NAGY ÁRPÁD MÁRKUS BÉLA

zFORTH HT-1080Z-re

programozói kézikönyv

Budapest, 1985

NAGY ÁRPÁD MÁRKUS BÉLA

zFORTH HT-1080Z-re

programozói kézikönyv

Budapest, 1985

Készült a Tudományszervezési és Informatikai Intézet közremüködésével a középfoku oktatási intézmények részére.

Sokszorositotta az Oktatáskutató Intézet sokszorosító-üzeme
1ooo példányban.

TARTALOMJEGYZÉK

BEVEZETÉS

A FORTH nyelvet az 1970-es évek elején alkotta CHARLES MOORE ameriaki programozó. Első alkalommal egy rádiótávcső vezérlésére használták fel. Az elmult években, főként a mikroszámitógépek tömeges elterjedésével népszerüsége rohamosan nő. Külön, kizárólag a FORTH felhasználásával foglalkozó csoportok alakultak, folyóiratok, könyvek jelentek meg.

Mi is a FORTH?

Magas szintü programozási nyelv, amely gyors, kevés helyigényü programok irását teszi lehetővé, nagyfoku szabadságot biztositva és időt, fáradságot takaritva meg a program készitőjének.

De ennél sokkal több is. Programkészitési eljárás és gondolkodási mód, egyben problémamegoldási stratégia. Használata a feladatok olyan megközelitését, elemzését tanitja meg, amely mindenkor hasznos lehet.

A zFORTH ennek a nyelvnek a Hiradástechnika Szövetkezet HT-1080Z tipusu mikroszámitógépére adaptált változata, mely alap szókészletét tekintve közel áll a FORTH '79 szabványhoz, de tartalmazza mindazokat a kiegészitéseket is, melyek speciálisan ennek a gépnek a felhasználását könnyitik meg.

1. A RENDSZER BETÖLTÉSE

A zFORTH programrendszert a BASIC SYSTEM parancsával tölthetjük be zFORTH néven. A beolvasás folyamatosságát a képernyő jobb felső sarkában villogó * karakter jelzi, melynek végén a program a verziószám kiirásával jelentkezik be. Ezután adhatunk parancsokat, utasitásokat. Ezek közül a legegyszerübb az üres utasitás, melynek elfogadását a NEWLINE billentyü leütése után megjelenő OK jelzi. Ennek hiányában a rendszer betöltését elölről kezdve meg kell ismételni, esetleg más lejátszási szint beállitása mellett.

A zFORTH a BASIC ROM bővités által biztositott szolgáltatásokat - kisbetük, villogó cursor és ismétlő billentyü - automatikusan elérhetővé teszi, azok élesitéséről gondoskodni külön nem kell.

1.1 Ujrainditás

Bizonyos alkalmakkor, például végtelen ciklusok vagy a vártnál tovább futó programok esetén szükség lehet azok végrehajtásának felfüggesztésére.

Ezt a gép hátulján elhelyezett RESET gomb megnyomásával tehetjük meg, melynek elfogadását a "zFORTH restart" üzenet jelzi. Ilyenkor visszaáll a tizes számrendszer használata, kiürül a veremtár és kikapcsolódik az esetleg használt magnetofon motorja is.

Előfordulhatnak olyan hibák is, amikor nem jelenik meg az emlitett szöveg. Ez esetben a hálózati kapcsoló ki- majd ujbóli bekapcsolása után ujra be kell tölteni a zFORTH-t.

2. AZ ELSŐ LÉPÉSEK

Hogyan tanulható meg a FORTH?

A legegyszerübben ugy, hogy leülünk a gép mellé és kipróbáljuk az egyes utasitásokat, mintapéldákat. A továbbiakban ehhez kivánunk segitséget adni. Ne feledjük, a legjobb tanitómester a gyakorlat!

A zFORTH a megadott utasitásokat soronként dolgozza fel. A tévesen beütött karakter törölhető a ← billentyüvel. A bevitt sor értelmezése a NEWLINE vagy pedig a 64-ik karakter beirása után kezdődik meg.

2.1 Szavak

A FORTH nyelvben a parancsokat, utasitásokat szavaknak hivjuk. Azonositásuk tetszőleges, megjelenithető karakterekből álló névvel történik, melyek belsejében nem fordulhat elő szóköz. Azonosnak tekinti a rendszer azokat a neveket, melyek hossza és első három karaktere egyezik, például:

STATE és STARS

A gép az egyes szavakat és a hozzájuk tartozó jelentés, müködés leirását a memória erre kijelölt részében, a szótárban (dictionary-ban) tartja nyilván.

Az input sor lezárása után az ugynevezett külső interpreter

a sor elejétől kezdve megkeresi a szavakat, melyeket legalább egy szóköz választ el egymástól, a könyvtárban és végrehajtja őket. Amennyiben nem találja meg, megnézi, számként értelmezhető-e. Ha igen, a számot elhelyezi a veremtárban, ellenkező esetben hibajelzést kapunk.

A sor sikeres interpretálását a gép "OK" kiirásával jelzi, ezután uj sort irhatunk be.

Amennyiben valamilyen hiba fordult elő, az interpretálás a hibás szónál abbamarad és a hiba okára utaló üzenet jelenik meg, mely tartalmazza - az ernyő közepén - a hibát előidéző szó nevét is.

Például beirva az XXX karaktersort.

* UNKNOWN WORD XXX

hibaüzenetet kapunk, mivel nem szerepel a szótárban és számként sem értelhezhető.

2.2 Számok

A számok tárolása 16 biten történik, egész formában. Ez azt jelenti, hogy tizes számrendszerben az ábrázolható legnagyobb szám 32767 mig a legkisebb -32768. Előjel nélkülinek tekintve a számot az ábrázolási tartomány Ø és 65535 közé esik.

A számok beirásánál a pozitiv előjelet nem kell megadni, a + jel hibát okoz.

Függetlenül a belső ábrázolástól, a beolvasás és kiiratás 2 és 72 között tetszőleges alapu számrendszerben történhet. Alapértelmezés a decimális, mely a DECIMAL szó végrehajtásával állitható vissza. Áttérhetünk a HEX szóval a hexadecimális

számrendszer használatára. A pillanatnyi érvényes alapszámot a BASE nevü rendszerváltozó tartalmazza, melynek értéke a korábbi feltételek betartásával megváltoztatható.

Hibás érték esetén

* BASE OUT OF RANGE

hibaüzenetet kapunk.

2.3 A veremtár

A veremtár, más néven paraméter stack, a memóriának adatok átmeneti tárolására szolgáló része. Push-down jellegü, vagyis közvetlenül csak a tetején elhelyezkedő, legutolsónak beirt szám férhető hozzá. Használata nagymértékben leegyszerüsiti a szavak között a paraméterek átadását. A legtöbb szó a veremtárra valamilyen hatást gyakorol, ezért az egyes szavak leirásánál megadjuk, mi történik a stack-ben.

Az összeadás, a + szó esetében például ez igy történik
(n1 n2 -- n)

Ez azt jelenti, hogy a + végrehajtása előtti n1 és n2 szám eltünt a tárból és helyettük a végeredmény, n került be, vagyis a stack-ben tárolt adatok száma eggyel csökkent. Ebben a leirási módban az utolsónak leirt szám, a legjobboldalibb helyezkedik el a veremtár tetején.

A továbbiakban n előjeles, u előjel nélküli számot jelöl, c pedig karaktert. Utóbbi olyan szám, melynek csak alsó nyolc bitje értékes, a felsők nem birnak jelentéssel.

Például leirva a következő sort:

3 8 +

a stack tartalma a következő (3 8 -- 11)

Természetesen csak azokat tüntettük fel, melyekre az adott szavak hatással vannak, a mélyebben lévő értékeket nem.

A veremtár mind kiürülés, mind megtelés ellen védett.

Üres stack esetén a rendszer

* STACK IS EMPTY

mig teli stack esetén

* STACK IS FULL

hibaüzenetet ad. Utóbbi esetben az addigi tartalom elvész.

2.4 A FORTH mint kalkulátor

Az előzőek hasznositására lássuk, hogyan végezhetünk el egyszerübb számitásokat.

A nyelv egyik érdekes, a veremtár használatából következő jellegzetessége a forditott jelölésmód, a post-fix notation, más néven RPN használata, mellyel a HP kalkulátoroknál is találkozhatunk.

Ez azt jelenti, hogy a müveletek operandusainak annak végrehajtása előtt már a stack-ben kell lennie. Két szám szorzatának a kiszámitása igy történik:

```
9 8 * . NEWLINE      72 OK
```

Aláhuzással azt a részt jelöltük, melyet válaszul a FORTH ir ki. Nagyon fontos uj szót ismertünk meg, a pontot, mely kiirja a veremtár tetején lévő előjeles számot. Azt a kifejezést, melyet BASIC-ben PRINT 3 * (8 + 5) + 9 alakban adunk meg, FORTH-ban igy számolhatunk ki:

```
8 5 + 3 * 9 + . NEWLINE      48 OK
```


Gyakorlásul javasoljuk, számolja ki a következő BASIC kifejezések értékét FORTH-ban:

```
15 + 8 - 3 * 4
(31 - 27) * (8 + 9) / 7
(2 * 12 + 4 * 8) * 6
```

A FORTH számábrázolásból következik, hogy a müveletek végeredménye mindig egész szám, másrészt szorzásnál, amikor a végeredmény meghaladja az ábrázolási tartományt (-32768, 32767) az eredmény hibás lesz. Az utóbbi esetben, tulcsordulásnál nem kapunk hibaüzenetet.

A használható aritmetikai utasitások a következők:

+	(n1 n2 -- n)	összeadás, n = n1+n2
-	(n1 n2 -- n)	kivonás, n = n1-n2
*	(n1 n2 -- n)	szorzás, n = n1*n2
/	(n1 n2 -- n)	osztás, n = n1/n2
/MOD	(u1 u2 -- ur uq)	osztás maradékkal, ahol ur a maradék és uq a hányados
1+	(n -- n+1)	egy hozzáadása
2+	(n -- n+2)	kettő hozzáadása
1-	(n -- n-1)	egy kivonása
2-	(n -- n-2)	kettő kivonása
2*	(n -- 2*n)	szorzás kettővel (balra léptetés)
2/	(n -- n/2)	osztás kettővel (jobbra léptetés)
NEGATE	(n -- -n)	előjel váltás (szorzás -1-gyel)
ABS	(n -- /n/)	abszolut érték képzése
MIN	(n1 n2 -- n)	minimális elem meghatározása
MAX	(n1 n2 -- n)	maximális elem meghatározása
*/	(n1 n2 n3 -- n)	normálás, n = n1 * n2/n3

Külön kell szólnunk a legutolsó müveletről, a normálásról. Eltérően a korábban megismertektől, a szorzás eredménye itt

teljes hosszában, 32 biten képződik és ezt használjuk fel az osztásnál. Ez azt eredményezi, hogy nem veszitünk a pontosságból. Használhatjuk például százalékszámitásnál. Hány százaléka 35 a 83-nak?

```
35  1ØØ  83  */  .  NEWLINE  42 OK
```

Vagy például mennyi egy 49 cm sugaru kör kerülete?

```
49  31416  5ØØØ  */  .  NEWLINE  3Ø7 OK
```

Utóbbi példánkban pi kétszeresének racionális tört közelitését használtuk.

2.5 Veremtár müveletek

Ahhoz, hogy kényelmesen használhassuk a veremtárat, szükségünk van olyan utasitásokra, melyek megváltoztatják az adatok sorrendjét a stack-ben.

Szemléltetésül lássunk néhány példát:

```
1 2 3 . . .        NEWLINE  3 2 1 OK
1 2 3 SWAP . . .   NEWLINE  2 3 1 OK
1 2 3 OVER . . . . NEWLINE  2 3 2 1 OK
1 2 3 ROT . . .    NEWLINE  1 3 2 OK
```

A stack-utasitások összefoglalása:

DROP	(n --)	törli a stack legfelső elemét
DUP	(n -- n n)	megduplázza a stack legfelső elemét
SWAP	(n1 n2 -- n2 n1)	megcseréli a stack két felső elemét
OVER	(n1 n2 -- n1 n2 n1)	kiemeli a stack második elemét
ROT	(n1 n2 n3 -- n2 n3 n1)	körbeforgatja a stack legfelső három elemét
?DUP	(n1 -- n1 n1) (Ø -- Ø)	megduplázza a stack felső elemét, ha az nem nulla

SP!	(--)	törli a veremtárból valamennyi elemet
'S	(-- u)	betölti a stack-be a legfelső elem cimét.

