3. Osnovni principi delovanja

1. Von Neummanov računalniški model; kaj je zanj značilno?

Von Neumanov računalniški model ima naslednje zahteve:

- sestavljajo ga trije osnovni deli: CPE, glavni pomnilnik in V/I sistem
- ima program, shranjen v glavnem pomnilniku, ki vodi delovanje stroja
- CPE jemlje (FETCH) ukaz za ukazom iz glavnega pomnilnika (iz naslova kamor kaže PC – program counter) in jih izvršuje (EXECUTE) ter pri tem povečuje PC (za 1 ali pa n, če gre za skok)

CPE – Razdelimo jo lahko na:

- *kontrolno enoto* (prevzema ukaze in operande in aktivira ustrezne operacije),
- *podatkovna enota*
  - *aritmetično-logična enota* (izvaja operacije)
  - *registri* (programsko dostopni [hitrejše delovanje] ali nedostopni [ima vsaka CPE za realizacijo delovanja]).

GLAVNI POMNILNIK – Sestavljen je iz več pomnilniških besed, katerih vsaka ima enoličen naslov.

- V njem so ukazi in operandi, ki jih uporablja CPE.
- Pri tem modelu računalnik ne loči med ukazi in operandi.

V/I SISTEM – je namenjen prenosu informacij v/iz zunanjega sveta.

- Sestavljen je iz V/I naprav.
- Nekatere izmed njih služijo tudi kot pomožni pomnilnik. Fizično je to največji del računalnika.

Obstajati mora mehanizem, ki ukaze naloži v glavni pomnilnik – bootstrap loader. Delovanje računalnika potem popolnoma določajo ukazi, v dveh korakih: fetch in execute. PC – PC +1 oz. PC – PC +n.
Von Neumannov model je ukazno pretokovni (ukazi določajo delovanje rač.), obstaja še podatkovno pretokovni.

2. Kaj je prostorska lokalnost pomnilniških dostopov?

Lokalnost pomnilniških dostopov: programi več kot enkrat uporabijo iste ukaze in operande in bolj pogosto tiste, ki so v pomnilniku blizu trenutno uporabljenim.

- Izkustveno pravilo: tipičen program 90% časa uporablja samo 10% ukazov.
- Povzroča jo način pisanja programov in že samo delovanje Von Neumannovih računalnikov (ukazi si zaporedno sledijo).
- Lokalnost omogoča pomnilniško hierarhijo.

Razdelimo jo lahko na:

prostorsko lokalnost – po pojavu naslova A(i) bodo verjetno naslednji naslovi po lokaciji v pomnilniku blizu naslova A(i).

- Če ni skoka, se ukazi jemljejo zaporedno.
- Strukture, kot so polja, so običajno na zaporednih naslovih in se uporabljajo zaporedoma po svojih indeksih
- Program je razdeljen na podprograme in procedure, ki jim je dodeljen tudi majhen zaporedni prostor, ker so večinoma kratke.

časovna lokalnost – program ob času t tvori naslove, ki jih je tvoril malo pred t in ki jih bo nekoliko po t.

- Zanke (isto zaporedje ukazov oz. naslovov se ponovi velikokrat).
- Začasne spremenljivke, zaradi katerih se naslovi določenih operandov pojavljajo večkrat.

Lokalnost lahko kvantificiramo še z delovno množico (Working Set) – V(t,T). Če se v času T pojavi N ukazov in v intervalu od t-T do t V(t,T), je množica V(t,T) veliko manjša od N. Vsebina zaporedno si sledečih množic V(t,T) se spreminja počasi (prekrivanje).

3. Kaj je časovna lokalnost pomnilniških dostopov? WS (Working set). Uporabnost.

Glej 2.

4. Amdahlov zakon in Case/Ambdahlova pravila (kaj pravijo in čemu so pogoj)?

Pove nam, kako močno lahko pohitrimo računalnik s paralelizmom določenih delov.
Če za faktor N pohitrimo delovanje pri vseh operacijah, razen pri f-tem deležu od vseh, potem je povečanje hitrosti računalnika *(ali drugače rečeno: če je delovanje N-krat hitrejše (1-f)-ti del časa izvajanja in enako hitro f-ti del)* enaka:

$$S(N) = \frac{1}{f + \frac{(1-f)}{N}} = \frac{N}{1 + (N-1) * f}$$

Maksimalna pohitritev, ki sledi iz enačbe: S(N) = 1/f.
Za večino problemov je bolje en hiter procesor, kot več počasnejših, razen izjemoma, ko je stopnja paralelnosti (1-f) visoka.

