機械語命令 (スタック・サブルーチン・他)

山本昌志*

2005 年 1 月 31 日

1　前回の復習と本日の学習

1.1　復習

今まで，学習した COMET II の命令は，次の通りである．これを思い出して，本日の学習内容と絡めて，理解を深める必要がある．

1.1.1　アセンブラ命令

開始	START	プログラムの先頭を示し，入口名と実行開始番地を定義
終了	END	プログラムの終わりを明示
領域確保	DS	プログラムで使うメインメモリーを予約
定義	DC	メインメモリーの初期値設定

1.1.2　機械語命令

- データ転送命令

データ転送	LD	メインメモリーやレジスターの内容を汎用レジスタに転送
データ転送	ST	汎用レジスタのデータをメインメモリーへ転送
アドレス転送	LAD	実効アドレスを汎用レジスターへ転送

- 算術，論理演算

算術加算	ADDA	1 語のデータを符号付き整数と見なし，加算を行う.
論理加算	ADDL	1 語のデータを符号無し整数と見なし，加算を行う.
算術減算	SUBA	1 語のデータを符号付き整数と見なし，減算を行う.
算術減算	SUBL	1 語のデータを符号無し整数と見なし，減算を行う.
論理積	AND	1 語のデータのビット毎の論理積の演算を行う.
論理和	OR	1 語のデータのビット毎の論理和の演算を行う.
排他的論理和	XOR	1 語のデータのビット毎の排他的論理和の演算を行う.

*国立秋田工業高等専門学校　電気工学科

- シフト

算術左シフト	SLA	レジスタの内容を符号ビットを除き左にシフト．空きには 0 が入る．
算術右シフト	SRA	レジスタの内容を符号ビットを除き左にシフト．空きには符号ビットが入る
論理左シフト	SLL	1 語全てを，左にシフト．空きには 0 が入る
論理右シフト	SRL	1 語全てを，右にシフト．空きには 0 が入る

- 比較

算術比較	CPA	データは符号付き整数と見なし，比較を行う．
論理比較	CPL	データは符号無し整数と見なし，比較を行う．

- 分岐

正分岐	JPL	SF と ZF がともに 0 の時，指定の実効アドレスに分岐
負分岐	JMI	SF が 1 の時，指定の実効アドレスに分岐
非零分岐	JNZ	ZF が 0 の時，指定の実効アドレスに分岐
零分岐	JZE	ZF が 1 の時，指定の実効アドレスに分岐
オーバーフロー分岐	JOV	OF が 1 の時，指定の実効アドレスに分岐
無条件分岐	JUMP	無条件に，指定の実効アドレスに分岐

1.2 本日の学習内容

教科書 [1] の p.76〜p.82 までである．

1.2.1 スタック操作

スタックの操作は．サブルーチンを呼び出すときにデータを待避させる時に主に使う．

プッシュ	PUSH	スタックにデータを入れる (プッシュする)
ポップ	PUSH	スタックにデータを取り出す (ポップする)

1.2.2 サブルーチン関係

比較とジャンプ命令は，セットで使われることが多く，数の比較を行い，その結果を受けて，処理の実行を変える．FORTRAN や C 言語では，

コール	CALL	サブルーチンを呼び出す (コールする)
リターン	RET	呼び出し元へ復帰する (リターンする)

1.2.3 その他

スーパーバイザーコール	SVC	OS の機能を呼び出す
ノンオペレーション	NOP	なにも実行しない

2 スタック操作

2.1 スタックとは

スタック (stack) を辞書で調べてみると，次のような意味が書かれている．

1. (干し草などの) 大きな山，積みわら (haystack);(物のきちんとした) 積み重ね
2. (図書館などの) 書棚の列，書架；((the 〜s)) (図書館の)(閉架) 書庫
3. (屋上の) 組み合わせ煙突 (chimney stack); 煙突，煙出し
4. 《コンピューター》スタック:最後に入れたデータを最初に取り出せるようにしたデータ構造

もちろん，情報科学の分野で使われるのは最後の意味で，図 1 のようなデータ構造である．データ構造であるから，データを蓄えることと，それを取り出すことができる．スタックの特徴は，最後に入れたデータが一番最初に取り出されることにある．取り出されるデータは，格納されている最新のデータで，最後に入れられたものが最初に取り出されることから，LIFO(last in first out, 後入れ先出し) と呼ばれる．スタックの途中のデータを取り出すことは許されないのである．

