1. Implement the logic equation below using 2 Input NAND Gates only. (6 pt.)

\[ Y = (A \cdot (B+C)) \cdot (D+E) \quad ; \text{A.L, B.H, C.L, D.H, E.L, Y.L} \]

2. The G GPU has been redesigned to initially boot from 8000 Hex. i.e. Fetch it's first instruction from 8000H in memory. Assuming that there is a low true Reset signal, explain what needs to be modified in the current design to facilitate booting from 8000 Hex. (2 pt.) Best Answer = Most Points!

Use Asynchronous set & CLR on FFs that make-up the PC

Specifically, CLR PC14:0 upon reset

and set PC15 upon reset.

\[ PC + 1 \quad \text{set/CLR +1} \]
3. A student has interfaced a \( 2^{10} \times 8 \) ROM with the G-CPU starting at address 8000 Hex. The student also has interfaced and placed \( 4^{12} \times 8 \) RAM in the highest \( 4^1 \) of the G-CPU memory map. What is the memory ranges for the ROM and RAM devices? (2 pt.)

**ROM Memory Range**

\[ 2K \\times 2^{10} = 2^{12} \]

**RAM Memory Range**

\[ 4K \\times 2^{12} = 2^{12} \]

8000 Hex to 2K

\[ \text{correct} \]

(Hex) for 5K

\[ \text{FOO0} \]

(Hex) to FFFF Hex

---

4. A vector stored in RAM consists of signed numbers that are one byte in length. Write the code to convert all the negative numbers to positive numbers. Assume B contains the length of the vector and Y contains the starting address of the vector. Also, if you need a counter in memory, use address FFFF Hex. (10 pt.)

```
STAB $FFFF ;save counter in memory

Top:
Ldab 0,Y ;get value

BP SKIP

Ldab #$FF ;B = -1
Sum -BA ;neg, no -1

Com A ;complement A

Staa 0,Y ;store pos, no.

INY ;J INC PTR

Skip: Ldab $FFFF ;get count

Sum BA ;count = count - 1

BEQ DONE ;check if zero

STAA $FFFF ;save counter

BNE Top
```

DONE: Ldab #0 ;loop

BEQ DONE

---

Page Score =
For the next set of questions, consult the G-CPU code in Appendix A and assume the following initial conditions: \( A = 5, X = \text{FF00}, \) Memory location \( \text{FF00} = \text{SO0}, \text{FF01} = \text{S11}, \text{FF02} = \text{S22}, \ldots, \text{FF08} = \text{S88}. \)

5. What is the effective address for the STAB instruction at \$8006? \( \text{FFE} \) Hex (2 pt.)

6. What is the effective address for the LDAB instruction at \$8009? \( \text{8008A} \) Hex (2 pt.)

7. How many times is the LDAA 0,X at address \$801C executed at run time? \( 2 \) Hex (2 pt.)

8. What values in SRAM memory below when the PC is fetching \$802? (3 pt.)

\[
\begin{align*}
0.5 & \text{FF00} = 44 \quad \text{Hex} & 0.5 & \text{FF01} = 33 \quad \text{Hex} & 0.5 & \text{FF02} = 22 \quad \text{Hex} \\
0.5 & \text{FF03} = 11 \quad \text{Hex} & 0.5 & \text{FF04} = 00 \quad \text{Hex} & 0.5 & \text{FF05} = 55 \quad \text{Hex}
\end{align*}
\]

9. What is the function of this short program?

\[ \text{Reverse the elements in the vector.} \]

\[ \text{i.e. 1st} \leftrightarrow \text{last}, \text{2nd} \leftrightarrow \text{2nd from last, etc.} \] (2 pt.)

10. Using only \( 2:1 \) decoders, flip-flops, \( 2:1 \) muxes, single bit full adders and any other elementary digital components, design the hardware required to implement the SUM_AY instruction. Label all buses and explain any new signals you'll need in the ALU, Controller, IR and/or Address Mux. Best design = Most points. (4 pt.)

\[ \text{Max } "3" = \text{SUM_AY, } "2" = \text{load data, } "1" = \text{INC_Y, } "0" = \text{hold} \]

\[ \text{P1} = \text{YINC (YLDL} \oplus \text{YAY)} \]
\[ \text{P2} = \text{YLDL (YINC} \oplus \text{YAY)} \]
\[ \text{P3} = \overline{\text{YINC (YLDL} \oplus \text{YAY)} \]
\[ \text{P4} = \overline{\text{YLDL (YINC} \oplus \text{YAY)} \]

