Operacijski sustavi

Dasko, Nino

Undergraduate thesis / Završni rad

2015

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:800082

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Image not found or type unknown

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

zir.nsk.hr

Image not found or type unknown

Sveučilište Jurja Dobrile u Puli

Fakultet za ekonomiju i turizam „Dr. Mijo Mirković“

NINO DASKO

OPERACIJSKI SUSTAVI

Završni rad

Pula, 2015.

Sveučilište Jurja Dobrile u Puli

Fakultet za ekonomiju i turizam „Dr. Mijo Mirković“

NINO DASKO

OPERACIJSKI SUSTAVI

Završni rad

Matični broj: 2027-E, redovni student

Studijski smjer: Informatika

Predmet: Osnove IKT-a

Mentor: Prof. dr. sc. Vanja Bevanda

Pula, 2015.

Nino Dasko

IZJAVA O AKADEMSKOJ ČESTITOSTI

Izjavljujem i svojim potpisom potvrđujem da je ovaj završni rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu, a što pokazuju korištene bilješke i bibliografija. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Student:

U Puli, 15. kolovoza 2015. _____

Sadržaj

I. Uvod ... 2

1. Uvod u operacijske sustave: funkcije i obilježja ... 3

2. Ulazno – izlazne operacije ... 5

2.1. Spajanje ulazno-izlaznih jedinica u računalo ... 6

2.2. Prijenos pojedinačnih znakova ... 8

2.3. Prekidni način rada procesora s jendim prekidnim ulazom ... 11

3. Međusobno isključivanje u višedretvenim sustavima ... 12

3.1. Općenito o procesima ... 12

3.2. Višedretveno ostvarenje zadataka ... 13

3.2.1. Model višedretvenosti ... 15

3.2.2. Sustav dretvi ... 16

3.2.3. Međusobno isključivanje ... 17

4. Jezgra operacijskog sustava ... 18

4.1. Jezgrin jednostavan model ... 18

4.2. Jednostavni model jezgre – struktura podataka ... 21

4.2.1. Lista postojećih dretvi – pasivno, aktivno, pripravno i blokirano stanje ... 22

4.3. Jezgrine funkcije ... 24

5. Upravljanje spremničkim prostorom ... 25

5.1. Magnetski diskovi ... 26

5.2. Magnetski disk kao pomoćni spremnik ... 27

5.3. Proces dodjeljivanja radnog spremnika straničenjem ... 28

5.3.1. Stranice i okviri ... 28

5.3.2. Virtualni spremnik ... 30

6. Datotečni sustav ... 31

6.1. Smještanje datoteka na disku ... 31

6.2. Načela ostvarivanja datotečnih funkcija ... 32

6.3. Posluživanje zahtjeva za pristup datotekama ... 33

7. Podjela operacijskih sustava ... 34

7.1. Operacijski sustavi namijenjeni osobnim računalima i njihov razvoj ... 34

Zaključak ... 36

Literatura 37
Popis slika 38

I. Uvod

Razvojem informacijskih tehnologija i programske potpore rada računalnog sustava, računala se u današnjem svijetu koriste u gotovo svim područima ljudskog djelovanja. U današnje vrijeme, računala su postala jednostavna za korištenje te su dostupna svima. Život bez računala postao je nezamisliv, a posao bez njih neostvariv.

Najkompliciraniji dio unutar samog računala jest softver računalnog sustava koji upravlja računalnim sustavom, a bez njega rad računala postaje nemoguć. Softver računalnog sustava dijeli se na sustavni i aplikacijski softver. Sustavni softver se nadalje dijeli na operacijski sustav, uslužni softver i jezike prevoditelje. Upravo je operacijski sustav, kao jedna od komponenti sustavnog softvera, tema ovog završnog rada.

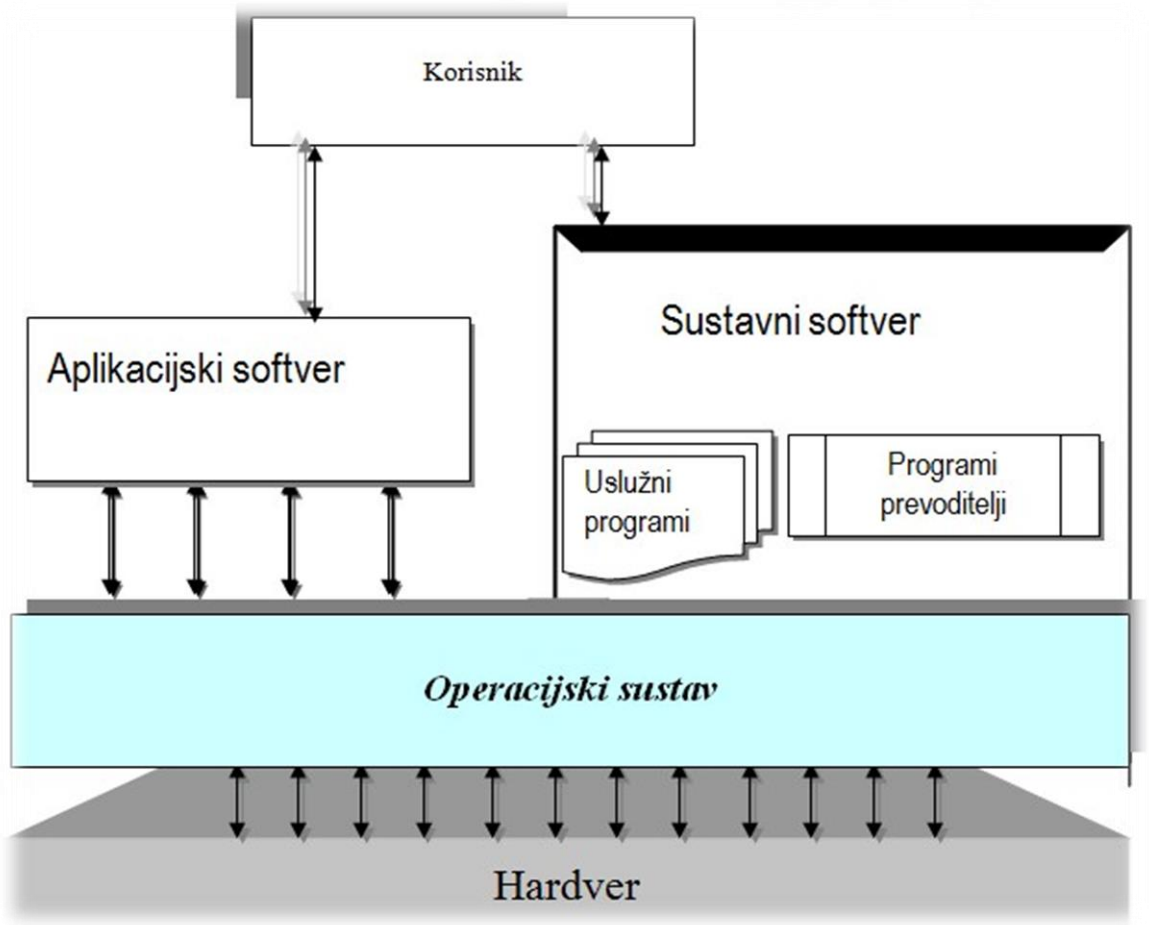
Operacijski sustav je skup programa koji omogućuje efikasno upravljanje računalnim hardverom, on pruža podršku za korištenje programa te djeluje kao posrednik između korisnika računala i samog računalnog hardvera. Glavna zadaća operacijskih sustava prvenstveno je optimizacija korištenja hardvera. Operacijski sustavi osobnih računala dizajnirani su tako da podržavaju sve, od kompleksnih igara do poslovnih aplikacija. S druge strane, operacijski sustavi za dlanovnike su dizajnirani tako da pružaju korisniku jednostavno računalno sučelje za izvršavanje programa. Dakle, neki operacijski sustavi dizajnirani su da budu praktični, neki da budu efikasni, a neki su kombinacija praktičnosti i efikasnosti.

Cilj ovog završnog rada je opisati obilježja operacijskog sustava, te objasniti bitne funkcije operacijskog sustava, imajući u vidu da neće biti govora o sigurnosnim funkcijama operacijskog sustava. Također, u radu neće biti govora o mobilnim operacijskim sustavima.

Sama struktura završnog rada sastoji se od sedam poglavlja koji su usko povezani s radom operacijskog sustava. Svako poglavlje opisano je tako da su istaknute najvažnije značajke potrebne za funkcioniranje operacijskog sustava. Rad započinje s obradom ulazno – izlaznih jedinica gdje se opisuje njihovo spajanje na računalo, prijenos pojedinačnih znakova i prekidni način rada procesora s jednim prekidnim ulazom. U trećem poglavlju obrađuje se međusobno isključivanje u višedretvenim sustavima. Četvrto poglavlje odnosi se na centralni dio modernog operacijskog sustava, odnosno na jezgru operacijskog sustava. Peto poglavlje govori o upravljanju spremničkim prostorom, a šesto govori o datotečnim sustavima. Rad završava sa sedmim poglavljem, u kojem se kratko opisuju dva najvažnija operacijska sustava poduzeća Microsoft.

1. Uvod u operacijske sustave: funkcije i obilježja

Kao što je u uvodnom dijelu rečeno, najsloženiji dio unutar svakog računala je sustavni softver čija je zadaća upravljanje računalnim sustavom. Sustavni softver predstavlja skup stalno prisutnih programa u računalnom sustavu koji omogućuju upotrebu računalnih resursa.



Slika 1. Softver računalnog sustava *(Izvor: B. Markić: Informatika, Ekonomski fakultet u Mostaru, Sveučilište u Mostaru, 2008.)*

Na Slici 1. prikazan je softver računalnog sustava. Uz sustavni softver na slici 1, prikazan je i aplikacijski softver. Aplikacijski softver "bliži" je korisniku i rješava konkretan korisnikov zahtjev, zadatak ili problem. Računalo uz pomoć aplikacijskog softvera rješava točno definiran korisnikov zadatak.[1]

[1] B. Markić: Informatika, Ekonomski fakultet Sveučilište u Mostaru, 2008. (Poglavlje 4, str. 1)

Na slici 1. također je prikazan operacijski sustav koji predstavlja posrednika između korisnika i računalne sklopovske opreme. Operacijski sustav korisniku olakšava implementaciju i održavanje programa, a istovremeno upravlja dodjelom resursa sustava tako osiguravajući njihovu učinkovitu primjenu. S gledišta načina komuniciranja između korisnika i računala te vremena odaziva računala na korisnikov zahtjev, postoje tri temeljna načina rada računalnog sustava, a to su: skupna obrada, obrada s podjelom vremena i rad u realnom vremenu.[2]

Skupovi zadataka koje programi operacijskog sustava uspješno rješavaju predstavljaju *funkcije* operacijskog sustava. Opće funkcije svakog operacijskog sustava jesu:

- Upravljanje zadacima
- Upravljanje podzadacima
- Upravljanje ulazom i izlazom
- Upravljanje memorijom
- Obrada prekida
- Zaštita
- Podržavane daljinske obrade
- Interpretiranje i izvođenje kontrolno-upravljačkih naredbi
- Podržavanje rada u računalnoj mreži[3]

U daljnjem radu, obradit će se neke od navedenih funkcija i navest će se njihovi zadaci.

