Istoric, tendințe şi perspective în evoluția microprocesoarelor

Prep. Claudia Mihaela Mark
Universitatea "Tibiscus" din Timişoara

ABSTRACT. In these thesis is presented the performance of the micrcontrollers on a exponential time frame of 30 years starting with their creation, it is to be observed that the miniaturization and the evolution of technology has contributed a lot to these amassing development. Also, the growth is coming to an end, due to physical barriers. At the end of the thesis are presented some of the possible future technology that will be in the next level of microcontrollers.

1 Etape în evoluția procesoarelor Pentium

Perioada 1993-1998 a fost marcată de o luptă dură pe piața microprocesoarelor, în care firma Intel a început să simtă prezența competiției formate din AMD, Cyrix sau NexGen. Tot în această perioadă s-a lansat şi standardul MMX, care este folosit şi în prezent.

Intel Pentium *(22 martie 1993)*
Intel Pentium a fost primul procesor superscalar creat de Intel (putea executa până la două instrucțiuni simultan). Intel nu a denumit acest procesor 80586, întrucât acest nume nu putea fi protejat de copyright. Astfel, Intel a folosit şi litere pentru a-şi diferenția produsele de cele ale concurenței.

Procesorul lucra cu o magistrală de date de 64 de biți (cu toate că a fost un procesor pe 32 de biți) şi a fost lansat inițial la viteze de 60 şi 66 de MHz. Au urmat însă, destul de rapid, versiuni de 75, 90, 100, 120, 133, 150, 166, 200, 233 MHZ. De fapt au existat trei versiuni de Pentium: prima versiune, care nu cuprindea decât două modele: Pentium la 60 şi la 66MHz, a doua versiune, care a adăugat instrucțiunile MMX şi o ultimă versiune,

care a micşorat distanţa dintre tranzistoare permiţând astfel viteze mai mari, care au ajuns până la 233MHz.

Intel Pentium a fost primul microprocesor pentru PC-uri care putea să calculeze mai mult de 100MIPS (milioane de instrucţiuni pe secundă). Tot pentru prima oară era posibilă construirea unor sisteme care să lucreze cu 2 procesoare în paralel (sisteme multiprocesor). Microprocesorul de la Intel avea 16Kb de cache încorporaţi în pastila de siliciu.

AMD K5 / Nexgen Nx586 *(1995)*

Firma AMD a reacţionat după doi ani, prin lansarea unui procesor comparabil, ca viteză, cu Intel Pentium. În 1995 a produs, totuşi, primul său procesor care era conceput integral de către companie, nemaifiind o simplă clonă a procesoarelor Intel.

Acest procesor s-a numit K5 şi avea viteze de la 75 la 166 MHz. Cu toate acestea, nu era un procesor mai rapid decât cele de la Intel, în plus, având o unitate de calcul în virgulă mobilă destul de slabă (ca şi Cyrix de altfel). Una din inovaţiile aduse de K5 era faptul că instrucţiunile x86 erau transformate intern în ROP (Risc OPerations). Aceste operaţii RISC se puteau executa în nucleul RISC al procesorului, care era mult mai rapid.

În acelaşi timp compania Nexgen lansa primul sau procesor: Nx586. Complexitatea procesorului K5 a dus la frecvenţe destul de mici, ceea ce a impus firmei AMD să cumpere compania Nexgen, care tocmai terminase design-ul noului său procesor, Nx686. Acest design a fost, ulterior, folosit de AMD în următoarea sa familie de procesoare, pe care avea să o lanseze în 1997.

Cyrix 6x86 *(Octombrie 1995)*

Acest chip era produs iniţial de către IBM, dat fiind că Cyrix nu avea unităţi de asamblare de procesoare, însă ulterior, odată cu achiziţionarea Cyrix de către National Semiconductor, a fost produs chiar de către compania Cyrix.

Procesorul a avut un succes destul de mare, întrucât era mai rapid decât un Intel Pentium la aceeaşi frecvenţă. De altfel, pentru a-l putea compara cu procesoarele de la Intel, specialiştii de la Cyrix au inventat ceea ce s-a numit ulterior P-Rating. De exemplu, procesorul Cyrix 6x86 care funcţiona la 150 de MHz a fost denumit 6x86PR200, ceea ce însemnă că era comparabil ca viteza cu un Pentium la 200.

Unul din marile dezavantaje ale acestui procesor a fost însă viteza foarte mică a calculelor în virgulă mobilă. Cu toate acestea, în aplicaţiile de tip office s-a dovedit cel puţin la fel de rapid ca şi un Pentium.

Intel Pentium Pro *(1 noiembrie 1995)*

Acest procesor a fost una dintre cele mai mari inovaţii tehnice produse de Intel până acum. Procesorul îngloba, pentru prima oară în istorie, pe lângă cache-ul Level1 de 8k pentru date şi 8k pentru instrucţiuni, şi un cache Level2 de 256Kb sau 512Kb.

Procesorul folosea un sistem complex de predicţie a ramurii de execuţie (branch prediction) şi execuţie speculativă (speculative execution) - în momentul în care execuţia programului ajungea la o bifurcaţie, ramura corectă nu era ştiută până în momentul în care se execută instrucţiunea condiţională; pentru ca procesorul să nu aştepte până în acea clipă, se alegea una din cele două ramuri şi se începea execuţia instrucţiunilor respective; dacă se dovedea că ramura aleasă a fost cea corectă aceasta însemna un câştig important de viteză.

Acest microprocesor transforma instrucţiunile x86 în microoperaţii care erau mai mici şi mai rapide. Acest lucru, cu toate că avea ca rezultat o viteză mai mare a instrucţiunilor de 32 de biţi, a dus la performanţe mai slabe în sistemele de operare care mai conţineau cod pe 16 biţi. Acesta a fost unul din motivele performanţei mai mici, comparabil cu Intel Pentium în Windows 95, de exemplu.

Intel Pentium MMX (Ianuarie 1997)

MMX nu înseamnă MultiMedia eXtension aşa cum s-ar înţelege iniţial; Intel a declarat că înseamnă Matrix Math eXtension. Acesta reprezintă un standard introdus de Intel ,care aduce câteva noi instrucţiuni ce uşurau, în principal, calculele matematice cu vectori.

AMD K6 (Aprilie 1997)

Ca urmare a cumpărării firmei Nexgen, AMD a reuşit să lanseze un nou procesor, K6, care avea viteze de la 166 la 266 MHz. Politica AMD a fost ca procesoarele sale să se vândă la aproape jumătate din preţul la care se vindeau procesoarele Intel. K6 încorpora instrucţiuni MMX (a căror licenţă a cumpărat-o de la Intel) devenind astfel un concurent pentru procesoarele Intel Pentium MMX.

Cyrix 6x86MX (30 Mai 1997)

6x86MX a adus nou extensiile MMX, precum şi viteze de ceas mai mari decât precedentele chip-uri de la Cyrix. Astfel cel mai performant model era 6x86MX PR266, care rula la 233MHz. De asemenea, mărimea memoriei cache Level2 s-a mărit de patru ori faţă de 6x86, ajungând la 64Kb.

