1. Using only 2 input NOR gates, implement the following logic equation (do not simplify).

Your final answer should only contain 2 input NOR Gates Only!  (4 pt);

\[ Y = (A \cdot B)(C+D) \]

; Y.H, A.L, B.H, C.L, D.L

key:

\[ \Rightarrow \theta = \theta \]

\[ \Rightarrow \theta = \theta \]

\[ \Rightarrow \theta = \theta \]

\[ \Rightarrow \theta = \theta \]

\[ \Rightarrow \theta = \theta \]

\[ \Rightarrow \theta = \theta \]
2. Assume that a vector of 8 bit 2's complement numbers are stored in memory starting at 2000 Hex. The number of elements in the vector is less than 256 and the last element is set to zero to denote the end of vector. Note: Other than the end of vector element, no other elements are zero.

Write a G-CPU program to count the number of negative numbers in the vector and place this sum at memory location 4000 Hex. You may assume that the value at 4000 Hex is initially zero. (10 pt.)

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

1  Ldx #$2000  #0x2000  also ok.

2  Top:  Ldaa 0,x  j get element
        INX  j inc ptr
        BEQ Done  j check if zero
        BP Top  j check if neg/pos.

3  Ldaa $4000  j neg no., inc memory counter
4  Ldad #1
5  sum-ba
6  staa $4000
7  BNE Top

Done  BEQ DONE  j not required, but shows you know
       where the program ends

3. Refer to the attached G-CPU program on the last page of this quiz. Assume the code was placed in ROM starting at $8000. Also assume there is RAM starting at $9000. Answer the questions below:

3A. What are the byte values in ROM memory for addresses $8000 to $8009? (5 pt.)

8000 [ 08 ]  8001 [ 00 ]  8002 [ 90 ]  8003 [ 09 ]

0.5 each

3B. If the clock is 10 MHz, how many seconds does it take to execute the first LDX #$9000 instruction? (1 pt.)

4 cycles  10 MHz  ⇒  T = 0.1 μsec

0.4 μsec

Page 2  Page Score =
3C. What does this program perform? The most precise answer will receive the most points. (2 pt.)

Counts number of zeros in a 36 element vector starting at address $9800$ ($9000$)

3D. Fill out the cycle diagram below for the second time the following two lines are executed in the program given in Appendix A. (8 pt.)

this instruction starts at address $8015$ in ROM memory

<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>N 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>8015</td>
<td>01</td>
<td>03</td>
<td>8015</td>
<td>1</td>
<td>0</td>
<td>9800</td>
<td>9800</td>
<td>Rom</td>
</tr>
<tr>
<td>2</td>
<td>8015</td>
<td>14</td>
<td>14</td>
<td>8015</td>
<td>1</td>
<td>B</td>
<td>9800</td>
<td>9800</td>
<td>Rom</td>
</tr>
<tr>
<td>3</td>
<td>8016</td>
<td>11</td>
<td>14</td>
<td>8016</td>
<td>1</td>
<td>2</td>
<td>9800</td>
<td>9800</td>
<td>Rom</td>
</tr>
<tr>
<td>4</td>
<td>8016</td>
<td>11</td>
<td>1</td>
<td>8016</td>
<td>1</td>
<td>2</td>
<td>9800</td>
<td>9800</td>
<td>Rom</td>
</tr>
<tr>
<td>5</td>
<td>8017</td>
<td>01</td>
<td>1</td>
<td>8017</td>
<td>1</td>
<td>2</td>
<td>9800</td>
<td>9800</td>
<td>Rom</td>
</tr>
<tr>
<td>6</td>
<td>9801</td>
<td>02</td>
<td>1</td>
<td>8018</td>
<td>0</td>
<td>2</td>
<td>9800</td>
<td>9800</td>
<td>Rom</td>
</tr>
</tbody>
</table>

4. We would like to add three new branch instructions to the G-CPU. These are

BRA Addr ; unconditional branch, always branch
BC Addr ; branch if there is a carry after a sum operation
BNC Addr ; branch if there is no carry after a sum operation

4A. What new hardware or signals must be added to the G-CPU to create these three new instructions? (2 pt.)

the PLA adder carry out signal (wire) needs to go to the controller, i.e. new input for controller

4B. Show what must be added to the G-CPU Controller ASM below for these three new instructions. (5 pt.)

BRA Addr 100000
BC Addr 100001
BNC Addr 100010

Page Score =
5. For the following problem, assume that you have the hardware you used in Lab #9 and the ASM Controller flow chart on the next page. Fill in the timing diagram below assume that the first six nibbles in ROM memory are 1, A, 4, 0, 7, B. i.e. ROM Contents: Addr0 [1], Addr1 [A], Addr2 [4], Addr3 [4], Addr4 [7], Addr5 [B]. (10 pt.)

<table>
<thead>
<tr>
<th>CLK</th>
<th>State</th>
<th>PC3:0</th>
<th>IN3:0</th>
<th>IR2:0</th>
<th>*A3:0</th>
<th>10100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 or X</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>A</td>
<td>1</td>
<td>0 or X</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Fill in all the blank area above! *A3:0 = Register A3:0 in the ALU

6. We would like to create a new instruction for the G-CPU called STX addr which stores the contents of X into memory starting at the address specified by the operand addr. Show the required hardware modification to the X Block (2 pt.) below and the changes required to the C-CPU ASM Controller Flow Chart (2 pt.)

**Hardware Modifications:**

- EXL = enable X Block Low (controller output)
- EXH = enable X Block High (controller output)

**ASM Flow Chart Modifications:**

- AddrL → MARL INC. PC
- AddrH → MARH INC. PC
- EXL
- R/W = 0, INC MAR
- EXH
- R/W = 0
G-CPU Code for Problem #3 and ASM Controller Flow Chart for Problem #5

```
ORG   $8000
LDX   #$9000 ;ptr to input data
LDY   #$8000 ;ptr to counter and result
LDAA  #$24  ;initial counter value
STAA  0,Y   
LDA   #0    
STAA  1,Y   

T1    
LDAA  0,X   
INX    
BNE   SKIP1
LDA   1,Y   
LDA   #1   
SUM_BA
STAA  1,Y   

SKIP1  
LDA   #0   
LDA   0,Y   
SUM_BA
STAA  #FF
STAA  0,Y   
BNE   T1   

END1  
BEQ    END1
```

---

**Flow Chart Diagram**

- **State 0**: IR.LD
- **State 1**: PC.INC
- **State 2**: A/2 => A, A+B => A, A*2 => A
- **State 3**: In3:0 => PC

**States and Operations**

- **TAB**: 000, 001, 010, 011, 100, 111
- **LDAA**: #data
- **SAR**: 010
- **ABA**: 011
- **SAL**: 100
- **JMP addr**: 111

**Operations**

- **A => B**
- **State 2**: In3:0 => A, PC.INC
- **State 3**: In3:0 => PC

---

*Page 5*