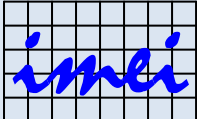


|  |  |
|--|--|
| <b>PODSTAWY TECHNIKI MIKROPROCESOROWEJ</b><br>Laboratorium<br>Elektrotechnika, studia stacjonarne pierwszego stopnia | <br>Instytut Metrologii,<br>Elektroniki i Informatyki |
| Temat: <b>Podprogramy, pętle i opóźnienia w asemblerze mikrokontrolerów rodziny MCS-51</b>                           |  |
| Opracowanie instrukcji: dr inż. Mirosław Kozioł  | Ćwiczenie 4  |

## Cel ćwiczenia

Celem ćwiczenia jest poznanie techniki programowania w języku asemblera mikrokontrolerów rodziny MCS-51 wykorzystującej podprogramy.

## Zagadnienia do przygotowania

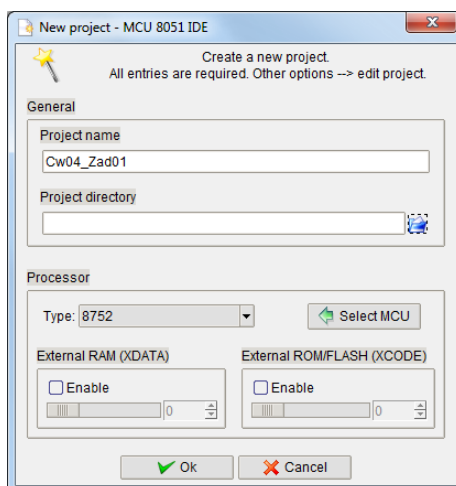
Przed przystąpieniem do zajęć należy przygotować lub powtórzyć informacje dotyczące:

- działań realizowanych przez rozkazów MOV, DJNZ, NOP, ACALL, LCALL, RET, SETB i CLR znajdujące się w liście rozkazów mikrokontrolerów rodziny MCS-51 oraz argumentów możliwych do zastosowania w tych rozkazach;
- sposobu umieszczania kodu podprogramów w stosunku do kodu programu głównego;
- obliczania czasu wykonania pojedynczego rozkazu przy znajomości częstotliwości sygnału zegarowego taktującego jednostkę centralną mikrokontrolera i liczby cykli maszynowych, w jakich wykonywany jest rozkaz.

## Program ćwiczenia

### Zadanie 1

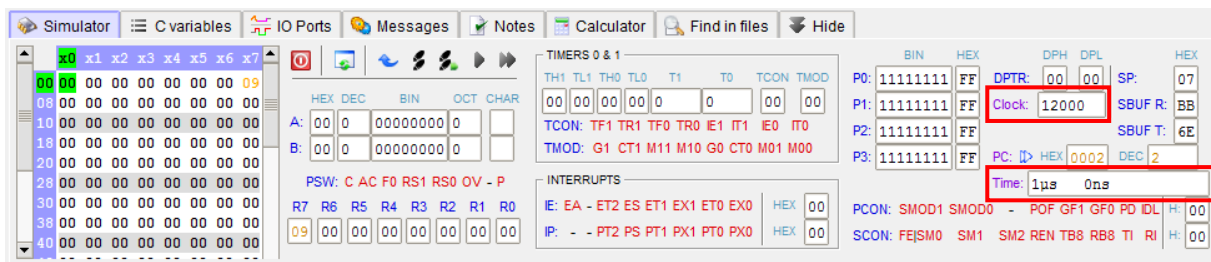
- W środowisku MCU 8051 IDE stwórz nowy projekt wprowadzając w oknie *New Project* ustawienia, jak na rysunku 1.



**Rys.1.** Widok okna *New project* z zalecanymi ustawieniami w części *Processor*

- Napisz program, który dla podanej przez prowadzącego częstotliwości sygnału zegarowego (12000 kHz) taktującego mikrokontroler realizuje opóźnienie o również podanym przez prowadzącego czasie trwania nie większym niż kilkadziesiąt mikrosekund.

