POWERLEVELING GUIDE ZA **6KA** PO OS

(VTOR KOLOKVIUM **TEORIJA** EDITION)

мацо

Wake up forsaken, време е да мета**rej**маме за оценка!

1. Управување со меморија (1 - 20lvl)

1)

Question 1
Incomplete answer
Marked out of 31.00
Flag question

Еден компјутерски систем има **256 KB** виртуелна меморија. Големината на физичката меморија е **8 KB**, организирани во рамки со големина од **2048 B**. Времето за вчитување на податок при погодок е **300 наносекунди**, а времето потребно за добивање на податок при промашување е **20 микросекунди**.

One computer system has **256 KB** virtual memory. The physical memory is **8 KB**, organized in frames of size **2048 B**. To read data when hit the system needs **300 nanoseconds**, whereas to read data when miss happens the system needs **20 microseconds**.

Форматот на адресите на овој систем е: / System address format:

Virtual address		Physical address	
Page	Offset	Frame	Offset
2^7	2^11	2^2	2^11

Нека процесорот генерира адреси кои припаѓаат на страниците **7d, 1d, 8, 7d, 63, 8, 5, 3c, 63, 8, 63, 5** и алгоритмот за замена е **First In First Out**. Пополнете ја следната табела, каде во колоната Погодоци треба да наведете 1 или 0 во зависност од тоа дали има погодок или не, а во колоната Рамка треба да ја наведете рамката во која е сместена соодветната страница.

The process generates addresses that belong to the pages **7d, 1d, 8, 7d, 63, 8, 5, 3c, 63, 8, 63, 5** and the algorithm for replacement is First In First Out. Fill the following table by entering 1 or 0 in the Hits column to show whether a hit or miss happened, and enter the frame for that page in the Frame column.

Страница / Page (hex)	Погодоци / Hits (0/1)	Рамка / Frame	
7d	0	0	
1d	0	1	
8	0	2	
7d	1	0	
63	0	3	
8	1	2	
5	0	0	
3c	0	1	
63	1	3	
8	1	2	
63	1	3	
5	1	0	

Вкупното време на извршување на мемориските барања на оваа програма е / Total execution time for this program is 1150 ns.

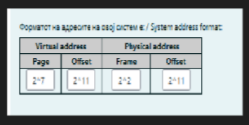
Виртуелната адреса / Vitrual address **28F9** (hex) се мапира во физичката адреса / is mapped into physical address F9 (hex).

Физичката адреса / Physical address **588** (hex) се мапира во виртуелната адреса / is mapped into virtual address 2D88 (hex).

1. Претварање на вредности:

Виртуелна меморија: 256KB = 256 * 1024 = 262 144B
Физичка меморија: 8KB = 8 * 1024 = 8192B
Организирани во рамки: 2048B
Време на погодок: 300ns
Време на промашок: 20μs = 20 * 1000 = 30 000ns

2.



Табела за формат на адреси:

Офсет се пресметува како 2 на степен од големината рамките и се запишува степенот.

2048B = 2^11 = 11 офсет = овој степен се запишува во двата офсет полиња.

Страница полето се пресметува како виртуелната меморија поделена со големината на рамките и се зима 2 на степен од резултатот.

262 144B / 2048B = 128 = 2^7 = овој степен се запишува во полето за страница.

Рамка полето се пресметува на ист начин како страница полето само сега физичката меморија е поделена со големината на рамките.

8192B / 2048B = 4 = 2^2 = овој степен се запишува во полето за рамка

3.

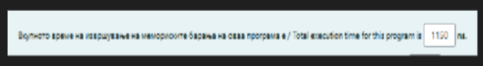
Табела за генерирање на адреси:

БИТНО!
Алгоритам: FIFO (First in First Out)
Погодок: 1
Промашок: 0
Рамки: 4 -> 2^2 од претходната табела, се нула индексират (0 - 3)

7d: 0 = Бидејќи прв пат се генерира, се сместува во рамка 0
1d: 0, се сместува во рамка 1
8: 0, се сместува во рамка 2
7d: 1 = Погодок! Не се прв пат генерира, смести го во рамката што беше прв пат генериран. Во овој случај 7d беше сместен во рамка 0.
63: 0, смести го во рамка 3
8: 1 = Погодок! Не се прв пат генерира, смести го во рамката што беше прв пат генериран. Во вој случај 8 беше сместен во рамка 2.
5: 0, смести го во рамка 0 = Почнува пак од рамка 0 бидејќи имаме само 4 рамки на располагање (0 - 3)
3c: 0, смести го во рамка 1
63: 1, 3
8: 1, 2
63: 1, 3
5: 1, 0

Третата колона е помошна (не се оценува)

4.



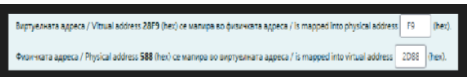
Вкупното време на извршување на мемориските барања на оваа програма е / Total execution time for this program is 1150 ns.

Формула: БрНаПогодок * ВремеНаПогодок + БрНаПромашок * ВремеНаПромашок

Време на погодок: 300ns
Време на промашок: 20μs = 20 * 1000 = 20 000ns
Број на Погодок: 6 = Ги броиме 1 во претходната табела во Погодоци колоната
Број на Промашок: 6 = Ги броиме 0 во претходната табела во Погодоци колоната

Вкупно време = 6 * 300ns + 6 * 20 000ns = 1800ns + 120 000ns = 121 800ns

5.



Виртуелната адреса / Vitrual address 28F9 (hex) се мапира во физичката адреса / is mapped into physical address F9 (hex).

Физичката адреса / Physical address 588 (hex) се мапира во виртуелната адреса / is mapped into virtual address 2D88 (hex).