Skupaj s sodelavcem R.P. Case-om, je Amdahl razvil še dve pravili:

- velikost glavnega pomnilnika v Bajtih mora biti najmanj enaka številu ukazov, ki jih v sekundi izvede CPE
- zmogljivost V/I sistema v bitih na sekundo mora biti najmanj enaka številu ukazov, ki jih v sekundi izvede CPE

Če računalnik ustreza tem pravilom, rečemo da je uravnotežen (bolj ekonomičen).

5. Kaj je DMA krmilnik in kako deluje?

- DMA (Direct Memmory Access) označuje neposredno komunikacijo med glavnim pomnilnikom in V/I napravo (nasprotje od programskega V/I, ko z V/I napravo komunicira le CPE). Računalnik lahko uporablja oba.
- Prenos je realiziran s posebno enoto - DMA krmilnikom, ki je ali samostojna ali del krmilnika naprave.
- Videti je kot določeno število registrov, v katere se lahko piše ali se iz njih bere. Na te registre lahko gledamo podobno kot na besede v glavnem pomnilniku – razlika je v tem, da pisanje ali branje pri njih lahko sproži neko operacijo v napravi ali odraža stanje po prejšnji operaciji. Pri vsakem krmilniku je točno določeno kaj se zgodi, če pišemo v nek register ali če beremo iz njega. (disk: pisanje v register -> bralno-pisalna glava se premakne na določeno sled; branje statusnega registra -> ugotovimo, kdaj je premik končan)
- DMA krmilnik sam komunicira z glavnim pomnilnikom in med prenosom sodelovanje CPE ni potrebno.
- Ker za prenos ni potrebno prevzemati ukazov, je hitrejši, vendar zaradi krmilnika tudi dražji.

6. V/I procesorji.

- Posebna izvedba DMA krmilnikov, ki omogoča prenose velikih količin podatkov ob minimalnem sodelovanju CPE.
- Osnovni cilj je razbremenitev CPE pri V/I prenosih, saj CPE samo sporoča svoje zahteve vhodno/izhodnim procesorjem, ki nato poskrbijo za podrobnosti pri izvrševanju prenosov podatkov
- Rešitev pa je še veliko dražja od DMA krmilnikov, zato se pogosto pojavlja predvsem pri večjih računalnikih.

Dva pristopa:

- Ločena V/I vodila: na vsak V/I procesor je priključeno določeno število V/I naprav.
- Koordinatni sistem vodil: vsak V/I procesor lahko komunicira z vsako V/I napravo. Bolj splošen, a dražji.

7 Kaj je to pomnilniško preslikan V/I?

- Pomnilniško preslikan V/I je del naslovnega prostora glavnega pomnilnika, ki je rezerviran za V/I naprave.
- Registri krmilnikov so v pomnilniškem naslovnem prostoru in so iz CPE videti kot pomnilniške besede.
- Za branje/pisanje lahko uporabljamo vse ukaze za dostop do pomnilnika,
  - tako da posebni V/I ukazi niso potrebni.

Poznamo še:

- ločen vhodno/izhodni prostor:
  - registri krmilnikov so v posebnem naslovnem prostoru, ki je ločen od pomnilniškega.
  - Za dostop do registrov so potrebni posebni V/I ukazi -> med izvajanjem teh ukazov CPE aktivira signal, ki pove, da se naslavlja V/I naslovni prostor

- posredno preko vhodno/izhodnih procesorjev:
  - tudi tu so registri krmilnikov so v posebnem naslovnem prostoru, vendar ta prostor iz CPE ni neposredno dostopen -> do njega imajo dostop V/I procesorji
  - glej še vprašanje 6

4. Predstavitev informacije in aritmetika

8. Števila v fiksni vejici. Glavna prednost komplementov in slabosti ostalih predstavitev števil. Katera predstavitev se danes največ uporablja? Prenos, preliv.

- Vejica je fiksna.
  - Številke na levi strani vejice predstavljajo celo število,
  - na desni pa ulomek (decimalni del).
- Vsaka številka ima svojo težo glede na pozicijo – pozicijska notacija.
- V glavnem se uporablja dvojiška predstavitev (desetiška se uporablja izključno za poslovne obdelave – banke, kjer je to celo uzakonjeno). Desetiška potrebuje za predstavitev števil več bitov in pa logična vezja za desetiško aritmetiko so bolj zapletena.

