C6-7. C2x aplicatii

Exemple

- Bit-reversed addressing
- MAC instruction
- FIR implementation
- Sine wave generation (digital oscillator)

C2x Addresing Modes – Review

1. Direct Addressing mode:

(AC)=(AC)+(0695h)

0695h = 0000 0110 1|001 0101b

DP=13 dma= 21

```
LDPK 13          ; DP=13=0dh
ADD 21  ; (AC)=(AC)+(21)
ADD 21,3         ; (AC)=(AC)+(21)*8
```

2. Indirect Addressing mode :

```
LARP 3           ;ARP=3
LRLK AR3,695h    ;AR3=0695h
ADD *, 3         ;(AC)=(AC)+(695h)*8
```

{ * | * + | * – | * 0 + | * 0 – | * BR0 + | * BR0 –} ;BRO – bit reversed

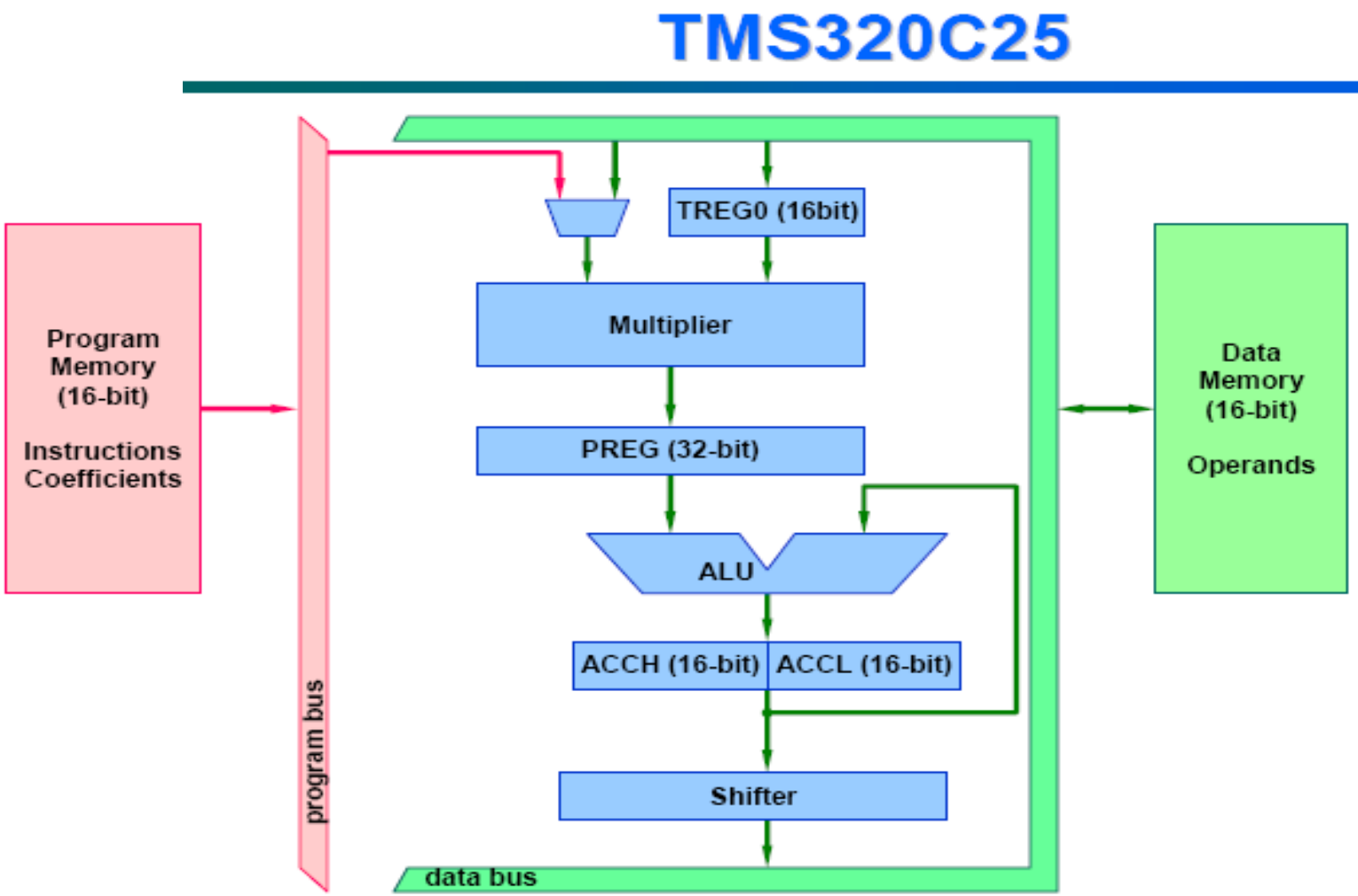
3. Immediate Addressing mode

```
ADLK 187,3       ; (AC)=(AC)+187h*8
                 ; ADLK   - add immediate value* 8 to accumulator
```

