# Operacijski sustavi - 2. kolokvij

## 1. Princip izrade programa

Prvo se otvori editor i napiše se izvorni (source) kod; npr.:

glavni.c

```
#include "zaglavlje.h"
void main(){
    int x, y, rez;
    scanf("%i,%i", &x, &y);
    rez = zbroji(x, y);
    printf("%i", rez);
}
```

zbroji.c

```
int zbroji(int a, int b)
{return a+b;}
```

↳ provjera da je kod ispravno napisan

zaglavlje.h

```
#include <stdio.h>
int zbroji(int a, int b);
```

Zatim se prevodi (compile) izvorni kod u binarni i kreira se tablica simbola. Na ovaj način kompajler zna kako su deklarirane sve korištene funkcije u programu => cpp glavni.c glavni.o . U datoteci glavni.o (objekt) nalazi se tablica simbola i binarni kod. U tablici simbola nalaze se sve funkcije i globalne varijable i služe linkeru za povezivanje

tablica => | name | value | CLASS | TYPE | SIZE | LINE | SECTION |
| --- | --- | --- | --- | --- | --- | --- |
| main | 00000000 | T | funkc. | | | .text |

↳ funkcija definirana, da je bilo 0 => nije definirana ↳ linker je mora pronaći

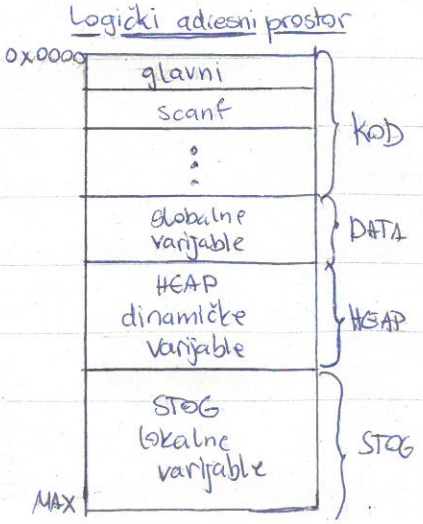
Postupak se ponavlja i s cpp zbroji.c zbroji.o

Funkcije scanf i printf su prethodno kompajlirane i nalaze se u standardnoj biblioteci lib (direktorij kompajlera), zatim slijedi postupak povezivanja (Link) svih komponenti u 1 cjelinu: link glavni.o + zbroji.o + lib, program.exe

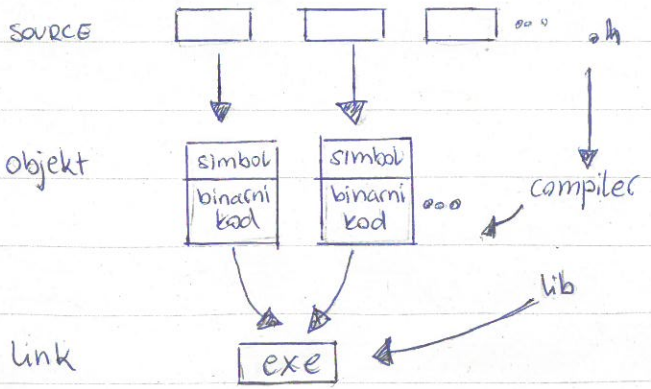
moduli koji tvore cjelinu; program u kojem su povezani moduli

Nakon procesa kompajliranja i linkanja generira se kod i podaci koji započinju od adrese 0 do MAX

