# FORTH

VIDEOTON

# TV-Computer

209.530.01.01 00/A

# FORTH

## Felhasználói kézikönyv

Felelős kiadó: Dr. Baráth Csaba
Készült a Forma-Art Kiadó gondozásában
KŐNYOMAT Sokszorosító és Szolgáltató Kisszövetkezet, 46/87.
Felelős vezető: ifj. Kasza Ferenc

## TARTALOMJEGYZÉK

# 1. BEVEZETÉS

## 1.1. A FORTH helye a világban

A FORTH programozási nyelv jelentősen különbözik az eddig ismert programozási nyelvektől, még lelkesebb hívei és védelmezői sem értenek egyet abban, hogy miért jó nyelv és mire lehet igazán jól használni. Tulajdonképpen nem is „csak" egy programozási nyelv, hanem egy integrált monitor; egyszerre interpreter, fordítóprogram és szövegszerkesztő, amelyekhez a virtuális memóriakezelés is biztosítva van. A FORTH sokkal inkább közel engedi a felhasználót a géphez, megbízható hibakezelése és nyomkövetési lehetőségei ellenére sem köti meg annyira a programozó kezét, mint pl. a PASCAL teszi. A FORTH lényegesen interaktívabb és sokkal kevesebb memóriát igényel, mint a hagyományos nyelvek túlnyomó többsége. (A teljes FORTH rendszer mindössze 14K terjedelmű és futási időben mindennel együtt sem foglal el a memóriából 18K-nál többet, ugyanez pl. egy PASCAL rendszernél 48K vagy több.)

## 1.2. A szó fogalma a FORTH-ban

A FORTH nyelv alapegysége a szó. A szó más nyelvek szubrutin fogalmához hasonlítható, az egész nyelv ezekből épül fel. A programozó egyre újabb szavakat definiál, eközben az egyik szó definíciója, vagy többé is, egy-egy önálló programmá válik. Célszerű minden új definiciót röviden megírni, 3–5 sor a kommenteken kívül a javasolt maximális hossz.

## 1.3. Felhasználói üzemmódok

A felhasználó a FORTH nyelvvel alapvetően két üzemmódban kommunikálhat. Egyrészt utasításokat adhat, amelyeket a FORTH azonnal végrehajt, másrészt programot írhat, amely programot a FORTH lefordít és meghíváskor végrehajt. A két üzemmód azonban szorosan összefügg, mert mint már említettük, a program is tulajdonképpen definíciósorozat, és a közvetlen utasítások itt valójában – a FORTH-ban magában vagy a programozó által – előredefiniált szavak sorozata. Mivel a FORTH-ban definiált szavak túlnyomó

többsége pontosan ugyanúgy működik mindkét esetben (tehát ha terminálról parancsként adják be, vagy ha egy másik definíció tagjaként kerül rá a vezérlés), ezért a nagyobb programok részét alkotó kisebb definíciók könnyen, gyorsan, közvetlenül tesztelhetők a képernyőnél.

### 1.4. A FORTH bővíthetősége

A FORTH nyelvben előredefiniált szavak listája e kézikönyv 10. fejezetében található azok magyarázatával együtt, azonban – mint az már az eddigiekből is kiderült – a felhasználó tetszés szerint bővítheti FORTH szavainak gyűjteményét. Az új definíciók a nyelv részévé válnak, a különböző alkalmazásoknak megfelelően speciális FORTH könyvtárak alakíthatók ki. Az új könyvtárakban a régi FORTH szavak új definícióval is szerepelhetnek, tehát a nyelv megengedi, hogy a felhasználó a rendszer által támogatott, előre definiált szavakat saját ízlésének és az új program környezetének megfelelően megváltoztassa. A legújabban beírt definíciókat ki lehet törölni. (Természetesen az előre definiált FORTH szavak védett szótárban vannak, azokat nem lehet „elfelejteni".) Mindezek mellett a felhasználónak lehetősége van új adattípusok, tömbtípusok vagy más adatszerkezetek bevezetésére, valamint új vezérlési szerkezetek kialakítására. A kész programok a változó követelményeknek megfelelően könnyen módosíthatók, hiszen az egész program kicsi, már belőtt programrészekből épül fel.
A FORTH strukturált nyelv, s mint ilyen, nincs benne címkefogalom vagy GOTO utasításhoz hasonló lehetőség.

### 1.5. A FORTH program kódja

A FORTH-ban írt program tárgykódja (a lefordított program) rendkívül tömör, sokszor az assembler nyelven írt kódnál is tömörebb. Ennek oka, hogy egy FORTH művelet meghívása – függetlenül attól, hogy az mit csinál és hogyan – mindig 2 byte-ot foglal el a tárgykódból. A FORTH programok tömörsége különösen nagy programok esetében érzékelhető jól.
A program a hívott műveletek címeit tartalmazza. Egy belső interpreter végigmegy a címlistákon és végrehajtja a megfelelő műveleteket. A címek – indirekciók sorozatán át – végül is egy assembly-ben megírt programmagra mutatnak. Ez a megoldás az ún. láncolt kód (threaded code).

# 2. ALAPISMERETEK

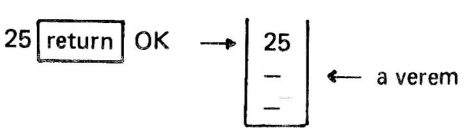
## 2.1. A verem fogalma

A FORTH nyelv használatának megtanulását a verem fogalmának megismerésével, ill. a feleleventésével kell kezdeni. A verem egy olyan számsorozat, amelyhez egymás után lehet mennyiségeket hozzáadni, majd elvenni oly módon, hogy mindig az utoljára betett mennyiséghez lehet először hozzáférni. Más szavakkal a verem egy olyan mechanizmus, amely mindig a tetején fogad és mindig onnan ad ki mennyiségeket. A FORTH két vermet használ, a paraméter vermet és a visszatérési (return) vermet. A paraméter verem a FORTH-ban definiált műveletek (szavak) közötti információcsere eszköze, a visszatérési vermet maga a FORTH rendszer visszatérési címek tárolására és egyéb rendszerfunkciókra használja. A felhasználó a FORTH nyelv használata során elsősorban a paraméter vermet látja, ezért a továbbiakban verem alatt a paraméter vermet értjük, és csak a visszatérési veremnél használjuk a megjelölést, hogy melyik veremről van szó. A verem tehát a műveletek közötti kapcsolatot biztosítja a FORTH nyelvben. Köznapi hasonlattal élve, a verem az önkiszolgáló étteremben levő tálcákhoz hasonlítható: a személyzet folyamatosan tölti fel a tálcahegyet, a vendégek pedig mindig a tetejéről vesznek el. Ha valaki nem talál tálcát (üres a verem), beír a panaszkönyvbe, azaz hibát jelez.

## 2.2. Veremműveletek

### *2.2.1. A számok*

A legegyszerűbb veremművelet a FORTH-ban egy szám leírása. A szám első közelítésben egy 16 biten ábrázolható egész szám lehet. A számjegyek között nem szabad szóközt hagyni, mert mint majd a későbbiekben látni fogjuk, a FORTH szavak közötti elválasztó jel mindig a szóköz. Egy szám begépelése egy közvetlen utasítás, melynek hatására a szám a verem tetejére kerül:



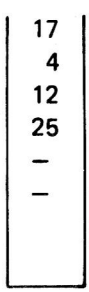
ahol 25 a szám, az aláhúzással megjelölt részeket a FORTH írja vissza, jelen esetben az OK-t, ami azt jelenti, hogy a FORTH végrehajtotta a kijelölt művelete(ke)t és inputra vár. (A 25-ös alatti kötőjelek azt jelzik, hogy régebben tetszőleges érték lehetett még a veremben, azok természetesen ott maradtak, csak lejjebb kerültek.)
A FORTH-ban több közvetlen utasítást is lehet egyszerre megadni, a FORTH a return hatására kezdi feldolgozni azokat.

A továbbiakban az utasításokat lezáró return-t nem írjuk ki.

A különböző FORTH műveleteket szóközzel kell elválasztani, így több szám begépelése:

```
25  12  4  17  OK
```

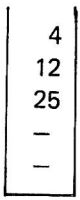
Ezután a verem tartalma:




A képernyőre a veremben levő számot a . (dot) művelettel lehet kiíratni. A . FORTH szó tehát éppen fordítottja az egy szám begépelése műveletnek, hatására a FORTH kiírja a verem tetején levő számot, és utána egy szóközt tesz. Az előző művelet után tehát:

```
. 17 OK
```

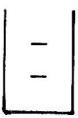
és a verem tartalma:




majd ezt többször ismételve:

```
...4  12  25  OK
```

és a verem tartalma:




*2.2.2. Egyszerű aritmetikai műveletek*

Az egyszerű aritmetikai műveletek (összeadás, kivonás, szorzás, osztás) a FORTH-ban előredefiniált szavak: + – * / /MOD MOD, a verem tetején várják operandusaikat és a művelet elvégzése után a verem tetején adják vissza az eredményt.
Ezek a műveletek az operandusokat törlik a veremből, túlcsordulást nem jeleznek. Pl:

```
16 4 * . 64 OK
```

és a verem üres. A * művelet tehát két értéket vár a veremben és egyet ad ott vissza. Ennek szokásos leírási módja:

```
* (n1 n2 – n3)
```

A zárójelek között írjuk le a verem tartalmát, a – jel előtt a művelet végrehajtása előtti (bemenő praméterek), a – jel után a művelet végrehajtása utáni (kimenő paraméterek) állapotot. A – jel mindkét oldalán jobbra van a

verem teteje. Ha egy FORTH művelet nem talál anyi értéket a veremben, amennyi bemenő paraméterre szüksége van, a FORTH hibát jelez (ld.: panaszkönyv).
A 4 alapművelet mellett felsorolt /MOD szó a /-hez hasonlóan osztás, csak visszaadja a veremben a hányados mellett a maradékot is, a MOD szó pedig csak a maradékot adja vissza:

/MOD (n1 n2 – maradék hányados)
MOD (n1 n2 – maradék).

Az aritmetikai műveletek leírását és a verem tartalmának alakulását mutatja a következő példa.

Számítsuk ki FORTH-ban

(100–(16+5)*(22–18))/4 értékét!

```
100 16 5 + 22 18 – * – 4 / . 4 OK
```

| Kezdetben | 100 | 16 | 5 | + | 22 | 18 | – | * | – | 4 | / | A végén |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| – | 100 | 16 | 5 | 21 | 22 | 18 | 4 | 84 | 16 | 4 | 4 | – |
| – | – | 100 | 16 | 100 | 21 | 22 | 21 | 100 | – | 16 | – | – |
| | | – | 100 | 1– | 100 | 21 | 100 | – | – | – | – | |
| | | | – | – | – | 100 | – | – | – | | – | |
| | | | – | | – | – | – | | | | | |
| | | | | | | – | | | | | | |

**Az aritmetikai kifejezések ilyen leírásmódját fordított lengyel jelölésnek is szokták nevezni.**

### *2.2.3. Veremkezelő műveletek*

A FORTH nyelv definiál néhány olyan műveletet, amely a veremben levő mennyiségek sorrendjét változtatja, ill. többszörözi ezeket az értékeket. Természetesen az alap veremműveletek segítségével a felhasználó tetszőleges továbbiakat definiálhat.
Az alap veremműveletek közül a SWAP felcseréli a verem tetején levő két mennyiséget:

SWAP (n1 n2 – n2 n1);

a DUP megduplázza a verem legfelső elemét:

DUP (n – n n);

a ROT három paraméteres művelet, a felülről harmadik elemet teszi a verem tetejére:

ROT (n1 n2 n3 – n2 n3 n1);

az OVER a verem második elemét teszi a verem tetejére oly módon, hogy ezt az elemet meghagyja harmadiknak is:

OVER (n1 n2 – n1 n2 n1);

a DROP pedig egyszerűen elveszi a verem tetején levő mennyiséget:

DROP (n –).

(Ne felejtsük: a veremleírásban a – jel mindkét oldalán jobbra van a verem teteje!)
A felsorolt veremműveletek (a ROT és OVER kivételével) 2–2 elemre is vonatkozhatnak, ekkor a megfelelő FORTH szó egy kettessel kezdődik:

2SWAP (n1 n2 n3 n4 – n3 n4 n1 n2)
2DUP (n1 n2 – n1 n2 n1 n2)
2DROP (n1 n2 –)

Figyeljük meg, hogy miben különbözik a DUP DUP és a 2DUP!

Ha pl. a verem tartalma (25 31 –):

a DUP DUP (25 31 – 25 31 31 31) lesz,
a 2DUP (25 31 – 25 31 25 31).

Példaként írjunk olyan FORTH szósorozatot az alap veremműveletek segítségével, amely helyettesíti a

4ROT (n1 n2 n3 n4 – n2 n3 n4 n1)

FORTH szót!

Javasoljuk az olvasónak, hogy lapozás előtt próbálja megoldani a feladatot! Egy megoldás: ROT ROT 2SWAP SWAP. Legyen pl.: n1=16, n2=49, n3=75, n4=32, akkor a verem tartalmának alakulása:

| Kezdet-ben | ROT | ROT | 2SWAP | SWAP |
|---|---|---|---|---|
| 32 | 49 | 75 | 32 | 16 |
| 75 | 32 | 49 | 16 | 32 |
| 49 | 75 | 32 | 75 | 75 |
| 16 | 16 | 16 | 49 | 49 |
| – | – | – | – | – |
| – | – | – | – | – |

További hasznos veremműveletek a MIN és MAX, amelyek a két felső elemet összehasonlítják, és a kisebbet, ill. nagyobbat hagyják a verem tetején:

MIN (n1 n2 – a kisebb),

ill.

MAX (n1 n2 – a nagyobb).

## 2.3. Konstansok, változók és számrendszerek

Mint láttuk, a FORTH-ban a verem a műveletek közötti kommunikáció fő eszköze, és mint ilyen, kitűnően alkalmas különböző mennyiségek átmeneti tárolására. Egy nagyobb programban azonban általában szükség van mennyiségek tartósabb megőrzésére. A FORTH erre a célra konstansok és változók definiálását biztosítja. A CONSTANT FORTH szó segítségével nevet lehet adni egy értéknek, pl.:

```
12 CONSTANT tucat OK
```

Látható, hogy a CONSTANT szónak egy bemenő paramétere van, azaz egy értéket vár a veremben, leírása után pedig elvár egy szót, amely a konstans neve lesz. A konstans nevére lehet a későbbiekben hivatkozni, és ez a hivatkozás a veremben hagyja a definiáláskor megadott értéket. Így a fenti definíció után:

```
tucat 5 + . 17 OK
```

Ezt a lehetőséget megállapodásszerűen akkor használjuk, ha az érték a program során nem változik.

A VARIABLE FORTH szóval egy 16 bites memóriacímet lehet megnevezni, erre a címre különböző értékeket lehet eltenni, ill. azokat elő lehet a címről venni. Pl. a

```
0 VARIABLE darab OK
```

változódefiníció hozzárendel a darab szóhoz egy memóriacímet és a veremben levő értéket (jelen esetben 0-t) leteszi erre a címre. A későbbiekben a darab-ra való hivatkozás a címet teszi a verem tetejére. Erről a címről a @ fetch) szóval lehet az ott levő 16 bites értéket elővenni, a ! (store) szóval pedig egy új 16 bites értéket lehet a címre letenni.
A ? (question) szó a @-hoz hasonlóan előveszi a változó értékét és kiírja azt a képernyőre (az érték után hagy egy szóközt). Pl.:

```
5 darab ! darab @. 5 OK
        darab ? 5 OK
```

Tehát a ? ekvivalens a @ . szósorozattal.

a változó értékét (azaz a címen levő tartalmat) a +! (plus-store) művelettel lehet növelni, ill. csökkenteni:

```
–2 darab +! darab ? 3 OK
```

Ha biztosak vagyunk benne, hogy a változóhoz használt értékek 1 byte-on is elférnek, azaz 0 és 255 közé esnek, használhatjuk a C! és C@ (c-store és c-fetch) szavakat, amelyek egy 8 bites értéket tesznek le (vesznek elő) a paraméterként kapott címre (címről).

Másfelől, ha várható, hogy az értékek 1 szón sem férnek el, a DVARIABLE szó segítségével duplaszavas változókat definiálhatunk. Ezeknek a tartalmát a 2! (two-store), 2@ (two-fetch) szavakkal lehet kezelni.

A változó területet az ALLOT szóval tudjuk tovább bővíteni. Ez egy pozitív egész számot vár el a verem tetején és ennyi byte-ot foglal le az utoljára definiált változó számára. Pl.: a

```
0 VARIABLE méret 10 ALLOT OK
```

szósorozatból a 0 VARIABLE méret szavak, mint láttuk, lefoglalnak 16 bitet (2 byte-ot), a 10 ALLOT szósorozat pedig további 10 byte-ot foglal le a méret szó címétől kezdve. Gyakorlatilag ez egy 12 elemű tömb, amelyiknek mondjuk a 7. elemébe a következő módon kell 82-t beletenni, ill. onnan elővenni

```
82 méret 6 + C! OK

méret 6 + C@. 82 OK
```

A FORTH-ban van még néhány előredefiniált szó, amely segíti a változók és tömbök használatát.

```
FILL (cim n b –)
```

művelet a címtől kezdve leteszi n byta-re a b értéket, az

```
ERASE (cim n –)
```

ugyanezt teszi b=0-ra. Ezek mellett vannak előredefiniált változók is, amelyek közül felhasználói szempontból a BASE nevű változó az egyik legfontosabb. Ezen keresztül biztosítja a FORTH, hogy a felhasználó tetszőleges számrendszerben dolgozhasson. A rendszer bekapcsolásakor a BASE változó tartalma 10, a továbbiakban az

n BASE !

szósorozattal át lehet váltani más számrendszerbe, a DECIMAL előredefiniált FORTH szóval vissza lehet állítani a 10-es számrendszert. A leggyakrabban használt 16-os számrendszernek is van eredeti FORTH szava, a HEX. A számrendszerek váltása csak a karakteres és bináris alak közötti konverziót érinti, a belső számfeldolgozás mindig bináris, így a számítás sebességére nincs befolyással, hogy melyik számrendszerben dolgozunk.
A 10. fejezetben felsorolt előredefiniált FORTH szavak között külön jelöljük a változókat.

## 2.4. Új FORTH szavak definiálása

Egy új FORTH szó definiálását mindig a : (colon) FORTH szó vezeti be, az ezután beírt szó az új FORTH szó neve, hagyományos értelemben ez idáig az eljárás feje. Ezt követi a szósorozat, amelyet az új szó meghívásakor végre kell hajtani, az eljárás törzse, majd a definíciót a ; (semicolon) szó zárja le. Például a 2.2.3. fejezetben leírt 4ROT szó definíciója:

```
:  4ROT      (n1 n2 n3 n4 – n2 n3 n4 n1)

     ROT   (n1 n2 n3 n4 – n1 n3 n4 n2)
     ROT   (n1 n3 n4 n2 – n1 n4 n2 n3)
     2SWAP(n1 n4 n2 n3 – n2 n3 n1 n4)
     SWAP  (n2 n3 n1 n4 – n2 n3 n4 n1)

;    OK
```

A zárójelek a komment jelei, a közöttük írt dolgokkal a FORTH nem foglalkozik. Ne felejtsük : minden FORTH szó, így a ( elé és után is szóközt

kell tenni! Viszont a ) nem FORTH szó, csak a komment határolójele -- így elé és utána nem kötelező a szóköz.)

A fenti definíciót írhattuk volna egy sorba is:

```
:   4ROT ROT ROT 2SWAP SWAP ; OK
```

de -- főleg bonyolultabb esetekben -- a program jobb áttekinthetősége miatt célszerűbb a verem tartalmát, és ha szükséges egyéb tudnivalókat, a definíció során többször (de legalább a fejében) kiírni.
A definíció beírása után az új szóra ugyanúgy lehet hivatkozni, mint a FORTH saját, előredefiniált szavaira.
Írjunk például definíciót egy szám köbének kiszámítására, ezt a későbbiekben fel fogjuk használni:

```
:   KOB (n -- n-kobe)
    DUP DUP (n -- n n n)
    * *          (n n n -- n-kobe)
;   OK
    5 KOB . 125 OK
```

A FORTH szavak legfeljebb 31 karakter hosszúak lehetnek, tetszőleges karakterekből állhatnak. Például egy szám is lehet művelet neve, s noha ezt a lehetőséget a FORTH programozók ritkán használják, maga a tény, hogy ez lehetséges, a rendszer nagymértékű rugalmasságát mutatja.
Gyakorlásként nézzük végig az alábbi, számrendszereket váltogató és a számokat különböző számrendszerekben kiíró programsorokat!

```
HEX OK

1BD7 4EF + . 20C6 OK

3E 6D * . 1A66 OK

: BINARY 2 BASE ! ; ( -- ; 2-es számrendszer ) OK

: TRINARY 3 BASE ! ; ( -- ; 3-as számrendszer ) OK
```

: OCTAL 8 BASE ! ; ( — ; 8-as számrendszer ) OK

DECIMAL 1395 BINARY . 10101110011 OK

DECIMAL 1395 TRINARY . 1220200 OK

DECIMAL 1395 OCTAL . 2563 OK

DECIMAL 1395 HEX . 573 OK

DECIMAL OK

# 3. VEZÉRLÉSI SZERKEZETEK

## 3.1. Feltételek, feltételes utasítások

A veremben levő mennyiségek logikai értéknek is felfoghatók, a veremben levő szám logikailag hamis, ha értéke 0 és logikailag igaz, ha értéke nem nulla. A FORTH-ban vannak olyan előredefiniált műveletek (és lehet számos további ilyet definiálni), amelynek egyik vagy több paramétere logikai érték. A verem leírásánál ezt az angol flag szó rövidítésével, f-fel jelöljük, pl.:

= (n1 n2 – f)

Értelemszerűen az = FORTH szó két bemenő paramétere, a két összehasonlítandó érték, az = szó meghívásakor a verem tetején van. A művelet elvégzése után a verem tetején 1 vagy 0 van, annak megfelelően, hogy egyenlőek voltak-e vagy sem. Hasonlóan működnek a

> (n1 n2 – f),
< (n1 n2 – f),
0> (n – f)
0< (n – f)

és 0= (n – f) FORTH szavak.