スタックにデータを積むことをプッシュ(push) と，スタックからデータを取り出すことをポップ (pup) と呼ぶ．これらの英語の意味は，

push ＜人・物を＞押す，突く

pop ポンという音を立てる，ひょいとやって来る [出て行く]，急にはいる [出る], ひょっこり現れる

である．

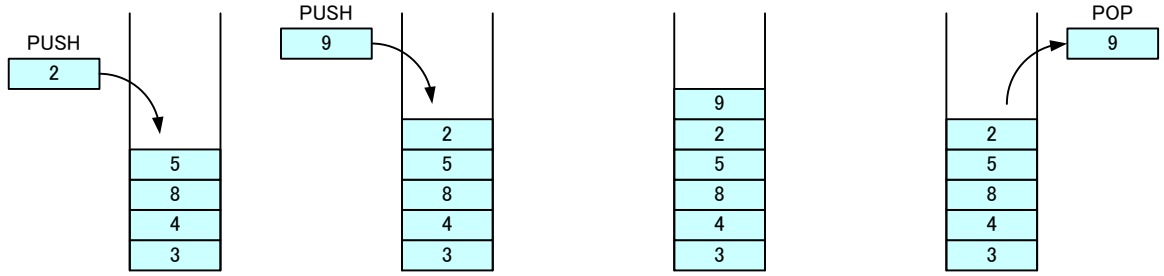

図 1: スタックのイメージ．メモリーへデータの出し入れをする．

COMET II の場合，スタック領域は主記憶 (メインメモリー) のどこかに必要量確保されている．確保されているアドレスはプログラマーは気にすることなく，PUSH と POP の命令を使うことができる．実際にスタック領域の最上位のアドレスは SP(スタックポインター) に格納されており，その操作は OS の仕事である．

2.2 スタックの利用方法

スタックは極めて単純なデータ構造なので，それを実際のコンピューターに実装することは易しい．単純ではあるが，それはかなり頻繁に使われる．もっとも多く使われるのが，サブルーチンを呼び出すときに，一時的にデータを避難させる場合である．実際にはコンパイルした後で出てくるのでプログラマーが直に記述することは少ないが，本当によく使われている．

CASL II のプログラムでもっとも多く使われる場面は，サブルーチンを呼び出すときにデータを一時的に雛させる場合である．これについては，後ほど，実際のプログラムで学習することになる．

2.3 スタックを操作する命令

2.3.1 プッシュ(PUSH)

内容

命令語	PUSH
語源	**POP**　(push:押す)
役割	スタック領域にデータを格納する．
書式	[ラベル]　PUSH　*adr*[,x]
動作	SP の値を 1 減らす．そして，スタック領域へアドレスを格納する．
フラグレジスタ	変化無し．

使用例

```
PUSH    DATA        ;DATA が示すアドレスをスタック領域に格納
PUSH    23          ;23 をスタック領域に格納
PUSH    0,GR1       ;GR1 の内容をスタック領域に格納
```

2.3.2 ポップ (POP)

内容

命令語	POP
語源	**POP**　(pop:出る)
役割	スタック領域からデータを取り出す.
書式	[ラベル]　POP　r
動作	スタックポインター SP が示しているアドレスの内容を取り出して レジスターに格納する．そして，SP の値を 1 増やす.
フラグレジスタ	変化無し.

使用例

```
PUP     GR1         ; スタック領域の最上段の値を GR1 へ格納
```

3　サブルーチン関係

3.1　サブルーチンとは

大量のパーツからできている自動車も機能別の部品に分けることにより，その構造が分かりやすくなる．例えば，エンジン，トランスミッション，サスペンション等，機能毎に説明されると分かりやすい．一つ一つの部品，例えば，ボルト，ベルト，シャフトの説明をされても自動車の全体は分からない．自動車の仕組みを分かるためには，機能毎に理解する方が断然簡単である．

プログラムも一緒である．それを構成する機能の集まりで理解した方が簡単である．Windows の 4 千万行もあるプログラムの各行を説明されても全く分からない．プログラムを機能毎に分けたものがサブルーチンと呼ばれるものの本質である．要するにプログラムを構成する機能単位の部品だと思えばよい．

図 2 と 3 にサブルーチンを使ったプログラムとそうでないプログラムのイメージを示す．圧倒的に，サブルーチンを使った図 3 の方が見通しがよい．実際に長いプログラムを書く場合，サブルーチンを使わないと不可能である．

具体的に，プログラムを機能別のサブルーチンに分けると，以下のようなメリットがある．

- プログラムの作成が容易になる.
- プログラムの内容が分かりやすくなる.
- サブルーチンの再利用が容易になる.

ちろん CASL II にもこれが当てはまり，出来るだけ機能毎にプログラムを書くようにすべきである．要するにサブルーチンを使えということである.

