# PLANIFICAREA LUCRĂRILOR DE LABORATOR LA DISCIPLINA "SISTEME CU MICROPROCESOARE"

- http://webspace.ulbsibiu.ro/adrian.florea/html/simulatoare/Planif_OPM.pdf -

**L1. INTRODUCERE.** Explicitare noţiuni: microarhitectură, ISA, cache, simulator, metodologii de simulare. Avantajele utilizării simulatoarelor în studiul microarhitecturilor de procesare pentru determinarea efectivă a celei optime din punct de vedere cost / performanţă, dar şi pentru înţelegerea principiilor fundamentale specifice arhitecturilor RISC de calcul, organizării memoriei, prin prisma unor aplicaţii scrise în limbaje de asamblare.

Prezentare generală a conţinutului laboratorului structurat pe simularea şi optimizarea arhitecturilor RISC scalare (MIPS, DLX) folosind simulatoarele *execution-driven* **SPIMSal**, **WinDLX** şi respectiv a unei maşini cu execuţie multiplă (procesor VLIW) folosind simulatorul **VLIW-DLX**. Evidenţierea conceptelor legate cache-uri – modul de organizare, regulile de mapare, algoritmii de înlocuire a blocurilor conflictuale, strategia de scriere – folosind simulatorul **PCspim-cache.** Utilizarea simulatorului *hibrid* **SATSim** aferent arhitecturii PowerPC pentru vizualizarea aspectelor arhitecturale specifice procesoarelor RISC superscalare (staţii de rezervare, unităţi funcţionale de execuţie, buffer de reordonare şi buffer de redenumire). Varianta îmbunătăţită **PSATSim** evidenţiază suplimentar efectul negativ al predicţiei greşite a instrucţiunilor de ramificaţie atât asupra vitezei de execuţie a procesorului cât şi asupra energiei consumate.

(- http://webspace.ulbsibiu.ro/adrian.florea/html/simulatoare/Lab_1_OPM.pdf -)

**L2. a) ARHITECTURA MICROPROCESOARELOR MIPS R2000/R3000 (MIPS – Microprocessor without Interlocked Pipeline Stages)**: schema bloc, regiştrii procesorului MIPS R2000, modurile de adresare, sintaxa asamblor, utilizarea memoriei şi convenţii de apel, formatul instrucţiunilor şi setul de instrucţiuni al procesorului MIPS R2000.

**b)** Aplicaţie: program de calcul a sumei de *n* numere întregi aflate în memorie la adresa 0x10010000. Afişare sumă şi scriere în memorie la adresa 0x10010030. Calcul medie aritmetică, afişare cât şi rest.

**L3. a) UTILIZAREA SIMULATORULUI SPIM (**http://webspace.ulbsibiu.ro/adrian.florea/html/simulatoare/spimsal.zip**):** apeluri sistem, pseudo-instrucţiuni folosite în limbaj de asamblare MIPS, ghid de utilizare al simulatorului; afişare mesaj pe consolă, citire mesaj de la tastatură; citire întreg de la tastatură şi verificare dacă este sau nu număr prim.

**b) Aplicaţii:**

**i)** Afişarea la consolă a primelor *n* numere prime [(iniţial $n = 24$, ulterior $n = 25 \times 24$); (parametrul *n* se dă de la tastatură)].

**ii)** Scrieţi un program care afişează primele *n* perechi de numere prime impare consecutive (n - număr natural citit de la tastatură). Exemplu: (3,5), (5,7), etc.

**iii)** Realizaţi un program care afişează divizorii unui număr *n* (n - număr natural citit de la tastatură) în două variante. Exemplu: pentru n=12 se va afişa – 2 2 3 1 şi respectiv $2^2$ $3^1$ $1^1$.

**iv)** Realizaţi un program care afişează toate numerele extra-prime de două şi de trei cifre. Un număr este extra-prim dacă el este prim şi toate numerele obţinute din permutări ale cifrelor sale sunt prime.

## L4. INVESTIGAŢII ARHITECTURALE UTILIZÂND SIMULATORUL SPIMSal:

**a)** Să se citească un şir de numere întregi de la tastatură, a cărui dimensiune este citită tot de la tastatură. Sortaţi şirul prin metoda "*bubblesort*", memoraţi succesiv datele la adresa 0x10012000 şi afişaţi şirul sortat pe consolă.

**b)** Realizaţi un program care citeşte de la tastatură două numere naturale *n* şi *k* (n>k) şi calculează şi afişează pe consolă valorile următoare:

$$C_n^k \text{ şi } A_n^k$$

**c)** Scrieţi un program care citeşte de la tastatură un număr natural *n* şi verifică dacă este perfect (este egal cu suma divizorilor săi strict mai mici decât numărul).

L5. EVIDENŢIEREA CONCEPTELOR LEGATE DE CACHE-URI – MODUL DE ORGANIZARE, REGULILE DE MAPARE, ALGORITMII DE ÎNLOCUIRE A BLOCURILOR CONFLICTUALE, STRATEGIA DE SCRIERE – FOLOSIND SIMULATORUL PCSPIM-CACHE

**a)** Să se verifice execuţia tuturor programelor implementate în limbaj de asamblare MIPS pe parcursul lucrărilor L2÷L4. Adaptaţi codul sursă pentru buna funcţionare pe noul simulator.

**b)** Analizaţi influenţa unor opţiuni de simulare de tipul (*Delayed Branches, Delayed Load* şi *Cache Setings*) asupra execuţiei programelor anterioare.

