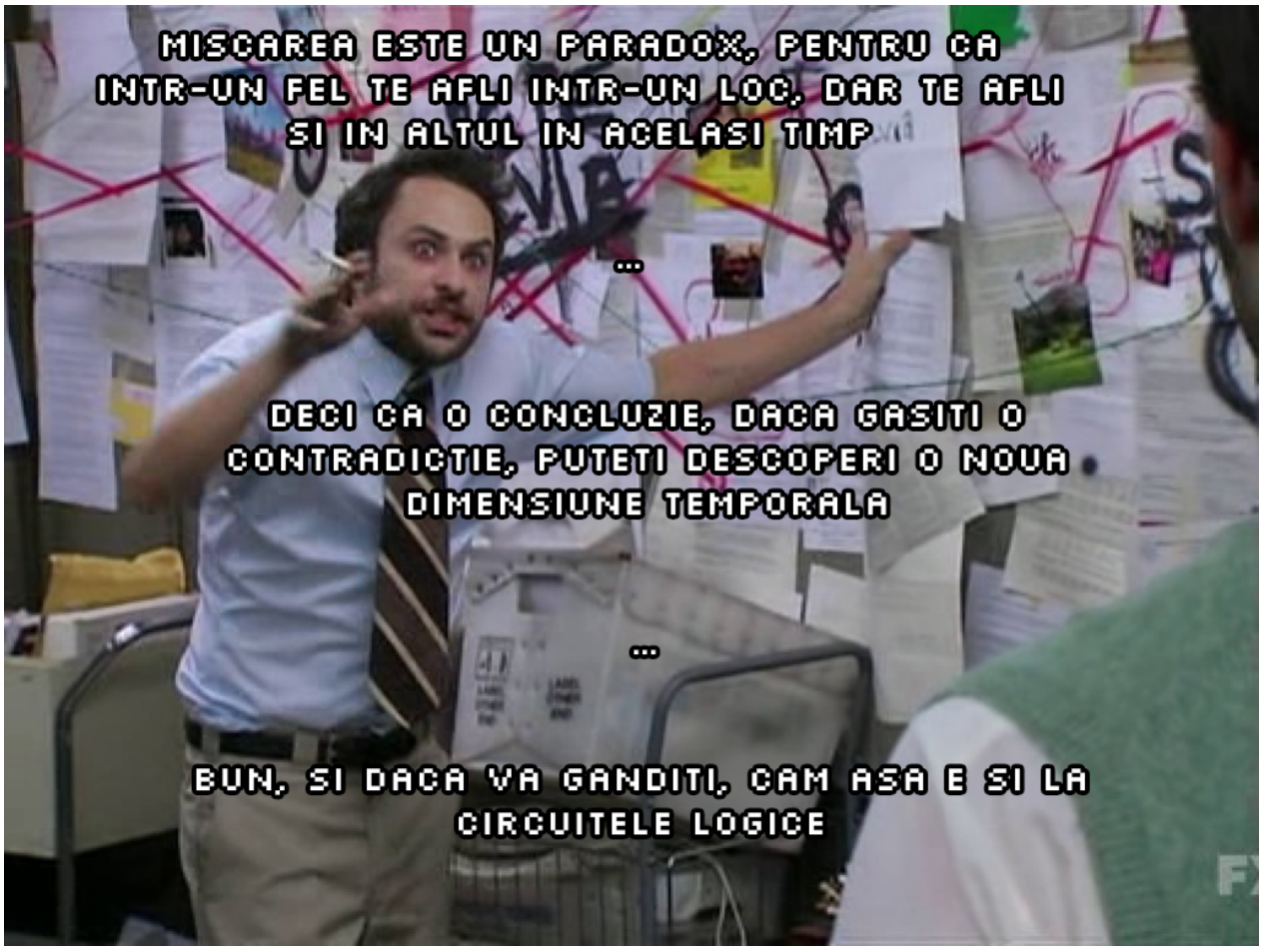
Tutoriat 4

Stan Bianca-Mihaela, Stăncioiu Silviu

February 6, 2021



Contents

1 Algebra Booleana

1.1 Operatii si proprietati

Vom incepe cu algebra booleana cu care probabil sunteti deja familiari de la Logica.
Ca un mic twist, la ASC avem notatiile:

- + <=> $\vee$ (SAU)
- . <=> $\wedge$ (SI)
- $\overline{\ }$ <=> $\neg$ (NEGATIA)

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

$\overline{0} = 1$

$\overline{1} = 0$

Dupa cum stim, pe o algebra booleana B_2 (aka unde avem doar multimea de valori 0,1) avem anumite axiome:

1. ASOCIATIVITATEA
   (x+y)+z=x+(y+z)
   $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
2. COMUTATIVITATEA
   x+y=y+x
   $x \cdot y = y \cdot x$
3. ABSORBTIA
   $x + (x \cdot y) = x \cdot (1 + y) = x$ (fiindca 1 + y = 1 sau "ceva" va da mereu 1)
   $x \cdot (x + y) = x$ (daca x e 0 => rezultatul e 0, daca x e 1 => rezultatul e 1 indiferent de y)
4. DISTRIBUTIVITATEA
   $x \cdot (y + z) = x \cdot y + x \cdot z$
   $x + (y \cdot z) = (x + y) \cdot (x + z)$
5. MARGINIREA
   0+x=x
   $0 \cdot x = 0$

1+x=1

$1 \cdot x = x$

6. COMPLEMENTAREA

$x + \overline{x} = 1$

$x \cdot \overline{x} = 0$

Pe langa acestea, mai definim si cateva operatii derivate:

- IMPLICATIA

  $x \to y = \overline{x} + y$

- DIFERENTA

  $x - y = x \cdot \overline{y}$

- ECHIVALENTA

  $x \leftrightarrow y = (x \to y) \cdot (y \to x) = (\overline{x} + y) \cdot (\overline{y} + x)$

- XOR

  $x \oplus y = x \cdot \overline{y} + \overline{x} \cdot y$

- NXOR

  $x * y = \overline{x \oplus y}$ (am notat cu $*$ pentru ca NXOR nu are un semn propriu-zis din cate stiu eu)

