超小型記号言語 取扱説明書(第7版)

■はじめに

「超小型記号言語」は数値計算とパズル的なプログラミングを行うために作った独自仕様のプログラミング言語です。

言語処理系は非常にコンパクトで、数値は32ビット符号付整数のみ、エラーチェックは皆無に近く、文法的に誤っていてもインタプリタは実行を継続します。

文法はFORTHと類似していて、逆ポーランド記法、ディクショナリの考えを取り入れています。

記号言語というその名の通り、1文字の記号が命令として動作します。

例えば、記号(！)はスタックから２つのデータをPULLして、これを「書込データ」と「書込アドレス」と見なしてメモリにデータを書き込む命令です。次のプログラム

#123 #2！

は、123と２をスタックにPUSHした後、スタックから２つのデータをPULLして、123を2番地のメモリに書き込みます。

なお、超小型記号言語では文字A～Zを数値0～25と同等に扱うため、Cは２を意味し

#123 C！

と書いても同じ動作を行います。あたかも変数Cに123を代入しているように見えます。これが超小型記号言語における変数への代入の仕組みです。

同様に、記号(@)を使うと

C @

で変数C(実際は2番地)を読み出すことができます。

■動作環境

日本語版Windows10で動作します。

Windows10以外でも動作するはずですが、確認はしていません。

■インストールと削除

適当なZIP解凍ツールで解凍し、解凍したファイル全体を1つのフォルダに入れます。

フォルダの中のvts.exeを実行することで「超小型記号言語」が起動します。(*1)

コマンドプロンプト画面に?を表示したら、超小型記号言語で記述されたプログラム(*2)のファイル名(hello.vtsなど)を入力すると実行します。

削除する場合は、フォルダごと削除してください。

(*1)１回目の起動で「PC が危険にさらされる可能性があります」と警告が出たら「詳細」から「実行」をクリックしてください。不安な方は、ダウンロードしたファイルをWindows Defenderでスキャンして、ファイルの安全性をチックしてください。

(*2)あらかじめメモ帳などのエディタでプログラムを作って、ファイル保存しておく必要があります。

■使用条件、保証範囲

本ソフト、ドキュメントは無保証です。複製、配布(ソースを改造しての配布を含む)は自由に行ってください。

■その他

・この取扱説明書(第7版)は、超小型記号言語(Ver1.6)に対応しています。

・本文書で使用する製品名は、各社、各組織の登録商標、または商標です。

・この取扱説明書では、システムディクショナリ(sys.dic)で定義されたワードの解説はしていません。必要に応じて、テキストエディタで開いて解読してください。

本編 文法解説

■数値、数式、数値出力

数値は32ビット符号付き整数(-2147483648～2147483647)のみで、数値の前には記号(＃)を付けます。

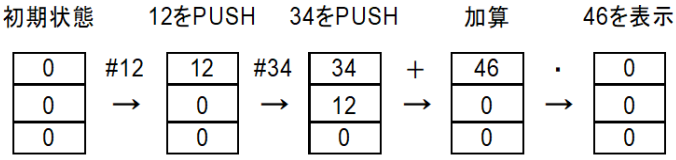
数式は逆ポーランド記法で、1文字以上のスペースで区切って

<数値1> <数値2> <演算子>

とします。たとえば

```
#12 #34 + .
```

は、12と34をスタックにPUSHした後、記号(＋)で12+34を計算し、記号(.)により計算結果46をコンソールに表示するプログラムです。



実行途中に数値をキーボードから入力する場合は、数値の代わりに記号(?)を使います。

```
? ? + .
```

この場合、記号(#)は不要です。

負数は扱えるのですが、定数として記述できません。代わりに

```
#0 #1234 -
```

のように0から正数の引算を行ってください。これで-1234がスタックにPUSHされた状態になります。

■データメモリ

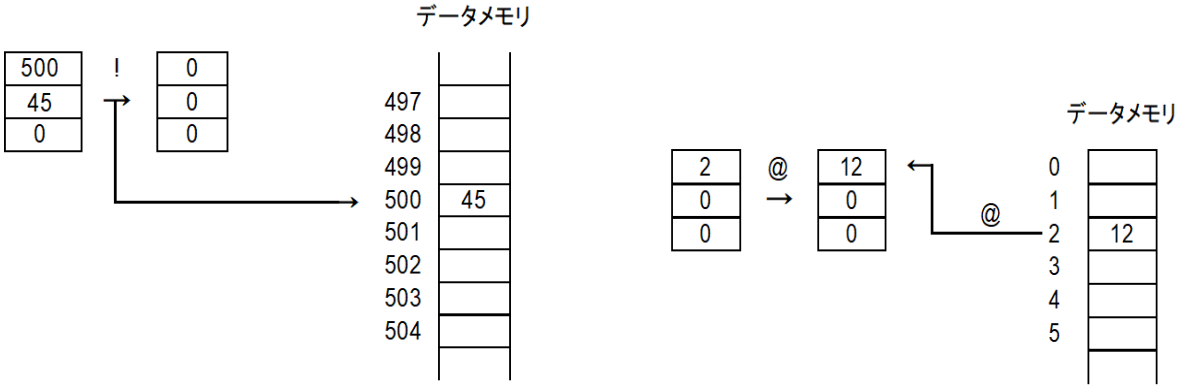
スタックとは別に、0番地～65535番地(1番地あたり32ビット)のデータメモリがあります。

<数値1> <数値2> !

