Biblioteki mikroprocesorowych procedur standardowych

Kolejny odcinek kursu poświęcamy prezentacji procedur dzielenia wielobajtowego. Listingi wszystkich programów prezentowanych w artykule dostępne są w postaci spakowanych plików poprzez sieć Internet, pod adresem strony Elektroniki Praktycznej www.atm.com.pl/~avt/ep.

Dzielenie wielobajtowe

Istnieje kilka metod dzielenia. Jedną z nich jest metoda z odtwarzaniem reszty. Sieć działań dla tej metody pokazano na **rys. 1**. Jej zasadą jest własność dzielenia binarnego, w której kolejna wyznaczana cyfra przyjmuje wartość 0, jeśli wynik odejmowania dzielnika od reszty cząstkowej jest ujemny (wtedy musimy odtworzyć poprzednią wartość reszty), albo 1 gdy różnica jest nieujemna.

Podobnie zachowujemy się dzieląc w dziesiętnym systemie liczbowym, ale mamy tu więcej możliwości wyboru. Ustalając kolejną cyfrę ilorazu dokonujemy mnożenia tej cyfry przez dzielnik i odejmujemy od reszty cząstkowej.

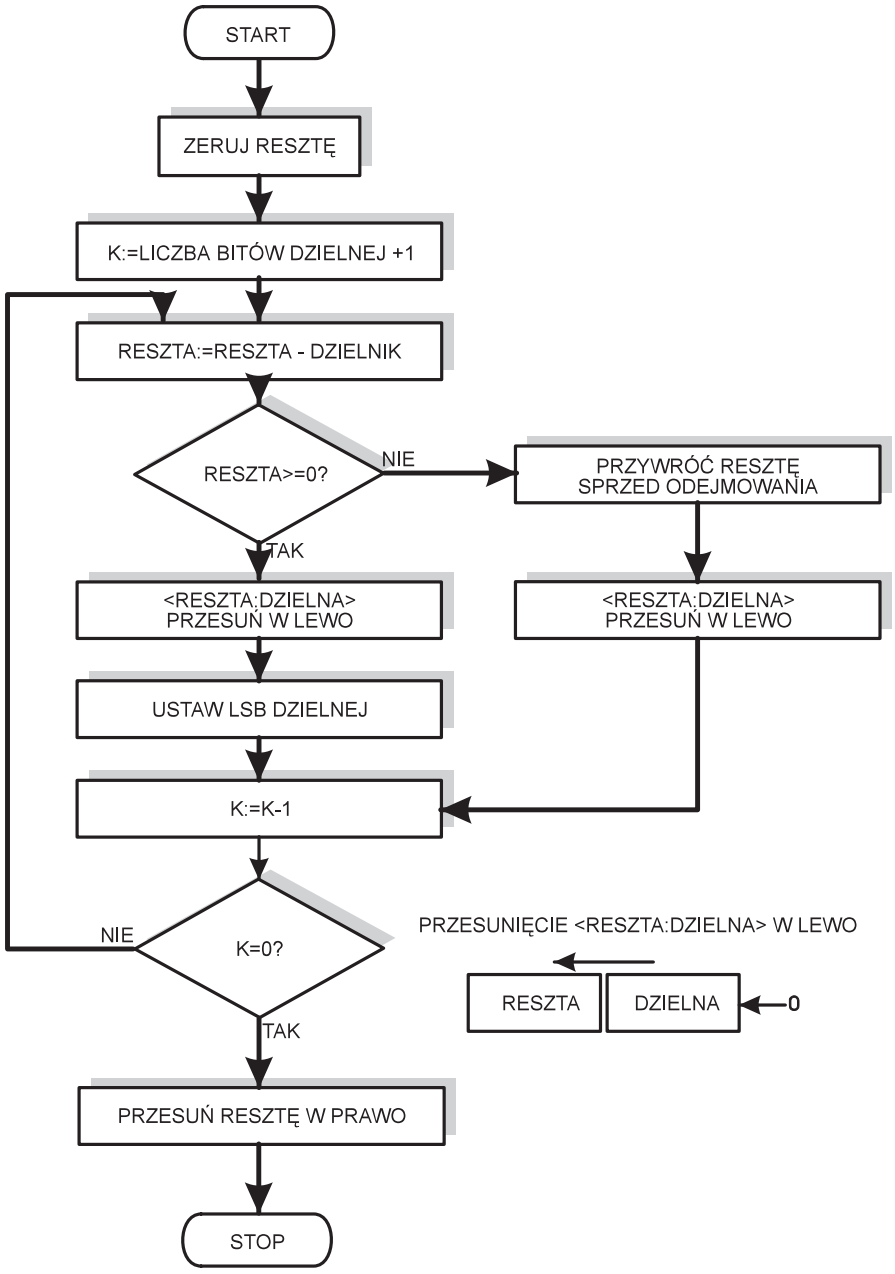
W dzieleniu binarnym mnożenie przez 1 jest przepisaniem dzielnika, zaś przez 0 daje oczywiście 0, a potem odjęcie oznacza odtworzenie reszty.

