Arhitectura Sistemelor de Calcul (ASC)
Examinarea finală
Varianta 2
(2022 - 2023)

Anul I, Semestrul I
21 ianuarie 2023
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 12 biți. (Fiecare răspuns corect valorează 1 punct)

binar	octal	zecimal	hexazecimal
0001 1111 1001	771	505	0x1F9
0001 1001 0100	624	404	0x194
0000 0111 1011	173	123	0x07B
0001 0100 1101	515	333	0x14D

a binar	a hexa	b binar	b hexa	a+b zecimal	a+b binar	a+b hexa
0000 0111 1011	0x07B	0000 1010 1001	0x0A9	292	0 0001 0010 0100	0x124

a binar	a hexa	b binar	b hexa	axb zecimal	axb binar	axb hexa
0000 0010 1010	0x02A	0000 0000 0111	0x007	294	0 0001 0010 0110	0x0126

Întrebarea 2. (13 puncte)

Completați tabelul de mai jos cu valorile numerice corecte. Toate numerele sunt întregi pe 8 biți. (Fiecare răspuns corect valorează 1 punct)

binar	octal	zecimal	hexazecimal
1110 0110	746 (sau 346)	-26	0xE6
1110 1111	757 (sau 357)	-17	0xEF

a zecimal	a binar	a hexa	b zecimal	b binar	b hexa
-17	1110 1111	0xEF	-7	1111 1001	0xF9

produs axb zecimal	produs axb binar	produs axb hexa
119	0000 0000 0111 0111	0x0077

3.2.	Patterson / Hennessy
3.4.	Ada Lovelace
3.6.	mereu pară
3.8.	1.75 biți
3.10.	64
3.12.	Y
3.14.	NOT(A NOR B)
3.16.	7F80 0000 și FF80 0000
3.18.	Direct Memory Access (DMA)
3.20.	POPCNT
3.22.	RISC
3.24.	mecanismul statistic p-value (se acceptă și media geometrică)
3.26.	mai mare
3.28.	ray tracing
3.30.	Fused-Multiply-Add (FMA)
3.32.	Executable and Linkable Format
3.34.	.text

Întrebarea 3. (17 puncte)

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

3.2. Numiți pe unul dintre autorii cărții PH (cartea de referință generală a cursului) ...

3.4. Conform cursului și suportului de laborator, primul programator este considerat ...

3.6. Avem un număr natural $a \geq 1$ pe N biți, expresia $a\&(a-1)$ este mereu pară, mereu impară sau poate fi pară sau impară în funcție de a? (aici & este AND pe biți)

3.8. Avem patru simboluri A, B, C și D cu probabilitățile de apariție {1/2, 1/4, 1/8, 1/8}. Care este entropia acestor simboluri?

3.10. Dacă avem o funcție logică cu 7 variabile binare atunci dimensiunea maximă a expresiei sale logice (numărul maxim de termeni în sumă) este ...

3.12. Simplificați maxim următoarea expresie logică *X AND (NOT(X) OR Y) OR (X AND Y) OR (NOT(X) AND Y)* (scrieți formula folosind doar AND, OR și NOT) ...

3.14. Scrieți următoarea expresie logică *A OR B* folosind doar NOR și NOT ...

3.16. În formatul IEEE-FP pe 32 de biți, +inf și –inf au reprezentările în hexazecimal ...

3.18. Tehnologia care permite comunicarea directă între memoria principală și memoria de stocare este ...

3.20. În arhitectura x86/x87, cum se numește instrucțiunea care numără câți biți de "1" sunt în reprezentarea binară a unui număr care se află în registrul eax?

3.22. Dacă ne interesează să îmbunătățim performanța per Watt (performanța raportată la cantitate de energie consumată) atunci vom folosi arhitecturi de calcul ...

3.24. Pentru a compara științific timpii de rulare pentru două programe vom folosi ...

3.26. Viteza memoriei cache L1 în comparație cu viteza memoriei cache L3 este ...

3.28. Tehnologia prezentă pe plăcile GPU moderne care permite calcularea în timp real a lumunizității dintr-o scenă de joc se numește ...

