|  |  |
| --- | --- |
|  | **Qatar University**  **College of Engineering**  **Department of Computer Science and Engineering** |

**CMPE 363 Computer Architecture and Organization II**

**Course Project Report**

**Fall 2023**

Project Title  
Hardwired Control CPU with Simple Input-Output Interface Module

Table of Contents

[1 Introduction 3](#_Toc152450778)

[2 RTL Description of the CPU 3](#_Toc152450779)

[3 Project Testing Program 4](#_Toc152450780)

[4 Data Unit (DU) Design 5](#_Toc152450781)

[4.1 DU Circuit Diagram 5](#_Toc152450782)

[4.2 Description 5](#_Toc152450783)

[5 Hardwired Control Unit (CU) Design 6](#_Toc152450784)

[5.1 CU Circuit Diagram 6](#_Toc152450785)

[5.2 CU Description 6](#_Toc152450786)

[6 I/O System 7](#_Toc152450787)

[6.1 I/O Interface Module Circuit Diagram 7](#_Toc152450788)

[6.2 I/O Interface Module Description 7](#_Toc152450789)

[7 Running the Simulated Computer 7](#_Toc152450790)

[8 Contributions of Each Team Member 8](#_Toc152450791)

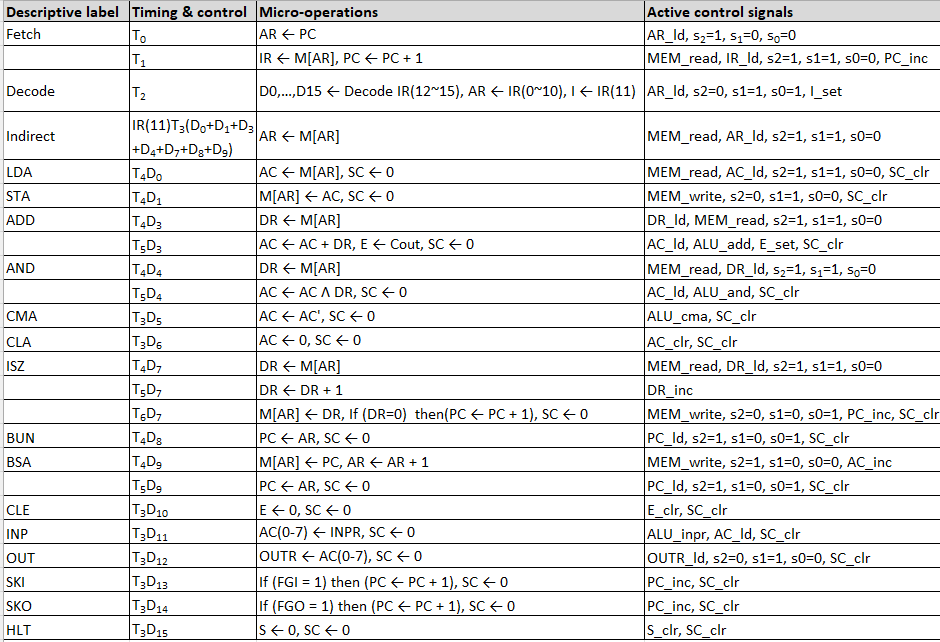
[9 Reflection and Conclusion 8](#_Toc152450792)

# Introduction

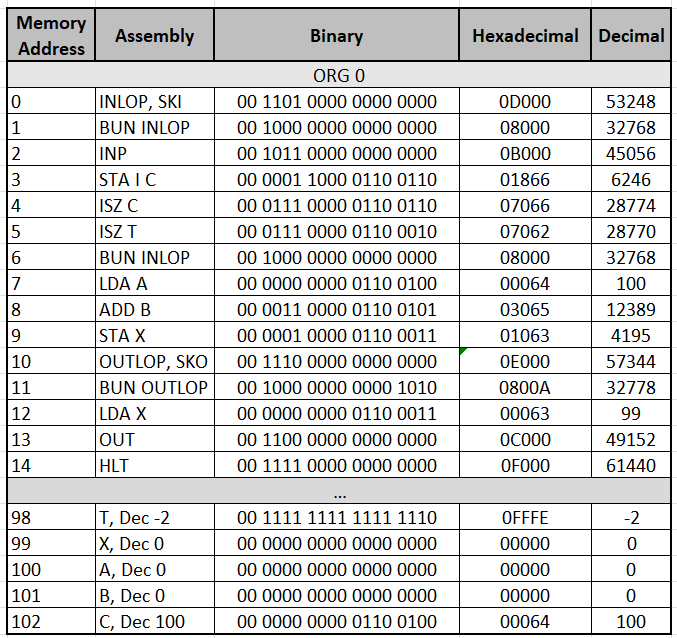
In this project, we applied various concepts we learned throughout both Computer Architecture courses 1 and 2, in order to build a basic 16-bit computer, composed of a memory unit, a data unit, a control unit, ALU, input/output modules, and a system bus.

