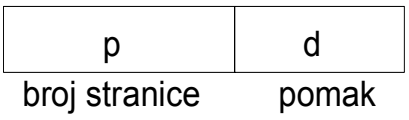
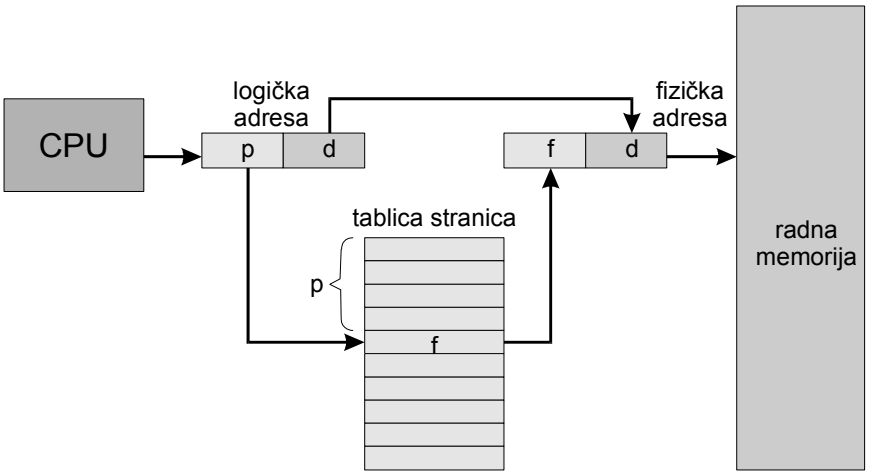
1. DODJELA MEMORIJE PO STRANICAMA (PAGING)

Svaki program sadrži svoj logički adresni prostor koji se dijeli na blokove iste veličine koji se zovu stranice (*pages*). Radna memorija dijeli se na blokove fiksne veličine koji se nazivaju okviri (*frames*). Kada se program upiše u memoriju, stranice se upisuju u slobodne memorijske okvire. Radi jednostavnosti prebacivanja programa s diska u radnu memoriju disk je podijeljen na okvire koji su jednake veličine kao okviri memorije.

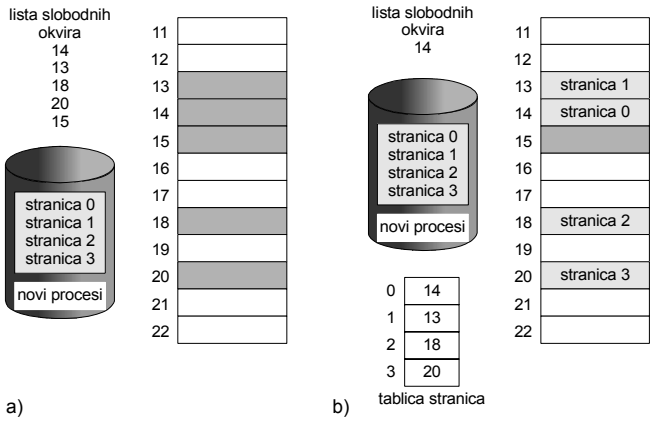


Logička adresa koju generira procesor dijeli se na broj stranice **p** i pomak unutar stranice **d**. Broj stranice **p** predstavlja pokazivač na redak tablice stranica. U tablici stranica upisane su početne adrese okvira **f**. Kombinacija početne adrese okvira **f** i pomaka **d** određuje fizičku adresu memorijske lokacije.