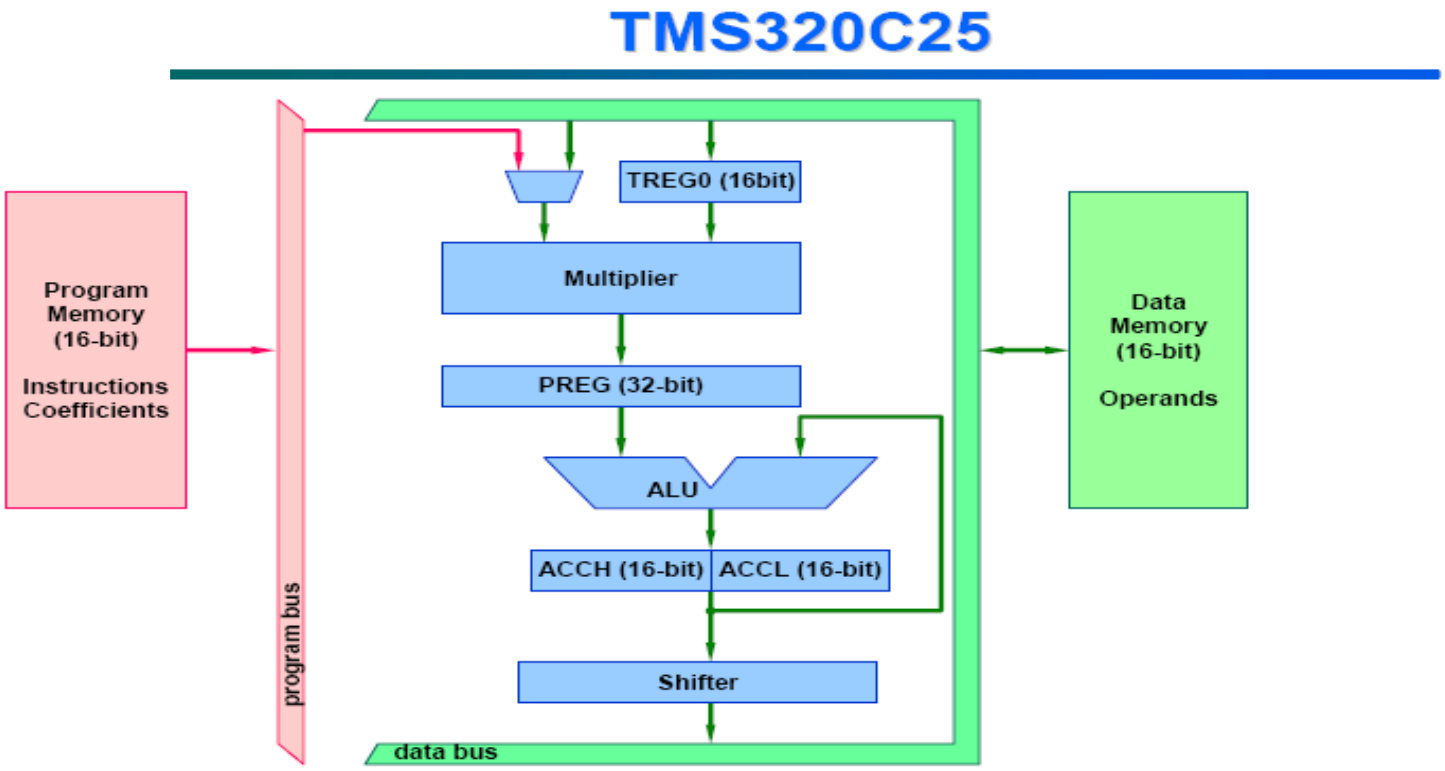
Ex1. Write the program sequence in C2x A.L. which compute:

(202H, 203H) = (200H) * (201H) ; B0 data memory block

Central ALU



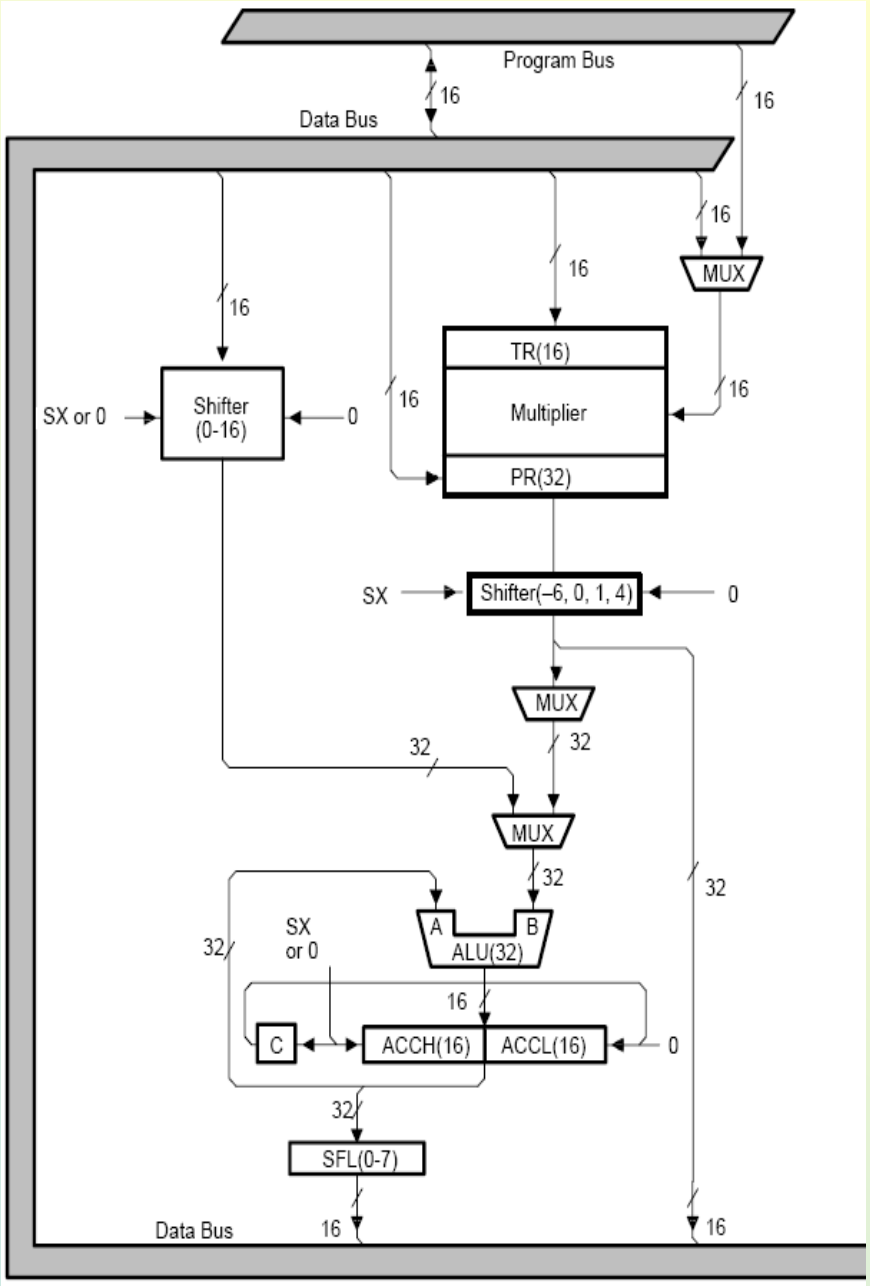
```
LARP 1                  ; ARP=1
LRLK  AR1,200h          ; AR1=200h, address data block B0
LT  *+                  ; T=(200h), AR1=201h
MPY *+                  ; P=T*(201h) , AR1=202h
PAC                     ;ACC=P
SACH*+                  ;(202h)=ACCH, AR1=203h
SACL*                   ;(203h)=ACCL ; big endian
```



