**Program de testare a parametrilor de performanta ai unui PC**

Damian Tudor Constantin

Universitate Tehnica din Cluj-Napoca

15.10.2023

Cuprins

[1. Introducere 3](#_Toc150755687)

[1.1 Context 3](#_Toc150755688)

[1.2 Specificatii 3](#_Toc150755689)

[1.3 Obiective 3](#_Toc150755690)

[2. Studiu bibliografic 4](#_Toc150755691)

[2.1 Procesorul 4](#_Toc150755692)

[2.2 Frecventa 4](#_Toc150755693)

[2.3 Dimensiunea memoriei 5](#_Toc150755694)

[2.4 Viteza de transfer a datelor 5](#_Toc150755695)

[2.5 Viteza de executie a operatiilor aritmetice 5](#_Toc150755696)

[3. Analiza 5](#_Toc150755697)

[3.1. Datele despre procesor 5](#_Toc150755698)

[3.2. Datele despre memoria cache și memoria RAM 6](#_Toc150755699)

[3.3. Viteza de execuție a operațiilor aritmetice 6](#_Toc150755700)

[3.4. Viteza de transfer a fișierelor 6](#_Toc150755701)

[3.5 Cazuri de utilizare 7](#_Toc150755702)

[4. Design 8](#_Toc150755703)

[5. Implementare 9](#_Toc150755704)

[5.1. Specificațiile procesorului 9](#_Toc150755705)

[5.2. Specificațiile memoriei cache 9](#_Toc150755706)

[5.3. Specificațiile memoriei RAM 10](#_Toc150755707)

[6. Testare si validare 10](#_Toc150755708)

[7. Concluzii 14](#_Toc150755709)

[8. Bibliografie 14](#_Toc150755710)

# 1. Introducere

## 1.1 Context

Scopul proiectului este de a crea un program care să măsoare performanța unui computer. Performanța unui computer poate fi folosită pentru a compara computere între ele și pentru a determina cât de puternic este un calculator. Performanța poate fi influențată de factori precum tipul de procesor, frecvență de lucru, cantitatea de memorie și viteza de transfer a datelor.

Acest program poate fi folosit de către oricine dorește să afle mai multe despre performanța calculatorului lor. Programul va oferi informații despre proprietățile calculatorului și despre modul în care acesta se descurcă în îndeplinirea anumitor sarcini.

## 1.2 Specificatii

Programul va fi creat folosind Visual Studio și limbajul C. Rezultatele vor fi afișate într-o aplicație de tip Java Swing. Utilizatorul va putea vedea informații despre procesor, memorie cache și memorie RAM. De asemenea, utilizatorul va putea vedea timpul de execuție al unor operații aritmetice.

## 1.3 Obiective

Proiectarea unui program care sa poată măsura parametrii de performanță ai unui PC, precum tipul procesorului, frecvența, dimensiunea memoriei, viteza de execuție a unor operații aritmetice.

**Planificare:**

Pentru a duce proiectul la bun sfârșit, voi respecta următoarele termene limită:

* Laborator 1: alegerea temei proiectului
* Laborator 2: scrierea primelor 2 capitole din documentație: Introducere și Studiu bibliografic
* Laborator 3: scrierea următoarelor 2 capitole din documentație: Analiză si Design

- Laborator 4: crearea interfeței grafice a aplicației, afișarea datelor despre procesor și scrierea următoarelor capitole din documentație

- Laborator 5: afișarea datelor despre memoria cache și memoria RAM

- Laborator 6: afișarea datelor legate de viteza de execuție a operațiilor aritmetice și viteza de alocare, realocare și dezalocare a memorii RAM.

- Laborator 7: predarea proiectului în forma finală

# 2. Studiu bibliografic

## 2.1 Procesorul

Procesorul este elementul principal de calcul și incorporează funcțiile unității centrale de prelucrare a informației a unui calculator sau a unui sistem electronic structurat funcțional.

Procesoarele sunt identificate prin doi parametri principali: viteză și lățime. Viteza este măsurată în megahertz (MHz) și reprezintă numărul de cicluri de ceas pe secundă care pot fi executate de procesor. Lățimea este măsurată în biți și reprezintă cantitatea de date pe care procesorul o poate procesa simultan.