- 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 działanie programu uruchamiając wbudowany w środowisko MCU8051IDE symulator. Przed rozpoczęciem symulacji w polu *Clock* na zakładce *Simulator* (patrz górne pole zaznaczone czerwoną ramką na rysunku 2) wprowadź częstotliwość sygnału zegarowego wyrażoną w kHz. W polu *Time* na tej samej zakładce (parz dolne pole zaznaczone czerwoną ramką na rysunku 2) podawany jest czas, jaki upłynął od chwili uruchomienia symulatora lub resetu mikrokontrolera. Posługując się tą informacją sprawdź, czy fragment kodu realizuje opóźnienie o założonym czasie (jak to zrobić wyjaśniono w dalszej części tej instrukcji). Jeśli nie, dokonaj modyfikacji programu, aby uzyskać opóźnienie o założonym czasie. ~~Poprawnie działający program zaprezentuj prowadzącemu zajęcia.~~



Rys.2. Widok zakładki *Simulator* z wyróżnionymi polami *Clock* i *Time*

## Zadanie 2

- a) W środowisku MCU 8051 IDE stwórz nowy projekt o nazwie *Cw04\_Zad02* z ustawieniami w części *Processor*, jak w zadaniu 1.
- b) Skopiuj kod programu napisany w ramach zadania 1 i zmodyfikuj go tak, aby instrukcje związane z realizacją opóźnienia były ujęte w postaci podprogramu według podanego poniżej szablonu.

```
CSEG AT 0000H
LCALL OP
SJMP $
```

```
OP:
;kod realizujący opóźnienie
RET
```

END

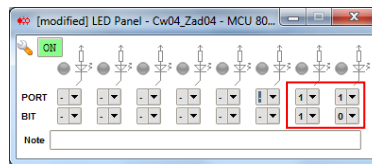
- c) Zapisz plik z kodem źródłowym programu i dokonaj jego asemblacji. W przypadku wykrycia błędów składniowych dokonaj modyfikacji kodu w celu ich usunięcia.
- d) Przetestuj działanie programu uruchamiając wbudowany w środowisko MCU 8051 IDE symulator. Zannotuj czas, jaki jest prezentowany w polu *Time* tuż przed wykonaniem rozkazu wywołania podprogramu realizującego opóźnienie i po wykonaniu rozkazu wyjścia z podprogramu. Na podstawie tych dwóch wartości określ, czy podprogram realizuje opóźnienie o założonym czasie trwania. Jeśli nie, dokonaj modyfikacji programu, aby uzyskać opóźnienie o założonym czasie. ~~Poprawnie działający program zaprezentuj prowadzącemu zajęcia.~~

### Zadanie 3

- W środowisku MCU 8051 IDE stwórz nowy projekt o nazwie *Cw04\_Zad03* z ustawieniami w części *Processor*, jak w zadaniu 1.
- Skopiuj kod programu napisany w ramach zadania 2 i zmodyfikuj go tak, aby realizował opóźnienie o czasie trwania podanym przez prowadzącego rzędu kilkudziesięciu milisekund.
- Zapisz plik z kodem źródłowym programu i dokonaj jego asemblacji. W przypadku wykrycia błędów składniowych dokonaj modyfikacji kodu w celu ich usunięcia.
- Przetestuj działanie programu uruchamiając wbudowany w środowisko MCU 8051 IDE symulator. Zanotuj czas, jaki jest prezentowany w polu *Time* tuż przed wykonaniem rozkazu wywołania podprogramu realizującego opóźnienie i po wykonaniu rozkazu wyjścia z podprogramu. Na podstawie tych dwóch wartości określ, czy podprogram realizuje opóźnienie o założonym czasie trwania. Jeśli nie, dokonaj modyfikacji programu, aby uzyskać opóźnienie o założonym czasie. Poprawnie działający program zaprezentuj prowadzącemu zajęcia.

