○はじめに

　サンプルソースプログラムを実行するためには、ミクロアセンブラを使用して、サンプルソースファイルから".COM"の拡張子を持つ実行ファイルを作成しておかなければなりません。　下記の例を参考にミクロアセンブラを起動し、実行ファイルを作成して下さい。

例　D:¥Samples>%PATH%¥uASM D:¥Samples¥Hello.asm

（%PATH%にはミクロアセンブラの格納フォルダ名を指定して下さい）

全てのサンプルソースファイルをアセンブルすると、以下の".COM"形式実行ファイルが出来上がります。

```
D:¥Samples>dir *.com /w
 ドライブ D のボリューム ラベルは [redacted] です
 ボリューム シリアル番号は [redacted] です

 D:¥Samples のディレクトリ

ARGSplit.COM   Calculat.COM   Calc_DOW.COM   Calendar.COM   DigClock.COM
Hello.COM      InputDec.COM   PrintDec.COM
               9 個のファイル                 13,927 バイト
               0 個のディレクトリ  455,468,634,112 バイトの空き領域
```

プログラムサイズが小さい簡単（単純）なプログラムから、順を追って説明して行きます。

○サンプル１：Hello.asm　　固定文字列を表示するサンプル

　プログラム中に埋め込まれた文字列"Hello, world!"を画面に表示します。

```
D:¥Samples>Hello↵
Hello, world!
```

プログラム中の文字列を好きな言葉に置き替えて、実行してみて下さい。　文字数に制限はありませんし、改行コードを含めることも出来ます。

○サンプル２：PrintDec.asm　固定数値(16 進数)を 10 進数に変換して表示するサンプル

　文字列ではなく、プログラム中に埋め込まれた数値を文字列に変換して表示します。

```
D:¥Samples>PrintDec↵
12345
```

サンプルソースでは固定数値を 10 進数で指定していますが、2 進数や 8 進数など一般的ではない表記の数値が 10 進数ではいくつになるのか調べるのに最適です。

○サンプル３：InputDec.asm　数字をキーボードから入力して 16 進数に変換するサンプル

　このプログラムは、プログラムを実行するとキー入力待ちになります。

好きな数字を５桁以内で入力して最後に Enter キーを押すと、入力した文字列を 16 進数に変換して内部の変数領域に格納します。その後変数に格納した 16 進数を PrintDec と同じプログラムで画面に表示するプログラムです。 キー入力待ちでは数字と Enter キー以外を入力するとビープ音が鳴ります。

```
D:¥Samples>InputDec↵
12345↵
 2361
```

プログラムに問題が無ければ入力した文字列と表示される文字列は同じになるはずですが、現状ではバグがあるため結果が異なって表示されています。

　デバッガを使用してバグの原因を調査し、正しい結果が得られるように修正してみるのも良い練習になると思います。

　また、アルファベットの'A'～'F'を入力できるように変更すれば、16 進数を 10 進数に変換して表示するプログラムに修正可能です。　但し、元々の入力は 10 進数だったことに注意してこれを 16 進数に変更することも忘れずに。

○サンプル４：Calculat.asm　32 ビット整数の加減乗除を行うサンプル(簡易電卓)

　このプログラムも、プログラムを実行するとキー入力待ちになります。

上のサンプル InputDec.asm と異なるのは、プロンプト（ガイドメッセージ）を表示して入力が必要なことを案内表示していることです。

二つの数値と演算記号を入力させ、四則演算の結果をメッセージと共に表示します。

演算記号は１文字入力するだけで次のデータ入力に進みます。

```
D:¥Samples>Calculat↵
Enter first value: 12345↵
Enter operator(+-*/): *
Enter second value: 12345↵
Calcurated result is  152399025
```

サブルーチンの作り方・使い方と、16 ビットのレジスタを組み合わせて 32 ビットの四則演算を実現する方法について参考にして下さい。

尚、乗算の計算では別のアルゴリズムを用いる 2 種類のサブルーチンを用意しています。それぞれの計算方法の違いを見比べて、メリット・デメリットを比較すると、より理解が深まるのではないでしょうか。

○サンプル５ ：ARGSplit.asm　起動パラメータを空白(スペース)で分割するサンプル

　次は、プログラムの実行途中で文字列を入力させるのではなく、プログラムの起動パラメータに指定した文字列を処理する例です。

プログラム名の後ろに任意の文字列を空白で区切って入力します。

```
D:¥Samples>ARGSplit one two three four five
one
two
three
four
five
```

このプログラムでは、起動パラメータを空白で区切って個別の文字列に分割し、それ等を各行に分けて画面に表示しています。

　これに続く二つのサンプルプログラムは、このサンプル５の一部をそのまま利用しています。

○サンプル６ ：Calc_DOW.asm　指定した日付の曜日を求めるサンプル

　起動パラメータに、年月日の３つの数字を入力します。 年は西暦です。

そうすると、入力した日の曜日が計算されて画面に表示されます。

```
D:¥Samples>Calc_DOW 2016 8 11
Thursday
```

誕生日を指定して、自分の誕生日は何曜日だったのか調べてみてはいかがでしょう。

○サンプル７ ：Calendar.asm　１ヶ月分のカレンダーを表示するサンプル

　曜日を求めるプログラムを応用して、１ヶ月分のカレンダーを表示するプログラムです。

起動パラメータに西暦年と月を指定します。

```
D:¥Samples>Calendar 2016 8
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

このサンプルプログラムを拡張して、１年分のカレンダーを横３ヶ月、縦４ヶ月の一覧表形式とする年間カレンダーを出力させるのも面白いと思います。

○サンプル８：DigClock.asm　デジタル時計をエミュレーションするサンプル

　プログラムを起動すると図の様にデジタル時計が表示されます。 秒が変わる度に表示を更新しますので、市販のデジタル時計と同じ動きに見えます。

コントロール+C キーを押して終了させます。

　内部的には７セグメント表示器に数字を出力する場合を想定して出力データを生成しています。　また、表示する文字の位置を制御するためにビデオサービス BIOS(INT 10H)を使用していますので、BIOS を呼び出す方法の参考にして下さい。