```
                    B.Y.O.ASSEMBLER
            Part 2:  A 6809 Forth Assembler

                  by Brad Rodriguez
```

A. INTRODUCTION

   Part 1 of this two-part series described the fundamental
   concepts used when writing postfix, structured assemblers in
   Forth. This article examines, in detail, an actual
   assembler for the Motorola 6809 microprocessor.

   I will begin with the "programmer's guide" for the 6809
   assembler. I've found, when building assemblers, that it's
   helpful to write this first. This is good practice, and is
   also in keeping with Rodriguez' First Rule for Metacompiler
   Writers: always keep in mind what you want the result to
   look like!

   In the off-chance that someone actually wishes to use this
   assembler on a 6809 project, I've also included a conversion
   chart from Motorola to Forth notation. If you're ever
   writing an assembler for others to use, please do them a
   favor and make one of these charts!

   Finally, I'll go over the source code, and (hopefully)
   explain all of the obscure tricks and techniques.

   The complete source code for the 6809 assembler accompanies
   this article. It is written for a fig-Forth derivative, and
   so will require translation for 79- or 83-Standard machines.
   This code was originally a Forth screen file; it has been
   translated to a text file for the editor's convenience.

B. PROGRAMMER'S GUIDE

   The syntax of 6809 assembler instructions is:

           operand addressing-mode opcode

   For many instructions the addressing mode is optional. Some
   instructions (TFR and EXG) have two operands.

   Valid non-indexed addressing modes are:

           nn #   immediate value
           nn <>   direct (DP-page) addressing
           nnnn   extended addressing
           nnnn []   extended indirect addressing

   Valid indexed addressing modes are:

           r 0,   zero offset
           r nn ,   constant offset
           r A,   accumulator A offset
           r B,   accumulator B offset
           r D,   accumulator D offset
           r ,+   autoincrement by 1
           r ,++   autoincrement by 2
           r ,-   autodecrement by 1
           r ,--   autodecrement by 2
           nn ,PCR   constant offset from PC

           r is one of the registers X, Y, U, or S
           nn is a twos-complement (signed) offset

   All of the indexed modes except autoincrement/decrement by 1
   also have an Indirect form. This is specified by appending
   the suffix [] after the addressing mode, e.g.:

           r ,++ []   indirect autoincrement by 2

```
        The TFR and EXG instructions take the form:

                src dst TFR         src dst EXG

        where src and dst can be any of the 16-bit registers D, X,
        Y, U, S, PC; or any of the 8-bit registers A, B, CCR, DPR.

        Branch offsets for relative jumps are computed internally by
        the assembler; the operand for a relative jump is the
        destination address.

        The following control structures are provided by the
        assembler:

                cc IF,   ...   THEN,
                        do code if cc satisfied

                cc IF,   ..1..   ELSE,   ..2..   THEN,
                        do code ..1.. if cc satisfied, else do
                        code ..2..

                BEGIN,   ...   cc UNTIL,
                        loop through code until cc satisfied;
                        always executes at least once

                BEGIN,   ..1..   cc WHILE,   ..2..   REPEAT,
                        loop through code until cc satisfied;
                        exit is evaluated and taken after code
                        ..1.. is executed (always at least once)

        where the condition code cc is any of the following:
            CC   carry clear        LS   lower or same
            CS   carry set          LT   less than
            EQ   equal/zero         MI   minus
            GE   greater or equal   NE   not equal/not zero
            GT   greater            PL   plus
            HI   higher             ALW  always
            HS   higher or same     NVR  never
            LE   less than or equal VC   overflow clear
            LO   lower              VS   overflow set

        Infinite loops should use the 'NVR' (never true) condition
        code:

                BEGIN,   ...   NVR UNTIL,


    C. FIGURE ONE: FORTH AND MOTOROLA ASSEMBLERS COMPARISON CHART

        This chart shows the Forth assembler's equivalent for all of
        the Motorola assembler instructions and addressing modes.
        This is not an exhaustive permutation of addressing modes
        and instructions; it is merely intended to illustrate the
        syntax for all possible addressing modes in each instruction
        group.

        Refer to a 6809 data sheet for descriptions of allowable
        operands, operand ranges, and addressing modes for each
        instruction.

        Instructions which require two operands (TFR and EXG) have
        the operands specified in the order: source, destination.
        (This is only significant for the TFR instruction.)


                    INDEXED ADDRESSING MODES

                -NON-INDIRECT--    ---INDIRECT----
        TYPE        MOTOROLA FORTH    MOTOROLA FORTH    POSTBYTE

        no offset      ,r     r 0,    [,r]     r 0, []  1rri0100
        5 bit offset   n,r    r n ,   defaults to 8-bit 0rrnnnnn
        8 bit offset   n,r    r n ,   [n,r]    r n , []  1rri1000
```

```
  16 bit offset   n,r     r n ,    [n,r]    r n , []  1rri1001

  A-reg offset    A,r     r A,     [A,r]    r A, []   1rri0110
  B-reg offset    B,r     r B,     [B,r]    r B, []   1rri0101
  D-reg offset    D,r     r D,     [D,r]    r D, []   1rri1011

  incr. by 1      ,r+     r ,+      not allowed       1rr00000
  incr. by 2      ,r++    r ,++    [,r++]   r ,++ []  1rri0001
  decr. by 1      ,r-     r ,-      not allowed       1rr00010
  decr. by 2      ,r--    r ,--    [,r--]   r ,-- []  1rri0011

  PC ofs. 8 bit   n,PCR   n ,PCR   [n,PCR]  n ,PCR [] 1xxi1100
  PC ofs. 16 bit  n,PCR   n ,PCR   [n,PCR]  n ,PCR [] 1xxi1101

  16 bit address    not allowed    [n]      n []      10011111

  where n = a signed integer value,
      r = X (00), Y (01), U (10), or S (11)
      x = don't care
```

                        INSTRUCTION SET

                Inherent addressing group

```
MOTOROLA        FORTH        MOTOROLA        FORTH

ABX             ABX,         MUL             MUL,
ASLA            ASLA,        NEGA            NEGA,
ASLB            ASLB,        NEGB            NEGB,
ASRA            ASRA,        NOP             NOP,
ASRB            ASRB,        ORA             ORA,
CLRA            CLRA,        ORB             ORB,
CLRB            CLRB,        ROLA            ROLA,
COMA            COMA,        ROLB            ROLB,
COMB            COMB,        RORA            RORA,
DAA             DAA,         RORB            RORB,
DECA            DECA,        RTI             RTI,
DECB            DECB,        RTS             RTS,
INCA            INCA,        SEX             SEX,
INCB            INCB,        SWI             SWI,
LSLA            LSLA,        SWI2            SWI2,
LSLB            LSLB,        SWI3            SWI3,
LSRA            LSRA,        SYNC            SYNC,
LSRB            LSRB,        TSTA            TSTA,
                            TSTB            TSTB,
```

                Register-register group

```
MOTOROLA        FORTH        MOTOROLA        FORTH

EXG s,d         s d EXG      TFR s,d         s d TFR
```

                Immediate-addressing-only group

```
MOTOROLA        FORTH        MOTOROLA        FORTH

ANDCC #n        n # ANDCC,   PSHS regs       n # PSHS,
CWAI #n         n # CWAI,    PSHU regs       n # PSHU,
ORCC #n         n # ORCC,    PULS regs       n # PULS,
                            PULU regs       n # PULU,
```

    Note: Motorola allows the PSH and PUL instructions to
    contain a register list. The Forth assembler requires
    the programmer to compute the bit mask for this list
    and supply it as an immediate argument.