Page Score =
11. For the G CPU code shown in Appendix A, fill in the cycle table below for the first pass for the lines of code shown below. Use the initial conditions shown on the previous page and show all answers in Hex. (9 pt.)

```
LDAB $FFFE ; first pass of this instruction
DECA

; assume this takes 2 cycles to execute

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Controller State</th>
<th>Addr Bus</th>
<th>Data Bus</th>
<th>IR</th>
<th>R/W</th>
<th>B Reg</th>
<th>Dev driving Addr Bus</th>
<th>Dev Driving Data Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>8012</td>
<td>05</td>
<td>04</td>
<td>1</td>
<td>01</td>
<td>PC</td>
<td>Rom</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8012</td>
<td>05</td>
<td>05</td>
<td>1</td>
<td>01</td>
<td>PC</td>
<td>Rom</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8013</td>
<td>FE</td>
<td>05</td>
<td>1</td>
<td>01</td>
<td>PC</td>
<td>Rom</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8014</td>
<td>FF</td>
<td>05</td>
<td>1</td>
<td>01</td>
<td>PC</td>
<td>Rom</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>FFFE</td>
<td>FD</td>
<td>05</td>
<td>1</td>
<td>01</td>
<td>MAR</td>
<td>RAM</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>8015</td>
<td>New Opcode</td>
<td>05</td>
<td>1</td>
<td>FD</td>
<td>PC</td>
<td>Rom</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>8015</td>
<td>New Opcode</td>
<td>05</td>
<td>1</td>
<td>FD</td>
<td>PC</td>
<td>Rom</td>
</tr>
</tbody>
</table>
```

New Opcode = 24 to 2F or 32 to 3F

12. The G CPU Next State Table and Controller can be found in Appendix B. The Controller was created via an 8 x 32 ROM. For the very first and last cycles of the LDAB $FFFE instruction above, show the addresses and corresponding data that need to be programmed in the ROM. (4 pt.)

**First Cycle Executed**

ROM Addresses = 0 - 00FF

Data = 05 81 03

**Last Cycle Executed**

ROM Addresses = 0900 - 09FF

Data = 03 07 FC

14. For the new SUM_AY instruction in Appendix A, show below what needs to be added to the Controller ASM Flow Chart assuming that this instruction will follow the INY instruction in the ASM Chart. (2 pt.)
Appendix B. G-CPU Controller & Next State Table:

Controller Input & Output

Controller Logic

ASM State Generation

Controller Outputs

Debug Purposes

Controller Inputs (State, Flags, Instruction)
Appendix A. G-CPU Code (address of instruction in memory shown on left):

Register $A = \text{Vector Length}$, Register $X = \text{Starting Address of Vector}$

\[ x = \$FF00 \quad A = 5 \]

8000 \( \checkmark \) Compute:
8001 \( \checkmark \) ORG \$8000
8002 \( \checkmark \) TXY \( y = FF00 \);
8003 \( \checkmark \) new instruction transfer \( X \Rightarrow Y \)
8004 \( \checkmark \) SUM_AY \( y = FF05 \);
8005 \( \checkmark \) new instruction \( A + Y \Rightarrow Y \)
8006, 7, 8 \( \checkmark \) DEY \( y = FF04 \);
8009, A \( \checkmark \) new instruction Decrement \( Y \)
800B \( \checkmark \) SHFA_R \( A = 2 \)
800C, D, E \( \checkmark \) TAB \( B = 2 \)
800F, 10, 11 \( \checkmark \) COMB \( B = FD \)
8012, 13, 14 \( \checkmark \) STAB \$FFFE \[ FFFE \ldots \]
8015 \( \checkmark \) 2nd pass
8016 \( \checkmark \) LDAB \$FFFE \[ FFFF \ldots \]
8017, 18 \( \checkmark \) 2nd pass
8019 \( \checkmark \) DECA \( A = 2 \) \( A = 1 \);
801C, D \( \checkmark \) new instruction Decrement \( A \)
801E, 1F \( \checkmark \) TAB \( B = 2 \) \( B = 2 \)
8020, 21 \( \checkmark \) BEQ \( A = 3 \) \( A = 2 \)
8022 \( \checkmark \) Done
8023 \( \checkmark \) STAA \$FFFF \[ FFFF \ldots \]
8024 \( \checkmark \) 2nd pass
8025 \( \checkmark \) LDAA 0, X \( A = 0 \)
8026 \( \checkmark \) 2nd pass
8027 \( \checkmark \) LDAB 0, Y \( B = 44 \)
8028 \( \checkmark \) STAA 0, Y \[ FF04 \ldots \]
8029 \( \checkmark \) 2nd pass
802A \( \checkmark \) STAB 0, X \[ FF06 \ldots \]
802B \( \checkmark \) INX \( x = FF01 \)
802C \( \checkmark \) 2nd pass
802D \( \checkmark \) DEY \( y = FF03 \);
802E \( \checkmark \) new instruction, Decrement \( Y \)
802F \( \checkmark \) BRA Loop \( \ldots \)
8030 \( \checkmark \) 2nd pass
8031 \( \checkmark \) BRA Done \( \ldots \)
8032 \( \checkmark \) 2nd pass
8033 \( \checkmark \) BRA Done \( \ldots \)
### G-CPU Controller Next State Table