Sklopovlje za dodjeljivanje memorije po stranicama:



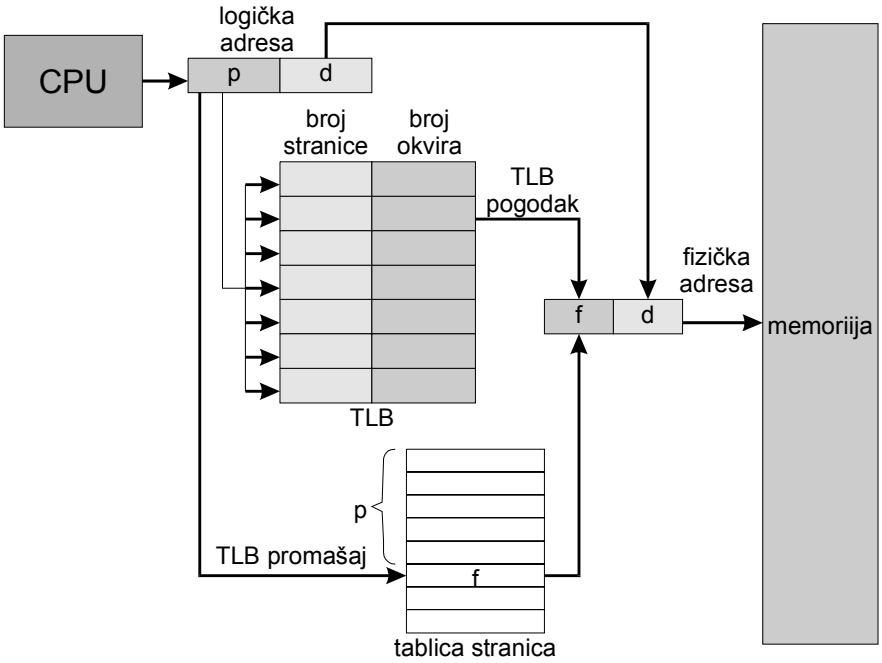
Pri pokretanju programa izračuna se potreban broj okvira i uspoređuje s brojem slobodnih okvira u memoriji. Ako je slobodan dovoljan broj okvira proces se upisuje u memoriju, stranicu po stranicu. U tablici stranica za svaku stranicu se upisuje i broj okvira u koji je ona upisana:



slobodni okviri:
a) prije dodjele memorije procesu
b) poslije dodjele memorije procesu

2. KORIŠTENJE TLB (TRANSLATION LOOKUP BUFFER)

Pri context switch-u između procesa potrebno je obnoviti tablicu stranica (MMU) u procesoru. Zbog velikog broja stranica može se javiti problem memorije i problem prebacivanja stranica iz PCB-a u procesor.
Zbog toga se u MMU zapisuje samo dio tablice stranica .
Straničenje (*paging*) pomoću asocijativnih spremnika (TLB – Translation Lookup Buffer):



Logička adresa se dijeli na dva dijela:

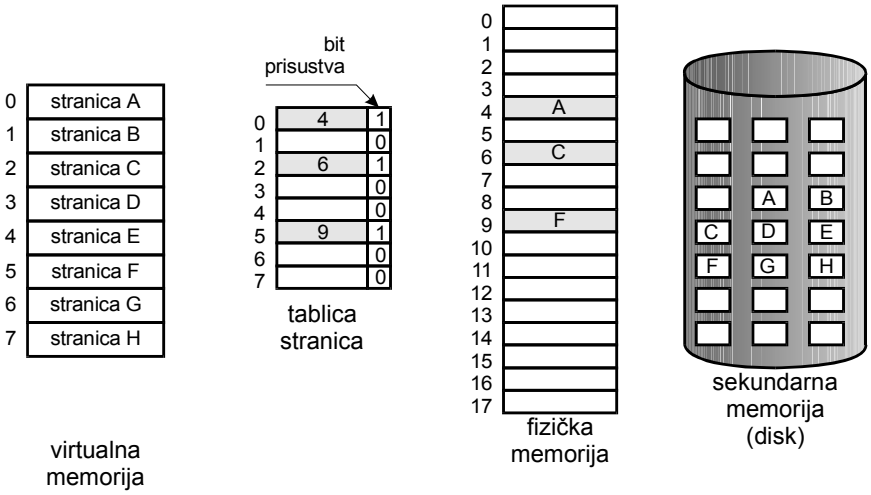
- *p – page*
- *d – displacement*

U TLB-u nađemo broj stranice *p* te očitamo broj okvira *f* (ako se stranica nalazi u TLB). Broj *f* (*frame*) zajedno sa brojem *d* čini fizičku adresu traženog podatka. Ako dođe do promašaja, pristupa se tablici stranica u radnoj memoriji te se broj stranice i broj okvira upisuju na slobodno mjesto u TLB (ako nema slobodnog mjesta, koristi se neki od algoritama za brisanje stranica iz TLB).

3. VIRTUALNA MEMORIJA

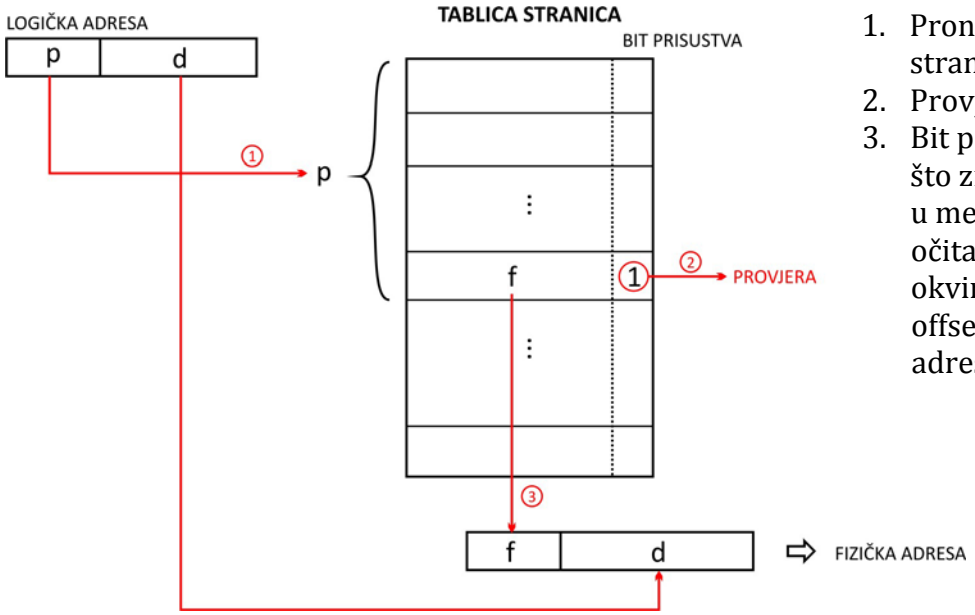
Virtualna memorija je strategija dodjele memorije koja dozvoljava da samo dio programa koji se izvodi bude u radnoj memoriji. To znači da program može biti i veći od radne memorije. Logički adresni prostor koji korisnik vidi razdvojen je od fizičkog adresnog prostora u kojem se program izvodi. Zato programer može raspolagati neograničenim logičkim prostorom iako se program zapravo izvodi u relativno malom fizičkom adresnom prostoru.

Zbog toga ćemo u tablici stranica držat samo onaj dio koji je potreban za funkcioniranje programa.



