Instytut Teleinformatyki

Wydział Fizyki, Matematyki i Informatyki
Politechnika Krakowska

Mikroprocesory i mikrokontrolery

„Wstęp do programowania w asemblerze 8051”

laboratorium: 01

autor: mgr inż. Michał Lankosz
dr hab. Zbisław Tabor, prof. PK

Kraków, 2014

Spis treści

Spis treści 2
1. Wiadomości wstępne 3
1.1. Niezbędne wiadomości 3
1.2. Szablon programu w asemblerze 8051 3
1.3. Środowisko programistyczne 4
2. Przebieg laboratorium 6
2.1. Zadanie 1. Na ocenę 3.0 (dst) 6
2.2. Zadanie 2. Na ocenę 4.0 (db) 7
2.3. Zadanie 3. Na ocenę 5.0 (bdb) 8

1. Wiadomości wstępne

Pierwsza część niniejszej instrukcji zawiera podstawowe wiadomości teoretyczne dotyczące omawianego tematu. Poznanie tych wiadomości umożliwi prawidłowe zrealizowanie praktycznej części laboratorium.

1.1. Niezbędne wiadomości

Pamięć programu, wewnętrzna pamięć RAM, mapa pamięci, rejestry, tryby adresowania: http://www.8052.com/tut8051.
Lista instrukcji asemblera 8051:
http://www.keil.com/support/man/docs/is51/is51_instructions.htm

Laboratorium asemblera 8051 jest realizowane w środowisku developerskim Keil microVision (środowisko cross-kompilacji C51 i symulacji pracy 8051), do fizycznego oprogramowania mikrokontrolera używamy programator Flip 3.4.7.

1.2. Szablon programu w asemblerze 8051

```
;-------------------------------------------------------------------------
; Definicje używanych stałych
;-------------------------------------------------------------------------
; LED EQU P1.6          ;przykładowa definicja stałej LED
;-------------------------------------------------------------------------
; Wektory przerwań i resetu
;-------------------------------------------------------------------------
; Reset Vector - do tego adresu wykonywany jest skok po resecie
   CSEG AT 00h
   LJMP Main            ; Skok poza przestrzeń adresową wektorów przerwań

; Interrupt vectors - jeśli dodajemy swoją obsługę przerwań
;    CSEG At 03h         ; wektor przerwania INT0 - podpięta procedura handleINT0
```



```
;   LJMP handleINT0

;-------------------------------------------------------------------------
; segment głównej funkcji programu
;-------------------------------------------------------------------------
CSEG AT 0x100

Main:
      ;
      ; główna funkcja programu
      ; ACALL MyProcedure       ; z funkcji głównej możemy wołać procedury
      ;
SJMP Main       ; pętla nieskończona - system wbudowany funkcjonuje tak długo jak długo jest
                ; zasilany

;-------------------------------------------------------------------------
; definicje procedur
;-------------------------------------------------------------------------
;MyProcedure:          ; procedura wołana przez funkcję Main
;
;RET                ; wyjście z procedury

;handleINT0:           ; procedura obsługi przerwania INT0
;
;RETI               ; wyjście z procedury obsługi przerwania

;-------------------------------------------------------------------------
; definicje danych
;-------------------------------------------------------------------------
;CSEG AT 20FFh
;squares:    db  0,1,1,1,2,2,2,2,2,3,3,3,3,3,3,3       ; przykładowe dane bajtowe zapisane pod adre-
sem 20FFh

END
```

1.3. Środowisko programistyczne

W trakcie laboratoriów z programowania kontrolera 8051 wykorzystywane jest środowisko µVision. Po uruchomieniu IDE zarządzanie projektami odbywa się z poziomu menu **Project**. Przy tworzeniu nowego projektu należy z reguły określić szereg opcji, które wpływają na proces budowania aplikacji dla docelowego procesora. Przede wszystkim należy określić rodzaj procesora: w przypadku używanego na laboratorium zestawu ZL2MCS51 jest to procesor Atmel, AT89C51RD2. Opcje te można zmieniać po utworzeniu projektu, wybierając projekt w oknie **Project** (jeśli niewidoczne, wybieramy polecenie **View->Project**) i wybierając polecenie **Options for Target** w menu **Project** (dostępne też pod prawym klawiszem myszy). Spośród innych opcji można zwrócić uwagę na możliwość określenia częstotliwości rezonatora kwarcowego: (menu **Project**, **Options for Target**, zakładka **Target**, pole **Xtal**). Laboratoryjne zastawy ZL2MCS51 posiadają rezonator kwarcowy o częstotliwości 11,0592MHz. Poprawne określenie częstotliwości pozwala w trybie debugowania wyznaczyć czas wykonywania fragmentów kodu. Kolejną opcją ważną przy oprogramowywaniu fizycznego kontrolera jest format wykonywalnego pliku, ustawiany na karcie **Output** okna **Options for Target**. Na karcie tej należy zaznaczyć opcję **Create HEX file**.

