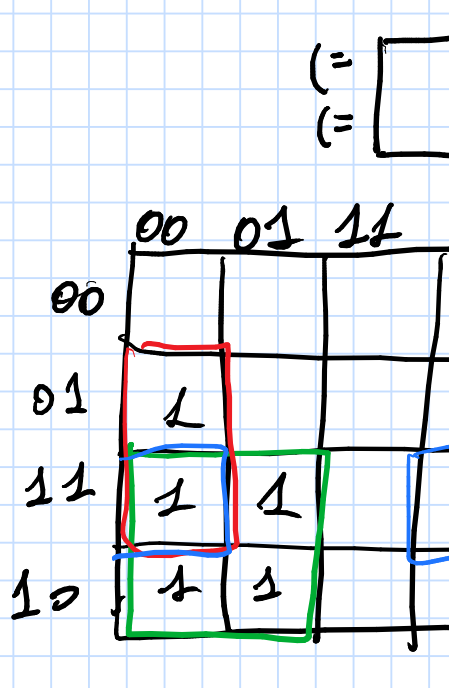
Reti logica 4 variabili in e 1 variabile out

$z = x_1x_2 > x_3x_4$

$x_1x_2x_3x_4$	z
0000	0
0001	0
0010	0
0011	0
0100	1
0101	0
0110	0
0111	0
1000	1
1001	1
1010	0
1011	0
1100	1
1101	1
1110	1
1111	0

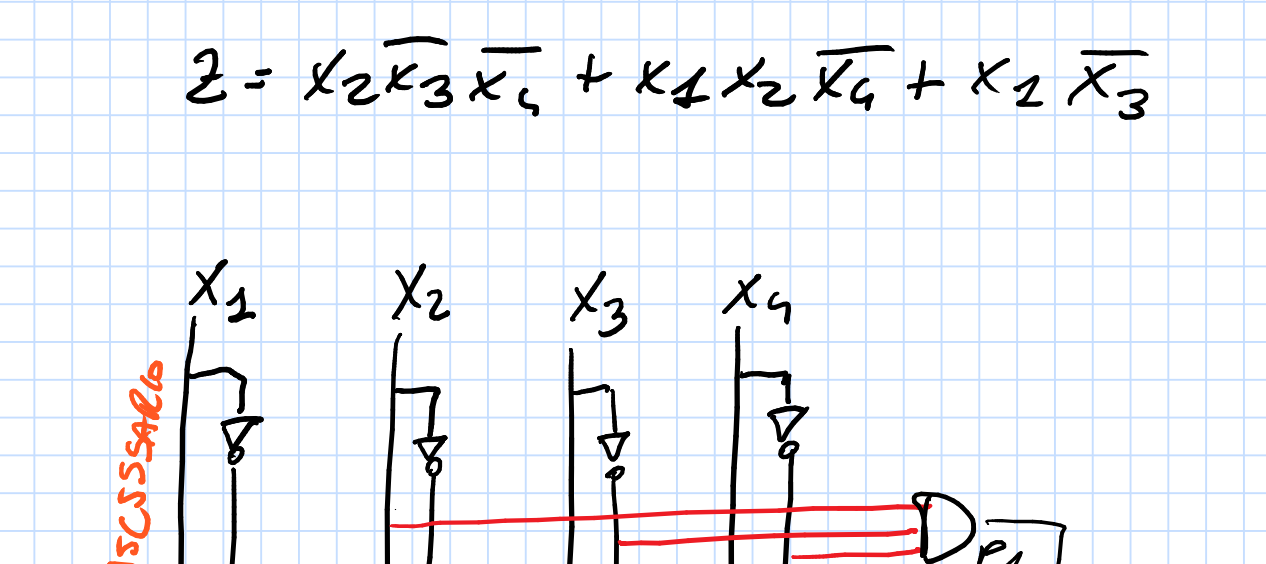


$P_1 = x_2\overline{x_3}\,\overline{x_4}$

$P_2 = x_1x_2\overline{x_4}$

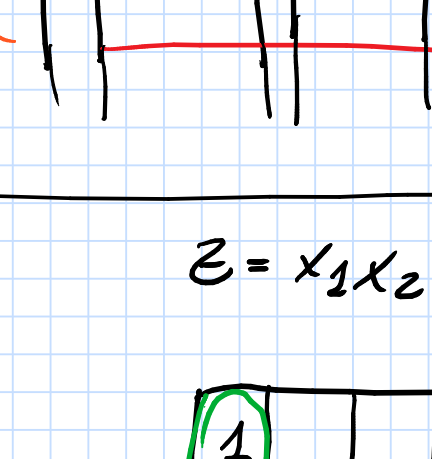
$P_3 = x_1\overline{x_3}$

$z = x_2\overline{x_3}\,\overline{x_4} + x_1x_2\overline{x_4} + x_1\overline{x_3}$



$z = x_1x_2 \geq x_3x_4$

$x_1x_2x_3x_4$	z
0000	1
0001	0
0010	0
0011	0
0100	1
0101	1
0110	0
0111	0
1000	1
1001	1
1010	1
1011	0
1100	1
1101	1
1110	1
1111	1



$P_1 = \overline{x_3}\,\overline{x_4}$

$P_2 = x_1x_2$

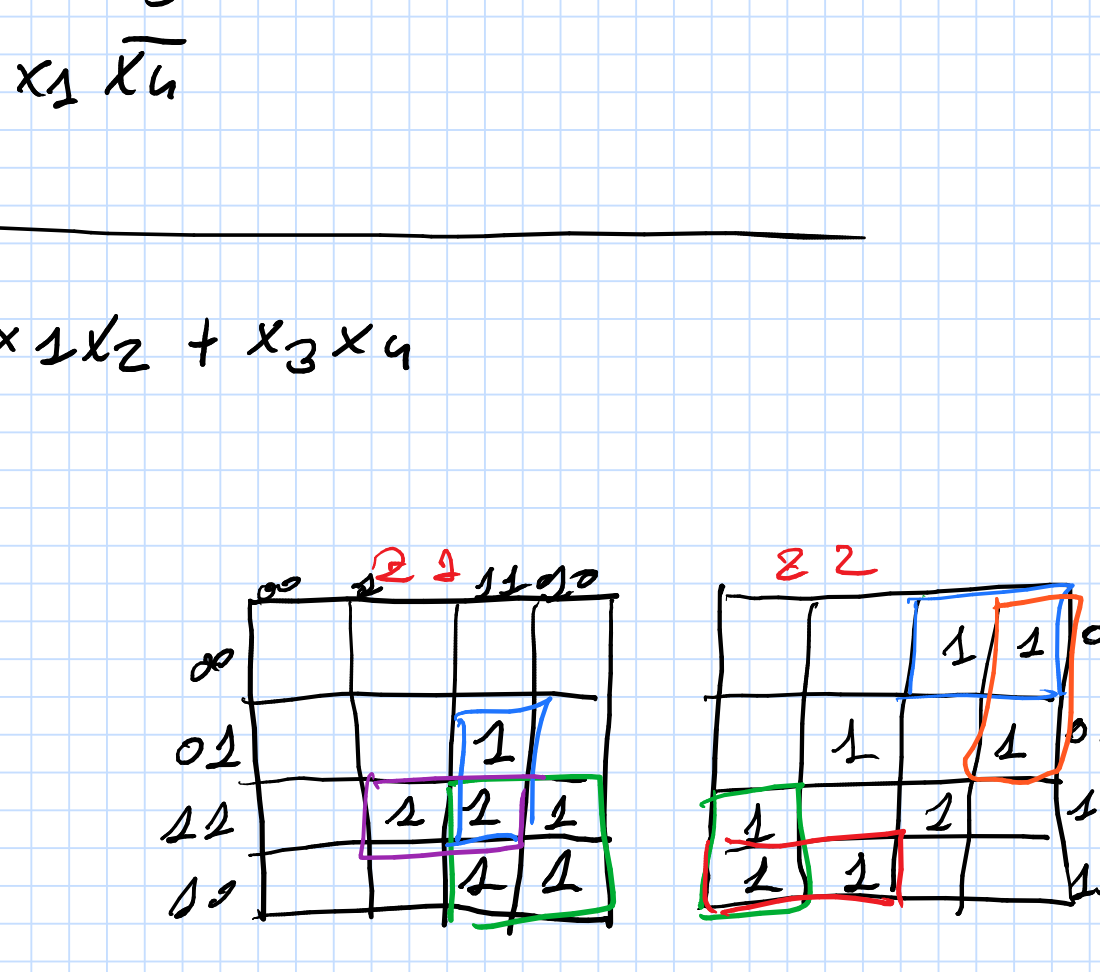
$P_3 = x_2\overline{x_3}$

$P_4 = x_1\overline{x_3}$

$P_5 = x_1\overline{x_4}$

$x_1x_2 + x_3x_4$

$x_1x_2x_3x_4$	$z_1z_2z_3$
0000	000
0001	001
0010	010
0011	011
0100	001
0101	010
0110	011
0111	100
1000	010
1001	011
1010	100
1011	101
1100	011
1101	100
1110	101
1111	110



$z_1\ P_1 = x_2x_3x_4$

$z_1\ P_2 = x_1x_3$

$z_1\ P_3 = x_1x_2x_4$

$z_2\ P_1 = \overline{x_1}\,\overline{x_2}x_3$

$z_2\ P_2 = \overline{x_1}x_3\overline{x_4}$

$z_2\ P_3 = x_1\overline{x_3}x_4$

$z_2\ P_4 = x_1\overline{x_2}\,\overline{x_3}$

$z_3\ P_1 = \overline{x_2}x_3$

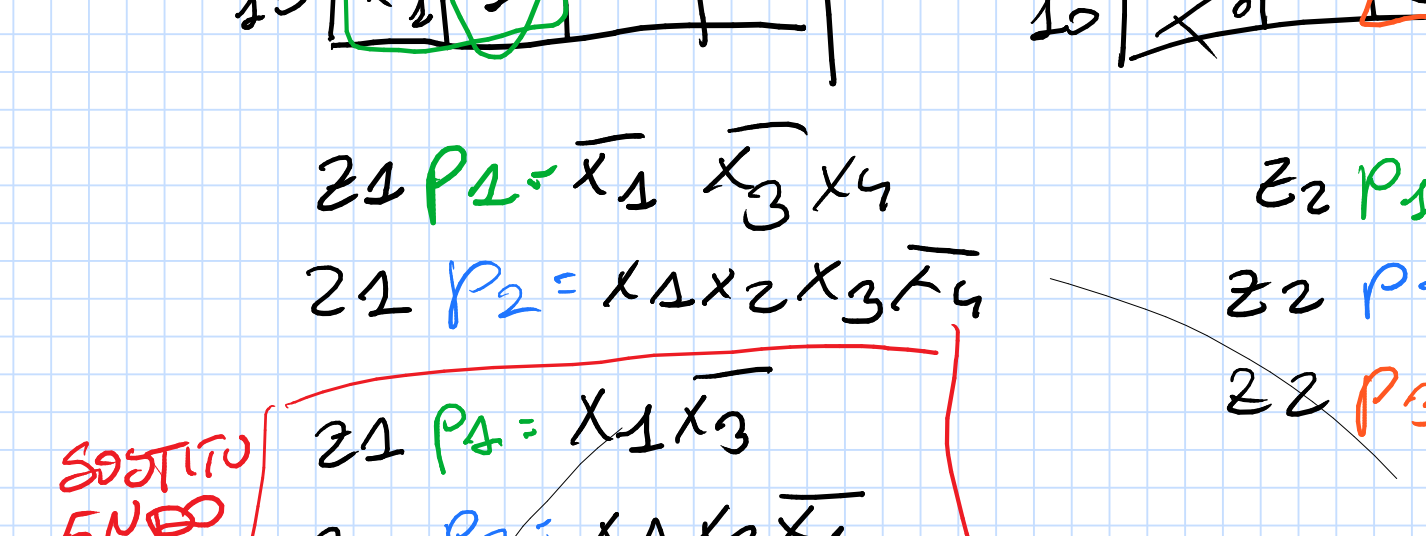
$z_3\ P_2 = x_2\overline{x_4}$

Reti logica 4 variabili in

$z = \left[\frac{x_1x_2}{x_3x_4}\right] \Rightarrow \frac{01}{10} = 1$ ① ②

$x_1x_2x_3x_4$	z_1z_2
0000	00
0001	00
0010	00
0011	01
0100	X
0101	01
0110	01
0111	01
1000	X
1001	10
1010	01
1011	01
1100	X
1101	11
1110	10
1111	01

Reti non completamente specificate



$z_1\ P_1 = \overline{x_1}\,\overline{x_3}x_4$

$z_1\ P_2 = x_1x_2x_3\overline{x_4}$

Sostituendo X con valori che ci convengono:

$z_1\ P_1 = x_1\overline{x_3}$

$z_1\ P_2 = x_1x_2\overline{x_4}$

$z_2\ P_1 = \overline{x_1}x_2$

$z_2\ P_2 = x_2x_4$

$z_2\ P_3 = x_1\overline{x_2}x_3$

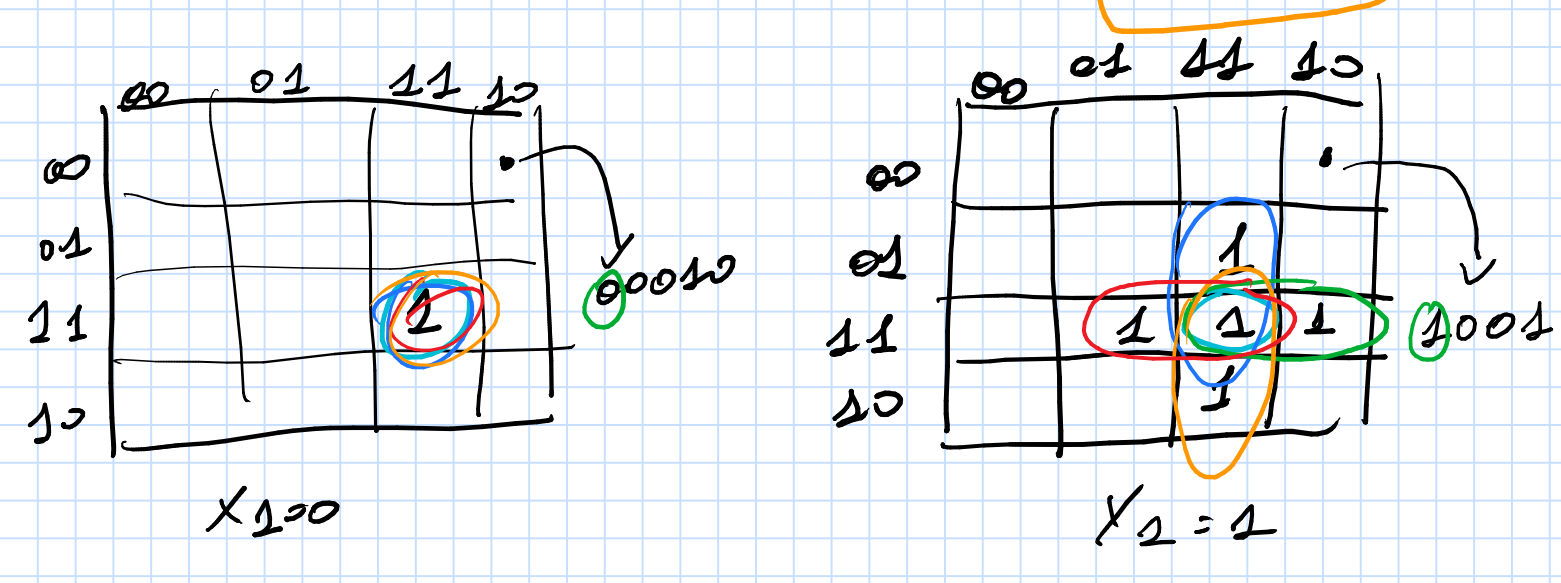
Mappe a 5 variabili

$z = (x_1 + x_2 + x_3 + x_4 + x_5) > 3$

$0+1+0+1+1 = 3 \not> 3$

$1\,1\,0\,1\,1 = 4 > 3$

Dopo >3:
01111
10111
11011
11101
11110
11111



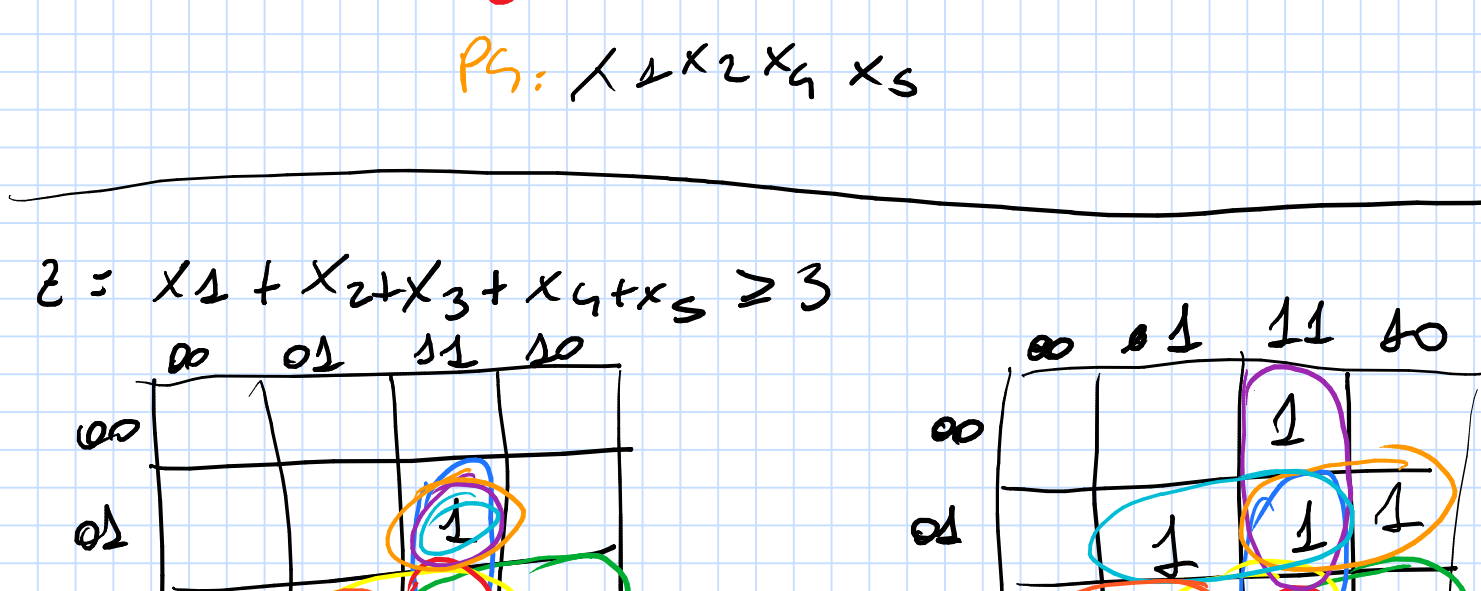
$P_1 = x_2x_3x_4x_5$

$P_2 = x_1x_3x_4x_5$

$P_3 = x_1x_2x_3x_5$

$P_4 = x_1x_2x_4x_5$

$z = x_1 + x_2x_3 + x_4x_5 \geq 3$

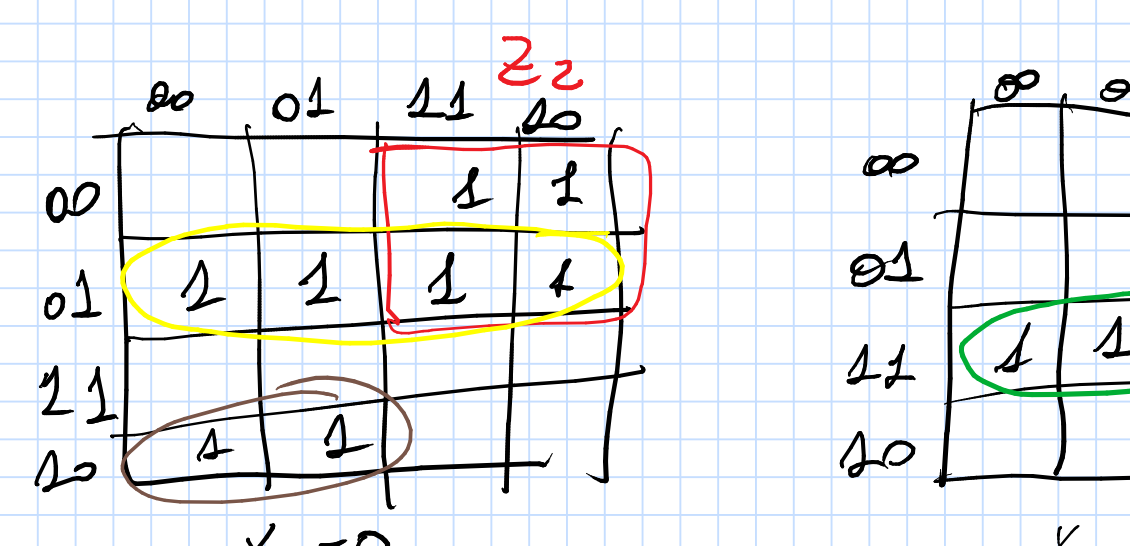


$P_1 = x_3x_4x_5$

$P_2 = x_2x_3x_4$

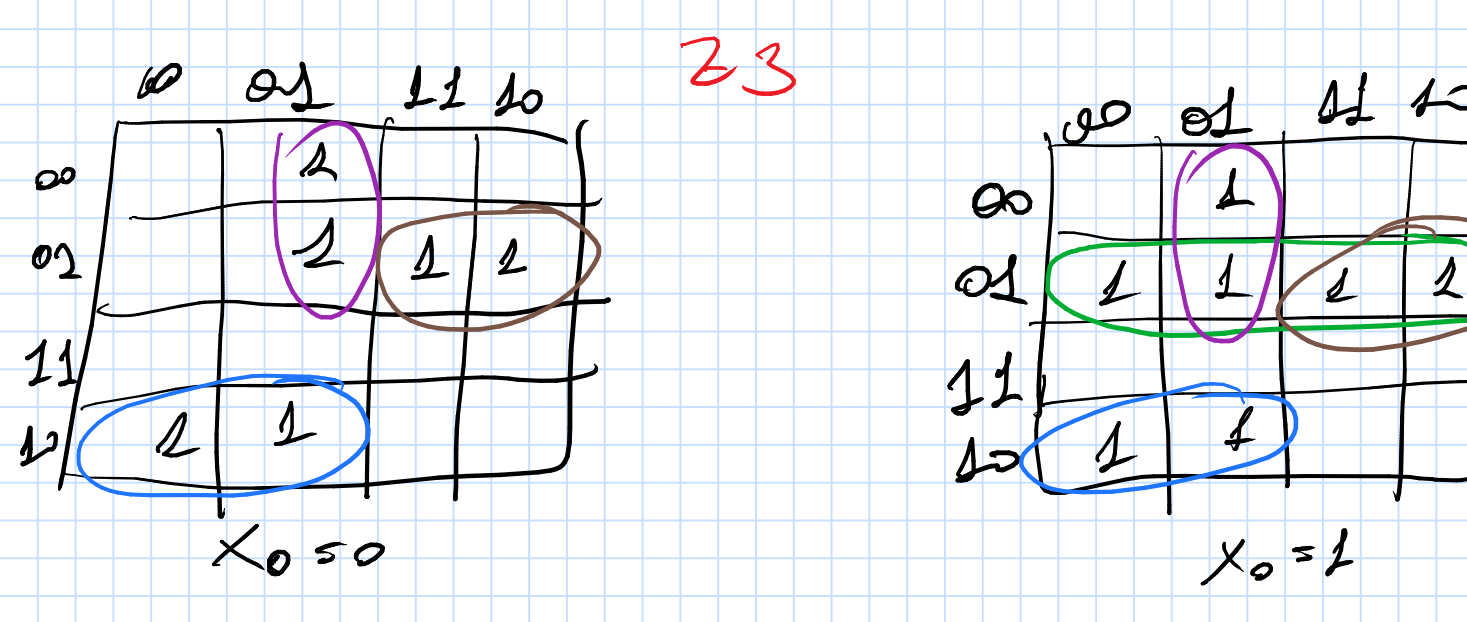
$P_3 = x_1x_4x_5$

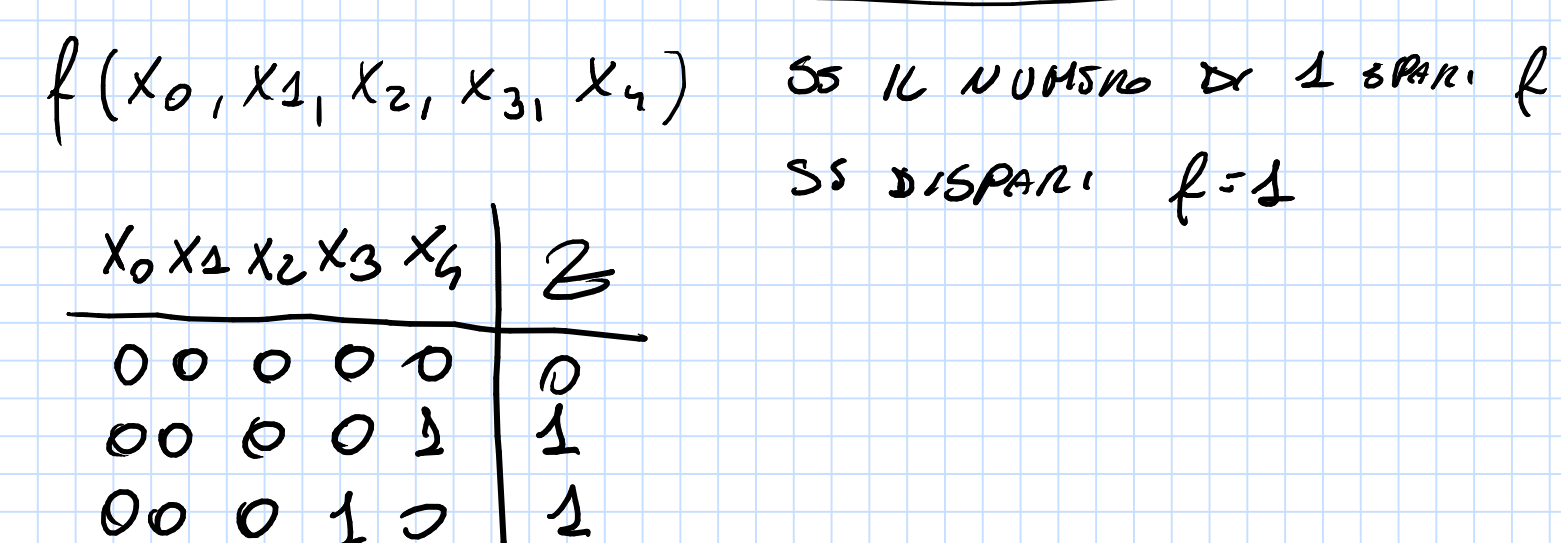
$f(x_0, x_1, x_2, x_3, x_4) = \sqrt{x_0x_1x_2x_3x_4} = z_1, z_2, z_3$ 5 BIT 0...31 $\sqrt{31} = 5,... \to 6 \to 110$



$P_1 = x_1x_2$

$P_2 = x_1x_3$



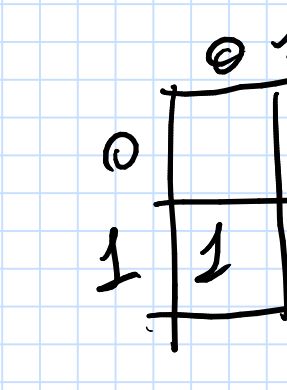


$f(x_0, x_1, x_2, x_3, x_4)$ se il numero di 1 è pari $f=0$, se dispari $f=1$

$x_0x_1x_2x_3x_4$	z
00000	0
00001	1
00010	1
00011	0
00100	1
00101	0
00110	0
00111	1
01000	1
01001	0
01010	0
01011	1
01100	0
01101	1
01110	1
01111	0
10000	1
10001	0
10010	0
10011	1
10100	0
10101	1
10110	1
10111	0
11000	0
11001	1
11010	1
11011	0
11100	1
11101	0
11110	0
11111	1

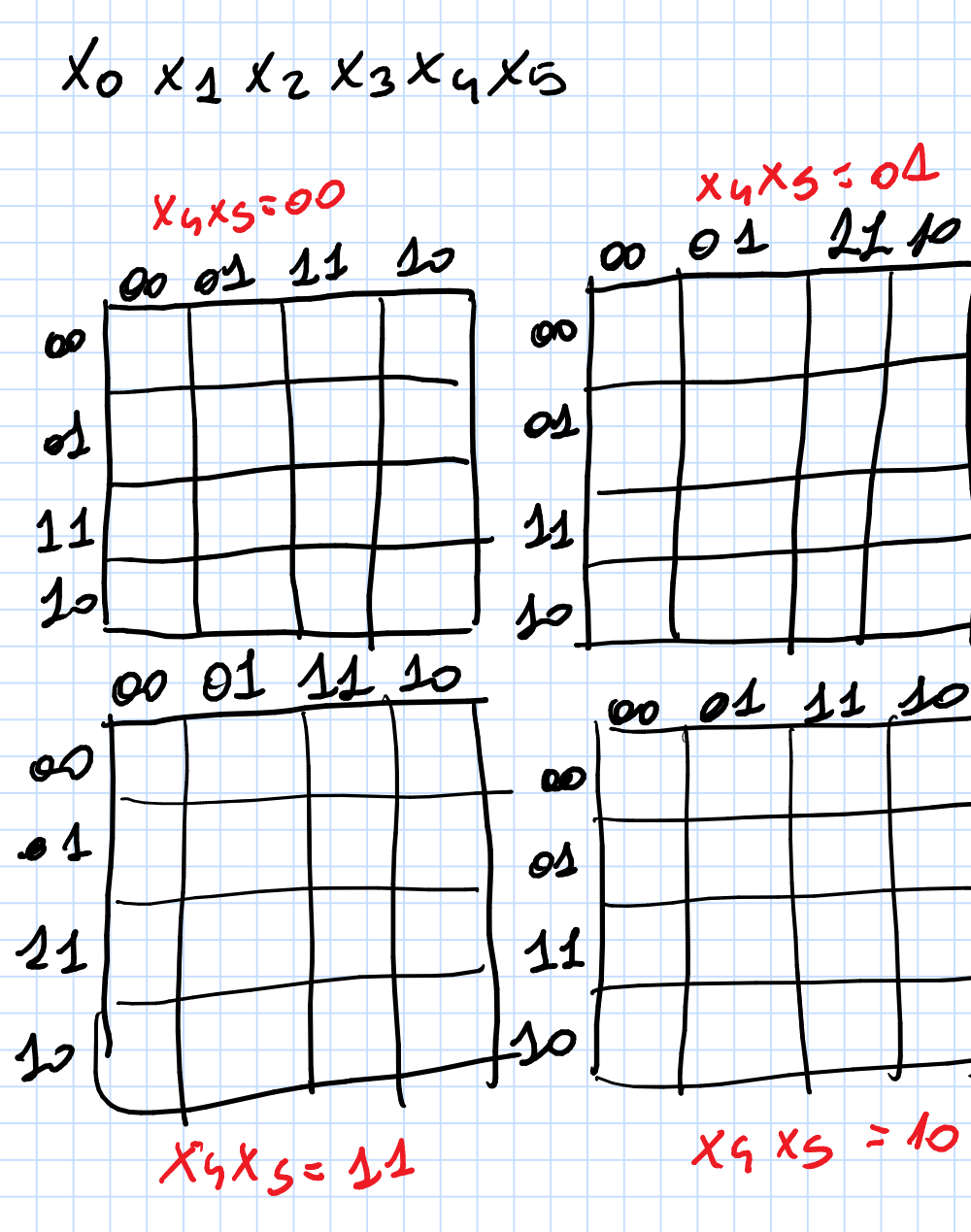
$f(x_0, x_1)$

x_0x_1	z
00	0
01	1
10	1
11	0



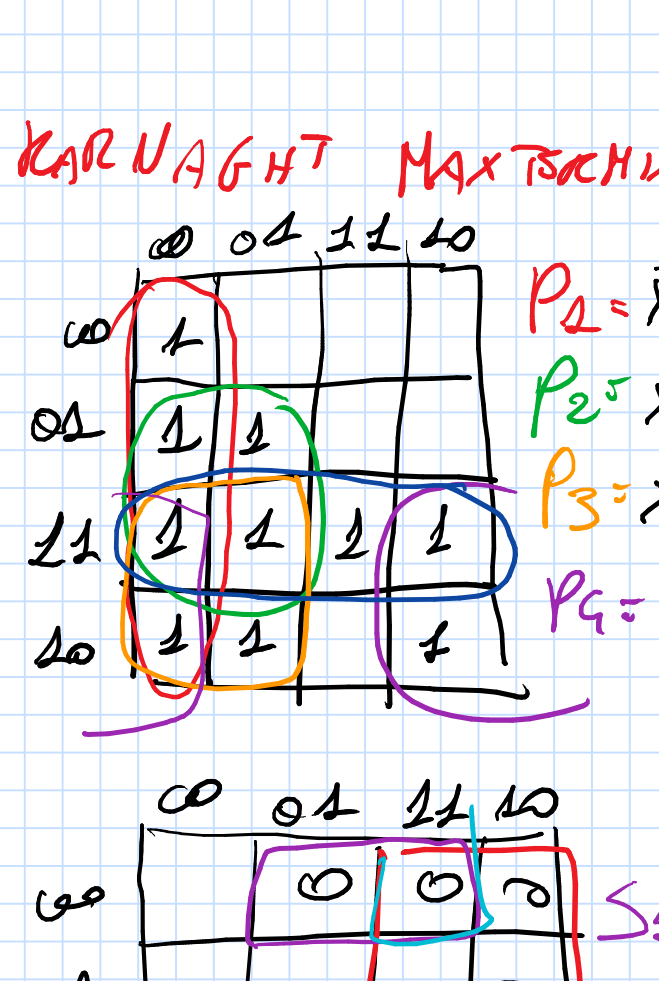
6 variabili

$x_0\ x_1\ x_2\ x_3\ x_4\ x_5$



Mappa Karnaugh Maxtermini

$x_1x_2x_3x_4$	z
0000	1
0001	0
0010	0
0011	0
0100	1
0101	1
0110	0
0111	0
1000	1
1001	1
1010	1
1011	0
1100	1
1101	1
1110	1
1111	1



$P_1 = \overline{x_3}\,\overline{x_4}$ $\quad P_5 = x_1x_2$

$P_2 = x_2\overline{x_3}$

$P_3 = x_1\overline{x_3}$

$P_4 = x_1\overline{x_4}$

$S_1 = (x_1+x_2+\overline{x_4})$

$S_2 = (x_2+\overline{x_3}\,\overline{x_4})$

$S_3\ (x_1+\overline{x_3})$

Moduli combinatori standard

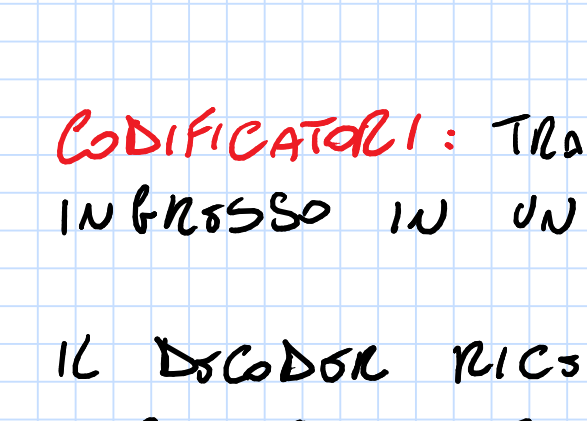
Codificatori: Codificatore / Decodificatore

Selettori: selettore di ingresso, multiplexer, selettore d'uscita, demultiplexer

Aritmetici: somma, ALU

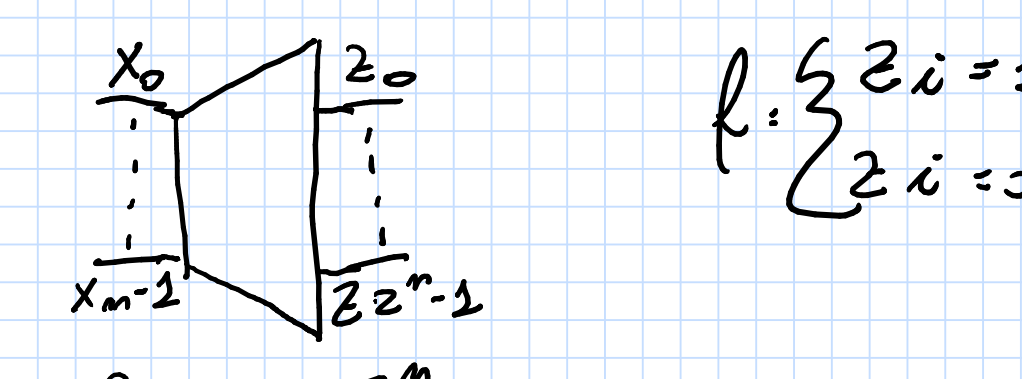
Codificatori: traducono il codice ricevuto in ingresso in un altro codice.

