Arhitectura Sistemelor de Calcul (ASC)
Examinarea finală
Varianta 2
(2024 - 2025)

Anul I, Semestrul I
7 februarie 2025
Cristian Rusu

Nume: _____

Prenume: _____

Grupa: _____

Completați aici totul cu majuscule.

Toate răspunsurile sunt în albastru.

Înainte de a începe, citiți cu atenție indicațiile următoare:

- Testul și rezolvarea sa vor fi disponbile online în zilele următoare.
- *Nu aveți voie cu laptop-uri sau alte dispozitive de comunicație.*
- *Nici calculatoarele de buzunar nu sunt permise.*
- *Vă rugăm să vă opriți telefoanele mobile.*
- Pentru întrebările cu răspunsuri multiple/simple folosiți tabelele puse la dispoziție.
- Acest test are 6 enunțuri totalizând 100 de puncte.
- Aveți la dispoziție 120 de minute pentru a completa examinarea.
- Mult succes!

Întrebarea 1. (22 puncte)

Completați tabelul de mai jos cu valorile numerice corecte. Toate numerele sunt naturale pe 16 biți. (Fiecare răspuns corect valorează 1 punct)

binar	octal	zecimal	hexazecimal
b0000000010011001	o00231	153	0x0099
b0000100100001010	o04412	2314	0x090A
b0000100101000000	o04500	2368	0x0940
b0000110100011100	o06434	3356	0x0D1C

a binar	a hexa	b binar	b hexa	a+b zecimal	a+b binar	a+b hexa
b0001000100	0x44	b0001000010	0x42	134	b0010000110	0x86

a binar	a hexa	b binar	b hexa	axb zecimal	axb binar	axb hexa
b0001111100	0x7C	b0000000110	0x06	744	1011101000	0x2E8

Răspunsurile care țin cont de overflow sunt de asemenea corecte.

Întrebarea 2. (13 puncte)

Completați tabelul de mai jos cu valorile numerice corecte. Toate numerele sunt întregi pe 10 biți. (Fiecare răspuns corect valorează 1 punct)

binar	octal	zecimal	hexazecimal
b1000001010	o7012	-502	0xE0A
b001000111011	o1073	571	0x23B

a zecimal	a binar	a hexa	b zecimal	b binar	b hexa
-404	b1001101100	0x26C	-26	b1111100110	0x3E6

produs axb zecimal	produs axb binar	produs axb hexa
10504	b10100100001000	0x2908

Răspunsurile care țin cont de overflow sunt de asemenea corecte. S-au punctat și cazurile în care octal si hexazecimal s-au considerat doar 10 biți (1 în loc de E sau 1 în loc de 7).

3.2.	Bjarne Stroustrup
3.4.	Babbage
3.6.	20 biți
3.8.	$\frac{8^n-1}{8-1}$
3.10.	(b_1 XOR a_1) OR (b_0 XOR a_0)
3.12.	!A AND !B
3.14.	(A–B)×(A–B) = A^2 + B^2 – 2(A×B) = A + B – 2(A×B)
3.16.	0xC3800000
3.18.	0xC0F80000
3.20.	verifică dacă ambele numere sunt pozitive
3.22.	WAR
3.24.	1 + 5 + 10 + 50 = 66 ns (ce căutăm e doar în memoria principală)
3.26.	suma = suma + (x[i] & 1) - not(x[i] & 1)
3.28.	Streaming SIMD Extensions
3.30.	CISC
3.32.	Double Data Rate RAM
3.34.	dispozitivele mobile, este memoria low-power

Întrebarea 3. (17 puncte)

Răspundeți la următoarele întrebări scurte. Completați în tabelul de pe pagina anterioară.

3.2. Cine a pus bazele limbajului de programare C++?

3.4. Cine a creat prima mașină de calcul mecanică programabilă?

3.6. În tabelul lui Mendeleev sunt 118 elemente. Fiecare element poate să aibă maxim 7 legături cu alți atomi. Câți biți are codificarea formulei CO_2 (codificare cu dimensiune fixă).

3.8. Dacă un număr scris în binar are reprezentarea un șir de n biți de 1 atunci valorea lui în zecimal este $2^n - 1$? Analog, care este valoarea în zecimal a unui număr care în octal este un șir de n cifre 1?

3.10. Avem două numere naturale a, b pe 2 biți (a este a_1a_0 și b este b_1b_0). Scrieți o expresie logică (folosind AND, OR, NOT, XOR) care verifică dacă a este diferit față de b?