Logički adresni prostor




grafički prikaz principa izrade programa

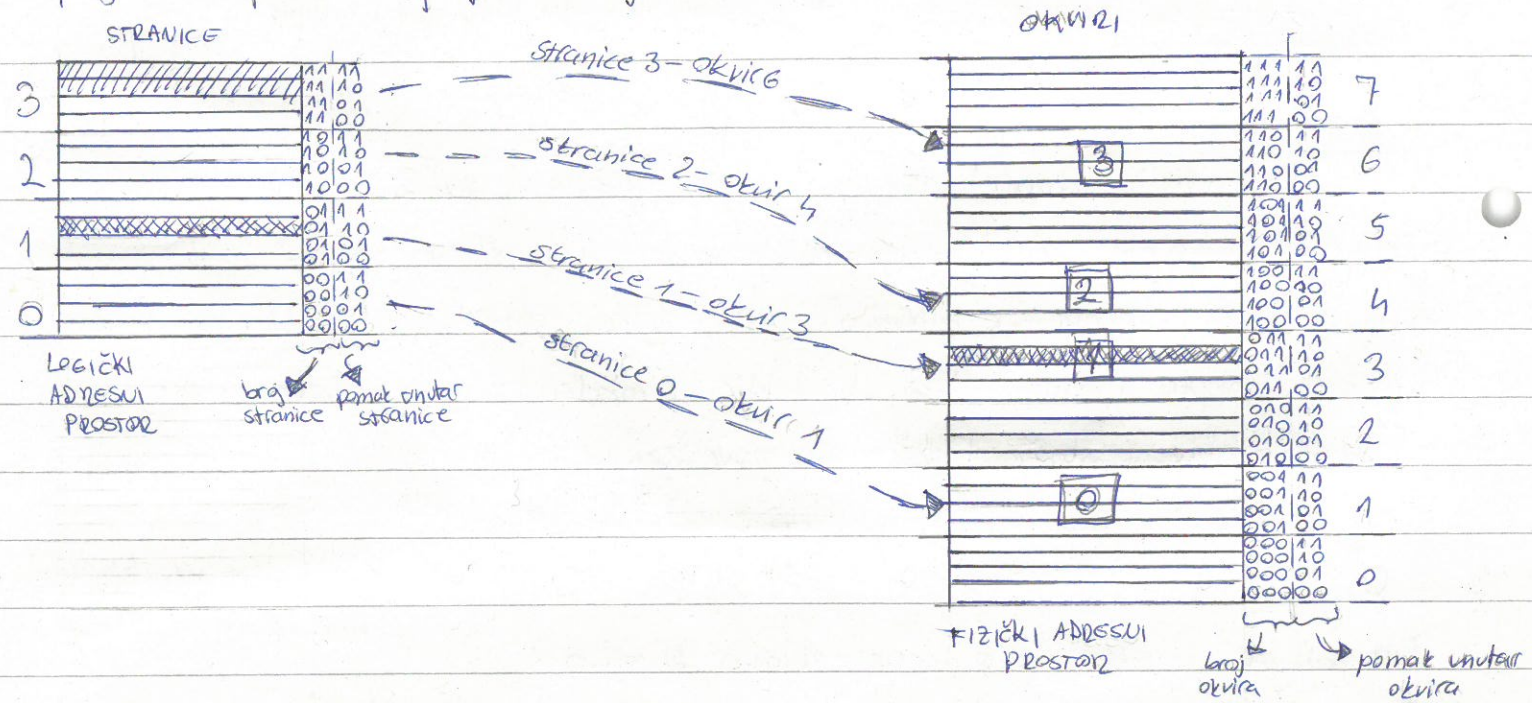


TREBA ZNATI SAM RJEŠITI AKO PROFESOR ZADA DRUGAČIJI BROJ LOKACIJA
(to on radi na satu)

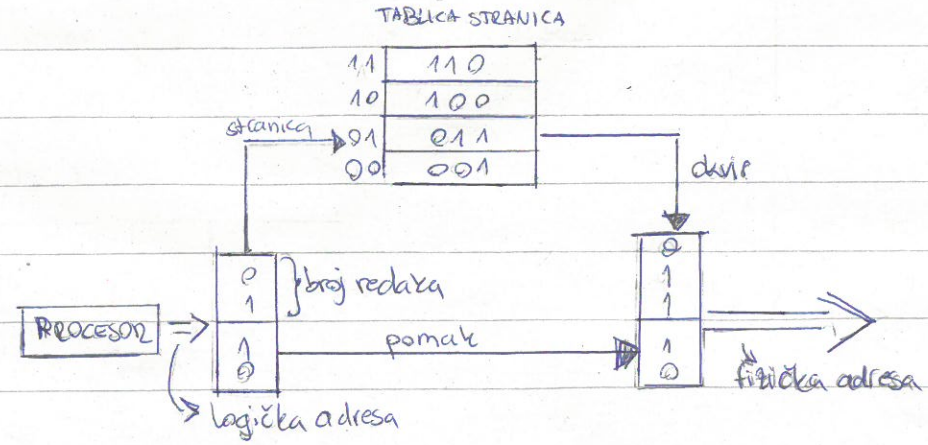
② Na jednostavnom školskom primjeru za program s 12 lokacija i memorijom od 64 lokacija objasniti princip dodjele memorije po stranicama. Objasniti za KONKRETNE brojeve.

