Lab 4: Basic Computer Organization

CEG 2136 B - Computer Architecture

Fall 2017

School of Electrical Engineering and Computer Science University of Ottawa

Course Coordinator: Dr. Sawsan Abdul-majid

Group 18

Felix Singerman 7970742

Nasser Haidar 8543506

Submission Date: December 4, 2017

Theoretical Part

1. Introduction of problem or to lab

In this lab, we are creating and analyzing a basic computer control unit. We will also use opcodes to write simple programs in machine code. This is important because it will help us understand how instructions are executed and how the memory addresses are retrieved.

In the prelab, we derived the equations of all the control signals which have to be generated by the Control Unit to control the registers and ALU for the CPU data path, the bus and the memory. For the hardware aspect of the lab, we analyzed the RTL expressions of Table 2, Table 3, Table 4 in the lab manual and wrote the logic expression for each of the control signals to draw the controller. For the programming component we designed a program to add the sequence given in the lab manual.

2. Discussion of problem

The problem for this lab consisted of designing, simulating, building and testing a control unit.

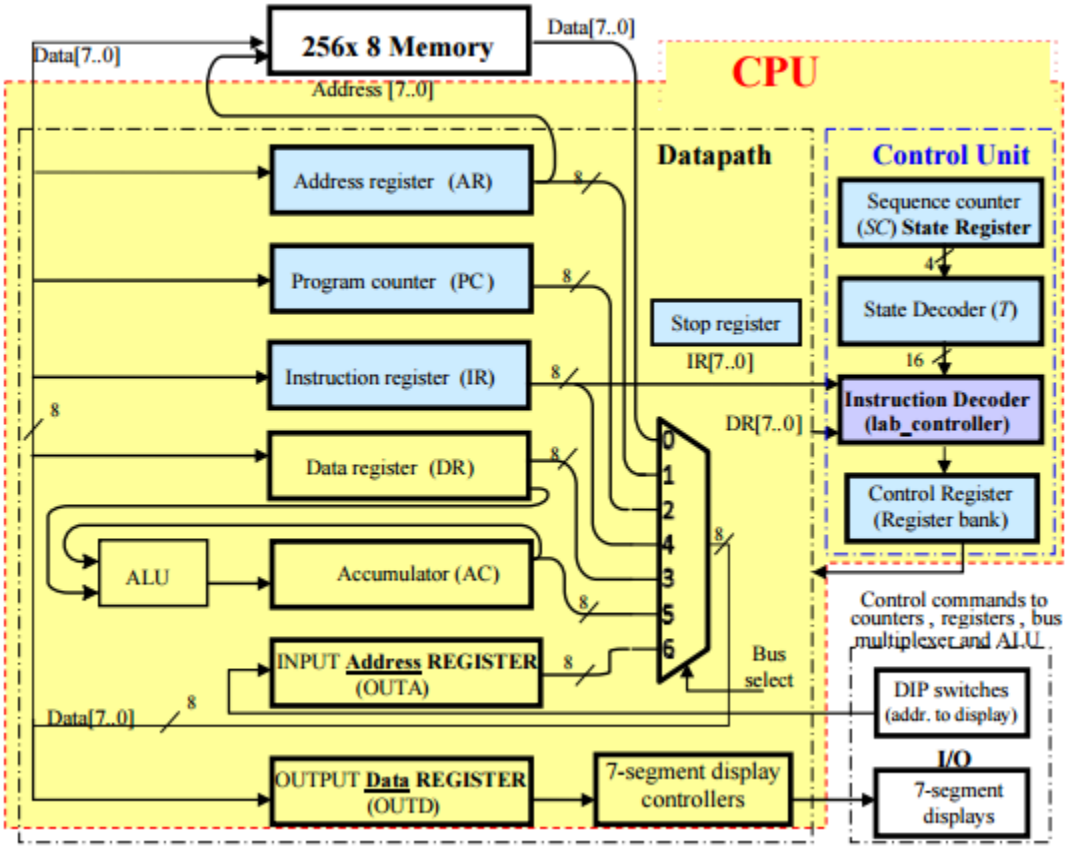


Figure 1: Computer block diagram

Type of Instruction	Symbol	Binary opcodes = hex opcodes Direct Addressing I = IR7 = 0	Indirect Addressing I = IR7 = 1	Description
Memory Reference ($IR_6 = 0$)	AND	00 000001=01	10 000001=81	AND AC to memory word
	ADD	00 000010=02	10 000010=82	Add a memory word to AC
	SUB	00 000011=03	10 000011=83	Subtract a memory word from AC
	LDA	00 000100=04	10 000100=84	Load AC from a memory location
	STA	00 001000=08	10 001000=88	Store AC to a memory location
	BUN	00 010000=10	10 010000=90	Branch unconditionally
	ISZ	00 100000=20	10 100000=A0	Increment content of memory location and skip the following instruction if the incremented number is 0
Register Reference ($IR_6 = 1$)	CLA	01 000001=41		Clear AC
	CMA	01 000010=42		Complement AC
	ASL	01 000100=44		Arithmetic left shift AC
	ASR	01 001000=48		Arithmetic right shift AC
	INC	01 010000=50		Increment AC
	HLT	01 100000=60		Halt. A *Stop* bit is set to 1, which prevents PC from being incremented.

Figure 2: Computer Instructions List

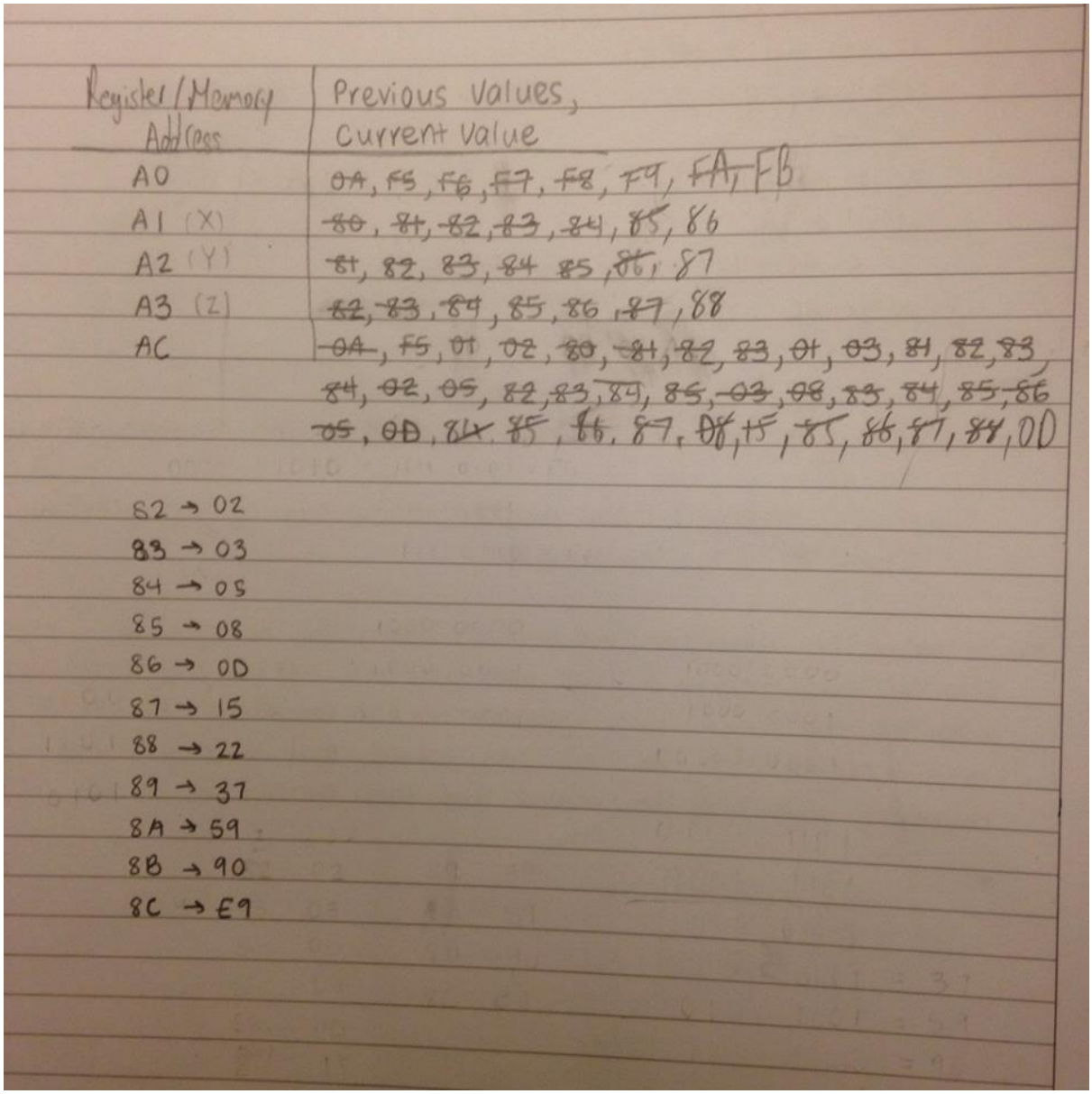
3. Discussion of algorithmic solution Explain the used algorithm to solve it and the block components you are going to use.

