PROGRAMAREA ÎN LIMBAJ DE ASAMBLARE

GENGE BÉLA

Capitolul 2
Clasificarea instrucțiunilor

Tipuri de instrucțiuni

- Instrucțiuni matematice pe octet.
- Instrucțiuni logice pe octet.
- Instrucțiuni matematice și logice cu constante.
- Instrucțiuni logice pe bit.
- Instrucțiuni de comparație pe bit.
- Instrucțiuni de comparație pe octet.
- Instrucțiuni de mutare.
- Instrucțiuni de control.
- Alte instrucțiuni.

Instrucțiuni matematice pe octet

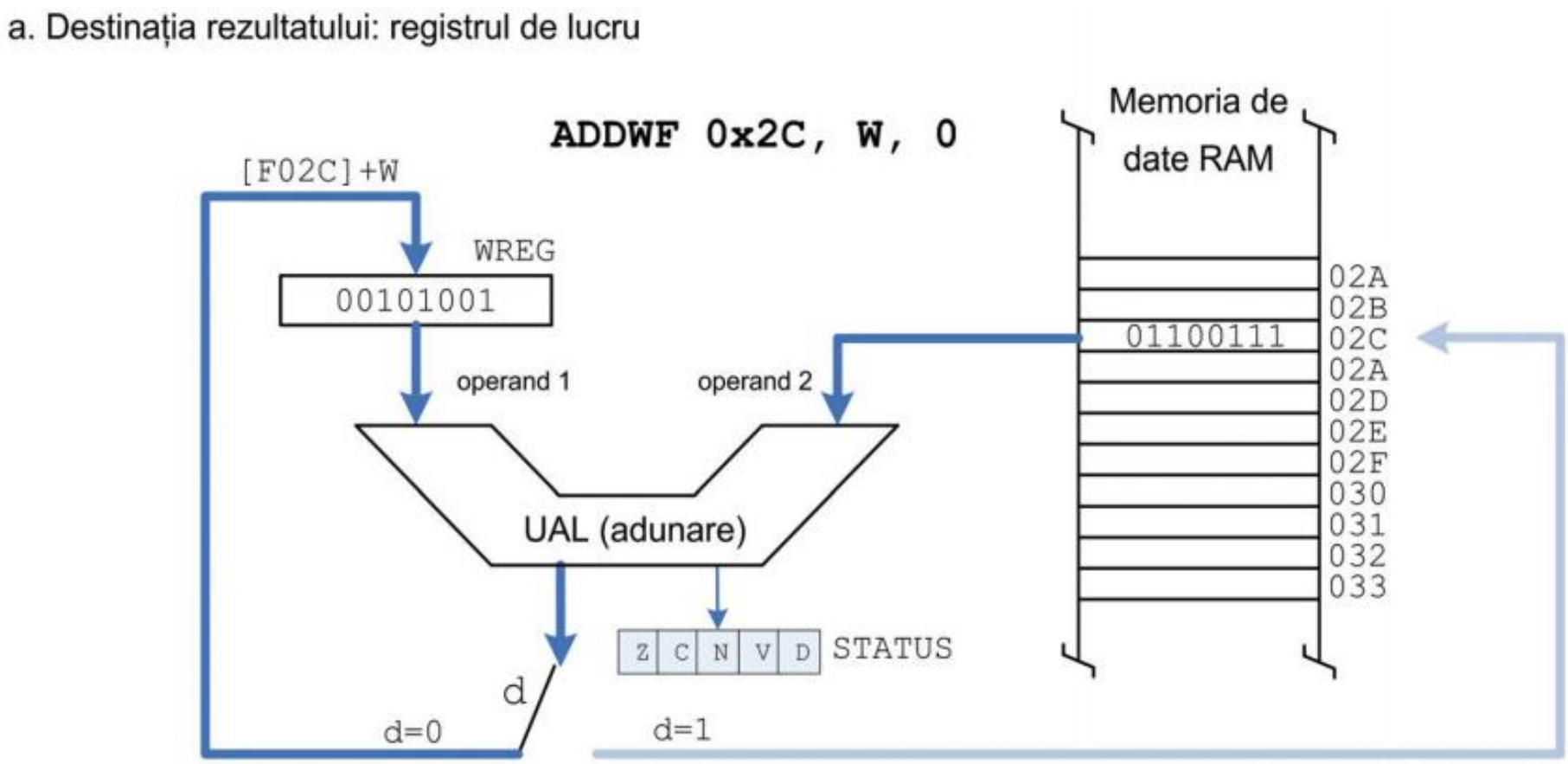
- Instrucțiuni ce realizează operații matematice asupra datelor pe 8 biți.
- Exemplu de instrucțiune:

```
ADDWF   f, d, a
```

- Adaugă conținutul WREG la registrul de la adresa f.
- Rezultatul îl transferă în WREG (d=0), sau la adresa f (d=1), înlocuind valoarea existentă.
- Adresa registrului f se selectează din Bank-ul de acces dacă a=0 și dintr-un alt Bank (0-7) dacă a=1 (prin regiștrii BSR).

Instrucțiuni matematice pe octet

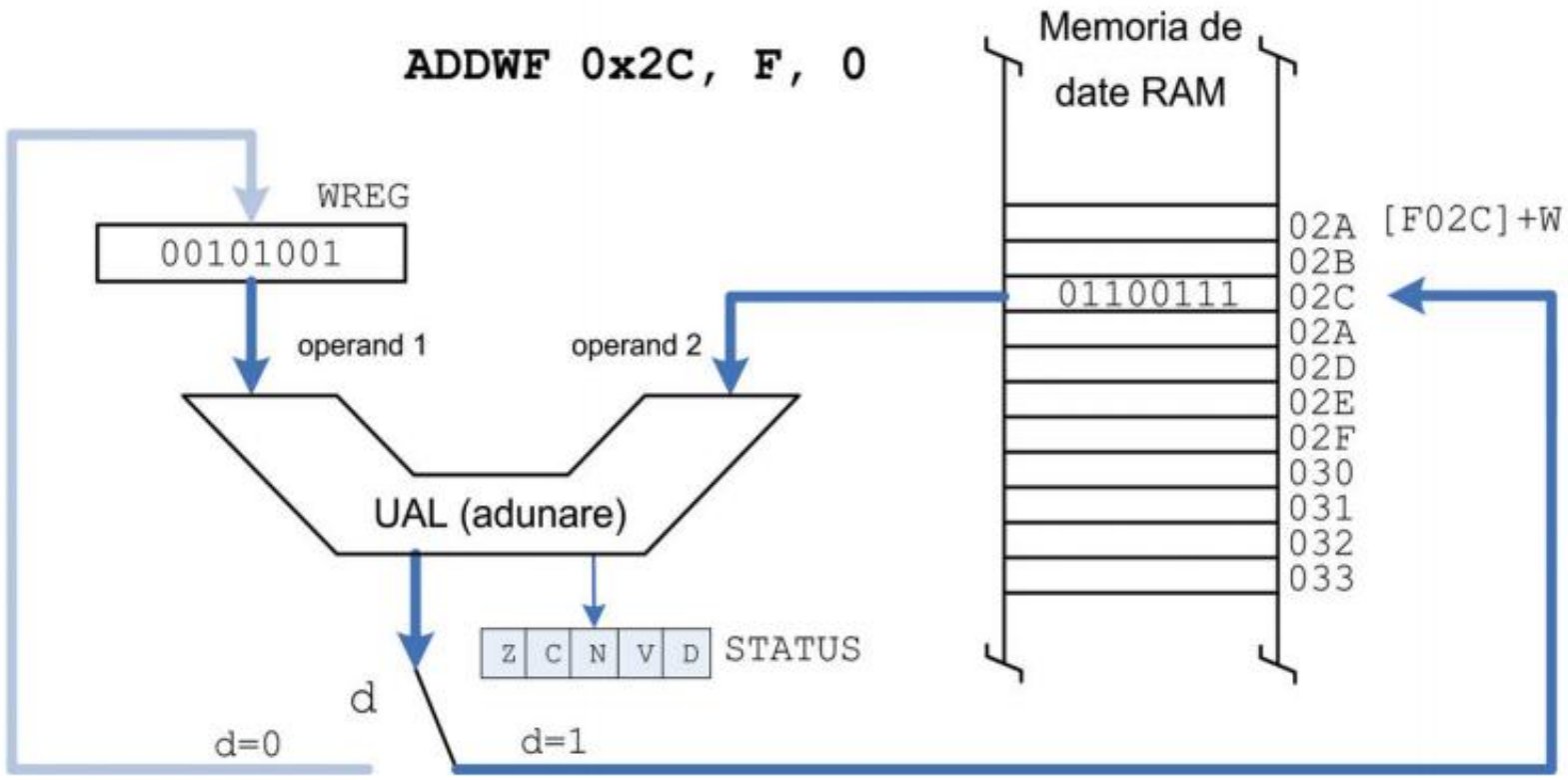
- Exemplu: înaintea execuției WREG=b'00101001'=h'29'=d'41'.



