LU01k - Binär codierte Fliesskommazahlen

Binäre Fliesskommazahlen

Siehe *http://www.ulthryvasse.de/gleitkommazahlen.html*

Zur Erinnerung: Fliesskommazahlen werden als Multiplikation dargestellt. Zum Beispiel $1.4735 * 10^5$.

- 1.4735 ist die Mantisse
- 10 ist die Basis
- 5 ist der Exponent

Oder als exponentielle Zahl dargestellt: 1.4735e5

Fliesskommazahlen werden zur binären Codierung in drei Elemente zerlegt:

- Vorzeichen: 0 = positiv / 1 = negativ
- Exponent: Ganzzahl zwischen -127 ($0000\ 0000_2$) und +128 ($1111\ 1111_2$)
- Mantisse: Binäre Ganzzahl, immer positiv

Das IEEE-Format definiert, wie die einzelnen Elemente binär codiert werden.

32 Bit Short-Format

- Bit 31: Vorzeichen (1 Bit)
- Bit 23 - 30: Exponent (8 Bit)
- Bit 0 - 22: Mantisse (23 Bit)

64 Bit Long-Format

- Bit 63: Vorzeichen (1 Bit)
- Bit 52 - 62: Exponent (11 Bit)
- Bit 0 - 51: Mantisse (52 Bit)

Dezimalzahl als binäre Fliesskommazahl

1. Notiere das Vorzeichen (+ = '0' / - = '1') und entferne das Vorzeichen.
2. Verschiebe den Dezimalpunkt bis eine ganze Zahl entsteht, die direkt vor dem Dezimalpunkt keine Nullen hat. Diese Zahl ist die Mantisse.
   - Bei einer Ganzzahl wird der Dezimalpunkt nach links verschoben (z.B. 1500 ⇒ 15).
   - Bei einem Dezimalbruch wird der Dezimalpunkt nach rechts verschoben (z.B. 4.178 ⇒ 4178).
3. Notiere die Anzahl Stellen um die der Dezimalpunkt bei 2. verschoben wurde als Exponent.
   - Der Dezimalpunkt wurde nach links verschoben: positiver Exponent (z.B. 1500 ⇒ 15 / Zwei Stellen nach links verschoben = 2).
   - Der Dezimalpunkt wurde nach rechts verschoben: negativer Exponent (z.B. 4.178 ⇒ 4178 / Drei Stellen nach rechts verschoben = -3).
4. Addiere den Bias zum Exponenten
   - Im Short-Format ist der Bias 127
   - Im Long-Format ist der Bias 1023
5. Notiere den Exponenten als Binärzahl mit 8 (Short-Format) oder 11 (Long-Format) Stellen.
6. Notiere die Mantisse als Binärzahl mit 23 (Short-Format) bzw. 52 (Long-Format) Stellen.
   - Ist die Binärzahl kürzer als die Anzahl Stellen ⇒ Fülle die Stellen mit führenden Nullen auf.
   - Ist die Binärzahl länger als die Anzahl Stellen ⇒ Schneide die überzähligen Stellen hinten ab.

Bias

Der Bias ist ein Korrekturwert für die binäre Speicherung des Exponenten. Dieser Korrekturwert wird zum Exponenten addiert, wodurch der Exponent in der binären Speicherform immer eine positive Zahl ist. Dadurch muss das Vorzeichen des Exponenten nicht gespeichert werden.

- Eine Fliesskommazahl im Short-Format hat einen Exponenten von -127 bis +128.
- Wir addieren den Bias von 127 zum Exponenten: Gespeichert wird dadurch ein Exponent von +0 bis +255.

Beispiele

In beiden Beispielen verwenden wir das 32 Bit Short-Format.

