C6-7. C2x applications

Examples

- Bit-reversed addressing
- MAC instruction
- FIR implementation
- Tables with instructions sequence
- Sine wave generation (digital oscillator)

C2x Addresing Modes – Review

1. Direct Addressing mode:

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

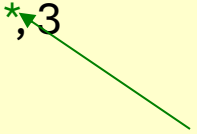
0695h = 0000 0110 1|001 0101b

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

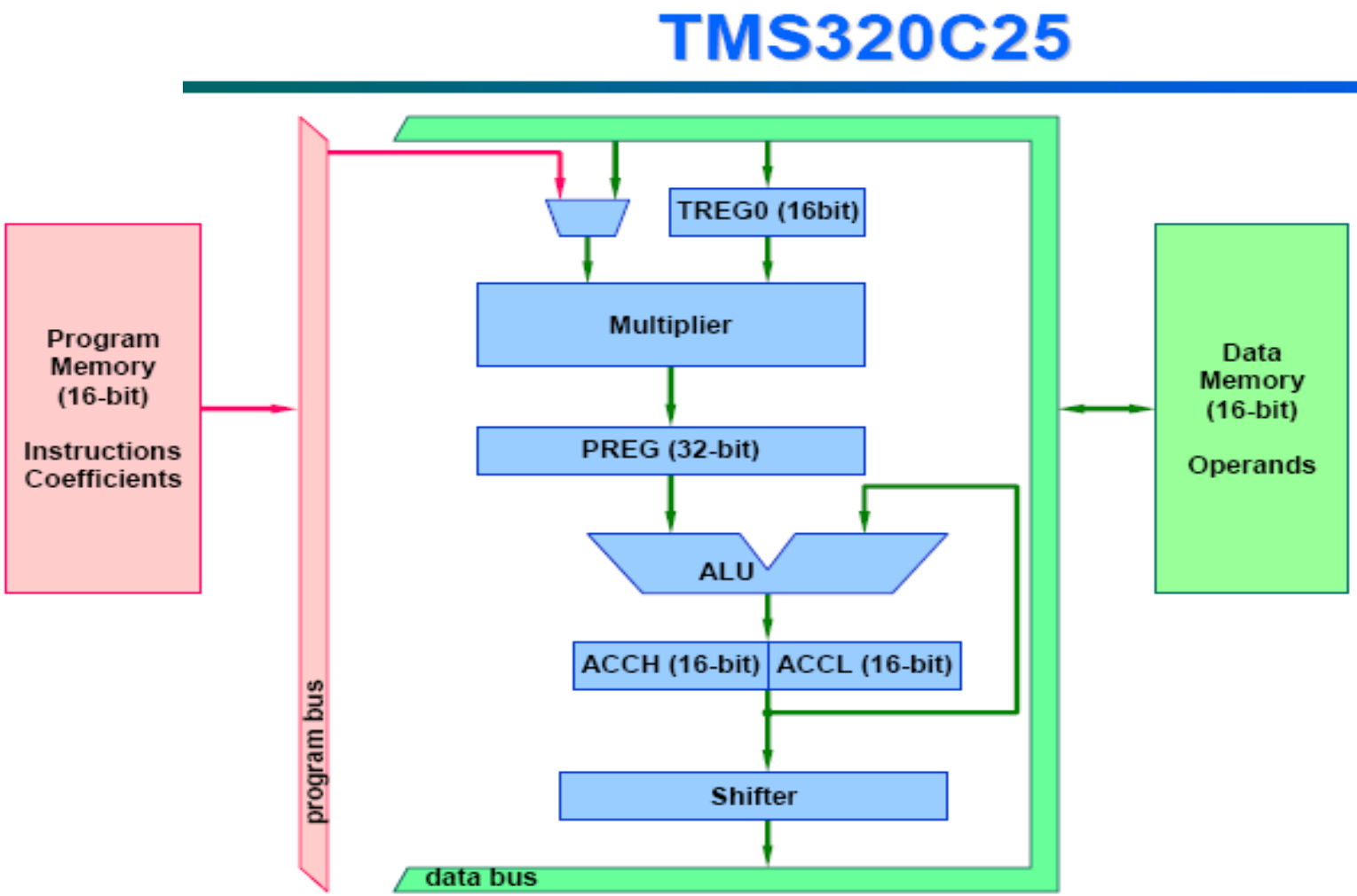
3. Immediate addressing mode

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

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

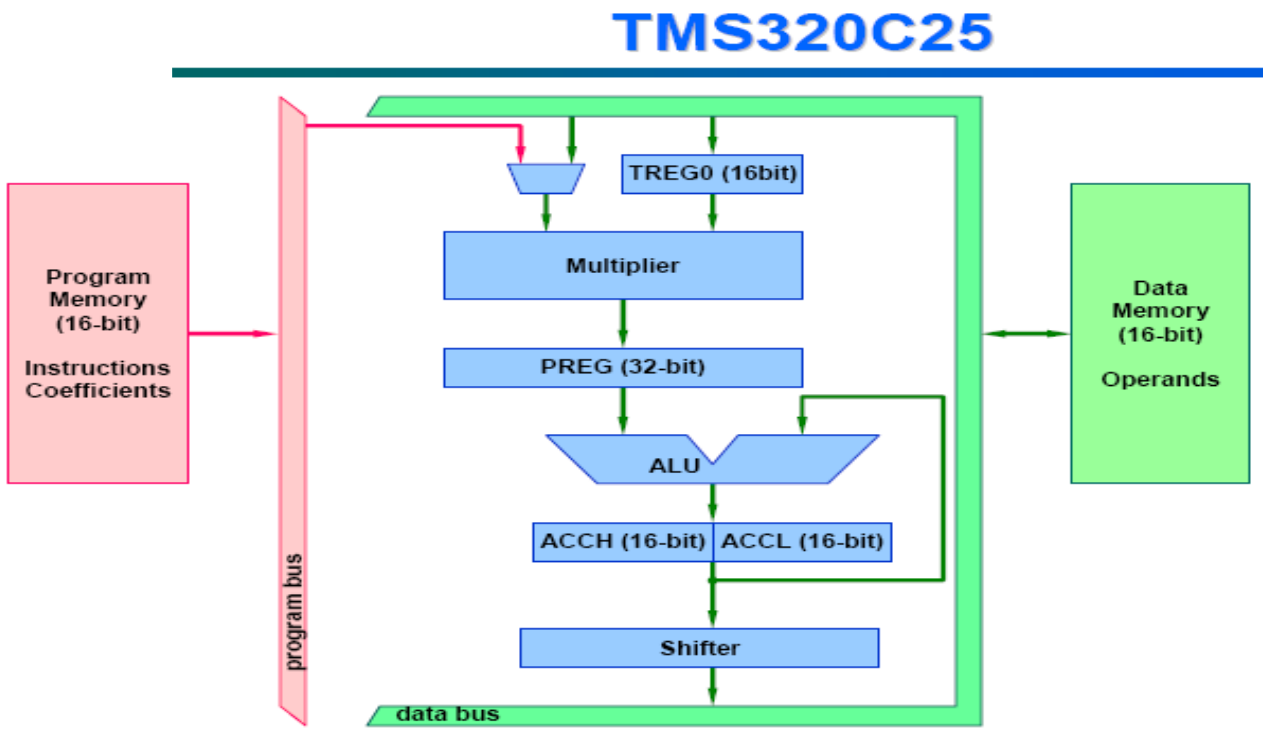
(202H, 203H) = (200H) * (201H) ; B0 data memory block

Central ALU



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

```
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
```

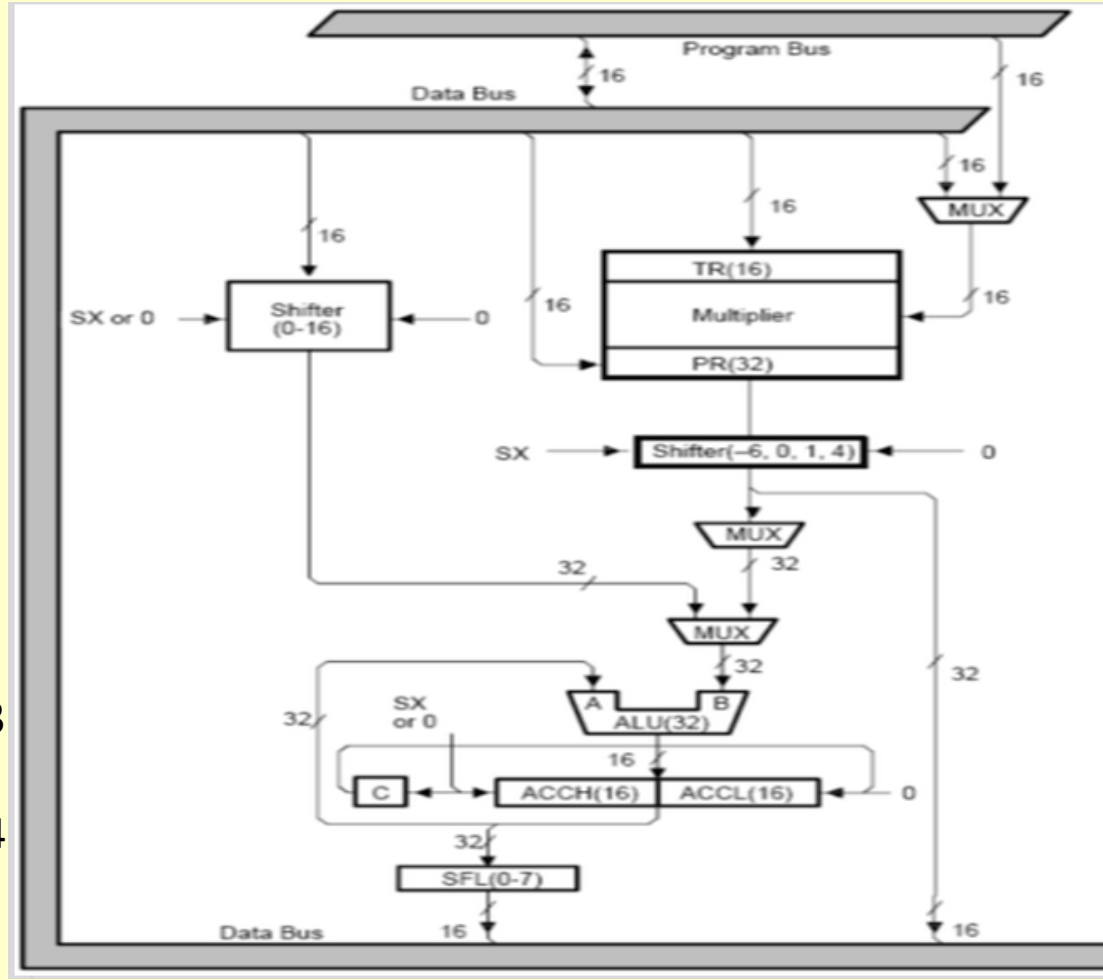


Ex.2 Write the program sequence in C2x A.L. which compute:

$$Y = AX1+BX2+CX3+DX4 ;$$

```
. 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. The beginning of block B0 (data memory) content may be found below:

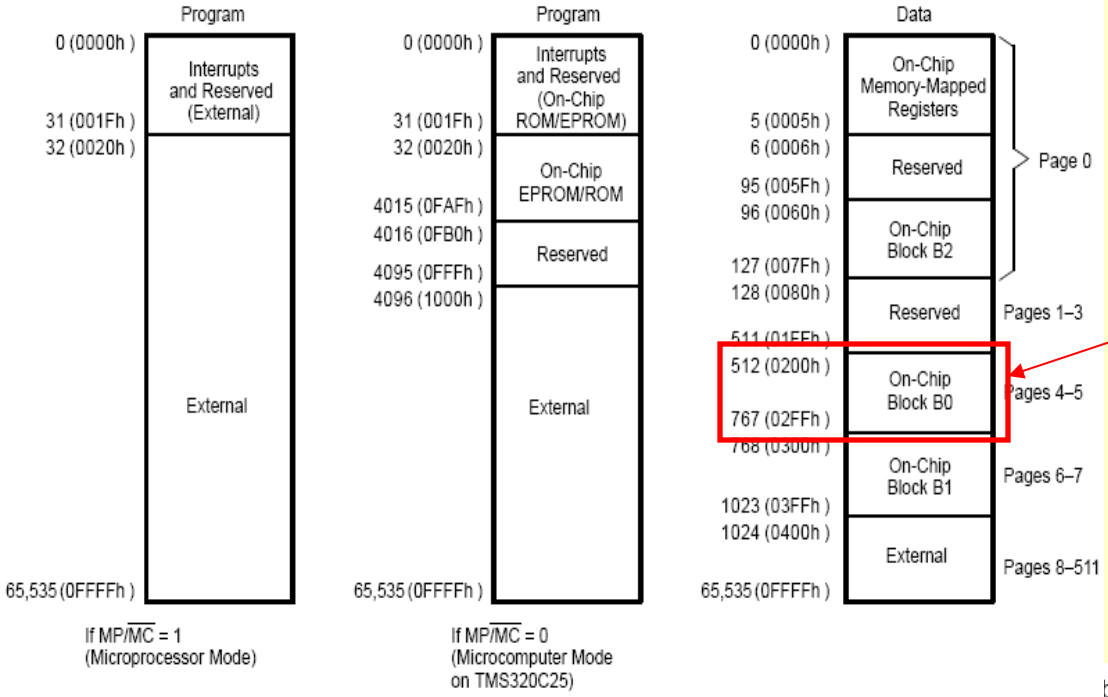
Block B0 :

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

What is the B2 (60h-67h) content after the C2x L.A. program sequence?

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

(a) Memory Maps After a CNFD Instruction



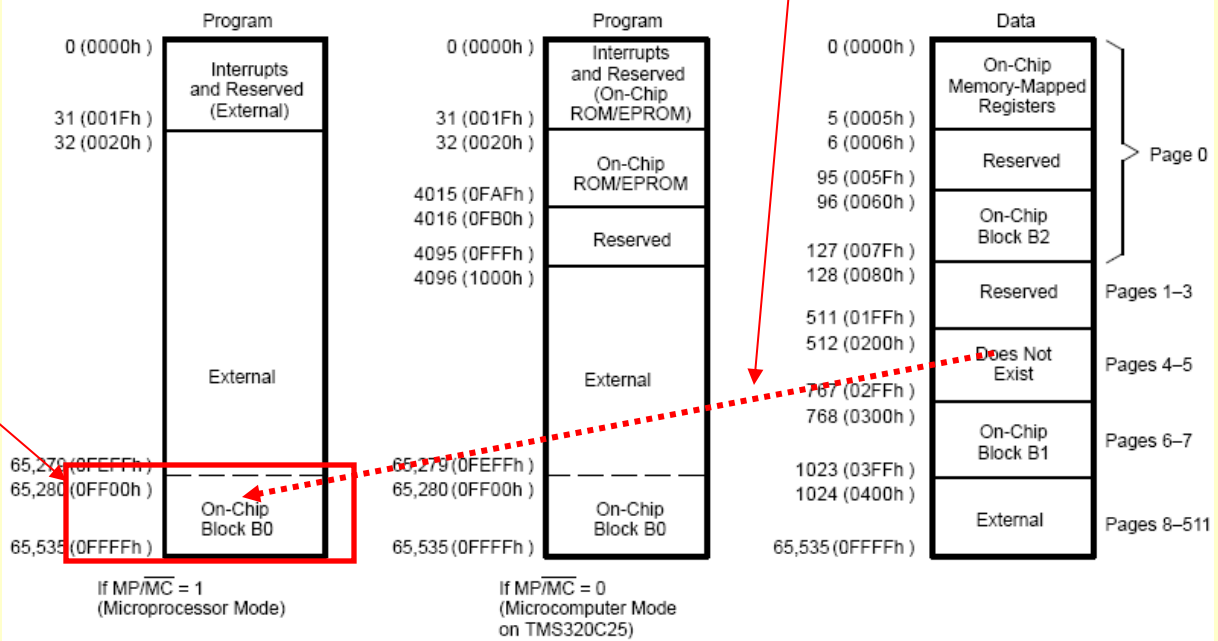
10h	200h
20h	201h
40h	202h
80h	203h
100h	204h
200h	205h
400h	206h
800h	207h

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

After CNFP

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

(b) Memory Maps After a CNFP Instruction



Solution:

Block B0 (200h-2FFh) content:

200h	**201h**	**202h**	**203h**	**204h**	**205h**	**206h**	**207h**
10h	20h	40h	80h	100h	200h	400h	800h
FF00h	FF01h	FF02h	FF03h	FF04h	FF05h	FF06h	FF07h

Content of block B2 (60h-67h) after the C2x L.A. program sequence?

