

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 2.0

| | |
|------------------------------------|----|
| 1. 情報技術に関する用語 | 1 |
| 2. 記号・図など | 1 |
| 3. プログラム言語 | 1 |
| 4. データベース言語 | 1 |
| 5. マーク付け言語（マークアップ言語） | 1 |
| 6. 表計算ソフトなどのソフトウェアパッケージ | 2 |
| 別紙1 アセンブラ言語の仕様 | 3 |
| 別紙2 プログラム言語 Perl の用例・解説 | 11 |
| 別紙3 表計算ソフトの機能・用語（ITパスポート試験用） | 18 |
| 別紙4 表計算ソフトの機能・用語（基本情報技術者試験用） | 22 |

平成 23 年 7 月 11 日

■ 改訂履歴

【Ver 2.0】 平成 23 年 7 月 11 日

| ページ | 変更点 |
|-------|-------------------|
| 18~27 | 「表計算ソフトの機能・用語」の改訂 |

【Ver 1.0】 平成 20 年 10 月 27 日 初版

本冊子に記載されている会社名又は製品名は、それぞれ各社の商標又は登録商標です。
なお、本冊子では、TM 及び®を明記していません。

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（IT パスポート試験用）」（18 ページ），
及び「別紙 4 表計算ソフトの機能・用語（基本情報技術者試験用）」（22 ページ）による。表計算ソフトの機能・用語（基本情報技術者試験用）は，
表計算ソフトの機能・用語（IT パスポート試験用）の内容を包含している。
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）>

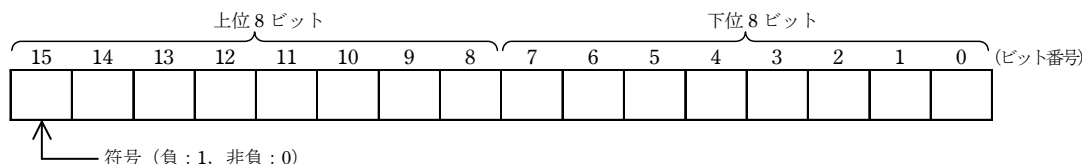
URL <http://www.jisc.go.jp/>

別紙 1 アセンブラ言語の仕様

1. システム 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、それ以外るとき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、それ以外るとき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、それ以外るとき0になる。 |
| ZF | 演算結果が零 (全部のビットが0) のとき1、それ以外るとき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 r, adr [, x] | r1 ← (r2) r ← (実効アドレス) | ○*1 |
| 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 | $r1, r2$ $r, \text{adr} [, x]$ | $r1 \leftarrow (r1) + (r2)$ $r \leftarrow (r) + (\text{実効アドレス})$ | ○ |
| 論理加算 ADD Logical | ADDL | $r1, r2$ $r, \text{adr} [, x]$ | $r1 \leftarrow (r1) +_L (r2)$ $r \leftarrow (r) +_L (\text{実効アドレス})$ | |
| 算術減算 SUBtract Arithmetic | SUBA | $r1, r2$ $r, \text{adr} [, x]$ | $r1 \leftarrow (r1) - (r2)$ $r \leftarrow (r) - (\text{実効アドレス})$ | |
| 論理減算 SUBtract Logical | SUBL | $r1, r2$ $r, \text{adr} [, x]$ | $r1 \leftarrow (r1) -_L (r2)$ $r \leftarrow (r) -_L (\text{実効アドレス})$ | |
| 論理積 AND | AND | $r1, r2$ $r, \text{adr} [, x]$ | $r1 \leftarrow (r1) \text{ AND } (r2)$ $r \leftarrow (r) \text{ AND } (\text{実効アドレス})$ | ○*1 |
| 論理和 OR | OR | $r1, r2$ $r, \text{adr} [, x]$ | $r1 \leftarrow (r1) \text{ OR } (r2)$ $r \leftarrow (r) \text{ OR } (\text{実効アドレス})$ | |
| 排他的論理和 eXclusive OR | XOR | $r1, r2$ $r, \text{adr} [, x]$ | $r1 \leftarrow (r1) \text{ XOR } (r2)$ $r \leftarrow (r) \text{ XOR } (\text{実効アドレス})$ | |

(3) 比較演算命令

| | | | | | | |
|----------------------------|-----|-----------------------------------|--|-----|-------|----|
| 算術比較 ComPare Arithmetic | CPA | $r1, r2$ $r, \text{adr} [, x]$ | ($r1$) 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 | | |
| 論理比較 ComPare Logical | CPL | $r1, r2$ $r, \text{adr} [, x]$ | 比較結果 | | FR の値 | |
| | | | | | SF | ZF |
| | | | $(r1) > (r2)$ | | 0 | 0 |
| | | | $(r) > (\text{実効アドレス})$ | | 0 | 1 |
| | | | $(r1) = (r2)$ | | 0 | 1 |
| | | | $(r) = (\text{実効アドレス})$ | 0 | 1 | |
| | | | $(r1) < (r2)$ | 1 | 0 | |
| | | | $(r) < (\text{実効アドレス})$ | 1 | 0 | |

(4) シフト演算命令

| | | | | |
|----------------------------------|-----|-----------------------|---|-----|
| 算術左シフト Shift Left Arithmetic | SLA | $r, \text{adr} [, x]$ | 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。 | ○*2 |
| 算術右シフト Shift Right Arithmetic | SRA | $r, \text{adr} [, x]$ | 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。 | |
| 論理左シフト Shift Left Logical | SLL | $r, \text{adr} [, x]$ | 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。 | |
| 論理右シフト Shift Right Logical | SRL | $r, \text{adr} [, x]$ | シフトの結果, 空いたビット位置には 0 が入る。 | |