Listing 1.

```
; procedura dzielenia 4-BAJTOWYCH liczb
; przez 2-BAJTOWE
; wejście:
;r0 - adres najstarszego bajtu dzielnej,
;r1 - adres najstarszego bajtu dzielnika,
; wyjście:
;R0 - adres ilorazu
;R6:R5 - reszta
;W deklaracji zmiennych programu
; wykorzystujacego tę procedurę zdefiniować
;r0reg       equ      0
;r5reg       equ      5
;r6reg       equ      6
dziel32_16:
    clr     a ; wyzeruj acc
    mov     r5,a
    mov     r6,a
    mov     r7,#33 ; licznik iteracji
    mov     a,@r1
    inc     r1
    orl     a,@r1
    dec     r1
    jz      dziel32_7
    ljmp    dziel32_1 ; liczymy
dziel32_2:
    push    r6reg ; przechowanie reszty
    push    r5reg
    inc     r1
    mov     a,r5  ; reszta minus dzielnik
    clr     c
    subb    a,@r1
    mov     r5,a
    dec     r1
    mov     a,r6
    subb    a,@r1
    mov     r6,a
    jnc     dziel32_3 ; skok, gdy różnica
    pop     r5reg     ; dodatnia
    pop     r6reg  ; odtworzenie reszty
dziel32_1:
    mov     a,#3
    add     a,r0
    mov     r0,a
    mov     a,@r0    ;*
    add     a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;* przesunięcie dzielnej
    dec     r0       ;* w lewo
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    push    psw      ; zachowanie stanu CY
    mov     a,r5     ;*
    add     a,r5     ;*
    mov     r5,a     ;*-> przesunięcie reszty
    mov     a,r6     ;*   w lewo
    addc    a,r6     ;*
    mov     r6,a     ;*
    pop     psw      ; odzyskanie stanu CY
    jnc     dziel32_10
    inc     r5
dziel32_10:
    djnz    r7,dziel32_2
    clr     c
    mov     a,r6
    rrc     a
    mov     r6,a
    mov     a,r5
    rrc     a
    mov     r5,a
    ret
dziel32_3:
    pop     acc      ; zdjęcie nieaktualnej
    pop     acc      ; reszty ze stosu
    mov     a,#3
    add     a,r0
    mov     r0,a
    mov     a,@r0    ;*
    add     a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    push    psw      ; zachowanie stanu CY
    mov     a,r5     ;*
    add     a,r5     ;*
    mov     r5,a     ;*-> przesunięcie reszty
    mov     a,r6     ;*   w lewo
    addc    a,r6     ;*
    mov     r6,a     ;*
    pop     psw ; odzyskanie stanu CY
    jnc     dziel32_9
    inc     r5
dziel32_9:
    push    r0reg
    inc     r0
    inc     r0
    inc     r0
    inc     @r0
    pop     r0reg
dziel32_4:
    djnz    r7,dziel32_2
    clr     c
    mov     a,r6
    rrc     a
    mov     r6,a
    mov     a,r5
    rrc     a
    mov     r5,a
    ret
dziel32_7:
    setb    psw.2 ; ustawienie OV
    ret
```

