計算機構成 夏季課題 2020年8月7日

浮動小数点形式に関する課題

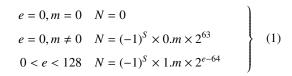
1. はじめに

講義では、5 ビット(符号 1 ビット,指数部 2 ビット,仮数部 2 ビット)の浮動小数点形式を例に解説した。本課題では、16 ビット(符号 1 ビット,指数部 7 ビット,仮数部 8 ビット)の浮動小数点形式に対するアセンブリ・プログラミングを通して、浮動小数点演算 [1] に関する理解を深める。

課題は 3 グループに分かれている。課題 1 から 5 までが Float 16 の出力に関する課題,課題 6 から 8 までが Float 16 の入力に関する課題,課題 9 が総括した課題である。 C プログラムを作成してから課題に取り掛かるとよいだろう。

2. 浮動小数点形式 Float16

この課題で扱う浮動小数点形式 Float16 を**図1**に示す. バイアスは 64 する. 符号ビット S, 指数部 e, 仮数部 m とすると,Float16 形式で表現できる数値は式 (1) のとおり. IEEE754 の単精度で指数部が 255 の場合は無限大もしくは非数であるが,Float16 では指数部が 127 であっても正規化数として扱うことにする.



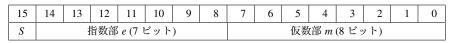


図1 Float16 形式

3. Float16を扱うCプログラム

数値を扱うアセンブリコードでは、テストデータ*1を用意するのが面倒である。このためテストデータを作成するためのヘッダファイルとプログラムを示す。

3.1 ヘッダファイル float16.h

Float16 形式を C 言語で扱うためのヘッダーファイルをリスト 1 に示す。Float16 形式はビットフィールドで定義してある。バイアスは F16BIAS と定義してある。このヘッダファイルを使った関数 PrintF16Formt をリスト 2 に示す。このプログラムは、Float16 形式を 16 進数 4 桁で表示する。

リスト1の12行目に関数 PrintF16Formt のプロトタイプ宣言がある。このように Float16 を使う関数は、リスト1の13 行以降にプロトタイプ宣言を追加すればよい。

関数 PrintF16Formt では,符号ビット Sign,指数部 Epart,仮数部 Mpart をそれぞれシフト操作と論理和で,変数iに16ビットのFloat16形式を格納している。ヘッダファイルに定義してある各部のビットフィールドと見比べて理解せよ.

次のように各フィールドに値を代入して、PrintF16Format を呼び出せば、16 進数 4 桁で 8324 と表示される.

```
Float16 x;
x.Sign=1;
x.Epart=3;
x.Mpart=0x24;
PrintF16Format(x);
```

リスト1 float16.h

```
#ifndef float16_h

#define float16_h

#include <stdio.h>
#define F16BIAS 64

typedef struct Float16 {
   unsigned int Sign : 1;
   unsigned int Epart : 7;
   unsigned int Mpart : 8;
} Float16;

void PrintF16Formt(Float16);

#endif /* float16_h */
```

リスト2 PrintF16Formt.c

```
#include "float16.h"
1
3
   void PrintF16Formt(Float16 x)
5
     unsigned int i;
     i = 0;
6
     i = i |
7
               (x.Sign << 15);
     i = i \mid (x.Epart << 8);
8
9
     i = i \mid x.Mpart;
10
     printf("%04X\n", i);
```

© 2020 Masaki Tomisawa

計算機構成 夏季課題 2020年8月7日

3.2 テストデータ作成用関数 FtoF()

IEEE754の単精度 (float 型)を Float16 に変換するプログラムを右に示す。IEEE754と Float16 は浮動小数点数形式の考え方は同じであり、大きな違いは指数部のサイズとバイアスである。仮数部はサイズだけが違う。指数部は Float16 のバイアスに変換し、仮数部は 16 ビットだけ切り出している。C プログラムで PrintF16Format (FtoF(3.14)) とすれば 3.14 を Float16 形式 4192_{16} に変換できる。 ちなみに、IEEE754で、3.14 は $d00f4940_{16}$ である。式 (1) で示したとおり、 IEEE754 にある無限大は考慮していない。

リスト3 ftof.c

```
#include "float16.h"
   // Convert IEEE754 to
2
                           F16Float
3
   Float16 FtoF(float f)
4
5
     Float16 f16;
     int i, e;
     i = *(int *)&f;
7
      f16.Sign = (i >> 31) & 0x00000001L;
      f16.Mpart = (i & 0x007FFFFFL) >> 15;
9
10
11
     e = ((i \& 0x7F800000L) >> 23);
12
     if (e == 0) f16.Epart = 0;
13
     else f16.Epart = e - 127 + F16BIAS;
14
     return f16:
15
```

4. 課題

課題で作成するアセンブリ・コードは、QtSPIMで実行できるコードとすること.

課題 1 実数 2.71828 を IEEE754 の単精度と Float16 に手計算で変換せよ.手計算の結果は,PrintF16Format 関数と FtoF 関数を使って検証せよ.

