1. Publicação nº	2. Versão	3. Data	5. Distribuição
INPE-2894-PRE/413		*Set., 1983*	☐ Interna ☒ Externa ☐ Restrita

4. Origem: *DPD/DIN* — Programa: *INTAL*

6. Palavras chaves - selecionadas pelo(s) autor(es)

FORTH
LINGUAGEM DE PROGRAMAÇÃO
LINGUAGEM EXTENSÍVEL

7. C.D.U.: *681.322.0*

8. Título *INPE-2894-PRE/413*

FORTH - UMA LINGUAGEM DE PROGRAMAÇÃO NÃO-CONVENCIONAL

10. Páginas: *09*

11. Última página: *07*

12. Revisada por

N.L. Vijaykumar
Nandamudi L. Vijaykumar

9. Autoria *Edson L. França Senne*
Maria Teresa B. Rios

13. Autorizada por

Nelson de Jesus Parada
Diretor Geral

Assinatura responsável

14. Resumo/Notas

FORTH é uma linguagem interativa que, devido ao seu conjunto restrito de funções básicas, pode ser aprendida com facilidade e implementada em computadores pequenos. A característica mais importante que a difere das linguagens convencionais é a sua extensibilidade: programar em FORTH consiste basicamente em definir novas funções a partir das já existentes. Estas novas funções são incorporadas à linguagem e nada as distingue das funções originais. Isto permite ao usuário definir suas próprias operações e criar seus próprios tipos de dados. Assim, o FORTH torna-se uma ferramenta bastante útil para definir linguagens orientadas para aplicações específicas. Este trabalho apresenta a linguagem e um simulador FORTH portátil, escrito em PASCAL, com um mínimo de funções primitivas que permite ao usuário familiarizar-se com o sistema FORTH.

15. Observações

Trabalho submetido para apresentação no 3º Simpósio sobre Desenvolvimento de Software para Micros a ser realizado no Rio de Janeiro, 05 a 07 dezembro/1983.

ABSTRACT

Forth is an interative language that, due to its restrict set of basic functions, can be easily learned and implemented in small computers. The most important characteristic which distinguishes it from the conventional languages, is its extensibility: programming in Forth consists basically in defining new functions from others already existent. These new functions are incorporated to the language and nothing distinguishes it from the original functions. This allows the user to define his own operations and create his own data types. Thus, Forth becomes a very useful tool to define oriented languages for specific applications. This work presents the language and a portable Forth simulator, written in PASCAL, with a minimal number of primitive functions that permit the user to get acquainted with the Forth System.

FORTH: UMA LINGUAGEM DE PROGRAMAÇÃO NÃO-CONVENCIONAL

Edson L.F. Senne
Maria R.B. Rios

Instituto de Pesquisas Espaciais - INPE
Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq
Caixa Postal 515 - 12200 - São José dos Campos - SP

SUMÁRIO

FORTH é uma linguagem interativa que, devido ao seu conjunto restrito de funções básicas, pode ser aprendida com facilidade e implementada em computadores pequenos. A característica mais importante que a difere das linguagens convencionais é a sua extensibilidade: programar em FORTH consiste basicamente em definir novas funções a partir das já existentes. Estas novas funções são incorporadas à linguagem e nada as distingue das funções originais. Isto permite ao usuário definir suas próprias operações e criar seus próprios tipos de dados. Assim, o FORTH torna-se uma ferramenta bastante útil para definir linguagens orientadas para aplicações específicas. Este trabalho apresenta a linguagem e um simulador FORTH portátil, escrito em PASCAL, com um mínimo de funções primitivas que permitam ao usuário familiarizar-se com o sistema FORTH.

1. INTRODUÇÃO

FORTH é uma linguagem de programação que foge dos padrões convencionais: possui recursos de um sistema Operacional, de um editor de textos, de um *"assembler"* e de um sistema monitor. Desta forma, quando se trabalha com FORTH, todos estes utilitários estão na memória principal, disponíveis ao usuário para uso imediato. É uma linguagem interativa que, devido ao seu conjunto restrito de funções básicas, pode ser aprendida com facilidade e implementada em computadores pequenos.

A característica mais importante da linguagem FORTH que a distingue das linguagens convencionais é a sua *extensibilidade*: programar em FORTH consiste basicamente em definir novas funções a partir das já existentes (Timin, 1981). Estas novas funções são incorporadas à linguagem e, no seu modo de utilização, nada as difere das funções originais. Esta característica tem conseqüências muito importantes: é muito fácil escrever programas modulares, definindo funções num esquema de complexidade crescente até obter uma função que, utilizando funções de complexidade menor, executa a tarefa desejada. Além disso, o usuário, ao definir suas próprias operações e criar seus próprios tipos de dados, pode adaptar a linguagem às suas necessidades particulares. Assim, o FORTH torna-se uma ferramenta bastante útil para definir linguagens orientadas para aplicações específicas.

