1. Narava računanja in strojev za računanje

Računanje je določanje vrednosti neki funkciji f(x), kjer je x vhodni podatek, z (z=f(x)) pa izhodi podatek. Funkcija je **izračunljiva**, če obstaja postopek, s katerim lahko določimo njeno vrednost za vse vhodne podatke x. **Algoritem** je definiran kot navodilo, ki v končnem številu korakov privede do želenega rezultata. Funkcija je lahko tudi **neizračunljiva**.

Church-Turingorva hipoteza: Nek problem je izračunljiv, če ga je v končnem številu korakov možno izračunati na nekem Turingovem stroju.

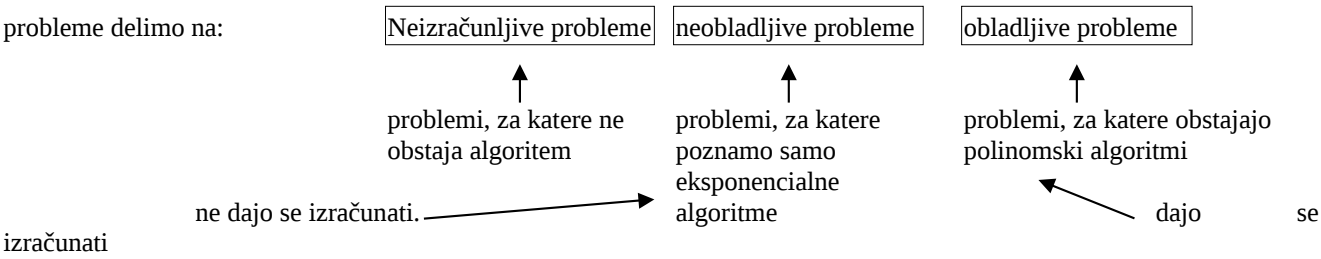
Turingov stroj je model računanja za matematično dokazovanje. Opišemo ga z neskončnim trakom (v obe smeri), ki je razdeljen na celice v katerih se nahajajo po ena izmed črk abecede. Bralno-pisalna glava se premika po tem traku in bere/piše iz/v celice pod kontrolo procesorja, ki je v enem izmed notranjih stanj. Kateri od ukazov se bo izvršil v nekem času je odvisno od notranjega stanja in od črke na traku. (*Knjiga STR 8*).

Deterministični Turingov stroj je nedvoumen turingov stroj, Nedeterministični Turingov stroj pa je ena izmed izboljšav Turingovih strojev pri katerem lahko stroj pri vsakem ukazu izbira med več operacijami in prehodi v nova notranja stanja.

Današnji računalniki so večinoma zgrajeni na **Von Neumannovem modelu računanja** – ideja o shranjenem programu.

Neobladljivi problemi: Čeprav za neko funkcijo lahko dokažemo, da je izračunljiva se lahko zgodi, da je ne morem izračunati zaradi prostorske in časovne kompleksnosti – prostor in čas sta pri današnjih računalnikih omejena, pri teoriji Turingovega stroja pa je trak neskončen. Teorija kompleksnosti je posebno področje matematike, ki se ukvarja s kompleksnostjo algoritmov.

Cookov izrek: Tisti NP problemi (Nedeterministično polinomski problemi), za katere lahko dokažemo tudi obratno polinomsko prevedbo so NP-polni. Za NP polne probleme je dokazan obstoj algoritma na deterministočnem Turingovem stroju.



2. Dosedanji razvoj strojev za računanje

Najstarejši stroj za opravljanje osnovnih aritmetičnih operacij je razvil Wilhelm Schickard leta 1623.Leta 1642 pa je podobe stroj naredil Blaise Pascal. Oba stroja sta za predstavitev desetiških števil uporabljala zobata kolesa z desetimi zobci in mehanizmom za prenos enote na naslednje kolo. Ista ideja se še danes uporablja pri števcih kilometrov v avtomobilih.
Pascalov stroj je imel dve skupimi s šestimi kolesci – dva šestmestna registra. Prvi je služil kot **akumulator**, drugi pa za vnašanje šestmestnega števila, ki ga je hotel uporabnih odšteti ali prišteti temu v akumulatorju. Izraz akumulator se uporablja zato, ker se rezultat operacije »akumulira« (spravi) v tem registru.

Charles Babbage: Izumil je stroj, ki je podoben današnjim računalnikom. Prva in enostavnejša izvedba njegovega stroja se imenuje **Diferenčni stroj** (difference engine), druga pa **Analitični stroj** (analzthical engine). – Zgradba *STR 27*.

1941 je bil dokončan prvi delujoči programsko vodeni računalnik za splošne namene z imenom **Z3** (avtor: Konrad Zuse). Imel je 2600 telefonskih relejev in pomnilnik velikosti 64 22-bitnih besed. Vsi ukazi so bili 8-bitni in shranjeni na luknjanem traku. Bil je binarni in s plavajočo vejico.
Zgradba *STR 31*.

Harward Mark I je bil narejen **1944**. Oblika ukaza je bila A1 A2 OP (A1 in A2 – naslova, OP – operacija. Rezultat se shrani v A2). Bil je desetiški s fiksno vejico.

ENIAC je bil prvi elektronski računalnik, zgrajen leta **1945**. Bil je desetiški in j euporabljal fiksno vejico. 18000 elektronk, 1500 relejev, 6000 stikal. Tedensko so menjavali po 3 elektronke.

EDVAC (1951) nastane po ENIACu. Prvič se pojavi pojem centralne procesne enote in glavnega pomnilnika. Ima **program shranjen v pomnilniku** (!!!). Oblika ukaza: A1 A2 A3 A4 OP – A1 in A2 sta naslova pomnilniških besed, v A3 gre rezultat, A4 pa je naslov naslednjega ukaza OP.