で、<数値1>を<数値2>が示す番地のデータメモリに書込みます。たとえば

```
#45 #500 !
```

は、45を500番地のデータメモリに書き込みます。(下図左)



逆に、データメモリから数値を読出す場合は

<数値1> @

とします。これで<数値1>が示す番地のデータが、スタックにPUSHされます。たとえば

```
#2 @
```

は、2番地のデータがスタックにPUSHされます。(上図右)

■変数、配列

アルファベット大文字(A～Z)を数値(0～25)と等価に扱います。この仕組みを利用することで

<数値> <変数名> !

という書式で、変数に数値を代入することができます。たとえば、変数Cに数値1000を代入するには

```
#1000 C !
```
としますが、実際には2番地に数値1000を書き込んでいて、次の式と等価です。
```
#1000 #2 !
```

変数の値を読出する場合は

<変数名> @

とします。次のプログラムは、変数Aと変数Bの乗算結果をコンソールに表示します。
```
A@ B@ * .
```

変数Z(25番地)以後のデータメモリ(26～60999番地)は配列として使用可能です。たとえば、配列Z(10)に5を代入するには
```
#5 Z #10 + !
```
とします。

一見複雑に見えますが、変数Zの番地番号(数値25)に数値10を加算した数値(35)が示す35番地をアクセスしているのがわかります。

なお、データメモリ61000～65535番地はシステムディクショナリで使用するので、通常のプログラミングでは使用しないでください。

■演算子

次の演算子が使用できます。

演算の種類	超小型記号言語での記号
算術演算子の加減乗除と剰余	+ - * / %
比較演算子の<、>、=、≠	< > = =^
論理演算子のAND、OR、NOT	& | ^

NOT以外は2項演算子で、スタックからPULLした2個のデータを演算し、結果をスタックに1個PUSHします。

比較演算は、比較の結果が真なら全ビット1(-1)、偽なら全ビット0(0)の値を取ります。

NOTのみ単項演算子で、スタックの増減はありません。

■ワード(サブルーチン)

超小型記号言語ではサブルーチンを「ワード」と呼びます。

プログラム中で記号(：)に続いて3文字固定長のワード名を記述すると、そのワードが実行されます。

ワードの定義は、メインプログラムの後に中括弧を使って

{<ワード名> <そのワードの処理内容を表すプログラム> }

と記述します。このとき、記号({)の直後に空白を入れず、逆に記号(})の前に空白または改行が必要です。

ワード呼び出し前後で引数や戻り値が必要な場合は、演算スタックを使用します。たとえば
```
#12 #34 :ABC
```
は、スタックに数値12と13をPUSHした後、ワードABCを実行しています。

これ以外に、ワード呼び出しの引数や戻り値にデータメモリ(変数A～Zを含む)使う方法もあります。

■システムディクショナリ(sys.dic)

汎用的なワードはシステムディクショナリと呼ぶファイル(sys.dic)で定義してあります。

システムディクショナリはプログラム実行直前に読み込まれます。

プログラム中で記号(：)に続いてシステムディクショナリで定義されたワードを記述すると、そのワードが実行されます。

■ユーザディクショナリ(usr.dic)

システムディクショナリとは別に、プログラマが自由にワードを定義できるユーザディクショナリ(usr.dic)があります。

ユーザディクショナリもプログラム実行直前に読み込まれます。

■条件分岐、ラベル

条件分岐命令は次の書式です。

<数値1> <数値2> <比較演算子> <分岐先ラベル名>]

ラベル名はアルファベット小文字(a～z)で、比較結果が真(0以外)なら同じラベル名のラベルへジャンプし、偽(0)ならジャンプしません。

たとえば、N>100のときラベルyへジャンプさせるには

```
N@ #100 > y]
```

とします。なお、分岐先のラベルは次の書式です。

[<ラベル名>

先程のプログラムの分岐先であれば

```
[y
```

と記述します。

ラベルのスコープはメインプログラム内、または各ワード定義内でのみ有効です。言い換えると、メインプログラムと各ワード定義プログラム内で、それぞれが最大26個のラベル名を使えることになります。

無条件ジャンプさせるには、初めから比較結果を真(0以外)とし

#1 <分岐先ラベル名>]

のように記述します。

■文字出力、コメント

文字列出力は、文字列を記号(")で囲みます。改行する場合は後に記号(;)を付けます。

```
“HELLO WORLD”;
```

コメント文字列は、文字列を小括弧で囲みます。

```
(KOREHA COMMENT DESU)
```

文字コードによる1文字出力は、文字コードの後に記号(¥)を付けます。

特に文字コードが改頁(12)の場合、画面クリアしカーソルは左上に移動します。

■実行終了

メインプログラム末でプログラムは実行終了します。

メインプログラム途中で実行終了させる場合は、メインプログラムの末までジャンプさせてください。

プログラムにワード定義がある場合は、ワード定義プログラムの直前にジャンプさせることになります。

以上

来歴(超小型記号言語 取扱説明書)
2018/09/17 新規作成、第1版とする
2018/09/17 誤記訂正し、第2版とする
2018/09/23 Ver1.4に対応し、第3版とする
2019/01/02 Ver1.5に対応し、第4版とする
2019/01/12 誤記訂正し、第5版とする
2020/05/30 全面改訂、PDF版とし、第6版とする
2021/04/25 Ver1.6に対応し、第7版とする