PROGRAMAREA ÎN LIMBAJ DE ASAMBLARE

GENGE BÉLA

Capitolul 4
Structuri de program.

Macro-uri

- Asamblorul permite definirea propriilor “instrucțiuni”.
- Noua definiție este plasată între MACRO-ENDM.
  - Echivalentul #define din C.

```
Bank2   MACRO
        MOVLW 2
        MOVWF BSR
        ENDM
```

- Utilizare:
  - Bank2

Macro-uri

- Sintaxa generală.

```
Etichetă MACRO [ARG1, ARG2, …]
         [INSTRUCȚIUNE 1]
         [INSTRUCȚIUNE 2]
         […]
         ENDM
```

- Exemplu: o nouă instrucțiune MOVLF
- Movlw MACRO Literal, Destinatie
  - MOVLW Literal
  - MOVWF Destinatie
  - ENDM

Macro-uri

- Problema cu codul anterior:
  - WREG și STATUS pot fi afectate.
  - Soluția: salvarea pe stivă.
- Movlw MACRO Literal, Destinatie
  - PUSH
  - MOVWF TOSL
  - MOVFF STATUS, TOSH
  - MOVLW Literal
  - MOVWF Destinatie
  - MOVF TOSL, W
  - MOVFF TOSH, STATUS
  - POP

Macro-uri

- Exemple de utilizare.
  - Movlf 0x21, VAR0
  - Movlf 0x57, 0x20
- Atenție!
  - “Apelul” macro-urilor e echivalent cu inserarea setului de instrucțiuni din corpul macro-ului.
- Exercițiu:
  - Scrieți un macro ce asigură o întârziere de 1us.
  - Scrieți un macro ce asigură o întârziere de 100us.

Macro-uri

- Problema cu etichete definite în Macro:
  - Inserarea multiplă a macroului duce la duplicate.
  - Soluția: cuvântul cheie *local* plasat înaintea etichetei face ca eticheta să fie redefinită pentru fiecare inserare de către asamblor.

Subrutine

- Pentru execuția aceluiași set de instrucțiuni fără a recurge la reinserarea codului se pot folosi **subrutine**.
  - Reprezintă o secvență de instrucțiuni cuprinse între o etichetă și o instrucțiune de revenire din subrutină.
- Se impune respectarea următoarelor:
  - Trebuie apelată prin CALL
  - Intrarea trebuie marcată printr-o etichetă.
  - Ieșirea trebuie să fie o instrucțiune de revenire.

Subrutine

- Sintaxa:

```
Etichetă:
          [INSTRUCȚIUNE 1]
          [INSTRUCȚIUNE 2]
          [...]
          RETURN sau RETLW sau RETFIE
```

- Subrutinele sunt în strânsă legătură cu stiva de apeluri! NU se recomandă algoritmi recursivi!
- Exercițiu:
  - Să se scrie o subrutină care inițializează pinii PORTD conform conexiunilor plăcuțelor de la laborator.

Subrutine

- O subrutină poate suprascrie o serie de regiștrii (WREG, STATUS, etc.)
- Salvarea poate fi realizată prin Fast Register Stack – o stivă cu unsingur nivel.
- Salvarea se realizează prin specificarea parametrului s al instrucțiunii CALL.

Subrutine

- Exemplu:
- Subrutina:
  - ....
  - RETURN FAST
- MAIN:
  - ....
  - CALL Subrutina, FAST

- Această metodă va salva și va restaura WREG, STATUS și BSR.

Tablouri

- Posibilități de implementare:
  - Instrucțiuni microcontroller.
  - Instrucțiuni tabelare.
  - Adresarea indirectă.

Tablouri – instrucțiuni uC

- Valorile sunt stocate în memoria program.
- Utilă în cazul în care sunt stocate valori constante.
- Metoda utilizată:
  - Instrucțiunea RETLW.
  - PC.