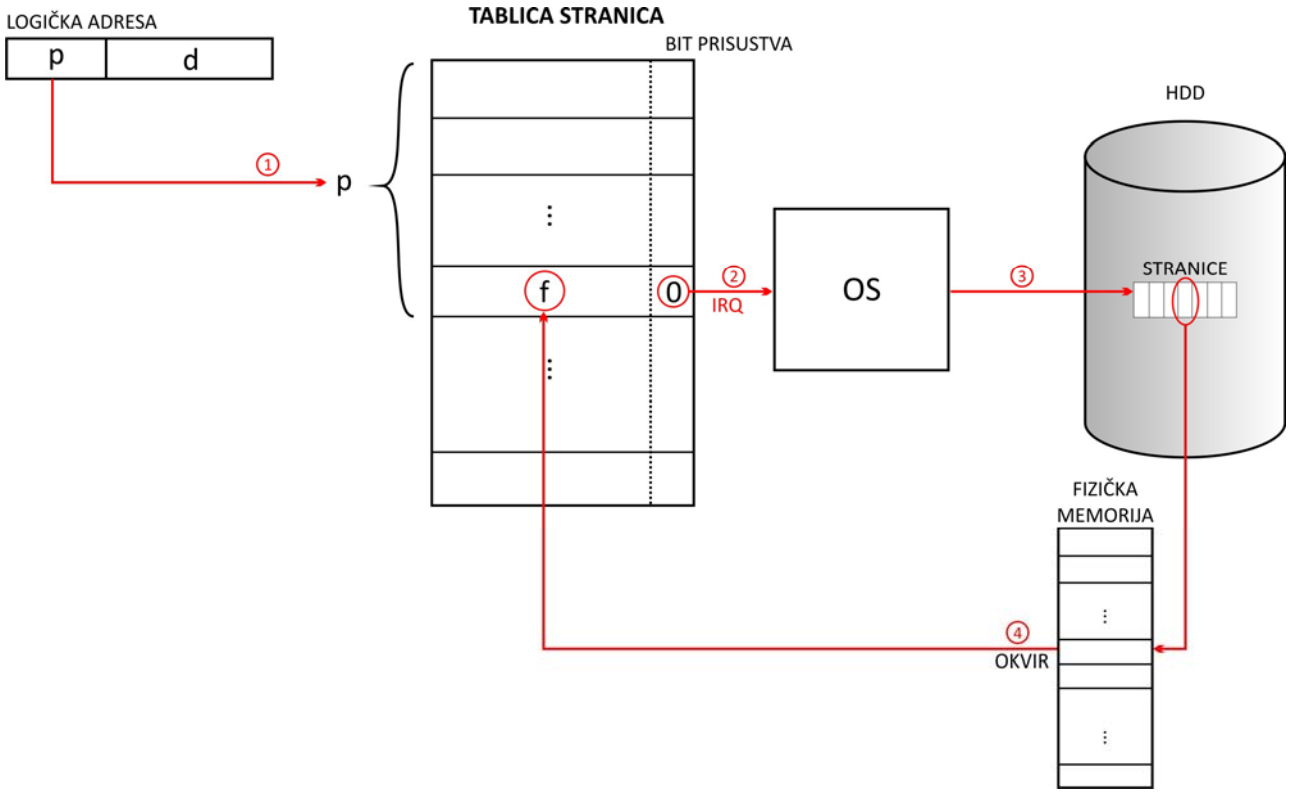
Bit prisustva u stanju 1 znači da se stranica nalazi u fizičkoj memoriji. Ako je bit prisustva u 0, potrebno ju je upisati u memoriju i tek tada izvršiti pristup.

a) kada je bit prisustva 1



1. Pronalazak okvira u tablici stranica
2. Provjera bita prisustva
3. Bit prisustva je 1
   što znači da se okvir nalazi u memoriji, pa možemo očitati početnu adresu okvira i dodavanjem offseta d generirat fizičku adresu.

b) kada je bit prisustva 0



1. Pronalazak okvira u tablici stranica.
2. Provjera bita prisustva (koji je 0).
3. Generira se IRQ koji na osnovi trenutnog procesa i stranice zatraži tu stranicu s HDD-a i dolazi do context switcha.
   Tražena stranica se spremi iz HDD-a u memoriju.
4. Ažurira se tablica stranica (jer se stranica sada nalazi u memoriji pa se bit prisutsva mora promijeniti iz 0 u 1).
5. Ponovo zatraži adresa (sada možemo dobiti fizičku adresu) – *situacija pod a*