Ez utóbbi három szó elnevezése jelzi, hogy definíciójuk a 0 >, a 0 <, ill. 0 = szósorozat.
Nincs a FORTH előredefiniált szókészletében, de könnyen definiálhatjuk a bemenő logikai érték komplemensét visszaadó NOT műveletet:

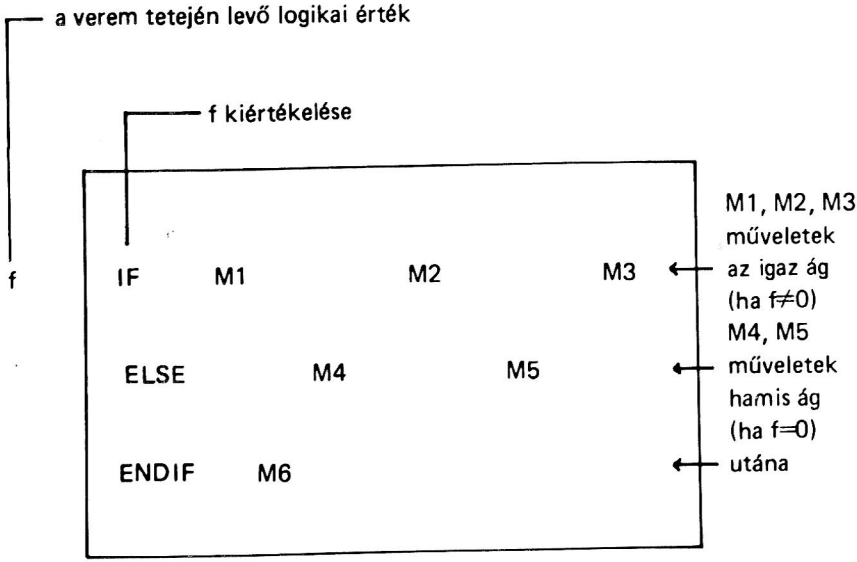
: NOT (f – f ; logikai érték komplemensének képzése)

0= ;

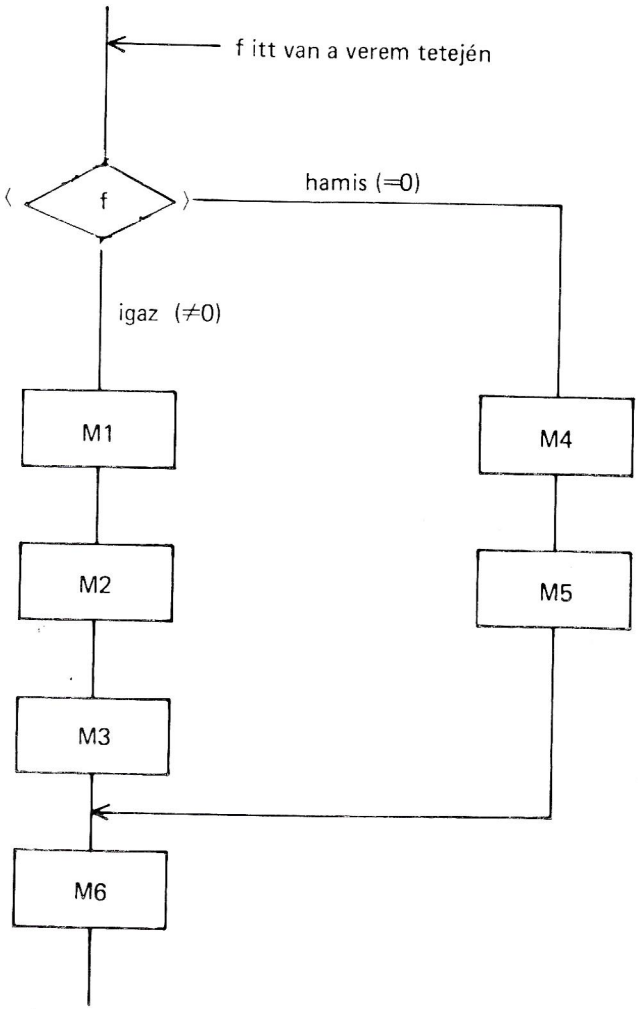
A logikai értékek felhasználásának egyik módja az IF . . ENDIF, ill. IF. . . ELSE. . . ENDIF szerkezet (lásd 3.1.a. és 3.1.b. ábra). Az IF szó kiértékeli a verem tetején levő logikai értéket. Amennyiben az igaz: akkor az IF és az ELSE, ill. az IF és az ENDIF közötti műveletek végrehajtása

következik, és az ELSE és ENDIF közötti szavak kimaradnak. Ha a feltétel hamis, akkor az IF és ELSE, ill IF és ENDIF közötti műveletek maradnak ki és a vezérlés az ELSE, ill. az ENDIF utáni első szóra kerül.

Az egyes vezérlési szerkezeteket két ábrán mutatjuk be. Az első a FORTH szemlélete, a „doboz" előtt a verem teteje látható; M1, M2,. . .-vel a soronkövetkező műveleteket jelöljük. A második ábra folyamatábrás, hagyományos ábrázolása az egyes vezérlési szerkezeteknek.




3.1.a. ábra
Az IF. . . ELSE. . .ENDIF szerkezet szemléltetése




3.1.b. ábra

Az IF...ELSE...ENDIF szerkezet folyamatábrás ábrázolása

Egy egyszerű példa az IF szerkezetre (a szabványos FORTH szókészletben ABS néven található meg):

```
: ABSZOLUT-ERTEK (n – n)
    DUP 0< (n – n f)
    IF MINUS
    ENDIF ; OK
15 ABSZOLUT-ERTEK . 15 OK
-20 ABSZOLUT-ERTEK . 20 OK
```

A MINUS FORTH szó a verem tetején levő szám 2-es komplemensét képezi (azaz megszorozza –1-gyel), és azt adja vissza a veremben.
Az IF szerkezet belsejében további IF szerkezetek szerepelhetnek, csak arra kell vigyázni, hogy minden IF-nek meglegyen az ENDIF párja. Ha véletlenül elfelejtenénk egy vagy több ENDIF-et, akkor a rendszer figyelmeztetést ad. A következő példa az egymásba skatulyázott IF szerkezetek használatát mutatja;

```
: OSZTALYOZO (n – )
    DUP 10 < IF ."APRO" ELSE
    DUP 20 < IF ."KICSI" ELSE
    DUP 30 < IF ."KOZEPES" ELSE
    DUP 40 < IF . "NAGY" ELSE
                . "ORIAS"
    ENDIF ENDIF ENDIF ENDIF DROP ;
25 OSZTALYOZO KOZEPES OK
 4 OSZTALYOZO APRO OK
73  OSZTALYOZO ORIAS OK
```

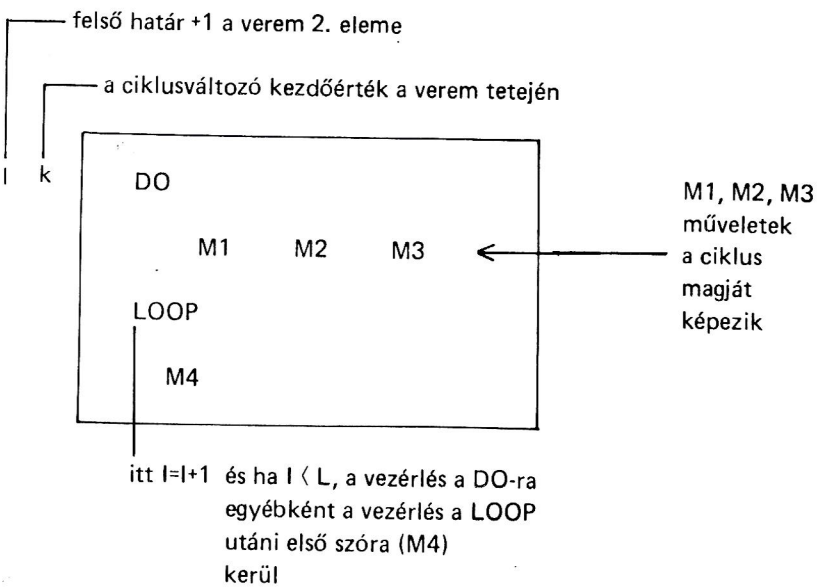
A vezérlési szerkezetek, így az IF...ELSE..ENDIF szerkezet is, csak definíció belsejében szerepelhetnek, közvetlen utasításként nem használhatók. A FORTH előredefiniált szókészletében szerepel az ENDIF szóval ekvivalens THEN szó is.

## 3.2. A DO. . .LOOP szerkezet

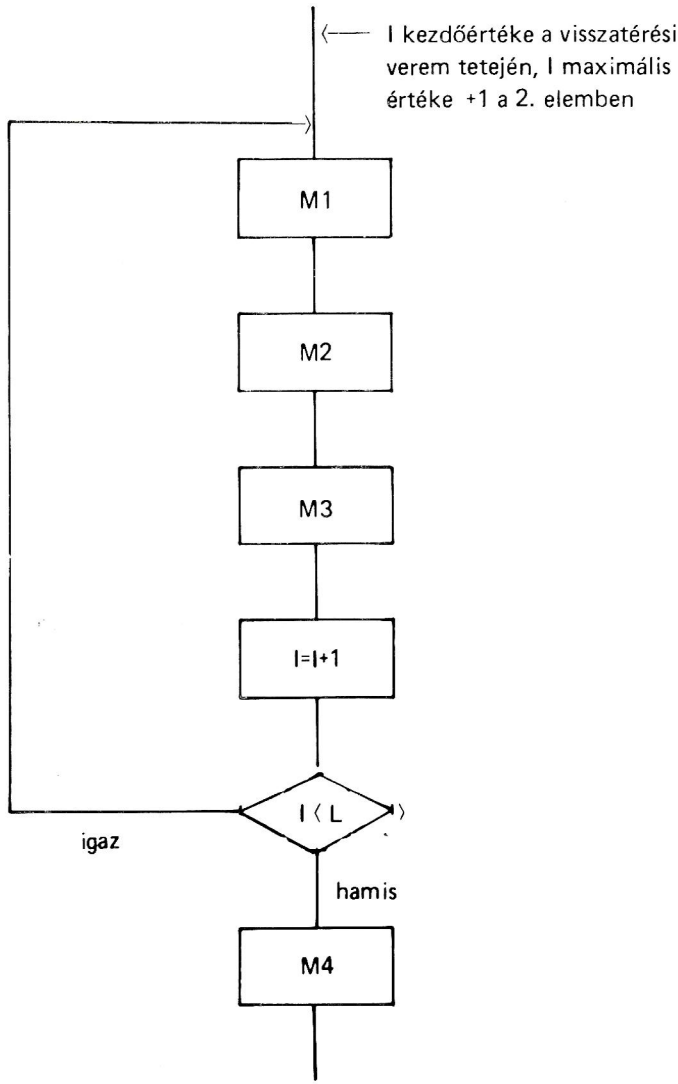
A DO. . .LOOP szerkezet egy egyszerű ciklus (3.2.a. és 3.2.b. ábra), amely a két szó közötti műveleteket n-szer végrehajtja, ahol n a DO. . .LOOP szerkezet két paraméterének a különbsége. A DO szónak paramétere van:

DO (I k –)

ahol I a felső határ +1, k pedig a kezdőérték, a LOOP szónak nincs paramétere. A ciklus magjában fel lehet használni az I ciklusváltozót, amely egy előredefiniált változó. A ciklusváltozó megőrzésére a FORTH a visszatérési vermet használja, ezért ciklus belsejében nem célszerű a visszatérési veremre hivatkozni.




3.2.a. ábra
A DO. . .LOOP szerkezet szemléltetése




3.2.b. ábra
A DO. . .LOOP szerkezet folyamatábrás ábrázolása

Pl.:

```
: DO-TESZT (– ; a számok kiíratása 0–9-ig)

    10 0 DO I . LOOP ;

DO-TESZT 0 1 2 3 4 5 6 7 8 9 OK
```

Fontos megjegyezni, hogy a DO. . .LOOP ciklus egy alkalommal biztosan lefut, hiszen az ellenőrzés a ciklusmag után van. Ez jól látható a

```
: DO-TESZT1

    –1 0 DO I . LOOP;

DO-TESZT1 0 OK
```

példából.

A DO. . .LOOP szerkezetben benne van, hogy a ciklusváltozó minden körében éppen eggyel nő. A lépésszám a

```
+LOOP (n –)
```

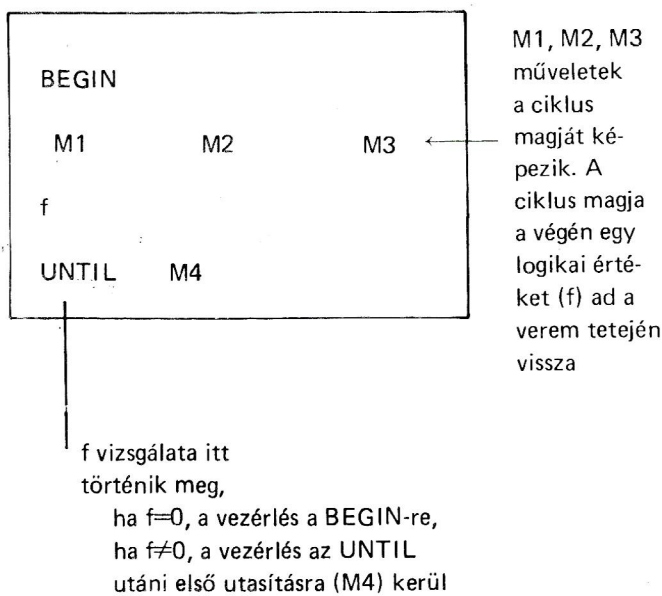
szóval változtatható, ahol n egész szám. Pl. ha csak minden harmadik számot akarjuk kiírni:

```
: DO-TESZT3 (– ; minden harmadik szám kiíratása 0–9-ig)

    10 0 DO I . 3 +LOOP ;

DO-TESZT3 0 3 6 9 ok
```

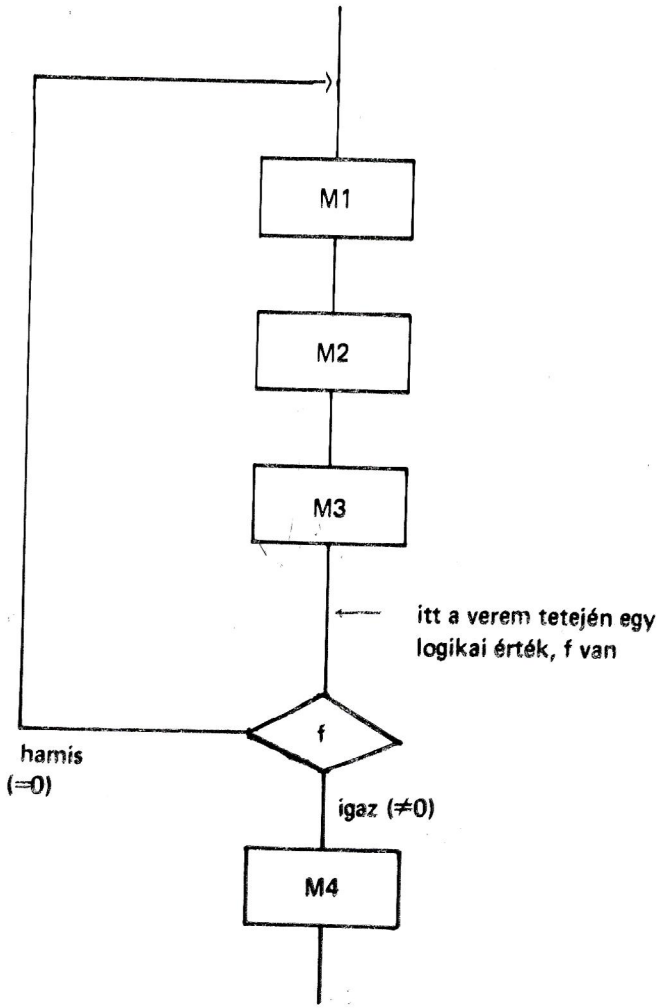
Hasonlóan az IF. . .ELSE. . .ENDIF szerkezethez, a DO. . .LOOP szerkezet is csak definíció belsejében használható.

### 3.3. A BEGIN. . .UNTIL szerkezet

A BEGIN. . .UNTIL szerkezet szintén egy ciklus (3.3.a. és 3.3.b. ábra), amely mindaddig végrehajtja a két szó közötti műveleteket, amíg a ciklus magjában beállított logikai érték igaz nem lesz. Ekkor a ciklus leáll, és a vezérlés az UNTIL utáni első szóra kerül. Ebben a szerkezetben a BEGIN szónak nincs paramétere, az UNTIL (f —) egy logikai értéket vár a verem tetején.




3.3.a. ábra
A BEGIN. . .UNTIL szerkezet szemléltetése




3.3.b. ábra
A BEGIN. . .UNTIL szerkezet folyamatábrás ábrázolása

Példaként írjunk programot az 1000-nél kisebb köbszámok kiíratására:

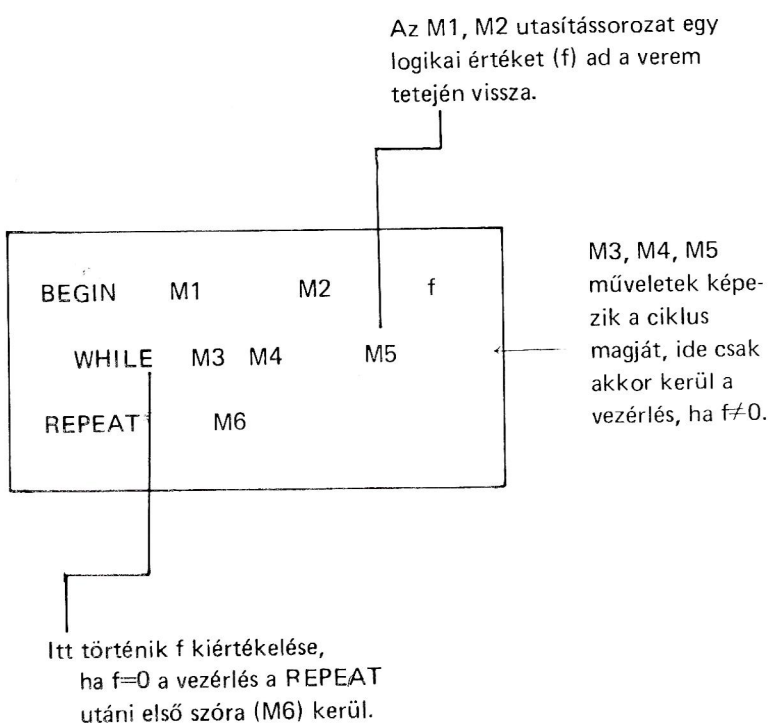
```
: KOBSZAMOK      (– ; köbszámok kiírása 1000-ig)
 0               (az index kezdeti értéke)
 BEGIN           (a ciklus kezdete)
     DUP 1+ SWAP (a szám növelése ; i+1 i)
     CR DUP .    (a szám kiírása új sorba ; i+1 i)
     KOB DUP .   (a szám köbreemelése és ; i+1 i köb)
                 (a köb kiírása)
     1000 =      (ellenőrzés, ahol a köbszám)
                 ( ) 999)
 UNTIL           (a ciklus vége)
 DROP            (az indexre már nincs szükség)
;OK
```

```
KOBSZAMOK

 0    0

 1    1

 2    8

 3   27

 4   64

 5  125

 6  216

 7  343

 8  512

 9  729

10 1000 OK
```

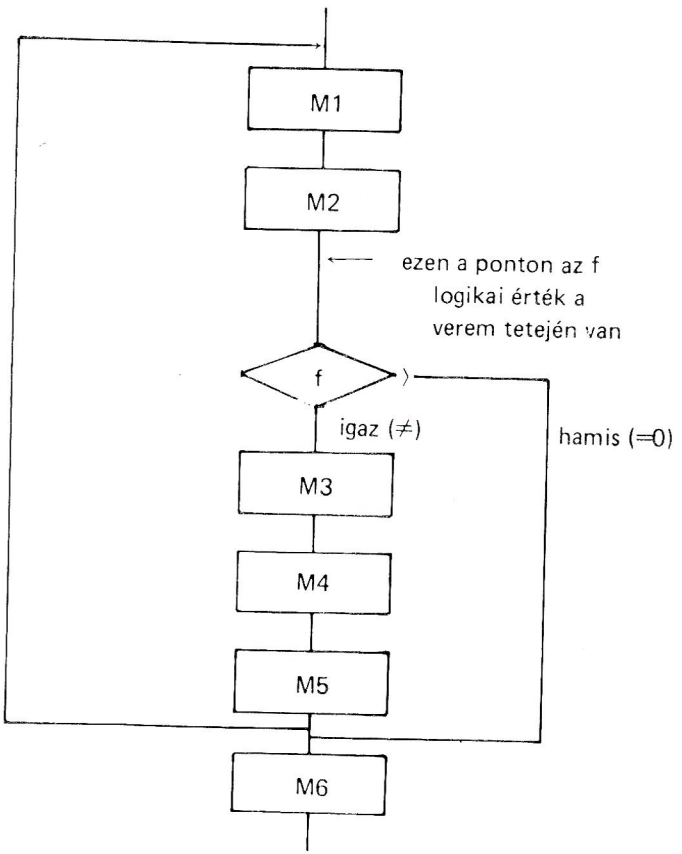
Ebben a programban felhasználtuk a 2.4. fejezetben megírt KOB definíciót. A programban kihasználtuk, hogy az 1000 éppen egy köbszám.
A többi vezérlési szerkezethez hasonlóan a BEGIN. . .UNTIL szerkezet is csak definíció belsejében használható.
A FORTH előredefiniált szókészletében szerepel az UNTIL szóval ekvivalens END szó is.

