The Z80 instruction at BEGIN and BEGIN1 (LD IX,LABL1–1) illustrates the usefulness of an assembler program. Here, it decodes the expression (LABL1 –1) to mean 'the address of the byte immediately before the byte whose address is LABL1', and assembles that address into the code. Most assemblers support some measure of expression evaluation, usually allowing one or two operands to be modified by a single arithmetic operator — normally '+' or '–'.

2) This program reverses the order of characters in each word of the string at LABL1, while maintaining the order of the words themselves:

6502

```
;
ORIGIN   ORG   $7000
LAST1    EQU   $0D
SPACE    EQU   $20
LABL1    DB    'THIS IS'
TERMN8   DB    LAST1
;
BEGIN    LDX   #$FF
LOOP0    JSR   RVSWRD
         CMP   #LAST1
ENDLP0   BNE   LOOP0
         RTS
;
;****REVERSE A WORD S/R****
LASTCH   DB    $00
LASTX    DB    $00
RVSWRD   TXA
         TAY
         INY
RVSLP0   INX
         LDA   LABL1,X
         PHA
         CMP   #SPACE
         BEQ   CLRSTK
         CMP   #LAST1
ENDRV0   BNE   RVSLP0
CLRSTK   PLA
         STA   LASTCH
         STX   LASTX
RVSLP1   PLA
         STA   LABL1,Y
         INY
         CPY   LASTX
ENDLP1   BNE   RVSLP1
         LDA   LASTCH
         RTS
```

There are several points of interest here: the use of JSR and CALL instructions, for example. The RVSWRD subroutine is similar in structure to the program given in Exercise 1, but it reverses only the characters of a word, not the whole string. In both the 6502 and Z80 versions, the index register (X and IX respectively) is used to pass the start address of the word to the subroutine, and the accumulator is used to pass back to the calling program the value of the character that terminated the work (either a space or the string terminator character). Passing values this way is a very common Assembly language technique, and must be used with care — especially if you are in the habit of pushing all CPU registers at the start of every subroutine (as demonstrated on page 258).

Another significant feature is the use of the Y register in the 6502 version, first to hold the start address of the word while X is used as an index on the stacking loop, then as an index on the 'un-stacking' loop while X holds the end address of the word. 'Address' is used imprecisely here as X and Y are single-byte registers, so neither can hold a full address. Instead, in this case they hold an offset to the address LABL1. In contrast, the Z80 IX and IY index registers can hold a full two-byte address.

In the Z80 version, IX and IY are not used at all — the HL and DE register pairs are used instead. Like the 6502 X and Y registers, these hold the word start and end addresses, but instead of being indexes on a base address, they are used as indirect addresses (the instruction LD A,(DE) means 'load the accumulator from the byte whose address is held in DE'). All the Z80 register pairs can be used in this way. An odd limitation of the instruction set is the lack of any two-byte comparison instruction. Thus, comparing the contents of DE and HL involves comparing E with L, then D with H. Similarly, in the 6502 version, X and Y are compared indirectly using a memory location, since there is no instruction for comparing X with Y.

Z80

```
         ORG   $C000
LAST1    EQU   $0D
SPACE    EQU   $20
LABL1    DB    'THIS IS A MESSAGE'
TERMN8   DB    LAST1
;
BEGIN    LD    DE,LABL1-1
LOOP0    CALL  RVSWRD
         CP    LAST1
ENDLP0   JR    NZ,LOOP0
         RET
;
;***REVERSE A WORD S/R***
LASTCH   DB    $00
RVSWRD   PUSH  DE
         POP   HL
         INC   HL
RVSLP0   INC   DE
         LD    A,(DE)
         PUSH  AF
         CP    SPACE
         JR    Z,CLRSTK
         CP    LAST1
ENDRV0   JR    NZ,RVSLP0
CLRSTK   POP   AF
         LD    (LASTCH),A
;
RVSLP1   POP   AF
         LD    (HL),A
         INC   HL
         LD    A,L
         CP    E
         JR    NZ,RVSLP1
         LD    A,H
         CP    D
ENDRV1   JR    NZ,RVSLP1
         LD    A,(LASTCH)
         RET
```