|  |  |
| --- | --- |
|  | **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

Submitted By:

1. << Muhammad Muhsen Khan – 202105963 >>
2. << Laith Nasrallah – 201906328 >>
3. << Abdallah Irhimeh – 202107313 >>

Table of Contents

[1 Introduction 2](#_Toc147395895)

[2 RTL Description of the CPU 2](#_Toc147395896)

[3 Project Testing Program 4](#_Toc147395897)

[4 Data Unit (DU) Design 5](#_Toc147395898)

[4.1 DU Circuit Diagram 5](#_Toc147395899)

[4.2 Description 5](#_Toc147395900)

[5 Hardwired Control Unit (CU) Design 5](#_Toc147395901)

[5.1 CU Circuit Diagram 5](#_Toc147395902)

[5.2 CU Description 5](#_Toc147395903)

[6 I/O System 5](#_Toc147395904)

[6.1 I/O Interface Module Circuit Diagram 5](#_Toc147395905)

[6.2 CU Description 5](#_Toc147395906)

[7 Running the Simulated PSC 5](#_Toc147395907)

[8 Contributions of Each Team Member 6](#_Toc147395908)

[9 Reflection and Conclusion 6](#_Toc147395909)

# Introduction

This project aims to create a hardware controlled CPU with a simple I/O interface. The I/O consists of 8 switches for input and 8 LEDs, with two buttons, one to activate input and one to activate output. The process of input detections is program controlled where a CPU program takes over the job of checking status flags and looping. There will also be a control unit and data unit along with an ALU that will reside inside the data unit. The control unit will have all the control signals with logic gates as well as the sequence counter and instruction decoder, it will take in status bits and the instruction. The data unit will have all registers and status flip flops along with the ALU and RAM. It will receive control signals from the control unit and update data as necessary, it will also read the program off the RAM and send those instructions for execution. The ALU is a simple circuit with four operations, ADD, AND, CMP, and REG\_IN which are controlled by control signals from the CU. The I/O interface is responsible for interfacing the CU and DU with the input and output devices.