Cyrix MII (14 Aprilie 1998)

Această versiune a chip-ului 6x86 a îmbunătățit puțin performanța FPU şi a atins viteze mai mari, ajungând la 300MHz (PR433). De asemenea, viteza bus-ului a ajuns la 100MHz.

Cyrix MediaGX (1998)

MediaGX a reprezentat încercarea lui Cyrix de a produce un chip care să integreze atât funcțiile de sunet şi video, cât şi controller-ul de memorie şi CPU-ul în sine. Scopul acestui chip a fost acela de a putea produce computere foarte ieftine şi, la vremea aceea, deja se vorbea de calculatoare sub 500$ (ceea ce era foarte puțin la acea dată) construite în jurul lui MediaGX. Cu toate acestea, nici unul din marii producători de computere nu a adoptat această soluție, astfel încât procesorul acesta, deşi revoluționar, nu a avut deloc succesul scontat.

2 Procesoarele Cyrix

Arhitectura x86

Compania Cyrix este unul dintre furnizorii de bază ai soluțiilor bazate pe microprocesoare, care a introdus noi standarde pe piața calculatoarelor personale. În ultimii zece ani, Cyrix a dezvoltat aproape o duzină de procesoare originale folosite în milioane de calculatoare din întreaga lume.

Fuziunea, din noiembrie 1997, cu National Semiconductor a avut două componente benefice pentru Cyrix: capacitatea de producție la nivel mondial a National Semiconductor şi infrastructura necesară acestei producții.

Primul produs Cyrix a fost un coprocesor matematic destinat creşterii vitezei de realizare a calculelor matematice. Succesul acestui coprocesor matematic a permis companiei Cyrix să distribuie, începând cu 1992, primul procesor din familia x86. Compania a dezvoltat rapid o linie de producție pentru procesoarele 486, şi apoi pentru procesoarele din generația a cincea 5x86, un CPU pentru sistemele PC (mobile şi desktop). În 1995, Cyrix a introdus procesorul din generația a şasea, 6x86, un procesor superscalar, bazat pe o superbandă de asamblare; în iunie 1997, a introdus procesorul MMX 6x86MX, iar în 1998 a apărut procesorul MII.

Procesorul Cyrix 5x86

Familia de procesoare 5x86 reprezintă o nouă generație, pe 64 de biți, compatibilă x86.

Unitatea centrală se bazează pe o bandă de asamblare cu şase niveluri, putând executa o instrucţiune într-un impuls de tact. Unitatea centrală 5x86 este divizată în următoarele blocuri funcţionale:

- unitatea pentru numere întregi (Integer Unit - IU)
- unitatea în virgulă flotantă (Floating Point Unit - FPU)
- unitatea cache (Write-Back Cache)
- unitatea pentru gestiunea memoriei (Memory Management Unit - MMU)
- unitatea de interfaţă cu magistrala (Bus Interface Unit - BIU).

Unitatea pentru numere întregi conţine:

- tamponul pentru instrucţiuni (Instruction Buffer - IB)
- unitatea de aducere a instrucţiunii (Instruction Fetch Unit - IF)
- unitatea de decodificare a instrucţiunii (Instruction Decoder Unit - ID).

Instrucţiunile sunt executate în unitatea pentru numere întregi sau în unitatea de calcul în virgulă flotantă. Cache-ul conţine cele mai recent utilizate date şi instrucţiuni şi asigură accesul rapid la aceste date din partea IU şi FPU.

Când apare o cerere de acces la o locaţie din memoria externă, MMU calculează adresa fizică pe care o trimite unităţii de interfaţă cu magistrala, care asigură interfaţarea unităţii centrale cu memoria externă şi celelalte circuite de pe placa de bază.

Unitatea pentru numere întregi citeşte, decodifică şi execută intrucţiunile într-o bandă de asamblare cu şase niveluri:

- nivelul de aducere al codului instrucţiunii (Instrucţion Fetch - IF) - citeşte din cache codul instrucţiunii următoare şi îl trimite spre decodificare nivelului următor din banda de asamblare. Se pot citi până la 128 de octeţi într-un impuls de tact;
- nivelul de decodificare a instrucţiunii (Instruction Decode - ID) - evaluează şirul de octeţi primit de la nivelul IF, determinând numărul de octeţi pentru fiecare instrucţiune şi tipul acesteia, pe care apoi le decodifică la viteza de o instrucţiune într-un impuls de tact;
- primul nivel de calcul al adresei (Address Calculation 1 - AC1) - dacă instrucţiunea are un operand în memorie, acest nivel calculează adresa de memorie liniară pentru instrucţiune;
- al doilea nivel de calcul al adresei (Address Calculation 2 - AC2) - realizează toate funcţiile de gestionare a memoriei, accesarea cache-ului şi a registrelor. Dacă detectează o instrucţiune în virgulă flotantă, aceasta este trimisă pentru execuţie unităţii în virgulă flotantă;

- nivelul de execuţie (Execution - EX) - execută instrucţiunea folosind operanzii furnizaţi de nivelurile pentru calculul adresei;
- nivelul write-back (WB) - ultimul nivel din IU, actualizează setul de registre sau trimite rezultatul unităţii de interfaţă cu memoria (Load/Store Unit) din MMU.

Unitatea cache

Procesorul Cyrix 5x86 conţine un cache unificat pentru date şi instrucţiuni de 16Kb, set-asociativ pe patru căi, organizat pe 1024 de linii. Scrierile în cache se fac prin metoda write-back. Memoria cache este organizată în patru bancuri a câte 256 linii fiecare, cu 16 octeţi pe linie. Fiecare linie cache are asociat câte un tag pe 21 de biţi şi un bit de valid (arată dacă linia conţine informaţii valide sau nu). Pe lângă aceşti biţi, fiecare linie mai conţine încă patru biţi care indică dacă conţinutul liniei a fost modificat (dirty bits), câte unul pentru fiecare dublu-cuvânt din linie. Aceşti ultimi patru biţi permit marcarea independentă a fiecărui dublu-cuvânt ca fiind modificat, în loc de a marca întreaga linie ca fiind modificată.

Unitatea de gestionare a memoriei

MMU translatează adresele liniare furnizate de IU în adrese fizice, pentru a putea fi folosite de unitatea cache şi unitatea de interfaţă cu magistrala. Mecanismul de paginare este cel standard x86.

Unitatea pentru gestionarea memoriei mai conţine un bloc (Load/Store Unit) care planifică accesele la memoria cache şi memoria externă şi implementează următoarele concepte: -reordonarea citirilor şi scrierilor - conferă o prioritate mai mare citirilor din memorie faţă de scrierile în memorie; -evitarea citirilor din memorie - elimină citirile inutile din memorie prin folosirea datelor existente deja în unitatea centrală (în cazul dependenţelor de tipul citire după scriere).