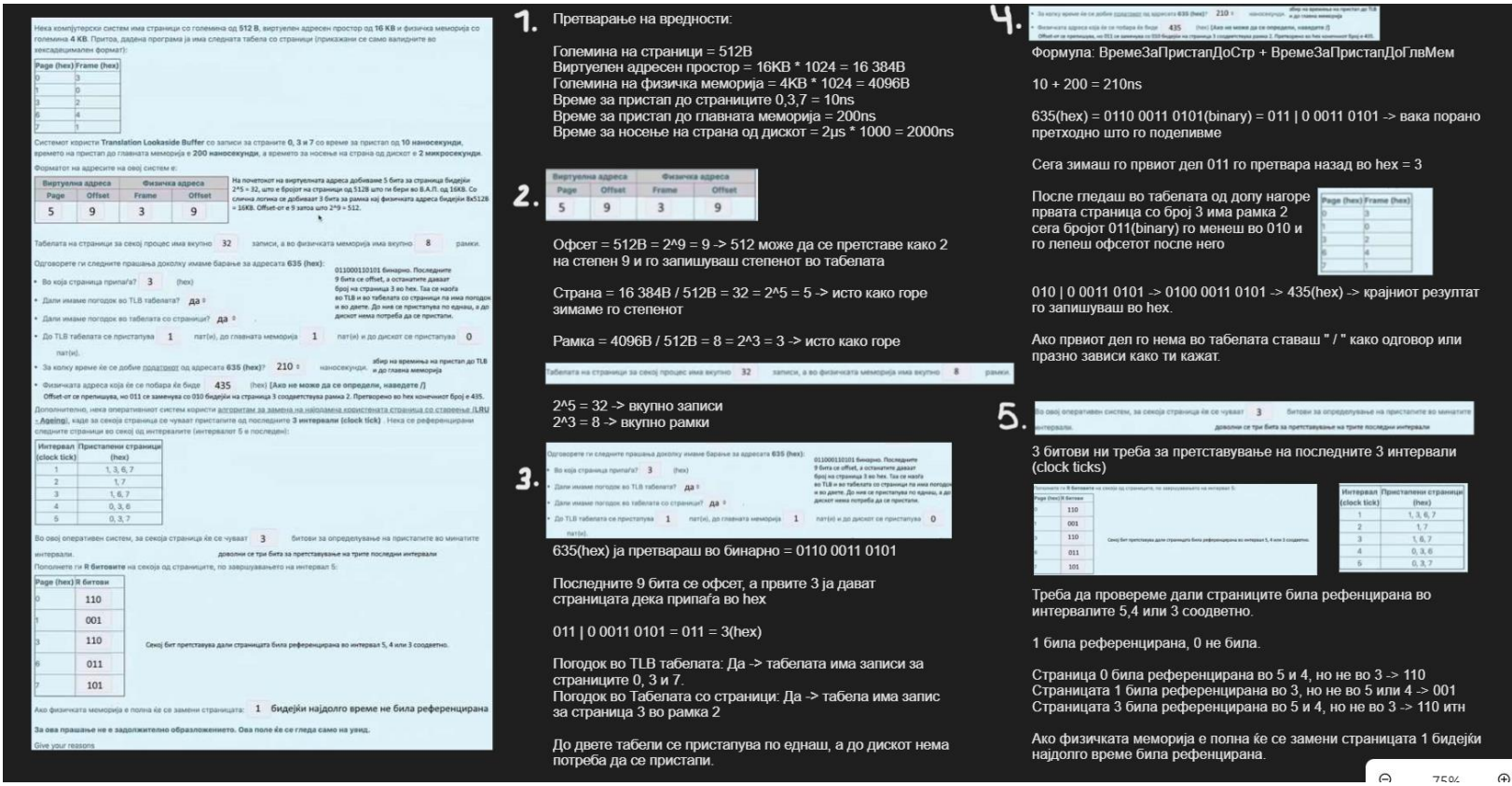
Мапирање виртуелна во физичка:

Виртуелната адреса: 28F9 (hex) = 0010 1000 1111 1001 (binary)
Ги броиш битовите во бинарната адреса = 16 бита
Треба да се 18 бита, фалат уште 2.
18те бита ги одредуваме со првата табела од задачата. Ги собираме степените во Виртуелната адреса колона = 2^7 + 2^11 = 18.
Бидејќи ни фалат уште 2 лепеме 0 на почетокот од бинарниот број.
Модифицираната бинарна адреса = 00 0010 1000 1111 1001
Следно ја делеш адресата на два дела. Првиот дел ќе е 7 бита, а вториот дел ќе е 11 бита.
Поделбата ја одредуваме според степените во првата табела 2^7 и 2^11.
Поделената адреса: 0000101 | 00011111001
Првиот дел со 7 бита во декадно е 5.
Во втората табела тоест таа што ни е за генерирање на адреси 5 има рамка 0 (гледаш од долу па нагоре).
Вредноста на првиот дел го менеш со 00 (2 бита за големината на рамката на физичката адреса).
00 | 00011111001 -> 0000011111001 -> F9 = Го претвараш назад во hex.

Мапирање физичка во виртуелна:(исто како претходно само обратно)

Физичка адреса: 588 (hex) = 0101 1000 1000 (binary)
Ги броиш битовите во бинарната адреса = 12 бита
Треба да се 13 бита, фале уште 1.
13те бита ги одредуваме со првата табела од задачата. Ги собираме степените во Физичката адреса колона = 2^2 + 2^11 = 13.
Бидејќи ни фале уште 1 лепеме 0 на почетокот од бинарниот број.
Модифицираната бинарна адреса = 0 0101 1000 1000
Следно ја делеш адресата на два дела. Првиот дел ќе е 2 бита, а вториот дел ќе е 11 бита.
Поделбата ја одредуваме според степените во првата табела 2^2 и 2^11.
Поделената адреса: 00 | 10110001000
Првиот дел со 2 бита во декадно е 0.
БИТНО! Сега гледаш кај колоната за рамки а не кај страници!!!
Во втората табела тоест таа што ни е за генерирање на адреси 0 има страница 5 (гледаш од долу па нагоре).
Вредноста на првиот дел го менеш со 00101. (Уште една 0 ставеш на почетокот за да се совпадне големината)
00101 | 10110001000 -> 0010110110001000 -> 2D88 = Го претвараш назад во hex

2)



1. Претварање на вредности:

Големина на страници = 512B
Виртуелен адресен простор = 16KB * 1024 = 16 384B
Големина на физичка меморија = 4KB * 1024 = 4096B
Време за пристап до страниците 0,3,7 = 10ns
Време за пристап до главната меморија = 200ns
Време за носење на страна од дискот = 2μs * 1000 = 2000ns

2. Офсет = 512B = 2^9 = 9 -> 512 може да се претставе како 2 на степен 9 и го запишуваш степенот во табелата

Страна = 16 384B / 512B = 32 = 2^5 = 5 -> исто како горе зимаме го степенот

Рамка = 4096B / 512B = 8 = 2^3 = 3 -> исто како горе

2^5 = 32 -> вкупно записи
2^3 = 8 -> вкупно рамки

3. 635(hex) ја претвараш во бинарно = 0110 0011 0101

Последните 9 бита се офсет, а првите 3 ја даваат страницата дека припаѓа во hex

011 | 0 0011 0101 = 011 = 3(hex)

Погодок во TLB табелата: Да -> табелата има записи за страниците 0, 3 и 7.
Погодок во Табелата со страници: Да -> табела има запис за страница 3 во рамка 2

До двете табели се пристапува по еднаш, а до дискот нема потреба да се пристапи.

