基礎ソフト実験　UNIX・アセンブラ実習　第４回

実習内容

サブルーチン呼び出しと復帰には，jsr 命令，rts 命令を用います．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
**
.section .text
**-------------------------------------
**      メインルーチン
**-------------------------------------
start:
        move.l  #0x12345678, %d1 /* レジスタ退避を学ぶため、わざとレジスタd1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 */
        stop    #0x2700


**-------------------------------------
**      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）    %d0:結果(最小値)
**-------------------------------------
MINIMUM:
        movem.l %a1/%d1,-(%a7)   /* レジスタの退避（push）（a1,d1の値をスタックに格納する） */
        moveq.l #LENGTH,%d1      /* d1 = LENGTH - 1 */
        subq.w  #1,%d1
        move.w  (%a1),%d0
```

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

        movem.l  (%a7)+,%a1/%d1   /* 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
**--------------------------------------
.section .data
        .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 –t 400 min.s ＜Enter キー＞

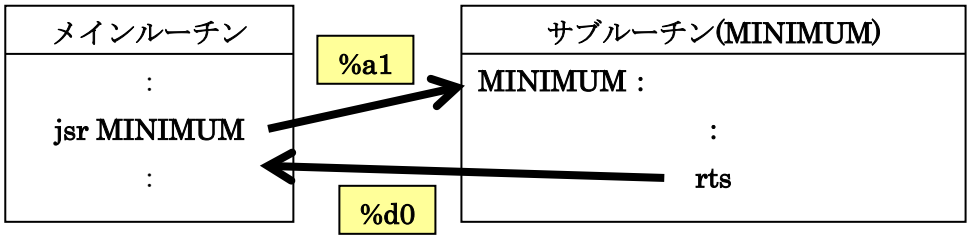
エラーメッセージがなければ、68000 エミュレータで実行してみましょう（実行前に、PC とスタック初期値の設定、ブレークポイントの設定も忘れずにしておくこと）

$ 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 d0 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。

ブレークポイントをメインルーチンの「jsr　MINIMUM」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Ⅰ）。

サブルーチン終了後に、実行する行のアドレスがスタックに自動的に格納されている

サブルーチンの１行目「 movem.l　 %a1/%d1, -(%a7)」は、「レジスタ a1 と d1 をシステムスタックに退避（push）する」という命令である。この movem.l を使うと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（注意Ⅱ）。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

d1：残りの探索対象データの個数

d0：最小値（結果）

の３つである。これらのうち、レジスタ d0 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 d1 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（注意Ⅲ，Ⅳ）。次頁の図は、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 d1 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 d0 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 d1 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。

（注意）

I. *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。*

II. *MOVEM でレジスタを退避する場合，転送順序は a7, a6, …, a1, a0, d7, d6,…,d1, d0 の順序となっている．逆に復帰する場合の転送順序は退避する場合の逆となっている（命令表 p.274 を参照）つまり、ソースコードに記載したレジスタの順序は影響を与えません。*

III. *PUSH と POP の対応が取れていないと，プログラムは正常に動作しない．*

IV. *サブルーチン初めで，サブルーチン本体の実行で値が変わる可能性のあるレジスタを PUSH する．そして，サブルーチンの終了前に PUSH したレジスタを POP する．この２つの処理は必ず対でおこなうこと．*

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、d1 を復帰（pop）する前のスタックポインタ（a7）が 0x4ff4 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/%d1

(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) 前回の課題２の（１）をサブルーチン化し，以下の計算をせよ．

$$\sum_{k=1}^{12} {}_7P_{x[k]} =$$

但し，x[k]は以下の表で与えられるものとする．

k	1	2	3	4	5	6	7	8	9	10	11	12
x[k]	**7**	**5**	**3**	**2**	**6**	**4**	**5**	**1**	**5**	**1**	**6**	**7**