```
CNFP                ; B0 viewed program memory (FF00h-FFFFh)
LARP 1              ; ARP=1
LRLK  AR1,60h       ; AR1=0060h
LRLK  AR0,4         ;AR0=4
RPTK 7              ;repeat next instr.  8 X
BLKP  FF00h,*BR0+   ;next
```

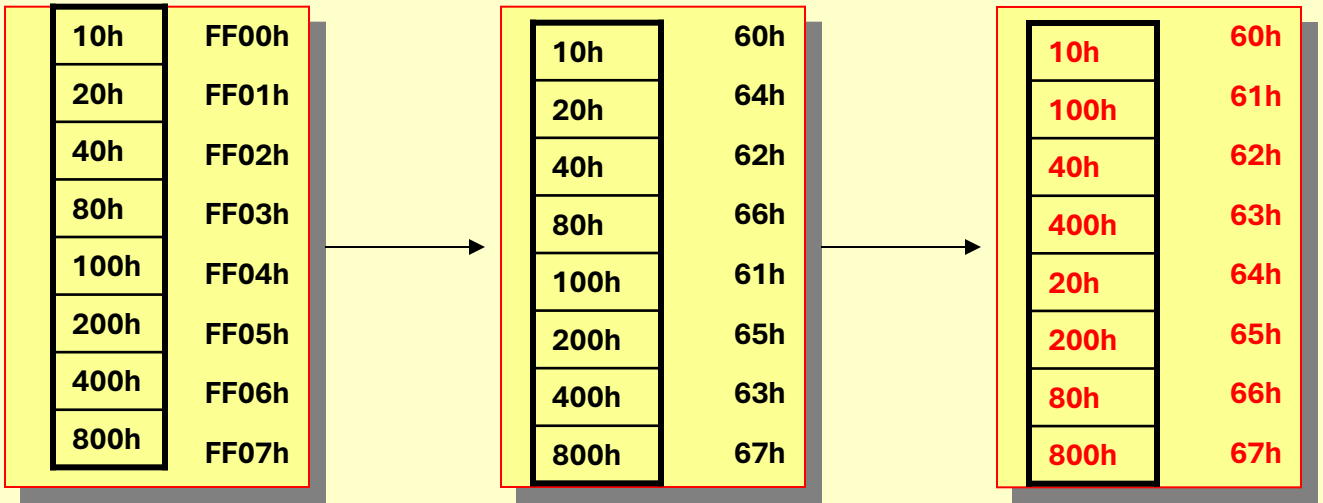
Bit-reversed addressing

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)

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

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

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



60h= 0110 0000+
0000 0100

64h= 0110 0100+
0000 0100

62h= 0110 0010+
0000 0100

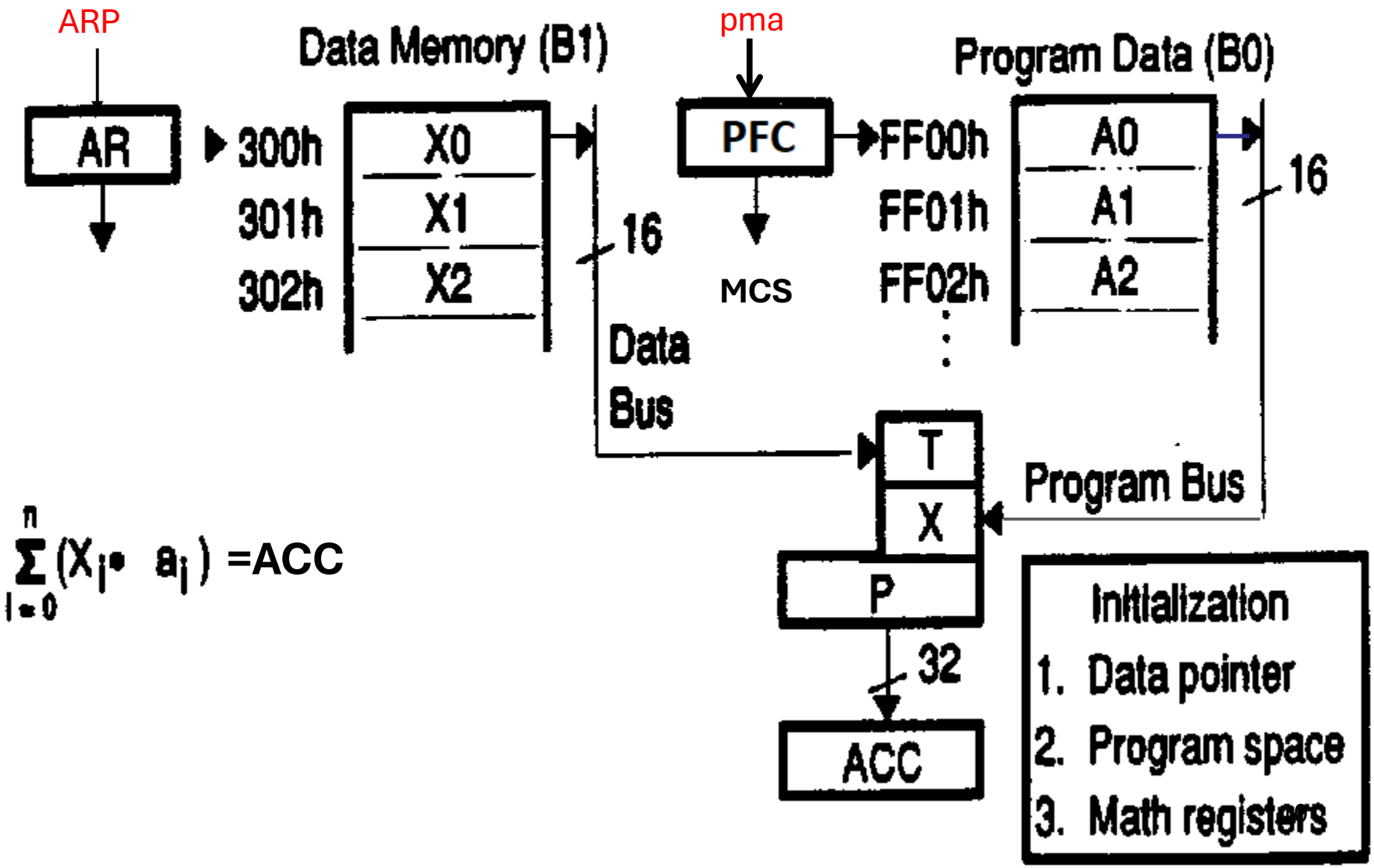
66h= 0110 0110 +
0000 0100

61h= 0110 0001+
0000 0100

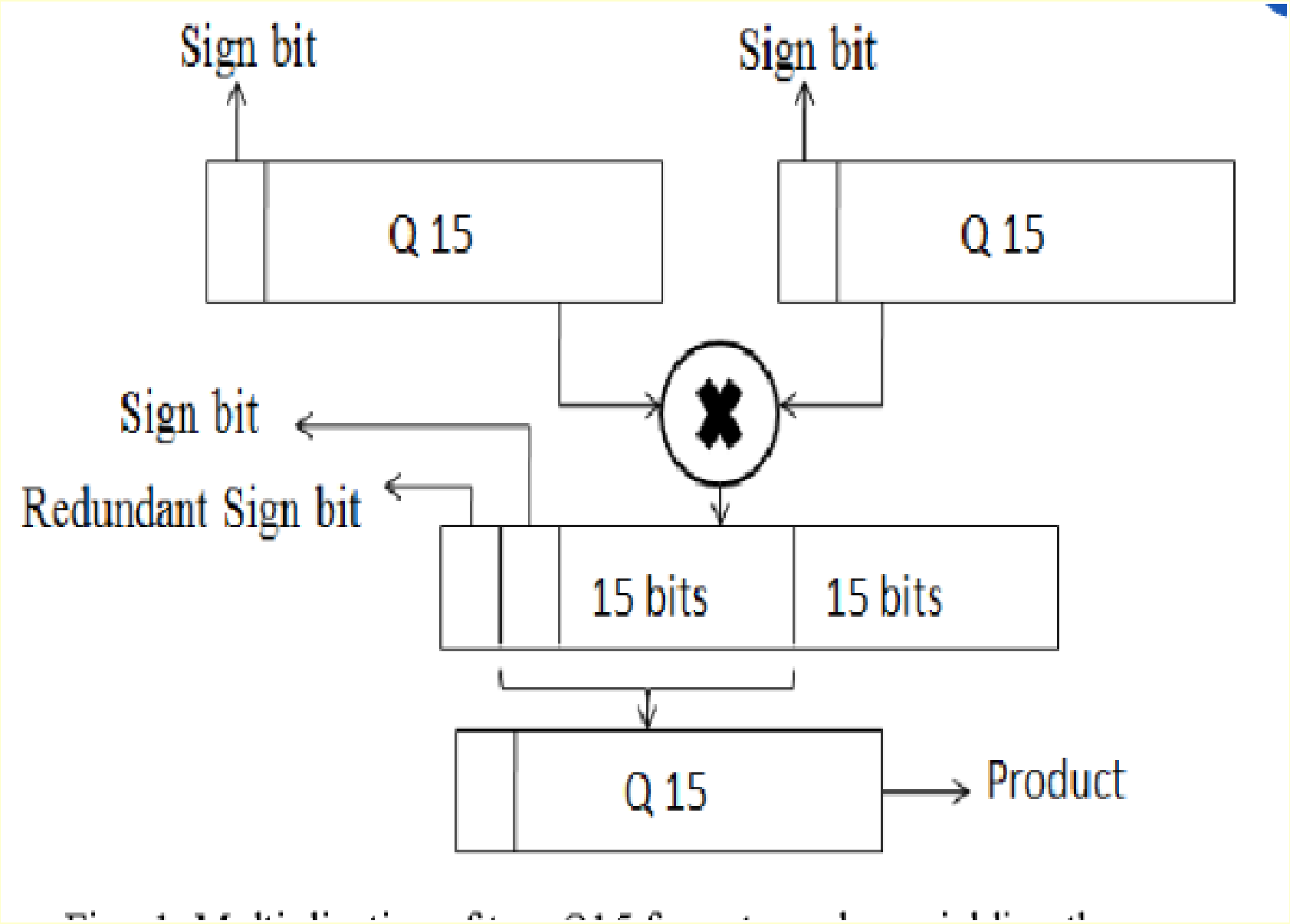
65h= 0110 0101...

Ex.4.

Multiply with Accumulate (MAC)



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


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

