Arhitectura calculatoarelor şi 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
- condiţia pentru susţinerea testelor scrise
  - prezenţa la laborator
    - cel mult 2 absenţe permise în fiecare jumătate de semestru

Cuprins - prima jumătate

I. Introducere

II. Circuite combinaţionale şi funcţii booleene

III. Circuite secvenţiale şi automate

IV. Reprezentări interne

V. Arhitectura şi organizarea calculatorului

I. Introducere

I.1. Evoluţie

Cum definim noţiunea de calcul?

- ce operaţii se pot realiza?
- evoluţie în timp
  - abacul: adunări
  - roţi dinţate (Leibniz, Pascal): adunări, înmulţiri
  - Babbage: instrucţiuni încărcate din exterior, calcul ramificat
  - von Neumann: program memorat; execuţie în secvenţă de instrucţiuni; ierarhii de memorii
  - calcul paralel, cuantic etc.

Maşini de calcul universale

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

Scurtă istorie (1)

- scrierea poziţională
  – indieni, arabi
- algebra booleană
  – George Boole, 1854
- teorema de incompletitudine
  – Kurt Gödel, 1935
- legătura între algebra booleană şi 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 ştiinţei, legile sunt determinate într-un fel sau altul de experiment sau de observaţii în lumea concretă
- repetabilitatea duce la ideea de legi empirice: adevăruri valabile de cele mai multe ori, conform observaţiilor

Legi empirice în informatică

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

Legea lui Amdahl (1)

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

Legea lui Amdahl (2)

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

- creştere de viteză generală cât mai mare
  - – îmbunătăţirea pronunţată a componentei (a)
  - – îmbunătăţirea componentelor cu o pondere (f_a) cât mai mare
    - deci mai des folosite

Localizare temporală

- dacă o locaţie 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 - instrucţiunile se repetă

Localizare spaţială

- dacă o locaţie de memorie este accesată la un moment dat, este foarte probabil ca şi locaţiile vecine să fie accesate în viitorul apropiat
- exemple
  - parcurgerea tablourilor
  - execuţia secvenţelor de instrucţiuni - aflate la adrese consecutive

Ordine fizică şi ordine logică

- instrucţiunile de executat se află în memorie în ordinea fizică
- sunt citite din memorie şi executate
  - regula: în ordinea în care sunt memorate (fizic)
  - excepţia: sărind peste un număr de instrucţiuni
- astfel rezultă ordinea logică a instrucţiunilor
  - poate diferi de la o rulare la alta
  - o instrucţiune se poate executa de 0, 1, 2, … ori

II. Circuite combinaţionale şi funcţii booleene

Semnal analogic şi semnal digital

- semnal analogic - continuu
  - dacă poate lua valorile *a* şi *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 şi 1)
  - există şi alte sisteme de calcul în afară de PC

Tipuri de circuite

- circuite combinaţionale
  - valorile ieşirilor depind exclusiv de valorile intrărilor
  - aceleaşi valori pe intrare produc întotdeauna aceleaşi valori la ieşire
- circuite secvenţiale
  - în afară de intrări, valorile ieşirilor depind şi de starea în care se află circuitul
  - evoluează în timp

Tabele de adevăr

- cum putem descrie funcţionarea unui circuit combinaţional?
- se aplică fiecare combinaţie posibilă de valori ale intrărilor
- şi se observă valorile ieşirilor pentru fiecare astfel de combinaţie
- ansamblul acestor corespondenţe formează un tabel de adevăr

Circuite şi funcţii booleene

- fiecărui tabel de adevăr îi corespunde o funcţie booleană
  - deci fiecărui circuit combinaţional îi corespunde o funcţie booleană

intrări			ieşiri		
I_1	**...**	**I_n**	**O_1**	**...**	**O_m**
0	0...0	0	?	?...?	?
0	0...0	1	?	?...?	?
...	...	...	...	...	...
1	1...1	1	?	?...?	?

