EE-222: Microprocessor Systems

Assignment # 1

6 Bit - Microprocessor

Group Members

Name	Reg. No
Muhammad Ahmed Mohsin	333060
Muhammad Umer	345834
Tariq Umar	334943

Report

Introduction

We developed a six bit microprocessor using different hardware components and using the basic knowledge of digital logic design and microprocessor systems. We used different hardware components like 2 – 1 4 bit MUX, 3 bit registers, an ALU unit to perform arithmetic operations such as addition and subtraction, Arduino for power supply and seven segment display and a program counter 74163 (4 bit program counter). Another special use case of Arduino in our particular circuit is of Instruction Decoding; we utilized the On-Chip ROM of Arduino, organized as 4096 x 8 bits, to store the flash program.

Design Methodology

- **4 to 6 Bit Mapping**

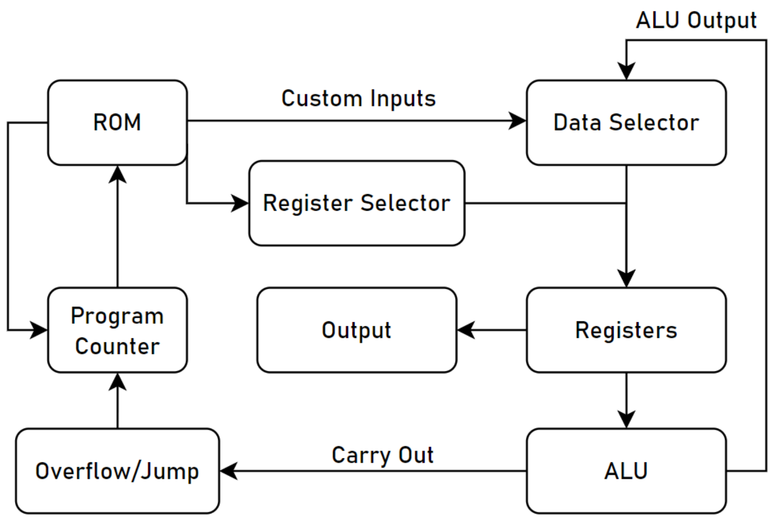
We were tasked to map the logic behind the development of a 4-bit microprocessor to a 6-bit microprocessor. In other words, we want to extend the storage capacity to what our outputs can build up to; instead of a maximum value of 2^4, we wish to make it 2^6. This is accomplished by concatenation of different modules throughout the circuit.

1. **Adder:** Two 4-Bit Adders (7483) are concatenated by connecting the C_{OUT} of the first adder to the C_{IN} of the second adder and leaving any extra internal full adders of the second adder unused.
2. **Registers:** 6-Bit D-Flip Flops are already available on an IC chip and hence do not need any external modifications.
3. **Data Selector:** Two 2:1 4-Bit Line MUX (74157) is concatenated by making the trigger conditions (enable, selection line) of the data selectors common.

- **ROM**

Each instruction being 8-bit wide restricts us to use a ROM that is organized as $Memory \times\ 8\,bits$. Since there isn't much that can be done with just 6-bit numbers, we are not deeply concerned with the total memory locations available to us, as such, we use the $4096 \times 8\ bit$ on-chip ROM soldered on Arduino MEGA2560.

Flowchart



Module Description

- **ROM & Program Counter**

The module that directs and drives the rest of the microprocessor; the user burns a specific code in HEX format to the ROM and the program counter's output addresses the stored instructions through its outputs. On each clock cycle, our counter increments by one, thereby also incrementing the address and the ROM decodes the next instruction in the appropriate memory location.

- **Data Selector**

Data selector is used to either update the storage units of our microprocessor with the outputs of the logic unit or it can also load custom values to the register. The two 6-bit input lines to the multiplexer are the custom inputs and the ALU outputs, amongst which the selector chooses what to store depending on the instruction stored in the ROM.

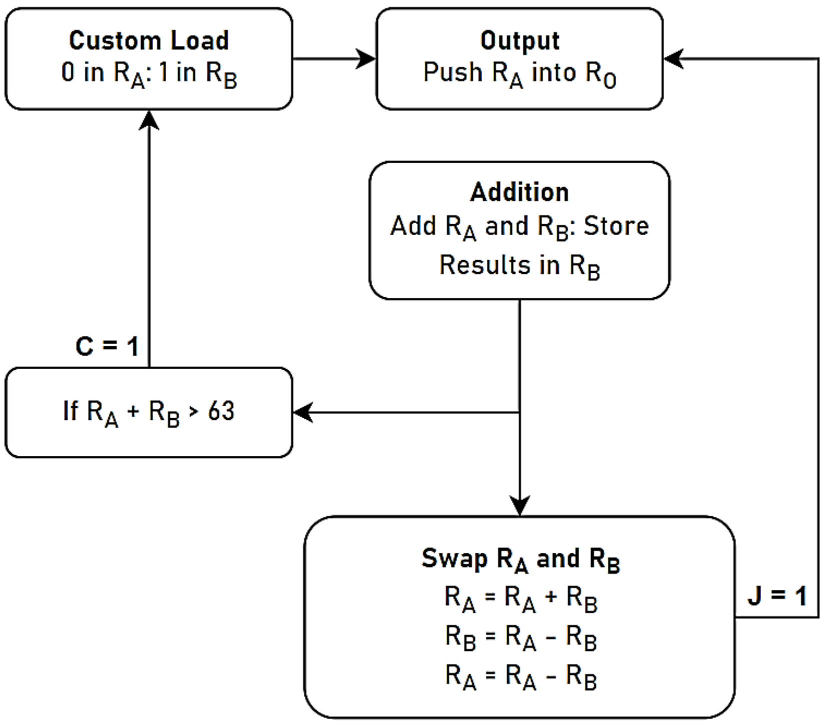
- **Registers & Register Selector**

Storage units of the proposed microprocessor; it stores the values coming from the output of the data selector into registers enabled by the register selector. The register selector is a simple 2 – 4 line demultiplexer and is used to enable any among the three registers available in our microprocessor, or don't enable any at all. The selection is, once again, directly dependent on the instruction stored in the ROM.