Il decoder riceve n ingressi ed è una rete combinatoria che restituisce 2^n uscite



2 2^m
3 4
4 8
16

$f: \begin{cases} z_i = 1 & x_{NBN} = i \\ z_i = 0 & \text{altrimenti} \end{cases}$ (numero binario naturale)



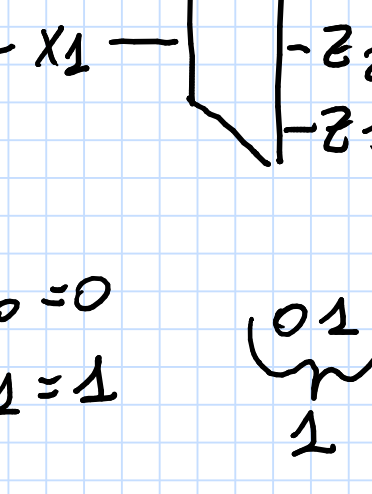
$x_0 = 0$, $x_1 = 1$ → 01 → 1

$x_0 = 1$, $x_1 = 0$, $x_2 = 1$ → 101 → 5

Trasforma da codice binario in un codice posizionale

Esercizio

Decoder 2 bit $n=2$ $2^n = 4$



x_0 x_1	z_0	z_1	z_2	z_3
0 0	1	0	0	0
0 1	0	1	0	0
1 0	0	0	1	0
1 1	0	0	0	1

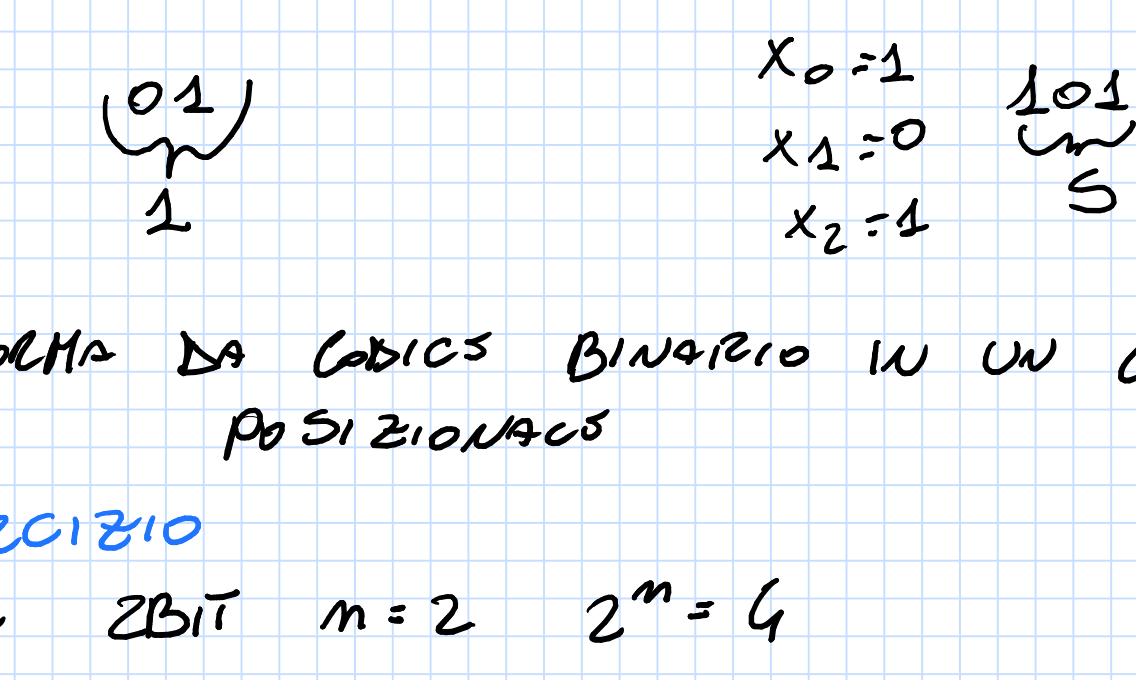
→ $\overline{x_0}\,\overline{x_1}$

$z_0 = \overline{x_0}\,\overline{x_1}$

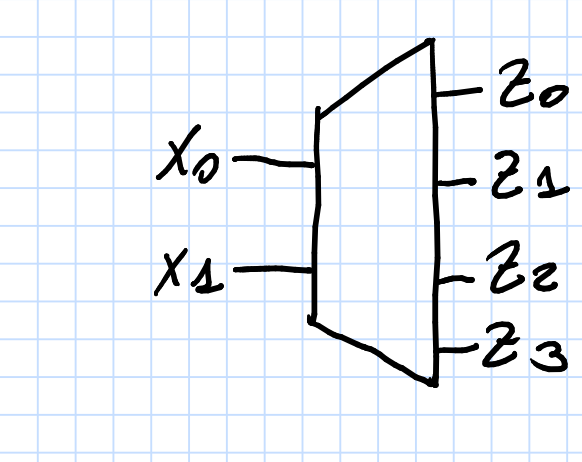
$z_1 = \overline{x_0}x_1$

$z_2 = x_0\overline{x_1}$

$z_3 = x_0x_1$



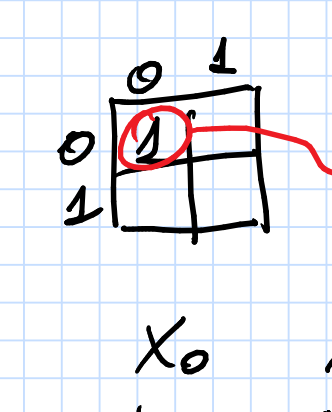
Coder

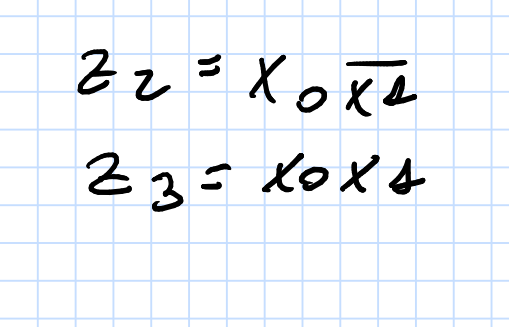


$x_i = 1$, $x_j = 0$, $\forall j \neq i$

$z_0 \ldots z_{m-1}$ NBN $= i$

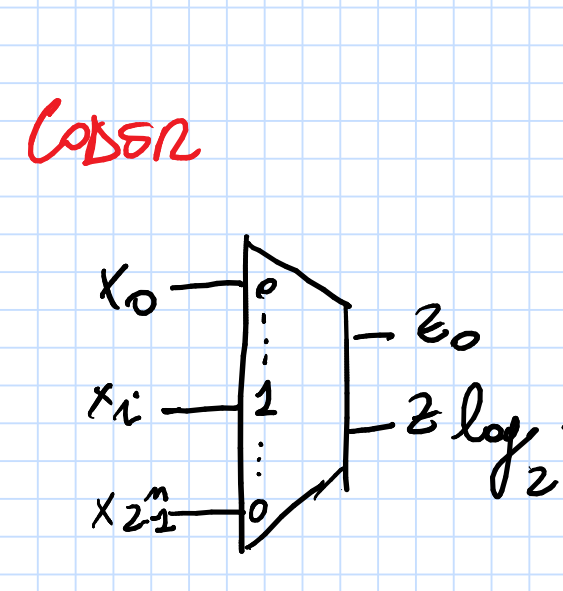
Codice posizionale → Codice binario





$x_0x_1x_2x_3$	z_0z_1
0000	x x
0001	1 1
0010	1 0
0011	x x
0100	0 1
0101	x x
0110	x x
0111	x x
1000	0 0
1001	x x
1010	x x
1011	x x
1100	x x
1101	x x
1110	x x
1111	x x