IAS (1952): Ustvari ga Neumann z sodelavci. Ima **naključni dostop do pomilnika**. Možen je bil tudi dostop do vseh bitov besede naenkrat. Bil je paralelen stroj. Oblika ukaza OP A – A je pomnilniški naslov besede prvega operanda, drugi operand je bil skoraj v vseh ukazih v registru, ki se imenuje AC (akumulator). Imel je tudi pomožni register MQ, ki se je uporabljal pri množenju in deljenju.
Prvič se pojavi pojem **programskega števca** in sicer s pravilom: Če ukaz ne zahteva drugače (skočni ukazi) si ukazi sledijo eden za drugim po naraščujočih naslovih. (PC := PC + 1).

Groschev zakon: zmogljivost = konstanta x cena^2

3 Osnovni principi delovanja

Obstajata Von Neumannov računalniški model in Von Neumannov model računanja. Lahko tudi rečemo, da je nek računalnik narejen v skaldu z Von Neumannovo arhitekturo. Von Neumannov stroj je vsak stroj, ki izpolnjuje naslednje pogoje:

- sestavljajo ga 3 osnovni deli: Centralna procesna enota, glavni pomnilnik, vhodno/izhodni sistem.
- je stroj s shranjenim programom. Program je shranjen v glavnem pomnilniku in vodi delovanje stroja.
- ukazi programa se izvajajo zaporedno eden za drugim (razen pri pogojnih skokih)

Zgradba tipičnega Von Neumannovega računalnika (*STR 50*):

1. **CPE** - Centralna procesna enota. Pogosto rečemo procesor. Danes večinoma mikroprocesor. Skoraj vse dogajanje se odvija v CPE ali pod njegovo kontrolo. Osnovna naloga je, da iz pomnilnika jemlje ukaze in jih izvršuje. Deljen je na 3 enote:
   - kontrolno ento – skrbi za prevzemanje ukazov in operandov ter za aktiviranje ustreznih operacij, ki se izvajajo v aritmetično logični enoti.
   - aritmetično logično enoto
   - registre (programsko dostopne in programsko nedostopne).
2. **Glavni pomnilnik** – imenuje se tudi primarni pomnilnik. V njem so ukazi in operandi. Sestavljen je iz pomnilniških besed. Vsaka od njih ima svoj enoličen naslov.
3. **Vhodno izhodni sistem** – služi za prenos informacije v ali iz zunanjega sveta. Pretvarja informacijo iz oblike, ki je uporabna za CPE v neko obliko, ki je bolj uporabna za človeka ali neko enoto.

Po vklopu deluje CPE tipično po naslednjem zaporedju:

1. Jemanje ukaza iz pomnilnika (**FETCH**) – CPE vzame ukaz (strojni ukaz) iz pomnilnika Zaporedje teh ukazov tvori program. Strojni ukaz se vedno prebere iz tiste besede glavnega pomnilnika na katero kaže programski števec (PC).
2. Izvrševanje v 1. koraku prevzetega ukaza (**EXECUTE**) – CPE izvrši operacijo in poskrbi, da je po končanem izvrševanju v PC naslov naslednjega ukaza. Pri tem upošteva, da so v pomnilniku ukazi razporejeni po naraščajočem vrstem redu.

Ta postopek se ponavlja dokler ne pride do kakšne prekinitve v tem primeru se vklopi servisni program, ki poskrbi, da se prekinitev servisira.

Zaradi pravila PC = PC + 1 imenujemo Von Neumannove računalnike **ukazno pretokovni računalniki**. Poznamo pa tudi **podatkovno pretokovne računalnike** pri katerih vrstni red narekujejo podatki in ne ukazi. Teh je malo in so bolj redki oziroma se pojavljajo le po laboratorijih.

Zaporedno izvajanje ukazov (značilno za Von Neumannove računalnike) je počasno. Obstaja vrsta problemov, ki po svoji naravi dovoljujejo **istočasno** ali **paralelno** izvajanje več operacij.

1966 je **Flynn** podal naslednjo klasifikacijo razširitev Von Neumannovega modela:

- število ukazov, ki se izvajajo naenkrat – tok ukazov
- število operandov, ki jih ukaz obdeluje naenkrat.

Po Flynnovi klasifikaciji delimo računalnike na:

1. **SISD** (Single Instruction stream, Single Data stream) – izvaja se en ukaz na eni sami ponovitvi operandov. Najpogostejša oblika današnjih računalnikov.
2. **SIMD** (Single Instruction stream, Multiple Data stream) – izvaja se en sam ukaz ampak se opravlja naenkrat na več ponovitvah operandov. Računalnik ima več aritmetično logičnih enot in njihovih registrov, še zmeraj pa samo en PC.
3. **MISD** (Multiuple Instruction stream, Single Data stream) – takih računalnikov ni.
4. **MIMD** (Multiple instruction stream, Multiple data stream) – izvaja se več ukazov na več operandih. Ponavadi gre za večprocesorske sisteme.

SIMD in MIMD se označujejo kot **paralelni računalniki**.

Za organizacijo pomnilnika imamo dve arhitekturi: **Harvardsko** in **Princetonsko**. Pomnilniška beseda je najmanjše število bitov s svojim naslovom. Pri pomnilniku je potrebno razlikovati med *naslovom* pomnilniške besede in *vsebino* pomnilniške besede. Dolžina pomnilniške besede in dolžina registrov sta lahko različna. Pomnilniškega dostopa ni moč prekinjati (nedeljivost). Pri večprocesorskih računalnikih je realiziran kot **blrano spreminjevalni pisalni dostop**. V enem dostopu se podatek **prebere**, **spremeni in vpiše nazaj**, ta čase je pomnilnik za ostale zaklenjen – pomembno za sinhronizacijo procesov.