Sve navedene funkcije operacijskog sustava imaju određena *obilježja* operacijskog sustava koja se očituju prilikom izvršavanja skupova zadataka. U neka od obilježja operacijskog sustava spadaju: Istovremenost, djelotvornost, djeljivost resursa, sigurnost, fleksibilnost, uporabljivost

[2] Ibid str. 2
[3] Ibid str. 3

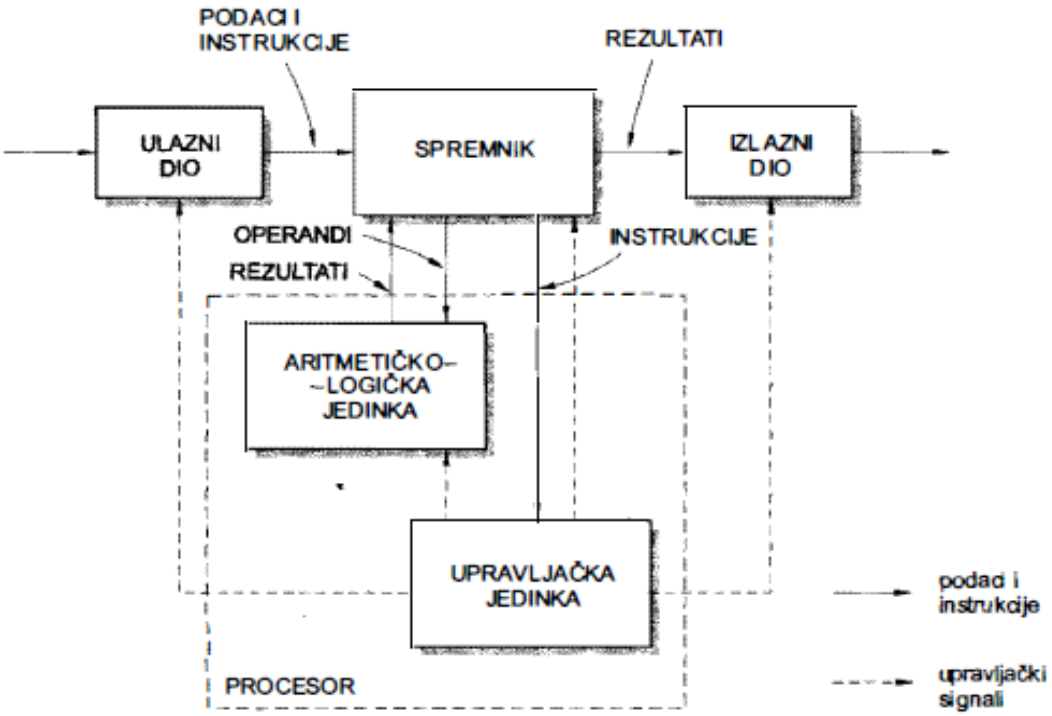
Istovremenost u operacijskom sustavu omogućava paralelno izvođenje više procesa. Djelotvornost operacijskog sustava odnosi se na djelotvorno korištenje svih resursa računalnog sustava. Dijeljenjem resursa povećava se djelotvornost računalnog sustava. Sigurnost operacijskog sustava odnosi se na zaštitu programa i podataka od neovlaštenog korištenja. Fleksibilnost omogućuje operacijskom sustavu da se prilagodi zahtjevima korisnika. Uporabljivost predstavlja korisnikov zahtjev prema operacijskom sustavu za jednostavnijom komunikacijom između korisnika i operacijskog sustava i obratno.

2. Ulazno – izlazne operacije

Većina današnjih računalnih sustava zasniva se na koncepcijskom modelu kojeg je 1945. godine opisao mađarski matematičar i fizičar John von Neumann. Prema Von Neumannovom modelu, svako računalo mora imati ulazni dio, izlazni dio, radni ili glavni spremnik, aritmetičko-logičku jedinku i upravljačku jedinku.[4]

Spremnik je centralni dio računala te se u njega pohranjuju podaci i instrukcije koji se u računalo unose preko ulaznog dijela i svi rezultati operacija iz aritmetičko-logičke jedinke. Izlazni dio služi za prijenos rezultata izračunavanja okolini. Računalo koristi upravljačku jedinku za dohvaćanje i dekodiranje instrukcija iz spremnika te tako upravlja aritmetičko-logičkom jedinkom i ulaznim i izlaznim dijelovima. Upravljačka jedinka također određuje koje operacije treba izvoditi aritmetičko-logička jedinka. Ilustracija povezanosti svih navedenih dijelova prikazana je na slici 2.

[4] Budin L., Golub M., Jakobović D., Jelenković L. : Operacijski sustavi, 1. Izdanje, Zagreb 2010., (str. 9)



Slika 2. Funkcijski Von Neumannov model računala (*izvor: Budin L., Golub M., Jakobović D., Jelenković L. : Operacijski sustavi, 1. Izdanje, Zagreb 2010., (str. 10*))

Nakon prethodno kratkog opisa dijelova Von Neumannova modela računala, opisa međusobne povezanosti tih dijelova i njihove ilustracije, u daljnjem tekstu pozornost će se posvetiti ulazno-izlaznim napravama. Pažnja će biti usmjerena prema spajanju ulazno-izlaznih jedinica u računalo, prijenosu znakova i prekidnom radu procesora.

2.1. Spajanje ulazno-izlaznih jedinica u računalo

Ulazno-izlazne jedinice predstavljaju sve dijelove računala s pomoću kojih podsustavi računala komuniciraju s čovjekom ili s drugim računalom. Ulazna računalna jedinica predstavlja svaki uređaj koji omogaćava unos podataka ili programa iz okoline u računalo. Najčešće korišteni ulazni uređaji svakog računala jesu:

- Miš
- Zaslon osjetljiv na dodir
- Mikrofon
- Skener
- Digitalni fotoaparat

- Tipkovnica

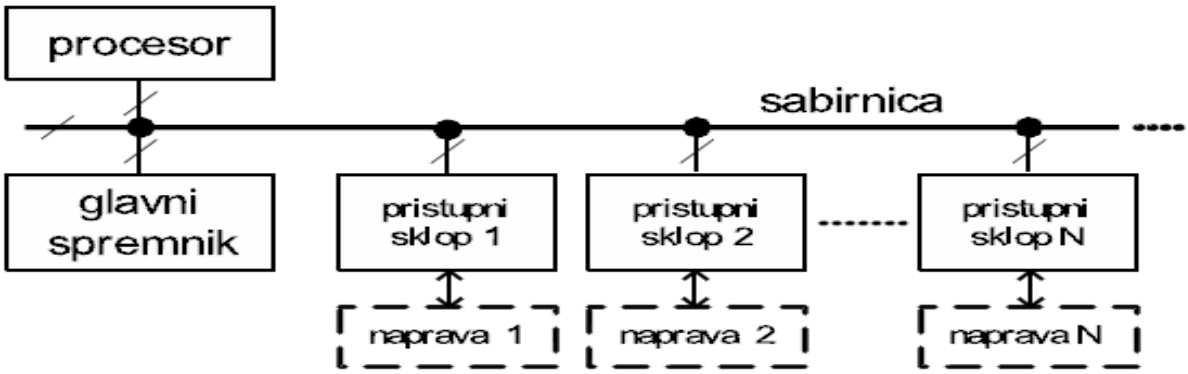
Izlazna računalna jedinica služi za pretvaranje binarno kodiranih informacija iz središnje jedinice u oblik razumljiv čovjeku i pogodan za korištenje stroju. U najčešće korištene izlazne uređaje spadaju:

- Monitor
- Pisač
- Zvučnik

Također, u skupinu ulazno-izlaznih naprava spadaju i uređaji za pohranu. Uređaji za pohranu jesu vanjski spremnici memorije koji služe za čuvanje računalnih podataka i programa, te njihov prijenos s jednog računala na drugo. Tu spadaju:

- Magnetska vrpca
- Magnetski disk
- Magnetska kartica
- Optički disk
- Memorijske kartice

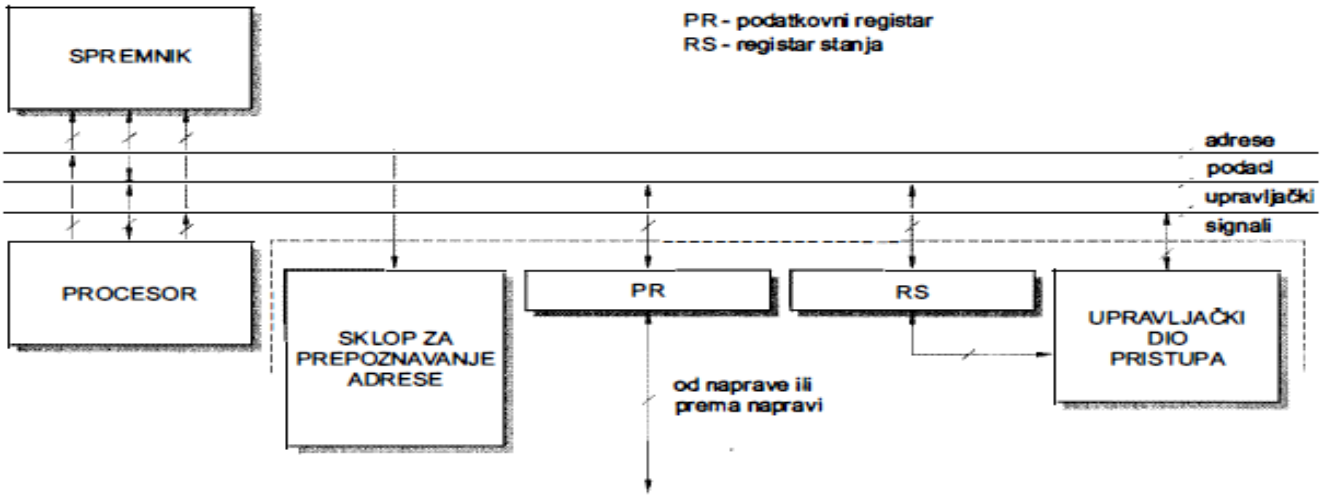
Navedene ulazno-izlazne naprave koje se spajaju u računalo razlikuju se u načinu i brzini rada te se iz tog razloga naprave ne spajaju izravno na glavnu sabirnicu. Naprava je spojena na međusklop, odnosno pristupni sklop računala koje je s jedne strane prilagođen toj napravi, a s druge strane prilagođava se protokolima sabirničkog sustava. Također, pristupni sklop omogućava sinkronizaciju rada procesora i ulazno-izlaznih naprava. Spajanje ulazno-izlaznih naprava na sabirnicu računala ilustrirano je na slici 3.



Slika 3. Spajanje ulazno-izlaznih naprava na sabirnicu računala (*Izvor: Jelenković: Operacijski sustavi – skripta 2014./2015.*)

2.2. Prijenos pojedinačnih znakova

Na sljedećoj slici, prikazan je detaljni shemaski prikaz pristupnog sklopa za prijenos pojedinačnih znakova. Na slici su prikazana dva registra: podatkovni registar pristupnog sklopa kojeg možemo promatrati kao jedan bajt spremnika i registar stanja u čije se bitove mogu pohraniti informacije o stanju pristupnog sklopa, kao i upravljački bitovi koji mogu modificirati ponašanje pristupa. Procesor te registre može adresirati jednako kao što adresira pojedine bajtove spremnika. Sklop za prepoznavanje adrese određuje koji je od registra adresiran, a posebni upravljački signal određuje da li se taj adresirani registar piše ili čita.