Ex.2 Write the program sequence in C2x A.L. which compute: ACC= Y = AX1+BX2+CX3+DX4 ;

```
. data
A  word …
B  word …
……
X1  word …
```

```
        ZAC             ;ACC=0
        LT  X1          ;T=X1
        MPY  A          ;P=A*X1
        LTA  X2         ; ACC=AX1, T=X2
        MPY  B          ; P=B*X2
        LTA  X3         ;ACC=ACC+BX2, T=X3
        MPY  C          ;P=C*X3
        LTA  X4         ;ACC=ACC+CX3, T=X4
        MPY  D          ;P=D*X4
        APAC            ;ACC=ACC+P
        SACH  Y1        ;store 32 bits result
        SACL  Y2        ; at Y1,Y2
```



Ex. 3:

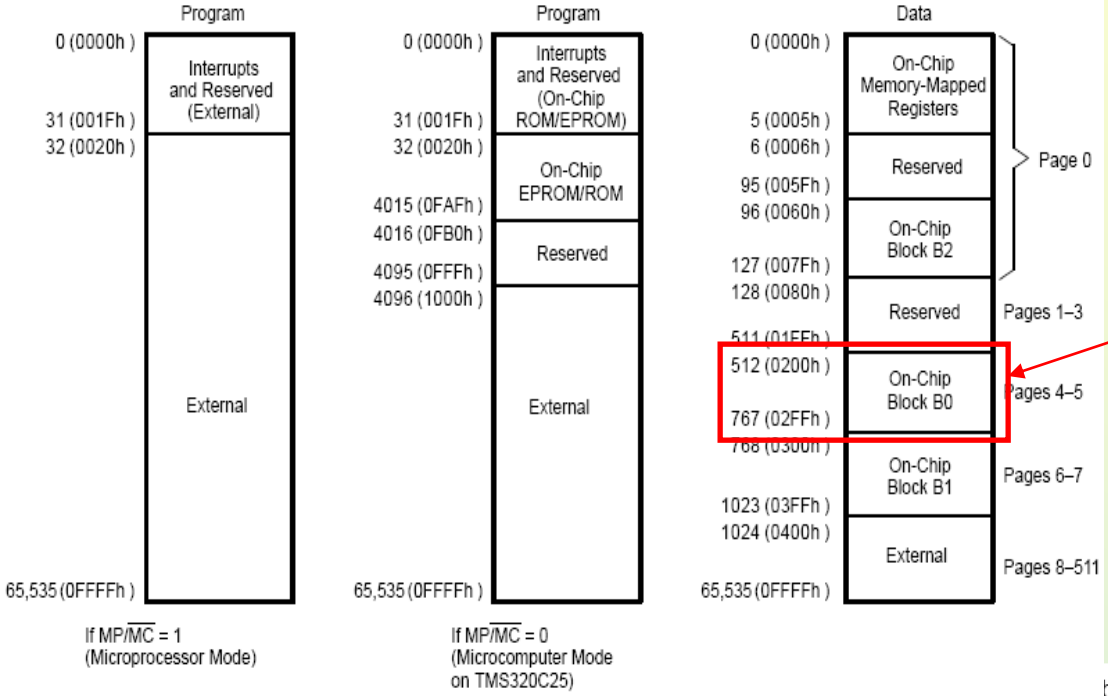
Blocul B0 (200h-2FFh) contine:

200h	201h	202h	203h	204h	205h	206h	207h
10h	20h	40h	80h	100h	200h	400h	800h

Care este continutul blocului B2 (60h-67h) dupa secventa urmatoare de program?

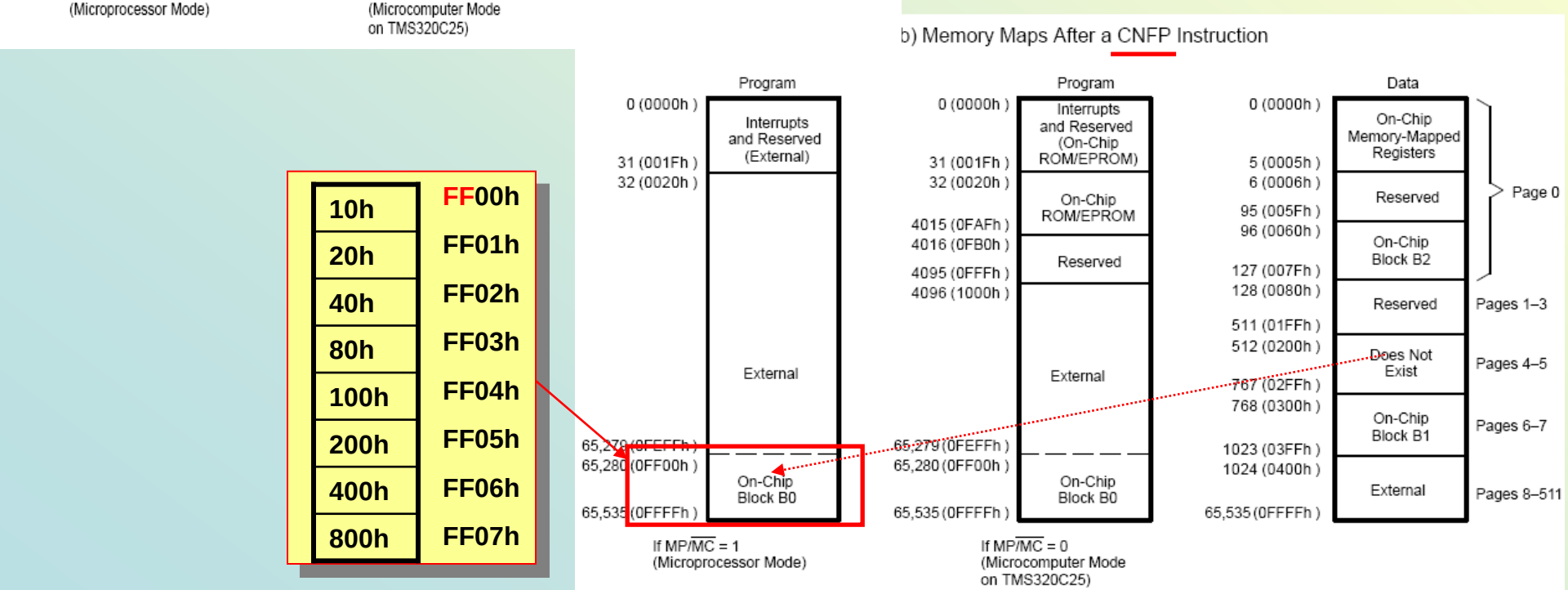
```
CNFP
LARP 1
LRLK  AR1,60h
LRLK  AR0,4
RPTK 7
BLKP  FF00h,*BR0+
```

