Testarea parametrilor de performanță ai unui PC

Iftene Ioan-Florin
Grupa 30231

Testarea parametrilor de performanță ai unui PC

Cuprins

1. Introducere 3
   1.1. Context 3
   1.2. Specificații 3
   1.3. Obiective 3
2. Studiu bibliografic 3
   2.1. Procesorul 3
   Memoria cache a procesorului 4
   2.2. Frecvența 4
   2.3. Dimensiunea memoriei 4
   2.4. Viteza de transfer a datelor 5
   2.5. Viteza de execuție a operațiilor aritmetice 5
3. Propunere de proiect 5
4. Plan de proiect 6
5. Analiză 6
   5.1. Datele despre procesor 6
   5.2. Datele despre memoria cache și memoria RAM 7
   5.3. Viteza de execuție a operațiilor aritmetice 7
   5.4. Viteza de transfer a fișierelor 7
   5.5. Cazuri de utilizre 8
6. Design 9
7. Implementare 10
   7.1. Specificațiile procesorului 10
   7.2. Specificațiile memoriei cache 10
   7.3. Specificațiile memoriei RAM 11
   7.4. Testarea vitezei de transfer a fișierelor între directoare 11
   7.5. Testarea vitezei de execuție a operațiilor aritmetico-logice 11
8. Testare și validare 12

Sistemul 2: 13

9. Concluzii 14
10. Referințe 15

1. Introducere

1.1. Context

Scopul proiectului este de a crea un program de testare a parametrilor de perofrmanță a unui PC. Parametrii de performanță pot fi folosiți pentru a compara computere între ele și ne arată cât de puternic este un calculator. Acești parametrii pot fi influențați de orice modificare adusă la nivelul procesorului, al frecvenței de lucru, al memoriei sau al vitezei de transfer a datelor.

Acest program poate fi folosit de către oricine dorește să vadă de ce este capabil calculatorul lor, ce proprietăți are și cât de bine se poate descurca în îndeplinirea anumitor obiective.

1.2. Specificații

Programul va fi realizat folosind Visual Studio și limbajul de programare C# și rezultatele se vor putea vedea într-o aplicație de tip Windows Forms. Utilizatorul va putea vedea caracteristicile procesorului pe care îl deține PC-ul, ale memoriei Cache și memoriei RAM, dar și timpul de execuție al anumitor 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.

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. [1]

Procesoarele pot identificate cu ajutorul a doi parametrii principali: viteză și lățime. Viteza unui procesor este măsurată în megahertz (MHz), adică în milioane de cicluri pe secundă. Lățimea unui procesor este determinată de regiștrii interni, de magistrala de intrare și ieșire a datelor și de magistrala de adrese de memorie. [2]

Memoria cache a procesorului

- *L1 Cache*: reprezintă memoria cache construită în interiorul procesorului și este cunoscută ca și memorie cache de nivel 1. Această memorie reține cele mai recente date, deci este prima în care se caută când sunt necesare anumite date. Este extrem de rapidă, dar are o dimensiune foarte mică.
- *L2 Cache:* Această memorie se află pe un cip de lângă procesor și este numită memorie cache ne nivelul 2. În acest loc se află date recente, care nu se regăsesc în L1 Cache. Are dimensiunea puțin mai mare decât cea de nivel 1. Unele procesoare au atât memoria cache de nivel 1, cât și cea de nivel 2 construite în interiorul lor, ceea ce duce la prezența unei memorii cache de nivel 3 pe cipul de lângă procesor.