2. A LINGUAGEM FORTH

Uma função (ou "palavra") FORTH é semelhante a uma sub-rotina ou subprograma em outras linguagens. No entanto, as regras de uso e a

implementação interna são exclusivas do FORTH. A linguagem é construída com um dicionário de funções primitivas que serão utilizadas na definição de novas funções mais complexas. Estas funções são acrescentadas ao dicionário juntamente com os endereços de posições da memória onde se iniciam suas definições. Isto permite que qualquer sequência de comandos seja executada sem chamadas a sub-rotinas.

Como as funções FORTH são formadas por chamadas a outras funções que são identificadas através dos endereços de suas definições, o código executável de um programa FORTH é praticamente uma coleção de endereços, o que o torna bastante reduzido.

Em geral, as funções FORTH *retiram* seus argumentos do topo de uma pilha de dados e *deixam* os resultados por elas produzidos também nesta pilha, sendo este o modo mais eficiente de troca de dados entre as funções. Como consequência, utiliza-se na escrita de programas em FORTH, a notação polonesa reversa, que é um modo natural de se comunicar com a máquina de pilha.

É importante notar que, usando a notação polonesa reversa, as funções FORTH podem ter um número qualquer de argumentos e resultados.

Outras linguagens utilizam pilha para troca de dados entre funções, mas sua operação é geralmente incorporada à própria linguagem e transparente ao usuário. Em FORTH, não só a pilha de dados, mas praticamente toda a máquina virtual pode ser controlada diretamente pelo usuário (James, 1980).

FORTH não impõe restrição quanto à definição de nomes por parte do usuário, sendo importante, portanto, que as palavras sejam separadas por espaços que constituem dessa forma os únicos delimitadores.

A definição de uma função em FORTH é feita utilizando a palavra ":", seguida do nome da função que se quer definir e dos comandos que a constituem, terminando a definição com a palavra ";". No exemplo abaixo ilustra-se a definição da função CUBO:

```
: CUBO DUP DUP * * ;
```

Uma vez definida, a nova palavra torna-se parte integrante da linguagem. O usuário pode entrar agora com

```
3 CUBO .
```

e irá receber como resposta (nota-se que a palavra "." tem o efeito de mostrar no terminal o valor no topo da pilha):

```
27
```

3. O PROGRAMA QFORTH

A linguagem FORTH consegue as características de eficiência e compactação através da compilação de seus programas em cadeias de endereços. Estes endereços referenciam posições de memória que contêm definições

para cada palavra do dicionário. Usando esse esquema de endereçamento indireto, conhecido como código alinhavado ("threaded code"), os programas FORTH são executados sem a sequência usual de chamadas a sub-rotinas (Posa, 1979).

O algoritmo de execução de código alinhavado é bastante simples e em linguagem simbólica *"assembly"* pode ser implementado facilmente. Em linguagem de alto-nível no entanto, sua implementação fica dificultada (Bell, 1973).

Por esta razão, o programa implementado QFORTH - *interpreta* os comandos FORTH de maneira a *simular* a execução de código alinhavado.

QFORTH utiliza as seguintes estruturas de dados principais:

a) STACK - pilha de execução onde os valores são colocados para ser operados. A variável TOS indica o topo dessa pilha.

b) DICTIONARY - uma lista onde são colocadas as funções definidas pelo usuário. Suas entradas possuem o formato:

   1- IMMED : um bit indicando se a função deve ser executada no modo interpretador ou no modo compilador;

   2- NAME : contém o nome da função;

   3- ADDRESS : contém o endereço de definição da função na memória (no interpretador denominada MEMORY).

   4- PARFIELD : contém o endereço de espaço alocado e dos parâmetros da função.

c) SOURCEFILE - arquivo utilizado para armazenamento de programas.

O núcleo de funções primitivas inclui:

a) JUMP - desvio incondicional;

b) JMPZ,JMPN - desvio condicional;

c) COMPILE - coloca a função que a segue na primeira posição disponível do dicionário;

d) HERE - coloca em STACK o endereço da primeira posição disponível no dicionário;

e) C, - coloca o valor apontado por TOS no dicionário;

f) IMMEDIATE - torna IMMED da última entrada do dicionário igual a 1;

g) : - coloca FORTH no modo compilador;

h) ; - coloca FORTH no modo interpretador;

i) <BUILDS, DOES> - permite criação de famílias de funções;

j) . - imprime valor apontado por TOS;

k) + , - , * , / - operações aritméticas sobre valores em STACK;

l) @ - coloca em STACK o valor contido em MEMORY (TOS);

m) ! - armazena valor em MEMORY (TOS);

n) PAGE - cria uma página para salvar novas funções;

o) SAVE - salva uma página em SOURCEFILE;

p) COMPILER - compila uma página;

q) LOAD - carrega uma página de SOURCEFILE;

r) EDITOR - edição de uma página;

s) FORGET - retira entradas do dicionário;

t) DLIST - lista nomes de funções presentes no dicionário.