4. Формула: ВремеЗаПристапДоСтр + ВремеЗаПристапДоГлвМем

10 + 200 = 210ns

635(hex) = 0110 0011 0101(binary) = 011 | 0 0011 0101 -> вака порано претходно што го поделивме

Сега зимаш го првиот дел 011 го претвара назад во hex = 3

После гледаш во табелата од долу нагоре првата страница со број 3 има рамка 2 сега бројот 011(binary) го менеш во 010 и го лепеш офсетот после него

010 | 0 0011 0101 -> 0100 0011 0101 -> 435(hex) -> крајниот резултат го запишуваш во hex.

Ако првиот дел го нема во табелата ставаш " / " како одговор или празно зависи како ти кажат.

5. 3 битови ни треба за претставување на последните 3 интервали (clock ticks)

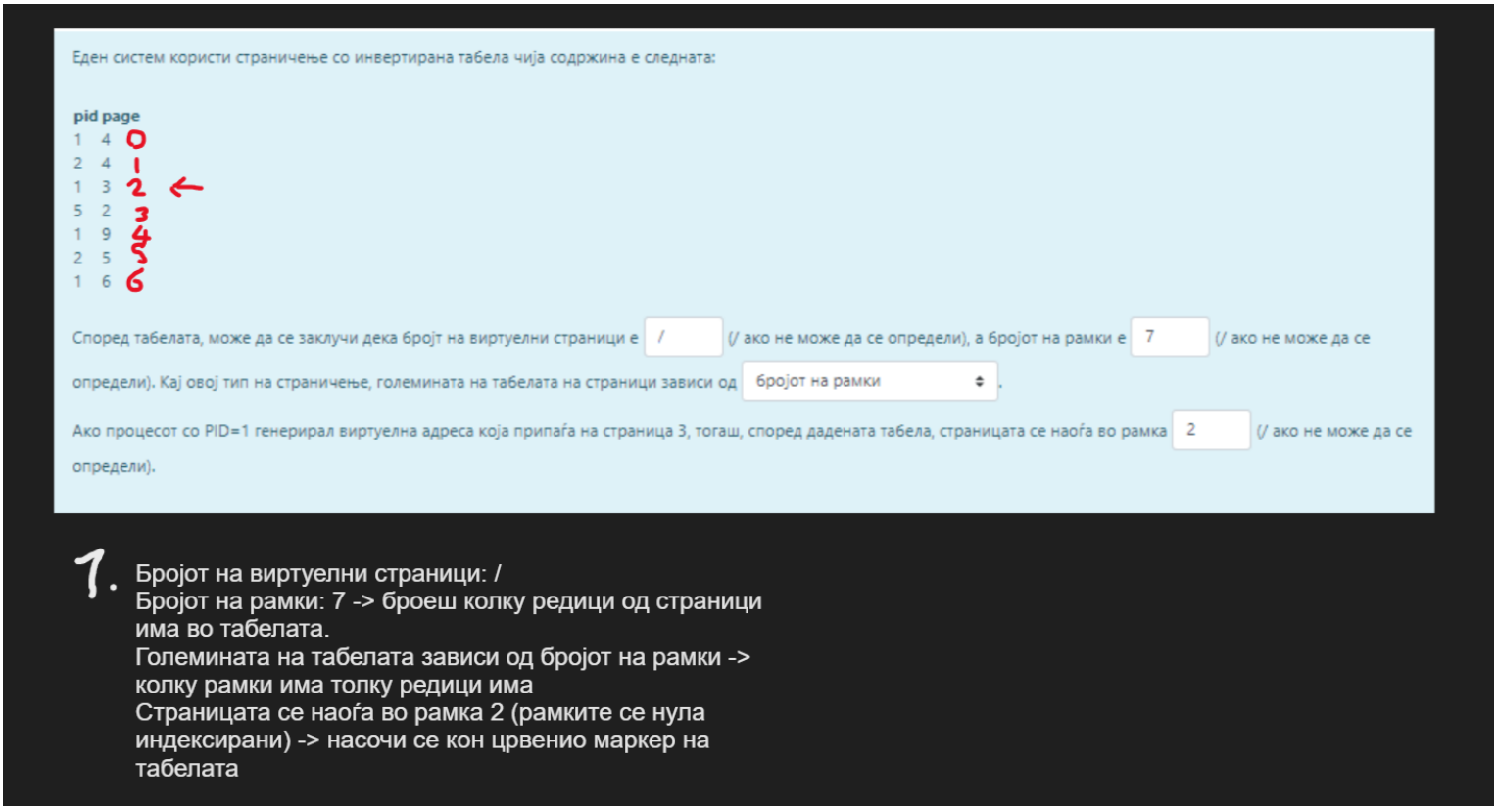
Треба да провериме дали страниците била рефенцирана во интервалите 5,4 или 3 соодветно.

1 била референцирана, 0 не била.

Страница 0 била референцирана во 5 и 4, но не во 3 -> 110
Страницата 1 била референцирана во 3, но не во 5 или 4 -> 001
Страницата 3 била референцирана во 5 и 4, но не во 3 -> 110 итн

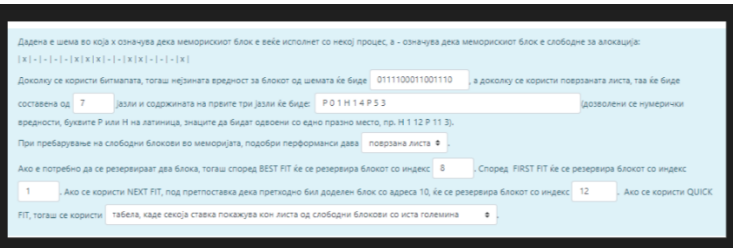
Ако физичката меморија е полна ќе се замени страницата 1 бидејќи најдолго време била рефенцирана.

