```
; Vocabulary 'forthInterpreter.Private'

; ==== 'forthInterpreter.Private.t.bye.h' ====================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    CODE BYE ( -- )
;
;    Returns to the caller of START
;
; 0:t.bye.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ebye2Eh, @function
  .endif
                                    ; // (label)
_t2Ebye2Eh:

    .pword    0
    .pword    0xffff00
    .pascii   <3>,"bye"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ebye, @function
  .endif

_t2Ebye:
                                                ; // bye
_FIRST_FLASH:
    ; // Restore registers
    mov       #_RBank + 36, w0
    mov       [w0--], w2                        ; // 36 :: 18
    mov       w2, SR
    mov       [w0--], w2                        ; // 34 :: 17
    mov       w2, RCOUNT
    mov       [w0--], w2                        ; // 32 :: 16
    mov       w2, TBLPAG
    mov       [w0--], w15                       ; // 20 :: 15
    mov       [w0--], w14                       ; // 28 :: 14
    mov       [w0--], w13                       ; // 26 :: 13
    mov       [w0--], w12                       ; // 24 :: 12
    mov       [w0--], w11                       ; // 22 :: 11
    mov       [w0--], w10                       ; // 20 :: 10
    mov       [w0--], w9                        ; // 18 ::  9
    mov       [w0--], w8                        ; // 16 ::  8
    mov       [w0--], w7                        ; // 14 ::  7
    mov       [w0--], w6                        ; // 12 ::  6
    mov       [w0--], w5                        ; // 10 ::  5
    mov       [w0--], w4                        ; //  8 ::  4
    mov       [w0--], w3                        ; //  6 ::  3
    mov       [w0--], w2                        ; //  4 ::  2
    mov       [w0--], w1                        ; //  2 ::  1
    mov       [w0]  , w0                        ; //  0 ::  0
    ; // Return to caller of forth.start
  .pword 0xDA4000                   ; // breakPoint t.bye
    nop
    return


; ==== 'forthInterpreter.Private.t.doCol' ====================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;   [ -- d ] (or ENTER)
;    DOCOL :: A code fragment, not a forth primary
;      IP PUSH RSP
;      PFA TO IP
;      NEXT
;
;    NEXT :: A code fragment, not a forth primary
;      [IP] TO W
;      CELL +TO IP
;      W JUMP
;
; 0:t.doCol
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2EdoCol, @function
  .endif
                                    ; // (label)
_t2EdoCol:
                                                ; // doCol
    mov       w8, [--w10]                       ; // IP PUSH RSP
    mov       TBLPAG, w2
    mov       w2, [--w10]

    inc2      w6, w8                            ; // PFA TO IP
    addc      w7, #0, w2
    mov       w2, TBLPAG

_t2Enext:

    tblrdl.w  [w8  ], w6                        ; // [IP] TO W   CELL +TO IP
    tblrdh.w  [w8++], w7

    push      w6                                ; // W JUMP
    push      w7
    return
```

```
; ==== 'forthInterpreter.Private.t.doCon' ===================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
;    A code fragment, not a forth primary
;      [PFA] PUSH PSP
;      NEXT
;
; 0:t.doCon
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2EdoCon, @function
  .endif
                                    ; // (label)
_t2EdoCon:
                                                ; // doCon
    mov       w0, [w9++]                        ; // [PFA] PUSH PSP
    mov       w1, [w9++]

    inc2      w6, w6                            ; // PFA
    addc      w7, #0, w7
    mov       w7, TBLPAG

    tblrdl.w  [w6], w0                          ; // [PFA]
    tblrdh.w  [w6], w1

    goto      _t2Enext                        ; // NEXT


; ==== 'forthInterpreter.Private.t.doVar' ===================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
;    A code fragment, not a forth primary
;      [PFA] PUSH PSP
;      NEXT
;
; 0:t.doVar
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2EdoVar, @function
  .endif
                                    ; // (label)
_t2EdoVar:
                                                ; // doVar
    mov       w0, [w9++]                        ; // [PFA] PUSH PSP
    mov       w1, [w9++]

    inc2      w6, w6                            ; // PFA
    addc      w7, #0, w7
    mov       w7, TBLPAG

    tblrdl.w  [w6], w0                          ; // [PFA]
    tblrdh.w  [w6], w1

    goto      _t2Enext                        ; // NEXT


; ==== 'forthInterpreter.Private.t.doDefer' ==================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    A code fragment, not a forth primary
;      [PFA] PUSH PSP
;      @
;      EXECUTE
;
; 0:t.doDefer
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2EdoDefer, @function
  .endif
                                    ; // (label)
_t2EdoDefer:
                                                ; // doDefer
    mov       w0, [w9++]                        ; // [PFA] PUSH PSP
    mov       w1, [w9++]

    inc2      w6, w6                            ; // PFA
    addc      w7, #0, w7
    mov       w7, TBLPAG

    tblrdl.w  [w6], w0                          ; // [PFA]
    tblrdh.w  [w6], w1

    mov       [w0++], w2                        ; // @
    mov       [w0  ], w1
    mov       w2, w0

    goto      _t2Eexecute                     ; // EXECUTE


; ==== 'forthInterpreter.Private.t.doUser' ===================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
;    A code fragment, not a forth primary
;      [PFA] UP + PUSH PSP
;      NEXT
```

```
;
; 0:t.doUser
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2EdoUser, @function
  .endif
                                    ; // (label)
_t2EdoUser:
                                                ; // doUser
  .pword 0xDA4000                   ; // breakPoint UNTESTED                          // !!! This code was not tested yet
!!!