Program se nalazi unutar diska. Primjer: Program ima 16 lokacija, a memorija ima 32 lokacije $\Rightarrow 32 = 2^5$

- logični adresni prostor dijeli se na blokove - stranice veličine $2^N$, tj. $N=2 \Rightarrow$ stranice $= 2^2 = 4$
- fizički adresni prostor dijeli se na blokove - okvire veličine $2^N$, tj $N=2 \Rightarrow$ okvira $= 2^2 = 4$
- program se proširi da je puna zadnja stranica



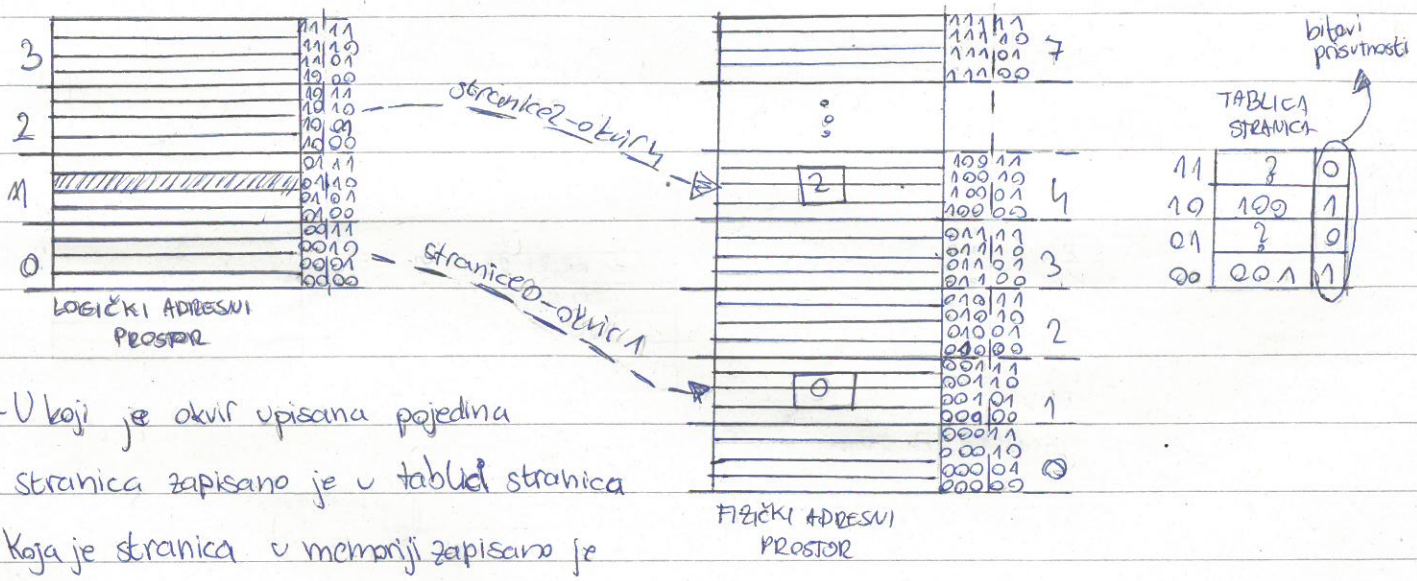

Svaka stranica logičkog adresnog prostora preslikaje se u proizvoljan okvir fizičkog adresnog prostora U koji je okvir upisana pojedina stranica zapiše se u tablici stranica.




Stranica s adrese 01|10 ide u memoriju a 011|10 (pomak ostaje isti), 0110 - logička adresa, 01110 - fizička adresa, Tablica radi preslikavanje iz logičkog u fizički adresni prostor