# RTL Description of the CPU

|  |  |  |  |
| --- | --- | --- | --- |
| **Descriptive label** | **Timing & control** | **Micro-operations** | **Active control signals** |
| Fetch | T0 | AR ← PC | AR\_LD, s2=1, s1=0, s0=0 |
|  | T1 | IR 🡨 M[AR], PC 🡨 PC + 1 | PC\_INC, IR\_LD, M\_READ, s2=1, s1=1, s0=0 |
| Decode | T2 | Decode IR(12-15), I 🡨 IR(11), AR 🡨 IR(0-10) | AR\_LD, IF (IR(11) = 1) then I\_SET, IF (IR(11) = 0) then I\_CLR, s2=0, s1=1, s0=1 |
| Indirect | IT3 | AR 🡨 M[AR] | AR\_LD, M\_READ, s2=1, s1=1, s0=0 |
| LDA (Load AC from Memory) | D0T4 | DR 🡨 M[AR] | DR\_LD, M\_READ, s2=1, s1=1, s0=0 |
|  | D0T5 | AC 🡨 DR, SC 🡨 0 | AC\_LD, SC\_CLR, ALU(DR), X=0 |
| STA (Store AC in Memory) | D1T4 | M[AR] 🡨 AC, SC 🡨 0 | M\_WRITE, SC\_CLR, s2=0, s1=1, s0=0 |
| ADD (Add AC to Memory) | D3T4 | DR 🡨 M[AR] | DR\_LD, M\_READ, s2=1, s1=1, s0=0 |
|  | D3T5 | AC 🡨 AC + DR, SC 🡨 0 | AC\_LD, ALU(ADD), SC\_CLR, X=0 |
| AND(AND AC with memory) | D4T4 | DR 🡨 M[AR] | DR\_LD, M\_READ, s2=1, s1=1, s0=0 |
|  | D4T5 | AC 🡨 AC Ʌ DR, SC 🡨 0 | AC\_LD, ALU(AND), SC\_CLR, X=0 |
| CMA(Complement AC) | D5T4 | AC 🡨 AC’, SC 🡨 0 | AC\_LD, ALU(CMA), SC\_CLR |
| CLA(Clear AC) | D6T4 | AC 🡨 0, SC 🡨 0 | AC\_CLR, SC\_CLR |
| ISZ(Increment memory and skip if 0) | D7T4 | DR 🡨 M[AR] | DR\_LD, M\_READ, s2=1, s1=1, s0=0 |
|  | D7T5 | DR 🡨 DR + 1 | DR\_INC |
|  | D7T6 | AC 🡨 DR, M[AR] 🡨 DR | AC\_LD, ALU(DR),M\_WRITE, s2=0, s1=0, s0=1, X=0 |
|  | D7T7 | If (Z=1) then PC 🡨 PC + 1, SC🡨 0 | PC\_INC, SC\_CLR |
| BUN(Unconditional branch) | D8T4 | PC 🡨 AR, SC 🡨 0 | PC\_LD, SC\_CLR, s2=1, s1=0, s0=1 |
| BSA(Branch and save return address) | D9T4 | M[AR] 🡨 PC, AR 🡨 AR + 1 | M\_WRITE, AR\_INC, s2=1, s1=0, s0=0 |
|  | D9T5 | PC 🡨 AR, SC 🡨 0 | PC\_LD, SC\_CLR, s2=1, s1=0, s0=1 |
| CLE(Clear E) | D10T4 | E 🡨 0, SC 🡨 0 | E\_CLR, SC\_CLR |
| INP(Input character) | D11T4 | AC(0-7) 🡨 INPR, FGI 🡨 0, SC 🡨 0 | ALU(INPR), AC\_LD, FGI\_CLR, SC\_CLR, X=1 |
| OUT(Output character) | D12T4 | OUTR 🡨 AC(0-7), FGO 🡨 0, SC 🡨 0 | OUTR\_LD, FGO\_CLR, SC\_CLR, s2=0, s1=1, s0=0 |
| SKI(Skip on input flag) | D13T4 | If (FGI=1) then PC 🡨 PC + 1, SC 🡨 0 | If (FGI = 1) then PC\_INC, SC\_CLR |
| SKO(Skip on output flag) | D14T4 | If (FGO=1) then PC 🡨 PC + 1, SC 🡨 0 | If (FGO = 1) then PC\_INC, SC\_CLR |
| HLT(Halt execution) | D15T4 | S 🡨 0, SC 🡨 0 | S\_CLR, SC\_CLR |

