EEL 3701C: Digital Logic & Computer Systems
Revision **0**

Lab #7 Report: **Assembly Programming**

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
April 17, 2025

REQUIREMENTS NOT MET

N/A

VIDEO FILE LINK

https://youtu.be/D0PMlahxjUo?si=xKvcWjgoS-V64An1

PROBLEMS ENCOUNTERED

N/A

FUTURE WORK/APPLICATIONS

In this lab, we gained hands-on experience on how a simple CPU executes instructions and how the addresses correspond to assembly code. We learned how to handle machine instructions into a .mif file as well as test G-IDE to simulate out code. This simple lab felt really complex, so understanding this lab is a good foundation for applications that are even more complex since most computers now have a lot more automated functions than this lab does.

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
April 17, 2025

PRE-LAB QUESTIONS OR EXERCISES

Part 1: Simulating Existing Code

1.2) Briefly discuss the purpose of this (eprom.mif) program.

The eprom.mif file stores the insctuctions for the GPU to follow.

EEL 3701C: Digital Logic & Computer Systems
Revision **0**
Lab #7 Report: **Assembly Programming**

Poche, Natalie
Class #: 11198
PI Name: Jaiden Magnan
April 17, 2025

PRE-LAB REQUIREMENTS (Design, Schematic, ASM Chart, VHDL, etc.)

Each section of the pre-lab requirements should be completed separately, and in order. Include each of the following items in order. Note that some of these items will not apply to every lab. Anything scanned or copied ***must be clear and legible***.

- Logic equations. (Note that logic equations do not contain activation levels.)
- **All** tables and figures should have captions with Figure/Table numbers and a description of for which part of the lab it references, e.g., ***Table 3****: Truth Table for Part B.*
- Truth tables and voltage tables (and/or next-state truth tables).
- When applicable, include Karnaugh Maps (i.e., K-Maps).
- Include hand-drawn circuits (when required). Label all input and output activation-levels and intermediate equations in the circuits.
- Include screenshots of the BDF designs of circuits.
  - Label all input and output activation-levels, i.e., use _L suffix for active-low signals and no suffix for active-high signals. **Add chip and pin numbers to any schematic that will be constructed.**
  - Images should be large enough so that inputs, outputs, labels, and parts are clearly visible and distinguishable to any reader.
  - Each BDF should have the following info on the top left corner (similar to the top right of this page):

    Last Name, First Name
    Lab #, Part #
    Class #
    PI Name:
    Description: (short description of what is to be accomplished in the design; perhaps an equation)

  - In Windows, I use the ***Snipping Tool***, which is now built into Windows. Just type “snip” in the Windows search box and then select ***Snipping Tool***.
- When necessary, include ASM Charts. These can be hand-drawn, but clear and legible. We recommend that you use resources like https://www.draw.io/ to create computer-generated ASMs.
- Truth tables or next-state truth tables should have the following characteristics.
  - Can be either typed or hand-written and scanned (must be clear and legible)
  - Must be in **counting order** (i.e., inputs of 000, 001, 010, 011, …, 111)
  - Clearly distinguish inputs from outputs (see the example below that uses a thick line)
  - If you are designing a state machine or a controller, clearly indicate and separate signal values both before the clock and after the clock (i.e., Q1 and $Q1^+$, respectively)
  - Tip: divide rows into consecutive groups of 4 (or 2 or 8) to make it easier for both you and your PI to read.
  - **Example**

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table: Caption for above table. It should reference the part of the lab, e.g., “Table 3: Truth Table for Part B.”

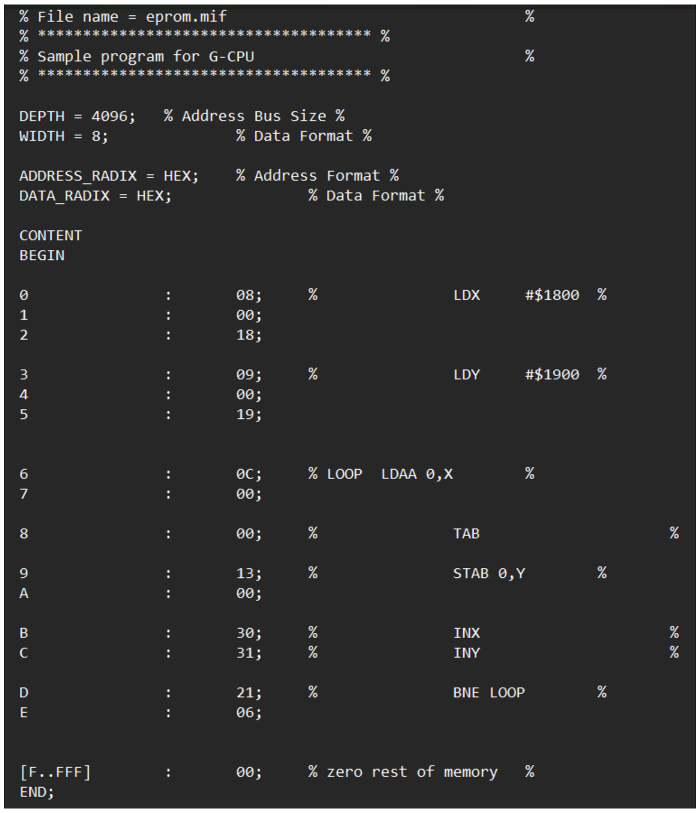
- Voltage tables should have the following characteristics.
  - Must be in counting order (i.e., inputs of LLL, LLH, LHL, … , HHH)
  - Use similar formatting to truth tables (described above)

