OSNOVI OPERATIVNIH SISTEMA

Pitanja za teoretski dio ispita

Marko Dunović

1. Koji su osnovni resursi sa kojim upravlja OS? Šta mora raditi OS kao resurs menadžer?

Operativni sistem je složeni softverski program koji služi kao interfejs između korisnika i hardvera računara. OS upravlja izvršavanjem svih drugih programa i resursa računarskog sistema.

Osnovne funkcije OS su:

- **Processor Management (upravljanje procesorom):** OS upravlja korištenjem CPU-a, raspoređuje procese i niti, prati njihov napredak i optimizuje korišćenje CPU-a
- **Memory Management (upravljanje memorijom):** OS upravlja RAM-om, osigurava da svaki proces dobije dovoljno memorije, upravlja memorijskim prostorom i optimizuje korištenje memorije
- **Device Management (upravljanje uređajima):** OS upravlja perifernim uređajima kao što su tastature, miševi, printeri, diskovi, itd., osigurava ispravan rad uređaja i omogućava komunikaciju između uređaja i sistema
- **File Management (upravljanje podacima):** OS upravlja datotekama i direktorijumima, omogućava korisnicima i aplikacijama da pristupe i manipulišu datotekama.

Kao resurs menadžer, OS mora:

- **Prepoznati resurs:** identifikovati dostupne resurse u sistemu
- **Voditi računa, ko, kada i koliko resursa dobija:** Pratiti koji procesi koriste koje resurse, kontrolisati pristup resursima i osigurati pravedno raspoređivanje
- **Dodijeliti resurs:** Dodjeljivati resurse procesima i aplikacijama prema njihovim potrebama i prioritetima
- **Osloboditi resurs:** Osloboditi resurse kada više nisu potrebni procesima kako bi bili dostupni drugim procesima.

2. Nabrojati najmanje 3 podatka iz CMOS RAM-a.

CMOS (Complementary Metal-Oxide Semiconductor) RAM sadrži podatke potrebne za osnovno funkcionisanje sistema. Neki od podataka uključuju:

- **Tip hard diska:** informacije o vrsti i konfiguraciji priključenog hard diska
- **Password (lozinka):** bezbjednosne informacije kao što su lozinka za pristup BIOS-u ili sistemu
- **Vrijeme i datum:** trenutno vrijeme i datum koji koristi sistem
- **Potrošnja električne energije:** informacije o napajanju i upravljanju energijom
- **BOOT sekvenca (A, C, CDROM):** redoslijed uređaja sa kojih se sistem pokušava podići.

3. Nabrojati najmanje 3 programa koja se nalaze u ROM čipu.

ROM (Read-Only Memory) čip sadrži osnovne start-up instrukcije koje se izvršavaju prilikom pokretanja računara. Ovi programi uključuju:

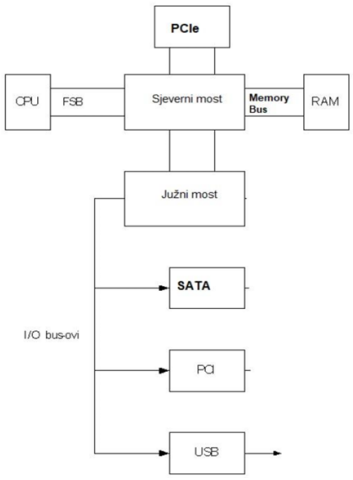
- **POST (Power On Self Test):** testira osnovne komponente hardvera kako bi se osiguralo da su funkcionalne
- **Setup instrukcije:** omogućava podešavanje i konfiguraciju sistema putem BIOS-a
- **BIOS (Basic Input Output System):** osnovne ulazno-izlazne rutine koje omogućavaju osnovnu komunikaciju između operativnog sistema i hardvera
- **BOOT instrukcije:** iniciraju učitavanje operativnog sistema sa disk jedinice ili drugog medija.

4. Nabrojati tipove dizajna operativnog sistema.

- **Monolitni sistemi:** OS je jedan veliki program koji radi u kernel modu, sve funkcije sistema su integrisane
- **Slovjevita arhitektura:** OS je podijeljen u slojeve, svaki sloj komunicira sa slojem iznad i ispod njega
- **Microkernel (Client-Server arhitektura):** osnovne funkcije su u microcernel-u, a dodatne funkcije su u user modu kao serveri
- **Virtuelne mašine:** OS simulira hardver računara kako bi omogućio izvođenje više operativnih sistema na istom fizičkom hardveru.

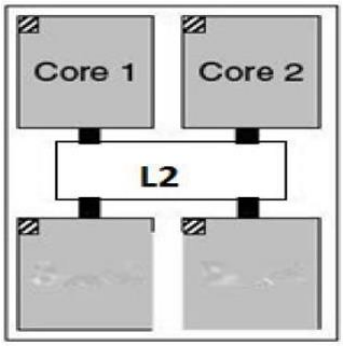
5. Nacrtati blok šemu PC sa sjevernim i južnim mostom.

Blok šema PC-a sa sjevernim i južnim mostom obuhvata centralni procesor (CPU), sjeverni most (Northbridge) koji povezuje CPU sa bržim komponentama poput RAM-a i grafičkog adaptera, i južni most (Southbridge) koji povezuje sa sporijim perifernim uređajima poput hard diska, USB portova i ostalih I/O uređaja.

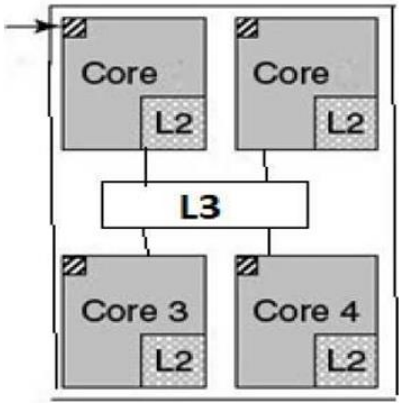


6. Nacrtati blok šemu quad-core (4 jezgre) čipa sa dijeljenim i odvojenim L2 kešom.

- **Dijeljeni L2 keš:** četiri jezgra dijele jedan L2 keš, što omogućava bolju komunikaciju između jezgara, ali može doći do zagušenja.

