# Proiect la Simularea si Optimizarea Arhitecturilor de Calcul

Fratila Andreea 244/1

Cuprins



## Tema proiect

Realizarea unui simulator pentru o arhitectura superscalara parametrizabila.Scopul principal este acela de a determina diferiti parametri de performanta, configuratii optime, pentru o arhitectura Harvard de memorie (cache-uri de instrucțiuni și date separate).

## A2

2.1 Determinaţi influenţa numărului maxim de instrucţiuni ce pot fi trimise simultan în execuţie, asupra ratei de procesare IR(IRmax).
2.2. La acest punct nu se va mai considera număr nelimitat de seturi de regiştrii generali. Se va determina numărul optim de seturi de regiştrii (2, 3, 4, ...IRmax) în variantele cu cache de date uniport (o singură instrucţiune cu referire la memorie se poate executa) sau biport (două instrucţiuni cu referire la memorie se pot executa: L+L sau L+S).
2.3. Pentru valoarea optimă determinată la punctul 2.2. a numărului de seturi de regiştrii, studiaţi comparativ performanţa (rata de procesare) pe două tipuri de cache de date (uniport sau biport).

# Detaliile arhitecturii hardware

Principalii parametrii ai arhitecturii sunt:

- **FR (rata de fetch)** - specifica numărul de instrucțiuni citite simultan din cache sau memorie într-un ciclu de tact; poate lua valori de 4, 8 sau 16 instructiuni.
- **IBS (instruction buffer size)** - dimensiunea buffer-ului de prefetch, măsurată în număr de instrucțiuni; plaja de valori: 4 (minim FR), 8, 16, 32; bufferul de prefetch este o coada ce lucrează după principiul FIFO (first in first out). Vor fi citite FR instrucțiuni simultan de la adresa specificata de PC (program counter) și depuse în partea superioară a bufferului. În același ciclu de execuție, instrucțiuni din partea inferioară sunt expediate spre unitățile de decodificare și execuție. O intrare în buffer va contine câmpurile:
  OPCODE - codul operatiei executata de instructiunea respectiva;
  PC_crt - adresa (Program Counter-ul) instrucțiunii curente;

DATE / INSTR - adresa din / la care se citesc / se scriu date din sau în memorie, în cazul instrucțiunilor cu referire la memorie, respectiv adresa instrucțiunii ținta în cazul instrucțiunilor de salt.

- **IRmax (issue rate maxim)** - numărul maxim de instrucțiuni, lansate în execuție simultan într-un ciclu de execuție, din buffer-ul de prefetch. Poate lua valorile: 2, 4, 8, 16 (maxim FR) instructiuni. Dacă rata de fetch este mai mica decât numărul maxim de instrucțiuni executate concurent într-un ciclu, atunci performanța este limitata de procesul de *fetch* instrucțiune. Considerăm executia instructiunilor "*in order*" - ordinea inițială a instrucțiunilor. O instrucțiune va fi executată abia după ce toate celelalte instrucțiuni de care ea depinde au fost executate.
- **Latenta** - numărul de cicli necesare execuției instrucțiunilor aritmetice, de salt și cele cu referire la memorie (în cazul în care accesele pentru obținerea datei sunt cu hit în cache). Inițial are valoarea 1.
- **Cache-ul de instrucțiuni (IC)** si **Cache-ul de date (DC)** - sunt cache-uri mapate direct, organizate în blocuri de capacități parametrizabile [4, 8, 16 (maxim IBS) locatii]. Încărcarea și evacuarea datelor în cache se face la nivel de bloc si nu la nivel de locatie.

V – bit de validare (0 – nu e valida data; 1 – valida;). Initial are valoare 0. Este necesar numai pentru programe auto modificabile la cookie-urile de instrucțiuni. La prima înscriere în cache este setat pe 1.

D – bit Dirty. Este necesar la scrierea în cache-ul de date (vezi pct.4).

SIZE_IC, SIZE_DC - dimensiunea cookie-urilor de instrucțiuni respectiv de date au plajă de valori de la 64 locatii (128, 256, ...) pâna la 8192 locatii.