- NAND

  $x * y = \overline{x \cdot y}$ (am notat cu $*$ pentru ca NAND nu are un semn propriu-zis din cate stiu eu)

- NOR

  $x * y = \overline{x + y}$ (am notat cu $*$ pentru ca NOR nu are un semn propriu-zis din cate stiu eu)

Aici avem un tabel de adevar cu cele mai importante operatii:

x	y	$x \cdot y$	x NAND y	$x + y$	x NOR y	$x - y$	$x \to y$	$x \oplus y$	x NXOR y
0	0	0	1	0	1	0	1	0	1
0	1	0	1	1	0	0	1	1	0
1	0	0	1	1	0	1	0	1	0
1	1	1	0	1	0	0	1	0	1

Si cateva proprietati:

1. IDEMPOTENTA

   x+x=x

   $x \cdot x = x$

2. LEGEA DUBLEI NEGATII

   $\overline{\overline{x}} = x$

3. LEGILE LUI MORGAN

   $\overline{x + y} = \overline{x} \cdot \overline{y}$

   $\overline{x \cdot y} = \overline{x} + \overline{y}$

   $\overline{\overline{x} + \overline{y}} = x \cdot y$

   $\overline{\overline{x} \cdot \overline{y}} = x + y$

4. ABSORBTIA BOOLEANA

   $x + \overline{x} \cdot y = x + y$

   $x \cdot (\overline{x} + y) = x \cdot y$

5. UNICITATEA COMPLEMENTULUI

   x+y=1 si $x \cdot y = 0 => y = \overline{x}$

   x+y=0 => $x = y = 0$

   $x \cdot y = 1 => x = y = 1$

1.2 Functiile booleene

O functie booleana este o funcie de forma:
$f : B_2^n -> B_2^k <=> f : \{0,1\}^n -> \{0,1\}^k$
Pentru aceste functii putem defini FND si FNC cu care probabil cunteti familiari de la logica.

- FND (FUNCTIA NORMALA DISJUNCTIVA) aka unde da functia 1
  $(.) + (.) + ... + (.) = 1$
- FNC (FUNCTIA NORMALA CONJUNCTIVA) aka unde da functia 0
  $(+) \cdot (+) \cdot ... \cdot (+) = 0$

Exemplul 1: [RESTANTA SEPTEMBRIE 2020]
Fie $f : B_2^3 -> B_2^2, f(x,y,z) = (f_1(x,y,z), f_2(x,y,z))$, unde:

- $f_1(x,y,z) = (x \oplus y)(y \oplus z)$
- $f_2(x,y,z) = 1$ d.d. secventa x, y, z este crescatoare (adica $x \leq y \leq z$).

Construiti tabelul de valori al lui f si scrieti f_1 si f_2 in FND si FNC.

index	x	y	z	$x \oplus y$	$y \oplus z$	$f_1(x,y,z)$	$f_2(x,y,z)$
(0)	0	0	0	0	0	0	1
(1)	0	0	1	0	1	0	1
(2)	0	1	0	1	1	1	0
(3)	0	1	1	1	0	0	1
(4)	1	0	0	1	0	0	0
(5)	1	0	1	1	1	1	0
(6)	1	1	0	0	1	0	0
(7)	1	1	1	0	0	0	1

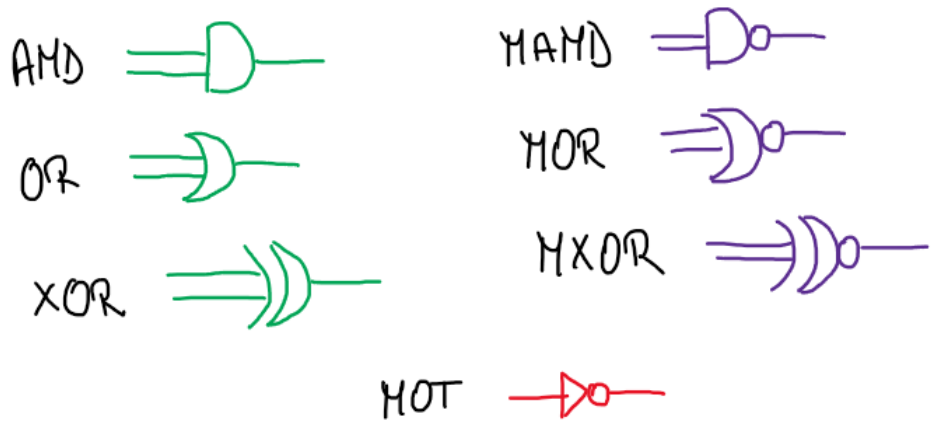
FND:

- $f_1 = \overline{x} \cdot y \cdot \overline{z} + x \cdot \overline{y} \cdot z$ (2)+(5)
- $f_2 = \overline{x} \cdot \overline{y} \cdot \overline{z} + \overline{x} \cdot \overline{y} \cdot z + \overline{x} \cdot y \cdot z + x \cdot y \cdot z$ (0)+(1)+(3)+(7)

FNC:

- $f_1 = (x+y+z) \cdot (x+y+\overline{z}) \cdot (x+\overline{y}+\overline{z}) \cdot (\overline{x}+y+z) \cdot (x+y+\overline{z}) \cdot (x+y+z)$ (0) . (1) . (3) . (4) . (6) . (7)
- $f_2 = (x+\overline{y}+z) \cdot (\overline{x}+y+z) \cdot (\overline{x}+y+\overline{z}) \cdot (\overline{x}+\overline{y}+z)$ (2) . (4) . (5) . (6)

2 Porti logice



IMPORTANT!
Orice circuit se poate implementa folosing doar porti NOR, respectiv NAND. Cum? Stim ca orice circuit se poate implementa doar cu porti AND, OR si NOT => daca aratam ca putem sa implementam aceste 3 porti doar cu porti NOR, respectiv NAND, am demonstrat ca orice circuit se poate impementa doar cu porti NOR, respectiv NAND.