    nop

    mov       w0, [w9++]                        ; // [PFA] PUSH PSP
    mov       w1, [w9++]

    inc2      w6, w6                            ; // PFA
    addc      w7, #0, w7
    mov       w7, TBLPAG

    tblrdl.w  [w6], w0                          ; // [PFA]
    tblrdh.w  [w6], w1

    mov       _VBank + 4, w2                ; // UP
    mov       _VBank + 6, w3

    add       w1, w3, w1                        ; // +
    add       w0, w2, w0
    addc      w1, #0, w1

    goto      _t2Enext                         ; // NEXT


; ==== 'forthInterpreter.Private.t.doDoes>' ===================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
;    A code fragment, not a forth primary
;      IP PUSH RSP
;      POP hardware call stack TO IP
;      [PFA] PUSH PSP
;      NEXT
;
; 0:t.doDoes>
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2EdoDoes3E, @function
  .endif
                                    ; // (label)
_t2EdoDoes3E:
                                                ; // doDoes>
  .pword 0xDA4000                   ; // breakPoint UNTESTED                          // !!! This code was not tested yet
!!!
    nop

    mov       w8, [--w10]                       ; // IP PUSH RSP
    mov       TBLPAG, w2
    mov       w2, [--w10]

    pop       TBLPAG                            ; // POP hardware call stack TO IP
    pop       w8

    mov       w0, [w9++]                        ; // [PFA] PUSH PSP
    mov       w1, [w9++]

    inc2      w6, w6                            ; // PFA
    addc      w7, #0, w7
    mov       w7, TBLPAG

    tblrdl.w  [w6], w0
    tblrdh.w  [w6], w1

    goto      _t2Enext                         ; // NEXT


; ==== 'forthInterpreter.Private.t.execute.h' =================================
;
;
; ( d -- ) simplified : ( d -- ) determined fixed 4 -> 0
;    CODE EXECUTE
;      POP PSP TO W
;      W JUMP
;
; 0:t.execute.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Eexecute2Eh, @function
  .endif
                                    ; // (label)
_t2Eexecute2Eh:

    .pword    _t2Ebye2Eh
    .pword    0xffff00
    .pascii   <7>,"execute"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Eexecute, @function
  .endif
```

```
_t2Eexecute:
                                                        ; // execute
    mov       w0, w6                                    ; // POP PSP TO W
    mov       w1, w7
    mov       [--w9], w1
    mov       [--w9], w0
    push      w6                                        ; // W JUMP
    push      w7
    return


; ==== 'forthInterpreter.Private.t.semi.h' ===================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;   [ d -- ] or EXIT
;    CODE SEMI
;      POP RSP TO IP
;      NEXT
;
; 0:t.semi.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Esemi2Eh, @function
  .endif
                                    ; // (label)
_t2Esemi2Eh:

    .pword    _t2Eexecute2Eh
    .pword    0xffff00
    .pascii   <4>,"semi"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Esemi, @function
  .endif

_t2Esemi:
                                                        ; // semi
    mov       [w10++], w2                               ; // POP RSP TO IP
    mov       w2, TBLPAG
    mov       [w10++], w8
    goto      _t2Enext                            ; // NEXT


; ==== 'forthInterpreter.Private.t.doLit.h' ==================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
;    CODE DOLIT
;      push inline value
;      NEXT
;
; 0:t.doLit.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2EdoLit2Eh, @function
  .endif
                                    ; // (label)
_t2EdoLit2Eh:

    .pword    _t2Esemi2Eh
    .pword    0xffff00
    .pascii   <5>,"doLit"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2EdoLit, @function
  .endif

_t2EdoLit:
                                                        ; // doLit
    mov       w0, [w9++]
    mov       w1, [w9++]
    tblrdl.w  [w8  ], w0
    tblrdh.w  [w8++], w1
    goto      _t2Enext                            ; // NEXT


; ==== 'forthInterpreter.Private.t.breakpoint.h' =============================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    CODE BREAKPOINT
;      stop execution here in the debugger
;      NEXT
;
; 0:t.breakpoint.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ebreakpoint2Eh, @function
  .endif
                                    ; // (label)
_t2Ebreakpoint2Eh:

    .pword    _t2EdoLit2Eh
    .pword    0xffff00
    .pascii   <10>,"breakpoint"
    .palign   2
```

```
  .ifdef    __C30ELF
  .type     _t2Ebreakpoint, @function
  .endif

_t2Ebreakpoint:
                                                ; // breakpoint
  .pword 0xDA4000                   ; // breakPoint t.breakpoint
    nop
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.cfa2pfa.h' ==================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    : cfa2pfa ( d -- d )
;      calculate PFA from CFA
;    ;
;
; 0:t.cfa2pfa.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecfa2pfa2Eh, @function
  .endif
                                    ; // (label)
_t2Ecfa2pfa2Eh:

    .pword    _t2Ebreakpoint2Eh
    .pword    0xffff00
    .pascii   <7>,"cfa2pfa"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecfa2pfa, @function
  .endif

_t2Ecfa2pfa:
                                                ; // cfa2pfa
    add       w0, #2, w0
    addc      w1, #0, w1
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.0=.h' ======================================
;
;
; ( d -- df ) simplified : ( d -- d ) determined fixed 4 -> 4
;    code 0= ( d -- df )
;      return $ffffffff when d = 0, and 0 otherwise