To start we needed to find the hardware equations for part 1 using the provided tables from the lab4 document. We then added the logic diagrams for these equations into lab4top file to complete the circuit diagram.

Function	Equation
$memwrite$	$T_9Y_4 + T_{10}Y_6$
AR_LOAD	$T_0 + T_2 + T_5IR'_6 + T_6IR'_6 + T_7X_2$
PC_LOAD	T_8Y_5
PC_INC	$T_2S' + T_5IR'_6S' + DR'_{(7-0)}S'Y_6(T_{11} + T_{12})$
DR_LOAD	$T_8(Y_0 + Y_1 + Y_2 + Y_3 + Y_6)$
DR_INC	T_9Y_6
IR_LOAD	T_3
AC_CLEAR	$T_5X_1IR_0$
AC_LOAD	$T_5X_1(IR_1 + IR_2 + IR_3) + T_9(\sum_{n=0}^{3}(Y_n)$
AC_INC	$T_5X_1IR_4$
$OUTD\_LOAD$	T_1
ALU_SEL2	$T_5X_1IR_1 + T_9Y_3 + T_9Y_0$
ALU_SEL1	$T_9Y_3 + T_5X_1I(\sum_{n=1}^{3}R_n)$
ALU_SEL0	$T_9Y_2 + T_5X_1(IR_3 + IR_1)$
BUS_SEL2	$T_0 + T_9Y_4$
BUS_SEL1	$T_2 + T_5 + T_{10}Y_6 + T_0$
BUS_SEL0	$T_8Y_5 + T_{10}Y_6 + T_9Y_4$
SC_CLEAR	$T_5X_1 + T_9(\sum_{n=0}^{4}Y_n) + T_8Y_5 + T_{12}Y_6$
$Halt$	$T_5X_1IR_5$

Figure 3: Hardware Equations

For the second part of the lab we needed to write a program that consecutively added each number of a given sequence of hexadecimal numbers and displays the hexadecimal number that causes the sum to be equal to zero.



Register / Memory Address	Previous values, Current value
A0	0A, F5, F6, F7, F8, F9, FA, FB
A1 (X)	80, 81, 82, 83, 84, 85, 86
A2 (Y)	81, 82, 83, 84 85, 86, 87
A3 (Z)	82, 83, 84, 85, 86, 87, 88
AC	0A, F5, 01, 02, 80, 81, 82, 83, 01, 03, 81, 82, 83, 84, 02, 05, 82, 83, 84, 85, 03, 08, 83, 84, 85, 86 05, 0D, 81, 85, 86, 87, 0F, 15, 85, 86, 87, 88, 0D

