



Alisa Bilal Zorić

GRAĐA RAČUNALA



Zaporešić, 2026.

Alisa Bilal Zorić

Grada računala

Nakladnik

Veleučilište s pravom javnosti *Baltazar Zaprešić*

Za nakladnika

Ivan Ružić

Recenzent

prof. dr. sc. Vladimir Šimović

Uredništvo

Ivan Ružić, glavni i odgovorni urednik

Gordana Šiber, izvršna urednica

Kristijan Čović, član

Alisa Bilal Zorić, članica

Bruno Raguž, član

Lektorica

Ana Klancir

Grafička urednica

Alisa Bilal Zorić

Podatak o izdanju

1. internetsko izdanje

Datum objave na mreži

17. rujna 2026.

Dostupno na

<https://www.bak.hr/studenti/knjiznica/baze-podataka-i-drugi-mrezni-izvori/mrezne-publikacije-veleucilista-baltazar-zapresic/>

ISBN

978-953-8037-29-0

©Veleučilište Baltazar Zaprešić

Alisa Bilal Zorić
GRADA RAČUNALA

Veleučilište Baltazar Zaprešić

Zaprešić, 2026.

Sadržaj

Predgovor	1
1 Uvod u građu računala	2
1.1 Osnovni pojmovi	2
1.2 Hardver i softver	2
Pitanja za ponavljanje	3
2 Povijesni razvoj računala	4
2.1 Rani alati i mehanička računala	4
2.2 Elektronička računala	8
2.3 Generacije računala	9
2.3.1 Prva generacija računala (elektronske cijevi)	9
2.3.2 Druga generacija računala (tranzistori)	10
2.3.3 Treća generacija računala (integrirani krugovi)	10
2.3.4 Četvrta generacija računala (osobna računala)	11
2.3.5 Peta generacija računala (umjetna inteligencija)	14
Pitanja za ponavljanje	15
3 Utjecaj Turingova stroja na razvoj računalstva	16
3.1 Motivacija	16
3.2 Turingov stroj	19
Pitanja za ponavljanje	23
4 Osnovni dijelovi računala	24
4.1 Memorija	25
4.2 Središnja procesorska jedinica (CPU)	29
4.2.1 Aritmetičko-logička jedinica (ALU)	31
4.2.2 RISC i CISC arhitektura procesora	31
4.3 Ulazno-izlazne jedinice (I/O)	33
4.3.1 Ulazni uređaji	34
4.3.2 Izlazni uređaji	34
4.4 Sabirnice	35
4.5 Operacijski sustav	37
Pitanja za ponavljanje	40
5 Booleova algebra i digitalna logika	41
5.1 Logika sudova	41

5.2	Logička vrata	44
5.3	Booleove jednakosti.....	46
5.4	Logički sklopovi	50
	Pitanja za ponavljanje	53
6	Zapis podataka u računalu.....	54
6.1	Brojevi sustavi.....	54
6.2	Brojevi sustavi – pretvorba	56
6.3	Prikaz negativnih brojeva.....	59
6.3.1	Prikaz realnih brojeva	61
6.4	Binarne operacije.....	62
6.5	Kod i kodiranje.....	65
	Pitanja za ponavljanje	68
7	Performanse i mjerenja	69
7.1	Takt procesora	69
7.2	Latencija.....	69
7.3	Protočnost.....	70
7.4	<i>Benchmarking</i> i usporedba sustava	71
	Pitanja za ponavljanje	73
8	Suvremeni trendovi u računalstvu	75
8.1	Superračunala.....	75
8.2	Kvantna računala	76
8.3	Energetski učinkoviti procesori i heterogene arhitekture	76
8.4	Post-silicijski razvoj i specijalizirane arhitekture	77
8.5	Green IT – održivost u računalnim sustavima.....	78
	Pitanja za ponavljanje	79
9	Literatura	81

Predgovor

Skripta *Građa računala* namijenjena je studentima prve godine prijediplomskog studija Informacijske tehnologije te služi kao temeljni nastavni materijal za istoimeni kolegij. Cilj skripte je upoznati studente s osnovnim načelima građe i funkcioniranja računalnih sustava, s naglaskom na razumijevanje uloge i međusobne povezanosti osnovnih hardverskih komponenti.

U skripti se obrađuju temeljni pojmovi računalne arhitekture, povijesni razvoj računala, brojevni sustavi i logičke osnove, središnja procesorska jedinica, memorijski i ulazno-izlazni sustavi te mjerenje performansi računalnih sustava. Posebna se pažnja posvećuje suvremenim arhitekturnim pristupima, energetske aspektima rada računala i metodama objektivne usporedbe sustava.

Sadržaj skripte osmišljen je tako da studentima omogući postupno usvajanje osnovnih koncepata, bez pretpostavke prethodnog dubinskog tehničkog predznanja. Naglasak je stavljen na razumijevanje načela rada računalnih sustava, a ne na detalje implementacije, čime se stvara čvrsta osnova za daljnje studijske kolegije iz područja informatike i informacijskih tehnologija.

1 Uvod u građu računala

Računalo je složen tehnički sustav koji omogućuje obradu podataka prema unaprijed zadanim uputama. Iako se u svakodnevnoj uporabi često doživljava kao jedinstvena cjelina, računalni sustav sastoji se od niza međusobno povezanih komponenti, od kojih svaka ima jasno definiranu ulogu. Razumijevanje građe računala podrazumijeva razumijevanje načina na koji te komponente surađuju kako bi omogućile izvršavanje programa i obradu informacija.

U ovom poglavlju uvode se osnovni pojmovi vezani uz računalni sustav te se objašnjava razlika između hardverskog i softverskog dijela računala. Ovi pojmovi predstavljaju temelj za razumijevanje svih kasnijih poglavlja skripte, u kojima se pojedine komponente obrađuju detaljnije.

1.1 Osnovni pojmovi

Računalni sustav je skup međusobno povezanih hardverskih i softverskih komponenti koje zajednički omogućuju unos podataka, njihovu obradu, pohranu i prikaz rezultata. Osnovna zadaća računalnog sustava jest pretvaranje ulaznih podataka u korisne informacije pomoću unaprijed definiranih postupaka.

Svaki računalni sustav, neovisno o veličini i namjeni, temelji se na istom osnovnom modelu rada. Taj model obuhvaća četiri osnovne funkcije: unos, obradu i pohranu podataka te izlaz rezultata. Unos se ostvaruje pomoću ulaznih uređaja, obrada se provodi unutar središnje procesorske jedinice, podaci i programi pohranjuju se u memorijskim jedinicama, dok se rezultati obrade prikazuju ili prosljeđuju putem izlaznih uređaja.

Važno je naglasiti da se računalni sustavi mogu značajno razlikovati po složenosti i snazi, od jednostavnih ugrađenih sustava do snažnih poslužitelja i superračunala. Unatoč tim razlikama, temeljna načela njihova rada ostaju ista, što omogućuje primjenu jedinstvenih koncepata u analizi građe i funkcioniranja različitih vrsta računala.

1.2 Hardver i softver

Računalni sustav sastoji se od dvije osnovne cjeline: hardvera i softvera. Hardver obuhvaća sve fizičke, opipljive dijelove računala, poput procesora, memorije, diskova, matične ploče te ulazno-izlaznih uređaja. Ove komponente čine tehničku osnovu sustava i omogućuju izvođenje računalnih operacija.

Softver predstavlja skup programa i podataka koji upravljaju radom hardvera i određuju kako će se računalni sustav ponašati. Bez softvera, hardver nema praktičnu uporabnu vrijednost, jer ne zna koje zadatke treba izvršavati. S druge strane, softver ne može funkcionirati bez hardvera koji osigurava fizičko izvršavanje instrukcija.

Uloga softvera može se promatrati na više razina. Operacijski sustav djeluje kao posrednik između korisnika, aplikacijskih programa i hardvera, omogućujući učinkovito upravljanje

resursima računala. Aplikacijski programi koriste usluge operacijskog sustava kako bi obavljali konkretne korisničke zadatke, dok se na najnižoj razini nalaze programi i upravljački programi koji izravno komuniciraju s hardverskim komponentama.

U nastavku skripte naglasak će biti stavljen na hardverske komponente računalnog sustava, uz objašnjenje njihove uloge u cjelokupnom procesu obrade podataka.

Pitanja za ponavljanje

1. Što je računalni sustav?
2. Koja je osnovna zadaća računalnog sustava?
3. Koje su četiri osnovne funkcije svakog računalnog sustava?
4. Koja je uloga ulaznih uređaja u računalnom sustavu?
5. Koja je uloga središnje procesorske jedinice u obradi podataka?
6. Koja je uloga memorije u računalnom sustavu?
7. Po čemu se računalni sustavi mogu razlikovati?
8. Što je hardver? Navedite nekoliko primjera.
9. Što je softver i koja je njegova uloga u radu računala?
10. Koja je uloga operacijskog sustava u računalnom sustavu?

2 Povijesni razvoj računala

Razvoj računala rezultat je dugotrajnog procesa u kojem su se ljudske potrebe za računanjem, pohranom podataka i automatizacijom postupno pretvarale u tehnička rješenja sve veće složenosti. Ovo poglavlje prikazuje kronološki razvoj računala – od najranijih pomagala za računanje, preko mehaničkih i elektroničkih strojeva, do suvremenih digitalnih sustava. Poseban naglasak stavljen je na tehnološke prekretnice koje su omogućile prijelaz iz jedne generacije računala u drugu.

2.1 Rani alati i mehanička računala

Računala kakva danas poznajemo nisu nastala naglo, već su rezultat tisućljetnog razvoja različitih pomagala i naprava za računanje. Iako su rani uređaji bili jednostavni i ograničenih mogućnosti, oni su imali presudnu ulogu u oblikovanju temelja računalne tehnologije.

Jedna od najstarijih poznatih naprava za računanje jest abak koji se koristio još prije više od 5000 godina u Babilonu. Abak se sastojao od okvira s kamenčićima ili kuglicama koje su se pomicale po žljebovima ili žicama, pri čemu je položaj pojedinog elementa predstavljao određenu brojčanu vrijednost. Na taj način omogućeno je izvođenje osnovnih aritmetičkih operacija, prvenstveno zbrajanja i oduzimanja, bez potrebe za pisanjem.



Slika 1 Abak (<https://www.pngegg.com/en/png-ctjms/download>)

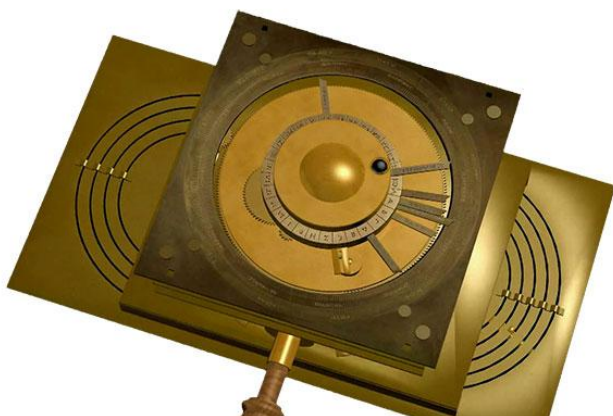
Razvijenije inačice abaka pojavile su se u Egiptu, Grčkoj i Kini, a posebno se ističe kineski abak, poznat kao suanpan, koji se koristio sve do modernog doba. Iako jednostavan, abak je omogućio standardizirano i znatno brže računanje, što ga čini jednim od prvih koraka prema razvoju računala.

Osim naprava za aritmetičko računanje, u povijesti nalazimo i složenije sustave koji su služili za praćenje prirodnih i astronomskih pojava. Jedan takav primjer je Stonehenge, megalitski spomenik u Engleskoj star više od 4000 godina. Smatra se da je korišten za astronomska promatranja, praćenje Mjesečevih mijena, solsticija i pomrčina, čime je omogućavao predviđanje vremenskih ciklusa.



Slika 2 Stonehenge (<https://www.stamfordstone.co.uk/history-of-stonehenge/>)

Izuzetno značajan primjer ranog „računalnog” sustava jest i mehanizam s Antikitere, starogrčki uređaj iz 2. stoljeća prije Krista, često nazivan najstarijim poznatim analognim računalom. Ovaj složeni mehanizam sastojao se od niza zupčanika i služio je za predviđanje kretanja nebeskih tijela, pomrčina i planetarnih ciklusa. Njegova konstrukcija svjedoči o iznimno visokoj razini tehničkog znanja antičke Grčke.



Slika 3 Mehanizam s Antikitere (<https://nova-akropola.com/mozaik/zanimljivosti/mehanizam-iz-antikitere/>)

Iako primitivne u usporedbi s današnjim računalima, ove preteče jasno pokazuju da je potreba za računanjem, predviđanjem i automatizacijom bila prisutna tisućama godina prije pojave moderne tehnologije.

Sedamnaesto stoljeće – prva prekretnica

Sedamnaesto stoljeće predstavlja jednu od prvih velikih prekretnica u povijesti računanja, ponajprije zahvaljujući razvoju novih matematičkih metoda koje su znatno ubrzale i pojednostavnile izvođenje složenih računa. Najvažniji doprinos tog razdoblja dao je škotski matematičar John Napier koji je 1614. godine uveo pojam logaritama i objavio prve logaritamske tablice.

Napierova ideja temeljila se na činjenici da se množenje može svesti na zbrajanje, a dijeljenje na oduzimanje logaritama. Time su složene i dugotrajne računske operacije

postale znatno jednostavnije i brže. Primjena logaritamskih tablica omogućila je efikasno izvođenje množenja, dijeljenja, potenciranja i korjenovanja, što je imalo velik utjecaj na razvoj matematike, astronomije i inženjerstva.

Iz Napierovih ideja proizašao je i razvoj logaritamskog računala, poznatog kao šiber. Šiber se temeljio na pomičnim logaritamskim skalama koje su omogućavale brzo izvođenje složenih računa bez potrebe za detaljnim ručnim izračunima. Iako je danas u potpunosti zamijenjen digitalnim kalkulatorima, šiber je stoljećima bio osnovni alat znanstvenika i inženjera.



Slika 4 Šiber (<https://www.kgz.hr/hr/dogadjanja/logaritamsko-racunalo/20481>)

Uvođenje logaritama može se smatrati prvim korakom prema automatizaciji računanja i razvoju preciznijih računskih naprava, čime je postavljen temelj za kasnije mehaničke i elektroničke sustave.

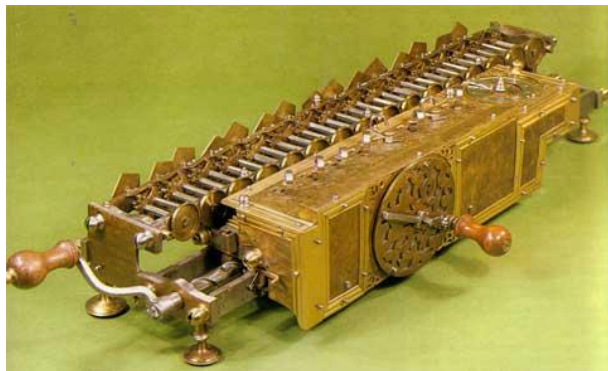
Mehanički strojevi za računanje

Razvoj mehaničkih strojeva za računanje započeo je u 17. stoljeću, paralelno s razvojem tehničkih znanosti. Prvi značajan mehanički uređaj konstruirao je Blaise Pascal 1642. godine. Njegov stroj, poznat kao Pascalina, koristio je sustav zupčanika i kotačića za automatsko zbrajanje i oduzimanje brojeva. Unos podataka vršio se okretanjem kotačića, dok je mehanizam prijenosa osiguravao ispravno prikazivanje rezultata.



Slika 5 Pascalina (<https://en.wikipedia.org/wiki/Pascaline>)

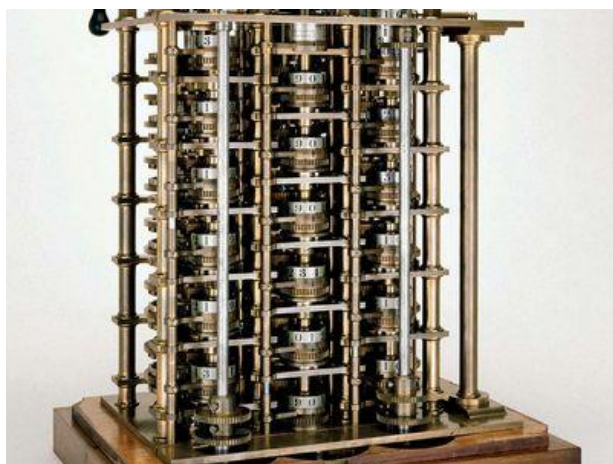
Iako inovativna, Pascalina je imala ograničene mogućnosti jer nije mogla izvoditi množenje i dijeljenje. Ovaj nedostatak pokušao je ukloniti Gottfried Wilhelm von Leibniz koji je 1672. godine razvio mehanički stroj sposoban za izvođenje svih osnovnih aritmetičkih operacija, uključujući vađenje korijena. Međutim, zbog tehničkih ograničenja i nepreciznosti mehanizama, Leibnizov stroj nije bio pouzdan u praksi.



Slika 6 Leibnizov mehanički stroj

(https://commons.wikimedia.org/wiki/File:Rechenmaschine_von_Leibniz_%28Nachbau%29_07.jpg)

Najznačajniji doprinos mehaničkom računanju dao je Charles Babbage u 19. stoljeću. Njegov diferencijalni stroj bio je namijenjen izradi matematičkih tablica, dok je kasniji analitički stroj sadržavao koncepte koji su temelj modernih računala: memoriju, procesorsku jedinicu, ulazne i izlazne uređaje. Analitički stroj bio je zamišljen kao programabilni sustav koji bi koristio bušene kartice, no nikada nije u potpunosti realiziran.



Slika 7 Babbageov diferencijalni stroj (<https://www.britannica.com/technology/Difference-Engine>)

U razvoju Babbageovih ideja važnu je ulogu imala Ada Lovelace koja je prva opisala algoritam namijenjen izvođenju na stroju. Zbog toga se smatra prvom programerkom u povijesti. Mehanički strojevi za računanje predstavljali su važan korak prema konceptu automatiziranog računanja, iako su ih u konačnici zamijenili elektronički sustavi.

2.2 Elektronička računala

Prijelaz s mehaničkih na elektroničke sustave omogućila je primjena električne energije i razvoj elektronskih cijevi. Elektronske cijevi omogućile su pojačavanje i kontrolu električnih signala, čime je postalo moguće izvoditi računske operacije pomoću električnih impulsa.

Jedan od prvih sustava koji je koristio električnu energiju za obradu podataka razvio je Herman Hollerith krajem 19. stoljeća. Njegov električni sortirni stroj koristio je bušene kartice za unos podataka i omogućio je brzu obradu velikih količina informacija. Najpoznatija primjena bila je obrada popisa stanovništva SAD-a 1890. godine koja je zahvaljujući ovom stroju trajala znatno kraće nego prethodni popisi.



Slika 8 Sortirni stroj (<https://www.hnf.de/en/permanent-exhibition/exhibition-areas/hall-of-fame/herman-hollerith-1860-1929.html>)

Hollerithov uspjeh doveo je do komercijalne proizvodnje strojeva za obradu podataka i osnivanja tvrtki koje su kasnije činile jezgru današnje kompanije IBM. Ovi sustavi postavili su temelje za razvoj računalne industrije i pokazali potencijal elektroničke obrade podataka.

Drugi svjetski rat – druga prekretnica

Drugi svjetski rat snažno je ubrzao razvoj računalne tehnologije. Vojne potrebe, osobito u balistici i kriptografiji, zahtijevale su brzo izvođenje velikog broja složenih izračuna. Mehanički strojevi nisu mogli zadovoljiti te zahtjeve, što je dovelo do razvoja novih elektromehaničkih i elektroničkih sustava.

Jedan od prvih takvih sustava bio je MARK I, elektromehaničko programibilno računalo razvijeno 1944. godine na Sveučilištu Harvard u suradnji s IBM-om. Iako spor u usporedbi s kasnijim sustavima, MARK I je omogućio automatizirano izvođenje dugotrajnih izračuna.



Slika 9 Mark 1 (<https://www.thoughtco.com/howard-aiken-and-grace-hopper-4078389>)

U Velikoj Britaniji razvijen je Colossus, prvo potpuno elektroničko računalo, korišteno za dešifriranje neprijateljskih poruka. U SAD-u je 1946. godine predstavljen ENIAC, prvo elektroničko računalo opće namjene. Rat je time postao snažan katalizator razvoja tehnologija koje su ubrzo našle civilnu primjenu.

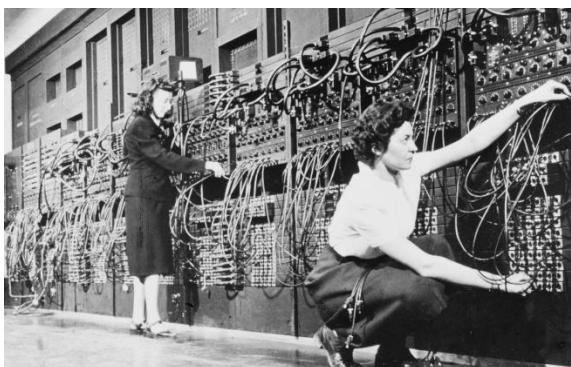
2.3 Generacije računala

Razvoj računala uobičajeno se prikazuje kroz podjelu na generacije, pri čemu svaka generacija označava razdoblje obilježeno dominantnom tehnologijom izrade elektroničkih sklopova. Promjena temeljne tehnologije dovela je do značajnih poboljšanja u brzini rada, pouzdanosti, veličini i potrošnji energije računala. U ovom poglavlju prikazan je razvoj računala od prvih elektroničkih sustava do suvremenih inteligentnih računalnih tehnologija.

2.3.1 Prva generacija računala (elektronske cijevi)

Računala prve generacije razvijena su tijekom 1940-ih i početkom 1950-ih godina, a njihova osnovna značajka bila je uporaba elektronskih cijevi kao temeljnih sklopnih elemenata. Elektronske cijevi omogućavale su izvođenje logičkih i aritmetičkih operacija znatno brže nego elektromehanički releji, no imale su brojne nedostatke.

Računala ove generacije bila su izuzetno velika, često su zauzimala cijele prostorije ili katove zgrada, trošila su ogromne količine električne energije te su proizvodila veliku količinu topline. Elektronske cijevi bile su sklone kvarovima, što je zahtijevalo stalno održavanje i česte prekide rada. Programiranje je bilo izrazito složeno jer su se programi unosili ručno, preklapanjem prekidača i spajanjem kabela, bez operacijskih sustava i viših programskih jezika.

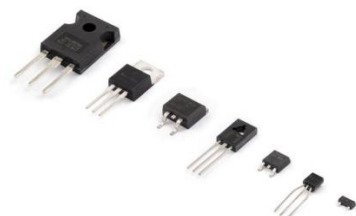


Slika 10 ENIAC (<https://www.aps.org/apsnews/2022/11/eniac-first-top-secret-program>)

Najpoznatiji predstavnik prve generacije računala je ENIAC, konstruiran 1946. godine u Sjedinjenim Američkim Državama. ENIAC je sadržavao više od 17.000 elektronskih cijevi i mogao je izvoditi tisuće operacija u sekundi, što ga je činilo višestruko bržim od prethodnih sustava. Iako nije bio programibilan u današnjem smislu, ENIAC je dokazao da je elektroničko računanje moguće i izuzetno učinkovito.

2.3.2 Druga generacija računala (tranzistori)

Druga generacija računala započela je izumom tranzistora 1947. godine. Tranzistor je postupno zamijenio elektronske cijevi i postao temeljna građevna jedinica novih računalnih sustava. Za razliku od elektronskih cijevi, tranzistori su bili znatno manji, pouzdaniji i energetski učinkovitiji.



Slika 11 Tranzistori (<https://www.automatika.rs/baza-znanja/tutorijali/tranzistori-pnp-i-npn-spoj-unipolarni-i-bipolarni-tranzistori.html>)

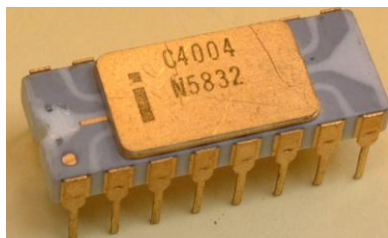
Primjena tranzistora omogućila je smanjenje dimenzija računala, smanjenje potrošnje električne energije te povećanje pouzdanosti sustava. Računala druge generacije manje su se zagrijavala, rjeđe su se kvarila i bila su stabilnija u radu. Ove prednosti dovele su do šire primjene računala u znanstvenim i poslovnim okruženjima.

Zahvaljujući tranzistorima, računalni sustavi postali su dostupniji, a njihova računalna snaga značajno je porasla u odnosu na prvu generaciju. Ovo razdoblje često se naziva početkom „silicijskog doba“ jer je silicij postao osnovni materijal računalne industrije.