Az 'S szó segitségével a veremtárban lévő tetszőleges elemhez hozzáférhetünk. Ehhez azt kell tudnunk, hogy minden, a stack-be bekerülő elem esetén a kapott cim kettővel csökken, mig elem kivételekor kettővel nő. Ez azt jelenti, hogy a veremtár a csökkenő memóriacimek felé épül.

A következő utasitássor és az OVER hatása megegyezik:

```
'S 2 + @
```

(Rövidesen megismerjük a @ szót is.)

2.6 Beolvasó és kiirató utasitások

Eddig a számok bevitelének és kiirásának legegyszerübb módját ismertük meg, melyek mellett további utasitások állnak rendelkezésünkre. A billentyüzetről egy karaktert a KEY szóval olvashatunk be. Ilyenkor a program végrehajtása csak azután folytatódik, mikor leütöttünk egy billentyüt:

```
KEY  DUP  .  NEWLINE  A  65 OK
```

Karaktert nemcsak beolvashatunk, hanem ki is irhatunk a képernyőre, méghozzá az EMIT szóval. Egy csillag kivitele például a következő módon történhet:

```
42  EMIT  NEWLINE  * OK
```

A BASIC INKEY$ függvényének FORTH megfelelője a ?TERMINAL szó. Segitségével vizsgálhatjuk meg, van-e lenyomva billentyü a tasztaturán és ha igen, melyik az.

Az egyes karakterek mellett kezelhetünk karakter sorozatokat,

stringeket is. Szöveget igy irhatunk ki:

." HELLO " NEWLINE HELLO OK

A ." egy FORTH szó, ezért utána mindig egy szóköznek kell állnia. A kiiratások formátumának vezérlését szolgálja a CR, PAGE, SPACE és SPACES szavak, melyek leirása az összefoglalóban található.

A memória tetszőleges cimétől kezdődően elhelyezhetünk a billentyüzetről adott hosszuságu stringeket az EXPECT szóval. A hibás leirás javitható a ← gombbal. Amennyiben rövidebb szöveget irunk be, a végét a NEWLINE billentyüvel zárjuk le. Ennek az utasitásnak az inverze a TYPE, amellyel egy cimtől kezdődően adott számu karaktert irathatunk ki.

KEY	(— c)	beolvas egy karaktert a tasztaturól
EMIT	(c --)	kivisz egy karaktert a képernyőre
CR	(--)	uj sort kezd kiiratáskor
SPACE	(--)	egy szóköz karaktert ir ki
SPACES	(n --)	n darab szóközt ir ki
?TERMINAL	(-- n)	n = Ø, ha nincs nyomva billentyü, egyébként annak kódja
EXPECT	(u n --)	u cimtől elhelyez legfeljebb n darab, billentyüzetről beolvasott karaktert, melyek végét a memóriában egy null karakter jelzi; a bevitelt a NEWLINE zárja le
TYPE	(u n --)	u cimtől kezdődően kivisz a képernyőre n darab karaktert
COUNT	(n -- n+1 n)	általában TYPE előtt használatos, FORTH belső formátumu szövegek kiirásánál

PAGE	(—)	törli a képernyőt; hatása az, mint a BASIC-ben a CLS utasitásé
."	(—)	olyan szövegek kiirására szolgál, melyek végét " karakter jelzi
.	(n —)	kiirja az aktuális számrendszerben az n számot
.R	(n1 n2 —)	n2 szélességü mezőn belül jobbra igazitva kiirja az n1 számot
U.	(u1 —)	kiirja az u1, előjel nélküli számot
BL	(— c)	rendszerkonstans; a szóköz karakter kódját adja meg

2.7 A memória kezelése

A szavaknak ez a csoportja a gép memóriájának irását és olvasását teszik lehetővé. Általában változókkal együtt használjuk őket, abszolut cimekkel ritkábban.

Felhasználásukat részletesebben a következő részben ismerjük meg.

! (n u —)	az u cimtől kezdődően elhelyezi a memóriában az n, 16 bites számot
C! (c u —)	elhelyezi a memória u cimén a c karaktert, vagyis a stack második elemének alsó nyolc bitjét
@ (u — n)	a memória u cimén elhelyezkedő 16 bites számot betölti a stack-be
C@ (u — c)	a memória u ciméről betölti a stack-be az ott található 8 bites karaktert; a felső 8 bit a veremtárban nulla lesz

+! (n u --) a memóriában az u cimen lévő 16 bites számhoz hozzáadja n-t

FILL (u n c --) a memóriát az u cimtől kezdődően n hosszuságban feltölti a c karakterrel

CMOVE (u1 u2 n --) a memória u1 ciméről n darab karaktert áthelyez u2 cimre; az átpakolás nem rontja el a forrásterületet

Próbáljuk ki a következő utasitássort:

HEX 78ØØ 2Ø 2A FILL CR 78ØØ 2Ø CR TYPE

Hatására a következő sor elején 32 darab csillag jelenik meg.

3. VÁLTOZÓK ÉS KONSTANSOK

Adatok tárolására eddig a veremtárat használtuk.

Általában szükség van, más programnyelvekhez hasonlóan változókra és konstansokra is. Ezek egy része, az ugynevezett rendszerváltozók és rendszerkonstansok, melyek már definiálva vannak. Ilyen többek között a BASE.

Magunk is definiálhatunk változókat és konstansokat.

Ezzel a szavak uj csoportját ismerjük meg, a definiáló szavakat. Közös jellemzőjük, hogy a szótárban uj elemeket hoznak létre. Definiáljunk például egy FIVE nevü konstanst, melynek értéke 5:

```
5 CONSTANT FIVE  NEWLINE  OK
FIVE  .  NEWLINE 5  OK
```

Valamennyi konstansra igaz, hogy végrehajtásakor a hozzárendelt számérték bekerül a veremtárba.

Hasonlóan definiáljuk a változókat.

```
3ØØ VARIABLE X  NEWLINE  OK
```

Itt a 3ØØ a kezdő érték, melynek megadása kötelező. A változó végrehajtásakor a stack-be annak a 16 bites memóriarésznek a cime kerül, amely az adott változóhoz tartozik és ahol annak értéke tárolódik. Ennek a cimnek az ismerete a programozó szá-

mára általában szükségtelen, az értékadás a memória kezelése cimü részben megismert szavakkal történik. A továbbiakban néhány BASIC utasitás FORTH megfelelőjét mutatjuk be.

BASIC	FORTH
LET X = 7	7 X !
PRINT X	X @ .
LET X = 6 * 8 - 21	6 8 * 21 - X !
PRINT X	X ?

3.1 Uj utasitások létrehozása

FORTH-ban nincs igazi megfelelője a hagyományos értelemben vett programirásnak. A feladat megoldása során az egyes feladatrészekre külön-külön uj utasitásokat hozunk létre, melyeket kipróbálunk, helyes müködésüket ellenőrizzük, majd felhasználjuk ujabb szavak létrehozásához.

Végül egyetlen szót kapunk, mely valójában a megirt programnak felel meg. A programtervezésnek ezt a módját hivjuk alulról felfelé történő épitkezésnek.

Hozzunk létre egy STAR nevü szót, amely kiir egy csillagot:

```
: STAR 42 EMIT ;  NEWLINE  OK
STAR  NEWLINE  * OK
```

A kettőspont és a pontosvessző közötti programrésszel már találkoztunk egy korábbi példában.

A következő, ujonnan definiált szó kiirja a leütött billentyü ASCII kódját:

```
: ?CODE  KEY  . ;  NEWLINE OK
?CODE ?CODE  ?CODE  NEWLINE  A 65  B  66  C  67  OK
```


A definiciónak nem kell egy sorban lennie, közben bármikor leüthetjük a NEWLINE billentyüt. Hatására uj sor kezdődik, de a pontosvessző végrehajtásáig nem jelenik meg az OK.

Ez azt jelzi, hogy a rendszer forditási állapotban van. Egyetlen megkötés, hogy a névnek és a definiáló szónak azonos sorban kell lennie.

3.2 Forditás

A FORTH két állapotban lehet, az egyik az interpretálás, a másik a forditás. Forditáskor a szavak általában nem hajtódnak végre, hanem beépülnek a szótárban a szó definiciójába. Végrehajtásuk a létrehozott szó futásakor történik meg. A definició törzsében szereplő számok is csak ekkor kerülnek a veremtárba és ekkor iródnak ki a ."-val megadott szövegek is.

Például a

```
: .H  HEX  U.  ;  NEWLINE  OK
```

sor nem változtatja meg a számrendszer alapszámát.

A hexadecimálisra való áttérés a .H szó végrehajtásakor történik meg.

Definiáljunk egy olyan szót, amely kiirja decimálisan a számrendszer alapszámát, de azt nem változtatja meg:

```
: ?BASE  BASE  @  DUP  DECIMAL  .  BASE  !  ;
DECIMAL  ?BASE  NEWLINE  1Ø  OK
2 BASE  !  ?BASE  NEWLINE  2  OK
```

A forditás menetének vezérléséhez szükség van olyan szavakra is, melyeket a FORTH nem fordit le a definicióban, hanem végrehajt. Ezeket közvetlen, immediate tipusuaknak nevezzük. Definiálásuk a normál szavakhoz hasonlóan történik, de a pontos-

vessző után irt IMMEDIATE utasitással megváltoztatjuk a tipusát.

```
: Z ." HI"  ; IMMEDIATE  NEWLINE  OK
: W Z  ; NEWLINE  HI OK
W NEWLINE  OK
```

A W szó valójában egy null utasitás, mivel a törzse üres amiatt, hogy nem szerepel benne a közvetlen végrehajtásu Z szó.

3.3 Ujradefiniálás

Uj szavaknak olyan nevet is adhatunk, amely már szerepel a szótárban. Erre a következő hibaüzenet figyelmeztet:

```
* REDEFINED  WORD
```

Természetesen minden szó, ami a korábbi definiciót használta, változatlanul fog müködni, azonban ujabb hivatkozásoknál a legutolsónak ezen a néven definiált szó fog müködni illetve lefordulni.

Programok kipróbálásánál a hibásan müködő szavakat ujradefiniálhatjuk azonos néven, de uj, módositott jelentéssel. Ilyenkor a korábbi változatok zavart nem okoznak, de feleslegesen foglalják a helyet.

3.4 Szavak törlése

Egyes szavakat a szótárból a FORGET utasitással törölhetünk ki.

Például

```
FORGET  Z  NEWLINE  OK
```


Vigyáznunk kell arra, hogy a FORGET nemcsak a megadott szót törli, hanem az összes, időben utána létrehozottat is!

Mielőtt hosszabb munkába fognánk, célszerü elsőnek egy semleges szót definiálni, pl.:

```
: TASK ;
```

Ezután a szavak törléséhez nem kell mindig más, az éppen első szó nevét megjegyezni. Elegendő begépelnünk a FORGET TASK parancsot.

Valamennyi felhasználói definició törölhető a rendszerből az EMPTY szóval. Ekkor csak azok a szavak maradnak meg, melyek a kazettáról való betöltésnél is a szótárban voltak.

A PURGE szó az utolsó szót törli ki. Használhatjuk definició belsejében is, ha azt nem kivánjuk befejezni.

Nem törölhetjük ki azokat a szavakat, melyek a zFORTH rendszerhez tartoznak. Ilyen kisérlet esetén hibajelzést kapunk:

```
FORGET  SWAP  NEWLINE
* MEMORY IS PROTECTED
```

3.5 A forditó ki- és bekapcsolása

Definició beirása közben szükség lehet arra, hogy a rendszert átmenetileg visszakapcsoljuk interpreter módba, mondjuk azért, mert korábban decimálisan számoltunk, de egy karakter kódját csak hexadecimálisan tudjuk.

Erre szolgál a <I és a I> szó.

```
: ABC  <I  HEX  I>  2E  .  ; ABC  NEWLINE  . OK
```

3.6 Hibák javitása definicióban

Soron belül, a NEWLINE beütése előtt a hibák javithatóak a ← (cursor balra) billentyüvel.

Amennyiben ismeretlen szó, (UNKNOWN WORD) hibaüzenetet kapunk, a hibás szótól kezdődően folytathatjuk a beirást, onnan, ahol a forditás abbamaradt.