II.1. Funcţii booleene

Structura algebrică

- mulţimea nevidă B, care conţine cel puţin două elemente: a, b, $a \neq b$
- mulţimea de operaţii binare $\{ + , \cdot \}$
- o operaţie unară $\{ \bar{\ } \}$
- închidere: $a + b \in B$

  $a \cdot b \in B$

  $\bar{a} \in B$

Funcții booleene

- B = {0,1}
- f : $B^n \rightarrow B^m$
  - funcţie: *n* variabile, *m* valori
  - circuit: *n* intrări, *m* ieşiri
- există $(2^m)^{2^n}$ astfel de funcţii
  - $n = 1$, $m = 1$: 4 funcţii unare cu o valoare
  - $n = 2$, $m = 1$: 16 funcţii booleene de 2 variabile şi 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 şi 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 şi teoreme în algebra booleană (2)

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

Calculatorul - operaţii elementare

- în calculatoarele actuale, operaţiile elementare sunt operaţiile logicii booleene
  - care simulează (între altele) şi operaţiile aritmetice elementare în baza 2
- un circuit combinaţional implementează de fapt o funcţie 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 (·)
    - 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)

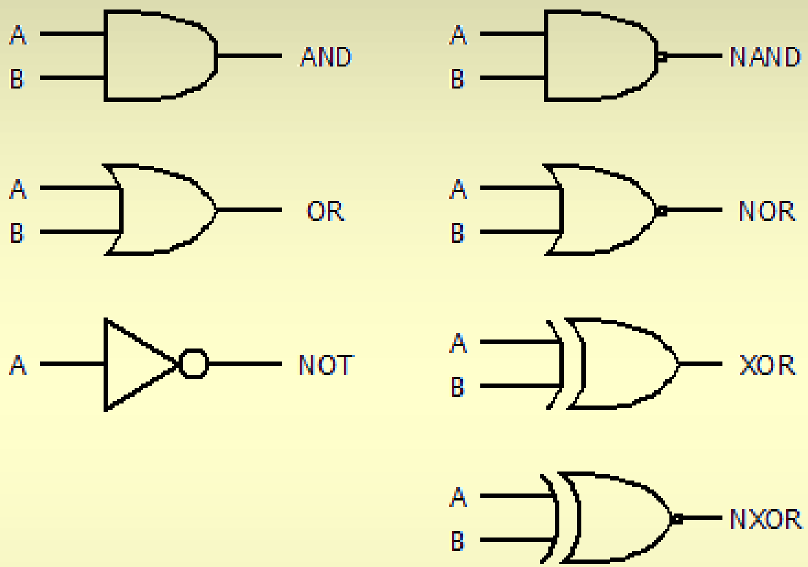
- porțile logice reprezintă implementările unor funcții booleene
- deci funcționarea fiecărei porți poate fi descrisă printr-un tabel de adevăr
  - corespunzător funcției booleene asociată porții
- porți elementare: AND, OR, NOT
- alte porți utile: NAND, NOR, XOR, NXOR

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



– operaţiile binare asociative pot fi extinse la operaţii cu orice număr finit de operanzi

Set minimal de generatori

- set de generatori - mulțime de tipuri de porţi prin care se poate implementa orice funcţie 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 funcţiilor booleene

- moduri de definire
  - tabel de adevăr
  - expresii conţinând variabile şi operaţii logice
  - în formă grafică
  - sigma-notaţie (Σ)
- în final, ne interesează să avem o expresie booleană
  - care permite implementarea prin porți

Σ-notaţia (1)

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

Σ-notaţia (2)

- Σ-notaţie 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 conţine
  - toate variabilele, legate prin conjuncţie
  - 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 funcţiilor elementare
  - o expresie echivalentă (**aceeaşi funcţie 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
  - inducţie perfectă
  - metoda Veitch-Karnaugh
  - metoda Quine-McCluskey
  - hibridizare (combinarea metodelor de mai sus)

Minimizare - rescriere algebrică

- același exemplu

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

(idempotență)

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

(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