

Page 1) 10 pt. Steph + Tomasz

Page 2) 14 pt. 3A/3B Yikai  
4 Lysny + Danny B

2 Marguez, Dan O.

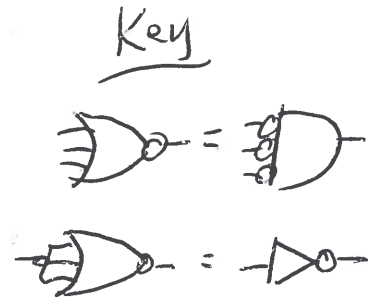
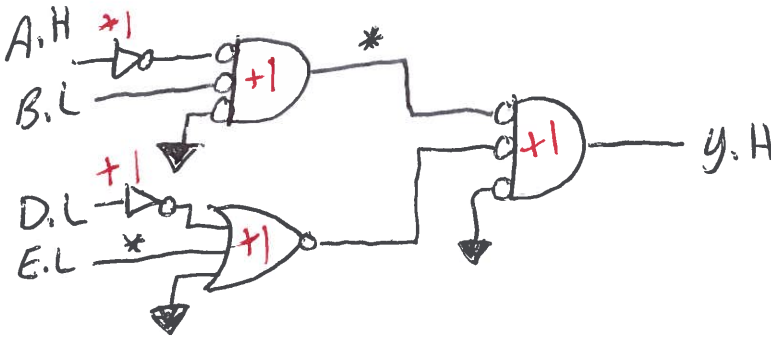
Page 3) 14 pt. C + D  
6 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 + 16 + 17 + 18 + 19 + 20 + 21 + 22 + 23 + 24 + 25 + 26 + 27 + 28 + 29 + 30 + 31 + 32 + 33 + 34 + 35 + 36 + 37 + 38 + 39 + 40 + 41 + 42 + 43 + 44 + 45 + 46 + 47 + 48 + 49 + 50 + 51 + 52 + 53 + 54 + 55 + 56 + 57 + 58 + 59 + 60 + 61 + 62 + 63 + 64 + 65 + 66 + 67 + 68 + 69 + 70 + 71 + 72 + 73 + 74 + 75 + 76 + 77 + 78 + 79 + 80 + 81 + 82 + 83 + 84 + 85 + 86 + 87 + 88 + 89 + 90 + 91 + 92 + 93 + 94 + 95 + 96 + 97 + 98 + 99 + 100

Page 4) 13 pt. Su + Patrick

Daniel C. #5 Alex B, Kevin, Tom TOTAL \_\_\_\_\_ (51)

1. Implement the logic equation below using only 3 Input NOR gates. (5 pt.)

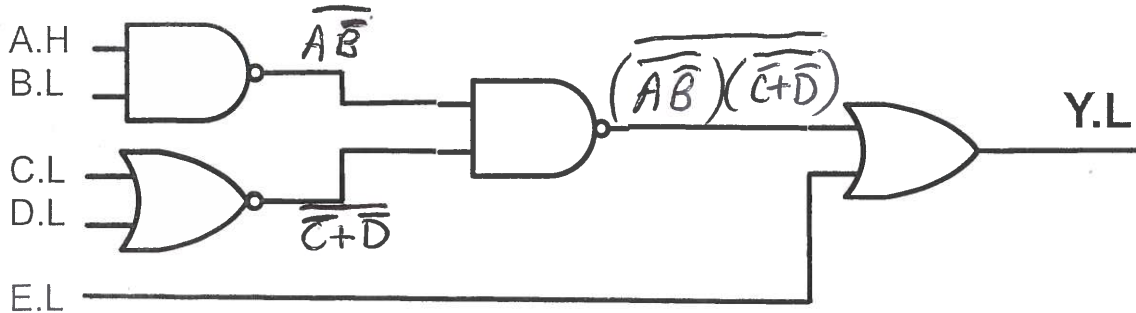
$$Y = \overline{(A*B)} * (D + \overline{E}) \quad ; A.H, B.L, D.L, E.L, Y.H$$