## 3.4. A BEGIN. . .WHILE. . .REPEAT szerkezet

A BEGIN. . .WHILE. . .REPEAT szerkezet (3.4.a. és 3.4.b. ábra) egy olyan ciklus, amely addig hajtja végre a WHILE és REPEAT szavak közötti műveleteket, a ciklus magját, amíg a BEGIN és WHILE szavak között beállított logikai érték hamis nem lesz. Miután ezt a logikai értéket a ciklusmag eleje értékeli ki, ebben a ciklusban (az eddigiekkel ellentétben) előfordulhat, hogy a ciklusmag végrehajtására egyszer sem kerül sor. Ebben a szerkezetben a BEGIN és a REPEAT szónak nincs paramétere, a WHILE (f —) szó egy logikai értéket vár a verem tetején:




3.4.a. ábra
A BEGIN. . .WHILE. . .REPEAT szerkezet szemléltetése




3.4.b. ábra
A BEGIN. . .WHILE. . .REPEAT szerkezet folyamatábrás ábrázolása

A BEGIN. . .WHILE. . .REPEAT szerkezet használatára a következő (4.1.) fejezetben mutatunk be egy tipikus példát. Ez a vezérlési szerkezet is csak definíció belsejében használható.

# 4. INPUT-OUTPUT MŰVELETEK

## 4.1. Input műveletek

Egy karakter beolvasása a KEY előredefiniált FORTH szóval történik. A KEY szó a beolvasott karakter ASCII kódját adja vissza a veremben. Az

```
: ASCIIKOD KEY . ; OK
```

definíció után meghívott ASCIIKOD szó inputra vár, majd az első beütött billentyű ASCII kódját kiírja a képernyőre.
A KEY DROP szósorozat alkalmas arra, hogy a program futását az első billentyű lenyomásáig felfüggesszük.
Az előző fejezetben megismert BEGIN. . .WHILE. . .REPEAT szerkezet és a KEY szó segítségével írhatunk olyan eljárást, amely egy bizonyos karakter (pl.: space, amely kódját a BL nevű FORTH konstans adja) beütéséig vár:

```
: SPACEVAR (– ; az első space-ig inputra vár)

        BEGIN KEY BL = NOT (Lásd 3.1. fejezet)

        WHILE (ne csináljon semmit)

        REPEAT ; OK
```

Lehetőségünk van egyszerre több karakter beolvasására is. A QUERY szó egy kraktersorozatot vár el az inputon, a karaktersorozat vagy 80 karakterből áll vagy tartalmazza a karaktersorozat végét jelző return-t. A beírt karakterek a terminál input pufferbe (TIB) kerülnek, a karaktersorozat végére egy vagy több 0 kerül. A TIB FORTH szó előredefiniált felhasználói változó, amely a terminál input puffer címét adja vissza a veremben.

Egy általunk meghatározott címre az

```
EXPECT (cim n –)
```

szó segítségével tudunk n karaktert beolvasni. Pl.:

```
0 VARIABLE MEMHELY 10 ALLOT OK

: BEOLV MEMHELY 5 EXPECT ; OK
```

A BEOLV szó egy maximum 5 karakterből álló sorozatot olvas be a billentyűzetről a MEMHELY nevű változó címétől kezdődő 5 byte-ra.

A QUERY és az EXPECT szavak hatására a program megáll, és a billentyűzetről várja egy sor begépelését. A FORTH lehetőséget ad karaktersorozat beolvasására a program szövegéből is. A WORD (c —) szó egy karakter ASCII kódját várja el a verem tetején, ez lesz a beolvasandó karaktersorozat határoló jele, és a programban utána következő krakтereket eddig a határolójelig (vagy bináris 0-ig) beolvassa a HERE nevű előredefiniált felhasználói változó címére úgy, hogy a karaktersorozat hosszát teszi az első helyre. A FORTH interpreter is a WORD szót használja a felhasználói parancs beolvasására, ezért csak definíció belsejében értelmes használni.

Pl. a

```
: SZOKIIRAS (— ; visszaírja a begépelt szót)

        BL WORD HERE COUNT TYPE ; OK
```

egyszerű definiált szó visszaírja a képernyőre az utána következő — szóközig tartó — karaktersorozatot.

```
SZOKIIRAS UJSZO UJSZO OK
```

A COUNT szó egy címet vár a verem tetején, előveszi onnan a karaktersorozat hosszát, és visszaadja a címét (tehát eggyel nagyobb címet, mint amekkorát kapott:

```
COUNT (cím — cím+1 hossz)
```

A TYPE szó kiírja a verem tetején levő hosszban, az alatta megadott címen kezdődő karaktersorozatot

TYPE (cím hossz –)

Figyeljük meg, mi történik, ha a fenti definíciót nem : és ; között írjuk be, hanem egyszerűen begépeljük:

BL WORD UJSZO HERE COUNT TYPE

A szópuffer tartalma, ahová UJSZO-t beírtuk, időközben megváltozott, az inputon utoljára szerepelt szó, TYPE került oda.

A WORD szó működésének megértéséhez programozzuk be az egyébként is igen hasznos SZOVEGELTESZ eljárást, amellyel egy saját magunk által definiált pufferbe tudjuk a program következő szavát elteni:

0 VARIABLE SZOVEG 40 ALLOT (sok hely a szövegnek)

: SZOVEGELTESZ (–)

SZOVEG 40 BLANKS (a SZOVEG nevű puffer)
(szóközökkel való)
(feltöltése)

BL WORD HERE COUNT (a szópuffer címe)

SZOVEG SWAP CMOVE ; OK (másolás SZOVEG-be)

A CMOVE (honnan hova n –)

szó n byte-ot másol át honnan címről hova címre.

További hasznos szövegkezelő szolgáltatás a

–TRAILING (cím hossz1 – cím hossz2)

szó, amely bemenő paraméterként a szöveg címét és a puffer hosszát kapja meg a veremben, kimenő paraméterként pedig meghagyja a verem második

elemeként a szöveg címét és a verem tetejére a szöveg igazi, a végéről levágott szóközök nélkül mért hosszát teszi, így, ha a SZOVEGELTESZ műveletünket ki akarjuk próbálni, ezt a következő SZOVEGKIIR művelet segítségével tehetjük:

```
: SZOVEGKIIR (–)

    SZOVEG 40 –TRAILING TYPE ; OK

SZOVEGELTESZ UJSZOVEG OK

SZOVEGKIIR UJSZOVEG OK
```

Ha a SZOVEGKIIR definícióból kihagytuk volna a –TRAILING-et, úgy az mindig 40 karaktert írna ki.
FORTH programozás közben a számok inputjára a többi inputhoz hasonlóan, általában a vermet használjuk. Előfordul azonban, hogy egy szó végrehajtása közben kifejezetten számot akarunk beolvasni. Ehhez segítséget a

NUMBER (cím – d)

FORTH szó nyújt, amely a kapott címen talált szövegből egy duplahosszúságú számot csinál az éppen érvényes számrendszerben.

```
: SZAMBEOLV (– n)

    CR ." ?"          (egy kérdőjel kiíratása új sorba)

    QUERY             (szövegbeolvasás)

    BL WORD HERE      (a szöveg átmásolása a)
                      (szópufferbe)

    NUMBER            (a szöveg konvertálása)
                      (duplahosszú számmá)

    DROP ; OK         (normál szám)
```

A SZAMBEOLV eljárás egy szám begépelésére vár, és azt a verem tetején hagyja. Ha nem számot kap, a NUMBER hibát jelez.

Példák a SZAMBEOLV művelet használatára:

```
SZAMBEOLV

? 523 OK

. 523 OK

SZAMBEOLV

? 34675 OK

. -30861 OK

SZAMBEOLV

? TYKL TYKL ?

SZAMBEOLV

? ABC ABC ?

HEX SZAMBEOLV . DECIMAL

? A3 A3 OK

HEX SZAMBEOLV . DECIMAL

? A24F -5DB1 OK
```

## 4.2. Output műveletek

Az eddigiekben már több output műveletet megismertünk, a legegyszerűbb mindenek előtt a . (dot), amellyel a verem tartalmát lehet kiíratni. Hasonlóan egyszerű output művelet a ." (dot-quote), amely az előző fejezet végén a SZAMBEOLV definíciójában szerepelt. A ." FORTH szó után az összes többi FORTH szóhoz hasonlóan kötelező a szóköz, amely (az idézőjel nélkül) megjelenik a képernyőn. Pl.:

```
." ALMA" ALMA OK
```

Természetesen a ." szónak a definíciók belsejében van igazi jelentősége, ennek segítségével lehet a programból a képernyőre adott szöveget kiíratni.

Egy karakter kiíratását az

```
EMIT (c –)
```

szó segítségével is megtehetjük, pl.:

```
65 EMIT A OK
```

Ugyanígy karaktersorozatot is ki tudunk írni:

```
73 EMIT 71 EMIT 69 EMIT 78 EMIT IGEN OK
```

Miután ez utóbbi karaktersorozatot lényegesen egyszerűbb ."-val kiírni

```
." IGEN" IGEN OK
```

ezért az EMIT használatának elsősorban azoknál a karaktereknél van jelentősége, amelyek nem írhatók be egy ." műveletbe. Ilyenek például a CR, lapemelés, HOME és a különböző kurzormozgató billentyűk, amelyek ily módon könnyen kiadhatók programból.

Szövegek kiírása adott címről az előző fejezetben már bemutatott TYPE szóval történik, hasonlóan foglalkoztunk már a szöveghossz (COUNT) és a

szóközszűrés (-TRAILING) műveleteivel. Ebben a fejezetben még a számkiírás különböző lehetőségeit mutatjuk be.
Egy előjeles számot meghatározott szélességben a

.R (n1 n2 –)

szóval lehet kiírni, a .R az n1 számot jobbra igazítva írja ki n2 szélességben. Előjeles duplahosszúságú számot a

D. (d –),

meghatározott szélességű duplahosszúságú előjeles számot a

D.R ( d n – )

szóval lehet kiírni. A .R és D.R szavak jól használhatók tabuláláskor, ezt mutatja a következő példa. A DMP szó 64 byte-ból álló darabokat ír ki a memóriából adott címtől kezdve. A 64 byte kiírása után szóközre folytatódik a memória kiírása, bármely más billentyű leütésére az eljárás véget ér. A példában bemutatjuk az aktuális számrendszer elmentését és visszaállítását is.

```
: DMP ( cím – )
        BASE
'CC
                (elmentése)
    HEX
    BEGIN
        DUP 64 + SWAP   (a következő blokk)
                        (címe)
          8 0 DO
              CR DUP I 8 * +
```

```
                DUP 0 4 D.R SPACE   (a cím)
                                    (kiírása)

                8 0 DO

                    DUP I + C

'CC                                 ( byte elővétele )

                    3 .R            ( a byte kiírása)

                    LOOP

                DROP

            LOOP

        DROP CR KEY BL -    ( vár egy billentyű )
                            ( leütésére, ha szó- )
                            (köz, folytatódik a )
                            ( ciklus)

    UNTIL

    DROP

    BASE    !   (az eredeti számrendszer visszaállítása)

    CR ;
```

Ez a példaprogram 4–5 sorban is leírható, a magyarázatok elhelyezése és a könnyebb érthetőség miatt írtuk ilyen szellősen.

A DMP művelet működése pl. a következő lehet:

SZÖVEG DMP

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 905 | 55 | 4A | 53 | 5A | 4F | 56 | 45 | 47 |
| 90D | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| 915 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| 91D | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| 925 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| 92D | 0 | 0 | 8C | 53 | 5A | 4F | 56 | 45 |
| 935 | 47 | 45 | 4C | 54 | 45 | 53 | DA | FA |
| 93D | 8 | E9 | 85 | 3 | 9 | 28 | 80 | 28 |

Mint az előző fejezetben láttuk, a SZOVEG változónak legfeljebb 40 byte helyet foglalhatunk le, memóriatérképünkön ez látszik, a negyvenedik byte után, 92D-től „szemét", illetve az utána definiált SZOVEGELTESZ lefordított alakja van a memóriában, az első sor az UJSZOVEG szónak az ASCII fordítása.

Előjel nélküli számok kiírása az

U. (u –)

szóval történik.

Például:

```
 30000 . 30000 OK

 30000 U. 30000 OK

-30000 . -30000 OK

-30000 U. 35536 OK
```

A FORT további lehetőségeket ad a kiíratások megformázására, ezek részletezésére itt nem térünk ki, de a rendelkezésre álló eszközök szerepelnek a függelékben (lásd a 10. fejezetben a ⟨###S SIGN #⟩ HOLD szavaknál).

# 5. FORRÁSSZÖVEGEK TÁROLÁSA ÉS SZERKESZTÉSE

## 5.1. Forrásszövegek tárolása

A 2–4. fejezetben tárgyalt példáknál mindig feltételeztük, hogy a szövegeket begépeljük a billentyűzeten és a beírt műveleteket a FORTH közvetlenül végrehajtja. Ez tanulási céloknak éppen megfelelő, de a folyamatos munkához nem elegendő. Sőt, még a tanulási fázisban is feltűnik (a képernyő mellett ülő Olvasó is, mire idáig jut az olvasásban, bizonyára követett el legalább egy gépelési hibát, és észrevette), hogy egy definíció javításának egyetlen módja az egész definíció újra begépelése. Másrészt, ha az olvasó a 4.1. és a 4.2. fejezetek olvasása (és példáinak kipróbálása) közben kikapcsolta gépét, akkor a DMP művelet kipróbálásához újra be kellett gépelnie a SZOVEG és a SZOVEGELTESZ szavak definícióját.

Miután a FORTH a [ return ] leütése után azonnal végrehajtja a beírt műveleteket, a definíciók eredeti alakja (a forrásszöveg) azonnal elvész, és csak a lefordított alak (a tárgykód) marad meg. A FORTH biztosítja azt a lehetőséget, hogy a forrásszövegeket háttértárolón megőrizzük.

A forrásszövegeket a FORTH darabokban tárolja, ezek a darabok 1024 byte (1K)-ot tartalmaznak, éppen kitöltenek egy képernyőt, így az angol szóhasználat után (screen) ezeket a darabokat a továbbiakban képernyő-nek nevezzük. A képernyő 1024 byte-ját a FORTH 16 sorra és soronként 64 karakterre bontott egységként kezeli. Minden képernyőt egy szám azonosít (0-tól a háttértároló méretharáig).

Megállapodás, hogy a 0. képernyőbe a képernyőfájl-ra vonatkozó tudnivalókat írjuk. A 0. képernyőt betölteni nem lehet (lásd LOAD), így nem szükséges a komment sorokat zárójelezni. A képernyők 0. sora zárójelek közé írva az adott képernyőre vonatkozó tudnivalókat tartalmazza, pl. mely kulcsszó definíciója található a képernyőben, mikor módosították utoljára stb. A 4–6. képernyők a FORTH rendszer hibaüzeneteit tartalmazzák. Az 1. képernyőbe

szokás írni, hogy a képernyőfájlból mely képernyőket kell betölteni. Ugyanide írjuk a képernyőfájl mentési paramétereit, ha a lefordított programot meg kívánjuk őrizni, így a képernyőfájlok 1 LOAD paranccsal fordíthatók, menthetők. A programsorokat a 7. képernyőtől kezdve szokás begépelni. Mintául a FORTH rendszerhez tartozó FORTHSCR.SCR file szolgálhat.

Egy képernyő tartalmához a legegyszerűbben a BLOCK szó segítségével lehet hozzáférni. Pl. a 3 BLOCK szósorozat után beolvassa a legrégebben változott pufferbe a 3. blokk tartalmát és visszaadja a verem tetején a puffer első byte-jának címét. Miután a képernyőnkénti blokkok száma, a B/SCR konstans megállapodásszerűen 1, így a 3. blokk egyben a 3. képernyőt is jelenti. Ennek kilistázása a LIST (n –) művelettel történik, ahol n a kilistázandó képernyő száma. A LIST kiadása után az aktuális képernyő az éppen kilistázott képernyő lesz, ezt a számot az SCR változó őrzi meg. Nyomtatóra a 3/LAP (n1 n2 –) paranccsal listázhatunk képernyőfájlokat vagy azok egy összefüggő részét. Egy lapra 3 képernyő listája kerül.

Új képernyőt lerögzíteni vagy egy régebben megírtat módosítani, a képernyők sorrendjét megváltoztatni, képernyőket másolni vagy törölni a FORTH rendszerrel adott FORTHEDI.COM képernyő-szerkesztő programmal lehet.

## 5.2. FORTH virtuális memória

A FORTH rendszer a háttértárolókat virtuális memóriakezeléssel éri el. A virtuális háttértároló kezelésnek két tárolószintje van: a külső tároló (diszk egység), illetve a központi memóriában kialakított memóriapuffer. Az információforgalom a tárolók között blokkos formában történik. A blokkok méretét – byte-okban megadott hosszát – a B/BUF (byte/blokk) rendszerkonstans tartalmazza.
A központi memóriában lévő memóriapufferben egyszerre több blokk tárolására van lehetőség.
A kiválasztott blokkok a memóriába automatikusan olvasódnak be, a memóriában módosított blokkok pedig a pufferterület felülírása előtt automatikusan kiíródnak a háttértárra. A blokkok mérete és a blokkpufferek száma a FORTHFIX.COM programmal módosíthatók.

## 5.3. FORTH képernyőszerkesztő

A képernyő-szerkesztő futtatható programjának neve FORTHEDI.COM. A szerkesztő betöltéséhez és indításához gépeljük be a szerkesztő nevét (FORTHEDI) és annak a képernyő-fájlnak a nevét, melyet módosítani akarunk. Ha nem adunk meg fájlnevet, akkor a képernyő-szerkesztő a FORTHSCR.SCR fájlt nyitja meg.
Ha nem adunk meg diszkmeghajtót, akkor a fájlt az aktuális diszkmeghajtón keresi, ha ott nem találja, akkor az A diszkmeghajtón.
Ha nem adtuk meg, akkor a fájl kiterjesztés az SCR. A képernyő-szerkesztő a képernyő-file megnyitása után parancs módba lép és parancs megadására vár. Parancs módból szerkesztési módba az E paranccsal, szerkesztési módból parancsmódba az ESC vezérlő karakterrel térhetünk át.
Parancs és szerkesztési módban is bármikor begépelhető a ^J, ill. LF vezérlő karakter, melynek hatására kiíródnak a képernyőre a rendelkezésre álló parancsok és vezérlő karakterek.
Az egyes parancsok és vezérlő kódok részletes értelmezését a következő oldalakon találjuk.

Parancs mód műveletei

^ Menü kiírása

E Szerkesztés kezdése

\+ Több üres képernyő beszúrása az aktuális fájlban.

– Több képernyő törlése az aktuális fájlból.

C Egyetlen képernyő másolása. Meg kell adni a forrás és a cél képernyő számát.

M Több képernyő mozgatása az aktuális fájlban.

S Az aktuális fájl adott képernyőinek átmásolása egy másik képernyő-fájlba.
Meg kell adni, hogy a forrás-fájl hányadik képernyőjétől kezdve hány

képernyőt kívánunk átmásolni, és a másolás a cél-fájl hányadik képernyőjétől kezdődjön. Ha a cél-fájl még nem létezik, akkor a parancs hatására létrejön a fájl, ha kérjük, így ezzel a paranncsal lehet új képernyő-fájlokat létrehozni. Ekkor az első 7 képernyőt (0–6-ig) szokás átmásolni.

T Tabulációs pontok távolságának beállítása.

U Képernyő-fájl váltás. Minden módosított puffer felíródik a diszkre és új képernyő-fájl neve adható meg. Ha az új fájlt nem lehet megnyitni, akkor újra az előző képernyő-fájl lesz nyitott.

I Az aktuális képernyő-fájl fejléceinek kiírása.

D Diszkkönyvtár listázása.

X A képernyő-szerkesztő elhagyása és FORTH interpreter módba áttérés. Az interpreter módból a képernyő-szerkesztőbe az EDIT paranccsal térhetünk vissza.

ESC Minden módosított puffert diszkre ír, lezárja a képernyő-fájlt és visszatér az operációs rendszerbe.

Edit mód vezérlő kódjai

(Kurzor parancsok)

⟨balra nyíl⟩ A kurzort balra mozgatja. Ha a sor elején állt, az előző sor végére pozicionál. A képernyő tartalma változatlan.

⟨jobbra nyíl⟩ A kurzort jobbra mozgatja. Ha már a sor végén állt, a következő sor elejére pozicionál. A képernyő tartalma változatlan.

⟨fölfelé nyíl⟩ Felfelé mozgatja a kurzort. Ha már a képernyő legfelső sorában állt, a bal margóra pozicionál. A képernyő tartalma változatlan.

⟨lefelé nyíl⟩ Lefelé mozgatja a kurzort. Ha már a legalsó sorban állt, a jobb margóra pozicionál. A képernyő tartalma változatlan.

CR A kurzort a következő sor elejére állítja. Ha a kurzor a legalsó sorban állt, a jobb margóra pozicionál.

I ⟨TAB⟩ A kurzort jobbra mozgatja a következő tabulációs pontra. Ha az aktuális sor utolsó tabulációs pontján állt, a következő sor első tabulációs pontjára ugrik.
A képernyő tartalma változatlan.

O A kurzort balra mozgatja az előző tabulációs pontra. Ha már a bal margót elérte, az előző sor utolsó tabulációs pontjára fog ugrani.
A képernyő tartalma változatlan.

(Szavas parancsok)

A A kurzort előre (jobbra) mozgatja a következő szó elejére.

F Visszafelé (balra) mozgatja a kurzort az előző szó elejére.

C Törli a kurzortól jobbra eső szót. Elhagyja a felesleges space-eket és a sor többi részét felhúzza a kurzorig.

(Sor parancsok)

P Törli az aktuális sort. A többi sor változatlan.

DL Törli az aktuális sort. Minden alatta levő sort eggyel feljebb mozgat, és a 15. sort blank-kel tölti fel.

IL Beír egy üres sort a kurzornál, minden sort eggyel lejjebb mozgat és a 15. sor előző tartalma elvész.