- **Example**

A(H)	B(H)	C(L)	Y(L)
L	L	L	H
L	L	H	H
L	H	L	L
L	H	H	H
H	L	L	H
H	L	H	H
H	H	L	H
H	H	H	L

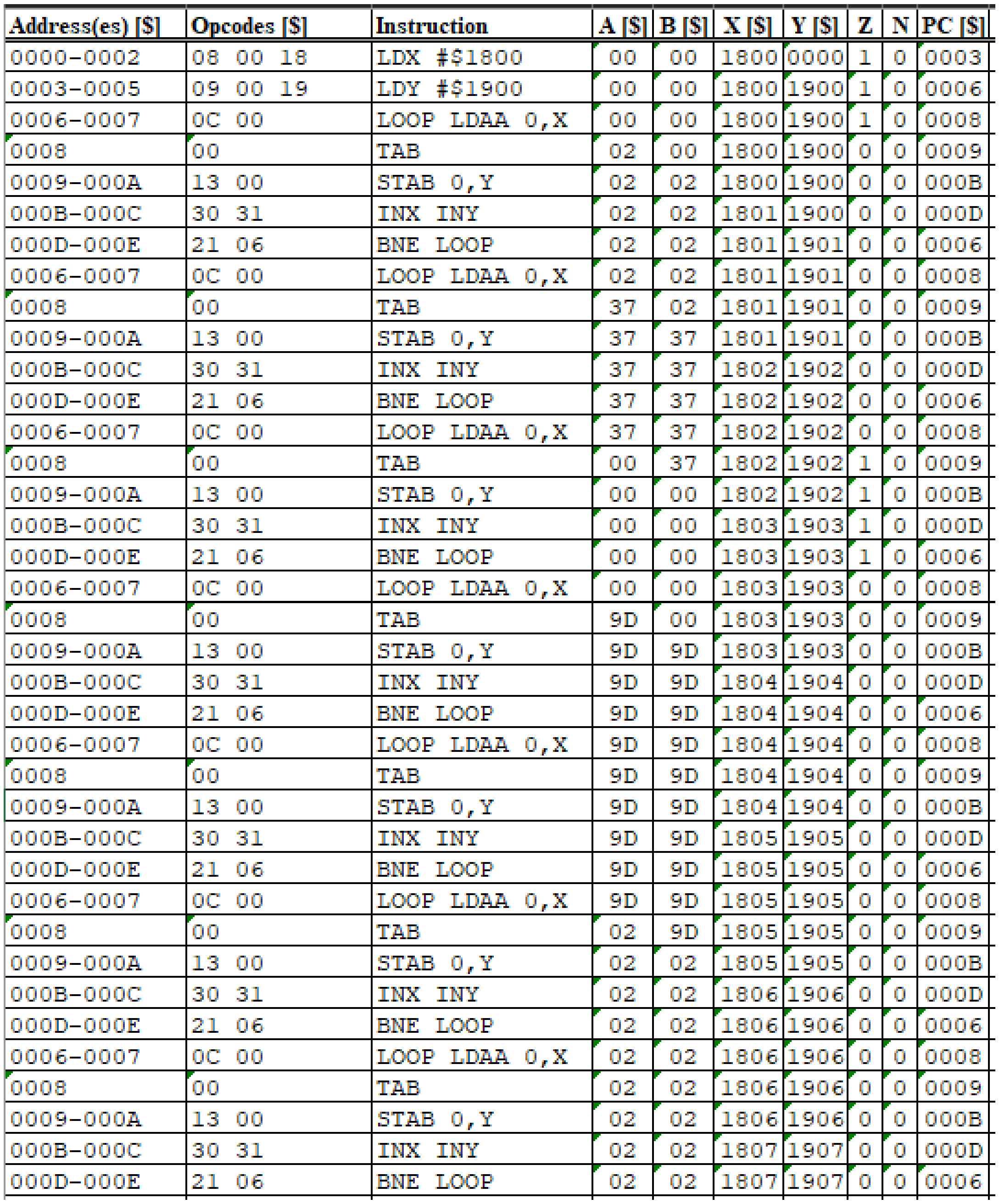
- Include **meaningfully annotated** functional simulations.
  - Using the grouping tool, ***group as many signals together as possible*** (when it makes sense to group them)! If you are simulating a basic logic equation, group all the inputs together. If you are simulating a circuit that includes MSI elements, group signals of the form X_{N-0}. The most-significant bit should appear first, ending with the least-significant bit. If you are simulating an ALU, group the buses together as just described.
  - Not every row of your voltage table must be annotated in the waveform simulation, but your choices of rows that you annotate must be ***encompassing***
  - If you are designing a state machine or a controller, your CLK signal should appear ***at the top*** of your inputs and outputs. The general order is CLK −> Reset −> state bits −> inputs −> outputs.
  - ***Tip***: Use Microsoft Paint to annotate your waveforms. An alternative is to print out the waveforms, annotate them by hand, then scan and upload
  - ***Hint***: Consider a truth table where the output is true in significantly less cases than it is false (or vice versa). If the output signal is active-high, it would be wise to annotate only the cases where the output voltage is HIGH.
- Include every line of VHDL programs, including both ***architecture*** and ***behavior*** sections.
- Include every line of any **MIF** files. If these are associated with assembly language programs, you can either put the assembly code as comments or separately include assembly language programs