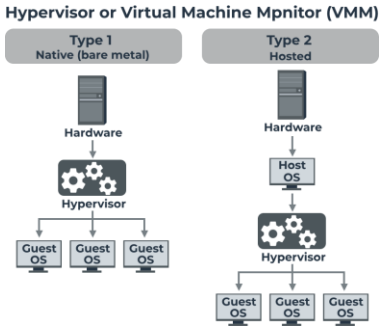


- **Odvojeni (separatni) L2 keš:** Svako jezgro ima svoj L2 keš, što smanjuje zagušenje, ali može smanjiti efikasnost komunikacije između jezgara.



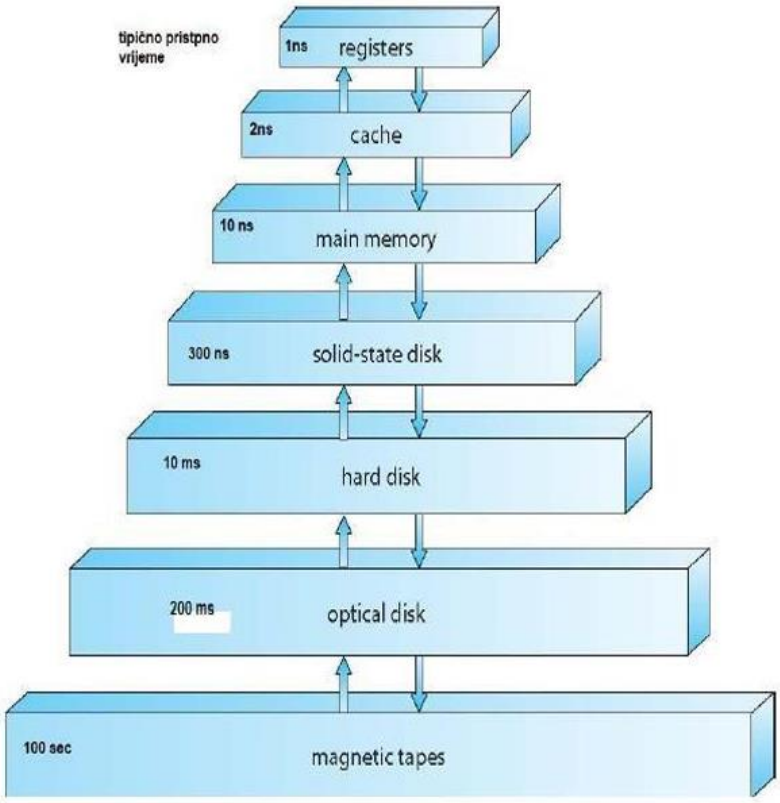
7. Nacrtati blok dijagram Hypervisor tip 1 i tip 2.

- **Hypervisor tip 1 (bare-metal):** direktno radi na hardveru i kontroliše više operativnih sistema. Primjeri: VMware ESXi, Microsoft Hyper-V.
- **Hypervisor tip 2 (hosted):** radi na vrhu operativnog sistema domaćina i omogućava rad virtuelnih mašina. Primjeri: VMware Workstation, Oracle VirtualBox.



8. Navesti hijerarhiju različitih memorija u odnosu na brzinu (tipično vrijeme pristupa).

- **Registri:** najbrža memorija u CPU-u sa najkraćim vremenom pristupa
- **Keš memorija (L1, L2, L3):** brza memorija koja služi kao posrednik između CPU-a i RAM-a
- **Glavna memorija (RAM):** primarna memorija računara, brža od sekundardne memorije
- **Sekundarna memorija (SSD, HDD):** spora memorija koja se koristi za dugotrajno čuvanje podataka.



9. Od čega se sastoji vrijeme disk I/O operacije?

Totalno vrijeme disk I/O operacije se sastoji od:

- **T_seek** – vrijeme pozicioniranja glave na željeni cilindar i odabir glave
- **T_rotacionog kašnjenja** – vrijeme potrebno da se željeni sektor pozicionira ispod glave za čitanje/pisanje
- **T_transfera** – vrijeme potrebno za prenos podataka između diska i memorije.

10. Od koja tri ciklusa se sastoji bazični način izvođenja instrukcija?

Bazični način izvođenja instrukcije na PC-u se sastoji iz tri ciklusa:

- **Fetch** – dohvatanje instrukcije iz memorije
- **Decode** – dekodiranje instrukcije kako bi se utvrdilo šta treba raditi

- **Execute** – izvršavanje same instrukcije.

11. Koja tri konteksta čine proces?

Proces se sastoji iz tri dijela:

- **Hardverski kontekst** – stanje svih hardverskih registara procesora
- **Softverski kontekst** – informacije o stanju procesa, kao što su prioriteti, identifikatori i privilegije
- **Virtuelni memorijski adresni prostor** – virtuelni adresni prostor koji koristi proces za pristup memoriji.

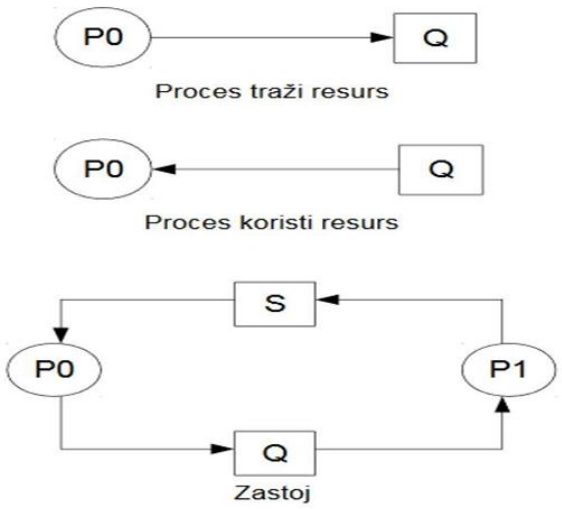
12. Koje informacije sadrži PCB (Process Control Block)? (Najmanje 6 stavki)

PCB (Process Control Block) je ključna struktura podataka u operativnom sistemu koja sadrži informacije o procesu. Informacije sadržane u PCB-u uključuju:

1. **Process ID:** jedinstven identifikator za svaki proces u OS-u
2. **Process State:** tekuće stanje procesa (npr. ready, running, waiting)
3. **Priviliegije procesa:** zahtjevane dozvole za pristup sistemskim resursima
4. **Pointer:** pokazivač na roditeljski proces
5. **Program counter:** pokazivač na adresu sljedeće instrukcije koja će biti izvedena u tom procesu
6. **CPU registri:** različiti CPU registri koje procesor koristi kada je u stanju running
7. **CPU scheduling Information:** prioriteti procesa i ostale informacije za raspoređivanje
8. **Memory management informacije:** informacije o page tabelama, memorijskim ograničenjima i segment tabelama (zavisno od OS MM)
9. **Accounting information:** informacije o potrošnji resursa
10. **IO status information:** uključuje listu I/O uređaja dodijeljenih procesu.

