Biblioteki mikroprocesorowych procedur standardowych

Kolejny odcinek prezentacji procedur dla procesorów rodziny MCS-51 poświęcamy przybliżeniu rozwiązań ułatwiających konwersję liczb zapisanych w różnych standardach.

Prezentowane procedury dostępne są w sieci Internet pod adresem www.atm.com.pl/~avt/ep (link „Nasze konto FTP").

I tak dotarliśmy do wyższego poziomu przetwarzania danych, jakim są wszelkiego rodzaju konwersje kodów. Procedury konwersji korzystają z czterech podstawowych działań arytmetycznych, wcześniej opisanych. Będziemy tu skrupulatnie korzystać z procedur, które zostały już opublikowane.

Procedura konwersji 2-cyfrowej liczby BCD na 1-bajtową liczbę binarną przedstawiona została na **list.1**. Zamianę realizuje się poprzez wielokrotne dodawanie liczby 10, gdy pozycja dziesiątek liczby BCD jest niezerowa i na koniec dodanie liczby jednostek. Przeniesienie wcale tu nie wystąpi, bo kod binarny jest bardziej pojemny od kodu BCD. Dla nas, ludzi, kod binarny jest z kolei mniej wygodny w interpretacji.

W kolejnych procedurach konwersji wielobajtowych liczb BCD na binarne oparto się na dwóch sposobach. Jednym z nich jest, jak w przykładzie wyżej, wielokrotne dodawanie wagi danej pozycji, czyli 1, 10, 100, 1000 itd. (**list. 2**, **list. 3**). Drugim sposobem jest wykorzystanie znanego z matematyki wzoru Hornera, który mówi, że każdą liczbę dziesiętną (i nie tylko) można przedstawić w postaci wielokrotnego iloczynu jej podstawy:

$$a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_2 \cdot 10^2 + a_1 \cdot 10 + a_0 = ((..((a_n \cdot 10 + a_{n-1}) 10 + a_{n-2}) \cdot 10 + \ldots) \cdot 10 + a_1) \cdot 10 + a_0$$

Na przykład liczbę 12345 zapiszemy jako:

$$(((1 \cdot 10 + 2) \cdot 10 + 3) \cdot 10 + 4) \cdot 10 + 5.$$

Drugi sposób znacznie upraszcza długość programu, ale z kolei spo-

Listing 1.

```
;PODPROGRAM 1-BAJTOWEJ KONWERSJI BCD->BIN
;WEJŚCIE:
;    R0 - SKĄD
;    R1 - DOKĄD
BCD1BIN:
     MOV @R1,#0
     MOV A,@R0
     SWAP A
     ANL A,#0FH
     JZ BCD1BIN1
     MOV R2,A
     CLR C
BCD1BIN2:
     MOV A,@R1
     ADDC A,#10
     MOV @R1,A
     DJNZ R2,BCD1BIN2
BCD1BIN1:
     MOV A,@R0
     ANL A,#0FH
     ADD A,@R1
     MOV @R1,A
     RET
```

Listing 2.

```
;PODPROGRAM KONWERSJI 3-BAJTOWEJ LICZBY BCD
;NA LICZBĘ 3-BAJTOWĄ NB
; WEJŚCIE
; DPTR-ADRES NAJSTARSZEGO BAJTU
;      LICZBY BCD W|XRAM
; R0 - ADRES NAJMŁODSZEGO BAJTU
;      LICZBY NB W|IRAM
BCD3_NB3:
            MOV R6,R0REG
            MOV R2,#3
            CLR A
BCD3NB37:
            MOV @R0,A
            DEC R0
            DJNZ R2,BCD3NB37
            MOVX A,@DPTR
            SWAP A
            ANL A,#0FH
            JZ BCD3NB31
            MOV R5,A
            MOV R2,#01H
            MOV R3,#86H
            MOV R4,#0A0H
BCD3NB38:
            MOV R0,R6REG
            LCALL ADD_3B
            DJNZ R5,BCD3NB38
BCD3NB31:
            MOVX A,@DPTR
            ANL A,#0FH
            JZ BCD3NB32
            MOV R5,A
            MOV R2,#00H
            MOV R3,#27H
            MOV R4,#10H
BCD3NB39:
            MOV R0,R6REG
            LCALL ADD_3B
            DJNZ R5,BCD3NB39
BCD3NB32:
            INC DPTR
            MOVX A,@DPTR
            SWAP A
            ANL A,#0FH
            JZ BCD3NB33
            MOV R5,A
            MOV R2,#00H
            MOV R3,#03H
            MOV R4,#0E8H
BCD3NB40:
            MOV R0,R6REG
            LCALL ADD_3B
            DJNZ R5,BCD3NB40
BCD3NB33:
            MOVX A,@DPTR
            ANL A,#0FH
            JZ BCD3NB34
            MOV R5,A
            MOV R2,#00H
            MOV R3,#00H
            MOV R4,#64H
BCD3NB41:
            MOV R0,R6REG
            LCALL ADD_3B
            DJNZ R5,BCD3NB41
BCD3NB34:
            INC DPTR
            MOVX A,@DPTR
            SWAP A
            ANL A,#0FH
            JZ BCD3NB35
            MOV R5,A
            MOV R2,#00H
            MOV R3,#00H
            MOV R4,#0AH
BCD3NB42:
            MOV R0,R6REG
            LCALL ADD_3B
            DJNZ R5,BCD3NB42
BCD3NB35:
            MOVX A,@DPTR
            ANL A,#0FH
            JZ BCD3NB36
            MOV R2,#0H
            MOV R3,#0H
            MOV R4,A
            MOV R0,R6REG
            LCALL ADD_3B
BCD3NB36:
            RET
;PODPROGRAM 3-BAJTOWEGO DODAWANIA
; R2:R3:R4 - PIERWSZY SKŁADNIK
; R0 - ADRES NAJMŁODSZEGO BAJTU
;      DRUGIEGO SKŁADNIKA I|SUMY W|IRAM
ADD_3B:
            CLR C
            MOV A,@R0
            ADD A,R4
            MOV @R0,A
            DEC R0
            MOV A,@R0
            ADDC A,R3
            MOV @R0,A
            DEC R0
            MOV A,@R0
            ADDC A,R2
            MOV @R0,A
            RET
```

