# ECE475 Computer Architecture

# LEGv8 Microprocessor Project

You are required to design, implement and test a simple 64-bit LEGv8processor using a Control Unit and Data Path design. Your design can be based on the block diagram shown below and discussed in class but must incorporate the changes and additions delineated in Figure 2. It should be implemented on the NexSys4 FPGA development board. You will conduct this project in groups of 4.

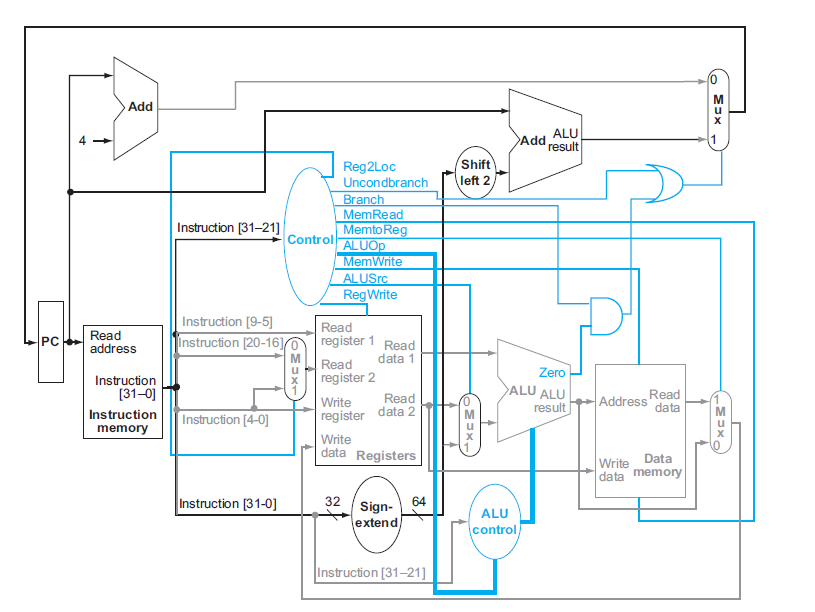
A single cycle implementation will be sufficient to pass the project but a multiple-cycle (fetch-decode-execute) State-Machine Control Unit design (with tailored execution cycles depending on the instruction) is required for full credit. A multiple cycle Control Unit may use more control lines than are shown in the single-cycle implantation below. 