13. Nacrtati deadlock sa dva procesa (P0 i P1) i dva resursa (S, Q).

Deadlock situacija: P0 drži S i traži Q, P1 drži Q i traži S.



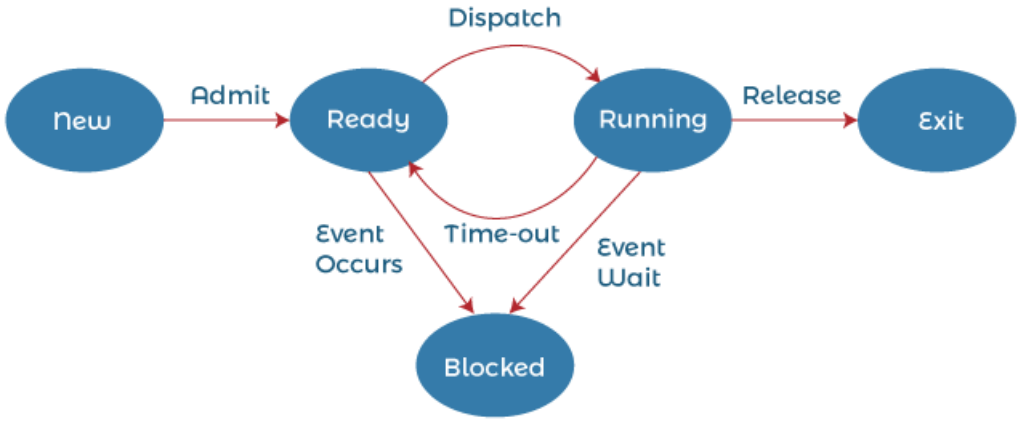
14. Nacrtati šemu osnovnih stanja procesa bez suspendovanih stanja.

Osnovna stanja procesa uključuju:

- new – proces se kreira
- ready – proces se čeka na dodjelu CPU-a
- running – proces se trenutno izvršava na CPU-u
- waiting (blocked) – proces čeka na neki događaj (npr. I/O operaciju)
- terminated – proces je završio izvršavanje.

Povezanost stanja:

- new ---> ready
- ready ---> running
- running ---> waiting (ako zahtjeva I/O)
- waiting ---> ready (kad I/O operacija završi)
- running ---> terminated



Fire State Process Model

15. Nacrtati stanje procesa sa jednim suspendovanim stanjem. Kada proces prelazi iz RUN u WAIT(BLOCKED) stanje?

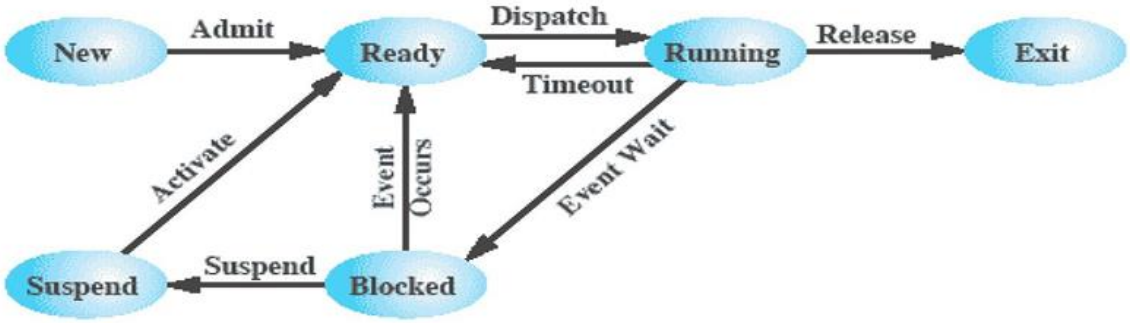
Stanja procesa sa suspendovanim stanjem uključuju:

- new – proces se kreira
- ready – proces se čeka na dodjelu CPU-a
- running – proces se trenutno izvršava na CPU-u
- waiting (blocked) – proces čeka na neki događaj (npr. I/O operaciju)
- suspended – proces je privremeno suspendovan i premješten u sekundarnu memoriju
- terminated – proces je završio izvršavanje.

Proces prelazi iz RUN u BLOCKED stanje kada proces zahtjeva nešto što mora čekati:

- zahtjev za OS u formi sistemskog poziva koji OS nije spreman da izvede trenutno (npr. traži se pristup resursu koji još nije moguć)
- iniciran je I/O i mora se čekati rezultat

- čeka poruku ili podatke od drugog procesa.

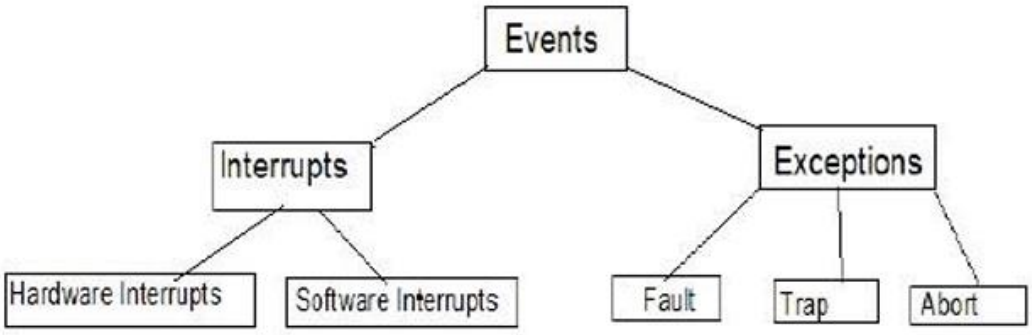


16. Nacrtati vezu između događaja (events), interrupt-ova i njihovu dalju podjelu.

Events (događaji) mogu biti inicirani od strane hardvera (npr. završetak I/O operacije) ili softvera (npr. sistemski poziv).

Interrupts (prekidi) signaliziraju CPU da je događaj zahtijevao pažnju. Prekidi mogu biti:

- hardverski prekidi – dolaze od hardverskih uređaja (npr. tastatura, mrežna kartica)
- softverski prekidi – generisani od strane softvera (npr. dijeljenje nulom).