Instrucțiuni matematice pe octet

- Exemplu: înaintea execuției WREG=b'00101001'=h'29'=d'41'.

b. Destinația rezultatului: registrul f



Instrucțiuni matematice pe octet

Instrucțiuni		Descriere	Cicluri instr.
ADDWF	f,d,a	Adună WREG cu f	1
ADDWFC	f,d,a	Adună WREG cu f şi cu bitul de transport	1
SUBWF	f,d,a	Scade WREG din f	1
SUBWFB	f,d,a	Scade f din WREG cu bitul de împrunut	1
SUBFWB	f,d,a	Scade WREG din f cu bitul de împrumut	1
INCF	f,d,a	Incrementează f	1
DECF	f,d,a	Decrementează f	1
MULWF	f,a	Înmulţeşte f cu WREG[1]	1
NEGF	f,a	Negare f	1

Instrucțiuni logice pe octet

- Instrucțiuni ce realizează operații logice pe 8 biți:
  - ȘI logic, SAU logic, SAU-EXCLUSIV, rotire pe biți, etc.
- Parametrii acestor instrucțiuni sunt aceiași ca și în cazul anterior (f, d, a).

Instrucțiuni logice pe octet

Instrucțiuni		Descriere	Cicluri instr.
ANDWF	f,d,a	ŞI logic între WREG şi f	1
IORWF	f,d,a	SAU logic între WREG şi f	1
XORWF	f,d,a	SAU-EXCLUSIV între WREG şi f	1
COMF	f,d,a	Complement f	1
CLRF	f,a	Resetare biți din registrul f	1
SETF	f,a	Setare biți din registrul f	1
SWAPF	f,d,a	Interschimbare semiocteți din registrul f	1
RLCF	f,d,a	Rotire f la stânga cu bit de transport	1
RLNCF	f,d,a	Rotire f la stânga fără bit de transport	1
RRCF	f,d,a	Rotire f la dreapta cu bit de transport	1
RRNCF	f,d,a	Rotire f la dreapta fără bit de transport	1

- Caz special este cel al instrucțiunilor ce utilizează bitul de transport (en: Carry).

Instrucțiuni logice pe octet

- Exemplu: rotirea spre stânga cu bit de transport.

RLCF f, d, a

- Exemplu: rotirea spre stânga fără bit de transport.

RLNCF f, d, a

Instrucțiuni matematice și logice cu constante

- Cu excepția MULLW, toate instrucțiunile vor avea formatul:
  - Un operand WREG.
  - Un operand o constantă (literal).

Instrucțiuni		Descriere	Cicluri instr.
ADDLW	k	Adună WREG cu literal	1
SUBLW	k	Scade WREG din literal	1
MULLW	k	Înmulțește WREG cu literal	1
ANDLW	k	ŞI logic între WREG şi literal	1
IORLW	k	SAU logic între WREG şi literal	1
XORLW	k	SAU-EXCLUSIV între WREG şi literal	1

- În cazul MULLW rezultatul este stocat în regiștrii PRODH:PRODL.

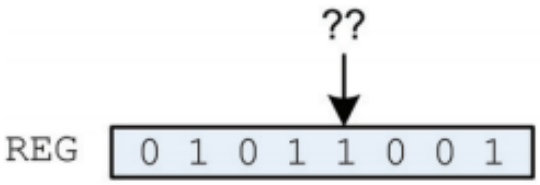
Instrucțiuni logice pe bit

- Instrucțiuni ce modifică starea unui singur bit.
- f – registrul asupra căruia se aplică.
- b – bitul modificat (0-7).
- a – utilizarea Access Bank-ului (0/1).

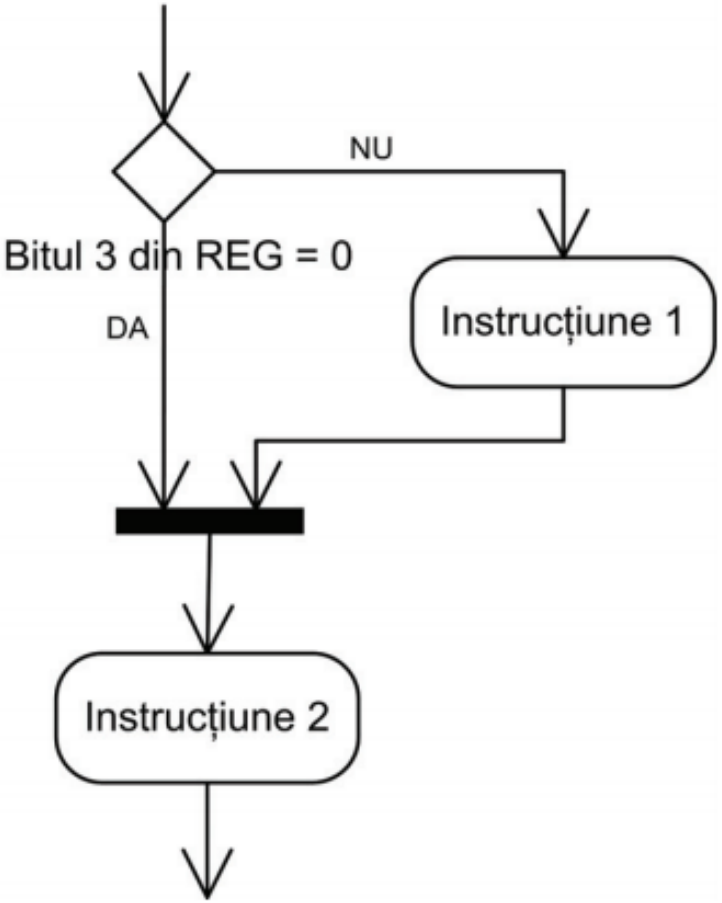
Instrucțiuni		Descriere	Cicluri instr.
BCF	f,b,a	Resetează bitul b din registrul f	1
BSF	f,b,a	Setează bitul b din registrul f	1
BTG	f,b,a	Neagă starea bitului b din registrul f	1

- Exemplu:
  - BCF FREG, 5, 0
  - Valoarea FREG înainte: 01110100
  - Valoarea FREG după: 01**0**10100

Instrucțiuni de comparație pe bit




```
BTFSC  REG, 3, 0
Instrucțiune 1
Instrucțiune 2
```