4. O MODO COMPILADOR E O MODO INTERPRETADOR

A linguagem FORTH tem dois modos de operação:

a) MODO 0 - executa função

b) MODO 1 - compila função

No modo 0 (modo interpretador) as funções, uma vez reconhecidas, caso possuam IMMED = 0, são prontamente executadas. Nesse modo de operação, funções com IMMED = 1 tornam-se inoperantes. Como as funções primitivas possuem todas IMMED = 0, este é o modo normal de operação da linguagem.

No modo 1 (modo compilador) os endereços das funções vão sendo montados no dicionário, constituindo assim a definição de uma nova função. Como é possível ter funções definidas com IMMED = 1, podem-se executar funções no tempo de compilação de outras funções. Esta possibilidade permite que se implementem funções de controle de fluxo em FORTH, tais como IF ... ELSE ... THEN, BEGIN ... UNTIL, DO ... LOOP e outras. Para ilustrar seja a construção:

```
BEGIN < funções 1 > WHILE < funções 2 > REPEAT
```

cuja semântica é dada por:

```
execute < funções 1 > ;
while TOS = 1 do begin
     execute < funções 2 > ;
     execute < funções 1 > ;
end;
```

```
: BEGIN HERE ; IMMEDIATE
: WHILE COMPILE JMPZ HERE 0 C, ! ; IMMEDIATE
: REPEAT COMPILE JUMP HERE 0 C, ROT SWAP ! HERE SWAP ! ; IMMEDIATE
```

Outra característica importante da linguagem FORTH é a *recursividade*. Podem-se definir funções recursivas muito facilmente, como a função FATORIAL:

```
: FATORIAL DUP 0 = IF DROP 1 ELSE DUP 1 - FATORIAL * THEN ;
```

Um outro exemplo, a função de Ackermann pode ser escrita como:

```
: ACK SWAP SUP 0 =
     IF DROP 1 + ELSE SWAP DUP 0 =
          IF DROP 1 - 1 ACK ELSE OVER 1 -
            SWAP 1 - ROT SWAP ACK ACK
          THEN
     THEN ;
```

Nesse caso, por exemplo:

3 2 ACK . irá resultar (prontamente!) em 29.

5. PALAVRAS DE DEFINIÇÃO

Como já foi dito, o usuário pode definir novas funções FORTH, que são incorporadas à linguagem e podem ser utilizadas da mesma forma que as funções primitivas. A palavra ":" que é utilizada para definir outras palavras é chamada *palavra de definição*. Neste interpretador FORTH esta é a única palavra de definição primitiva. Os interpretadores mais conhecidos possuem, além desta, as palavras "VARIABLE" (que define variáveis), "CONSTANT" (que define constantes), "CODE ... NEXT" (que define funções escritas em ASSEMBLY).

Entretanto, novas palavras de definição podem ser criadas pelo usuário. Estas novas palavras dão origem a "famílias" de funções que podem possuir qualquer número de elementos. A palavra de definição "VARIABLE" define uma família de palavras, onde cada membro possui nomes e valores diferentes, mas a execução dessas palavras é idêntica: ao digitar o nome de um dos membros dessa família, o seu endereço é colocado no topo da pilha.

É importante notar que a criação e a utilização de uma palavra de definição implicam uma sequência de eventos que estão esquematizados na Figura 1.

O primeiro evento cria uma função que definirá uma nova família de palavras. Neste momento, a família ainda não possui nenhum membro. O segundo evento utiliza esta nova palavra de definição para criar um novo membro de família. O terceiro evento ocorre toda vez que um membro da família é executado (Harris, 1980).