Memoria cache poate fi de tip Unified Cache, sau de tip Split Cache. Cea unified stochează la un loc datele și instrucțiunile, pe când cea de tip split este alcătuită din două unități: una pentru stocarea instrucțiunilor, numită I-cache, și una pentru stocarea datelor, numită D-cache. [3][4]

Din punct de vedere al mapării datelor, memoria cache poate folosi mapare directă, complet-asociativă sau set-asociativă (numită și n-way set associative mapping, care mapează o locație din memoria principală la oricare ”N” locații din L1 cache). În ceea ce privește politica de scriere în memoria cache, unele folosesc write-back iar altele write-through. [4]

2.2. Frecvența

Viteza ciclului de ceas a unui calculator este măsurată ca și frecvență și este exprimată, în general, ca și număr de cicluri pe secundă. Un oscilator de cristal controlează viteza clock-ului folosind un quartz de argint. Când este aplicată tensiune pe quartz, acesta începe să oscileze periodic armonic, perioada fiind dictată de forma și dimensiunea cristalului de argint. În urma aplicării tensiunii, cristalul emană curent alternant, care reprezintă de fapt semnalul de ceas care formează baza de timp în care operează calculatorul. Un PC obișnuit rulează milioane de astfel de cicluri pe secundă, de aceea viteza se măsoară în megahertz (un hertz = un ciclu pe secundă). [2][5]

2.3. Dimensiunea memoriei

Dimensiunea memoriei se referă la capacitatea memoriei, dată de regiștrii de adresă disponibili în procesor și se măsoară în bytes.

Capacitatea memoriei reprezintă cantitatea de memorie care se poate folosi de către un dispozitiv electronic, precum un calculator, un laptop, un smartphone sau alte dispozitive inteligente. Performanța unui dispozitiv și eficiența operațiilor I/O este dependentă de capacitateea memoriei

În cazul procesoarelor cu o arhitectura pe 32 de biți, ele pot adresa memorii cu capacitate maximă de 4 GB. Cele cu arhitectură pe 64 de biți pot adresa memorie cu capacitate nelimitată. Sistemele de operare influențează și ele capacitatea adresabilă a memoriei. Capacitatea memoriei unui dispozitiv este dependentă de memoria de stocare principală, de memoria RAM și de orice alte dispozitive externe de stocare. De obicei, memoria principala e volatilă, iar cea secundară este nevolatilă. [6]

2.4. Viteza de transfer a datelor

Viteza de transfer a datelor reprezintă cantitatea de date digitale care sunt transferate între două locuri (puncte) într-o anumită perioadă de timp. Rata de transfer a datelor reprezintă de fapt o măsurare a vitezei cu care componente din rețea pot schimba date. Este măsurată în bits/bytes pe secundă. Din motive practice, se folosesc măsurătorile în MBps sau KBps. [7]

Pentru a măsura viteza de transfer a datelor, aplicația va genera fișiere de diferite dimensiuni și le va transfera între directoare diferite, de un anumit număr de ori, apoi se va realiza media timpilor efectuării acestor transferuri.

2.5. Viteza de execuție a operațiilor aritmetice

Viteza de execuție reprezintă viteza cu care un dispozitiv computational poate executa operații aritmetico-logice ca și instrucțiuni. Se poate măsura cu ajutorul funcției clock(), din <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.[8] O altă metodă de măsurare este folosirea unui obiect de tipul Stopwatch, accesibil în C#. Acest obiect poate fi folosit pentru a măsura timpul în milisecunde sau în pulsuri de ceas. Se aplelează metoda Start() înaintea secțiunii de cod care se dorește a fi măsurată și metoda Stop() imediat după și astfel vom avea în obiectul folosit timpul rezultat.[11]

3. Propunere de proiect

Acest proiect își propune să ofere utilizatorului o aplicație pe care să o poată utiliza pentru a-și testa și cunoaște calculatorul.

Prin această aplicație, utilizatorul va vedea ce procesor are (numele procesorului, arhitectura, tipul procesorului, dimensiunea spațiului de adrese, frecvența, familia, numărul de nuclee), date despre memoria RAM (numele producătorului, frecventa, capacitatea, latimea de bandă), date despre memoria cache (nivelul memoriei, dimensiunea, tipul, politica de scriere,

tipul de mapare al datelor), timpii de execuție ai unor operatii aritmetice, precum adunarea, scăderea, înmulțirea și împărțirea și viteza medie de transfer al unor fișiere de diferite dimensiuni (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 MB).

