**CEG2136**

**Lab 4**

**Basic Computer Organization**

Zi Qi Lin – 7889797

Ian Rodriguez – 7737416

TAs: Mohammed, Moayad, Heli, Yuktika

Nov. 10, 2015

**Theory**

Introduction

The purpose of this lab is to analyse the structure of a basic computer. We devised and designed its control unit, and will use opcodes to write simple programs in machine code. The design must function in simulation, and also on the DE2-115 Altera development board.

Problem

We are to implement a basic computer. The memory of the computer has a capacity of 26 words of 8 bits. Memory-reference instruction in this lab is 2 byte long. The msb byte carries the opcode, and the lsb byte carries the operand address. There are three types of reference instructions, register reference instructions, memory reference instructions in direct addressing mode, and memory reference in indirect addressing mode. Register reference instructions reads the opcode directly. Memory reference instructions in direct addressing mode requires three memory read cycles. One to fetch opcode, one to fetch operand address, and last one to get operand and execute the instruction. Memory reference instructions in indirect addressing mode requires four memory read cycles, many are similar to the direct addressing mode. One to get opcode, then read address of operand address, then get the operand address, and last get operand and execute the instruction. Using the DIP switches to point to the memory location, the 7 segment display on the board will display it in hexadecimal format.

Algorithmic Solution

The use of memory instructions and machine codes to implement the computer. Derive the logic equation of each control signal using basic logic gates. Then write a pseudo code that this computer will follow, which will later be implemented using opcodes. It adds the numbers of a Fibonacci sequence. The machine code is stored in the memory.

**Design**

Circuit Design

ALU

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S2 | S1 | S0 | Operation | Description |
| 0 | 0 | 0 | AC + DR | Add |
| 0 | 0 | 1 | AC + DR’+1 | Subtract |
| 0 | 1 | 0 | Ashl AC | Arithmetic shift left AC |
| 0 | 1 | 1 | Ashr AC | Arithmetic shift right AC |
| 1 | 0 | 0 | AC ∧ DR | Logic AND |
| 1 | 0 | 1 | AC ∨ DR | Logic OR |
| 1 | 1 | 0 | DR | DR transfer |
| 1 | 1 | 1 | AC’ | Compliment AC |

Control Signals

|  |  |
| --- | --- |
| **Memory** |  |
| memwrite | T10Y6+T9Y4 |
| **CPU registers** |  |
| AR\_load | T0+T2+T5(IR6)'+T7X2 |
| PC\_load | T8Y5 |
| PC\_inc | T2\*S'+T5(IR6)'+(T11+T12)Y6\*S'(DR[7-0]) |
| DR\_load | T8Y0+T8Y1+T8Y2+T8Y3+T8Y6 |
| DR\_inc | T9Y6 |
| IR\_load | T3 |
| AC\_clear | T5\*X1\*IR0 |
| AC\_load | T5X1(IR1+IR2+IR3) + T9(Y0+Y1+Y2+Y3) |
| AC\_inc | T5\*X1\*IR4 |
| OUTD\_load | T1 |
| **CPU ALU** |  |
| ALU\_sel2 | T5X1\*IR1+T9Y3+T9Y0 |
| ALU\_sel1 | T5X1\*IR2+T5X1\*IR3+T9Y3+T10Y6 |
| ALU\_sel0 | T9Y2+T5X1\*IR3+T5X1\*IR[7-0]+T10Y6 |
| **Bus** |  |
| BusSel2 | T0+T9Y4 |
| BusSel1 | T2+T5+T10Y6+T0 |
| BusSel0 | T8Y5+(T10)Y6+T9Y5 |
| **Control Unit** |  |
| SC\_clear | T5X1+T12Y6+T9(Y0+Y1+Y2+Y3+Y4+Y5) |
| Halt | T5X1\*IR5 |

Analysis of the program:

* Load from address a0 to AC
* Compliment AC
* Store AC to a0
* Increment counter of a0
* Branch to address 20
* Stop
* Load number from the memory location of a1 to AC
* Add number from the memory location of a2 to AC
* Store AC to the memory location of a3
* Load from a1 to AC
* Increment AC
* Store AC to the memory address a1
* Increment AC
* Store AC to the memory address a2
* Increment AC
* Store AC to memory address a3
* Branch to the memory location of 05

Pseudo code:

int x=0, y=1, z=2, counter = -10;

array a = new array[12];

a[x]=0;

b[y]=1;

while (counter != 0) {

a[z] = a[x] + a[y];

x++;

y++;