Dolžina naslovnega registra in programskega števca morata biti enaki dolžini naslova. Izbira prekratkega naslovnega registra in PC je lahko usodna za nadalnji razvoj družine procesorjev (Intel 8086).

Dolžina podatkovnega registra določa koliko podatkov se lahko obenem iz pomnilnika prenese v CPE.

* Lastnosti pomnilnika v Von Neumannovem računalniku:

1. **Pomnilnik je enodimenzionalen** – Vsaka od besed ima svoj enoličen naslov. Pomnilnik ima obliko linearnega vektorja.
2. **Ni razlike med ukazi in operandi** – Ukaze lahko obravnavamo kot operande in obratno.
3. **Pomen ni sestavni del operandov** – Iz neke pomnilniške besede ni razvidno kaj je zapisano v njej (ukaz, število, …).

Vhod/Izhod Von Neumannovega računalnika:

- programski vhod in izhod: Z napravo komunicira CPE z zaporedjem ukazov, ki poskrbijo za vse podrobnosti pri prenosu. Je počasen, ker je treba za vsak prenos izvršiti množico ukazov. Je pa poceni.
- direktni dostop do pomnilnika (**DMA**) – z pomnilnikom komunicira poseben kontroler DMA je zelo hiter in razbremeni procesor, ki v tem času lahko dela kaj drugega. Je pa drag.

naslovni prostor:

- pomnilniško preslikan vhod-izhod. Registri krmilnikov so v pomnilniškem naslovmen prostoru in so za CPE isti kot pomnilniške besede. Uporabljajo se lahko vsi ukazi za delo s pomnilnikom.
- ločen vhodno izhodni prostor. Registri krmilnikov so v posebnem naslovnem prostoru. Za dostop do njih so potretni posebni ukazi
- posredno preko vhodno izhodnih procesorjev. CPE sporoča svoje zahteve vhodno izhodnim proceosrjem, ki imajo dostop do registrov vhodno izhodne naprave in ti procesorji poskrbijo za prenos podatkov. Pogosta rešitev pri večjih računalnikih.

lokalnost pomnilniških dostopov (prostorska in časovna) se izkorišča za grandnjo predpomnilnika. Vzrok za časovno lokalnost so zanke in začasne spremenljivke, za prostorsko pa pravilo PC = PC + 1, ter organiziranost programov v procedure, podatkovne strukture kot so polja (dostopi po indeksih).

Amdahlov zakon pravi, da je povečanje hitrosti računalnika omejeno s procentom časa, v katerem ostane hitrost nespremenjena. Iz tega zakona sledi, da nam več procesorjev ne pomaga prav dosti, če lahko določen čas recimo dela samo eden. Sledi tudi, da je pomembno pohitriti ravno tiste operacije, ki so najpogostejše ter da je blje imeti en hitrejši procesor kot pa dva počasnejša.

Case in **Amhdal** sta zasnovala dve pravili po katerih sodimo dobro zgrajen računalnik:

1. Velikost glavnega pomnilnika v bajtih mora biti najmanj enaka številu ukazov, ki jih v eni sekundi izvede CPE.
2. Zmogljivost vhodno izhodnega sistema v bitih na sekundo mora biti najmanj enaka številu ukazov, ki jih v eni sekundi izvede CPE

Smisel obeh pravil je v tem, da podajata pogoje pri katerih je računalnik *uravnotežen* – neuravnotežen računalnik je manj ekonomičen. Pravili veljata predvsem za splošne računalnike in ne recimo za računalnike, ki so namenjeni za veliko število numeričnih izračunov. Pri teh je seveda število vhodno izhodnih podatkov dosti manjše.

Vodilo v računalniku povezuje med seboj CPE, glavni pomnilnik, vhodno izhodne naprave, etc etc etc. Realizirano je lahko z žicami, po katerih se prenašajo signali. Signale delimo na:

1. **Podatkovne signale**. Po njih se prenašajo podatki. Njihovo število je enako največjemu številu bitov, ki se lahko prenesejo naenkrat. To število se imenuje širina prenosne poti.
2. **Naslovni signali**. Z njimi je določen naslov pomnilniške besede, ali naslov vhodno izhodne enote, na katero se nanaša prenos podatkov. Število naslovnih signalov določa velikost naslovnega prostora.
3. **Kontrolni signali**. Ti signali določajo smer prenosa (branje ali pisanje), število prenesenih podatkov ter časovno zaporedje dogodkov med prenosom.

V nekaterih sistemih se podatkovni in naslovni signali prenašajo na isti liniji, tako da se v enem trenutku prenašajo podatkovni v drugem pa naslovni signali. Temu primeru uporabe pravimo **multipleksiranje**. V vsakem trenutku lahko poteka po enem vodilu samo en prenos, ki ga vodi gospodar vodila. Ostale naprave so v tem primeru sužnji. Če uporabljamo DMA krmilnike je obenem možnih več gospodarjev vodila in je potrebno urediti mehanizem, ki ureja **arbitražo vodila**. Imenuje se **arbeiter**. DMA je lahko gospodar in suženj, procesor pa je lahko le gospodar (razen v večprocesorskih sistemih).