**Memoria cache a procesorului**

Memoria cache a procesorului este o memorie de mare viteză care este utilizată pentru a stoca date și instrucțiuni care sunt utilizate frecvent. Acest lucru ajută la accelerarea procesorului, deoarece nu trebuie să acceseze memoria principală, care este mai lentă.

Memoria cache este organizată în niveluri, cu L1 Cache fiind cel mai rapid și cel mai mic și L3 Cache fiind cel mai lent și cel mai mare.

Memoria cache poate fi de două tipuri: Unified Cache sau Split Cache. Unified Cache stochează atât datele, cât și instrucțiunile, în timp ce Split Cache are două unități separate: una pentru date și una pentru instrucțiuni.

Din punct de vedere al mapării datelor, memoria cache poate folosi trei metode: mapare directă, complet-asociativă sau set-asociativă. Mapare directă înseamnă că fiecare locație din memoria principală este mapată la o singură locație din memoria cache. Complet-asociativă înseamnă că fiecare locație din memoria principală poate fi mapată la orice locație din memoria cache. Set-asociativă înseamnă că fiecare locație din memoria principală poate fi mapată la oricare dintre o serie de locații din memoria cache.

În ceea ce privește politica de scriere în memoria cache, memoria cache poate folosi două metode: write-back sau write-through. Write-back înseamnă că datele sunt scrise în memoria principală numai atunci când nu mai sunt necesare în memoria cache. Write-through înseamnă că datele sunt scrise în memoria principală imediat după ce sunt scrise în memoria cache.

## 2.2 Frecventa

Viteza ciclului de ceas a unui calculator este măsurată în cicluri pe secundă. Un oscilator de cristal controlează viteza clock-ului folosind un cristal de argint. Când este aplicată tensiune pe cristal, acesta începe să oscileze periodic, iar perioada de oscilație este determinată de forma și dimensiunea cristalului. În urma aplicării tensiunii, cristalul emană un semnal electric care reprezintă semnalul de ceas al calculatorului. Un PC obișnuit rulează milioane de astfel de cicluri pe secundă, de aceea viteza se măsoară în megahertz

## 2.3 Dimensiunea memoriei

Capacitatea memoriei unui dispozitiv este cantitatea de date pe care o poate stoca. Această capacitate este determinată de procesor și de sistemul de operare.

Procesoarele cu o arhitectură pe 32 de biți pot adresa memorie cu o capacitate maximă de 4 GB. Cele cu arhitectură pe 64 de biți pot adresa memorie cu o capacitate nelimitată. Sistemele de operare pot limita, de asemenea, capacitatea memoriei adresabile.

Memoria unui dispozitiv este împărțită în două tipuri: memoria RAM și memoria de stocare. Memoria RAM este memoria de lucru a dispozitivului. Este volatilă, ceea ce înseamnă că datele sunt șterse atunci când dispozitivul este oprit. Memoria de stocare este memoria pe termen lung a dispozitivului. Este nevolatilă, ceea ce înseamnă că datele sunt păstrate chiar și atunci când dispozitivul este oprit.

## 2.4 Viteza de transfer a datelor

Viteza de transfer a datelor este cantitatea de date care pot fi transferate între două dispozitive într-o anumită perioadă de timp. Este măsurată în unități de bit sau byte pe secundă.

Pentru a măsura viteza de transfer a datelor, aplicația va genera fișiere de diferite dimensiuni și le va transfera între două directoare diferite, de un anumit număr de ori. Apoi, aplicația va calcula media timpilor de transfer pentru a obține o estimare a vitezei de transfer a datelor.

## 2.5 Viteza de executie a operatiilor aritmetice

Viteza de execuție este viteza cu care un dispozitiv computational poate executa operații aritmetico-logice. Se poate măsura cu ajutorul funcției clock(), din biblioteca <time.h>. Această funcție returnează numărul de impulsuri de ceas de la începerea programului. Prin împărțirea acestui număr la constanta CLOCKS\_PER\_SEC vom obține timpul de rulare al programului, în secunde.

