

Conform Curs 0x00, sistemul de calcul Fugaku are memorie

- ☐ a. 537212 GB
- ☒ b. 2560000 GB
- ☐ c. alt raspuns
- ☐ d. 5120000 GB
- ☐ e. 7630848 GB
- ☐ f. 5087232 GB



Răspunsul corect este: 5087232 GB

Cine a proiectat masina Diferentiala 2?

- ☐ a. Alan Turing
- ☐ b. Claude Shannon
- ☐ c. George Boole
- ☐ d. Blaise Pascal
- ☐ e. Konrad Zuse
- ☐ f. Gottfried Wilhelm von Leibniz
- ☐ g. Ada Lovelace
- ☒ h. Charles Babbage
- ☐ i. altcineva
- ☐ j. John von Neumann



Răspunsul corect este: Charles Babbage

Avem un sistem de calcul cu 2 registrii pe 4 biti fiecare: eax si ebx. In eax avem valoarea 0xA iar in ebx avem valoarea 0xB. Rezultatul operatiei de adunare  $eax \leftarrow eax + ebx$  este in zecimal (numerele sunt naturale):

- ☐ a. 6
- ☐ b. D
- ☐ c. 1
- ☐ d. 16
- ☐ e. 10
- ☐ f. 17
- ☐ g. F
- ☒ h. 5
- ☐ i. altceva
- ☐ j. E



Răspunsul corect este: 5

Avem un sistem de calcul cu 2 registrii pe 8 biti fiecare: eax si ebx. In eax avem valoarea 0x77 iar in ebx avem valoarea 0x4B. Rezultatul operatiei de adunare  $eax \leftarrow eax + ebx$  este in zecimal (numerele sunt intregi):

- ☐ a. -4
- ☐ b. -60
- ☐ c. altceva
- ☐ d. -16
- ☒ e. -62
- ☐ f. 0
- ☐ g. -57
- ☐ h. -10



Răspunsul corect este: -62

Avem un sistem de calcul cu 2 registrii pe 32 biti fiecare: eax si ebx. In eax avem valoarea 0xF1E2 iar in ebx avem valoarea 0xC3A7. Rezultatul operatiei  $\text{eax} \leftarrow \text{eax OR ebx}$  este in zecimal:

- ☐ a. 61922
- ☐ b. 57005
- ☐ c. altceva
- ☐ d. 43775
- ☒ e. 62439
- ☐ f. 40621
- ☐ g. 50087
- ☐ h. 57007



Răspunsul corect este: 62439

Rezultatul operatiei  $(1101 + 1101) \times (1101 + 1101)$  considerand numere naturale este in zecimal:

- ☐ a. 1111110001
- ☐ b. 1001111011
- ☐ c. 1110111100
- ☐ d. 1010100100
- ☒ e. altceva
- ☐ f. 1110111001



Răspunsul corect este: 1010100100

Entropia unor evenimente care au probabilitatile

- ☐ a. 2
- ☐ b.  $1/3, 1/6, 1/6, 1/6, 1/6]$  este:  
{
- ☐ c. 1
- ☐ d. 2.252
- ☐ e. 2.75
- ☐ f. 0
- ☒ g. altceva



Răspunsul corect este: 2.252

Simplificati expresia  $!(x * z + !x * y + y * z)$ :

- ☐ a.  $x + y + !z$
- ☐ b.  $(x * y) * (x + z) + (y + !z)$
- ☒ c.  $x * !z + !x * !y$
- ☐ d. altceva
- ☐ e.  $x + y$
- ☐ f.  $y + !z$



Răspunsul corect este:  $x * !z + !x * !y$

Fie numarul FP care are reprezentarea hexa 0xc44c0800, atunci valoarea zecimala este:

- ☐ a. -1313.3125
- ☐ b. 1313.3125
- ☐ c. -15872.0
- ☐ d. 15872.0
- ☐ e. 816.125
- ☒ f. -816.125
- ☐ g. altceva



Răspunsul corect este: -816.125



Fie următoarele două instrucțiuni  $\%eax \leftarrow 1$ ,  $\%ebx \leftarrow eax - 8$ , ce fel de hazard apare?

