

PLANIFICAREA LUCRĂRILOR DE LABORATOR LA DISCIPLINA “SISTEME CU MICROPROCESOARE”

- http://webspaces.ulbsibiu.ro/adrian.florea/html/simulatoare/Planif_OP_M.pdf -

L1. INTRODUCERE. Explicare noțiuni: [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 / performanță, dar și pentru înțelegerea principiilor fundamentale specifice arhitecturilor RISC de calcul, organizării memoriei, prin prisma unor aplicații scrise în limbaje de asamblare.

Prezentare generală a conținutului laboratorului structurat pe simularea și optimizarea arhitecturilor RISC scalare (MIPS, DLX) folosind simulatoarele *execution-driven* **SPIMSal**, **WinDLX** și respectiv a unei mașini cu execuție multiplă (procesor VLIW) folosind simulatorul **VLIW-DLX**. Evidențierea conceptelor legate cache-uri – modul de organizare, regulile de mapare, algoritmi 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 (stații de rezervare, unități funcționale de execuție, buffer de reordonare și buffer de redenumire). Varianta îmbunătățită **PSATSim** evidențiază suplimentar efectul negativ al predicției greșite a instrucțiunilor de ramificație atât asupra vitezei de execuție a procesorului cât și asupra energiei consumate.

(- http://webspaces.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, regiștrii procesorului MIPS R2000, modurile de adresare, sintaxa asambler, utilizarea memoriei și convenții de apel, formatul instrucțiunilor și setul de instrucțiuni al procesorului MIPS R2000.

b) Aplicație: program de calcul a sumei de n numere întregi aflate în memorie la adresa 0x10010000. Afișare sumă și scriere în memorie la adresa 0x10010030. Calcul medie aritmetică, afișare cât și rest.

L3. a) UTILIZAREA SIMULATORULUI SPIM

(<http://webspaces.ulbsibiu.ro/adrian.florea/html/simulatoare/spimsal.zip>): apeluri sistem, pseudo-instrucțiuni folosite în limbaj de asamblare MIPS, ghid de utilizare al simulatorului; afișare mesaj pe consolă, citire mesaj de la tastatură; citire întreg de la tastatură și verificare dacă este sau nu număr prim.

b) Aplicații:

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

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

iii) Realizați un program care afișează divizorii unui număr n (n - număr natural citit de la tastatură) în două variante. Exemplu: pentru $n=12$ se va afișa – 2 2 3 1 și respectiv $2^2 3^1 1^1$.

iv) Realizați un program care afișează toate numerele extra-prime de două și de trei cifre. Un număr este extra-prim dacă el este prim și toate numerele obținute din permutări ale cifrelor sale sunt prime.

L4. INVESTIGAȚII ARHITECTURALE UTILIZÂND SIMULATORUL SPIMSal:

a) Să se citească un șir de numere întregi de la tastatură, a cărui dimensiune este citită tot de la tastatură. Sortați șirul prin metoda “*bubblesort*”, memorați succesiv datele la adresa 0x10012000 și afișați șirul sortat pe consolă.

b) Realizați un program care citește de la tastatură două numere naturale n și k ($n > k$) și calculează și afișează pe consolă valorile următoare:

$$C_n^k \text{ și } A_n^k$$

c) Scrieți un program care citește de la tastatură un număr natural n și verifică dacă este perfect (este egal cu suma divizorilor săi strict mai mici decât numărul).

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

a) Să se verifice execuția tuturor programelor implementate în limbaj de asamblare MIPS pe parcursul lucrărilor L2÷L4. Adaptați codul sursă pentru buna funcționare pe noul simulator.

b) Analizați influența unor opțiuni de simulare de tipul (*Delayed Branches*, *Delayed Load* și *Cache Settings*) asupra execuției 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** secvența de cod și să se vizualizeze dinamic modificările de stare ale memorie cache și realizați 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 întâi a tuturor elementelor de pe o linie – cod sursa **cache_loop_i.s**) și se vizualizează comparativ rata de hit în cache.

L6. SEMINAR DE APLICAȚII. (I)

L7. ARHITECTURA MICROPROCESOARELOR DLX. Regiștrii microprocesorului DLX. Setul optimizat de instrucțiuni, diferențe MIPS – DLX (instrucțiuni pentru rest, instrucțiuni de transfer și de salt condiționat). Implementarea non-pipeline respectiv pipeline (problema 24, 29 (a, b) și 50 cap.11 – SOAC).