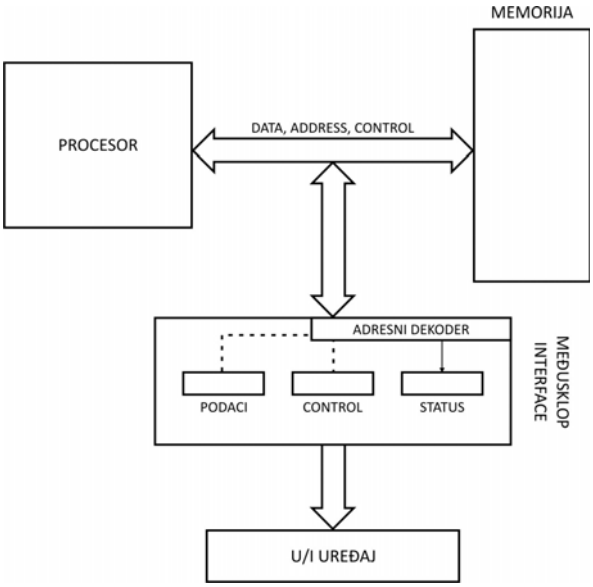
NAPOMENA: OS mora voditi računa o slobodnoj memoriji tj. o slobodnim okvirima.

4. STRATEGIJA IZMJENE STRANICA

Ako je tablica stranica popunjena (nema mjesta za nove stranice) potrebno je izvršiti izmjenu stranica nekim od sljedećih algortmima:

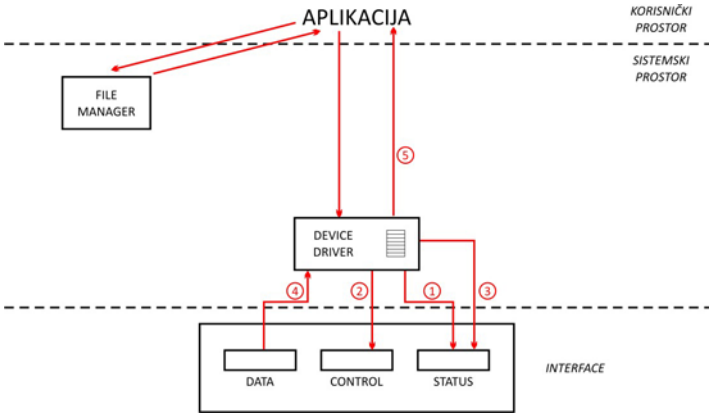
1. FIFO (First In First Out) – zamjenjuje se stranica koja je prva unesena u memoriju
2. optimalan algoritam – optimalno bi bilo zamijeniti onu stranicu koja će se u budućnosti najkasnije koristiti
3. LRU (Last Recently Used) – zamjenjuje se stranica koja se u prošlosti najkasnije koristila

5. PROGRAMSKO OBAVLJANJE U/I OPERACIJA



1. Procesor preko međusklopa zatraži od uređaja podatke (tako da u control registar upiše naredbu koja će inicijalizirati vezu U/I uređaja i međusklopa).
2. U/I uređaj upiše podatke u podatkovni registar; postavi status „podaci upisani“.
3. Procesor pročita taj podatak (Procesor nezna nakon koliko će podatak biti unesen, te stalno čita status).

OS daje podršku za rad sa U/I uređajima . (Npr kod korištenja f-je fscanf (stdio,“format“, adresa) → dovoljno je specificirati koji uređaj koristimo i na koju adresu upisujemo. Unutar operacijskog sustava se kontrolira koji su sve uređaji priključeni i za svaki uređaj postoji lista pokazivača na funkcije koje može koristiti. Svaki uređaj je predstavljen kao datoteka i ta lista uređaja se naziva **file menager**, a polje pokazivača na funkcije koje se mogu koristiti se naziva **device driver.**

