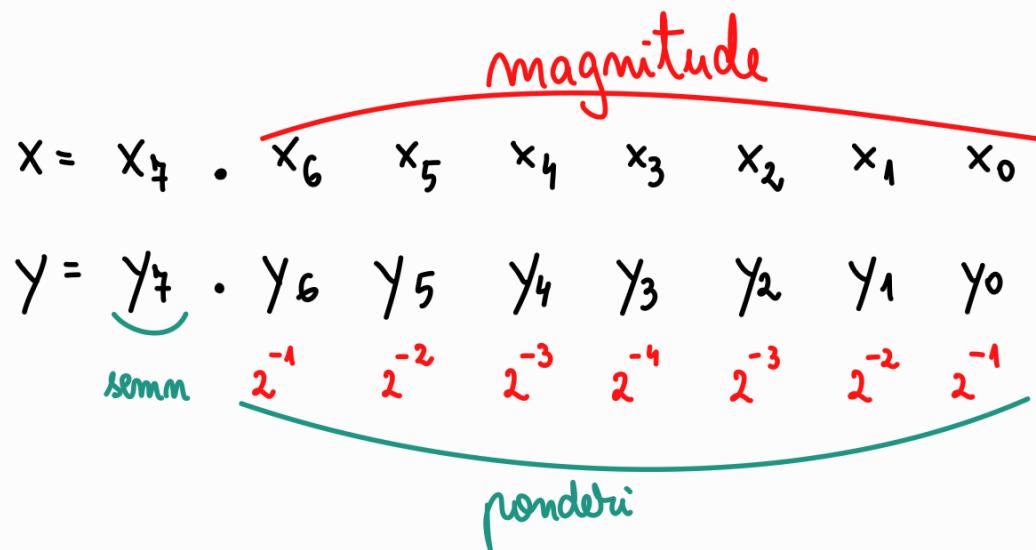


## 4.2. Înmulțire binară secvențială pentru numere în Semm-Mătrime

Fie  $x, y$  - 2 numere fractionare, pe 8 biti, reprezentate în SM



Producătorul  $P = x \cdot y$  este un nr. fractionar în SM, pe 16 biti.

$$P = p_{15} \cdot p_{14} \ p_{13} \dots \ p_2 \ p_1 \ p_0$$

- ▷ MSB este semnul  $p_{15} = x_7 \oplus y_7$
- ▷ părțiile de magnitudine ale operandelor au 7 biti  $\Rightarrow$  magnitudinea produsului va avea 14 biti:  $p_{14}, p_{13}, \dots, p_1$
- ▷ al 16-lea bit este  $p_0$  și are valoarea 0

→ În sistemele de calcul folosim un nr. de biti multiplu de 8

## ALGORITM

multiplier 2

declare register A[7:0], Q[7:0], M[7:0], COUNT[7:0];  
declare bus INBUS[7:0], OUTBUS[7:0];

înmulțitor

deînmulțit

ceil(log<sub>2</sub> m)

m - nr. biti

BEGIN: A := 0, COUNT := 0, }  
INPUT: M := INBUS;

Semnal de control C<sub>0</sub>

Q := INBUS; → C<sub>1</sub>

TEST1: if Q[0] = 0 then go to RSHIFT,

ABB : A[7:0] := A[6:0] + M[6:0]; → C<sub>2</sub>  
fără bit de semn

RSHIFT: A[7]:=0, A[6:0].Q := A.Q[7:1], } → C<sub>3</sub>  
INCR : COUNT := COUNT + 1;

TEST2: if COUNT ≠ 1 then go to TEST1,

SIGN: A[7] := Q[0] exor M[7], Q[0]:= 0; → C<sub>4</sub>

OUTPUT: OUTBUS := A; → C<sub>5</sub>

OUTBUS := Q; → C<sub>6</sub>

ENB : - - - - - - - - - - - - - - - - - ENB

→ la m înțeleg mai facem o shiftare

Eлементe de pseudolimbaj

### ① declare registers

- ▶ definește registri; specifică
  - ↳ numele, S/I
  - ↳ lățimea
- ▶ operatorul de concatenare: •; ex. A[6 : 0].Q := A.Q[7 : 1]