Pentru a folosi aplicația, utilizatorul va rula executabilul și astfel se va deschide o fereastră pe care vor apărea automat datele procesorului, ale memoriei Cache, ale memoriei RAM și viteza de execuție a operațiilor aritmetico-logice. Pentru a testa viteza de transfer a datelor, în aceeași fereastră, utilizatorul va avea opțiunea să selecteze că dorește acest lucru, prin apăsarea unui buton. După apăsarea butonului, va trebui să aleagă dimensiunea fișierelor pe care se va măsura viteza de transfer, iar apoi va putea vedea rezultatele obținute.

4. Plan de proiect

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

- 10 Octombrie 2022: alegerea temei proiectului
- 24 Octombrie 2022: scrierea primelor 2 capitole din documentație: Introducere și Studiu bibliografic
- 7 Noiembrie 2022: scrierea următoarelor 3 capitole din documentație: Propunere de proiect, Plan de proiect, Analiză
- 21 Noiembrie 2022: crearea interfeței grafice a aplicației, afișarea datelor despre procesor și scrierea următoarelor capitole din documentație
- 5 Decembrie 2022: afișarea datelor despre memoria cache și memoria RAM
- 19 Decembrie 2022: afișarea datelor legate de viteza de execuție a operațiilor aritmetice și viteza de transfer a fișierelor între directoare.
- 9 Ianuarie 2023: predarea proiectului în forma finală

5. Analiză

5.1. Datele despre procesor

Pentru ca utilizatorul să poată vedea date despre procesorul pe care îl utilizează calculatorul lui, se va folosi clasa ManagementObjectSearcher din C#, care primește ca și parametru un obiect de tip ObjectQuery, care reprezintă o interogare prin intermediul căreia se pot selecta toate datele despre o anumită componentă a calculatorului. Pentru a extrage datele despre procesor, obiectul de tip ObjectQuery trimis ca si argument constructorului obiectului de tip ManagementObjectSearcher va fi creat folosind stringul ”select * from Win32_Processor”. Se va parcurge apoi mulțimea mare de date extrase și se vor selecta doar cele mai relevante pentru utilizator, spre a fi afișate. [9]

5.2. Datele despre memoria cache și memoria RAM

Se va proceda similar cu modul de extragere al datelor pentru procesor, doar că se va schimba stringul din ObjectQuery.

Datele extrase pentru memoria cache, legate de tipul de mapare, locatie, nivel și politica de scriere sunt primite ca și coduri, fiecare cod reprezentând altceva. Pentru ca utilizatorul să înțeleagă despre ce e vorba, pe interfața aplicației va apărea 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 [10]. Se vor folosi dicționare, usor de utilizat in C#, pentru a mapa însemnătatea codurilor.

Pentru memoria RAM se va proceda exact la fel, folosindu-se stringul specific memoriei RAM și se vor afișa în aplicație doar datele relevante pentru utilizator.

5.3. Viteza de execuție a operațiilor aritmetice

Pentru a măsura viteza de execuție a operațiilor aritmetice se va folosi metoda utilizării unui obiect de tipul StopWatch, descrisă în capitolul 2, subcapitolul 2.5. Se vor efectua mai multe măsurări, se va face media lor și se va afișa utilizatorului în interfața aplicației.

