Operacijski sistemi ISD/RST

doc. dr Panče Panov

Predavanje 2

25. 2. 2024

Vsebina za danes

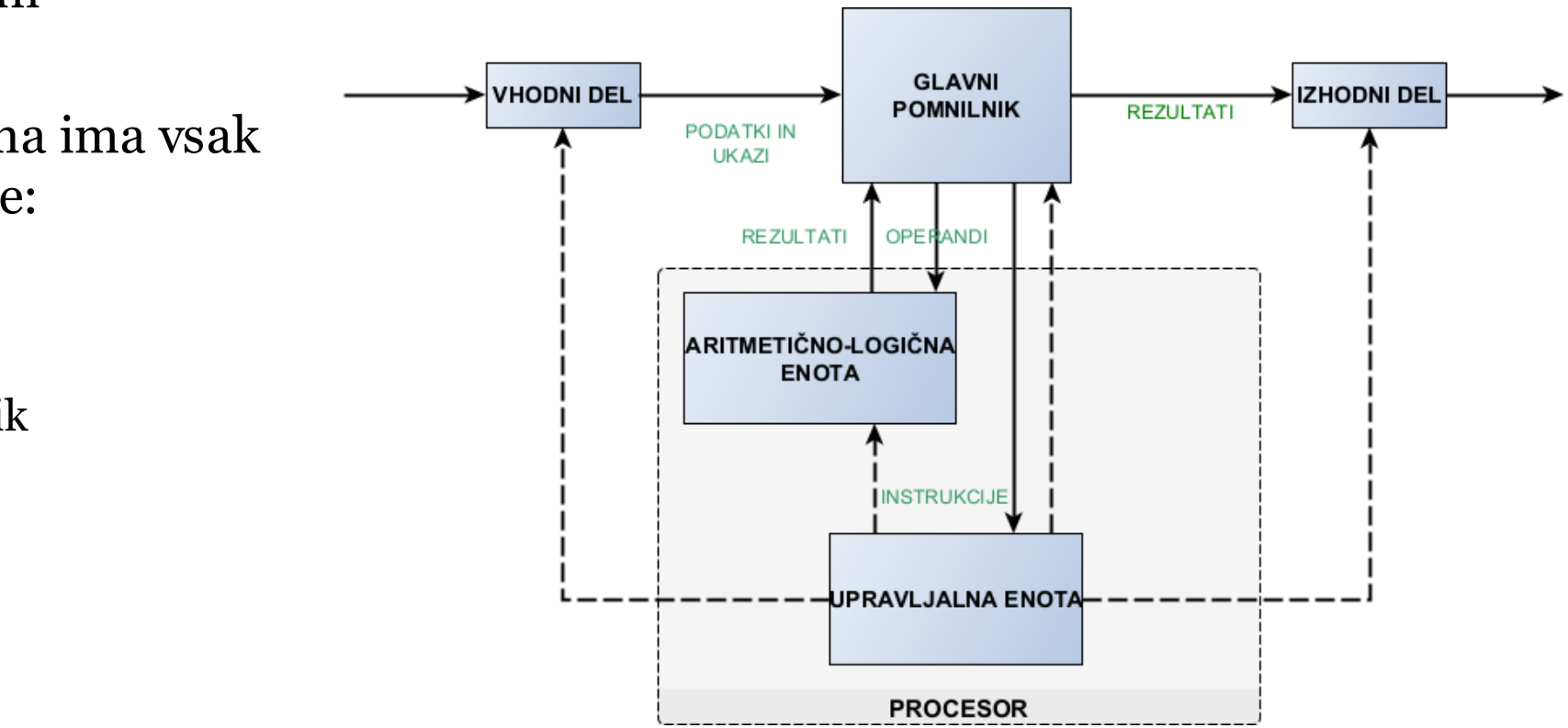
- Von Neumanov model računalnika
- Organizacija in delovanje sodobnih računalniških sistemov
- Življenski cikel enega programa znotraj računalniškega sistema

VON NEUMANOV MODEL RAČUNALNIKA

Funkcijski model računalnika (1)

- Sodobni računalniški sistemi še vedno temeljijo na konceptualnem modelu računalnika, ki ga je leta 1945 opisal John von Neumann

- Po modelu von Neumanna ima vsak računalnik naslednje dele:
  - vhodni del
  - izhodni del
  - delovni ali glavni pomnilnik
  - aritmetično-logična enota
  - upravljalna enota



Funkcijski model računalnika (2)

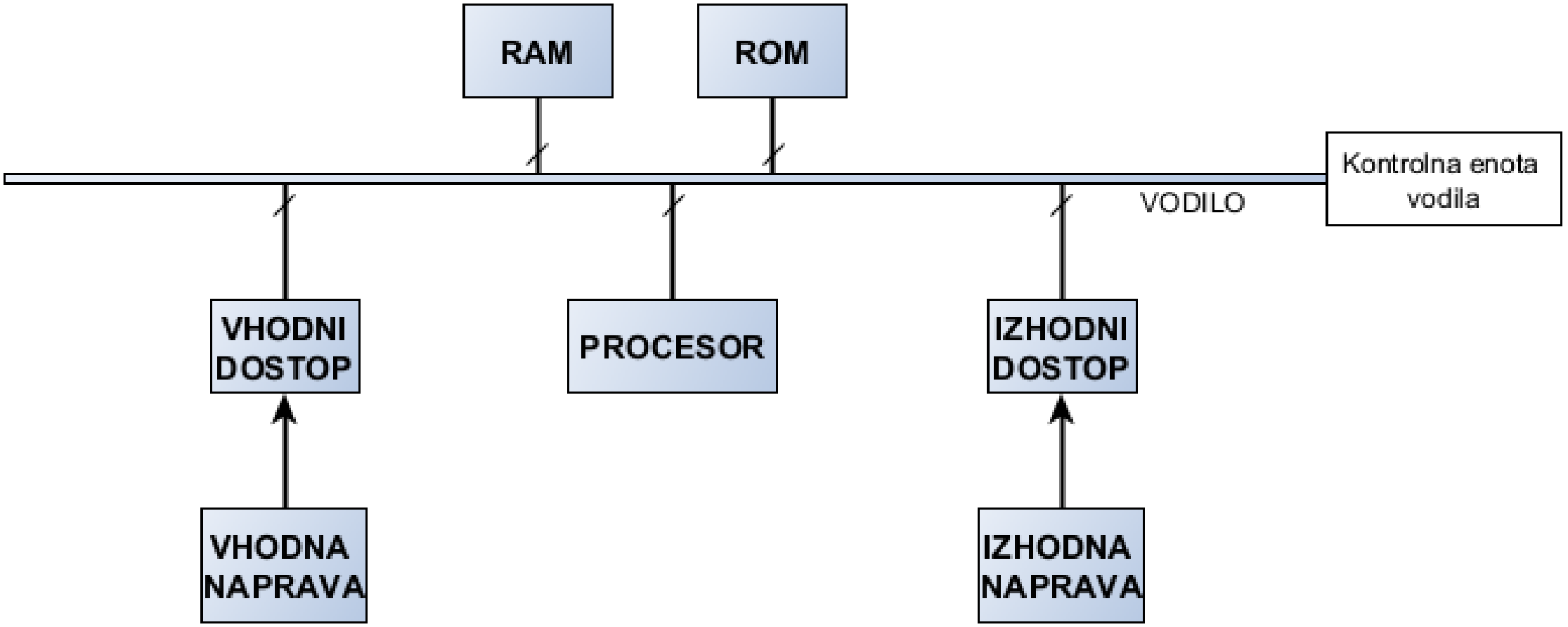
- **vhodni del**
  - preko vhodnega dela iz okolja vnašamo podatke ter ukaze iz programa
- **izhodni del**
  - preko izhodnega dela v okolje prenašamo rezultate izvajanja programa
- **delovni ali glavni pomnilnik**
  - uporabljamo ga za hrambo vseh podatkov ter ukazov iz programa, ki so vnešeni od zunaj, ter za hrambo rezultatov izvajanja programa
- **aritmetično-logična enota**
  - izvaja aritmetično-logične operacije, ki so določeni s strani ukazov
- **upravljalna enota**
  - prejema ukaze iz pomnilnika, izvaja dekodiranje ukazov in na osnovi tega upravlja z aritmetično-logično enoto ter z vhodnim in izhodnim delom
  - v sodobnih sistemih sta aritmetično-logična enota ter upravljalna enota združeni v procesorju, ki dodatno vsebuje še množico registrov

