EEL 3701C: Digital Logic & Computer Systems
Revision **0**
Lab **6** Report: **Elementary CPU Design**

Stern, Arion
Class #: 10844
Erick Zayas Ramos
July 30, 2025

REQUIREMENTS NOT MET

N/A

VIDEO FILE LINK

https://youtube.com/shorts/Wb5eRAEknrk?feature=share

PROBLEMS ENCOUNTERED

For part 1, the only problems I encountered were minor errors in my table, which I revised and corrected, and issues regarding how I should annotate the simulation, but my PI clarified that the way I did it was adequate. For part 2, I had a lot of errors writing the program, which required me to debug using the G-IDE and ask my PI for logic or syntax clarification. I also had a couple of mistakes in my table, which I corrected by comparing the values to those in the G-IDE and simulation.

FUTURE WORK/APPLICATIONS

This lab strengthened skills in using assembly language for data processing, looping, and conditional logic. Future work could focus on applying these techniques to larger and more complex programs or systems. The concepts practiced here are also widely applicable in embedded systems development. Expanding on this foundation could include working with different data formats, optimizing memory usage, or building programs that interact with hardware devices in real-world applications.

PRE-LAB QUESTIONS OR EXERCISES

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

1. Simulating Existing Code:

Briefly describe the purpose of this program:

The program starts at EPROM address 0000 and goes until 000E

Loads X register with immediate address ($1800)

Loads Y register with immediate address($1900)

Performs a loop:

{

Loads A with Data at X + displacement value (displacement value initially = 0)

Transfers Data from A to B

Stores the data in B to memory at location Y + displacement value (displacement value initially = 0)

X and Y are both incremented, moving to the next address

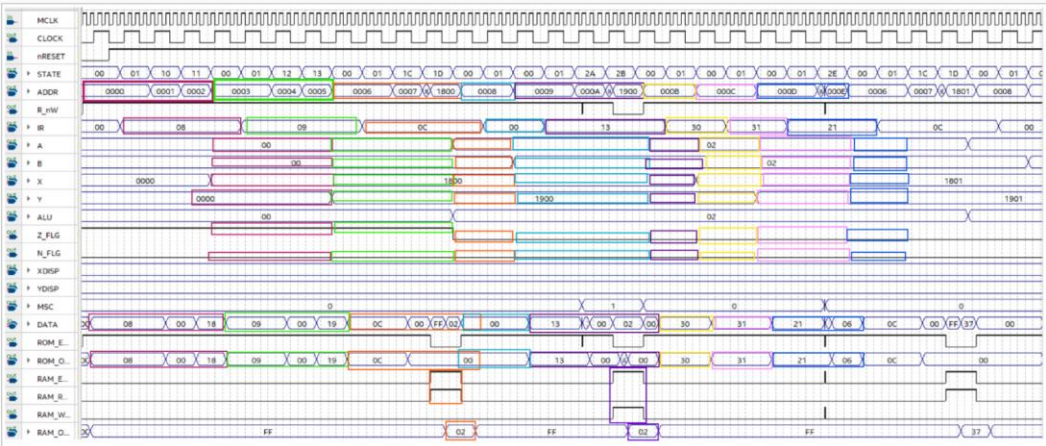
BNE condition at the end of the loop forces the program to loop back to address 6 if the value in REGA does not equal 0; thus, the program loops until a zero is stored in REGA.

}

In summary, the program copies a table of bytes starting from address $1800 in memory to a destination table at $1900. It uses the X register to point to the source and the Y for the destination. Each byte is loaded into A, transferred to B, and then stored in the destination address through the Y register. The program does this until a zero byte is loaded into REGA from the X register.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0000-0002	08 00 18	LDX #$1800	00	00	1800	0000	1	0	0003	Load the address $1800 into X register
0003-0005	09 00 19	LDY #$1900	00	00	1800	1900	1	0	0006	Load the address $1900 into Y register
0006-0007	0C 00	LDAA 0,X	02	00	1800	1900	0	0	0008	Load the value at X + 0 into REGA (02)
0008	0	TAB	02	02	1800	1900	0	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	02	02	1800	1900	0	0	000B	Store REGB (02) into REGY at @1900
000B	30	INX	02	02	1801	1900	0	0	000C	Increment X address by 1
000C	31	INY	02	02	1801	1901	0	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	02	02	1801	1901	0	0	0006	Branch to instruction at 06 since BNE is 0
0006-0007	0C 00	LDAA 0,X	37	02	1801	1901	0	0	0008	Load the value at X + 1 into REGA (37)
0008	0	TAB	37	37	1801	1901	0	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	37	37	1801	1901	0	0	000B	Store REGB (37) into REGY at @1901
000B	30	INX	37	37	1802	1901	0	0	000C	Increment X address by 1
000C	31	INY	37	37	1802	1902	0	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	37	37	1802	1902	0	0	0006	Branch to instruction at 06 since BNE is 0
0006-0007	0C 00	LDAA 0,X	00	00	1802	1902	1	0	0008	Load the value at X + 2 into REGA (00)
0008	0	TAB	00	00	1802	1902	1	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	00	00	1802	1902	1	0	000B	Store REGB (00) into REGY at @1902
000B	30	INX	00	00	1803	1902	1	0	000C	Increment X address by 1
000C	31	INY	00	00	1803	1903	1	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	00	00	1803	1903	1	0	000F	BNE is 1, so increment PC (0s for rest of memory)

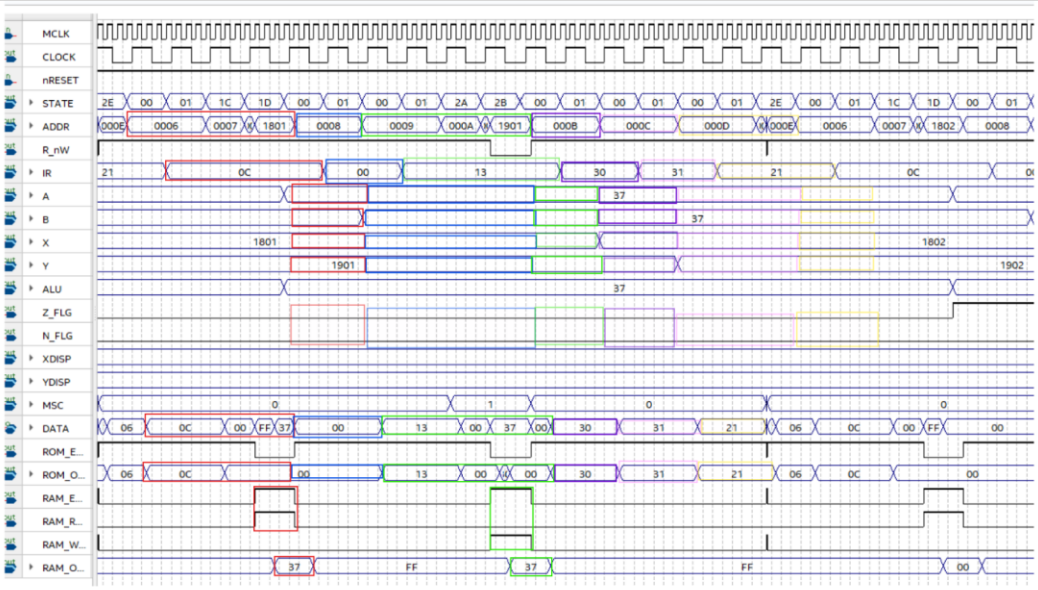
Figure 1: Lab7_PartA Table (Part 1)