L8. a) UTILIZAREA SIMULATORULUI DLX

(<http://webspacespace.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*. Evidențierea efectului defavorabil al **hazardurilor RAW** în cadrul arhitecturilor pipeline (super)scalare asupra performanței de procesare. Modificarea programului de test *fact.s* și

simularea hazardurilor structurale și a conflictelor de nume (WAW). Vizualizarea tehnicii de **forwarding** – câștigul de performanță obținut.

c) Aplicație: scrieți un program care citește un număr întreg de la tastatură prin intermediul modulului **Input.s** (vezi lucrarea *Investigații Arhitecturale Utilizând Simulatorul DLX*) și calculează maximul, minimul și suma cifrelor numărului și le depune succesiv în memoria DLX la adresa 0x1500.

L9. a) INVESTIGAȚII ARHITECTURALE UTILIZÂND SIMULATORUL

DLX. Probleme propuse spre rezolvare – conjectura lui Goldbach, perechi de numere prime impar consecutive, aplicații de sortare cu șiruri de numere.

b) UTILIZAREA SIMULATORULUI VLIW-DLX: ilustrarea principiilor fundamentale ale procesării VLIW (very long instruction word). Rolul software-lui în detecția și eliminarea hazardurilor RAW. Relația dintre instrucțiunea multiplă și instrucțiunile RISC primitive și independente, care vor fi alocate unităților de execuție în conformitate strictă cu poziția lor în instrucțiunea multiplă (număr/latențe). Optimizări software în procesoarele VLIW (loop unrolling, software pipelining). Avantaje / Dezavantaje față de procesoarele superscalare. Aplicabilitate – sisteme dedicate (procesoare de semnal, procesoare multimedia). Aplicație rezolvată – execuția pe un procesor VLIW a unui program care înmulțește două matrici pătratice.

L10. SATSIM: SIMULAREA ARHITECTURILOR SUPERSCALARE FOLOSIND ANIMAȚIE INTERACTIVĂ

(<http://webspaces.ulbsibiu.ro/adrian.florea/html/simulatoare/SATSimG.ZIP>). Înțelegerea conceptelor legate de procesarea *pipeline* a instrucțiunilor, execuția *out of order*, impactul negativ asupra performanței al predicției greșite a branch-urilor și al miss-urilor în cache. Evidențierea, prin simulare la nivel de execuție – pas cu pas, a aspectelor arhitecturale specifice procesoarelor RISC superscalare (stații de rezervare, unități funcționale de execuție, buffer de reordonare și buffer de redenumire).

L11. PSATSIM: INSTRUMENT SOFTWARE DE EVALUARE A PERFORMANȚEI DE PROCESARE ȘI A CONSUMULUI DE PUTERE ÎN MICROARHITECTURILE SUPERSCALARE.

a) **Modelarea consumului de putere** (CACTI, WATTCH). Parametrii tehnologici, clasificarea consumului de putere în funcție de tipul resursei arhitecturale (*structuri matriceale* – memoria cache, decodificatoare de adresă, setul de regiștrii generali, predictorul de salturi; *memorii adresabile după conținut* – asociative – stațiile de rezervare, logica de reactivare a instrucțiunilor situate în buffer-ul de reordonare în așteptarea efectuării fazei de commit; *circuite logice combinaționale* – implementarea unităților funcționale de execuție (ALU), logica de verificare a dependențelor de date, selecția instrucțiunilor spre stațiile de rezervare și unitățile funcționale). **Influența caracteristicilor arhitecturale specifice procesoarelor RISC superscalare asupra consumului de putere și respectiv asupra performanței.**

b) **Predictorul de salturi** – una din principalele cauze care determină consum ridicat de putere în arhitecturi. **Evidențierea efectului negativ al predicției greșite a instrucțiunilor de ramificație** (parametrizarea acurateții de predicție) **atât asupra vitezei de execuție a procesorului cât și asupra energiei consumate.**

L12. Seminar de aplicații. (II)

L13. Evaluare practică privind materia studiată la laborator.