

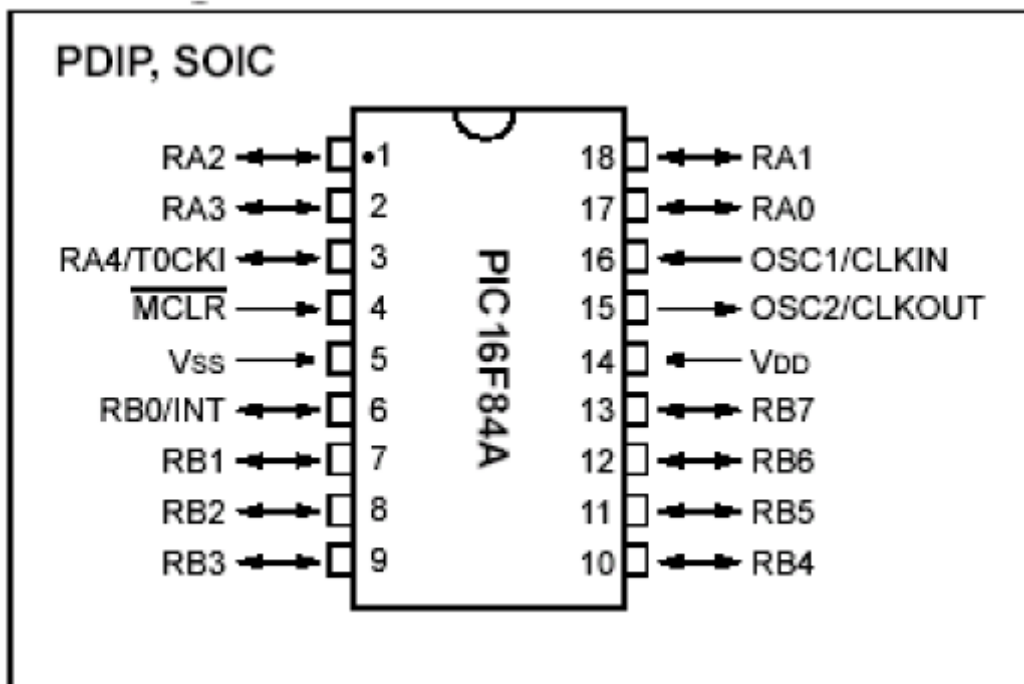
Laborator 3

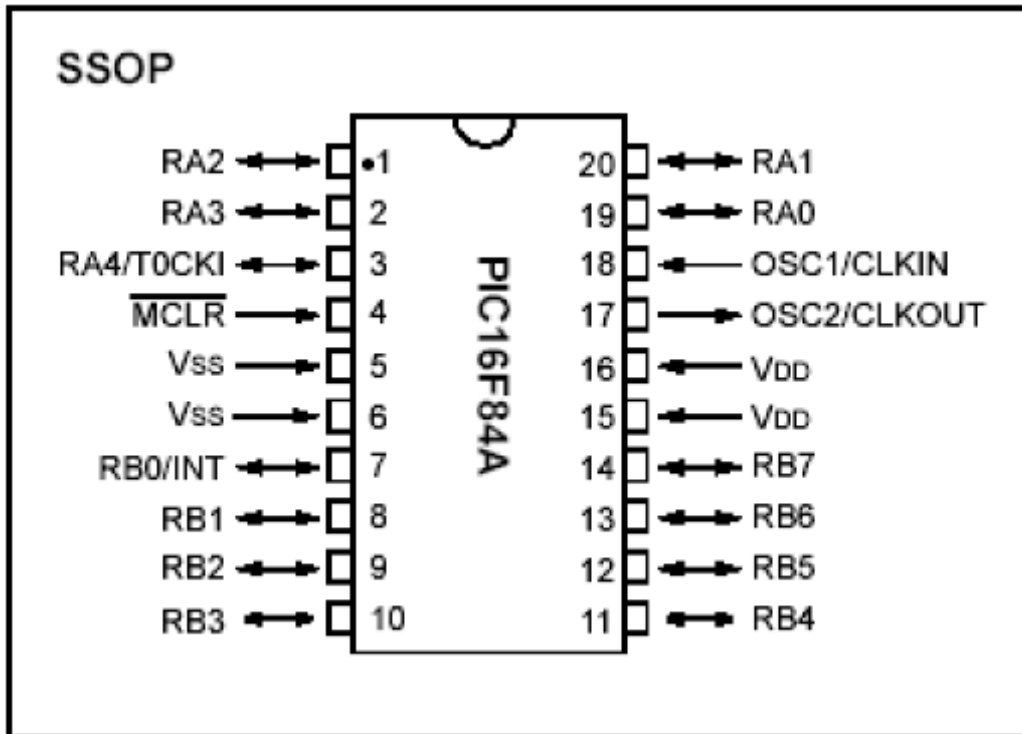
Detalii de arhitectura

PIC16F84 este un microcontroller RISC pe 8 biti. Functioneaza pe o arhitectura de tip Harvard. Are un numar de doar 35 de instructiuni. Toate instructiunile se executa intr-un ciclu cu exceptia salturilor care se executa in doua cicluri. Are 68 octeti de RAM si 64 octeti de EEPROM. Instructiunile sunt codificate pe 14 biti. Utilizeaza o magistrala de date pe 8 biti . Contine o stiva de date pe 8 niveluri. Exista 15 registre cu functii speciale (SFR) . Modurile de aderasre pot fi: directa, indirecta si relativa.