The values in the table before the first branch all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM enables and RAM_OUT when relevant.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0000-0002	08 00 18	LDX #$1800	00	00	1800	0000	1	0	0003	Load the address $1800 into X register
0003-0005	09 00 19	LDY #$1900	00	00	1800	1900	1	0	0006	Load the address $1900 into Y register
0006-0007	0C 00	LDAA 0,X	02	00	1800	1900	0	0	0008	Load the value at X + 0 into REGA
0008	0	TAB	02	02	1800	1900	0	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	02	02	1800	1900	0	0	000B	Store REGB (02) into REGY at @1900
000B	30	INX	02	02	1801	1900	0	0	000C	Increment X address by 1
000C	31	INY	02	02	1801	1901	0	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	02	02	1801	1901	0	0	0006	Branch to instruction at 06 since BNE is 0

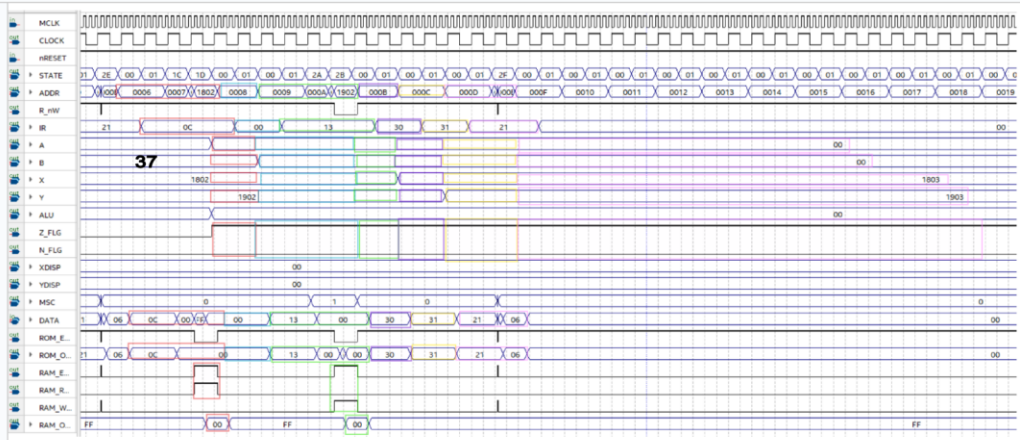
Figure 2: Annotated Simulation for Part 1 Original Values 1/3 (Part 1)



The values in the table after the first branch all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM enables and RAM_OUT when relevant.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0006-0007	0C 00	LDAA 0,X	37	02	1801	1901	0	0	0008	Load the value at X + 1 into REGA
0008	0	TAB	37	37	1801	1901	0	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	37	37	1801	1901	0	0	000B	Store REGB (37) into REGY at @1901
000B	30	INX	37	37	1802	1901	0	0	000C	Increment X address by 1
000C	31	INY	37	37	1802	1902	0	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	37	37	1802	1902	0	0	0006	Branch to instruction at 06 since BNE is 0

Figure 3: Annotated Simulation for Part 1 Original Values 2/3 (Part 1)



The values in the table after the second branch all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM enables and RAM_OUT when relevant.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0006-0007	0C 00	LDAA 0,X	00	00	1802	1902	1	0	0008	Load the value at X + 2 into REGA
0008	0	TAB	00	00	1802	1902	1	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	00	00	1802	1902	1	0	000B	Store REGB (00) into REGY at @1902
000B	30	INX	00	00	1803	1902	1	0	000C	Increment X address by 1
000C	31	INY	00	00	1803	1903	1	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	00	00	1803	1903	1	0	000F	BNE is 1, so increment PC (0s for rest of memory)

Figure 4: Annotated Simulation for Part 1 Original Values 3/3 (Part 1)

Text Editor - C:/QuartusProjects/Lab7/computer - computer - [sram.mif]

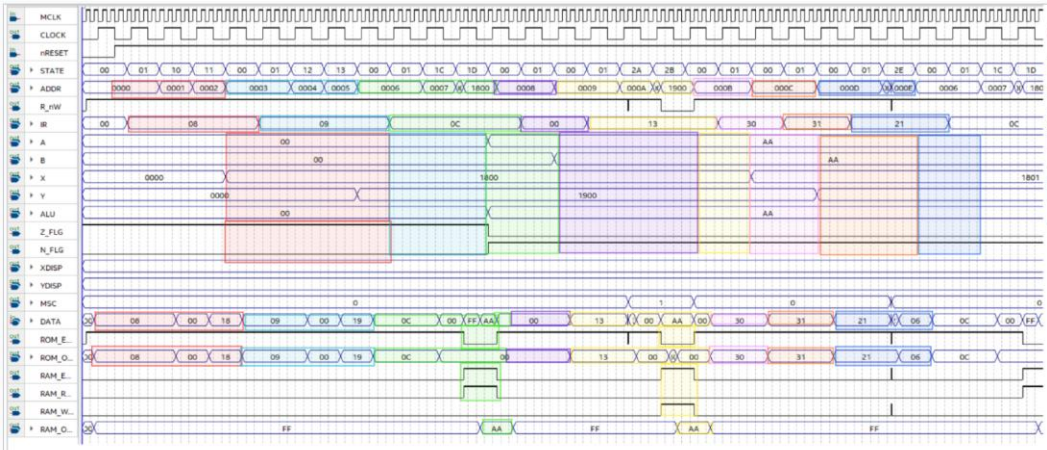
File Edit View Project Processing Tools Window Help

```
1   % File name = sram.mif                %
2   % ************************************ %
3
4   DEPTH = 4096;  % Address Bus Size %
5   WIDTH = 8;  % Data Bus Size %
6
7   ADDRESS_RADIX  = HEX;    % Address Format %
8   DATA_RADIX     = HEX;    % Data Format %
9
10  CONTENT
11  BEGIN
12  --[0..7FF]  :  FF;    % zero memory  %
13  --800       :  02;
14  --801       :  37;
15  --802       :  00;
16  --803       :  9D;
17  --804       :  9D;
18  --805       :  02;
19  --806       :  02;
20  --807       :  02;
21  --808       :  02;
22  --809       :  02;
23  --80A       :  02;
24  --80B       :  00;
25  --[80C..FFF]   :  00;
26
27  [0..7FF] :  FF;   % zero memory  %
28  800         :  AA;
29  801         :  BB;
30  802         :  00;
31  803         :  04;
32  [804..FFF]  :  00;
33
34  END;
35
36
```