Figure : General LEGv8 Processor design for single cycle implementation

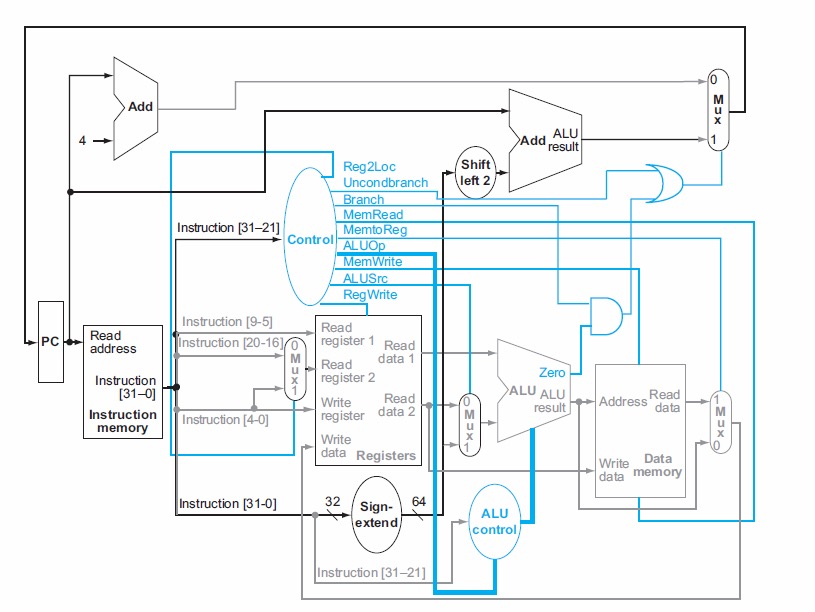
## II. Requirements

1. There should be 2 blocks of memory one 1024 x 32 bits for instructions and one 1024 x 64 bits for data.
2. There should be a register file consisting of 32 64-bit registers that follow the LEGv8 naming conventions. (Register 31 is the zero register for example)
3. There should be a 64-bit ALU with a zero output to determine whether or not the calculation resulted in a zero value.
4. Your data unit should have two changes from the LEGv8 processor above:
5. Data and memory are NOT byte-addressed. BRAM in the FPGA implementation of instruction memory has 32-bits in each memory location so for example Address 0 has 32-bits and Address 1 has 32-bits. This results in two changes in the data path:
   1. The Program Counter register (PC) is incremented by 1 instead of 4.
   2. There is no need to shift the branch address by 2 to the left to multiply it by 4. The branch address is accurate as it is.

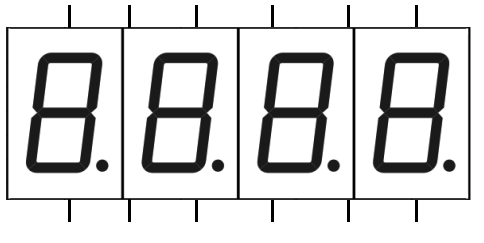
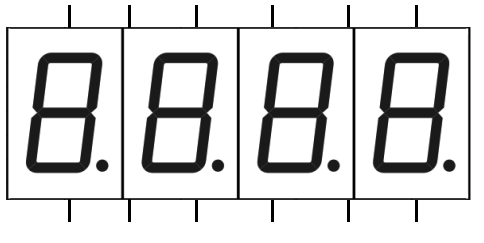
**N.B. THE FACT THAT THIS PROCESSOR DOES NOT HAVE BYTE ADDRESSING AFFECTS BOTH THE ASSEMBLY CODE PRODUCED (Data memory is not byte addressed) AND THE HARDWARE. ADJUST BOTH TO SUIT.**

1. In addition to writing data to memory, the processor should be able to write out to the 8 7-segment displays on the NexSys4 board. It should do this by writing to address 2048 and sending the results to a 7-segment display controller (Memory Mapped I/O). This is used to display values that are required in the code as well as to help in debugging your design.

The modified data path is shown below:



**1**



**7-seg Control**

ALU result(10)