Controlul ramificaţiilor, prezicerea ramificaţiilor, dependenţele între date, unitatea în virgulă flotantă, unitatea de interfaţă cu magistrala vor fi prezentate la procesorul 6x86.

<u>Procesorul Cyrix 6x86</u>

Procesorul Cyrix 6x86 este cel mai performant dintre procesoarele de generaţia a şasea compatibile x86. Îmbunătăţirea performanţelor este realizată prin utilizarea unei arhitecturi superscalare, bazate pe o superbandă de asamblare.

Cyrix 6x86 este un procesor superscalar, deoarece conţine două benzi de asamblare separate ce permit procesarea mai multor instrucţiuni în

acelaşi timp. Folosirea unei tehnologii de procesare avansate şi creşterea numărului de niveluri în benzile de asamblare (superpipelining) permit procesorului 6x86 să atingă frecvenţe de lucru mai mari de 100MHz.
prin folosirea caracteristicilor arhitecturale unice, procesorul 6x86 elimină multe dintre dependenţele între date şi conflictele la accesarea resurselor, rezultând o performanţă optimă atât pentru programele pe 16 biţi cât şi pentru cele pe 32 de biţi.

Procesorul Cyrix 6x86 conţine două cache-uri:

- un cache unificat (pentru date şi pentru instrucţiuni) de 16Ko dual port;
- un cache de instrucţiuni de 256 octeţi.

Deoarece cache-ul unificat poate conţine instrucţiuni şi date în orice raport, acesta oferă o rată a hit-urilor (numărul de accese în cache, raportat la numărul total de accese) mai mare comparativ cu două cache-uri separate pentru date şi pentru instrucţiuni, având dimensiuni egale. O creştere a lăţimii de bandă a transferurilor cache-unitatea întreagă este realizată prin suplimentarea cache-ului unificat cu un mic cache de instrucţiuni foarte rapid, complet asociativ. Prin includerea acestui cache de instrucţiuni, se evită conflictele excesive între accesele pentru date şi pentru cod în cache-ul unificat.

Unitatea în virgulă flotantă din procesor permite executarea instrucţiunilor în virgulă flotantă în paralel cu instrucţiunile întregi. Aceasta conţine o coadă de instrucţiuni pe patru niveluri şi o coadă pentru datele scrise tot pe patru niveluri , pentru a facilita execuţia paralelă.

Procesorul 6x86 este alimentat la 3.3V ducând la un consum redus pentru toate frecvenţele de lucru. În plus, 6x86 mai posedă un mod de suspendare pe nivel scăzut, posibilitatea de a întrerupe tactul şi modul de management al sistemului (SMM) pentru aplicaţiile sensibile la alimentare.

Principalele blocuri funcţionale

Procesorul Cyrix 6x86 conţine cinci mari blocuri funcţionale:

- Unitatea întreagă (Integer Unit - IU)
- Unitatea cache (Cache Unit)
- Unitatea de gestionare a memoriei (Memory Management Unit - MMU)
- Unitatea în virgulă flotantă (Floating Point Unit - FPU)
- Unitatea de interfaţă cu magistrala (Bus Interface Unit - BIU).

Cache-ul conţine cele mai recent utilizate date şi instrucţiuni pentru a permite accese rapide la informaţii din partea IU şi FPU.

Adresele fizice sunt calculate de MMU şi sunt trimise unității cache şi unității de interfață cu magistrala. BIU oferă o interfață între placa sistem externă şi unitațile interne ale procesorului.

Unitatea întreagă

Unitatea de calcul cu numere întregi oferă o execuție paralelă a instrucțiunilor în două benzi de asamblare pentru numere întregi cu şapte niveluri. Cele două benzi de asamblare (X şi Y) pot procesa simultan câteva instrucțiuni.

Benzile de asamblare întregi conțin următoarele niveluri de prelucrare: -aducerea codului instrucțiunii (Instruction Fetch *IF) , -primul decodificator pentru instrucțiuni (Instruction Decode 1 *ID1) , -al doilea decodificator pentru instrucțiuni (Instruction Decode 2 *ID2) , -primul bloc de calculare a adresei (Address Calculation 1 *AC1) , -al doilea bloc de calculare a adresei (Address Calculation 2 *AC2) , -execuție (Execute *EX) , -writeback (WB).

Nivelul de aducere al codului instrucțiunii (IF) este împărțit de cele două benzi de asamblare, aduce câte 16 octeți de cod din unitatea cache într-un singur ciclu de tact. În acest nivel se caută orice instrucțiune de salt ce poate apare în fluxul de cod şi poate afecta secvențierea normală a programului. Dacă este detectată o instrucțiune de salt necondiționat sau una de salt condiționat, logica de prezicere a salturilor din acest nivel generează o posibilă adresă destinație pentru instrucțiunea de salt. Apoi IF aduce codul instrucțiunilor începând cu această adresă.

Funcția de decodificare a codului instrucțiunii este realizată de nivelurile ID1 şi ID2. Nivelul ID1, folosit de ambele benzi de asamblare, evaluează şirul de octeți de cod transmis de nivelul IF şi determină numărul de octeți pentru fiecare instrucțiune. Acest nivel poate trimite cel mult două instrucțiuni într-un impuls de tact nivelului ID2, câte una pentru fiecare bandă de asamblare.

Cele două niveluri ID2 decodifică instrucțiunile şi le trimite uneia din cele două benzi de asamblare X sau Y spre execuție. Banda de asamblare este aleasă bazată pe tipul instrucțiunilor aflate deja în fiecare bandă şi cât de repede se presupune că se vor termina.

Funcția de calculare a adreselor este realizată tot în două niveluri: AC1 şi AC2. Dacă instrucțiunea are o referință la un operand în memorie, AC1 calculează o adresă de memorie liniară pentru instrucțiune.

Nivelul AC2 realizează toate funcțiile de gestiunea memoriei cerute, accesele la cache şi accesele la setul de registre. Dacă AC2 detectează o

instrucțiune în virgulă flotantă, aceasta este trimisă spre prelucrare unității FPU.
În nivelul de execuție (EX), se execută instrucțiunile folosind operanzii primiți din nivelul AC2.
Nivelul writeback (WB) este ultimul din unitatea de lucru cu numere întregi. În acest nivel sunt stocate rezultatele execuției sau în registre sau în tamponul de scriere din unitatea cache.

Procesarea în inordine

Dacă o instrucțiune este executată mai repede decât instrucțiunea precedentă din cealaltă bandă de asamblare, instrucțiunile sunt completate în inordine. Toate instrucțiunile sunt prelucrate în ordine până la nivelul EX. În timp ce în nivelurile EX şi WB instrucțiunile pot fi executate în inordine.

Dacă există dependențe de date între cele două instrucțiuni, este necesară intervenția unui bloc care să asigure execuția corectă a programului. Astfel, chiar dacă instrucțiunile sunt executate în inordine, excepțiile şi scrierile din cadrul instrucțiunilor sunt întotdeauna efectuate în ordinea cerută de program.