;
; 0:t.0=.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E03D2Eh, @function
  .endif
                                    ; // (label)
_t2E03D2Eh:

    .pword    _t2Ecfa2pfa2Eh
    .pword    0xffff00
    .pascii   <2>,"0="
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E03D, @function
  .endif

_t2E03D:
                                                ; // 0=
    ior       w0, w1, w0                        ; // Or low and high word
    sub       #1, w0                            ; // Subtract one, borrow set if 0= holds
    subb      w0, w0, w0                        ; // when borrow set $ffffffff, 0 otherwise
    mov       w0, w1
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.branch.h' ==================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    code branch ( -- )
;      branch to the following inline absolute address
;
; 0:t.branch.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ebranch2Eh, @function
  .endif
                                    ; // (label)
_t2Ebranch2Eh:

    .pword    _t2E03D2Eh
    .pword    0xffff00
    .pascii   <6>,"branch"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ebranch, @function
  .endif
```

```
_t2Ebranch:
                                                    ; // branch
    tblrdl.w  [w8], w2                              ; // [IP] -> IP
    tblrdh.w  [w8], w3
    mov       w2, w8
    mov       w3, TBLPAG
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.fbranch.h' ==================================
;
;
; ( df -- ) simplified : ( d -- ) determined fixed 4 -> 0
;    code fbranch
;      branch to the following inline absolute address
;      when TOS = 0, skip that address otherwise (and
;      do not branch then).
;
; 0:t.fbranch.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Efbranch2Eh, @function
  .endif
                                    ; // (label)
_t2Efbranch2Eh:

    .pword    _t2Ebranch2Eh
    .pword    0xffff00
    .pascii   <7>,"fbranch"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Efbranch, @function
  .endif

_t2Efbranch:
                                                    ; // fbranch
    ior       w0, w1, w0                            ; // Or low and high word, branch if zero
    bra       z, 1f                                 ; // B/ calculate new IP (branch)
    bra       2f                                    ; // B/ Skip over current IP
1:
    tblrdl.w  [w8], w2                              ; // [IP] -> IP // Branch
    tblrdh.w  [w8], w3
    mov       w2, w8
    mov       w3, TBLPAG
    bra       3f
2:
    mov       TBLPAG, w2                            ; // IP++       // No branch
    add       w8, #2, w8
    addc      w2, #0, w2
    mov       w2, TBLPAG
3:
    mov       [--w9], w1                            ; // Drop df
    mov       [--w9], w0
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.drop.h' =====================================
;
;
; ( d -- ) simplified : ( d -- ) determined fixed 4 -> 0
;    CODE DROP
;      drop TOS
;      NEXT
;
; 0:t.drop.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Edrop2Eh, @function
  .endif
                                    ; // (label)
_t2Edrop2Eh:

    .pword    _t2Efbranch2Eh
    .pword    0xffff00
    .pascii   <4>,"drop"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Edrop, @function
  .endif

_t2Edrop:
                                                    ; // drop
    mov       [--w9], w1
    mov       [--w9], w0
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.dup.h' ======================================
;
;
; ( d -- d d ) simplified : ( d -- d d ) determined fixed 4 -> 8
;    CODE DUP
;      dup TOS so NOS and TOS are equal
;      NEXT
;
; 0:t.dup.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
```

```
  .type     _t2Edup2Eh, @function
  .endif
                                    ; // (label)
_t2Edup2Eh:

    .pword    _t2Edrop2Eh
    .pword    0xffff00
    .pascii   <3>,"dup"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Edup, @function
  .endif

_t2Edup:
                                                    ; // dup
    mov       w0, [w9++]
    mov       w1, [w9++]
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.@.h' ========================================
;
;
; ( dAddress -- dData ) simplified : ( d -- d ) determined fixed 4 -> 4
;    CODE @ ( d -- d ) // fetch
;     get dData from dAddress
;     NEXT
;
; 0:t.@.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E402Eh, @function
  .endif
                                    ; // (label)
_t2E402Eh:

    .pword    _t2Edup2Eh
    .pword    0xffff00
    .pascii   <1>,"@"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E40, @function
  .endif

_t2E40:
                                                    ; // @
    mov       [w0++], w2                            ; // Hmm this supports only 64 k of RAM
    mov       [w0  ], w1                            ; // ... anyway ... move contents of dAddress to dData
    mov       w2, w0
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.!.h' ========================================
;
;
; ( dData dAddress -- ) simplified : ( d d -- ) determined fixed 8 -> 0
;    CODE ! ( d d -- ) // store
;      store dData at dAddress
;      NEXT
;
; 0:t.!.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E212Eh, @function
  .endif
                                    ; // (label)
_t2E212Eh:

    .pword    _t2E402Eh
    .pword    0xffff00
    .pascii   <1>,"!"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E21, @function
  .endif

_t2E21:
                                                    ; // !