課題 2 Float 16 形式で正数の最大値と、0 でない最小値を 10 進数と2 進数で示せ、

課題3 右に示した Float16 形式の仮数部だけを 10 進数に変換する C プログラムをアセンブリ・コードに変換せよ. アセンブリでの関数名は MpartP とせよ. 関数 printf() による数字の出力については,配布した fact-loop.s を参照せよ. 詳細は, QtSPIM の Help で System Calls に記載されている.

|課題4| Float16 形式を 10 進数で出力するアセンブリ・プログラムを作成せよ。アセンブリでの関数名は F16P とせよ。仮数部の出力は,課題 3 の解答を使用すればよい。出力例を右表に示す。指数部は,例のように 2 のべき乗で出力すればよい。文字列" $*2^("$ と")"の出力は配布した fact-loop.s を参照せよ。詳細は,QtSPIM の Help で System Calls に記載されている。

課題5 課題4で作成したアセンブリ・コードの動作を検証するためにテストデータを作成せよ。テストデータの例を右表に示す。テストデータには、正しい結果が得られる入力データと、その出力データを含める。ゼロ、課題2の解答、非正規化数など、プログラムの動作を検証する上で適切な入力データとその出力データも含めること。

なお、本来ならば、テストデータは課題4と一緒に作成する.

リスト4 MpartPrint.c

```
1
   void MpartPrint(Float16 x)
2
     unsigned int m;
3
4
     m = x.Mpart;
     while (m != 0) {
5
       m = (m << 1) + (m << 3);
6
7
       printf("%d", m >> 8);
8
       m = m \& 0x00ff;
9
10
```

表 1 出力例

Float16 形式	出力	
0000	+0	
0027	+0.15234375*2^(-63)	
AAAA	-1.6640625*2^(-22)	
8080	-0.5*2^(-63)	
8888	-1.53125*2^(-56)	

表 2 テストデータ (例)

24 = 7 7 1 1 7 7 (14)		
入力	出力	コメント
0000	+0	正のゼロ
8000	-0	負のゼロ
		正の最大値
		0 でない最小値
		正の非正規化
		負の非正規化
以下略		

計算機構成 夏季課題 2020年8月7日

課題 6 符号なし整数 (32 ビット) を 10 で除算する C プログラムを右に示す.この C プログラムをアセンブリ・コードに変換せよ.アセンブリでの関数名は div10 とせよ.

除算命令は遅いので、シフト命令と加算命令で実装する方法がある。簡単な例として、205 倍して 11 ビット左シフト $\left(2^{11}=2048\right)$ すれば $0.1\left(\approx\frac{205}{2048}\right)$ 倍することができる。この C プログラムでは、32 ビットの精度で $0.1\left(=\frac{858993459}{8589934592}\right)$ 倍している [2]。

課題 7 文字列を 32 ビットの固定小数点形式に変換する方法について考える。例えば、文字列 "27.75" を入力すると、符号なし整数として (001BC000)₁₆ を出力する。中央に小数点があり、上位 16 ビットが整数、下位 16 ビットが少数である。右のプログラムは、整数部 x を 13 行目の returnで 16 ビット左シフトしている。

数字の文字列を 32 ビットの固定小数点形式に変換する アセンブリ・コードを作成せよ. 課題 6 で作成した div10 を使うこと. アセンブリでの関数名は, StoF32 とせよ.

QtSPIM での入出力については、QtSPIM の Help の System Calls に記載されている.

リスト5 Div10.c

```
1
   unsigned int Div10(unsigned int k)
2
     unsigned int j, m, n, p, q, r;
3
4
     j = (k + 1) << 1;
5
     m = j + (j << 1);
     n = m + (m >> 4);
7
    p = n + (n >> 8);
     r = p + (p >> 16);
9
     q = r >> 6;
10
     return q;
11
```

リスト6 StoFix32.c

```
1 unsigned int StoFix32(char *s)
2
3
      // Integer part
     int x = 0:
4
     while (*s != '.') {
6
       x = (x << 1) + (x << 3) + *s - '0';
       s = s + 1;
8
    // Fractional part
10
    int y = 0;
11
    // your code
12
13
      return (x << 16) + ( y & 0x00ff ) ;
14
```

[課題 8] 課題 7 で変換した 32 ビットの固定小数点形式を,式(1)に従うように,正規化して Float 16 形式に変換する アセンブリ・コードを作成せよ。また,このアセンブリ・コードに対するテストデータを示せ。テストデータを示す ことにより,作成したアセンブリ・コードが正しく動作する引数の範囲が分かるようにせよ.

| 課題 9 | Float 16 同士を加算し、結果を出力するアセンブリ・コードを作成せよ。教科書の 195 頁の「浮動小数点加算」と 198 頁の例題「2 進の浮動小数点加算」が参考になる。

5. おわりに

課題レポートの提出については次回の授業で伝える。課題をとおして、QtSPIMの使い方に習熟しておくこと。課題について、不明な点があれば、UNIPAのQ&Aを使って問い合わせること。

参考文献

- [1] ジョン・L. ヘネシー, デイビッド・A. パターソン: コンピュータの構成と設計 第5版上, pp. 190-198, 日経 BP 社 (2015).
- [2] Vowels, R. A.: Divide by 10, Australian Computer Journal, Vol. 24, No. 3, pp. 81–85 (1992).

© 2020 Masaki Tomisawa 3