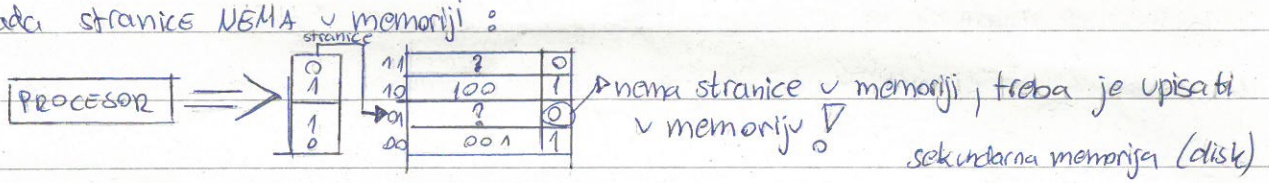
(3) Na istom primjeru objasniti princip virtualne memorije i što se događa kad promašaja stranice. (Jedina razlika je u bitu pristupa)

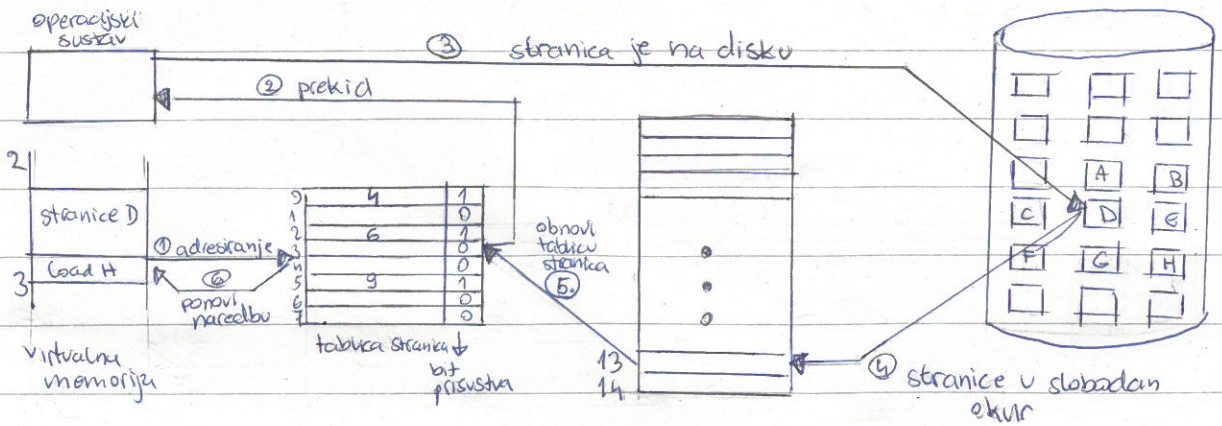
Virtualna memorija - omogućuje nam da ne moramo cijeli program prebacivati u memoriju već samo dijelove. Prednost je što program može biti veći od radne memorije



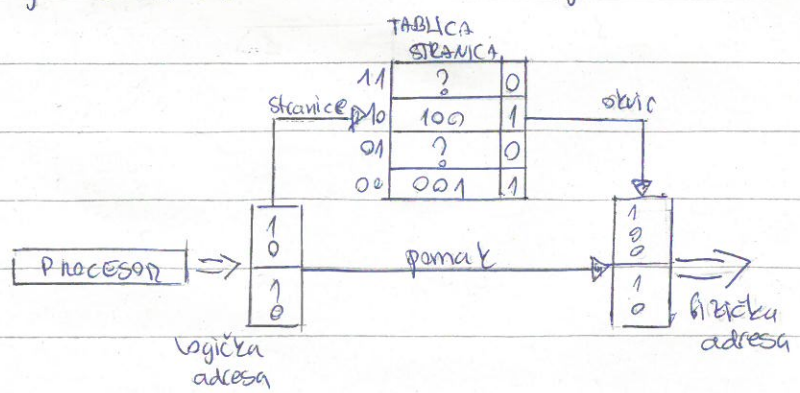

- U koji je okvir upisana pojedina stranica zapisano je u tablici stranica
- Koja je stranica u memoriji zapisano je bitom prisustva

Primjer kada stranice NEMA u memoriji:



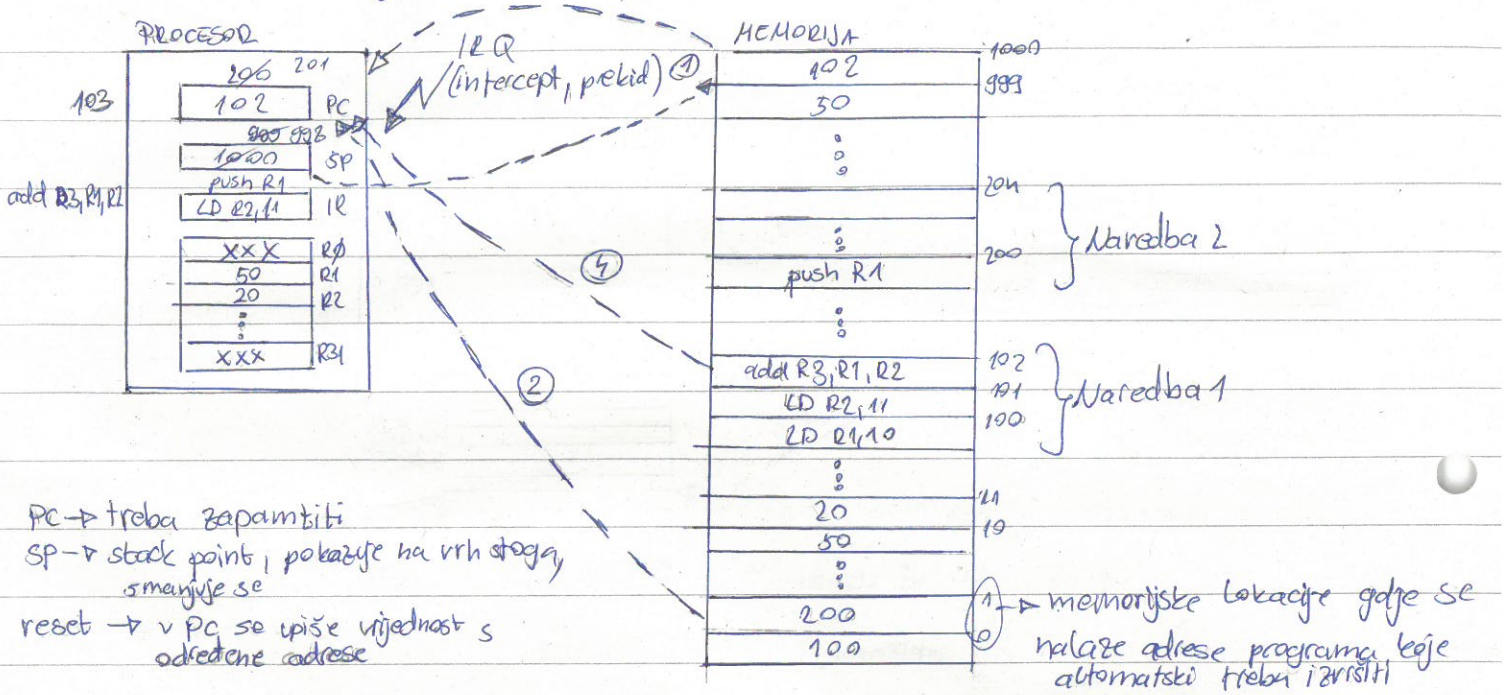




Primjer kada stranice IMA u memoriji:



## (4) Obavljanje U/I operacija pomoću zahtjeva za prekidom.

Kada se dogodi prekid nakon izvršavanja naredbe treba zapamtiti gdje nastaviti, a to kaže PC (adrese sljedeće naredbe).