3.7 Kilépés a szóból

Általában a szóból való kilépés a pontosvesszőnél történik. Időnként szükség lehet ettől eltérő helyen való visszatérésre. Ezt az EXIT szóval jelölhetjük meg. Használatával esetenként futási idő és memória takaritható meg, hiszen nem kell bonyolult programstrukturákkal a végrehajtást a definició végéhez irányitani.

A harmadik fejezetben megismert szavak:

CONSTANT xxx	(n —)	xxx nevü konstans definiálása
xxx	(— n)	xxx végrehajtásakor a stack-be a konstans értéke (n) kerül
VARIABLE	(n —)	xxx nevü, n kezdőértékü változó definiálása
xxx	(— u)	xxx végrehajtásakor a veremtárba a változónak fenntartott 2 byte-os terület cime kerül
: xxx	(—)	uj szó definiálásának a kezdete, xxx néven
;	(—)	lezárja a kettősponttal megkezdett definiciót
<I	(—)	kikapcsolja a forditót

I>	(—)	bekapcsolja a forditót
FORGET xxx	(—)	törli a szótárból az xxx nevü és az összes utána definiált szót
PURGE	(—)	törli az utolsó, illetve definición belül a már megkezdett szót
EMPTY	(—)	törli valamennyi felhasználói definiciót a szótárból
IMMEDIATE	(—)	közvetlen végrehajtásuvá teszi az utolsónak definiált szót
EXIT	(—)	visszatér az előző, a definiciót meghivó szóba

4. LOGIKAI KIFEJEZÉSEK, RELÁCIOK

Számok és konstansok mellett FORTH-ban léteznek logikai értékek is, ugynevezett Flag-ek is. Ezek nem külön tipusok, hanem meghatározott értékü 16 bites számok.

Hamis értékünek tekintjük azt a számot, mely nulla és igaznak mindazokat, melyeknek értéke nem nulla. Amennyiben logikai müveleteket is végzünk velük, célszerü az igaz logikai értékhez az egyet rendelni.

4.1 Összehasonlitások

Logikai értéket általában összehasonlitások, relációk állitanak elő, melyek közül a zFORTH-ban a következők használhatóak:

=	(n1 n2 -- f)	f = 1, ha n1 = n2 f = Ø, ha n1 ≠ n2
Ø<	(n -- f)	f = 1, ha n < Ø f = Ø egyébként
<	(n1 n2 -- f)	f = 1, ha n1 < n2 f = Ø egyébként
U<	(u1 u2 -- f)	f = 1, ha u1 < u2 f = Ø egyébként

Az U< , ellentétben a < relációval, az összehasonlitott számokat előjel nélkülinek tekinti, ezért előnyösen használható például memória cimek összehasonlitására.

4.2 Logikai müveletek

A flag értékének invertálására szolgál a NOT szó:

NOT	(n -- f)	f = Ø ha n ≠ Ø f = 1 ha n = Ø

A következő szavak a logikai müveleteket a veremtár tetején lévő 16 bites számok között bitenként végzik el.

Helyes eredményt adnak flag-ekre is, ha az igaz értéknek az 1 felel meg:

AND	(n1 n2 -- n)	n = n1 ∧ n2	(logikai "és")
OR	(n1 n2 -- n)	n = n1 ∨ n2	(logikai "vagy")
XOR	(n1 n2 -- n)	n = n1 ⊕ n2	(logikai "kizáró vagy")

5. FELTÉTELES PROGRAMSZERKEZETEK, CIKLUSOK

Programok irása során szükség van olyan utasitásokra,melyek bizonyos programrészek egyszeri vagy többszöri végrehajtását valamilyen feltételtől teszik függővé. A következőkben ezekkel ismerkedünk meg.

5.1 Az IF szó

Mint minden strukturavezérlő utasitás, csak definició belsejében fordulhat elő.

```
: ?? IF ." NEM " THEN ." NULLA" ;
Ø ?? NEWLINE NULLA OK
61 ?? NEWLINE NEM NULLA OK
```

Az IF és a THEN közötti utasitássor attól függően hajtódik végre, hogy az IF a stack tetején igaz logikai értéket talál-e.

Kijelölhetünk olyan részt is, ami a feltétel nem teljesülése esetén hatásos:

```
: ?PAR 2 /MOD DROP IF ." PARATLAN" ELSE
  ." PAROS" THEN ;
 6 ?PAR NEWLINE PAROS OK
21 ?PAR NEWLINE PARATLAN OK
```


5.2 Határozatlan ciklusok

A határozatlan ciklusok valamilyen feltétel teljesülésétől függően ismétlik meg egy programrész végrehajtását.

Irjunk olyan szót, amely megvárja, amig a billentyüzetről A betü nem érkezik:

```
: ?A BEGIN KEY 65 = UNTIL ;
```

A BEGIN és UNTIL közötti rész addig ismétlődik, amig a BEGIN a stack tetején hamis értéket talál.

Vannak esetek, amikor zavaró, hogy az előbbi ciklusnak a végén van a feltételvizsgálat és ezért mindenkor legalább egyszer lefut. Ilyenkor használható a másik fajta ciklus-utasitás, melynek formája:

..... BEGIN	..... WHILE	..... REPEAT	
IF xxx THEN	IF:(f --)		az xxx szó hajtódik végre, ha IF végrehajtásakor f≠Ø, ha f=Ø, a végrehajtás a THEN után folytatódik
IF xxx ELSE yyy THEN	IF:(f --)		az xxx szó hajtódik végre, ha IF végrehajtásakor f≠Ø volt, egyébként yyy; a végrehajtás mindkét esetben a THEN után folytatódik
BEGIN xxx UNTIL	UNTIL: (f --)		addig ismétlődik xxx szó vagy szócsoport, amig UNTIL végrehajtásakor f=Ø
BEGIN xxx WHILE yyy REPEAT	WHILE: (f --)		ha WHILE végrehajtásakor f=Ø, a program a REPEAT utáni részszel folytatódik, egyébként yyy-t hajtja végre, majd visszatér xxx-hez

5.3 Határozott ciklusok

A ciklusszervező utasitások most ismertetésre kerülő csoportja a BASIC FOR .... NEXT utasitásához hasonlóak; a ciklus magja meghatározott számuszor fut le.

A mintapéldákat ismét BASIC megfelelőjükkel adjuk meg:

```
BASIC                    FORTH
10 FOR I = 1 TO 8        : XX 8 Ø DO ." *" LOOP ; XX
20 PRINT "*" ;           NEWLINE ******** OK
30 NEXT I
RUN
********
```

Valamennyi DO ciklus indulásakor a stack-ben kell lenni a kezdő és a végértéknek, melyet a DO szó kivesz a veremtárból:

```
DO  (végérték  kezdőérték --)
```

5.3.1 LOOP

A legegyszerübb esetben a ciklust egy LOOP utasitás zárja le. Ilyenkor a ciklus változója mindig eggyel nő. Végrehajtása akkor fejelződik be, amikor a ciklusváltozó eléri vagy meghaladja a végértéket.

5.3.2 +LOOP

A +LOOP szó esetén a ciklusváltozó nem eggyel, hanem a veremtár tetején lévő értékkel nő. Ha a növekmény pozitiv, a ciklus befejeződésének feltétele megegyezik a LOOP-al. Negativ növekmény esetén a ciklus akkor fejeződik be, ha az index kisebb a végértéknél.

5.3 .3 /LOOP

A ciklusváltozó (index) itt is a stack tetején lévő értékkel nő, a ciklusvége vizsgálat azonban a számokat előjel nélkü-

lieknek tekinti. Többek között memória cimekkel kapcsolatban előnyös a használata.

5.3.4 A ciklusváltozó

A mindenkori legbelső ciklus változójának az értékét a stackben az I szóval kapjuk meg, mig J a külső DO ciklus változóját tölti be a veremtárba:

```
: W1 5 Ø DO I . LOOP ;  NEWLINE  OK
W1  NEWLINE  Ø 1 2 3 4  OK

: W2 4 1 DO 3 Ø DO I J * . LOOP 2 +LOOP ;
W2  NEWLINE  Ø 1 2 Ø 3 6  OK
```

Megkaphatjuk a további külső ciklusok változóját is, ha a ciklusban a megfelelő helyre beirjuk a következő programrészt:

```
..... 'RP  n  4  *  +  .....
```

ahol n megadja, hányadik ciklus változóját kérjük.

A fenti programrész n = Ø esetén I-vel, n = 1 esetén pedig J-vel ekvivalens.

5.3.5 Kilépés a DO ciklusból

A DO ciklusból kiléphetünk az előzőekben leirt feltételek teljesülése előtt is a LEAVE szó használatával.

A LEAVE egyenlővé teszi a ciklusváltozót a végértékkel, aminek következtében a ciklus végrehajtása a megfelelő LOOP utasitásnál szabályosan be fog fejeződni. Célszerüen egy IF struktura belsejében áll és a LOOP előtt kerül közvetlenül végrehajtásra.

A következő, DEL szó végrehajtása például bármelyik billentyü leütésére félbeszakad:

```
: DEL 65ØØØ  Ø  DO  ?TERMINAL IF LEAVE THEN  1
  /LOOP  ;
```

5.4 A CASE struktura

Az előzőektől eltérő feltételes programszerkezet kialakitását teszi lehetővé a CASE utasitás, melynek felépitése:

```
..... CASE ..... OF ..... ENDOF ..... OF ..... ENDOF
..... CASEND ....
CASE  (n --)          CASEND (--)
OF    (n1 --)         ENDOF  (--)
```

A CASE kiemeli a stack tetején levő számot. Ezután minden egyes OF szó ellenőrzi, hogy a stack tetején lévő érték egyezik-e azzal, amit a CASE talált. Ha igen, végrehajtódik az OF és ENDOF között szócsoport, majd a program futása a CASEND után folytatódik. Amennyiben az egyenlőség nem teljesül, a futás az ENDOF után folytatódik.

Ez a struktura előnyösen használható menü jellegü feladatoknál. A következő mintaprogram szövegesen kiirja, milyen szám van a stack-ben, ha az nulla és három közé esik:

```
: ?HM  CASE  Ø  OF ." NULLA"  ENDOF
             1 OF ."  EGY "   ENDOF
             2 OF ."  KETTŐ"  ENDOF
             3 OF ."  HAROM"  ENDOF
             ." NEM TUDOM "   CASEND ;
```

5.5 Struktura-vezérlő utasitások egymásba ágyazása

A különböző tipusu struktura-vezérlő utasitások tetszőleges mélységben egymásba ágyazhatóak. Az egyetlen szabály, amit be kell tartanunk az, hogy minden megkezdett strukturát megfelelően le kell zárni.

Hibás például a következő megoldás:

```
..... BEGIN  3  +LOOP ...
```


Ilyen esetben a forditóprogram

```
= INCORRECT STRUCTURE
```

hibajelzést ad.

6. ADATTÁROLÁS KAZETTÁN

Mind forrásnyelvi programok, mind tetszőleges adatok tárolása megoldható kazettán. Az adatok ki- és bevitele minden esetben 1 kbyte (1Ø24 byte) hosszuságu blokkokban történik a magnetofon és a memóriának egy erre a célra kijelölt része, a block-buffer között. Az egyes blokkok azonositása a szalagon 1 és 32767 közé eső számmal történik.

6.1 A buffer

A buffer kezdőcimét a FIRST rendszerkonstans mutatja, hosszát B/BUF. A bufferbe adatokat ténylegesen a FIRST +4 cimtől vihetünk be, az első négy byte a rendszer számára van fenntartva. A buffer alapállapotban szóközökkel van feltöltve, a végét pedig egy 16 bites nulla jelzi.

6.2 A BLOCK szó

Kazettáról a bufferbe egy blokkot a BLOCK szóval tölthetünk be, amely visszaadja a buffer tényleges kezdőcimét.

Amennyiben a bufferben nem az a blokk található, mint amelyiknek a számát megadtuk, a BLOCK betölti azt a kazettáról. Amikor a szalagon megtalálja a keresett részt, a képernyő jobb felső sarkában ezt egy F betü és a mellette villogó csillag jelzi.

Az F helyén megjelenő kérdőjel azt mutatja, hogy a szalagnak azon a részén zFORTH formátumu, de eltérő számu blokk található, amelyet nem olvas be a bufferbe.

Negativ blokkszámot megadva a legelső blokk kerül beolvasásra a kazettáról, annak tényleges blokkszámától függetlenül.