Pentru NOR:

- NEGATIA:

$$\overline{x} = x \; NOR \; x$$

- AND:

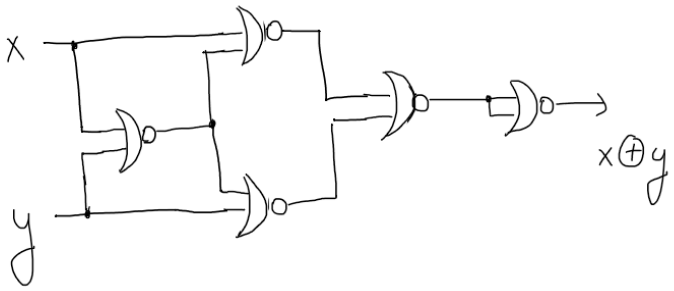
$$x \cdot y = \overline{\overline{x \cdot y}} = \overline{\overline{x} + \overline{y}} = \overline{\overline{x} \; NOR \; \overline{y}} = (xNORx)NOR(yNORy)$$

- OR:

$$x \quad + \quad y \qquad = \qquad \overline{x \; NOR \; y} \qquad = \qquad (x \; NOR \; y) \; NOR \; (x \; NOR \; y) \quad (1)$$

- XOR: Prin anumite "optimizari" in calcul putem obtine o formula cu doar 5 NOR-uri (vezi Exemplul 4 pentru demonstratie):

$$x \oplus y = [(x \; NOR \; (x \; NOR \; y)) \; NOR \; (y \; NOR \; (x \; NOR \; y))] \\ NOR \; [(x \; NOR \; (x \; NOR \; y)) \; NOR \; (y \; NOR \; (x \; NOR \; y))] \quad (2)$$



Pentru NAND:

- NEGATIA:

$$\overline{x} = \overline{x \cdot x} = x \; NAND \; x$$

- AND:

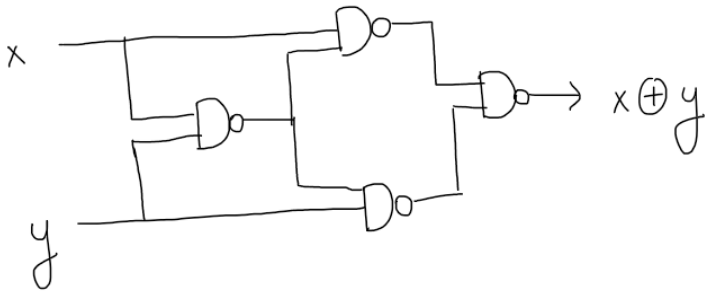
$$x \cdot y = \overline{\overline{x \cdot y}} = \overline{x \; NAND \;} = (x \; NAND \; y) \; NAND \; (x \; NAND \; y)$$

- OR:

$$x + y = \overline{\overline{x + y}} = \overline{\overline{x} \cdot \overline{y}} = \overline{x} \; NAND \; \overline{y} = (x \; NAND \; x) \; NAND \; (y \; NAND \; y)$$

- XOR: Prin anumite "optimizari" in calcul, putem sa ajungem la o formula cu doar 4 NAND-uri (vezi Exemplul 5 pentru demonstratie):

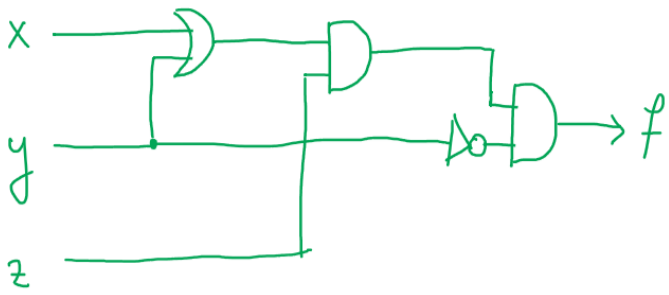
$$x \oplus y = ((x \; NAND \; y) \; NAND \; x) \; NAND \; ((x \; NAND \; y) \; NANDy)$$



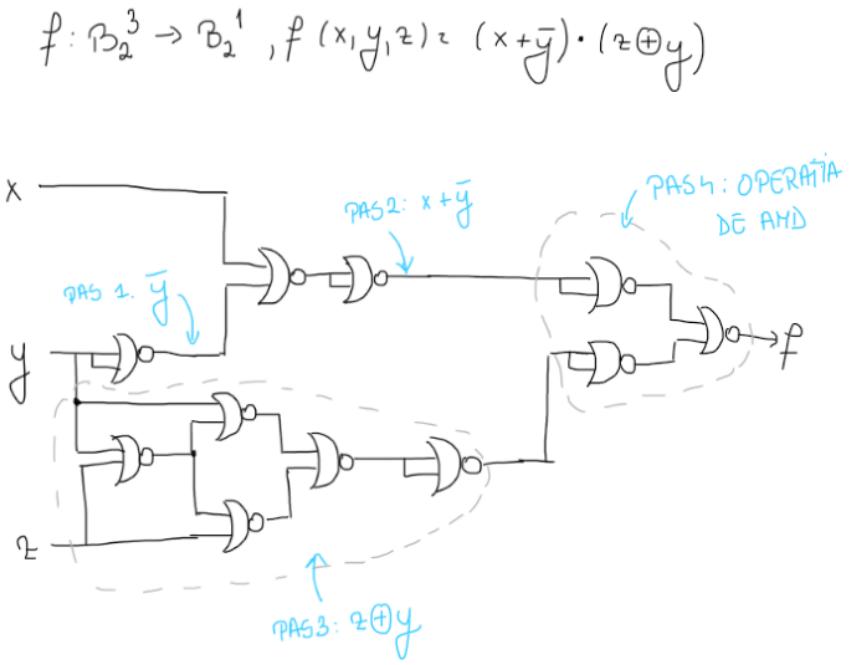
Ca sa intelegem mai bine cum se lucreaza cu aceste porti logice, vom face cateva exemple:

Exemplul 1: Reprezentati prin porti logice functia:

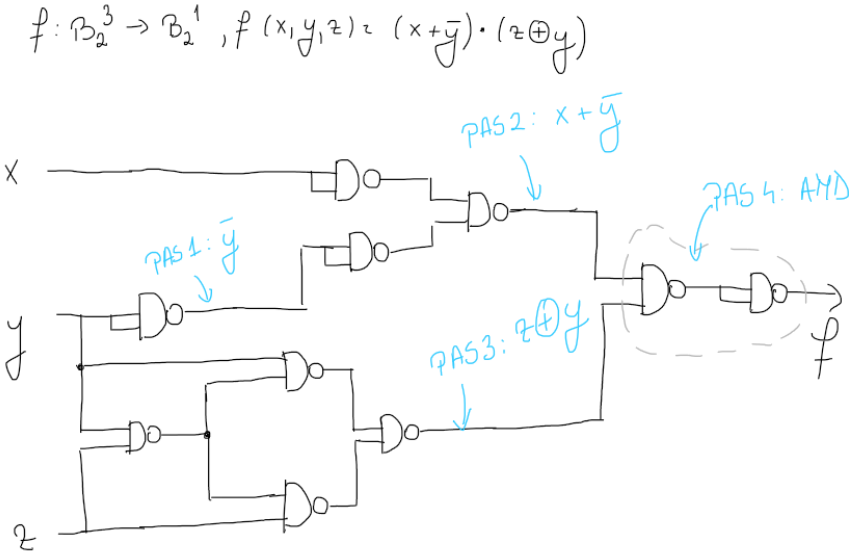
$$f : B_2^3 \to B_2$$
$$f(x, y, z) = (x + y) \cdot z \cdot \overline{y}$$



Exemplul 2: Implementati functia $f : B_2^3 -> B_2^1$, $f(x) = (x + \overline{y}) \cdot (z \oplus y)$ doar cu porti NOR si apoi doar cu porti NAND.
Varianta cu NOR:



Varianta cu NAND:



Exemplul 3 [RESTANTA SEPTEMBRIE 2020]
Implementati poarta NXOR printr-un circuit care contine doar porti NOR.
Prima metoda care ne vine in minte este sa expimam x NXOR y $= \overline{x} \cdot y + x \cdot \overline{y}$ si sa inlocuim operatiile de NEGATIE, AND si OR asa cum stim din formulele pentru NOR.
Aceasta rezolvare este corecta, dar rezulta in:

- 3 porti NOR de la NEGATIE

- 3+3 porti NOR de la AND
- 3 porti NOR de la OR

=> in total vom avea 12 porti NOR, ceea ce nu e ideal.
Dragulici nu specifica ca numarul de porti logice sa fie minim, dar cu siguranta nu se supara daca rezolvati cu numar minim. Si o sa va ajute si pe voi sa nu va incurcati, pentru ca a desena 12 porti doar pentru un XNOR nu e usor.
Asa ca vom incerca sa ajungem la numarul minim de porti cu care se poate reprezenta NXOR.
Keep in mind, la examen daca vreti sa folositi varianta "optimizata" cu 5 porti, trebuie sa aratati cum ati ajuns acolo.

$x \text{ NXOR } y = \overline{x \oplus y}$

$x \text{ NOR } y = \overline{x + y}$

Stim că:

$x \oplus y = x \cdot \bar{y} + y \cdot \bar{x}$

$\Downarrow$

$\overline{x \oplus y} = \overline{x \cdot \bar{y} + y \cdot \bar{x}}$

Stim că $x \cdot \bar{x} = 0$ și $y \cdot \bar{y} = 0$

=> pot să adaug $x \cdot \bar{x}$ și $y \cdot \bar{y}$

$\overline{x \oplus y} = \overline{x \cdot \bar{y} + y \cdot \bar{x} + x \cdot \bar{x} + y \cdot \bar{y}} =$

$= \overline{\bar{x}(x+y) + \bar{y}(x+y)}$

A B

$A = \bar{x}(x+y)$

$A = \overline{\bar{A}} = \overline{\overline{\bar{x} \cdot (x+y)}} = \overline{\bar{x}} + \overline{(x+y)} =$ (de Morgan)

$= \overline{x + (\overline{x+y})} = x \text{ NOR } (\overline{x+y}) =$

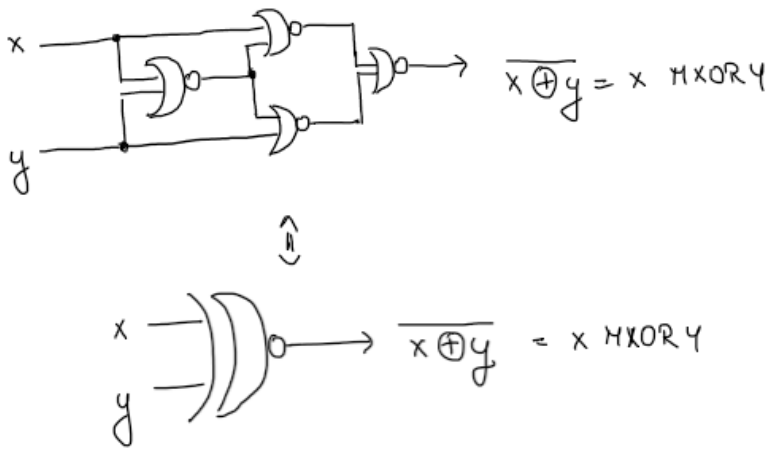
$\Rightarrow \boxed{A = x \text{ NOR } (x \text{ NOR } y)}$

Analog $B = y \text{ NOR } (x \text{ NOR } y)$

$\overline{x \oplus y} = \overline{[x \text{ NOR } (x \text{ NOR } y)] + [y \text{ NOR } (x \text{ NOR } y)]} =$

$= (x \text{ NOR } (x \text{ NOR } y)) \text{ NOR } (y \text{ NOR } (x \text{ NOR } y))$

Iar diagrama va fi:


x
y
$\overline{x \oplus y}$ = x NXOR y
x
y
$\overline{x \oplus y}$ = x NXOR y

Exemplul 4 Implementati poarta XOR printr-un circuit care contine doar porti NOR.