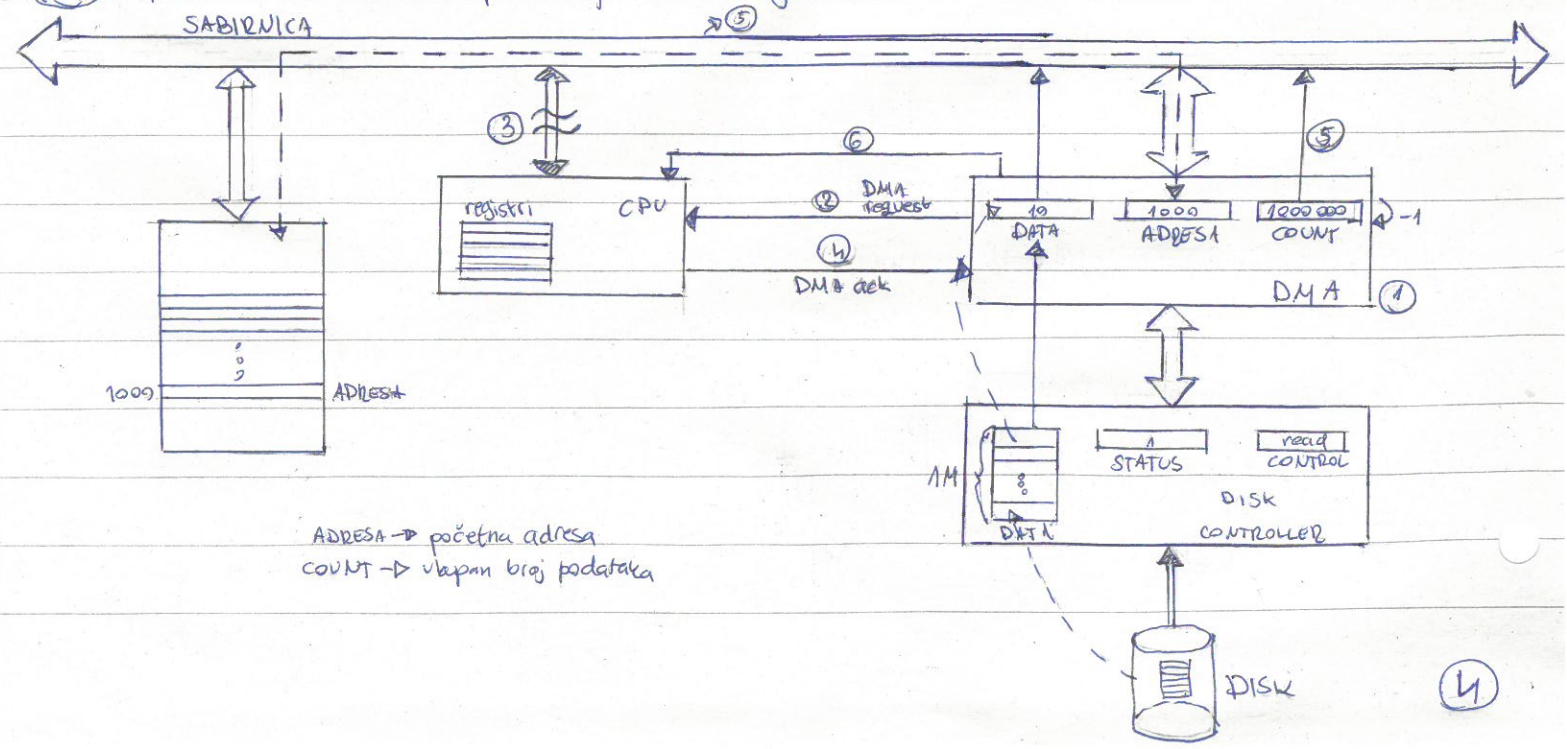

(1) postavljanje vrijednosti na vrh stoga, smanjimo SP za 1

(2) adresu sljedeće naredbe staviti u PC te je ponovno izvršavati (treba znati gdje je)

(3) Izvršavamo Naredbu 2

(4) Nakon završetka Naredbe 2 vraćamo se na 1. naredbu gdje smo stali, a to smo zapamtili pomoću SP

## (5) DMA – Direktan pristup memoriji



① INICIJALIZACIJA DMA prijenosa (program koji direktno upisuje u memoriju

② DMA request – DMA postavlja zahtjev za prijenosom

③ Procesor se odspoji i oslobodi sabirnicu, tj. sučelje prema sabirnici stavi u stanje visoke impedancije