Slika 4. Shematski prikaz pristupnog sklopa za prijenos pojedinačnih znakova *(Izvor: Budin L. i ostali, Zagreb 2010., (str.35))*

Prilikom obavljanja izlazne operacije, odnosno prilikom prijenosa sadržaja nekog bajta spremnika prema izlaznoj napravi, procesor mora:

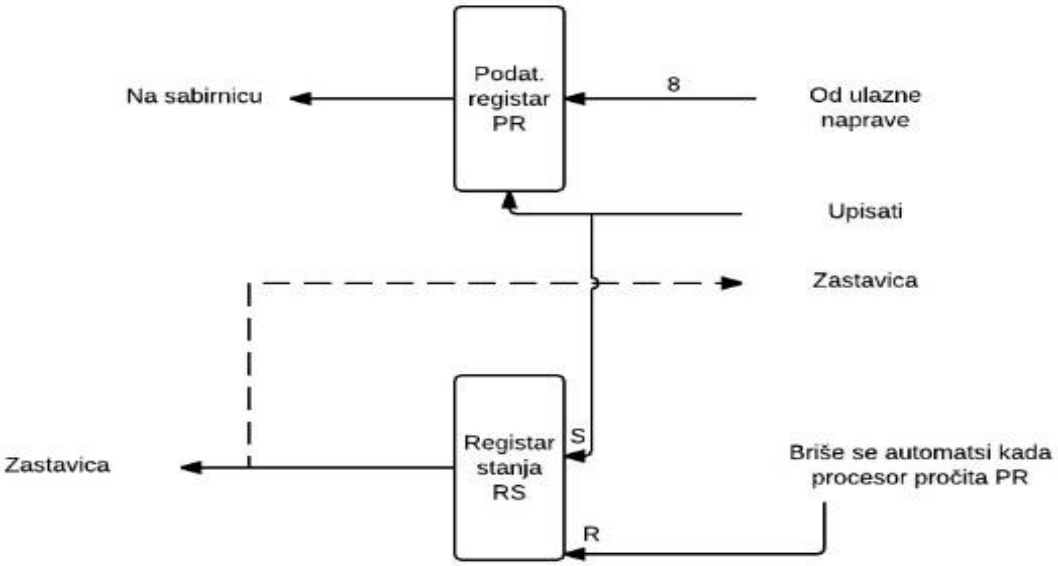
- Jednom instrukcijom dobaviti taj sadržaj iz spremnika u svoj registar
- Drugom instrukcijom prenijeti taj znak u podatkovni registar pristupnog sklopa[5]

Pristupni sklop i upravljačko sklopovlje naprave dužni su osigurati da se taj sadržaj prenese dalje prema izlaznoj napravi. Prilikom prenošenja uzastopnih znakova, procesor ne smije staviti sljedeći znak u podatkovni registar pristupnog sklopa ukoliko znak prije nije još

[5] Ibid str. 35.

potrošen. U registru stanja pristupnog sklopa izlazna zastavica, koja predstavlja posebni bit, ostaje postavljena onoliko dugo dok znak ne bude prihvaćen od strane izlazne naprave.

Na sličan takav način obavlja se i ulazna operacija, odnosno prenošenje jednog znaka iz ulazne naprave u radni spremnik. Upravo je na sljedećoj slici prikazan prijenos pojedinačnog znaka u radni spremnik.



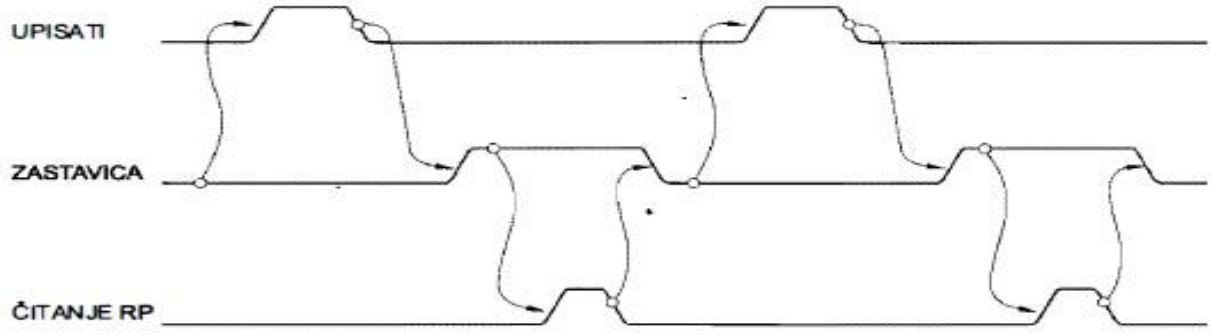
Slika 5. Shematski prikaz ulaza pojedinačnog znaka (*Izvor: Budin L. i ostali, Zagreb 2010., (str 36.)*)

U slijedećem tekstu detaljno je opisana slika 5. Od neke ulazne naprave dolazi osam vodiča preko kojih dolazi osam bitova jednog bajta. Dovedeni bitovi upisuju se u podatkovni registar (PR) tek u trenutku kada upravljački sklop naprave posebnim signalom to naredi. Istovremeno s upisom novog sadržaja u podatkovni registar postavlja se i posebni bit koji je označen kao *zastavica*, što je simbolizirano dovođenjem signala *upisati* do oznake S uz taj bit. Vrijednost posebnog bita *zastavica*, pretvara se u električni signal te se vraća natrag prema ulaznoj napravi. Upravljački sklop naprave mora biti napravljen tako da postavi signal *upisati* samo u trenutku kada je *zastavica* spuštena. Takvim mehanizmom stvorena je sinkronizacija sa strane ulazne naprave. Na taj će način ulazna naprava pohraniti novi znak u podatkovni registar samo onda kada je *zastavica* spuštena te će se automatski pri tom upisu podići.

Procesor sinkronizaciju obavlja izvođenjem kratkog programskog odsječka u kojem uzastopno čita registar stanja (RS) pristupnog sklopa onoliko dugo dok se ispitivanjem bita *zastavica* ne ustanovi da je on postavljen. Nakon toga treba pročitati podatkovni registar (PR).

Prenošenjem sadržaja registra PR u jedan od registra procesora, pristupni sklop postaje pripravan za prihvat novog znaka. Upravljački dio pristupnog sklopa pri čitanju registra PR automatski spušta bit *zastavica* u nulu te je to označeno oznakom R uz bit *zastavica*.

Vodiče nazvane *upisati* i *zastavica* moguće je vidjeti na izvodima pristupnog sklopa te se na njima nalaze električni signali. Ako bi se na njih spojio elektronički uređaj osciloskop[6], i ako bi se s njime promatralo što se događa na njima prilikom uzastopnog unošenja niza znakova, primjetile bi se promjene napona kao one ilustrirane na slici 6.



Slika 6. Ilustracija protokola dvožičnog rukovanja *(Izvor: Budin L. i ostali, Zagreb 2010., (str. 37.))*

Prva dva reda na slici 6. predstavljaju ilustraciju promjene napona. Strelice na slici označavaju uvjetovanost odnosno uzročnost pojedinih zbivanja. Kružić na niskoj razini signala *zastavica* označava da ulazna naprava smije generirati signal *upisati*. Kružić na padajućem bridu signala *upisati* uzrokuje podizanje signala *zastavica*, a podignuta *zastavica* predstavlja preduvjet za čitanje registra PR. Treći red predočava pojavu čitanja registra podataka te uzrokuje spuštanje signala *zastavica*.[7]

Takvo ponašanje pristupnog sklopa određuje protokol komuniciranja između ulazne naprave i računala koji je nazvan 'dvožično rukovanje'. Važno je napomenuti da se na sličan način može obavljati i prijenos pojedinačnih znakova iz računala prema izlaznoj napravi.

[6] Osciloskop - elektronički uređaj koji stvara dvodimenzionalni graf jedne ili više električkih potencijalnih razlika. Vodoravna linija predstavlja vrijeme, dok uspravna predstavlja električni napon, električnu struju ili neki drugi signal.
[7] Ibid str. 37. – 38.

2.3. Prekidni način rada procesora s jendim prekidnim ulazom

Sasvim je jasno da gore navedeni oblik prijenosa znakova ima svoje nedostatke. Veliki nedostatak je programsko ispitivanje bita *zastavica.* Procesor, čekajući podizanje signala *zastavica,* troši svoje vrijeme i sabirničke cikluse izvodeći bespotrebne instrukcije. Procesor bi za vrijeme čekanja na dolazak znakova mogao obavljati neku drugu dretvu[8], a kad bi pristigao novi znak u podatkovni registar pristupnog sklopa, bilo bi potrebno poslati posebno upozorenje kako bi se prekinulo izvođenje te dretve. Takvo se upozorenje procesoru ostvaruje mehanizmom prekidnog načina rada procesora.

Model takvog pristupnog sklopa treba biti modificiran na način da se podizanje bita *zastavica* u registru stanja poprati generiranim električkim signalom koji se pomoću posebnog vodiča dovodi do procesora. Procesor upravo taj signal prepoznaje kao upozorenje na dolazak novog znaka. Pojavom prekidnog signala procesor počinje sa sustavskim načinom rada. Na taj način pozivat će se podprogrami koji čine jezgru operacijskog sustava te se taj način rada naziva i "jezgreni način rada". Prilikom prelaska u jezgreni način rada, procesor djeluje na pet načina. Procesor prvo privremeno onemogući daljnje prekidanje. Nakon toga, djeluje na adresni dio sabirnice na način da adresira odvojeni dio spremnika. Tim postupkom, spremnik se dijeli na korisnički dio i sustavski dio. Također, aktivira drugi sustavski registar kazaljke stoga[9] koji je smješten u procesoru. Procesor također djeluje tako da sprema trenutačni sadržaj programskog brojila[10] u sustavski stog[11]. Peti način na koji procesor djeluje prilikom prelaska u jezgreni način rada jest da u programsko brojilo smješta adresu na kojoj počinje podprogram za obradu prekida. Adresa koja se mora staviti u programsko brojilo dobavlja se iz jednog od skrivenih registara unutar procesora. Na tu se adresu treba smjestiti prva instrukcija programskog odsječka u kojem će se obaviti posluživanje prekida.

Procesor u svojim registrima ima spremljene sadržaje koji čine trenutačni kontekst dretve koja se upravo izvodi, te se oni moraju sačuvati kako bi se prekinata dretva kasnije mogla nesmetano nastaviti. Takav je kontekst najjednostavnije spremiti na sustavski stog na kojem se već nalazi adresa programskog brojila koju je procesor automatski spremio prilikom pojave prekidnog signala. Nakon što podprogram za obradu prekida obavi svoje zadatke, potrebno je

[8] Dretva – Predstavlja najmanji slijed programskih instrukcija koji se mogu izvoditi samostalno.
[9] Registar kazaljke stoga – Predstavlja mjesto u koje se zapisuje najmanja adresa iz skupine uzastopnih bajtova.
[10] Programsko brojilo – Predstavlja registar koji sadrži adresu sljedeće instrukcije koju treba obaviti.
[11] Sustavski stog – Predstavlja stog koji je dostupan samo u nadzornikom načinu rada te omogućuje obradu prekida.

pokrenuti prekinutu dretvu korisničkog programa. Vraćanje prekinute dretve obavlja se na način da se prvo sa sustavskog stoga posebnim instrukcijama vraćaju sadržaji registra. Nakon te operacije na stogu ostaje samo sadržaj programskog brojila. Prije vraćanja tog sadržaja u procesor, procesoru se mora omogućiti prekid. Prekid se omogućuje posebnom instrukcijom koja djeluje tako da će prekidanje biti omogućeno tek nakon što se posebnom instrukcijom u procesor vrati sadržaj programskog brojila. Time možemo zaključiti da će novi prekid biti moguć tek u trenutku kada se procesor u potpunosti vrati na izvođenje dretve koja je prekinuta.

Nakon navedenog, može se zaključiti da će procesor u gore opisanom načinu rada izvoditi dvije dretve, a to su:

- Dretva koja se izvodi u korisničkom načinu rada
- Dretva koja se izvodi u jezgrenom načinu rada[12]

3. Međusobno isključivanje u višedretvenim sustavima

3.1. Općenito o procesima

Računalo obavlja neki korisni zadatak tako da izvodi programe pripremljene u višem programskom jeziku. Ti programi određuju sve aktivnosti koje računalni sustav treba obaviti. Aktivnosti se odnose na unošenje ulaznih podataka, prikaz izlaznih podataka te na način njihova pohranjivanja.[13]

Program koji je spremljen u strojnom obliku u računalnom spremniku predstavlja niz instrukcija koje procesor mora razumjeti i koje mora znati obaviti. Kada program počne s izvođenjem, njemu se mogu pripisati neka vremenska svojstva kao što su:

- Trenutak početka izvođenja;
- Trenutak završetka izvođenja;
- Trajanje izvođenja;
- Zaustavljanje izvođenja.

[12] Ibid str. 41.
[13] Ibid str. 61.

Samim time, izvođenje programa dobiva obilježje računalnog procesa. Procesi u raznim fazama svojeg izvođenja različito troše pojedine dijelove računalnog sustava, te se iz tog razloga posao mora organizirati tako da se u istom vremenskom razdoblju izvodi više zadataka, što dovodi do toga da više procesa može istodobno napredovati ako se izvode u različitim dijelovima računalnog sustava.