### ② declare bus

- ▶ definește magistrale specificând
  - ↳ numele, S/I
  - ↳ lățimea

### 3. Execuție sincronă

- ↳ operații non-conflictuale: executate concurențial, separate prin  $\sqcap$
- ↳ operații secvențiale: executate secvențial, separate prin  $\sqcup$

4. Operatorul de atribuire este  $\sqcup =$  și este folosit pentru încărcarea de valori binare în registri sau magistrale
- ▶ exor indică o operație cablată: în  $A[7] := Q[0]$  exor  $M[7]$  este implementată printr-o poartă EXOR

### 5. Controlul fluxului de execuție

- ↳ salt necondiționat: go to TEST1
- ↳ salt condiționat: if  $Q[0] = 0$  then go to RSHIFT

### 6. Citire/scriere simultană din/în registre

$$A[7] := Q[0] \text{ exor } M[7], Q[0] := 0$$

- ▶ bit-ul  $Q[0]$  este atât citit cât și scris în același ciclu de ceas

► În raport cu frontul declansător al semnalului de tact, citirea se realizează înainte



### Elementele platformei hardware

- ▶ acumulatorul  $A$ : este folosit pentru adunarea produselor de 1-bit la produsul parțial; are facilități de deplasare la dreapta; stochează biții mai semificativi ai produselor parțiale
- ▶ registrul înmulțitor  $Q$ : încărcat, inițial, cu înmulțitorul  $X$ ; are facilități de deplasare la dreapta (numele  $Q$  provine de la platforma HW specifică împărțirii binare)
- ▶ registrul deînmulțit  $M$ : stochează deînmulțitul  $Y$ ;
- ▶ sumatorul paralel: pe 7 biți; utilizat pentru adunarea produselor de 1-bit la produsul parțial
- ▶ contorul  $COUNT$ : păstrează evidența numărului de iterații executate
- ▶ unitatea de control generează semnalele de control ( $c_0, \dots, c_6$ ,  $END$ ) în secvența corectă de execuție a algoritmului

Algoritmul folosește metoda a 3-a de înmulțire (păstrarea fiecărui produselor de 1 bit):

$$\begin{cases} P_i = P_{i-1} + x_i \cdot y \\ P_{i+1} = P_i \cdot 2^{-1} \end{cases} \longrightarrow \begin{array}{l} \text{etichetele TEST 1 și Abb} \\ \text{eticheta RSHIFT} \end{array}$$

Produsele parțiale:

- ▶ până la setarea semnului rezultatului, (eticheta **SIGN**), toate produsele parțiale sunt pozitive, reprezentate în S.-M., independent de semnele operanzilor
- ▶ la începutul algoritmului,  $P_0$  ocupă registrul  $A$  ( $P_0 := 0$ )
- ▶ după prima iterare,  $P_1$  ocupă întreg registrul  $A$  și MSB-ul lui  $Q$  ( $P_1$  este în  $A[7 : 0].Q[7]$ )
- ▶ după a doua iterare,  $P_2$  ocupă întreg  $A$ -ul și primii 2 MSBs ai lui  $Q$  ( $P_2$  este în  $A[7 : 0].Q[7 : 6]$ )
- ▶ în general,  $P_{i+1}$  ocupă un bit suplimentar în  $Q$  la fiecare nouă iterare; aceasta se petrece la eticheta **RSHIFT**, unde  $A$  concatenat cu  $Q$  este deplasat la dreapta

În fiecare iterare, registrul  $Q$  este deplasat la dreapta:

- ▶ la începutul algoritmului,  $Q[0]$  conține pe  $x_0$
  - ▶ după prima iterare,  $Q[0]$  conține pe  $x_1$
  - ▶ după a doua iterare,  $Q[0]$  conține pe  $x_2$
- ⇒ în oricare iterare, bitul curent al lui  $X$ ,  $x_i$ , se află în  $Q[0]$ .  
Acesta este motivul pentru care la **TEST1**, algoritmul testează bitul  $Q[0]$ .