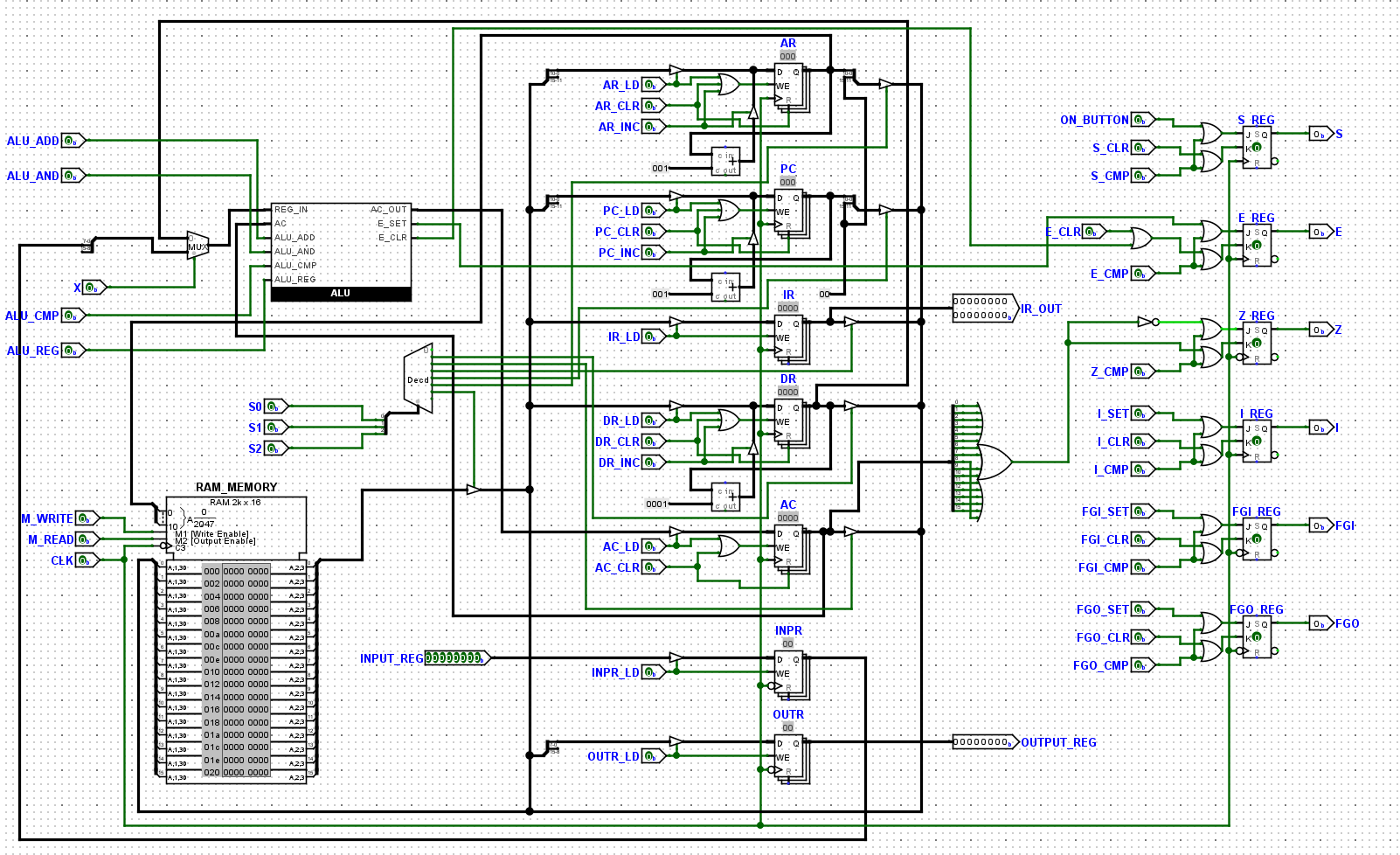
# Project Testing Program

Input/Output program: **(USE COLOUR CODING TO FIND OUT BRANCH LOCATIONS)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory Address** | **Assembly** | **Binary** | **Hexadecimal** | **Decimal** |
|  | ORG 0 |  |  |  |
| 0 | W, SKI | 1101 xxxx xxxx xxxx | D000 | 53248 |
| 1 | BUN W | 1000 0000 0000 0000 | 8000 | 32768 |
| 2 | INP | 1011 xxxx xxxx xxxx | B000 | 45056 |
| 3 | STA A | 0001 0000 0110 0100 | 1064 | 4196 |
| 4 | X, SKI | 1101 xxxx xxxx xxxx | D000 | 53248 |
| 5 | BUN X | 1000 0000 0000 0100 | 8004 | 32772 |
| 6 | INP | 1011 xxxx xxxx xxxx | B000 | 45056 |
| 7 | STA B | 0001 0000 0110 0101 | 1065 | 4197 |
| 8 | CLA | 0110 xxxx xxxx xxxx | 6000 | 24576 |
| 9 | Y, ISZ D | 0111 0000 0110 0111 | 7067 | 28775 |
| 10 | BUN Z | 1000 0000 0001 1110 | 801E | 32798 |
| 11 | LDA E | 0000 0000 0110 1000 | 0068 | 104 |
| 12 | F, SKO | 1110 xxxx xxxx xxxx | E000 | 57344 |
| 13 | BUN F | 1000 0000 0000 1100 | 800C | 32780 |
| 14 | OUT | 1100 xxxx xxxx xxxx | C000 | 49152 |
| 15 | HLT | 1111 xxxx xxxx xxxx | F000 | 61440 |
|  | ORG 30 |  |  |  |
| 30 | Z, LDA E | 0000 0000 0110 1000 | 0068 | 104 |
| 31 | ADD C | 0011 1000 0110 0110 | 3866 | 14438 |
| 32 | STA E | 0001 0000 0110 1000 | 1068 | 4200 |
| 33 | ISZ C | 0111 0000 0110 0110 | 7066 | 28774 |
| 34 | BUN Y | 1000 0000 0000 1001 | 8009 | 32777 |
|  | ORG 100 |  |  |  |
| 100 | A, Dec 0 | 0000 0000 0000 0000 | 0000 | 0 |
| 101 | B, Dec 0 | 0000 0000 0000 0000 | 0000 | 0 |
| 102 | C, Dec 100 | 0000 0000 0110 0100 | 0064 | 100 |
| 103 | D, Dec -2 | 1111 1111 1111 1101 | FFFD | 65534 |
| 104 | E, Dec 0 | 0000 0000 0000 0000 | 0000 | 0 |

