1 Compile, Assemble, Link, Load, and Go!

1.1 Overview

1.2 Exercises

1. What is the Stored Program concept and what does it enable us to do?

2. How many passes through the code does the Assembler have to make? Why?

3. What are the different parts of the object files output by the Assembler?

4. Which step in CALL resolves relative addressing? Absolute addressing?

5. What step in CALL may make use of the $at register?

6. What does RISC stand for? How is this related to pseudoinstructions?
2 Floating Point

2.1 Overview

The IEEE 754 standard defines a binary representation for floating point values using three fields:

- The *sign* determines the sign of the number (0 for positive, 1 for negative)
- The *exponent* is in **biased notation** with a bias of 127
- The *significand or mantissa* is akin to unsigned, but used to store a fraction instead of an integer

The below table shows the bit breakdown for the single precision (32-bit) representation.

<table>
<thead>
<tr>
<th>Sign</th>
<th>Exponent</th>
<th>Significand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>8 bits</td>
<td>23 bits</td>
</tr>
</tbody>
</table>

There is a double precision encoding format that uses 64 bits. This behaves the same as the single precision but uses 11 bits for the exponent (and thus a bias of 1023) and 52 bits for the significand.

How a float is interpreted depends on the values in the exponent and significand fields:

For normalized floats:

$$\text{Value} = (-1)^{\text{Sign}} \times 2^{\text{Exp} - \text{Bias}} \times 1.\text{significand}_2$$

For denormalized floats:

$$\text{Value} = (-1)^{\text{Sign}} \times 2^{\text{Exp} - \text{Bias} + 1} \times 0.\text{significand}_2$$

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Significand</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Anything</td>
<td>Denorm</td>
</tr>
<tr>
<td>1-254</td>
<td>Anything</td>
<td>Normal</td>
</tr>
<tr>
<td>255</td>
<td>0</td>
<td>Infinity</td>
</tr>
<tr>
<td>255</td>
<td>Nonzero</td>
<td>NaN</td>
</tr>
</tbody>
</table>

2.2 Exercises

1. How many zeroes can be represented using a float?

2. What is the largest finite positive value that can be stored using a single precision float?

3. What is the smallest positive value that can be stored using a single precision float?

4. What is the smallest positive normalized value that can be stored using a single precision float?

5. What is the smallest floating point greater than 1? 2? 4? 32?

- Now, for any power of 2, where $x = 2^y$, what is the difference between $x$ and the next largest floating point?

6. Cover the following numbers from binary to decimal or from decimal to binary:

- $0x00000000$
- $8.25$
- $0x00000F00$
- $39.5625$
- $0xFF94BEEF$
- $-\infty$