**c)** Se consideră următorul programul de test **g_cache.s** care realizează în paralel suma a câte 8 elemente – 1,1,1,1,2,2,2,2 stocate în memorie la adresa 0x10000480 respectiv a altor 8 elemente 3,3,3,3,4,4,4,4 de la adresa 0x10000500. Să se ruleze **pas cu pas** secvenţa de cod şi să se vizualizeze dinamic modificările de stare ale memorie cache şi realizaţi o statistică privind **impactul asupra ratei de hit în cache-uri** a următorilor factori: – dimensiunea blocului de date, gradul de asociativitate, algoritmul de înlocuire, strategia de scriere. Implementarea altor programe de test.

**d)** Se consideră următorul programul de test **cache_loop.s** care realizează în paralel suma elementelor unei matrici Array_A[10][20] mai întâi toate elementele unei coloane. Se aplica metoda *loop interchange* (insumarea intai a tuturor elementelor de pe o linie – cod sursa **cache_loop_i.s**) si se vizualizeaza comparativ rata de hit in cache.

## L6. SEMINAR DE APLICAŢII. (I)

## L7. ARHITECTURA MICROPROCESOARELOR DLX.

Regiştrii microprocesorului DLX. Setul optimizat de instrucţiuni, diferenţe MIPS – DLX (instrucţiuni pentru rest, instrucţiuni de transfer şi de salt condiţionat). Implementarea non-pipeline respectiv pipeline (problema 24, 29 (a, b) şi 50 cap.11 – SOAC).

## L8. a) UTILIZAREA SIMULATORULUI DLX

(http://webspace.ulbsibiu.ro/adrian.florea/html/simulatoare/dlx.zip): configurarea WinDLX, încărcarea programelor de test, simularea benchmark-urilor, sistemul de ferestre.

**b)** Rezultate cantitative bazate pe simularea benchmark-urilor *Fact.s*, *Invers.s*, rutina *Input.s*. Evidenţierea efectului defavorabil al **hazardurilor RAW** în cadrul arhitecturilor pipeline (super)scalare asupra performanţei de procesare. Modificarea programului de test fact.s şi simularea hazardurilor structurale şi a conflictelor de nume (WAW). Vizualizarea tehnicii de **forwarding** – câştigul de performanţă obţinut.

**c)** Aplicaţie: scrieţi un program care citeşte un număr întreg de la tastatură prin intermediul modulului **Input.s** (vezi lucrarea *Investigaţii Arhitecturale Utilizând Simulatorul DLX*) şi

calculează maximul, minimul şi suma cifrelor numărului şi le depune succesiv în memoria DLX la adresa 0x1500.

**L9. a) INVESTIGAŢII ARHITECTURALE UTILIZÂND SIMULATORUL *DLX*.** Probleme propuse spre rezolvare – conjectura lui Goldbach, perechi de numere prime impar consecutive, aplicaţii de sortare cu şiruri de numere.

**b)** UTILIZAREA SIMULATORULUI VLIW-DLX**:** ilustrarea principiilor fundamentale ale procesării VLIW (very long instruction word). Rolul software-lui în detecţia şi eliminarea hazardurilor RAW. Relaţia dintre instrucţiunea multiplă şi instrucţiunile RISC primitive şi independente, care vor fi alocate unităţilor de execuţie în conformitate strictă cu poziţia lor în instrucţiunea multiplă (număr/latenţe). Optimizări software in procesoarele VLIW (loop unrolling, software pipelining). Avantaje / Dezavantaje faţă de procesoarele superscalare. Aplicabilitate – sisteme dedicate (procesoare de semnal, procesoare multimedia). Aplicaţie rezolvată – execuţia pe un procesor VLIW a unui program care înmulţeşte două matrici pătratice.

**L10. *SATSIM*: SIMULAREA ARHITECTURILOR SUPERSCALARE FOLOSIND ANIMAŢIE INTERACTIVĂ** (http://webspace.ulbsibiu.ro/adrian.florea/html/simulatoare/SATSimG.ZIP)**.** Înţelegerea conceptelor legate de procesarea *pipeline* a instrucţiunilor, execuţia *out of order*, impactul negativ asupra performanţei al predicţiei greşite a branch-urilor şi al miss-urilor în cache. Evidenţierea, prin simulare la nivel de execuţie – pas cu pas, a aspectelor arhitecturale specifice procesoarelor RISC superscalare (staţii de rezervare, unităţi funcţionale de execuţie, buffer de reordonare şi buffer de redenumire).

L11. PSATSIM: INSTRUMENT SOFTWARE DE EVALUARE A PERFORMANŢEI DE PROCESARE ŞI A CONSUMULUI DE PUTERE ÎN MICROARHITECTURILE SUPERSCALARE.

**a) Modelarea consumului de putere** (CACTI, WATTCH). Parametrii tehnologici, clasificarea consumului de putere în funcţie de tipul resursei arhitecturale (*structuri matriceale* – memoria cache, decodificatoare de adresă, setul de regiştrii generali, predictorul de salturi; *memorii adresabile după conţinut* – asociative – staţiile de rezervare, logica de reactivare a instrucţiunilor situate în buffer-ul de reordonare în aşteptarea efectuării fazei de commit; *circuite logice combinaţionale* – implementarea unităţilor funcţionale de execuţie (ALU), logica de verificare a dependenţelor de date, selecţia instrucţiunilor spre staţiile de rezervare şi unităţile funcţionale). **Influenţa caracteristicilor arhitecturale specifice procesoarelor RISC superscalare asupra consumului de putere şi respectiv asupra performanţei**.

**b) Predictorul de salturi** – una din principalele cauze care determină consum ridicat de putere în arhitecturi. **Evidenţierea efectului negativ al predicţiei greşite a instrucţiunilor de ramificaţie** (parametrizarea acurateţii de predicţie) **atât asupra vitezei de execuţie a procesorului cât şi asupra energiei consumate**.

**L12**. Seminar de aplicaţii. (**II**)

**L13**. Evaluare practică privind materia studiată la laborator.