Listing 2.

```
; procedura dzielenia 6-BAJTOWYCH liczb
; przez 3-BAJTOWE Z ZAOKRĄGLENIEM WYNIKU
; wejście:
;r0 - adres najstarszego bajtu dzielnej,
;r1 - adres najstarszego bajtu dzielnika,
; wyjście:
;R0 - adres ilorazu
;R6:R5:R4 - reszta, R6 najstarszy
;UWaga. Podprogram korzysta z podprogramu
;dzielenia liczb 6-bajtowych przez 3-bajtowe
;o nazwie DZIEL48_24.
;W deklaracji zmiennych programu
;wykorzystującego tę procedurę zdefiniować
;r2reg       equ      2
;r3reg       equ      3
;r7reg       equ      7
dziel48_24Z:
    LCALL   DZIEL48_24
    MOV     R1,R2REG
    CLR     C
    MOV     A,@R1
    RRC     A
    MOV     R7,A
    INC     R1
    MOV     A,@R1
    RRC     A
    MOV     R3,A
    INC     R1
    MOV     A,@R1
    RRC     A
    MOV     R2,A
    MOV     A,R6
    CJNE    A,R7REG,DZIEL4824Z1
    MOV     A,R5
    CJNE    A,R3REG,DZIEL4824Z1
    MOV     A,R4
    CJNE    A,R2REG,DZIEL4824Z1
DZIEL4824Z2:
    MOV     A,#5
    ADD     A,R0
    MOV     R0,A
    MOV     A,@R0
    ADD     A,#1
    MOV     @R0,A
    DEC     R0
    MOV     A,@R0
    ADDC    A,#0
    MOV     @R0,A
    DEC     R0
    MOV     A,@R0
    ADDC    A,#0
    MOV     @R0,A
    DEC     R0
    MOV     A,@R0
    ADDC    A,#0
    MOV     @R0,A
    RET
DZIEL4824Z1:
    JNC     DZIEL4824Z2
    RET
```



Rys. 1.

W zależności od liczby bajtów dostajemy procedurę mniej lub bardziej skomplikowaną, lecz ciągle realizującą ten sam algorytm. Wprawdzie ten algorytm dzielenia z odtwarzaniem reszty nie należy do najszybszych, ale autorowi to wystarczało. Oto te procedury:

- dzielenie liczb 6-bajtowych przez 3-bajtowe z zaokrągleniem wyniku (**listing 1**),
- dzielenie liczb 6-bajtowych przez 3-bajtowe (**listing 2**),
- dzielenie liczb 6-bajtowych przez 2-bajtowe,
- dzielenie liczb 4-bajtowych przez 2-bajtowe,
- dzielenie liczb 4-bajtowych przez 1-bajtowe (**listing 3**).

Dodatkowego komentarza wymaga dzielenie z zaokrągleniem wyniku (procedura DZIEL48_24Z). Każde dzielenie daje resztę, która zadecyduje o zaokrągleniu ilorazu. Jeśli podwojona wartość reszty będzie większa od dzielnika, to dodajemy do najmłodszej pozycji ilorazu jedynkę, w przeciwnym przypadku wynik bez korekty uznajemy za poprawny.

Mirosław Lach, AVT

Listing 3.

```
; procedura dzielenia 4-BAJTOWYCH liczb
; PRZEZ 1-BAJTOWE
; wejście:
;r0 - adres dzielnej,
;r1 - adres dzielnika,
; wyjście:
;R0 - adres ilorazu
;R5 - reszta
;W deklaracji zmiennych programu
;wykorzystującego tę procedurę zdefiniować
;r0reg      equ     0
;r5reg      equ     5
dziel328:
    clr     a ; wyzeruj acc
    mov     r5,a
    mov     r7,#33 ; licznik iteracji
    mov     a,@r1
    jz      dziel328_7
    ljmp    dziel328_1 ; liczymy
dziel328_2:
    push    r5reg ; przechowanie reszty
    mov     a,r5  ; reszta minus dzielnik
    clr     c
    subb    a,@r1
    mov     r5,a
    jnc     dziel328_3 ; skok, gdy różnica
                       ; dodatnia
    pop     r5reg ; odtworzenie reszty
dziel328_1:
    inc     r0
    inc     r0
    inc     r0
    mov     a,@r0    ;*
    add     a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;* -> przesunięcie
    dec     r0       ;*    dzielnej w lewo
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    mov     a,r5     ;*
    rlc     a        ;*-> przesunięcie reszty
    mov     r5,a     ;*   w lewo
    djnz    r7,dziel328_2
    clr     c
    mov     a,r5
    rrc     a
    mov     r5,a
    ret
dziel328_3:
    pop     acc      ;zdjecie nieaktualnej
                     ;reszty ze stosu
    inc     r0
    inc     r0
    inc     r0
    mov     a,@r0    ;*
    add     a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*-> przesunięcie dzielnej
    dec     r0       ;*   w lewo
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    dec     r0       ;*
    mov     a,@r0    ;*
    addc    a,@r0    ;*
    mov     @r0,a    ;*
    mov     a,r5     ;*
    rlc     a        ;*-> przesunięcie reszty
    mov     r5,a     ;*   w lewo
    push    r0reg
    inc     r0
    inc     r0
    inc     r0
    inc     @r0
    pop     r0reg
    djnz    r7,dziel328_2
    clr     c
    mov     a,r5
    rrc     a
    mov     r5,a
    ret
dziel328_7:
    setb    psw.2 ; ustawienie OV
    ret     ; dzielenie przez zero
```