Are 4 surse de intreruperi :

- Pinul extern RB0/INT
- La depasirea TMR 0 (overflow)
- La schimbarea starii pinilor<4:7>
- Al incheierea scrierii in EEPROM





Memoria program a microcontrollerului contine 1024 cuvinte, care pot contine 1024 instructiuni. (un cuvint din memoria de program are 14 biti) .

Pentru programare este indicata folosirea fisierului header oferit de Microchip, P16F84A.inc .El contine definirea configurarii, denumirea registrelor si adresele lor, denumirea bitilor din registre. Pentru o programare mai relaxata, putem asocia anumite adrese din memoria cu anumite denumiri. De exemplu STATUS pentru registrul STATUS. Aceasta varianta de programare are doua mari avantaje: ofera usurinta prin lipsa necesitatii de memorare a adreselor pentru SFR si ofera portabilitate. In cazul in care vrem sa folosim acelasi program pentru alt microcontroller este suficient sa schimbam fisierul header, deoarece etichetele esentiale(asa cum este STATUS) sunt redefinite acolo.

Fisierul header pentru PIC16F84A este listat mai jos.

LIST

; P16F84A.INC Standard Header File, Version 2.00 Microchip Technology, Inc.

NOLIST

; This header file defines configurations, registers, and other useful bits of
; information for the PIC16F84 microcontroller. These names are taken to match
; the data sheets as closely as possible.

; Note that the processor must be selected before this file is
; included. The processor may be selected the following ways:

; 1. Command line switch:

; C:\ MPASM MYFILE.ASM /PIC16F84A

; 2. LIST directive in the source file

; LIST P=PIC16F84A

; 3. Processor Type entry in the MPASM full-screen interface

```

;=====
; Revision History
;=====
;Rev: Date: Reason:
;1.00 2/15/99 Initial Release
;=====
; Verify Processor
;=====
IFNDEF __16F84A
MESSG "Processor-header file mismatch. Verify selected processor."
ENDIF
;=====
; Register Definitions
;=====
W EQU H'0000'
F EQU H'0001'
;---- Register Files-----
INDF EQU H'0000'
TMR0 EQU H'0001'
PCL EQU H'0002'
STATUS EQU H'0003'
FSR EQU H'0004'
PORTA EQU H'0005'
PORTB EQU H'0006'
EEDATA EQU H'0008'
EEADR EQU H'0009'
PCLATH EQU H'000A'
INTCON EQU H'000B'
OPTION_REG EQU H'0081'
TRISA EQU H'0085'
TRISB EQU H'0086'
EECON1 EQU H'0088'
EECON2 EQU H'0089'
;---- STATUS Bits -----
IRP EQU H'0007'
RP1 EQU H'0006'
RP0 EQU H'0005'
NOT_TO EQU H'0004'
NOT_PD EQU H'0003'
Z EQU H'0002'
DC EQU H'0001'
C EQU H'0000'
;---- INTCON Bits -----
GIE EQU H'0007'
EEIE EQU H'0006'
TOIE EQU H'0005'

```

```

INTE EQU H'0004'
RBIE EQU H'0003'
T0IF EQU H'0002'
INTF EQU H'0001'
RBIF EQU H'0000'
;----- OPTION_REG Bits -----
NOT_RBPU EQU H'0007'
INTEDG EQU H'0006'
T0CS EQU H'0005'
T0SE EQU H'0004'
PSA EQU H'0003'
PS2 EQU H'0002'
PS1 EQU H'0001'
PS0 EQU H'0000'
;----- EECON1 Bits -----
EEIF EQU H'0004'
WRERR EQU H'0003'
WREN EQU H'0002'
WR EQU H'0001'
RD EQU H'0000'
;=====
; RAM Definition
;=====
_MAXRAM H'CF'
_BADRAM H'07', H'50'-H'7F', H'87'
;=====
; Configuration Bits
;=====
_CP_ON EQU H'000F'
_CP_OFF EQU H'3FFF'
_PWRTE_ON EQU H'3FF7'
_PWRTE_OFF EQU H'3FFF'
_WDT_ON EQU H'3FFF'
_WDT_OFF EQU H'3FFB'
_LP_OSC EQU H'3FFC'
_XT_OSC EQU H'3FFD'
_HS_OSC EQU H'3FFE'
_RC_OSC EQU H'3FFF'
LIST
*****

```

Organizarea memoriei

Memoria este organizata pe doua zone: memoria de program si memoria de date. Fiecare bloc are propria magistrala. De aceea cele doua zone pot fi accesate in acelasi ciclu. Memoria de date se imparte in : memoria RAM de uz general si registrele cu

functii speciale SFR . Memoria EEPROM se afla intre adresele 0H – 03FH si este asociata blocului memoriei de date. Ea poate fi accesata doar indirect.

