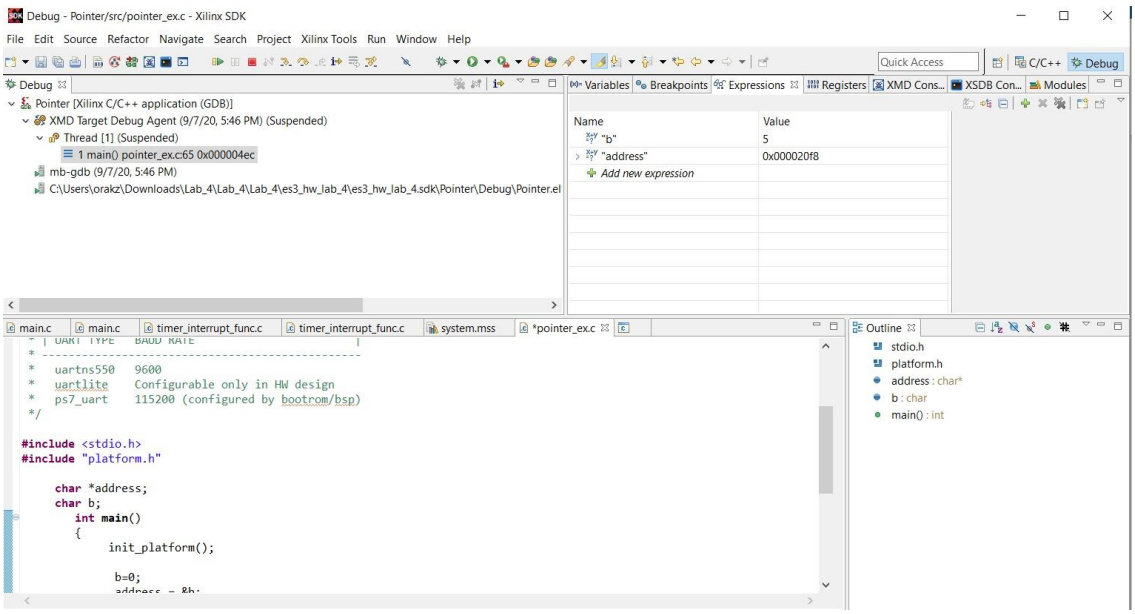

Figure 2

You can see the value of the variable is changed from 0 to 5 using the pointer variable "**address**".

II. Pointer Application

Objectives

- To learn how to use pointer variables
- To build an application to swap the values of variables using pointers

Activity Summary

1. Create an application named Swap_values
2. Write the code for the application
3. Program the FPGA (if necessary), setup Run Configuration, and run the application

Activities

The activities here involve swapping the values of two variables and displaying the new value on the 7-segment display. Create the Swap_values application. The file structure here is similar to that of the 7-segment display application and the files needed are already included in the "Swap_Values" inside "Lab5 - Codes folder". The following steps have already been implemented for you. Just follow them to understand what has been done.

1. Unfinished code template can be found in main.c
2. Create two variable and assign in initial value.
3. A swapping function has been defined and declared, call the function from the main. In a forever loop in the main(), output the new value of variable to the 7-segment display. [**Hint**: You can check the value of variable being swapped in the Debug mode.]
4. Run the application and the swapped value will be displayed.
5. Modify the Swap function; swap the values of two variable without using a third variable. Run the application to validate the results.

III. Pointer and Arrays

Objectives

- To build an application for sorting the values of an array
- To learn how to use pointers with Data structures

Activity Summary

1. Create a new application called Sorting
2. Write the application code
3. Program the FPGA (if necessary), setup Run Configuration, and run the application

Activities

The activities here involved sorting the values of an array in ascending or descending order using pointers. The file structure here is similar previous exercise and files needed are already included in the "Sorting" inside "Lab5 - Codes folder". The following steps are suggested:

1. Unfinished code template can be found in main.c.
2. Define an array of size MAX. Initially MAX is defined as 10 but it can be changed.
3. In the main, get the values from slide switches and store it in the array.
4. The code for swapping values has been written for you. All you have to do is to call the `sorting` function.
5. The code for displaying the sorted values on 7-segemnt display is is also written. Once completed, run the application.
    - If BTNL is pressed, sorted values will be displayed one by one on 7-Segment display.

EXTRA: Implement Merge Sorting algorithm on Basys3 board.

- User should be able to input values one by one until, BTNL is pressed.
- Upon pressing BTNL perform sorting and display the sorted values one by one.
- Note: Algorithm of merge sort can be found on the next page

MergeSort(arr[], l, r)
If r > l
1. Find the middle point to divide the array into two halves:
middle m = l+ (r-l)/2
2. Call mergeSort for first half:
Call mergeSort(arr, l, m)
3. Call mergeSort for second half:
Call mergeSort(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3:
Call merge(arr, l, m, r)

Source: https://en.wikipedia.org/wiki/Merge_sort; https://www.geeksforgeeks.org/merge-sort/