- ☐ a. altceva
- ☐ b. WAR
- ☒ c. RAW
- ☐ d. WAW
- ☐ e. RAR



Răspunsul corect este: RAW

Avem un sistem de calcul cu ierarhia memoriei L1->L2->L3->RAM. Care dintre urmatoarele afirmatii este corecta?

- ☐ a. miss rate-ul creste pe masura ce inaintam (de la L1 la RAM) in ierarhie
- ☐ b. nici una
- ☒ c. timpul de acces creste pe masura ce inaintam (de la L1 la RAM) in ierarhie
- ☐ d. miss rate-ul la L1 este cel mai mic
- ☐ e. miss rate-ul la RAM este de 100%
- ☐ f. timpul de acces la L2 este mai mic decat la L1
- ☐ g. timpul de acces la RAM este cel mai mic



Răspunsul corect este: timpul de acces creste pe masura ce inaintam (de la L1 la RAM) in ierarhie

Avem un program care se poate paraleliza in proportie de 50%. Care este accelerarea necesara pentru partea paralela a programului astfel incat sa avem o accelerare totala a performantei de 2 ori:

- ☐ a. 6
- ☐ b. nu se poate
- ☐ c. 5
- ☐ d. 2
- ☒ e. 4
- ☐ f. 1
- ☐ g. 3



Răspunsul corect este: 3

Ce este adevarat despre setul de instructiuni CISC (in comparatie cu setul de instructiuni RISC)?

- ☐ a. este un set de instructiuni introdus recent in arhitecturi de calcul
- ☒ b. codul sursa scris este mai scurt (ca numar de instructiuni) decat cod sursa RISC
- ☐ c. are instructiuni de dimensiune fixa
- ☐ d. suporta putine metode de adresare
- ☐ e. hardware-ul sistemului de calcul este simplu, software-ul este complicat
- ☐ f. altceva
- ☐ g. operatiile care pot avea loc sunt doar registru-registru



Răspunsul corect este: codul sursa scris este mai scurt (ca numar de instructiuni) decat cod sursa RISC

In 2021, Intel a introdus a 12-a generatie de procesoare din gama lor. Procesorul i7-12700K are numarul de core-uri egal cu

- ☐ a. 8
- ☐ b. 16
- ☐ c. 4
- ☒ d. 12
- ☐ e. altceva
- ☐ f. 2



Răspunsul corect este: 12

Se dau doua numere naturale pe N biti: a si b. Calculam  $(a+b) \times (a+b)$ . Pe maxim cati biti este rezultatul acestei operatii?

- ☐ a.  $2N+1$
- ☐ b.  $N$
- ☐ c.  $N+1$
- ☐ d. altceva
- ☐ e.  $2N$
- ☒ f.  $2(N+1)$



Răspunsul corect este:  $2(N+1)$

Avem un sistem de calcul care are urmatoarele flag-uri: Sign Flag (SF, testeaza daca rezultatul este negativ), Zero Flag (ZF, testeaza daca rezultatul este zero), Overflow Flag (OF, testeaza daca rezultatul unei operatii cu semn este overflow) si Unsigned Overflow - denumit si Carry - Flag (CF, testeaza daca rezultatul unei operatii fara semn este overflow). Consideram instructiunea "jg eticheta" (signed greater - adica mai mare cu semn), cum putem scriere aceasta conditie de salt folosind flag-urile de mai sus?

- ☐ a.  $\text{not}(\text{CF}) \text{ AND } \text{not}(\text{ZF})$
- ☒ b.  $(\text{SF XOR OF}) \text{ OR } \text{ZF}$
- ☐ c. altceva
- ☐ d.  $\text{not}(\text{SF XOR OF})$
- ☐ e.  $\text{CF OR ZF}$
- ☐ f.  $\text{SF XOR OF}$
- ☐ g.  $\text{not}(\text{SF XOR OF}) \text{ AND } \text{not}(\text{ZF})$



Răspunsul corect este:  $\text{not}(\text{SF XOR OF}) \text{ AND } \text{not}(\text{ZF})$

Se dau 4 variable IEEE FP: a, b, c, d. Vrem sa calculam  $\text{valoarea1} = a + b + c$  si  $\text{valoarea2} = b + c + d$ . La compilare, compilatorul observa ca  $(b + c)$  este o operatie care se repeta si in consecinta inlocuiesc codul anterior cu urmatorul:  $\text{temp} = b + c$  si  $\text{valoarea1} = a + \text{temp}$  si  $\text{valoarea2} = \text{temp} + d$ . Ce puteti spune despre  $\text{valoarea1}$  si  $\text{valoarea2}$ ?