Prenos po vodilu delimo na:

1. **Sinhronski prenos**: Čas prenosa je vedno enak T, ki predstavlja celo število urinih period. Če imamo na vodilu različno hitre enote je ta čas vedno enak najpočasnejši enoti. Hitrost vseh naprav je ista kot hitrost najpočasnejše naprave. Če podatek ni prenesen se urin cikel lahko podaljša, še zmeraj pa je enak številu T urinih period.
2. **Asinhrosnki prenos**: Čas prenosa ni vnaprej določen. Traja toliko časa dokler naprava ne aktivira ustreznega kontrolnega signala za konec prenosa. Najhitrejše enote prej, počasnejše kasneje. Obstaja tudi vezje,. ki konča predolge cikle in javi error.

Pri hitrih računalnikih se uporablja sinhronski prenos, ker bi preveč asa porabili za pot kontrolnih signalov (predvsem potrditvenega signala) zaradi zamud po žicah (6/ns/m).

Časovni parametri pri prenosu so:

1. **Vspostavitveni čas (Setup time)** – v tem času mora biti podatek stabilen, da ga enota lahko prebere, nanaša se na stran, ki prejema informacijo.
2. **Čas dostopa (Access time)** – čas, ki preteče od trenutka vspostavitve naslovnih in kontrolnih signalov do trenutka, ko enota da podatek.
3. **Držalni čas (Hold time)** – čas, ki ga zahtevamo in v katerem je podatek še zmeraj prisoten na podatkovnih linijah po koncu prenosa.

PMS notacija: **M** – Memory (pomnilnik),. **L** – Link (povezava), **K** – Control (krmiljenje), **S** – Switch (stikalo), **T** – Transducer (pretvornik), **D** – Data-operation (podatkovna operacija), **P** – Procesor (procesor), **X** – External (zunanjost).

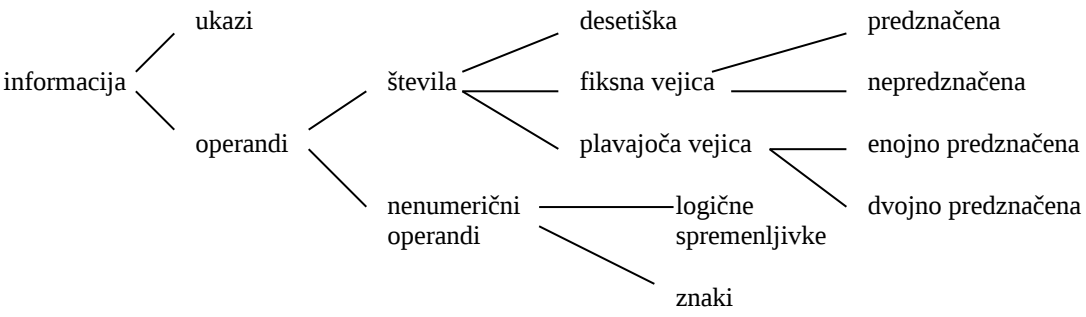
Pc - centralna procesna enota
Pio - vhodno izhodni procesor
Mp - glavni pomnilnik
Mc - predpomnilnik
Kio - krmilnik vhodno izhodne naprave
Km - krmilnik pomnilnika

Šest nivojev, ki jih imamo pri večini današnjih računalnikov:

nivo 5	Višji programski jeziki		
		prevajanje ali interpretiranje	
nivo 4	Zbirni jezik		
		prevajanje	
nivo 3	Operacijski sistem		
		delno interpretiranje	običajna meja med fizičnim in programskim delom računalnika
nivo 2	Običajni strojni jezik		
		interpretiranje	
nivo 1	Mikroprogramski jezik		
		interpretiranje	
nivo 0	Digitalna elektronika		

Opremo računalnika delimo na fizično ali strojno (Hardware) ter programsko (software).

4. Predstavitev informacije in aritmetika



Nenumerične znake predstavimo z abecedo, ki je lahko urejena po **ASCII** ali **EBCDIC** standardih.

- Števila v ***fiksni vejici*** :

Števila lahko opišemo kot zaporedje števil, ko so ločene z vejico. Levo od vejice je celo število, deno od vejice pa je število, ki predstavlja ulomek. Lahko ga posplošimo na uteži oblike r^i, kjer je r **baza** ali **radiks** številskega sistema. Pri današnjih računalnikih večinoma 2 ali 10. V prvem govorimo o binarni predstavitvi in binarni aritmetiki, v drugem pa o desetiški predstavitvi in o desetiški aritmetiki.

za predstavitev predznačenih števil imamo 4 načine(tabela *STR 92*):

1. **Predznak in velikost**. Najbolj levi bit predstavlja predznak števila. 0 – ozitivno, 1 – negativno.
2. **Predstavitev z odmikom**. K številu se najprej prišteje konstanta, ki zagotovi, da je število pozitivno. Tej kostanti pravimo odmik (bias). Pri računanju je odmik potrebno upoštevati!
3. **Eniški komplement**. Najbolj levi bit pomeni predznak. pri negativnih številih vse bite, vključno z predznakom invertiramo (zamenjamo 0 z 1 in 1 z 0). S takimi števili lahko delamo, kot da so pozitivna, pri rezultatu je potrebno pri prenosu iz najbolj levega bita prišteti 1.
4. **Dvojiški komplement**. Bite invertiramo in prištejemo 1. Ni potrebno upoštevanje prenosa kot pri eniškem komplementu.

