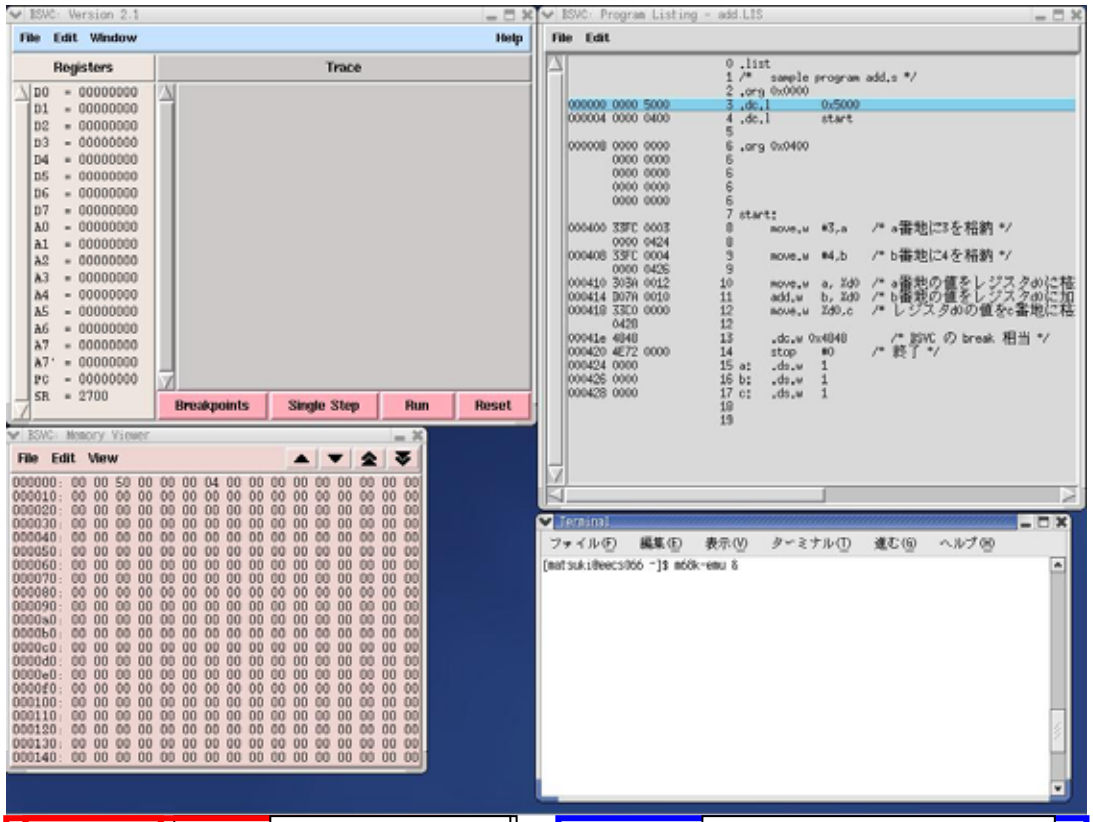
基礎実験１　UNIX・アセンブラ実習　第４回

2007 年 4 月 19 日

実習内容

エミュレータ（m68k-emu）を使って，いくつかの簡単なプログラム実行を試しながら，アセンブラのプログラムに慣れることを行う．エミュレータを使って，分岐命令における各種レジスタとメモリの値の変化を注意深く観察し，CPU の振る舞いについて理解を深めていく．