Soluționarea dependențelor de date

Când două instrucțiuni care sunt executate în paralel accesează aceeaşi dată sau acelaşi registru, poate apare una din următoarele tipuri de dependențe de date: -citire după scriere (Read-After-Write - RAW) , -scriere după citire (Write-After-Read - WAR) , -scriere după scriere (Write-After-Write - WAW).

Dependențele între date în mod normal necesită serializarea execuției instrucțiunilor implicate. Însă, 6x86 implementează următoarele trei mecanisme ce permit execuția paralelă a instrucțiunilor ce conțin dependențe între date: -redenumirea registrelor (Register Renaming) , -înaintarea datelor (Data Forwarding) , -evitarea datelor (Data Bypassing). În continuare, se vor descrie pe scurt aceste meacnisme.

Redenumirea registrelor

Procesorul Cyrix 6x86 conține 32 registre fizice de uz general. Fiecare din cele 32 de registre din fişierul de registre poate fi desemnat a fi unul din registrele de uz general din arhitectura x86 (EAX, EBX, ECX, EDX, ESI, EDI, EBP şi ESP). Pentru fiecare operație de scriere într-un registru este selectat un nou registru fizic, pentru a se reține temporar şi data precedentă. Redenumirea registrelor elimină efectiv toate dependințele WAW şi WAR. Pentru programator este transparent acest mod de redenumire a registrelor;

este transparent atât pentru sistemul de operare, cât şi pentru programele aplicație.

Înaintarea datelor (Data Forwarding)

Doar redenumirea registrelor, nu poate elimina dependențele de tipul citire după scriere (RAW). 6x86 foloseşte două tipuri de data forwarding împreună cu redenumirea registrelor pentru a elimina acest tip de dependențe: -înaintarea operandului (operand forwarding), - apare când prima dintr-o pereche de instrucțiuni efectuează o citire din registru sau memorie iar această dată este necesară celei de-a doua instrucțiuni. CPU execută operația de citire şi furnizează data citită ambelor instrucțiuni; - înaintarea rezultatului (result forwarding) - apare atunci când prima dintr-o pereche de instrucțiuni execută o operație (cum ar fi ADD) iar rezultatul ei este citit de o a doua instrucțiune. CPU-ul execută operația primei instrucțiuni şi depune rezultatul operației în destinațiile ambelor instrucțiuni simultan.

Evitarea datelor (Data Bypassing)

Pe lângă redenumirea registrelor şi înaintarea datelor, 6x86 conține o a treia tehnică de eliminare a dependențelor de date, denumită evitarea datelor. Aceasta reduce scăderilor în performanță ale acelor dependențe de tipul RAW din memorie ce nu pot fi eliminate cu ajutorul înaintării datelor.

Evitarea datelor apare când prima dintr-o pereche de instrucțiuni scrie în memorie şi următoarea citeşte aceeaşi dată din memorie. 6x86 reține data din prima instrucțiune şi o pasează celeilalte instrucțiuni, astfel eliminându-se un ciclu de citire din memorie.

Prezicerea salturilor

Procesorul 6x86 foloseşte un tabel al adreselor destinație (Branch Target Buffer - BTB) cu 256 de intrări, set asociativ pe 4 căi, pentru menținerea adreselor destinație ale instrucțiunile de salt şi a altor informații necesare prezicerii acestor salturi. În timpul aducerii codului instrucțiunii sunt căutate instrucțiunile de salt în fluxul de instrucțiuni. Dacă este descoperită o instrucțiune de salt necondiționat, CPU-ul accesează BTB pentru a afla adresa destinație a instrucțiunii de salt. Dacă această adresă există în BTB, CPU-ul începe să aducă instrucțiunile de la noua adresă.

În cazul salturilor condiționate, BTB mai menține o serie de informații cu privire la istoricul efectuării saltului respectiv (pentru a se putea lua decizia de efectuare sau nu a saltului). Dacă instrucțiunea de salt condiționat este găsită în BTB, 6x86 începe aducerea instrucțiunilor de la

adresa prezisă. Dacă instrucţiunea nu este găsită în BTB, 6x86 prezice neexecutarea saltului şi aducerea instrucţiunilor va continua cu adresa următoare. Decizia de efectuare sau nu a saltului este luată pe baza unui algoritm de prezicere a salturilor.

Odată ce a fost adus codul unei instrucţiuni de salt condiţionat, aceasta este decodificată şi distribuită spre execuţie benzii de asamblare X. Instrucţiunea trece prin nivelurile benzii de asamblare X şi este terminată sau în nivelul EX sau în WB, în funcţie de instrucţiunea care a setat indicatorii de condiţii: -dacă instrucţiunea care a setat indicatorii de condiţii este executată în paralel cu instrucţiunea de salt condiţionat, atunci aceasta este terminată în nivelul WB, -dacă instrucţiunea care a setat indicatorii de condiţii a fost executată înaintea instrucţiunii de salt, atunci aceasta se va termina în EX.

Instrucţiunile de salt condiţionat corect prezise se vor executa într-un singur impuls de tact. Dacă după terminarea execuţiei instrucţiunii de salt condiţionat s-a detectat o prezicere eronată a saltului, CPU-ul goleşte benzile de asamblare şi începe execuţia de la adresa corectă. Procesorul 6x86 în cazul unei instrucţiuni de salt condiţionat aduce în avans atât instrucţiunea prezisă cât şi cealaltă, dar o trimite benzii de asamblare spre execuţie doar pe cea prezisă. Astfel că, în cazul unei prezicери eronate, instrucţiunea de la adresa neprezisă nu va mai fi citită din cache, deoarece a fost adusă deja. Dacă instrucţiunea de salt condiţionat a fost rezolvată în nivelul EX, atunci întârzierea în cazul unei prezicери eronate este de patru impulsuri de tact, iar dacă instrucţiunea de salt a fost rezolvată doar în WB, atunci întârzierea este de cinci impulsuri de tact.

Deoarece instrucţiunea de revenire dintr-o subrutină (RET) este dinamică, procesorul 6x86 menţine adresele pentru aceste instrucţiuni într-o stivă cu opt intrări. Adresa de revenire este introdusă în stiva adreselor de revenire de către instrucţiunea CALL, şi este scoasă de către instrucţiunea RET corespunzătoare.

Execuţia speculativă

Procesorul 6x86 are posibilitatea de a executa speculativ instrucţiunile următoare unei instrucţiuni în virgulă flotantă sau a unei instrucţiuni de salt. Execuţia speculativă permite benzilor de asamblare să execute continu instrucţiuni după un salt, fără a fi necesară blocarea benzii de asamblare până la obţinerea rezultatului execuţiei instrucţiunii de salt condiţionat. Acelaşi mecanism este folosit pentru a se executa instrucţiuni în virgulă flotantă în paralel cu instrucţiunile de numere întregi.