Ha nulla blokkszámot adunk meg, a buffer tartalmától függetlenül mindig a tényleges kezdőcimet kapjuk meg és elmarad a szalagról olvasás.

6.3 A LOAD

A LOAD szó abban tér el a BLOCK-tól, hogy a kivánt blokk betöltése után azt végre is hajtja az interpreter. Paraméterezése megegyezik a BLOCK-al. Nem adja vissza a buffer kezdőcimét.

6.4 UPDATE

Az UPDATE utasitással megjelölhetjük a buffer tartalmát, hogy az megváltozott, módosult. Ha ezt megtettük, a BLOCK és LOAD utasitás nem hajtódik végre, hanem

```
# BUFFER IS NOT EMPTY
```

figyelmeztetést kapunk. Vagy töröljük a buffer tartalmát az EMPTY-BUFFERS szóval, vagy visszairjuk a kazettára a FLUSH utasitással.
Az UPDATE használata nélkül a buffer korábbi tartalma elveszhet.

6.5 A READY flag

A READY flag rendszerváltozó, melynek alapértéke nulla (hamis). Ha tartalma nem nulla, minden olyan szó, amely bekapcsolná a magnetofont, a

```
RECORDER READY (Y) ?
```

üzenetet irja ki és csak akkor folytatódik, ha egy Y-t irunk be.

6.6 A magnetofon kijelölése

Az aktuális magnetofon kijelölését a DRIVE rendszerváltozó tartalmazza, melynek kezdeti értéke nulla. Ekkor minden müvelet a beépitett magnetofonra vonatkozik. A külső magnetofonra ugy kapcsolhatunk át, hogy a DRIVE változóba 16-ot irunk:

```
DECIMAL 16 DRIVE !
```

6.7 BUFFER

A memóriában lévő buffert egy adott számu blokkhoz rendelhetjük anélkül is, hogy azt a kazettáról töltenénk be. Erre például akkor lehet szükség, amikor létrehozunk egy blokkot vagy eleve tudjuk, hogy nem akarjuk majd kimenteni szalagra a tartalmát, csak átmeneti időre kell. Hasonlóan a BLOCK-hoz, visszaadja a stack-ben a buffer tényleges kezdőcimét.

6.8 További, a blokkok kezelésével kapcsolatos szavak

-->	(--)	ha egy LOAD szóval betöltött blokk végén áll, betölti a kazettáról a következő, eggyel nagyobb sorszámu blokkot,
DIR	(--)	megkeresi a szalagon az első blokkot, kiirja annak számát és első sorát, mely általában a tartalomra utaló megjegyzést tartalmaz; a BREAK hatására futása félbeszakad, mig tetszőleges billentyü benyomására folytatódik;
BLK	(-- n)	rendszerváltozó; LOAD végrehajtása közben a blokk számát tartalmazza, egyébként nulla.

7. AZ EDITOR

Az editor 1 Kbyte-os blokkokban, ugynevezett screen-ekben elhelyezkedő forrásnyelvi FORTH programok irását illetve javitását teszi lehetővé. A buffer területen lévő blokkon dolgozik; szövegszerkesztéshez a blokk kijelölése a BLOCK szóval azonos módon történik:

EDIT	(n --)	ahol n a blokk száma, amit szerkeszteni akarunk.

Amennyiben uj, a kazettán még nem létező számu blokkot kivánunk létrehozni, a bufferhez a BUFFER szóval kell hozzárendelni.

7.1 Az editor állapotai

A szövegszerkesztő program két állapotban lehet; az egyikben lehetséges a kiterjedt szövegkezelő szolgáltatások elérése, mig a másikban szöveget irhatunk be.

7.1.1 Szerkesztő mód

Ezt az üzemmódot a képernyőn az aláhuzásos cursor jelzi.

A szövegszerkesztő utasitások a következőek:

billentyü	funkció
I	- áttérés beirás (inzert) üzemmódba
D	- szó törlése a cursortól
M	- beszurja a cursor helyére a D-vel törölt szót
L	- törli a sort a cursortól
U	- beszurja a cursortól az L-lel törölt sort
SHIFT/K	- kiejti a cursor által megjelölt sor
SHIFT/B	- üres sort szur be a cursor-al megjelölt sor fölé
SHIFT/S	- törli az ernyőt a cursor-tól
X	- beszurja a cursor helyére az utolsónak törölt vagy beirt karaktert
B	- szóközt szur be a cursor helyére

7.1.2 Beirás üzemmód

A szerkesztőprogramnak ezt az üzemmódját a villogó cursor jelzi. Ilyenkor irhatunk be a blokkba forrásnyelvi programokat.
A visszatérés a szerkesztő üzemmódba a BREAK billentyüvel lehetséges.

7.1.3 Mindkét üzemmódban használható szolgáltatások

billentyü	funkció
←	- cursor balra
→	- cursor jobbra
↑	- cursor fel
↓	- cursor le

billentyü	funkció
SHIFT/↑	- home, a cursort az ernyő bal felső sarkába állitja
SHIFT/↓	- kilépés az editorból a FORTH-ba
SHIFT/←	- törli a cursor-tól balra eső karaktert
SHIFT/→	- a tabulálás a következő szó elejére
CLEAR	- törli a cursor-al megjelölt karaktert
NEWLINE	- a cursort a következő sor elejére állitja

7.2 Az editorhoz tartozó szavak

EDIT	(n —)	blokk kijelölése szerkesztéshez
SCR	(— n)	rendszerváltozó, az utolsónak editált blokk számát tartalmazza
C/L	(— n)	rendszerkonstans; azt adja meg, hány karakterből áll egy sor

8. SZÁMKONVERZIÓ

8.1 Kimeneti számkonverzió

Számok kiiratására a már megismerteken tulmenően magunk is definiálhatunk uj szavakat. Legyen például .A egy olyan szó, amely minden esetben négy számjegyet ir ki, függetlenül az értéktől:

```
: .A Ø  <# # # # # #>      TYPE ;
37 .A NEWLINE  ØØ37  OK
```

`<#  (d -- d)`

A számkonverzió kezdetét jelzi; a veremtárban egy duplapontos, vagyis 32 bites, előjel nélküli számot vár. Ezt legegyszerübben egy szimplapontos számból ugy állithatjuk elő, hogy egy nullát töltünk még be a stack-be.

`#>  (d -- u n)`

A számkonverzió lezárására szolgál, visszaadja a kikonvertált karaktersorozat kezdőcimét és a hosszát olyan formában, hogy a TYPE szóval az kiiratható.

`#  (d1 -- d2)`

Számkonverzió belsejében fordul elő. Minden egyes végrehajtáskor egy számjegyet konvertál át karakterré a számból a legkisebb helyértéktől kezdődően.

`#S  (d -- Ø Ø)`

Szintén számkonverzió belsejében fordul elő; folyamatosan konvertálja a számokat addig, amig az értékes jegyek tartanak, vagyis biztositja a nulla elnyomást a szám elején.

`HOLD  (c --)`

Segitségével a veremtár tetején levő, c kódu karaktert beszurhatjuk a konvertált szám megfelelő jegyei közé.

Például az alább definiált .F szó mindig két tizedesjegyet is kinyomtat:

```
: .F Ø <# # # 46 HOLD #S #>     TYPE SPACE;
1Ø .F NEWLINE  Ø.1Ø  OK
326Ø .F NEWLINE  32.6Ø  OK
```

Eddig nem beszéltünk a számok előjeléről és a negativ számokról.
Az előjel kiirására szolgál a

`SIGN  (n d -- d)` szó.

Végrehajtásakor, ha n negativ volt, a karakter stringbe beszur egy negativ előjelet, ha n pozitiv, semmilyen karakter nem kerül be a stringbe.

A már ismert . szó definiciója az előzőek alapján igy adható meg:

```
: . DUP  ABS Ø <# #S SIGN #>  TYPE SPACE;
```

Irjunk olyan szót, amelyik a stack tetjén lévő karaktert hexadecimálisan mindig két számjeggyel irja ki, de nem rontja el a korábbi számrendszert:

```
: .B BASE @ SWAP HEX Ø <# # # #> TYPE SPECE BASE ! ;
DECIMAL 3 .B 255 .B 127 .B NEWLINE  Ø3 FF 7F  OK
```

A következőben olyan uj szót hozunk létre, amely hasonlóan működik mint a .R, de az előjelet a szám végére irja és az elnyomott nullák helyére csillagokat tesz.

```
: ¤.R  OVER  ROT ABS Ø  <#  SIGN   #S   #>
       ROT OVER OVER  -  DUP Ø <
       IF NEGATE Ø DO 42 EMIT LOOP ELSE DROP THEN DROP
       TYPE;

 5 4  ¤.R  NEWLINE  ¤5  OK
-1Ø 4 ¤.R  NEWLINE  ¤1Ø- OK
```

8.2 Bemeneti számkonverzió

Karakteres, string formában adott számok bináris alakura konvertálásához két szó áll rendelkezésre.

>BINARY (d1 n1 -- d2 n2)

Az n1+1 cimtől kezdődően addig konvertálja a karaktereket, az aktuális számrendszerben, amig meg nem találja az első nem konvertálható karaktert, melynek a cime n2.

d1 a konverzióhoz szükséges 32 bites munkaterület, a konverzió kezdetén célszerüen nulla. Az eredmény d2-ben keletkezik, szintén 32 biten!

NUMBER (u -- n)

Az u+1 cimen kezdődő előjeles számot az aktuális számrendszerben bináris alakura konvertálja, az eredmény n.

9. SPECIÁLIS zFORTH SZAVAK

Ezek a szavak nem találhatóak meg a FORTH szabvány utasitásai között, mert kifejezetten a HT-1Ø8ØZ tipusu mikroszámitógép lehetőségeinek a teljesebb kihasználását szolgálják.

9.1 Grafika

Hasonlóan a BASIC-hez, a grafikai utasitások a képernyőt pontonként kezdik, melyek száma soronként 128, oszloponként pedig 48.

SET (u1 u2 --)	bekapcsolja a képernyőn az u1, u2 koordinátáju pontot, ahol u1 = sorszám (Ø, 47) u2 = oszlopszám (Ø, 127)
RESET (u1 u2 --)	törli az u1, u2 koordinátáju képpontot
POINT (u1 u2 -- f)	teszteli az u1, u2 koordinátáju képpontot; f=1, ha a pont be van állitva, f=Ø egyéb esetekben

A következő szó egy vizszintes vonalat huz a megadott sorba: (sor --):

```
: LINE  128  Ø  DO DUP  I  SET LOOP DROP ;
```

9.2 Hanggenerátor

A beépitett hanggenerátor kezelésének megkönnyitésére külön szó van:

SOUND (u1 u2 --)

A hanggenerátor u2 csatornájára kiadja az u1, nyolc bites értéket. u2 nulla és tizenöt közé kell, hogy essen.
A géphez eredetileg mellékelt "Használati útmutató" 30. oldalán közzétett, a tenger morajlását utánzó BASIC programot zFORTH-ba igy irhatjuk át:

```
: TENGER Ø Ø 14 6Ø 255 16 16 16 199 31 Ø Ø
         Ø Ø Ø  Ø  16 Ø  DO   I SOUND LOOP ;
```

A hanggenerátort a CSEND szóval kapcsolhatjuk ki:

```
: CSEND  16 Ø DO Ø I SOUND LOOP ;
```

9.3 Véletlenszám generátor

Véletlenszámok előállitására szükség lehet mind egyes játékoknál, mind különböző szimulációs és modellezési feladatok esetében.

RND (n1 - n2)

Ez a szó előállit egy véletlenszámot, n2-t, amely Ø és n1 közé esik; n1 lehet negativ is.

A következő szó beállitja a képernyő egymást követő képpontjait. Az egyenletesen kifehéredő ernyő a generátor egyenletes eloszlását bizonyitja.

```
: TEST PAGE CRS-OFF BEGIN 48 RND 128 RND SET
       ?TERMINAL UNTIL ;
```

A szó futása bármelyik billentyü lenyomására abbamarad.

9.4 Portok kezelése

Különböző hardver bővitések kezeléséhez szükség van nemcsak a memória, hanem a portok olvasására és irására is.

P! (c u --)	az u pontra kiviszi a c karaktert, vagyis a veremtár második elemének alsó nyolc bitjét
P@ (u -- c)	az u portról beolvas nyolc bitet, c felső nyolc bitje nulla lesz, hasonlóan a C@ szóhoz.

Mindkét esetben a port cime, u Ø és 65535 közé eső tetszőleges érték lehet.