Vodilo (1)

- Iz funkcijske sheme von Neumannovega računalnika vidimo, da so posamezni deli med seboj povezani
  - vsaka povezava, ki predstavlja tok podatkov, ukazov ali upravljalnih signalov, je sestavljena iz večjega števila linij, ki tvorijo vodilo

- Pozamezne enote so povezane preko vodila (ang. bus)
  - prenos električnih signalov v obliki bitov
  - za prenos vsakega bita potrebujemo eno linijo vodila
  - vsaka povezava v funkcijskem modelu se nanaša na prenos določenega števila bitov
  - medsebojno direktno povezovanje vseh posameznih delov računalnika je nesmiselno,
  - zato je ustvarjeno vodilo, na katerega so priklopljeni vsi posamezni deli računalnika
  - z vodilom upravlja ločena upravljalska enota

Vodilo (2)

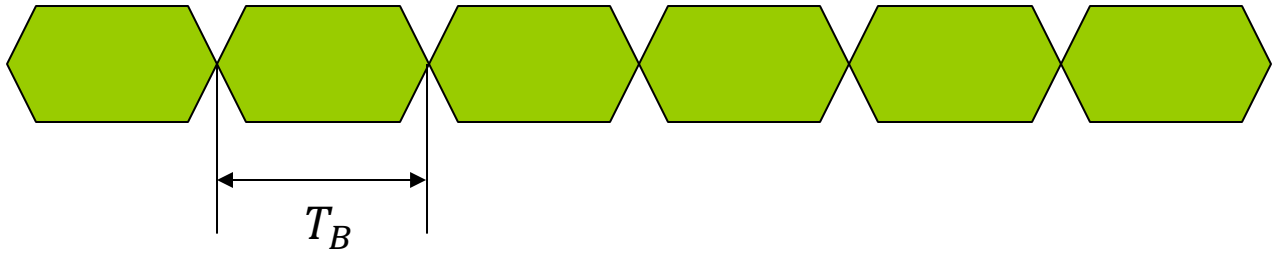

RAM
ROM
Kontrolna enota vodila
VODILO
VHODNI DOSTOP
PROCESOR
IZHODNI DOSTOP
VHODNA NAPRAVA
IZHODNA NAPRAVA

Vodilo (3)

- Vsa izmenjava podatkov, ukazov ter kontrolnih signalov se izvaja preko skupnega vodila

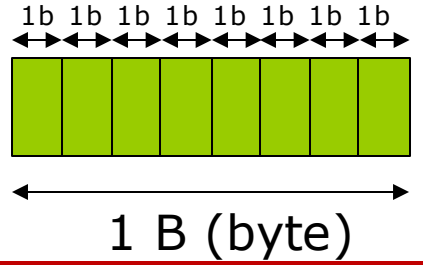
- Preko skupnega vodila se v enem trenutku izvaja samo en prenos
  - za dober izkoristek povezave se prenosi ustvarjajo v določenem časovnem intervalu
  - deljeni čas na vodilu (ang. time share)

- Časovna rezina na vodilu (ang. bus time slot)
  - v eni časovni rezini, ki traja T_B sekund, lahko ustvarimo samo en prenos
  - npr. vodilo, ki ima časovno rezino $T_B = 100$ ns, omogoča, da imamo v 1 sekundi 10 milijonov prenosov. Če lahko v eni rezini prenesemo 1 B, to pomeni, da se preko vodila lahko prenese 10 MB podatkov.



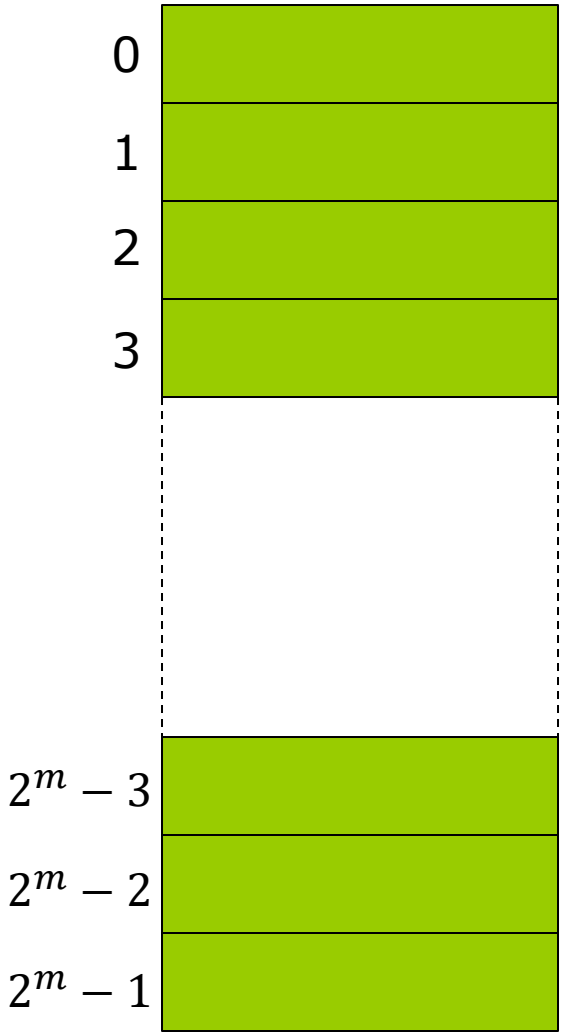
Hranilnik in delovni pomnilnik

- Centralni del vsakega računalniškega sistema predstavlja hranilnik
- Pri von Neumannovem modelu se centralni hranilnik imenuje delovni pomnilnik
- Pri izvajanju programov v delovnem pomnilniku morajo biti ukazi iz programa ter podatki, s katerimi ti ukazi operirajo oz. delujejo
- V sodobnih sistemih je najmanjša hranilna enota informacije 1 B (byte)
  - je skupina 8 bitov
  - do vsakega bajta informacije v hranilniku lahko dostopamo z direktnim dostopom



Naslovi in naslovni prostor

- Vsak bajt v hranilniku ima svojo zaporedno številko, ki se imenuje naslov
- S pomočjo naslova dostopamo do vsakega posameznega bajta
- Naslovi bajtov znotraj hranilnika se izražajo v številkah, zapisanih v dvojiškem sistemu
- Če za zapisovanje naslova uporabljamo m bitov, lahko zapišemo 2^m različnih naslovov
  - s številom m določamo velikost naslovnega prostora
  - naslovni prostor vključuje naslove od 0 do $2^m - 1$



Količina informacije in velikost naslovnega prostora

- Količino informacije merimo v enoti B (byte)
  - $1\ B = 2^3 = 8\ bit$
- Velikost pomnilnika, ki shranjuje določeno količino informacije izražamo v kB, MB, GB, TB
- kB (kilobyte)
  - $1\ kB = 2^{10}B = 1024\ B$
- MB (megabyte)
  - $1\ MB = 2^{20}B = 1024\ kB$
- GB (gigabyte)
  - $1\ GB = 2^{30}B = 1024\ MB$
- TB (terabyte)
  - $1\ TB = 2^{40}B = 1024\ GB$