Procesorul are posibilitatea de execuţie în patru niveluri de speculaţie. După generarea unei noi adrese prin mecanismul de predicţie, CPU-ul salvează starea curentă (registrele, indicatorii de condiţii, etc.), incrementează numărătorul nivelului de speculaţie şi începe execuţia fluxului de instrucţiuni prezis.
Odată ce instrucţiunea de salt a fost rezolvată, CPU-ul decrementează nivelul de speculaţie. Pentru un salt corect prezis este ştearsă starea resurselor salvate la intrarea în nivelul de speculaţie curent. Pentru un salt prezis eronat, procesorul 6x86 generează adresa corectă pentru următoarea instrucţiune şi foloseşte valorile de stare salvate pentru a restaura starea curentă, într-un singur impuls de tact.

Pentru a se menţine compatibilitatea, nu sunt permise scrierile în memorie sau cache, până când nu este rezolvată instrucţiunea de salt. Execuţia speculativă continuă până când apare una din următoarele condiţii: -este decodificată o nouă instrucţiune de salt sau de calcul în virgulă flotantă şi nivelul de speculaţie este patru (maximul) , -apare o excepţie sau o eroare, -tamponul de scriere este plin, -se încearcă modificarea unei resurse a cărei stare nu a fost salvată (registrele segment, indicatorii sistem).

Cache-ul unificat de date şi instrucţiuni

Procesorul Cyrix 6x86 conţine un cache unificat şi un cache de instrucţiuni. Cache-ul unificat cu dimensiunea de 16Ko funcţionează ca un cache primar (L1) de date şi ca un cache secundar (L2) de instrucţiuni. Configurat ca un cache set-asociativ pe patru căi, conţine până la 16Ko de cod şi date în 512 linii. Cache-ul este dual-port şi permite executarea a două din operaţiile următoare în paralel: -citirea unui cod de instrucţiune, -citirea unei date (de către banda X, banda Y sau FPU) , -scrierea unei date (de către banda X, banda Y sau FPU).

Acest cache foloseşte un algoritm de replasare pseudo-LRU (Last Recently Used) şi poate fi configurat să aloce o nouă linie de cache doar la un miss de citire, sau şi la citire şi la scriere.

Cache-ul de instrucţiuni de 256 octeţi complet asociativ serveşte drept cache de instrucţiuni primar (L1). Cache-ul de instrucţiuni este încărcat din cache-ul unificat prin magistrala de date internă. Citirile codurilor de instrucţiuni din unitatea pentru numere întregi care se găsesc în cache-ul de instrucţiuni nu mai accesează cache-ul unificat. Dacă instrucţiunea nu este găsită în cache-ul de instrucţiuni, linia din cache-ul unificat care conţine instrucţiunea respectivă, este transferată atât cache-ului de instrucţiuni cât şi unităţii pentru numere întregi.

Acest cache foloseşte tot algorimtul de replasare pseudo-LRU. Pentru a se asigura operarea corectă în cazul codului automodificabil, orice scriere în cache-ul unificat este verificată cu conținutul cache-ului de instrucțiuni. Dacă a fost modificată o locație care este prezentă şi în cache-ul de instrucțiuni, atunci linia ce conține respectiva locație este dezactivată.

Unitatea de gestionare a memoriei

Unitatea de gestionare a memoriei **(Memory Management Unit - MMU)** a procesorului Cyrix 6x86, translatează adresele liniare furnizate de IU într-o adresă fizică, pentru a putea fi utilizată în continuare de cache şi interfața cu magistrala. MMU include două mecanisme de paginare, un mecanism tradițional şi un mecanism specific lui 6x86 cu pagini de dimensiuni variabile.

Mecanismul de paginare cu dimensiunea paginilor variabilă

Acest mecanism de paginare permite programelor să mapeze pagini cu dimensiunea între 4Ko şi 4Go. Folosirea paginilor de dimensiuni mari poate duce la sporirea performanței unor anumite aplicații.

Mecanismul tradițional de paginare

Mecanismul tradițional de paginare a fost îmbunătățit la 6x86 prin adăugarea unui cache pentru tabelul directorilor (Directory Table Entry - DTE) şi un TLB victimă. TLB-ul principal este cu mapare directă şi conține 128 de intrări pentru tabelul paginilor. Cache-ul DTE cu patru intrări complet asociative conține accesele cele mai recente la DTE.

TLB-ul victimă conține liniile din TLB principal care au fost înlocuite datorită unui miss în TLB. Dacă se face referirea la o pagină ce are PTE-ul în TLB-ul victimă, linia aceasta este schimbată cu o linie din TLB-ul primar.

Unitatea în virgulă flotantă

Interfața dintre unitatea în virgulă flotantă (FPU) a procesorului 6x86 şi unitatea pentru numere întregi este realizată printr-o magistrală internă pe 64 de biți. Setul de instrucțiuni FPU al procesorului 6x86 este compatibil x87 şi aderă standardului IEEE-754.

Procesorul Cyrix 6x86 execută instrucțiunile întregi în paralel cu instrucțiunile în virgulă flotantă. Instrucțiunile întregi pot fi executate în inordine cu respectarea instrucțiunilor FPU.

Aşa cum s-a mai spus, instrucțiunile FPU sunt întotdeauna executate în banda de asamblare X. Nivelul pentru calculul adresei din banda X verifică apariția excepțiilor de gestionare a memoriei şi accesează operanzii

din memorie folosiți de FPU. Dacă nu apare nici o excepție, se salvează starea curentă a procesorului în AC2 şi trimite instrucțiunea în virgulă flotantă spre execuție FPU-ului. Apoi unitatea centrală poate executa orice instrucțiune întreagă următoare, speculativ şi în inordine.

Unitatea centrală 6x86 poate trimite până la patru instrucțiuni FPU în coada de aşteptare a FPU. CPU-ul continuă cu execuția speculativă şi în inordine până când apare una dintre condițiile ce cauzează oprirea execuției speculative. Pe măsură ce FPU termină de executat o instrucțiune în virgulă flotantă, este decrementat nivelul speculativ şi sunt şterse valorile de stare salvate la începutul acestei instrucțiuni. Unitatea în virgulă flotantă mai conține şi un set de patru tampoane de scriere pentru a preveni întreruperile datorate scrierilor speculative.

Procesoarele Cyrix 6x86MX şi MII

Aceste procesoare au la bază nucleul procesorului 6x86, îmbunătățit cu cele 57 instrucțiuni multimedia noi, compatibile cu tehnologia MMX. În plus, 6x86MX şi MII lucrează la frecvențe mai mari, conțin un cache de dimensiune mai mare, un tampon destinat translatării adreselor liniare în adrese fizice (TLB) pe două niveluri şi un cache destinat adreselor de salt îmbunătățit.

Pentru a oferi suportul pentru operațiile multimedia, cache-ul poate fi transformat într-o memorie RAM scratchpad. Această memorie funcționează ca o memorie privată pentru CPUşi nu participă în operațiile cache.

3 Evoluția procesoarelor

O privire asupra dezvoltării procesoarelor de la origini până în prezent ne poate permite să caracterizăm arhitecturile microprocesoarelor contemporane şi chiar să încercăm să prevedem cum vor arată cele de mâine.