3)



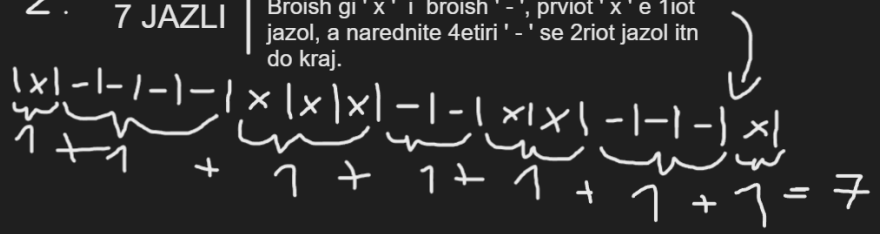
1. Бројот на виртуелни страници: /
Бројот на рамки: 7 -> броеш колку редици од страници има во табелата.
Големината на табелата зависи од бројот на рамки -> колку рамки има толку редици има
Страницата се наоѓа во рамка 2 (рамките се нула индексирани) -> насочи се кон црвенио маркер на табелата

4)



1. 0 1111 000 11 00 111 0 | ' x ' = 0, ' - ' = 1, za sekoj ' x ' i za sekoj ' 0 ' gi menes so 0 i 1 soodvetno.

2. 7 JAZLI | Broish gi ' x ' i broish ' - ', prviot ' x ' e 1iot jazol, a narednite 4etiri ' - ' se 2riot jazol itn do kraj.



3. P = proces = ' x ', H = hole = ' - '
treba da se napisat prvite 3 jazli -> ova e bitno, citaj ubavo pred da gi pisuvas
P 0 1 = prviot jazol x e proces, pocnuva kaj indeks 0 i ima dolzina 1
H 1 4 = posle imame jazol od ' - ' toa e hole, pocnuva kaj indeks 1 i ima dolzina od 4
P 5 3 = tretiot jazol x e proces, pocnuva kaj indeks 5 i ima dolzina od 3
kraen rezultat: P 0 1 H 1 4 P 5 3 -> site gi razdvojuvas so prazno mesto

4. Linked List

5. Bitno: Sekoj jazol so ' - ' oznacuva jazol so slobodni memoriski blokovi, a sekoj poseben ' - ' e poseben sloboden memoriski blok

Treba da rezerviras 2 memoriski bloka.

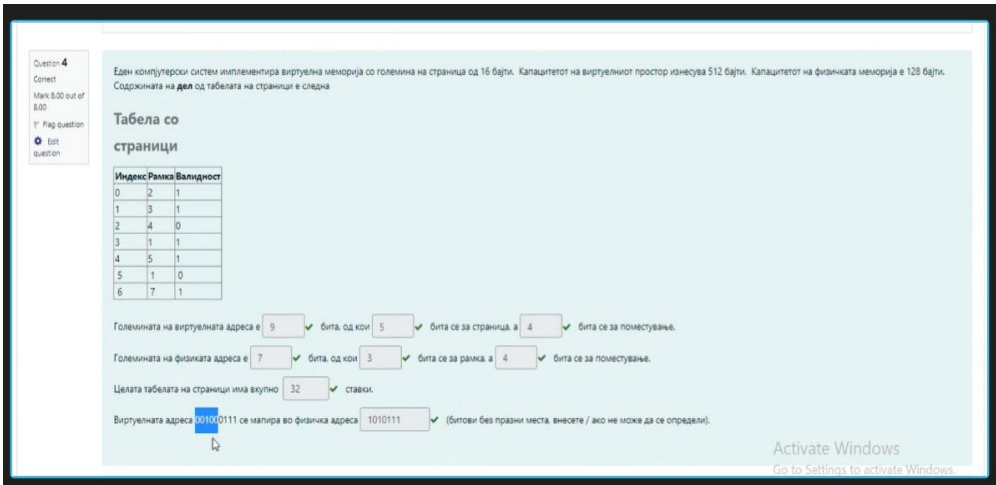
BEST FIT traze najdobriot prv jazol so se sovpagja | pocnuvas od indeks 0 toa e prviot jazol x. Iteriras niz jazlite se dodeka ne dojdes do indeks 8. Tuka ima dva ' - ' so e tocna brojka na memoriski bloka so ni treba da rezervirame

FIRST FIT traze prviot jazol kade mozes da gi smestes memoriskite bloka | pocnuvas od 0 toa e prvio jazol x. Iteriras niz jazlite se dodeka ne dojdes do indeks 1. Tuka ima 4etiri ' - ' so e prviot jazol so dovolno slobodni mesta za da se smestat memoriskite bloka

NEXT FIT cuva istorija od poslednata rezervacija vo ovoj slucaj poslednata bila na indeks 10, i trazes first fit posle taj indeks toest indeks 12

QUICK FIT mora da ima tabela kade sekoja stavka pokazuva kon lista od slobodni blokovi so ista golemina

5)



1. Големината на виртуелната адреса: 512B = 2^9
Бита за страница: 512B / 16B = 32 = 2^5
Бита за поместување: 9 - 5 = 4

Големината на физичката адреса: 128B = 2^7
Бита за рамка: 128B / 16B = 8 = 2^3
Бита за поместување = 7 - 3 = 4