Po stworzeniu nowego projektu można (czasami nawet należy) do niego dodać plik z kodem źródłowym (**File->New...**). Stworzony plik źródłowy należy dodać do projektu (prawy klawisz na nazwie projektu w oknie **Project**, lub menu **Project**, polecenie **Manage**). W celu kompilacji należy wybrać polecenie **Build** z menu **Project** (skrót klawiaturowy F7). Środowisko µVision umożliwia symulację działania programu. W tym celu należy z menu **Debug** wybrać opcje **Start/Stop debug session**. Symulator umożliwia podgląd stanu elementów mikrokontrolera, w tym rejestrów (**View->Registers Window**), pamięci programu i IRAM (**View->Memory Windows**), wejść/wyjść (**Peripherals->I/O Ports**). Po starcie programu przejście do następnej instrukcji następuje w trybie debugowania po naciśnięciu klawisza F11 (lub menu **Debug->Step** lub odpowiedni przycisk na pasku narzędzi). Przy przeglądaniu mapy pamięci należy w oknie **Memory** wpisać w polu **Address** adres pierwszego wyświetlanego bajtu n.p. C:1000h jeśli chcemy wyświetlić zawartość pamięci programu od adresu 1000h lub I:20h jeśli chcemy wyświetlić zawartość wewnętrznej RAM od adresu 20h. Zawartość pamięci można w trybie debugowania modyfikować, klikając dwukrotnie na komórkę pamięci.

Literatura:

[1] http://www.8052.com/tut8051
[2] http://www.keil.com/support/man/docs/is51/is51_instructions.htm

2. Przebieg laboratorium

Druga część instrukcji zawiera zadania do praktycznej realizacji, które demonstrują zastosowanie technik z omawianego zagadnienia.

2.1. Zadanie 1. Na ocenę 3.0 (dst)

Należy przeanalizować pracę mikrokontrolera poprzez wykonanie symulacji krokowej instrukcja po instrukcji na podstawie programu przedstawionego na Listingu 1.

Listing 1

```
Stala EQU 1000
DanaL DATA 20h
DanaH DATA 21h
WynikL DATA 30h
WynikH DATA 31h

CSEG AT 0
JMP start

CSEG AT 100h
  start:
    MOV A,DanaL
    ADD A,#low(Stala)
    MOV WynikL,A
    MOV A,DanaH
    ADDC A,#high(Stala)
    MOV WynikH,A
END
```


Po załadowaniu programu w symulatorze należ.y zmienić zawartość komórek pamięci wewnętrznej RAM:

pod adres 20h wpisać wartość 09h
pod adres 21h wpisać wartość 0Ah

Proszę prześledzić działanie programu w trybie debugowania, zwracając uwagę na wszystkie zmiany rejestrów i pamięci RAM.
Proszę sprawdzić poprawność obliczeń (dodawania szesnastobitowego). W jaki sposób są zapisane liczby 16-bitowe w pamięci?

Proszę po zresetowaniu programu, zmienić zawartość pamięci:

pod adres 20h wpisać wartość 18h
pod adres 21h wpisać wartość 00h

i ponownie prześledzić działanie zwracając uwagę na flagę przeniesienia (CY).
Proszę zmodyfikować program tak, aby w podobny sposób wykonywał odejmowanie dwóch liczb 16–bitowych (rozkaz SUBB) również. ze sprawdzeniem działania.

2.2. Zadanie 2. Na ocenę 4.0 (db)

Proszę w trybie debugowania przeanalizować działanie programu kopiującego dane z jednego obszaru pamięci RAM do innego (Listing 2). Proszę zwrócić uwagę na adresowanie komórek pamięci.

Listing 2

```
MY_KILLING_COPY SEGMENT CODE
Zrodlo DATA 20h
Cel DATA 30h
IleDanych EQU 16

CSEG AT 0
JMP start

RSEG MY_KILLING_COPY
  start:
    MOV R0,#Zrodlo
    MOV R1,#Cel
```



```
    MOV R3,#IleDanych
  Petla:
    MOV A,@R0
    MOV @R1,A
    INC R0
    INC R1
  DJNZ R3,Petla
END
```

W przykładzie przedstawionym na Listingu 2 kod programu nie jest umiejscawiany na sztywno w pamięci programu pod zadanym przez programistę adresem (dyrektywa CSEG na Listingu 1). Użyta tu dyrektywa RSEG w odniesieniu do segmentu kodu powoduje, że asembler ulokuje kod w najbardziej (według niego) wygodnym miejscu pamięci kodu.

Proszę napisać program, który skopiuje dane z pamięci programu do wewnętrznej pamięci RAM. Przydatne instrukcje:

MOV DPTR,#etykieta
MOVC A,@A+DPTR

Do umieszczenia danych w pamięci programu służy dyrektywa DB, na przykład użyta w taki sposób:

etykieta: db 11,21,4,18,8

przy czym najprościej jest tą etykietę umieścić na końcu programu, czyli poniżej ostatniego rozkazu. Etykiety wskazują fizyczny adres wyrażenia znajdującego się tuż po niej, czyli w powyższym przypadku "etykieta" jest adresem komórki w pamięci programu, w której znajduje się pierwsza liczba tablicy danych - "11" – i ten adres podajemy w instrukcji ładującej wskaźnik DPTR.

2.3. Zadanie 3. Na ocenę 5.0 (bdb)

Generator liczb pseudolosowych oparty o opóźniony ciąg Fibonacciego jest opisany na stronie Wikipedii. Proszę napisać w asemblerze generator liczb pseudolosowych dla pary (j,k) = (7, 10) z operacją dodawania, generujący liczby całkowite w przedziale od 0 do 3. Dziesięć startowych liczb (ja używałem 4, 5, 3, 3, 3, 4, 6, 2, 6, 5) powinno być zapisanych w pamięci programu i skopiowanych do pamięci IRAM przez procedurę inicjalizującą generator. Procedura generująca liczby powinna zwracać liczbę pseudolosową w rejestrze A. Ponieważ każda nowo wygenerowana liczba będzie użyta do wygenerowania którejś z następnych liczb pseudolosowych, wynik losowania

musi również trafiać do pamięci IRAM, w której należy zorganizować bufor cykliczny na generowane liczby pseudolosowe.

