附録4　アセンブラCASLの仕様

名　称
ハードウェア　COMET
アセンブラ言語　CASL

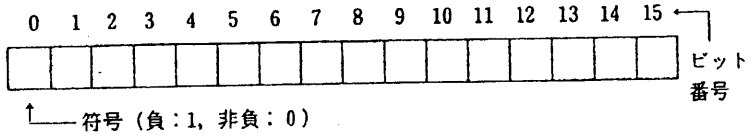
〔アセンブラ言語の仕様〕

1. ハードウェア COMET の仕様

1.1 処 理 装 置

(1) COMET は，1語16ビットの計算機であって，アクセスできるアドレスは0番地から65535番地までである。

(2) 1語のビット構成は，次の通りである。



(3) 数値は，16ビットの2進数により表現する。負数は，2の補数表記である。

(4) 制御方式は逐次制御で，2語長の命令語をもつ。

(5) レジスタとして，GR（16ビット），PC（16ビット），FR（2ビット）をもつ。

GR（汎用レジスタ，General register）は5個あり，汎用レジスタ0番から4番までとする。この5個のレジスタは，算術，論理，比較，シフト演算などに用いる。このうち，1番から4番までのレジスタは，指標レジスタ（Index register）としても用いる。また4番のレジスタは，さらにスタックポインタ（Stack pointer）として用いる。

スタックポインタは，スタックの最上段（Stack top）のアドレスを保持しているレジスタである。

PC（プログラムカウンタ，Program counter）は，実行中の命令語の先頭アドレスを保持し，命令の実行が終わると，次に実行する命令語の先頭アドレスが設定される。一般に，命令の実行が終わるとPCに2がアドレス加算（注）され，分岐，コール，リターン命令の場合は，新たに分岐先のアドレスが設定される。

(注) アドレス加算：被演算データを符号のない数値とみなし，その和を65536で割った剰余を値（和の下位16ビット）とする。アドレス減算もこれに準じた定義とする。

FR（フラグレジスタ，Flag register）は，ロードアドレス命令および算術，論理，シフトの各演算命令の実行の結果，GRに設定されたデータが，負，零，正のいずれであるかの情報，または比較演算命令の実行により得られた，2数間の大小関係の情報を保持する。

すなわち，実行結果により，FRは次の表の通り設定される（ただし，比較については，「1.2(4)の

比較演算命令」参照)。

右表で，負は GR の符号ビットが 1，零は GR の全ビットが 0，正は符号ビットが 0 で，かつ零でないデータをいう。

	GR に設定されたデータ		
	負	零	正
FR の値	10	01	00

FR の第 1 (左端の) ビットは GR の符号を示し，第 2 ビットは GR が零か否かを示す。

FR の値は，条件付き分岐命令で参照する。

その他の命令の実行によって，FR の値は変更されない。

(6) 命令語の構成

2 語長の命令語をもつ。その構成については定義しない。

(7) 命令の表記

各命令の説明には，次の表記を用いる。

GR	GR の値を番号とする汎用レジスタ (ただし，0≦GR≦4)。
XR	XR の値を番号とする指標レジスタ (ただし，1≦XR≦4)。
SP	スタックポインタ (汎用レジスタ 4 番)。
adr	ラベル名 (ラベル名に対応する番地を示す) または 10 進定数 (ただし，−32768≦adr≦65535 とする。adr はアドレスとして 0〜65535 の値をもつが，32768〜65535 の値を負の 10 進数で記述することもできる)。
実効アドレス	adr と XR の内容とのアドレス加算値またはその値が示す番地。
(X)	X番地の内容，Xがレジスタの場合はレジスタの内容。
[ ]	[ ] に囲まれた部分は省略可能であることを示す。XR を省略した場合は指標レジスタによる修飾を行わない。

1.2 命　　令

命令およびその機能を示す。命令は，アセンブラの表記法で記述する。

命　令	書き方		命 令 の 説 明
	命 令 コード	オペランド	

(1) ロード，ストア命令

ロード LoaD	LD　GR, adr [, XR]	(実効アドレス) を GR に設定する
ストア STore	ST　GR, adr [, XR]	(GR) を実効アドレスが示す番地に格納する