Part1: Simulation Existing Code



```
% File name = eprom.mif                                 %
% ************************************* %
% Sample program for G-CPU                              %
% ************************************* %

DEPTH = 4096;   % Address Bus Size %
WIDTH = 8;              % Data Format %

ADDRESS_RADIX = HEX;    % Address Format %
DATA_RADIX = HEX;               % Data Format %

CONTENT
BEGIN

0               :       08;     %               LDX     #$1800  %
1               :       00;
2               :       18;

3               :       09;     %               LDY     #$1900  %
4               :       00;
5               :       19;


6               :       0C;     % LOOP  LDAA 0,X        %
7               :       00;

8               :       00;     %               TAB                     %

9               :       13;     %               STAB 0,Y        %
A               :       00;

B               :       30;     %               INX                     %
C               :       31;     %               INY                     %

D               :       21;     %               BNE LOOP        %
E               :       06;


[F..FFF]        :       00;     % zero rest of memory   %
END;
```

Figure 1: Lab 7 Part 1.1 - Code from the eprom.mif file



Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]
0000-0002	08 00 18	LDX #$1800	00	00	1800	0000	1	0	0003
0003-0005	09 00 19	LDY #$1900	00	00	1800	1900	1	0	0006
0006-0007	0C 00	LOOP LDAA 0,X	00	00	1800	1900	1	0	0008
0008	00	TAB	02	00	1800	1900	0	0	0009
0009-000A	13 00	STAB 0,Y	02	02	1800	1900	0	0	000B
000B-000C	30 31	INX INY	02	02	1801	1900	0	0	000D
000D-000E	21 06	BNE LOOP	02	02	1801	1901	0	0	0006
0006-0007	0C 00	LOOP LDAA 0,X	02	02	1801	1901	0	0	0008
0008	00	TAB	37	02	1801	1901	0	0	0009
0009-000A	13 00	STAB 0,Y	37	37	1801	1901	0	0	000B
000B-000C	30 31	INX INY	37	37	1802	1902	0	0	000D
000D-000E	21 06	BNE LOOP	37	37	1802	1902	0	0	0006
0006-0007	0C 00	LOOP LDAA 0,X	37	37	1802	1902	0	0	0008
0008	00	TAB	00	37	1802	1902	1	0	0009
0009-000A	13 00	STAB 0,Y	00	00	1802	1902	1	0	000B
000B-000C	30 31	INX INY	00	00	1803	1903	1	0	000D
000D-000E	21 06	BNE LOOP	00	00	1803	1903	1	0	0006
0006-0007	0C 00	LOOP LDAA 0,X	00	00	1803	1903	0	0	0008
0008	00	TAB	9D	00	1803	1903	0	0	0009
0009-000A	13 00	STAB 0,Y	9D	9D	1803	1903	0	0	000B
000B-000C	30 31	INX INY	9D	9D	1804	1904	0	0	000D
000D-000E	21 06	BNE LOOP	9D	9D	1804	1904	0	0	0006
0006-0007	0C 00	LOOP LDAA 0,X	9D	9D	1804	1904	0	0	0008
0008	00	TAB	9D	9D	1804	1904	0	0	0009
0009-000A	13 00	STAB 0,Y	9D	9D	1804	1904	0	0	000B
000B-000C	30 31	INX INY	9D	9D	1805	1905	0	0	000D
000D-000E	21 06	BNE LOOP	9D	9D	1805	1905	0	0	0006
0006-0007	0C 00	LOOP LDAA 0,X	9D	9D	1805	1905	0	0	0008
0008	00	TAB	02	9D	1805	1905	0	0	0009
0009-000A	13 00	STAB 0,Y	02	02	1805	1905	0	0	000B
000B-000C	30 31	INX INY	02	02	1806	1906	0	0	000D
000D-000E	21 06	BNE LOOP	02	02	1806	1906	0	0	0006
0006-0007	0C 00	LOOP LDAA 0,X	02	02	1806	1906	0	0	0008
0008	00	TAB	02	02	1806	1906	0	0	0009
0009-000A	13 00	STAB 0,Y	02	02	1806	1906	0	0	000B
000B-000C	30 31	INX INY	02	02	1807	1907	0	0	000D
000D-000E	21 06	BNE LOOP	02	02	1807	1907	0	0	0006

