Biblioteki mikroprocesorowych procedur standardowych

Wielu początkujących programistów mikrokomputerów jednoukładowych może odczuwać brak zaplecza w postaci biblioteki gotowych procedur. Bez niej droga do sukcesu, czyli dobrego gotowego programu jest dłuższa i bardziej uciążliwa. Biblioteka taka, zawierająca szczegółowo udokumentowane i sprawdzone moduły programowe jest bardzo cenna i oddaje nieocenione usługi. Gromadzi się więc te zasoby. Wystarczy do nich sięgnąć i dołączyć odpowiedni fragment do tworzonego oprogramowania. Namiastkę takiej biblioteki właśnie proponujemy.

Od Czytelnika wymagamy tylko znajomości listy podstawowych rozkazów tych mikrokontrolerów, dla których te procedury są przeznaczone. Zaczniemy od biblioteki dla najbardziej popularnej rodziny mikroprocesrów, jaką jest niewątpliwie rodzina '51. Prezentowane w artykule procedury są dostępne w sieci Internet pod adresem www.atm.com.pl/~avt.

Oferowany poprzez Internet kod źródłowy został sprawdzony w praktyce w czasie konstrukcji oprogramowania systemowego kilku sterowników. Procedury mają różnorodny charakter i przeznaczenie. Każda procedura jest przedstawiona w formie podprogramu. Proponowane podprogramy są używane w części oprogramowania zajmującego się przetwarzaniem odebranych danych, czyli tam, gdzie istnieje minimalne „przywiązanie" programu do otoczenia mikroprocesora. Z czasem do tej biblioteki dołączymy opracowania częściowo zależne od otoczenia procesora, oraz zajmiemy się obsługą przerwań.

Procedury publikowane w tej rubryce są dostępne na koncie WWW firmy AVT w Internecie. Dla osób nie posiadających dostępu do Internetu publikujemy pełny tekst źródłowy procedur. Nie przewidujemy rozprowadzania tych bibliotek na dyskietkach. Uznaliśmy bowiem, że coraz większa popularność Internetu może skłoni potencjalnych odbiorców tego oprogramowania do zainteresowania się tą ogólnoświatową siecią.

Arytmetyka

Zaczniemy od procedur arytmetycznych. Operacje są wykonywane na liczbach interpretowanych jako liczby całkowite, których zapis pokazano na **rys.1**

Rys. 1. Przyjęty przez autora sposób zapisu liczb wielobajtowych.

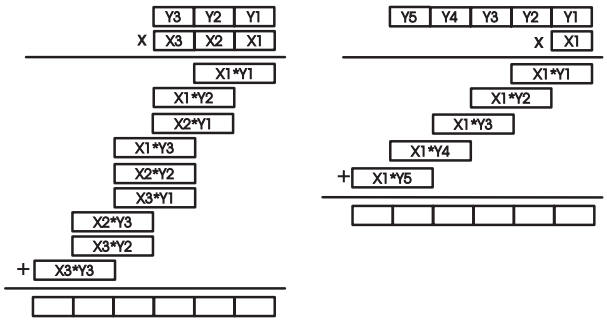
(starszy bajt jest pamiętany w komórce o młodszym adresie). Długość liczb, obliczana w bajtach, jest zmienna, zależnie od potrzeb.

W czasie układania własnego programu warto przeznaczyć pewną liczbę komórek pamięci wewnętrznej na bufor wyników pośrednich. Autor przyjął, że taki bufor jest umiejscowiony w trzech starszych bankach pamięci o adresie początkowym 08H i długości 24 bajtów. W procesorach '51, po ich zerowaniu, w tym obszarze jest lokalizowany stos. Dno stosu zostało więc przeniesione do adresu 70H. Szesnaście komórek przeznaczonych na stos wystarcza. Jeśli jednak zaczyna nam brakować pamięci wewnętrznej, np. ze względu na liczne, zagnieżdżające się odwołania do stosu, to musimy sięgnąć po nieco mocniejszy typ z rodziny '51, np. 80C32. Ma on 256 bajtów, z czego starsze 128 bajtów jest adresowanych tylko pośrednio, ponieważ dzielą one adresy z rejestrami funkcji specjalnych, dostępnych z kolei tylko bezpośrednio.