9.5 Képernyő kezelés

Az eddig megismert PAGE, CR, SPACE, SPACES, EMIT szavakon kivül, melyek minden FORTH rendszerben megtalálhatóak, a zFORTH további lehetőségeket biztosit:

CRS-ON	(--)	bekapcsolja a cursort
CRS-OFF	(--)	kikapcsolja a cursort
BCRS	(--)	villogó cursort kapcsol be
UCRS	(--)	aláhuzásos cursort kapcsol be
SCRS	(u1 u2 --)	a cursort az ernyő u1, u2 pozíciójába állitja ahol u1 = sor száma (Ø,15) u2 = oszlop száma (Ø,63)
?CRS	(--u1 u2)	megadja a cursor pozicióját; u1 = sor száma u2 = oszlop száma
VDR	(--u)	konstans, megadja a memóriába illesztett képernyő első pozíciójának, vagyis a bal felső sarokban lévő karakternek a memóriacimét

10. RENDSZERPARANCSOK

A következő szócsoport szavai a rendszer állapotát módositják.

COLD	(--)	hidegstart; hatására a rendszer abba az állapotba kerül, amilyenben a kazettáról való betöltés után, közvetlenül van
QUIT	(--)	visszaadja a vezérlést a külső interpreternek, amely ezután egy utasitássor beirását várja; ilyenkor nem iródik ki az OK
ABORT	(--)	hasonlóan a QUIT-hez, visszaadja a vezérlést az interpreternek, de egyuttal törli a veremtárat és a számkonverziót decimálisra állitja
DECIMAL	(--)	a számkonverzió alapszámát tizesre állitja
HEX	(--)	a számkonverzió alapszámát átkapcsolja 16-osra, vagyis hexadecimálisra
BYE	(--)	visszaadja a vezérlést a BASIC-nek

11. SZÓTÁRKEZELÉS

A szótárban az egyes szavak egymás után helyezkednek el, a növekvő memóriacimek irányában. Kezelését az eddigi szavak automatikusan elvégezték.

A szótárban mi is elhelyezhetünk különböző egy illetve két byte-os számokat, vagy definiálhatunk ujabb, a könyvtárat kezelő szavakat.

C,	(c --)	a c karaktert elhelyezi a szótárban, a szabad tárolóhelyek száma 1-gyel csökken
,	(n --)	az n 16 bites számot elhelyezi a szótárban a következő helyre, a szabad tárolóhelyek száma 2-vel csökken
HERE	(-- u)	megadja a szótárban az első szabad memóriahely cimét
?FREE	(-- u)	megadja a szótárban szabadon maradt tárolóhelyek (byte-ok) számát
ALLOT	(n --)	n értékét hozzáadja a szótár szabad helyére mutató pointer értékéhez.

A , szót például a következő módon definiálhatjuk le:

```
: , HERE ! 2 ALLOT ;
```

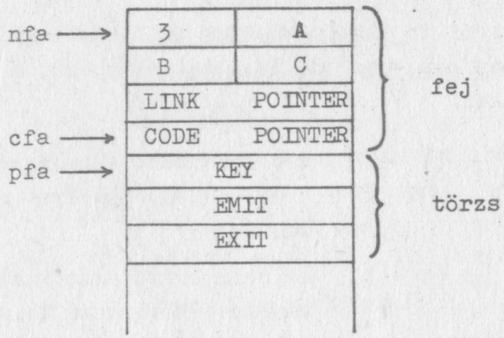
11.1 A szavak felépitése

A szavak, melyek a szótárban találhatóak, két fő részből épülnek fel. A fej tartalmazza a nevet, a tipus megjelölését, mig a törzsben találjuk meg definiciók esetén a müködés leirását, konstansok és változók esetén pedig magát az értéket.

Ehhez a felépitésmódhoz néhány jellegzetes fogalom tartozik.

Például a

: ABC KEY EMIT ; a következő módon épül fel:



Az nfa az a cim, mely a név elejére mutat.
A cfa azt a cimet jelenti, ahol a szó tipusára utaló érték található.
A pfa a definició törzsének a kezdőcimét jelenti.

Ha ismerjük a szóhoz tartozó pfa értékét, a szót közvetlenül is végrehajthatjuk egy EXECUTE utasitással.

EXECUTE (pfa —)

A szavak pfa cimének a meghatározására a név alapján két szavunk van:

FIND xxx (— pfa) ha megtalálta
(— ø) ha nincs benne a szótárban
' xxx (— pfa)

Interpreter módban elhelyezi a veremtárban xxx pfa cimét. Definició belsejében használva xxx cime akkor kerül be a stack-be, ha a definiált szót végrehajtjuk.

CFA (nfa — cfa) a szó név mezőjének kezdőciméből meghatározza a cfa-t

12. FORDITÁS VEZÉRLÉSE

Forditás közben annak menetét több szó segitségével módosithatjuk illetve létrehozhatunk olyan, természetesen IMMEDIATE tipusu szavakat, melyek befolyásolják a forditás menetét.

Azt, hogy a rendszer milyen állapotban van, a STATE rendszerváltozó alapján dönthetjük el.

STATE (-- u) tartalma = Ø interpreter módban
= 1 forditás közben

12.1 Literálok

Kettőspontos definició belsejében a leirt számok automatikusan beépülnek. A stack tetején lévő szám is beépithető azonos későbbi végrehajtással:

LITERAL (n --) n-t beirja a definicióba; a szó végrehajtásakor n visszakerül a stack-be;

12.2 Immediate szó leforditása

Definición belül annak törzsébe beépithetünk közvetlen végrehajtásu szavakat is, melyek egyébként végrehajtódnának:

COMPILE xxx (--)

13. REKURZIÓ

zFORTH-ban lehetőség van rekurziv szavak definiálására is. Kettőspontos definición belül önmagára a SELF (--) szóval hivatkozunk.

14. MEGJEGYZÉSEK ELHELYEZÉSE A PROGRAMBAN

A forrásnyelvi programokban elhelyezhetünk megjegyzéseket is:

(COMMENT)

A nyitó zárójel önálló FORTH szó, ezért utána szóköznek kell állnia. A megjegyzés az első csukó zárójelig vagy annak hiányában a sor végéig tart.

15. A RETURN-STACK

FORTH-ban az eddig használt paraméter stack mellett létezik egy másik, ugynevezett return-stack is. Ebben tárolja a rendszer a definiciók kifejtése közben a visszatérési cimeket ahhoz hasonlóan, mint gépi kódu szubrutinok hivásánál a gépi stack és ide kerülnek a DO ciklusok esetén a ciklusváltozók és a végértékek is.

Definición belül jól kihasználható átmeneti tárolóként. Nagyon fontos azonban, hogy a szóból való kilépés előtt minden, oda átrakott számot kivegyünk. Ellenkező esetben a rendszer megsérülése következhet be!

>R	(n --)	a stack tetejét áttölti a return-stack-be
R>	(-- n)	a return stack legfelső elemét átrakja a paraméter stack-be
I	(-- n)	a return-stack legfelső elemét lemásolja a stack-be annak megváltoztatása nélkül
J	(-- n)	a return-stack harmadik elemét másolja be a paraméter stack-be anélkül, hogy azt elrontaná

16. A PAD

PAD (-- u)

Megadja a karakterkezeléshez használható átmeneti tárolóterület kezdőcimét. Helye változó, de ujabb szótárelem létrehozásáig illetve valamely régebbi törléséig állandó.

17. A SZÓTÁR TARTALMÁNAK KILISTÁZÁSA

A VLIST (--) utasitással kilistázhatjuk a szótárban lévő szavak neveit. A VLIST futása leállitható bármelyik billentyü leütésével. A képernyőn kiirásra kerül a szó pfa cime, a név hossza és azon első karakterek, melyek alapján azonositásuk történik.

18. TÖMBÖK DEFINIÁLÁSA

Változók számára nemcsak két byte-ot foglalhatunk le, hanem tetszőleges számut is. Például tartsunk fenn a T tömb számára 4Ø byte-ot:

```
<BUILDS T 4Ø ALLOT  NEWLINE  OK
```

A <BUILDS utasitással definiált T szó változó tipusu lesz, tehát végrehajtásakor a stack-be a pfa cim kerül.

A <BUILDS nem foglal le helyet a törzsben.

19. DEFINIÁLÓ SZAVAK DEFINIÁLÁSA

A következőkben a FORTH talán leghatékonyabb eszközét ismerjük meg, a definiáló szavak definiálásának a módját. Ezekkel a szavakkal tetszőleges uj adat és programstrukturák kialakitására nyilik mód.

Hozzunk létre olyan szót, amellyel karakteres, tehát egy byte-os változókat definiálhatunk:

```
: CVAR <BULIDS C, ;
```

Ennek használata:

```
5 CVAR W  NEWLINE  OK
W C@ .  NEWLINE 5  OK
```

Egydimenziós tömbök definiálására szolgál a következő példa BUF szava:

```
: BUF <BUILDS ALLOT ;
  BUF xxx      (u --) u byte-os buffert foglal le, melynek
                      neve xxx
      xxx      (u --) xxx végrehajtásakor a veremtárba a
                      buffer kezdőcime kerül
```

19.1 DOES>

Az előző pontban a <BUILDS szót felhasználó uj definiáló szavak közös jellemzője volt, hogy változó jellegü szavakat

definiáltak; az ujdonságot a forditási időben elvégzett utasitások megadásának lehetősége jelentette.

A DOES> szóval megadhatjuk az uj definiáló szóval definiált utasitások futási idejü müködését is.

Karakteres konstansok definiálására szolgál a CKON szó, melynek definiciója:

```
: CKON <BUILDS C, DOES> C@ ;
```

A definiáló szó, CKON használatakor azok a szavak hajtódnak végre, melyek a <BUILDS és a DOES> között vannak, vagyis a konstans értéke a C, szó hatására bekerül a konstans törzsébe.

A definiált szó futásakor betöltődik a stack-be a szónak a pfa cime és végrehajtódnak azok az utasitások, melyek a definiáló szóban a DOES> és a pontosvessző között vannak.

Példánknál maradva a C@ a pfa cimről kiolvassa a konstanst és elhelyezi a veremtárban.

20. A WORD SZÓ

Az interpreternek beirt sor a memóriának erre a célra fenntartott részébe kerül, a terminál input bufferbe. Innen az egymást követő szavak kiemelhetőek a WORD segitségével:

WORD (c -- u)

A szó végrehajtása előtt a stack-ben el kell helyezni a stringet határoló karakter kódját. Ez általában szóköz, de tetszőleges más karakter is lehet.

A karaktersorozat átmásolódik a memória u+1 cimétől kezdődően, az u cimen levő byte tartalmazza a karakterek számát.

A WORD ujabb végrehajtásakor az input sor következő szava kerül átmásolásra.

A zFORTH az input sor feldolgozása közben maga is ezt a szót használja, igy segitségével egyes stringeket "ellophatunk" az interpreter elől.

Például azt akarjuk, hogy minden (*-gal kezdődő és *-gal záródó karaktersorozat iródjon ki a képernyőre a következő sorban:

```
: (* 42 WORD CR COUNT TYPE ; IMMEDIATE
```

Megjegyzés: nyilvánvaló, hogy a string belsejében nem szerepelhet a * karakter.

Használjuk ki a <BUILDS és DOES> nyujtotta lehetőségeket olyan definiáló szó létrehozására, amely olyan szövegkonstansokat definiál, melyek végrehajtásukkor kiiródnak a képernyőre:

```
: SCON    <I HEX I>   <BUILDS  22  WORD  >R
          I HERE I 1+  CMOVE R>    1+ ALLOT
          DOES>  COUNT TYPE ;

SCON SC GOOD BYE"  NEWLINE  OK
SC  NEWLINE  GOOD BYE OK
```

21. SYSTEM UTASITÁS

Az alap változat szükség esetén kiegészithető különböző bővitésekkel, mint például assembler, dupla pontosságu aritmetikai utasitások stb.