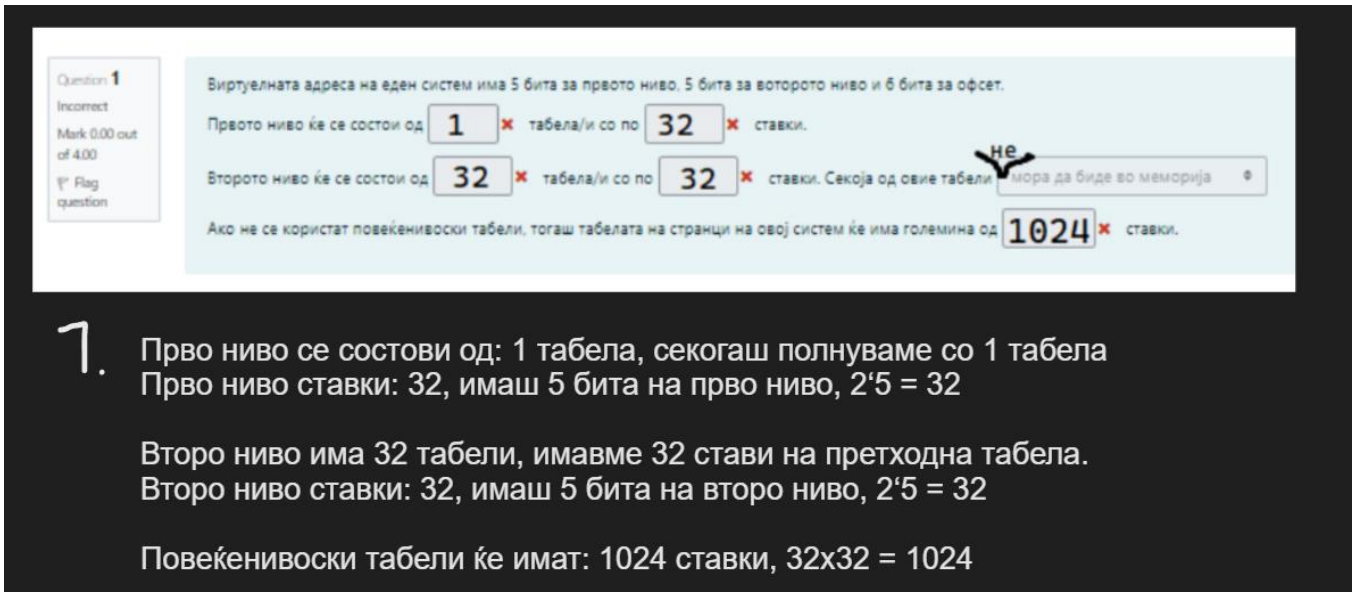
Целата табела на страници
има вкупно ставки: 512B / 16B = 32

001000111 ја поделеш на два дела, првиот дел ќе е 5 бита, а вториот 4

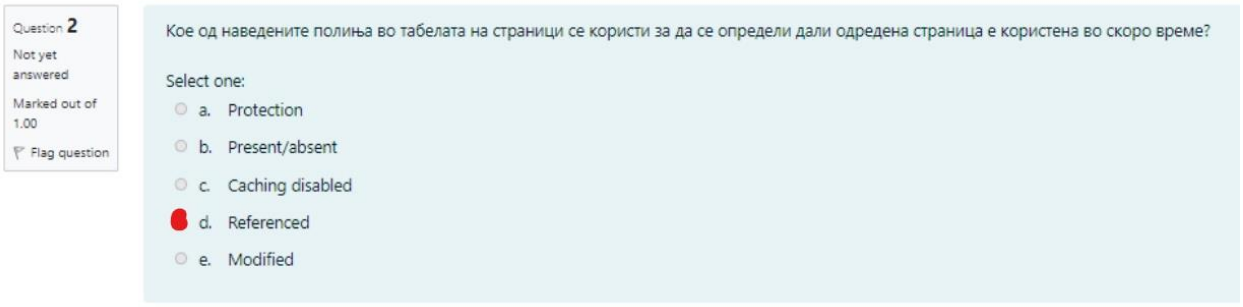
00100 | 0111 -> 00100 -> 4 -> го тражеш во индекс колоната 4 се мапира за 5 -> 00100 = 0101

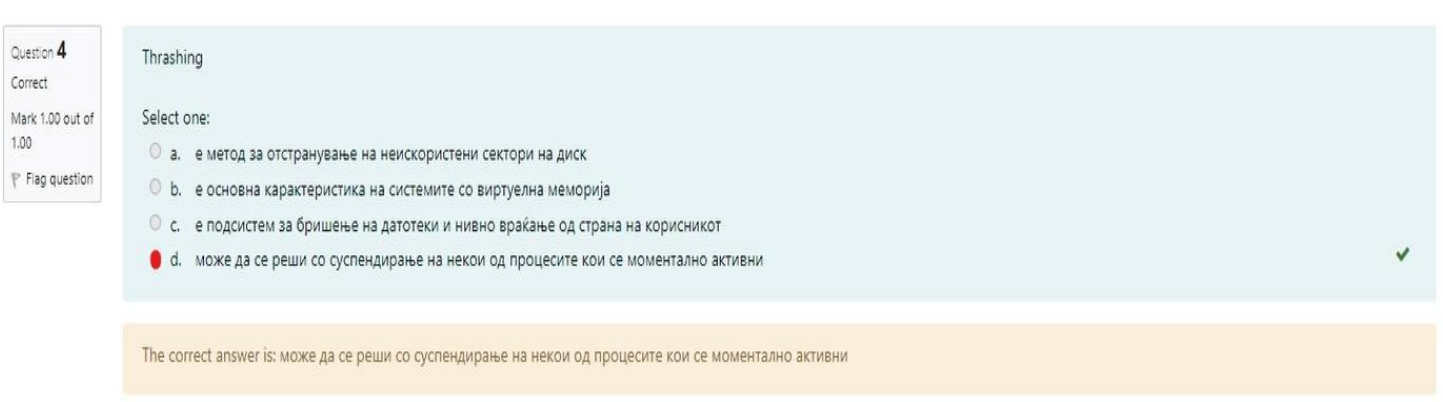
0101 | 0111 -> 01010111 -> 57(hex)

6)



7) Заокружување





Question 4
Not answered
Marked out of 2.00
Flag question

Кај алгоритмот Clock Page Replacement, во моментот кога се случува грешка (page fault), стрелката покажува на страница со R=1. Што ќе се случи?

Select one:

- R ќе биде поставен на 0 и стрелката ќе биде поместена на следната позиција
- Страницата ќе биде исфрлена и стрелката ќе остане на истата позиција
- Страницата ќе биде исфрлена и стрелката ќе се помести на следната позиција
- R ќе биде поставен на 1 и стрелката ќе биде поместена на следната позиција

Question 5
Partially correct
Mark 0.40 out of 2.00
Flag question

Зголемување на големината на стрницата ќе предизвика:

Select one or more:

- Намалување на табелата на страници
- Зголемување на TLB
- Намалуање на TLB
- Зголемување на табелата на страници
- Зголемување на внатрешна фрагментација
- Намалување на внатрешна фрагментација

Кои од наведените искази се точни?