1. エミュレータ(m68k-emu)の実行画面



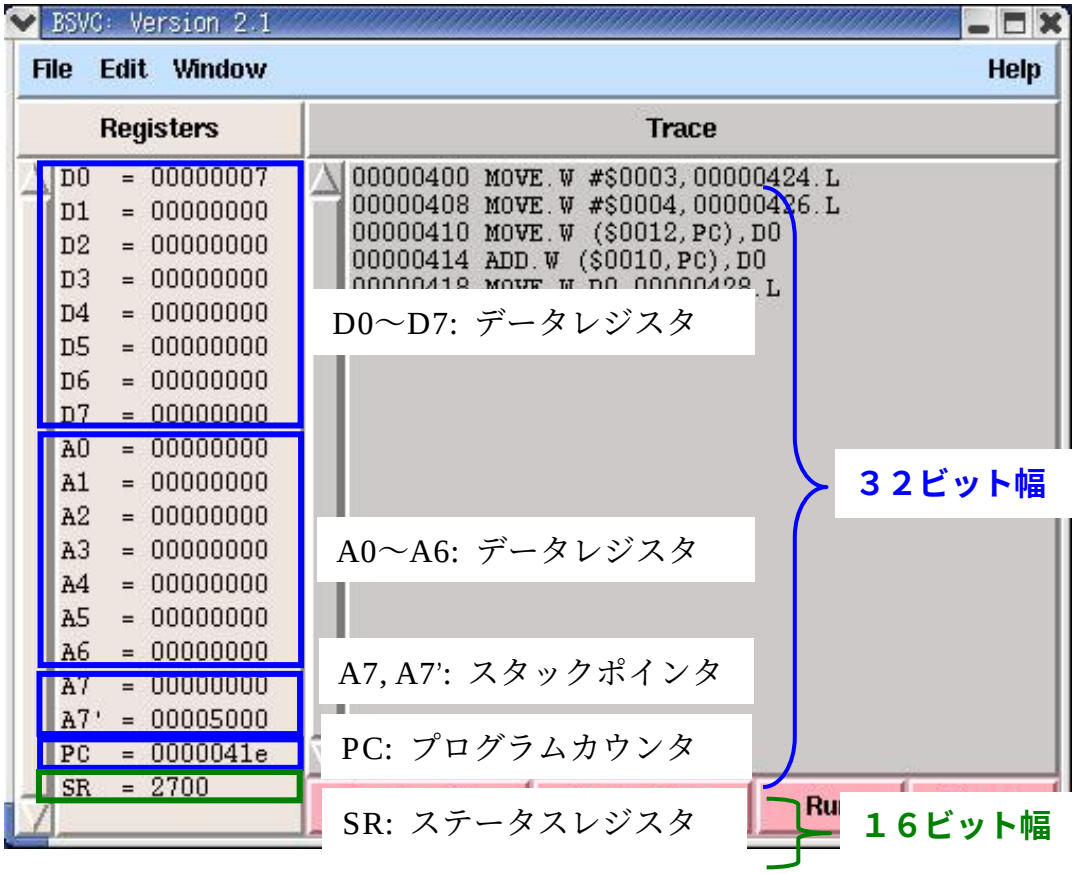
メモリ領域

CPU 内

各種レジスタの値（**刻々と変化する**）

m68k-emu が読み込んだ実行プログラム

エミュレータ(m68k-　　U の全てのレ　　やステータスレジスタも）を観察　　メモリアドレス　　もできます．この機能は，680　　するのに役立ちます．また，作成したプログラム中のバグの発見にも役立ちます．



レジスタの値は，１６進数の８桁（つまり３２ビット＝４バイト）での表示です.

例）D0 = 000000007

メモリ中のデータの値は，１６進数の２桁（つまり８ビット＝１バイト）に区切られての表示です.

2. ファイルの準備

今日の実習では、アセンブラの方法と、エミュレータの使い方について学びます。

まず、アセンブラソースプログラムファイルを用意します。次の命令を Terminal 上で実行して下さい。

```
$ /u/matsuki/jishu3-setup  ユーザ名  ＜Enter キー＞
```

3. 擬似命令 equ

ここでは, 擬似命令 equ について学ぶ. サンプルプログラム sub2.s を emacs で開いて見てみましょう.