### Zadanie 4

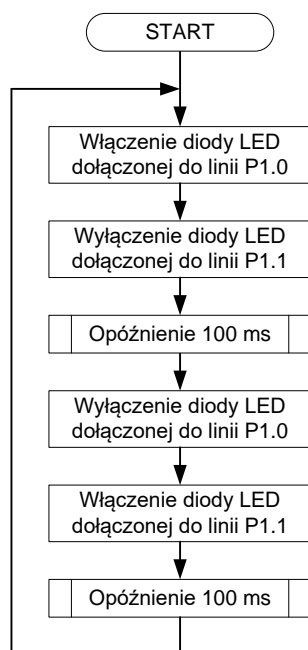
- W środowisku MCU 8051 IDE stwórz nowy projekt o nazwie *Cw04\_Zad04* z ustawieniami w części *Processor*, jak w zadaniu 1.
- Dołącz do dwóch najmłodszych linii portu P1 diody LED. W tym celu z menu *Virtual HW* wybierz *LED Panel*. Następnie dokonaj przyporządkowania linii portów jak na rysunku 3.



**Rys.3.** Widok okna *LED Panel* z właściwym dla zadania 4 dołączeniem diod LED do linii portu P1

Upewnij się, że klawisz w lewym górnym rogu okna jest w pozycji ON. Jeśli nie, przetącz go w tę pozycję.

- Skopiuj kod programu napisany w ramach zadania 3 i zmodyfikuj część główną programu w taki sposób, aby w nieskończonej pętli realizowane było zadanie ujęte algorytmem przedstawionym na rysunku 4. Do włączania i wyłączenia diod LED wykorzystaj instrukcje SETB i CLR.



**Rys.4.** Algorytm działania programu głównego dla punktu c) zadania 4

- d) Zapisz plik z kodem źródłowym programu i dokonaj jego asemblacji. W przypadku wykrycia błędów składniowych dokonaj modyfikacji kodu w celu ich usunięcia.
- e) Przetestuj działanie programu uruchamiając wbudowany w środowisko MCU 8051 IDE symulator. Jeśli program nie realizuje założonego zadania dokonaj jego modyfikacji. **Poprawnie działający program zaprezentuj prowadzącemu zajęcia.**
- ~~f) Po stwierdzeniu poprawnej pracy programu uruchom go na makiemie IME-LabTM z dołączoną do złącza P4 płytką IME-LabTM-DB01. Przed wgraniem programu do makiety zmień wartość występującą po dyrektywie CSEG-AT na 2000H i przeprowadź asemblację programu. Proces wgrywania programu do systemu mikroprocesorowego znajdującego się na makiemie został opisany w instrukcji do makiety.~~
- ~~g) Jeśli naprzemienne zapalanie i gaszenie diod LED nie będzie dobrze widoczne, w części głównej programu zwiększ opóźnienie pomiędzy przełączaniem diod LED do 500 ms przez powielenie wywołania podprogramu realizującego opóźnienie 100 ms.~~

## Sprawozdanie z ćwiczenia

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

- kod źródłowy programu realizującego opóźnienie (wraz z komentarzami) napisany w ramach realizacji zadania 1 oraz algorytm jego działania,
- zrzut ekrany programu MCU 8051 IDE po wykonaniu ostatniego rozkazu kodu programu z zadania 1 realizującego opóźnienie,
- kod źródłowy programu głównego i podprogramu realizującego opóźnienie (wraz z komentarzami) napisany w ramach realizacji zadania 2 oraz algorytmy ich działania,
- zrzut ekrany programu MCU 8051 IDE po wykonaniu ostatniego rozkazu kodu programu z zadania 2 realizującego opóźnienie,
- kod źródłowy programu głównego i podprogramu realizującego opóźnienie (wraz z komentarzami) napisany w ramach realizacji zadania 3 oraz algorytmy ich działania,

- zrzut ekranu programu MCU 8051 IDE po wykonaniu ostatniego rozkazu kodu programu z zadania 3 realizującego opóźnienie,
- opis słowny wyjaśniający sposób (wraz z wszelkiego rodzaju obliczeniami), w jaki zrealizowano opóźnienia w zadaniu 1, 2 i 3; opis ten powinien być przedstawiony pełnymi zdaniami.

