2005/11/23

Osamu Tamura

ForCy 文法

16 bit スタックマシンへの後置記法プログラム。ワード、変数、定数、コメントで構成される。

大分類	小分類	要素	例
ワード	システム定義	文字列	dup +
	ユーザ定義	文字列	:main
変数	変数	符号なし 16 ビット整数	;x ;y ;count
	配列	変数の１次元配列	[10]a [0x20]b
定数	数値	符号なし 16 ビット整数	34 0x1f 'Q'
	文字列	バイト列（０終端）	"string" "¥x7f¥xff"

ワード登録	`... :name`	コロンに続けて名称
	`... self ... :name`	再帰呼び出し
ワード実行		最終登録ワードを実行
変数		
宣言	`;name`	セミコロンに続けて名称
参照	`name`	変数の”値”をスタックトップに積む
代入	`=name`	スタックトップの値を変数に代入
配列		
宣言	`[size]name`	size の配列領域を確保
参照	`index name`	name[index]の値をスタックトップに積む
代入	`index =name`	スタックトップの値を name[index]に代入
文字列	`"string"`	中間コード上の文字列開始位置を内部ポインタにセット。
		文字列は 0 終端
	`index @s`	内部ポインタから index 番目の文字をスタックトップに積む
コメント	/*　*/	区間をコメント扱い
	//	行末までをコメント扱い
	,	末尾 ‘,’ のワードをコメント扱い

・ワードはスペース、タブもしくは改行で区切る（半角文字のみ）。

・ワード、変数名、配列名は半角 15 文字まで。

・最終登録ワードから実行する。名称は任意。

・文字列は半角 255 文字まで。Shift-JIS コード可。

・コメント /* */, // の長さは任意。ただし ‘,’ コメントは半角 14 文字まで。

フロー制御		
	`{ ... } do`	無限ループ
	`{ ... 条件式 while ... } do`	真で継続（偽で脱出）
	`{ ... 条件式 break ... } do`	真で脱出（偽で継続）
	`{ ... 条件式 continue ... } do`	真で{ } 節の先頭から再開（偽で継続）
	`{ ... 指値更新 ... } 終値 指値 for`	指値が終値に等しくなるまで繰り返し
	`終値 指値 { ... 指値更新 ... } for`	（指値更新：スタックトップに ++,--など任意）
	`条件式 { ... } if`	真で{ } 節を実行
	`条件式 { ... } { ... } ifelse`	真で左{ } 節、偽で右{ } 節を実行
	`比較値 {`	
	`　値1 { ... } case`	
	`　値2 { ... } case`	比較値と一致する値の左節のみ実行
	`　 ...`	一致がなければ末尾を実行
	`} switch`	
	`return`	実行中のワードからリターン
	`n { w0 w1 ... wk } of`	n 番目のワードを実行
	`{ w0 w1 ... wk } interrupt`	割り込み実行ワードのテーブル設定） w0: 10mS 毎に呼び出されるワード w1: 1Sec 毎に呼び出されるワード

・of, interrupt では、ユーザ定義ワードのみ指定可。

for

・終了後に指値と終値が残るので破棄 .. する。

・continue 成立時、以降はスキップされる。

```
{
    ...             // ここで指値操作する
    条件 continue
    ++              // 成立時は実行されない！
} n 0 for ..
```

・指値（ループカウンタ）を変数で使うときは

```
;i
{
    dup =i
    ...
    ++
} 10 0 for ..
```

・指値が終値を飛び越すときは

```
{
    ...
    3 +
    10 min
} 10 0 for ..
```

などとするか while で記述する。

if, ifelse

・条件式は、{ } 節の後でもよい。

・{ } 節の直下で while, break, continue は使えない。

switch

・{ } 節の直下で while, break, continue は使えない。

・終了後に比較値が残るので破棄 . する。

return

・データスタック管理はプログラマが行うこと。

比較演算	==	(u1 u2 -- flag)
	!=	スタックから”１項”を除き、比較の結果を積む。
	<	
	<=	
	>	
	>=	
単項演算	++	(u1 -- u2)
	--	スタックから１項を除き、結果を積む。
	~	
	!	
二項演算	+	(u1 u2 -- u3)
	-	スタックから２項を除き、結果を積む。
	*	
	/	
	%	
	&	
	|	
	^	
	&&	
	||	
	<<	
	>>	
	min	
	max	

・２値の比較後は始めの値が残るので不要なら破棄する。

x y < nip { ... } if

x y < { ... } if .

（これを { } 内ですると釣り合わなくなる）

・データは符号なし整数なので、x 0 >= は常に成立（無意味）。

スタック操作	.	(X --)
		スタックから１項目を除く。Forth:drop
	..	(X1 X0 --)
		スタックから２項目を除く。Forth:2drop
	@	(Xu .. X0 u -- Xu .. X0 Xu)
		スタックトップから指定番目 u の値をスタックに積む。
	dup	(X -- X X)
		スタックトップをコピーする。（ 0 @ と等価）
	over	(X1 X0 -- X1 X0 X1)
		スタックの２項目をコピーする。（ 1 @ と等価）
	=	(Xu+1 .. X1 X0 u -- X0 .. X1)
		スタックトップの値を指定番目 u の項目にコピーする。
	nip	(X1 X0 -- X0)
		スタックの２項目を削除する。（ 0 = と等価）
	swap	(X1 X0 -- X0 X1)
		スタック上位の２項目を交換する。
	_ds	(Xu-1 .. X0 -- Xu-1 .. X0 u)
		データスタック上の項目数をスタックに積む。（デバッグ用）
	_rs	(-- u)
		リターンスタック上の項目数をスタックに積む。（デバッグ用）

コンソール 入出力	@c?	(-- flag) コンソール文字入力の有無をスタックに積む。
	@c	(-- char) コンソール入力文字をスタックに積む。
	@s	(u -- char) 最後出の文字列定数のスタックトップ番目の文字をスタックに積む。
	%c	(char --) スタックトップの値をコンソールに出力する。
	%s	(--) 最後出の文字列定数をコンソールに出力する。
	%d	(u --) スタックトップの値を十進数でコンソールに出力する。
	%x	(u --) スタックトップの値を十六進数でコンソールに出力する。
システム	sfr	(u -- u) 特殊機能レジスタ [u] の値をスタックに積む。
	=sfr	(u1 u2 --) 特殊機能レジスタ [u2] に値 u1 をセットする。
	ep	(u -- u) プログラムメモリ [u] の値をスタックに積む。
	=ep	(u1 u2 --) プログラムメモリ [u2] に値 u1 をセットする。
	ed	(u -- u) データメモリ [u] の値をスタックに積む。
	=ed	(u1 u2 --) データメモリ [u2] に値 u1 をセットする。
	clk	(-- u) システム経過時間 (10mS 単位) をスタックに積む。
	sec	(-- u) システム経過時間 (秒単位) をスタックに積む。