^R — Az aktuális sor átmásolása a sorverembe. A kurzor a következő sorra áll. A sorverem 16 sort tartalmazhat.
A képernyő tartalma változatlan.

^T — A sorverem felső sora az akt. sorba íródik (amelyikben a kurzor áll). Az akt. sor előző tartalma elvész. A kurzort az előző sorba mozgatja. Ha a sorverem üres, nincs művelet.

(Karakter utasítások)

DC — Törli a kurzorpozícióban levő karaktert, a sor másik felét balra mozgatja. A többi sor változatlan marad.

^G — Törli a kurzortól balra eső karaktert, a kurzort és a sor többi részét egy pozícióval balra mozgatja.
A többi sor változatlan.

IC — Betesz egy space-t a kurzorpozícióba, a sor többi részét jobbra mozgatja.
A jobb végén túlcsorduló karakterek elvesznek.
A többi sor változatlan.

^V — Szövegbeszúrás-felülírás mód váltó.
Szövegbeszúrás esetén a karakterek beíródnak a kurzor pozícióba, a sor jobbra tolódik. A sor jobb végén túlcsorduló karakterek elvesznek. Felülírás módban a begépelt karakter felülírja a kurzornál álló karaktert.

(String utasítások)

^Q — String-keresés.

^E — String-keresés és -felülírás.
Az új string egyenlő hosszú v. rövidebb kell legyen, mint a keresett.

(Képernyő parancsok)

| | |
|---|---|
| CLR | Törli az egész screen-t és a kurzort a screen bal felső sarkába állítja. |
| ^N | Diszkre írja az aktuális screen-t és a következő screen-re tér. |
| ^B | Diszkre írja az aktuális screen-t és az előző screen-re tér. |
| ^W | Felírja diszkre az akt. screen tartalmát és visszatér a további szerkesztéshez. |
| ^L | „Eldob" minden módosítást és visszaállítja a szerkesztés előtti állapotot. |

(Vegyes utasítások)

| | |
|---|---|
| ^J | Vezérlő kódok menüjének kiírása.<br>A screen tartalma változatlan. |
| ESC | Elhagyja a szerkesztési módot és parancs módba tér. |

A képernyő szerkesztő indítása:

)FORTHEDI fájlnév, ahol a fájlnév a szerkesztendő képernyőfájl neve. A kiegészítésnél a .SCR az alapértelmezés – ha megfelel, nem kell kiírni. A fájlnévnél pedig az alapértelmezés a FORTHSCR.SCR, tehát a )FORTHEDI( CR ) parancs hatására az aktuális fájl a FORTHSCR.SCR lesz.

## 5.4. FORTHFIX program

A FORTH indulásakor foglal le a számára kijelölt memóriarész végéről néhány puffert. Ezek rendre a legmagasabb címtől visszafele:

- képernyőpufferek,
- USER tábla,
- visszatérési verem,
- terminál input puffer (TIB),,
- paraméter verem.

Felhasználói program esetén magának a FORTH-nak a kijelentkezése szükségtelen. Ezért az letiltható. Ugyanilyen meggondolásból a felhasználó kérheti vagy letilthatja egy másik file megnyitását programja indulásakor. 8 bites, I8080 processzoros, UPM operációs rendszerrel rendelkező más VIDEOTON gépre is áttehető a FORTH. A képernyőkezelő parancsokat kell mindössze installálni.
A fentebbiek megoldására szolgál a FORTHFIX.COM program. Indítása a következő: )FORTHFIX fájlnév, ahol a fájlnév egy futtatható FORTH program neve. A .COM kiterjesztés elhagyható. A FORTHFIX programmal a következő értékek állíthatók.

- *Kijelentkezés*
  A program indításkor és befejezéskor is a FORTH írni fog a képernyőre, valamint a C begépelésekor a vezérlés visszaadódik a monitornak.

- *Bevesz*
  A program indításakor a programnév után megadott nevű fájlt a FORTH megnyitja.
  Ha fájl nevet vagy kiegészítést mégsem adnak meg, a FORTH a fájl nevet vagy a kiegészítést az alapértelmezés szerinti fájlnévként megadott fájlnévből veszi.

- *Alapfile*
  A program indításakor a FORTH megnyit egy kötött nevű fájlt, mégpedig azt, amelyet az alapértelmezés szerinti fájlnévként jelöl ki a későbbiekben.

– *USER tábla*
A felhasználói változók területének a hossza.

– *VV és TIB*
A visszatérési verem és a terminál input puffer együttes hossza.

– *BYTE/SZEKTOR*
Szektoronkénti byte-ok száma

– *#BUF*
Screen-pufferek száma, minimum 2 szükséges.

– *B/BUF*
Egy képernyő-puffer hossza.

– *LIMIT*
A FORTH által használt memória végcíme+1.
Ez csak 16 bites forth esetén állítható.

– *GÉPTIPUS*
A képernyőkezelő szekvenciák installálása a megadott géptípusra.
Ezt csak 8 bites FORTH esetén kérdezi le a program.

A FORTHFIX program indítása:

)FORTHFIX fájlnév, ahol a fájlnév a módosítandó FORTH nyelvű program neve. A COM kiegészítés az alapértelmezés, ezt nem fontos kiírni. A program indulásakor visszakérdez a fájlnévre. Csak a nagy Y betűt fogadje el igennek, a többi karaktert nemnek tekinti.

# 6. SZÓTÁRAK A FORTH-BAN

## 6.1. A szótárak

Az 1.5. pontban említettük, hogy a FORTH szavak egymáshoz vannak láncolva. A FORTH lehetőséget ad arra, hogy a különböző szótárakat is láncoljuk. A szótár logikailag összefüggő szavak halmaza. A szótárak együttese pedig egy fastruktúrát alkot, felfogható egy többszintű szótárként is.

A FORTH egyidejűleg két kitüntetett szótárt kezel: az egyik, amelyben a keresés először történik, ez az úgynevezett CONTEXT szótár; a másik, amelyikbe a fordítás folyik, ennek neve CURRENT szótár. Ha a rendszer a felhasznált szó definícióját nem találja a CONTEXT szótárban, a keresést tovább folytatja a CURRENT szótárban is. Bármely szótár kijelölhető CURRENT és CONTEXT szótárnak is. Pl. az EDITOR szó kijelöli az EDITOR szótárat CONTEXT szótárnak, az EDITOR DEFINITIONS szósorozat pedig CURRENT szótárnak is.

A szótárak segítségével ugyanazzal a névvel több, más-más funkciójú szó definiálható.

A szótárak „információelfedő" szerepüknél fogva nagyban segítik a moduláris programozást. Ennek az információelfedésnek az a lényege, hogy egy-egy nagyobb absztrakt objektum leírásánál el tudjuk rejteni azokat a lokális, a külvilágra nem tartozó FORTH szavakat, amelyek az adott objektum belügyei, esetleg veszélyes, ellenőrzés nélküli funkciókat valósítanak meg. Azokat a műveleteket, amelyeknek globális funkciójuk van, a 0. szintű FORTH nevű szótárban definiáljuk. Az itt elmondottakra példa az 5. fejezetben bemutatott EDITOR szótár, amely a külön modulként létező szerkesztő program betöltése után érhető el. Ez tartalmaz olyan FORTH szavakat, amelyek nem szerkesztési üzemmódban egész más jelentéssel bírhatnak, ezért a szavai általában nem elérhetőek, csak akkor, amikor explicit behívjuk az EDITOR szótárat.

Új szótárak szabadon definiálhatók. A szótárban való keresés ilyenkor is végighalad a CONTEXT szótártól a fa gyökeréig, a FORTH szótárig.

## 6.2. Egy szótárelem felépítése

A szótárelem (vagyis a FORTH szó térbeli képe) két fő részből áll, a bejegyzés fejéből és törzséből. Egy szótárbejegyzés feje a következőket tartalmazza:

a. a bejegyzés neve (ez változó hosszú)

b. az előző bejegyzés nevének a címe

c. a bejegyzés által használt gépi kód kezdőcíme

E mezőknek a kezdőcíme rendre a névmező címe (NFA), az előző bejegyzés névmezőjének a címe (LFA) és a kódmező címe (CFA).
A szótárbejegyzés törzse (ezt paraméter mezőnek is hívják, és címe a PFA) tartalmazza azokat az információkat, amelyek alapján a szótári elem dolgozik. Kettőspont definíció esetén a paraméter mező tartalmazza a definícióban szereplő szavak végrehajtási címeit (azaz a CFA-kat), amelyek végén a ;S címe van, ezzel fejeződik be a definíció végrehajtása. Kettőspont-definíció esetén egy szótári bejegyzés a következőképpen épül fel:

| | |
|---|---|
| NFA | név |
| LFA | mutató az előző NFA-ra |
| CFA | kódcím mutató |
| PFA | végrehajtási cím1 |
| | végrehajtási cím2 |
| | |
| | a ;S végrehajtási címe |

A FORTH lehetőséget ad a különböző mezők egymásba konvertálására: a PFA szó egy szótári bejegyzés NFA-ját viszi át a PFA-jába; a CFA szó a PFA-t a CFA-ba; az LFA a PFA-t az LFA-ba és az NFA a PFA-t az NFA-ba.
Nincs definiálva, de könnyen elvégezhető a CFA és LFA közötti konverzió, hiszen CFA=LFA+2, és PFA=CFA+3.
CONSTANT és VARIABLE esetén a PFA-tól kezdve csak egyetlen szó található, ahol a konstans, illetve a változó értéke van.
Ezeket az ismereteket felhasználva definiálhatjuk a FORTH-ban a rekurziót:

```
: MYSELF          (egy művelet meghívja önmagát)
    LATEST        (az utolsó szó első byte-jának a címe)
    PFA CFA       (a konverzió a végrehajtási címre)
    9             (a cím lefordítása)  (vesse!)
; IMMEDIATE       (fordítás alatt fusson le ez a definíció)
```

Példaként programozzuk be a faktoriális kiszámítását:

```
: FAKT            ( n – n! )
    –DUP          ( duplázzon, ha n≠0)
    IF DUP 1–
       MYSELF *
    ELSE 1
    ENDIF ;
5 FAKT . 120 OK
```

A MYSELF szóra azért van szükség, ill. azért nem lehet helyére a FAKT szót írni, mert a szótárban a FORTH csak azokat a szavakat találja meg, amelyeknek fordítása már megtörtént.

# 7. A FORTH KITERJESZTHETŐSÉGE

A FORTH rendszer – amint az eddigiekből látható – könnyen bővíthető, mert minden művelet, amelyet a felhasználó definiál, a rendszer szerves részévé válik.
Ezen kívül további lehetőség a FORTH kiterjesztésére a meglevők mellé új adat- és vezérlés struktúrák definiálása. A FORTH biztosítja az eszközöket ezekhez a kiterjesztésekhez. Az alábbiakban ezeket az eszközöket mutatjuk be, és adunk ízelítőt használatukból.

A FORTH-ban különböző szintű tevékenységeket lehet végezni. A legalacsonyabb szintű tevékenység egy, a szótárban már létező szó meghívása és futtatása, nevezzük ezt nulladik szintnek. Eggyel magasabb szint egy definiáló szó használata. Ilyenek például a már megismert VARIABLE vagy : szavak; ezek létrehoznak egy szótári elemet (egy szót), amelyben egy nulladik szintű tevékenységet lehet végrehajtani (1-es szint). Az ennél is magasabb szintű tevékenység az, amikor az 1-es szinten használt definiáló szavakat készítjük el (2-es szint).
A FORTH-ban minden definiáló szó egy olyan kis fordítóprogramnak tekinthető, amelynek célja egy új szerkezet elhelyezése a szótárban. Hasonlóan ahhoz, ahogy az 1-es szintű tevékenységek (egy-egy új szó létrehozása) kibővítik a FORTH nyelvet, a definiáló szavak létrehozása (a 2-es szint) kibővíti a FORTH fordítóprogramot. A FORTH-ban alapvetően két szó szolgálja ezt a célt, a ⟨BUILDS és a DOES⟩ . A ⟨BUILDS szó segítségével leírhatjuk a keletkező adatstruktúra méretét, az esetleges inicializálás módját, a DOES⟩ az adatelérési algoritmust adja meg.

A ⟨BUILDS . . . DOES⟩ szerkezetet a következőképpen használjuk:

```
: SZERKEZET ⟨BUILDS xxxx DOES⟩ yyyy ;
```

ahol az xxxx és yyyy szavak tetszőleges szósorozatot alkotnak. Eredetileg ez egy normális kettőspont-definíció, ami 2-es szintűvé teszi, az éppen a ⟨BUILDS és DOES⟩ szavak használata. A ⟨BUILDS után következő xxxx szavak írják le a SZERKEZET-tel definiált új szó felépítését. A DOES⟩ -t

követő yyyy szavak határozzák meg, hogy az új szó mit csináljon. Az yyyy szavakat a FORTH a SZERKEZET definíciójába fordítja, és az új szó meghívásakor kerül rájuk a vezérlés. Ezek szerint a ⟨BUILDS-et követő xxxx szavak fordításkor, a DOES⟩ után írt yyyy szavak pedig végrehajtáskor futnak le.

A SZERKEZET definiáló szót ezek után így használjuk:

```
SZERKEZET PELDA
```

A verem tetején értékeket lehet megadni a SZERKEZET szó definíciójától függően. Ez így a PELDA szó definiálása, tehát egy 1-es szintű művelet. A továbbiakban a PELDA-t a többi szóhoz hasonlóan lehet meghívni (nullás szintű művelet), ekkor futnak le a SZERKEZET szó definíciójában a DOES ⟩ után megadott yyyy szavak.

E szerkezet használatára a legegyszerűbb példa a VARIABLE szó definiálása:

```
: VARIABLE ⟨BUILDS , DOES⟩ ;
```

A VARIABLE szó meghívásakor a ⟨BUILDS hatására a FORTH beolvassa a következő szót és beteszi azt a szótárba. A VARIABLE-nek meg kell adni egy értéket a verem tetején, a ⟨BUILDS után következő "," szó ezt beteszi a szótárba. Ez az érték a változó kezdőértéke. A definiált változó meghívásakor a DOES⟩ visszaadja annak szótárbeli címét és erről a címről lehet a változó értékét a @ művelettel kiolvasni, illetve ide lehet új értéket a ! művelettel betenni. Ebben a példában a DOES⟩ után már nem szerepel FORTH szó, csak a ";" ennek jelentése most is az, hogy vége a definíciónak, a változóval nincs több teendő.

Gyakorlásképpen definiáljuk az n byte-ból felépülő CARRAY tömböt, és kezdjük el használni.

```
: CARRAY                    ( n – ; byte-okból álló )
                            ( tömböt definiáló szó)
      ⟨BUILDS
```

```
        0 DO  0  'C,                ( helyfoglalás byte- )
    LOOP                            ( oknak ciklusban )
    DOES> + ; OK

64 CARRAY TOMB OK

3 5 TOMB C! OK

5 TOMB C@ . 3 OK
```

A CARRAY tömb definíciójánál meg kell adni a definiálandó tömb hosszát, a <BUILDS utáni ciklus eddig az n számig megy, azaz ennyi byte-nak foglal le helyet. A DOES> utáni + szó mondja meg, hogy mit kell csinálni majd a CARRAY-vel definiált szó meghívásakor. A 64 CARRAY TOMB szósorozat az új definiáló szó használatát mutatja. Definiáltunk egy 64 byte-os tömböt TOMB névvel. A következő sor a TOMB nevű tömb hatodik elemébe betesz egy hármast. Ekkor fut le a DOES> utáni + szó, hiszen a tömb első eleme a DOES> -zal visszaadott cím, ehhez ötöt hozzáadunk, így kapjuk meg a tömb hatodik elemét.
Amint ígértük, ez csak ízelítő a FORTH rendszer kiterjeszthetőségéből, de az itt leírtakat elsajátítva az Olvasónak lehetősége nyílik számos más struktúra kialakítására is.

# 8. A FORTH INDÍTÁSA

A FORTH rendszer a diszken 4 fájlból áll.

a., A FORTH0.COM fájl tartalmazza az alaprendszert.

)FORTH0 fájlnév ⟨CR⟩

Hatására elindul a FORTH. Megnyitja a fájlnév nevű fájlt. Kijelentkezik. Kiírja az 'ok' üzenetet, amely jelzi, hogy parancsot vár. Ha a fájlnév kiegészítését nem adtuk meg, akkor .SCR kiegészítést feltételez. Ha egyáltalán nem adtunk meg fájlnevet, akkor a FORTHSCR.SCR fájlt fogja megnyitni. Ha szükségünk van a FORTHSCR.SCR fájlban található kulcsszavakra, akkor tegyük ezt a fájlt aktuálissá a USING paranccsal, ha indításkor nem ezt adtuk volna meg. Ezután 1 LOAD-dal fordíttathatjuk be a kulcsszavakat, vagy ha nem mindre van szükségünk, akkor a kulcsszó kezdőképernyőjének sorszámát kell megadni a LOAD-nak.
Ha a lefordított kulcsszavakkal együtt ki kívánjuk menteni a programot, mentés előtt néhány értéket be kell állítani (a számok decimálisak):

```
LATEST      12 +ORIGIN  !        ( legmagasabb NFA )
HERE        30 "ORIGIN  !        ( DP )
```

A FORTHFIX programnál részletezett kijelentkezik, bevesz, alapfájl flag-ek is állíthatók rendre az 57, 59, 61, +ORIGIN szón. Az alapértelmezés szerinti fájl neve a DEFAULT-FILE T"12345678123" paranccsal állítható be. 11 karakteren kell a fájlnevet megadni. A kiegészítés elé nem kell '.' karakter. Ha úgy kívánjuk, hogy a kimentett program indításakor rögtön egy kulcsszó (pl. a START) végrehajtásával kezdődjön, akkor a CFA-jának címét egy szóra le kell tenni:

```
' START CFA 44 +ORIGIN !
```

Ha pedig úgy kívánjuk, hogy a kimentett program az aktuális képernyőfájl valahányadik képernyőjének értelmezésével induljon, a BOOT-SCREEN nevű változóba tároljuk el a képernyő sorszámát.
Ezután már csak a SAVE parancs kiadása marad:
SAVE fájlnév ⟨CTR⟩
A fájlnév kiegészítése mindenképpen .COM lesz, akármit is adunk meg. Ha a kimentett programban szeretnénk a fenti értékeken módosítani, ezt a FORTHFIX.COM programmal tehetjük meg. Mentésre példát a FORTHSCR.SCR fájl 1. képernyőjében találunk.

b., A FORTHFIX.COM fájl tartalmazza az 5.4. fejezetben leírt funkciókat. Indítása a program leírásánál található.

c., A FORTHEDI.COM fájl tartalmazza a teljes FORTH.COM fájlt és a képernyő-szerkesztő programot.
Indítása a program leírásánál található.

d., A FORTHSCR.SCR fájl tulajdonképpen egy minta képernyőfájl. Néhány, a FORTH-ban való programozást segítő kulcsszó definícióját tartalmazza.
Ez az alapértelmezés szerinti fájl a FORTHO.COM és a FORTHEDI.COM programok számára.

# 9. FORTH GYORS REFERENCIA

*Konvenciók*

n, n1 . . . 16 bites előjeles szám

d, d1 . . . 32 bites előjeles szám

u . . . 16 bites nem előjeles szám

addr . . . memória cím

b . . . 8 bit

c . . . 7 bites ASCII-karakter

f . . . flag

ff . . . hamis flag

tf . . . igaz flag

len . . . hossz

count . . . byte-szám

A leírás szerkezete:

KULCSSZÓ STACK MAGYARÁZAT

A verem teteje jobbra van. A nyíl baloldalán az input verem, jobb oldalán az output verem látható.
Ez a gyűjtés nem tartalmaz minden szót, ahhoz lásd a kézikönyv FORTH szójegyzék fejezetét, amely tartalmazza a teljes listát.