Tablouri – instrucțiuni uC

- Exemplu:
- TEMPF EQU 0X00

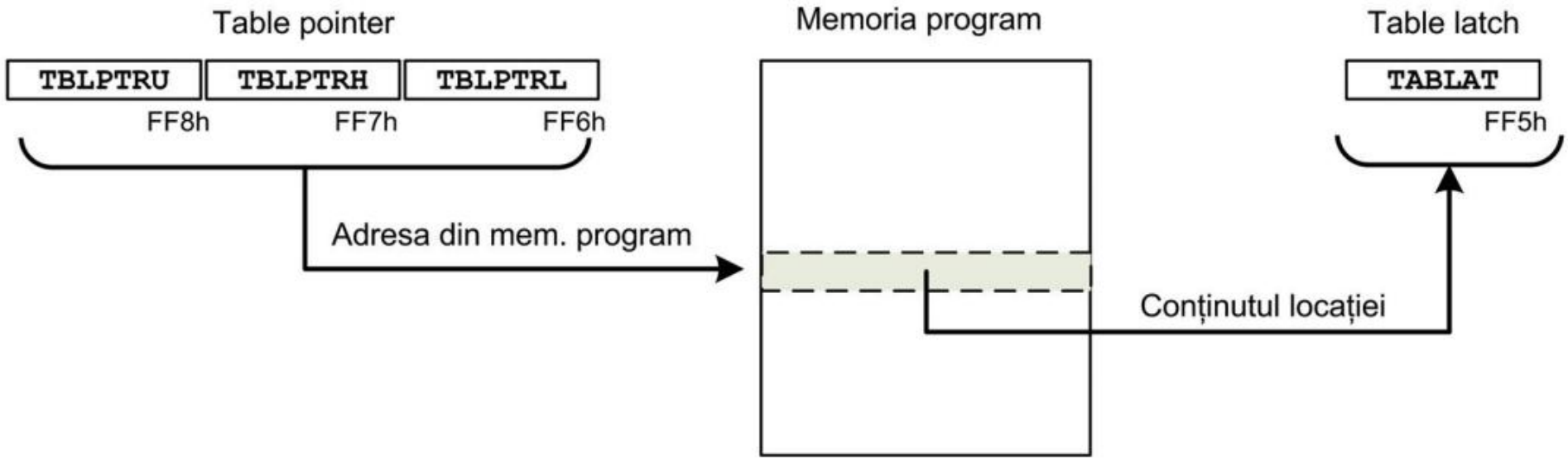
- Tablou:
  - ADDWF WREG, W
  - MOVFF PLC, TEMPF
  - ADDWF PLC, F; PLC=PLC+2*Index
  - RETLW d’10’
  - RETLW d’15’
  - RETLW d’37’

Tablouri – instrucțiuni dedicate

- Folosirea instrucțiunilor tabelare.
- Permit citirea memoriei program octet cu octet și scrierea acesteia prin blocuri de minim 32 de octeți.
- Citirea tabelară: TBLRD
- Scrierea tabelară: TBLWT
- Citirea/scrierea mută datele prin intermediul registrului TABLAT (SFR din RAM).
- Registrul de adresare este TBLPTR (pe 21 de biți): TBLPTRU:TBLPTRH:TBLPTRL

Tablouri – instrucțiuni dedicate

- Regiștrii de adresare sunt utilizați la citirea/scrierea memoriei program cu sinstrucțiunile TBLRD și TBLWT.



Tablouri – instrucțiuni dedicate

- Citirea/scrierea se recomandă a fi realizate prin instrucțiuni care modifică toți cei 3 regiștrii ai TBLPTR.
  - TBLRD* TBLPTR->TABLAT
  - TBLRD*+ TBLPTR->TABLAT, TBLPTR++
  - TBLRD*- TBLPTR->TABLAT, TBLPTR--
  - TBLRD+* TBLPTR-- , TBLPTR->TABLAT

- .. similar pentru TBLWT – a se vedea foaia de catalog.

Tablouri – instrucțiuni dedicate

- Exemplu:
- Tablou:
  - db 0x15, 0x23, 0x17, 0x20, …
- Inițializarea adresării:
- Init:
  - MOVLW low Tablou
  - MOVWF TBLPTRL
  - MOVLW high Tablou
  - MOVWF TBLPTRH
  - MOVLW upper Tablou
  - MOVWF TBLPTRU
  - RETURN

Tablouri – instrucțiuni dedicate

- Apelul subrutinei de inițializare:
  - CALL Init
- Accesarea elementelor (incremental):
  - TBLRD*+
  - MOVF TABLAT, W
- Accesarea elementelor (indexat):
  - MOVLW d’3’
  - ADDWF TBLPTRL, F
  - MOVLW d’0’
  - ADDWFC TBLTRH, F
  - MOVLW d’0’
  - ADDWFC TBLTRU, F
  - TBLRD*
  - MOVF TABLAT, W

Tablouri – adresarea indirectă

- Datele sunt stocate în memoria RAM.
- Permite implementarea tablourilor de variabile -> pot fi modificate pe parcursul execuției.
- Pasul 1: stocarea valorilor în memoria RAM.
- Exemplu: scrierea unei subrutine pentru stocarea valorilor în BANK2.

Tablouri – adresarea indirectă

- Datele sunt stocate în memoria RAM.
- Permite implementarea tablourilor de variabile -> pot fi modificate pe parcursul execuției.
- Pasul 1: stocarea valorilor în memoria RAM.
- Stocare:
  - LFSR FSR0, 0X200
  - MOVLW d’101’
  - MOVWF POSTINC0
  - MOVLW d’87’
  - MOVWF POSTINC0
  - ….
  - RETURN

Tablouri – adresarea indirectă

- Pasul 2: returnare valoare.

Tablouri – adresarea indirectă

- Pasul 2: returnare valoare.
- Returnare_valoare:
  - LFSR FSR0, 0X200
  - ADDWF FSR0L, F
  - MOVLW d’0’
  - ADDWFC FSR0H, F
  - MOVF INDF0, W
  - RETURN
- Apel:
  - CALL Stocare
  - MOVLW d’3’
  - CALL Returnare_valoare