3.12. Simplificați maxim expresia logică !((A AND B) OR (A XOR B)).

3.14. Pentru două valori digitale A și B aveți nevoie să calculați A XOR B, dar sistemul vostru de calcul știe să facă doar operații aritmetice ($+$,$-$,$\times$). Cum calculați A XOR B în această situație?

3.16. în formatul single float IEEE, care este reprezentarea hexazecimală a numărului -256?

3.18. în formatul single float IEEE, care este reprezentarea hexazecimală a numărului -7.75?

3.20. Fie A și B două numere în format IEEE 754 FP pe 32 de biți. Care este rolul secvenței următoare? NOT(A AND 0x80000000)$\gg$31 AND (NOT((B>>30) AND 0x3)$\gg$1))

3.22. Pe un sistem cu arhitectura de calcul RISC-V avem următoarele două instrucțiuni: lw x5, 0(x1) și add x1, x2, x3. Ce fel de hazard este prezent în această secvență?

3.24. Avem un sistem de calcul cu memorie ierarhică: memorie cache L1 de 1MB cu viteza de acces 1ns, memorie cache L2 de 5MB cu viteza de acces 5ns, memorie cache L3 de 10MB cu viteza de acces 10ns și memoria principală RAM de 32GB cu viteza de acces 50ns. Care este timpul maxim de acces în memorie pe un astfel de sistem?

3.26. Într-o variabilă *suma* vrem diferența dintre câte numere impare sunt în vectorul x și câte numere pare sunt în același vector. Care este expresia logică/aritmetică fără *dacă* care calculează corect suma?

3.28. Acronimul SSE înseamnă

3.30. Arhitectura de calcul x86 este de tipul

3.32. Acronimul DDR RAM înseamnă

3.34. LP-DDR este folosită în

Întrebarea 4. (11 puncte)

Avem un fișier în care vrem să scriem o serie de numere pozitive (nu știm exact câte vor fi). Numerele de scris au valori între 0 si $2^{32} - 1$. Avem două posibilități: i) putem să scriem un număr a ca fiind unsigned int pe 32 de biți (adică un număr natural pe 32 de biți) sau ii) putem să scriem un număr a ca fiind un șir de caractere ASCII. Răspundeți la următoarele întrebări și realizați următoarele cerințe:

4.1. (3 puncte) Dacă avem un număr a cu k cifre, pe câți biți este numărul dacă îl considerăm un șir de caractere? Care este valoarea maximă pentru k?

4.2. (4 puncte) Dacă avem nevoie să scriem n numere cu k cifre în medie, de câți biți avem nevoie dacă considerăm fiecare număr ca fiind un șir de caractere? (atenție! după ce le scriem în fișier, vom dori să le și citim corect înapoi)

4.3. (4 puncte) Avem o listă de n numere cu k cifre fiecare (în medie) și vrem să știm cum e mai bine să scriem în fișier numerele (între cele două variante i) și ii)). Ce se întâmplă dacă lista de numere este foarte lungă?

4.1. 8 biți per caracter ASCII, în total $8k$ biți. $k = 10$ pentru $2^{32} - 1$.

4.2. $8kn + 8(n-1)$: $8kn$ biți pentru numere și $8(n-1)$ biți pentru separatori (un caracter ASCII suplimentar ca să știm când se termină un număr și când începe celălalt - separatorul poate să fie orice caracter ASCII non-numeric: spațiu, virgulă, etc.).

4.3. Dacă numerele sunt scrise pe 32 de biți atunci avem $32n$ biți în total (aici nu avem nevoie de separatori pentru că fiecare număr este codificat cu dimensiune fixă, nu variabilă). Atunci, merită să scriem pe biți dacă $32n \leq 8kn + 8(n-1)$ sau $k \geq \frac{1+24n}{8n}$. Adică, dacă numerele sunt scurte (ca număr de cifre) atunci șir de caractere este eficient, altfel e bine să scriem pe 32 de biți. Avem de ales între codare pe dimensiune fixă (32 de biți) versus codarea pe dimensiune variabilă (caractere ASCII). Dacă lista este foarte lungă atunci $n \to \infty$ și avem $k \geq \frac{24}{8} = 3$ ca scrierea pe biți să fie eficientă.