$x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$

Stim ca: $x \cdot \bar{x} = 0$

$y \cdot \bar{y} = 0$

2) $x \oplus y = \bar{x} \cdot y + x \cdot \bar{y} + x \cdot \bar{x} + y \cdot \bar{y} =$

$= \bar{x} \cdot (x+y) + \bar{y} \cdot (x+y) =$

$A = \bar{\bar{A}}$ $= \overline{\overline{\bar{x} \cdot (x+y)}} + \overline{\overline{\bar{y} \cdot (x+y)}} =$

de Morgan $= \overline{\overline{(\bar{\bar{x}} + (x+y))}} + \overline{\overline{(\bar{\bar{y}} + (x+y))}} =$

$A = \bar{\bar{A}}$ $= \overline{\overline{[x + (\overline{x+y})]}} + \overline{\overline{[y + (\overline{x+y})]}}$

B C

$A = \bar{\bar{A}}$

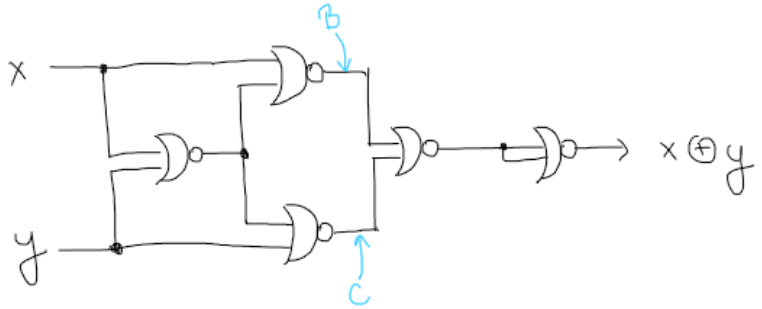
$x \oplus y = B + C = \overline{\overline{B + C}} = \overline{B \text{ NOR } C} =$

$= (B \text{ NOR } C) \text{ NOR } (B \text{ NOR } C)$

unde $B = \overline{x + (\overline{x+y})} = x \text{ NOR } (x \text{ NOR } y)$

$C = \overline{y + (\overline{x+y})} = y \text{ NOR } (x \text{ NOR } y)$

Iar diagrama va fi:



Exemplul 5 Implementati poarta XOR printr-un circuit cu numar minim de porti si doar cu porti NAND.

$$x \oplus y = x \cdot \bar{y} + \bar{x} \cdot y = \overline{\overline{x \cdot \bar{y} + \bar{x} \cdot y}} =$$

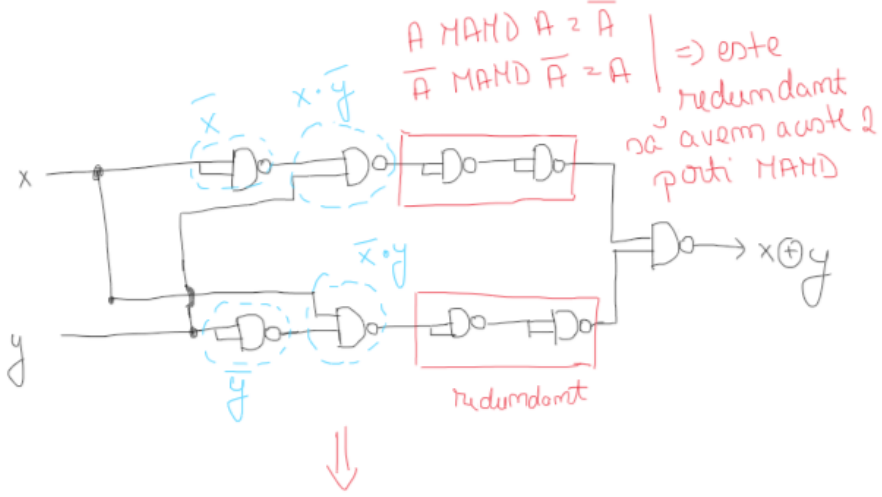
$$= \overline{\overline{(x \cdot \bar{y})} \cdot \overline{(\bar{x} \cdot y)}} =$$

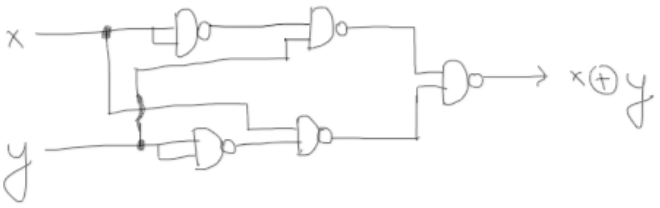
$$= \overline{\overline{x \cdot (y \text{ NAND } y)} \cdot \overline{(x \text{ NAND } x) \cdot y}} =$$

$$= \overline{\overline{(x \text{ NAND } (y \text{ NAND } y))} \text{ NAND } \overline{(x \text{ NAND } (y \text{ NAND } y))}} \cdot$$

$$\cdot \overline{\overline{(x \text{ NAND } x) \text{ NAND } y} \text{ NAND } \overline{(x \text{ NAND } x) \text{ NAND } y}}$$

Chiar daca aceasta nu este forma cu numar minim de porti logice, vom desena diagrama pentru ca asa ne va fi foarte usor sa observam unde se realizeaza redundanta.

