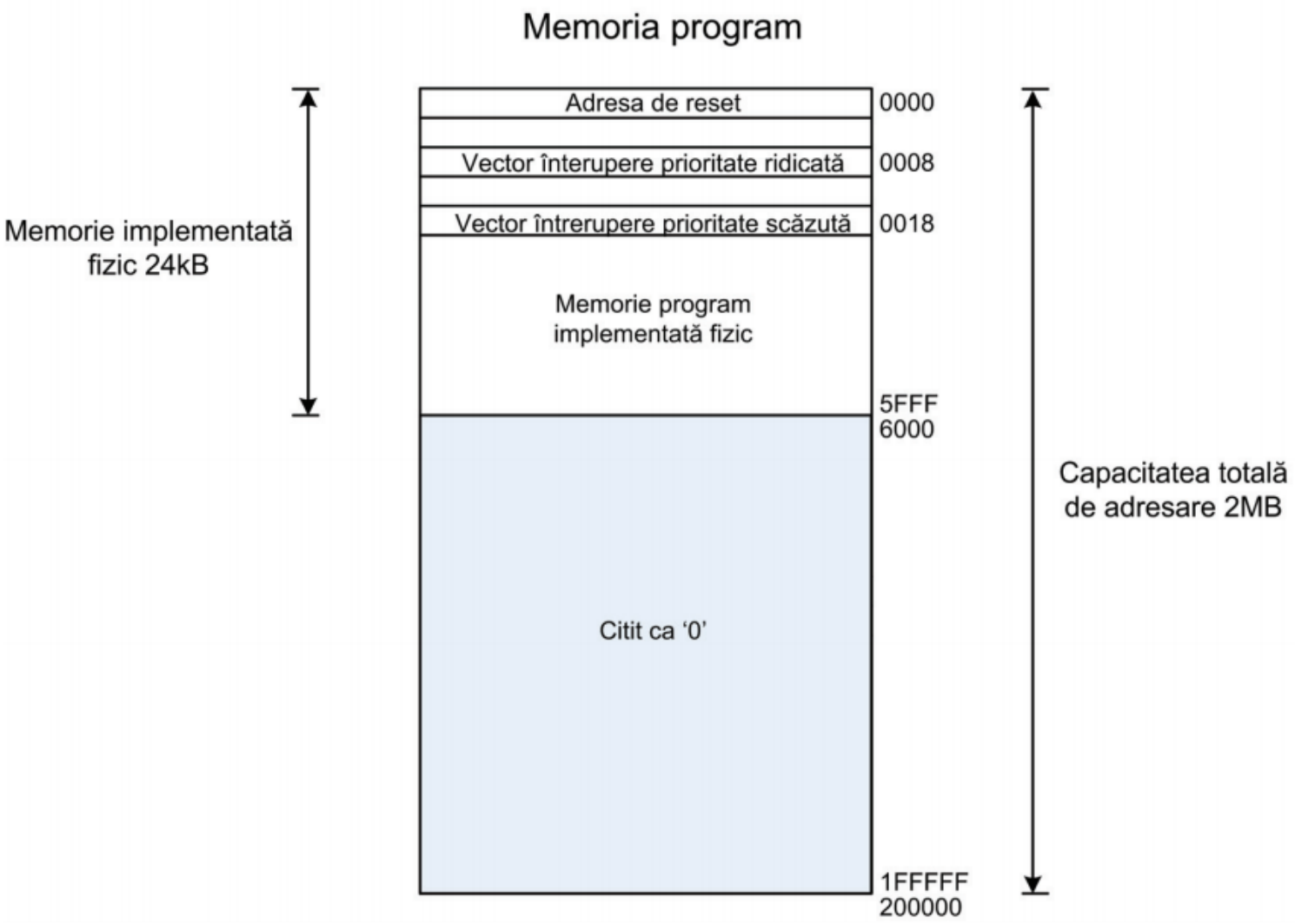
PROGRAMAREA ÎN LIMBAJ DE ASAMBLARE

GENGE BÉLA