Ground error -0.5  
Floating -1

No Key -0.5  
2 Input NORs -1  
Missing inverter -1 ea.  
Extra Inverter -0.5 or 1 depending on situation

2. For the circuit below, derive Y.L. DO NOT SIMPLIFY! (5 pt.)



$$Y.L = \overline{\overline{A} \overline{B}} + \overline{C + \overline{D}} + \overline{E} = \overline{A + B} + \overline{C D} + \overline{E} = (\overline{A} + \overline{B})(\overline{C} \overline{D}) + \overline{E}$$

3A. For the simple CPU in **Appendix A** and the following ROM contents, fill out the Cycle Table below for instructions stored in ROM. **Assume all registers are initially reset to zero.** Use 'X' to indicate a "don't care" condition and show **all answers in HEX.** (9 pts.)

Address =>	0	1	2	3	Show ALL VALUES IN HEX!		
Data (Hex) =>	F7	F1	F7	F2			
Cycle	State	Input3:0	IR	PC	Reg. A3:0	MSA1:0	MSC2:0
1 (reset)	0	7	0	0	0	01	xxx
2	1	7	7	0	0	01	xxx
3	3	1	7	1	0	01	xxx
4	0	1	7	1	0	01	xxx
5	1	1	1	1	0	01	xxx
6	2	7	1	2	0	00	xxx
7	0	2	1	3	7	01	xxx
8	1	2	2	3	7	11	110
		+1.5	+1.5	+1.5	+1	+1.5	+0.5

Handwritten notes:   
 - "Jump 1" and "Jump 2" with arrows pointing to MSA1:0 values.   
 - "Execution" bracket covering cycles 1-3.   
 - "Label #7" bracket covering cycles 4-5.   
 - "SAR" bracket covering cycles 7-8.   
 - Red circles around Reg. A3:0 values 0, 7, 7.   
 - Red circles around MSA1:0 values 00, 11.   
 - Red circles around MSC2:0 values 110.

3B. After the instructions above are executed, what should the contents of Register A:30 be in the 9<sup>th</sup> cycle?

3 HEX (1 pt.)

4. In your simple CPU used in **Lab #9**, what **HARDWARE modifications** and **new HARDWARE** is required to add the new instruction below? (4 pt.)

STAA Addr ;store register A to memory specified by the 8 bit address operand "Addr"

IR2:0 → IR3:0 } +1

PC3:0 → PC7:0

+1 MAR or Temp reg to hold operand (address)

+1 Addr mux to select between PC and MAR

+1 Tri-states (buffers) on output bus to connect to input bus

New control signals: R/W, MAR\_LD, Addr\_sel

5. See Appendix B to write the required program below. Write your code in the left column first and then wrap around to the right column for extra lines/space. (8 pt.)

$-127 + 127 = 0$  constant = 0x7F

```

+1 5 Ldaa #0; ; clear SUM
    staa 0x4080
+1 Ldab #0x7F ; constant
+1 Ldy #0x40801 ; input ptr
  
```

```

Top: [ Ldaa 0,y ; last value?
+1 [ BEQ Done
+1 [ Sum_ba ; = -127?
    [ BNE SKIP
  
```

```

↓
+1 Ldaa 0x4080; sum=
    INCA ; sum+=1
    staa 0x4080
Skip: +1 INY
+1 BNE Top
Done: BEQ Done
  
```

6. See the attached G-CPU program in Appendix C. Assuming that the code was placed in ROM starting at address 0x0. Answer the questions below:

6A. Based on the program execution, what size SRAM must exist in the G-CPU memory map?

RAM size C000-FFFF 16Kx8 (1 pt.)

6B. What is the -RAM\_CE Logic Equation?  $-RAM\_CE = A15 * A14$  (1 pt.)  
OR =  $A15 * A14 * D5.L$

6C. If the clock is 50 MHz, how many seconds does it take to execute the STAB 0xFFFF instruction? (1 pt.)

