ARHITECTURA SISTEMELOR DE CALCUL SEMINAR 0x00

NOTIȚE SUPORT SEMINAR

Cristian Rusu

CONVERSII, EX 1

				0x1111

hexa: 0x1111

binar:

baza 4:

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

				0x1111

hexa: 0x1111

binar: 0001 0001 0001 0001

baza 4:

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

				0x1111

hexa: 0x1111

binar: 0001 0001 0001 0001

baza 4: 00 01 00 01 00 01 00 01

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

				0x1111

hexa: 0x1111

binar: 0001 0001 0001 0001

baza 4: 00 01 00 01 00 01 00 01 = 01010101

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

				0x1111

hexa: 0x1111

binar: 0001 0001 0001 0001

baza 4: 00 01 00 01 00 01 00 01 = 01010101

baza 8: 0 001 000 100 010 001

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

				0x1111

hexa: 0x1111

binar: 0001 0001 0001 0001

baza 4: 00 01 00 01 00 01 00 01 = 01010101

baza 8: 0 001 000 100 010 001 = 10421

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

				0x1111

hexa: 0x1111

binar: 0001 0001 0001 0001

baza 4: 00 01 00 01 00 01 00 01 = 01010101

baza 8: 0 001 000 100 010 001 = 10421

baza 10: 4369

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa:

baza 4:

baza 8:

baza 10:

0$_{hex}$	=	0$_{dec}$	=	0$_{oct}$	0	0	0	0
1$_{hex}$	=	1$_{dec}$	=	1$_{oct}$	0	0	0	1
2$_{hex}$	=	2$_{dec}$	=	2$_{oct}$	0	0	1	0
3$_{hex}$	=	3$_{dec}$	=	3$_{oct}$	0	0	1	1
4$_{hex}$	=	4$_{dec}$	=	4$_{oct}$	0	1	0	0
5$_{hex}$	=	5$_{dec}$	=	5$_{oct}$	0	1	0	1
6$_{hex}$	=	6$_{dec}$	=	6$_{oct}$	0	1	1	0
7$_{hex}$	=	7$_{dec}$	=	7$_{oct}$	0	1	1	1
8$_{hex}$	=	8$_{dec}$	=	10$_{oct}$	1	0	0	0
9$_{hex}$	=	9$_{dec}$	=	11$_{oct}$	1	0	0	1
A$_{hex}$	=	10$_{dec}$	=	12$_{oct}$	1	0	1	0
B$_{hex}$	=	11$_{dec}$	=	13$_{oct}$	1	0	1	1
C$_{hex}$	=	12$_{dec}$	=	14$_{oct}$	1	1	0	0
D$_{hex}$	=	13$_{dec}$	=	15$_{oct}$	1	1	0	1
E$_{hex}$	=	14$_{dec}$	=	16$_{oct}$	1	1	1	0
F$_{hex}$	=	15$_{dec}$	=	17$_{oct}$	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4:

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

1111 1111 0000 0000

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00

baza 8:

baza 10:

0$_{hex}$	=	0$_{dec}$	=	0$_{oct}$	0	0	0	0
1$_{hex}$	=	1$_{dec}$	=	1$_{oct}$	0	0	0	1
2$_{hex}$	=	2$_{dec}$	=	2$_{oct}$	0	0	1	0
3$_{hex}$	=	3$_{dec}$	=	3$_{oct}$	0	0	1	1
4$_{hex}$	=	4$_{dec}$	=	4$_{oct}$	0	1	0	0
5$_{hex}$	=	5$_{dec}$	=	5$_{oct}$	0	1	0	1
6$_{hex}$	=	6$_{dec}$	=	6$_{oct}$	0	1	1	0
7$_{hex}$	=	7$_{dec}$	=	7$_{oct}$	0	1	1	1
8$_{hex}$	=	8$_{dec}$	=	10$_{oct}$	1	0	0	0
9$_{hex}$	=	9$_{dec}$	=	11$_{oct}$	1	0	0	1
A$_{hex}$	=	10$_{dec}$	=	12$_{oct}$	1	0	1	0
B$_{hex}$	=	11$_{dec}$	=	13$_{oct}$	1	0	1	1
C$_{hex}$	=	12$_{dec}$	=	14$_{oct}$	1	1	0	0
D$_{hex}$	=	13$_{dec}$	=	15$_{oct}$	1	1	0	1
E$_{hex}$	=	14$_{dec}$	=	16$_{oct}$	1	1	1	0
F$_{hex}$	=	15$_{dec}$	=	17$_{oct}$	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8: 1 111 111 100 000 000

baza 10:

0hex	=	0dec	=	0oct	0	0	0	0
1hex	=	1dec	=	1oct	0	0	0	1
2hex	=	2dec	=	2oct	0	0	1	0
3hex	=	3dec	=	3oct	0	0	1	1
4hex	=	4dec	=	4oct	0	1	0	0
5hex	=	5dec	=	5oct	0	1	0	1
6hex	=	6dec	=	6oct	0	1	1	0
7hex	=	7dec	=	7oct	0	1	1	1
8hex	=	8dec	=	10oct	1	0	0	0
9hex	=	9dec	=	11oct	1	0	0	1
Ahex	=	10dec	=	12oct	1	0	1	0
Bhex	=	11dec	=	13oct	1	0	1	1
Chex	=	12dec	=	14oct	1	1	0	0
Dhex	=	13dec	=	15oct	1	1	0	1
Ehex	=	14dec	=	16oct	1	1	1	0
Fhex	=	15dec	=	17oct	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8: 1 111 111 100 000 000 = 177400

baza 10:

0$_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
1$_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
2$_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
3$_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
4$_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
5$_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
6$_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
7$_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
8$_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
9$_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
A$_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
B$_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
C$_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
D$_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
E$_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
F$_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 1

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8: 1 111 111 100 000 000 = 177400

baza 10: 65280

0_{hex}	=	0_{dec}	=	0_{oct}	0	0	0	0
1_{hex}	=	1_{dec}	=	1_{oct}	0	0	0	1
2_{hex}	=	2_{dec}	=	2_{oct}	0	0	1	0
3_{hex}	=	3_{dec}	=	3_{oct}	0	0	1	1
4_{hex}	=	4_{dec}	=	4_{oct}	0	1	0	0
5_{hex}	=	5_{dec}	=	5_{oct}	0	1	0	1
6_{hex}	=	6_{dec}	=	6_{oct}	0	1	1	0
7_{hex}	=	7_{dec}	=	7_{oct}	0	1	1	1
8_{hex}	=	8_{dec}	=	10_{oct}	1	0	0	0
9_{hex}	=	9_{dec}	=	11_{oct}	1	0	0	1
A_{hex}	=	10_{dec}	=	12_{oct}	1	0	1	0
B_{hex}	=	11_{dec}	=	13_{oct}	1	0	1	1
C_{hex}	=	12_{dec}	=	14_{oct}	1	1	0	0
D_{hex}	=	13_{dec}	=	15_{oct}	1	1	0	1
E_{hex}	=	14_{dec}	=	16_{oct}	1	1	1	0
F_{hex}	=	15_{dec}	=	17_{oct}	1	1	1	1

CONVERSII, EX 2

0xFEED

hexa: 0xFEED

binar:

baza 4:

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

				0xFEED

hexa: 0xFEED

binar: 1111 1110 1110 1101

baza 4:

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

				0xFEED

hexa: 0xFEED

binar: 1111 1110 1110 1101

baza 4: 11 11 11 10 11 10 11 01

baza 8:

baza 10:

0hex	=	0dec	=	0oct	0	0	0	0
1hex	=	1dec	=	1oct	0	0	0	1
2hex	=	2dec	=	2oct	0	0	1	0
3hex	=	3dec	=	3oct	0	0	1	1
4hex	=	4dec	=	4oct	0	1	0	0
5hex	=	5dec	=	5oct	0	1	0	1
6hex	=	6dec	=	6oct	0	1	1	0
7hex	=	7dec	=	7oct	0	1	1	1
8hex	=	8dec	=	10oct	1	0	0	0
9hex	=	9dec	=	11oct	1	0	0	1
Ahex	=	10dec	=	12oct	1	0	1	0
Bhex	=	11dec	=	13oct	1	0	1	1
Chex	=	12dec	=	14oct	1	1	0	0
Dhex	=	13dec	=	15oct	1	1	0	1
Ehex	=	14dec	=	16oct	1	1	1	0
Fhex	=	15dec	=	17oct	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

				0xFEED

hexa: 0xFEED

binar: 1111 1110 1110 1101

baza 4: 11 11 11 10 11 10 11 01 = 33323231

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

				0xFEED

hexa: 0xFEED

binar: 1111 1110 1110 1101

baza 4: 11 11 11 10 11 10 11 01 = 33323231

baza 8: 1 111 111 011 101 101

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

				0xFEED

hexa: 0xFEED

binar: 1111 1110 1110 1101

baza 4: 11 11 11 10 11 10 11 01 = 33323231

baza 8: 1 111 111 011 101 101 = 177355

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

				0xFEED

hexa: 0xFEED

binar: 1111 1110 1110 1101

baza 4: 11 11 11 10 11 10 11 01 = 33323231

baza 8: 1 111 111 011 101 101 = 177355

baza 10: -275

0_{hex}	=	0_{dec}	=	0_{oct}	0	0	0	0
1_{hex}	=	1_{dec}	=	1_{oct}	0	0	0	1
2_{hex}	=	2_{dec}	=	2_{oct}	0	0	1	0
3_{hex}	=	3_{dec}	=	3_{oct}	0	0	1	1
4_{hex}	=	4_{dec}	=	4_{oct}	0	1	0	0
5_{hex}	=	5_{dec}	=	5_{oct}	0	1	0	1
6_{hex}	=	6_{dec}	=	6_{oct}	0	1	1	0
7_{hex}	=	7_{dec}	=	7_{oct}	0	1	1	1
8_{hex}	=	8_{dec}	=	10_{oct}	1	0	0	0
9_{hex}	=	9_{dec}	=	11_{oct}	1	0	0	1
A_{hex}	=	10_{dec}	=	12_{oct}	1	0	1	0
B_{hex}	=	11_{dec}	=	13_{oct}	1	0	1	1
C_{hex}	=	12_{dec}	=	14_{oct}	1	1	0	0
D_{hex}	=	13_{dec}	=	15_{oct}	1	1	0	1
E_{hex}	=	14_{dec}	=	16_{oct}	1	1	1	0
F_{hex}	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa:

baza 4:

baza 8:

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4:

baza 8:

baza 10:

0$_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
1$_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
2$_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
3$_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
4$_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
5$_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
6$_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
7$_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
8$_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
9$_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
A$_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
B$_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
C$_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
D$_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
E$_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
F$_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00

baza 8:

baza 10:

0hex	=	0dec	=	0oct	0	0	0	0
1hex	=	1dec	=	1oct	0	0	0	1
2hex	=	2dec	=	2oct	0	0	1	0
3hex	=	3dec	=	3oct	0	0	1	1
4hex	=	4dec	=	4oct	0	1	0	0
5hex	=	5dec	=	5oct	0	1	0	1
6hex	=	6dec	=	6oct	0	1	1	0
7hex	=	7dec	=	7oct	0	1	1	1
8hex	=	8dec	=	10oct	1	0	0	0
9hex	=	9dec	=	11oct	1	0	0	1
Ahex	=	10dec	=	12oct	1	0	1	0
Bhex	=	11dec	=	13oct	1	0	1	1
Chex	=	12dec	=	14oct	1	1	0	0
Dhex	=	13dec	=	15oct	1	1	0	1
Ehex	=	14dec	=	16oct	1	1	1	0
Fhex	=	15dec	=	17oct	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8:

baza 10:

0$_{hex}$	=	0$_{dec}$	=	0$_{oct}$	0	0	0	0
1$_{hex}$	=	1$_{dec}$	=	1$_{oct}$	0	0	0	1
2$_{hex}$	=	2$_{dec}$	=	2$_{oct}$	0	0	1	0
3$_{hex}$	=	3$_{dec}$	=	3$_{oct}$	0	0	1	1
4$_{hex}$	=	4$_{dec}$	=	4$_{oct}$	0	1	0	0
5$_{hex}$	=	5$_{dec}$	=	5$_{oct}$	0	1	0	1
6$_{hex}$	=	6$_{dec}$	=	6$_{oct}$	0	1	1	0
7$_{hex}$	=	7$_{dec}$	=	7$_{oct}$	0	1	1	1
8$_{hex}$	=	8$_{dec}$	=	10$_{oct}$	1	0	0	0
9$_{hex}$	=	9$_{dec}$	=	11$_{oct}$	1	0	0	1
A$_{hex}$	=	10$_{dec}$	=	12$_{oct}$	1	0	1	0
B$_{hex}$	=	11$_{dec}$	=	13$_{oct}$	1	0	1	1
C$_{hex}$	=	12$_{dec}$	=	14$_{oct}$	1	1	0	0
D$_{hex}$	=	13$_{dec}$	=	15$_{oct}$	1	1	0	1
E$_{hex}$	=	14$_{dec}$	=	16$_{oct}$	1	1	1	0
F$_{hex}$	=	15$_{dec}$	=	17$_{oct}$	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8: 1 111 111 100 000 000

baza 10:

$\mathbf{0}_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
$\mathbf{1}_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
$\mathbf{2}_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
$\mathbf{3}_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
$\mathbf{4}_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
$\mathbf{5}_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
$\mathbf{6}_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
$\mathbf{7}_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
$\mathbf{8}_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
$\mathbf{9}_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
$\mathbf{A}_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
$\mathbf{B}_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
$\mathbf{C}_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
$\mathbf{D}_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
$\mathbf{E}_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
$\mathbf{F}_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8: 1 111 111 100 000 000 = 177400

baza 10:

0$_{hex}$	=	0$_{dec}$	=	0$_{oct}$	0	0	0	0
1$_{hex}$	=	1$_{dec}$	=	1$_{oct}$	0	0	0	1
2$_{hex}$	=	2$_{dec}$	=	2$_{oct}$	0	0	1	0
3$_{hex}$	=	3$_{dec}$	=	3$_{oct}$	0	0	1	1
4$_{hex}$	=	4$_{dec}$	=	4$_{oct}$	0	1	0	0
5$_{hex}$	=	5$_{dec}$	=	5$_{oct}$	0	1	0	1
6$_{hex}$	=	6$_{dec}$	=	6$_{oct}$	0	1	1	0
7$_{hex}$	=	7$_{dec}$	=	7$_{oct}$	0	1	1	1
8$_{hex}$	=	8$_{dec}$	=	10$_{oct}$	1	0	0	0
9$_{hex}$	=	9$_{dec}$	=	11$_{oct}$	1	0	0	1
A$_{hex}$	=	10$_{dec}$	=	12$_{oct}$	1	0	1	0
B$_{hex}$	=	11$_{dec}$	=	13$_{oct}$	1	0	1	1
C$_{hex}$	=	12$_{dec}$	=	14$_{oct}$	1	1	0	0
D$_{hex}$	=	13$_{dec}$	=	15$_{oct}$	1	1	0	1
E$_{hex}$	=	14$_{dec}$	=	16$_{oct}$	1	1	1	0
F$_{hex}$	=	15$_{dec}$	=	17$_{oct}$	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

CONVERSII, EX 2

1111 1111 0000 0000				

binar: 1111 1111 0000 0000

hexa: 0xFF00

baza 4: 11 11 11 11 00 00 00 00 = 33330000

baza 8: 1 111 111 100 000 000 = 177400

baza 10: -256

0$_{hex}$	=	0_{dec}	=	0_{oct}	0	0	0	0
1$_{hex}$	=	1_{dec}	=	1_{oct}	0	0	0	1
2$_{hex}$	=	2_{dec}	=	2_{oct}	0	0	1	0
3$_{hex}$	=	3_{dec}	=	3_{oct}	0	0	1	1
4$_{hex}$	=	4_{dec}	=	4_{oct}	0	1	0	0
5$_{hex}$	=	5_{dec}	=	5_{oct}	0	1	0	1
6$_{hex}$	=	6_{dec}	=	6_{oct}	0	1	1	0
7$_{hex}$	=	7_{dec}	=	7_{oct}	0	1	1	1
8$_{hex}$	=	8_{dec}	=	10_{oct}	1	0	0	0
9$_{hex}$	=	9_{dec}	=	11_{oct}	1	0	0	1
A$_{hex}$	=	10_{dec}	=	12_{oct}	1	0	1	0
B$_{hex}$	=	11_{dec}	=	13_{oct}	1	0	1	1
C$_{hex}$	=	12_{dec}	=	14_{oct}	1	1	0	0
D$_{hex}$	=	13_{dec}	=	15_{oct}	1	1	0	1
E$_{hex}$	=	14_{dec}	=	16_{oct}	1	1	1	0
F$_{hex}$	=	15_{dec}	=	17_{oct}	1	1	1	1

https://en.wikipedia.org/wiki/Binary_number

OPERAȚII BINARE, EX 3

$$\begin{array}{r|l} 0101\ 1100\ 1111\ 0011 & \\ 1111\ 1111\ 0000\ 0000 & + \\ \hline & \end{array} \qquad \begin{array}{r|l} 1111\ 1111\ 1111\ 1111 & \\ 0000\ 0000\ 0000\ 0001 & + \\ \hline & \end{array}$$

- **care sunt operanzii (zecimal)?**

.

OPERAȚII BINARE, EX 3

$$\begin{array}{r|l} 0101\ 1100\ 1111\ 0011 & \\ 1111\ 1111\ 0000\ 0000 & + \\ \hline & \end{array} \qquad \begin{array}{r|l} 1111\ 1111\ 1111\ 1111 & \\ 0000\ 0000\ 0000\ 0001 & + \\ \hline & \end{array}$$

- **care sunt operanzii (zecimal)?**
  - stânga: 23795 și -256
  - dreapta: -1 și +1

OPERAȚII BINARE, EX 3

$$\begin{array}{r|l} 1111\ 1111\ 1111\ 1111 & \\ 1000\ 0000\ 0000\ 0000 & + \\ \hline & \end{array} \qquad \begin{array}{r|l} 1000\ 0000\ 0000\ 0000 & \\ 0000\ 0000\ 0000\ 0001 & + \\ \hline & \end{array}$$

- **care sunt operanzii (zecimal)?**

OPERAȚII BINARE, EX 3

$$\begin{array}{r|l} 1111\ 1111\ 1111\ 1111 & \\ 1000\ 0000\ 0000\ 0000 & + \\ \hline & \end{array} \qquad \begin{array}{r|l} 1000\ 0000\ 0000\ 0000 & \\ 0000\ 0000\ 0000\ 0001 & + \\ \hline & \end{array}$$

- **care sunt operanzii (zecimal)?**
  - stânga: -1 și -32 768
  - dreapta: -32 768 și +1

.

OPERAȚII BINARE, EX 4

$$\begin{array}{r|l} 0101\ 1100\ 1111\ 0011 & \\ 0101\ 1100\ 1111\ 0011 & \text{AND} \\ \hline & \end{array}$$

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

$$\begin{array}{r|l} 1101\ 1100\ 1111\ 0011 & \\ 1101\ 1100\ 1111\ 0011 & \text{XOR} \\ \hline & \end{array}$$

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

OPERAȚII BINARE, EX 4

$$\begin{array}{r|l} 0000\ 0000\ 1111\ 1111 & \\ 0000\ 0001\ 0000\ 0000 & \text{AND} \\ \hline & \end{array}$$

$$\begin{array}{r|l} 1100\ 0110\ 1001\ 1110 & \\ 1001\ 1111\ 0110\ 1100 & \text{XOR} \\ 1100\ 0110\ 1001\ 1110 & \text{XOR} \\ \hline & \end{array}$$

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

ÎNTREBĂRI SCURTE, EX 5

a) $2^N - 1$

b) $2^{N-1} - 1$ și -2^{N-1}

c) aproximativ $\log_2 x$, exact sunt $\mathrm{ceil}(\log_2 (x+1))$

d) 4k

e) ceil (k / 4)

f) ceil ($k \log_2 10$)

ceil() rotunjește mereu în sus

BINARY FIXED-POINT, EX 6

...	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	...

- **½ = 0.5**
- **¼ = 0.25**
- **1/8 = 0.125**
- **1/16 = 0.0625**
- **...**

- **Calculați reprezentările pentru**

(a) 101.101;
(b) 111.001;
(c) 1110.00111;

(a) 3.75;
(b) 12.3125;
(c) 3.078125;

BINARY FIXED-POINT, EX 6

...	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	...

- **½ = 0.5**
- **¼ = 0.25**
- **1/8 = 0.125**
- **1/16 = 0.0625**
- **...**

- **Calculați reprezentările pentru**

(a) 101.101; 5.625
(b) 111.001;
(c) 1110.00111;

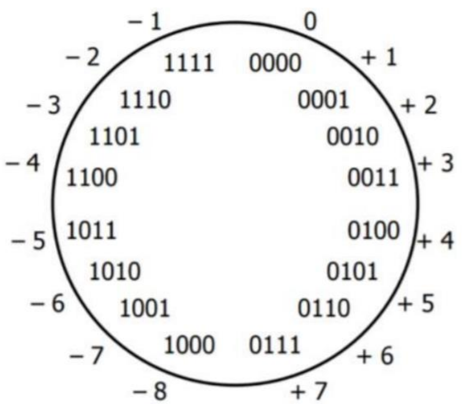
(a) 3.75; 11.11
(b) 12.3125;
(c) 3.078125;

COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	1	1	1	1	0	0	0	1
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**
  - pornim de la faptul că folosim aritmetică modulo
  - fixăm și suntem pe 8 biți

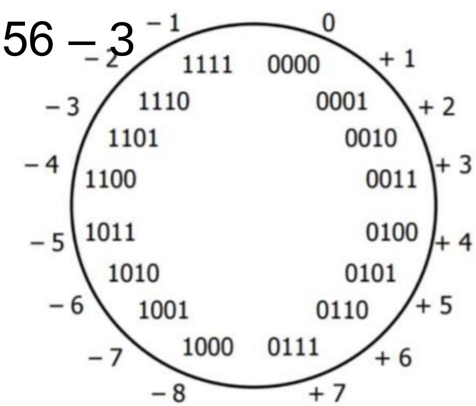


COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**
  - pornim de la faptul că folosim aritmetică modulo
  - fixăm și suntem pe 8 biți
  - deci, să scădem 3 e echivalent cu a aduna 256 – 3
  - -3 ≡ 256 – 3 =



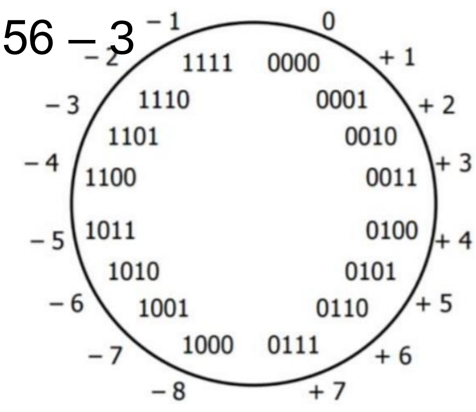
.

COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**
    - pornim de la faptul că folosim aritmetică modulo
    - fixăm și suntem pe 8 biți
    - deci, să scădem 3 e echivalent cu a aduna 256 – 3
    - -3 ≡ 256 – 3 = 1 0000 0000 – 0000 0101



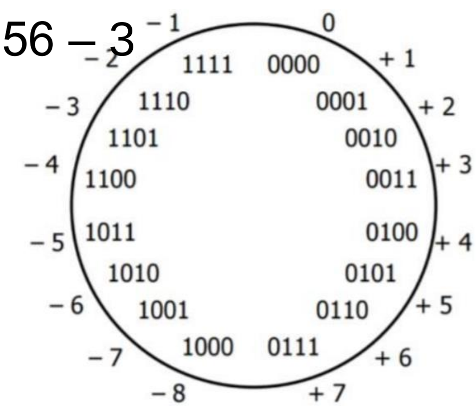
COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	1	1	1	1	0	0	0	1
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**
    - pornim de la faptul că folosim aritmetică modulo
    - fixăm și suntem pe 8 biți
    - deci, să scădem 3 e echivalent cu a aduna 256 – 3
    - -3 ≡ 256 – 3 = 1 0000 0000 – 0000 0101

      = 1 + 1111 1111 – 0000 0101



.

COMPLEMENT FAȚĂ DE DOI, EX 7

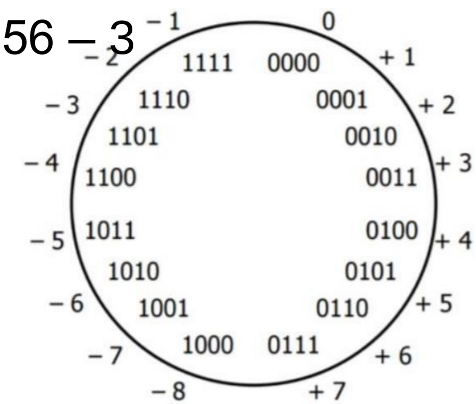
bit b_i:	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**
  - pornim de la faptul că folosim aritmetică modulo
  - fixăm și suntem pe 8 biți
  - deci, să scădem 3 e echivalent cu a aduna 256 – 3
  - -3 ≡ 256 – 3 = 1 0000 0000 – 0000 0101

    = 1 + 1111 1111 – 0000 0101

    = 1 + (3 cu biții inversați)



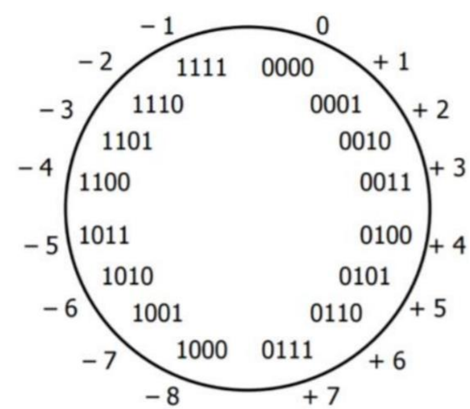
COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	**1**	**1**	**1**	**1**	**0**	**0**	**0**	**1**
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**

$$-\left(-2^N + \sum_{i=0}^{N-1} b_i 2^i\right) =$$



COMPLEMENT FAȚĂ DE DOI, EX 7

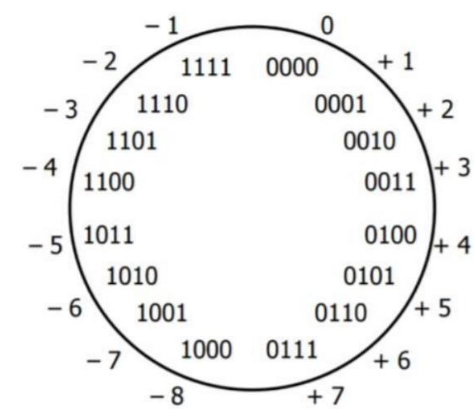
bit b_i:	1	1	1	1	0	0	0	1
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**

$$-\left(-2^N + \sum_{i=0}^{N-1} b_i 2^i\right) =$$

$$2^{N+1} = \sum_{i=0}^{N} 2^i + 1$$



.

COMPLEMENT FAȚĂ DE DOI, EX 7

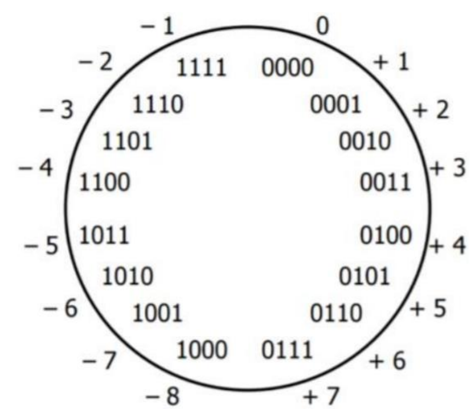
bit b_i:	1	1	1	1	0	0	0	1
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**

$$-\left(-2^N + \sum_{i=0}^{N-1} b_i 2^i\right) = 2^N - \sum_{i=0}^{N-1} b_i 2^i$$

$$2^{N+1} = \sum_{i=0}^{N} 2^i + 1$$



COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	1	1	1	1	0	0	0	1
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

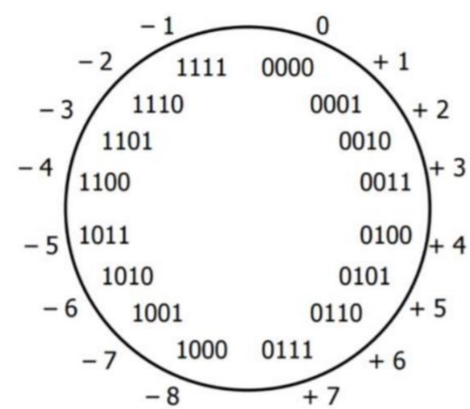
- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**

$$-\left(-2^N + \sum_{i=0}^{N-1} b_i 2^i\right) = 2^N - \sum_{i=0}^{N-1} b_i 2^i$$

$$= \sum_{i=0}^{N-1} 2^i + 1 - \sum_{i=0}^{N-1} b_i 2^i$$

$$2^{N+1} = \sum_{i=0}^{N} 2^i + 1$$



COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	1	1	1	1	0	0	0	1
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

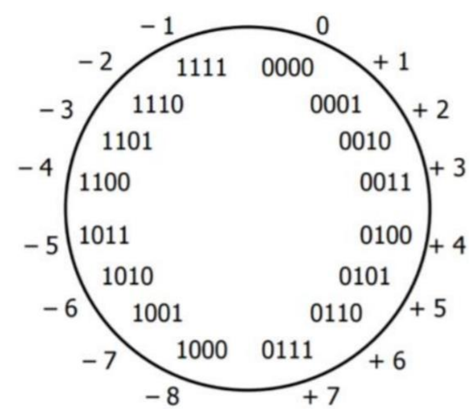
- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**

$$-\left(-2^N + \sum_{i=0}^{N-1} b_i 2^i\right) = 2^N - \sum_{i=0}^{N-1} b_i 2^i$$

$$= \sum_{i=0}^{N-1} 2^i + 1 - \sum_{i=0}^{N-1} b_i 2^i$$

$$= \sum_{i=0}^{N-1} (1 - b_i) 2^i + 1$$

$$2^{N+1} = \sum_{i=0}^{N} 2^i + 1$$



COMPLEMENT FAȚĂ DE DOI, EX 7

bit b_i:	1	1	1	1	0	0	0	1
2^i:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- $x = -b_{N-1}2^{N-1} + \sum_{i=0}^{N-2} b_i 2^i$

- **ca să reprezentăm un număr negativ, luăm valoarea pozitivă a numărului, îi inversăm biții și adunăm unu**
- **de ce funcționează această procedură?**

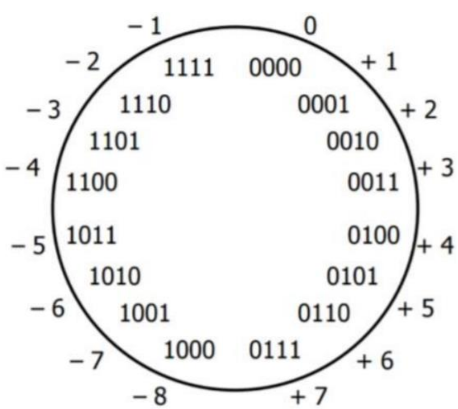
$$-\left(-2^N + \sum_{i=0}^{N-1} b_i 2^i\right) = 2^N - \sum_{i=0}^{N-1} b_i 2^i$$

$$2^{N+1} = \sum_{i=0}^{N} 2^i + 1$$

$$= \sum_{i=0}^{N-1} 2^i + 1 - \sum_{i=0}^{N-1} b_i 2^i$$

$$= \sum_{i=0}^{N-1} (1-b_i) 2^i + 1$$

$$= (\text{inversam bitii}) + 1$$



LOGARITM ÎNTREG, EX 9

bit b_i:	0	1	1	1	0	0	0	1
2^i:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- **arătați că** $\lfloor \log_2 x \rfloor = i_{\max}$
- **pornim de la reprezentarea binară și aplicăm logaritmul**

$$x = \sum_{i=0}^{N-1} b_i 2^i$$

$$\log_2 x = \log_2 \left(\sum_{i=0}^{N-1} b_i 2^i \right)$$

LOGARITM ÎNTREG, EX 9

bit b_i:	0	1	1	1	0	0	0	1
2^i:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- **arătați că** $\lfloor \log_2 x \rfloor = i_{\max}$
- **pornim de la reprezentarea binară și aplicăm logaritmul**

$$x = \sum_{i=0}^{N-1} b_i 2^i$$

$$\log_2 x = \log_2 \left(\sum_{i=0}^{N-1} b_i 2^i \right)$$

$$= \log_2 \left(2^{i_{\max}} \left(\sum_{i=0}^{N-1} b_i \frac{2^i}{2^{i_{\max}}} \right) \right)$$

LOGARITM ÎNTREG, EX 9

bit b_i:	0	1	1	1	0	0	0	1
2^i:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- **arătați că** $\lfloor \log_2 x \rfloor = i_{\max}$
- **pornim de la reprezentarea binară și aplicăm logaritmul**

$$x = \sum_{i=0}^{N-1} b_i 2^i$$

$$\begin{aligned}
\log_2 x &= \log_2 \left(\sum_{i=0}^{N-1} b_i 2^i \right) \\
&= \log_2 \left(2^{i_{\max}} \left(\sum_{i=0}^{N-1} b_i \frac{2^i}{2^{i_{\max}}} \right) \right) \\
&= \log_2 2^{i_{\max}} + \log_2 \left(\left(\sum_{i=0}^{N-1} b_i \frac{2^i}{2^{i_{\max}}} \right) \right)
\end{aligned}$$

.

