Part 1: Floating Point Representation

Encode the following four numbers/expressions into Single Precision IEEE 754 Floating-Point format. Be sure to round as needed. Partition the bits to show each field of the Floating-Point format.

- A.) 15.2
  \[0|10000010|11100110011001100110011|\]
- B.) -726.15
  \[1|1001000|0110101100100110011010|\]
- C.) \((2^{24} + 3)\)
  \[0|10010111|0000000000000000000000010|\]
- D.) \(+\infty\)
  \[0|11111111|00000000000000000000000000000000|\]

Part 2: Code on different types of architecture

Using the following function:

\[A = \frac{(B + C)^3 \times (D - C)}{B \times C \times D}\]

Code this function for:

- A generic stack architecture
  - MULTiply, DIVide (first operand popped is the denominator, ADDition, SUBtraction are the available arithmetic operations, DUPlicate produces a copy of the top item on the stack

- A generic accumulator architecture
  - MULTiply, DIVide, ADDition, SUBtraction are the available arithmetic operations

- A generic register-register architecture (not MIPS).
  - MULTiply, DIVide, ADDition, SUBtraction are the available arithmetic operations
<table>
<thead>
<tr>
<th>Instruction #</th>
<th>Stack</th>
<th>stack comments</th>
<th>Accumulator</th>
<th>Register-Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>push B</td>
<td>B</td>
<td>load B</td>
<td>load R0, B</td>
</tr>
<tr>
<td>2</td>
<td>push C</td>
<td>C, B</td>
<td>add C</td>
<td>load R1, C</td>
</tr>
<tr>
<td>3</td>
<td>add</td>
<td>B+C</td>
<td>store A</td>
<td>load R2, D</td>
</tr>
<tr>
<td>4</td>
<td>dup</td>
<td>(B+C), (B+C)</td>
<td>mult A</td>
<td>add R3, R0, R1</td>
</tr>
<tr>
<td>5</td>
<td>dup</td>
<td>(B+C), (B+C), (B+C)</td>
<td>mult A</td>
<td>mult R4, R3, R3</td>
</tr>
<tr>
<td>6</td>
<td>mult</td>
<td>(B+C)^2, (B+C)</td>
<td>store A</td>
<td>mult R4, R4, R3</td>
</tr>
<tr>
<td>7</td>
<td>mult</td>
<td>(B+C)^3</td>
<td>load D</td>
<td>sub R5, R2, R1</td>
</tr>
<tr>
<td>8</td>
<td>push D</td>
<td>D, (B+C)^3</td>
<td>sub C</td>
<td>mult R4, R4, R5</td>
</tr>
<tr>
<td>9</td>
<td>push C</td>
<td>C, D, (B+C)^3</td>
<td>mult A</td>
<td>mult R5, R0, R1</td>
</tr>
<tr>
<td>10</td>
<td>sub</td>
<td>(D-C), (B+C)^3</td>
<td>div B</td>
<td>mult R5, R5, R2</td>
</tr>
<tr>
<td>11</td>
<td>mult</td>
<td>(B+C)^3 * (D-C)</td>
<td>div C</td>
<td>div R6, R4, R5</td>
</tr>
<tr>
<td>12</td>
<td>push B</td>
<td>B, (B+C)^3 * (D-C)</td>
<td>div D</td>
<td>store A, R6</td>
</tr>
<tr>
<td>13</td>
<td>push C</td>
<td>C, B, (B+C)^3 * (D-C)</td>
<td>store A</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>push D</td>
<td>D, C, B, (B+C)^3 * (D-C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>mult</td>
<td>B*C, D, (B+C)^3 * (D-C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>mult</td>
<td>B<em>C</em>D, (B+C)^3 * (D-C)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>div</td>
<td>(B+C)^3 * (D-C)/BCD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>pop A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part 3: MIPS Architecture

[0x0040A030] add $1, $0, $0 // int x

[0x0040A034] lw $2, 80($10) // int a used in your C code

[0x0040A038] loop: slti $3, $1, 25

[0x0040A03C] beq $3, $0, end

[0x0040A040] addi $4, $2, 45

[0x0040A044] add $5, $4, $4

[0x0040A048] lw $6, 0($11) // int b used in your C code

[0x0040A04C] sub $7, $5, $6

[0x0040A050] sw $7, 0($11)

[0x0040A054] addi $11, $11, 4

[0x0040A058] addi $1, $1, 1

[0x0040A05C] j loop

[0x0040A060] end: addi $8, $2, 100

[0x0040A064] sw $8, 80($10)

Use this code from a portion of a program above, to answer the questions below.

• Encode the instruction at address [0x0040A05C] into a standard binary MIPS format. Identify the format and addressing mode(s) used by this instruction.

Format: J-type
Addressing Mode(s) used: pseudo-direct

|000010|0000100000001010000001110|
• Encode the instruction at address [0x0040A048] into a standard binary MIPS format. Identify the format and addressing mode(s) used by this instruction.
  Format: I-type
  Addressing Mode(s) used: base/offset, Register

  |100011|01011|00110|0000000000000000

• Encode the instruction at address [0x0040A03C] into a standard binary MIPS format. Identify the format and addressing mode(s) used by this instruction.
  Format: I-type
  Addressing Mode(s) used: pc-relative, Register

  |000100|00011|00000|0000000000100

• Encode the instruction at address [0x0040A04C] into a standard binary MIPS format. Identify the format and addressing mode(s) used by this instruction.
  Format: R-type
  Addressing Mode(s) used: Register

  |000000|00101|00110|00111|00000|1000010

• Write a C equivalent for the code above.

    int x, a, b;
    
    for (x = 0; x < 25; x++)
    
      b[x] = (2*(a + 45)) - b[x];

    a = a + 100;
    
    .
    .
    .