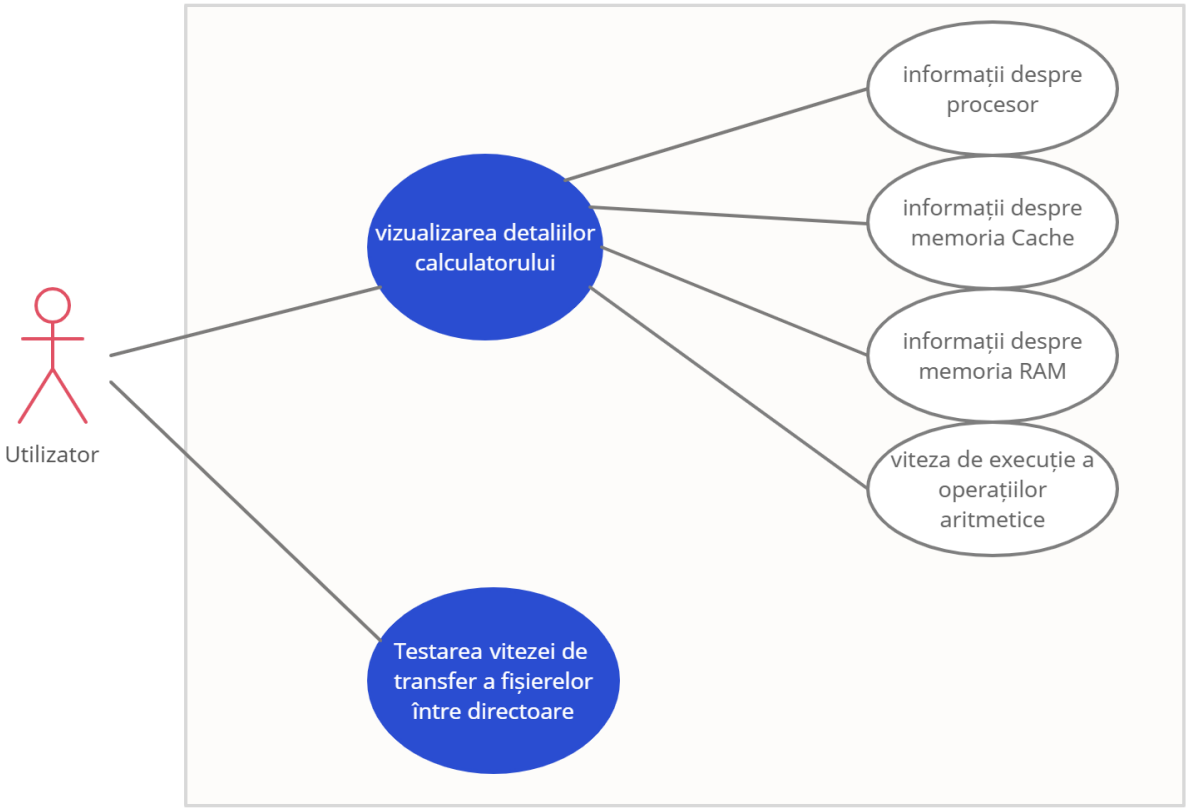
5.4. Viteza de transfer a fișierelor

Pentru a testa viteza de transfer a datelor, se va folosi un algoritm care generează un fișier aleator, de o dimensiune data. Dimensiunea va fi oferită aplicației de către utilizator, acesta având opțiunea sa aleagă fișiere cu dimensiune multiplă a lui 2 MB. Va fi folosită apoi clasa StopWatch() pentru a crea o variabilă cu care se va măsura timpul în care se va copia fișierul într-un alt director și se va începe procedura de copiere a fișierului, dupa care se va salva timpul rezultat și se 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.

Obiectul de tip StopWatch ne va ajuta să măsurăm timpul în milisecunde, deci, pentru a-i oferi utilizatorului un rezultat de forma MBps, îl vom obține utilizând formula:

$$\text{VitezaDeTransfer [MBps]} = \frac{\text{dimensiunea fișierului } [MB] * 1000}{\text{timpul de transfer măsurat } [ms]}$$

5.5. Cazuri de utilizre



Figură 1: 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
  - se încarcă rezultatele vitezei de execuție a operațiilor aritmetice pe procesorul deținut

  Scenariu alternativ:
  - nu se pot prelua informațiile despre componentele calculatorului
  - se va afișa un mesaj de eroare

- Cazul de utilizare 2: Testarea vitezei de transfer a fișierelor între directoare