Ključno vezje za realizacijo vseh operacij je paralelni binarni seštevalnik. Z njim so narejene vse operacije, tudi odštevanje, množenje in deljenje, ter tudi operacije v plavajoči vejici. Pravimo mu tudi n-bitni seštevalnik.
Seštevalnik s plazovitim prenosom je najpočasnejši. Plazoviti prenos se imenuje zato, ker se prenašalni bit Ci, prenaša iz najbolj nizkega mesta (skrajno desno) v ostale seštevalnike.
Seštevalnik z vnaprejšnjim prenosom je boljši. Pri njem se vrednost prenosa Cn računa iz vseh n vhodnih spremenljivk naenkrat in ne večkrat kot pri plazovnem.
Pri množenju poznamo matrični množilnik in matrični delilnik. n-bitni matrični množilnik je zgrajen iz $(n-1)^2$ 1-bitnih polnih seštevalnikov, in n^2 AND vrat. Če imamo za operande števila, ki so predstavljena v dvojiškem komplementu si pomagamo z **Boothovim algoritmom**.
Možnost prekoračitve se preverja na dva načina. S posebnim bitom ali pa s servisnim programom.

- števila v plavajoči vejici

  Za predstavitev porabimo manj ševilk. Lažje je napisati 6×10^8 kot pa 600000000. Vsa števila v plavajoči vejici so predstavljena kot $m \times r^e$ kjer so m, r in e cela števila m – *mantisa*, r – *baza* ali *radiks*, e – *eksponent*. Števila morajo biti predstavljena enolično, kar pomeni, da so normalizirana. Za mantiso se danes najpogosteje uporabljata predznak in velikost. V binarni obliki je prvi bit vedno enak 1 in predstavlja **normalni bit**, kar pomeni, da je število normalizirano. Števila, ki niso normalizirana so denormalizirana. To so pogosto števila, ki so premajhna, da bi jih predstavili v normalizirani obliki. Temu dogodku pravimo **podkoračitev** (underflow). Pri računanju s števili v plavajočui vejici imamo *enojno natančnost* ali *dvojno natančnost*.
  Za predstavitev števil v plavajoči vejici in računanje z njimi se danes uporablja standard IEEE 754 (knjiga *STR 106*):

  1. **Normalizirana števila** – eksponent nima vseh bitov enakih 0 ali vseh bitov enakih 1.
  2. **Denormalizirana števila** – exponent ima vse bite enake 0, mantisa pa je različna od 0. S tem je rešen problem podkoračitve (underflow).
  3. **Ničli** – eksponent in mantisa imata vse bite enake 0.
  4. **Neskončnosti** – eksponent ima vse bite 1, mantisa pa vse bite 0 – rešen problem prekoračitve (overflow)
  5. **Neveljavna števila** – eksponent ima vse bite 1, mantisa pa je različna od 0.

5.Ukazi

Ukaz mora vsebovati informacijo o operaciji, ki naj jo izvrši ter o operandih nad katerimi se bo izvršila. Polje, ki vsebuje informacijo ooperaciji imenujemo operacijska koda. Tako razdelitev ukaza imenujemo format ukaza. Pomembno je razlikovanje med ukazi in operacijami. Izvajanje iste operacije lahko dosežemo z več različnimi ukazi – vsak od njih ima lahko drugačen format. S stališča matematike so vsi isti, s stališča računalnika pa je pomen predvsem o dostopu do operandov.

dimenzije osnovnih lastnostih pri določitvi definicije ukaza:

1. Način shranjevanja operandov v CPE
2. Število eksplicitnih operandov v ukazu
3. Lokacija operandov in načini naslavljanja
4. Operacije
5. Vrsta in dolžina operandov

S programsko dostopnimi registri v katerih so shranjeni ukazi se da doseči dve pomembni lastnosti: večjo hitrost in krajši ukaz.

3 osnovni načini shranjevanja operandov v CPE:

1. **Akumulator**. Pomnilnik v CPE je narejen iz enega samega registra, ki mu pravimo akumulator. Vanj lahko shranimo en operand. To rešitev srečamo še danes v kombinaciji z ostalimi registri. ukaza za prenos sta LOAD in STORE.
2. **Sklad** (stack). Povečanje pomnilnika v CPE lahko naredimo s skladom, ki se obnaša kot FIFO buffer. Ukaza za delo sta PUSH in POP. Aritmetični izrazi so zapisani v postfix notaciji. Računalnikom, ki uporabljajo takšno arhitekturo pravimo *skladovni računalniki* ali *računalniki s skladovno arhitekturo*.
3. **Množica registrov**. V CPE imamo več registrov, lahko so splošnonamenski ali pa razdeljeni po pomenu. Lahko so programsko dostopni ali pa nedostopni.

Danes poznamo:

- 3+1 operandne računalnike.
  OP3 = OP2 + OP1
  PC = OP4
- 3 operandni računalniki
  OP3 = OP2 + OP1
  PC = PC + 1
- 2 operandni računalniki
  OP2 = OP2 + OP1
  PC = PC + 1
- 1 operandni računalniki
  AC = AC + OP
  PC = PC + 1
- brez operandni računalniki
  sklad(vrh) = sklad(vrh) + sklad(vrh-1)
  PC = PC + 1

Glede na lokacijo operandov poznamo danes 3 vrste računalnikov: **registrsko-registerske**, **registrsko-pomnilniške**, ter p**omnilniško-pomnilniške** računalnike. Glede na naslavljanje pa obstaja: takojšnje naslavljanje (operand je podan kot vrednost), neposredno naslavljanje (operand je podan z naslovom pomnilniške lokacije), posredno naslavljanje (*pomnilniško* – pomnilniški naslov lokacije kjer je shranjen pomnilniški naslov operanda; *registersko* – shranjen naslov registra odmik iz katerega se izračuna pomnilniški naslov operanda).