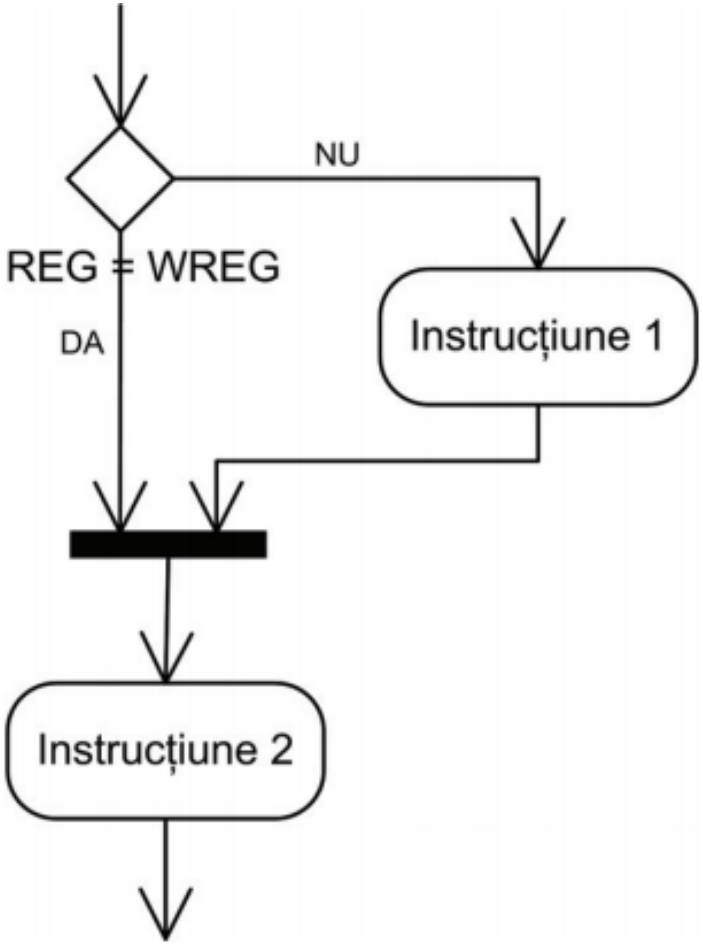
Instrucțiuni logice pe bit

- Exemplu implementare “if-else”:

```
3           BTFSS   REG, 1, 0
4           GOTO    FALSE
5   TRUE:
6           ; Instructiuni pentru bit = 1
7           GOTO    NEXT
8   FALSE:
9           ; Instructiuni pentru bit = 0
10  NEXT:
11          ; Instructiuni executate dupa testare
```

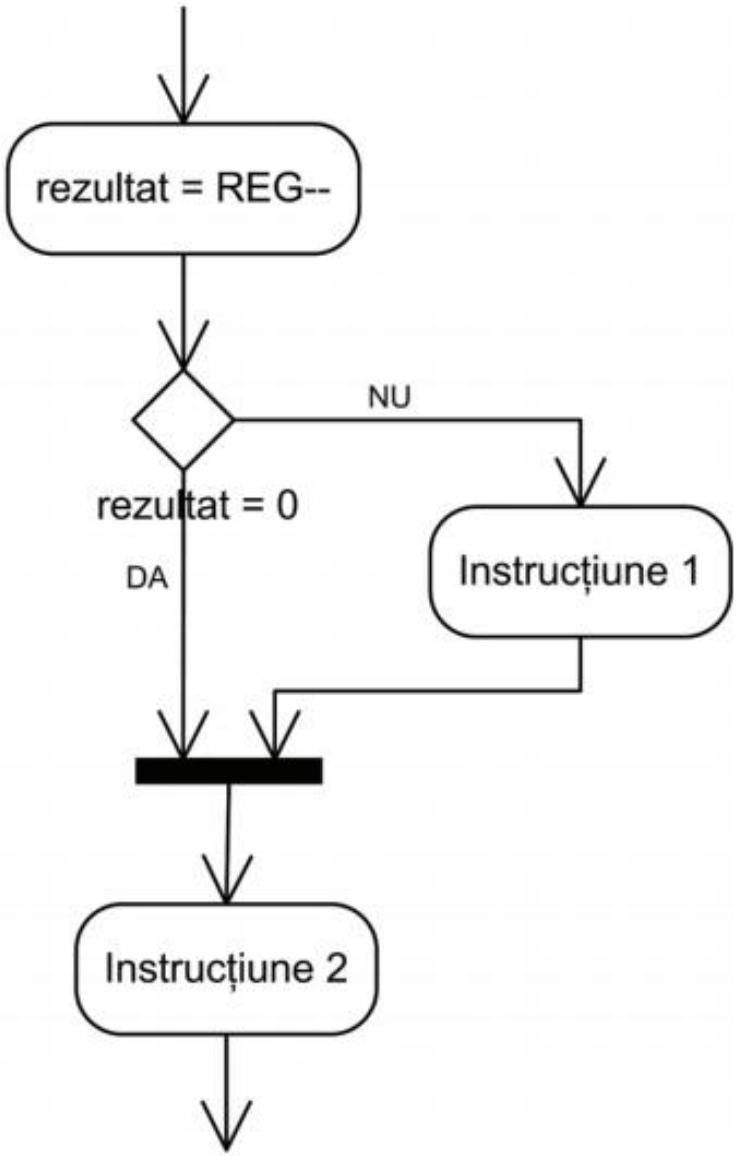
Instrucțiuni de comparație pe bit

```
CPFSEQ  REG,0
Instrucțiune 1
Instrucțiune 2
```



Instrucțiuni de comparație pe bit

```
DECFSZ  REG, F, 0
Instrucțiune 1
Instrucțiune 2
```



Instrucțiuni de comparație pe bit

```
        MOVLW   D'100'
        MOVWF   REG
Bucla_for:
        ; Instructiuni
        DECFSZ  REG, F, 0
        GOTO    Bucla_for
```

Instrucțiuni de comparație pe bit

Instrucțiuni		Descriere	Cicluri instr.
CPFSEQ	f,a	Compară f cu WREG, salt dacă f=WREG	1 (2 sau 3)
CPFSGT	f,a	Compară f cu WREG, salt dacă f>WREG	1 (2 sau 3)
CPFSLT	f,a	Compară f cu WREG, salt dacă f<WREG	1 (2 sau 3)
TSTFSZ	f,a	Testează f, salt dacă f=0	1 (2 sau 3)
INCFSZ	f,d,a	Incrementează f, salt dacă rezultatul = 0	1 (2 sau 3)
INCFSNZ	f,d,a	Incrementează f, salt dacă rezultatul ≠ 0	1 (2 sau 3)
DECFSZ	f,d,a	Decrementează f, salt dacă rezultatul = 0	1 (2 sau 3)
DECFSNZ	f,d,a	Decrementează f, salt dacă rezultatul ≠ 0	1 (2 sau 3)

Instrucțiuni de mutare

- Instrucțiuni ce asigură mutarea datelor dintr-o locație în alta.