Izvođenje procesa obavlja se tako da procesor izvodi niz instrukcija. Kako bi se jednostavnije i lakše savladala složenost zasnivanja i izgradnje programskih sustava i kako bi se omogućilo bolje iskorištavanje računalnih sredstava sustava, pojedini se zadaci unutar jednog procesa dijele na podzadatke. To je dovelo do toga da se u suvremene operacijske sustave uvedu mehanizmi koji podržavaju izvođenje procesa s više dretvi, odnosno suvremeni operacijski sustavi postaju višedretveni.

3.2. Višedretveno ostvarenje zadataka

Organizacija programskih zadataka u višedretvenom radu morala bi biti osmišljena tako da podaci budu jasno razlučivi. Da bi se to ostvarilo, većina programa može se podijeliti na samo tri podzadatka, a to jesu:

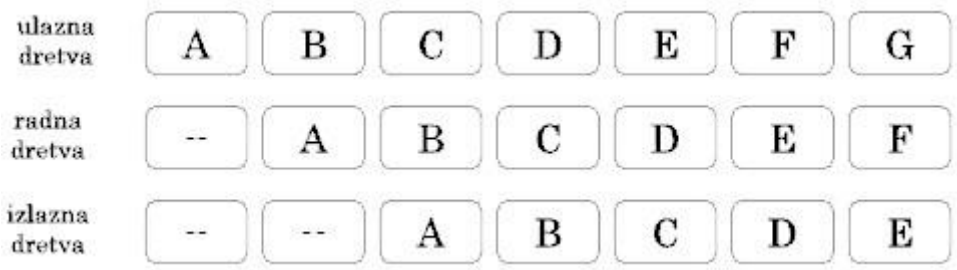
- Podzadatak za čitanje ulaznih podataka;
- Podzadatak za obradu podataka;
- Podzadatak za obavljanje izlaznih operacija.

Vidljivo je da se podzadaci unutar jednog zadatka moraju obavljati redoslijedom koji je upravo naveden. Ako bi se isti programski zadatak ponavljao, tada se pojedini podzadaci mogu obavljati istovremeno. Za obavljanje takvog posla bile bi dovoljne tri dretve:

- Ulazna dretva
- Radna dretva
- Izlazna dretva

Ulazna dretva bila bi prva i njen zadatak bio bi dobavljanje ulaznih podataka, proslijeđivanje istih radnoj dretvi te ponovno dobavljanje novih ulaznih podataka. Radna dretva bila bi druga, a njen zadatak bio bi preuzimanje podataka od ulazne dretve, obrada tih podataka i predaja rezultata obrade izlaznoj dretvi. Nakon toga bila bi spremna za novo preuzimanje podataka od

ulazne dretve. Posljednja dretva je izlazna dretva i njen zadatak je preuzimanje rezultata od strane radne dretve, obavljanje izlazne operacije i vraćanje na preuzimanje sljedećih rezultata. Slika 7. prikazuje upravo iznad opisano princip cjevovodnog rada dretvi.



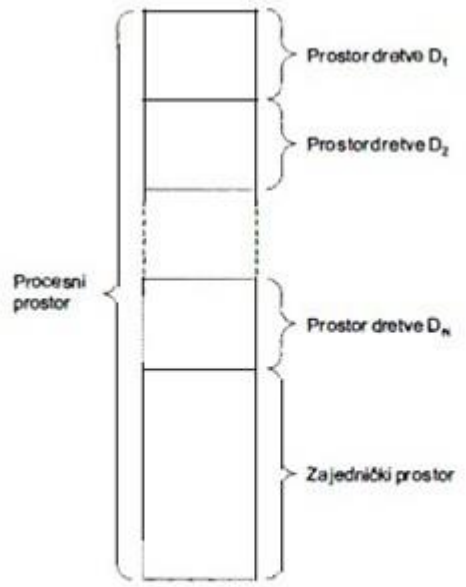
Slika 7. Princip cjevovodnog rada dretvi (*Izvor: Leonardo Jelenković: Materijal za predavanja: Operacijski sustavi, 2014./2015.)*

Načelo cjevovodnog rada koristi se i u procesorima, gdje se preklapaju pojedine faze izvođenja uzastopnih instrukcija. Ponašanje sklopovlja u procesorima može se namjestiti tako da pojedine faze traju jednako i da se izvođenje svake faze obavlja u različitim sklopovskim dijelovima. Prilikom obavljanja zadatka u višedretvenom načinu, koji je opisan u tekstu, takvo se ponašanje vrlo vjerojatno neće dogoditi. Istodobnost bi se mogla postići ukoliko bi se na raspolaganju imalo tri procesora. Tako bi svaka operacija bila prepuštena jednom procesoru.
Sada dolazimo do problema sinkronizacije. Ne može se očekivati da će pojedine faze navedenog zadatka imati jednaku dužinu trajanja, te je iz tog razloga potrebno predvidjeti mehanizme sinkronizacije dretvi. Prva, ulazna dretva, mora dojaviti drugoj, radnoj dretvi, da je dobavila ulazne podatke. Radna dretva u tom trenutku može biti zauzeta nekom drugom radnjom, te će dobavljeni podaci morati biti privremeno pohranjeni. Također, postoji mogućnost da je radna dretva obavila sve svoje zadatke, te mora čekati na dolazak novih podatka. Isto tako, mora se uskladiti rad radne dretve i izlazne dretve.

3.2.1. Model višedretvenosti

Krenimo od pretpostavke da su sve dretve unutar istog procesa. Tada sve dretve dijele njegov spremnički prostor, te svaka dretva ima skup instrukcija, skup podataka i vlastiti stog. Između ostalog, svim dretvama procesa na raspolaganju stoji adresni prostor procesa.

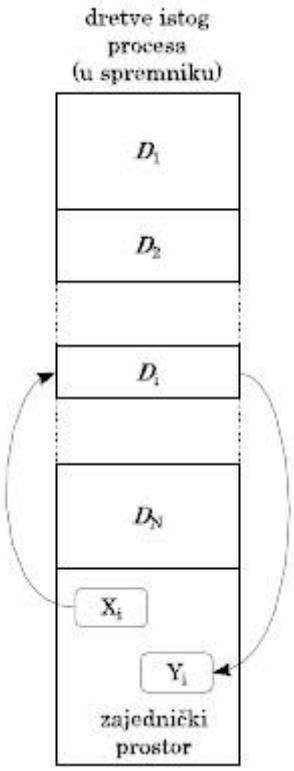
Svaka dretva zauzima svoj dio korisničkog spremničkog prostora. Taj prostor podijeljen je na dio u koji su smještene instrukcije dretve, dio u kojem je smješten stog dretve, te na dio u kojem su smješteni lokalni podaci dretve[14]. Procesni adresni prostor kojeg, kao što je već navedeno, mogu koristiti sve dretve, sastoji se od još jednog zajedničkog prostora u koji se mogu smjestiti zajedničke varijable svih dretvi. Takav adresni prostor, prikazan je na slici 8.



Slika 8. Podjela procesnog adresnog prostora *(Izvor: Budin L. i ostali, Zagreb 2010. (str.64.))*

Prilikom razmatranja višedretvenosti možemo pretpostaviti da je u sustavu osigurana pravilna promjena konteksta dretvi i da se dretve nesmetano mogu izvoditi, prekidati i nastavljati. Shvaćanje ponašanja dretvi možemo jednostavnije shvatiti kroz jednostavan primjer. Pretpostavimo da dretva D_i kreće s čitanjem podataka iz podskupa koji se nalazi u zajedničkom dijelu spremnika, odnosno iz svoje domene, koju možemo nazvati X_i. Na kraju čitanja podataka, dretva će pisati svoje rezultate u jedan podskup koji se nalazi u zajedničkom dijelu spremnika – u svoju kodomenu nazvanu Y_i. Iz navedenog, možemo zaključiti da dretva nakon obavljene obrade podataka, koristeći samo svoj dio adresnog prostora, završava i zapisuje rezultate u svoju kodomenu. Slika 9. prikazuje ilustraciju na kojoj dretva čita ulazne podatke iz dvoje domene X_i i zapisuje svoje rezultate u svoju kodomenu Y_i.

[14] Ibid str. 64.

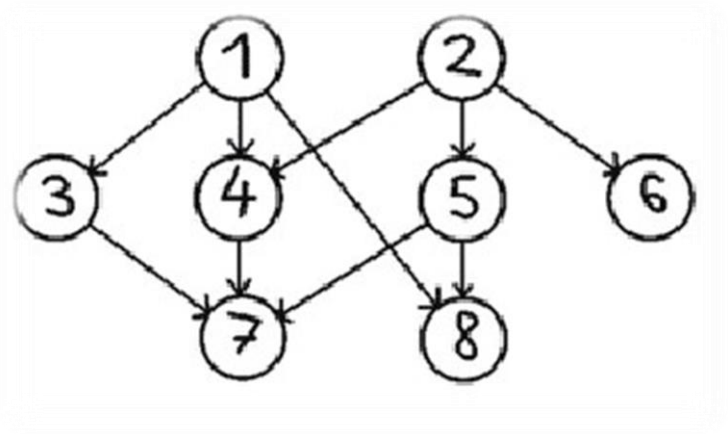


Slika 9. Dretva, domena i kodomena (*Izvor: Leonardo Jelenković: Materijal za predavanja: Operacijski sustavi, 2014./2015.)*

Dva podzadatka jednog zadatka ne moraju biti međusobno povezana. Dva su pozadatka međusobno nezavisna ako nemaju zajedničkih lokacija u svojim domenama i kodomenama, te se ona mogu izvoditi bilo kakvim redoslijedom. U drugom slučaju, podzadaci mogu ovisiti jedan o drugome. Redoslijed izvođenja postaje bitan ukoliko jedan podzadatak čita svoje ulazne podatke iz lokacija u koji neki drugi podzadatak piše svoje rezultate.

3.2.2. Sustav dretvi

Programski zadaci u pravilu se stastoje od više podzadataka, te se u takvom sustavu mora za svaki par zadataka utvrditi zavisnost ili nezavisnost, kako bi se mogao utvrditi redoslijed izvođenja. Sustav podzadataka pretvorit će se u sustav dretvi te će zajedničkim djelovanjem obaviti zadani zadatak. Ako se zadatak može rastaviti na lanac podzadataka tada se ramatranjem njihovih domena i kodomena mogu utvrditi međusobno zavisni i nezavisni zadaci te se sustav može prikazati usmjerenim grafom koji to uzima u obzir. Na slici 10. prikazan je primjer sustava podzadataka.



Slika 10. Grafički prikaz podzadatka (*Izvor: Leonardo Jelenković: Materijal za predavanja: Operacijski sustavi, 2014./2015.)*

Na prikazanom primjeru možemo vidjeti da se dretva 1 mora obaviti prije dretve 3, 4, 8 i 7. Dretve koje se nalaze na istom putu međusobno su zavisne i moraju se izvoditi određenim redoslijedom. Dretve koje se nalaze na različitim putevima su nezavisne i mogu se izvoditi paralelno, odnosno proizvoljnim redoslijedom. Nezavisne dretve mogu biti izvođene istodobno ako na raspolaganju imaju više fizičkih procesora ili prividno istodobno ako se izvode na istom procesoru. Kako bi bilo moguće izvesti navedeni posao, odnosno kako bi se omogućilo pokretanje, zaustavljanje i komunikacija među dretvama, potrebno je imati mehanizme sinkronizacije. Također, međusobno zavisne dretve moraju imati nekoliko načina razmjene podataka. Jedan od najjednostavnijih načina jest taj da jedna dretva neposredno piše u domenu druge dretve.

3.2.3. Međusobno isključivanje

Važno je na raspolaganju imati mehanizam međusobnog isključivanja koji osigurava da se određena sredstva računalnog sustava koriste pojedinačno. Dijelovi dretvi koji koriste određeno zajedničko sredstvo nazivaju se kritičkim odsječcima[15]. Inače, nezavisne dretve smiju prolaziti kroz svoj kritički odsječak samo pojedinačno. Dakle, kritični odsječci predstavljaju neprekidne dijelove obrade prekida. Kad se to uzme u obzir, ako pretpostavimo da se dretve izvode na jednoprocesorskom sustavu, dretva može osigurati svoj isključivi rad

[15] Ibid str. 68.

tako da onemogući prekidanje te tako osigurava da nijedan prekid ne može proći do procesora. Takva zabrana prekidanja riješit će problem samo u jednoprocesorskom sustavu.