*STACK műveletek*

| | | |
|---|---|---|
| –DUP | n ⟶ n? | duplikálja az n-t, ha nem zero |
| 2DROP | d ⟶ | töröl egy dupla hosszúságú számot a stack-ből. |
| 2DUP | d ⟶ d d | duplikál egy dupla hosszúságú számot. |
| 2SWAP | d1 d2 ⟶ d2 d1 | kicserél két dupla hosszúságú számot a stack tetején. |
| )R | n ⟶ | átviszi n-t a "return stackbe". |
| DEPTH | ⟶ n | Leteszi n-t a stack-be, n a stack-ben levő 2 byte-os elemek száma. |
| DROP | n ⟶ | töröl egy szimpla hosszúságú számot |
| DUP | n ⟶ n n | duplikál egy szimpla hosszúságú számot. |
| OVER | n1 n2 ⟶ n1 n2 n1 | a stack tetejére másolja a stack második értékét. |
| PICK | n1 ⟶ n2 | az n1. stack-elemet a stack tetejére másolja. |
| R | ⟶ n | n-t átmásolja a visszatérési stack-ből. |
| R) | ⟶ n | kiveszi n-t a visszatérési stack-ből |
| ROLL | n1 ⟶ n2 | n1. elemet kiveszi és a stack tetejére teszi. |
| ROT | n1 n2 n3 ⟶ n2 n3 n1 | a harmadik elemet a stack tetejére teszi |

| | | |
|---|---|---|
| SWAP | n1 n2 ⟶ n2 n1 | kicserél két szimpla hosszúságú számot a stack tetején. |

*Számrendszerek*

| | | |
|---|---|---|
| BINARY | ⟶ | beállítja a bináris rendszert. |
| DECIMAL | ⟶ | beállítja a decimális rendszert. |
| HEX | ⟶ | hexadecimálisra állítja a bázist. |
| BASE | ⟶ addr | ez a rendszerváltozó tartalmazza az érvényes számrendszert. |

*Aritmetikai és logikai műveletek*

| | | | | | | |
|---|---|---|---|---|---|---|
| * | n1 | n2 | ⟶ | szorzat | | szorzás |
| */ | n1 | n2 | n3 | ⟶ n4 | | n4 = n1 * n2/n3 |
| */MOD | n1 | n2 | n3 | ⟶ n4 | n5 | n4 maradék, n5 hányados |
| + | n1 | n2 | ⟶ | összeg | | összeadás |
| +– | n1 | n2 | ⟶ | n3 | | n1 felveszi n2 előjelét |
| – | n1 | n2 | ⟶ | különbség | | kivonás |
| / | n1 | n2 | ⟶ | hányados | | osztás |
| /MOD | n1 | n2 | ⟶ | n3 | n4 | n3 maradék, n4 hányados |
| 1+ | | n1 | ⟶ | n2 | | inkrementálás eggyel |
| 1– | | n1 | ⟶ | n2 | | dekrementálás eggyel |
| 2+ | | n1 | ⟶ | n2 | | inkrementálás kettővel |
| 2– | | n1 | ⟶ | n2 | | dekrementálás kettővel |
| 2* | | n1 | ⟶ | n2 | | előjeles szorzás kettővel |
| 2/ | | n1 | ⟶ | n2 | | előjeles osztás kettővel |
| ABS | | n | ⟶ | abszolút | | leteszi n abszolút értékét |
| AND | u1 | u2 | ⟶ | and | | bitenkénti logikai "és" |
| D+ | d1 | d2 | ⟶ | összeg | | kétszavas szám összege |
| D– | d1 | d2 | ⟶ | különbség | | kétszavas szám kivonása |
| DABS | | d | ⟶ | abszolút | | abszolút érték |

| | | | | | | |
|---|---|---|---|---|---|---|
| DMINUS | | d | → | −d | | előjelet cserél |
| M* | n1 | n2 | → | d | | dupla pontosságú szorzat |
| M/ | d | n1 | → | n2 | n3 | vegyes nagyságú osztó |
| M/MOD | ud2 | u2 | → | u3 | u4 | nem előjeles vegyes osztó |
| MAX | n1 | n2 | → | max | | leteszi a nagyobb számot |
| MIN | n1 | n2 | → | min | | leteszi a kisebb számot |
| MINUS | | n | → | −n | | cseréli az előjelet |
| MOD | n1 | n2 | → | maradék | | leteszi a n1/n2 maradékát |
| OR | u1 | u2 | → | or | | bitenkénti logikai "vagy" |
| S−>D | | n | → | d | | duplaszó hosszúságú szám egy szimplából |
| TOGGLE | addr | b | → | | | kizáró "vagy" a b és az addr tartalma között |
| U* | u1 | u2 | → | ud | | nem előjeles szorzás |
| U/ | ud1 | u1 | → | u2 | u3 | nem előjeles osztás |
| XOR | u1 | u2 | → | xor | | bitenkénti kizáró "vagy" |

*Összehasonlítás*

| | | | | |
|---|---|---|---|---|
| 0< | n | → | f | igaz, ha n < zero |
| 0= | n | → | f | igaz, ha n = zero |
| 0> | n | → | f | igaz, ha n > zero |
| < | n1 n2 | → | f | igaz, ha n1 < n2 |
| = | n1 n2 | → | f | igaz, ha n1 = n2 |
| > | n1 n2 | → | f | igaz, ha n1 > n2 |
| D< | d1 d2 | → | f | igaz, ha d1 < d2 |
| D= | d1 d2 | → | f | igaz, ha d1 = d2 |
| D> | d1 d2 | → | f | igaz, ha d1 > d2 |
| U< | u1 u2 | → | f | igaz, ha u1 < u2 |

*Memória műveletek*

| | | |
|---|---|---|
| ! | n addr → | eltárolja n-t az addr címre ("store") |
| !L | n seg offset→ | elrakja az n 16 bitjét a címre (hosszú tárolás) |

| | | |
|---|---|---|
| | n ⟶ | szó tárolása a következő szabad szótári szóra |
| +ORIGIN | n ⟶ addr | eltolás az origin-tól |
| 2! | d addr ⟶ | 32 bites tárolás |
| 2@ | addr ⟶ d | 32 bites betöltés |
| @ | addr ⟶ n | 16 bites betöltés |
| @L | seg offset ⟶ n | 16 bites betöltés (hosszú töltés) |
| ALLOT | n ⟶ | helyet foglal |
| C! | b addr ⟶ | byte tárolása az addr címen |
| C!L | b.seg offset ⟶ | byte „hosszú tárolása" |
| C, | b ⟶ | byte tárolása a következő szabad szótári byte-on |
| C@ | addr ⟶ b | byte betöltése |
| C@L | seg offset ⟶ b | byte „hosszú betöltése" |
| +! | n addr ⟶ | n-t ad a címen levő értékhez |

*Diszk hozzáférés*

| | | |
|---|---|---|
| --> | ⟶ | következő screen-nel folytatni a fordítást |
| BLK | ⟶ addr | változó, érvényes blokk |
| BLOCK | n ⟶ addr | diszkblokk olvasás a címre |
| BOOT-SCREEN | ⟶ addr | változó, hideg intításkor betöltendő screen számát tartalmazza |
| DISK-ERROR | ⟶ addr | változó, diszk státusz |
| EMPTY-BUFFERS | ⟶ | törli a diszk-puffereket |
| FLUSH | ⟶ | módosított puffert diszkre ír |
| LOAD | n ⟶ | n. screen fordítása/végrehajtása |
| RW | addr block f ⟶ | flag: 0 olvasás, 1 írás |
| SCR | ⟶ addr | változó, érvényes képernyő sorszáma |
| SCREEN-FCB | ⟶ addr | screen-FCB címét teszi a stack-be |
| THRU | n1 n2 ⟶ | betölti a screen-eket n1-től n2-ig |
| UPDATE | ⟶ | kijelöli az utolsó blokkot írásra |

*Operációs rendszer interface*

| | | |
|---|---|---|
| ? TERMINAL | ⟶ f | igaz, ha karakter érkezett |
| BYE | ⟶ | visszatérés az operációs rendszerhez |
| CLEARSCREEN | ⟶ | törli a képernyőt |
| CONSOLE | ⟶ | kiválasztja a konzolt outputként |
| CR | ⟶ | kiír egy CR-LF-et |
| EMIT | C ⟶ | egy karaktert küld az output eszközre |
| FDOS | funkc param ⟶ reg1 reg2 | hívja az operációs rendszer megfelelő funkcióját |
| GOTOXY | X Y ⟶ | beállítja a kurzort |
| KEY | ⟶ C | olvas egy ASCII karaktert |
| PRINTER | ⟶ | kiválasztja a printer outputként |
| SAVE | ⟶ | memóiraképet felírja |

*Output formázás*

| | | |
|---|---|---|
| # | d ⟶ d | a duplaszámból ASCII karaktert generál |
| #⟩ | d ⟶ addr n | előkészíti a stringet a TYPE-nak |
| #S | d ⟶ d | ASCII szöveget generál a szöveg output pufferben |
| . | n ⟶ | kiírja az adott számot |
| ." | | kiír egy stringet a "-ig |
| .CPU | ⟶ | kiírja a CPU azonosítót |
| .FCB | addr ⟶ | kiírja a file nevét az FCB-ből |
| .LINE | line scr ⟶ | kiírja a szöveg egy sorát |
| .R | n mezőszélesség ⟶ | jobbra igazítva ír |
| .STACK | ⟶ | kiírja a paraméterstack tartalmát |
| ⟨# | ⟶ | output string kezdete |
| ? addr | ⟶ | kiírja az addr tartalmát |
| BL | ⟶ blank | ASCII blanket tesz a verembe |
| C/L | ⟶ n | a soronkénti karakterszámot beteszi a verembe |
| D. | d ⟶ | dupla szám nyomtatása |

| | | |
|---|---|---|
| D.R | d mezőszélesség ⟶ | dupla szám kiírása jobbra igazítva |
| HOLD | C ⟶ | karaktert szúr be output stringbe |
| ID. | addr⟶ | kiírja a definíció nevét |
| MESSAGE | n⟶ | kiírja a 4-es screen n. sorát |
| OUT | ⟶ addr | változó, az output karakterek száma |
| SIGN | n d⟶ d | előjelet szúr be az output stringbe |
| SPACE | ⟶ | kiír egy space-t |
| SPACES | n⟶ | kiír n space-t |
| TYPE | addr n ⟶ | n karakter hosszú string kiírása |
| U. | n⟶ | kiír egy előjel nélküli számot |

*Konverzió és string-kezelés*

| | | |
|---|---|---|
| -TRAILING | addr n1 ⟶ addr n2 | levágja a szöveg végi blank-eket |
| BLANKS | addr n ⟶ | n blank-et tesz az addr címre |
| CMOVE | addr1 addr2 len⟶ | len számú byte-ot átmozgat a kisebb címtől kezdve |
| COUNT | addr1 ⟶ addr2 n | előkészítés a TYPE-hoz |
| DIGIT | c n1 ⟶ff vagy n2 tf | konvertálja a c-t az n1 bázis alapján |
| ENCLOSE | addr c ⟶ addr n1 n2 n3 | szöveg elemző primitív |
| ERASE | addr n ⟶ | n byte-ot kinulláz |
| EXPECT | addr count ⟶ | stringet olvas a klaviatúráról |
| FILL | addr n b ⟶ | a b-t n byte hosszan tárolja az addr címtől |
| MATCH | n1 n2 n3 n4 ⟶ f n5 | string hasonlítás az editornak |
| MOVE | addr1 addr2 len ⟶ | szavas mozgatás |
| NUMBER | addr ⟶ d | dupla számmá konvertálás |
| QUERY | ⟶ | input string a TIB-be |

| | | |
|---|---|---|
| RMOVE | addr1 addr2 len ⟶ | len számú byte-ot átmozgat, a magasabb címtől kezdi |
| S= | addr addr2 len ⟶ f | igaz, ha a két string egyenlő |
| WORD | C-- | inputot olvas a c delimiterig |

*DISPLAY*

| | | |
|---|---|---|
| 3/LAP | n1 n2 ⟶ | kiírja a screen-eket n1-től n2-1-ig, hármat 1-1 lapra |
| INDEX | n1 n2 ⟶ | az n1 n2 screen-ek címsorainak kiírása |
| LIST | n ⟶ | az n. screen kiírása |
| TRIAD | n ⟶ | kiír 3 screen-t az n-től |
| VLIST | ⟶ | kiírja a CONTEXT szótárat |

# 10. A FORTH SZAVAK LISTÁJA ÉS MAGYARÁZATA

Ebben a fejezetben felsoroljuk a FORTH-ban előredefiniált összes szót az ASCII karakterek növekvő sorrendjében. Előbb minden szó szimbolikus leírását adjuk meg, majd röviden ismertetjük a szó jelentését.
A szimbolikus leírásban a következő jelöléseket alkalmazzuk:

cim 16 bites memóriacím

b 1 byte (b az alsó 8 bit, a felső 8 bit 0)

c ASCII karakter (c az alsó 7 bit, a felső 9 bit 0)

d 32 bites előjeles egész (duplaszó)

n 16 bites előjeles egész

u 16 bites előjel nélküli egész

f logikai érték (hamis, ha =0; igaz, ha ≠0)

tf igaz logikai érték (nem nulla)

ff hamis logikai érték (nulla)

A szimbolikus leírásban a — jel bal oldalán a veremnek a művelet végrehajtása előtti, jobb oldalán a művelet végrehajtása utáni állapotát írjuk el. A verem teteje mindkét oldalon jobbra van.
Amelyik szónak van külön angol elnevezése, azt * *-ok között adjuk meg.

További jelölések a szövegben:

pfa a szó paraméter-mezőjének címe,

nfa a szó névmezőjének címe,

lfa a szót megelőző szó nfa-ját tartalmazó cím,

cfa a szó kódmezőjének címe.

Az adott szónál jelezzük, ha csak a 16 bites FORTH-ban van meg vagy pedig a FORTHSCR.SCR fájlban található.

! (n cim –) * store *

A 16 bites n értéket leteszi a cim-re.

!CSP * store c-s-p *

Leteszi a veremmutató aktuális értékét a CSP nevű változóba. Ezt a műveletet a fordító ellenőrzési célokra használja.

!L (n seg offset –)

A 16 bites n értéket leteszi a szegmens, eltolás értékekkel megadott címre. Csak 16 bites FORTH esetén!

# (d1 – d2) * sharp *

A d1 duplahosszú szóból a következő (a legkisebb helyiértékű) számjegy ASCII kódját beteszi a számpuffer következő pozíciójára (ennek címe a HLD változóban van). A verem tetején (d2) a BASE-ben levő számmal való osztás eredményét kapjuk vissza. A # műveletet ⟨# és #⟩ között lehet alkalmazni. (Lásd még #S).

#> (d – cim n) * sharp-greater *

A számkonverzió befejezése, a verem második elemében visszaadja a számpufferben kialakult szöveg címét, a verem tetején pedig a hosszát; ezután meg lehet hívni a TYPE műveletet.

#BUFF (– n) * sharp-buff *

Visszaadja a verem tetején (n) a rendszerhez rendelt pufferek számát. Az input-output műveletek normális végrehajtásához szükséges, hogy a visszaadott szám 1-nél nagyobb legyen.

#S (d1 – d2) * sharp-s *

A d1 duplahosszú számot a # művelet felhasználásával számjegyenként átkonvertálja ASCII kódú karaktersorozattá és a számpufferbe teszi. A #S műveletet ⟨# és #⟩ között lehet alkalmazni.

&FF (– 255)

Konstans, értéke decimális 255.

' (– cim) * tick *

Használata: ' xxxx

Az xxxx-szel jelölt szó paraméter-mezőjének a címét, azaz pfa-ját adja vissza a verem tetején. Fordítási direktívaként, szódefiníció belsejében a címet literálként fordítja le. Ha az xxxx szó nincs a szótárban, hibajelzést kapunk.

( * paren *

Használata: ( xxxx)
Megjegyzés kezdetét jelzi. Az xxxx szöveget az interpreter nem dolgozza fel. A ( után kötelező a szóköz, a )-nek a ( szóval egy sorban kell lennie.

(.") ( – ) * bracket-dot-quote *

A ." szó hatására lefordított futásidejű eljárás, amely a definíció belsejében utána következő szöveget kiírja a kiválasztott output eszközre (lásd még: .").

(;CODE) * bracket-semi-colon-code *

A ;CODE szó hatására lefordított futásidejű eljárás.
Assembly utasítások követik. Ezen utasítássor 1. byte-jának címét építi be az utoljára definiált szó kódmezőjébe, azaz CFA-jába (lásd még: ;CODE).

(+LOOP) ( n – ) * bracket-plus-loop *

A +LOOP szó hatására lefordított futásidejű eljárás. Megnöveli n-nel a ciklusváltozót és ellenőrzi, hogy nincs-e vége a ciklusnak (lásd még: +LOOP).

(ABORT) * bracket-abort *

Akkor kerül rá a vezérlés, amikor egy hiba után a WARNING változó értéke –1. Általában az ABORT szót hajtja végre, de (kellő óvatossággal) a felhasználók is megváltoztathatják. Ha pl. azt akarjuk, hogy a USER-ERROR szót hajtsa végre a rendszer hiba esetén, akkor

–1 WARNING !
' USER-ERROR CFA ' (ABORT) !

sorozatot kell végrehajtani.

(DO) ( n1 n2 – ) * bracket-do*

A DO szó hatására lefordított futásidejű eljárás. A ciklusvezérlő paramétereket a visszatérési verembe teszi át (lásd még: DO).

(FIND) * bracket-find *

( cim1 cim2 – pfa b tf )
( cim1 cim2 – ff )

A szótárban cim2-től (ez egy szótárbeli szó, a névmező címének, nfa-jának a címe) kezdve a szótár végéig az nfa-kban keresi a cim1-en

található szónak megfelelő mintát. Ha talált ilyen mintát, akkor a logikai érték igaz, és ilyenkor visszaadja a talált szó paraméter-mezőjének címét, a pfa-ját, valamint a névmező első byte-ját. Ha nem talált a szótár végéig megfelelő mintát, csak a hamis logikai értéket adja vissza a veremben.

(LINE) ( n1 n2 – cim n ) * bracket-line *

Visszaadja a verem második elemében az n2-vel jelölt képernyő n1-dik sorának címét. A verem tetején (n) a sor hosszát kapjuk vissza (64).

(LIT") ( – – cím ) * bracket-lit-quote *

A LIT" futásidejű eljárása. A verem tetejére teszi az őt követő szövegkonstans címét.
A szövegkonstans ezután pl. COUNT TYPE szekvenciával listáztatható.

(LOOP) * bracket-loop *

A LOOP szó hatására lefordított futásidejű eljárás. Eggyel megnöveli a ciklusváltozót és ellenőrzi, hogy nincs-e vége a ciklusnak (lásd még: LOOP).

(NUMBER) ( d1 cím1 – d2 cím2 ) * bracket-number *

A cim1+1 címen levő ASCII szöveget az aktuális számrendszernek megfelelően duplaszavas számmá alakítja és hozzáadja d1-hez, ez az érték lesz d2. Cim2 az első nem konvertálható számjegy címe. A NUMBER használja.

(OF) ( n1 n2 – n1 )

Az OF futásidejű eljárása.

* ( n1 n2 n3 ) * times *

Visszaadja a verem tetején a két előjeles szám szorzatát (n3).

*/ ( n1 n2 n3 – – n4 ) * times-divide *

Az n1*n2 szorzatot egy 32 bites duplaszón állítja elő, majd ezt a duplahosszú számot elosztja n3-mal. Az eredmény (n4) 16 bites.

*/MOD ( n1 n2 n3 – n4 n5) * times-divide-mod *

Az n1*n2 szorzatot egy 32 bites duplaszón állítja elő, majd ezt a duplahosszú számot elosztja n3-mal. Az eredményt a verem tetején 16 biten n5-ben, a maradékot a verem második elemeként n4-ben kapjuk vissza.

+ ( n1 n2 – n3 ) * plus *

Visszaadja a verem tetején a két előjeles szám (n1, n2) összegét (n3).

+! ( n cim – – ) * plus-store *

A cim-en levő értékhez hozzáad n-t.

+– ( n1 n2 – n3) * plus-minus *

Ha n2 negatív, n3=–n1, egyébként n3=n1, tehát ez a művelet mindig n1 számértékét adja vissza a verem tetején n2 előjelétől függő előjellel.

+BUF ( cim1 – cim2 f ) * plus-buff *

A cim1 diszkpuffercím helyett a következő puffer cim2 címét adja vissza a verem második elemeként. Az f logikai érték akkor lesz hamis, ha a PREV változó értéke cim2.

+LOOP futáskor: ( n1 – ) * plus-loop *
fordításkor: ( cim n2 – )

A DO. . .+LOOP szerkezet része, csak definícióban használható. Futásidőben a +LOOP a fenti szerkezetnek az a része, amely ellenőrzi, hogy vissza kell-e térni a DO-hoz vagy sem. Az előjeles n1 hozzáadódik a

ciklusváltozóhoz és ezt az értéket hasonlítja össze a ciklushatárral. A ciklusnak vége van, ha pozitív n1 esetén a ciklusváltozó nem kisebb, negatív n1 esetén pedig nem nagyobb a ciklushatárnál. A +LOOP szó fordításkor a (+LOOP) futásidejű szóra és a veremben levő cim-re fordul le. A verem tetején levő n2 fordításidejű hibaellenőrzéshez kell.

+ORIGIN ( n – cim ) * plus-origin *

Az indítási paraméter-terület n-edik byte-jának címét adja vissza. Az indítási paraméterek megváltoztatásához lehet felhasználni.

, ( n – ) * comma *

Beteszi az n értéket a szótár következő üres szavába és kettővel megnöveli a szótár szabad címmutatóját.

– ( n1 n2 – n3 ) * subtract *

Visszaadja a verem tetején a két előjeles szám különbségét (n3=n1-n2).

—) ( – ) * next-screen *

Az interpretálás a következő képernyőn folytatódik.

–1 ( – –1 )

Konstans, értéke mínusz egy.

–DUP ( n1 – n1 n1 ) * dash-dup *
( 0 – 0 )

A verem tetején levő értéket akkor duplázza meg, ha nem 0.
Gyakorlatilag egy IF előtt érdemes használni, meg lehet vele takarítani egy ELSE DROP ágat.

-FIND ( – pfa b tf ; ha talál ) * dash-find *
( – ff ; ha nem talál)

A forrásszöveg következő (szóközzel befejezett) szavát a HERE változó címére leteszi, és a CONTEXT, majd a CURRENT szótárban keres ilyen nevű elemet. Ha talál, akkor visszaadja a veremben a szó pfa-ját, fölötte a névmező első byte-ját és egy igaz logikai értéket (hasonlóan a (FIND)-hoz). Ha nem talál, akkor csak a hamis logikai értéket adja vissza a verem tetején.

-TRAILING ( cim n1 – cim n2) * dash-trailing *

A cim-en kezdődő n1 hosszú karaktersorozat végéről levágja a szóközöket és n2-ben az így megrövidített szöveg hosszát adja vissza.

( n – ) * dot *

Kiírja az előjeles 16 bites n számot az aktuális számrendszernek megfelelően. A számot egy szóköz követi.

* dot-quote *

Használata: ."xxxx"

Definíció belsejében az "-lel határolt xxxx karaktersorozat a (.") futásidejű eljárás utáni címre fordul le, ennek feladata lesz a szöveg kiírása az aktuális output eszközre. Ha nem definícióban fordul elő, akkor azonnal kiírja a szöveget a határoló "-ig (lásd még: (.") ).

.CPU * dot-c-p-u *

Kiírja a mikroprocesszor nevét az ORIG+22H címről, ahol 32 bites 36 alapú egészre kódolt alakban van letéve.

.LINE ( n1 n2 – ) * dot-line *

Kiírja a képernyőre az n2-dik képernyő n1-dik sorát a sorvégi szóközök nélkül.

.R ( n1 n2 – ) * dot-r *

Kiírja az n1 számot egy n2 hosszúságú mezőbe, jobbra igazítva. Szóközt nem ír utána.

.STACK ( – ) * dot-stack *

Kiírja a verem elemeit anélkül, hogy megváltoztatná a vermet. A FORTHSCR.SCR fájlban található.