```
LARP  1              ; AR1 current AR register
LRLK   AR1,300h      ;AR1=300h
ZAC                  ;AC=0
CNFP                 ;B0 >> FF00h (Program Memory)
SPM  1               ;PM=1, product shifted 1 bit to left
RPTK   N             ;N<256
MAC    FF00,*+       ; => APAC
APAC                 ;     LT  *+    ;T=(AR)ARP
                     ;     MPY     ; (PC)
                     ;    (PFC)=PFC+1
                     ;     AR1=AR1+1
```

LARP 1	Sets **AR1** as the current auxiliary register (ARP = 1)
LRLK AR1, 300h	Loads AR1 with address **0x300h** (points to data RAM — the input samples or coefficients)
ZAC	Clears the **accumulator** (AC = 0) — fresh start for the MAC sum
CNFP	Reconfigures **B0 block** as program memory at **0xFF00h** — this is where the filter coefficients are stored
SPM 1	Sets **product shift mode = 1** → the multiplier output is **left-shifted 1 bit** (compensates for Q15 redundant sign bit)
RPTK N	Repeats the **next instruction N+1 times** (N < 256), so it runs the MAC N+1 times
MAC FF00h, *+	**The operation is:** load coefficient from FF00h (updates each repeat), multiply by data at *AR1, add P to AC, then increment *AR1.
APAC	Final **Add P to AC** — adds the last pending product to the accumulator

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**

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

BLKD – Move Mdata>>Mdata: BLKD 300h,*+ **B1** → **B0(data)**

(300h-307h) → (200h-207h)

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

BLKP FF00h,*BR0+ FF00h-FF07H → 60h-67h

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**

LDPK 0		0								DP=0
LAC 76h										Acc=800h
SUB 72h,1										Acc=0

5. 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**	**76h**	**72h**	**CNF**	**OBS.**
LARP 7			7							
LRLK AR6,70h					70h					
LRLK AR7,200h						200h				
RPTK 7										RPTC=8
BLKD 300h,*+										#
CNFP									1	
LRLK AR0,4				4						
LARP 6			6							
RPTK 7										RPTC=8
BLKP FF00h,*BR0+										#
LDPK 0		0								
LAC 76h	800h									
SUB 72h,1	0									

- Comments

Ex.6. Fill in the table below :

		ACC	DP	ARP	T	P	AR1	AR2	PM	100h	200h	201h	284h
	Setup values	800h				10h			1	110h	2	4	100h
1	LDPK 5												
2	LRLK AR1, 200h												
3	LRLK AR2, 100h												
4	LARP 1												
5	ZAC												
6	ADD 4, 2												
7	LT *+												
8	MPYA *, 2												
9	XORK 80h												
10	SUB *, 1, 1												

8. MPYA – multiply and acumulate precedent product shifted cf. PM bits

MPYA *,2 ; (ACC) = (ACC) + (P) precedent shifted left by 1 bit (PM=1) = *2; ARP=2
; 400h+10h*2 =420h
; (P) = (T) x (AR1)=80h

9. 0000 0100 0010 0000 b **xor**
0000 0000 1000 0000 b
0000 0100 1010 0000 b → 4A0h

10. (ACC)=(ACC)-(100h)*2; 4A0h-220h=280h, ARP=1

		ACC	DP	ARP	T	P	AR1	AR2	PM	100h	200h	201h	284h
	Valori initiale					10h			1	110h	2	40h	100h
1	LDPK 5		5										
2	LRLK AR1, 200h						200h						
3	LRLK AR2, 100h							100h					
4	LARP 1			1									
5	ZAC	0											
6	ADD 4, 2	400h											
7	LT *+				2		201h						
8	MPYA *, 2	420h		2		80h							
9	XORK 80h	4A0h											
10	SUB *, 1, 1	280h		1									

8. MPYA – multiply and acumulate precedent product shifted cf. PM bits

MPYA *,2 ; (ACC) = (ACC) + (P) precedent shifted left by 1 bit (PM=1) = *2; ARP=2
; 400h+10h*2 =420h
; (P) = (T) x (AR1)=80h

9. 0000 0100 0010 0000 b **xor**
0000 0000 1000 0000 b
0000 0100 1010 0000 b → 4A0h

10. (ACC)=(ACC)-(100h)*2; 4A0h-220h=280h, ARP=1

FIR on conventional DSP

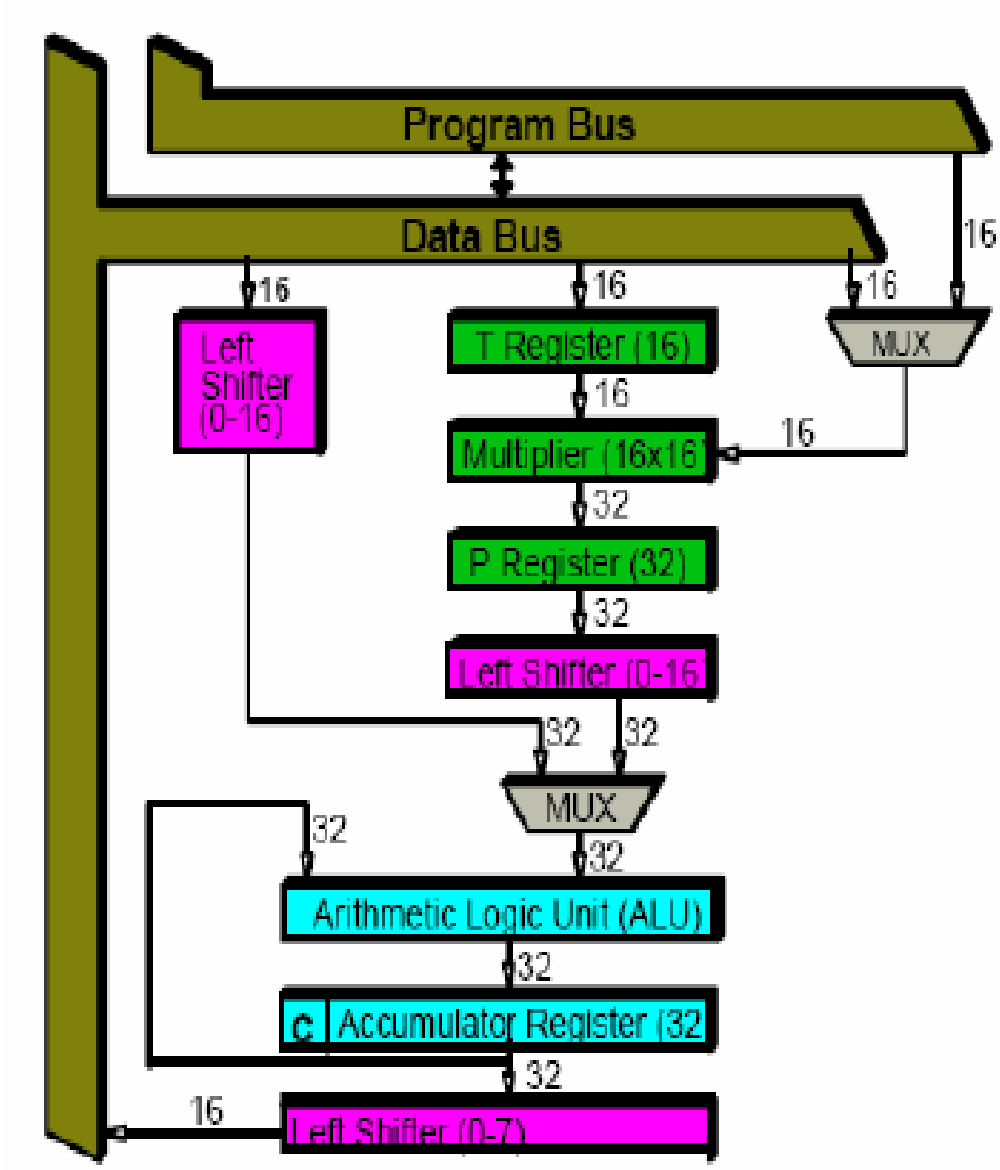
- TMS320C2x

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

RPTK N-1
MACD

APAC || MPY;
APAC;
```

Exécution en N cycles



C7. C2x Applications (cont.)

Ex.7.

FIR Filtering: A Motivating Problem

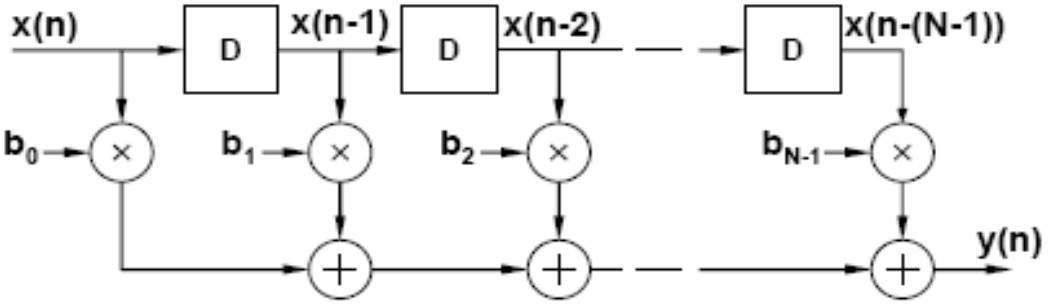
Transfer Function

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

Difference Equation

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

Block Diagram

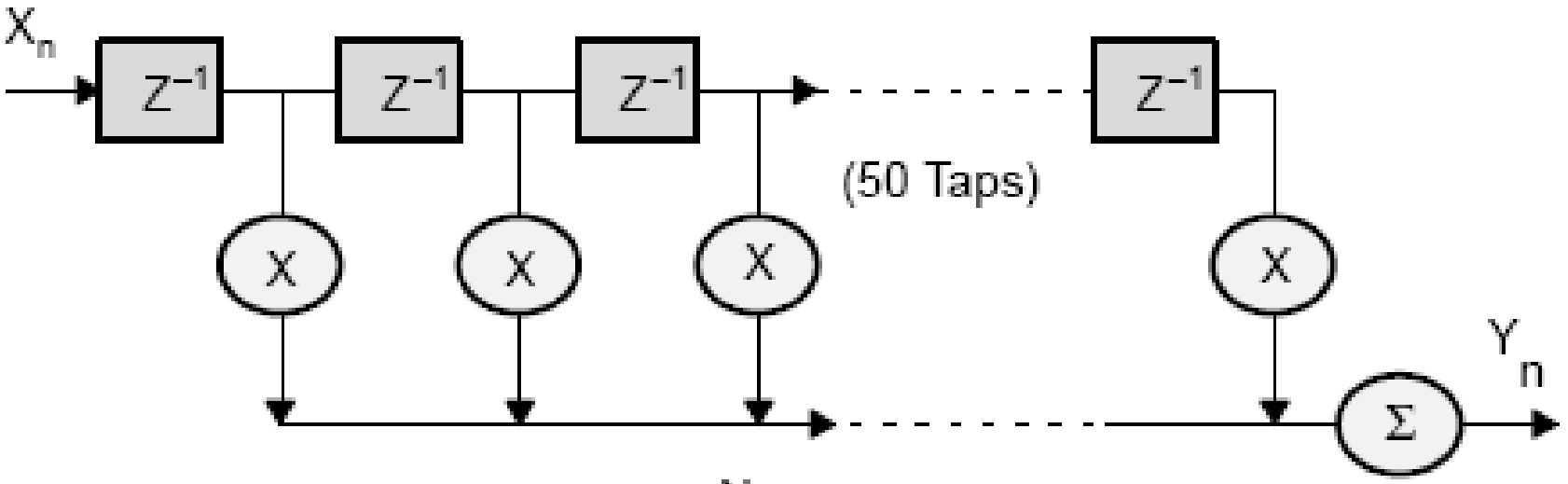


High-Level Language Description

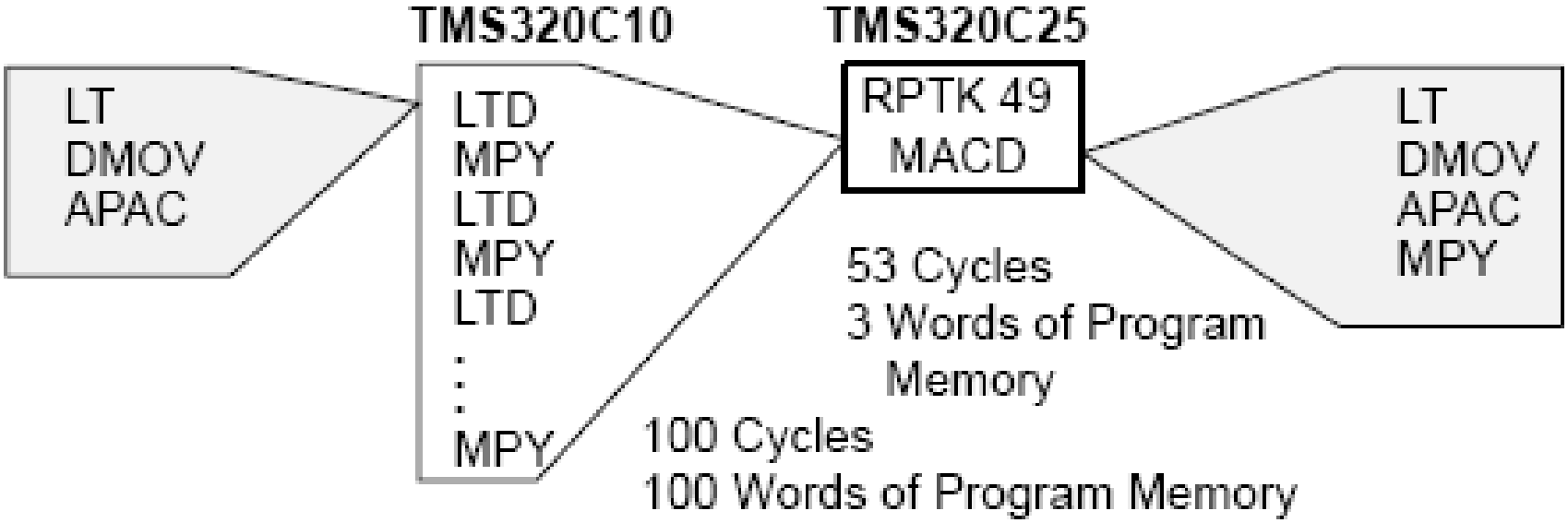
```
y = 0;x[0]=xin;
for(k=N-1; k>=0; k--) {
    y += x[k]*b[k];
    x[k]=x[k-1];
}
```