  Actor: utilizatorul

  Scenariu principal:
  - utilizatorul deschide aplicația
  - se încarcă fereastra principală
  - utilizatorul apasă butonul pentru testarea vitezei de transfer
  - utilizatorul selectează dimensiunea fișierului pe care să se facă testarea
  - apar rezultatele testelor în interfața aplicației

  Scenariu alternativ:
  - nu s-a putu genera un fișier de dimensiunea selectată
  - se va afișa un mesaj de eroare

6. Design

Interfața grafică a aplicației este creată cu ajutorul Windows Forms și a limbajului de programare C#. Informațiile legate de specificațiile hardware sunt afișate într-o formă care conține un text box. Sub text box se află două butoane. Unul din 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 nouă formă, în care se află un combo box, de unde utilizatorul poate seta dimensiunea fișierelor pe care să fie făcute testele, și un text box în care vor fi afișate rezultatele. Cel de-al doilea deschide o formă cu un text box 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 2 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.[12]

7. Implementare

7.1. Specificațiile procesorului

Aplicația afișează cele mai importante proprietăți ale procesorului. Pentru a le extrage s-a folosit clasa ManagementObjectSearcher. 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"`.

Unele informații legate de procesor, precum numele, producătorul, lățimea de date, viteza clock-ului, dimensiunile memoriilor cache interne, au putut fi extrase direct. Pentru altele (tipul procesorului, familia, arhitectura) a fost necesară crearea unor dicționare care să decodifice valoarea primită de la obiectul de tip ManagementObjectSearcher pentru datele respective. Un exemplu de dicționar, care extrage familia procesorului, poate fi vizualizat în următoarea imagine:

```
Dictionary<string, string> processorFamily = new Dictionary<string, string>(){
    {"1", "Other"},
    {"2", "Unknown"},
    {"3", "8086"},
    {"198", "Intel(R) Core(TM) i7 processor"},
    {"205", "Intel(R) Core(TM) i5 processor"},
    {"207", "Intel(R) Core(TM) i9 processor"},
    {"206", "Intel(R) Core(TM) i3 processor"},
    {"182", "AMD Athlon(TM) XP Family"},
    {"131", "AMD Athlon(TM) 64 Processor Family"}
};
```

Figură 2: Dicționar pentru familia procesorului

7.2. 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 ManagementObjectSearcher, singurul lucru diferit fiind query-ul specificat ca și argument la crearea obiectului, în acest caz fiind: `"select * from Win32_CacheMemory"`.

În acest caz, informațiile pentru care a fost necesară crearea de dicționare sunt nivelul memoriei cache, tipul, politica de scriere și asociativitatea. Datele legate de numele memoriei, scop, dimensiunea blocului de date și a memoriei au putut fi extrase direct cu ajutorul obiectului de tip ManagementObjectSearcher.

```
Dictionary<string, string> cacheAssociativity = new Dictionary<string, string>(){
    {"1", "Other"},
    {"2", "Unknown"},
    {"3", "Direct Mapped"},
    {"4", "2-way Set-Associative"},
    {"5", "4-way Set-Associative"},
    {"6", "Full Associative"},
    {"7", "8-way Set-Associative"},
    {"8", "16-way Set-Associative"},
    {"9", "Error"}
};
```

Figură 3: Dicționar pentru tipul de asociativitate al memoriei cache

7.3. 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 ManagementObjectSearcher este: `"select * from Win32_PhysicalMemory".`

Nu a fost necesară utilizarea de dicționare pentru afișarea datelor, aproape fiind extrase în mod direct, în forma dorită. Singura informație pentru care au fost necesare modificări a fost legată de capacitatea memoriei, care venea în bytes și am transformat-o în GB, pentru a fi mai ușor de citit de către utilizator.

7.4. Testarea vitezei de transfer a fișierelor între directoare

Pentru a testa viteza de transfer a fișierelor între directoare am ales să mai creez o formă special pentru această operație. Aceasta se va deschide prin apăsarea butonului ”Test Disk” din forma principală a aplicației. Am adăugat în forma pentru testarea discului un drop-down list din care utilizatorul va alege dimensiunea fișierelor pe care dorește să facă testele. După ce a fost selectată o dimensiune, testarea va începe și datele vor fi afișate în timp real, pe măsură ce se termină câte o iterație de efectuat, iar la final va fi afișată media.