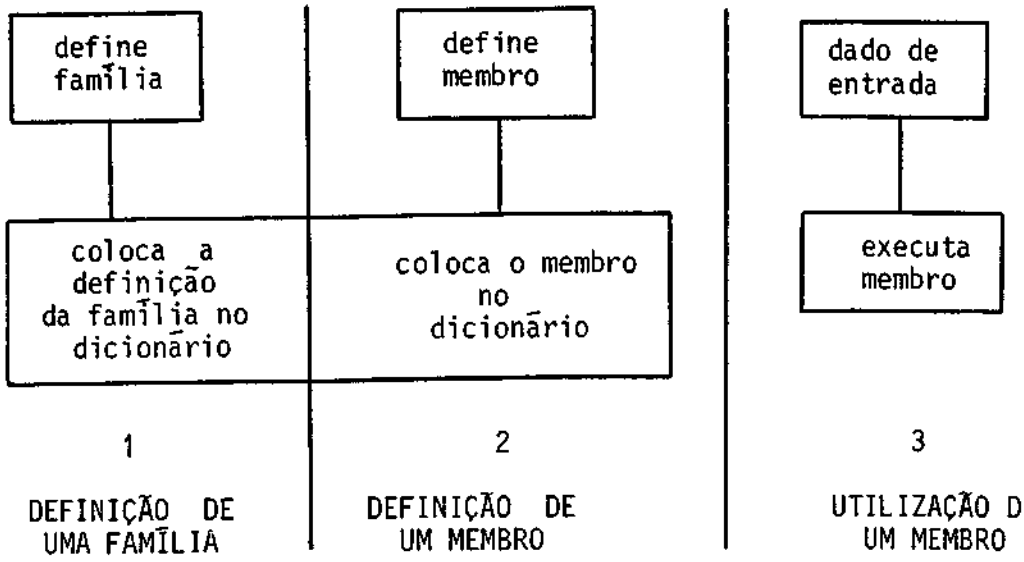

Fig. 1 - Sequência de eventos

O exemplo abaixo ilustra a definição de uma família de funções:

```
: VAR <BUILDS 1 ALLOT DOES> ;
```

A palavra de definição ":" indica que uma nova função vai ser definida, colocando o FORTH no modo compilador; isto faz com que o nome que a segue seja colocado no campo NAME do dicionário. O endereço da memória onde se iniciará a definição de VAR é colocado no campo ADDRESS. Como esta função não reserva espaço de memória, nem recebe parâmetros, o campo restante é preenchido com zero. A seguir, o programa traduz os comandos entre o nome e o ";" em uma cadeia de código na memória.

Ao ser encontrado o ";", o FORTH retorna ao modo intérprete. Para definir um novo membro da família VAR deve-se teclar a palavra de definição seguida do nome do novo membro, por exemplo VAR A.

Uma vez que o programa está no modo intérprete, a função VAR é executada.

Ao executar a primitiva <BUILDS, o interprete realiza duas ações:

a) coloca o nome que segue a VAR no campo NAME do dicionário;

b) coloca em PARFIELD o valor da primeira posição disponível no dicionário.

A primitiva DOES>, ao ser executada no contexto <BUILDS ... DOES>, atualiza o campo ADDRESS com o endereço de definição do membro A da família VAR.

Ao utilizar a nova palavra A, função DOES> será executada fora do contexto <BUILDS ... DOES> e desta forma colocará em STACK o endereço contido em PARFIELD.

Outras estruturas podem ser facilmente construídas com palavras de definição, como por exemplo MATRIZ, STRING e funções de manipulação de arquivos (Rios e Senne, 1983),

6. CONCLUSÃO

Neste trabalho discutiu-se a linguagem FORTH, uma linguagem que, devido às suas características de velocidade, compactação, expansibilidade, portabilidade e estruturação, é ideal para implantação em microcomputadores.

Foram discutidas a filosofia da linguagem, as suas características mais importantes e também algumas de suas funções primitivas.

As idéias aqui discutidas foram incorporadas a um programa de computador - o programa QFORTH - que simula um ambiente de programação FORTH. Este programa está escrito em PASCAL padrão, sendo desta forma transportável e tem cerca de 600 linhas fonte.

REFERÊNCIAS BIBLIOGRÁFICAS

BELL, J.R. Threaded Code, *Communications of the ACM*, *16*(6):370-372, June, 1973.

HARRIS, K. FORTH extensibility: or how to write a compiler in 25 words or less, *BYTE*, *5*(8):164-184, Aug., 1980.

JAMES, J.S. What is FORTH? A Tutorial Introduction, *BYTE*, *5*(8):100-126, Aug., 1980.

POSA, J.G. Programming microcomputer systems with high-level languages, *Electronics*, *(18)*:105-112, Jan., 1979.

RIOS, M.T.B.; SENNE, E.L.F. Um interpretador FORTH portátil com manipulação de arquivos, escrito em PASCAL, *6º Congresso Nacional de Matemática Aplicada e Computacional*, ITA, São José dos Campos, SP, Set., 1983.

TIMIN, M.E. The FORTH alternative, *Dr. Dobb's Journal*, *6*(59):57-59, Sept., 1981.