Ezeket a kazettáról a SYSTEM szóval tölthetünk be:

SYSTEM (n --) ahol n annak a kiterjesztésnek a száma, amit be akarunk tölteni

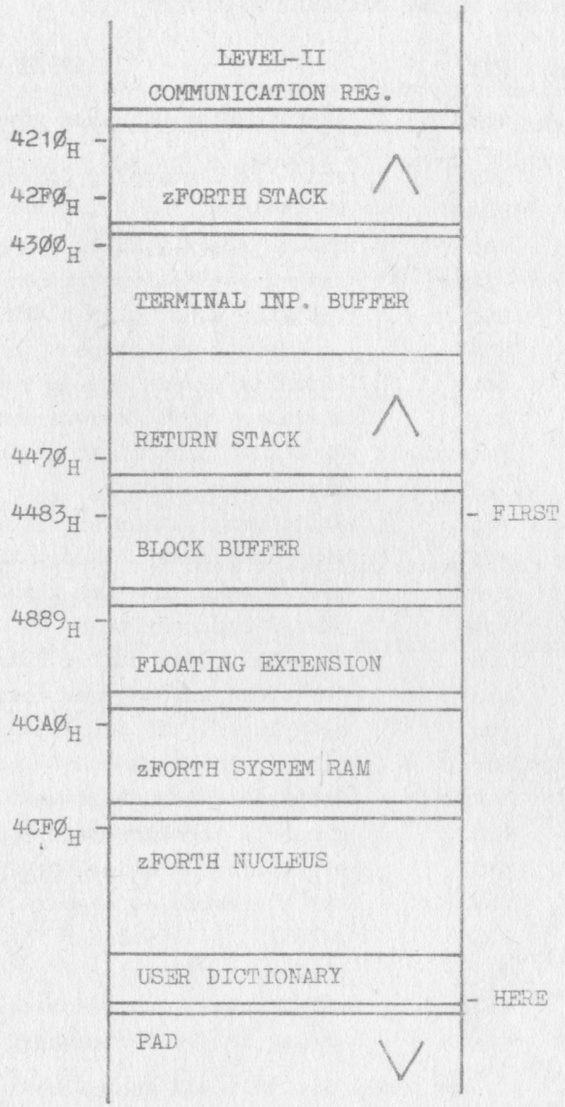
Az esetleg megjelenő REDEFINED WORD üzenetet hagyjuk figyelmen kivül, más hibák esetén ujra töltsük be a kiterjesztést.

A. függelék: Hibaüzenetek leirása

Az alábbiakban összefoglaljuk a zFORTH hibaüzeneteit:

STACK IS EMPTY	- a veremtár kiürülését jelzi
STACK IS FULL	- a veremtár megtelését jelzi
UNKNOWN WORD	- a szó nem található meg a szótárban
REDEFINED WORD	- ilyen nevü szó már található a szótárban
BASE OUT OF RANGE	- hibás érték található a BASE változóban
MEMORY IS PROTECTED	- a törölni kivánt szó védett vagy pedig elfogyott a szótárban a szabad tárterület, nem lehet ujabb szavakat definiálni
INCORRECT STRUCTURE	- hibásan egymásba ágyazott vagy le nem zárt strukturavezérlő utasitások
MISSING NAME	- lemaradt az uj szótári elem neve a definiáló szó után
SYSTEM ERROR 1	- jelzi, hogy megsérült a rendszer; ujra be kell tölteni kazettáról
TAPE ERROR	- kazettáról történő olvasásnál jelzi a szalag hibáját
FORMAT ERROR	- a beolvasott szalag nem zFORTH formátumu
MODE ERROR	- azt jelzi, hogy a szó használata azon a helyen nem megengedett /pl.: interpreter módban az IF, LOOP, .../

B. függelék: zFORTH memória térkép



1. ábra

C. függelék: zFORTH rendszer változók

rel. cim	név	jelentés
-22	work	munkaterület az egyes gépi kódu rutinokhoz
-16	DRIVE	aktuális magnetofon egység száma
-14	READY	készenléti flag
-1Ø	rsp	return stack pointer
Ø	STATE	interpreter állapot flag
2	BLK	betöltés alatt álló blokk száma
4	PUNCT	beolvasott szám tipus
6	emit	aktuális outputt driver rutin cime
8	key	aktuális input driver rutin cime
1Ø	?terminal	aktuális input teszt rutin cime
12	h	dictionary pointer
14	link	dictionary link első elemének cime
16	sysh	user dictionary kezdő cime
18	sysl	user dictionary link kezdő cime
2Ø	top	user dictionary vége
22	tib	terminál input buffer kezdete
24	width	definiált szó nevének maximális hossza
26	num	szám konvertáló szó cfa-ja
28	err	hibakezelő flag
3Ø	elink	hibaüzenet lánc kezdete
32	hld	kimeneti számkonverzió segédpointer
34	psc	strukturavédelem segédváltozója
36	pscd	strukturavédelem segédváltozója
38	BASE	aktuális számrendszer
4Ø	>IN	WORD segédváltozója
44	rndseek	véletlenszám generátor változója
46	twork	szalag beolvaló munkaterülete

Megj.: A nagybetüvel irt változók azonos néven elérhetőek zFORTH-ban is.

D. függelék: zFORTH lebegőpontos kiterjesztés V1.Ø

Bevezetés

A zFORTH lebegőpontos aritmetikai kiterjesztése lehetővé teszi valós számok ábrázolását és a velük történő számolást. Az alapmüveleteken kivül használhatóak a standard függvények is.

A FORTH megszokott szolgáltatásain tul, módot ad az ugynevezett kalkulátor üzemmód használatára, melyben egyedi számitások végezhetőek gyorsan és kényelmesen, a zFORTH erőforrások támogatása mellett.

A zFORTH V3.1 jelzésü változata ezt a kiterjesztést a kazettáról történő betöltés után már tartalmazza.

Számábrázolás

A lebegőpontos számok ábrázolása négy byte-on történik, azaz egy dupla szóban. Mivel a duplapontosságu számok ábrázolása szintén négy byte-on történik, ahol az lehetséges, kezelésükhöz azonos FORTH szavakat használunk.

Az ábrázolható legnagyobb illetve legkisebb abszolut értékü szám

1.7Ø141E+38 illetve 2.93874E-39

Ez, akárcsak az egyes függvények pontossága megegyezik azzal, amit a BASIC nyujt egyszeres pontosságu változók esetén, azaz hat decimális számjegy.

Változók és konstansok

Definiálásukra két szó szolgál :

2VARIABLE xxx	/ fp -- /	xxx nevü, fp kezdőértékü változó definiálása
2CONSTANT xxx	/ fp -- /	xxx nevü, fp értékü konstans definiálása

Megj.: A továbbiakban a lebegőpontos számokat fp-vel jelöljük, a szövegben pedig számnak hivjuk, ahol ez félreérthető lenne.

Veremtár müveletek

2DROP	/ fp -- /	törli a stack tetején lévő számot
2DUP	/ fp -- fp fp /	megduplázza a stack legfelső elemét
2SWAP	/ fp1 pf2 -- fp2 fp1 /	megcseréli a stack legfelső két elemét
2OVER	/ fp1 fp2 -- fp1 fp2 fp1 /	kiemeli a veremtárban lévő második számot
2ROT	/fp1 fp2 fp2 -- fp2 fp3 fp1 /	körbeforgatja a stack legfelső három elemét

Memória müveletek

2!	/ fp u -- /	elhelyezi fp-t a memória u cimétől kezdődően
2@	/ u -- fp /	betölti a stack-be a memória u cimén lévő számot

Aritmetikai alapmüveletek

F+	/ fp1 fp2 -- fp /	fp=fp1+fp2
F-	/ fp1 fp2 -- fp/	fp=fp1-fp2
F*	/ fp1 fp2 -- fp /	fp=fp1*fp2
F/	/ fp1 fp2 -- fp /	fp=fp1/fp2

Összehasonlitások

FØ=	/ fp -- f /	f=1, ha fp nulla f=Ø egyébként
FØ<	/ fp -- f/	f=1, ha fp negativ F=Ø, ha fp pozitiv
FØ>	/ fp -- f /	f=1, ha fp nagyobb nullánál f=Ø, ha fp nulla vagy negativ
F=	/ fp1 fp2 -- f /	f=1, ha fp1 és fp2 azonos f=Ø egyébként
F<	/ fp1 fp2 .. f /	f=1, ha fp1 kisebb mint fp2 f=Ø egyébként
F>	/ fp1 fp2 -- f /	f=1, ha fp1 nagyobb mint fp2 f=Ø egyébként
FMIN	/ fp1 fp2 -- fp/	fp=fp1 és fp2 közül a kisebb
FMAX	/ fp1 fp2 -- fp/	fp=fp1 és fp2 közül a nagyobb

Függvények

FABS	/ fp1 -- fp2/	abszolutérték képzés
FNEGATE	/ fp1 -- fp2 /	előjelváltás
LN	/ fp1 -- fp2 /	természetes alapu logaritmus
EXP	/ fp1 -- fp2 /	LN inverze
SQR	/ fp1 -- fp2 /	négyzetgyök
PWR	/ fp1 fp2 -- fp /	Hatványozás, $fp=fp1^{fp2}$
INT	/fp1 -- fp2 /	egészrész képzés
FRAC	/ fp1 -- fp2 /	törtrész leválasztása

SIN	/ fp1 -- fp2 /	szinusz
COS	/ fp1 -- fp2 /	koszinusz
TAN	/fp1 -- fp2/	tangens
ATN	/ fp1 -- fp2 /	TAN inverze

A trigonometrikus függvények argumentumát radiánban kell megadni illetve az radiánban áll elő. A fok-radián átszámitást két zFORTH szóval végezhetjük el:

R>D	/ fp1 -- fp2 /	fp1= a szög radiánban fp2= a szög fokban
D>R	/fp1 -- fp2 /	fp1= a szög fokban fp2= a szög radiánban

Előre definiált konstansként rendelkezésre áll a p1 értéke:

PI	/ -- fp /	elhelyezi a veremtárba a pi lebegőpontos értékét

Tipuskonverziók

A lebegőpontos és a szavas, 16-bites egész számok közötti átalakitásra szolgál:

S>F	/ n -- fp /	egész szám konvertálása lebegőpontossá
F>S	/ fp - - n /	lebegőpontos szám konvertálása egésszé

Amennyiben be van töltve a duplapontosságu - double precision - kiterjesztés is, használhatóak a következő szavak is:

D>F	/ d -- fp /	32-bites egész szám átalakitása lebegőpontossá
F>D	/ fp -- d /	lebegőpontos szám konvertálása 32-bites - kétszeres pontosságu - egésszé

A lebegőpontos értékek nem alakithatóak át egész tipusuakká akkor, ha nem férnek bele azok ábrázolási tartományába. Ilyen esetekben

* OVERFLOW

hibaüzenetet kapunk és a program futása megszakad.

Számok beolvasása

A BASIC INPUT utasitásához hasonló a F# IN szó, mely egy lebegőpontos számot olvas be a billentyüzetről :

F# IN / -- fp /

A számok leirása a BASIC-ban megszokott módon történhet. A gépelési hibák javithatóak a "kurzor balra" billentyüvel. Az input végét a <NEWLINE> leütése jelzi. Hibásan megadott szám esetén

WRONG, TRY AGAIN:

üzenet iródik ki, ami után megismételhető az input.

FNUMBER

Karaktersorozat lebegőpontos számmá konvertálására szolgál.

FNUMBER / u -- fp /

Hasonlóan a NUMBER-hez, a memória u+1 cimén kezdődő és szóközzel vagy null karakterrel lezárt stringet alakit át. Ha ez számként nem értelmezhető,

* UNKNOWN WORD

hibajelzést kapunk.

A PUNCT nevü rendszerváltozó értéke FNUMBER végrehajtása után 2 lesz, ami azt jelzi, hogy lebegőpontos számot olvastunk be.

(FNUM)

(FNUM) / u -- 1 /
/ u -- fp Ø /

A memória u+1 cimén kezdődő stringet konvertálja át lebegőpontos számmá és elhelyezi a veremtárban / fp /. A konverzió sikertelen voltát a stack legfelső, 1 / igaz / értéke jelzi.

Megj.: Valamennyi bemeneti számkonverziós szó tizes számrendszerben dolgozik BASE tartalmától függetlenül.

Számok kiiratása

F.	/ fp -- /	lebegőpontos szám kiiratása az aktuális kimeneti perifériára
F.R	/ fp u -- /	lebegőpontos szám kiirása u szélességü mezőn belül jobbra igazitva

Függetlenül BASE értékétől, a kiiratás mindig tizes számrendszerben történik.

