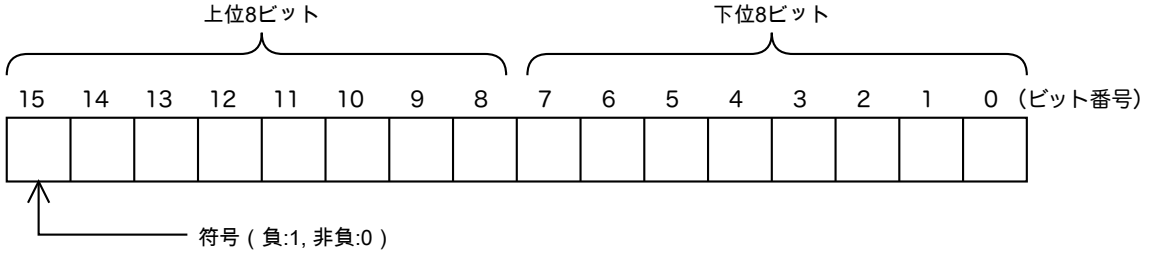
情報処理技術者試験 アセンブラ言語の仕様

「試験で使用する情報処理用語・プログラム言語など Ver 2.2（平成 24 年 5 月 22 日） 別紙 1」より

1 システム COMET II の仕様

1.1 ハードウェアの仕様

1. 1 語は 16 ビットで，そのビット構成は，次のとおりである。



2. 主記憶の容量は 65536 語で，そのアドレスは 0〜65535 番地である。
3. 数値は，16 ビットの 2 進数で表現する。負数は，2 の補数で表現する。
4. 制御方式は逐次制御で，命令語は 1 語長又は 2 語長である。
5. レジスタとして，GR（16 ビット），SP（16 ビット），PR（16 ビット），FR（3 ビット）の 4 種類がある。

   GR（汎用レジスタ，General Register）は，GR0〜GR7の 8 個があり，算術，論理，比較，シフトなどの演算に用いる。このうち，GR1〜GR7のレジスタは，指標レジスタ（index register）としてアドレスの修飾にも用いる。

   SP（スタックポインタ，Stack Pointer）は，スタックの最上段のアドレスを保持している。

   PR（プログラムレジスタ，Program Register）は，次に実行すべき命令語の先頭アドレスを保持している。

   FR（フラグレジスタ，Flag Register）は，OF（Overflow Flag），SF（Sign Flag），ZF（Zero Flag）と呼ぶ 3 個のビットからなり，演算命令などの実行によって次の値が設定される。これらの値は，条件付き分岐命令で参照される。

   OF　算術演算命令の場合は，演算結果が-32768〜32767 に収まらなくなったとき 1 になり，それ以外のとき 0 になる。論理演算命令の場合は，演算結果が 0〜65535 に収まらなくなったとき 1 になり，それ以外のとき 0 になる。

   SF　演算結果の符号が負（ビット番号 15 が 1）のとき 1，それ以外のとき 0 になる。

   ZF　演算結果が零（全部のビットが 0）のとき 1，それ以外のとき 0 になる。
6. 論理加算又は論理減算は，被演算データを符号のない数値とみなして，加算又は減算する。

1.2 命令

命令の形式及びその機能を示す。ここで，一つの命令コードに対し 2 種類のオペランドがある場合，上段はレジスタ間の命令，下段はレジスタと主記憶間の命令を表す。

命令	書き方 命令コード　オペランド	命令の説明	FRの設定

1.2.1 ロード，ストア，ロードアドレス命令

ロード LoaD	LD	r1,r2	r1 ← (r2)	○*1
		r,adr[,x]	r ← (実効アドレス)	
ストア STore	ST	r,adr[,x]	実効アドレス ← (r)	—
ロードアドレス Load ADdress	LAD	r,adr[,x]	r ← 実効アドレス	