Progresele făcute de tehnologia calculatoarelor sunt absolut uluitoare; sporul de performanță se datorează unor procesoare din ce în ce mai sofisticate şi mai rapide, şi unor memorii de capacități din ce în ce mai mari.

Răspunzătoare pentru creşterea exponențială a performanței sunt însă în cea mai mare măsură microprocesoarele. Vom studia evoluției microprocesoarelor de la origini până în prezent; vom încerca să

caracterizăm arhitecturile procesoarelor contemporane şi să extrapolăm din datele la dispoziție, în privința posibilelor evoluții viitoare.

Domeniul tehnologiilor de calcul este volatil şi se mişcă cu o viteză foarte mare; orice previziune este cel puțin hazardată.

3.1 Aspecte economice

Primul microprocesor a fost creat de firma Intel în 1971. Numele său era Intel 4004, şi era un procesor pe 4 biți. Apariția primului microprocesor a fost un pas cu numeroase consecințe în evoluția ulterioară a sistemelor de calcul. Diferența între microprocesor şi metodele folosite era că procesorul strânge pe o singură pilulă de siliciu toate unitățile funcționale importante necesare executării programelor; comunicația între ele este rapidă şi eficace, permițând dintr-o dată un salt calitativ.

Nu mai puțin importantă este reducerea de cost care urmează unei astfel de integrări. Cu siguranță că principalul motiv al evoluției explozive a tehnologiei circuitelor integrate nu este de natură tehnologică, ci economică: spirala prețurilor din ce în ce mai scăzute face echipamentele de calcul din ce în ce mai accesibile, cererea creşte, ducând la venituri mai ridicate pentru fabricanți, care investesc mai mult în cercetare/dezvoltare şi linii tehnologice, obținând densități mai mari, permițând integrarea mai multor circuite precum şi costuri şi mai scăzute. Cu toată scăderea de preț, veniturile globale ale industriei semiconductoarelor au crescut foarte mult.

Pentru a menține această spirală echipamentele de calcul trebuie să mărească productivitatea muncii, direct sau indirect. Experții afirmă că acesta este doar începutul şi că în viitor fiecare individ va depinde de zeci de dispozitive de calcul în fiecare clipă. Nu suntem prea departe de acest punct: chiar în ziua de astăzi, o maşină modernă are în medie 15 microprocesoare, care controlează, reglează şi diagnostichează tot felul de parametri, de la injecție până la frâne.

3.2 Aspecte cantitative

Reducerea dimensiunilor are o consecință foarte importantă: traseele pe care trebuie să le parcurgă curentul electric între dispozitive devin mai scurte, deci se pot parcurge mai rapid. Proiectanții pot face deci procesorul să funcționeze cu un ceas mai rapid.

Observație: Majoritatea covârşitoare a procesoarelor contemporane funcționează în mod *sincron*: întreaga lor funcționare este orchestrată de un

tact de ceas, care garantează că feluritele părți sunt sincronizate. Din ce în ce mai mult însă se tinde spre scheme cu multiple semnale de ceas, sau chiar scheme asincrone.
Faptul că avem siliciu la dispoziție pentru a implementa mai mulți tranzistori înseamnă că:
1). Putem muta mai multe circuite auxiliare pe acelaşi cip. Evoluția procesoarelor cunoaşte câteva salturi calitative: când miniaturizarea făcea posibilă integrarea unui nou dispozitiv pe acelaşi circuit integrat, se realiza un salt de performanță. Astfel, au fost integrate succesiv: unități din ce în ce mai mari de procesare (8, 16, 32, acum 64 de biți), coprocesoare aritmetice, unități de management al memoriei, cache-uri de nivel 1 şi chiar 2;
2). Deşi inginerii folosesc tranzistorii suplimentari pentru a construi circuite mai sofisticate, care pot executa mai repede şi mai eficient programele. Metoda fundamentală folosită este de a face mai multe lucruri în paralel.
Împreună aceste fenomene (viteza ceasului, integrarea pe o singură pastilă şi exploatarea paralelismului) contribuie la creşterea performanței totale a procesoarelor. Aşa cum am povestit şi cu alte ocazii, măsurarea performanței unui calculator se face evaluând sistemul pe mai multe programe (deci performanța depinde foarte mult şi de compilatorul folosit), care de obicei fac parte din suite de teste standardizate (*benchmark suites*). Cele mai folosite pentru a evalua procesoare sunt cele din seria SPEC (*Standard Performance Evaluation Corporation)*.

3.3 Tehnologii arhitecturale

Profesorului John Hennessy, de la universitatea Stanforda vede două tehnologii arhitecturale ca fiind esențiale: exploatarea paralelismului la nivel de instrucțiune (*Instruction Level Parallelism*, ILP) şi ierarhii sofisticate de memorie (cache-uri).

ILP
Paralelismul la nivel de instrucțiune constă în independența instrucțiunilor din programe una de alta, ceea ce ne permite să executăm mai multe instrucțiuni simultan. Să observăm că toate procesoarele contemporane îl exploatează prin două forme:

- Execuția pe bandă de asamblare (*pipeline*) a instrucțiunilor succesive;
- Execuția în paralel a instrucțiunilor independente: procesoarele de tip VLIW (*very long instruction word*) aleg la compilare instrucțiuni

care merg în paralel, iar procesoarele superscalare fac această alegere în timpul execuției.

Astfel, în 1985 au apărut primele procesoare cu banda de asamblare, în 1990 primele procesoare de tip VLIW, iar în 1995 procesoare foarte sofisticate superscalare, care pot executa instrucțiunile în ordini foarte diferite de cea din program (*out-of-order execution*).

Cache-uri

Deşi atât procesoarele cât şi memoriile cresc constant în viteză, creşterea procesoarelor este cu 50% mai rapidă decât a memoriilor.Există o disparitate crescândă între nevoile de date (şi instrucțiuni) ale procesorului şi ceea ce memoriile pot oferi. Durata unui acces la memorie ajunge la zeci de cicli de ceas pentru procesoarele contemporane. Întârzierea accesului este şi mai exacerbată în cazul sistemelor care au mai multe procesoare, în care caz timpii de acces la date pot ajunge la mii de cicli.

Din această cauză se construiesc cache-uri, care sunt memorii mai mici şi mai rapide, care se plasează între procesor şi memoria principală, şi în care sunt aduse datele pentru prelucrare. Proiectanții au reuşit să sporească eficacitatea cache-urilor folosind două metode:

(a) *prin folosirea unor cache-uri din ce în ce mai mari, plasate din ce în ce mai aproape de procesor.* Această evoluție este clar vizibilă:

Primele procesoare nu aveau nici un fel de cache, pentru că memoriile erau suficient de rapide pentru a le servi cu date. În 1980 au apărut cache-uri (L1) sub forma unor circuite speciale, care în 1984 au fost integrate pe aceeaşi pilulă de siliciu cu procesorul central, după care (1986) a apărut un al doilea nivel de cache (L2), mai mare şi ceva mai lent, care în procesoarele moderne (1995) este la rândul lui adesea integrat cu circuitul microprocesorului, pentru a permite un acces rapid. Au apărut niveluri terțiare de cache (1999).