LOGARITM ÎNTREG, EX 9

bit b_i:	0	1	1	1	0	0	0	1
2^i:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

- **arătați că** $\lfloor \log_2 x \rfloor = i_{\max}$
- **pornim de la reprezentarea binară și aplicăm logaritmul**

$$x = \sum_{i=0}^{N-1} b_i 2^i$$

$$\begin{aligned}
\log_2 x &= \log_2 \left(\sum_{i=0}^{N-1} b_i 2^i \right) \\
&= \log_2 \left(2^{i_{\max}} \left(\sum_{i=0}^{N-1} b_i \frac{2^i}{2^{i_{\max}}} \right) \right) \\
&= \log_2 2^{i_{\max}} + \log_2 \left(\left(\sum_{i=0}^{N-1} b_i \frac{2^i}{2^{i_{\max}}} \right) \right) \\
&= i_{\max} + C, \ C < 1
\end{aligned}$$

ARHITECTURA SISTEMELOR DE CALCUL SEMINAR 0x01

NOTIȚE SUPORT SEMINAR

Cristian Rusu

PACHET, EX 1

- **care este cantitatea de informație din pachet când scoatem cărțile pentru prima dată?**

PACHET, EX 1

- **care este cantitatea de informație din pachet când scoatem cărțile pentru prima dată?**

- **este 0, cărțile sunt ordonate crescător**

- **amestecăm aleator cărțile, câtă informație avem acum?**
    - în câte feluri putem combina cele 52 de cărți?

PACHET, EX 1

- **care este cantitatea de informație din pachet când scoatem cărțile pentru prima dată?**

- **este 0, cărțile sunt ordonate crescător**

- **amestecăm aleator cărțile, câtă informație avem acum?**
  - în câte feluri putem combina cele 52 de cărți?
  - 52!
  - deci informația este $\log_2(52!)$
    - cum calculăm valoarea asta?

PACHET, EX 1

- **care este cantitatea de informație din pachet când scoatem cărțile pentru prima dată?**

- **este 0, cărțile sunt ordonate crescător**

- **amestecăm aleator cărțile, câtă informație avem acum?**
  - în câte feluri putem combina cele 52 de cărți?
  - 52!
  - deci informația este $\log_2(52!)$
    - cum calculăm valoarea asta?
    - $\log_2(a \times b) = \log_2(a) + \log_2(b)$
    - $\log_2(52!) = 225.6$ biți
    - cu aproximarea lui Stirling:

PACHET, EX 1

- **care este cantitatea de informație din pachet când scoatem cărțile pentru prima dată?**

- **este 0, cărțile sunt ordonate crescător**

- **amestecăm aleator cărțile, câtă informație avem acum?**
  - în câte feluri putem combina cele 52 de cărți?
  - 52!
  - deci informația este $\log_2(52!)$
    - cum calculăm valoarea asta?
    - $\log_2(a \times b) = \log_2(a) + \log_2(b)$
    - $\log_2(52!) = 225.6$ biți
    - cu aproximarea lui Stirling: $\log_2(52!) \approx 52\log_2(52) - 52\log_2 e = 221.4$ biți
  - algoritmic, cum amestecăm cărțile (eficient)?
    - aveți la dispoziție o funcție care returnează o valoare aleatoare în intervalul [0,1]

PACHET, EX 1

- **care este cantitatea de informație din pachet când scoatem cărțile pentru prima dată?**

- **este 0, cărțile sunt ordonate crescător**

- **amestecăm aleator cărțile, câtă informație avem acum?**
  - algoritmic, cum amestecăm cărțile (eficient)?
    - aveți la dispoziție o funcție care returnează o valoare aleatoare în interalul [0,1]
      - considerăm cărțile sortate crescător
      - calculăm i = round(52*rand())
      - selectăm din pachet cartea i
      - swap cartea i cu cartea 52
      - calculăm i = round(51*rand())
      - selectăm din pachet cartea i
      - swap cartea i cu cartea 51
      - ...
  - verificați algoritmul “Fisher–Yates shuffle”

URNA, EX 2

$$H(X) = E(I(X)) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^{N} -p_i \log_2 p_i$$

- **în urnă: 5 bile roșii, 3 bile albastre**
- **observăm la extragere o bilă albastră, câtă informație am primit?**

URNA, EX 2

$$H(X) = E(I(X)) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^{N} -p_i \log_2 p_i$$

- **în urnă: 5 bile roșii, 3 bile albastre**
- **observăm la extragere o bilă albastră, câtă informație am primit?**

$$I(\text{bila albastra}) = \log_2\left(\frac{1}{\frac{3}{8}}\right) = \log_2\left(\frac{8}{3}\right) = 1.42 \text{ biti}$$

- **cât era entropia urnei inainte de extragere?**

URNA, EX 2

$$H(X) = E(I(X)) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^{N} -p_i \log_2 p_i$$

- **în urnă: 5 bile roșii, 3 bile albastre**
- **observăm la extragere o bilă albastră, câtă informație am primit?**

$$I(\text{bila albastra}) = \log_2\left(\frac{1}{\frac{3}{8}}\right) = \log_2\left(\frac{8}{3}\right) = 1.42 \text{ biti}$$

- **cât era entropia urnei inainte de extragere?**

$$H(\text{urna}) = \frac{5}{8}\log_2\left(\frac{8}{5}\right) + \frac{3}{8}\log_2\left(\frac{8}{3}\right) = 0.95 \text{ biti}$$

- **cât este entropia urnei după extragere?**

.

URNA, EX 2

$$H(X) = E(I(X)) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i} = \sum_{i=1}^{N} -p_i \log_2 p_i$$

- **în urnă: 5 bile roșii, 3 bile albastre**
- **observăm la extragere o bilă albastră, câtă informație am primit?**

$$I(\text{bila albastra}) = \log_2\left(\frac{1}{\frac{3}{8}}\right) = \log_2\left(\frac{8}{3}\right) = 1.42 \text{ biti}$$

- **cât era entropia urnei inainte de extragere?**

$$H(\text{urna}) = \frac{5}{8}\log_2\left(\frac{8}{5}\right) + \frac{3}{8}\log_2\left(\frac{8}{3}\right) = 0.95 \text{ biti}$$

- **cât este entropia urnei după extragere?**

$$H(\text{urna dupa extragere}) = \frac{5}{7}\log_2\left(\frac{7}{5}\right) + \frac{2}{7}\log_2\left(\frac{7}{2}\right) = 0.86 \text{ biti}$$

- **întrebare suplimentară: continuați calculul entropiei considerând că extragem în continuare (una câte una) toate bilele albastre**
- **întrebare suplimentară: repetați calculul entropiei considerând că extragem (una câte una) toate bilele roșii din urna originală**

CODURI PENTRU ERORI, EX 7

- tot ce e *D* este sunt biți de date
- tot ce e *P* sunt biți de paritate

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

- să presupun că D_{01} se schimbă (din 0 în 1, sau invers)
- câți biți din mesaj se schimbă?

CODURI PENTRU ERORI, EX 7

- **tot ce e *D* este sunt biți de date**
- **tot ce e *P* sunt biți de paritate**

D_{00}	$\boxed{D_{01}}$	D_{02}	$\boxed{P_{0\ell}}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	$\boxed{P_{c1}}$	P_{c2}	$\boxed{P_{c\ell}}$

- **să presupun că D_{01} se schimbă (din 0 în 1, sau invers)**
- **câți biți din mesaj se schimbă?**
  - 3 biți de paritate + bitul de date
- **care este distanța Hamming minimă?**

CODURI PENTRU ERORI, EX 7

- **tot ce e *D* este sunt biți de date**
- **tot ce e *P* sunt biți de paritate**

D_{00}	$\boxed{D_{01}}$	D_{02}	$\boxed{P_{0\ell}}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	$\boxed{P_{c1}}$	P_{c2}	$\boxed{P_{c\ell}}$

- **să presupun că D_{01} se schimbă (din 0 în 1, sau invers)**
- **câți biți din mesaj se schimbă?**
  - 3 biți de paritate + bitul de date
- **care este distanța Hamming minimă?**
  - 4 biți se schimbă, deci 4

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

- **pare corect, toți biții de paritate se potrivesc cu ce observăm**

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

- **nu este corect: bitul de paritate P_{c1} semnaleaza o eroare, dar biții de paritate de linie nu semnalează nimic: deci problema este chiar P_{c1} care a fost corupt**

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

.

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

- **nu este corect: bitul de paritate $P_{0\ell}$ este greșit, bitul de paritate P_{c0} este greșit => bitul de date D_{00} este greșit deci trebuie schimbat din 0 în 1; verificare: bitul total $P_{c\ell}$ este 0 deci ne trebuie în mesaj un număr impar de biți (deci cu schimbare este corect)**

.

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

- **nu este corect: toți biții de paritate de rând și coloane sunt OK, dar bitul de paritate total $P_{c\ell}$ nu e OK => deci bitul în eroare este chiar $P_{c\ell}$**

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

CODURI PENTRU ERORI, EX 7

- **este acest mesaj corect?**
  - dacă nu, de ce?

D_{00}	D_{01}	D_{02}	$P_{0\ell}$
D_{10}	D_{11}	D_{12}	$P_{1\ell}$
D_{20}	D_{21}	D_{22}	$P_{2\ell}$
P_{c0}	P_{c1}	P_{c2}	$P_{c\ell}$

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

- **nu este corect: toți biții de paritate $P_{1\ell}$, $P_{2\ell}$, P_{c1} și P_{c2} sunt greșiți, iar bitul de paritate $P_{c\ell}$ este corect => eroare este undeva pe linia/coloana 2/3: deci putem detecta erori de 2 biți dar nu le putem corecta**

RATA ENTROPIEI, EX 10

- lungimea medie a mesajului

- entropia în cazul nostru

- rata entropiei

RATA ENTROPIEI, EX 10

- **lungimea medie a mesajului**
  - $1xp + 2x(1-p) = 2 - p$

- **entropia în cazul nostru**

- **rata entropiei**

RATA ENTROPIEI, EX 10

- **lungimea medie a mesajului**
  - 1x*p* + 2x(1-*p*) = 2 – *p*

- **entropia în cazul nostru**

$$H = p\log_2\frac{1}{p} + (1-p)\log_2\frac{1}{1-p}$$

- **rata entropiei**

RATA ENTROPIEI, EX 10

- **lungimea medie a mesajului**
  - 1x*p* + 2x(1-*p*) = 2 – *p*

- **entropia în cazul nostru**

$$H = p\log_2\frac{1}{p} + (1-p)\log_2\frac{1}{1-p}$$

- **rata entropiei**

$$R = \frac{p\log_2\frac{1}{p} + (1-p)\log_2\frac{1}{1-p}}{2-p}$$

- **cum maximizăm?**

RATA ENTROPIEI, EX 10

- **lungimea medie a mesajului**
  - 1xp + 2x(1-p) = 2 – p

- **entropia în cazul nostru**

$$H = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$$

- **rata entropiei**

$$R = \frac{p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}}{2-p}$$

- **cum maximizăm?**
  - derivăm $R' = \frac{2 \log_2 \frac{1}{p} - \log_2 \frac{1}{1-p}}{(2-p)^2}$ și egalăm cu zero $\longrightarrow$

RATA ENTROPIEI, EX 10

- **lungimea medie a mesajului**
  - 1xp + 2x(1-p) = 2 – p

- **entropia în cazul nostru**

$$H = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}$$

- **rata entropiei**

$$R = \frac{p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}}{2-p}$$

- **cum maximizăm?**
  - derivăm $R' = \frac{2 \log_2 \frac{1}{p} - \log_2 \frac{1}{1-p}}{(2-p)^2}$ și egalăm cu zero $\longrightarrow$ $\log_2 \frac{1-p}{p^2} = 0$
  - p este soluția ecuației

RATA ENTROPIEI, EX 10

- **lungimea medie a mesajului**
  - 1x*p* + 2x(1-*p*) = 2 – *p*

- **entropia în cazul nostru**

$$H = p\log_2 \frac{1}{p} + (1-p)\log_2 \frac{1}{1-p}$$

- **rata entropiei**

$$R = \frac{p\log_2 \frac{1}{p} + (1-p)\log_2 \frac{1}{1-p}}{2-p}$$

- **cum maximizăm?**
  - derivăm $R' = \frac{2\log_2 \frac{1}{p} - \log_2 \frac{1}{1-p}}{(2-p)^2}$ și egalăm cu zero $\longrightarrow$ $\log_2 \frac{1-p}{p^2} = 0$
  - *p* este soluția ecuației $p^2 + p - 1 = 0$

RATA ENTROPIEI, EX 10

- **lungimea medie a mesajului**
  - 1x*p* + 2x(1-*p*) = 2 – *p*

- **entropia în cazul nostru**

$$H = p \log_2 \frac{1}{p} + (1-p)\log_2 \frac{1}{1-p}$$

- **rata entropiei**

$$R = \frac{p \log_2 \frac{1}{p} + (1-p)\log_2 \frac{1}{1-p}}{2-p}$$

- **cum maximizăm?**
  - derivăm $R' = \frac{2\log_2 \frac{1}{p} - \log_2 \frac{1}{1-p}}{(2-p)^2}$ și egalăm cu zero $\longrightarrow \log_2 \frac{1-p}{p^2} = 0$
  - *p* este soluția ecuației $p^2 + p - 1 = 0 \longrightarrow p = \frac{\sqrt{5}-1}{2}$

care este legătura cu raportul de aur?

MAXIMIZAREA ENTROPIEI, EX 11

- **considerăm că avem probabilități astfel încât $p_1 < p_2$**
- **avem că $p_1 + \varepsilon < p_2 - \varepsilon$ pentru ε destul de mic**
- **vrem să arătăm că**

$$H(\{p_1 + \epsilon, p_2 - \epsilon, p_3, \ldots, p_N\}) > H(\{p_1, p_2, p_3, \ldots, p_N\})$$

MAXIMIZAREA ENTROPIEI, EX 11

- **considerăm că avem probabilități astfel încât $p_1 < p_2$**
- **avem că $p_1 + \varepsilon < p_2 - \varepsilon$ pentru ε destul de mic**
- **vrem să arătăm că**

$$H(\{p_1 + \epsilon, p_2 - \epsilon, p_3, \ldots, p_N\}) > H(\{p_1, p_2, p_3, \ldots, p_N\})$$

- **trebuie să arătăm că**

$$H(\{p_1 + \epsilon, p_2 - \epsilon, p_3, \ldots, p_N\}) - H(\{p_1, p_2, p_3, \ldots, p_N\}) > 0$$

- **diferența dintre cele două entropii se scrie ca**

MAXIMIZAREA ENTROPIEI, EX 11

- **considerăm că avem probabilități astfel încât $p_1 < p_2$**
- **avem că $p_1 + \varepsilon < p_2 - \varepsilon$ pentru ε destul de mic**
- **vrem să arătăm că**

$$H(\{p_1 + \epsilon, p_2 - \epsilon, p_3, \ldots, p_N\}) > H(\{p_1, p_2, p_3, \ldots, p_N\})$$

- **trebuie să arătăm că**

$$H(\{p_1 + \epsilon, p_2 - \epsilon, p_3, \ldots, p_N\}) - H(\{p_1, p_2, p_3, \ldots, p_N\}) > 0$$

- **diferența dintre cele două entropii se scrie ca**

$$-p_1 \log_2\left(\frac{p_1 + \epsilon}{p_1}\right) - \epsilon \log_2(p_1 + \epsilon) - p_2 \log_2\left(\frac{p_2 - \epsilon}{p_2}\right) + \epsilon \log_2(p_2 - \epsilon) > 0$$

- **acum, vrem să folosim** $\log_2(1 + x) \approx x + O(x^2)$

MAXIMIZAREA ENTROPIEI, EX 11

- **diferența dintre cele două entropii se scrie ca**

$$-p_1 \log_2\left(\frac{p_1+\epsilon}{p_1}\right) - \epsilon \log_2(p_1+\epsilon) - p_2 \log_2\left(\frac{p_2-\epsilon}{p_2}\right) + \epsilon \log_2(p_2-\epsilon) > 0$$

- **acum, vrem să folosim** $\log_2(1+x) \approx x + O(x^2)$
- **pe partea stângă a inegalității rezultă:**

.

MAXIMIZAREA ENTROPIEI, EX 11

- **diferența dintre cele două entropii se scrie ca**

$$-p_1 \log_2\left(\frac{p_1+\epsilon}{p_1}\right) - \epsilon \log_2(p_1+\epsilon) - p_2 \log_2\left(\frac{p_2-\epsilon}{p_2}\right) + \epsilon \log_2(p_2-\epsilon) > 0$$

- **acum, vrem să folosim** $\log_2(1+x) \approx x + O(x^2)$
- **pe partea stângă a inegalității rezultă:**

$$-\epsilon - \epsilon \log_2 p_1 + \epsilon + \epsilon p_2 + O(\epsilon^2) = \epsilon \log_2 \frac{p_2}{p_1} + O(\epsilon^2)$$

- **este această expresie mereu pozitivă?**

.

MAXIMIZAREA ENTROPIEI, EX 11

- **diferența dintre cele două entropii se scrie ca**

$$-p_1 \log_2\left(\frac{p_1+\epsilon}{p_1}\right) - \epsilon \log_2(p_1+\epsilon) - p_2 \log_2\left(\frac{p_2-\epsilon}{p_2}\right) + \epsilon \log_2(p_2-\epsilon) > 0$$

- **acum, vrem să folosim** $\log_2(1+x) \approx x + O(x^2)$
- **pe partea stângă a inegalității rezultă:**

$$-\epsilon - \epsilon \log_2 p_1 + \epsilon + \epsilon p_2 + O(\epsilon^2) = \epsilon \log_2 \frac{p_2}{p_1} + O(\epsilon^2)$$

- **este această expresie mereu pozitivă?**
  - da, pentru că am presupus $p_1 < p_2$

CODARE PREFIX, EX 12

- exemple de coduri:

CODARE PREFIX, EX 12

- **exemple de coduri:**
  - 1
  - 01
  - 001
  - 0001
  - ...

- **indiferent de p_i, fiecare simbol are un bit în plus față de precedentul, dar simbolurile cele mai frecvente tot primesc coduri mai scurte decât cele mai puțin frecvente**
- **entropia**
- **lungimea medie a codării**

CODARE PREFIX, EX 12

- **exemple de coduri:**
  - 1
  - 01
  - 001
  - 0001
  - ...

- **indiferent de p_i, fiecare simbol are un bit în plus față de precedentul, dar simbolurile cele mai frecvente tot primesc coduri mai scurte decât cele mai puțin frecvente**
- **entropia** $H = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}$
- **lungimea medie a codării** $L = \sum_{i=1}^{N} i p_i$

- **când avem $H = L$?**

CODARE PREFIX, EX 12

- **exemple de coduri:**
  - 1
  - 01
  - 001
  - 0001
  - ...

- **indiferent de p_i, fiecare simbol are un bit în plus față de precedentul, dar simbolurile cele mai frecvente tot primesc coduri mai scurte decât cele mai puțin frecvente**
- **entropia** $H = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}$
- **lungimea medie a codării** $L = \sum_{i=1}^{N} i p_i$

- **când avem *H = L*?** $\sum_{i=1}^{N} p_i (i - \log_2 \frac{1}{p_i}) = 0 \longrightarrow i = \log_2 \frac{1}{p_i}$, deci $p_i = 2^{-i}$

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 1 \text{ pentru } N \to \infty$$

BALANȚA, EX 3

- **12 bile, una dintre ele are o greutate diferită față de celelalte**
- **găsiți bila diferită cu un număr minim de cântăriri**

- **întrebări inițiale:**
  - câte cântăriri avem nevoie cu soluția “simplă”?
    - cântărim două câte două bilele (6 experimente): un experiment va reduce opțiunile la două bile, apoi comparăm fiecare bilă cu una din cele care este cu sigurantă standard (2 experimente)
    - deci cu 8 cântăriri putem sigur găsi bila
  - câtă informație trebuie să descoperim?

BALANȚA, EX 3

- **12 bile, una dintre ele are o greutate diferită față de celelalte**
- **găsiți bila diferită cu un număr minim de cântăriri**

- **întrebări inițiale:**
  - câte cântăriri avem nevoie cu soluția “simplă”?
    - cântărim două câte două bilele (6 experimente): un experiment va reduce opțiunile la două bile, apoi comparăm fiecare bilă cu una din cele care es cu siguranță standard (2 experimente)
    - deci cu 8 cântăriri putem sigur găsi bila
  - câtă informație trebuie să descoperim?
    - care e bila diferită? $\log_2(12) = 3.58$
    - dacă este mai ușoară sau mai grea: 1 bit (dacă nu știm care situatie este

BALANȚA, EX 3

- **să analizăm sistematic**
  - avem 3 valori ce trebuie specificate când facem o cântarire
    - câte bile putem în stanga balanței
    - câte bile putem în dreapta balanței
    - câte bile rămân pe masă
  - toate posibilitățile care ne interesează:

BALANȚA, EX 3

- **să analizăm sistematic**
  - avem 3 valori ce trebuie specificate când facem o cântarire
    - câte bile putem în stanga balanței
    - câte bile putem în dreapta balanței
    - câte bile rămân pe masă

  - toate posibilitățile care ne interesează:
    - 6 cu 6, 0 pe masă
    - 5 cu 5, 2 pe masă
    - 4 cu 4, 4 pe masă
    - 3 cu 3, 6 pe masă
    - 2 cu 2, 8 pe masă
    - 1 cu 1, 10 pe masă

BALANȚA, EX 3

- **6 cu 6, 0 pe masă**
  - care sunt rezultatele posibile?
    - balanța cade spre stânga
    - balanța cade spre dreapta
    - balanța este echilibrată

  - care sunt probabilitățile pentru fiecare posibilitate?

BALANȚA, EX 3

- **6 cu 6, 0 pe masă**
  - care sunt rezultatele posibile?
    - balanța cade spre stânga (1/2)
    - balanța cade spre dreapta (1/2)
    - balanța este echilibrată (0)
  - care este entropia acestui alfabet?

$$H(6 \text{ cu } 6) = \frac{1}{2}\log_2(2) + \frac{1}{2}\log_2(2) = 1 \text{ bit}$$

.

BALANȚA, EX 3

- **5 cu 5, 2 pe masă**
  - care sunt rezultatele posibile?
    - balanța cade spre stânga (5/12)
    - balanța cade spre dreapta (5/12)
    - balanța este echilibrată (2/12 = 1/6)
  - care este entropia acestui alfabet?

$$H(5 \text{ cu } 5) = 2 \times \frac{5}{12}\log_2\left(\frac{12}{5}\right) + \frac{1}{6}\log_2(6) = 1.48 \text{ biti}$$

.

BALANȚA, EX 3

- **4 cu 4, 4 pe masă**
  - care sunt rezultatele posibile?
    - balanța cade spre stânga (1/3)
    - balanța cade spre dreapta (1/3)
    - balanța este echilibrată (1/3 = 4/12)
  - care este entropia acestui alfabet?

$$H(4 \text{ cu } 4) = 3 \times \frac{1}{3} \log_2(3) = 1.58 \text{ biti}$$

BALANȚA, EX 3

- **3 cu 3, 6 pe masă**
  - care sunt rezultatele posibile?
    - balanța cade spre stânga (1/4)
    - balanța cade spre dreapta (1/4)
    - balanța este echilibrată (1/2 = 6/12)
  - care este entropia acestui alfabet?

$$H(3 \text{ cu } 3) = 2 \times \frac{1}{4}\log_2(4) + \frac{1}{2}\log_2(2) = 1.5 \text{ biti}$$

BALANȚA, EX 3

- **Rezumat:**
    - 6 cu 6, 0 pe masă H = 1 bit
    - 5 cu 5, 2 pe masă H = 1.48 biți
    - 4 cu 4, 4 pe masă H = 1.58 biți
    - 3 cu 3, 6 pe masă H = 1.5 biți

    - ce alegem?

BALANȚA, EX 3

- **Rezumat:**
  - 6 cu 6, 0 pe masă H = 1 bit
  - 5 cu 5, 2 pe masă H = 1.48 biți
  - 4 cu 4, 4 pe masă H = 1.58 biți
  - 3 cu 3, 6 pe masă H = 1.5 biți

  - ce alegem?

BALANȚA, EX 3

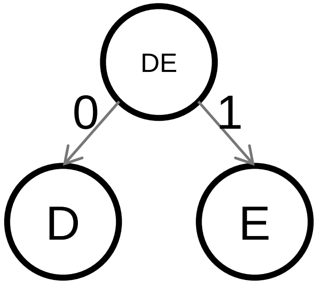
- **4 cu 4, 4 pe masă H = 1.58 biți**
  - facem experimentul, care sunt posibilitățile?
    - (caz 1) balanța cade la stânga
      - bilele pe partea stângă sunt mai grele
      - bilele pe partea dreaptă sunt mai ușoare
      - bilele de pe masa au aceeași greutate
    - (caz 2) balanța cade la dreapta
      - bilele pe partea dreaptă sunt mai grele
      - bilele pe partea stângă sunt mai ușoare
      - bilele de pe masa au aceeași greutate
    - (caz 3) balanța este echilibrată
      - bilele de pe masă conțin bila diferită

  - dacă este cazul 3, am redus numărul de bile de verificat la 4
  - dacă este cazul 1 sau 2, nu știm dacă bila este pe stânga sau pe dreapta dar știm sigur că nu e pe masa

BALANȚA, EX 3

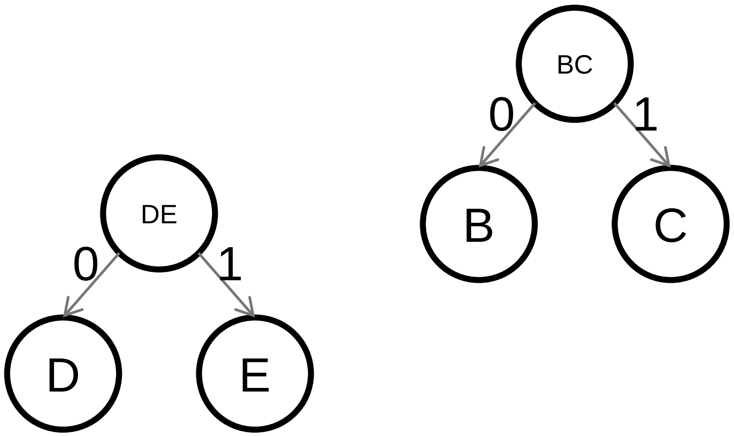
- **4 cu 4, 4 pe masă H = 1.58 biți**
  - (caz 3) balanța este echilibrată
    - bilele de pe masă conțin bila diferită
  - să numerotăm bilele:
    - pe partea dreaptă 1 2 3 4
    - pe partea stângă 5 6 7 8
    - pe masă 9 10 11 12
  - ce măsuratoare urmează?
    - 1 2 3 9 vs 4 5 10 11
      - dacă e echilibru, bila 12 este defectă (mai grea sau mai ușoară)
      - dacă balanța cade pe stânga
        - fie 9 e diferită și e grea
        - fie 10 sau 11 sunt diferite și una dintre ele e mai ușoară
        - măsuram 10 vs 11
          - dacă sunt egale, 9 e diferită și e mai grea sigur
          - dacă nu sunt egale, cea mai ușoară este cea diferită
      - dacă balanța cade pe dreapta ... procedura este similară
  - continuați voi ... cu celelalte cazuri

HUFFMAN, EX 6



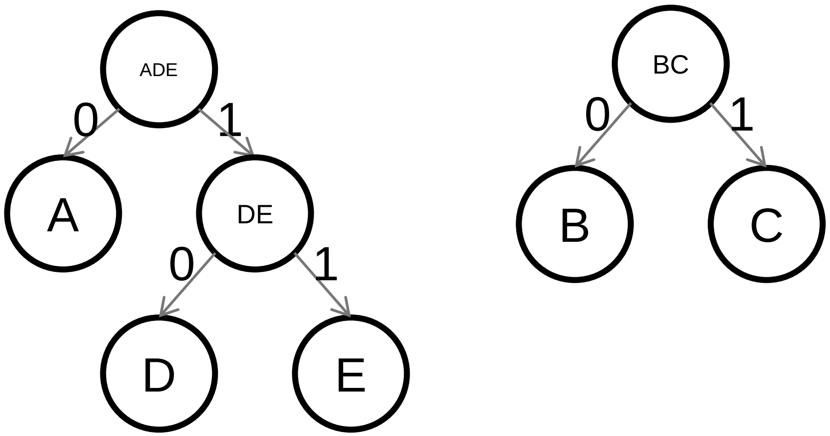
luăm două elemente cele mai improbabile (D și E, era OK C și E)
probabilitatea DE este 1/6 + 1/12 = 3/12 = 1/4

HUFFMAN, EX 6



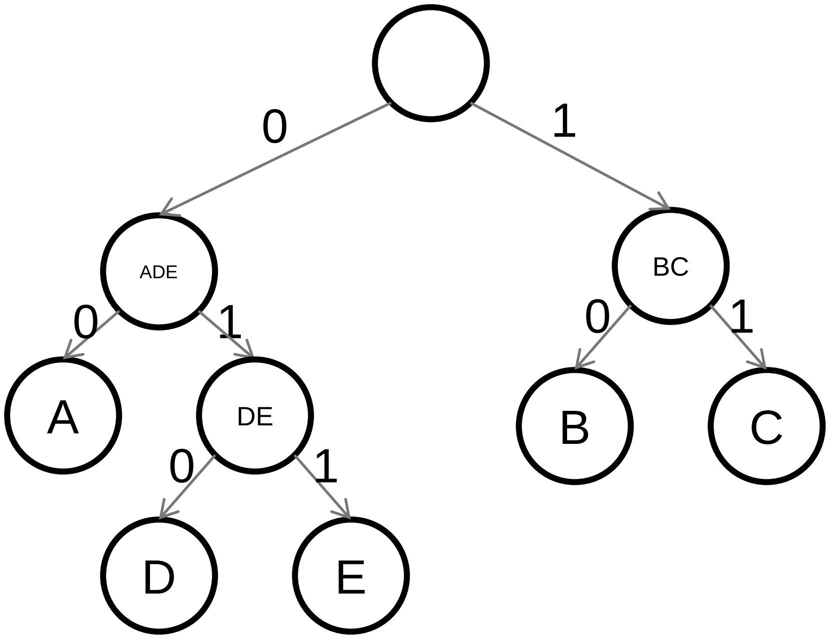
următorul element de adăugat acum este C cu probabilitate 1/6
dar 1/6 este mai puțin decât probabilitatea DE, deci C nu va fi singur
probabilitatea BC este ¼ + 1/6 = 10/24 = 5/12

HUFFMAN, EX 6



următorul element este A și are probabilitate 1/3
nu contează pe care arbore îl punem, fie împreună cu DE fie cu BC
(dacă îl puneți lângă BC codul lui A e diferit dar are aceeași dimensiune)

HUFFMAN, EX 6



arborele complet și codurile:

A = 00

B = 10

C = 11

D = 010

E = 011

ARHITECTURA SISTEMELOR DE CALCUL SEMINAR 0x02

NOTIȚE SUPORT SEMINAR

Cristian Rusu

TABELE DE ADEVĂR, EX 1

$X = A + BC$

A	B	C	BC	X

TABELE DE ADEVĂR, EX 1

$X = A + BC$

A	B	C	BC	X
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

TABELE DE ADEVĂR, EX 1

$X = A + BC$

A	B	C	BC	X
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	

.