3.30. Instrucțiunea assembly x86/x87 care calculează $c \leftarrow c + a \times b$ simultan se numește ...

3.32. Am discutat la curs despre fișiere binare ELF. Acronimul ELF vine de la ...

3.34. Segmentul dintr-un executabil ELF în care găsim codul masină de executat este:

Întrebarea 4. (14 puncte)

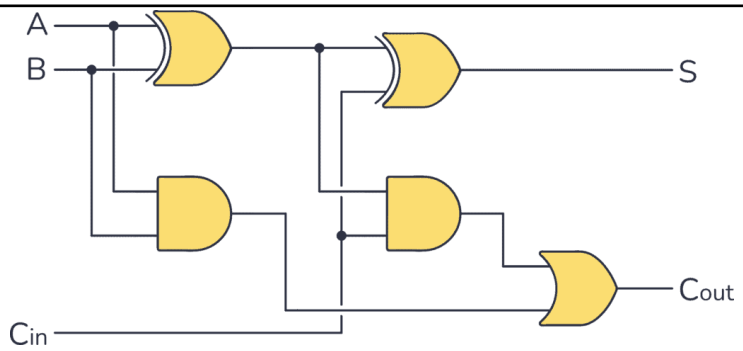
Se dă schema digitală din figura de mai jos. Răspundeți la următoarele întrebări:

4.1. (3 puncte) Care sunt funcțiile logice pentru cele două ieșiri?

4.2. (3 puncte) Ați văzut la curs formule asemănătoare? Este schema de mai jos echivalentă cu o schemă/circuit de la curs?

4.3. (3 puncte) Dacă schema de mai jos este echivalentă cu o schemă de la curs, este schema de mai jos mai eficientă decât cea de la curs?

4.4. (5 puncte) Scrieți funcțiile logice pentru ieșiri, și simplificați-le maxim, dacă știm că $B = 1$. Rezultatul final să fie scris folosind doar AND, OR și NOT.



4.1. $S = A \oplus B \oplus C_{\text{in}}$ și $C_{\text{out}} = AB + (A \oplus B)C_{\text{in}}$.
4.2. Este adunarea binară de 1 bit din Cursul 0x04 dar acolo am avut $S = A \oplus B \oplus C_{\text{in}}$ și $C_{\text{out}} = AB + (A + B)C_{\text{in}}$. Expresiile pentru S sunt identice iar cele două expresii pentru C_{out} sunt echivalente (verificare cu tabel de adevăr sau explicând că este din cauza termenului AB).
4.3. Preferăm această variantă cu $C_{\text{out}} = AB + (A \oplus B)C_{\text{in}}$ pentru că refolosește rezultatul $A \oplus B$ care oricum este obligatoriu necesar pentru S. Varianta din curs calculează $A + B$ în plus și este deci mai ineficientă când vrem ambele valori S și C_{out} (altfel, dacă am vrea să calculăm doar C_{out} am folosi formula din curs, pentru că AND e mai simplu/eficient decât XOR).
4.4. $S = A \oplus 1 \oplus C_{\text{in}} = !(A \oplus C_{\text{in}}) = AC_{\text{in}} + !A!C_{\text{in}}$ și $C_{\text{out}} = A + (A \oplus 1)C_{\text{in}} = A + !AC_{\text{in}} = A + C_{\text{in}}$. Acest calcul este echivalent cu adunarea $a + 11...11$ (a plus N biți de 1, dacă a este pe N biți).

Figura este preluată de la https://www.build-electronic-circuits.com/full-adder/

Întrebarea 5. (16 puncte)

Vi se dă numărul real 6.875. Realizați următoarele cerințe:

5.1. (4 puncte) Să se reprezinte numărul în format IEEE FP pe 32 de biți în hexazecimal.

5.2. (4 puncte) Care este următorul număr în format IEEE FP pe 32 de biți după 6.875? Dați reprezentarea hexazecimală și zecimală.