# 3. Analiza

## 3.1. Datele despre procesor

Pentru a afișa date despre procesor, aplicația va folosi clasa ManagementObjectSearcher din Java Swing. Pentru a extrage datele despre procesor, aplicația va crea un obiect ObjectQuery cu interogare "select \* from Win32\_Processor". Apoi, aplicația va parcurge rezultatele interogării și va selecta doar cele mai relevante pentru utilizator, spre a fi afișate.

## 3.2. Datele despre memoria cache și memoria RAM

Procedura de extragere a datelor pentru memoria cache și memoria RAM este similară cu cea pentru procesor. Singura diferență este că interogarea ObjectQuery va fi diferită pentru fiecare componentă.

Pentru memoria cache, aplicația va folosi interogarea "select \* from Win32\_CacheMemory". Datele extrase pentru memoria cache, legate de tipul de mapare, locație, nivel și politică de scriere sunt primite ca și coduri. Pentru ca utilizatorul să înțeleagă despre ce e vorba, aplicația va afișa direct ceea ce reprezintă fiecare cod în dreptul câmpurilor memoriei cache. Însemnătatea fiecăruia va fi luată folosind documentația oficială de la Microsoft a clasei Win32\_CacheMemory, din sursa. Aplicația va folosi dicționare pentru a mapa însemnătatea codurilor.

Pentru memoria RAM, aplicația va folosi interogarea "select \* from Win32\_PhysicalMemory". Procedura de afisare a datelor este similară cu cea pentru memoria cache.

## 3.3. Viteza de execuție a operațiilor aritmetice

Pentru a măsura viteza de execuție a operațiilor aritmetice, aplicația va folosi clasa StopWatch din Java Swing. Aplicația va efectua mai multe măsurări, va face media lor și o va afișa utilizatorului în interfața aplicației.

## 3.4. Viteza de transfer a fișierelor

Pentru a testa viteza de transfer a datelor, aplicația va folosi un algoritm care generează un fișier aleatoriu, de o dimensiune dată. Dimensiunea va fi oferită aplicației de către utilizator, acesta având opțiunea să aleagă fișiere cu dimensiune multiplă a lui 2 MB. Apoi, aplicația va folosi clasa StopWatch pentru a măsura timpul în care se va copia fișierul într-un alt director. După ce fișierul este copiat, aplicația va salva timpul rezultat și îl va afișa utilizatorului. Aplicația va șterge apoi fișierele create și va repeta aceste operații de generare de fișiere, copiere și măsurare a timpului de un anumit număr de ori, pentru a face o medie și a o oferi utilizatorului.

Pentru a calcula viteza de transfer în MBps, aplicația va folosi formula:

**VitezaDeTransfer [MBps] = (dimensiunea fișɪeruluɪ [MB] / (timpul de transfer mă𝑠𝑢rat [s])) \* 1000**

## 3.5 Cazuri de utilizare

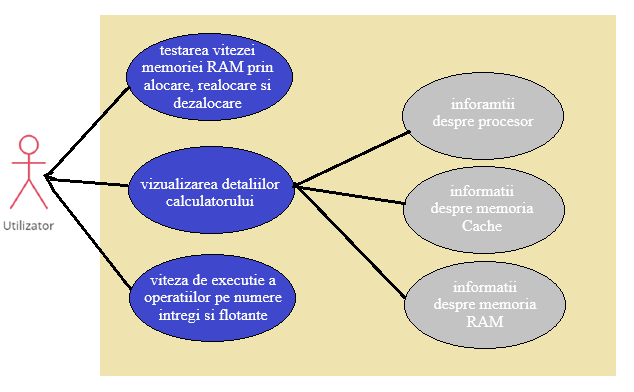


Figure : Diagrama cazurilor de utilizare

Cazul de utilizare 1: Vizualizarea detaliilor calculatorului

Actor: utilizatorul

Scenariu principal:

- utilizatorul deschide aplicația

- se încarcă fereastra principală

- apar date legate de procesorul calculatorului

- apar date despre memoria Cache prezentă pe calculatorul utilizatorului

- apar informații despre memoria RAM a sistemului

Scenariu alternativ:

- nu se pot prelua informațiile despre componentele calculatorului

Cazul de utilizare 2: Testarea performantei procesorului

Actor: utilizatorul

Scenariu principal:

- utilizatorul deschide aplicația

- se încarcă fereastra principală

- apar date legate de procesorul calculatorului

- apar date despre memoria Cache prezentă pe calculatorul utilizatorului

- apar informații despre memoria RAM a sistemului

- utilizatorul apasa pe butonul “Test CPU”

- se incarca fereastra cu timpii pentru fiecare program care solicita procesorul (DigitsOfPi, Factorial, Factorization, Fibonacci, FastFourierTrasform, MatrixMultiplication)

- se va calucla si afisa un scor bazat pe timpii de executie

Scenariu alternativ:

- nu se pot prelua timpii de executie, va aparea un mesaj de eroare in consola

Cazul de utilizare 3: Testarea performantei a memorii RAM

Actor: utilizatorul

Scenariu principal:

- utilizatorul deschide aplicația

- se încarcă fereastra principală

- apar date legate de procesorul calculatorului

- apar date despre memoria Cache prezentă pe calculatorul utilizatorului

- apar informații despre memoria RAM a sistemului

- utilizatorul apasa pe butonul “Test RAM”

- se incarca fereastra si se va afisa scorul obtinut in urma unor alocari, realocari, dezalocari si utilizarea memoriei alocata

Scenariu alternativ:

- nu se poate prelua scorul obtinut in urma alocarii, realocarii, dezalocarii si utilizarea memoriei alocata

# 4. Design

Interfața grafică a aplicației este creată cu ajutorul Java Swing și a limbajului de programare C. Informațiile legate de specificațiile hardware sunt afișate într-o fereastră care conține un câmp de text. Sub câmpul de text se află două butoane. Unul dintre ele este folosit pentru testarea vitezei de transfer a fișierelor, iar celălalt este folosit pentru a testa viteza de execuție a unor operații aritmetice. Primul din ele deschide o fereastră nouă, în care se află o casetă de combinație, din care utilizatorul poate seta dimensiunea fișierelor pe care să fie făcute testele, și un câmp de text în care vor fi afișate rezultatele. Cel de-al doilea deschide o fereastră cu un câmp de text în care vor fi afișate rezultatele legate de timpul de execuție al operațiilor aritmetice.

Pentru a putea calcula viteza de transfer a fișierelor a fost necesară crearea unor fișiere pe care să fie făcute testele. Pentru acest lucru, am folosit clasa Random și am făcut o funcție GenerateRandomFile care primește ca și argumente dimensiunea fișierului și numele și crează un nou fișier aleator, generând numărul necesar de bytes cu metoda nextBytes din Random. Legat de directoarele între care se face transferul fișierelor pentru test, am creat două directoare ”in” și ”out” în directorul curent al proiectului.

Pentru a extrage specificațiile componentelor care ne interesează (CPU, memorie Cache, memorie RAM) am folosit clasa ManagementObjectSearcher, clasă din C foarte utilă pentru această sarcină. Cu ajutorul acestei clase se pot extrage orice informații de care avem nevoie legate de un anumit sistem, prin introducerea unui query în care se specifică ce componentă ne interesează.

Pentru timpii care trebuie măsurați în cadrul testării parametrilor de performanță (viteza de execuție a operațiilor și viteza de transfer a fișierelor între directoare) este folosită clasa Stopwatch. Această clasă are o serie de metode și proprietăți prin care ne oferă posibilitatea să măsurăm cu precizie ridicată timpii dintre două operații. Stopwatch folosește QueryPerformanceCounter (QPC), care reprezintă cea mai bună variantă când vine vorba de măsurători pentru intervale relativ scurte de timp care au loc pe același sistem sau pe o mașină virtuală. Această metodă oferă rezultate mult mai precise față de utilizarea directă a instrucțiunilor pe procesor RDTSC și RDTSCP, care pot oferi rezultate departe de adevăr pe unele versiuni de Windows.

# 5. Implementare

## 5.1. Specificațiile procesorului

Aplicația afișează cele mai importante proprietăți ale procesorului. Pentru a le extrage s-a folosit clasa IWbemServices. Când am instanțiat un obiect de tipul clasei, în argumentul constructorului s-a introdus un query specific, pentru a căuta în informațiile procesorului:

"select \* from Win32\_Processor".