Figure 2: Lab 7 Part 1 - Register Table

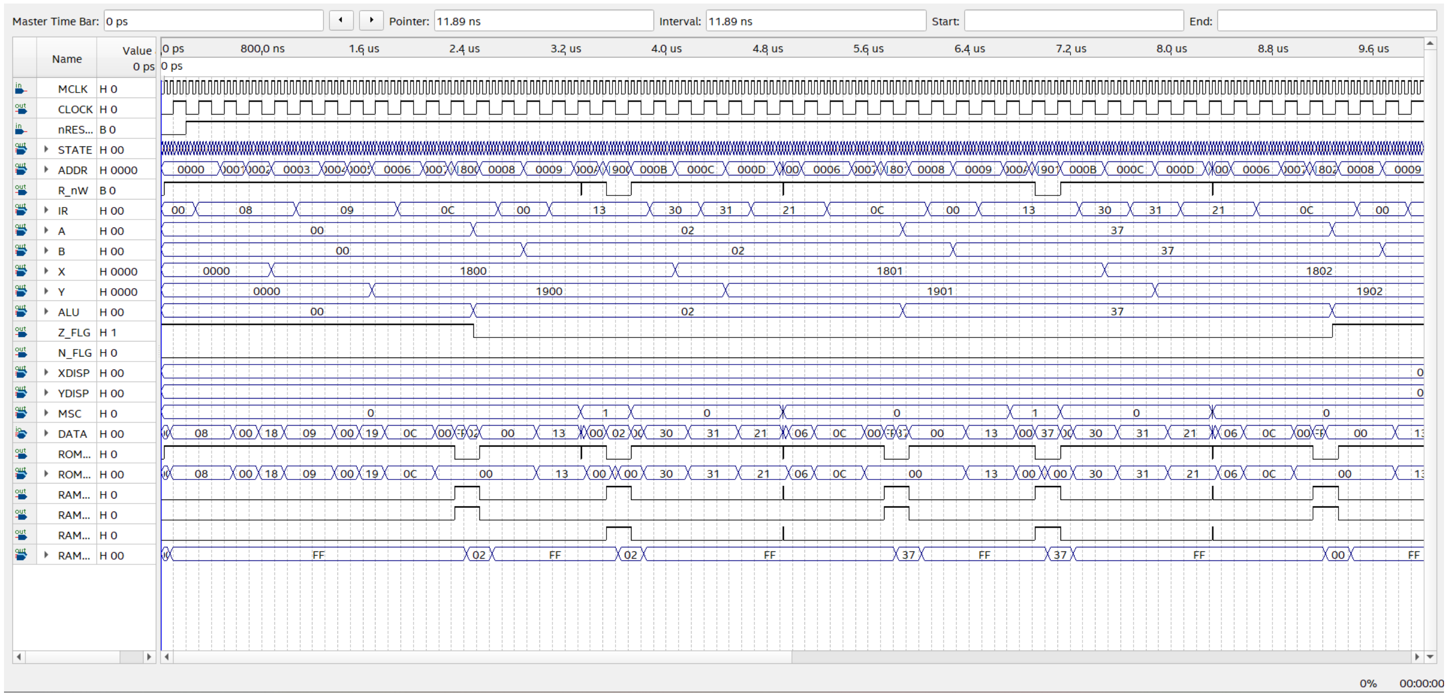
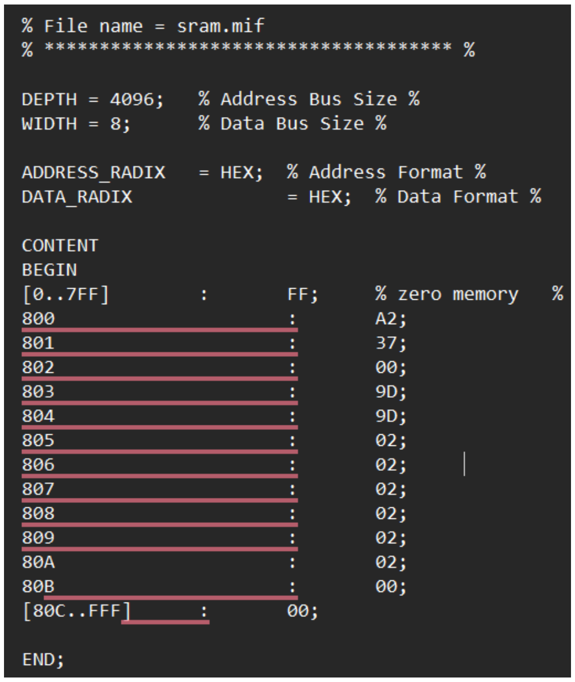