2.3.3 Treća generacija računala (integrirani krugovi)

Treća generacija računala započela je razvojem integriranih krugova krajem 1950-ih godina. Integrirani krugovi omogućili su smještanje većeg broja tranzistora i drugih elektroničkih komponenti na jednu malu poluvodičku pločicu. Time su dodatno smanjene dimenzije sklopova, a pouzdanost i brzina rada znatno su povećane.

Računala treće generacije bila su manja, brža i jeftinija od svojih prethodnika, što je omogućilo njihovu širu primjenu izvan velikih institucija. U ovom razdoblju započinje i intenzivniji razvoj računalnog softvera te postupno uvođenje standardiziranih sustava za upravljanje računalnim resursima.



Slika 12 Mikroprocesor – Intel 4004 (<https://www.extremetech.com/computing/105029-intel-4004-the-first-cpu-is-40-years-old-today>)

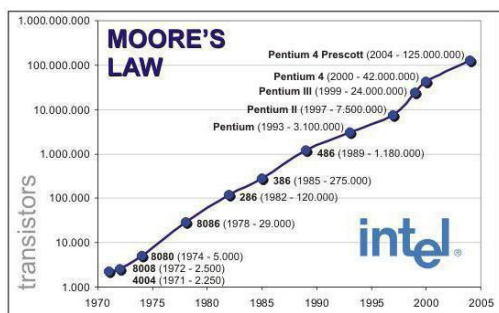
Posebno važan događaj bio je razvoj mikroprocesora koji je omogućio smještanje svih osnovnih funkcija središnje procesorske jedinice na jedan integrirani krug. Time su postavljeni temelji za razvoj osobnih računala.

2.3.4 Četvrta generacija računala (osobna računala)

Četvrta generacija računala obilježena je masovnom primjenom mikroprocesora i pojavom osobnih računala. Razvoj integriranih sklopova vrlo visoke gustoće omogućio je smještanje velikog broja tranzistora na jednu poluvodičku pločicu, čime su računala postala iznimno snažna, kompaktna i dostupna širokom krugu korisnika.

Važnu ulogu u ovom razdoblju imao je tzv. Mooreov zakon, koji je 1965. godine formulirao Gordon Moore, suosnivač tvrtke Intel. Moore je uočio da se broj tranzistora na integriranom krugu približno udvostručuje u pravilnim vremenskim razmacima, što dovodi do stalnog rasta računalne snage i smanjenja cijene po tranzistoru. Iako je zakon kasnije revidiran, on je desetljećima vrlo precizno opisivao stvarni razvoj računalne tehnologije i snažno utjecao na planiranje razvoja mikroprocesora.

„Mooreov zakon, Mooreov zakon eksponencijalnoga rasta opis, porasta broja tranzistora u čipovima tijekom vremena. Prvotno je zamijećeno udvostručenje toga broja svake godine, a potom tijekom jedne i pol ili dviju godina. Osnova je za procjenu ulaganja u elektroničku, informatičku i komunikacijsku tehnologiju. Nazvan po G. Mooreu, koji je 1965. postavio prvu procjenu.“ (Tehnički leksikon, 2007).



Slika 13 Mooreov zakon (<https://hr.puntomarinero.com/the-essence-of-mooreand-39/>)

Početak ere osobnih računala najčešće se veže uz pojavu računala Altair 8800 iz 1975. godine. Ovo računo bilo je jedno od prvih široko dostupnih mikroračunala, a prodavalo se u obliku kompleta za samostalno sastavljanje. Altair nije imao tipkovnicu ni zaslon, a unos podataka i programa odvijao se pomoću prekidača i svjetlosnih indikatora. Iako iznimno ograničenih mogućnosti, Altair 8800 potaknuo je razvoj softvera i interes šire javnosti za osobna računala.



Slika 14 Altair 8800 (<https://www.emustudio.net/documentation/user/altair8800/>)

Prava popularizacija osobnih računala započela je 1981. godine predstavljanjem računala IBM PC. Ovaj sustav koristio je mikroprocesor Intel 8088 i postavio je tehničke i arhitekturne standarde koji su se ubrzo proširili cijelom industrijom. Otvorena arhitektura IBM PC-a omogućila je drugim proizvođačima izradu kompatibilnih računala, što je dovelo do brzog širenja tržišta osobnih računala. Posebno važna bila je suradnja s tvrtkom Microsoft, koja je za IBM PC razvila operacijski sustav, čime je postavljen temelj za standardizaciju softverskih rješenja na osobnim računalima i daljnji razvoj tržišta računalnog softvera.



Slika 15 IBM PC (<https://www.itpro.com/hardware/361331/the-true-story-behind-the-ibm-personal-computer>)

Paralelno s razvojem IBM-kompatibilnih računala, značajan doprinos razvoju osobnih računala dala je i tvrtka Apple. Prva računala Apple I i Apple II bila su namijenjena krajnjim korisnicima i nudila su jednostavnije korištenje u odnosu na tadašnje sustave. Apple II

posebno se istaknuo kao jedno od prvih uspješnih kućnih računala. Veliku prekretnicu predstavljala je pojava grafičkog korisničkog sučelja i uporaba miša, što je prvi put šire primijenjeno u računalima Apple Lisa i kasnije Apple Macintosh. Time je rad na računalu postao intuitivniji i dostupniji korisnicima bez tehničkog predznanja.



Slika 16 Apple Lisa (<https://museum.syssrc.com/artifact/416/>)

U ovom razdoblju osobna računala ulaze u svakodnevnu uporabu u poslovanju, obrazovanju i kućanstvima. Računala više nisu ograničena na velike institucije, već postaju standardni alat za rad i komunikaciju. Razvoj grafičkih korisničkih sučelja dodatno je pojednostavio korištenje računala i omogućio njihovu širu prihvaćenost.

Tablica 1 Povijesni razvoj računalnih tehnologija i performansi

Tehnologija – sastavnica	Godina	Broj osnovnih operacija u sekundi
elektromehanika – releji	1940.	10
elektronika – elektronske cijevi	1945.	10^3
elektronika – tranzistori	1950.	10^4
mikroelektronika – sklopovi niskog stupnja integracije	1960.	10^5
mikroelektronika – sklopovi srednjeg stupnja integracije	1980.	10^6
mikroelektronika – sklopovi vrlo visokog stupnja integracije	2000.	10^9

Nakon 2000. godine daljnji porast računalnih performansi više se ne može jednostavno opisati isključivo povećanjem broja osnovnih operacija u sekundi za jednu procesorsku jezgru. Razvoj mikroelektronike ušao je u fazu vrlo visokog stupnja integracije, pri čemu se rast računalne snage sve više ostvaruje kombinacijom većeg broja procesorskih jezgri, paralelne obrade, vektorskih instrukcija i specijaliziranih akceleratora. Istodobno su fizička ograničenja, poput potrošnje energije i zagrijavanja, usporila rast radnog takta procesora.

Suvremeni računalni sustavi postižu visoke performanse primjenom heterogenih arhitektura koje objedinjuju središnje procesorske jedinice, grafičke procesore i

namjenske akceleratora, osobito za obradu podataka i umjetnu inteligenciju. Zbog toga se u novijoj literaturi performanse računala sve češće izražavaju različitim mjerama, poput operacija s pomičnim zarezom u sekundi ili specijaliziranih mjera za paralelne sustave, umjesto jedinstvene vrijednosti broja osnovnih operacija u sekundi. Ovakav pristup odražava složenost suvremenih računalnih arhitektura i naglašava prijelaz s jednostavnog povećanja brzine pojedinačne komponente na sustavno iskorištavanje paralelizma i specijalizacije.

2.3.5 Peta generacija računala (umjetna inteligencija)

Peta generacija računala započela je početkom 1980-ih godina, ponajprije u Japanu, s ambicioznim ciljem razvoja inteligentnih računalnih sustava koji bi imali sposobnost učenja, zaključivanja i donošenja odluka. Za razliku od prethodnih generacija, koje su bile prvenstveno obilježene tehnološkim napretkom u izradi elektroničkih sklopova, peta generacija stavila je u središte istraživanja softversku inteligenciju i kognitivne sposobnosti računala. Time je započeo sustavni razvoj umjetne inteligencije kao ključnog područja računalnih znanosti.

U ovom razdoblju pojavljuju se nova istraživačka područja, među kojima se posebno ističu umjetna inteligencija, ekspertni sustavi, robotika i obrada prirodnog jezika. Umjetna inteligencija nastoji razviti metode kojima bi računala mogla oponašati određene aspekte ljudske inteligencije, poput učenja, zaključivanja i prilagodbe novim situacijama. Ekspertni sustavi usmjereni su na rješavanje specijaliziranih problema u određenim područjima, pri čemu računalo preuzima ulogu stručnjaka. Razvoj robotike i sustava za obradu prirodnog jezika dodatno je proširio mogućnosti interakcije između čovjeka i računala.

Računala pete generacije temelje se na primjeni umjetne inteligencije u kombinaciji s paralelnom obradom podataka. Za razliku od ranijih sustava, koji su se oslanjali na sekvencijalno izvođenje instrukcija, suvremeni sustavi koriste velik broj procesorskih jezgri i paralelnih algoritama kako bi postigli visoke performanse. Razvoj mikroprocesorskih čipova doveo je do razine integracije u kojoj jedan čip može sadržavati više milijuna, a danas i milijarde elektroničkih komponenti, čime su stvoreni preduvjeti za izvođenje složenih AI algoritama u stvarnom vremenu.

Iako je veliki dio vizije pete generacije još uvijek predmet istraživanja, mnoge tehnologije već su našle praktičnu primjenu. Sustavi za prepoznavanje govora, strojno prevođenje, analiza prirodnog jezika i inteligentni sustavi za preporuke primjeri su primjene umjetne inteligencije u svakodnevnom životu. Ovi sustavi ne djeluju samo prema unaprijed zadanim pravilima, već koriste metode strojnog učenja koje im omogućuju prilagodbu i poboljšavanje performansi na temelju dostupnih podataka.

Jedan od glavnih ciljeva pete generacije računala jest razvoj sustava koji mogu komunicirati s korisnicima putem prirodnog jezika te samostalno učiti i organizirati informacije. Takvi sustavi trebali bi razumjeti značenje izgovorenih ili pisanih riječi, a ne samo njihov oblik, te donositi odluke na temelju konteksta. Dodatno, integracija različitih

senzora omogućuje računalima interakciju s okolinom, čime se približavaju ljudskoj sposobnosti opažanja i prilagodbe promjenjivim uvjetima.

U kontekstu budućeg razvoja, kvantno računanje i nanotehnologija smatraju se tehnologijama koje bi mogle značajno utjecati na daljnji napredak pete generacije. Kvantna računala imaju potencijal za rješavanje izuzetno složenih problema u vrlo kratkom vremenu, dok nanotehnologija omogućuje izradu sve manjih i energetski učinkovitijih elektroničkih komponenti. Iako su ove tehnologije još u ranoj fazi razvoja, one predstavljaju važne smjerove istraživanja u suvremenom računalstvu.

Peta generacija računala ne označava jasno zatvoreno povijesno razdoblje, već kontinuirani proces u kojem se granice mogućnosti računala neprestano pomiču. Razvoj naprednih algoritama, umjetne inteligencije i novih hardverskih tehnologija postupno vodi prema računalnim sustavima koji će imati sve veću autonomiju u radu i donošenju odluka, čime se bitno mijenja uloga računala u suvremenom društvu.

Pitanja za ponavljanje

1. Što je abak i kakvu je ulogu imao u razvoju računanja?
2. Koji je doprinos Johna Napiera razvoju računalnih metoda?
3. Tko je konstruirao prvi značajan mehanički stroj za računanje i kako se zvao taj stroj?
4. Koja je bila uloga Ade Lovelace u razvoju računalstva?
5. Koja je tehnologija omogućila prijelaz s mehaničkih na elektroničke sustave?
6. Koju su ulogu u razvoju računalne tehnologije imali događaji tijekom Drugog svjetskog rata?
7. Što je ENIAC i zašto je važan u povijesti računalstva?
8. Na temelju čega se računala dijele na generacije?
9. Koja je tehnologija obilježila treću generaciju računala?
10. Što je mikroprocesor i zašto je važan za razvoj osobnih računala?
11. Koje je računalo često povezano s početkom ere osobnih računala?
12. Koju je ulogu imao IBM PC u razvoju računalne industrije?
13. Koji je doprinos tvrtke Apple popularizaciji osobnih računala?
14. Što opisuje Mooreov zakon?
15. Koje su glavne značajke pete generacije računala?

3 Utjecaj Turingova stroja na razvoj računalstva

Razvoj računalstva nije započeo izgradnjom fizičkih računala, već razmatranjem temeljnih pitanja o tome što znači računati i koje se sve probleme može riješiti postupkom koji se sastoji od konačnog broja jasno definiranih koraka. Ovo poglavlje obrađuje motivaciju za formalizaciju pojma algoritma te prikazuje Turingov stroj kao teorijski model koji je snažno utjecao na razvoj računalnih znanosti i modernih računala.

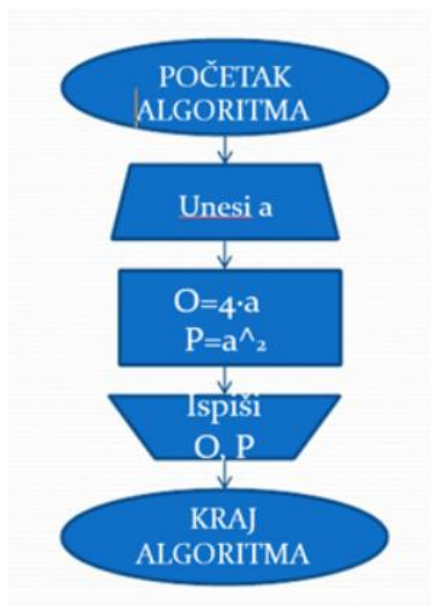
3.1 Motivacija

Težnja za realizacijom računala javila se iz potrebe za obradom podataka. U mnogim praktičnim situacijama obrada velikih količina podataka čovjeku iziskuje previše vremena ili je sklona pogreškama. Obrada podataka predstavlja svrsishodnu aktivnost čiji je cilj dobivanje tražene informacije iz raspoloživih podataka pomoću jasno definiranog postupka.

Svaki proces obrade podataka može se opisati pomoću triju osnovnih sastavnica:

- podataka koji se obrađuju,
- algoritma koji opisuje način njihove transformacije te
- izvršitelja koji taj algoritam provodi.

Izvršitelj može biti čovjek, ali i stroj, pod uvjetom da je algoritam dovoljno precizno definiran da ga je moguće mehanički slijediti.



Slika 17 Primjer algoritma za računanje opsega i površine kvadrata

Algoritam predstavlja postupak kojim se matematički ili računalni problem razlaže na niz jednoznačno određenih koraka. Da bi se neki postupak smatrao algoritmom, mora zadovoljiti nekoliko temeljnih svojstava:

- određenost – svaki korak mora biti jasno i nedvosmisleno definiran,

- konačnost – algoritam se mora završiti nakon konačnog broja koraka te
- širina primjene – područje nad kojim se algoritam izvodi (mora biti primjenjiv na cijeli razred ulaznih podataka, a ne samo na jedan konkretan slučaj).

Upravo su ta svojstva omogućila formalno razmatranje algoritama neovisno o tome tko ih ili što izvršava.

U nastavku su prikazana dva klasična primjera algoritamskog razmišljanja koja ilustriraju kako se problemi mogu svesti na jednostavne i ponovljive korake.

Euklidov algoritam

Euklidov algoritam jedan je od najstarijih poznatih algoritama, a koristi se za pronalaženje najveće zajedničke mjere, odnosno najvećeg zajedničkog djelitelja dvaju prirodnih brojeva a i b . Ideja algoritma temelji se na ponavljanom oduzimanju manjeg broja od većeg dok se ne dobije isti rezultat.

Koraci algoritma:

1. korak: odaberi dva prirodna broja a i b .

Prijeđi na sljedeći korak.

2. korak: usporedi brojeve.

Prijeđi na sljedeći korak.

3. korak: Ako su brojevi jednaki – to je rezultat.

Ako nisu jednaki, prijeđi na sljedeći korak.

4. korak: Ako je prvi broj manji od drugog, zamijeni im mjesta.

Prijeđi na sljedeći korak.

5. korak: Oduzmi drugi broj od prvog i promatraj drugi broj i ostatak.

Prijeđi na korak broj 2.

Postupak započinje odabirom dvaju prirodnih brojeva, nakon čega se uspoređuju njihove vrijednosti. Ako su brojevi jednaki, algoritam se zaustavlja jer je pronađen traženi rezultat. Ako nisu, manji broj se oduzima od većeg, a dobivena razlika ponovno se uspoređuje s manjim brojem. Ovaj se postupak ponavlja sve dok oba broja ne postanu jednaka.

Primjer rada algoritma: $a = 12$, $b = 16$; pronađite najveću zajedničku mjeru (nzm).

1. korak (promotri brojeve a i b)

2. korak (usporedi brojeve)

3. korak $a \neq b$

4. korak $a = 16$; $b = 12$

5. korak $16 - 12 = 4$, usporediti 12 i 4
3. korak $a \neq b$
4. korak $a = 12; b = 4$
5. korak $12 - 4 = 8$, usporediti 4 i 8
3. korak $a \neq b$
4. korak $a = 8; b = 4$
5. korak: $8 - 4 = 4$, usporediti 4 i 4
3. korak $a = b = 4$ ZAUSTAVI ALGORITAM!

REZULTAT: $nzm = 4$

Na primjeru brojeva 12 i 16 algoritam postupno reducira problem sve dok se ne dobije vrijednost 4 koja predstavlja najveću zajedničku mjeru. Ovaj primjer jasno pokazuje kako se složeniji problem može riješiti ponavljanjem jednostavnih, jasno definiranih koraka, što je temelj algoritamskog razmišljanja.

Problem naprtnjače

Problem naprtnjače jedan je od klasičnih primjera u računarstvu i teoriji algoritama koji ilustrira kako se stvarni, naizgled intuitivni problemi mogu formalizirati i riješiti algoritamskim postupkom. U ovom problemu zadana je naprtnjača određene nosivosti, odnosno kapaciteta C te skup od N predmeta koje je moguće ponijeti. Svaki predmet opisan je svojom težinom w_k i vrijednošću v_k , pri čemu je cilj odabrati takav udio svakog predmeta F_k da ukupna težina ne premaši kapacitet naprtnjače, dok se istodobno maksimizira ukupna vrijednost predmeta u naprtnjači.

U razmatranoj inačici problema dopušteno je dijeljenje predmeta, pri čemu se vrijednost dijela predmeta mijenja proporcionalno njegovoj težini. Time se problem svodi na pronalaženje optimalne kombinacije cjelovitih predmeta i njihovih dijelova, što omogućuje rješenje pomoću jednostavnog i učinkovitog algoritamskog postupka.

Algoritam započinje razvrstavanjem svih predmeta prema omjeru njihove vrijednosti i težine. Predmeti s većim omjerom imaju veću vrijednost po jedinici težine i stoga imaju prednost pri odabiru. Nakon toga, naprtnjača se postupno puni tako da se uvijek odabire predmet s najvećim omjerom vrijednosti i težine i u cijelosti se smješta u naprtnjaču sve dok to dopušta preostali kapacitet. Kada sljedeći predmet ne može stati u cijelosti, uzima se samo onaj njegov dio koji točno popunjava preostali prostor u naprtnjači.

Primjer: Parametri i varijable problema: $C, N, w_1, v_1, F_1, \dots, w_N, v_N, F_N$

Primjer ulazne supstitucije:

$$C = 14 \text{ kg}, N = 3,$$

- $w_1 = 4 \text{ kg}, v_1 = 30 \$$
- $w_2 = 6 \text{ kg}, v_2 = 48 \$$
- $w_3 = 7 \text{ kg}, v_3 = 50 \$$

Rješenje: Algoritam

1. korak: Razvrstaj objekte na temelju omjera vrijednosti i težine

predmet 2 = $48/6 = 8 \$/\text{kg}$;

predmet 1 = $30/4 = 7,5 \$/\text{kg}$;

predmet 3 = $50/7 = 7,14 \$/\text{kg}$

2. korak: Ponavljaj sve dok se naprtnjača ne pretrpa

Iz skupa predmeta uzmi predmet s najvećim omjerom vrijednost/težina i smjesti ga cijelog u naprtnjaču.

3. korak: Izvadi iz naprtnjače posljednji predmet i razdijeli ga tako da upravo njegovi dijelovi popune naprtnjaču.

Izlazna supstitucija:

U naprtnjaču stavljamo:

Cijeli predmet 1 i cijeli predmet 2 i $4/7$ predmeta 3

Ukupna vrijednost u naprtnjači: 106,57 \$

Napomena: Opisana inačica problema naprtnjače predstavlja tzv. *frakcijsku verziju* problema, u kojoj je dopušteno dijeljenje predmeta na dijelove čija je vrijednost proporcionalna njihovoj težini. U općem obliku problema, gdje dijeljenje predmeta nije dopušteno, rješenje zahtijeva složenije algoritamske pristupe.

Problem naprtnjače pokazuje kako se optimizacijski zadaci mogu riješiti algoritamskim pristupom koji se sastoji od jasno definiranih i ponovljivih koraka. Ovakvi primjeri prirodno vode prema temeljnim pitanjima računarstva: kako formalno opisati algoritam, kako osigurati da se postupak uvijek pravilno izvrši te kako konstruirati univerzalni stroj koji može provoditi takve postupke neovisno o vrsti problema. Upravo je razmatranje takvih pitanja dovelo Alana Turinga do uvođenja koncepta Turingova stroja koji predstavlja temeljni model izračunljivosti u računalnim znanostima.

3.2 Turingov stroj

Britanski matematičar i logičar **Alan Mathison Turing** objavio je 1936. godine rad *On Computable Numbers, with an Application to the Entscheidungsproblem*, u kojem je predstavio apstraktni model računanja poznat kao Turingov stroj. Cilj tog rada bio je formalno opisati pojam izračunljivosti, odnosno odgovoriti na pitanje koji se problemi mogu riješiti algoritamskim postupkom.

Osnovna ideja Turingova stroja jest da se svaki postupak računanja može raščlaniti na vrlo jednostavne, gotovo nedjeljive elementarne operacije. Time je računanje svedeno na mehaničko izvođenje unaprijed definiranih koraka, neovisno o značenju podataka koji se obrađuju. Upravo ta apstrakcija omogućila je precizno matematičko razmatranje algoritama i njihovih svojstava.

Turingov stroj predstavlja temeljni koncept u teoriji računarstva. Iako se ne radi o stvarnom fizičkom uređaju, on služi kao univerzalni model računala pomoću kojeg se analiziraju algoritmi, izračunljivost problema i granice automatiziranog računanja. Problem se smatra izračunljivim ako postoji Turingov stroj koji, za svaki dopušteni ulaz, u konačnom broju koraka daje ispravan rezultat. Ovaj koncept ima izravan utjecaj na razvoj modernih računala, programskih jezika i teorije algoritama.

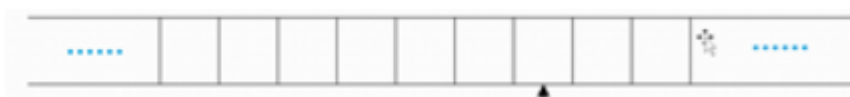
Formalni opis Turingova stroja

Formalno, Turingov stroj opisuje se n-torkom:

$$TS = (Q, S, T, P, b, q_0, q_f, \delta)$$

- Q = konačan skup unutarnjih stanja stroja
- S = konačan skup vanjske abecede stroja
- T = konačan skup vanjske abecede / {prazni simbol, b }
- b = prazan simbol (engl. *blank*)
- P = skup naredbi za pomak glave za čitanje i pisanje
- q_0 = početno stanje stroja
- q_f = konačno stanje stroja
- δ = logička funkcija stroja $S \times Q \rightarrow S \times P \times Q$

TS ima vanjsku memoriju – beskonačna vrpca, neograničena slijeva i zdesna



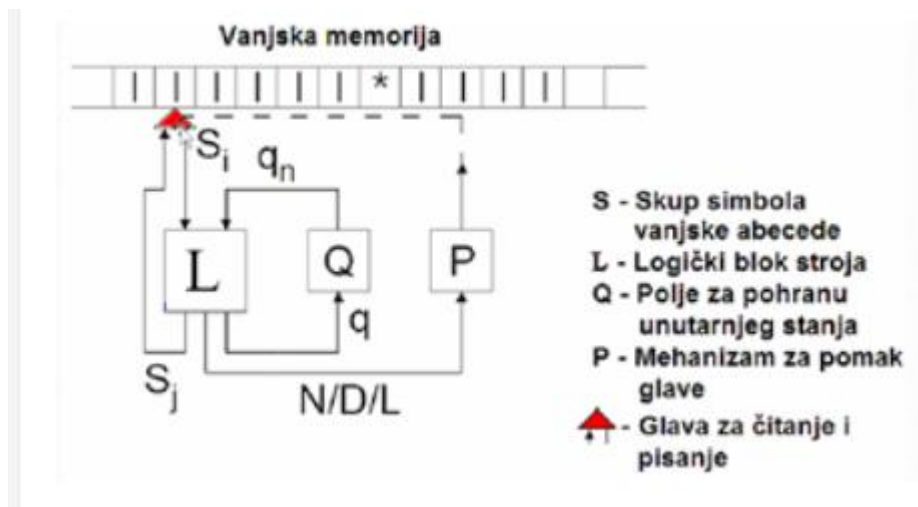
Vrpca je podijeljena na polja:

- Svako polje sadrži jedan simbol iz S (može i b , prazno mjesto)
- Upis simbola u polje na vrpci ili čitanje simbola obavlja se pomičnom glavom za čitanje i pisanje (R/W)

Turingov stroj ima i dvije unutarnje memorije:

- q = trenutno unutarnje stanje
- p = trenutna naredba za pomak R/W glave

- Sastoji se od R (udesno), L (ulijevo), ili N (nema pomaka)
- Pomak za samo jedno polje



Slika 18 Model Turingova stroja

Elementi vanjske abecede

Unutarnja stanja stroja					
	q_1	q_2	q_2	q_4	...
Δ	$\Delta D q_4$	$\Delta D q_3$	$\Delta D q_1$	$\Delta D q_3$	
I	$\alpha N q_2$	$\beta N q_2$	$I D q_1$	$I L q_1$	
α	$\alpha L q_1$	$\alpha D q_2$	$I L q_3$	$\Delta D q_4$	
β	$\beta L q_1$	$\beta D q_2$	$\Delta L q_3$	$I L q_4$	
....					

trojka

Slika 19 Primjer funkcionalne sheme Turingova stroja

Opis rada Turingova stroja

Rad Turingova stroja odvija se u diskretnim vremenskim trenucima koji se nazivaju taktovima. Na početku rada definira se početno stanje q_0 , sadržaj vrpce koji predstavlja ulaznu informaciju te početni položaj R/W glave. U svakom taktu stroj očitava simbol ispod glave i , na temelju trenutačnog stanja, primjenjuje prijelaznu funkciju δ .