Întrebarea 5. (17 puncte)

Ni se dă un alfabet de simboluri $X = \{A, B, C, D\}$ cu probalitățile $p_i = \{0.5, 0.25, 0.125, 0.125\}$. Răspundeți la următoarele întrebări.

5.1. (2 puncte) Care este entropia alfabetului X?

5.2. (5 puncte) Calculați o codificare pentru alfabetul X folosind algoritmul lui Huffman. Care este lungimea medie a mesajelor codificate astfel? Explicați rezultatul.

5.3. (3 puncte) După fiecare simbol se adaugă un nou simbol E (este un delimitator pentru a identificare individual caracterele). Care este noua entropie a lui X?

5.4. (7 puncte) Ce se întâmplă în general dacă unui aflabet X cu n simboluri cu probabilități p_i îi adăugăm un nou simbol care este un delimitator pentru restul simbolurilor? (adică se adaugă după fiecare simbol) Găsiți răspunsul doar în funcție de entropia alfabetului inițial.

5.1. $H = 1.75$ biți.

5.2. A: 1, B: 01, D: 001, C: 000. Lungimea medie este 1.75 și este egală cu entropia pentru că probabilitățile sunt puteri exacte ale lui 2.

5.3. $H = 1.875$ biți (echivalent cu $1.75/2 + 1$ biți).

5.4. Vom nota $H_{\text{old}} = -\sum p_i \log_2 p_i$ și $q_0 = 1/2$ este noul delimitator iar $q_i = \frac{p_i}{2}$ sunt noile probabilități ale simbolurilor (observați că $q_0 + \sum q_i = 1$). Noua formulă pentru entropie este:

$$\begin{aligned} H_{\text{new}} &= -\sum q_i \log_2 q_i \\ &= -\frac{1}{2}\log_2\frac{1}{2} - \sum q_i \log_2 q_i \\ &= \frac{1}{2} - \frac{1}{2}\sum p_i(\log_2 p_i - 1) \\ &= \frac{1}{2} + \frac{1}{2}H_{\text{old}} + \sum \frac{p_i}{2} \\ &= 1 + \frac{1}{2}H_{\text{old}} \end{aligned} \tag{1}$$

Întrebarea 6. (20 puncte)

Avem o secvență de cod care primește un șir de caractere s și îl transformă într-un număr natural r care corespunde șirului. Presupunem lungime șirului la maxim 10 caractere. Răspundeți la următoarele cerințe:

6.1. (4 puncte) Pentru s = '54321' explicați pas cu pas ce se întâmplă în algoritm.

6.2. (4 puncte) Secvența folosește o operație logică AND între două condiții. Acest lucru este mai lent decât un OR logic, pentru că ambele condiții trebuie evaluate pentru a progresa în cod. Schimbați AND cu OR astfel încât codul să funcționeze în continuare la fel.

6.3. (4 puncte) Puteți să scrieți secvența fără dacă (branchless)?

6.4. (8 puncte) După cum puteți observa din exemplul de la 6.1., calculele sunt secvențiale. Asta înseamnă ca nu putem profita la maxim de capabilitațile ILP ale CPU-ului. Puteți modifica codul astfel încât ILP să fie îmbunătățit?

```
r = 0
pentru i=0,...,lungime(s)-1
    daca s[i] >= '0' si s[i] <= '9' atunci
        r = r*10 + (s[i] - '0')
    altfel
        return ERROR
```

6.1. r = ((((0*10 + 5)*10 + 4)*10 + 3)*10 + 2)*10 + 1.

6.2. Secvența este următoarea.

```
r = 0
pentru i=0,...,lungime(s)-1
    daca s[i] < '0' sau s[i] > '9' atunci
        return ERROR
    altfel
        r = r*10 + (s[i] - '0')
```

6.3. Da, secvența este următoarea.

```
r = 0
pentru i=0,...,lungime(s)-1
    r = r*10 + not((s[i] < '0') | (s[i] > '9'))*(s[i] - '0')
```

Se acceptă și răspunsul că secvența este imposibilă din cauza return. Chiar sti atunci, la un return se putea face tot o verificare branchless.

6.4. Da, secvența este următoarea.

```
r = 0
puteri = {1000000000, 100000000, ..., 100, 10, 1}
index = 10 - lungime(s) + 1
pentru i=0,...,lungime(s)-1
    r = r + puteri[index]*not((s[i] < '0') | (s[i] > '9'))*(s[i] - '0')
    index = index + 1
```