Operacije dekimo na:

1. Aritmetične in logične operacije
2. Prenosi podatkov
3. Kontrolne operacije
4. Operacije v plavajoči vejici
5. Sistemske operacije
6. Vhodno izhodne operacije

Na zgradbo ukazov vplivajo

Dolžina pomnilniške besede,	Število eksplicitno podanih operandov v ukazu,
Vrsta in število registrov v CPE,	Dolžina pomnilniškega naslova

RISC/CISC dilema (število ukazov).

RISC – Reduced Instruction Set Computer **CISC** – Complex Instruction Set Computer

S povečevanjem ukazov so hoteli zmanjšati dolžino operacijske kode, poenostaviti gradnjo prevajalnikov in povečati hitrost delovanja. Do RISC računalnikov je pripeljalo sledeče:

1. Težave pri uporabi kompleksnih ukazov v prevajalnikih. Velik del ukazov se uporablja zelo redko in v večini primerov za podobne operacije (dodati je treba še kak ukaz).
2. Spremenjeno razmerje med hitrostjo glavnega pomnilnika in CPE. Z uporabo predpomnilnikov se je razmerje med hitrostjo CPU in pomnilnikom dosti zmanjšalo.
3. Paralelizem v CPE. CPE lahgko dela več stvari obenem.

6 točk po katerih ima računalnik RISC arhitekturo:

1. Večina ukazov se izvrši v eni periodi CPE ure. Izjema so ukazi za množenje in deljenje.
2. Registrsko registreska zasnova.
3. Ukazi so realizirani s trdo ožičeno logiko in so zato dosti hitrejši kot mikroprogramsko realizirani.
4. Malo ukazov in načinov naslavljanja
5. Vsi ukazi imajo isto dolžino
6. Dobri prevajalniki (upoštevati morajo zgradbo CPE).

Odgovor na to, kaj je boljše RISC ali CISC še ni čisto določen.

6. Centralna procesna enota (CPE)

CPE deluje po dveh korakih. 1. ***FETCH***, 2 ***EXECUTE***.
V 1. se prevzame ukaz iz pomnilnika. Naslov ukaza je shranjen v PC. V 2. pa se ukaz izvede. Po koncu ukaza se poveča PC. Ta posotpek se ponavlja dokler ne pride do prekinitve. Takrat se sproži servisni program, ki poskrbi za servisiranje prekinitve. Nato CPU nadaljuje delo, ki ga je opravljal pred tem. Da program ne čuti prekinitve je poitrebno vse programsko dostopne registre shraniti. Za to imamo več možnosti, lahko uporabimo sklad ali pa kopijo registrov. Kakorkoli program ne sme čutiti, da je bil prekinjen.

Zgradbo CPE razdelimo na kontrolno enoto in podatkovno enoto.

Koraki pri izvrševanju ukazov (RISC):

1. Prevzem ukaza
2. Dekodiranje ukaza/dostop do registrov
3. Izvrševanje
4. Dostop do pomnilnika
5. Shranjevanje rezultata

Enote za zmogljivost CPE:

CPI = clocks per instruction; **MIPS** = milions instructions per second;**MFLOPS** = milions floating point operations per secod;
SPEC = System preformace evalution cooperative – **SPECmark89**, **SPECint92** & **SPECfp92** (najbolj c0oL).

Delovanje kontrolne enote CPE lahko označimo kot končni avtomat (prehajanje med stanji). Kontrolna enota je lahko realizirana s **trdo ožičeno logiko** ali pa **mikroprogramsko**. Trdo ožičena je realizirana s množico elektronskih komponent, večinoma flip-flopov in sekvenčnih vezij.

Mikroprogramska kontrolna enota je realizirana z mikroprogramom. Vsako stanje diagrama prehajanj stanj je en mikroukaz. Mikroprogramska kontrolna enota zgleda kot en majhen računalnik. Realizacija kontrolne enote postane pisanje programov zanjo.

Pri prekinitvi CPE najprej pogleda v posebni register v katerem ima shranjen prioritetni nivo. Upoštevajo se samo prekinitve, ki imajo višji prioritetni nivo, kot pa program, ki se trenutno izvaja. Če je na prekinitveni vhod priključenih več naprav, lahko preverjamo katera je zahtevala prekinitev na dva načina: **Programsko izpraševanje** ali pa **Marjetična veriga**. Programsko izpraševanje je počasnejše, za marjetično verigo pa je potrebno nekaj več ožičene logike pri napravah, ki prekinitve zahtvevajo.

- **Cevovodno procesiranje:**

Cevovodno procesiranje je način za uvajanje paralelnega izvajanja ukazov v procesorju. Ukaz se izvršuje čez več stopenj. Podobno kot pri tekočem traku se v vsaki stopnji izvede nekaj. Zmogljivost cevovoda je določena s hitrostjo izstopanja ukazov iz cevi cevovoda. Cevovod je ponavadi zgrajen iz 5 delov:

1. IF – Instruction Fetch – prevzem ukaza
2. ID – Instruction decode – dekodiranje ukaza/dostop do registrov.
3. EX – Execute – izvrševanje
4. MEM – Memory access – dostop do pomnilnika
5. WB – Write Back – shranjevanje ukaza

Cevovodne nevarnosti delimo na **strukturne nevarnosti**, **podatkovne nevarnosti** in **kontrolne nevarnosti**.

7. Glavni pomnilnik in predpomnilnik

Z izrazom glavni pomnilnik označujemo prostor iz katerega CPE jemlje ukaze in operande ter shranjuje rezultate.