/ ( n1 n2 – n3) * divide *

Visszaadja a verem tetején a két előjeles szám hányadosát (n3=n1/n2).

/MOD ( n1 n2 – n3 n4 ) * divide-mod *

Visszaadja a verem tetején a két előjeles szám hányadosát (n4=n1/n2) és az osztás maradékát (n3). A maradék előjele megegyezik az osztandóéval.

0 ( – 0 )

Konstans, értéke nulla.

0< ( n – f ) * zero-less *

Ha a verem tetején levő szám (n) kisebb, mint nulla, akkor igaz logikai értéket ad vissza, egyébként hamisat.

0= ( n – f ) * zero-equals *

Ha a verem tetején levő szám (n) nulla, akkor igaz logikai értéket ad vissza, egyébként hamisat.

0> ( n – f ) * zero-greater *

Ha a verem tetején levő szám (n) nagyobb, mint nulla, akkor igaz logikai értéket ad vissza, egyébként hamisat.

0BRANCH ( f – ) * zero-branch *

A feltételes elágazás futásidejű eljárása. Ha a logikai érték hamis, akkor a kódban következő szó által mutatott címre ugrik előre vagy hátra. Az IF, az UNTIL és a WHILE szavak használják.

1 ( – 1)

Konstans, értéke egy.

1+ ( n1 – n2 ) * one-plus *

A verem tetején levő számot (n1) eggyel megnöveli (n2).

1– ( n1 – n2 ) * one-minus *

A verem tetején levő számot (n1) eggyel csökkenti (n2).

2 ( – 2 )

Konstans, értéke kettő.

2! ( d cim – ) * two-store *

Elteszi a 32 bites d értéket a cim-re.

2+ ( n1 – n2 ) * two-plus *

A verem tetején levő számot (n1) kettővel megnöveli (n2).

2– ( n1 – n2 ) * two-minus *

A verem tetején levő számot (n1) kettővel csökkenti (n2).

2* ( n1 – n2 ) * two-times *

A verem tetején levő számot (n1) megszorozza ketővel (n2).

2/ ( n1 – n2 ) * two-divide *

A verem tetején levő számot (n1) elosztja kettővel (n2).

2@ ( cim – d ) * two-fetch *

Visszaadja a verem tetején a címen levő 32 bites számot.

2DROP ( n1 n2 – ) * two-drop *

Törli a verem két felső elemét.

2DUP ( n1 n2 – n1 n2 n1 n2 ) * two-dup *

Megduplázza a verem két felső elemét. Megegyezik az OVER OVER szósorozattal.

2SWAP (n1 n2 n3 n4 – n3 n4 n1 n2 ) * two-swap *

Megcseréli a verem felső két pár elemét.

3/LAP ( n1 n2 – ) * három-slash-lap *

Kilistázza az aktuális képernyőfájlból a képernyőket n1-től n2-1-ig. Minden lapra fejlécet ír. Egy lapra 3 képernyő kerül.

3DROP ( n1 n2 n3 – )

Törli a verem három felső elemét. Csak 16 bites FORTH esetén!

4 ( – 4 )

Konstans, értéke négy.

4DROP ( n1 n2 n3 n4 – )

Törli a verem négy felső elemét. Csak 16 bites FORTH esetén!

* colon *

Használata: : xxxx yyyy ;

Létrehoz egy új szót xxxx névvel a CURRENT szótárban. Az új szó az yyyy szósorozatból áll, a szósorozatot ; zárja. A CONTEXT szótár ugyanaz lesz, mint a CURRENT, beállítja a fordítási módot.

* semi-colon *

Lezárja a szó definiálását és megállítja a fordítást. A futásidejű ;S-re fordul le.

;S * semi-colon-s *

Befejezi az éppen végrehajtott szót, ill. ha képernyőt interpretálunk, akkor annak az interpretálását. Ezt a futásidejű szót fordítja be a szótárdefiníciók végén levő ; szó is, és visszaadja a vezérlést a hívó eljárásnak.

( n1 n2 – f ) * less-than *

Ha a verem második eleme (n1) kisebb, mint a verem tetején levő (n2), akkor igaz logikai értéket ad vissza, egyébként hamisat.

<# * less-sharp *

A számkonverzió nyitó művelete. A számkonverzió a következő műveletek sorozatából áll:

<# # # #S HOLD SIGN #>

A számkonverzió a veremben levő duplahosszú számon történik, és az eredmény a PAD előtti számpufferben keletkezik.

## ⟨BUILDS

Használata egy kettőspont-definícióban:

```
: xxxx ⟨BUILDS yyyy
       DOES⟩ zzzz ;
```

Az új xxxx szó definiáló szó, amelyet a következőképpen használunk:

```
xxxx nnnn
```

Ilyenkor keletkezik egy nnnn nevű új szó és egyúttal az yyyy szavak végrehajtására kerül sor. Amikor később az nnnn szót hívjuk meg, akkor a verembe az nnnn szó pfa-ja kerül, és a zzzz szavak végrehajtása történik meg. A ⟨BUILDS . . . DOES⟩ szerkezet magasszintű futásidejű eljárások megírását teszi lehetővé (nem pedig assembly kódú eljárását, mint a ;CODE).

## =

( n1 n2 – f ) * equals *

Ha a verem két felső eleme egyenlő, akkor igaz logikai értéket ad vissza, egyébként hamisat.

## ⟩

( n1 n2 – f ) * greater-than *

Ha a verem második eleme (n1) nagyobb, mint a verem tetején levő (n2), akkor igaz logikai értéket ad vissza, egyébként hamisat.

## ⟩R

( n – ) * to-r *

A verem tetején levő számot a visszatérési verembe teszi át. Egy definíción belül párban kell állnia az R⟩ szóval.

## ?

( cim – ) * question-mark *

A cim-en levő értéket kiírja az érvényben levő számrendszerben. A FORTHSCR.SCR fájlban található.

? COMP * query-comp *

Hibaüzenetet ad, ha éppen nem fordítás folyik.

? CSP * query-c-s-p *

Hibaüzenetet ad, ha a veremmutató aktuális értéke nem egyezik meg a CSP változóban megőrzöttel.

? DECIMAL

Hibaüzenetet ír, ha az aktuális számrendszer nem a 10-es.

? ERROR ( f n – ) * query-error *

Az n. számú hibaüzenetet kiírja, ha a verem második elemében lévő logikai érték igaz.

? EXEC * query-exec *

Hibaüzenetet ad, ha éppen fordítás folyik.

? LOADING * query-loading *

Hibaüzenetet ad, ha nem diszkről való betöltés folyik. (Nem volt LOAD kiadva.)

? PAIRS ( n1 n2 – ) * query-pairs *

Hibaüzenetet ad, ha a verem két felső eleme nem egyezik meg. Az üzenet arra utal, hogy fordításkor a feltételes vagy ciklusképző szavak párosítása nem megfelelő.

? STACK * query-stack *

Hibaüzenetet ad, ha a verem túlcsordult.

? TERMINAL ( – f ) * query-terminal *

Lekérdezi, hogy valamelyik billentyű éppen le van-e nyomva. Igaz logikai értéket ad vissza, ha igen. Ha karakter érkezett be, mindaddig igaz logikai értéket ad vissza ez a kulcsszó, amíg KEY-vel ki nem olvassuk a karaktert.

@ ( cim – n ) * fetch *

Beteszi a verembe a cim-en levő 16 bites értéket.

@L ( seg offset – n)

Beteszi a verembe a szegmens, eltolás értékkel megadott cím szavas tartalmát. Csak 16 bites FORTH esetén!

ABORT

Törli a vermeket és végrehajtási állapotba áll át.

ABS ( n – u )

Visszaadja a verem tetején levő szám (n) abszolút értékét (u).

AGAIN ( cim n – ; fordításkor )

Használata egy definíció belsejében:

BEGIN . . . AGAIN

Futásidőben az AGAIN visszaadja a vezérlést a megfelelő BEGIN-re. A ciklus magától nem ér véget.
Fordításkor az AGAIN a BRANCH futásidejű eljárást fordítja be és a verem második elemében levő címet. A verem tetején levő n a fordításidejű ellenőrzéshez kell.

ALLOT ( n – )

A DP nevű szótármutató változóhoz hozzáadja az n előjeles számot. Terület foglalására használható.

AND ( n1 n2 – n3 )

A verem két felső elemén (n1,n2) végzett bitenkénti logikai ÉS művelet eredményét adja vissza (n3).

ASCII

Használata definícióban: ASCII cccc.
A WORD kulcsszóval megkeresteti az utána álló szöveget. A szöveg első karakterének ASCII kódját literálként beépíti a definícióba.

B/BUF ( – n ) * b-slash-buf *

Konstans, amely visszaadja a veremben a képernyőblokk hosszát. A képernyőblokk megállapodásszerűen 16 db 64 byte-os sorból áll, így e konstans értéke 1024.

B/SCR ( – n ) * b-slash-screen *

Konstans, amely visszaadja a képernyőnkénti blokkok számát. A képernyő megállapodás szerint 1 blokkból áll, így e konstans értéke 1.

BACK ( cim – )

Befordítja a következő szabad szótári címre a veremben megadott visszaugrási címet. A ciklusvéget jelző kulcsszavak használják a ciklus elejére ugráshoz.

BASE ( – cim )

Felhasználói változó, amely az aktuális számrendszer értékét tartalmazza.

BEGIN ( – cim n ; fordításkor )

Használata kettőspont-definíció belsejében:

BEGIN . . . UNTIL
BEGIN . . . AGAIN
BEGIN . . . WHILE . . . REPEAT

Futásidőben a BEGIN egy olyan szósorozat elejét jelzi, amelyet többször végre lehet hajtani. Ez a visszatérési pont a megfelelő UNTIL, AGAIN vagy REPEAT szavakból. A vezérlés UNTIL esetén akkor adódik vissza a BEGIN-hez, ha a verem tetején levő logikai érték hamis, AGAIN és REPEAT esetén mindig visszakerül.
Fordításkor a BEGIN visszaadja a verem második elemében a visszatérési címet és a verem tetején az n értéket a fordításidejű ellenőrzéshez.

BINARY

Hatására az aktuális számrendszer a kettes lesz. A FORTHSCR.SCR fájlban található.

BL ( – c) * b-l *

Konstans, amely a szóköz karakter ASCII kódját adja vissza a verem tetején.

BLANKS ( cim u – )

A címen kezdődő u hosszúságú memóriadarabot szóközökkel tölti fel.

BLK ( – cim ) * b-l-k *

Felhasználói változó. Ha 0, akkor a terminál input pufferből történik az interpretálás, egyébként értéke a LOAD-dal interpetált blokk címe.

BLOCK ( n – cim )

Visszaadja a verem tetején annak a blokk-puffernek a címét, amely az n-edik blokkot tartalmazza. Ha a blokk nincs a memóriában, akkor beolvassa a diszkről abba a pufferbe, amelynek a tartalma legrégebben

változott. Ha annak a blokknak a tartalma, amelyet most éppen felülírunk változott (a puffer meg van jelölve az UPDATE-tel), akkor a felülírás előtt a puffer tartalmát kiviszi a diszkre (lásd még: BUFFER, R/W, UPDATE, FLUSH).

BOOT-SCREEN ( – cim )

Visszaadja a verem tetején annak a változónak a címét, amely a hidegindításkor betöltendő képernyő sorszámát tartalmazza.
Ha a változó értéke nulla, a képernyőbetöltés elmarad (lásd LOAD).

BRANCH

A feltétel nélküli elágazás futásidejű eljárása. A BRANCH-et az ELSE, az AGAIN és a REPEAT fordítja be.

BUFFER ( n – cim )

Veszi a következő memória puffert és hozzárendeli az n. blokkhoz. Ha a puffer előző tartalma meg volt jelölve UPDATE-tel, akkor az kiíródik a diszkre. A blokkot nem olvassa be a diszkről. A verem tetején visszaadja a blokk-puffer címét. Felhasználónak nem ajánlott a használata.

BYE

Lezárja a diszken levő, a képernyőt tartalmazó fájlt és visszaadja a vezérlést az operációs rendszernek.

C! ( b cim – ) * c-store *

A 8 bites b értéket leteszi a verem tetején levő cim-re.

CIL ( b seg offset – )

A 8 bites b értéket leteszi a szegmens, eltolás értékekkel megadott címre. Csak 16 bites FORTH esetén! .

C,  ( b – )  * c-comma *

A verem tetején levő 8 bites értéket leteszi a szótár következő üres byte-jára. A szótár szabadhely mutatóját eggyel megnöveli.

C/L  ( – n )  * c-slash-l *

Konstans, az EDITOR által használt sorhosszot adja vissza a verem tetején byte-okban. A FORTHSCHR.SCR fájlban található.

C@  ( cim – b )  * c-fetch *

Visszaadja a verem tetején levő cím 8 bites tartalmát (b).

C@L  ( seg offset – b )

Visszaadja a szegmens, eltolás értékekkel megadott cím 8 bites tartalmát (b). Csak 16 bites FORTH esetén!

CASE  ( – n; fordításnál )

Használata: n CASE
n1 OF utasítássorozat1 ENDOF

.....

nk OF utasítássorozatk ENDOF
utasítássorozatX ENDCASE

A CASE egy IF...ELSE sorozat logikai egyszerűsítése, kényelmesebben kezelhető leírási módja.
Végrehajtáskor a CASE előtt megadott paramétert (n) sorra összehasonlítják az OF-ok az előttük álló paraméterekkel (n1,n2, . . .,nk) mindaddig, míg egyezést nem találnak. Ekkor az ezen OF-hoz tartozó utasítássorozat végrehajtódik és a vezérlés az ENDCASE-re kerül.

Ha végig nem talált egyezést, az utasítássorozatX fog végrehajtódni. Vigyázat! Az ENDCASE törli a verem legfelső elemét (hogy a veremből eltüntesse n-t). Így, ha az utasítássorozatX a verembe adatot tesz, ügyelni kell a helyes sorrendre.

CASE:

Definíciós szó. Használata:

CASE: cccc xxxx0 xxxx1 . . . ;

Létrehozza a cccc kulcsszót, melynek törzsébe sorra beépíti az xxxx0, xxxx1 stb. kulcsszavakat.
Programelágazás valósítható meg vele. A cccc szó végrehajtásakor egy sorszámot vár a verem tetején. Ha a sorszám 0, a vezérlés az xxxx0 szóra kerül, majd a cccc után állóra.
Ha a sorszám 1, a vezérlés az xxxx1 szóra kerül, majd a cccc után állóra stb.
Hibás n paraméter esetén nincs hibaüzenet, csak hibás működés.

CFA ( pfa – cfa ) * c-f-a *

Egy definíció paraméter mező címéből (pfa) visszaadja ugyanannak a definíciónak a kód-mező címét (cfa).

CLEARSCREEN

Törli a képernyőt.

CMOVE (cim1 cim2 n ––) * c-move *

A verem tetején megadott számú (n) byte-ot átmásol cim1-ről cim2-re. A másolás a cim1-en kezdődik és halad a magasabb memóriacímek felé. (lásd még RMOVE)

CMOVEL ( seg1 offset1 seg2 offset2 n – )

n db byte-ot mozgat át az első címről a másodikra. Mindkét címet szegmens, eltolás formában kell megadni. Csak 16 bites FORTH esetén!

COLD

Hidegindítás. A szótármutató állításával a FORTH előredefiniált szókészletén kívül mindent töröl a szótárból és az ABORT művelet meghívásával újr aindítja a FORTH-ot.

COM: ( AH AL DX – AL AH )

Az INT 14H-t hajtja végre a veremben megadott regiszter értékekkel. Csak 16 bites FORTH esetén!

COMPILE

Definiáló szó belsejében használható. COMPILE-t tartalmazó szó végrehajtásakor az őt követő szó cfa-ja befordítódik a szótárba.

COMTYPE

COM kiegészítést tesz az FCB-ben levő fájl nevébe.

CON–PR ( – cim )

Változó, amely az aktuális output eszközt mutatja: 0-konzol, 1-printer.

CONSOLE

Az aktuális output eszköz a képernyő lesz (lásd még PRINTER).

CONSTANT ( n – )

Használata: n CONSTANT cccc

Definiáló szó, amely létrehoz egy cccc szót, amelynek paramétermezője n-et tartalmaz. Később, a cccc szó futásakor visszaadja a veremben az n értéket.

CONTEXT ( – cim )

Felhasználói változó, amely annak a szótárnak a címét tartalmazza, amelyben az interpreter először keresi a szavakat.

COUNT ( cim – cim+1 u)

Visszaadja a verem tetején a cim-en kezdődő hossz, karakter, . . . , karakter típusú karaktersorozat hosszát (u) (ez az első byte tartalma), és a tényleges karaktersorozat első karakterének címét. A COUNT-ból visszakapott paraméterek alkalmasak az adott szöveg TYPE-pal való kiírására.

CR * c-r- *

Végrehajt egy kocsivissza és egy új sor műveletet az aktuális output eszközön. Az OUT változó értékét 0-ra állítja.

CRT: ( AH – AX )

Az INT 16H-t hajtja végre a veremben megadott regiszterértékkel. Csak 16 bites FORTH esetén!

CREATE

Használata: CREATE cccc

Definiáló szó, amely a cccc szó szótári fejrészéből az NFA-t és az LFA-t hozza létre. Az új szó a CURRENT szótárban jön létre. Ahhoz, hogy ezután elérhető legyen, meg kell hívni a SMUDGE szót is.

CSP ( – cim ) * c-s-p *

Felhasználói változó, a veremmutató értékét őrzi a fordítási ellenőrzésekhez.

CURRENT ( – cim )

Felhasználói változó. Tartalma annak a szótárnak a címe, amelyben az újonnan definiált szavak kerülnek (lásd még: DEFINITIONS).

D+ ( d1 d2 – d3 ) * d-plus *

Visszaadja a verem felső 32 bitjén a két duplahosszú szám (d1, d2) összegét.

D+– ( d1 n – d2 ) * d-plus-minus *

Ha n negatív, akkor d2=-d1, egyébként d2=d1, tehát ez a művelet mindig d1 számértéket adja vissza a verem felső 32 bitjén, n előjelétől függő előjellel.

D. ( d – ) * d-dot *

Az aktuális számrendszernek megfelelően kiírja az előjeles duplahosszú d számot. A verem tetején a duplahosszú szám felső 16 bitje van. A szám kiírását egy szóköz követi.

D.R ( d n – ) * d-dot-r *

A verem második elemében levő előjeles duplahosszú d számot jobbra igazítva, a verem tetején megadott n széles mezőben írja ki.

D( ( d1 d2 – f )

Összehasonlítja a verem tetején levő 2 duplaszavas értéket. Ha d1 kisebb, mint d2, akkor a verem tetején (f) igaz logikai értéket ad vissza, egyébként hamisat.

D= ( d1 d2 – f )

Összehasonlítja a verem tetején álló 2 duplaszavas értéket (d1, d2). Egyezés esetén igaz logikai értéket ad vissza, egyébként hamisat.

D) ( d1 d2 – f )

Összehasonlítja a verem tetején álló 2 duplaszavas értéket (d1, d2). Ha d1 nagyobb, mint d2, akkor a verem tetején (f) igaz logikai értéket ad vissza, egyébként hamisat.

DABS ( d1 – d2 )

Visszaadja a verem tetején levő duplaszám (d1) abszolút értékét (d2).

DATUMOK ( – cim )

4 szó hosszúságú változó. A FORTH egyes részei módosításának utolsó dátumát tartalmazza 1–1 szám karakteres év, hó alakban. A FORTH négy része sorrendben: mag, DAMAN, pakolt decimális lebegőpontos aritmetika, szinkron vonali algoritmus. A megfelelő dátum nulla, ha a rész nincs lefordítva.

DECIMAL

A BASE változó értékét 10-re állítja, az aktuális számrendszer a tízes lesz.

DEFAULT-FILE ( – cim )

Visszaadja a verem tetején annak a változónak a címét, amely az alapértelmezés szerinti fájlnevet tartalmazza.

DEFINITIONS

Használata: cccc DEFINITIONS

Beállítja a CURRENT szótárat a CONTEXT szótárral megegyezőre úgy, hogy a következő definíciók ebbe a szótárba kerüljenek. A cccc szó beírásakor a cccc lesz a CONTEXT szótár. Ha utána DEFINITIONS következik, akkor a CURRENT szótár is a cccc lesz.

DEPTH ( – n )

Visszaadja a verem tetején, hogy ezen a visszaadott számon (n) kívül hány 16 bites érték van a veremben. Pl.:

```
9 8 DEPTH . . . 2 8 9 OK
```

DIGIT ( c n1 – n2 tf)
( c n1 – ff )

A verem második elemében levő ASCII karaktert a verem tetején levő n1 számrendszernek megfelelően konvertálja, és az ennek megfelelő bináris számot a verem második elemében, a verem tetején pedig igaz logikai értéket ad vissza. Ha a konverzió nem sikerült, csak egy hamis logikai értéket tesz a verembe (ff).

DIS

Használata: DIS cccc
Másodlagos kulcsszó törzsét visszafejti és kiírja az aktuális output eszközre.

DISK-ERROR ( – cim )

A diszkkezelés munkaváltozója, az utoljára írt vagy olvasott szektorra vonatkozó információt tartalmazza a verem tetején visszaadott címen. 0, ha nem volt hiba.

DL

A képernyő aktuális sorát kitörli, az alatta levő sorokat eggyel feljebb hozza, a legalsó sorba felhoz egy üres sort.

DLITERAL *d-literal*

( d – ; fordításkor )
( – d ; végrehajtáskor)

Fordításkor a verem tetején levő 32 bites értéket literálnak fordítja. Később annak az algoritmusnak a futásakor, amely a DLITERAL-t tartalmazza, visszaadja ugyanezt a 32 bites értéket a verem tetején.

DMINUS ( d1 – d2 )

Visszaadja a verem tetején levő duplahosszú szám (d1) kettes komplemensét (d2), (d2= -d1).