# Data Unit (DU) Design

## DU Circuit Diagram



## Description

The DU includes a 2k x 16 RAM, ALU, 3 bit decoder, 2 input MUX, 7 registers (AR and PC 11 bits, IR, DR, and AC 16 bits, INPR and OUTER 8 bits), and also 5 status flip flops of the type JK.

The ALU carries out the ADD, AND, CMP, DR, and INPR operations where ADD adds DR to AC, AND does the AND operation between DR and AC, CMP complements the AC, DR loads the value of DR into AC, and INPR loads the value of INPR to AC.

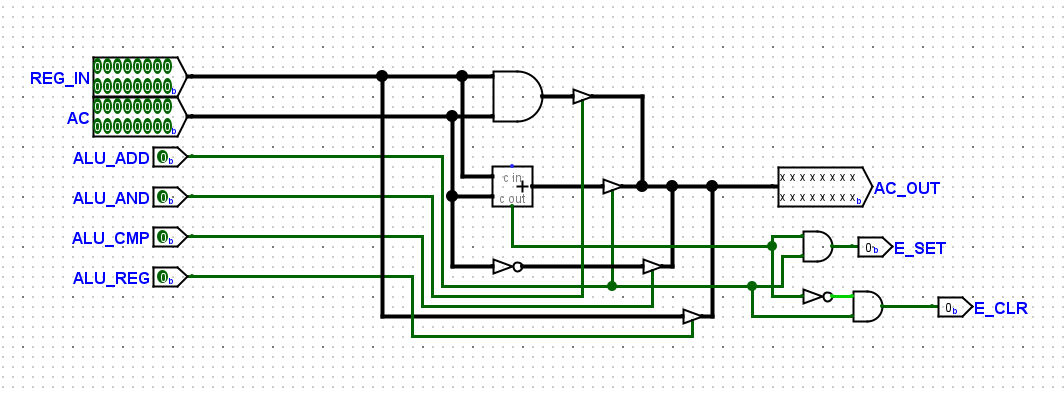
The DU receives control signals from the CU and synchronizes with the clock to carry out those signals, such signals dictate varies aspects of the DU such as which register’s value should be carried on the common bus, which register should receive the values from the bus, whether the memory is in read or write mode, which ALU output to choose, which flip flop to either SET, CLEAR, or COMPLEMENT.