Listing 3

```
; PODPROGRAM KONWERSJI LICZBY 4-BAJTOWEJ BCD
; NA 4-BAJTOWĄ NB
; WEJŚCIE: R0-ADRES NAJSTARSZEGO BAJTU
;            LICZBY BCD
;         R1-ADRES NAJMŁODSZEGO BAJTU
;            LICZBY NB
; UŻYWA OBSZARU BUFOR+8:BUFOR+11 JAKO
; ROBOCZEGO,ALE GO CHRONI
KONW8BCD4B:
            push bufor+8
            push bufor+9
            push bufor+10
            push bufor+11
            PUSH R1REG
            MOV R2,#4
            CLR A
K8B9:
            MOV @R1,A
            DEC R1
            DJNZ R2,K8B9
            POP R1REG
            MOV A,@R0
            SWAP A
            ANL A,#0FH
            JZ K8B2
            MOV R3,A
            MOV BUFOR+8,#00H;10^7=10 000 000
            MOV BUFOR+9,#98H
            MOV BUFOR+10,#96H
            MOV BUFOR+11,#80H
            LCALL ADD4BUF
K8B2:
            MOV A,@R0
            ANL A,#0FH
            JZ K8B3
            MOV R3,A
            MOV BUFOR+8,#00H
            MOV BUFOR+9,#0FH;10^6
            MOV BUFOR+10,#42H
            MOV BUFOR+11,#40H
            LCALL ADD4BUF
K8B3:
            INC R0
            MOV A,@R0
            SWAP A
            ANL A,#0FH
            JZ K8B4
            MOV R3,A
            MOV BUFOR+8,#00H
            MOV BUFOR+9,#01H;10^5
            MOV BUFOR+10,#86H
            MOV BUFOR+11,#0A0H
            LCALL ADD4BUF
K8B4:
            MOV A,@R0
            ANL A,#0FH
            JZ K8B5
            MOV R3,A
            MOV BUFOR+8,#00H
            MOV BUFOR+9,#00H;10^4
            MOV BUFOR+10,#27H
            MOV BUFOR+11,#10H
            LCALL ADD4BUF
K8B5:
            INC R0
            MOV A,@R0
            SWAP A
            ANL A,#0FH
            JZ K8B6
            MOV R3,A
            MOV BUFOR+8,#00H
            MOV BUFOR+9,#00H
            MOV BUFOR+10,#03H;10^3
            MOV BUFOR+11,#0E8H
            LCALL ADD4BUF
K8B6:
            MOV A,@R0
            ANL A,#0FH
            JZ K8B7
            MOV R3,A
            MOV BUFOR+8,#00H
            MOV BUFOR+9,#00H
            MOV BUFOR+10,#00H;10^2
            MOV BUFOR+11,#64H
            LCALL ADD4BUF
K8B7:
            INC R0
            MOV A,@R0
            SWAP A
            ANL A,#0FH
            JZ K8B8
            MOV R3,A
            MOV BUFOR+8,#00H
            MOV BUFOR+9,#00H
            MOV BUFOR+10,#00H
            MOV BUFOR+11,#0AH;10^1
            LCALL ADD4BUF
K8B8:
            MOV A,@R0
            ANL A,#0FH
            MOV BUFOR+8,#00H
            MOV BUFOR+9,#00H
            MOV BUFOR+10,#00H
            MOV BUFOR+11,A
            MOV R3,#1
            LCALL ADD4BUF
            pop bufor+11
            pop bufor+10
            pop bufor+9
            pop bufor+8
            ret
; PODPROGRAM DODAWANIA 4|BAJTÓW
; W|PODPROGRAMIE KONWERSJI
; LICZBY 4-BAJTOWEJ BCD NA 4-BAJTOWĄ BCD
ADD4BUF:
            PUSH R0REG
            PUSH R1REG
            mov r0,r1reg
ADD4B1:
            PUSH R0REG
            MOV R1,#BUFOR+11
            MOV R2,#4
            LCALL AD
            POP R0REG
            DJNZ R3,ADD4B1
            POP R1REG
            POP R0REG
            RET
```

walnia jego wykonanie, trzeba bowiem tracić czas na aktualizację parametrów sterowania pętlami, wywoływanie podprogramów itp.