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

TMS320C25 Sum of Products Example

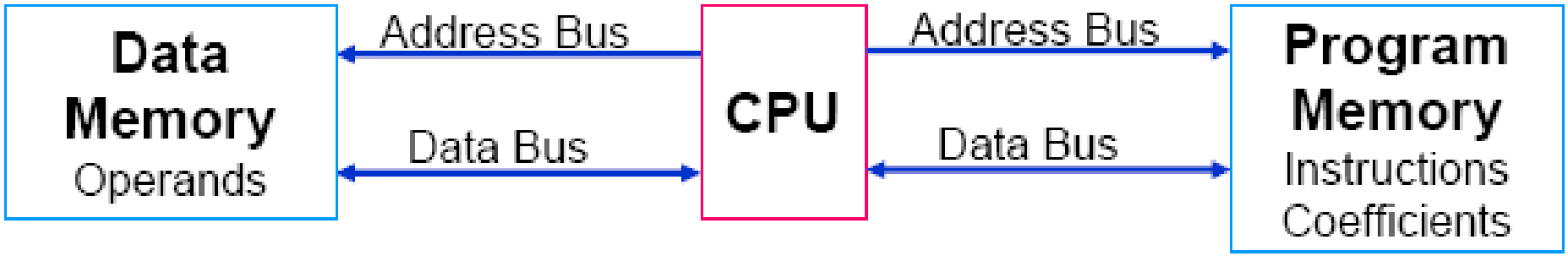


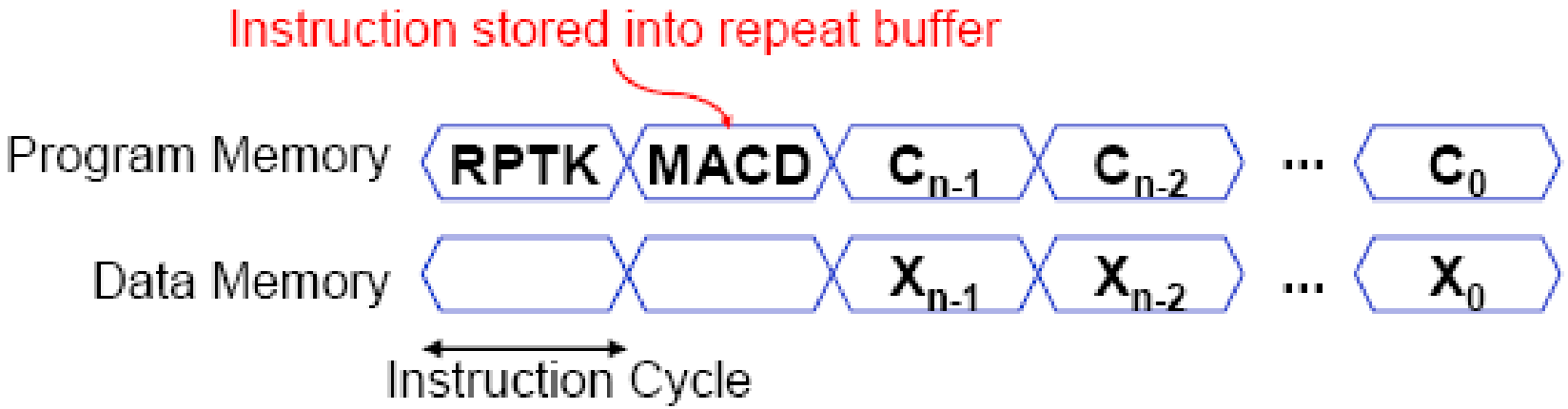
$$Y_n = \sum_{k=0}^{N} b_k \; X_{(n-k)}$$



Repeat Buffer in ‘C25

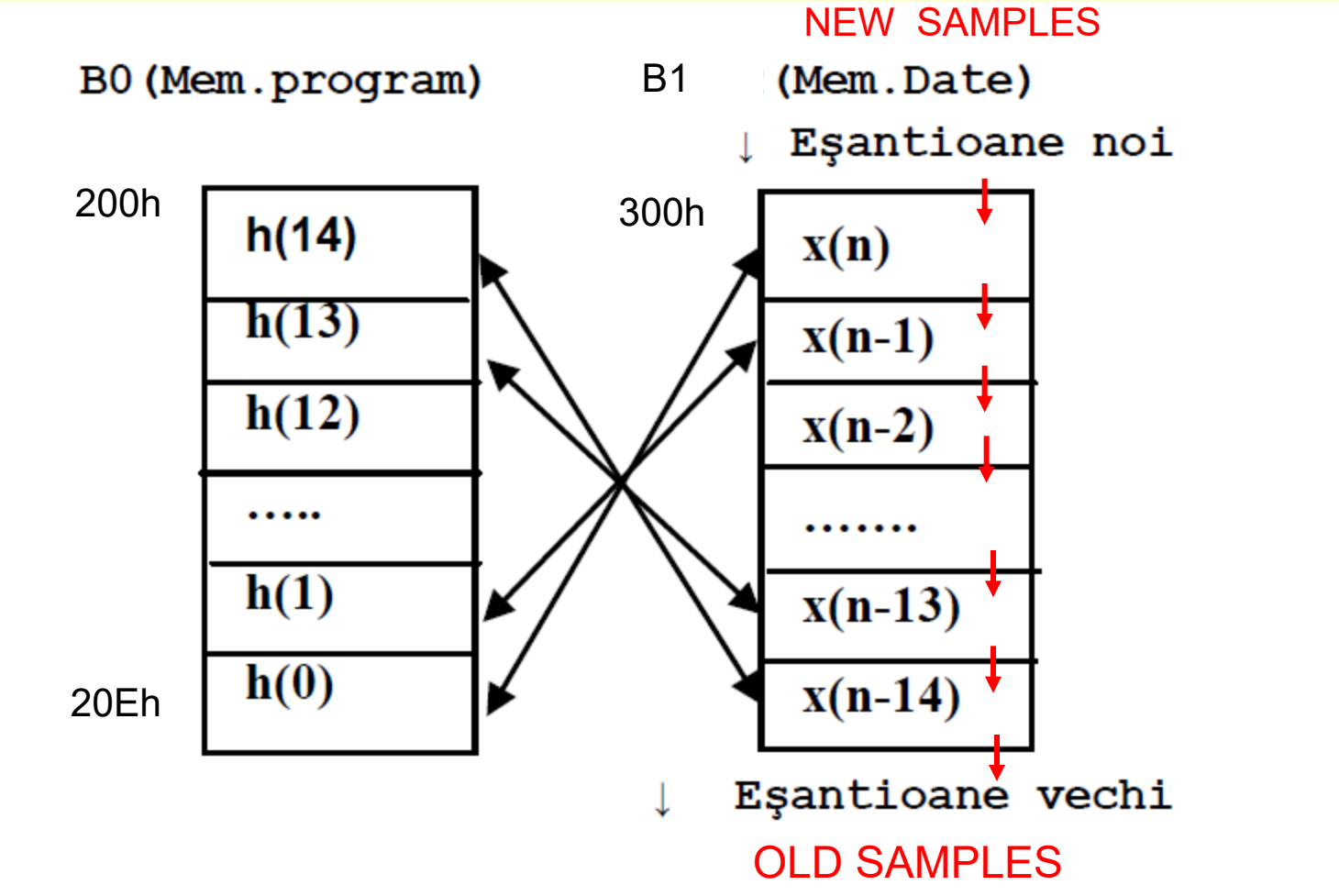
```
    RPTK    n-1          ; repeat the following
*                        ; instruction n times
    MACD    Xn-1,Cn-1    ; perform MAC with
*                        ; delay line update
```





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

N=15



b(0)= -8,99696 * 10^{-3}

Q15= b(0) * 2^{15} = -294,8 ~ -295 = FE9Dh (C2)

```
FIRCOEF
        data 1400h  ;h0
        data  2574h ;h1
        ……
        data  FE9Dh;h14
```

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

N=15

```
.....
LARP  0                 ; ARP=0, AR0- current register
CNFD                    ; B0 –M. Data
LRLK AR0, 20eh          ; coef. >>200h-20eh
RPTK   14               ; repeat 15 times next instr.
BLKP  FIRCOEF, *-       ; FIRCOEF >> B0
CNFP                    ; B0 >> M. Program (FF00-FF0Eh), h14,h13,...h1,h0
SPM 0                   ; PM=0
LARP 1                  ; ARP=1, AR1- current register
LRLK AR1, 30eh          ;AR1=30eh
MPYK 0                  ;T*0=P=0
ZAC                     ;ACC=0
RPTK  14                ;repeat 15x MACD
MACD  FF00h, *-         ;h(14)* x(1)+ h(13) * x(2)+ h(12)* x(3)+ ...   h(0)*x(15)
APAC                    ;ACC = ACC + h(0)*x(15)

.............
```

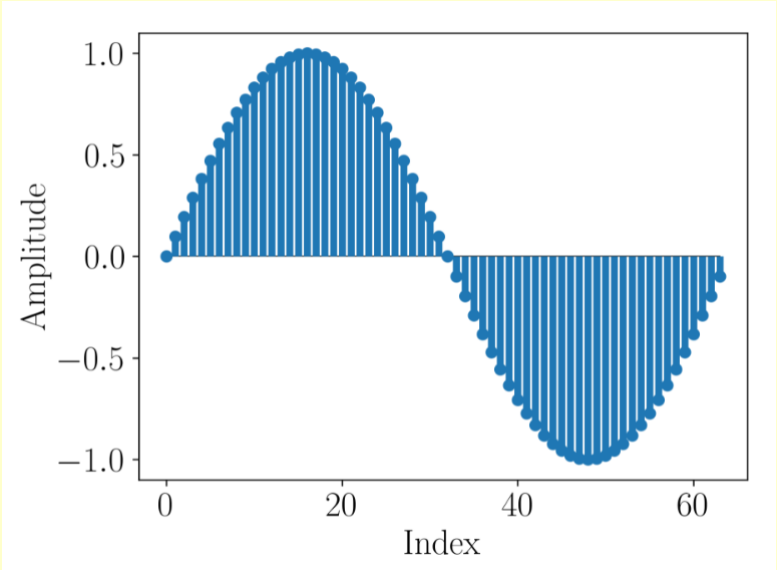
HW 1. What function implements the below program sequence (AL C2x) and where is stored the result and comments the instructions?