4. Jezgra operacijskog sustava

Jezgra operacijskog sustava centralni je dio modernog operacijskog sustava. Jezgrina zadaća je upravljanje sklopovljem na najnižoj mogućoj razini kao i pružanje različitih usluga procesima i programima. Jezgra upravlja računalnim resursima i predstavlja sloj između fizičkog računalnog sklopovlja i korisničkih programa. Takav sloj nazivamo i jezgrenom okolinom, a razlog postojanja takvog sloja je u dizajnu većine modernih operacijskih sustava.[16]

Jezgra operacijskog sustava izvršava se u nadglednom načinu rada što znači da ima ovlasti nad svim sklopovljem računala. Jezgra operacijskog sustava može mijenjati sadržaje registra centralnog procesora, te može upravljati prekidima i pristupati ovlaštenim adresnim prostorima.

Jezgra operacijskog sustava mora omogućiti izvršavanje više programa, odnosno, procesa pri čemu je također potrebna sinkronizacija procesa, komunikacija među procesima te dijeljenje memorije među procesima. Jezgra operacijskog sustava mora se brinuti da svakom programu bude dodijeljen vlastiti adresni prostor te da se može izvršavati konkurentno s ostalim programima. Također, zadaće jezgre operacijskog sustava jesu upravljanje memorijom i briga o stanju svih ulazno/izlaznih jedinica. [17]

4.1. Jezgrin jednostavan model

Operacijski sustav prije pokretanja procesa mora osigurati kompletno okruženje za njegovo izvođenje. Operacijski sustav procesu dodjeljuje potrebni adresni prostor spremnika te s njim povezuje potrebne datoteke i ulazno – izlazne naprave. Prilikom pokretanja

[16] http://sistemac.carnet.hr/node/76, 27. svibanj 2015.
[17] http://sistemac.carnet.hr/node/78, 27. svibanj 2015.

programa u računalnom sustavu nastaje proces koji se sastoji od najmanje jedne dretve, odnosno od najmanje jednog niza instrukcija.

Jezgra se sastoji od skupine funkcija koje se pozivaju sklopovskim ili programskim prekidima. Komunikacija među dretvama izvodi se pozivanjem odgovarajuće jezgrine funkcije koja za njih obavlja zadani posao. Kada neka dretva želi obaviti ulaznu ili izlaznu operaciju ona to obavlja preko jezgre, a kada dretva obavi zadani posao, ona to javlja jezgri i jezgra obustavlja njezino izvođenje.

U daljnjem tekstu, kako bi se što bolje objasnile funkcije jezgre, govoriti će se o jednostavnim modelu jezgre. Jednostavni model jezgre zasnovan je na sljedećim pretpostavkama:

- U adresnom prostoru procesa smješteni su svi dretveni prostori;
- Čitav adresni prostor procesa dohvatljiv je svim dretvama;
- Izvođenje dretvi obavlja se u jednoprocesorskom sustavu;
- U sustavu postoje i ulazno – izlazne naprave koje ovdje uzimamo u razmatranje kako bismo jezgrom obuhvatili i obradu sklopovskih prekida;
- U sustavu djeluje sklopovski sat koji izaziva periodne prekide s periodom Tq.[18]

Periodni prekidi koje izaziva sklopovski sat služe za zaustavljanje izvođenje dretvi koje predugo traju, kao i za generiranje zadanih vremenskih intervala. U sustavu ćemo imati tri različite vrste prekida, a to su:

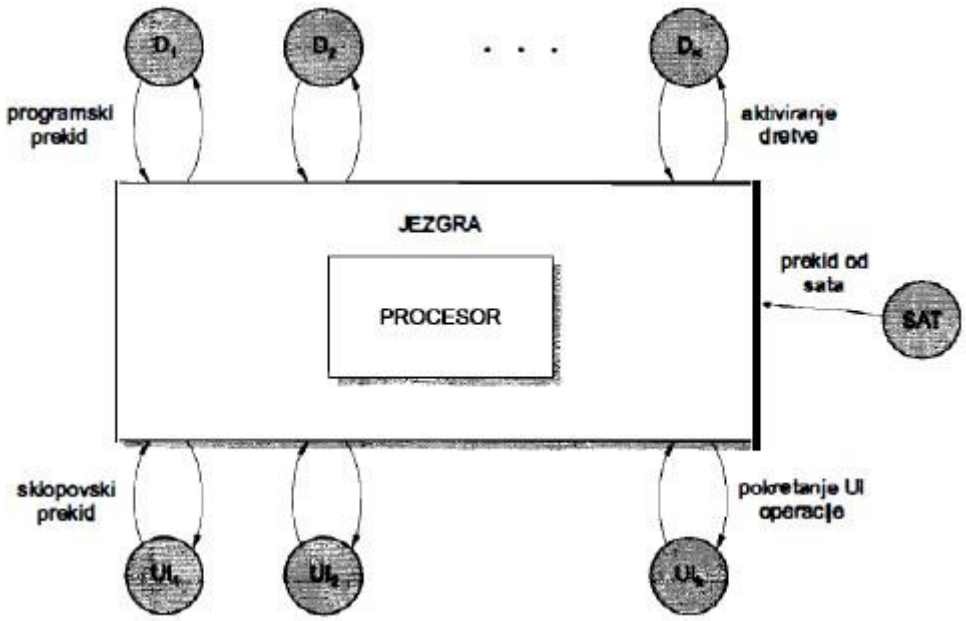
- Sklopovski prekidi od ulazno – izlaznih naprava;
- Periodni sklopovski prekidi od sata;
- Programski prekidi koje izazivaju dretve.

Navedene tri vrste prekida služe nam za povezivanje jezgrinih funkcija. Znamo da prekidom procesor ulazi u sustavski način rada s onemogućenim daljnjim prekidanjem. U sustavskom načinu rada, procesor adresira svoj sustavski adresni prostor te upotrebljava svoj sustavski registar kazaljke stoga. Sve potrebne strukture podataka i programski odsječci koji nam omogućuju ostvarenje jezgrinih funkcija nalaze se u sustavskom adresnom prostoru. Nakon što se izvede prekidom pozvana jezgrina funkcija, pokreće se dretva koju odabire jezgra. Dakle, unutar jezgre se moraju donositi odluke koje utječu na daljnje odvijanje dretvi, a takve

[18] Budin L., Golub M., Jakobović D., Jelenković L. : Operacijski sustavi, 1. Izdanje, Zagreb 2010., (str. 90.)

se odluke donose u skladu s određenim pravilima i na temelju odgovarajućih prekida o dretvama koje se čuvaju u strukturi podataka.

Može se reći da je jezgra prva hijerarhijski izgrađena razina iznad procesora. Na slici 12. Možemo vidjeti ilustrirani prikaz jezgre. Ona se sastoji od strukture podataka jezgre i jezgrinih funkcija. Prikazani krugovi s upisanim slovima D simboliziraju dretve. Strelice koje su usmjerene od dretve prema jezgri označavaju pozive jezgrinih funkcija koje su ostvarene programskim prekidima, a strelice koje su usmjerene od jezgre prema dretvi označavaju da dretve mogu biti pokrenute samo djelovanjem jezgre. Krugovi s upisanim slovima UI simboliziraju ulazno/izlazne naprave. Ulazno/izlazna operacija pokreće se iz jezgre operacijskog sustava te je simbolizirano strelicom usmjerenom od jezgre prema napravi, a strelice usmjerene od naprave prema jezgri simboliziraju prekid pokrenute operacije.



Slika 12. Ilustrirani prikaz jezgre *(Izvor: Budin L. i ostali, Zagreb 2010., (str. 91.))*

Na slici je, osim dretvi i ulazno – izlaznih naprava, prikazan sat. Sat prekidom poziva jezgrinu fukciju, te je to simboliziranom strelicom od sata prema jezgri.

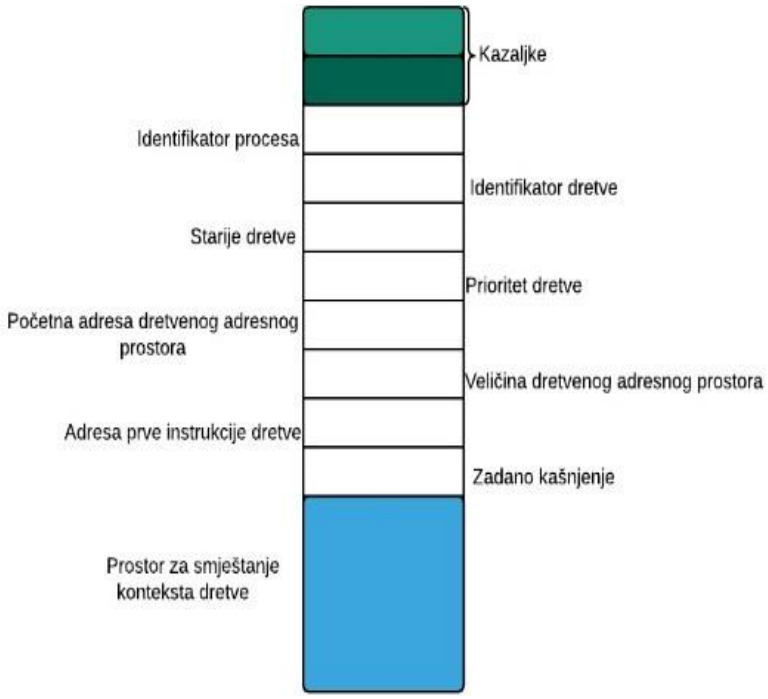
4.2. Jednostavni model jezgre – struktura podataka

U ovom odjeljku rada, opisat će se struktura podataka jednostavnog modela jezgre te će se opisati uloga pojedinih komponenti te strukture.

Kao što je već navedeno, jezgra se sastoji od svoje strukture podataka i funkcija smještenih u sustavskom dijelu spremnika. Sve informacije i podaci o dretvama koje su potrebne za donošenje odluka o njihovom izvođenju, moraju biti pohranjene u strukturi podataka jezgre. Podatke je najprikladnije smjestiti u jedan zapis koji se zove opisnik dretve. Osim podataka koji su važni za opis stanja dretve, u opisniku su predviđena i mjesta za smještanje kazaljki koje omogućuju lako premještanje iz jedne liste u drugu.