Figure 3: Lab 7 Part 1 - Functional Simulation for Original Code



```
% File name = sram.mif
% ************************************** %

DEPTH = 4096;   % Address Bus Size %
WIDTH = 8;      % Data Bus Size %

ADDRESS_RADIX   = HEX;  % Address Format %
DATA_RADIX              = HEX;  % Data Format %

CONTENT
BEGIN
[0..7FF]        :       FF;     % zero memory   %
800                     :       A2;
801                     :       37;
802                     :       00;
803                     :       9D;
804                     :       9D;
805                     :       02;
806                     :       02;
807                     :       02;
808                     :       02;
809                     :       02;
80A                     :       02;
80B                     :       00;
[80C..FFF]      :       00;

END;
```

Figure 4: Lab 7 Part 1 - Changed sram.mif for 800 from 02 to A2

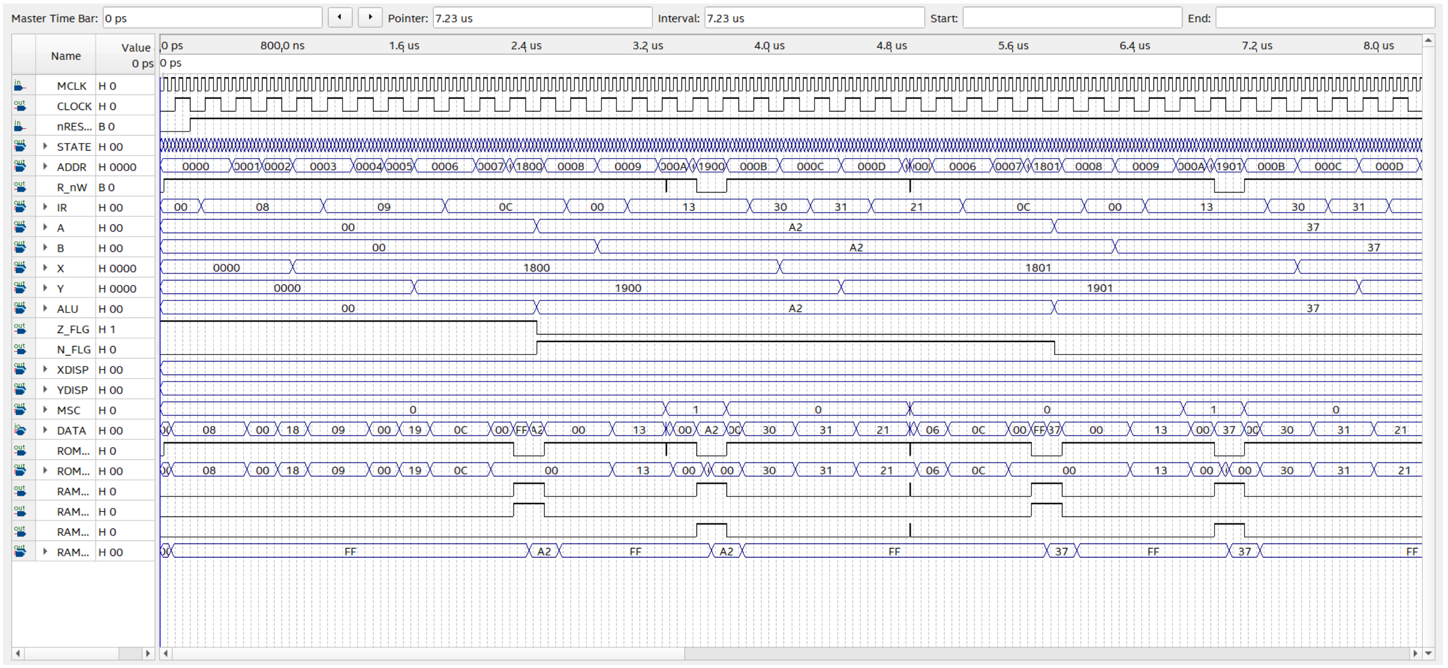
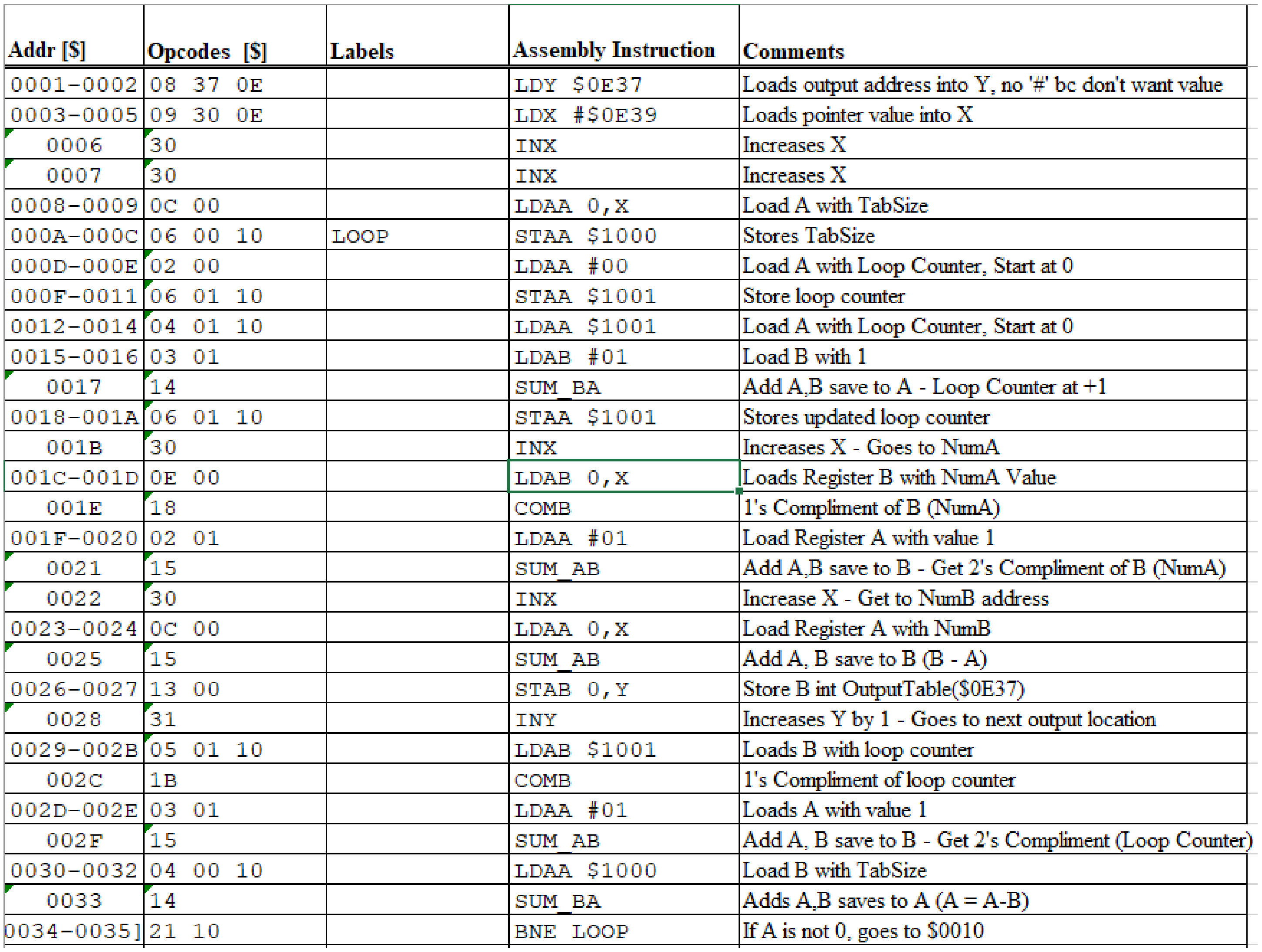


