Proiect la Simularea si Optimizarea Arhitecturilor de Calcul

Bumbar Constantin-Marius:

Indrumator: Conf.univ.dr.ing. Adrian Florea

Cuprins

Tema proiect……………………………………………………………………..2

Detaliile arhitecturii hardware…………………………………………..3

Necesar resurse sistem……………………………………………………..6

Implementare…………………………………………………………………...7

Observatii si concluzii………………………………………………………..10

Bibliografie……………………………………………………………………….16

**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 instructiuni si 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 numarul de instructiuni 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, masurata în numar de instructiuni; plaja de valori: 4 (minim FR), 8, 16, 32; buffer-ul de prefetch este o coada ce lucreaza dupa principiul FIFO (first in first out). Vor fi citite FR instructiuni simultan de la adresa specificata de PC (program counter) si depuse în partea superioara a buffer-ului. În acelasi ciclu de executie, instructiuni din partea inferioara sunt expediate spre unitatile de decodificare si executie. O intrare în buffer va contine câmpurile:

OPCODE - codul operatiei executata de instructiunea respectiva;

PC\_crt - adresa (Program Counter-ul) instructiunii curente;

DATE / INSTR - adresa din / la care se citesc / se scriu date din sau în memorie, în cazul instuctiunilor cu referire la memorie, respectiv adresa instructiunii tinta în cazul instructiunilor de salt.

* **IRmax (issue rate maxim)** - numarul maxim de instructiuni, lansate în executie simultan într-un ciclu de executie, din buffer-ul de prefetch. Poate lua valorile: 2, 4, 8, 16 (maxim FR) instructiuni. Daca rata de fetch este mai mica decât numarul maxim de instructiuni executate concurent într-un ciclu, atunci performanta este limitata de procesul de *fetch* instructiune. Consideram executia instructiunilor “*in order*” - ordinea initiala a instructiunilor. O instructiune va fi executata abia dupa ce toate celelalte instructiuni de care ea depinde au fost executate.
* **Latenta** - numarul de cicli necesari executiei instructiunilor aritmetice, de salt si cele cu referire la memorie (în cazul în care accesele pentru obtinerea datei sunt cu hit în cache). Initial are valoarea 1.
* **Cache-ul de instruc**t**iuni (IC)** si **Cache-ul de date (DC)** - sunt cache-uri mapate direct, organizate în blocuri de capacitati parametrizabile [4, 8, 16 (maxim IBS) locatii]. Încarcarea si 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 valoarea 0. Este necesar numai pentru programe automodificabile la cache-urile de instructiuni. 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 cache-urilor de instructiuni respectiv de date au plaja de valori de la 64 locatii (128, 256, ...) pâna la 8192 locatii.

TAG = data *div* SIZE\_D(I)C

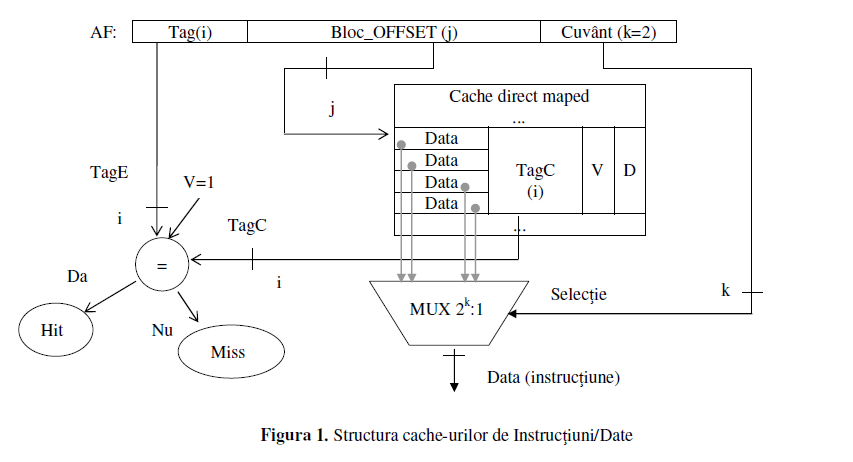
BLOC\_OFFSET = [ data *mod* SIZE\_D(I)C ] *div* BLOC\_SIZE (FR)