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

```
LDPK    4       ;
LARK    AR1,300h
LARP    1
ZAC
RPTK    99
ADD     *+
SACL    1h
SACH    0h
```

Ex.8 Signals generation methods – sine wave

Algorithm	Speed	Quality at low freq.	Accuracy	Stability	Orthogonality of sin/cos pair	Memory requirements	Modulation capability
Polynoms	Slow	Excellent	Good	Excellent	Excellent	Medium	Excellent
Lookup	Medium	Excellent	Fair	Excellent	Good	Huge	Good
Complex Oscillator	Fast	Limited	Good	Poor	Good	Low	Limited
2nd order oscillator	Fastest	Poor	Good	Excellent	Good	Low	Difficult



https://www.embedded.com/whats-your-sine-finding-the-right-algorithm-for-digital-frequency-synthesis-on-a-dsp/
https://thewolfsound.com/sound-synthesis/wavetable-synthesis-algorithm/

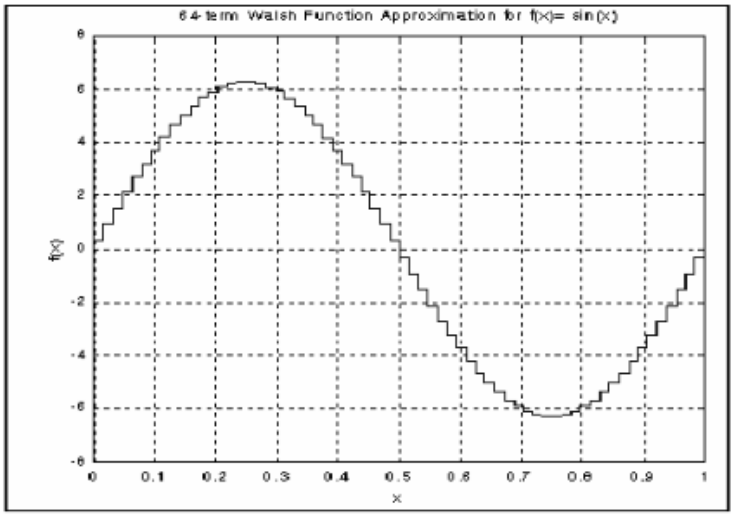


Figure 3. Walsh function approximated sine wave

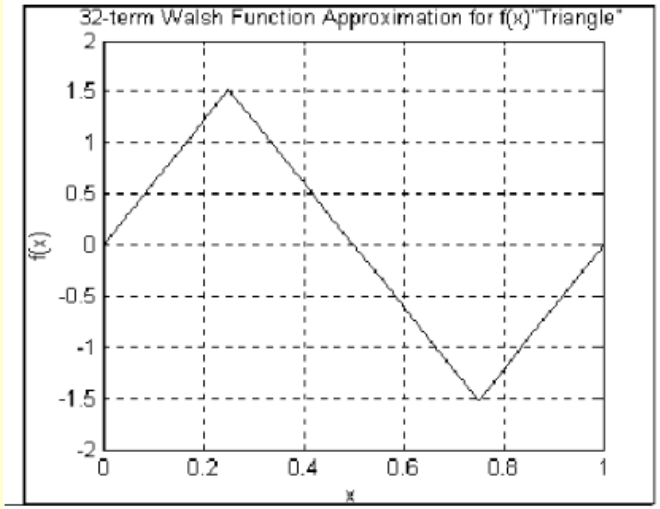


Figure 4. Walsh function approximated full triangular wave

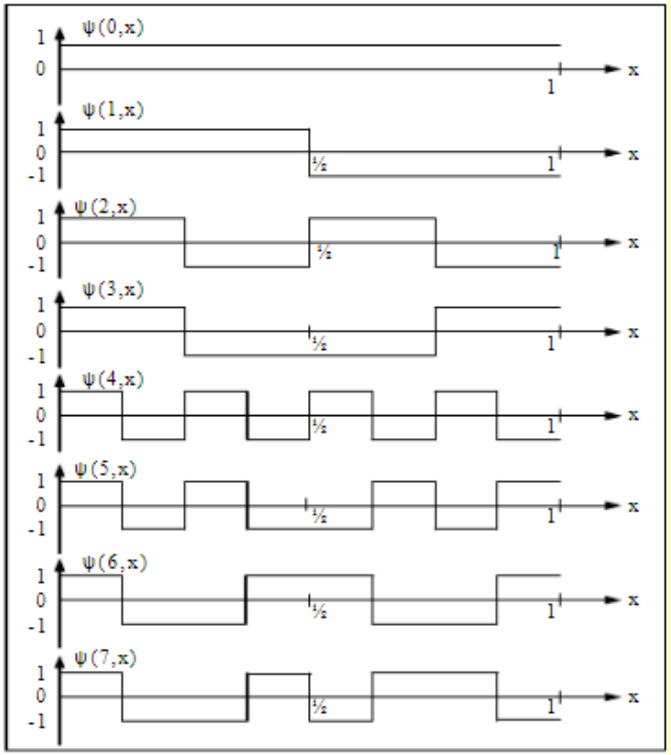


Figure 2. First eight continuous Walsh functions.

Sine Wave Generation — DSP (TMS320C2x)

Four methods:

- Look-up tables (requires $fs/fe = m, m \in \mathbb{N}$)
- Taylor series: $\sin x \approx x/1! - x^3/3! + x^5/5! - x^7/7! + \ldots$, for $0 \leqslant x \leqslant \pi/2$
- Walsh functions
- **Harmonic oscillator (recursive)** ← developed below

Derivation via Z-transform

The impulse response of a discrete sine is:

h(n) = sin(n·ωs·Te) = sin(n·θ) $y(n) = h(n) * x(n) = \sin(n\theta) * \delta(n)$

The input is a unit impulse: **x(n) = δ(n)**, so X(z) = 1, with θ = ωs·Te = 2π·fs/fe = 2π/n

The Z-transform of the output:

$$Y(z) = H(z) \cdot X(z) = \frac{z \sin\theta}{z^2 - 2z\cos\theta + 1} \cdot 1 = \frac{z^{-1}\sin\theta}{1 - 2z^{-1}\cos\theta + z^{-2}}$$

Rearranging:

$$Y(z) \cdot (1 - 2z^{-1}\cos\theta + z^{-2}) = z^{-1}\sin\theta \cdot X(z)$$

Taking the inverse Z-transform:

$$y(n) - 2\cos\theta \cdot y(n-1) + y(n-2) = \sin\theta \cdot x(n-1)$$

Recursive formula (corrected sign):

$$y(n) = \underbrace{\sin\theta}_{b_0} \cdot x(n-1) + \underbrace{2\cos\theta}_{a_1} \cdot y(n-1) \underbrace{-1}_{a_2} \cdot y(n-2), \quad n = 0, 1, 2, \ldots$$

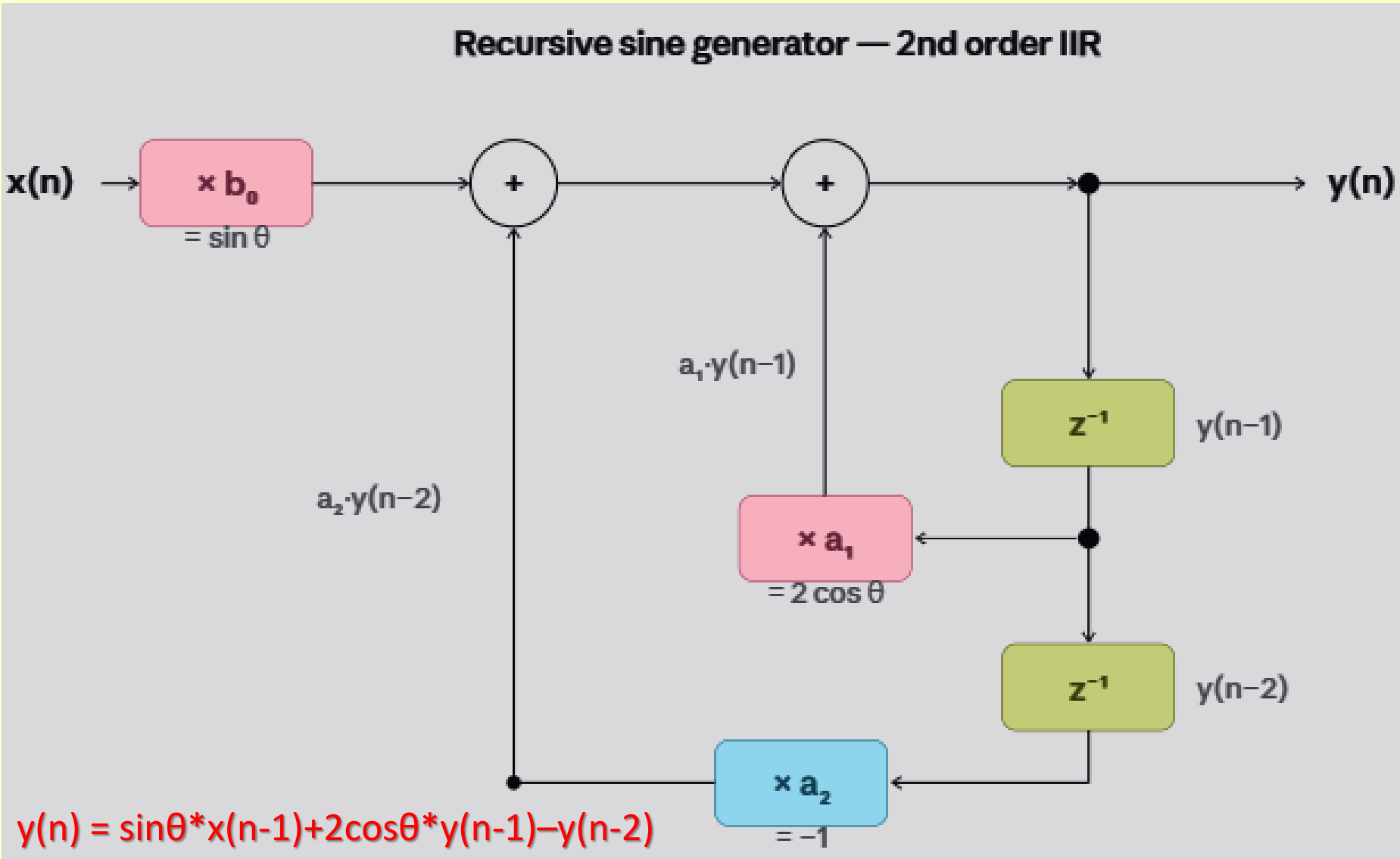
Initial conditions: $y(-2) = y(-1) = 0, x(0) = 1, x(n) = 0$ for $n > 0$

