ARITHMETIC, LOGIC INSTRUCTIONS

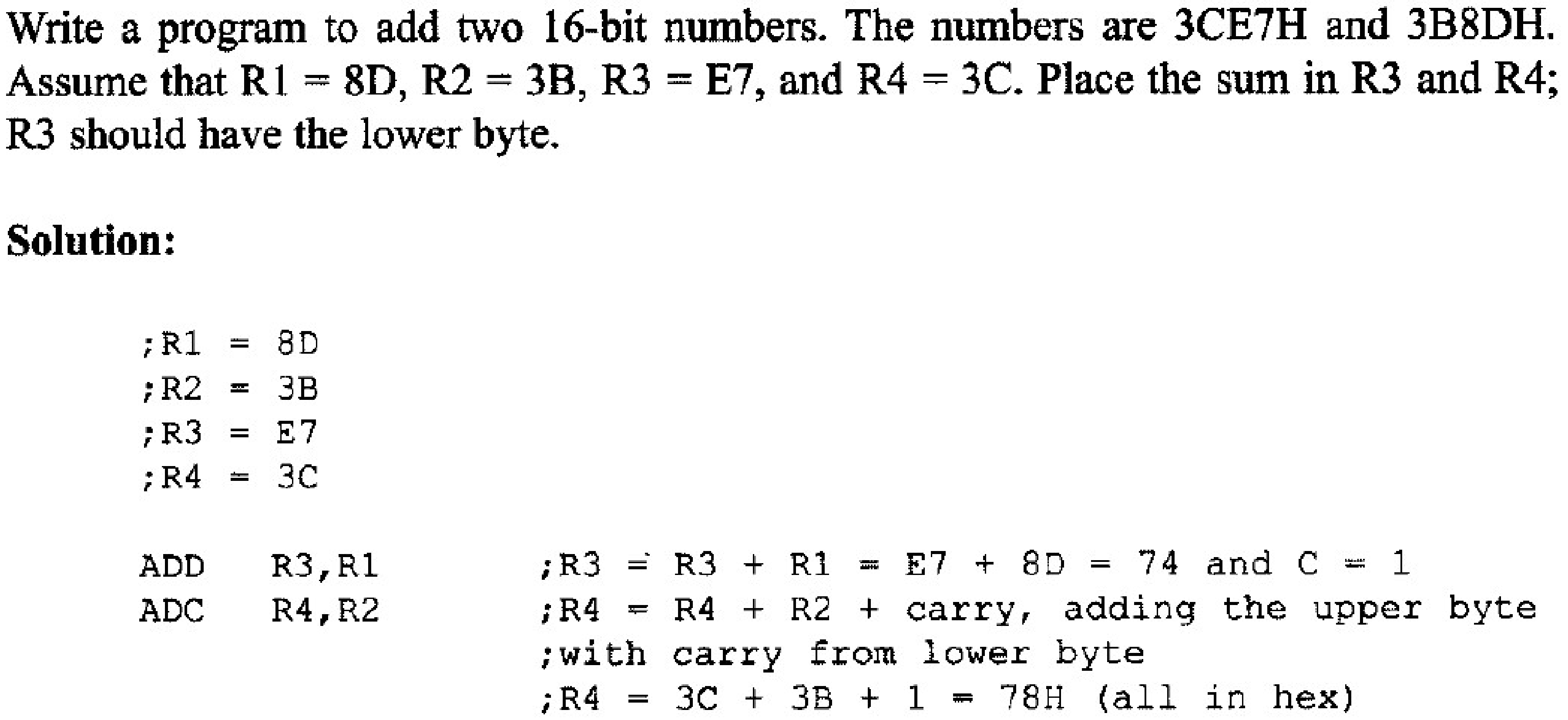
ARITHMETIC INSTRUCTIONS

- Unsigned numbers are defined as data in which all the bits are used to represent data and no bits are set aside for the positive or negative sign.
- This means that the operand can be between 00 and FFH (0 to 255 decimal) for 8-bit data.