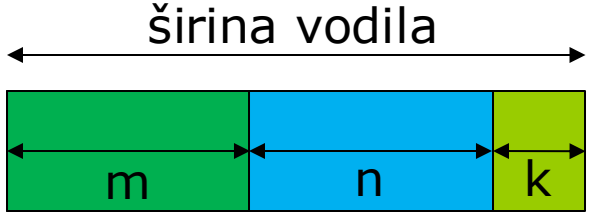
- Koliko velik naslovni prostor lahko naslovimo, če imamo na voljo m=8 bitov?
  - $2^8 = 512\ B$
- Koliko velik naslovni prostor lahko naslovimo, če imamo na voljo m=16 bitov?
  - $2^{16} = 2^6 * 2^{10} = 2^6 * 1\ kB = 64\ kB$
- Koliko velik naslovni prostor lahko naslovimo, če imamo na voljo m=32 bitov?
  - $2^{32} = 2^2 * 2^{30} = 2^2 * 1\ GB = 4\ GB$
- Koliko velik naslovni prostor lahko naslovimo, če imamo na voljo m=64 bitov?
  - $2^{64} = 2^4 * 2^{60} = 16\ EB\ (eksabyte)$

Model preprostega računalnika

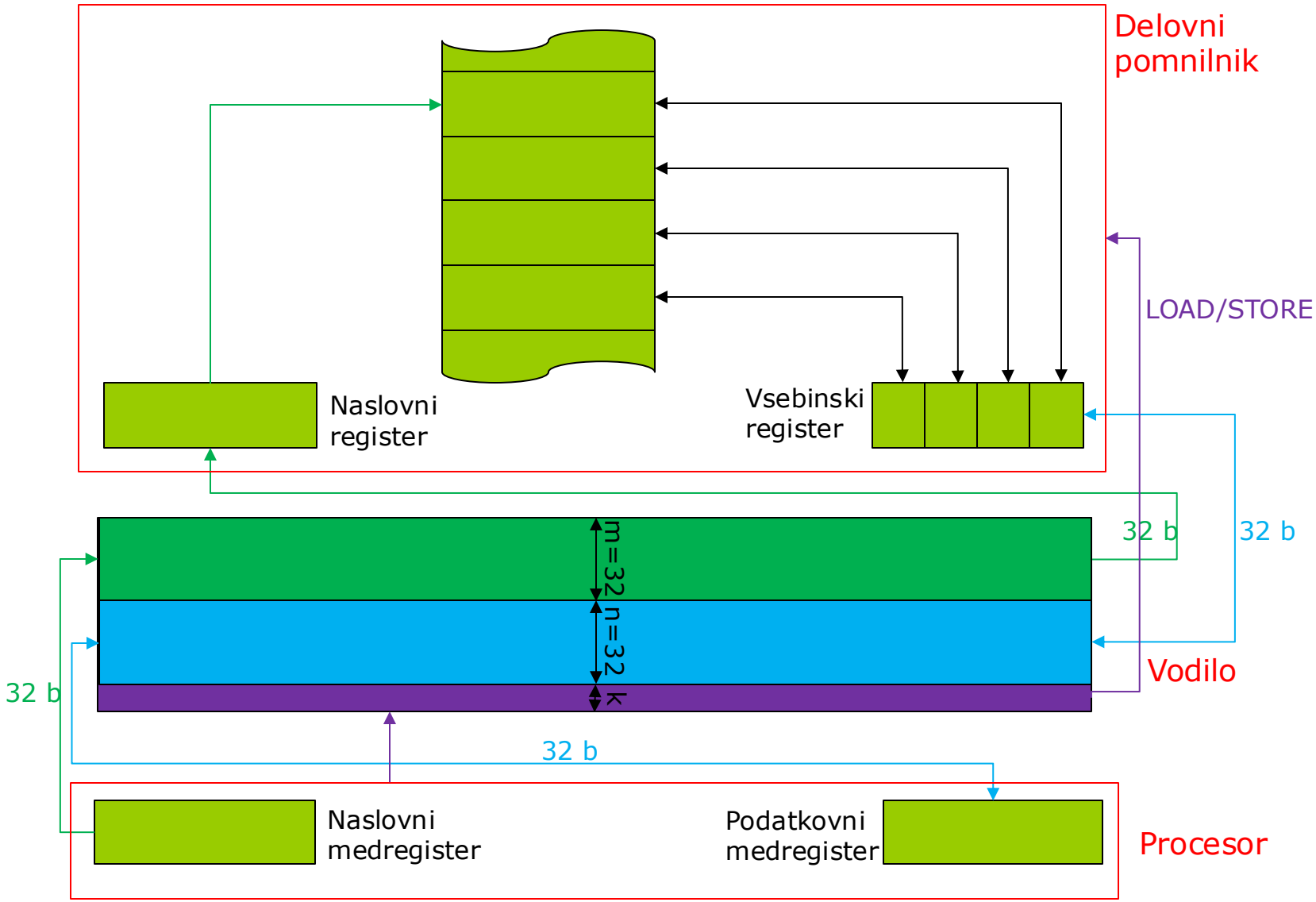
- Za opisovanje osnovnih lasnosti računalnika lahko opazujemo model preprostega računalnika, ki ga sestavljata samo procesor in delovni pomnilnik

- Pri tem imamo naslednje predpostavke:
  - v delovnem pomnilniku imamo naloženo strojno obliko programa v obliki niza ukazov, ki so shranjeni v zaporedne pomnilniške lokacije
  - v delovnem pomnilniku imamo vse vhodne podatke, ki jih program potrebuje pri izvajanju
  - program bo vse rezultate izvajanja shranjeval v delovni pomnilnik

- Delovni pomnilnik in procesor sta med seboj povezana preko vodila
  - vodilo je sestavljeno iz treh delov
  - naslovni del, ki vsebuje m linij
  - podatkovni del, ki vsebuje n linij
  - kontrolni del, ki vsebuje k linij



Povezava procesorja in pomnilnika preko vodila



Cikli na vodilu

- Procesor je povezan na vodilo preko svojih registrov
  - imenujemo jih medregistri, ker ustvarjajo povezavo z vodilom
  - iz naslovnega medregistra imamo povezavo z naslovnim delom vodila
  - podatkovni medregister ustvarja dvosmerno komunikacijo: podatki se lahko prenašajo iz procesorja v pomnilinik ali iz pomnilnika v processor

- V enem ciklu na vodilu (v eni časovni rezini) se lahko izvede eno shranjevanje (STORE) ali eno branje vsebine iz pomnilnika (LOAD)

- Pomnilniški cikel in cikel na vodilu imata končno trajanje T_B
  - ta čas je omejen s strani materialov, ki tvorijo vodilo in čipe
  - pri sistemih z enim vodilom vsi deli računalnika komunicirajo preko tega vodila
  - cikli na vodilu se morajo deliti med vse dele računalnika, ki hočejo prenašati podatke
  - vodilo lahko postane ozko grlo in omeji hitrost računalnika

STORE in LOAD pomnilniški cikel

- STORE cikel
  1. procesor postavi naslov iz naslovnega medregistra na naslovno vodilo
  2. procesor postavi vsebino, ki se bo shranila, na podatkovno vodilo
  3. procesor poda signal preko kontrolnega vodila, da gre za STORE ukaz
  4. pomnilnik prejme naslov v svoj naslovni register, aktivira naslovljeno pomnilniško lokacijo ter v njo shrani vsebino
- LOAD cikel
  1. procesor postavi naslov iz svojega naslovnega medregistra na naslovno vodilo
  2. procesor poda signal preko kontrolnega vodila, da gre za LOAD ukaz
  3. pomnilnik prejme naslov z naslovnega vodila ter prestavi naslovljeno vsebino v vsebinski register in od tam prenese vsebino na podatkovno vodilo
  4. procesor prejme vsebino v podatkovni medregister preko podatkovnega vodila