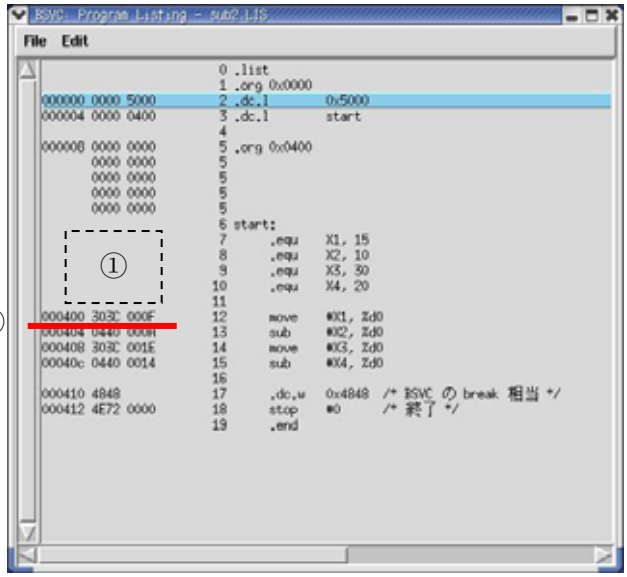
```
.org  0x0000
.dc.l 0x5000
.dc.l start

.org  0x0400
start:
      .equ      X1, 15
```

```
        .equ    X2, 10
        .equ    X3, 30
        .equ    X4, 20

        move    #X1, %d0
        sub     #X2, %d0
        move    #X3, %d0
        sub     #X4, %d0

        .dc.w   0x4848  /* BSVC の break 相当 */
        stop    #0      /* 終了 */
        .end
```

前回の実習に習って, この sub2.s をアセンブルし, BSVC のエミュレータを使って, 実行しましょう.

擬似命令 equ は, 指定された値をシンボル化するものである. 例えば, 上のプログラムでは,「.equ X1, 15」において, 15(10 進数)を X1 という名前でシンボル化している. この様子を BSVC の「Program Listing」で見てみよう（下図）. そうすると, まず, 気づくことは擬似命令の部分は, アセンブルしたコード（機械語）がないということである（下図①）. これは, アセンブルした段階で, シンボル化した変数は, すべて対応する値で置換されるためである. そのため,「equ」に相当する機械語は表示されない（存在しない）のである.

400 番地の命令は,「move #X1, %d0」であり, シンボル化した X1 が使われている. それに対応する機械語では,「303C 000F」とある（下図②）. 前半の「303C」の意味は,「レジスタ d0 に, ソース・オペランドの値を転送（move）しなさい」というものである※. そして, 後半の意味はソース・オペランドに指定した値そのものであり, ここでは 16 進数で 0F（10 進数で 15）が指定されている. X1 でシンボル化した 15 という値が, アセンブルした時点（機械語になった時点）で, 機械語の命令に直接反映されていることが分かる.