(2) ロードアドレス命令

ロードアドレス Load Effective Address	LEA　GR, adr [, XR]	実効アドレスを GR に設定する GR の値により FR を設定する

命　令	書き方		命令の説明
	命令コード	オペランド	

(3) 算術，論理演算命令

算術加算 ADD arithmetic	ADD　GR, adr [, XR]	(GR) と (実効アドレス) に指定した演算を施し，結果を GR に設定する なお，算術減算では (GR) から (実効アドレス) を減算する 演算結果により FR を設定する
算術減算 SUBtract arithmetic	SUB　GR, adr [, XR]	
論理積 AND	AND　GR, adr [, XR]	
論理和 OR	OR　GR, adr [, XR]	
排他的論理和 Exclusive OR	EOR　GR, adr [, XR]	

(4) 比較演算命令

算術比較 ComPare Arithmetic	CPA　GR, adr [, XR]	(GR) と (実効アドレス) の算術比較または論理比較を行い，比較結果により FR に次の値を設定する
論理比較 ComPare Logical	CPL　GR, adr [, XR]	

比較結果	FR のビット値
(GR)＞(実効アドレス)	00
(GR)＝(実効アドレス)	01
(GR)＜(実効アドレス)	10

(5) シフト演算命令

算術左シフト Shift Left Arithmetic	SLA　GR, adr [, XR]	(GR)を符号を除き実効アドレスで指定したビット数だけ左または右にシフトする シフトの結果，あいたビット位置には左シフトのときは0，右シフトのときには符号と同じものが入る シフトの結果により FR を設定する
算術右シフト Shift Right Arithmetic	SRA　GR, adr [, XR]	
論理左シフト Shift Left Logical	SLL　GR, adr [, XR]	(GR)を符号を含み実効アドレスで指定したビット数だけ左または右にシフトする シフトの結果，あいたビット位置には0が入る シフトの結果により FR を設定する
論理右シフト Shift Right Logical	SRL　GR, adr [, XR]	

(6) 分岐命令

正分岐 Jump on Plus or Zero	JPZ　adr [, XR]	FR の値により，実効アドレスに分岐する。分岐しないときは次の命令に進む
負分岐 Jump on MInus	JMI　adr [, XR]	
非零分岐 Jump on Non Zero	JNZ　adr [, XR]	
零分岐 Jump on ZEro	JZE　adr [, XR]	
無条件分岐 unconditional JuMP	JMP　adr [, XR]	無条件に実効アドレスに分岐する

命令	分岐するときの FR の値
JPZ	00, 01
JMI	10
JNZ	00,10
JZE	01

命　令	書き方		命令の説明
	命令コード	オペランド	

(7) スタック操作命令

プッシュ PUSH effective address	PUSH adr [, XR]	SP から 1 をアドレス減算した後，実効アドレスを(SP)番地に格納する
ポップ POP up	POP GR	(SP)番地の内容を GR に格納した後，SP に 1 をアドレス加算する

(8) コール，リターン命令

コール CALL subroutine	CALL adr [, XR]	SP から 1 をアドレス減算した後，PC の現在地に 2 をアドレス加算した値を (SP) 番地に格納し，実効アドレスに分岐する
リターン RETurn from subroutine	RET	(SP)番地の内容を取り出した後，SP に 1 をアドレス加算し，先に取り出した内容(番地)に分岐する(取り出した内容を PC に設定する)

2. アセンブラ言語 CASL の仕様

COMET のためのアセンブラ言語は CASL といい，その仕様は次の通りである。

2.1 命令の種類

CASL は 4 種類の擬似命令 START, END, DS, DC, 3 種類のマクロ命令 IN, OUT, EXIT および機械語命令（COMET の命令）から成る。