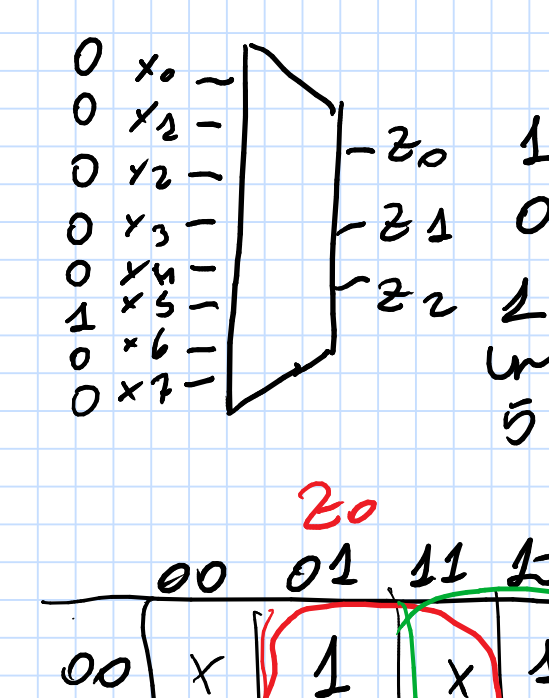
Solo quelli con 1 uno



$P_1 = x_2$

$P_2 = x_3$

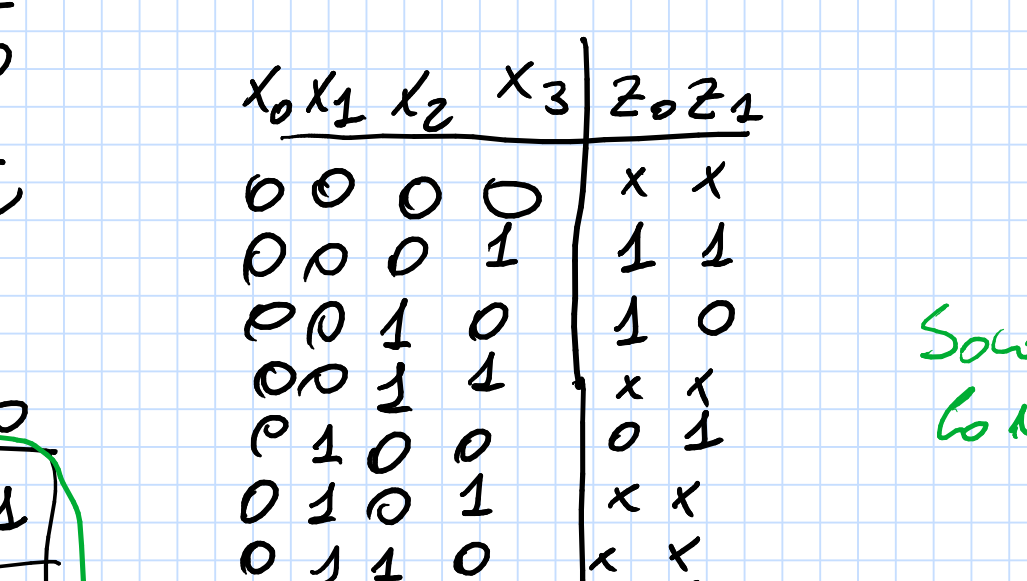
$z_0 = x_2 + x_3$



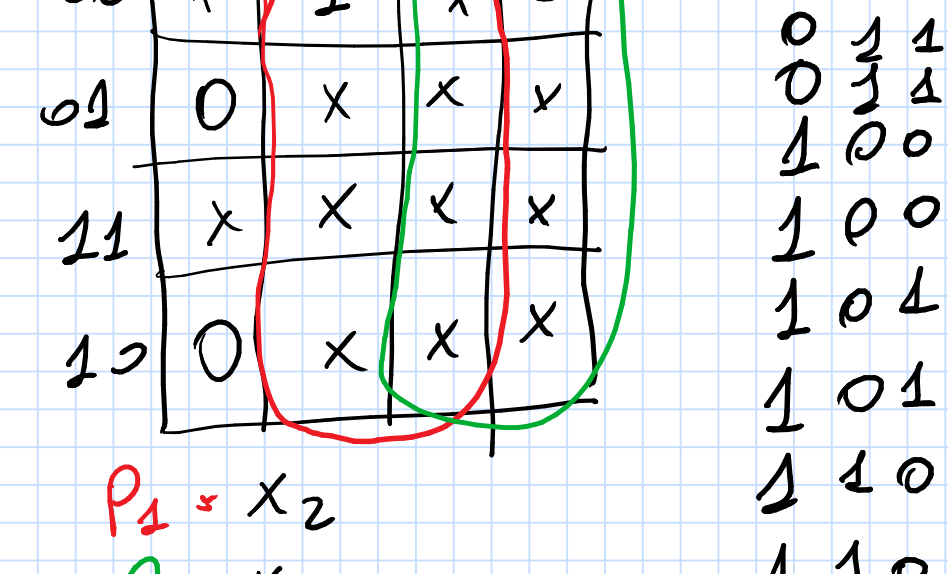
$P_1 = x_1$

$P_2 = x_3$

$z_1 = x_1 + x_3$



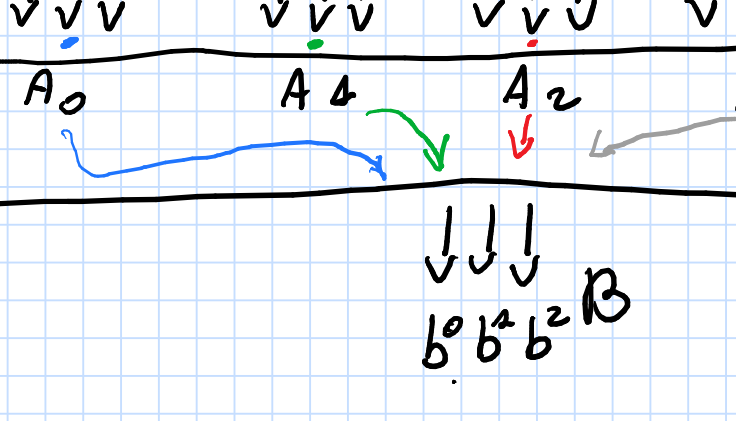
Transcoder



(+1) % 4

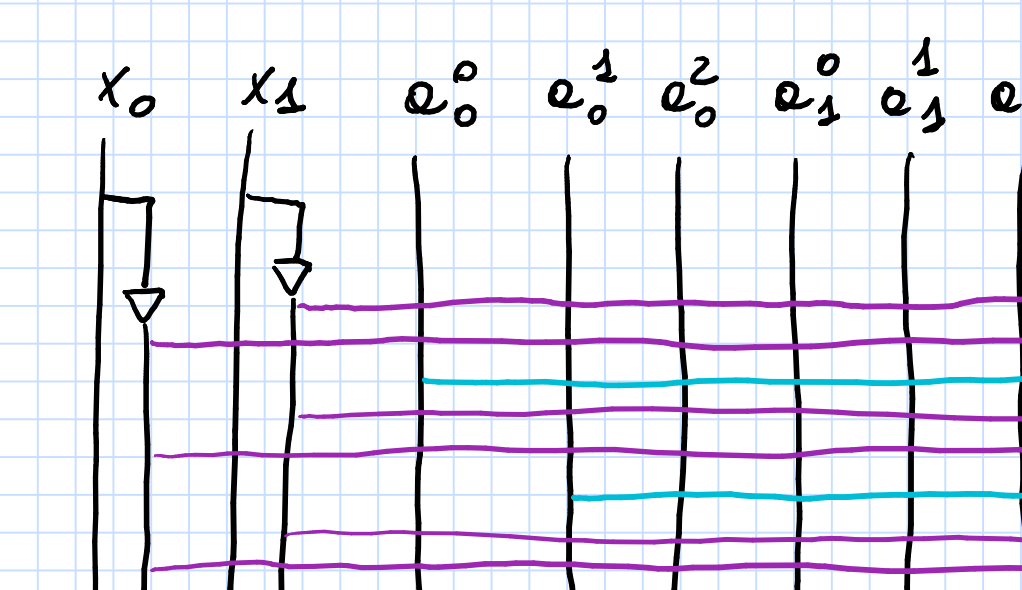
I selettori

Selettori di ingresso → multiplexer → MUX

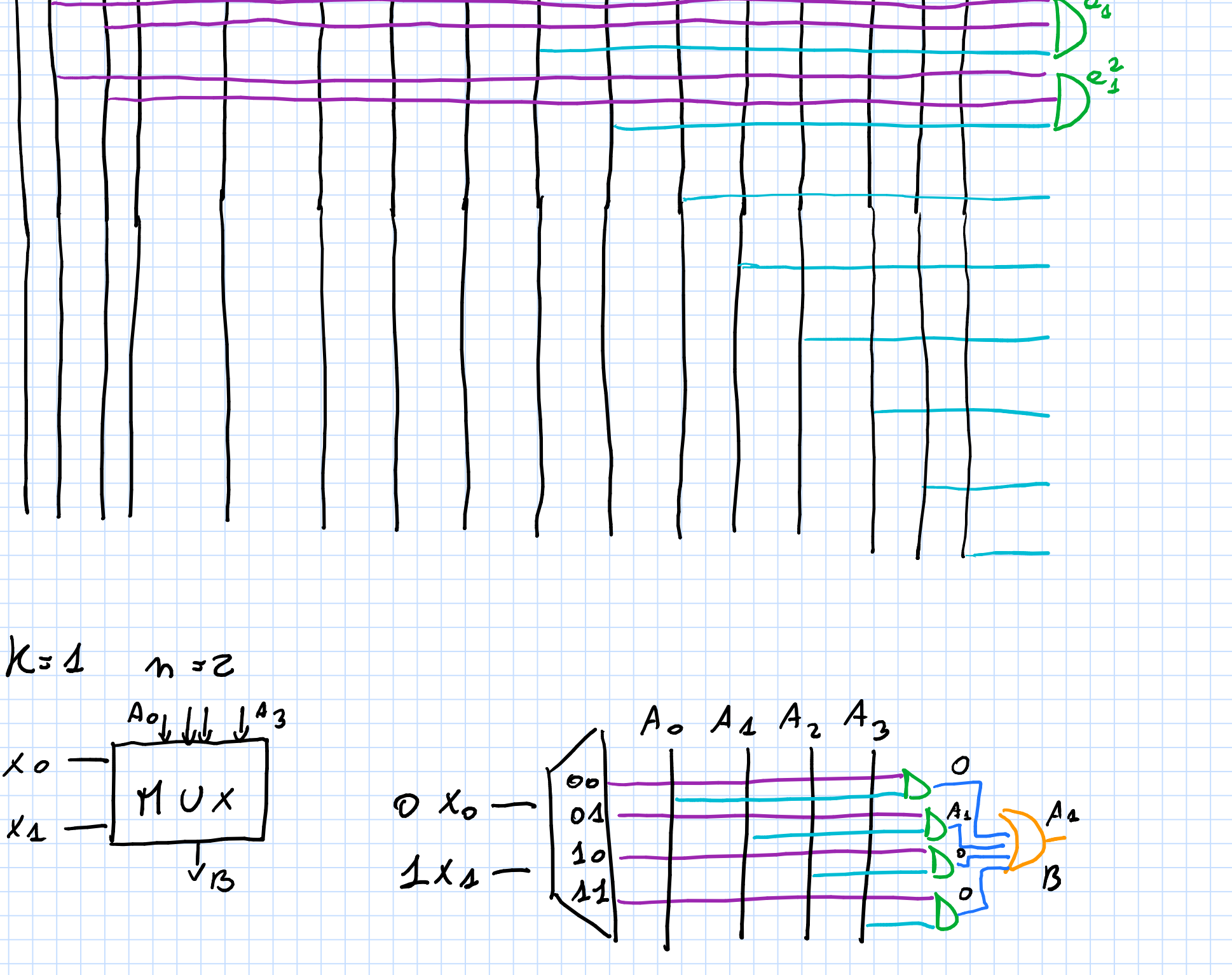


$k=3$

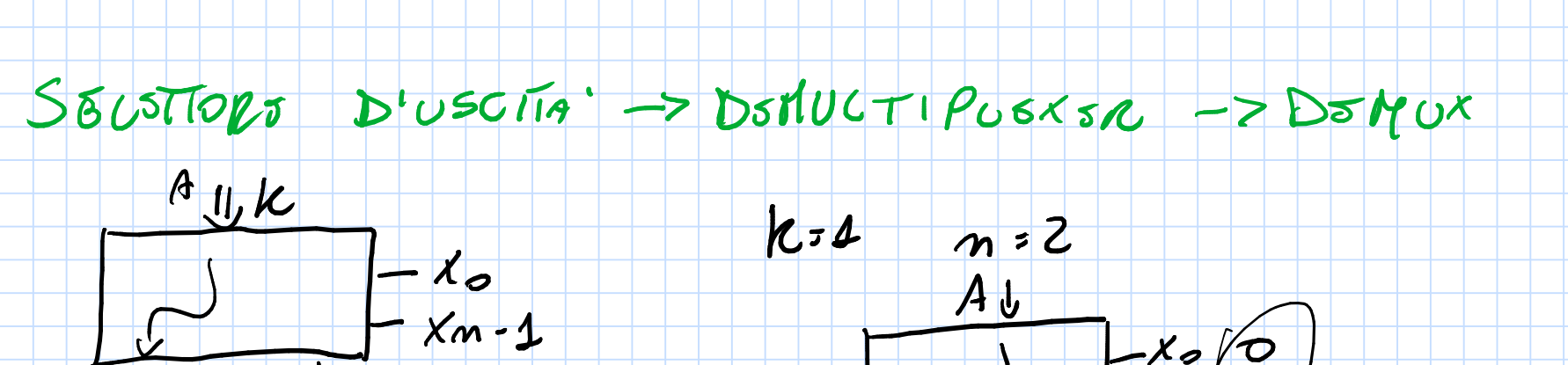
$n=2$



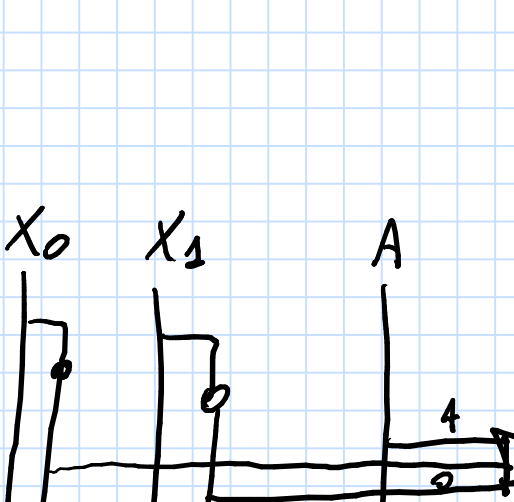
x_0	0	0	1	1
x_1	1	0	0	1



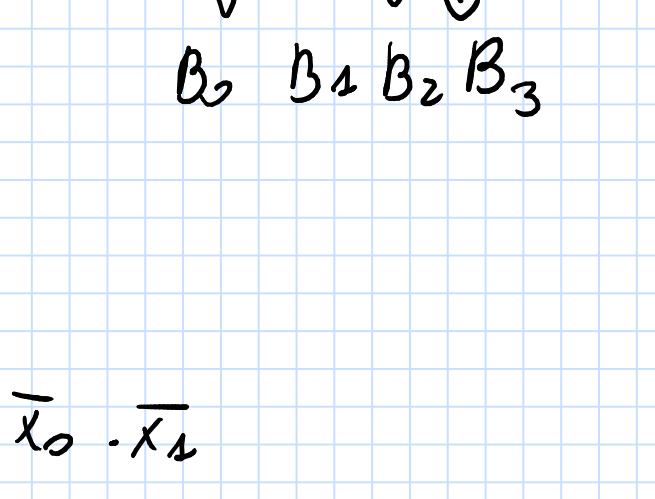
$k=1$ $n=2$



Selettori d'uscita → demultiplexer → DEMUX



$k=1$ $n=2$

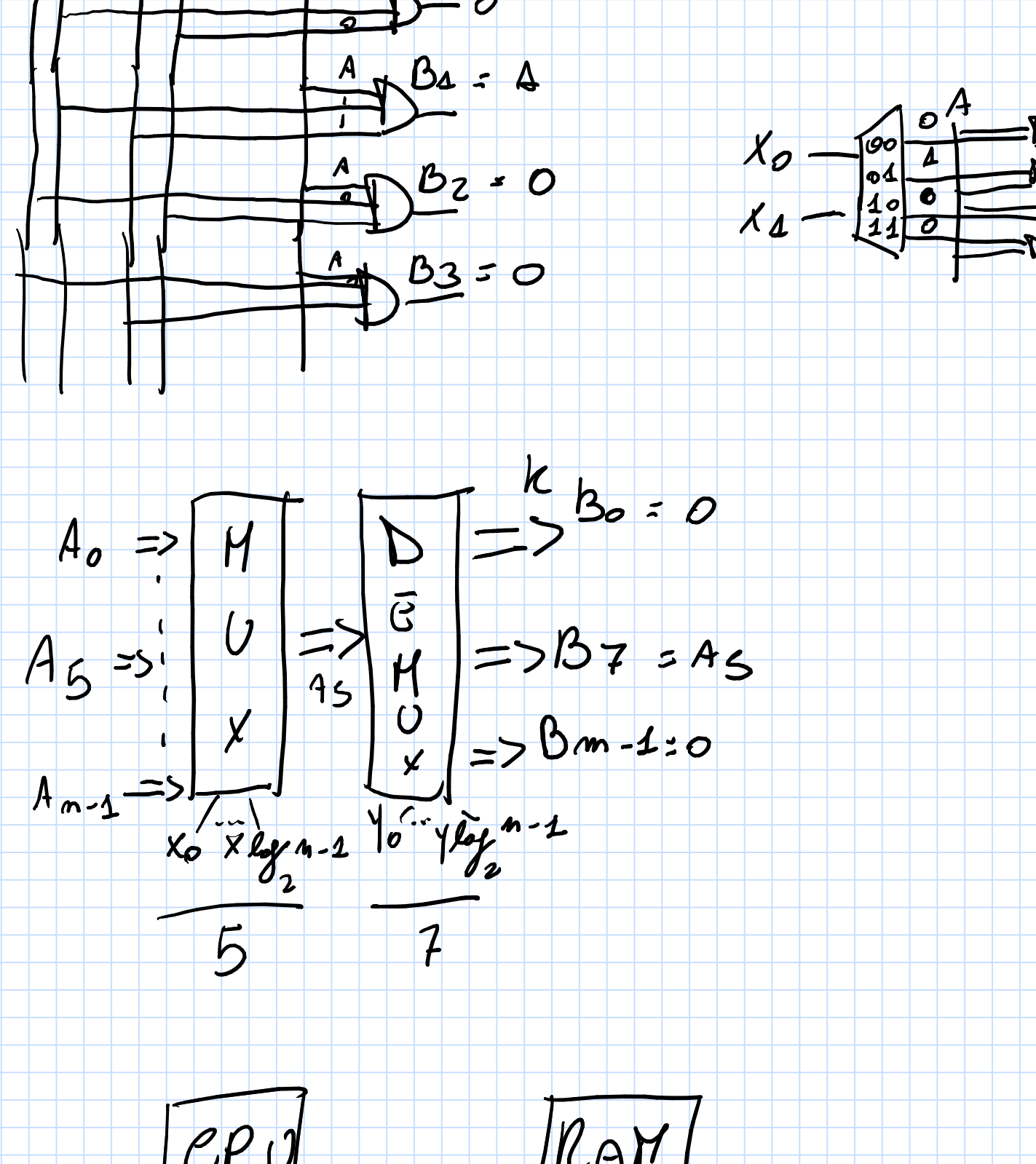


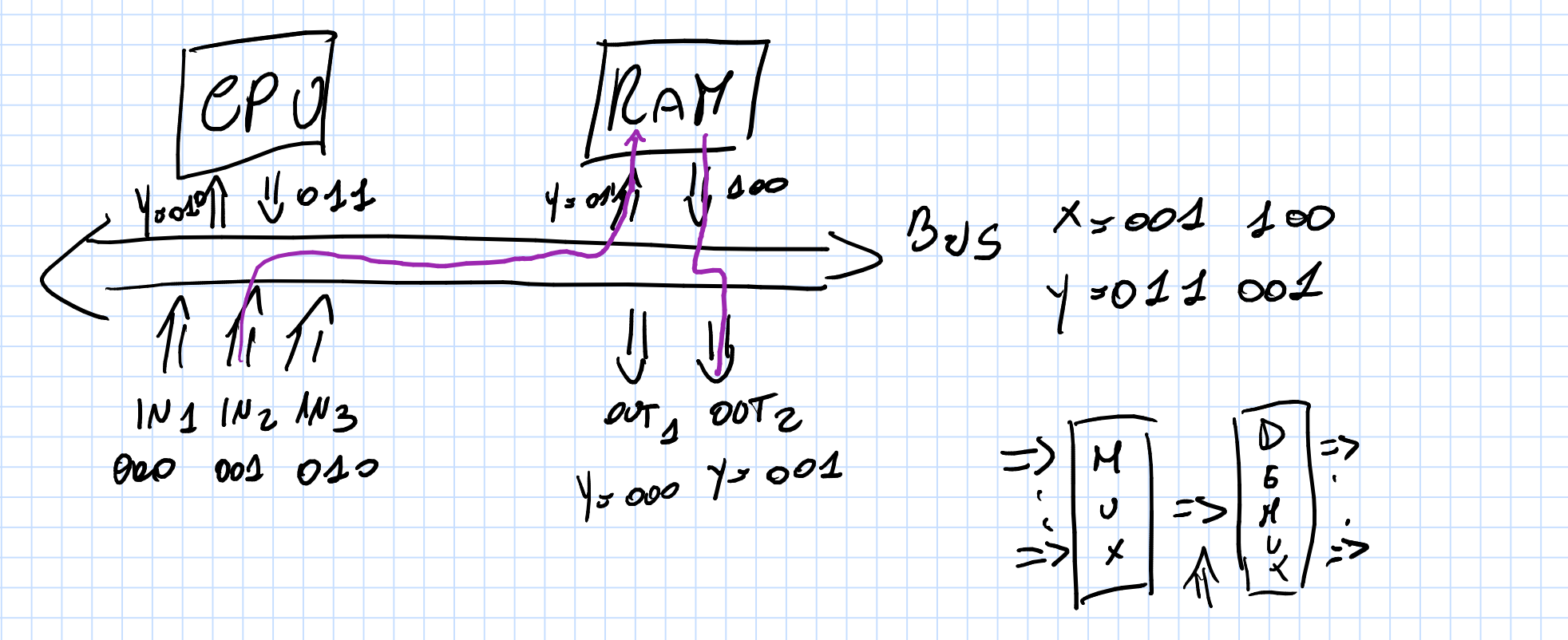
$B_0 = A\cdot\overline{x_0}\cdot\overline{x_1}$

$B_1 = A$

$B_2 = 0$

$B_3 = 0$





$x = 001\ 100$

$y = 011\ 001$

int x, y;   z = 35   y = 5
int z = x & y;   z = x & y; ⇐   x = 0011
z = 1   y = 0101
                                     0001

NOTAZIONE POSIZIONALE

UN SIMBOLO (CIFRA) ASSUME UN SIGNIFICATO DIVERSO IN BASE ALLA POSIZIONE CHE OCCUPA

BASE = 10 ⇒ {0,1,2,3,4,5,6,7,8,9} (CIFRE)

$2334 = 4\cdot10^0 + 3\cdot10^1 + 3\cdot10^2 + 2\cdot10^3$ (cifre numerate 3 2 1 0)

999999999999 = 1000000000000 − 1
⇓
$10^{12} - 1$

$[0, b^n - 1]$ NUMERI INTERI POSITIVI

BASE = 8 {0,1,2,3,4,5,6,7}
BASE = 16 {0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f}
BASE = 2 {0,1}

$(2334)_{10} = 2334$

$(2334)_8 = 2\cdot8^3 + 3\cdot8^2 + 3\cdot8^1 + 4\cdot8^0$

$(2334)_{16} = 2\cdot16^3 + 3\cdot16^2 + 3\cdot16^1 + 4\cdot16^0$

$[0 \ldots 10^{12} - 1]$ $4\cdot10^{3} = 4000$
$[0 \ldots 8^{12} - 1]$ $73248 : 10^{3} = 73$
$[0 \ldots 16^{12} - 1]$

INT x = 3;
y = x · 2;   y = 6

x = 11;
y = x · 2;   y = 110   V   y = x << 1;   MOLTIPLICARE x 2

37   101 = 5
370   1010 = 5 · 2
            10 = 5/2 = 2

BASE 2 : {0,1}   BINARY DIGITS (BIT)

8 BIT = 1 BYTE
2^{10} = 1024 BYTE = 1 KILOBYTE
2^{20} = 1024 KILOBYTE = 1 MEGABYTE
2^{30} = 1024 MEGABYTE = 1 GIGABYTE
4 GB = $4\cdot2^{30}$ BYTE = $2^2\cdot2^{30} = 2^{32}$ BYTE

$(11001)_2 = 1\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = (25)_{10}$

$(2334)_{10}$ 2334 : 10 = 233 resto 4; 233 : 10 = 23 resto 3; 23 : 10 = 2 resto 3; 2 : 10 = 0 resto 2

25 : 2 = 12 resto 1; 12 : 2 = 6 resto 0; 6 : 2 = 3 resto 0; 3 : 2 = 1 resto 1; 1 : 2 = 0 resto 1

11001

```
 101 +       111 +
   1 =        11 =
 110        1010
```

NUMERI NEGATIVI

−25

MODULO E SEGNO   8 CIFRE: 1 SEGNO, 7 MODULO

$[-2^7 - 1 \ldots 2^7 - 1]$ INTERVALLO

```
00011001 + 25          0001100
10011001 = -25         1001100
10110010 = -50 ?       0000000 +0  |-> NO
                       1000000 -0
```

```
            25
=>      00011001      8 CIFRE
CMPL1=> 11100110 +    COMPLEMENTO A 1
               1
        11100111 = -25
CMPL 2
```

```
 11100111
 00011000
        1
 00011001  <- 25
```

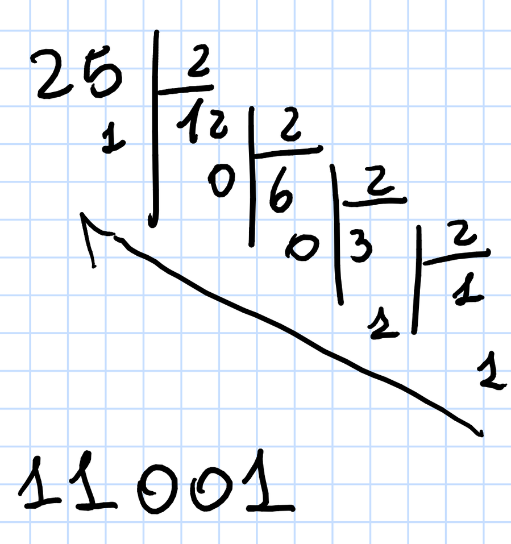
RAPPRESENTAZIONE 0   SOMMA ? ESSERE

```
  00000000              00011001 + 25
  11111111 +            11100110 = -25
         1              00000000    0
1|00000000
```

MODULO ARITMETICO

```
 0101 +      1
 0011 =      0
 01 10       0  -> 0
             ↓
             1
```

a + b + r → s
           ↳ r'



a	b	r	s	r'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S

a \ br	00	01	11	10
0		1		1
1	1		1	

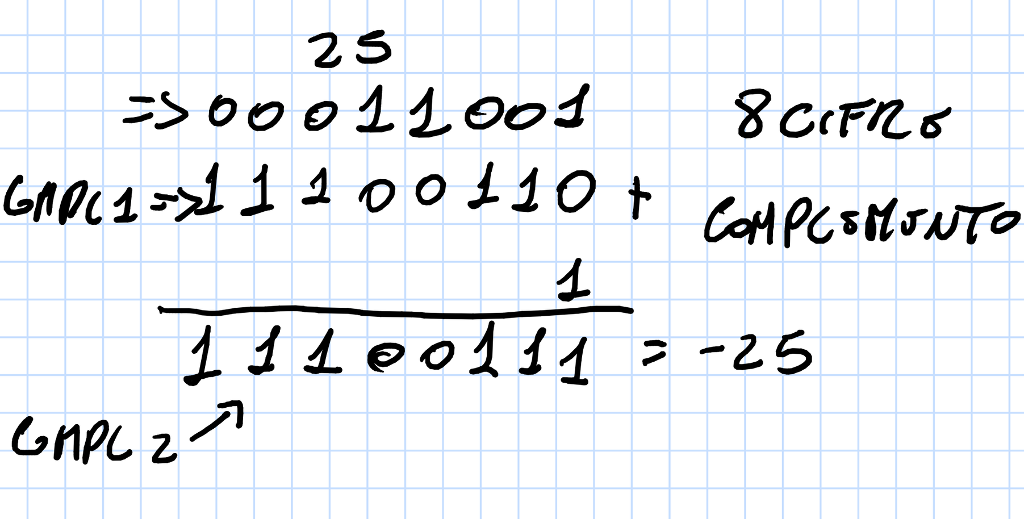
$S = a\bar{b}\bar{r} + \bar{a}\bar{b}r + abr + \bar{a}b\bar{r}$

R'

a \ br	00	01	11	10
0			1	
1		1	1	1

$R' = ar + br + ab$

$A = a_3\,a_2\,a_1\,a_0$ +
$B = b_3\,b_2\,b_1\,b_0$ =

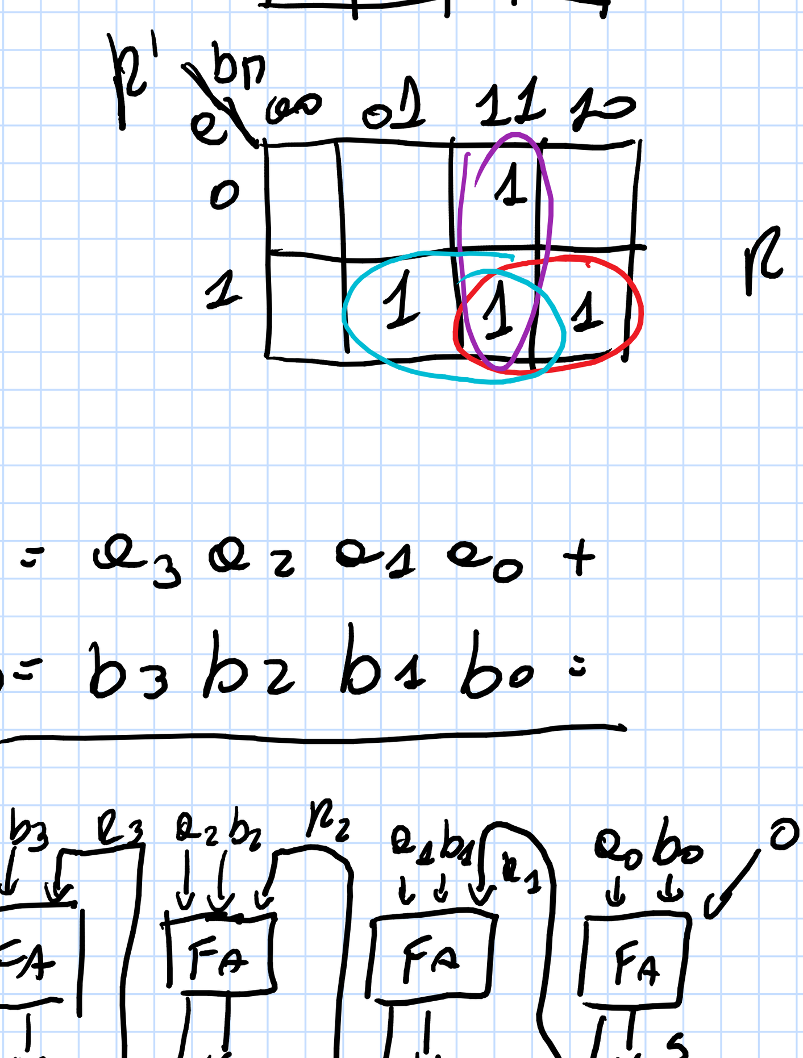


SOMMATORE A PROPAGAZIONE DI RIPORTO
2 · n LIVELLI

Δs   ΔR

T = (n − 1) ΔR + Δs

UNITA' ARITMETICO LOGICA



$\bar{B} + 1 = -B$

X_0	X_1	R_0	OP	
0	0	0	B	
0	0	1	B+1	INC
0	1	0	$\bar{B}$	NOT
0	1	1	−B	CAMBIO SEGNO
1	0	0	A+B	SOMMA
1	0	1	A+B+1	A AND B
1	2	0	A−B−1	A OR B
1	1	1	A−B	DIFFERENZA