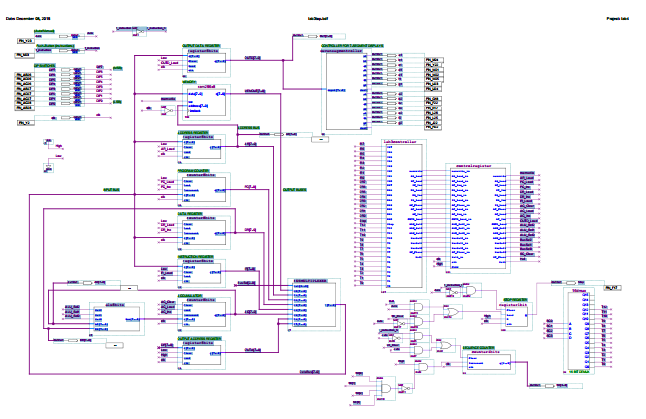
z++;

counter++;

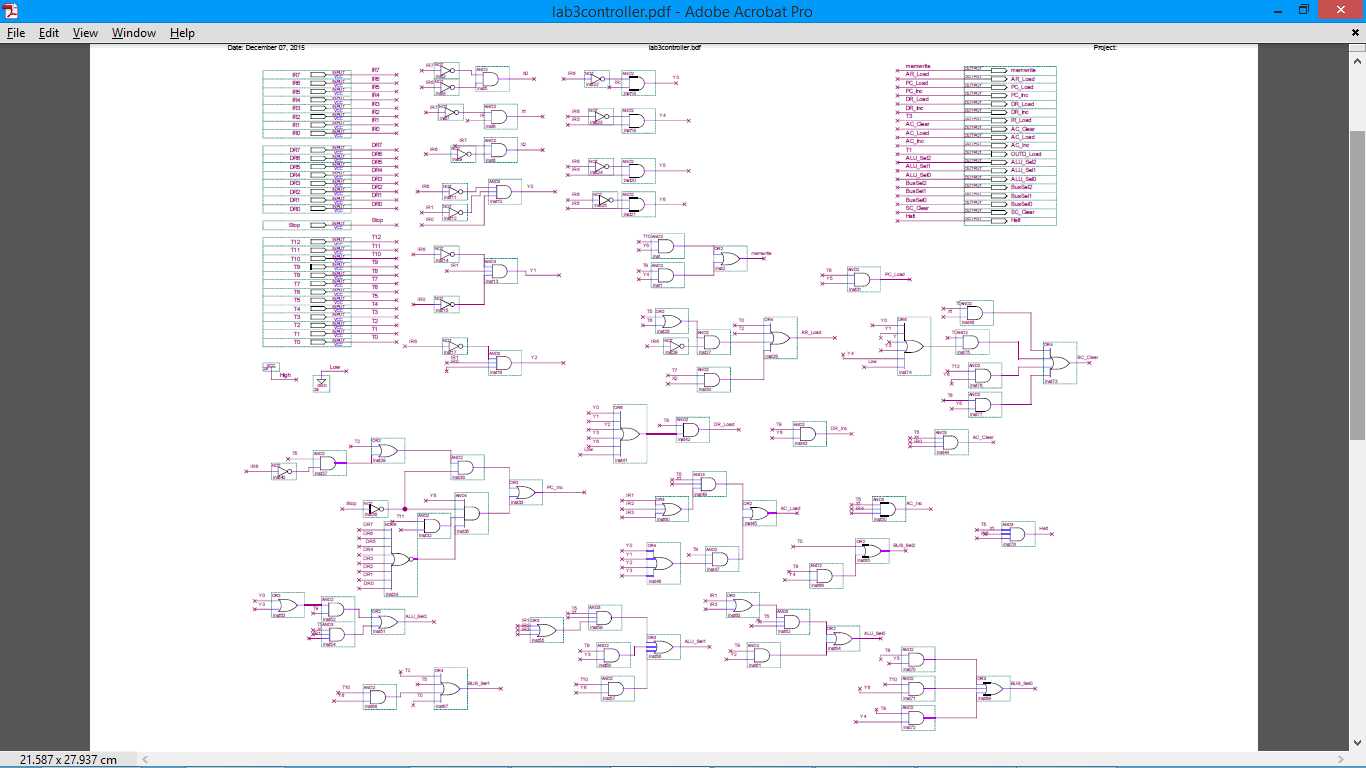
}

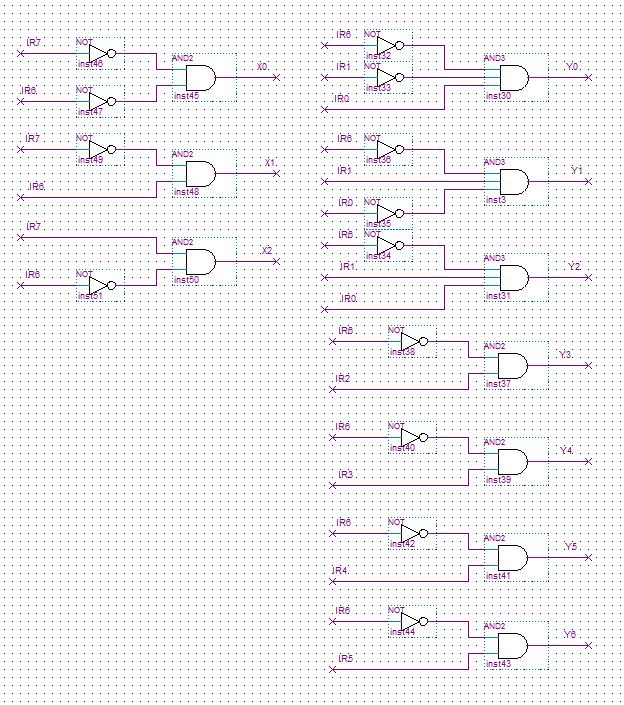