Schritt	Beispiel 1: -87900		Beispiel 2: 934.7531	
	Zwischenresultat	**Binäre Speicherung**	**Zwischenresultat**	**Binäre Speicherung**

Schritt	Beispiel 1: -87900		Beispiel 2: 934.7531	
	Zwischenresultat	**Binäre Speicherung**	**Zwischenresultat**	**Binäre Speicherung**
1. Vorzeichen codieren und entfernen	87900	**1**… …. …. …. …. …. …. ….	934.7531	**0**… …. …. …. …. …. …. ….
2. Dezimalpunkt verschieben	Mantisse=879		Mantisse=9347531	
3. Exponent notieren	Exponent=2		Exponent= -4	
4. Addiere 127 (Bias) zum Exponenten	Exponent=2+127=129		Exponent=-4+127=123	
5. Exponent als Binärzahl		1**100 0000 1**… …. …. …. …. ….		0**011 1101 1**… …. …. …. …. ….
6. Mantisse als Binärzahl		1100 0000 1**000 0000 0000 0011 0110 1111**		0011 1101 1**100 0111 0101 0000 1110 0101**[1]

Für Fliesskommazahlen im Long-Format gilt das gleiche Vorgehen. Die Unterschiede in der binären Speicherung sind:

- Die binäre Speicherung des Exponenten umfasst 11 Stellen.
- Der Bias ist 1023 statt 127.
- Die binäre Speicherung der Mantisse umfasst 52 Stellen

Exponentielle Zahl als binäre Fliesskommazahl

Die Umwandlung einer exponentiellen Zahl wie -8.79E2 bzw. $-8.79*10^2$ kann über den Umweg einer „normalen“ Dezimalzahl erfolgen.

Schritt	Beispiel 1: -8.79E4		Beispiel 2: 9.347531E2	
	Zwischenresultat	**Binäre Speicherung**	**Zwischenresultat**	**Binäre Speicherung**
0. Umwandlung	-8.79E4 = -87900		9.347531E2 = 934.7531	
Nun folgen die Schritte 1-6 von oben.				

Sie können den Umweg über eine Dezimalzahl auch auslassen:

- Führen Sie die Schritte 1 bis 3 aus.
- Im Schritt 3 addieren Sie den ursprünglichen Exponenten zur Anzahl der Stellen, um die der Dezimalpunkt verschoben wurde.
- Führen Sie anschliessend Schritt 5 und 6 aus.

Schritt	**Beispiel 1: -8.79E4**		**Beispiel 2: +9.347531E2**	
	Zwischenresultat	**Binäre Speicherung**	**Zwischenresultat**	**Binäre Speicherung**
1. Vorzeichen codieren und entfernen	8.79E4	**1**…	9.347531E2	**0**… ....
2. Dezimalpunkt verschieben	Mantisse=879		Mantisse=9347531	
3. Exponent notieren	Exponent=-2 + 4 = 2		Exponent=-6 + 2 = -4	
4. Addiere 127 (Bias) zum Exponenten	Exponent=2+127=129		Exponent=-4+127=123	
5. Exponent als Binärzahl		1**100 0000 1**…		0**011 1101 1**… ....
6. Mantisse als Binärzahl		1100 0000 1**000 0000 0000 0011 0110 1111**		0011 1101 1**100 0111 0101 0000 1110 0101**

Rundungsfehler

Binäre Zahlen können nur Brüche wie 1/2, 1/4, 1/8, 1/16, … darstellen. Daher kann es zu Rundungsfehlern kommen, wenn diese ins Dezimalsystem übertragen werden. Dies liegt daran, dass zur Darstellung eines Dezimalbruchs relativ viele binäre Stellen benötigt werden.

Beispiel

0.73_{10} = $10\ 1110\ 1011\ 1000\ 0101\ 0001\ 1111_{2}$

Somit brauchen wir 26 binäre Stellen um den Bruch korrekt abzubilden. Bei einer Fliesskommazahl im Short-Format (32 Bit) stehen aber nur 23 binäre Stellen für die Mantisse zur Verfügung. Somit könnten wir nur eine Annäherung abspeichern: 0.729999781_{10}

Selbst mit dem Long-Format (64 Bit) stossen wir schnell an Grenzen.

---

m114-A1G, m114-A1F, m114-A1E

1)

Die Mantisse hätte 24 binäre Stellen. Da aber nur 23 Bit Speicherplatz vorhanden sind, wurde die letzte Stelle abgeschnitten