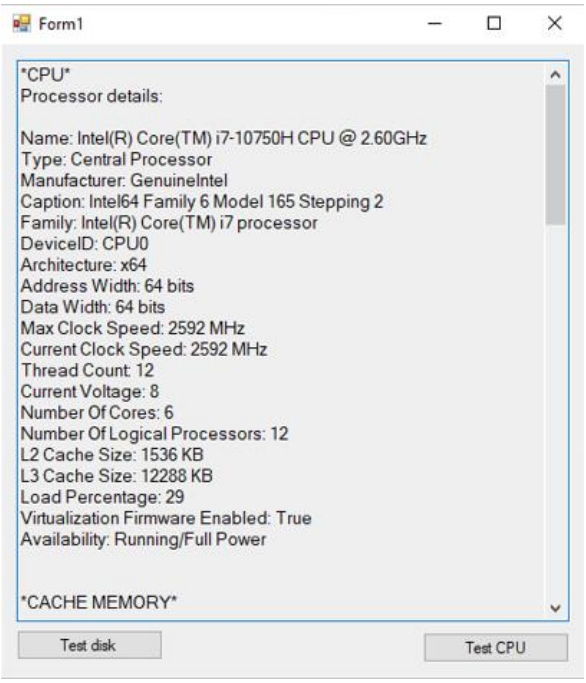
Pentru a testa, am creat 2 foldere ”in” și ”out” în folderul curent al aplicației, între care se va efectua transferul. Pentru fișierul care urmează a fi transferat, a fost făcută metoda GenerateRandomFile, care primește dimensiunea fișierului ca și argument și îl crează în folderul ”in” dacă nu există. Am folosit apoi o variabilă de tip Stopwatch pe care o pornesc înaintea efectuării operații de copiere a fișierului din folderul ”in” în folderul ”out”, operație disponibilă prin intermediul metodei Copy din File, și o opresc după efectuarea operației. Astfel, dacă împărțim dimensiunea fișierului în MB la timpul în care s-a efectuat transferul, accesibil prin intermediul metodei ElapsedMilliseconds din clasa Stopwatch, se va obține viteza de transfer a fișierului între cele 2 directoare. Pentru ca rezultatele să fie cât mai realiste, am efectuat 5 transferuri pentru fiecare dimensiune aleasă și apoi am afișat media lor.

7.5. Testarea vitezei de execuție a operațiilor aritmetico-logice

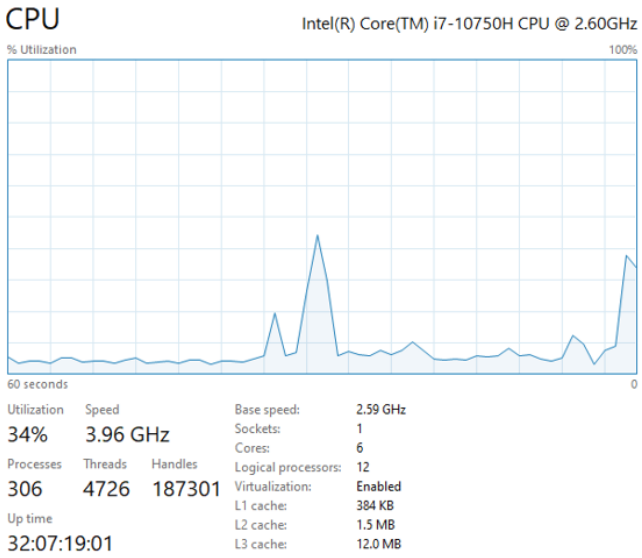
Testarea vitezei de execuție a operațiilor aritmetico-logice a fost realizată cu ajutorul unei variabile de tip Stopwatch, prin care se începe măsurarea înaintea efectuării unei operații și se oprește imediat dupa finalizarea ei. Au fost efectuate 100 de iterații pe numere aleatoare pentru toate operațiile măsurate, iar rezultatele au fost realizate efectuând media tuturor iterațiilor și sunt exprimate în tick-uri de ceas, cu ajutorul metodei ElapsedTicks din intermediul clasei Stopwatch. Operațiile aritmetice a căror performanță s-a măsurat sunt adunarea, scăderea, înmulțirea și împărțirea. Pentru toate cele 4 s-au efectuat câte 2 seturi de măsurători: o dată pentru numere întregi și o dată pentru numere de tip double. Operațiile logice pentru care s-au efectuat măsurători sunt AND, OR, XOR, NOT.