Instrucțiuni		Descriere	Cicluri instr.
MOVWF	f,a	Mută WREG în f	1
MOVF	f,d,a	Mută f	1
MOVFF	f_s,f_d	Mută f_s(sursa) în f_d(destinație)	2
MOVLW	k	Mută literal în WREG	1
MOVLB	k	Mută literal (4 biți) în BSR	1
LFSR	f,k	Mută literal (12 biți) în FSR(f)	2

Instrucțiuni de control

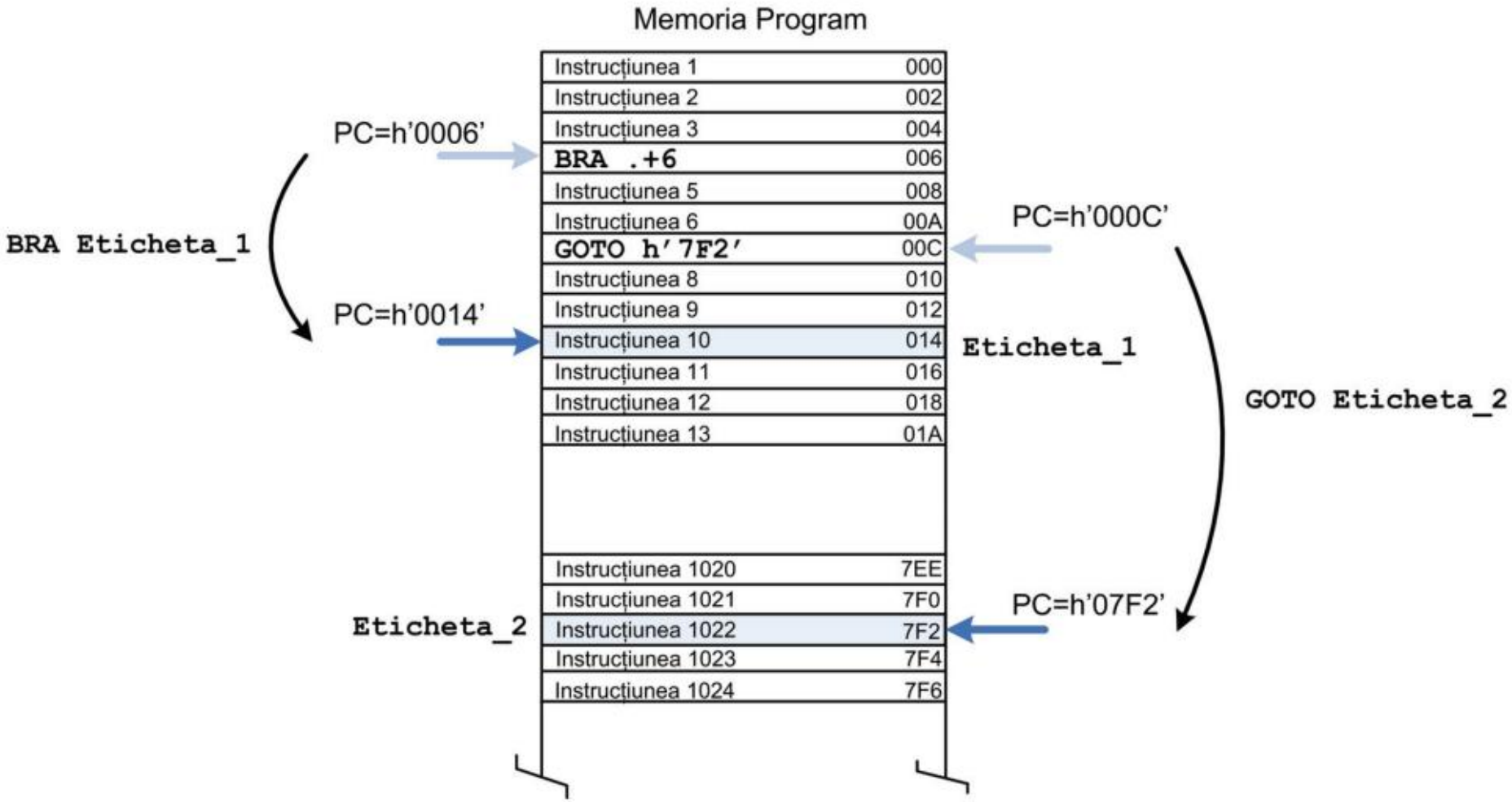
- Instrucțiuni ce controlează fluxul de execuție a instrucțiunilor.
  - Afectează valoarea numărătorului de program.

Instrucțiuni		Descriere	Cicluri instr.
BC	n	Salt dacă bitul de transport (carry) este 1	1 (2)
BN	n	Salt dacă bitul negativ este 1	1 (2)
BOV	n	Salt dacă bitul de depășire (overflow) este 1	1 (2)
BZ	n	Salt dacă bitul zero este 1	1 (2)
BNC	n	Salt dacă bitul de transport (carry) este 0	1 (2)
BNN	n	Salt dacă bitul negativ este 0	1 (2)
BNOV	n	Salt dacă bitul de depășire (overflow) este 0	1 (2)
BNZ	n	Salt dacă bitul zero este 0	1 (2)
BRA	n	Salt necondiționat	2
GOTO	n	Salt la adresa / etichetă	2
PUSH		Pune în stivă (Salvează PC în stivă)	1
POP		Scoate din stivă (Reface PC)	1
CALL	n,s	Apel de subrutină	2
RETURN	s	Revenire din subrutină	2
RETLW	k	Revenire din subrutină cu literal în WREG	2
RETFIE	s	Revenire din rutina de tratare a întreruperii	2
NOP		Nici o operație	1

Instrucțiuni de control

- Diferența dintre instrucțiunile BRA și GOTO:
  - Ambele realizează un salt necondiționat.
  - BRA: salt la o adresă relativă față de poziția curentă.
  - GOTO: salt la o adresă absolută.