OVERFLOW QUANDO I BIT NON BASTANO PER DESCRIVERE UN RISULTATO

PROVARE A SCRIVERE UNA RETE LOGICA
BIT OVERFLOW (1,0) (RESTITUITO)
BIT SIGNIFICATIVO OPERANDI
BIT SIGNIFICATIVO RISULTATO

S = 101

D F A A A 2 S 6

1011,1111,1010,1010,1010,0001,0101,0110

BFAAA156

0 1 2 3 4 5 6 7 8 9 A B C D E F

0 => 0000

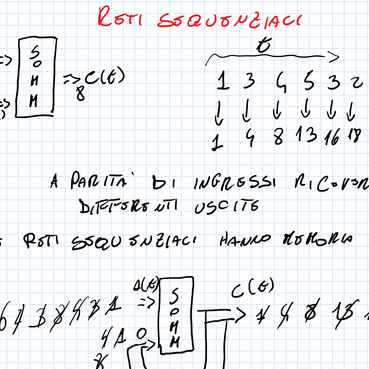
1 => 0001

2 => 0010

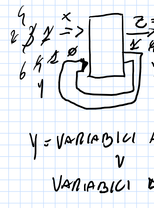
⋮

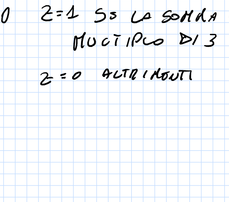
F => 1111

ALU (RIPASSO)



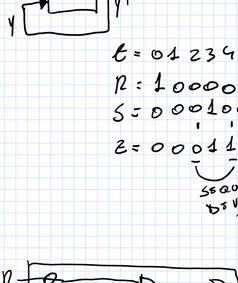
RETI SEQUENZIALI

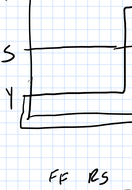




A PARITÀ DI INGRESSI RICORRENTE DIFFERENTI USCITE

LE RETI SEQUENZIALI HANNO MEMORIA





z = 1 SE LA SOMMA È MULTIPLO DI 3

z = 0 ALTRIMENTI

y = VARIABILI ANSELCO ∨ VARIABILI DI STATO

OTTIMIZZARE UNA RETE SEQUENZIALE VUOL DIRE DIMINUIRE IL NUMERO DI Y

FLIP FLOP RS

R	S	z
0	0	non
0	1	1
1	0	0
1	1	–

t = 0 1 2 3 4 5

R = 1 0 0 0 0 0

S = 0 0 0 1 0 0

z = 0 0 0 1 1 1

SEQUENZIALE DI VALORE AUSILIARIO 2 BIT

R	S	y	z = y'
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	X

$z = y^1 = \bar{R}y + S$

$z = (S+y)\cdot\bar{R}$

FF RS

$z\,(S+y)\cdot\bar{R} = \overline{[(S+y)\cdot\bar{R}]} = \overline{(S+y)} + \bar{\bar{R}} =$

$\overline{(S+y)} + R = (S \uparrow y) \uparrow R$

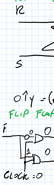
STATI {0, 1}

$k = 2^k$ STATI

FLIP FLOP RS ASINCRONO

LATCH

FLIP FLOP (ASINCRONO)

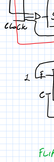


R	S	Q	$x \vee \bar{x}$
0	0	y	$\bar{y}$
0	1	1	0
1	0	0	1
–	–		

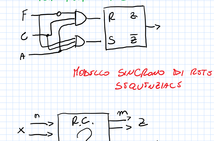
$0 \vee y = (\overline{0 \vee y}) = \bar{y}$ $\quad 0 \vee \bar{y} = (\overline{0 \vee \bar{y}}) = \bar{\bar{y}} = y$

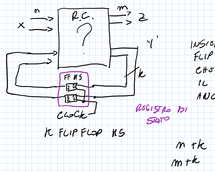
FLIP FLOP FC

Clock :0



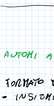
CONFIGURAZIONE A MASTER-SLAVE



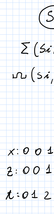


CLOCK ALTO SI MODIFICA MASTER
CLOCK BASSO SI MODIFICA SLAVE

FLIP FLOP FAC



MODULO SINCRONO DI TESTO SEQUENZIALE



INSIEME DI FLIP FLOP CHE CONDIVIDONO IL CLOCK VIENE ANCHE DETTO REGISTRO

REGISTRO DI STATO

K FLIP FLOP KS

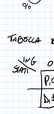
m tk INGRESSI
m+k USCITE

m+k TESTI COMBINATORI

SINTETIZZARE TESTI SEQUENZIALI

AUTOMI A STATI FINITI :

FORMATO DA:
- INSIEME FINITO DI STATI $\{S_1, S_2, \dots, S_n\}$
- INSIEME DI CONFIGURAZIONI D'INGRESSO $\{x_1, x_2, \dots, x_3\}$
- INSIEME DI CONFIGURAZIONI D'USCITA $\{z_1, z_2, \dots, z_3\}$



$\sum (S_i, x_k) = S_j$

$\omega (S_i, x_k) = z_k$

TESTO SEQUENZIALE
2-0 SE ω_{N-1} = PARI
2-1 ALTRIMENTI

x: 0 0 1 1 0 1 1 1 0 0 0 1
z: 0 0 1 0 0 1 0 1 1 1 0
t: 0 1 2 3 4 5 6 7 8 9 10 11

$X = \{0, 1\}$ $\quad Z = \{0, 1\}$ $\quad S = \{PARI, DISPARI\}$

TABELLA DI FLUSSO → CODIFICA DEGLI STATI

STATI / INGR.	0	1
P	P, 0	D, 1
D	D, 0	P, 0

STATI	Y
P	0
D	1

TABELLA DELLA TRANSIZIONE

x / y	0	1
0	0, 0	1, 1
1	1, 1	0, 0

MAPPA y'

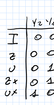
x / y	0	1
0	0	1
1	1	0

$y' = \bar{x}\, y + x\, \bar{y}$

MAPPA z

x / y	0	1
0	0	1
1	1	0

$z = \bar{x}\, y + x\, \bar{y}$



RISOLUZIONE
2-0 SE L'INGRESSO È PRECEDENTE DELLO STESSO
2-1 ALTRIMENTI

X:	0	1	0	1	1	0	0	0	1	0	1
Z:	0	0	1	0	0	0	0	1	0	0	1
T:	0	1	2	3	4	5	6	7	8	9	10	11

	0	1
I	2, 0	U, 0
2	2x, 0	2x, 0
U	Ux, 0	Ux, 0
2x	I, 1	I, 0
Ux	I, 0	I, 1

	Y_2	Y_1	Y_0
I	0	0	0
2	0	0	1
U	0	1	1
2x	0	1	1
Ux	1	0	0

	0	1
000	001, 0	010, 0
001	011, 0	011, 0
010	100, 0	100, 0
011	000, 1	000, 0
100	000, 0	000, 1
101	xxx, x	$Y_2' Y_1' Y_0'$, z
110	xxx, x	xxx, x
111	xxx, x	xxx, x

4 TESTI A 4 INGRESSI

MAPPA Y_2'

$Y_1 Y_0$ / Y_2	00	01	11	10
00	1			
01	1	X	X	
11	X	X		
10			X	

$Y_2 \bar{Y}_0$

MAPPA Y_1'

$Y_1 Y_0$ / Y_2	00	01	11	10
00		1	1	
01	X	X	X	
11			X	
10				

$\bar{Y}_1 Y_0$

$\bar{Y}_2 \bar{Y}_1 x$

MAPPA Y_0'

$Y_1 Y_0$ / Y_2	00	01	11	10
00	1		1	
01	X	X	X	
11	X	X	X	
10				

$\bar{Y}_2 Y_0$

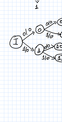
$Y_2 \bar{Y}_0 \bar{x}$

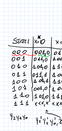
MAPPA z

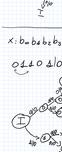
$Y_1 Y_0$ / Y_2	00	01	11	10
00				1
01				
11	X	X	X	X
10		1		

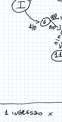
$Y_1 Y_0 \bar{x}$

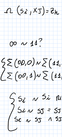
$x_2 x$

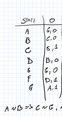


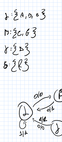


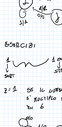


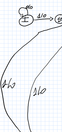


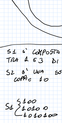












010 | 110 | 111 |
1 1 1 1

PALINDROMA?

$z(t) = x(t-2)\, x(t-1)\, x(t)$

010 110 111
↓
1

	STATI	x=0	x=1
000	I	0,0	1,0
001	0	00,0	01,0
010	1	10,0	11,0
011	00	00,1	01,0
100	01	10,1	11,0
101	10	00,0	01,1
110	11	10,0	11,1

STATI	x=0	x=1
000	001,0	010,0
001	011,0	100,0
010	101,0	110,0
011	011,1	100,0
100	101,1	110,0
101	001,0	100,1
110	101,0	110,1
111	xxx,x	xxx,x

$y_2 y_1 y_0$ $y_2' y_1' y_0', z$

$y_2y_1 \backslash y_0x$	00	01	11	10
00			1	
01	1	1	1	
11	1	1	x	x
10	1	1	1	

$Y_2' = Y_0 x + Y_1\overline{Y_0} + Y_2\overline{Y_0}$

---

x : 0010|0110|1001|1101|
0 1 1 0

1|00|x
1|11|x
↑

---

x : b0 b1 b2 b3 b4 $z=1 \Rightarrow b_0=b_1=b_4$
$b_2=b_3$

x = 00110| 10|1110|1|0|00 1|001|11001|
1 0 0 0 0 0 1

---

x: b0 b1 b2 b3 b4 $z=1 \Rightarrow \begin{cases} b_0=b_3 \\ b_1=b_2=b_4 \end{cases}$

0110 1|0 1100|01100|

---

1 INGRESSO x   0000 Q 0000   1 SSE N° DI 1 IN Q È PARI
1 USCITA z   0 ALTRIMENTI

1010000 101000110 0000|
START Q STOP

1 INGRESSO x   1010 Q 1010

1 USCITA z   1111 Q 1111   DI 1 IN Q È PARI

$b_0 = b_2$
$b_1 = b_3 = b_4$

CODIFICA

STATI	CODIFICA
I	0000
0	0001
1	0010
00	0011
01	0100
10	0101
11	0110
000	0111
010	1000
101	1001
111	1010
X0X0	1011
X1X1	1100

FLUSSO

STATI	0	1
I	0,0	1,0
0	00,0	01,0
1	10,0	11,0
00	000,0	001,0
01	010,0	011,0
10	100,0	101,0
11	110,0	111,0
000	X0X0,0	I,0
010	I,0	X1X1,0
101	X0X0,0	I,0
111	I,0	X1X1,0
X0X0	I,1	I,0
X1X1	I,0	I,1

TRANSIZIONI

STATI	0	1
0000	001,0	0010,0
0001	0011,0	0100,0
0010	0101,0	0110,0
0011	0111,0	0000,0
0100	1000,0	0000,0
0101	0000,0	1001,0
0110	0000,0	1010,0
0111	1011,0	0000,0
1000	0000,0	1100,0
1001	0000,1	0000,0
1010	0000,0	0000,1
1011		
1100		
1101	xxxx,x	xxxx,x
1110	xxxx,x	xxxx,x
1111	xxxx,x	xxxx,x

DA CONTINUARE

$\begin{cases} b_0 = b_1 \\ b_2 = b_3 = b_4 \end{cases}$

$\Sigma(S_i, X_J) = S_k$ $S_i \sim S_J$

$\Omega(S_i, X_J) = Z_h$ $\forall x_k\ \Sigma(S_i, X_k) \sim \Sigma(S_J\, x_k)$

$\forall x_k\ \Omega(S_i, x_k) = \Omega(S_J, x_k)$

00 ~ 11?

$\begin{cases} \Sigma(00,0) \sim \Sigma(11,0)?\ 000 \sim 110? \\ \Sigma(00,1) \sim \Sigma(11,1)?\ 001 \sim 111? \end{cases}$ | $\Omega(00,0) = \Omega(11,0)?\ 0=0$ SI
$\Omega(00,1) = \Omega(11,1)?\ 0=0$ SI

$S_i \sim S_i$ RIFLESSIVA
$S_i \sim S_J = S_J \sim S_i$ TRANSITIVA
$S_i \sim S_J \wedge S_J \sim S_k \Rightarrow S_i \sim S_k$ SIMMETRICA

STATI	0	1	
A	G,0	E,1	–
B	C,0	A,1	–
C	E,1	G,0	
D	B,0	E,1	
E	G,0	A,1	
F	D,1	F,0	
G	A,1	G,0	

TABELLA SCALA

	A	B	C	D	E	F	G
A	•						
B	CG AG	•					
C	X	X	•			X	
D	~~AG~~	~~BC AG~~	X	•			
E		CG	X	~~DG AE~~	•		
F	X	X	DE FG	X	X	•	
G	X	X	AG	X	X	~~AB~~	•

INIZIALE

A ~ B ⇒ C ~ G, A ~ E

~~A ~ B~~   ~~C ~ F~~
~~B ~ D~~   ~~C ~ G~~
~~B ~ E~~   D ~ E
~~F ~ F~~

~~A ~ B~~   ~~C ~ F~~
~~B ~ E~~

A ~ B   C ~ G
A ~ E
B ~ E

α:{A, B, E}
β:{C, G}
γ={D}
δ:{F}

n NUM STATI
k INGRESSI

$n^2 \cdot 2^k \cdot n^2 = O(n^4 \cdot 2^k)$

ESERCIZI

1 ～～～ 100
↓ START   ↓ STOP

3 4 5 6 7 8 9 10 11 12 13 14 15
1 0 0 0 0 0 1 0 0 0 0 0 1

z = 1 SE IL NUMERO DI 1
È MULTIPLO DI 3 MA NON
DI 6

1|1|100
1|10|1
1|11|1
1|110|1
1|111|1

S1 È COMPOSTA SOLO DA UN NUMERO COMPRESO TRA 1 E 3 DI COPPIE 10, E TERMINA CON UNO 0

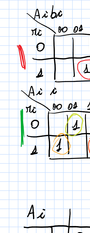
S2 È UNA SEQ. GENERICA CHE HA TANTE COPPIE 10

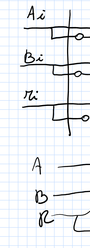
S1 $\begin{cases} 100 \\ 10100 \\ 1010100 \end{cases}$

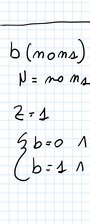
100 0000111 10
S2 1

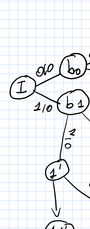
00110100 0 10 11110
S1 1

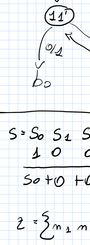
ESERCIZI AUTOMI

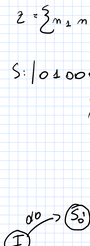


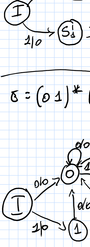


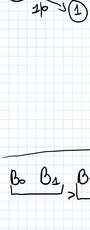


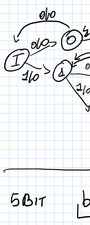


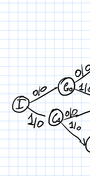










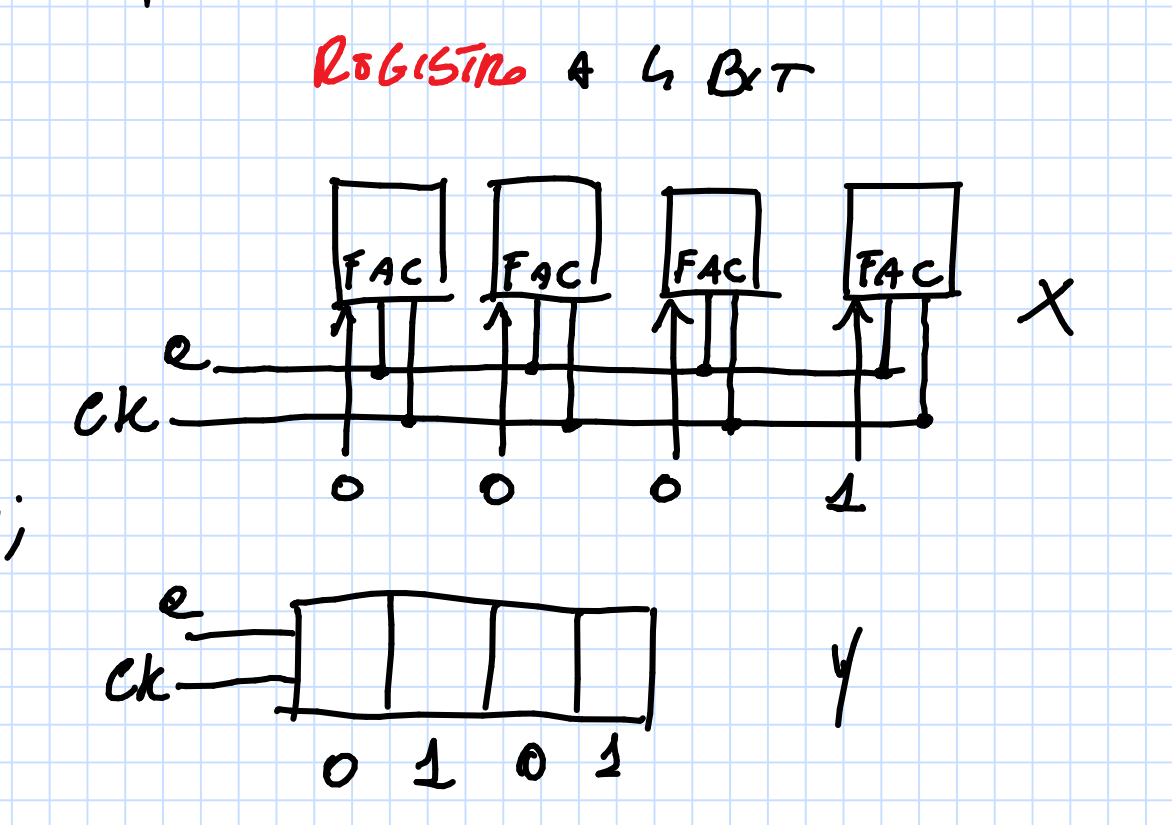


REGISTRI
REGISTRI FUNZIONE
RAM

UN REGISTRO È UN INSIEME DI FLIP FLOP CHE CONDIVIDONO CLOCK E ABILITAZIONE

REGISTRO A 4 BIT

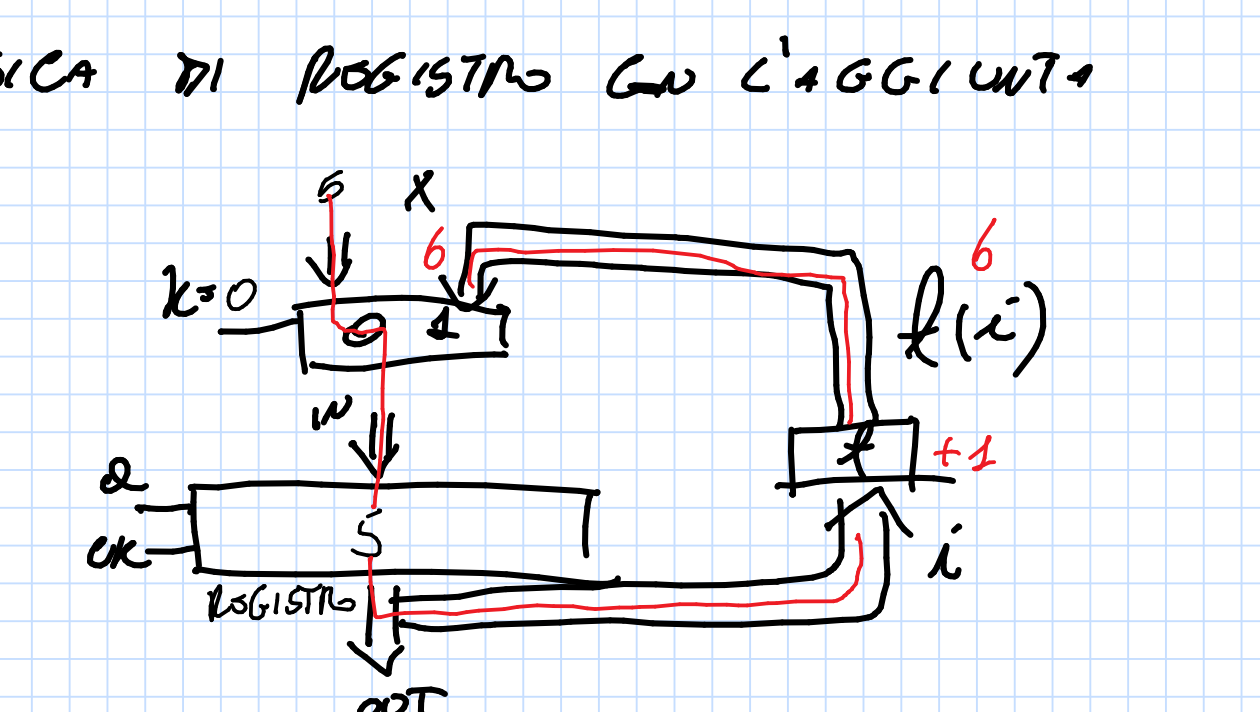
$X = 1; e_x = 1, e_y = 0;$
$Y = 5; e_y = 1; e_x = 0;$