5.3. (4 puncte) Care este numărul în format IEEE FP pe 32 de biți înainte de 6.875? Dați doar reprezentarea hexazecimală.

5.4. (4 puncte) Care este numărul $\frac{6.875}{2}$ în format IEEE FP pe 32 de biți? Dați reprezentarea hexazecimală și zecimală.

5.1. 0x40DC0000.
5.2. 0x40DC0001 și $6.875 + 2^{-21}$.
5.3. 0x40DBFFFF.
5.4. 0x405C0000 și 3.4375.

Întrebarea 6. (18 puncte)

La curs am vorbit despre legea lui Amdahl care calculează cu cât se îmbunătățește viteza de calcul dacă considerăm calcul paralel cu s core-uri de calcul. În multe situații, dacă adăugăm noi core-uri de calcul atunci sistemul suferă și de o încetinire inițială (asta pentru că noile core-uri de calcul trebuie sincronizate între ele, lucru care necesită ciclii de ceas). Răspundeți la următoarele întrebări și realizați următoarele cerințe:

6.1. (3 puncte) Pentru $s = 4$ și $p = 0.8$ calculați îmbunătățirea vitezei cu Amdahl.

6.2. (5 puncte) Din păcate, când adăugăm un nou core de calcul, plătim și o penalizare în viteza generală de calcul de 5% pentru fiecare core nou adăugat peste cel inițial. Pornind de la formula inițială a lui Amdahl, calculați noul speed-up în această situație pentru un s general (variabilă) pentru programul cu $p = 0.8$.

6.3. (4 puncte) Pentru că avem această penalizare de la 6.2., pentru un număr de core-uri foarte mare este posibil să nu avem nicio îmbunătățire în timpul de calcul. Pentru $p = 0.8$, pentru care s nu avem o îmbunătățire în viteza de calcul?

6.4. (4 puncte) La punctul 6.2. am presupus că de fiecare dată când adăugăm un core nou de calcul timpul general de rulare a programului crește cu 5% (adică viteza crește liniar cu numărul de core-uri). Din păcate, în lumea reală vom plăti o penalizare din ce în ce mai mare pe măsură ce adăugăm core-uri tot mai multe. În aplicații reale, când adăugăm primul core (peste cel inițial) plătim 5% penalizare, pentru două core-uri adăugate plătim 20% penalizare, pentru trei core-uri adăugate plătim 45% ș.a.m.d. Adică avem un cost pătratic de plătit. Pornind de la 6.2., calculați noul speed-up în această situație pentru s general (variabilă) și $p = 0.8$.

6.5. (2 puncte) Pentru $s = 4$ calculați cele două noi speed-up-uri de la 6.2. și 6.4.

6.1. $S = \frac{1}{0.2+\frac{0.8}{4}} = 2.5$.
6.2. $S = \frac{1}{0.2+\frac{0.8}{s}+0.05\times(s-1)} = \frac{20s}{s(s+3)+16}$ (pentru $s = 1$ avem $S = 1$, deci e consistent la fel ca Amdahl original).
6.3. $0.2 + \frac{0.8}{s} + 0.05 \times (s-1) = 1$ care e adevărat când $s = 16$ (și când $s = 1$ dar atunci e cazul unicore, iar pentru $1 < s < 16$ avem speed-up).
6.4. $S = \frac{1}{0.2+\frac{0.8}{s}+0.05\times(s-1)^2} = \frac{20s}{s^3-2s^2+5s+16}$.
6.5. $S = 1.82$ și $S = 1.18$ (ambele mai mici decât 2.5-ul inițial, care nu presupunea nicio penalizare).

Notă: Se acceptă și rezolvarea care implică creșterea de cost cu $(1.05)^{(s-1)}$, adică noua lege să fie $S = \frac{1}{0.2+\frac{0.8}{s}\times(1.05)^{(s-1)}}$. În ambele cazuri ideea este că această penalizare suplimentară se întâmplă doar la numitor și doar la partea paralelizabilă.