Evoluția arhitecturilor calculatoarelor (1)

Author: Petre IORDANESCU, Date: August 2020
Categories: Software Design & Architectures

PRIMII PASI

De la abac și până la computerele modern de azi este o cale dar nu așa de lungă... Sigur că dacă privim cu mult în urmă, între abac și prima arhitectură este o cale considerabilă în timp. Prima arhitectură a fost definită în anii 1800 de către Charles Babbage[1], matematician, filozof, inventator și inginer mecanic. Babbage a gindit o mașină de calcul (pe care a numit-o „Analytical Engine[2]") a cărei arhitectură de bază cuprindea trei componente: MOARA, COMANDA si MAGAZIA. Cele trei componente sunt echivalentul actual al ALU, CPU și memoria. Și cam așa arată lucrurile și astazi...

URMAREA

În timp au fost aduse diverse detalii acestei arhitecturi, menite să îmbunătățească performanța. Prima și cea mai folosită idee de creștere a performanței a fost cea a creșterii frecvenței ceasului de comandă. Sigur, pare simplu, dar creșterea frecvenței atrage după sine și este limitată de o multitudine de probleme, precum: capacitatea dintre două trasee paralele, instabilitatea la schimbări minore în mediu— temperatura, umiditatea — dificultatea de a genera o frecvență ridicată constantă și stabilă, etc.

Astfel au apărut idei mai bune decât creșterea „brută" a performanței, idei bazate pe lucrul în paralel sau cvasi-paralel și bine definit (în bandă de asamblare[3], concept pus la punct de Ford[4] în fabrica sa de automobile). Toate ideile noi converg pe creșterea performanței prin conceptul simplu „dacă un om sapa un șanț într-o zi atunci doi oameni vor săpa același șanț în jumătate de zi". Cu alte cuvinte, creșterea performanței este de fapt o creștere a productivității. O altă idee utilă a fost crearea unui paralelism real (nu cvasi pri creșterea productivității) la nivelul procesorului (nu prin utilizarea de mai multe procesoare sau core-uri în practica curentă), paralelism care să fie complet transparent sistemului de operare care nu avea nevoie de informația că procesorul are două unități aritmetice pentru întregi și nici una pentru virgulă mobilă. Soluția era folosită la scara larga „dar în jos" pentru unitățile de virgulă mobilă care erau componente scumpe și de aceea au fost făcute opționale (coprocesoare matematice), în lipsă fiind emulate software.

Astfel de la primele implementări, ideea măririi numărului de unități de procesare aritmetică părea atractivă, ideea creșterii productivității neavând inca nici o „viziune" aplicabilă. Crearea unui număr mai mare decât unu de unități de procesare aritmetică atrăgea după sine o serie de probleme din care cea mai mare nu era hardware ci era la nivel de logică de funcționare coerenta a procesorului. Practic gândiți-vă ca aveți de efectuat două lucruri într-o ordine obligatorie, T1 și apoi T2. Apoi că aveți doi muncitori (M1 și M2) si oricare dintre ei au cunoștințele pentru a face atât T1 cât și T2. Și le dați celor doi muncitori, în același moment câte unul din cele două lucruri pentru a fi făcute și lui M1 îi dați T1 ir lui M2 îi dați T2 pentru că în această ordine v-au venit cererile și de aceea T1 trebuie terminat înaintea lui T2. Daca M2 lucrează mai repede din diverse motive, veți avea surpriza să vedeți ca T2 s-a terminat înaintea lui T1 ceea ce este puțin neplăcut. Va trebui găsită o metodă ori de a-l întârzia pe M2 ceea ce nu veți dori și nici el nu va dori pentru că și-a făcut socoteala că pleacă mai devreme ori de de a-l face pe M1 să lucreze în același ritm cu M2 ceea ce va fi foarte greu și riscant.

Soluția a gasit-o Robert Tomasulo[5], inginer la IBM (astazi este un algoritm bine cunoscut sub numele de „Tomasulo algorithm[6]". Și a implementat-o prima oară pe mașina IBM 360/91[7] (vezi panoul frontal in imagine). Ideea a plecat de la faptul că instrucțiunile aduse din memorie pot fi executate de către diverse unități ale procesorului și din start vor fi plasate într-o coadă în ordinea în care ar trebui să se termine. Tomasulo a spus că vor aștepta chiar dacă se termină însa nu „era rândul lor" și ÎȘI VOR PRODUCE EFECTELE DOAR CÂND LE VA VENI RÂNDUL. Aceasta înseamnă predictibilitate pentru programatori care altfel nu ar putea lucra. Astfel, logic, au „apărut" două cozi: „issue queue" în care instrucțiunile așteaptă să le vină rândul în a-și produce efectele după terminare iar cea de-a doua coadă numită „reorder queue" în care instrucțiunile sunt mutate în ordinea terminării și așteaptă să își producă efectele în ordinea în care trebuie (gândiți că o o instrucțiune terminată își „ține" efectele în reorder queue până cind îi vine rândul). Astfel a fost creata prima arhitectura „out of order execution" adică instrucțiunile nu se execută neapărat în ordinea sosirii, dar efectele lor sunt ca și cum s-ar executa în acea ordine.

O arhitectură modernă și mai complexă ce a putut fi implementată peste ceva ani de alți producători (Intel de exemplu abia cu seria 586[8]). O arhitectură care a dus la creșterea semnificativă a performanței prin utilizarea de unități ce lucrează în paralel (procesor catalogat ca n-scalar), larg folosită și azi practic în toate procesoarele de uz general, RISC, CISC, ARM, etc. ca idee, telefoanele smart (si nu numai) de astăzi au astfel de procesoare.

RECOMANDARI

Personal recomand, mai ales celor ce doresc sa afle mai multe despre istoria si evolutia calculatoarelor si a arhitecturilor, citirea tuturor articolelor si materialelor referențiate in notele de subsol, ele fiind un material bibliografic inestimabil si care pune in evidenta modul si evolutia gindirii oamenilor.

RENware team si Petre IORDANESCU, 2020

(mai multe articole gasiti pe: http://www.renware.eu/articles)

[1] https://en.wikipedia.org/wiki/Charles_Babbage
[2] https://en.wikipedia.org/wiki/Analytical_Engine
[3] https://en.wikipedia.org/wiki/Assembly_line
[4] http://history.alberta.ca/energyheritage/oil/early-industrialization-and-exploration-1776-1920/scientific-understanding-innovation-and-industrialization/henry-ford-and-the-assembly-line.aspx, https://www.history.com/this-day-in-history/fords-assembly-line-starts-rolling
[5] https://en.wikipedia.org/wiki/Robert_Tomasulo
[6] https://en.wikipedia.org/wiki/Tomasulo_algorithmhttps://en.wikipedia.org/wiki/Tomasulo_algorithmhttps://en.wikipedia.org/wiki/Tomasulo_algorithm
[7] https://en.wikipedia.org/wiki/IBM_System/360_Model_91
[8] https://en.wikipedia.org/wiki/Pentium