(a) Memory Maps After a CNFD Instruction



10h	200h
20h	201h
40h	202h
80h	203h
100h	204h
200h	205h
400h	206h
800h	207h

(b) Memory Maps After a CNFP Instruction



10h	FF00h
20h	FF01h
40h	FF02h
80h	FF03h
100h	FF04h
200h	FF05h
400h	FF06h
800h	FF07h

Rezolvare:

Blocul B0 (200h-2FFh) dupa CNFP continue (FF00h-FF07h):

F000h	F001h	F002h	F003h	F004h	F005h	F006h	F007h
10h	20h	40h	80h	100h	200h	400h	800h

Care este continutul blocului B2 (60h-67h) dupa secventa urmatoare de program?

```
CNFP                ;conf Bloc B0 mem program (FF00h-FFFFh)
LARP 1              ; ARP=1
LRLK  AR1,60h       ; AR1=60h
LRLK  AR0,4         ;AR0=4
RPTK 7              ;repeta urmatoarea instr. de 8 x
BLKP  FF00h,*BR0+
```

200h	201h	202h	203h	204h	205h	206h	207h
10h	20h	40h	80h	100h	200h	400h	800h

Dupa CNFP

FF00h	FF01h	FF02h	FF03h	FF04h	FF05h	FF06h	FF07h
10h	20h	40h	80h	100h	200h	400h	800h

BLKP FF00h,*BR0+

60h	61h	62h	63h	64h	65h	66h	67h
10h	100h	40h	400h	20h	200h	80h	800h

Adresarea bit-reversed

0 (000) => 0 (000)
1 (001) => 4 (100)
2 (010) => 2 (010)
3 (011) => 6 (110)
4 (100) => 1 (001)
5 (101) => 5 (101)
6 (110) => 3 (011)
7 (111) => 7 (111)

Mem. Prog.(B0)

10h	FF00h
20h	FF01h
40h	FF02h
80h	FF03h
100h	FF04h
200h	FF05h
400h	FF06h
800h	FF07h

Mem. Data (B2)

10h	60h
20h	64h
40h	62h
80h	66h
100h	61h
200h	65h
400h	63h
800h	67h

10h	60h
100h	61h
40h	62h
400h	63h
20h	64h
200h	65h
80h	66h
800h	67h

60h= 0110 0000 +
0100
64h= 0110 0100 +
0100
62h= 0110 0010 +
0100
66h= 0110 0110

BLKP – mută bloc din memoria de program în memoria de date

Sintaxa : ad.dir.: [etichetă] BLKP pma,dma [;com]
ad.dir.: [etichetă] BLKP pma, {ind}[,ARP următor] [;com]

Operanzi : $0 \leq pma \leq 65535$
$0 \leq dma \leq 127$
$0 \leq$ ARP următor ≤ 7

Cuvinte : 2

Execuţie: (PC)+2→PC
(PFC)→MCS
pma→PFC
Dacă (controlul de repetare) ≠0, atunci
(pma, adresată de PFC)→dma,
Modifica AR(ARP) şi ARP conform specificaţiei,
(PFC)+1→PFC,
(controlul de repetare)-1→controlul de repetare.

În caz contrar:
(pma, adresată de PFC)→dma,
Modifică AR(ARP) şi ARP conform specificaţiei.

Descriere: Cuvinte consecutive de memorie sunt mutate dintr-un bloc sursă de memorie de program intr-un bloc de destinaţie de memorie de date. Adresa initial (cea mai mică) a blocului sursă este definite de al doilea cuvâ al instrucţiunii. Adresa initială a blocului destinaţie este definite fie de dma conţinută în codul instrucţiunii (la adresa directă), fie de registrul AR curent (la adresa indirectă). În cazul modului de adresare indirectă, atât AR cât şi ARP pot fi modificaţi ca de obicei. În cazul modului de adresare direct dma este utilizată ca adresă destinaţie pentru mutarea blocului, dar nu este modificată la execuţia repetată a instrucţiunii. De aceea conţinutul memorie la adresa dma va fi acelaşi cu conţinutul memoriei la ultima adresă pma din secvenţa repetată.

Dacă trebuie mutate mai multe cuvinte, este necesară folosirea instrucţiunilor RPT sau RPTK împreună cu BLKP în modul de adresare indirectă. Numărul de cuvinte ce vor fi mutate este cu unul mai mare decât numărul conţinut în controlul de repetări, RPTC, la începutul instrucţiunii. La sfârsitul acestei instrucţiuni, RPTC conţine zero şi , dacă se foloseşte adresarea indirectă, AR(ARP) va fi modificat şi va conţine adresa de după sfârşitul blocului destinaţie.