Pri delovanju pomnilnika so najpomembnejšetri stvari:

1. **Hitrost pomnilnika**. To merimo s povprečnim povprečnim številom besed, ki jih je pomnilnik sposoben v eni sekundi sprejeti ali pa oddati. Ta hitrost je odvisna od dostopnega časa do pomnilniških elementov. Pogosto govorimo tudi o pasovni širini ali prepustnosti pomnilnika.
2. **Mehanizmi za dodeljevanje pomnilniškega prostora** programom ter za **zaščito** tako dodeljenega prostora pred drugim programom.
3. **Podatkovne poti**, ki povezujejo pomnilnik s CPE in vhodno izhodnimi napravami ali procesorji.

Pomnilnike delimo glede na način dostopa:

1. **Naključni dostop**. Pri tem načinu se da dostopati do nalkjučnega naslova v pomnilniku. Čas dostopa je povsod enak.
2. **Zaporedni dostop**. Pri tem načinu je dostop do neke besede odvisen od naslova besede, do katere je bil narejen dostop tik pred tem. Da pridemo do neke besede je potrebno iti čez vsebesede po vrsti do tam. Primer je naprimer magnetni trak.
3. **Krožni dostop**. To je posebna oblika zaporednega dostopa. Posebnost je le v tem, da ne rabimo »prevrtevati« na začetek oziroma na konec, ampak krožimo okoli. Primer je recimo krožnica.
4. **Direktni dostop**. Primer je trdi disk. Gre za kombinacijo zaporednega in krožnega dostopa.

Poznamo bralne pomnilnike ter bralno pisalne pomnilnike. Pri slednjih je pogosto prisotno vprašanje **obstojnosti**. Shranjena informacija se lahko čez čas izgubi. Temu so lahko krivi destruktivno branje, dinamično shranjevanje ter odklop vira energije. Pri dinamičnih pomnilnikih je potrebno inforacijo »osveževati«. Statični pomnilniki ne potrebujejo osveževanja.

Metabiti so tisti biti v pomnilniški besedi, ki opisujejo pomen preostalih bitov. Pogosto jim pravimo označevalni biti. Večina današnjih računalnikov jih nima ali pa jih uporablja za korekcijo napak v pomnilniku (parity bits). Metabiti pridejo v poštev, če je CPE zgrajena tako, da jih zna izkoriščati (se zaveda programa). Prednosti pri uporabi metabitov:

1. **Manjše število ukazov**. Pri uporabi metabitov je lahko za vsako aritmetično operacijo dovolj le en ukaz in ne cela vrsta, da se določi kakšnega tipa je število ipd.
2. **Avtomatska pretvorba operandov**. Pri mnogih izračunih se pojavijo meašni tipi operandov ter tudi različnih dolžin. Na računalnikih brez metabitov so potrebni ukazi ali celo procedure, ki pretvorijo vse operande na isti tip in dolžino. Na računalnikih z metabiti to ni potrebno, ker CPU lahko pretvorbo takoj opravi sam.

3. **Avtomatsko računanje in pretvarjanje indeksov**. Z metabiti lahko definiramo »podatkovne deskriptorje«, ki nam opisujejo eno ali večdimenzionalna polja.Z njimi lahko CPU izračuna naslov besede v kateri je želeni element polja.CPU se »zaveda« razlike med elementarnimi podatki in polji.

4. **Avtomatsko vzpostavljanje klicnih parametrov**. CPE se začne zavedati tudi programske strukture, posebno podprogramov in procedur. CPU lahko avtomatsko poveže klicne parametre.

5. **Ugotavljanje nesmiselnih operacij**. Operacije kot so množenje ukazov in podobno se takoj zaznajo.

6. **Ugotavljanje nedefiniraih operandov**. Vsak operand, ki ga CPE uporabi kot vhodni podatek v neki operaciji mora biti pred tem definiran. Če ni je to skoraj zanesljivo napaka. Z metabiti je mogoče ugotoviti kdaj je podatek definiran.

Zaščita glavnega pomnilnika je mehanizem, ki omogoča zaščito pomnilniškega prostora, ki je dodeljenen enemu programou pred drugim programom, saj bi to lahko privedlo do razpada sistema (crash). S pojavom večuporabniških in multiprogramskih računalnikov je ta problem še bolj pomemben, saj zagotavlja varnost sistema ter podatkov.

Najpreprostejši primer zaščiite je par registrov, ki določa zgornjo in spodnjo mejo zaščitenega dela vendar to ni praktično, ker je lahko program »razmetan« po celem pomnilniku. Danes se uporabljajo drugačni mehanizmi.Glavni pomnilnik je razdeljen na **bloke** ali **strani** reda velikosti (512, 1024, 2048, 4096, …), kjer je vsakemu programu dodelje pomnilniški prostor celega števila teh segmentov. Začetek vsakega segmenta ima definicijo komu pripada. To so metabiti, ki jih imenujemo **zaščitni ključ**.

Hitrost pomnilnika je moč povečati na več načinov:

1. Hitrejši pomnilniški elementi.
2. Večje število naenkrat prenesenih bitov.
3. Predpomnilnik
4. S pomnilniškim prepletanjem. – pomnilnik razdelimo na module. Vsak modul lahko izvaja neodvisno operacijo (na enega pišemo iz drugega beremo).

Predpomnilnik je hiter pomnilnik, ki ga priključimo med CPE in glavni pomnilnik. Pogosto se uporablja Harwardska arhitektura. Izkorišča pojma kot sta prostorska in časovna lokalnost pomnilniških dostopov. Verjetnost zadetka je več kot **90%** pri zgrešitvah pa se zažene zamenjevalna strategija predpomnilnika, ki poskrbi za prenos podatka iz glavnega pomnilnika.