(b) *Pe de altă parte metodele de management ale cache-urilor sunt din ce în ce mai complicate:*

Au apărut cache-uri care servesc procesorul de îndată ce primul cuvânt a sosit, chiar dacă restul sunt pe drum (*early restart*, 1992), cache-uri care nu blochează procesorul când datele lipsesc, ci îi permit să continue execuția (*non-blocking*, 1994) şi tot felul de alte tehnologii sofisticate, pe care le-am expus în alte părți (cache-uri victimă, buffere de scriere, instrucțiuni speciale (*prefetching*) de management al cache-ului etc.). Menționăm multiprocesoarele simetrice şi protocoalele de coerență ale cache-urilor pentru astfel de sisteme; toate procesoarele moderne sunt construite pentru a fi folosite în sisteme multi-procesor, şi includ astfel de dispozitive.

3.4 Hardware şi software.

Istoria modernă a procesoarelor contrapune două paradigme pentru creşterea performanţei, bazate pe software şi respectiv pe hardware. Există o simbioză totală între hardware şi software în aceste timpuri ale noastre. Procesoarele se proiectează odată cu compilatoarele care le folosesc; relaţia dintre ele fiind strânsă: compilatorul trebuie să genereze cod care să exploateze caracteristicile arhitecturale, altfel codul generat nu va fi folosit.

Metodele de creştere a performanţei cu ajutorul compilatoarelor se numesc şi *statice*, pentru că programul este analizat şi optimizat o singură dată, înainte de a fi pornit în execuţie. Metodele bazate pe hardware se numesc *dinamice*, pentru că sunt aplicate în timp ce programul se execută.

În anii '80 a apărut ideea de a face procesoarele mai simple pentru a le permite să meargă mai repede. Astfel de arhitecturi au fost numite RISC: *Reduced Instruction Set Computer*, în contrast cu celelalte, *Complex ISC*.

Există lucruri care se pot face numai static şi există lucruri care se pot face numai dinamic.

La ora actuală distincţia RISC/CISC aproape că s-a estompat. De exemplu, Pentium, un procesor tipic CISC, traduce în mod automat instrucţiunile în instrucţiuni de tip RISC în hardware, după care le execută. Pe de altă parte, toate procesoarele RISC au în plus extensii la setul de instrucţiuni (gen CISC) pentru a le mări performanţa; toate procesoarele au extensii speciale pentru multimedia.

Graniţele dintre super-scalar şi VLIW tind să se estompeze; cu siguranţă un model mixt este preferabil, pentru că poate lua ce e mai bun din fiecare tehnologie.

Crusoe

Anul acesta compania Transmeta a anunţat apariţia unui nou procesor, numit Crusoe, care exploatează la maximum tehnologiile statice (compilarea). Compania Transmeta lansează procesorul care poate simula alte procesoare, inclusiv cele ale firmei Intel (având angajt pe cel mai bun programator al planetei, Linus Torvalds, creatorul sistemului de operare Linux).

Transmeta a lansat Crusoe în luna ianuarie; compania promovează din nou întoarcerea la simplitate (care a fost sugerată atât de curentul RISC, cât şi de modelele VLIW. Echipa care a lucrat la Transmeta este compusă în ingineri care au lucrat la IBM.

Proprietăţile lui Crusoe nu sunt destul de clare:

- cipul nu are un ceas mai rapid decât procesoarele Intel (versiunile de Crusoe disponibile acum merg doar la 400Mhz, comparativ cu Pentium, care ajunge la 800);
- cipul consumă mai puțină energie şi are nevoie de mai puțină răcire. Transmeta afirmă că aceasta îl face ideal pentru laptop-uri. Din păcate, principalul consumator de energie într-un laptop nu este procesorul, ci ecranul şi discul; astfel avantajele noului cip vor fi marginale;

Accesul la memorie

In ultimii 10 ani viteza memoriilor a crescut cu 10% pe an, în timp ce viteza procesoarelor a crescut cu o rata de 60%. Se concluzionează că această disparitate va continua să se accentueze iar prețul relativ al unui acces la memorie (măsurat în cicli de ceas) va continua să crească.

Putere

Un alt factor care limitează evoluția circuitelor integrate este consumul de putere; în urmă cu 15 ani un procesor consuma 2 wați; astăzi un procesor ca Alpha 80364 consumă 100W; de aici rezultă limitări pentru ceas (puterea consumată creşte cu frecvența ceasului), şi necesitatea unor dispozitive speciale de răcire; dar tehnologia lucrează în direcția favorabilă: miniaturizarea duce la scăderea puterii necesare. Un alt factor care duce la scăderea puterii consumate este scăderea tensiunilor de alimentare; deşi dimensiunea tranzistorilor a scăzut continuu, dimensiunile circuitelor fabricate au crescut, iar puterea consumată a crescut şi ea.

Complexitate

Procesoarele cele mai moderne au peste 25 de milioane de tranzistoare. Astfel de circuite sunt foarte greu de verificat şi testat. În prezent o companie ca Intel cheltuieşte 40% din buget pentru proiectare şi dezvoltare, şi 60% pentru verificare şi testare!

O altă problemă importantă este legată de liniile tehnologice de fabricație: o astfel de instalație costă în prezent două miliarde de dolari. Puține companii îşi pot permite investiții de o asemenea valoare pentru o tehnologie care se schimbă în 3 ani!

Sârmele

Miniaturizarea nu va putea continua în acelaşi ritm exponențial: peste ceva vreme am ajunge la necesitatea de a face un tranzistor mai mic decât un

atom, ceea ce e evident imposibil. Dar înainte de a atinge un astfel de prag, vor apărea cu siguranţă alte probleme.

O analiză interesantă a fost făcută de Mark Horowitz, profesor la universitatea Stanford, într-un articol intitulat "Viitorul sârmelor". Articolul porneşte de la caracteristicile electrice ale semiconductorilor şi analizează o serie de scenarii posibile pentru tehnologiile de fabricaţie. Textul ia în considerare o mulţime de factori, cum ar fi geometria sârmelor, capacităţi şi rezistenţe, disiparea puterii etc.

Autorul observă că în general sârmele vor evolua în sensul dorit: vor deveni mai scurte, iar viteza de transmisiune a informaţiei nu va scădea, relativ la dimensiunea circuitului. Dacă am lua un microprocesor de astăzi şi l-am reduce la scară, sârmele nu ar constitui un impediment în funcţionarea sa corectă.