Figure 5: Modified Data in sram.mif (Part 1)

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0000-0002	08 00 18	LDX #$1800	00	00	1800	0000	1	0	0003	Load the address $1800 into X register
0003-0005	09 00 19	LDY #$1900	00	00	1800	1900	1	0	0006	Load the address $1900 into Y register
0006-0007	0C 00	LDAA 0,X	AA	00	1800	1900	0	1	0008	Load the value at X + 0 into REGA (AA)
0008	0	TAB	AA	AA	1800	1900	0	1	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	AA	AA	1800	1900	0	1	000B	Store REGB (AA) into REGY at @1900
000B	30	INX	AA	AA	1801	1900	0	1	000C	Increment X address by 1
000C	31	INY	AA	AA	1801	1901	0	1	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	AA	AA	1801	1901	0	1	0006	Branch to instruction at 06 since BNE is 0
0006-0007	0C 00	LDAA 0,X	BB	AA	1801	1901	0	1	0008	Load the value at X + 1 into REGA (BB)
0008	0	TAB	BB	BB	1801	1901	0	1	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	BB	BB	1801	1901	0	1	000B	Store REGB (BB) into REGY at @1901
000B	30	INX	BB	BB	1802	1901	0	1	000C	Increment X address by 1
000C	31	INY	BB	BB	1802	1902	0	1	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	BB	BB	1802	1902	0	1	0006	Branch to instruction at 06 since BNE is 0
0006-0007	0C 00	LDAA 0,X	00	00	1802	1902	1	0	0008	Load the value at X + 2 into REGA (00)
0008	0	TAB	00	00	1802	1902	1	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	00	00	1802	1902	1	0	000B	Store REGB (00) into REGY at @1902
000B	30	INX	00	00	1803	1902	1	0	000C	Increment X address by 1
000C	31	INY	00	00	1803	1903	1	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	00	00	1803	1903	1	0	000F	BNE is 1, so increment PC (0s for rest of memory)

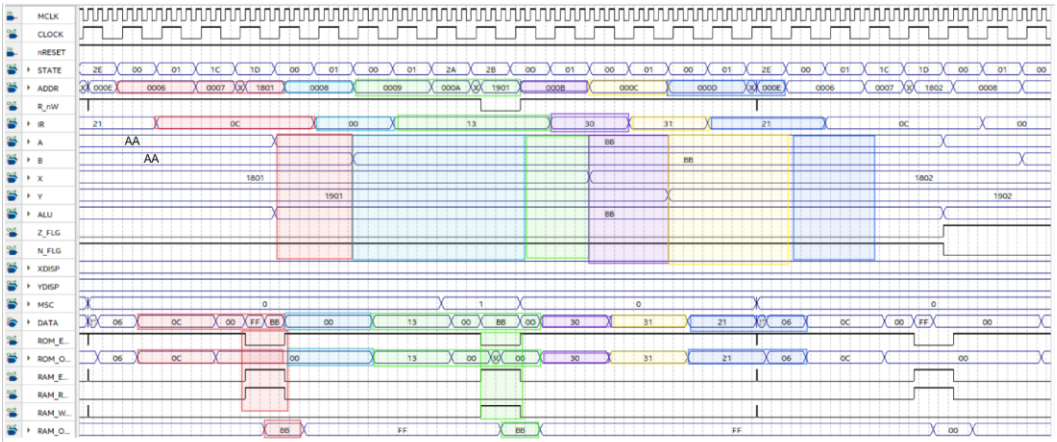
Figure 6: Lab7_PartA Table for Part 1 new values (Part 1)



The values in the table before the first branch all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM/ROM enables and RAM_OUT when relevant.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0000-0002	08 00 18	LDX #$1800	00	00	1800	0000	1	0	0003	Load the address $1800 into X register
0003-0005	09 00 19	LDY #$1900	00	00	1800	1900	1	0	0006	Load the address $1900 into Y register
0006-0007	0C 00	LDAA 0,X	AA	00	1800	1900	0	1	0008	Load the value at X + 0 into REGA (AA)
0008	0	TAB	AA	AA	1800	1900	0	1	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	AA	AA	1800	1900	0	1	000B	Store REGB (AA) into REGY at @1900
000B	30	INX	AA	AA	1801	1900	0	1	000C	Increment X address by 1
000C	31	INY	AA	AA	1801	1901	0	1	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	AA	AA	1801	1901	0	1	0006	Branch to instruction at 06 since BNE is 0

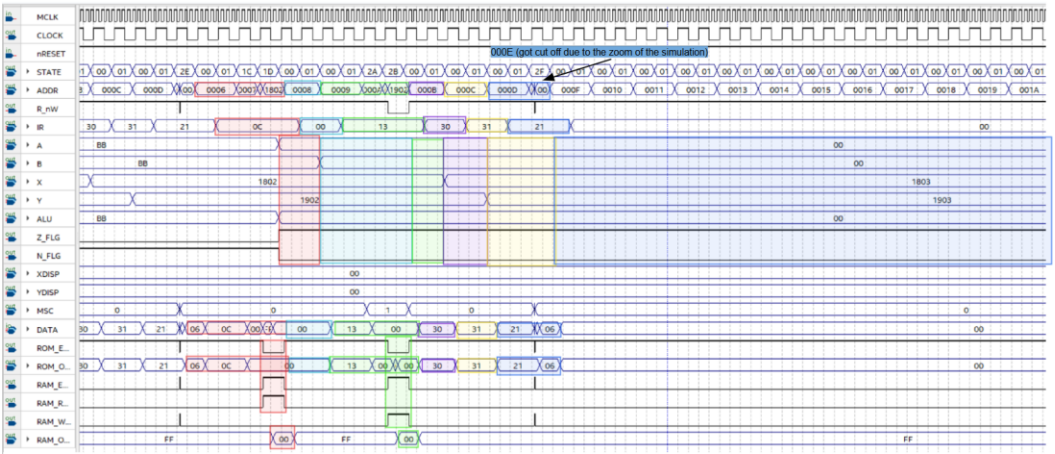
Figure 7: Annotated Simulation for Part 1 New Values 1/3 (Part 1)



Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0006-0007	0C 00	LDAA 0,X	BB	AA	1801	1901	0	1	0008	Load the value at X + 1 into REGA (BB)
0008	0	TAB	BB	BB	1801	1901	0	1	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	BB	BB	1801	1901	0	1	000B	Store REGB (BB) into REGY at @1901
000B	30	INX	BB	BB	1802	1901	0	1	000C	Increment X address by 1
000C	31	INY	BB	BB	1802	1902	0	1	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	BB	BB	1802	1902	0	1	0006	Branch to instruction at 06 since BNE is 0

The values in the table after the first branch all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM/ROM enables and RAM_OUT when relevant.

Figure 8: Annotated Simulation for Part 1 New Values 2/3 (Part 1)



Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0006-0007	0C 00	LDAA 0,X	00	00	1802	1902	1	0	0008	Load the value at X + 2 into REGA (00)
0008	0	TAB	00	00	1802	1902	1	0	0009	Transfer REGA to REGB
0009-000A	13 00	STAB 0,Y	00	00	1802	1902	1	0	000B	Store REGB (00) into REGY at @1902
000B	30	INX	00	00	1803	1902	1	0	000C	Increment X address by 1
000C	31	INY	00	00	1803	1903	1	0	000D	Increment Y address by 1
000D-000E	21 06	BNE 06	00	00	1803	1903	1	0	000F	BNE is 1, so increment PC (0s for rest of memory)

The values in the table after the second branch all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM/ROM enables and RAM_OUT when relevant.

Figure 9: Annotated Simulation for Part 1 New Values 3/3 (Part 1)

2. NEW PROGRAM CREATION:

```
1  NEG1     EQU      $FF
2
3           org      $03A7
4  InTab    dc.b     $73, $1C, $3,
5                    $1, $2,
6                    $73, $37,
7                    $7F, $7E,
8
9
10
11          org      0
12          LDX      #InTab
13
14          ldaa     0,X      ; load low address byte
15          staa     $1000    ; store in $1000
16          INX
17
18          ldaa     0,X      ; load high address byte
19          staa     $1001    ; store in $1001
20          INX
21
22          LDY      $1000    ; load the stored address
23                           ; into Y reg
24
25          ldaa     0,X      ; load table size into REGA
26          STAA     $1002    ; store in $1002
27          INX
28
29 Loop     ldaa     0,X
30          INX
31          ldab     0,X
32
33 ; TEMP store A in SRAM
34          staa     $100A
35 ; Temp store B in SRAM
36          stab     $100B
37 ; Load B with 1
38          Ldab     #$1
39 ; COMP A and add 1 to get 2's comp, store in A
40          COMA
41          SUM_BA
```

Figure 10: G-IDE Program 1/2 (Part 2)

```
42  ; Load temp B back into B
43          LDAB    $100B
44  ; B - A (SUM with A and store in A)
45          SUM_BA
46  ; If negative: B is smaller,
47  ;store B in Y reg then to after A condition
48          STAB    0,Y
49          BN      afterA
50
51  ; Else load the temp A value back in REGA
52  ; store A in Y reg
53
54          ldaa    $100A
55          staa    0,Y
56
57  afterA
58
59          INX     ; inc X and Y
60          INY
61
62          ldaa    $1002; DECA
63          ldab    #NEG1
64          SUM_BA
65          staa    $1002
66
67          BNE     Loop
68
69  end     BEQ     end
70
```

Figure 11: G-IDE Program 2/2 (Part 2)

	Addr [$]	Opcodes [$]	Assembly Instruction	Comments
2	0000	08 A7 03	LDX #InTab (#$03A7)	Load the input table into X
3				
4	0003	0C 00	LDAA 0,X	load low address byte
5	0005	06 00 10	STAA $1000	store in $1000
6	0008	30	INX	Inc X
7				
8	0009	0C 00	LDAA 0,X	load high address byte
9	000B	06 01 10	STAA $1001	store in $1001
10	000E	30	INX	Inc X
11	000F	0B 00 10	LDY $1000	load the stored address into Y reg
12				
13	0012	0C 00	LDAA 0,X	load table size into REGA
14	0014	06 02 10	STAA $1002	store in $1002
15	0017	30	INX	Inc X
16				
17	0018	0C 00	LDAA 0,X	Start of loop, load A
18	001A	30	INX	Inc X
19	001B	0E 00	LDAB 0,X	load b
20				
21	001D	06 0A 10	STAA $100A	TEMP store A in SRAM
22	0020	07 0B 10	STAB $100B	Temp store B in SRAM
23				
24	0023	03 01	LDAB #$1	Load B with 1
25	0025	1A	COMA	Comp A
26	0026	14	SUM_BA	Sum them to make 2's Comp -A
27				
28	0027	05 0B 10	LDAB $100B	Load temp B back into B
29	002A	14	SUM_BA	Sum B with negative A (B - A) store in A
30				
31	002B	13 00	STAB 0,Y	Store B in Y
32	002D	22 34	BN afterA (34)	Branch to after A condition if Neg Flag true (B smaller)
33				
34	002F	04 0A 10	LDAA $100A	Load Temp A value back
35	0032	11 00	STAA 0,Y	store A in Y reg
36				
37	0034	30	INX	inc X
38	0035	31	INY	inc Y
39				
40	0036	04 02 10	LDAA $1002	load table size into A
41	0039	03 FF	LDAB #NEG1 (FF)	load Neg 1 (FF) into B
42	003B	14	SUM_BA	sum them to decrease table size by one
43	003C	06 02 10	STAA $1002	store the value back in SRAM
44				
45	003F	21 18	BNE loop	Loop if count != 0
46				
47	0041	20 41	BEQ end	infinite loop