1.2.2 算術，論理演算命令

算術加算 ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr[,x]	r ← (r) + (実効アドレス)	
論理加算 ADD Logical	ADDL	r1,r2	r1 ← (r1) +L (r2)	
		r,adr[,x]	r ← (r) +L (実効アドレス)	
算術減算 SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) - (r2)	
		r,adr[,x]	r ← (r) - (実効アドレス)	
論理減算 SUBtract Logical	SUBL	r1,r2	r1 ← (r1) -L (r2)	
		r,adr[,x]	r ← (r) -L (実効アドレス)	
論理積 AND	AND	r1,r2	r1 ← (r1) AND (r2)	○*1
		r,adr[,x]	r ← (r) AND (実効アドレス)	
論理和 OR	OR	r1,r2	r1 ← (r1) OR (r2)	
		r,adr[,x]	r ← (r) OR (実効アドレス)	
排他的論理和 eXclusive OR	XOR	r1,r2	r1 ← (r1) XOR (r2)	
		r,adr[,x]	r ← (r) XOR (実効アドレス)	

1.2.3 比較演算命令

算術比較 ComPare Arithmetic	CPA	r1,r2	(r1)と(r2)，又は(r)と(実効アドレス)の算術比較又は論理比較を行い，比較結果によって，FRに次の値を設定する。	○*1
		r,adr[,x]		
論理比較 ComPare Logical	CPL	r1,r2		
		r,adr[,x]		

比較結果	FRの値 SF	ZF
(r1) > (r2)	0	0
(r) > (実効アドレス)		
(r1) = (r2)	0	1
(r) = (実効アドレス)		
(r1) < (r2)	1	0
(r) < (実効アドレス)		

1.2.4 シフト演算命令

算術左シフト Shift Left Arithmetic	SLA	r,adr[,x]	符号を除き(r)を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には，左シフトのときは0，右シフトのときは符号と同じものが入る。	○*2
算術右シフト Shift Right Arithmetic	SRA	r,adr[,x]		
論理左シフト Shift Left Logical	SLL	r,adr[,x]	符号を含み(r)を実効アドレスで指定したビット数だけ左又は右にシフトする。 シフトの結果，空いたビット位置には，0が入る。	
論理右シフト Shift Right Logical	SRL	r,adr[,x]		

1.2.5 分岐命令

<table>
<tr><td>正分岐
Jump on Plus</td><td>JPL adr[,x]</td><td rowspan="6">FRの値によって，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。
<table><tr><td rowspan="2">命令</td><td colspan="3">分岐するときのFRの値</td></tr><tr><td>OF</td><td>SF</td><td>ZF</td></tr><tr><td>JPL</td><td></td><td>0</td><td>0</td></tr><tr><td>JMI</td><td></td><td>1</td><td></td></tr><tr><td>JNZ</td><td></td><td></td><td>0</td></tr><tr><td>JZE</td><td></td><td></td><td>1</td></tr><tr><td>JOV</td><td>1</td><td></td><td></td></tr></table></td><td rowspan="6">—</td></tr>
<tr><td>負分岐
Jump on MINUS</td><td>JMI adr[,x]</td></tr>
<tr><td>非零分岐
Jump on Non Zero</td><td>JNZ adr[,x]</td></tr>
<tr><td>零分岐
Jump on ZEro</td><td>JZE adr[,x]</td></tr>
<tr><td>オーバーフロー分岐
Jump on OVerflow</td><td>JOV adr[,x]</td></tr>
<tr><td>無条件分岐
unconditional Jump</td><td>JUMP adr[,x]</td><td>無条件に実効アドレスに分岐する。</td></tr>
</table>

1.2.6 スタック操作命令

<table>
<tr><td>プッシュ
PUSH</td><td>PUSH adr[,x]</td><td>SP ← (SP) -L 1,
(SP)←(実効アドレス)</td><td rowspan="2">—</td></tr>
<tr><td>ポップ
POP</td><td>POP r</td><td>r ← ((SP)),
SP ← (SP) +L 1</td></tr>
</table>