TABELE DE ADEVĂR, EX 1

X = A + BC

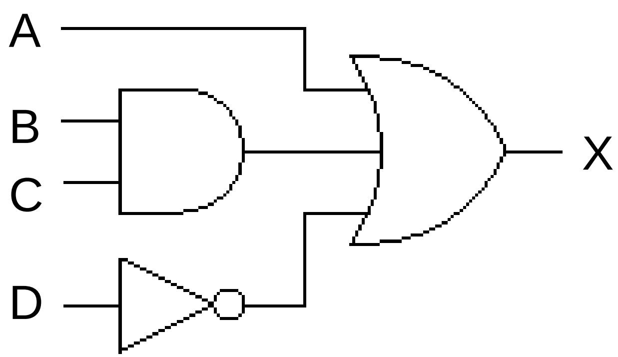
A	B	C	BC	X
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

DESENAȚI CIRCUITUL, EX 2

$X = A + BC + \overline{D}$

DESENAȚI CIRCUITUL, EX 2

$X = A + BC + \overline{D}$



DE MORGAN, EX 3

$$X = \overline{(\overline{A\bar{B}})(\bar{B} + C)}$$

$$=$$

$$=$$

$$=$$

$$=$$

$$=$$

DE MORGAN, EX 3

$$X = \overline{(\overline{A\bar{B}})(\bar{B} + C)}$$

$$= \overline{(\overline{A\bar{B}})} + \overline{(\bar{B} + C)}$$

$$=$$

$$=$$

$$=$$

$$=$$

.

DE MORGAN, EX 3

$$
\begin{aligned}
X &= \overline{(\overline{A\bar{B}})(\bar{B}+C)} \\
&= \overline{(\overline{A\bar{B}})} + \overline{(\bar{B}+C)} \\
&= A\bar{B} + \overline{(\bar{B}+C)} \\
&= \\
&= \\
&=
\end{aligned}
$$

DE MORGAN, EX 3

$$X = \overline{(\overline{A\bar{B}})(\bar{B} + C)}$$

$$= \overline{(\overline{A\bar{B}})} + \overline{(\bar{B} + C)}$$

$$= A\bar{B} + \overline{(\bar{B} + C)}$$

$$= A\bar{B} + (\bar{\bar{B}}\bar{C})$$

$$=$$

$$=$$

DE MORGAN, EX 3

$$X = \overline{(\overline{A\bar{B}})(\bar{B} + C)}$$

$$= \overline{(\overline{A\bar{B}})} + \overline{(\bar{B} + C)}$$

$$= A\bar{B} + \overline{(\bar{B} + C)}$$

$$= A\bar{B} + (\bar{\bar{B}}\bar{C})$$

$$= A\bar{B} + (B\bar{C})$$

$$=$$

DE MORGAN, EX 3

$$
\begin{aligned}
X &= \overline{(\overline{A\bar{B}})(\bar{B}+C)} \\
&= \overline{(\overline{A\bar{B}})} + \overline{(\bar{B}+C)} \\
&= A\bar{B} + \overline{(\bar{B}+C)} \\
&= A\bar{B} + (\bar{\bar{B}}\bar{C}) \\
&= A\bar{B} + (B\bar{C}) \\
&= A\bar{B} + B\bar{C}
\end{aligned}
$$

DE MORGAN, EX 3

$$X = \overline{(\bar{A} + C)(\overline{AB})}$$

$$=$$

$$=$$

$$=$$

$$=$$

$$=$$

.

DE MORGAN, EX 3

$$
\begin{aligned}
X &= \overline{(\bar{\mathrm{A}} + \mathrm{C})(\overline{\mathrm{AB}})} \\
&= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + \overline{(\overline{\mathrm{AB}})} \\
&= \\
&= \\
&= \\
&=
\end{aligned}
$$

DE MORGAN, EX 3

$$
\begin{aligned}
X &= \overline{(\bar{\mathrm{A}} + \mathrm{C})(\overline{\mathrm{AB}})} \\
&= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + \overline{(\overline{\mathrm{AB}})} \\
&= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + (\mathrm{AB}) \\
&= \\
&= \\
&=
\end{aligned}
$$

DE MORGAN, EX 3

$$X = \overline{(\bar{\mathrm{A}} + \mathrm{C})(\overline{\mathrm{AB}})}$$

$$= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + \overline{(\overline{\mathrm{AB}})}$$

$$= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + (\mathrm{AB})$$

$$= (\bar{\bar{\mathrm{A}}}\overline{\mathrm{C}}) + (\mathrm{AB})$$

$$=$$

$$=$$

DE MORGAN, EX 3

$$X = \overline{(\bar{\mathrm{A}} + \mathrm{C})(\overline{\mathrm{AB}})}$$

$$= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + \overline{(\overline{\mathrm{AB}})}$$

$$= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + (\mathrm{AB})$$

$$= (\overline{\bar{\mathrm{A}}}\overline{\mathrm{C}}) + (\mathrm{AB})$$

$$= (\mathrm{A}\overline{\mathrm{C}}) + (\mathrm{AB})$$

$$=$$

DE MORGAN, EX 3

$$
\begin{aligned}
X &= \overline{(\bar{\mathrm{A}} + \mathrm{C})(\overline{\mathrm{AB}})} \\
&= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + \overline{(\overline{\mathrm{AB}})} \\
&= \overline{(\bar{\mathrm{A}} + \mathrm{C})} + (\mathrm{AB}) \\
&= (\overline{\bar{\mathrm{A}}}\,\overline{\mathrm{C}}) + (\mathrm{AB}) \\
&= (\mathrm{A}\overline{\mathrm{C}}) + (\mathrm{AB}) \\
&= \mathrm{A}(\mathrm{B} + \overline{\mathrm{C}})
\end{aligned}
$$

DE MORGAN, EX 3

- !(!A+!B) = AB
- !(!A!B) = A+B
- !(A+B+C) = !A!B!C
- !(ABC) = !A+!B+!C
- !(A+B)!A!B = !A!B
- !(AB)(!A+!B) = !A+!B
- !(A+B)(!A+!B) = !A!B
- !A!B!(AB) = !A!B
- C+!(CB) = 1
- !(AB)(!A+B)(!B+!B) = !A!B

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

$$(A + C)A(D + \bar{D}) + AC + C \quad //\text{distribuim, invers}$$

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

$$(A + C)A(D + \bar{D}) + AC + C \quad //\text{distribuim, invers}$$

$$(A + C)A + AC + C \quad //\text{suma variabila si complement}$$

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

$$(A + C)A(D + \bar{D}) + AC + C \quad //\text{distribuim, invers}$$

$$(A + C)A + AC + C \quad //\text{suma variabila si complement}$$

$$A((A + C) + C) + C \quad //\text{distribuim, invers}$$

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

$$(A + C)A(D + \bar{D}) + AC + C \quad //\text{distribuim, invers}$$

$$(A + C)A + AC + C \quad //\text{suma variabila si complement}$$

$$A((A + C) + C) + C \quad //\text{distribuim, invers}$$

$$A(A + C) + C \quad //\text{asociem, idempotent}$$

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

$$(A + C)A(D + \bar{D}) + AC + C \quad //\text{distribuim, invers}$$

$$(A + C)A + AC + C \quad //\text{suma variabila si complement}$$

$$A((A + C) + C) + C \quad //\text{distribuim, invers}$$

$$A(A + C) + C \quad //\text{asociem, idempotent}$$

$$AA + AC + C \quad //\text{distribuim}$$

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

$$(A + C)A(D + \bar{D}) + AC + C \quad //\text{distribuim, invers}$$

$$(A + C)A + AC + C \quad //\text{suma variabila si complement}$$

$$A((A + C) + C) + C \quad //\text{distribuim, invers}$$

$$A(A + C) + C \quad //\text{asociem, idempotent}$$

$$AA + AC + C \quad //\text{distribuim}$$

$$A + (A + 1)C \quad //\text{idempotent, identitate, factor}$$

SIMPLIFICĂRI, EX 4

$$(A + C)(AD + A\bar{D}) + AC + C$$

$$(A + C)A(D + \bar{D}) + AC + C \quad //\text{distribuim, invers}$$

$$(A + C)A + AC + C \quad //\text{suma variabila si complement}$$

$$A((A + C) + C) + C \quad //\text{distribuim, invers}$$

$$A(A + C) + C \quad //\text{asociem, idempotent}$$

$$AA + AC + C \quad //\text{distribuim}$$

$$A + (A + 1)C \quad //\text{idempotent, identitate, factor}$$

$$A + C \quad //\text{identitate de doua ori}$$

SIMPLIFICĂRI, EX 4

$$\bar{A}(A + B) + (B + AA)(A + \bar{B})$$

.

SIMPLIFICĂRI, EX 4

$$\bar{A}(A + B) + (B + AA)(A + \bar{B})$$

$$\bar{A}(A + B) + (B + A)(A + \bar{B}) \text{ //AA este A}$$

SIMPLIFICĂRI, EX 4

$$\bar{\mathrm{A}}(\mathrm{A} + \mathrm{B}) + (\mathrm{B} + \mathrm{AA})(\mathrm{A} + \bar{\mathrm{B}})$$

$$\bar{\mathrm{A}}(\mathrm{A} + \mathrm{B}) + (\mathrm{B} + \mathrm{A})(\mathrm{A} + \bar{\mathrm{B}}) \quad //\mathrm{AA} \text{ este } \mathrm{A}$$

$$(\mathrm{A} + \mathrm{B})(\bar{\mathrm{A}} + \mathrm{A} + \bar{\mathrm{B}}) \quad // \text{ factor } \mathrm{A} + \mathrm{B}$$

SIMPLIFICĂRI, EX 4

$$\bar{A}(A + B) + (B + AA)(A + \bar{B})$$

$$\bar{A}(A + B) + (B + A)(A + \bar{B}) \ //AA \text{ este } A$$

$$(A + B)(\bar{A} + A + \bar{B}) \ // \text{ factor } A + B$$

$$(A + B)(1 + \bar{B}) \ // \text{ variabila sau complement}$$

.

SIMPLIFICĂRI, EX 4

$\bar{A}(A + B) + (B + AA)(A + \bar{B})$

$\bar{A}(A + B) + (B + A)(A + \bar{B})$ //AA este A

$(A + B)(\bar{A} + A + \bar{B})$ // factor A + B

$(A + B)(1 + \bar{B})$ // variabila sau complement

$A + B$ // 1 sau orice este 1

.

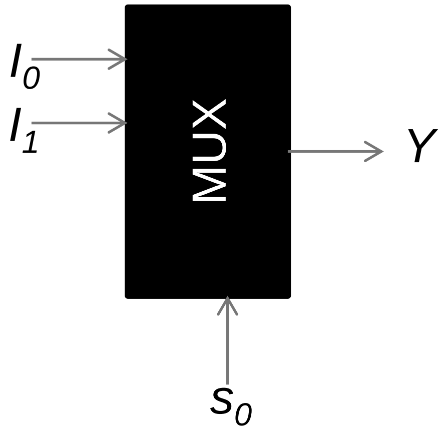
SIMPLIFICĂRI, EX 4

a) A+0 = A
b) !Ax0 = 0
c) A+!A = 1
d) A+A = A
e) A+AB = A
f) A+!AB = A+B
g) A(!A+B) = AB
h) AB+!AB = B
i) (!A!B+!AB) = !A
j) A(A+B+C+...) = A
k) subpuncte
   a) A+B
   b) 1
   c) 1
l) A+A!A = A
m) AB+A!B = A
n) !A+B!A = !A
o) (D+!A+B+!C)B = B
p) (A+!B)(A+B) = A
q) C(C+CD) = C
r) A(A+AB) = A
s) !(!A+!A) = A
t) !(A+!A) = 0
u) D+(D!CBA) = D
v) !D!(DBCA) = !D
w) AC+!AB+BC = AC+!AB
x) (A+C)(!A+B)(B+C) = AB+!AC
y) !A+!B+AB!C = !A+!B+!C

$(A+B)^2+(A+B)^3+A+3!A+A^3 = 1$

MUX, EX 5 A

- **MUX, două intrări, un semnal s de selecție și o ieșire**

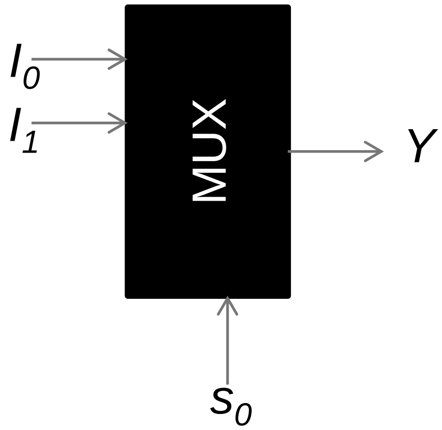


I_0	I_1	s_0	Y
*	*	0	I_0
*	*	1	I_1

- **care este relația ieșire-intrare?**
  - Y = ?

MUX, EX 5 A

- **MUX, două intrări, un semnal s de selecție și o ieșire**

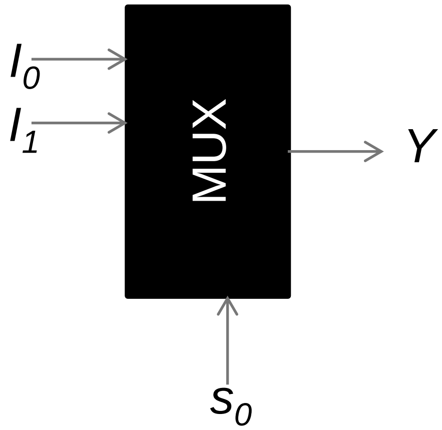


I_0	I_1	s_0	Y
*	*	0	I_0
*	*	1	I_1

- **care este relația ieșire-intrare?**
  - $Y = I_0\bar{s}_0 + I_1 s_0$

MUX, EX 5 B

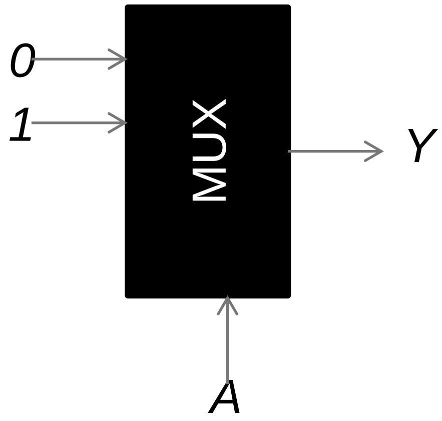
- **MUX, două intrări, un semnal s de selecție și o ieșire**



- **un MUX este un circuit universal, adică poate implementa porți NOT, OR și AND**
  - implementați o poartă NOT cu un MUX

MUX, EX 5 B

- **MUX, două intrări, un semnal s de selecție și o ieșire**

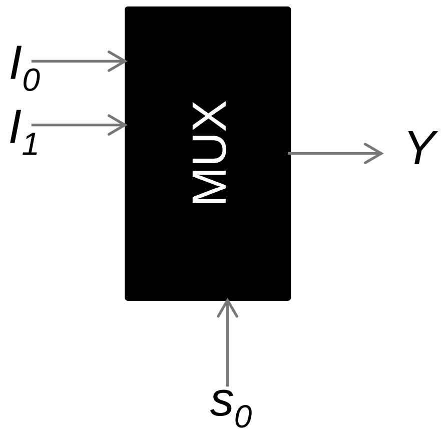


I_0	I_1	$s_0(A)$	Y
0	1	0	I_1
0	1	1	I_0

- **un MUX este un circuit universal, adică poate implementa porți NOT, OR și AND**
  - implementați o poartă NOT cu un MUX: Y = NOT A

MUX, EX 5 B

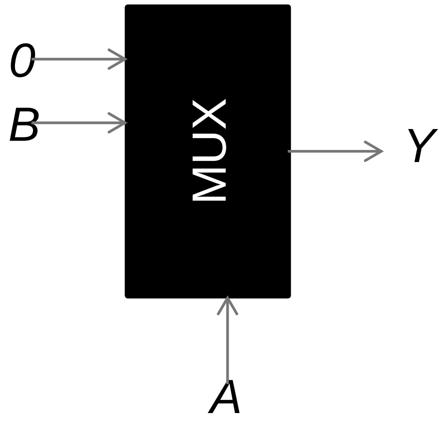
- **MUX, două intrări, un semnal s de selecție și o ieșire**



- **un MUX este un circuit universal, adică poate implementa porți NOT, OR și AND**
  - implementați o poartă AND cu un MUX

MUX, EX 5 B

- **MUX, două intrări, un semnal s de selecție și o ieșire**

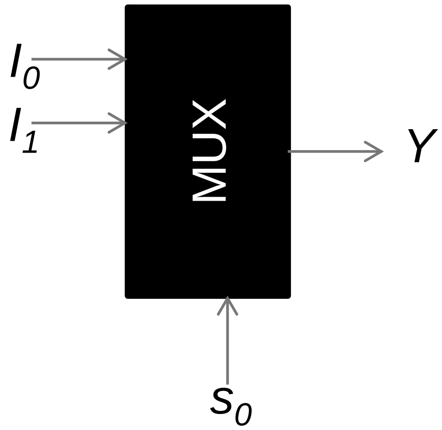


I_0	I_1 (B)	s_0 (A)	Y
0	B	0	I_0 (0)
0	B	1	I_1 (B)

- **un MUX este un circuit universal, adică poate implementa porți NOT, OR și AND**
  - implementați o poartă AND cu un MUX: Y = A AND B

MUX, EX 5 B

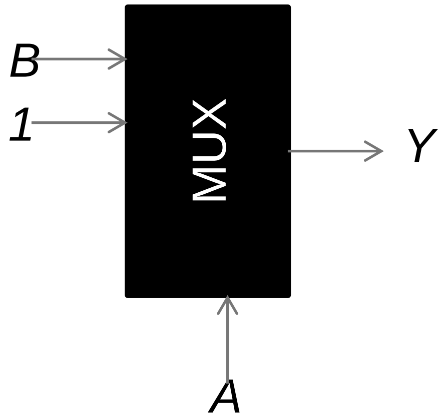
- **MUX, două intrări, un semnal s de selecție și o ieșire**



- **un MUX este un circuit universal, adică poate implementa porți NOT, OR și AND**
    - implementați o poartă OR cu un MUX

MUX, EX 5 B

- **MUX, două intrări, un semnal s de selecție și o ieșire**

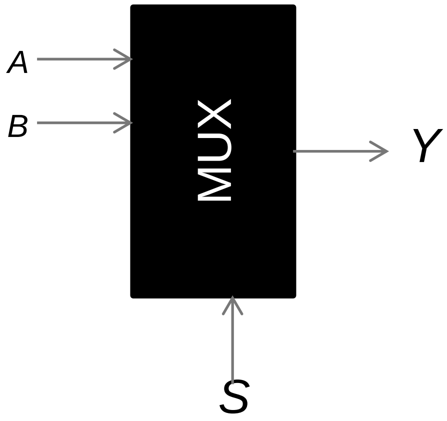


I_0 (B)	I_1	s_0 (A)	Y
B	1	0	I_0 (B)
B	1	1	I_1 (1)

- **un MUX este un circuit universal, adică poate implementa porți NOT, OR și AND**
  - implementați o poartă OR cu un MUX: Y = A OR B

MUX, EX 5 C

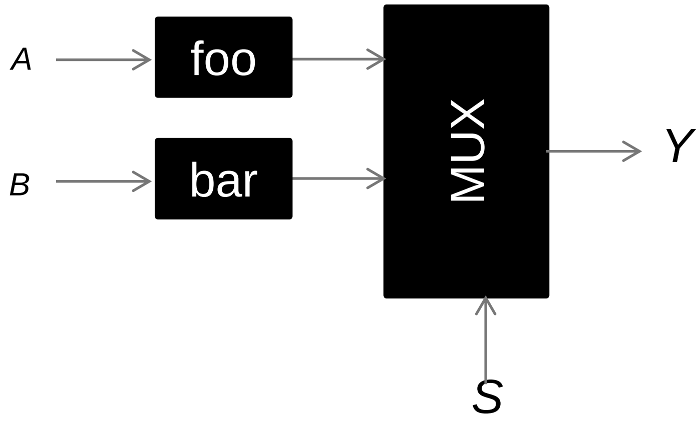
- Y = S ? foo(A) : bar(B)



- cum implementăm “if”-ul de mai sus?

MUX, EX 5 C

- **Y = S ? foo(A) : bar(B)**

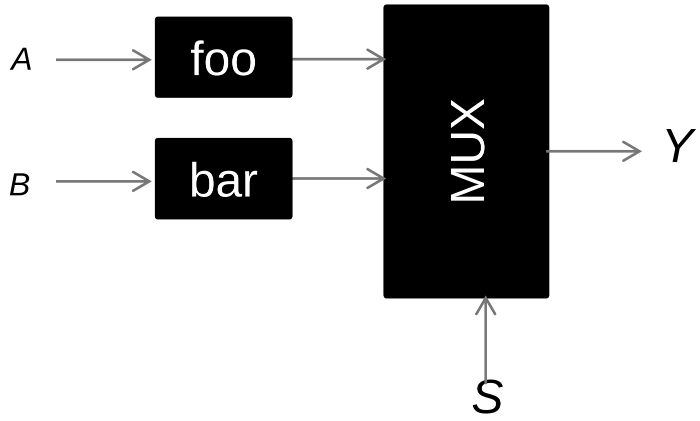


I_0 foo(X)	I_1 bar(Y)	S	Y
*	*	0	I_1
*	*	1	I_0

- **care e diferența cu un limbaj de programare?**

MUX, EX 5 C

- **Y = S ? foo(A) : bar(B)**

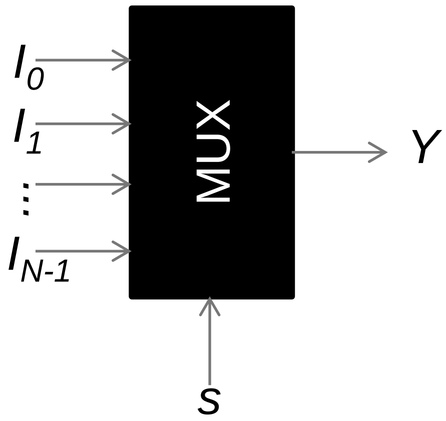


I_0 foo(X)	I_1 bar(Y)	S	Y
*	*	0	I_1
*	*	1	I_0

- **care e diferența cu un limbaj de programare?**
  - indiferent de valoarea lui S, se execută foo(A) și bar(B)
  - doar că la ieșire vedem doar una dintre funcții (cea selectată de S)

MUX, EX 5 D

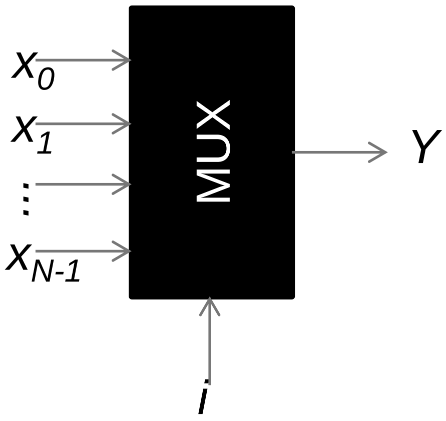
- vrem să accesăm un element al unui vector x_i



- ce putem la intrări?
- ce este semnalul s?

MUX, EX 5 D

- vrem să accesăm x_i

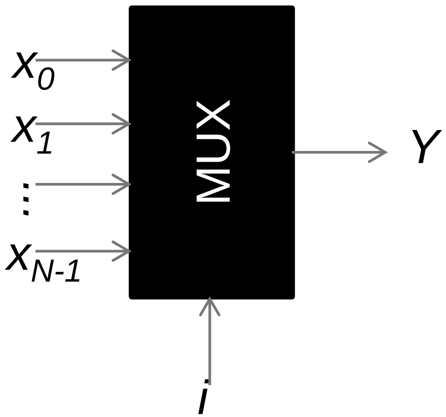


- ce putem la intrări? punem vectorul x
- ce este semnalul s? punem index-ul i

- care este dimensiunea intrării?
- care este dimensiunea lui s?