Predstavitev predznačenih števil:

1.Predznak in velikost:

- najbolj levi bit predstavlja predznak → 0 pozitivno in 1 negativno.
- Pri seštevanju in odštevanju je treba bit za predznak obravnavati drugače kot ostale bite.
- Imamo dve ničli: +0, -0.
- Predstavitev je dobra pri množenju in deljenju (ki sta redki operaciji)
- uporablja se za podajanje *mantise* v predstavitvi s plavajočo vejico (IEEE 754)

2.Predstavitev z odmikom:

- vsa števila so pozitivna (k številu se najprej prišteje konstanta - odmik),
- najbolj levi bit tudi tu določa predznak, vendar ga ni potrebno obravnavati drugače od ostalih → 0 negativno
- če so vsi biti 0 predstavlja to najbolj negativen eksponent (število s takim eksponentom pa je po absolutni vrednosti najbližje ničli)
- Slabost: odmik je treba zmeraj popravljati (pri seštevanju odštevati in pri odštevanju prištevati).
- Prednost: vrednost zaporedja bitov odraža velikost števila
- Uporablja se za podajanje *eksponenta* v predstavitvi IEEE 754 (števila v plavajoči vejici)

3.Eniški komplement:

- Tudi tu najbolj levi bit določa predznak → 0 pozitivno in 1 negativno
- Pozitivna števila so predstavljena enako kot pri načinu ''predznak in velikost''
- predstavitev negativnega števila dobimo tako, da najprej zapišemo pozitivnega, zatem pa vse bite invertiramo.
- Invertiranje bitov je ekvivalentno odštevanju števila od $2^n - 1$.
- Prednost: vse bite obravnavamo enako.
- Pri prenosu iz n-1. mesta moramo k rezultatu prišteti 1 (absolutna vrednost rezultata je večja od $2^n - 1$ in je ne moremo predstaviti z n biti+)
- Imamo dve ničli.

4.Dvojiški komplement:

- kot eniški, le da po invertiranju prištejemo še ena (aritmetično je isto kot odštevanje od 2^n)
- Pri prenosu iz n-1. mesta ni treba prištevati enke.
- Imamo samo eno ničlo.
- Najbolj pogosto uporabljen.

Glavna prednost komplementov:

- pri seštevanju in odštevanju (teh operacij je dosti več kot množenja in deljenja) bit za predznak upoštevamo povsem enako kot ostale bite.
- Pri dvojiškem komplementu pa poleg tega nimamo dveh predstavitev za ničlo.

Do prenosa pride pri prekoračitvi obsega NEpredznačenih števil:

ko je rezultat operacije izven področja $0 \leq x \leq 2^n - 1$

Do preliva pride pri prekoračitvi obsega PREDznačenih števil:

ko je rezultat operacije izven področja $-2^{n-1} \leq x \leq 2^{n-1} - 1$

9. Kje se uporablja predstavitev z odmikom?

Predstavitev z odmikom se uporablja npr. pri standardu IEEE 754 (števila v plavajoči vejici) in sicer za predstavitev eksponenta.

10. Predstavitev numeričnih operandov v plavajoči vejici. m, r, e, podliv, preliv, normalizacija, normalizirana števila.

- Omogoča predstavitev tudi zelo majhnih in zelo velikih števil.
- Število razbijemo na tri dele: mantiso m, eksponent e in bazo r.
- Eksponent in mantisa sta predstavljena s fiksno vejico,
  - zaradi praktičnih razlogov za bazo uporabljamo samo 2 ali 10.
  - Eksponent pove, na katerem mestu je decimalna vejica, ki plava levo (e++) ali desno (e--).
- Ker je baza konstantna, lahko rečemo, da je število v plavajoči vejici predstavljeno s parom (m,e).

Preliv:

- glej 8.

Podliv (v fiksni vejici ga ni):

- do njega pride pri številih, ki so po absolutni vrednosti premajhna, da bi jih bilo mogoče predstaviti v normalizirani obliki
- Zato jih denormaliziramo ali zaokrožimo na 0.