<table>
<thead>
<tr>
<th>Present State</th>
<th>Opcode</th>
<th>Flag</th>
<th>Next State</th>
<th>Max Select</th>
<th>Control</th>
<th>REG INC</th>
<th>ADDR SEL</th>
<th>PC MAR</th>
<th>MAR LD</th>
<th>X,Y LD</th>
<th>X,Y Loading</th>
<th>Disp Regs</th>
<th>Present State Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000</td>
<td>000000</td>
<td>XX</td>
<td>000001</td>
<td>01</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 generic instruction fetch</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000001</td>
<td>10</td>
<td>01</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 Transfer A to B (TBA)</td>
</tr>
<tr>
<td>000001</td>
<td>000001</td>
<td>XX</td>
<td>000001</td>
<td>10</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA #data, state 1</td>
</tr>
<tr>
<td>000001</td>
<td>000011</td>
<td>XX</td>
<td>000011</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA #data, state 2</td>
</tr>
<tr>
<td>000001</td>
<td>000011</td>
<td>XX</td>
<td>000011</td>
<td>00</td>
<td>00</td>
<td>0001</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA #data, state 3</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 1</td>
</tr>
<tr>
<td>000001</td>
<td>000001</td>
<td>XX</td>
<td>000001</td>
<td>10</td>
<td>01</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 2</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 3</td>
</tr>
<tr>
<td>000001</td>
<td>000011</td>
<td>XX</td>
<td>000011</td>
<td>00</td>
<td>00</td>
<td>0001</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 4</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 5</td>
</tr>
<tr>
<td>000001</td>
<td>000001</td>
<td>XX</td>
<td>000001</td>
<td>10</td>
<td>01</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 6</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 7</td>
</tr>
<tr>
<td>000001</td>
<td>000001</td>
<td>XX</td>
<td>000001</td>
<td>10</td>
<td>01</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 8</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 LDA addr, state 9</td>
</tr>
<tr>
<td>000001</td>
<td>000011</td>
<td>XX</td>
<td>000011</td>
<td>00</td>
<td>00</td>
<td>0001</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 1</td>
</tr>
<tr>
<td>000001</td>
<td>000011</td>
<td>XX</td>
<td>000011</td>
<td>00</td>
<td>00</td>
<td>0001</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 2</td>
</tr>
<tr>
<td>000001</td>
<td>000011</td>
<td>XX</td>
<td>000011</td>
<td>00</td>
<td>00</td>
<td>0001</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 3</td>
</tr>
<tr>
<td>000001</td>
<td>000011</td>
<td>XX</td>
<td>000011</td>
<td>00</td>
<td>00</td>
<td>0001</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 4</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 5</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 6</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 7</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 8</td>
</tr>
<tr>
<td>000001</td>
<td>000000</td>
<td>XX</td>
<td>000000</td>
<td>00</td>
<td>10</td>
<td>0000</td>
<td>0</td>
<td>1</td>
<td>1000</td>
<td>0000</td>
<td>00</td>
<td>00</td>
<td>00 STAA addr, state 9</td>
</tr>
</tbody>
</table>

---

**Addr of Upper 3 bytes**

\[
\text{DS}0483D2 \text{ Di Do msa msb msc} \text{ Adde} \text{ A} \text{ Di Do pc, vol} \text{ Adda} \text{ 3 stmts} = 058300 \text{ Hex}
\]

\[
\text{LL LL LL HH H H H H H H} = 058103 \text{ Hex}
\]

**Addr 900 Hex**

\[
\text{mbe mbe mbe TLD TldTld} \text{ Adde} \text{ A} \text{ A pc, vol} \text{ Adda} \text{ 3 stmts} = 010400 \text{ Hex}
\]

---

**Page 7**

Page Score =