1.2.7 コール，リターン命令

<table>
<tr><td>コール
CALL subroutine</td><td>CALL adr[,x]</td><td rowspan="2">SP ← (SP) -L 1,
(SP)←(PR),
PR ← 実効アドレス</td><td rowspan="2">—</td></tr>
<tr><td>リターン
RETurn form subroutine</td><td>RET</td></tr>
<tr><td>リターン
RETurn form subroutine</td><td>RET</td><td>PR ← ((SP)),
SP ← (SP) +L 1</td><td>—</td></tr>
</table>

1.2.8 その他

<table>
<tr><td>スーパーバイザコール
SuperVisor CALL</td><td>SVC adr[,x]</td><td>実効アドレスを引数として割出しを行う。
実行後のGRとFRは不定となる。</td><td rowspan="2">—</td></tr>
<tr><td>ノーオペレーション
No operation</td><td>NOP</td><td>何もしない。</td></tr>
</table>

注

r, r1, r2　いずれも GRを示す。指定できる GRは GR0〜GR7

adr　アドレスを示す。指定できる値の範囲は 0〜65535

x　指標レジスタとして用いる GRを示す。指定できる GRは GR1〜GR7

[ ]　[ ]内の指定は省略できることを示す。

( )　( )内のレジスタ又はアドレスに格納されている内容を示す。

実効アドレス
　adrと xの内容との論理加算値又はその値が示す番地

←　演算結果を，左辺のレジスタ又はアドレスに格納することを示す。

+L, -L　論理加算，論理減算を示す。

FRの設定

○：設定されることを示す。

○*1: 設定されることを示す。ただし、OFには 0 が設定される。

○*2: 設定されることを示す。ただし、OFにはレジスタから最後に送り出されたビットの値が設定される。

—：実行前の値が保持されることを示す。

1.3 文字の符号表

1. JIS X 0201 ラテン文字・片仮名用 8 ビット符号で規定する文字の符号表を使用する。
2. 次に符号表の一部を示す。

行＼列	02	03	04	05	06	07
0	間隔	0	@	P	`	p
1	!	1	A	Q	a	q
2	"	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
10	*	:	J	Z	j	z
11	+	;	K	[	k	{
12	,	<	L	\	l	|
13	-	=	M	]	m	}
14	.	>	N	^	n	~
15	/	?	O	_	o	

1 文字は 8 ビットからなり，上位 4 ビットを列で，下位 4 ビットを行で示す。 例えば，間隔，4，H，\のビット構成は，16 進表示で，それぞれ 20，34，48，5C である。ビット構成が 21〜7E（及び表では省略している A1〜DF）に対応する文字を図形文字という。 図形文字は，表示（印刷）装置で，文字として表示（印字）できる。

3. この表にない文字とそのビット構成が必要な場合は，問題中で与える。

2 アセンブラ言語CASL IIの仕様

2.1 言語の仕様

1. CASL IIは，COMET IIのためのアセンブラ言語である。
2. プログラムは，命令行および注釈行からなる。
3. 1命令は1命令行で記述し，次の行へ継続できない。
4. 命令行および注釈行は，次に示す記述の形式で，行の1文字目から記述する。

行の種類		記述の形式
命令行	オペランドあり	[ラベル]{空白}{命令コード}{空白}{オペランド}[{空白}[コメント]]
	オペランドなし	[ラベル][空白][命令コード][{空白}[{;}[コメント]]]
注釈行		[空白]{;}[コメント]

注

[ ] ［ ］内の指定が省略できることを示す。

{ } { } 内の指定が必須であることを示す。

ラベル その命令の（先頭の語の）アドレスを他の命令やプログラムから参照するための名前である。 長さは1〜8文字で，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，英大文字又は数字のいずれでもよい。 なお，予約語であるGR0〜GR7は，使用できない。