Tema1:

Blocul B0 contine:

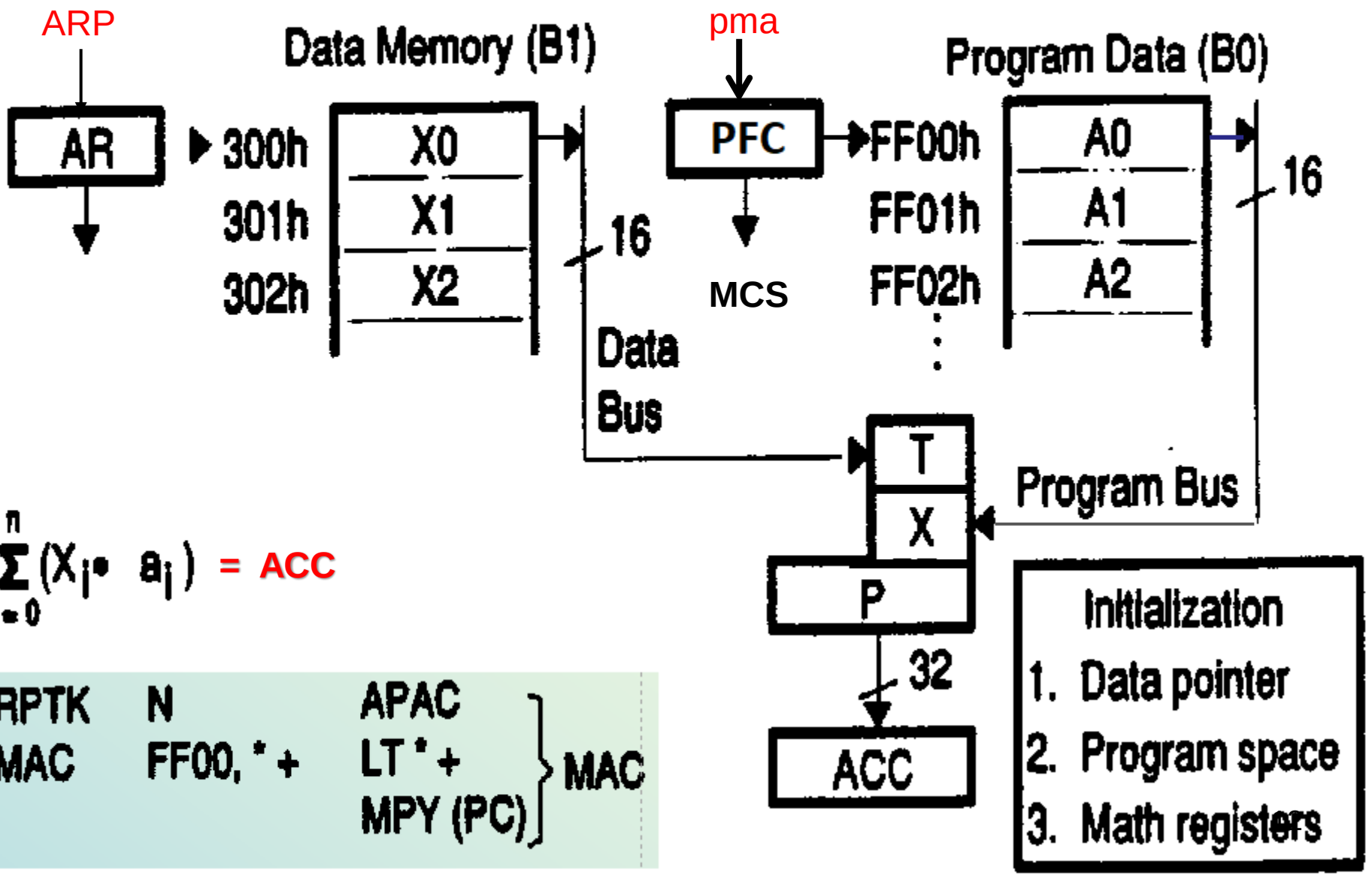
200h	201h	202h	203h	204h	205h	206h	207h
10h	20h	40h	80h	100h	200h	400h	800h

Care este continutul blocului B1 (100h-107h) dupa secventa urmatoare de program?

```
CNFP
LARP 1
LRLK  AR1,107h
LRLK  AR0,4
RPTK 7
BLKP  FF00h,*BR0-
```

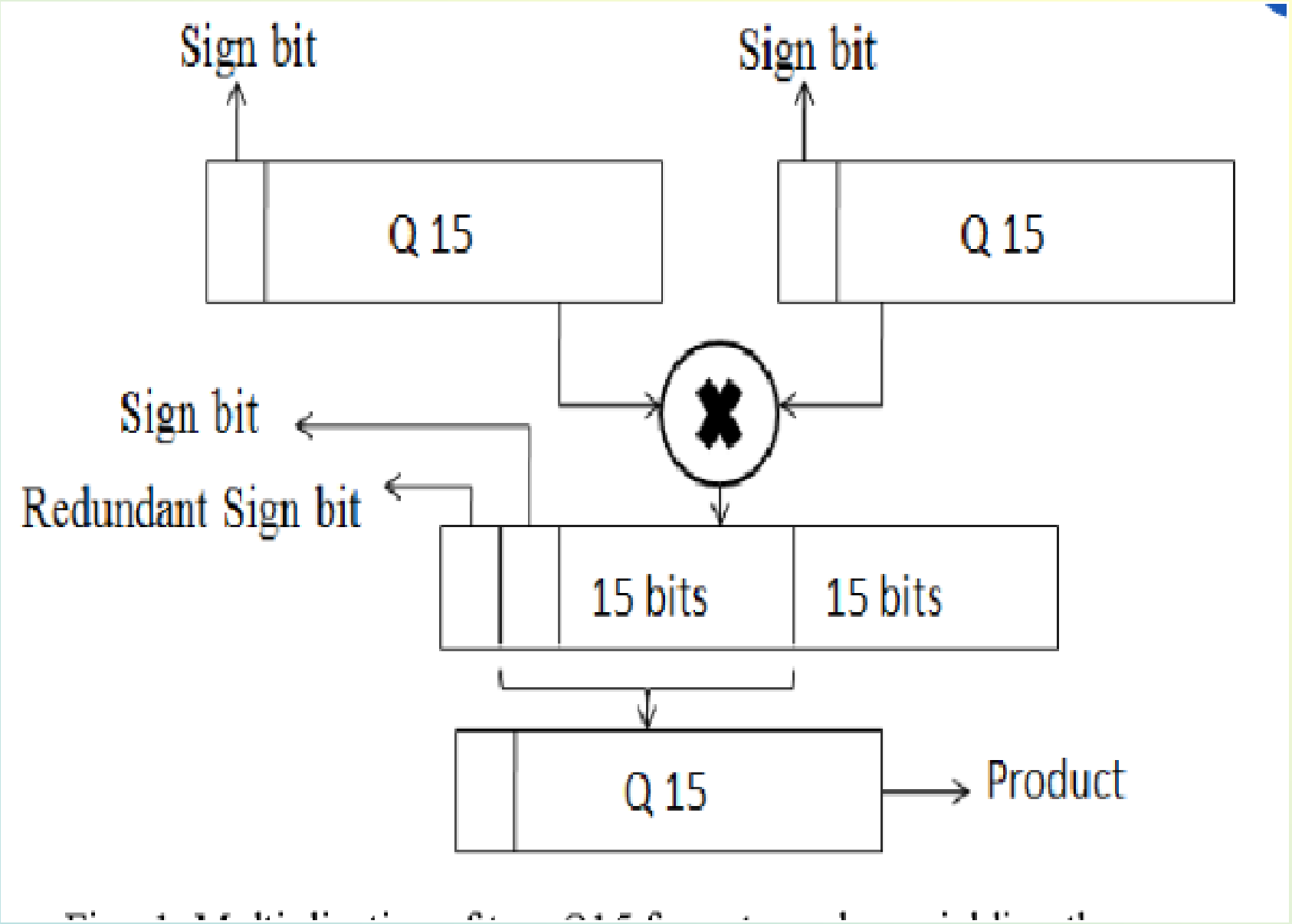
Ex.4.