    mov       [--w9], w3                            ; // dData -> w3,w2
    mov       [--w9], w2
    mov       w2, [w0++]                            ; // Hmm this supports only 64 k of RAM
    mov       w3, [w0  ]                            ; // ... anyway ... move dData to dAddress
    mov       [--w9], w1                            ; // And drop an item
    mov       [--w9], w0
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.f@.h' =======================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    code f@ ( d -- d )
;      fetch a value from code space (flash)
;
; 0:t.f@.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
```

```
  .type     _t2Ef402Eh, @function
  .endif
                                        ; // (label)
_t2Ef402Eh:

    .pword    _t2E212Eh
    .pword    0xffff00
    .pascii   <2>,"f@"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ef40, @function
  .endif

_t2Ef40:
                                                    ; // f@
    mov       TBLPAG, w3                            ; // Save TBLPAG
    mov       w1, TBLPAG                            ; // Set up flash address into TBLPAG, w2
    mov       w0, w2
    tblrdl.w  [w2], w0                              ; // Read two words from flash
    tblrdh.w  [w2], w1
    mov       w3, TBLPAG                            ; // Restore TBLPAG
    goto      _t2Enext                            ; // NEXT


; ==== 'forthInterpreter.Private.t.+.h' ======================================
;
;
; ( d1 d2 -- d ) simplified : ( d d -- d ) determined fixed 8 -> 4
;    CODE +
;      add d1 and d2
;      NEXT
;
; 0:t.+.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E2B2Eh, @function
  .endif
                                        ; // (label)
_t2E2B2Eh:

    .pword    _t2Ef402Eh
    .pword    0xffff00
    .pascii   <1>,"+"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E2B, @function
  .endif

_t2E2B:
                                                    ; // +
    add       w1, [--w9], w1
    add       w0, [--w9], w0
    addc      w1, #0, w1
    goto      _t2Enext                            ; // NEXT


; ==== 'forthInterpreter.Private.t.negate.h' =================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    CODE negate
;      negate TOS
;      NEXT
;
; 0:t.negate.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Enegate2Eh, @function
  .endif
                                        ; // (label)
_t2Enegate2Eh:

    .pword    _t2E2B2Eh
    .pword    0xffff00
    .pascii   <6>,"negate"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Enegate, @function
  .endif

_t2Enegate:
                                                    ; // negate
    neg     w1, w1
    neg     w0, w0
    subb    #0, w1
    goto      _t2Enext                            ; // NEXT


; ==== 'forthInterpreter.Private.t.-.h' ======================================
;
;
; ( d1 d2 -- d ) simplified : ( d d -- d ) determined fixed 8 -> 4
;    : - ( d1 d2 -- d ) negate + ;
;      subtract d2 from d1
;
; 0:t.-.h
;
```

```
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E2D2Eh, @function
  .endif
                                    ; // (label)
_t2E2D2Eh:

    .pword    _t2Enegate2Eh
    .pword    0xffff00
    .pascii   <1>,"-"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E2D, @function
  .endif

_t2E2D:
    bra       _t2EdoCol                         ; // DOCOL :: Execute following pointer list
                                                    ; // -
    .pword    _t2Enegate                        ; // negate
    .pword    _t2E2B                             ; // +

    .pword    _t2Esemi                          ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.and.h' ====================================
;
;
; ( d1 d2 -- d ) simplified : ( d d -- d ) determined fixed 8 -> 4
;    CODE and
;      and d1 and d2
;      NEXT
;
; 0:t.and.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Eand2Eh, @function
  .endif
                                    ; // (label)
_t2Eand2Eh:

    .pword    _t2E2D2Eh
    .pword    0xffff00
    .pascii   <3>,"and"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Eand, @function
  .endif

_t2Eand:
                                                  ; // and
    and       w1, [--w9], w1
    and       w0, [--w9], w0
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.or.h' =====================================
;
;
; ( d1 d2 -- d ) simplified : ( d d -- d ) determined fixed 8 -> 4
;    CODE or
;      or d1 and d2
;      NEXT
;
; 0:t.or.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Eor2Eh, @function
  .endif
                                    ; // (label)
_t2Eor2Eh:

    .pword    _t2Eand2Eh
    .pword    0xffff00
    .pascii   <2>,"or"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Eor, @function
  .endif

_t2Eor:
                                                  ; // or
    ior       w1, [--w9], w1
    ior       w0, [--w9], w0
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.xor.h' ====================================
;
;
; ( d1 d2 -- d ) simplified : ( d d -- d ) determined fixed 8 -> 4
;    CODE xor
;      or d1 and d2
;      NEXT
;    CODE xor
;      or d1 and d2
;      NEXT
;
; 0:t.xor.h
```

```
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Exor2Eh, @function
  .endif
                                    ; // (label)
_t2Exor2Eh:

    .pword    _t2Eor2Eh
    .pword    0xffff00
    .pascii   <3>,"xor"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Exor, @function
  .endif

_t2Exor:
                                                ; // xor
    xor       w1, [--w9], w1
    xor       w0, [--w9], w0
    goto      _t2Enext                      ; // NEXT