Prijelazna funkcija preslikava par očitano simbola i trenutačnog stanja u trojku koja određuje novi simbol koji će se zapisati na vrpcu, pomak glave te novo unutarnje stanje stroja. Nakon izvršenja ove operacije započinje sljedeći takt, u kojem stroj nastavlja rad iz novog stanja i s novim položajem glave. Tijekom rada stroj postupno transformira početnu informaciju u niz međukoraka, sve dok ne dosegne konačno stanje.

U trenutku kada stroj dosegne konačno stanje q_f , rad stroja se zaustavlja, a sadržaj vrpce predstavlja konačni rezultat računanja. Postoje dva moguća krajnja ishoda rada Turingova

stroja: prihvati, kada se stroj zaustavi u prihvatljivom stanju, i odbijanje, kada se stroj zaustavi u neprihvatljivom stanju ili zapadne u beskonačnu petlju.

Primjer rada Turingova stroja – inkrement dekadskog broja

Kao ilustraciju rada Turingova stroja može se promatrati zadatak povećanja dekadskog broja zapisanog na vrpici za jedan. Pretpostavlja se da se R/W glava u početnom trenutku nalazi lijevo od zapisanog broja. Kako bi se izvršio inkrement, stroj se najprije pomiče udesno sve dok ne naiđe na prazno polje, čime pronalazi kraj zapisa broja. Zatim se glava pomiče ulijevo kako bi se pozicionirala na najmanje značajnu znamenku.

Ako je očitana znamenka manja od 9, ona se jednostavno zamjenjuje sljedećom znamenkom i stroj se zaustavlja. Ako je očitana znamenka 9, ona se zamjenjuje s 0, a postupak se nastavlja ulijevo, pri čemu se prenosi vrijednost jedinice. Ovaj se proces ponavlja sve dok se ne naiđe na znamenku manju od 9 ili dok se ne dosegne početak broja.

Iako se ovaj postupak na prvi pogled čini složenim, on se u potpunosti sastoji od niza elementarnih prijelaza definiranih prijelaznom funkcijom Turingova stroja. Time se jasno pokazuje kako se i složenijsi računski zadaci mogu svesti na niz vrlo jednostavnih operacija.

Koraci algoritma:

1. $(b, q_0) \rightarrow (b, D, q_0)$ – R/W se pomiče udesno i ostaje u q_0
2. $(b, q_0) \rightarrow (b, D, q_0)$ – R/W se pomiče udesno i ostaje u q_0
3. $(1, q_0) \rightarrow (1, D, q_1)$ – stroj traži najmanje značajnu znamenku
4. $(2, q_1) \rightarrow (2, D, q_1)$
5. $(6, q_1) \rightarrow (6, D, q_1)$
6. $(9, q_1) \rightarrow (9, D, q_1)$
7. $(b, q_1) \rightarrow (b, L, q_2)$ – naišli na „b”, vraćamo se ulijevo
8. $(9, q_2) \rightarrow (0, L, q_2)$ – inkrementira se najmanje značajna znamenka
9. $(6, q_2) \rightarrow (7, N, q_f)$ – prenosimo desetice, zaustavljamo TS

	q_0	q_1	q_2
0	$0 D q_1$	$0 D q_1$	$1 N q_1$
1	$1 D q_1$	$1 D q_1$	$2 N q_1$
2	$2 D q_1$	$2 D q_1$	$3 N q_1$
3	$3 D q_1$	$3 D q_1$	$4 N q_1$
4	$4 D q_1$	$4 D q_1$	$5 N q_1$
5	$5 D q_1$	$5 D q_1$	$6 N q_1$
6	$6 D q_1$	$6 D q_1$	$7 N q_1$
7	$7 D q_1$	$7 D q_1$	$8 N q_1$
8	$8 D q_1$	$8 D q_1$	$9 N q_1$
9	$9 D q_1$	$9 D q_1$	$0 L q_2$
b	$b D q_0$	$b L q_2$	$1 N q_1$

Slika 20 Funkcionalna shema

Turingov stroj predstavlja jedan od najvažnijih koncepata u teoriji računarstva jer omogućuje formalno razumijevanje pojmova algoritma, izračunljivosti i granica računalne obrade podataka. Njegovo razumijevanje važno je za studente i stručnjake računarstva jer čini teorijsku osnovu na kojoj se temelje suvremeni računalni sustavi i algoritmi.

Pitanja za ponavljanje

1. Koja su tri osnovna elementa svakog procesa obrade podataka?
2. Što je algoritam?
3. Koja temeljna svojstva mora imati algoritam i objasnite svako od njih?
5. Što predstavlja problem naprtnjače u teoriji algoritama?
6. Što je Turingov stroj i koja mu je osnovna svrha u teoriji računarstva?
8. Što znači da je problem izračunljiv?
10. Koji su osnovni dijelovi Turingova stroja?
11. Što predstavlja vrpca u modelu Turingova stroja?
12. Koja je uloga glave za čitanje i pisanje (R/W glave)?
13. Kada se rad Turingova stroja zaustavlja te koja su dva moguća ishoda rada Turingova stroja?

4 Osnovni dijelovi računala

Računalo je elektronički uređaj namijenjen automatskoj obradi podataka i izvršavanju zadataka prema unaprijed definiranim uputama. Njegova osnovna svrha jest pretvaranje ulaznih podataka u korisne informacije primjenom algoritama koji su zapisani u obliku programa. Zahvaljujući velikoj brzini obrade, pouzdanosti i mogućnosti pohrane velike količine podataka, računala su postala temeljni alat suvremenog društva.

Osnovne funkcionalne cjeline svakog računala čine središnja procesorska jedinica, memorija, sustavi za trajnu pohranu podataka, ulazne jedinice te izlazne jedinice. Ove komponente zajedno omogućuju interakciju korisnika s računalom i izvršavanje širokog raspona zadataka, od jednostavnih aritmetičkih operacija do složenih znanstvenih, inženjerskih i poslovnih analiza.

Primjena računala donijela je brojne prednosti, osobito u pogledu povećanja učinkovitosti i produktivnosti rada. Računala mogu izvoditi velik broj operacija u vrlo kratkom vremenu uz visoku razinu točnosti, što je u mnogim područjima nemoguće postići ručnim radom. Omogućena je i pohrana te organizacija velikih količina podataka, kao i njihovo brzo pretraživanje i obrada. Nadalje, razvoj računalnih mreža i interneta omogućio je globalnu komunikaciju i pristup informacijama neovisno o fizičkoj lokaciji korisnika.

S druge strane, široka primjena računala donosi i određene izazove. Povećana ovisnost o tehnologiji može uzrokovati probleme u slučaju kvarova ili gubitka podataka, dok sigurnosni rizici, poput zlonamjernog softvera i neovlaštenog pristupa, zahtijevaju stalnu primjenu zaštitnih mjera. Također, automatizacija procesa može imati društvene i ekonomske posljedice, kao što su promjene na tržištu rada i potreba za stalnim prilagođavanjem novim znanjima i vještinama.

Unatoč tim izazovima, računala su danas nezamjenjiva u gotovo svim područjima ljudske djelatnosti. Razumijevanje njihove građe i načina rada stoga predstavlja temeljno znanje za svakog studenta informacijskih i tehničkih znanosti.

Računalo je složeni sustav koji se sastoji od sklopovlja (engl. *hardware*) i programske podrške (engl. *software*). Sklopovlje obuhvaća sve fizičke komponente računala, elektroničke, električne i mehaničke dijelove, dok programska podrška čini skup programa koji omogućavaju upravljanje sklopovljem i izvršavanje korisničkih zadataka. Iako se računala međusobno razlikuju po veličini, snazi i namjeni, njihova osnovna organizacija u velikoj se mjeri temelji na istim načelima.

Temeljnu strukturu modernih računala opisao je John von Neumann 1945. godine, definirajući arhitekturu koja se i danas koristi u većini računalnih sustava. Prema tom modelu, računalo se sastoji od memorije, središnje procesorske jedinice te ulazno-izlaznog podsustava. Posebnost von Neumannove arhitekture jest u tome što su i podaci i programske instrukcije pohranjeni u istoj memoriji, u binarnom obliku.

Von Neumannov model računala temelji se na četiri ključne komponente:

1. Memorija – Služi za pohranu podataka i instrukcija u binarnom obliku (0 i 1).
2. Upravljačka jedinica – Dekodira i upravlja izvršenjem instrukcija te koordinira rad ostalih dijelova računala.
3. Aritmetičko-logička jedinica (ALU) – Izvršava aritmetičke i logičke operacije na podacima koji se pohranjuju u memoriji, prema uputama upravljačke jedinice.
4. Ulazno-izlazni podsustav – Upravlja unosom podataka i izlazom rezultata obrade. Ulazni uređaji (npr. tipkovnica, miš) omogućuju unos podataka u računalo, dok izlazni uređaji (npr. monitor, pisač) prikazuju rezultate.

Načelo rada von Neumannova modela temelji se na sekvencijalnom izvršavanju instrukcija, pri čemu se instrukcije dohvaćaju iz memorije, dekodiraju i izvršavaju jedna po jedna. Ovakav pristup omogućio je razvoj univerzalnih računala koja se mogu programirati za različite zadatke bez promjene sklopovlja, što je predstavljalo važan korak u razvoju suvremenog računalstva.

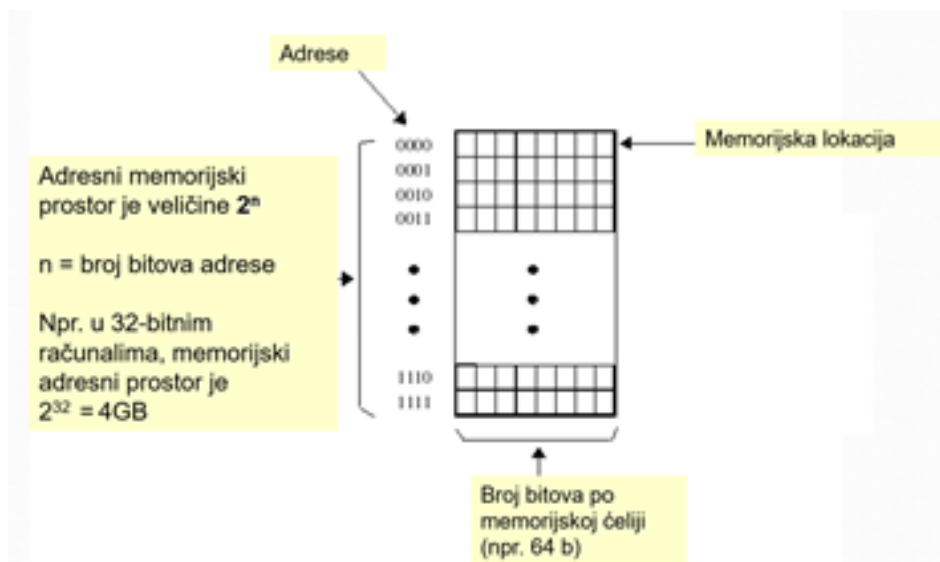
Razvoj računalnog sklopovlja i programske podrške doveo je do iznimno snažnih i učinkovitih sustava koji danas mogu izvoditi milijarde operacija u sekundi. Operacijski sustavi preuzeli su ulogu posrednika između korisnika i sklopovlja, upravljajući resursima računala i omogućujući jednostavno korištenje različitih aplikacija.

U sljedećim potpoglavljima detaljno će se razmotriti osnovni dijelovi računala, s posebnim naglaskom na memoriju, ulazno-izlazne jedinice, središnju procesorsku jedinicu i aritmetičko-logičku jedinicu.

4.1 Memorija

Memorija je jedan od temeljnih dijelova svakog računalnog sustava jer omogućuje pohranu podataka i programskih instrukcija potrebnih za izvršavanje zadataka. Bez memorije računalo ne bi bilo u stanju zadržati informacije potrebne za obradu, niti bi moglo izvršavati programe. Svi podaci u memoriji pohranjuju se u binarnom obliku, kao nizovi bitova koji mogu poprimiti vrijednost 0 ili 1.

Memorija se sastoji od velikog broja memorijskih lokacija, pri čemu svaka lokacija ima jedinstvenu adresu putem koje joj se pristupa. Veličina adresnog prostora memorije određena je brojem bitova koji se koriste za adresiranje, a ukupni broj mogućih memorijskih lokacija izražava se formulom 2^n , gdje je n broj bitova adrese. Ovakav način organizacije omogućuje izravan i brz pristup podacima, neovisno o njihovom položaju u memoriji.



Slika 21 Memorija računala

Uloga memorije posebno je važna u procesu izvršavanja programa. Tijekom rada računala, procesor neprestano dohvaća instrukcije i podatke iz memorije, obrađuje ih te po potrebi vraća rezultate natrag u memoriju. Brzina pristupa memoriji izravno utječe na ukupne performanse sustava, što je dovelo do razvoja različitih vrsta memorije s različitim karakteristikama.

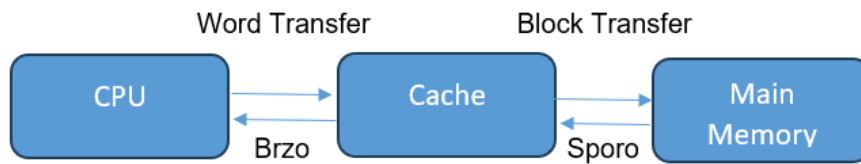
Vrste memorije

Jedna od osnovnih vrsta memorije je ROM (eng. *Read-Only Memory*), memorija namijenjena isključivo čitanju podataka. Podaci pohranjeni u ROM-u trajni su i ne gube se isključivanjem računala. Ova memorija sadrži osnovne programe potrebne za pokretanje sustava, kao što su programi za inicijalizaciju sklopovlja i učitavanje operacijskog sustava. Iako naziv sugerira potpunu nemogućnost promjene sadržaja, suvremeni oblici ROM-a, poput EPROM-a (eng. *Erasable Programmable Read-Only Memory*) i EEPROM-a (eng. *Electrically Erasable Programmable Read-Only Memory*), omogućuju brisanje i ponovno programiranje podataka posebnim postupcima.

RAM (eng. *Random Access Memory*) predstavlja glavnu radnu memoriju računala. Ona omogućuje čitanje i pisanje podataka na proizvoljnim adresama, pri čemu je vrijeme pristupa približno jednako za sve memorijske lokacije. RAM je volatilna memorija, što znači da se njezin sadržaj briše nakon isključivanja napajanja. U RAM-u se tijekom rada računala nalaze operacijski sustav, aktivni programi i podaci koji se trenutačno obrađuju. Zbog svoje brzine i fleksibilnosti, RAM ima ključnu ulogu u svakodnevnom radu računalnih sustava.

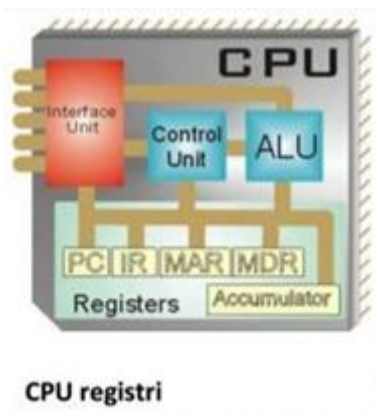
Cache memorija, često nazvana i priručna memorija, mala je memorija vrlo velike brzine smještena neposredno uz procesor ili unutar njega. Njezina je svrha smanjiti vrijeme pristupa podacima koji se često koriste. Budući da je pristup RAM-u znatno sporiji od pristupa procesoru, cache memorija služi kao posrednik između procesora i glavne

memorije. U nju se pohranjuju podaci i instrukcije za koje postoji velika vjerojatnost da će uskoro ponovno biti potrebni, čime se značajno ubrzava rad sustava.



Slika 22 Odnos cache i glavne memorije

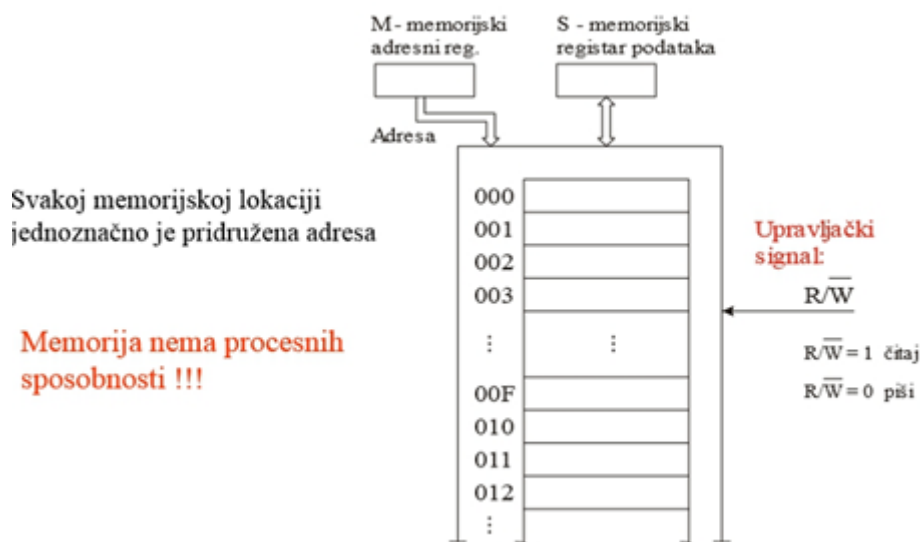
Registri procesora su posebna vrsta memorije smještena unutar središnje procesorske jedinice. Njihov kapacitet je vrlo ograničen, ali je brzina pristupa najveća u cijelom memorijskom sustavu. Registri služe za pohranu operanda, međurezultata i upravljačkih informacija koje procesor trenutačno koristi prilikom izvršavanja instrukcija. Zbog svoje uloge, registri imaju presudan utjecaj na brzinu izvršavanja programa i učinkovitost rada procesora.



Slika 23 CPU registri

Operacije memorijske jedinice

Bez obzira na vrstu, memorijska jedinica izvodi dvije osnovne operacije: pisanje (WRITE) i čitanje (READ). Operacija pisanja podrazumijeva pohranu podataka u određenu memorijsku lokaciju, dok operacija čitanja omogućuje dohvaćanje prethodno pohranjenih podataka. Ove se operacije odvijaju sinkronizirano s radom procesora i predstavljaju osnovu svih računalnih postupaka.

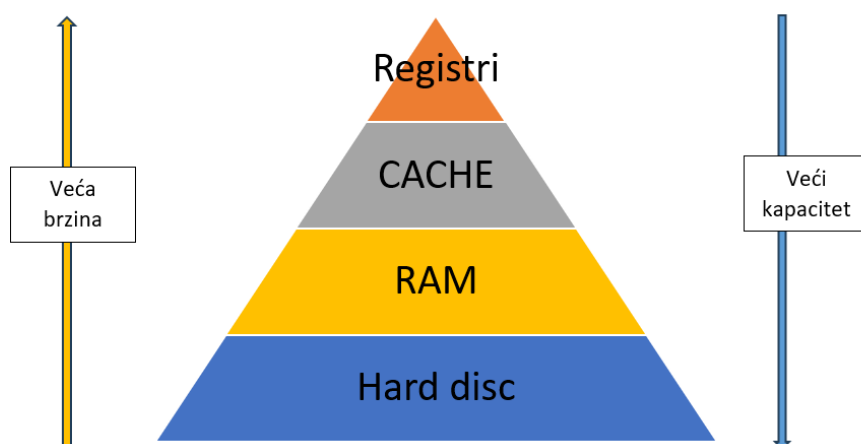


Slika 24 Memorija

Memorijska hijerarhija

Zbog razlika u brzini, kapacitetu i cijeni memorijskih tehnologija, memorijski sustavi organizirani su u hijerarhiju. Postoje četiri glavne razine:

1. Registri procesora: vrlo brza memorija malog kapaciteta unutar procesora.
2. *Cache* memorija: podijeljena u razine (L1, L2, L3), služi za pohranu najčešće korištenih podataka.
3. Glavna memorija (RAM): primarna radna memorija koja pohranjuje aktivne programe i podatke.
4. Sekundarna memorija: uređaji poput tvrdih diskova i SSD-ova, koriste se za dugotrajnu pohranu podataka.



Slika 25 Grafički prikaz memorijske hijerarhije

Memorija predstavlja temeljni element svakog računalnog sustava jer omogućuje pohranu i obradu informacija. Različite vrste memorije i njihova hijerarhijska organizacija omogućuju postizanje optimalne ravnoteže između brzine, kapaciteta i troškova, čime se osiguravaju visoke performanse i pouzdan rad modernih računala.

4.2 Središnja procesorska jedinica (CPU)

Središnja procesorska jedinica (engl. *Central Processing Unit*, CPU) predstavlja temeljnu komponentu svakog računala i često se naziva „mozgom” računalnog sustava. CPU je zadužen za izvršavanje programskih instrukcija, obradu podataka te upravljanje protokom informacija između memorije i ostalih dijelova računala. Sve operacije koje računalo izvodi, neovisno o njihovoj složenosti, u konačnici se svode na aktivnosti koje provodi procesor.

CPU se sastoji od nekoliko funkcionalnih cjelina, od kojih su najvažnije upravljačka jedinica, aritmetičko-logička jedinica te skup registara. Zajedničkim djelovanjem ovih dijelova omogućeno je dohvaćanje instrukcija iz memorije, njihovo dekodiranje i izvršavanje u skladu s definiranim algoritmom.

Upravljačka jedinica (engl. *Control Unit*, CU) ima središnju ulogu u koordinaciji rada procesora. Ona dohvaća instrukcije iz memorije, dekodira njihov operacijski kod te generira upravljačke signale koji određuju koje će se operacije izvršiti i kojim redoslijedom. Upravljačka jedinica ne obrađuje podatke izravno, već nadzire i usmjerava rad ostalih dijelova procesora: aritmetičko-logičku jedinicu, registre i memorijski podsustav.

Aritmetičko-logička jedinica (ALU) izvršava osnovne računske i logičke operacije nad podacima. Aritmetičke operacije su zbrajanje, oduzimanje, množenje i dijeljenje, logičke operacije poput AND, OR, NOT i XOR. ALU omogućuje manipulaciju podacima i donošenje logičkih odluka koje su nužne za grananje i ponavljanje u programima.

Registri procesora

Registri su najbrži oblik memorije u računalnom sustavu i nalaze se izravno unutar procesora. Njihov kapacitet je vrlo malen, ali je brzina pristupa znatno veća nego kod *cache* memorije i RAM-a. Registri služe za privremeno pohranjivanje podataka, instrukcija i međurezultata koje procesor trenutačno koristi, čime se izbjegava sporiji pristup glavnoj memoriji.

Uloga registara posebno je važna tijekom izvršavanja instrukcija jer se većina operacija odvija isključivo nad podacima koji se nalaze u registrima. Zbog toga broj, vrsta i organizacija registara imaju velik utjecaj na performanse procesora.