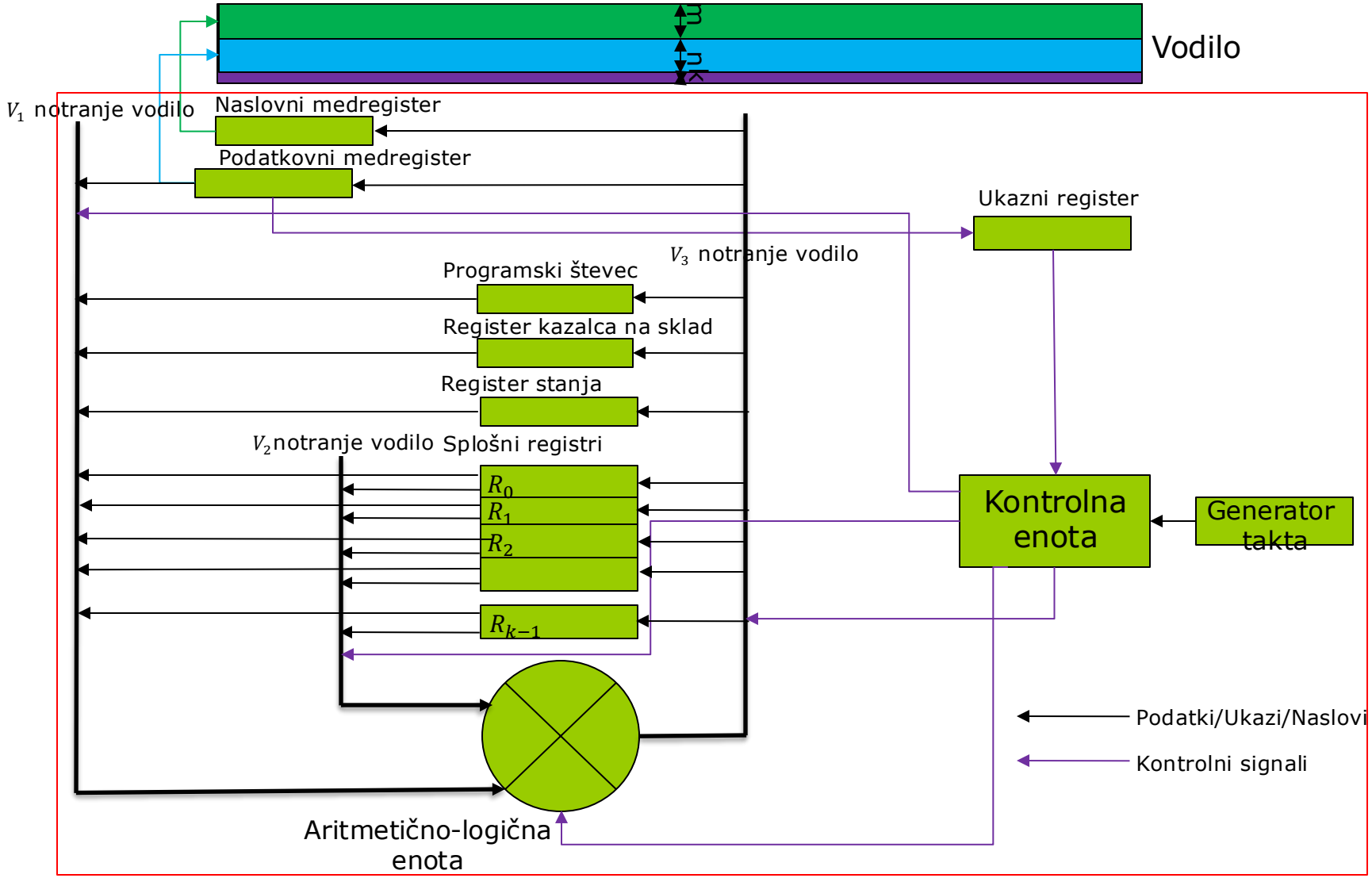
Procesor

- Delovni pomnilnik se uporablja samo za hranjenje nizov bitov
- Interpretacija pomena teh shranjenih bitov se dogaja zunaj pomnilnika
- V našem preprostem modelu računalnika se dogaja interpretacija znotraj procesorja
  - v sodobnih sistemih so to različne vhodno-izhodne naprave (npr. zaslon)
- Znotraj procesorja imamo dvojiško intepretacijo vsebine iz pomnilnika:
  - kot so denimo ukazi strojne kode nekega programa
  - kot osnovni podatkovni tipi
- Procesor naslavlja ukaze in podatke na različne načine
  - ukazi se pridobivajo z naslova, na katerega kaže ločeni register, ki ga imenujemo programski števec
  - podatki se naslavljajo v teku izvajanja ukazov; lokacijo podatkov pridobi iz samih ukazov

Struktura procesorja

- Procesor v Von Neumannovem modelu računalnika sestavljajo:
  - aritmetično-logične enota
  - kontrolne enota
  - množice registrov
- Tukaj si bomo ogledali zelo poenostavljen model procesorja
  - razumevanje preprostega modela procesorja nam bo pomagala pri razumevanju delovanja sodobnih procesorjev, ki so veliko bolj kompleksni
  - Predpostavimo, da je velikost naslovnega in podatkovnega vodila enaka
  - vsi registri so enako veliki
- Osnovne lastnosti in obnašanje procesorja so določeni z množico registrov ter množico ukazov, ki jih procesor lahko izvede
  - Registri se uporabljajo za shranjevanje vseh informacij, ki pridejo v procesor, pa tudi za tiste, ki izhajajo iz procesorja
  - Množica ukazov je določena s fizično izvedbo aritmetično-logične enote ter kontrolne enote

Preprosti model procesorja



Registri procesorja (1)

- Naslovni medregister
  - se uporablja za naslavljanje pomnilnika ali ostalih delov računalnika
- Podatkovni medregister
  - posreduje pri izmenjavi podatkov med procesorjem in ostalih delov računalnika
- Ukazni register (ang. Instruction register)
  - v ukazni register se shranjuje ukaz, ki se pridobi iz pomnilnika. Dekodiranje ukaza izvaja kontrolna enota.
- Programski števec (ang. Program counter)
  - vsebuje naslov naslednjega ukaza, ki se bo izvedel
  - kontrolna enota avtomatično posodablja programski števec, tako da se ukazi izvajajo zaporedno.