Instrucțiuni de control



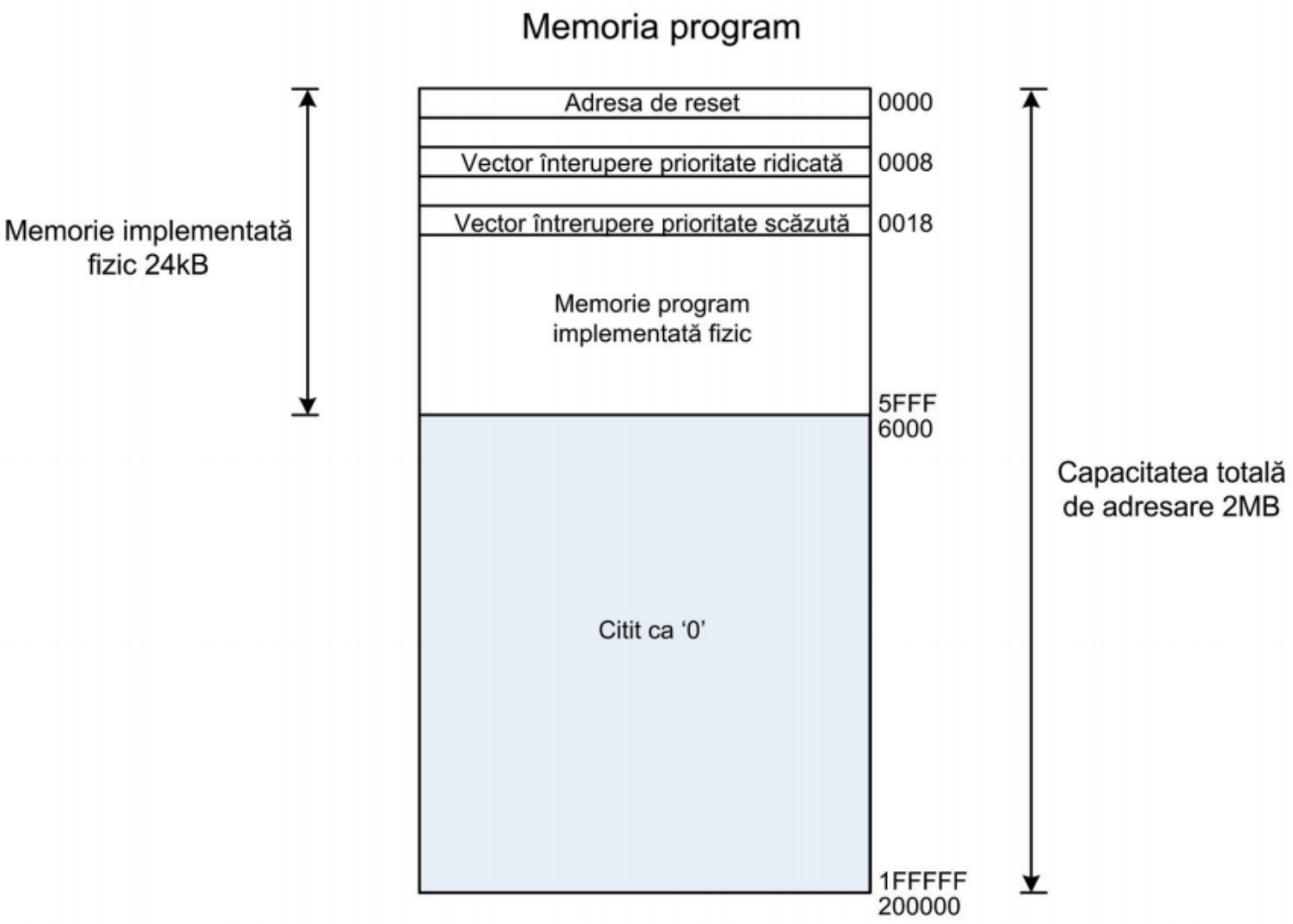
Organizarea memoriei

- Capacitatea de adresare: 2MB
  - Magistrală de adrese de 21 biți (2MByte=2^{21}Byte).
  - PC pe 21 biți.
- Din cei 2MB în cazul PIC18F4455 sunt implementați fizic doar 24KB.
- Memoria program e implementată sub forma unei memorii Flash.
  - Conform foii de catalog memoria poate fi ștearsă/rescrisă de aprox. 100.000 ori.

Organizarea memoriei

- Accesarea unei locații cuprinse între limita superioară implementată și limita de adresare va returna valoarea **zero** (instrucțiune NOP – No OPeration).
- Vectori de întrerupere:
  - 0008h.
  - 0018h.
- Adresa de reset: 0000h.

Organizarea memoriei

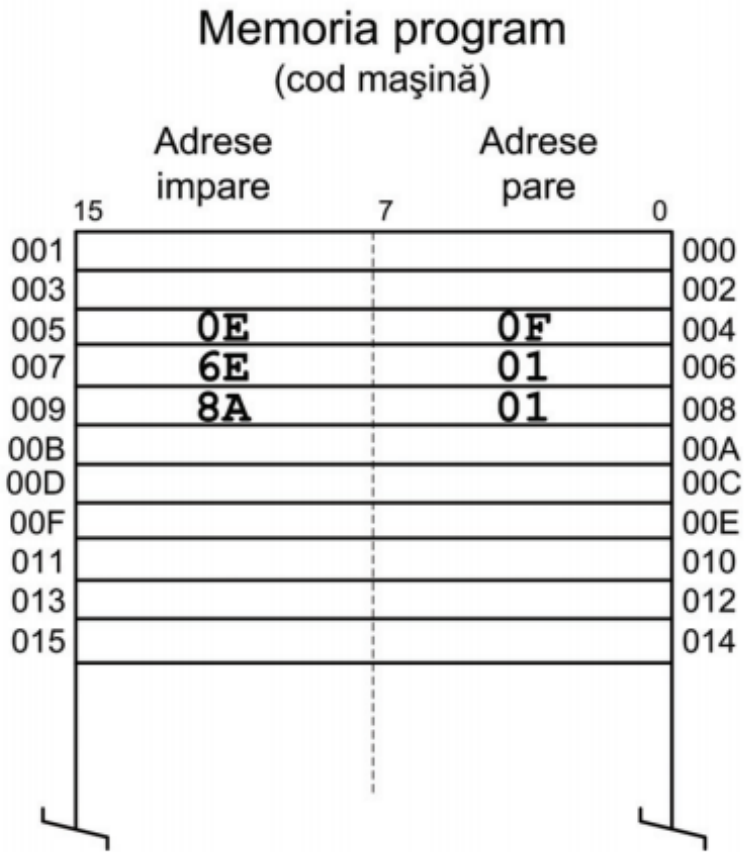


Organizarea memoriei

- Memoria poate fi scrisă/citită cu instrucțiuni TBLWT/TBLRD
- Permite implementarea aplicațiilor **bootloader.**

Instrucțiuni:

```
MOVLW d'15'
MOVWF 0x01,0
BSF   0x01,5,0
```



Numărătorul de program

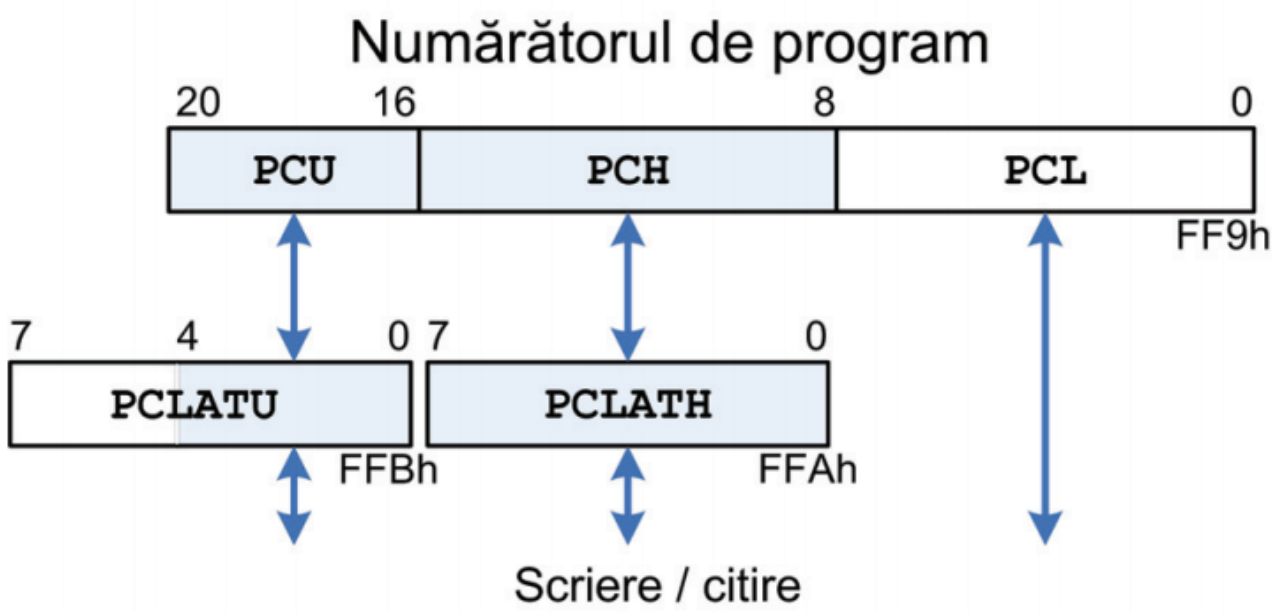
- PC (Program Counter): conține adresa următoarei instrucțiuni.
- PC este pe 21 biți împărțiși pe 3 regiștrii:
  - PLC (Program Counter Low): PC:<7-0>.
  - PCH (Program Counter High): PC:<15-8>.
  - PCU (Program Counter Upper): PC:<20-16>.
- PC funcționează ca un numărător binar fiind incrementat:
  - De două ori pentru instrucțiuni simple.
  - De patru ori pentru instrucțiuni extinse (GOTO).

Numărătorul de program

- Modificarea PC poate fi realizată din program.
- Problema:
  - Modificarea PC (21biți) se poate realiza prin modificarea a 8 biți deodată cu instrucțiuni care în sine vor duce la incrementarea PC.
  - Se pierde valoarea PC deoarece nu se pot accesa simultan toți cei 21 de biți.
- Soluția:
  - Doar PCL poate fi citit/scris direct.
  - Pentru PCH și PCU s-au introdus doi regiștrii tampon PCLATH și PCLATU.