(1) 擬似命令

擬似命令は，アセンブラの制御，定数の定義，プログラム連結のために必要なデータの生成などを行う。擬似命令の機能は次の表の通りである。

擬似命令	機　　能
START	プログラムの先頭の定義 プログラムの実行開始番地の定義 他のプログラムとの連結のための入口名の定義
END	プログラムの終わりの定義
DS	領域の確保
DC	定数の定義

(2) マクロ命令

マクロ命令は，あらかじめ定義された命令群とオペランドの情報により，目的の機能を果たす命令群を生成する。

CASL で用意されているマクロ命令は，入出力およびプログラムの終了などを行う命令である。CASL では機械語の入出力命令などを定義していないので，このような処理はオペレーティングシステムにまかせる。マクロ命令が現われると，CASL はオペレーティングシステムを呼ぶための命令群を生成する。ただし，生成される命令群の語数は不定とする。

マクロ命令を実行すると，GR の内容は保存されるが，FR の内容は不定となる。

マクロ命令の機能は，右の表の通りである。

マクロ命令	機　　能
IN	入　力
OUT	出　力
EXIT	プログラムの実行終了

(3) 機械語命令

「1.2 命令」で説明した 23 種類の機械語命令がある。

2.2 命令の形式

擬似命令，マクロ命令，機械語命令は，ラベル欄，命令コード欄，オペランド欄，注釈欄をもつ。各欄は次の通り定義する。

ラベル欄　　第 1 文字からラベルの文字数分（最大 6 文字）。

命令コード欄　① ラベルを付けないとき　第 2 文字以降任意の文字位置から。

② ラベルを付けたとき　ラベルのあと少なくとも一つの空白をおいたあと，任意の文字位置から。

オペランド欄　命令コードのあと少なくとも一つの空白をおいたあと，第 72 文字までとする。次の

行に継続することはできない。

注釈欄　　行中にセミコロン（；）があると，それ以降の終わりまで注釈として扱う（ただし，DC 命令の文字列中の；を除く）。

なお，第1文字位置に；がある場合，または；の前に空白しかない場合は，その行全体を注釈として扱う。

注釈欄には，処理系で許す任意の文字を書くことができる。

CASL の各命令の式は次の通りである。

ラベル	命令コード	オペランド	読み方
label	START	[実行開始番地]	START
空　白	END	空　白	END
[label]	DC	定　数	Define Constant
[label]	DS	領域の語数	Define Storage
[label]	IN	入力領域，入力文字長	INput
[label]	OUT	出力領域，出力文字長	OUTput
[label]	EXIT	空　白	EXIT
[label]	機械語命令（「1.2 命令」参照）		

空白で示した欄は，記入してはならない。

(1) レジスタの指定

機械語命令のオペランド欄のレジスタの指定は，汎用レジスタ番号に対応する 0～4 の数字で行うものとするが，次の記号で指定することもできる。

レジスタの記号による指定	汎用レジスタ番号
GR0	0
GR1	1
GR2	2
GR3	3
GR4	4

(2) ラベル欄

ラベル欄の label は，ラベルである。ラベルは 6 文字以内で，先頭の文字は英大文字でなければならない。以降の文字は英大文字，数字のいずれでもよい。DC, DS, IN, OUT, EXIT および機械語の命令に付けられたラベルは，その領域または命令語（マクロ命令のときは命令群）の先頭の語のアドレスを示す。

START 命令に付けられたラベルは，別のプログラムから入口名として参照できる。

2.3 擬 似 命 令

(1) START　　[実行開始番地]