REGISTRI FUNZIONI

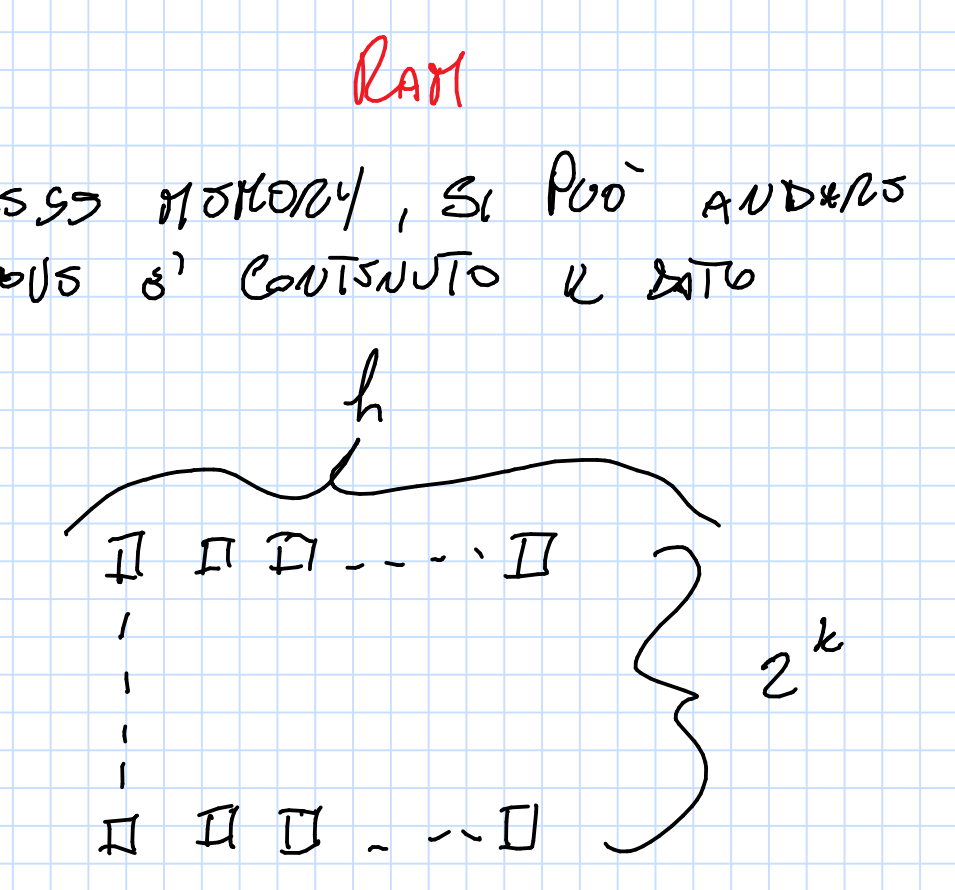
SONO DEI REGISTRI ARRICCHITI, HANNO LA FUNZIONE CLASSICA DI REGISTRO CON L'AGGIUNTA DI FUNZIONI

$e = 1; k = 0;$



RAM

RANDOM ACCESS MEMORY, SI PUÒ ANDARE DIRETTAMENTE AL PUNTO DOVE È CONTENUTO IL DATO



OGNI LINEA DI RAM È IDENTIFICATA DA UNA POSIZIONE CHE VIENE INDICATA CON K BIT

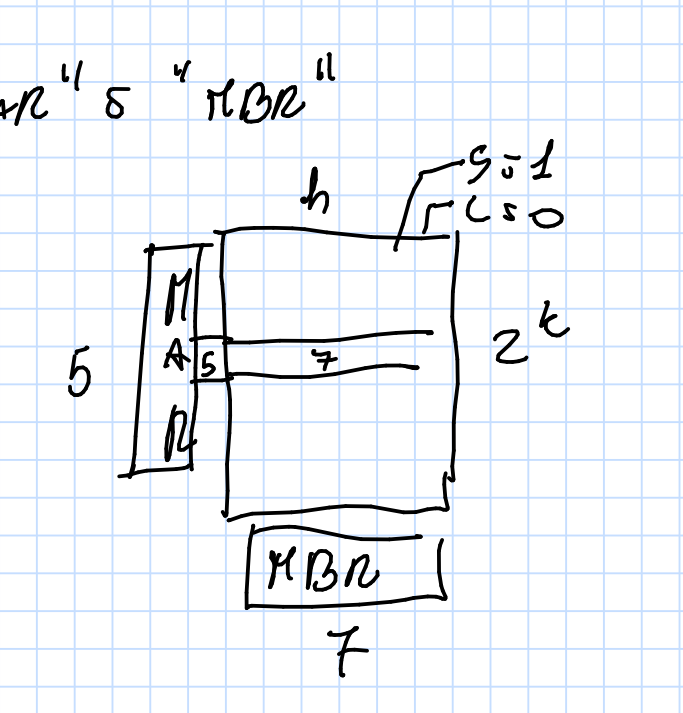
K BIT
0000 -- 0101 — INDIRIZZO DI MEMORIA

LA PAROLA È UN DATO CHE UNA LINEA DI RAM CONTIENE.

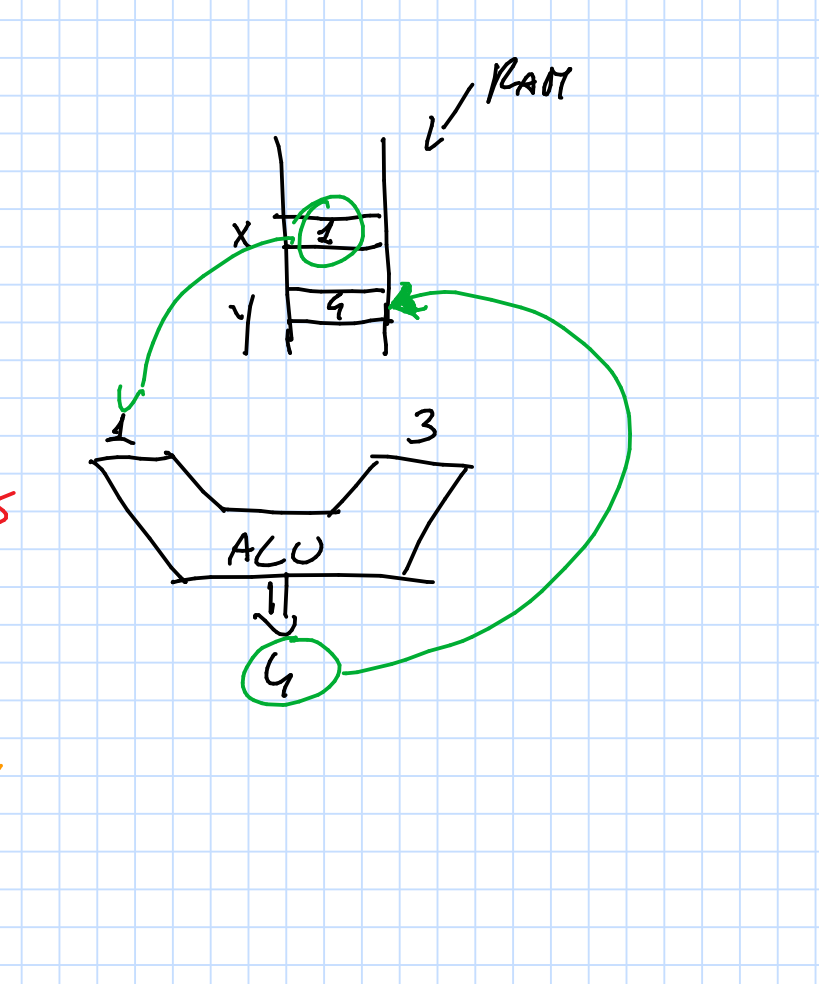
LA RAM È COMPOSTA DA UNA MATRICE DI FLIP FLOP $h \times 2^k$

E CONTIENE DUE REGISTRI "MAR" E "MBR"

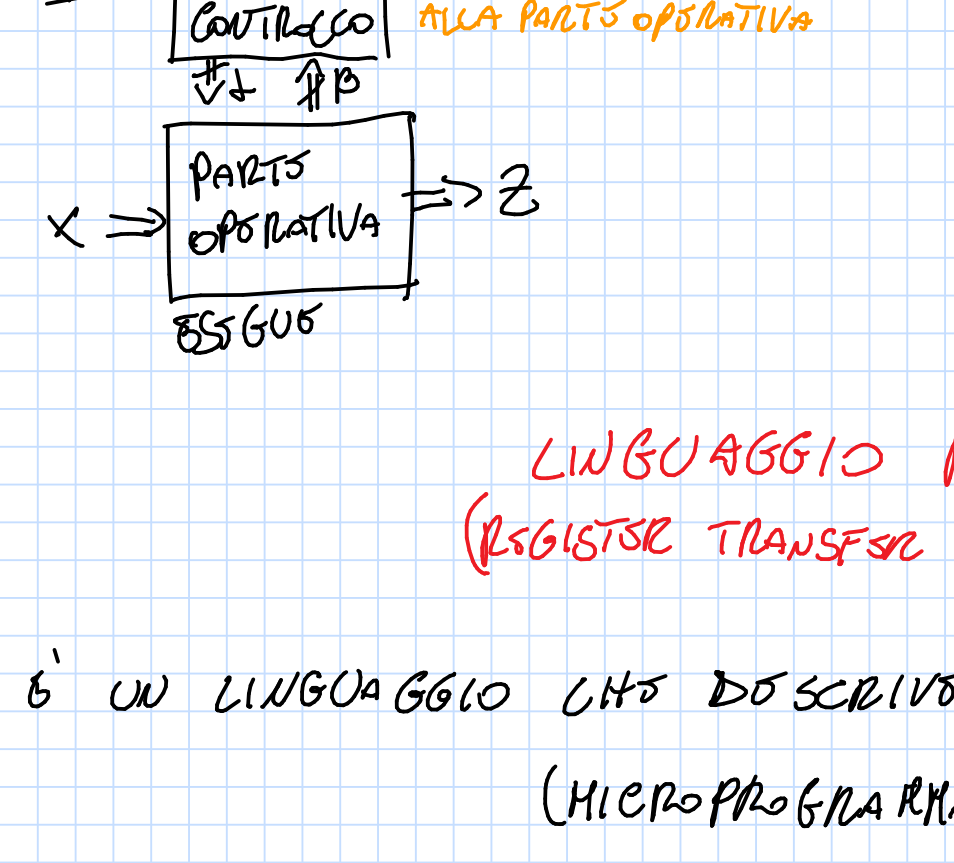
MAR: MEMORY ADDRESS REGISTER
MBR: MEMORY BUFFER REGISTER



Y = X+3 ? { LOAD X, ADD 3, STORE Y }



SISTEMA DI ELABORAZIONE DELLE INFORMAZIONI



LINGUAGGIO RTL
(REGISTER TRANSFER LANGUAGE)

È UN LINGUAGGIO CHE DESCRIVE L'HARDWARE (MICROPROGRAMMAZIONE)

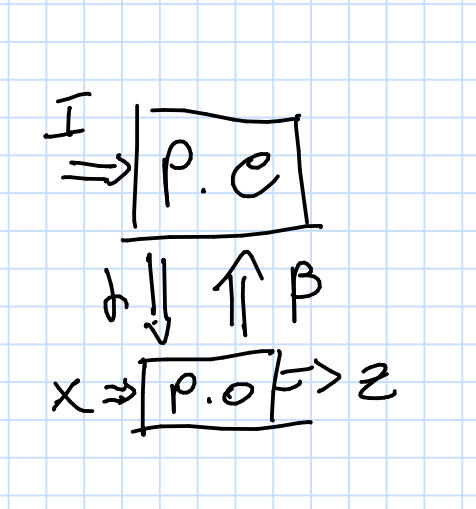
MICROOPERAZIONE = OPERAZIONE ELEMENTARE (SOMMA, COPIA ECC.)
MICROPASSI = UN INSIEME DI MICROOPERAZIONI CHE VENGONO ESEGUITE CONTEMPORANEAMENTE (NELLO STESSO TEMPO DI CLOCK)
MICROSEQUENZA = UNA SEQUENZA DI MICROPASSI

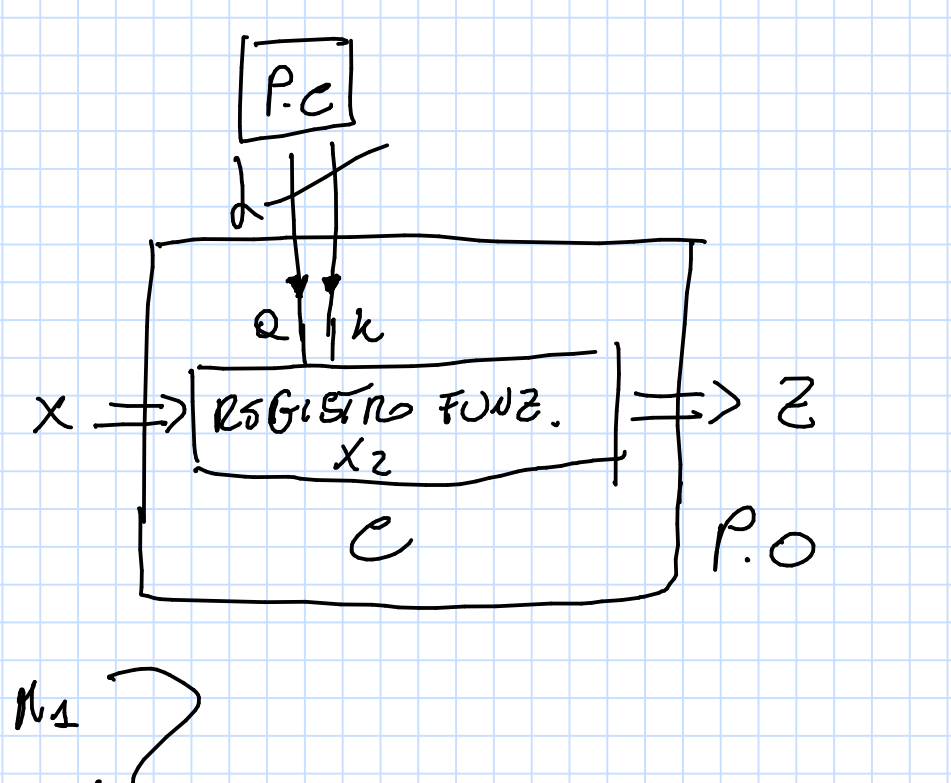
<DATO> —> <REGISTRO>
<REGISTRO> —> <REGISTRO>
<DATO> <OPERATORE> <DATO> —> <REGISTRO>
f (<REGISTRO>) —> <REGISTRO> (UGUALE)

REALIZZARE UN SISTEMA CHE CALCOLA

$Z = X \cdot 4$ X => [5] => X·4

00000101 = 5
00010100 = 5·4

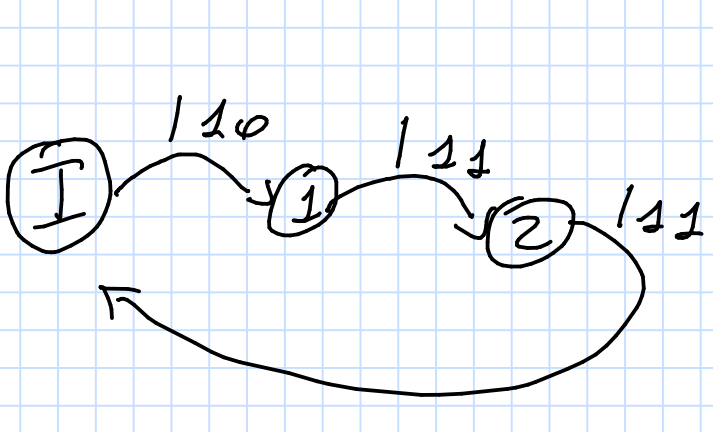




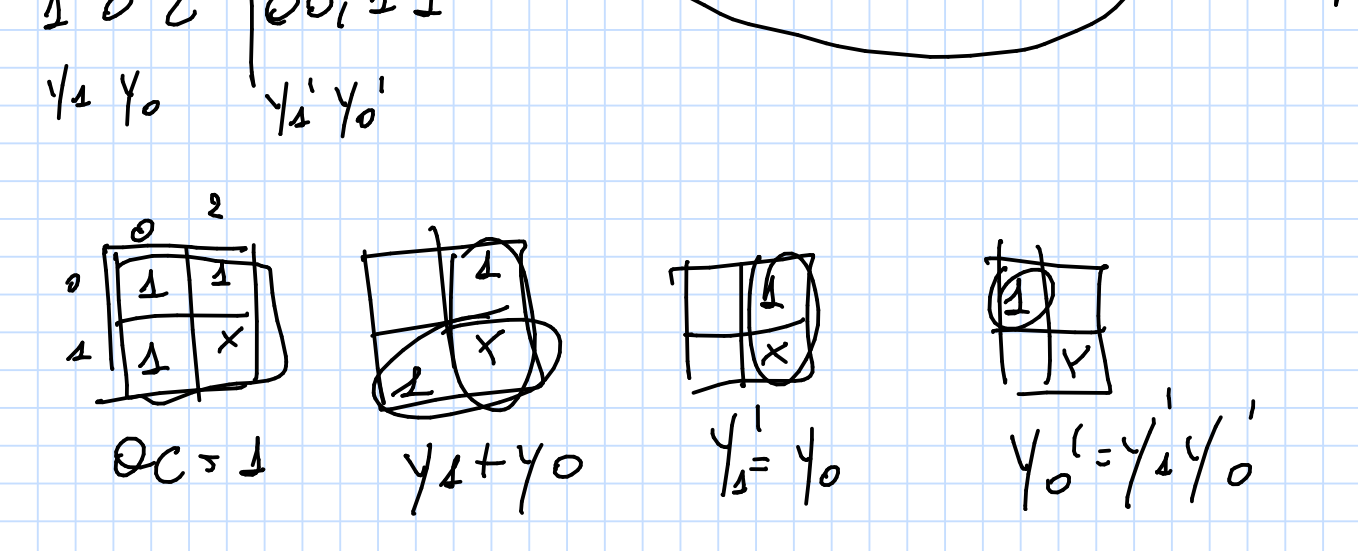
CLOCK
1: DATO —> C ; // X->C M1
2: SHL(C) —> C ; // C·2->C M2
3: SHL(C) —> C ; // C·2->C M2
} M-SEQ