; ==== 'forthInterpreter.Private.t.not.h' ====================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
; 0:t.not.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Enot2Eh, @function
  .endif
                                    ; // (label)
_t2Enot2Eh:

    .pword    _t2Exor2Eh
    .pword    0xffff00
    .pascii   <3>,"not"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Enot, @function
  .endif

_t2Enot:
                                                ; // not
    com       w0, w0
    com       w1, w1
    goto      _t2Enext                      ; // NEXT


; ==== 'forthInterpreter.Private.t.cells.h' ==================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    : CELLS ( d -- d ) 2* 2* ;
;      multiply d by CELL
;
; 0:t.cells.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecells2Eh, @function
  .endif
                                    ; // (label)
_t2Ecells2Eh:

    .pword    _t2Enot2Eh
    .pword    0xffff00
    .pascii   <5>,"cells"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecells, @function
  .endif

_t2Ecells:
                                                ; // cells
    sl      w0, w0                              ; // Shift left low word,
                                                ; // shift a zero to bit 0
                                                ; // shift bit 15 into carry
    rlc     w1, w1                              ; // rotate left high word
                                                ; // carry into bit 0, bit 15 into carry
    sl      w0, w0                              ; // And again
    rlc     w1, w1
    goto      _t2Enext                      ; // NEXT


; ==== 'forthInterpreter.Private.t.=.h' ======================================
;
;
; ( d1 d2 -- df ) simplified : ( d d -- d ) determined fixed 8 -> 4
;    : = - 0= ; // return true when d1 = d2, false otherwise
;
; 0:t.=.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E3D2Eh, @function
  .endif
                                    ; // (label)
_t2E3D2Eh:
```

```
    .pword    _t2Ecells2Eh
    .pword    0xffff00
    .pascii   <1>,"="
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E3D, @function
  .endif

_t2E3D:
    bra       _t2EdoCol                       ; // DOCOL :: Execute following pointer list
                                                  ; // =
    .pword    _t2E2D                           ; // - 0=
    .pword    _t2E03D

    .pword    _t2Esemi                        ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.=marker.h' ==================================
;
;
; ( d -- df ) simplified : ( d -- d ) determined fixed 4 -> 4
;    : =marker marker.mask and marker.mask = ; // return true when d is a header marker, false otherwise
;
; 0:t.=marker.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E3Dmarker2Eh, @function
  .endif
                                    ; // (label)
_t2E3Dmarker2Eh:

    .pword    _t2E3D2Eh
    .pword    0xffff00
    .pascii   <7>,"=marker"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E3Dmarker, @function
  .endif

_t2E3Dmarker:
    bra       _t2EdoCol                       ; // DOCOL :: Execute following pointer list
                                                  ; // =marker
    .pword    _t2Emarker2Emask                 ; // marker.mask and marker.mask =
    .pword    _t2Eand
    .pword    _t2Emarker2Emask
    .pword    _t2E3D

    .pword    _t2Esemi                        ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.2+.h' ======================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    code 2_ ( d - d )
;      add 2 to TOS
;
; 0:t.2+.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E22B2Eh, @function
  .endif
                                    ; // (label)
_t2E22B2Eh:

    .pword    _t2E3Dmarker2Eh
    .pword    0xffff00
    .pascii   <2>,"2+"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E22B, @function
  .endif

_t2E22B:
                                                  ; // 2+
    add       w0, #2, w0
    addc      w1, #0, w1
    goto      _t2Enext                        ; // NEXT


; ==== 'forthInterpreter.Private.t.2-.h' ======================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    code 2_ ( d - d )
;      subtract 2 from TOS
;
; 0:t.2-.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E22D2Eh, @function
  .endif
                                    ; // (label)
_t2E22D2Eh:

    .pword    _t2E22B2Eh
```

```
    .pword    0xffff00
    .pascii   <2>,"2-"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E22D, @function
  .endif

_t2E22D:
                                                ; // 2-
    sub       w0, #2, w0
    subb      w1, #0, w1
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.cfa2ffa.h' ==================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;   : cfa2ffa ( d -- d )                        // Determine Flag Field Address (FFA) from CFA
;   begin
;   2- dup f@ =marker                          // Serarch backwards from CFA till marker found
;   until
;   ;
;
; 0:t.cfa2ffa.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecfa2ffa2Eh, @function
  .endif
                                    ; // (label)
_t2Ecfa2ffa2Eh:

    .pword    _t2E22D2Eh
    .pword    0xffff00
    .pascii   <7>,"cfa2ffa"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecfa2ffa, @function
  .endif

_t2Ecfa2ffa:
    bra       _t2EdoCol                         ; // DOCOL :: Execute following pointer list
                                                    ; // cfa2ffa
_cfa2ffa_loop:                                  ; // begin
    .pword    _t2E22D                            ; //   2-