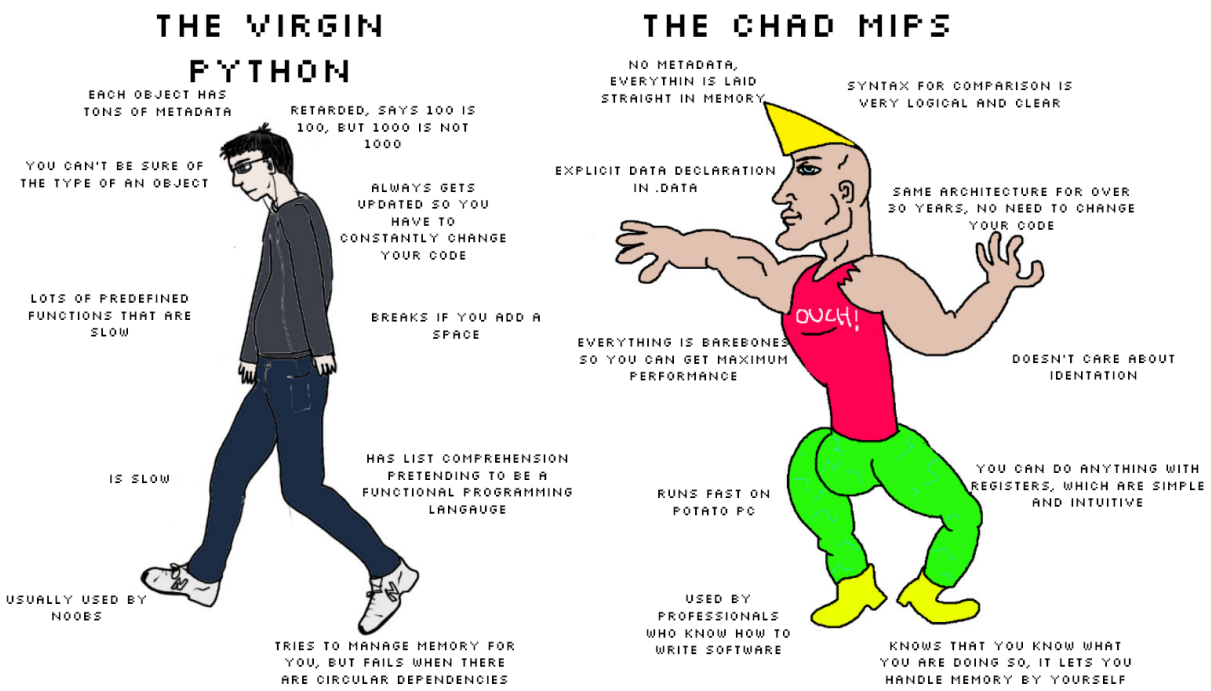




$$\Rightarrow x \oplus y = \Big((x \text{ NAND } x) \text{ NAND } y\Big) \text{ NAND } \Big((y \text{ NAND } y) \text{ NAND } x\Big)$$

Exemplul 6 Implementati poarta NXOR printr-un circuit cu numar minim de porti si doar cu porti NAND.
Avand atatea exemple mai sus, incercati exercitiul acesta singuri.

3 MIPS



3.1 Șiruri de caractere

Sunt tablouri unidimensionale de un byte și sunt finalizate cu caracterul '\0'. Această convenție de a termina șirurile de caractere cu '\0' este comună în multe limbaje de programare. În felul acesta se pot parcurge caracterele, iar când se ajunge la capătul șirului să se oprească, fără a mai fi nevoie de o variabilă auxiliară în care să se țină lungimea șirului.
Pentru a verifica dacă un caracter este egal cu '\0' se poate folosi beqz.

Exemple:
Problema 1 Să se afișeze pe ecran lungimea unui șir de caractere.

```
.data
    str:.asciiz "Sir de caractere" # sirul de caractere
.text
main:

    li $t0, 0                      # counterul
    lb $t1, str($t0)               # iau primul caracter din sir

    loop:

        beqz $t1, exit             # daca am ajuns la sfarsitul sirului ne oprim

        addi $t0, 1
        lb $t1, str($t0)           # ia urmatorul caracter din sir
        j loop

exit:

    move $a0, $t0
    li $v0, 1
    syscall

    li $v0, 10
    syscall
```

problema1.s

PRINT BYTE
Dacă avem un caracter stocat într-un registru îl putem afișa pe ecran. Apelul de sistem pentru asta are codul 11. În $a0 vom copia caracterul pe care vrem să-l afișăm.
Exemplu PRINT BYTE:

```
.data
    ch:.byte 'a'
.text
main:

    lb $a0, ch # copiem in $a0 caracterul pe care vrem sa-l afisam
    li $v0, 11 # punem in $v0 codul pentru apelul de sistem, adica 11
    syscall    # afiseaza caracterul pe ecran

    li $v0, 10
    syscall
```

printbyte.s

READ STRING
Avem apel de sistem pentru citirea șirurilor de caractere de la tastatură. Codul pentru acest apel de sistem este 8.
Pentru a citi un șir de caractere de la tastatură trebuie să facem următoarele lucruri:

- Punem în registrul $a0 adresa de memorie la care vrem să reținem șirul de caractere
- Punem în registrul $a1 dimensiunea maximă a șirului.
- Punem în registrul $v0 codul pentru apelul de sistem, adică 8.

Exemplu READ STRING:

```
.data
    str:.space 100 # sir de caractere de lungime 99
                   # lungimea este 99, nu 100
                   # deoarece ultimul caracter
                   # este '\0'
.text
main:

                   # citeste sirul de caractere
    la $a0, str    # pune in $a0 adresa de memorie la
                   # care vreau sa retin sirul de caractere
    li $a1, 99     # dimensiunea maxim  a sirului
    li $v0, 8      # codul pentru apelul de sistem este 8
    syscall

                   # afiseaza sirul de caractere
    la $a0, str
    li $v0, 4
    syscall
                   # exit
    li $v0, 10
    syscall
```

readstring.s

Mai multe exerciții
Problema 1: Se citește un șir de caractere de dimensiune maximă 99. Să se afișeze pe ecran caracterele de pe poziții pare.
Problema 2: Se dă un șir de caractere la nivel de memorie, să se modifice șirul adăugând un + 1 pe codul ASCII al fiecărui element.
Exemplu: ”abc xyz” -> ”bcd!yz{”
Problema 3: Prima parte din problema din prima zi de advent of code din 2017: https://adventofcode.com/2017/day/1

3.2 Lucrul cu numere în virgulă mobilă

Regiștri

În MIPS avem regiștri $\$f0$-$\$f31$ dedicați pentru lucrul cu numere în virgulă mobilă. Fiecare registru este pe 32 de biți, deci poate ține un single. Pentru double, sunt folosiți doi regiștri (unul după altul). De exemplu dacă am vrea să ținem un double în regiștrii $\$f0$, respectiv $\$f1$, 32 de biți vor fi ținuți în $\$f0$, iar restul vor fi ținuți în $\$f1$.

