

6502 Assembler for VolksForth#

See [6502 Assembler in Forth](#) for the original FIG-Forth version of this assembler

```
\ 6502 Assembler clv10oct87
\ Basis: Forth Dimensions VOL III No. 5)
\ internal loading 04may85BP/re)
\ Forth-6502 Assembler clv10oct87
\ Basis: Forth Dimensions VOL III No. 5)
```

```
CR .( Loading 6502 Assembler...) CR
```

```
Onlyforth Assembler also definitions
```

```
\ Forth-83 6502-Assembler 20oct87re
```

```
: end-code context 2- @ context ! ;
```

```
Create index
```

```
$0909 , $1505 , $0115 , $8011 ,
$8009 , $1D0D , $8019 , $8080 ,
$0080 , $1404 , $8014 , $8080 ,
$8080 , $1C0C , $801C , $2C80 ,
```

```
| Variable mode
```

```
: Mode: ( n -) Create c,
Does> ( -) c@ mode ! ;
```

```
0 Mode: .A 1 Mode: #
2 | Mode: mem 3 Mode: ,X
4 Mode: ,Y 5 Mode: X)
6 Mode: )Y $F Mode: )
```

```
\ upmode cpu 20oct87re
```

```
| : upmode ( addr0 f0 - addr1 f1)
IF mode @ 8 or mode ! THEN
1 mode @ $F and ?dup IF
0 DO dup + LOOP THEN
over 1+ @ and 0= ;
```

```
: cpu ( 8b -) Create c,
Does> ( -) c@ c, mem ;
```

```
00 cpu brk $18 cpu clc $D8 cpu cld
$58 cpu cli $B8 cpu clv $CA cpu dex
$88 cpu dey $E8 cpu inx $C8 cpu iny
$EA cpu nop $48 cpu pha $08 cpu php
$68 cpu pla $28 cpu plp $40 cpu rti
$60 cpu rts $38 cpu sec $F8 cpu sed
$78 cpu sei $AA cpu tax $A8 cpu tay
$BA cpu tsx $8A cpu txa $9A cpu txs
$98 cpu tya
```

```
\ m/cpu 20oct87re
```

```
: m/cpu ( mode opcode -) Create c, ,
Does>
```

```

dup 1+ @ $80 and IF $10 mode +! THEN
over $FF00 and upmode upmode
IF mem true Abort" invalid" THEN
c@ mode @ index + c@ + c, mode @ 7 and
IF mode @ $F and 7 <
  IF c, ELSE , THEN THEN mem ;

```

```

$1C6E $60 m/cpu adc $1C6E $20 m/cpu and
$1C6E $C0 m/cpu cmp $1C6E $40 m/cpu eor
$1C6E $A0 m/cpu lda $1C6E $00 m/cpu ora
$1C6E $E0 m/cpu sbc $1C6C $80 m/cpu sta
$0D0D $01 m/cpu asl $0C0C $C1 m/cpu dec
$0C0C $E1 m/cpu inc $0D0D $41 m/cpu lsr
$0D0D $21 m/cpu rol $0D0D $61 m/cpu ror
$0414 $81 m/cpu stx $0486 $E0 m/cpu cpx
$0486 $C0 m/cpu cpy $1496 $A2 m/cpu ldx
$0C8E $A0 m/cpu ldy $048C $80 m/cpu sty
$0480 $14 m/cpu jsr $8480 $40 m/cpu jmp
$0484 $20 m/cpu bit

```

\ Assembler conditionals 20oct87re

```

| : range? ( branch -- branch )
dup abs $7F u> Abort" out of range " ;

```

```

: [[ ( BEGIN) here ;
: ?] ( UNTIL) c, here 1+ - range? c, ;
: ?[ ( IF) c, here 0 c, ;
: ?[[ ( WHILE) ?[ swap ;
: ]? ( THEN) here over c@ IF swap !
ELSE over 1+ - range? swap c! THEN ;
: ][ ( ELSE) here 1+ 1 jmp
swap here over 1+ - range? swap c! ;
: ]] ( AGAIN) jmp ;
: ]]? ( REPEAT) jmp ]? ;

```

\ Assembler conditionals 20oct87re

```

$90 Constant CS $B0 Constant CC
$D0 Constant 0= $F0 Constant 0<>
$10 Constant 0< $30 Constant 0>=
$50 Constant VS $70 Constant VC

```

```

: not $20 [ Forth ] xor ;

: beq 0<> ?] ; : bmi 0>= ?] ;
: bne 0= ?] ; : bpl 0< ?] ;
: bcc CS ?] ; : bvc VS ?] ;
: bcs CC ?] ; : bvs VC ?] ;

```

\ 2inc/2dec winc/wdec 20oct87re

```

: 2inc ( adr -- )
dup lda clc 2 # adc
dup sta CS ?[ swap 1+ inc ]? ;

: 2dec ( adr -- )
dup lda sec 2 # sbc
dup sta CC ?[ swap 1+ dec ]? ;

```

```

: winc ( adr -- )
  dup inc 0= ?[ swap 1+ inc ]? ;

: wdec ( adr -- )
  dup lda 0= ?[ over 1+ dec ]? dec ;

: ;c:
  recover jsr end-code ] 0 last ! 0 ;

\ ;code Code code>          bp/re03feb85

```

Onlyforth

```

: Assembler
  Assembler [ Assembler ] mem ;

: ;Code
  [compile] Does> -3 allot
  [compile] ; -2 allot Assembler ;
immediate

: Code Create here dup 2- ! Assembler ;

: >label ( adr -)
  here | Create immediate swap ,
  4 hallot heap 1 and hallot ( 6502-align)
  here 4 - heap 4 cmove
  heap last @ count $1F and + ! dp !
  Does> ( - adr) @
  state @ IF [compile] Literal THEN ;

: Label
  [ Assembler ] here >label Assembler ;

```

Onlyforth