ARITHMETIC INSTRUCTIONS

- **Addition of unsigned numbers**
- **ADC and addition of 16-bit numbers**
  - When adding two 16-bit data operands, we need to be concerned with the propagation of a carry from the lower byte to the higher byte.
  - This is called *multibyte addition* to distinguish it from the addition of individual bytes.
  - The instruction ADC (ADD with carry) is used on such occasions

Example 5-3



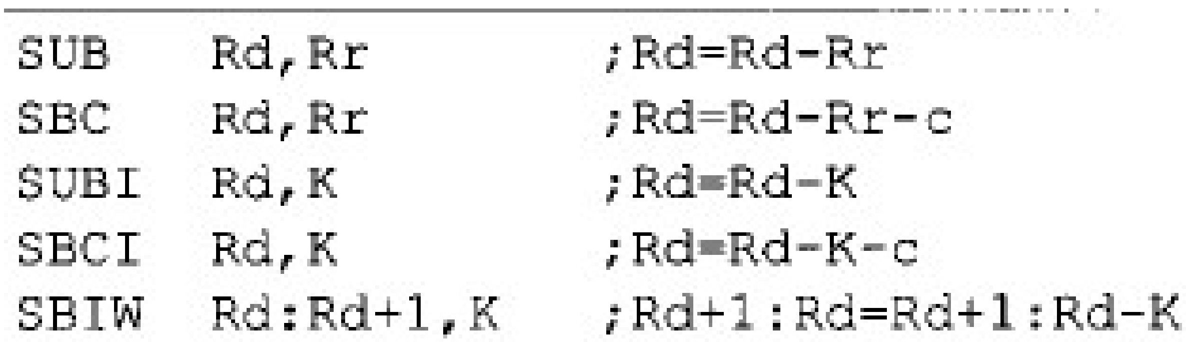
Write a program to add two 16-bit numbers. The numbers are 3CE7H and 3B8DH. Assume that R1 = 8D, R2 = 3B, R3 = E7, and R4 = 3C. Place the sum in R3 and R4; R3 should have the lower byte.

Solution:

```
;R1  =  8D
;R2  =  3B
;R3  =  E7
;R4  =  3C

ADD   R3,R1        ;R3 = R3 + R1 = E7 + 8D = 74 and C = 1
ADC   R4,R2        ;R4 = R4 + R2 + carry, adding the upper byte
                   ;with carry from lower byte
                   ;R4 = 3C + 3B + 1 = 78H (all in hex)
```