Normalizacija:

- Za računanje je zaželeno, da so števila predstavljena na nek enolično definiran način.
- Ta način naj bo tak, da omogoča največjo možno natančnost računanja.
- Število v plavajoči vejici je normalizirano takrat, ko mantisa na levi strani nima ničel.
  - Drugače povedano: prva številka mantise mora biti različna od nič.
- Števila, ki niso normalizirana, so denormalizirana in jih lahko normaliziramo s pomikanjem mantise v levo ali desno

11 Standard IEEE 754. Enojna in dvojna natančnost. Rezervirane kombinacije bitov. 5 vrst števil.

- baza r = 2
- Eksponent je predstavljen v načinu **»*predstavitev z odmikom*«**
- Mantisa je predstavljena v načinu **»*predznak in velikost*«**
- Uporabljena je ***implicitna predstavitev normalnega bita***
- več formatov: 32 bitni(enojna natančnost), 64 bitni (dvojna) in 80 bitni (1+15+64; samo interno računanje)
- dovoljeno je računanje z denormaliziranimi števili (ena vrednost eksponenta je rezervirana)

Enojna natančnost:

- predznak S
- 8-bitni eksponent E z odmikom 127
- 23-bitna mantisa m
- vrednost: $(-1)^S(1,m)2^{E-127}$

Dvojna natančnost:

- predznak S
- 11 bitni eksponent E z odmikom 1023
- 52-bitna mantisa m
- vrednost: $(-1)^S(1,m)2^{E-1023}$

Standard IEEE 754 določa predstavitev petih vrst števil:

1. *Normalizirana števila:* E nima vseh bitov 0 ali vseh bitov 1.
2. *Denormalizirana števila:* E ima vse bite 0, m pa je različna od 0.
3. *Ničli:* E in m imata vse bite 0, predznak pa določa +0 in -0.
4. *Neskončnosti:* E ima vse bite 1, m vse bite 0, predznak pa določa +neskončno in -neskončno.
5. *Neveljavna števila:* E ima vse bite 1, m pa nima vseh bitov 0.

12. Zaokroževanje števil v računalniku (IEEE).

- Rezultat računanja (interno se uporablja 80 bitni format) mora biti enak matematično točni vrednosti in zaokrožen na mantiso dolžine 23 oz. 52 bitov.
- Zaokrožimo k najbližjemu še predstavljivemu številu, v primeru enake oddaljenosti pa k sodemu številu.
- Zadošča če mantiso podaljšamo za dodatne 3 bite, na osnovi katerih vrednost mantise povečamo za 1 ali pa ne, bite pa odvržemo:
  - **varovalni** (guard): potrebujemo dodatno mesto, ker sta vsota in razlika lahko za 1 mesto daljša od vhodnih operandov
  - **zaokroževalni** (round): 1 mesto ni dovolj za zaokroževanje, v vseh primerih pa zadoščata 2
  - **lepljivi** (sticky): zaradi pravila zaokroževanja k sodemu številu (logična disjunkcija ven padajočih bitov)

5. Ukazi

13. Kam lahko shranjujemo operande v CPE?

- Ni nujno, da obstaja možnost shranjevanja operandov v CPE.
- Kljub temu imajo vsi današnji računalniki v CPE majhen pomnilnik, v katerega je možno shraniti enega ali več operandov
  - programsko dostopni registri.
  - Omogoča večjo hitrost in krajše ukaze.

Razlikujemo tri možnosti za shranjevanje operandov v CPE:

1.Akumulator: imamo en sam (ali mogoče dva) register, v katerega lahko shranimo en operand.

- Kratki ukazi, preprosto a staro.
- V ukazih ga ni treba eksplicitno navajati (zelo kratki).
- Vmesne rezultate je treba prenašati nazaj v pomnilnik, ker ni prostora.
- Posledica: počasnost, promet med CPE in gl. pom.

2.Sklad: podoben akumulatorskemu → pri obeh direktno dostopen samo en operand (LIFO),

- promet med CPE in GP je manjši, ker je prostora za več operandov.
- Še vedno kratki ukazi in preprostost.
- Njihova prednost je v računanju dolgih matematičnih izrazov v postfiksni obliki, a je takih izrazov malo – več je prireditev.

3.Množica registrov: število je od 8 do 100.

- Zelo razširjena možnost.
- Ločimo dve rešitvi, glede na svobodo pri uporabi registrov:
  - vsi so ekvivalentni – splošnonamenski registri,
  - registri različnih dolžin: ena skupina za računanje z naslovi, druga za ostalo računanje
- Druga rešitev je bolj toga, vendar omogoča prihranke pri gradnji CPE.
- Ta rešitev je pomembna za cevovodne računalnike, kjer registri morajo shranjevati vmesne rezultate.