Listing 1.

```
;PODPROGRAM WIELOBAJTOWEGO dodawania
; WEJŚCIE:
;   R0 - ADRES NAJMŁODSZEGO
;        BAJTU PIERWSZEGO SKŁADNIKA
;   R1 - ADRES NAJMŁODSZEGO BAJTU DRUGIEGO
;        SKŁADNIKA, NIE MODYFIKOWANY
;   R2 - LICZBA BAJTÓW
; WYJŚCIE:
;   R0 - ADRES NAJMŁODSZEGO BAJTU WYNIKU
;   R2=0
;   PRZENIESIENIE C I PRZEPEŁNIENIE OV JEST
;   USTAWIANE W TAKI SPOSÓB, JAK W ROZKAZIE
;   ADDC
;   STAN WSKAŹNIKA P NIE ODPOWIADA OGÓLNYM
;   ZASADOM JEGO USTAWIENIA


AD:
            CLR C
            PUSH R0REG
AD1:
            MOV A,@R0
            ADDC A,@R1
            MOV @R0,A
            DEC R0
            DEC R1
            DJNZ R2,AD1
            POP R0REG
            RET
```



Rys. 2. Sposób mnożenia liczb wielobajtowych.

Dodawanie wielobajtowe

Dodawanie jednobajtowego jest trywialne, bowiem zapewniają to rozkazy dodawania ADD i dodawania z przeniesieniem ADDC z różnorodnymi argumentami. My zajmiemy się dodawaniem wielobajtowym. Uniwersalna procedura wielobajtowego dodawania przedstawiona jest na **listingu 1**.

Częstym błędem początkujących programistów jest brak zerowania wskaźnika przeniesienia C. Warto przyjąć od teraz założenie, że wywoływana procedura pracuje w środowisku programowym o nieznanych parametrach wejściowych (z wyjątkiem parametrów początkowych, które ma ona przetwarzać) i powinna sama je ustalić. Unikniemy wtedy niespodzianek, bowiem zasadnicza część procedury pracuje poprawnie, a błąd tkwi gdzie indziej.

Wynik dodawania jest umieszczany w miejsce pierwszego składnika sumy pośrednio adresowanego rejestrem R0.

Odejmowanie

Przez analogię do dodawania, tworzymy podprogram wielobajtowego odejmowania o adresie SUB (**listing 2**).

Stan wskaźników programowych po wykonaniu podprogramu odejmowania, czy jak poprzednio dodawania, nie jest bez znaczenia, bowiem może być istotny w dalszej części budowanego oprogramowania. Najważniejszy będzie dla nas stan wskaźnika przeniesienia C. Tam zostanie zapisana pożyczka (przeniesienie). Oprócz tego poprawnie zostanie ustawiony znacznik przepełnienia OV, używany w operacjach na liczbach zapisanych w kodzie uzupełnieniowym do dwóch.

Przy ustawieniu CY należy interpretować otrzymaną różnicę, jak liczbę ujemną. Wynik odejmowania jest zapisany w kodzie uzupełnieniowym do dwóch. Jest to odmiana kodu binarnego, służącego do zapisu liczb ze znakiem. Liczby dodatnie są w tym kodzie zapisane identycznie jak w naturalnym kodzie binarnym, zaś zapis liczb ujemnych powstają przez zanegowanie bitów modułu (wartości bezwzględnej) tej liczby i dodanie jedynki. Np. +5 to 0101, zaś -5 to zanegowane 0101, czyli 1010, plus 1, czyli w końcu 1011.

Ciekawą właściwością tego kodu (w skrócie nazywanego U2) jest to, że ponowna negacja bitów liczby ujemnej i dodanie 1 spowoduje otrzymanie liczby dodatniej równej co do wartości bezwzględnej liczbie ujemnej. W naszym przykładzie 1011 po inwersji to 0100, a plus 1 daje 0101, czyli 5. Prosta procedura dokonuje uzupełnienia do dwóch liczby wielobajtowej (**listing 3**).

