

# Arhitectura calculatoarelor și sisteme de operare

**Prof. dr. Henri Luchian**  
**Lect. dr. Vlad Rădulescu**

# Evaluare

- examinare
  - două teste scrise din materia de curs
    - câte unul pentru fiecare jumătate de semestru
  - un test practic la laborator
    - limbaj de asamblare
- condiția pentru susținerea testelor scrise
  - prezența la laborator
    - cel mult 2 absențe permise în fiecare jumătate de semestru

# Cuprins - prima jumătate

- I. Introducere
- II. Circuite combinaționale și funcții booleene
- III. Circuite secvențiale și automate
- IV. Reprezentări interne
- V. Arhitectura și organizarea calculatorului

# I. Introducere

# I.1. Evoluție

# Cum definim noțiunea de calcul?

- ce operații se pot realiza?
- evoluție în timp
  - abacul: adunări
  - roți dințate (Leibniz, Pascal): adunări, înmulțiri
  - Babbage: instrucțiuni încărcate din exterior, calcul ramificat
  - von Neumann: program memorat; execuție în secvență de instrucțiuni; ierarhii de memorii
  - calcul paralel, cuantic etc.

# Mașini de calcul universale

- o mașină de calcul universală se poate comporta ca oricare mașină de calcul particulară
  - deci poate rezolva orice problemă pe care o poate rezolva o mașină de calcul particulară
- exemplu - calculatorul
  - în funcție de programul executat, rezolvă probleme de: calcul matricial, grafică, tehnoredactare etc.

## Scurtă istorie (1)

- scrierea pozițională
  - indieni, arabi
- algebra booleană
  - George Boole, 1854
- teorema de incompletitudine
  - Kurt Gödel, 1935
- legătura între algebra booleană și circuite
  - Claude Shannon, 1938



## Scurtă istorie (2)

- calculatorul neumannian
  - John von Neumann, 1946
- tranzistorul
  - Shockley, Brittain, Bardeen, 1947
- circuitele integrate

## I.2. Legi empirice

# Legi empirice

- în orice domeniu al științei, legile sunt determinate într-un fel sau altul de experiment sau de observații în lumea concretă
- repetabilitatea duce la ideea de legi empirice: adevăruri valabile de cele mai multe ori, conform observațiilor

# Legi empirice în informatică

- legea "90:10" (Donald Knuth)
  - 90% din timpul de execuție al unui program este utilizat pentru 10% din instrucțiuni
- legea lui Amdahl
  - eficiența maximă în îmbunătățirea unui sistem (concret sau abstract) se atinge dacă se optimizează subsistemul cel mai folosit
- legile localizării - spațială, temporală

# Legea lui Amdahl (1)

- considerăm un sistem (hardware, software) și o anumită componentă a sa
- componenta respectivă lucrează un procentaj  $f_a$  din timpul de lucru al sistemului
- și este îmbunătățită, astfel încât lucrează de  $a$  ori mai rapid decât înainte
- de câte ori mai rapid devine sistemul?

## Legea lui Amdahl (2)

$$A(a, f_a) = \frac{1}{(1 - f_a) + \frac{f_a}{a}}$$

- creștere de viteză generală cât mai mare
  - îmbunătățirea pronunțată a componentei ( $a$ )
  - îmbunătățirea componentelor cu o pondere ( $f_a$ ) cât mai mare
    - deci mai des folosite

# Localizare temporală

- dacă o locație de memorie este accesată la un moment dat, este foarte probabil să fie accesată din nou în viitorul apropiat
- exemple
  - variabilele sunt folosite în mod repetat
  - bucle de program - instrucțiunile se repetă

# Localizare spațială

- dacă o locație de memorie este accesată la un moment dat, este foarte probabil ca și locațiile vecine să fie accesate în viitorul apropiat
- exemple
  - parcurgerea tablourilor
  - execuția secvențelor de instrucțiuni - aflate la adrese consecutive



# Ordine fizică și ordine logică

- instrucțiunile de executat se află în memorie în ordinea fizică
- sunt citite din memorie și executate
  - regula: în ordinea în care sunt memorate (fizic)
  - excepția: sărind peste un număr de instrucțiuni
- astfel rezultă ordinea logică a instrucțiunilor
  - poate diferi de la o rulare la alta
  - o instrucțiune se poate executa de 0, 1, 2, ... ori

# II. Circuite combinaționale și funcții booleene

# Semnal analogic și semnal digital

- semnal analogic - continuu
  - dacă poate lua valorile  $a$  și  $b$ , atunci poate lua orice valoare din intervalul  $[a,b]$
- semnal digital - discret
  - are câteva niveluri (valori) distincte pe care le poate lua
  - calculator - semnal digital cu 2 niveluri (0 și 1)
  - există și alte sisteme de calcul în afară de PC

