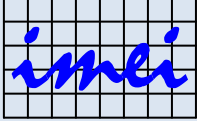


<b>PODSTAWY TECHNIKI MIKROPROCESOROWEJ</b> Laboratorium Elektrotechnika, studia stacjonarne pierwszego stopnia	 Instytut Metrologii, Elektroniki i Informatyki
<b>Temat: Konwersja danych i ich prezentacja na 7-segmentowych wyświetlaczach LED</b>	
Opracowanie instrukcji: dr inż. Mirosław Kozioł	<b>Ćwiczenie 3</b>

## Cel ćwiczenia

Celem ćwiczenia jest pokazanie wykorzystania wybranych operacji arytmetycznych do realizacji konwersji danych oraz statycznego sterowania 7-segmentowym wyświetlaczem LED.

## Zagadnienia do przygotowania

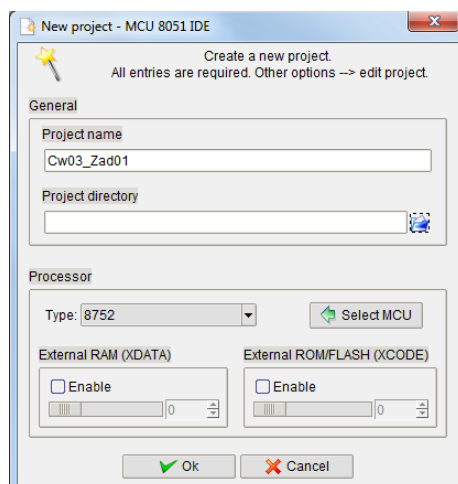
Przed przystąpieniem do zajęć należy przygotować lub powtórzyć informacje dotyczące:

- działań realizowanych przez rozkazy DIV, MOV i MOVC znajdujące się na liście rozkazów mikrokontrolerów rodziny MCS-51 oraz argumentów możliwych do zastosowania w tych rozkazach;
- sposobu umieszczania stałych definiowanych z wykorzystaniem dyrektywy DB w stosunku do kodu programu głównego;
- znaczenia dyrektyw DSEG i DS stosowanych w programach pisanych w asemblerze mikrokontrolerów rodziny MCS-51,
- działań arytmetycznych, jakie należy wykonać na liczbie 8-bitowej w celu wydzielenia cyfry setek, dziesiątek i jedności dla jej zapisu w systemie dziesiętnym.

## Program ćwiczenia

### Zadanie 1

- W środowisku MCU 8051 IDE stwórz nowy projekt wprowadzając w oknie *New Project* ustawienia, jak na rysunku 1.



Rys.1. Widok okna *New project* z zalecanymi ustawieniami

- b) Zakładając, że w komórce wewnętrznej pamięci danych pod adresem 30H zapisana jest 8-bitowa liczba napisz program, który określi wartość cyfry setek, dziesiątek i jedności dla zapisu dziesiętnego tej liczby i zapisze je do komórek pamięci określonych odpowiednio przez etykiety CYFRA\_S, CYFRA\_D, CYFRA\_J. Przy pisaniu programu uwzględnij przedstawioną poniżej strukturę kodu źródłowego.

```

DSEG AT 30H      ;dyrektywa - adres początkowy sekcji danych
                 ;(wewnętrzna pamięć RAM, adresowanie bezpośrednie)
DANA:    DS 1    ;dyrektywa DS - rezerwacja pamięci dla symbolu 'DANA'
                 ;(1 bajt, adres 30H)
CYFRA_S: DS 1    ;dyrektywa DS - rezerwacja pamięci dla symbolu BCD_S
                 ;(1 bajt, adres 31H)
CYFRA_D: DS 1    ;uzupełnić komentarz
CYFRA_J: DS 1    ;uzupełnić komentarz

CSEG AT 0000h   ;dyrektywa - adres początkowy sekcji kodu programu

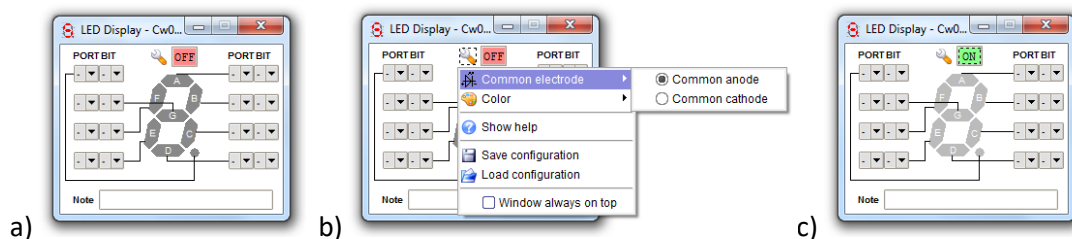
MOV DANA,#x     ;x to wartość podana przez prowadzącego zajęcia
                 ;kod własnej części programu
SJMP $         ;rozkaz skoku bezwarunkowego (pętla nieskończona)
END