Slika 13. prikazuje sadržaj opisnika dretve. U opisniku dretve su uz mjesta za kazaljke predviđena i mjesta za sljedeće parametre:

- Identifikator procesa – označava kojem procesu pripada dretva
- Identifikator dretve – omogućuje međusobno razlikovanje dretvi
- Stanje dretve – parametar koji označava u kojem se stanju dretva trenutačno nalazi
- Prioritet dretve – parametar koji određuje prednost dretvi pri dodjeljivanju procesora
- Početna adresa dretvenog adresnog prostora i veličina dretvenog adresnog prostora– opisuju smještanje dretvenog adresnog prostora unutar procesnog adresnog prostora
- Adresa prve instrukcije – sadrži adresu na kojoj je smještena adresa prve instrukcije dretve
- Zadano kašnjenje – predstavlja parametar koji će odrediti odgađanje izvođenja dretve za zadani broj perioda otkucaja sata
- Prostor za smještanje konteksta – služi za smještanje konteksta dretve onda kada dretva bude prekinuta u svom izvođenju

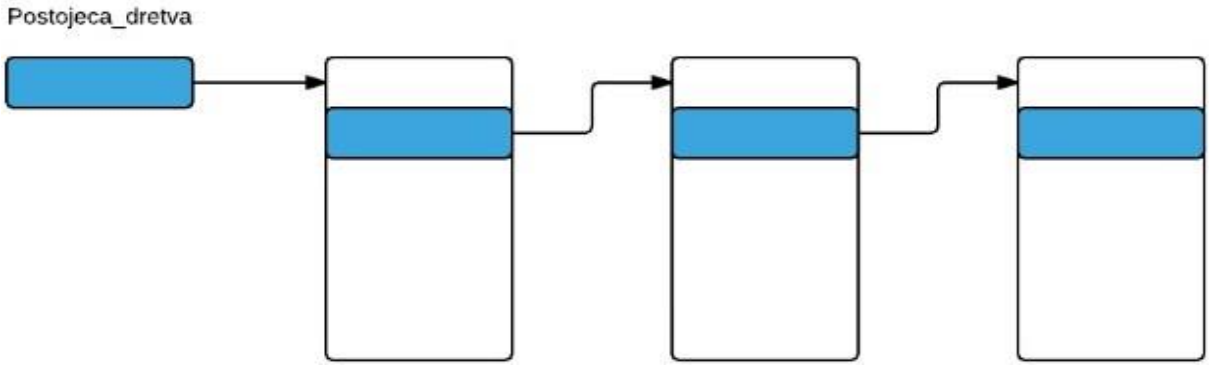


Slika 13. Sadržaj opisnika dretve *(Izvor: Budin L. i ostali, Zagreb 2010., (str. 92.))*

U samoj strukturi podataka jezgre mogu se stvoriti zaglavlja listi u koje se svrstavaju opisnici dretvi. Opisnici dretvi mogu se premještati iz jedne liste u drugu i to pomoću određenih jezgrinih funkcija, a smještanje opisnika dretve u pojedinu listu, odredit će njezino trenutačno stanje. Lista dretvi može se nalaziti u pasivnom, aktivnom, pripravnom i blokiranom stanju dretvi. O tim listama biti će, ukratko, riječ u nastavku ovog odjeljka.

4.2.1. Lista postojećih dretvi – pasivno, aktivno, pripravno i blokirano stanje

Dretve koje se nalaze u adresnom prostoru procesora poželjno je smjestiti u jednu listu kako bi se njihovi opisnici praktičnije mogli pregledavati. Jedna takva lista prikazana je na slici 14.

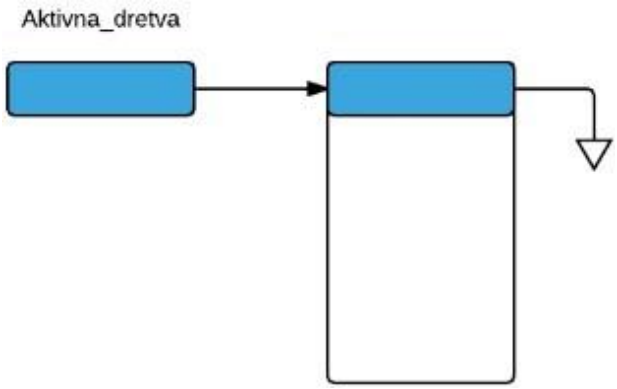


Slika 14. Lista postojećih dretvi *(Izvor: Budin L. i ostali, Zagreb 2010., (str. 93.))*

Prva po redu kazaljka svakog opisnika služi za povezivanje u druge liste te se iz tog razloga povezivanje u ovu listu obavilo drugom po redu kazaljkom opisnika. Tako smo postigli da možemo uvijek obići sve opisnike u listi postojećih dretvi neovisno u kojoj se on listi trenutačno nalazi.

Dretva se nalazi u *pasivnom* stanju ukoliko se njen opisnik nalazi isključivo u listi pod nazivom Postojeca_dretva.

Aktivna dretva je ona dretva čije instrukcije procesor upravo izvodi. U jednoprocesorskom sustavu to može biti samo jedna dretva. Jezgra će u posebnu listu postaviti opisnik te dretve. U toj aktivnoj listi smije se nalaziti samo jedan opisnik dretve. Jedna takva lista prikazana je na slici 15.



Slika 15. Lista aktivne dretve *(Izvor: Budin L. i ostali, Zagreb 2010., (str. 93.))*

Prilikom svakog pohranjivanja konteksta, sadržaji registara procesora smjestit će se na za to predviđeno mjesto unutar opisnika koji se nalazi u toj listi. Nakon smještanja, taj će se opisnik premjestiti u neku drugu listu, a u listu Aktivna_dretva premjestit će se opisnik koji upravo treba biti aktiviran.

Kao što je prije navedeno, u jednoprocesorskom sustavu samo je jedna dretva aktivna, dok sve ostale dretve moraju čekati na svoj red. Upravo je u ovom odjeljku riječ o redoslijedu izvođenja dretvi, odnosno o tome na koji se način odabire dretva koja će biti puštena u procesor.

Načina odabira dretve koja će biti puštena u procesor ima nekoliko. Najjednostavniji oblik raspoređivanja je taj da se odabiru dretve onim redoslijedom kojim su postajale pripravne za izvođenje. U drugom obliku raspoređivanja, odabiru se dretve prema njihovom prioritetu. Iz tog razloga potrebno je u strukturi podataka jezgre oblikovati red pripravnih dretvi.

Može se dogoditi da nema ni jedne dretve koja je pripravna za izvođenje. U tom trenutku, procesor aktivira latentnu dretvu koja je najnižeg prioriteta, te je njen zadatak trošenje vremena procesora. Latentna se dretva aktivira u trenutku kada se u redu pripravnih dretvi ne nalazi niti jedna druga dretva.

Dretve tijekom svog izvođenja mogu biti *blokirane* čekajući na ispunjenje nekog uvjeta za njihovo daljnje napredovanje[19]. Dretve mogu biti blokirane na nekoliko načina, a u ovom primjeru modela jezgre pretpostavit će se da dretve mogu biti blokirane na tri načina, a to su:

- Čekanjem na binarnom semaforu: Binarni semafor predstavlja jezgrin mehanizam koji omogućuje međusobno isključivanje dretvi[20], te se sastoji od jedne varijable koja može poprimiti dvije varijable (0 i 1).
- Čekanjem na općem semaforu: Opći semafor radi na istom principu kao i binarni semafor, ali se razlikuje po tome što njegova varijabla može poprimiti vrijednost cijelog broja
- Čekanjem na završetak ulazno/izlazne operacije.

4.3. Jezgrine funkcije

U daljnjem tekstu kratko će biti govora o dvjema jezgrinim funkcijama i to jezgrina funkcija 'ulazak u jezgru' i jezgrina funkcija 'izlazak iz jezgre'. Poziv jezgrine funkcije – *ulazak u jezgru* – zbiva se pod utjecajem sklopovskog ili programskog prekida[21]. Time je, u

[19] Ibid str. 97.
[20]Idem str. 97.
[21] Ibid str. 102.

jednoprocesorskom sustavu, osigurano međusobno isključeno obavljanje jezgrinih procedura. Prekidom pozvana jezgrina funkcija, mora na svome početku najprije premjestiti kontekst sa sustavskog stoga u opisnik dretve koji se nalazi u redu Aktivna_dretva te će se ta instrukcija opisati na sljedeći način:

Pohraniti kontekst u opisnik Aktivna_dretva;

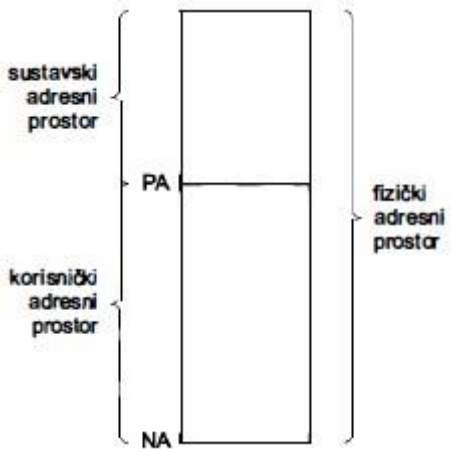
Prilikom praktične izvedbe jezgre pohranjivanje konteksta može se skratiti tako da se on smješta neposredno u opisnik aktivne dretve.

Sam *izlazak iz jezgre*, svodi se na aktiviranje jedne od dretvi. U većini slučajeva, dretva koja će se aktivirati biti će ona koja se nalazi na prvom mjestu u redu Pripravne_dretve. Postupak aktiviranja jedne od dretvi može se skraćeno opisati sljedećom instrukcijom:

Aktivirati prvu dretvu iz reda Pripravne_dretve

5. Upravljanje spremničkim prostorom

U dosadašnjem radu prilikom opisivanja procesa i dretvi procesa zaključili smo da svaki proces djeluje unutar svojeg spremničkog prostora, a ako se proces sastoji od više dretvi, onda se spremnički prostor procesa dijeli na dretvene prostore i jedan zajednički prostor u koji mogu pristupiti sve dretve. Nadalje, zna se da se svaki program prilikom izvođenja mora prevesti u strojni oblik te se mora pohraniti u radni spremnik i to u obliku strojnih instrukcija poznatih procesoru. Jedan dio radnog spremnika rezerviran je za strukture podataka i funkcije operacijskog sustava te se naziva "sustavski adresni prostor". Preostali dio radnog spremnika naziva se korisnički adresni prostor i koristi korisničkom načinu rada te određuje maksimalni mogući procesni adresni prostor.



Slika 16. Ilustracija fizičkog adresnog prostora (*Izvor: Budin L. i ostali, 1. Izdanje, Zagreb 2010., (str. 188.)*)

Slika 16. prikazuje fizički adresni prostor računala. Programi koji se planiraju izvoditi u takvom računalu moraju biti pripremljeni tako da se mogu smjestiti u "korisnički adresni prostor"[22]. Ukoliko postoji program koji je već smješten u korisničkom adresnom prostoru, ostali programi moraju čekati na svoje izvođenje. Svi programi se moraju u pripravnom obliku za izvođenje pohraniti u neki pomoćni spremnik ili dopunski vanjski spremnik kako bi se, kada za to dođe vrijeme, što brže mogli premjestiti u radni spremnik.

5.1. Magnetski diskovi

U današnje vrijeme, kao pomoćni spremnici najviše se koriste magnetski diskovi. Također, magnetski diskovi služe za smještanje datoteka, te kao temelj za ostvarivanje baza podataka, a samim time i informacijskih sustava.

Diskovi se na računalo priključuju pomoću prikladnog upravljačkog sklopovlja, a jedinka magnetskog diska sastoji se od dvije komponente. Prva komponenta je elektromehanički dio koji čine jedna ili više okruglih ploča koje se vrte konstantnom brzinom i mehanizma megnetskih glava koje se mogu pomicati iznad tih ploča. Druga komponenta jest upravljački sklop koji je stastavljen od mikroprocesora, spremnika, sučelja prema elektromehaničkom dijelu i sučelja prema sabirnici računala koje se ponaša kao pristupni sklop s neposrednim pristupom radnom spremniku.

[22] Ibid str. 188.

Slika 16. prikazuje paket magnetskih diskova i logičku organizaciju diska. Paket diskova međusobno je odvojen zračnim razmakom kako bi bio omogućen pristup glavi za čitanje/pisanje podataka. Logička organizacija diska podijeljena je na koncentrične staze, u koje se ravnomjerno raspoređuju podaci. Kao što je površina diska podijeljena na staze, tako je i staza podijeljena na sektore. Svi sektori na jednom disku jednake su veličine[23].



Slika 17. Prikaz paketa diskova i logička organizacija diska (*Izor: http://www.fpz.unizg.hr/hgold/ES/DE/m.%20disk.htm, 4. lipnja 2015.*)

5.2. Magnetski disk kao pomoćni spremnik

Možemo pretpostaviti da su svi programi pripravni za izvođene smješteni na vanjski pomoćni spremnik. Svaki od njih, prije početka izvođenja, mora biti prebačen u radni spremnik. Tim postupkom, program postaje proces sa svojim procesnim adresnim prostorom. Sve dretve koje pripadaju tom procesu koriste taj adresni prostor te se mogu u njemu nesmetano odvijati, ali taj procesni adresni prostor ne može biti veći od dijela spremnika koji je predviđen za korisnički način rada. Ako se sve dretve procesa koji zauzima radni spremnik blokiraju, biti će potrebno u radnom spremniku obaviti zamjenu programa.