Circuit Diagrams

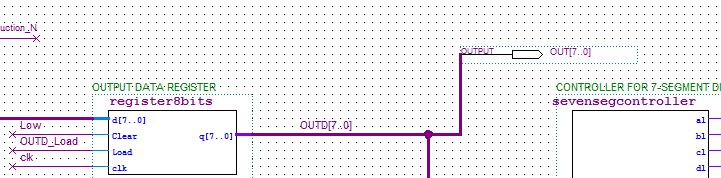
Lab3Top



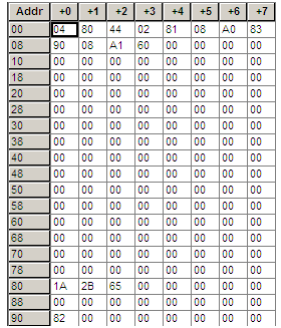
Lab3controller







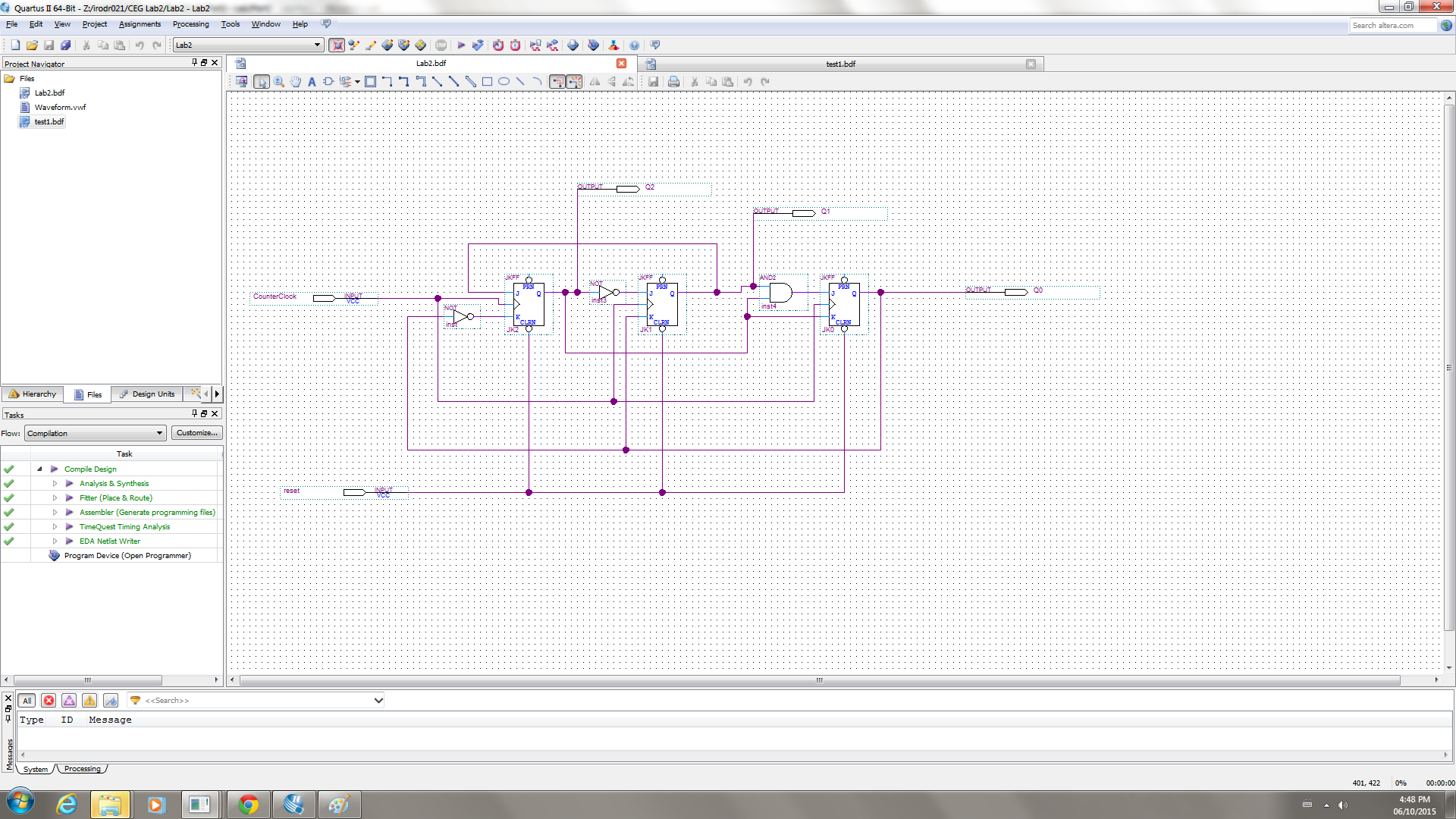