Tipuri de date

Se declară ca și celelate tipuri de date în .data.

Pentru numerele în virgulă mobilă avem tipurile:

- .float - tip de date pe 32 de biți folosit pentru a stoca numere single
- .double - tip de date pe 64 de biți folosit pentru a stoca numere double

Exemple de date declarate

a:.float 10.54
b:.double 20.31

Instrucțiuni pentru pentru transferarea datelor din regiștri și invers

Când lucrăm cu single și double, de regulă postfixăm numele instrucțiunii noastre cu .s, respectiv .d. Din cauza faptului că pentru double sunt folosiți doi regiștri, instrucțiunile pentru double funcționează doar pe regiștrii cu număr par (ex: $\$f0$, $\$f2$, $\$f4$, etc). În unele implementări de MIPS, regula rămâne valabilă și pentru instrucțiunile pentru numere single.

Instrucțiuni pentru încărcarea valorilor în regiștri:

l.s/ l.d $\$fd$, *mem* (load single/ load double, încarcă în registrul $\$fd$, valoarea aflată în memorie în *mem*. În cazul la double, va fi folosit și următorul registru pentru a se stoca în el. Este valabil pentru orice instrucțiune de transfer al datelor pe double)

li.s/ li.d $\$fd$, *const* (Încarcă o valoarea constantă *const* în registrul $\$fd$)

Instrucțiuni pentru salvarea valorilor din regiștri în memorie:

s.s/ s.d $\$fs$, *mem* (store signle/ store double, salvează valoarea din registrul $\$fs$ la adresa *mem*)

Exemple de folosire a acestor instrucțiuni:

```
l.s $f0, a # încarcă în registrul $f0 valoarea din a (adică 10.54)
l.d $f2, b # încarcă în regiștrii $f2 și $f3 valoarea din b (adică 20.31)
li.s $f4, 15.3 # încarcă în registrul $f4 valoarea constantă 15.3
li.d $f6, 21.9 # încarcă în registrul $f6 valoarea constantă 21.9
s.s $f4, a # salvează valoarea din registrul $f4 la adresa de memorie a lui a
```

s.d $f6, b # salvează valoarea din registrul $f6 (respectiv $f7) la adresa de memorie a lui b.

instrucțiuni aritmetice

add.s/ add.d *dest*, *src*1, *src*2 (adună numerele din regiștri *src*1 și *src*2 și salvează rezultatul în registrul *dest*. Și la aceste instrucțiuni rămâne valabil comportamentul pentru dobule, sunt folosiți câte doi regiștri)

sub.s/ sub.d *dest*, *src*1, *src*2 (asemănător cu add.s/ add.d, realizează scăderea dintre valorile din registrul *src*1 și *src*2, iar rezultatul îl salvează în registrul *dest*. sub.d este pentru double)

mul.s/ mul.d *dest*, *src*1, *src*2 (realizează operația $dest = src1 \cdot src2$)

div.s/ div.d *dest*, *src*1, *src*2 (realizează operația $dest = src1 \ / \ src2$)

neg.s/ neg.d *dest*, *src* (realizează operația $dest = -src$)

abs.s/ abs.d *dest*, *src* (realizează operația $dest = \|\text{src}\|$)

IO + apeluri de sistem

Coduri pentru apeluri de sistem:

2 - PRINT SINGLE (afișează pe ecran un număr single, se încarcă în $\$f12$ valoarea de afișat, în $v0$ valoarea 2 (codul pentru apelul de sistem), iar apoi se scrie syscall)

3 - PRINT DOUBLE (afișează pe ecran un număr double, se încarcă in $\$f12$, respectiv $\$f13$ valoarea de afișat, în $v0$ valoarea 3 (codul pentru apelul de sistem), iar apoi se scrie syscall)

6 - READ SINGLE (citește un single de la tastatură, se încarcă în $v0$ valoarea 6 (codul pentru apelul de sistem), apoi se scrie syscall. După ce a reușit să citească numărul de la tastatură, sistemul ne va returna valoarea citită în registrul $\$f0$)

7 - READ DOUBLE (citește un double de la tastatură, se încarcă în $v0$ valoarea 7 (codul pentru apelul de sistem), apoi se scrie syscall. După ce a reușit să citească numărul de la tastatură, sistemul ne va returna valoarea citită în regiștri $\$f0$, respectiv $\$f1$)

Exemple de apeluri de sistem

READ SINGLE
li $v0, 6
syscall mov.s $f2, $f0 # În $f0 am primit valoarea citită de la tastatură. O vom copia în regisrul $f2 cu ajutorul lui mov.s (folosim mov.d în cazul în care avem double). Sintaxa pentru mov.s este mov.s *regd*, *regs*. Instrucțiunea copiază valoarea din registrul *regs* în

registrul *regd* (analog pentru mov.d).