**A computer screen shot of text

Description automatically generated**

Figure : Query pentru aflarea detaliilor despre processor

## 5.2. Specificațiile memoriei cache

Similar cu afișarea specificațiilor procesorului, aplicația afișează și cele mai relevante informații despre memoria cache. S-a folosit un obiect al aceleași clase IWbemServices, singurul lucru diferit fiind query-ul specificat ca și argument la crearea obiectului, în acest caz fiind:

"select \* from Win32\_CacheMemory".

**A computer screen with text

Description automatically generated**

Figure : Query pentru aflarea detaliilor despre memoria Cache

## 5.3. Specificațiile memoriei RAM

Datele legate de memoria RAM au fost extrase în aceeași manieră ca și cele despre procesor și memoria RAM. Stringul dat ca și argument la crearea obiectului IWbemServices este: "select \* from Win32\_PhysicalMemory".

A computer screen with text

Description automatically generated

Figure : Query pentru aflarea detaliilor despre memoria RAM

# 6. Testare si validare

Pentru verificarea informațiilor pe care aplicația le oferă despre procesor, am comparat rezultatele cu cele oferite de Windows prin intermediul Task Manager. După cum se poate vedea mai jos, informațiile care se găsesc atât în aplicație cât și în Task Manager (numele procesorului, viteza clock-ului – base speed în Task Manager, dimensiunea cache-ului de nivel 2 și de nivel 3, numărul de procesoare logice, statusul virtualizării și gradul de utilizare) coincid.

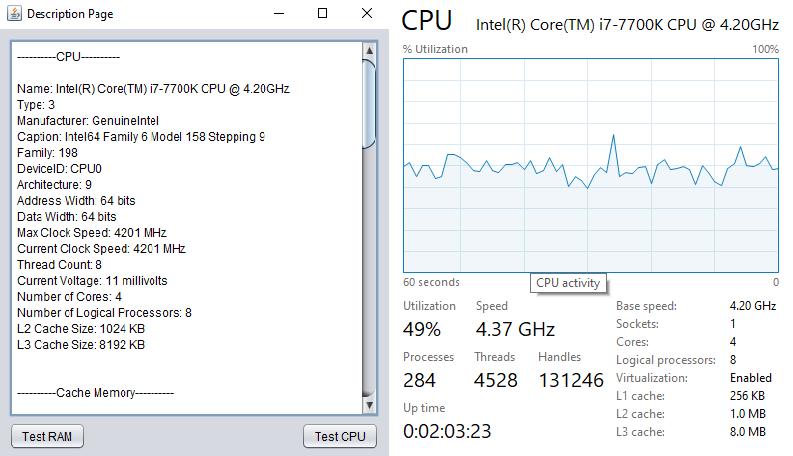
****

Figure : Inforamtii procesor preluate din aplicatie in comparatie cu cele preluate din Task Manager

Datele legate de memoriile cache, extrase din aplicație, pot fi observate mai jos. Se poate vedea că dimensiunea lor coincide cu dimensiuniile specificate pentru memoriile cache în figura de mai sus.

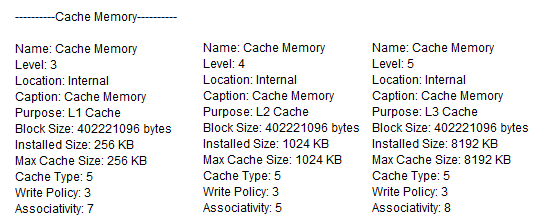
****

Figure : Specificatiile memoriilor cache, luate din aplicatie

Datele legate de memoriile RAM, extrase din aplicație, pot fi observate mai jos. Se poate vedea că dimensiunea lor coincide cu dimensiunea specificata in Task Manager.