                Indexed-addressing-only group
                (with example addressing modes)

```
MOTOROLA        FORTH        MOTOROLA        FORTH

LEAS D,U        U ,D LEAS,   LEAX [,S++]     S ,++ [] LEAX,
```

```
        LEAU -5,Y       Y -5 , LEAU,    LEAY [1234]    1234 [] LEAY,

                        General-addressing group
                     (with example addressing modes)

        MOTOROLA        FORTH          MOTOROLA        FORTH

        ADCA #20        20 # ADCA,     LDA #20         20 # LDA,
        ADCB <30        30 <> ADCB,    LDB <30         30 <> LDB,
        ADDA 2000       2000 ADDA,     LDD 2000        2000 LDD,
        ADDB [1030]     1030 [] ADDB,  LDS [1030]      1030 [] LDS,
        ADDD ,S         S 0, ADDD,     LDU ,X          X 0, LDU,
        ANDA 23,U       U 23 , ANDA,   LDX 23,Y        Y 23 , LDX,
        ANDB A,X        X A, ANDB,     LDY A,S         S A, LDY,
        ASL  B,Y        Y B, ASL,      LSL B,U         U B, LSL,
        ASR  D,X        X D, ASR,      LSR D,S         S D, LSR,
        BITA ,S+        S ,+ BITA,     NEG ,X+         X ,+ NEG,
        BITB ,X++       X ,++ BITB,    ORA ,S++        S ,++ ORA,
        CLR  ,Y-        Y ,- CLR,      ORB ,U-         U ,- ORB,
        CMPA ,U--       U ,-- CMPA,    ROL ,Y--        Y ,-- ROL,
        CMPB -5,PCR     -5 ,PCR CMPB,  ROR 12,PCR      12 ,PCR ROR,
        CMPD [,Y]       Y 0, [] CMPD,  SBCA [,U]       U 0, [] SBCA,
        CMPS [7,Y]      Y 7 , [] CMPS, SBCB [7,U]      U 7 , [] SBCB,
        CMPU [A,S]      S A, [] CMPU,  STA [A,X]       X A, [] STA,
        CMPX [B,U]      U B, [] CMPX,  STB [B,Y]       Y B, [] STB,
        CMPY [D,X]      X D, [] CMPY,  STD [D,S]       S D, [] STD,
        EORA [,Y+]      Y ,+ [] EORA,  STS [,U+]       U ,+ [] STS,
        EORB [,U++]     U ,++ [] EORB, STU [,Y++]      Y ,++ [] STU,
        COM  [,S-]      S ,- [] COM,   STX [,S-]       S ,- [] STX,
        DEC [,X--]      X ,-- [] DEC,  STY [,X--]      X ,-- [] STY,
        INC [5,PCR]     5 ,PCR [] INC, SUBA [3,PCR]    3 ,PCR [] SUBA,
        JMP [300]       300 [] JMP,    SUBB [300]      300 [] SUBB,
        JSR 1234        1234 JSR,      SUBD 1234       1234 SUBD,
                                       TST #2          2 # TST,
```

        Note that, in the Forth assembler,
            #   signifies Immediate addressing, and
            <> signifies Direct addressing.

        Many instructions do not allow immediate addressing.
        Refer to the Motorola data sheet.

                        Branch instructions

```
        MOTOROLA        FORTH          MOTOROLA        FORTH

        BCC label       adrs BCC,      BLT label       adrs BLT,
        BCS label       adrs BCS,      BMI label       adrs BMI,
        BEQ label       adrs BEQ,      BNE label       adrs BNE,
        BGE label       adrs BGE,      BPL label       adrs BPL,
        BGT label       adrs BGT,      BRA label       adrs BRA,
        BHI label       adrs BHI,      BRN label       adrs BRN,
        BHS label       adrs BHS,      BSR label       adrs BSR,
        BLE label       adrs BLE,      BVC label       adrs BVC,
        BLO label       adrs BLO,      BVS label       adrs BVS,
        BLS label       adrs BLS,
```

        The branch instructions in the Forth assembler expect
        an absolute address. The relative offset is computed,
        and the "long branch" form of the instruction is used
        if necessary.

