DESIGN REQUIREMENTS

1. Create four parallel scalar functional units with vector registers $V_i$, $V_j$, $V_k$ created from $R_i$, $R_j$, $R_k$ of the units.

2. Add a 32-bit adder to add the two 16-bit product results from each unit after a vector multiply, as part of a matrix row x column instruction.

3. Put results into a 32 bit scalar accumulator, then into a neuron transfer function.

4. Create an embedded code stack, controlled by a program counter via a master control unit with a final state machine that implements the simple pipeline of fetch, decode, execute, and write-back; plus any special states.

5. Then embed a carefully crafted assembly language code segment to demonstrate the functionality of your instruction set and circuitry in the minimal amount of time that you can defend as providing comprehensive testing of all scalar, vector, matrix, and neuron machine instructions and hardware.

MACHINE INSTRUCTION SET

SCALAR ARITHMETIC ADDITION

00h (OP-CODE = 00000000) $R_i + \text{counter#1}$ → $R_k$
01h (OP-CODE = 00000001) $R_i + \text{counter#2}$ → $R_k$
02h (OP-CODE = 00000010) $R_i + R_j$ → $R_k$
03h (OP-CODE = 00000011) $\text{counter#1} + \text{counter#2}$ → $R_k$

SCALAR ARITHMETIC SUBTRACTION

04h to 07h (OP-CODE = 000001XX) Reserved for subtraction instructions

SCALAR ARITHMETIC MULTIPLICATION

08h (OP-CODE = 00001000) $R_i \times \text{counter#1}$ → $R_k$, $\text{overflow}$ → $R_k+1$ (wrap to $R_0$)
09h (OP-CODE = 00001001) $R_i \times \text{counter#2}$ → $R_k$, $\text{overflow}$ → $R_k+1$ (wrap to $R_0$)
0Ah (OP-CODE = 00001010) $R_i \times R_j$ → $R_k$, $\text{overflow}$ → $R_k+1$ (wrap to $R_0$)
0Bh (OP-CODE = 00001011) $\text{counter#1} \times \text{counter#2}$ → $R_k$, $\text{overflow}$ → $R_k+1$ (wrap to $R_0$)

SCALAR ARITHMETIC DIVISION

0Ch to 0Fh (OP-CODE = 000011XX) Reserved for division instructions

SCALAR ARITHMETIC COMPARISON

10h (OP-CODE = 00010000) Compare $R_i$ with $\text{counter#1}$ → $R_k$
11h (OP-CODE = 00010001) Compare $R_i$ with $\text{counter#2}$ → $R_k$
12h (OP-CODE = 00010010) Compare $R_i$ with $R_j$ → $R_k$
13h (OP-CODE = 00010011) Compare $\text{counters}$ → $R_k$

SCALAR LOGICAL AND

20h (OP-CODE = 00100000) $R_i \text{ AND} \text{counter#1}$ → $R_k$
21h (OP-CODE = 00100001) $R_i \text{ AND} \text{counter#2}$ → $R_k$
22h (OP-CODE = 00100010) $R_i \text{ AND} R_j$ → $R_k$
23h (OP-CODE = 00100011) $\text{AND counters}$ → $R_k$

SCALAR LOGICAL OR

24h (OP-CODE = 00100100) $R_i \text{ OR} \text{counter#1}$ → $R_k$
25h (OP-CODE = 00100101) $R_i \text{ OR} \text{counter#2}$ → $R_k$
26h (OP-CODE = 00100110) $R_i \text{ OR} R_j$ → $R_k$
27h (OP-CODE = 00100111) $\text{OR counters}$ → $R_k$

CLEAR

30h (OP-CODE = 00110000) Clear $R_i$

MAC (Multiply, Accumulate). Accumulator = Accumulator + ($R_i \times R_j$) considering overflow also

40h (OP-CODE = 01000000) Step 1: Accumulator → $R_k+3$ (wrap to $R_0+$)
Step 2: $\text{Overflow}$ → $R_k+4$ (wrap to $R_0+$)
Step 3: $R_i \times R_j$ → $R_k$, $\text{overflow}$ → $R_k+1$ (wrap to $R_0$)
Step 4: ($R_k+3$)+$R_k$ → $R_k$
Step 5: ($R_k+4$)+($R_k+1$)+$\text{Carry}$ → $R_k+1$

VECTOR-ARRAY / MATRIX & NEURON INSTRUCTIONS

$V_i$, $V_j$, and $V_k$ are created from $R_i$, $R_j$, and $R_k$ of the four parallel functional units

VECTOR ARITHMETIC ADDITION

82h (OP-CODE = 10000010) $V_i + V_j$ → $V_k$

VECTOR ARITHMETIC SUBTRACTION

84h to 87h (OP-CODE = 100001XX) Reserved for subtraction instructions

VECTOR ARITHMETIC MULTIPLICATION

0Ah (OP-CODE = 10001010) $V_i \times V_j$ → $V_k$, $\text{overflow}$ → $V_k+1$ (wrap to $R_0$)

VECTOR ARITHMETIC DIVISION

8Ch to 8Fh (OP-CODE = 100011XX) Reserved for division instructions

MATRIX ROW x COLUMN (i.e., Dot-Product)

C0h (OP-CODE = 11000000) $V_i \times V_j$ → $V_k$, $\text{overflow}$ → $V_k+1$ (wrap to $R_0$)

$V_k(1) + V_k(2) + V_k(3) + V_k(4)$

32-Bit Scalar Accumulator

NEURON TRANSFER FUNCTION

E0h (OP-CODE = 11100000) $V_i \times V_j$ → $V_k$, $\text{overflow}$ → $V_k+1$ (wrap to $R_0$)

$V_k(1) + V_k(2) + V_k(3) + V_k(4)$ → 32-Bit Scalar Accumulator → Neuron Transfer Function