プログラムの先頭を示す。すなわち，プログラムの最初にこれを書かなければならない。

実行開始番地は，このプログラム内で定義されているラベル名とし，このプログラムの実行開始番地を指定する。省略した場合は，プログラムの先頭から実行を開始する。

(2) END

プログラムの終わりを示す。プログラムの最後にこれを書かなければならない。

(3) DC　　｜定数

定数で指定した定数データを格納する。定数には，10進定数，16進定数，文字定数，アドレス定数の4種類がある。

定数の種類	書き方		説明
	命令コード	オペランド	
10進定数	DC	n	nで指定した10進数値を1語の2進数データとして格納する。ただし，nが−32768～32767の範囲にないときは，その下位16ビットを格納する
16進定数	DC	#h	hは4桁の16進数(16進数字は0～9，A～F)とする。hで指定した16進数値を1語の2進数データとして格納する(0000≦h≦FFFF)
文字定数	DC	'文字列'	文字列の左端から1文字ずつ，連続する語の下位8ビットに，文字データを格納する。すなわち最初の文字は第1語の8～15ビットに，2番目の文字は第2語の第8～15ビットに，……と順次，文字列の文字数分文字データを格納する。各語の0～7ビットには0のビットが入る。文字列には，間隔および任意の図形文字(「1.3文字の組」参照)を書くことができる。ただし，アポストロフィ(')は書けない 文字列の長さは0（文字列が空）であってはならない
アドレス定数	DC	ラベル名	ラベル名に対応するアドレス値を1語の2進数データとして格納する ラベル名がこのプログラム内で定義されていない場合，アセンブラはアドレスの決定を保留し，アドレスの決定をオペレーティングシステムにまかせる(「3.1(5)未定義ラベル」参照)

(4) DS　　｜領域の語数

指定した語数の領域を確保する。

領域の語数は，10進定数(≧0)で指定する。領域の語数を0とした場合，領域は確保しない。ただし，ラベル欄のラベル名は有効である。

2.4 マクロ命令

(1) IN　　｜入力領域，入力文字長

あらかじめ割り当てた入力領域に1レコードのデータ（文字データ）を入力する（入力装置の割当てについては，「3.1(3)入出力装置の割当て」参照)。

オペランド欄の入力領域，入力文字長には，ラベル名を書く。

入力領域は，80語長（80文字分）の作業域のラベル名とし，この領域に（先頭番地から）1文字を1語に対応させて，順次入力する。各語の第0～7ビットには，0のビットを格納する（DC命令の文字定数に同じ)。入力したデータの長さ（入力レコードの文字数）を，入力文字長で指定した領域（1語）に2進データの形で格納する。レコードの区切符号（鍵盤入力のときの復帰符号など）は，格納しない。

入力データが80語に満たない場合，入力領域の残りの部分は実行前のデータを保持する。入力データが80文字を越える場合，以降の文字は無視する。

次の場合，入力文字長に0または−1を格納する。

0：空のレコードの入力（タイプライタで復帰符号だけが入力されたときなど）

−1：EOF（end of file）が検出された（カード読取装置など）

(2) OUT　　| 出力領域，出力文字長

出力領域に格納されているデータ（文字データ）をあらかじめ割り当てた出力装置に1レコードとして出力する（出力装置の割当てについては，「3.1 (3)入出力装置の割当て」参照)。

オペランド欄の出力領域，出力文字長にはラベル名を書く。

出力領域は出力しようとするデータが1文字1語で（DC 命令の文字定数に同じ，ただし第0〜7ビットの値は0でなくてもよい）格納されている領域のラベル名とする。出力文字長は，1レコードとして出力しようとするデータの長さ（文字数）を2進データの形で格納している領域（1語）のラベル名とする。

出力の際，レコードの区切符号（タイプライタ出力のときの復帰符号など）が必要な場合には，オペレーティングシステムが自動的に挿入出力する。また，各語の第0〜7ビットの削除もオペレーティングシステムが行う。

(3) EXIT

プログラムの実行を終了する（制御をオペレーティングシステムに戻す)。

2.5 命令，領域の相対位置

アセンブラにより生成される命令語や領域の相対位置は，アセンブラ言語での記述順序とする。また生成された命令語，領域は，主記憶上で連続した領域を占める。

3. CASL 利用の手引

3.1 オペレーティングシステム

プログラムの実行に関して，利用者プログラムとオペレーティングシステムとの間に，次の取決めがある。

(1) プログラムの起動

プログラムはオペレーティングシステムにより起動される。プログラムが格納される番地は不定とするが，(2)で述べるスタック領域も含めてプログラムの実行に支障を与えないものとする。