Oznaka registra	Naziv registra	Veličina	Funkcija registra
AC	Akumulator	16 bita	Pohranjuje podatke za operacije aritmetičko-logičke jedinice (ALU)
PC	Programsko brojilo	12 bita	Pokazuje na sljedeću instrukciju koja će se izvršiti
IR	Instrukcijski registar	16 bita	Pohranjuje trenutačnu instrukciju
DR	Registar podataka	16 bita	Pohranjuje podatke

AR	Adresni registar	12 bita	Pohranjuje adresu memorijske lokacije
TR	Privremeni registar	16 bita	Pohranjuje privremene podatke
INTR / OUTR	Ulazni i izlazni registri	po 8 bita	Pohranjuju ulazne i izlazne podatke

Među najvažnijim registrima nalaze se akumulator (AC), koji služi za pohranu operandi i rezultata aritmetičkih operacija te MQ registar, koji se koristi za proširenje akumulatora pri radu s većim brojevima. Memorijski adresni registar (MAR) sadrži adresu memorijske lokacije kojoj se pristupa, dok memorijski registar podataka (MDR) privremeno pohranjuje podatke koji se čitaju iz memorije ili u nju zapisuju. Instrukcijski registar (IR) sadrži trenutačnu instrukciju koja se izvršava, dok kontrolni i statusni registri pohranjuju informacije o stanju procesora i rezultatima operacija.

Zajedničko djelovanje registara omogućuje procesoru da instrukcije izvršava brzo i učinkovito, uz minimalna kašnjenja uzrokovana pristupom sporijim razinama memorijske hijerarhije.

Ciklus izvršavanja instrukcija

Izvršavanje programa u procesoru odvija se kroz ponavljajući strojni ciklus koji se sastoji od nekoliko faza. U fazi dohvaćanja (engl. *fetch*) upravljačka jedinica dohvaća instrukciju iz memorije na adresi koju pokazuje programsko brojilo (PC) i pohranjuje je u instrukcijski registar. Nakon toga se vrijednost programskog brojila povećava kako bi pokazivala na sljedeću instrukciju.

U fazi dekodiranja instrukcija se analizira kako bi se utvrdilo koju operaciju treba izvršiti i nad kojim podacima. Na temelju dekodirane instrukcije, upravljačka jedinica generira odgovarajuće upravljačke signale. U fazi izvršavanja (engl. *execute*) izvršava se zadana operacija, što može uključivati aritmetičku obradu u ALU-u, prijenos podataka između registara ili pristup memoriji.

Važno je naglasiti da se sadržaj dohvaćen iz memorije u fazi dohvaćanja interpretira kao instrukcija, dok se u fazi izvršavanja sadržaj memorije tumači kao podatak. Ova razlika ključna je za razumijevanje načina rada procesora i von Neumannove arhitekture.

Procesor neprestano ponavlja ovaj ciklus sve dok se ne izvrši posljednja instrukcija programa ili dok ne dođe do prekida rada.

Središnja procesorska jedinica predstavlja srž svakog računalnog sustava. Njena sposobnost brzog dohvaćanja, obrade i izvršavanja instrukcija omogućuje rad svih ostalih komponenti računala. Brzina takta, organizacija registara, učinkovitost ALU-a i način upravljanja instrukcijama izravno utječu na ukupne performanse sustava. Razumijevanje unutarnje strukture i načina rada CPU-a stoga je ključno za razumijevanje funkcioniranja modernih računala.

4.2.1 Aritmetičko-logička jedinica (ALU)

Aritmetičko-logička jedinica (ALU) predstavlja temeljnu funkcionalnu cjelinu unutar središnje procesorske jedinice koja provodi sve osnovne računske i logičke operacije nad podacima. Sve matematičke obrade i logičke usporedbe koje procesor izvodi oslanjaju se na rad ALU-a, što ovu jedinicu čini glavnim dijelom svakog procesora.

ALU izvršava aritmetičke operacije poput zbrajanja, oduzimanja, množenja i dijeljenja, kao i logičke operacije nad bitovima podataka AND, OR, NOT i XOR. Osim toga, ALU sudjeluje u usporedbama vrijednosti, primjerice pri provjeri jednakosti ili odnosa veće-manje, što je nužno za uvjetno grananje i ponavljanje u programima.

Operandi nad kojima ALU izvršava operacije najčešće se nalaze u registrima procesora, osobito u akumulatoru. Nakon izvršavanja operacije, rezultat se pohranjuje u registar ili se prosljeđuje drugoj jedinici na daljnju obradu. Rad ALU-a u potpunosti je kontroliran upravljačkom jedinicom koja određuje koju će operaciju ALU izvršiti i nad kojim podacima.

Strukturno, ALU se može promatrati kao skup aritmetičkih i logičkih sklopova koji rade paralelno. U modernim procesorima ALU je dodatno optimizirana te često postoji više aritmetičko-logičkih jedinica koje omogućuju paralelno izvršavanje operacija, osobito u višejezgrenim i superskalarnim arhitekturama.

Brzina i učinkovitost ALU-a imaju izravan utjecaj na performanse procesora jer se gotovo sve operacije nad podacima oslanjaju na ovu jedinicu. Napredak u tehnologiji omogućio je razvoj iznimno brzih i složenih ALU-ova koji su sposobni obavljati velik broj operacija u vrlo kratkom vremenu.

ALU je nezaobilazan element svakog računalnog sustava. Bez nje računalo ne bi moglo izvršavati ni najosnovnije izračune niti donositi logičke odluke, što jasno potvrđuje njezinu središnju ulogu u radu procesora i cijelog računala.

4.2.2 RISC i CISC arhitektura procesora

Način na koji procesor izvršava programe uvelike ovisi o arhitekturi skupa instrukcija, odnosno o tome kako su instrukcije oblikovane, koliko su složene i kako komuniciraju s memorijom. Tijekom povijesnog razvoja računalnih sustava razvila su se dva temeljna koncepta arhitekture procesora: CISC (eng. *Complex Instruction Set Computer*) i RISC (eng. *Reduced Instruction Set Computer*). Ova podjela predstavlja različite dizajnerske filozofije koje su imale značajan utjecaj na razvoj modernih procesora.

CISC arhitektura temelji se na ideji da procesor posjeduje velik broj složenih instrukcija, pri čemu jedna instrukcija može obuhvatiti više elementarnih operacija. Takve instrukcije mogu istovremeno dohvaćati podatke iz memorije, izvoditi aritmetičke ili logičke operacije te pohranjivati rezultate natrag u memoriju. Instrukcije su često promjenjive duljine i mogu zahtijevati više taktova za izvršavanje. Kako bi se upravljalo složenošću takvih instrukcija, CISC procesori često koriste mikrokod koji razlaže složene instrukcije na jednostavnije unutarnje korake.

Povijesno gledano, CISC arhitektura razvijena je u razdoblju kada su memorija i prevoditelji bili ograničeni, pa je cilj bio smanjiti broj instrukcija u programu i time uštedjeti memorijski prostor. Zbog toga su CISC procesori dugo dominirali osobnim računalima, a najpoznatiji primjer CISC arhitekture je x86 obitelj procesora. Međutim, složenost instrukcija i dekodiranja otežava učinkovito cjevovodno izvršavanje te povećava potrošnju energije.

Suprotan pristup predstavlja RISC arhitektura koja se temelji na korištenju manjeg broja jednostavnih instrukcija. U RISC arhitekturi instrukcije su obično jednake duljine, jednostavne za dekodiranje i najčešće se izvršavaju u jednom taktu. Posebno je naglašeno razdvajanje instrukcija za rad s memorijom i instrukcija za izvođenje aritmetičkih i logičkih operacija. Podaci se u RISC procesorima obrađuju gotovo isključivo unutar registara, dok se memoriji pristupa samo pomoću posebnih *load* i *store* instrukcija.

Jednostavnost instrukcija omogućuje iznimno učinkovito cjevovodno izvršavanje, što rezultira velikom protočnošću i boljom energetsom učinkovitošću. Zbog toga je RISC arhitektura posebno pogodna za mobilne uređaje, ugrađene sustave i sustave s više jezgri. Tipični primjeri RISC arhitekture su ARM i RISC-V procesori, dok se MIPS arhitektura često koristi u obrazovne svrhe zbog svoje jednostavnosti i preglednosti.

Tablica 2 Usporedba CISC i RISC arhitekture

Značajka	CISC	RISC
Broj instrukcija	velik	mali
Duljina instrukcije	promjenjiva	fiksna
Izvršavanje instrukcija	više taktova	često 1 takt
Rad s memorijom	direktno u instrukcijama	<i>load/store</i>
Hardver	složen	jednostavniji
Energetska učinkovitost	niža	viša

Važno je naglasiti da u suvremenim procesorima razlika između CISC i RISC arhitekture više nije strogo izražena. Moderni x86 procesori, iako se izvane ponašaju kao CISC arhitektura, interno razlažu složene instrukcije u jednostavnije mikrooperacije koje se izvršavaju prema RISC načelima. S druge strane, RISC procesori koriste napredne tehnike poput spekulativnog izvršavanja, višestrukog izdavanja instrukcija i dubokih cjevovoda, čime se povećavaju performanse, ali i složenost unutarnje arhitekture.

Zbog toga se danas CISC i RISC ne promatraju kao strogo odvojene kategorije, već kao različiti povijesni i konceptualni pristupi dizajnu procesora. Stvarne performanse procesora ne ovise isključivo o tome pripada li određenoj arhitekturi, već o cijelom sklopu

čimbenika, uključujući mikroarhitekturu, organizaciju memorije, učinkovitost prevoditelja i karakteristike softvera.

4.3 Ulazno-izlazne jedinice (I/O)

Računalo komunicira s ulazno-izlaznim (I/O) uređajima putem sučelja, drajvera i komunikacijskih protokola, čime se osigurava interoperabilnost i ispravno funkcioniranje različitih uređaja. Ova komunikacija omogućuje razmjenu podataka između računala, korisnika i vanjskog okruženja.

Sučelje predstavlja način povezivanja računala i periferne opreme te može biti fizičko ili logičko.

Fizičko sučelje je stvarni priključak ili komunikacijski kanal kojim se uređaj povezuje s računalom. Najčešći primjeri su USB priključci, HDMI i DisplayPort za video izlaz, Ethernet za mrežnu komunikaciju te bežične tehnologije poput Bluetootha i Wi-Fi veze. Razvoj sučelja doveo je do univerzalnih priključaka (npr. USB-C) koji mogu istovremeno prenositi podatke, video signal i električnu energiju.

Logičko sučelje predstavlja apstraktni sloj koji omogućava operacijskom sustavu prepoznavanje, adresiranje i upravljanje uređajem. Ono definira kako se uređaj vidi unutar sustava (npr. kao ulazni, izlazni ili kombinirani uređaj) i kako se razmjenjuju podaci između hardvera i softvera.

Drajver je specijalizirani softverski program koji omogućava operacijskom sustavu komunikaciju s određenim hardverskim uređajem. Svaki drajver prilagođen je konkretnom uređaju i operacijskom sustavu te služi kao posrednik između hardvera i aplikacija.

Drajveri prevode naredbe operacijskog sustava u oblik razumljiv uređaju te vraćaju rezultate u standardiziranom obliku. Moderni operacijski sustavi koriste *plug-and-play* mehanizme koji omogućuju automatsko prepoznavanje i instalaciju drajvera pri spajanju uređaja. U naprednijim sustavima, dio upravljanja uređajima preuzet je već u fazi pokretanja računala (UEFI firmware¹), čime se omogućuje rad osnovnih I/O uređaja i prije učitavanja operacijskog sustava.

Komunikacijski protokoli definiraju skup pravila prema kojima se podaci prenose između računala i I/O uređaja.

Protokoli niže razine se bave fizičkim prijenosom podataka i sinkronizacijom komunikacije. Primjeri ovih protokola su USB protokol za periferne uređaje ili Ethernet protokol za žične mreže.

¹ UEFI (eng. *Unified Extensible Firmware Interface*) je nasljednik klasičnog BIOS-a. To je program pohranjen u memoriji na matičnoj ploči koji se pokrene odmah nakon uključivanja računala. Njegove glavne uloge su inicijalizacija hardvera (CPU, RAM, diskovi, grafika, tipkovnica), omogućavanje komunikacije s osnovnim I/O uređajima te pokretanje operacijskog sustava.

Protokoli više razine omogućuju smisleniju razmjenu podataka i usluga. Primjerice, u mrežnoj komunikaciji TCP/IP omogućuje pouzdan prijenos podataka, dok se HTTP koristi za komunikaciju između klijenta i poslužitelja. U kontekstu I/O sustava, sve je češća primjena mrežnih i virtualnih protokola kod udaljenih i virtualiziranih uređaja (npr. mrežni pisači, cloud I/O).

4.3.1 Ulazni uređaji

Ulazni uređaji služe za unos podataka i naredbi u računalo. Svi ulazni podaci moraju se pretvoriti u digitalni oblik, odnosno u niz nula i jedinica, kako bi ih računalo moglo obraditi.



Slika 26 Pretvorba signala

Najčešće korišteni ulazni uređaji su:

- tipkovnica
- miš
- skener
- mikrofon

Dodatni ulazni uređaji su još i upravljačka palica (*joystick*), osjetilna ploha (*touchpad*), web kamera, A/D pretvornici, grafičke ploče, čitači crtičnih kodova i pametne ploče. U novije vrijeme sve važniju ulogu imaju uređaji temeljeni na virtualnoj (VR) i proširenoj stvarnosti (AR) te sustavi za prepoznavanje glasa i slike koji često koriste ugrađene algoritme umjetne inteligencije.

4.3.2 Izlazni uređaji

Izlazni uređaji pretvaraju digitalne podatke iz računala u oblik razumljiv korisniku. Oni omogućuju prikaz rezultata obrade u vizualnom, zvučnom ili fizičkom obliku.

Najčešći izlazni uređaji su:

- monitori
- pisači (tintni, laserski, matrični, termički i višenamjenski)
- ploteri
- zvučnici
- projektori

Razvojem tehnologije, izlazni uređaji postaju sve napredniji, s većim rezolucijama, boljom energetsom učinkovitošću i mogućnošću mrežnog povezivanja.

Integracija ulazno-izlaznih uređaja u računalne sustave nužna je za funkcionalnost računala i kvalitetnu interakciju korisnika s računalom. Ovaj proces uključuje odgovarajuću hardversku podršku, instalaciju drajvera i primjenu komunikacijskih protokola. Samo pravilno konfigurirani i podržani I/O uređaji mogu pouzdano funkcionirati u računalnom okruženju.

Sigurnost ulazno-izlaznih (I/O) uređaja

Sigurnost ulazno-izlaznih uređaja predstavlja važan aspekt sigurnosti računalnih sustava. Nepravilna uporaba ili neovlašteni pristup I/O uređajima može dovesti do ozbiljnih sigurnosnih problema kao i problema s privatnošću.

Ranjivosti u tim uređajima mogu omogućiti neovlašten pristup osjetljivim podacima, primjerice zloupotrebom kamere ili mikrofona, kao i gubitak ili oštećenje podataka uslijed kompromitacije pisača ili pohranjenih dokumenata. Također, I/O uređaji mogu poslužiti kao ulazna točka za unošenje zlonamjernog softvera u računalni sustav, osobito kada se koriste nepouzdana ili neprovjereni USB uređaji.

Posebnu pažnju potrebno je posvetiti zaštiti uređaja koji prikupljaju osobne i osjetljive podatke, kao što su kamere, mikrofoni i različiti senzori, kako bi se osigurala privatnost korisnika te usklađenost s važećim propisima o zaštiti podataka. Smanjenje sigurnosnih rizika postiže se redovitim ažuriranjem operacijskog sustava, drajvera i *firmware* uređaja, korištenjem pouzdanih I/O uređaja provjerenih proizvođača te ograničavanjem pristupa uređajima primjenom sigurnosnih postavki i lozinki. Dodatno, praćenje aktivnosti uređaja omogućuje pravovremeno uočavanje neuobičajenog ponašanja, dok pravilno podešavanje postavki privatnosti za kamere, mikrofone i senzore doprinosi većoj razini zaštite.

Primjena navedenih mjera doprinosi sigurnijem radu računalnog sustava i značajno smanjuje rizike povezane s uporabom ulazno-izlaznih uređaja.

4.4 Sabirnice

Sabirnica je osnovni sklop u računalu koji omogućuje komunikaciju između različitih komponenti računalnog sustava. Predstavlja zajednički električni spojni put kojim se prenose podaci, adrese i upravljački signali između procesora, memorije, ulazno-izlaznih uređaja i ostalih podsustava. Korištenjem sabirnice smanjuje se broj vodiča i priključaka potrebnih za povezivanje pojedinih jedinica, budući da svaka komponenta priključena na sabirnicu koristi zajednički komunikacijski kanal. Na taj se način pojednostavljuje struktura sustava i omogućuje lakše proširenje računala dodavanjem novih uređaja.

Iako je sabirnica fizički zajednička veza, u određenom trenutku komunikacija se odvija samo između dviju komponenti, dok su ostale logički odvojene od prijenosa. Time se sprječavaju sukobi u prijenosu podataka i osigurava ispravno funkcioniranje sustava.

Prema položaju u računalnom sustavu razlikuju se vanjske i unutarnje sabirnice. Vanjska sabirnica povezuje glavne komponente računala, kao što su procesor, memorija i ulazno-izlazni uređaji, dok unutarnja sabirnica unutar procesora povezuje aritmetičko-logičku jedinicu, upravljačku jedinicu i registre. Ova unutarnja komunikacija odvija se vrlo velikim brzinama i ima značajan utjecaj na ukupne performanse procesora.

S obzirom na funkciju koju obavljaju, sabirnice se dijele na upravljačke, podatkovne i adresne. Upravljačka sabirnica prenosi signale kojima se upravlja radom sustava, kao što su naredbe za čitanje i pisanje, zahtjevi i potvrde prekida te signal vremenskog vođenja. Sabirnica podataka služi za dvosmjerni prijenos podataka između komponenti, a njezina širina najčešće iznosi 32 ili 64 bita, što izravno utječe na količinu podataka koja se može prenijeti u jednom taktu. Adresna sabirnica koristi se za prijenos adresa memorijskih lokacija ili registara, čime se određuje kojoj se jedinici pristupa tijekom komunikacije.

U računalnim sustavima postoje različite arhitekture sabirnica. Unutarprocesorske sabirnice omogućuju komunikaciju između više jezgri procesora, dok systemske sabirnice povezuju procesor, glavnu memoriju i ulazno-izlazne uređaje. U visokoparalelnim i distribuiranim sustavima koriste se posebne sabirnice i komunikacijske mreže koje omogućuju razmjenu podataka između većeg broja računala ili procesorskih jedinica.

Brzina sabirnice ima značajan utjecaj na performanse računala jer određuje koliko se brzo podaci mogu prenositi između komponenti. Da bi komunikacija bila usklađena, sve komponente moraju se pridržavati definiranih sabirničkih protokola. U praksi se koriste različiti protokoli ovisno o namjeni, kao što su PCI Express² za povezivanje internih uređaja, USB za vanjske periferije te SATA za pohranu podataka.

Prema načinu prijenosa podataka, sabirnice mogu biti paralelne ili serijske. Paralelne sabirnice omogućuju istodoban prijenos više bitova putem zasebnih vodova, zbog čega postižu visoke brzine na kratkim udaljenostima. Međutim, takav način prijenosa zahtijeva složeniju infrastrukturu, veću potrošnju energije i osjetljiviji je na električne smetnje. Zbog tih ograničenja paralelne sabirnice danas se sve rjeđe koriste.

Serijske sabirnice prenose podatke bit po bit jednim komunikacijskim kanalom. Takav prijenos pouzdaniji je na većim udaljenostima, manje je osjetljiv na smetnje te zahtijeva manje prostora i energije. Zbog navedenih prednosti, suvremeni računalni sustavi uglavnom se oslanjaju na serijske sabirnice, kao što su USB, SATA i Ethernet.

Razvoj sabirničkih tehnologija kontinuirano napreduje s ciljem povećanja propusnosti, smanjenja latencije i poboljšanja energetske učinkovitosti. Time se omogućuje brža i pouzdanija komunikacija između komponenti, što izravno doprinosi ukupnim performansama suvremenih računalnih sustava.

Prekidi i izravni pristup memoriji

² PCI Express (PCIe) je suvremena serijska sabirnica namijenjena povezivanju komponenti koje zahtijevaju veliku propusnost i nisku latenciju. Koristi serijski prijenos podataka, ali postiže vrlo visoke brzine zahvaljujući paralelnom radu više komunikacijskih linija koje se nazivaju staze ili *laneovi*.

U radu sabirnica važno mjesto zauzimaju mehanizmi prekida i izravnog pristupa memoriji jer omogućuju učinkovitiju komunikaciju između procesora i ulazno-izlaznih uređaja. Bez tih mehanizama procesor bi morao stalno nadzirati stanje uređaja, što bi značajno smanjilo ukupne performanse sustava.

Prekid (engl. *interrupt*) je signal kojim ulazno-izlazni uređaj ili neki drugi dio sustava obavještava procesor da zahtijeva pažnju. Kada procesor primi prekid, privremeno zaustavlja izvođenje trenutnog programa, sprema njegovo stanje te izvršava posebnu rutinu za obradu prekida. Nakon obrade, procesor nastavlja rad tamo gdje je stao. Na taj se način izbjegava nepotrebno stalno provjeravanje stanja uređaja i omogućuje učinkovitije korištenje procesorskog vremena.

Prekidi mogu nastati kao posljedica različitih događaja, poput završetka ulazno-izlazne operacije, dolaska podataka s periferije ili pojave pogreške u radu uređaja. Sustav prekida omogućuje brzu reakciju računala na vanjske događaje te ima važnu ulogu u višezadaćnim operacijskim sustavima.

Za prijenos većih količina podataka koristi se mehanizam izravnog pristupa memoriji (eng. *DMA – Direct Memory Access*). Kod DMA prijenosa podaci se prenose izravno između ulazno-izlaznog uređaja i glavne memorije, bez stalnog sudjelovanja procesora. Procesor inicijalno postavlja parametre prijenosa, a nakon završetka prijenosa biva obaviješten prekidom. Time se značajno smanjuje opterećenje procesora i povećava ukupna propusnost sustava.

Kombinacija prekida i DMA mehanizama omogućuje da sabirnice rade učinkovitije, uz manju latenciju i bolju raspodjelu resursa. Ovi mehanizmi posebno su važni u sustavima s velikim brojem ulazno-izlaznih operacija, kao što su sustavi za pohranu podataka, mrežni uređaji i multimedijske aplikacije.

4.5 Operacijski sustav

Operacijski sustav je skup programa koji povezuje i objedinjuje sve sklopovske dijelove računala te omogućuje njihovu učinkovitu uporabu. On predstavlja posrednika između korisnika, aplikacijskog softvera i hardverskih resursa računala, čime omogućuje korisnicima rad s računalom bez potrebe za poznavanjem računalnog jezika ili detalja o unutarnjem funkcioniranju hardvera.

U prvim računalnim sustavima operacijski su sustavi bili ograničeni na izvođenje jedne zadaće u određenom trenutku. Razvojem računalne tehnologije uvedena je mogućnost istodobnog izvođenja više zadaća, poznata kao višezadaćnost (engl. *multitasking*) koja omogućuje da više programa dijeli procesorsko vrijeme i ostale resurse na učinkovit način. Time se značajno povećava iskoristivost sustava i poboljšava korisničko iskustvo.

Operacijski sustav služi kao temelj za korištenje aplikacijskog softvera te osigurava kontrolu nad procesorom, memorijom, sustavom datoteka i ulazno-izlaznim uređajima. On upravlja dodjelom resursa pojedinim programima, nadzire njihovo izvođenje i

osigurava stabilan i siguran rad sustava. Bez operacijskog sustava većina suvremenog softvera ne bi mogla funkcionirati jer se oslanja na njegove usluge za pristup hardveru.

Operacijski sustavi danas su prisutni u velikom broju uređaja, a najčešće se koriste na osobnim računalima, mobilnim uređajima i igraćim konzolama. Moderni operacijski sustavi korisnicima nude grafičko korisničko sučelje (GUI) koje omogućuje jednostavno upravljanje računalom, pokretanje programa i obavljanje različitih zadataka. Raspon tih zadataka kreće se od osnovnih aktivnosti, poput pregledavanja internetskih sadržaja, do složenih poslova kao što su obrada multimedije, znanstveni izračuni i razvoj softvera.

Razvojem operacijskih sustava naglasak je sve više stavljen na učinkovitost, sigurnost i prilagodljivost različitim vrstama hardvera, čime oni postaju nezaobilazan dio svakog suvremenog računalnog sustava.

Primjer rada operacijskog sustava

Kada se računalno uključi, najprije se pokreće sistemski *firmware* (BIOS ili suvremeniji UEFI) koji inicijalizira osnovne sklopovske dijelove računala. Tijekom tog postupka provjerava se ispravnost procesora, radne memorije, osnovnih ulazno-izlaznih uređaja i dostupnost uređaja za pohranu podataka. Nakon uspješne inicijalizacije *firmware* pronalazi operacijski sustav na disku i započinje njegovo učitavanje u memoriju.