STAB 0xFFFF Execution Time =  $5 * \frac{1}{50 * 10^6} = 10^{-7}$  seconds 0.1  $\mu$ sec

6D. Assuming that the loop for the code in Appendix C is executed twice, show all SRAM addresses/data modified by the loop. (3 pt.)

ADDRESS (Hex)	DATA (Hex)
+1 { C000	FF +0.5
C001	03 ← FF ; 2nd Pass +0.5
C002	64 +0.5
<del>C003</del>	<del>65</del> +0.5
<del>C004</del>	<del>66</del> +0.5
<del>C005</del>	<del>67</del> +0.5
<del>C006</del>	<del>68</del> +0.5
<del>C007</del>	<del>69</del> +0.5
<del>C008</del>	<del>6A</del> +0.5
<del>C009</del>	<del>6B</del> +0.5
<del>C00A</del>	<del>6C</del> +0.5
<del>C00B</del>	<del>6D</del> +0.5
<del>C00C</del>	<del>6E</del> +0.5
<del>C00D</del>	<del>6F</del> +0.5
<del>C00E</del>	<del>70</del> +0.5
<del>C00F</del>	<del>71</del> +0.5
<del>C010</del>	<del>72</del> +0.5
<del>C011</del>	<del>73</del> +0.5
<del>C012</del>	<del>74</del> +0.5
<del>C013</del>	<del>75</del> +0.5
<del>C014</del>	<del>76</del> +0.5
<del>C015</del>	<del>77</del> +0.5
<del>C016</del>	<del>78</del> +0.5
<del>C017</del>	<del>79</del> +0.5
<del>C018</del>	<del>7A</del> +0.5
<del>C019</del>	<del>7B</del> +0.5
<del>C01A</del>	<del>7C</del> +0.5
<del>C01B</del>	<del>7D</del> +0.5
<del>C01C</del>	<del>7E</del> +0.5
<del>C01D</del>	<del>7F</del> +0.5
<del>C01E</del>	<del>80</del> +0.5
<del>C01F</del>	<del>81</del> +0.5
<del>C020</del>	<del>82</del> +0.5
<del>C021</del>	<del>83</del> +0.5
<del>C022</del>	<del>84</del> +0.5
<del>C023</del>	<del>85</del> +0.5
<del>C024</del>	<del>86</del> +0.5
<del>C025</del>	<del>87</del> +0.5
<del>C026</del>	<del>88</del> +0.5
<del>C027</del>	<del>89</del> +0.5
<del>C028</del>	<del>8A</del> +0.5
<del>C029</del>	<del>8B</del> +0.5
<del>C02A</del>	<del>8C</del> +0.5
<del>C02B</del>	<del>8D</del> +0.5
<del>C02C</del>	<del>8E</del> +0.5
<del>C02D</del>	<del>8F</del> +0.5
<del>C02E</del>	<del>90</del> +0.5
<del>C02F</del>	<del>91</del> +0.5
<del>C030</del>	<del>92</del> +0.5
<del>C031</del>	<del>93</del> +0.5
<del>C032</del>	<del>94</del> +0.5
<del>C033</del>	<del>95</del> +0.5
<del>C034</del>	<del>96</del> +0.5
<del>C035</del>	<del>97</del> +0.5
<del>C036</del>	<del>98</del> +0.5
<del>C037</del>	<del>99</del> +0.5
<del>C038</del>	<del>9A</del> +0.5
<del>C039</del>	<del>9B</del> +0.5
<del>C03A</del>	<del>9C</del> +0.5
<del>C03B</del>	<del>9D</del> +0.5
<del>C03C</del>	<del>9E</del> +0.5
<del>C03D</del>	<del>9F</del> +0.5
<del>C03E</del>	<del>A0</del> +0.5
<del>C03F</del>	<del>A1</del> +0.5
<del>C040</del>	<del>A2</del> +0.5
<del>C041</del>	<del>A3</del> +0.5
<del>C042</del>	<del>A4</del> +0.5
<del>C043</del>	<del>A5</del> +0.5
<del>C044</del>	<del>A6</del> +0.5
<del>C045</del>	<del>A7</del> +0.5
<del>C046</del>	<del>A8</del> +0.5
<del>C047</del>	<del>A9</del> +0.5
<del>C048</del>	<del>AA</del> +0.5
<del>C049</del>	<del>AB</del> +0.5
<del>C04A</del>	<del>AC</del> +0.5
<del>C04B</del>	<del>AD</del> +0.5