A következő szavak segitségével a kiirt számok formátuma vezérelhető:

AUTO	/ -- /	automatikus formátum kezelés; hatása azonos a BASIC formátum nélküli működésével
		ez az alapértelmezés
SCIENTIC	/ -- /	valamennyi számnál kiiródik a decimális exponens is
FIXED	/ -- /	fixpontos kiiratást ir elő, a beállitott számu tizedesjeggyel
DIGIT	/ u -- /	SCIENTIC és FIXED tipusu kiiratáskor hatásos; beállitja a kiirt tört jegyek számát u-ra, a tizedespontot is beleértve

COMMAS	/ -- /	FIXED tipusu kiiratásnál az egész rész minden harmadik számjegye közé vessző iródik
NOCOMMAS	/ -- /	törli a COMMAS szó hatását
ISIGN	/ -- /	végrehajtása után a pozitiv számok elé kiiródik a + jel
ESIGN	/ -- /	az előjel a számok után iródik ki
NOSIGN	/ -- /	hatályon kivül helyezi az ESIGN és ISIGN kijelöléseket

Kalkulátor üzemmód

Alapállapotban a zFORTH nem ismeri fel közvetlenül a beirt lebegőpontos számokat, azokat normál vagy kétrészes pontosságuként próbálja meg értelmezni.

Kalkulátor üzemmódban valamennyi szám lebegőpontosként értelmeződik mind definició belsejében mind azon kivül. Ezért a zFORTH-t kalkulátorként használhatjuk különböző számitásokhoz, melyek elvégzéséhez nem kivánunk uj szavakat definiálni, de leegyszerüsödik a lebegőpontos számok megadása is /nem kell az F# IN-t használni./

Az üzemmódok átkapcsolása a

SETCALC és a NOCALC

szavakkal történik. Előbbi bekapcsolja, utóbbi pedig kikapcsolja a kalkulátor üzemmódot. Mindkét szó IMMEDIATE tipusú, vagyis definición belül, bekapcsolt compiler mellett is végrehajtásra kerülnek.

Hibaüzenetek

A lebegőpontos kiterjesztéssel a hibaüzenetek kibővülnek.

* OVERFLOW	Túlcsordulás, az eredmény nagyobb, mint a legnagyobb ábrázolható szám. Ezt az üzenetet kapjuk akkor is, amikor a lebegőpontos szám nem konvertálható át 16 vagy 32 bites egésszé.

* DIVIDE BY Ø	Lebegőpontos osztás nullával.
* NUMEROUS ERROR	Számolási hiba. Ez keletkezik akkor, amikor egy függvény operandusa hibás, nem tartozik bele annak értelmezési tartományába. Például gyökvonás negativ számból, negativ szám logaritmusa, stb.

E. függelék: zFORTH ASSEMBLER V1.Ø

Az ASSEMBLER a HT-1080Z tipusu iskolaszámitógépen futó zFORTH programozási nyelv kiterjesztése. Használatával lehetővé válik olyan szavak definiálása, melyek törzse gépi kódu, igy megoldhatóak speciális, a magasszintü definicióknál rövidebb végrehajtási időt igénylő feladatok is.

Az assemblert a zFORTH rendszer alatt kazettáról a SYSTEM utasitással tölthetjük be :

1Ø SYSTEM <NEWLINE>

A betöltés kezdetét a

LOADING zFORTH ASSEMBLER

végét pedig a képernyőn megjelenő

LOADED .....

üzenet jelzi.

Szavak definiálása

A gépi kódu szavak definiálása a CODE utasitással kezdődik, mely után az uj szó nevét kell megadni az általános zFORTH szabályok szerint, mig lezárása a C; szóval történik.

Fontos különbség a CODE és a : illetve a C; és a ; között, hogy az assembly szintü utasitások forditása közben a zFORTH interpreter üzemmódban marad, ami felismerhető az OK promt jelről. Ebből következően a zFORTH lehetőségei, szolgáltatásai maradéktalanul kihasználhatóak. Ez például az operandusok számitásánál jelenthet nagy segitséget.

Az utasitáskészlet

A zFORTH ASSEMBLER tartalmazza a Z8Ø processzor utasitásainak zömét. Nem kerültek megvalósitásra az indexregiszterekkel / IX és IY / kapcsolatos müveletek és azok, melyek ritkán használatosak vágy már meglévő utasitásokkal könnyen helyettesithetőek.

Az alkalmazott mnemonikok többnyire megegyeznek az INTEL 8Ø8Ø processzor jelöléseivel illetve ahol ez nem lehetséges, a Z8Ø szabványos mnemonikjaival. Néhány esetben, ahol indokolt volt, egyedi jelülést kellett bevezetni.

Az implementált utasitások felsorolása és jelöléseik a függelék végén lévő táblázatban találhatóak.

Az operandusok kijelölése

A FORTH környezet sajátosságaihoz igazodva az operandusok, cimek és regiszterek illetve regiszterpárok megadása a gépi utasitás előtt történik. Más szavakkal ez azt jelenti, hogy a veremtárban ezeket még az utasitás leforditása előtt el kell helyezni.

Lássunk néhány példát :

8Ø8Ø assembler	Z8Ø assembler	żFORTH assembler
MVI A, 9	LD A,9	9 a mvi
ORA E	OR E	e ora
PUSH PSW	PUSH AF	psw push
LXI H,1Ø24	LD HL,1Ø24	1Ø24 h lxi
MOV L,E	LD L,E	e l mov
IN Ø	IN (A),Ø	Ø in
RST 1	RST 8	1 rst
CALL ØBØ39H	CALL ØBØ39H	HEX BØ39 call

Cimkék, programstrukturák

Hasonlóan a FORTH-hoz, az ASSEMBLER-ben sem kell cimkéket és abszolut hivatkozási cimeket használni. A feltételes prog ramvégrehajtás és a ciklusok megadása a zFORTH-ban megszokottakkal analóg módon történik.

A következő strukturavezérlő utasitások használhatóak:

a/ cond if ... w1 ... then w2

Végrehajtódik w1 abban az esetben, ha teljesül a cond feltétel, majd a program w2-vel folytatódik.

b/ cond if ... w1 ... else ... w2 ... than w3

A cond feltétel teljesülése esetén w1, nem teljesülésekor pedig w2 hajtódik végre, majd a program folytatódik w3-mal.

c/ begin ... w ... again

Végtelen ciklusban ismétli a w szót vagy szócsoportot.

d/ begin ... w ... cond until wa

Ismétlődik w végrehajtása mindaddig, amig a cond feltétel igaz nem lesz.

e/ begin ... w1 ... cond while ... w2 ... repeat w3

w1 végrehajtása után, amennyiben a cond feltétel igaz, végrehajtja w2-t majd ismét w1 kerül sorra. A feltétel nem teljesülésekor a ciklus futása abbamarad és a while után w3-mal folytatódik a program.

Feltételkódok

z - nulla	nz - nem nulla
p - pozitiv	mi - negativ
cs - átvitel van	nc - átvitel nincs
pe - paritás páros	po - paritás páratlan

Visszatérés az interpreterbe

Minden szóban, melyet CODE-dal definiáltunk, meg kell adni legalább egy olyan pontot, ahonnan a program visszatér a zFORTH-ba, célszerüen a szó végén, közvetlenül a C; előtt. Erre a célra két utasitást használhatunk:

NEXT - visszatérés az interpreterhez a veremtár ellenőrzésével

XNEXT - visszatérés az interpreterhez a veremtár ellenőrzése nélkül.

Az utóbbit a müködés gyorsitása érdekében alkalmazzuk akkor, amikor a szó a stack-ben lévő elemek számát biztosan nem változtatja meg.

Szubrutinok definiálása

A zFORTH ASSEMBLER-rel nemcsak végrehajtható szavakat, hanem szubrutinokat is definiálhatunk SUBR definiáló szóval. A szubrutin végét itt is a C; jelzi. A hivó programba való visszatérés természetesen itt is, mint a hagyományos assembly programoknál egy ret utasitás végrehajtásával történik, melyet meg kell adnunk.

A definiált szubrutin nevét leirva, pontosabban a szót végrehajtva a veremtárba betöltődik az első végrehajtható utasitás cime, melyet felhasználhatunk az utána következő call utasitásban.

Ily módon belső szubrutinok használata esetén is elkerülhető az abszolut cimek használata illetve ismerete, amint az az 5. és 6. mintapéldában látható.

Regiszterek használata

A gépi kódu szavak nem ronthatják el a BC regiszterpár tartalmát. A többi regisztere a processzornak szabadon használható.

Mintapéldák

1./ 4+ / n -- n+4 / Hozzáad négyet a stack tetején lévő számhoz.

```
CODE 4+  h pop  4 d lxi  d dad  h push  XNEXT  C;
```

2./ SGN / n -- sgn / Betölti a stack-be a signum /n/ függvény értékét, vagyis
sgn = Ø , ha n= Ø
sgn = 1 , ha n > Ø
sgn = -1, ha n < Ø

```
CODE SGN  h pop  Ø d lxi  h a mov  l ora
          nz if  h a mov  a ora
                 mi if  d dcx  else  d inx  then
                then
          d push
      XNEXT  C ;
```

3./ UCS / adr cnt -- / A memóri adr cimétől kezdődő cnt hosszuságu stringben a kisbetüket nagybetüvé konvertálja.

```
CODE UCS  d pop  h pop
          begin  d a mov  e ora  nz while  HEX
                 m a mov  61 cpi
                 nc if  7B cpi
                        cs if  2Ø sui  a m mov
                           then
                     then
                 h inx  d dcx  repeat
      NEXT  C;
```

4./ OCTAL / -- /

A számrendszer alapszámát átkapcsolja oktálisba, azaz nyolcas alapu lesz a számkiirás és beolvasás.

```
CODE OCTAL  8 h lxi  BASE shld  XNEXT  C;
```

5./ SPAR

Páros paritásura igazitja a HL cimen lévő byte-t.

```
SUBR SPAR  m a mov  a ora  rpe  128 xri  a m mov  ret
    C;
```

6./ parity / adr - cnt - - /

Páros paritásura alakitja a memória adr cimétől kezdődő, cnt hosszuságu karaktersorozatot.

```
CODE parity  d pop  h pop
             begin  d a mov  e ora
               nz while  SPAR call  h inx  d dcx
             repeat
    NEXT  C;
```

7./ UREC / -- c /

Beolvas egy karaktert a 8ØH és 81H cimen lévő 8251 tipusu USART-ról a veremtárba.

```
CODE UREC  begin  HEX  81 in  1 ana  nz until
             8Ø in  6F ani  a l mov  Ø h mvi
    NEXT  C;
```

8./ CSWAP / n1 -- n2 /

Megcseréli a stack tetején lévő szó két byte-ját.

```
CODE CSWAP  d pop  d a mov e d mov  a e mov  XNEXT C;
```

UTASITÁSOK

zFORTH	Z8Ø	8Ø8Ø
nop	NOP	NOP
hlt	HALT	HLT
di	DI	DI
ei	EI	EI
pchl	JP /HL/	PCHL
sphl	LD SP,HL	SPHL
xthl	EX SP,HL	XTHL
xchg	EX DE,HL	XCHG
daa	DAA	DAA
cma	CPL	CMA
stc	SCF	STC
cmc	CCF	CMC
y add	ADD s	ADD y
y adc	ADC s	ADC y
y sub	SUB s	SUB y
y sbb	SBC s	SBB y
y ana	AND s	ANA y
y ora	OR s	ORA y
y xra	XOR s	XRA y
y cmp	CP s	CMP y
z dad	ADD HL,dd	DAD z
w pop	POP qq	POP w
w push	PUSH qq	PUSH w
t stax	LD /vv/,A	STAX t
t ldax	LD A,/vv/	LDAX t
y inr	INC s	INR y
y dcr	DEC s	DCR
z inx	INC dd	INX z
z dcx	DEC dd	DCX z
b rst	RST p	RST b

zFORTH	Z8Ø	8Ø8Ø
n out	OUT /n/,A	OUT n
n in	IN A,/n/	IN n
n adi	ADD n	ADI n
n aci	ADC n	ACI n
n sui	SUB n	SUI n
n sbi	SBC n	SBI n
n ani	AND n	ANI n
n ori	OR n	ORI n
n xri	XOR n	XRI n
n cpi	CP n	CPI n
nn shld	LD /nn/,HL	SHLD nn
nn lhld	LD HL,/nn/	LHLD nn
nn sta	LD /nn/,A	STA nn
nn lda	LD A,/nn/	LDA nn
nn cnz	CALL NZ,nn	CNZ nn
nn cz	CALL Z,nn	CZ nn
nn cnc	CALL NC,nn	CNC nn
nn cc	CALL C,nn	CC nn
nn cpo	CALL PO,nn	CPO nn
nn cpe	CALL PE,nn	CPE nn
nn cp	CALL P,nn	CP nn
nn cm	CALL M,nn	CM nn
nn call	CALL nn	CALL nn
rnz	RET NZ	RNZ
rz	RET Z	RZ
rnc	RET NC	RNC
rc	RET C	RC
rpo	RET PO	RPO
rpe	RET PE	RPE
rp	RET P	RP
rm	RET M	RM
ret	RET	RET

zFORTH	Z8Ø	8Ø8Ø
nn jmp	JP nn	JMP nn
y'y mov	LD s,s'	MOV y,y'
n y mvi	LD s,n	MVI y,n
nn z lxi	LD dd,nn	LXI z,nn
rld	RLD	-
rrd	RRD	-
rlca	RLCA	RLC
rrca	RRCA	RRC
rla	RLA	RAL
rra	RRA	RAR
y rlc	RLC s	-
y rrc	RRC s	-
y rl	RL s	-
y rr	RR s	-
y sla	SLA s	-
y sra	SRA s	-
y srl	SRL s	-
lai	LD A,I	-
loa	LD I,A	-
exx	EXX	-
ex	EX AF,AF'	-
neg	NEG	-
imØ	IMØ	-
im1	IM1	-
im2	IM2	-
reti	RETI	-
b y bit	BIT b,s	-
b y set	SET b,s	-
b y res	RES b,s	-

zFORTH	Z8Ø	8Ø8Ø
ldi	LDI	-
ldir	LDIR	-
ldd	LDD	-
lddr	LDDR	-
cpx	CPI	-
cpir	CPIR	-
cpd	CPD	-
cpdr	CPDR	-
z dac	ADC HL,dd	-
z dsc	SBC HL,dd	-
x inp	IN r,/C/	-
x outp	OUT /C/,r	-