BLOC\_SIZE, FR - dimensiunea în locatii a blocului din cache-ul de date respective instructiunii;

data – data emisa din program;

TAG - câmp de identificare al datei;

BLOC\_OFFSET - offset-ul de bloc din cache;



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

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

• **Memoria principal**a **-** (care se acceseaza numai la *miss* în cache) va avea o latenta parametrizabila de N\_PEN (10, 15, 20) tact-procesor.

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

• Presupunem existenta unui numar suficient de mare (maxim IRmax) de **seturi de registrii generali**: un set de registrii generali este necesar pentru executia unei instructiuni de tip aritmetico-logic sau cu referire la memorie.

Programul va simula fisiere *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 permutari si înmultiri de matrici.

**Necesar resurse sistem**

*Componente Hardware:*

* 1.6 GHz or faster processor.
* 1 GB of RAM (1.5 GB if running on a virtual machine)
* 10 GB (NTFS) of available hard disk space.
* 5400 RPM hard drive.
* DirectX 9-capable video card running at 1024 x 768 or higher display resolution.

### Supported Architectures

* 32-bit (x86)
* 64-bit (x64)

*Componente Software:*

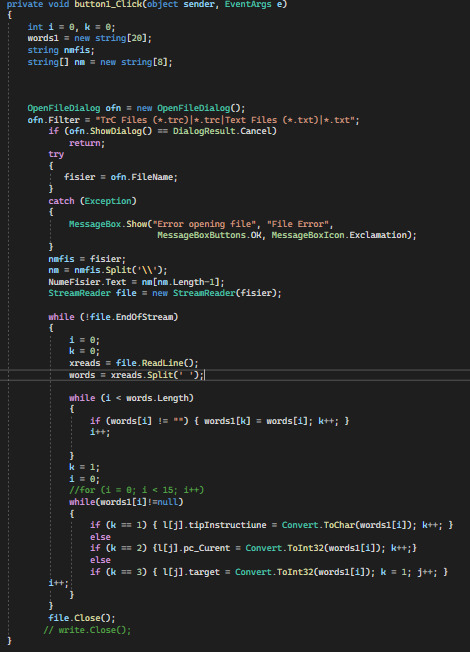
**Supported Operating Systems**

* Windows 8.1 (x86 and x64)
* Windows 8 (x86 and x64)
* Windows 7 (x86 and x64)
* Windows Server 2008 R2 SP1 (x64
* Windows Server 2012 (x64)