Na list. 1 liczba jest pobierana z komórki pamięci wewnętrznej o adresie zapisanym w rejestrze R0, a wynik konwersji jest zapisywany do komórki pamięci wewnętrznej o adresie zapisanym w rejestrze R1. Oczywiście adresy te muszą być różne ze względu na dwukrotne odwoływanie się do tej samej komórki pamięci.

Za pomocą procedury z list. 2 dokonywana jest konwersja dziesiętnej liczby 6-cyfrowej zapisanej jako liczba BCD, czyli mieszczącej się w trzech bajtach. Dla odmiany procedura ta pobiera przetwarzaną liczbę z pamięci zewnętrznej, stąd obecność wskaźnika DPTR. Konwersja zaczyna się od najstarszej cyfry, czyli od pozycji o wadze 10^5 i kończy się na pozycji o wadze 10^0. Wynik konwersji jest zapisany w pamięci wewnętrznej procesora w trzech kolejnych komórkach o adresach od [R0-2] do [R0], przy czym R0 oznacza zawartość rejestru R0, zapis [x] oznacza adres komórki zapisany w rejestrze x.

Za pomocą procedury z listingu 3 dokonywana jest konwersja dziesiętnej liczby 8-cyfrowej zapisanej jako liczba BCD, czyli zajmującej cztery bajty. Dane wejściowe jak i wynik działania procedury są umieszczane w pamięci wewnętrznej procesora.

Za pomocą kolejnej procedury można dokonać konwersji dziesiętnej liczby 5-cyfrowej zapisanej w kodzie ASCII, mieszcząca się w pięciu bajtach. Różnica pomiędzy zapisem BCD a zapisem w kodzie ASCII jest tylko taka, że zapis BCD wykorzystuje cztery bity na zapis jednej cyfry, zaś w kodzie ASCII cyfry są kodowane jako liczby od 30H do 39H. W przypadku kodu ASCII wystarczy więc wymaskować cztery starsze bity po to, aby dostać kod BCD w młodszej tetradzie.

Konwersja odwrotna, czyli zamiana liczby binarnej na ciąg liczb BCD polega z kolei na wielokrotnym dzieleniu liczby binarnej przez 10 i wychwytywaniu reszty z dzielenia jako właśnie cyfr BCD. Pierwszą cyfrą jest najmłodsza cyfra liczby BCD, zaś ostatnią - najstarsza. Chciałoby się, żeby kolejność cyfr była odwrotna, co byłoby zgodne z naszym przyzwyczajeniem do zapisywania liczb, jednak jest to niemożliwe: w celu uzyskania najstarszej cyfry trzeba wykonać najwięcej dzieleń przez 10, czyli jakby wykonać jedno dzielenie przez 10^n. Ta metoda została zastosowana w kolejnej procedurze, której listingu nie zmieścimy ze względu na jej dużą objętość.

Ponieważ dzielenie to nic innego jak wielokrotne odejmowanie, innym sposobem jest oczywiście zliczanie odejmowań określonej wagi, aż do momentu wykrycia przepełnienia. Rozpoczynamy od pozycji z wagą największą i od liczby odejmujemy tę wagę tak długo, aż wynik będzie mniejszy od wagi. Ilość wykonanych odejmowań z wynikiem dodatnim da nam cyfrę najstarszą. Od pozostałej reszty odejmujemy wagę kolejnej, młodszej pozycji itd. Postępujemy tak długo aż dojdziemy do pozycji jednostek. Procedura realizująca podany algorytm dostępna jest w sieci Internet.

Kolejną procedurą, która może być przydatna jest procedura konwersji liczby zapisanej w kodzie ASCII na liczbę BCD. Taka procedura została przedstawiona na **list. 4**. Nie wymaga ona szerszego komentarza, wystarczy tylko zauważyć wyżej wspomniane różnice.

Mirosław Lach, AVT

Uwaga! *Do tej części "kursu" w Internecie dostępne jest siedem plików źródłowych.*

Listing 4

```
;PODPROGRAM SKŁADANIA ZNAKOW ASCII W|LICZBĘ BCD
;WEJŚCIE:
;   R0 - ADRES PRAWEGO ZNAKU
;   R1 - ADRES NAJMŁODSZEGO BAJTU
;   R5 - LICZBA ZNAKÓW
ASCII_BCD:
            MOV A,@R0
            ANL A,#0FH
            MOV @R1,A
            DEC R0
            DJNZ R5,ASCBCD1
            RET
ASCBCD1:
            MOV A,@R0
            ANL A,#0FH
            SWAP A
            ORL A,@R1
            MOV @R1,A
            DEC R0
            DEC R1
            DJNZ R5,ASCII_BCD
            RET
```