- ☐ a. varianta optimizata ca numar de operatii consuma si mai putina memorie (sau mai putini registrii)
- ☐ b. noul calcul este mult mai lent
- ☒ c. nu e garantat ca in varianta optimizata valorile calculate sunt aceleasi
- ☐ d. altceva
- ☐ e. e garantat ca in varianta optimizata valorile calcule sunt aceleasi



Răspunsul corect este: nu e garantat ca in varianta optimizata valorile calculate sunt aceleasi



Alegeti una dintre cele 8 mari idei din arhitectura calculatoarelor conform Patterson & Hennessy si explicati-o pe scurt.

Legea lui Moore spune ca performanta va creste foarte mult pentru dimensiuni ale componentelor din ce in ce in mici, pana cand nu va mai fi rentabila (sau fizic posibil) aceasta imbunatatire. La un moment dat, aceasta evolutie va stagna.

Ce face urmatoarea secventa de cod, compilata cu godbolt x86-64 gcc 11.2, flag -O1?

```
func3(int*, int):  
test esi, esi  
jle .L4  
mov rax, rdi  
lea edx, [rsi-1]  
lea rcx, [rdi+4+rdx*4]  
mov edx, 1  
.L3:  
imul edx, DWORD PTR [rax]  
add rax, 4  
cmp rax, rcx  
jne .L3  
.L1:  
mov eax, edx  
ret  
.L4:  
mov edx, 1  
jmp .L1
```

- ☐ a. altceva
- ☐ b. calculeaza suma elementelor dintr-un vector
- ☐ c. verifica daca un numar dat este prim
- ☒ d. calculeaza produsul elementelor dintr-un vector
- ☐ e. calculeaza maximul
- ☐ f. calculeaza lungimea unui sir de caractere
- ☐ g. calculeaza minimul



Răspunsul corect este: calculeaza produsul elementelor dintr-un vector

Fie programul urmato:

.data

<x>:

00 00 00 00

<formatPrintf>:

00 0a 64 25

.text

<f>:

push ebp

mov ebp,esp

push 0x565561e6

push 0x565561f2

mov eax,0x565561c2

\*\*\*\*\*

jmp eax

mov eax,DWORD PTR [ebp+0x8]

inc eax

mov DWORD PTR [ebp+0x8],eax

pop eax

pop eax

pop ebp

ret

<main>:

push 0x565561fe

```
push 0x565561fe  
call 0x565561ad <f>  
pop  edx  
mov  DWORD PTR ds:0x56559008,0x0  
jmp  0x5655620a <final>  
mov  DWORD PTR ds:0x56559008,0x1  
jmp  0x5655620a <final>  
mov  DWORD PTR ds:0x56559008,0x2  
jmp  0x5655620a <final>  
mov  DWORD PTR ds:0x56559008,0x3  
jmp  0x5655620a <final>  
push DWORD PTR ds:0x56559008  
push 0x5655900c  
call 0xf7e1ede0 <printf>  
pop  eax  
pop  eax  
mov  eax,0x1  
xor  ebx,ebx  
int  0x80
```

Linia cu \* va fi inlocuita cu: add eax,0x5. Ce se va afisa pe ecran?

- ☒ a. Segmentation Fault
- ☐ b. 3
- ☐ c. 1
- ☐ d. 2
- ☐ e. 0



Răspunsul corect este: 0

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

- ☐ a. altceva
- ☐ b. lea
- ☐ c. mov
- ☒ d. popcnt
- ☐ e. lzcnt



Răspunsul corect este: popcnt