Memoria de program

PIC16F84 are un indicator de adresa numit PC. El poate indica o adresa intre 0000H si 03FFH. Accesarea unei locatii care depaseste 03FFH genereaza o trunchiere. De exemplu la adresele 40H si 440H se afla acelasi lucru. Vectorul RESET se afla la adresa 000H, iar cel de intrerupere la 0004H. Daca nu se folosesc intreruperi, programul se poate organiza continuu de al adresa 0000H.

Memoria de date

Este organizata in doua zone: SPR(Special Purpose Register) si GPR(General Purpose Registers).

adresa	Banc 0	Banc 1	adresa
00H	INDF*	INDF*	80H
01H	TMR0	OPTION	81H
02H	PCL	PCL	82H
03H	STATUS	STATUS	83H
04H	FSR	FSR	84H
05H	PORTA	TRISA	85H
06H	PORTB	TRISB	86H
07H			87H
08H	EEDATA	EECON1	88H
09H	EEADR	EECON2	89H
0AH	PCLATH	PCLATH	8AH
0BH	INTCON	INTCON	8BH
0CH			8CH
.			.
.			.
.			.
4FH			CFH
50H			D0H
.			.
.			.
.			.
7FH			FFH

harta memoriei

Dupa cum se poate observa din harta memoriei, exista doua bancuri. Fiecare are 12 adrese rezervate pentru SFR si 68 de adrese de uz general. Ultimele 48 de adrese nu sunt implementate fizic. Memoria poate fi adresata direct prin adresa sau prin nume (pentru SFR) sau cu ajutorul registrelor speciale FSR si INDF . FSR contine adresa iar INDF va contine valoarea de la acea adresa.

Este foarte importanta intelegerea modului de utilizare a registrului STATUS.

Bit	7	6	5	4	3	2	1	0
Denumire	IRP	RP1	RP0	TO	PD	Z	DC	C

Semnificatia acestor biti este:

C	fanion de transport
DC	fanion de transport pe niblu
Z	fanion de zero
PD	cadere de puetre (activ pe 0)
TO	time-out (activ pe 0)
RPO	selectia segmentului

Aplicatie :

Compararea a doua variabile:

Creati un proiect nou folosind PIC16F84A. Deschideti un fisier nou. Salvati-l cu numele unu.asm. Il adaugati in proiectul current. Editati fisierul nou creat si introduceti in el urmatoarele instructiuni :

```
list p=16F84A ; tipul controlerului  
#include <p16F84A.inc>
```

```
__CONFIG _CP_OFF & _WDT_OFF & _PWRTE_OFF & _XT_OSC
```

```
var_a EQU 0CH ; asigneaza variabilei o adresa in memoria RAM  
var_b EQU 0DH ; asigneaza variabilei o adresa in memoria RAM  
var_c EQU 0BH ; asigneaza variabilei o adresa in memoria RAM
```

```
movlw 5; pun in acumulator valoarea 5  
movwf var_a; pun 5 in var_a  
movlw 4; pun in acumulator valoarea 4  
movwf var_b; pun 4 in var_b  
movlw 0; pun in acumulator valoarea 0  
movwf var_c; pun 0 in var_c
```

```
movf var_a,W; muta prima vairabila in registrul W  
xorwf var_b,W; compara var_b cu var_a care se afla in W  
btfsc STATUS,Z; testare Z.  
goto EQUAL; Daca Z=1, Egalitate, mergi la rutina de egalitate  
goto NOT_EQUAL; inegale. mergi la rutina pentru inegalitate
```

```
EQUAL  
; instructiuni  
movlw 1  
movwf var_c  
goto end_
```

```
NOT_EQUAL
; instructiuni
movlw 2
movwf var_c
goto end_

end_

end
```

Se testeaza fanionul de zero, Z. Operatia XOR afecteaza acest fanion. Daca cele doua numere sunt egale , operatia are ca rezultat trecerea fanionului Z in 0. Daca cele doua numere sunt egale se apeleaza procedura EQUAL unde variabilei c ii este atribuita valoarea 1. Daca cele doua numere nu sunt egale se apeleaza procedura NOT_EQUAL, unde variabilei c ii este atribuita valoarea 2 .