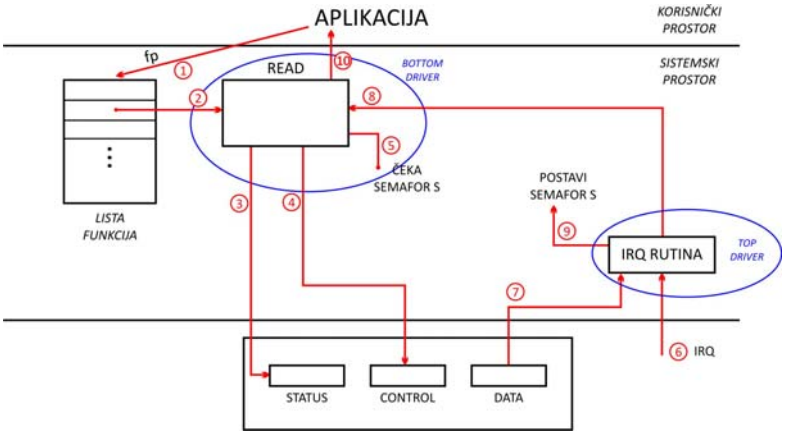


1. Provjeri status uređaja
2. Ako je status zadovoljavajući onda se upisuje naredba
3. Provjera statusa dali je podatak došao
4. Podatak se pohrani u memorijsko područje device drivera
5. Podatak se šalje na adresu koju je poslala aplikacija.
   (**Princip poolinga**)

Nedostatak ovakvog pristupa je aktivno čekanje.

6. OBAVLJANJE U/I OPERACIJA POMOĆU IRQ

Glavna prednost obavljanja U/I operacija pomoći IRQ je izvjegavanje aktivnog čekanja.



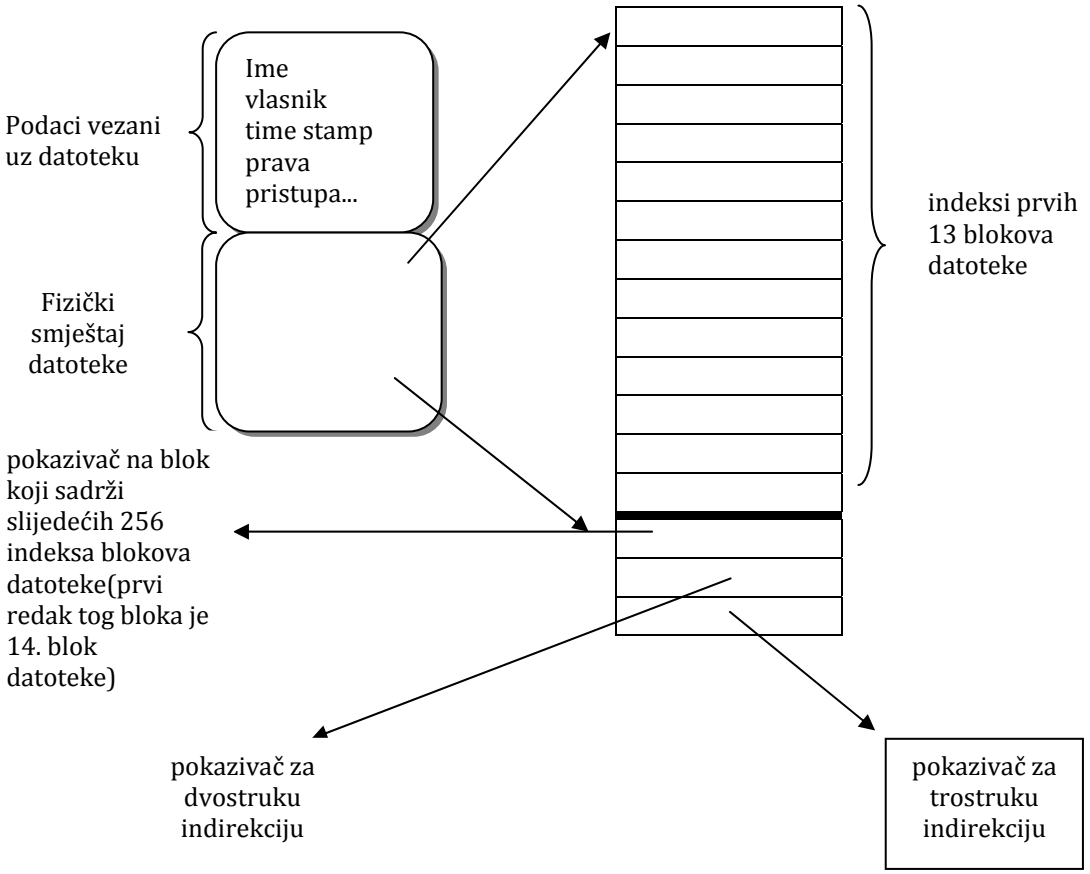
1. Aplikacija zatraži rad s uređajem preko file pointera **fp**
2. Odabere se READ funkcija iz liste funkcija (lista pokazivača na funkcije)
3. READ provjerava status dali je uređaj spreman
4. Zatraži podatak
5. Postavi funkciju READ u stanje čekanja dok se ne dobave podaci
6. Dobave se podaci
7. Podaci se spreme u IRQ rutinu
8. Podaci se spreme u READ
9. Pozove se POSTAVI S (odblokira se semafor)
10. Proslijedi podatke