 D. INTERNAL GLOSSARY AND DESCRIPTION

    This assembler was written before I acquired the habit of
    "shadow screen" documentation. So, I'll document all of the
    unusual words and features here. (Please refer to the
    program listing.)

    The word WITHIN is a common Forth extension. The words
    5BIT? and 8BIT? decide if a given value will fit in a 5-bit
    or 8-bit signed integer, so we can choose the correct
    indexed addressing mode.

The synonym (,) is defined because later I redefine , as an
addressing mode.

ALIGN etc. deserve some comment. This assembler was
originally written for a metacompiler running on a 68000
system, which insisted upon word-aligning the DP after each
Forth word was interpreted. This meant that any 6809
instruction which assembled an odd number of bytes would
have a filler byte added -- with catastrophic results! I
fixed this by causing all of the assembler words to do the
aligning themselves. Thus, the 68000 Forth never inserted
any filler, and I always knew when the DP had been adjusted.
HERE and C, were redefined accordingly.

The word W, allows 6809 word operands to be compiled on
either big-endian or little-endian host machines. >< is a
byte-swap operator provided by the Forth I used.

Some 6809 instructions have a one-byte opcode, and some have
two bytes. Opcodes are stored as a 16-bit value. If the
high 8 bits are nonzero, they are the first opcode byte.
OPCODE, lays down one or two bytes, accordingly.

The MODE variable indicates whether the addressing mode is
Immediate, Direct, Indexed, or Extended. # and <> set the
first two modes; Extended is the default when no mode is
set. (This is the first addressing-mode technique described
in the previous article.)

The Indexed addressing modes in the 6809 add a "postbyte" to
the opcode, which contains mode information and a register
number (for X, Y, U, or S). INDEXREG puts the two-bit
register number into the postbyte, and also sets MODE. Note
that the postbyte is passed on the stack. (This is the
second addressing-mode technique.)

XMODE defines the "simple" Indexed modes. These modes each
have a fixed postbyte, modified only by the register number
(as supplied by INDEXREG). For example, the word ,++
fetches the postbyte 81 hex and then invokes INDEXREG to
insert the two register bits.

The word , rearranges the stack and builds the postbyte for
the constant-offset Indexed mode.

The word ,PCR provides the postbyte for the PC-relative
Indexed mode. Since this has no register operand, INDEXREG
isn't used. MODE must be explicitly set to 20 hex.

The word [] indicates indirection. For the Indexed
addressing modes, this is done by setting the "indirect" bit
in the postbyte. (This is an example of the third
addressing-mode technique.) For the Direct addressing mode,
[] must change the mode to Indexed and supply the Extended
Indirect postbyte.

Register definitions are simple CONSTANTs. Note that the
synonyms W, IP, RP, and SP are defined for the registers X,
Y, S, and U. I use these synonyms when writing Forth
kernels.

INHOP defines the Inherent (no operands) instructions.

IMMOP defines the Immediate-only instructions. These all
expect a byte operand, and they check to make sure that the
addressing mode was specified (MODE=0).

RROP defines the register-register instructions, TFR and
EXG. Note that it doesn't check that both operands are the
same size; the programmer is presumed to know better.

+MODE is a fudge. All of the general-addressing
instructions form their opcodes by adding 0, 10, 20, or 30

hex to a base value, EXCEPT those instructions whose Direct
opcode takes the form 0x hex. These instructions use 6x for
Indexed mode, and 7x for Extended, so the assembler assumes
a "base opcode" of 4x, and adds 10, 20, or 30 hex. (There
is no Immediate mode for these instructions.) Then, if the
resulting opcode is 5x, we know that Direct mode was
specified, and the opcode should really be 0x hex. +MODE
applies the MODE value to the base opcode; it then checks
for 5x opcodes and changes them to 0x.