    .pword    _t2Edup                            ; //   dup
    .pword    _t2Ef40                            ; //   f@
    .pword    _t2E3Dmarker                       ; //   =marker
    .pword    _t2Efbranch                        ; // until
    .pword    _cfa2ffa_loop                     ; //

    .pword    _t2Esemi                          ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.cfa2lfa.h' ==================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    : cfa2lfa ( d -- d ) cfa2ffa 2- ; // Get Link Field Address from CFA
;
; 0:t.cfa2lfa.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecfa2lfa2Eh, @function
  .endif
                                    ; // (label)
_t2Ecfa2lfa2Eh:

    .pword    _t2Ecfa2ffa2Eh
    .pword    0xffff00
    .pascii   <7>,"cfa2lfa"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecfa2lfa, @function
  .endif

_t2Ecfa2lfa:
    bra       _t2EdoCol                         ; // DOCOL :: Execute following pointer list
                                                    ; // cfa2lfa
    .pword    _t2Ecfa2ffa                       ; // cfa2ffa 2-
    .pword    _t2E22D

    .pword    _t2Esemi                          ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.cfa2nfa.h' ==================================
;
;
; ( d -- d ) simplified : ( d -- d ) determined fixed 4 -> 4
;    : cfa2nfa ( d -- d ) cfa2ffa 2+ ; // Get Name Field Address from CFA
;
; 0:t.cfa2nfa.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecfa2nfa2Eh, @function
  .endif
                                    ; // (label)
```

```
_t2Ecfa2nfa2Eh:

    .pword    _t2Ecfa2lfa2Eh
    .pword    0xffff00
    .pascii   <7>,"cfa2nfa"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecfa2nfa, @function
  .endif

_t2Ecfa2nfa:
    bra       _t2EdoCol                       ; // DOCOL :: Execute following pointer list
                                                ;    // cfa2nfa
    .pword    _t2Ecfa2ffa                     ; // cfa2ffa 2+
    .pword    _t2E22B

    .pword    _t2Esemi                        ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.vallot.h' ==================================
;
;
; ( d -- ) simplified : ( d -- ) determined fixed 4 -> 0
;    : VALLOT ( d -- ) CELLS VHERE @ + VHERE ! ; // allot d cells to variable space
;
; 0:t.vallot.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Evallot2Eh, @function
  .endif
                                    ; // (label)
_t2Evallot2Eh:

    .pword    _t2Ecfa2nfa2Eh
    .pword    0xffff00
    .pascii   <6>,"vallot"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Evallot, @function
  .endif

_t2Evallot:
    bra       _t2EdoCol                       ; // DOCOL :: Execute following pointer list
                                                ;    // vallot
    .pword    _t2Ecells                       ; // CELLS -> bytes
    .pword    _t2Evhere                       ; // vhere
    .pword    _t2E40                           ; // @
    .pword    _t2E2B                           ; // +
    .pword    _t2Evhere                       ; // vhere
    .pword    _t2E21                           ; // !

    .pword    _t2Esemi                        ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.(is).h' ====================================
;
;
; ( dtoken ddefer -- ) simplified : ( d d -- ) determined fixed 8 -> 0
;    : (is) ( dtoken ddefer -- ) cfa2pfa f@ ! ;
;    Resolve a deferred word with execution token ddefer to dtokenq
;    i.e. set the deferred word to execute dtoken
;
; 0:t.(is).h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E28is292Eh, @function
  .endif
                                    ; // (label)
_t2E28is292Eh:

    .pword    _t2Evallot2Eh
    .pword    0xffff00
    .pascii   <4>,"(is)"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E28is29, @function
  .endif

_t2E28is29:
    bra       _t2EdoCol                       ; // DOCOL :: Execute following pointer list
                                                ;    // (is)
    .pword    _t2Ecfa2pfa                     ; // cfa2pfa f@ !
    .pword    _t2Ef40
    .pword    _t2E21

    .pword    _t2Esemi                        ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.noop.h' ====================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    code noop ( -- )
;      wastes a few cycles
;
; 0:t.noop.h
;
```

```
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Enoop2Eh, @function
  .endif
                                    ; // (label)
_t2Enoop2Eh:

    .pword    _t2E28is292Eh
    .pword    0xffff00
    .pascii   <4>,"noop"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Enoop, @function
  .endif

_t2Enoop:
                                                ; // noop
    goto      _t2Enext                          ; // NEXT


; ==== 'forthInterpreter.Private.t.definitions.h' ==============================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    : definitions ( -- ) context @ current ! ;
;    make compiler vocabulary (current) to the currently topmost
;    search vocabulary (context)
;
; 0:t.definitions.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Edefinitions2Eh, @function
  .endif
                                    ; // (label)
_t2Edefinitions2Eh:

    .pword    _t2Enoop2Eh
    .pword    0xffff00
    .pascii   <11>,"definitions"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Edefinitions, @function
  .endif

_t2Edefinitions:
    bra       _t2EdoCol                         ; // DOCOL :: Execute following pointer list
                                                    ; // definitions
    .pword    _t2Econtext                       ; // context @ current!