```



- c) Zapisz plik z kodem źródłowym programu i dokonaj jego asemblacji. W przypadku wykrycia błędów składniowych dokonaj modyfikacji kodu w celu ich usunięcia.
- d) Przetestuj program uruchamiając go symulatorze wbudowanym w środowisko MCU 8051 IDE w trybie pracy krokowej. W przypadku uzyskania niepoprawnych wyników wyznaczania cyfry setek, dziesiątek i jedności zmodyfikuj kod, aby wykonanie programu dawało poprawne wyniki. Poprawnie działający program zaprezentuj prowadzącemu zajęcia.

## Zadanie 2

- a) Pracując nadal na tym samym projekcie, do portów P1, P2 i P3 dołącz 7-segmentowe wyświetlacze LED. W tym celu z menu *Virtual HW* wybierz *LED Display*. W wywołanym oknie, jak na rysunku 2 a), dokonaj przyporządkowania odpowiedniego portu i jego linii dla sterowania poszczególnymi segmentami danego wyświetlacza według zaleceń prowadzącego.



**Rys.2.** Widoki okna *LED Display* prezentujące wprowadzanie kolejnych ustawień

- b) Po kliknięciu na ikonę , z wywołanego menu podręcznego (patrz rysunek 2 b) określ, czy dany wyświetlacz jest układem o wspólnej anodzie, czy katodzie zgodnie z zaleceniami prowadzącego zajęcia.
- c) Upewnij się, że klawisz na prawo od ikony  jest w pozycji ON (patrz rysunek 2 c). Jeśli nie, przełącz go w tę pozycję.

- d) Działania opisane w punktach od a) do c) powtórz dla pozostałych dwóch portów.
- e) Dla przyjętego rodzaju wyświetlacza (wspólna anoda lub wspólna katoda) i dołączenia poszczególnych segmentów wyświetlaczy do linii portów mikrokontrolera dokonaj wyznaczenia wartości, które przesłane do portu dadzą możliwość prezentację na wyświetlaczu cyfr od 0 do 9.
- f) Na podstawie informacji podanych przez prowadzącego dokonaj sprawdzenia poprawności wyznaczenia tych wartości.
- g) Bazując na wiedzy zdobytej podczas realizacji zadania 3 w ćwiczeniu 2 oraz przedstawionej na wykładzie do tego przedmiotu, zmodyfikuj program napisany w ramach zadania 1 niniejszego ćwiczenia w taki sposób, aby cyfry setek, dziesiątek i jedności były prezentowane na wyświetlaczach LED podłączonych odpowiednio do portu P1, P2 i P3. Podczas realizacji tego zadania wykorzystaj kody cyfr wyznaczone w punkcie e) tego zadania.
- h) Zapisz plik z kodem źródłowym programu i dokonaj jego asemblacji. W przypadku wykrycia błędów składniowych dokonaj modyfikacji kodu w celu ich usunięcia.
- i) Przetestuj program uruchamiając go symulatorze wbudowanym w środowisko MCU 8051 IDE w trybie pracy krokowej. W przypadku niepoprawnej prezentacji cyfr na wyświetlaczu zmodyfikuj kod, aby wykonanie programu dawało poprawne wyniki. Poprawnie działający program zaprezentuj prowadzącemu zajęcia.

