Floating Point and Intro to MIPS

CS230 Tutorial 03
Floating Point

The idea is to encode a number with a fractional part in binary. We split into three parts: sign (S), fraction (F) and exponent (E). A fourth part, called the bias (B) is always already known!

The formula is:
\[ (-1)^S \times 1.F \times 2^{E-B} \]
Floating Point Example

Formula: \((-1)^S \times 1.F \times 2^{E-B}\)

Example 8-bit format: 1-bit sign, 3-bit exponent, 4-bit fraction, bias = 3

\((-1)^1 \times 1.1011 \times 2^{6-3}\) simplify \(-1 \times 1.1011 \times 2^3\) multiply \(-1101.1\) convert \(-13.5\)
Floating Point Example

Formula: \((-1)^S \times 1.F \times 2^{E-B}\)

Example 8-bit format: 1-bit sign, 3-bit exponent, 4-bit fraction, bias = 3

\[
\begin{align*}
6.375 & = 6 \div 2 = 3 \text{ R } 0 & 0.375 \times 2 & = 0.75 \\
3 & = 3 \div 2 = 1 \text{ R } 1 & 0.75 \times 2 & = 1.5 \\
1 & = 1 \div 2 = 0 \text{ R } 1 & 0.5 \times 2 & = 1.0
\end{align*}
\]
Floating Point Practice

Convert the following decimal numbers to floating point:

3.5
-4.75

Convert the following floating point numbers to decimal:

10110110
01001101

The format is: 1-bit sign, 3-bit exponent, 4-bit fraction, bias = 3

Recall the formula: \((-1)^S \times 1.F \times 2^{E-B}\)
Floating Point Practice - Solutions

Convert the following decimal numbers to floating point:

- $3.5 = (-1)^0 \times 1.1100 \times 2^{4-3} = 01001100$
- $-4.75 = (-1)^1 \times 1.0011 \times 2^{5-3} = 11010011$

Convert the following floating point numbers to decimal:

- $10110110 = (-1)^1 \times 1.0110 \times 2^{3-3} = -1.0110 = -1.375$
- $01001101 = (-1)^0 \times 1.1101 \times 2^{4-3} = 11.101 = 3.625$

The format is: 1-bit sign, 3-bit exponent, 4-bit fraction, bias = 3

Recall the formula: $(-1)^S \times 1.F \times 2^{E-B}$
## Special Cases

<table>
<thead>
<tr>
<th>Exponent</th>
<th>Fraction</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000...</td>
<td>000000</td>
<td>0</td>
</tr>
<tr>
<td>000000...</td>
<td>non-zero</td>
<td>Subnormal : $(-1)^s \times 0.F \times 2^{1-B}$</td>
</tr>
<tr>
<td>111111...</td>
<td>000000</td>
<td>+/- Infinity</td>
</tr>
<tr>
<td>111111...</td>
<td>non-zero</td>
<td>NaN</td>
</tr>
</tbody>
</table>
Floating Point Addition

Just line up the decimal points!

Example:

\[
\begin{align*}
11.001 \times 2^4 &+ 1101.001 \times 2^2 \\
\quad &+ 110.01101 \times 2^4 \\
\hline
1110.01101 \times 2^4
\end{align*}
\]
Floating Point Addition Practice

Add the following sets of binary decimals:

\[
\begin{align*}
1.0010 & \times 2^4 & & 1.00101 & \times 2^6 \\
+1.101101 & \times 2^6 & & +1.01 & \times 2^3
\end{align*}
\]
Floating Point Addition Practice - Solutions

Add the following sets of binary decimals:

1.0010 × 2^4
+1.101101 × 2^6
\[= 1.101101 \times 2^6\]

0.010010 × 2^6
+1.101101 × 2^6
\[= 1.111101 \times 2^6\]

1.00101 × 2^0
+1.01 × 2^3
\[= 1.01101 \times 2^3\]

1.111111 × 2^6
+1010.0 \times 2^6
\[= 1011.00101 \times 2^6\]
MIPS Intro

We’ve learned about the simple MIPS operations add and sub during lecture. Let’s introduce addi!

*addi* takes a target register, a source register, and an immediate

Example:

```
addi $3, $0, 4  
```

Sets register 3 to the value in register 0, plus 4
- Register 0 always has value zero
- So the result here is always 4

```
addi $2, $5, 3  
```

Sets register 2 to the value in register 5, plus 3
MIPS First Program

Now that we know add, sub, and addi, let’s write a simple program. We want to calculate this formula:

\[ z = 2x + y - 12 \]

We assume the value for \( x \) is in \$1\,\$, the value for \( y \) is in \$2\,\$, and we are trying to put the result (the value for \( z \)) in \$3\,\$.

Notice that \$1\ means “register 1”, and \$2\ means “register 2” etc. This is a common shorthand, and is required in assembly language.
MIPS First Program - Solution

add $3, $1, $1
add $3, $3, $2
addi $4, $0, 12
sub $3, $3, $4
MIPS Programming Practice

Write your own programs that compute the following formulas.

\[ z = (x - 5) + (y - 13) \]
\[ z = (x + y + 3) - (x + 2) \]

Again we assume the value for \( x \) is in $1, the value for \( y \) is in $2, and we are trying to put the result (the value for \( z \)) in $3.
MIPS Programming Practice - Solutions

\[ z = (x - 5) + (y - 13) \]
addi $3, $1, -5
addi $4, $2, -13
add $3, $3, $4

\[ z = (x + y + 3) - (x + 2) \]
add $3, $1, $2
addi $3, $3, 3
addi $4, $1, 2
sub $3, $3, $4
Reading MIPS Code Practice

What formula do the following programs compute? Assume the value for x is in $1, the value for y is in $2, and the value for z is in $3 at the end.

```
addi $3, $1, -2
sub $3, $3, $2
addi $4, $0, 67
sub $3, $4, $3
```

```
add $3, $1, $1
add $3, $3, $1
addi $4, $2, 13
add $3, $3, $4
add $3, $3, $4
```
What formula do the following programs compute? Assume the value for x is in $1, the value for y is in $2, and the value for z is in $3 at the end.

```
addi $3, $1, -2
sub $3, $3, $2
addi $4, $0, 67
sub $3, $4, $3
```

\[ z = 67 - ((x - 2) - y) \]

```
add $3, $1, $1
add $3, $3, $1
add $3, $3, $1
add $3, $3, $1
addi $4, $2, 13
add $3, $3, $4
add $3, $3, $4
```

\[ z = 3x + 2(y + 13) \]
Assignment Reminders

● Always test your code on the university servers before you hand it in!
● Hand in the .asm file (*not the .mips file*) for programming questions.
● Submit a .txt XOR a .pdf for all other questions.
  ○ Do not submit both for the same question!
  ○ You may submit a .pdf for one question and a .txt for a different question.