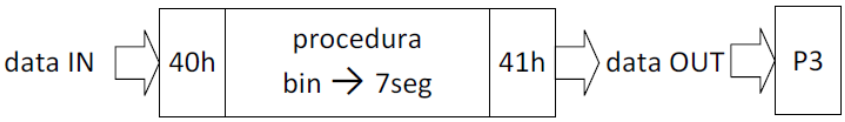
Zadanie **BIN ⇨ 7seg**

należy stworzyć procedurę (rys.1) wykonującą poniższe czynności:
- pobieranie z wejściowego rejestru procesora danej (kod binarny naturalny)
- przekształcenie pobranej danej na kod wyświetlacza siedmio-segmentowego
- umieszczenie kodu w rejestrze wyjściowym

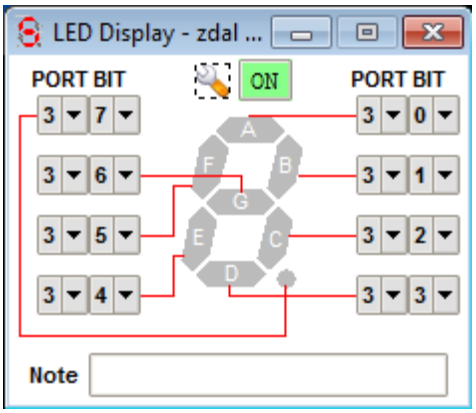


Rys.1. Schemat blokowy przepływu danych procedury

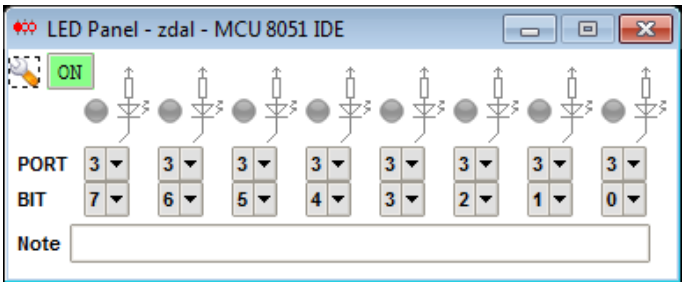
Procedura musi tworzyć kod zapewniający sterowanie wyświetlaczem zgodnie z załączoną tabelą. Podstawową zasadą działania takiej procedury jest przypisanie do każdej wartości wejściowej, kodu wyjściowego włączającego odpowiednie segmenty wyświetlacza np.:
data IN = 01h (liczba 1 ⇨ znak na wyświetlaczu 1) ⇨ data OUT = F9h (11111001b)
Wybrany segment wyświetlacza (rys.2) jest włączony (świeci) gdy sygnał nim sterujący ma stan niski („0" logiczne) np.: P3.1=0 -> świeci segment B.
Metod przypisania kodu wyjściowego do danej wejściowej jest wiele. Najczęściej stosowne to:
- porównywanie danej wejściowej z wartością oczekiwaną (np. mnemonik ***cjne***)
- stworzenie tablicy z kodami wyjściowymi i odczytywanie ich z tablicy na podstawie danych wejściowych – metoda preferowana

W przypadku drugiej metody tablica musi (w naszym systemie) być częścią programu. Jej utworzenie powinno wykorzystywać dyrektywę ***DB***. Odczyt tablicy jest możliwy poprzez rejestr indeksowy ***DPTR*** (adresowanie pośrednie).

IN	znak na wyświetlaczu 7-seg	IN	znak na wyświetlaczu 7-seg
00h	**0**	10h	**F**
01h	**1**	11h	**G**
02h	**2**	12h	**h**
03h	**3**	13h	**H**
04h	**4**	14h	**L**
05h	**5**	15h	**o**
06h	**6**	16h	**P**
07h	**7**	17h	**u**
08h	**8**	18h	**U**
09h	**9**	…	-
0Ah	**A**	…	-
0Bh	**b**	…	-
0Ch	**c**	…	-
0Dh	**C**	…	-
0Eh	**d**	…	-
0Fh	**E**	FFh	-



Rys.2. Schemat połączeń wyświetlacza 7-seg.



Rys.3. Schemat połączeń diod LED

Dyrektywa ***DB*** <***coś***> powoduje umieszczenie, w trakcie ładowania programu (kodów), wartości ***coś*** we wskazanym miejscu pamięci programu np.:

```
        …
ORG     3000h           ; adres wybrany arbitralnie
        DB C0h, F9h, A4h    ; dane wybrane arbitralnie
        …
```

Ten fragment programu powoduje umieszczenie, w trakcie ładowania programu, w przestrzeni adresowej programu od adresu 3000h trzy dane. Pierwsza **C0h** w adresie **3000h**, druga **F9h** w adresie **3001h**, itd.

Po uruchomieniu programu może on odczytać te dane (oczywiście nie może ich zmienić gdyż są umieszczone w przestrzeni adresowej programu – pamięć stała np. ROM)np.:

```
…
mov     DPTR,#3000h   ; do DPTR liczba 3000h (DPTR jako jedyny może być ładowany liczbą 16-b)
mov     A,#00h        ; zerowanie akumulatora
movc    A,@A+DPTR     ; do akumulatora zawartość komórki o adresie A+DPTR (3000h)
mov     R0,a          ; do R0 zawartość akumulatora
inc     DPTR          ; inkrementacja zawartości DPTR (teraz będzie 3001h)
mov     A,#00h        ; zerowanie akumulatora
movc    A,@A+DPTR     ; do akumulatora zawartość komórki o adresie A+DPTR (3001h)
mov     R1,a          ; do R1 zawartość akumulatora
…
```

Po wykonaniu: w R0 będzie C0h, a w R1 będzie F9h. Oczywiście fragment programu realizujący to zadanie można napisać w różny sposób.