17. Navesti algoritme (minimalno pet) za kratkoročno raspoređivanje procesa (CPU scheduler).

1. First-Come, First-Served (FCFS) Scheduling – procesi se izvršavaju redoslijedom kojim stižu
2. Shortest-Job-Next (SJN) Scheduling – proces sa najkraćim vremenom izvršavanja se izvršava prvi
3. Priority Scheduling – procesi se izvršavaju prema prioritetima, gdje procesi sa višim prioritetima imaju prednost
4. Shortest Remaining Time – sličan SJN ali uzima u obzir preostalo vrijeme izvršavanja
5. Round Robin (RR) Scheduling – procesi se izvršavaju u kružnom redoslijedu sa fiksnim vremenskim kvantom
6. Multiple-Level-Queues Scheduling – procesi se razvrstaju u više redova prema prioritetima i različiti redovi imaju različite strategije raspoređivanja.

18. Šta čini HW kontekst, tj. u čemu je on sadržan?

Hardverski kontekst se odnosi na stanje CPU registra koje je specifično za trenutno izvršavanje procesa. Ovo uključuje:

- Program Counter (PC) – pokazivač na sljedeću instrukciju za izvršavanje
- Register set – svi CPU registri koje koristi proces
- Stack pointer – pokazivač na vrh stoga.

Kada se proces ne izvodi, sadržaj ovih registara se premješta u memoriju (PCB) kako bi se omogućilo izvođenje drugog procesa, a kasnije se vraća (restaurira) kad se proces ponovo aktivira.

19. Šta opisuje SW kontekst?

Softverski kontekst opisuje softverski status procesa i uključuje:

- identifikacije procesa – jedinstveni identifikator procesa (PID)
- prioritete – prioriteti procesa za raspoređivanje
- potrošnju resursa – informacije o resursima koje proces koristi (CPU vrijeme, memorija)
- privilegije – dozvole i privilegije procesa
- kvote – ograničenja na resurse koje proces može koristiti.

20. Šta su CPU bound i I/O bound procesi?

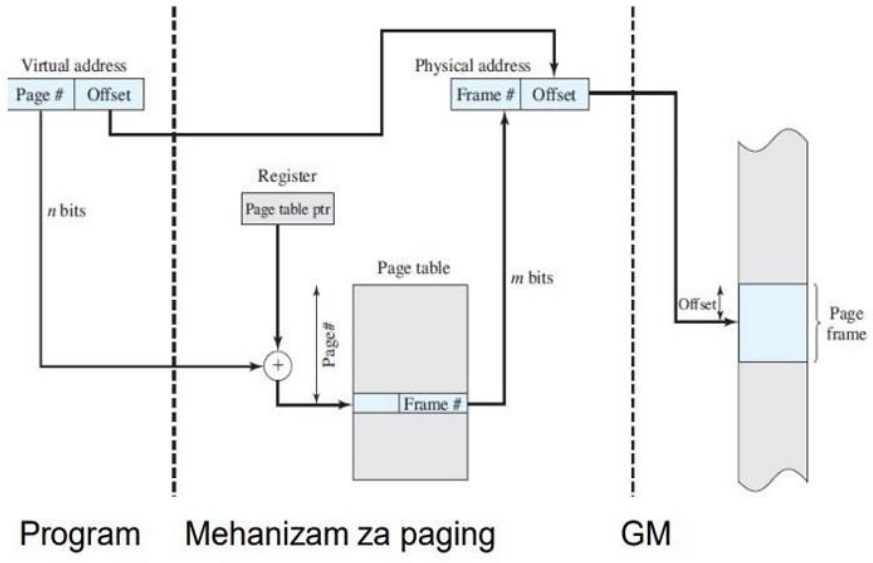
U toku rada, procesor se naizmjenično bavi računskim operacijama (CPU operacijama) i I/O operacijama. U zavisnosti od vremena provedenog u različitim tipovima operacija, procesi mogu biti:

- **CPU-bound** procesi su procesi koji većinu svog vremena provode izvršavajući instrukcije na CPU-u. Oni koriste CPU intenzivno i imaju minimalne I/O operacije.
- **I/O-bound** procesi su procesi koji većinu svog vremena provode čekajući završetak I/O operacija (npr. čitanje/pisanje sa diska). Oni imaju mnogo I/O operacija i relativno malo vremena provode na CPU-u.

21. Nacrtati mehanizam za VM paging (od virtuelne do fizičke adrese preko PMT).

Opis mehanizma VM paging-a:

- Virtuelna adresa se dijeli na dvije komponente: Page number i offset
- Page number se koristi za indeksiranje u Page Map Table (PMT) da bi se pronašao odgovarajući frame number
- Frame number se kombinuje sa offset-om da bi se formirala fizička adresa
- PMT (Page Map Table) sadrži mape koje povezuju virtuelne stranice sa fizičkim okvirima u memoriji.



22. Navesi najmanje pet načina upravljanja glavnom memorijom.

Tehnike upravljanja memorijom:

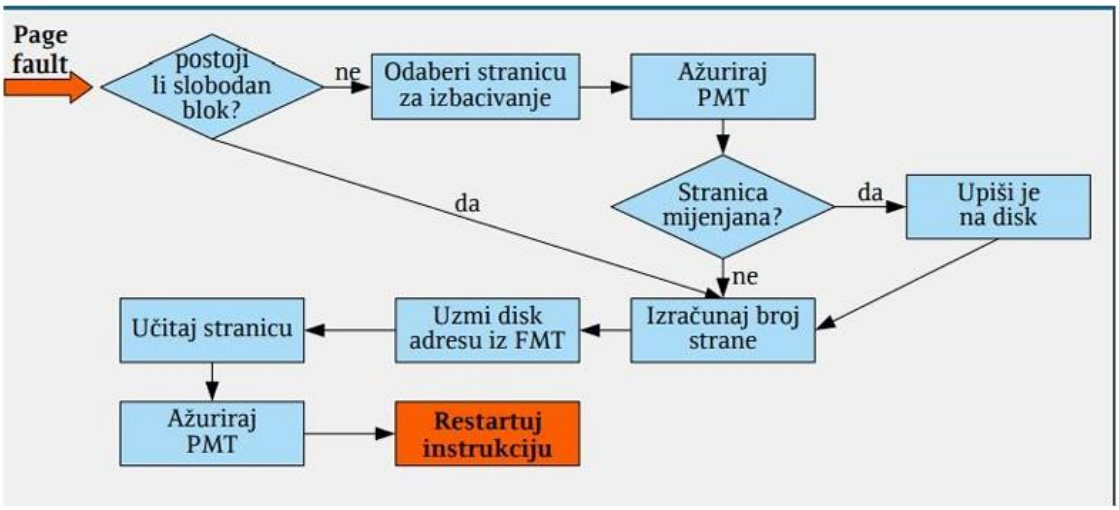
- **jednostavno kontinuirano upravljanje memorijom** – ovaj način upravljanja memorijom koristi se u najjednostavnijim operativnim sistemima, gdje je memorija podijeljena na jedinstveni kontinuirani blok; procesi se dodjeljuju ovim blokovima bez ikakvog particioniranja – ovo može dovesti do fragmentacije memorije
- **upravljanje memorijom sa fiksnim particijama** – memorija je podijeljena na fiksne veličine particija, svaki proces dobija jednu particiju, bez obzira na to koliko memorije stvrano koristi; može dovesti do unutrašnje fragmentacije jer se memorija koristi neefikasno ako proces ne koristi cijelu memoriju
- **upravljanje memorijom sa dinamičkim particijama** – memorija je podijeljena na dinamičke particije koje odgovaraju tačno potrebama procesa – smanjuje unutrašnju fragmentaciju, ali može uzrokovati vanjsku fragmentaciju jer se memorija može podijeliti na mnogo malih, neiskoristivih blokova
- **stranično upravljanje (paging)** – memorija se dijeli na fiksne veličine blokova nazvanih stranica (pages); virtuelne adrese procesa se mapiraju na fizičke adrese koristeći tabelu stranica (page table), ovo eliminiše vanjsku fragmentaciju, ali može doći do unutrašnje unutar svake stranice
- **jednostavna segmentacija** – memorija se dijeli na različite segmente koji odgovaraju logičnim jedinicama procesa kao što su kod segment, data segment i stack segment
- **virtuelna memorija sa segmentima** – kombinuje segmentaciju sa virtuelnom memorijom
- **virtuelna memorija sa segmentima i paging-om** – koristi segmentaciju zajedno sa paging-om

23. Navesti osnovne korake koji se provode kada se desi page fault (ili nacrtati dijagram).

Koraci kada se desi page fault:

1. OS detektuje page fault – OS detektuje da tražena stranica nije u fizičkoj memoriji (RAM-u)
2. provjera slobodnih frejmova – OS provjerava da li ima slobodnih frejmova u fizičkoj memoriji

3. Selektovanje stranice za izbacivanje (ako nema slobodnih frejmova) – OS bira koju stranicu će izbaciti iz memorije, ovo se radi na osnovu algoritma zamjene stranica, npr. LRU (Least Recently Used)
4. provjera da li je stranica bila modifikovana – ako je stranica koja se izbacuje bila modifikovana (M bit je postavljen) mora se zapisati nazad na disk
5. dobijanje disk adrese potrebne stranice – na osnovu broja tražene stranice dobija se disk adresa iz file tabele (FMT – File Map Table)
6. učitavanje stranice sa diska – potrebna stranica se učitava sa diska u slobodni frejm u fizičkoj memoriji, a page entry se ažurira kako bi pokazivao na novi frejm
7. restartovanje instrukcije – instrukcija koja je izazvala page fault se restartuje, sada je tražena stranica dostupna u fizičkoj memoriji.



24. Od kojih polja se sastoji PTE (Page Table Entry) kod paging-a sa virtuelnom memorijom?

Polja PTE uključuju:

- broj frejma (okvira) – indeks fizičkog frejma u koji je stranica mapirana
- present/absent bit (P bit) – indikator da li je trenutno stranica u fizičkoj memoriji (1) jeste, (0) nije
- oznaka privilegije pristupa – dozvole za pristup stranici (read, write, execute)
- modified bit (M bit) – indikator da li je stranica modifikovana od kad je učitana u memoriju
- referenced bit (R bit) – indikator da li je stranica referencirana (korištena) od kad je učitana u memoriju
- cache disable bit – indikator da je keširanje stranice dozvoljeno ili ne.

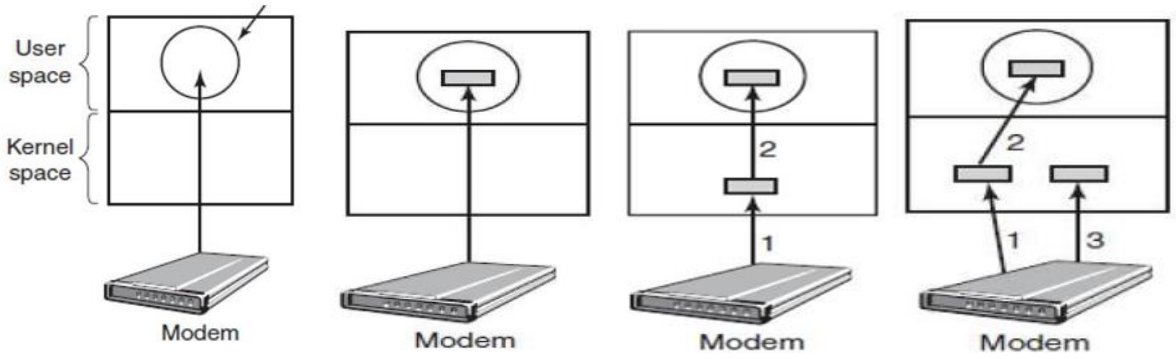
25. Šta je TLB (Translation Lookaside Buffer)?

TLB je specijalni brzi keš za PTE (Page Table Entry). TLB funkcioniše kao memorijski keš i sadrži PTE-ove koji su najčešće korišteni. Kada procesor treba prevesti virtuelnu adresu u fizičku, prvo provjerava TLB. Ako je odgovarajući PTE pristuan u TLB-u (TLB hit), prevođenje se vrši brzo bez potrebe za

pristupom glavnoj memoriji. Ako nije (TLB miss), mora se pristupiti tabeli stranica u glavnoj memoriji, što je sporije.