82 → 02
83 → 03
84 → 05
85 → 08
86 → 0D
87 → 15
88 → 22
89 → 37
8A → 59
8B → 90
8C → E9

Design Part

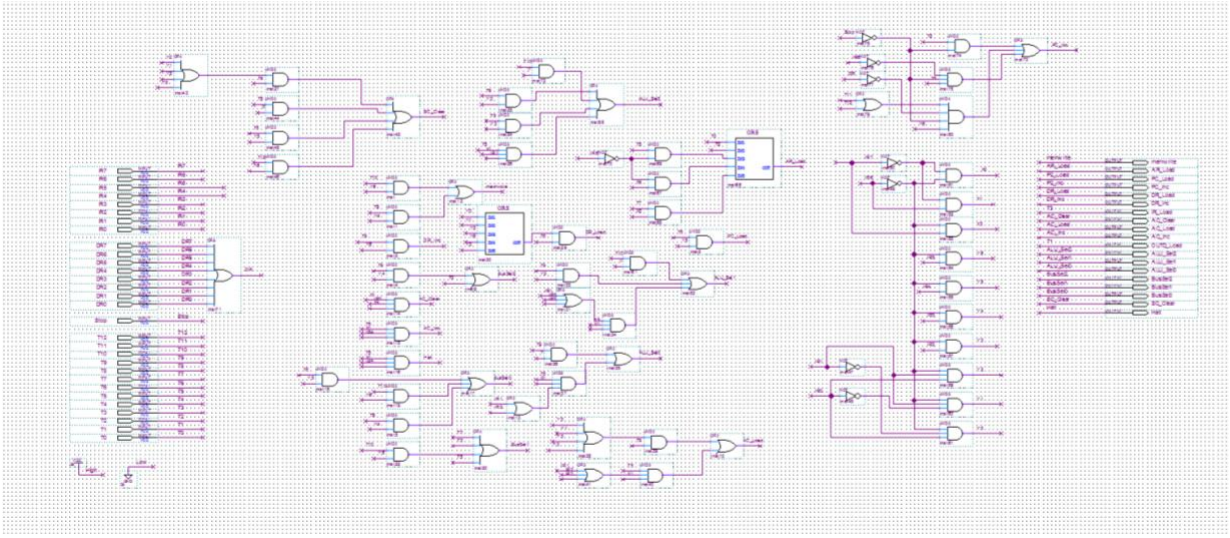


Figure 6: Completed Circuit Diagram lab4top

2. **Discussion of used components**

AND Gate
A gate where the output is 1 if and only if both inputs have a value of 1, otherwise, its output is 0.

NOT Gate
A gate where the output is 1 if the input is 0, and vice versa.

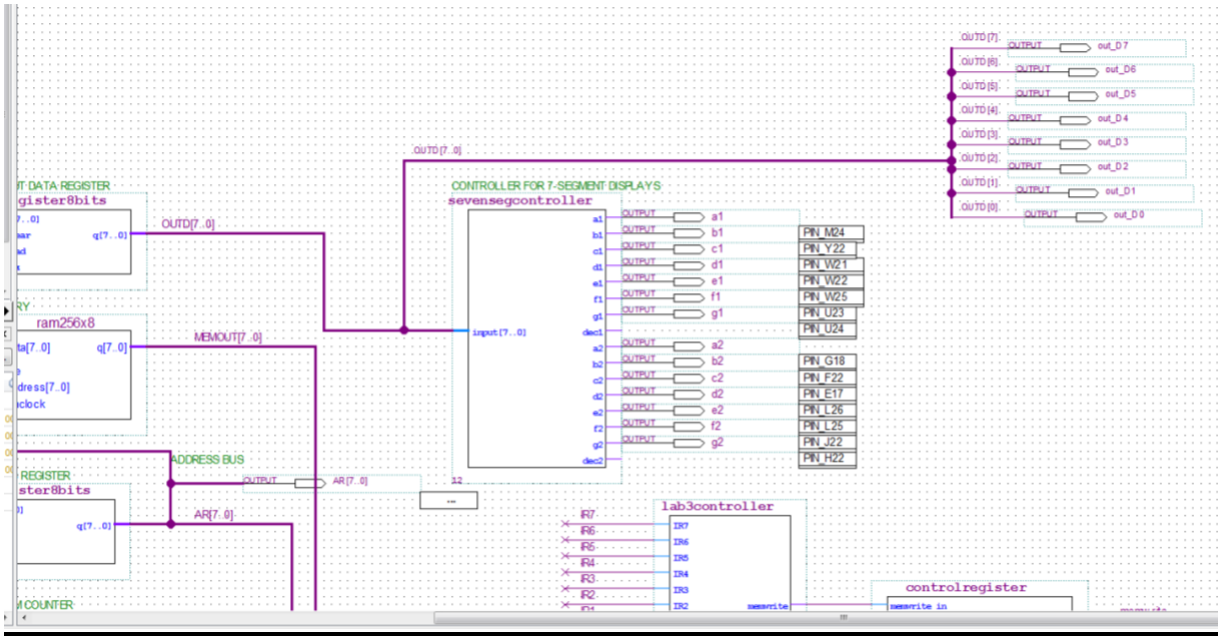
XOR Gate
A gate where the output is 1 if only one of the inputs is 1.