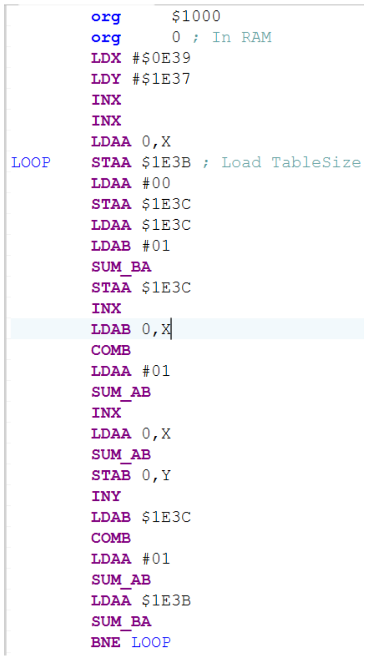
Figure 5: Lab 7 Part 1 - Functional Simulation with changed sram.mif

Part 2: New Program Creation



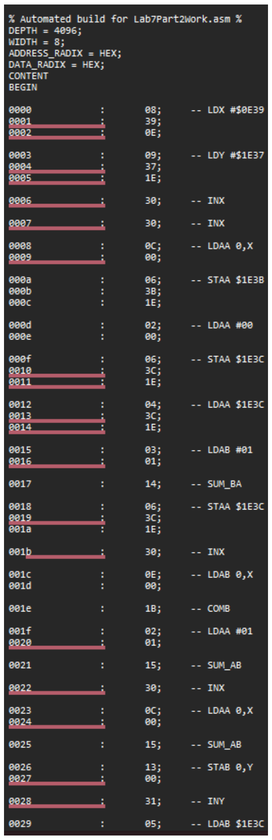
Addr [$]	Opcodes [$]	Labels	Assembly Instruction	Comments
0001-0002	08 37 0E		LDY $0E37	Loads output address into Y, no '#' bc don't want value
0003-0005	09 30 0E		LDX #$0E39	Loads pointer value into X
0006	30		INX	Increases X
0007	30		INX	Increases X
0008-0009	0C 00		LDAA 0,X	Load A with TabSize
000A-000C	06 00 10	LOOP	STAA $1000	Stores TabSize
000D-000E	02 00		LDAA #00	Load A with Loop Counter, Start at 0
000F-0011	06 01 10		STAA $1001	Store loop counter
0012-0014	04 01 10		LDAA $1001	Load A with Loop Counter, Start at 0
0015-0016	03 01		LDAB #01	Load B with 1
0017	14		SUM_BA	Add A,B save to A - Loop Counter at +1
0018-001A	06 01 10		STAA $1001	Stores updated loop counter
001B	30		INX	Increases X - Goes to NumA
001C-001D	0E 00		LDAB 0,X	Loads Register B with NumA Value
001E	18		COMB	1's Compliment of B (NumA)
001F-0020	02 01		LDAA #01	Load Register A with value 1
0021	15		SUM_AB	Add A,B save to B - Get 2's Compliment of B (NumA)
0022	30		INX	Increase X - Get to NumB address
0023-0024	0C 00		LDAA 0,X	Load Register A with NumB
0025	15		SUM_AB	Add A, B save to B (B - A)
0026-0027	13 00		STAB 0,Y	Store B int OutputTable($0E37)
0028	31		INY	Increases Y by 1 - Goes to next output location
0029-002B	05 01 10		LDAB $1001	Loads B with loop counter
002C	1B		COMB	1's Compliment of loop counter
002D-002E	03 01		LDAA #01	Loads A with value 1
002F	15		SUM_AB	Add A, B save to B - Get 2's Compliment (Loop Counter)
0030-0032	04 00 10		LDAA $1000	Load B with TabSize
0033	14		SUM_BA	Adds A,B saves to A (A = A-B)
0034-0035]	21 10		BNE LOOP	If A is not 0, goes to $0010

Figure 6: Lab 7 Part 2 - Table



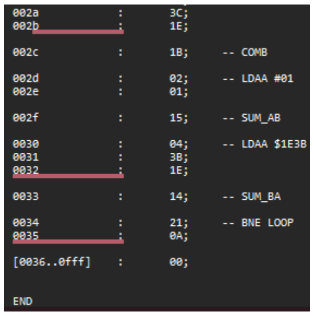
```
        org     $1000
        org     0 ; In RAM
        LDX #$0E39
        LDY #$1E37
        INX
        INX
        LDAA 0,X
LOOP    STAA $1E3B ; Load TableSize
        LDAA #00
        STAA $1E3C
        LDAA $1E3C
        LDAB #01
        SUM_BA
        STAA $1E3C
        INX
        LDAB 0,X
        COMB
        LDAA #01
        SUM_AB
        INX
        LDAA 0,X
        SUM_AB
        STAB 0,Y
        INY
        LDAB $1E3C
        COMB
        LDAA #01
        SUM_AB
        LDAA $1E3B
        SUM_BA
        BNE LOOP
```

Figure 7: Lab 7 Part 2 - Assembly Code



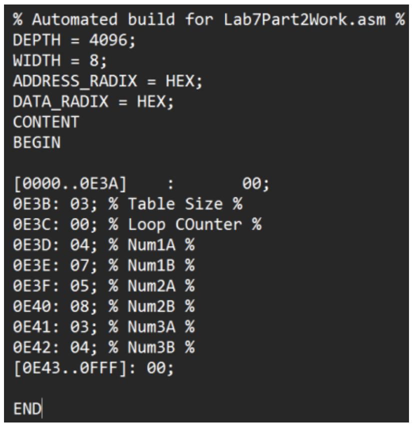
```
% Automated build for Lab7Part2Work.asm %
DEPTH = 4096;
WIDTH = 8;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

0000        :       08;     -- LDX #$0E39
0001        :       39;
0002        :       0E;

0003        :       09;     -- LDY #$1E37
0004        :       37;
0005        :       1E;

0006        :       30;     -- INX

0007        :       30;     -- INX

0008        :       0C;     -- LDAA 0,X
0009        :       00;

000a        :       06;     -- STAA $1E3B
000b        :       3B;
000c        :       1E;

000d        :       02;     -- LDAA #00
000e        :       00;

000f        :       06;     -- STAA $1E3C
0010        :       3C;
0011        :       1E;

0012        :       04;     -- LDAA $1E3C
0013        :       3C;
0014        :       1E;

0015        :       03;     -- LDAB #01
0016        :       01;

0017        :       14;     -- SUM_BA

0018        :       06;     -- STAA $1E3C
0019        :       3C;
001a        :       1E;

001b        :       30;     -- INX

001c        :       0E;     -- LDAB 0,X
001d        :       00;

001e        :       1B;     -- COMB

001f        :       02;     -- LDAA #01
0020        :       01;

0021        :       15;     -- SUM_AB

0022        :       30;     -- INX

0023        :       0C;     -- LDAA 0,X
0024        :       00;

0025        :       15;     -- SUM_AB

0026        :       13;     -- STAB 0,Y
0027        :       00;

0028        :       31;     -- INY

0029        :       05;     -- LDAB $1E3C
```