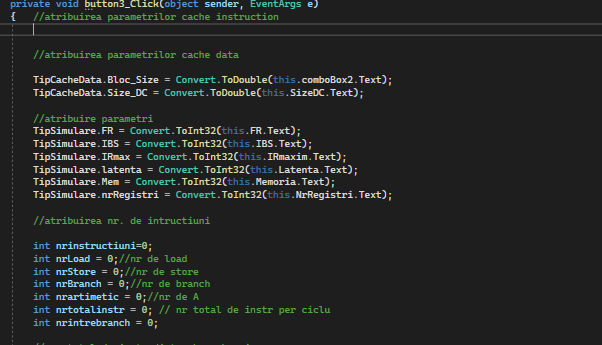
Aplicatie: **Visual studio 2015- C#**

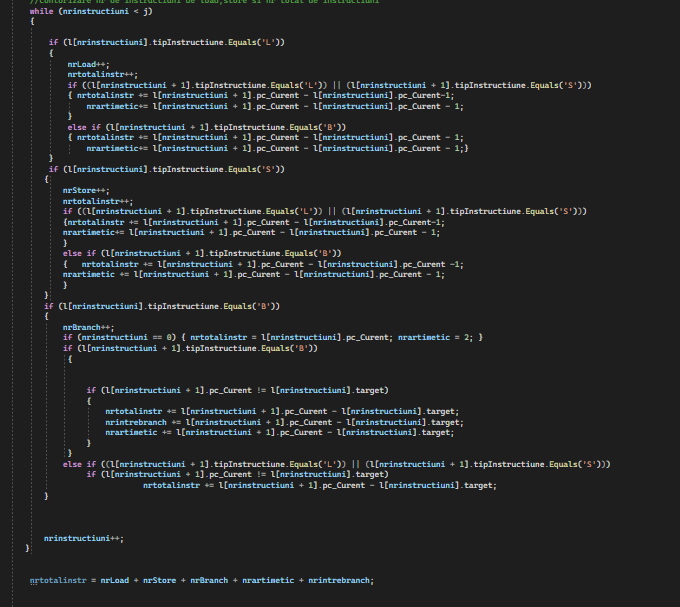
**Implementare**

-implementarea de open file dialog - deschiderea fisierelor .trc

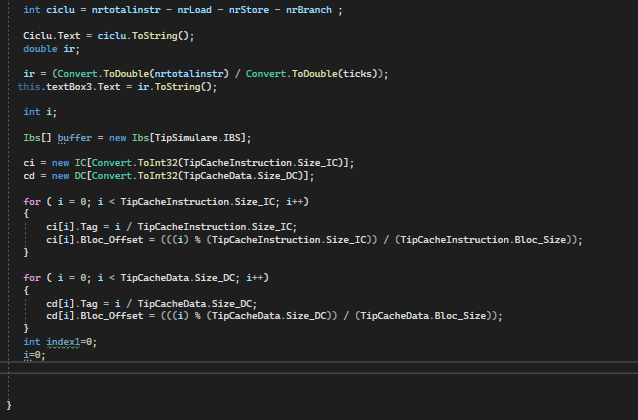


-implementarea butonului “Start simulare”



-calculul numarului total de instructiuni

-numarul de instructiuni pe un singlu ciclu de tact



**Observatii si 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 memoriei= 10

IBS=32

IC=128

DC=256

IR=2 ,4,8,16

FR=4 8 16

**In urma simularilor pe cele 8 benchmarkuri am obtinut:**

Se observa o crestere a ratei de la IR=2 la IR=4 destul de semnificativa in cazul trace-urilor cu aproximativ 30% iar in cazul “perm” cu circa 80% de la 1.2 la 2 . Daca IR=8 observam o crestere dar nu la fel de semnificativa cu maxim 20% iar in unele cazuri si cu 5-10% .Atunci cand IR=16 o sa avem o crestere foarte scazuta cu 2-3% mai putin in cazul trace-ului “perm” . Deci cresterea IR max=16 nu ne ofera un Issue rate optimal pt ca desi o sa putem executa 16 instructiuni pe ciclu de executie rata de issue nu este scalabila decat intr-un procent destul de scazut datorita dependentelor intre instructiuni, branch-uri gresit predictionate , miss-urilor de pe cache de instructiuni/date.

Figurile urmatoare ilustreaza evolutia ratei de issue in functie de cache-ul de date/instructiuni :

Concluzia este una interesanta, întrucât ca rezultate

, I-Cache optim este de 256 locatii iar D-Cache optim de 4k locatii. Cresterea lor nu ne ofera o imbunatatire 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 registri fizici asupra performantelor, considerând un cache biport pe date.

Pentru 2, 3 si 4 seturi s-au obtinut 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, marirea setului de registri nu conduce la o crestere spectaculoasa a performantei, aspect datorat probabil si gradului limitat de paralelism (executie In Order pe programe neoptimizate), determinat de catre dependentele RAW între instructiunile 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, adica o crestere relativa de 10% în favoarea celor biport.

**Bibliografie**

Computer architecture – HENNESSY SI PATTERSON

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

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