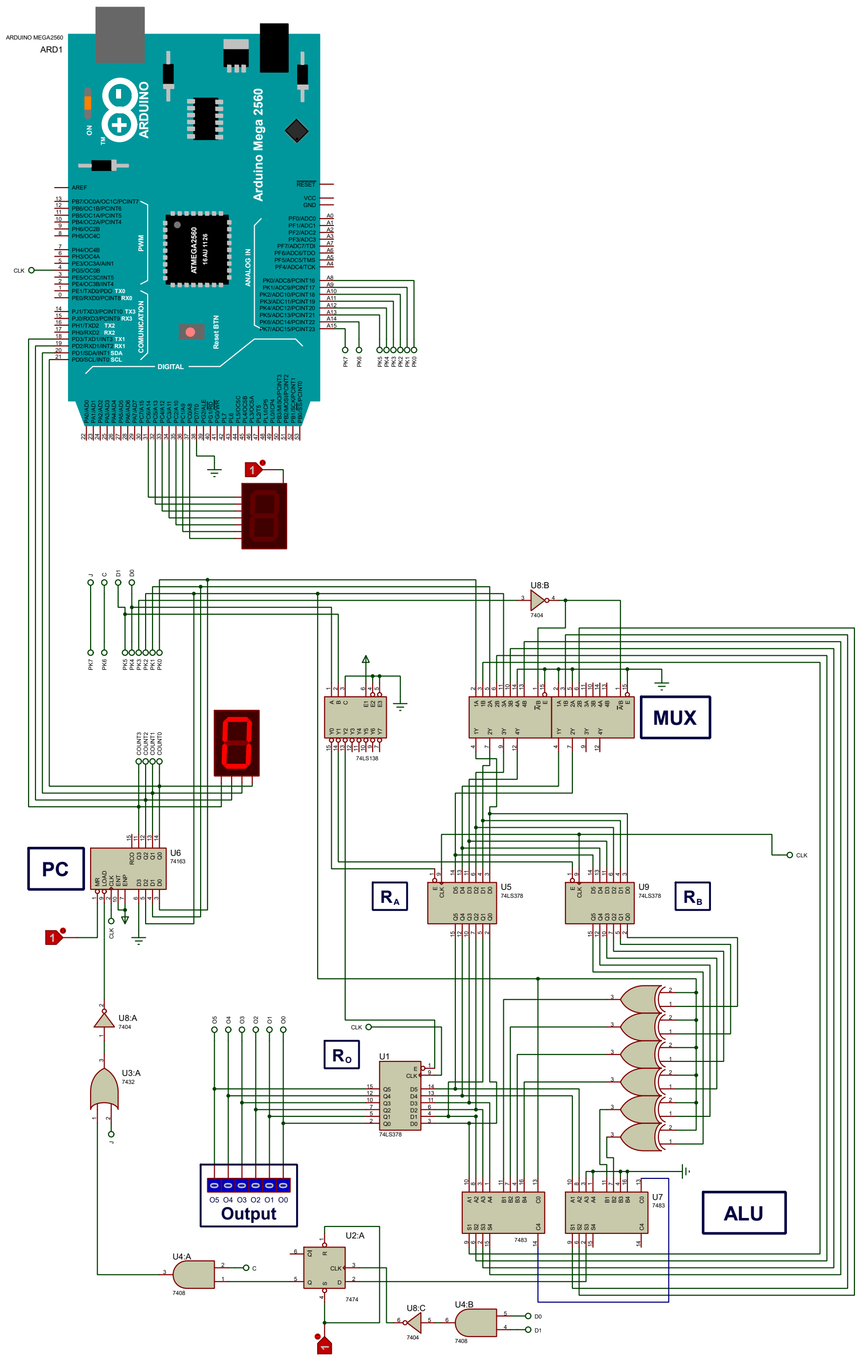
- **ALU**

Logical unit of the microprocessor; it is used to perform all the arithmetic calculations that are required to implement algorithms, etc. It comprises of two 4-bit adders cascaded together with the carry out of one going as an input to the other, and their outputs are fed back into the data selector.

Fibonacci Series (Pictorial Representation)



Simulation Schematic



Conclusion

In this assignment, we enhance our skills on hardware as well as software and learn to implement the basic knowledge to develop a microprocessor from scratch. We built a microprocessor that can perform arithmetic operations such as addition, subtraction, etc. as well as output famous series such as Fibonacci Series, within the ranges of 6 – bits. The concept of a program counter, along with the ability to custom load values for the counter to address to, makes looping/branching possible and allows us to do a lot more with just 6 – bits available.

A special use – case of Arduino can be seen in our implementation of a 6 – bit Microprocessor; the On-Chip ROM of Arduino is acting as our flash memory, and it is also used to convert a 6 bit binary number to 2 4 – bit BCD numbers being displayed on a 7SEG display.