情報処理技術者試験

試験で使用する
情報技術に関する用語・プログラム言語など

Ver 1.0

平成 20 年 10 月 27 日

独立行政法人 情報処理推進機構
IT人材育成本部 情報処理技術者試験センター

これまでの内容に，マーク付け言語（マークアップ言語）に関する項目を追加しました。
そのほかの項目については変更ありません。

1. 情報技術に関する用語

試験で使用する情報技術に関する用語は，日本工業規格（JIS）に制定されているものについては，その規定に従う。

2. 記号・図など

試験で使用する代表的な記号・図などは，次の仕様に従う。次以外については，問題文中で定義する。

情報処理用流れ図など	：JIS X 0121
決定表	：JIS X 0125
計算機システム構成の図記号	：JIS X 0127
プログラム構成要素及びその表記法	：JIS X 0128

3. プログラム言語

① 基本情報技術者試験において，ソフトウェア開発分野に関する試験問題に出題するプログラム言語は，C，COBOL，Java，アセンブラ言語（CASLⅡ）の 4 言語とする（4 言語のほかに，表計算ソフトによる試験問題を出題する）。

② 情報セキュリティスペシャリスト試験において，セキュアプログラミングに関する試験問題に出題するプログラム言語は，C++，Java，Perl の 3 言語のいずれかとする。

③ 仕様などは，次による。

C	：JIS X 3010
COBOL	：JIS X 3002
Java	：The Java Language Specification, Third Edition（JLS 3.0）（[1]） （URL　http://java.sun.com/docs/books/jls/index.html）
アセンブラ言語	：「**別紙 1**　アセンブラ言語の仕様」（3 ページ）による。
C++	：JIS X 3014
Perl	：「**別紙 2**　プログラム言語 Perl の用例・解説」（11 ページ）による。

注（[1]）　なお，JLS 3.0 で追加された機能については，主に次の機能を問題中で使用する。ただし，これらの 3 機能に限定するものではない。

・Generics（総称）
・拡張された for 文
・enum（列挙）型

また，メタデータは範囲外とする。

4. データベース言語

試験で使用するデータベース言語は，次の仕様に従う。

SQL	：JIS X 3005 規格群

5. マーク付け言語（マークアップ言語）

試験で使用するマーク付け言語は，次の仕様に従う。

HTML	：JIS X 4156
XML	：JIS X 4159

6. 表計算ソフトなどのソフトウェアパッケージ

表計算ソフト　　:「**別紙 3**　表計算ソフトの機能・用語」(18 ページ）による。ここに規定されていない機能・用語などについては，問題文中で定義する。

表計算ソフト以外のソフトウェアパッケージの機能・用語などは，問題文中で定義する。

＜JIS の参照（日本工業標準調査会ホームページ）＞

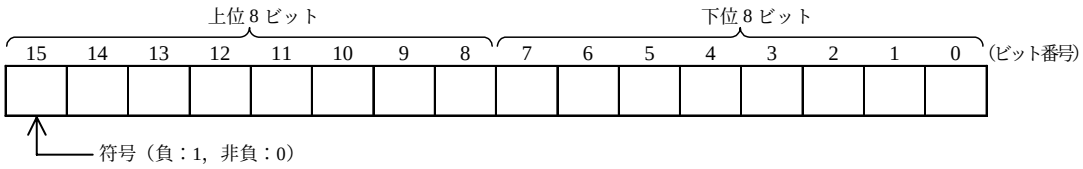
URL　http://www.jisc.go.jp/

別紙１　アセンブラ言語の仕様

1. システム COMETⅡ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		命令の説明	FRの設定
	命令コード	オペランド		

(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	

(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)	

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

(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]		

(5) 分岐命令

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。分岐しないときは，次の命令に進む。	—
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。	

命令	分岐するときの FR の値 OF	SF	ZF
JPL		0	0
JMI		1	
JNZ			0
JZE			1
JOV	1		

(6) スタック操作命令

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	

(7) コール，リターン命令

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	

(8) その他

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 は不定となる。	—
ノーオペレーション No OPeration	NOP	何もしない。	