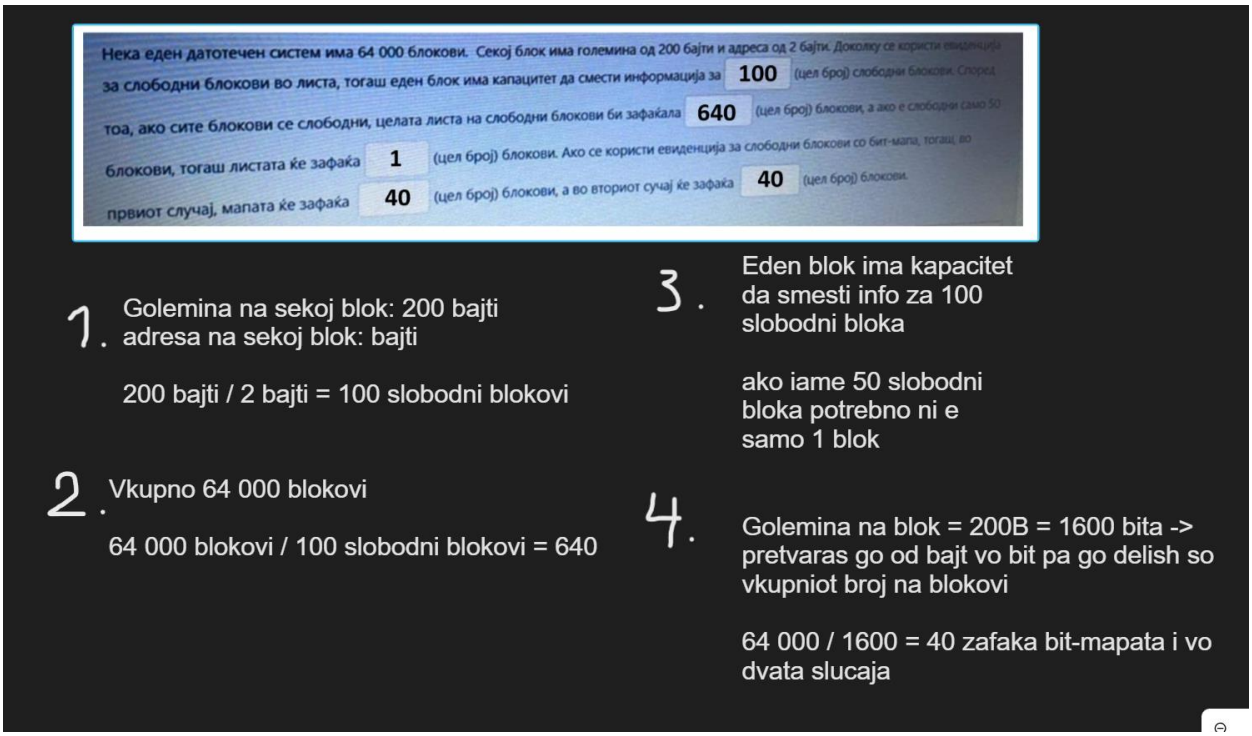
Select one or more:

- a. За да се најде ставка во TLB, мора да се споредуваат сите негови ставки поединечно ✔
- b. Покажувач кон табелата на страници се чува во PCB ✔
- c. Табелата на страници се чува во PCB
- d. Користење на TLB го зголемува времето на мапирање виртуелна во физичка адреса
- e. TLB ги содржи сите ставки од табелата на страници

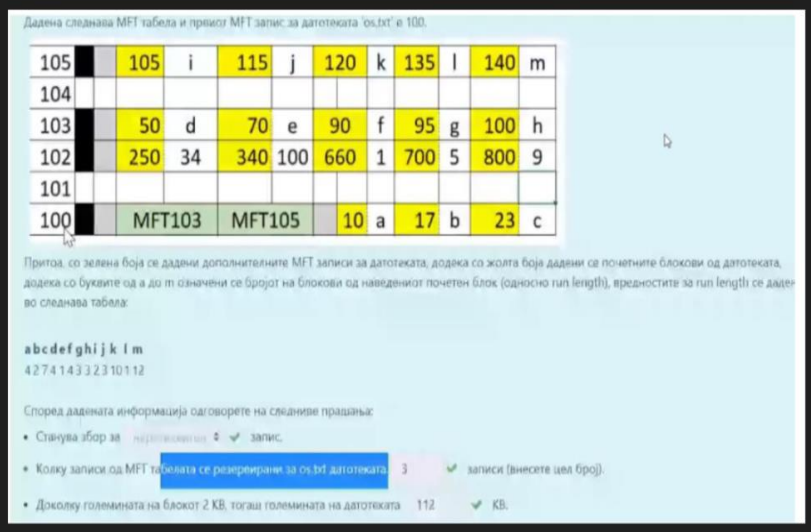
Thrashing настанува кога големината на физичката меморија е [помала] од бројот на страници кои ги користата процесите. Во овој случај, искористеноста на CPU [се намалува]. Наједноставно решение на проблемот е [убивање на процес].

1. Датотечни Системи (21 - 40lvl)

1)



2)



1. Bitno:
Rezidenten zapis: ako ja sobere datotekata samo vo eden MFT zapis.
Nerezidenten zapis: ako ja sobere vo dva ili povekje.
prviot zapis e vo 100.

Gledaj kaj redica 100 vo tabelata, ima uste 2 MFT zapisi oboeni zeleni.

Vkupno MFT zapisi: 3
Zapisot e nerezidenten.

gledas koi zapisi se za os.txt -> tie gi sodrzat bukvite od a do m.

100, 103 i 105. -> na mestoto od goleminite imame bukvi

sega gi sobiras goleminite sto ni se dadeni vo vtorata tabela ispod bukvite.

4 + 2 + 7 + 4 +1 + 4 + 3 + 3 + 2 + 3 + 10 + 1 + 12 = 56

Zbirot go mnozes: 56 * 2KB = 112KB

3)

Датотеката **/home/teacher/exam.txt** има i-node со број 9701.
Процесот со PID=80 го извршува следниот код:

```
int fd1=open("/home/teacher/exam.txt",O_RDONLY); //open for reading
read(fd1,buffer1,150); //read 150 bytes in buffer1
int fd2=open("/home/teacher/exam.txt",O_RDONLY);
read(fd2, buffer2, 250); //read 250 bytes in buffer2
```

Процесот со PID=81 го извршува следниот код:

```
int fd3=open("/home/teacher/exam.txt",O_RDONLY); //open for writing
read(fd3, buffer3, 300); //read 300 bytes in buffer2
```

Процесот со PID=90 го извршува следниот код:

```
ln /home/teacher/exam.txt /home/assistant/os.txt //create hard  link (ln source_file link)
ln /home/teacher/exam.txt /home/dean/os.txt //create hard link
```

Пополнете ја содржината на дадените табели по извршување на кодот од процесите. Ако некои од полињата не се потребни, оставете ги празни.