Listing 2.

```
;PODPROGRAM WIELOBAJTOWEGO ODEJMOWANIA
; WEJŚCIE:
;   R0 - ADRES NAJMŁODSZEGO BAJTU ODJEMNEJ
;   R1 - ADRES NAJMŁODSZEGO BAJTU ODJEMNIKA
;   R2 - LICZBA BAJTÓW
; WYJŚCIE:
;   R0 - ADRES NAJMŁODSZEGO BAJTU WYNIKU
;   R2=0
;   POŻYCZKA C I PRZEPEŁNIENIE OV SĄ
;   USTAWIANE W TAKI SPOSÓB, JAK W ROZKAZIE
;   SUBB
;   STAN WSKAŹNIKA P NIE ODPOWIADA OGÓLNYM
;   ZASADOM JEGO USTAWIENIA


SUB:
            CLR C
            PUSH R0REG
SUB1:
            MOV A,@R0
            SUBB A,@R1
            MOV @R0,A
            DEC R0
            DEC R1
            DJNZ R2,SUB1
            POP R0REG
            RET
```

Listing 3.

```
;PODPROGRAM UZUPEŁNIENIA DO 2
;PISAŁ I TESTOWAŁ: MIROSLAW LACH,AVT
; WEJŚCIE:
;   R0 - ADRES NAJMŁODSZEGO BAJTU LICZBY
;   R2 - LICZBA BAJTÓW
;   LOKALNA ZMIENNA R2REG JEST ADRESEM REJESTRU
;   R2 Z BANKU 0 PAMIĘCI
;   LOKALNA ZMIENNA R0REG JEST ADRESEM REJESTRU
;   R0 Z BANKU 0 PAMIĘCI
;WYJŚCIE:
;   R0 - ADRES NAJMŁODSZEGO BAJTU LICZBY
;   UZUPEŁNIONEJ DO DWÓCH
R2REG       EQU     2
R0REG       EQU     0
UZUP:
    PUSH    R2REG   ; CHOWAMY REJESTRY R0 I R1
    PUSH    R0REG   ; JESZCZE SIĘ PRZYDADZĄ
UZUP1:
    MOV     A,@R0
    CPL     A       ; INWERSJA BITÓW
POSZCZEGOLNYCH BAJTOW
    MOV     @R0,A
    DEC     R0      ; MODYFIKACJA WSKAŹNIKA
                    ; ADRESOWEGO
    DJNZ    R2,UZUP1 ; CZY SĄ JESZCZE JAKIEŚ
                    ; BAJTY?
    POP     R0REG   ; WYSTARCZY, TERAZ
                    ; ODTWARZAMY R0 I R2
    POP     R2REG   ; PO TO, ABY DODAĆ JEDYNKĘ
    MOV     A,@R0   ; DO PIERWSZEGO BAJTU
    ADD     A,#1    ; DODAJEMY JEDYNKĘ
    MOV     @R0,A   ; TRZEBA TO JESZCZE
                    ; SCHOWAĆ
    DEC     R0      ; TERAZ ZMODYFIKOWAĆ
                    ; WSKAŹNIK ADRESOWY
    DJNZ    R2,UZUP2 ; CZY TO JEST JEDYNY BAJT?
    RET             ; TAK, TO BYŁ JEDYNY BAJT,
                    ; CZYLI KONIEC PROCEDURY
UZUP2:
    MOV     A,@R0   ; DO POZOSTAŁYCH BAJTÓW
    ADDC    A,#0    ; TRZEBA DODAĆ ZERO PO TO
    MOV     @R0,A   ; ABY DODAĆ EWENTUALNE
                    ; PRZENIESIENIE
    DEC     R0      ; MODYFIKACJA WSKAŹNIKA
                    ; ADRESOWEGO, JUŻ NIEDALEKO
    DJNZ    R2,UZUP2 ; CZY TO WSZYSTKIE BAJTY,
                    ; JEŚLI NIE - NA POCZĄTEK
                    ; PETLI
    RET             ; UFF, ZAKOŃCZYLIŚMY
```