26. Nacrtati 4 primjera buffer-ovanja (uključujući slučaj bez buffer-ovanja).

- Nebaferovan input – podaci se direktno prenose između I/O uređaja i aplikacije bez korištenja bafera (međuspremnika)
- bafer u korisničkom prostoru – podaci se prvo prenose u bafer u korisničkom prostoru, a zatim aplikacija obrađuje podatke iz tog bafera
- baferovanje u kernelu i korisničkom prostoru – podaci se prvo prenose u bafer u kernel prostoru, a zatim se kopiraju u bafer u korisničkom prostoru
- dvostruko baferovanje u kernelu – prodaci se prenose između dva bafera u kernel prostor, dok jedan bafer prima podatke, drugi se koristi za obradu.

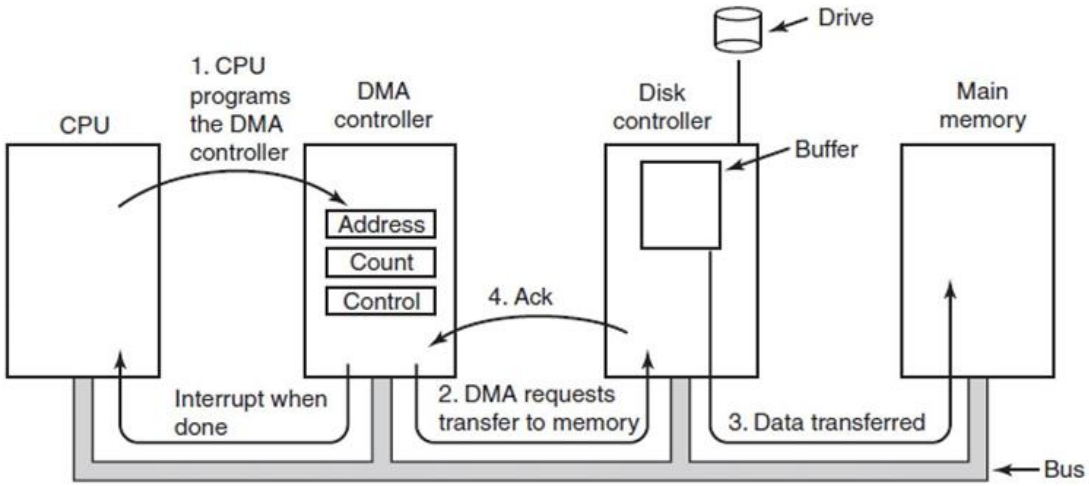


27. Nacrtati blok šemu koja obuhvata CPU, DMA kontroler, Disk kontroler, GM i navesti operacije za DMA transfer.

Operacije za DMA transfer:

1. programiranje DMA kontrolera – CPU postavlja registre DMA kontrolera kako bi znao šta da prebacuje i gdje
2. komanda disk kontroleru – CPU izdaje komandu disk kontroleru da učita podatke sa diska u svoj interni bafer
3. provjera ispravnosti podataka – kada su podaci ispravni (checksum), DMA može startovati
4. iniciranje transfera – DMA kontroler inicira transfer izdajući zahtjev za čitanje disk kontroleru preko busa, disk kontroler ne pravi razliku između zahtjeva od CPU-a ili DMA kontrolera
5. pisanje u memoriju – memorijska adresa gdje disk treba da piše podatke iz svog internog bafera je na bus adresnoj liniji, zatim slijedi pisanje na tu adresu u glavnoj memoriji
6. signal potvrde – kada je pisanje završeno disk kontroler šalje signal potvrde DMA kontroleru preko busa
7. inkrementovanje adrese i dekrementovanje brojača – DMA kontroler inkrementuje memorijsku adresu za sljedeći podatak i dekrementuje brojač bajtova
8. ponavljanje procesa – ako je brojač bajtova veći od 0 koraci se ponavljaju od iniciranja transfera (korak 4) do signalizacije potvrde (korak 6) sve dok brojač bajtova ne postane 0

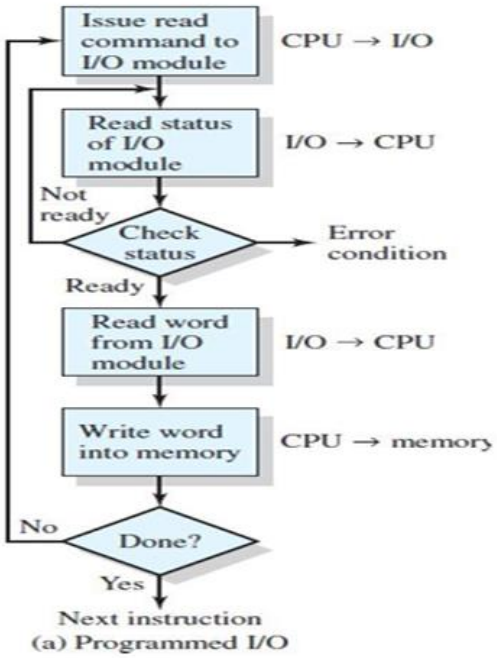
9. obavještavanje CPU-a – kada brojač bajtova postane 0, DMA kontroler prekida CPU kako bi ga obavjestio da je transfer gotov.



28. Nacrtati blok šemu programiranog I/O.

Programirani I/O uključuje:

- CPU direktno upravlja I/O operacijama – CPU šalje komande i podatke direktno I/O uređaju koristeći I/O portove ili memorijski mapirane I/O registre
- polling – CPU kontinuirano provjerava status I/O uređaja (polling) kako bi odredio kada je uređaj spreman za prijem ili slanje podataka
- transfer podataka – kada je I/O uređaj spreman, CPU prenosi podatke između glavne memorije i I/O uređaja.



29. Navesti 4 registra I/O kontrolera koji se koriste za komunikaciju sa CPU.

Skup registara koji se koriste za komunikaciju sa CPU se naziva I/O port i ti registri mogu biti dužine od 1 do 4 bajta i to su obično:

- data-in register – registar sa kojeg CPU čita input sa uređaja
- data-out register – registar u koji CPU piše da bi poslao output uređaju
- status register – registar iz koga CPU dobija status uređaja, CPU koristi ovaj registar za provjeru stanja I/O uređaja
- control register – registar u koji CPU piše komande ili mijenja postavke uređaja.

