UNIVERZITET METROPOLITAN BEOGRAD

CS120 - ORGANIZACIJA RAČUNARA

Paralelne arhitekture

Lekcija 13

PRIRUČNIK ZA STUDENTE

CS120 - ORGANIZACIJA RAČUNARA

Lekcija 13

PARALELNE ARHITEKTURE

UVOD

Paralelizam može biti predstavljen na različite načine. Na najnižem nivou, može biti napravljen na CPU čipu, kao što je pipelining i superskalarni CPU.

Mada su računari sve brzi i bolji, potreba za njima je sve veća i veća.
Paralelizam može biti predstavljen na različite načine. Na najnižem nivou, može biti napravljen na CPU čipu, kao što je pipelining i superskalarni CPU, koji su projektovani sa višestrukim funkcionalnim jedinicama. Oni mogu biti napravljeni od vrlo dugačkih instrukcijskih reči, sa podrazumevanim paralelizmom. Specijalne osobine mogu biti dodate u procesor i dopustiti da on opslužuje višestruke niti (threads) u istom trenutku. Najzad, višestruki procesori mogu biti realizovanu zajedno na istom čipu.

Na sledećem nivou, dopunski procesori (**CPU ploče**) sa dopunskim kapacitetom obrade, mogu biti dodati u sistem. Obično, ovi blokovski procesori imaju i posebne funckije, kao sto je mrežna paketna obrada, multimedijalno procesiranje ili kriptografija. Za posebne aplikacije,oni mogu imati faktor ubrzavanja od 5 za 10 puta.

Međutim, dobiti faktor ubrzavanja od sto, hiljadu ili milion puta, neophodno je umnožavanje procesora i njihov paralelni rad. Ova je glavna ideja za velike multiprocesore i multi računare (cluster računare). Konačno, sada je moguće skupiti zajedno više kompjutera u jednu organizaciju preko Interneta (grid computers). Ovi sistemi još nisu komercijalno napravljeni, ali su oni interesantni potencijal za budućnost.
Kada su dva CPU-a ili procesorska elementa, zatvorena zajedno, imaju veliku širinu propusnog opsega, malo vreme zadržavanja između njih i brzo izračunavaju. Za njih se može se reći da su tesno skopčani (engl. tightly coupled). Obratno, kada su oni daleko-razdvojeno, imaju nizak
propusni opseg, veliko vreme zadržavanja i sporo izračunavaju. Za njih se može se reći da su slobodno-labavo skopčani (engl. loosely coupled).
**U ovoj lekciji su prikazana projektovana načela za različite forme paralelizama, sa raznim primerima. Startuje sa paralelizmom na
jednom čipu, a postepeno se pomera ka većim i većim slobodno skopčanim sistemima, a na kraju će se završiti sa umreženim računarima.**

Poglavlje 1

FLINOVA TAKSONOMIJA ARHITEKTURA

PARALELNE ARHITEKTURE RAČUNARA

Ukratko, koliko god da je dostupno kompjuterske snage, za mnoge korisnike, naročito u nauci, inžinjerstvu, industriji, CPU snage nikada nije dovoljno.

Mada su taktne brzine stalno u porastu, one ne mogu biti povećavane beskonačno.

Mada su taktne brzine stalno u porastu, one ne mogu biti povećavane beskonačno. Brzine svetlosti su već glavni problem za dizajnere vrhunskih računara, a osobine elektrona i fotona koji se kreću brzo, su takođe nejasni u potpunosti. Toplotna disipacija je izuzetan problem procesora i superkompjutera. Najzad, veličine tranzistora se nastavljaju smanjivati do neke tačke, kada je tranzistor veličine nekoliko atoma u kvantnoj mehanici (Heisenbergov princip neodredenosti) i može postati veliki problem.

Dakle, da bi rukovali sve većom problematikom, računarske arhitekte grade paralelnu arhitekturu računara.

Ako ne bi bilo moguće napraviti kompjuter, sa jednim procesorom i ciklusnim vremenom ciklusa od 0.001 nsec, onda se to može uraditi sa jednim kompjuterom koji ima 1000 procesora, od kojih svaki ima vreme ciklusa od 1 nsec.

Iako ovako projektovani računar, koristiti spore procesore koji čine jedan kompjuter, ukupna CPU snaga je ista.

Paralelizam može biti predstavljen na različite načine. Na najnižem nivou, može biti napravljen na CPU čipu, kao što je pipelining i superskalarni CPU, koji su projektovani sa višestrukim funkcionalnim jedinicama.

Oni mogu biti napravljeni od vrlo dugačkih instrukcijskih reči, sa podrazumevanim paralelizmom. Specijalne osobine mogu biti dodate u procesor i dopustiti da on opslužuje *višestruke niti* (engl. threads) u istom trenutku. Najzad, višestruki procesori mogu biti realizovanu zajedno na istom čipu.

Na sledećem nivou, *dopunski procesori* (ekstra CPU ploče-engl.boards) sa dopunskim kapacitetom obrade, mogu biti dodati u sistem. Obično, ovi blokovski procesori imaju i posebne funkcije, kao što je mrežna paketna obrada, multimedijalno procesiranje ili kriptografija.

Za posebne aplikacije, oni mogu imati faktor ubrzavanja od 5 za 10 puta.

TAKSONOMIJA ZA PARALELNE RAČUNARE

Mnoge vrste odnosno klasifikacije za paralelne kompjutere su bila predložene i ugrađene odavno, tako je prirodno postaviti pitanje kako ih na neki način kategorisati u taksonomiju.

Mnoge vrste odnosno klasifikacije za paralelne računare su bile predložene i ugrađene odavno, tako je prirodno postaviti pitanje kako ih na neki način kategorisati u taksonomiju.

Mnogi istrazivači su se trudili, da miksuju rezultate (*Flynn, 1972; i Treleaven, 1985*). Jedino ***Flynn' s***, šema je zaživela, ali je i ona vrlo kruto aproksimirana.

Flinova klasifikacija je bazirana na dva koncepta:

- *instrukcijski nizovi* (engl. streams)
- *nizovi podataka* (engl. data streams).

Jedan instrukcijski niz odgovara jednom programskom brojaču, PC. Sistem sa n CPU-a ima n programskih brojača, otuda i n instrukcijskih nizova.

Tokovi instrukcija	Tokovi podataka	Ime	Primeri
1	1	SISD	Klasična Von Neumannova mašina
1	Višestruki	SIMD	Vektorski superračunar, matrični procesor
Višestruki	1	MISD	Nema?
Višestruki	Višestruki	MIMD	Multiprocesor, multiračunar

Slika 1.1 Flynn-ova taksonomija za paralelene računare [Izvor: Autor]

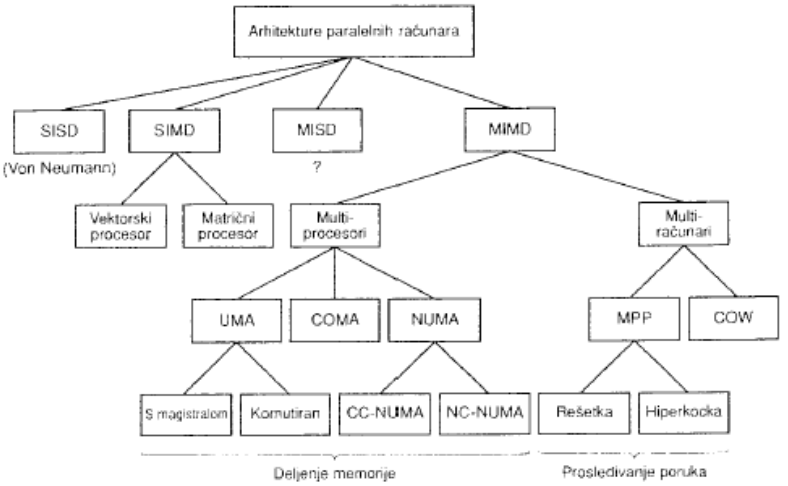
Data-nizovi se sastoje od skupova operanada. Na primer, praćenje temperature na više senzora, predstavlja višestruke data nizove, po jedan, za svaki senzor. Istrukcijski i data nizovi su u izvesnoj meri nezavisni, tako postoje četiri kombinacije, što je prikazano na slici 1.
SISD - Jednostruki tok intrukcija i jednostruki tok podataka (engl. Single Instruction Single Data) je upravo klasičan, sekvencijalni von Neumann kompjuter.
SIMD - Jednostruki tok instrukcija i višestruki tok podataka (engl. Single Instruction Multiple Data). Pentium SSE instrukcije su SIMD. Ipak, postoji jedna nova oblast, u kojoj se neke od ideja iz SIMD sveta koriste: to su stream procesori. Ove mašine su specifično dizajnirane, da upravljaju multimedijalnim operacijama i mogu postati važne u budućnosti.
MISD - Višestruki tok instrukcija i jednostruki tok podataka (engl. Multiple Instruction Single Data). Pitanje je da li uopšte MISD mašine postoje, mada neki ljudi smatraju pipeline mašine kao MISD.
Najzad, imamo

MIMD - Višestruki tok instrukcija i višestruki tok podataka (engl. Multiple Instruction Multiple Data) koje su sastavljene od višestrukih nezavisnih CPU-i koji rade kao deo većeg sistema. Najveći deo paralelnih procesora spada u ovu kategoriju. I multiprocesori i multikompjuteri su MIMD mašine.

PROŠIRENA TAKSONOMIJA ZA PARALELNE RAČUNARE

U proširenoj taksonomiji, MIMD kategorija će biti razdvojena u multiprocesore (mašine sa deljivom memorijom), i multikompjutere (mašine sa prosledjivanjem poruka).

SIMD se razdvaja u dve podgrupe. Prva je za numeričke super-kompjutere i druge mašinekoje operišu na vektorima, tj rade iste operacije na svakom vektorskom elementu. Druga je zaparalelni tip mašine, u kojem master kontrolna jedinica emituje instrukcije za više nezavisnih ALU-a. U proširenoj taksonomiji, *MIMD* kategorija će biti razdvojena u multi procesore (mašine sa deljivom memorijom), i multikompjutere (mašine sa prosledjivanjem poruka). Tri vrste multi-procesora postoje, zavisno od toga kako je zajednička memorija implementirana na njima.



Slika 1.2 Proširena taksonomija [Izvor: Autor]

Flinova taksonomija arhitektura

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 2

PARALELIZAM RAČUNANJA

PARALELIZAM NA NIVOU ČIPA

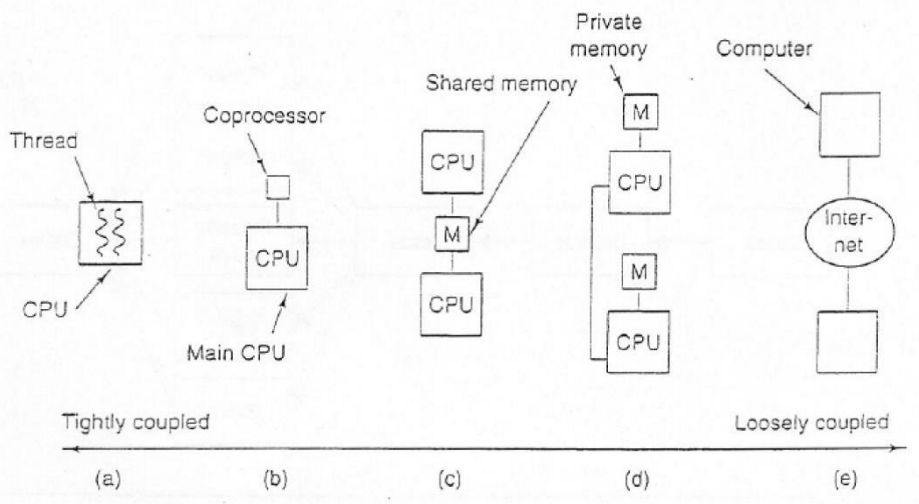
Kada su dva CPU-a ili procesorska elementa, zatvorena zajedno, imaju veliku širinu propusnog opsega, malo vreme zadržavanja između njih i brzo izračunavaju.

Medutim, dobiti faktor ubrzavanja od sto, hiljadu ili milion puta, neophodno je umnožavanjeprocesora i njihov paralelni rad .

Ova je glavna ideja za velike multiprocesore i multi-računare cluster računare).
Konačno, sada je moguće skupiti zajedno više kompjutera u jednu organizaciju preko Interneta (engl. grid-computers). Ovi sistemi još nisu komercijalno napravljeni, ali su oni interesantni potencijal za budućnost.
Kada su dva CPU-a ili procesorska elementa, zatvorena zajedno, imaju veliku širinu propusnog opsega, malo vreme zadržavanja izmedu njih i brzo izračunavaju. Za njih se može se reći da su tesno skopčani (engl. tightly coupled). Obratno, kada su oni daleko-razdvojeno, imaju nizak propusni opseg, veliko vreme zadržavanja i sporo izračunavaju. Za njih se može se reći da su slobodno labavo skopčani (engl. loosely coupled).

U ovom lekciji su prikazana projektovana načela za različite forme paralelizama, sa raznim primerima. Startuje sa paralelizmom na jednom čipu, a postepeno se pomera ka većim i većim slobodno skopčanim sistemima, a na kraju će se završiti sa umreženim računarima.

Ovaj spektar paralelnih tehnika je prikazan na slici 1.



Slika 2.1 Spektar paralelnih tehnika [Izvor: Autor]

PARALELIZAM NA NIVOU ČIPA (ON-CHIP PARALELLISM)

Jedan način za povećavanja propusne moći na jednom CPU čipu, je izvršavanje više instrukcija u isto vreme.

Jedan način za povećavanja propusne moći na jednom CPU čipu, je izvršavanje više instrukcija u isto vreme.

U ovoj sekciji, ćemo videti neke od načina ubrzanja pomoću paralelizama na nivou jednog čipa, uključujući:

- ***instrukcijski nivo paralelizama***
- *više-nitnu obradu* - (engl. multithreding)
- ***stavljanje više procesora na jedan čip***

Ali u svim slučajevima, ideja je više aktivnosti u jednom trenutku.

Na najnižem nivou, jedan način za postizanje paralelizama, je korišćenje više instrukcija u jednom taktu. Višeprocesni CPU dolaze u dve varijante:

1. ***superskalarni procesori***
2. *VLIW procesori* (engl. Very Long Instruction Word CPU)

U najvecem delu opštih CPU konfiguracija, na određenim tačkama pipelining-a, jednainstrukcija je spremna za izvršenje.

Superskalarni procesori su sposobni da obrađuju više instrukcija u jednom ciklusu.

INSTRUKCIJSKI NIVO PARALELIZMA

Ako jedna od instrukcija koja je potrebna funkcionalnoj jedinici nije dostupna, rezultat neće biti ostvaren, tj. instrukcija neće biti emitovana na izvršenje.

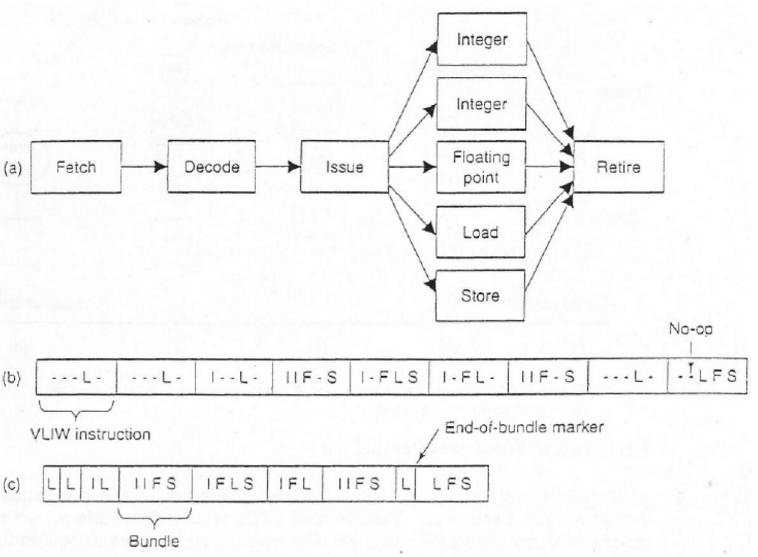
Broj instrukcija stvarno realizovanih, zavisi od realizacije procesora. Hardver određuje maksimalan broj koji može biti realizovan, obično dve do šest instrukcija. Medutim, ako jedna od instrukcija koja je potrebna funkcionalnoj jedinici nije dostupna, rezultat neće biti ostvaren, tj. instrukcija neće biti emitovana na izvršenje.

Druga forma instrukcijskog nivoa paralelizama je nađena u **VLIW** vrlo duge niti procesoru (engl. Very Long Instruction Word).

U originalnoj formi, VLIW mašine zaista imaju duge reci koje sadrže instrukcije koje upotrebljavaju više funkcionalnih jedinica (Slika 2 a**)).**

Na primer, pipeline na slici 2a), gde CPU ima pet funkcionalanih jedinica i može istovremeno obaviti dve integer operacije, jednu operaciju sa pokretnim zarezom, jednu operaciju punjenja (load), ili jednu operaciju upisa (store).

VLIW instrukcija za ovu mašinu sadrži pet opcoda i pet parova za operande (jedan opcod I jedan par operanda po funkcionalnoj jedinici). Sa 6 bita po opcodu, 5 bita po registaru, i 32 bita po memorijskoj adresi, instrukcije od 134 bita, su zaista sasvim dugačke.



Slika 2.2 CPU pipeline tehnika i VLIW [Izvor: Autor]

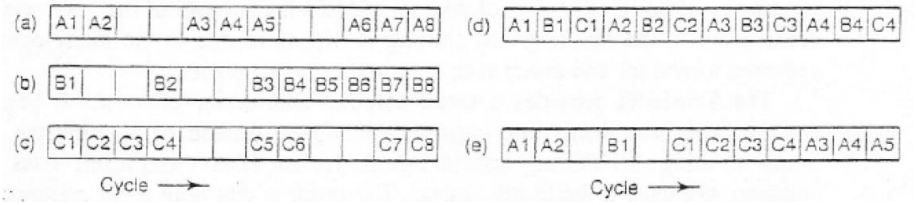
Ovaj dizajn pokazuje svoje mane, jer nije svaka VLIW instrukcija sposobna da iskoristi svaku funkcionalnu jedinicu, tako da je glavni operacioni kod za mnoge instrukcije je beskorisni **NOP** (engl. no operation), koji se korsiti kao punilac za **VLIW**, što je ilustrovano na slici 2 b). Stoga,moderne **VLIW** mašine, uvode način da obeleže skup instrukcija, koje mogu da se grupišu
zajedno (engl. bundle), na primer sa jednom "kraj od skupa" bitom (end of bundle bit), kao na slici.2 c). Procesor može izvršiti ceo skup (engl. bundle) odjednom.

TEHNIKA VIŠE-NITI NA ČIPU (ON-CHIP MULTI-THREADING)

Svi savremeni pipeline-CPU, imaju jedan svojstveni problem: kada se dogodi promašaj u L1 i L2 keš-memoriji, dolazi do dugog čekanja.

Instrukcijski nivo paralelizma nije jedina forma od paralelizma niskog nivoa. Drugi, je memorijski nivo paralelizma, u kojem se višestruke memorijske operacije emituju u isto vreme (engl. memory pipelining).
Svi savremeni pipeline-CPU, imaju jedan svojstveni problem: kada se dogodi promašaj u L1i L2 keš- memoriji, dolazi do dugog čekanja, sve dok tražena reč (i njegova pripadajuća keš linija ne budu učitane iz memorije u keš, a pipeline čeka za to vreme. Ovakvu situaciju
razrešavamo preko tehnike više-niti na čipu (engl. On Chip Multi threading), koja dopušta procesoru kontrolu nad više niti.

Ukratko, ako je 1 nit blokirana, CPU pruza šansu za pokretanje druge
niti i tako omogućava CPU hardveru da bude potpuno zaposlen. Na slici 3 a,b,c imamo tri niti, A, B, i C, za 12 mašinskih ciklusa.



Slika 2.3 Multi-threading tehnika [Izvor: Autor]

Za vreme prvog ciklusa, nit A izvršava instrukciju A1. Ova instrukcija se izvrši u jednom ciklusu (L1 hit), pa se u drugom ciklusu startuje instrukcija A2. Na žalost, ova instrukcija ima promašaj miss na nivou L1 kesa, tako da se sledeća 2 ciklusa propuštaju, dok reč stigne iz L2 keša. Nit A nastavlja tek u ciklusu 5. Slično, niti B i C takođe imaju povremena čekanja, kao na slici. U ovom slučaju, ako jedna instrukcija zastane, sledeća ne može biti emitovana-otpočeta.

Ima više modela za multi-threading.

Prvi pristup, je procesor sa sposobnostima da izvrši jednu instrukciju po taktu (engl. fine-grained multi-threading), koji je prikazan na slici 1 (d),

Procesor se izvršava po sistemu *Round Robin*, sa drugačijim nitima u susednim ciklusima kao na slici 1.d). Kada se pojavi četvrti ciklus, memorijska operacija koja je inicijalizovana u A1, je završena, tako instrukcija A2 može biti pokrenuta, čak i da traži rezultate od A1.
U ovom slučaju maksimalno kašnjenje je dva ciklusa, tako da sa tri niti
uvek imamo operaciju, završenu na vreme. Ako memorija izaziva kašnjenje od četiri ciklusa, moramo upotrebiti četiri niti da bi osigurali neprekidni rad CPU hardvera.

SIMULTANI MULTITHREDING

Sa superskalarnim CPU, treći mogući način za rad miltithredinga je raspoloživ i zove se simultani multithreding.

Pošto različite niti nemaju ništa jedan sa drugom, svaka od njih mora imati svoj registarski set.

Kada se jedna instrukcija pokrene, koristi se pokazivač, koji ukazuje na njen registarski set, tako da hardver zna koji registarski set koristi. Zato, čip je konstruisan tako da je prilagođen za maksimalan broj niti koje mogu biti pokrenute istovremeno.

Memorijske operacije nisu jedini razlog za gubljenje brzine. Ponekad, jedna instrukcija zahteva rezultat od prethodne instrukcije koja još nije kompletirana. Ponekad neka instrukcija ne može biti startovana, jer ona izvršava uslovni skok, a uslov još nije određen. U opštem slučaju, kada bi pipeline imao k faza, CPU bi morao da ima barem k niti, da pokrene *Round Robin* algoritam, tako da da bude uvek jedna instrukcija od jedne niti u pipeline, da ne bi došlo do konflikta, a CPU je tada u punoj brzini, tj opterećenju.

Naravno, ne može biti raspoloživo dovoljno niti, a potrebno je uvek imati onoliko niti koliko ima pipeline faza, tako da neki dizajneri imaju drugačije pristup (engl. coarse-grained multi threading), koji je prikazan na slici 3.e.
Ovde nit A startuje i nastavlja da izvršava instrukcije dok se ne stane, a kada instrukcija stane (L1 miss), izgubiće se jedan ciklus. To se
dešava sa instrukcijom A2.

U toj tački događa se prebacivanje na nit B i B1 počinje da se izvršava. Pošto prva instrukcijaniti B, B1 takođe napravi zastoj, dešava se prebacivanje druge niti B na nit C, instrukcija C1 se pokreće u 6. ciklusu. Jedan ciklus se uvek gubi, kada jedna instrukcija stane.

Coarse-grained multi threading je eventualno manje uspešan od *fine-grained multi threading*, ali ipak on je uvelikoj prednosti jer je potrebno mnogo manje niti, da bi se maksimalno angažovao CPU.

Bez obzira koji se multithreding koristi, na neki način je potrebno voditi evidenciju, koja operacija pripadati kojoj niti. Sa fine-rained multithreading, jedina mogucnost je, priključiti nit identifikator svakoj operaciji, dok se ona kreće kroz pipeline. Sa coarse-grained multi threading, postoji više mogućnosti: jedna mogućnost je kada se prebacuju niti, tada treba očistiti pipeline i tek onda startovati sledeću nit. U tom slučaju, samo je jedna nit u jednom trenutku u pipeline i njen identitet je poznat.

Sa superskalarnim CPU, treći mogući način za rad miltithredinga je paspoloživ i zove se simultani **multithreding**.

Ovo je približno fine-coarse-grained multithreading, u kome je pojedinačnim nitima dopušteno izvršavanje po dve instrukcije po ciklusu, ali ipak kada se dogodi zaustavljanje, instrukcije se odmah zahtevaju od sledeće niti u sekvenci, da bi CPU bio potpuno zauzet.

Poglavlje 3

HOMOGENI MULTIPROCESORI NA ČIPU

MULTIPROCESORI NA JEDNOM ČIPU (SINGLE-CHIP MULTIPROCESORS)

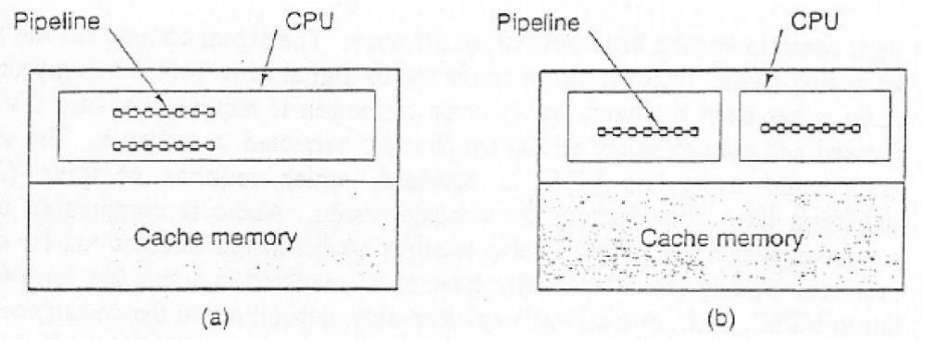
Stavljanjem dva CPU-a u isto kućiste, deljenjem memorije, diskova i mrežnih interfejsa, performanse servera mogu često biti udvostručeno, bez dupliranje cena.

Dok multithreading obezbeđuje značajne performanse za skromnu cenu, za neke aplikacije se se zahtevaju mnogo veće performanse od mogućnosti koju multitreding pruža. Da bi se dobila ova povećanja, razvijen je multiprocesorski čip.

Sa napretkom u VLSI tehnologiji, sada je moguće staviti dva ili više snažnih CPU–a na jedan čip. Ovakvi CPU-i, uvek dele isti L1 keš, L2 keš i glavnu memoriju, nazvani su multi procesori, a tipična oblast primene je velika Web serverska farma (server farm) koji sadrži više servera.

Stavljanjem dva CPU-a u isto kućiste, deljenjem memorije, diskova i mrežnih interfejsa, performanse servera mogu često biti udvostručeno, bez dupliranje cena.

Za jedan čip multiprocesora malih razmera, postoje dve dizajnerske tehnike.



Slika 3.1 Homogeni CPU u jednom chipu [Izvor: Autor]

U prvom slučaju, prikazanom na slici 1.a), ovo je stvarno samo jedan čip, ali ipak on ima sekundarni pipelining, čime se potencijalno duplira brzina izvršavanja instrukcija.

U drugom slučaju, prikazanom na slici 1.b), postoje 2 zasebna CPU jezgra (core) na čipu, svaki od njih sadrže pun CPU. Pod terminom jezgro, podrazumeva se veliko integrisano kolo, a to može biti CPU, I/O kontroler ili keš. Jezgra se smeštaju u čip na modularan način.

Drugi dizajn dopusta deljenje resursa, kao što su funkcionalne jedinice (keš memorija, na primer), da bude deljeni izmedu procesora. Stavljanje dva ili više CPU jezgra, na isti čip, je relativno lako uraditi.

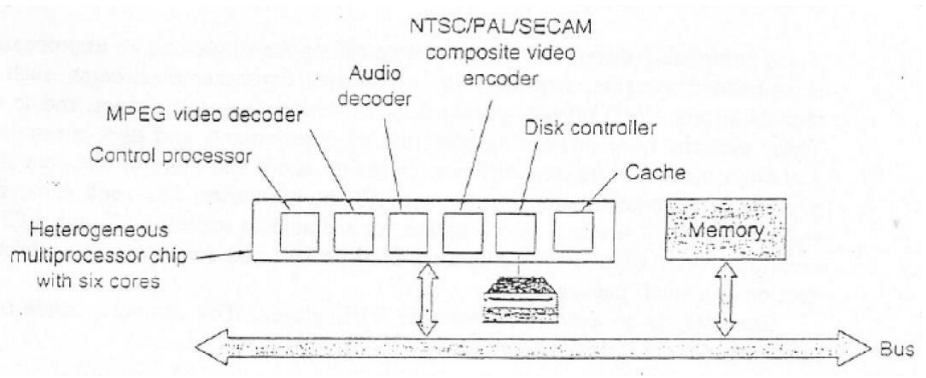
HETEROGENI MULTIPROCESORI

Ove opservacije brzo vode do heterogenog multiprocesorskog čipa koji sadrži više jezgra, posebno dizajniranih za audio-video aplikacije.

Potpuno drugačiji aplikacioni zahtevi postoje u takozvanim *embedded sistemima*, koji se koriste naročito u audiovizuelnoj potrošačkoj elektronici, kao sto su televizori, DVD plejeri,kamere, konzole za igru, celularni telefoni, i dr. Ovi sistemi zahtevaju visoke performanse. Mada ovi uređaji izgledaju drugačije u stvarnosti, mnogi od njih izgledaju, kao mali kompjuteri sa jednim ili više CPU, memorijom, I/O kontrolerom i nekim izbornim I/O uređajima.

Posmatrajmo, na primer, portabl DVD plejer. Računar unutar njega mora da radi sa sledećim funkcijama:

1. **kontrola servo mehanizma za read/write glave**
2. **analogno-digitalna konverzija**
3. **korekcija greške**
4. **dekripcija i digitalno upravljanje**
5. **MPEG-2 video dekompresija**
6. **Audio dekompresija**
7. **Kodovanje izlaza za NTSC, PAL ili SECAM televizora.**



Slika 3.2 Primer heterogenog procesora [Izvor: Autor]

Funckije jezgra prikazane na slici 2 su sve različite, svaka od njih je pažljivo dizajnirana da bude ekstremno dobra za najmanju moguću cenu.

Na primer, DVD video signal je komprimovan korisćenjem šeme Grupe eksperata za film, kao sto je *MPEG-2* (engl. Motion Picture Expert Group).

KOPROCESORI

Postojanje proverenih načina za izvršenje paralelizma na samom čipu dopušta nam korak napred i pokazuje kako se računar može ubrzati dodavanjem drugog specijalizovanog procesora.

Ona, je takođe sinhrona magistrala, sa maksimalnom propusnom moći od 300 Mbps. Dve magistrala su povezane mostom, slično kao veza, koja jekorišćena za konekciju između PCI i ISA magistrala. Treća magistrala je device register bus, ona je za vrlo male brzine, asinhronahandshaking
magistrala,koja dozvoljava procesorima da pristupe registrima I/O uređaja.
Koristi se za prenos male količine podataka.

Postojanje proverenih načina za izvršenje paralelizma na samom čipu, dopušta nam sada korak napred i pokazuje kako se računar može ubrzati sa dodavanjem drugog specijalizovanog procesora-koprocesora.

Ovi koprocesori mogu biti raznovrsni, od malih do velikih. U IBM-u 360 mainframes i svim njegovim naslednicima, postoje nezavisni I/O kanali za ulaz/izlaz. Slično, CDC 6600 je imao 10 nezavisnih procesora za I/O.

Koprocesori su bili upotrebljavani u grafici i aritmetici pokretne tačke. Svaki DMA čip može biti koprocesor.

U nekim slučajevima, CPU predaje koprocesoru jednu instrukciju ili skup instrukcija da ih koprocesor **izvrši, a u drugim slučajevima, koprocesor je manje/više nezavisan.**

Fizički, koprocesori mogu biti u odvojenom kućištu (IBM360, I/O kanali), integrisan na ploči mrežni procesori), kao deo glavnog CPU čipa (FPU pokretna tačka).

U svim slučajevima, treba razlikovati procesor i koprocesor, koji je tu samo kao pomoć.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

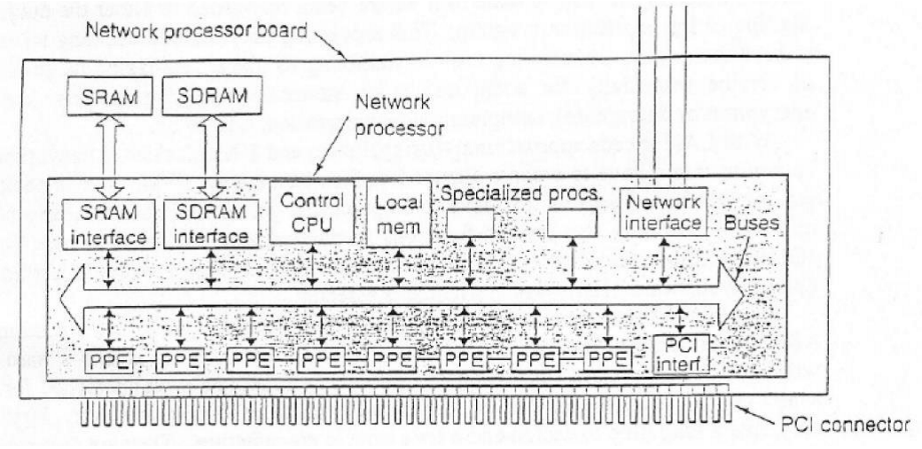
MREŽNI PROCESORI

Mrežni procesori su programabilni uređaji, koji mogu rukovati dolaznim i odlaznim paketima u realnom vremenu.

Mnogo vrsta uredaja su povezani na mrežu. Krajnji korisnici imaju*lične računare* (desktop ili notebook) , *PDAs* (palmtops), i mobilne telefone. Kompanije imaju PC i servere kao krajnje sisteme. Međutim, postoje brojni uredaji čija je funkcija posredovanje na mreži, uključujući rutere, switch uređaje, firewall uređaje, Webproxy uređaje, *balanser za optrećenja na mreži* (engl. load balancer). Zavisno od paketa i mreže, jedan dolazni paket može imati različite potrebe za procesiranjem pre nego što bude prosleđen ka odredištu ili ka aplikacionom programu.

Sa LAN brzinama od 40 Gbps i 1KB paketima, mrežni računari mogu procesirati skoro 5 milion paketa/sec.

Sa 64-bajtnim paketima, broj paketa koje može procesirati po sekundi, podiže se na blizu 80 miliona. Ove funkcije se dešavaju u 12-200ns (uz to uvek, višestruko kopiraju pakete) pa im je hardver bitniji od softvera.
Mrežni procesori su programabilni uređaji, koji mogu rukovati dolaznim i odlaznim paketima u realnom vremenu. Opšti dizajn je plug-in kartica, koja sadrži-mrežni procesor na čipu, sa memorijom i odgovarajućom logikom. Jedna ili više mrežni linija se konektuju na ploču i one su rutiraju do mrežnih procesora. Paketi se tamo izvlače,procesiraju i šalju na različite mrežne linije (za ruter) ili se šalju izvan, ka glavnoj sistemskoj magistrali, PCI.
Tipična mrežno-procesorska kartica i čip su prikazani na slici 4.



Slika 3.3 Mrežni procesor [Izvor: Autor]

Poglavlje 4

Multiprocesori i multiračunari

VIŠEPROCESORSKI SISTEMI (SHARED-MEMORY MULTIPROCESSORS)

U bilo kom paralelnom računarskom sistemu, procesori koji rade na različitim delovima istog posla-procesa, moraju komunicirati jedan sa drugim da bi razmenjivali informacije.

Sistemi sa višestrukim procesorima se dele u dve klase:

- **multiprocesorski sistemi ili paralelni sistemi i**
- **multikompjuterski sistemi**

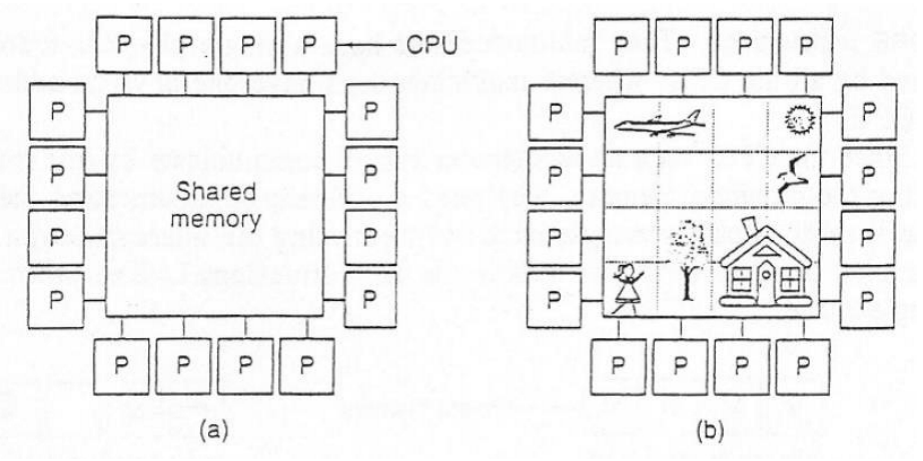
U bilo kom paralelnom računarskom sistemu, procesori koji rade na različitim delovima istog posla procesa, moraju komunicirati jedan sa drugim da bi razmenjivali informacije.

Dva različita dizajna, *multiprocesori* I *multikompjuteri*, su predloženi i implementirani. Ključna razlika između ova dva dizajna je prisustvo ili odsustvo deljive memorije (engl. shared memory).

Ove razlike su i uticale na to kako će ovi sistemi biti dizajnirani, ugrađivani, iprogramirani, kao i na njihovu cenu.
Paralelni računari u kojima svi CPUi dele zajedničku memoriju nazivaju se multiprocesori i prikazani su simbolički na slici 1. Svi procesori koji rade zajedno u multiprocesorskom sistemu, mogu deliti jedan virtuelni adresni prostor, mapiran na zajedničkoj memoriji.
Bilo koji proces može čitati ili pisati reč u memoriji, izvršavanjem **LOAD/STORE** instrukcija.Ništa drugo mu nije potrebno.Hardver radi ostalo. **Dva procesa mogu komunicirati, tako što jedan od njih upisuje podatke u memoriju, a drugi ih čita.**

Sposobnost da dva (ili više) procesa komuniciraju između sebe sa prostim čitanjem i pisanjem po memoriji je razlog zašto su multiprocesori tako popularni. Oni su lak model za programere,da ih razumeju/programiraju i podesni su za široku problematiku. Na primer, posmatrajmo program koji obrađuje bitmapiranu sliku i lista sve objekte u njoj. Jedna kopija ove slike se nalazi u memoriji kao sto je pikazano na slici 1b).

Svaki od 16 CPU-a izvršava po jedan proces, kome je dodeljena jedna 16-tina memorije za analizu. Ipak, svaki proces ima pristup celoj slici-memoriji, što je bitno, pošto neki objekati mogu okupirati više sekcija.



Slika 4.1 Multiprocesorski sistem [Izvor: Autor]

MULTIRAČUNARI

Drugi mogući dizajn za paralelnu arhitekturu, je dizajn, u kome svaki CPU ima svoju sopstvenu memoriju, dostupnu samo sebi. Takav dizajn je nazvan multiračunar.

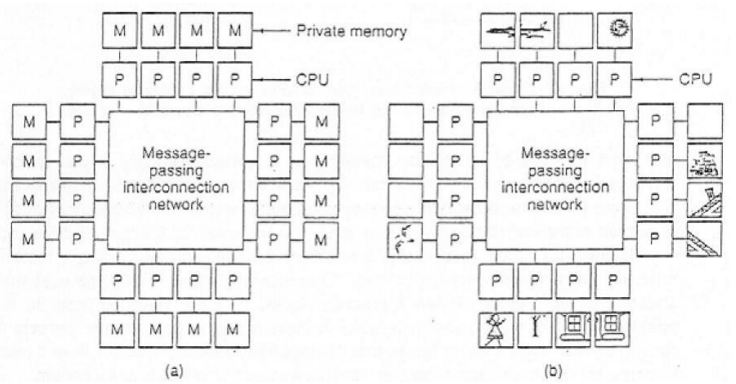
Ako proces otkrije da jedan od njegovih objekata produžava izvan granica sekcije, on tada sledi objekat u sledeću sekciju i čita reči i iz te sekcije.

U ovom primeru, neki objekti će biti otkriveni od više procesa, tako da je potrebna neka koordinacija na kraju, koja određuje koliko ukupno ima kuća, drveća i aviona.Pošto svi CPUi u multiprocesoru vide istu memoriju, zato postoji samo jedna kopija operativnog sistema. Stoga, postoji samo jedna tabela stranica i jedna tabela procesa. Kada se proces blokira, njegov CPU snima njegova stanja u tabele operativnog sistema i tada gleda u tu tabelu da bi pronašao novi proces koji će dalje da se izvršava.

Po tome se razlikuju multi-procesori od multi-računari, multiprocesor ima jednu kopiju operativng sistema a kod multi-računara svaki računar ima svoju kopiju operativnog sistema.

Multiprocesor, kao i svi kompjuteri, mora imati I/O uređaje, kao što su diskovi, mrežni adapteri, i druga oprema. U nekim multi-procesorskim sistemima, samo izvesni CPU-i imaju pristup I/O uređajima, i prema tome imaju specijalne I/O funkcije. U drugim sistemima, svaki CPU ima jednak pristup za svaki I/O uređaj. Kada svaki CPU ima jednak pristup za sve module memorije i sve I/O uređaje, taj sistem je nazvan **SMP** (**Simetričan Multiprocesor**).
Drugi mogući dizajn za paralelnu arhitekturu, je dizajn, u kome svaki CPU ima svoju sopstvenu memoriju, dostupnu samo sebi. Takav dizajn je nazvan multikompjuter, ili ponekada distribuirani memorijski sistem i on je ilustrovan na slici 2. Ključni aspekt razlikovanja multi-kompjutera od multi-procesora je taj što svaki CPU u multikompjuteru ima svoju privatnu, lokalnu memoriju kojoj može pristupiti preko **LOAD/STORE** instrukcija, ali ni jedan drugi CPU ne može pristupiti njegovoj memoriji preko LOAD/STORE instrukcija.

Prema tome, multi-procesori imaju jedan fizički adresni prostor deljen od svih CPU, dok multi-kompjuteri imaju jedan fizički adresni prostor po jednom CPU.



Slika 4.2 Multikomjuterski sistem [Izvor: Autor]

MULTIRAČUNARI - OPIS RADA CPU-OVA

Pošto CPU-i na multiračunaru, ne mogu komunicirati sa čitanjem i pisanjem po zajedničkoj memoriji, njima je potreban drugačiji komunikacioni mehanizam.

Pošto CPU-i na multi-računaru, ne mogu komunicirati sa čitanjem i pisanjem po zajedničkoj memoriji, njima je potreban drugačiji komunikacioni mehanizam. To oni obavljaju preko prenosa poruka, korišćenjem mreže između njih.

Odsutnost hardverski deljene memorije na multikompjuterima ima bitne implikacije za softversku strukturu. Postojanje jednog virtualnog adresnog prostora svim procesima omogućava da čitaju i pišu po celoj memoriji, samo sa izvršenjem LOAD/STORE instrukcije, a to je nemoguće na multi-računar.

Na multi-računaru, komunikacija između procesa često koristi softverske primitive kao što su send/receive. One daju softveru drugačiju, i znatno složeniju strukturu, nego na multi-procesoru.

Send/receive primitive su takođe sredstvo za ispravno deljenje podataka, i njihovo postavljanje u optimalan položaj na multi-računaru.
Ukratko, programiranje multi-računara je mnogo teže od programiranja multi-procesora.

Pored ovih uslova, zašto bi bilo ko gradio multi-računara, kada su multi-procesori lakši za programiranje?

Odgovor je jednostavan: veliki multi-računari su mnogo jednostavniji i jeftiniji za izgradnju od multi-procesora sa istim brojem CPU-a.

Implementacija memorije, deljene između nekoliko stotina CPU-a je suštinski dobra, ali je izrada multi-računara sa 10.000 CPU-a ili više, još bolja. Postoje multi-računari sa 50.000 CPU.

Ovako imamo dilemu: multi-procesori su teški za realizaciju ali laki za programiranje, a multi-računari su laki za realizaciju ali teški za programiranje. Ovo zapažanje nas dovodi do ideje stvaranja hibridnog sistema koji je relativno lak za izgradnju i relativno lak za programiranje. U ovom projektu, glavna je realizaciju zajedničke memorije koja može biti implementirana na različite načine, a svaki ima svoje prednosti i mane.

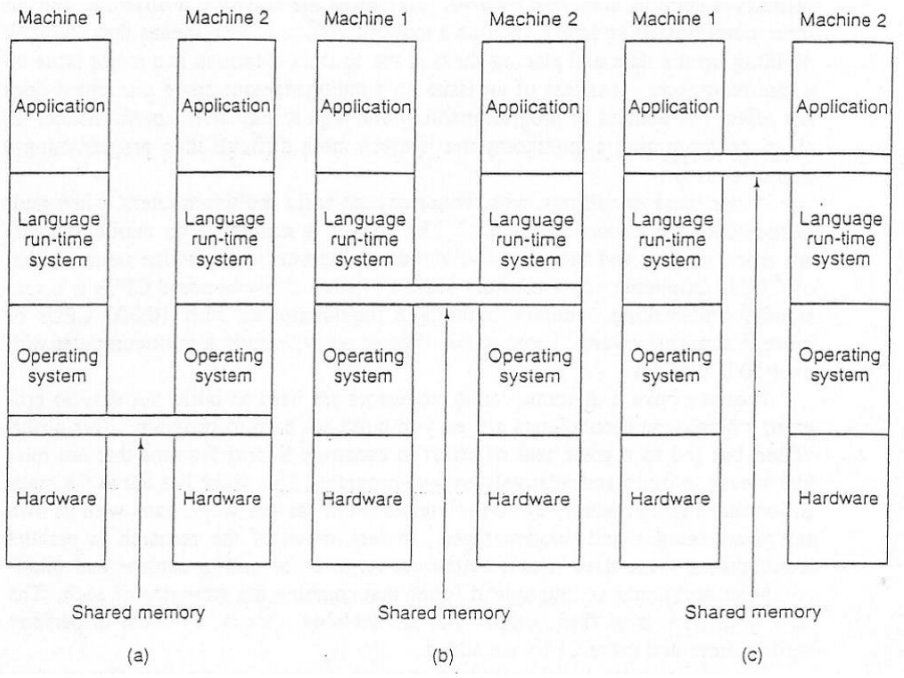
HIBRIDNI SISTEMI

U hibridnom dizajnu, ima jedna kopija operativnog sistema sa jednim skupom tabela, kao na primer tabela alokacije memorije.

Jedan način izgradnje hibridnih sistema je baziran na činjenici da savremeni kompjuterski sistemi nisu monolitni, nego su konstruisani kao serija slojeva (engl. layers **).** Što znači da je otvorena mogućnost za implementaciju deljene memorije na bilo koji od nekoliko slojeva, kao
što je prikazano na slici 3.

Na slici. 3.a) vidimo da se deljena memorija implementira u hardveru, kao pravi multiprocesor. U ovom dizajnu, ima jedna kopija operativnog sistema sa jednim skupom tabela, kao na primer tabela alokacije memorije.
Kada proces zahteva više memorije, on se obrati operativnom sistemu, koji onda gleda u njegovoj tabeli za slobodnu stranicu i mapira
stranicu unutar adresnog prostora procesa koji je traži. U tom slučaju, to je jedna-jedinamemorija i operativni sistem čuva zapis o svakom procesu i njegovim stranicama. Postoji mnogo načina za implementaciju hardverski deljene memorije.

Druga mogućnost je koristiti multikompjuterski hardver i imati operativni sistem koji simulira deljenu memoriju, tako što kreira jedinstveni-zajednički virtuelni adresni prostor za sve računare. Ovo se naziva distribuirana deljena memorija **DSM** (engl. Distributed shared memory), pri čemu se svaka stranica locira u jednoj od memorija svakog računara, slika 3.(b).



Slika 4.3 Realizacija deljive memorije [Izvor: Autor]

Treća mogućnost je imati korisnički nivo implementacije za deljenu memoriju. U ovom slučaju, programski jezik obezbeđuje neku vrstu memorijske apstrakcije, koja se implementira na kroz kompajlirani runtime sistem.

Poglavlje 5

GRID I KLAUD RAČUNARI

GRID RAČUNARI

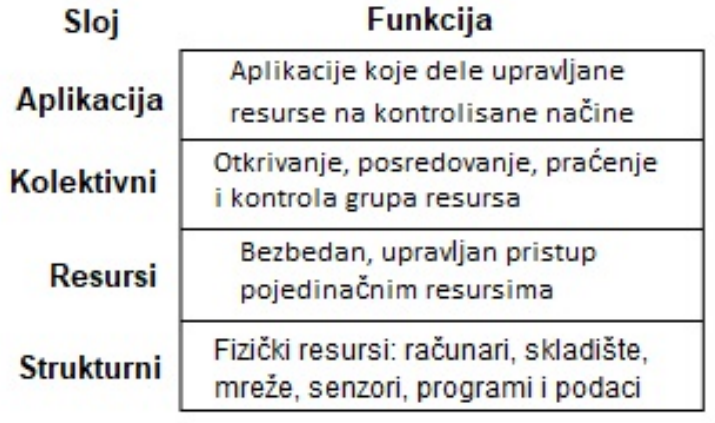
Virtuelizacija je simulacija softvera ili hardvera na kome drugi softver radi.

Mnogi od današnjih problema u nauci, inženjerstvu, industriji, ekologiji i u drugim područjima velikih su razmera i interdisciplinarne prirode.

Za njihovo rešavanje potrebni su stručnost, iskustvo, znanje, oprema, softver i podaci iz više organizacija, često iz različitih zemalja.

Evo nekoliko primera:

1. **Naučnici koji pripremaju misiju na Mars.**
2. **Konzorcijum koji realizuje složen projekat (npr. branu iii letelicu).**
3. **Međunarodni tim koji koordinira slanje pomoći u područja pogođena prirodnim katastrofama.**



Slika 5.1 Slojevi grid nivoa [Izvor: Autor]

Neke od ovih aktivnosti su dugoročne saradnje a neke druge su pak jednokratne, ali im je svima zajedničko to da brojne pojedinačne organizacije i resursi moraju udružiti rad kako bi ostvarili zajednicki cilj.
Donedavno je bilo vrlo teško ostvariti razmenu podataka i deljenje resursa između različitih organizacija čiji računari imaju različite operativne sisteme, baze podataka i protokole. Međutim, narasla potreba za saradnjom velikog obima između više organizacija podstakla je razvoj sistema i tehnologija za povezivanje udaljenih racunara u mrežu zvanu **rešetka** (engl. grid).

Rešetka u izvesnom smislu predstavlja sledeći korak duž ose na slici 1.

Ona se može zamisliti kao veoma veliki, međunarodni, labavo povezan, heterogeni klaster.

Svrha rešetke je da obezbedi tehničku infrastrukturu grupi organizacija koje zbog zajedničkog cilja žele da se udruže u virtuelnu organizaciju.

Virtuelna organizacija s brojnim i promenljivim članstvom mora biti fleksibilna tako da omogući članovima saradnju u područjima koja im se čine odgovarajućim i da im istovremeno omogući kontrolu nad njihovim sopstvenim resursima koliku god to žele.

Da bi takva virtuelna organizacija mogla da funkcioniše istraživači mreža razvijaju odgovarajuće usluge alate i protokole.

HIJERARHIJSKI SLOJEVI KOD GRID RAČUNARA

Jedan od načina modelovanja rešetke je pomoću hijerarhijskih slojeva.

Strukturni sloj (engl.fabric layer) na dnu je predstavljen skupom komponenti od kojih je napravljena mreža.To uključuje CPU, diskove, mreže i senzori na strani hardvera, i programi i podaci sa softverske strane. Ovo su resursi koje mreža daje na raspolaganje na kontrolisan način.

Na sledećem nivou je *sloj resursa* (engl. resource), koji upravlja pojedinačnim resursima. U mnogim slučajevima, resursi koji učestvuju u mreži imaju lokalni proces koji upravlja njime i omogućava pristup daljinskim korisnicima. Ovaj sloj pruža jedinstveni interfejs ka višim slojevima za ispitivanje karakteristika i statusa, praćenje individualnih resursa i njihovo bezbedno korišćenje.

Sledeći je *kolektivni sloj* (engl. collective), koji se bavi grupama resursa. Jedna od njegovih funkcije su pronalaženje resursa, kojim korisnik može da locira dostupne cikluse CPU, prostor na disku, ili specifične podatke. Kolektivni sloj je takođe odgovoran za replikaciju podataka, upravljanje primanjem novih članova i resursa mreža, održavanje baza podataka sa pravilima o tome ko šta može da koristi.

Na vrhu je *sloj aplikacija* (engl. application), u kome se nalaze programi korisnika. On koristi niže slojeve da bi pribavio potrebne dozvole da bi pribavio potrebne dozvole za upotrebu određenih resursa, podnosio zahteve za korišćenje, nadgledao kretanje ovih zahteva, obrađivao greške, i obaveštavao korisnike o rezultatima.

Bezbednost je ključ uspešne rešetke . Vlasnici resursa skoro uvek insistiraju, na održavanju stroge kontrole nad svojim resursima i žele da utvrde ko će ih koristi, koliko dugo i kom stepenu. Bez dobrog obezbeđenja, nijedna organizacije neće staviti svoje resurse na raspolaganje. S druge strane, ako je korisnik treba da ima nalog i lozinku za korišćenje za svaki računar u rešetci koji je želeo da koristi, upotreba rešetke je nepodnošljivo naporna. Zbog toga, rešetka mora da stvori bezbednosni model koji će pomiriti ove dve suprostavljene potrebe.

Jedna od ključnih karakteristika bezbednosnog modela jeste jedinstveno prijavljivanje.

Prvi korak pri korišćenju rešetke je potvrđivanje identiteta korisnika i dobijanje dozvola, digitalno potpisanog dokumenta u kome stoji za koga se posao obavlja. Dozvole se mogu prenositi (delegirati) tako da se, kada pri nekom izračunavanju zatreba i dodatno izračunavanje, lako može identifikovati proces potomak.

Kada se dozvole pokažu na udaljenom računaru, one treba da se preslikaju u lokalni sistem bezbednosti.

Na kraju u rešetkama su potrebni i mehanizmi za stvaranje pravila pristupa, za njihovo održavanje i ažuriranje.

KLAUD RAČUNARI

Računarstvo u oblaku (engl. cloud computing) predstavlja isporuku računarskih resursa i skladišnih kapaciteta kao uslugu za heterogenu grupu krajnjih korisnika.

Termin *Računarstvo u oblaku* (engl.Cloud Computing) može da se prevede na srpski jezik na više načina, kao što su računarstvo u oblaku, računarstvo u oblacima, klaud kompjuting, klaud računarstvo, itd.

Bez obzira što u literaturi koja je dostupna na srpskom jeziku postoje brojni prevodi ovog engleskog termina, u ovoj lekciji je korišćen termin **klaud računarstvo.**

Prvi komponenta definisanja pojma klaud računarstva odnosi se na nivo „kako klaud radi“.

Kada se sagledava na ovom nivou, klaud računarstvo je sposobnost i mogućnost korišćenja skupa računarskih resursa u cilju isporuke aplikacija i usluga klijentima.

Na ovaj način, klaud predstavlja pomeranje upravljačkih aplikacija sa nivoa određenog resursa, kao što su računari, serveri, memorija, skladišni prostor, na upravljanje na nivou aplikacija, gde se ovaj nivo može dinamički da se širi i sakuplja na zahtev klijenta, korišćenjem skupa zajedničkih resursa.

Arhitektura klaud računarstva odnosi se na usluge i podatke koji egzistiraju u deljenom, dinamičkom i skalabilnom skupu resursa zasnovanom na tehnologijama virtuelizacije i/ili skaliranim aplikativnim okruženjima.

Klaud računarstvo obuhvata skup računarskih resursa poput hardverskih uređaja,sistema za upravljanje bazama podataka, sistema za skladištenje podataka i interfejsa, koji obezbeđuju isporuku računarskih usluga u formi servisa.

Servisi klaud računarstva obezbeđuju isporuku aplikacija, virtuelnog prostora za skladištenje podataka ili celokupne računarske infrastrukture putem globalne računarske mreže u cilju zadovoljavanja korisničkih zahteva i/ili potreba.

Usvajanje koncepta klaud računarstva donosi brojne prednosti za organizacione korisnike i kao najznačajnije prednosti mogu da se izdvoje sledeće:

- brža nabavka i implementacija opreme,
- manji početni kapitalni troškovi i
- mogućnost trenutnog proširivanja računarskih resursa.

TIPOVI KLAUD RAČUNARSTVA PREMA MODELU USLUGA

Klaud računarstvo je stil računarstva u kome se skalabilni i elastični računarski resursi dostavljaju klijentima (potrošačima) u vidu usluge (eng. as a service) korišćenjem tehnologije.

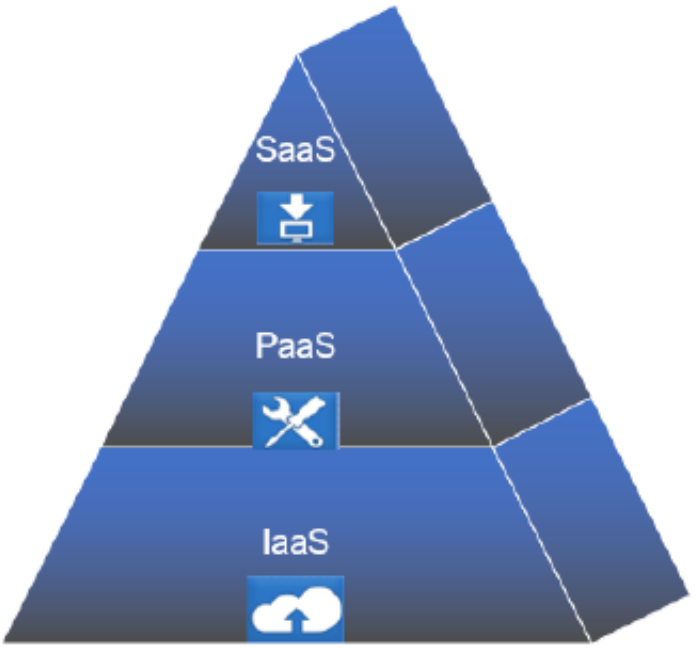
Prema viđenju organizacije **CompTIA** (engl. Computing Technology Industry Association), mogu da se izdvoje tri osnovna modela usluga klaud računarstva:

- **softver kao usluga** (eng. Software as aService – ***SaaS***);
- **infrastruktura kao usluga** (eng. Infrastructure as aService – ***IaaS***) i
- **platforma kao usluga** (eng. Platform as aService – ***PaaS***).

Mnogi provajderi klaud usluga koriste opisnije nazive modela klaud usluga, u ciljupostizanja većeg marketinškog efekta, kao što su:

- **komunikacije kao usluga** (eng.Communication as aService ***CaS***),
- **bilo šta kao usluga** (eng. Anything as aService - ***XaaS***),
- **desktop kao usluga** (eng. Desktop as aService – ***DaaS***),
- **poslovni proces kao usluga** (eng. Business Process as aService - ***BPaaS***) i
- **skladište podataka kao usluga**(eng. Storage as aService).

Naravno, sve navedene usluge se uklapaju u tri osnovna modela usluga klaud računarstva. Kao napomena navodi se da se stalno kreiraju novi modeli usluga klaud računarstva



Slika 5.2 Tipovi klaud računarstva prema modelu usluga [Izvor: Autor]

Poglavlje 6

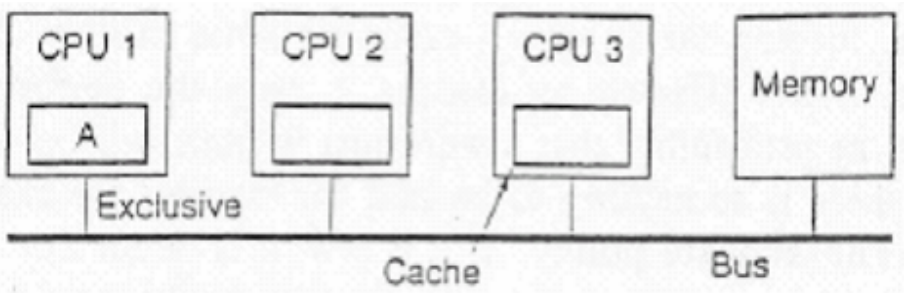
Pokazne vežbe

PRIMER 1:

Primer #1 (10 min)

ZADATAK (10 minuta):
U sistemu na slici 1, primenjuje se šema pisati kroz **WT** (engl. write through) keš koherentni protokol.
Sistem ima sledeće karakteristike:

- **3 procesora sa sopstvenim keš-om i snooping keš koherentnim protokolom**
- **CPU su 32bitni**
- **WT (write through) keš koherentni protokol**
- **deljiva RAM memorija raspoloživa preko zajedničke magistrale**
- **R1 ima ulogu akumulatora**
- **početne vrednosti memorijskih reči A=1, B=2, C=6**



Slika 6.1 Sistem sa WT keš koherentnim protokolom [Izvor: Autor]

Date su sekvence programa koje izvršavaju ova 3 procesora, respektivno:

	CPU1:	CPU2:	CPU3:
step1:	mov R2, 3	mov R5, 4	mov R3, 6
step2:	load A	load B	load C
step3:	add R1, R2	mul R1,R5	div R1, R3
step4:	store A	store B	store C

Slika 6.2 Sekvence programa [Izvor: Autor]

Koliko je vrednost memorijskih lokacija A, B i C posle ovih sekvenci

Rešenje:

Pažljivom analizom sva 3 programa dolazimo do sledećeg zaključka:

CPU1	CPU2	CPU3
A+3->A	Bx4->B	C/6->C
1+3->4	2x4->8	6/6->1
A=4	B=8	C=1

Slika 6.3 Rešenje zadatka [Izvor: Autor]

PRIMER 2:

Primer #2 (10 min)

ZADATAK (10 minuta):
U sistemu na slici 1, primenjuje se šema pisati kroz WT (engl. write through) keš koherentni protokol.
Sistem ima sledeće karakteristike:

- **3 procesora sa sopstvenim keš-om i snooping keš koherentnim protokolom**
- **CPU su 32bitni**
- **WT (write through) keš koherentni protokol**
- **deljiva RAM memorija raspoloživa preko zajedničke magistrale**
- **R1 ima ulogu akumulatora**
- **početne vrednosti memorijskih reči A=1, B=2, C=6**

Date su sekvence programa koje izvršavaju ova 3 procesora, respektivno:

	CPU1:	CPU2:	CPU3:
step1:	mov R2, 3	mov R5, 4	mov R3, 6
step2:	load A	load B	load C
step3:	add R1, R2	mul R1,R5	div R1, R3
step4:	store A	store B	store C

Slika 6.4 Sekvence programa [Izvor: Autor]

Da li se memorijske lokacije nalaze u kešu i kolike su njihove vrednosti u kešu?

Rešenje:
U step 2, dešavaju se load instrukcije koje dovode memorijske reči A, B i C zajedno sa ostatkom linije u keš memoriju CPU1, CPU2, CPU3

respektivno. Nakon toga u step 3 se obavlja ALU operacija kroz registre, a iza toga step 4 u kome se obavljaju store operacije za reči A, B i C.
Nakon step2 imamo sledeću situaciju:

memorija	A=1	B=2	C=6
CPU1 keš	cached(A)=1		
CPU2 keš		cached(B)=2	
CPU3 keš			cached(C)=6

Slika 6.5 Stanje keša nakon koraka 2 [Izvor: Autor]

U step 4, dešavaju se store operacije koje prati lokalni keš hit i remote cache miss, tako da nema invalidacije. Sve store operacije su local write hit, upis ide i u keš i u memoriju, simultano.

memorija	A=4	B=8	C=1
CPU1 keš	cached(A)=4		
CPU2 keš		cached(B)=8	
CPU3 keš			cached(C)=1

Slika 6.6 Stanje keša nakon koraka 4 [Izvor: Autor]

PROJEKTNI ZADATAK IZ PREDMETA CS120

Predmet CS120 od predispitnih obaveza sadrži i PZ koji se sastoji od tri mini-projekta.

Na predmetu CS120 - Organizacija računara svaki student samostalno radi projektni zadatak (PZ), koji se sastoji od tri mini-projekta.

Mini-projekti brane se u petoj, desetoj i petnaestoj nedelji semestra!

Svaki student dužan je da tok izrade projekta izveštava svake nedelje u obliku kratkih izveštaja (do jednog pasusa) preko LAMS lekcija u okviru dodatnih aktivnosti interaktivnih lekcija.

O formatu mini-projekata svi studenti (tradicionalni i Internet) biće obavešteni mejlom od strane predmetnog profesora/asistenta.

Poglavlje 7

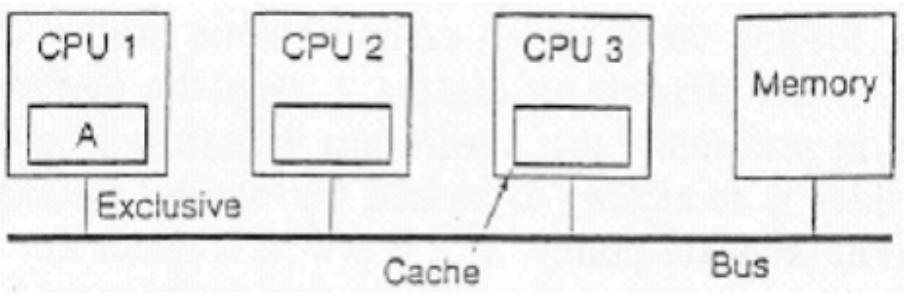
Zadaci za samostalni rad

ZADACI ZA SAMOSTALNI RAD 1,

Dodatni primeri za samostalno vežbanje #1 (15 minuta)

ZADATAK (15 minuta):
U sistemu na slici 1, primenjuje se šema pisati kroz WT (engl. write through) keš koherentni protokol.
Sistem ima sledeće karakteristike:

- **3 procesora sa sopstvenim keš-om i snooping keš koherentnim protokolom**
- **CPU su 32bitni**
- **WT (write through) keš koherentni protokol**
- **deljiva RAM memorija raspoloživa preko zajedničke magistrale**
- **R1 ima ulogu akumulatora**
- **početne vrednosti memorijskih reči A=1, B=2, C=6**



Slika 7.1 Sistem sa WT keš koherentnim protokolom [Izvor: Autor]

Date su sekvence programa koje izvršavaju ova 3 procesora, respektivno:

	CPU1:	**CPU2:**	**CPU3:**
step1:	mov R2, 3	mov R5, 4	mov R3, 6
step2:	load A	load A	load C
step3:	add R1, R2	load B	div R1, R3
step4:	store A	mul R1, R5	store C
step5:	mov R2, R6	store B	add R5, R3
step6:	load B	load A	mul R1, R5

Slika 7.2 Sekvence izvršavanja programa [Izvor: Autor]

Koliko je vrednost memorijskih lokacija A, B i C posle ovih sekvenci

Rešenje:

Pažljivom analizom sva 3 programa dolazimo do sledećeg zaključka:

CPU1	CPU2	CPU3
A+3->A	Bx4->B	C/6->C
1+3->4	2x4->8	6/6->1
A=4	B=8	C=1

Slika 7.3 Rešenje zadatka [Izvor: Autor]

ZADACI ZA SAMOSTALNI RAD 2,

Dodatni primeri za samostalno vežbanje #2 (15 minuta)

ZADATAK (15 minuta):
U sistemu na slici 1, primenjuje se šema pisati kroz **WT** (engl. write through) keš koherentni protokol.
Sistem ima sledeće karakteristike:

- **3 procesora sa sopstvenim keš-om i snooping keš koherentnim protokolom**
- **CPU su 32bitni**
- **WT (write through) keš koherentni protokol**
- **deljiva RAM memorija raspoloživa preko zajedničke magistrale**
- **R1 ima ulogu akumulatora**
- **početne vrednosti memorijskih reči A=1, B=2, C=6**

Date su sekvence programa koje izvršavaju ova 3 procesora, respektivno:

	CPU1:	CPU2:	CPU3:
step1:	mov R2, 3	mov R5, 4	mov R3, 6
step2:	load A	load A	load C
step3:	add R1, R2	load B	div R1, R3
step4:	store A	mul R1, R5	store C
step5:	mov R2, R6	store B	add R5, R3
step6:	load B	load A	mul R1, R5

Slika 7.4 Sekvence izvršavanja programa [Izvor: Autor]

Da li se memorijske lokacije nalaze u kešu i kolike su njihove vrednosti u kešu?
REŠENJE:U step 2, dešavaju se load instrukcije koje dovode memorijske reči A, B i C zajedno sa ostatkom linije u keš memoriju CPU1, CPU2, CPU3
respektivno. Nakon step 2 i 3 imamo sledeću situaciju:

Nakon step2 imamo sledeću situaciju:

	CPU1	CPU2	CPU3
step2:	load A	load A	load C
memorija	A=1	B=2	C=6
CPU1 keš	cached(A)=1		
CPU2 keš	cached(A)=1		
CPU3 keš			cached(C)=2

Zapažamo, reč A nalazi se u 2 CPU keša.

Slika 7.5 Nakon koraka 2. [Izvor: Autor]

U step3 imamo još jednu load operaciju:

	CPU1	CPU2	CPU3
step3:	add R1, R2	load B	div R1, R3

Nakon step3 imamo sledeću situaciju:

memorija	A=1	B=2	C=6
CPU1 keš	cached(A)=1		
CPU2 keš	cached(A)=1	cached(B)=2	
CPU3 keš			cached(C)=6

Slika 7.6 Nakon koraka 3 [Izvor: Autor]

U step 4, dešavaju se store operacije koje prati lokalni keš hit i remote cache miss, tako da nema invalidacije. Sve store operacije su local write hit, upis ide i u keš i u memoriju, simultano.

memorija	A=4	B=8	C=1
CPU1 keš	cached(A)=4		
CPU2 keš		cached(B)=8	
CPU3 keš			cached(C)=1

Slika 7.7 Stanje keša nakon koraka 4 [Izvor: Autor]

Poglavlje 8

Domaći zadatak

DOMAĆI ZADATAK #13

Izrada domaćeg zadatka #13 okvirno traje 30 minuta.

ZADATAK (30 minuta): U sistemu na slici 1, primenjuje se šema pisati kroz **WT** (write through) keš koherentni protokol.
Sistem ima sledeće karakteristike:

- **3 procesora sa sopstvenim keš-om i snooping keš koherentnim protokolom**
- **CPU su 32bitni**
- **WT (write through) keš koherentni protokol**
- **deljiva RAM memorija raspoloživa preko zajedničke magistrale**
- **R1 ima ulogu akumulatora**
- **početne vrednosti memorijskih reči A, B, C**

A ima vrednost broj_indeksa % 6 + 3

B ima vrednost broj_indeksa % 5 + 1

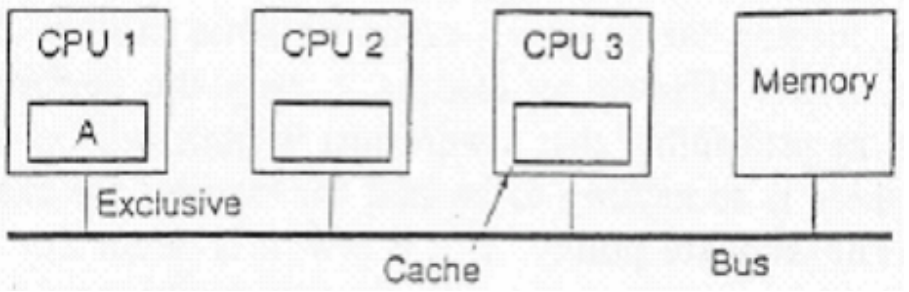
C ima vrednost broj_indeksa % 7 + 3

PRIMER: Za indeks br: 5435;

A=8 (%6=5 + 3=8);

B=1 (%5=0 + 1=1);

C=6 (%7=3 + 3=6);



Slika 8.1 Sistem sa WT keš koherentnim protokolom [Izvor: Autor]

	CPU1:	CPU2:	CPU3:
step1:	mov R2, 3	mov R5, 4	mov R3, 6
step2:	load A	load A	load C
step3:	add R1, R2	load B	div R1, R3
step4:	store A	mul R1, R5	store C
step5:	mov R2, R6	store B	add R5, R3
step6:	load B	load A	mul R1, R5

Slika 8.2 Sekvence izvršavanja programa [Izvor: Autor]

a) Kolika je vrednost memorijskih lokacija A, B i C posle ovih sekvenci? b) Da li se memorijske lokacije nalaze u kešu, i kolike su njihove vrednosti u kešu?

Poglavlje 9

Zaključak

ZAKLJUČAK

Rezime lekcije #13 - Paralelne arhitekture

U lekciji #13 najpre je bilo reči o Flinovoj taksonomiji arhitektura.

Sve je teže ubrzavati računare samo povećavanjem radnog takta jer se tome suprotstavlja problem sa odvođenjem oslobođene toplote a javljaju se i drugi problemi. Zbog toga se projektanti u cilju ubrzavanja računara, sve više okreću ka paralelnom radu.

Paralelizam se može uvesti na više nivoa od sasvim niskog na kome se procesorski elementi međusobno vrlo tesno povezani do veoma visokog na kome su veze između njih veoma labave.

Jedan oblik takvog paralelizma je paralelizam na nivou instrukcija, gde jedna instrukcija ili niz instrukcija započinju više operacija koje mogu paralelno izvršavati različite funkcionalne jedinice. Drugi oblik paralelizma na čipu je rad sa više programskih niti. Treći oblik je multi procesor na jednom čipu, gde se na čip smeštaju dva ili više procesorskih jezgara koje rade paralelno.

Na sledećem višem nivou su koprocesori, najčešće smešteni na dodatne lokacije. Osnovnom sistemu obezbeđuju dodatne procesorske kapacitete za rad u određenim specijalizovanim područjima, kao što su mrežni protokoli ili multimedija.

Multi računari su sistemi sa velikim brojem procesora koji ne dele zajedničku memoriju, već svaki procesor ima sopstvenu memoriju, a komunikacija među njima se obavlja prosleđivanjem poruka.

Na kraju, na najvišem nivou se grid i klaud računari. to su sistemi u kojima su cele organizacije povezene preko interneta da bi udruživale računarsku snagu, podatke i druge resurse.

Literatura:

1. A. Tanenbaum, Structured Computer Organization, Chapter 08, pp. 553 – 652