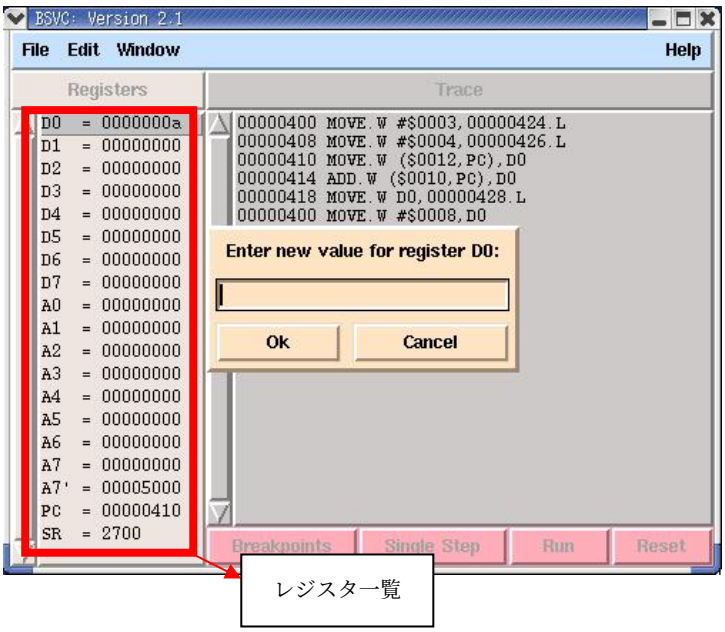


※アセンブラ命令資料の p.266 を参照

4. エミュレータ(m68k-emu)の諸機能

(1) レジスタへの値の設定

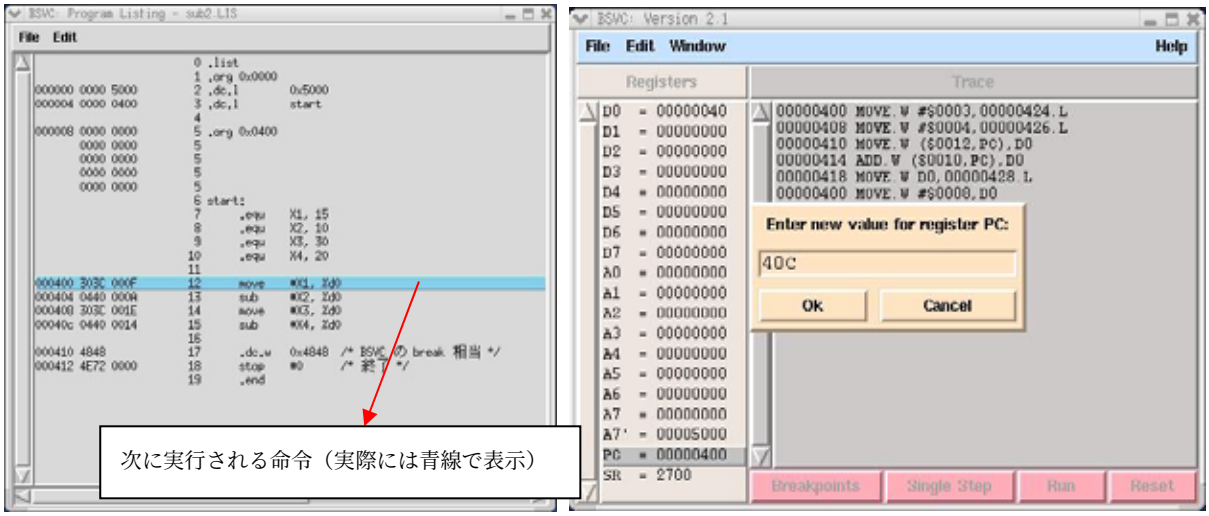
レジスタの値を設定する方法を説明する．ここでは，例としてレジスタ D0 の値を設定してみる．BSVC の画面（下図）の左側にあるレジスタの一覧から，設定したいレジスタ（ここでは D0）をマウスでダブルクリックする（選択されると灰色になる）．そうすると，画面の様にレジスタの値を設定するダイアログが開くので，そこに任意の値を入力し，「OK」のボタンをクリックする．その結果，レジスタの値が変更される．ここでは，レジスタ D0 に 40（16 進数）を入力しておきましょう．



(2) プログラムカウンタに値を設定してのプログラム実行開始

この後の説明のために，「Reset」ボタンを１回クリックしておきましょう．

プログラムカウンタの値は，レジスタ一覧の中の下から２番目にある，「ＰＣ」と書かれた所にある．プログラムカウンタは，次に実行する命令が格納されている，アドレスを保持している．左下図の例においては，現在プログラムカウンタの値が「00000400」となっている（ここでは，400 の上位の桁の 0 は無視して話を進める）ことから，次に実行されるのは，Program Listing 中の青線が引かれている行「move #X1, %d0」である．ここで，プログラムカウンタを先のレジスタの値の設定を利用して，直接変更してみよう．レジスタ一覧の中にあるプログラムカウンタをダブルクリックし，その値を「40C」に変更する§（右下図）．それにより，プログラムを強制的に 40C 番地から始めることが可能になる．ここで，注意することは，プログラムカウンタを直接変更しただけであり，400 番地，404 番地，406 番地の命令は実行されずに飛ばされたということである．そのため，プログラム終了時の計算結果は(1)でレジスタ D0 に入力した値 40（16 進数）から X4 の値 20（10 進数）を引いたものになっているはずである．計算どおりになったか確認せよ．



