UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU

FACULTATEA DE INGINERIE

Proiect la

Simularea și Optimizarea Arhitecturilor de Calcul

**Autori:**

Răzvan-Andrei Gâță

Alexandra Herman

**Semigrupa:**

241/3

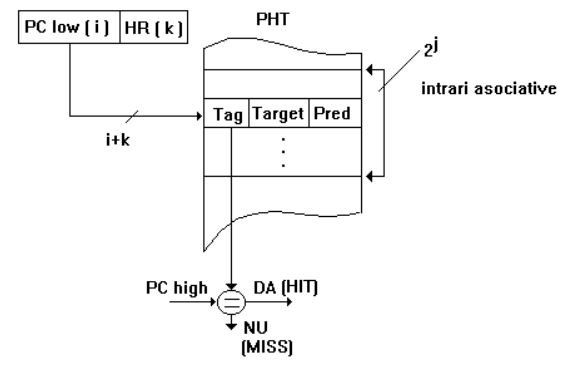
**Coordonator științific laborator:**

Andrei Patrausanu

2023 - 2024

# Tema Proiectului – Introducere Teoretică

Tema proiectului constă în implementarea unei scheme de predicție corelată pe două niveluri de tip **Gag (Global History Register, Global Prediction History Table)** – pentru referință a se analizeze figura 1.1.



**Figura 1.1**

Schema de predicție implementată va fi folosită pentru a analiza performanța diferitelor configurații (tip de arhitectura, numărul biților de predicție, etc.) și pentru a determina configurația optimă.

## Fișierele Trace

Simularea este efectuată pe benchmark-urile Stanford, 8 fişiere trace cu extensia (\*.tra). Aceste fişiere sunt o prelucrare a programelor scrise în mnemonică de asamblare (\*.ins) şi a trace-urilor originale (\*.trc), cu scopul de a evidenţia toate salturile (inclusiv cele care nu se fac). Întregul fişier trace (\*.trc) este o înlănţuire de triplete , unde TipInstr poate lua una din valorile ‘B’ – branch, ‘L’ – load, ‘S’ – store; AdrCrt reprezintă valoarea registrului PC – adresa instrucţiunii curente, iar AdrDest reprezintă adresa de memorie a datei accesate – în cazul instrucţiunilor cu referire la memorie (‘L’ / ‘S’) sau adresa destinaţie a saltului – în cazul instrucţiunilor de salt şi ramificaţie (‘B’). Există totuşi o deficienţă a trace-urilor cu extensia \*.trc: nu evidenţiază salturile care nu se fac. Din acest motiv s-au generat noile trace-uri (fişierele \*.tra).

Acestea conţin doar branch-urile (atât cele care se fac cât şi cele care nu se fac) şi exclude instrucţiunile Load / Store. Forma acestor fişiere este următoarea:

BT 12 30

BS 32 98

BM 100 33

NT 36 37

BRA 2 100

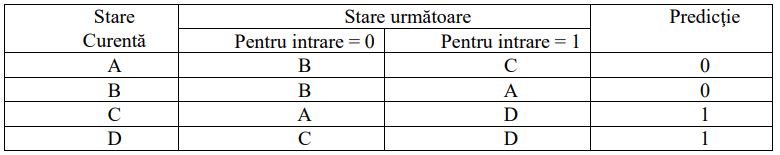
MOV PC, RA (RETURN)

Reprezentarea salturilor se face tot sub forma unor triplete , unde TipBr se prezintă sub forma unei codificări pe două caractere, primul dintre ele indică dacă saltul se face (‘B’ – saltul se face, ‘N’ – saltul nu se face), iar al doilea caracter indică tipul saltului: ‘T’ sau ‘F’ – salturi condiţionate, ‘S’ – apeluri de tip Call, ‘M’ – apeluri de tip Return, ‘R’ – salturi necondiţionate. Alegerea acestei codificări a fost inspirată de mnemonicile întâlnite în sursa în limbaj de asamblare (BT, BF – salt condiţionat, BSR – instrucţiune de tip Call, BRA – salt necondiţionat şi MOV PC, RA – instrucţiune de tip Return).

## Automatul de predicție

Este descris printr-un şir de caractere cu un format mai special, ce prezintă atât numărul de stări, tranziţiile între stări cât şi predicţia aferentă fiecărei stări. Pentru o mai bună înţelegere a funcţionării automatului exemplificăm:

Automatul: BCBAADCD:12 – pe 2 biți și tabelul din figura 1.2 care descrie funcționarea automatului.



**Figura 1.2**

Prima parte a şirului până la caracterul ‘:’ reprezintă tranziţiile pentru starea ‘A’ cu intrare 0, apoi cu intrare 1, apoi tranziţiile din ‘B’ pentru aceleaşi intrări, etc. Numărul din a doua parte este “văzut” în binar sub forma “0010” şi reprezintă ieşirile asociate fiecărei stări în parte (stării ‘A’ îi este asociat cel mai puţin semnificativ bit, în acest caz bitul ‘0’, următorul bit lui ‘B’, bitul ‘0’, următorul bit lui ‘C’, bitul ‘1’, etc).

