OPÉRATIONS ARITHMÉTIQUES SUR MICROCONTROLEURS 8 BITS

```
; Multiplication 16 X 16 de R6:R7 X R4:R5
; résultat 32 bits dans R4:R5:R6:R7, R0 est utilisé, R1, R2, R3 inchangés
; Multiplicande X multiplicateur MSBM:LSBM X MSBm:LSBm
;        MSBM:LSBM (R6:R7)
;  X     MSBm:LSBm (R4:R5)
;  ---------------
;        LSBMxLSBm      (MSB1:LSB1)   RES1 =                    LSB1
;     MSBMxLSBm         (MSB2:LSB2)   RES2 =    MSB1+LSB2+LSB3
;     LSBmxMSBm         (MSB3:LSB3)   RES3 =    MSB2+MSB3+LSB4+retenue
;  MSBMxMSBm            (MSB4:LSB4)   RES4 = MSB4+retenue
;  ---------------
;  RES4:RES3:RES2:RES1
; Taille de MUL16 : 41 octets

MUL16:  MOV   A,R7          ; LSBM multiplicande
        MOV   R0,A          ; sauve dans R0
        MOV   B,R5          ; LSBm multiplicateur
        MUL   AB            ; MSB1:LSB1 = B:A = LSBM X LSBm
        MOV   R7,A          ; sauve le RES1
        MOV   A,R5          ; LSBm multiplicateur
        XCH   A,B           ; dans B, dans A le MSB1
        XCH   A,R6          ; sauve dans R6, dans A MSBM multiplicande
        MOV   R1,A          ; sauve MSBM
        MUL   AB            ; MSB2:LSB2 = B:A = MSBM X LSBm
        ADD   A,R6          ; MSB1 + LSB2
        MOV   R6,A          ; RES2 = MSB1 + LSB2
; les trois lignes suivantes ne sont nécessaires que pour un résultat 32 bits
        MOV   A,B           ; MSB2
        ADDC  A,#0          ; MSB2 + retenue partielle de RES2
; FF X FF=FE01, la retenue additionnée à MSB2 l'amène au maximum à FE + 1 = FF
        MOV   R5,A          ; RES3 = MSB2 (inutile si résultat sur 16 bits)
        MOV   A,R4          ; MSBm multiplicateur
        MOV   B,R0          ; LSBM multiplicande
        MUL   AB            ; MSB3:LSB3 = B:A = MSBm X LSBM
        ADD   A,R6          ; RES2 = RES2 + LSB3
        MOV   R6,A          ; sauve RES2 définitif
; arrêt ici si résultat sur 16 bit seul désiré dans R6:R7
        MOV   A,B           ; MSB3
        ADDC  A,R5          ; MSB2 + MSB3 + retenue partielle
        MOV   R5,A          ; sauve RES3 temporaire
        CLR   A
        RLC   A             ; prend la retenue de RES3
        XCH   A,R4          ; échange avec MSBm (sauve RES4)
        MOV   B,R1          ; MSBM
        MUL   AB            ; MSB4:LSB4 = B:A = MSBM X MSBm
        ADD   A,R5          ; LSB4 + RES3
        MOV   R5,A          ; RES3 = retenue + MSB2 + MSB3 + LSB4
        MOV   A,B           ; MSB4
        ADDC  A,R4          ; MSB4 + retenue(s)
        MOV   R4,A          ; sauve MSB4
; résultat dans R4:R5:R6:R7
        RET
```

■ ***Listing multiplication 16x16***

Les opérations arithmétiques sur les microprocesseurs sont souvent limitées aux opérations sur 8 bits. Cette précision est souvent insuffisante et il est nécessaire de créer ses sous-programmes pour faire des opérations avec une précision supérieure.

Les sous-programmes d'addition et de soustraction ne posent pas de problème, par contre la multiplication et la division sont plus délicates.

Dans 90% des applications, une précision de 16 bits est suffisante, ce qui nécessite des multiplications de 16x16 avec résultat sur 32 bits et des divisions 32/32 avec un résultat sur 16 ou 32 bits. Quand on dispose d'une multiplication 8x8 (cas du 8051, 68HC05, 68HC11), la multiplication multiprécision s'effectue comme les multiplications décimales classiques en calculant des produits partiels puis en additionnant ceux-ci pour obtenir le résultat. Au contraire de la multiplication, la présence de division 8 bits dans le microprocesseur ne simplifie pas le calcul de la division multiprécision. On est obligé, comme dans le cas de la divi-

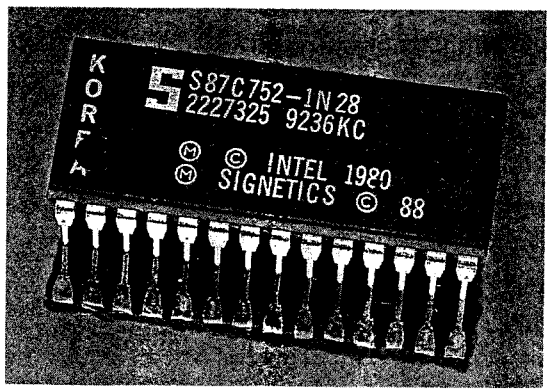
Division de 732 par 23

/ 23	732	
7	320	
/ 23	—↑	7/23 = 0
73	203	
23	—↑	73/23 = 3
4	203	
42	031	
/ 23	—↑	42/23 = 1
19	résultat 21 reste 19	

■ *Figure 1*

sion décimale classique de faire des soustractions successives pour arriver au résultat. Pour optimiser le calcul celui-ci est effectué d'une manière non conventionnelle, la **figure 1** explicite l'algorithme utilisé en l'appliquant à une division décimale «classique».
Dans le cas d'opérations signées, il faudra au préalable rendre les arguments positifs avant d'effectuer les opérations puis inverser le résultat si nécessaire. On pourra se passer de ces inversions si on effectue une multiplication 16x16 avec un résultat sur 16 bits. Dans ce cas les 16 bits de poids faible du résultat sont valides même si l'opération est signée. Nous verrons prochainement comment utiliser ses sous-programmes pour réaliser d'autres fonctions (racine carrée, sinus, arc tangente ...)
Les listings fournis donnent deux exemples, une multiplication 16x16 et une division par 32 en assembleur AS1 que l'on retrouvera sur le serveur (3615 ERP) avec le code objet correspondant.