Figure 12: Hand Assembly for G-IDE Program (Part 2)

```
DEPTH = 4096;
WIDTH = 8;
ADDRESS_RADIX = HEX;
DATA_RADIX = HEX;
CONTENT
BEGIN

0000     :  08;   -- LDX #InTab
0001     :  A7;
0002     :  03;

0003     :  0C;   -- ldaa 0,X
0004     :  00;

0005     :  06;   -- staa $1000
0006     :  00;
0007     :  10;

0008     :  30;   -- INX

0009     :  0C;   -- ldaa 0,X
000a     :  00;

000b     :  06;   -- staa $1001
000c     :  01;
000d     :  10;

000e     :  30;   -- INX

000f     :  0B;   -- LDY $1000
0010     :  00;
0011     :  10;

0012     :  0C;   -- ldaa 0,X
0013     :  00;

0014     :  06;   -- STAA $1002
0015     :  02;
0016     :  10;

0017     :  30;   -- INX

0018     :  0C;   -- ldaa 0,X
0019     :  00;

001a     :  30;   -- INX

001b     :  0E;   -- ldab 0,X
001c     :  00;

001d     :  06;   -- staa $100A
001e     :  0A;
001f     :  10;

0020     :  07;   -- stab $100B
0021     :  0B;
0022     :  10;

0023     :  03;   -- Ldab #$1
0024     :  01;

0025     :  1A;   -- COMA

```

Figure 13: EPROM.MIF to Test Program 1/2 (Part 2)

```
107     0026     :  14;   -- SUM_BA
108
109     0027     :  05;   -- LDAB $100B
110     0028     :  0B;
111     0029     :  10;
112
113     002a     :  14;   -- SUM_BA
114
115     002b     :  13;   -- STAB 0,Y
116     002c     :  00;
117
118     002d     :  22;   -- BN afterA
119     002e     :  34;
120
121     002f     :  04;   -- ldaa $100A
122     0030     :  0A;
123     0031     :  10;
124
125     0032     :  11;   -- staa 0,Y
126     0033     :  00;
127
128     0034     :  30;   -- INX
129
130     0035     :  31;   -- INY
131
132     0036     :  04;   -- ldaa $1002
133     0037     :  02;
134     0038     :  10;
135
136     0039     :  03;   -- ldab #NEG1
137     003a     :  FF;
138
139     003b     :  14;   -- SUM_BA
140
141     003c     :  06;   -- staa $1002
142     003d     :  02;
143     003e     :  10;
144
145     003f     :  21;   -- BNE Loop
146     0040     :  18;
147
148     0041     :  20;   -- BEQ end
149     0042     :  41;
150
151     [0043..03a6]   :  00;
152
153     03a7     :  73;   -- InTab dc.b $73,$1C,$3,$1,$2,$73,$37,$7F,$7E,
154     03a8     :  1C;
155     03a9     :  03;
156     03aa     :  01;
157     03ab     :  02;
158     03ac     :  73;
159     03ad     :  37;
160     03ae     :  7F;
161     03af     :  7E;
162
163     [03b0..0fff]   :  00;
164
165
166     END
```

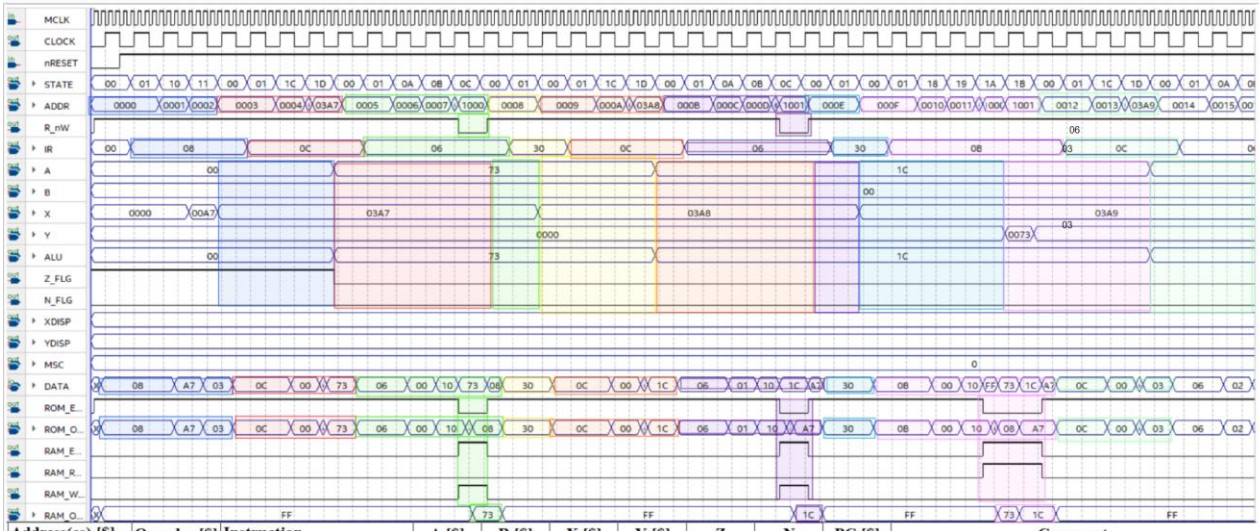
Figure 14EPROM.MIF to Test Program 2/2 (Part 2)

