情報科学演習 D
アセンブラ言語処理系の仕様

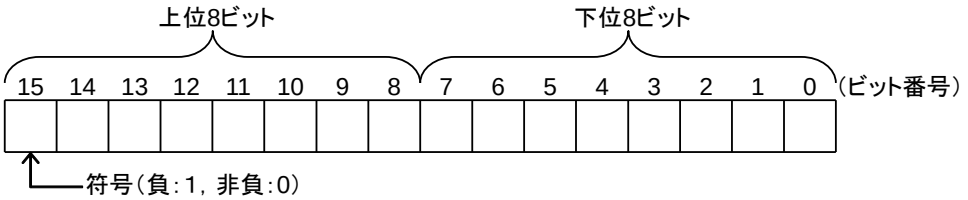
1　情報科学演習 D で使用するアセンブラ言語処理系について

情報科学演習 D で使用するアセンブラ言語処理系（COMETII 及び CASLII）の仕様は，IPA（情報処理推進機構）の情報処理技術者試験や一年次の情報科学基礎で利用されているアセンブラ処理系とは一部仕様（特にスタックポインタの扱い）が異なる．最終課題を作成する前に，本演習で用いる処理系の仕様を良く理解すること．

2　システム COMETII の仕様

2.1　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 語長または 2 語長である．
5. レジスタとして，GR（16 ビット），PR（16 ビット），FR（3 ビット）の 3 種類がある．
   - GR（汎用レジスタ，General Register）は，GR0~GR8 の 9 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~GR8 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用いる．また，GR8 は SP（スタックポインタ，Stack Pointer）を兼ねている．SP は，スタックの最上段のアドレスを保持しているレジスタである．
   - 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，それ以外のときは 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に 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．

2.2 命令

命令の形式及びその機能を表 1∼ 表 10 に示す．これらの表では，1 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．なお，表 1∼ 表 10 中の表記については，下記のとおりである．

- r, r1, r2 は，いずれも GR を表す．指定できる GR は GR0∼GR8.
- adr は，アドレスを示す．指定できる値の範囲は 0∼65535.
- x は，指標レジスタとして用いる GR を示す．指定できる GR は GR1∼GR8.

表 1: ロード，ストア，ロードアドレス命令

命令	書き方		命令の説明	FR の設定
	命令コード	オペランド		
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 ADress	LAD	r, adr [, x]	r ← 実効アドレス	

表 2: 算術，論理演算命令

命令	書き方		命令の説明	FR の設定
	命令コード	オペランド		
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)	

- [] は，その指定が省略できることを示す.
- () は，その内側のレジスタ又はアドレスに格納されている内容を表す.
- ← は，演算結果を左辺のレジスタまたはアドレスに格納することを示す.
- $+_L$,$-_L$ は，論理加算，論理減算を示す.
- FR の設定内容は以下のとおりである.
  - 「◯」は，FR が設定されることを示す.
  - 「$◯_{*1}$」は，FR が設定されることを示す．ただし，OF には 0 が設定される.
  - 「$◯_{*2}$」は，FR が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.
  - 「−」は，FR に実行前の値が保持されることを示す.

表 3: 比較演算命令

命令	書き方		命令の説明	FR の設定
	命令コード	オペランド		
算術比較 ComPare Arithmetic	CPA	r1, r2	(r1) と (r2)，又は (r) と (実効アドレス) の算術比較または論理比較を行い，結果に基づき FR を変更する (表 4).	$◯_{*1}$
		r, adr [, x]		
論理比較 ComPare Logical	CPL	r1, r2		
		r, adr [, x]		

表 4: 比較演算命令での FR 変更内容

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

表 5: シフト演算命令

命令	書き方		命令の説明	FR の設定
	命令コード	オペランド		
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シフトする．シフトにより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) を実行アドレスで指定したビット数シフトする．シフトにより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]		

2.3 文字符号表

本演習で用いるアセンブラ言語処理系では，JIS X0201 ラテン文字・仮名名用 8 ビット符号で規定する文字の符号表を使用する．符号表の一部を表 11 に示す．1 文字は 8 ビットからなり，上位 4 ビットを列で，下位 4 ビットを行で表す．例えば，間隔，4，H，¥ のビット構成は，16 進数表示で，それぞれ 20，34，48，