Tijekom pokretanja operacijskog sustava provode se dodatne provjere i inicijalizacije uređaja, učitavaju se potrebni upravljački programi te se priprema grafičko korisničko sučelje. Korisnik taj proces najčešće prepoznaje po porukama o učitavanju sustava, primjerice prikazu zaslona s informacijom o pokretanju operacijskog sustava. Nakon dovršetka učitavanja, korisniku je omogućena prijava u sustav i započinje radno okruženje.

Nakon prijave korisnik može pokretati aplikacijske programe, poput internetskog preglednika, programa za obradu teksta ili drugih aplikacija. Operacijski sustav pritom upravlja prikazom programa na zaslonu, dodjeljuje im procesorsko vrijeme i osigurava pristup potrebnim resursima. Tijekom rada s aplikacijama operacijski sustav se brine za ispravan prijenos podataka unesenih putem tipkovnice i miša, njihov prikaz na zaslonu te pohranu podataka na uređajima za pohranu.

Osim toga, operacijski sustav upravlja komunikacijom s izlaznim uređajima, poput pisača. Prilikom ispisa dokumenta sustav poziva odgovarajuće servisne procedure i nadzire tijek ispisa, pri čemu korisnika obavještava o eventualnim problemima, primjerice ako pisač nije uključen, ako nema papira ili ako nedostaje toner. Na taj način operacijski sustav osigurava koordiniran i pouzdan rad svih dijelova računalnog sustava tijekom svakodnevne uporabe.

BIOS i UEFI

BIOS (eng. *Basic Input Output*) i UEFI (eng. *Unified Extensible Firmware Interface*) predstavljaju sistemski softver koji se izvršava odmah nakon uključivanja računala i ima zadatac

inicijalizirati hardver te pokrenuti operacijski sustav. BIOS je stariji i jednostavniji sustav koji se koristi još od prvih osobnih računala. Njegova funkcionalnost ograničena je osnovnim provjerama hardvera i pokretanjem operacijskog sustava, a sučelje je tekstualno i namijenjeno radu pomoću tipkovnice. BIOS ima tehnička ograničenja, poput nemogućnosti rada s diskovima većima od 2 TB i sporijeg postupka pokretanja sustava.

UEFI je suvremena zamjena za BIOS koja donosi niz poboljšanja. UEFI omogućuje brže pokretanje računala, podršku za velike diskove te grafičko korisničko sučelje s podrškom za miš. Za razliku od BIOS-a, UEFI ima modularnu strukturu i može koristiti vlastite upravljačke programe, čime se omogućuje naprednije upravljanje hardverom već u ranoj fazi pokretanja sustava.

Jedna od važnih razlika između BIOS-a i UEFI-ja odnosi se na sigurnost. UEFI podržava mehanizme poput sigurnog pokretanja sustava (*Secure Boot*) kojim se sprječava učitavanje neovlaštenog ili zlonamjernog softvera tijekom pokretanja računala. BIOS takve mehanizme izvorno ne podržava, što ga čini manje otpornim na suvremene sigurnosne prijetnje.

Iako oba sustava imaju istu osnovnu zadaću, UEFI je prilagođen zahtjevima modernih računalnih sustava i danas se koristi u gotovo svim novim računalima. BIOS se uglavnom može pronaći na starijoj opremi ili u kompatibilnom načinu rada unutar UEFI sustava.

Različiti operacijski sustavi

Među najpoznatijim i najraširenijim operacijskim sustavima ističe se Microsoft Windows koji se najčešće koristi na osobnim računalima. Windows je namijenjen širokom krugu korisnika i poznat je po grafičkom korisničkom sučelju, velikoj kompatibilnosti s hardverom i bogatom ekosustavu aplikacija. Zbog jednostavnosti uporabe i snažne podrške proizvođača softvera i hardvera, Windows se široko primjenjuje u kućnom, obrazovnom i poslovnom okruženju.

Linux predstavlja skupinu operacijskih sustava otvorenog koda koji se temelje na Linux jezgri. Odlikuje se visokom stabilnošću, sigurnošću i prilagodljivošću, zbog čega se često koristi na poslužiteljima, u ugrađenim sustavima i znanstvenim okruženjima. Linux postoji u mnogim distribucijama koje se razlikuju po namjeni i korisničkom sučelju, a posebno je cijenjen među stručnjacima i naprednim korisnicima.

Operacijski sustav MacOS razvija tvrtka Apple i namijenjen je isključivo njihovim računalima. Poznat je po visokoj razini integracije hardvera i softvera, stabilnosti te naglasku na sigurnosti i korisničkom iskustvu. MacOS se često koristi u kreativnim djelatnostima, poput grafičkog dizajna, obrade videa i glazbene produkcije.

U području mobilnih uređaja dominantnu ulogu ima Android, operacijski sustav temeljen na Linux jezgri. Android se koristi na velikom broju pametnih telefona i tableta različitih proizvođača, a odlikuje se otvorenošću, velikom prilagodljivošću i širokom ponudom aplikacija. Zbog svoje fleksibilnosti, Android je prisutan i u drugim vrstama uređaja, kao što su pametni televizori i automobilski sustavi.

Drugi značajan mobilni operacijski sustav je iOS koji je razvijen za Appleove pametne telefone i tablete. iOS je poznat po jednostavnom i intuitivnom korisničkom sučelju, visokoj razini sigurnosti te strogo kontroliranom sustavu distribucije aplikacija. Takav pristup osigurava stabilan i pouzdan rad uređaja, ali uz manju mogućnost prilagodbe u odnosu na Android.

Raznolikost operacijskih sustava omogućuje prilagodbu različitim vrstama uređaja i potrebama korisnika, od osobnih računala i poslužitelja do mobilnih i ugrađenih sustava. Odabir operacijskog sustava ovisi o namjeni računala, zahtjevima za performansama, sigurnošću i razinom prilagodbe.

Pitanja za ponavljanje

1. Što je računalno i koja je njegova osnovna svrha?
2. Koje su osnovne funkcionalne cjeline svakog računala?
4. Koje su četiri ključne komponente von Neumannova modela računala i zašto je on značajan?
6. Zašto je memorija nužna za rad računala?
7. Koja je osnovna razlika između ROM i RAM memorije?
8. Što je *cache* memorija i koja je njezina uloga?
9. Što su registri procesora i zašto su važni?
10. Koje su dvije osnovne operacije memorijske jedinice?
11. Što je memorijska hijerarhija i koje su njezine glavne razine?
12. Što je središnja procesorska jedinica (CPU) i koja je njena uloga u računalu?
13. Koje su osnovne faze ciklusa izvršavanja instrukcija?
14. Što je ALU i koje vrste operacija izvršava?
15. Koja je osnovna razlika između CISC i RISC arhitekture?
17. Koja je uloga drajvera u radu ulazno-izlaznih uređaja?
18. Što su sabirnice i koja je njihova uloga u računalu?
19. Koje vrste sabirnica imamo i koje su razlike među njima?
20. Što je operacijski sustav i koja je njegova osnovna uloga?
21. Koja je uloga i osnovna razlika BIOS-a i UEFI-ja?

5 Booleova algebra i digitalna logika

Rad računala temelji se na obradi podataka i izvođenju logičkih operacija. Podaci i njihova obrada mogu se svrstati u tri osnovne skupine. Prvu skupinu čine brojevi, nad kojima se provode računske operacije. Drugu skupinu predstavljaju znakovi, kod kojih se naglasak stavlja na organizaciju zapisa, sortiranje i pretraživanje. Treću skupinu čine upravljački znakovi koji služe za kodiranje, dekodiranje i upravljanje izvršavanjem programa.

U digitalnim sustavima svi se podaci na kraju svode na binarni oblik, odnosno na vrijednosti 0 i 1. Na primjer, pri zbrajanju binarnih vrijednosti moguće je izravno zbrojiti 0 i 1, ali se isto može opisati i logičkim sudom. Ako je vrijednost A jednaka 1, a vrijednost B jednaka 0, tada je rezultat operacije $A + B$ jednak 1. Takav način razmišljanja vodi prema formalnom opisu logičkih operacija.

5.1 Logika sudova

Kako bi se mogli izgraditi digitalni sustavi sposobni za izvođenje logičkih i računskih operacija, potrebno je formalizirati pravila prema kojima se te operacije odvijaju. Mogućnost digitalnih sklopova da obrađuju podatke temelji se na njihovoj sposobnosti izvođenja jednostavnih logičkih operacija. Matematičku analizu logike prvi je sustavno razradio engleski matematičar George Boole sredinom 19. stoljeća. Njegov rad postavio je temelje Booleove algebre koja se danas smatra jednim od temeljnih matematičkih alata informatike i digitalne elektronike.

Osnovni element Booleove algebre jest logički sud ili izjava. Logički sud mora biti jednoznačno određen te može imati samo jednu od dvije moguće vrijednosti: istinitu ili neistinitu.

Zagreb je glavni grad Republike Hrvatske.	ISTINA
Zemlja je ravna ploča.	NEISTINA

Primjer sudova: **SUD A:** „Nema ulja u motoru.”

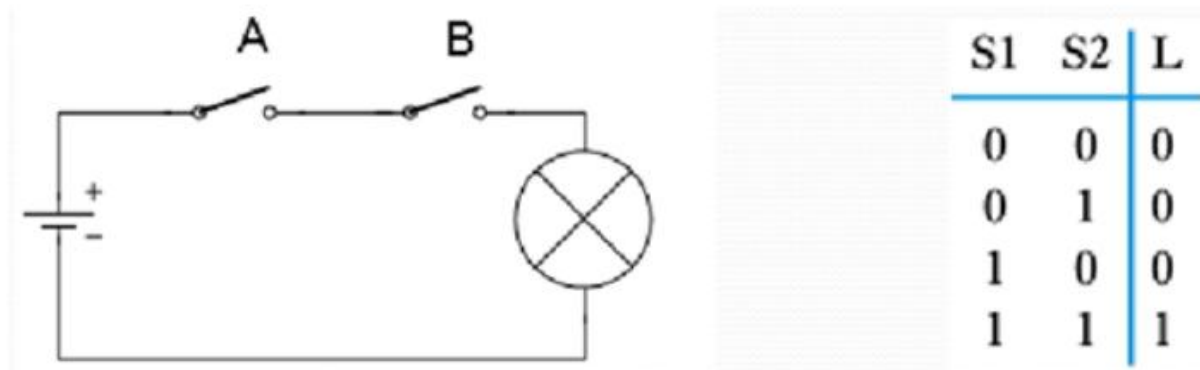
SUD B: „Temperatura motora je prešla 90 °C.”

Svaki od tih sudova u danom trenutku ima točno određenu logičku vrijednost.

Logički sudovi mogu se povezivati pomoću osnovnih logičkih operacija, među kojima su najvažnije operacije I, ILI i negacija. Te operacije omogućuju izgradnju složenijih logičkih izraza na temelju jednostavnih sudova.

Logički kombinator I

Logička operacija I, koja se naziva i konjunkcija, djeluje na dvije izjave i ima istinit rezultat samo u slučaju kada su obje izjave istinite. Ako se sudovi A i B povežu operacijom I, tada je rezultat istinit jedino kada su i A i B istiniti. Takav se odnos može prikazati i pomoću jednostavnog električnog sklopa u kojem su dvije sklopke povezane serijski. Žarulja u tom sklopu svijetli samo onda kada su obje sklopke uključene, što odgovara situaciji u kojoj su oba logička suda istinita.



Operacija I može se zapisivati na različite načine, ovisno o području primjene.

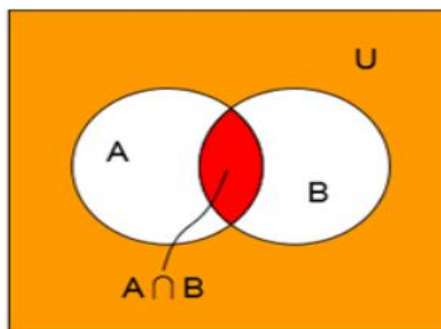
Algoritamski

- ako (A istinit) i (B istinit)
- onda f istinit
- inače f neistinit

Računarska notacija: $f = A \cdot B = AB$

Simbolička logika: $f = A \wedge B$

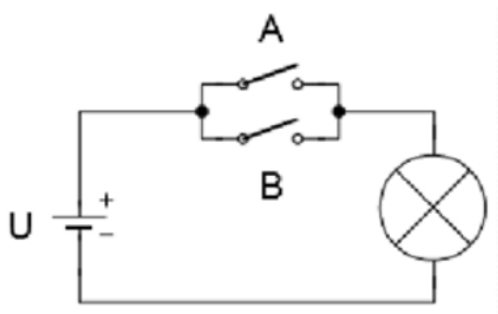
Teorija skupova: $f = A \cap B$



Logički kombinator II

Logička operacija II, poznata i kao disjunkcija, djeluje na dvije izjave i ima istinit rezultat ako je barem jedna od izjava istinita. Ako se sudovi A i B povežu operacijom II, rezultat će biti istinit u svim slučajevima osim kada su oba suda neistinita. U električkom modelu

ova se operacija može prikazati pomoću paralelno spojenih sklopki. Žarulja svijetli ako je uključena barem jedna sklopka, što odgovara istinitosti barem jednog logičkog suda.



I operacija ILI ima različite zapise.

Algoritamski prikaz:

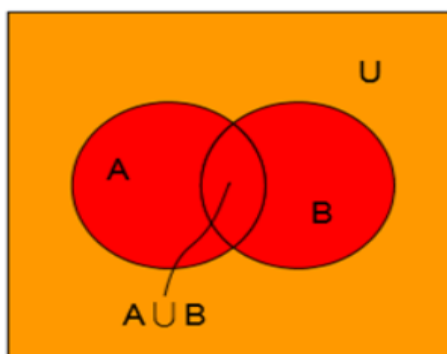
ako (A istinit) ili (B istinit)

- onda f istinit
- inače f neistinit

Računarska notacija: $f = A + B$

Simbolička logika: $f = A \vee B$

Teorija skupova: $f = A \cup B$



Logički kombinator NE

Treća osnovna logička operacija jest negacija, koja se još naziva komplement ili inverzija. Negacija djeluje na jedan logički sud i daje rezultat suprotne vrijednosti. Ako je sud istinit, njegova negacija je neistinita, a ako je sud neistinit, njegova negacija je istinita.

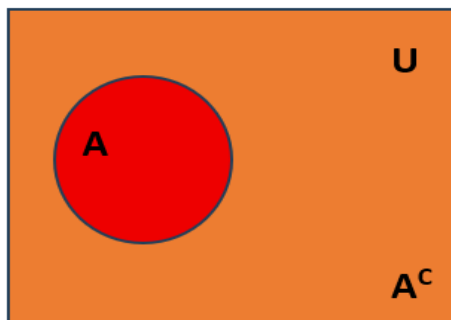
Algoritamski zapis:

- ako (A istinit)
- onda f neistinit
- inače f istinit

Računarska notacija: $f = \bar{A}$

Simbolička logika: $f = \neg A$

Teorija skupova: $f = A^c$



Ove osnovne logičke operacije čine temelj Booleove algebre i omogućuju izgradnju složenih logičkih funkcija. Njihova primjena nalazi se u projektiranju digitalnih sklopova, radu procesora, upravljanju programskim tokovima i obradi podataka u svim suvremenim računalnim sustavima.

5.2 Logička vrata

Logička vrata predstavljaju elektroničke sklopove koji služe za ostvarenje logičkih operacija u digitalnim sustavima. Njihov rad temelji se na matematičkoj logici i pravilima Booleove algebre, a koriste se za obradu binarnih podataka. Svaka logička vrata izvode točno određenu logičku funkciju nad jednim ili više ulaznih signala te na temelju toga daju izlazni signal.

Rad računala zasniva se na činjenici da elektronički sklopovi mogu razlikovati samo dva stabilna stanja. Ta se stanja u digitalnoj logici interpretiraju kao logička nula i logička jedinica, pri čemu svako stanje odgovara određenoj razini napona. Logička nula obično je povezana s nižom razinom napona, primjerice 0 V, dok je logička jedinica povezana s višom razinom napona, primjerice 5 V. Upravo ta dva stanja omogućuju pouzdanu i jednoznačnu obradu informacija u digitalnim sustavima.

Složeni elektronički sklopovi u računalu izgrađeni su od jednostavnih osnovnih elemenata koji se nazivaju logička vrata. Iako suvremeni procesori i memorijski sustavi sadrže vrlo velik broj tranzistora i logičkih elemenata, cijeli sustav može se promatrati kao kombinacija velikog broja jednostavnih logičkih vrata. Takvi složeni logički sklopovi mogu sadržavati stotine, tisuće, a u današnjim računalima i milijune osnovnih logičkih vrata.

Bez obzira na razinu složenosti nekog logičkog sklopa, njegovo se ponašanje uvijek može opisati na formalan način. Svaki logički sklop moguće je prikazati shematskim crtežom, zapisati pripadajući logički izraz te definirati njegovo ponašanje pomoću tablice istinitosti. Time se omogućuje precizna analiza rada sklopa i provjera njegove ispravnosti.

Svaka vrsta logičkih vrata ima jasno definiranu logiku rada koja omogućuje izgradnju složenijih logičkih funkcija. Osnovna logička vrata, poput vrata I, ILI i NE, predstavljaju temelj iz kojeg se izvode svi složeniji digitalni sklopovi. Kombiniranjem tih osnovnih vrata moguće je realizirati aritmetičke operacije, upravljanje tokom programa, memorijske elemente i brojne druge funkcije unutar računalnog sustava.

Funkcioniranje logičkih vrata može se opisati na dva osnovna načina. Prvi način je tablica istinitosti, koja za svaku moguću kombinaciju ulaznih vrijednosti prikazuje odgovarajuću izlaznu vrijednost. Drugi način je zapis pomoću Booleove algebre, koji matematički opisuje odnos između ulaznih i izlaznih signala. Oba prikaza koriste se u projektiranju i analizi digitalnih sklopova te omogućuju jednoznačno razumijevanje rada logičkih vrata.

Logička vrata	Tablica istinitosti	Booleov izraz																	
AND 	<table><tr><th colspan="2">INPUT</th><th rowspan="2">OUTPUT</th></tr><tr><th>A</th><th>B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		OUTPUT	A	B	0	0	0	1	0	0	0	1	0	1	1	1	$C = A * B$
INPUT		OUTPUT																	
A	B																		
0	0	0																	
1	0	0																	
0	1	0																	
1	1	1																	
OR 	<table><tr><th colspan="2">INPUT</th><th rowspan="2">OUTPUT</th></tr><tr><th>A</th><th>B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		OUTPUT	A	B	0	0	0	1	0	1	0	1	1	1	1	1	$C = A + B$
INPUT		OUTPUT																	
A	B																		
0	0	0																	
1	0	1																	
0	1	1																	
1	1	1																	
NOT 	<table><tr><th>INPUT</th><th rowspan="2">OUTPUT</th></tr><tr><th>A</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	INPUT	OUTPUT	A	0	1	1	0	$B = A$										
INPUT	OUTPUT																		
A																			
0	1																		
1	0																		
NAND 	<table><tr><th rowspan="2">OUTPUT</th><th colspan="2">INPUT</th></tr><tr><th>B</th><th>A</th></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr></table>	OUTPUT	INPUT		B	A	1	0	0	1	0	1	1	1	0	0	1	1	$C = A * B \text{ gore}$
OUTPUT	INPUT																		
	B	A																	
1	0	0																	
1	0	1																	
1	1	0																	
0	1	1																	
NOR 		$C = A + B \text{ gore}$																	

	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th><th rowspan="2">OUTPUT</th></tr> <tr> <th>A</th><th>B</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	1	1	0	0	0	1	0	1	1	0	
INPUT		OUTPUT																	
A	B																		
0	0	1																	
1	0	0																	
0	1	0																	
1	1	0																	
XOR 	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th><th rowspan="2">OUTPUT</th></tr> <tr> <th>A</th><th>B</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	0	1	0	1	0	1	1	1	1	0	C = A krug B
INPUT		OUTPUT																	
A	B																		
0	0	0																	
1	0	1																	
0	1	1																	
1	1	0																	
XNOR 	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th><th rowspan="2">OUTPUT</th></tr> <tr> <th>A</th><th>B</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	INPUT		OUTPUT	A	B	0	0	1	1	0	0	0	1	0	1	1	1	C = AB krug gore
INPUT		OUTPUT																	
A	B																		
0	0	1																	
1	0	0																	
0	1	0																	
1	1	1																	

5.3 Booleove jednakosti

Booleove jednakosti (zakoni Booleove algebre) definiraju temeljna pravila po kojima se logički izrazi mogu mijenjati, pojednostavljivati i međusobno dokazivati kao ekvivalentni. Njihova je osnovna svrha pojednostavljenje logičkih funkcija kako bi se smanjio broj logičkih vrata i veza u digitalnim sklopovima, što izravno doprinosi manjoj složenosti, manjoj potrošnji energije i većoj pouzdanosti sustava.

Najvažnije Booleove jednakosti grupiraju se u nekoliko osnovnih skupina prikazanih u Tablici.

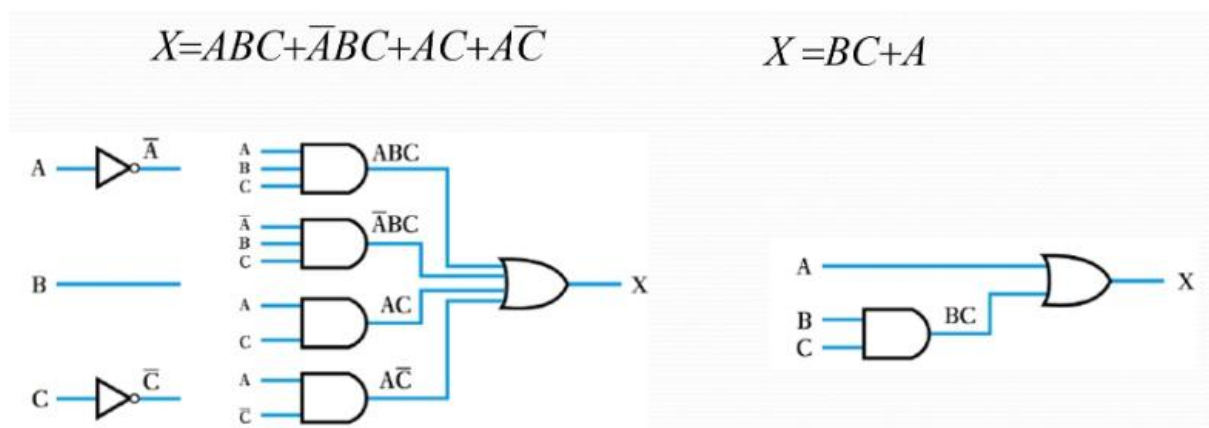
Tablica 3 Teoremi Booleove algebre

Zakon	ILI (+)	I (·)
Komutativnost	$A + B = B + A$	$A \cdot B = B \cdot A$
Asocijativnost	$(A + B) + C = A + (B + C)$	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Distributivnost	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	
Neutralni element	$A + 0 = A$	$A \cdot 1 = A$
Idempotentnost	$A + A = A$	$A \cdot A = A$

Komplementarnost	$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$
De Morganovi zakoni	$\overline{A + B} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$
Involutivnost	$\bar{\bar{A}} = A$	
Anihilacija	$A + 1 = 1$	$A \cdot 0 = 0$
Apsorpcija	$A \cdot (A + B) = A$	$A + A \cdot B = A$

Primjenom ovih jednakosti moguće je dokazati ekvivalentnost različitih logičkih izraza te pronaći njihov minimalni oblik. U praksi se Booleove jednakosti često koriste zajedno s tablicama istinitosti, osobito pri projektiranju kombinaćijskih digitalnih sklopova.

Primjer 1.