Address(es) [S]	Opcodes [S]	Instruction	A [S]	B [S]	X [S]	Y [S]	Z	N	PC [S]	Comments
0000-0002	08 A7 03	LDX #InTab (#$03A7)	00	00	03A7	0000	1	0	0003	Load the input table into X
0003-0004	0C 00	LDAA 0,X	73	00	03A7	0000	0	0	0005	load low address byte (73)
0005-0007	06 00 10	STAA $1000	73	00	03A7	0000	0	0	0008	store in $1000
0008	30	INX	73	00	03A8	0000	0	0	0009	Inc X
0009-000A	0C 00	LDAA 0,X	1C	00	03A8	0000	0	0	000B	load high address byte (1C)
000B-000D	06 01 10	STAA $1001	1C	00	03A8	0000	0	0	000E	store in $1001
000E	30	INX	1C	00	03A9	0000	0	0	000F	Inc X
000F-0011	0B 00 10	LDY $1000	1C	00	03A9	1C73	0	0	0012	load the stored address into Y reg (1C73)
0012-0013	0C 00	LDAA 0,X	03	00	03A9	1C73	0	0	0014	load table size into REGA (03)
0014-0016	06 02 10	STAA $1002	03	00	03A9	1C73	0	0	0017	store in $1002
0017	30	INX	03	00	03AA	1C73	0	0	0018	Inc X
0018-0019	0C 00	LDAA 0,X	01	00	03AA	1C73	0	0	001A	Start of loop, load A (01)
001A	30	INX	01	00	03AB	1C73	0	0	001B	Inc X
001B-001C	0E 00	LDAB 0,X	01	02	03AB	1C73	0	0	001D	load b (02)
001D-001F	06 0A 10	STAA $100A	01	02	03AB	1C73	0	0	0020	TEMP store A in SRAM at $100A
0020-0022	07 0B 10	STAB $100B	01	02	03AB	1C73	0	0	0023	Temp store B in SRAM at $100B
0023-0024	03 01	LDAB #$1	01	01	03AB	1C73	0	0	0025	Load B with 1
0025	1A	COMA	FE	01	03AB	1C73	0	1	0026	Comp A
0026	14	SUM_BA	FF	01	03AB	1C73	0	1	0027	Sum them to make 2's Comp -A (FF)
0027-0029	05 0B 10	LDAB $100B	FF	02	03AB	1C73	0	1	002A	Load temp B back into B
002A	14	SUM_BA	01	02	03AB	1C73	0	0	002B	Sum B with negative A (B - A) store in A
002B-002C	13 00	STAB 0,Y	01	02	03AB	1C73	0	0	002D	Store B in Y
002D-002E	22 34	BN afterA (34)	01	02	03AB	1C73	0	0	002F	Branch to after A condition if Neg Flag true (B smaller)
002F-0031	04 0A 10	LDAA $100A	01	02	03AB	1C73	0	0	0032	Load Temp A value back
0032-0033	11 00	STAA 0,Y	01	02	03AB	1C73	0	0	0034	store A in Y reg
0034	30	INX	01	02	03AC	1C73	0	0	0035	inc X
0035	31	INY	01	02	03AC	1C74	0	0	0036	inc Y
0036-0038	04 02 10	LDAA $1002	03	02	03AC	1C74	0	0	0039	load table size into A
0039-003A	03 FF	LDAB #NEG1 (FF)	03	FF	03AC	1C74	0	0	003B	load Neg 1 (FF) into B
003B	14	SUM_BA	02	FF	03AC	1C74	0	0	003C	sum them to decrease table size by one
003C-003E	06 02 10	STAA $1002	02	FF	03AC	1C74	0	0	003F	store the value back in SRAM at $1002
003F-0040	21 18	BNE loop	02	FF	03AC	1C74	0	0	0018	Loop if count != 0
0041-0042	20 41	BEQ end	X	X	X	X	X	X	0041	infinite loop (does not get to this line bc of loop)
0018-0019	0C 00	LDAA 0,X	73	FF	03AC	1C74	0	0	001A	Start of loop, load A (73)
001A	30	INX	73	FF	03AD	1C74	0	0	001B	Inc X
001B-001C	0E 00	LDAB 0,X	73	37	03AD	1C74	0	0	001D	load b (37)
001D-001F	06 0A 10	STAA $100A	73	37	03AD	1C74	0	0	0020	TEMP store A in SRAM at $100A
0020-0022	07 0B 10	STAB $100B	73	37	03AD	1C74	0	0	0023	Temp store B in SRAM at $100B
0023-0024	03 01	LDAB #$1	73	01	03AD	1C74	0	0	0025	Load B with 1
0025	1A	COMA	8C	01	03AD	1C74	0	1	0026	Comp A
0026	14	SUM_BA	8D	01	03AD	1C74	0	1	0027	Sum them to make 2's Comp -A (8D)
0027-0029	05 0B 10	LDAB $100B	8D	37	03AD	1C74	0	1	002A	Load temp B back into B
002A	14	SUM_BA	C4	37	03AD	1C74	0	1	002B	Sum B with negative A (B - A) store in A
002B-002C	13 00	STAB 0,Y	C4	37	03AD	1C74	0	1	002D	Store B in Y

Figure 15: Lab7_PartA prediction table for G-IDE program 1/2 (Part 2)

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
002D-002E	22 34	BN afterA (34)	C4	37	03AD	1C74	0	1	0034	Branch to after A condition if Neg Flag true (B smaller)
002F-0031	04 0A 10	LDAA $100A	X	X	X	X	X	X	X	Load Temp A value back (Skip because of N flag)
0032-0033	11 00	STAA 0,Y	X	X	X	X	X	X	X	store A in Y reg (Skip because of N flag)
0034	30	INX	C4	37	03AE	1C74	0	1	0035	inc X
0035	31	INY	C4	37	03AE	1C75	0	1	0036	inc Y
0036-0038	04 02 10	LDAA $1002	02	37	03AE	1C75	0	0	0039	load table size into A
0039-003A	03 FF	LDAB #NEG1 (FF)	02	FF	03AE	1C75	0	0	003B	load Neg 1 (FF) into B
003B	14	SUM_BA	01	FF	03AE	1C75	0	0	003C	sum them to decrease table size by one
003C-003E	06 02 10	STAA $1002	01	FF	03AE	1C75	0	0	003F	store the value back in SRAM
003F-0040	21 18	BNE loop	01	FF	03AE	1C75	0		0018	Loop if count != 0
0041-0042	20 41	BEQ end	X	X	X	X	X	X	0041	infinite loop (does not get to this line bc of loop)
0018-0019	0C 00	LDAA 0,X	7F	FF	03AE	1C75	0	0	001A	Start of loop, load A (01)
001A	30	INX	7F	FF	03AF	1C75	0	0	001B	Inc X
001B-001C	0E 00	LDAB 0,X	7F	7E	03AF	1C75	0	0	001D	load b (02)
001D-001F	06 0A 10	STAA $100A	7F	7E	03AF	1C75	0	0	0020	TEMP store A in SRAM
0020-0022	07 0B 10	STAB $100B	7F	7E	03AF	1C75	0	0	0023	Temp store B in SRAM
0023-0024	03 01	LDAB #$1	7F	01	03AF	1C75	0	0	0025	Load B with 1
0025	1A	COMA	80	01	03AF	1C75	0	1	0026	Comp A
0026	14	SUM_BA	81	01	03AF	1C75	0	1	0027	Sum them to make 2's Comp -A (81)
0027-0029	05 0B 10	LDAB $100B	81	7E	03AF	1C75	0	1	002A	Load temp B back into B
002A	14	SUM_BA	FF	7E	03AF	1C75	0	1	002B	Sum B with negative A (B - A) store in A
002B-002C	13 00	STAB 0,Y	FF	7E	03AF	1C75	0	1	002D	Store B in Y
002D-002E	22 34	BN afterA (34)	FF	7E	03AF	1C75	0	1	0034	Branch to after A condition if Neg Flag true (B smaller)
002F-0031	04 0A 10	LDAA $100A	X	X	X	X	X	X	X	Load Temp A value back (Skip because of N flag)
0032-0033	11 00	STAA 0,Y	X	X	X	X	X	X	X	store A in Y reg (Skip because of N flag)
0034	30	INX	FF	7E	03B0	1C75	0	1	0035	inc X
0035	31	INY	FF	7E	03B0	1C76	0	1	0036	inc Y
0036-0038	04 02 10	LDAA $1002	01	7E	03B0	1C76	0	1	0039	load table size into A
0039-003A	03 FF	LDAB #NEG1 (FF)	01	FF	03B0	1C76	0	0	003B	load Neg 1 (FF) into B
003B	14	SUM_BA	00	FF	03B0	1C76	1	0	003C	sum them to decrease table size by one
003C-003E	06 02 10	STAA $1002	00	FF	03B0	1C76	1	0	003F	store the value back in SRAM
003F-0040	21 18	BNE loop	00	FF	03B0	1C76	1	0	0041	Loop if count != 0
0041-0042	20 41	BEQ end	00	FF	03B0	1C76	1	0	0041	infinite loop (executes this line because the count = 0

