Še nekaj principov delovanja računalnikov

(Nadaljevanje poglavja Osnovni principi delovanja računalnikov)

Vhod in izhod

Osnovna naloga V/I sistema je pretvorba informacije iz ene oblike v drugo

- izjema so naprave za shranjevanje informacije, ki tudi spadajo v to skupino
  - rečemo jim pomožni pomnilniki (npr. magnetni disk, optični disk, magnetni trak)
    - cena, obstojnost informacije

Osnovni način delovanja V/I sistema je prenos podatkov

- med GP in V/I napravami ali
- med CPE in napravami

Razlike med rač. glede izvedbe V/I so velike

- pri znanstvenem računanju malo V/I prenosov
- pri poslovnem veliko

2 skupini izvedb V/I sistema :

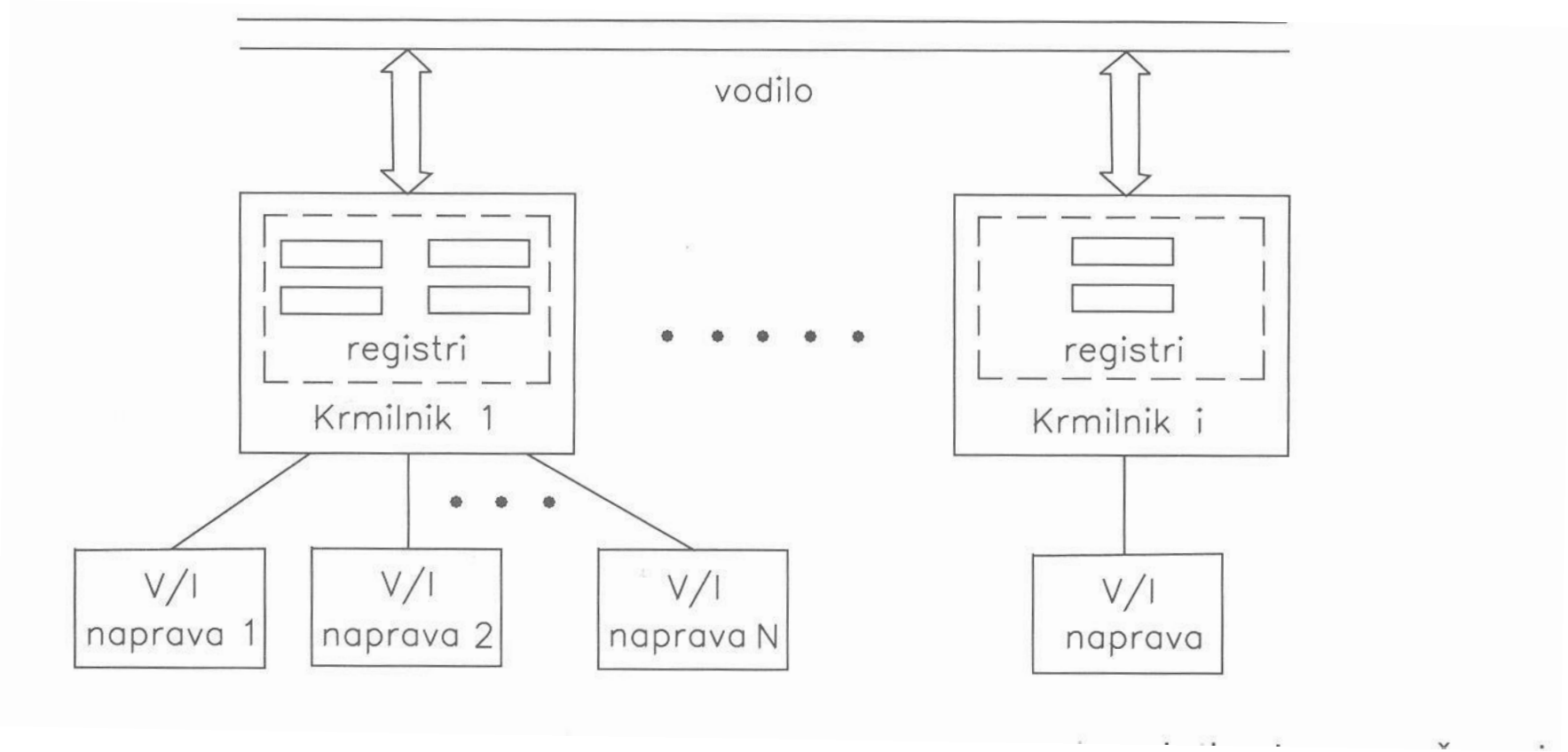
1. **Programski vhod/izhod** (**programmed I/O**)
   - z V/I napravo komunicira CPE
   - vsak podatek se prenese iz GP v CPE in nato v napravo ali obratno
   - prenos je realiziran z zaporedjem ukazov
   - hiba je počasnost in zasedenost CPE
2. **Neposredni dostop do pomnilnika** (**direct memory access - DMA**)
   - naprava komunicira neposredno z GP
   - zato rabimo **DMA krmilnik**, ki nadomesti CPE
   - posebna izvedba DMA krmilnikov so **vhodno/izhodni procesorji**

Pri mnogih računalnikih srečamo oba načina dostopa

- za počasne naprave je primeren programski vhod/izhod
- za hitre oz. podatkovno zahtevne je nujen DMA, ker bi bil programski prepočasen

Vsaka V/I naprava je priključena preko **krmilnika naprave** (device controller)

- vezje, ki omogoča prenos podatkov v napravo in iz nje
  - lahko preprost (register), lahko kompliciran (specializiran računalnik)
- na nekatere krmilnike je mogoče priključiti več naprav
- s krmilnikom komuniciramo preko njegovih registrov
  - pisanje in branje pri njih sproži neko operacijo v napravi ali odraža stanje po prejšnji operaciji
    - npr. s pisanjem v ukazni register krmilnika magnetnega diska dosežemo premik bralno-pisalne glave na določeno sled
    - z branjem statusnega registra pa lahko ugotovimo, kdaj je premik končan



Krmilniki vhodno/izhodnih naprav

Registri krmilnikov so lahko v istem naslovnem prostoru kot GP, lahko pa v posebnem

Ločimo 3 izvedbe:

1. **Pomnilniško preslikan vhod/izhod** (**memory mapped I/O**)
   - registri krmilnikov so v pomnilniškem naslovnem prostoru
   - iz CPE so videti kot pomnilniške lokacije
   - iz njih bere in vanje piše z ukazi za dostop do pom.
   - ni posebnih V/I ukazov
2. **Ločen vhodno/izhodni prostor**
   - registri krmilnikov so v posebnem naslovnem prostoru
   - za dostop do registrov so potrebni *posebni V/I ukazi*
   - pri tem CPE aktivira tudi določen(e) signal(e), ki pove(jo), da se naslavlja V/I naslovni prostor
3. **Posredno preko vhodno/izhodnih procesorjev**
   - tudi tu so registri krmilnikov v posebnem naslovnem prostoru, ki pa iz CPE ni neposredno dostopen
   - vmes so še vhodno/izhodni procesorji (razbremenijo CPE)
   - pri velikih računalnikih

Vzporedni (paralelni) računalniki

Von Neumann: zaporedno izvajanje ukazov

Mnogi problemi po svoji naravi dovoljujejo istočasno oz. paralelno izvajanje več operacij

Zato so von Neumann-ov model razširili

Flynn-ova klasifikacija (1966) uporablja 2 kriterija:

- tok ukazov (instruction stream): koliko ukazov se izvršuje naenkrat
- tok podatkov (data stream): koliko ponovitev operandov* en ukaz obdeluje naenkrat

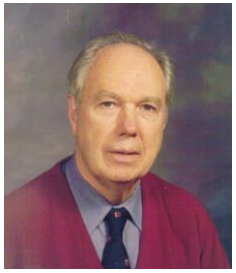
Npr.,

ADD A1, A2, A3 ; A1 $\leftarrow$ A2 + A3

N paralelnih ponovitev:

A1(*i*) $\leftarrow$ A2(*i*) + A3(*i*), *i* = 1, ..., *N*

Flynn-ova klasifikacija



Flynn-ova klasifikacija računalnikov:

- **SISD** (Single Instruction stream, Single Data stream)
  - izvajajo naenkrat en ukaz na eni zbirki operandov
  - najbolj zmogljivi so vektorski računalniki
- **SIMD** (Single Instruction stream, Multiple Data stream)
  - izvajajo en ukaz na več zbirkah operandov (*N*)
  - imajo eno kontrolno enoto in *N* ALE ter *N* množic registrov
- **MISD** (Multiple Instruction stream, Single Data stream)
  - ne obstajajo (najbližje temu so 'stream' procesorji, ki podobno cevovodu izvajajo različne operacije na istem toku ukazov)
- **MIMD** (Multiple Instruction stream, Multiple Data stream)
  - izvajajo več ukazov na več zbirkah operandov
  - multiprocesorji, multiračunalniki

MIMD: več CPE

- *tesno povezani* (tudi shared memory): skupen pomnilnik
- *rahlo povezani* (tudi distributed memory): povezani preko V/I enot

Večjedrne (multicore) računalnike (več CPE na istem čipu) lahko štejemo med tesno povezane MIMD

- "pravi" oz. veliki MIMD pa imajo po več tisoč jeder (rekord je trenutno nekaj milijonov)

SPMD: Single Program – Multiple Data:

- Programi na MIMD pogosto tečejo tako, da isti program teče na več procesorjih (oz. jedrih), pogojni stavki pa določajo, kaj se izvaja na posameznem procesorju

SIMD in MIMD so **paralelni računalniki**

- najbolj zmogljivi superračunalniki so paralelni
- zmogljivost se običajno meri v številu operacij v plavajoči vejici na sekundo
  - GFLOPS (Giga FLOPS – Floating Point Operations Per Second) pomeni 10^9 operacij / s
    - Cray 1988, 1GFLOPS
  - TFLOPS (Tera FLOPS) pomeni 10^{12} operacij / s
  - PFLOPS (Peta FLOPS) pomeni 10^{15} operacij / s
    - trenutno je rekord 34 PFLOPS
- od leta 1988 se povečuje zmogljivost za 2x na leto
- današnji PCji: nekaj GFLOPS

Amdahlov zakon

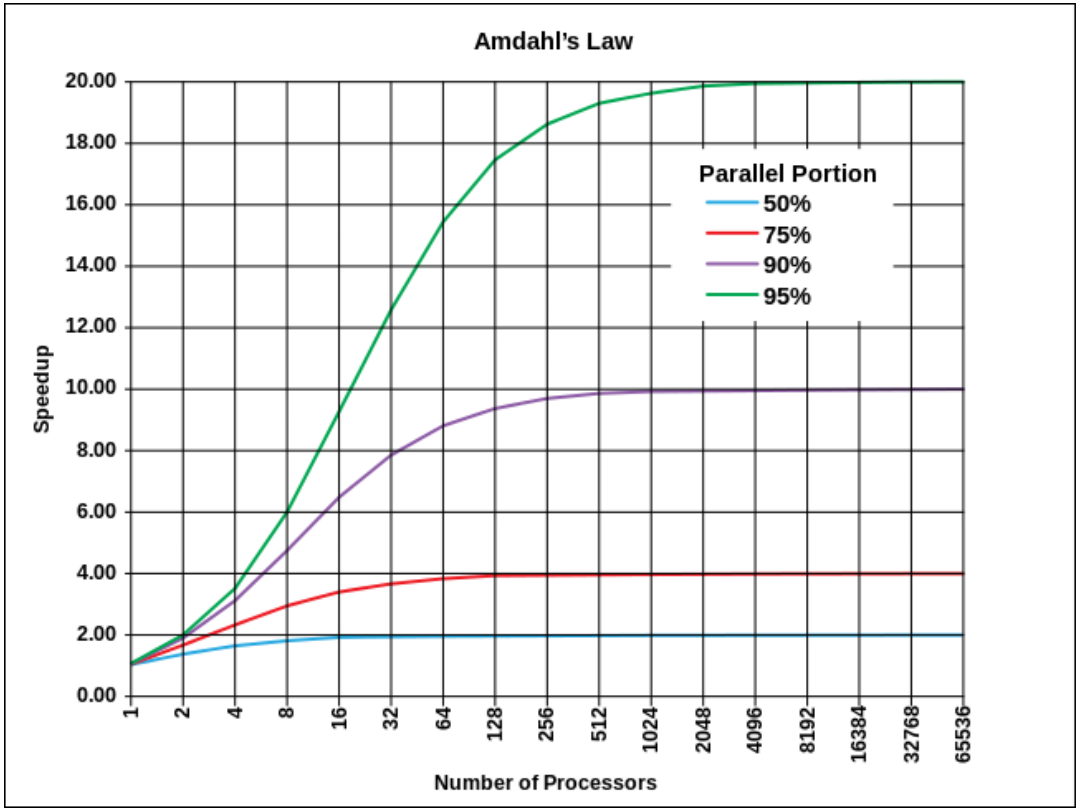
Vzemimo, da pohitrimo delovanje določenega dela operacij

- *f* je *zaporedni del(ež)* programa
- *1 - f* je *vzporedni del(ež)* programa
  - pri njem je delovanje *N*-krat hitrejše (npr. paralelno izvajanje *N* procesorjev)

Povečanje hitrosti računalnika je tedaj (Gene Amdahl, 1967):

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

- npr. če je $f = 0{,}1$, hitrosti računalnika ne moremo povečati za več kot 10-krat, tudi če preostalih 90% časa zmanjšamo na 0 (pohitrimo za faktor $N \to \infty$)
- koliko nam paralelni računalnik koristi, je odvisno od problema



Gustafsonov zakon

Gustafsonov zakon:

- lahko pa rešimo večji problem
- če povečujemo problem, se zaporedni del *f* zmanjšuje in pohitritev postane skoraj linearna:

  $S(N) \approx N$

Gustafsonov zakon je poskus, da se obide omejitve, ki jih postavlja Amdahlov zakon

- ne morem te prepeljati hitreje, lahko pa vas gre 5
- ni vedno možno ☹

Računalnik kot zaporedje navideznih računalnikov

Večine uporabnikov arhitektura računalnika (pravzaprav) posebno ne zanima

- programske jezike lahko implementiramo na različnih računalnikih

Tanenbaum, 1984:

- Računalnik kot zaporedje navideznih računalnikov
- Vsak nivo si lahko predstavljamo kot navidezni računalnik, ki ima za “strojni” jezik kar jezik tega nivoja (večina uporabnikov se spodnjih nivojev niti ne zaveda)

6 nivojev:

Nivo 5: Višji prog. jezik
- prevajanje ali interpretiranje

Nivo 4: Zbirni jezik
- prevajanje

Nivo 3: Operacijski sistem
- interpretiranje

Nivo 2: Strojni jezik
- interpretiranje

Nivo 1: Mikroprogramski jezik
- interpretiranje

Nivo 0: Digitalna logika

2 mehanizma za prehod med nivojema:

- Prevajanje (prevajalnik)
  - izvorni program v enem jeziku
  - ciljni program (object program) v drugem (nižjem) jeziku
    - izvornega načelno ne rabimo več
- Interpretacija (interpreter)
  - izvorni program se prevaja sproti
    - ukaz se prevede in izvrši
    - rabimo ga ves čas
    - bolj fleksibilno
    - večja prenosljivost
    - manjša hitrost
- delno prevajanje
  - prevajanje v vmesno kodo, ki se jo interpretira
    - npr. Java

Strojna in programska oprema računalnika

Delitev

- hardware
- software
- firmware
  - program, ki je vgrajen v HW napravo (kot ROM ali bliskovni pomnilnik) in skrbi za njeno osnovno funkcionalnost

Strojna in programska oprema sta funcionalno ekvivalentni

- poljuben računalnik bi se načeloma dalo realizirati samo z elektroniko (dovolj kompleksno)