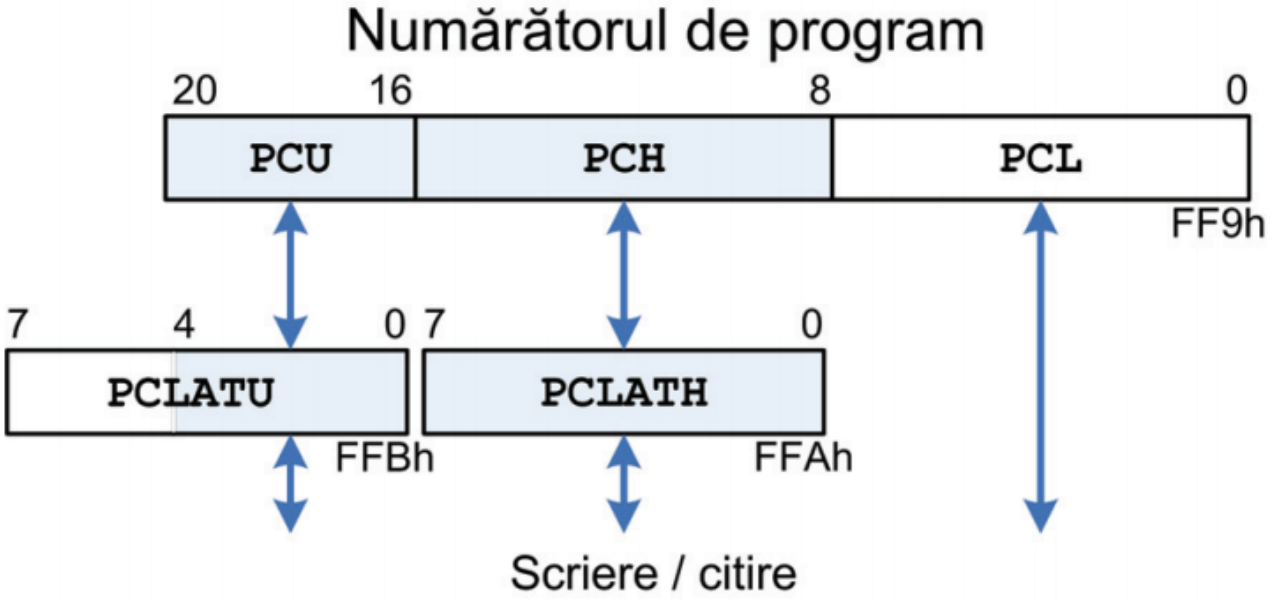
Capitolul 3
Nume simbolice. Adresarea indirectă.