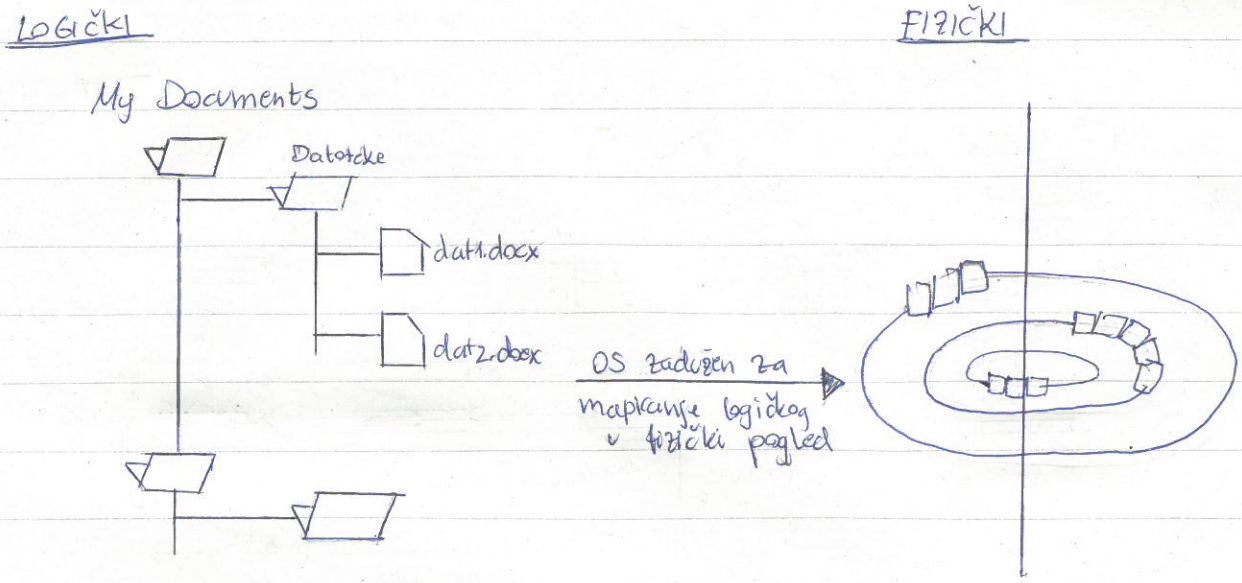
④ Procesor dojavi DMA sklopu da je sabirnica slobodna i da može proslijediti podatke DMA – Acknowledge

⑤ DMA sklop izvodi prijenos podataka na adresu, smanji se COUNT, a adresa se inkrementira. Ovaj postupak ponavljamo sve dok ne prenesemo sve podatke, tj. dok je COUNT ≠ 0

⑥ DMA sklop pomoću zahtjeva za prekidom (IRQ) dojavi procesoru da je prijenos završio. Procesor ponovno preuzima sabirnicu i nastavlja s izvođenjem procesa.

## ⑥ File System

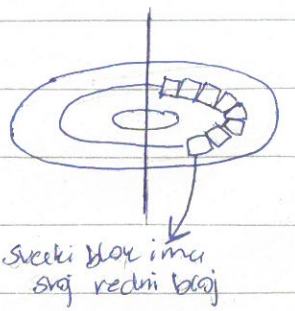
File (datoteka) → može biti program ili podatak. Razlikujemo korisnički (logički) i fizički pogled.




Svaka datoteka ima binarni sadržaj koji dijelimo na manje blokove i spremamo na disk.

Postoje 3 opcije spremanja na disk.

1) kontinuirano

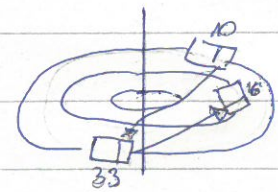



Sprema se jedan blok za drugim, treba samo znati adresu prvog bloka. Brzina očitanja je optimalna jer nema velikih micanja glave, jednostavno i brzo, dobro kod read-only podataka

Predност → kod traženja nekog specifičnog bloka, ne moramo krenuti od 0 nego odmah možemo izračunati gdje se nalazi.