## Sprawozdanie z ćwiczenia

Sprawozdanie z ćwiczenia powinno być dostarczone prowadzącemu zajęcia w określonej przez niego formie (pisemnej lub elektronicznej) i zawierać:

- kod programu napisany podczas realizacji zadania 2 wraz z komentarzami (informacje dotyczące komentowania kodu programu znajdują się w dalszej części niniejszej instrukcji),
- algorytmy pracy całego programu zrealizowanego w ramach zadania 2 (informacje dotyczące tworzenia schematów blokowych algorytmów pracy programów znajdują się w dalszej części niniejszej instrukcji),
- opis słowny uzupełniający schemat blokowy algorytmu oraz kod programu w celu jasnego zrozumienia działania programu. Opis ten powinien wyjaśniać idee, jaką posłużono się przy realizacji kodu odpowiedzialnego za wydzielanie cyfry setek, dziesiątek i jedności. Należy również opisać procedurę, według której ustalono kody dla cyfr 7-segmentowego wyświetlacza LED.

## Komentowanie kodu programu

Komentarze umieszczane w kodzie programu powinny opisywać działanie danego rozkazu lub grupy rozkazów z punktu widzenia funkcji, jakie pełnią w tworzonym programie. Dla przykładu, jeśli w kodzie stworzonego programu wykorzystano instrukcję

```
MOVC A, @A+DPTR
```

do pobrania z pamięci kodu cyfry w celu jej prezentacji na 7-segmentowym wyświetlaczu LED, to rozkaz ten powinien być opatrzony następującym komentarzem

```
MOVC A, @A+DPTR    ;Pobranie kodu cyfry dla prezentacji  
                   ;na wyświetlaczu 7-segmentowym
```

Przykład niepoprawnego komentarza przedstawiono poniżej.

```
MOVC A, @A+DPTR ;Pobranie wartości z pamięci kodu
```

Jego niepoprawność polega na tym, że nie daje on żadnej informacji odnośnie funkcji, jaką dany rozkaz pełni w tworzonej programie, a tylko powiela informację prezentowaną przez jego mnemonik i operand.

## Tworzenie schematów blokowych algorytmów pracy programu

Graficzna prezentacja algorytmu działania programu za pomocą schematu blokowego powinna być tworzona w oparciu o zdefiniowane figury geometryczne (bloki), które reprezentują pewne czynności wykonywane w programie. Tworząc schemat blokowy algorytmu łączymy odpowiednie bloki liniami zakończonymi strzałkami. Jeśli linia jest linią łamaną, to załamania powinno być realizowane pod kątem prostym.

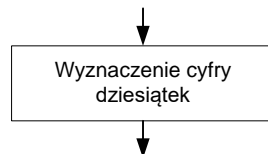
Początek i koniec algorytmu wyznaczają bloki graniczne rysowane w postaci owali, jak na rysunku 3.



**Rys.3.** Wygląd bloków reprezentujących początek i koniec algorytmu

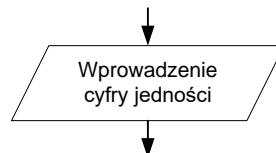
Początek algorytmu oznaczany jest przez blok z napisem START, koniec przez blok z napisem STOP. Blok symbolizujący początek algorytmu ma tylko jedną strzałkę wychodzącą i żadnej wchodzącej, natomiast blok symbolizujący koniec algorytmu ma co najmniej jedną strzałkę wchodzącą.

Wszelkiego rodzaju podstawienia i obliczenia oznaczane są przez blok procesu reprezentowany przez prostokąt, jak na rysunku 4. Blok ten ma dokładnie jedną strzałkę wchodzącą i jedną wychodzącą. Wewnątrz bloku wprowadzany jest opis omawiający wykonywane działania.



**Rys.4.** Wygląd bloku procesu z przykładowym opisem realizowanego działania

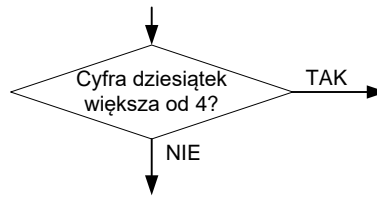
Wprowadzanie i wyprowadzanie danych oznaczane jest tym samym blokiem reprezentowanym przez równoległobok, jak na rysunku 5.



**Rys.5.** Wygląd bloku wprowadzania i wyprowadzania danych z przykładowym opisem realizowanego działania

W jego obrębie należy umieścić odpowiedni opis wskazujący, czy blok dotyczy pobrania czy wyprowadzania danych oraz jakie dane są pobierane lub wyprowadzane.