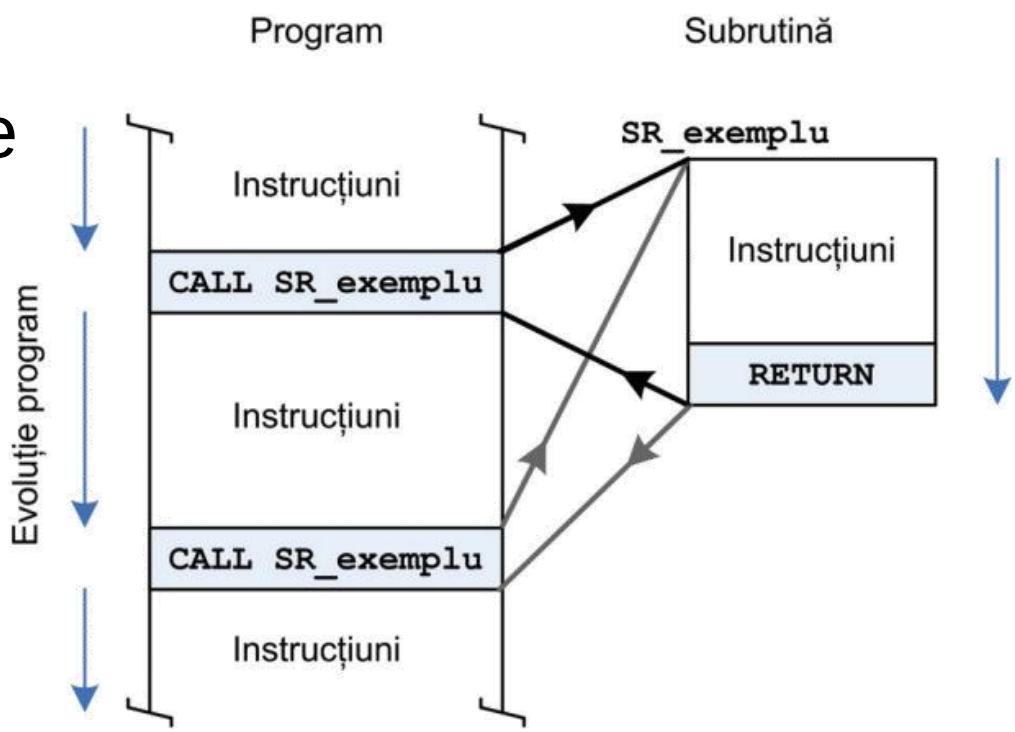
Numărătorul de program

- 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.



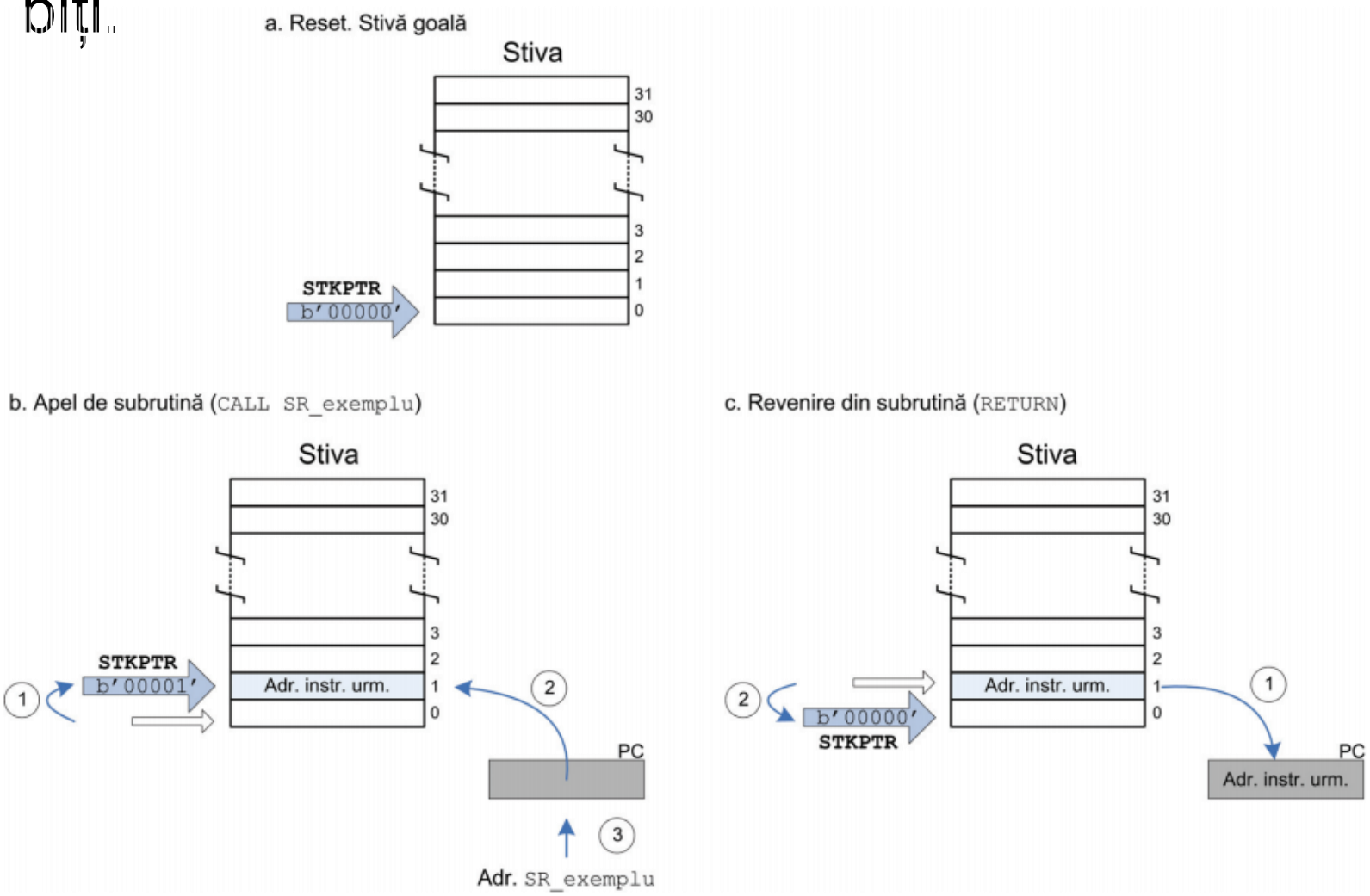
Stiva adreselor de revenire

- Apelul unei subrutine înseamnă:
  - Salvarea adresei de revenire.
  - Încărcarea adresei subrutinei în PC.
  - Execuția subrutinei.
  - Încărcarea în PC a adresei de revenire.