空白 1文字以上の間隔文字の列である。

命令コード
命令ごとに記述の形式が定義されている。

オペランド
命令ごとに記述の形式が定義されている。

コメント 覚え書きなどの任意の情報であり，処理系で許す任意の文字を書くことができる。

2.2 命令の種類

命令は，4種類のアセンブラ命令（START，END，DS，DC），4種類のマクロ命令（IN，OUT）および機械語命令（COMET IIの命令）からなる。 その仕様を次に示す。

命令の種類	ラベル	命令コード	オペランド	機能
アセンブラ命令	ラベル	START	[実行開始番地]	プログラムの先頭を定義 プログラムの実行開始番地を定義 他のプログラムで参照する入口名を定義
		END		プログラムの終わりを明示
	[ラベル]	DS	語数	領域を確保
	[ラベル]	DC	定数[,定数]…	定数を定義
マクロ命令	[ラベル]	IN	入力領域,入力文字長領域	入力装置から文字データを入力
	[ラベル]	OUT	出力領域,出力文字長領域	出力装置へ文字データを出力
	[ラベル]	RPUSH		GRの内容をスタックに格納
	[ラベル]	RPOP		スタックの内容をGRに格納
機械語命令	[ラベル]	（「1.2 命令」を参照）		

2.3 アセンブラ命令

アセンブラ命令は，アセンブラの制御などを行う。

1.

START	[実行開始番地]

START命令は，プログラムの先頭を定義する。

実行開始番地は，そのプログラム内で定義されたラベルで指定する。指定がある場合はその番地から，省略した場合は START命令の次の命令から，実行を開始する。

また，この命令につけられたラベルは，他のプログラムから入口名として参照できる。

2.

END	

END命令は，プログラムの終わりを定義する。

3.

DS	語数

DS命令は，指定した語数の領域を確保する。

語数は，10 進定数（≥ 0）で指定する。語数を 0 とした場合，領域は確保しないが，ラベルは有効である。

4.

DC	定数[,定数]…

DC命令は，定数で指定したデータを（連続する）語に格納する。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	命令の説明
10進定数	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	#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	'文字列'	文字列の文字数（>0）分の連続する領域を確保し，最初の文字は第1語の下位8ビットに，2番目の文字は第2語の下位8ビットに，…と順次文字データとして格納する。 各語の上位8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を書くことができる。 ただし，アポストロフィ（'）は2個続けて書く。
アドレス定数	ラベル	ラベルに対応するアドレスを1語の2進数データとして格納する。

2.4 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って， 目的の機能を果たす命令群を生成する（語数は不定）。

1.

IN	入力領域,入力文字長領域

IN命令は，あらかじめ割り当てた入力装置から，1 レコードの文字データを読み込む。

入力領域は，256 語長の作業域のラベルであり，この領域の先頭から，1 文字を 1 語に対応させて順次入力される。レコードの区切り符号（キーボード入力の復帰符号など）は，格納しない。格納の形式は，DC命令の文字定数と同じである。入力データが 256 文字に満たない場

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 256 文字を超える場合， 以降の文字は無視される。

入力文字長領域は，1 語長の領域のラベルであり，入力された文字の長さ（≥ 0）が 2 進数で格納される。ファイルの終わり（end of file）を検出した場合は，-1 が格納される。

IN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。

2.

OUT	出力領域,出力文字長領域

OUT命令は，あらかじめ割り当てた出力装置に，文字データを，1 レコードとして書き出す。

出力領域は，出力しようとするデータが 1 文字 1 語で格納されている領域のラベルである。格納の形式は，DC命令の文字定数と同じであるが， 上位 8 ビットは，OS が無視するので 0 でなくてもよい。

出力文字長領域は，1 語長の領域のラベルであり， 出力しようとする文字の長さ（≥ 0）を 2 進数で格納しておく。

OUT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。

3.

RPUSH	

RPUSH命令は，GRの内容を，GR1，GR2，…，GR7の順でスタックに格納する。

4.