Primjer 2. Minimizirajte sljedeći Booleov izraz.

$$A \cdot C \cdot (\bar{A} + B) + B \cdot \bar{C} \cdot (A + \bar{B})$$

Rješenje:

$$A \cdot C \cdot \bar{A} + A \cdot C \cdot B + B \cdot \bar{C} \cdot A + B \cdot \bar{C} \cdot \bar{B} =$$

$$A \cdot \bar{A} \cdot C + A \cdot C \cdot B + B \cdot \bar{C} \cdot A + B \cdot \bar{B} \cdot C =$$

$$0 \cdot C + A \cdot C \cdot B + B \cdot \bar{C} \cdot A + 0 \cdot C =$$

$$A \cdot C \cdot B + B \cdot \bar{C} \cdot A =$$

$$A \cdot B \cdot (C + \bar{C}) =$$

$$A \cdot B$$

Primjer 3. Minimizirajte sljedeći Booleov izraz.

$$A \cdot (\bar{B} + C) + B \cdot (A \cdot C + B)$$

Rješenje:

$$A \cdot (\bar{B} + C) + B \cdot (A \cdot C + B) =$$

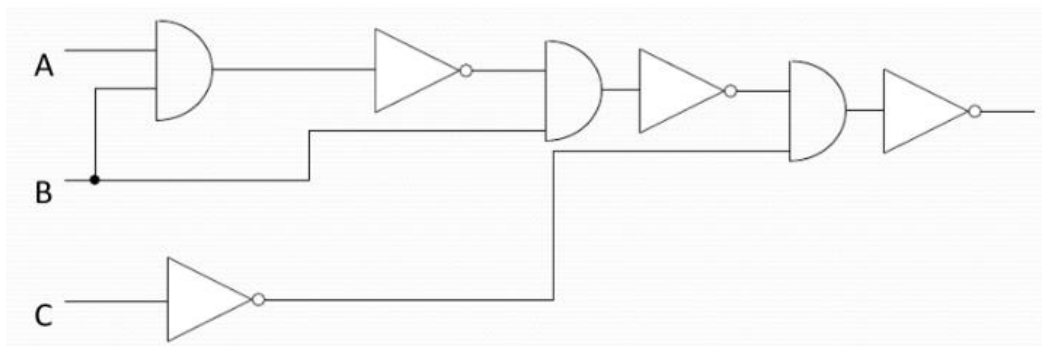
$$A \cdot \bar{B} + A \cdot C + A \cdot B \cdot C + B =$$

$$A \cdot \bar{B} + \bar{C} \cdot B + B \cdot (A \cdot C + 1) =$$

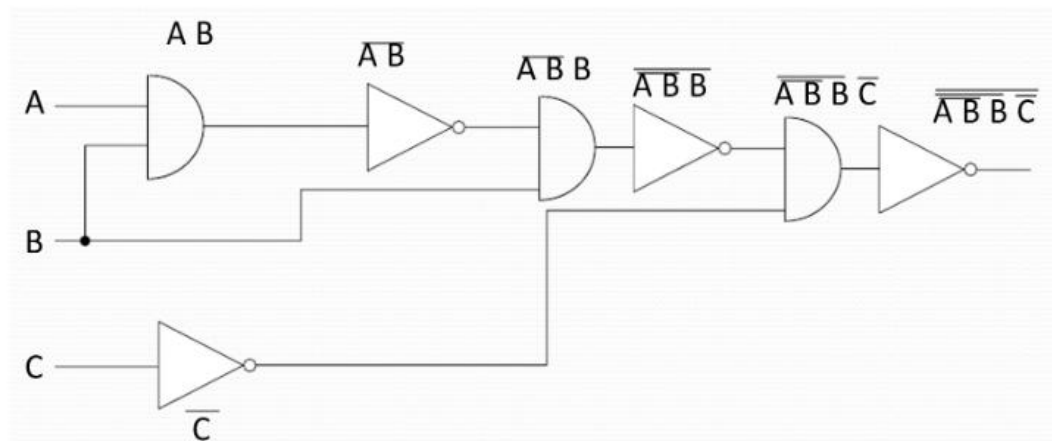
$$A \cdot \bar{B} + B =$$

$$A \cdot \bar{B}$$

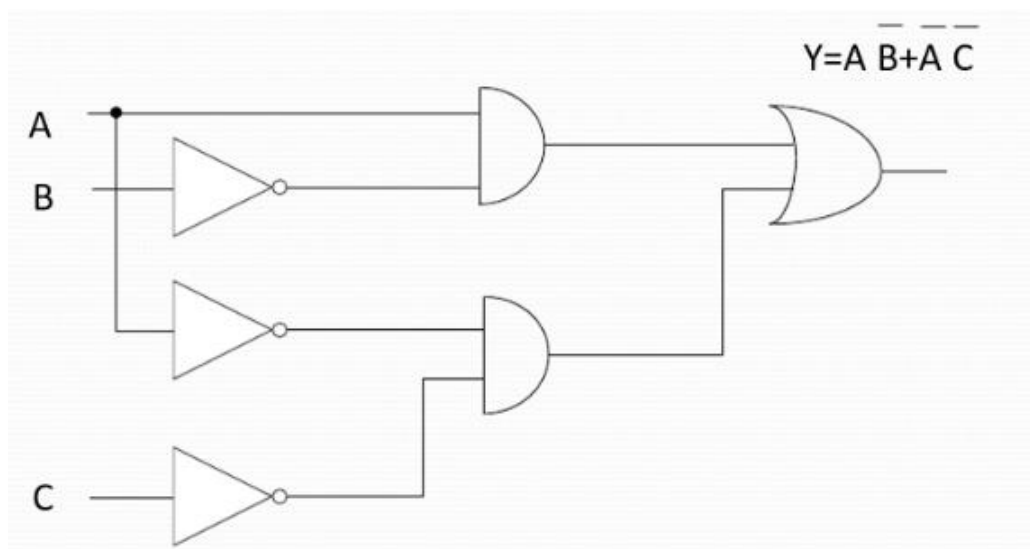
Primjer 4. Napišite logički izraz za sljedeći logički sklop.



Rješenje:



Primjer 5. Napišite tablicu istinitosti za sljedeći logički sklop.

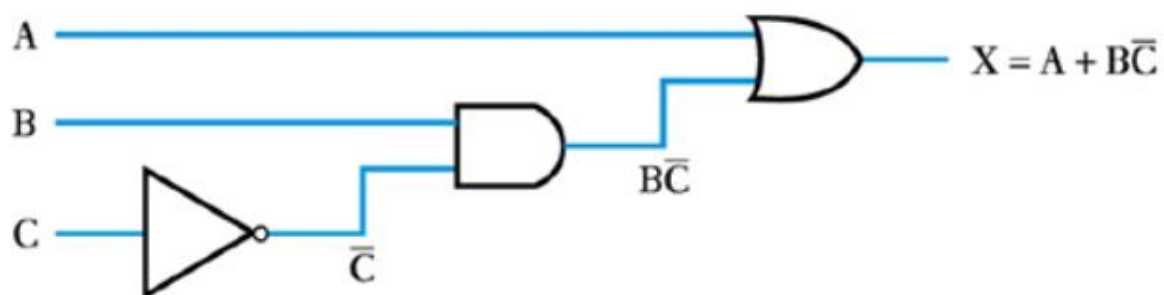


Rješenje:

A	B	C	$\bar{B}$	$A \cdot \bar{B}$	$\bar{A}$	$\bar{C}$	$\bar{A} \cdot \bar{C}$	Y
0	0	0	1	0	1	1	1	1
0	0	1	1	0	1	0	0	0
0	1	0	0	0	1	1	1	1
0	1	1	0	0	1	0	0	0
1	0	0	1	1	0	1	0	1
1	0	1	1	1	0	0	0	1
1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0

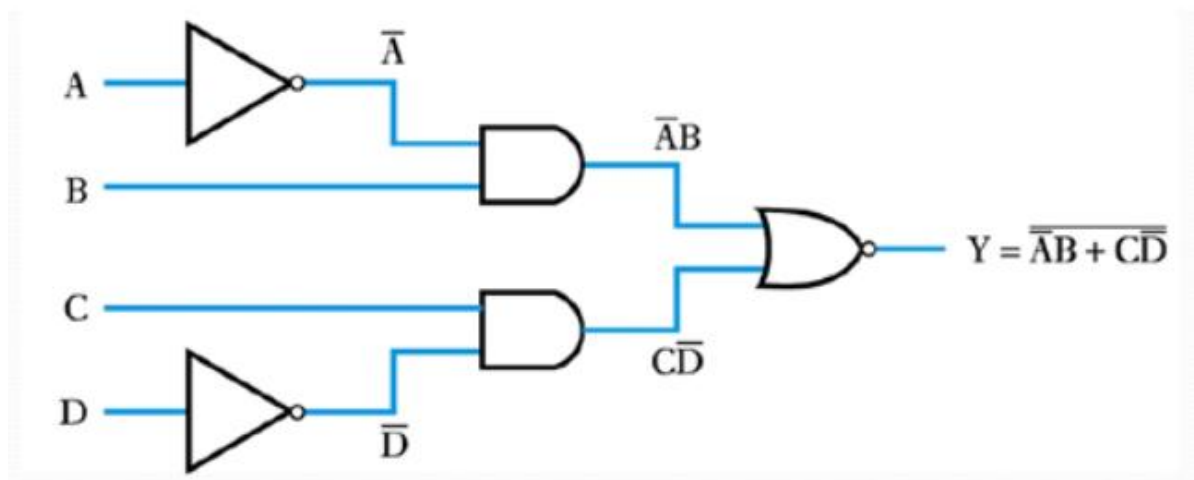
Primjer 6. Pretvorite Booleov izraz $X = A + B\bar{C}$ u logička vrata.

Rješenje:



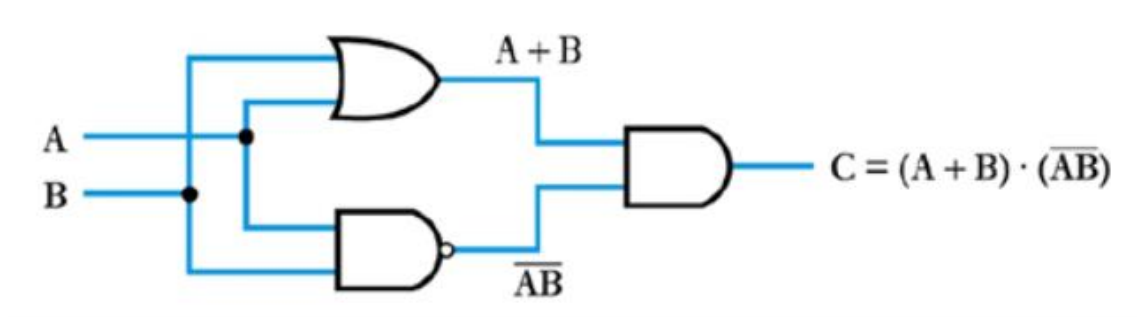
Primjer 7. Pretvorite Booleov izraz $Y = \overline{\overline{A}B + C\overline{D}}$ u logička vrata.

Rješenje:



Primjer 8. Prikažite Booleovim izrazom i logičkim vratima sljedeću tvrdnju: izlaz sklopa je istinit ako je barem jedan od ulaza istinit, ali ne i ako su oba istinita, drugim riječima, izlaz je istinit ako su: A ILI B istiniti te A I B nisu istiniti.

Rješenje:



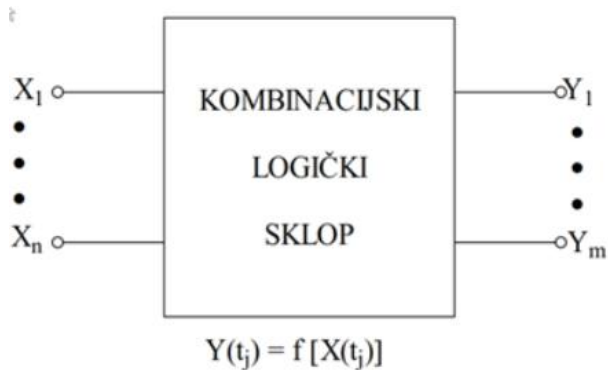
5.4 Logički sklopovi

Digitalni logički sklopovi osnova su građe digitalnih elektroničkih računala. U tu skupinu ubrajaju se i osnovni logički sklopovi za ostvarivanje funkcija I (AND), ILI (OR), NE (NOT) i NILI (NOR). Ovi sklopovi omogućuju izvođenje osnovnih logičkih operacija nad binarnim signalima.

Međutim, osnovne logičke funkcije same po sebi nisu dovoljne za izgradnju složenih digitalnih sustava. Za realizaciju složenijih zadataka, kao što su kodiranje, dekodiranje, ispitivanje pariteta, pretvorbe kodova i slične operacije, koriste se kombinacijski i sekvencijalni logički sklopovi.

Kombinacijski logički sklopovi

Kombinacijski logički sklopovi su međusobno povezane grupe logičkih sklopova kod kojih je izlazni signal u svakom trenutku isključivo funkcija trenutačnih ulaznih signala. Kod ovih sklopova ne postoji memorija prethodnih stanja.



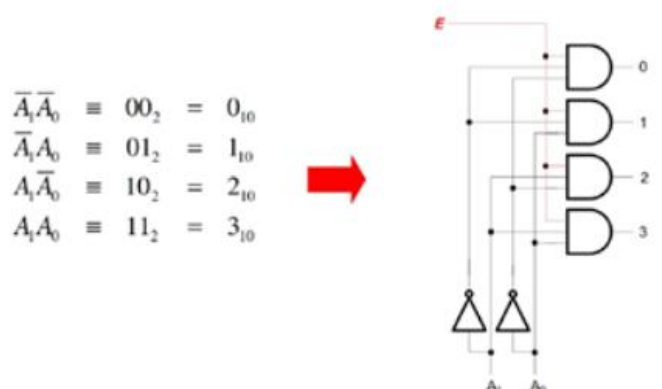
Slika 27 Kombinacijski logički sklop

Tipični primjeri kombinacijskih logičkih sklopova su:

- dekoderi
- multipleksori
- demultipleksori
- aritmetički sklopovi

Dekoder je kombinacijski logički sklop kod kojeg za svaki izlaz postoji točno jedna binarna ulazna riječ za koju će taj izlaz biti u stanju 1, dok su za sve ostale ulazne kombinacije svi ostali izlazi u stanju 0.

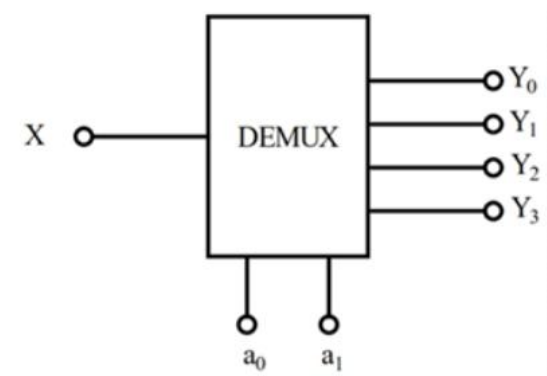
Dekoder se najčešće sastoji od n I-sklopova, a označava se omjerom broja ulaza i izlaza, primjerice dekodeer 2/4, što znači da ima 2 ulaza i 4 izlaza.



Slika 28 Dekoder 2/4

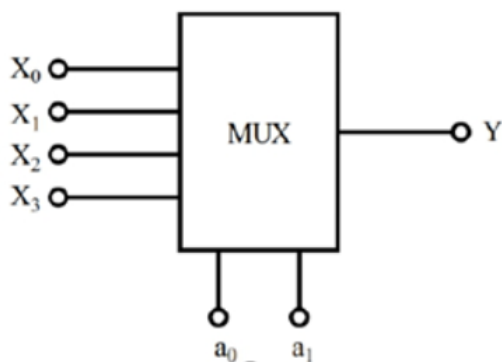
Dekoder je univerzalni kombinacijski modul jer se njegovom primjenom mogu ostvariti sve logičke funkcije s onoliko varijabli koliko dekodeer ima ulaza.

Demultipleksor je logički sklop koji prima podatak na jednom ulazu i raspodjeljuje ga na jedan od najviše 2^n mogućih izlaza, ovisno o stanju adresnih (upravljačkih) ulaza. Logička vrijednost podatkovnog ulaza X pojavit će se na onom izlazu koji odgovara kombinaciji upravljačkih bitova (npr. $a_0, a_1, \dots$). Demultipleksori se često koriste za raspodjelu signala i upravljanje podacima u digitalnim sustavima.



Slika 29 Demultipleksor

Multipleksor je logički sklop koji povezuje više ulaznih linija na jednu izlaznu liniju. Na temelju stanja adresnih (selektorskih) ulaza odabire se jedna ulazna linija čija se vrijednost proslijeđuje na izlaz. Multipleksori se koriste u digitalnim krugovima za usmjeravanje podataka i kontrolu signala.



Slika 30 Multipleksor

Zbog velikog troška i vremena potrebnog za dizajn svakog pojedinog specijaliziranog čipa, razvijeni su programibilni logički sklopovi opće namjene.

PLA (eng. *Programmable Logic Array*) koristi činjenicu da se svaka Booleova funkcija može izraziti pomoću logičkih operacija NE, I i ILI. PLA omogućuje programibilno definiranje logičkih funkcija prema potrebama sustava.

Tipični PLA sklopovi mogu sadržavati nekoliko stotina logičkih sklopova, dvadesetak ulaza i desetak izlaza.

Aritmetički sklopovi

Aritmetički sklopovi realizirani su unutar aritmetičko-logičke jedinice (ALU) procesora. Njihova osnovna namjena je izvođenje operacija nad cijelim brojevima.

Osnovni aritmetički sklopovi su:

- binarno zbrajalo (zbrajanje, oduzimanje, množenje i dijeljenje)
- jedinica za logičke operacije (I, ILI, NE i ostala logička vrata)
- sklop za posmak (množenje i dijeljenje potencijama broja 2)

Pitanja za ponavljanje

1. Zašto se svi podaci u digitalnim sustavima svode na binarni oblik?
2. Što je logički sud?
3. Koje su tri osnovne logičke operacije u Booleovoj algebri?
4. Što su logička vrata?
5. Koja je osnovna razlika između kombinacijskih i sekvencijalnih logičkih sklopova?
6. Što su kombinacijski logički sklopovi?
7. Navedite tipične primjere kombinacijskih logičkih sklopova.
8. Što je dekodler?
9. Što su multipleksor i demultipleksor?
10. Koja je osnovna namjena aritmetičkih sklopova?

6 Zapis podataka u računalu

Računala se sastoje od elektroničkih sklopova koji mogu raspoznavati samo dva različita stanja. Zbog takve građe svi podaci koji ulaze u računalo moraju biti prikazani u obliku u kojem postoje samo ta dva stanja.

Brojevni sustav predstavlja način zapisivanja brojeva i njihovo tumačenje. U računalstvu se najčešće koristi položajni brojevni sustav, u kojem položaj znamenke u zapisu određuje njezinu vrijednost. Svaki položajni brojevni sustav definiran je vlastitim skupom znamenaka, a ukupan broj različitih znamenaka naziva se osnova ili baza brojevnog sustava.

Kako bi se razlikovao brojevni sustav u kojem je broj zapisan, uz broj se u indeksu navodi njegova baza. Najčešće korišteni brojevni sustavi su:

- binarni (0, 1)
- oktalni (0, 1, 2, 3, 4, 5, 6, 7)
- dekadski (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
- heksadekadski (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

Postoje različite metode pomoću kojih se brojevi mogu pretvarati iz jedne baze u drugu, što je česta potreba u radu s računalnim sustavima.

6.1 Brojevni sustavi

Osnovna namjena računala je pohranjivanje, pretraživanje i izvođenje operacija nad podacima (brojevima, znakovima i signalima). Brojevni sustav koji se koristi za zapis podataka mora zadovoljiti sljedeće zahtjeve:

- jednostavno izvođenje logičkih i aritmetičkih operacija
- jednostavna realizacija pomoću elektroničkih elemenata
- dovoljno velik kapacitet i preciznost zapisa
- mogućnost zapisa signala pomoću brojevnog sustava

Zbog tehnološke izvedbe računala, podaci se u računalu zapisuju koristeći binarni brojevni sustav.

Binarni brojevni sustav je sustav baze 2, koji koristi znamenke 0 i 1.

Osnovne jedinice zapisa

BIT zauzima jedno binarno mjesto koje može poprimiti vrijednost {0, 1}. Naziv dolazi od engleskog izraza *binary digit* i predstavlja najmanju količinu informacije.

BAJT (BYTE) čini 8 bitova. Naziv dolazi od engleskog izraza *binary term*.

RIJEČ je zapis podataka u računalima duljine 1, 2, 4, 8, ... bajtova. Duljina riječi ovisi o arhitekturi procesora.

Pregled brojevnihi sustava

- **B = 2** {0, 1} – binarni brojevni sustav
- **B = 8** {0, 1, 2, 3, 4, 5, 6, 7} – oktalni brojevni sustav
- **B = 10** {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} – dekadski brojevni sustav
- **B = 16** {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F} – heksadekadski brojevni sustav

Heksadekadski brojevni sustav često se koristi jer je blizak binarnom i oktalnom sustavu, dok je zapis brojeva znatno kraći i pregledniji.

Kapacitet binarnog zapisa

Ukupan broj različitih brojeva koji se mogu prikazati binarnim zapisom duljine n bitova iznosi 2^n :

- $2^8 = 256$
- $2^{10} = 1\,024$
- $2^{16} = 65\,536$
- $2^{32} = 4\,294\,967\,296$

Najveći cijeli nepredznačni broj (*unsigned integer*) koji se može prikazati n -bitnim binarnim zapisom iznosi $2^n - 1$.

Na primjer, za $n = 8$ bitova: $2^8 - 1 = 256 - 1 = 255$.

Jedinice kapaciteta

- kilobajt (kB) = $1\,024\text{ B} = 2^{10} \approx 10^3$
- megabajt (MB) = $1\,024\text{ kB} = 2^{20} \approx 10^6$
- gigabajt (GB) = $1\,024\text{ MB} = 2^{30} \approx 10^9$
- terabajt (TB) = $1\,024\text{ GB} = 2^{40} \approx 10^{12}$
- petabajt (PB), eksabajt (EB), zetabajt (ZB), ...

Tablica 4 Prikaz brojeva u različitim bazama

binarni	dekadski	heksadekadski
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F ⁸

6.2 Brojevni sustavi – pretvorba

U radu s računalima često je potrebno pretvarati brojeve iz jednog brojevnog sustava u drugi. Najčešće pretvorbe su između binarnog, dekadskog, oktalnog i heksadekadskog sustava. U nastavku su opisane osnovne metode pretvorbe, uz ilustrativne primjere.

Pretvorba iz binarnog u dekadski sustav

Pretvorba iz binarnog u dekadski sustav temelji se na težinama pojedinih bitova. Svaka znamenka binarnog broja množi se s odgovarajućom potencijom broja 2, ovisno o njezinu položaju.

Opći oblik:

$$(b_{n-1}b_{n-2} \dots b_0)_2 = \sum_{i=0}^{n-1} b_i \cdot 2^i$$

Primjer:

Pretvoriti binarni broj 10101111_2 u dekadski sustav.

$$\begin{aligned}
 &1 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\
 &= 128 + 32 + 8 + 4 + 2 + 1 = 175
 \end{aligned}$$

$$10101111_2 = 175_{10}$$

Pretvorba iz dekadskog u binarni sustav

Pretvorba iz dekadskog u binarni sustav izvodi se uzastopnim dijeljenjem s 2, pri čemu se pamte ostaci. Binarni zapis dobiva se čitanjem ostataka odozdo prema gore.

Primjer:

Pretvoriti dekadski broj 175_{10} u binarni sustav.

$$\begin{array}{rcl} 175 : 2 & = & 87; r = 1 \\ 87 : 2 & = & 43; r = 1 \\ 43 : 2 & = & 21; r = 1 \\ 21 : 2 & = & 10; r = 1 \\ 10 : 2 & = & 5; r = 0 \\ 5 : 2 & = & 2; r = 1 \\ 2 : 2 & = & 1; r = 0 \\ 1 : 2 & = & 0; r = 1 \end{array}$$

Čitanjem ostataka odozdo prema gore dobiva se:

$$175_{10} = 10101111_2$$

Pretvorba iz binarnog u oktalni sustav

Binarni sustav ima bazu 2, a oktalni sustav bazu 8, pri čemu vrijedi:

$$8 = 2^3$$

Iz toga slijedi da jedna znamenka oktalnog sustava odgovara trima znamenkama binarnog sustava.

Binarno	Oktalno
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Postupak:

1. Binarni broj se grupira u trojke bitova, počevši s desne strane.
2. Svaka trojka zamjenjuje se odgovarajućom oktalnom znamenkom.

Primjeri:

$$\begin{aligned}
&101\,011\,111_2 \\
&101_2 = 5_8, 011_2 = 3_8, 111_2 = 7_8 \\
&101\,011\,111_2 = 537_8
\end{aligned}$$

$$10111101_2 = 10\,111\,101_2 \text{ (krajnju lijevu grupu nadopunimo nulama)} = 010\,111\,101_2 = 275_8$$

$$1110110111_2 = 001\,110\,110\,111_2 = 1677_8$$

$$101111011000_2 = 101\,111\,011\,000_2 = 5730_8$$

Pretvorba iz oktalnog u binarni sustav

Pretvorba iz oktalnog u binarni sustav izvodi se tako da se svaka oktalna znamenka zamijeni njezinim binarnim ekvivalentom od tri bita.

Primjer:

$$\begin{aligned}
&537_8 \\
&5_8 = 101_2, 3_8 = 011_2, 7_8 = 111_2 \\
&537_8 = 101\,011\,111_2
\end{aligned}$$

Pretvorba iz binarnog u heksadekadski sustav

Binarni sustav ima bazu 2, a heksadekadski sustav bazu 16, pri čemu vrijedi:

$$16 = 2^4$$

Jedna znamenka heksadekadskog sustava odgovara četirima znamenkama binarnog sustava.