Registri procesorja (2)

- Register kazalca sklada (ang. Stack pointer)
  - omogoča shranjevanje podatkov ali ukazov v sklad
- Register stanja (ang. Status register, Flag register, Condition Code Register)
  - se uporablja za zapisovanje različnih zastavic (ang. Flag), ki označujejo pravilnost ali nepravilnost pri izvajanju ukazov ali pri delu procesorja
  - na osnovi zastavic se kontrolna enota odloča, kako izvajatii programme
- Množica splošnih registrov
  - uporabljajo se za shranjevanje operandov in rezultatov
  - uporabljajo se za pripravo naslova za naslednje dostopanje do pomnilnika
- Registri procesorja so med seboj povezani preko notranjih vodil
  - povezava je tudi z aritmetično-logično enoto

Kontrolna enota procesorja

- Kontrolna enota upravlja s celotnim delovanjem procesorja
- Izvaja dekodiranje ukaza, ki se nahaja v ukaznem registru
  - Na osnovi dekodiranega ukaza izvaja aktivacijo notranjih vodil
  - Sporoča aritmetično–logični enoti, kakšno operacijo naj izvede
- Avtomatično prehaja iz izvajanja enega ukaza na izvajanje naslednjega ukaza
- Kontrolna enota kakor tudi ostale enote računalnika delajo v ritmu, ki ga narekuje generator takta
  - generira elektronske impulze
  - vse operacije so na ta način sinhronizirane
  - v sodobnih procesorjih je takt v GHz (gigahertz)
  - takt določa hitrost samega procesorja

Aritmetično-logična enota

- Aritmetično-logična enota izvaja aritmetične, logične in shift operacoke z operandi, ki jih napeljejo notranja vodila (v našem modelu S_1 in S_2), ter rezultat poda na notranje vodilo S_3
  - Aritmetične operacije: seštevanje, odštevanje, inkrement, dekrement
  - Logične operacije: AND, OR, XOR, negacija
  - Shift operacije: aritmetični, logični, rotacija

- Operacija, ki jo izvaja aritmetično-logična enota, je določena s strani kontrolne enote, ki dekodira ukase

- Kontrolna enota prav tako določa, od kod vzeti vsebino za operande (iz katerega registra) in kam poslati rezultat izvedbe operacije

- V skladu s tem aktivira samo določene povezave na notranjih vodilih

Kako dela procesor? (1)

- Predpostavimo, da imamo v pomnilniku programsko kodo v obliki niza ukazov v pravilnem vrstnem redu za izvajanje
- Ta niz ukazov, ki jih lahko izvede procesor, se imenuje **strojna programska koda**
- V programski števec postavimo naslov prvega ukaza strojne kode
- Procesor se obnaša kot avtomat, ki (ko je enkrat vklopljen) zaporedno izvaja ukaze iz strojnega programa

Kako dela procesor? (2)

- **Ponavljaj {**
  - prinesi iz pomnilnika ukaz, na katerega kaže programski števec;
  - dekodiraj pridobljeni ukaz, določi, katero operacijo izvede ALU;
  - povečaj vrednost programskega števca tako, da kaže na naslednji ukaz, ki bo izveden;
  - določi lokacijo operandov ter lokacijo rezultatov operacije;
  - operande napelji na ALU ter izvedi operacijo;
  - rezultat operacije shrani na določeno lokacijo
- **} vse dokler je procesor vklopljen;**

Faze delovanja procesorja: prevzem ukaza

- V prvi fazi se izvaja prevzem ukaza (ang. fetch)
- Vsebina programskega števca se prenaša v naslovni medregister
- Kontrolna enota iniciira aktivnost prenašanja ukaza iz pomnilnika
- Ukaz pride v podatkovni medregister in se od tam prenese v ukazni register

Faze delovanja procesorja: dekodiranje ukaza

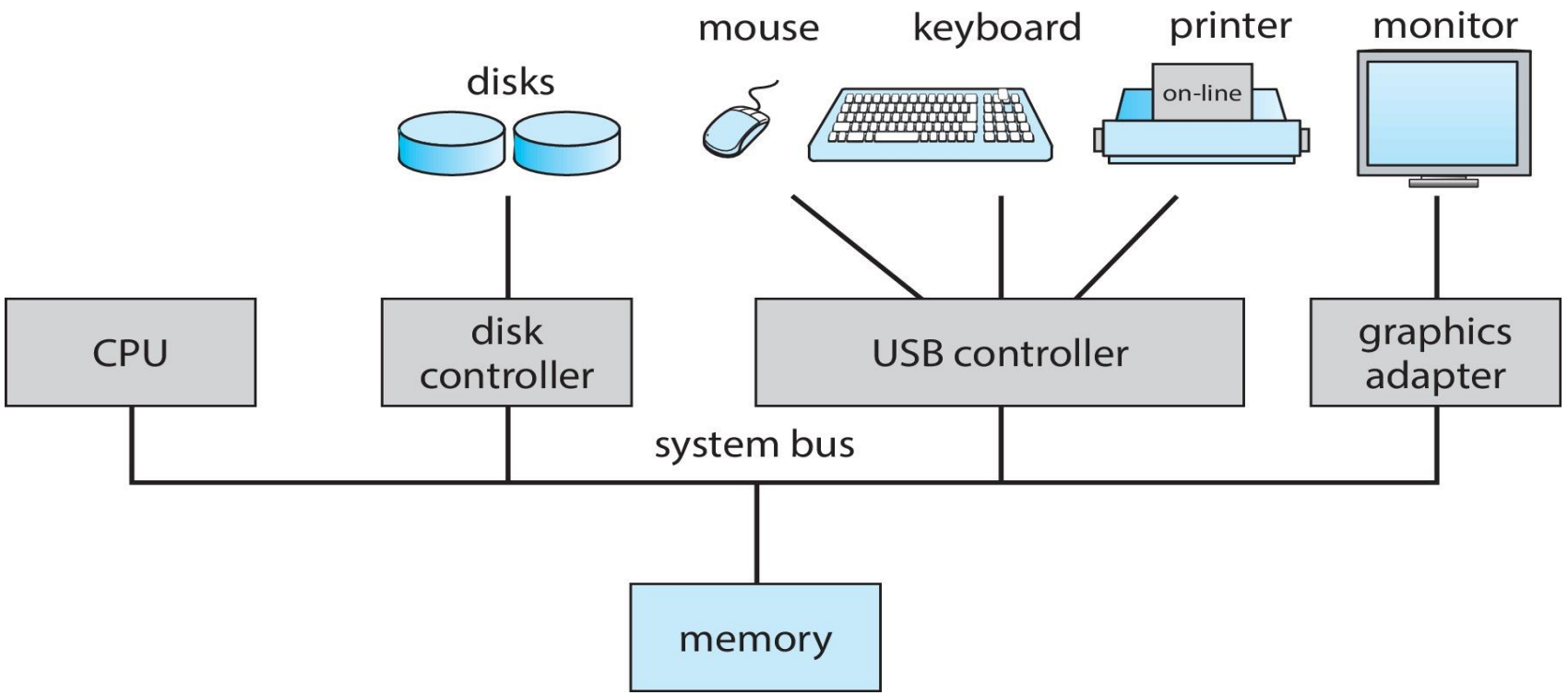
- V drugi fazi kontrolna enota izvaja dekodiranje ukaza (ang. decoding)
- Na osnovi množice bitov ukaza (operacijska koda) kontrolna enota ugotovi, katero operacijo je treba izvesti
- Poveča vrednost programskega števca tako, da kaže na naslednji ukaz
- Pošilja signale ALU in jo obvesti, katero operacijo je treba izvesti
- Na osnovi druge množice bitov (naslovni del ukaza) ugotovi, iz katerih registrov pridejo operandi, kam je treba shraniti rezultat in katera interna vodila aktivirati
- Če naslovni del ukaza določi, da prihaja operand iz pomnilnika, je treba prenesti naslov operanda v naslovni medregister in začeti prenašati operanda iz pomnilnika v podatkovni medregister, od tam pa v ALU

Faze delovanja procesorja: izvajanje operacij

- V tretji fazi se izvajajo operacije v ALU (ang. execution)
- ALU izvaja operacijo in shranjuje rezultat preko vodila S_3 v odredišče
- Vrednosti določenih zastavic, ki so odvisne od pridobljenega rezultata, se shranjujejo v register stanja
  - s tem označujemo uspešnost/neuspešnost izvedenega ukaza
  - to informacijo potem izkoristi operacijski sistem za svoje delovanje

ORGANIZACIJA IN DELOVANJE SODOBNIH RAČUNALNIŠKIH SISTEMOV

Organizacija sodobnega računalniškega sistema (1)



Vir: Silberschatz et al. Operating system concepts (Global edition), str. 7

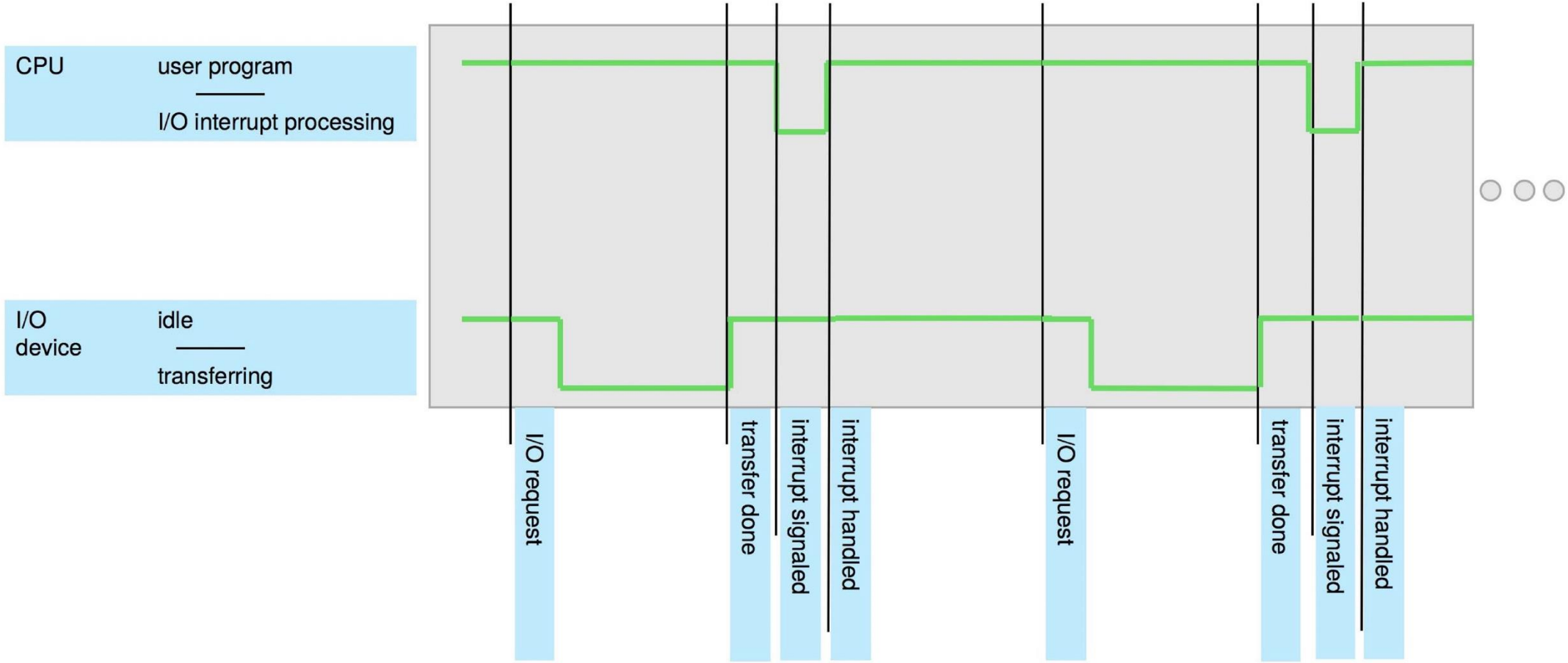
Organizacija sodobnega računalniškega sistema (2)

- Sodobni računalniški sistemi vključujejo tudi številčne vhodno/izhodne (V/I) naprave zraven dosedanjih komponent iz Von Neummanovega modela: pomnilnik, procesor in vodilo
- Ena ali več centralnih procesorkih enot (CPE) ter krmilniki naprav (ang. device controllers) se povezujejo preko skupnega vodila (ang. bus) in imajo s tem dostop do deljenega pomnilnika (ang. shared memory)
  - CPE premika podatke iz glavnega pomnilnika (ali v glavni pomnilnik) v lokalne medpomnilnike (ali iz lokalnih medpomnilnikih)
- Vsak krmilnik naprave je odgovoren zlasti za eno vrsto naprav
  - ima svoj lokalni medpomnilnik (ang. local buffer)ter nekaj registrov z točno določeno funkcijo
  - odgovoren je za prenos podatkov iz V/I naprav do medpomnilnika
- Operacijski sistemi imajo gonilnike (ang. driver) za vsak krmnilnik
  - zagotavlja preostalemu operacijskemu sistemu enoten vmesnik za vsako napravo
- Vhodno-izhodne naprave in CPE lahko izvajajo naloge sočasno
  - Sočasno izvajanje (ang. concurrent execution) nalog CPE in naprav, ki tekmujejo za pomnilniški cikel (ang. memory cycle)

Izvajanje V/I operacij in prekinitve

- Za začetek V/I operacij, gonilnik naprave prvo naloži vsebino v registre krmnilnika naprave
- Sočasno krmnilnik preveri vsebino registrov ter opredeli kakšno nalogo mora izvesti (npr. prebrati en znak iz tipkovnice)
- Vhod/izhod podatkov se ustvarja iz same naprave v lokalni medpomnilnik krmnilnika naprave
- Krmnilnik naprave obvesti gonilnika naprave ko zaključi s prenosom podatkov
- Obveščanje poteka s pomočjo prekinitve (ang. interrupt)
- Prekinitve so ena od najbolj pomembnih mehanizmov za delovanje računalniških sistemov

Časovnica poteka prekinitve



Vir: Silberschatz et al. Operating system concepts (Global edition), str. 8

Prekinitev v računalniškem sistemu

- Strojna oprema lahko pošlje prekinitev kadarkoli
  - To naredi s pomočjo električnega signala preko sistemskega vodila
  - S tem obvesti CPE ali kakšno drugo napravo

- **Izjema (ang. trap ali exception)** je prekinitev, ki jo ustvari programska oprema (ang. software-generated interrupt) in je povzročena s strani napake ali zahteve uporabnika

- Operacijski sistem **temelji na principu prekinitve (ang. interrupt driven)**

Prekinitveni vektor

- Ko je delovanje CPE prekinjeno se kontrola prenese k modulu za prekinitev (ang. interrupt service routine)

- To se naredi s pomočjo **prekinitvenega vektorja (ang. interrupt vector)**, ki vsebuje začetne naslove (v pomnilnku) vseh servisnih rutin
  - Servisne rutine so programi, ki se izvedejo ko nastane prekinitev

- Po koncu izvajanja se vrne kontrola CPE ter ta nadaljuje z izvajanjem prekinjenega programa
  - Za ta namen prekinitveni sistem mora shraniti naslov prekinjenega ukaza (ang. interrupted instruction)

Prekinitveni vektor pri Intel procesorjih

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Vir: Silberschatz et al. Operating system concepts (Global edition), str. 11

Implementacija prekinitev znotraj računalniškega sistema (1)

- Strojna oprema CPE ima linijo za oddajanje zahtevkov za prekinitev (ang. interrupt – request line)

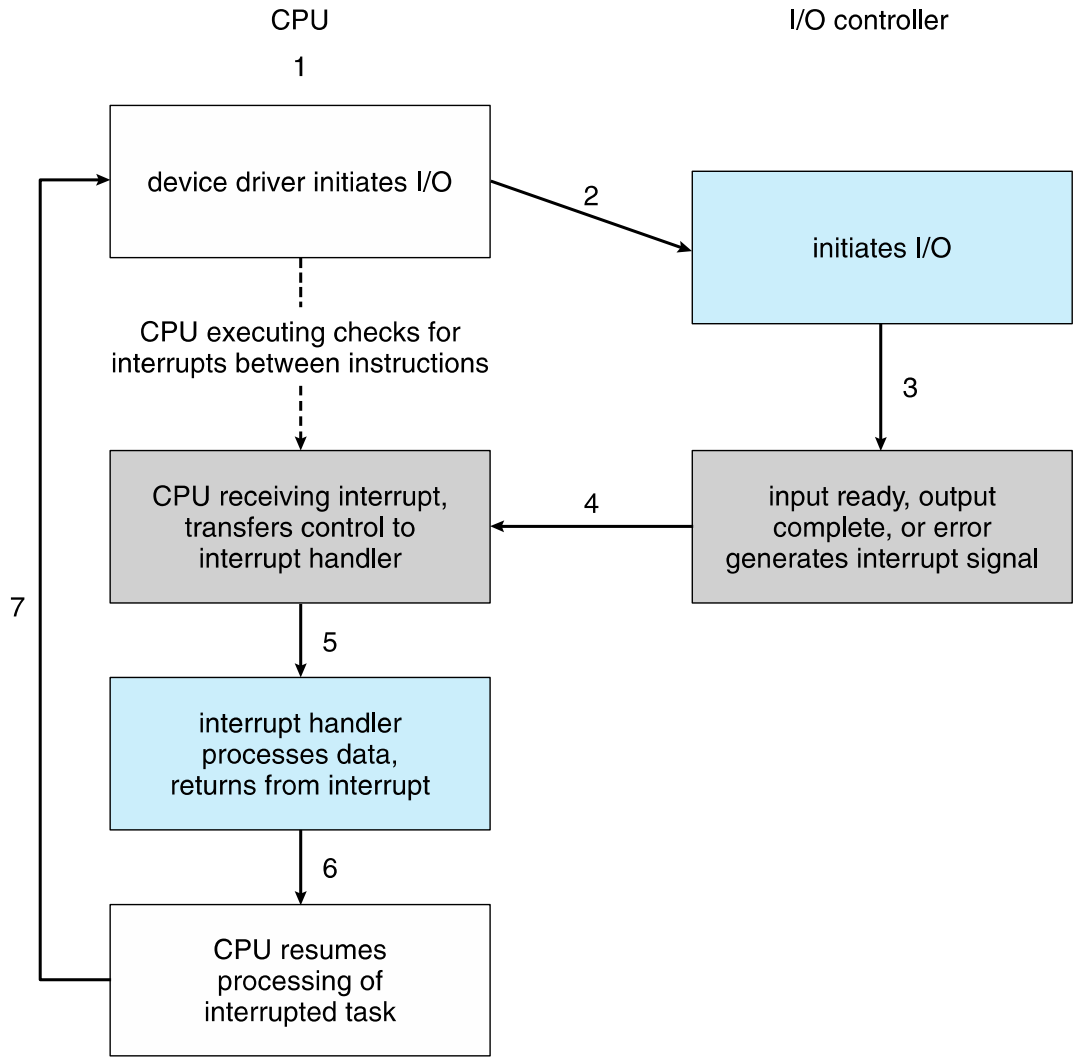
- CPE preverja to linijo po vsakem izvedenem ukazu

- Ko CPE zazna da je krmnilnik oddal signal z uporabo prekinitvene linije:
  - prebere številko prekinitve
  - skoči na servisni prekinitveni program (ang. interrupt – handling routine), ki se nahahaja v pomnilniku
  - ločeni segmenti programske kode določajo, katera dejanja je treba sprejeti za vsako vrsto prekinitve
  - potem začne izvajati prekinitveni program

Implementacija prekinitev znotraj računalniškega sistema (2)

- Nadzornik prekinitve (ang. interrupt handler)
- Shrani stanje CPE: operacijski sistem ohranja stanje CPE s shranjevanjem stanja v registrih in stanja programskega števca (ang. program counter)
- Ugotovi razlog za prekinitev
  - pripravljenost vhodno/izhodne naprave (ang. polling interrupt)
  - vektorske prekinitve (ang. vectored interrupt)
- Izvede prekinitev
- Poskrbi za vrnitev v stanje pred prekinitvijo

Cikel V/I prekinitve



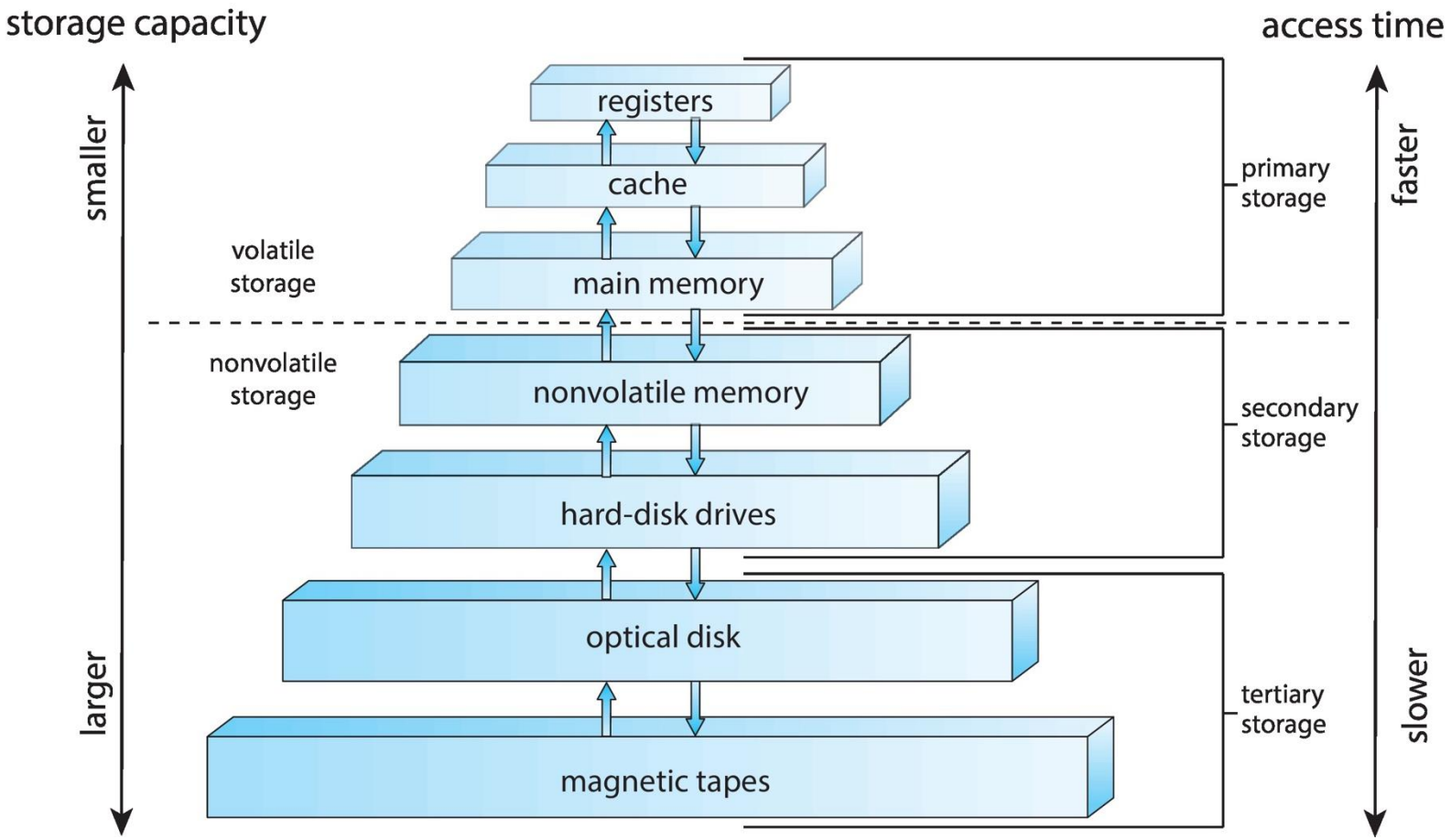
Vir: Silberschatz et al. Operating system concepts (Global edition), str. 10

Struktura pomnilnika (ang. Storage Structure)

- Glavni pomnilnik (ang. main memory) – edini večji hranilni medij, do katerega lahko CPE neposredno dostopa
  - **Naključni dostop (ang. random access)**
  - Pogosto je neobstojen **(ang. volatile)**
- Dodatni hranilnik (ang. secondary storage) – razširitev glavnega pomnilnika, ki omogoča **obstojno (ang. nonvolatile)** shranjevanje velike količine podatkov
- Trdi diski (ang. hard disks) – toge kovinske ali steklene plošče, ki so pokrite z magnetnim snemalnim materialom
  - Površina diska je logično razdeljena v **steze (ang. tracks)**, ki so potem razdeljene v **odseke (ang. sectors)**
  - **Krmilnik diska (ang. disk controller)** določa logično sodelovanje med napravo in računalnikom
- **Diski s bliskovnim pomnilnikom (ang. solid-state disks)** – hitrejši kot trdi diski ter obstojni
  - Različne tehnologije izdelave
  - Vedno bolj popularni

Pomnilniška hierarhija (ang. Storage Hierarchy)

- Hranilne sisteme lahko organiziramo v obliko hierarhije glede na različne kriterije:
  - hitrost,
  - cena,
  - obstojnost.
- **Predpomnenje (ang. caching)** – kopiranje informacije v hitrejši hranilnik; na glavni pomnilnik lahko gledamo kot na predpomnilnik dodatnega hranilnika



Vir: Silberschatz et al. Operating system concepts (Global edition), str. 13

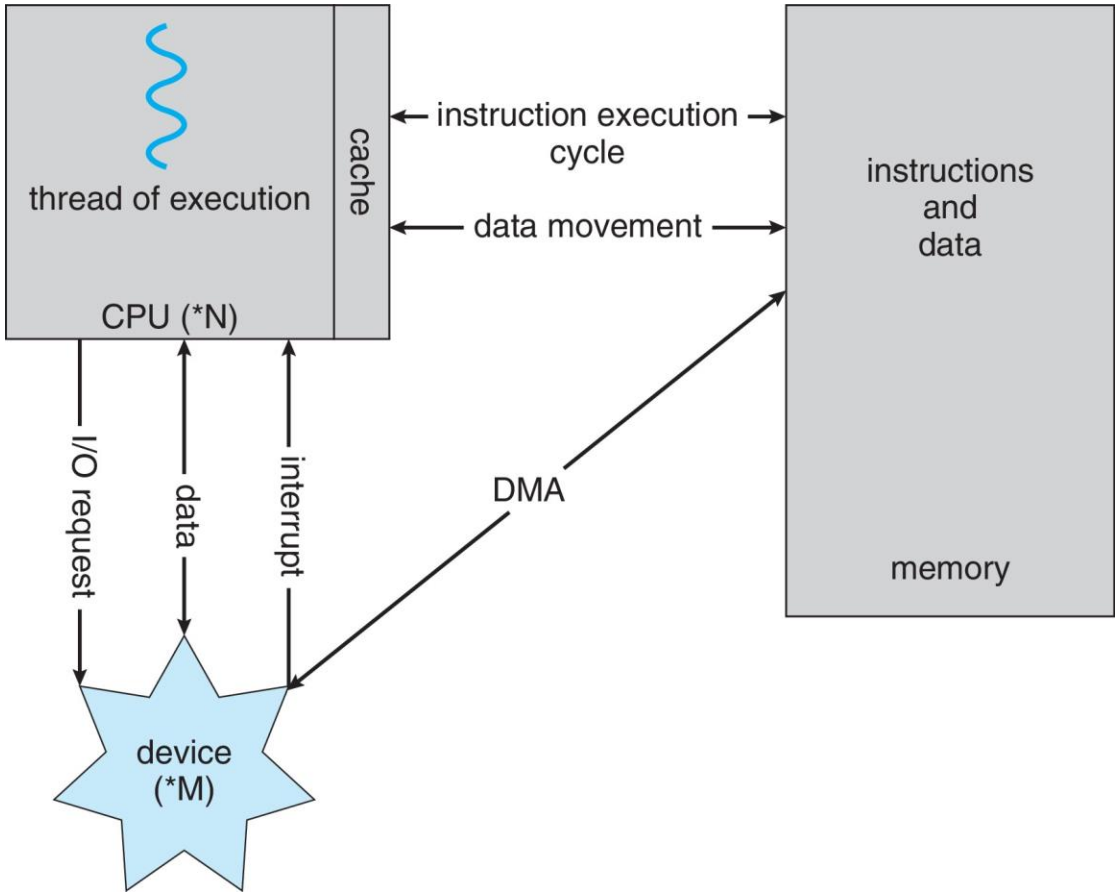
Struktura vhodno/izhodnih operacij (ang. I/O Structure)

- Po začetku vhodno/izhodnih (V/I) operacij se kontrola vrne uporabniškemu programu samo po koncu V/I operacije
  - Ukaz „wait“ omogoča, da je CPE nedejavna do naslednje prekinitve
  - Zanka čakanja (konflikt za dostop do pomnilnika)
  - Največ en V/I zahtevek se izvaja naenkrat, sočasne V/I operacije niso dovoljene

- Po začetku V/I operacij se kontrola vrne uporabniškemu programu brez čakanja na konec V/I operacije
  - **Sistemski klic (ang. system call)** – zahtevek, ki se pošlje operacijskemu sistemu, da omogoči uporabniku čakanje za konec V/I operacije
  - **Tabela s statusi naprav (ang. device-status table)** vsebuje zapis o tipu, naslovu in stanju vsake V/I naprave
  - Operacijski sistem indeksira to tabelo ter določa stanje V/I naprave ter spreminja zapise pri pojavljanju prekinitve

Struktura neposrednega dostopa do pomnilnika (ang. Direct Memory Access)

- Uporablja se za V/I operacije z visoko hitrostjo, ko so naprave zmožne hitrosti prenosa, primerljive s hitrostjo pomnilnika
- Upravljalnik naprave prenese blok podatkov iz medpomnilnika direktno v glavni pomnilnik brez vpliva CPE
- Za vsak blok podatkov se ustvari samo ena prekinitev, raje kot ena prekinitev za vsak bajt podatkov



Vir: Silberschatz et al. Operating system concepts (Global edition), str. 15

Predavanja 2 – Koncepti in terminologija

- von Neumanov model računalnika
  - vodilo, časovna rezina na vodilu
  - hranilnik in delovni pomnilnik
  - naslovi in naslovni prostor
  - količina informacije in velikost naslovnega prostora
  - Cikli na vodilu: STORE, LOAD
- Procesor
  - struktura: registri, kontrolna enota, aritmetično – logična enota, notranja vodila, generator takta
  - Registri: naslovni medregister, podatkovni medregister, ukazni register, programski števec, register kazalca sklada, register stanja, splošni registry
  - kontrolna enota: dekodira ukaze, upravlja z notranjimi vodili
  - ALU: izvaja operacije
  - Faze delovanja procesorja: prevzem ukaza, dekodiranje, izvajanje
- Organizacija sodobnih računalniških sistemov
  - Prekinitve: prekinitveni vektor, nadzornik prekinitve
  - Struktura pomnilnika in pomnilniška hirearhija
  - V/I operacije in neposredni dostop do pomnilnika (DMA)