Machine code



Components Used

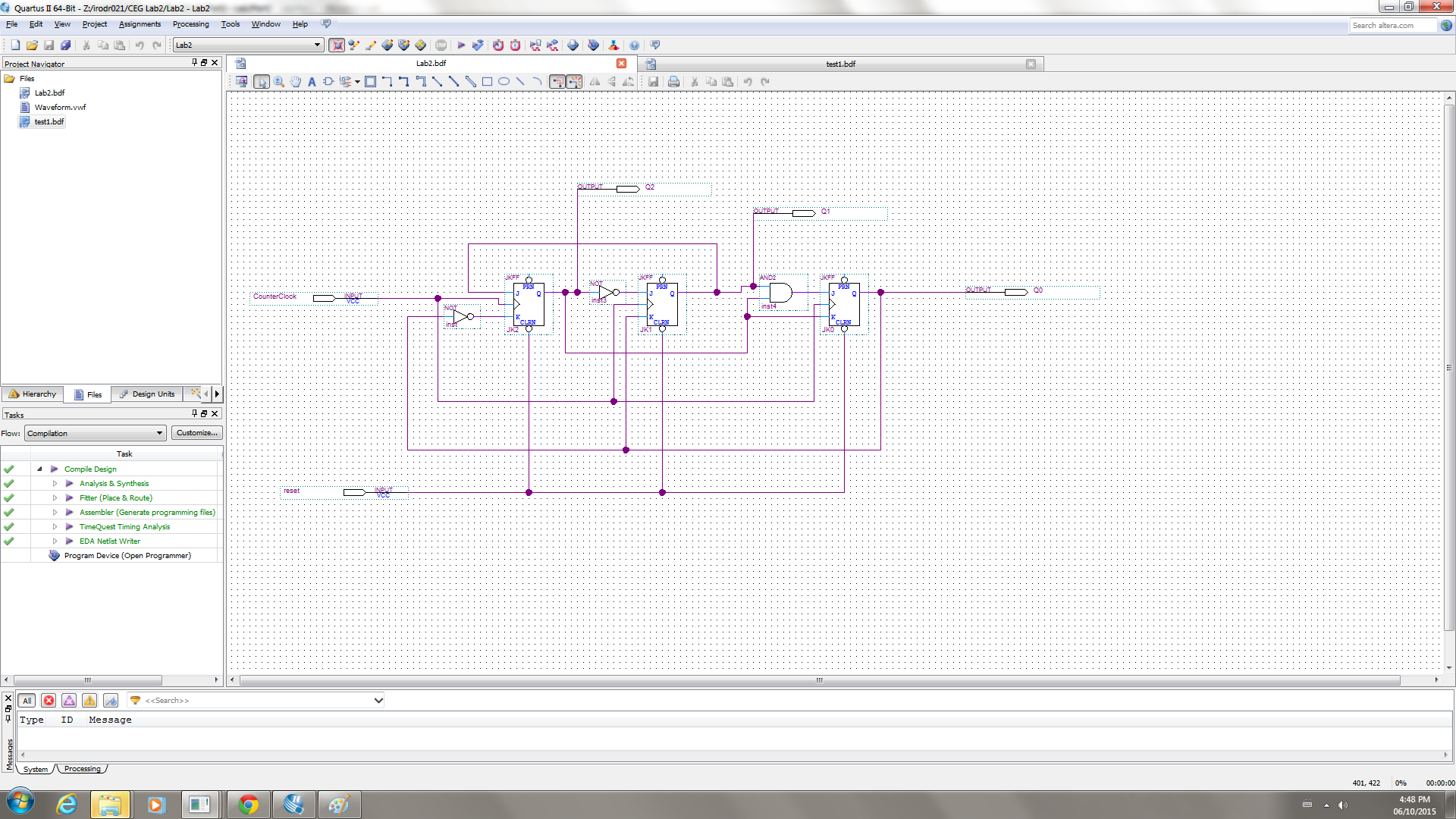
* AND gate: Gives high (1) output when both inputs are high (1).