Sličan mehanizam kao kod promjene konteksta kod prebacivanja procesora s izvođenja jedne dretve na drugu, ali ovdje moramo zamijeniti sadržaje svih spremničkih lokacija, pohraniti u pomoćni spremnik sadržaje lokacija procesa koji se prekida te napuniti lokacije sadržajima koji pripadaju procesu koji će započeti ili nastaviti s izvođenjem.

U tablicama koje se nalaze u operacijskom sustavu, za svaki proces mora postojati popis sektora diska u kojima se nalazi smješten program. Takvu tablicu možemo zamisliti kao poredak u kojem su redom pohranjeni brojevi sektora pomoćnog spremnika. Sektori jednog

[23] http://www.fpz.unizg.hr/hgold/ES/DE/m.%20disk.htm, 4. lipnja 2015.

programa ne moraju biti smješteni jedan pored drugog, ali je važno da budu čitani i smješteni u radni spremnik pravim redoslijedom.

5.3. Proces dodjeljivanja radnog spremnika straničenjem

5.3.1. Stranice i okviri

Logički adresni prostor je skup logičkih adresa, odnosno adresa koju generira procesor[24]. Logički adresni prostor programa naziva se i logičkim adresnim prostorom procesa. Sve su adrese unutar adresnog prostora logičke i započinju s nultom adresom. Tijekom izvođenja dretvi tog procesa, procesor će u svojim registrima generirati te logičke adrese. Fizičku adresu iz logičkih adresa određuje spremnički međusklop i to na putu iz registara procesora do fizičkog radnog spremnika. Logički adresni prostor dijeli se na jednako velike dijelove koje nazivamo stranicama[25]. Kako bi lakše shvatili takvu podjelu logičkog adresnog prostora, veličina stranice će biti potencija broja 2. Tada možemo m adresnih bitova podijeliti na dvije skupine bitova. Prva skupina bila bi skupina p bitova koji određuju adresu unutar pojedine stranice, a druga skupina bila bi skupina $r = m - p$ bitova koji određuju radni broj stranice u logičkom adresnom prostoru. Dakle, zaključno tome možemo reći da je veličina stranice jednaka 2^p i da se adrese unutar te stranice kreću u marginama od 0 do 2^{p-1}. Ukupno 2^r stranica moguće je adresirati u takvom logičkom adresnom prostoru.

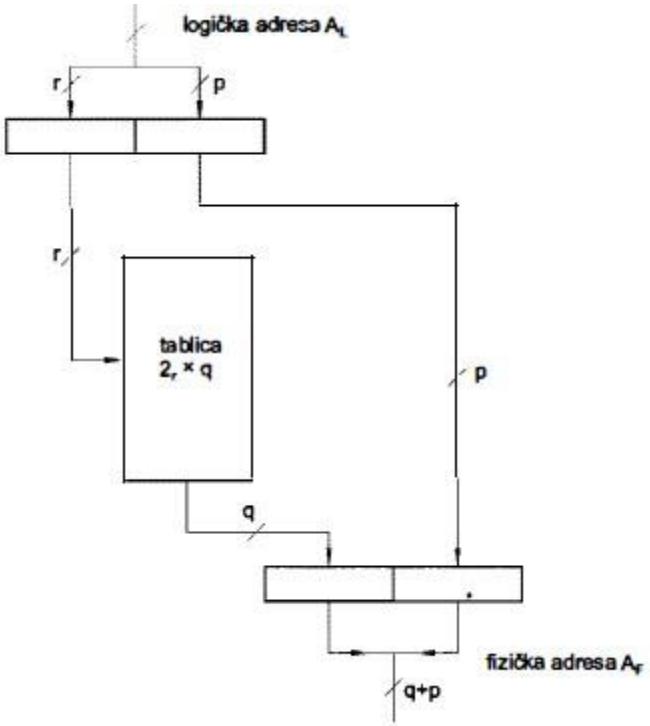
Fizički radni spremnik dijeli se na dijelove koji su jednake veličine kao i stranice logičkog adresnog prostora. Ti dijelovi zovu se okviri, a u jedan takav okvir može se smjestiti jedna stranica logičkog adresnog prostora. Broj postojećih okvira određuje veličina fizičkog spremnika računala. Ukoliko je broj postojećih okvira potencija broja 2, okviri se mogu adresirati s q bitova i to na način da se redni brojevi kreću u marginama od 0 do $2^q - 1$.

Prilikom dodjeljivanja okvira fizičkog spremnika može se javiti problem fragmentacije. Da bi se izbjegao takav problem stranice logičkog adresnog prostora mogu se smjestiti u okvire fizičkog spremnika proizvoljnim redoslijedom. Prilikom spremanja, u posebnoj tablici mora se voditi evidencija kako bi se znalo u kojem okviru je smještena pojedina stranica. Prilikom svakog pristupa fizičkom spremniku koristit će se ta tablica. Kako bi se prevođenje logičke adrese u fizičku adresu moglo obaviti što brže, tablica mora biti sastavni dio spremničkog

[24] http://www.zemris.fer.hr/predmeti/aior/Predavanja/PDF/11virtualna-memorija.pdf
[25] Budin L., Golub M., Jakobović D., Jelenković L. : Operacijski sustavi, 1. Izdanje, Zagreb 2010., (str. 227.)

međusklopa. Tablice se, radi svoje veličine, ne mogu u cijelosti spremiti unutar spremničkog međusklopa te se iz tog razloga samo jedan dio tablice sprema u priručni registar međusklopa.

Kako su veličine stranica i okvira potencije broja dva, samo prevođenje logičke adrese u fizičku adresu jednostavan je postupak. Logičku adresu prvog bajta u stranici čine redni brojevi stranica izraženi pomoću *r* bitova nadopunjeni s desne strane s *p* bitova s vrijednostima 0, dok relativnu adresu unutar stranice daju sadržaji donjih *p* bitova. Donji *p* bitovi, mogu se neposredno dovesti do fizičkog spremnika te će oni odrediti relativnu adresu unutar okvira. Početnu adresu okvira dobivamo tako da se rednom broju okvira, koji je izražen s *q* bitova, pridoda s desne strane *p* bitova koji imaju vrijednost 0. Možemo zaključiti da se prevođenje logičke adrese u fizičku adresu obavlja na način da se donjih *p* bitova logičke adrese propusti ka prema fizičkom spremniku, a gornji r bitovi logičke adrese zamijene s *q* bitovima fizičke adrese.



Slika 18. Tablično prevođenje fizičke adrese u logičku adresu jezgre *(Izvor: Budin L. i ostali, Zagreb 2010., (str. 213.))*

U primjeru je bilo pretpostavljeno da se sve stranice programa nalaze u okvirima fizičkog spremnika. Važno je napomenuti da u praksi to nije tako i da se sve stranice ne nalaze u okvirima fizičkog spremnika, već je samo jedan dio stranica smješten u fizički spremnik dok su ostale stranice pohranjene u vanjskom spremniku.

5.3.2. Virtualni spremnik

Virtualna memorija je strategija dodjele memorije koja dozvoljava da samo dio programa koji se izvodi bude u radnoj memoriji[26]. Osnovna prednost takvog pristupa je u tome što program može biti i veći od radne memorije. Tako korisnički program može poprimiti proizvoljnu veličinu, a sustav koji upravlja memorijom preslikava logički prostor korisnika u ograničeni prostor u radnoj memoriji.

U strukturi podataka operacijskog sustava za svaki proces postoji procesni informacijski blok koji sadrži opisnike dretvi i opisnik virtualnog procesnog adresnog prostora. U *m* – bitovnoj arhitekturi, procesni adresni prostor može zauzimati svih 2^m adresa, ali u praktičnim postupnica upravljanja spremničkim prostorom poželjno je ograničiti veličinu virtualnog adresnog prostora koji će se stvarno koristiti. U suprotnom, ako bi se cijeli adresni prostor dao na raspolaganje procesu, morale bi se osigurati velike tablice za prevođenje logičkih adresa u fizičke adrese. Iz tog razloga, razumno je logički adresni prostor ograničiti u skladu s potrebama pojedinih programa.

Programska pomagala koja pripremaju program osiguravaju osnovni adresni prostor za instrukcije i stog svake dretve, minimalni prostor za lokalne podatke i zajednički procesni podatkovni prostor koji se dijeli na statički dio i dinamički dio. Statički se dio tijekom izvođenja procesa ne mijenja, dok se dinamički dio koristi tijekom izvođenja za podatke koji nastaju i nestaju. Ti podaci mogu se spremati u isti adresni prostor, ali se prilikom smještanja posebnim API[27] funkcijama, koje su namijenjene za upravljanje spremničkim prostorom, može tražiti rezerviranje područja logičkih adresa te oslobađanje nekih područja logičkog adresnog prostora.

Opisnik virtualnog adresnog prostora procesa mora zadržavati tablicu koja opisuje rezervirani adresni prostor. Tablica može biti jednostavna i opisivati samo granice rezerviranih adresa.

Prilikom pokretanja programa mora se uspostaviti opisnik virtualnog adresnog prostora te se mora rezervirati odgovarajući prostor na disku. Nadalje, u radni spremnik se mora smjestiti najmanje jedna stranica koja sadrži početne instrukcije strojnog programa. Ostale stranice mogu se dohvatiti kada se ukaže potreba za njima, odnosno kada se generira adresa unutar

[26] http://www2.fsr.ba/nastava/os/kolokvij2/8-virtualna-memorija.pdf, 6. lipnja 2015.
[27] API – engl. Application Programming Interface (Aplikacijsko programsko sučelje), http://lab405.fesb.hr/igraf/Frames/fP2_4.htm, 6. lipnja 2015.

procesora kojom se traži pristup do određene stranice. Takav način straničenja naziva se straničenje na zahtjev.

Može se desiti da je veličina programa veća od samog raspoloživog fizičkog adresnog prostora, te će se s vremenom svi raspoloživi okviri ispuniti. Tada će punjenje nove stranice biti moguće nakon što se neki od punih okvira isprazne. Okvir će se isprazniti na način da se stranica koja je u njemu smještena prepiše natrag na disk.

Odabir stranice koja će morati biti izbačena iz svog okvira određuje se teorijski zasnovanim strategijama. Prva strategija je FIFO[28] strategija gdje se izbacuje ona stranica koja je najduže u radnom spremniku. Druga strategija je LRU[29] strategija gdje se izbacuje ona stranica koja se najduže ne koristi te se ona može samo približno primijeniti. Treća strategija je optimalna strategija ili OPT gdje se izbacuje ona stranica koja se neće najduže upotrebljavati. Ona je teorijski najprikladnija strategija ali je neostvariva jer bi se za njeno provođenje trebao znati cijeli proces.

6. Datotečni sustav

6.1. Smještanje datoteka na disku

Opisnikom je opisana svaka datoteka u datotečnom sustavu. Opisnik svake datoteke sadrži važne atribute koji mogu biti: naziv, tip datoteke, lozinka, ime vlasnika datoteke, prava pristupa, opis smještaja datoteke na vanjskom spremniku koji je jedan od najvažnijih atributa opisnika datoteke i dr.

Kako se na disk mogu adresirati samo sektori, datoteke je potrebno smještati na disk na način da ih se podijeli na blokove bajtova čija je veličina jednaka veličini sektora. Jedan fizički disk može se upotrebljavati ili kao jedan jedinstveni adresni prostor ili se on može podijeliti na logički razdvojene adresne potprostore, tzv. sveske.[30] Svaki pojedini svezak ima vlastitu datotečnu tablicu u kojoj su smješteni opisnici svih datoteka smještenih u tom svesku.

[28] FIFO – engl. first – in, first – out.
[29] LRU – engl. least recently used
[30] Budin L., Golub M., Jakobović D., Jelenković L. : Operacijski sustavi, 1. Izdanje, Zagreb 2010., (str. 246.)