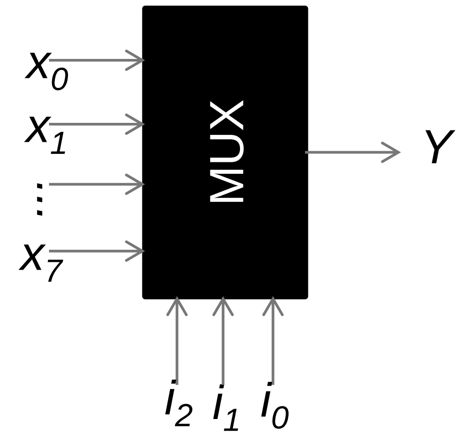
MUX, EX 5 D

- vrem să accesăm x_i



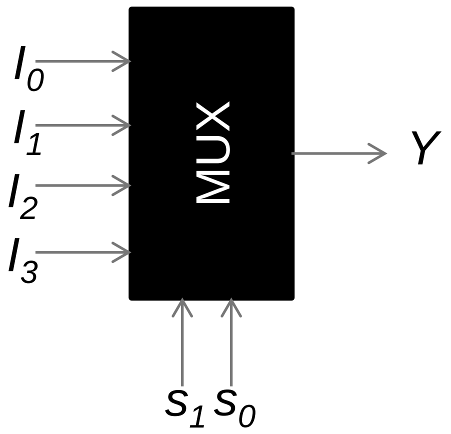
s_0 (i)	Y
000	I_0 (x_0)
001	I_1 (x_1)
010	I_2 (x_2)
011	I_3 (x_3)
100	I_4 (x_4)
101	I_5 (x_5)
110	I_6 (x_6)
111	I_7 (x_7)

- ce putem la intrări? punem vectorul x
- ce este semnalul s? punem index-ul i
- care este dimensiunea intrării? N
- care este dimensiunea lui s? ceil($\log_2$ N)

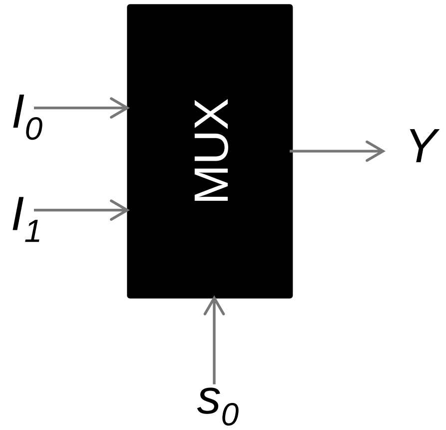


MUX, EX 5 E

- **un MUX cu 4 intrări**
  - automat știm că semnalul s are doi biți

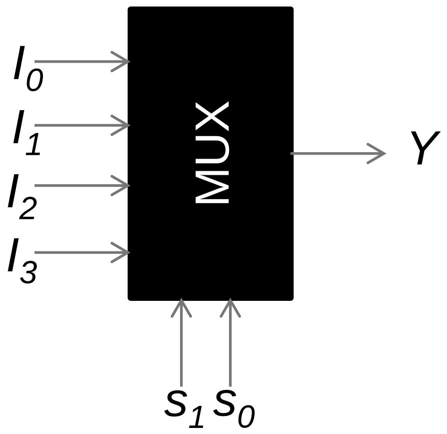


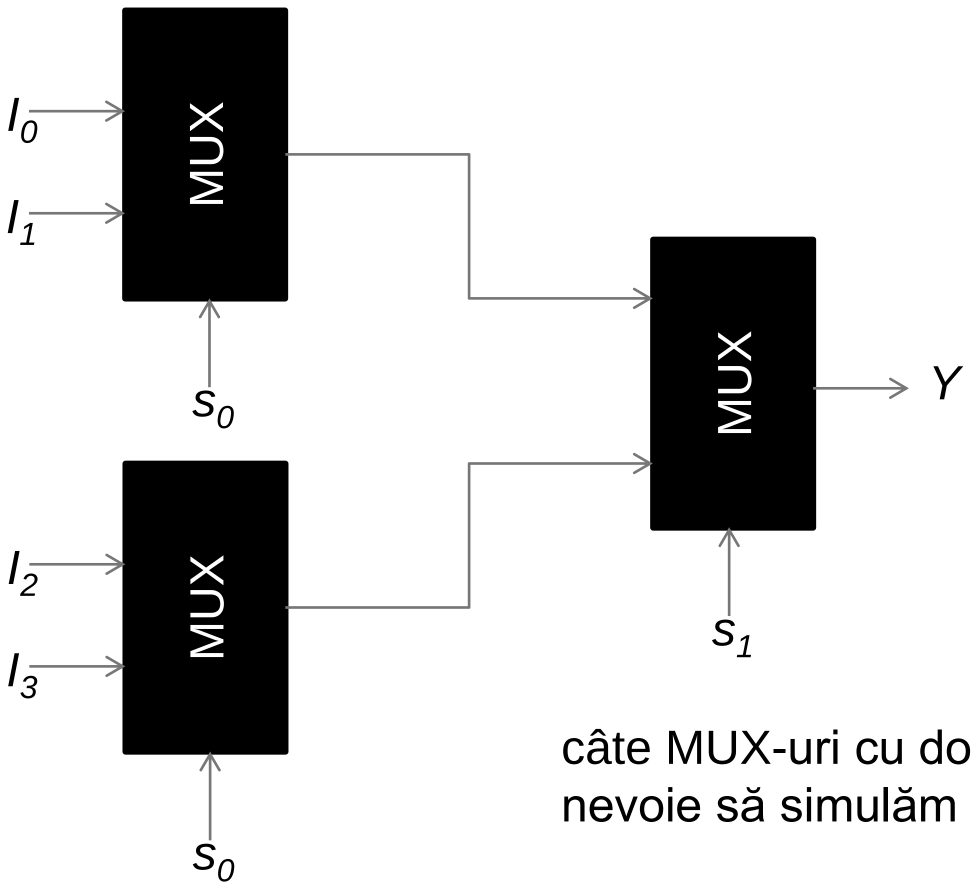
- **avem la dispoziție un MUX cu 2 intrări**



MUX, EX 5 E

- **un MUX cu 4 intrări**
  - automat știm că semnalul s are doi biți

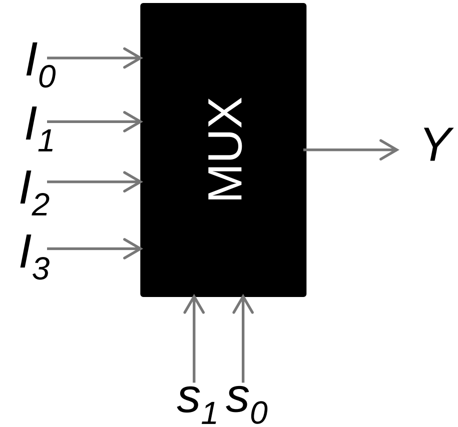


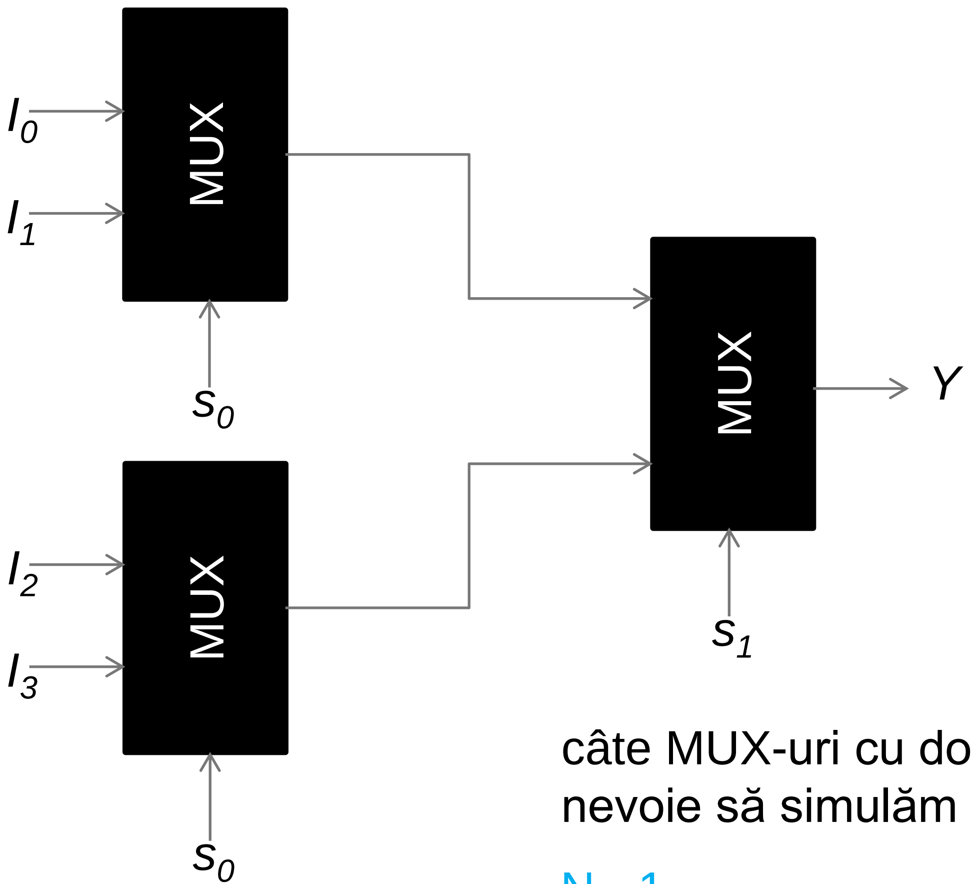


câte MUX-uri cu două întrari avem nevoie să simulăm un MUX cu N intrări?

MUX, EX 5 E

- **un MUX cu 4 intrări**
  - automat știm că semnalul s are doi biți





câte MUX-uri cu două întrari avem nevoie să simulăm un MUX cu N intrări?

N - 1

MUX, EX 7 A

- **numarul nostru x este**

x_3	x_2	x_1	x_0

- **deplasare normală cu 2 la dreapta**

0	0	x_3	x_2

echivalent cu o împărțire la 2^2

- **deplasare aritmetică cu 2 la dreapta**

x_3	x_3	x_3	x_2

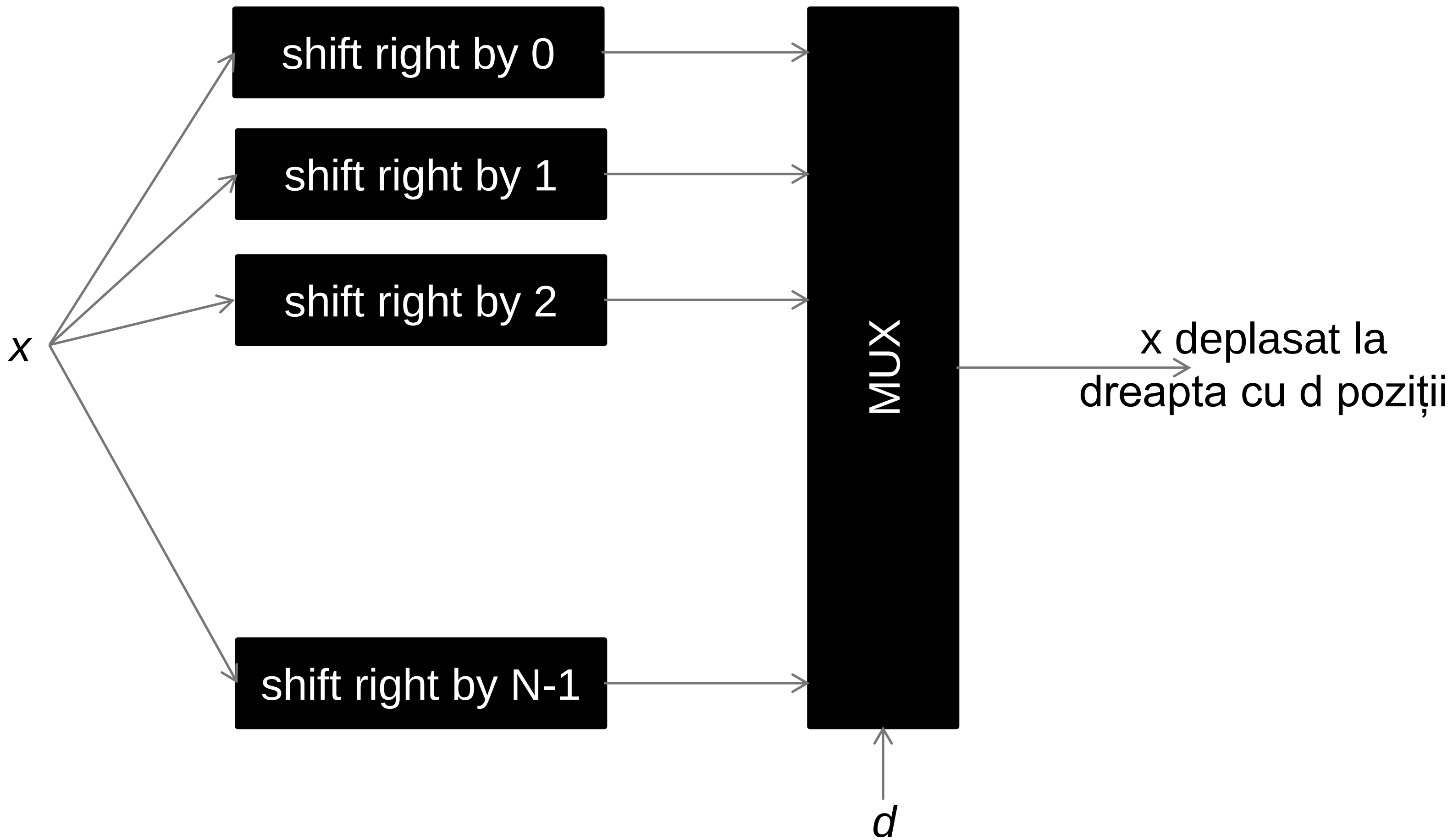
echivalent cu o împărțire la 2^2

- **deplasare circulară cu 2 la dreapta**

x_1	x_0	x_3	x_2

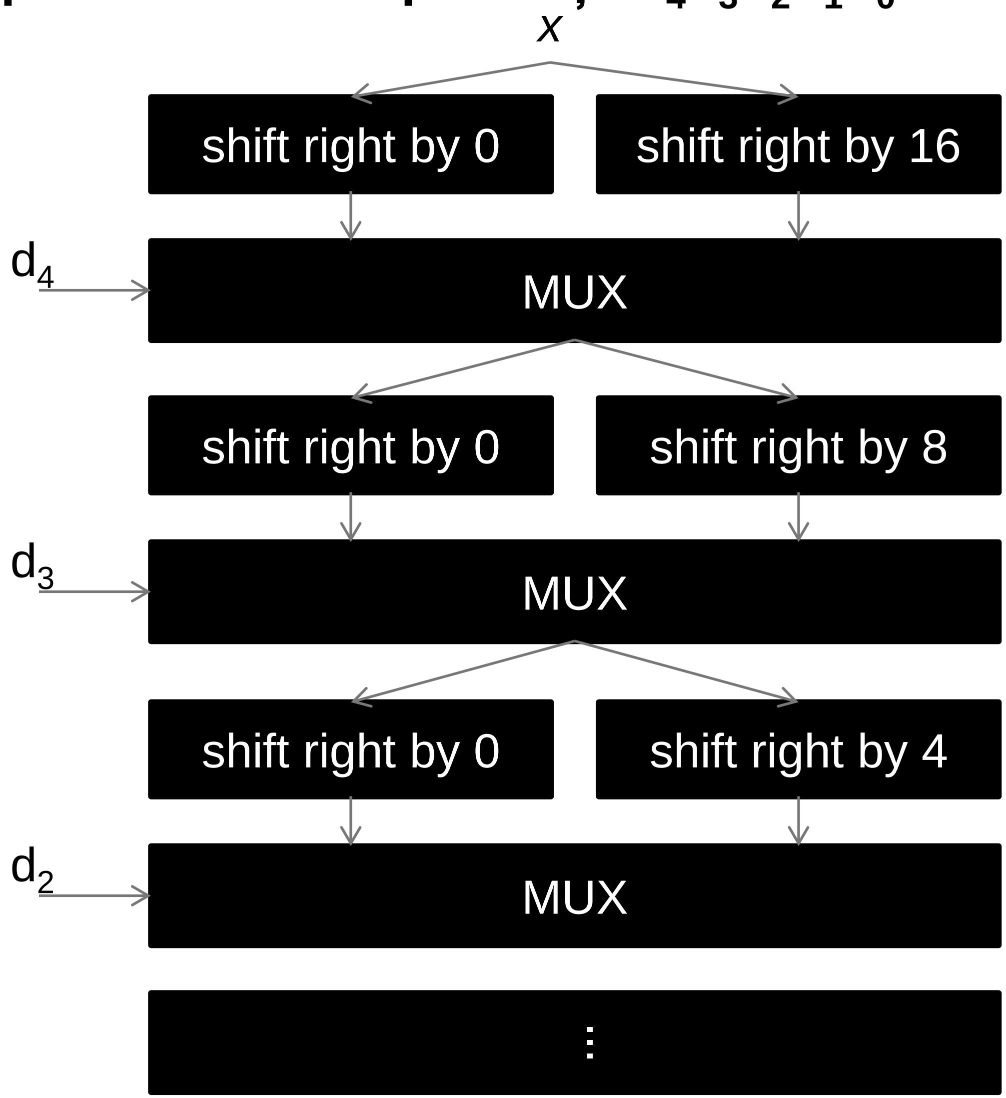
MUX, EX 7 B

- **deplasare a unui numar x cu d poziții la dreapta**



MUX, EX 7 C

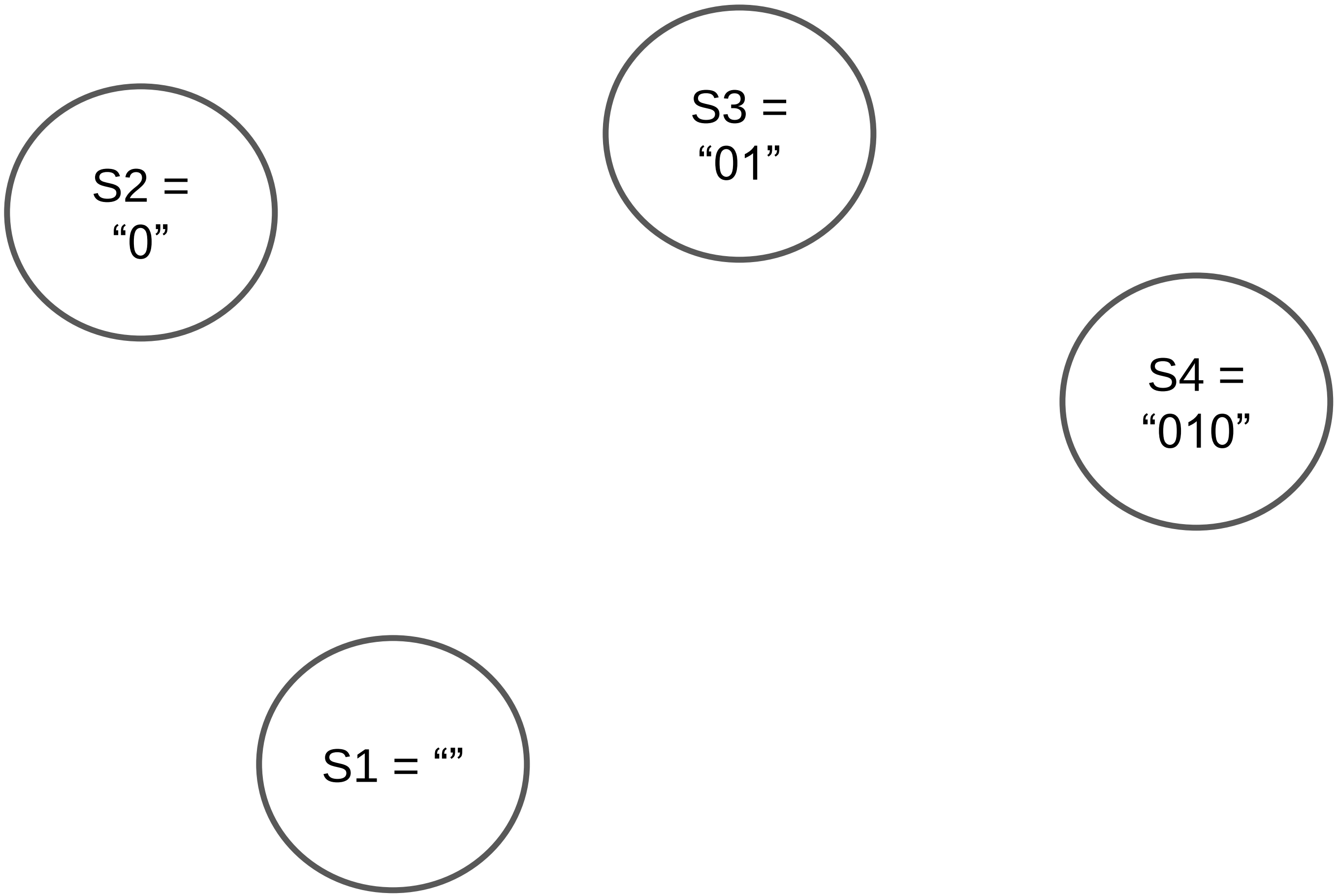
- presupunem că x este pe 32 de biți
- deplasarea d este pe 5 biți: $d_4d_3d_2d_1d_0$



avem nevoie de $\log_2 d$ MUX-uri

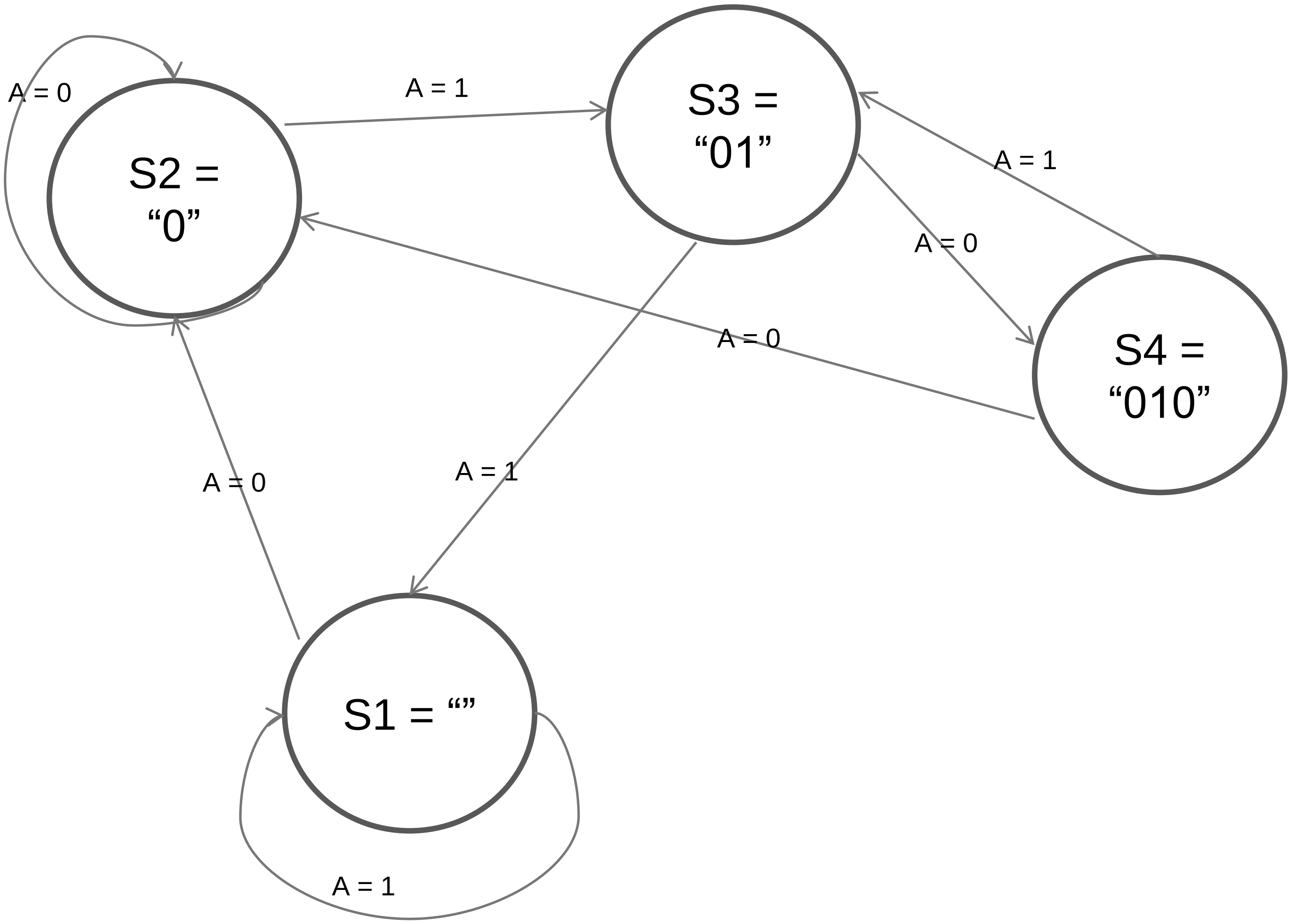
SECVENȚIAL, EX 10

- **definim 4 stări**

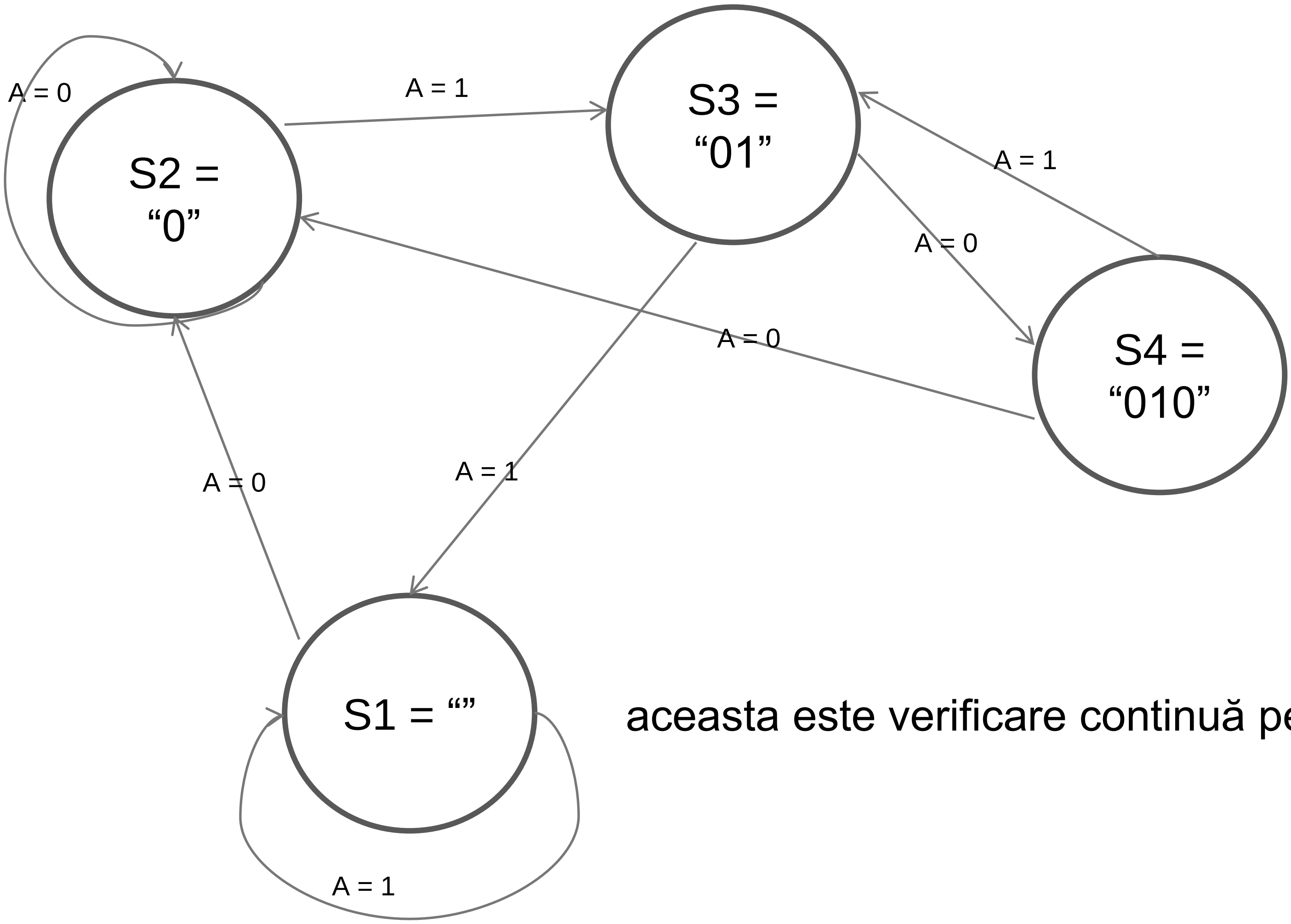


- care sunt tranzițiile între aceste stări?

SECVENȚIAL, EX 10

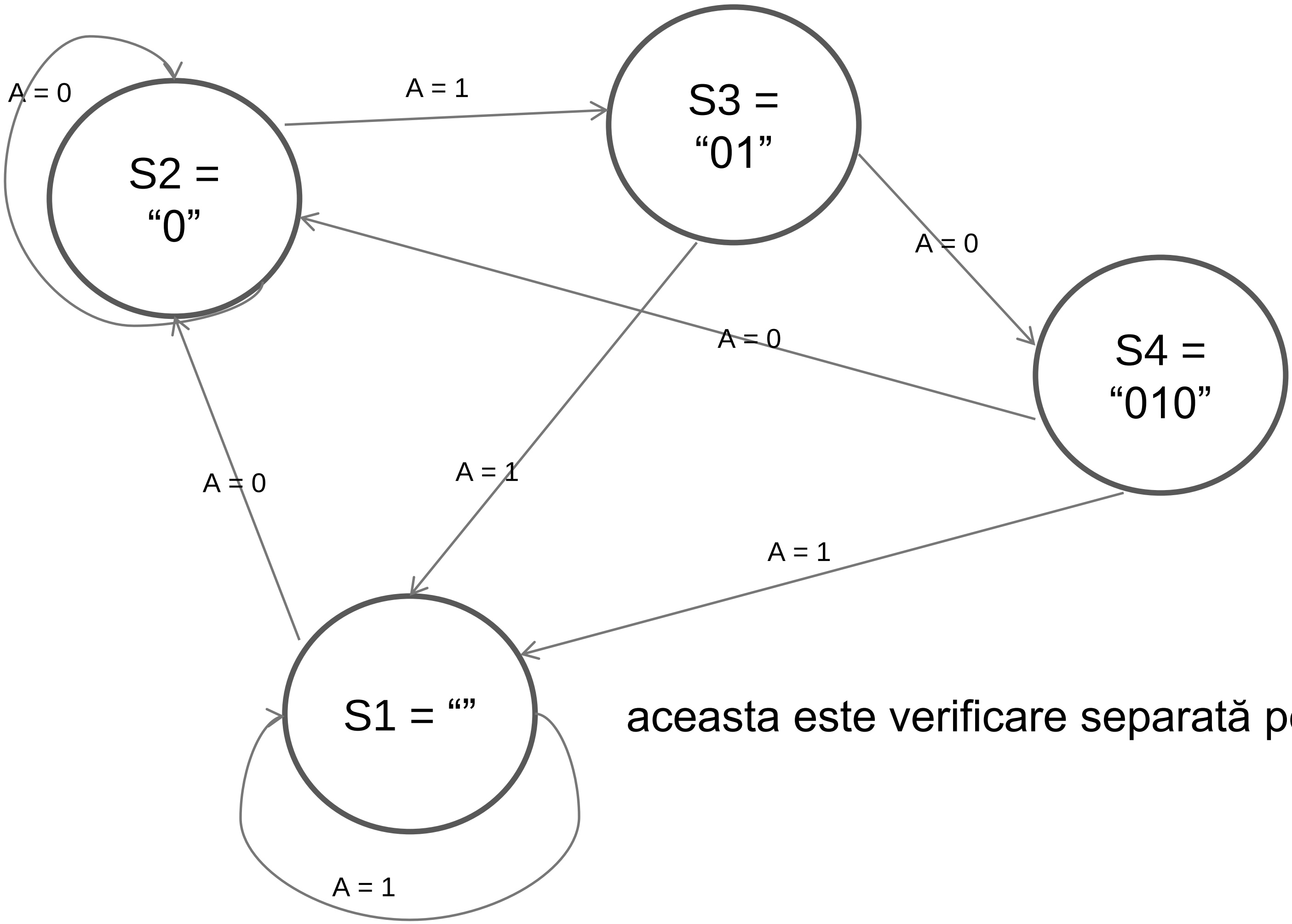


SECVENȚIAL, EX 10



aceasta este verificare continuă pentru 010

SECVENȚIAL, EX 10

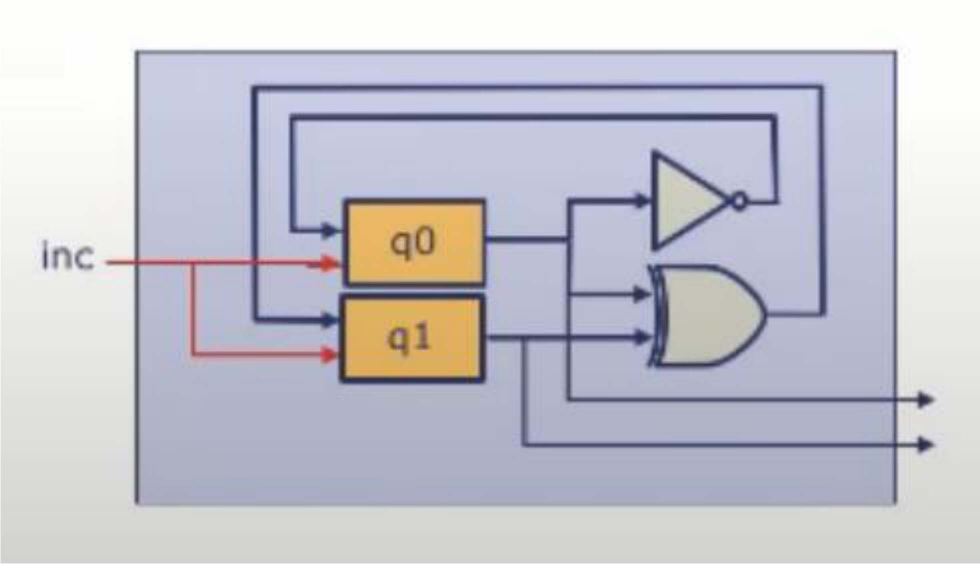


aceasta este verificare separată pentru 010

COUNTER 2 BIȚI, EX 12

- $q_0^{(t+1)} = !\mathrm{INC} \times q_0^{(t)} + \mathrm{INC} \times !q_0^{(t)}, \; q_1^{(t+1)} = !\mathrm{INC} \times q_1^{(t)} + \mathrm{INC} \times (q_1^{(t)} \otimes q_0^{(t)})$

$q_1^{(t)}$	$q_0^{(t)}$	$q_1^{(t+1)}$	$q_0^{(t+1)}$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0



- **aici INC e pe post de semnal Enable**

COUNTER 2 BIȚI, EX 12

- counter cod Gray

$q_2^{(t)}$	$q_1^{(t)}$	$q_0^{(t)}$	$q_2^{(t+1)}$	$q_1^{(t+1)}$	$q_0^{(t+1)}$
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	1

- $q_0^{(t+1)} = \ !q_2^{(t)}!q_1^{(t)}!q_0^{(t)} + !q_2^{(t)}!q_1^{(t)}q_0^{(t)} + q_2^{(t)}q_1^{(t)}!q_0^{(t)} + q_2^{(t)}q_1^{(t)}q_0^{(t)}$
  $= q_2^{(t)}q_1^{(t)} + !q_2^{(t)}!q_1^{(t)}$
- $q_1^{(t+1)} = \ldots$
- $q_2^{(t+1)} = \ldots$

CMMDC, EX 13

- **implementați algoritmul lui Euclid pentru CMMDC folosind un circuit secvențial.**
- **exemplu: se dau a = 15, b = 6**
  - pasul 1: a = 9, b = 6 (a := a – b)
  - pasul 2: a = 3, b = 6 (a := a – b)
  - pasul 3: a = 6, b = 3 (a,b := b,a)
  - pasul 4: a = 3, b = 3 (a := a – b)
  - pasul 5: a = 0, b = 3
  - răspunsul este 3
- **codul python:**

```
def cmmdc(a, b):
    if a == b: return b
    elif a > b: return cmmdc(a-b, b)
    else: return cmmdc(b, a)
```

CMMDC, EX 13

- **codul python:**

```
def cmmdc(a, b):
    if a == b: return b
    elif a > b: return cmmdc(a-b, b)
    else: return cmmdc(b, a)
```

- avem variabilele $a^{(t)}$ și $b^{(t)}$
- conform algoritmului de mai sus avem ecuațiile de evoluție
  - facem ceva doar dacă $a^{(t)}$!= 0

CMMDC, EX 13

- **codul python:**

```
def cmmdc(a, b):
    if a == b: return b
    elif a > b: return cmmdc(a-b, b)
    else: return cmmdc(b, a)
```

- avem variabilele $a^{(t)}$ și $b^{(t)}$
- conform algoritmului de mai sus avem ecuațiile de evoluție
  - facem ceva doar dacă $a^{(t)} != 0$
  - $p = (a^{(t)} > b^{(t)})$
  - $a^{(t+1)} = ?$
  - $b^{(t+1)} = ?$

CMMDC, EX 13

- **codul python:**

```
def cmmdc(a, b):
    if a == b: return b
    elif a > b: return cmmdc(a-b, b)
    else: return cmmdc(b, a)
```

- avem variabilele $a^{(t)}$ și $b^{(t)}$
- conform algoritmului de mai sus avem ecuațiile de evoluție
  - facem ceva doar dacă $a^{(t)} != 0$
  - $p = (a^{(t)} > b^{(t)})$
  - $a^{(t+1)} = p \times (a^{(t)} - b^{(t)}) + !p \times b^{(t)}$
  - $b^{(t+1)} = p \times b^{(t)} + !p \times a^{(t)}$

CMMDC, EX 13

- **codul python:**

```
def cmmdc(a, b):
        if a == b: return b
        elif a > b: return cmmdc(a-b, b)
        else: return cmmdc(b, a)
```

ce fel de circuite sunt a și b?