# Tipuri de circuite

- circuite combinaționale
  - valorile ieșirilor depind exclusiv de valorile intrărilor
  - aceleași valori pe intrare produc întotdeauna aceleași valori la ieșire
- circuite secvențiale
  - în afară de intrări, valorile ieșirilor depind și de starea în care se află circuitul
  - evoluează în timp

## Tabele de adevăr

- cum putem descrie funcționarea unui circuit combinațional?
- se aplică fiecare combinație posibilă de valori ale intrărilor
- și se observă valorile ieșirilor pentru fiecare astfel de combinație
- ansamblul acestor corespondențe formează un tabel de adevăr

# Circuite și funcții booleene

- fiecărui tabel de adevăr îi corespunde o funcție booleană
  - deci fiecărui circuit combinațional îi corespunde o funcție booleană

intrări			ieșiri		
$I_1$	...	$I_n$	$O_1$	...	$O_m$
0	0...0	0	?	?...?	?
0	0...0	1	?	?...?	?
...	...	...	...	...	...
1	1...1	1	?	?...?	?

# II.1. Funcții booleene

# Structura algebrică

- mulțimea nevidă  $B$ , care conține cel puțin două elemente:  $a, b, a \neq b$
- mulțimea de operații binare  $\{ +, \cdot \}$
- o operație unară  $\{ \bar{\phantom{x}} \}$
- închidere:  
 $a + b \in B$   
 $a \cdot b \in B$   
 $\bar{a} \in B$



# Funcții booleene

- $B = \{0,1\}$
- $f : B^n \rightarrow B^m$ 
  - funcție:  $n$  variabile,  $m$  valori
  - circuit:  $n$  intrări,  $m$  ieșiri
- există  $(2^m)^{2^n}$  astfel de funcții
  - $n = 1, m = 1$ : 4 funcții unare cu o valoare
  - $n = 2, m = 1$ : 16 funcții booleene de 2 variabile și cu o valoare

# Tabele de adevăr

$a$	$f_0(a)$	$f_1(a)$	$f_2(a)$	$f_3(a)$
0	0	0	1	1
1	0	1	0	1
	$= 0$	$= a$	$= \bar{a}$	$= 1$

$a$	$b$	$F_0$	$F_1$	$F_2$	$F_3$	$F_4$	$F_5$	$F_6$	$F_7$	$F_8$	$F_9$	$F_{10}$	$F_{11}$	$F_{12}$	$F_{13}$	$F_{14}$	$F_{15}$
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

# Axiome și teoreme în algebra booleană (1)

identitate	$X + 0 = X$	$X \cdot 1 = X$
constante	$X + 1 = 1$	$X \cdot 0 = 0$
idempotență	$X + X = X$	$X \cdot X = X$
involuție	$\overline{\overline{X}} = X$	
complementaritate	$X + \overline{X} = 1$	$X \cdot \overline{X} = 0$
comutativitate	$X + Y = Y + X$	$X \cdot Y = Y \cdot X$
asociativitate	$(X + Y) + Z = X + (Y + Z)$	$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$
distributivitate	$X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$	$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

# Axiome și teoreme în algebra booleană (2)

unificare	$X \cdot Y + X \cdot \bar{Y} = X$	$(X + Y) \cdot (X + \bar{Y}) = X$
absorbție	$X + X \cdot Y = X$ $(X + \bar{Y}) \cdot Y = X \cdot Y$	$X \cdot (X + Y) = X$ $(X \cdot \bar{Y}) + Y = X + Y$
De Morgan	$\overline{X + Y + \dots} = \bar{X} \cdot \bar{Y} \cdot \dots$	$\overline{X \cdot Y \cdot \dots} = \bar{X} + \bar{Y} + \dots$
generalizare (dualitate)	$\overline{f(X_1, \dots, X_n, 0, 1, +, \cdot)} = f(\bar{X}_1, \dots, \bar{X}_n, 1, 0, \cdot, +)$	

# Calculatorul - operații elementare

- în calculatoarele actuale, operațiile elementare sunt operațiile logicii booleene
  - care simulează (între altele) și operațiile aritmetice elementare în baza 2
- un circuit combinațional implementează de fapt o funcție booleană
  - cum obținem expresia funcției booleene pornind de la tabelul de adevăr?