(5) 分岐命令

| 正分岐 Jump on Plus | JPL | $\text{adr} [, x]$ | 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 | $\text{adr} [, x]$ | <table border="1"> <thead> <tr> <th rowspan="2">命令</th> <th colspan="3">分岐するときの FR の値</th> </tr> <tr> <th>OF</th> <th>SF</th> <th>ZF</th> </tr> </thead> <tbody> <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> </tbody> </table> | | 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 | | |
| 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 非零分岐 Jump on Non Zero | JNZ | $\text{adr} [, x]$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 零分岐 Jump on Zero | JZE | $\text{adr} [, x]$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| オーバフロー分岐 Jump on Overflow | JOV | $\text{adr} [, x]$ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 無条件分岐 unconditional JUMP | JUMP | $\text{adr} [, x]$ | 無条件に実効アドレスに分岐する。 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

(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

| | | | |
|-------------------------------|-----------------|---|---|
| スーパーバイザコール 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 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 (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 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 ($0000 \leq h \leq FFFF$)。 |
| 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関係処理を行いアドレスを決定する（プログラムの関係）。
- (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。

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 | | |
| 主OP | 副OP | r/r1 | x/r2 | | 機械語命令 | 意味 | |
| 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 |
| 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。

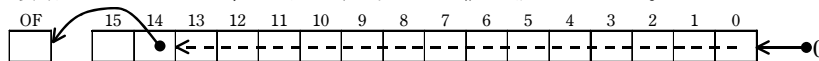
〔例〕 IN 命令

| | | |
|-------|------|-----------|
| 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 |

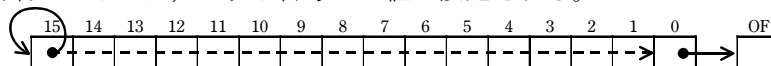
3. シフト演算命令におけるビットの動き

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。

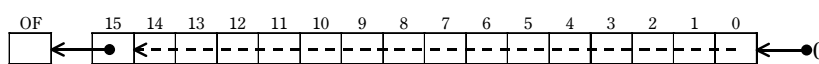
- (1) 算術左シフトでは、ビット番号 14 の値が設定される。



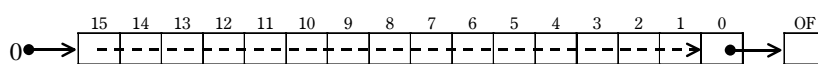
- (2) 算術右シフトでは、ビット番号 0 の値が設定される。



- (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り返し
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。 |
| | <code>\$var = "hello"; print '\$var ', "\$var ", `echo world`;</code> ----- 変数 <code>var</code> に文字列 "hello" を代入する。文字列のスカラ <code>'\$var '</code> , <code>"\$var "</code> , <code>`echo world`</code> を出力する。 <code>"\$var "</code> は変数を展開し、 <code>`echo world`</code> はコマンドの出力を展開するので、出力は <code>"\$var hello world"</code> となる。 |
| | <code>\n</code> ----- 制御文字 (改行) である。 |
| | <code>\r</code> ----- 制御文字 (復帰) である。 |
| | <code>\t</code> ----- 制御文字 (水平タブ) である。 |
| リストリテラル | <code>('a', 'b', 'c')</code> ----- リスト ('a', 'b', 'c') である。 |
| | <code>('a', 'b', 'c')[0]</code> ----- リスト ('a', 'b', 'c') の 1 番目の要素 'a' である。 |
| | <code>()</code> ----- 空リストである。 |
| | <code>('a' => 'alpha', 'b' => 'bravo', 'c' => 'charlie')</code> ----- 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

| | |
|--------------|--|
| スカラー変数 | \$var スカラー変数 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。 |

| | |
|---|--|
| <code>=~, !~</code> | <pre>\$html_contents =~ //</pre> <p>変数 <code>html_contents</code> の値に、文字列 “” が含まれているときに真を返す。</p> <pre>\$html_contents !~ / /</pre> <p>変数 <code>html_contents</code> の値に、文字列 “ ” が含まれていないときに真を返す。</p> |
| <code>*, /, %</code> | <pre>314 * 34</pre> <p>314 と 34 の乗算である。</p> <pre>6 / 469</pre> <p>6 を 469 で割る除算である。</p> <pre>34 % 6</pre> <p>34 を 6 で割る剰余演算である。</p> |
| <code>+, -, .</code> | <pre>3.14 + 2.72</pre> <p>3.14 と 2.72 の加算である。</p> <pre>220 - 8125</pre> <p>220 から 8125 を引く減算である。</p> <pre>"IPA"."JITEC"</pre> <p>文字列 “IPA” と “JITEC” の連結である。</p> |
| <code><, >, <=, >=, lt, gt, le, ge</code> | <pre>1 < 2</pre> <p>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 <code>></code>, <code><=</code>, <code>>=</code> がある。</p> <pre>"b" lt "a"</pre> <p>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 <code>gt</code>, <code>le</code>, <code>ge</code> がある。</p> |
| <code>==, !=, <=>, eq, ne, cmp</code> | <pre>1 <=> 2</pre> <p>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 <code>==</code>, <code>!=</code> がある。</p> <pre>"b" cmp "a"</pre> <p>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 <code>eq</code>, <code>ne</code> がある。</p> |
| <code>&&</code> | <pre>\$x >= 0 && \$x < 10</pre> <p>変数 <code>x</code> の値が 0 以上かつ 10 未満なら真を返す。</p> |
| <code> </code> | <pre>\$x < 0 \$x >= 10</pre> <p>変数 <code>x</code> の値が 0 未満又は 10 以上なら真を返す。</p> |
| <code>..</code> | <pre>@card = (1 .. 52)</pre> <p>1 から 52 までの連続する整数を配列変数 <code>card</code> に代入する。</p> |
| <code>=, +=, -=, *=, /=, %=</code> | <pre>\$a = 1</pre> <p>変数 <code>a</code> に 1 を代入する。</p> <pre>\$a += 10</pre> <p>変数 <code>a</code> の値に 10 を加算して <code>a</code> に代入する。 代入演算子には、ほかに <code>-=</code>, <code>*=</code>, <code>/=</code>, <code>%=</code> がある。</p> |
| <code>=>, ,</code> | <pre>%hash = ('a' => 'alpha', 'b' => 'bravo', 'c' => 'charlie')</pre> <p><code>a</code> に <code>alpha</code>, <code>b</code> に <code>bravo</code>, <code>c</code> に <code>charlie</code> を結び付けたハッシュをハッシュ変数 <code>hash</code> に代入する。</p> |
| <code>not</code> | <pre>not \$a</pre> <p>変数 <code>a</code> の論理否定である。</p> |