SHL = SHIFT LEFT

00 I | 01,10
01 1 | 10,11
10 2 | 00,11
$Y_1 Y_0$ | $Y_1' Y_0'$

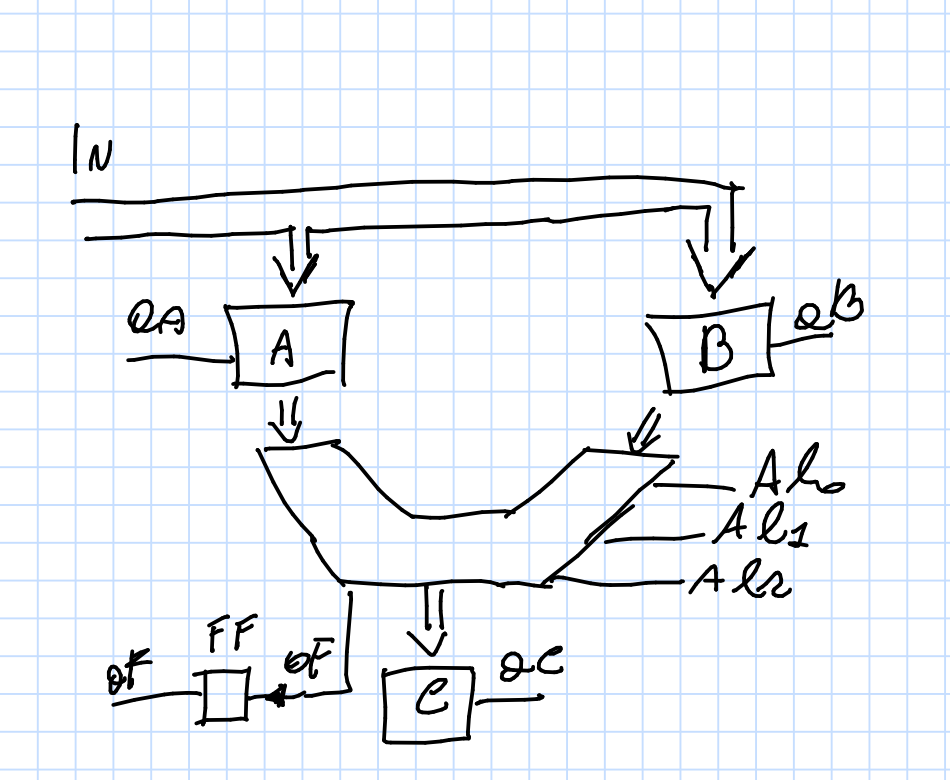


μ	ec	kc
M1	1	0
M2	1	1



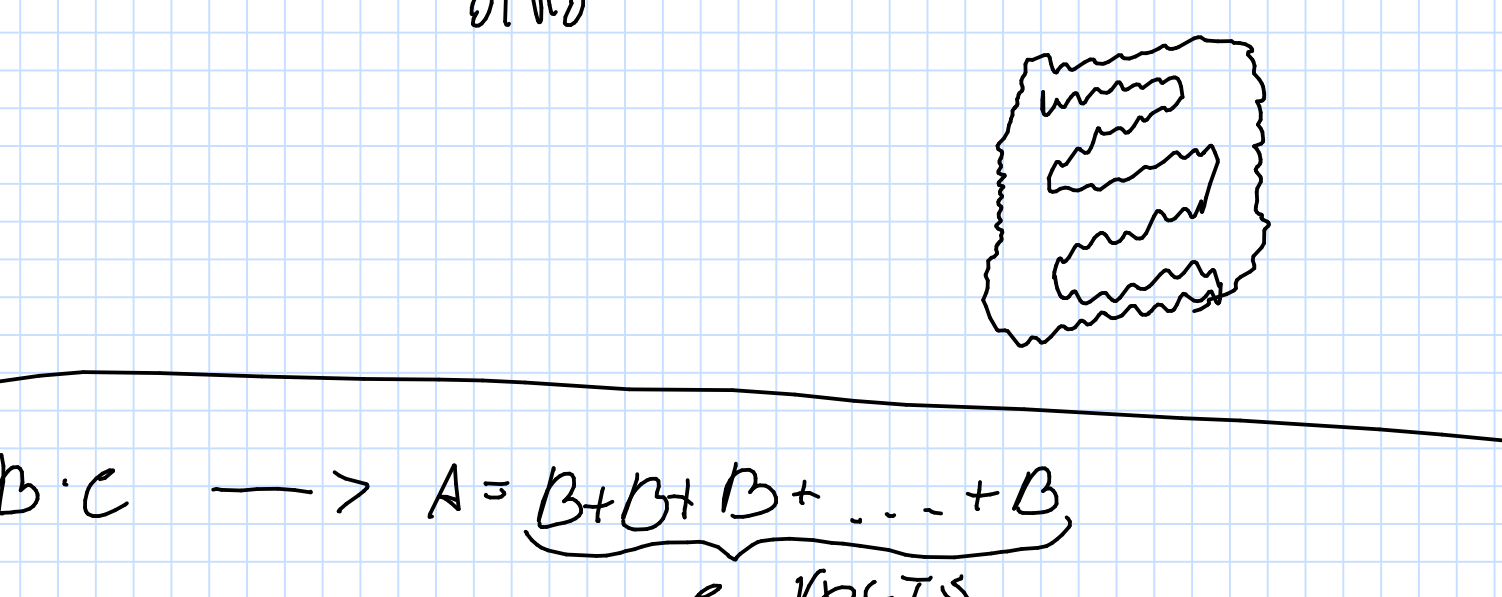
DATI DUE DATI A, B RESTITUISCE IN C A+B SE LA SOMMA È CORRETTA E C=0 SE È SCORRETTA (OVERFLOW)

ASM { C = A+B ; C=0 SE OF



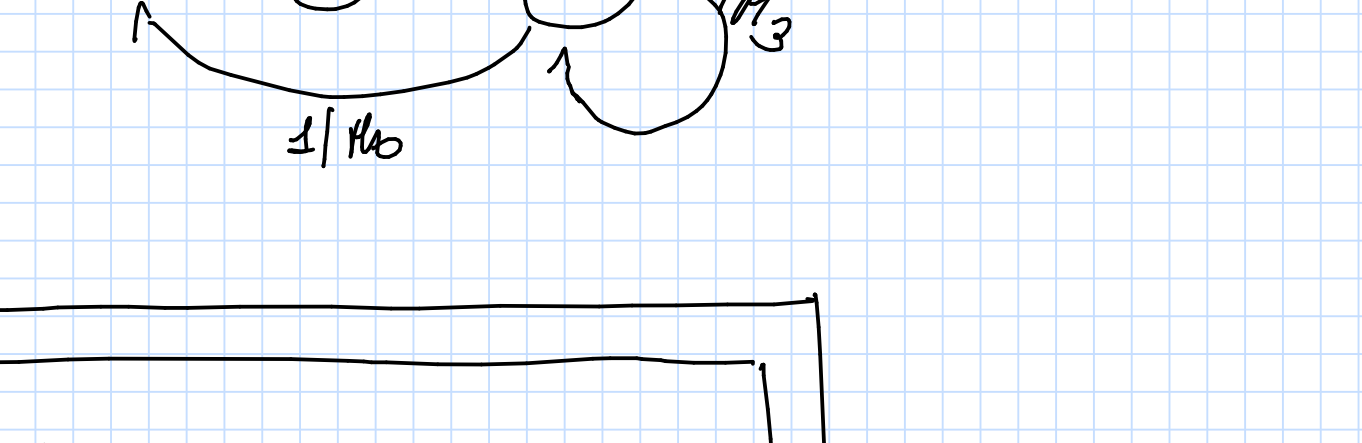
1: DATO —> A ; M1
2: DATO —> B ; M2
3: A+B —> C, OF —> F ; M3
IF (F == 1) THEN
0 —> C ; 0 —> F ; M4
ELSE
∅ ; M0

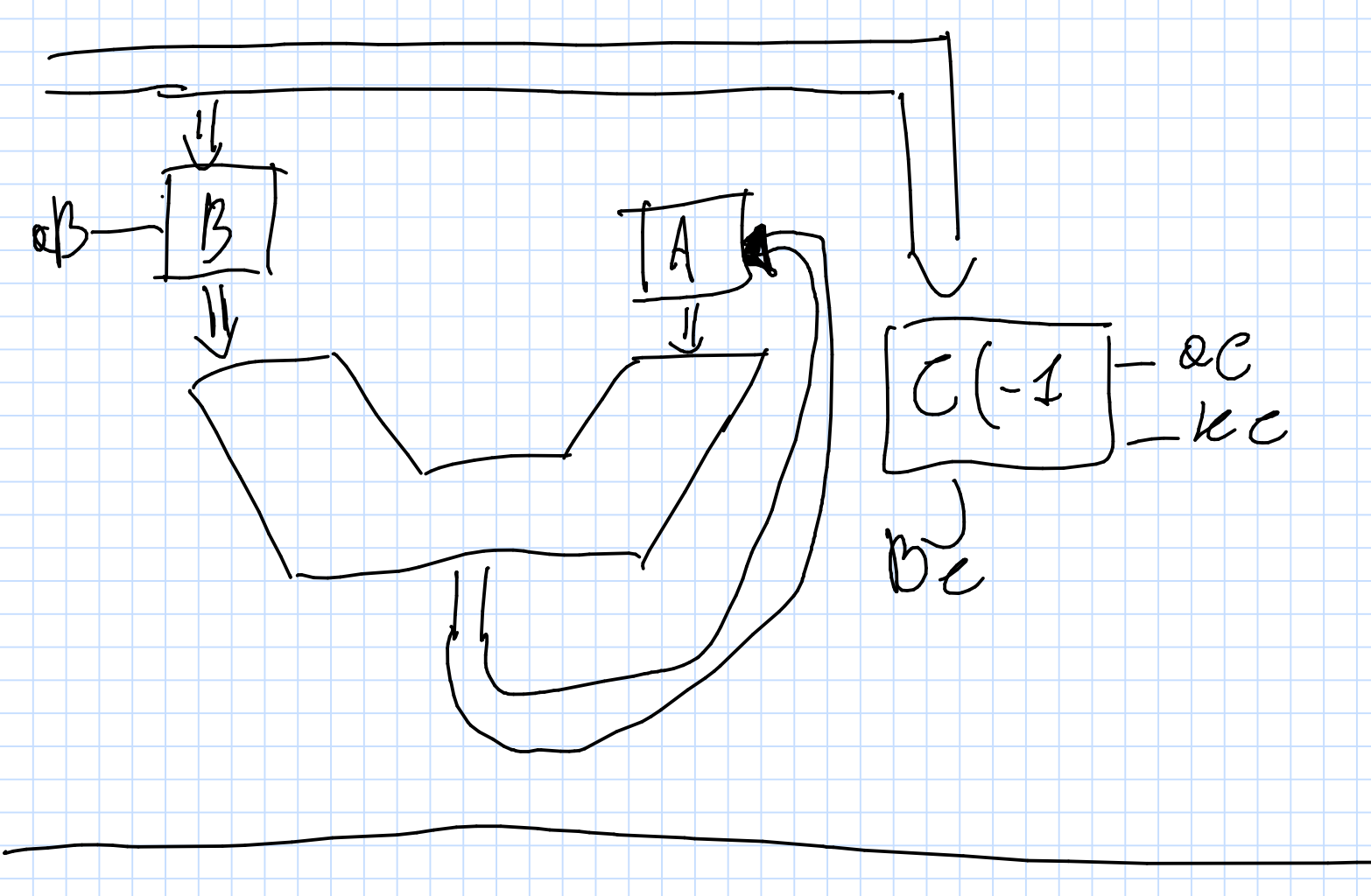
μ	eA	eB	Al0	Al1	Al2	eF	eC	Zc	ef
M0	0	0	X	X	X	0	0	X	X
M1	1	0	X	X	X	0	0	X	X
M2	0	1	X	X	X	0	0	X	X
M3	0	0	1	0	0	1	1	0	0
M4	0	0	X	X	X	1	1	1	1

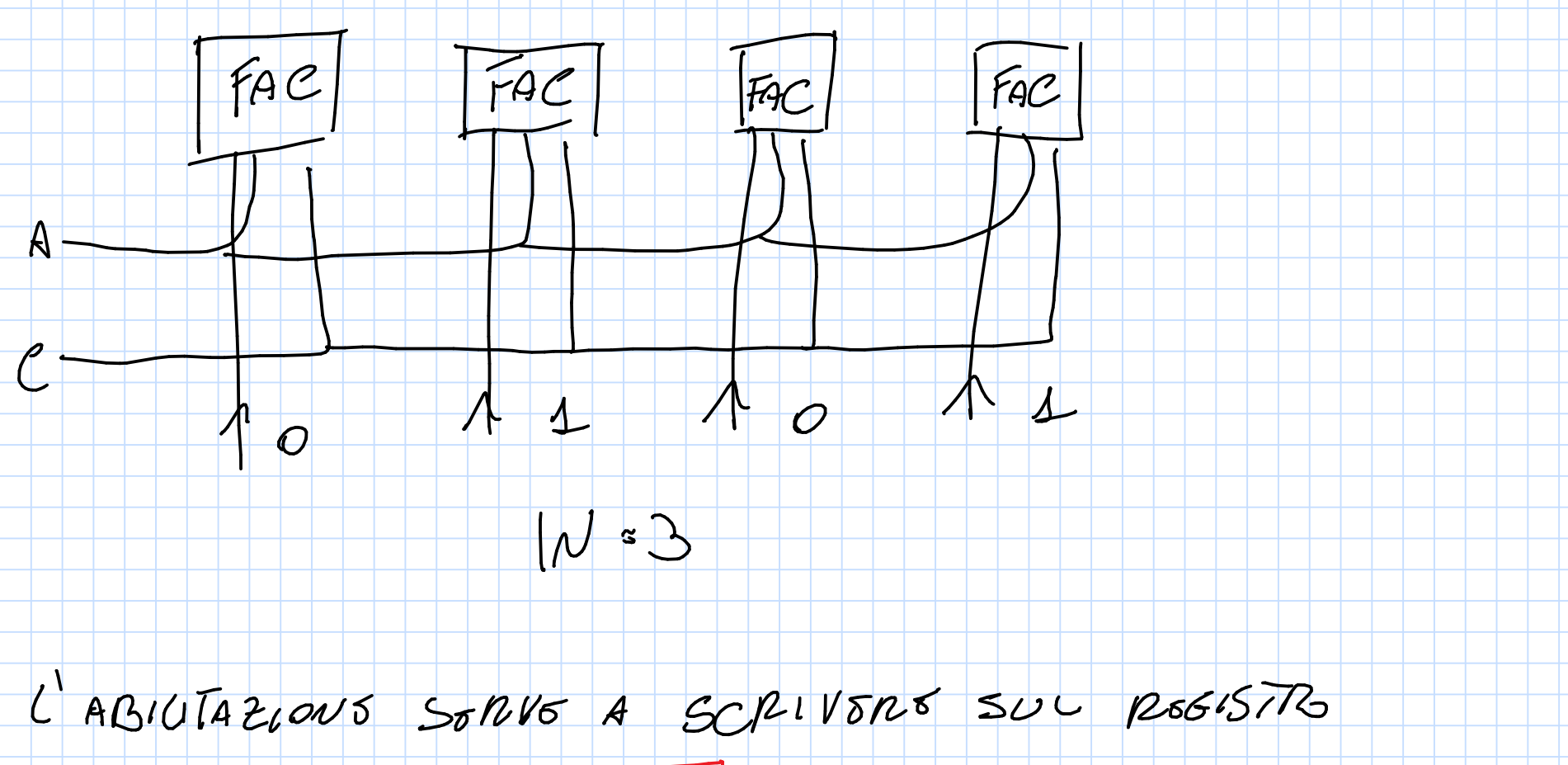


$A = B \cdot C \longrightarrow A = \underbrace{B + B + B + \ldots + B}_{C \text{ VOLTE}}$

DATO —> B; 0 —> A; M1
DATO —> C; M2
CICLO: IF (B == 0) THEN
A+B —> A, DECR (C) —> C, goto ciclo; M3
ELSE:
∅ M0

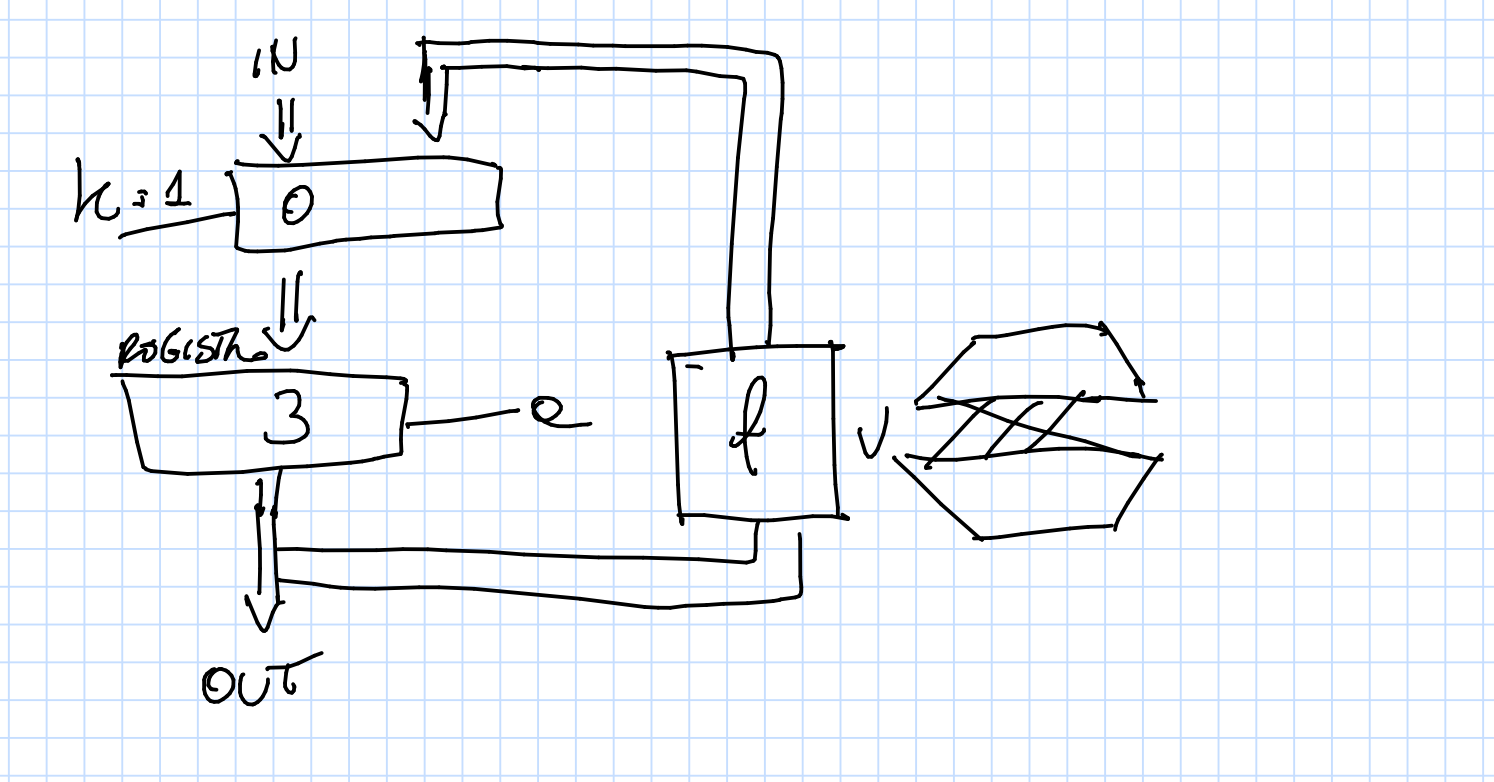






L'ABILITAZIONE SERVE A SCRIVERE SUL REGISTRO

NON PER LEGGERE



$A = B \cdot C$

$A = \underbrace{B + B + \ldots + B}_{C \text{ volte}}$

$I \Rightarrow$ Pc

$X \Rightarrow$ Po $\Rightarrow z$

RTL

DATO → B;