NOR Gate
A gate where the output is 1 if and only if both inputs are 0, otherwise, its output is 0.

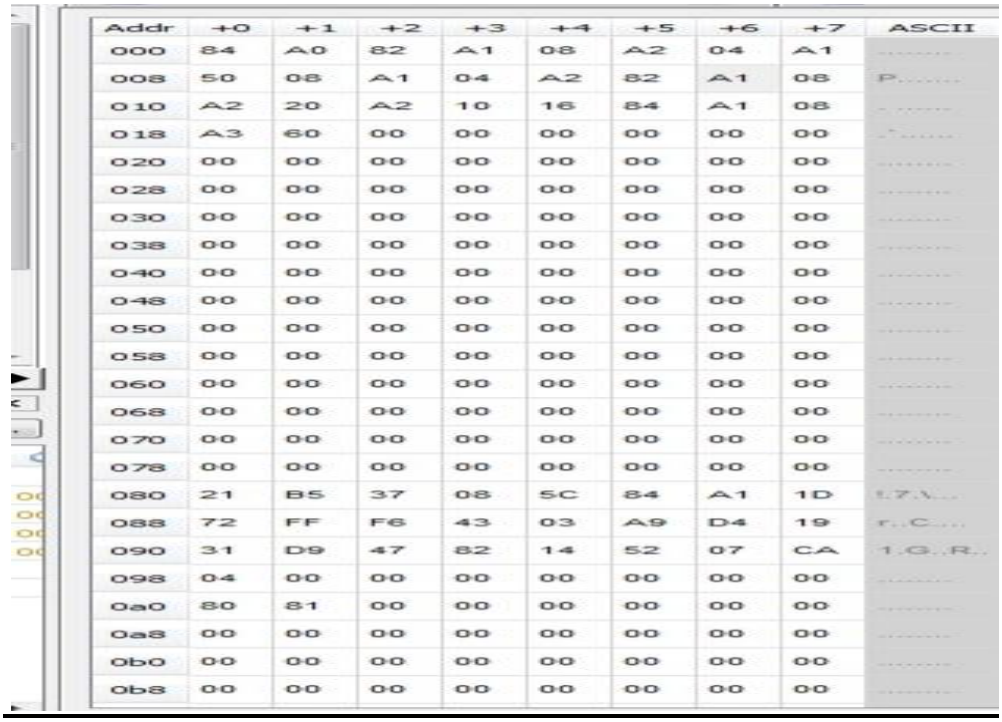
Circuit Diagrams
Refer to “Design Part”

3. Discussion of actual solution

After deriving the equations for the control signals we implemented them in Quartus.