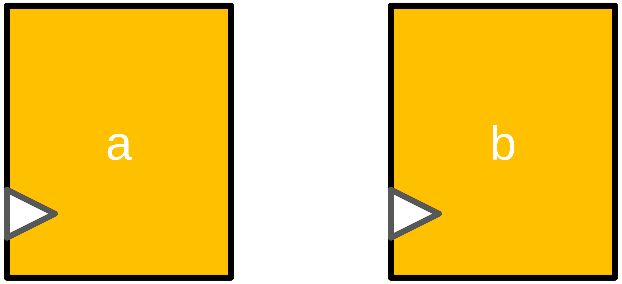
.

CMMDC, EX 13

- **codul python:**

```
def cmmdc(a, b):
    if a == b: return b
    elif a > b: return cmmdc(a-b, b)
    else: return cmmdc(b, a)
```

când sunt acești regiștri activi?



.

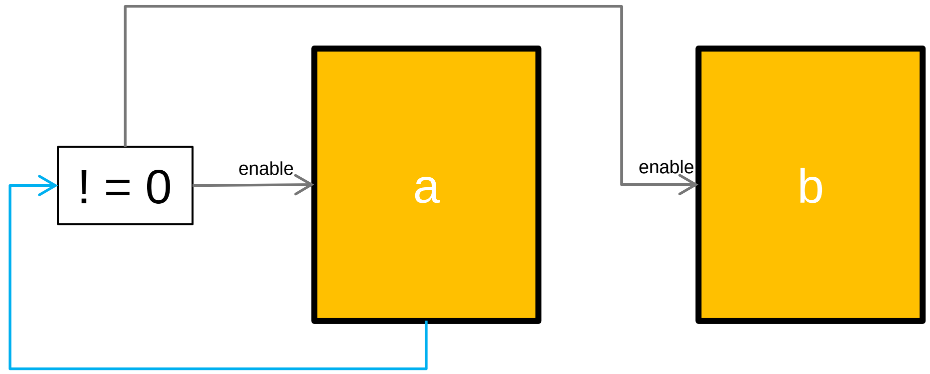
CMMDC, EX 13

- **codul python:**

```
def cmmdc(a, b):
    if a == b: return b
    elif a > b: return cmmdc(a-b, b)
    else: return cmmdc(b, a)
```

$p = (a^{(t)} > b^{(t)})$
$a^{(t+1)} = p \times (a^{(t)} - b^{(t)}) + !p \times b^{(t)}$
$b^{(t+1)} = p \times b^{(t)} + !p \times a^{(t)}$

cum se schimbă a și b?



CMMDC, EX 13

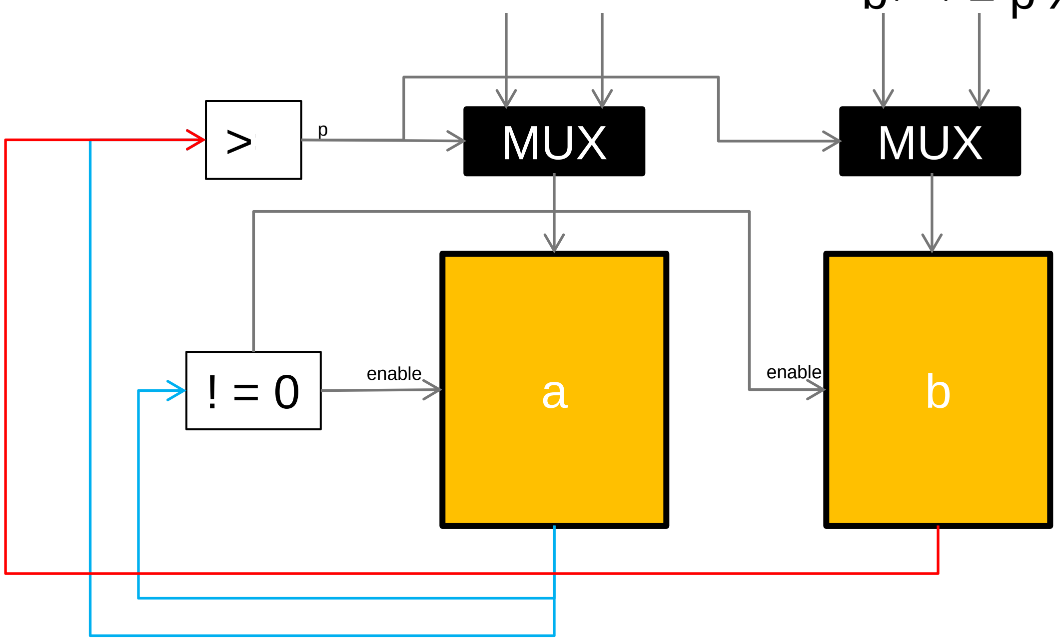
- **codul python:**

```
def cmmdc(a, b):
    if a == b: return b
    elif a > b: return cmmdc(a-b, b)
    else: return cmmdc(b, a)
```

$p = (a^{(t)} > b^{(t)})$

$a^{(t+1)} = p \times (a^{(t)} - b^{(t)}) + !p \times b^{(t)}$

$b^{(t+1)} = p \times b^{(t)} + !p \times a^{(t)}$



care sunt întrările pentru MUX 1?

CMMDC, EX 13

- **codul python:**

def cmmdc(a, b):

if a == b: return b

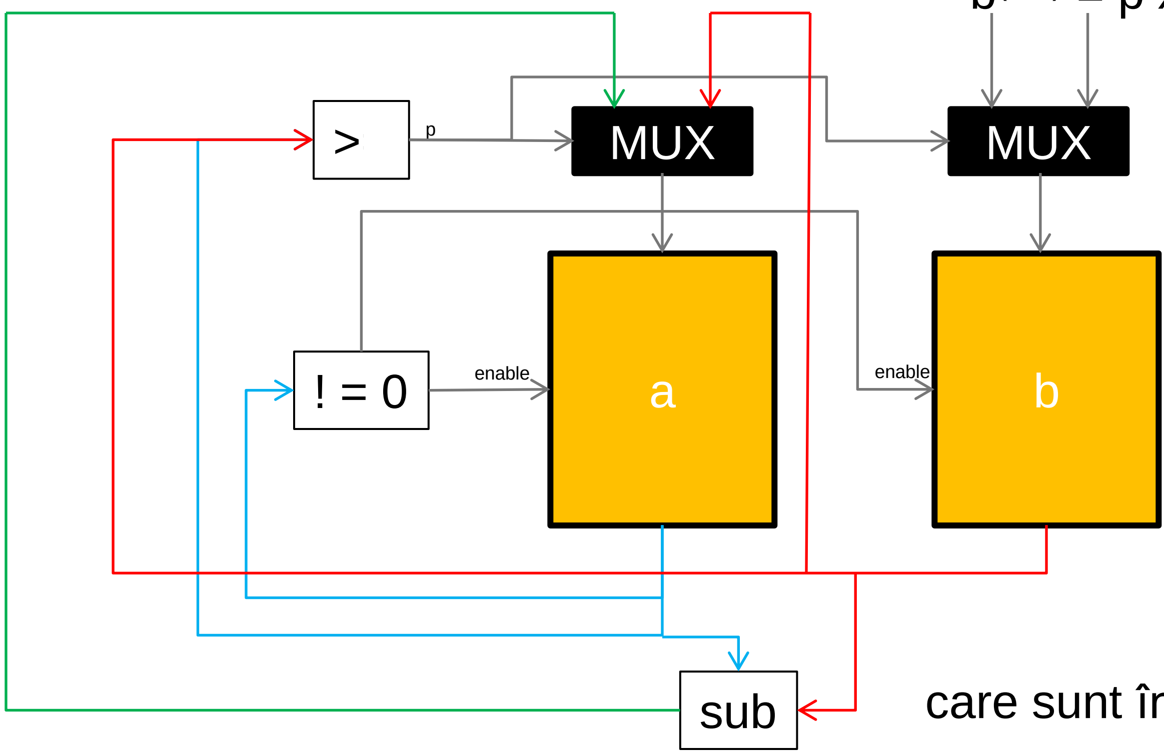
elif a > b: return cmmdc(a-b, b)

else: return cmmdc(b, a)

$p = (a^{(t)} > b^{(t)})$

$a^{(t+1)} = p \times (a^{(t)} - b^{(t)}) + !p \times b^{(t)}$

$b^{(t+1)} = p \times b^{(t)} + !p \times a^{(t)}$



care sunt întrările pentru MUX 2?

CMMDC, EX 13

- **codul python:**

def cmmdc(a, b):

if a == b: return b

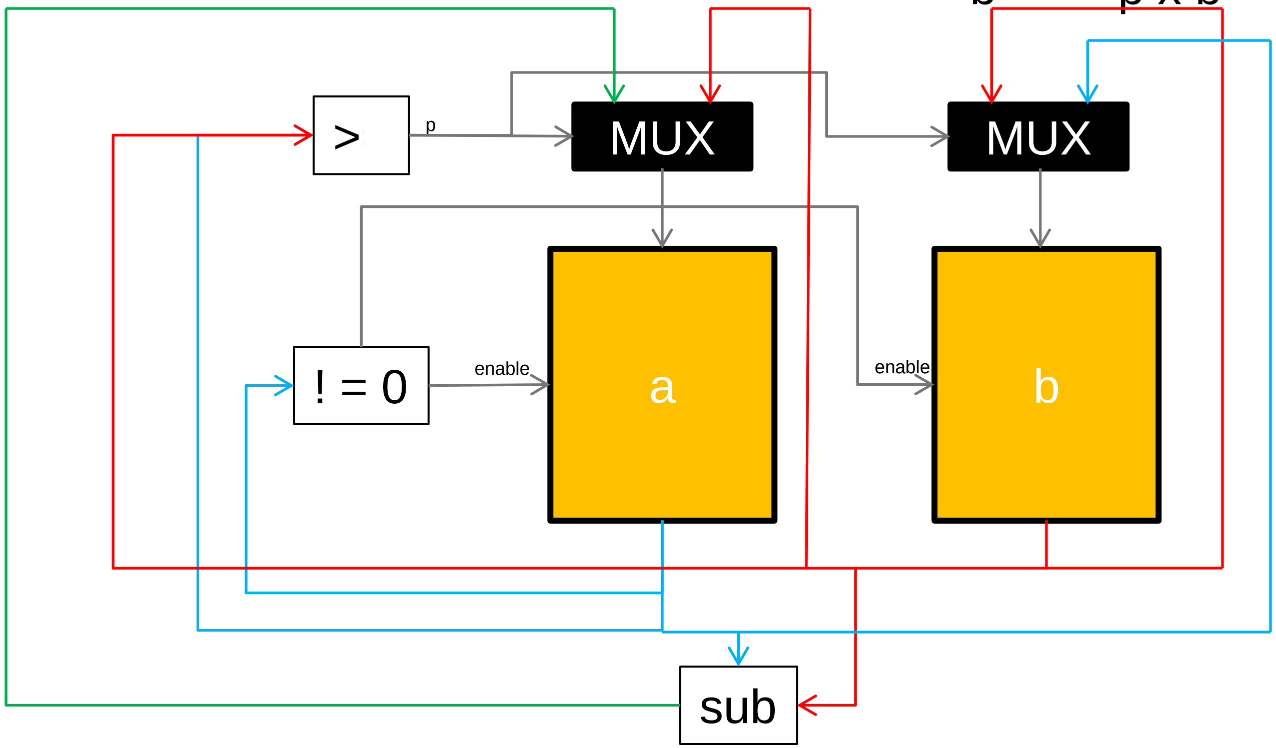
elif a > b: return cmmdc(a-b, b)

else: return cmmdc(b, a)

$p = (a^{(t)} > b^{(t)})$

$a^{(t+1)} = p \times (a^{(t)} - b^{(t)}) + !p \times b^{(t)}$

$b^{(t+1)} = p \times b^{(t)} + !p \times a^{(t)}$



ARHITECTURA SISTEMELOR DE CALCUL SEMINAR 0x03

NOTIȚE SUPORT SEMINAR

Cristian Rusu

ÎNMULȚIREA NUMERELOR, EX. 1

1 0 1 0	*a*
1 1 0 1	*b*
	s

ÎNMULȚIREA NUMERELOR, EX. 1

				1	0	1	0	*a*
				1	1	0	1	*b*
				1	0	1	0	*s*
			0	0	0	0		
		1	0	1	0			
	1	0	1	0				
1	0	0	0	0	0	1	0	

ÎNMULȚIREA NUMERELOR, EX. 1

1 0 1 0	*a = 10*
1 1 0 1	*b = 13*

---

1 0 1 0	*s = 130*
0 0 0 0	
1 0 1 0	
1 0 1 0	

---

1 0 0 0 0 0 1 0

ÎNMULȚIREA NUMERELOR, EX. 2

1 0 1 0	*a*
1 1 0 1	*b*
0 0 0 0	*s*

ÎNMULȚIREA NUMERELOR, EX. 2

	1 1 1 1 1 0 1 0	*a*
	1 1 1 1 1 1 0 1	*b*
	1 1 1 1 1 0 1 0	*s*
	0 0 0 0 0 0 0 0	
	1 1 1 1 1 0 1 0	
	1 1 1 1 1 0 1 0	⋮
	0 0 0 1 0 0 1 0	

ÎNMULȚIREA NUMERELOR, EX. 2

```
      1 1 1 1 1 0 1 0        a = -6

      1 1 1 1 1 1 0 1        b = -3
      ---------------
      1 1 1 1 1 0 1 0        s  = 18
    0 0 0 0 0 0 0 0
  1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
                 ⋮
      ---------------
      0 0 0 1 0 0 1 0
```

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția:
- **a x 16**
  - soluția:
- **a x 3**
  - soluția:
- **a x 7**
  - soluția:
- **a / 8**
  - soluția:
- **a mod 16**
  - soluția:
- **a x 72**
  - soluția:

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția: a << 1, sau a + a
- **a x 16**
  - soluția:
- **a x 3**
  - soluția:
- **a x 7**
  - soluția:
- **a / 8**
  - soluția:
- **a mod 16**
  - soluția:
- **a x 72**
  - soluția:

.

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția: a << 1, sau a + a
- **a x 16**
  - soluția: a << 4
- **a x 3**
  - soluția:
- **a x 7**
  - soluția:
- **a / 8**
  - soluția:
- **a mod 16**
  - soluția:
- **a x 72**
  - soluția:

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția: a << 1, sau a + a
- **a x 16**
  - soluția: a << 4
- **a x 3**
  - soluția: a << 1 + a
- **a x 7**
  - soluția:
- **a / 8**
  - soluția:
- **a mod 16**
  - soluția:
- **a x 72**
  - soluția:

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția: a << 1, sau a + a
- **a x 16**
  - soluția: a << 4
- **a x 3**
  - soluția: a << 1 + a
- **a x 7**
  - soluția: a << 3 – a
- **a / 8**
  - soluția:
- **a mod 16**
  - soluția:
- **a x 72**
  - soluția:

.

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția: a << 1, sau a + a
- **a x 16**
  - soluția: a << 4
- **a x 3**
  - soluția: a << 1 + a
- **a x 7**
  - soluția: a << 3 – a
- **a / 8**
  - soluția: a >> 3
- **a mod 16**
  - soluția:
- **a x 72**
  - soluția:

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția: a << 1, sau a + a
- **a x 16**
  - soluția: a << 4
- **a x 3**
  - soluția: a << 1 + a
- **a x 7**
  - soluția: a << 3 – a
- **a / 8**
  - soluția: a >> 3
- **a mod 16**
  - soluția: a & 0x000F
- **a x 72**
  - soluția:

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a x 2**
  - soluția: a << 1, sau a + a
- **a x 16**
  - soluția: a << 4
- **a x 3**
  - soluția: a << 1 + a
- **a x 7**
  - soluția: a << 3 – a
- **a / 8**
  - soluția: a >> 3
- **a mod 16**
  - soluția: a & 0x000F
- **a x 72**
  - soluția: a << 6 + a << 3

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a mod 16**
  - soluția: a & 0x000F

- **a div 16**
  - soluția:

ÎNMULȚIREA RAPIDĂ, EX. 3

- **a mod 16**
  - soluția: a & 0x000F

- **a div 16**
  - soluția: (a & FFF0) >> 4, sau doar a >> 4

  - de asemenea: a = a & FFF0 + a & 000F = div cu 16 + mod cu 16

ÎNMULȚIREA RAPIDĂ, EX. 4

- **a x 2**
  - soluția:

ÎNMULȚIREA RAPIDĂ, EX. 4

- **a x 2**
  - soluția: a = (a < 0) ? -((-a) << 1)) : a << 1

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

1 0 1 0 1 0

1 0

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

$\boxed{1}$ 0 1 0 1 0

1 0

0

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

1 0 1 0 1 0

1 0

---

0 1

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

$$\begin{array}{l} 1\,0\,1\,0 \\ 1\,0 \\ \hline 0\,1\,0 \end{array}$$

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

1 0 1 0

1 0

0 1 0 1

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

1 0

1 0

---

0 1 0 1 0

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

1 0

1 0

0 1 0 1 0 1

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

1 0 1 0 1 0	*a*
1 0	*b*
0 1 0 1 0 1	*s*

ÎMPĂRȚIREA, EX. 5

- **101010 / 10**

1	0	1	0	1	0

a = *42*

1	0

b = *2*

0	1	0	1	0	1

s = *21*

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

1 1 1 0 1 0 1 0 1

1 1

.

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

$\boxed{1}$	1	1	0	1	0	1	0	1

1	1

0

.

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

1 1 1 0 1 0 1 0 1

1 1

0 1

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

1 0 1 0 1 0 1

1 1

0 1 0

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

1 0 1 0 1 0 1

1 1

0 1 0 0

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

1 0 1 0 1 0 1

1 1

0 1 0 0 1

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

			1	0	0	1	0	1

1	1

0	1	0	0	1	1

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

1 1 0 1

1 1

0 1 0 0 1 1 1

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

							0	1

1	1

0	1	0	0	1	1	1	0

.

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

0 1

acesta este restul, 1 în zecimal

1 1

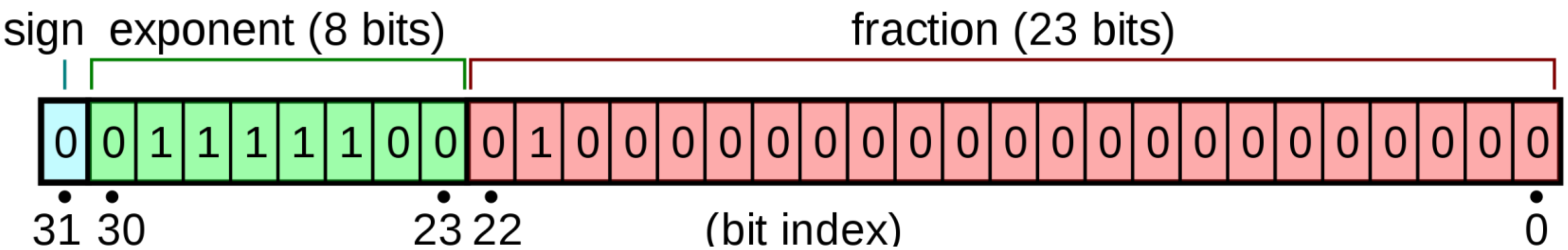
0 1 0 0 1 1 1 0 0

ÎMPĂRȚIREA, EX. 5

- **111010101 / 11**

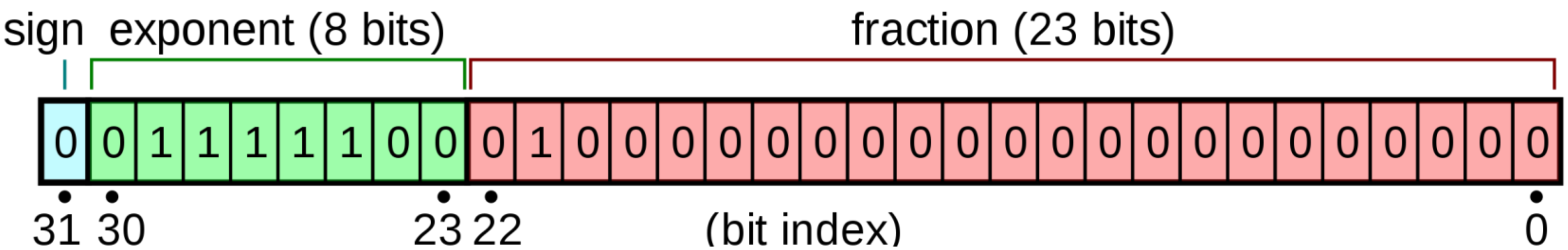
1 1 1 0 1 0 1 0 1	*a = 469*
1 1	*b = 3*
0 1 0 0 1 1 1 0 0	*s = 156*
0 1	acesta este restul, 1 în zecimal

CONVERSIA ÎN IEEE FP, EX. 7



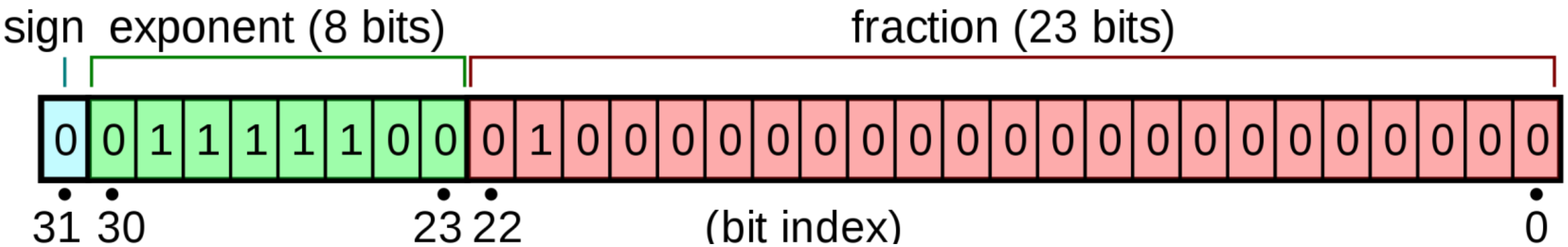
- -1313.3125
  - partea întreagă este: ?
  - partea fracționară: ?

CONVERSIA ÎN IEEE FP, EX. 7



- -1313.3125
  - partea întreagă este: 1313
  - partea fracționară: 0.3125
    - 0.3125 × 2 = 0.625 => 0
    - 0.625 × 2 = 1.25 => 1
    - 0.25 × 2 = 0.5 => 0
    - 0.5× 2 = 1.0 => 1
  - deci, $1313.3125_{10} = ?_2$

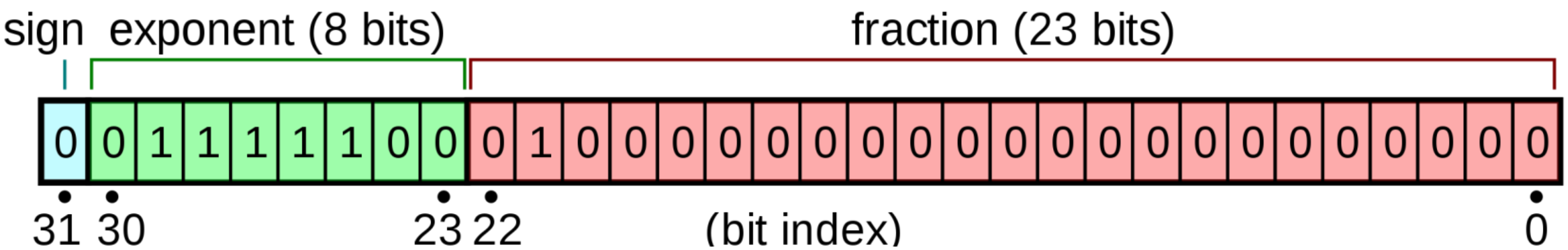
CONVERSIA ÎN IEEE FP, EX. 7



- -1313.3125
  - partea întreagă este: 1313
  - partea fracționară: 0.3125
    - 0.3125 × 2 = 0.625 => 0
    - 0.625 × 2 = 1.25 => 1
    - 0.25 × 2 = 0.5 => 0
    - 0.5× 2 = 1.0 => 1
  - deci, $1313.3125_{10} = 10100100001.0101_2$
  - normalizare: ?

  - mantisa este ?
  - exponentul este ?
  - semnul este ?

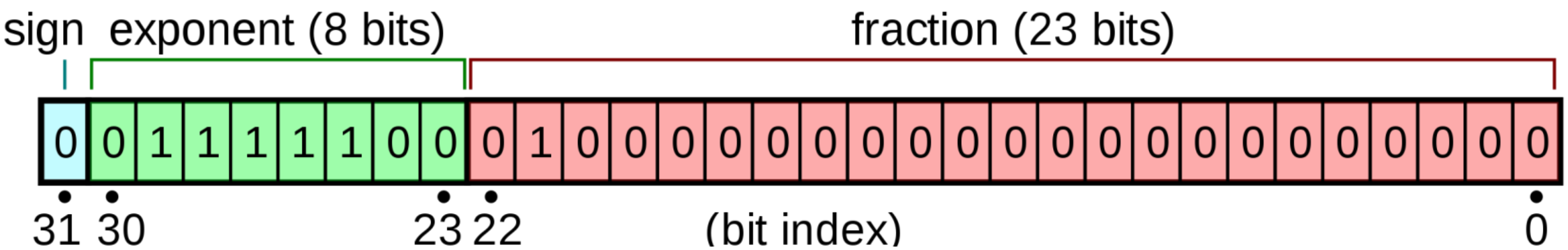
CONVERSIA ÎN IEEE FP, EX. 7



- -1313.3125
  - partea întreagă este: 1313
  - partea fracționară: 0.3125
    - 0.3125 × 2 = 0.625 => 0
    - 0.625 × 2 = 1.25 => 1
    - 0.25 × 2 = 0.5 => 0
    - 0.5× 2 = 1.0 => 1
  - deci, $1313.3125_{10} = 10100100001.0101_2$
  - normalizare: $10100100001.0101_2 = 1.01001000010101_2 \times 2^{10}$

  - mantisa este ?
  - exponentul este ?
  - semnul este ?