Nedostatak → kada treba proširiti datotek, treba prekopirati cijeli podatak

2) Vezane liste

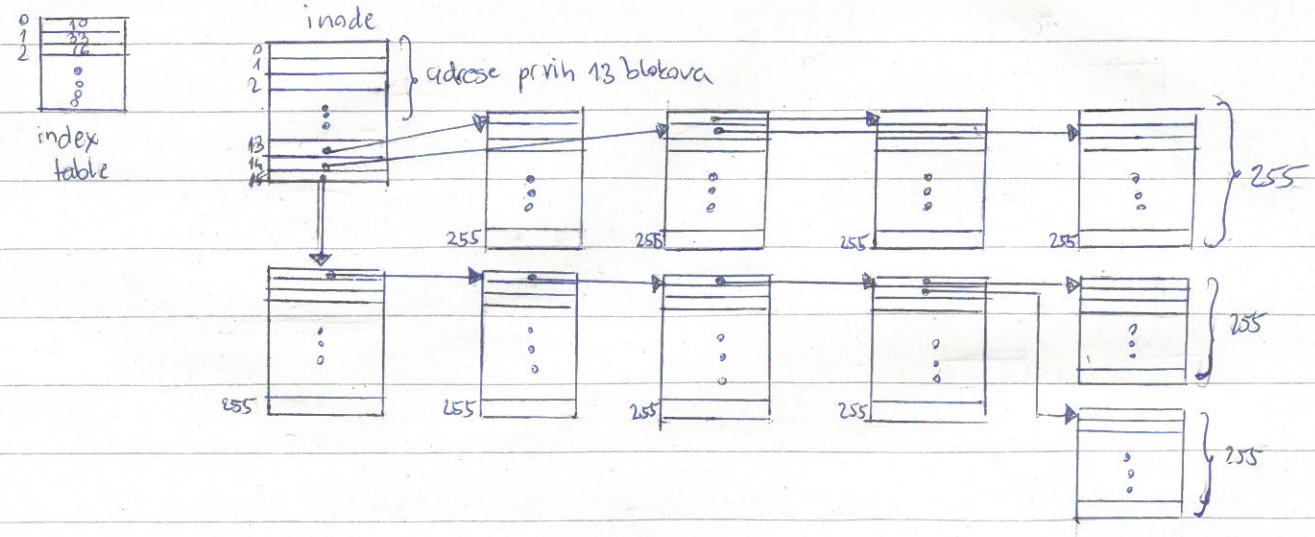



Svaki blok sadrži adresu sljedećeg bloka. Ovo je rješenje nedostatka kontinuiranog prijenosa. Brzina je loša, ali defragmentacijom se može lijepo složiti da nije sve razbacano

Nedostatak → ako želimo naći određeni blok, moramo krenuti od prvog, nema direktan pristup

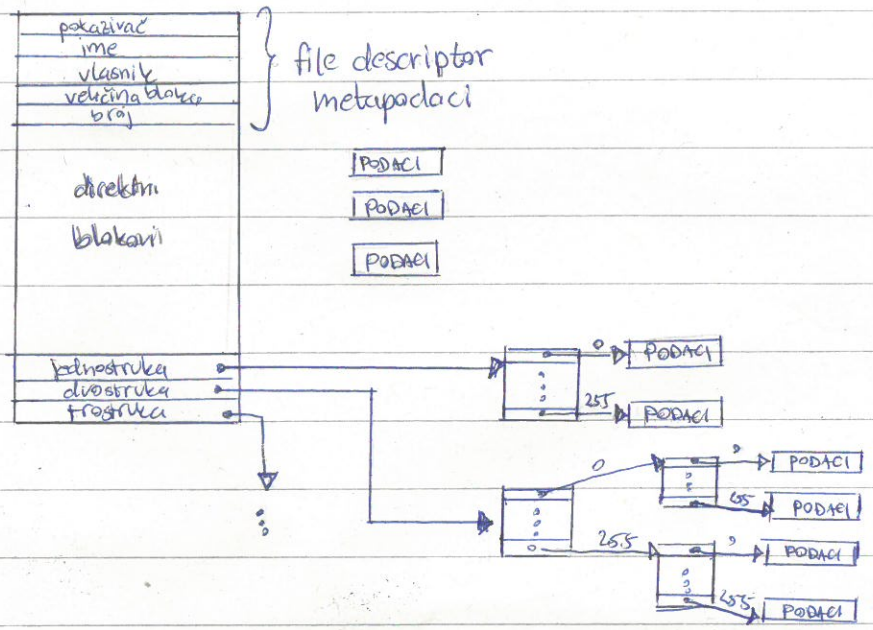
3) Indeksno

Indeksni blok sadrži pokazivače na svaki blok podataka. Kompromis između vezane liste i kontinuiranog spremanja. Brzina je manja ako su blokovi razbacani po disku. Rješenje je defragmentacija diska ali ako nešto naknadno appendamo, opet će se razbacati.



File Control Block (FCB) se kod Unix sustava naziva Inode (Index node) i sadrži sve informacije o datoteci i informacije o smještaju datoteke. To su metapodaci koji uključuju sve informacije vezane uz datoteku osim stvarnih podataka datoteke. FCB je deskriptor procesa.

Inode struktura




Ovo ne znam je li treba... Ako profesor objasni onda treba ako ne bykbye