Exerciții și probleme

Problema 1: Se dau două numere de tip single stocate în memorie, să se interschimbe valorile celor două numere și să se afișeze pe ecran noile valori.

```
.data

    x:.float 11.4     # x este un single declarat in memorie cu valoarea implicita 11.4
    y:.float 34.5     # y -> single cu valoarea 34.5

.text
main:

    l.s $f0, x        # incarc in $f0 valoarea din x
    l.s $f2, y        # incarc in $f2 valoarea din y
                      # am ales $f2 in loc de $f1
                      # deoarece in unele medii de lucru
                      # nu putem folosi $f-urile impare
                      # nici pentru operatii cu single

    s.s $f0, y        # pun in y valoarea din f0 (adica ce era inainte in x)
    s.s $f2, x        # pun in x valoarea din f2 (adica ce era inainte in y)

    li $v0, 2         # codul pentru a afisa un single pe ecran
    mov.s $f12, $f2   # pun in $v12 valoarea din f2 (adica ce avem acum in x)
    syscall           # afisez pe ecran

    li $v0, 11        # afisez spatiu intre rezultate
    li $a0, ' '       # pun in a0 caracter pentru spatiu
                      # in unele medii (spim) putem face direct
                      # fara a mai declara in memorie spatiul
    syscall

    li $v0, 2
    mov.s $f12, $f0   # acum afisez valoarea din $f0 (adica ce avem acum in y)
    syscall

    li $v0, 10
    syscall
```

problema1_1.s

Branch instructions

Sunt asemănătoare cu cele de la word-uri, deși diferă puțin sintaxa și modul de lucru. Fiecare insrucțiune de verificare a condițiilor începe cu "c.", este urmată de "eq", "lt" sau "le", iar apoi este urmată de ".s" sau ".d" (în funcție dacă folosim single sau double).
Instrucțiunile:

c.eq.s/ c.eq.d fs, ft (comparison equal, verifică dacă fs și ft sunt egale (deși nu aș recomanda să faceți asta vreodată cu float-uri... in niciun limbaj de programare). Dacă sunt egale, atunci seteaza bitului de condiție (parte din procesor) valoarea 1, altfel valoarea 0)

c.lt.s/ c.lt.d fs, ft (comparison less than, verifică daca fs este mai mic decât ft, iar dacă este, setează valoarea bitului de condiție valoarea 1, altfel valoarea 0)

c.le.s/ c.le.d fs, ft (comparison less than or equal, verifică daca fs este mai mic sau egal cu

ft, iar dacă este, setează valoarea bitului de condiție valoarea 1, altfel valoarea 0)

Asta nu e tot ce trebuie să facem, aceste instrucțiuni doar au modificat valoarea bitului de condiție. noi acum trebuie ne ducem la un branch dacă această valoare este 1 sau 0.

Pentru asta avem următoarele instrucțiuni:

bc1t label (continuă executarea de la linia unde este *label*, dacă bitul de condiție are valoarea 1)

bc1f label (continuă executarea de la linia unde este *label*, dacă bitul de condiție are valoarea 0)

Exerciții și probleme

Problema 1: Să se afișeze pe ecran toate valorile pozitive mai mici sau egale cu n (single citit de la tastatură) pornind de la 0 și incrementând cu un step de 0.1.

```
.data

    n:.float 0.0            # numarul n, il vom tine si in memorie (doar in scop didactic)

.text
main:

    li $v0, 6               # citeste n
    syscall
    s.s $f0, n              # vom pastra in $f0 valoarea lui n

    li.s $f2, 0.1           # pasul cu care vom incrementa counterul
    li.s $f4, 0.0           # counterul nostru, initializat cu 0

    loop:

        c.le.s $f0, $f4     # daca counterul nostru este >= f0, atunci am terminat
                            # deci seteaza bitului de conditie valoarea 1
        bc1t exit           # daca bitul de conditie este 1, am terminat
                            # din pacate nu avem branching ca la word-uri
                            # si nici nu putem verifica daca un numar e mai mare(sau egal)
                            # cu altul, trebuie sa verificam daca sunt mai mici(sau egale)

        li $v0, 2           # afiseaza numarul curent
        mov.s $f12, $f4
        syscall

        li $v0, 11
        li $a0, ' '
        syscall

        add.s $f4, $f4, $f2 # incrementeaza counterul cu step
                            # adica $t4 = $t4 + $t2

        j loop

exit:

    li $v0, 10
    syscall
```

problema1_2.s

Conversii

Putem copia bit cu bit valorile din regiștri normali în regiștri în virgulă mobilă (și invers) folosind următoarele instrucțiuni:

mtc1 rs, fd (copiază din registrul normal rs, în registrul pentru floating point fd)

mfc1 rd, fs (copiază din registrul pentru floating point fs, in registrul normal rd)

Aceste instrucțiuni nu fac conversie între tipuri, ci doar copiază bit cu bit valorile. Pentru a converti avem instrucțiunea cvt, care se postfixează cu ".dest.sursa", unde *dest* si *sursa* reprezintă tipul către care vrem să convertim, respectiv tipul de la care convertim. Dest și sursă pot avea valorile "s", "d", respectiv "w" (pentru word).

Sintaxa: cvt.dest.sursa ds, sr -> convertește valoarea din sr (de tipul sursa) în valoare de tip dest și salvează rezultatul în registrul ds.

Exerciții și probleme

Problema 1: Se citește un număr de tip single de la tastatură, să se convertească la word, iar apoi să se afișeze pe ecran rezultatul.

```
1  .data
2  .text
3  main:
4
5      li $v0, 6
6      syscall
7
8      cvt.w.s $f2, $f0 # convertim pe $f0 (citit) in word, si tinem resultatul in $f2
9                       # noi am vrea sa-l tinem in $t0 pentru ca este word, dar compilerul nu
                            ne lasa
10                      # asa ca trebuie sa punem un registru pentru float-uri ca destinatie
11     mfc1 $t0, $f2    # punem rezultatul din $f2 in $t0 (in $t0, este reprezentat intern ca
           un
12                      # word, dar este in registru de float, deci il copiem intr-un registru
13                      # pentru word-uri)
14
15     li $v0, 1        # afisam pe ecran rezultatul
16     move $a0, $t0
17     syscall
18
19     li $v0, 10
20     syscall
```

problema1_3.s

Mai multe exerciții

Problema 1: Se citesc n $\in N^*$ și n numere de tip single. Să se calculeze media lor aritmetică.

References

[1] Dumitru Daniel Drăgulici. *Curs Arhitectura Sistemelor de Calcul.*

[2] Larisa Dumitrache. *Tutoriat 2019*

[3] Bogdan Macovei. *Laboratoare ASC 2019/ 2020*

[4] Advent Of Code. *Day 1 2017*