Ontsnappingen in forth
EXIT LEAVE WHILE THROW

Voor Forth-gg Bilthoven augustus 2023 an

EXIT

ALIGNED (32bits), beginnerscode

```
: ALIGNED ( adr -- adr2 )   dup 4 mod
    dup 3 = if drop 1 + else
    dup 2 = if drop 2 + else
    dup 1 = if drop 3 + else
    dup 0 = if drop     then then then then ;       \ 47
```

Testmateriaal

```
-5 aligned -4 aligned                              . .
-3 aligned -2 aligned -1 aligned 0 aligned . . . .
 1 aligned  2 aligned  3 aligned 4 aligned . . . .
 5 aligned                                         .
\ 32bits
 hx 8000,0003 aligned hx u.
 hx 7FFF,FFFD aligned hx u.
\ 16bits
 hx 8003 aligned hx u.
 hx 7FFD aligned hx u.
```

Met EXIT

```
: ALIGNED ( adr -- adr2 )   dup 4 mod
    dup 3 = if drop 1 + exit then
    dup 2 = if drop 2 + exit then
    dup 1 = if drop 3 + exit then
    dup 0 = if drop     exit then ;                \ 45
```

Gestroomlijnd

Twee DUP's, twee DROP's en twee EXIT's minder

```
: ALIGNED ( adr -- adr2 )   dup 3 and
    dup 3 = if drop 1 + exit then
    dup 2 = if drop 2 + exit then
        1 = if      3 +      then ;                \ 34
```

Nog iets korter

```
: ALIGNED ( adr -- adr2 )   dup 3 and
    dup 3 = if drop  1+ exit then
    dup 2 = if        + exit then
            if      3 +      then ;                \ 26
```

Brain wave

Per bit naar boven afronden, getriggerd door DUP 2 = IF +

```
: ALIGNED ( adr -- adr2 )
    dup 1 and +
    dup 2 and + ;                                 \ 10
```

Brain wave 2

Drie varianten met -4 AND zonder IF
(Bij -4 zijn alle bits gezet behalve de laagste twee)

```
: ALIGNED ( adr -- adr2 ) 3 +    -4 and ;          \ 6
: ALIGNED ( adr -- adr2 ) 1-     -4 and cell+ ;    \ 5
: ALIGNED ( adr -- adr2 ) negate -4 and negate ;   \ 5
```

✺ Vraag: hoe zou de 64 bits ALIGNED er uitzien?

LEAVE

```
... do ... if ... leave then ... loop
... do ... ?leave ... loop
... do ... if ... i leave then ... loop ?
... do ... if ... i unloop exit then ... loop ... ;
```

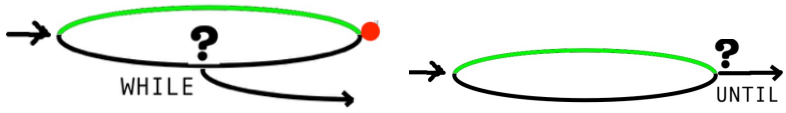
```
: TEL1 ( -- ? )
    100,0000 0
    do key?   if key drop i leave then
    loop ;
```

Oplossingen

```
: TEL2 ( -- i | true )
    100,0000 0
    do key?   if key drop i unloop exit then
    loop true ;

: TEL3 ( -- i | true )
    true             \ dummy
    100,0000 0
    do key?   if key drop i and leave then
    loop ;
```

WHILE



Je kunt een eenvoudige BEGIN-WHILE-REPEAT altijd herschrijven als een BEGIN-UNTIL door de aanvangsvoorwaarde buiten de lus te testen.

```
: TYPE1 ( a n -- )
    bounds
    BEGIN   2dup <>   WHILE count emit   REPEAT
    2drop ;                                    \ 10

: TYPE2 ( a n -- )
    dup if  bounds
            BEGIN   count emit 2dup =   UNTIL
    then    2drop ;                            \ 11
```

```
begin ... while ... repeat
begin ... while ... again then
```

```
begin ...
      while ...
      while ...
again then then ?
```

```
begin ...
      while ...
      while ...
again then ...
      then ... ?
```

