Uvod v računalništvo (UvR)

Organizacija računalniških sistemov

Danijel Skočaj
Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Literatura: Invitation to Computer Science, poglavje 5

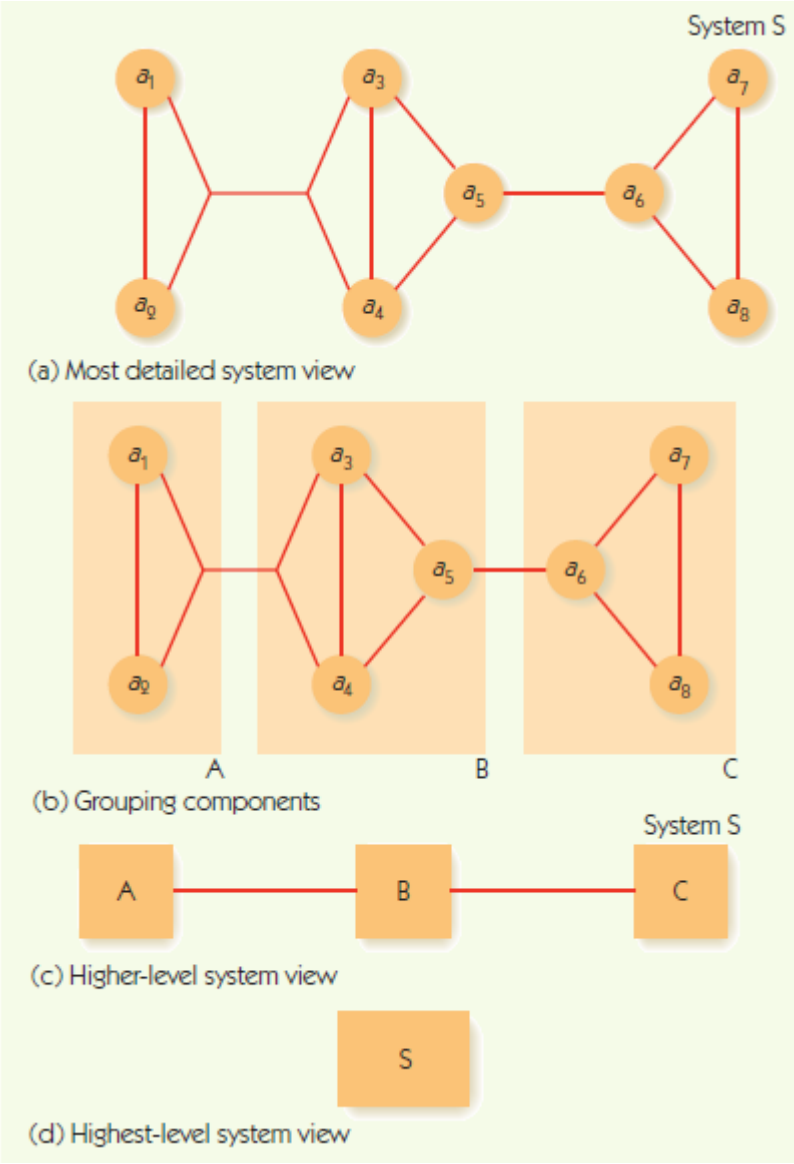
v1.0
Št. leto 2013/14

Cilji predavanja

- Znati našteti značilnosti Von Neumannove arhitekture
- Opisati komponente pomnilnika RAM, predpomnilnika
- Opisati naprave za shranjevanje velikih količin podatkov ter razložiti njihovo delovanje
- Narisati komponente tipične ALE in ilustrirati njeno delovanje
- Opisati krmilno enoto in razložiti kako implementira shranjen program
- Orisati komponente tipičnega Von Neumannovega stroja
- Pokazati zaporedje korakov med izvajanjem tipičnega ukaza
- Opisati Von Neumannove sisteme za vzporedne procesiranje

Uvod

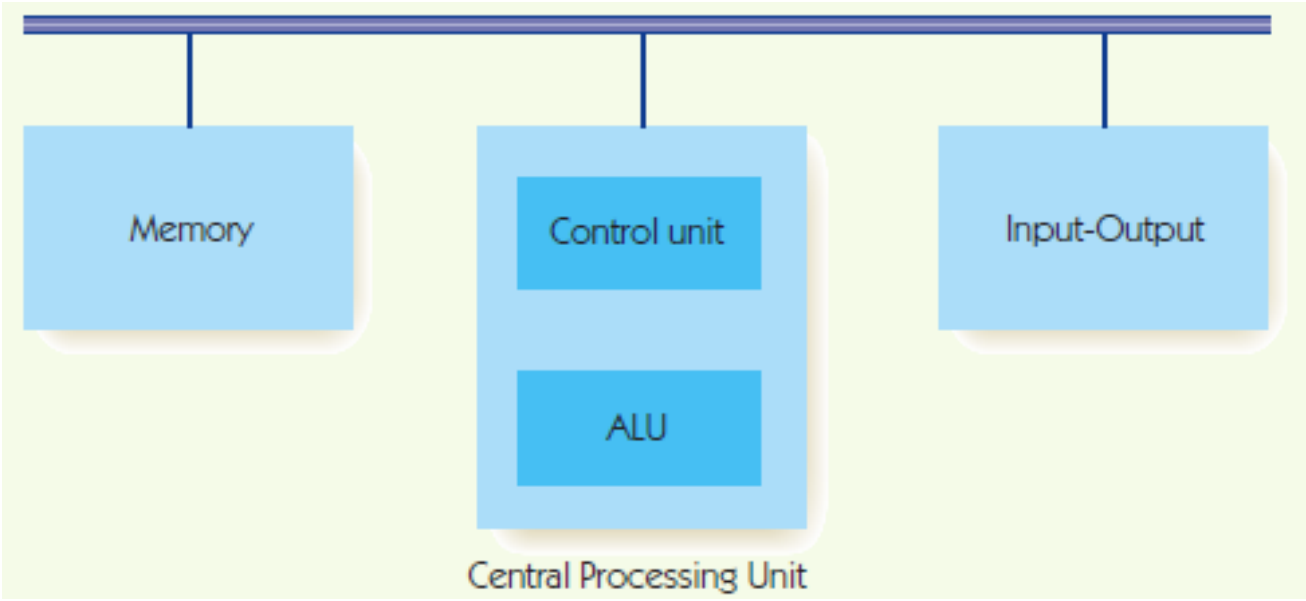
- Višji novo abstrakcije
- Funkcionalne enote organizacije računalniškega sistema
- Hierarhija abstrakcij
  - tranzistorji
  - vrata
  - vezja
- Različen pogled na osnovne enote



Komponente računalniškega sistema

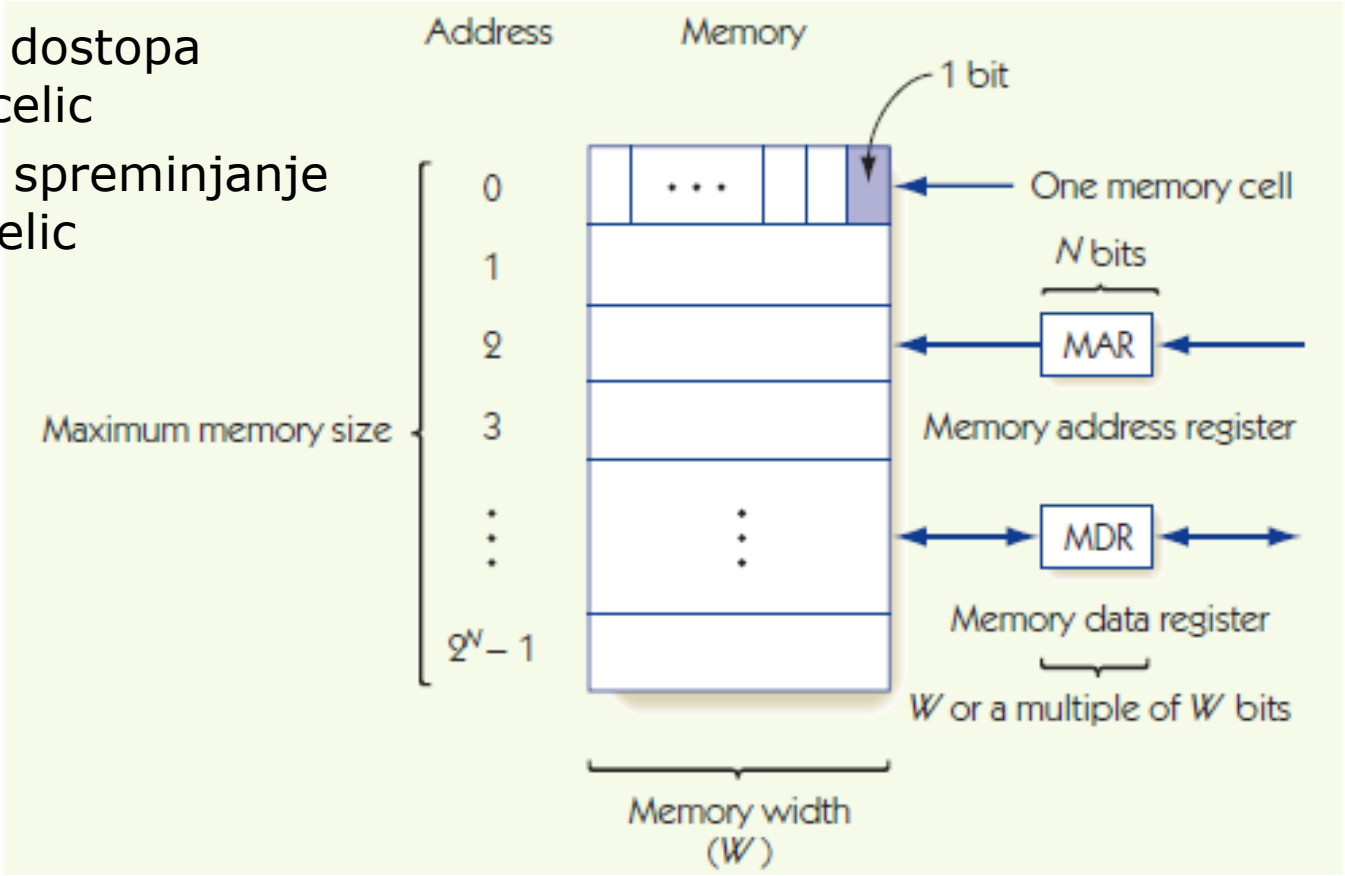
- Von Neumannova arhitektura
  - osnova za skoraj vse moderne računalnike
- Značilnosti:
  - štirje glavni podsistemi:
    - pomnilnik
    - vhod/izhod
    - aritmetično-logična enota
    - krmilna enota

    } centralna procesna enota
  - koncept shranjenega programa (v pomnilniku)
  - zaporedno izvajanje ukazov



Pomnilnik

- Funkcijska enota za shranjevanje in branje podatkov
- Pomnilnik z naključnim dostopom (Random Access Memory – RAM)
  - celice z naslovi
  - enak čas dostopa do vseh celic
  - branje in spreminjanje vsebin celic



Pomnilnik

- Maksimalna velikost pomnilnika (naslovni prostor) je odvisna od dolžine naslova
  - 16-bit: 2^{16}=65.536=64*2^{10}=64KB
  - 32-bit: 2^{32}= 4.294.967.296 =4*2^{30}=4GB
  - 64-bit: 2^{64}= 18.446.744.073.709.551.616=17mlrdGB

- Enote:
  - 2^{10}=1K - kilo
  - 2^{20}=1M - mega
  - 2^{30}=1G - giga
  - 2^{40}=1T - tera
  - 2^{50}=1P - peta

N	Maximum Memory Size (2^N)
16	65,536
20	1,048,576
22	4,194,304
24	16,777,216
32	4,294,967,296
40	1,099,511,627,776
50	1,125,899,906,842,624

Pomnilnik

- Naslov in vsebina pomnilniške lokacije
- Dve osnovni pomnilniški operaciji:
  - branje oz. nalaganje: `value = Fetch(address)`
    - nedestruktivno branje
  - pisanje oz. shranjevanje: `Store(address,value)`
    - destruktivno pisanje
- Čas dostopa do pomnilnika
  - čas za izvedbo ene operacije branja oz. pisanja
  - enak za vseh 2^N naslovov
  - 5 do 10 nsec = 5 do 10 * 10^{-9} sec

- MAR: Memory Address Register – naslovni register
  - vsebuje naslov pomnilniške lokacije
- MDR: Memory Data Register – podatkovni register
  - prejme podatke, oz. vsebuje podatke za pisanje

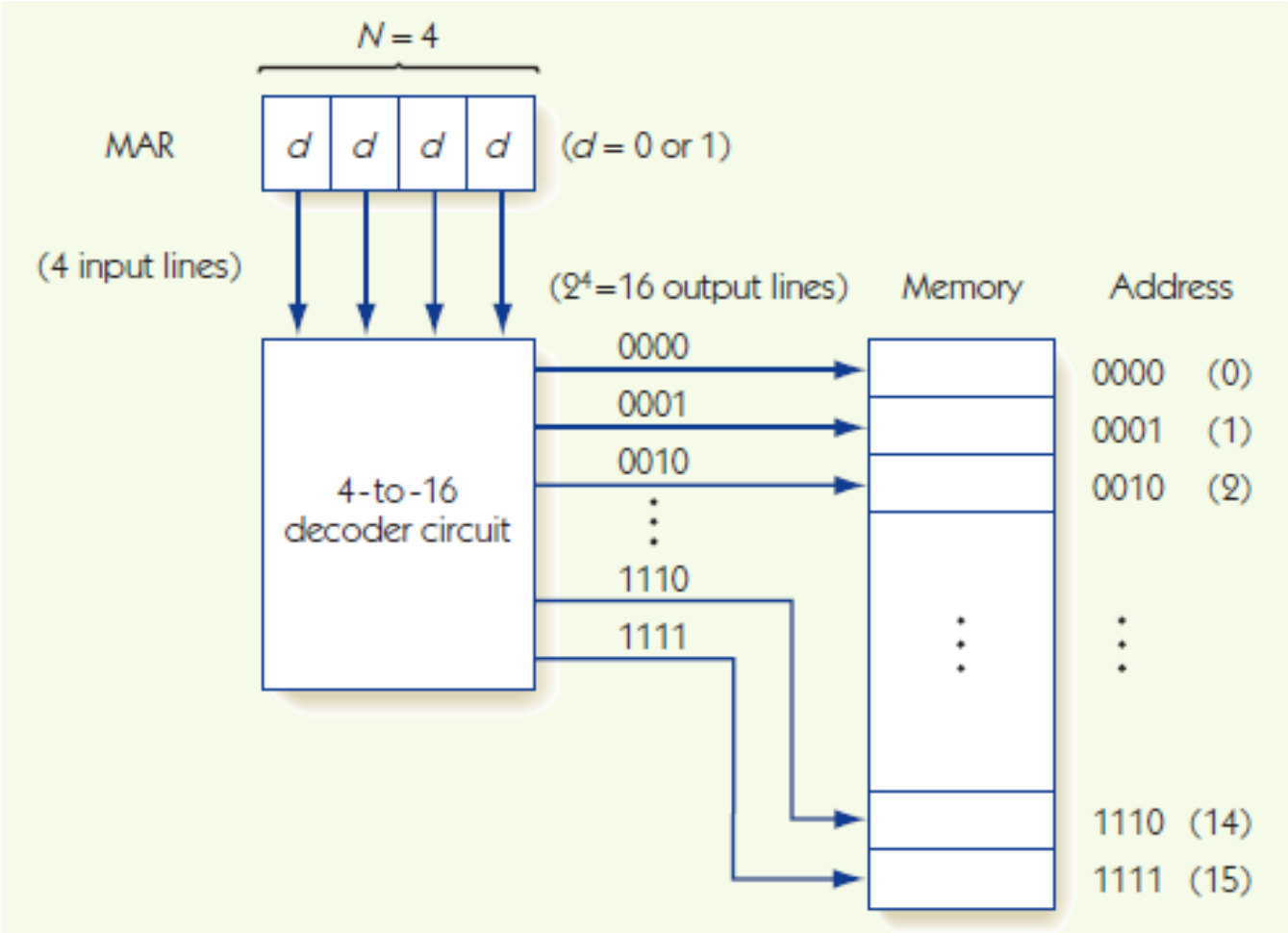
Branje in pisanje

- Branje s pomnilniške lokacije:
  1. Naloži naslov v MAR
  2. Dekodiraj naslov v MAR
  3. Kopiraj vsebino te pomnilniške lokacije v MDR

- Pisanje na pomnilniško lokacijo:
  1. Naloži naslov v MAR
  2. Naloži vrednost v MDR
  3. Dekodiraj naslov v MAR
  4. Shrani vsebino MDR na to pomnilniško lokacijo

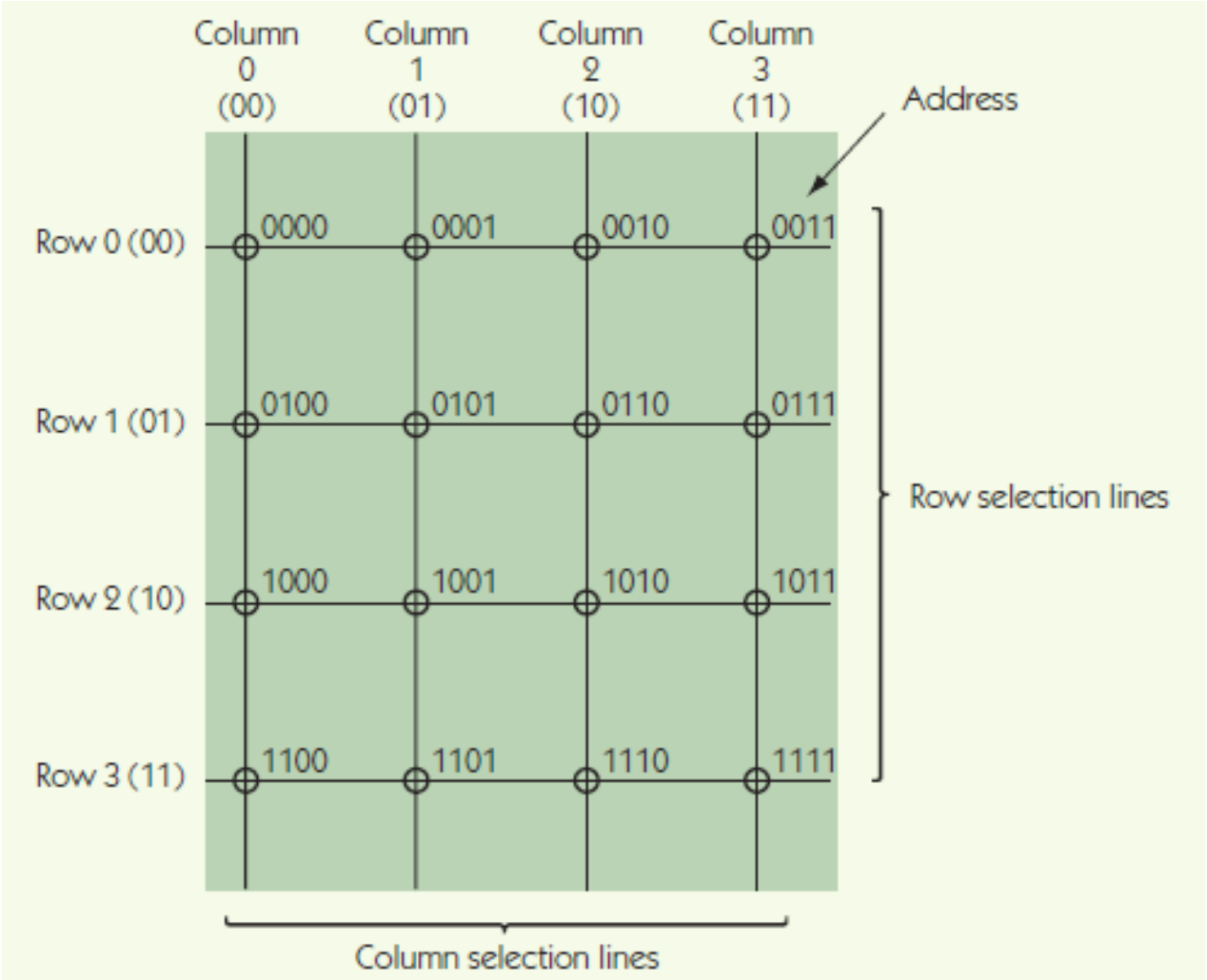
Naslavljanje

- Dekodirnik pretvori MAR v signal za specifično pomnilniško lokacijo



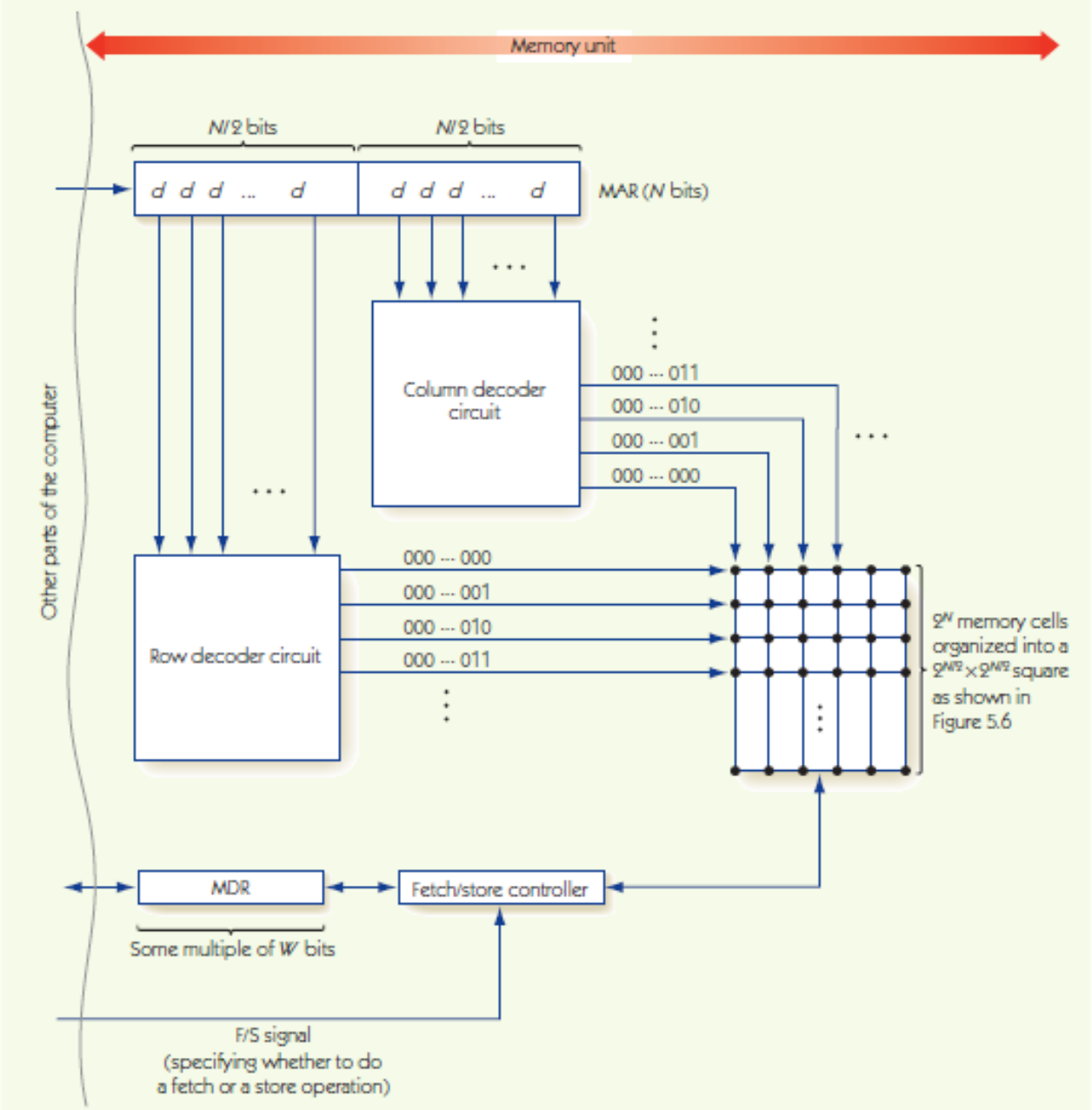
Naslavljanje

- Dvodimenzionalna pomnilniška organizacija



Pomnilnik

- Krmilnik za branje/pisanje
- Celotno vezje pomnilnika

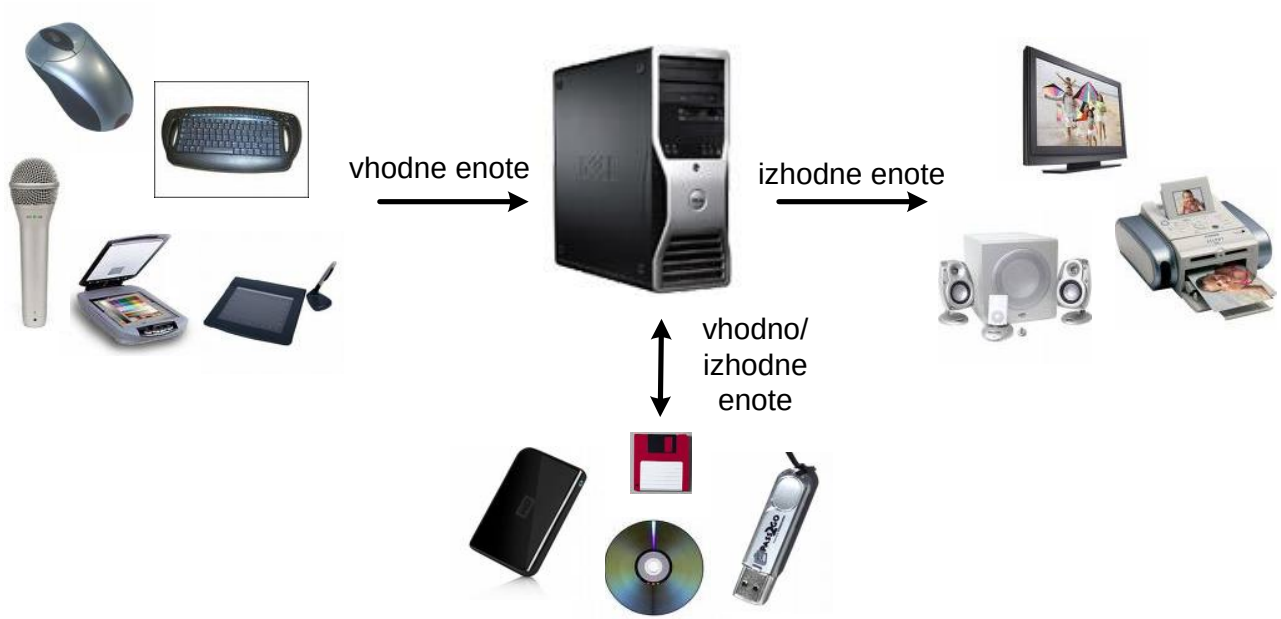


Predpomnilnik

- Von Neumannovo ozko grlo
  - pomnilnik RAM je počasen v primerjavi s procesorjem
  - podatkovne poti so ozke
- Hiter pomnilnik je zelo drag
- Rešitev: dvo-nivojski pomnilnik: RAM + predpomnilnik
- Princip lokalnosti
  - velika verjetnost je, da bo isti podatek kmalu ponovno uporabljen
  - velika verjetnost je, da do bodo kmalu uporabljeni podatki, ki so blizu (po naslovu) trenutno uporabljenemu podatku
- Uporaba predpomnilnika:
  1. Poglej v predpomnilnik in uporabi podatek, če je tam.
  2. Če ga ni, dostopaj do pomnilnika RAM.
  3. Kopiraj še *k* naslednjih podatkov iz pomnilnika RAM v predpomnilnik in zavrzi stare podatke iz njega.
- Tipično je predpomnilnik 5-10 krat hitrejši od RAMa
- Zelo pomemben je odstotek zadetkov podatkov že v predpomnilniku (cache hit rate)

Vhodno-izhodne naprave

- Dve skupini
  - Naprave namenjene prenosu informacij med CPE in glavnim pomnilnikom ter zunanjim svetom
    - Človekom: tipkovnica, zaslon, miška, tiskalnik, risalnik, zvočniki,…
    - Računalnikom: omrežna kartica, TK linija
  - Pomožni pomnilnik za shranjevanje podatkov
    - CD-ji, DVD-ji, magnetni trakovi, prenosni diski, ipd.



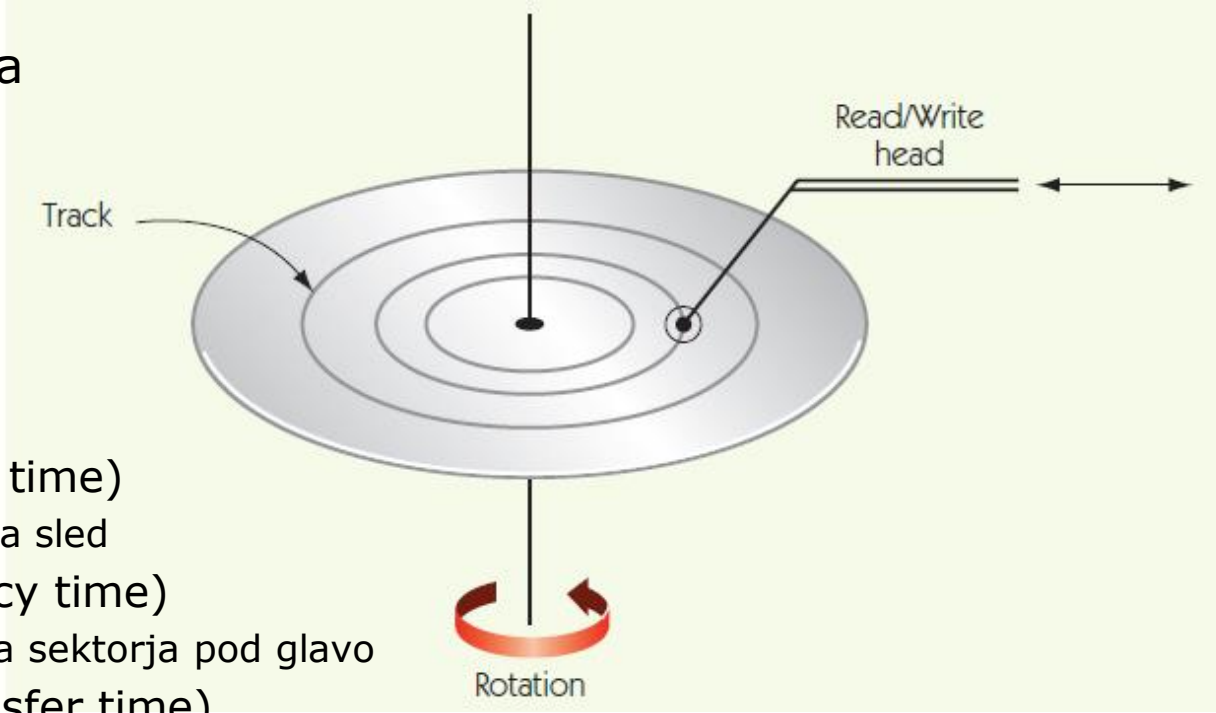
Zunanji pomnilnik

- Notranji pomnilnik:
  - RAM
    - netrajni pomnilnik, podatki se po izklopu računalnika izgubijo
    - vanj lahko pišemo in iz njega beremo
  - ROM: read-only-memory
    - lahko ga samo beremo
    - tovarniško trajno zapisani podatki in ukazi
    - BIOS

- Zunanji pomnilnik – trajni
  - naprave za shranjevanje z neposrednim dostopom
    - DASD (Direct Access Storage Devices)
    - vsaka lokacija ima naslov, ki ga lahko neposredno dosežemo
    - čas dostopa ni uniformen
    - trdi disk, CD, DVD, itn.
  - naprave za shranjevanje z zaporednim dostopom
    - SASD (Sequential access storage devices)
    - ne moremo dostopati do lokacij neposredno, ampak samo zaporedno
    - magnetni trak

Naprave z neposrednim dostopom

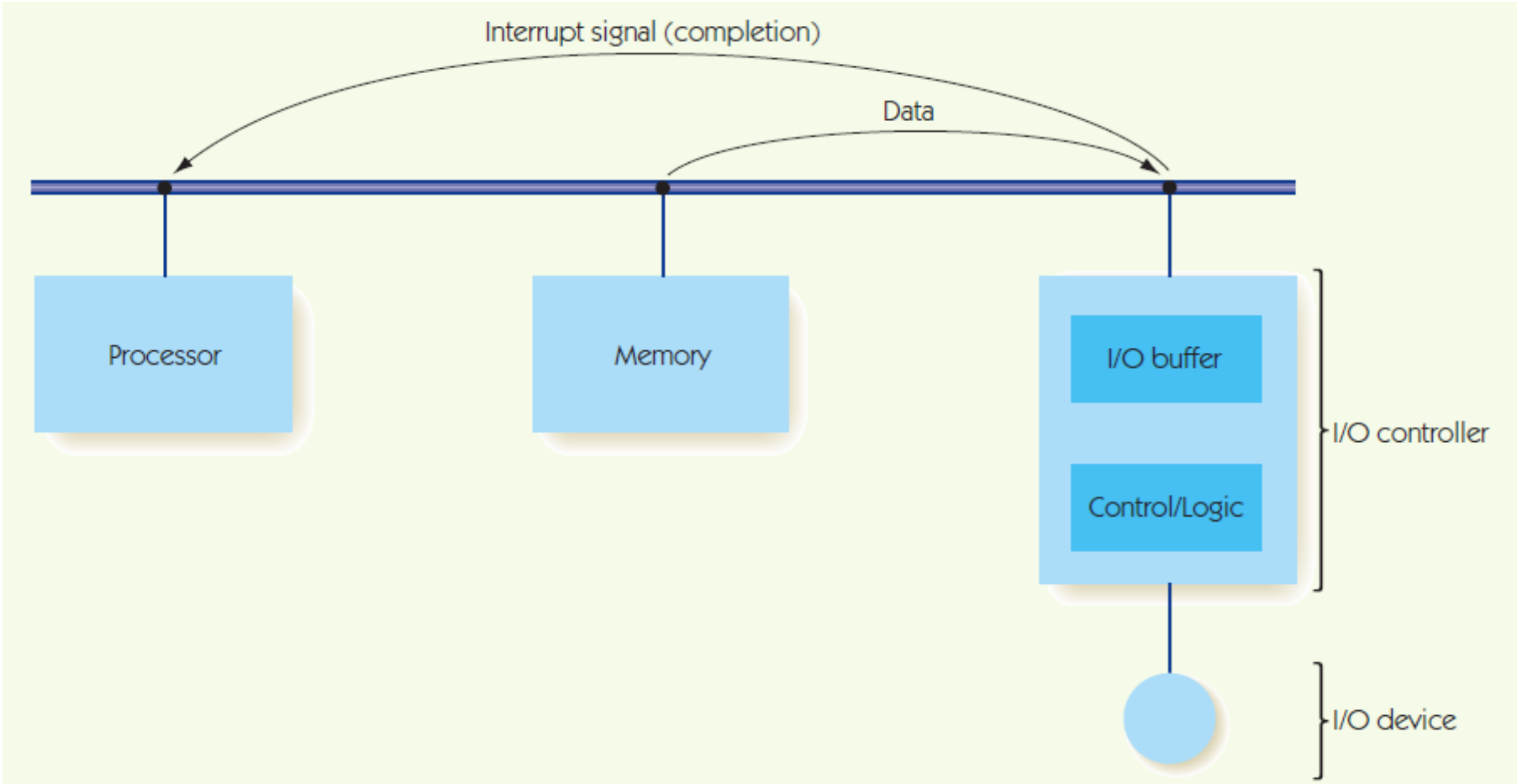
- Sestavni deli:
  - površina (surface) vsebuje več sledi
  - sledi (tracks): koncentrični krogi
  - sektorji (sectors): naslovljivi segmenti na sledi z vnaprej določeno dolžino
  - bralno pisalna glava



  - Čas dostopa:
    - čas iskanja (seek time)
      - premik glave na sled
    - zakasnitev (latency time)
      - rotacija začetka sektorja pod glavo
    - čas prenosa (transfer time)
      - rotacija celega sektorja pod glavo

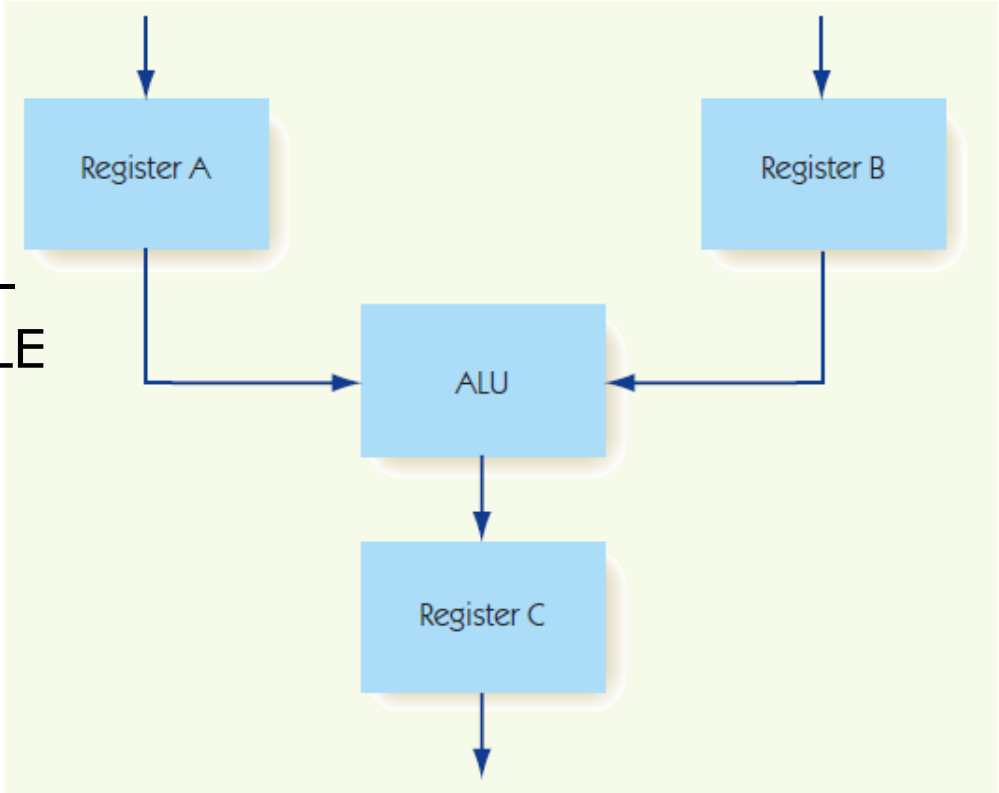
V/I krmilnik

- V/I naprave so več razredov počasnejše od RAMa
- V/I krmilnik
  - krmili V/I naprave, osvobodi procesor



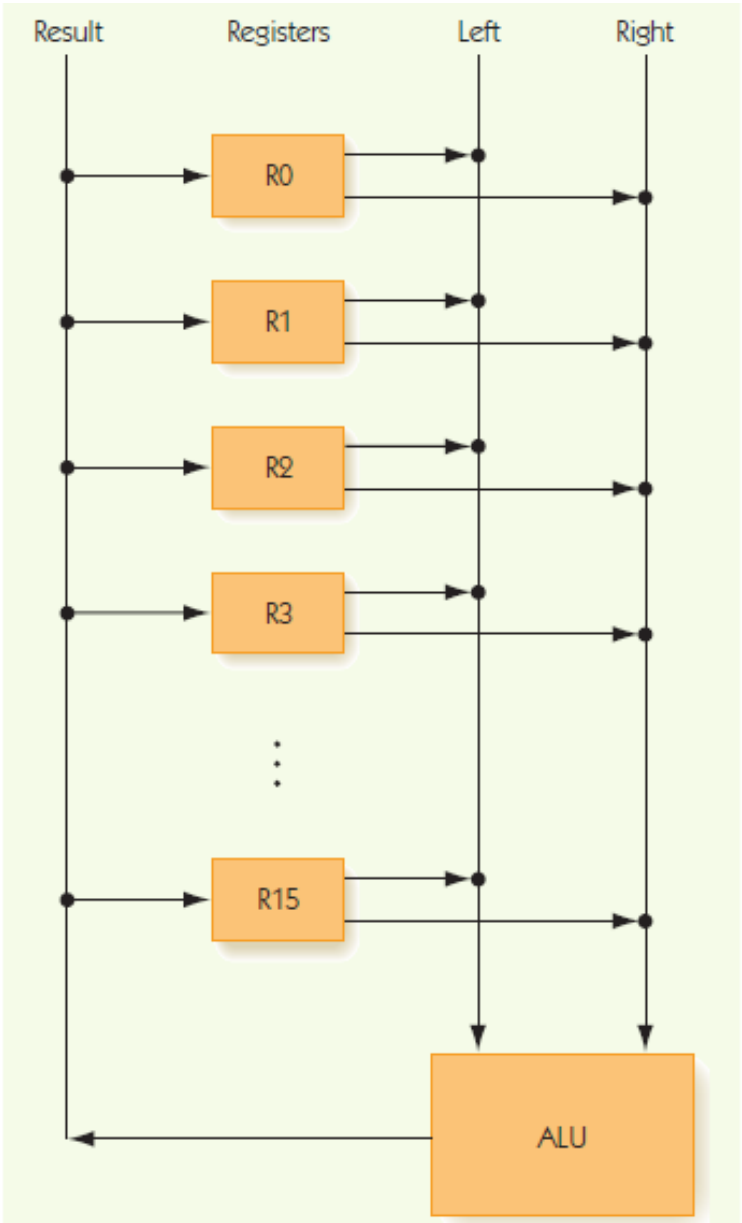
Aritmetično-logična enota

- ALE je del procesorja (skupaj s krmilno enoto)
- Vsebuje vezja
  - za aritmetiko: +, -, *, /
  - za primerjanje in logiko: =, IN, ALI, NE
- Vsebuje registre
  - izredno hitre namenske pomnilniške enote povezane z vezjem ALE
- Podatkovna pot (Data path)- kako potuje informacija v ALE
  - iz registrov na vezja
  - iz vezij nazaj na registre



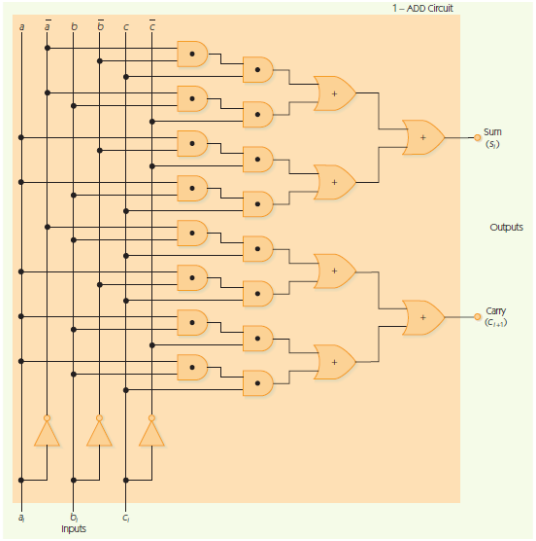
Organizacija ALE

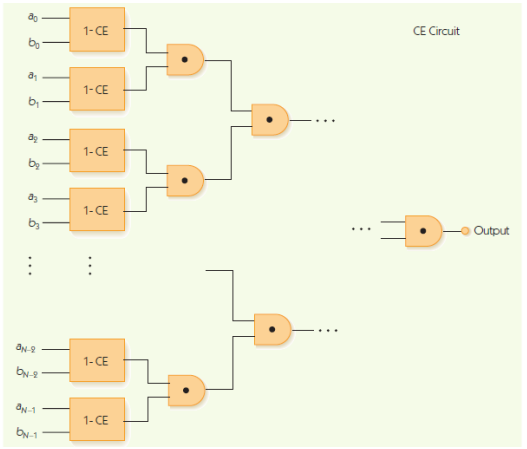
- Podatkovna pot s 16 registri



Vezja ALE

- Vezja za izvajanje operacij:
  - a+b
  - a=b
  - a-b
  - a*b, a/b
  - a<b, a>b
  - a AND b, a OR b, NOT a

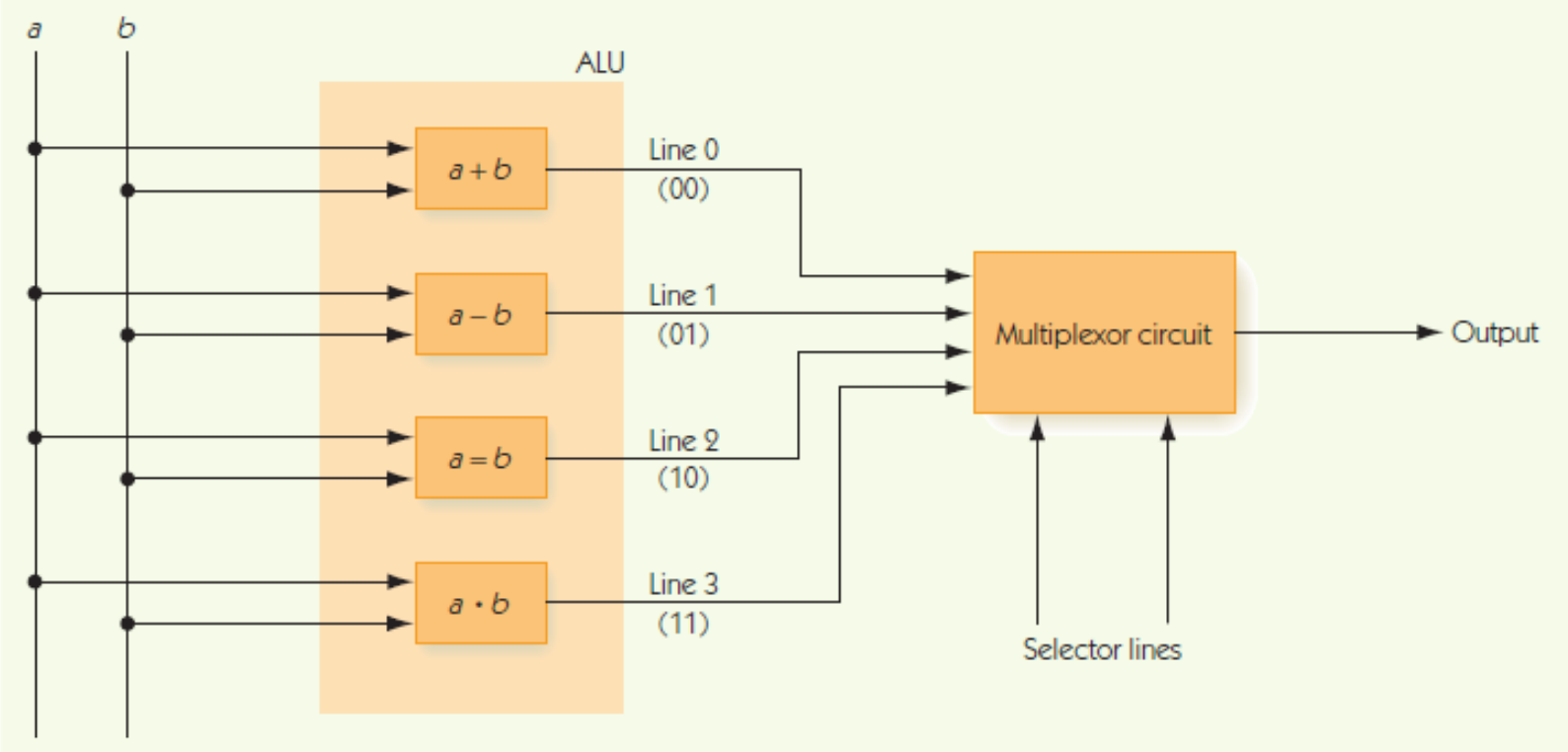




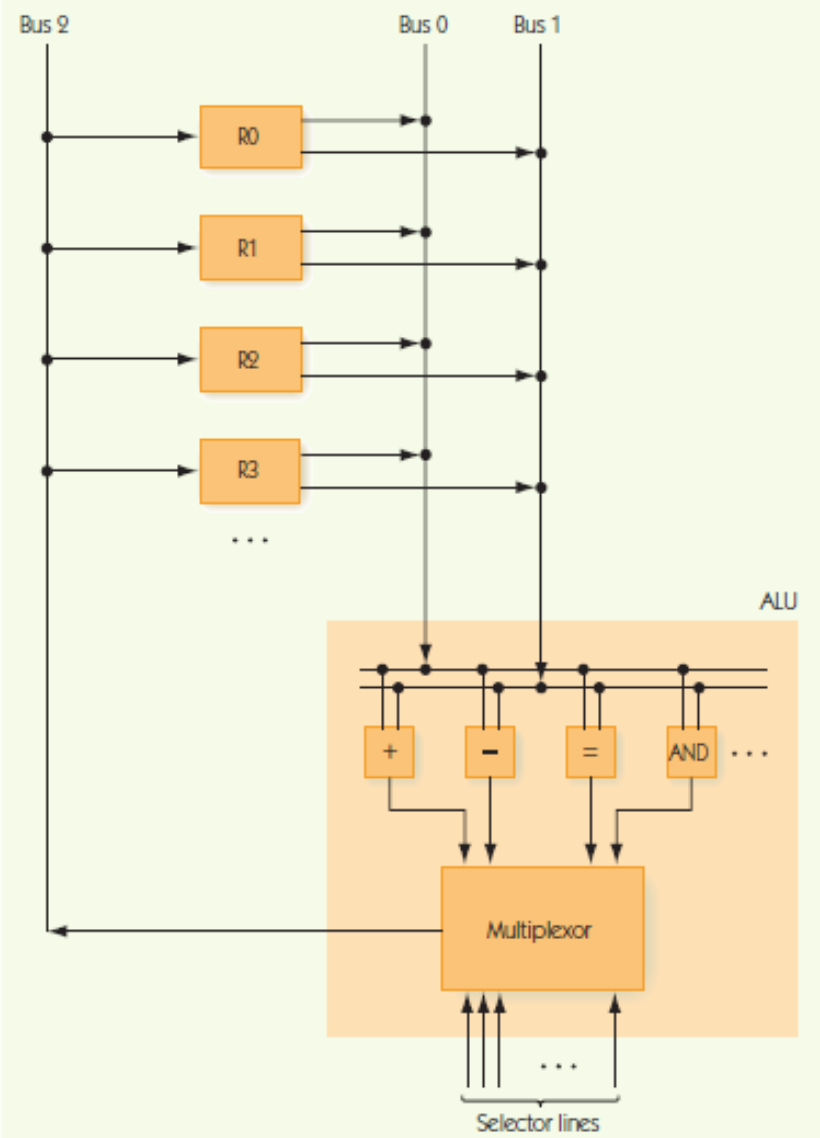
- Kako izbrati katero operacijo izvesti?
  - opcija 1: dekoder signalizira katero vezje naj se uporabi
  - opcija 2: požene se vsa vezja, z multiplekserjem se izbere pravi izhod (običajna implementacija)

ALE

- Uporaba multiplekserja za izbor pravega rezultata ALE
  1. Podatki pridejo (od zunaj) na registre ALE
  2. Signal pride na multiplekser in signalizira pravo operacijo
  3. Rezultat se zapiše nazaj v registre, od koder gre ven iz ALE



Organizacija ALE



Krmilna enota

- Program je shranjen v pomnilniku
  - Zaporedje ukazov v strojnem jeziku

- Krmilna enota
  - prebere iz pomnilnika naslednji ukaz, ki naj se izvede
  - dekodira ukaz (ugotovi kaj naj se stori)
  - izvede ukaz – pošlje ustrezni ukaz
    - ALE
    - pomnilniku, ali
    - V/I krmilnikom

Strojni ukazi

- Format ukaza v strojnem jeziku:
  - koda operacije
  - naslovi pomnilniških lokacij z operandi

primer št. bitov:	8	16	16

- primer:
  - op code: 9
  - naslov X: 99
  - naslov Y: 100
  - ADD X,Y: sešteje števili in zapiše rezultat nazaj na lokacijo Y
  - ADD X,Y: 00001001 0000000001100011 0000000001100100

Nabor strojnih ukazov

- Nabor ukazov (instruction set)
- Strojni ukazi so implementirani za posamezen čip
  - procesorji niso kompatibilni na nivoju strojnih ukazov
- Računalniki RISC
  - Reduced Instruction Set Computers
  - malo ukazov, ki se zelo hitro izvedejo
  - enostaven design strojne opreme
- Računalniki CISC
  - Complex Instruction Set Computers
  - velika množica ukazov, tudi bolj kompleksnih
  - kompleksen design strojne opreme
- Moderni računalniki: mešanica pristopov RISC in CISC
- Štiri skupine strojnih ukazov:
  - Prenos podatkov
  - Aritmetika
  - Primerjanje
  - Vejitve

Strojni ukazi

1. Prenos podatkov
   - prenos podatkov
     - pomnilniška lokacija -> register ALE
     - register ALE -> pomnilniška lokacija
     - ena pomnilnišpka lokacija -> druga pomnilniška lokacija
     - en register ALE -> drugi register ALE
   - Primeri:
     - LOAD X — naloži vsebino pomnilniške lokacije v register R
     - STORE X — shrani vsebino regostra R na omnilniško lokacijo X
     - MOVE X,Y — Kopiraj vsebino pomn. lokacije X na pomn. lokacijo Y

2. Aritmetika
   - Aritmetične in logične operacije v ALE: +, -, *, /, IN, ALI, NE
   - Primeri
     - ADD X,Y,Z — vr(Z)=vr(X)+vr(Y) — tro-naslovni ukaz
     - ADD X,Y — vr(Y)=vr(X)+vr(Y) — dvo-naslovni ukaz
     - ADD X — R=vr(X)+R — eno-naslovni ukaz

Strojni ukazi

3. Primerjanje
   - rezultat primerjanja postavi vrednosti bitov pogojnih kod
   - primer:
     - COMPARE X,Y — primerja vrednosti pomn. lokacij X in Y in postavi vrednosti pogojnih kod:

vr(X)>vr(Y)	GT=1, EQ=0, LT=0
vr(X)=vr(Y)	GT=0, EQ=1, LT=0
vr(X)<vr(Y)	GT=0, EQ=0, LT=1

4. Vejitve
   - spreminjanje normalnega zaporednega toka ukazov
   - tipično po ukazu za primerjanje
   - primeri:
     - JUMP X — vzemi naslednji ukaz s pomn. lokacije X
     - JUMPGT X — skoči samo, če je indikator GT postavljen na 1
     - JUMPGE X — skoči, če sta GT ali EQ indikatorja postavljena na 1
     - HALT — ustavi izvajanje programa

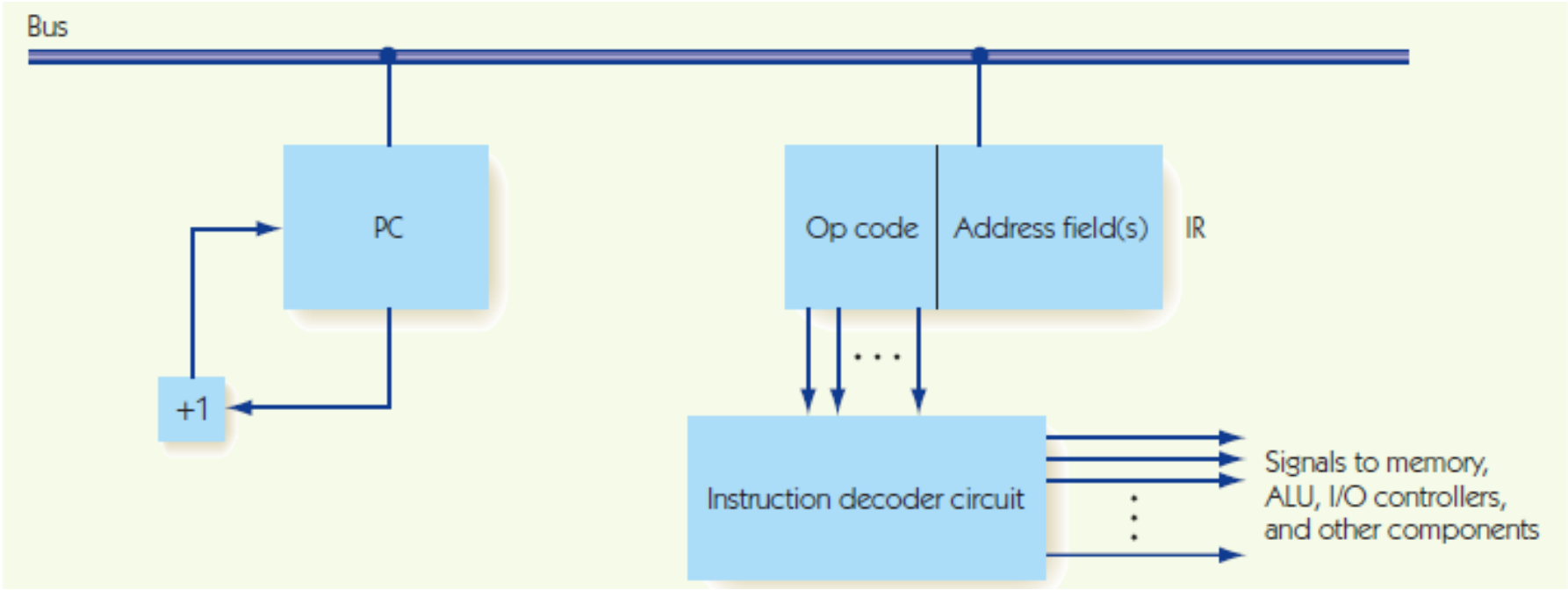
Primer zaporedja strojnih ukazov

Address	Contents
100	Value of *a*
101	Value of *b*
102	Value of *c*

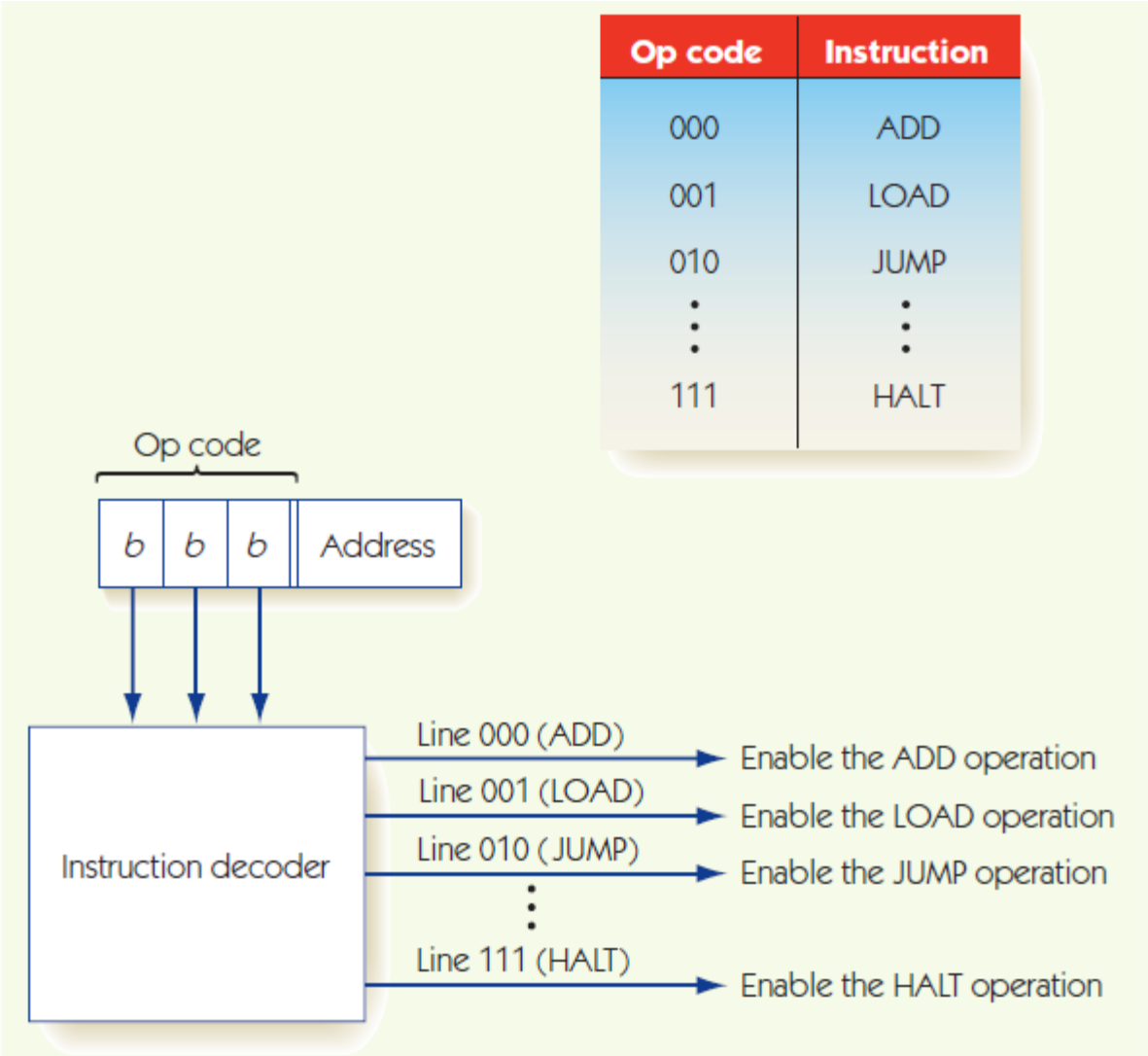
Algorithmic notation	Address	Contents	(Commentary)
		⋮	
1. Set *a* to the value *b* + *c*	50	LOAD 101	Put the value of *b* into register R.
	51	ADD 102	Add *c* to register R. It now holds *b* + *c*.
	52	STORE 100	Store the contents of register R into *a*.
2. If *a* > *b* then	50	COMPARE 100, 101	Compare *a* and *b* and set condition codes.
set *c* to the value *a*	51	JUMPGT 54	Go to location 54 if *a* > *b*.
Else	52	MOVE 101, 102	Get here if *a* ≤ *b*, so move *b* into *c*
set *c* to the value *b*	53	JUMP 55	and skip the next instruction.
	54	MOVE 100, 102	Move *a* into *c*.
	55	• • •	Next statement begins here.

Registri in vezja krmilne enote

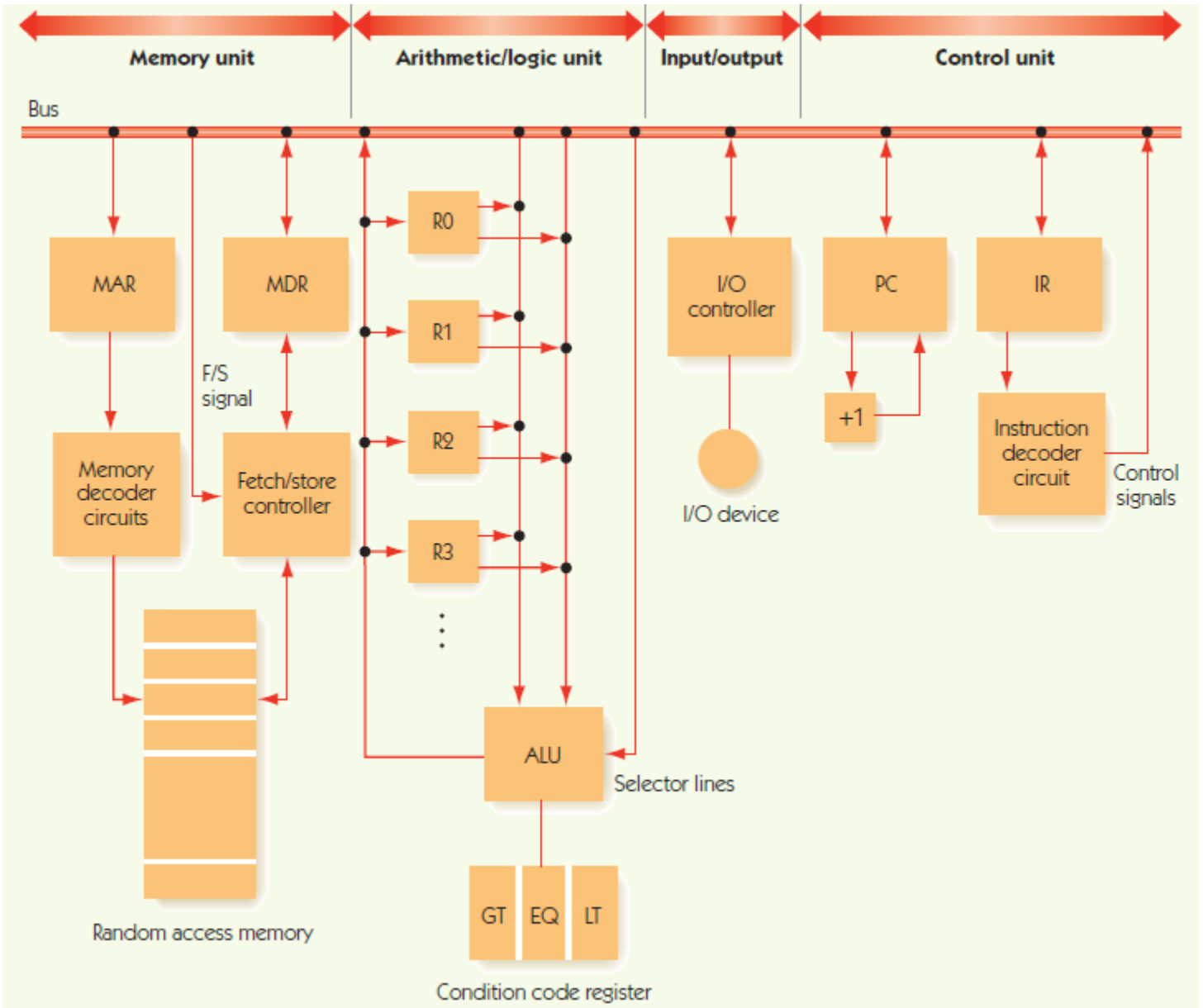
- Programski števec (program counter)
  - vsebuje naslov naslednjega ukaza
- Ukazni register (instruction register)
  - vsebuje kodo trenutnega ukaza
- Vezje za dekodiranje ukaza
  - dekodira kodo ukaza in pošlje signale vezjem za posamezne ukaze



Dekodirnik ukazov



Von Neumannova arhitektura



Von Neumannov cikel

- Cikel naloži-dekodiraj-izvedi:
  1. Faza nalaganja
     - naloži naslednji ukaz iz pomnilnika
  2. Faza dekodiranja ukaza
  3. Faza izvajanja
     - drugačna za vsak ukaz

```
while ukaz ni HALT ali napaka
        Faza nalaganja ukaza
        Faza dekodiranja ukaza
        Faza izvajanja ukaza
end while
```

Primer

- Notacija:

CON(A)	Contents of memory cell A
A -> B	Send value in register A to register B (special registers: PC, MAR, MDR, IR, ALU, R, GT, EQ, LT, +1)
FETCH	Initiate a memory fetch operation
STORE	Initiate a memory store operation
ADD	Instruct the ALU to select the output of the adder circuit
SUBTRACT	Instruct the ALU to select the output of the subtract circuit

Von Neumannov cikel

- Faza nalaganja
  - enaka za vse ukaze:
  1. PC -> MAR
  2. FETCH
  3. MDR -> IR
  4. PC + 1 -> PC

- Faza dekodiranja
  - enka za vse ukaze
  1. IR_{op} -> instruction decoder

- Faza izvajanja
  - za vsak ukaz različna

Nabor ukazov: primer

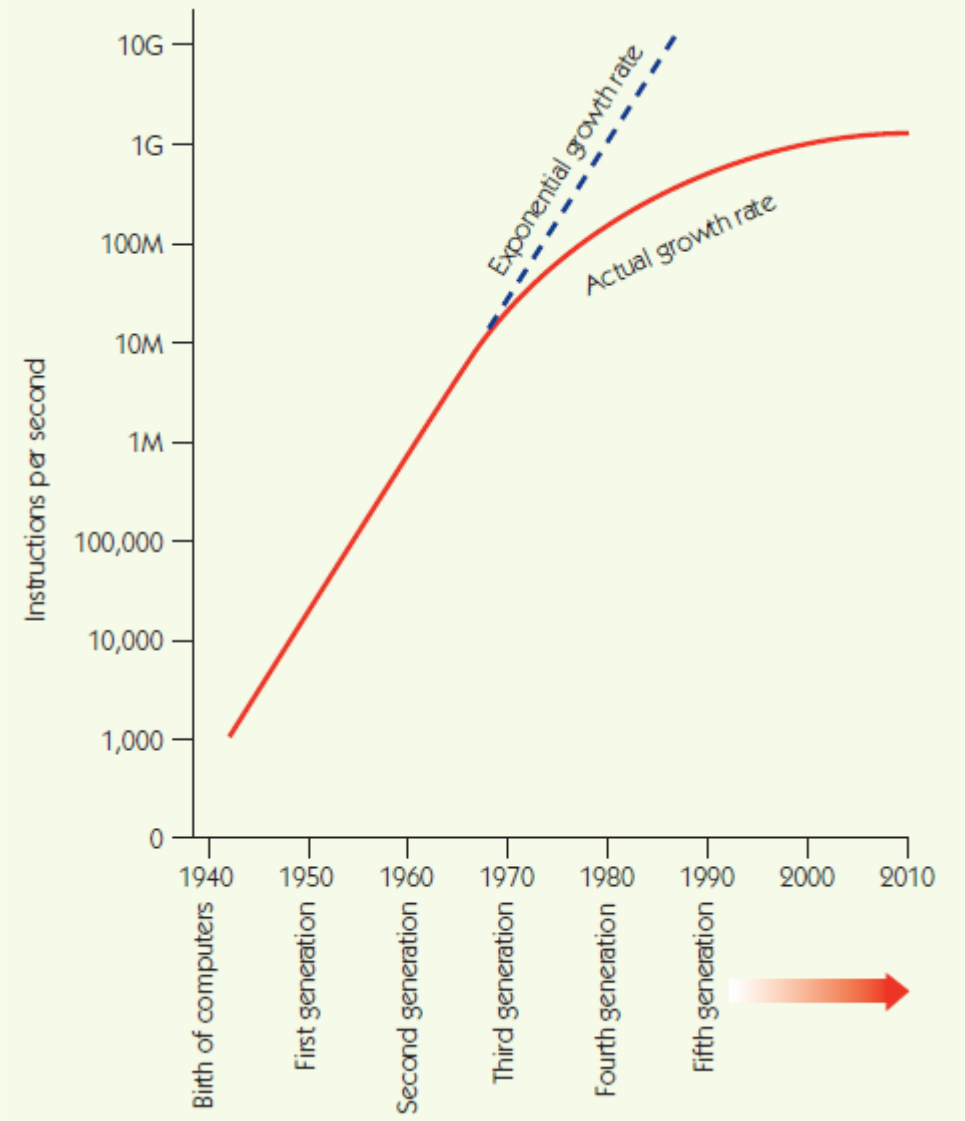
Binary Op Code	Operation	Meaning
0000	LOAD X	CON(X) → R
0001	STORE X	R → CON(X)
0010	CLEAR X	0 → CON(X)
0011	ADD X	R + CON(X) → R
0100	INCREMENT X	CON(X) + 1 → CON(X)
0101	SUBTRACT X	R – CON(X) → R
0110	DECREMENT X	CON(X) – 1 → CON(X)
0111	COMPARE X	if CON(X) > R then GT = 1 else 0 if CON(X) = R then EQ = 1 else 0 if CON(X) < R then LT = 1 else 0
1000	JUMP X	Get the next instruction from memory location X.
1001	JUMPGT X	Get the next instruction from memory location X if GT = 1.
1010	JUMPEQ X	Get the next instruction from memory location X if EQ = 1.
1011	JUMPLT X	Get the next instruction from memory location X if LT = 1.
1100	JUMPNEQ X	Get the next instruction from memory location X if EQ = 0.
1101	IN X	Input an integer value from the standard input device and store into memory cell X.
1110	OUT X	Output, in decimal notation, the value stored in memory cell X.
1111	HALT	Stop program execution.

Izvajanje ukazov: primeri

- LOAD X
  1. IR_{ADDR} -> MAR
  2. FETCH
  3. MDR -> R
- STORE X
  1. IR_{ADDR} -> MAR
  2. R -> MDR
  3. STORE
- ADD X
  1. IR_{ADDR} -> MAR
  2. FETCH
  3. MDR -> ALU
  4. R -> ALU
  5. ADD
  6. ALU -> R
- JUMP X
  1. IR_{ADDR} -> PC
- COMPARE X
  1. IR_{ADDR} -> MAR
  2. FETCH
  3. MDR -> ALU
  4. R -> ALU
  5. SUBTRACT
- JUMPGT
  1. IF GT=1 THEN
     IR_{ADDR} -> PC

Ne-Von Neumannove arhitekture

- Rast hitrosti procesorjev ni več eksponentna
- Fizikalne omejitve
  - hitrost svetlobe
  - minimalna bližina vrat
  - segrevanje vezja
- Problemi, ki jih rešujemo so vse večji
- Von Neumanovo ozko grlo



Paralelno procesiranje

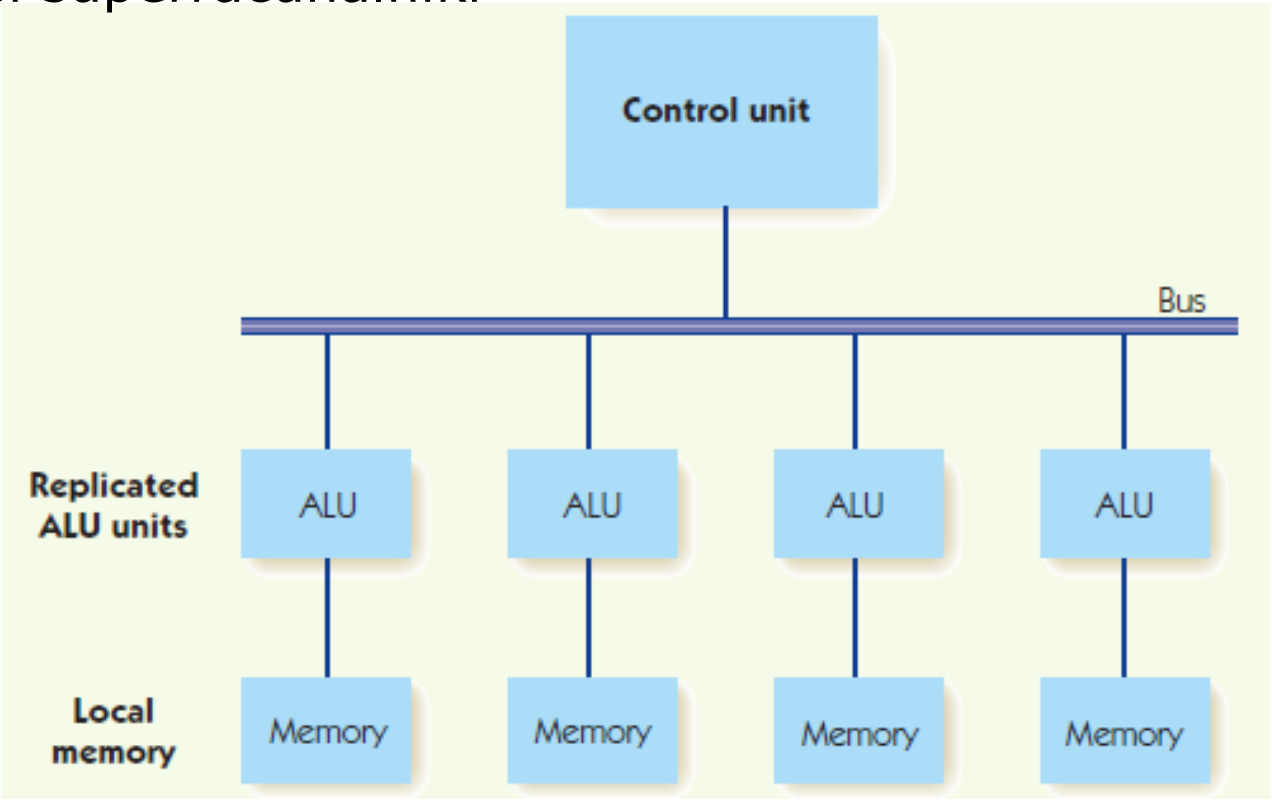
"Če ne zmoreš zgraditi naprave, ki bi delala dvakrat hitreje, zgradi napravo, ki bo naredila dve stvari hkrati. Rezultat bo identičen."

- Rešitev je paralelno procesiranje
- Večje število procesorjev
  - v preteklosti superračunalniki
  - več-jedrni procesorji (dual-core, quad-core)
  - na desetine, stotine, tisočine procesorjev

- Dva pristopa:
  - SIMD
    - Single Instruction, Multiple Data Stream
  - MIMD
    - Multiple instruction, Multiple Data Stream

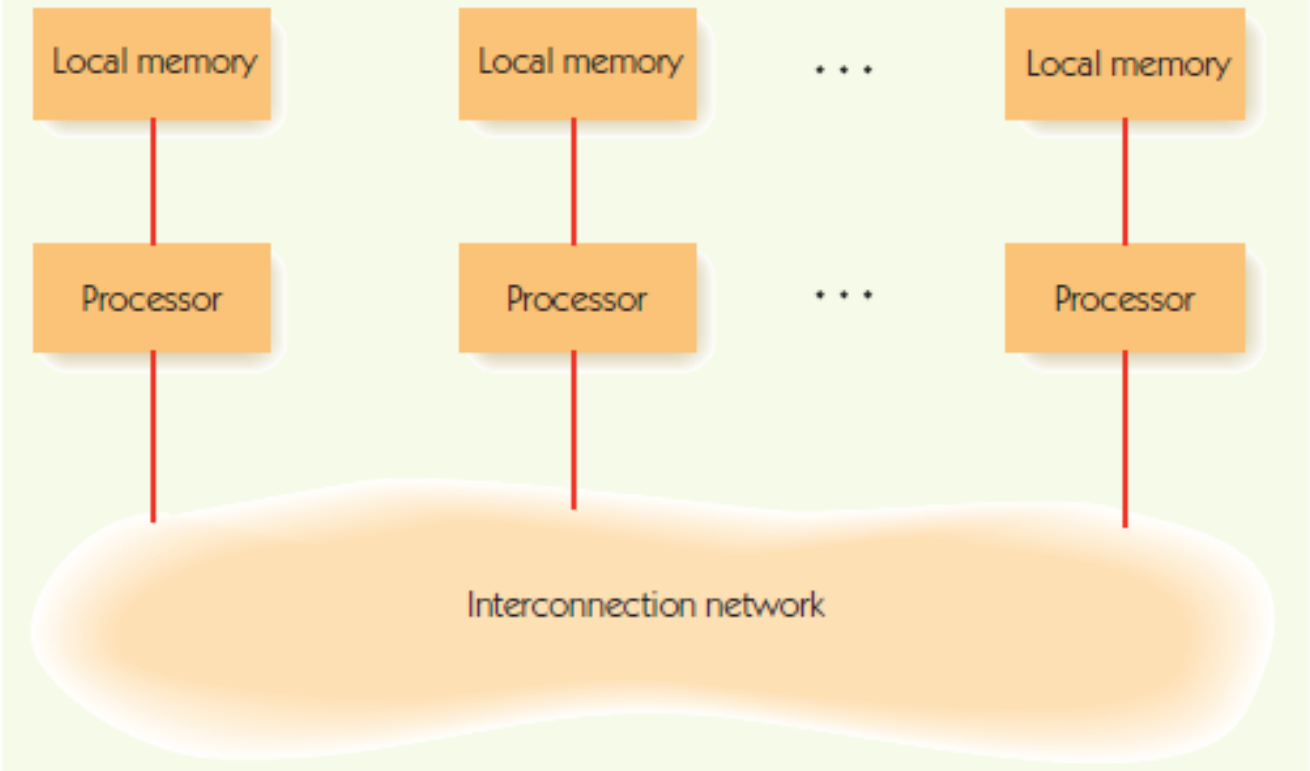
Model SIMD

- En ukaz, več podatkovnih tokov
- Računalnik izvrši hkrati natančno en ukaz na več podatkih
- Ena krmilna enota, več ALE (vsaka ALE dela na svojih podatkih)
- Vektorske operacije
- Starejši superračunalniki



Model MIMD

- Več ukaznih tokov, več podatkovnih tokov
- Računalnik izvrši hkrati več ukazov nad različnimi podatki
- Razmnoženi procesorji, vsak opravlja svoje delo
- Komunikacija lahko upočasni delovanje



Model MIMD

- Navadi procesorji primerni
- Skalabilnost: vedno lahko dodajamo nove procesorje

- Različni sistemi MIMD
  - namenski sistemi; novejši superračunalniki
  - gruče računalnikov (cluster computing)
    - standardni računalniki povezani preko LAN ali WAN
  - grid computing
    - različni računalniki, oddaljeni, povezani preko Interneta
    - primer SETI@home

- Velik izziv: paralelni algoritmi
  - da se zagotovi izkoriščenost velikega števila procesorjev

Povzetek

- Abstrakcija kompleksnosti sistemov
- Von Neumanova arhitektura standard modernega računalništva
  - pomnilnik, V/I, ALE, krmilna enota
  - programi shranjeni v pomnilniku
  - zaporedno izvajanje ukazov
- Pomnilnik sestavljen iz naslovljivih pomnilniških lokacij
- Zunanji pomnilnik je trajen; diski, sledi, sektorji
- V/I enote so počasne -> V/I krmilniki
- ALE izvaja računanja, premika podatke v/iz registrov
- Krmilna enota nalaga, dekodira in izvaja ukaze
- Ukazi so zapisani v strojnem jeziku
- Arhitekture za paralelno procesiranje omogočajo izvajanje več ukazov hkrati