（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) 右に符号表の一部を示す。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。16 進表示で，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。
(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。

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	

2. アセンブラ言語 CASLⅡの仕様

2.1 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，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，RPUSH，RPOP）及び機械語命令（COMETⅡ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 けたの 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，間隔及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 の順序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 の順序で 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, r2	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，一つの 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。

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，COMETⅡの理解を助けるため又は COMETⅡ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，COMETⅡ，CASLⅡ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。ここで，OP の数値は 16 進表示で示す。

15　11　7　3　0 15　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			—	1	CPA	r1,r2	compare arithmetic
	5			—	1	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

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〕IN 命令

```
LABEL  IN      IBUF,LEN
```

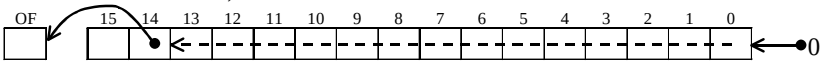
↓マクロ生成

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

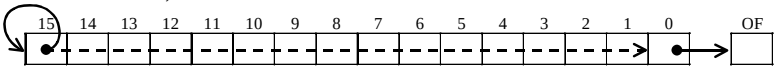
3. シフト演算命令におけるビットの動き

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。

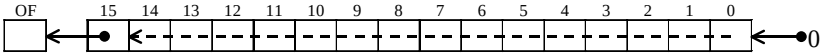
(1) 算術左シフトでは，ビット番号 14 の値が設定される。



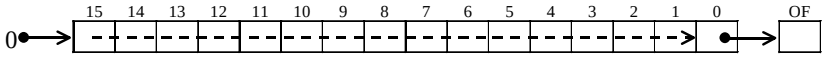
(2) 算術右シフトでは，ビット番号 0 の値が設定される。



(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。



4. 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           ;
        SUBA    GR2,GR2         ; Count = 0
        AND     GR1,GR1         ; 全部のビットが'0'？
        JZE     RETURN          ; 全部のビットが'0'なら終了
MORE    LAD     GR2,1,GR2       ; Count = Count + 1
        LAD     GR0,-1,GR1      ; 最下位の'1'のビット１個を
        AND     GR1,GR0         ;   '0'に変える
        JNZ     MORE            ; '1'のビットが残っていれば繰返し
RETURN  LD      GR0,GR2         ; GR0 = Count