J.L. VERN




```
; Division 32 bits de ACCU32 par R4:R5:R6:R7. Le résultat est dans ACCU32
; et le reste de la division dans R0:R1:R2:R3. R4:R5:R6:R7 sont inchangés,
; B est modifié et vaut 0 en sortie de la division. ACCU32 est constitué de
; 4 octets ACCU32+0:ACCU32+1:ACCU32+2:ACCU32+3 (ACCU32+0 est le MSB)
; La division par 0 donne pour résultat FFFFH.
; Taille de div32 : 67 octets
; temps d'execution : 7 + ( 35 + [11] ) X 32 + 4
; soit dans le pire cas (FFFFH / 1) : 46 X 32 + 11 = 1483 cycles
; Dans certains cas, on est sûr de n'avoir un résultat que sur 16 bits et on
; peut modifier le programme pour n'effectuer que 16 décalages et donc diviser
; le temps par deux (par exemple quand le résultat est issu d'une règle de 3 de
; type Y = X * n / p avec p > n). Dans ce cas, on initialisera la division avec
; le numérateur dans R2:R3:ACCU32+0:ACCU32+1, on supprimera l'initialisation
; de R2 et R3 à 0, et on chargera B avec 16 au lieu de 32.

ACCU32  EQU     8H              ; registre 32 bits externe

DIV32:  CLR     A               ; initialise le MSB du numérateur
        MOV     R0,A
        MOV     R1,A
        MOV     R2,A            ; à supprimer si on est sur d'avoir un
        MOV     R3,A            ; résultat sur 16 bits (voir en en-tête)
        MOV     B,#32           ; compteur de décalage
;       7 cycles
DIV320: CLR     C               ; multiplie le numérateur par 2 et laisse la
        MOV     A,ACCU32+3      ; place pour le résultat (voir plus bas)
        RLC     A
        MOV     ACCU32+3,A
        MOV     A,ACCU32+2
        RLC     A
        MOV     ACCU32+2,A
        MOV     A,ACCU32+1
        RLC     A
        MOV     ACCU32+1,A
        MOV     A,ACCU32+0
        RLC     A
        MOV     ACCU32+0,A
;
        MOV     A,R3            ; et pousse les bits dans R0:R1:R2:R3 <- C
        RLC     A
        MOV     R3,A            ; Dans une division sur papier, le numérateur
        MOV     A,R2            ; ne "bouge pas" et c'est le dénominateur qui
        RLC     A               ; est divisé par 2 (ou par 10 si on est en
        MOV     R2,A            ; en base 10). Ici, c'est le numérateur qui
        MOV     A,R1            ; "bouge" et est multiplié par 2 et le
        RLC     A               ; dénominateur reste fixe. On pourrait faire
        MOV     R1,A            ; les divisions sur papier exactement de la
        MOV     A,R0            ; même manière.
        RLC     A
        MOV     R0,A

        MOV     A,R3            ; compare le numérateur et le dénominateur
        SUBB    A,R7            ; la retenue est propagée
        MOV     A,R2            ; vers le MSB
        SUBB    A,R6            ; (une manière de comparer sur le 8051 est de
        MOV     A,R1            ; faire une soustraction et de tester C.
        SUBB    A,R5            ; Ici la comparaison est effectuée sur deux
        MOV     A,R0            ; mots de 32 bits : R0:R1:R2:R3 et R4:R5:R6:R7)
        SUBB    A,R4
        JC      DIV321          ; dénominateur > numérateur, résultat = 0
        35 cycles
; dénominateur <= numérateur, effectue la soustraction et résultat = 1
        MOV     R0,A            ; sauve le nouveau MSB du numérateur
        MOV     A,R3            ; recalcule la soustraction
        SUBB    A,R7            ; numérateur - dénominateur
        MOV     R3,A            ; en sauvant cette fois le résultat
        MOV     A,R2            ; (Les bits à droite de la partie qui sert à la
        SUBB    A,R6            ; comparaison du numérateur ne sont pas
        MOV     R2,A            ; affectés par la soustraction. Dans une
        MOV     A,R1            ; division effectuée sur papier, on complète de
        SUBB    A,R5            ; manière sous-entendue le dénominateur avec
        MOV     R1,A            ; des zéros, il en est de même ici)
; le numérateur était plus grand que le dénominateur, un 1 est donc poussé dans
; le résultat. Celui-ci prend place dans la place laissée libre par le
; numérateur lorsqu'il est décalé. A la fin de la division, le résultat occupe
; donc entièrement la place qui était prise initialement par le numérateur. Le
; décalage du résultat et du numérateur s'effectue donc en une seule même
; opération.
        INC     ACCU32+3        ; en fait met un 1 dans le LSB (voir au dessus)
;       11 cycles
DIV321: DJNZ    B,DIV320        ; passe ainsi les 32 bits
        RET
;       4 cycles
```

■ *Listing division par 32*