(2) スタック領域

プログラムの起動時に，オペレーティングシステムは，スタック領域を確保し，スタック領域の最後のアドレスに1をアドレス加算した値を SP に設定する。この領域は，プログラムでスタックとして利用される。

スタック領域は，試験問題のプログラムで使用するのに十分な容量が確保されているものとする。

(3) 入出力装置の割当て

IN 命令に対応する入力装置，OUT 命令に対応する出力装置の割当ては，プログラムの実行に先立ってオペレータが行う。入出力装置には，コンソール表示装置，タイプライタ，カード読取装置，カードせん孔装置などがある。

(4) 入出力装置

各種の入出力装置からのデータの入出力は，IN，OUT マクロ命令で行うが，媒体や装置による入出力手続きの違いはすべてオペレーティングシステムが吸収し，システムの標準手続き（異常処理を

含む)，標準形式で行う。したがって，このマクロ命令の使用者は，入出力装置の違いを意識する必要はない。

(5) 未定義ラベル

アセンブラは，機械語命令のオペランド中のラベル，DC 命令のアドレス定数のラベルのうち，そのプログラム内で定義されていないラベルを，他のプログラムの START 命令のラベル（他のプログラムの入口名）と解釈する。

この場合，アセンブラはアドレスの決定を保留し，その決定をオペレーティングシステムにまかせる。オペレーティングシステムは，実行に先立って他のプログラムの START 命令のラベルとの結合処理を行いアドレスを決定する（プログラムの連結)。

3.2 未定義事項

プログラムの実行等に関し，本仕様で定義しない事項は，処理系によるものとする。

〔参考資料〕

参考資料は，COMET の理解を助けるため，または COMET の処理系作成者の便宜のための資料である。したがって，COMET，CASL の仕様に影響を与えるものではない。

1. 命令語の構成

命令語の構成は定義しないが，次のような構成を想定する。

(OP の数値は 16 進表示)

0　4　8　12　16　　31 ←ビット番号

第1語				第2語	命令語とアセンブラとの対応		
OP		GR	XR	adr	アセンブラ命令		意　味
主 OP	副 OP						
0	0				未使用		
1	0				LD	GR, adr, XR	load
	1				ST	GR, adr, XR	store
	2				LEA	GR, adr, XR	load effective address
2	0				ADD	GR, adr, XR	add
	1				SUB	GR, adr, XR	subtract
3	0				AND	GR, adr, XR	and
	1				OR	GR, adr, XR	or
	2				EOR	GR, adr, XR	exclusive or
4	0				CPA	GR, adr, XR	compare arithmetic
	1				CPL	GR, adr, XR	compare logical
5	0				SLA	GR, adr, XR	shift left arithmetic
	1				SRA	GR, adr, XR	shift right arithmetic
	2				SLL	GR, adr, XR	shift left logical
	3				SRL	GR, adr, XR	shift right logical
6	0	— —			JPZ	adr, XR	jump on plus or zero
	1	— —			JMI	adr, XR	jump on minus
	2	— —			JNZ	adr, XR	jump on non zero
	3	— —			JZE	adr, XR	jump on zero
	4	— —			JMP	adr, XR	unconditional jump
7	0	— —			PUSH	adr, XR	push effective address
	1			— — — — — — — — — —	POP	GR	pop up

8	0 1	―― ――		――――――――	CALL　adr, XR RET	call subroutine return from subroutine
9 \ F					その他の命令 (入出力命令など)	

2. マクロ命令

マクロ命令が生成する命令群は定義しない（語数不定）が，次の例のような命令群を生成することを想定する。

[例1]　　IN

```
label     IN      ibuf, len
          ↓マクロ生成
label     PUSH    ibuf          ; push first parameter address
          PUSH    len           ; push second parameter address
          CALL    XXXIN         ; call subroutine
          LEA     GR4, 2, GR4   ; restore SP
```