    .pword    _t2E40
    .pword    _t2Ecurrent
    .pword    _t2E21

    .pword    _t2Esemi                          ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.vhere.h' ====================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.vhere.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Evhere2Eh, @function
  .endif
                                    ; // (label)
_t2Evhere2Eh:

    .pword    _t2Edefinitions2Eh
    .pword    0xffff00
    .pascii   <5>,"vhere"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Evhere, @function
  .endif

_t2Evhere:
    bra       _t2EdoVar                         ; // DOVAR :: push following pword
    .pword    _VBank + 0


; ==== 'forthInterpreter.Private.t.chere.h' ====================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.chere.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Echere2Eh, @function
  .endif
                                    ; // (label)
_t2Echere2Eh:

    .pword    _t2Evhere2Eh
    .pword    0xffff00
    .pascii   <5>,"chere"
    .palign   2

  .ifdef    __C30ELF
```

```
  .type     _t2Echere, @function
  .endif

_t2Echere:
    bra       _t2EdoVar                         ; // DOVAR :: push following pword
    .pword    _VBank + 4


; ==== 'forthInterpreter.Private.t.last.h' ==================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.last.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Elast2Eh, @function
  .endif
                                    ; // (label)
_t2Elast2Eh:

    .pword    _t2Echere2Eh
    .pword    0xffff00
    .pascii   <4>,"last"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Elast, @function
  .endif

_t2Elast:
    bra       _t2EdoVar                         ; // DOVAR :: push following pword
    .pword    _VBank + 8


; ==== 'forthInterpreter.Private.t.current.h' ===============================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.current.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecurrent2Eh, @function
  .endif
                                    ; // (label)
_t2Ecurrent2Eh:

    .pword    _t2Elast2Eh
    .pword    0xffff00
    .pascii   <7>,"current"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecurrent, @function
  .endif

_t2Ecurrent:
    bra       _t2EdoVar                         ; // DOVAR :: push following pword
    .pword    _VBank + 12


; ==== 'forthInterpreter.Private.t.context.h' ===============================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.context.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Econtext2Eh, @function
  .endif
                                    ; // (label)
_t2Econtext2Eh:

    .pword    _t2Ecurrent2Eh
    .pword    0xffff00
    .pascii   <7>,"context"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Econtext, @function
  .endif

_t2Econtext:
    bra       _t2EdoVar                         ; // DOVAR :: push following pword
    .pword    _VBank + 16


; ==== 'forthInterpreter.Private.t.up.h' ====================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.up.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Eup2Eh, @function
  .endif
                                    ; // (label)
_t2Eup2Eh:

    .pword    _t2Econtext2Eh
    .pword    0xffff00
```

```
    .pascii   <2>,"up"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Eup, @function
  .endif

_t2Eup:
    bra       _t2EdoVar                       ; // DOVAR :: push following pword
    .pword    _VBank + 20


; ==== 'forthInterpreter.Private.t.(main).h' ==================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.(main).h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2E28main292Eh, @function
  .endif
                                    ; // (label)
_t2E28main292Eh:

    .pword    _t2Eup2Eh
    .pword    0xffff00
    .pascii   <6>,"(main)"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2E28main29, @function
  .endif

_t2E28main29:
    bra       _t2EdoDefer                     ; // DODEFER :: execute following pword
    .pword    _VBank + 24


; ==== 'forthInterpreter.Private.t.cell.h' ====================================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.cell.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecell2Eh, @function
  .endif
                                    ; // (label)
_t2Ecell2Eh:

    .pword    _t2E28main292Eh
    .pword    0xffff00
    .pascii   <4>,"cell"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecell, @function
  .endif

_t2Ecell:
    bra       _t2EdoCon                       ; // DOCON :: push following pword
    .pword    4


; ==== 'forthInterpreter.Private.t.initial.vallot.h' ==========================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.initial.vallot.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Einitial2Evallot2Eh, @function
  .endif
                                    ; // (label)
_t2Einitial2Evallot2Eh:

    .pword    _t2Ecell2Eh
    .pword    0xffff00
    .pascii   <14>,"initial.vallot"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Einitial2Evallot, @function
  .endif

_t2Einitial2Evallot:
    bra       _t2EdoCon                       ; // DOCON :: push following pword
    .pword    7


; ==== 'forthInterpreter.Private.t.marker.mask.h' =============================
;
;
; ( -- d ) simplified : ( -- d ) determined fixed 0 -> 4
; 0:t.marker.mask.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Emarker2Emask2Eh, @function
  .endif
                                    ; // (label)
```

```
_t2Emarker2Emask2Eh:

    .pword    _t2Einitial2Evallot2Eh
    .pword    0xffff00
    .pascii   <11>,"marker.mask"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Emarker2Emask, @function
  .endif

_t2Emarker2Emask:
    bra       _t2EdoCon                       ; // DOCON :: push following pword
    .pword    16776960


; ==== 'forthInterpreter.Private.t.cold.h' ====================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    : COLD ( -- )
;
;      VBANK VHERE !
;      0     UP    !
;      BYE
;    ;
;
; 0:t.cold.h
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Ecold2Eh, @function
  .endif
                                    ; // (label)
_t2Ecold2Eh:

    .pword    _t2Emarker2Emask2Eh
    .pword    0xffff00
    .pascii   <4>,"cold"
    .palign   2

  .ifdef    __C30ELF
  .type     _t2Ecold, @function
  .endif

_t2Ecold:
    bra       _t2EdoCol                       ; // DOCOL :: Execute following pointer list
                                                ;    ; // cold
    .pword    _t2EdoLit                       ; // VBANK VHERE !