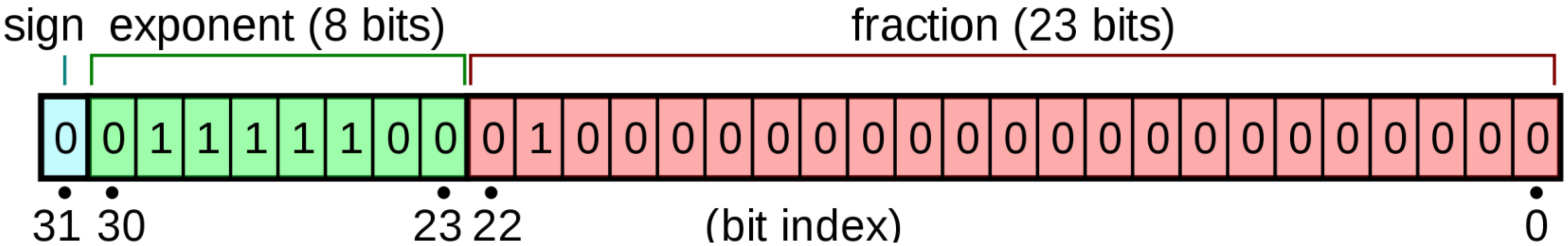
CONVERSIA ÎN IEEE FP, EX. 7



- -1313.3125
  - partea întreagă este: 1313
  - partea fracționară: 0.3125
    - 0.3125 × 2 = 0.625 => 0
    - 0.625 × 2 = 1.25 => 1
    - 0.25 × 2 = 0.5 => 0
    - 0.5× 2 = 1.0 => 1
  - deci, $1313.3125_{10} = 10100100001.0101_2$
  - normalizare: $10100100001.0101_2 = 1.01001000010101_2 \times 2^{10}$

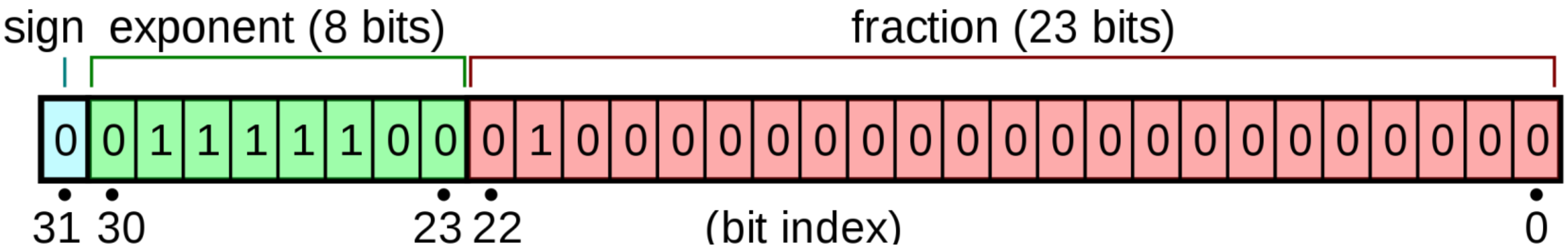
  - mantisa este 01001000010101000000000
  - exponentul este $10 + 127 = 137 = 10001001_2$
  - semnul este 1

CONVERSIA ÎN IEEE FP, EX. 8



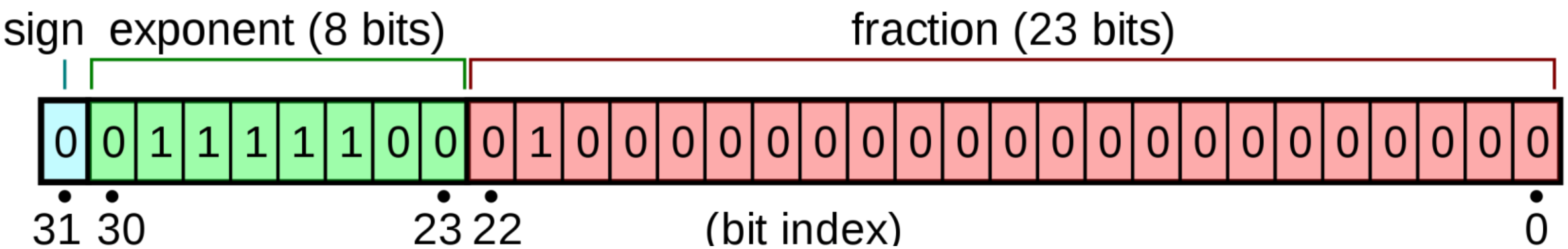
- **calculăm abs(a)**

CONVERSIA ÎN IEEE FP, EX. 8



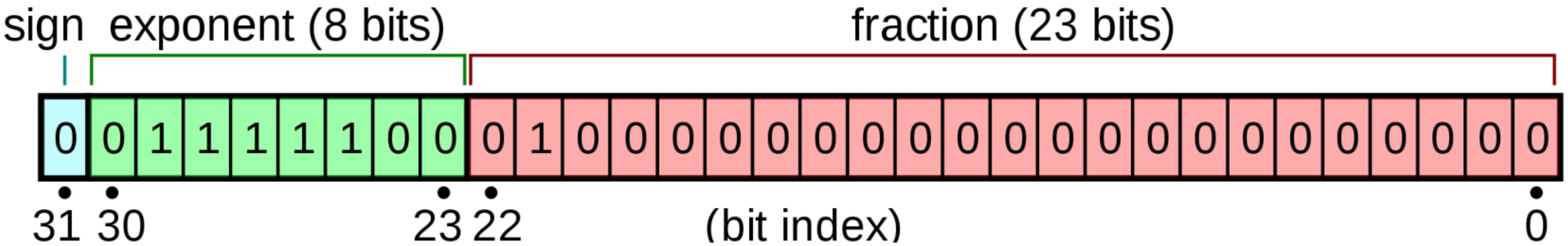
- **calculăm abs(a)**
    - **soluția: a = a & ~(1 << 31)**

CONVERSIA ÎN IEEE FP, EX. 8



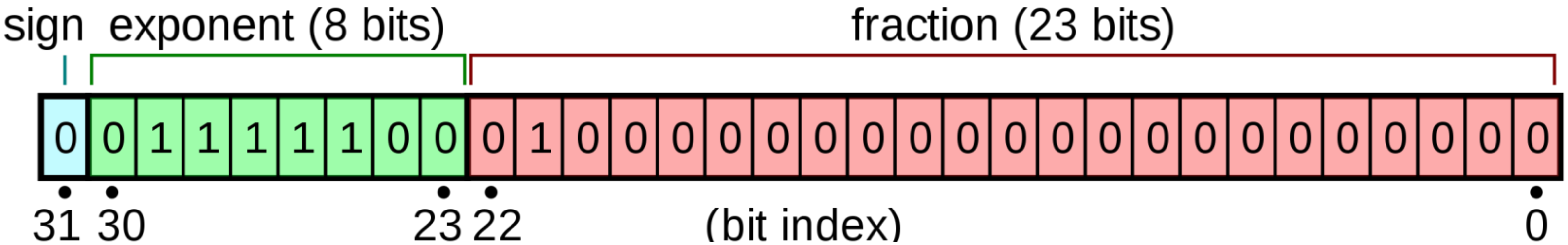
- **schimbați semnul lui a**

CONVERSIA ÎN IEEE FP, EX. 8



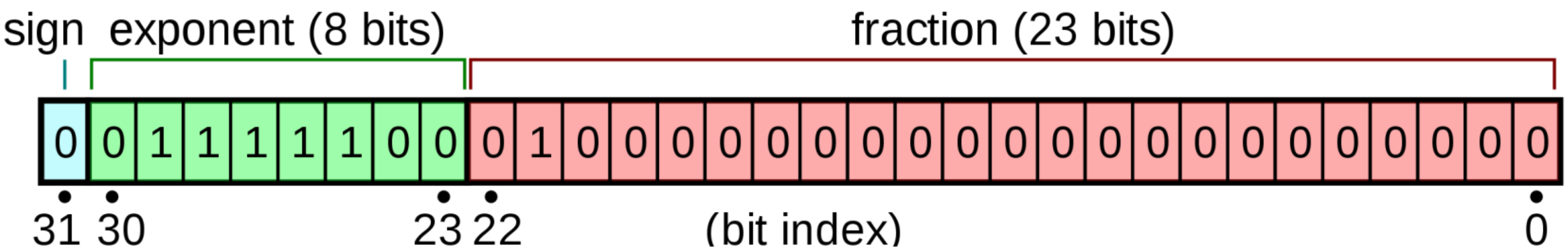
- **schimbați semnul lui a**
  - **soluția: a = a ^ (1 << 31)**

CONVERSIA ÎN IEEE FP, EX. 8



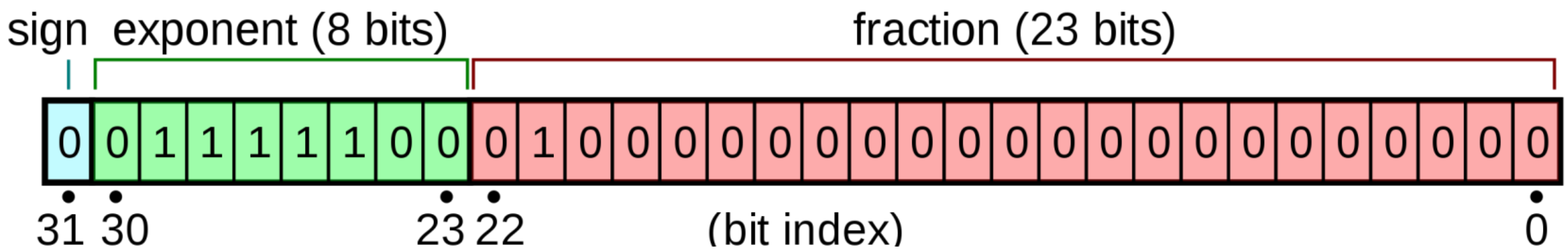
- **împărțiți a la 4**

CONVERSIA ÎN IEEE FP, EX. 8



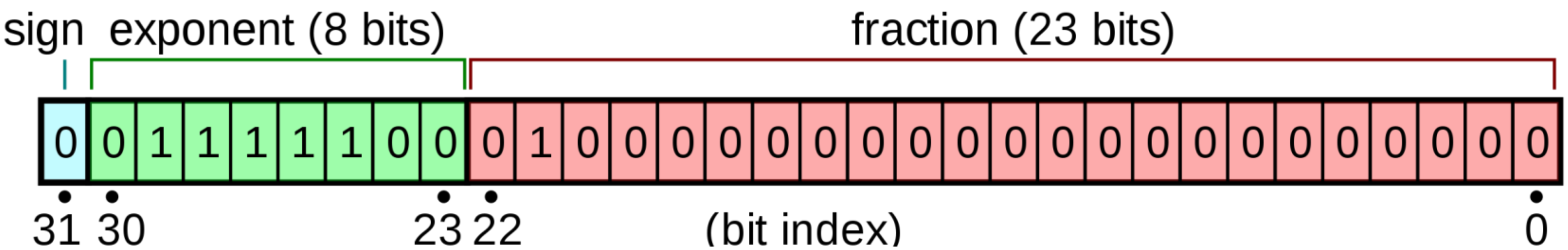
- **împărțiți a la 4**
  - **soluția:**
    - **vrem exponentul, unde se află?**
      - **MASK =**
    - **extragem exponent =**
    - **dacă exponent > 1 atunci ...**
    - **trebuie să actualizăm a**
      - **a =**

CONVERSIA ÎN IEEE FP, EX. 8



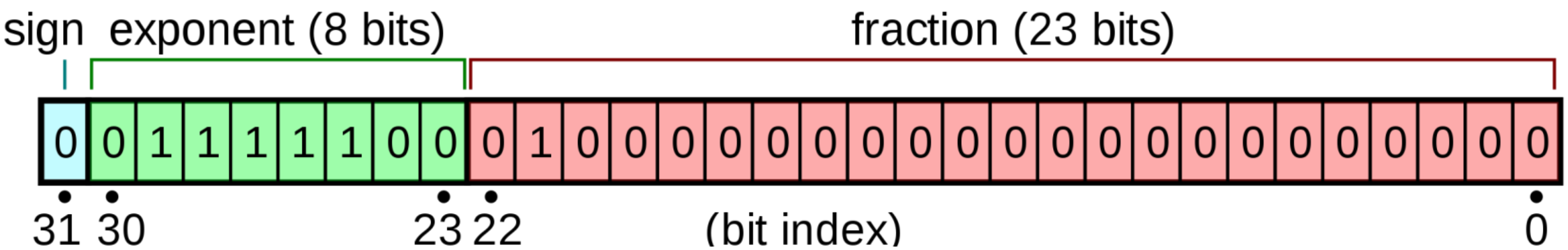
- **împărțiți a la 4**
  - **soluția:**
    - **vrem exponentul, unde se află?**
      - **MASK = 0x7F800000**
    - **extragem exponent =**
    - **dacă exponent > 1 atunci ...**
    - **trebuie să actualizăm a**
      - **a =**

CONVERSIA ÎN IEEE FP, EX. 8



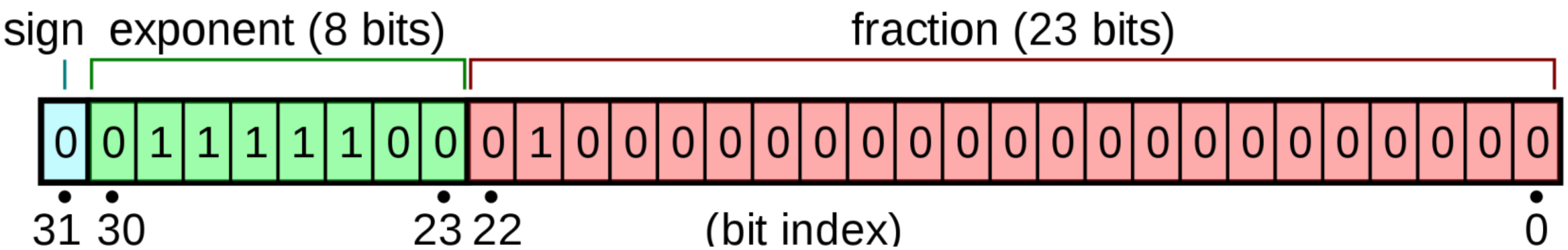
- împărțiți a la 4
  - soluția:
    - vrem exponentul, unde se află?
      - MASK = 0x7F800000
    - extragem exponent = (a & MASK) >> 23
    - dacă exponent > 1 atunci ...
    - trebuie să actualizăm a
      - a =

CONVERSIA ÎN IEEE FP, EX. 8



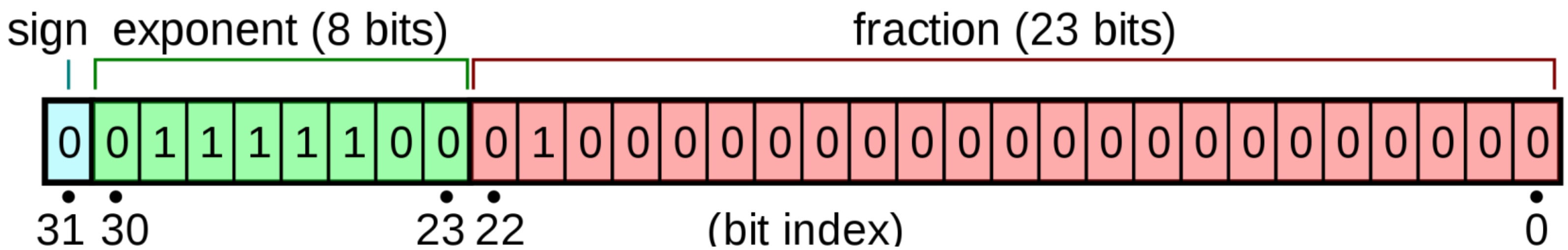
- **împărțiți a la 4**
  - **soluția:**
    - **vrem exponentul, unde se află?**
      - **MASK = 0x7F800000**
    - **extragem exponent = (a & MASK) >> 23**
    - **dacă exponent > 1 atunci exponent = exponent – 2, altfel a = 0**
    - **trebuie să actualizăm a**
      - **a =**

CONVERSIA ÎN IEEE FP, EX. 8



- **împărțiți a la 4**
  - **soluția:**
    - **vrem exponentul, unde se află?**
      - **MASK = 0x7F800000**
    - **extragem exponent = (a & MASK) >> 23**
    - **dacă exponent > 1 atunci exponent = exponent – 2, altfel a = 0**
    - **trebuie să actualizăm a**
      - **a = (a & ~MASK) | (exponent << 23)**

CONVERSIA ÎN IEEE FP, EX. 8



- **împărțiți a la 4**
  - **soluția:**
    - **vrem exponentul, unde se află?**
      - **MASK = 0x7F800000**

MASK	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

    - **extragem exponent_var = (a & MASK) >> 23 = 124**

a & MASK	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(a & MASK) >> 23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0

    - **dacă exponent_var > 1 atunci exponent_var = exponent_var – 2, altfel a = 0**

exponent_var – 2 = 122	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0

    - **trebuie să actualizăm a**
      - **a = (a & ~MASK) | (exponent_var << 23)**

a & ~MASK	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
exponent_var << 23	0	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(a & ~MASK) | (exponent_var << 23)	0	0	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FP ÎN HEX, EX. 9

d) 0xDEADBEEF = 0b11011110101011011011111011101111

- S = 1
- E = 10111101
- M = 01011011011111011101111
- (-1)^S 1.M * 2^{E-127} = (-1) 1.01011011011111011101111 $2^{189-127}$
  = - 1.01011011011111011101111 2^{62}
  = -6259853398707798000

e) 0x44361000 = 0b01000100001101100001000000000000

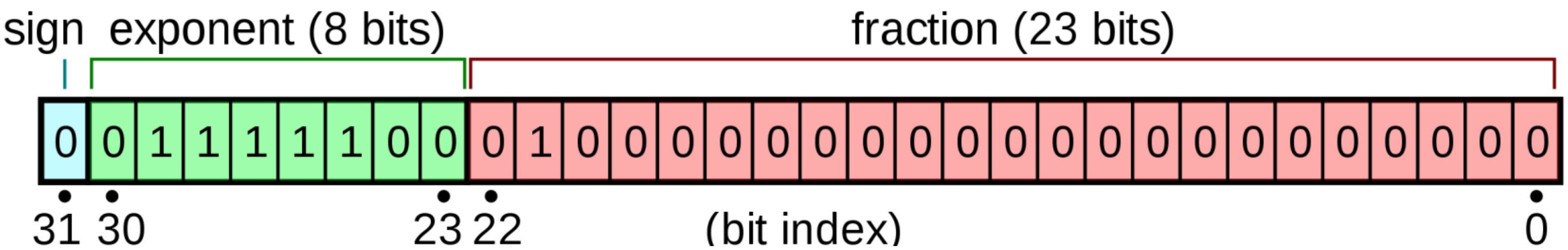
- S = 0
- E = 10001000
- M = 01101100001000000000000
- (-1)^S 1.M * 2^{E-127} = (1) 1. 01101100001000000000000 $2^{136-127}$
  = 1. 01101100001000000000000 2^{9}
  = 728.25

j) 0xC00010FF = 0b11000000000000000001000011111111

- S = 1
- E = 10000000
- M = 00000000001000011111111
- (-1)^S 1.M * 2^{E-127} = (-1) 1.00000000001000011111111 $2^{128-127}$
  = -1.00000000001000011111111 2^{1}
  = -2.001037359237671

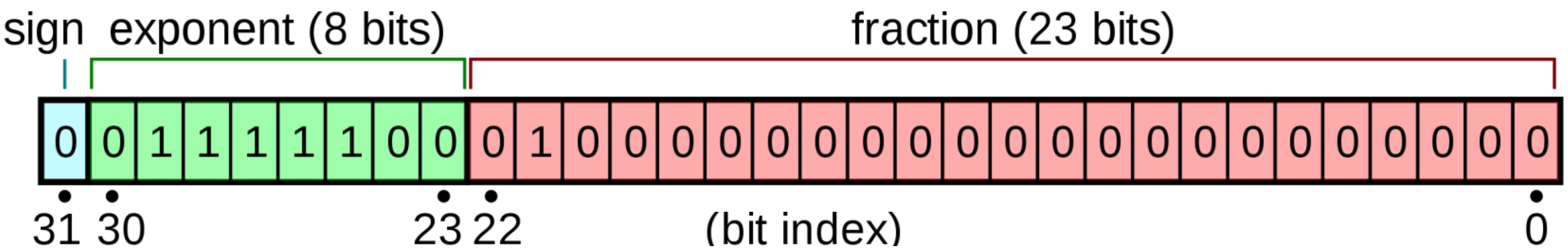
0xFFFFFFFF și 0xFFFF0000 sunt NaN (convenție),

ZERO ÎN IEEE FP, EX. 10



- **setați s = 0, e = 0, f = 0**

ZERO ÎN IEEE FP, EX. 10



- **setați s = 0, e = 0, f = 0**
- **$a = (-1)^0 \times 1.00\ldots00 \times 2^{-127} = 2^{-127} \neq 0$**

ADUNARE IEEE 754 FP, EX. 11

- **0.2 + 0.3**

ADUNARE IEEE 754 FP, EX. 11

- **0.2 + 0.3**

- **primul pas, trecem fiecare număr în format**
  - 0.2 = + 1.10011001100110011001100 x 2^{-3} = 0.199999988079071044
  - 0.3 = + 1.00110011001100110011001 x 2^{-2} = 0.29999998211860657

.

ADUNARE IEEE 754 FP, EX. 11

- **0.2 + 0.3**

- **primul pas, trecem fiecare număr în format**
  - 0.2 = + 1.10011001100110011001100 x 2^{-3} = 0.19999998807907104
  - 0.3 = + 1.00110011001100110011001 x 2^{-2} = 0.29999998211860657

- **al doilea pas, alinierea**
  - 0.2 = + 0.11001100110011001100110|000 x 2^{-2}
  - 0.3 = + 1.00110011001100110011001|000 x 2^{-2}

ADUNARE IEEE 754 FP, EX. 11

- **0.2 + 0.3**

- **primul pas, trecem fiecare număr în format**
  - 0.2 = + 1.10011001100110011001100 x 2^{-3} = 0.19999998807907104
  - 0.3 = + 1.00110011001100110011001 x 2^{-2} = 0.29999998211860657

- **al doilea pas, alinierea**
  - 0.2 = + 0.11001100110011001100110|000 x 2^{-2}
  - 0.3 = + 1.00110011001100110011001|000 x 2^{-2}

- **al treilea pas, adunăm**
  - 0.2 + 0.3 = 1.11111111111111111111111|000 x 2^{-2}

ADUNARE IEEE 754 FP, EX. 11

- **0.2 + 0.3**

- **primul pas, trecem fiecare număr în format**
  - 0.2 = + 1.10011001100110011001100 x 2^{-3} = 0.19999998807907104
  - 0.3 = + 1.00110011001100110011001 x 2^{-2} = 0.29999998211860657

- **al doilea pas, alinierea**
  - 0.2 = + 0.110011001100110011001100110|000 x 2^{-2}
  - 0.3 = + 1.00110011001100110011001|000 x 2^{-2}

- **al treilea pas, adunăm**
  - 0.2 + 0.3 = 1.11111111111111111111111|000 x 2^{-2}

- **al patrulea pas, normalizare (dacă e necesar)**
  - 0.2 + 0.3 = 1.11111111111111111111111|00 x 2^{-2}

ADUNARE IEEE 754 FP, EX. 11

- **0.2 + 0.3**

- **primul pas, trecem fiecare număr în format**
  - 0.2 = + 1.10011001100110011001100 x 2^{-3} = 0.19999998807907104
  - 0.3 = + 1.00110011001100110011001 x 2^{-2} = 0.29999998211860657

- **al doilea pas, alinierea**
  - 0.2 = + 0.11001100110011001100110|000 x 2^{-2}
  - 0.3 = + 1.00110011001100110011001|000 x 2^{-2}

- **al treilea pas, adunăm**
  - 0.2 + 0.3 = 1.11111111111111111111111|000 x 2^{-2}

- **al patrulea pas, normalizare (dacă e necesar)**
  - 0.2 + 0.3 = 1.11111111111111111111111|00 x 2^{-2}

- **pasul final: + 1.11111111111111111111111 x 2^{-2} = 0.4999999701976776**

ÎMPĂRȚIREA RAPIDĂ, EX. 12

- **a / 19**

ÎMPĂRȚIREA RAPIDĂ, EX. 12

- **a / 19**

$$a \times \frac{1}{19} \approx \frac{a \times \frac{2938661835}{2^{32}} + \frac{a - a \times \frac{2938661835}{2^{32}}}{2^{1}}}{2^{4}}$$

$$a \times \frac{1}{19} \approx (a \times 2938661835 \times 2^{-32} + (a - a \times 2938661835 \times 2^{-32}) \times 2^{-1}) \times 2^{-4}$$

$$a \times \frac{1}{19} \approx a \times \frac{7233629131}{137438953472}$$

- **1/19 = 0.05263157894**
- **7233629131 / 137438953472 = 0.05263157895**

ÎMPĂRȚIREA RAPIDĂ, EX. 12

- **a / 19**

$$a \times \frac{1}{19} \approx \frac{a \times \frac{2938661835}{2^{32}} + \frac{a - a \times \frac{2938661835}{2^{32}}}{2^1}}{2^4}$$

$$a \times \frac{1}{19} \approx (a \times 2938661835 \times 2^{-32} + (a - a \times 2938661835 \times 2^{-32}) \times 2^{-1}) \times 2^{-4}$$

$$a \times \frac{1}{19} \approx a \times \frac{7233629131}{137438953472}$$

- **soluția generală**

$$\frac{a}{D} \approx \frac{\frac{aC}{2^X} + \frac{a - \frac{aC}{2^X}}{2^Y}}{2^Z}$$

$$D \approx \frac{2^{X+Y+Z}}{C \times (2^Y - 1) + 2^Z}$$

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția:

.

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
    - soluția: 0.0999999046325683593750

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția: 0.099999904632568359375

- **care este diferența dintre valoarea calculată și 0.1**
  - soluția:

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția: 0.099999904632568359375
- **care este diferența dintre valoarea calculată și 0.1**
  - soluția: 0.1 - 0.099999904632568359375
- **care este eroarea (de timp) după 100 de ore de operare**
  - soluția:

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția: 0.099999904632568359375

- **care este diferența dintre valoarea calculată și 0.1**
  - soluția: 0.1 - 0.099999904632568359375

- **care este eroarea (de timp) după 100 de ore de operare**
  - soluția: 100x60x60x10x(0.1 - 0.099999904632568359375) ≈ 0.34

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția: 0.099999904632568359375
- **care este diferența dintre valoarea calculată și 0.1**
  - soluția: 0.1 - 0.099999904632568359375
- **care este eroarea (de timp) după 100 de ore de operare**
  - soluția: 100x60x60x10x(0.1 - 0.099999904632568359375) ≈ 0.34
- **care este eroarea dacă reprezentăm 0.1 în formatul IEEE 754 FP?**
  - soluția:

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția: 0.099999904632568359375
- **care este diferența dintre valoarea calculată și 0.1**
  - soluția: 0.1 - 0.099999904632568359375
- **care este eroarea (de timp) după 100 de ore de operare**
  - soluția: 100x60x60x10x(0.1 - 0.099999904632568359375) ≈ 0.34
- **care este eroarea dacă reprezentăm 0.1 în formatul IEEE 754 FP?**
  - soluția: 100x60x60x10x(0.1 - 0.099999999403953552) ≈ 0.021

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția: 0.099999904632568359375
- **care este diferența dintre valoarea calculată și 0.1**
  - soluția: 0.1 - 0.099999904632568359375
- **care este eroarea (de timp) după 100 de ore de operare**
  - soluția: 100x60x60x10x(0.1 - 0.099999904632568359375) ≈ 0.34
- **care este eroarea dacă reprezentăm 0.1 în formatul IEEE 754 FP?**
  - soluția: 100x60x60x10x(0.1 - 0.0999999940395355) ≈ 0.021
- **dacă rachetele SCUD pot atinge o viteza MACH 5, care este distanța pe care racheta o poate parcurge în timpul eroare calculat?**
  - soluția:

RACHETE RAPIDE, EX. 19

- **calculați aproximarea binară**
  - soluția: 0.099999904632568359375

- **care este diferența dintre valoarea calculată și 0.1**
  - soluția: 0.1 - 0.099999904632568359375

- **care este eroarea (de timp) după 100 de ore de operare**
  - soluția: 100x60x60x10x(0.1 - 0.099999904632568359375) ≈ 0.34

- **care este eroarea dacă reprezentăm 0.1 în formatul IEEE 754 FP?**
  - soluția: 100x60x60x10x(0.1 - 0.0999999940395355) ≈ 0.021

- **dacă rachetele SCUD pot atinge o viteza MACH 5, care este distanța pe care racheta o poate parcurge în timpul eroare calculat?**
  - soluția: 1715 m/s * 0.34 s ≈ 583 m, 1715 m/s * 0.021 s ≈ 36 m

FAST INVERSE SQUARE ROOT, Q III

```c
float Q_rsqrt( float number )
{
	long i;
	float x2, y;
	const float threehalfs = 1.5F;

	x2 = number * 0.5F;
	y  = number;
	i  = * ( long * ) &y;                       // evil floating point bit level hacking
	i  = 0x5f3759df - ( i >> 1 );               // what the duck?
	y  = * ( float * ) &i;
	y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//	y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
```

https://github.com/id-Software/Quake-III-Arena/blob/dbe4ddb10315479fc00086f08e25d968b4b43c49/code/game/q_math.c

FAST INVERSE SQUARE ROOT, Q III

```
552 float Q_rsqrt( float number )
553 {
554         long i;
555         float x2, y;
556         const float threehalfs = 1.5F;
557 
558         x2 = number * 0.5F;
559         y  = number;
560         i  = * ( long * ) &y;                       // evil floating point bit level hacking
561         i  = 0x5f3759df - ( i >> 1 );               // what the duck?
562         y  = * ( float * ) &i;
563         y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
564 //      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
```

- **prima instrucțiune interesantă e pe linia 560**

- **de exemplu, 0.5 este:**

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

- **aceeași biți dar văzuți de data asta ca un întreg:**
  - $2^{24} + 2^{25} + 2^{26} + 2^{27} + 2^{28} + 2^{29} = 1056964608$ (atât este i)
  - de ce avem nevoie de i?

https://github.com/id-Software/Quake-III-Arena/blob/dbe4ddb10315479fc00086f08e25d968b4b43c49/code/game/q_math.c

FAST INVERSE SQUARE ROOT, Q III

```
552 float Q_rsqrt( float number )
553 {
554         long i;
555         float x2, y;
556         const float threehalfs = 1.5F;
557
558         x2 = number * 0.5F;
559         y  = number;
560         i  = * ( long * ) &y;                                  // evil floating point bit level hacking
561         i  = 0x5f3759df - ( i >> 1 );               // what the duck?
562         y  = * ( float * ) &i;
563         y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
564 //      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
```