Informacje dotyczące komentowania kodu programu oraz tworzenia schematów blokowych algorytmów pracy programów znajdują się w instrukcji do ćwiczenia 3.

## Wykorzystanie pułapek podczas symulacji

Podczas przeprowadzania symulacji wykonania programu przez mikrokontroler dobrze jest wykorzystać możliwość ustawiania pułapek (ang. *breakpoint*) w kodzie programu. Pozwalają one na zatrzymanie wykonywania kodu programu, gdy dojdzie do wykonywania rozkazu oznaczonego pułapką.

Aby ustawić pułapkę należy przy kursorze ustawionym nad numerem linii kliknąć lewy klawisz myszy. Spowoduje to pojawienie się czerwonego prostokąta, jak na rysunku 5.

```

1 CSEG AT 0000H
2
3 MOV R7,#9 ;Pierwszy rozkaz kodu opóźnienia
4 DJNZ R7,$
5 NOP ;Ostatni rozkaz kodu opóźnienia
6
7 SJMP $
8
9 END

```

**Rys.5.** Widok kodu programu z ustawioną pułapką

Należy pamiętać, że w momencie zatrzymania wykonywania kodu programu przez osiągnięci linii oznaczonej pułapką, rozkaz występujący w tej linii nie został jeszcze wykonany.



Użycie tej techniki w pomiarze czasu wykonania fragmentu kodu programu może przebiegać następująco. Załóżmy, że fragment kodu przedstawiony na rysunku 5 od linii 3 do 5 włącznie powinien realizować opóźnienie o czasie trwania 20 ms. Aby sprawdzić, czy tak jest rzeczywiście, należy postawić dwie pułapki: jedna na pierwszej linii kodu realizującego opóźnienie i drugą na linii kodu występującej bezpośrednio za ostatnią linią kodu realizującego opóźnienie. Pokazano to na rysunku 6.

```

1 CSEG AT 0000H
2
3 MOV R7,#9 ;Pierwszy rozkaz kodu opóźnienia
4 DJNZ R7,$
5 NOP ;Ostatni rozkaz kodu opóźnienia
6
7 SJMP $
8
9 END

```

**Rys.6.** Widok kodu programu z ustawionymi pułapkami umożliwiającymi pomiar czasu wykonania kodu fragmentu programu

Teraz możemy przejść do symulacji wykonania kodu programu przez mikrokontroler przez wciśnięcie klawisza z ikoną . Oczywiście wcześniej należy dokonać asemblacji kodu programu przez wciśnięcie klawisza z ikoną . Oczywiście, w przypadku wystąpienia błędów podczas asemblacji należy usunąć błędy i przeprowadzić ponowną asemblację kodu programu.


Ponieważ w rozpatrywanym przykładzie pierwszy rozkaz programu jest również pierwszym rozkazem kodu realizującego opóźnienie, dlatego po przejściu do symulacji następuje od razu zatrzymanie wykonywania kodu na linii 3. Pokazano to na rysunku 7.

```

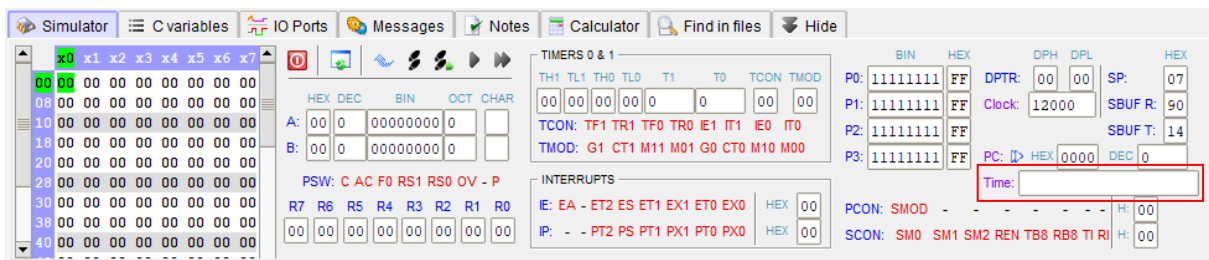
1 CSEG AT 0000H
2
3 MOV R7,#9 ;Pierwszy rozkaz kodu opóźnienia
4 DJNZ R7,$
5 NOP ;Ostatni rozkaz kodu opóźnienia
6
7 SJMP $
8
9 END

