

Prevajalnik (compiler) pretvori izvirno kodo v zbirni jezik in zbirnik pretvori zbirni jezik v strojno kodo.

`and x5, x6, 0x ff7f` se uporablja za brisanje (reset) bita, saj je le en bit 0 in vredlo 1, operacija je pa and.

`ori r1, r2, #0x00008000` se uporablja za nastavljanje (set) bita.

gledi postopek reševanja naloge o največjem številu določenega nestandardnega formata IEEE-754 plavajoče vejice (floating-point).

V nekem nestandardnem formatu plavajoče vejice imamo en bit za predznak, 3 bite za eksponent in 4 bite za mantiso (mantissa) odmika eksponenta pa je določen na enak način kot pri IEEE-754. Katero je največje normirano število?

1 bit ... predznak

3 bit ... biased exponent

4 bit ... mantissa

Odmik enak kot pri IEEE-754.

1. Izračun odmika (offset) - pri IEEE-754 je mu enec bias.

$$\text{bias} = 2^{n-1} - 1$$

$$= 2^3 - 1$$

$$= 4 - 1$$

$$= 3$$

2. Izračun eksponenta (exponent)

E ... biased exponent

e ... exponent

$$e = E - \text{bias}$$

$$= 6 - 3$$

$$= 3$$

Pozornitveno je vrednost NaN(α) neizvenit
 $E_{\max} = 111_{(2)} = 7_{(10)}$
 $E = 110_{(2)} = 6$

3. Izračun mantise (mantissa)

V naravnih pixe 4 bit. V mantisi je bit pred večico redno 1, zato se mantissa = 0,1111 naredi: 1,1111. 1 pred večico je implicitna.

4. Vse združim.

$$1,1111 \cdot 2^3 = 1111,1 = 15,5$$

če bi naloga zahtevala največjo nenormirano vrednost, bi bil E=000 in mantissa = 1111, enako kot prej, vendar ne bi bilo implicitne 1, zato 0,1111.

$$E=000 \text{ mi pomembno}$$

$$\text{Rezultat: } 0,1111 \cdot 2^{-2} = 0,001111_{(2)}$$

$$e = 1 - \text{bias}$$

$$= 0,234375$$

$$e = 1 - 3 =$$

$$= -2$$

je pomembno

CPE se deli na dva dela, kontrolno enoto (control unit), ki posilja signale (o vrstnem redu opravljanja ukazov in delov ukazov), in podatkovno enoto (datapath), ki vključuje ALE in registre.

Enocikelni procesor ob vsaki periodi ure gre skozi vsa stanja, zato je diagram prehajanja stanj pri njem neuporaben, perioda ure je pa dolga, kolikor je dolg najdaljši ukaz.

Večcikelni procesor ima za vsak del ukaza drugačne dolžine periode ure, zato je vsak ukaz dolg več periodnih ur, diagram prehajanja stanj pa koristen, naj vsako stanje ima svojo

urino periodo in procesor ne gre vedno nesasi vna stanja.

Kontrolna enota vsebuje končni avtomat (FSM - Finite State Machine).

MIPS (Millions of Instructions Per Second) ni vedno merodajen - zanesljiv (ni merodajen - informacija ni natančna) -, saj različni ukazi (pri različnih arhitekturah) trajajo različno dolgo. Četudi je bolj natančen pri enocikelnih procesorjih, je preč ker natančen, da bi bil zanesljiv, ne bolj je pa nezanesljiv pri večcikelnih procesorjih (multi-cycle processor) in cevovodnem procesorju (pipeline processor).

Ukazi, ki se izstavi (izbere) za opravitev (execution)
 v danem trenutku večiznstavitevnega procesorja je odvisno od vrste večiznstavitevnega procesorja (multi-issue processor). Poznam dve vrsti, statično večiznstavitevnost (VLIW - Very Long Instruction Word) in dinamično izstavitevnost (superscalar processor), ki izbirata na različen način.

Večiznstaviti procesor (multi-issue processor) je procesor, ki ima več cevovodov (pipeline), ki delujejo hkrati.

Ni mogoče reči zagotovo, ali je enocikelni procesor hitrejš od večcikelnega (zaradi specifikacij Logike signalov...).

Procesorji imajo običajno več ukazov load kot store, saj programi običajno imajo več ukazov za branje iz pomnilnika kot za pisanje varj.

Ygredi postopek računanja števila izvedbo rekurzivne funkcije.

Na naslednji strani.

Uklop se začne na naslovu 0x00014000 in narašča po nadajočih naslovih do naslova 0x00011000. Največ kolikokrat se lahko izvede rekurzivna funkcija, če en okvir zavzame 64 bajtov?

$$\text{Size} = 0x00014000 - 0x00011000 = 0x00003000 \\ = 3 \cdot 16^3 = 3 \cdot (2^4)^3 = 3 \cdot 2^{12}$$

$$x = \frac{\text{Size}}{64} = \frac{3 \cdot 2^{12}}{2^6} = 3 \cdot 2^6 = 3 \cdot 64 = 192$$

O: Rekurzivna funkcija se izvede 192-krat.

Uklop tažlaga naloge ojalt ukazu.

Ukaj naredi ukaz jalt ra, a0, 0x100?

jalt $ra, rx1, offset$
return address immediate value
Part of where PC jumps to.

PC jumps to: $rx1 + offset$

PC (Program Counter) širani naslov ukaza, ki ga bo naslednjega izvedel (execute).

jalt ra, a0, 0x100

a. $a0 \leftarrow 0x100$ $PC \leftarrow ra$

b. $PC \leftarrow ra$; $a0 \leftarrow 0x100$

c. $PC \leftarrow a0 + 0x100$; $x10 \leftarrow PC + 4$

d. $x1 \leftarrow PC + 4$; $PC \leftarrow x10 + 0x100$

Pravilen odgovor: d., saj se $x1 = ra$ shtani naslov naslednjega

ukaza v programu glede na mesto, kjer je PC (Program Counter) trenutno. Dolžina ukaza je 4 byte (word), zato +4, $x_1 \leftarrow PC + 4$. $a_0 = x_{10}$, mesto, na katerega bo program skočil je $rs1 + offset$, zato $x_{10} + 0x100$, $PC \leftarrow x_{10} + 0x100$.

Ukaz not x_1, x_2 , kjer x_1 postane eniški komplement ed x_2 = najmanj ukazi implementiram = xor $x_1, x_2, -1$.

Ponati eniški komplement

$$\begin{array}{rcl} x_1 = 15 = 00\dots 01111 & x_1 = 00\dots 01111 \\ x_2 = 11\dots 10000 & -1 = 11\dots 11111 \\ \hline xor & \hline \\ & x_2 = 11\dots 10000 \end{array}$$

Ukaz bLe (branch if less or equal) oblike
bLe x_1, x_2, imm bi = najmanj ukazi naredil na naslednji način.

bgt $x_1, x_2, skip$ # If $x_1 > x_2$, skip the jump
jal x_0, imm # Mesto, na katere skoči je imm
skip:

pomehi, če $x_2 > x_1$, kar
je funkcionalno enako
 $x_1 \leq x_2$, potem jump.

Pozoren nem na naslednji primer, kjer haloga spravi, najmanj koliko bitov bi potrebovali za zapis določenega ukaza.

Yatko najmanj bitov bi potrebovali za zapis aritmetičnega ukaza
» 3 registerjski operandi, če je vseh ukazov (= vseh ukazov [trenaj dodatek]) - 85, procesor pa ima 64 registrov.

3 registrski operandi

$$\text{rtUkazov} = 85$$

$$\text{rtRegistr} = 64$$

Učem prvo potenco 2 večjo od rtUkazov in prvo potenco 2 večjo od rtRegistr, saj potrebujem število bitnih mest, s katerim bom lahko zapisal poljubni ukaz in poljubno število registerja (od 64 prisotnih v procesorju).

Danem procesorju v nalogi.

$$128 \geq 85$$

$$64 \geq 64$$

$$2^7 > 85$$

$$2^6 \geq 64$$

$$\log_2(2^7) = 7 \log_2 2 = 7(1) = 7 \quad \log_2(2^6) = 6 \log_2 2 = 6(1) = 6$$

Vrednosti sta standardni za poljubni ukaz (7) in register (6) v procesorju.

V mojem primeru je število operandov v ukazu 3, zato

$$\text{register} = 3 \cdot 6 = 18 \text{ bit}$$

$$\text{vrstaUkaza} = 7 \text{ bit}$$

$$\begin{aligned} \text{VelikostUkaza} &= \text{vrstaUkaza} + \text{register} = \\ &= 7 + 18 \\ &= 25 \end{aligned}$$

O. Potrebovali bi najmanj 25 bitov.

Iledi primer naloge računanja verjetnosti zadetka.

Učilnika je verjetnost zadetka, če povprečni dostopni čas je upravo 2ns-predponnikha znaš 5ns, brez njega pa 60ns?

zadetek v predpomnilnik: $2\text{ns} \cdot t_{\text{cache}}$

zadetek brez predpomnilnika (v pomnilnik): $60\text{ns} \cdot t_{\text{mem}}$
poravnati čas: 5ns (t) \rightarrow

Hverjetnost zadetka v predpomnilniku:

$$t = H \cdot t_{\text{cache}} + (1-H) \cdot t_{\text{mem}}$$

$$t = H t_{\text{cache}} + t_{\text{mem}} - H t_{\text{mem}}$$

$$t - t_{\text{mem}} = H (t_{\text{cache}} - t_{\text{mem}}) / (t_{\text{cache}} - t_{\text{mem}})$$

$$\frac{t - t_{\text{mem}}}{t_{\text{cache}} - t_{\text{mem}}} = H$$

$$H = \frac{5\text{ns} - 60\text{ns}}{2\text{ns} - 60\text{ns}} = \frac{-55\text{ns}}{-58\text{ns}} = 0,948 = 94,8\% \approx 95\%$$

Pomnilniško preslikan vhod-izhod pomeni, da procesor pri branju/pisanju - read/write - uporablja isti mehanizem za branje/pisanje - read/write - v register in branje/pisanje - read/write - v pomnilnik, zato v oba primerih uporablja enak ukaz Load/store. Vendar ne v načalah pri ARM zahteva poznavanje vrstoka opisane značilnosti, kar je, da so registri vhodno-izhodnih naprav v pomnilniškem naslednjem prostoru, značilno je za ARM in RISC-V procesor. Intel procesorji imajo ločeni del (element v arhitekturi) za naslove registrrov in zato uporabljajo drugačne ukaze za branje/pisanje - read/write - registr in branje/pisanje - read/write - v spomin, uporabljajo tudi ločene ukaze, npr. IN/OUT (morda ne velja IN/OUT specifično za Intel procesor).

Vrste pomnilnika so SRAM, DRAM in trdi disk, ki je lahko Flash, EEPROM, EPROM.

SRAM sta cache in registri, DRAM je RAM

(v vsakdanjem smislu) in Flash, EEPROM in EPROM so trdi disk. Utratice pomenijo SRAM - Static Random Access Memory-, DRAM - Dynamic Random Access Memory-, Flash, EEPROM - Electrically Erasable Programmable Read-Only Memory-, in EPROM - Erasable Programmable Read-Only Memory-. Flash je podvrsta EEPROM.

Čas dostopa je pri SRAM 0,5 ns do 2,5 ns, pri DRAM 50 ns (in 60 ns je dolžina cikla) in pri ostalih je približno 100 000-krat več kot pri DRAM.

Čas cikla t_c je $t_c = t_a + \text{čakanje}$, kjer je ta čas dostopa, hitrost dostopa pa je po $b_a = \frac{1}{t_c}$.

Destruktivno branje je vrsta branja (read) podatka, ki ob branju izbriše podatek in ga nato ponano napiše (write) na mestu pred brisanjem, značilno je za DRAM. Ostali pomnilniki nimajo značilnosti destruktivnega branja.

Pomnilnike delimo na ključne (volatile), ki ob izklopu iz električne izgubijo naboј in zato vse podatke, ki so v njih shranjeni, medije spadajo SRAM in DRAM, in neključne (non-volatile), ki ob izklopu iz električne ne izgubijo naboј in zato tudi ne izgubijo podatkov, ki so v njih shranjeni, medije spadajo EPROM, EEPROM in Flash. Kar omogoča neključnim (non-volatile) pomnilnikom, da naboј ne izgubi, so plavajoča vrata (floating gate), ki naboј shranjuje.

SRAM celico sestavlja 6 tranzistorjev, ki tvorijo zapah (latch), električno strukturo -vezje-, ki lahko drži bit informacije (0 ali 1) stalno, dokler je napajanje prisotno, zato ko napajanja ni, izgubi vse podatke. DRAM celico pa sestavlja 1 tranzistor in 1

kondenzator.

Gledi se želim razlik med statičnimi (SRAM) in dinamičnimi (DRAM) pomnilniki RAM.

DRAM	SRAM
destruktivno branje	nima destruktivnega branja
ima refresh	nima refresh
rešavlja 1 kondenzator in 1 tranzistor	rešavlja 6 tranzistorjev
počasnejši	hitrejši
cenejši na bit	dražji na bit
uporaba kot glavnih nizvodnikov RAM	uporaba kot predpotrošilnik

CPE in glavni pomnilnik (GP) komunicirata z uporabo teh vrst signalov, rešaberi so enosmerni (one-way), drugi dvosmerni (two-way).

(control signals)

Vrste signalov so kontrolni -krmilni - signali^I preko kontrolnega vodila, ki organizirajo procese pri izvrševanju ukazov. In potreba dvosmerno, naslavljeni signali (address signals) preko naslovnega vodila (data bus), ki poveže glavnemu pomnilniku - DRAM - je glavni pomnilnik -, do katerega naslovu želi CPE dostopati, poteka enosmerno od CPE do DRAM, in podatkovni signali (data signals) preko podatkovnega vodila (data bus), ki prenosi vrednosti in potreba dvosmerno.

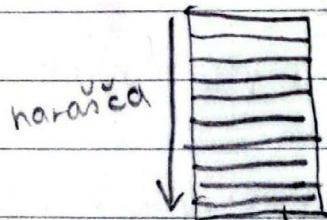
CPE, ko želi prebrati podatek iz DRAM, mu pošlje naslovni signal (address signal) preko naslovnega vodila (address bus), DRAM nato prebere podatek na ponarem naslovu

in poilje CPE podatkovni signal preko podatkovnega vodila, ki je prebrana vrednost v obliki električne napetosti (0 ali 1) in taka. Ko ga CPE prejme, pozna vrednost na naslovu.

Ko želi CPE zapisati (write) vrednost na določen naslov poilje naslovni signal (address signal) preko naslovnega vodila (Address bus) in podatkovni signal (data signal) preko podatkovnega vodila (data bus), s katero pole vrednost, ki jo hoče napisati (write) proti DRAM. DRAM sprejme naslov in vrednost in nato vpiše vrednost na želeni naslov.

Naslovni register (MAR - Memory Address Register) hrani naslovne podatke, podatkovni register (MDR - Memory Data Register -) hrani vrednosti in kontrolna enota - KE - (CU - Control Unit -) hrani kontrolne signale, vse tri hranijo vrednosti, ki bodo bile poslane med CPE in DRAM (običajno je DRAM izbran za glavni pomnilnik - GP -) v obliki signalov (signal) čez svoje vodilo (bus).

Pri RISC-V sklad narača v smeri padačih naslovov, zato skladovni kazalec (stack pointer) kaže na zadnji podatek - podatek, ki je na dnu sklada -.



Zadnji vstavljeni podatek je na dnu sklada, skladovni kazalec (stack pointer) pa vedno kaže na zadnji vstavljen podatek.

Množenje registrja $x1 \times 5$ bi najlažje implementirali na naslednji način.

$sLLi \times 2, \times 1, 2 \# \text{Množenje} \geq 2^2 = 4$

add $\times 2, \times 1, \times 2 \# \text{Prvič enkrat začetna vrednost,}$
 $\# ki je shranjena v x1.$

Sledi primer naloge računanje časa izvajanja programa z uporabo cevovoda.

Umao 3-stopenjski cevovod, ki izvaja 3 operacije O1 (traja 3 ns), O2 (4 ns) in O3 (2 ns). V kolika času traja izvedba 5 ukazov (do konca zadnjega ukaza), če izberemo največjo možno frekvenco ure?

$$\text{čas} = (k-1+N) \cdot t_{\text{cikel}}$$

k ... k-stopenjski cevovod

N ... število ukazov

t_{cikel} ... čas izvajanja urine periode

$$t_{\text{cikel}} = \max(t_{O1}, t_{O2}, t_{O3}) = \max(3 \text{ ns}, 4 \text{ ns}, 2 \text{ ns}) = 4 \text{ ns}$$

$$\text{čas} = (3-1+5) \cdot 4 \text{ ns}$$

$$= 7 \cdot 4 \text{ ns}$$

$$= 28 \text{ ns}$$

ALU je kombinacijsko vezje - vezje, ki ne shranjuje vrednosti, je brez stanja (in zato tudi brez pomnilnika, naš stanja shranjuje informacije) in ure (ne uporablja urine periode osiroma jo ignorira) - (combinational circuit).

DRAM je vedno sifronsko vezje, saj ima stanja (in zato shranjuje vrednosti, naš je vrsta pomnilnika), in ure (pisanje - write - poteka v urini periodi).

SRAM je lahko sifronsko ali asifronsko vezje.

Slledi primer računske naloge, ki je podobna nalogi na 6.

strani zapiskov, le, da v nedanji iščem velikost takojšnjega operanda (immediate value size)

32-bitni ukaz ima 2 registrska operanda in 1 takojšnji operand, zapisan v $2^7 K$. Vseh ukazov je 72, procesor pa ima 128 registrs.

Velikina je največja vrednost takojšnjega registra?

$$\log_2 128 = \log_2 (2^7) = 7 \log_2 2 = 7(1) = 7$$

$$2^7 > 128$$

Velikost posameznega registra v bitih.

$$2^7 > 72$$

$$\log_2 (128) = \log_2 (2^7) = 7 \log_2 2 = 7(1) = 7 \rightarrow \text{Velikost opcode posameznega ukaza. v bitih.}$$

$$2^7 > 72$$

$$2 \text{ registrska operanda, zato } 2 \cdot 7 = 14$$

$$32 - 14 - 7 = 32 - 21 = 11$$

$$\text{Zaradi } 2^7 K - 2^{11-1} - 1 = 2^{11-1} - 1 = 2^{10} - 1 = 1023$$

O. Največja vrednost takojšnjega registra je 1023.

Uredi primer naloge računanja pohitritev vzorednega računanja z uporabo Amdahlovega zakona.

Velikina je pohitritev vzorednega računanja, če je zaporedni del ř programa 20%, vzoredni del ř programa pa lahko teče na 16 procesorjih hkrati?

zaporedni del ř... sequential part ... λ

parallelni del ř... parallel part ... μ

sterilo procesorjev... number of processors... N

$$x = \frac{1}{\lambda + \frac{\mu}{N}}$$

$$\alpha + p = 1$$

$$\alpha = 20\% = 0,20$$

$$x = \frac{1}{0,20 + \frac{0,80}{16}}$$

$$p = 1 - \alpha$$

$$p = 1 - 0,20$$

$$p = 0,80$$

$$x = \frac{1}{0,20 + 0,05}$$

$$x = \frac{1}{0,25}$$

$$x = 4$$

O: Pohitritev vzporednega računanja je 4-kratna.

V nalogi ne uporablja, vendar je mogoče izračunati tudi največjo možno pohitritev (teoretično pohitritev) vzporednega računanja.

$$N \rightarrow \infty \quad x = \frac{1}{0,20 + \frac{0,80}{\infty}} = \frac{1}{0,20 + 0} = \frac{1}{0,20} = 5$$

O: Največja možna pohitritev (teoretična pohitritev) vzporednega računanja je 5-kratna pohitritev.

Gledi primer naloge računanja povprečnega števila čakalnih period pri izvajanjih programa.

V ukaznem predporavnilniku je zgrisitvena kozen 50 urinih period, verjetnost zadetka pa 0,98. Koliko bo v povprečju čakalnih period pri izvajanjih programa ≈ 200 ukaz?

število čakalnih period na ukaz... all cycles per instruction ... ≈

verjetnost zadetka = 0,98, kozen zgrisitve = 50

verjetnost zgrisitve + verjetnost zadetka = 1

verjetnost zgrisitve = 1 - verjetnost zadetka

verjetnost zgrisitve = 1 - 0,98 = 0,02

$z = \text{verjetnost zgrisitve} \cdot \text{kozen zgrisitve}$

$\approx 0,02 \cdot 50 = 1$

≈ 1

$\text{st Cakalnih Period} = \text{st Ukazov}$

$\text{st Cakalnih Period} = 200 \cdot 4$

$\text{st Cakalnih Period} = 200$

Verjetnost zadeшка \propto večanjem velikosti bloka
najprej raste, nato začne padati.

I sledi še en primer računanje nistanardne
plavajoče vejice (floating-point) normiranega števila, točkat
najmanjšega normiranega števila.

V nekem nistanardnem formatu plavajoče vejice imamo en bit
za predznak, 3 biti za eksponent in 4 biti za mantiso, odmik
eksponenta pa je določen na enak način kot IEEE-754, karor tudi
de normirana števila, neskončnosti in NaN. V katero je najmanjše
normirano število (po absolutni vrednosti)?

E: 001-110, 000... normirano število, 111... neskončnost/NaN, v bit
najmanjša mantisa: 0000

najmanjši E: 001

$$1,0000 \cdot 2^{-2} = 0,0100 = 0,25 = \frac{1}{4}$$

$$e = E - 3$$

$$e = 1 - 3$$

$$e = -2$$

Najmanjše normirano število
v danem IEEE-754.

Procesorji imajo v naboru ukazov več ukazov load
kot ukazov store, saj ukazi load, ki halagajo v register operande,
ki so manjši od velikosti registra, lahko razširjajo nizko ali
predznak, ukazi store pa nesesar ne razširjajo. Ukazi load v
RISC-V so lb, lhu, lh, lhu, lw, - morda tudi lwu-, medtem ko
so store sb, sh, sw.

unsigned

Edini zapis, ki ima dve predstavitvi za nčlo je eniki komplement ($1'K$)

Logična funkcija vsote enobitnega polnega številkovnika je $s = a \oplus b \oplus c$, ozkoma z drugo označbo za XOR $s = a \oplus b \oplus c$

a, b, c ... vhod (input)

s ... sum

$s = a \oplus b \oplus c$ pomeni, da če je sodo število enic, bo izhod 0, če pa liko število enic, bo izhod 1.

$$s = 1 \oplus 1 \oplus 1 = 1$$

$$s = 1 \oplus 0 \oplus 0 = 1$$

$$s = 0 \oplus 1 \oplus 1 = 0$$

$$s = 0 \oplus 0 \oplus 0 = 0$$

Preliv se pri dvojškem komplementu ($2'K$) pojavi, ko ima vsota drugačen predznak kot števili.

Desnormirana števila so števila v plavajoči vejici (floating-point), ki imajo $E=0$ (vse bite eksponenta imajo 0, pri tem je E biased exponent in e exponent, $e=E-127$ bi bila enačba naprimer) in nimajo implicitne 1, temveč 0 (pred vejico je 0 namesto 1), 1,011 je normirano število, medtem ko je 0,011 desnormirano. Uporabljajo se za zapis jih računanje z vrednostmi, ki so dolgi, majhni, da jih v normirani obliki ne bi mogli zapisati (saj bi imeli vrednost 0-neničelni biti, bi se izgubili -).

Predpomnilniki so pomembni pomnilniki, ki skrajšajo časovne dolge in pospešijo dostop do podatkov.

Verjetnost konfliktnih zgradi I izračunjam najbolje - pri SRAM (predpomnilnik) - \rightarrow povečanjem stopnic

asociativnosti, saj večja asociativnost omogoča, da se več blokov lečko nahaja v istem setu predpomnilnika).

Verjetnost kapacitetnih zgrešitev (capacity misses) najbolje = manjšam \rightarrow povečanjem velikosti predpomnilnika (, saj ne pojavijo zaradi premajhe velikosti predpomnilnika, kar povzroči, da podatki ene druge izringe iz predpomnilnika, zato je potrebno, da se izrinjeni podatki ponovno naložijo vam - tedaj pride do zgrešitve) - **SRAM**.

Cevovod ni vrsta procesorja, temveč je tehnika izvajanja ukazov, ki je več ukazov izvršuje librati (ki se uporablja pri veččinkeljnih procesorjih).

Enčinkelni in veččinkelni procesor pa sta vrsta procesorja.

Zaporedni dostop (serial access) je način branja podatkov, kjer je čas dostopa odvisen od prejšnjega naslova. Uporabljaj se pri kartah. (primer na Powerpoint-u je magnetni trak).

Ukrožni dostop (rotational access) je način branja podatkov, ki temelji na zaporednem dostopu, le, da je čas dostopa v povprečju $\frac{1}{2}$ periode vrtenja, primer sta DVD in trdi disk (hard drive).

Kombinacija zaporednega in krožnega dostopa je način dostopa, poglej Powerpoint-, ki je hitrejši od zaporednega dostopa in krožnega dostopa.

Rafalni ali eksplosivni način, kjer pride do več zaporednih dostopov do pomnilnika z uporabo istega naslova, je primer zaporednega dostopa (serial access). Če bi imel podatke na naslovih 0x1000, 0x1001, 0x1002, 0x1003 bi moral le 0x1000, vendar bi prišli vti 4.

CAM (Content Addressable Memory) je vel vrst, ki se običajno uporablja v mreži za svoj namen, je vrsta pomnilnika,

ki hrani še podatke glede na vsebine - asociativno -, običajno je zato njegova potreba energije večja in je manjše velikosti, saj stane veliko glede na bit), je zelo hiter.

SRAM, DRAM in Flash so polprevodniški - solid state - pomnilniki, pri katerih, ker nimajo gibljivih delcev, pride do manj tehničnih napak.

Napake pri shranjenju (ne shranjevanju) podatkov se delijo na mehke napake (soft errors), ki so manjše napake, ki ne nastanejo zaradi trajne napake na hardware-u, in trde napake (hard error), ki so večje napake, ki nastanejo zaradi trajnih napak na hardware-u.

Za zaznavanje in popravljanje mehkih napak se uporabljajo kode za detekcijo in korekcijo napak v obliki dodatnih bitov.

MIMD (Multiple Instruction stream, Multiple Data stream) delim na tesno povezane (shared memory), ki ga je lažje sprogramirati, vendar "scales poorly", in rahlo povezani (distributed memory), ki ga je težje sprogramirati, in "scales well".
Tesno povezani (shared memory) imajo skupen pomnilnik, rahlo povezani (distributed memory) pa so povezani preko vhodno/izhodnih cnot.

SPMD (Single Program - Multiple Data) - glej Powerpoint Amdahlov zakon -.

Gustafsonov zakon nadgrajuje Amdahlov zakon, tako da trdi, da

Ostalo bi rekel, da je le ne teorija → Powerpoint-ov.

Tomasulov algoritem odpravi nevernosti, ki izvrajo iz imenskih odrivnosti (WAR in WAW) brez preimenovanja registerov.

Imenske odrivnosti (WAR in WAW) lahko odpravim brez Tomasulovega algoritma s preimenovanjem registerov, če imam na voljo dodatne registre.

Kontrolno enoto CPE se lahko realizira na dva načina, s rdlo ozičeno logiko (hard-wired logic) in mikroprogramiranjem.

Rdlo ozičena logika (hard-wired logic) omogoča hitrejše izvrševanje ukazov, vendar ukazov ni mogoče spremniti (ali nadgrajevati) brez fizičnega posega.

Mikroprogramiranje (microprogramming) omogoča počasnejše izvrševanje ukazov, ampak je ukaze mogoče spremniti (ali nadgrajevati) brez fizičnega posega.