|  |  |  |
| --- | --- | --- |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



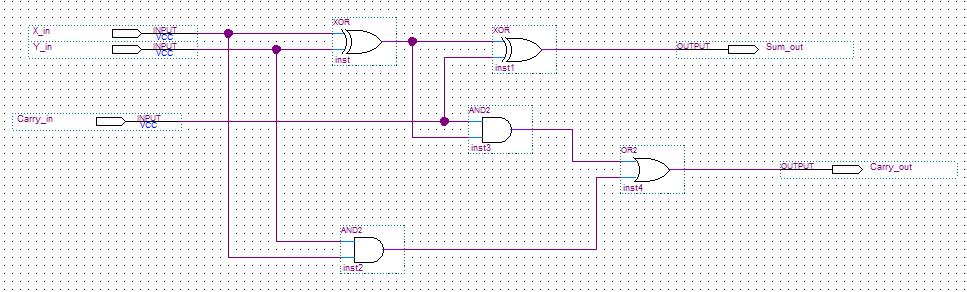
* NOT gate: Inverts the input.

|  |  |
| --- | --- |
| A | Y |
| 0 | 1 |
| 1 | 0 |



* OR gate: Gives high (1) when either input or both are high (1).

|  |  |  |
| --- | --- | --- |
| A | B | Y |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



Discussion of Tool

Quartus II software was used to create and develop the circuits used. The circuits in Quartus II software was then programmed to the Altera DE2-115 board. The Altera DE2-115 board was used for the demonstration as it showed a physical example of the whole lab.

Actual Solution

First, equations were derived to find out the equation for each control signal. Using AND, OR, and NOR gates, the equations were implemented in the lab controller. Finally, a mif file was added which contained the machine code for the programs used.

Challenging Problems

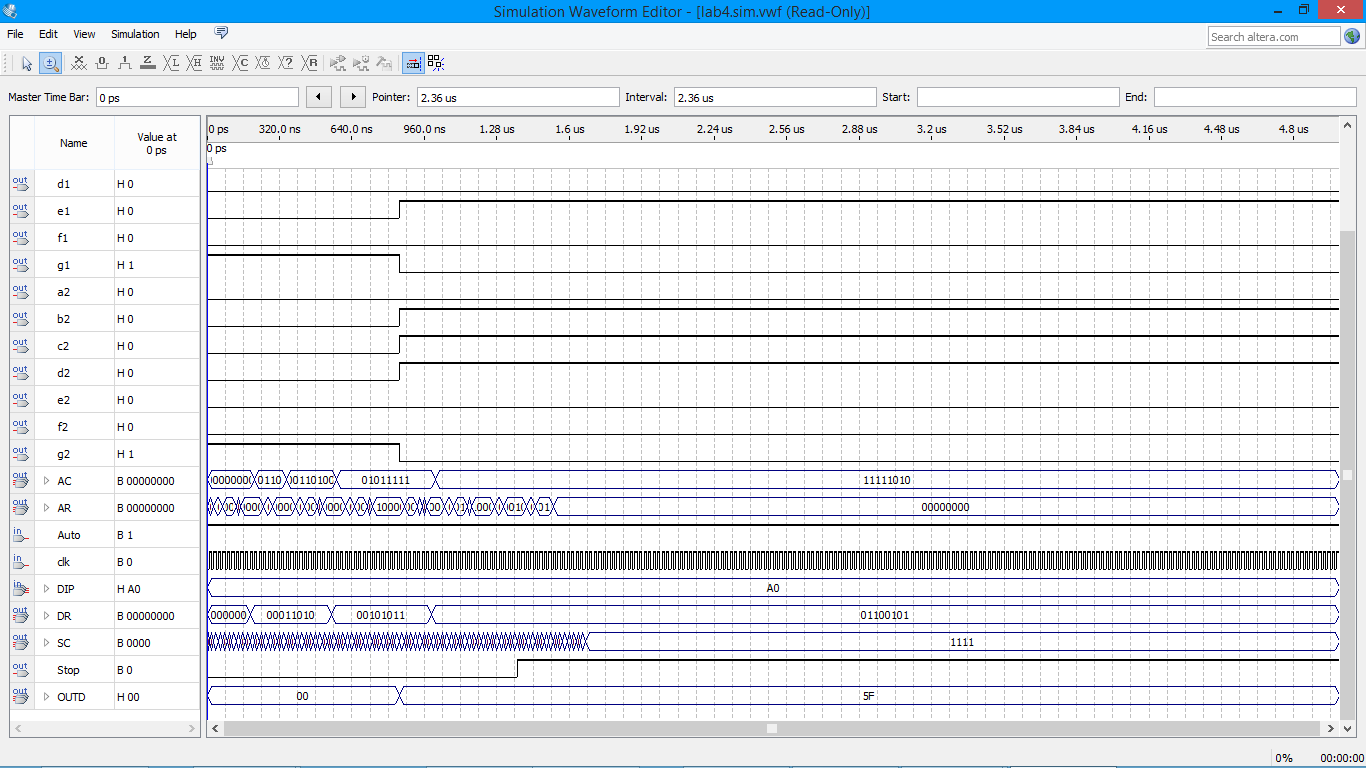
A challenging problem was implementing the lab controller and getting it to function. If there were any mistakes made while deriving the equations or even creating the component into the quartus simulator, the lab controller would not function leading to the whole lab not functioning.