- **prima instrucțiune interesantă e pe linia 560**

- **dacă avem în M mantisa și exponentul în E atunci**
  - numărul nostru in FP este 2^{23} x E + M
  - iar valoarea numărului este (1 + M / 2^{23}) x 2^{E-127}
  - observăm că:

$$\begin{aligned}\log_2\left(\left(1+\frac{M}{2^{23}}\right)\times 2^{E-127}\right) &= \log_2\left(1+\frac{M}{2^{23}}\right)+\log_2\left(2^{E-127}\right)\\ &= \log_2\left(1+\frac{M}{2^{23}}\right)+E-127\\ &\approx \frac{M}{2^{23}}+E-127+\mu \quad \text{(am folosit } \log_2(1+x)\approx x, \mu \text{ este o corectie)}\\ &= \frac{1}{2^{23}}(2^{23}\times E+M)+\mu-127\\ &= \frac{1}{2^{23}}(\text{reprezentarea pe biti})+\mu-127\end{aligned}$$

https://github.com/id-Software/Quake-III-Arena/blob/dbe4ddb10315479fc00086f08e25d968b4b43c49/code/game/q_math.c

FAST INVERSE SQUARE ROOT, Q III

```
552 float Q_rsqrt( float number )
553 {
554         long i;
555         float x2, y;
556         const float threehalfs = 1.5F;
557
558         x2 = number * 0.5F;
559         y  = number;
560         i  = * ( long * ) &y;                       // evil floating point bit level hacking
561         i  = 0x5f3759df - ( i >> 1 );               // what the duck?
562         y  = * ( float * ) &i;
563         y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
564 //      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
```

- **prima instrucțiune interesantă e pe linia 560**

- **de ce calculăm log(y)? defapt vrem 1/sqrt(y). dar:**

$$\log_2\left(\frac{1}{\sqrt{y}}\right) = \log_2\left(y^{-1/2}\right) = -\frac{1}{2}\log_2(y) = -(i \gg 1)$$

- **suntem pe linia 561 acum. de unde apare acel număr hexa?**

https://github.com/id-Software/Quake-III-Arena/blob/dbe4ddb10315479fc00086f08e25d968b4b43c49/code/game/q_math.c

FAST INVERSE SQUARE ROOT, Q III

```
552 float Q_rsqrt( float number )
553 {
554         long i;
555         float x2, y;
556         const float threehalfs = 1.5F;
557 
558         x2 = number * 0.5F;
559         y  = number;
560         i  = * ( long * ) &y;                       // evil floating point bit level hacking
561         i  = 0x5f3759df - ( i >> 1 );               // what the duck?
562         y  = * ( float * ) &i;
563         y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
564 //      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
```

- **suntem pe linia 561 acum. de unde apare acel număr hexa?**

$$Y = \frac{1}{\sqrt{y}} \text{ atunci } \log_2(Y) = \log_2(\frac{1}{\sqrt{y}}) \text{ rezulta}$$

$$\frac{1}{2^{23}}(M_Y + 2^{23} \times E_Y) + \mu - 127 = -\frac{1}{2}\left(\frac{1}{2^{23}}(M_y + 2^{23} \times E_y) + \mu - 127\right)$$

$$\begin{aligned} M_Y + 2^{23} \times E_Y =& \frac{3}{2} \times 2^{23}(127 - \mu) - \frac{1}{2}(M_y + 2^{23} \times E_y) \\ =& \text{0x5f3759df} - (i \gg 1) \quad (\mu \text{ este ales pentru a minimiza eroarea}) \end{aligned}$$

- **iar apoi pe linia 562 y este transformat înapoi în FP**

https://github.com/id-Software/Quake-III-Arena/blob/dbe4ddb10315479fc00086f08e25d968b4b43c49/code/game/q_math.c

FAST INVERSE SQUARE ROOT, Q III

```
552 float Q_rsqrt( float number )
553 {
554         long i;
555         float x2, y;
556         const float threehalfs = 1.5F;
557 
558         x2 = number * 0.5F;
559         y  = number;
560         i  = * ( long * ) &y;                                   // evil floating point bit level hacking
561         i  = 0x5f3759df - ( i >> 1 );               // what the duck?
562         y  = * ( float * ) &i;
563         y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
564 //      y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
```

- **pe linia 563 e o iterație din metoda lui Newton**
    - îmbunătățeste rezultatul (e o metoda de optimizare)

https://github.com/id-Software/Quake-III-Arena/blob/dbe4ddb10315479fc00086f08e25d968b4b43c49/code/game/q_math.c

ARHITECTURA SISTEMELOR DE CALCUL SEMINAR 0x04

NOTIȚE SUPORT SEMINAR

Cristian Rusu

ÎNTREBĂRI SCURTE, EX. 1

a) adresa de memorie cea mai mare accesibilă este

ÎNTREBĂRI SCURTE, EX. 1

a) **adresa de memorie cea mai mare accesibilă este 2^{32} = 4GB (4.294.967.296 bytes)**

b) **instrucțiunea este *jne etichetă*, unde *jne* are opcode 0110**

- adresa de memorie cea mai mare accesibilă este

* Sistemele modern sunt pe 64 biți, 2^{64} este 18,446,744,073,709,551,616 (adică 17,179,869,184 GB)

ÎNTREBĂRI SCURTE, EX. 1

a) **adresa de memorie cea mai mare accesibilă este 2^{32} = 4GB (4.294.967.296 bytes)**

b) **instrucțiunea este *jne etichetă*, unde *jne* are opcode 0110**

- adresa de memorie cea mai mare accesibilă este 2^{28}

* Sistemele modern sunt pe 64 biți, 2^{64} este 18,446,744,073,709,551,616 (adică 17,179,869,184 GB)

ÎNTREBĂRI SCURTE, EX. 1

a) **adresa de memorie cea mai mare accesibilă este 2^{32} = 4GB (4.294.967.296 bytes)**

b) **instrucțiunea este *jne etichetă*, unde *jne* are opcode 0110**

- adresa de memorie cea mai mare accesibilă este 2^{28} = 0.25 GB

c) **avem instrucțiunea add R1, R2**

- opcode este 0011

* Sistemele modern sunt pe 64 biți, 2^{64} este 18,446,744,073,709,551,616 (adică 17,179,869,184 GB)

ÎNTREBĂRI SCURTE, EX. 1

a) **adresa de memorie cea mai mare accesibilă este 2^{32} = 4GB (4.294.967.296 bytes)**

b) **instrucțiunea este *jne etichetă*, unde *jne* are opcode 0110**

- adresa de memorie cea mai mare accesibilă este 2^{28} = 0.25 GB

c) **avem instrucțiunea add R1, R2**

- opcode este 0011
- operația suportă putem avea 2^{14} = 16384 regiștri diferiți

d) **avem instrucțiunea add R1, R2, R3**

- opcode este 0100

* Sistemele modern sunt pe 64 biți, 2^{64} este 18,446,744,073,709,551,616 (adică 17,179,869,184 GB)

ÎNTREBĂRI SCURTE, EX. 1

a) **adresa de memorie cea mai mare accesibilă este 2^{32} = 4GB (4.294.967.296 bytes)**

b) **instrucțiunea este *jne etichetă*, unde *jne* are opcode 0110**

- adresa de memorie cea mai mare accesibilă este 2^{28} = 0.25 GB

c) **avem instrucțiunea add R1, R2**

- opcode este 0011
- operația suportă putem avea 2^{14} = 16384 regiștri diferiți

d) **avem instrucțiunea add R1, R2, R3**

- opcode este 0100
- vom avea 9.33 biți pentru fiecare reprezentare a unui registru: două poziții suportă 2^{9} = 512 iar o poziție 2^{10} = 1024

* Sistemele modern sunt pe 64 biți, 2^{64} este 18,446,744,073,709,551,616 (adică 17,179,869,184 GB)

COD ASSEMBLY, EX. 2

- **while loop**

```
sum = 0;
i = 0;
while (i < 10) {
        sum = sum + i;
        i = i + 1;
}
```

```
.global main

main:
        ; initializare
        mov $0, i
        mov $0, sum

        ; while loop
et_loop:
        mov sum, %eax
        mov i, %ecx
        add %ecx, %eax
        mov %eax, sum

        inc i

        cmp $10, i
        jne et_loop

        ; afiseaza suma
        mov sum, %eax
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

COD ASSEMBLY, EX. 2

- **while loop**

```
sum = 0;
i = 0;
while (i < 10) {
        sum = sum + i;
        i = i + 1;
}
```

- **rezultatul este?**

```
.global main

main:
        ; initializare
        mov $0, i
        mov $0, sum

        ; while loop
et_loop:
        mov sum, %eax
        mov i, %ecx
        add %ecx, %eax
        mov %eax, sum

        inc i

        cmp $10, i
        jne et_loop

        ; afiseaza suma
        mov sum, %eax
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

COD ASSEMBLY, EX. 2

- **while loop**

  sum = 0;

  i = 0;

  while (i < 10) {

  sum = sum + i;

  i = i + 1;

  }

- **rezultatul este?**
  - 45

```
.global main

main:
        ; initializare
        mov $0, i
        mov $0, sum

        ; while loop
et_loop:
        mov sum, %eax
        mov i, %ecx
        add %ecx, %eax
        mov %eax, sum

        inc i

        cmp $10, i
        jne et_loop

        ; afiseaza suma
        mov sum, %eax
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

FOR LOOP, EX. 3

```
int sum = 0;
int i = 0;
for (i = 0; i < 10; i++)
    sum += i;
```

- **care este diferența între i++ și ++i**
  - este rezultatul același?
  - e o variantă mai eficientă decât cealaltă?

FOR LOOP, EX. 3

```
int sum = 0;
int i = 0;
for (i = 0; i < 10; i++)
    sum += i;
```

- **care este diferența între i++ și ++i**
    - este rezultatul același?
    - e o variantă mai eficientă decât cealaltă?

```
int i = 1;
i++; // == 1 și i == 2
```

```
int i = 1;
++i; // == 2 și i == 2, deci compilatorul nu are nevoie de o variabilă temporară
```

FOR LOOP, EX. 3

- **implementare mai eficientă**
  - mai puțină memorie
  - mai puține accesări ale memoriei

```
.global main

main:
        ; initializare
        mov $0, i
        mov $0, sum

        ; while loop
et_loop:
        mov sum, %eax
        mov i, %ecx
        add %ecx, %eax
        mov %eax, sum

        inc i

        cmp $10, i
        jne et_loop

        ; afiseaza suma
        mov sum, %eax
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

FOR LOOP, EX. 3

- **implementare mai eficientă**
  - mai putină memorie
  - mai puține accesări ale memoriei
- **totul în regiștri**
  - ca programul acesta să fie identic cu while
    - la sfârșit
    - mov %eax, sum
    - mov $10, i
- **se poate cu mai puține instrucțiuni?**

```
main:
        ; initializare
        xor %eax, %eax
        xor %ecx, %ecx

        ; while loop
et_loop:
        add %ecx, %eax

        inc %ecx

        cmp $10, %ecx
        jne et_loop

        ; afiseaza suma
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

FOR LOOP, EX. 3

- **implementare mai eficientă**
  - mai putină memorie
  - mai puține accesări ale memoriei
- **totul în regiștri**
  - ca programul acesta să fie identic cu while
    - la sfârșit
    - mov %eax, sum
    - mov $10, i
- **se poate cu mai puține instrucțiuni?**
  - da, parcurgere inversă
- **se poate cu și mai puține instrucțiuni?**

```
main:
        ; initializare
        xor %eax, %eax
        mov $9, %ecx

        ; while loop
et_loop:
        add %ecx, %eax

        dec %ecx
        jnz et_loop

        ; afiseaza suma
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

FOR LOOP, EX. 3

- **implementare mai eficientă**
  - mai putină memorie
  - mai puține accesări ale memoriei
- **totul în regiștri**
  - ca programul acesta să fie identic cu while
    - la sfârșit
    - mov %eax, sum
    - mov $10, i
- **se poate cu mai puține instrucțiuni?**
  - da, parcurgere inversă
- **se poate cu și mai puține instrucțiuni?**
  - da: mov $45, %eax

https://godbolt.org/

```
main:
        ; initializare
        xor %eax, %eax
        mov $9, %ecx

        ; while loop
et_loop:
        add %ecx, %eax

        dec %ecx
        jnz et_loop

        ; afiseaza suma
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

FOR LOOP, EX. 3

- **se poate mai eficient?**
  - loop unrolling

```
int sum = 0;
int i = 0;
for (i = 0; i < 10; i++)
    sum += i;
```

.

FOR LOOP, EX. 3

- **se poate mai eficient?**
  - loop unrolling

```
int sum = 0;
int i = 0;
for (i = 0; i < 10; i+=2) {
    sum += i;
    sum += i+1;
}
```

  - de ce am vrea să facem așa ceva?

FOR LOOP, EX. 3

- **se poate mai eficient?**
  - loop unrolling

```
int sum = 0;
int i = 0;
for (i = 0; i < 10; i+=2) {
    sum += i;
    sum += i+1;
}
```

  - de ce am vrea să facem așa ceva?
    - mai puține salturi

FOR LOOP, EX. 3

- **se poate mai eficient?**
  - loop unrolling

```
main:
        ; initializare
        xor %eax, %eax
        mov $10, %ecx

        ; while loop
et_loop:
        add %ecx, %eax
        dec %ecx

        add %ecx, %eax
        dec %ecx

        jnz et_loop

        sub $10, %eax

        ; afiseaza suma
        push %eax
        push $formatPrint
        call printf
        pop %ebx
        pop %ebx

        ; flush
        push $0
        call fflush
        pop %ebx

        ; exit
        mov $1, %eax
        mov $0, %ebx
        int $0x80
```

.

PIPELINE HAZARDS, EX. 4

a) %eax ← %ebx + %ecx

%eax ← %ebx + %edx

b) %ebx ← %ecx + %eax

%eax ← %edx + %eax

c) %eax ← %ebx + %ecx

%edx ← %eax + %edx

d) %eax ← 6

%eax ← 3

%ebx ← %eax + 7

PIPELINE HAZARDS, EX. 4

a) %eax ← %ebx + %ecx

%eax ← %ebx + %edx WAW

b) %ebx ← %ecx + %eax

%eax ← %edx + %eax

c) %eax ← %ebx + %ecx

%edx ← %eax + %edx

d) %eax ← 6

%eax ← 3

%ebx ← %eax + 7

.

PIPELINE HAZARDS, EX. 4

a) %eax ← %ebx + %ecx

%eax ← %ebx + %edx WAW

b) %ebx ← %ecx + %eax

%eax ← %edx + %eax WAR

c) %eax ← %ebx + %ecx

%edx ← %eax + %edx

d) %eax ← 6

%eax ← 3

%ebx ← %eax + 7

PIPELINE HAZARDS, EX. 4

a) %eax ← %ebx + %ecx

%eax ← %ebx + %edx WAW

b) %ebx ← %ecx + %eax

%eax ← %edx + %eax WAR

c) %eax ← %ebx + %ecx

%edx ← %eax + %edx RAW

d) %eax ← 6

%eax ← 3

%ebx ← %eax + 7

PIPELINE HAZARDS, EX. 4

a) %eax ← %ebx + %ecx

%eax ← %ebx + %edx WAW

b) %ebx ← %ecx + %eax

%eax ← %edx + %eax WAR

c) %eax ← %ebx + %ecx

%edx ← %eax + %edx RAW

d) %eax ← 6

%eax ← 3

%ebx ← %eax + 7 WAW, rezultatul poate fi 10 sau 13

BRANCH PREDICTION, EX. 6

- ce face algoritmul?

```
while (na > 0 && nb > 0)
{
    if (*A++ <= *B++) {
        *C++ = *A++; --na;
    } else {
        *C++ = *B++; --nb;
    }
}

while (na > 0) {
    *C++ = *A++; --na;
}

while (nb > 0) {
    *C++ = *B++; --nb;
}
```

.

BRANCH PREDICTION, EX. 6

- **ce face algoritmul?**
  - merge sort

- **câte instrucțiuni de salt avem?**

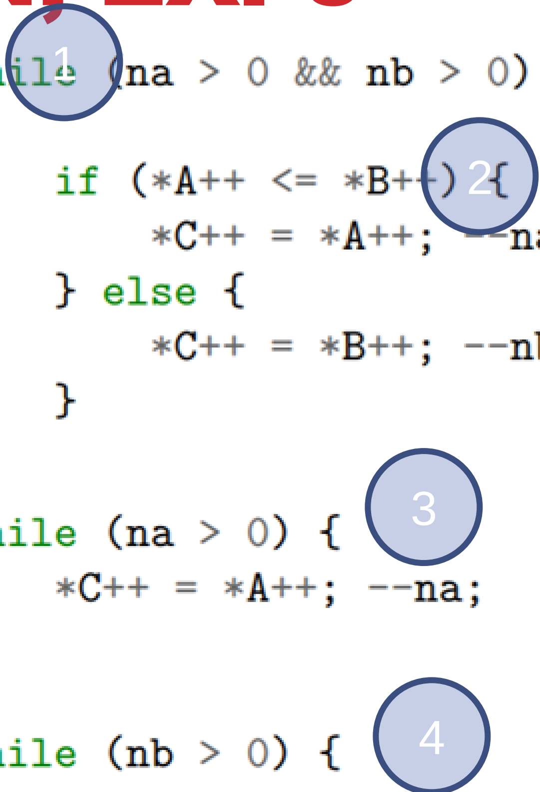
```
while (na > 0 && nb > 0)
{
    if (*A++ <= *B++) {
        *C++ = *A++; --na;
    } else {
        *C++ = *B++; --nb;
    }
}

while (na > 0) {
    *C++ = *A++; --na;
}

while (nb > 0) {
    *C++ = *B++; --nb;
}
```

BRANCH PREDICTION, EX. 6

- **ce face algoritmul?**
  - merge sort

- **câte instrucțiuni de salt avem?**
  - 4

```
while (na > 0 && nb > 0)
{
    if (*A++ <= *B++) {
        *C++ = *A++; --na;
    } else {
        *C++ = *B++; --nb;
    }
}

while (na > 0) {
    *C++ = *A++; --na;
}

while (nb > 0) {
    *C++ = *B++; --nb;
}
```



BRANCH PREDICTION, EX. 6

- **ce face algoritmul?**
  - merge sort
- **câte instrucțiuni de salt avem?**
  - 4
- **predicția pentru fiecare?**

```
while (na > 0 && nb > 0)
{
    if (*A++ <= *B++) {
        *C++ = *A++; --na;
    } else {
        *C++ = *B++; --nb;
    }
}
while (na > 0) {
    *C++ = *A++; --na;
}

while (nb > 0) {
    *C++ = *B++; --nb;
}
```

1
2
3
4

BRANCH PREDICTION, EX. 6

- **ce face algoritmul?**
  - merge sort
- **câte instrucțiuni de salt avem?**
  - 4
- **predicția pentru fiecare?**
  - Salt 1: sare mereu
  - Salt 2: în general, nu știm
  - Salt 3: sare mereu
  - Salt 4: sare mereu

```
while (na > 0 && nb > 0)        1
{
    if (*A++ <= *B++) {         2
        *C++ = *A++; --na;
    } else {
        *C++ = *B++; --nb;
    }
}
while (na > 0) {                3
    *C++ = *A++; --na;
}

while (nb > 0) {                4
    *C++ = *B++; --nb;
}
```

BRANCH PREDICTION, EX. 6

- **ce face algoritmul?**
  - merge sort
- **câte instrucțiuni de salt avem?**
  - 4
- **predicția pentru fiecare?**
  - Salt 1: sare mereu
  - Salt 2: în general, nu știm
  - Salt 3: sare mereu
  - Salt 4: sare mereu
- **cum eliminăm Saltul 2?**
  - int cmp = (*A <= *B)
  - int min = *B ^ ((*B ^ *A) & (-cmp))
    - *C++ = min
    - A += cmp, na -= cmp
    - B += !cmp, nb -= !cmp

```
while (na > 0 && nb > 0)
{
    if (*A++ <= *B++) {
        *C++ = *A++; --na;
    } else {
        *C++ = *B++; --nb;
    }
}
while (na > 0) {
    *C++ = *A++; --na;
}

while (nb > 0) {
    *C++ = *B++; --nb;
}
```

.

TO UPPER, EX. 7

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

https://en.m.wikipedia.org/wiki/File:ASCII-Table-wide.svg

TO UPPER, EX. 7

- un algoritm simplu de toUpper()

```
void toUpper(char *buff, int count) {
    for (int i = 0; i < count; ++i)
    {
        if (buff[i] >= 'a' && buff[i] <= 'z')
            buff[i] -= 32;
    }
}
```

- branchless?

TO UPPER, EX. 7

- un algoritm simplu de toUpper()

```
void toUpper(char *buff, int count) {
    for (int i = 0; i < count; ++i)
    {
        if (buff[i] >= 'a' && buff[i] <= 'z')
            buff[i] -= 32;
    }
}
```

- branchless? mai bine?

```
void toUpper(char *buff, int count) {
    for (int i = 0; i < count; ++i)
    {
        buff[i] = buff[i]*!((buff[i] >= 'a' && buff[i] <= 'z'))
                    + (buff[i] - 32)*(buff[i] >= 'a' && buff[i] <= 'z');
    }
}
```

TO UPPER, EX. 7

- un algoritm simplu de toUpper()

```
void toUpper(char *buff, int count) {
    for (int i = 0; i < count; ++i)
    {
        if (buff[i] >= 'a' && buff[i] <= 'z')
            buff[i] -= 32;
    }
}
```

- branchless?

```
void toUpper(char *buff, int count) {
    for (int i = 0; i < count; ++i)
    {
      buff[i] -= 32*(buff[i] >= 'a' && buff[i] <= 'z');
    }
}
```

https://www.godbolt.org
toUpper() cu SIMD: https://www.youtube.com/watch?v=bVJ-mWWL7cE

COD ASSEMBLY, EX. 7 (VECHI)

```
        .globl  f
f:
        movl    $1, %r8d
        jmp     .LBB0_1
.LBB0_6:
        incl    %r8d
.LBB0_1:
        movl    %r8d, %ecx
        imull   %ecx, %ecx
        movl    $1, %edx
.LBB0_2:
        movl    %edx, %edi
        imull   %edi, %edi
        movl    $1, %esi
        .align  16, 0x90
.LBB0_3:
        movl    %esi, %eax
        imull   %eax, %eax
        addl    %edi, %eax
        cmpl    %ecx, %eax
        je      .LBB0_7
        cmpl    %edx, %esi
        leal    1(%rsi), %eax
        movl    %eax, %esi
        jl      .LBB0_3
        cmpl    %r8d, %edx
        leal    1(%rdx), %eax
        movl    %eax, %edx
        jl      .LBB0_2
        jmp     .LBB0_6
.LBB0_7:
        pushq   %rax
.Ltmp0:
        movl    $.L.str, %edi
        xorl    %eax, %eax
        callq   printf
        movl    $1, %eax
        popq    %rcx
        retq

.L.str:
        .asciz  "%d %d\n"
        .size   .L.str, 7
```

COD ASSEMBLY. EX. 7 (VECHI)

```
        .globl  f
f:
        movl    $1, %r8d
        jmp     .LBB0_1
.LBB0_6:
        incl    %r8d
.LBB0_1:
        movl    %r8d, %ecx
        imull   %ecx, %ecx
        movl    $1, %edx
.LBB0_2:
        movl    %edx, %edi
        imull   %edi, %edi
        movl    $1, %esi
        .align  16, 0x90
.LBB0_3:
        movl    %esi, %eax
        imull   %eax, %eax
        addl    %edi, %eax
        cmpl    %ecx, %eax
        je      .LBB0_7
        cmpl    %edx, %esi
        leal    1(%rsi), %eax
        movl    %eax, %esi
        jl      .LBB0_3
        cmpl    %r8d, %edx
        leal    1(%rdx), %eax
        movl    %eax, %edx
        jl      .LBB0_2
        jmp     .LBB0_6
.LBB0_7:
        pushq   %rax
.Ltmp0:
        movl    $.L.str, %edi
        xorl    %eax, %eax
        callq   printf
        movl    $1, %eax
        popq    %rcx
        retq

.L.str:
        .asciz  "%d %d\n"
        .size   .L.str, 7
```

verifică $x^2 + y^2 = z^2$, cu condiția $x \leq y$

.

ARHITECTURA SISTEMELOR DE CALCUL SEMINAR 0x05

NOTIȚE SUPORT SEMINAR

Cristian Rusu

TIMPI CACHING, EX. 1

a) timpul total de acces memorie este (din curs)

TIMPI CACHING, EX. 1

a) **timpul total de acces memorie este (din curs)**

1 ns + (0.1 x (5 ns + (0.01 x (10 ns + (0.002 x 50 ns)))))

TIMPI CACHING, EX. 1

a) **timpul total de acces memorie este (din curs)**

1 ns + (0.1 x (5 ns + (0.01 x (10 ns + (0.002 x 50 ns)))))

în cazul nostru

1 ns + (A x (5 ns + (0.01 x (10 ns + (0.002 x 50 ns))))) = t_{RAM} / 2

.

TIMPI CACHING, EX. 1

a) **timpul total de acces memorie este (din curs)**

1 ns + (0.1 x (5 ns + (0.01 x (10 ns + (0.002 x 50 ns)))))

în cazul nostru

1 ns + (A x (5 ns + (0.01 x (10 ns + (0.002 x 50 ns))))) = t_{RAM} / 2

b) **1 ns + (0.1 x (A ns + (0.01 x (10 ns + (0.002 x 50 ns))))) = t_{RAM} / 10**

TIMPI CACHING, EX. 1

a) **timpul total de acces memorie este (din curs)**

$1\text{ ns} + (0.1 \times (5\text{ ns} + (0.01 \times (10\text{ ns} + (0.002 \times 50\text{ ns})))))$

în cazul nostru

$1\text{ ns} + (A \times (5\text{ ns} + (0.01 \times (10\text{ ns} + (0.002 \times 50\text{ ns}))))) = t_{RAM} / 2$

b) $\mathbf{1\text{ ns} + (0.1 \times (A\text{ ns} + (0.01 \times (10\text{ ns} + (0.002 \times 50\text{ ns}))))) = t_{RAM} / 10}$

c) $\mathbf{1\text{ ns} + (0.1 \times (5\text{ ns} + (0.01 \times (10\text{ ns} + (A \times 50\text{ ns}))))) = t_{RAM}}$

TIMPI CACHING, EX. 1

a) **timpul total de acces memorie este (din curs)**

$1 \text{ ns} + (0.1 \times (5 \text{ ns} + (0.01 \times (10 \text{ ns} + (0.002 \times 50 \text{ ns})))))$

în cazul nostru

$1 \text{ ns} + (A \times (5 \text{ ns} + (0.01 \times (10 \text{ ns} + (0.002 \times 50 \text{ ns}))))) = t_{RAM} / 2$

b) $\mathbf{1 \text{ ns} + (0.1 \times (A \text{ ns} + (0.01 \times (10 \text{ ns} + (0.002 \times 50 \text{ ns}))))) = t_{RAM} / 10}$

c) $\mathbf{1 \text{ ns} + (0.1 \times (5 \text{ ns} + (0.01 \times (10 \text{ ns} + (A \times 50 \text{ ns}))))) = t_{RAM}}$

d) pentru L1, să trecem de la 1ns la 0.9ns ne costă 100$

pentru L2, să trecem de la 5ns la 4.5ns ne costă 25$

pentru L3, să trecem de la 10ns la 9ns ne costă 10$

$A \text{ ns} + (0.1 \times (B \text{ ns} + (0.01 \times (C \text{ ns} + (0.002 \times 50 \text{ ns}))))) = t_{RAM}/1000$

vrem: minimize $10 A + 2.5 B + C$

rezolvați pentru A, B și C

.

ÎNTREBĂRI SCURTE, EX. 2

a)

ÎNTREBĂRI SCURTE, EX. 2

a) utilizare CPU (eventual sisteme multi-core), media artimetică

b)

ÎNTREBĂRI SCURTE, EX. 2

a) utilizare CPU (eventual sisteme multi-core), media artimetică

b) wall-clock time, media aritmetică

c)

ÎNTREBĂRI SCURTE, EX. 2

a) utilizare CPU (eventual sisteme multi-core), media artimetică

b) wall-clock time, media aritmetică

c) memoria RAM, maximum

d)

ÎNTREBĂRI SCURTE, EX. 2

a) utilizare CPU (eventual sisteme multi-core), media artimetică

b) wall-clock time, media aritmetică

c) memoria RAM, maximum

d) performanță per Watt, media artimetică sau maximum

e)

ÎNTREBĂRI SCURTE, EX. 2

a) utilizare CPU (eventual sisteme multi-core), media artimetică

b) wall-clock time, media aritmetică

c) memoria RAM, maximum

d) performanță per Watt, media artimetică sau maximum

e) wall-clock time, 50/90/99th percentile mediana

f)

ÎNTREBĂRI SCURTE, EX. 2

a) utilizare CPU (eventual sisteme multi-core), media artimetică

b) wall-clock time, media aritmetică

c) memoria RAM, maximum

d) performanță per Watt, media artimetică sau maximum

e) wall-clock time, 50/90/99th percentile mediana

f) wall-clock time speedup, media aritmetică

g)

ÎNTREBĂRI SCURTE, EX. 2

a) utilizare CPU (eventual sisteme multi-core), media artimetică

b) wall-clock time, media aritmetică

c) memoria RAM, maximum

d) performanță per Watt, media artimetică sau maximum

e) wall-clock time, 50/90/99th percentile mediana

f) wall-clock time speedup, media aritmetică

g) minimum = când zgomotul/erorile din sistem sunt minime (*best-case behavior*), maximum = când zgomotul/erorile din sistem sunt maxime (*worst-case behavior*)

h)

ÎNTREBĂRI SCURTE, EX. 2

a) **utilizare CPU (eventual sisteme multi-core), media artimetică**

b) **wall-clock time, media aritmetică**

c) **memoria RAM, maximum**

d) **performanță per Watt, media artimetică sau maximum**

e) **wall-clock time, 50/90/99th percentile mediana**

f) **wall-clock time speedup, media aritmetică**

g) **minimum = când zgomotul/erorile din sistem sunt minime (*best-case behavior*), maximum = când zgomotul/erorile din sistem sunt maxime (*worst-case behavior*)**

h) **dacă măsurăm cât mai bine și exact fiecare componentă, putem optimiza cât mai bine (semnificativ)**

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	
1	9	3	
2	8	2	
3	2	20	
4	10	2	

.

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B
1	9	3	3
2	8	2	4
3	2	20	0.1
4	10	2	5
Media	7.25	6.75	3.025

Concluzia:

.

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B
1	9	3	3
2	8	2	4
3	2	20	0.1
4	10	2	5
Media	7.25	6.75	3.025

Concluzia: Program B este de 3 ori mai rapid decât Program A

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B	B/A
1	9	3	3	
2	8	2	4	
3	2	20	0.1	
4	10	2	5	
Media	7.25	6.75	3.025	

Concluzia: Program B este de 3 ori mai rapid decât Program A

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B	B/A
1	9	3	3	0.33
2	8	2	4	0.25
3	2	20	0.1	10
4	10	2	5	0.2
Media	7.25	6.75	3.025	2.7

Concluzia: Program B este de 3 ori mai rapid decât Program A
Concluzia:

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B	B/A
1	9	3	3	0.33
2	8	2	4	0.25
3	2	20	0.1	10
4	10	2	5	0.2
Media	7.25	6.75	3.025	2.7

Concluzia: Program B este de 3 ori mai rapid decât Program A
Concluzia: Program A este de 2.7 ori mai rapid decât Program B

Care e problema?

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B	B/A
1	9	3	3	0.33
2	8	2	4	0.25
3	2	20	0.1	10
4	10	2	5	0.2
Media	7.25	6.75	3.025	2.7

Concluzia: Program B este de 3 ori mai rapid decât Program A
Concluzia: Program A este de 2.7 ori mai rapid decât Program B

Nu luați media aritmetică a rapoartelor A/B sau B/A

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B	B/A
1	9	3	3	0.33
2	8	2	4	0.25
3	2	20	0.1	10
4	10	2	5	0.2
Media	(a) 7.25	(a) 6.75	(g) 1.57	(g) 0.64

Concluzia: Program B este de 3 ori mai rapid decât Program A
Concluzia: Program A este de 2.7 ori mai rapid decât Program B

Nu luați media aritmetică a rapoartelor A/B sau B/A
Luați media geometrică a rapoartelor A/B sau B/A

- în acest caz, media rapoartelor este raportul medilor

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B	B/A
1	9	3	3	0.33
2	8	2	4	0.25
3	2	20	0.1	10
4	10	2	5	0.2
Media	(a) 7.25	(a) 6.75	(g) 1.57	(g) 0.64

Vrem să comparăm Program A vs. Program B: cine este mai rapid? A sau B?

TIMPI DE RULARE, EX. 3

Test	Program A	Program B	A/B	B/A
1	9	3	3	0.33
2	8	2	4	0.25
3	2	20	0.1	10
4	10	2	5	0.2
Media	(a) 7.25	(a) 6.75	(g) 1.57	(g) 0.64

Vrem să comparăm Program A vs. Program B: cine este mai rapid? A sau B?

- rulăm programele de mai multe ori
- comparăm linie cu linie în tabelul de mai sus
- pentru fiecare linie decidem cine câștigă (A sau B)
- apoi calculăm: care este probabilitatea ca A să fie mai rapid decât B dacă am observat că în n cazuri (din totalul de N) A este mai rapid decât B
- p-value

https://en.wikipedia.org/wiki/P-value

ÎNMULȚIRE COMPLEXĂ, EX. 4

a)

ÎNMULȚIRE COMPLEXĂ, EX. 4

a) $z = (a+bi)x(c+di) = ac - bd + i(ad + bc)$

b)

ÎNMULȚIRE COMPLEXĂ, EX. 4

a) $z = (a+bi)x(c+di) = ac - bd + i(ad + bc)$

b) 2 adunări, 4 înmulțiri

c)

ÎNMULȚIRE COMPLEXĂ, EX. 4

a) $z = (a+bi)x(c+di) = ac - bd + i(ad + bc)$

b) 2 adunări, 4 înmulțiri

c) calculăm $S1 = ac$, $S2 = bd$ și $S3 = (a+b)x(c+d)$

$z = S1-S2 + i(S3-S1-S2)$

ÎNMULȚIRE COMPLEXĂ, EX. 4

a) $z = (a+bi)x(c+di) = ac - bd + i(ad + bc)$

b) 2 adunări, 4 înmulțiri

c) calculăm $S1 = ac$, $S2 = bd$ și $S3 = (a+b)x(c+d)$

$z = S1-S2 + i(S3-S1-S2)$

5 adunări, 3 înmulțiri

d)

ÎNMULȚIRE COMPLEXĂ, EX. 4

a) $z = (a+bi)x(c+di) = ac - bd + i(ad + bc)$

b) 2 adunări, 4 înmulțiri

c) calculăm $S1 = ac$, $S2 = bd$ și $S3 = (a+b)x(c+d)$

$z = S1-S2 + i(S3-S1-S2)$

5 adunări, 3 înmulțiri

d) C1 – costul unei adunări

C2 – costul unei înmulțiri

ÎNMULȚIRE COMPLEXĂ, EX. 4

a) $z = (a+bi)x(c+di) = ac - bd + i(ad + bc)$

b) 2 adunări, 4 înmulțiri

c) calculăm $S1 = ac$, $S2 = bd$ și $S3 = (a+b)x(c+d)$

$z = S1-S2 + i(S3-S1-S2)$

5 adunări, 3 înmulțiri

d) C1 – costul unei adunări

C2 – costul unei înmulțiri

$2C1 + 4C2 > 5C1 + 3C2$

$C2/C1 > 3$

ÎNMULȚIRE MATRICE, EX. 4

e) algoritmul lui Strassen

- vrem să calculăm C = AB (unde A și B sunt matrice)