Multiply with Accumulate (MAC)



$$\sum_{i=0}^{n} (X_i \cdot a_i) = \text{ACC}$$

```
RPTK   N          APAC          }
MAC    FF00, *+   LT *+         } MAC
                  MPY (PC)      }
```


Sign bit
Sign bit
Q 15
Q 15
Sign bit
Redundant Sign bit
15 bits
15 bits
Q 15
Product

Ex5. Fill in the table below and comment where appropriate execution .
The initial state for the execution sequence is flag CNF=0, (B0 – Data Memory):

Address	300h	301h	302h	303h	304h	305h	306h	307h
Content	**100h**	**200h**	**400h**	**800h**	**700h**	**500h**	**300h**	**100h**

	ACCL	DP	ARP	AR0	AR6	AR7	77h	72h	CNF	OBS.
LARP 7										
LRLK AR6,70h										
LRLK AR7,200h										
RPTK 7										8 x
BLKD 300h,*+										#
CNFP										
LRLK AR0,4										
LARP 6										
RPTK 7										
BLKP FF00h,*BR0+										#
LDPK 0										
LAC 76h										
SUB 72h,1										

- Comments

BLKD – Move Mdata>>Mdata: **B1** → **B0(data)**

(300h-3FFh)→ (200h-2FFh)

Addresss B1	300h	301h	302h	303h	304h	305h	306h	307h
Address B0(d)	200h	201h	202h	203h	204h	205h	206h	207h
Content	**100h**	**200h**	**400h**	**800h**	**700h**	**500h**	**300h**	**100h**

BLKD – Move Mdata>>Mdata: **B1** → **B0(data)**

(300h-3FFh) → (200h-2FFh)

Addresss B1	300h	301h	302h	303h	304h	305h	306h	307h
Address B0(d)	200h	201h	202h	203h	204h	205h	206h	207h
Content	**100h**	**200h**	**400h**	**800h**	**700h**	**500h**	**300h**	**100h**

CNFP – B0 program mem.: **B0(p)**

(FF00h-FFFFH)

Address B0(p)	FF00h	FF01h	FF02h	FF03h	FF04h	FF05h	FF06h	FF07h
Content	**100h**	**200h**	**400h**	**800h**	**700h**	**500h**	**300h**	**100h**

BLKP FF00h,*BR0+

BLKP – Move Mprog. >>*BR0 Mdata: **B0(p)** → **B2**

FF00h-FFFFh → 60h-7Fh

Address B0(p)	FF00h	FF01h	FF02h	FF03h	FF04h	FF05h	FF06h	FF07h
Address B2	70h	74h	72h	76h	71h	75h	73h	77h
Content	**100h**	**200h**	**400h**	**800h**	**700h**	**500h**	**300h**	**100h**

Ex.6. Fill in the table below :

		ACC	DP	ARP	T	P	AR1	AR2	PM	300h	200h	201h	284h
	Setup values					10h			1	10h	2	4	100h
1	LDPK 5												
2	LRLK AR1, 200h												
3	LRLK AR2, 300h												
4	LARP 1												
5	ZAC												
6	ADD 4, 2												
7	LT *+												
8	MPYA *, 2												
9	XORK 80h												
10	SUB *, 1, 1												