# Forme normale

- forma normală disjunctivă (FND)
  - pentru fiecare linie care produce valoarea 1 la ieșire - termen conjuncție ( $\cdot$ )
    - conține fiecare variabilă a funcției: negată dacă variabila este 0 pe acea linie, nenegată dacă este 1
  - acești termeni sunt legați prin disjuncție (+)
- forma normală conjunctivă (FNC): dual
- exemplu:  $F_9(x, y) = \bar{x} \cdot \bar{y} + x \cdot y = (x + \bar{y}) \cdot (\bar{x} + y)$

## II.2. Diagrame logice

# Alfabetul diagramelor logice (1)

- porțile logice reprezintă implementările unor funcții booleene
- deci funcționarea fiecărei porți poate fi descrisă printr-un tabel de adevăr
  - corespunzător funcției booleene asociată porții
- porți elementare: AND, OR, NOT
- alte porți utile: NAND, NOR, XOR, NXOR

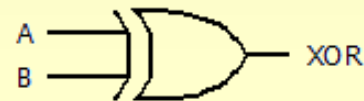
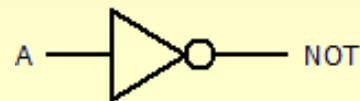
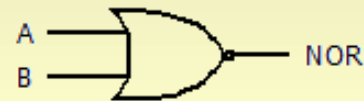


# Alfabetul diagramelor logice (2)

A	NOT
0	1
1	0

A	B	AND	OR	NAND	NOR	XOR	NXOR
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

# Simbolurile porților



- operațiile binare asociative pot fi extinse la operații cu orice număr finit de operanzi



## Set minimal de generatori

- set de generatori - mulțime de tipuri de porți prin care se poate implementa orice funcție booleană
  - set minimal de generatori - set de generatori cu numărul minim de tipuri de porți
- se poate cu 3 (NOT, AND, OR)
  - formele normale (disjunctivă, conjunctivă)
  - se poate și cu 2 (NOT și AND, NOT și OR)
  - minimal - 1 (NAND, NOR)

# Temă

- arătați că următoarele mulțimi de tipuri de porți sunt seturi de generatori:
  - NOT, AND
  - NOT, OR
  - NAND
  - NOR

## II.3. Implementarea circuitelor prin funcții booleene

# Definirea funcțiilor booleene

- moduri de definire
  - tabel de adevăr
  - expresii conținând variabile și operații logice
  - în formă grafică
  - sigma-notație ( $\Sigma$ )
- în final, ne interesează să avem o expresie booleană
  - care permite implementarea prin porți

## $\Sigma$ -notația (1)

- exemplu - "majoritatea dintre k intrări"
  - valoarea funcției: 1 dacă majoritatea variabilelor au valoarea 1, 0 în caz contrar
    - pentru 3 variabile:  $f(x_1, x_2, x_3) = \Sigma(3, 5, 6, 7)$
- $\Sigma$ -notația corespunde formeii normale disjunctive
  - fiecare număr din paranteză reprezintă un termen conjuncție
  - $\Sigma$  denotă disjuncția termenilor

## $\Sigma$ -notația (2)

- $\Sigma$ -notație dată - câte variabile sunt necesare?
  - cea mai mică putere a lui 2 care cuprinde cel mai mare număr dintre paranteze
    - pentru exemplul nostru:  $2^2 < 7 < 2^3 \rightarrow n = 3$
- termenul corespunzând unui număr conține
  - toate variabilele, legate prin conjuncție
  - fiecare variabilă este: negată dacă îi corespunde un 0; nenegată pentru 1
    - exemplu:  $3_{(10)} = 011_{(2)} \rightarrow \overline{x_1} \cdot x_2 \cdot x_3$



## Minimizare (1)

- forma normală disjunctivă a funcției majoritate din 3

$$f(A, B, C) = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C} + A \cdot B \cdot C$$

- număr mare de aplicări ale funcțiilor elementare
- o expresie echivalentă (**aceeași funcție booleană**) mai simplă ar face circuitul
  - mai rapid
  - mai ieftin
  - mai fiabil

## Minimizare (2)

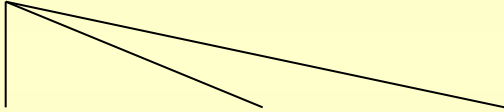
- cum putem simplifica expresia dată de forma normală disjunctivă?
  - rescriere echivalentă
    - utilizarea legilor și axiomelor algebrei booleene
  - inducție perfectă
  - metoda Veitch-Karnaugh
  - metoda Quine-McCluskey
  - hibridizare (combinarea metodelor de mai sus)

# Minimizare - rescriere algebrică

- același exemplu

$$f = \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C} + A \cdot B \cdot C$$

(idempotență)


$$= \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C} + A \cdot B \cdot C + A \cdot B \cdot C + A \cdot B \cdot C$$

(unificare)


$$= B \cdot C + A \cdot C + A \cdot B$$

- dificil pentru expresii complexe

# Temă

- determinați forma normală disjunctivă și studiați minimizarea prin rescriere algebrică pentru funcția "imparitate"
  - valoarea funcției este: 1 dacă numărul de intrări cu valoarea 1 este impar; 0 în caz contrar