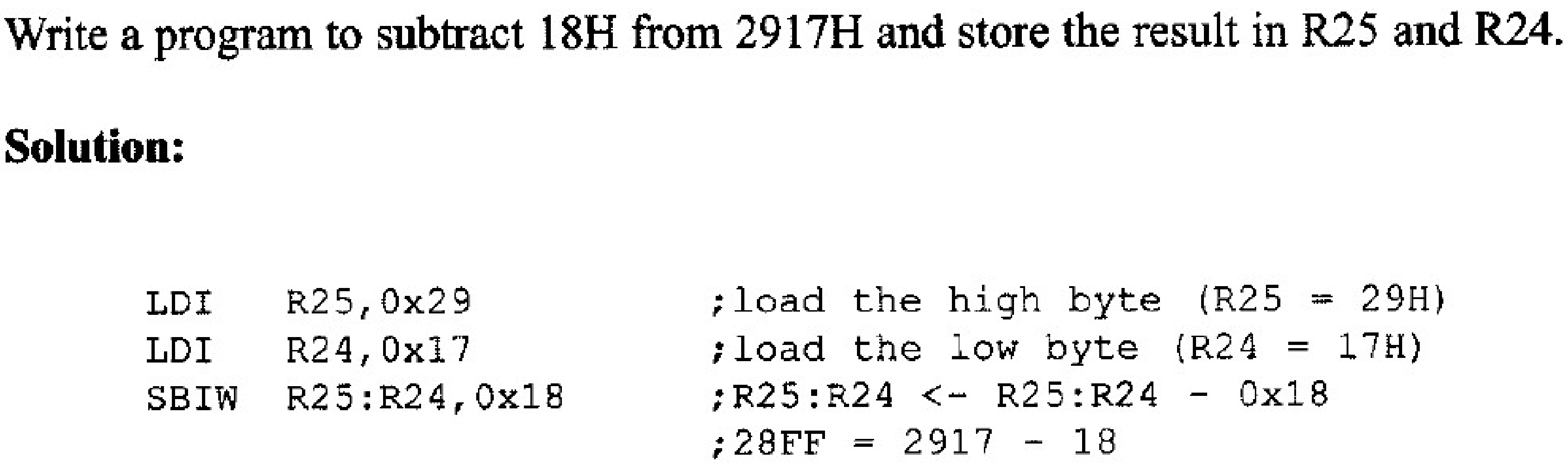
ARITHMETIC INSTRUCTIONS

- Subtraction of unsigned numbers
  - In the AVR we have five instructions for subtraction:
    - SUB
    - SBC
    - SUBI
    - SBCI
    - SBIW



```
SUB    Rd,Rr          ;Rd=Rd-Rr
SBC    Rd,Rr          ;Rd=Rd-Rr-c
SUBI   Rd,K           ;Rd=Rd-K
SBCI   Rd,K           ;Rd=Rd-K-c
SBIW   Rd:Rd+1,K      ;Rd+1:Rd=Rd+1:Rd-K
```