FIR on conventional DSP

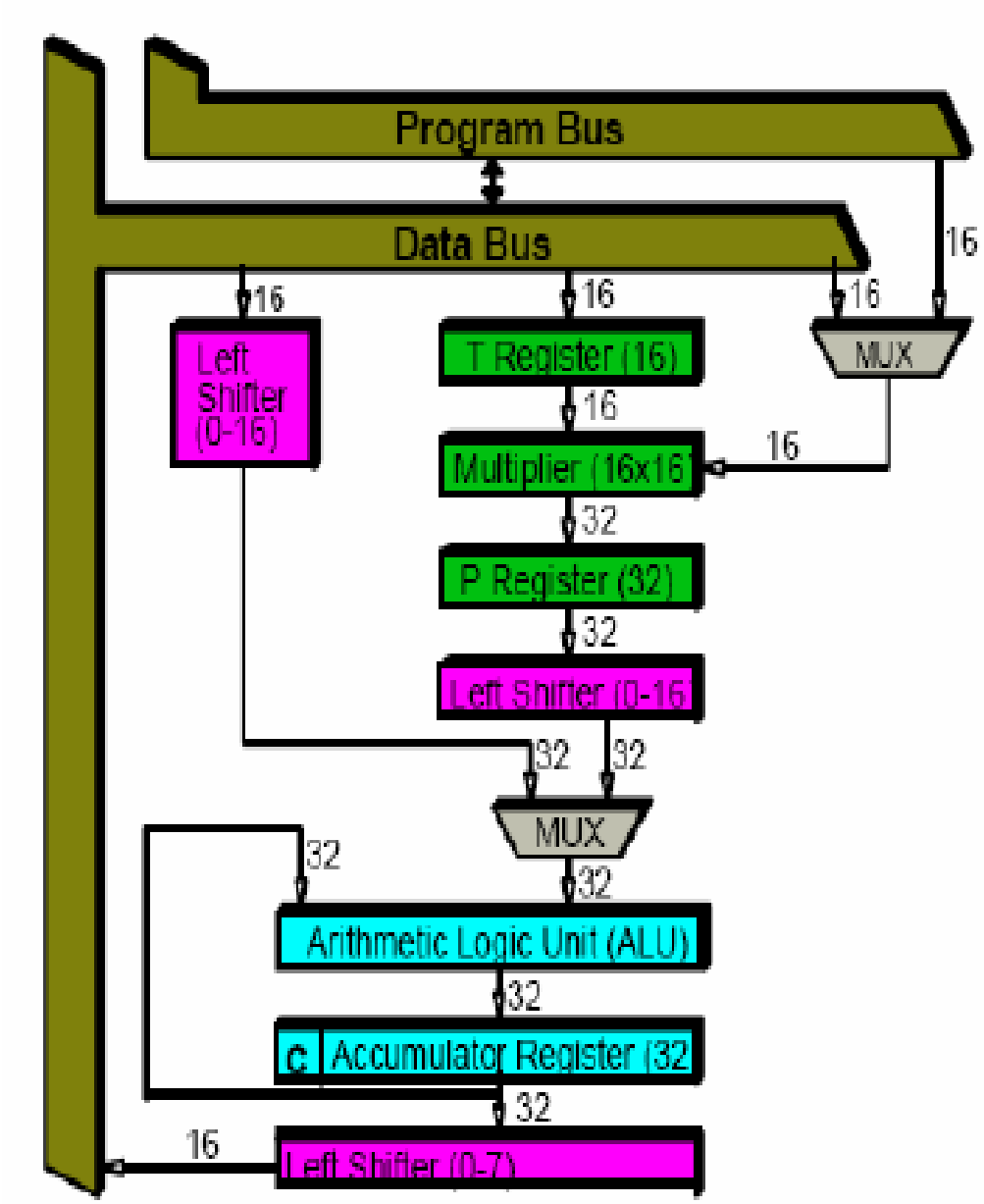
- TMS320C2x

```
ZAC || LTD;
LTD || MPY;

RPTK N-1
MACD

APAC || MPY;
APAC;
```

Exécution en N cycles



Ex.7.

FIR Filtering: A Motivating Problem

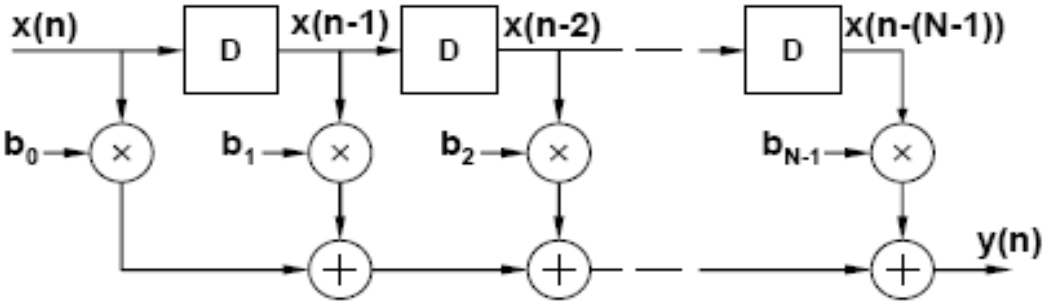
Transfer Function

$$H(z) = \sum_{k=0}^{N-1} b_k z^{-k}$$

Difference Equation

$$y[n] = \sum_{k=0}^{N-1} b_k x[n-k]$$

Block Diagram



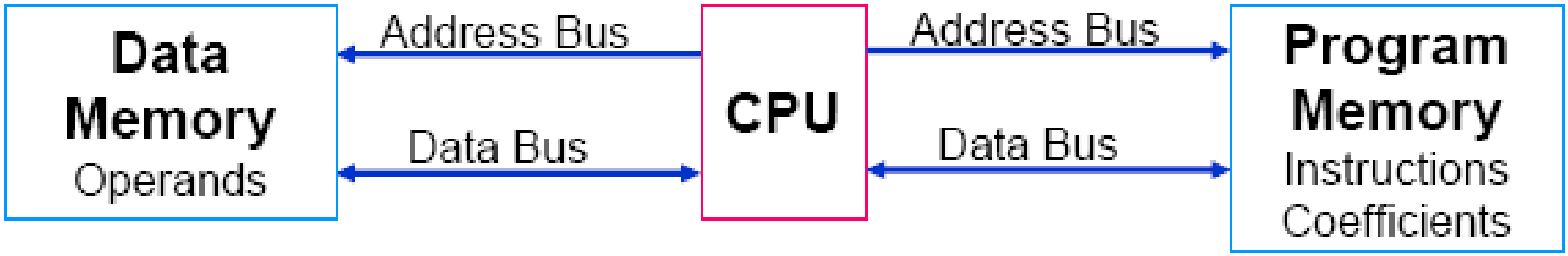
High-Level Language Description

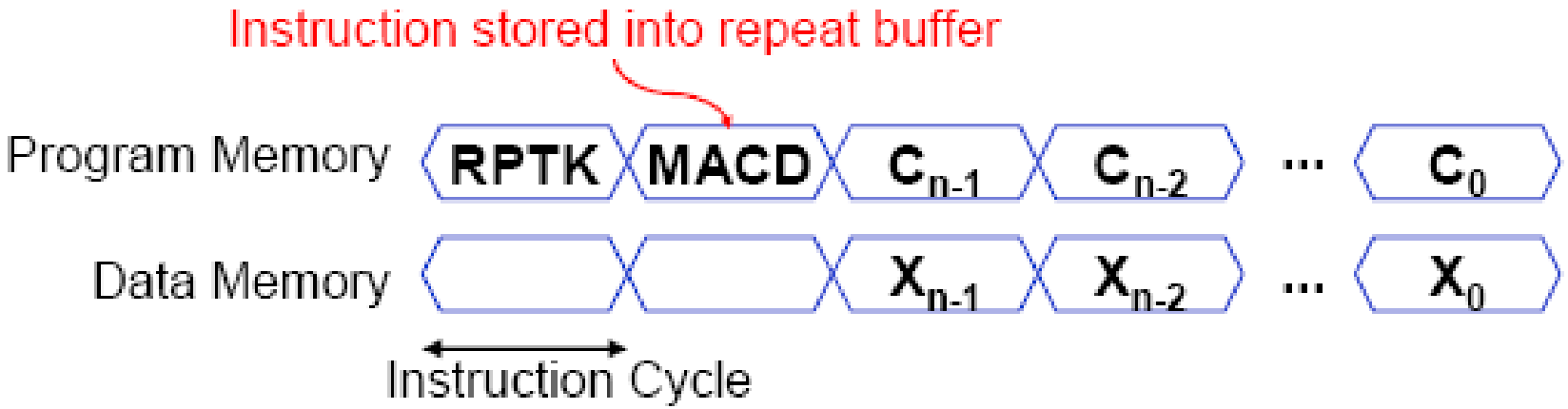
```
y = 0;x[0]=xin;
for(k=N-1; k>=0; k--) {
    y += x[k]*b[k];
    x[k]=x[k-1];
}
```