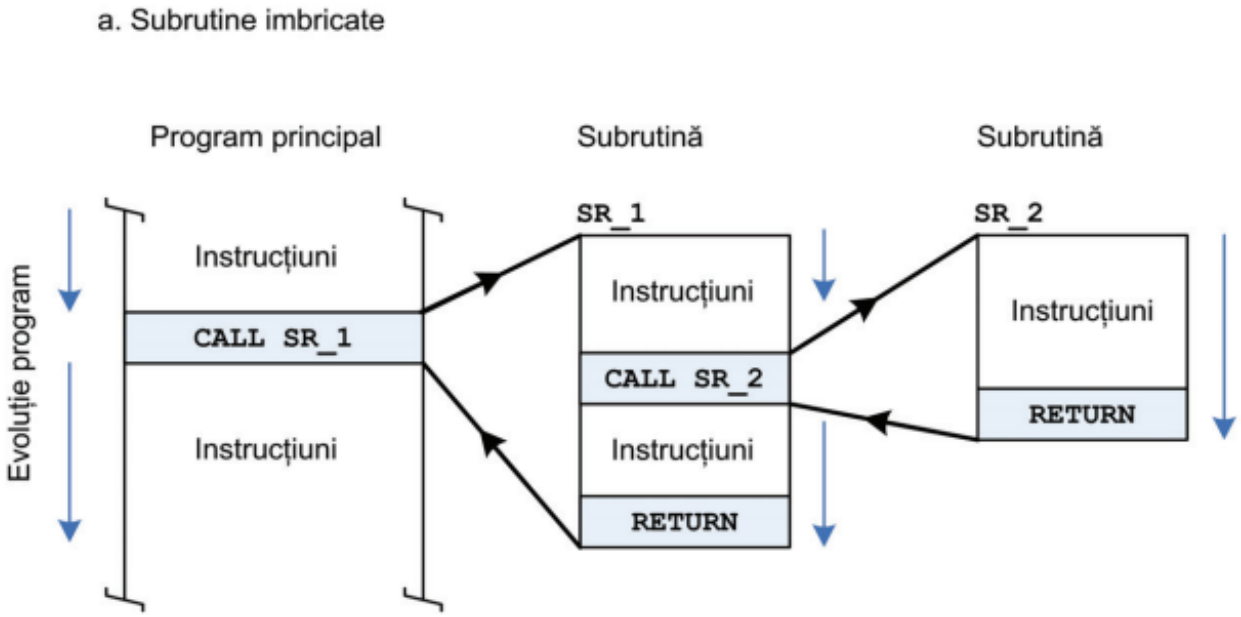
Stiva adreselor de revenire

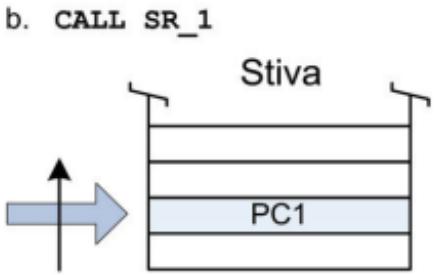
- Dimensiunea stivei: 31 de regiștrii a câte 21 de biți.

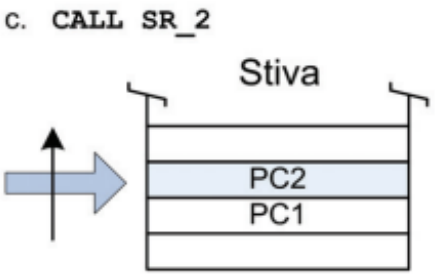


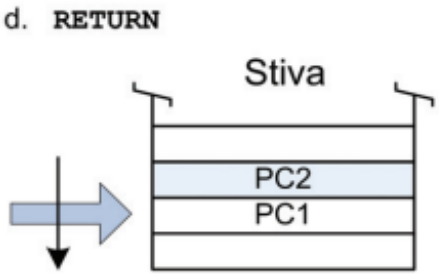
Stiva adreselor de revenire

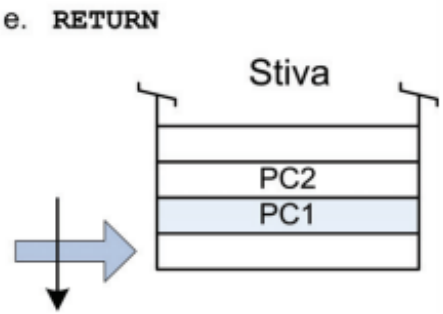
- Apeluri imbricate.





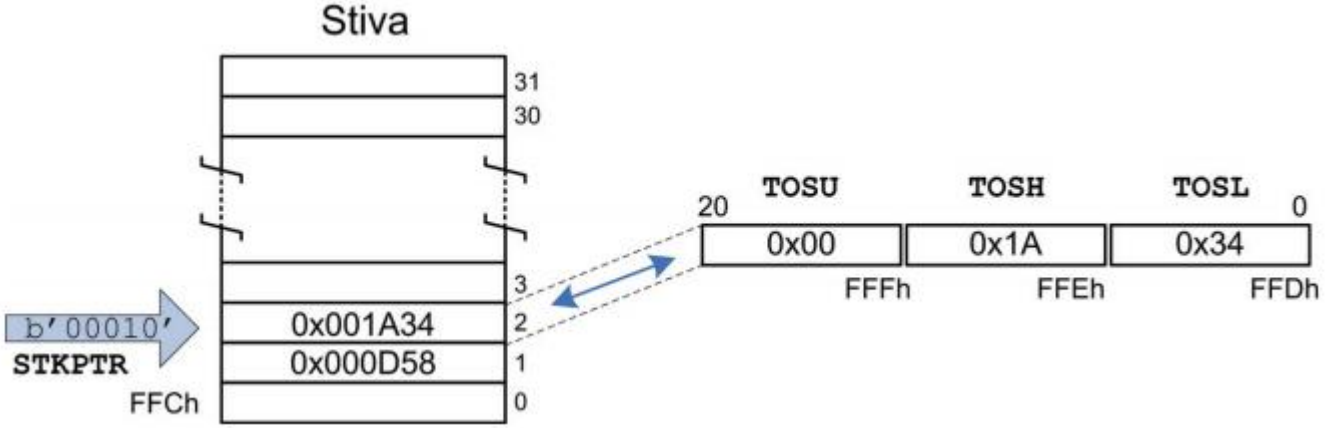






Stiva adreselor de revenire

- Structura stivei de adresare:



- TOS: Top Of Stack.
  - TOSL: TOS Low.
  - TOSH: TOS High.
  - TOSU: TOS Upper.

Stiva adreselor de revenire

- Registrul STKPTR:

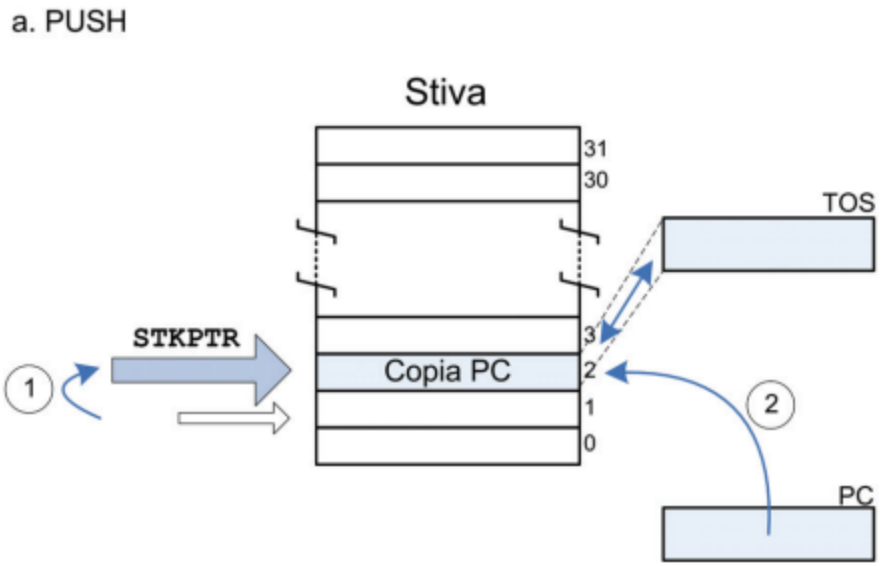
7							0
STKFUL	STKUNF	-	SP4	SP3	SP2	SP1	SP0

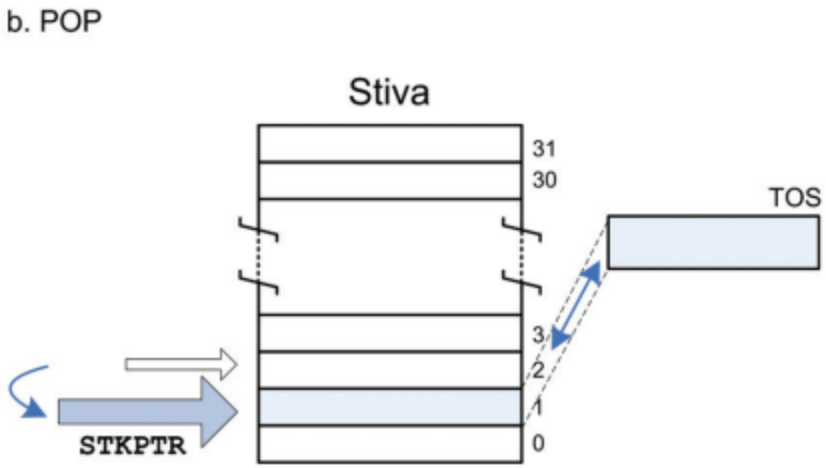
- STKPTR<4:0>: valori între 0 și 31.
- Când se ajunge la valoarea 31, bitul STKFUL se pune pe 1.
- Când se ajunge la valoarea 0 , bitul STKUNF se pune pe 1.