Figure 8: Lab 7 Part 2 - eprom.mif



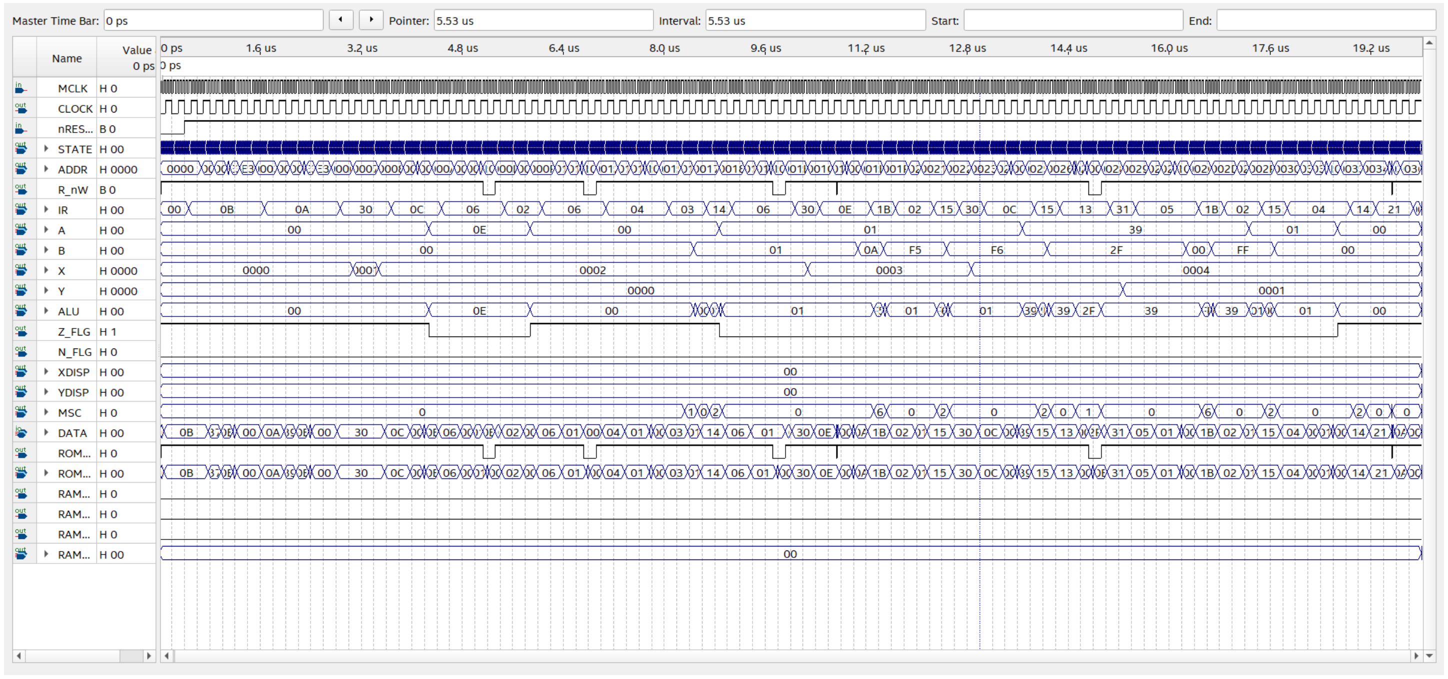
```
002a            :       3C;
002b            :       1E;

002c            :       1B;     -- COMB

002d            :       02;     -- LDAA #01
002e            :       01;

002f            :       15;     -- SUM_AB

0030            :       04;     -- LDAA $1E3B
0031            :       3B;
0032            :       1E;

0033            :       14;     -- SUM_BA

0034            :       21;     -- BNE LOOP
0035            :       0A;

[0036..0fff]    :       00;


END
```

Figure 9: Lab 7 Part 2 - eprom.mif Continued



```
% Automated build for Lab7Part2Work.asm %
DEPTH = 4096;
WIDTH = 8;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

[0000..0E3A]    :       00;
0E3B: 03; % Table Size %
0E3C: 00; % Loop COunter %
0E3D: 04; % Num1A %
0E3E: 07; % Num1B %
0E3F: 05; % Num2A %
0E40: 08; % Num2B %
0E41: 03; % Num3A %
0E42: 04; % Num3B %
[0E43..0FFF]: 00;

END
```

Figure 10: Lab 7 Part 2 - sram.mif



Master Time Bar: 0 ps   Pointer: 5.53 us   Interval: 5.53 us   Start: End:

Figure 11: : Lab 7 Part 2 - Functional Simulation