

FER university of zagreb  
faculty of electrical engineering and computing

Prof.dr.sc. Slobodan Ribarić

**Višeprocorski sustavi, višezvezgani i grafički procesori**

1. Oblici i razine paralelizma
2. Paralelne arhitekture
3. Višeprocorski SIMD sustavi
4. Vektorski procesori
5. Multiprocorski sustavi – višeprocorski MIMD sustavi
6. Koherencija priručne memorije u multiprocorskom sustavu
7. Sinkronizacija procesa i dretvi
8. Višezvezgani procesori
9. Grafički procesori

FER university of zagreb  
faculty of electrical engineering and computing

**1. Oblici i razine paralelizma**

- do osamdesetih godina prošlog stoljeća – prevladavao paralelizam na bitovnoj razini (engl. bit-level):
  - povećanje duljine riječi procesora – 4-bitni (mikro)procesor, 8-bitni, 16-bitni, 32-bitni, ...
  - glavni motivi: povećanje adresirljivog prostora, poboljšani prikaz brojeva s pomičnim zarezom
- od devedesetih godina – paralelizam na razini instrukcija
  - ILP – instruction-level parallelism:
    - protočnost (engl. pipelining)
    - superskalarnost

FER university of zagreb  
faculty of electrical engineering and computing

ILP

Paralelne instrukcije:

- ako su dvije instrukcije paralelne one se mogu izvršiti istodobno u protočnoj strukturi bez izazivanja zastoja (engl. stall) uz pretpostavku da protočna struktura ima dovoljno resursa, tj. da ne postoji strukturni hazard.

Vrijedi:

- dvije instrukcije koje su ovisne nisu paralelne,
- dvjema ovisnima instrukcijama ne može se promijeniti redosljed izvođenja,
- instrukcije kojima se može promijeniti redosljed izvođenja su paralelne instrukcije

FER university of zagreb  
faculty of electrical engineering and computing

Vrste ovisnosti:

1. Podatkovna ovisnost

Neka instrukcija  $j$  podatkovno ovisna od instrukcije  $i$  ako vrijedi:

- instrukcija  $i$  generira rezultat koji koristi instrukcija  $j$ ,  
ili
- instrukcija  $j$  je podatkovno ovisna od instrukcije  $k$ , a instrukcija  $k$  je podatkovno ovisna od instrukcije  $i$  (tzv. *tranzitivna ovisnost*)

RAW – Read After Write hazard

FER university of zagreb  
faculty of electrical engineering and computing

Primjer:

```

i      loop: load F0, 0(R1)
j      add F4, F0, F2
        store 0(R1), F4
        .
i'     subi R1, R1, 8
j'     bnez R1, loop

```

Prikaz zavisnosti pomoću grafa zavisnosti (engl. dependence graph)

FER university of zagreb  
faculty of electrical engineering and computing

## 2. Imenska ovisnost (engl. name dependence)

- događa se kad dvije instrukcije koriste isti registar ili memorijsku lokaciju ali kada ne postoji tok podataka između te dvije instrukcije

Pretpostavimo da instrukcija *i* prethodi instrukciji *j* programskom slijedu;

Dvije vrste imenske ovisnosti:

- antiovisnost: instrukcija *j* je antiovisna od instrukcije *i* ako izlaz instrukcije *j* "prekrije" ulaz instrukcije *i* (WAR)
- izlazna ovisnost: javlja se kada instrukcija *i* i instrukcija *j* pišu u isti registar ili memorijsku lokaciju (WAW)

FER university of zagreb  
faculty of electrical engineering and computing

Primjer:

```

i1      load R1, A
i2      add R2, R1
i3      add R3, R4
i4      mul R4, R5
i5      com R6 ; jedinični komplement sadržaja reg. R6
i6      mul R6, R7 ; R6*R7 -> R6

```

Detektirajmo ovisnosti:

```

i1 - i2      podatkovna ovisnost
i3 - i4      antiovisnost
i5 - i6      podatkovna ovisnost, izlazna ovisnost

```

FER university of zagreb  
faculty of electrical engineering and computing

## 3. Upravljačka ovisnost

- odnosi se na situacije gdje se redoslijed izvršavanja instrukcija ne može odrediti prije samog trenutka izvođenja (engl. run time);

Primjer:

```

if p1 {
    i1;
};
if p2 {
    i2;
};

```

- instrukcija *i1* upravljački je ovisna o uvjetu (ili ishodu procesa) *p1*
- instrukcija *i2* upravljački je ovisna o uvjetu (ili ishodu procesa) *p2*

FER university of zagreb  
faculty of electrical engineering and computing

Ograničenja u vezi upravljačke ovisnosti:

- instrukcija koja je upravljački ovisna o grananju ne može se preseliti na mjesto prije grananja;
- instrukcija koja ne ovisi o grananju ne može se preseliti na mjesto poslije grananja

FER university of zagreb  
faculty of electrical engineering and computing

Paralelizam:

1. raspoloživi paralelizam u programima
2. iskorišteni paralelizam

Raspoloživi paralelizam – sadržan je u karakteru samog problema i programskom rješenju

- raspoloživi funkcijski paralelizam – očituje se u logičkom rješenju problema i javlja se u formalnim opisima rješenja (npr. dijagram toka, grafovi toka podataka)
- raspoloživi podatkovni paralelizam – izvire iz same prirode podataka svojstvenih problemu te iz struktura podataka koje se rabe u rješavanju problema

FER university of zagreb  
faculty of electrical engineering and computing

Četiri razine (zrnatosti) raspoloživog funkcijskog paralelizma:

- paralelizam na razini instrukcija (ILP) – fino zrnati
- paralelizam na razini programskih petlji – srednje zrnati
- paralelizam na razini procedura, funkcija i potprograma – srednje zrnati
- paralelizam na razini programa – grubo zrnati

FER university of zagreb  
faculty of electrical engineering and computing

Primjer - paralelizam na razini programskih petlji – srednje zrnati

Promotrimo programsku petlju kojom se zbrajaju dva jednodimenzionalna polja svako od 1000 elemenata:

```
for (i = 1; i <= 1000; i = i + 1)
    x[i] = x[i] + y[i];
```

Svaka od 1000 iteracija može se u izvesti istodobno tako da se izvođenje preklapa

FER university of zagreb faculty of electrical engineering and computing

Razine iskorištenog paralelizma

- raspoloživi funkcijski paralelizam mogu iskoristiti arhitektura procesora i operacijski sustav procesora da i se povećala brzina obrade

Razine iskorištenog paralelizma:

- na razini instrukcija
- na razini dretvi (engl. thread)
- na razini procesa
- na korisničkoj razini

FER university of zagreb faculty of electrical engineering and computing

- paralelizam na razini instrukcija – isključivo se koristi u arhitekturi procesora
- paralelizam na razini dretvi i procesa – arhitektura i operacijski sustav
- paralelizam na korisničkoj razini (višezadačni rad – *multitasking*, višeprogramski rad – *multiprogramming*, obrada dodjeljivanjem vremena – *time-sharing*) – operacijski sustav

FER university of zagreb faculty of electrical engineering and computing

Podatkovni paralelizam može se iskoristiti na dva načina:

- izravno uporabom arhitektura procesora koje podržavaju paralelne operacije na podatkovnim elementima (SIMD, procesori upravljani tokom podataka (engl. data-flow))
- pretvorbom raspoloživog podatkovnog paralelizma u funkcijski tako da se uporabom naredbi iz viših programskih jezika označe paralelno izvodljive operacije na podatkovnim elementima

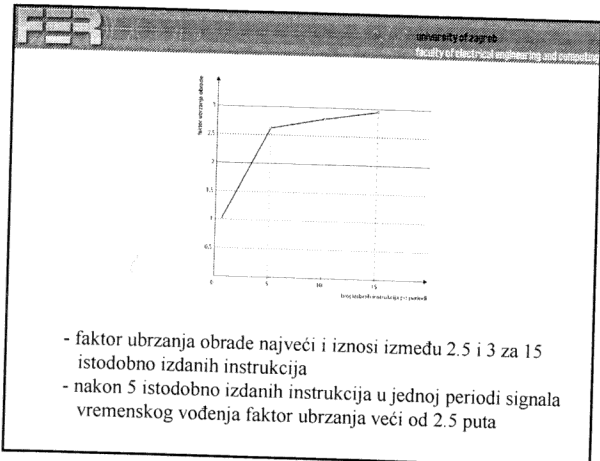
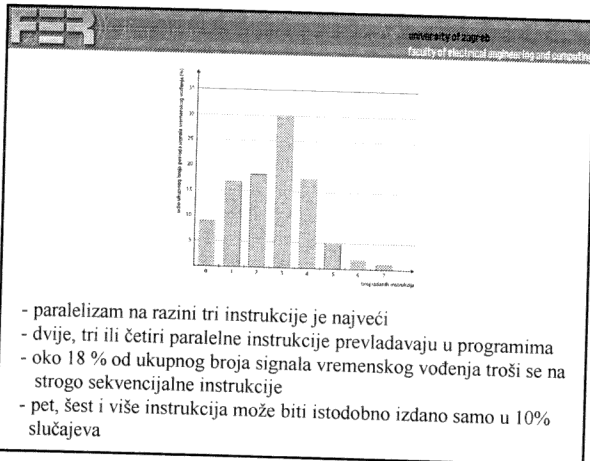
FER university of zagreb faculty of electrical engineering and computing

Paralelizam na razini instrukcija

RISC / CISC procesori

Type of instruction	Number of instructions per cycle (IPC)
RISC	~15
CISC	~18
CISC (with a sub-category)	~22

Podaci na slici dobiveni su kao srednja vrijednost brojnih ispitnih programa čije je izvođenje simulirano na tzv. virtualnim agresivno oblikovanim strojevima – paralelno izvođenje pod idealnim uvjetima u kojima nema ograničenja raspoloživih resursa i uz uvjet savršenog predviđanja grananja

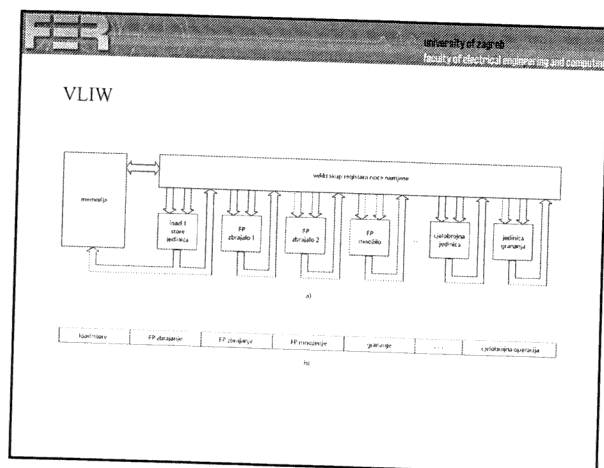


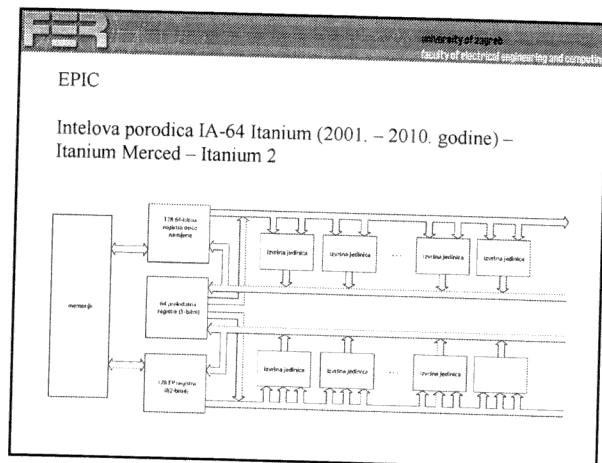
Paralelizam na razini instrukcija je *implicitni paralelizam* koji prevodilac ili sklopovlje treba otkriti u slijednom programu

- statički tijekom prevođenja programa ili
- dinamički tijekom izvođenja programa

Djelotvornije iskorištenje paralelizma može postići uporabom *eksplicitnog paralelizma* u samoj instrukciji ili nizu instrukcija

Pristup: VLIW – Very Long Instruction Word (kombinacija koncepta horizontalnog mikroprogramiranja, superskalarne obrade, višestrukosti funkcijskih jedinica)  
EPIC – Explicitly Parallel Instruction Computing)





FER university of zagreb faculty of electrical engineering and computing

Instrukcijska riječ za procesor IA-64 arhitekture duljine je 128 bita i sadržava tri instrukcije tipa RISC koje tvore tzv. "svežanj" (engl. *bundle*). Svaka od instrukcija u svežnju duljine je 41 bita pa u svežnju tri instrukcije zauzimaju 123 bita, dok se preostalih 5 bitova koriste kao predložak koji sadržava informaciju o njihovom raspoređivanju (engl. *scheduling*). Točnije, 41-bitna "instrukcija" zapravo nije prava instrukcija (u Intelu je nazivaju slog (engl. *syllable*)) jer se ona mora kombinirati s 5-bitnim predloškom. Predložak određuje tipove instrukcija u svežnju i određuje koje će se instrukcije izvesti paralelno.

- FER university of zagreb faculty of electrical engineering and computing
- Tipovi instrukcija u svežnju mogu biti sljedeći:
- A tip – odnosi se na cjelobrojnu jedinicu (I-unit) i određuje operacije cjelobrojne ALU jedinice (npr. zbroji, oduzmi, logičko I, logičko ILI, usporedi),
  - I tip – odnosi se na cjelobrojnu jedinicu, ali određuje operacije koje nisu cjelobrojne aritmetičko-logičke, već se odnose na tzv. multimedijske operacije kao što multimedijski posmak, ispitivanje bitova, premještanje,
  - M tip – odnosi se na jedinicu za pristup memoriji (M-unit) i određuje *load* i *store* instrukcije za cjelobrojne i FP (*floating-point*) registre,
  - F tip – odnosi se na jedinicu za operacije brojevima s pomičnim zarezom (F-unit) i određuje operacije brojevima s pomičnim zarezom,

FER university of zagreb faculty of electrical engineering and computing

B tip odnosi se na jedinicu grananja (B-unit) i određuje uvjetno, bezuvjetno grananje te pozive potprograma,

L + X tip – izvršavaju se u jedinici grananja ili u cjelobrojnoj jedinici i određuju mješovite i specijalne instrukcije (npr. *nop*, *stop*).

FER university of zagreb  
faculty of electrical engineering and computing

Specifikacija paralelnosti u izvođenju instrukcija može se proširiti i na dijelove nekoliko svežanja. Zahvaljujući 5-bitnom predlošku pojednostavljen je postupak dekodiranja instrukcija i izdavanja instrukcija.

Mogući su sljedeći načini kombiniranja triju instrukcija u svežnju (ovisno o binarnom uzorku u 5-bitnom predlošku):

$i_1 \parallel i_2 \parallel i_3$  – sve se tri instrukcije izvode paralelno,  
 $i_1 \& i_2 \parallel i_3$  – prvo se izvodi instrukcija  $i_1$ , a zatim se instrukcije  $i_2$  i  $i_3$  izvode paralelno,  
 $i_1 \parallel i_2 \& i_3$  – prvo se instrukcije  $i_1$  i  $i_2$  izvode paralelno, a onda instrukcija  $i_3$ ,  
 $i_1 \& i_2 \& i_3$  – sve se tri instrukcije u svežnju izvode slijedno.

FER university of zagreb  
faculty of electrical engineering and computing

Jedna od najvažnijih metoda za nalaženje i djelotvorno iskorištenje paralelizma na razini instrukcija u protočnoj i superskalarnoj arhitekturi procesora je *metoda špekulacije* ili *nagađanja* (engl. *speculation*). Pojednostavljeno, špekulacija je pristup koji dopušta prevodiocu ili procesoru nagađanje o svojstvima i/ili ishodu neke instrukcije, i to tako da omogućiti započinjanje ostalih instrukcija koje mogu ovisiti o instrukciji na koju se špekulacija odnosi.

Primjer: nagađanje da se ne radi o RAW hazardu  
 nagađanje o ishodu instrukcije uvjetnog grananja

Špekulacija se izvodi uporabom prevodioca ili sklopovljem procesora a u novije vrijeme kombiniraju se obje izvedbe

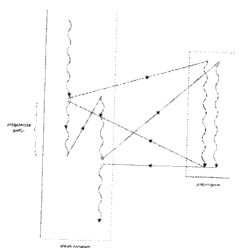
FER university of zagreb  
faculty of electrical engineering and computing

Paralelizam na razini dretvi i procesa

Dretva (engl. thread) može se objasniti pomoću jednostavnog modela računala J. Backusa – računalo predočeno memorijskom jedinicom, procesorom i spojnim putom između njih

- u memorijskoj jedinici pohranjeni su program i podaci
- pretpostavimo da procesor može pristupiti cijelom adresnom prostoru koji je određen kapacitetom memorije
- program se izvodi tako da se u programsko brojilo upiše adresa prve instrukcije i procesor započinje izvođenje programa
- izvođenje programa možemo promatrati kao da procesor provlači kroz program (slijed instrukcija) ili "nit" ili instrukcijsku dretvu

FER university of zagreb  
faculty of electrical engineering and computing



Ilustracija dretve

**FER** university of zagreb  
faculty of electrical engineering and computing

Razlika između dretve i procesa

- program koji se izvršava organiziran je u jedan ili veći broj procesa

Proces se sastoji barem od jedne instrukcijske dretve te predstavlja izvršljivi programski odsječak kojem je dodijeljen

- programsko brojilo,
- adresni prostor,
- registri,
- varijable

Svaki proces ima svoj virtualni procesor

**FER** university of zagreb  
faculty of electrical engineering and computing

- ako se radi o jednoprocorskom sustavu tada se stvarni procesor dijeli između više procesa

**FER** university of zagreb  
faculty of electrical engineering and computing

Računalnom procesu, budući da je riječ o procesu u vremenu, pridruženi i neki vremenski atributi:

- trenutak početka izvođenja,
- trenuci zaustavljanja izvođenja,
- trenutak završetka izvođenja,
- trajanje izvođenja

Iako je proces samostalna i nezavisna jedinka vrlo često je u interakciji s ostalim procesima

- s obzirom na interakciju s ostalim procesima i njihovu međuzavisnost možemo identificirati tri stanja u kojima se proces može nalaziti

**FER** university of zagreb  
faculty of electrical engineering and computing

Stanja u kojima se proces može nalaziti:

- aktivno stanje procesa (engl. *running*) – to je proces kojem je dodijeljen procesor i koji se upravo izvodi,
- pripravno stanje procesa (engl. *ready*) – proces je spreman za izvođenje i čeka na dodjelu procesora,
- blokirani proces (engl. *blocked*) – proces čeka na ispunjenje nekog uvjeta za njegovo daljnje napredovanje (npr. proces čeka na neki vanjski događaj, čeka na istek nekog vremenskog intervala).



FER university of zagreb faculty of electrical engineering and computing

Operacijski sustav održava *tablicu procesa* (engl. *process table*) u kojoj svaki element tablice odgovara jednom procesu i sadržava informaciju potrebnu za rukovanje procesom (engl. *process management*), informaciju potrebnu za rukovanje memorijom (engl. *memory management*) te informaciju potrebnu za rukovanje datotekama (engl. *file management*).

Sva se informacija koja se odnosi na određeni proces mora pohraniti u tablicu procesa i to svaki put kada proces iz aktivnog stanja prelazi u stanje pripravnosti ili blokiranosti

FER university of zagreb faculty of electrical engineering and computing

Višeprogramski način rada – više procesa dijeli u vremenu jedan procesor

- ako računarski sustav ima više procesora – onda je moguće svakom procesu, umjesto virtualnog procesora, dodijeliti procesor i tada se međusobno nezavisni procesi mogu izvoditi ISTODOBNO
- proces u najjednostavnijem obliku ima samo jednu dretvu i jedno programsko brojilo
- suvremeni višenamjenski računalni sustavi podržavaju veći broj dretvi u jednom procesu (takve se dretve često nazivaju i "laki" procesi; engl. *lightweight process*)

FER university of zagreb faculty of electrical engineering and computing

Razlika između procesa i dretve:

Diagram illustrating the difference between processes and threads. It shows a computer system (računarski sustav) containing three separate processes: proces A, proces B, and proces C. Each process has its own program counter (PC) and program number (programsko brojilo).

- pretpostavimo da imamo u rač. sustavu tri procesa A, B i C
- svaki od njih ima jednu dretvu, programsko brojilo te svaki proces djeluje u tri različita adresna prostora

FER university of zagreb faculty of electrical engineering and computing

Jedan proces u računarskom sustavu koji ima tri dretve:

Diagram illustrating a single process in a computer system (računarski sustav) that has three threads (dretve). Each thread has its own program counter (PC) but they share a common program number (programsko brojilo).

- dretve imaju svaka svoje programsko brojilo ali dijele zajednički adresni prostor koji je dodijeljen tom procesu

FER university of zagreb  
faculty of electrical engineering and computing

Prisutnost više dretvi u jednom procesu koje dijele isti adresni prostor zahtijeva, pored tablice procesa, još i tablicu dretve (opisnik dretve) s informacijom o identifikatoru dretve, pripadnost dretve određenom procesu, stanju i prioritetu dretve, početnoj adresi dretvenog adresnog procesa, sadržaju programskog brojala, te sadržaju registara – kontekst dretve

Dretva može biti jednako kao i proces u *aktivnom stanju*, *stanju pripravnosti* i *stanju blokiranosti*

Stanje procesa s više dretvi ne može se jednoznačno opisati (npr. jedna dretva procesa može biti u aktivnom stanju, dvije u stanju pripravnosti a jedna u stanju blokiranosti). U kojem je stanju proces?

FER university of zagreb  
faculty of electrical engineering and computing

Proces s više dretvi može se izvoditi u jednoprocesorskom računarskom sustavu tako da se procesor dodjeljuje naizmjenice pojedinim dretvama.

U tom se slučaju prenosi upravljanje s dretve na dretvu uz uzastopno pohranjivanje i obnavljanje konteksta dretvi.

U slučaju izvođenja više dretvi u jednoprocesorskom sustavu teško se može očekivati ubrzanje odvijanja procesa. Iznimno: ako procesor izvodi neku drugu dretvu za vrijeme dok druga zbog nekog razloga mora čekati

FER university of zagreb  
faculty of electrical engineering and computing

Potpuno iskorištenje paralelizma na razini dretvi postiže se u višeprocorskom sustavu u kojem je svaki procesor zadužen za jednu od više raspoloživih dretvi u procesu – višedretveni rad (engl. *multithreading*).

Opaska: izraz “višedretveni rad” (engl. *multithreading*) upotrebljava se za izvođenje više dretvi na jednom procesoru (analogija s višeprogramskim radom (engl. *multiprogramming*))

Izvođenje većeg broja dretvi u višeprocorskom sustavu često se naziva i hiperdretveni rad (engl. *hyperthreading*)

Pozor: Intel u Netburst mikroarhitekturi koristi izraz *hiperdretvenost* u kontekstu višedretvenosti

FER university of zagreb  
faculty of electrical engineering and computing

Paralelizam na razini dretvi (engl. *thread-level parallelism*; TLP) bitna je alternativa paralelizmu na razini instrukcija (ILP) jer se može ostvariti jednostavnije i jeftinije od ILP-a.

S druge strane, u mnogim je primjenama paralelizam na razini dretvi svojstven i prirodan samom problemu.

Potrebno je istaknuti da se ove dvije razine paralelizma u implementaciji arhitekture procesora međusobno ne isključuju, štoviše, one se podupiru i doprinose većoj performansi procesora

FER university of zagreb  
faculty of electrical engineering and computing

### Paralelizam na korisničkoj razini

- *višeprogramskog rada* u kojem se procesor djelotvorno koristi tako da se oblikuju pripralni procesi koji pripadaju različitim korisničkim programima,
- *višezadačni način rada* (engl. *multitasking*) sličan je višeprogramskom načinu rada te ga stoga neki autori poistovjećuju s njim, međutim, postoji jedna razlika – istodobno aktivni procesi u višezadačnom načinu rada pripadaju *istom* korisniku,
- pseudoparalelizam na korisničkoj razini postiže se *radom u vremenskoj podjeli* (engl. *time-sharing*) kojim se nudi istodobna usluga većem broju korisnika koji pristupaju računarskom sustavu terminalima

FER university of zagreb  
faculty of electrical engineering and computing

### Podatkovni paralelizam

- namjenskom arhitekturom koja dopušta paralelne operacije na podatkovnim elementima, npr. SIMD (*Single Instruction Stream Multiple Data Stream*) arhitektura ili arhitektura upravljana tokom podataka (engl. *dataflow*)
- pretvorba podatkovnog paralelizma u funkcijski paralelizam tako da se na slijedan način izraze paralelno izvodljive operacije na podatkovnim elementima uporabom jezičnog konstrukta za specifikaciju programskih petlji

FER university of zagreb  
faculty of electrical engineering and computing

SIMD računarski sustavi iskorištavaju podatkovni paralelizam djelovanjem na vektorima ili dvo- i višedimenzionalnim poljima podataka.

Na primjer, jednom SIMD instrukcijom mogu se obaviti zbrajanja 64 para raspoloživih operanada u 64 aritmetičko-logičke jedinice, i to u jednoj periodi signala vremenskog vođenja.

Osnovne zamisli SIMD-a nalazimo i u arhitekturi suvremenih procesora:

- SIMD instrukcije koje su namijenjene poboljšanju performansi za multimedijske aplikacije – istodobno izvođenje jedne operacije na većem broju operanada u većem broju ALU

FER university of zagreb  
faculty of electrical engineering and computing

SIMD instrukcije rekonfiguriraju jednu ALU koja se koristi dugim riječima u više manjih ALU koje djeluju paralelno na kraćim operandima. Tako se, na primjer, 64-bitna ALU pretvara u dvije 32-bitne ALU ili u četiri 16-bitne ALU ili u osam 8-bitnih ALU koje djeluju istodobno na operandima odgovarajuće duljine.

Npr. podaci duljine 8 bita koriste za definiranje vrijednosti primarnih boja (R, G, B) slikovnih elemenata ili se 16-bitni podaci koriste za prikaz vrijednosti zvučnog uzorka.

FER university of zagreb faculty of electrical engineering and computing

Primjer:

Godine 1997. tvrtka Intel je proširila skup instrukcija za procesore Pentium i PentiumPro s 57 SIMD instrukcija koje su nazvane MMX (*Multi Media Extensions*).

MMX instrukcije djeluju nad skupom 64-bitnih registara koji su nazvani MMX registri. Zapravo, MMX instrukcije koriste osam registara za podatke s pomičnim zarezom (*floating-point registre*) koji se koriste kao MMX 64-bitni registri.

Svaki od 64-bitna MMX registra sadržava ili jedan 64-bitni cijeli broj ili vektor koji se sastoji od 2, 4 ili 8 cjelobrojnih podataka (sukcesivno kraćih operandi, npr. osam 8-bitnih operandi).

Različite aritmetičke i logičke MMX instrukcije te MMX instrukcije za uspoređivanje podataka i njihovo preuređenje djeluju na 64-bitnim i 32-bitnim operandima te 8-, 4- i 2-komponentnim nezavisnim vektorima koji se mogu "upakirati" u 64-bitnu riječ.

FER university of zagreb faculty of electrical engineering and computing

Izvođenje MMX instrukcije paralelnog zbrajanja dvaju vektora

FER university of zagreb faculty of electrical engineering and computing

Procesoru Pentium III (1998.) pridodano je još 70 SIMD instrukcija, nazvanih SSE (*Streaming SIMD Extensions*) koje djeluju na osam dodatnih registara duljine 128 bita i podržavaju operacije jednostruke točnosti podacima s pomičnim zarezom (engl. *single precision floating-point operation*) tako da se istodobno mogu izvršiti četiri operacije s pomičnim zarezom na četiri 32-bitna operanda.

Proširenje skupa SIMD instrukcija nastavilo se pa su 2001. skupu instrukcija dodane nove 144 instrukcije SSE2 koje podržavaju SIMD operacije na 64-bitnim podacima s pomičnim zarezom.

Godine 2007. tvrtka AMD uvodi SSE5 sa 170 novih instrukcija, a 2008. Intel uvodi skup SIMD instrukcija AVE – *Advanced Vector Extension* kojim se SSE registri proširuju s 128 na 256 bita duljine te se skup SIMD instrukcija povećava na više od tri stotine!

FER university of zagreb faculty of electrical engineering and computing

### 3. Višeprocorski SIMD sustavi

Prema Flynnovoj klasifikaciji arhitekture tri se kategorije arhitekture odnose na paralelne računarske sustave: MISD, SIMD i MIMD

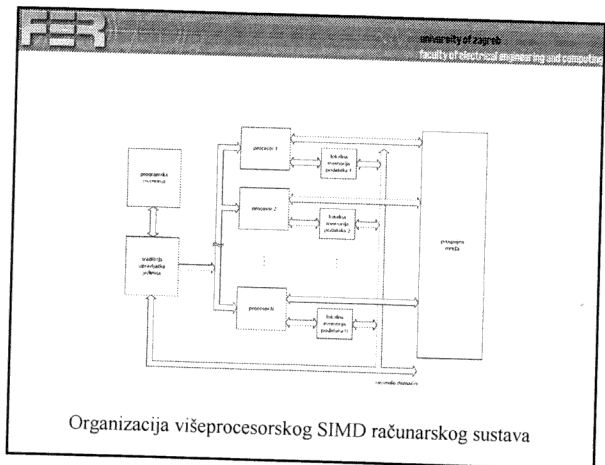
- višeprocorski SIMD,
- višeprocorski MIMD – multiprocorski sustavi

FER university of zagreb  
faculty of electrical engineering and computing

Osnovna značajka SIMD arhitekture je istodobno izvođenje iste instrukcije od strane većeg broja procesora koji djeluju na različitim, višestrukim tokovima podataka.

Višeprocorski SIMD računarski sustavi namijenjeni su rješavanju složenih problema s visokim stupnjem inherentnog paralelizma, prvenstveno sadržanog u podacima.

Višeprocorski SIMD sustavi iskorištavaju *podatkovni paralelizam*.



FER university of zagreb  
faculty of electrical engineering and computing

SIMD arhitektura u nekoliko je posljednjih godina dobila na značenju, posebno u multimedijskoj primjeni (naročito u računalnoj grafici) kao jedan od djelotvornih pristupa oblikovanja trodimenzionalnih virtualnih okruženja u stvarnom vremenu.

FER university of zagreb  
faculty of electrical engineering and computing

#### 4. Vektorski procesori

Jedan od najdjelotvornijih načina iskorištavanja podatkovnog paralelizma postiže se u računarskim sustavima koji se svrstavaju u SIMD kategoriju arhitekture i temelje se na *vektorskom procesoru* (engl. *vector processor*)

Vektorski procesor obavlja aritmetičke i logičke operacije na operandima koji su vektori.

FER university of zagreb faculty of electrical engineering and computing

Primjer:  
Razmotrimo računanje sume dvaju 64-dimenzionalnih vektora  $\mathbf{x}$  i  $\mathbf{y}$ .  
Rezultat je vektor  $\mathbf{w}$ :  
$$\mathbf{w} = \mathbf{x} + \mathbf{y}.$$
  
U "običnom" jednoprocorskom sustavu ta bi se operacija izvela na temelju programskog odsječka:  

```
for i = 1 to 64
    w(i) := x(i) + y(i);
```

  
U vektorskom procesoru gornja bi se operacija izvela samo *jednom vektorskom instrukcijom*, odnosno instrukcijom tipa *vektor-vektor* kojoj su operandi dva 64-dimenzionalna vektora, a rezultat, koji se dobiva u vektorskoj aritmetičko-logičkoj jedinici (vektorska ALU), također je 64-dimenzionalni vektor.

FER university of zagreb faculty of electrical engineering and computing

Vektorska ALU može istodobno zbrojiti sve odgovarajuće komponente obaju vektora. Svaki od vektora, koji predstavlja operand u vektorskoj instrukciji, smješten je u vektorski registar, npr.  $V_i$ , odnosno  $V_j$ , a rezultat se smješta u vektorski registar  $V_k$ .

Vektorska instrukcija specificira veliku količinu posla i jednakovrijedna je, u potonjem slučaju, cijeloj programskoj petlji.

-Uporaba vektorske ALU u kojoj se istodobno izvode operacije nad svim komponentama vektora u vektorskim je procesorima ipak rijetka.

FER university of zagreb faculty of electrical engineering and computing

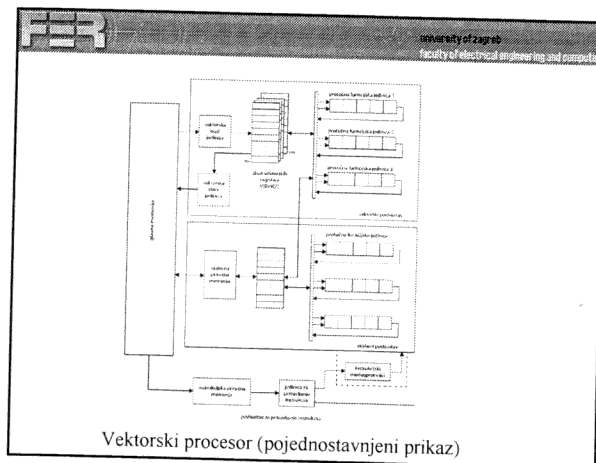
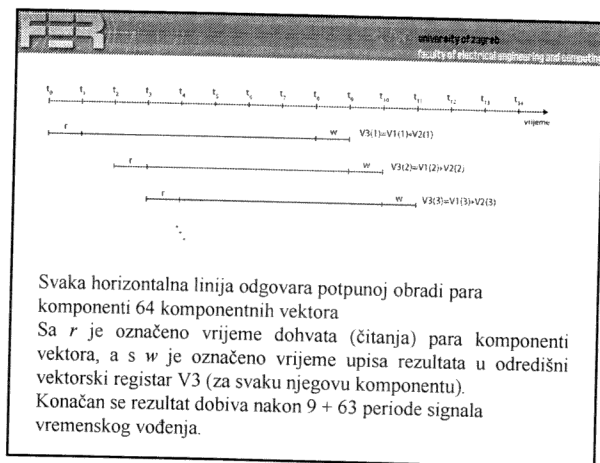
Umjesto, na primjer, vektorskog množila brojevima s pomičnim zarezom (koji bi istodobno generirao sve produkte), u praksi se koristi protočno množilo s relativno velikim brojem protočnih segmenata te se svaki parcijalni rezultat (umnožak dviju komponentata) dobiva u svakoj periodi signala vremenskog vođenja čije je trajanje određeno vremenom obrade u jednom protočnom segmentu.

Razlozi takva rješenja nisu tehnološka ograničenja, već ekonomski faktor.

FER university of zagreb faculty of electrical engineering and computing

Primjer:  
Razmotrimo izvođenje vektorske instrukcije množenja dvaju 64-komponentnih vektora u 7-segmentnom protočnom množilu brojevima s pomičnim zarezom:

MULTV V1, V2, V3 ; V3 = V1 \* V2



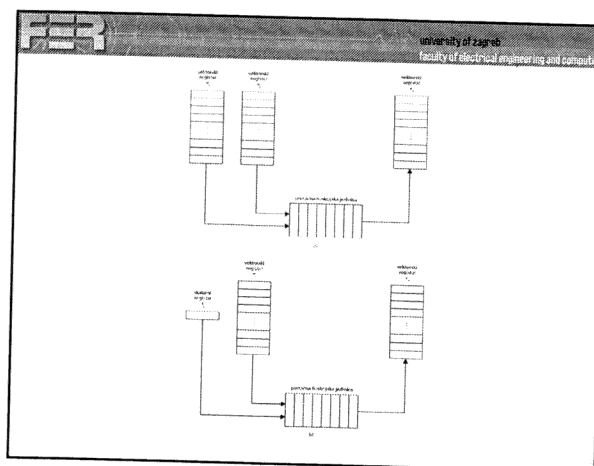
FER university of zagreb faculty of electrical engineering and computing

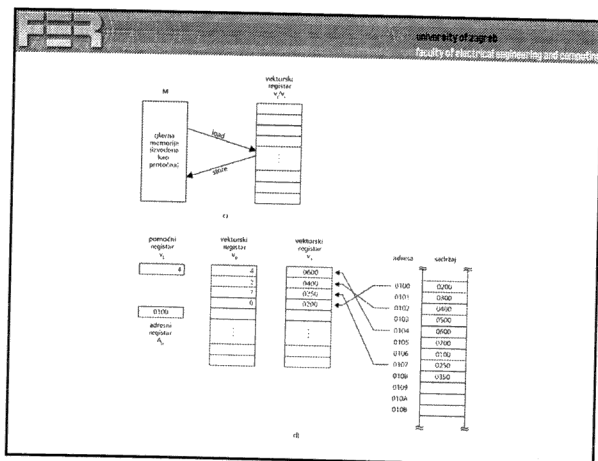
Vektorski procesor podržava šest tipova vektorskih instrukcija:

- instrukcije vektor-vektor
 
$$f_1: V_i \rightarrow V_k$$

$$f_2: V_i \times V_j \rightarrow V_k$$
 gdje su  $V_i$  i  $V_j$  izvorišni vektorski registri, a  $V_k$  odredišni vektorski registar;
- instrukcije vektor-skalar
 
$$f_3: s_j \times V_i \rightarrow V_k$$
 gdje je  $s_j$  skalarni registar;
- instrukcija vektor-memorija
 
$$f_4: M \rightarrow V_i \text{ za operaciju load}$$

$$f_5: V_i \rightarrow M \text{ za operaciju store,}$$
 gdje je  $M$  glavna memorija;





FER university of zagreb faculty of electrical engineering and computing

iv) instrukcija redukcije vektora. Formalno se taj tip instrukcije može opisati kao:

$$f_6: V_i \rightarrow s_j$$

$$f_7: V_i \times V_j \rightarrow s_j$$

v) instrukcije okupljanja (engl. *gather*) ili raspršivanja (engl. *scatter*)

$$f_8: M \rightarrow V_i \times V_0 \quad \text{okupljanje}$$

$$f_9: V_i \times V_0 \rightarrow M \quad \text{raspršivanje}$$

- Operacijom okupljanja iz memorije se dohvaćaju elementi različiti od nule, i to tako da vektorski registar  $V_0$  sadržava indekse (kazaljke) na podatke u memoriji, a vektorski registar  $V_1$  sadržava podatke koji se iz memorije dohvaćaju i oblikuju tzv. rijetko popunjeni vektor.

FER university of zagreb faculty of electrical engineering and computing

Operacija raspršivanja obrnuta je operacija u odnosu na operaciju okupljanja. Pomoću nje se u memoriju pohranjuje rijetko popunjeni vektor;

vi) instrukcije maskiranja – taj tip instrukcije upotrebljava vektor maskiranja  $V_m$  (engl. *mask vector*) pomoću kojeg sažima ili proširuje izvorni vektor. Formalno maskiranje može se opisati kao preslikavanje:

$$f_{10}: V_0 \times V_m \rightarrow V_1$$

Npr. Ispituje se jesu li komponente vektora  $V_0$  različite od nula, i to samo oni elementi vektora koji odgovaraju jedinicama u maskinom registru  $V_m$ . U vektorskom registru  $V_1$  pohranjuju se indeksi koji odgovaraju elementima u ispitnom registru  $V_0$  i koji su različiti od 0.

FER university of zagreb faculty of electrical engineering and computing

Paralelizam u izvođenju vektorskih instrukcija postiže se uporabom protočnih funkcijskih jedinica s velikim brojem protočnih segmenata ili većim brojem istorodnih funkcijskih jedinica koje djeluju istodobno, ili kombinacijom uporabe većeg broja istorodnih funkcijskih jedinica koje su izvedene kao protočne s velikim brojem protočnih segmenata.

Svaka od paralelnih protočnih jedinica naziva se *traka* (engl. *lane*).



Performansa vektorskog procesora povećava se uporabom metode koja se naziva *višetračna vektorska obrada* (engl. *multilane vector processing*)

Kako se vektorska obrada odražava na performansu procesora? Performansa takva procesora ovisi o mnogo faktora, na primjer o:

- i) razini "vektORIZACIJE" programa – koliki dio od ukupne obrade može biti izveden vektorskim operacijama,
- ii) prosječnoj duljini vektora, odnosno prosječnom broju komponenti vektora,
- iii) razini ulančavanja vektora – mogućnosti izdavanja sljedeće vektorske instrukcije neposredno nakon što prethodna vektorska instrukcija generira prvu komponentu vektora rezultata,
- iv) razini preklapanja vektorskih, skalarnih i operacija pristupa memoriji.

Performansa vektorskog procesora može se prikazati njemu prilagođenim Amdahlovim zakonom, gdje je faktor ubrzanja obrade  $s$  u odnosu na skalarni procesor jednak:

$$s = \frac{1}{(1-f) + f/k}$$

gdje je  $f$  dio vektoriziranog koda, a  $k$  brzina vektorske jedinice u odnosu na skalarnu jedinicu.  $k$  je funkcija duljine vektora i tipa operacije.

s- ubrzanje

Ilustracija Amdahlovog zakona za vektorske procesore

FER university of zagreb  
faculty of electrical engineering and computing

Primjer:

Ilustrirajmo izvođenje operacije  $y = sx + y$ , gdje su  $y$  i  $x$  64 komponentni vektori pri čemu je svaka komponenta predložena u notaciji broja s pomičnim zarezom dvostruke točnosti (duljine 64 bita), a  $s$  je skalar također dvostruke točnosti, na skalarnom procesoru i vektorskom procesoru.

Pretpostavit ćemo da se  $y$ ,  $x$  i  $s$  nalaze na početku izvođenja pohranjeni u memoriji tako da je početna adresa za  $x$  jednaka  $\$s0$ , za  $y$   $\$s1$ , a za skalar  $s$   $\$sp$ .

FER university of zagreb  
faculty of electrical engineering and computing

Programski odsječak za skalarni procesor izgleda ovako:

```
ld f0, $sp ; dohvati skalar s i pohrani ga u floating-point
; registar f0
addi r4, $s0, #512 ; gornja adresa lokacije na kojoj
; se nalazi vektor x
opet: ld f2, 0($s0) ; dohvati x(i)
mul f2, f0, f2 ; s x(i)
ld f4, 0($s1) ; dohvati y(i)
add f4, f2, f4 ; s x(i) + y(i)
st f4, 0($s1) ; pohrani rezultat na y(i)
addi $s0, $s0, #8 ; inkrementiraj indeks za x
addi $s1, $s1, #8 ; inkrementiraj indeks za y
sub $t0, r4, $s0 ; izračunaj granicu
bne $t0, $zero, opet ; provjeri je li sve obavljeno
```

FER university of zagreb  
faculty of electrical engineering and computing

(Opaska: adresna znatost memorije je bajtna, zato je gornja granica 512, tj.  $64 \times 8$ , gdje je 64 broj komponenti vektora, a svaka je njegova komponenta duljine 8 bajtova.)

FER university of zagreb  
faculty of electrical engineering and computing

Programski odsječak za vektorski procesor izgleda ovako:

```
ld f0, $sp ; dohvati skalar s i pohrani ga u
; floating-point registar f0
ldv v1, 0($s0) ; dohvati vektor x i pohrani ga u
; vektorski registar v1
mulvs v2, v1, f0 ; množenje vektora x sa
; skalarom s
ld v3, 0($s1) ; dohvati vektor y i pohrani ga u
; vektorski registar v3
addv v4, v2, v3 ; pribroji y produktu sx
stv v4, 0($s1) ; pohrani rezultat
```

FER university of zagreb faculty of electrical engineering and computing

Usporedba gornjih programskih odsječaka:

- vektorski procesor značajno reducira promet instrukcija – on zahtijeva samo šest instrukcija, dok se kod skalarnog procesora zahtijeva skoro 600 instrukcija (programska petlja).
- druga zanimljiva razlika je u učestalosti hazarda – za skalarni procesor svaka *add* instrukcija mora čekati na *mul* instrukciju te svaka *st* instrukcija mora čekati na *add* instrukciju.
- za vektorski će procesor svaka vektorska instrukcija biti u zastoju samo za prvi element svakog od vektora, tako da će ostali elementi glatko protjecati kroz protočnu strukturu.

FER university of zagreb faculty of electrical engineering and computing

Zamisao vektorske obrade stara je više od pedeset godina. Prva vektorska superračunala pojavljuju se početkom sedamdesetih godina prošlog stoljeća (CDC STAR-100, TI ASC, 1972.), Cray-1 (1976.) da bi devedesetih godina na tržištu bio veći broj višeprocorskih vektorskih superračunala (Cray Y-MP 816, Hitachi S-820, Fujitsu VP-2000, NEC Sx-8).

Danas vektorska obrada nije rezervirana samo za superračunala, već se pojavljuje i u procesorima koji se rabe u multimedijским aplikacijama (procesori s instrukcijskim podskupovima MMX, SSE, AltiVec) i procesorima namijenjenim igračim konzolama.

FER university of zagreb faculty of electrical engineering and computing

Razvoj tehnologije visokog stupnja integracije (VLSI) omogućio je izvedbe vektorskih mikroprocesora – vektorskih procesora s jednom (npr. SPERT-II procesor) ili više jezgri ostvarenih na jednom čipu.

- Cell dvojezgreni procesor (razvijen u suradnji tvrtki IBM, Toshiba i Sony, 2006.) koji se sastoji od dvije jezgre, odnosno od dva procesora PowerPC (naziva se PPE – *Power Processing Element*) koji djeluju pod "konvencionalnim" operacijskim sustavom te se koriste za distribuciju poslova osam vektorskih jedinica SPE (*Synergistic Processing Element*) koje imaju vlastiti SIMD skup instrukcija. Svaka od SPE, jedinica, koja je u stvari 128 bitni RISC procesor SIMD organizacije, ima svoju vlastitu memoriju izvedene statičkim RAM-om umjesto priručne memorije. PPE i SPE povezani su pomoću visokopropusne (> 300 GB/s) prstenaste sabirnice podataka nazvane EIB - *Element Interconnect Bus*.

FER university of zagreb faculty of electrical engineering and computing

### 5. Multiprocorsorski sustavi – višeprocorsorski MIMD sustavi

MIMD arhitektura obilježena je višestrukim instrukcijskim tokom i višestrukim tokom podataka. Svaki od procesora pribavlja i izvršava svoje vlastite instrukcije na svojim podacima. Višeprocorsorski MIMD sustavi ili *multiprocorsorski sustavi* iskorištavaju *paralelizam na razini procesa i dretvi*.

- U multiprocorsorskom sustavu svaki procesor može izvršavati njemu dodijeljen *proces*.
- Svaki od procesa može imati više dretvi tako da se izvođenje jednog procesa s većim brojem dretvi može povjeriti većem broju procesora. Višedretvena arhitektura temeljena na MIMD-u, u načelu, dopušta istodobno izvođenje većeg broja procesa s izdvojenim adresnim prostorima i izvođenje više dretvi koji dijele adresni prostor.

FER university of zagreb faculty of electrical engineering and computing

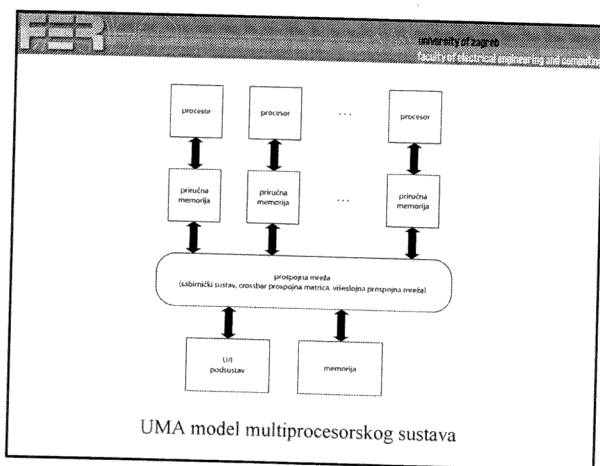
Osnovna značajka multiprocesorskog sustava jest veći broj procesora približno jednakih (vrlo često) identičnih obilježja koji na izvjestan način dijele zajednički memorijski prostor.

Ovisno o broju procesora i načinu organizacije memorijskog sustava multiprocesorski se sustavi mogu klasificirati u sljedeće skupine:

- sustavi s uniformnim pristupom memoriji UMA (engl. *Uniform Memory Access*),
- sustavi s neuniformnim pristupom memoriji NUMA (engl. *Nonuniform Memory Access*),
- sustavi samo s priručnom memorijom COMA (engl. *Cache-Only Memory Architecture*).

FER university of zagreb faculty of electrical engineering and computing

U UMA modelu multiprocesorskog sustava svi procesori dijele zajedničku fizičku memoriju i svi imaju jednako vrijeme pristupa svakoj od riječi u memoriji (uniformni, odnosno ujednačeni pristup memoriji). Svaki od procesora može imati i svoju vlastitu priručnu memoriju organiziranu u jednu ili više razina. Na sličan način kao što dijele memoriju, procesori dijele resurse U/I podsustava.



FER university of zagreb faculty of electrical engineering and computing

UMA model se još naziva i *multiprocesorski sustav sa središnjom dijeljenom memorijom* (engl. *centralized shared-memory*) ili *simetrični multiprocesorski sustav s dijeljenom memorijom SMP* (engl. *symmetric shared-memory multiprocessor*) – zato što memorija ima isti ("simetrični") odnos spram svih procesora.

- Komunikacija procesora sa zajedničkom memorijom i U/I podsustavom ostvaruje se prosvojnom mrežom koja ovisno o zahtijevanoj propusnosti može biti ostvarena sabirničkim sustavom, crossbar prosvojnom matricom ili višerazinskom prosvojnom mrežom (npr. Omega).

FER university of zagreb faculty of electrical engineering and computing

Značajke:

- UMA model pogodan je za relativno mali broj procesora (manji od 100) jer je za taj broj procesora još uvijek moguće ostvariti dijeljenje i uniformni pristup memoriji. UMA model još se naziva i *čvrsto povezan multiprocesorski sustav* (engl. *tightly coupled*) zbog visokog stupnja dijeljenja zajedničkih resursa (memorije i U/I podsustava).
- UMA model multiprocesorskog sustava najčešće se koristi zato što je podesan za primjene opće namjene i izvedbu obrade dodjeljivanjem vremena u višekorisničkim okruženjima.

FER university of zagreb faculty of electrical engineering and computing

Multiprocesorski sustavi s neuniformnim pristupom memoriji NUMA nazivaju se još i *sustavi s porazdijeljenom memorijom* (engl. *distributed-memory multiprocessor*) imaju umjesto centralizirane memorije, memoriju porazdijeljenu procesorima.

FER university of zagreb faculty of electrical engineering and computing

Zbirka svih lokalnih memorija oblikuje globalni adresni prostor kojem mogu pristupiti svi procesori u sustavu. Zbog takve organizacije memorije razlikujemo dvije vrste pristupa memoriji:

- brzi pristup memoriji (kraće vrijeme pristupa) kada procesor pristupa svojoj lokalnoj memoriji,
- sporiji pristup (dulje vrijeme pristupa) kada procesor pristupa "udaljenoj" memoriji koja je, zapravo, lokalna memorija nekog drugog procesora. Dulje vrijeme pristupa uzrokovano je dodatnim kašnjenjima jer se "udaljenoj" memoriji pristupa kroz prosječnu mrežu.

FER university of zagreb faculty of electrical engineering and computing

- Multiprocesorski sustavi oblikovani u skladu s modelom NUMA imaju veliki broj procesora, npr. nekoliko stotina ili tisuća, i zato se za toliki broj procesora teško može realizirati središnja memorija sa zahtjevanom, odnosno prihvatljivom propusnosti (engl. *memory bandwidth*).
- Svaki čvor u NUMA modelu sastoji od procesora, lokalne memorije, U/I podsustava i sučelja za pristup prospojnoj mreži. Ovisno o izvedbi NUMA modela, čvor može biti sastavljen od određenog broja procesora i lokalnih memorijskih modula tako da čini procesorsku nakupinu (engl. *cluster*).
- Primjer takve organizacije je multiprocesorski sustav velikih razmjera (engl. *large-scale multiprocessor*) Cedar.

Multiprocesorski sustavi temeljeni na modelu NUMA često se zbog načina na koji je ostvarena veza između procesora nazivaju i *labavo povezanim* (engl. *loosely coupled*).

**FER** university of zagreb  
faculty of electrical engineering and computing

*Multiprocesorski sustavi temeljeni na modelu COMA su poseban slučaj NUMA multiprocesorskog sustava u kojem je umjesto glavne memorije porazdijeljena priručna memorija tako da sve priručne memorije oblikuju globalni adresni prostor. Udaljeni pristup priručnoj memoriji ostvaruje se pomoću distribuiranih direktorija.*

**FER** university of zagreb  
faculty of electrical engineering and computing

- Multiprocesorski sustavi temeljeni na UMA modelu dijele zajednički memorijski prostor tako da procesori mogu međusobno izmjenjivati podatke instrukcijama *load* i *store* (zato se i nazivaju sustavi s dijeljenom memorijom).
- U sustavima temeljenim na NUMA modelu komunikacija između procesora odvija se porukama koje procesori izmjenjuju preko prospojne mreže – takvi se multiprocesorski sustavi nazivaju još i *multiprocesorski sustavi s proseljivanjem poruka* (engl. *message-passing multiprocessor*).

**FER** university of zagreb  
faculty of electrical engineering and computing

### 6. Koherencija priručne memorije u multiprocesorskom sustavu

- problem koherencije, i to za multiprocesorski sustav sa središnjom dijeljenom memorijom (UMA model)
- budući da svaki procesor ima svoju privatnu priručnu memoriju, ali dijeli i središnju zajedničku memoriju, u svakoj od priručnih memorija pohranjeni su "privatni" podaci koji se odnose na lokalni proces, ali i zajednički podaci koji se odnose na procese drugih procesora i koji su dohvaćeni u pojedine priručne memorije na temelju komunikacije procesora, odnosno njihovim pristupom zajedničkoj memoriji.

**FER** university of zagreb  
faculty of electrical engineering and computing

- istodobno se dijeljeni podatak nalazi u zajedničkoj memoriji, a njegove kopije distribuirane su u priručne memorije pojedinih procesora.

Sada nastupa problem!

Ako neki od procesora promijeni vrijednost zajedničkog podatka u svojoj priručnoj memoriji, taj se podatak treba promijeniti i u zajedničkoj memoriji, ali i u svim priručnim memorijama ostalih procesora koji koriste taj podatak (uvjet koherencije priručne memorije)

FER  
university of zagreb  
faculty of electrical engineering and computing

Primjer:

Pretpostavimo, radi jednostavnosti, da imamo multiprocesorski sustav koji ima samo dva procesora (jezgre) i da je ostvaren kao sustav sa središnjom dijeljenom memorijom. Dakle, sustav ima zajedničku memoriju, a svaki od procesora ima priručnu memoriju. Neka priručne memorije koriste tehniku obnavljanja sadržaja memorije „pohranjivanje-kroz“.

- Pretpostavimo da oba procesora (*procesor 1* i *procesor 2*) dijele zajedničku varijablu  $V$  koja je pohranjena u središnjoj dijeljenoj memoriji na lokaciji  $X$ .
- Pretpostavimo, također, da se u trenutku  $t = 0$  varijabla  $V$  ne nalazi u priručnim memorijama *procesora 1* i *procesora 2*.

FER  
university of zagreb  
faculty of electrical engineering and computing

- U trenutku  $t = 1$  *procesor 1* dohvaća (čita) varijablu  $V$  iz središnje memorije i pohranjuje je u svoju priručnu memoriju.
- U sljedećem trenutku ( $t = 2$ ) *procesor 2* dohvaća (čita) varijablu  $V$  iz središnje memorije i pohranjuje je u svoju priručnu memoriju.
- U ovom trenutku ( $t = 2$ ) koherencija podataka nije narušena: oba procesora imaju pohranjenu vrijednost varijable  $V$  jednaku onoj koja je pohranjena na memorijskoj lokaciji  $X$  u središnjoj memoriji.
- Pretpostavimo da u sljedećem trenutku ( $t = 3$ ) *procesor 1* mijenja vrijednost varijable  $V$  u  $V'$  i pohranjuje je u svoju priručnu memoriju. Uporabom tehnike „pohranjivanje-kroz“ *procesor 1* upisuje novu vrijednost  $V'$  na memorijsku lokaciju  $X$  u središnjoj memoriji.

FER  
university of zagreb  
faculty of electrical engineering and computing

- Sada u vremenskom trenutku  $t = 3$  imamo sljedeću situaciju: *procesor 1* u svojoj priručnoj memoriji ima pohranjenu novu vrijednost varijable  $V$ , tj.  $V'$ , središnja dijeljena memorija ima također na memorijskoj lokaciji  $X$  pohranjenu novu vrijednost  $V'$ .
- Međutim, *procesor 2* ima u svojoj priručnoj memoriji pohranjenu staru vrijednost varijable  $V$ . Ako u sljedećem trenutku ( $t = 4$ ) *procesor 2* dohvaća varijablu  $V$  iz svoje priručne memorije – dohvatit će staru vrijednost  $V$ , a ne  $V'$ . Došlo je do povrede koherencije – umjesto prave vrijednosti  $V'$  (koja je pohranjena i priručnoj memoriji *procesora 1* i u središnjoj memoriji), *procesor 2* koristit će se starom vrijednosti  $V$ .