## 1. Sumator paralel:

- ▶ pe 7 biți pentru că produsul parțial  $P_i$  și deînmulțitorul  $Y$  sunt în S.-M. iar în S.-M. adunarea nu poate calcula semnul rezultatului în mod corect
- ▶ transportul de ieșire al adunării este stocat în  $A[7]$ 
  - ▶ ⇒ se evită **overflow**-ul

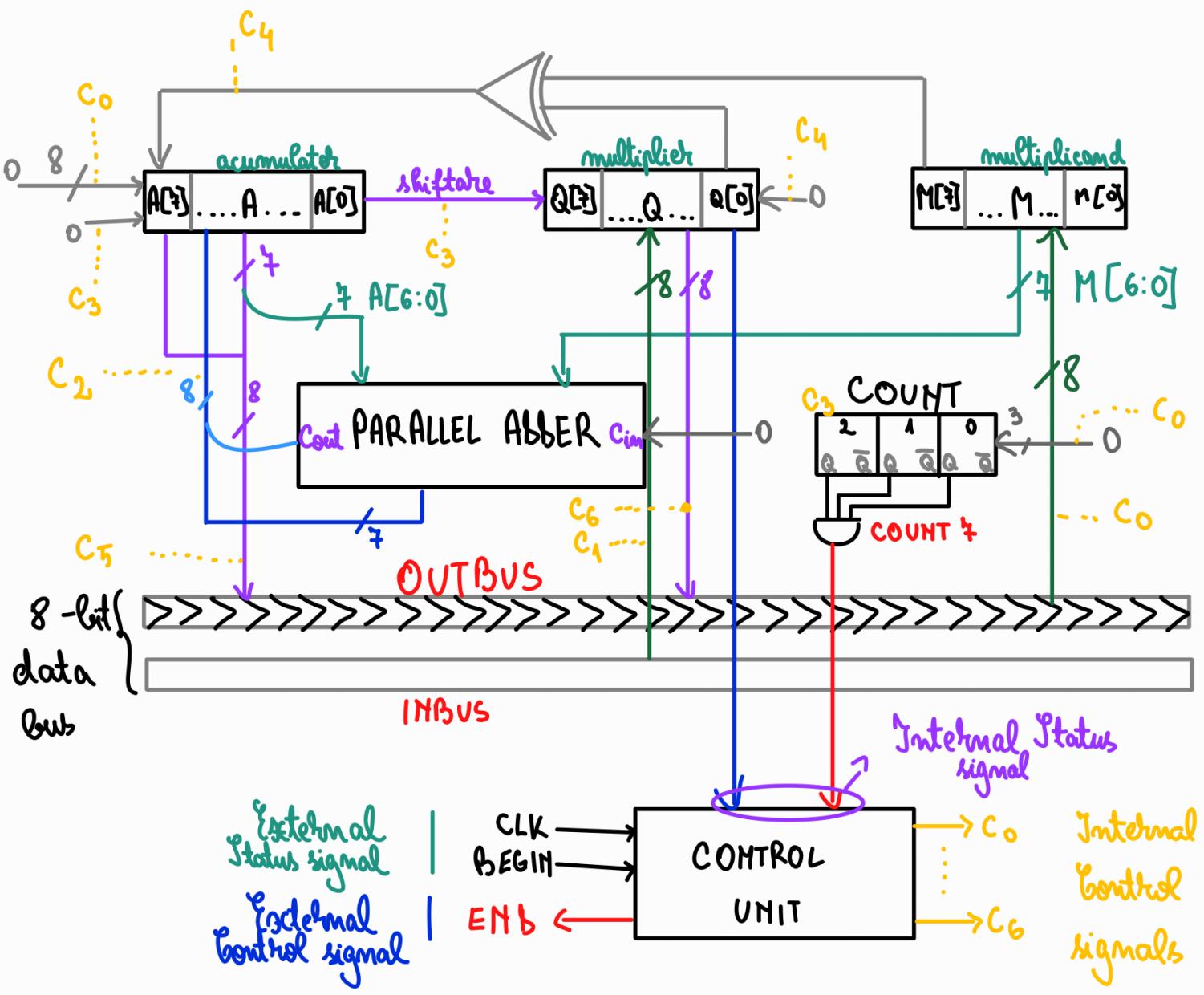
## 2. Contorul:

- ▶ numără 7 iterări:
  - ▶ pentru că magnitudinea înmulțitorului  $X$  are o lățime de 7 biți
- ▶ incrementat la eticheta **RSHIFT**
- ▶ semnalul **COUNT7** este activat când conținutul numărătorului este 7 ( $7_{(10)} = 111_{(2)}$ )

## 3. Unitatea de control:

- ▶ secvențiază operațiile prin activarea semnalelor de control ( $c_0, \dots, c_6, END$ )
- ▶ în fiecare ciclu de tact, cel puțin un semnal de control este activat
- ▶ semnalele de control sunt reprezentate prin linie întreruptă  

- ▶ liniile de date sunt reprezentate prin linie solidă —



$C_{out} \rightarrow A[7]$

Exercitii

① operații fractionare

→ 4 biti, SM

$$x = -0.625$$

$$y = -0.875$$

$$\begin{array}{r|l} 0.625 & \cdot 2 \\ 1.250 & \cdot 2 \\ 0.500 & \cdot 2 \\ 1.000 & \end{array}$$

$$\begin{array}{r|l} 0.875 & \cdot 2 \\ 1.750 & \cdot 2 \\ 1.500 & \cdot 2 \\ 1.000 & \end{array}$$

+ bit de semn

$$\Rightarrow x = -101 \cdot 2^{-3} = -5 \cdot 2^{-3} = 1101_{SM}$$

$$y = -111 \cdot 2^{-3} = -7 \cdot 2^{-3} = 1111_{SM}$$

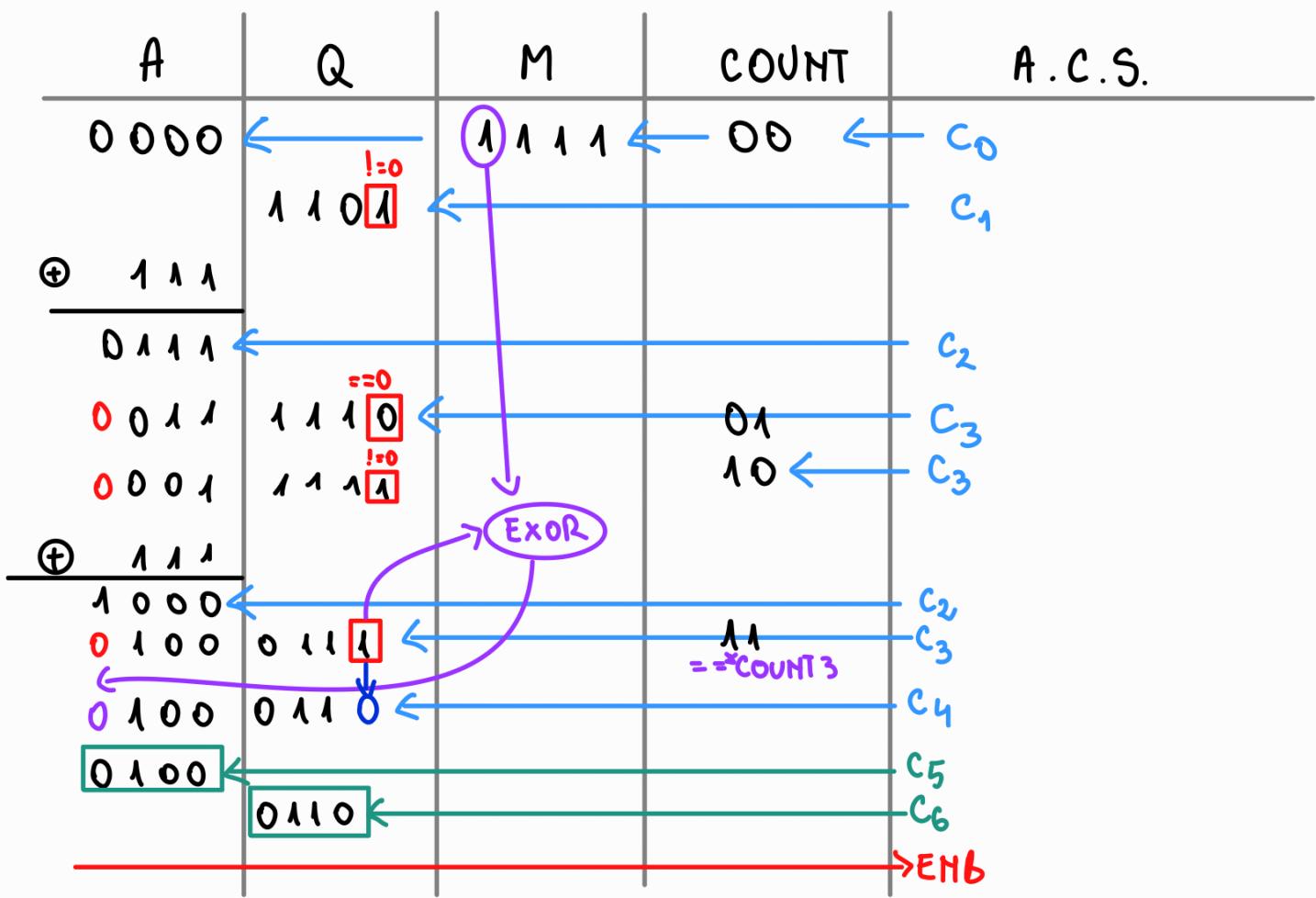
Calcul  $P = x \cdot y = (-5 \cdot 2^{-3}) \cdot (-7 \cdot 2^{-3}) = 35 \cdot 2^{-6}$