# Hardwired Control Unit (CU) Design

## CU Circuit Diagram

A diagram of a machine

Description automatically generated



ALU

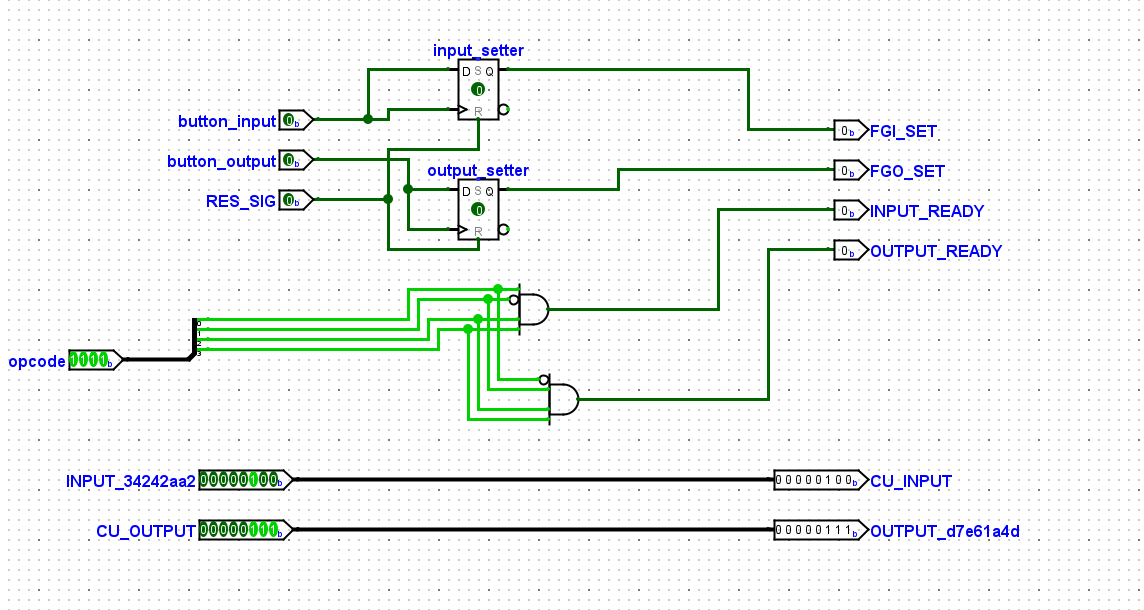
## CU Description

# The CU contains a 3 bit counter set on count mode (the Reset pin of the counter has been set to synchronous so the counter is only reset with the clock allowing other instructions to execute at the same time as a sequence counter clear instruction), a 3 bit decoder for timing signals, a 4 bit decoder for opcode signals, a central circuit for activating control signals, and an array of control signal outputs on the right.

The CU is responsible for sending the control signals to the DU and it does this due to a series of logic gates controlling which signal turns on at which moment. The timing signals and opcode signals are ANDed in the center according to the RTL description table. The OR gates are responsible for turning on any of the control signals as soon as one of the RTL description table signals corresponds to a signal that needs to be activated in that moment. For example, the S2 signal needs to activate for 10 signals which include IT3 and D0T4 so these outputs are put in the input of the OR gate connected to the S2 signal so as soon as one of the conditions is met the S2 signal activates.

# I/O System

## I/O Interface Module Circuit Diagram



## I/O Interface Module Description

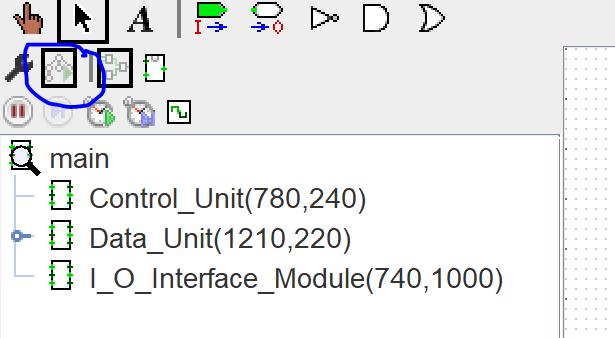
The I/O interface contains inputs for an input button and output button as well as a signal that resets the d flip flops responsible for setting FGI or FGO. The opcode is also inputted and checked for the SKI and SKO instructions to allow an LED to flash letting the user know that the input button can be pressed to pass the input or the output button can be pressed to receive the output. The bottom two lines are only responsible for passing the inputs from the user to the DU and passing the outputs from the CU to the user.

