アセンブラプログラムのデバックの方法

アセンブラプログラムのデバックは、gdb（gnu debugger）を使って行うことができます。

gdb の起動

gdb は、単独でも起動することができますが、emacs から起動すると便利です。実行プログラムを a.out とすると、まず、emacs から、

```
M-x gdb
```

と入力します。`Run gdb (like this): gdb` とプロンプトでるので、ここで、a.out と入力し、リターンします。そこで、gdb の window が開かれるはずです。

ブレークポイントの設定と実行開始

課題のプログラムは main から始まるので、まず、ここで停止するように、break コマンドで main にブレークポイントを設定します。(gdb)とプロンプトがでるので、ここで、

```
(gdb) break main
```

と入力します。次に、run コマンド main まで実行します。

```
(gdb) run
```

すると、実行が始まり、main で停止するはずです。

プログラムの disassemble

ここで、プログラムがどのようなコードになっているかについて、確認してみましょう。メモリ上の機械語になったプログラムをアセンブリプログラムで表示するのが disassemble コマンドです。disassemble とは、アセンブルの反対、つまり、機械語からアセンブラに直すことです。main から始まるプログラムを disassemble してみましょう。

```
(gdb) disassemble main
```

main のところに、任意のラベル名を書くことでそのプログラムを disassemble することができます。

プログラムのステップ実行

１命令づつ実行するコマンドが、stepi です。

```
(gdb) stepi
```

ここで、stepi コマンドを実行するごとに１命令づつ実行されているのがわかるはずです。

レジスタの表示

step 実行している途中で、レジスタの表示をして見ましょう。表示には２つの方法があります。

```
(gdb) info registers
```

では、すべてのレジスタの表示を行います。個別のレジスタを表示する場合には、

```
(gdb) print $レジスタ名
```

で表示させることができます。

実行の再開、ブレークポイントの設定

continue コマンドは実行を次のブレークポイントまで（もしくは終わりまで)、実行を再開するコマンドです。

```
(gdb) continue
```

さて、main にブレークポイントを設定しましたが、main の代わりにラベル名を書くことで、そのラベルの前で実行を止めることができます。また、アドレスを指定したい場合には

```
(gdb) break *アドレス
```

で任意のアドレスで実行を中断することができます。

データの表示

データの表示を行うコマンドが x コマンドです。

```
(gdb)  x  アドレス
```

で、アドレスの内容をプリントすることができます。x のあとには、データ表示のフォーマットができて、例えば、x/のあとに、表示するデータの数、10 進（ｄ)、１６進（ｘ)、８進（o）とそのあとに、b(byte), h(half), w(word)と指定します。たとえば、

```
(gdb) x/10dw  0x10000
```

では、0x10000 番地から、32 ビットごと（w）に１０進（ｄ）で、１０ワード表示するという意味になります。詳しくは、**help x** としてみてください。

他のコマンドについても、help コマンドで調べることができます。