| | |
|------------|--|
| and | <code>\$a < 0 and \$b == 0</code> ----- 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 | <code>\$a < 0 or \$b == 0</code> ----- 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。 <code>\$a < 0 xor \$b == 0</code> ----- 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 | <pre>if (\$var == 1) { print "a"; } elseif (\$var == 2) { print "b"; } else { print "c"; }</pre> ----- 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 | <pre>\$i = 1; while (\$i <= 10) { print \$i++, "\n"; }</pre> ----- 変数 i の値を 1 から 1 ずつ増やし、10 回出力する。 |
| for | <pre>for (\$i = 1; \$i <= 10; \$i++){ print "\$i\n"; }</pre> ----- 変数 i の値を 1 から 1 ずつ増やし、10 回出力する。 |
| foreach | <pre>foreach \$i (1, 3, 5) { print "\$i\n"; }</pre> ----- 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 | <pre>for (\$i = 1; \$i <= 10; \$i++) { next if \$i % 2; print "\$i\n"; }</pre> ----- 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、偶数を実出力する。 |

6. 正規表現

| | |
|---|--|
| \ | <code>/\.\^\\$[\ \\+*\?{\(\)\}\ \ /</code> ----- 次の 1 文字そのものを表す。“ <code>.\^\\$[\ \\+*\?{\(\)\}\ \ </code> ” と一致する。 |
| . | <code>/www.ipa.go.jp/</code> ----- 改行文字以外の任意の 1 文字と一致する。“ <code>wwwdipa,go@jp</code> ” と一致する。 |
| ^ | <code>/^ab/</code> ----- 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。 |

| | |
|------------|--|
| \$ | <code>/yz\$/</code> 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。 |
| + | <code>/go+d/</code> 直前の 1 文字 o の 1 回以上の繰返しと一致する。“god”や“goood”と一致するが、“gd”とは一致しない。 |
| * | <code>/go*d/</code> 直前の 1 文字 o の 0 回以上の繰返しと一致する。“gd”、“god”や“goood”と一致する。 |
| ? | <code>/colou?r/</code> 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} | <code>/co{2}l/</code> 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。 <code>/go{1,3}d/</code> 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“goood”とは一致しない。 |
| (…) | <code><(h.)>/</code> () 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, … | <code><(.)><([bp])>JITEC<\/\2><\/\1>/</code> 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。 |
| […] | <code><h[12r]>/</code> [] 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。 <code>/[^0-9]/</code> [] 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。 |
| … … | <code><(a href img src)=/</code> 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 | <pre>sub greeting { print "hello Perl\n"; }</pre> “hello Perl”を出力するサブルーチン <code>greeting</code> を定義する。 |
| 呼出し | <code>subroutine (\$arg1, \$arg2);</code> サブルーチン <code>subroutine</code> を引数 <code>arg1</code> と <code>arg2</code> で呼び出す。() を省略して“ <code>subroutine \$arg1, \$arg2;</code> ”とする表記もある。 |
| 戻り | <code>return -1;</code> サブルーチンから抜け出し、値 -1 を返す。 |

8. モジュール

| | |
|-----|---|
| use | <pre>use CGI;</pre> <p>モジュール CGI を 1 度だけ読み込み，利用可能にする。</p> |
|-----|---|

9. メソッド呼出し

| | |
|----|---|
| -> | <pre>\$object->method1(arg1);</pre> <p>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。</p> |
| | <pre>Class->method2(arg1, arg2);</pre> <p>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。</p> |

10. 文字列操作関数

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

11. 配列・ハッシュ操作関数

| | |
|-------|--|
| keys | <pre>%hash = ('a' => 'alpha', 'b' => 'bravo', 'c' => 'charlie'); foreach \$key (keys %hash) { print "\$key\n"; }</pre> <p>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。</p> |
| shift | <pre>\$next = shift @queue;</pre> <p>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。</p> |
| sort | <pre>@pile = sort @jumble;</pre> <p>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。</p> <pre>@pile = sort {\$b <=> \$a} @jumble;</pre> <p>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。</p> |
| split | <pre>@fields = split ',', \$csv;</pre> <p>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。</p> |

12. 検索・置換関数

| | |
|-------------|--|
| m/…/ 又は /…/ | <pre>\$html_contents =~ //i;</pre> <p>変数 html_contents の値が，文字列 “” 又は “” 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。</p> |
| s/…/…/ | <pre>\$html_contents =~ s/ /\n/gi;</pre> <p>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。</p> |

| | |
|------------------------------|--|
| \$`, \$&, \$', \$1, \$2, ... | <pre>'The date is 1970-01-23.' =~ /([0-9]{4})-([0-9]{2})-([0-9]{2})/;</pre> <pre>print "String before the date: \$\`n";</pre> <pre>print "Date: \$&n";</pre> <pre>print "String after the date: \$('n";</pre> <pre>print "Year: \$1\n", "Month: \$2\n", "Day: \$3\n";</pre> <p>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。</p> |
|------------------------------|--|

13. 入出力操作関数

| | |
|--------------|---|
| open | <pre>open LOG, '>>cgi.log';</pre> <p>ファイル <code>cgi.log</code> を追記モードで開き、ファイルハンドル <code>LOG</code> に対応付ける。</p> |
| <filehandle> | <pre>\$line = <USER_FILE>;</pre> <p>ファイルハンドル <code>USER_FILE</code> から 1 行を読み込んで変数 <code>line</code> に代入する。</p> |
| <> | <pre>@records = <>;</pre> <p>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 <code>records</code> に代入する。</p> |
| print | <pre>print LOG "sync.\n";</pre> <p>ファイルハンドル <code>LOG</code> に対応するファイルに文字列を出力する。</p> |
| close | <pre>close LOG;</pre> <p>ファイルハンドル <code>LOG</code> に対応するファイルを閉じる。</p> |