Oplossingen

```
begin ...
      while ...
      while ...
again else ... then
      else ... then
```

```
begin ...
      while ...
      while ...
again then ... exit
      then ... ;
```

Of met UNTIL

```
begin ...
      while ...
      while ...
until ...
      else ... then
      else ... then
```

```
begin ...
      while ...
      while ...
until ... exit
      then ... exit
      then ... ;
```

```
decimal
100    0 value STAP   ?dup drop
to stap

: T ( x -- )
    begin dup .
        dup 9 mod while
        dup 11 mod while
        stap +
    again ." Ik voel me niet gehoord :( "
        else ." elf " then
        else ." negen " then
    drop ;
```

THROW

```
: THROW ( n -- ? ) ?dup if Herstel-CATCH-toestand then ;
```

(Toestand = ip sp rp inputstroom)

CATCH is een EXECUTE die nog wat extra doet

```
: CATCH ( token -- ... 0 | n<>0 )
    Save-toestand-op-de-returnstack
    execute
    Verwijder-toestand-van-de-returnstack
    0 ;
```

```
: FORTH-GG  ( -- n )                \ n in 0..9
    key   ch 0 -   9 over u<
    if 13 throw ( ↯ ) then          \ 13 and throw
    ." F" ;
: BOSLAAN   forth-gg  40    + ." o" ;
: BILTHOVEN boslaan   300   + ." r" ;
: UTRECHT   bilthoven 2000  + ." t" ;
: NL        utrecht   10000 + ." h " ;
```

```
: TEST ( -- )
    ['] nl catch ( ↪ ) 13 =
    if   ." Dit was geen cijfer " exit
    then . ;
```

```
: CIJFER ( -- )
    begin
        ['] nl catch ( ↪ ) 13 <>
    until . ;
```

Je kunt het ook overdrijven

Hailstone-rij van een getal x(0)

- Het eerste getal is x(0) (unsigned)
- Berekening van de volgende getallen:
  voor oneven x(i) ➔ x(i+1) = x(i) * 3 + 1
  voor even x(i) ➔ x(i+1) = x(i) / 2
- Stop bij x(n)=1 (zijn er oneindig lange rijen?)
- n is de lengte van de rij

✺ Gevraagd: na hoeveel stappen eindigt een Hailstone-rij die met x(0) begint? Detecteer overflow.

```
\ 4 versies voor berekenen van het Volgende Getal
: VG0 \ x(i) -- x(i+1)                \ Zonder overflowtest
    dup 1 and
    if  3 * 1+                        \ Dit kan vastlopen
    else 2/
    then ;

: VG1 \ x(i) -- x(i+1)                \ Met overflowtest
    dup 1 and
    if   3 um* if drop -1 then 1+     \ x*3+1 ( of 0 )
    else 1 rshift                     \ x/2
    then ;

: VG2 \ x(i) -- x(i+1)                \ Zonder IF
    dup 1 and >r
    3 over um* 0<> or 1+              \ x*3+1?
    swap 1 rshift                     \ x/2
    r> pick nip nip ;

: VG3 \ x(i) -- x(i+1)                \ Zonder IF
    dup 1 and >r
    3 over um* 0<> or 1+ r@ 0<> and \ x*3+1? | 0
    swap 1 rshift        r> 0=  and \ x/2    | 0
    or ;
```

Een Hailstone-rij afdrukken

```
' vg1   0 value VG   ?dup drop
to vg
: HS \ x(0) --
    0                              \ teller
    begin 1+ swap vg execute       \ i x(i)
          dup u.
          tuck 2 u<                \ x(i) i vlag
    until .s                       \ x(n) n
    drop ?exit ."  Overflow " ;
```

x(n)=1 ➔ geslaagd
x(n)=0 ➔ overflow of x(0) was 0

<><>