<del>C04C</del>	<del>AE</del> +0.5
<del>C04D</del>	<del>AF</del> +0.5
<del>C04E</del>	<del>B0</del> +0.5
<del>C04F</del>	<del>B1</del> +0.5
<del>C050</del>	<del>B2</del> +0.5
<del>C051</del>	<del>B3</del> +0.5
<del>C052</del>	<del>B4</del> +0.5
<del>C053</del>	<del>B5</del> +0.5
<del>C054</del>	<del>B6</del> +0.5
<del>C055</del>	<del>B7</del> +0.5
<del>C056</del>	<del>B8</del> +0.5
<del>C057</del>	<del>B9</del> +0.5
<del>C058</del>	<del>BA</del> +0.5
<del>C059</del>	<del>BB</del> +0.5
<del>C05A</del>	<del>BC</del> +0.5
<del>C05B</del>	<del>BD</del> +0.5
<del>C05C</del>	<del>BE</del> +0.5
<del>C05D</del>	<del>BF</del> +0.5
<del>C05E</del>	<del>C0</del> +0.5
<del>C05F</del>	<del>C1</del> +0.5
<del>C060</del>	<del>C2</del> +0.5
<del>C061</del>	<del>C3</del> +0.5
<del>C062</del>	<del>C4</del> +0.5
<del>C063</del>	<del>C5</del> +0.5
<del>C064</del>	<del>C6</del> +0.5
<del>C065</del>	<del>C7</del> +0.5
<del>C066</del>	<del>C8</del> +0.5
<del>C067</del>	<del>C9</del> +0.5
<del>C068</del>	<del>CA</del> +0.5
<del>C069</del>	<del>CB</del> +0.5
<del>C06A</del>	<del>CC</del> +0.5
<del>C06B</del>	<del>CD</del> +0.5
<del>C06C</del>	<del>CE</del> +0.5
<del>C06D</del>	<del>CF</del> +0.5
<del>C06E</del>	<del>D0</del> +0.5
<del>C06F</del>	<del>D1</del> +0.5
<del>C070</del>	<del>D2</del> +0.5
<del>C071</del>	<del>D3</del> +0.5
<del>C072</del>	<del>D4</del> +0.5
<del>C073</del>	<del>D5</del> +0.5
<del>C074</del>	<del>D6</del> +0.5
<del>C075</del>	<del>D7</del> +0.5
<del>C076</del>	<del>D8</del> +0.5
<del>C077</del>	<del>D9</del> +0.5
<del>C078</del>	<del>DA</del> +0.5
<del>C079</del>	<del>DB</del> +0.5
<del>C07A</del>	<del>DC</del> +0.5
<del>C07B</del>	<del>DD</del> +0.5
<del>C07C</del>	<del>DE</del> +0.5
<del>C07D</del>	<del>DF</del> +0.5
<del>C07E</del>	<del>E0</del> +0.5
<del>C07F</del>	<del>E1</del> +0.5
<del>C080</del>	<del>E2</del> +0.5
<del>C081</del>	<del>E3</del> +0.5
<del>C082</del>	<del>E4</del> +0.5
<del>C083</del>	<del>E5</del> +0.5
<del>C084</del>	<del>E6</del> +0.5
<del>C085</del>	<del>E7</del> +0.5
<del>C086</del>	<del>E8</del> +0.5
<del>C087</del>	<del>E9</del> +0.5
<del>C088</del>	<del>EA</del> +0.5
<del>C089</del>	<del>EB</del> +0.5
<del>C08A</del>	<del>EC</del> +0.5
<del>C08B</del>	<del>ED</del> +0.5
<del>C08C</del>	<del>EE</del> +0.5
<del>C08D</del>	<del>EF</del> +0.5
<del>C08E</del>	<del>F0</del> +0.5
<del>C08F</del>	<del>F1</del> +0.5
<del>C090</del>	<del>F2</del> +0.5
<del>C091</del>	<del>F3</del> +0.5
<del>C092</del>	<del>F4</del> +0.5
<del>C093</del>	<del>F5</del> +0.5
<del>C094</del>	<del>F6</del> +0.5
<del>C095</del>	<del>F7</del> +0.5
<del>C096</del>	<del>F8</del> +0.5
<del>C097</del>	<del>F9</del> +0.5
<del>C098</del>	<del>FA</del> +0.5
<del>C099</del>	<del>FB</del> +0.5
<del>C09A</del>	<del>FC</del> +0.5
<del>C09B</del>	<del>FD</del> +0.5
<del>C09C</del>	<del>FE</del> +0.5
<del>C09D</del>	<del>FF</del> +0.5