35	:	2
17	:	2 1
8	:	2 1
4	:	2 0
2	:	2 0
1	:	2 0
0		1

$$\Rightarrow 35 = \frac{100011}{32 \quad 21} \Rightarrow 0.1000110 = 35$$

ca să avem  
8 biti

$$\Rightarrow P = 0.1000110_{SM} \quad \text{dim M se adună fără bit de semn}$$



## ② operații întregi

→ 4 biți, întregi

$$x = -6 = 1\ 110 \text{ SM}$$

$$y = +7 = 0\ 111 \text{ SM}$$

$$P = x \cdot y = -6 \cdot 7 = -42 = 1\ 0101010 \text{ SM}$$

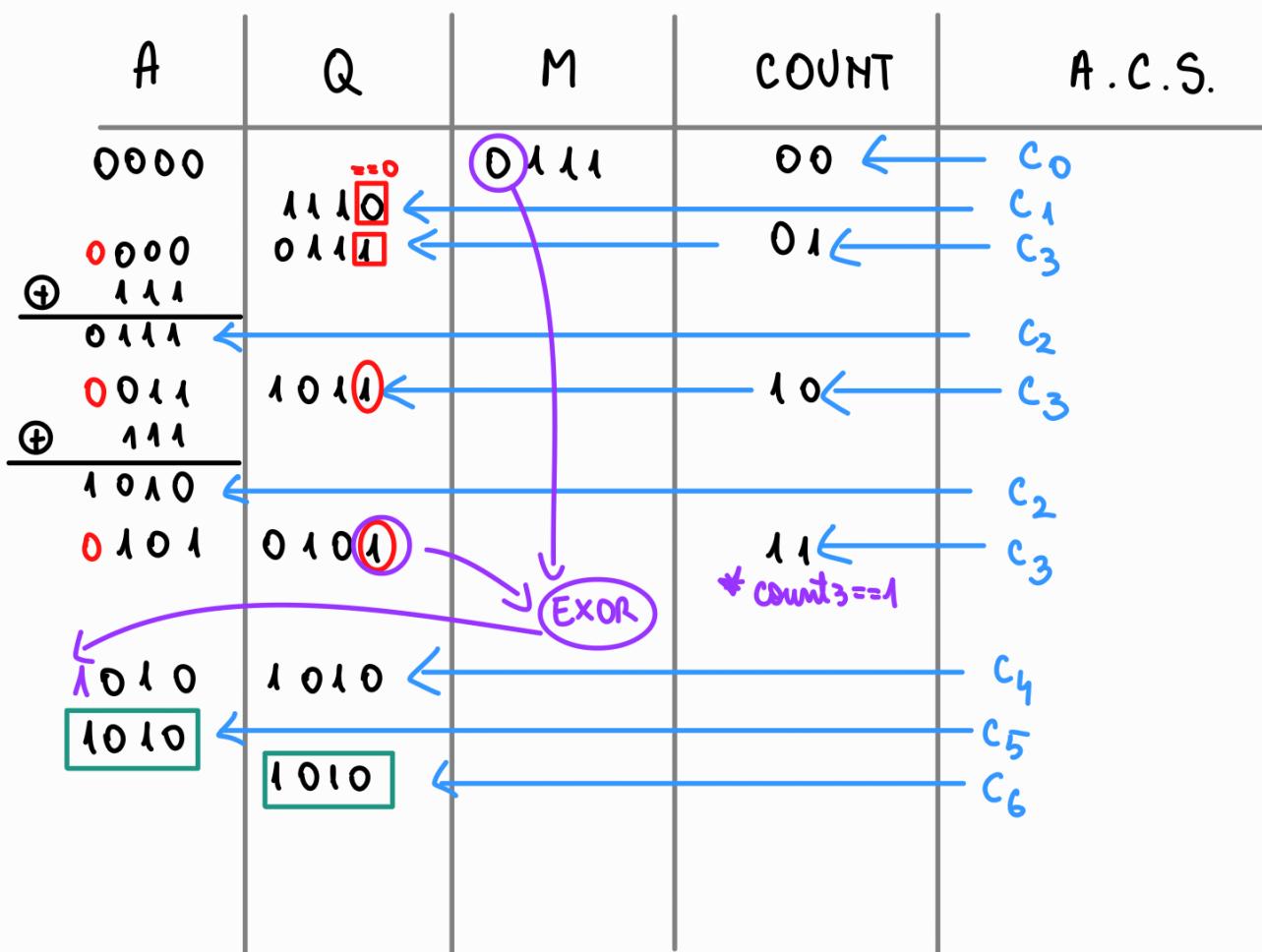
42	2							
21	2	0						
10	2	1						
5	2	0						
2	2	1						
1	0							