Example 5-6

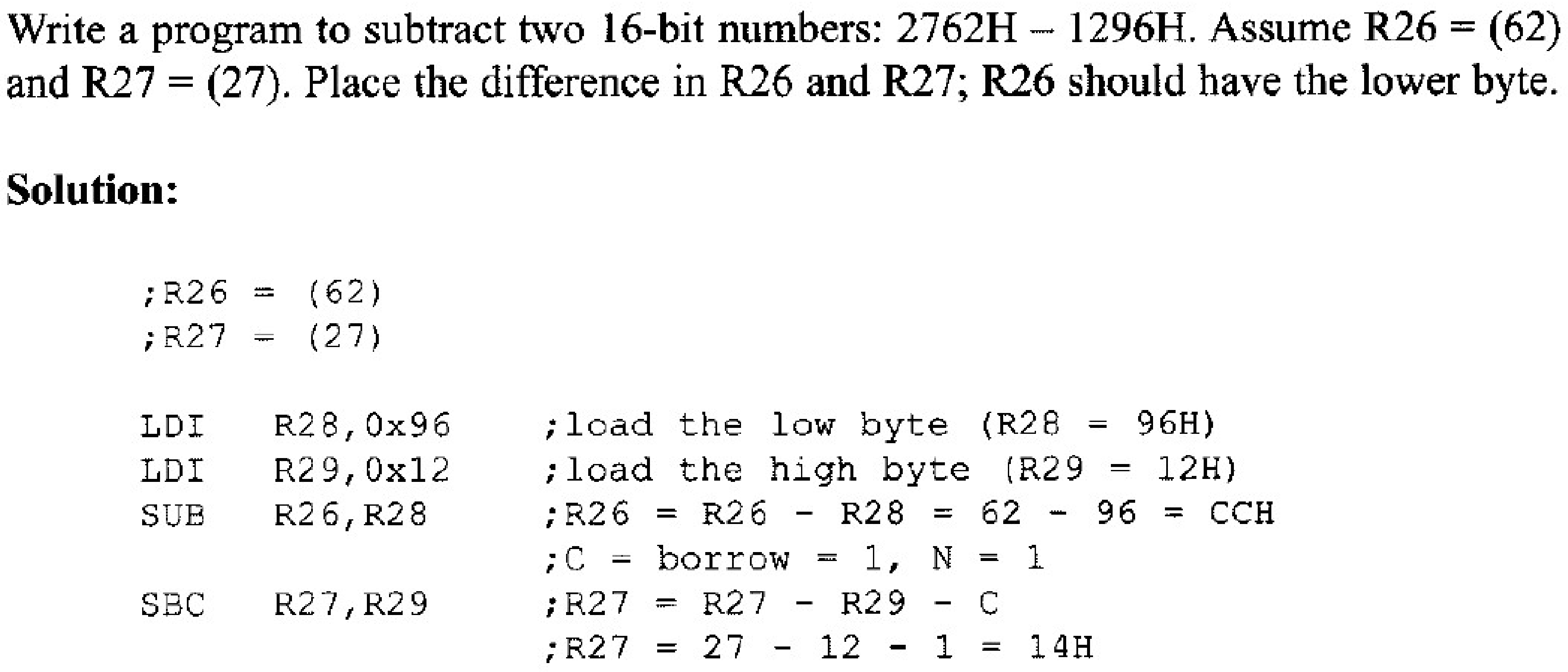


Write a program to subtract 18H from 2917H and store the result in R25 and R24.

Solution:

```
LDI   R25,0x29          ;load the high byte (R25 = 29H)
LDI   R24,0x17          ;load the low byte (R24 = 17H)
SBIW  R25:R24,0x18      ;R25:R24 <- R25:R24 - 0x18
                        ;28FF = 2917 - 18
```

Example 5-7



Write a program to subtract two 16-bit numbers: 2762H – 1296H. Assume R26 = (62) and R27 = (27). Place the difference in R26 and R27; R26 should have the lower byte.

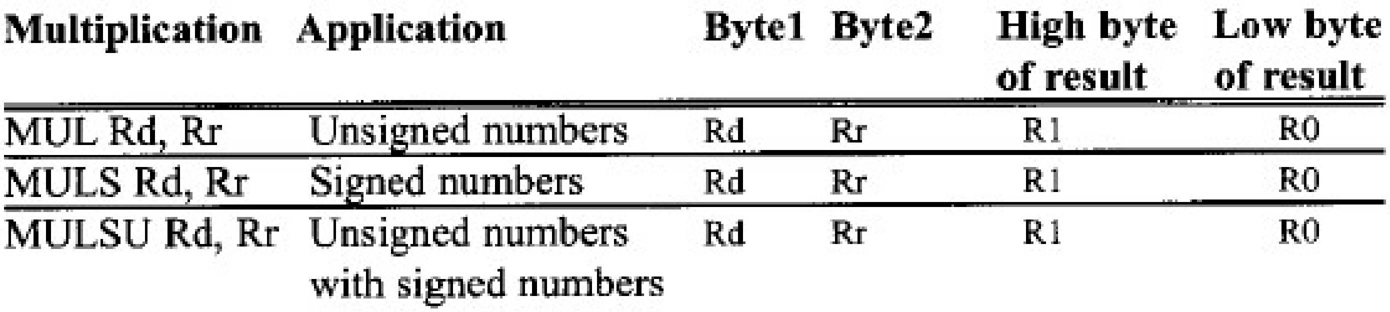
Solution:

```
;R26 = (62)
;R27 = (27)

LDI   R28,0x96     ;load the low byte (R28 = 96H)
LDI   R29,0x12     ;load the high byte (R29 = 12H)
SUB   R26,R28      ;R26 = R26 - R28 = 62 - 96 = CCH
                   ;C = borrow = 1, N = 1
SBC   R27,R29      ;R27 = R27 - R29 - C
                   ;R27 = 27 - 12 - 1 = 14H
```