TAG = data *div* SIZE_D(I)C

BLOC_OFFSET = [ data *mod* SIZE_D(I)C ] *div* BLOC_SIZE (FR)

BLOC_SIZE, FR - dimensiunea în locații a blocului din cache-ul de date respective instrucțiunii;

data – data emisa din program;

TAG - câmp de identificare al datei;

BLOC_OFFSET - offset-ul de bloc din cache;

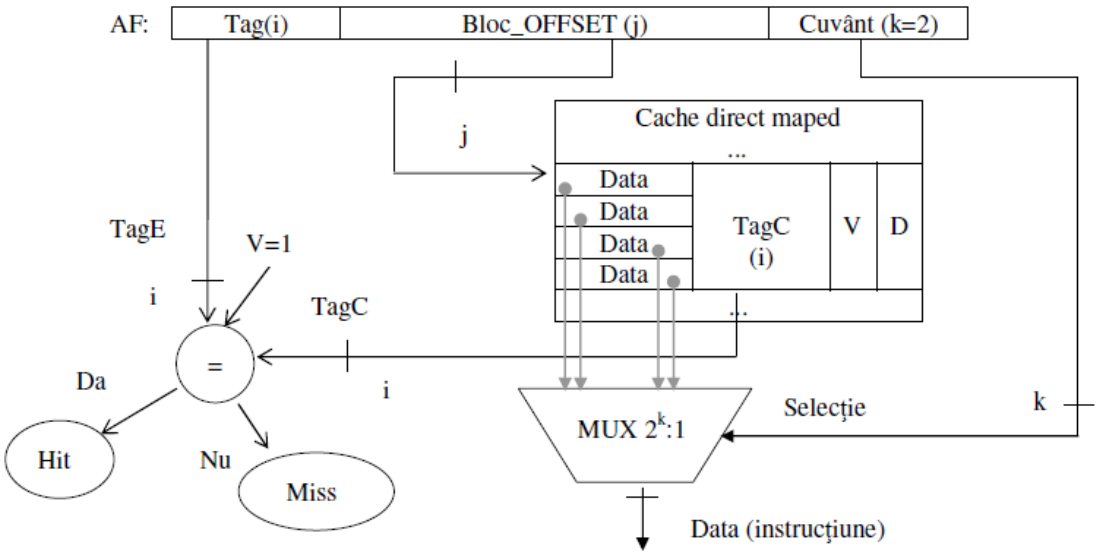



**Figura 1.** Structura cache-urilor de Instrucţiuni/Date

În cazul cache-urilor mapate direct, datele vor fi memorate în același loc de fiecare data când sunt accesate. Din acest motiv vom ști la fiecare acces ce data va fi evacuata din cache.

La o căutare în cache (IC sau DC) se ia tag-ul valorii căutate și se verifica daca ea exista la indexul sau la offsetul de bloc respectiv. În caz afirmativ spunem ca avem acces cu HIT, altfel MISS în cache și trebuie actualizat cache-ul.

- **Memoria principal**a **-** (care se accesează numai la *miss* în cache) va avea o latenta parametrizabila de N_PEN (10, 15, 20) tact-processor.

Este posibila execuția a doua instrucțiuni cu referire la memorie de genul: Load + Load sau Load + Store.

- Presupunem existența unui număr suficient de mare (maxim IRmax) de **seturi de registrii generali**: un set de regiștri generali este necesar pentru execuția unei instrucțiuni de tip aritmetico-logic sau cu referire la memorie.