14. システムインタフェース

| | |
|--------|--|
| die | <pre>open(FILE, 'a_file') or die 'cannot open a_file';</pre> <p>ファイル <code>a_file</code> 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。</p> |
| system | <pre>system 'a.out';</pre> <p>コマンド <code>a.out</code> を実行し、コマンドが終了するまで待機する。</p> |

別紙3 表計算ソフトの機能・用語 (ITパスポート試験用)

表計算ソフトの機能，用語などは，原則として次による。

なお，ワークシートの保存，読出し，印刷，罫線作成やグラフ作成など，ここで示す以外の機能などを使用するときには，問題文中に示す。

1. ワークシート

- (1) 列と行とで構成される昇目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，10,000 行とする。
- (2) ワークシートの列と行のそれぞれの位置は，列番号と行番号で表す。列番号は，最左端列の列番号を A とし，A，B，…，Z，AA，AB，…，AZ，BA，BB，…，BZ，…，IU，IV と表す。行番号は，最上端行の行番号を 1 とし，1，2，…，10000 と表す。
- (3) 複数のワークシートを利用することができる。このとき，各ワークシートには一意のワークシート名を付けて，他のワークシートと区別する。

2. セルとセル範囲

- (1) ワークシートを構成する各升をセルという。その位置は列番号と行番号で表し，それをセル番地という。
[例] 列 A 行 1 にあるセルのセル番地は，A1 と表す。
- (2) ワークシート内のある長方形の領域に含まれる全てのセルの集まりを扱う場合，長方形の左上端と右下端のセル番地及び“～”を用いて，“左上端のセル番地～右下端のセル番地”と表す。これを，セル範囲という。
[例] 左上端のセル番地が A1 で，右下端のセル番地が B3 のセル範囲は，A1～B3 と表す。
- (3) 他のワークシートのセル番地又はセル範囲を指定する場合には，ワークシート名と“!”を用い，それぞれ“ワークシート名!セル番地”又は“ワークシート名!セル範囲”と表す。
[例] ワークシート“シート1”のセル範囲 B5～G10 を，別のワークシートから指定する場合には，シート1!B5～G10 と表す。


3. 値と式

- (1) セルは値をもち，その値はセル番地によって参照できる。値には，数値，文字列，論理値及び空値がある。
- (2) 文字列は一重引用符“'”で囲って表す。
[例] 文字列“A”，“BC”は，それぞれ'A'，'BC' と表す。
- (3) 論理値の真を true，偽を false と表す。
- (4) 空値を null と表し，空値をもつセルを空白セルという。セルの初期状態は，空白セルとする。
- (5) セルには，式を入力することができる。セルは，式を評価した結果の値をもつ。

- (6) 式は、定数、セル番地、演算子、括弧及び関数から構成される。定数は、数値、文字列、論理値又は空値を表す表記とする。式中のセル番地は、その番地のセルの値を参照する。
- (7) 式には、算術式、文字式及び論理式がある。評価の結果が数値となる式を算術式、文字列となる式を文字式、論理値となる式を論理式という。
- (8) セルに式を入力すると、式は直ちに評価される。式が参照するセルの値が変化したときには、直ちに、適切に再評価される。

4. 演算子

- (1) 単項演算子は、正符号“+”及び負符号“-”とする。
- (2) 算術演算子は、加算“+”，減算“-”，乗算“*”，除算“/”及びべき乗“^”とする。
- (3) 比較演算子は、より大きい“>”，より小さい“<”，以上“>=”，以下“<=”，等しい“=”及び等しくない“<>”とする。
- (4) 括弧は丸括弧“()”を使う。
- (5) 式中に複数の演算及び括弧があるときの計算の順序は、次表の優先順位に従う。

| 演算の種類 | 演算子 | 優先順位 |
|-------|----------------|---|
| 括弧 | () | 高  低 |
| べき乗演算 | ^ | |
| 単項演算 | + , - | |
| 乗除演算 | * , / | |
| 加減演算 | + , - | |
| 比較演算 | > , < , = , <> | |

5. セルの複写

- (1) セルの値又は式を、他のセルに複写することができる。
- (2) セルを複写する場合で、複写元のセル中にセル番地を含む式が入力されているとき、複写元と複写先のセル番地の差を維持するように、式中のセル番地を変化させるセルの参照方法を相対参照という。この場合、複写先のセルとの列番号の差及び行番号の差を、複写元のセルに入力された式中の各セル番地に加算した式が、複写先のセルに入る。

[例] セルA6に式 A1 + 5 が入力されているとき、このセルをセルB8に複写すると、セルB8には式 B3 + 5 が入る。

- (3) セルを複写する場合で、複写元のセル中にセル番地を含む式が入力されているとき、そのセル番地の列番号と行番号の両方又は片方を変化させないセルの参照方法を絶対参照という。絶対参照を適用する列番号と行番号の両方又は片方の直前には“\$”を付ける。

[例] セルB1に式 \$A\$1 + \$A2 + A\$5 が入力されているとき、このセルをセルC4に複写すると、セルC4には式 \$A\$1 + \$A5 + B\$5 が入る。

(4) セルを複写する場合で、複写元のセル中に、他のワークシートを参照する式が入力されているとき、その参照するワークシートのワークシート名は複写先でも変わらない。

[例] ワークシート“シート2”のセルA6に式 シート1!A1 が入力されているとき、このセルをワークシート“シート3”のセルB8に複写すると、セルB8には式 シート1!B3 が入る。