TEMA !!!!! Study the material attached on lecture site digfilt.pdf

Repeat Buffer in ‘C25

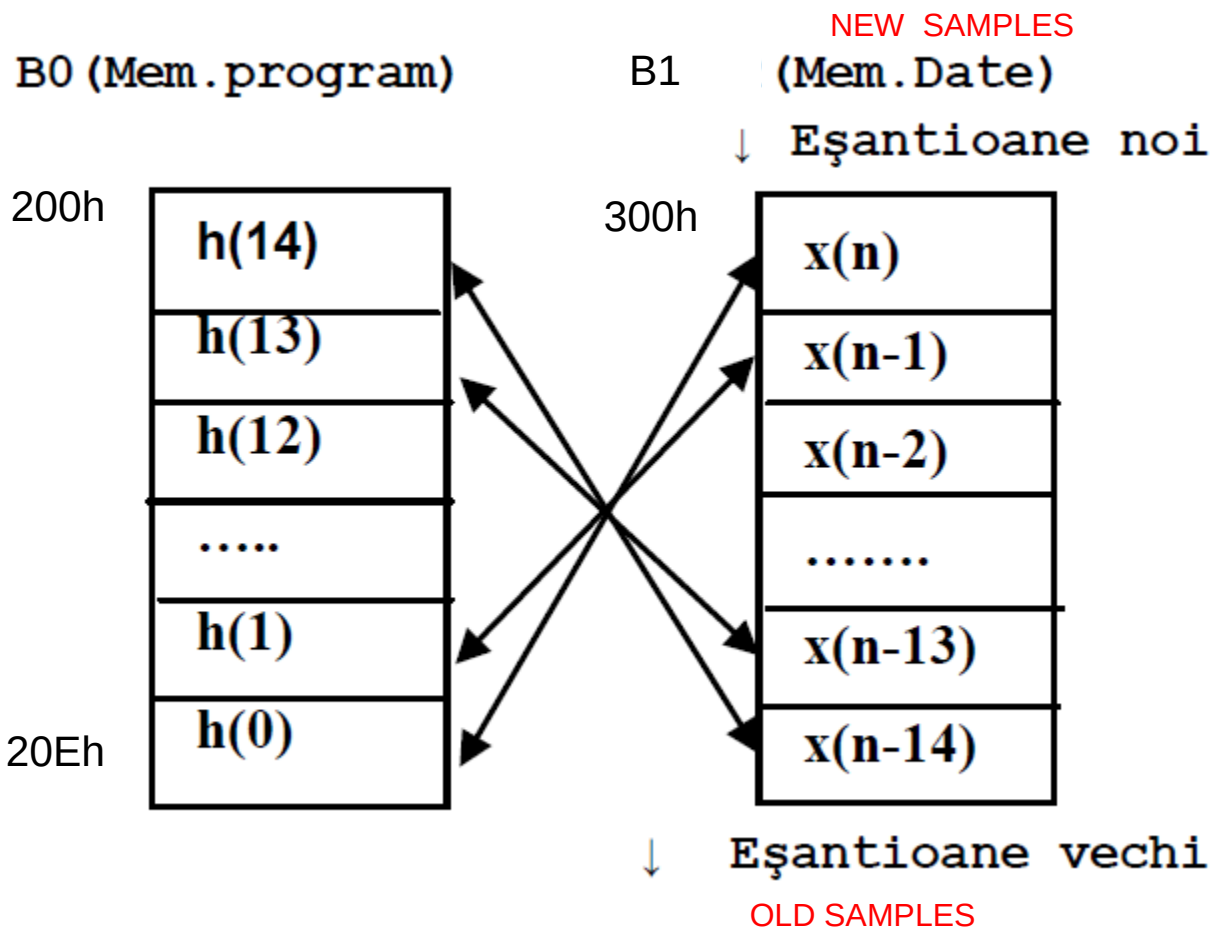
```
    RPTK    n-1          ; repeat the following
*                        ; instruction n times
    MACD    Xn-1,Cn-1    ; perform MAC with
*                        ; delay line update
```





$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k)$$

N=15



HW 1. What function implements the below program sequence (AL C2x) and where is stored the result and comments the instructions?

Address	300h	301h						363h
Content	X(0)	X(1)						X(99)

```
LDPK    4       ;
LARK    AR1,300h
LARP    1
ZAC
RPTK    99
ADD     *+
SACL    81h
SACH    80h
```

Ex.9. Fill in the table below

		ACC	DP	ARP	T	P	AR3	AR4	PM	60h	61h	70h	FF00h	CNF	C
	Valori initiale					10h			1	4	8	20h	2	1	0
1	LDPK 0														
2	LRLK AR3, 60h														
3	LRLK AR4, 70h														
4	LARP 3														
5	LALK 20h														
6	SUB *,1														
7	LTA *+														
8	MPY *, 4														
9	ROR														
10	MAC FF00h,70h														
11	PAC														
12	SUB *, 1														

Ex. 10. What function implements the below program sequence and where is stored the result ?

Address	**300h**	**301h**						**363h**
Content	**X(0)**	**X(1)**						**X(99)**

```
LDPK    4          ;bloc B1
LARK    AR1,300h
LARP    1
ZAC
RPTK    99
ADD     *+
SACL    80h
```