8. Testare și 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.



Figură 5: Informații procesor luate din aplicație



Figură 4: Informații procesor luate 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 4.

CACHE MEMORY

-> Primary Cache Memory Details:	-> Secondary Cache Memory Details:	-> Tertiary Cache Memory Details:
Name: Cache Memory	Name: Cache Memory	Name: Cache Memory
Level: Primary	Level: Secondary	Level: Tertiary
Location: Internal	Location: Internal	Location: Internal
Caption: Cache Memory	Caption: Cache Memory	Caption: Cache Memory
Purpose: L1 Cache	Purpose: L2 Cache	Purpose: L3 Cache
Block Size: 1024 bytes	Block Size: 1024 bytes	Block Size: 1024 bytes
Installed Size: 384 KB	Installed Size: 1536 KB	Installed Size: 12288 KB
Max Cache Size: 384 KB	Max Cache Size: 1536 KB	Max Cache Size: 12288 KB
Cache Type: Unified	Cache Type: Unified	Cache Type: Unified
Write Policy: Write Back	Write Policy: Write Back	Write Policy: Write Back
Associativity: 8-way Set-Associative	Associativity: 4-way Set-Associative	Associativity: 16-way Set-Associative

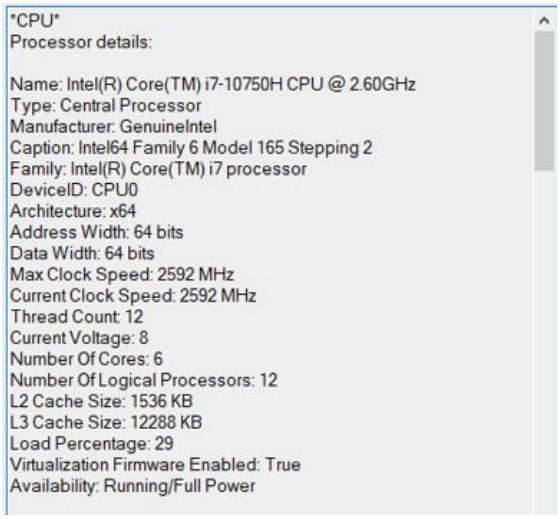
Figură 6: Specificațiile memoriilor cache, luate din aplicație

- 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:

Sistemul 1:



Figură 7: Procesorul sistemului 1

Sistemul 2:

Figură 8: Procesorul sistemului 2

Rezultatele obținute pentru fiecare procesor se pot vedea în tabelul de mai jos:

Operația testată	**Sistemul 1**	**Sistemul 2**
Adunare (numere întregi)	0.28 ticks	0.29 ticks
Adunare (numere flotante)	0.33 ticks	0.41 ticks
Scădere (numere întregi)	0.28 ticks	0.29 ticks
Scădere (numere flotante)	0.36 ticks	0.30 ticks
Înmulțire (numere întregi)	0.31 ticks	0.29 ticks
Înmulțire (numere flotante)	0.35 ticks	0.31 ticks
Împărțire (numere întregi)	0.25 ticks	0.32 ticks
Împărțire (numere flotante)	0.3 ticks	0.28 ticks
AND	0.28 ticks	0.31 ticks
OR	0.28 ticks	0.35 ticks
XOR	0.29 ticks	0.32 ticks
NOT	0.31 ticks	0.34 ticks
Generarea primelor 1000 numere prime	2523.35 ticks	2659.49 ticks
Generarea primelor 1000 numere din șirul lui Fibonacci	476.32 ticks	1384.21 ticks