Listing 4.

```
; PODPROGRAM MNOŻENIA LICZB DWUBAJTOWYCH
; PISAŁ I TESTOWAŁ: MIROSLAW LACH, AVT
; WEJŚCIE:
; R0 - ADRES NAJMŁODSZEGO BAJTU MNOŻNEJ
; R1 - ADRES NAJMŁODSZEGO BAJTU MNOŻNIKA
; WYJŚCIE:
; ILOCZYN JEST UMIESZCZANY W MIEJSCE CZYNNIKÓW
; ILOCZYNU, GDZIE ZŁOŻENIE BAJTÓW
; (R0-1):(R0):(R1-1):(R1) DAJE WYNIK (R0-1) -
; BAJT NAJSTARSZY, ZAŚ R0 I R1 SĄ WARTOŚCIAMI
; WEJŚCIOWYMI ADRESÓW
; ZALECA SIĘ, ABY MNOŻNA I MNOŻNIK ZAJMOWAŁY
; CZTERY KOLEJNE BAJTY
; UŻYWANE ZASOBY:
; ACC,B,R0,R1,R2,R3,R4,R5
; W DEKLARACJACH ZMIENNYCH ZDEFINIOWAĆ
; BEZPOŚREDNIE ADRESY REJESTRÓW JAKO
;   R2REG   EQU     2
;   R3REG   EQU     3
;   R4REG   EQU     4
;   R5REG   EQU     5

MNOZ2B:
    MOV A,@R0       ; W ACC JEST X
    MOV B,@R1       ; W B JEST Y
    MUL AB          ; X*Y
    MOV R2,A        ; REJESTRY R5-R2
            ; PRZECHOWUJĄ WYNIK POŚREDNI
    MOV R3,B        ; SUMY CZĄSTKOWEJ
    DEC R0
    MOV A,@R0       ; W ACC JEST X+1
    MOV B,@R1       ; W B JEST Y
    MUL AB          ; (X+1)*Y
    ADD A,R3
    MOV R3,A
    CLR A
    ADDC A,B
    MOV R4,A
    CLR A
    ADDC A,#0
    MOV R5,A
    INC R0
    DEC R1
    MOV A,@R0       ; W ACC JEST X
    MOV B,@R1       ; W B JEST Y+1
    MUL AB          ; X*(Y+1)
    ADD A,R3
    MOV R3,A
    MOV A,R4
    ADDC A,B
    MOV R4,A
    CLR A
    ADDC A,R5
    MOV R5,A
    DEC R0
    MOV A,@R0       ; W ACC JEST X+1
    MOV B,@R1       ; W B JEST Y+1
    MUL AB          ; (X+1)*(Y+1)
    ADD A,R4
    MOV R4,A
    MOV A,R5
    ADDC A,B
    MOV R5,A
    MOV @R0,R5REG
    INC R0
    MOV @R0,R4REG
    MOV @R1,R3REG
    INC R1
    MOV @R1,R2REG
    RET
```

Mnożenie

W mikroprocesorach rodziny '51 istnieje rozkaz mnożenia, MUL AB, ale dotyczy on mnożenia jednobajtowego. Mnożna i mnożnik są umieszczone w akumulatorze i rejestrze B. Iloczyn trafia z powrotem do tej pary rejestrów, przy czym młodsza część znajduje się w akumulatorze, a starsza jest umieszczona w B.

Rozkaz ten będzie bardzo użyteczny do wykonania mnożenia wielobajtowego. Na **rys. 2** pokazano znaną z lekcji arytmetyki metodę pisemnego mnożenia liczb wielocyfrowych. Ta właśnie zasada jest wykorzystana do budowy algorytmu mnożenia liczb wielobajtowych.

Mnożenie liczb wielobajtowych wykonujemy poprzez mnożenie pojedynczych bajtów za pomocą rozkazu MUL. Tak otrzymane iloczyny cząstkowe musimy odpowiednio dodawać do wyniku, czyli z właściwym przesunięciem, według przepisu pokazanego na rys. 2. Nie należy zapominać o dodawaniu iloczynu cząstkowego do następnego bajtu, po to aby nie zgubić ewentualnego przeniesienia.

Proponujemy następujące rodzaje podprogramów mnożenia różniące się długością mnożonych liczb:

- mnożenie liczb trzybajtowych;
- mnożenie liczb dwubajtowych (**listing 4**);
- mnożenie liczby pięciobajtowej i jednobajtowej;
- mnożenie liczby dwubajtowej i jednobajtowej (**listing 5**).

Opis przygotowania parametrów wejściowych oraz umiejscowienie wyniku zawarto w komentarzach w nagłówkach plików.

Mirosław Lach, AVT

Listing 5.

```
; PODPROGRAM MNOŻENIA LICZBY
; DWUBAJTOWEJ I JEDNOBAJTOWEJ
; WEJŚCIE:
; R1:R0 - MNOŻNA
; R2 - MNOŻNIK
; WYJŚCIE:
; R5:R4:R3 - ILOCZYN, R5 - NAJSTARSZY
MNOZ2Z1B:
     MOV A,R0
     MOV B,R2
     MUL AB
     MOV R3,A
     MOV R4,B
     MOV A,R1
     MOV B,R2
     MUL AB
     ADD A,R4
     MOV R4,A
     MOV A,B
     ADDC A,#0
     MOV R5,A
     RET
```

Procedury przedstawione w artykule dostępne są poprzez Internet w pliku spakowanym KURS1.ARJ