    .pword    _VBank
    .pword    _t2Evhere
    .pword    _t2E21
    ; // VALLOT all defined target variables
    ; // NOTE :: use this after VHERE was setup
    .pword    _t2Einitial2Evallot              ; // INITIAL.VALLOT VALLOT
    .pword    _t2Evallot
    ; // Set CHERE to point to first free FLASH location
    .pword    _t2EdoLit                       ; // LAST_FLASH CHERE !
    .pword    _LAST_FLASH
    .pword    _t2Echere
    .pword    _t2E21
    ; // Set LAST to point to last header definition (this one)
    .pword    _t2EdoLit                       ; // ' COLD LAST !
    .pword    _t2Ecold2Eh
    .pword    _t2Elast
    .pword    _t2E21
    .pword    _t2Elast                        ; // LAST @ CONTEXT !
    .pword    _t2E40
    .pword    _t2Econtext
    .pword    _t2E21
    .pword    _t2Edefinitions                 ; // DEFINITIONS ( context @ current ! )
    .pword    _t2EdoLit                       ; // 0 UP !
    .pword    0
    .pword    _t2Eup
    .pword    _t2E21
    .pword    _t2EdoLit                       ; // ' (MAIN) IS NOOP
    .pword    _t2Enoop
    .pword    _t2EdoLit
    .pword    _t2E28main29
    .pword    _t2E28is29
    .pword    _t2EdoLit                       ; // ' noop cfa2ffa drop
    .pword    _t2Enoop
    .pword    _t2Ecfa2ffa
    .pword    _t2Edrop
    .pword    _t2E28main29                      ; // (MAIN)
    .pword    _t2Ebye                         ; // BYE :: Return to caller of forth.start

    .pword    _t2Esemi                        ; // SEMI :: return to calling pointer list


; ==== 'forthInterpreter.Private.t.lastdef' ===================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
; 0:t.lastdef
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _t2Elastdef, @function
  .endif
                                    ; // (label)
_t2Elastdef:
```

```
                                                ; // lastdef
    .pascii       "This is the end."

_LAST_FLASH:



; Vocabulary end.
; Vocabulary 'forthInterpreter.Public'

; ==== 'forthInterpreter.Public.forth.start' ==================================
;
;
; ( -- ) simplified : ( -- ) determined fixed 0 -> 0
;    Start the forth engine by jumping into COLD.
;    COLD will run some code and when finished it
;    will return control to the caller of START
;    trhough the word BYE.
;
; 0:forth.start
;
  .ifdef    __C30ELF                ; // (debug-info-func)
  .type     _forth2Estart, @function
  .endif
                                    ; // (label)
_forth2Estart:

    ; // Start the forth engine by jumping into COLD.
    ; //
    ; // Save registers

  .pword 0xDA4000                   ; // breakPoint forth.start
    nop
                                          ; // forth.start
    mov       w0, _RBank                ; //  0 ::  0 : old w0
    mov       #_RBank + 2, w0
    mov       w1 , [w0++]                 ; //  2 ::  1
    mov       w2 , [w0++]                 ; //  4 ::  2
    mov       w3 , [w0++]                 ; //  6 ::  3
    mov       w4 , [w0++]                 ; //  8 ::  4
    mov       w5 , [w0++]                 ; // 10 ::  5
    mov       w6 , [w0++]                 ; // 12 ::  6
    mov       w7 , [w0++]                 ; // 14 ::  7
    mov       w8 , [w0++]                 ; // 16 ::  8
    mov       w9 , [w0++]                 ; // 18 ::  9
    mov       w10, [w0++]                 ; // 20 :: 10
    mov       w11, [w0++]                 ; // 22 :: 11
    mov       w12, [w0++]                 ; // 24 :: 12
    mov       w13, [w0++]                 ; // 26 :: 13
    mov       w14, [w0++]                 ; // 28 :: 14
    mov       w15, [w0++]                 ; // 30 :: 15
    mov       TBLPAG, w2
    mov       w2 , [w0++]                 ; // 32 :: 16
    mov       RCOUNT, w2
    mov       w2 , [w0++]                 ; // 34 :: 17
    mov       SR, w2
    mov       w2 , [w0]                   ; // 36 :: 18

    ; // Initialize forth interpreter

    mov       #_PS, w9                  ; // Set PSP to lowest  address in PS

    mov       #_RS, w3                  ; // Set RSP one beyond highest address in RS
    mov       #_RSSize, w10
    add       w3, w10, w10

    ; // Jump into COLD, setting up initial W contents on the fly

    mov       #tblpage  ( _t2Ecold), w2         ; // Set up t.cold as the entry point
    mov       w2, TBLPAG                      ; // Which needs be a forth secondary word
    mov       #tbloffset( _t2Ecold), w8

    mov       w8    , w6                      ; // Set up initial W to point to t.cold
    mov       TBLPAG, w7

    push      w6                              ; // JUMP IP (into t.cold)
    push      w7

    return



; Vocabulary end.
```