14. Kaj je značilno za skladovni računalnik?

- Za skladovni računalnik je značilno, da imajo pomnilnik v CPE realiziran v obliki sklada, zato so **brezoperandni.**
- Operacije se izvajajo nad operandi na vrhu sklada, ki jih ni potrebno navajati eksplicitno.
- Pri vsaki operaciji potrebujemo PUSH in POP.

15. Primer skladovnega računalnika.

Primer skladovnega računalnik: Atlas, računalniki podjetja Burroughs, HP 3000. Po letu 1980 jih ni več.

16. Število eksplicitnih operandov.

- Manjše število pomeni krajše ukaze in tudi manjšo moč ukazov.
- Večje število zahteva bolj zapleteno CPE.
- Na število eksplicitnih operandov vpliva tudi tip operacij, ki jih bo računalnik opravljal.
- Računalnike z m eksplicitnimi ukazi imenujemo ***m-operandni*** ali ***m-naslovni***, kar ne pomeni, da imajo vsi ukazi m operandov, najpomembnejši ukazi (ALE) imajo m operandov.

Imamo pet skupin:

1. **3+1 operandni:**
   - prvi trije operandi so za operacijo (npr. OP3 <- OP2 + OP1),
   - 4. pa za naslov naslednjega ukaza (namesto današnjega PC-ja).
   - Takšnih računalnikov ni več, bil pa je recimo EDVAC.
   - Niso imeli pomnilnika za shranjevanje operandov v CPE, uporabljali pa so GP s krožnim dostopom, zato so ga pohitrili z razporejanjem ukazov in operandov (pohitritev do 200x)

2. **3 operandni:**
   - po letu 1980 praktično vsi, ki imajo operande v registrih (load-store računalniki)
   - OP3 <- OP2 + OP1, PC <-PC + 1.
   - Ker se pojavi pomnilnik z naključnim dostopom, potreba po pametnem razvrščanju ukazov in operandov odpade.

3. **2 operandni:**
   - OP2 <- OP1 + OP2 (rezultat operacije shranimo v prostor, na katerem je eden od vhodnih operandov)
   - Za mnogo primerov velja, da rezultat ene operacije takoj uporabimo kot vhodni operand v naslednji, nato pa ga ne potrebujemo več -> ni potrebe po 3 operandih
   - OP1 in OP2 sta lahko v enem izmed registrov ali v pomnilniku (kot pri 3OP).
   - 1 ½ naslovni računalniki – pogost pojav: en operand v registru (krajši naslov) en v pomnilniku.
   - Do 1980 najpogostejša vrsta, ker opustitev enega eksplicitnega operanda pogosto ne vpliva na hitrost računanja.

4. **1 operandni:**
   - CPE ima 1 ali max. 2 akumulatorja za shranjevanje operandov.
   - AC <- AC + OP1; AC=akumulator, OP1=eksplicitni operand
   - PC <- PC + 1

5. **Brezoperandni (skladovni) računalniki:**
   - SKLAD(vrh) <- SKLAD(vrh) + SKLAD(vrh-1).
   - Operandov v ukazu ni treba eksplicitno navajati, ker se operacije izvajajo nad operandi na vrhu sklada -> krajši ukazi kot kjerkoli drugje
   - Potrebujemo pa 2 ukaza z enim eksplicitnim operandom – PUSH in POP

17. Kaj je značilno za 3+1 operandne računalnike?
Glej 16.

18. Lokacija operandov.
- Problem lokacije se pojavi večinoma samo pri ALE ukazih, saj pri drugih ukazih že sama narava ukaza določa izbiro.
- Poleg tega je ta problem omejen skoraj samo na 2 in 3-operandne računalnike, vendar je ravno teh največ.
- Operandi so lahko shranjeni v CPE, v GP ali v V/I krmilnikih.
- V veliki večini primerov se uporabljata samo prvi dve možnosti.