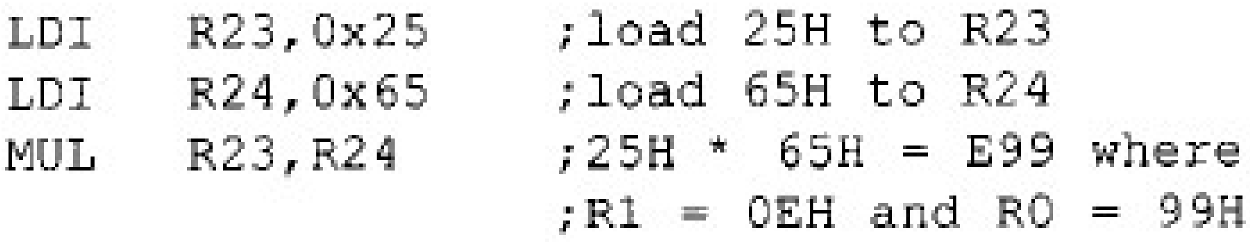
Multiplication

- MUL is a byte-by-byte multiply instruction.
- In byte-by-byte multiplication, operands must be in registers.
  - After multiplication, the 16-bit unsigned product is placed in R1 (high byte) and R0 (low byte).
  - Notice that if any of the operands is selected from R0 or R1 the result will overwrite those registers after multiplication.

Multiplication



Multiplication	Application	Byte1	Byte2	High byte of result	Low byte of result
MUL Rd, Rr	Unsigned numbers	Rd	Rr	R1	R0
MULS Rd, Rr	Signed numbers	Rd	Rr	R1	R0
MULSU Rd, Rr	Unsigned numbers with signed numbers	Rd	Rr	R1	R0

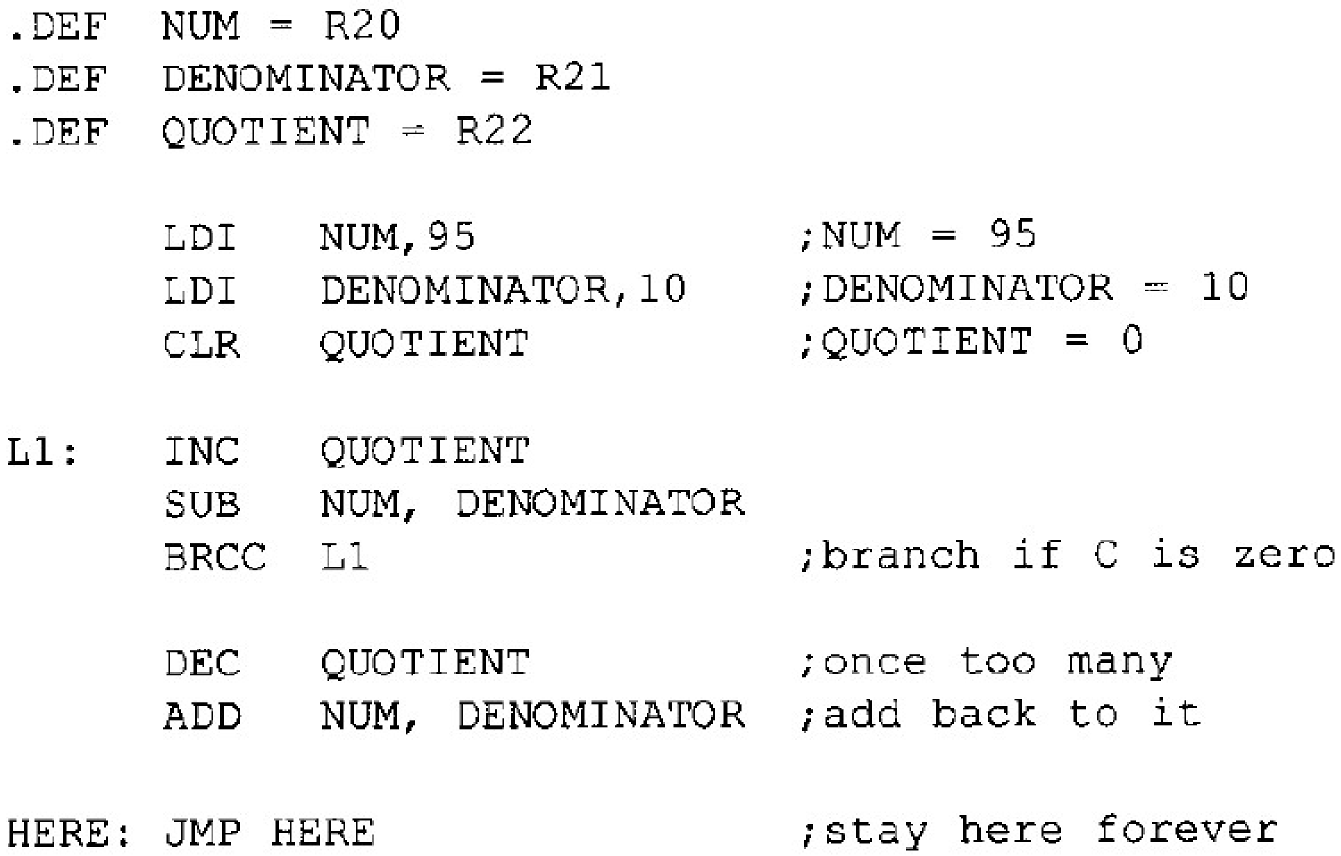


```
LDI    R23,0x25     ;load 25H to R23
LDI    R24,0x65     ;load 65H to R24
MUL    R23,R24      ;25H * 65H = E99 where
                    ;R1 = 0EH and R0 = 99H
```