A használt röviditések :

dd = BC,HL,DE,SP	W = b,h,d,psw
qq = BC,HL,DE,AF	z = b,h,d,sp
r = A,B,C,D,E,H,L	x = a,b,c,d,e,h,l
s = A,B,C,D,E,H,L,/HL/	y = a,b,c,d,e,h,l,m
n = 8-bites érték	nn = 16-bites érték
b = Ø,1,2,3,4,5,6,7	p = ØØH,Ø8H,1Øh ...
vv = BC,DE	t = b,d

F. függelék: zFORTH duplapontosságu kiterjesztés V1.Ø

Bevezetés

A zFORTH ezen kiterjesztése lehetővé teszi a FORTH szabvány kétszeres pontosságu - double precision - számokat kezelő szavainak használatát. Segitségükkel több értékes jeggyel dolgozhatunk és a pontosság mellett megnő az ábrázolási tartomány is.

Ne felejtsük el, hogy ezek az egyszeres pontosságuaknál lassabban hajtódnak végre és csökkentik a rendelkezésre álló szabad memóriaterületet is, ezért csak indokolt esetben használjuk.

Betöltése kazettáról a

DECIMAL 2Ø SYSTEM NEWLINE

utasitás sorral történik.

A betöltés kezdetét a

LOADING DOUBLE-PRECISION EXTENSION

végét pedig a

LOADED, nnnn BYTES FREE

üzenet jelzi, ahol nnnn a szótárban rendelkezésre álló szabad memódia mérete.

Betöltés után valamennyi addig definiált szó a rendszer részévé válik és nem törölhető a FORGET, EMPTY, PURGE vagy COLD szavakkal.

Számábrázolás

A kétszeres pontosságu számok ábrázolása 32 biten, azaz 4 byte-on történik. Akárcsak az egyszeres pontosságuak, lehetnek előjelesek vagy előjel nélküliek. A veremtár tetején mindig a szám legnagyobb helyiértékü két byte-ja található.

Az ábrázolási tartomány előjeles számoknál

-2147483648 ÷ +2147483647

előjel nélküli számoknál pedig

Ø ÷ +4294967295

vagyis mindkét esetben legalább kilenc decimális számjegy.

Kétszeres pontosságu számok megadása

A duplapontosságu számokat az input sorban mind definició belsejében, mind azon kivül a számjegyek közé illetve közvetlenül a szám mögé irt megkülönböztető irásjelekkel adunk meg, melyek:

. , / : -

Alkalmazhatóak vegyesen és egy számon belül lehet belőlük több is. Helyük a beolvasott érték szempontjából közömbös, azt nem befolyásolja.

Például a következő számok értéke megegyezik:

83.11.19 831119. 83111.9 83-11-19 83.11:19 83/11/19

Azt, hogy az utolsónak beolvasott szám milyen tipusu volt, a PUNCT nevü rendszerváltozó tartalmazza. Értéke nulla, ha a szám 16-bites volt, 32-bites esetén pedig egy.

Tipuskonverzió

Kétszeres pontosságu számokat egyszeres pontossággá a felső 16 bit levágásával alakithatunk át, mely a DROP szóval oldható meg. Helyes eredményt természetesen csak akkor kapunk, ha az eredeti szám ábrázolható volt 16 biten.

S>D	/ n -- d /	előjeles 16-bites számot alakit át 32-bitessé

Előjel nélküli szimplapontosságu számot előjel nélküli duplapontossággá egy nulla rátöltésével alakithatunk át.

Amennyiben be van töltve a lebegőpontos kiterjesztés is, használhatóak a következő szavak is:

F>D	/ fp -- d /	lebegőpontos szám konvertálása duplapontossággá
D>F	/ d -- pf /	duplapontosságu szám konvertálása lebegőpontossá

Megj.: A továbbiakban az előjeles számot d-vel, az előjel nélkülit pedig du-val jelöljük.

Változók és konstansok

Definiálásukra két szó áll rendelkezésre:

2VARIABLE xxx	/ d -- /	xxx nevü, d kezdőértékü változó definiálása
2CONSTANT xxx	/ d -- /	xxx nevü, d értékü konstans definiálása

Megj.: A továbbiakban a duplapontosságu számokat d-vel vagy du-val jelöljük attól függően, hogy előjelesek vagy előjel nélküliek és a szövegben az egyszerüség kedvéért számoknak hívjuk, kivéve, ahol ez félreérthető lenne.

Veremtár müveletek

2DROP	/ d -- /	törli a stack legfelső elemét
2DUP	/ d -- d d /	megduplázza a stack tetején lévő számot
2SWAP	/ d1 d2 -- d2 d1 /	megcseréli a stack legfelső két elemét
2OVER	/ d1 d2 -- d1 d2 d1 /	kiemeli a stack második elemét
2ROT	/ d1 d2 d3 -- d2 d3 d1 /	körbeforgatja a stack legfelső három elemét

Memória müveletek

2!	/ d u -- /	elhelyezi d-t a memória u cimén
2@	/ u -- d /	betölti a stack-be a memória u cimén lévő számot

Összehasonlitások

DØ=	/ d -- f /	f = 1, ha d nulla f = Ø, egyébként
DØ<	/ d -- f /	f = 1, ha d kisebb mint nulla f = Ø egyébként
D=	/ d1 d2 -- f /	f = 1, ha d1 és d2 egyenlő f = Ø egyébként
D<	/ d1 d2 -- f /	f = 1, ha d1 kisebb mint d2 f = Ø egyébként
DU<	/ du1 du2 -- f /	f = 1, ha du1 kisebb mint du2 f = Ø egyébként /előjel nélküli/
DMIN	/ d1 d2 -- d /	d = d1 és d2 közül a kisebb
DMAX	/ d1 d2 -- d /	d = d1 és d2 közül a nagyobb

Aritmetika

DABS	/ d1 -- d2 /	abszolut érték képzése
DNEGATE	/ d1 -- d2/	előjel váltás
D+	/ d1 d2 -- d/	d=d1+d2
D-	/ d1 d2 -- d /	d=d1-d2
U*	/ u1 u2 -- du /	du=u1xu2 előjel nélküli szorzás
M*	/ n1 n2 -- d /	d=n1xn2 előjeles szorzás
M+	/ d1 n -- d /	d=d1+n vegyes tipusu összeadás

U/MOD	/ du1 u1 -- u2 u3 /	a du duplapontos számot elosztja u1-el; u2=maradék u3= hányados
M/	/ d n1 -- n1 /	n1=d/n előjeles osztás

Számok kiiratása

D.	/ d -- /	duplapontos, előjeles szám kiiratása
DU.	/ du -- /	duplapontos, előjel nélküli szám kiiratása
D.R	/d u -- /	duplapontos szám kiiratása u szélességü mezőn belül jobbra igazitva, előjellel
DU.R	/ du u -- /	mint D.R, csak előjel nélküli a kiiratás

Számok beolvasása

Duplapontosságu számok beolvasására illetve előállitására az eddig megismerten tul további szavak állnak rendelkezésre.

(DNUM) / u -- d Ø / vagy / u -- 1 /

A memória u+1 cimén kezdődő és szóközzel vagy null karakterrel lezárt stringet konvertálja át duplapontos számmá és elhelyezi a veremtárban. A sikeres konverziót a stack tetején lévő nulla jelzi. Ha az átalakitás nem végezhető el, nem ad vissza számot csak egy 1 / "igaz" / logikai flag-et.

A konverzió mindig az aktuális számrendszerben történik.

DNUMBER / u -- d / vagy / u -- n /

A memória u+1 cimén kezdődő és szóközzel vagy null karakterrel lezárt stringet konvertálja át számmá. Az eredmény 16 vagy 32 bites az inputtól függően - volt-e benne irásjel. Az eredmény duplapontos, ha a PUNCT rendszerváltozó tartalma 1, egyszeres pontosságu, ha nulla. Sikertelen konverzió esetén hibajelzést ad.

D# IN / -- d /

A billentyüzetről vár egy szám megadására. Az eredmény mindig duplapontosságu, a beirt szám tipusát PUNCT tartalmazza. Hibás szám esetén

WRONG, TRY AGAIN :

üzenetet kapunk, ami után megismételhetjük a szám megadását.

IN / -- n /

Funkciója megegyezik D# IN -el, kivéve, hogy csak egyszeres pontosságu számot fogad el.

PUNCT

Rendszerváltozó, tartalma az utolsónak beolvasott vagy stringből átolvasott szám tipusát jelzi. Lásd DNUMBER.

G. függelék: zFORTH nyomtató illetsztés V1.1

Ennek a kiterjesztésnek a használata lehetőséget ad a számitógéphez kapcsolt mátrixnyomtatón szövegek kiirására illetve screen-ek kilistázására.

Betöltése kazettáról a

8 SYSTEM NEWLINE

utasitás sorral történik; a forditás végét a

PRINTER EXTENSION LOADED, nnnn BYTES FREE

üzenet jelzi, ahol nnnn a szabadon maradt memória méretét adja meg decimálisan.

A nyomtató kezeléséhez rendelkezésre álló szavak a következőek:

PRINT / — /

Aktuális kimeneti eszközként a nyomtatót jelöli ki. Hatására minden karakter a képernyő helyett a nyomtatón iródik ki. A korábban használt kiiratást vezérlő szavak értelemszerüen itt is müködnek /CR, PAGE, TYPE, EMIT, SPACE, SPACES, ." , stb./. A kijelölés érvényes a CRT szó végrehajtásáig vagy a RESET gomb megnyomásáig.

CRT / -- /

Aktuális kimeneti eszközként kijelöli a képernyőt. Ez a zFORTH alapállapota. A RESET gomb megnyomásakor automatikusan végrehajtásra kerül.

-TRAILING / addr n1 -- addr n2 /

Meghatározza az addr cimen kezdődő, n1 hosszuságu string n2 tényleges hosszát ugy, hogy a szöveg végén ne legyenek szóköz

karakterek. Közvetlenül TYPE utasitás előtt használva elkerülhetjük a nyomtató fej felesleges mozgását, vagyis meggyorsithatjuk a kiiratást egyes nyomtatók esetén.

LLIST / n -- /

Screen nyomtatóra történő kiiratását előiró szó. Előtte nem kell kiadni a PRINT utasitást. A screen kijelölése a BLOCK szónál mondottakkal azonos módon történik, vagyis

n > Ø esetén listázásra kerül az n sorszámu screen,

n = Ø esetén listázásra kerül a buffer pillanatnyi tartalma

n < Ø esetén listázásra kerül a szalagon következő legelső screen tartalma.

Felelős kiadó: Somlyay Endre I/85.szept.