Tabel 1: Rezultatele testelor pe cele 2 procesoare

Analizând datele din tabelul 1, am ajuns la concluzia că procesorul sistemului 1 este cu 34.7% mai eficient decât procesorul sistemului 2, în ceea ce privește efectuarea tuturor operațiilor testate. Pentru operațiile simple (adunare, scadere, înmulțire, împărțire, AND, OR, XOR, NOT), diferența dintre cele 2 procesoare este de doar 5.2%, mai eficient fiind tot procesorul sistemului 1.

Pe lângă rezultatele extrase în urma măsurării cu aplicația creată, am realizat comparația între cele 2 procesoare cu ajutorul a 3 site-uri specializate în benchmark, pentru a vedea diferența dintre concluziile obținute cu ajutorul programului meu și cele specificate pe respectivele site-uri. Conform cpu.userbenchmark.com, procesorul sistemului 1 este cu 32.6% mai performant decât cel al sistemului 2. www.cpubenchmark.net indică o diferență de 29.3%, iar www.cpu-monkey.com specifică o un procent de 31.2% în favoarea procesorului primului sistem.

9. Concluzii

În urma acestui proiect am reușit realizarea unei aplicații de tip benchmark care specifică detalii legate de componentele unui calculator (procesor, memorie cache, memorie RAM) și îi testează performanțele în ceea ce privește viteza de transfer a datelor și viteza de execuție a operațiilor aritmetico-logice. Pentru a putea compara mai bine 2 sisteme între ele din punctul de vedere al procesorului, aplicația testează și viteza cu care se realizează generarea primelor 1000 de numere prime și a primelor 1000 de numere din șirul lui Fibonacci. Astfel, utilizatorul are un program cu care poate vedea ce componente are, cât de bine se descurcă în realizarea unor cerințe simple și poate să și compare cu performanțele altor sisteme.

10. Referințe

[1] Procesor. (fără an). Preluat de pe https://ro.wikipedia.org/wiki/Procesor

[2] Processor Specifications. (2001, June 8). Preluat de pe https://www.informit.com/articles/article.aspx?p=130978&seqNum=4

[3] Cache Memory Performance. (fără an). Preluat de pe https://www.geeksforgeeks.org/cache-memory-performance/

[4] Cache Memory. (fără an). Preluat de pe https://www.techtarget.com/searchstorage/definition/cache-memory

[5] Clock Rate. (fără an). Preluat de pe https://en.wikipedia.org/wiki/Clock_rate

[6] Memory Capacity. (fără an). Preluat de pe https://www.techopedia.com/definition/8297/memory-capacity

[7] What is Data Transfer Rate? (fără an). Preluat de pe https://www.geeksforgeeks.org/what-is-data-transfer-rate/

[8] Ferreira, C. (2020, April 30). *8 Ways to Measure Execution Time in C/C++*. Preluat de pe Level Up Coding: https://levelup.gitconnected.com/8-ways-to-measure-execution-time-in-c-c-48634458d0f9

[9] Win32_Processor class. (fără an). Preluat de pe https://learn.microsoft.com/en-us/windows/win32/cimwin32prov/win32-processor?redirectedfrom=MSDN

[10] Win32_CacheMemory class. (fără an). Preluat de pe https://learn.microsoft.com/en-us/windows/win32/cimwin32prov/win32-cachememory

[11] Stopwatch Class. (fără an). Preluat de pe https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch?view=netcore-3.1

[12] Acquiring high-resolution time stamps. (2022, January 4). Preluat de pe https://learn.microsoft.com/en-gb/windows/win32/sysinfo/acquiring-high-resolution-time-stamps?redirectedfrom=MSDN#Guidance