7. DESKRIPTOR DATOTEKA

Datoteka (file) je skup povezanih informacija ili podataka koji su spremljeni na nekom sekundarnom mediju pod zajedničkim imenom.
Deskriptor datoteke (naziva se **inode**) je struktura (tablica) koja operacijskom sustavu služi za pristup datoteci.
Sadrži :

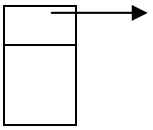
- podatke vezane uz datoteku:
  - ime datoteke, vlasnik, tip, veličina, prava pristupa
  - time stamp (vrijeme stvaranja, modificiranja..)
- podatke vezane uz fizički smještaj datoteke:
  - lokacija – pokazivač na uređaj i mjesto na uređaju gdje je datoteka pohranjena

Inode se zapisuje na disku u super bloku.



8. NAČINI ZAPISA DATOTEKA NA DISK

1. **princip kontinuirane alokacije**
   - memorija se dijeli na blokove određene veličine
   - traži se kontinuirani slijed tih blokova u koji se može zapisati ta datoteka
   - ovisno o veličini datoteke, tj. ako veličina nije višekratnik veličine bloka, zadnji blok ne mora biti popunjen
   - operacijski sustav mora samo znati ime datoteke, adresu prvog bloka i veličinu datoteke
   - mana ove metode zapisivanja je ta što dodavanje podataka u istu datoteku stvara rupe u memoriji jer se datoteka povećava i prepisuje na drugo mjesto u memoriji
2. **princip vezane liste**
   - svaki blok sadrži pokazivač na slijedeći blok



   - operacijski sustav mora imati ime datoteke, veličinu i pokazivač na prvi blok (njegovu adresu) a prvi blok sadrži adresu slijedećeg, itd.
   - mana ove metode je ta što kontroler mora čitati gdje se nalazi slijedeći blok koji se može nalaziti na drugom disku ili slično, pa je taj postupak spor
   - ako želimo pročitati samo jedan blok datoteke (npr. u bazi podataka), moramo proći kroz cijelu vezanu listu, ovisno o tome gdje se taj blok nalazi, što je također jako sporo

3. **indeksna alokacija**
   - omogućuje pojednostavljeno pretraživanje
   - da bi operativni sustav mogao pročitati neku datoteku, on mora imati ime datoteke i tablicu čiji svaki redak sadrži broj bloka u kojem se nalazi dio datoteke