(注)XXXIN は，入力を行うシステムプログラムのラベル名（入口名）である。

[例2]　　EXIT

```
label     EXIT
          ↓マクロ生成
label     JMP     XXXEXT
```

(注)XXXEXT は，利用者プログラムの終了処理を行うシステムプログラムのラベル名（入口名）である。

3. 主プログラムと副プログラム

CASL アセンブラには，主プログラムと副プログラムの区別はない。いいかえれば，アセンブラが生成する目的プログラムは，両者とも同じ形式である。

主プログラムの指定は，オペレーティングシステムによるプログラムの連結時に行うものと想定する。

〔プログラム例1〕

プログラム MDUMP は，指定された主記憶領域のデータを 16 進表示で，編集出力する副プログラムである。MDUMP は，16 進変換を行うために副プログラム HCONV を利用する。

MDUMP は，指定された領域の語の内容を，1行に 8 語ずつ変換出力する。ただし，変換される語が格納されている番地の上位 13 ビットが等しいものを，1行（1レコード）に編集して出力する。

〔変換する領域を，#1005 番地から#102E 番地にしたときの出力例〕

```
ADDRESS   0/8   1/9   2/A   3/B   4/C   5/D   6/E   7/F
 #1000                                  0045  0058  0041
 #1008    004D  0049  004E  0041  0054  0049  004F  004E
 #1010    0020  004F  0046  0020  0044  0041  0054  0041
 #1018    002D  0050  0052  004F  0043  0045  0053  0053
```

```
#1020    0049  004E  0047  0020  0054  0045  0043  0048
#1028    004E  0049  0043  0049  0041  004E  0053

MDUMP  START ; 機能：指定された領域の内容を16進数字に変換して出力する。
;              入力：GR1  変換する領域の先頭番地
;                    GR2  変換する領域の最後の番地
;              注意：先頭番地＞最後の番地の場合，最初の1語のみ処理する。
;                    このプログラムを実行すると，GR0, FR の内容は不定となる。
;-------------------------------------------------------------------
               PUSH  0, GR1          ；レジスタの退避
               PUSH  0, GR2
               PUSH  0, GR3
               LEA   GR3, -1, GR1    ；GR3 ←変換する領域の先頭番地－1
               OUT   MIDASI, LEN50   ；見出しの出力
               :
               : 1行の編集処理を行うループ
               :
LOOP0          LD    GR0, SPACE      ；GR0 7空白を設定 ──────┐
               LEA   GR2, 38         ；                      │
LOOP1          ST    GR0, CAREA, GR2 ；空白を格納            │ 変換結果を格納
               LEA   GR2, -1, GR2    ；GR2 ← GR2－1          │ する39語の領域
               JPZ   LOOP1           ；GR2<0 になるまでくり返す┘ を空白でクリア
               :
               : 1語の編集処理を行うループ   変換結果の格納番地を計算する
               :
LOOP2          LEA   GR3, 1, GR3     ；変換する語の番地を1進める
               LEA   GR2, 0, GR3     ；変換する語の番地を GR2 に設定
               AND   GR2, MASK7      ；変換する語の番地の下位3ビットを取り出す
               ST    GR2, WRK
               SLA   GR2, 2          ；4倍する
               ADD   GR2, WRK        ；結果として，GR2 ←下位3ビット×5
               LEA   GR1, CAREA, GR2 ；格納番地＝CAREA の番地＋下位3ビット×5
               :
               :                     16進変換，および1行として出力するかの判定
               :
               LD    GR0, 0, GR3     ；変換する語のデータを GR0 に設定
               CALL  HCONV           ；データの16進変換
               LEA   GR2, -35, GR2   ；下位3ビットは7 (GR2 が35)か？
               JZE   PRINT           ；7のとき行の終わり，出力処理へ
               CPL   GR3, 1, GR4     ；領域の最後か？(スタック内の GR2 と比較)
```

```
        JM1   LOOP2          ;次の語の変換へ
;― LOOP2 終わり ―
;
;                            番地の16進変換，および出力処理
;
PRINT   LEA   GR0, 0, GR3    ;最後に変換した語の番地を取り出す
        AND   GR0, FFF8      ;下位3ビットを0にする
        LEA   GR1, AAREA     ;変換結果の格納番地をセット
        CALL  HCONV          ;番地の16進変換
        OUT   OBUF, LEN50    ;編集結果出力
        CPL   GR3, 1, GR4    ;領域の最後か？
        JMI   LOOP0          ;次の行の編集処理へ
;  ― LOOP0 終わり ―
        POP   GR3            ;レジスタの復元
        POP   GR2
        POP   GR1
        RET
OBUF    DC    '      #'      ;出力領域の最初の4文字――――――――出
AAREA   DC    '      '       ;番地を16進数に変換した値の格納域　　力
CAREA   DS    39             ;変換結果格納領域（8語分）――――――域
FFF8    DC    #FFF8          ;下位3ビットをクリアするためのマスク
LEN50   DC    50
MASK7   DC    7              ;下位3ビットを取り出すためのマスク
MIDASI  DC    '   ADDRESS  0/8  1/9  2/A  3/B  4/C  5/D  6/E  7/F'
SPACE   DC    ' '
WRK     DS    1              ;変換する語の番地の下位3ビット
        END
HCONV   START ;  機能  :  1語の2進数を4けたの16進数字に変換する
;
;                入力  :  GR0  変換する2進数データ（1語）
;                         GR1  変換結果を格納する4語長の領域の先頭番地
;                注意  :  この副プログラムを実行するとFRは不定となる
;-------------------------------------------------------------------
        PUSH  0, GR1         ;レジスタの退避
        PUSH  0, GR2
        PUSH  0, GR3
;
        ST    GR0, DATA      ;変換するデータの設定
        LEA   GR2, 12        ;シフトするビット数を12に設定
```

```
        :
        :   データから4ビットずつ取り出し，変換する
        :
LOOP    LD      GR3, DATA          ；変換するデータの設定
        SRL     GR3, 0, GR2        ；シフトするビット数は12,8,4,0と変化する
        AND     GR3, F             ；GR3に変換対象となる4ビットを取り出す
        LD      GR3, HEXTBL, GR3   ；テーブルをひく
        ST      GR3, 0, GR1        ；変換した16進数字を格納
        LEA     GR1, 1, GR1        ；格納番地の更新
        LEA     GR2, -4, GR2       ；シフトするビット数から4減算
        JPZ     LOOP               ；ビット数が負になるまでくり返す（4回）
        :
        POP     GR3                ；レジスタの復元
        POP     GR2
        POP     GR1
        RET
        :
DATA    DS      1                  ；変換するデータ
F       DC      #000F              ；4ビット取り出すためのマスク
HEXTBL  DC      '0123456789ABCDEF' ；16進数字への変換テーブル
        END
```