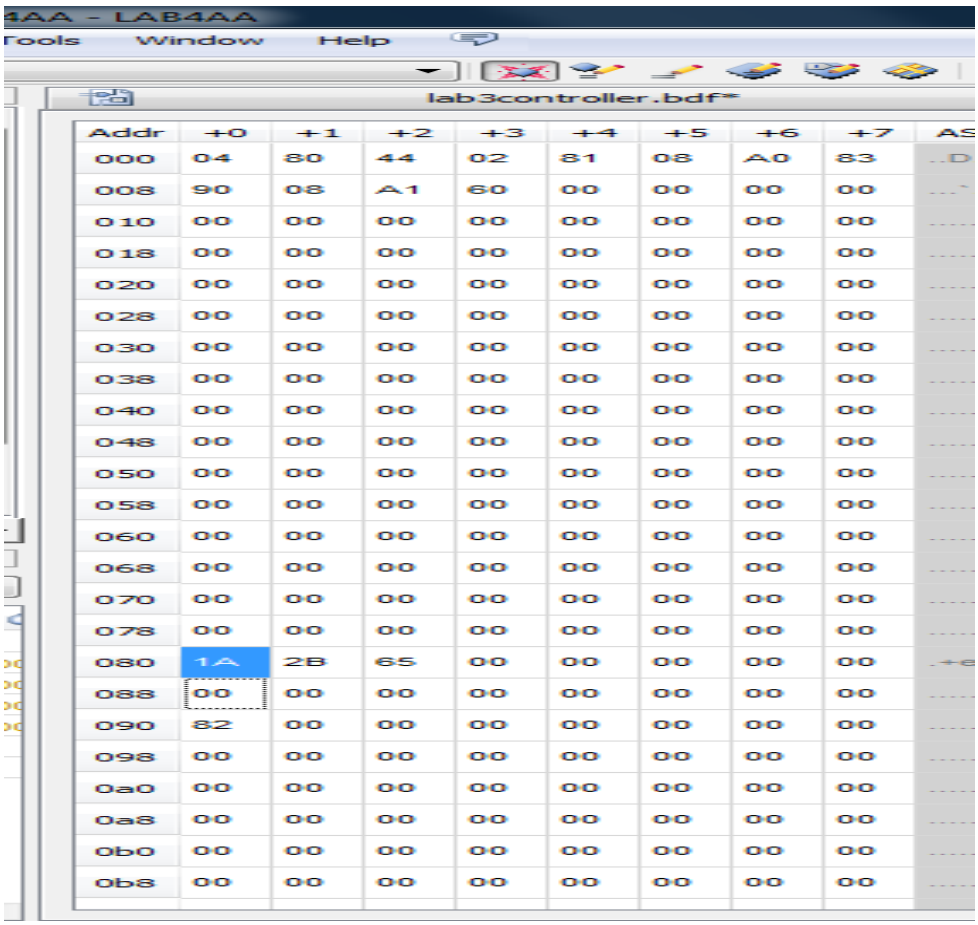
We then created a mif file for the second part of the lab.



Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
000	84	A0	82	A1	08	A2	04	A1	.....
008	50	08	A1	04	A2	82	A1	08	P.....
010	A2	20	A2	10	16	84	A1	08	. .....
018	A3	60	00	00	00	00	00	00	.`.....
020	00	00	00	00	00	00	00	00	.....
028	00	00	00	00	00	00	00	00	.....
030	00	00	00	00	00	00	00	00	.....
038	00	00	00	00	00	00	00	00	.....
040	00	00	00	00	00	00	00	00	.....
048	00	00	00	00	00	00	00	00	.....
050	00	00	00	00	00	00	00	00	.....
058	00	00	00	00	00	00	00	00	.....
060	00	00	00	00	00	00	00	00	.....
068	00	00	00	00	00	00	00	00	.....
070	00	00	00	00	00	00	00	00	.....
078	00	00	00	00	00	00	00	00	.....
080	21	B5	37	08	5C	84	A1	1D	!.7.\...
088	72	FF	F6	43	03	A9	D4	19	r..C....
090	31	D9	47	82	14	52	07	CA	1.G..R..
098	04	00	00	00	00	00	00	00	.....
0a0	80	81	00	00	00	00	00	00	.....
0a8	00	00	00	00	00	00	00	00	.....
0b0	00	00	00	00	00	00	00	00	.....
0b8	00	00	00	00	00	00	00	00	.....