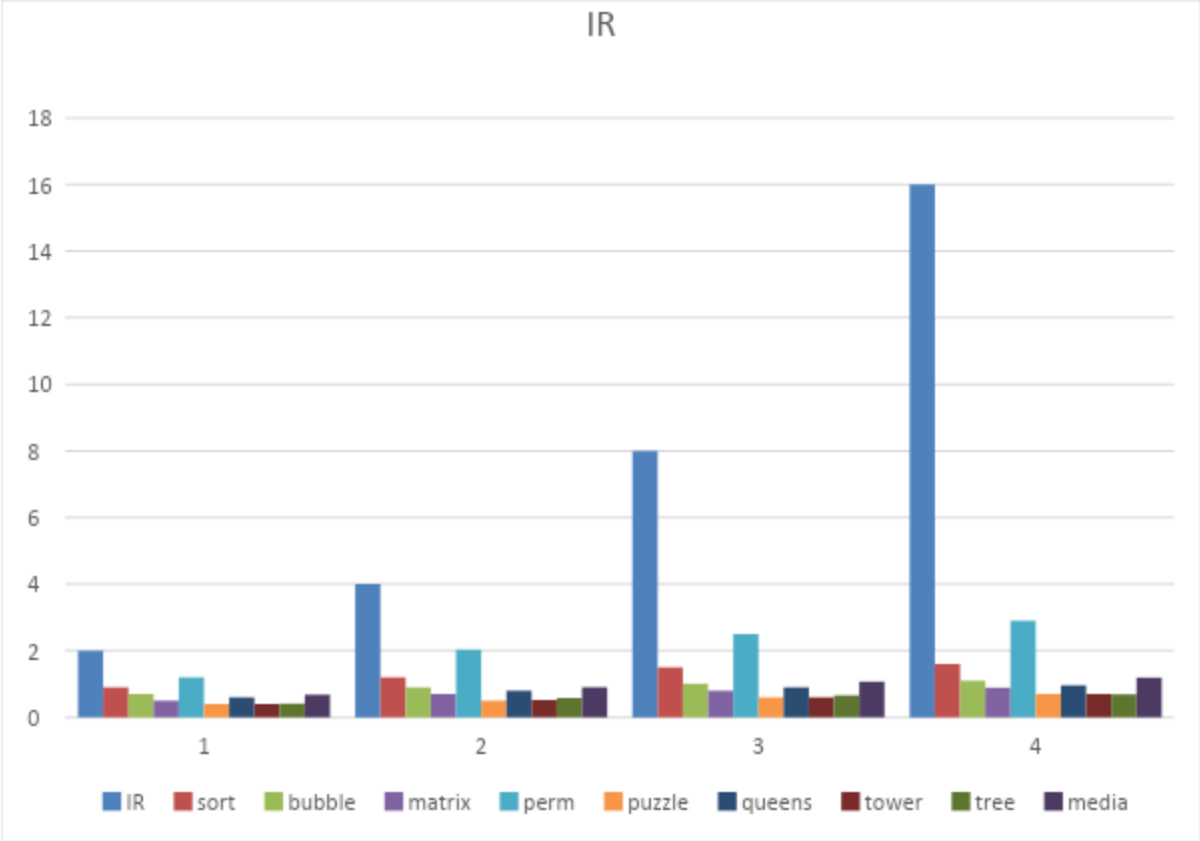
Programul va simula fișiere *trace* (*.trc), rulate pe arhitectura HSA (Hartfield Superscalar Architecture). Este vorba de 8 benchmark-uri Stanford, care cuprind probleme clasice de sortare, problema turnurilor din Hanoi, problema damelor, generare de permutări și inmultiri de matrici.

# Observații și concluzii

**2.1** Determinaţi influenţa numărului maxim de instrucţiuni ce pot fi trimise simultan în execuţie asupra ratei de procesare IR(IRmax).

Parametrii simulatorului:
Latenta pentru hit in cache=1
Latenta memorie= 10
IBS=32
IC=128
DC=256
IR=2 ,4,8,16
FR=4 8 16

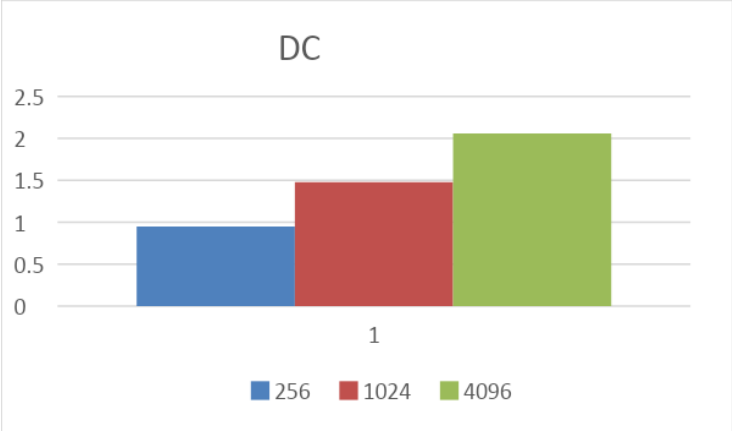
# În urma simulărilor pe cele 8 benchmark-uri am obtinut:

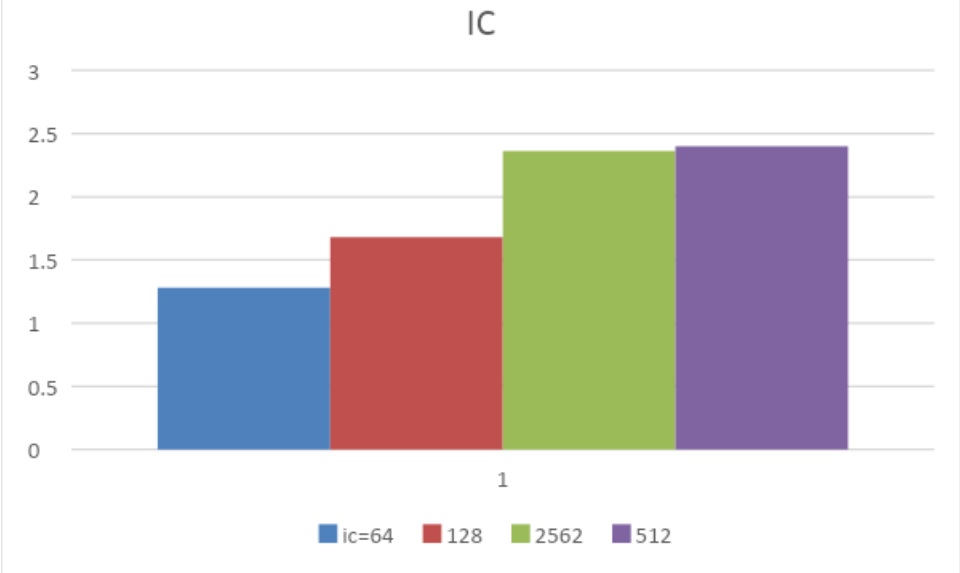



Se observa o creștere a ratei de la IR=2 la IR=4 destul de semnificativă in cazul track-urilor cu aproximativ 30%, iar in cazul “perm” cu circa 80% de la 1.2 la 2 . Daca avem IR=8 observam o crestere, dar nu la fel de semnificativă, cu maxim 20%, iar in unele cazuri și cu 5-10%. Atunci cand IR=16 o sa avem o crestere foarte scăzută cu 2-3% mai puțin in cazul track-ului “perm”. Deci creșterea IR max=16 nu ne oferă un Issue rate optimal pt ca desi o sa putem executa 16 instrucțiuni pe ciclu de execuție, rata de issue nu este scalabilă decât într-un procent destul de scăzut datorită dependențelor dintre

instructiuni, branch-uri  gresit predictionate , miss-urilor de pe cache de instrucțiuni/date.

Figurile  următoare ilustrează  evoluția ratei de issue in funcție de  cache-ul de date/instrucțiuni :



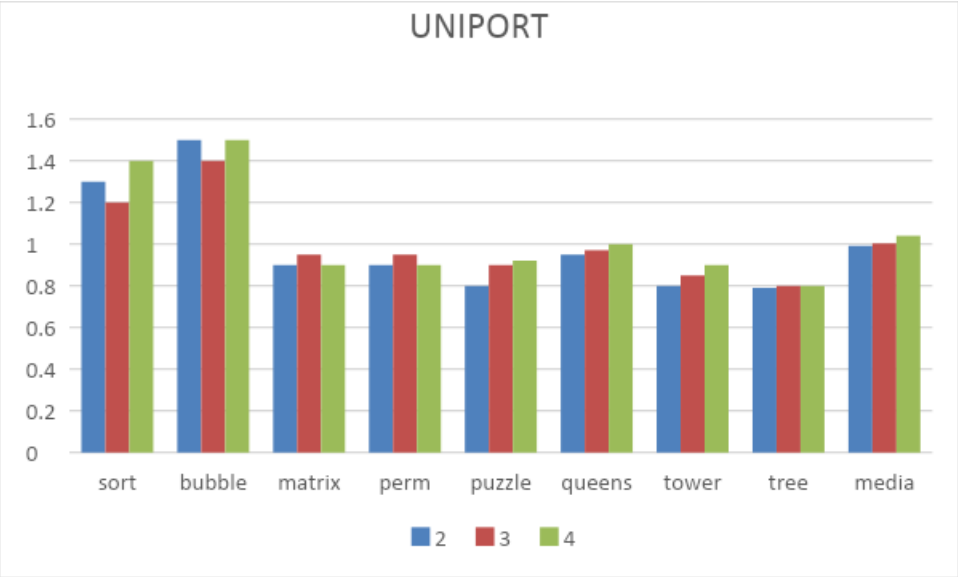
Concluzia este una interesanta, întrucât că rezultate
, I-Cache optim este de 256 locatii iar D-Cache optim de 4k locații. Creșterea lor nu ne oferă o îmbunătățire de performanta!

