基礎実験１　UNIX・アセンブラ実習　第６回

2007 年 5 月 26 日

実習内容

サブルーチン呼び出しと復帰には，jsr 命令，rts 命令が関係します．今日の実習では，jsr 命令，rts 命令でのスタックエリアや各種レジスタの変化を観察し，サブルーチン呼び出しと復帰のメカニズムについて理解を深めます．

1. プログラムの準備

今日の実習のために、アセンブラソースプログラムファイル min.s を用意します。emacs を使って、下記の min.s を入力しましょう。この min.s は、与えられた数値データ（**9,5,3,7,6,4,8**）の中から最小値を探し出すものである。

```
**
** 最小値を探索する
** min.s
**
        .org    0x0000
        .dc.l   0x5000
        .dc.l   start

        .org    0x0400
**------------------------------------
**      メインルーチン
**------------------------------------
start:
        move.l  #0x12345678, %d0 /* レジスタ退避を学ぶため、わざとレジスタd0に値を入れている */
        lea.l   DATA,%a1         /* サブルーチンに移る前の準備としてa1にDATAのアドレスを格納 */
        jsr     MINIMUM          /* MINIMUMサブルーチンに処理を移す */
        .dc.w   0x4848
        stop    #0

**------------------------------------
**      サブルーチン（最小値探索）
**      入力（引き数）    %a1:探索対象データの先頭アドレス
**      出力（戻り値）    %d1:結果(最小値)
**------------------------------------
MINIMUM:
```

```
        movem.l  %a1/%d0,-(%a7)   /* レジスタの退避（push）（a1,d0の値をスタックに保存する） */
        moveq.l  #LENGTH,%d0      /* d0 = LENGTH - 1 */
        subq.w   #1,%d0
        move.w   (%a1),%d1

LOOP1:
        adda.w   #2,%a1           /* a2 = a2 + 2  */
        cmp.w    (%a1),%d1
        bcs      LABEL1
        move.w   (%a1),%d1
LABEL1:
        subq.w   #1,%d0
        bne      LOOP1

        movem.l  (%a7)+,%a1/%d0   /* レジスタの復帰（pop） */
        rts                       /* サブルーチン呼び出し元に戻る */
**--------------------------------------
** データエリア
**--------------------------------------
        .org     0x0500           /* データ領域の開始番地 */
        .equ     LENGTH, 7        /* データの個数 */
        DATA:    dc.w    9,5,3,7,6,4,8
```

プログラムの入力が出来たら、実行ファイルを作るためにアセンブルをしましょう。

```
$ m68k-as min.s  ＜Enter キー＞
```

エラーメッセージがなければ、68000 エミュレータで実行してみましょう。

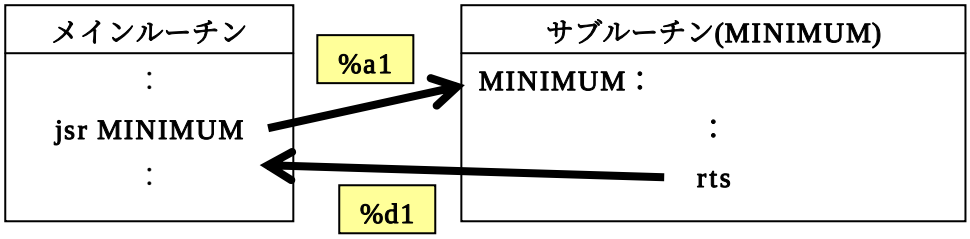
```
$ m68k-emu & ＜Enter キー＞
```

File メニューから、Load Program を実行し、min.abs を選択してましょう。同時に Memory Viewer、Program Listing のウィンドウも起動しておきましょう。

2. jsr 命令と rts 命令とスタック領域

このプログラムは、メインルーチンとサブルーチンに分けて作られている。メインルーチンでは、サブルーチンに処理を移行する前の準備として、レジスタ a1 に探索対象となるデータのアドレスを格納し、その後 jsr 命令でサブルーチン MINIMUM に処理を移行する。サブルーチンではレジスタ a1 で指示されたデータ列の中から最小値を探索し、その結果をレジスタ d1 に格納する。そして、最後に rts

命令でメインルーチンに戻ってくる。



このサブルーチン呼び出しの仕組みを理解するために、前回の実習で行ったブレークポイントおよびステップ実行を使ってみましょう。特に pc（プログラムカウンタ）と、a7(スタックポインタ)に注目しましょう（金子先生の講義資料も参照）。

ブレークポイントをメインルーチンの jsr 命令の行に設定し、一度「Reset」ボタンを押した後、「Run」ボタンを押しましょう。jsr 命令で停止した際、pc の値を覚えておきましょう。その後、「Step」実行で１行だけ実行し、サブルーチンに処理が移行したら、次のことを確認しましょう。