        POP     GR2             ;
        POP     GR1             ;
        RET                     ; 呼出しプログラムへ戻る
        END                     ;
```

別紙２　プログラム言語 Perl の用例・解説

Perl を使用した問題では，各問題文中に注記がない限り，次に示す用例に従って記述する。なお，用例は，解答で使用する演算子，関数，予約語などを制限するものではない。

種類	用例 解説

1. 注釈

#	#ここにコメントを書く 行末までが注釈となる。

2. リテラル

スカラ	123 10 進数 123 である。
	12.3 10 進数 12.3 である。
	4E-5 10 進数 4×10^{-5} である。
	0x9f 16 進数 9F である。
	0147 8 進数 147 である。
	0b010111 2 進数 010111 である。
	$var = "hello"; print '$var ', "$var ", `echo world`; 変数 var に文字列“hello”を代入する。文字列のスカラ '$var ', "$var ", `echo world` を出力する。"$var " は変数を展開し，`echo world` はコマンドの出力を展開するので，出力は“ $var hello world ”となる。
	\n 制御文字（改行）である。
	\r 制御文字（復帰）である。
	\t 制御文字（水平タブ）である。
リストリテラル	('a', 'b', 'c') リスト ('a', 'b', 'c') である。
	('a', 'b', 'c')[0] リスト ('a', 'b', 'c') の 1 番目の要素 'a' である。
	() 空リストである。
	('a' => 'alpha', 'b' => 'bravo', 'c' => 'charlie') キー a, b, c に，それぞれ値 alpha, bravo, charlie を結び付けたハッシュである。

ファイルハンドル	STDIN
	標準入力である。
	STDOUT
	標準出力である。
	STDERR
	標準エラー出力である。
	ARGV
	コマンドラインから指定されたファイル名のリストを順に読み込むためのファイルハンドルである。

3. 変数

スカラ変数	$var
	スカラ変数 var である。
配列変数	@ary
	配列変数 ary である。
配列要素	$ary[6]
	配列変数 ary の 7 番目の要素である。
ハッシュ変数	%hash
	ハッシュ変数 hash である。
ハッシュ要素	$hash{'a'}
	ハッシュ変数 hash の要素のうち，キー a に結び付けられた値である。
局所的な変数	{my $var;}
	{} 内を有効範囲とする変数 var の宣言である。
$_	$_ = "abc"; if (/b/) print "match";
	パターンマッチの演算子が省略されたとき，$_ の文字列 “ abc ” が // 内のパターン b と一致するかどうかを判定し，“ match ” が出力される。
@ARGV	@ARGV
	コマンドライン引数のリストを格納する配列変数である。
@_	@_
	サブルーチンに渡す引数のリストを格納する配列変数である。

4. 演算子

->	$object->method1
	オブジェクト object のメソッド method1 を呼び出す。
	Class->method2
	クラス Class のメソッド method2 を呼び出す。
++, --	$a++
	変数 a を評価した後に 1 を加算する。
	--$b
	変数 b から 1 を減算した後に評価する。
!, +（単項), -（単項）	!$a
	変数 a の論理否定である。
	+123
	正の数 123 である。
	-123
	負の数 123 である。

=~, !~	$html_contents =~ /<b>/
	変数 html_contents の値に，文字列“ <b> ”が含まれているときに真を返す。
	$html_contents !~ / /
	変数 html_contents の値に，文字列“ ”が含まれていないときに真を返す。
*, /, %	314 * 34
	314 と 34 の乗算である。
	6 / 469
	6 を 469 で割る除算である。
	34 % 6
	34 を 6 で割る剰余演算である。
+, -, .	3.14 + 2.72
	3.14 と 2.72 の加算である。
	220 – 8125
	220 から 8125 を引く減算である。
	"IPA"."JITEC"
	文字列“ IPA ”と“ JITEC ”の連結である。
<, >, <=, >=, lt, gt, le, ge	1 < 2
	数値 1 と 2 を比較し，演算子の左側が右側より小さいので真を返す。数値の関係演算子には，ほかに >，<=，>= がある。
	"b" lt "a"