表 6: 分岐命令

命令	書き方		命令の説明	FR の設定
	命令コード	オペランド		
正分岐 Jump on PLus	JPL	adr [, x]	FR の値によって，実効アドレスに分岐する．分岐しない時は次の命令に進む．分岐の条件は表 7 を参照.	-
負分岐 Jump on MInus	JMI	adr [, x]		
非零分岐 Jump on Non Zero	JNZ	adr [, x]		
零分岐 Jump on ZEro	JZE	adr [, x]		
オーバーフロー分岐 Jump on OVerflow	JOV	adr [, x]		
無条件分岐 unconditional JUMP	JUMP	adr [, x]	無条件に実効アドレスに分岐する．	

表 7: 分岐命令の分岐条件

命令コード	分岐するときの FR の値		
	OF	SF	ZF
JPL		0	0
JML		1	
JNZ			0
JZE			1
JOV	1		

表 8: スタック操作命令

命令	書き方		命令の説明	FR の設定
	命令コード	オペランド		
プッシュ PUSH	PUSH	adr [, x]	SP ← (SP) $-_L$ 1, (SP) ← 実効アドレス	-
ポップ POP	POP	r	r ← ((SP)) (SP) ← (SP) $+_L$ 1	
連続プッシュ Rush PUSH	RPUSH		GR の内容を, GR1, GR2, ···, GR8 の順序でスタックに格納する．	
連続ポップ Rush POP	RPOP		スタックの内容を順次取り出し，GR8, GR7, ···, GR1 の順序で GR に格納する．	

5C である．16 進数表示で，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．

表 9: コール，リターン命令

命令	書き方	命令の説明	FR の設定
	命令コード　オペランド		
コール CALL, subroutine	CALL　adr [, x]	SP ← (SP) $-_L$ 1, (SP) ← (PR), PR ← 実効アドレス	-
リターン RETurn from subroutine	RET	PR ← ((SP)) (SP) ← (SP) $+_L$ 1	

表 10: その他の命令

命令	書き方	命令の説明	FR の設定
	命令コード　オペランド		
スーパバイザコール SuperVisor Call	SVC　adr [, x]	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 の値は不定．	-
ノーオペレーション No OPeration	NOP	何もしない．	

表 11: 文字の符号表（一部）

	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	

3　アセンブラ言語 CASLII の仕様

3.1　言語の仕様

1. CASLII は，COMETII のためのアセンブラ言語である．
2. プログラムは，命令行及び注釈行からなる．
3. 1 命令は 1 命令行で記述し，次の行へ継続できない．
4. 命令行及び注釈行は，表 12 に示す形式で，行の 1 文字目から記述する．なお，表 12 を読むにあたり下記の注意事項に留意すること．
   - [] は，その中の指定が省略できることを示す．
   - {} は，その中の指定が必須であることを示す．
   - ラベルは，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，英大文字または数字のいずれでも良い．なお，予約語である GR0~GR8 は使用できない．
   - 空白は，1 文字以上の間隔文字の列である．
   - 命令コードについては，命令ごとに記述の形式が定義されている．
   - オペランドについては，命令ごとに記述の形式が定義されている．
   - コメントは，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．

3.2　命令の種類

命令は，4 種類のアセンブラ命令（START，END，DS，DC），2 種類のマクロ命令（IN，OUT）及び機械語命令（COMETII の命令）からなる．その仕様を表 13 に示す．

3.3　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．なお，DS 命令および DC 命令は，RET 命令と END 命令の間にのみ記述することができる．

3.3.1　START 命令

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を指定する．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．

表 12: 行の様式

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]

3.3.2 END 命令

END 命令は，プログラムの終わりを定義する．

3.3.3 DS 命令

DS 命令は，指定した語数の領域を確保する．語数は，10 進数（≥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．

3.3.4 DC 命令

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（表 14 を参照）．

3.4 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語を生成する（語数は不定）．

表 13: 命令の様式

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
機械語命令	[ラベル]	2.2 節を参照		

表 14: DC 命令の様式

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．

3.4.1 IN 命令

IN 命令は，あらかじめ割り当てられた入力装置から，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 の値は不定となる．

3.4.2 OUT 命令

OUT 命令は，あらかじめ割り当てられた出力装置に，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．

3.5 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, r2 は，いずれも GR を表す．指定できる GR は GR0∼GR8.
- x は，指標レジスタとして用いる GR である．指定できる GR は GR1∼GR8.
- adr は，アドレスを示す．アドレスは，10 進定数，16 進定数，アドレス定数またはリテラルで指定する．リテラルは，1 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．CASLII 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．

3.6 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. 空行を用いることはできない．

4 プログラム実行の手引

4.1 OS

プログラムの実行に関して次の取り決めがある．

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，実行に先だって他のプログラムの入口名との連系処理を行いアドレスを決定する（プログラムの連系）．

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きの違いを吸収し，システムでの標準の形式及び手続き（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．