Binarni zapis	Heksadecimalni zapis
0000	0
0001	1
0010	2
0100	3
1000	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B

1100	C
1101	D
1110	E
1111	F

Postupak:

1. Binarni broj se grupira u četvorke bitova s desne strane.
2. Svaka četvorka zamjenjuje se odgovarajućom heksadekadskom znamenkom.

Primjeri:

$$110101000_2 = 1\ 1010\ 1000_2 = (\text{krajnje lijevu grupu nadopunimo nulama}) = 0001\ 1010\ 1000_2 = 1A8_{16}$$

$$101110111011_2 = 1011\ 1011\ 1011_2 = BBB_{16}$$

$$10101111_2 = 1010\ 1111_2 = AF_{16}$$

Pretvorba iz heksadekadskog u binarni sustav

Pretvorba iz heksadekadskog u binarni sustav izvodi se tako da se svaka heksadekadaska znamenka zamijeni njezinim binarnim ekvivalentom od četiri bita. Nule na početku binarnog zapisa mogu se po potrebi izostaviti.

Primjeri:

$$10A_{16} = 0001\ 0000\ 1010_2 = 100001010_2$$

$$2F_{16} = 0010\ 1111_2 = 101111_2$$

$$C3_{16} = 1100\ 0011_2$$

6.3 Prikaz negativnih brojeva

U računalnim sustavima cijeli brojevi prikazuju se u binarnom obliku koristeći različite metode za prikaz predznaka i vrijednosti broja. Za označavanje predznaka koristi se najljeviji bit

- najljeviji bit = **0** → pozitivan broj
- najljeviji bit = **1** → negativan broj

U nastavku su opisane najvažnije metode zapisa cijelih brojeva u računalu.

Metoda predznaka i apsolutne vrijednosti (*sign-magnitude*)

Kod ove metode cijeli broj se sastoji od:

- jednog bita za predznak
- preostalih $n - 1$ bitova za apsolutnu vrijednost

Predznak „+” prikazuje se binarnom znamenkom 0, a predznak „-” binarnom znamenkom 1. Za zapis predznaka dovoljan je jedan bit, dok se ostali bitovi koriste za pohranu vrijednosti broja. To znači da:

- u 8-bitnom registru imamo 7 bitova za vrijednost broja
- u 16-bitnom registru imamo 15 bitova
- u 32-bitnom registru imamo 31 bit

Raspon brojeva koji se može prikazati n-bitnim zapisom iznosi:

$$-(2^{n-1} - 1) \text{ do } +(2^{n-1} - 1)$$

Primjeri (8-bitni zapis):

- $+17 = 00010001$
- $-17 = 10010001$

Nedostaci metode:

- broj 0 ima dva zapisa: $+0$ i -0
- aritmetičke operacije (zbrajanje i oduzimanje) su složene jer je potrebno posebno pratiti predznak

Zbog navedenih nedostataka ova metoda se rijetko koristi u praksi.

Metoda predznaka i 1-komplementa

Kod metode 1-komplementa negativan broj dobiva se invertiranjem svih bitova pozitivnog broja:

- $0 \rightarrow 1$
- $1 \rightarrow 0$

Primjeri (8-bitni zapis):

- $+17 = 00010001 \rightarrow -17 = 11101110$
- $+127 = 01111111 \rightarrow -127 = 10000000$
- $+0 = 00000000 \rightarrow -0 = 11111111$

Nedostatak metode:

- također postoji dvostruki zapis broja 0
- aritmetičke operacije su i dalje relativno složene

Zbog toga se i ova metoda rijetko koristi u modernim računalima.

Metoda predznaka i 2-komplementa

Metoda 2-komplementa danas je standardni način prikaza negativnih cijelih brojeva u računalnim sustavima.

Postupak dobivanja 2-komplementa:

1. Izračuna se 1-komplement binarnog broja (invertiranje svih bitova)
2. Dobivenom zapisu pribroji se 1

Dobiveni broj predstavlja zapis negativnog broja.

Primjeri (8-bitni zapis):

Kako izgleda zapis broja 101 u 8-bitnom registru?

Najprije zapišimo broj +101:

$$101 = 64 + 32 + 4 + 1 \Rightarrow 1100101_2$$

Dopunjujemo na 8 bitova:

$$+101 = 01100101$$

1-komplement:

$$10011010$$

Dodamo 1:

$$\begin{array}{r} 10011011 \\ -101 = 10011011 \end{array}$$

- $+17 = 00010001 \rightarrow -17 = 11101111$
- $+127 = 01111111 \rightarrow -127 = 10000001$
- $+0 = 00000000 \rightarrow -0 = 00000000$

Prednosti metode 2-komplementa:

- broj 0 ima samo jedan zapis
- zbrajanje i oduzimanje pozitivnih i negativnih brojeva izvodi se istim sklopovima
- jednostavna i učinkovita implementacija u sklopovlju

Zbog navedenih prednosti metoda 2-komplementa koristi se u gotovo svim modernim procesorima.

6.3.1 Prikaz realnih brojeva

Realni brojevi sastoje se od cjelobrojnog dijela (prije decimalne točke) i decimalnog dijela (nakon decimalne točke).

U svakodnevnom životu realne brojeve zapisujemo u obliku s fiksnom točkom (npr. 312.422, 15.81, 2515436.2513). Takav zapis nije prikladan za vrlo velike ili vrlo male

brojeve, pa se koristi eksponencijalni zapis koji se sastoji od mantise (normalizirane vrijednosti) i eksponenta.

Primjer:

$$3.12422 \cdot 10^2$$

3.12422 je normalizirana mantisa, a eksponent je 2.

Na sličan način se zapisuju realni brojevi u računalima. Za zapis realnih brojeva u računalu koristi se IEEE 754 standard.

Postupak zapisa:

1. Odredi se predznak (0 za +, 1 za -)
2. Broj se pretvori u binarni oblik
3. Zapiše se u normaliziranom obliku $1.x \cdot 2^e$
4. Eksponentu se pribraja 127
5. Popune se 32 bita:
 - 1 bit predznaka
 - 8 bitova eksponenta
 - 23 bita mantise

Brojevi u IEEE 754 standardu se često prikazuju u heksadekadskom obliku radi kraćeg zapisa.

6.4 Binarne operacije

Binarno zbrajanje

Binarno zbrajanje temelji se na zbrajanju pojedinih bitova, pri čemu se po potrebi pojavljuje prijenos (*carry*) u sljedeći viši bit.

0+0	= 0
0+1	= 1
1+0	= 1
1+1	= 0 i 1 dalje

Primjer 1: Binarno zbrojite sljedeća dva broja: $11011_{(2)}$ i $1011_{(2)}$

Rješenje:

prijenos	1	1		1	1	
+		1	1	0	1	1
			1	0	1	1

rezultat	1	0	0	1	1	0
----------	---	---	---	---	---	---

Primjer 2: Binarno zbrojite sljedeća dva broja: $111100_{(2)}$ i $1110_{(2)}$

Rješenje:

prijenos	1	1	1	1			
+		1	1	1	1	0	0
				1	1	1	0
rezultat	1	0	0	1	0	1	0

Primjer 3: Binarno zbrojite sljedeća dva broja: $1000101111_{(2)}$ i $11101111_{(2)}$

Rješenje:

prijenos			1	1	1		1	1	1	1	
+		1	0	0	0	1	0	1	1	1	1
				1	1	1	0	1	1	1	1
rezultat		1	1	0	0	0	1	1	1	1	0

Binarno zbrajanje u računalu izvodi se pomoću logičkih sklopova (poluzbrajala i punih zbrajala), a isti postupak koristi se neovisno o duljini binarnog zapisa.

Binarno množenje

Binarno množenje temelji se na istim principima kao i dekadsko množenje, ali je zbog korištenja samo dviju znamenaka (0 i 1) postupak jednostavniji. U praksi se binarno množenje svodi na zbrajanje pomaknutih binarnih brojeva.

Primjer 1: Binarno pomnožite sljedeća dva broja: $111_{(2)}$ i $1010_{(2)}$

Rješenje:

prijenos	1	1	1						
+		1	1	1	*	1	0	1	0
			0	0	0				
				1	1	1			
					0	0	0		
rezultat	1	0	0	0	1	1	0		

Binarno oduzimanje

Binarno oduzimanje u računalima najčešće se ne izvodi izravno, već se svodi na binarno zbrajanje primjenom metode dvojnog komplementa. Time se izbjegava potreba za posebnim sklopovima za oduzimanje jer se i zbrajanje i oduzimanje mogu izvoditi istim aritmetičkim sklopovima.

Oduzimanje dvaju brojeva:

$$A - B$$

svodi se na:

$$A + (-B)$$

Drugim riječima, umanjitelj se najprije pretvara u negativan broj pomoću dvojnog komplementa, a zatim se pribraja umanjeniku.

Postupak binarnog oduzimanja metodom dvojnog komplementa

1. Umanjenik i umanjitelj potrebno je svesti na isti broj bitova. Ako je umanjitelj kraći, s lijeve strane mu se dodaje potreban broj nula.
2. Izračuna se dvojni komplement umanjitelja:
 - izračuna se komplement (invertiranje bitova)
 - dobivenom zapisu pribroji se 1
3. Zbroji se umanjenik i dobiveni dvojni komplement.
4. Krajnja lijeva jedinica (prijenos) se odbacuje, čime se dobiva ispravan rezultat.

Primjer 1: Binarno oduzmite sljedeća dva broja: $11011_{(2)} - 1011_{(2)}$

Prvo primijetimo da umanjenik ima 5 bitova, a umanjitelj 4. Stoga treba umanjitelja dopuniti s 0: $01011_{(2)}$

Komplement tog broja je $10100_{(2)}$, a dvojni komplement je $10101_{(2)}$

Zbrojimo $11011_{(2)}$ i $10101_{(2)}$

Na kraju odbacimo krajnje lijevu jedinicu

prijenos		1	1	1	1	1	
Umanjenik			1	1	0	1	1
Umanjitelj	+		0	1	0	1	1
Dvojni komplement umanjitelja			1	0	1	0	1
zbroj		1	1	0	0	0	0
Rezultat (razlika)			1	0	0	0	0

Metoda dvojnog komplementa omogućuje da se zbrajanje i oduzimanje izvode pomoću istih sklopova u aritmetičko-logičkoj jedinici (ALU), što pojednostavljuje dizajn procesora i povećava učinkovitost.

Binarno dijeljenje

Binarno dijeljenje realizira se na sličan način kao dekadsko dijeljenje, ali zbog binarne prirode brojeva postupak je jednostavniji. U praksi se binarno dijeljenje svodi na ponovljeno oduzimanje i pomicanje bitova.

Osnovna pravila binarnog dijeljenja su:

- ako je djelitelj veći od djeljenika → rezultat je 0
- ako je djelitelj manji ili jednak djeljeniku → rezultat je 1, a od djeljenika se oduzima djelitelj

Postupak se zatim ponavlja nad ostatkom.

Postupak binarnog dijeljenja (ručni algoritam)

1. Djeljenik i djelitelj poravnaju se po najznačajnijem bitu.
2. Provjerava se može li se djelitelj oduzeti od trenutnog dijela djeljenika:
 - ako može → u kvocijent se upisuje 1 i oduzima se djelitelj
 - ako ne može → u kvocijent se upisuje 0
3. Sljedeći bit djeljenika se „spušta“ (pomak ulijevo).
4. Postupak se ponavlja dok se ne obrade svi bitovi djeljenika.

Poseban slučaj: dijeljenje s potencijama broja 2

Ako je djelitelj potencija broja 2, dijeljenje se svodi na pomak udesno:

$$A \div 2^n = A \gg n$$

Primjer:

$$101100_2 \div 2 = 10110_2$$

6.5 Kod i kodiranje

Iz potrebe da računala međusobno komuniciraju i razmjenjuju podatke nastala je potreba za normama za razmjenu podataka između računala i čovjeka, kao i između različite računalne opreme. Bilo je potrebno definirati skup znakova i simbola koji omogućuju jednoznačnu razmjenu podataka.

Uz pomoć unaprijed definiranih pravila, različitim znakovima koje koristimo (slova, brojevi, posebni znakovi) pridjeljuju se binarni brojevi, pri čemu svaki binarni broj predstavlja jedan znak. Takav postupak pridjeljivanja binarnih brojeva znakovima vanjskog svijeta naziva se kodiranje.

Kako bi se izbjegli problemi nekompatibilnosti između različitih sustava, bilo je nužno uvesti normiranje kôdova. Normiranje kôda predstavlja točno definiran način na koji se slova, brojevi i posebni znakovi pretvaraju u binarni oblik prihvatljiv računalu.

U računalima se za kodiranje znakova koristi više kôdnih sustava, od kojih je povijesno najznačajniji ASCII kôd (eng. *American Standard Code for Information Interchange*).

ASCII kôd

ASCII kôd je norma koja propisuje pridjeljivanje sedmerobitnih binarnih brojeva znakovima, čime se omogućuje prikaz ukupno 128 različitih znakova (2⁷).

- prva 32 znaka (0–31) namijenjena su upravljačkim (kontrolnim) znakovima (npr. prijelaz u novi red, kraj retka, zvučni signal) – ti se znakovi ne prikazuju na zaslonu
- znakovi od 32 do 126 su ispisivi znakovi (slova, znamenke, interpunkcijski znakovi, razmaci)
- na poziciji 127 nalazi se kontrolni znak DEL

Tablica 5 ASCII znakovi

Dec. broj	C konst.	Znak	Dec. broj	Znak
0	'\0'	Nul znak (NULL)	16	znak prekida veze (DLE)
1		početak zaglavlja (SOH)	17	provjera uređaja 1 (DC1)
2		početak teksta (STX)	18	provjera uređaja 2 (DC2)
3		kraj teksta (ETX)	19	provjera uređaja 3 (DC3)
4		kraj prijenosa (EOT)	20	provjera uređaja 4 (DC4)
5		kraj upita (ENQ)	21	negativna potvrda (NAK)
6		Potvrda (ACK)	22	sinkrono mirovanje (SYN)
7	'\a'	Alarm (BEL)	23	kraj prijenosnog bloka (ETB)
8	'\b'	Backspace (BS)	24	otkaži (CAN)
9	'\t'	vodoravni tabulator (HT)	25	kraj medija (EM)
10	'\n'	sljedeći red/novi red (LF)	26	Zamjena (SUB)
11	'\v'	okomiti tabulator (VT)	27	Escape (ESC)
12	'\f'	nova stranica (FF)	28	razdjelnik datoteka (FS)
13	'\r'	skok na početak reda (CR)	29	razdjelnik grupe (GS)
14		pomak van (SO)	30	razdjelnik zapisa (RS)
15		pomak unutra (SI)	31	razdjelnik jedinice (US)

Dec. broj	Znak	Dec. broj	Znak	Dec. broj	Znak	Dec. broj	Znak
32	razmak	48	0	65	A	80	P
33	!	49	1	66	B	81	Q
34	"	50	2	67	C	82	R
35	#	51	3	68	D	83	S
36	\$	52	4	69	E	84	T
37	%	53	5	70	F	85	U
38	&	54	6	71	G	86	V
39	'	55	7	72	H	87	w
40	(	56	8	73	I	88	X
41	)	57	9	74	J	89	Y
42	*	58	:	75	K	90	Z
43	+	59	;	76	L	91	[
44	,	60	<	77	M	92	\
45	-	61	=	78	N	93	]
46	.	62	>	79	O	94	^
47	/	63	?			95	_
		64	@				

Dec. broj	Znak	Dec. broj	Znak
96	`	112	p
97	a	113	q
98	b	114	r
99	c	115	s
100	d	116	t
101	e	117	u
102	f	118	v
103	g	119	w
104	h	120	x
105	i	121	y
106	j	122	z
107	k	123	{
108	l	124	
109	m	125	}
110	n	126	~
111	o	127	DEL

Kako bi se omogućio zapis dodatnih znakova, uveden je prošireni ASCII kôd, koji koristi 8 bitova, pa omogućuje prikaz 256 različitih znakova. Na taj je način omogućeno zapisivanje slova specifičnih za pojedine europske jezike, pa tako i hrvatska slova č, ć, ž, š i đ.

Za prikaz hrvatskih znakova u praksi se najčešće koriste dva standarda:

- Central European (Windows-1250)
- Central European (ISO 8859-2)

Međutim, ni prošireni ASCII kod nije dovoljan za prikaz svih znakova svih svjetskih jezika. Zbog toga se danas koristi UNICODE standard. U UNICODE-u se svaki znak zapisuje pomoću 16 binarnih znamenki (ukupno 65 536 različitih znakova (2^{16})), čime se omogućuje prikaz velikog broja znakova iz različitih jezika i pisama. Bez obzira o kojem se jeziku radi, isti znak uvijek ima isti kôd, što uklanja probleme nekompatibilnosti između različitih sustava.

Važno je naglasiti da je prvih 128 UNICODE znakova u potpunosti identično znakovima standardnog ASCII koda, čime je osigurana kompatibilnost sa starijim sustavima.

Pitanja za ponavljanje

1. Zašto se u računalima koristi binarni brojevni sustav?
2. Što su bit, bajt i riječ?
3. Zašto se heksadekadski sustav često koristi u računalstvu?
4. Kako se pretvara broj iz binarnog u dekadski sustav?
5. Kako se pretvara broj iz dekadskog u binarni sustav?
6. Kako se pretvara broj iz binarnog u oktalni sustav?
7. Kako se pretvara broj iz binarnog u heksadekadski sustav?
8. Što je 1-komplement i kako se dobije?
9. Što je 2-komplement i zašto se koristi u modernim računalima?
10. Koje su prednosti metode 2-komplementa?
11. Na čemu se temelji binarno zbrajanje?
12. Kako se u računalima najčešće izvodi binarno oduzimanje?
13. Što je kodiranje?
14. Koja je razlika između ASCII i UNICODE standarda?

7 Performanse i mjerenja

Performanse računala opisuju koliko brzo i učinkovito računalni sustav može izvršavati zadane programe i obraditi podatke. U kontekstu računalne arhitekture performanse se ne mogu opisati jednom jedinom mjerom, već se analiziraju kroz više međusobno povezanih pojmova. Među najvažnijima su takt, latencija i protočnost koji zajedno omogućuju razumijevanje ponašanja procesora i cjelokupnog sustava.

7.1 Takt procesora

Takt procesora (engl. *clock frequency*) predstavlja osnovnu brzinu rada procesora i izražava se u hercima (Hz). Jedan takt odgovara jednom periodičnom impulsu internog sata procesora kojim se sinkronizira rad njegovih sklopova.

Ako je frekvencija procesora, primjerice 3 GHz, to znači da se u jednoj sekundi generiraju 3 milijarde taktnih impulsa. Trajanje jednog takta tada iznosi približno 0,33 ns. Tijekom jednog takta procesor može započeti ili završiti dio operacije, ovisno o arhitekturi i vrsti instrukcije.

Važno je naglasiti da veća frekvencija takta ne jamči nužno veće performanse jer:

- jedna instrukcija često zahtijeva više taktova,
- različite instrukcije imaju različitu složenost i trajanje,
- pristupi memoriji mogu značajno usporiti izvršavanje.

Zbog toga se takt promatra kao potencijalna brzina rada, ali ne i kao pouzdana mjera stvarnih performansi sustava.

7.2 Latencija

Latencija (engl. *latency*) označava vrijeme potrebno da se jedan zadatak izvrši od početka do kraja. U računalnim sustavima latencija se može odnositi na:

- vrijeme izvršavanja jedne instrukcije,
- vrijeme izvršavanja funkcije ili procesa,
- vrijeme potrebno za obradu jednog zahtjeva (npr. mrežnog, diskovnog ili korisničkog).

Latencija se najčešće izražava u sekundama ili milisekundama po zadatku (*seconds per task*). U kontekstu procesora, latencija je usko povezana s brojem taktova potrebnih za izvršavanje instrukcije, što se često opisuje pomoću pojma **CPI** (*cycles per instruction*).

Primjer: Ako instrukcija zahtijeva 5 taktova, a trajanje jednog takta iznosi 0,33 ns (≈ 3 GHz), tada je latencija te instrukcije približno 1,65 ns.

Latencija je posebno važna u sustavima kod kojih je brzina odziva presudna, kao što su interaktivne aplikacije, sustavi u stvarnom vremenu ili mrežni servisi i baze podataka.

7.3 Protočnost

Protočnost (engl. *throughput*) označava broj zadataka koje sustav može obraditi u jedinici vremena. Za razliku od latencije, koja mjeri vrijeme potrebno za jedan zadatak, protočnost mjeri ukupni kapacitet sustava.

Protočnost se najčešće izražava kao:

- zadaci u sekundi (*tasks per second*),
- instrukcije po sekundi,
- transakcije po sekundi.

Primjerice, procesor može imati relativno veliku latenciju za pojedinu instrukciju, ali zahvaljujući paralelizmu, cjevovodnoj obradi (*pipeline*) i višejezgrenoj arhitekturi može istovremeno obrađivati više instrukcija, što rezultira visokom protočnošću.

Protočnost je osobito važna kod poslužiteljskih sustava, paralelnih i višejezgrenih procesora te podatkovnih centara i sustava za obradu velikih količina podataka.

Odnos latencije i protočnosti

Latencija i protočnost su međusobno povezani, ali predstavljaju različite aspekte performansi te su često u kompromisnom odnosu. Sustav može imati:

- nisku latenciju, ali ograničenu protočnost (brz odgovor, mali kapacitet),
- visoku protočnost, ali veću latenciju (sporiji pojedinačni zadaci, ali velik ukupni učinak).

Klasičan primjer je cjevovodni procesor, kod kojeg latencija jedne instrukcije može biti veća, ali se u svakom taktu završava nova instrukcija čime se značajno povećava ukupna protočnost sustava. Zbog toga se u modernoj arhitekturi procesora naglasak često stavlja na povećanje protočnosti, dok se latencija posebno optimizira samo za kritične dijelove programa.

Izvršno vrijeme kao temeljna mjera performansi

Unatoč postojanju brojnih metrika, u računalnoj arhitekturi vrijedi temeljno pravilo:

Izvršno vrijeme programa jedina je potpuno pouzdana mjera performansi.

Izvršno vrijeme obuhvaća sve ključne utjecaje kao što su arhitektura procesora, frekvencija takta, broj i vrsta instrukcija, organizacija memorijskog sustava te učinkovitost prevoditelja (kompajlera).

Zbog toga se takt, latencija i protočnost koriste kao analitički alati za razumijevanje ponašanja sustava, dok se stvarna usporedba performansi temelji na izmjerenom vremenu izvršavanja stvarnih programa ili standardiziranih testova (*benchmarka*).

7.4 *Benchmarking* i usporedba sustava

Kako bi se performanse različitih računalnih sustava mogle objektivno uspoređivati, nužno je koristiti standardizirane metode mjerenja. Jednostavne usporedbe poput frekvencije procesora ili broja jezgri često su nedostatne te mogu dovesti do pogrešnih zaključaka. Razlog tome je što stvarne performanse ovise o mnogim čimbenicima, uključujući arhitekturu procesora, memorijski sustav, prevoditelje i vrstu radnog opterećenja. Zbog toga se u praksi koriste *benchmark* testovi, odnosno unaprijed definirani skupovi programa i radnih opterećenja koji služe za mjerenje i usporedbu performansi računalnih sustava.

Benchmarking je postupak mjerenja performansi računalnog sustava izvođenjem unaprijed definiranih programa ili testova te analizom dobivenih rezultata. Temeljni ciljevi *benchmarkinga* su osigurati ponovljivost mjerenja, omogućiti usporedivost različitih sustava te procijeniti ponašanje sustava u stvarnim ili reprezentativnim uvjetima rada. *Benchmark* testovi nastoje oponašati tipične radne zadatke za koje je sustav namijenjen. Važno je naglasiti da nijedan *benchmark* ne može u potpunosti obuhvatiti sve moguće primjene jer su pojedini procesori i sustavi optimizirani za određene klase zadataka.

Benchmark testovi mogu se podijeliti u nekoliko osnovnih skupina: reprezentativni programi, kernelški *benchmark* testovi, sintetički *benchmark* testovi i jednostavni programi.

Reprezentativni programi

Ova skupina obuhvaća stvarne aplikacije ili njihove dijelove koji predstavljaju tipične korisničke zadatke. Primjeri uključuju:

- kompajlere,
- alate za kompresiju podataka,
- znanstvene i inženjerske simulacije,
- multimedijske aplikacije.

Najpoznatiji primjer su **SPEC CPU *benchmark* testovi** koji se koriste za mjerenje performansi procesora u realističnim i standardiziranim radnim uvjetima.

Kernelški *benchmark* testovi

Kernelški *benchmark* testovi mjere performanse specifičnih dijelova programa, najčešće jezgrenih numeričkih operacija, kao što su numeričke petlje, matrične operacije i algoritmi linearne algebre.

Primjer takvog testa je **Linpack** koji se često koristi za mjerenje performansi superračunala, osobito u operacijama s pomičnim zarezom.