Najbrži prijenos podataka s diska u radni spremnik i obrnuto moguće je postići kada su datoteke kompaktno smještene u uzastopne sektore na disku. Takav način smještanja datoteka izaziva fragmentaciju prostora sveska. Smještanje datoteka na disk tim načinom možemo usporediti s dinamičkim dodjeljivanjem radnog spremnika, te se radi toga kao rješenje nameće način smještanja datoteka koji podsjeća na dodjeljivanje radnog spremnika straničenjem.

Nadalje, 'logički adresni prostor' datoteke započinje nultom adresom i završava adresom za jedan manjom od duljine datoteke. 'Adresu' unutar adresnog prostora datoteke određuje datotečna kazaljka. Datoteka se može podijeliti na blokove, a veličina bloka ekvivalentna je veličini nakupine sektora diska. Na način na koji se stranice smještaju u okvire radnog spremnika tako se i blokovi razmještaju u nakupine sektora na disku.

6.2. Načela ostvarivanja datotečnih funkcija

Datotečni sustav koristeći zbirku API funkcija omogućava odvijanje standardnih operacija s datotekama. Neke standardne operacije jesu: stvaranje i uništavanje, otvaranje i zatvaranje, čitanje i pisanje.

Stvaranje nove datoteke predstavlja uvođenje novog opisnika u datotečnu tablicu te oblikovanje svih potrebnih podataka za dohvat datoteke.

Nakon stvaranja datoteke i prije njene upotrebe, datoteku je potrebnu otvoriti. Prilikom otvaranja, imenu datoteke dodaje se identifikator ID koji pomaže da se iz adresnog prostora procesa pristupi otvorenoj datoteci. Prilikom otvaranja datoteke potrebno je u adresni prostor operacijskog sustava s diska prenijeti aktivnu kopiju opisnika datoteke. Također, potrebno je u adresnom prostoru operacijskog sustava rezervirati međuspremnike u koje će se smjestiti dijelovi datoteka, te je potrebno uspostaviti konekciju između adresnog prostora procesa i otvorene datoteke pomoću identifikatora.

Nakon što se datoteka otvori potrebno ju je pročitati. Čitanje datoteka obavlja se pomoću funkcija sadržanih u bibliotekama pojedinih jezičnih procesora.

6.3. Posluživanje zahtjeva za pristup datotekama

Pristup svakoj datoteci odvija se putem zbirke API funkcija ili putem sučelja operacijskog sustava. Operacijski sustav direktno utječe na trajanje čekanja i to metodom određivanja redoslijeda posluživanja. U nastavu će biti opisano nekoliko metoda.

Najjednostavnija metoda raspoređivanja jest posluživanje redom prispijeća[31]. U toj metodi, zahtjevi se poslužuju redoslijedom kojim su i dolazili, a njen nedostatak je taj što često uzrokuje veliko prosječno čekanje.

Metoda posluživanja s najkraćim vremenom premještanja odnosi se na strategiju u kojoj se odabiru zahtjevi koji su najbliži trenutnoj poziciji glave za čitanje. Primjenom takve metode može doći do izgladnjivanja nekih zahtjeva, odnosno u teoriji se može dogoditi neodređeno dugo odgađanje posluživanja udaljenijeg zahtjeva.

Metoda posluživanja pregledavanjem odnosi se na metodu u kojoj se glava za čitanje pomiče u jednom smjeru i poslužuje sve zahtjeve na koje naiđe. Prosječno trajanje posluživanja može se znatno smanjiti ovom metodom, ali i ona ima svoje nedostatke. Može se dogoditi da novi zahtjev pristigne neposredno iza glave za čitanje te će morati pričekati promjenu smjera glave za čitanje.

Metoda posluživanja kružnim pregledavanjem slična je kao prethodno opisana metoda. Razlika je u tome što se glava pomiče u jednom smjeru i redom čita staze, a nakon što dođe do kraja staze, brzim se hodom vraća na početak i počinje novo pregledavanje.

Odabir metode kojom će se posluživati ovisi ponajprije o prirodi promatranog sustava i o načinu upotrebe datotečnog sustava, a sam učinak odabrane metode ovisi ponajviše o brojnosti i vrsti zahtjeva u sustavu. Metoda posluživanja zahtjeva ostvaruje se kao posebni modul unutar operacijskog sustava koji se po potrebi može prilagoditi ili promijeniti[32].

[31] Ibid str. 252.
[32] Ibid str. 255.

7. Podjela operacijskih sustava

Operacijski sustavi predstavljaju temeljni softver svakog računalnog sustava, te je njihova zadaća koordinacija hardvera i softvera. Operacijski sustavi mogu se podijeliti na više načina. Temeljna podjela uključuje broj računala, zadataka, procesora i korisnika koje operacijski sustav može posluživati, upravljati, odnosno izvršavati. Prema broju korisnika koje operacijski sustav može posluživati razlikuju se jednokorisnički te višekorisnički operacijski sustavi. Prema broju programa koje operacijski sustav može istovremeno izvršavati, operacijski sustav dijeli se na jednozadaćni i višezadaćni. Ukoliko operacijski sustav pruža potporu računalnom sustavu u radu s više procesora, tada govorimo o višeprocesorskom operacijskom sustavu. Konačno, operacijski sustav može biti samostojeći ukoliko upravlja samo s jednim računalom ili mrežni ukoliko upravlja s više umreženih računala.

7.1. Operacijski sustavi namijenjeni osobnim računalima i njihov razvoj

Ovisno o računalnim sustavim na kojima se izvršavaju, operacijske sustave možemo podijeliti na desktop i server. Server operacijski sustavi instalirani su na većinom na superračunalima, dok se desktop operacijski sustavi koriste za rad osobnih računala.

Većina današnjih osobnih računala i dalje je jednokorisničko, što znači da ima jednu osobu koja u datom trenutku koristi jednu aplikaciju. Operacijski sustavi namijenjeni za osobna računala izvršavaju naredbe tako da poslužuju samo jednog korisnika i izvršavajući samo jedan zadatak u tom određenom trenutku. Dok s druge strane, neka osobna računala koriste višekorisničke i višezadaćne operacijske sustave koji imaju mogućnost posluživanja dva ili više korisnika.

Što se tiče razvoja operacijskih sustava, DOS (diskovni operacijski sustav) je doživio procvat 1978. Godine prilikom razvoja mikroprocesora *Intel 8080* koji je koristio veću količinu memorije te je bio mnogo brži od prethodnih procesora. Seattle Computer Products pokrenuo je razvoj operacijskog sustava, kojeg su nazvali 80-DOS, a pravo distribucije dobio je Microsoft Corporation koji je poslije kreirao popularni MS DOS koji je prodavao po visokoj cijeni.

Microsoft Windows imao je cilj olakšanja korištenja MS DOS-a. Windows predstavlja operacijsko okruženje koje djeluje u svrhu proširenja mogućnosti DOS-a, na način na podržava višezadaćnost. Također, koristi programe koji upravljaju memorijom, sadrži GUI[33], a to sve olakšava uporabu računala. Windows za ulazni medij ne koristi više samo tipkovnicu, već omogućuje u uporabu miša. Windows, također, omogućuje umrežavanje, a aplikacije pisane za Windows su jednostavne za upotrebu.

Windows XP je operativni sustav za osobna računala proizveden od strane Microsofta. 1990-ih je započeo razvoj individualnih projekata "Odiseja" i "Neptun", koji su preteča Windows XP-a. Oba projekta su 2000. godine spojena u jedan pod nazivom "*Whistler*", koji je bio namijenjen potrošačkom i poslovnom tržištu. Krajem 2001., Whistler postaje Windows XP koji je napredniji od MS DOS-a po pitanju sigurnosti, stabilnosti i učinkovitosti. Windows XP uvodi znatno redizajnirano grafičko korisničko sučelje, te smanjuje softversko piratstvo.

Microsoft je prestao pružati podršku korisnicima Windows XP-a 2009. godine, a 2014. je potpuno prestala i produžena potpora koja je uključivala besplatnu tehničku podršku, osiguranje i izmjene dizajna.

[33] GUI - (eng. graphical user interface) grafičko korisničko sučelje.

Zaključak

U završnom radu detaljno je objašnjen pojma operacijski sustavi kao i njegove temeljne funkcije. Ukratko, operacijski sustavi predstavljaju skup programa koji omogućuju efikasno upravljanje računalnim hardverom, pružaju podršku za korištenje programa, te djeluju kao posrednik između korisnika računala i samog računalnog hardvera.

Analizirajući funkcije operacijskih sustava, može se zaključiti kako oni imaju ključnu ulogu u funkcioniranju svih računalnih sustava budući da bez operacijskih sustava, računalo ne može obavljati funkciju. U današnjem svijetu razvijene tehnologije, uloga operacijskih sustava postaje sve značajnija.

Glavna zadaća operacijskih sustava prvenstveno je optimizacija korištenja hardvera, a to uključuje upravljanje memorijom, izvršavanje zadaće, upravljanje diskom, grafički prikaz na zaslonu te čitanje/pisanje ulazno/izlaznih jedinica. Moderni operacijski sustav korisnicima danas omogućuje upravljanje računalom uporabom miša, obavljanje većeg broja programa istovremeno, jednostavnost korištenja, pouzdanost i stabilnost.

S obzirom na konstantno ažuriranje operacijskih sustava, nitko ne može jasno predvidjeti što će budućnost donijeti. Želje korisnika se mijenjaju na dnevnoj bazi, a smo oni operacijski sustavi koji u potpunosti prilagode potrebama korisnika, bit će uistinu uspješni.

Literatura

a) Knjige:

1. Budin L., Golub M., Jakobović D., Jelenković L. : Operacijski sustavi, 1. Izdanje, Zagreb 2010.
2. Silbershatz A., Galvin P.B., Gagne G.: Operating system concepts with Java, sixth edition, USA 2004.
3. B. Markić: Informatika, Ekonomski fakultet Sveučilište u Mostaru, 2008.

b) Web:

1. http://sistemac.carnet.hr/node/76, 27.5.2015.
2. http://sistemac.carnet.hr/node/78, 27.5.2015.
3. http://lab405.fesb.hr/igraf/Frames/fP2_4.htm, 6.6.2015.
4. http://www.zemris.fer.hr/predmeti/aior/Predavanja/PDF/11virtualna-memorija.pdf, 6.6.2015.
5. http://osnove.tel.fer.hr/LABOS/OSCILOF/Opocetna.htm, 19.5.2015.
6. http://www.fpz.unizg.hr/hgold/ES/DE/m.%20disk.htm, 28.5.2015.
7. http://www.zemris.fer.hr/~leonardo/os/dodatno/OS-skripta-za-srednje-skole/OS_uvod_1.pdf, 10.5.2015.
8. http://informatika.efos.hr/wp-content/uploads/2012/04/P5_UI_jedinice_nova.pdf, 14.5.2015.
9. http://web.math.pmf.unizg.hr/~borismil/os/vj2.html, 18.5.2015
10. http://www.3news.co.nz/technology/businesses-urged-to-ditch-xp-2013040910#axzz3jIl4OEAe, 15.8.2015.

Popis slika

Slika 1. Softver računalnog sustava

Slika 2. Funkcijski Von Neumannov model računala

Slika 3. Spajanje ulazno-izlaznih naprava na sabirnicu računala

Slika 4. Shematski prikaz pristupnog sklopa za prijenos pojedinačnih znakova

Slika 5. Shematski prikaz ulaza pojedinačnog znaka

Slika 6. Ilustracija protokola dvožičnog rukovanja

Slika 7. Princip cjevovodnog rada dretvi

Slika 8. Podjela procesnog adresnog prostora

Slika 9. Dretva, domena i kodomena

Slika 10. Grafički prikaz podzadatka

Slika 11. Sabirnički povezan višeprocesorski sustav

Slika 12. Ilustrirani prikaz jezgre

Slika 13. Sadržaj opisnika dretve

Slika 14. Lista postojećih dretvi

Slika 15. Lista aktivne dretve

Slika 16. Ilustracija fizičkog adresnog prostora

Slika 17. Prikaz paketa diskova i logička organizacija diska

Slika 18. Tablično prevođenje fizičke adrese u logičku adresu jezgre