Technika Mikroprocesorowa

Laboratorium 1

Wprowadzenie do środowiska programowego, podstawowe operacje na rejestrach i komórkach pamięci.

Cel ćwiczenia: Podstawowym celem ćwiczenia jest zapoznanie się ze środowiskiem programowym do którego należy:

- Środowisko programistyczne **M-IDE**,
- Symulator **SIM-52**.

Drugi cel to napisanie pierwszego programu i jego uruchomienie.
W ramach ćwiczenia należy opanować:

- operacje na plikach(open, close, save),
- edycję programu,
- ustawianie parametrów programów,
- uruchamianie napisanych programów na symulatorze,
- uruchamianie krokowe,
- zastawianie pułapek,
- obserwacja rejestrów.

Zadania do wykonania:

1. Przeczytać opis srodowiska.pdf

2. Uruchomić **M-IDE** zawiera edytor, kompilator asemblera (asemblator) oraz symulator, jednak na zajęciach będzie używany symulator **SIM-52**

3. W M-IDE w okienku edycji proszę wpisać poniższy kawałek kodu:

```
        MOV A,#22H      ; instrukacja MOV wpisuje liczbę 22H do
                        ; akumulatora A (ACC)

KONIEC: LJMP KONIEC     ;"KONIEC:" jest etykietą (label) czyli znacznikiem do
                        ; którego można się odowłać, tak jak to jest zrobione
                        ; za pomocą instrukcji „LJMP". Czyli procesor
                        ; wykonując instrukcję"LJMP KONIEC" skacze do
                        ; etykiety „KONIEC" czyli do samej siebie.
                        ; „W ten sposób „zatrzymujemy" program

        END             ; ta instrukcją kończy program
; po średniku wpisujemy komentarz który jest ignorowany przez asemblator
```

UWAGA1:

Każdy program należy kończyć pętlą nieskończoną np.
KONIEC: LJMP KONIEC
Oraz instrukcją
END

4. Napisany program należy zapisać w wybranym katalogu, w nazwie pliku proszę nie używać polskich liter oraz znaków specjalnych.

5. Skompilować program **BUILD/BUILD lub F9**

6. W wybranym katalogu powstaną 2 dodatkowe pliki:

*.lst
*.hex

*.hex to plik zawierający kod maszynowy i służy do programowania pamięci programu
*.lst otwieramy w SIM-52

7. Otworzyć SIM-52

8. Otwieramy plik *.lst

9. Wykonać program krokowo (**klawisz strzałka w prawo ->**)

10. Zapisać wybraną liczbę do patrz **UWAGA2**:
    a. rejestru R1,
    b. komórki pamięci RAM o adresie 50H,
    c. komórki pamięci XRAM o adresie 50H
    d. portu P1.

11. Napisać program w asemblerze sumujący zawartość dwóch rejestrów P1 i P2 i zapisujący wynik dodawania do komórki pod wybranym adresem:

(R1) + (R2) -> (adres #3) – wybrany adres w RAM i dodatkowo wynik zapisać także w XRAM

UWAGA2

```
MOV A,#22H      ;wpisuje do A liczbę 22H (# oznacza że 22H jest stałą
MOV A, 22H      ;wpisuje do A wartość komórki o adresie 22H w pamięci RAM (takie
                ;adresowanie nazywamy bezpośrednim )
MOV A, R1       ;wpisuje do A wartość z rejestru R1
MOV R1,#12H     ;wpisuje do R1 wartość z 12H użyjemy tej liczby jako asresu w
                ;następnej instrukcji
MOV A, @R1      ;wpisuje do A wartość z komórki pamięci RAM o adresie znajdującym
                ;się w rejestrze R1 (takie adresowanie nazywamy pośrednim )
```

Oczywiście instrukcje:

```
MOV 22H,A      ;wpisuje do komórki o adresie 22H w pamięci RAM wartość A
MOV R1,A       ;wpisuje do R1 wartość z rejestru A
MOV @R1,A      ;wpisuje do komórki pamięci RAM o adresie znajdującym się
               ;w rejestrze R1 wartość rejestru A (akumulatora)
```

Do komunikacji z zewnętrzną pamięcią RAM (XRAM) używamy instrukcji MOVX . Do komunikacji z XRAM zawsze korzystamy z adresowania pośredniego z użyciem rejestru szesnastobitowego DPTR lub rejestrów indeksowych Ri (R1, R2)

Odczyt komórki pamięci XRAM

```
MOVX A,@Ri     ;wpisuje do A wartość z komórki pamięci XRAM o adresie
               ;znajdującym się w rejestrze R0 lub R1. Przy czym w rejestrach
               ;tych znajduje się młodszy bajt adresu a w P2 starszy bajt adresu
MOVX A,@DPTR   ;wpisuje do A wartość z komórki pamięci RAM o adresie
               ;znajdującym się w rejestrze DPTR.  Rejestr DPTR skałada się
               ;z DPH i DPL i możemy starszy i młodszy bajt ustawić za
               ;pomocą instrukcji MOV
```

Zapis komórki pamięci XRAM

```
MOVX @Ri,A     ; adresowanie analogiczne jak powyżej
MOVX @DPTR,A   ; adresowanie analogiczne jak powyżej
```

Elementy wymagane przy sprawozdaniu:

- Napisany program z komentarzami (kod oraz opis programu)
- rozszyfrować kody maszynowe z pliku *.hex i każdemu kodowi przyporządkować odpowiedni mnemonik/komendę asemblera.
- opis użytych rozkazów,
- opis zmienionych rejestrów.

SKRÓTOWY OPIS formatu HEX

Przykład pliku HEX

```
:10010000214601360121470136007EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```

Start code – (czyli ":")

Byte count – liczba bajtów danych w lini

Address – od którego są wpisywane bajty do pamieci stałej (ROM, EPROM....)

Record type – Typ rekordu

- **00**, rekord danych zawierajacy 16 bitowy adres.
- **01**, Koniec pliku. Zwykle wygląda tak ':00000001FF'.
- **02**,Rozszerzony *Extended Segment Address Record*, segment-base address. Used when 16 bits are not enough, identical to 80x86 real mode addressing. The address specified by the 02 record is multiplied by 16 (shifted 4 bits left) and added to the subsequent 00 record addresses. This allows addressing of up to a megabyte of address space. The address field of this record has to be 0000, the byte count is 02 (the segment is 16-bit). The least significant hex digit of the segment address is always 0.
- **03**, Rekord z adresem początkowy segmenty, procesory 80x86.
- **04**, Rekord z rozszerzonym adresem (32bity).
- **05**, rekord z adresem 32bitowym dla procesorów 80386..

Data – bajty danych

Checksum – suma kontrolna