	文字列“ b ”と“ a ”を比較し，演算子の左側が右側より小さくないので偽を返す。文字列の関係演算子には，ほかに gt，le，ge がある。
==, !=, <=>, eq, ne, cmp	1 <=> 2
	数値 1 と 2 を比較し，演算子の左側が右側より大きければ 1，等しければ 0，小さければ −1 を返すので，この場合は −1 を返す。数値の比較演算子には，ほかに ==，!= がある。
	"b" cmp "a"
	文字列“ b ”と“ a ”を比較し，演算子の左側が右側より大きければ 1，等しければ 0，小さければ −1 を返すので，この場合は 1 を返す。文字列の比較演算子には，ほかに eq，ne がある。
&&	$x >= 0 && $x < 10
	変数 x の値が 0 以上かつ 10 未満なら真を返す。
||	$x < 0 || $x >= 10
	変数 x の値が 0 未満又は 10 以上なら真を返す。
..	@card = (1 .. 52)
	1 から 52 までの連続する整数を配列変数 card に代入する。
=, +=, -=, *=, /=, %=	$a = 1
	変数 a に 1 を代入する。
	$a += 10
	変数 a の値に 10 を加算して a に代入する。 代入演算子には，ほかに -=，*=，/=，%= がある。
=>, ,	%hash = ('a' => 'alpha', 'b' => 'bravo', 'c' => 'charlie')
	a に alpha，b に bravo，c に charlie を結び付けたハッシュをハッシュ変数 hash に代入する。
not	not $a
	変数 a の論理否定である。

and	`$a < 0 and $b == 0` 変数 a が 0 より小さいか，変数 b が 0 と等しいかという二つの関係式の論理積である。
or, xor	`$a < 0 or $b == 0` 変数 a が 0 より小さいか，変数 b が 0 と等しいかという二つの関係式の論理和である。
	`$a < 0 xor $b == 0` 変数 a が 0 より小さいか，変数 b が 0 と等しいかという二つの関係式の排他的論理和である。

注 演算の優先順位は，上表の枠の順である。

5. 文

if	`if ($var == 1) {`
	`  print "a";`
	`} elsif ($var == 2) {`
	`  print "b";`
	`} else {`
	`  print "c";`
	`}`
	変数 var の値が 1 なら“ a ”を，2 なら“ b ”を，それ以外なら“ c ”を出力する。

while	`$i = 1;`
	`while ($i <= 10) {`
	`  print $i++, "\n";`
	`}`
	変数 i の値を 1 から 1 ずつ増やし，10 回出力する。

for	`for ($i = 1; $i <= 10; $i++){`
	`  print "$i\n";`
	`}`
	変数 i の値を 1 から 1 ずつ増やし，10 回出力する。

foreach	`foreach $i (1, 3, 5) {`
	`  print "$i\n";`
	`}`
	変数 i にリストの各要素 1，3，5 を順に代入し，3 回出力する。

next	`for ($i = 1; $i <= 10; $i++) {`
	`  next if $i % 2;`
	`  print "$i\n";`
	`}`
	変数 i が 2 で割り切れないとき，ループ本体の next 行より後を実行しないので，偶数を出力する。

6. 正規表現

\	`/\.\^\$\[|\+\*\?\{\(\)\/\\/` 次の 1 文字そのものを表す。“ .^$[|+*?{()/\ ”と一致する。
.	`/www.ipa.go.jp/` 改行文字以外の任意の 1 文字と一致する。“ wwwdipa,go@jp ”と一致する。
^	`/^ab/` 先頭が“ ab ”である文字列と一致する。“ abc ”と一致するが，“ cab ”とは一致しない。

$	/yz$/
	末尾が“ yz ”である文字列と一致する。“ xyz ”と一致するが,“ yza ”とは一致しない。
+	/go+d/
	直前の 1 文字 o の 1 回以上の繰返しと一致する。“ god ”や“ gooood ”と一致するが,“ gd ”とは一致しない。
*	/go*d/
	直前の 1 文字 o の 0 回以上の繰返しと一致する。“ gd ”,“ god ”や“ gooood ”と一致する。
?	/colou?r/
	直前の 1 文字 u の 0 回又は 1 回の出現と一致する。“ color ”又は“ colour ”と一致する。
{m}, {m,n}	/co{2}l/
	直前の 1 文字 o の 2 回の繰返しと一致する。“ cool ”と一致するが,“ col ”や“ coool ”とは一致しない。
	/go{1,3}d/
	直前の 1 文字 o の 1 ～ 3 回の繰返しと一致する。“ god ”や“ good ”と一致するが,“ gd ”や“ gooood ”とは一致しない。