DO ( – cim n ; fordításkor )
( n1 n2 – ; végrehajtáskor )

Használata egy kettőspont-definíció belsejében:

DO . . . LOOP vagy
DO . . . +LOOP

Futásidőben a DO egy műveletsorozat ciklusban történő végrehajtását kezdi el. A ciklusváltozó (I) kezdőértéke a verem tetején van (n2), a ciklushatár a verem második eleme (n1). Ezeket a DO végrehajtáskor átteszi a visszatérési verembe. Amikor a műveletsorozat a LOOP-hoz ér, a ciklusváltozó értéke eggyel megnő. Ha az új érték kisebb a ciklushatárnál, a ciklus folytatódik tovább, ha nem, akkor a paraméterek törlődnek a visszatérési verem tetejéről és a végrehajtás a LOOP utáni szón folytatódik tovább. Mivel az ellenőrzés a ciklus végén történik, a ciklus egyszer mindenképpen végrehajtódik. (Lásd még: I, LOOP, +LOOP, LEAVE.)
Fordításkor a (DO) futásidejű eljárásra fordul le, a verem második elemében levő cím a ciklusmag eleje (a DO utáni szó címe), a verem tetején levő szám (n) a hibakezeléshez kell.

DOES⟩

Definiáló szó definíciójában szereplő szó. Használata:

```
: xxxx ⟨BUILDS yyyy
       DOES⟩ zzzz ;
```

Lásd a ⟨BUILDS-nél.

DOS+ ( DX CX BX AX – DX CX BX AX )

Az INT 21H-t hajtja végre a veremben megadott regiszter értékekkel. Csak 16 bites FORTH esetén!

DP ( – cim ) * d-p *

Felhasználói változó, a szótár szabadhely-mutatója. Tartalmát a HERE szó olvassa, az ALLOT pedig állítja.

DPL ( – cim ) * d-p-l *

Felhasználói változó, a duplahosszú input-szám tizedesjegyeinek számát tartalmazza. Használható az output tizedesjegyei számának a tárolására is. 16 bites input szám esetén értéke az alapértelmezés szerint –1.

DR:FN

Leválasztja a következő szót (egy fájlnevet) a terminál input pufferben és beteszi a screen-fcb-be kiegészítés nélkül.

DROP ( n – )

Törli a verem tetején levő elemet a veremből.

DUMP ( cim n – )

A címtől kezdődően n byte-ot dump-ol. A FORTHSCR.SCR fájlban található.

DUP ( n – n n )

Megduplázza a verem tetején levő elemet (lásd még: 2DUP).

DVARIABLE ( d – ; fordításkor)
( – cim ; végrehajtásnál)

Definíciós szó. Használata: d DVARIABLE cccc

Definiálja a cccc nevű duplaszavas változót. Az ezt követő cccc hivatkozás a verembe teszi a duplaszavas változó címét.

ELSE ( cim1 n1 – cim2 n2 ; fordításkor )

Használata egy kettőspont-definíció belsejében:

IF . . . ELSE . . . ENDIF

Futásidőben az ELSE-re az IF igaz ága után kerül a vezérlés, hatására a végrehajtás az ENDIF utáni szón folytatódik. A verem tartalmát nem változtatja.
Fordításkor a futásidejű BRANCH-re fordul le, helyet hagyva az elágazási címnek. Ezt a címet adja vissza a verem második elemében (cim2).
Az ELSE ezen kívül a verem második elemében megkapott címre (cim1) beírja az őt követő szó címét.
A verem tetején visszakapott n2 érték a hibaellenőrzéshez kell.

EMIT ( c – )

A verem tetején levő ASCII karaktert kiírja az aktuális output eszközre. Az OUT változó értékét eggyel megnöveli.

EMPTY-BUFFERS

Törli az összes blokk-puffer tartalmát anélkül, hogy kiírná a diszkre az UPDATE-tel megjelölt blokkokat. Új diszk behelyezésekor is tanácsos kiadni, nehogy tartalma a régi tartalommal összekeveredjen.

ENCLOSE ( cim1 c – cim1 n1 n2 n3 )

A FORTH interpreter belső eljárása, amelyet a WORD szó is használ. A verem tetején megkapott karakter (c) a határolójel. A veremben visszakapott értékek közül n1 a cim1-en kezdődő szöveg első nem-határolójel karakterének, n2 a szöveg első határolójelének, n3 pedig a szöveghez nem tartozó első karakternek a cim1-től számított relatív címe. A bináris 0 mindig határolójelnek számít.

END

Az UNTIL szinonímája (lásd ott).

ENDCASE ( cim1 n1 . . . cimN nN n – ; fordításnál )
( n – ; futásnál )

Fordításnál a cim1, . . . , cimN-ekre eltárolja a szabadhelymutató akutális értékét (az ENDOF ide, az ENDCASE-re ugrik).
Az n1, . . . , nN, n a hibaellenőrzéshez kell.
Futásnál törli a verem legfelső elemét (lásd még CASE).

ENDIF ( cim n – ; fordításkor)

Használata egy kettőspont-definíció belsejében:

IF . . . ENDIF

IF . . . ELSE . . . ENDIF

Futáskor az ENDIF csupán az IF-től vagy az ELSE-től induló ugrás célpontja. Az ENDIF másik neve: THEN. Fordításkor a szabadhelymutató aktuális értékét elteszi a cim-re. A verem tetején levő n a hibakezeléshez szükséges.

ENDOF ( n1 – cim n2 n3 ; fordításnál)

Fordításnál beépíti a BRANCH kulcsszót. n1, n2, n3 a hibaellenőrzéshez kellenek.

A cím pedig a BRANCH utáni szó címe, ahova az ENDCASE az ugrási hely címét beépíti. Végrehajtásnál ugrás történik az ENDCASE-re (lásd még CASE).

ERASE ( cim u – )

A verem második elemében levő cim-től kezdve u db byte-ot kinulláz.

ERROR ( n – )

Hibajelzés és a rendszer újraindítása. Megvizsgálja a WARNING változót. Ha annak tartalma 1, akkor a képernyőfájl 4. képernyőjének elejétől számított n-dik sort kiírja. Ha a WARNING változó 0, akkor a hibaüzenet száma (n) jelenik meg. A vezérlést a SHOW-ERROR kapja meg. Egyéb esetben (WARNING=—1) az (ABORT) szó végrehajtására kerül sor, ami meghívja az ABORT-ot.

EXECUTE ( cim – )

Végrehajtja azt a szót, amelynek cfa-ja a verem tetején levő cím. A cfa-t fordítási címnek is nevezik.

EXIT

A visszatérési verem tetején levő értéket eldobja. Így annak a kulcsszónak a végrehajtása után, amelyben az EXIT szerepel, a vezérlés egy szinttel fentebbre adódik.

EXPECT ( cim n – )

A billentyűzetről beolvas CR-ig, de legfeljebb n darab karaktert és ezeket a verem 2. elemében levő címre írja. A szöveg végére egy vagy több nulla kerül. A puffert hagyjuk nagyobbra!

FDOS ( c1 p – c2 )

E művelet segítségével lehet a monitor szolgáltatásait a FORTH-ból meghívni. A verem 2. elemében van a műveleti kód (c1), a verem tetején az input paraméter (p). A művelet végrehajtása után az akkumulátorban visszakapott kód van a verem tetején (c2).

FENCE ( – cim )

Felhasználói változó, amely azt a címet tartalmazza, amely alatt levő szavakat nem lehet FORGET-tel törölni.

FILL ( cim u b – )

A verem harmadik elemében levő címtől kezdődő u db byte-ot feltölti a verem tetején levő byte-tal.

FIRST ( – cim )

Konstans, amely az első blokkpuffer címét adja vissza.

FLD ( – cim )

Felhasználói változó, az output mező szélességének meghatározásánál használjuk.

FLUSH

Kiírja a diszkre az összes UPDATE-tel megjelölt blokkot. Ez a funkció automatikusan végrehajtódik, ha képernyőfájlokat cserélünk vagy ha kilépünk az operációs rendszerbe a BYE paranccsal.

FORGET

Használata: FORGET xxxx

Törli a szótárból az xxxx és az utána definiált összes szót. Ha e művelet meghívásakor a CURRENT és a CONTEXT szótár nem egyezik meg, hibajelzést kapunk.

FORTH

Az alapszótár neve. A szó meghívása után a FORTH nevű szótár lesz a CONTEXT szótár. További felhasználói szótárak definiálásáig az új definíciók a FORTH szótárba kerülnek. A FORTH közvetlen szó, így fordítási időben is rögtön végrehajtódik. A definícióban utána álló kulcsszavakat a FORTH szótárban fogja keresni a fordító.

GOTOXY ( y x – )

A képernyő x. sorának y. oszlopába pozicionál.

HERE ( – cim )

Visszaadja a verem tetején a szótár következő szabad helyének címét.

HEX

A BASE változóba 16-ot tesz. Az aktuális számrendszer a hexadecimális lesz.

HLD ( – cim ) * h-l-d *

Felhasználói változó, amely az output számkonverzió utolsó számjegyének a címét tartalmazza.

HOLD ( c – )

Számkonverzió közben a ⟨# és #⟩ műveletek között lehet használni. Az output szöveg aktuális pozíciójába beírja a verem tetején levő kraktert. Például egy számban a tizedesjegyek elé szúrható be vele (2E HOLD) egy tizedespont.

I ( – n )

A DO . . . LOOP, ill. DO . . . +LOOP ciklus belsejében használható, mint ciklusváltozó. A ciklusváltozó értékét a verem tetejére teszi (lásd J).

ID. ( cim – ) * i-d-dot *

A verem tetején levő nfa-ból kiírja a hozzátartozó definíció nevét.

IF ( f – ; futáskor )
( – cim n ; fordításkor)

Használata: IF . . . ENDIF
IF . . . ELSE . . ENDIF

Futáskor az IF a verem tetején levő f logikai érték szerint kiválasztja, hogy a végrehajtás melyik ágon folytatódjon. Ha f logikai értéke igaz, a vezérlés az IF utáni szóra kerül. Ha f hamis, a vezérlés az ELSE, ill. annak hiányában az ENDIF utáni szóra kerül. Mind az IF, mind az ELSE ág végrehajtása után a vezérlés az ENDIF utáni szóra adódik. Fordításkor a OBRANCH futásidejű eljárást fordítja be és helyet foglal az elágazási cím számára. A verem tetején visszakapott n számot és az alatta levő cim-et a megfelelő ELSE vagy ENDIF fordításkori hibaellenőrzésre, ill. az elágazási cím elhelyezésére használja fel.

IL

A képernyő aktuális sorát és az attól lentebb levőket egy sorral lentebb rollozza. Az aktuális sor egy üres sor lesz.

IMMEDIATE

Megjelöli az utoljára definiált szót oly módon, hogy amikor fordításkor ez a szó meghívódik, akkor nem a fordítására, hanem közvetlen végrehajtására kerül sor. Az IMMEDIATE-tel definiált szó fordítását a [COMPILE] szó előírásával lehet elérni.

IN ( – cim )

Felhasználói változó, az input szövegpuffer elejétől számított relatív címet tartalmazza, ahonnan a következő szöveg olvasása kezdődik. Többek között a WORD szó használja és állítja.

INDEX ( n1 n2 – )

Kiírja az n1-től az n2-ig az összes képernyő első sorát. Célszerű – megállapodás szerint – minden képernyő első sorába a képernyő azonosítását szolgáló megjegyzés-sort írni.

INTERPRET

A külső interpreter meghívása, a terminálról vagy diszkről jövő input szöveget a STATE változó tartalmától függően végrehajtja vagy lefordítja. Az egyes szavakat előbb a CONTEXT, majd a CURRENT szótárban keresi, ha egyikben sem találja, akkor az aktuális számrendszernek megfelelően bináris számmá konvertálja. Ha ez sem sikerül, akkor hibaüzenetet kapunk, visszaírja a szót és egy kérdőjelet. A szöveg olvasása a WORD ill. a NUMBER szavaknál leírt módon történik.

J ( – n )

Egymásba skatulyázott DO . LOOP, ill. DO . . . +LOOP szerkezet ciklusváltozója.
Használata:

DO . . . DO . . . J . . . LOOP . . . LOOP

A külső ciklus ciklusváltozójának értékét a verem tetejére teszi.

KEY ( – c )

A billentyűzeten beírásra vár, majd visszaadja a meghívása után leütött első billentyű ASCII kódját.

LATEST ( – cim )

Visszaadja a CURRENT szótár legfelső elemének nfa-ját.

LEAVE ( – )

A DO . . . LOOP, ill. DO . . . +LOOP ciklus belsejében használható. A ciklushatárt egyenlővé teszi a ciklusváltozó aktuális értékével, tehát a legközelebbi LOOP-nál, ill. +LOOP-nál a ciklus normálisan befejeződik.

LFA ( cim1 – cim2 ) * l-f-a *

A verem tetején kapott pfa-ból (cim1) kiszámítja a megfelelő szó lfa-ját.

LIMIT ( – n )

Konstans, a diszkpufferként felhasználható legnagyobb memóriacím utáni cím. Általában ez a legnagyobb memóriacím a rendszerben.

LIST ( n – )

Kilistázza a képernyő-fájl n-dik képernyőjét az aktuális output eszközre. Az SCR változóba beteszi az n értéket. Szükséges, hogy az aktuális számrendszer a DECIMAL legyen.

LIT ( – n )

Kettőspont-definíció belsejében minden 16 bites szám elé befordítódik a LIT futásidejű eljárás. Végrehajtáskor visszaadja a 16 bites értéket a verem tetején.

LIT"

Fordítás közben használható a következő formában: LIT" szöveg".
Befordítja a (LIT") futásidejű eljárást és mögé teszi a szöveg hosszát 1 byte-on, majd a szöveget. A szabadhely mutatót megnöveli a megfelelő értékkel.

LITERAL (n – ; fordításkor )

Fordításkor a verem tetején levő értéket (n) 16 bites literálként

lefordítja. Ez közvetlen definíció, azaz kettőspont-definíció belsejében azonnal lefut. Javasolt használata:

: xxxx . . . [számítás] LITERAL . . . ;

A fordítás a számítás elvégzésének idejére abbamarad, a számítás eredménye a verem tetejére kerül, a LITERAL pedig befordítja a keletkező szóba (lásd [ és ]).

LOAD ( n – )

Elkezdi az n-dik képernyő interpretálását, amely ;S-ig, –)-ig vagy a képernyő végéig tart. (A –) szó után az interpretálás a következő képernyővel folytatódik, lásd még –) és ;S). A 0. képernyőt nem lehet betölteni. Oda a képernyőfájlra vonatkozó információkat szokás írni.

LOOP ( cim n – ; fordításkor )
( – ; végrehajtáskor)

Használata kettőspont-definíció belsejében:

DO . . . LOOP

Futáskor megnöveli eggyel a ciklusváltozót és összehasonlítja a ciklushatárral. Amíg a ciklusváltozó kisebb, mint a ciklushatár, a vezérlés visszakerül a DO szóhoz; amikor a ciklusváltozó eléri a ciklushatárt, a végrehajtás a LOOP után folytatódik tovább.
Fordításkor befordítja a (LOOP) futásidejű eljárást. A verem 2. elemében levő cim-et berakja a (LOOP) után. Ez a (DO)-ra való visszaugráshoz szükséges. A verem tetején levő szám (n) a hibaellenőrzéshez szükséges.

LPT: ( AH AL DX – AL AH )

Az INT 17H-t hajtja végre a veremben megadott regiszter értékekkel. Csak 16 bites FORTH esetén!

M* ( n1 n2 – d ) * m-times *

Vegyes hosszúságú aritmetikai művelet. A verem tetején levő két 16 bites számot (n1, n2) összeszorozza és az eredményt 32 biten adja vissza (d).

M/ ( d n1 – n2 n3 ) * m-slash *

Vegyes hosszúságú aritmetikai művelet. Előjeles osztás, ahol az osztandó (d) duplahosszú (32 bites), az osztó 16 bites (n1), a verem 2. elemében visszaadott maradék (n2) és a verem tetején visszaadott hányados (n3) 16 bitesek. A maradék előjele megegegyezik az osztandóéval.

M/MOD ( ud1 u2 – u3 ud4 )

Vegyes hosszúságú aritmetikai művelet. Előjel nélküli osztás, ahol az osztandó (ud1) 32 bites, az osztó 16 bites (u2), a verem 2. elemében visszaadott maradék (u3) 16 bites és a verem tetején visszaadott hányados (ud4) 32 bites.

MATCH ( cim1 u1 cim2 u2 – f u3 )

Szövegrész mintaillesztéses keresése, az editor program használja. A veremben megadott paraméterek közül cim1 az aktuális pozíció címe, u1 a szövegterületen levő byte-ok száma, cim2 a megadott minta címe, u2 a minta hossza. A verem 2. elemében visszakapott logikai érték jelzi, hogy talált-e a mintához illeszkedő szövegrészt. A verem tetején visszakapott u3 a megtalált szövegig, ill. ha nem volt, akkor a szövegterület végéig a relatív byte-számot mutatja a szöveg (cim1) elejéhez képest.

MAX ( n1 n2 – n3 )

Visszaadja a verem tetején levő két érték (n1, n2) közül a nagyobbat (n3).

MEMORY ( – cim )

A SAVE által használt változó. A címe kerül a verembe.

MESSAGE ( n – )

Kiírja az aktuális output eszközre a 4. képernyő elejétől számított n. sort. A verem tetején megadott szám (n) lehet pozitív és negatív is. A MESSAGE kísérő információk kiíratására szolgál. Ha a WARNING változó értéke 0, akkor az üzenet egy szám lesz.

MIN ( n1 n2 – n3 )

Visszaadja a verem tetején levő két érték (n1, n2) közül a kisebbet (n3).

MINUS ( n1 – n2 )

Visszaadja a verem tetején levő szám kettes komplemensét, azaz n2=–n1.

MOD ( n1 n2 – n3 )

Visszaadja a verem tetején a két előjeles számon elvégzett osztás (n1/n2) maradékát (n3). A maradék előjele megegyezik az osztandóéval.

MOVE ( cim1 cim2 n – )

n db szót másol át cim1-ről cim2-re. A kisebb címek felől kezdi a mozgatást. Csak 16 bites FORTH esetén!

NEXT

A FORTH belső interpretere. Minden kódművelet visszatérési rutinja.

NEXT-LINK ( – cim )

Visszaadja a verem tetején a belső interpreter címét. CODE definíció végén levő ugrás létrehozásához használja az assembler.

NFA ( cim1 – cim2 )

Egy definíció pfa-jából (cim1) kiszámítja annak nfa-ját (cim2).

NOOP

FORTH szó, amelynek hatására semmi sem történik.

NUMBER ( cim – d )

A verem tetején levő címen talált szöveget az aktuális számrendszernek megfelelően előjeles duplaszavas számmá (d) alakítja át. Ha a szövegben volt tizedespont, akkor az utolsónak a címtől számított helyét beteszi a DPL változóba (a tizedespontnak más hatása nincs). Ha a számkonverzió nem sikerül, hibajelzést kapunk.

OCTAL

Az aktuális számrendszernek a 8-ast jelöli ki. A FORTHSCR.SCR fájlban található.

OF ( n1 – n2 ; fordításnál )
( n1 n2 – n1 ; futásnál)

Fordításnál beépíti a (OF)-ot. n1, n2 a hibaellenőrzés miatt kell. Futásnál n1-et és n2-t hasonlítja. Egyezés esetén az OF utáni utasítássorozatra kerül a vezérlés, egyébként az ezen OF-hoz tartozó ENDOF utáni szóra. Az OF a kezdeti paramétert (n1-et), amit a CASE előtt adtunk meg, a többi OF számára a veremben hagyja (lásd CASE).

OR ( n1 n2 – n3)

A verem tetején levő két 16 bites elemen (n1, n2) végzett bitenkénti logikai VAGY művelet eredményét adja vissza (n3).

OUT ( – cim )

Felhasználói változó, amelynek tartalmát az EMIT növeli. A felhasználó is változtathatja értékét, ez a képernyő alakítását segítheti. A CR 0-ra állítja.

OVER (n1 n2 – n1 n2 n1 )

A verem 2. elemét a verem tetejére másolja úgy, hogy az eredeti elemek megmaradnak.

P! ( n1 n2 – )

Írás az I8088 n1-dik portjára. Csak 16 bites FORTH esetén!

P@ ( n1 – n2 )

Olvasás az I8088 n1-dik portjáról. Csak 16 bites FORTH esetén!

PAD ( – cim )

A HERE-től állandó távolságra levő címet ad vissza a verem tetején. Ez a cím a szöveg-output-puffer címe.

PC! ( b n – )

Írás az I8080 n-dik portjára.

PC@ ( n – b )

Olvasás az I8080 n-dik portjáról.

PCKEY ( – c 0 vagy c )

Karakterre várakozik a klaviatúráról. Visszaadja az első leütött karakter kódját. Csak 16 bites FORTH esetén!

PFA ( cim1 – cim2)

Egy definíció nfa-jából (cím1) kiszámítja annak pfa-ját (cím2)

PICK ( n1 – n2 )

A verem felülről számított n1-dik elemét (n2) a verem tetejére másolja úgy, hogy az eredeti elemek megmaradnak (pl. a 2 PICK szósorozat ekvivalens az OVER szóval, lásd OVER).

PREV ( – cím )

Felhasználói változó, annak a diszkpuffernek a címét tartalmazza, amelyikre utoljára hivatkoztunk. Ezt a diszkpuffert lehet az UPDATE szóval megjelölni, hogy később ki kell írni a diszkre.

PREV-FILE ( – cím )

A USING által használt változó. Új fájl kijelölésekor a régi fájl nevét ide menti ki ideiglenesen a USING, hogyha az új fájlt valami miatt nem lehet megnyitni, a régi fájl maradhasson az aktuális. A verem tetején a változó címe van.

PRINTER

Az aktuális output eszköz a printer lesz (lásd CONSOLE).

QUERY

Beolvas 80 karaktert (vagy "return"-ig) a terminálról. A szöveget a terminál input pufferbe teszi, ennek címe a TIB változóban van. Az IN változót 0-ra állítja.