Process table

PID=80

Index	Value
0	/
1	/
...	
7	
8	
9	

PID=81

Index	Value
0	/
1	/
...	
5	
6	
7	

PID=90

Index	Value
0	/
1	/
2	/
3	
4	
5	

System File Table

Address	Name	Value
2000	mode	
	offset	
	ref.count	
	i-node ptr.	
	...	
2040	mode	
	offset	
	ref.count	
	i-node ptr.	
2080	mode	
	offset	
	ref.count	
	i-node ptr.	
	...	
20C0	mode	
	offset	
	ref.count	
	i-node ptr.	

I-node Table

Address	Name	Value
B000	number	9701
	link count	
	ref. count	
	...	
B400	number	9639
	link count	
	ref. count	
	...	
B800	number	6760
	link count	
	ref. count	
	...	
BC00	number	8385
	link count	
	ref. count	
	...	

1. TABELITE:

okej, odeme po red so procesite

exam.txt ima i-node 9701.

Proces PID = 80 -> zabelezuvame deka ovoj proces samo go otvara faljot exam.txt za citanje (read), cita 150 bytes vo buffer1, pak go otvara fajlot po vtor pat samo za citanje i cita 250 bytes vo buffer2.

Sega gledame vo i-node tabelata, adresata za i-node na exam.txt e B000.

Sledno gledame na System File tabelata kaj adresa 2000 i 2040.

Za prvoto citanje:
mode: r (read) -> ova go odredivme na pocetokot
offset: 150 -> ova go odreduvame kaj linijata: read(fd1,buffer1,150)
ref count: 1 -> ovde sekogas e 1!
i-node ptr: B000 -> ova adresa ja odredivme vo i-node tabelata.

Za vtoroto citanje:
mode: r (read) -> ova go odredivme na pocetokot
offset: 250 -> ova go odreduvame kaj linijata: read(fd2,buffer2, 250)
ref count: 1 -> ovde sekogas e 1!
i-node ptr: B000 -> ova adresa ja odredivme vo i-node tabelata.

Adresite 2000 i 2040 gi zapisuvame vo proces tabelata so PID=80 edno po drugo(BITNO!).

Proces PID=81 -> zabelezuvame deka ovoj proces samo go otvara exam.txt za citanje i cita 300 bytes vo buffer3.

Sledno gledame na System File tabelata kaj adresa 2080.

Zapisuvame:
mode: r (read) -> ova go odredivme na pocetokot
offset: 300 -> ova go odreduvame kaj linijata: read(fd3,buffer3,300)
ref count: 1 -> ovde sekogas e 1!
i-node ptr: B000 -> ova adresa ja odredivme vo i-node tabelata.

Adresata 2080 ja zapisuvame vo proces tabelata so PID=81.

Proces PID=90, ovoj proces prave 2 hard links 😋🥵

Bitno:
ln -> hard link
ln -s -> soft link
prviot argument e fajlot kon kogo se POKAZUVA
vtoriot e onoj na koj mu go menuvame POKAZUVACOT

Gledame prva linija:

assistant/os.txt POKAZUVA sega kon i nodeot na exam.txt -> bidejki e vtor argument go smenivme pokazuvacot.

bidejki e HARD link sega kaj i-node za exam.txt link count se inkrementira za 1 (samo vazi za HARD link)

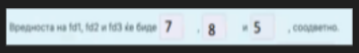
Gledame vtora linija:

Isto taka prave hard link kako prvata linija, inkrementiraj link count za 1.

Link count: 2 (sto inkrementiravme) + 1 (od samiot file) = 3 -> ova go zapisuvas vo i-node tabelata.

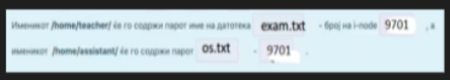
Ref count: 3 -> ova brojka dovagja od so go otvaravme fajlot 3 pati. ova go zapisuvas vo i-node tabelata.

Drugite polinja vo tabelata prazni gi ostavas ili " / " zavisi kako ti kazat da gi ostaves

2. 

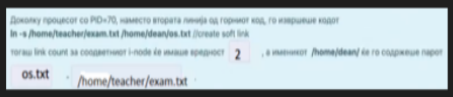
Gledas vo proces tabelata:

fd1 e adresa 2000, indeksot na ovaa adresa e 7
fd2 e adresa 2040, indeksot na ovaa adresa e 8
fd3 e adresa 2080, indeksto na ovaa adresa e 5

3. 

imenikot home/teacher go sodrzi exam.txt vo nego.
i-node za nego ni e dadeno 9701.

imenikot home/assistant go sodrzi os.txt vo nego.
i-node za nego e isto 9701.

4. 

ovde pisuva deka proces so pid=70, a takov ne postoe. Pa pretpostavuvam deka pid=90 bidejki tamo praveme links.

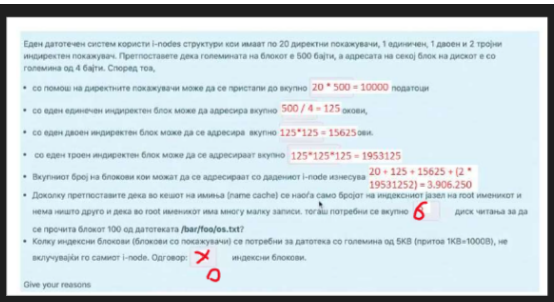
Ja gledame vtorata linija kaj pid=90:
sega treba da prave soft link namesto hard.

soft link ne se broi vo link count.
soft link e samo apsolutna pateka kon originalniot file, a hard link e celosna kopija koja pokazuva kon istiot i-node

soft link count = 1 (abs path) + 1 (od samiot file) = 2

imenikot home/dean ke go sodrzi os.txt i absolutnata pateka -> home/teacher/exam.txt

4)



1. Imas 20 direktni pokazuvaci i goleminata na blokot e 500 bajti.

Moze da se pristapi do: 20 * 500 = 10 000 podatoci

2. Golemina na blok: 500 bajti
Adresata na sekoj blok: 4 bajti

So eden EDINECEN indirekten blok moze da se adresira vkupno: 500 bajti / 4 bajti = 125

3. EDINECEN indirekten blok: 125 blokovi

So eden DVOEN indirekten blok moze da se adresira vkupno 125^2 = 15 625 blokovi

So eden TROEN indirekten blok moze da se adresira vkupno 125^3 = 1 953 125 blokovi

4. Vkupniot broj na blokovi e sumata na site ovie vrednosti plus direktnite pokazuvaci.

BITNO: imame 2 TROJNI indirektni blokovi, znaci se mnozat po 2 pred da se sumirat.