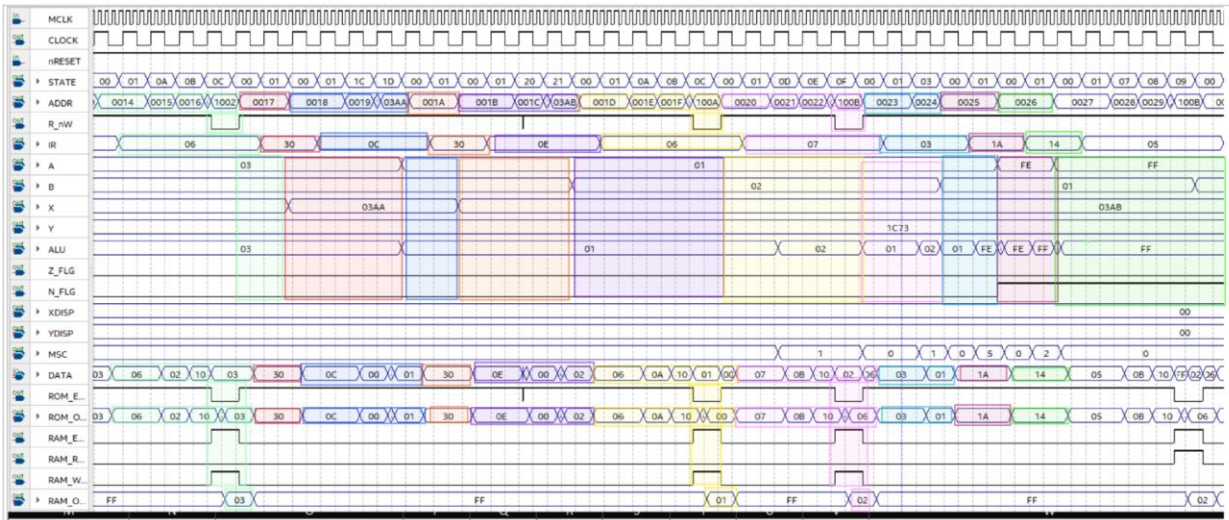
Figure 16: Lab7_PartA prediction table for G-IDE program 2/2 (Part 2)



The values in the table all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM/ROM enables and RAM_OUT when relevant.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0000-0002	08 A7 03	LDX #InTab (#$03A7)	00	00	03A7	0000	1	0	0003	Load the input table into X
0003-0004	0C 00	LDAA 0,X	73	00	03A7	0000	0	0	0005	load low address byte (73)
0005-0007	06 00 10	STAA $1000	73	00	03A7	0000	0	0	0008	store in $1000
0008	30	INX	73	00	03A8	0000	0	0	0009	Inc X
0009-000A	0C 00	LDAA 0,X	1C	00	03A8	0000	0	0	000B	load high address byte (1C)
000B-000D	06 01 10	STAA $1001	1C	00	03A8	0000	0	0	000E	store in $1001
000E	30	INX	1C	00	03A9	0000	0	0	000F	Inc X
000F-0011	0B 00 10	LDY $1000	1C	00	03A9	1C73	0	0	0012	load the stored address into Y reg
0012-0013	0C 00	LDAA 0,X	03	00	03A9	1C73	0	0	0014	load table size into REGA (03)

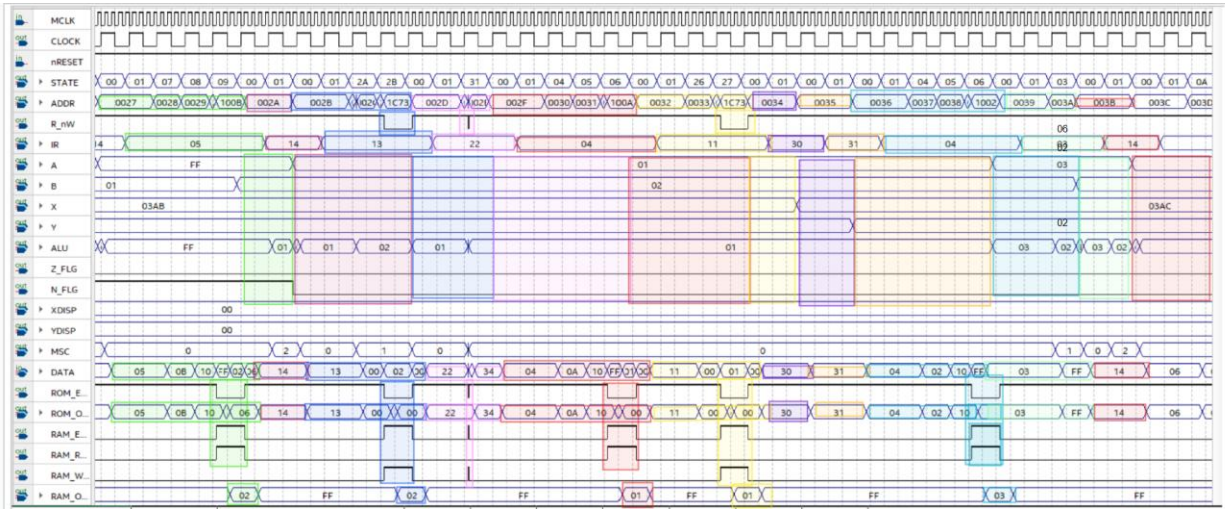
Figure 17: Annotated Simulation for G-IDE program 1/4 (Part 2)