Division of unsigned numbers

- AVR has no instruction for divide operation.
- We can write a program to perform division by repeated subtraction.
  - In dividing a byte by a byte, the numerator is placed in a register and the denominator is subtracted from it repeatedly.
  - The quotient is the number of times we subtracted and the remainder is in the register upon completion.

Division of unsigned numbers

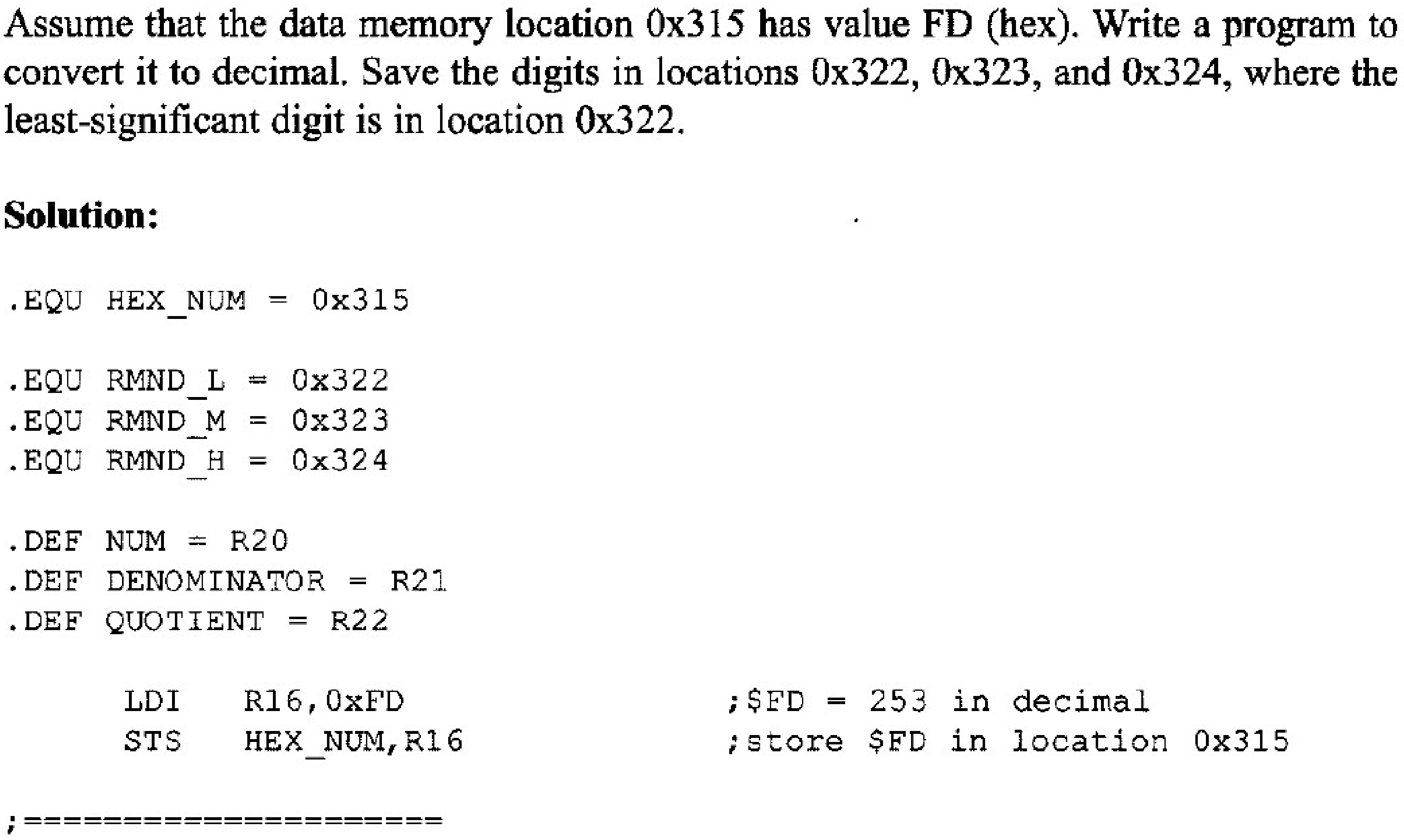


```
.DEF   NUM = R20
.DEF   DENOMINATOR = R21
.DEF   QUOTIENT = R22

       LDI    NUM,95              ;NUM = 95
       LDI    DENOMINATOR,10      ;DENOMINATOR = 10
       CLR    QUOTIENT            ;QUOTIENT = 0

L1:    INC    QUOTIENT
       SUB    NUM, DENOMINATOR
       BRCC   L1                  ;branch if C is zero

       DEC    QUOTIENT            ;once too many
       ADD    NUM, DENOMINATOR    ;add back to it

HERE: JMP HERE                    ;stay here forever
```

An application for division

- Converted Hex numbers to decimal.
- We do that by dividing it by 10 repeatedly, saving the remainders.