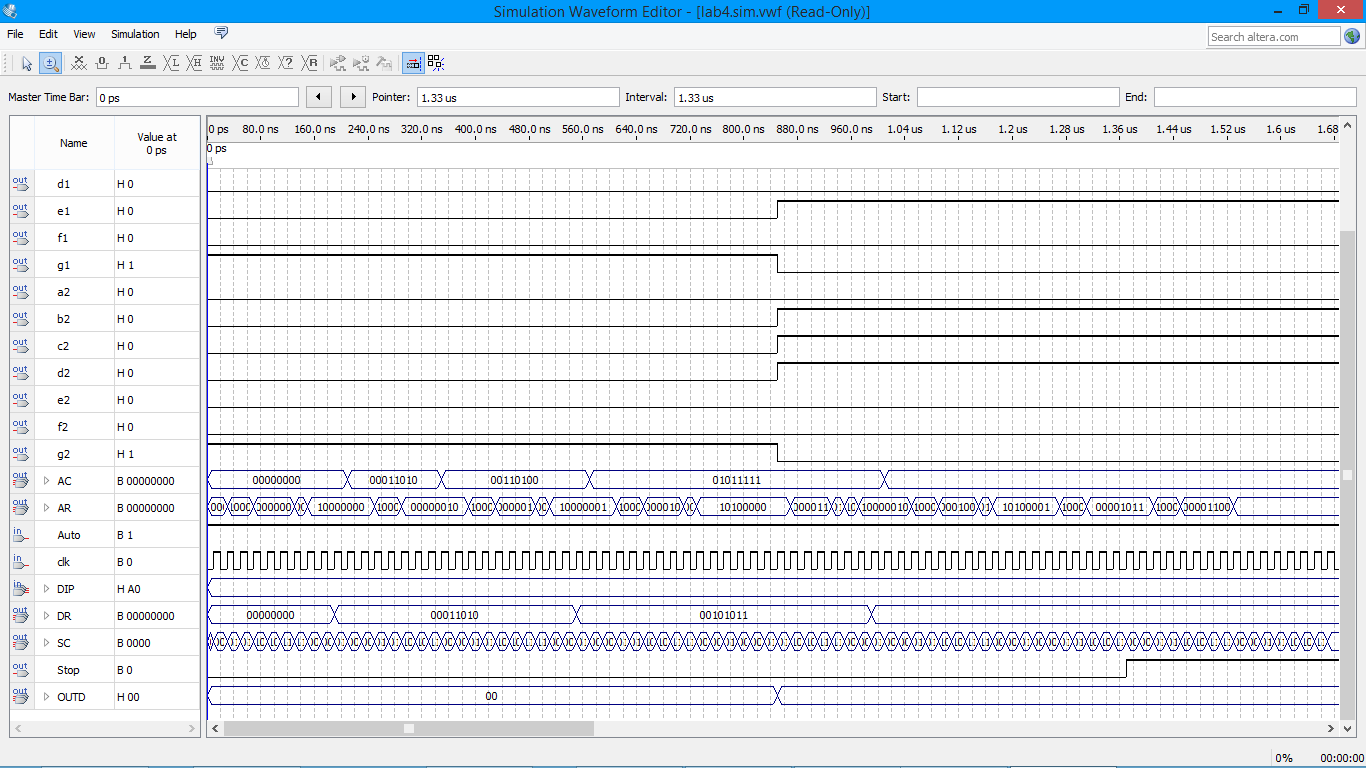
**Real Implementation**

Simulation Results

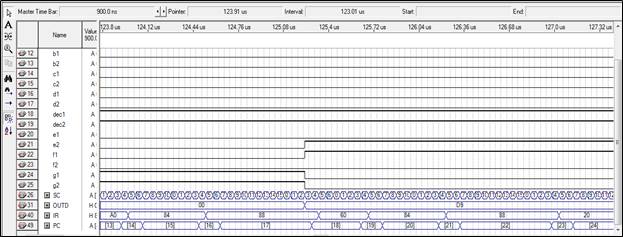
Waveform:

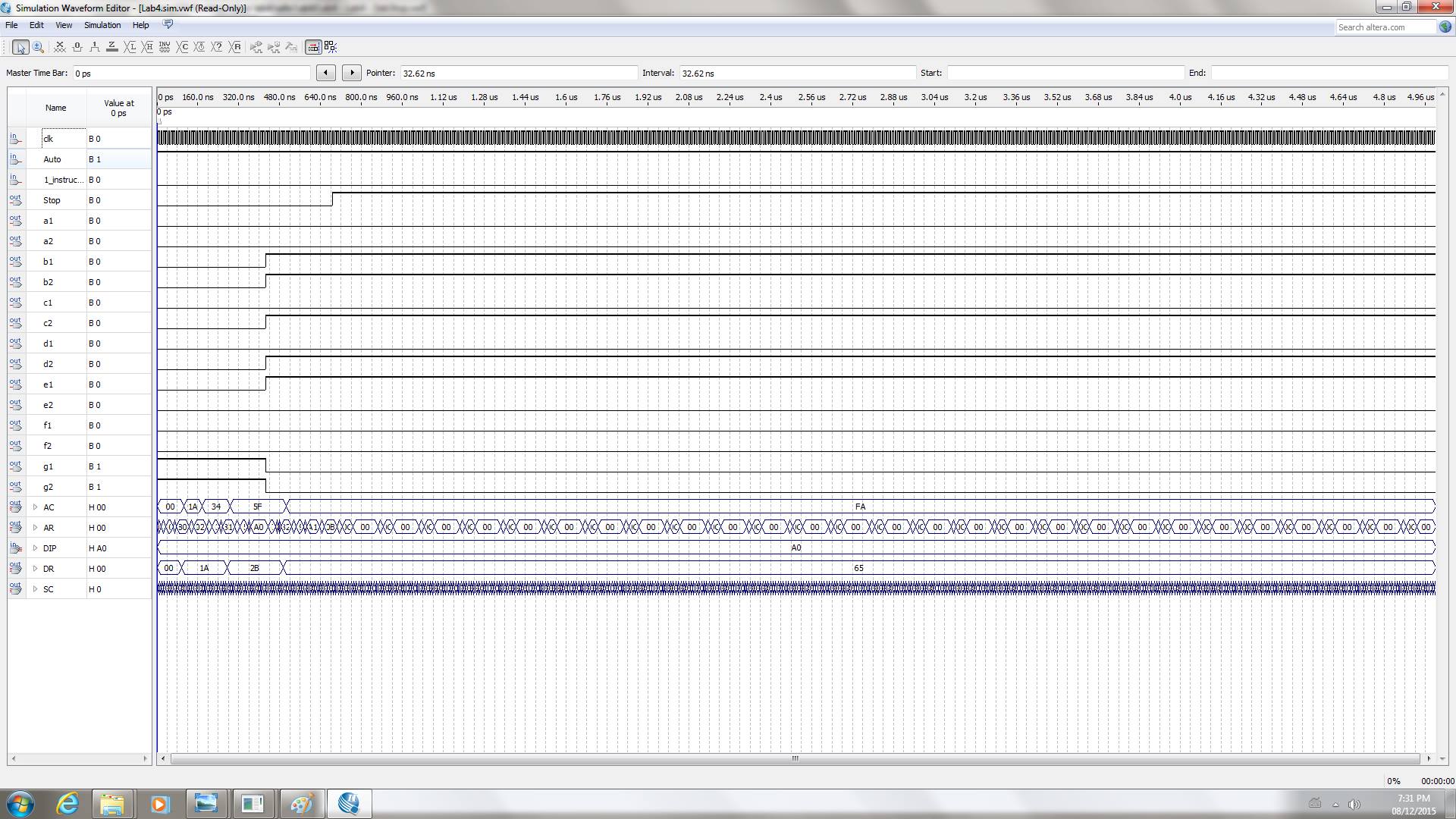
HARDWARE:

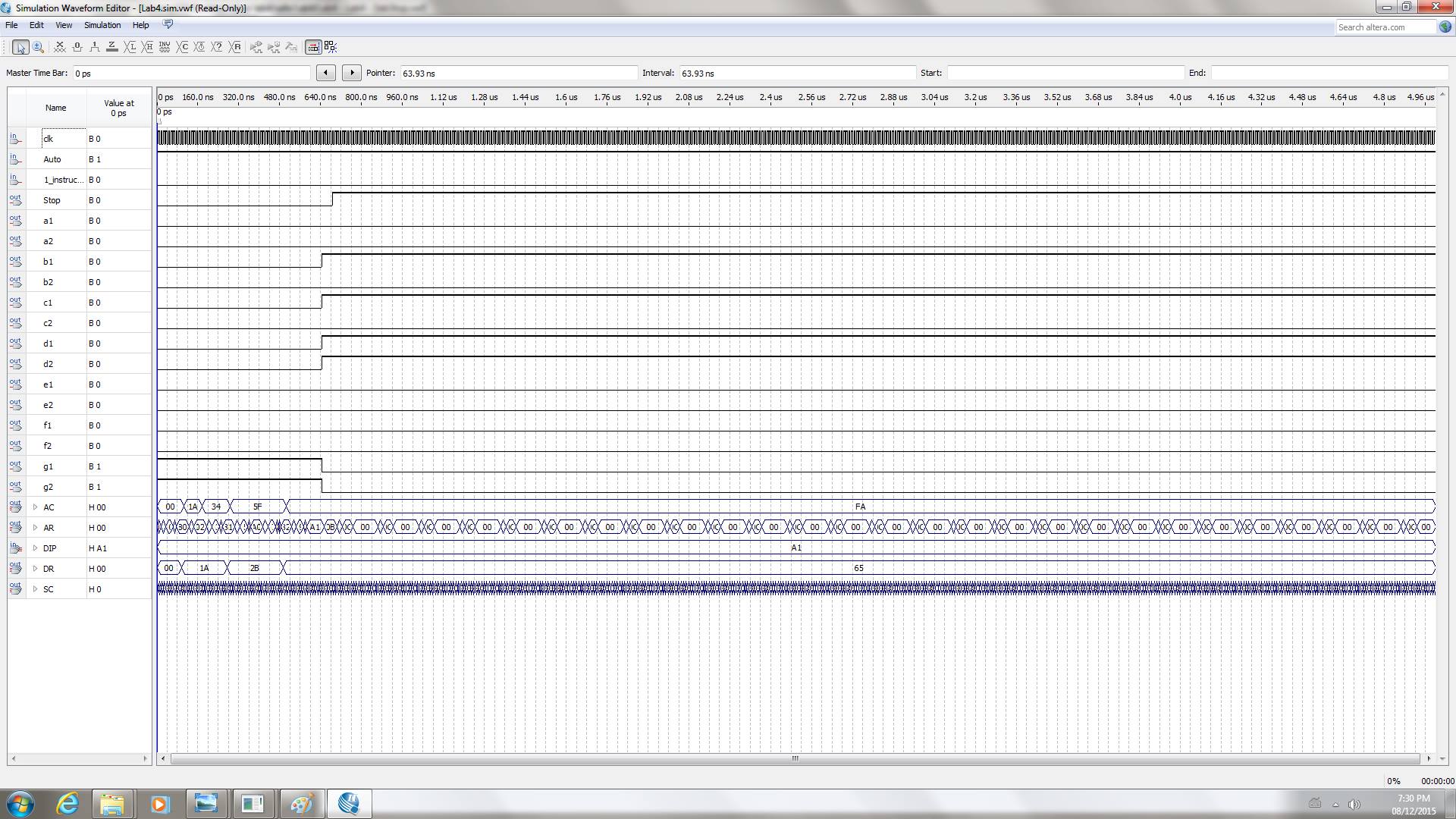




SOFTWARE:







Verification

The real implementations were the same as the theoretical results.

The project was successfully verified in person. The design that was implemented was found to be a working design.

**Discussion**

The objective for this lab was to analyse the structure of a basic computer. The control unit was devised and designed to function with the opcodes to write simple programs in machine code. The lab controller was implemented by deriving equations for each control signal.

After, a given program was analyzed and it was found to be a program that was performing the Fibonacci series. Then a program that constantly added numbers in a given series until the sum was equal to zero was implemented. The program was stopped after D9 was added.

Possible errors were that the lab controller was not implemented correctly. If the lab controller did not work then the programs could not function. It is important that each control signal had the correct equation and that each control signal was implemented correctly. Another error is that the program did not halt at the correct time.