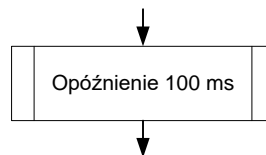
Wszelkiego rodzaju decyzje oznaczane są przez blok warunkowy, nazywany też decyzyjnym, reprezentowany przez romb. Jego symbol przedstawiono na rysunku 6.



**Rys.6.** Wygląd bloku warunkowego z przykładowym opisem realizowanego działania

Wewnątrz bloku umieszczany jest warunek w formie pytania sformułowanego w taki sposób, aby odpowiedź można było udzielić w postaci potwierdzenia (TAK) lub zaprzeczenia (NIE). Blok ten ma jedną strzałkę wchodzącą i dwie wychodzące. Jedna ze strzałek wychodzących związana jest ze spełnieniem warunku przedstawionego w bloku, druga z jego niespełnieniem. W pierwszym przypadku linię dla strzałki wychodzącej podpisujemy słowem TAK, w drugim przypadku słowem NIE. Każda linia ze strzałką powinna być wyprowadzana z innego wierzchołka rombu.

Zdefiniowany wcześniej proces i opisany innym algorytmem, a wykorzystywany w bieżącym algorytmie oznaczany jest blokiem jak na rysunku 7.



**Rys.7.** Wygląd bloku zdefiniowanego procesu z przykładowym opisem realizowanego działania

Blok ten można utożsamić z podprogramem w przypadku języka asemblera lub funkcją w przypadku języka C. Podobnie jak w przypadku bloku procesu reprezentowanego przez prostokąt, blok ten ma również jedną strzałkę wejściową i jedną wyjściową.

W niektórych przypadkach algorytm, który chcemy zaprezentować w postaci schematu blokowego, jest na tyle złożony, że połączenia wielokrotnie się krzyżują. Aby zwiększyć jego czytelność można w takich przypadkach stosować łączniki stronicowe, obowiązujące w ramach jednej strony, na której narysowany jest schemat. Ich symbol przedstawiono na rysunku 8.



**Rys.8.** Wygląd łączników stronicowych z przykładowym oznaczeniem

Realizacja połączenia polega na umieszczeniu w określonym miejscu symbolu łącznika ze strzałką wchodzącą (łącznik źródłowy) i określonym oznaczeniem w środku (np. liczbą) oraz w drugim miejscu algorytmu takiego samego symbolu z takim samym oznaczeniem, ale strzałką wychodzącą (łącznik docelowy).

**UWAGA:** Należy pamiętać, aby w schemacie blokowym algorytmu nie nadużywać łącznika stronicowego, gdyż w takim przypadku staje się on nieczytelny. Rysując schemat blokowy algorytmu należy rozplanować takie ułożenie poszczególnych bloków, aby połączenia między nimi jak najrzadziej się krzyżowały.

Jeśli schemat blokowy algorytmu nie mieści się na jednej stronie, to do połączenia jego fragmentów można użyć łącznika międzystronicowego, którego symbol przedstawiono na rysunku 9.



**Rys.9.** Wygląd łączników międzystronicowych z przykładowym oznaczeniem

Działa on w ten sam sposób, co łącznik stronicowy, lecz obowiązuje pomiędzy stronami. Najczęściej łączniki wyjściowe umieszczane są u dołu strony, a łączniki wejściowe u góry strony.

Należy pamiętać, że algorytm działania programu w postaci schematu blokowego powinien być zaprezentowany w sposób na tyle uniwersalny, aby na jego podstawie można było napisać kod programu w dowolnym języku programowania. Dlatego informacje umieszczane w poszczególnych blokach algorytmu nie mogą być fragmentami kodu, lecz powinny być w miarę możliwości formułowane słownie.

## Literatura

- [1] Wykłady do przedmiotu.
- [2] Tomasz Starecki: „*Mikrokontrolery 8051 w praktyce*”, Wydawnictwo BTC, Warszawa, 2002
- [3] Tomasz Starecki: „*Mikrokontrolery jednocukładowe rodziny 51*”, Wydawnictwo NOZOMI, Warszawa, 1996.
- [4] Andrzej Rydzewski: „*Mikrokontrolery jednocukładowe rodziny MCS-51*”, WNT, Warszawa, 1992.
- [5] Ryszard Krzyżanowski: „*Układy mikroprocesorowe*”, Wydawnictwo MIKOM, Warszawa, 2004.