(…)	/<(h.)>/
	() 内の文字列と一致するパターンを部分パターンとしてまとめる。“ <h1> ”と一致した場合は“ h1 ”が,“ <hr> ”と一致した場合は“ hr ”が, まとめられる。
\1, \2, …	/<(..)><([bp])>JITEC<\/\2><\/\1>/
	左から順に () 内のパターンと一致した文字列が \1, \2, … に割り当てられる。“ <h1>JITEC</h1> ”と一致するが,“ <td>JITEC</p> </td> ”とは一致しない。
[…]	/<h[12r]>/
	[] 内で指定した文字 1, 2 又は r のどれか一つと一致する。“ <h1> ”,“ <hr> ”と一致するが,“ <h3> ”や“ <HR> ”とは一致しない。
	/[^0-9]/
	[] 内で指定した 0 ～ 9 以外の 1 文字と一致する。“ a ”と一致するが,“ 3 ”とは一致しない。
…|…	/<(a href|img src)=/
	| で区切られた“ a href ”又は“ img src ”のどちらか一方と一致する。“ <a href= ”や“ <img src= ”と一致するが,“ <A HREF= ”や“ <img height= ”とは一致しない。

7. サブルーチン

定義	sub greeting {
	  print "hello Perl\n";
	}

	“ hello Perl ”を出力するサブルーチン greeting を定義する。
呼出し	subroutine ($arg1, $arg2);
	サブルーチン subroutine を引数 arg1 と arg2 で呼び出す。() を省略して“ subroutine $arg1, $arg2; ”とする表記もある。
戻り	return -1;
	サブルーチンから抜け出し, 値 −1 を返す。

8. モジュール

use	use CGI; モジュール CGI を 1 度だけ読み込み，利用可能にする。

9. メソッド呼出し

->	$object->method1(arg1); 演算子 -> を使って，オブジェクト object のメソッド method1 を引数 arg1 で実行する。
	Class->method2(arg1, arg2); 演算子 -> を使って，クラス Class のメソッド method2 を引数 arg1 及び arg2 で実行する。

10. 文字列操作関数

chomp	chomp @lines; 配列変数 lines の各要素の末尾にある改行文字を削除する。
eval	eval $exp_str; 変数 exp_str の内容を Perl プログラムとして解釈し実行する。
length	length $long_str; 変数 long_str に格納される文字列の文字数を返す。

11. 配列・ハッシュ操作関数

keys	%hash = ('a' => 'alpha', 'b' => 'bravo', 'c' => 'charlie');
	foreach $key (keys %hash) {
	  print "$key\n";
	}
	ハッシュ変数 hash のキーのリストを取り出し，各キーを出力する。この場合は，“ a ”，“ b ”，“ c ” を順不同に出力する。

shift	$next = shift @queue; 配列変数 queue の先頭要素を取り除いて詰め，取り除いた値を変数 next に代入する。
sort	@pile = sort @jumble; 配列変数 jumble の値を文字列の大小比較によって昇順に整列し，配列変数 pile に代入する。
	@pile = sort {$b <=> $a} @jumble; 配列変数 jumble の値を数値の大小比較に従って降順に整列し，配列変数 pile に代入する。
split	@fields = split ',', $csv; 変数 csv の値をコンマで区切って分割したリストを配列変数 fields に代入する。

12. 検索・置換関数

m/…/ 又は /…/	$html_contents =~ /<b>/i; 変数 html_contents の値が，文字列 “ <b> ” 又は “ <B> ” を含んでいるかどうかを判定する。i は，大文字，小文字の区別をしないオプションである。
s/…/…/	$html_contents =~ s/ /\n/gi; 変数 html_contents の中の文字列 “ ”，“ ”，“ ” 又は “ ” を改行文字に置換する。g は，一致したすべての文字列を置換するオプションである。

$`, $&, $', $1, $2, …	'The date is 1970-01-23.' =~ /([0-9]{4})-([0-9]{2})-([0-9]{2})/; print "String before the date: $`\n"; print "Date: $&\n"; print "String after the date: $'\n"; print "Year: $1\n", "Month: $2\n", "Day: $3\n";