⇒  $-42 = 1\ 0101010 \text{ SM}$

sign

pt. a ajunge la 8 biți

32 8 2



③ fractionare  $\xrightarrow{\text{COUNT}=4}$

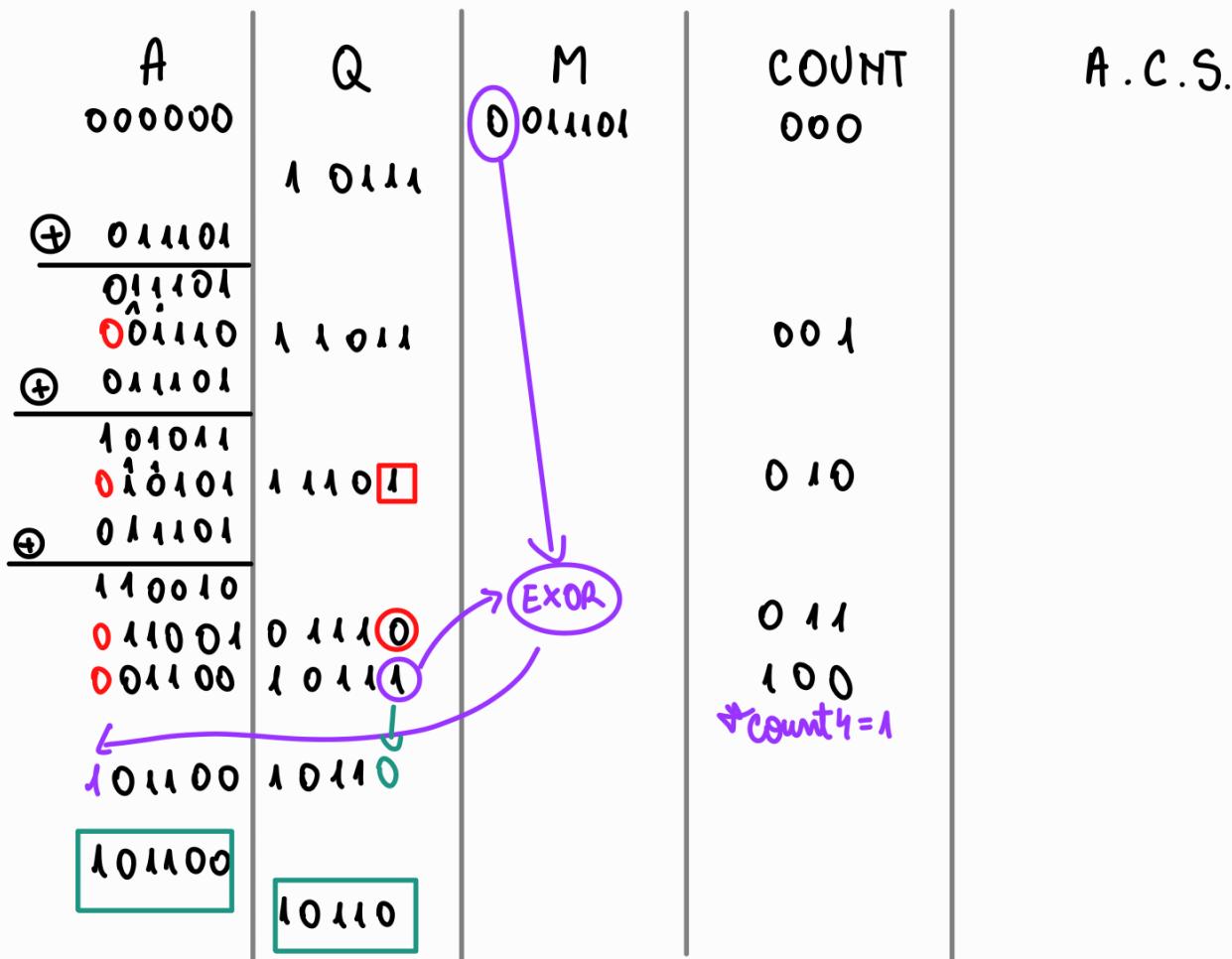
$x - 5\text{bit}$

$y - 6\text{bit}$

$$x = -7 \cdot 2^{-4} = 10111_{\text{SM}}$$

$$y = +29 \cdot 2^{-5} = 0011101_{\text{SM}}$$

$$P = -7 \cdot 29 \cdot 2^{-9} = -203 \cdot 2^{-9} \Rightarrow 10\text{ bits}, \\ = 111001011 \cdot 2^{-9}$$



$$P = 1.011100 10110 \cancel{X} = 11001011 \cdot 2^{-9}$$

$P_M$        $P_S$        $\downarrow S$

④ → mti. integri

$x$  - 4 bits  
 $y$  - 4 bits

$$x = +13 = 0\ 1101$$

$$y = -11 = 1\ 1011$$

$$P = -13 \cdot 11 = 143 \longrightarrow P = \boxed{\beta_9} \cdot \beta_8 \cdots \beta_1 \boxed{\beta_0}$$

$$= 0\ 10001111$$

Sign

A	Q	M	COUNT	A.C.S.
00000	01101	11011	000	
$\oplus$ 1011				
01011				
00101	10110		001	
00010	11011		010	
$\oplus$ 1011				
01101				
00110	11101		011	
$\oplus$ 1011				
10001				
01000	11110		100	
00100	01111		= count 4	
00100	01111			
$P = \boxed{S} 0010001111$				