```

**Rys.7.** Widok kodu programu po uruchomieniu symulacji (na zielono zaznaczony jest rozkaz, który zostanie wykonany)


Gdyby rozkaz, od którego chcemy mierzyć czas wykonania przez mikrokontroler pewnego fragmentu programu nie był pierwszym rozkazem w całym programie, należy wcisnąć klawisz  znajdujący się na zakładce *Simulator*. Spowoduje to wykonywanie kodu programu aż do napotkania pierwszej pułapki.

W chwili zatrzymania wykonywania programu na początku kodu realizującego opóźnienie należy w polu *Time*, zaznaczonym czerwoną ramką na rysunku 8, sprawdzić, jaki czas upłynął od początku uruchomienia wykonywania kodu programu.



**Rys.8.** Widok zakładki *Simulator* z zaznaczonym polem *Time*

W pokazanym przykładzie pole to jest puste, ponieważ jeszcze nie został wykonany żaden rozkaz programu. Dlatego przyjmujemy, że czas wynosi 0  $\mu$ s.

Aby wykonać fragment kodu realizujący opóźnienie, klikamy klawisz z ikoną . Wykonywanie programu zatrzyma się wtedy na następnej pułapce (patrz rysunek 9).

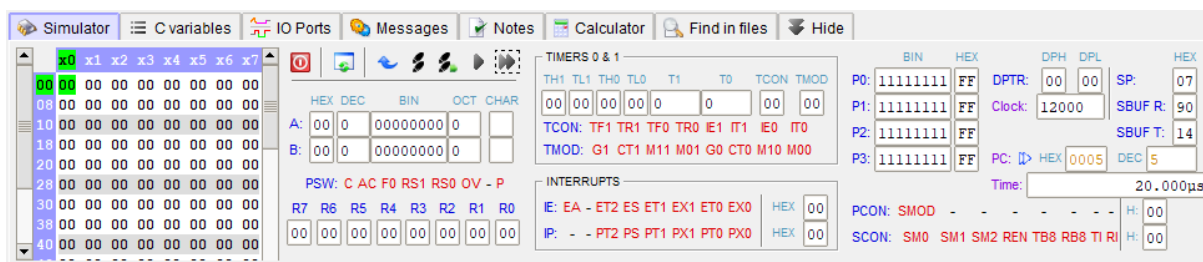
```

1 CSEG AT 0000H
2
3 MOV R7,#9 ;Pierwszy rozkaz kodu opóźnienia
4 DJNZ R7,$
5 NOP ;Ostatni rozkaz kodu opóźnienia
6
7 SJMP $
8
9 END

```

**Rys.9.** Widok kodu programu po zatrzymaniu się na drugiej pułapce

Tym razem w polu *Time* widzimy wartość 20  $\mu$ s, co pokazano na rysunku 10.



**Rys.10.** Widok zakładki *Simulator* po zatrzymaniu się na drugiej pułapce

Jeśli teraz od drugiego czasu (20 µs) odejmiemy pierwszy (0 µs), to uzyskamy czas, w jakim został wykonany kod programu od linii 3 do 5 włącznie (20 µs).

Należy pamiętać, że czas wykonania rozkazu zależy od przyjętej częstotliwości sygnału zegarowego taktującego mikrokontroler. Wartość częstotliwości tego sygnału, wyrażoną w kHz, wpisuje się w pole *Clock*, znajdujące się trochę powyżej pola *Time*.

## Literatura

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