Stiva adreselor de revenire

- Instrucțiunea PUSH:
  - Incrementează indicatorul de stivă.
  - Copiază PC (adresa instrucțiunii ce urmează după PUSH pe stivă).
  - Asemănătoare CALL, dar fără suprascrierea PC.
- Instrucțiunea POP:
  - Decrementează indicatorul de stivă.
  - TOS se modifică cu conținutul stivei de pe noua poziție.

Stiva adreselor de revenire





Memoria de date RAM

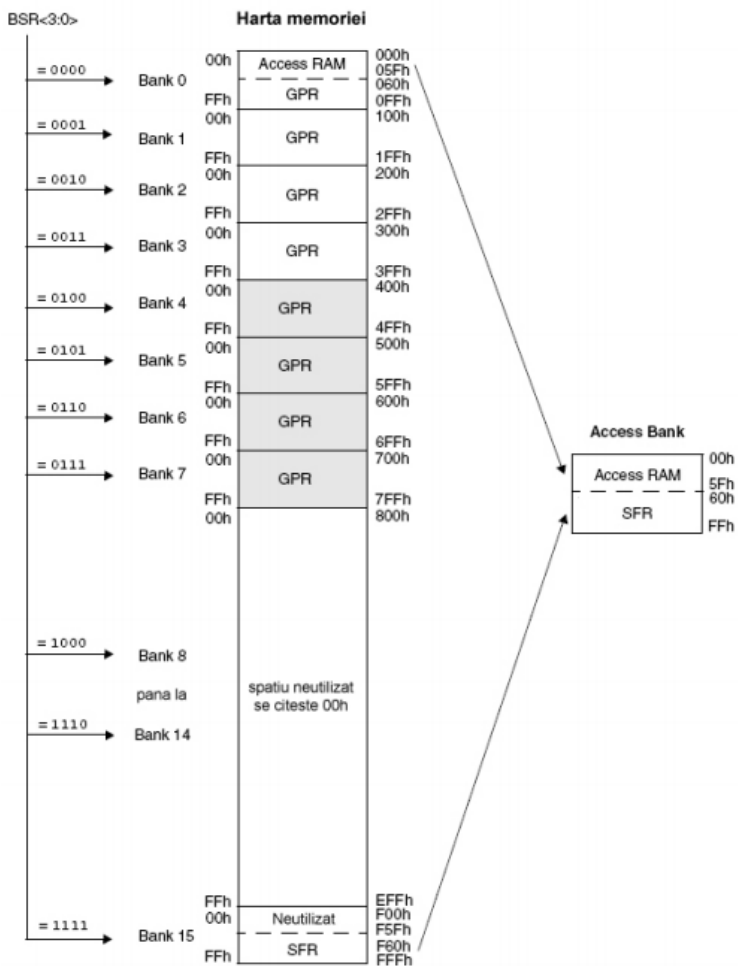
- Capacitatea de adresare penu uC din familia PIC18 este de 4KB.
- Din 4KB, PIC18F4455 implementează 2KB.
- Fiecare octet poartă denumirea de registru.
- Două categorii:
  - General Purpose Registers (GPR).
  - Special Function Registers (SFR).
- GPR și SFR sunt mapați în același spațiu de adrese ce permite accesarea lor prin același set de instrucțiuni.

Memoria de date RAM

- Pentru adresarea a 4KB e nevoie de 12 biți de adresă.
- Problemă:
  - La majoritatea instrucțiunilor câmpul de adresă e de 8 biți.
- Soluții implementate:
  - Utilizarea Bank-urilor.
  - Utilizarea Access Bank-ului.
  - Utilizarea instrucțiunii MOVFF.
  - Utilizarea pointerilor.

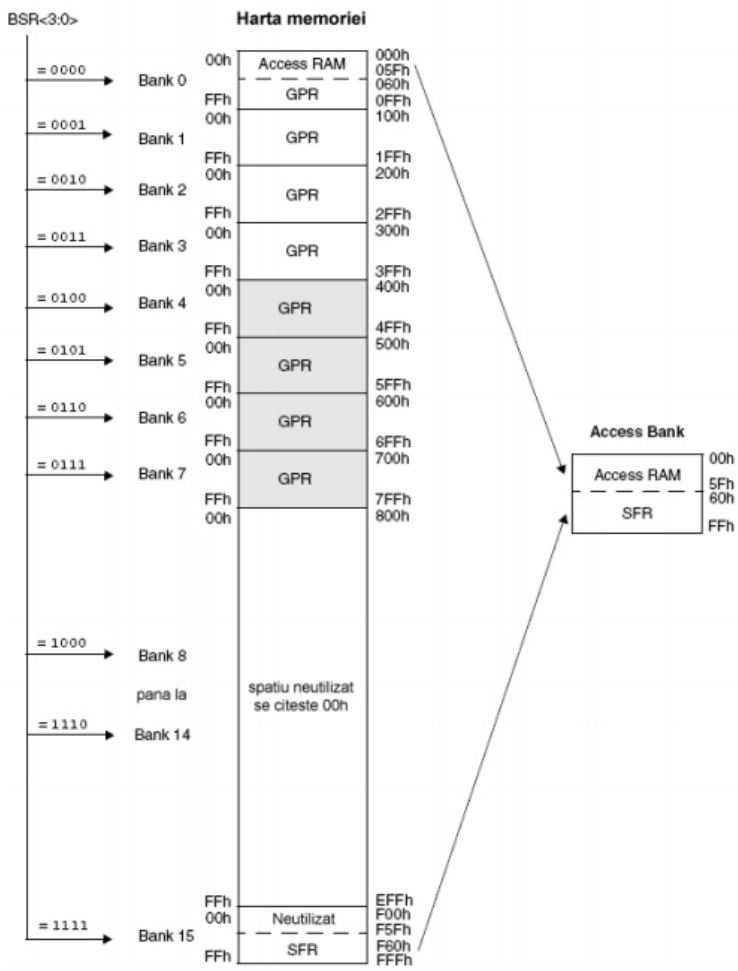
Memoria de date RAM

- 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

- SFR sunt implementați în partea superioară a Bank-ului 15 (160 de octeți).
- Bank-urile 4-7 sunt folosite pentru comunicarea prin USB.
- Pentru formarea adresei se utilizează 4 biți de selectare a bank-ului și 8 biți pentru selectarea registrului.



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

Memoria de date RAM

- Exemplu: modificarea registrului de la adresa 0x10 din Bank-ul 2:
  - MOVLB 0X02
  - MOVLW D’101’
  - MOVWF 0x10, BANKED

Memoria de date RAM

- Utilizarea BSR ridică inconveniente serioase pentru accesarea SFR și a GPR aflați în Bank-uri diferite.
- Soluția: Access Bank (AB).
  - Zonă virtuală de memorie accesibilă fără modificarea Bank-ului.
  - Mapează 96 de regiștrii GPR (Bank 0_ și 160 de regiștrii SFR (Bank 15).
- Pentru utilizarea AB, bitul a din instrucțiuni se pune pe 0, sau se folosește simbolul ACCESS.

Memoria de date RAM

- Exemplu utilizare AB:
  - CLRF TRISA, ACCESS
  - BSF TRISA, 4, ACCESS
  - INCF 0X07, ACCESS