Example 5-8

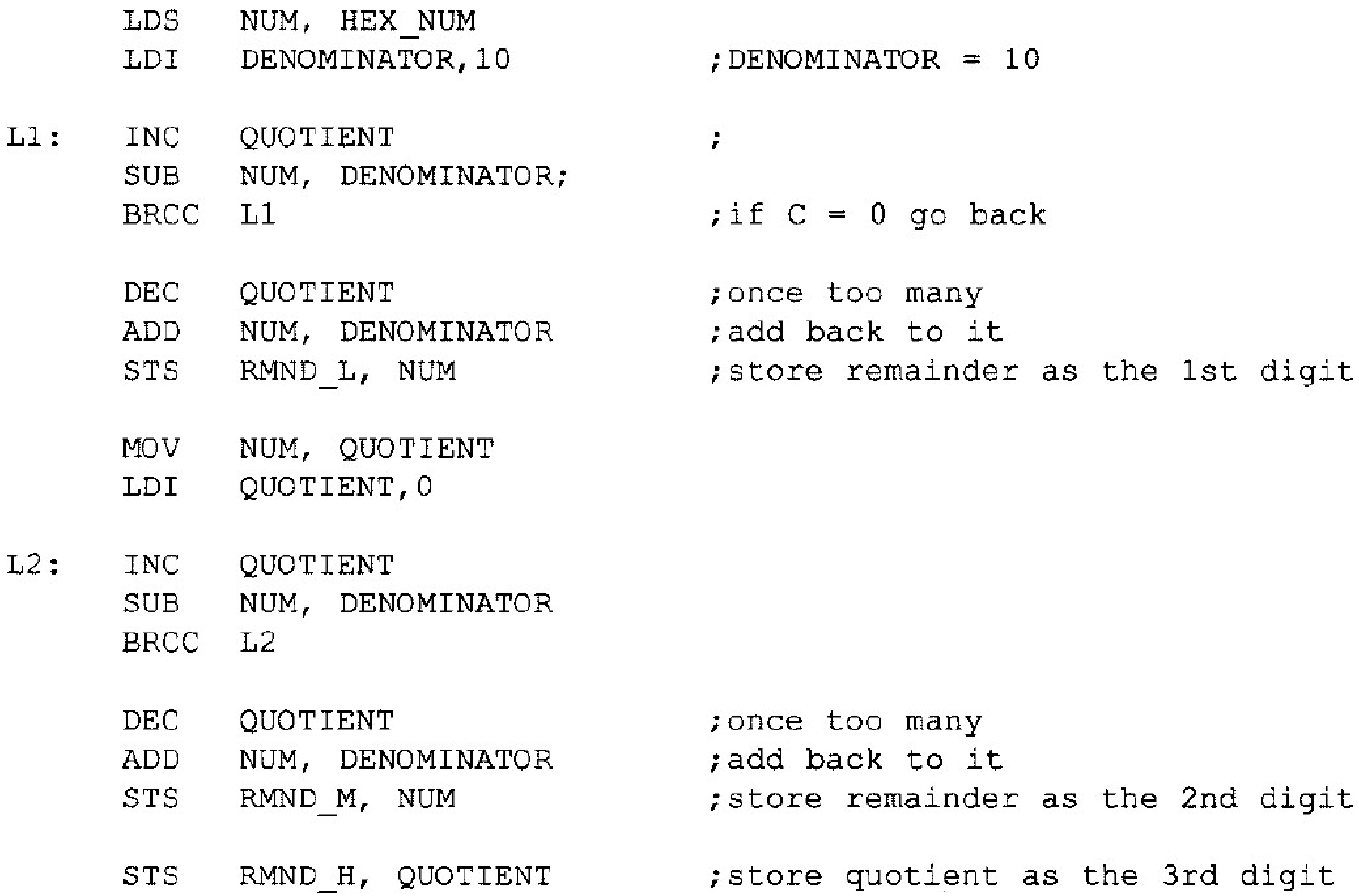


Assume that the data memory location 0x315 has value FD (hex). Write a program to convert it to decimal. Save the digits in locations 0x322, 0x323, and 0x324, where the least-significant digit is in location 0x322.

Solution:

```
.EQU HEX_NUM = 0x315

.EQU RMND_L = 0x322
.EQU RMND_M = 0x323
.EQU RMND_H = 0x324

.DEF NUM = R20
.DEF DENOMINATOR = R21
.DEF QUOTIENT = R22

      LDI    R16,0xFD                ;$FD = 253 in decimal
      STS    HEX_NUM,R16             ;store $FD in location 0x315

;=====================
```

Example 5-8



```
        LDS     NUM, HEX_NUM
        LDI     DENOMINATOR,10              ;DENOMINATOR = 10

L1:     INC     QUOTIENT                    ;
        SUB     NUM, DENOMINATOR;
        BRCC    L1                          ;if C = 0 go back

        DEC     QUOTIENT                    ;once too many
        ADD     NUM, DENOMINATOR            ;add back to it
        STS     RMND_L, NUM                 ;store remainder as the 1st digit

        MOV     NUM, QUOTIENT
        LDI     QUOTIENT,0

L2:     INC     QUOTIENT
        SUB     NUM, DENOMINATOR
        BRCC    L2

        DEC     QUOTIENT                    ;once too many
        ADD     NUM, DENOMINATOR            ;add back to it
        STS     RMND_M, NUM                 ;store remainder as the 2nd digit

        STS     RMND_H, QUOTIENT            ;store quotient as the 3rd digit
```