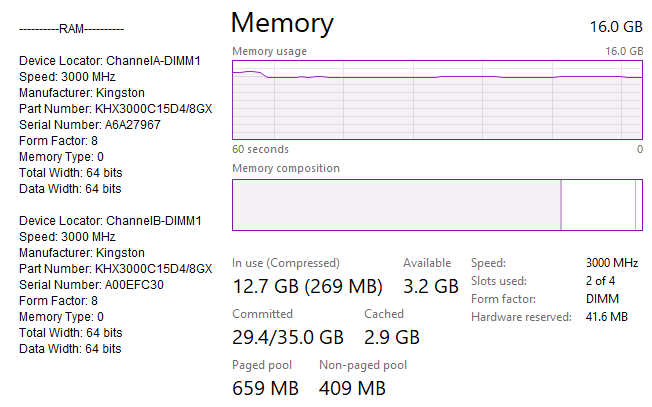
****

Figure : Inforamtiile memoriei RAM preluate din aplicatie in comparatie cu cele preluate din Task Manager

* Comparație între 2 procesoare folosind aplicația

Pentru a testa modul în care aplicația efectuează testarea procesorului, voi realiza o comparație între 2 sisteme cu procesoare diferite și voi compara rezultatele obținute prin intermediul aplicației cu rezultate de pe site-urile de specialitate.

Specificațiile procesoarelor pentru care s-a realizat testarea sunt următoarele:

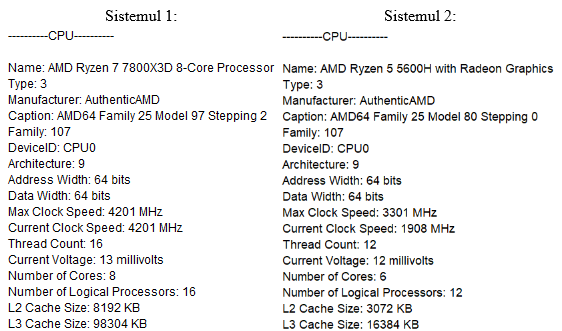
****

Figure : Procesorul sistemului 1 Procesorul sistemului 2

**A screenshot of a computer

Description automatically generated**

Figure : Rezultatele testeleor pe cele 2 procesoare

**A screenshot of a computer

Description automatically generated**

Figure : Rezulatele testelor memoriei RAM a celor 2 sisteme

# 7. Concluzii

Obiectivului acestui proiect a fost de a crea un program care să evalueze performanța unui calculator cu o interfață prietenoasă. Acest demers nu a reprezentat doar o provocare tehnică, ci și o oportunitate de a înțelege mai bine cum funcționează calculatoarele, analizând operațiunile pe care de obicei le luăm de-a gata.

Unul dintre cele mai solicitante aspecte ale proiectului a fost dezvoltarea testelor de performanță, care au evaluat tipul de procese, frecvența, memoria, viteza de transfer a datelor și execuția operațiilor aritmetice și logice. Colectarea acestor date diverse într-un sistem de evaluare coerent a fost ca un puzzle care a necesitat soluții inovatoare.

Proiectul a necesitat cercetare intensivă, explorând diverse abordări, multe dintre ele dovedindu-se nepotrivite din diverse motive. Cu toate acestea, perseverența și inovația care au rezultat din aceste provocări m-au ajutat să dezvolt un sistem unic, eficient și precis de evaluare a performanței.

Acest proiect nu numai că m-a ajutat să înțeleg mai bine calculatoarele, ci și a evidențiat importanța evaluării performanței pentru a le utiliza mai eficient. Am combinat cunoștințele mele de programare cu înțelegerea lucrurilor tehnice ale calculatoarelor pentru a crea o unealtă care ne-a ajutat să înțelegem mai bine cum funcționează calculatoarele. În concluzie, acest proiect a fost o experiență interesantă care m-a ajutat să înțeleg atât partea de software, cât și partea de hardware a calculatoarelor.

# 8. Bibliografie

1. <https://ro.wikipedia.org/wiki/Procesor>
2. <https://stackoverflow.com/questions/8067282/how-to-develop-performance-benchmark-program>
3. <https://levelup.gitconnected.com/8-ways-to-measure-execution-time-in-c-c-48634458d0f9>
4. <https://www.techtarget.com/searchstorage/definition/cache-memory>
5. <https://stackoverflow.com/questions/32763841/benchmarking-ram-performance-uwp-and-c-sharp>
6. https://simplesolution.dev/java-get-physical-memory-ram-information-oshi-library/
7. <https://stackify.com/how-to-monitor-cpu-memory-and-disk-usage-in-java/>
8. <https://learn.microsoft.com/en-us/windows/win32/api/wbemcli/nn-wbemcli-iwbemservices>