	LABORATORIUM ***PODSTAWY TECHNIKI MIKROPROCESOROWEJ***
L IV	***Sterowanie wskaźnikiem binarnym***

I. PRZYPOMNIENIE

1. Tworzenie pliku źródłowego

Plik źródłowy powinien zawierać następujące elementy:

- dyrektywę NAME,
- dyrektywę CSEG,
- program użytkowy,
- dyrektywę kończącą program END.

PRZYKŁAD

```
NAME czerwone
CSEG at 0000h
     clr P3.3
     jmp $
END
```

2. Tworzenie nowego projektu

A. Z menu *Project* wybierz opcję *Close Project*... w celu zamknięcia domyślnego projektu.

B. Z menu *Project* wybierz opcję *New Project*... W katalogu *d:\Student\Lxx* utwórz nowy projekt.

C. W okienku *CPU* wybierz odpowiedni typ procesora (ADuC 845).

D. Korzystając z menu *File/New*... otwórz okienko edytora plików źródłowych, a następnie przy pomocy menu *File/Save As*...utwórz nowy plik.

E. Klikając prawym klawiszem myszy na folderze *Source Group 1* w okienku menadżera projektów rozwiń lokalne menu, wybierz opcję *Add Files to Group 'Source Group 1'* i dodaj utworzony plik do projektu.

3. Ustawianie opcji środowiska

Kliknij prawym klawiszem myszy na folderze *Target 1* w okienku menadżera projektów, rozwiń lokalne menu i wybierz opcję *Options for Target 'Target 1'*.

Sprawdź ustawienia w zakładce *Debug!!!*.

4. Kompilacja projektu

A. *Project/Build Target* − kompilowanie tylko tych plików, które zostały zmienione po ostatnim procesie kompilacji,

B. *Project/Rebuild all target files* − ponowne przekompilowanie wszystkich plików,

C. *Project/Translate d:\Student\Lxx\test.asm* – przeprowadzenie procesu asemblacji.

5. Uruchomienie programu

A. Włącz zasilacz płytki ADuC 845 – powinna się zaświecić dioda LED.

B. Wprowadź mikroprocesor ADuC 845 w tryb debagowania – przytrzymaj przycisk *SERIAL DOWNLOAD*, a następnie wciśnij przycisk *RESET*.

C. Korzystając z menu *Debug / Start/Stop Debug Session* uruchom debugger.

UWAGA

Jeżeli system ADuC845 nie został wyzerowany pojawi się okienko informujące o oczekiwaniu na sygnał RESET. Należy wówczas ponownie przeprowadzić procedurę wprowadzania w tryb debagowania.

Wybranie funkcji *Go* powoduje uruchomienie programu użytkownika w systemie ADuC 845. Debugger traci wówczas kontrolę nad systemem.

Wybierając opcję *Stop Running* z menu *Debug* można przerwać działanie funkcji *Go*, lecz program w systemie ADuC 845 będzie wykonywany nadal.

W celu ponownego nawiązania komunikacji należy całkowicie opuścić *Debugger* (przycisk *Stop Debugging* w okienkach dialogowych) i ponownie go uruchomić z menu *Debug / Start/Stop Debug Session*.

II. PRZEBIEG ĆWICZENIA

A. Zapoznaj się ze schematem ideowym płytki z linijkami diodowymi (schemat dołączony na końcu instrukcji).

B. Napisz i uruchom poniższe programy.

PROG. 1.

Zaświecenie linijki diod czerwonych poprzez odpowiednie ustawienie bitu sterującego P3.3.

Wskazówki:

- plik źródłowy powinien zawierać 5 linijek programu.

PROG. 2.

Zaświecenie linijki diod zielonych poprzez odpowiednie ustawienie bitów portu P3 oraz P2.

Wskazówki:

- uaktywnij linijkę diod zielonych rozkazem *clr P3.5;*
- wyzeruj port P2 rozkazem *mov P2,#0*.

PROG. 3.

Cykliczne świecenie i gaszenie jednej diody.

Wskazówki:

- wykorzystaj rozkaz **cpl,**
- uruchom program w trybie krokowym i w trybie *Go*.

PRZYKŁAD

```
NAME dioda
CSEG at 0000h
     clr P3.3
     clr P3.5
     mov P2,#0
Petla:
     cpl P2.0
     jmp Petla
END
```

PROG. 4.