Verification:

- $y(0) = \sin\theta \cdot x(-1) + 2\cos\theta \cdot y(-1) - y(-2) = 0$ ✓
- $y(1) = \sin\theta \cdot x(0) + 2\cos\theta \cdot y(0) - y(-1) = \sin\theta$ ✓
- $y(2) = \sin\theta \cdot 0 + 2\cos\theta \cdot \sin\theta - 0 = \sin(2\theta)$ ✓

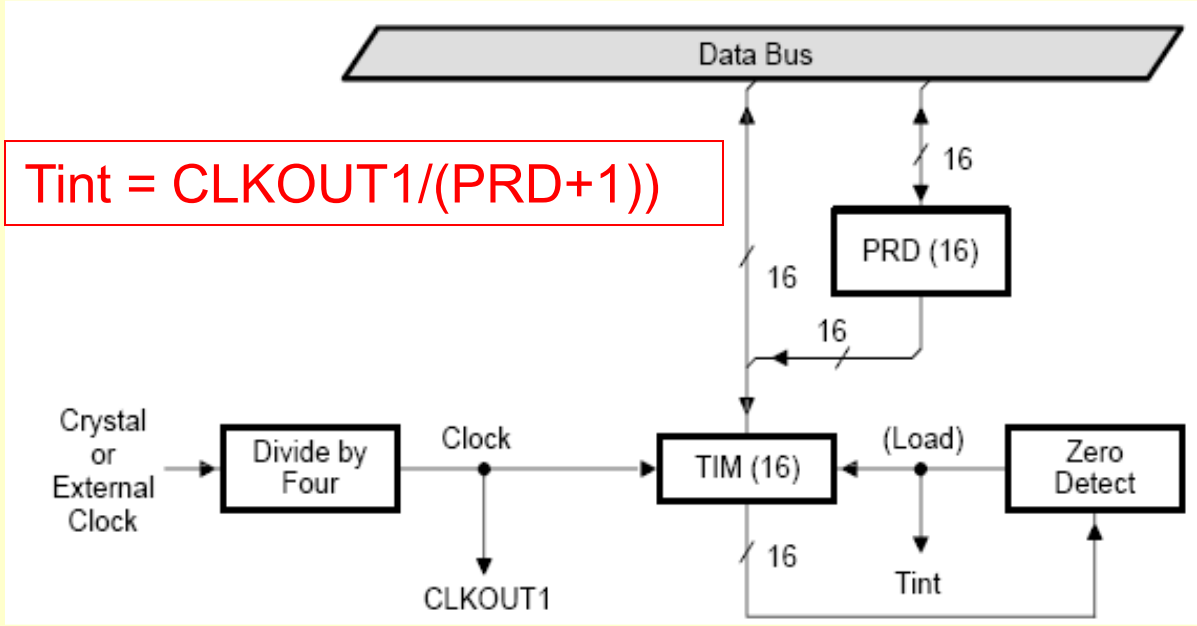
Example Parameters

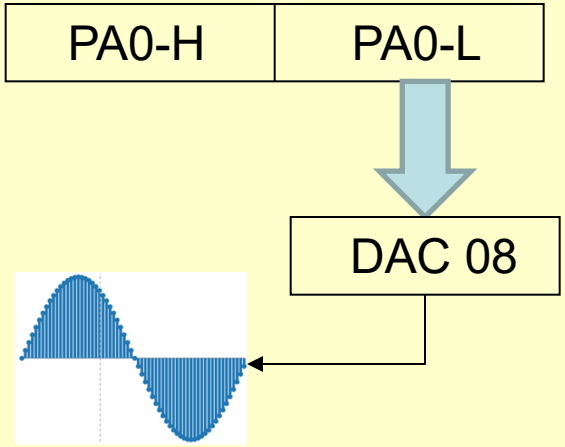
Parameter	Value
fe (sampling freq.)	40 kHz
fs (signal freq.)	1 kHz
n = fe/fs	40 samples/period
θ = 360°/n	9°
fclk	20 MHz
PRD = fclk/(4·fe) − 1	124



$b_0 = \sin\theta$ (forward gain) $a_1 = 2\cos\theta$ (first feedback) $a_2 = -1$ (second feedback)

$\theta = 2\pi / n$ $n = fe / fs$ Initial: $y(-1) = y(-2) = 0$





Interrupt Name	Memory Location	Priority	Function
$\overline{\text{RS}}$	0h	1 (highest)	External reset signal
$\overline{\text{INT0}}$	1h	2	External user interrupt #0
$\overline{\text{INT1}}$	2h	3	External user interrupt #1
INT2	3h	4	External user interrupt #2
	8–17h		Reserved locations
TINT	18h	5	Internal timer interrupt
RINT	1Ah	6	Serial port receive interrupt
XINT	1Ch	7 (lowest)	Serial port transmit interrupt
TRAP	1Eh	N/A	TRAP instruction address

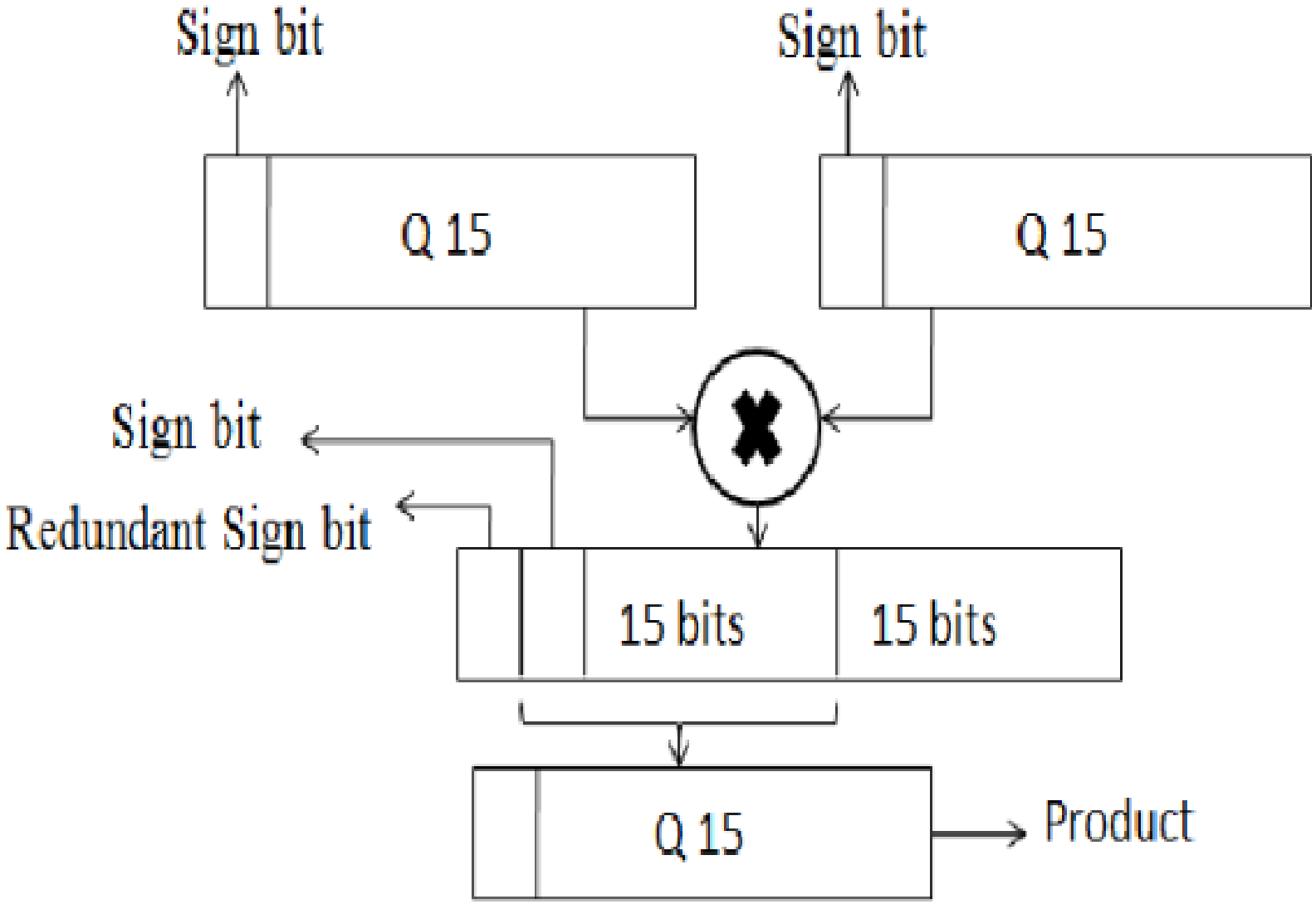
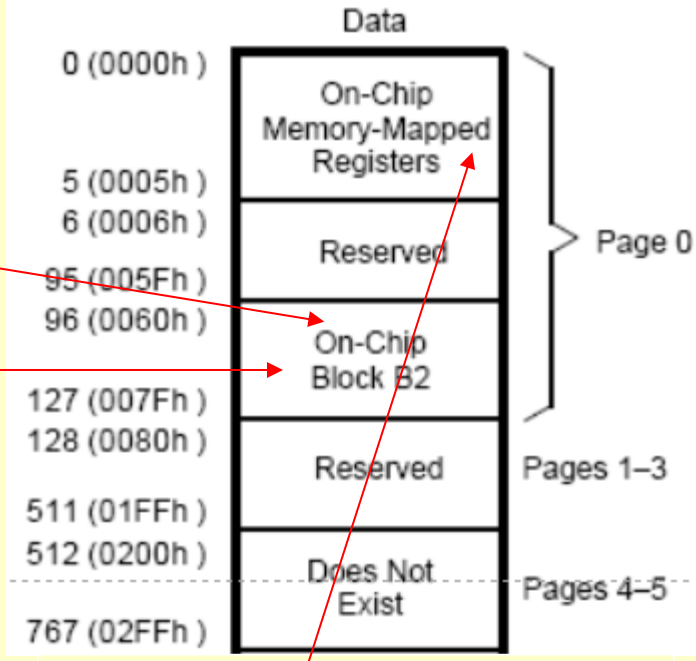
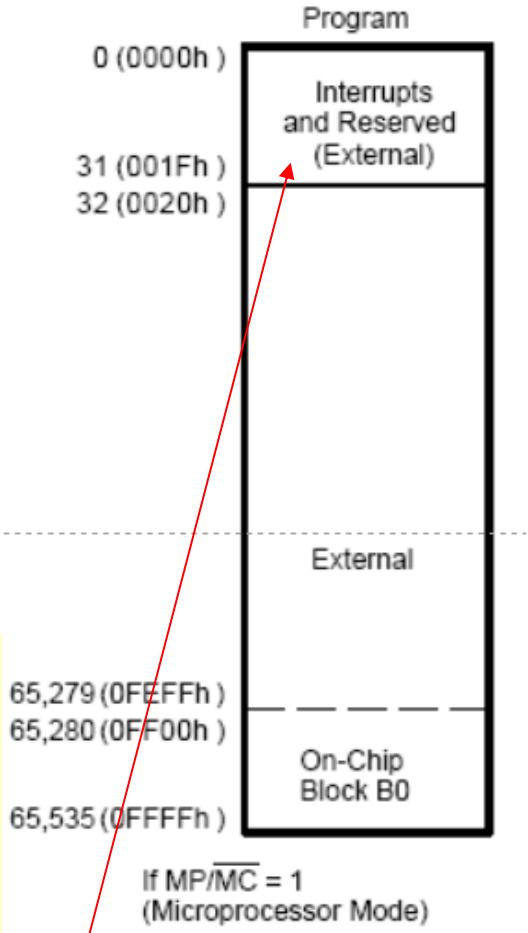


