1. Using only 2 input OR gates and Inverters, implement the following logic equation (do not simplify).

Your final answer should only contain 2 input OR Gates and Inverters Only! (6 pt);

\[ Y = (A + C) \overline{(A + B)} (B + C) \]

Assume: Y.H, A.H, B.L and C.L

\[ \overline{0} = \text{领悟} \]

\[ \overline{D} = \text{领悟} \]
2. For the circuit below, determine the logic equation for Y.L. Also show all intermediate terms as their low true equivalents. I.e. After every gate, write the low true version for the logic term. (6 pt.)

\[ Y.L = \frac{A \cdot B \cdot C}{\overline{D} \cdot E \cdot F} \cdot (G) \]

3. Assume that several bytes are stored in memory starting at 2000 Hex. The bytes are assumed to be in 2's complement (signed number) format. Write a short G-CPU program to replace each positive number with a 1 and replace each negative number with -1. I.e. Read the value, if it is positive, replace it with 1. Else if it is negative, replace with -1. Continue on through memory until a zero is read in. This is the terminating value. If the value read is zero, end your code in an infinite loop. You may assume you also have G-CPU instructions: DECA, INCA and BRA unconditional. The Best Code = The Most Points! (8 pt)

Write your answer in the leftmost column lines. If you need more room, wrap around to the right column lines.

```
LDX #$2000

Top
LDAA 0
BEQ DONE
BPL Positive
LDAA #$FF
BN $0000 Negative

Positive
LDAA #1
STAA 0
INX
BRA Top

Negative
STAA 0
INX 1
BRA Top

DONE BRA DONE
```
4. See the attached G-CPU program in Appendix A at the end of our quiz. Assume the code was placed in EPROM starting at $6000$. Also assume there is SRAM starting at $A000$. Answer the questions below:

4A. What is the effective address for the first LDY #$A000$ instruction? $6000$ Hex (1 pt.)

4B. What is the effective address for STAB 1,Y the first time the loop is executed? $A001$ Hex (1 pt.)

4C. What are the values stored at $6008$ and $6009$? $03, 01$ Hex (1 pt.)

4D. If the clock is $2$ MHz, how many seconds does it take to execute the SUM_BA instruction? (1 pt.)

\[ T = 0.5 \text{msec}, \quad 2 \text{states} \Rightarrow 10^{-6} \text{sec} \]

4E. Show all SRAM addresses and associated data modified by the code above. (5 pt.)

\[
\begin{align*}
A000: [PA] \quad & A001: [02] \quad A006: [A0] \quad A007: [00] \\
A002: [00] \quad & A003: [08] \quad A004: [09] \quad A005: [60] \\
& A008: [01] \quad A009: [03]
\end{align*}
\]

4F. Fill out the cycle diagram below for the SUM_BA & BEQ instructions for the first pass of the loop. (8 pt.)

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Addr Bus</th>
<th>Data Bus</th>
<th>IR</th>
<th>PC</th>
<th>R/W</th>
<th>X Reg</th>
<th>Y Reg</th>
<th>Reg Driving Addr Bus</th>
<th>Device Driving Data Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>600A</td>
<td>14</td>
<td>03</td>
<td>600A</td>
<td>1</td>
<td>6000</td>
<td>A000</td>
<td>PC</td>
<td>EPROM</td>
</tr>
<tr>
<td>2</td>
<td>600A</td>
<td>14</td>
<td>14</td>
<td>600A</td>
<td>1</td>
<td></td>
<td></td>
<td>PC</td>
<td>EPROM</td>
</tr>
<tr>
<td>3</td>
<td>600B</td>
<td>20</td>
<td>14</td>
<td>600B</td>
<td>1</td>
<td></td>
<td></td>
<td>PC</td>
<td>EPROM</td>
</tr>
<tr>
<td>4</td>
<td>600B</td>
<td>20</td>
<td>20</td>
<td>600B</td>
<td>1</td>
<td></td>
<td></td>
<td>PC</td>
<td>EPROM</td>
</tr>
<tr>
<td>5</td>
<td>600C</td>
<td>20</td>
<td>20</td>
<td>600C</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

4G. For the previous problem, assume the size of the EPROM and SRAM are $8K \times 8$ each. What is the range of each memory in the CPU memory map?

EPROM Range $6000 - 7FFF$ (1 pt.)

SRAM Range $A000 - BFFF$ (1 pt.)

4H. What is the logic decode equation required for each device? Assume the usual CPU control bus signals including a /DS (data strobe) signal.

\[ /\text{EPROM\_CE} = /\text{DS} \cdot /R/W \cdot [\overline{A15} \cdot A14 \cdot A13] \quad (1 \text{ pt.}) \]

\[ /\text{SRAM\_CE} = /\text{DS} \cdot A15 \cdot A14 \cdot A13 \quad (1 \text{ pt.}) \]
5. We would like to add index addressing to our simple CPU in Lab #9. The new register for index addressing will be the W register. There will be no displacement register but instead we will implement instructions that read/write data using W as a pointer and automatically inc or dec W. See Appendix B.

The new controller ASM signals required for the W block therefore are: INC_W, DEC_W and LD_W
Note: Assume only one or none of these signals is true in a particular state in the controller ASM and that they are all high true signals.

5A. Show the required hardware to create the W register block. Also show any additional hardware needed to support index addressing in your Lab #9 CPU and make sure you use the new control signals shown above. Label all wires using bus notation and also label all devices & signals employed in the design. You can use any generic device used in lab this semester but do not use any 74XX parts. (6 pt.)

5B. Show the changes/states that must be added to your ASM controller flowchart to implement LDW #data and STAB W+. Show only the signals that are true in your ASM state. Pick two new/open opcodes. (3 pt.)
Appendix A. G-CPU Code for Problem #4:

```
ORG $6000
LDAA #$0FF
LDX #$6000
LDY #$0000
T1
LDAB  #1
SUM_BA  BEQ  END1

LDAB  0,X
STAB  1,Y
LDAB  1,X
STAB  0,Y
INX
INX
INY
INY
BRA  T1

END1 BRA  END1
```

```
6000  02  2F  LDA A  1  3F  A  5FF
6002  08  7D  LDX  2  00  6000
6005  09  7D  LDY  3  00  0000
6007  01  00  6008  03  7D  LDAB  4  1  T1
6009  01  00  600A  14  7D  SUM_BA
600B  20  7D  600C  18  7D  BEQ  END1
600D  0E  7D  600E  00  7D  600F
6010  13  7D  6011  0E  7D  6012  01  00  6013
6014  13  7D  6015  30  6016  30
```

Appendix B. W Register Instructions:

- **LDAA W+**: Use W as a pointer to read data into A and then increment W once.
- **LDAA W-**: Use W as a pointer to read data into A and then decrement W once.
- **LDAB W+**: Use W as a pointer to read data into B and then increment W once.
- **LDAB W-**: Use W as a pointer to read data into B and then decrement W once.
- **LDW #data**: Load immediate data into W.

```
LDAA W+ ; use W as a pointer to read data into A and then increment W once.
LDAA W- ; use W as a pointer to read data into A and then decrement W once.
LDAB W+ ; use W as a pointer to read data into B and then increment W once.
LDAB W- ; use W as a pointer to read data into B and then decrement W once.
LDW #data ; load immediate data into W.
```