Uporaba in dodeljevanje pomnilnika se odvijajo predvsem po »dogodkih« v računalniku. Pomnilnik se razdeljuje pri **Pisanju programa,** kjer programer ustvarja podatkovna polja, procedure, začasne spremenljivke in podobno, nadaljuje pri **prevajanju**, kjer prevajalnik iz programske kode tvori strojno kodo Pri preprostih programskih jezikih je lahko rezultat prevajanja že končna preslikava logičnega v fizični prostor. Skoraj vsi programi pri svojem **delovanju** zavzemajo nek prostor ter potrebujejo dodatne programe. Vse programe je potrebno **povezati** v celoto (to delo opravi linker) in nato to celoto naložiti v glavni pomnilnik. Pri izvajanju programa se tudi spreminjajo fizični naslovi.

8. Navidezni pomnilnik

Vzrok za uporabo navideznega pomnilnika je predvsem njegova nizka cena. Organizacija pomnilnika je pogosto v naslednjem zaporedju:

S pomnilniško hierarhijo hočemo doseči, da CPE vidi velik in počasen pomnilnik kot hiter. Čas dostopa je ponavadi odvisen na katerem nivoju pomnilniške hierarhije se nahaja želena beseda. To je ena izmed slabosti. Potrebno je tudi nekaj več logike za prehajanje in zamenjevanje vsebine po hierarhiji navzgor in navzdol, vendar ekonomski razlogi odtehtajo vse to.

Navidezni naslov je naslov med CPE in preslikovalnikom. Ta naslov se v preslikovalniku, ki je med CPUjem in predpomnilnikom preslika v fizični naslov.

Odstranjevanje

Odstranjevanje (paging) je posebna oblika organizacije navideznega pomnilnika. Pomožni pomnilnik je razdeljen na **bloke** enakih velikosti, ki jim pogosto pravimo **strani**. Vse strani skupaj sestavljajo navidezni pomnilnik. Na enako velike bloke, ki jim pravimo **okviri strani** je razdeljen tudi glavni pomnilnik. **Preslikovalna funkcija** je definirana preko posebne tabele z imenom **tabela strani**. Polja v tabeli se imenujejo deskriptor strani. Sestavljen je iz več bitov:

- **bit veljavnosti** – če je 1 so vsi parametri v deskriptorju pravilni. Če je 0 stran ni definirana in parametri nimajo pomeana.
- **bit prisotnosti (zadetka).** – Take strani poimenujemo aktivne strani.
- **zaščitni ključ RWX** - pove kakšna vrsta dostopa je dovoljena.
- **bit spremembe** – ko se stran prenese v glavni pomnilnik, se ta bit vedno postavi na 0. Če pride do pisanja v kateregakoli od naslovov se bit postavi na 1, kar pomeni, da se je stran v času »bivanja« spremenila.
- **številka okvira FN** – ta parameter podaja številko okvirja v katerem je stran. – naslov strani v glavnem pomnilniku.

Preslikovalna funkcija dela na načinu **linearnega preslikovanja**.

Segmentacija

Podobno kot odstanjevanje samo s to razliko, da so segmenti različno dolgi, strani pa so enako dolge. (podobneje se mi zdaj ne da pisat – pol enih ponoči).

9. Vhod in izhod

Vsak procesor ima možnost vhoda in izhoda. Vse realizacije morajo vsebovati sredstva s katerimi zagotavljamo naslednje tri funkcije:

1. **Izbiro ali naslavljaje** posamezne V/I naprave.
2. **Prenos podatkov** v ali iz naprave
3. **Usklajevanje ali koordinacija časovnih parametrov** V/I naprave s parametri CPE oziroma glavnega pomnilnika.

Če uporabimo vhodno izhodni krmilnik se programerjev pogled na vhodno izhodno napravo poenostavi. Krmilnik je iz stališča CPE videti kot skupek registrov na katere lahko gledamo kot na posebne pomnilniške lokacije.

Naslavljanje vhodno izhodnih naprav lahko vršimo na 3 načine:

1. **Pomnilniško preslikan vhod/izhod**. Registri naprav so videti enako kot pomnilniške lokacije. Za delo z njimi lahko uporabim vse ukaze za delo s pomnilnikom. Posebni vhodno izhodni ukazi niso potrebni.
2. **Ločen vhodno izhodni prostor**. Registri naprav so v posebnem naslovnem prostoru, ki ga lahko naslavljamo le preko posebnih ukazov za delo z njimi.
3. **Posredno preko vhodno izhodnih procesorjev**. Večina večjih računalnikov ima enega ali več vhodno izhodnih procesorjev, ki poskrbijo za komuniciranje med napravamo I prenos podatkov v ali iz njih.

Prenos vhodno/izhodnih podatkov:

1. **programski vhod/izhod**. – prenos je ves čas pod nadzorom CPE.
2. **programski vhod/izhod z uorabo prekinitev.** – Izkoriščenost računalnika je boljša, če se za obveščanje o prenosu uporabijo prekinitve (za začetek, konec, error). Ta čas med prekinitvami CPE lahko dela kaj drugega.
3. **neposreden dostop do pomnilnika.** – Pri napravah, ki zahtevajo hiter prenos informacij je programski vhod/izhod prepočasen. Tu nastopijo DMA kontrolerji, ki opravijo prenos direktno v pomnilnik, CPE pa lahko ta čas počne kaj drugega. DMA krmilnik je neke vrste računalnik, ki je sposoben narediti vse kar se tiče prenosa sam.
4. **Vhodno izhodni procesorji** – za periferijo (vhod in izhod) poskrbijo periferni procesorji ali V/I kanali. priključeni so na posebna vodila Osnovni cilj je razbremeniti CPE. Vsaka naprava se opremi z DMA kontrolerjem, ki so pod nadzorom vhodno izhodnih procesorjev.