30. Navesti četiri sloja I/O softvera.

I/O softver je tipično organizovan u četiri sloja:

1. I/O softver na korisničkom nivou – ovo su aplikacije i biblioteke koje direktno koriste I/O operacije, npr. aplikacije mogu koristiti sistemske pozive za čitanje ili pisanje podataka
2. OS softver nezavisan od uređaja – ovo su dijelovi operativnog sistema koji pružaju opšte I/O usluge, nezavisne od specifičnih I/O uređaja
3. drajveri uređaja – specifičan softver za svaki I/O uređaj koji upravlja komunikacijom između operativnog sistema i fizičkog uređaja
4. rukovaoci prekidima – ovi slojevi upravljaju prekidima koje generišu I/O uređaji.

31. Navesti ili nacrtati tri osnovna načina komuniciranja CPU sa I/O registrima.

Način komunikacije CPU-a sa I/O registrima:

1. Port mapiran I/O (izolovani I/O) – I/O uređaji imaju posebne adrese koje se razlikuju od memorijskih adresa, CPU koristi posebne instrukcije za čitanje i pisanje podataka u ove I/O portove
2. Memorijski mapiran I/O – I/O uređaji dijele isti memorijski prostor sa glavnom memorijom, direktno su povezani sa određenim memorijskim lokacijama, što omogućava CPU-u da koristi standardne memorijske instrukcije za interakciju sa uređajima
3. Hibridni I/O – kombinacija memorijski mapiranog i port mapiranog I/O, pentium procesori koriste ovu metodu

32. Navesti tri osnovna načina izvođenja I/O operacija.

Načini izvođenja I/O operacija:

- Programirani I/O (PIO) – CPU direktno kontroliše I/O operacije i prenosi podatke između I/O uređaja i glavne memorije, provjerava status uređaja i vrši transfer podatataka u kontinuitetu
- Interrupt driven I/O – I/O uređaj šalje prekid CPU-u kada je spreman za prijem ili slanje podataka, CPU zatim preusmjerava svoju pažnju na obradu prekida i izvršava potrebne I/O operacije, ovo smanjuje vrijeme čekanja CPU-a jer ne mora u kontinuitetu provjeravati stanje uređaja

- I/O pomoću DMA (Direct Memory Access) – DMA kontroler upravlja transferom podataka između I/O uređaja i glavne memorije bez uključivanja CPU-a, CPU inicira transfer i nastavlja sa drugim zadacima, dok DMA kontroler izvršava transfer, nakon završenog transfera, DMA kontroler prekida CPU da bi ga obavjestio o završetku operacije.

33. Izračunati koje su maksimalne veličine fajlova kod Unix/Linux file sistema u slučajevima: a) 12 direktnih blokova, b) jednostrukog indirektnog adresiranja, c) dvostrukog indirektnog adresiranja, d) trostrukog indirektnog adresiranja, ako je veličina bloka 2 KB, a adresiranje 32-bitno.

Veličina bloka: 2 KB, adresiranje: 32-bitno (4 bajta po pokazivaču)

a) 12 direktnih blokova – direktni blokovi omogućavaju direktno adresiranje blokova podataka, maksimalna veličina fajla: 12 blokova * 2 KB = 24 KB
b) jednostruko indirektno adresiranje – koristi jedan blok za pohranu pokazivača na stvarne blokove podataka, broj pokazivača u jednostrukom indirektnom bloku: 2048 bajtova / 4 bajta po pokazivaču = 512 pokazivača => maksimalna veličina fajla: 512 blokova * 2 KB = 1028 KB (1 MB)
c) dvostruko indirektno adresiranje – koristi blok za pohranu pokazivača na druge blokove pokazivača, broj pokazivača u dvostrukom indirektnom bloku: 512 pokazivača * 512 pokazivača = 262144 pokazivača => maksimalna veličina fajla: 262144 * 2 KB = 512 MB
d) trostruko indirektno adresiranje – koristi blok za pohranu pokazivača na blokove koji sadrže pokazivače na druge blokove pokazivača, broj pokazivača u trostrukom indirektnom adresiranju: 512 * 512 * 512 = 134,217,728 pokazivača => maksimalna veličina fajla: 134217728 * 2 KB = 256 GB.

34. Koja su tri osnovna načina pristupa fajlovima?

Osnovni načini pristupa fajlovima:

- sekvencijalni pristup – pristup se obavlja redom, jedan po jedan bajt ili zapis, od početka do kraja
- direktni pristup – omogućava direktno pozicioniranje na određenu tačku u fajlu i pristup sadržaju
- indeksirani sekvencijalni pristup – kombinuje sekvencijalni i direktni pristup.

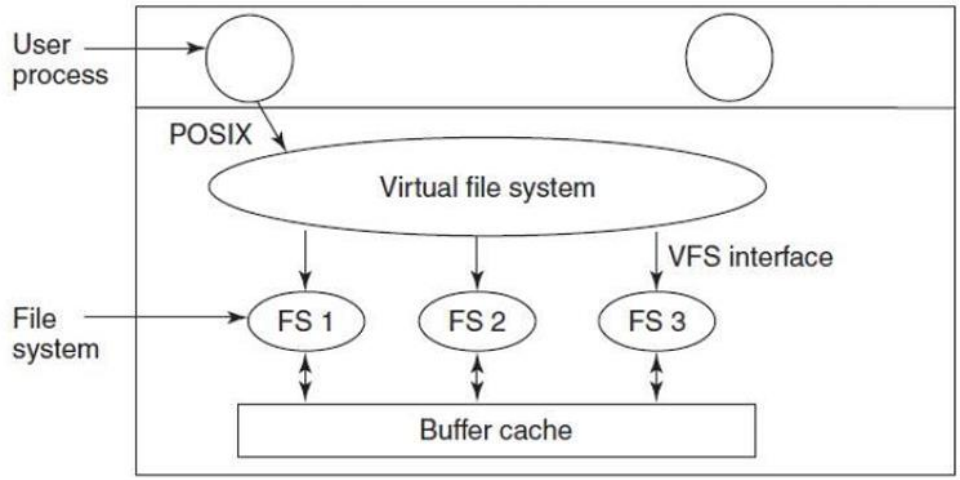
35. Nabrojati bar 5 tipičnih atributa fajla.

Tipični atributi fajla su:

- Ime fajla – jedinstveno ime koje identifikuje fajl u direktorijumu
- ID korisnika i grupe – ID korisnika i grupe koji imaju prava nad fajlom
- Vremenske oznake – vremena kreiranja, izmjena i pristupa fajlu
- vrsta fajla – tip fajla
- prava pristupa – dozvole za čitanje, pisanje i čitanje fajla
- veličina fajla – veličina fajla u bajtovima.