〔プログラム例2〕

フィボナッチ数を求める副プログラム。

フィボナッチ数列は，最初の2項が，$a_1=0$，$a_2=1$で始まり，以降は順次，反復公式$a_n=a_{n-1}+a_{n-2}$により定義される数列である。

入力：GR1　何番目のフィボナッチ数を求めたいかをGR1（GR1の値をnとする）で与える。このプログラムでは，$1 \leqq n \leqq 24$でなければ，結果を保証しない。

出力：GR2　フィボナッチ数列の第n項の値。

```
FNUMB   START
        CPA     GR1, CONST3
        JPZ     NEXT               ；GR1≧3 なら NEXT へ
        LEA     GR2, -1, GR1       ；GR2←GR1 −1
        RET
NEXT    PUSH    0, GR1             ；GR1 保存
        LEA     GR1, -1, GR1       ；
        CALL    FNUMB              ；1つ前を求める
        PUSH    0, GR2             ；GR2 保存
```

```
        LEA     GR1, -1, GR1    ; 2つ前を求める
        CALL    FNUMB           ;
        ST      GR2, WRK        ;
        POP     GR2             ; GR2 復元 an=an-1+an-2
        ADD     GR2, WRK        ;
        POP     GR1             ; GR1 復元
        RET
CONST3  DC      3
WRK     DS      1
        END
```