en

Write data

we

Figure 2: Modified Data path for LEGv8 processor

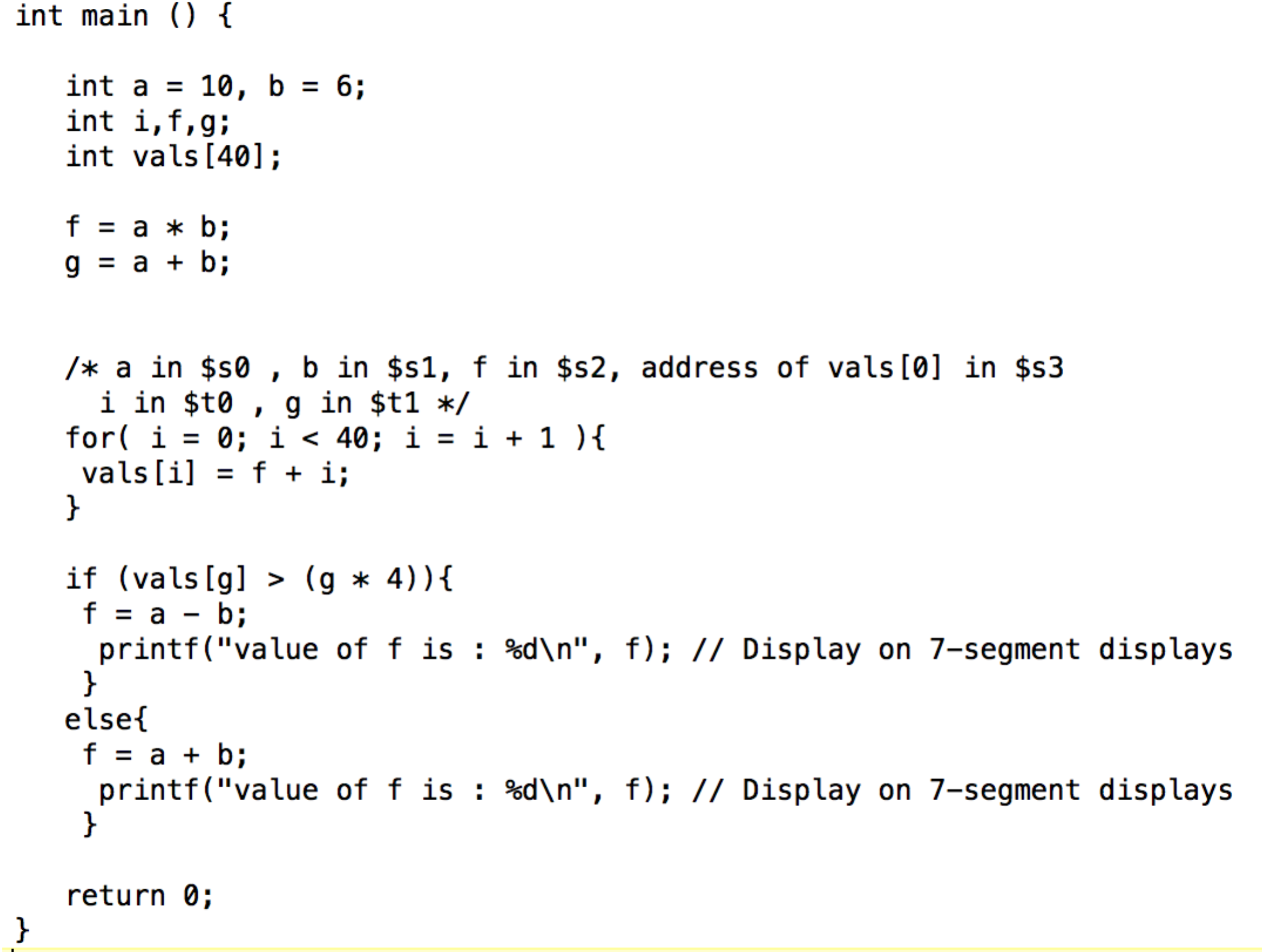
1. You are required to implement the following instructions in your microprocessor instruction set:

