#cec32xNotes, cec32x_ws_14_WS02_logic_operations

WS 2. Logic Operations In C

In this workshop we play with logic operations in C.

Preparation

- Go to the **Dev tools** section under **Modules** of the CEC 320 Canvas page.
- Read the **Project file structure for simulator-based projects** section of `cec32x_devtool_20_organization_of_proj_files.pdf` to understand the file structure of the simulator-based template project.
- Download `expl_010_template_for_simulator_prjc_with_c.zip`, the zipped file for the simulator-based template project. Unzip the file to have a folder structure as described in the above pdf file.
- Change the project folder to `cec32x_ws02_logic_operations`. Change the name of the project management files to this name as well.
- Build and run the project to make sure there is nothing wrong.
- Then, you can remove all the “problem solving” code, the code “does something”, in the `main.c` file to prepare for the programming of the tasks of this workshop.
- Add the following code in the `main.c` file for data preparation

```
uint16_t a[8] = {0XFFFF};
uint16_t b[8] = {0X0000};

uint16_t mask;
uint16_t value;
```

Programming tasks

Task 1. Assign bits using mask without affecting other bits

(30 points)

Consider `uint16_t` values. Assume the right most bit of a value is bit0 and the right most one bit15. We need to practice of assigning values of bits 3 and 4 to be 0 and 1, respectively, without affecting the other bits. Perform this operation to both `a[0]` and `b[0]`. Print out the values of the new values of `a[0]` and `b[0]` in hexadecimal. See the handouts of WS 1 about how to print in hexadecimal. Note that you need to use variable `mask` to do the preparation of the bits to be changed and use variable `value` to assign the values to these bits. See the code of Prob 1 of HW 2 for how to use the **mask** and the **value**.

Task 2. Perform bitwise OR and AND operations

(10 points)

Perform the bitwise OR operation of the new values of `a[0]` and `b[0]` and save the result in `a[1]`. Also perform the bitwise AND operation of the new values of `a[0]` and `b[0]` and save the result in `b[1]`. Print out the values of `a[1]` and `b[1]` using the hexadecimal format.

Task 3. Perform bitwise NOT and XOR operations

(10 points)

Perform the bitwise NOT (inverse) operation of new values of `a[1]` and save the result in `a[2]`. Also perform the bitwise XOR operation of the new values of `a[1]` and `b[1]` and save the result in `b[2]`. Print out the values of `a[2]` and `b[2]` using the hexadecimal format.

Task 4. Perform LOGIC OR and AND operations

(10 points)

Repeat Task 2 using LOGIC OR and AND operations and save the results in`a[3]` and `b[3]`, respectively. Print out the values of `a[3]` and `b[3]` using the hexadecimal format.

Task 5. Show the results in the Memory Window

(20 points)

Print out the address of arrays `a` and `b` using the techniques learned in WS 1.

Then use these addresses to see the results of `a` and `b` in the Memory window directly. Note that you need to set up the data format to be unsigned 16 bit integer and the display should be hexadecimal. Take a screenshot of the memory and compare the values with the printout results.

Submission of your work

(20 points)

Each student needs to submit their own results—code snippet for the main function and the screenshots for the above printout results in the form of a pdf file. Put your name at the top of the page before the code snippet. Name your file as cec320_ws2_lastname_firstname(or initial).pdf. You also need to zip all your project files (not including the build files) into a single zip file and upload this file as well.