# Running the Simulated Computer

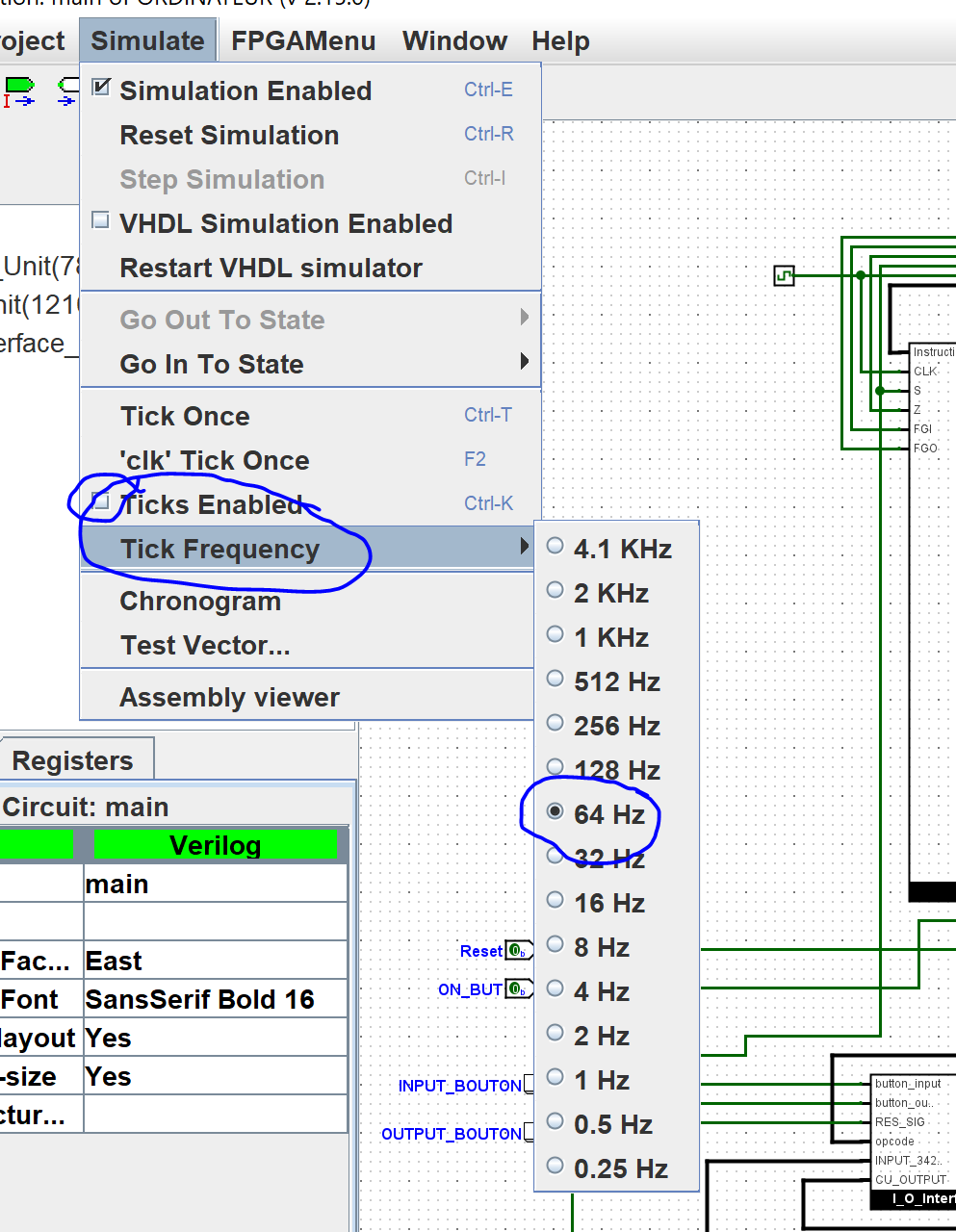
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory Address** | **Assembly** | **Binary** | **Hexadecimal** | **Decimal** |
|  | ORG 0 |  |  |  |
| 0 | W, SKI | 1101 xxxx xxxx xxxx | D000 | 53248 |
| 1 | BUN W | 1000 0000 0000 0000 | 8000 | 32768 |
| 2 | INP | 1011 xxxx xxxx xxxx | B000 | 45056 |
| 3 | STA A | 0001 0000 0110 0100 | 1064 | 4196 |
| 4 | X, SKI | 1101 xxxx xxxx xxxx | D000 | 53248 |
| 5 | BUN X | 1000 0000 0000 0100 | 8004 | 32772 |
| 6 | INP | 1011 xxxx xxxx xxxx | B000 | 45056 |
| 7 | STA B | 0001 0000 0110 0101 | 1065 | 4197 |
| 8 | CLA | 0110 xxxx xxxx xxxx | 6000 | 24576 |
| 9 | Y, ISZ D | 0111 0000 0110 0111 | 7067 | 28775 |
| 10 | BUN Z | 1000 0000 0001 1110 | 801E | 32798 |
| 11 | LDA E | 0000 0000 0110 1000 | 0068 | 104 |
| 12 | F, SKO | 1110 xxxx xxxx xxxx | E000 | 57344 |
| 13 | BUN F | 1000 0000 0000 1100 | 800C | 32780 |
| 14 | OUT | 1100 xxxx xxxx xxxx | C000 | 49152 |
| 15 | HLT | 1111 xxxx xxxx xxxx | F000 | 61440 |
|  | ORG 30 |  |  |  |
| 30 | Z, LDA E | 0000 0000 0110 1000 | 0068 | 104 |
| 31 | ADD C | 0011 1000 0110 0110 | 3866 | 14438 |
| 32 | STA E | 0001 0000 0110 1000 | 1068 | 4200 |
| 33 | ISZ C | 0111 0000 0110 0110 | 7066 | 28774 |
| 34 | BUN Y | 1000 0000 0000 1001 | 8009 | 32777 |
|  | ORG 100 |  |  |  |
| 100 | A, Dec 0 | 0000 0000 0000 0000 | 0000 | 0 |
| 101 | B, Dec 0 | 0000 0000 0000 0000 | 0000 | 0 |
| 102 | C, Dec 100 | 0000 0000 0110 0100 | 0064 | 100 |
| 103 | D, Dec -2 | 1111 1111 1111 1101 | FFFD | 65534 |
| 104 | E, Dec 0 | 0000 0000 0000 0000 | 0000 | 0 |