Organizarea memoriei



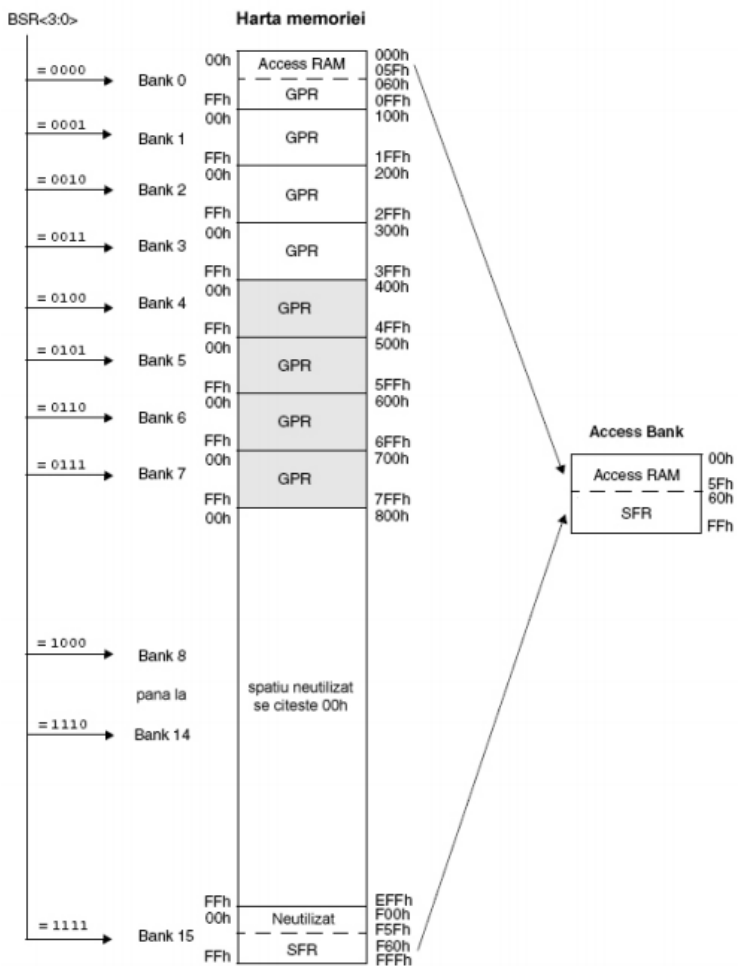
Numărătorul de program

- Scrierea unei valori în registrul PCL (ex. MOVWF PCL) transferă simultan și conținutul regiștrilor tampon.
- La citirea registrului PCL (MOVF PCL,W) se transferă simultan și regiștrii PCU:PCH în regiștrii tampon.



Memoria de date RAM

- Utilizarea Bank-urilor: mem. de date este împărțită în Bank-uri.
  - Nu e necesară specificarea a 12 biți ci doar 8 biți.
- Total de 16 Bank-uri, 8 bank-uri implementate de PIC18F4455.
- Fiecare Bank conține 256 de regiștrii.



Memoria de date RAM

- BSR: Bank Select Register.
  - Registru de selectare a bank-ului.
- Pentru modificarea conținutului registrului este disponibilă instrucțiunea MOVLB (Move Literal to BSR).
- Selectarea Bank-ului 2:
  - MOVLB 0X02
- sau:
  - MOVLW 0X02
  - MOVWF BSR

Exercițiu

- Presupunem:
  - Buton: conectat pe PORTB5.
  - Releu: conectat pe PORTC6
- Se cere:
  - Să se scrie o aplicație pentru PIC18F4455 care la fiecare apăsare a butonului închide și deschide consecutiv releul de două ori (timpul de menținere pe închis/deschis se va ignora).

Variabile și nume simbolice

- Noțiunea de “variabilă” din limbaje de programare de nivel înalt.
- La uC “variabila” poate fi implementată prin orice registru (GPR sau SFR).
- Utilizarea variabilei se poate realiza prin:
  - Adresa fizică.
  - Atașarea unui nume simbolic.

Variabile și nume simbolice

- Noțiunea de “variabilă” din limbaje de programare de nivel înalt.
- La uC “variabila” poate fi implementată prin orice registru (GPR sau SFR).
- Utilizarea variabilei se poate realiza prin:
  - Adresa fizică.
  - Atașarea unui nume simbolic.
- Exemplu: stocare 0x7A în TRISB de la adresa 0x93.

Variabile și nume simbolice

- Definirea de nume simbolice:
  - EQU: EQUivalent to
- Atenție!
  - Denumirile simbolice pot fi utilizate ca și constante sau ca și regiștrii.
- Definirea unui număr mare de nume simbolice:

```
CBLOCK      0X00
VAR0
VAR1
...
ENDC
```

Problemă

- Până în momentul de față s-a specificat explicit adresa registrului în cadrul instrucțiunii.

- Fiecare instrucțiune a inclus și adresa registrului:
  - ADDWF 0X3C, W, ACCESS
  - ADDWF 0X3C, F, ACCESS

Problemă

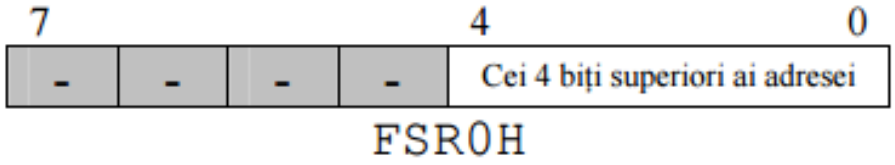
- Se cere:
  - Ștergerea (valoare = 0) primilor 32 de regiștrii din Bank0.
- Soluția?