	文字列“The date is 1970-01-23.”に対して，一致した部分の前の文字列，一致した文字列，一致した部分の後ろの文字列をそれぞれ変数 `，&，' に代入する。また，() で囲まれた部分パターンと一致した文字列を，1 番目から順に変数 1，2，3 に代入する。これらを利用し，“String before the date: The date is ”，“Date: 1970-01-23”，“String after the date: .”，“Year: 1970”，“Month: 01”，“Day: 23”の 6 行を出力する。

13. 入出力操作関数

open	open LOG, '>>cgi.log';
	ファイル cgi.log を追記モードで開き，ファイルハンドル LOG に対応付ける。
<*filehandle*>	$line = <USER_FILE>;
	ファイルハンドル USER_FILE から 1 行を読み込んで変数 line に代入する。
<>	@records = <>;
	標準入力（コマンドライン引数があるときは，コマンドライン引数で指定されたファイル）から順にデータを読み込み，すべての行を配列変数 records に代入する。
print	print LOG "sync.\n";
	ファイルハンドル LOG に対応するファイルに文字列を出力する。
close	close LOG;
	ファイルハンドル LOG に対応するファイルを閉じる。

14. システムインタフェース

die	open(FILE, 'a_file') or die 'cannot open a_file';
	ファイル a_file を開く。開くのに失敗したとき，“cannot open a_file”というメッセージを出力して実行を終了する。
system	system 'a.out';
	コマンド a.out を実行し，コマンドが終了するまで待機する。

別紙３　表計算ソフトの機能・用語

表計算ソフトの機能，用語などは，原則として次による。

1. **ワークシート**

表計算ソフトの作業領域をワークシートという。ワークシートの大きさは 256 列（列 A から列 Z，列 AA から列 AZ，さらに列 BA から列 BZ と続き，列 IV まで続く），10,000 行（行 1 から行 10,000 まで）とする。

2. **セル**

(1) ワークシートを縦・横に分割したときの一つのます目をセルという。列 A 行 1 のセルは A1 と表す。

(2) 長方形の形をしたセルの集まりを範囲として指定することができる。範囲の指定は A1 ～ B3 のように表す。

(3) 範囲に名前を付けることができる。範囲名は［　］を用いて，“ セル A1 ～ B3 に［金額］と名前を付ける ” などと表す。

(4) データが入力されていないセルを，空白セルという。

3. **セルへの入力**

(1) セルに数値，文字列，計算式を入力できる。

(2) セルを保護すると，そのセルへの入力を不可能にすることができる。セルの保護を解除すると，そのセルへの入力が再び可能になる。

(3) セル A1 に数値 5 を入力するときは，“ セル A1 に 5 を入力 ” と表す。

(4) セル B2 に，文字列 ABC を入力するときは，“ セル B2 に 'ABC' を入力 ” と表す。

(5) セル C3 に，セル A1 とセル B2 の和を求める計算式を入力するときは，“ セル C3 に計算式 A1＋B2 を入力 ” などと表す。

4. **セルの内容の表示**

(1) セルに数値を入力すると，右詰めで表示される。

(2) セルに文字列を入力すると，左詰めで表示される。

(3) セルに計算式を入力すると，計算結果が数値ならば右詰めで，文字列ならば左詰めで表示される。

(4) セルの内容の表示については，左詰め，中央揃(そろ)え，右詰めに変更できる。

5. **計算式**

(1) 計算式には，数学で用いられる数式が利用できる。

(2) 計算式で使用する算術演算子は，“ ＋ ”（加算），“ － ”（減算），“ ＊ ”（乗算），“ ／ ”（除算）及び “ ＾ ”（べき算）とする。

(3) 算術演算子による計算の優先順位は，数学での優先順位と同じである。

6. **再計算**

(1) セルに計算式を入力すると，直ちに計算結果を表示する。

(2)セルの数値が変化すると，そのセルを参照しているセルも自動的に再計算される。この再計算は A1，A2，A3，…，B1，B2，B3，… の順に 1 回だけ行われる。

7. **関数**

(1) 計算式には次の表で定義する関数を利用することができる。

関数名と使用例	解　　説
合計(A1 ～ A5)	セル A1 からセル A5 までの範囲のすべての数値の合計を求める。
平均(B2 ～ F2)	セル B2 からセル F2 までの範囲のすべての数値の平均を求める。
