```
\
\ 65816 Target Assembler Release 1p1
\ Copyright (c) 2006 Samuel A. Falvo II
\
\ This software is placed under LGPL v2 license.
\
\ THIS ASSEMBLER WILL NOT DETECT INCORRECT COMBINATIONS
\ OF INSTRUCTIONS AND ADDRESSING MODES.  Therefore, it
\ is assumed you know what you are doing when using
\ this assembler!!
\
\ It's written for ANSI Forth (GForth in particular),
\ but ought to be trivially ported to most any Forth
\ environment I can personally think of.
\
\ Converting this to a host assembler (vs. a target
\ assembler) equally should be pretty easy.  All code-
\ generation has been factored out into the *,-series of
\ words.  Address fix-up code, e.g., in the program flow
\ convenience words, may need to be addressed as well.
\

: binary    2 base ! ;
: 2+        2 + ;

base @

\ ============================================================================
\ assembler cursor ("*" in typical assemblers)
\ ============================================================================

variable    hbase   ( base address of image in host address space )
variable    tbase   ( base address of image in target address space )
variable    &*      ( 0 <= tbase <= * <= $FFFF )
: *!        &* ! ;  ( *=$xxxx becomes $xxxx *! )
: *@        &* @ ;  ( BEQ *+4 becomes *@ 4 + BEQ, )
: .scr      .s cr ;
: t>h       tbase @ - hbase @ + ;
: h>t       hbase @ - tbase @ + ;
: poke      t>h c! ;
: peek      t>h c@ ;

: 256/      8 rshift ;
: *,        dup 255 and *@ dup 1+ *! poke 256/ ;
: *,b       *, drop ;
: *,w       *, *, drop ;
: *,l       *, *, *, drop ;
: *,d       *, *, *, *, drop ;

\ ============================================================================
\ Addressing Modes
\ ============================================================================

variable 16bit ( immediates )
variable modemask

: amode     create c, does> c@ modemask c! ;

binary
\ ---------------------------------------------------------------
\ 6502 Original         New as of               New as of
\ Addressing Mode       65C02                   65802/65816
\ ---------------------------------------------------------------

01001 amode #,          10010 amode (dp),       10111 amode [],y,
00101 amode dp,                                 00111 amode [dp],
01101 amode $,                                  01111 amode $l,
10101 amode dp,x,                               11111 amode $l,x,
11101 amode $,x,                                00011 amode (,s),
11001 amode $,y,                                10011 amode (,s),y,
00001 amode (,x),
10001 amode (),y,
```

```
: #b,       #, 16bit off ;  ( forcing 8-bit immediate values )
: #w,       #, 16bit on ;   ( forcing 16-bit immediate values )
: dp,y,     dp,x, ;         ( for the benefit of the stx dp,y instruction )

: imm?      modemask c@ 01001 = ;
: abs?      modemask c@ 01001 and 01001 = ;
: 24bit?    modemask c@ 01111 and 01111 = ;

: operands,
    imm? IF 16bit @ IF *,w ELSE *,b THEN EXIT THEN  ( MUST come first )
    abs? IF 24bit? IF *,l ELSE *,w THEN EXIT THEN
    *,b ;

\ ======================================================================
\ Group-0 instructions: instructions with no operands
\ ======================================================================

hex

: grp0:    create c, does> c@ *,b ;

00 grp0: brk, ( historical interest only; use SYS instead )
40 grp0: rti,
60 grp0: rts,

08 grp0: php,      0A grp0: sla,      0B grp0: phd,
18 grp0: clc,      1A grp0: ina,      1B grp0: tas,
28 grp0: plp,      2A grp0: rla,      2B grp0: pld,
38 grp0: sec,      3A grp0: dea,      3B grp0: tsa,
48 grp0: pha,      4A grp0: sra,      4B grp0: phk,
58 grp0: cli,      5A grp0: phy,      5B grp0: tad,
68 grp0: pla,      6A grp0: rra,      6B grp0: rtl,
78 grp0: sei,      7A grp0: ply,      7B grp0: tda,
88 grp0: dey,      8A grp0: txa,      8B grp0: phb,
98 grp0: tya,      9A grp0: txs,      9B grp0: txy,
A8 grp0: tay,      AA grp0: tax,      AB grp0: plb,
B8 grp0: clv,      BA grp0: tsx,      BB grp0: tyx,
C8 grp0: iny,      CA grp0: dex,      CB grp0: wai,
D8 grp0: cld,      DA grp0: phx,      DB grp0: stp,
E8 grp0: inx,      EA grp0: nop,      EB grp0: xba,
F8 grp0: sed,      FA grp0: plx,      FB grp0: xce,

: tcs, tas, ;   ( synonyms )
: tsc, tsa, ;
: tcd, tad, ;
: tdc, tda, ;