The computer functions according to the descriptions provided in the following section, focusing first and foremost on the “Timing & Control” column, which associates an opcode (denoted by ) with a common clock cycle (denoted by ), using AND gates. The system then uses more logic gates, in addition to multiplexers, some priority encoders, and status flags to execute a certain operation (increment, load, or clear) in a certain register.

# RTL Description of the CPU



# Project Testing Program



# Data Unit (DU) Design

## DU Circuit Diagram

A diagram of a computer

Description automatically generated

(input/output not included in this section)

## Description

The Data Unit is the largest unit in our computer. As shown in the diagram above, it’s divided into 5 parts:

1. **Memory**: Here, the HEX code of the program is written into the RAM before starting the simulation. Each instruction, consisting of 4 HEX values or 16 binary bits, is read by the Memory, which loads the opcode bits into the Instruction Register. The memory also reads values AC, DR, or PC, depending on the instruction, using a priority encoder connected to a multiplexer.
2. **Registers**: This part contains three 16-bit registers: IR, AC, and DR, and two 11-bit registers: PC and AR, as per the project description. The IR reads the four opcode bits from the memory and uses a demultiplexer to translate them into 1 of 16 “D” values. All other register use the D and T values, combined with logic gates, multiplexers, and adders to perform load, increment or clear, as explained in the introduction. The functions of all registers are identical to what we studied in the course. For instance, PC uses comparators and an adder to “skip” during instructions ISZ, SKI, and SKO. In addition to the value of DR and the input/output flags.
3. **ALU**: Performs 3 operations: Addition of AC and DR, loading from INPR to AC, and complements AC.
4. **Flags**: LEDs to display the status of start/stop (S), zero ALU output (Z), indirect bit (I), extended/carry bit (E), and input/output flags (FGI/FGO).

# Hardwired Control Unit (CU) Design

## CU Circuit Diagram

A diagram of a circuit

Description automatically generated

## CU Description

The Control Unit consists of 3 T-flip-flops, which use a decoder to convert the clock cycle number to binary values displayed through the outputs of 8 D-flip-flops. Logic gates are used to determine the D and T values at which the sequence counter resets.

# I/O System

## I/O Interface Module Circuit Diagram

A diagram of a circuit

Description automatically generated

## I/O Interface Module Description

A simple I/O interface module consisting of two 8-bit registers. INPR uses switches (flipped on/off manually) to read input values, and OUTR uses LEDs to display output values. Zero values of the registers are used to determine the status of the FGI and FGO flags.

# Running the Simulated Computer

Using the sample programs given in the course description.

1. Addition: (HEX) 17+60=77. We can see the value 77 stored in memory.

A close-up of a number

Description automatically generated

1. Input/Output: We can see the input value on the switches is equal to the output value on the LEDs.

A diagram of a circuit

Description automatically generated

1. Loop: We see the value of Y, set to FFFB (-5 decimal) in the beginning of the program, increasing until it reaches zero, causing the computer to halt.

A close-up of numbers

Description automatically generated A close-up of numbers

Description automatically generated A grey rectangular with black numbers

Description automatically generated

1. We also tested using our testing program from Section 3 and were able to add to values from the input switches and display the result on the output LEDs.

A diagram of a circuit

Description automatically generated

# Reflection and Conclusion

The project was interesting and challenging at the same time. We saw how the concepts we studied in the lectures can be applied and turned into a functioning CPU with an I/O module and all other units required.

We faced some difficulties with parts such as the common bus and the I/O module. However, the most demanding part of the project was to debug/troubleshoot the entire circuit in order to make it function as required. This included making small changes all over the project, such as changing the RTL descriptions of certain instructions, which lead to changes in all of the units.