（１） pc の値が jsr 命令の行のアドレスからサブルーチンの先頭行のアドレスに変わる。

（２） スタックポインタ（レジスタ a7）が指すシステムスタックに、jsr 命令の次の行のアドレスが格納されている。

（注意）サブルーチンを利用するためには、サブルーチンからメインルーチンに処理が戻るときのために、どこに戻れば良いのかを記憶しておく必要がある。そのため、サブルーチン終了後に実行すべき行（jsr 命令の次の行）のアドレスをシステムスタック領域と呼ばれる場所に格納しておくのである。

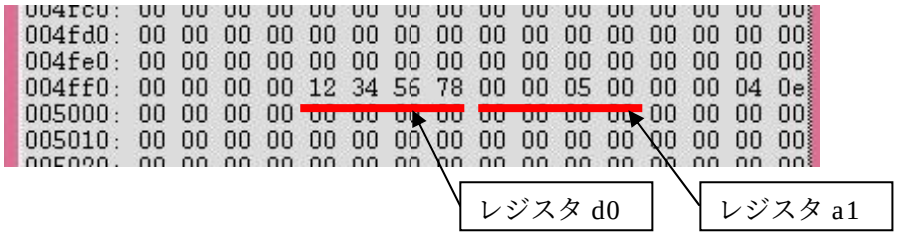
サブルーチンの 1 行目「 movem.l　%a1/%d0, -(%a7)」は、「レジスタ a1 と d0 をシステムスタックに退避（push）する」という命令である。これは move.l を 2 回使って、レジスタ a1 と d0 をそれぞれシステムスタックに push しても同じであるが、この movem.l を使うと 1 行で複数のアドレスレジスタ・データレジスタを一度に扱うことができる（命令表 p.274 参照）。このサブルーチンで実際に使っているレジスタは

a1：探索対象データのアドレスを指す

d0：残りの探索対象データの個数

d1：最小値（結果）

の３つである。これらのうち、レジスタ d1 はメインルーチンに結果を返すために使っている。一方、レジスタ a1 と d0 はサブルーチン内で値が変わってしまうため、そのままではサブルーチン呼び出し前のレジスタ値は失われてしまう。それを防ぐために、サブルーチンの初めに、システムスタック領域にレジスタの値を退避させるのである。下図は、サブルーチンの 1 行目を実行した後のシステムスタック領域の様子である。レジスタ a1 と d0 の値が順に退避されているのを確認しましょう。



サブルーチン内の最小値探索の過程は「Step」ボタンを押しながら、順次確認しましょう。最小値探索のためのループが終了した時点で、レジスタ d1 には答えが格納されているはずです。そして、その後、退避しておいたレジスタを復帰（pop）させます。これによりレジスタ a1 と d0 は、サブルーチン呼び出し前の値に戻っているはずです（レジスタ値を確認）。それと同時にスタックポインタ（レジスタ a7'）も変化しているはずです（レジスタ値を確認）。サブルーチンの最後に rts 命令によって、スタック領域に格納しておいた「戻りアドレス」を pc に復帰させることになります。このようにスタック領域は、一時的に数値やアドレスを保持しておくために使われ、特にサブルーチン処理などでは、重要な役割を果たします。

課題１． 実習中のプログラム min.s に関して，次の設問に解答せよ

(1) ラベル LOOP1 の次の行において、なぜアドレスに２を足すのかを説明せよ。

(2) プログラム min のサブルーチン中の以下の命令について、レジスタ a1、d0 を復帰（pop）する前のスタックポインタ a7 が 0x4ff0 のとき、レジスタを復帰した後の a7 はいくらになるか？

movem.l (%a7)+, %a1/%d0

(3) 実習中のプログラム min において、探索対象データ中の数値「８」が格納されているメモリアドレスはいくらか。

(4) プログラム min の結果、最小値はいくらなったか。

課題２． 条件付分岐命令 bcs の意味を説明せよ。

課題３．以下の命令が何を行っているか，なるべく分かりやすく，詳しく説明せよ．

(1) move.w (%a0)+, %d0

(2)
```
.equ DATA, 0xabcd
move.w #DATA, %d0
move.b %d0, %d1
```

(3)
```
.equ OFF, 4
move.l OFF(%a0), %d0
```

(4) (各命令の CCR の変化についても説明すること)
```
move.w #5, %d0
sub.w #7, %d0
add.w #7, %d0
```

(5) (各命令の CCR の変化についても説明すること)
```
move.w #0x7fff, %d0
add.w #0x0002, %d0
```

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