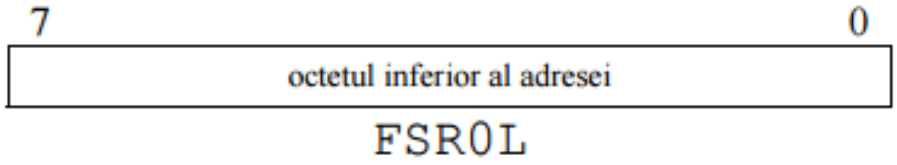
Soluție: adresarea indirectă

- Soluția: regiștrii de adresare (pointeri).
  - Avantaj: modificare adresă din program, în timpul execuției.
- Echivalentul din C:
  - Variabila pointer.
  - Operatorul * pentru accesarea conținutului.
- Pentru stocarea adresei avem 3 perechi de regiștrii FSR (**File Select Registers**):
  - FSR0.
  - FSR1.
  - FSR2.

Regiștrii de adresare

- Regiștrii FSRn sunt folosiți pentru a stoca adresele absolute (pe 12 biți).
- FSRn sunt formați din doi regiștrii:
  - FSRnH.
  - FSRnL.

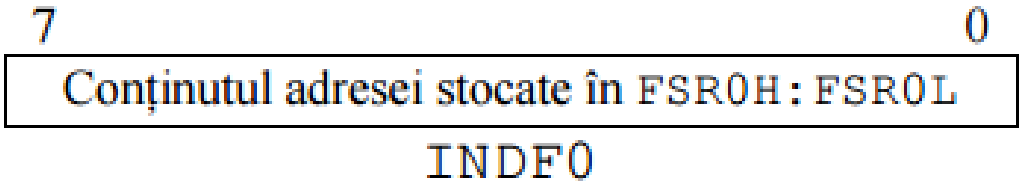




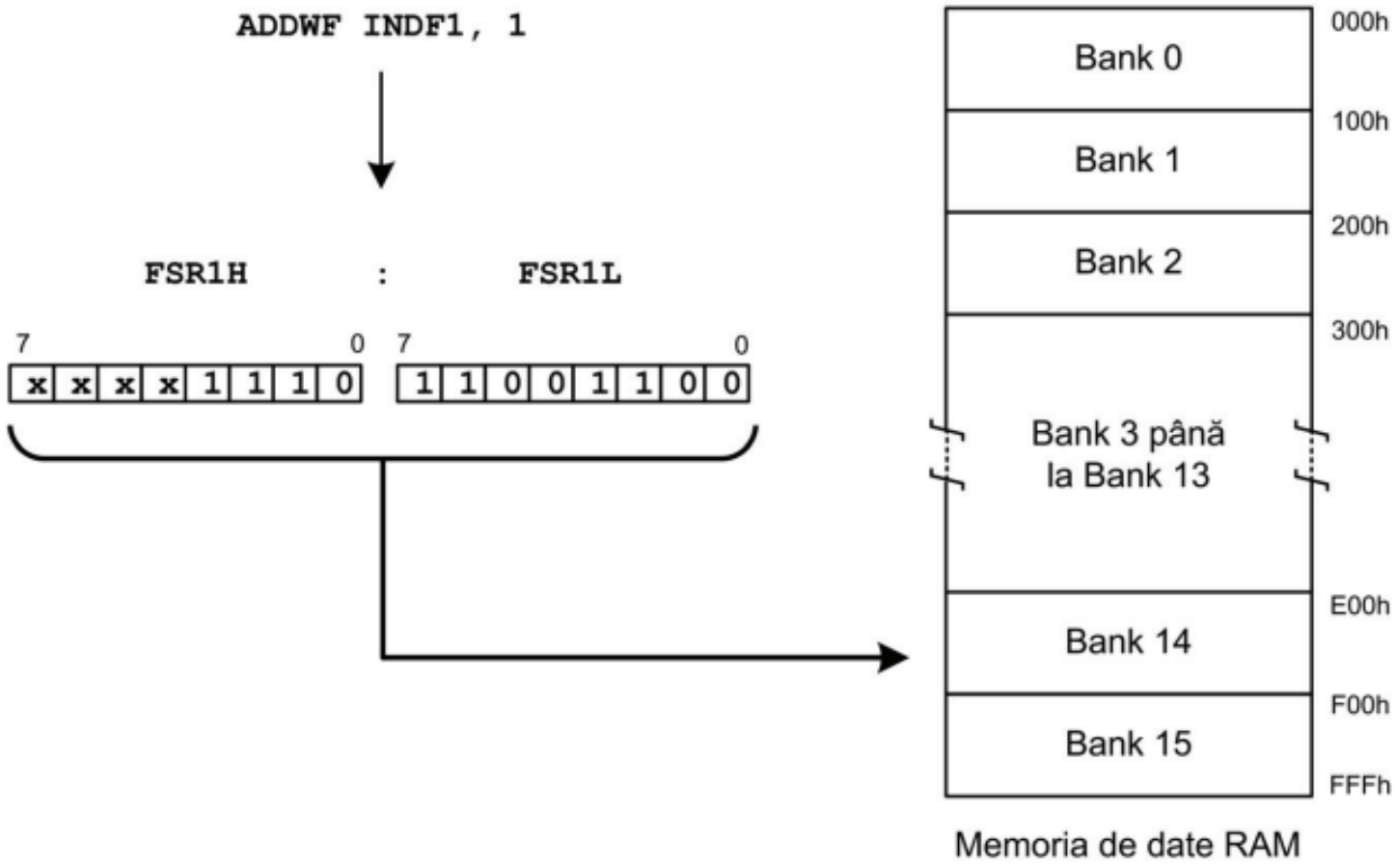
- Accesibili prin instrucțiuni uzuale.
- Instrucțiunea specială:
  - LFSR FSR0, 0X200