平方根(I6)	セル I6 の値（正の数値でなければならない）の正の平方根を求める。
標準偏差(D5 ～ D19)	セル D5 からセル D19 までの範囲のすべての数値の標準偏差を求める。
最大(C3 ～ E7)	セル C3 からセル E7 までの範囲のすべての数値のうちの最大値を求める。
最小([得点])	[得点] と名前を付けた範囲のすべての数値のうちの最小値を求める。
IF(B3 ＞ A4, '北海道', '九州')	第 1 引数に指定された論理式が真（成立する）ならば第 2 引数が，偽（成立しない）ならば第 3 引数が求める値となる。左の例では，セル B3 が A4 より大きければ文字列 '北海道' が，それ以外の場合には文字列 '九州' が求める値となる。論理式中では，比較演算子として，＝，≠，＞，＜，≦，≧を利用することができる。第 2 引数，第 3 引数に，更に IF 関数を利用して，IF 関数を入れ子にすることができる。
個数(G1 ～ G5)	セル G1 から G5 までの範囲のうち，空白セルでないセルの個数を求める。
条件付個数(H5 ～ H9, '＞25')	第 1 引数に指定された範囲のうち，第 2 引数に指定された条件を満たすセルの個数を求める。左の例では，セル H5 から H9 までの範囲のうち，値として25より大きな数値を格納しているセルの個数を求める。
整数部(A3)	セル A3 の値（数値でなければならない）を超えない最大の整数を求める。 例えば， 整数部(3.9) ＝ 3 整数部(－3.9) ＝ －4 となる。
剰余(C4, D4)	セル C4 の値を被除数，D4 の値を除数とし，被除数を除数で割ったときの剰余を求める。剰余の値は常に除数と同じ符号をもつ。“ 剰余 ” 関数と “ 整数部 ” 関数は，次の関係を満たしている。 剰余(x, y) ＝ x － y ＊ 整数部(x／y)
論理積(論理式 1, 論理式 2, …)	引数として指定された論理式がすべて真であれば，真を返す。引数のうち一つでも偽のものがあれば，偽を返す。引数として指定できる論理式の数は任意である。
論理和(論理式 1, 論理式 2, …)	引数として指定された論理式がすべて偽であれば，偽を返す。引数のうち一つでも真のものがあれば，真を返す。引数として指定できる論理式の数は任意である。
否定(論理式)	引数として指定された論理式が真であれば偽を，偽であれば真を返す。
注　“ 合計 ”, “ 平均 ”, “ 標準偏差 ”, “ 最大 ”, “ 最小 ” は，引数で指定された範囲のセルのうち，値として数値以外を格納しているものは無視する。	

(2) 関数の引数には，セルを用いた計算式，範囲，範囲名，論理式を指定することができる。

8. セルの複写

(1) セルに入力された数値，文字列，計算式を他のセルに複写することができる。

(2) セルに入力された計算式が他のセルを参照している場合は，複写先のセルでは相対的にセルが自動的に変更される。例えば，セル A6 に合計(A1 ～ A5)を入力した場合，セル A6 をセル B7 に複写すると，セル B7 の計算式は合計(B2 ～ B6)となる。

9. 絶対参照

(1) 計算式を複写しても参照したセルが変わらない参照を絶対参照といい，記号 $ を用いて $A $1 などと表す。例えば，セル B1 に計算式 $A $1＋5 を入力した場合，セル B1 をセル C4 に複写してもセル C4 の計算式は $A $1＋5 のままである。

(2) 絶対参照は行と列の一方だけについても指定可能であり，$A 1，A $1 などと表す。例えば，セル D2 に計算式 $ C1－3 を入力した場合，セル D2 をセル E3 に複写すると，セル E3 の計算式は $ C2－3 とな

る。また，セル G3 に計算式 F $2－3 を入力した場合，セル G3 を H4 に複写すると，セル H4 の計算式は G $2－3 となる。

10. **マクロ**

(1) ワークシートには幾つかのマクロを保存できる。マクロはマクロ P，マクロ Q などと表す。

(2) マクロについては“ マクロ P を実行するとワークシートを保存する。”,“ セル A1 からセル A10 までを昇順に並べ替える手続をマクロ Q に登録する。”,“ マクロ R ：数値を入力。”,“ C 列のデータがその数値以下のものを抽出する。” などと記述する。

11. **その他**

ワークシートの“ 保存 ”,“ 読出し ”,“ 印刷 ” や，罫線機能，グラフ化機能など市販されている多くの表計算ソフトに備わっている機能は使用できるものとする。

Ver 1.0：平成 20 年 10 月 27 日

■試験で使用する情報技術に関する用語・プログラム言語など■

IPA 独立行政法人 情報処理推進機構
IT人材育成本部 情報処理技術者試験センター

〒113-8663　東京都文京区本駒込 2-28-8
文京グリーンコートセンターオフィス 15 階
TEL 03-5978-7600（代表）
FAX 03-5978-7610

詳しくは･･･
http://www.jitec.ipa.go.jp