: sys, 00 *,b *,b ;

\ ======================================================================
\ Group-1 opcodes
\ ======================================================================

: grp1:     create c, does> c@ modemask c@ or *,b operands, ;

binary

00000000 grp1: ora,     00100000 grp1: and,
01000000 grp1: eor,     01100000 grp1: adc,
10000000 grp1: sta,     10100000 grp1: lda,
11000000 grp1: cmp,     11100000 grp1: sbc,

\ ======================================================================
\ Group-2 opcodes
\ ======================================================================

binary

: wonkymode modemask c@ 11100 and 01000 over = 01000 and xor ;
: grp2a:    create c, does> c@ modemask c@ 11100 and or *,b operands, ;
: grp2b:    create c, does> c@ modemask c@ 11000 and or *,b operands, ;
```

```
: grp2c:    create c, does> c@ wonkymode or *,b operands, ;

00000010 grp2a: asl,    10000110 grp2b: stx,
00100010 grp2a: rol,    10000100 grp2b: sty,
01000010 grp2a: lsr,    11000110 grp2b: dec,
01100010 grp2a: ror,    11100110 grp2b: inc,

10100010 grp2c: ldx,    10100000 grp2c: ldy,

\ ======================================================================
\ Group-3 opcodes
\ ======================================================================

binary

: grp3a:    create c, does> c@ wonkymode 01100 and or *,b operands, ;
: grp3b:    create c, does> c@ wonkymode 01000 and or *,b operands, ;

11100000 grp3a: cpx,    11000000 grp3a: cpy,
00010100 grp3b: trb,    00000100 grp3b: tsb,

\ ======================================================================
\ Group-4 opcodes -- whatever's left over
\ ======================================================================

decimal

: range
    dup abs 128 U> IF ." Branch target possibly out of range" cr THEN ;

: offset8,  *@ 1+ - range *,b ;
: offset16, *@ 1+ - *,w ;
: grp4a:    create c, does> c@ *,b offset8, ;
: grp4b:    create c, does> c@ *,b offset16, ;

binary

00010000 grp4a: bpl,
00110000 grp4a: bmi,
01010000 grp4a: bvc,
01110000 grp4a: bvs,
10010000 grp4a: bcc,
10110000 grp4a: bcs,
11010000 grp4a: bne,
11110000 grp4a: beq,

10000000 grp4a: bra,    ( although not conditional, it fits the bit pattern )
10000010 grp4b: brl,

: cop,      00000010 *,b *,b ;

: jmp,      01001100 *,b *,w ;  \ JMP $xxxx
: jml,      01011100 *,b *,l ;  \ JML $xxxxxx
: jpi,      01101100 *,b *,w ;  \ JMP ($xxxx)
: jpx,      01111100 *,b *,w ;  \ JMP ($xxxx,X)
: jli,      11011100 *,b *,w ;  \ JMP [$xxxx]

: jsr,      00100000 *,b *,w ;  \ JSR $xxxx
: jsl,      00100010 *,b *,l ;  \ JSL $xxxxxx
: jsx,      11111100 *,b *,w ;  \ JSR ($xxxx,X)

: mvn,      01010100 *,b *,b *,b ;
: mvp,      01000100 *,b *,b *,b ;

: pea,      11110100 *,b *,w ;
: pei,      11010100 *,b *,b ;
01100010 grp4b: per,

: rep,      11000010 *,b *,b ;
: sep,      11100010 *,b *,b ;

: stza,     10011100 *,b *,w ; \ STZ $xxxx
```

```
: stzz,     01100100 *,b *,b ; \ STZ $xx
: stzzx,    01110100 *,b *,b ; \ STZ $xx,X
: stzax,    10011110 *,b *,w ; \ STZ $xxxx,X

: wdm,      01000010 *,b *,b ;

\ ============================================================================
\ Conveniences and structured control flow
\ ============================================================================
decimal

: blt,      bcc, ;
: bge,      bcs, ;
: cma,      cmp, ;
: swa,      xba, ;

: ifcs,     *@ dup 2+ bcc, ;
: ifcc,     *@ dup 2+ bcs, ;
: ifvs,     *@ dup 2+ bvc, ;
: ifvc,     *@ dup 2+ bvs, ;
: ifmi,     *@ dup 2+ bpl, ;
: ifpl,     *@ dup 2+ bmi, ;
: ifeq,     *@ dup 2+ bne, ;
: ifne,     *@ dup 2+ beq, ;

: then,     *@ over 2+ - swap 1+ poke ;

: iflt,     ifcc, ;
: ifge,     ifcs, ;

: begin,    *@ ;
: whilemi,  ifmi, swap ;
: whilepl,  ifpl, swap ;
: whilevs,  ifvs, swap ;
: whilevc,  ifvc, swap ;
: whilecs,  ifcs, swap ;
: whilecc,  ifcc, swap ;
: whileeq,  ifeq, swap ;
: whilene,  ifne, swap ;
: again,    jmp, ;
: repeat,   again, then, ;
: untilmi,  bpl, ;
: untilpl,  bmi, ;
: untilvs,  bvc, ;
: untilvc,  bvs, ;
: untilcs,  bcc, ;
: untilcc,  bcs, ;
: untileq,  bne, ;
: untilne,  beq, ;

: :=        *@ constant ;

\ ============================================================================
\ Done.
\ ============================================================================

base !
```