Ločimo torej tri variante:

1. Registrsko – registrski računalniki:
- vsi operandi ALE ukazov so v registrih CPE.
- Imenujemo jih tudi LOAD/STORE računalniki, ker je treba vsak operand prenesti iz in nazaj v pomnilnik.
- Imajo kratke, preproste ukaze.
- Čas izvrševanja ALE ukazov je vedno enak in ga lahko vnaprej določimo (dobro za paralelizem).
- Slabost je v tem, da je za rešitev istega problema potrebnih več ukazov, kot pri tistih računalnikih, ki imajo ALE ukaze lahko tudi v pomnilniku.
- sem spada večina 3-operandnih računalnikov

2. Registrsko – pomnilniški računalniki:
- Ed operand je v registru ali v pomnilniku, drugi pa so vedno registrih.
- V to množico spadajo računalniki, ki imajo ene ali dva akumulatorja in registrsko-registrskih ukazov nimajo – akumulatorski računalniki.
- Pri ALE operacijah lahko uporabljamo pomnilniške operande, ne da bi jih prej prenesli v registre.
- Ukazi so daljši, vendar jih je potrebno manj.
- Časa za izvajanje ne moremo naprej predvideti, saj je odvisen od lokacije operandov.
- Večina 2-operandnih računalnikov spada v to skupino

3. Pomnilniško – pomnilniški računalniki:
- vsak operand je lahko ali v pomnilniku ali v registru.
- So najbolj splošni in omogočajo veliko število rešitev pri istem problemu.
- Lahko delamo tudi brez uporabe registrov, vendar je to slabo.
- Ukazi so zapleteni.
- V to skupino spadajo t.i. CISC računalniki, med katerimi je posebno znana serija VAX

19. Kakšne načine naslavljanja poznamo? Obseg naslovov in uporaba.
Način naslavljanja je način podajanja operandov v pomnilniku. Lahko jih razdelimo v tri osnovne skupine:

1. **Takojšnje naslavljanje:**
   - operand je podan kar z vrednostjo, ki je del ukaza – dodatni dostop do pomnilnika ni potreben.
   - Temu operandu pravim takojšnji ali literal.
   - Običajno se označuje z znakom #.
   - Korist je največja pri aritmetičnih ukazih, primerjavah in v ukazih za prenos podatkov.
   - Uporabno za konstante.
2. **Neposredno naslavljanje:**
   - operand je podan z naslovom ?1 dodaten dostop do pomnilnika.
   - Dolg pomnilnik zahteva tudi dolge ukaze.
   - Če pomnilniški prostor povečamo – drugačna zgradba ukazov - ni kompatibilnosti za nazaj.
   - Naslov je lahko podan tudi samo z delom naslova.
   - Uporablja se za registrsko naslavljanje (operand je podan z naslovom registra v CPE) in za operande na konstantnih naslovih.
3. **Posredno naslavljanje:**
   - naslov pomnilniškega operanda v ukazu je podan preko neke druge vrednosti – ta je lahko:
   - Ta druga vrednost je lahko v:
     1. glavnem pomnilniku (**pomnilniško posredno naslavljanje**)
        - v ukazu je pomnilniški naslov lokacije na kateri je shranjen pomnilniški naslov operanda

- v primerjavi z neposrednim imamo en dodaten dostop do pomnilnika

2. ali v enem od registrov (**registrsko posredno naslavljanje ali relativno naslavljanje**)
   - v ukazu je naslov registra in t.i. ***odmik***
   - Iz vsebine registra in odmika se izračuna pomnilniški naslov operanda
   - Nima dodatnega dostopa do pomnilnika
   - spreminjanje naslova v registru je lažje kot spreminjanje naslova v pomnilniku
   - Mnogi računalniki zato pomnilniškega neposrednega naslavljanja sploh nimajo
   - Praktično vsi pa imajo registrsko posredno naslavljanje, ki ga lahko razdelimo v več skupin:

     3.1. Bazno naslavljanje (A=R2+D):
     - v ukazu sta podana naslov registra (bazni register) in odmik (variabilna dolžina).
     - Dolžina registra R2 je praviloma enaka ali daljša od dolžine pomnilniškega naslova

     3.2. Indeksno naslavljanje
     - Če je dolžina odmika enaka dolžini pomnilniškega naslova (obsega cel pomnilniški prostor), to imenujemo indeksno naslavljanje.
     - Primerno za dostop do elementov polja.

     3.3. Pred-dekrementno naslavljanje
     - je lahko bazno ali indeksno in avtomatsko zmanjšuje indeks,

     3.4. Po-inkrementno
     - pa avtomatsko zvečuje indeks.
     - Po in Pred-Dekrementno lahko skupaj tvorita strukturo sklad.