Problema apare însă din faptul că suprafaţa circuitelor nu scade; o mare problemă sunt sârmele care *traversează mai multe module*. Lungimea acestora rămâne constantă în milimetri, dar cum viteza ceasului creşte mereu, aceasta înseamnă că semnalele electrice *nu mai au timp* să parcurgă sârmele de la un capăt la altul. La 1Ghz, lumina străbate în vid 30 de centimetri. Dar viteza luminii în solide este mai mică iar viteza de propagare scade semnificativ odată cu numărul de "consumatori" ai sârmei (adică o sârmă conectată la 3 circuite e mult mai lentă decât una cuplată la doar două). De asemenea, liniile lungi de transmisiune vor avea nevoie de amplificatoare, care încetinesc substanţial semnalul;va rezulta că circuitele viitorului *nu vor mai putea comunica prin semnale globale*: va fi imposibil pentru o sârmă să unească diferitele părţi ale circuitului. Aceasta este o consecinţă de cea mai mare importanţă pentru arhitecturile viitoare!

Zgomotul

Pe măsură ce tranzistorii sunt mai mici, sârmele sunt mai subţiri şi consumul de putere este mai mare, circuitele sunt mai sensibile la zgomot, fie el termic, din mediu (de exemplu radiaţii cosmice) sau, în curând, chiar efecte cuantice! Fenomenele de transport din semiconductori pe care se bazează tranzistorul sunt fenomene statistice: dar dimensiunile devin atât de mici încât numai câţiva electroni produc semnalele, statistica nu mai operează iar excepţiile vor începe să apară.

3.5 Generația următoare

Câteva tendințe:

Trace cache: este un cache pentru instrucțiuni care, în loc de a păstra instrucțiunile în ordinea adreselor lor, le menține în ordinea în care este probabil să fie executate.

Execuția speculativă şi predicția valorilor: principala limitare în calea execuției paralele a instrucțiunilor sunt *dependențele* între instrucțiuni. Dacă prima instrucțiune durează mult, atunci a doua nu se poate executa. Soluția folosită este de *a prezice* valoarea rezultată şi de a executa şi instrucțiunea dependentă. Când rezultatul primei instrucțiuni soseşte, este comparat cu cel prezis; dacă predicția a fost corectă, toate sunt bune, altfel instrucțiunea dependentă este re-executată. Există felurite forme de predicție a valorilor, unele folosite deja de multă vreme (cum ar fi predicția salturilor, care este descrisă în PC Report -din august 1999.

Execuția predicată, care este folosită de procesoare de prelucrare de semnal ca Texas Instruments C6X, şi care va fi una din trăsăturile fundamentale ale noii arhitecturi de la Intel, Merced. Execuția predicată evită execuția instrucțiunilor de salt (de exemplu, când avem o structură de genul if-then-else, o arhitectură predicată poate executa ambele ramuri ale condiției).

Multi-procesoare

O evoluție naturală este de a face saltul de la mai multe procesoare legate printr-o magistrală comună (ca în cazul sistemelor cu multiprocesoare simetrice) în procesoare strâns cuplate, pe aceeaşi pilulă de siliciu.

Astfel de scheme există deja: procesorul pentru mainframes IBM S/390 are două nuclee identice, care execută sincron acelaşi program: în cazul în care rezultatele nu sunt identice se execută o excepție şi programul este reluat. Acesta este un exemplu în care mai multe resurse sunt folosite pentru o fiabilitate sporită, dar IBM a anunțat că viitorul lor procesor G5 va conține două nuclee independente pe aceeaşi pilulă, permițând realizarea unor sisteme multi-procesor cu un singur cip.

Multithreading

O evoluție naturală ar fi spre a exploata alte forme de paralelism decât cel la nivel de instrucțiune (ILP).Calculatoarele moderne exploatează paralelismul la nivel de proces, dar există forme intermediare, şi s-ar putea să vedem arhitecturi din ce în ce mai orientate spre acestea:

- Paralelism la nivel de buclă: în care iterații succesive ale unei bucle sunt executate în paralel.
- Paralelism la nivel de *thread*;

Există o mulțime de inovații arhitecturale legate de aceste tehnologii, încă ne-integrate în produse comerciale:

- *Thread-level data speculation*: este o metodă de a implementa paralelismul la nivel de buclă, lansând câte un thread pentru fiecare iterație a buclei. De exemplu, proiectul STAMPede de la Universitatea Carnegie Mellon, condus de profesorul Todd Mowry explorează această alternativă (http://www.cs.cmu.edu/~tcm/ STAMPede.html).
- *Simultaneous multi-threading*, propus la universitatea din Seattle în 1995. Această tehnologie menține starea fiecărui thread în hardware şi permite comutarea rapidă între thread-uri. Se pot distinge două variante, ca în caseta "Multithreading": una din variante- în fiecare ciclu de ceas putem executa instrucțiuni dintr-un alt thread, şi alta- în fiecare ciclu, instrucțiunile din thread-uri diferite candidează pentru unități funcționale diferite.

Avantajul unor astfel de scheme este că, dacă un thread execută operații care au nevoie de mult timp (de exemplu accese la memorie), punem alte thread-uri care sunt gata de execuție, folosind mai eficient unitățile funcționale ale procesorului.

Următoarele proiecte de cercetare încearcă să privească nu numai în viitorul imediat, ci să anticipeze peisajul calculatoarelor peste zece ani şi mai mult. La acea dată barierele impuse de fizică vor fi atinse, astfel va trebui să ne aşteptăm la o încetinire a vertiginoasei creşteri de performanță.

IRAM, Smart Memory

Proiectul IRAM (*Intelligent RAM*) de la Berkeley este condus de David Patterson (http://iram.cs.berkeley.edu/) şi îşi propune integrarea tehnologii-lor de fabricație a memoriilor şi procesoarelor (la ora actuală liniile de fabricație sunt complet diferite).

Un proiect foarte asemănător este cel de la Stanford, al profesorului Mark Horowitz, numit Smart Memories
(http://velox. stanford.edu/smart_memories/).

Aceste proiecte încearcă să depăşească problema accesului lent la memorie prin distribuirea unităților de procesare printre memorii, astfel încât accesul să fie paralel şi rapid.

RAW

Maşina RAW constă din multe procesoare, relativ simple, construite pe aceeaşi pilulă de siliciu. Aceste procesoare cooperează pentru a executa o singură aplicație, care este paralelizată automat de compilator.

Imagine

Un alt proiect interesant este *Imagine*(http://cva.stanford.edu /imagine/cva_imagine.html), dezvoltat la universitatea Stanford sub conducerea lui William Dally. Proiectul propune un nou model de programare, orientat spre multimedia, în care paralelismul datelor este explicat prin noțiunea de flux (*stream*). De exemplu, pentru a afişa scene mai complicate pe ecran, prelucrarea transformă datele dintr-un flux de obiecte într-un flux de poligoane, care devin un flux de triunghiuri, apoi un flux de pixeli etc.

Bibliografie:

[Bar02] **Barry B. Brey**, *Intel Microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium III, and Pentium IV: Architecture, Programming, and Interfacing*, Pearson Education, 2002

[Sin02] **Naresh K. Sinha**, *Microprocessor-Based Control Systems*, Kluwer Academic Publishers, 2002

[Hea02] **Steve Heath**, *Embedded Systems Design*, Elsevier Science, 2002

Berkeley University Home Page, http://bwrc.eecs.berkeley.edu/CIC/