Figure 4: .mif file for addition program

Figure 5: Simulation for the addition program



Addr	+0	+1	+2	+3	+4	+5	+6	+7	AS
000	04	80	44	02	81	08	A0	83	..D
008	90	08	A1	60	00	00	00	00	....`
010	00	00	00	00	00	00	00	00	.....
018	00	00	00	00	00	00	00	00	.....
020	00	00	00	00	00	00	00	00	.....
028	00	00	00	00	00	00	00	00	.....
030	00	00	00	00	00	00	00	00	.....
038	00	00	00	00	00	00	00	00	.....
040	00	00	00	00	00	00	00	00	.....
048	00	00	00	00	00	00	00	00	.....
050	00	00	00	00	00	00	00	00	.....
058	00	00	00	00	00	00	00	00	.....
060	00	00	00	00	00	00	00	00	.....
068	00	00	00	00	00	00	00	00	.....
070	00	00	00	00	00	00	00	00	.....
078	00	00	00	00	00	00	00	00	.....
080	1A	2B	65	00	00	00	00	00	..+e
088	00	00	00	00	00	00	00	00	.....
090	82	00	00	00	00	00	00	00	.....
098	00	00	00	00	00	00	00	00	.....
0a0	00	00	00	00	00	00	00	00	.....
0a8	00	00	00	00	00	00	00	00	.....
0b0	00	00	00	00	00	00	00	00	.....
0b8	00	00	00	00	00	00	00	00	.....

Figure 6: .mif file for consecutive numbers added

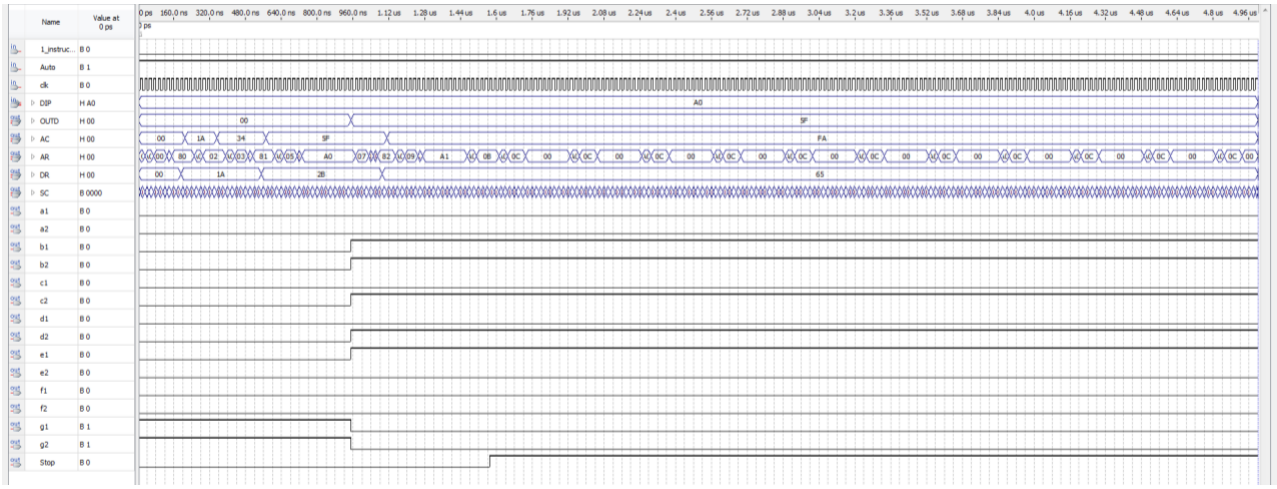


Figure 7: Simulation for the consecutive numbers added program

4. Discussion of tool

Altera DE2-115 Board
This is a circuit board which consists of multiple pins, buttons, and LEDs. It allows us to visualize and test our designed circuits from the Quartus software.

5. Discussion of challenging problems

We did not come across many issues during this lab. Apart from a few issues in equation calculation, the lab went off without a hitch. We just had to go over our equations with the TA to see where we went wrong.

By conducting this lab, we learned the design of controllers for a basic computer and how machine code is used within the basic computer to get desired results. Although we encountered some issues, overall we were successful with our implementation of this lab