```
for (i = 0; i < row_length_A; i++)
{
    for (k = 0; k < column_length_B; k++)
    {
        sum = 0;
        for (j = 0; j < column_length_A; j++)
        {
            sum += A[i][j] * B[j][k];
        }
        C[i][k] = sum;
    }
}
```

- pe blocuri:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{1,2} &= \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\
\mathbf{C}_{2,1} &= \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{2,2} &= \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}
\end{aligned}$$

$$\begin{aligned}
\mathbf{M}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{M}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{M}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{M}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{M}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{M}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{M}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{1,2} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{2,1} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{2,2} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}$$

ÎNMULȚIRE MATRICE, EX. 4

e) algoritmul lui Strassen

- vrem să calculăm C = AB (unde A și B sunt matrice)

```
for (i = 0; i < row_length_A; i++)
{
    for (k = 0; k < column_length_B; k++)
    {
        sum = 0;
        for (j = 0; j < column_length_A; j++)
        {
            sum += A[i][j] * B[j][k];
        }
        C[i][k] = sum;
    }
}
```

- pe blocuri:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{1,1} & \mathbf{C}_{1,2} \\ \mathbf{C}_{2,1} & \mathbf{C}_{2,2} \end{bmatrix}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{1,2} &= \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\
\mathbf{C}_{2,1} &= \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{2,2} &= \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}
\end{aligned}$$

$O(n^3)$

$$\begin{aligned}
\mathbf{M}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{M}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{M}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{M}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{M}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{M}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{M}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned}$$

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{1,2} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{2,1} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{2,2} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned}$$

$O(n^{2.8})$

LUCRU CU VECTOR/MATRICE, EX. 5

- produs scalar

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	…	x_9

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	…	y_9

- cum calculăm eficient?

LUCRU CU VECTOR/MATRICE, EX. 5

- **produs scalar**

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	…	x_9

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	…	y_9

- **cum calculăm eficient?**
  - c += x_i y_i
  - instrucțiune Fast Multiply-Add (fma)
  - aceeași operație pe date diferite (SIMD)
  - foarte ușor de paralelizat
    - atenție, rezultatul va fi diferit (adunarea nu mai e asociativă)
    - cu *p* procesoare ne așteptăm să fim de *p* ori mai rapizi
  - exploatează cache la maxim: datele sunt continue în memorie

LUCRU CU VECTOR/MATRICE, EX. 5

- **produs matrice-vector**

$$\begin{pmatrix} 2 & 3 \\ 4 & 5 \\ -1 & 6 \end{pmatrix} \times \begin{bmatrix} 4 \\ 7 \end{bmatrix} = \begin{pmatrix} (2 * 4) + (3 * 7) \\ (4 * 4) + (5 * 7) \\ (-1 * 4) + (6 * 7) \end{pmatrix} = \begin{bmatrix} 29 \\ 51 \\ 38 \end{bmatrix}$$

- **cum calculăm eficient?**

https://mercris.wordpress.com/2012/06/29/linear-algebra-cheat-sheet-for-machine-learning/

LUCRU CU VECTOR/MATRICE, EX. 5

- **produs matrice-vector**

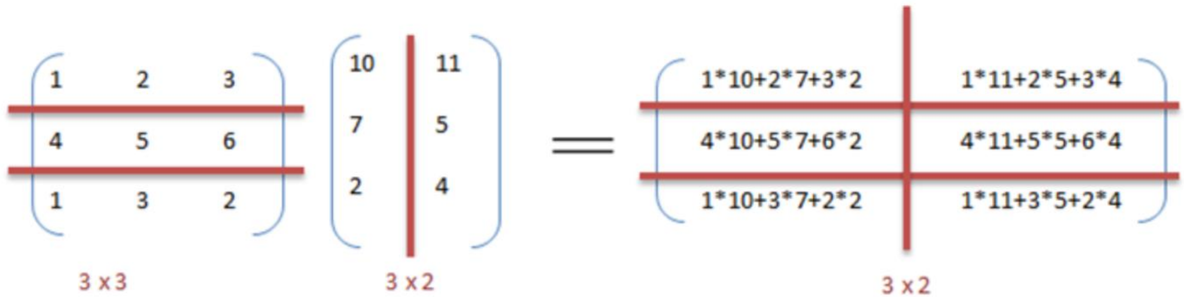
$$\begin{pmatrix} 2 & 3 \\ 4 & 5 \\ -1 & 6 \end{pmatrix} \times \begin{pmatrix} 4 \\ 7 \end{pmatrix} = \begin{pmatrix} (2 * 4) + (3 * 7) \\ (4 * 4) + (5 * 7) \\ (-1 * 4) + (6 * 7) \end{pmatrix} = \begin{pmatrix} 29 \\ 51 \\ 38 \end{pmatrix}$$

- **cum calculăm eficient?**
  - c += x_i y_i
  - *n* produse scalare (se pot realiza în paralel)
  - cum exploatăm eficient cache-ul?
    - dacă putem forța cache-ul să țină vectorul atunci cache miss rate va fi cu siguranță <= 50%

https://mercris.wordpress.com/2012/06/29/linear-algebra-cheat-sheet-for-machine-learning/

LUCRU CU VECTOR/MATRICE, EX. 5

- **produs matrice-matrice**



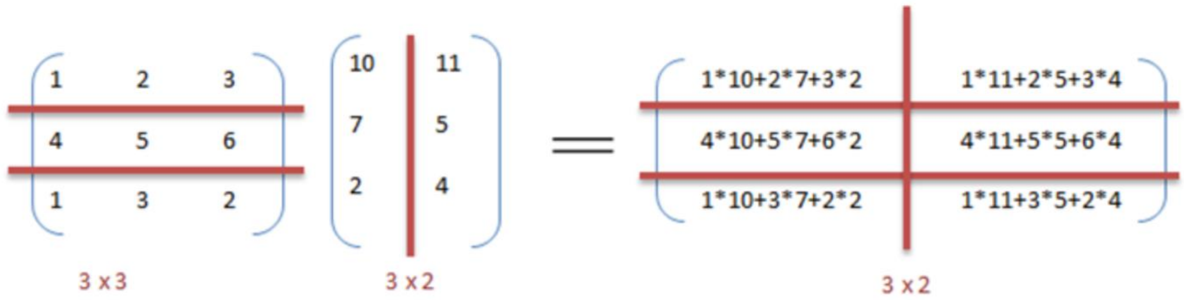
- **cum calculăm eficient?**

```
# varianta A
for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];

# varianta B
for (int j = 0; j < n; ++j)
  for (int i = 0; i < n; ++i)
     for (int k = 0; k < n; ++k)
        C[i][j] += A[i][k] * B[k][j];

# varianta C
for (int k = 0; k < n; ++k)
  for (int i = 0; i < n; ++i)
     for (int j = 0; j < n; ++j)
        C[i][j] += A[i][k] * B[k][j];
```

https://mercris.wordpress.com/2012/06/29/linear-algebra-cheat-sheet-for-machine-learning/

LUCRU CU VECTOR/MATRICE, EX. 5

- **produs matrice-matrice**



- **cum calculăm eficient?**
  - c += x_i y_i
  - n^2 produse scalare (se pot realiza în paralel)
  - dintre cele 3 variate din dreapta, care

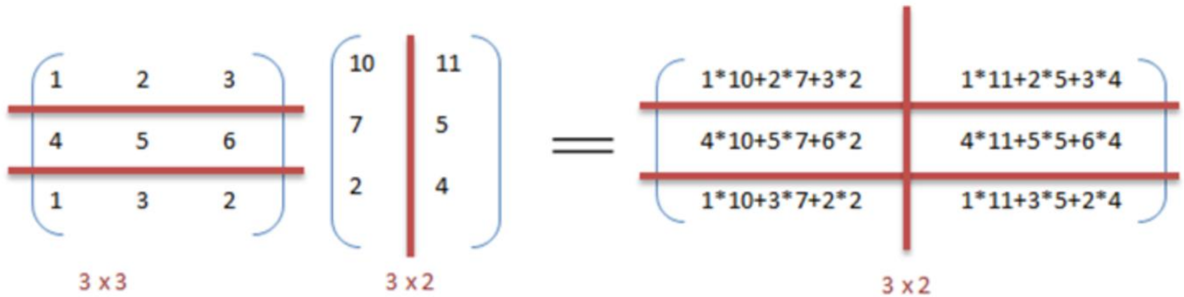
  este mai rapidă în C? (testați și explicați)
  - cum exploatăm eficient cache-ul?

```
# varianta A
for (int i = 0; i < n; ++i)
  for (int j = 0; j < n; ++j)
    for (int k = 0; k < n; ++k)
      C[i][j] += A[i][k] * B[k][j];

# varianta B
for (int j = 0; j < n; ++j)
  for (int i = 0; i < n; ++i)
    for (int k = 0; k < n; ++k)
      C[i][j] += A[i][k] * B[k][j];

# varianta C
for (int k = 0; k < n; ++k)
  for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
      C[i][j] += A[i][k] * B[k][j];
```

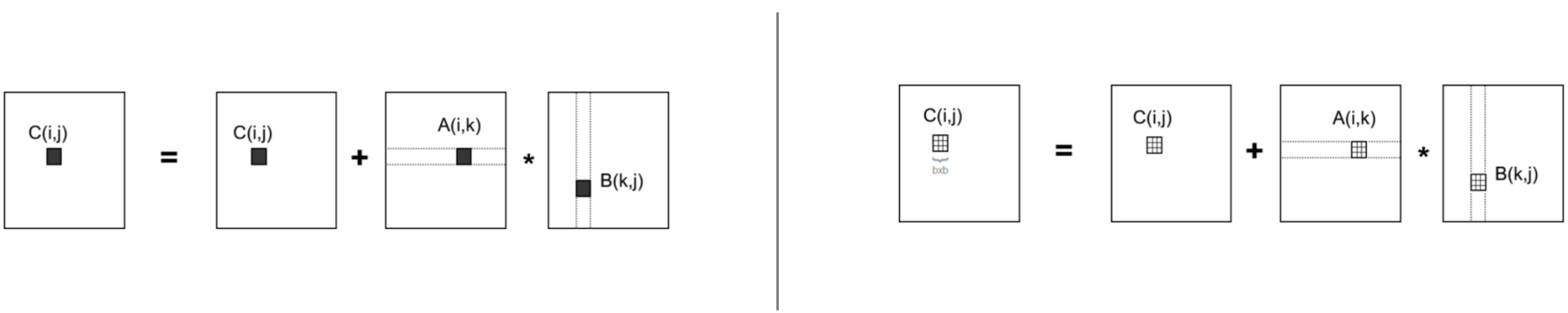
https://mercris.wordpress.com/2012/06/29/linear-algebra-cheat-sheet-for-machine-learning/

LUCRU CU VECTOR/MATRICE, EX. 5

- **produs matrice-matrice**



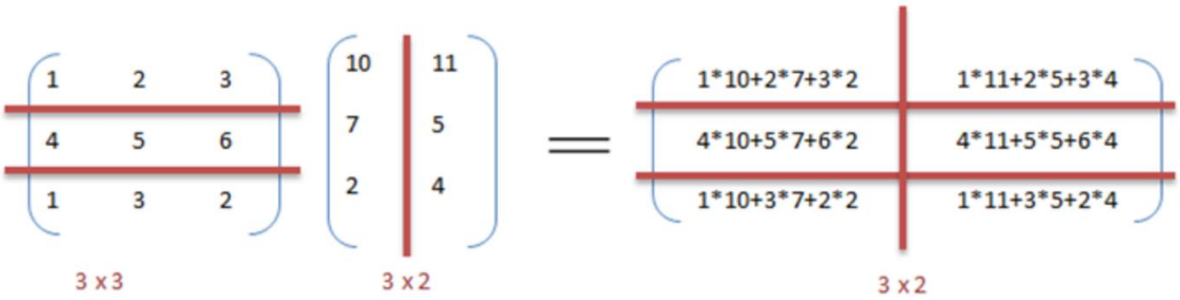
- **cum calculăm eficient?**
  - c += x_i y_i
  - *n^2* produse scalare (se pot realiza în paralel)
  - cum exploatăm eficient cache-ul?
    - calcul pe blocuri, nu pe linii sau coloane



https://en.wikipedia.org/wiki/Communication-avoiding_algorithm

LUCRU CU VECTOR/MATRICE, EX. 5

- **produs matrice-matrice**



- **cum calculăm eficient?**
  - $c \mathrel{+}= x_i \, y_i$
  - n^2 produse scalare (se pot realiza în paralel)
  - cum exploatăm eficient cache-ul?
    - calcul pe blocuri, nu pe linii sau coloane

```
for (ii = 0; ii < SIZE; ii += BLOCK_SIZE)
  for (kk = 0; kk < SIZE; kk += BLOCK_SIZE)
    for (jj = 0; jj < SIZE; jj += BLOCK_SIZE)
      maxi = min(ii + BLOCK_SIZE, SIZE);
      for (i = ii; i < maxi; i++)
        maxk = min(kk + BLOCK_SIZE, SIZE);
        for (k = kk; k < maxk; k++)
          maxj = min(jj + BLOCK_SIZE, SIZE);
          for (j = jj; j < maxj; j++)
            C[i][j] = C[i][j] + A[i][k] * B[k][j];
```

https://en.wikipedia.org/wiki/Communication-avoiding_algorithm

PERFORMANȚA MULTI-CORE, EX. 6

a) **S_{Amdahl} = 1,78 și $S_{gustafson}$ = 4,5**

b) **pentru Amdahl**

dacă un program are timpul de execuție T atunci T = (1-p)T + pT (facem distincția între partea paralelizabilă și cel secvențială), dacă avem s core-uri atunci pT devine p/sT, deci accelerarea (raportul dintre timpul inițial și nou timp cu s core-uri) este S = T/((1-p)T + p/s T) = 1/ (1-p + p/s)

pentru Gustafson

sistemul este capabil să execute E = (1-p)E + pE iar partea care se poate îmbunătății este doar pE, care devine δpE, deci execuția este îmbunătățită ((1-p)E + δpE)/E = 1 – p + δp

c) **S_{Amdahl} = 1 și $S_{gustafson}$ = 1 (nicio îmbunătățire)**

d) **S_{Amdahl} = 1 și $S_{gustafson}$ = 1 (nicio îmbunătățire)**

e) **L_{Amdahl} = 1/(1-p), S_{Amdahl} < 1/(1-p)**

f) **$L_{gustafson} = \infty$**

g) **verificați explicațile de la punct b) și țineți cont de rezultatele la limitele pentru p, s și δ**

PERFORMANȚA CICLII, EX. 7

a) 2.1

b) 1.7

c) înainte de modificare N x 2.1 / f, după 0.8 x N x 1.7 / f

d) 1.85

PERFORMANȚA SISTEME, EX. 8

a) p = 80/145, p = 40/145 și p = 25/145 pentru sistemul X

p = 50/140, p = 50/140 și p = 40/140 pentru sistemul Y

b) 2.24 pentru sistemul X și 2.86 pentru sistemul Y

c) cpu time pe sistemul X = 145 x 2.24 / f

cpu time pe sistemul Y = 140 x 2.86 / (1.2 x f)

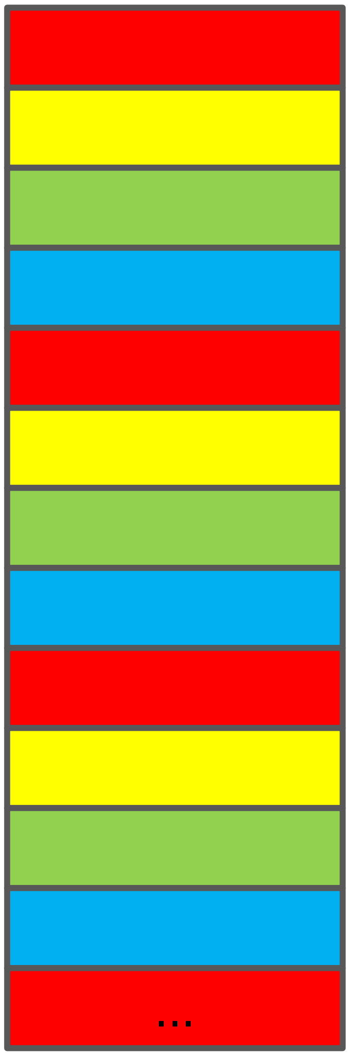
accelerarea este raportul valorilor: Y / X ≈ 1.03 (sistemul X este cu 3% mai rapid decât Y)

CACHE, EX. 9

a) $2^{32} / 2^{5} = 2^{27}$, $(2^{4} \times 2^{10}) / 2^{5} = 2^{9} = 512$ **blocuri**

b) **un exemplu în care cache are doar 4 blocuri, cea mai simplă idee: un bloc din memoria principală dacă e în cache poate să fie doar într-o poziție din cache cu aceeași culoare**

memoria principală



cache

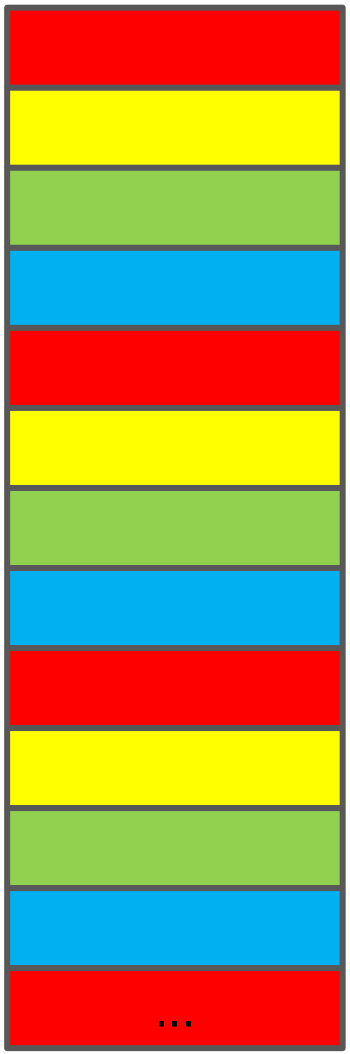
1) trebuie să știm ce culoare suntem
2) după ce știm culoare, trebuie să știm care block din memoria principală este cel corect (din toate cele roșii)
3) trebuie să știm unde în bloc este byte-ul pe care îl vrem

CACHE, EX. 9

a) $2^{32} / 2^5 = 2^{27}$, $(2^4 \times 2^{10}) / 2^5 = 2^9 = 512$ blocuri

b) un exemplu în care cache are doar 4 blocuri, cea mai simplă idee: un bloc din memoria principală dacă e în cache poate să fie doar într-o poziție din cache cu aceeași culoare

memoria principală



cache

1) trebuie să știm ce culoare suntem

2) după ce știm culoare, trebuie să știm care block din memoria principală este cel corect (din toate cele roșii)

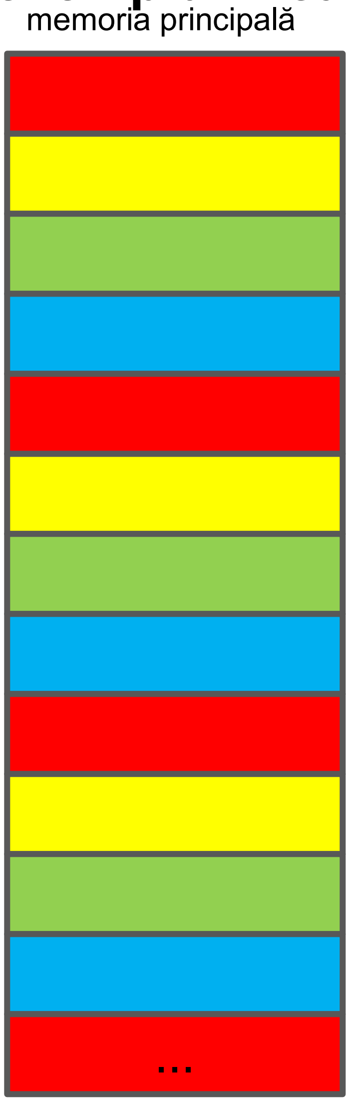
3) trebuie să știm unde în bloc este byte-ul pe care îl vrem

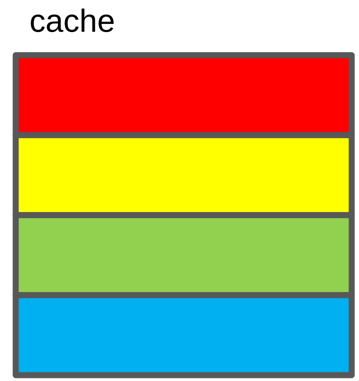
(2)	(1)	(3)

CACHE, EX. 9

a) $2^{32} / 2^5 = 2^{27}$, $(2^4 \times 2^{10}) / 2^5 = 2^9 = 512$ blocuri

b) un exemplu în care cache are doar 4 blocuri, cea mai simplă idee: un bloc din memoria principală dacă e în cache poate să fie doar într-o poziție din cache cu aceeași culoare





1) trebuie să știm ce culoare suntem

2) după ce știm culoare, trebuie să știm care block din memoria principală este cel corect (din toate cele roșii)

3) trebuie să știm unde în bloc este byte-ul pe care îl vrem

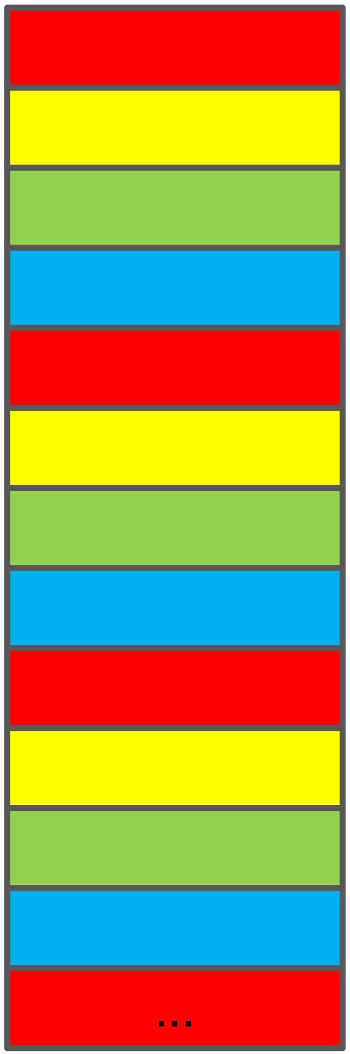
TAG	INDEX	OFFSET

CACHE, EX. 9

a) $2^{32} / 2^5 = 2^{27}$, $(2^4 \times 2^{10}) / 2^5 = 2^9 = 512$ blocuri

b) un exemplu în care cache are doar 4 blocuri, cea mai simplă idee: un bloc din memoria principală dacă e în cache poate să fie doar într-o poziție din cache cu aceeași culoare

memoria principală



cache

1) trebuie să știm ce culoare suntem

2) după ce știm culoare, trebuie să știm care block din memoria principală este cel corect (din toate cele roșii)

3) trebuie să știm unde în bloc este byte-ul pe care îl vrem

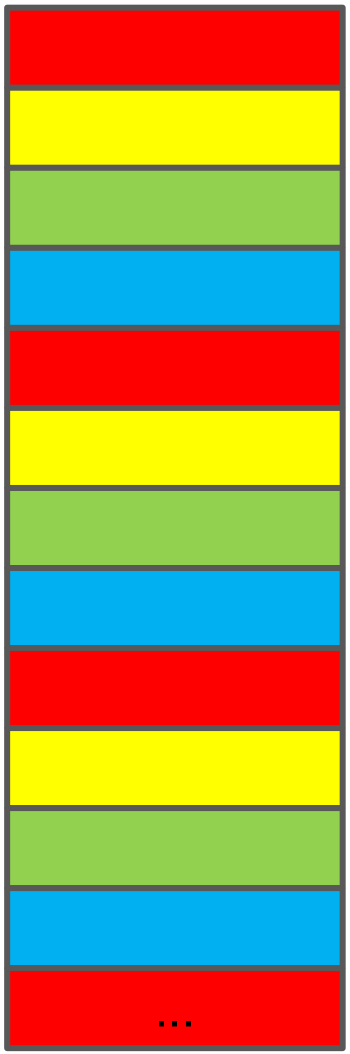
18 biți (ce rămâne pentru a identificare care bloc de aceeași culoare e cel corect)	9 biți (pentru că sunt 2^9 = 512 blocuri/culori în cache)	5 biți (pentru că sunt 2^5 = 32 bytes posibili în bloc)

CACHE, EX. 9

a) $2^{32} / 2^5 = 2^{27}$, $(2^4 \times 2^{10}) / 2^5 = 2^9 = 512$ blocuri

b) un exemplu în care cache are doar 4 blocuri, cea mai simplă idee: un bloc din memoria principală dacă e în cache poate să fie doar într-o poziție din cache cu aceeași culoare

memoria principală



tehnica aceasta de 1 la 1 se numește *direct mapping*
dacă un bloc din memoria principală poate să fie în mai multe blocuri din cache (nu doar unul singur) atunci tehnica se numește *N-set associative mapping* (unde N este numărul de blocuri din cache în care un bloc din memorie poate fi copiat (este fie acolo, fie nu e in cache)

această nouă tehnică oferă mai multa flexibilitate (1 la 1 este prea limitat)

1) trebuie să știm ce culoare suntem

2) după ce știm culoare, trebuie să știm care block din memoria principală este cel corect (din toate cele roșii)

3) trebuie să știm unde în bloc este byte-ul pe care îl vrem

18 biți (ce rămâne pentru a identificare care bloc de aceeași culoare e cel corect)	9 biți (pentru că sunt 2^9 = 512 blocuri/culori în cache)	5 biți (pentru că sunt 2^5 = 32 bytes posibili în bloc)