Sintetički *benchmark* testovi

Sintetički *benchmark* testovi ne predstavljaju stvarne programe, već umjetno generiraju mješavinu instrukcija s ciljem opterećivanja određenih dijelova sustava. Tipični primjeri su:

- **Dhrystone** (cjelobrojne operacije),
- **Whetstone** (operacije s pomičnim zarezom).

Iako su jednostavni i brzi za izvođenje, sintetički *benchmark* testovi često ne odražavaju realne uvjete rada te ih treba tumačiti s oprezom.

Jednostavni (*toy*) programi

Jednostavni programi, poput sortiranja nizova ili traženja prostih brojeva, koriste se prvenstveno u edukacijske svrhe. Oni nisu prikladni za ozbiljne usporedbe sustava, ali su korisni za ilustraciju osnovnih pojmova i utjecaja pojedinih arhitekturnih odluka.

Mjerne veličine u *benchmarkingu*

U postupku *benchmarkinga* koriste se različite mjerne veličine, od kojih su najčešće vrijeme izvršavanja i propusnost.

Vrijeme izvršavanja (eng. *execution time*) je vrijeme potrebno da se *benchmark* program u potpunosti izvrši. Ovo je najpouzdanija mjera performansi jer izravno pokazuje koliko brzo sustav rješava zadani problem.

Propusnost (eng. *throughput*) je broj zadataka koji se mogu izvršiti u jedinici vremena. Ova se mjera često koristi kod poslužiteljskih, paralelnih i višeprocorskih sustava.

CPI (*Cycles Per Instruction*) označava prosječan broj taktova potreban za izvršavanje jedne instrukcije.

IPC (*Instructions Per Cycle*) označava prosječan broj instrukcija izvršenih u jednom taktu.

Ove mjere korisne su za analizu arhitekture procesora, ali same po sebi nisu dovoljne za izravnu usporedbu različitih sustava.

MIPS (*Millions of Instructions Per Second*)

MFLOPS (*Millions of Floating-Point Operations Per Second*)

Iako su ove mjere povijesno značajne, mogu biti zavaravajuće jer ovise o skupu instrukcija, vrsti programa te načinu kompajliranja.

Zbog toga se danas koriste uglavnom u ograničenim kontekstima ili za usporedbe unutar iste arhitekture.

Normalizacija i prosjeci performansi

Kod usporedbe više *benchmark* testova često se koristi normalizacija rezultata u odnosu na referentni sustav. Time se dobivaju relativni omjeri performansi koji omogućuju jasniju i pregledniju usporedbu.

Za objedinjavanje više rezultata koriste se različite vrste srednjih vrijednosti:

- aritmetička sredina (najčešće za vremena izvršavanja),
- geometrijska sredina (za normalizirane performanse),
- harmonijska sredina (za brzine i stope izvršavanja).

Nepravilna upotreba prosjeka može dovesti do pogrešnih zaključaka, zbog čega se u računalnoj arhitekturi posebno naglašava pravilna metodologija mjerenja.

Ograničenja *benchmark* testova

Iako su *benchmark* testovi neizostavan alat za procjenu performansi, oni imaju i svoja ograničenja:

- ne mogu obuhvatiti sve moguće stvarne primjene,
- rezultati ovise o operacijskom sustavu, kompajleru i postavkama sustava,
- proizvođači mogu optimizirati hardver i softver za određene *benchmark* testove.

Zbog toga se rezultati *benchmarkinga* uvijek moraju tumačiti u kontekstu konkretne namjene sustava, a ne kao apsolutna mjera njegove kvalitete.

Benchmarking predstavlja temeljni alat za usporedbu performansi računalnih sustava. Korištenjem standardiziranih testova moguće je objektivno procijeniti sposobnosti procesora i cijelih računalnih sustava, no dobivene rezultate treba analizirati pažljivo i uz razumijevanje njihovih ograničenja. U konačnici, najvažniji kriterij performansi ostaje vrijeme izvršavanja stvarnih aplikacija koje korisnik u praksi zaista koristi.

Pitanja za ponavljanje

1. Što opisuju performanse računala?
2. Što je takt procesora i u čemu se izražava?
3. Koja je razlika između latencije i protočnosti?
4. Kako paralelizam i cjevovodna obrada utječu na protočnost?
5. Zašto se izvršno vrijeme programa smatra najpouzdanijom mjerom performansi?
6. Što je *benchmarking*?

7. Koji su temeljni ciljevi *benchmarkinga*?
8. Koje su osnovne skupine *benchmark* testova?
9. Što su reprezentativni *benchmark* programi?
10. Što su sintetički *benchmark* testovi i zašto ih treba tumačiti s oprezom?
11. Zašto mjere poput MIPS i MFLOPS mogu biti zavaravajuće?
12. Koji su glavna ograničenja *benchmark* testova?

8 Suvremeni trendovi u računalstvu

Razvoj računalnih sustava posljednjih desetljeća obilježen je snažnim rastom računalne snage, ali i promjenama u arhitekturi, energetske učinkovitosti i načinu korištenja računalnih resursa. U ovom poglavlju daje se pregled odabranih suvremenih trendova u računalstvu, s naglaskom na superračunala, kvantna računala i nove procesorske arhitekture.

8.1 Superračunala

Superračunala su specijalizirani računalni sustavi namijenjeni izvođenju izuzetno zahtjevnih računalnih zadataka, kao što su klimatsko i meteorološko modeliranje, numeričke simulacije u fizici i kemiji, biomedicinska istraživanja te umjetna inteligencija i strojno učenje.

Povijest superračunala započinje 1960-ih godina, kada je Seymour Cray u tvrtki Control Data Corporation razvio prva komercijalna superračunala. Kasnije osniva Cray Research koja desetljećima postavlja standarde u području visokih performansi (HPC – *High Performance Computing*).

Za razliku od osobnih računala, superračunala se sastoje od tisuća ili milijuna procesorskih jezgri povezanih u klastere te koriste masivni paralelizam. Njihove se performanse mjere u FLOPS-ima (*Floating Point Operations Per Second*) jer su operacije s pomičnim zarezom osnova znanstvenih izračuna.

Primjeri suvremenih superračunala

Fugaku

Superračunalo Fugaku razvila je tvrtka Fujitsu, a u rad je pušteno 2021. godine. Temelji se na ARM procesorima A64FX te je jedno vrijeme bilo najbrže superračunalo na svijetu. Posebno se istaknulo primjenama u epidemiološkim simulacijama tijekom pandemije COVID-19.

Summit

Summit je superračunalo razvijeno u suradnji IBM-a i NVIDIA-e, s naglaskom na umjetnu inteligenciju i hibridnu CPU–GPU arhitekturu. Kombinira POWER9 procesore i NVIDIA GPU akceleratora čime ostvaruje iznimno visoku protočnost za AI i znanstvene primjene.

Frontier

Frontier je prvo *exascale* superračunalo na svijetu, sposobno izvoditi više od 10^{18} operacija u sekundi. Razvila ga je tvrtka Hewlett Packard Enterprise u suradnji s AMD-om. Frontier predstavlja prijelaz prema energetski učinkovitijim superračunalima s heterogenim arhitekturama.

8.2 Kvantna računala

Kvantna računala temelje se na zakonima kvantne mehanike, koji se bitno razlikuju od klasične fizike. Za razliku od klasičnog bita, koji može imati vrijednost 0 ili 1, kvantni bit (*qubit*) može se nalaziti u superpoziciji stanja.

Ideja kvantnog računala pojavila se početkom 1980-ih, a dodatno ju je razvio Richard Feynman koji je istaknuo da klasična računala nisu učinkovita za simulaciju kvantnih sustava.

Kvantna računala nisu općenito brža od klasičnih računala, ali za određene probleme omogućuju eksponencijalno smanjenje broja potrebnih operacija.

Zbog izuzetne osjetljivosti *qubita* na vanjske utjecaje, kvantni sustavi rade na temperaturama bliskima apsolutnoj nuli, što je dovelo do razvoja kvantnih računala kao usluge u oblaku.

Usporedba klasičnog i kvantnog bita

Klasični i kvantni bit dijele činjenicu da mjerenjem daju jednu od dvije vrijednosti, no razlike su temeljne:

- klasični bit ima jedno definirano stanje,
- *qubit* prije mjerenja može biti u superpoziciji,
- stanje *qubita* ovisi o načinu mjerenja,
- mjerenje *qubita* nepovratno mijenja njegovo stanje,
- s povećanjem broja *qubita* raste eksponencijalna računska moć sustava.

Primjene kvantnih računala uključuju:

- simulaciju kvantnih sustava,
- modeliranje kemijskih reakcija,
- razvoj novih materijala i lijekova,
- kriptografiju i optimizacijske probleme.

8.3 Energetski učinkoviti procesori i heterogene arhitekture

Suvremeni trendovi u računalstvu snažno su usmjereni prema energetskej učinkovitosti i heterogenim arhitekturama koje kombiniraju:

- CPU jezgre,
- GPU akceleratori,
- NPU (AI akceleratori),

- FPGA sklopove.

Ovakvi sustavi omogućuju izvršavanje specijaliziranih zadataka uz znatno manju potrošnju energije, što je izrazito bitno za podatkovne centre, mobilne uređaje i održivo računarstvo.

Suvremeni trendovi u računalstvu obilježeni su pomakom od jednostavnog povećanja frekvencije prema paralelizmu, heterogenim arhitekturama i energetskej učinkovitosti. Razvoj superračunala i kvantnih računala pokazuje da se budućnost računalstva temelji na inovativnim arhitekturnim rješenjima i pametnom upravljanju resursima.

8.4 Post-silicijski razvoj i specijalizirane arhitekture

Razvoj računalnih sustava u novije vrijeme sve se više udaljava od isključivog oslanjanja na klasičnu silicijsku tehnologiju. Iako su integrirani krugovi i dalje temelj suvremenih računala, fizička ograničenja poput zagrijavanja, potrošnje energije i daljnjeg smanjivanja dimenzija tranzistora potiču razvoj novih pristupa u oblikovanju računalnih sustava. U tom kontekstu pojavljuju se arhitekture i tehnologije koje ne nastoje samo povećati performanse postojećih sustava, već uvode i drugačije načine obrade podataka.

Jedan od takvih pristupa je neuromorfno računalstvo (eng. *neuromorphic computing*) koje se temelji na oponašanju načina rada ljudskog mozga. Umjesto strogo sekvencijalnog izvršavanja instrukcija, obrada se odvija kroz mrežu umjetnih neurona i sinapsi. Takav pristup posebno je pogodan za zadatke prepoznavanja uzoraka, učenja i obrade senzorskih podataka, uz znatno manju potrošnju energije u odnosu na klasične arhitekture.

Razvoj optičkog, odnosno fotoničkog računalstva (eng. *photonic computing*) temelji se na uporabi svjetlosnih signala za prijenos i obradu podataka. Za razliku od električnih signala, svjetlost omogućuje vrlo velike brzine prijenosa uz manju disipaciju topline. Zbog toga se takva tehnologija razmatra kao rješenje za sustave koji zahtijevaju izuzetno veliku protočnost, osobito u podatkovnim centrima i sustavima za obradu velikih količina podataka.

U suvremenim računalnim sustavima značajan problem predstavlja prijenos podataka između procesora i memorije. Zbog toga se razvija koncept računalstva u memoriji (eng. *in-memory computing*), pri kojem se dio obrade podataka odvija unutar same memorije. Primjena novih elemenata, poput memristora, omogućuje smanjenje potrebe za prijenosom podataka, čime se povećava brzina rada i smanjuje potrošnja energije.

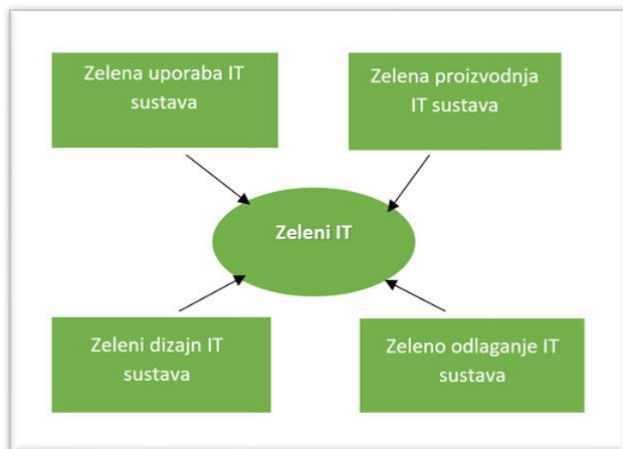
Daljnji smjer razvoja predstavlja nanoračunalstvo (eng. *nanocomputing*) koje se temelji na uporabi materijala i struktura na razini nanometara. Posebno se ističe primjena ugljičnih nanocijevi koje omogućuju izradu tranzistora vrlo malih dimenzija i potencijalno predstavljaju alternativu klasičnim silicijskim sklopovima.

Istodobno se razvija i računalstvo na rubu (eng. *edge computing*), pri kojem se obrada podataka premješta bliže izvoru njihova nastanka. Takav pristup smanjuje latenciju i potrebu za prijenosom podataka prema udaljenim poslužiteljima. U tim sustavima često se koriste specijalizirani akceleratori za umjetnu inteligenciju koji omogućuju obradu podataka izravno na uređajima kao što su senzori, mobilni uređaji i ugrađeni sustavi.

Navedeni pristupi pokazuju da se razvoj računalstva sve više usmjerava prema specijaliziranim i energetski učinkovitim rješenjima. Umjesto univerzalnih arhitektura koje pokušavaju učinkovito rješavati sve vrste zadataka, naglasak se stavlja na prilagodbu arhitekture konkretnim problemima i načinu obrade podataka.

8.5 Green IT – održivost u računalnim sustavima

Razvoj računalnih sustava tijekom posljednjih desetljeća bio je usmjeren prvenstveno na povećanje performansi, kapaciteta pohrane i brzine obrade podataka, pri čemu suvremeni arhitekturni i tehnološki pristupi dodatno naglašavaju važnost energetske učinkovitosti i održivosti. Međutim, paralelno s tehnološkim napretkom raste i svijest o ekološkim posljedicama masovne digitalizacije. Računalni sustavi troše značajne količine električne energije, generiraju toplinu koja zahtijeva dodatne sustave hlađenja te proizvode sve veće količine elektroničkog otpada. U tom kontekstu razvija se koncept Green IT-a, odnosno zelene informacijske tehnologije, koji integrira načela održivosti u projektiranje, implementaciju i korištenje računalnih sustava.



Slika 31 Holistički pristup Green IT-u (Bilal Zorić, Kalamir, 2024)

Green IT obuhvaća projektiranje, proizvodnju, uporabu i zbrinjavanje informatičke opreme na način koji minimizira negativan utjecaj na okoliš kroz cijeli životni ciklus proizvoda. Takav pristup podrazumijeva smanjenje potrošnje energije, optimizaciju korištenja resursa, produženje životnog vijeka opreme i odgovorno upravljanje elektroničkim otpadom. U kontekstu građe računala, to znači da se održivost ne promatra kao dodatna značajka sustava, nego kao sastavni dio arhitekture procesora, memorijskih hijerarhija, sustava napajanja i organizacije podatkovnih centara.

Posebno izražen izazov predstavlja potrošnja energije podatkovnih centara koji čine temelj suvremene digitalne infrastrukture. Povećanje korištenja usluga u oblaku, *streaming* servisa i sustava temeljenih na umjetnoj inteligenciji dovodi do kontinuiranog rasta računalnih opterećenja. Istraživanja pokazuju da optimizacija infrastrukture može značajno smanjiti ukupnu potrošnju energije, primjerice kroz virtualizaciju servera, konsolidaciju radnih opterećenja i primjenu naprednih sustava hlađenja.

Virtualizacija omogućuje bolje iskorištavanje fizičkog hardvera, čime se smanjuje broj potrebnih servera i ukupna energetska potrošnja sustava. Ujedno se razvijaju sofisticirani sustavi hlađenja, tekućinsko hlađenje i iskorištavanje prirodnih klimatskih uvjeta, čime se dodatno povećava učinkovitost infrastrukture.

Održivi pristup računalnim sustavima obuhvaća i primjenu načela kružnog gospodarstva u upravljanju hardverom. Umjesto linearnog modela „proizvedi – koristi – odbaci“, potiče se produženje životnog vijeka opreme kroz modularni dizajn, mogućnost nadogradnje pojedinih komponenti te sustavnu reciklažu. Zamjena memorijskog modula, SSD diska ili grafičke kartice bez odbacivanja cijelog uređaja smanjuje količinu elektroničkog otpada i potrebu za novom proizvodnjom. Takav pristup ima izravnu poveznicu s konstrukcijom računalnih sustava jer potiče standardizaciju, kompatibilnost i dizajn usmjeren na dugoročnu uporabu.

Green IT ne odnosi se isključivo na hardver, već i na razvoj softvera. Održivi softverski inženjering podrazumijeva pisanje učinkovitog koda koji zahtijeva manje procesorskog vremena i manje memorijskih pristupa. Optimizirani algoritmi, bolja lokalnost podataka u *cache* memoriji i racionalna uporaba paralelizma rezultiraju kraćim vremenom izvođenja i manjom potrošnjom energije. Time se performanse i održivost povezuju kao dvije međusobno komplementarne, a ne suprotstavljene dimenzije.

Dodatnu dimenziju optimizacije donosi primjena umjetne inteligencije u upravljanju infrastrukturom. Algoritmi strojnog učenja koriste se za predviđanje opterećenja sustava, dinamičko upravljanje potrošnjom energije i optimizaciju sustava hlađenja. Studije pokazuju da sustavi vođeni umjetnom inteligencijom mogu značajno smanjiti potrošnju energije podatkovnih centara prilagodbom rada sustava u stvarnom vremenu.

Green IT stoga predstavlja evoluciju pristupa računalnim sustavima u kojem se performanse, pouzdanost i skalabilnost promatraju zajedno s održivošću. Integracijom ekoloških načela u dizajn i uporabu računalne opreme moguće je istodobno postići visoku razinu tehnološke učinkovitosti i smanjiti negativan utjecaj na okoliš. U tom smislu, održivost postaje sastavni dio suvremenog razumijevanja računalne arhitekture i jedan od važnih smjerova budućeg razvoja informacijskih tehnologija.

Pitanja za ponavljanje

1. Što su i za koje se vrste problema najčešće koriste superračunala?
2. Što znači pojam masivnog paralelizma u superračunalima?

3. Što je kvantno računalo?
4. Što je *qubit* i po čemu se razlikuje od klasičnog bita?
5. Što je superpozicija u kvantnom računarstvu?
6. Za koje vrste problema kvantna računala mogu biti posebno korisna?
7. Zašto je rad kvantnih računala tehnološki vrlo zahtjevan?
8. Objasnite što se podrazumijeva pod post-silicijskim razvojem računalnih sustava i navedite razloge njegova nastanka.
9. Opišite osnovnu ideju nanoračunalstva i navedite primjer tehnologije koja se u tom području koristi.
10. Što je Green IT?
11. Koji su glavni ciljevi koncepta Green IT?
12. Kako virtualizacija doprinosi smanjenju potrošnje energije u podatkovnim centrima?
13. Kako modularni dizajn računalne opreme doprinosi održivosti?
14. Kako optimizirani softver može doprinijeti energetskej učinkovitosti računalnih sustava?

9 Literatura

1. Bashe, C. J., Johnson, L. R., Palmer, J. H., & Pugh, E. W. (1986). *IBM's Early Computers*. MIT Press.
2. Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5), 755–768.
3. Bianchini, R., & Rajamony, R. (2004). Power and energy management for server systems. *Computer*, 37(11), 68–74.
4. Bilal Zorić, A., & Kalamir, M. (2024). Ususret održivoj budućnosti: ključni trendovi zelenog IT-a. *ET²eR-ekonomija, turizam, telekomunikacije i računarstvo*, 6(2), 72-80.
5. Brock, D. C. (2006). *Understanding Moore's Law: Four Decades of Innovation*. Chemical Heritage Press.
6. Brock, D. C. (2010). *The Microprocessor: A Biography*. MIT Press.
7. Brooks, F. P. (2010). *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley.
8. Campbell-Kelly, M., Aspray, W., Yost, J. R., Tinn, H., & Díaz, G. C. (2023). *Computer: A History of the Information Machine* (3rd ed.). Routledge.
9. Ceruzzi, P. E. (2003). *A History of Modern Computing* (2nd ed.). MIT Press.
10. Copeland, B. J. (2002). *Artificial Intelligence: A Philosophical Introduction*. Wiley-Blackwell.
11. Copeland, B. J. (2006). *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*. Oxford University Press.
12. Copeland, B. J. (2017). *Turing: Pioneer of the Information Age*. Oxford University Press.
13. Cortada, J. W. (1993). *Before the Computer: IBM, NCR, Burroughs, and Remington Rand and the Industry They Created, 1865–1956*. Princeton University Press.
14. Devlin, K. (2008). *The Language of Mathematics: Making the Invisible Visible*. W.H. Freeman.
15. Feigenbaum, E., & Shrobe, H. (1993). The Japanese national Fifth Generation project: Introduction, survey, and evaluation. *Future Generation Computer Systems*, 9(2), 105–117.
16. Freeth, T., Bitsakis, Y., Moussas, X., et al. (2006). Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism. *Nature*, 444, 587–591.
17. Gleick, J. (2011). *The Information: A History, a Theory, a Flood*. Pantheon.
18. Grier, D. A. (2005). *When Computers Were Human*. Princeton University Press.
19. Hennessy, J. L., & Patterson, D. A. (2017). *Computer Organization and Design: The Hardware/Software Interface* (5th ed.). Morgan Kaufmann.
20. Hennessy, J. L., & Patterson, D. A. (2021). *Computer Architecture: A Quantitative Approach* (6th ed.). Morgan Kaufmann.
21. Hills, G., Lau, C., Wright, A., Fuller, S., Bishop, M. D., Srimani, T., ... & Shulaker, M. M. (2019). Modern microprocessor built from complementary carbon nanotube transistors. *Nature*, 572(7771), 595–602.

22. IEEE Computer Society. (2019). *IEEE Standard for Floating-Point Arithmetic (IEEE 754-2019)*. IEEE.
23. Ifrah, G. (2001). *The Universal History of Computing: From the Abacus to the Quantum Computer*. Wiley.
24. Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., & Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, 97, 219–235.
25. Koomey, J. G. (2011). Growth in data center electricity use 2005 to 2010. *Analytics Press*.
26. Lin, S., Ning, S., Zhu, H., Zhou, T., Morris, C. L., Clayton, S., ... & Wang, Z. (2024). Neural network methods for radiation detectors and imaging. *Frontiers in Physics*, 12, 1334298.
27. Malone, M. S. (1995). *The Intel Trinity*. HarperCollins.
28. Masanet, E., Shehabi, A., Lei, N., Smith, S., & Koomey, J. (2020). Recalibrating global data center energy-use estimates. *Science*, 367(6481), 984–986.
29. Mooreov zakon. (2007). *Tehnički leksikon*. Leksikografski zavod Miroslav Krleža. <https://tehnicki.lzmk.hr/clanak/mooreov-zakon>
30. Murugesan, S. (2008). Harnessing Green IT: Principles and practices. *IT Professional*, 10(1), 24–33.
31. Nilsson, N. J. (2010). *The Quest for Artificial Intelligence*. Cambridge University Press.
32. Norvig, P., & Russell, S. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
33. Patterson, D. A., & Hennessy, J. L. (2017). *Computer Organization and Design RISC-V Edition*. Morgan Kaufmann.
34. Ribarić, S. (2011). *Građa računala, arhitektura i organizacija računarskih sustava*.
35. Riordan, M., & Hoddeson, L. (1997). *Crystal Fire: The Birth of the Information Age*. W.W. Norton.
36. Rojas, R., & Hashagen, U. (2002). *The First Computers: History and Architectures*. MIT Press.
37. Shastri, B. J., Tait, A. N., Ferreira de Lima, T., Pernice, W. H., Bhaskaran, H., Wright, C. D., & Prucnal, P. R. (2021). Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics*, 15(2), 102–114.
38. Shehabi, A., Smith, S. J., Sartor, D., et al. (2016). United States data center energy usage report. Lawrence Berkeley National Laboratory.
39. Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5), 1484–1509.
40. SPEC. (2023). *SPEC CPU Benchmark Documentation*. Standard Performance Evaluation Corporation. <https://www.spec.org>
41. Stallings, W. (2015). *Computer Organization and Architecture* (10th ed.). Pearson.
42. Struik, D. J. (1987). *A Concise History of Mathematics*. Dover Publications.
43. Swade, D. (2001). *The Difference Engine*. Penguin.
44. Toole, B. A. (1998). *Ada, the Enchantress of Numbers*. Strand Publishing.
45. Top500.org. (2024). *TOP 500 Supercomputer Sites*. <https://www.top500.org>

46. Van Heddeghem, W., Lambert, S., Lannoo, B., et al. (2014). Trends in worldwide ICT electricity consumption from 2007 to 2012. *Computer Communications*, 50, 64–76.
47. Wei Wang. (n.d.). *Performance Metrics and Measurement*.
48. Zuse, K. (2007). *The Computer – My Life*. Springer.