     3.5. Velikostno indeksno naslavljanje
     - namesto prištevanja (odštevanja) odmika uporablja množenje z odmikom.
     - Za dostop do elementov polja.

20. Kakšna je razlika med baznim in indeksnim naslavljanjem?

- Pri obeh imamo nek bazni register, ki mu prištejemo odmik in tako dobimo naslov operanda.
- Razlika je v tem, da je dolžina odmika pri indeksnem naslavljanju enaka dolžini pomnilniškega naslova (isto bitov)
  - to lahko dosežemo tudi tako, da kot odmik podamo register+odmik (dolžina tako izračunanega naslova je vedno enaka dolžini pomnilniškega naslova pri vsaki dolžini odmika).

21. Kaj je značilno za pozicijsko neodvisno naslavljanje. Kaj so pozicijsko neodvisni programi?

- Omogoča pozicijsko neodvisne programe → to so programi, ki se pravilno izvajajo na poljubnih naslovih.
- To lahko rešimo tudi v okviru navideznega pomnilnika, sicer pa tako, da uporabljamo le ***takojšnje*** (vrednost operanda ne sme biti naslov) in ***relativno naslavljanje*** (posredno registrsko). Glavno je, da program ne vsebuje ukazov, ki vsebujejo absolutne naslove.
- Pri relativnem naslavljanju moramo ob premestitvi programa samo na začetku spremeniti začetno vsebino prvega registra
- Za to je še posebej primerno indeksno naslavljanje, pri katerem je lahko en register namenjen samo pozicijski neodvisnosti.
- Nekateri računalniki omogočajo celo PC-relativno naslavljanje – indeksni register je kar PC.

22. Operacije.

Niso tako pomembne, ker lahko večino problemov rešimo z zelo osnovnimi operacijami. Razdelimo jih v več skupin:

1. Aritmetične in logične: osnovne računske operacije, Boolove operacije, pomiki.
2. Prenosi podatkov: load (pom->reg), store (reg->pom), push (pom ali reg->sklad), pop (sklad->reg ali pom), move (reg->reg ali pom->pom).
3. Kontrolne: pogojni skoki (beq), brezpogojni skoki (j), klici (call) in vrnitve iz procedur.
4. Operacije v plavajoči vejici: manjši računalniki jih nimajo, v CPE je posebna enota -> enota za operacije v plavajoči vejici.
5. Sistemske: spreminjajo parametre delovanja računalnika (stop).
6. Vhodno/izhodne operacije: ukazi za prenos med GP ali CPE in V/I napravo.

23. Vrsta in dolžina operandov.

- Bit
- Znak: 8 bitov (zanki predstavljeni z abecedo ASCII ali EBCDIC).
- Celo število: 16 ali 32 bitov, predznačena števila v fiksni vejici (dvojiški komplement).
- Realno število: 32 bitov za enojno in 64 bitov za dvojno natančnost v plavajoči vejici (IEEE 754).
- Desetiško število: niz 8-bitnih znakov (pakirano (dve BCD številki v vsakem 8-bitnem znaku) ali nepakirano (po en ASCII ali EBCDIC znak v vsakem 8-bitnem znaku)).

Kako je podan tip operanda v ukazu?

- V operacijski kodi (tip operanda določen z ukazom)
- Z metabiti (vsakemu operandu dodamo nekaj bitov ki povejo, za kakšen operand gre)

24. Problem poravnanosti.

- Sestavljeni operandi: iz več pomnilniških besed.
- Dostop do npr. 8 8-bitnih besed naenkrat je narejen kot dostop do recimo 8 paralelno delujočih pomnilnikov.
- Pri n-bitov dolgem naslovu spodnji 3 biti naslova določajo, v katerem od njih je neka pomnilniška beseda, zgornjih n-3 bitov pa naslov besede znotraj vsakega pomnilnika.
- To pomeni, da je istočasen dostop do S besed nekega operanda možen samo, če je naslov A deljiv z S. Če je deljiv pomeni, da je sestavljeni operand poravnan
- Če to ni izpolnjeno, je do takšnega operanda potrebnih več pomnilniških dostopov, kar seveda upočasni delovanje

Načina za shranjevanje sestavljenih pomnilniških operandov:

- **Pravilo debelega konca –** naslov sestavljenega operanda je enak naslovu besede, ki vsebuje najtežji (največ vreden) del operanda, preostale besede vsebujejo po padajoči teži urejene dele operanda
- **Pravilo tankega konca –** naslov sestavljenega operanda je enak naslovu besede, ki vsebuje najlažji (najmanj vreden) del operanda, preostale besede vsebujejo po naraščajoči teži urejene dele operanda

25. Zgradba ukazov (spremenljiva, fiksna, hibridno)?

Zgradba ukazov je podana s formatom ukaza. Ta je sestavljen iz ***operacijske kode***, ***načina naslavljanja*** in ***naslovnega polja*** (operand).

- **Fiksna dolžina:** število eksplicitnih operandov vedno enako, način naslavljanja pa je vsebovan v operacijski kodi. Število formatov je majhno -> lažje realizacija, večja hitrost -> tipično RISC (ARM, HIP – dolžina ukazov je 32 bitov)
- **Spremenljiva dolžina:** poljubno število eksplicitnih operandov, vsak je lahko podan z enim od velikega števila načinov naslavljanja -> veliko formatov (tudi količina dela, ki ga opravi posamezen ukaz, je zelo različna). Z uporabo kratkih formatov za bolj pogoste ukaze se želi doseči, da so programi kratki …(število operandov je vsebovano v operacijski kodi.) (VAX)
- **Hibridni način:** ga dobimo, če na eni strani zmanjšamo spremenljivost dolžine in moči ukazov ter na drugi strani uporabimo več dolžin ukazov (IBM 370)

Dolžina ukaza je odvisna od:

- dolžine pomnilniške besede (dobro je, da je dolžina ukaza mnogokratnik dolžine pomn. besede
- števila eksplicitnih operandov v ukazu (za vsak eksplicitni operand mora biti v ukazu podano, kje je shranjen)
- vrste in števila registrov v CPE (splošno namenski in …)
- dolžine pomnilniškega naslova (pri uporabi neposrednega naslavljanja mora biti v ukazu pomnilniški naslov)

26. Kaj je ortogonalnost ukazov? Kje pride v poštev?.

- Pomeni medsebojno neodvisnost parametrov, ki jih ukaz vsebuje:
  - informacija o operaciji je neodvisna od informacije o operandih
  - informacija o enem operandu je neodvisna od informacije o ostalih operandih
- Za vsak operand lahko uporabimo vse načine naslavljanja in vse dolžine operandov.
  - S tem je lažje programiranje – ker ni izjem, so pravila za podajanje operandov preprosta in enaka pri vseh ukazih.
  - Po drugi strani pa odstopanje od ortogonalnosti poenostavi zgradbo CPE in jo naredi hitrejšo.
- **Pri RISC računalnikih (registrsko - registrski) je ortogonalnost ukazov izgubila svoj pomen:**
  - če so vsi operandi lahko samo v registrih, obstaja za njihovo podajanje še itak samo ena možnost.
  - Ortogonalnost je torej samoumevna in trivialna.
  - Zakomplicira zgradbo CPE in jo upočasni

27. CISC vs. RISC dilema. Prednosti, slabosti. Cevovod.

- Po mnenju nekaterih so boljši računalniki z veliki številom ukazov **CISC računalniki**
- Po mnenju drugih pa je prednost na strani računalnikov z majhnim številom ukazov **RISC računalniki**

1. Razlogi za povečanje števila ukazov-CISC:

1.1. Semantični prepad

- z njim označujemo razliko med računalnikom, kot ga vidi programer v višjem programskem jeziku in med tistim, kar vidi programer v strojnem jeziku
- Včasih so ta prepad označevali kot krivca za prevelike in počasne programe ter za zapletenost prevajalnikov
- Računalniki so zato v razvoju imeli čedalje večje število ukazov, kljub temu da se velik del ukazov rabi zelo redko.

1.2. Mikroprogramiranje

- Omogoča zelo preprosto dodajanje in spreminjanje ukazov
  - Poskus zmanjševanja semantičnega prepada
- Dekodiranje mikroukazov ima za posledico nekoliko počasnejše izvajanje vseh ukazov

1.3. Razmerje med hitrostjo glavnega pomnilnika in CPE

- Jemanje ukazov iz glavnega pomnilnika je bilo počasno.
- En kompleksen ukaz se je zato izvršil hitreje kot ekvivalentno zaporedje preprostih ukazov.