6. 関数

式には次の表で定義する関数を利用することができる。

| 書式 | 解 説 |
|------------------------------------|--|
| 合計(セル範囲 ¹⁾) | セル範囲に含まれる数値の合計を返す。 [例] 合計(A1 ~ B5) は、セル範囲 A1 ~ B5 に含まれる数値の合計を返す。 |
| 平均(セル範囲 ¹⁾) | セル範囲に含まれる数値の平均を返す。 |
| 標本標準偏差(セル範囲 ¹⁾) | セル範囲に含まれる数値を標本として計算した標準偏差を返す。 |
| 母標準偏差(セル範囲 ¹⁾) | セル範囲に含まれる数値を母集団として計算した標準偏差を返す。 |
| 最大(セル範囲 ¹⁾) | セル範囲に含まれる数値の最大値を返す。 |
| 最小(セル範囲 ¹⁾) | セル範囲に含まれる数値の最小値を返す。 |
| IF(論理式, 式1, 式2) | 論理式の値が true のとき式 1 の値を、false のとき式 2 の値を返す。 [例] IF(B3 > A4, '北海道', C4) は、セル B3 の値がセル A4 の値より大きいとき文字列“北海道”を、それ以外るときセル C4 の値を返す。 |
| 個数(セル範囲) | セル範囲に含まれるセルのうち、空白セルでないセルの個数を返す。 |
| 条件付個数(セル範囲, 検索条件の記述) | セル範囲に含まれるセルのうち、検索条件の記述で指定された条件を満たすセルの個数を返す。検索条件の記述は比較演算子と式の組で記述し、セル範囲に含まれる各セルと式の値を、指定した比較演算子によって評価する。 [例1] 条件付個数(H5 ~ L9, > A1) は、セル範囲 H5 ~ L9 のセルのうち、セル A1 の値より大きな値をもつセルの個数を返す。 [例2] 条件付個数(H5 ~ L9, = 'A4') は、セル範囲 H5 ~ L9 のセルのうち、文字列“A4”をもつセルの個数を返す。 |
| 整数部(算術式) | 算術式の値以下で最大の整数を返す。 [例1] 整数部(3.9) は、3 を返す。 [例2] 整数部(-3.9) は、-4 を返す。 |
| 剰余(算術式1, 算術式2) | 算術式1の値を被除数、算術式2の値を除数として除算を行ったときの剰余を返す。関数“剰余”と“整数部”は、剰余(x,y) = x - y * 整数部(x / y) という関係を満たす。 [例1] 剰余(10,3) は、1 を返す。 [例2] 剰余(-10,3) は、2 を返す。 |
| 平方根(算術式) | 算術式の値の非負の平方根を返す。算術式の値は、非負の数値でなければならない。 |
| 論理積(論理式1, 論理式2, ...) ²⁾ | 論理式1, 論理式2, ... の値が全て true のとき、true を返す。それ以外るとき false を返す。 |
| 論理和(論理式1, 論理式2, ...) ²⁾ | 論理式1, 論理式2, ... の値のうち、少なくとも一つが true のとき、true を返す。それ以外るとき false を返す。 |
| 否定(論理式) | 論理式の値が true のとき false を、false のとき true を返す。 |

| | |
|--------------------------------------|---|
| 切上げ (算術式, 桁位置) | 算術式の値を指定した桁位置で、関数“切上げ”は切り上げた値を、関数“四捨五入”は四捨五入した値を、関数“切捨て”は切り捨てた値を返す。ここで、桁位置は小数第1位の桁を0とし、右方向を正として数えたときの位置とする。 [例1] 切上げ (-314.159, 2) は、-314.16 を返す。 [例2] 切上げ (314.159, -2) は、400 を返す。 [例3] 切上げ (314.159, 0) は、315 を返す。 |
| 四捨五入 (算術式, 桁位置) | |
| 切捨て (算術式, 桁位置) | |
| 結合 (式1, 式2, ...) ²⁾ | 式1, 式2, ... のそれぞれの値を文字列として扱い、それらを引数の順につないでできる一つの文字列を返す。 [例] 結合 ('北海道', '九州', 123, 456) は、文字列“北海道九州123456”を返す。 |
| 順位 (算術式, セル範囲 ¹⁾ , 順序の指定) | セル範囲の中での算術式の値の順位を、順序の指定が0の場合は昇順で、1の場合は降順で数えて、その順位を返す。ここで、セル範囲の中に同じ値がある場合、それらを同順とし、次の順位は同順の個数だけ加算した順位とする。 |
| 乱数 () | 0 以上 1 未満の一樣乱数 (実数値) を返す。 |
| 表引き (セル範囲, 行の位置, 列の位置) | セル範囲の左上端から行と列をそれぞれ 1, 2, ... と数え、セル範囲に含まれる行の位置と列の位置で指定した場所にあるセルの値を返す。 [例] 表引き (A3 ~ H11, 2, 5) は、セル E4 の値を返す。 |
| 垂直照合 (式, セル範囲, 列の位置, 検索の指定) | セル範囲の左端列を上から下に走査し、検索の指定によって指定される条件を満たすセルが現れる最初の行を探す。その行に対して、セル範囲の左端列から列を 1, 2, ... と数え、セル範囲に含まれる列の位置で指定した列にあるセルの値を返す。 ・検索の指定が0の場合の条件：式の値と一致する値を検索する。 ・検索の指定が1の場合の条件：式の値以下の最大値を検索する。このとき、左端列は上から順に昇順に整列されている必要がある。 [例] 垂直照合 (15, A2 ~ E10, 5, 0) は、セル範囲の左端列をセル A2, A3, ..., A10 と探す。このとき、セル A6 で 15 を最初に見つけたとすると、左端列 A から数えて 5 列目の列 E 中で、セル A6 と同じ行にあるセル E6 の値を返す。 |
| 水平照合 (式, セル範囲, 行の位置, 検索の指定) | セル範囲の上端行を左から右に走査し、検索の指定によって指定される条件を満たすセルが現れる最初の列を探す。その列に対して、セル範囲の上端行から行を 1, 2, ... と数え、セル範囲に含まれる行の位置で指定した行にあるセルの値を返す。 ・検索の指定が0の場合の条件：式の値と一致する値を検索する。 ・検索の指定が1の場合の条件：式の値以下の最大値を検索する。このとき、上端行は左から順に昇順に整列されている必要がある。 [例] 水平照合 (15, A2 ~ G6, 5, 1) は、セル範囲の上端行をセル A2, B2, ..., G2 と探す。このとき、15 以下の最大値をセル D2 で最初に見つけたとすると、上端行 2 から数えて 5 行目の行 6 中で、セル D2 と同じ列にあるセル D6 の値を返す。 |