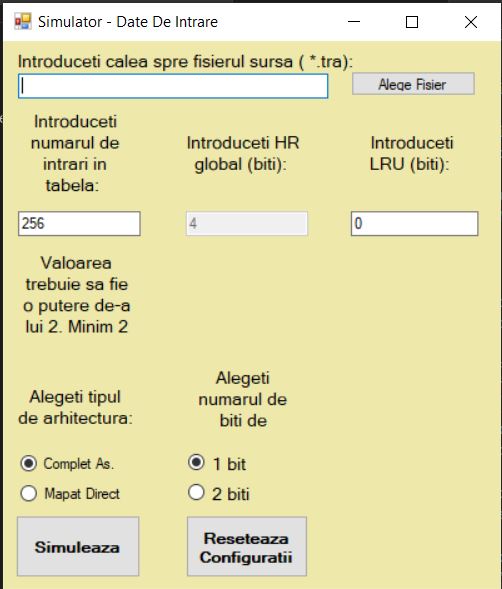
## Tabela PHT

Tabela de predicţii PHT (Prediction History Table) este adresată cu un index rezultat din concatenarea a două informaţii ortogonale: PClow (i biţi), semnificând gradul de localizare al saltului, respectiv registrul de predicţie (HR- History Register pe k biţi), semnificând "contextul" în care se situează saltul în program. Ambele contribuţii s-au făcut cu scopul eliminării interferenţelor branch-urilor în tabela de predicţie. Adresarea PHT exclusiv cu HR duce la serioase interferenţe (mai multe salturi pot accesa aceelaşi automat de predicţie din PHT), cu influenţe evident defavorabile asupra performanţelor. PHT poate avea diferite grade de asociativitate (iniţial considerăm tabela PHT complet asociativă). Un cuvânt din această tabelă are un format similar cu cel al cuvântului dintr-un BTB.

În cazul proiectării tabelei PHT complet asociativă evacuarea se face prin metoda LRU (cel mai puţin recent folosit).

# Ghid de Utilizare a Simulatorului

La deschiderea aplicației, se poate observa fereastra input (figura 2.1), unde utilizatorul va trebui să completeze cu detaliile necesare simularii precum: alegerea fișierului trace, numărul de intrări în tabela PHT, numărul de biți de predicție, etc.

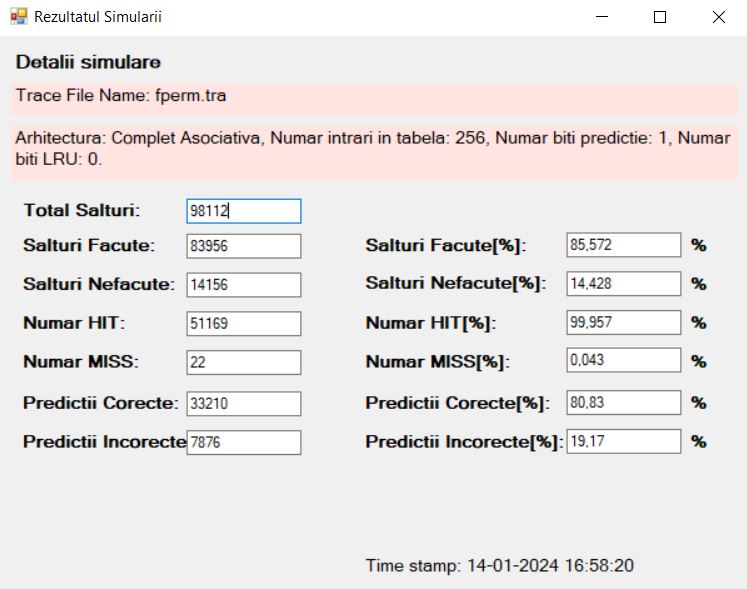


**Figura 2.1**

Se pot observa următoarele elemente:

* Butonul “Alege Fisier”, prin intermediul căruia utilizatorul va putea alege fișierul trace (.tra);
* Butoanele radio “Complet As.” “Mapat Direct”, de unde utilizatorul poate selecta arhitectura dorita;
* Butonul “Simuleaza”, pornește simularea;
* Butonul “Reseteaza Configuratii", resetează toate câmpurile la valoarea lor inițială;
* Trei casete de text unde utilizatorul poate alege numărul de intrări în tabela PHT, numărul de biți LRU și numărul de biți pentru HR global;

După ce butonul de simulare este apăsat se va deschide fereastra cu rezultatul simulării (figura 2.2), unde utilizatorul va putea vedea diferite informații despre simulare.