Fig. 1. Multiplication of two Q15 format numbers yielding the product in Q15 format itself.

```
fclk    equ 20000   ; kHz
 fe     equ 40; sampling feq.
PERD    equ fclk/(4*fe)-1; 124
COS     EQU 60h  ; cos 9°
yN      EQU 70h ;
yN_1    EQU 71H;
yN_2    EQU 72H;
 aorg 0       ;reset vector
LALK    perd    ; Accl=PRD
SACL    3       ;PRD= Accl
LALK  8     ; interrupt mask for TINT
              ; Accl=0008h
OR 4          ; ACC=ACC U (IMR)
SACL 4        ; (IMR) =Accl
b  start      ;jump to start
 aorg 18h    b  TINT_ISR
```





Register Name	Address Location
DRR(15–0)	0
DXR(15–0)	1
TIM(15–0)	2
PRD(15–0)	3
IMR (5–0)	4
GREG(7–0)	5

Interrupt Name	Memory Location
$\overline{RS}$	0h
$\overline{INT0}$	1h
$\overline{INT1}$	2h
$\overline{INT2}$	3h
	8–17h
TINT	18h
RINT	1Ah
XINT	1Ch
TRAP	1Eh

Interrupt Mask Register (IMR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED										XINT	RINT	TINT	$\overline{INT2}$	$\overline{INT1}$	$\overline{INT0}$

$$y(n) = \sin\theta * x(n-1) + 2\cos\theta * y(n-1) - y(n-2)$$

```
start:  spm    1                ; PM=01, ACC=ACC+2*P
        rovm                    ; normal, OVM=0
        ssxm                    ; sign extension, SXM=1
        lalk   7e6dh            ; ACCL= cos 9° in Q15
        sacl   COS              ; COS=ACCL = cos 9° in Q15
        lalk   1350h            ; ACCL= sin 9° in Q15
        sacl   yN_1             ; yN_1= ACCL= sin 9°
        zac                     ; ACC=0
        sacl   yN_2             ; yN_2= 0
        eint
loc:    b loc                   ; wait interrupt
        end
```

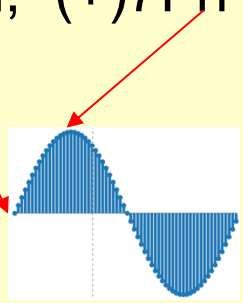
$$y(n) = \sin\theta\, x(n-1) + 2\cos\theta\, y(n-1) - y(n-2),\ n>1$$

$$y(0) = \sin\theta \cdot x(-1) + 2\cos\theta \cdot y(-1) - y(-2) = 0 \checkmark$$

$$y(1) = \sin\theta \cdot x(0) + 2\cos\theta \cdot y(0) - y(-1) = \sin\theta \checkmark$$

$$y(2) = \sin\theta \cdot 0 + 2\cos\theta \cdot \sin\theta - 0 = \sin(2\theta) \checkmark$$

```
TINT_ISR:
        lt      COS      ; T0 reg = cos 9° in Q15 ; B2 addr. 60h
        mpy     yN_1     ; P reg = cos 9° * y(n-1) in Q30
        pac              ; ACC = 2 * cos 9° * y(n-1) in Q30 (PM=1)
        sub     yN_2,15; ACC = ACC - y(n-2) in Q30
        dmov    yN_1     ; y(n-2) = y(n-1)
        sach    yN_1,1   ; y(n-1) = new y in Q15
        rptk    6        ;executes SFR 7 times → ÷128 → DAC range [0, 255]
        sfr              ;ACCH=ACCH/128
        adlk    128,16   ;scaling  (-) 80h =0,  00=80h,  (+)7Fh=FFh;
                         ;C2-  shifted binary code
        sach    yN       ;ACCH= yN
        out     yN, PA0  ; output new value of yN
        nop
        eint
        ret
```



$$y(n) = \cancel{\sin\theta * x(n-1)} \cancel{+}2*\cos\theta *y(n-1)–y(n-2) ,n>1$$

After SACH, ACC = $y(n) \times 2^{30}$. Tracing the scaling for the DAC:

Instruction	ACCH content	y(n)=−1	y(n)=0	y(n)≈+1
Before SFR	$y(n) \times 2^{14}$	−16384 = 0xC000	0	+16383 = 0x3FFF
After **8** SFR (rptk 7)	y(n) × **64**	−64	0	+63
+ adlk 128	y(n)×64+128	**64**	128	**191**
After **7** SFR (**rptk 6**)	y(n) × **128**	−128	0	+127
+ adlk 128	y(n)×128+128	**0** ✓	128 ✓	**255** ✓

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		0												
2	LRLK AR3, 60h						60h								
3	LRLK AR4, 70h							70h							
4	LARP 3			3											
5	LALK 20h	20h													
6	SUB *,1	18h													
7	LTA *+	38h			4		61h								
8	MPY *, 4			4		20h									
9	ROR	1Ch													
10	MAC FF00h,70h	3Ch				40h									
11	PAC	7Ch													
12	SUB *, 1	3Ch													

		ACC	DP	ARP	T	P	AR2	AR3	PM	C	64h	65h	70h	210h	FF20h	CNF
	Valori initiale					20h			00	0	0Ah	4h	0Dh	2h	6h	1
1	LDPK 4		4													0.2
2	LRLK AR2, 64h						64h									0.2
3	LRLK AR3, 70h							70h								0.2
4	LARP 2			2												0.2
5	LALK 32h	32h														0.2
6	SUB *+, 1	1Eh					65h									0.5
7	LTA *-	3Eh			4		64h									0.5
8	MPY *, 3			3		28h										0.75
9	MAC FF20h, 10h	66h				0Ch										0.75
10	ADD*-, 2	9Ah						6Fh								0.5

		ACC	DP	ARP	T	P	AR2	AR3	PM	C	64h	65h	70h	210h	FF20h	CNF
	Valori initiale					20h			00	0	0Ah	4h	0Dh	2h	6h	1
1	LDPK 4															
2	LRLK AR2, 64h															
3	LRLK AR3, 70h															
4	LARP 2															
5	LALK 32h															
6	SUB *+, 1															
7	LTA *-															
8	MPY *, 3															
9	MAC FF20h, 10h															
10	ADD*-, 2															

Ex. 10. What function implements the program sequence below, and where does it store the result?

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

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

Q15 Format & DAC08 Offset Binary Conversion

Q15 Fixed-Point Format (16-bit)

Bit 15 (MSB)	Bits 14 … 0
Sign bit (0=+, 1=−)	15 fractional bits
Value range:	−1.0 … +0.9999695
Resolution:	2^{-15} = 0.0000305

Q15 Value	Hex	Meaning
+1.0 (clipped)	7FFFh	Maximum positive
+0.996 (≈+A)	7F00h	Peak tri/rect HIGH
0.0	0000h	Zero crossing
−0.996 (≈−A)	8100h	Trough / rect LOW
−1.0	8000h	Maximum negative
0.1 (step)	0CCCh	Triangular slope

Scaling: Q15 → 8-bit Offset Binary for DAC08

```
; ACC contains y(n) in Q15 (16-bit signed)
; Goal: convert to 8-bit unsigned [0..255]
;   −A (8100h) → 00h = 0
;    0 (0000h) → 80h = 128
;   +A (7F00h) → FFh = 255
    rptk   6    ; repeat next instr 7 times
    sfr  ; ACC >>= 1  (total: divide by 128)
    ; ACC[31:16] = ACCH = y(n)/128
    adlk   128, 16 ; ACCH += 128 (offset shift)
    sach   yN_OUT  ; store ACCH as byte
    out    yN_OUT, PA0
```

Q15 hex	÷128	+128	DAC out
7F00h (+A)	+127	+128	FFh = 255
0000h (0)	0	+128	80h = 128
8100h (−A)	−127	+128	01h = 1
8000h (−1)	−128	+128	00h = 0

SPM=1 (in sine): NOT needed here — no multiply-accumulate used