注 ¹⁾ 引数として渡したセル範囲の中で、数値以外の値は処理の対象としない。

²⁾ 引数として渡すことができる式の個数は、1 以上である。

別紙4 表計算ソフトの機能・用語（基本情報技術者試験用）

表計算ソフトの機能，用語などは，原則として次による。

なお，ワークシートの保存，読出し，印刷，罫線作成やグラフ作成など，ここで示す以外の機能などを使用するときには，問題文中に示す。

1. ワークシート

- (1) 列と行とで構成される升目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，10,000 行とする。
- (2) ワークシートの列と行のそれぞれの位置は，列番号と行番号で表す。列番号は，最左端列の列番号を A とし，A，B，…，Z，AA，AB，…，AZ，BA，BB，…，BZ，…，IU，IV と表す。行番号は，最上端行の行番号を 1 とし，1，2，…，10000 と表す。
- (3) 複数のワークシートを利用することができる。このとき，各ワークシートには一意のワークシート名を付けて，他のワークシートと区別する。

2. セルとセル範囲

- (1) ワークシートを構成する各升をセルという。その位置は列番号と行番号で表し，それをセル番地という。
[例] 列 A 行 1 にあるセルのセル番地は，A1 と表す。
- (2) ワークシート内のある長方形の領域に含まれる全てのセルの集まりを扱う場合，長方形の左上端と右下端のセル番地及び“～”を用いて，“左上端のセル番地～右下端のセル番地”と表す。これを，セル範囲という。
[例] 左上端のセル番地が A1 で，右下端のセル番地が B3 のセル範囲は，A1～B3 と表す。
- (3) 他のワークシートのセル番地又はセル範囲を指定する場合には，ワークシート名と“!”を用い，それぞれ“ワークシート名!セル番地”又は“ワークシート名!セル範囲”と表す。
[例] ワークシート“シート1”のセル範囲 B5～G10 を，別のワークシートから指定する場合には，シート1!B5～G10 と表す。

3. 値と式

- (1) セルは値をもち，その値はセル番地によって参照できる。値には，数値，文字列，論理値及び空値がある。
- (2) 文字列は一重引用符“'”で囲って表す。
[例] 文字列“A”，“BC”は，それぞれ'A'，'BC' と表す。
- (3) 論理値の真を true，偽を false と表す。
- (4) 空値を null と表し，空値をもつセルを空白セルという。セルの初期状態は，空白セルとする。
- (5) セルには，式を入力することができる。セルは，式を評価した結果の値をもつ。

- (6) 式は、定数、セル番地、演算子、括弧及び関数から構成される。定数は、数値、文字列、論理値又は空値を表す表記とする。式中のセル番地は、その番地のセルの値を参照する。
- (7) 式には、算術式、文字式及び論理式がある。評価の結果が数値となる式を算術式、文字列となる式を文字式、論理値となる式を論理式という。
- (8) セルに式を入力すると、式は直ちに評価される。式が参照するセルの値が変化したときには、直ちに、適切に再評価される。

4. 演算子

- (1) 単項演算子は、正符号“+”及び負符号“-”とする。
- (2) 算術演算子は、加算“+”，減算“-”，乗算“*”，除算“/”及びべき乗“^”とする。
- (3) 比較演算子は、より大きい“>”，より小さい“<”，以上“>=”，以下“<=”，等しい“=”及び等しくない“<>”とする。
- (4) 括弧は丸括弧“()”を使う。
- (5) 式中に複数の演算及び括弧があるときの計算の順序は、次表の優先順位に従う。

| 演算の種類 | 演算子 | 優先順位 |
|-------|----------------|---|
| 括弧 | () | 高  低 |
| べき乗演算 | ^ | |
| 単項演算 | + , - | |
| 乗除演算 | * , / | |
| 加減演算 | + , - | |
| 比較演算 | > , < , = , <> | |

5. セルの複写

- (1) セルの値又は式を、他のセルに複写することができる。
- (2) セルを複写する場合で、複写元のセル中にセル番地を含む式が入力されているとき、複写元と複写先のセル番地の差を維持するように、式中のセル番地を変化させるセルの参照方法を相対参照という。この場合、複写先のセルとの列番号の差及び行番号の差を、複写元のセルに入力された式中の各セル番地に加算した式が、複写先のセルに入る。

[例] セルA6に式 A1 + 5 が入力されているとき、このセルをセルB8に複写すると、セルB8には式 B3 + 5 が入る。

- (3) セルを複写する場合で、複写元のセル中にセル番地を含む式が入力されているとき、そのセル番地の列番号と行番号の両方又は片方を変化させないセルの参照方法を絶対参照という。絶対参照を適用する列番号と行番号の両方又は片方の直前には“\$”を付ける。

[例] セルB1に式 \$A\$1 + \$A2 + A\$5 が入力されているとき、このセルをセルC4に複写すると、セルC4には式 \$A\$1 + \$A5 + B\$5 が入る。

(4) セルを複写する場合で、複写元のセル中に、他のワークシートを参照する式が入力されているとき、その参照するワークシートのワークシート名は複写先でも変わらない。

[例] ワークシート“シート2”のセルA6に式 シート1!A1 が入力されているとき、このセルをワークシート“シート3”のセルB8に複写すると、セルB8には式 シート1!B3 が入る。