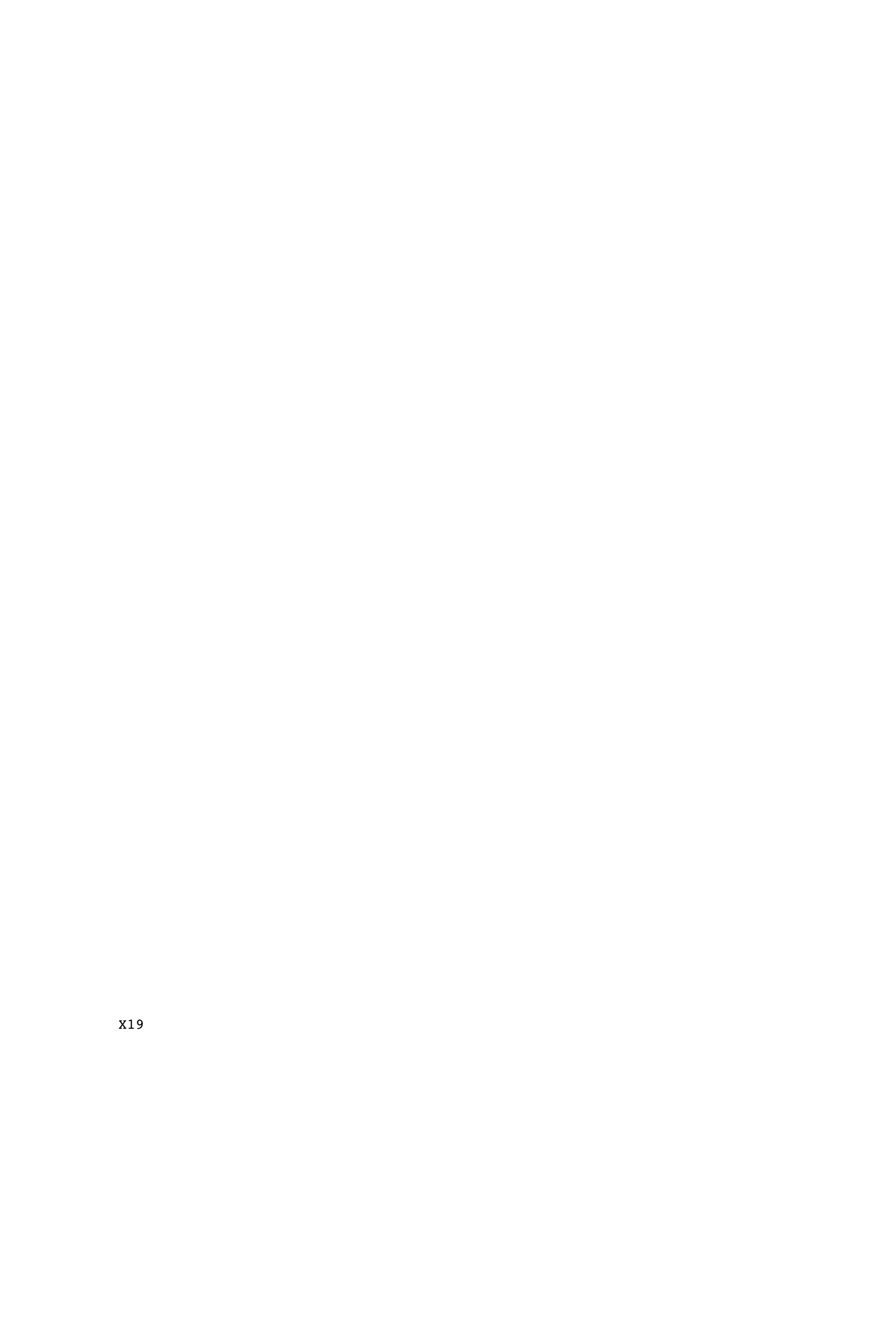
* **add**
* **addi**
* **sub**
* **subi**
* **lsl**
* **lsr**
* **mul**
* **cbz**
* **cbnz**
* **and**
* **orr**
* **eor**
* **ldur**
* **stur**
* **b**

## III. Demonstration

To demonstrate the LEGv8 processor, you should convert the following code to LEGv8 machine code, load it into the instruction memory and run it. Depending on the values of a and b, the value displayed on the 7-segment displays should vary.

This demonstration should take place on or before **May 10th 2017.**



X21

X21

X20

X9

X10

X19

Write-up

The project should be written up with the following sections:

1) Problem specification.

Here you write again in your own words the problem itself. This may be accompanied with diagrams and figures.

2) Requirement Specification.

Here you determine what modules are required to complete the design. Although the block diagram is given to you,

1. You need to explain what each of the blocks does and why it is there.
2. You need to explain the internal modules of each block (e.g. register file)

3) Design Specification

Here you explain why the modules are connected the way they are in the diagram and expand on your Control Unit. You need to explain here how each of the instructions (or instruction types) will be implemented in your design.

4) Implementation and Testing

Here you need to show your LEGv8 assembly code and resulting Machine code that was coded into the instruction memory. You also need to discuss your means of testing (simulation or real testing) and how you went about debugging your design.

5) Discussion

Here you discuss what you learnt, what you did well and what mistakes you made in your design. You should reflect here on what parts of the design process you would do differently and why and what parts of the design itself you would improve. Here you should also clearly state the contributions of each partner to the project.

Extra Credit Options

1. Adjust your design so that Data and Instruction Memory and not separated
2. Write an assembler for you code to translate your assembly into machine code and store it in a .coe file to instantiate BRAM