§ レジスター一覧には８桁の表記で PC の値が表示されているが，ダイアログで，値を入力するときは，40C の上位の桁の 0 は省略して入力することが出来る．

課題１

- **sub2.s を以下のように書き換えた（sub3.s）．この sub3.s をステップ実行し，各 sub 命令実行直後の PC（プログラムカウンタ），データレジスタ D0，SR（ステータスレジスタ）の値を調べなさい．さらに，その SR の値の意味について説明しなさい．**

```
.org      0x0000
.dc.l     0x5000
.dc.l     start

.org      0x0400
.section .text
.even
start:
    .equ   X1, 15
    .equ   X2, 10
    .equ   X3, 30
    .equ   X4, 20

    move   #X3, %d0
    sub     #X3, %d0
    move   #X4, %d0
    sub     #X1 %d0
    move   #X2, %d0
    sub     #X3, %d0
```

```
        .dc.w    0x4848  /* BSVC の break 相当 */
        stop     #0      /* 終了 */
        .end
```

5. 無条件分岐命令の振る舞い

ここでは，ステップ実行を用いて，無条件分岐命令を使ったときのプログラムの流れを確認する．特にプログラムカウンタＰＣの値の変化にも注意すること．

課題２

lenz.s についての問題．

(1) 「bra SLEN_LP」の実行直後のプログラムカウンタの値を報告しなさい．

(2) 1 行の命令を実行するのに 1 秒かかるとする．このとき，clr 命令（プログラムスタート）からプログラムカウンタが 000418 になるまで（.dc.w 0x4848 まで）に要する時間を求めよ．

- **lenz.s は，「/u/matsuki/jishu3-setup ユーザ名　＜Enter キー＞」の実行によって作成されるファイルである．その中身は次の通り．**

```
**
**      $00 で終了するデータの長さを求めるプログラム
**


**
**      STR から始めて、$00 が見つかるまで
**      1 バイトずつカウントする
**      カウント結果はレジスタ d0 に格納する
**


.org    0x0000
.dc.l   0x5000
.dc.l   start

.org    0x0400
.section .text
.even
start:
        clr.l   %d0     /* カウンタ(d0)のクリア  */
        lea     STR, %a0
SLEN_LP:
        move.b  (%a0)+, %d1
```

```
        cmpi.b  #0, %d1 /* d1 と??が等しければ、??へジャンプする  */
        beq     STR_END
        addq.l  #1, %d0 /* 1 カウントアップする  */
        bra     SLEN_LP

STR_END:
        ** 実行の終り
        .dc.w   0x4848  /* BSVC の break 相当 */
        stop    #0      /* 終了 */

.section .data
.even
STR:    dc.b    'a','b','c','d','e',0
        .end
```

6. cmp 命令と条件分岐でのステータスレジスタとプログラムカウンタの変化

課題 3

lenz.s についての問題

(1) 「cmpi.b #0, %d1」での処理内容について説明しなさい

(2) (1)の実行直後の SR の値を調べなさい．また，その意味（なぜそうなったのか）を説明しなさい．

(3) 「beq STR_END」の実行直後のプログラムカウンタの値を，ステップ実行によって観察し，結果を報告しなさい

今日の実習はここまでです。

参考 Web ページ: http://www.db.is.kyushu-u.ac.jp/kaneko/as/index.html