6. 関数

式には次の表で定義する関数を利用することができる。

| 書式 | 解 説 |
|------------------------------------|--|
| 合計(セル範囲 ¹⁾) | セル範囲に含まれる数値の合計を返す。 [例] 合計(A1 ~ B5) は、セル範囲 A1 ~ B5 に含まれる数値の合計を返す。 |
| 平均(セル範囲 ¹⁾) | セル範囲に含まれる数値の平均を返す。 |
| 標本標準偏差(セル範囲 ¹⁾) | セル範囲に含まれる数値を標本として計算した標準偏差を返す。 |
| 母標準偏差(セル範囲 ¹⁾) | セル範囲に含まれる数値を母集団として計算した標準偏差を返す。 |
| 最大(セル範囲 ¹⁾) | セル範囲に含まれる数値の最大値を返す。 |
| 最小(セル範囲 ¹⁾) | セル範囲に含まれる数値の最小値を返す。 |
| IF(論理式, 式1, 式2) | 論理式の値が true のとき式 1 の値を、false のとき式 2 の値を返す。 [例] IF(B3 > A4, '北海道', C4) は、セル B3 の値がセル A4 の値より大きいとき文字列“北海道”を、それ以外の場合セル C4 の値を返す。 |
| 個数(セル範囲) | セル範囲に含まれるセルのうち、空白セルでないセルの個数を返す。 |
| 条件付個数(セル範囲, 検索条件の記述) | セル範囲に含まれるセルのうち、検索条件の記述で指定された条件を満たすセルの個数を返す。検索条件の記述は比較演算子と式の組で記述し、セル範囲に含まれる各セルと式の値を、指定した比較演算子によって評価する。 [例1] 条件付個数(H5 ~ L9, > A1) は、セル範囲 H5 ~ L9 のセルのうち、セル A1 の値より大きな値をもつセルの個数を返す。 [例2] 条件付個数(H5 ~ L9, = 'A4') は、セル範囲 H5 ~ L9 のセルのうち、文字列“A4”をもつセルの個数を返す。 |
| 整数部(算術式) | 算術式の値以下で最大の整数を返す。 [例1] 整数部(3.9) は、3 を返す。 [例2] 整数部(-3.9) は、-4 を返す。 |
| 剰余(算術式1, 算術式2) | 算術式1の値を被除数、算術式2の値を除数として除算を行ったときの剰余を返す。関数“剰余”と“整数部”は、剰余(x,y) = x - y * 整数部(x / y) という関係を満たす。 [例1] 剰余(10,3) は、1 を返す。 [例2] 剰余(-10,3) は、2 を返す。 |
| 平方根(算術式) | 算術式の値の非負の平方根を返す。算術式の値は、非負の数値でなければならない。 |
| 論理積(論理式1, 論理式2, ...) ²⁾ | 論理式1, 論理式2, ... の値が全て true のとき、true を返す。それ以外の場合 false を返す。 |
| 論理和(論理式1, 論理式2, ...) ²⁾ | 論理式1, 論理式2, ... の値のうち、少なくとも一つが true のとき、true を返す。それ以外の場合 false を返す。 |
| 否定(論理式) | 論理式の値が true のとき false を、false のとき true を返す。 |

| | |
|--------------------------------------|---|
| 切上げ (算術式, 桁位置) | 算術式の値を指定した桁位置で、関数“切上げ”は切り上げた値を、関数“四捨五入”は四捨五入した値を、関数“切捨て”は切り捨てた値を返す。ここで、桁位置は小数第1位の桁を0とし、右方向を正として数えたときの位置とする。 [例1] 切上げ (-314.159, 2) は、-314.16 を返す。 [例2] 切上げ (314.159, -2) は、400 を返す。 [例3] 切上げ (314.159, 0) は、315 を返す。 |
| 四捨五入 (算術式, 桁位置) | |
| 切捨て (算術式, 桁位置) | |
| 結合 (式1, 式2, ...) ² | 式1, 式2, ... のそれぞれの値を文字列として扱い、それらを引数の順につないでできる一つの文字列を返す。 [例] 結合 ('北海道', '九州', 123, 456) は、文字列“北海道九州123456”を返す。 |
| 順位 (算術式, セル範囲 ¹⁾ , 順序の指定) | セル範囲の中での算術式の値の順位を、順序の指定が0の場合は昇順で、1の場合は降順で数えて、その順位を返す。ここで、セル範囲の中に同じ値がある場合、それらを同順とし、次の順位は同順の個数だけ加算した順位とする。 |
| 乱数 () | 0 以上 1 未満の一樣乱数 (実数値) を返す。 |
| 表引き (セル範囲, 行の位置, 列の位置) | セル範囲の左上端から行と列をそれぞれ 1, 2, ... と数え、セル範囲に含まれる行の位置と列の位置で指定した場所にあるセルの値を返す。 [例] 表引き (A3 ~ H11, 2, 5) は、セル E4 の値を返す。 |
| 垂直照合 (式, セル範囲, 列の位置, 検索の指定) | セル範囲の左端列を上から下に走査し、検索の指定によって指定される条件を満たすセルが現れる最初の行を探す。その行に対して、セル範囲の左端列から列を 1, 2, ... と数え、セル範囲に含まれる列の位置で指定した列にあるセルの値を返す。 ・検索の指定が0の場合の条件：式の値と一致する値を検索する。 ・検索の指定が1の場合の条件：式の値以下の最大値を検索する。このとき、左端列は上から順に昇順に整列されている必要がある。 [例] 垂直照合 (15, A2 ~ E10, 5, 0) は、セル範囲の左端列をセル A2, A3, ..., A10 と探す。このとき、セル A6 で 15 を最初に見つけたとすると、左端列 A から数えて 5 列目の列 E 中で、セル A6 と同じ行にあるセル E6 の値を返す。 |
| 水平照合 (式, セル範囲, 行の位置, 検索の指定) | セル範囲の上端行を左から右に走査し、検索の指定によって指定される条件を満たすセルが現れる最初の列を探す。その列に対して、セル範囲の上端行から行を 1, 2, ... と数え、セル範囲に含まれる行の位置で指定した行にあるセルの値を返す。 ・検索の指定が0の場合の条件：式の値と一致する値を検索する。 ・検索の指定が1の場合の条件：式の値以下の最大値を検索する。このとき、上端行は左から順に昇順に整列されている必要がある。 [例] 水平照合 (15, A2 ~ G6, 5, 1) は、セル範囲の上端行をセル A2, B2, ..., G2 と探す。このとき、15 以下の最大値をセル D2 で最初に見つけたとすると、上端行 2 から数えて 5 行目の行 6 中で、セル D2 と同じ列にあるセル D6 の値を返す。 |
| 照合検索 (式, 検索のセル範囲, 抽出のセル範囲) | 1 行又は 1 列を対象とする同じ大きさの検索のセル範囲と抽出のセル範囲に対して、検索のセル範囲を左端又は上端から走査し、式の値と一致する最初のセルを探す。見つかったセルの検索のセル範囲の中での位置と、抽出のセル範囲の中での位置が同じセルの値を返す。 [例] 照合検索 (15, A1 ~ A8, C6 ~ C13) は、セル範囲 A1 ~ A8 をセル A1, A2, ... と探す。このとき、セル A5 で 15 を最初に見つけたとすると、セル範囲 C6 ~ C13 の上端から数えて 5 番目のセル C10 の値を返す。 |