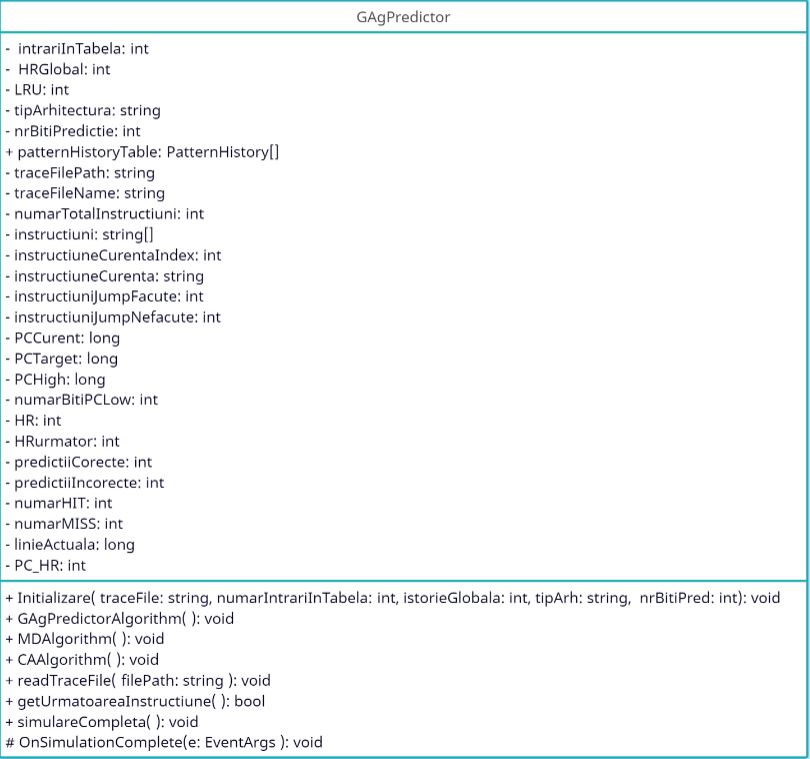
**Figura 2.2**

Prin intermediul ferestrei de rezultat se pot obține detalii precum numărul de salturi făcute, numărul de salturi nefăcute, numărul de HIT-uri, etc., acestea fiind disponibile sub formă de valoare numerică sau procent.

# Structura Software

Aplicația a fost dezvoltată în Visual Studio 2022, folosind ca limbaj de programare C#. Proiectul este alcătuit din 2 obiecte de tip Windows Form și încă 4 clase auxiliare. Obiectele de tip Windows Form au fost folosite pentru a crea o interfață între utilizator și aplicația implementată, astfel acestea reprezintă fereastra de input (figura 2.1) și fereastra de output (figura 2.2).

Clasa principală o reprezintă clasa GagPredictor.cs:



Dintre metodele acestei clase, cele mai importante sunt getUrmatoareaInstructiune() si simulareCompleta(). Aceste două metode împreună cu metodele apelate de către acestea, practic, reprezintă tot algoritmul implementat.

Alte clase care joacă un rol important, dar nu sunt atât de complexe încât să merite prezentarea unei diagrame de clasă sunt:

* Converter.cs folosită pentru metodele de conversie între decimal și binar sau binar și decimal.
* Form1.cs folosită pentru a crea interfața de input;
* outForm.cs folosită pentru a crea interfața de output;

# Resurse Hardware și Software Necesare

Programul a fost realizat în limbajul de programare C#, folosind Visual Studio 2022.

Pentru ca fișierele trace să fie potrivite pentru aplicația pe care am implementat-o, acestea trebuie să fie salvate cu extensia .tra. Conținutul fișierelor trebuie să fie formatat după urmatoarea regulă, pe fiecare rând va trebuie să se regăsească:

**[Tip\_Salt Adresa\_Curenta Adresa\_Tinta]**

Aceste trei informații de pe fiecare rând vor fi separate printr-un singur caracter ‘space’.

# Concluzii

În urma realizării acestui proiect am aprofundat și am înțeles principiul de funcționare al predictorului de tip GAg și a automatului de predicție pe 1 sau 2 biți.

Una dintre posibilele îmbunătățiri pe care le putem face în viitor este să numărăm și numărul de predicţii incorecte datorate exclusiv adresei de salt incorecte din tabelă, acest aspect nu l-am luat în considerare în implementarea noastră.

CUPRINS

[1. Tema Proiectului – Introducere Teoretică 2](#_Toc156146386)

[1.1. Fișierele Trace 2](#_Toc156146387)

[1.2. Automatul de predicție 3](#_Toc156146388)

[1.3. Tabela PHT 4](#_Toc156146389)

[2. Ghid de Utilizare a Simulatorului 5](#_Toc156146390)

[3. Structura Software 7](#_Toc156146391)

[4. Resurse Hardware și Software Necesare 9](#_Toc156146392)

[5. Concluzii 10](#_Toc156146393)