36. Nacrtati poziciju VFS-a (Virtual File System) u Unix/Linux sistemima.

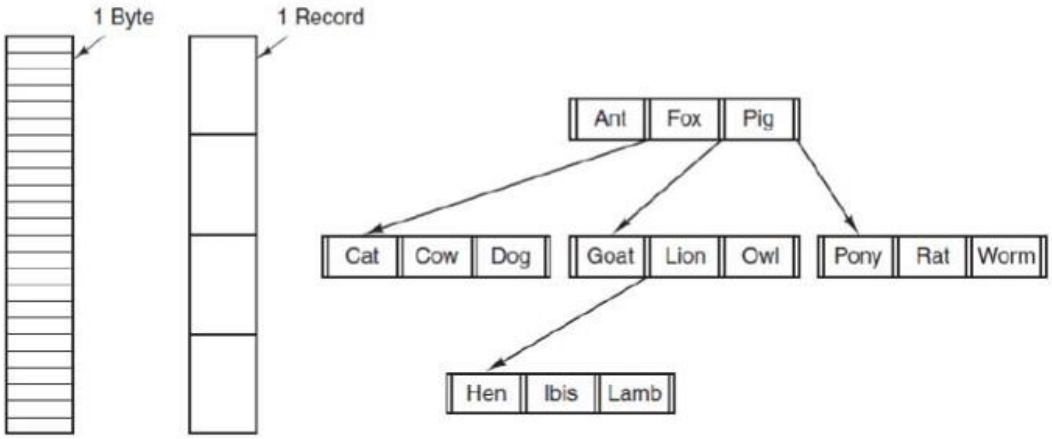
VFS je apstrakcijski sloj unutar kernela Unix/Linux sistema koji omogućava dosljedan način pristupa različitim fajl sistemima. VSF se nalazi između sistemskih poziva fajlova na nivou korisnika i specifičnih implementacija fajl sistema unutar kernela. Kada aplikacija izvrši sistemski poziv vezan za fajl (npr. open, read, write) VSF interfejs prosljeđuje te pozive odgovarajućem fajl sistemu (npr. ext4, NTFS).



37. Nacrtati tri uobičajene varijante strukture fajlova.

Tri uobičajene varijante strukture fajlova:

- struktura bajtova – fajl se tretira kao niz bajtova bez posebne strukture, tipično za teksutalne fajlove
- struktura zapisa – fajl se sastoji od niza zapisa, gdje svaki zapis može imati fiksnu ili promjenljivu dužinu, korisno za baze podataka
- struktura stabla – fajl je organizovan kao hijerarhijsko stablo gdje svaki čvor može predstavljati direktorijum ili fajl, tipično za fajl sisteme sa kompleksnom hijerarhijom.



38. Navesti tri načina alokacije blokova u fajlu.

- Kontinualna alokacija – svi blokovi fajla su alocirani kao jedan neprekidni segment u memoriji
- alokacija ulančanom listom – blokovi fajla su uvezani pokazivačima, svaki blok sadrži pokazivač na sljedeći blok
- alokacija ulančanom listom upotrebom tabele u memoriji (FAT – File Allocation Table) – koristi se tabela u memoriji koja sadrži pokazivače na blokove fajla.

39. Navesti tri osnovna načina za pristup fajlovima na fajl sistemu.

- Sekvencijalni pristup – proces može čitati sve bajtove ili zapise u fajlu po redu, počevši od početka i prolazeći ih jedan po jedan
- direktni pristup – omogućava direktan pristup bilo kojem zapisu ili bajtu u fajlu pomoću njegove adrese
- indeksirani sekvencijalni pristup – koristi indekse za brži pristup određenim dijelovima fajla, indeksi sadrže pokazivače na zapise ili bajtove omogućavajući kombinaciju sekvencijalnog i direktnog pristupa.

40. Objasniti disk blokove (Unix) i cluster-e (Windows) kod fajl sistema.

- Disk blokovi (Unix) – kontinuirani niz sektora na disku koji čine najmanju jedinicu za pohranu podataka u Unix/Linux fajl sistemima, veličine blokova su obično 1K, 2K, 4K, itd., disk blokovi omogućavaju efikasno upravljanje prostorom na disku i brzi pristup podacima.
- Cluster-i (Windows) – kontinuirani niz sektora na disku koji čine najmanju jedinicu za pohranu podataka u Windows fajl sistemima (npr. FAT, NTFS), veličine cluster-a obično su 1K, 2K, 4K, 8K, 16K, 32K, 64K, cluster-i omogućavaju upravljanje prostorom na disku i brzi pristup podacima, ali mogu dovesti do unutrašnje fragmentacije.

41. Od čega se sastoji svaki MFT record kod NTFS fajl sistema?

MFT (Master File Table) je ključna struktura u NTFS (New Technology File System) fajl sistemu koja sadrži informacije o svim fajlovima i direktorijumima na volumenu. Svaki fajl i direktorijum ima svoj zapis (record) u MFT-u. MFT record je standardno veličine 1 KB.

Sastav MFT record-a:

- zaglavlje recorda (Record Header) – sadrži osnovne informacije o Recordu kao što su tip recorda i veličina recorda

- standardni atribut (Standard Information Attribute) – sadrži informacije o fajlu kao što su vremenske oznake (kreiranje, izmjene), atributi fajla (skriven, sistemski, samo za čitanje, itd.)
- atribut imena (File Name Attribute) – sadrži ime fajla ili direktorijuma
- atribut sigurnosnog identifikatora (Security Identificator Attribute) – sadrži sigurnosni identifikator SID koji određuje vlasnika fajla i sigurnosne informacije
- atributi podataka (Data Attributes) – sadrže listu disk adresa na kojima su locirani fajl blokovi
- atributi proširenja (Extended Attributes) – mogu sadržati dodatne informacije o fajlu kao što su alternativni tokovi podataka
- atribut indeksnih root-ova (Index Root Attribute) – koristi se za organizaciju direktorijuma i indeksiranje fajlova unutar direktorijuma
- atribut indeksnih zapisa (Index Allocation Attribute) – sadrži informacije o fizičkom rasporedu blokova koji čine indeksne strukture direktorijuma.

MFT record sadrži sve potrebne informacije za upravljanje fajlovima i direktorijumima na NTFS volumenu, uključujući sigurnosne atribute, informacije o alokaciji prostora, i dodatne podatke koji omogućavaju efikasno upravljanje fajl sistemom.