| | |
|---|---|
| 照合一致(式, セル範囲, 検索の指定) | <p>1行又は1列を対象とするセル範囲に対して, セル範囲の左端又は上端から走査し, 検索の指定によって指定される条件を満たす最初のセルを探す。見つかったセルの位置を, セル範囲の左端又は上端から 1, 2, ... と数えた値とし, その値を返す。</p> <ul style="list-style-type: none"> ・ 検索の指定が 0 の場合の条件: 式の値と一致する値を検索する。 ・ 検索の指定が 1 の場合の条件: 式の値以下の最大値を検索する。このとき, セル範囲は左端又は上端から順に昇順に整列されている必要がある。 ・ 検索の指定が -1 の場合の条件: 式の値以上の最小値を検索する。このとき, セル範囲は左端又は上端から順に降順に整列されている必要がある。 <p>[例] 照合一致(15, B2 ~ B12, -1) は, セル範囲 B2 ~ B12 をセル B2, B3, ... と探す。このとき, 15 以上の最小値をセル B9 で最初に見つけたとすると, セル B2 から数えた値 8 を返す。</p> |
| 条件付合計(検索のセル範囲, 検索条件の記述, 合計のセル範囲 ¹⁾) | <p>行数及び列数が共に同じ検索のセル範囲と合計のセル範囲に対して, 検索と合計を行う。検索のセル範囲に含まれるセルのうち, 検索条件の記述で指定される条件を満たすセルを全て探す。検索条件の記述を満たした各セルについての左上端からの位置と, 合計のセル範囲中で同じ位置にある各セルの値を合計して返す。</p> <p>検索条件の記述は比較演算子と式の組で記述し, 検索のセル範囲に含まれる各セルと式の値を, 指定した比較演算子によって評価する。</p> <p>[例1] 条件付合計(A1 ~ B8, > E1, C2 ~ D9) は, 検索のセル範囲である A1 ~ B8 のうち, セル E1 の値より大きな値をもつ全てのセルを探す。このとき, セル A2, B4, B7 が見つかったとすると, 合計のセル範囲である C2 ~ D9 の左上端からの位置が同じであるセル C3, D5, D8 の値を合計して返す。</p> <p>[例2] 条件付合計(A1 ~ B8, = 160, C2 ~ D9) は, 検索のセル範囲である A1 ~ B8 のうち, 160 と一致する値をもつ全てのセルを探す。このとき, セル A2, B4, B7 が見つかったとすると, 合計のセル範囲である C2 ~ D9 の左上端からの位置が同じであるセル C3, D5, D8 の値を合計して返す。</p> |

注¹⁾ 引数として渡したセル範囲の中で, 数値以外の値は処理の対象としない。

²⁾ 引数として渡すことができる式の個数は, 1 以上である。

7. マクロ

(1) ワークシートとマクロ

ワークシートには複数のマクロを格納することができる。

マクロは一意的なマクロ名を付けて宣言する。マクロの実行は, 表計算ソフトのマクロの実行機能を使って行う。

[例] マクロ: Pro

例は, マクロ Pro の宣言である。

(2) 変数とセル変数

変数の型には, 数値型, 文字列型及び論理型があり, 変数は宣言することで使用できる。変数名にセル番地を使用することはできない。

[例] 数値型: row, col

例は, 数値型の変数 row, col の宣言である。

セルを変数として使用でき, これをセル変数という。セル変数は, 宣言せずに使用できる。

セル変数の表現方法には、絶対表現と相対表現とがある。

セル変数の絶対表現は、セル番地で表す。

セル変数の相対表現は、次の書式で表す。

| 書式 | 解 説 |
|----------------------|---|
| 相対(セル変数, 行の位置, 列の位置) | セル変数で指定したセルを基準のセルとする。そのセルの行番号と列番号の位置を 0 とし, 下又は右方向を正として数え, 行の位置と列の位置で指定した数と一致する場所にあるセルを表す変数である。 |

[例1] 相対(B5, 2, 3) は, セル E7 を表す変数である。

[例2] 相対(B5, -2, -1) は, セル A3 を表す変数である。

(3) 配列

数値型, 文字列型又は論理型の配列は宣言することで使用できる。添字を “[” 及び “] ” で囲み, 添字が複数ある場合はコンマで区切る。添字は 0 から始まる。

なお, 数値型及び文字列型の変数及び配列の要素には, 空値を格納することができる。

[例] 文字列型: table[100, 200]

例は, 100 × 200 個の文字列型の要素をもつ 2 次元配列 table の宣言である。

(4) 宣言, 注釈及び処理

宣言, 注釈及び処理の記述は, “共通に使用される擬似言語の記述形式” に従う。

処理の記述中に式又は関数を使用する場合, その記述中に変数, セル変数又は配列の要素が使用できる。

[例] 数値型: row

```
row: 0, row < 5, 1
  ・相対(B5, row, 0)    順位(相対(C5, row, 0), G5 ~ G9, 0)
```

例は, セル C5, C6, ..., C9 の各値に対して, セル範囲 G5 ~ G9 の中での順位を調べ, その順位をセル B5, B6, ..., B9 に順に代入する。

Ver 2.0 : 平成 23 年 7 月 11 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>