PCREL lays the postbyte and operand for PC-relative
addressing. If the signed offset fits in 8 bits, the
postbyte is modified for the short form and a single-byte
operand is used. Otherwise, the long form postbyte and
two-byte operand is used.

NOTINDIR? checks the indirection bit in the postbyte.

COFSET lays the postbyte and operand for constant-offset
Indexed addressing. If the offset fits in 5 bits, and
indirection is not used, then the postbyte is modified for
the 5-bit offset. Otherwise, if the offset fits in 8 bits,
the postbyte is modified for the 8-bit form (single-byte
operand); if the offset requires 16 bits, the long form is
used (two-byte operand).

EXTIND lays the postbyte and operand for Extended Indirect
addressing.

INDEXED lays the postbyte and (if required) operand for all
of the Indexed modes. COFSET, PCREL, and EXTIND are the
special cases which require operands; a CASE statement
identifies these by the postbyte value. All other postbyte
values are "simple" modes which have no operands; they are
handled in the default clause of the CASE statement.

Some 6809 Immediate instructions require an 8-bit operand,
and some a 16-bit operand. Others don't allow Immediate
mode. IMMED handles this by testing the second parmeter in
an opcode word defined by GENOP, and laying one or two
operand bytes, accordingly. If "zero" operand bytes are
indicated, this means that this addressing mode is invalid
with this instruction.

GENOP defines the instructions which can have any addressing
mode. It selects one of four actions depending on MODE. As
noted above, the parameter "immedsize" can be used to
disallow Immediate mode for any instruction.

INXOP defines the instructions which can have Indexed
addressing mode. It checks that the MODE value is correct
(20 hex) before assembling the instruction.

CONDBR and UNCBR build the branch instructions. They are
identical, except in how they modify the instruction when
the long form is required. CONDBR makes the long form by
prefixing a 10 hex byte; UNCBR makes the long form by
substituting an alternate opcode (both opcodes are stored in
a 16-bit value). Both of these words take an absolute
address, and compute the relative offset from the branch
instruction; the short or long form of the branch is then
automatically chosen.

CS through NVR are condition code CONSTANTs for the
structured conditionals. These are actually the opcodes
which must be assembled by the conditionals such as IF, and
UNTIL, . As noted in the previous article, IF, and UNTIL,
must actually assemble the logical inverse of the stated
condition; this is handled in the 6809 assembler by defining
each of these constants to contain the "inverse" opcode.
For example, the constant CS (carry set) actually contains
the opcode for a BCC instruction. This is because the
phrase CS IF, must assemble a BCC.

Structured conditionals were described in detail in the
previous article. Since the condition codes are in fact
opcodes, the requisite conditional jumps can be assembled
directly with C, . Also, these conditionals use the
fig-Forth "compiler security": each conditional leaves a
constant value on the stack, which much be verified (by
?PAIRS) by its matching word. For example, IF, leaves 2 on
the stack; ELSE, and ENDIF, (a.k.a. THEN,) do a 2 ?PAIRS to
resolve this.

ENTERCODE CODE and ;CODE are, of necessity, "tuned" to a
particular Forth implementation. For example, since this
fig-Forth model's CREATE automatically sets up the code
field properly, it's not necessary for CODE to patch the
code field to point to the new machine code. Other Forth
models have a different assumption, so some phrase such as
        CREATE HERE 2- !
will no doubt be needed to set the code field pointer.

ENTERCODE (and thus CODE and ;CODE) also use !CSP to check
for stack imbalances. This means that each CODE definition
must end with ;C (which uses ?CSP to resolve !CSP). ;C also
unSMUDGEs the Forth word being defined. These are also
fig-Forth usages, which may not apply to your Forth model.

Finally, NEXT, is an example of a simple assembler macro.
It assembles the two-instruction sequence which is the inner
interpreter (NEXT) of an indirect-threaded 6809 Forth. NEXT
is a synonym.