2.2

La acest punct nu se va mai considera număr nelimitat de seturi de regiştrii generali. Se va determina numărul optim de seturi de regiştrii (2, 3, 4, ...IRmax) în variantele cu cache de date uniport (o singură instrucţiune cu referire la memorie se poate executa) sau biport (două instrucţiuni cu referire la memorie se pot executa: L+L sau L+S).

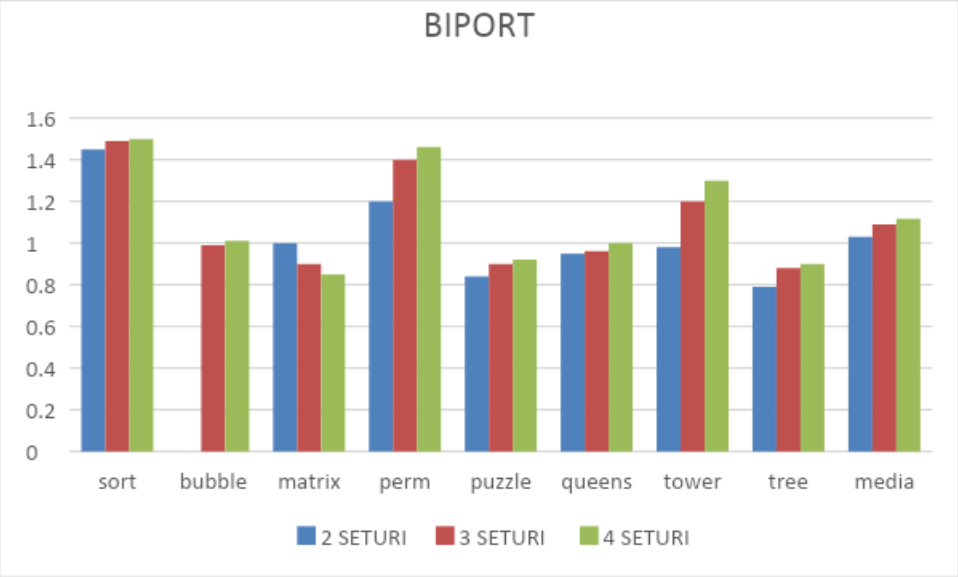
**Parametrii :** IBS=16 FR=8, IR=4

CACHE UNIPORT DE DATE :




Cresterea relativa de performanta între doua seturi fizice si 4 seturi fizice este de doar 5.1%.

**CACHE BIPORT PE DATE :**



În figura , se prezinta influenta numarului de
seturi de regiștri fizici asupra performanțelor, considerând un cache biport pe date.
Pentru 2, 3 și 4 seturi s-au obținut rate de procesare de 1.03, 1.09 si 1.11 instr./tact, adica cresteri relative de 5.8% respectiv 7.8%, relativ mici.
Asadar, mărirea setului de regiștri nu conduce la o crestere spectaculoasă  a performanței, aspect datorat probabil și gradului limitat de paralelism  (executie In Order pe programe neoptimizate), determinat de către dependențele RAW între instrucțiunile din buffer.

2.3. Pentru valoarea optimă determinată la punctul 2.2. a numărului de seturi de regiştrii, studiaţi comparativ performanţa (rata de procesare) pe două tipuri de cache de date (uniport sau biport).

DECI in urma rezultatelor obtinute  medii armonice ale ratelor de procesare de 1.01 pt cache de date uniport, respectiv  1.11 instr./tact pt cele biport, adică o creștere relativă de 10% în favoarea celor biport.

## Bibliografie

Computer architecture – HENNESSY SI  PATTERSON

Suportul de curs de la materia “Simularea și optimizarea arhitecturilor de calcul”, Prof. Lucian Vintan, Facultatea de Inginerie Hermann Oberth, Sibiu.

Suportul de laborator la materia “Simularea și Optimizarea Arhitecturilor de Calcul”, Prof. Adrian Florea, Facultatea de Inginerie “Hermann Oberth”, Sibiu.