RPOP	

RPOP命令は，スタックの内容を順次取り出し，GR7，GR6，…，GR1の順で GRに格納する。‘

2.5 機械語命令

機械語命令のオペランドは，次の形式で記述する。

r, r1 , r　GRは，記号 GR0〜GR7で指定する。

x　指標レジスタとして用いる GRは，記号 GR1〜GR7で指定する。

adr　アドレスは，10 進定数，16 進定数，アドレス定数又はリテラルで指定する。リテラルは，ひとつの 10 進定数，16 進定数又は文字定数の前に等号（=）を付けて 記述する。CASL II は，等号の後の定数をオペランドとする DC命令を生成し， そのアドレスを adr の値とする。

2.6 その他

1. アセンブラによって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。ただし，リテラルから生成される DC命令は，END命令の直前にまとめて配置される。
2. 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 プログラム実行の手引

3.1 OS

プログラムの実行に関して, 次の取決めがある。

1. アセンブラは，未定義ラベル（オペランド欄に記述されたラベルのうち，そのプログラム内で定義されていないラベル）を，他のプログラムの入口名（START命令のラベル）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 OS に任せる。OS は，実行に先立って他のプログラムの入口名との連係処理を行いアドレスを決定する（プログラムの連係)。
2. プログラムは，OS によって起動される。プログラムがロードされる主記憶の領域は不定とするが，プログラム中のラベルに対応するアドレス値は，OS によって実アドレスに補正されるものとする。
3. プログラムの起動時に，OS はプログラム用に十分な容量のスタック領域を確保し，その最後のアドレスに 1 を加算した値を SPに設定する。
4. OS は，CALL命令でプログラムに制御を渡す。プログラムを終了し OS に制御を戻すときは，RET命令を使用する。
5. 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利用者が行う。
6. OS は，入出力装置や媒体による入出力手続の違いを吸収し，システムでの標準の形式及び手続（異常処理を含む）で入出力を行う。したがって，IN，OUT命令では，入出力装置の違いを意識する必要はない。

3.2 未定義事項

プログラムの実行等に関し，この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 II の理解を助けるため又は COMET II の処理系作成者に対する便宜のための資料である。したがって，COMET II，CASL II の仕様に影響を与えるものではない。

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。ここで，OPの数値は 16 進表示で示す。

15 11 7 3 015 0←――― ビット番号

第1語				第2語	命令語長	命令語とアセンブラとの対応		
OP		r/r1	x/r2	adr		機械語命令		意味
主OP	副OP							
0	0	—	—	—	1	NOP		no operation
1	0				2	LD	r,adr,x	load
	1				2	ST	r,adr,x	store
	2				2	LAD	r,adr,x	load address
	4			—	1	LD	r1,r2	load
2	0				2	ADDA	r,adr,x	add arithmetic
	1				2	SUBA	r,adr,x	subtract arithmetic
	2				2	ADDL	r,adr,x	add logical
	3				2	SUBL	r,adr,x	subtract logical
	4			—	1	ADDA	r1,r2	add arithmetic
	5			—	1	SUBA	r1,r2	subtract arithmetic
	6			—	1	ADDL	r1,r2	add logical
	7			—	1	SUBL	r1,r2	subtract logical
3	0				2	AND	r,adr,x	and
	1				2	OR	r,adr,x	or
	2				2	XOR	r,adr,x	exclusive or
	4			—	1	AND	r1,r2	and
	5			—	1	OR	r1,r2	or
	6			—	1	XOR	r1,r2	exclusive or
4	0				2	CPA	r,adr,x	compare arithmetic
	1				2	CPL	r,adr,x	compare logical
	4			—	2	CPA	r1,r2	compare arithmetic
	5			—	2	CPL	r1,r2	compare logical
5	0				2	SLA	r,adr,x	shift left arithmetic
	1				2	SRA	r,adr,x	shift right arithmetic
	2				2	SLL	r,adr,x	shift left logical
	3				2	SRL	r,adr,x	shift right logical
6	1	—			2	JMI	adr,x	jump on minus
	2	—			2	JNZ	adr,x	jump on non zero
	3	—			2	JZE	adr,x	jump on zero
	4	—			2	JUMP	adr,x	unconditional jump
	5	—			2	JPL	adr,x	jump on plus
	6	—			2	JOV	adr,x	jump on overflow
7	0	—			2	PUSH	adr,x	push
	1		—	—	1	POP	r	pop
8	0	—			2	CALL	adr,x	call subroutine
	1	—	—	—	1	RET		return from subroutine
9 〜 E						その他の命令		
F	0	—			2	SVC	adr,x	supervisor call