Accesarea conținutului

- Regiștrii FSRn sunt folosiți pentru a stoca adresele absolute (pe 12 biți).
- Pentru fiecare FSRn (FSR0, FSR1, FSR2) există perechea INDFn (INDF0, INDF1, INDF2) – Indirect File Operand.



Accesarea conținutului



Exemple

- Incrementarea valorii unui registru de la adresa 0x200.
  - LFSR FSR0, 0X200
  - INCF INDF0, F
- Mutarea conținutului registrului de la adresa 0x200 în WREG:
  - LFSR FSR0, 0X200
  - MOVFF INDF0, WREG

Exemple

- Exerciții.
  - Ștergerea regiștrilor de la adresele 0x00 la 0xFA din BANK1.
  - Inițializarea regiștrilor de la adresele 0x10 la 0x55 din BANK4 cu valori decrementate succesiv începând de la 255 (ex: 255, 254, 253, ...).
  - Inițializarea regiștrilor de la adresele 0x10 la 0x20 din BANK4 cu valori decrementate cu 10 începând de la 255 (ex: 255, 245, 235, ...).

Adresarea indexată

- Incrementarea FSRnL prin instrucțiuni uzuale nu duce la incrementarea automată a FSRnH în cazul unui overflow.
- Soluția:
  - Fiecare pereche FSRn și INDFn este prevăzută cu patru operanzi adiționali.
  - Sunt de fapt regiștrii virtuali care prin circuite integrate incrementează/decrementează automat regiștrii de adresă.

Accesare conținut

- INDFn – accesează conținutul de la adresa FSRn.
- POSTDECn – accesează conținutul de la adresa FSRn și decrementează FSRn cu 1.
- POSTINCn – accesează conținutul de la adresa FSRn și incremenează FSRn cu 1.
- PREINCn – incrementează FSRn cu 1 și accesează conținutul de la noua adresă.
- PLUSWn – adună valoarea (de la -127 la 128) WREG la FSRn, apoi accesează valoarea de la noua adresă.

Accesare conținut – exemple

- LFSR FSR0, 0X100
- CLRF POSTINC0 ; FSR0++
- CLRF POSTDEC0 ; FSR0--
- CLRF PREINC0 ; ++FSR0
- MOVLW D’5’
- CLRF PLUSW0 ; 5+FSR0

Exercițiu

- Se consideră un afișaj de 7 segmente legat pe PORTB.
- Să se implementeze o aplicație pe PIC18F4455 care afișează numerele de la 0 la 9. NU se ia în calcul timpul de execuție.
- Cerințe:
  - Să se identifice constantele numerice pentru afișarea pe 7 segmente.
  - Să se stocheze în BANK2 începând cu adresa 0x00 constantele utilizând adresarea indirectă.
  - Să se acceseze valorile utilizând adresarea indirectă.

Exercițiu

- Detalii tehnice:
  - LED-urile de pe afișaj sunt active ‘LOW’ (scriem 0 pentru a le aprinde).
  - Schema de legare: