UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Gregor Zebec

Koper, 2011

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

HIPERNITNOST V PROCESORSKI ARHITEKTURI

Gregor Zebec

Koper, december 2011

Mentor: doc. dr. Peter Korošec

POVZETEK

Procesor je najpomembnejši del računalnika, ki skrbi za izračunavanje, procesiranje podatkov in za nadzor ter upravljanje drugih enot računalnika. Ker lahko s pomočjo možganov pomnimo informacije, jih obdelujemo, lahko procesor primerjamo s človeškimi možgani, saj je računalnik brez procesorja popolnoma neuporaben, prav tako kot človek brez možganov. Procesor je zapleteno integrirano vezje. Od njega je odvisna sleherna komponenta v računalniku. Največkrat je hitrost procesorja označena v megahercih (MHz) ali gigahercih (GHz). Frekvenca procesorja določa hitrost procesorja, ne pa tudi njegove zmogljivosti. Le-ta je primarno odvisna od tega, kako so strokovnjaki zasnovali procesor. Procesor je v arhitekturnem smislu zelo zapletena enota računalniškega sistema in prav zaradi zapletenosti njegove izdelave je na svetu zelo malo izdelovalcev čipov (Intel, AMD …).

V svoji zaključni nalogi sem se posvetil Intelovim procesorjem oziroma tehnologiji za omogočanje zmogljivejšega in učinkovitejšega delovanja procesorjev. Omenjeni tehnologiji pravimo hipernitnost. V zaključni nalogi je ta tehnologija predstavljena skozi različna poglavja, ki zajemajo delovanje tehnologije, spremembe v procesorski arhitekturi, tehnike optimizacije za boljše delovanje aplikacij in podporo operacijskim sistemom.

ABSTRACT

A processor can be compared to the human brain because a computer without a processor is completely useless in the same way as a human being would be without the brain. A processor is a complicated integrated circuit and every single component in a computer depends on it. The speed of the processor is measured in megahertz (MHz) or gigahertz (GHz). Although the frequency of a processor defines its speed it does not define its capacity. The capacity of a computer depends on the way the experts design the processor. As an architectural structure it is a very complicated piece of a computer system and due to this fact there are few manufacturers of these chips in the world (Intel, AMD …).

In the thensis the focus is placed on Intel processors and the whole technology to improve the processors capacity and efficiency. This technology is known as hyper-threading. The HT technology is presented in the thesis through different chapters covering the way this technology works, changes that were made in the processor architecture, methods used to optimize the functioning of applications and the operating systems supports.

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

SEZNAM KRATIC

ACPI – napredni vmesnik za konfiguracijo in porabo

APIC – napredni programabilni krmilnik prekinitev

CMP – večprocesiranje na čipu

CPE – centralno procesna enota

DRAM – dinamični bralno-pisalni pomnilnik

DTLB – podatkovni preslikovalni medpomnilnik

FIFO – prvi noter, prvi ven

HT – hipernitnost

IP – kazalec na naslednji ukaz

ITLB – ukazni preslikovalni medpomnilnik

MADT – opisna tabela naprednega programabilnega krmilnika prekinitev

MT – večopravilni način

RAT – tabela dodatnih imen registrov

ROM – bralni pomnilnik

SMP – hkratno večprocesiranje

SMT – hkratna večnitnost

SRAM – statični bralno-pisalni pomnilnik

ST – enoopravilni način

TLP – vzporednost na nivoju niti

1 UVOD

Arhitektura vsakega računalnika je v največji meri določena s številom in vrsto ukazov, vendar je neko arhitekturo mogoče uresničiti na veliko načinov. Tu se ponuja analogija z gradbeništvom, kjer arhitektura določa obliko in izgled zgradb. Njihova kvaliteta in trajnost pa sta odvisni od zidarskih in drugih del, ki navzven običajno niso vidna. To »zidarstvo« je pri računalnikih realizacija in ta na koncu določa zmogljivost računalnika. Zmogljivost je pri vsakem Von Neumannovem računalniku[3] določena predvsem z zmogljivostjo enote, ki izvršuje ukaze in v veliki meri določa delovanje celotnega računalnika. Zaradi njene centralne vloge to enoto označujemo kot centralno procesno enoto ali CPE (angl. central processing unit – CPU). Med najpomembnejše proizvajalce procesorjev spadata AMD in Intel, vendar sem se v tej nalogi osredotočil le na Intelove procesorje. Procesor je osrednja računalniška enota, ki skrbi za izvajanje računanja nad podatki. Zgrajen je večinoma iz silicija, njegovi ovoji pa so se skozi zgodovino zelo spreminjali (keramika, aluminij)[4]. Procesorji se delijo po številu tranzistorjev, jeder, niti, po tipu podnožja in velikosti predpomnilnika. CPE je računski del procesorja, ki je prav tako narejen iz silicija in skupaj s tranzistorji, ki jih računalniški sistem uporablja, predstavlja glavnega krivca za proizvodnjo toplote. Prav zaradi teh razlogov in želje po višji stopnji procesorske izkoriščenosti se je podjetje Intel odločilo za izgradnjo nove tehnologije, imenovane hipernitnost (angl. hyperthreading)[14], ki predstavlja glavno temo te naloge. Hipernitnost je tehnologija, ki vsako jedro fizičnega procesorja razdeli na dva logična procesorja, med katerima operacijski sistem porazdeljuje procese. Do tukaj je tehnologija videti transparentna, v resnici pa hipernitnost zahteva podvajanje določenih delov procesorja.

Proces je lahko v petih različnih stanjih. Eno izmed stanj predstavlja blokirano stanje, kar pomeni, da se proces ne izvaja, temveč čaka na izvedbo nekega dogodka, ki predstavlja pogoj za nadaljevanje izvajanja. Zato ima v takih primerih procesor proste cikle, ki jih lahko s pomočjo drugega logičnega procesorja ta tehnologija izkoristi ter tako med neizvajanjem prvega procesa izvaja kakšen drugi proces na drugem logičnem procesorju. Takšno izvajanje pomeni višjo oziroma boljšo procesorsko izkoriščenost, saj v primeru neizvajanja procesa le ta ne čaka na delo, temveč prične z izvajanje drugega. Pohitritev, ki je v teoriji s to tehnologijo pridobljena, znaša približno 30 %. Glede na to, da dodatni registri zasedejo le 5 % celotne površine procesorja, je pohitritev za 30 odstotkov v primerjavi z dodatnimi stroški zaradi spreminjanja arhitekture procesorja visoka. Res pa je, da se lahko z uporabo te

tehnologije zaradi tekmovanja niti za skupne vire zgodi padec v učinkovitosti računalniškega sistema. Tekmovanje za skupne vire je skoraj nemogoče preprečiti, vendar se tekmovanju lahko z učinkovitimi tehnikami optimizacije poskušamo v čim večji meri izogniti. Prav preprečevanje tekmovanja za skupnimi viri in s tem bolj učinkovito delovanje računalniškega sistema predstavlja enega izmed največjih problemov, s katerimi se ta tehnologija spopada.

1.1 Opredelitev problema

Zaradi velikih potreb in zahtev uporabnikov so želje po boljših, učinkovitejših oziroma zmogljivejših procesorjih vse večje. Ker s temi potrebami uporabnikov naraščajo tudi stroški izdelave zmogljivejših procesorjev, so s pomočjo hipernitnosti skušali te stroške minimizirati ter hkrati povečati zmogljivost procesorjev. Poleg samega dodajanja gradnikov bi tu rad izpostavil, da hipernitnost za delovanje uporablja logične procesorje. Vsak fizični procesor, ki seveda to tehnologijo podpira, je sestavljen iz več tako imenovanih logičnih procesorjev, za katere velja, da uporabljajo razdeljive, podvojene in skupne vire[28]. Učinkovita raba skupnih virov predstavlja najmočnejše »orožje« te tehnologije. Ker pri novostih pogosto pride do novih problemov, tudi ta tehnologija ne predstavlja nobene izjeme. Problemov, ki se z uporabo te tehnologije pojavijo, je kar nekaj, vendar so v primerjavi z izboljšavo, ki jo ta tehnologija omogoča, zanemarljivi.

1.2 Cilj naloge

Za zaključno nalogo sem si izbral naslov »Hipernitnost v procesorski arhitekturi«, saj pred tem nisem bil s to tehnologijo dobro seznanjen. Ker hipernitnost predstavlja dokaj novo, vendar kljub temu zelo pogosto tehnologijo, ki jo dandanes uporablja večina Intelovih procesorjev, je cilj naloge predstaviti njeno delovanje, arhitekturo, podporo operacijskih sistemov ter uporabo optimizacijskih tehnik za učinkovitejše delovanje aplikacij na procesorjih, ki hipernitnost podpirajo.

1.3 Struktura naloge

Zaključna naloga je sestavljena iz devetih poglavij.
V prvem poglavju »Uvod« je predstavljen problem in opredeljen cilj diplomske naloge ter opisana struktura naloge.

V drugem poglavju »Zgodovina« je na kratko predstavljena zgodovina Intelovega proizvajanja procesorjev skozi različna obdobja. Iz vsakega obdobja razvoja so na kratko predstavljeni in opisani najbolj pomembni procesorji.

V tretjem poglavju »Enoprocesorski sistemi« so opisane določene značilnosti in delovanje enoprocesorskih sistemov ter metode, ki jih ti sistemi uporabljajo za učinkovitejše delovanje.

V četrtem poglavju »Večprocesorski sistemi« sem na kratko predstavil nekaj večprocesorskih sistemov ter se posvetil glavni temi diplomske naloge, ki je hipernitnost v procesorski arhitekturi.

V petem poglavju »Arhitektura« sem se osredotočil samo na procesorsko arhitekturo procesorja s hipernitnostjo. Opisal in predstavil sem različne arhitekturne enote, kot so npr. vhodni podsistem, enota z neurejenim izvrševanjem, pomnilniški podsistem itd.

V šestem poglavju »Optimizacijske tehnike« so bolj podrobno opisane in predstavljene nekatere optimizacijske tehnike, ki omogočajo učinkovitejše izkoriščanje omenjene tehnologije.

V sedmem poglavju »Operacijski sistemi in aplikacije« sem se osredotočil predvsem na operacijske sisteme Windows. Tukaj sem opisal, kateri operacijski sistemi podpirajo hipernitnost in kateri je ne podpirajo ter kakšne funkcije nam operacijski sistem za delo s hipernitnostjo omogoča. Poleg tega sem se na kratko dotaknil zgradbe in delovanja aplikacij pod vplivom te tehnologije.

V osmem poglavju »Zaključek« sem na kratko povzel vsebino zaključne naloge ter podal mnenje o tehnologiji.

V devetem poglavju »Literatura« so našteti vsi viri, ki so mi bili v pomoč pri nastajanju zaključne naloge.

2 ZGODOVINA

Intel je največje ameriško tehnološko podjetje, ki sta ga, z odhodom iz Fairchild Semiconductor, leta 1968 ustanovila Gordon E. Moore (kemični inženir in fizik) in Robert Noyce (fizik in so-izumitelj integriranega vezja). Kasneje se jima je pridružil sedanji strateški vodja Andy Grove (kemični inženir), ki je med leti 1980 in 1990 tudi sam vodil podjetje. To podjetje je konec 90. let postalo tudi eno izmed največjih in najmočnejših podjetij na svetu. Moč in pomembno vlogo si je Intel pridobival skozi različne faze. V času razvoja se je Intel v zgodovino zapisal s proizvajanjem polprevodnikov in pomnilniških naprav. Intelovo poslovanje se je v 70. letih nekoliko povečalo z razširjanjem in povečevanjem svojih proizvodnih procesov in proizvajanjem širšega nabora izdelkov. Kljub vsemu je še vedno prevladovala proizvodnja pomnilniških naprav.

Pomnilniški register (angl. memory register), bralno-pisalni pomnilnik (angl. random access memory), statični bralno-pisalni pomnilnik (angl. static random access memory – SRAM), dinamični bralno-pisalni pomnilnik (angl. dynamic random access memory – DRAM) in bralni pomnilnik (angl. read only memory – ROM) so bile prve pomnilniške naprave, ki jih je podjetje Intel proizvedlo. S proizvajanjem pomnilniških naprav je Intel presegel vso konkurenco in si tako v 70. letih prisvojil vodilno vlogo na trgu. Kljub prevladovanju so Intelovi inženirji Marcian Hoff, Federico Faggin, Stanley Mazor in Masatoshi Shima nadaljevali z izdelovanjem mikroprocesorjev in tako 15. novembra 1971 izdelali prvi Intelov mikroprocesor 4004[6], ki je bil razvit za japonsko podjetje Busicom.

Po izdelanem prvem procesorju se je proizvodnja le-teh nadaljevala in prevzemala vodilno vlogo v Intelovem poslovanju. Povečevanje konkurence v izdelovanju pomnilniških naprav, ki je nastopila v sredini 80. let, je povzročila preusmeritev Intela v mikroprocesorsko proizvodnjo. Povečalo se je število japonskih proizvajalcev polprevodnikov, kar je po letu 1983 pomenilo dramatično zmanjšanje dobičkonosnosti tega trga. Nenadni uspeh IBM osebnih računalnikov in zmanjšana dobičkonosnost trga sta Intel prisilila v preusmeritev v proizvodnjo mikroprocesorjev in k spremembi temeljnih vidikov poslovnega modela. Ta odločitev se je izkazala za pravilno in dobičkonosno, ko je Intel leta 1978 izdelal procesor 8086[6] ter kasneje še njegovo izboljšano različico 8088[6]. Podjetje Intel je začelo pomembno tržno in prodajno akcijo, s katero so želeli pridobiti čim večje število kupcev svojih procesorjev. Akcija se je z ustanovitvijo podjetja IBM PC leta 1981 izkazala za

uspešno, saj so IBM-ovi osebni računalniki vsebovali Intelove procesorje. Leta 1982 je podjetje IBM PC izdelalo in predstavilo svoj prvi osebni računalnik, ki je uporabljal Intelov procesor 80286. Procesor je sestavljala 16-bitna arhitektura in je vseboval 134.000 tranzistorjev. Tri leta kasneje je Intel na trg postavil nov, nekoliko hitrejši in boljši 80386 procesor. Uporabniku osebnega računalnika je omogočal izvajanje več programov hkrati. Ta procesor je imel 32-bitno arhitekturo in vseboval več kot 275.000 tranzistorjev, kar je dvakrat toliko, kot jih je vseboval njegov predhodnik 80286[6]. Proizvodnja mikroprocesorjev se je iz leta v leto izboljševala in povečevala. Tako je leta 1989 Intel predstavil še nekoliko izboljšano različico procesorja 80386[6], procesor 80486[6]. Sestavljalo ga je 1,2 milijona tranzistorjev in lahko je izvajal 40 milijonov ukazov na sekundo (angl. million instructions per second – MIPS)[6]. Pri oblikovanju 80486 mikroprocesorja se je Intel upiral na RISC (angl. reduced instruction-set computing) tehnologijo, ki vsebuje manjši nabor ukazov v primerjavi s CISC (angl. complex instruction set computer) tehnologijo. Procesorji, ki se nanašajo na RISC, uporabljajo enostavnejše ukaze, ki so enako dolgi in od katerih se lahko večina izvede v eni urini periodi[2].

Naslednji pomemben procesor v Intelovi zgodovini je bil Pentium. Izdelan je bil v 90. letih, natančneje leta 1993. Sposoben je bil izvajati 188 milijonov ukazov na sekundo. Sestavljen je bil iz 3,1 milijona tranzistorjev. Njegova hitrost je bila kar petkrat hitrejša od predhodnika 80486[6]. Prihod Pentiuma je za Intel pomenil začetek obdobja oblikovanja čipov in matičnih plošč. Ta sprememba je pripomogla k večji proizvodnji in prodaji osebnih računalnikov s Pentium procesorjem. Velika in uspešna prodaja osebnih računalnikov, nastop multimedije in pojava interneta so številni razlogi, da se je to podjetje usmerilo k izdelovanju močnejših in učinkovitejših procesorjev. Tako je bil leta 1995 izdelan procesor Pentium Pro[6], ki ga je sestavljajo že neverjetnih 5,5 milijonov tranzistorjev. Izvesti je bil sposoben 300 milijonov ukazov na sekundo. V svojo linijo Pentium procesorjev je Intel dodal novo tehnologijo MMX (angl. multimedia extensions). To tehnologijo sestavlja nov niz ukazov, ki so bili dodani za izboljšanje multimedijskih zmogljivosti osebnih računalnikov. Intel je nadaljeval svojo dolgoletno strategijo oblikovanja zmogljivejših procesorjev za najzahtevnejše uporabnike in tako preostalim uporabnikom omogočil uporabo manj zmogljivih procesorjev po sprejemljivi ceni. Z uvedbo Pentiuma II[6], maja 1997, je Intel sprejel novo strategijo razvoja različnih mikroprocesorjev za vsak nivo računalniškega trga. Procesor je bil sestavljen iz 7,5 milijona tranzistorjev. Dosegal je hitrosti do 300MHz. Prvotno je bil namenjen za namizne računalnike, vendar je bil kmalu prilagojen tudi za uporabo v prenosnih računalnikih. Leta

1998 je Intel izdelal mikroprocesor Pentium II Xeon. Namenjen je bil za uporabo v strežnikih višjega cenovnega razreda in delovnih postajah. Pri podjetju Intel so se istočasno usmerili k izdelovanju omrežnih čipov. Leta 1998 je Intel izdelal 64-bitni mikroprocesor i860[6] z 32-bitno aritmetično logično enoto, ki se je prav tako kot njegov predhodnik, procesor 80486, usmeril k uporabi RISC tehnologije. Zgradba i860 je bila razdeljena na tri dele: seštevalnik, množilnik in grafični procesor. Imel je ločene cevovode za aritmetično-logično enoto, množilnik in seštevalnik, lahko pa je izvajal do tri ukaze na urin cikel. Edinstvena značilnost tega mikroprocesorja je bila, da so bili cevovodi programsko dostopni in so prevajalniku omogočali natančno urejanje strojnih ukazov.

Leta 1999 je Intel vzporedno predstavil dva mikroprocesorja Pentium III in Pentium III Xeon[6]. Pentium III procesorji so na začetku dosegali hitrosti od 450 do 500MHz, vendar so zadnje verzije procesorja Pentium III dosegale hitrosti tudi do 1.400MHz. Ta generacija procesorjev je uporabljala 32-bitno arhitekturo in sestavljalo jo je kar neverjetnih 9,5 milijonov tranzistorjev. Pomemben procesor v Intelovi zgodovini je bil Pentium IV, ki je uporabljal 32-bitno in nekoliko kasneje tudi 64-bitno arhitekturo. V zgodovino se je ta procesor zapisal tudi zaradi tega, ker je to prvi Intelov procesor, ki je podpiral novo tehnologijo imenovano hipernitnost. Slednjo tehnologijo je Intel začel nameščati v svoje najnovejše procesorje, kot so: Atom, Core i3, Core i5, Core i7, Itanium Pentium 4 in Xeon[6].

3 ENOPROCESORSKI SISTEMI

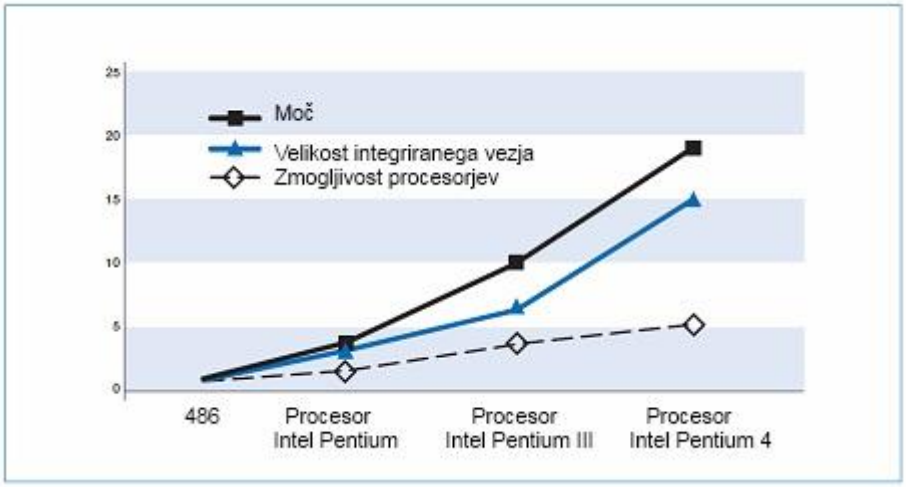
Prizadevanja za izboljševanje učinkovitosti delovanja sistemov z enim procesorjem so bila usmerjena v izdelovanje zmogljivejših procesorjev. Tak pristop do procesorskega projektiranja je pripeljal do tega, da so bili procesorji sposobni izvajati več ukazov na urin cikel, povečevalo pa se je tudi vzporedno izvajanje ukazov (angl. instruction-level paralelism) in pospešilo shranjevanje v predpomnilnik (angl. cache). Višje frekvence urinega cikla in povečevanje števila ukazov, ki se lahko izvedejo v vsaki sekundi, lahko pripomorejo k višji zmogljivosti procesorjev. Ker se število ukazov, izvajajočih v super-cevovodni mikroarhitekturi, povečuje, ravnanje z dogodki, ki cevovod motijo, kot npr. zgrešitve v predpomnilniku (angl. cache misses), prekinitve in napačno napovedovanje skokov (angl.branch mispredictions), zahtevajo dodatno delo.

Vzporedno izvajanje ukazov se nanaša na metode za povečevanje števila ukazov, ki se lahko izvedejo v enem urinem ciklu. Npr. implementacija super-skalarnih procesorjev[8] za vzporedno izvajanje ukazov vsebuje več enot za izvrševanje (angl. execution units). Izvrševalna enota je del procesorja, ki izvaja operacije računalniškega programa. Takšna implementacija super-skalarnih procesorjev omogoča, da se v vsakem urinem ciklu izvede večje število ukazov. Poznamo urejeno (angl. in-order execution) in neurejeno izvajanje (angl. out-of-order execution) programskih ukazov. Značilnost urejenega izvajanja ukazov predstavlja čakanje na pripravljenost podatkov, ki se ne nahajajo v predpomnilniku, kljub temu, da je drugi programski ukaz na voljo za izvajanje. Ker z urejenim izvajanjem računalniški sistem zaradi čakanja na pripravljenost podatkov v veliki meri ne izkorišča računalniški sistem, je nastal nekoliko učinkovitejši način izvajanja programskih ukazov, ki mu pravimo neurejeno izvajanje. Takšen načina izvajanja ukazov omogoča učinkovitejšo rabo ukaznih ciklov[1]. Izvajanje ukazov je odvisno od podatkov oziroma operandov, ki so na voljo. Tako se lahko zgodi, da so na voljo podatki drugega ukaza pred prvim. V primeru uporabe urejenega izvajanja bi moral prvi ukaz čakati, dokler podatki zanj niso pripravljeni, medtem ko se lahko z uporabo neurejenega izvajanja najprej izvede drugi ukaz pred prvim. Kar tak način dovoljuje, je to, da se ukazi izvršujejo v pravilnem ali nepravilnem vrstnem redu (npr. drugi ukaz se lahko izvede pred prvim). Ta prilagodljivost izvajanja ukazov izboljšuje učinkovitost delovanja procesorjev, saj omogoča izvajanje s krajšimi zamudami zaradi čakanj na pripravljenost podatkov, ki jih ukaz potrebuje.

[1] Ukazni cikel sestavlja prevzemanje naslednjega ukaza, dekodiranje ukazov in njihovo izvrševanje [9].

Dostop do glavnega pomnilnika je hitrejši kot dostop do trdega diska, vendar je kljub vsemu počasnejši v primerjavi s hitrostmi in potrebami procesorjev. Ena izmed tehnik zmanjševanja zakasnitev dostopov do glavnega pomnilnika je dodajane hitrega predpomnilnika v »bližino« procesorja. Predpomnilnik nam zagotavlja hiter pomnilniški dostop do podatkov oziroma ukazov, ki jih procesor trenutno potrebuje, oziroma jih bo najverjetneje potreboval. Z dodajanjem predpomnilnikov je pridobitev na hitrosti večja, vendar sta posledično tudi količina oddane toplote in cena višji. Prav zaradi teh dveh razlogov je procesorska arhitektura zasnovana tako, da so predpomnilniki po velikosti manjši ter se nahajajo v bližini procesorskega jedra. V primeru vsebovanja podatkov predpomnilniki zmanjšajo število dostopov do glavnega pomnilnika. Ker je predpomnilnik po velikosti manjši, se zgodi, da leta zahtevanih podatkov ne vsebuje. V tem primeru pravimo, da je prišlo do zgrešitve v predpomnilniku. Takšne zgrešitve v predpomnilniku zahtevajo dostop do glavnega pomnilnika oziroma trdega diska in povzročijo ustavitev in čakanje procesorja na pripravljenost podatkov.

Tehnike za izboljševanje delovanja procesorjev dajejo poudarek velikosti integriranega vezja in procesorski moči. Zaradi omejene vzporednosti v poteku ukazov nobena od tehnik ne omogoča popolne izkoriščenosti super-skalarnih procesorjev. Kot rezultat – podvajanje enot za izvrševanje ne podvoji zmogljivosti procesorja. Zaradi števila izgubljenih procesorskih ciklov v »počasnem« pomnilniškem podsistemu tudi podvajanje urinega cikla ne podvoji zmogljivosti.



Slika 1 Zmogljivost procesorjev v primerjavi z velikostjo integriranega vezja in močjo

Slika 1 Zmogljivost procesorjev v primerjavi z velikostjo integriranega vezja in močjo

prikazuje relativno povečevanje učinkovitosti in cene integriranega vezja ter procesorske moči v zadnjih desetih letih[2]. Ta primerjava predpostavlja, da štiri naštete generacije procesorjev sestavlja ista silicijeva tehnologija in da se izboljševanje hitrosti normalizira na učinkovitost Intel 80486TM[3] procesorja.

3.1 Večopravilnost

V računalništvu večopravilnost predstavlja sposobnost računalniškega sistema, da lahko izvaja več opravil hkrati. Večopravilnost je v nasprotju z enoopravilnostjo, kjer zahtevamo, da se naenkrat lahko izvaja samo en proces. Na enoprocesorskih sistemih večopravilni sistemi ukazov dejansko ne izvajajo sočasno, saj je takšen sistem sestavljen iz enega samega procesorja. Namesto dejanskega vzporednega izvajanja ukazov pa večopravilnost pri enoprocesorskem sistemu pomeni preklapljanje med aktivnimi procesi (angl. context switching)[4]. Ker se preklapljanje med procesi izvaja izredno hitro, uporabniki mislimo, da se procesi izvajajo vzporedno. Takšna implementacija računalniškim sistemom omogoča dobro izkoriščanje časa, ki bi bil v nasprotnem primeru lahko namenjen čakanju na vhodno/izhodno napravo ter na uporabniški vhod, tako da je porabljen za izvajanje drugega opravila, ki vhodne/izhodne naprave ne potrebuje.

Zgodnje večopravilne sisteme so sestavljali nabori programskih aplikacij, ki so med seboj sodelovale in si porazdeljevale čas. Takšen pristop podpira velika večina operacijskih sistemov in ga pogosto srečamo pod imenom sodelovalna večopravilnost (angl. cooperative multitasking). Značilnost sodelovalne večopravilnosti predstavlja dejstvo, da se procesor izvajajočim opravilom ne odvzema, temveč opravila prostovoljno oddajajo procesorski čas drugemu opravilu. Programi, ki uporabljajo takšen način večopravilnosti, morajo biti sposobni prepuščati procesorski čas drugim. Ti, ki tega ne počnejo dovolj pogosto, povzročijo neodzivnost programa, dokler se procesorski čas ne odda drugemu. Prav neodzivnost programov v takšnem sistemu predstavlja največjo slabost, saj lahko v takšnem primeru

[2] Podatki so približni in namenjeni zgolj prikazovanju poteka, ne pa dejanske učinkovitosti.

[3] Čeprav je v tem primeru uporabljena zgodovina Intelovih procesorjev, bi tudi drugi proizvajalci procesorjev v tem obdobju imeli podoben potek.

[4] Menjavanje konteksta je računalniški proces shranjevanja in obnavljanja procesorskega konteksta za nadaljevanje izvajanja po prekinitvi. Lahko gre za menjavanje konteksta v registru *(angl. register context* switch), opravilu *(angl. task context switch),* niti *(angl. thread context switch)* ali v procesu *(angl. process context switch)*.

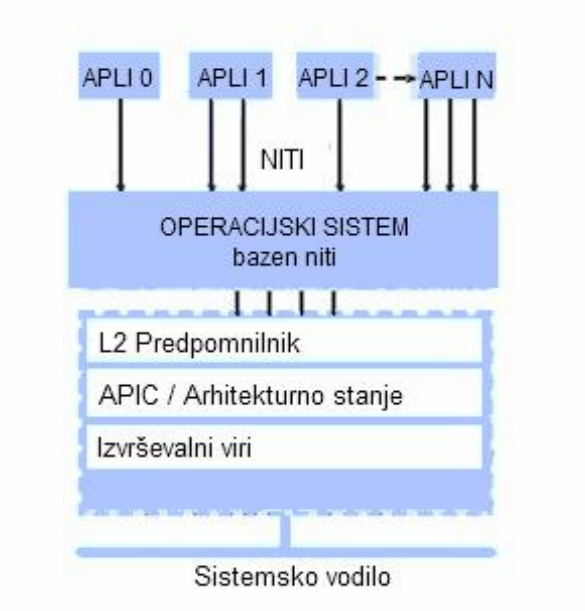
preneha z delovanjem celoten sistem, ker se nobeno opravilo ne more izvajati, dokler program, ki je prenehal z delovanjem, ne prepusti delovanja drugemu opravilu. Prednost sodelovalne večopravilnosti je določanje točke razdelitve in posledično preprečevanje nezaželenih interakcij med procesi ter tako preprečevanje neodzivnosti opravil.

Nekoliko boljši način za reševanje omenjenih težav, ki se pojavijo pri sodelovalni večopravilnosti, predstavlja predkupna večopravilnost (angl. preemptive multitasking). Predkupna večopravilnost je metoda razdeljevanja procesorskega časa med delujočimi programi. Ustvarja časovno deljivo okolje, v katerem procesor delujočim programom dodeljuje časovne rezine različnih dolžin. Dolžine časovnih rezin so odvisne od operacijskega sistema in so lahko enake oziroma prilagojene potrebam posameznih programov. Programom, ki tečejo v ozadju, lahko, na primer, pripada daljša časovna rezina neodvisno od velikosti obremenitve programov, ki tečejo v ospredju in obratno.

Večopravilnost na večprocesorskih sistemih še vedno vključuje preklapljanje med procesi, saj je po številu še vedno več opravil, ki jih je potrebno opraviti, kot pa procesorjev. Zapomniti pa si je potrebno, da lahko vzporedno izvajamo toliko procesov, kot znaša število procesorjev v sistemu.

3.2 Večnitenje

Opazka, da procesorji izgubljajo dragoceni čas pri izvajanju številnih opravil med čakanjem na konec določenega dogodka, je pripeljala do vprašanja, ali lahko med zamudnim čakanjem na določeno opravilo procesorji istočasno izvajajo drugo opravilo. Odgovor na vprašanje predstavlja razširitev ideje večopravilnosti tako, da lahko posamezni programi sočasno izvajajo več opravil. Vsakemu opravilu z drugimi besedami pravimo nit (angl. thread), ki sama ali z drugimi niti tvorijo proces. Nit predstavlja funkcionalno enoto procesa. Med seboj se niti lahko izvajajo neodvisno ena od druge. Vsaka nit ima svoj sklad, stanje, programski števec, registre in svojo identifikacijsko oznako. Z ostalimi nitmi skupnega procesa si deli skupni naslovni prostor, globalne spremenljivke, odprte datoteke in otroke (niti, ki so bile ustvarjene znotraj posamezne niti). Dostopajo lahko do skupnih datotek in dovoljeno jim je dostopanje do vseh podatkov znotraj izvajajočega procesa.



Slika 2 Več aplikacij teče na eno-procesorskem sistemu

Slika 2 prikazuje aplikacijo, ki se izvaja na enoprocesorskem sistemu. Kot je iz slike razvidno, lahko aplikacijo sestavlja ena oziroma več niti, med katerimi je ena nit vedno primarna, ostale pa sekundarne. Sekundarne niti so ustvarjene ob klicih za kreiranje niti s strani operacijskega sistema. Vsaka nit lahko uporablja procesorske vire.

Večnitni operacijski sistemi (angl. multithreading operating systems) omogočajo vzporedno izvajanje niti med čakanjem na vhodno/izhodno napravo oziroma na konec nekega opravila. Za izkoriščenje večnitnosti programi vsebujejo del kode, ki se lahko vzporedno izvaja. Vzporednost izvajanja kode je dosežena z razdeljevanjem dolgih programov na logična področja. Če določena aplikacija izvaja med seboj neodvisne operacije, potem lahko vsaka operacija predstavlja svojo nit. Njihovo izvrševanje predvideva in nadzoruje operacijski sistem. Vzporedno izvajajoča področja lahko opravljajo različna opravila, kot npr. program Microsoft Word med pisanjem omogoča prešteviličevanje dokumenta (prešteviličevanje izvaja ena nit, »tipkanje« pa druga). Na eno-procesorskih sistemih se niti izvajajo zaporedno in ne sočasno. Za navidezno sočasnost poskrbi procesor s hitrim preklapljanjem med nitmi oziroma z menjavanjem konteksta. Takemu načinu z drugimi besedami pravimo vzporedno izvajanje opravil (angl. task parallelism).

Vzporedno izvajanje opravil omogoča »sočasno« izvajanje opravil in posledično programov. Posledico preklapljanja med nitmi predstavlja dodatno delo (angl. overhead). Ker dodatno

delo zahteva določen čas, je ideja o preklapljanju niti pripeljala do nastanka vprašanja glede učinkovitosti izvajanja dveh niti. Vzporedno izvajanje opravil večprocesorskim in večjedrnim sistemom za povečevanje hitrosti dovoljuje vzporedno delovanje niti, kar posledično povzroči boljšo izkoriščenost razpoložljivih virov v sistemu. Operacijski sistemi, ki podpirajo večnitne programe, lahko z vzporednim izvajanjem opravil pripomorejo k izboljšanju učinkovitosti izvajanja programov ter povečanju uporabniške odzivnosti.

Poznamo pa tudi tako imenovano podatkovno vzporednost (angl. data parallelism), kjer se lahko eno opravilo izvaja na več nitih. Pri takšni implementaciji se niti razlikujejo samo v podatkih, ki jih obdelujejo[8].

V realnem svetu veliki programi pogosto uporabljajo več kot dve niti. Vsaki prošnji za zapis v podatkovno bazo, na primer, programska oprema za delo z bazami priredi novo nit. Tako nobena vhodno/izhodna operacija ne preprečuje izvrševanja novih zahtev in preprečuje nastanek tako imenovanega »ozkega grla« (angl. bottleneck) na vodilu[8]. Na nekaterih strežniških sistemih lahko takšen pristop pomeni izvajanje tudi do več tisoč niti.

3.3 Večprocesiranje

Večprocesiranje (angl. multiprocessing) predstavlja način uporabe dveh ali več centralno procesnih enot oziroma zmožnost dodeljevanja opravil med njimi. Tradicionalna Intelova arhitektura večprocesorskih sistemov vsebuje od 2 do 512 procesorjev. V teoriji naj bi se z uporabo dveh procesorjev namesto enega učinkovitost izvajanja podvojila, vendar temu ni tako. V praksi večprocesiranje pripomore k izboljšani učinkovitosti, vendar pod pogoji, da matična plošča, procesor in operacijski sistem večprocesiranje podpirajo. Povečana učinkovitost je opazna v programskih aplikacijah, ki tak način delovanja podpirajo, medtem ko v ostalih aplikacijah temu ni tako. Te aplikacije sestavlja več niti, kar pomeni, da so aplikacije razdeljene v manjše, med seboj neodvisno izvajajoče rutine. Rezultat takšne razdelitve aplikacij je izboljšano delovanje sistema.

Poznani sta dve vrsti večprocesiranja, simetrično večprocesiranje (angl. symmetrical multiprocessing – SMP) in asimetrično večprocesiranje (angl. asymmetrical multiprocessing). Pri asimetričnem načinu je eden oziroma več procesorjev namenjenih opravljanju izključno specifičnih opravil, kot je npr. vodenje operacijskega sistema. Ostali procesorji opravljajo vsa

ostala opravila, kot so npr. uporabniške aplikacije. Izkazalo se je, da asimetrična konfiguracija ne omogoča optimalnega delovanja. Pri takšnem sistemu se lahko zgodi, da so procesorji, ki poganjajo operacijski sistem, polno izkoriščeni, medtem ko vsi ostali procesorji niso izvajali ničesar. Ta nastali problem je pomenil povod za nastanek simetričnega večprocesiranja, pri katerem so obremenitve procesorjev uravnotežene. Simetrija se nanaša na dejstvo, da se lahko niti, ki pripadajo operacijskemu sistemu ali katerikoli drugi aplikaciji, izvajajo na kateremkoli procesorju. Na ta način so vse računalniške obremenitve enakomerno porazdeljene med vse računalniške vire (npr. procesor, trdi disk, pomnilnik itd.). Danes so v uporabi predvsem simetrični večprocesorski sistemi, ki zaradi težavnosti paralelizacije delovne obremenitve ne zagotavljajo *n*-kratno povečanje (*n* nam pove število procesorjev) učinkovitosti sistema.

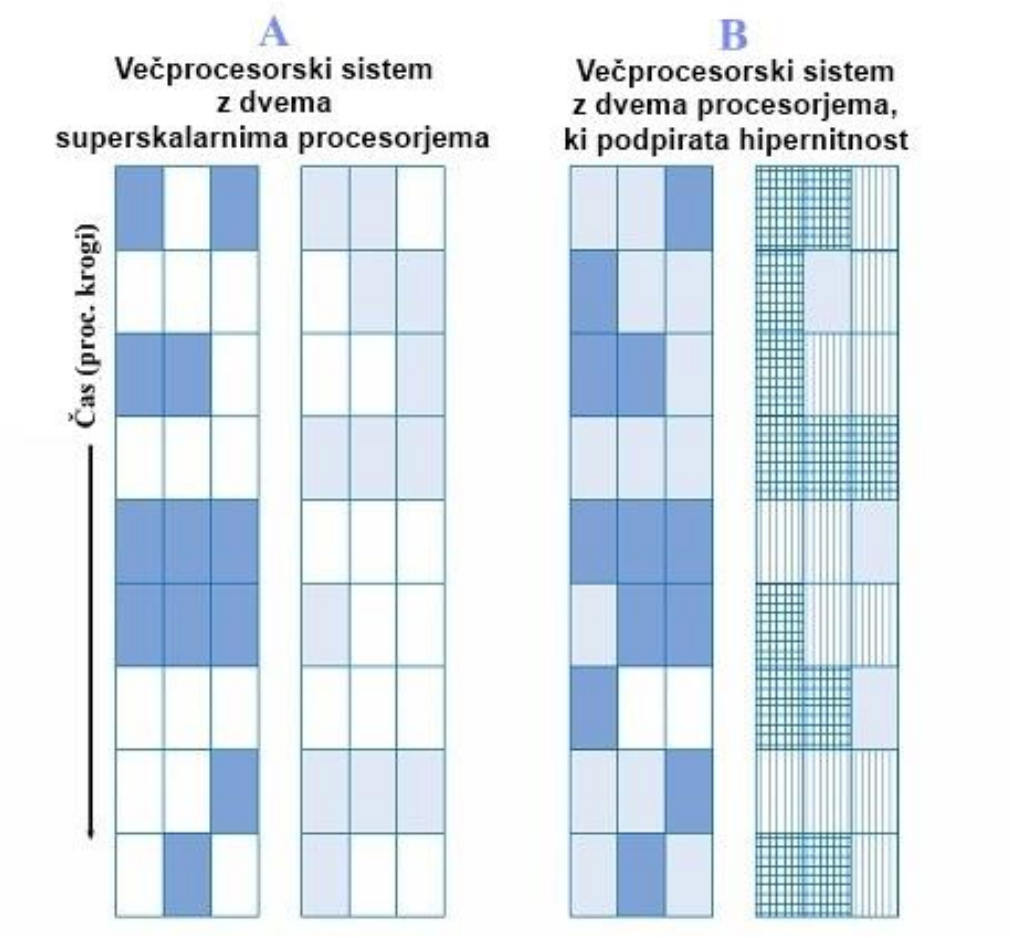
V primeru, da dve niti potrebujeta dostop do istega vira, npr. dostop do trdega diska, dostop do zapisa v zbirki podatkov ali dostop do kateregakoli drugega sistemskega vira, mora ena nit obvezno čakati na drugo. Zamude, ki so posledica takega čakanja, so lahko velike, zato minimizacija le-teh predstavlja pomemben dejavnik načrtovanja strojne in programske opreme. To je tudi največji dejavnik, ki preprečuje popolno izkoriščenost večprocesorskih sistemov, saj je implementacija niti, ki se ne bi potegovale za isti vir, skoraj nemogoča. Drugi pomemben dejavnik, ki preprečuje popolno izkoriščenost sistema, predstavlja sinhronizacija med nitmi. V programu, zasnovanem z nitmi, je medsebojna komunikacija zelo pomembna, vendar tudi zahtevna. Pomembnost medsebojne komunikacije niti si poglejmo na naslednjih primerih. Zamude pri izvajanju niti nastajajo zaradi čakanja niti na pripravljenost podatkov (prva nit pripravlja podatke za drugo, vendar podatki še niso pripravljeni). V primeru, da si dve niti delita območje pomnilnika in je obema pisanje v skupno območje dovoljeno, mora nit, ki je prva pričela s pisanjem, preveriti, ali je prišlo do prepisa podatkov oziroma mora drugi niti preprečiti dostop do kritičnega področja, v tem primeru dostop do podatkov. Dodatno sistemsko delo predstavlja del nitnega upravljanja, za katerega skrbi operacijski sistem oziroma aplikacija. Operacijski sistem, zaradi večjega števila procesorjev, skrbi za njihovo usklajeno delovanje. Rezultat tega predstavlja dejstvo, da se z naraščanjem števila procesorjev zmanjšuje prispevek vsakega procesorja k delovanju celotnega sistema.

4 VEČPROCESORSKI SISTEMI

Tako v visoko kot nizko zmogljivih strežniških trgih se večprocesorski računalniki uporabljajo za doseganje višjih zmogljivosti. Dodajanje procesorjev lahko pripomore k učinkovitejšemu delovanju aplikacij, saj se lahko tako na več procesorjih vzporedno izvaja več niti, ki pripadajo eni ali različnim aplikacijam. Večprocesorski sistemi so v uporabi že nekaj let, zato so programerji že dobro seznanjeni z izkoriščanjem večprocesorskih računalnikov za učinkovitejše delovanje.

Današnje spletne aplikacije so sestavljene iz vzporedno izvajajočih niti oziroma procesov. Takšen način izvajanja lahko bistveno pohitri delovanje sistema. Poleg spletnih so tudi ostale aplikacije zmeraj bolj podvržene vzporednem delovanju. V ta namen so Intelovi inženirji za izboljšanje učinkovitosti glede na število tranzistorjev in porabo energije implementirali vzporednost na nivoju niti (angl. thread-level parallelism – TLP).

V zadnjih letih so bile obravnavane številne tehnike izkoriščanja vzporednega delovanja procesov. Prvo tehniko predstavlja večprocesiranje na čipu (angl. chip multiprocessing – CMP), kjer sta na enem čipu postavljena dva procesorja, ki imata poln nabor izvrševalnih in arhitekturnih virov. Lahko si delita tudi velik predpomnilnik, ki se poleg njiju nahaja na čipu. Kljub vsemu pa je CMP čip bistveno večji v primerjavi z navadnim eno-jedrnim čipom in zato posledično tudi dražji. Druga tehnika z nitnim preklapljanjem omogoča procesorju izvajanje več niti hkrati. Takšen pristop, kjer procesor preklaplja med nitmi, se imenuje supernitnost (angl. time-slice multithreading ali superthreading). Supernitnost sicer lahko zapravlja izvrševalne reže, vendar omogoča učinkovito minimizacijo dolgih časovnih zamud ob dostopih do pomnilnika v primeru zgrešitve v predpomnilniku. Tretja tehnika izkoriščanja vzporednega delovanja procesov se imenuje dogodkovna-preklopna večnitnost (angl. switch-on-event multithreading), kjer se preklapljanja med dogodki zgodijo ob dolgih premorih (npr. zgrešitev v predpomnilniku zahteva dostop do glavnega pomnilnika). Vendar obe tehniki, tako supernitnost kot dogodkovna-preklopna večnitnost, ne dosegajo optimalnega prekrivanja številnih neučinkovito uporabljenih virov, kot je npr. napačno napovedovanje vejitev itd. Zato je nastala četrta tehnika izkoriščanja vzporednega delovanja procesov, ki se imenuje hkratno večnitenje (angl. simultaneous multithreading – SMT).



Slika 3 (Levo) Večprocesorski sistem z dvema superskalarnima procesorjema. (Desno) – Večprocesorski sistem s tehnologijo hipernitnosti.

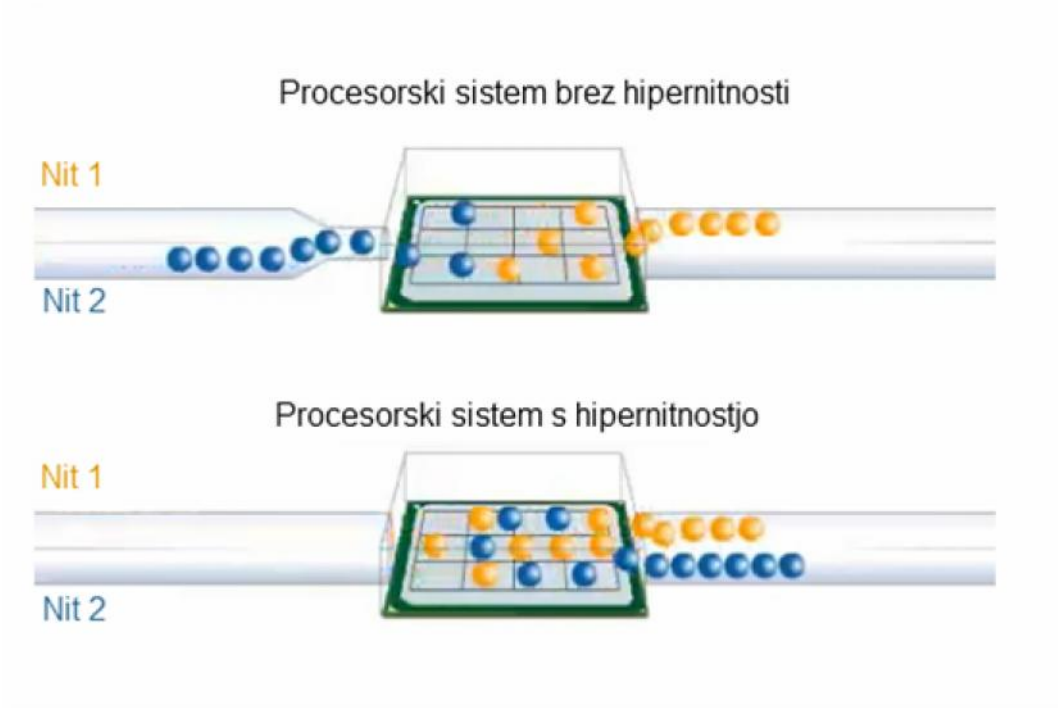
Značilnost SMT sistema je omogočanje vzporednega izvajanja niti brez nitnega preklapljanja. Takšen način posledično omogoča shranjevanje kontekstov v različne registre na čipu. Ukazi različnih kontekstov se tako izvajajo vzporedno, kar prikazuje slika 3. Leva stran slike prikazuje večprocesorski sistem z dvema skalarnima procesorjema, kjer temno modra in svetlo modra barva predstavljata niti, ki ju procesorja izvajata. Niti, obarvane s temno modro barvo, se izvajajo na prvem procesorju, medtem ko se niti, obarvane s svetlo modro barvo, izvajajo na drugem. Območja obarvana z belo barvo predstavljajo prazen neizkoriščen prostor. V primeru večprocesorskega sistema z dvema procesorjema je število praznih izvajalnih enot veliko. Rešitev na slednji problem predstavlja desna stran slike, ki predstavlja večprocesorski sistem s hipernitnostjo (angl. hyper-threading). Takšen večprocesorski sistem omogoča boljše izkoriščanje izvajalnih enot, saj je število neizkoriščenih izvajalnih enot (obarvanih z belo barvo) veliko manjše kot pri večprocesorskem sistemu, ki ga prikazuje leva

stran slike. Pri večprocesorskem sistemu s hipernitnostjo je prisotnih več procesorjev. Vsakega izmed njih sestavljata dva logična procesorja. Vsak izmed njiju lahko izvaja eno nit. Tako prvi logični procesor prvega fizičnega procesorja izvaja niti obarvane s temno modro barvo. S svetlo modro barvo obarvane niti izvaja drugi logični procesor prvega fizičnega procesorja. Enaka razdelitev velja tudi za drugi fizični procesor, kjer logična procesorja izvajata temno modre, s črtami obarvane in svetlo modre, s črtami obarvane niti.

Iz do sedaj povedanega vidimo, da lahko enoprocesorski sistemi kot večprocesorski sistemi s hipertnitnostjo veliko bolje izkoriščajo enote za izvajanje, kar posledično omogoča učinkovitejše in hitrejše delovanje celotnega sistema. Več o sami hipernitnosti, njeni arhitekturi in njenem delovanju sledi v naslednjih poglavjih.

4.1 Hipernitnost

Povpraševanje po višjih procesorskih zmogljivostih je pripeljalo do potrebe po ponovnem preučevanju tradicionalnih pristopov procesorskega projektiranja. Mikroarhitekturne tehnike za izboljševanje procesorskih zmogljivosti, kot so npr. super cevovodnost, napovedovanje vejitev, super skalarno izvrševanje, neurejeno izvajanje in uporaba predpomnilnika, povečujejo procesorsko kompleksnost, število tranzistorjev in porabo energije. Posledično je delovanje procesorjev hitrejše, vendar izboljšana učinkovitost procesorskega delovanja ni zagotovljena. Za primer vzemimo kodo, ki povzroča veliko število predpomnilniških zgrešitev. Višja procesorska frekvenca povzroča pogostejše zgrešitve v predpomnilniku in ne izboljšuje stopnje procesorske izkoriščenosti. In prav želja po povečevanju stopnje procesorske izkoriščenosti je pripeljala do nastanka nove tehnologija, ki ji pravimo hipernitnost. Le-ta predstavlja obliko hkratnega večnitenja. Ta novo nastala tehnologija predstavlja nadgradnjo supernitnosti in je neke vrste supernitnost brez omejitve, kar pomeni, da morajo vsi ukazi, ki jih vhodni podsistem (angl. front end) izda, pripadati eni niti. Slika 4 prikazuje princip delovanja zgoraj omenjenih tehnologij.



Slika 4 Primerja delovanja med procesorjem, ki hipernitnosti ne podpira, in tistim, ki to tehnologijo podpira.

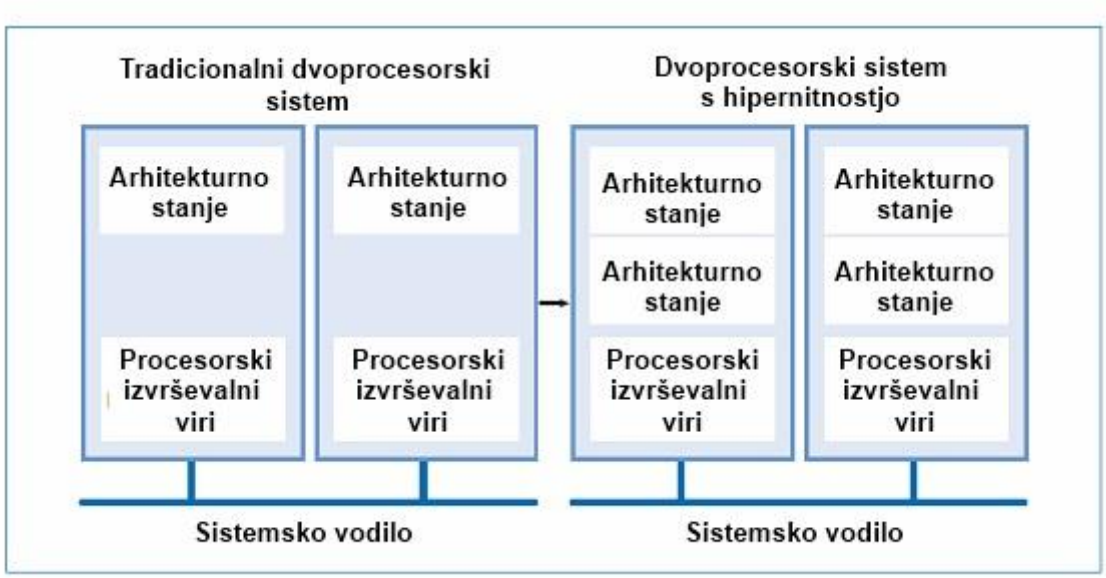
Na njej lahko vidimo procesorski sistem brez in s hipernitnostjo. Razlika v delovanju je očitna, saj lahko procesor brez hipernitnosti hkrati dejansko izvaja eno samo nit, medtem ko lahko sistem z omenjeno tehnologijo dejansko poganja niti vzporedno.

Hipernitnost z omogočanjem vzporednega delovanja programskih niti različnih aplikacij na enem procesorju povečuje procesorsko učinkovitost. Arhitekturno je procesor, ki to tehnologijo podpira, sestavljen iz dveh logičnih procesorjev, od katerih ima vsak svoje podvojeno arhitekturno stanje (angl. architectural state)[5]; glej sliko 5. Vsak logični procesor je lahko ustavljen, prekinjen ali usmerjen v izvajanje točno določene niti, neodvisno od drugih logičnih procesorjev na čipu. Za razliko od tradicionalne dvoprocesorske konfiguracije, ki uporablja ločene fizične procesorje, si logični procesorji delijo izvrševalne vire procesorskega jedra (angl. core processor resources). Ti viri so enota za izvrševanje (angl. execution engine), predpomnilnik (angl. cache), vmesnik sistemskega vodila (angl. system-bus interface) in

[5] Arhitekturno stanje je podvojeno za vsak logični procesor. Sestavljajo ga podatkovni registri *(angl. data register)*, segmentni register *(angl. segment register)*, krmilni register *(angl. control register)*, razhroščevalni register *(angl. debug register)* in večina specifičnih registrov *(angl. model specific registers – MSR)*. Vsak logični procesor ima tudi napredni programabilni krmilnik prekinitev *(angl. advanced programmable interrupt controller – APIC)*.

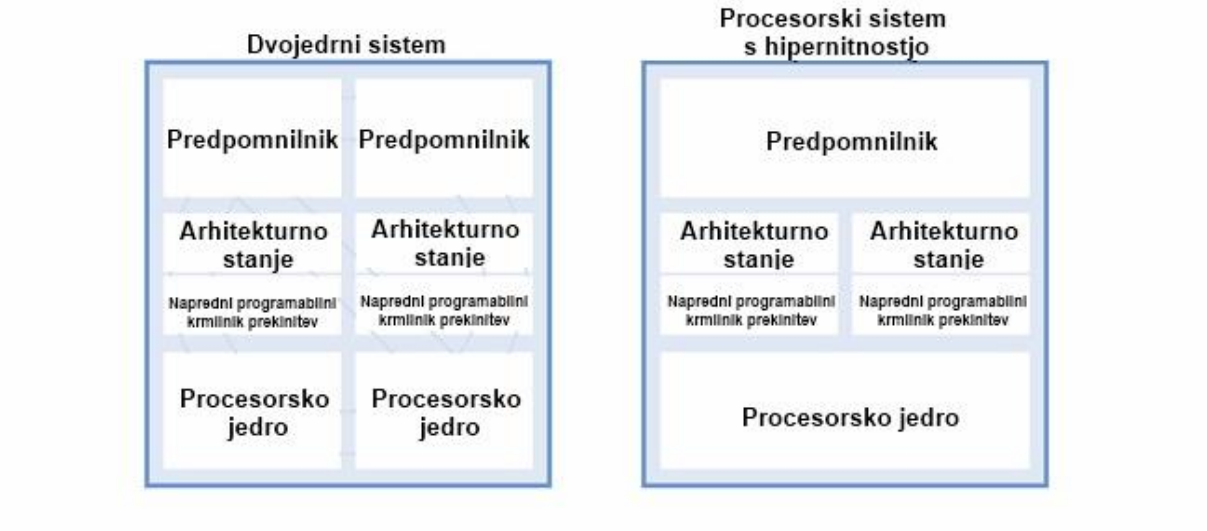
strojna programska oprema (angl. firmware). Z implementacijo dveh logičnih procesorjev na enem čipu je ta tehnologija v operacijske sisteme in visoko zmogljive aplikacije uvedla pojem vzporednega delovanja procesov ali niti. Ta konfiguracija dovoljuje izvajanje niti na vsakem logičnem procesorju. Jedro procesorja ukaze obeh niti vzporedno razpošilja v izvrševanje. Procesorsko jedro z uporabo neurejenega razvrščanja ukazov (angl. out-of-order instruction scheduling) ohranja enote za izvrševanje čim bolj zaposlene v vsakem urinem ciklu.

Hipernitnost v procesorsko arhitekturo uvede številne spremembe. Eno izmed sprememb predstavlja že omenjeno podvojeno arhitekturno stanje procesorja. »Grobo« spremembo arhitekture in primerjavo med arhitekturo dvoprocesorskega sistema z dvoprocesorskim sistemom s hipernitnostjo prikazuje slika 5. Leva stran slike prikazuje tradicionalni večprocesorski sistem z dvema fizičnima procesorjema. Procesorji imajo za vsako podvojeno arhitekturno stanje svoj nabor procesorskih izvrševalnih enot. Desna stran slike prikazuje dvoprocesorski sistem s hipernitnostjo. Iz slike je razvidno, da imajo takšni sistemi podvojeno arhitekturno stanje, vendar ima vsak logični procesor svoj nabor procesorskih izvrševalnih virov. Pri razvrščevanju niti ločeno arhitekturno stanje operacijski sistem obravnava kot dva ločena logična procesorja. Vsak logični procesor se lahko na prekinitve odziva neodvisno od drugih. Medtem ko prvi logični procesor sledi eni programski niti, lahko drugi procesor vzporedno sledi drugi programski niti. Ker si dve niti delita nabor izvrševalnih virov, lahko druga nit uporablja vire tako, kot da bi se izvajala ena sama nit. Rezultat je povečana izraba izvrševalnih virov v vsakem fizičnem procesorju. Hipernitnost predstavlja nov pristop povečevanja učinkovitosti procesorjev namenjenih za strežnike, visoko zmogljive delovne postaje in namizne računalnike.



Slika 5 Tradicionalni večprocesorski sistem na levi strani in Dual Intel Xeon procesor, ki uporablja hipernitnost na desni strani.

Operacijski sistemi in aplikacije na procesorjih s hipernitnostjo sicer delujejo pravilno, vendar optimalna učinkovitost kljub vsemu ni zagotovljena. Hipernitnost ne omogoča večprocesorskega skaliranja (angl. multiprocessor scaling). Večprocesorsko skaliranje predstavlja stopnjo obremenitve pretoka podatkov. Odvisno je od razpoložljivosti ostalih procesorjev, izraženih kot kvocient delovne obremenitve na večprocesorskem računalniku in pretoka na primerljivem eno-procesorskem računalniku[5]. Za aplikacije je značilna uporaba približno 35 % izvrševalnih virov procesorja. Ideja novo nastale tehnologije je uporabljanje in izkoriščanje vsaj 50 % virov. Ta tehnologija pri izvajanju večnitnih operacijskih sistemov in aplikacij v enoprocesorskih računalnikih omogoča 30% pridobitev na učinkovitosti v primerjavi s procesorji, ki te tehnologije ne podpirajo[6]. Pri večprocesorskih računalnikih poraba energije linearno narašča s številom fizičnih procesorjev, medtem ko je povečanje učinkovitosti pri večprocesorskih sistemih močno odvisno od aplikacije do aplikacije[5].



Slika 6 Dva logična procesorja ne zagotavljata enake učinkovitosti kot procesorski sistem s hipernitnostjo.

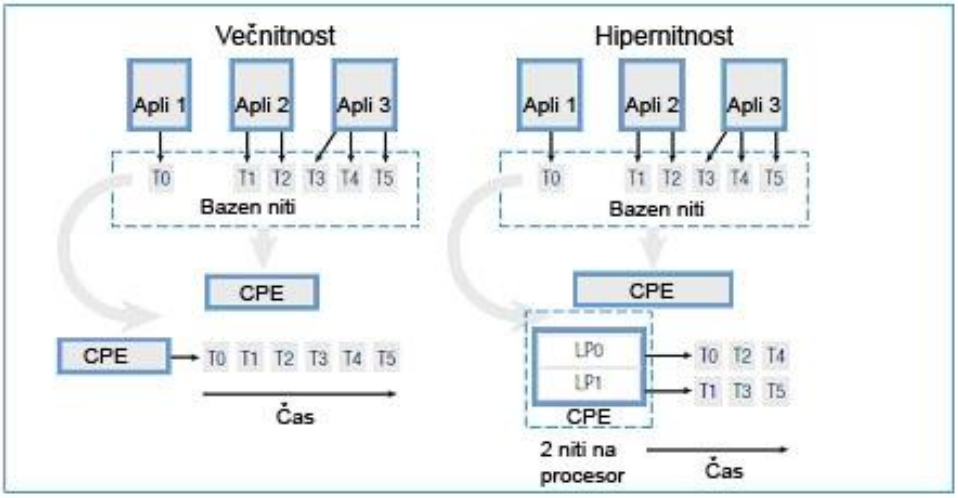
Kot je bilo že povedano, se arhitekturi večprocesorskega sistema z dvema procesorjema in sistema s hipernitnostjo razlikujeta. Razliko v arhitekturi obeh sistemov prikazuje slika 6. Leva stran zgornje slike predstavlja dvojedrni sistem. Desna stran slike predstavlja računalniški sistem s hipernitnostjo. Iz slike je razvidno, da se razliki med obema sistemoma pojavljata v procesorskem jedru in predpomnilniku. Takšen sistem je sestavljen iz logičnih procesorjev, od katerih vsak:

- ima svojo arhitekturno stanje,
- lahko programsko kodo neodvisno od drugega logičnega procesorja izvaja vzporedno,
- je lahko neodvisno prekinjen ali ustavljen.

Logična procesorja si med seboj delita:
- enoto za izvrševanje,
- predpomnilnik,
- strojno programsko opremo in
- vmesnik sistemskega vodila.

Skoraj vsi sodobni operacijski sistemi razdelijo svojo delovno obremenitev na procese ali niti, ki se za delovanje na procesorju lahko neodvisno razporejajo in razpošiljajo. Takšno razdelitev je mogoče najti v številnih visoko zmogljivih aplikacijah, kot so npr. orodja za delo s podatkovnimi bazami, znanstveni računalniški programi, programi za multimedijo itd. Zaradi večje procesorske moči je večina sodobnih operacijskih sistemov za delovanje v

večprocesorskem okolju zasnovana tako, da se lahko procesi oziroma niti razpošiljajo med procesorji.



Slika 7 Čas, ki ga nit n na enoprocesorskem računalniku potrebuje za izvedbo, je daljši kot čas, ki ga nit n potrebuje na enoprocesorskem računalniku, ki uporablja hipernitnost.

Kot je bilo že omenjeno, lahko sistemi s hipernitnostjo hitreje izvajajo določene aplikacije v primerjavi s sistemi, ki te tehnologije ne podpirajo. Časovno primerjavo med sistemom z večnitnostjo in sistemom s hipernitnostjo prikazuje slika 7. Leva stran slike prikazuje sistem z večnitnostjo, kjer se izvajajo tri aplikacije. Vsako aplikacijo sestavlja ena ali več niti. Tako npr. prvo aplikacijo sestavlja nit 0, drugo niti 1 in 2 ter zadnjo niti 3, 4 in 5. Ker sistem, ki ga prikazuje leva stran slike, vsebuje en procesor, posledično ne omogoča vzporednega izvajanja niti. Tako se vse niti izvajajo zaporedno, zato je čas, ki ga CPE potrebuje za izvedbo aplikacij, nekoliko daljši v primerjavi s sistemom, ki hipernitnost podpira. Desna stran slike prikazuje računalniški sistem s hipernitnostjo. Kot je iz slike razvidno, je procesor razdeljen na dva logična procesorja LP0 in LP1, od katerih lahko vsak izvaja svojo nit. Tako je opaziti, da se lahko nit 0 in nit 1 izvajata sočasno, medtem ko se na sistemu, ki hipernitnosti ne podpira, nit 0 in nit 1 izvajata zaporedno. Razdelitev fizičnega procesorja na dva logična procesorja omogoča vzporedno izvajanje niti, kjer se posamezna nit izvaja na svojem logičnem procesorju.

4.1.1 Implementacija

Hipernitnost temelji na tem, da je v vsakem trenutku samo določen del procesorskih virov namenjen za izvajanje programskih kod. Preostali neuporabljeni viri se uporabljajo za vzporedno izvajanje druge aplikacije oziroma druge niti iste aplikacije. Intel Xeon procesor ustvari dva logična procesorja, ki si med seboj delita različne vire centralno procesne enote. Operacijski sistem in aplikacije posledično zaznajo dva logična procesorja, ki si med seboj lahko razdelita delovno obremenitev. Glede na do sedaj povedano, hipernitnost izgleda kot zelo kompleksna tehnologija, vendar temu ni tako. Strojni opremi ne doda bistvene kompleksnosti, saj samo podvoji nekatere vire ter prvotni procesorski arhitekturi doda nekatere manjše strukture. Za lažje razumevanje vpliva hipernitnosti na mikroarhitekturo in učinkovitost procesorja Pentium 4 Xeon si poglejmo implementacijo te tehnologije.

Eden od ciljev te tehnologije je, da je v primeru obstoja samo ene niti hitrost njenega izvajanja enakovredna hitrosti izvajanja niti na procesorju, ki te tehnologije ne podpira. Zaradi tega ima procesor implementirana dva načina delovanja: eno-opravilen način (angl. single-task – ST) in večopravilen način (angl. multi-task – MT). V eno-opravilnem načinu je aktiven samo en logični procesor, ki lahko uporablja vse razpoložljive vire. Drugi logični procesor je pri eno-opravilnem načinu ustavljen z ukazom HALT. Ob pojavitvi druge niti postane drugi logični procesor aktiven in tako fizični procesor preklopi iz eno-opravilnega načina na večopravilni način. Z aktivacijo neuporabljenega logičnega procesorja razpolaga operacijski sistem. Za predstavitev dveh logičnih procesorjev, tako operacijskemu sistemu kot uporabniku, je procesor Intel Xeon sposoben vzdrževati informacije dveh ločenih in neodvisnih kontekstov niti. To so dosegli z razdelitvijo procesorskih virov na tri skupine: podvojeni viri (angl. replicted resource), razdeljeni viri (angl. partitioned resources) in skupni viri (angl. shared resources). Vse tri skupine virov in njihove pripadnike prikazuje tabela 1.

Tabela 1 Podvojeni, razdeljeni in skupni viri sistema s hipernitnostjo.

Skupine virov	Viri
Podvojeni viri	- Logika preimenovanja registrov (angl. register rename logic)

	- Programski števec (angl. program counter ali instruction pointer) - Ukazni preslikovalni medpomnilnik (angl. instruction translation lookaside buffer) - Napovedovalec s povratnim skladom (angl. return stack predictor) - Ostali arhitekturni registri[6]
Razdeljeni viri	- Medpomnilnik za nalaganje/shranjevanje (angl. load/store buffer) - Medpomnilniki za ponovno urejanje (angl. re-order buffers) - Razne vrste, kot sta npr. vrsta mikrooperacij (angl. uop queue), razvrščevalna vrsta (angl. scheduling queue)
Skupni viri	- Predpomnilnik - Mikroarhitekturni registri - Enota za izvrševanje

4.1.1.1 Podvojeni viri

Za ohranitev dveh popolnoma neodvisnih kontekstov na vsakem logičnem procesorju obstajajo viri, ki morajo biti podvojeni. Med tako imenovane podvojene vire spada tudi programski števec. Programski števec je register, ki hrani naslov trenutno izvajajočega ukaza oziroma naslov naslednjega ukaza za izvajanje. Za poganjanje večjega števila procesov potrebujemo toliko programskih števcev, kot znaša število logičnih procesorjev. Intel Xeon procesor vsebuje dva logična procesorja, zato sta posledično prisotna dva programska števca. Prav tako vsakemu logičnemu procesorju pripada tabela dodatnih imen registrov (angl.

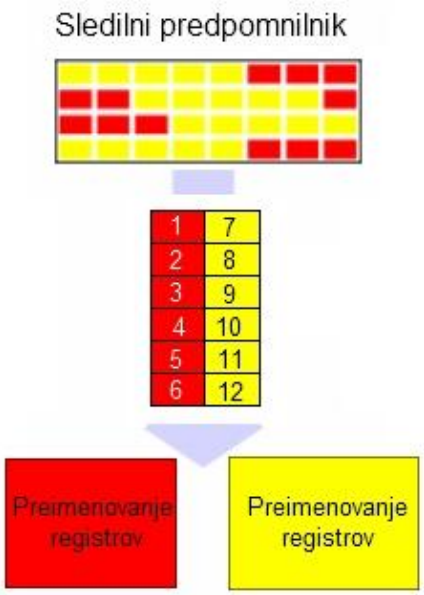
[6] Arhitekturni registri so registri, ki se uporabljajo za razdeljevanje podatkov med pomnilnikom in funkcionalnimi enotami na čipu.

register alias table – RAT)[7]. Tabela skrbi za preslikovanje 8 registrov celih števil in 8 registrov števil s plavajočo vejico v skupno 128 splošno namenskih registrov in 128 registrov plavajoče vejice[7]. Prav tako med podvojene vire spada ukazni preslikovalni medpomnilnik, katerega sestavlja fiksno število zapisov. Uporablja se za hitrejše preslikovanje navideznih naslovov v fizične naslove. Napovedovalec s povratnim skladom se uporablja za učinkovitejšo napovedovanje ciljnih naslovov, ki bodo uporabljeni.

4.1.1.2 Razdeljeni viri

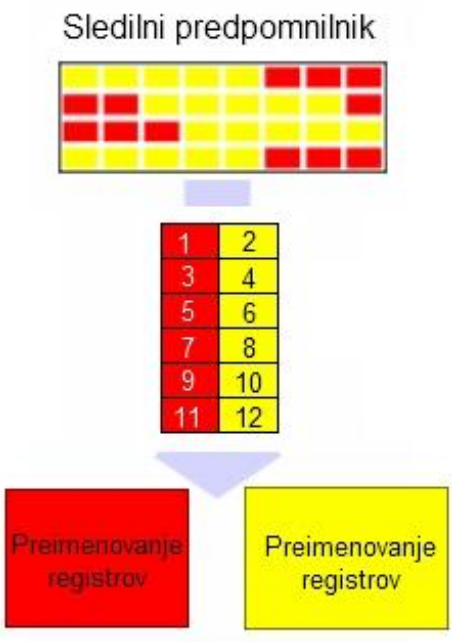
Med razdeljive vire spadajo tisti viri, pri katerih lahko logični procesorji vsebujejo največ polovico zapisov. Prednost razdeljevanja virov predstavlja enostavnost uporabe takih virov. Tak način uporabe virov predstavlja zelo dobro izbiro v primerih, ko je uporaba struktur pogosta in težko napovedljiva. Npr. razdeljevanje je dobra izbira za številne vrste v cevovodu, ki se uporabljajo za medpomnjenje, saj omogoča izogibanje zastojem v cevovodu. Te vrste so polno zasedene večino časa. Poznamo statično (angl. statically partitioned queues) in dinamično razdeljene vrste (angl. dynamically partitioned queues). Statično razdeljena vrsta je razdeljena na dva dela tako, da prva polovica zapisov pripada prvemu logičnemu procesorju, medtem ko druga polovica zapisov pripada drugemu logičnemu procesorju. Primer statično razdeljene vrste prikazuje slika 8. Tukaj lahko vidimo, da se v sledilnem predpomnilniku nahajajo vsi ukazi (obarvani z rdečo in rumeno barvo), ki so nato postavljeni v čakalno vrsto. V njej se nahajajo, dokler ne pridejo na vrsto za izvajanje. Ker je čakalna vrsta, kot je bilo že omenjeno, razdeljena na statičen način, so ukazi posledično razdeljeni na dva dela tako, da prva polovica ukazov pripada prvemu logičnemu procesorju, druga polovica vrste pa drugemu.

[7] Ohranja preslikovanje naslovnega registra, potrebnega za rokovanje pravilnih podatkovnih odvisnosti. Pravilne podatkovne odvisnosti predstavlja rokovanje z novimi fizičnimi registri vsakega novega rezultata, zapisanega v registru. RAT tabela ohranja informacije o lokacijah najnovejših primerkov arhitekturnih registrov. Preslikave med logičnim in fizičnim registrov so zapisane v RAT tabeli, zato da lahko vsak ukaz s poizvedbami indeksiranimi z naslovi virov logičnega registra ugotovi svoje vire fizičnega registra.



Slika 8 Statično razdeljena vrsta.

Pri dinamično razdeljeni vrsti vsakemu logičnemu procesorju pripada natanko polovica zapisov, vendar ne glede na pozicijo v vrsti.



Slika 9 Dinamično razdeljena vrsta.

Tako kot pri statično razdeljeni vrsti imamo tudi v tem primeru vse ukaze spravljene v sledilnem predpomnilniku, vendar, kot je bilo že povedano, vsakemu logičnemu procesorju pripada natanko polovica zapisov, ne glede na pozicijo v vrsti.

Naslednji primer bo obe razdelitvi naredil še nekoliko jasnejši. Če, na primer, obstaja vrsta z 20 zapisi, potem bi logični procesor 0 z uporabo statičnega razdeljevanja vseboval zapise od 0 do 9 in logični procesor 1 bi vseboval zapise od 10 do 19. Pri dinamičnem razdeljevanju je razdelitev nekoliko drugačna. Vsak procesor vsebuje natanko polovico vpisov, torej 10. Vendar le-ti niso razdeljeni statično, kar pomeni, da bi lahko dinamična delitev izgledala takole: logični procesor 0 bi vseboval zapise [1, 2, 3, 5, 7, 9, 10, 12, 14, 15], logični procesor 1 pa zapise [4, 6, 8, 11, 13, 16, 17, 18, 19, 20]. Primer dinamično razdeljene vrste prikazuje slika 9.

Na vsak logični procesor in nit ima dinamično ali statično razdeljevanje zelo podoben učinek, saj za obe razdelitvi velja, da vsakemu logičnemu procesorju po številu pripada natanko polovica zapisov. Kljub vsemu obstaja med tema dvema razdelitvama pomembna razlika. Logika razvrščanja ukazov v datotekah zabeležb[8] (angl. register file) in izvrševalnih enotah pripada skupini skupnih virov. Del Xeon mikroarhitekture se ne zaveda hkratne večnitnosti, zato se razporejevalnik ne zaveda, da koda, s katero razporeja, pripada več nitim. Le-ta gleda na celotno vrsto ukazov kot na posamezni ukazni tok. Razporejevalnik v čakalni vrsti ocenjuje odvisnosti ukazov, primerja potrebe ukazov z enotami za izvrševanje in šele nato razporedi ukaz za izvajanje. Čakalna vrsta se za razliko od razporejevalnika zaveda, katere niti pripadajo katerim logičnim procesorjem. Čakalna vrsta pri Xeon procesorju je dinamično razdeljena in s tem onemogoča monopoliziranje enega logičnega procesorja. Če čakalna vrsta ne bi imela omejenega števila zapisov vsakega logičnega procesorja, bi se dogajalo, da bi ukazi enega logičnega procesorja zapolnili vrsto in s tem preprečili izvajanje ukazov drugega logičnega procesorja.

Drugi razdeljeni vir predstavlja medpomnilnik za ponovno urejanje. Njegova naloga je urejanje tistih ukazov v vrstni red programa, ki so zaključili z izvajanjem v neurejenem vrstnem redu. Medpomnilnik za ponovno urejanje ohranja urejeni vrstni red ukazov. Ukazi so dodani na konec seznama v primeru razpošiljanja ukazov ter premaknjeni na začetek seznama

[8] Datoteka zabeležb je polje procesorskih registrov, ki se nahaja v centralno procesni enoti. Moderno integrirane datoteke zabeležb so pogosto implementirane s hitrim statičnim bralno-pisalnim pomnilnikom.

v primeru izvedbe. V takšnem primeru bodo ukazi izvedeni v enakem vrstnem redu, kot so bili razposlani. Medpomnilnik je implementiran v obliki krožnega medpomnilnika[29].

4.1.1.3 Skupni viri

Skupni viri so tisti viri, ki dajejo tej tehnologiji izjemen pomen. Več kot je skupnih virov med logičnimi procesorji, večja je lahko procesorska učinkovitost. Primarni razred skupnih virov predstavljajo izvrševalne enote, kot so npr. enote za operacije nad celimi števili, enote za operacije nad števili, zapisanimi s plavajočo vejico, in enote za nalaganje/shranjevanje. Te enote se hkratne večnitnosti ne zavedajo, kar z drugimi besedami pomeni, da med seboj ne ločijo niti. Ukaz predstavlja samo ukaz, ne glede na to, kateri niti pripada. Prav tako se hkratne večnitnosti ne zaveda 128 splošno namenskih registrov in 128 registrov za števila, zapisana s plavajočo vejico. Podatki, ki jih vsebujejo, so samo podatki, ne zavedajo pa se, da ti podatki lahko pripadajo več kot eni sami niti.

Skupni viri predstavljajo največji potencial te tehnologije, vendar po drugi strani predstavljajo tudi njeno največjo slabost. Težave se pojavijo, ko ena nit poskuša monopolizirati kritični vir[9]. Takrat lahko en »požrešnež« (angl. resource hog) pripomore k nepravilnemu in izredno počasnemu delovanju. Tako kot sodelovalna večopravilnost je tudi hipernitnost odvisna od monopoliziranja skupnih virov. Na primer, če dve niti, ki izvajata operacije nad števili, zapisanimi s plavajočo vejico, poskušata na istem fizičnem procesorju izvesti dolg niz kompleksnih večcikelnih ukazov, potem je od aktivnosti razvrščevalnika in kompozicije čakalne vrste odvisno, ali lahko ena nit zapolni enoto za operacije nad števili, zapisanimi s plavajočo vejico, medtem ko druga nit stoji in čaka, dokler eden izmed njenih ukazov ne preide iz razvrščevalne v čakalno vrsto. Na procesorjih, ki ne podpirajo hkratnega večnitenja, vsaki niti pripada enakovreden čas izvrševanja zato, ker je po izteku procesorskega časa ta s strani razvrščevalnika dodeljen drugi niti. Podobno velja za supernitnost, kjer prav tako nobena nit ne more zapolniti čakalne vrste. Pri procesorjih, ki podpirajo hkratno večnitnost, pa je zaradi nitnega »bojevanja« za dragocen, vendar omejen izvrševalni vir opazen upad v učinkovitosti procesorjev. Zato lahko trdimo, da za določene primere hipernitnost predstavlja najbolj učinkovito rešitev, za druge pa najslabšo možno[7].

[9] Kritični vir ali deljeni vir je vir, do katerega dostopajo vsi procesorji hkrati.

5 ARHITEKTURA

Hipetnitnost omogoča razdelitev fizičnega procesorja na dva logična procesorja. Kot je bilo že povedano, vsakemu logičnemu procesorju pripada svoje arhitekturno stanje in skupni nabor fizičnih izvrševalnih virov. Iz programskega oziroma iz arhitekturnega vidika to pomeni, da operacijski sistem in uporabniški programi lahko razporejajo procese ali niti med logičnimi procesorji enako, kot poteka razporejanje med fizičnimi procesorji v večprocesorskem sistemu. Iz mikroarhitekturnega vidika to pomeni, da se ukazi logičnih procesorjev na skupnih izvrševalnih virih izvajajo vzporedno. Takšen način izvajanja omogoča učinkovitejšo izrabo procesorskih virov.

Slika 10 Spremembe v procesorski arhitekturi, ki jih hipernitnost vpelje.

Hipernitnost k prvotni procesorski arhitekturi ne doprinese bistvenih sprememb, temveč podvoji logiko za preimenovanje registrov, programski števec, ukazni preslikovalni medpomnilnik, napovedovalca s povratnim skladom in ostale arhitekturne registre. K osnovni procesorski arhitekturi pa doda pretočni medpomnilnik ukazov (angl. instruction streaming buffer), kazalec na naslednji ukaz (angl. next instruction pointer – next IP), sledilni predpomnilnik kazalcev na naslednji ukaz (angl. trace cache next instruction pointer), ukazni preslikovalni medpomnilnik (angl. instruction translation lookaside buffer – ITLB), medpomnilnik, ki polni sledilni predpomnilnik (angl. trace cache fill buffer), in tabelo dodatnih imen registrov. Omenjene spremembe v procesorski arhitekturi, ki jih hipernitnost uvede, prikazuje slika 10.

5.1 Velikost in kompleksnost integriranega vezja

Velika večina tehnik za povečevanje zmogljivosti procesorjev je iz generacije v generacijo postajala vse bolj zapletena in je pogosto povečevala velikost integriranega vezja ter porabo energije. Zaradi omejenega vzporednega delovanja ukazov (vsi ukazi se zaradi medsebojne odvisnosti ne morejo vzporedno izvajati), podvajanje števila enot za izvrševanje posledično ne zagotavlja dvakratnega povečanja procesorske zmogljivosti. Prav tako podvajanje urine hitrosti ne zagotavlja dvakratnega povečanja procesorske zmogljivosti, saj se določeno število procesorskih ciklov izgubi ob napačno napovedovani vejitvi.

Tehnologija hipernitnosti omogoča višjo zmogljivost procesorjev ob minimalnem zvišanju stroškov zaradi povečanja integriranega vezja. Logični procesorji si delijo skoraj vse vire fizičnega procesorja, vključno s predpomnilnikom, enotami za izvrševanje, napovedovanjem vejitev, logiko nadzorovanja in vodila. Povečano integrirano vezje je posledica podvojenega arhitekturnega stanja, dodatne krmilne logike ter nekaj dodatnih procesorskih virov (npr. logika preimenovanja registrov, kazalec na naslednji ukaz itd.). Dodatni tranzistorji, ki so posledica podvojenega arhitekturnega stanja in krmilne logike, izredno malo vplivajo na samo povečanje integriranega vezja.

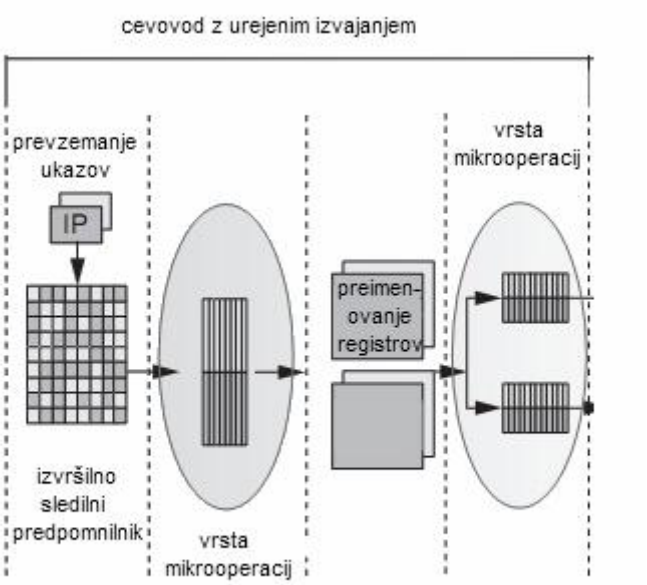
Podvojene strukture, označene na sliki, povečajo kompleksnost in učinkovitost delovanja procesorjev. Tabela dodatnih imen registrov preslika arhitekturne registre v fizične registre. Sledenje arhitekturnih registrov se izvaja neodvisno za vsak logični procesor, zahtevajoč ločeno tabelo za vsak logični procesor. Podvajanje kazalca na naslednji ukaz in s tem posledično povezana krmilna logika omogočata neodvisno sledenje spremembam programa za vsak logični procesor. Obstajata dve vrsti kazalcev na naslednji ukaz: prvi se uporablja za L1 ukazni predpomnilnik in shranjuje dekodirane ukaze. Drugi kazalec na naslednji ukaz se uporablja za logiko dekodiranja ukazov v primeru zgrešitve v izvršilno sledilnem predpomnilniku. Prav tako je podvojen tudi napovedovalec s povratnim skladom. Uporablja se za natančno sledenje parom klic/vrnitev in omogoča njihovo boljše in naprednejše napovedovanje. Pretočni medpomnilnik ukazov in medpomnilnik za polnjenje sledilnega predpomnilnika sta medpomnilnika vhodnega podsistema, ki omogočata učinkovitejše vnaprejšnje prevzemanje ukazov[29]. Prav tako so podvojeni tudi registri naprednega programabilnega krmilnika prekinitev, ki omogočajo neodvisno izvajanje prekinitev na vsakem logičnem procesorju.

Hipernitnost je povzročila spreminjanje številnih osnovnih načrtov za neurejeno izvrševanje enonitnih procesov. Oblikovalci so morali oblikovati številne nove algoritme za omogočanje delitve logike med obema logičnima procesorjema. Prav tako je bilo potrebno pregledati vse ostale algoritme za določevanje prioritet mikrooperacij[10] različnih logičnih procesorjev. Npr. algoritmi, ki so za določanje prioritet mikroopercij uporabljali njihovo starost, so zaradi nejasnosti pri določanju starosti in prioritete ukazov dveh logičnih procesorjev postali veliko bolj zapleteni. Poleg tega je bila pozornost namenjena potencialnemu nastanku »živega objema« (angl. livelock)[11], kjer en logični procesor blokira delovanje ostalih. Algoritmi so bili oblikovani tako, da so že sami po sebi preprečevali pojavitev živega objema, vendar so bili kljub temu za vsak slučaj dodani alternativni algoritmi za reševanje zgoraj omenjenega problema.

5.2 Vhodni podsistem

Naloga vhodnega podsistema cevovoda je dostavljanje ukazov kasnejšim stopnjam cevovoda. Cevovod je razdeljen na cevovod z urejenim izvrševanjem, cevovod z neurejenim izvrševanjem in urejeno zaključevanje mikrooperacij. Pomemben del cevovoda predstavljajo vrsta prevzetih ukazov (angl. fetch queue), vrsta mikrooperacij in vrsta za zaključevanje (angl. retire queue). Vrste omogočajo maksimalno izkoriščanje učinkovitosti procesorja. Njihov cilj je prekrivanje pomnilniških zakasnitev zaradi dostopa do pomnilnika z opravili drugega logičnega procesorja.

[10] Mikooperacija je manjši del ukaza. Več mikrooperacij skupaj predstavlja celoten ukaz.
[11] Predstavlja poseben primer stradanja virov, kar pomeni, da nek proces v nedogled čaka na vire.

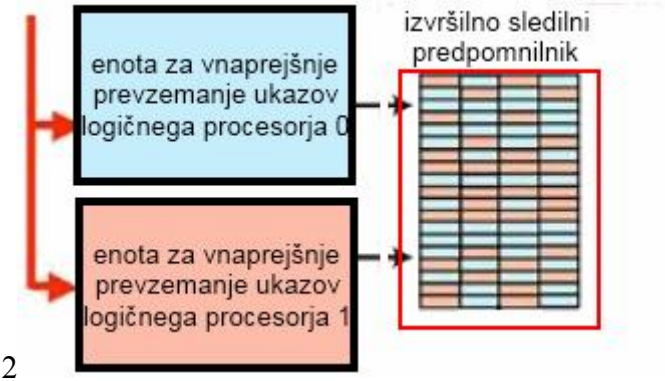


Slika 11 Pregled osnovnega cevovoda z urejenim izvajanjem.

Slika 11 prikazuje osnovni cevovod z urejenim izvrševanjem ukazov. Kot je bilo že povedano, je cevovod sestavljen iz treh delov. Prvi del oziroma cevovod z urejenim izvrševanjem se prične z izvršilno sledilnim predpomnilnikom, v katerem so shranjene mikrooperacije. Izvršilno sledilni predpomnilnik se uporablja namesto L1 predpomnilnika. Drugi korak cevovoda predstavlja prevzemanje mikrooperacij iz sledilnega predpomnilnika (ali mikrokodnega ROM-a v primeru kompleksnih ukazov) za vsako nit posebej in shranjevanje le-teh v vrsto prevzetih mikrooperacij. Trenutno nezaposlene ali ustavljene niti dovoljujejo ostalim uporabo celotne pasovne širine za prevzemanje mikrooperacij. Prevzete mikrooperacije so shranjene v vrsto, ki je razdeljena tako, da vsaki niti pripada polovica zapisov. V tretjem koraku se izvaja preimenovanje registrov ter dodeljevanje le-teh mikrooperacijam s strani dodeljevalca. Pravičnost dodeljevanja je zagotovljena z omejitvijo števila tipov medpomnilnika (medpomnilnik za ponovno urejanje, medpomnilnik za nalaganje in medpomnilnik za shranjevanje) za posamezno nit. V primeru, da obe niti vsebujeta mikrooperacije v vrsti mikrooperacij, dodeljevalec preklaplja med njimi. Če je prva nit prenehala z delovanjem, potem je drugi niti namenjena celotna logika dodeljevanja. Naslednji korak v cevovodu z urejenim izvrševanjem predstavlja preimenovanje arhitekturnih oziroma logičnih registrov s pomočjo tabele dodatnih imen registrov. Omenjena tabela se uporablja za sledenje preimenovanju registrov, saj vsaki niti pripada ločen niz arhitekturnih registrov, ki morajo biti podvojeni za vsako nit.

5.2.1 Izvršilno sledilni predpomnilnik

Izvršilno sledilni predpomnilnik je predpomnilnik, ki logiki neurejenega izvrševanja dostavlja tri mikrooperacije na urin cikel. Vsebuje sledi ukazov, ki so že bili prevzeti in dekodirani. Prevzemanje in izvajanje večine programskih ukazov poteka iz izvršilno sledilnega predpomnilnika. V primeru, da se podatki ne nahajajo v njem, se prevzemanje ukazov preusmeri na L2 predpomnilnik. Je primer set asociativnega predpomnilnika z asociativnostjo 8, kar pomeni, da je predpomnilnik razdeljen na sete. Vsak set pa sestavlja 8 blokov. Vsebuje lahko do 12.000 mikrooperacij. Primer izvršilno sledilnega predpomnilnika v procesorski arhitekturi prikazuje slika 12.



Slika 12

Slika 12 Izvršilno sledilni predpomnilnik.

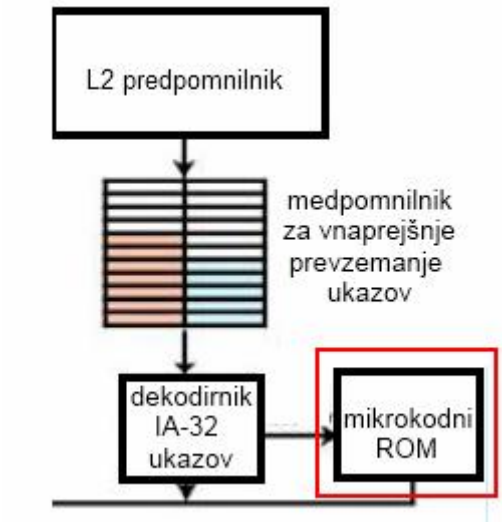
V splošnem so mikrooperacije v predpomnilniku razporejene po skupinah in lahko predstavljajo samostojen blok ali dinamično »sled« ukaza. Sled predstavlja spoj večjega števila blokov. To enoti za prevzemanje ukazov procesorja omogoča prevzemanje številnih blokov brez vejitev v izvrševanju. Sledi so shranjene v izvršilno sledilnem predpomnilniku glede na programski števec prvega ukaza v sledi in napovedovanja vejitev. To omogoča shranjevanje različnih sledilnih poti, ki se pričnejo na istem naslovu. Vsaka pot predstavlja različno vejo rezultatov. V prevzemni fazi cevovoda se zadetek trenutnega programskega števca v izvršilno sledilnem predpomnilniku preverja z napovedovanjem vejitev. V primeru zadetka prevzemanje ukazov ne poteka v navadnem predpomnilniku oziroma glavnem pomnilniku, ampak izvršilno sledilni predpomnilnik nadaljuje s prevzemanjem enot do konca sledilne linije oziroma do napačne napovedi, ki to linijo prekine. Če zadetka ni oziroma v primeru zgrešitve, se prične graditi nova sled. V vsakem urinem ciklu dva programska števca neodvisno sledita poteku izvajanja dveh programskih niti med dostopanjem logičnih

procesorjev do izvršilno sledilnega predpomnilnika. V primeru sočasnega dostopanja logičnih procesorjev do predpomnilnika je dostop obema omogočen v izmeničnih urinih ciklih. V prvem urinem ciklu, na primer, poteka izvrševanje ukazov prvega logičnega procesorja, v naslednjem ciklu se izvršujejo ukazi drugega logičnega procesorja. V primeru zaustavitve logičnega procesorja oziroma njegove nezmožnosti uporabe izvršilno sledilnega predpomnilnika je drugemu logičnemu procesorju v vsakem nadaljnjem urinem ciklu omogočen dostop do predpomnilnika. Izvršilno sledilni predpomnilnik za zamenjavo zapisov uporablja algoritem najdlje neuporabljenih zapisov (angl. least recently used – LRU)[12][10]. Izvršilno sledilni predpomnilnik vsebuje napovedovalca vejitev, ki se uporablja za usmerjanje prevzemanja ukazov v izvršilno sledilnem predpomnilniku. Napovedovalec je manjši od napovedovalca vhodnega podsistema, saj je njegova naloga napovedovanje vejitev podprogramov, ki se trenutno nahajajo v izvršilno sledilnem predpomnilniku[26].

5.2.2 Mikrokodni bralni pomnilnik

Mikrokodni bralni pomnilnik se uporablja zato, ker izvršilno sledilni predpomnilnik ne obravnava kompleksnih ukazov, kot so, na primer, premikanje nizov in rokovanje z napakami ter prekinitvami. Takrat sledilni predpomnilnik mikrokodnemu bralnemu pomnilniku pošilja kazalce na mikrooperacije. Krmilnik mikrokodnega bralnega pomnilnika prevzame potrebne mikrooperacije iz vrste mikrooperacij in nato vhodni podsistem nadaljuje z njihovim prevzemanjem iz izvršilno sledilnega predpomnilnika. Vsakemu izmed dveh logičnih procesorjev pripada svoj kazalec na mikrooperacijo, ki se uporablja za neodvisno nadzorovanje pretoka v primeru, če oba logična procesorja hkrati izvajata kompleksne ukaze. Mikrooperacije, ki pridejo iz mikrokodnega bralnega pomnilnika, so shranjene v pravilno urejeno vrsto, ki omogoča nemoten pretok mikrooperacij v enoto z neurejenim izvrševanjem. Mikrokodni bralni pomnilnik v procesorski arhitekturi prikazuje slika 13.

[12] LRU algoritem zamenjuje tiste zapise, ki najdlje časa niso bili uporabljeni, zato zahteva sledenje časa uporabe določenega zapisa.



Slika 13 Mikrokodni bralni pomnilnik.

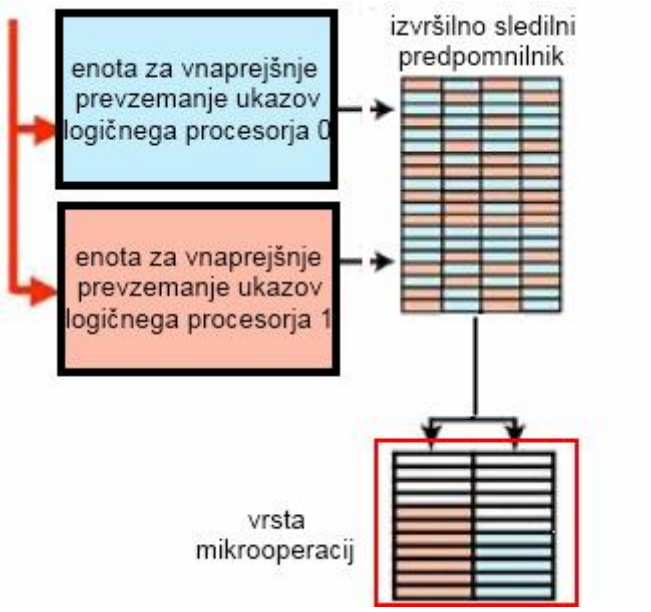
5.2.3 Ukazni preslikovalni medpomnilnik in logika napovedovanja vejitev

Ukazni preslikovalni medpomnilnik (angl. instruction translation lookaside buffer – ITLB) omogoča hitrejše preslikovanje navideznih naslovov v fizične. Vzdržuje najbolj sveže podatke o zadnjih preslikavah. Ukazni preslikovalni medpomnilnik prejme zahtevo za dostavo novih ukazov iz izvršilno sledilnega predpomnilnika. Linearni naslov kazalca na naslednji ukaz preslika v fizični naslov, ki ukaz nato vrne in zahtevo posreduje L2 predpomnilniku. Vrnjeni ukaz se postavi v pretočni medpomnilnik (angl. streaming buffer), ki ukaz zadrži, dokler ni dekodiran. Vsak logični procesor ima svoj ukazni preslikovalni medpomnilnik in nabor programskih števcev, ki se uporabljajo za sledenje prevzemanju ukazov logičnih procesorjev. Logika prevzemanja ukazov je pristojna za pošiljanje zahtev L2 predpomnilniku, ki deluje po načelu: tisti, ki prvi pride, prvi melje (angl. first-come first-served). Vsak logični procesor ima rezerviran vsaj en prostor za zahteve (angl. request slot). Na ta način lahko oba logična procesorja sočasno zahtevata prevzem ukaza. Vsak logični procesor ima svoj 64 zlogov velik pretočni medpomnilnik, ki ukaze ohranja v pripravljenosti za dekodiranje. Logika napovedovanja vejitev (tabela preteklih vejitev in medpomnilnik vejitev) omogoča procesorjem prevzemanje in izvajanje ukazov med izvajanjem trenutnega ukaza. Napovedovalec vejitev vhodnega podsistema lahko vsebuje največ 4.000 zapisov vejitev, ki se uporabljajo za shranjevanje informacij o njih. Obstajata dinamično in statično napovedovanje vejitev. Pri statičnem napovedovanju se odločitev o vejitvi zgodi pred izvedbo ukaza za vnaprejšnje prevzemanje ukazov. Pri dinamičnem napovedovanju pa se

odločitev o vejitvi zgodi med izvajanjem programa v odvisnosti od vejitve, ki se je nazadnje izvedla[27]

5.2.4 Vrsta mikrooperacij

Vrsta, ki jo prikazuje slika 14, loči vhodni podsistem od enote z neurejenim izvrševanjem v poteku cevovoda (angl. pipeline flow). Vrsta mikrooperacij spada med razdeljive vire. Vsakemu logičnemu procesorju pripada polovica zapisov. Takšna razdelitev obema logičnima procesorjema omogoča neodvisno nadaljevanje izvajanja, kljub ustavitvi izvrševanja ali ustavitvi vhodnega podsistema drugega logičnega procesorja, kot je npr. zgrešitev v sledilnem predpomnilniku.

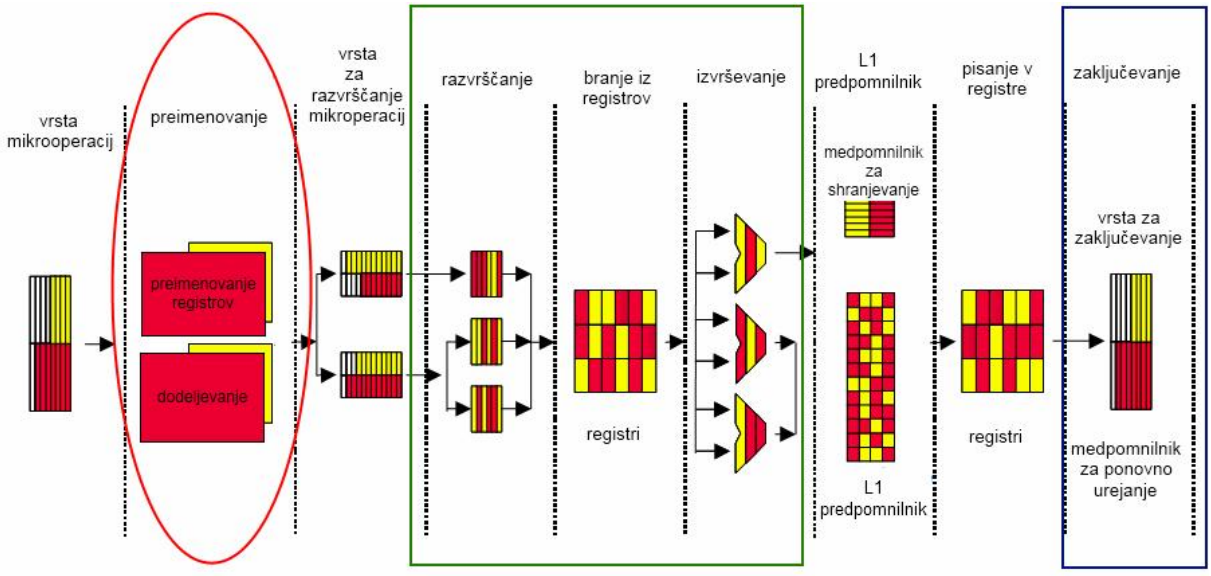


Slika 14 Vrsta mikrooperacij.

5.3 Enota z neurejenim izvrševanjem

Enota z neurejenim izvajanjem sestoji iz funkcij dodeljevalca (angl. allocator), preimenovanja registrov (angl. register rename), razvrščanja ukazov (angl. instruction scheduling) in izvrševalnih enot. Enota skrbi za ponovno urejanje ukazov, katerim omogoča čim hitrejše izvajanje. Procesor poskuša poiskati čim večje število ukazov, ki jih lahko izvede v enem urinem ciklu. Enota z neurejenim izvrševanjem izvaja ukaze kljub nepravilnemu vrstnemu redu (npr. drugi ukaz se izvede pred prvim). S preiskovanjem čim večjega števila ukazov

programa na enkrat, enota z neurejenim izvrševanjem lahko izsledi veliko število med seboj neodvisnih ukazov, ki so pripravljeni za izvajanje. Tako se pospeši izvajanje programa.



Slika 15 Podroben prikaz cevovoda enote z neurejenim izvajanjem.

Cevovod enote z neurejenim izvajanjem je razdeljen na tri osnovne dele, kot to prikazuje slika 15. Prvi del cevovoda, ki je obarvan z rdečo barvo, sestavljata logika za dodeljevanje in logika za preimenovanje registrov. Dodeljevalec mikrooperacijam dodeljuje strojne registre, medtem ko se preimenovanje registrov uporablja za preimenovanje IA-32 registrov v fizične registre. Preimenovanje registrov poteka z uporabo tabele dodatnih imen registrov. Drugi, z zeleno barvo označen del cevovoda, sestoji iz razvrščanja ukazov in enot za izvrševanje. Po postavitvi mikrooperacij v vrsto mikrooperacij razvrščevalnik le-te razporedi v ustrezno čakalno vrsto. Po postavitvi mikrooperacij v čakalno vrsto so mikrooperacije posredovane ustrezni enoti za izvrševanje. Zadnji del cevovoda predstavlja zaključevanje mikrooperacij, ki je na zgornji sliki označeno s temno modro barvo. Po izvedbi so mikrooperacije postavljene v medpomnilnik za ponovno urejanje, ki je razdeljen med obema logičnima procesorjema. Uporabljeni podatki so na podlagi napovedovanja vejitev odstranjeni iz L1 podatkovnega predpomnilnika, tako da se lahko novi podatki zapišejo vanj.

5.3.1 Dodeljevalec

Dodeljevalec jemlje mikrooperacije iz vrste mikrooperacij in jih dodeljuje medpomnilnikom, kot npr. medpomnilnik za ponovno urejanje in medpomnilnik za nalaganje in shranjevanje. Ti medpomnilniki so potrebni za izvajanje mikrooperacij, ker shranjujejo podatke ob preklapljanjih med procesi. Strojni medpomnilniki vključujejo 126 zapisov v medpomnilniku za ponovno urejanje, 128 registrov celih števil, 128 registrov števil v plavajoči vejici, 48 zapisov v medpomnilniku za nalaganje in 24 zapisov v medpomnilniku za shranjevanje. Nekateri od omenjenih medpomnilnikov so razdeljeni tako, da lahko vsak logični procesor uporablja največ polovico zapisov. Torej lahko vsak logični procesor pri taki razdelitvi uporablja največ 63 zapisov v medpomnilniku za ponovno urejanje, 24 zapisov v medpomnilniku za nalaganje in 12 zapisov v medpomnilniku za shranjevanje. Če v vrsti mikrooperacij obstajajo mikrooperacije obeh logičnih procesorjev, potem dodeljevalec za dodeljevanje virov v vsakem urinem ciklu izbira med mikrooperacijami obeh logičnih procesorjev. Če slednji porabi vse svoje potrebne vire, kot so npr. zapisi v medpomnilniku za shranjevanje, mu dodeljevalec pošlje signal za ustavitev in nato nadaljuje z dodeljevanjem virov logičnemu procesorju, ki svojih virov še ni porabil. Če vrsto mikrooperacij polnijo mikrooperacije samo enega logičnega procesorja, potem mu v vsakem urinem ciklu zaradi optimizacije dodeljevanja pasovne širine dodeljevalec poskuša dodeliti vse vire. Z omejevanjem števila virov prej omenjenim medpomnilnikom poskuša dodeljevalec uveljaviti nepristranskost in preprečuje nastop smrtnih objemov[13].

5.3.2 Logika za preimenovanje registrov

Preimenovanje registrov se uporablja za omogočanje večjega števila poti izvrševanja brez konfliktov med različnimi enotami za izvrševanje, ki poskušajo uporabljati iste registre. Logika preimenovanja registrov preimenuje arhitekturne (logične) IA-32 registre v fizične registre. To osmim splošno namenskim IA-32 celoštevilskim registrom omogoča dinamično razširljivost pri uporabi 128 fizičnih registrov. Logika preimenovanja uporablja tabelo dodatnih imen registrov. Ker vsak logični procesor ohranja in sledi svojemu celotnemu arhitekturnemu stanju, ima vsak logični procesor svojo tabelo dodatnih imen registrov. Preimenovanje registrov se izvaja vzporedno z dodeljevanjem in deluje na tistih mikrooperacijah, katerim je dodeljevalec dodelil vire. Ko se dodeljevanje in preimenovanje

[13] Nabor procesov je v smrtnem objemu, če vsak med njimi čaka na dogodek, ki ga lahko sproži le nek drugi proces iz istega nabora.

registrov zaključi, so mikrooperacije postavljene v dve vrsti. Ena izmed njih se uporablja za pomnilniške operacije (shranjevanje in nalaganje) in se imenuje vrsta pomnilniških ukazov, druga, ki se uporablja za splošne operacije, pa se imenuje vrsta splošnih ukazov. Obe vrsti spadata med razdeljive vire, kar pomeni, da sta vrsti razdeljeni med oba logična procesorja.

5.3.3 Razvrščanje ukazov

Razvrščevalniki predstavljajo pomemben del enote z neurejenim izvajanjem. Pet razvrščevalnikov se uporablja za razvrščanje številnih tipov mikrooperacij v različne izvrševalne enote in lahko skupaj na urin cikel razpošljejo do 6 mikrooperacij. Razvrščevalniki določajo, kdaj so mikrooperacije pripravljene za izvajanje. Odločitev temelji na pripravljenosti njihovih vhodnih operandov in na razpoložljivosti izvrševalnih enot. Vrsti pomnilniških ukazov in splošnih ukazov (vsebuje vse preostale ukaze, kot npr. za delo vhodno/izhodno napravo) pošiljata mikrooperacije petim vrstam za razvrščanje. Ukazi so v obe vrsti shranjeni po načelu prvi notri, prvi ven (angl. first in first out – FIFO). Pošiljanje mikrooperacij poteka v vsakem urinem ciklu z izredno hitrim menjavanjem med mikrooperacijami obeh logičnih procesorjev. Vsakemu razvrščevalniku pripada svoja vrsta za razvrščanje z 8 do 12 zapisi. Med temi zapisi je neodvisno od pripadnosti logičnemu procesorju izbrana mikrooperacija, ki je nato posredovana izvrševalni enoti. Izbor mikrooperacij je odvisen od vhodov in razpoložljivosti izvrševalnih virov. Razvrščevalnik lahko, na primer, v enem urinem ciklu razpošlje dve mikrooperaciji vsakega logičnega procesorja. V izmik zastojem in zagotavljanju pravičnosti je število zapisov v vrstah za razvrščanje mikrooperacij logičnega procesorja omejeno.

5.3.4 Izvrševalne enote

Izvrševalne enote so enote, kjer poteka dejansko izvrševanje ukazov. Izvrševalne enote so zasnovane za optimizacijo celotnega delovanja s čim hitrejšim izvajanejm najbolj pogostih primerov ukazov. Izvrševalna enota in pomnilnik sta v veliki meri »neodvisna« od logičnih procesorjev. Po koncu izvajanja mikrooperacije je ta postavljena v medpomnilnik za ponovno urejanje, ki ločuje stopnjo izvrševanja cevovoda od stopnje zaključevanja cevovoda. Medpomnilnik za ponovno urejanje spada med razdeljive vire, saj lahko vsak logični procesor uporablja polovico njegovih zapisov.

5.3.5 Enota za zaključevanje mikrooperacij

Enota za zaključevanje shranjuje rezultate mikrooperacij v uporabniku vidne registre in iz medpomnilnika za ponovno urejanje odstranjuje mikrooperacije, ki so prenehale z izvajanjem in se ne navezujejo na nobeno ostalo mikrooperacijo. Logika zaključevanja mikrooperacij sledi mikrooperacijam obeh logičnih procesorjev, ko so le-te pripravljene za zaključitev. S preklapljanjem med logičnima procesorjema so mikrooperacije logičnih procesorjev odstranjene v vrstnem redu izvajanja programa. Odstranjevanje mikrooperacij obeh logičnih procesorjev poteka izmenično – najprej enega in nato drugega (angl. back and forth). Na urin cikel lahko enota za zaključevanje hkrati obravnava tri mikrooperacije. Če logični procesor ni pripravljen na zaključitev nobene mikrooperacije, je celotna pasovna širina zaključevanja namenjena drugemu logičnemu procesorju. Po zaključitvi se na novo shranjeni podatki zapišejo v L1 podatkovni predpomnilnik in odstranijo iz medpomnilnika za ponovno urejanje. Izbirna logika izbira podatke obeh logičnih procesorjev za njihovo shranjevanje v predpomnilniku.

5.4 Pomnilniški podsistem

Pomnilniški podsistem vključuje podatkovni preslikovalni predpomnilnik, L1 podatkovni predpomnilnik, L2 predpomnilnik in L3 predpomnilnik. Dostop do pomnilniškega podsistema je v veliki meri neodvisen od logičnih procesorjev. Pomnilniški razvrščevalnik pomnilniškemu podsistemu pošilja mikrooperacije nalaganja in shranjevanja neodvisno od logičnega procesorja. Omenjeni podsistem le-te obravnava po takem vrstnem redu, kot vstopajo vanj.

5.4.1 Podatkovni preslikovalni predpomnilnik

Ker je dostop do tabele strani[14] v glavnem pomnilniku časovno potraten in se preslikovanje naslovov izvaja ob vsakem dostopu do pomnilnika, je dobro imeti predpomnilnik, ki skrajšuje čas dostopov do tabele strani. Za to skrbi podatkovni preslikovalni predpomnilnik (angl. data translation lookaside buffer – DTLB). Deluje enako kot vsi ostali predpomnilniki, vendar ne vsebuje ukazov in operandov, temveč pretvarja navidezne naslove v fizične naslove glavnega pomnilnika. Takšen način delovanja pospešuje preslikovanje naslovov. Podatkovni

[14] Tabela strani je tabela, ki hrani povezave med navideznimi in fizičnimi naslovi.

preslikovalni predpomnilnik vsebuje 64 popolnoma asociativnih zapisov (vsak zapis se lahko preslika v stran velikosti 4KB ali 4MB). Logična procesorja si podatkovni preslikovalni predpomnilnik delita med seboj, zato vsak zapis vanj vsebuje identifikacijsko oznako procesorja. Vsakemu logičnemu procesorju pripada register rezervacij (angl. reservation register), ki se uporablja za zagotavljanje poštenosti in nadaljevanja izvajanja pri zgrešitvah v podatkovnem preslikovalnem predpomnilniku.

5.4.2 Predpomnilniki

L1 podatkovni predpomnilnik je niz asociativni pomnilnik s stopnjo asociativnosti 4 (angl. 4-way set associative memory). Ta podatkovni predpomnilnik vsebuje vrstice dolžine 64 zlogov. V L1 predpomnilniku se izvaja tako imenovano pisanje skozi (angl. write through), kar pomeni, da se njegovi zapisi sočasno zapisujejo tudi v L2 predpomnilnik. L2 in L3 predpomnilnika sta primera niz asociativnega pomnilnika s stopnjo asociativnosti 8 s 128 zlogov dolgimi vrsticami. Oba predpomnilnika uporabljata fizično naslavljanje. Logični procesorji, ne glede na pripadnost mikrooperacij, prenesejo podatke v predpomnilnik. Oba logična procesorja si delita vse zapise na vseh treh predpomnilniških ravneh. Ker si logični procesorji med seboj delijo podatke iz predpomnilnika, obstaja možnost, da prihaja do »spora« v predpomnilniku (logični procesorji zahtevajo iste podatke). Vendar kljub temu obstaja možnost za delitev podatkov v predpomnilniku. Npr., prvi logični procesor lahko prevzame ukaze ali podatke, ki jih drugi logični procesor potrebuje. V modelu proizvajalec – porabnik prvi logični procesor proizvaja podatke, medtem ko jih drugi logični procesor uporablja. Uporaba modela proizvajalec – porabnik zmanjšuje oziroma preprečuje spore v predpomnilniku.

6 TEHNIKE OPTIMIZACIJE

Ponavadi posamezne aplikacije za učinkovitejše delovanje potrebujejo optimizacijo določenih področij. Poglavje povzema optimizacijo različnih področij za učinkovitejše delovanje večnitnih aplikacij. Ta področja so:
- sinhronizacija niti,
- optimizacija sistemskega vodila,
- optimizacija pomnilnika,
- optimizacija vhodnega podsistema in
- optimizacija izvrševalnih virov.

6.1 Sinhronizacija niti

Aplikacije z večjim številom niti za zagotavljanje pravilnosti izvajanja ukazov uporabljajo različne tehnike sinhronizacije. Nepravilna sinhronizacija niti lahko povzroči nepravilno izvajanje aplikacij, saj lahko, npr., ena nit prebere podatke in jih nato spremeni, istočasno pa druga nit prav tako prebere iste podatke, ki pa niso še spremenjeni in je rezultat operacije posledično napačen. Za sinhronizacijo niti so pogosto uporabljene različne tehnike kodiranja in sistemskih klicev. Te tehnike vključujejo čakalne zanke, krožne ključavnice, kritična področja itd. Izbira optimalnega sistemskega klica, glede na pogoje oziroma implementacijo sinhronizacijske kode z vzporednim izvajanjem, ima velik pomen pri minimizaciji sinhronizacije niti.

Ključni koraki sinhronizacije niti predstavljajo:
- sinhronizacija za krajše časovno obdobje,
- sinhronizacija s krožno ključavnico[15],
- sinhronizacija za daljše časovno obdobje in
- preprečevanje uporabe skupnih podatkov.

6.1.1 Sinhronizacija za krajše časovno obdobje

Pogostost in trajanje sinhronizacije, ki je potrebna, je odvisna od značilnosti aplikacij. Čakalne zanke se uporabljajo v primerih, ko sinhronizacijska zanka potrebuje izredno hiter

[15] Je ključavnica, v kateri nit čaka in v zanki ponavljajoče preverja na dostopnost ključavnice. Nit z uporabo take ključavnice ostaja aktivna, vendar ne izvaja ničesar, temveč porablja procesorski čas.

odziv aplikacije. V večini primerov se čakalna zanka uporablja takrat, ko mora ena nit čakati krajše časovno obdobje, da druga nit doseže točko sinhronizacije. Sestavlja jo zanka, v kateri poteka primerjanje spremenljivke sinhronizacije z vnaprej določeno vrednostjo. Na modernih mikroprocesorjih takšna zanka omogoča vzporedno branje zahtev niti, ki se ponavljajoče vrti v zanki. Te zahteve se običajno izvajajo neurejeno. Vsaki izmed prebranih zahtev je dodeljen medpomnilnik. V primeru pisanja s strani ene niti med nalaganjem druge niti procesor zagotavlja preprečevanje neskladnosti v pomnilniški urejenosti.

6.1.2 Sinhronizacija s krožno ključavnico

Krožne ključavnice se uporabljajo v primeru, ko veliko število niti želi spreminjati sinhronizacijsko spremenljivko. Ta spremenljivka mora biti zaklenjena s ključavnico, ki nitim preprečuje nenamerno prepisovanje vrednosti spremenljivke. Ko se ključavnica odklene, niti tekmujejo med seboj za pridobitev ključavnice. To tekmovanje povzroča upadanje učinkovitosti sistema. Eno izmed rešitev zgoraj omenjenega problema predstavlja uporaba tehnike programskega cevovoda. Uporablja se za rokovanje s podatki, ki si jih morajo niti deliti med seboj. Za preprečevanje tekmovalnosti med nitmi za pridobitev ključavnice so pravice dostopa do posamezne ključavnice dodeljene največ dvema nitima.

6.1.3 Sinhronizacija za daljše časovno obdobje

Pri uporabi čakalne zanke, pri kateri ne pričakujemo, da bo nit le-to hitro zapustila, mora aplikacija zagotoviti, da mora nit za svoje izvajanje v čakalni zanki čakati daljše časovno obdobje ter da aplikacije uporabljajo storitve operacijskega sistema za blokiranje čakajočih niti. Posledico blokiranja čakajočih niti predstavlja sproščanje procesorskih virov, kar pomeni, da lahko druga izvajajoča nit uporablja vse procesorske izvrševalne vire, ki so na voljo. Na procesorjih, ki podpirajo hipernitnost, operacijski sistemi uporabljajo ukaz HLT v primerih, če je eden logični procesor aktiven in drugi neaktiven. HLT ukaz omogoča prostemu logičnemu procesorju prehod v neaktivno stanje in s tem drugemu logičnemu procesorju omogoči uporabo vseh procesorskih virov. Operacijski sistemi, ki te tehnike ne uporabljajo, izvajajo ukaze na obeh logičnih procesorjih, kjer drugi konstantno preverja, ali je kakšno opravilo v vrsti za izvajanje. Takšnemu preverjanju z drugimi besedami pravimo »prazna zanka«. Ta zanka povzroča porabljanje izvrševalnih virov, ki bi v nasprotnem primeru bili uporabljeni za izvajanje ukazov aktivnega logičnega procesorja. Če nit ostane

nezaposlena določen čas, aplikacija uporabi tehniko blokiranje niti oziroma kakšno drugo metodo sproščanja procesorskih virov. Tehnika blokiranja niti omogoča nadzorni niti manj procesorskih ciklov, namenjenih za čakanje in vrtenje v zanki.

6.1.4 Preprečevanje napačne uporabe skupnih podatkov

Preprečevanje napačne uporabe skupnih podatkov ima izredno pomembno vlogo v primerih delitve skupnih podatkov med dvema ali več nitmi. Napačna uporaba skupnih podatkov se nanaša na podatke, ki jih ena nit uporablja in se nahajajo na isti predpomnilniški liniji[16] kot različni podatki, ki jih uporablja druga nit. Primer napačne uporabe skupnih podatkov se zgodi, ko se privatni podatki niti in sinhronizacijska spremenljivka niti nahajajo znotraj meje predpomnilniških linij velikosti 64 zlogov za pisanje ali velikosti 128 zlogov za pisanje. V primeru spremembe sinhronizacijske spremenljivke s strani ene niti je "umazani bit" predpomnilniške linije zapisan v glavni pomnilnik in posodobljen za vsak procesor, ki si deli vodilo. Napačna uporaba skupnih virov zmanjšuje učinkovitost delovanja v primerih, ko se dve niti izvajata na različnih fizičnih procesorjih ali dveh logičnih procesorjih fizičnega procesorja. V primeru, ko se dve niti izvajata na različnih fizičnih procesorjih, se vpliv na učinkovitost delovanja nanaša na odstranjevanje podatkov iz predpomnilnika za ohranjanje predpomnilniške koherence. V primeru, da se dve niti izvajata na različnih logičnih procesorjih fizičnega procesorja, se vpliv na učinkovitost delovanja nanaša na pomnilniško urejenost podatkov.

6.2 Dodatno delo in breme zaradi sinhronizacije

Za dostopanje niti do skupnih virov le-te uporabljajo signale, imenovane semaforji (angl. semaphore). Semaforji predstavljajo mehanizem pogojne sinhronizacije med nitmi oziroma procesi, z namenom zagotavljanja medsebojnega izključevanja dostopanja procesov do kritičnih področij[17]. Semaforji so podatkovna struktura, ki vključuje celoštevilsko spremenljivko, nad katero sta definirani (atomarni) operaciji vstopa in sprostitve kritičnega področja. Problemi se pojavijo, ko dodatno delo, kot posledica uporabe mehanizmov sinhronizacije (semaforji, ključavnice ali ostale metode), potrebuje preveč časa v primerjavi s

[16] Je najmanjša pomnilniška enota, ki se lahko prenaša med predpomnilnikom in glavnim pomnilnikom. Predpomnilniška linija vsebuje dejanske podatke, ki so bili prevzeti iz glavnega pomnilnika.
[17] Je področje, v katerem se lahko naenkrat nahaja samo en proces hkrati.

časom, ki je potreben za izvedbo sekcije. Če mehanizmi sinhronizacije porabljajo preveč časa, potem nitkanje določenih delov aplikacij nima nobenega smisla. Zato je z identifikacijo problema potrebno bodisi zmanjšati sinhronizacijski čas, bodisi odstraniti nit.

6.3 Optimizacija sistemskega vodila

Upravljanje s prometom na vodilu v sistemih s hipernitnostjo lahko bistveno pripomore k povečanju učinkovitosti sistema. Programerji lahko s podatki upravljajo s povečevanjem lokalnosti podatkov ter ukazov za ohranjanje pasovne širine vodila[18] (angl. conserve bus bandwidth) in z izogibanjem prekomerni uporabi ukazov za vnaprejšnje prevzemanje ukazov ter podatkov. Intel Xeon procesorji vsebujejo enoto za samodejno vnaprejšnje prevzemanje ukazov (angl. hardware prefetcher)[19]. Prekomerna uporaba programskega vnaprejšnjega prevzemanja ukazov (angl. software prefetcher) povzroča poslabšano delovanje sistema.

Upravljanje prometa na vodilu ima velik vpliv na učinkovitost večnitne programske opreme. Ključni koraki sinhronizacije sistemskega vodila za doseganje visoke prepustnosti in hitrega odziva na zahteve so:
- izboljševanje lokalnosti podatkov in
- izogibanje prekomerni uporabi programskih ukazov za vnaprejšnje prevzemanje (angl. prefetching) ukazov ter omogočanje samodejnega vnaprejšnjega prevzemanja ukazov realiziranega s strojno opremo.

6.3.1 Izboljšanje lokalnosti podatkov

Ohranjanje pasovne širine vodila lahko pripomore k višjim zmogljivostim sistema, saj preobremenjevanje vodila povzroča počasnejše delovanje sistema. Eden izmed načinov ohranjanja razpoložljivosti pasovne širine vodila predstavlja izboljševanje lokalnosti kode in podatkov. Povečevanje lokalnosti podatkov zmanjšuje število odstranjevanj iz predpomnilniških linij in zahtev po prevzemanju podatkov. Ta tehnika prav tako zmanjšuje število prevzemanj ukazov iz sistemskega pomnilnika.

[18] Uporabljanje vodila na tak način, da se lahko preostale operacije sočasno izvajajo, ne da bi se izvajanje le-teh upočasnilo.
[19] Je enota, ki samodejno analizira potrebe procesorja in prevzame podatke in ukaze iz glavnega pomnilnika v L2 podatkovni predpomnilnik, ki bodo v prihodnosti verjetno potrebovani.

6.3.2 Izogibanje prekomernemu vnaprejšnjemu prevzemanju ukazov in podatkov

Intel procesorji vsebujejo enoto za vnaprejšnje prevzemanje ukazov, ki deluje samodejno. Ta enota omogoča, da na podlagi vzorcev prej izvedenih ukazov "ugane", katere podatke bo procesor potreboval v naslednji operaciji in tako poskrbi za njihov "predčasen" prihod v predpomnilnik. V večini situacij enota za vnaprejšnje prevzemanja ukazov in podatkov pripomore k zmanjševanju zamud v pomnilniškem sistemu brez posredovanja programskih ukazov, ki bi zahtevali vnaprejšnje prevzemanje ukazov. Uporaba programskih ukazov za vnaprejšnje prevzemanje ukazov praktično neizogibno pušča posledice v delovanju sistema. To pa zaradi tega, ker njihova prekomerna uporaba preobremenjuje sistemsko vodilo. Prav tako enoti za vnaprejšnje prevzemanje ukazov in podatkov preprečuje prevzemanje podatkov, ki jih procesorsko jedro dejansko potrebuje. Poleg tega zapravlja tudi pomembne izvrševalne vire, s čimer povzroča upočasnjeno izvajanje ukazov.

6.4 Optimizacija pomnilnika

Podatki v L1 podatkovnem predpomnilniku so označeni in indeksirani v linearne naslove[20], medtem ko so podatki v L2 in L3 predpomnilniku označeni in indeksirani v fizične naslove. Dva logična procesorja lahko naslavljata podatke na istem linearnem naslovu, ki je lahko preslikan v različne fizične naslove. Ko nastopi tekmovanje za dostop do naslovov, ki se nahajajo eden poleg drugega, lahko pride do ponavljajočega odstranjevanja iz pomnilnika ter dodeljevanja predpomnilniških linij. Različne tehnike se uporabljajo za reševanje slednjega problema. Eno izmed tehnik predstavlja nastavljanje sklada vsake niti z uporabo odmika, ki ni večkratnik 64KB ali 1MB. Za optimizacijo pomnilnika se lahko uporabljajo tudi tehnike kot sta:

- tehnika predpomnilniških blokov in
- optimizacija skupnega prostora.

6.4.1 Tehnike predpomnilniških blokov

Obstaja veliko dejavnikov, ki vplivajo na delovanje predpomnilnika. Enega izmed dejavnikov, ki vplivajo na delovanje predpomnilnika, predstavlja lokalnost podatkovnega predpomnilnika. Za izkoriščanje omenjenega dejavnika je uporabljena tehnika

[20] So naslovi, preračunani iz virtualnih naslovov virtualnega pomnilnika, realiziranega s segmentacijo.

predpomnilniških blokov. Ta tehnika preoblikuje vzorec dostopa do pomnilnika v bloke, ki ustrezajo podatkovnemu predpomnilniku. Vsak podatkovni element v nizu je ponovno uporabljen v podatkovnem bloku, tako da posamezni blok podatkov ustreza podatkovnemu predpomnilniku. Učinkovitost tehnike predpomnilniških blokov je odvisna od aplikacije do aplikacije. Ker L2 predpomnilnik vsebuje ukaze in podatke, prevajalniki pogosto z združevanjem povezanih blokov ukazov izkoriščajo prednost lokalnosti ukazov. Tehnika predpomnilniških blokov vpliva predvsem na video in avdio aplikacije, kjer obdelava slike lahko poteka na manjših delih od celotnega video okvirja ali okvirja slike. Učinkovitost omenjene tehnike je v veliki meri odvisna tudi od velikosti podatkovnega bloka, velikosti procesorskega predpomnilnika ter od dejstva, kolikokrat so bili določeni podatki ponovno uporabljeni[27].

6.4.2 Optimizacija skupnega pomnilnika

Uporaba predpomnilnika in uporaba skupnih podatkov je ključnega pomena za visoko zmogljive večnitne aplikacije. Osnovno enoto deljenja podatkov med procesorji predstavlja predpomnilniška linija. Procesorji s hipernitnostjo uporabljajo skupen predpomnilniški prostor. Pri uporabi skupnega pomnilnika se lahko pojavita dva problema. Prvega predstavlja slabša učinkovitost sistema, ki nastopi v primeru, ko več procesorjev poskuša sočasno dostopati do skupnega pomnilnika. Drugega pa predstavlja problem v koherenci pomnilnika[21], ki se pojavi takrat, ko eden izmed procesorjev spremeni kopijo neke vrednosti in vse kopije posledično ne zavzemajo iste vrednosti. Optimizacija skupnega pomnilnika zajema:

- minimizacijo deljenja podatkov med fizičnimi procesorji in
- izogibanje 64K preimenovanju na prvem nivoju podatkovnega predpomnilnika.

6.4.2.1 Minimizacija deljenja podatkov med fizičnimi procesorji

V primeru, ko se dve niti, ki si delita podatke, izvajata na dveh fizičnih procesorjih, branje iz oziroma pisananje v skupen vir vključuje številne transakcije na vodilu, kot, na primer, zahteve po spremembi lastništva in pošiljanje podatkov po vodilu. Eno izmed tehnik minimizacije deljenja skupnih podatkov predstavlja kopiranje podatkov na lokalni sklad za spremenljivke. Če je potrebno, se lahko rezultati niti kasneje shranijo nazaj v skupno

[21] Kopije v predpomnilniku so koherentne v primeru, da vse vrednosti kopij zavzemajo enako vrednost.

pomnilniško lokacijo. Ta pristop lahko prav tako minimizira čas, ki je potreben za sinhronizacijo dostopa do skupnih podatkov.

6.4.2.2 Izogibanje 64K preimenovanju na prvem nivoju podatkovnega predpomnilnika

64 zlogov velika podatkovna struktura ali niz mora biti poravnan tako, da je njen naslov večkratnik števila 64. Razvrščanje podatkov po padajoči dolžini predstavlja hevristiko za naravno poravnavo (angl. natural alignment)[22]. Dokler 16 zlogov velike meje niso presežene, naravna poravnava ni nujno potrebna, čeprav je uveljavitev le-te preprosta.

Nekateri operacijski sistemi ustvarjajo prostor v skladu, poravnan na mejo velikosti 1MB. Dve niti, ki se izvajata vzporedno, imata zelo verjetno veliko spremenljivk sklada, lociranih na območjih naslovov, ki ustrezajo 64K »izravnalnem« pogoju. Za dostop do katerekoli spremenljivke, deklarirane v skladu, se zato uporablja 64K preimenovanje[23]. Rešitev problema predstavlja dodeljevanje vrednosti odmika sklada (angl. offset value), ki se spreminja ob vsakem odmiku sklada vsake niti. Če 64K preimenovanje povzroča probleme, se za rešitev le-teh uporablja VTune Performance Analyzer[8], ki sledi 64K preimenovanju dogodkov. To nam omogoča najdbo omenjenega problema, povezanega z skladom niti in najdbo problema v ostalih delih kode. Določanje vrednosti odmika sklada, ponovno prevajanje in nato ponovno testiranje z VTune Performance Analyzer, se uporablja za obravnavanja zgoraj omenjenega problema. Za določanje vrednosti odmika sklada je potrebno razporediti pomnilnik za vsak programski sklad vsake niti. Za vsako nit, na primer, dodelimo vrednost odmika sklada v velikosti, ki je bila določena. Manipuliranje z odmikom sklada zmanjša število 64K preimenovanj[27].

[22] Naravna poravnava je poravnava, ki omogoča usklajenost naslovov vseh objektov z večkratniki velikosti objekta.
[23] V primeru, da dva navidezna naslova naslavljata predpomnilniški liniji, ki se nahajata 64KB narazen, bosta naslavljala isto predpomnilniško linijo v L1 podatkovnem predpomnilniku. Nastali problem se obravnava kot problem 64K preimenovanja.

7 PODPORA OPERACIJSKIM SISTEMOM

Operacijski sistemi in aplikacije na sistemih s procesorji, ki uporabljajo hipernitnost, zaznajo dvakrat večje število procesorjev, kot jih dejansko sistem vsebuje. To je posledica tega, da operacijski sistemi logične procesorje obravnavajo kot fizične procesorje. Za izboljšanje delovanja sistema imajo operacijski sistemi implementirani dve optimizacijski tehniki. Prvo predstavlja uporaba ukaza HALT. Uporaba tega ukaza se pripeti ob aktivnosti samo enega logičnega procesorja. Ukaz procesorju omogoča prehod iz eno-opravilnega logičnega procesorja 0 (angl. single-task logical processor 0 – ST0) na eno-opravilni logični procesor 1 ali obratno. V obeh načinih je hkrati aktiven samo eden logičen procesor. Razdeljeni viri so v obeh načinih preoblikovani tako, da lahko aktiven logični procesor uporablja vse vire. Operacijski sistemi, ki te tehnike optimizacije ne uporabljajo, se spopadajo s problemom prazne zanke (angl. idle loop). Te zanke izrabljajo zelo pomembne izvrševalne vire, s katerimi lahko povečajo učinkovitost izvajanja drugega aktivnega logičnega procesorja. Druga metoda optimizacije se pojavlja v povezavi z dodeljevanjem programskih niti logičnim procesorjem. V splošnem operacijski sistemi za boljšo učinkovitost sistema niti najprej dodeljujejo logičnim procesorjem različnih fizičnih procesorjev, nato se dodeljevanje preusmeri na isti fizični procesor. Ta optimizacijska tehnika programskim nitim dovoljuje uporabo izvrševalnih virov različnih fizičnih procesorjev, takoj ko je to mogoče.

Nadaljevanje poglavja sem zaradi večjega števila uporabnikov posvetil Windows operacijskim sistemom.

7.1 BIOS podpora hipernitnosti

BIOS (angl. basic input/output system) omogoča dve pomembni funkciji za hipernitnost, ki sta omogočanje in onemogočanje te tehnologije ter nastavitev zaporedja zagonov logičnih procesorjev (angl. logical processor startup sequence).

7.1.1 Zaporedje zagonov logičnih procesorjev

Zaporedje, v katerem so logični procesorji zagnani, je zelo pomembno, predvsem ko se na sistemu s hipernitnostjo poganja programska oprema, ki te tehnologije ne podpira. Za zagon logičnih procesorjev je odgovoren BIOS. BIOS skrbi za seznam vseh delujočih logičnih

procesorjev, ki ga po kreaciji dostavi operacijskemu sistemu v obliki opisne tabele naprednega programabilnega krmilnika prekinitev (angl. multiple advanced programmable interrupt controller description table – MADT)[19]. Definicija tabele je določena po specifikaciji naprednega vmesnika za konfiguracijo in porabo (angl. advanced configuration and power interface – ACPI)[24][19]. Operacijski sistem poskuša izkoristiti delovanje logičnih procesorjev po istem vrstnem redu razvrstitve v opisni tabeli naprednega programabilnega krmilnika prekinitev. Sistem BIOS najprej kreira seznam prvih logičnih procesorjev vseh fizičnih procesorjev, sledi pa mu seznam vseh drugih logičnih procesorjev. Ta strategija zagotavlja, da operacijski sistem poskuša uporabiti logične procesorje v podanem zaporedju. Takšno zapisovanje lahko pripomore k optimalnejši učinkovitosti operacijskega sistema[10].

7.1.2 Omogočanje in onemogočanje hipernitnosti

BIOS dovoljuje omogočanje oziroma onemogočanje hipernitnosti. V primeru, da je hipernitnost onemogočena, operacijski sistem aktivira samo vse prve logične procesorje. Na teh procesorjih MADT tabele operacijskemu sistemu zagotavljajo podatke samo o vseh prvih logičnih procesorjih, saj so vsi drugi neuporabni. V splošnem so večnitne Windows aplikacije bolj učinkovite na procesorjih s hipernitnostjo v primerjavi z delovanjem na procesorjih, ki te tehnologije ne podpirajo. Od same aplikacije je odvisna pridobitev na učinkovitosti, ki je dosežena s preprečevanjem tekmovalnosti za skupne procesorske vire.

7.2 Windows modeli licenciranja

Operacijski sistem vse logične procesorje zazna kot samostojen procesor. To pomeni, da orodja ali storitve, ki prikazujejo informacije o procesorjih, kot sta npr: upravitelj opravil (angl. task manager) ali zmogljivost monitorja (angl. performance monitor), prikazujejo informacije o vsakem logičnem procesorju, ki ga operacijski sistem uporablja.

Trenutni model licenciranja operacijskega sistema Windows za omogočanje hipernitnosti predstavlja pridobitev licenc procesorjev za vsak fizični procesor. Dobro je vedeti, da

[24] Specifikacija za učinkovito upravljanje porabe energije namiznih in mobilnih računalnikov. Ta specifikacija določa, kako računalniški vhodno/izhodni sistem, operacijski sistem in zunanje enote komunicirajo med seboj glede porabe energije. ACPI je ključna komponenta Intelove tehnologije.

programski produkti, ki so bili izdelani pred uvedbo hipernitnosti, te tehnologije ne bodo podpirali in bodo vsak logični procesor smatrali za fizičnega.

7.2.1 Operacijski sistemi, ki hipernitnosti ne podpirajo

Vse različice operacijskega sistema Windows 2000 podpirajo hipernitnost in na sistemih, ki to tehnologijo omogočajo, delujejo brezhibno. Kot rezultat ta verzija operacijska sistema obravnava vsak logičen procesor kot samosvoj fizični procesor. Operacijski sistem poskuša aktivirati vsakega od njih v takem zaporedju, kot ga je BIOS določil. Proces aktiviranja procesorjev traja toliko časa, dokler število aktiviranih procesorjev ni ekvivalentno številu procesorjev, določenih v procesorski licenci operacijskega sistema. Različne verzije Windows 2000 podpirajo različno število procesorjev. Koliko procesorjev podpira določena verzija operacijskega sistema Windows 2000, prikazuje tabela 2.

Tabela 2 Največje število procesorjev, ki jih različice operacijskega sistem Windows 2000 podpirajo

Verzija operacijskega sistema Windows 2000	Največje število procesorjev
Windows 2000 Professional	2
Windows 2000 Standard Server	4
Windows 2000 Advanced Server	8
Windows 2000 Datacenter Server	32

7.2.1.1 Nepravilno zaporedje zagonov logičnih procesorjev

Slika 16 prikazuje primer pravilnega zagona logičnih procesorjev. V tem primeru BIOS aktivira logične procesorje v priporočljivem zaporedju. Številke označujejo številko logičnega procesorja. Na sliki sta s sivo barvo obarvana procesorja, ki ju operacijski sistem dejansko uporablja. V takem primeru operacijski sistem uporablja prva logična procesorja vsakega fizičnega procesorja. Takšna razporeditev omogoča optimalno delovanje operacijskega sistema glede na konfiguracijo strojne opreme.

Slika 16 Logični procesorji so razporejeni v priporočljivem zaporedju.

V primeru nepravilne razporeditve logičnih procesorjev (logični procesorji niso razporejeni po priporočljivem zaporedju) operacijski sistem uporabi oba logična procesorja iz istega fizičnega procesorja. Omenjen primer prikazuje spodnja slika, kjer številke označujejo zaporedje logičnih procesorjev.

Slika 17 Logični procesorji niso razporejeni v priporočljivem zaporedju.

V tem primeru operacijski sistem uporablja dva logična procesorja prvega fizičnega procesorja. Posledico takšne konfiguracije predstavlja slabše delovanje sistema, zaradi tekmovalnosti za skupne vire fizičnega procesorja in neizkoriščenosti drugega logičnega procesorja.

7.2.1.2 Uporaba drugega logičnega procesorja

Če licenca operacijskega sistema dovoljuje več procesorjev, kot jih podpira sistem (število fizičnih procesorjev je manjše od največjega števila procesorjev, ki jih operacijski sistem dovoljuje), potem operacijski sistem samodejno uporablja tudi drugi logični procesor. V tem primeru lahko sklepamo, da če BIOS omogoča hipernitnost, potem so vsi logični procesorji uporabljeni ne glede na razporeditev.

7.2.2 Operacijski sistemi, ki podpirajo hipernitnost

Vsak procesor, ki podpira hipernitnost, vsebuje identifikacijsko številko procesorja, ki ga za delovanje posameznega logičnega procesorja uporablja operacijski sistem. V primeru, da sistem vsebuje procesorje s hipernitnostjo, so le-ti razdeljeni na dva logična procesorja, s katerima upravlja operacijski sistem. Slednji lahko upravlja s tolikšnim številom logičnih ali fizičnih procesorjev, kot znaša licenčni limit operacijskega sistema. Licenčni limit omejuje število procesorjev, s katerimi lahko operacijski sistem razpolaga. Slika 18 prikazuje primer operacijskega sistema Windows XP Professional, katerega licenčni limit znaša dva. Tako kot na prejšnji sliki, so tudi na tej sliki logični procesorji, ki jih operacijski sistem dejansko uporablja, obarvani s sivo barvo.

Slika 18 Operacijski sistem uporablja vse štiri logične procesorje.

V primeru, ki ga prikazuje zgornja slika, Windows operacijski sistem uporablja oba logična procesorja na fizičnem procesorju po vrstnem redu, ki ga določi BIOS. Operacijski sistem se sprehaja po seznamu procesorjev, ki ga je določil BIOS, dokler ne zazna dveh niti. Vsaka posamezna nit se izvaja na svojem logičnem procesorju vsakega fizičnega procesorja. Rezultat takšnega izvajanja predstavlja uporabnost obeh fizičnih procesorjev. Takšna strategija omogoča optimalno delovanje Windows operacijskih sistemov na podlagi konfiguracije strojne opreme.

Vsaka različica operacijskih sistemov lahko podpira različno število tako fizičnih kot logičnih procesorjev. Tabela 3 prikazuje licenčni limit različnih novejših verzij Windows operacijskih sistemov.

Tabela 3 Največje število procesorjev, ki jih različni operacijski sistemi Windows podpirajo

Verzija operacijskega sistema Windows	Največje število procesorjev	Največje število logičnih procesorjev
Windows 7 Professional, Ultimate, Enterprise	2	4
Windows 7 Starter, Home Basic, Home Premium	1	2
Windows Vista Professional, Ultimate, Enterprise	2	4
Windows Vista Starter, Home Basic, Home Premium	1	2
Windows XP Home Edition	1	2
Windows XP Professional	2	4
Windows Server 2008 R2 Itanium	64	256
Windows Server 2008 R2 Datacenter	64	256
Windows Server 2008 R2 Enterprise	8	32
Windows Server 2008 R2 HPC	4	16
Windows Server 2008 R2 Web	4	16
Windows Server 2008 R2 Standard	4	16
Windows Server 2008 R2 Foundation	1	4
Windows Server 2003, Standard	4	8
Windows Server 2003, Enterprise	8	16
Windows Server 2003, Datacenter	32	32

Če je število logičnih procesorjev večje od števila procesorjev, ki jih operacijski sistem dejansko podpira, se zgodi, da ostali tako imenovani »odvečni« logični procesorji niso uporabljeni. Slednji primer prikazuje slika 19.

Slika 19 Štirje delujoči in dva nedelujoča procesorja.

Iz tabele 3 je razvidno, da operacijski sistem Windows XP Professional podpira dva fizična oziroma štiri logične procesorje. V primeru, ki ga prikazuje zgornja slika, lahko vidimo, da sistem sestavljajo trije fizični oziroma šest logičnih procesorjev. Ker je število logičnih procesorjev večje od števila, ki jih operacijski sistem dejansko podpira, pride do dvojne omejitve. Dvojna omejitev pomeni, da se najprej pregleda število fizičnih procesorjev. Če je število teh večje, so le-ti odvečni procesorji za sistem neuporabni. Nato se pregleda še število logičnih procesorjev. V primeru, da je število logičnih procesorjev večje od števila, ki ga operacijski sistem dejansko podpira, sta zadnja dva logična procesorja glede na vrstni red logičnih procesorjev za sistem neuporabna. Za popolno izkoriščanje procesorske arhitekture je treba paziti na pravilno izbiro različice operacijskega sistema.

8 ZAKLJUČEK

Intelova tehnologija hipernitnosti vpelje koncepte hkratnega večnitenja v procesorsko arhitekturo. Ta korak predstavlja novo usmeritev v izdelavi Intelovih procesorjev. S časom postajajo tehnike za povečevanje zmogljivosti procesorjev in zmanjševanje tranzistorjev ter porabo električne energije zmeraj pomembnejše.

V tej implementaciji hipernitnosti se na vsakem fizičnem procesorju nahajata dva logična procesorja. Logični procesorji imajo svoje arhitekturno stanje in si med seboj delijo skoraj vse fizične izvrševalne in strojne procesorske vire. Prvi cilj takšne implementacije predstavlja minimalno povečanje stroškov v zameno za čim večje povečanje procesorske zmogljivosti. Drugi cilj pa predstavlja nadaljevanje izvajanja na enem logičnem procesorju, kljub zaustavitvi drugega, in doseganje višje zmogljivosti, kljub aktivnosti le enega izmed njih.

Doseganju omenjenih ciljev je kljubovala učinkovita izbira algoritmov za razdeljevanje in združevanje številnih virov. Povečanje zmogljivosti procesorjev, ki jo ta tehnologija doseže, znaša 30 %. Potencial hipernitnosti je ogromen, zato lahko v prihodnosti pričakujemo še številne boljše oziroma zmogljivejše implementacije te tehnologije.

9 LITERATURA

[1] Intel Corporation. Wikipedia, 2011. URL: http://en.wikipedia.org/wiki/Intel_Corporation (citirano 25. 10. 2010).

[2] SERŠ Maribor, Strokovna gimnazija, 2002, Dejan Haber. URL: http://www.s-sers.mb.edus.si/gradiva/w3/sistemi/dolge.html (citirano 25. 10. 2010).

[3] Von Neumann architecture Wikipedia. URL: http://en.wikipedia.org/wiki/Von_Neumann_architecture.

[4] Dušan Kodek, januar 1994. Arhitektura računalniških sistemov, 1. izdaja. Ljubljana, založba BI-TIM d.o.o.,

[5] Dušan Kodek, maj 1993. Mikroprocesorski sistemi, 1. izdaja. Ljubljana, založba BI-TIM d.o.o.,

[6] David Harris, 2004. Introduction to CMOS VLSI Design. URL: http://www.cmosvlsi.com/lect22.ppt.

[7] Multiprocessor throughput scalability. URL: http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/multiproc_throughput_scal.htm (citirano 2. 11. 2010).

[8] Intel Corporation, januar 2003. URL: http://www.cslab.ntua.gr/courses/advcomparch/2008/material/readings/Intel%20Hyper-Threading%20Technology.pdf (citirano 3. 11. 2010).

[9] Računalniška organizacija (RO), 2006, Mira Trebar. URL: https://www.kiberpipa.org/projekti/fri/browser/or/vaje/or_vaja4.pdf?rev=94.

[10] Jon "Hannibal" Stokes, februar 2002. Introduction to Multithreading, Superthreading and Hyperthreading. URL: http://www.ic.unicamp.br/~cortes/mo601/artigos_referencias/MultiThreading_stokes.pdf (citirano 4. 11. 2010).

[11] David L. Hill, Desktop Products Group, Intel Corp. Glenn Hinton, Desktop Products Group, Intel Corp. David A. Koufaty, Desktop Products Group, Intel Corp. J. Alan Miller, Desktop Products Group, Intel Corp. Michael Upton, CPU Architecture, Desktop Products Group, Intel Corp, 2002. Hyper-Threading Technology Architecture and Microarchitecture, URL: ftp://download.intel.com/technology/itj/2002/volume06issue01/art01_hyper/vol6iss1_art01.pdf (citirano 10. 11. 2010).

[12] Nimrod Megiddo in Dharmendra S. Modha, april 2004. Outperforming LRU with an Adaptive Replacement Cache Algorithm. URL: http://www.almaden.ibm.com/cs/people/dmodha/ARC.pdf (citirano 1. 12. 2010).

[13] Debbie Marr, 2002. Hyper-Threading Technology in the Netburst™ Microarchitecture. URL: http://www.hotchips.org/archives/hc14/2_Mon/05_marr.pdf (citirano 26. 12. 2010).

[14] Hyper-thrading. Wikipedia, 2011. URL: http://en.wikipedia.org/wiki/Hyper-threading (citirano 16. 12. 2010).

[15] H. Gilbert, januar 2007. Introduction to PC Hardwar. URL: http://www.yale.edu/pclt/PCHW/Hyperthreading.htm (citirano 18. 1. 2011).

[16] Cay S. Horstmann in Gary Cornell, 2003. Core Java. URL: www.horstmann.com/corejava/cj7v2ch1.pdf (citirano 19. 2. 2011).

[17] Charles M. Kozierok, april 2001. The PC Guide. URL: http://www.pcguide.com/ref/cpu/arch/extSMP-c.html (citirano 3. 3. 2011).

[18] Venkatesh Chakrapani, julij 2000. ACPI (Advanced Configuration and Power Interface), URL; http://searchwinit.techtarget.com/definition/ACPI (citirano 25. 3. 2011).

[19] Tahir Cele, 2005. Hyper-Threading Technology Architecture and Micro-Architecture. URL: www.cmpe.boun.edu.tr/courses/cmpe511/fall2005/Tahir_Celebi.ppt (citirano 15.10.2011).

[20] James R. Bulpin, februar 2005. Operating system support for simultaneous multithreaded processors. URL: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-619.pdf (citirano 6.11.2011).

[21] Intel Corporation, junij 2005. IA-32 Intel® Architecture Optimization. URL: http://www.inf.ethz.ch/personal/iyves/pnc11/IntelOpt.pdf (citirano 17.7.2011).

[22] Humayun Khalid, Ph.D, avgust 2002. An Introduction to Hyper-Threading Technology in the Intel Xeon Processor Architecture. URL: http://ftp.dell.com/app/3q02-Ht1.pdf (citirano 28.10.2010).

[23] Phil Kerly, oktober 2008. Cache Blocking Technique on Hyper-Threading Technology Enabled Processors. URL: http://software.intel.com/en-us/articles/cache-blocking-technique-on-hyper-threading-technology-enabled-processors/ (citirano 8.10.2011).

[24] John Sladek, julij 2011. Modern microprocessors. URL: http://www.lighterra.com/papers/modernmicroprocessors/.

[25] Microsoft Corporation, 2002. Windows Support for Hyper-Threading Technology. URL: http://download.microsoft.com/download/5/7/7/577a5684-8a83-43ae-9272-ff260a9c20e2/Hyper-thread_Windows.doc (citirano 25. 7. 2011).

[26] Trace cache. Wikipedia. URL: http://en.wikipedia.org/wiki/CPU_cache#Trace_cache.

[27] Baruch Zoltan Francisc, 2002. Structure of Computer Systems. URL: http://users.utcluj.ro/~baruch/book_ssce/SSCE-Branch-Prediction.pdf (citirano 3. 10. 2011).

[28] John Stokes, 2002. Introduction to Multithreading, Superthreading and Hyperthreading, URL: http://arstechnica.com/old/content/2002/10/hyperthreading.ars/4 (citirano 14. 11. 2011).

[29] Instruction prefetch. Wikipedia. URL: http://en.wikipedia.org/wiki/Instruction_prefetch.