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〕IN命令

```
LABEL  IN IBUF,LEN
```

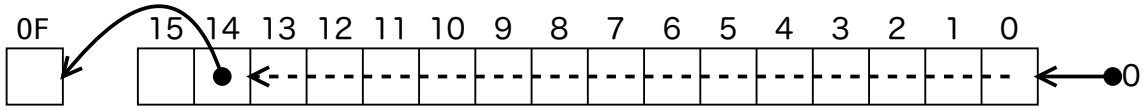
マクロ生成

```
LABEL PUSH 0,GR1
      PUSH 0,GR2
      LAD GR1,IBUF
      LAD GR2,LEN
      SVC 1
      POP GR2
      POP GR1
```

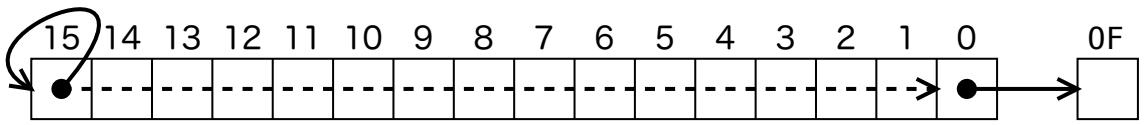
シフト演算命令におけるビットの動き

シフト演算命令において，例えば，1 ビットのシフトをしたときの動き及び OFの変化は，次のとおりである。

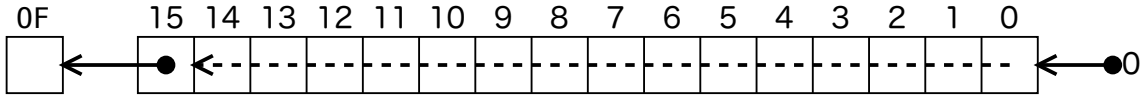
1. 算術左シフトでは，ビット番号 14 の値が設定される。



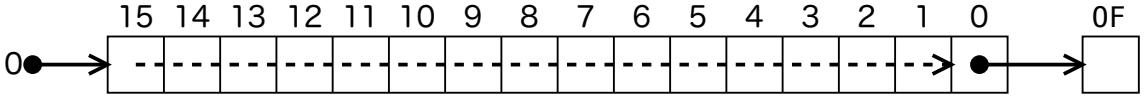
2. 算術右シフトでは，ビット番号 0 の値が設定される。



3. 論理左シフトでは，ビット番号 15 の値が設定される。



4. 論理右シフトでは，ビット番号 0 の値が設定される。



プログラムの例

```
COUNT1  START                   ;
;       入力    ; GR1:検索する語
;       処理    ; GR1 中の'1' のビットの個数を求める
;       出力    ; GR0:GR1 中の'1' のビットの個数
        PUSH    0,GR1           ;
```

```
        PUSH    0,GR2           ; Count = 0
        SUBA    GR2,GR2         ; 全部のビットが'0'?
        AND     GR1,GR1         ; 全部のビットが'0' なら終了
        JZE     RETURN          ; Count = Count + 1
MORE    LAD     GR2,1,GR2       ; 最下位の'1' のビット 1 個を
        LAD     GR0,-1,GR1      ; '0' に変える
        AND     GR1,GR0         ; '1' のビットが残っていれば繰返し
        JNZ     MORE            ; GR0 = Count
RETURN  LD      GR0,GR2         ;
        POP     GR2             ;
        POP     GR1             ;
        RET                     ; 呼出しプログラムへ戻る
        END                     ;
```