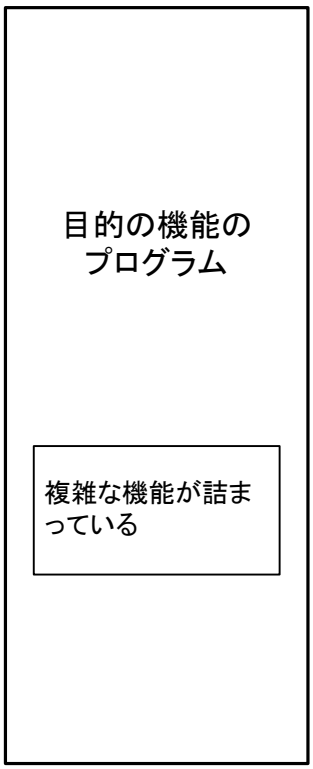

図 2: メインルーチンだけのプログラム

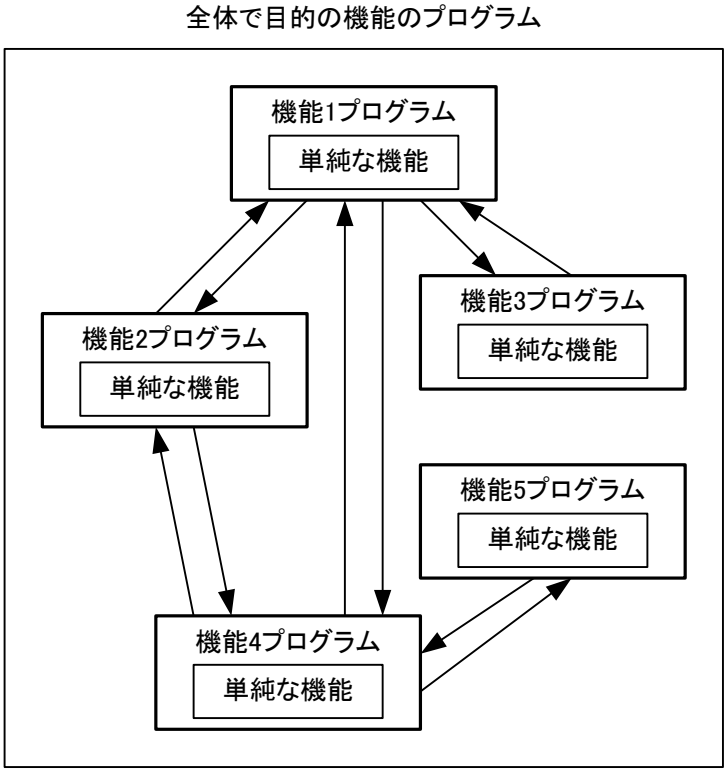

図 3: 機能毎 (サブルーチン) に分割されたプログラム．矢印は，データの流れをあらわす．

3.2 サブルーチンを操作する命令

サブルーチンを呼び出す命令 CALL とサブルーチンから戻る命令 RET は，ペアで使われる．アセンブラ命令 START で指定される最初に実行されるルーチン[1]も OS がコールする．プログラム中ではあらわに現れないが，やはり CALL と RET のペアになっているのである．

3.2.1 コール (CALL)

内容

[1] メインルーチン

命令語　　CALL
語源　　**CALL**　(call:呼ぶ)
役割　　サブルーチンへ制御を移す.
書式　　[ラベル]　CALL　*adr*[,x]
動作　　プログラムレジスタ PR の値を指定のアドレスに変える．現在のアドレスはスタック領域に待避させる.
フラグレジスタ　　変化無し.

使用例

```
CALL    KAKEZAN          ;アドレス KAKEZAN を呼び出し
```

3.2.2 リターン (RET)

内容

命令語　　RET
語源　　**RET**urn　(return:帰る，戻る)
役割　　サブルーチンから呼び出し元へ制御が戻る.
書式　　[ラベル]　RET
動作　　スタックポインター (SP) が示すアドレスの値をプログラムレジスター (PR) にセットする．そして，PR の値を，1 増加させる.
フラグレジスタ　　変化無し.

使用例

```
RET          ;呼び出し元へ戻る
```

4 その他の命令

4.0.1 スーパーバイザーコール (SVC)

内容

命令語　　SVC
語源　　**S**uper**V**isor **C**all　(supervisor:監督者，管理人)
役割　　OS の機能を呼び出す.
書式　　[ラベル]　SVC　*adr*[,x]
動作　　指定されたアドレスをコールする.
フラグレジスタ　　アセンブラーに依存.

使用例

```
        SVC   #A000         ;A000 へジャンプ
```

OS の機能 (サブルーチンみたいなもの) へ制御を移すので，実際の動作は OS に依存する．諸君はこの機能を使うことはないだろう．諸君が使っているシミュレーターの機能をよく調べて使うなら別である．

4.0.2 ノーオペレーション (NOP)

内容

命令語	NOP
語源	**N**o **OP**eration　(operatin:操作，演算)
役割	何もしない．
書式	[ラベル]　NOP
動作	プログラムレジスター (PR) の値を 1 増加させる．
フラグレジスタ	変化無し．

使用例

```
      L1   NOP         ; 何もしないが，ラベルは有効
```

この命令は，なにも動作はしない．しかし，ラベルは有効であるため，ラベルを付けたい場合に使われる．

参考文献

[1] 東田幸樹, 山本芳人, 広瀬啓雄. アセンブラ言語 CASL II. 工学図書 (株), 2002 年.