20 + 125 + 15 625 + (2 * 1 953 125)
= 3 906 205

5. Treba da se procita blokot 100 od datotekata /bar/foo/os.txt

Prvo gledame dali spagja vo direktnite blokovi. Ne spagja bidejki tie se od 0-19, a treba 100 da citame.

Vtoro gledame dali spagjame vo indirektnite EDINICEN blokovi. DA, spagjame bidejki se adresirat 125 bloka a treba da citame 100, ako NE spagjavme preminuvame na DVOJNITE da provereme itn.

Treto sumirame:
1 citanje za root ili " / " +
1 citanje za bar +
1 citanje za foo +
1 citanje za os.txt +
1 citanje za ind EDINECEN +
1 citanje za data = 6 disk citanjta

6. Imame 5KB = 5000B datoteka, mozeme da ja smesteme vo 10 direktni blokovi, bidejki 1 e so golemina 500B.

Cim se smestuva vo direktni ni treba 0 indeksni blokovi.

Ako ne se smestuvase vo direktnite i otide vo indirektnite togas broeme kolku bloka ni trebat.

1. Дискови & SSD (41 - 50lvl)

1)

Претпоставете дека секој од системите RAID 1 и RAID 5 кои ги поседува една компанија имаат по 8 диска и сите дискови имаат иста големина. Одговорете:

Колку место за складирање на податоци е достапно за секоја од конфигурациите (внесете цел број)?

- **RAID 1** 4 диска **RAID 5** 7 диска.

За секоја од конфигурациите, кој е минималниот број на оштетени дискови за да **МОЖЕ** да има губење на податоците?

- **RAID 1** 2 диска **RAID 5** 2 диска.

За секоја од конфигурациите, кој е минималниот број на оштетени дискови за СИГУРНО да има губење на податоци?

- **RAID 1** 5 диска **RAID 5** 2 диска.

Give your reasons

1. RAID(Shadow Legends)

Bitno:
n: 8 diska so ista golemina

Prvo prasanje:
Raid1 = n / 2 = 4 -> mesto za skladiranje e polovina od dostapnite diskovi
Raid5 = 7 -> n - 1 -> bidejki se trose 1/n za XOR blokovi za parnost sto go namaluva prostorot za 1 disk

Vtoro prasanje:
Raid1 = 2 -> se nastanuva zaguba ako eden disk i negoviot soodveten backup se ostetat.
Raid5 = 2 -> moze da gi presmeta i vrati zagubenite podatoci od samo eden disk bidejki moze da presmeta samo edinicna parnost.

Treto prasanje:
Raid1 = 5 -> (n/2) + 1 diska za SIGURNO ke nastane zaguba.
Raid5 = 2 -> isto kako vtoroto prasanje.

2)

Time le

Дадена е состојбата на дел од SSD во кој се сместени податоци за датотеките 1, 2, 3 и 4. Секое непразно поле ја означува датотеката чиишто податоци ги содржи. Која ќе биде содржината секое поле од блокот C ако се избрише датотеката 4? Ако има потреба, празните полиња да се означат со вредност 0.

Block A

1	1	1
1	1	1
1	2	2
2	2	2

Block B

2	2	2
2	2	2
2	3	3
3	3	3

Block C

Block D

3	3	3
3	3	3
3	3	4
4	4	4

Block C

3	3	3
3	3	3
3	3	0
0	0	0

3) Заокружување

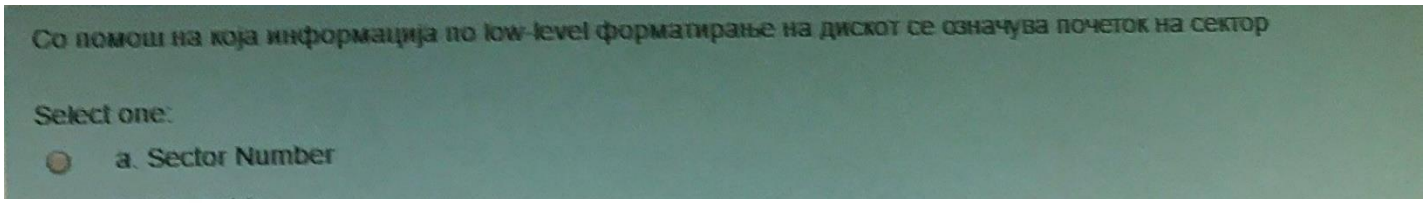
Кое од следниве правила важи кај SSD?

Select one:

- a. Читањето е 10 пати поскапо од бришењето, бришењето е 10 пати поскапо од запишувањето
- b. Читањето е 10 пати поскапо од запишувањето, бришењето е 10 пати поскапо од читањето
- c. Запишувањето е 10 пати поскапо од бришењето, бришењето е 10 пати поскапо од читањето
- d. Запишувањето е 10 пати поскапо од читањето, бришењето е 10 пати поскапо од запишувањето

Clear my choice

Give your reasons



Question 4

Incorrect

Mark 0.00 out of 2.00

Flag question

Edit question

Кај кој тип на RAID е најлесно опоравувањето и враќањето на податоци по пад на диск?.

Select one or more:

- 1
- 2 ✖
- 3
- 4 ✖
- 5
- 10

Your answer is incorrect.

The correct answers are: 1, 10

Question 2

Not yet answered

Marked out of 1.00

Flag question

Кој го вчитува Master Boot Record-от кај еден компјутерски систем?

Select one:

- a. Соодветниот индексен јазел
- b. NTFS партицијата
- c. BIOS-от
- d. Јадрото на оперативниот систем