QUIT

Törli a visszatérési vermet és a vermet, leállítja a fordítást és a vezérlést visszaadja a terminálra. Nem ad hibaüzenetet.

R ( – n )

A visszatérési verem tetején levő számot átmásolja a verem tetejére. (A szám ott marad a visszatérési verem tetején is.)

R/W ( cím n f – )

A FIG-FORTH szabványos diszkre író – diszkről olvasó eljárása. A verem 3. elemében levő cím a forrás, ill. cél blokkpuffer címe, az n a kiírni, ill. beolvasni kívánt blokk sorszáma. A verem tetején levő logikai érték (f) mutatja, hogy írásról (f=0) vagy olvasásról (f=1) van szó. Az R/W megkeresi a kijelölt blokkot a diszken, végrehajtja a kijelölt műveletet és hibaellenőrzést végez.

R) ( – n ) * from-r *

A visszatérési verem tetején levő elemet a verem tetejére teszi és törli a visszatérési veremből (lásd még: R és )R).

RØ ( – cím ) * R-zero *

Felhasználói változó, amely a visszatérési veremmutató kezdő értékét tartalmazza (lásd még: RP!).

REC ( – cím )

A diszkkezelés munkaváltozója.

REPEAT ( cím n – ; fordításkor)
( – ; végrehajtáskor)

Használata egy kettőspont-definíción belül:

BEGIN . . . WHILE . . . REPEAT

Futásidőbben a REPEAT feltétel nélküli ugrást jelent a megfelelő BEGIN-t követő szóra.

Fordításkor a BRANCH futásidejű eljárást és a visszaugrás címét fordítja be. A verem tetején levő szám (n) a hibaellenőrzéshez kell.

RMOVE ( cím1 cím2 n – )

Cím1-ről mozgat cím2-re n hosszban byte-onként úgy, hogy a mozgatást a szöveg végéről kezdi.

ROLL ( n1 – n2 )

A verem n1-dik elemét (n2) kiveszi a helyéről és a verem tetejére teszi. (3 ROLL = ROT lásd ott.)

ROT ( n1 n2 n3 – n2 n3 n1 )

Megforgatja a verem felső három elemét oly módon, hogy az eddigi 3. elem (n1) kerül a verem tetejére.

RP!

Inicializálja a visszatérési verem mutatóját az RO felhasználói változóban levő értékkel.

RP@ ( – cím )

Visszaadja a visszatérési verem mutatójában levő aktuális értéket.

S ) D ( n – d )

A verem tetején levő 16 bites előjeles értéket (n) 32 bitessé alakítja.

S0 ( – cím ) * s-zero *

Felhasználói változó, amely a veremmutató kezdőértékét tartalmazza (lásd még SP!).

S=  ( cím1 cím2 n – f )

Összehasonlítja a cím1-en és cím2-n kezdődő szövegeket n byte hosszúságban. A verem tetején visszakapott logikai érték (f) igaz, ha a két szöveg egyenlő.

SAVE

Használata: SAVE fájlnév.
Kiírja a megadott fájlba a 0 +ORIGIN által visszaadott címtől kezdődően a szabadhely mutatóig tartó memóriaszakasz tartalmát.
Mentés előtt aktualizálni kell néhány hidegindítási értéket:

```
DECIMAL                  LATEST    12   +ORIGIN !
HERE 28 +ORIGIN !        HERE      30   +ORIGIN !
VOC-LINK @ 32 + ORIGIN !
```

SCR  ( – cím )

Felhasználói változó, annak a képernyőnek a számát tartalmazza, amelyre utoljára hivatkoztak a LIST szóval.

SCREEN-FCB  ( – cím )

Ez a képernyőfájlok FCB-je. Az FCB címe kerül a verem tetejére.

SCRTYPE

‚SCR' kiegészítést tesz a screen-fcb-ben található file nevébe.

SEC-READ

A diszkkezelés által használt belső szó.

SEC-WRITE

A diszkkezelés által használt belső szó.

SEC/BLK ( – n )

Konstans, amely visszaadja, hogy hány szektorból áll egy FORTH blokk.

SET-IO

A diszkkezelés által használt belső szó.

SHIFT ( n1 n2 – n3 )

Az n1 előjeles, 16 bites szám bitjeit léptetí az n2-ben mutatott értékekkel. Ha n2 pozitív, akkor jobbra léptet, ha n2 negatív, akkor balra. n3 a léptetések után keletkezett új előjeles érték.

SHOW-ERROR

Az ERROR hívja. Hibaüzenetet ír ki a képernyőre. Ha képernyőfájlból folyik a fordítás, akkor megjelenik a hibaészlelés helye is (hányadik képernyő hányadik sora). A vezérlés a QUIT-re kerül.

SIGN ( n d – d )

Ha a verem 2. eleme (n) negatív, akkor tesz egy mínuszjelet az output szövegbe a verem tetején levő szám (d) elé. Az n számot törli a veremből, a d-t meghagyja a verem tetején, a ⟨# és #⟩ szavak között lehet használni.

SMUDGE

Ellentettjére állít egy bitet, az ún. smudge-bitet egy definíció névmezőjében. Ha ez a bit egy szónál be van állítva, akkor az interpeter nem találja meg a szót, ez fordítás közben ellenőrzési lehetőség. A bitet a ; kulcsszó törli, ha a kulcsszódefiniálásban nem volt hiba.

SP@ ( – cím )

A veremmutató értékét adja vissza, ez a cím a verem tetején levő elem címe, az SP@ meghívása előtt. Pl.: az 1 2 ASP@ @ szósorozat és. . . után 2 2 1 OK választ kapunk.

SPI

Az SO felhasználásával kezdőértékezi a veremmutatót.

SPACE

Kiír egy szóközt az aktuális output eszközre.

SPACES ( n – )

Kiír n szóközt az aktuális output eszközre.

STATE ( – cím )

Felhasználói változó, tartalma egy egy-bites információ, hogy éppen interpretálás vagy fordítás folyik-e – ha nem 0, akkor fordítás.

SWAP ( n1 n2 – n2 n1 )

Megcseréli a verem tetején levő két elemet.

SZIN ( – cím )

Változó, mely a karatkeríráshoz szükséges háttérszint tartalmazza. Csak 16 bites FORTH esetén!

T" ( cím – )

Csak végrehajtási módban használható.
Formája: T" szöveg"
A címtől kezdődően eltárolja az " határoló karakterig tartó szöveget.

TASK

Üres eljárás, az alkalmazások elkülönítésére szolgál.

THEN

Az ENDIF szinonímája.

THRU ( n1 n2 – )

n1-től n2-ig betölti a LOAD paranccsal az aktuális képernyőfájlból a képernyőket.

TIB

Felhasználói változó, a terminál input puffer címét tartalmazza.

TOGGLE ( cím b – )

A verem 2. elemében levő címen található értéket a verem tetején levő b bitminta szerint komplementálja (kizáró VAGY műveletet végez).

TRAVERSE ( cím1 n – cím2 )

Átlépi egy adott szótárelem névmezőjét. Ha a verem tetején levő szám (n) 1, akkor a 2. elem (cím1) a név hosszúságbyte-jának címe, és a visszakapott cím (cím2) a név utolsó karaktere. Ha n=–1, akkor fordítva.

TRIAD ( n – )

Kilistáz 3 képernyőt úgy, hogy az n-dik közöttük van és az első osztható hárommal. Az aktuális számrendszernek a DECIMAL-nak kell lennie.

TYPE ( cím u – )

A verem második elemében levő címről kiír u db byte-ot az aktuális output eszközre. Az OUT változó értékét u-val megnöveli.

U* ( u1 u2 – ud )

Visszaadja a verem tetején 32 biten (ud) a verem két felső elemében levő (u1 és u2) előjel nélküli 16 bites szám előjel nélküli szorzatát.

U. ( u – )

Az aktuális számrendszernek megfelelően kiírja a verem tetején levő 16 bites előjel nélküli számot (u).

U/ ( ud u1 – u2 u3 )

Elosztja a verem 2. elemében levő előjel nélküli duplahosszú számot (ud) a verem tetején levő 16 bites számmal (u1) és visszaadja szintén előjel nélkül a verem tetején a hányadost (u3) és alatta a maradékot (u2).

U< ( u1 u2 – f )

Összehasonlítja a verem tetején levő két előjel nélküli 16 bites számot (u1 és u2). Ha a verem 2. eleme (u1) a kisebb, akkor a verem tetején visszakapott logikai érték (f) igaz. Memóriacímek összehasonlítását csak ezzel a művelettel lehet elvégezni.

UNTIL ( f – ; futáskor )
( cím n – ; fordításkor )

Használata egy kettőspont-definíció belsejében:

BEGIN . . . UNTIL

Futáskor a megkapott logikai értéktől függően a megfelelő BEGIN

utáni szóra (f=hamis), ill. az őt követő szóra (f=igaz) adja át a vezérlést. Fordításkor a OBRANCH-et és a verem 2. elemében levő címet fordítja be.

A verem tetején levő érték (n) a hibaellenőrzéshez kell.

UPDATE

Egy bit beállításával megjelöli azt a blokkot, amelyre utoljára hivatkoztunk. Ezután, ha a blokkot tartalmazó pufferre szükség van, a megjelölt blokk automatikusan kiíródik a diszkre. A PREV változó tartalmazza, hogy melyik blokkra hivatkoztunk utoljára.

USE ( – cím )

Felhasználói változó, annak a blokkpuffernek a címét tartalmazza, amelyet legközelebb fel lehet használni; ebbe a pufferbe írtunk legrégebben.

USER ( n – )

Definiáló szó, használata:

n USER cccc

Egy cccc nevű felhasználói változót definiál. A cccc szó paraméter-mezője tartalmazza n-et, amely a felhasználói változó terület elejéhez képest számított relatív cím. Később, a cccc szó futásakor a felhasználói változó terület eleje címének és n-nek az összege kerül a verembe, mint a cccc nevű felhasználói változó címe.

USING

Használata: USING fájlnév
Kicseréli az aktuális képernyőfájlt a USING után megadottal.

VARIABLE

Definiáló szó, használata:

n VARIABLE cccc

Futáskor létrehoz egy cccc nevű változót, amelynek paraméter-mezőjébe bekerül az n érték. Ez a változó kezdőértéke. Később, a cccc szó futásakor a paramétermezőjének címe kerül a verem tetejére, ezt felhasználhatjuk a változó értékének kiolvasására vagy megváltoztatására.

VOC-LINK ( – cím)

Felhasználói változó, az utoljára létrehozott szótár definíciójában levő egyik mezőnek a címe. Az összes szótárnév ezen a mezőn keresztül van láncra fűzve.

VOCABULARY

Definiáló szó, használata:

VOCABULARY cccc

Ezzel egy cccc nevű szótárat definiálunk. A cccc szó használata a továbbiakban azt jelenti, hogy ez lesz a CONTEXT szótár, amelyben az interpreter először keres. A cccc DEFINITIONS szósorozat után a CURRENT szótár is a cccc lesz, így ebbe kerülnek az új definíciók is. A FIG-FORTH-ban a láncolás úgy van megoldva, hogy a cccc szótár tartalmazza mindazokat a szavakat, amelyek abban a szótárban benne voltak, amelyben cccc-t definiáltuk. Megállapodásszerűen a szótárneveket IMMEDIATE-nek deklaráljuk (lásd még: VOC-LINCK).

VLIST

Kilistázza a CONTEXT szótárban levő neveket. Tetszőleges billentyű leütése véget vet a listázásnak.

WARM

Melegindítás. Az EMPTY-BUFFERS és ABORT szavak meghívása, a szótárak tartalma nem változik (lásd még COLD).

WARNING ( – cím )

Felhasználói változó, tartalma az üzenetküldés vezérlésére szolgál. Ha WARNING=1, a képernyőfájl 4. képernyőjében levő üzenetek jelennek majd meg. Ha WARNING=0, csak az üzenet száma jelenik meg. Ha WARNING=1, az (ABORT) futásidejű eljárás hívódik meg (lásd még: MESSAGE, ERROR).

WHILE ( f – ; futáskor )
( cím1 n1 – cím1 n1 cím2 n2 ; fordításkor )

Használata egy kettőspont-definíció belsejében:

BEGIN . . . WHILE . . . REPEAT

Futáskor a veremben levő logikai értéktől függően a WHILE-tól a vezérlés a REPEAT utáni szóra kerül (f=hamis), ill. lefutnak a WHILE és REPEAT közötti szavak és a végrehajtás ezután a BEGIN utáni szón folytatódik (f=igaz).
Fordításkor a WHILE a OBRANCH-et fordítja be és helyet foglal az elágazási cím számára (cím2). A verem tetején visszakapott n2 szám a fordítási hibaellenőrzéshez kell.

WIDTH ( – cím )

Felhasználói változó, amely egy kulcsszó fordításakor nevének maximálisan figyelembe vett karaktereinek számát tartalmazza. Az alapértelmezés 31, de értéke 1 és 31 között változhat.

WORD ( c – )

Beolvassa az input szövegből a soronkövetkező karaktereket a c határolójelig. A beolvasott szöveg hosszát a HERE címre, a szöveget magát a HERE+1-től kezdve teszi le. A c karakter első értékes helyen

levő, azaz „vezető-c" karakterenkénti előfordulásai nem számítanak határolójelnek. Ha a BLK változó értéke 0, a szöveget a terminál input pufferből veszi, ha nem, akkor a BLK-ban kijelölt blokkból (lásd még: BLK, IN).

XOR ( n1 n2 – n3 )

A verem két felső 16 bites elemén (n1 és n2) végzett bitenkénti logikai kizáró VAGY művelet eredményét (n3) adja vissza.

[ ( – ) * bal szögletes zárójel *

Átlépés fordítási módból interpretálási módba. Kötelező a ] szóval együtt használni.

Használata szódefiníció belsejében:

: xxxx [szavak] más-szavak ;

A [ és ] közötti szavak fordítására nem kerül sor, fordításkor lefutnak (lásd még: LITERAL,]).

[COMPILE] ( – )

Használata egy kettőspont-definíció belsejében pl.:

: xxxx [COMPILE] FORTH ;

Hatására az IMMEDIATE szavak fordítására is sor kerül, ezek a szavak egyébként fordításkor futnának le. A fenti példában a FORTH szótár kiválasztására xxxx futása közben és nem definíciójának fordítása közben kerül sor.

] ( – ) *jobb szögletes zárójel *

Visszatérés interpretálási módból fordítási módba, a [ szóval együtt kell használni (lásd ]).

clearscreen

Törli a képernyőt, kurzort nem állít. Csak 16 bites FORTH esetén!

video-io ( DX CX BX AX – DX CX BX AX )

Az INT 10H-t hajtja végre a veremben megadott regiszter értékekkel. Csak 16 bites FORTH esetén!

# A FÜGGELÉK

## Kapcsolat az UPM operációs rendszerrel

Az FDOS FORTH szó segítségével a FORTH-ból meg lehet hívni az UP/M szolgáltatásait. Az FDOS szóval a FORTH-on belül kezünkben tarthatjuk az UP/M fájlokat, lehetőségünk nyílik fájlleíró blokkok és diszkpufferek definiálására.

FDOS ( c1 p – c2 )

ahol c1 az UP/M művelet kódja, p az input paraméter, c2 a végrehajtott művelet után az akkumulátorban visszakapott kód. Az FDOS meghívása előtt be kell tenni a verembe az UP/M művelet kódját és ezt követően az input paramétert, ez utóbbi kerül a DE regiszterbe. Az UP/M művelet végrehajtása után a verem tetején az akkumulátor tartalmát és egy 8 bites 0-t kapunk vissza. A következő táblázat az FDOS-nak megadható kódokat és jelentésüket adja meg:

| Műveleti kód | Paraméter | FDOS | Akkumulátor kód | Jelentése |
|---|---|---|---|---|
| 0 | * | | * | rendszer alapállapotba állítása |
| 1 | * | | karakter (L-A) | 1 karakter olvasása konzolról |
| 2 | karakter (E) | | * | 1 karakter írása konzolra |
| 3 | * | | karakter (L-A) | lyukkártyaolvasóról olvasás |
| 4 | karakter (E) | | * | lyukasztó output |

| Műveleti kód | Paraméter | FDOS | Akkumulátor kód | Jelentése |
|---|---|---|---|---|
| 5 | karakter (E) | | * | listázó eszközre írás |
| 6 | OFFH (E) | | karakter (L;A) | közvetlen input |
| 6 | karakter (E) | | * | közvetlen output |
| 7 | * | | byte (L;A) | ~~i/o byte lekérdezése~~ megjelenés, mint a 06-os funkció! |
| 8 | ~~byte~~ ( | | * (L;A) | ~~i/o byte beállítása~~ megjelenés, mint az 1-es funkció |
| 9 | cím (DE) | | * | szöveg nyomtatása adott címről |
| 10 | cím (DE) | | * | szöveg beolvasása adott címre |
| 11 | * | | státusz (L;A) | konzol státusz lekérdezése |
| 13 | * | | * | diszk alapállapotba állítása |
| 14 | diszk-azonosító (E) | | * (L;A) | diszk kiválasztása |
| 15 | fájlleíró blokkcím (DE) | | státusz (L;A) | fájl megnyitás |
| 16 | fájlleíró blokkcím (DE) | | státusz (L;A) | fájl lezárás |
| 17 | fájlleíró blokkcím (DE) | | státusz (L;A) | az első fájl megkeresése |

| Műveleti kód | Paraméter | FDOS | Akkumulátor kód | Jelentése |
|---|---|---|---|---|
| 18 | fájlleíró<br>blokkcím<br>(DE) | | státusz<br>(L;A) | a következő fájl megkeresése |
| 19 | fájlleíró<br>blokkcím<br>(DE) | | státusz<br>(L;A) | fájl törlése |
| 20 | fájlleíró<br>blokkcím<br>(DE) | | státusz<br>(L;A) | szekvenciális olvasás |
| 21 | fájlleíró<br>blokkcím<br>(DE) | | státusz<br>(L;A) | szekvenciális írás |
| 22 | fájlleíró<br>blokkcím<br>(DE) | | státusz<br>(L;A) | fájl létrehozás és megnyitás |
| 23 | fájlleíró<br>blokkcím<br>(DE) | | státusz<br>(L;A) | fájl átnevezés |
| 25 | * | | diszk-azonosító<br>(L=A) | aktuális diszk lekérdezése |
| 26 | cím<br>(DE) | | * | DMA cím beállítása |
| ~~28~~ | * | | * | ~~írástiltás a diszken~~ |
| ~~30~~ | fájlleíró<br>blokkcím | | státusz | ~~fájl attributum beállítás~~ |
| ~~32~~ | OFFH | | kód | ~~felhasználói kód lekérdezése~~ |

| | | | | |
|---|---|---|---|---|
| 32 | kód | | * | ~~felhasználói kód beállítása~~ minus! |
| 33 | fájlleíró blokkcím (DE) | | státusz (L=A) | random olvasás ✓ |
| 34 | fájlleíró blokkcím (DE) | | státusz (L=A) | random írás ✓ |
| 35 | fájlleíró blokkcím (DE) | | státusz (L=A) | fájlméret meghatározása ✓ |
| 36 | fájlleíró blokkcím (DE) | | státusz | random rekord beállítása ✓ |

A * jelölés az inputon ignorált értéket, az outputon érdektelen eredményt mutat. (Azokat is meg kell adni, csak figyelmen kívül maradnak!)

# B FÜGGELÉK

## A FORTH hibaüzenetei

BASE DECIMAL legyen #13

Bizonyos műveleteknél követelmény, hogy az aktuális számrendszer a decimális (a BASE változó tartalma 10) legyen. Ezzel elkerülhetjük, hogy a képernyő fájltól messzelevő képernyőket címezzünk meg.

csak fordítás során használd, definícióban #17

Egy fordító szót kettőspont-definíción kívül használtunk.

a direktívák nincsenek párban ~~#20~~ #19

A ? PAIRS szó által adott hibaüzenet, meghivásakor nem volt egyenlő a verem két felső eleme. Akkor kaphatjuk, ha pl. az IF-ELSE-ENDIF struktúrák hibásan vannak egymásba ágyazva. Pl.:

IF . . . IF ELSE ELSE ENDIF ENDIF

a definíció hiányos #20

Egy kettőspont-definícióban a ";" megjelenésekor valamelyik vezérlési szerkezet hiányos volt, azaz hiányzik az ENDIF, UNTIL, REPEAT vagy AGAIN szavak valamelyike.

a könyvtár tele van

Nincs több hely a kettőspont-definícióknak.

hibás diszkcím

A diszken elérhetetlen blokkot címeztünk meg.

diszk hiba!

Nem megengedett diszkművelet hívása, ez nem hardver hiba.

a verem üres #1 ✓

Olyan műveletet hívtunk meg, amely a veremből akart olvasni, a verem pedig üres volt. A ? STACK szó adja ki.

csak végrehajtás során használd

A meghívott szó nem szerepelhet kettőspont-definícióban.

illegális dimenzió a tömbdefinícióban

Tömbdeklaráláskor kiadott hibaüzenet: vagy lehetetlen (negatív) tömbdimenziót adtunk meg, vagy nincs elég memória a tömbnek.

védett könyvtárban van #15 ✓

A FENCE szó alatt levő definíciókra nem szabad a FORGET szót kiadni.

újradefiniálva #4 ✓

Egy kettőspont-definícióban megadott név már létzik. Az előzetes lefordított kód fölé íródik az új kód, csak az új definíció „elfelejtése" után lehet újra hozzáférni. Ez nem hiba, csak figyelmeztetés.

csak töltés során használd #22

Ilyen a –) szó, amelyet csak képernyők betöltésekor kaphat az interpreter.

nincs definiálva #5

Olyan szóra történt hivatkozás, amely sem a CONTEXT, sem a CURRENT szótárban nincs benne, és az aktuális számrendszerbeli számnak sem értelmezhető.

a verem tele van.

Ez a hibaüzenet akkor jelenik meg, ha a könyvtár teteje a verem aljával összeér.

VIDEOTON

ELEKTRONIKAI VÁLLALAT
SZÁMÍTÁSTECHNIKAI GYÁRA