Best

6E. Fill out the cycle diagram below for execution of the LDAA 5,Y and STAB 0,X instructions in the **SECOND PASS** of the loop. Show ALL VALUES IN HEX. (12 pt.)

Cycle	State	Addr Bus	Data Bus	Reg Driving Addr Bus	Device Driving Data Bus	IR	PC	X Reg	Y Reg	R/-W	
1	0	0x12	0D	PC	Rom	0F	12	0001	0001	1	
2	5 1	0012	0D.5	PC	Rom	0D	12			1	
3	1E	0013	05	PC	Rom	0D	13			1	
4	1F	0006	64	Y BLK	Rom	0D	14			1	
5	0	0014	12	PC	Rom	0D	14			1	
6	5 1	0014	12.5	PC	Rom	12	14			1	
7	28	0015	<del>00</del> 00	PC	Rom	12	15			1	
8	29	0001	FF	X BLK	CPU	12	16	0001	0001	0	
		+1	+2	+3	+1	+1	+1	+0.5	+0.5	+1	
7. Extra Credit. How many times does the loop run in HEX? (1 pt.)										32	Hex

+1 each (64)  
 +1 each (FF)  
 0.5 each

Appendix B. Programming Problem #4

Write a G-CPU assembly program to sum the number of -127 values stored in RAM starting at address 0x4081 and **terminating with a zero** (last value in the array). The final "SUM" should be stored at 0x4080 and should be used as temporary storage for the SUM during your program execution. Use the Y register as your input pointer and register B to hold the required constant.

Assume the following new instructions are available:

DECA, decrement Register A (A-1 => A),

INCA, increment Register A (A+1 => A).

Appendix C. G-CPU Code for Problems 5 - 10:

```

ORG      0x0 ;this tells the assembler to start the program at addr zero
LDAB    #0xFE 0,1
STAB    0xFFFF 2,3,4
LDAB    #100 5,6
STAB    0xFFFE 7,8,9
LDY     #0x0 A,B,C
LDX     #0xC000 D,E,F
Top:    LDAB    3,Y } 1st pass y=0
        * LDAA   5,Y } 2nd pass =1
        * STAB   0,X } 1st pass X=C000
        STAA   1,X } =C001
        INY
        INX
        LDAB    0xFFFF
        LDAA    0xFFFE
        SUM_BA
        STAA    0xFFFE
        BNE    Top
End1:   BEQ    END1
    
```

	Addr	Data
	0	03
LDAB	1	FE
	2	07
STAB	3	FF
	4	FF
LDAB	5	03
	6	64
STAB	7	07
	8	FE
	9	FF
LDY	A	09
	B	00
	C	00
LDX	D	08
	E	00
	F	C0
LDAB	10	
	11	

FFFF [FE]

FFFE [64]

100 = 0x64

16 \* 5 = 80  
16 \* 6 = 96

0110 0100  
+ 1111 1110  
-----  
01100010 62  
2nd Pass → 60