This is the program used for testing, it takes in two numbers and stores them, then using a loop utilizing the ISZ instruction and indirect address it adds the two numbers and outputs the result on the LEDs. There are a couple sets that need to be carried out before being able to test the program **(INCASE OF ANY MISTAKES ON STARTING UP THE CPU SCROLL BELOW FOR STEPS)**:

1. Open the main circuit and click on open hierarchy panel on the top of the application as shown below:



1. Go to the data\_unit and right click on the RAM then click “edit contents” then add the above instructions in the right memory location, ORG 30 is 1E in hex and ORG 100 is 64 in hex
2. Go back to main and enable ticks and add a good speed such as 32 bits as show below



1. Once the clock has started in the main circuit put the reset circuit high for a few moments then put it to low to reset all registers (reset happens after rising edge of clock)
2. Then press the ON\_BUTTON to turn the CPU on, when the input light is flashing flip the switches to an input of your choice and then press the INPUT\_BUTTON, the input light will continue flashing for another input and repeat the same step again.
3. Then the input light will stop flashing and after a bit of time after all instructions have been executed the output button will flash, press the output button and the LEDs should light up in the binary representation of the sum of the two numbers you passed initially.

**INCASE OF MISTAKES WHILE CLOCK TICKS ARE RUNNING PRESS THE RESET BUTTON AND REPEAT THE ABOVE STEPS**

# Contributions of Each Team Member

Muhammad Muhsen Khan: Created RTL description, created test assembly code and created report. Logisim contributions include creating the Data\_Unit and helping with main. [40%]

Abduallah Irhimeh: Created Control\_Unit and I/O module. [30%]

Laith Nasrallah: Created Main and ALU. [30%]

# Reflection and Conclusion

This project taught us a lot about the inner workings of a CPU and how clock signals with decoders are used to drive various components and make various things happen. The control unit helped us discover how to control CU signals and how to decode instructions, the data unit helped us understand the memory architecture of a computer and the main helped us visualize how the various components interconnect and interact with each other. This project was decently difficult and required us a couple days to complete. The instructor was very helpful with understanding the project and answering any questions we had. The main challenge in this project was testing as it requires frequent changes in the RAM to set up the test program.