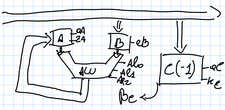
DATO → C, A ← 0;

C: IF (Bc == 0) THEN

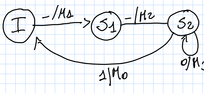
A+B → A, DEC(C) → C, GOTO C;

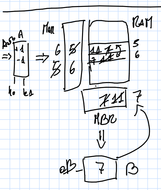
ELSE

ϕ



f:	M	oA	zA	oB	oC	Alo	Ahi	Ale	kc
	M_0	0	X	0	0	X	X	Y	X
	M_1	0	X	1	0	X	X	X	X
	M_2	1	1	0	1	X	X	X	0
	M_3	1	0	0	1	1	0	0	1





DATO → A; M_1

A → MAR; M_2

M[MAR] → MBR; M_3

MBR → B; M_4

WC(A) → A; M_5

A → MAR; M_2

M[MAR] → MBR; M_3

16 colpi di clock

DEC(A) → A; M_6

A → MAR; M_2

M[MAR] → MBR; M_3

WC(A) → A; M_6

A → MAR; M_2

B → MBR; M_7

MBR → M[MAR]; M_5

A → MAR; M_2

B → MBR; M_8

MBR → M[MAR]; M_7

⇓

DATO → A;

A → MAR, WC(A) → A;

M[MAR] → MBR, A → MAR, DEC(A) → A;

MBR → B, M[MAR], A → MAR, WC(A) → A;

MBR → M[MAR], B → MBR, A → MAR;

MBR → M[MAR];

DATO → A;

DATO → B;

A+B → C, OF → FF;

IF FF == 1 THEN

0 → C, 0 → FF;

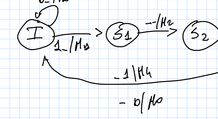
ELSE

ϕ

f:

ID:	ϕ

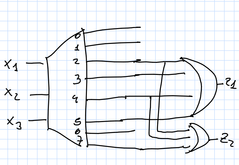
I_1	DATO → A; M_1
	DATO → B; M_2
	A+B → C, OF → FF; M_3
	IF FF == 1 THEN
	0 → C, 0 → FF; M_4
	ELSE
	ϕ M0

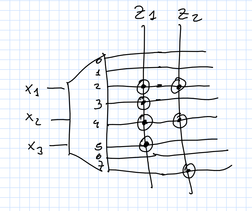


CA RCM

LA RCM È UNA RAPPRESENTAZIONE MATRICIALE DI UNA TABELLA DI VERITÀ

X_1	X_2	X_3	Z_1	Z_2
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1





0 0
0 0
1 1
1 0
1 1
1 0
0 0
0 1

IL CALCOLATORE

UN CALCOLATORE È UN PARTICOLARE SISTEMA DI ELABORAZIONE DELL'INFORMAZIONE ED È UNA MACCHINA CHE HA UN PROGRAMMA MEMORIZZATO.

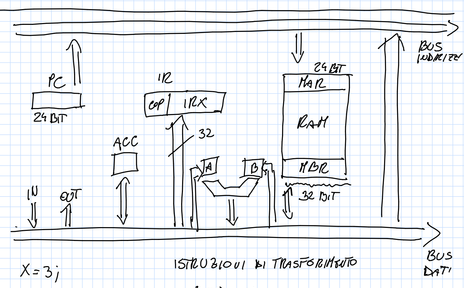
UN PROGRAMMA VIENE CARICATO IN RAM E PUÒ FINIRE IN DUE STATI

FETCH (PRELIEVO) — ESECUTE (ESECUZIONE)

DURANTE LA FASE DI FETCH VIENE PRELEVATA DALLA RAM L'ESECUZIONE DA ESEGUIRE E SI PASSA ALLA FASE DI ESECUTE CHE ESEGUE L'ISTRUZIONE E POI RITORNA IN FETCH

IL PC (PROGRAM COUNTER) TIENE CONTO DELLA POSIZIONE CHE STA PER ESSERE ESEGUITA

L'IR (ISTRUCTION REGISTER) CONTIENE L'ISTRUZIONE CHE DEVE ESSERE ESEGUITA



X = 3;
Y = 4;
Z = X+Y;

IN
STORE 2001(X)
IN
STORE 2373(Y)
LOAD 2001
ADD 2373
STORE 5238

ISTRUZIONI DI TRASFERIMENTO
- LOAD X
- STORE X

ISTRUZIONI ARITMETICHE
- ADD X
- SUB X

ISTRUZIONI INPUT/OUTPUT
- IN X => PORTA IL DATO ALL'ACCUMULATORE
- OUT X => PRENDE IL DATO DALL'ACCUMULATORE E LO PORTA IN OUT

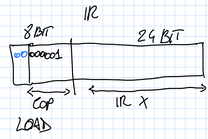
ISTRUZIONI DI CONTROLLO
JUMP X
JUMPZ X

LOAD X DIRETTO M[X] -> ACC — LOAD 2001 [3] ACC

LOAD# X IMMEDIATO X -> ACC — LOAD#2001 [2001] ACC

LOAD@ X INDIRETTO M[M[X]] -> ACC LOAD@ 2001

LOAD %X RELATIVO M[PC+X] -> ACC LOAD%2001 M[5+2001]



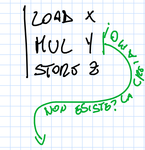
LOAD 1 00
LOAD# X 10
LOAD@ X 01
LOAD% X 11

LOAD X M[X] -> ACC
IRX -> MAR;
M[MAR] -> MBR;
MBR -> ACC;

LOAD@ X M[M[X]] -> ACC
IRX -> MAR;
M[MAR] -> MBR;
MBR -> MAR;
M[MAR] -> MBR;
MBR -> ACC

LOAD#X
IRX -> ACC

LOAD% X M[X+PC] -> ACC
IRX -> A;
PC -> B;
A+B -> MAR;
M[MAR] -> MBR;
MBR -> ACC;



MUL X => M[X] · ACC => ACC

IRX -> MAR;
M[MAR] -> MBR;
MBR -> A;
B ← 0
A ← M[X]
A+B -> B
IF (OR(ACC) == 1) THEN
    A+B -> B, DEC(ACC) -> ACC, GOTO C;
ELSE
    B -> ACC;

FETCH 00000000
PC -> MAR
M[MAR] -> MBR
MBR -> IR

FETCH -> EXECUTE
AZZERARE L'IR

MACCHINA A REGISTRI

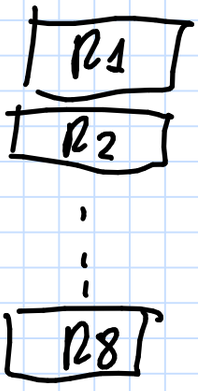
x	5	751
y	3	2031
z		5327

x

y

$z = (x+2) \cdot (y-3);$

```
IN
STORE 751
IN
STORE 2031
---
LOAD 751, R1
ADD # 2, R1
STORE 5327
LOAD 2031, R2
SUB # 3, R2
MUL R1 R2
STORE 5327
```



LOAD x M[x] -> ACC

LOAD R1, x M[x] -> R1

LOAD R1, x

IRx -> MAR;

M[MAR] -> MBR;

MBR -> R1;

LOAD R2, x

IRx -> MAR;

M[MAR] -> MBR;

MBR -> R2

MACCHINA INTEL (ASSEMBLY)

REGISTRI

- EAX
- EBX
- ECX
- EDX
- EDI
- ESI

} USO GENERALE

- EBP
- ESP

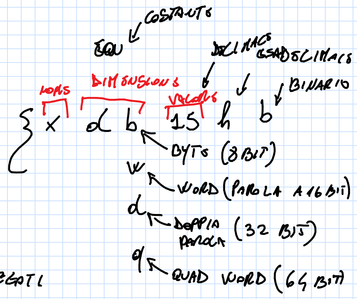
EBP: BASE POINTER

ESP: STACK POINTER

```
nasm -f elf32 prova.asm
ld -m elf_i386 prova.o -o PROVA
./PROVA
```

SECTION .DATA

} DATI INIZIALIZZATI



SECTION .BSS

} DATI NON INIZIALIZZATI

y RESW 1 ← NUMERO DI VARIABILI DA INIZIALIZZARE

SECTION .TEXT

<CODICE>

global _START

_START:

MOV (COPIA)

- <REG> <REG>
- <REG> <MEM>
- <MEM> <REG>
- <REG> <IMM>
- <MEM> <IMM>

ZF = 1 SE IL RISULTATO DELLA ALU È 0

CF = 1 SE C'È UN RIPORTO CHE ECCEDE IL NUMERO DI BIT USATI PER LA RAPPRESENTAZIONE DEI NUMERI

OF = 1 SE C'È STATO OVERFLOW

SF = 1 SE IL RISULTATO È NEGATIVO

	JZ	JNZ	JC	JNC	JO	JNO	JS	JNS
SALTA	ZF=1	ZF=0	CF=1	CF=0	OF=1	OF=0	SF=1	SF=0

ISTRUZIONI DI SALTO

CON SEGNO				
JGE	<ETICHETTA>	>=	SF = OF	JNL <ETICHETTA>
JG	<ETICHETTA>	>	ZF = 0 AND SF = OF	JNLE <ETICHETTA>
JLE	<ETICHETTA>	<=	ZF = 1 AND SF != OF	JNG <ETICHETTA>
JL	<ETICHETTA>	<	SF != OF	JNGE <ETICHETTA>
JE	<ETICHETTA>	=	ZF = 1	

SENZA SEGNO				
JAE	<ETICHETTA>	>=	CF = 0	JNB <ETICHETTA>
JA	<ETICHETTA>	>	CF = 0 AND ZF = 0	JNBE <ETICHETTA>
JBE	<ETICHETTA>	<=	CF = 1 OR ZF = 1	JNA <ETICHETTA>
JB	<ETICHETTA>	<	CF = 1	JNAE <ETICHETTA>

ISTRUZIONI LOGICHE

OPERATORI BINARI: AND, OR, XOR <OP1> <OP2>

AND AX, BX; AX AND BX → AX

NOT <OP1>: NOT BIT A BIT

NEG <OP1>: NOT (<OP1>) + 1 = -<OP1>

SHIFT LOGICO	SHIFT ARITMETICO	ROTAZIONE
SHL r/mem, imm/CL	SAL r/mem, imm/CL	ROL ...
SHR r/mem, imm/CL	SAR r/mem, imm/CL	ROR ...

MOLTIPLICAZIONE

MOLTIPLICAZIONE CON SEGNO

IMUL <OP>

MOLTIPLICAZIONE SENZA

MUL <OP>

SE <OP>

OTTIMIZZAZIONI

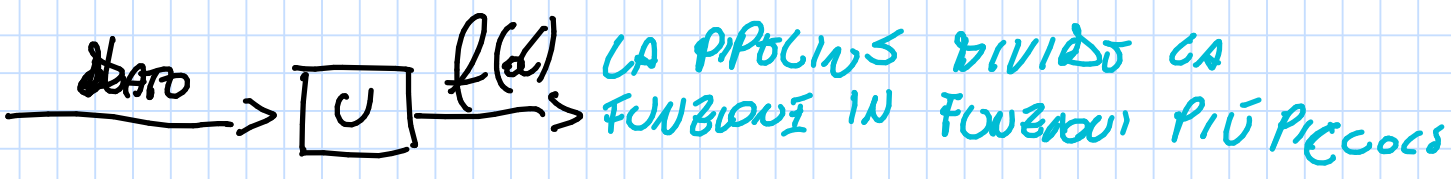
I PROCESSORI MODERNI UTILIZZANO ALCUNI PROCESSI DI PARALLIZZAZIONE HARDWARE TRA CUI

- PIPELINE
- PARALLELISMO

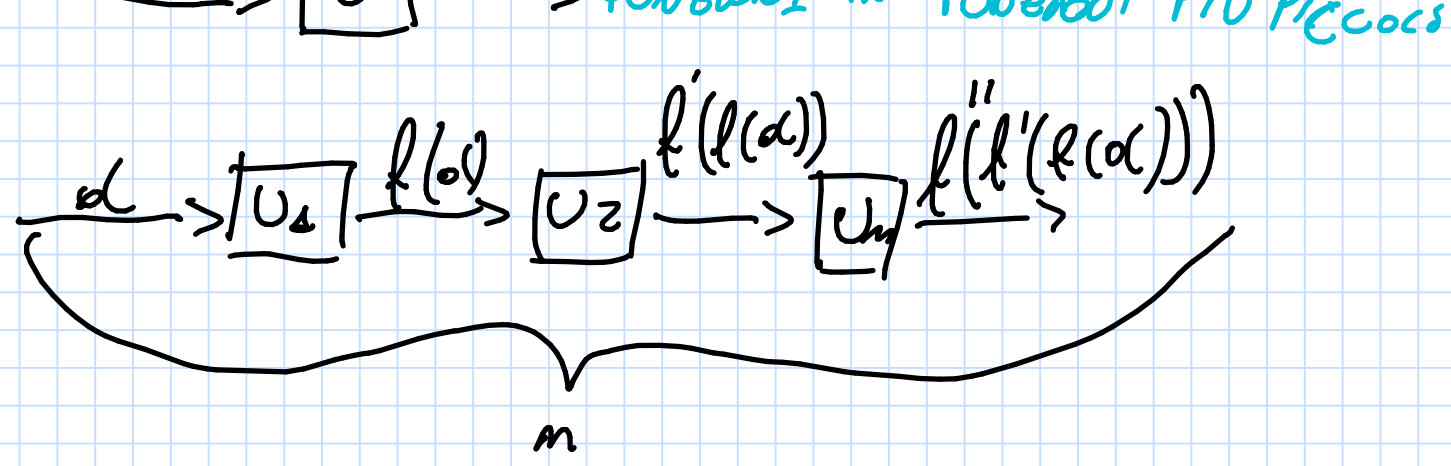
} NEI CALCOLATORI ATTUALI VENGONO UTILIZZATI CONTEMPORANEAMENTE

PIPELINE

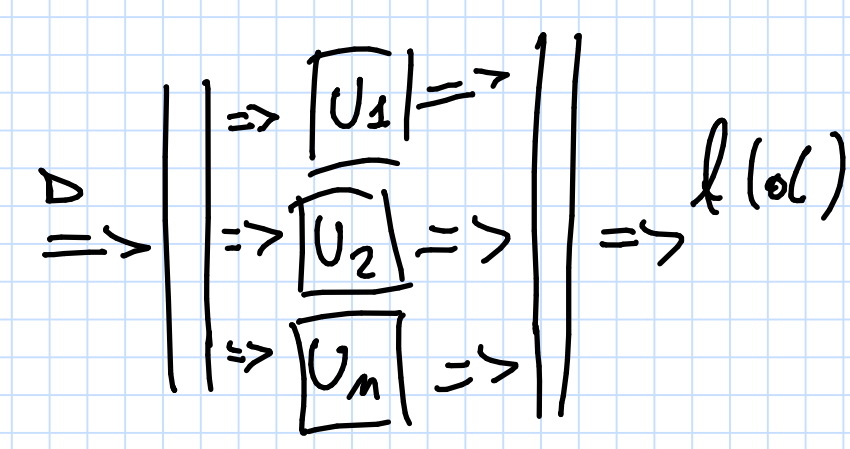
LA PIPELINE E' UN SISTEMA MOLTO SIMILE AD UNA CATENA DI MONTAGGIO



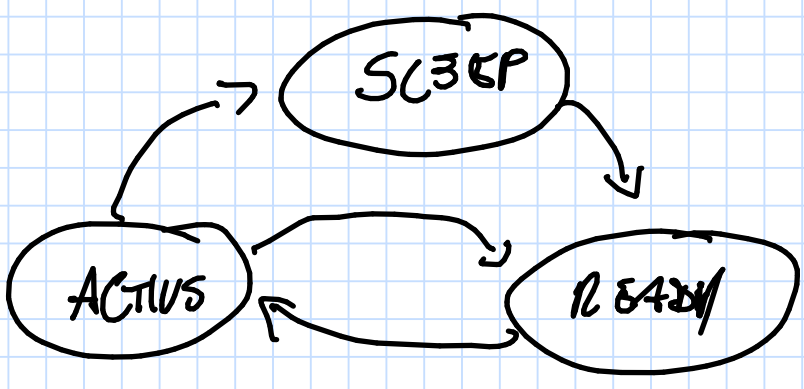
LA PIPELINE DIVIDE LA FUNZIONE IN FUNZIONI PIÙ PICCOLE



PARALLELISMO

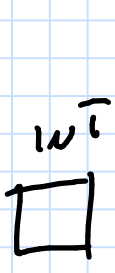


PROCESSI



OGNI PROGRAMMA UTILIZZA TUTTI I REGISTRI, QUINDI PER PASSARE DA UN PROGRAMMA ALL'ALTRO DOVO PRIMA SALVARE I CONTENUTI DI QUEI REGISTRI. L'INSIEME DI TUTTI I REGISTRI SI CHIAMA STATO DEL PROCESSO

IL MECCANISMO E'




```
FETCH:
    if  INT = 1
        <SALVA STATO>;
        <GESTIONE INT> -> PC;
        MASK <- 1;
    ELSE
        PC -> MAR, INC(PC) -> PC;
        M[MAR] -> MBR;
        MBR -> IR;
```

DMA

CHE STA PER "DIRECT MEMORY ACCESS"

ED E' UN UNITA' CHE SI OCCUPA DI GESTIRE LE PERIFERICHE DI I/O

QUINDI SERVE A TRASFERIRE I DATI DA E VERSO LA MEMORIA

RAM --> DISCO
DISCO --> RAM