Cykliczne świecenie i gaszenie wszystkich diod czerwonych.

Wskazówki:

- wykorzystaj rozkaz **cpl,**
- uruchom program w trybie krokowym i w trybie *Go.*

PRZYKŁAD

```
NAME linijka
CSEG at 0000h
     clr P3.3
Petla:
     mov A,P2
     cpl A
     mov P2,A
     jmp Petla
END
```

PROG. 5.

Cykliczne świecenie i gaszenie diod z wykorzystaniem pętli opóźniających.

Wskazówki:

- wykorzystaj rozkaz **cjne** lub **djnz;**
- uruchom program w trybie GO;
- jeżeli diody nadal nie „mrugają" wykorzystaj dwie pętle opóźniające (umieść jedną w środku drugiej).

PRZYKŁAD

```
NAME miganie_cjne
CSEG at 0000h
     clr P3.3
Petla:
     mov A,P2
     cpl A
     mov P2,A
     mov R0,#0
     mov R1,#0
Stop1:
     inc R0
     cjne R0,#200,Stop1
     mov R0,#0
     inc R1
     cjne R1,#250,Stop1
     jmp Petla
END
```

PRZYKŁAD

```
NAME miganie_djnz
CSEG at 0000h
     clr P3.3
Petla:
     mov A,P2
     cpl A
     mov P2,A
     mov R0,#200
     mov R1,#250
Stop1:
     djnz R0,Stop1
     mov R0,#200
     djnz R1,Stop1
     jmp Petla
END
```

PROG. 6.

Wyślij na wyświetlacz wynik działania dowolnej operacji arytmetyczno logicznej (np. add, orl, anl).

Wskazówki:

- wykorzystując pętle opóźniające możesz również wysłać kolejno na wyświetlacz argumenty operacji.

C. Zapamiętaj składnię i znaczenie poniższych rozkazów.

cjne R0,#255,etykieta

znaczenie – porównaj R0 z liczbą 255 (dziesiętną)

jeżeli R0≠255, to skocz do *etykieta*

jeżeli R0=255, to wykonaj następny rozkaz

djnz R0,etykieta

znaczenie – zmniejsz R0 o jeden

jeżeli R0≠0, to skocz do *etykieta*

jeżeli R0=0, to wykonaj następny rozkaz

III. DODATKOWE ZAGADNIENIA

1. Zegar systemowy

Mikroprocesor ADuC 845 jest stabilizowany rezonatorem kwarcowym (zegarkowym) 32,768kHz. Częstotliwość ta jest zwielokrotniana przy użyciu wbudowanej pętli PLL, przy czym wartość mnożnika jest zależna od ustawienia w rejestrze PLLCON.

Po sygnale RESET domyślnie jest ustawiona częstotliwość cyklu maszynowego równa 1,572864MHz.

Liczba cykli maszynowych potrzebnych do zrealizowania pojedynczego rozkazu wynosi 1 – 9 i jest zależna od rodzaju wykonywanej operacji.

PRZYKŁAD

Jaki jest czas potrzebny do wykonywania instrukcji *mov R0,#0*:

- potrzebna liczba cykli maszynowych wynosi 2 (patrz lista rozkazów mikroprocesora ADuC 845),
- czas trwania jednego cyklu maszynowego 1/1,57286MHz = 0,63578μs ≈ 0,63μs,
- czas realizacji przykładowego rozkazu (czas trwania cyklu rozkazowego) Tr = 2 x 0,63μs = 1,27μs.

2. Obliczanie czasu trwania realizacji programu

A. Zmodyfikuj ostatni program PROG. 5.

PROG. 7.

Ustaw okres migania diod na ≈2 sekundy.

B. Oblicz czas wykonywania poszczególnych programów i pętli programowych z punktu II. 2.

IV. PRZYGOTOWANIE DO NASTĘPNYCH ZAJĘĆ

1. Wiedza teoretyczna

A. Cykl rozkazowy.

B. System wejścia – wyjścia.

2. Wiadomości z ćwiczenia pierwszego

A. Umiejętność posługiwania się poznanymi rozkazami.

B. Znajomość zasad tworzenia pętli opóźniających.

C. Znajomość kodu źródłowego pisanych programów.