The values in the table all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM/ROM enables and RAM_OUT when relevant.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0014-0016	06 02 10	STAA $1002	03	00	03A9	1C73	0	0	0017	store in $1002
0017	30	INX	03	00	03AA	1C73	0	0	0018	Inc X
0018-0019	0C 00	LDAA 0,X	01	00	03AA	1C73	0	0	001A	Start of loop, load A (01)
001A	30	INX	01	00	03AB	1C73	0	0	001B	Inc X
001B-001C	0E 00	LDAB 0,X	01	02	03AB	1C73	0	0	001D	load b (02)
001D-001F	06 0A 10	STAA $100A	01	02	03AB	1C73	0	0	0020	TEMP store A in SRAM at $100A
0020-0022	07 0B 10	STAB $100B	01	02	03AB	1C73	0	0	0023	Temp store B in SRAM at $100B
0023-0024	03 01	LDAB #$1	01	01	03AB	1C73	0	0	0025	Load B with 1
0025	1A	COMA	FE	01	03AB	1C73	0	1	0026	Comp A
0026	14	SUM_BA	FF	01	03AB	1C73	0	1	0027	Sum them to make 2's Comp -A (FF)

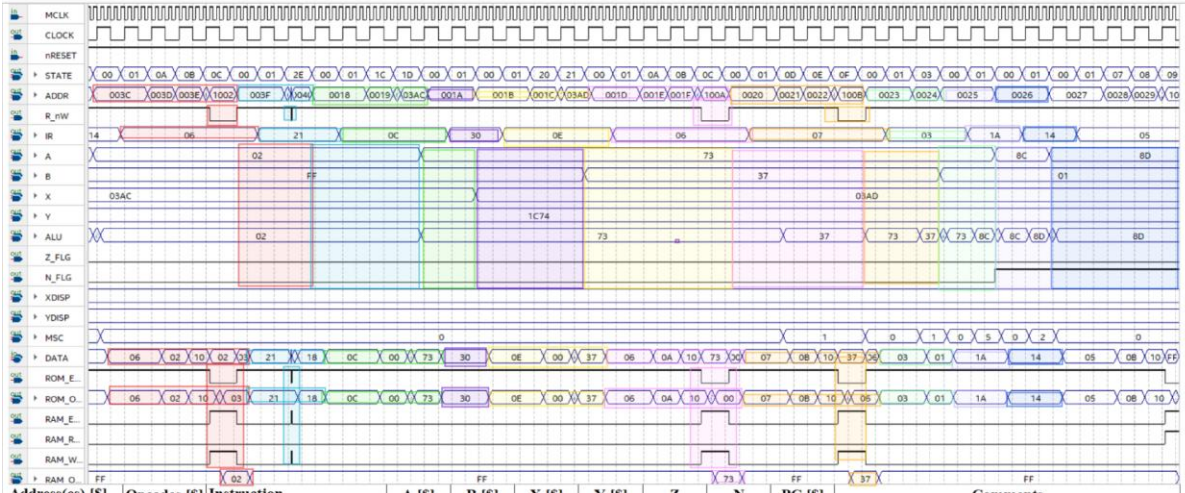
Figure 18: Annotated Simulation for G-IDE program 2/4 (Part 2)



The values in the table all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM/ROM enables and RAM_OUT when relevant.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
0027-0029	05 0B 10	LDAB $100B	FF	02	03AB	1C73	0	1	002A	Load temp B back into B
002A	14	SUM_BA	01	02	03AB	1C73	0	0	002B	Sum B with negative A (B - A) store in A
002B-002C	13 00	STAB 0,Y	01	02	03AB	1C73	0	0	002D	Store B in Y
002D-002E	22 34	BN afterA (34)	01	02	03AB	1C73	0	0	002F	Branch to after A condition if Neg Flag true (B smaller)
002F-0031	04 0A 10	LDAA $100A	01	02	03AB	1C73	0	0	0032	Load Temp A value back
0032-0033	11 00	STAA 0,Y	01	02	03AB	1C73	0	0	0034	store A in Y reg
0034	30	INX	01	02	03AC	1C73	0	0	0035	inc X
0035	31	INY	01	02	03AC	1C74	0	0	0036	inc Y
0036-0038	04 02 10	LDAA $1002	03	02	03AC	1C74	0	0	0039	load table size into A
0039-003A	03 FF	LDAB #NEG1 (FF)	03	FF	03AC	1C74	0	0	003B	load Neg 1 (FF) into B
003B	14	SUM_BA	02	FF	03AC	1C74	0	0	003C	sum them to decrease table size by one

Figure 19: Annotated Simulation for G-IDE program 3/4 (Part 2)



The values in the table all match the values in the simulation. I color coded the simulation to highlight the addresses, IR, A, B, X, Y, Z-flag, N-flag, data, and ROM_OUT. I also highlighted the RAM/ROM enables and RAM_OUT when relevant. Blank row is for when the loop restarted.

Address(es) [$]	Opcodes [$]	Instruction	A [$]	B [$]	X [$]	Y [$]	Z	N	PC [$]	Comments
003C-003E	06 02 10	STAA $1002	02	FF	03AC	1C74	0	0	003F	store the value back in SRAM at $1002
003F-0040	21 18	BNE loop	02	FF	03AC	1C74	0	0	0018	Loop if count != 0
0041-0042	20 41	BEQ end	X	X	X	X	X	X	0041	infinite loop
0018-0019	0C 00	LDAA 0,X	73	FF	03AC	1C74	0	0	001A	Start of loop, load A (73)
001A	30	INX	73	FF	03AD	1C74	0	0	001B	Inc X
001B-001C	0E 00	LDAB 0,X	73	37	03AD	1C74	0	0	001D	load b (37)
001D-001F	06 0A 10	STAA $100A	73	37	03AD	1C74	0	0	0020	TEMP store A in SRAM at $100A
0020-0022	07 0B 10	STAB $100B	73	37	03AD	1C74	0	0	0023	Temp store B in SRAM at $100B
0023-0024	03 01	LDAB #$1	73	01	03AD	1C74	0	0	0025	Load B with 1
0025	1A	COMA	8C	01	03AD	1C74	0	1	0026	Comp A
0026	14	SUM_BA	8D	01	03AD	1C74	0	1	0027	Sum them to make 2's Comp -A (8D)

Figure 20: Annotated Simulation for G-IDE program 4/4 (Part 2)