ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА КОМП’ЮТЕРНИХ ТЕХНОЛОГІЙ

**Лабораторна робота №1**

«Дослідження особливостей архітектури підсистеми»

**з курсу «Архiтектура та програмне забезпечення обчислювальних систем»**

Виконав:

студент групи ПА-22-2

Овдієнко Андрій Володимирович

Дніпро, 2023

**Зміст**

[**Завдання** 3](#_Toc129688365)

[**Опис характеристик обчислювальної техніки на якій виконувався чисельний експеримент** 5](#_Toc129688366)

[**Cкріни про те, що перемноження відбувається вірно на прикладі невеликих матриць** 14](#_Toc129688367)

[**Створити таблиці, куди занести результати засічок часу** 15](#_Toc129688368)

[**Створити графіки, на яких відображати дані з таблиць** 16](#_Toc129688369)

[**Код для перевірки програми на невеликих числах, при N=3** 19](#_Toc129688370)

[**Код програми, при N=22000** 23](#_Toc129688371)

[**Висновки** 26](#_Toc129688372)

**Завдання**

На мові С/C++ напишіть програму яка буде перемножувати квадратну матрицю розміром NxN на вектор розмірність N. Величину N виберіть таким чином, щоб максимально використати всю можливу оперативну пам’ять. Обчисліть затрати часу на дану дію. Перемножте також вектор на матрицю. Порівняйте затрачений час в обох випадках. Зробіть висновки. Запропонуйте зміну алгоритму таким чином, щоб затрачений час був приблизно однаковий.

Пояснення: Найшвидшим способом обходу є прямий послідовний. Це означає, що після звернення у програмі до деякого елемента відбувається звернення до елемента, що у пам'яті йде слідом за ним. Розглянемо розміщення у пам'яті двовимірного масиву у програмі мовою С/C++.

float A[N][N];

Відомо, що в мові С/C++ масиви розташовуються в пам'яті рядками (спочатку йдуть елементи першого рядка, потім елементи другого рядка і т.д.). Отже, у пам'яті він розміститься так:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **…** | A0,0 | A0,1 | A0,2 | … | A0,N-1 | A1,0 | A1,1 | A1,2 | … | A1,N-1 | … | AN-1,0 | AN-1,1 | AN-1,2 | … | AN-1,N-1 | **…** |

Отримуємо два варіанти перебору елементів масиву:

|  |  |
| --- | --- |
| **Швидко:**  **for** (i=0;i<N;i++)  **for** (j=0;j<N;j++) A[i][j]=x; | **Повільно:**  **for** (j=0;j<N;j++)  **for** (i=0;i<N;i++) A[i][j]=x; |

Розглянемо задачу перемноження двох квадратних матриць N×N. Якщо безпосередньо запрограмувати відому формулу: , наприклад, мовою С/C++, отримаємо наступний фрагмент програми:

for (i=0;i<N;i++)

for (k=0;k<N;k++)

for (j=0;j<N;j++) C[i][k]+=A[i][j]\*B[j][k];

Зауважимо, що у цьому випадку масив A перебирається рядками, а масив B - по стовпцям (дивимося на внутрішній цикл). Знаючи, що масиви в мові С/C++ зберігаються рядками, приходимо до висновку, що елементи масиву A перебираються послідовно, а елементи масиву B - ні. В даному випадку порядок обходу масиву C практично не важливий, оскільки між зверненнями до різних його елементів проходить багато часу. Щоб прискорити програму, потрібно, щоб принаймні у внутрішньому циклі елементи масивів перебиралися послідовно. Для цього необхідно або заздалегідь транспонувати масив B, або переставити цикли таким чином:

for (i=0;i<N;i++)

for (j=0;j<N;j++)

for (k=0;k<N;k++) C[i][k]+=A[i][j]\*B[j][k];

За результатами виконання лабораторної роботи скласти звіт:

Виміряти час перемноження матриці на вектор, вектора на матрицю, а також час оптимізованого способу перемноження вектора на матрицю. Виміряти час для розмірів матриці від N=1000 до максимального розміру матриці, який вдалося розмістити в оперативній пам'яті з кроком 1000.

* До звіту додати скріни або опис характеристик обчислювальної техніки на якій виконувався чисельний експеримент
* Додати до звіту скріни про те, що перемноження відбувається вірно на прикладі невеликих матриць.
* Виконати перемноження та засікти час для різних типів даних: int, float, double.
* Створити таблиці, куди занести результати засічок часу.
* Створити графіки, на яких відображати дані з таблиць.
* Порівняти час перемноження різних типів даних та різних способів перемноження.

**Опис характеристик обчислювальної техніки на якій виконувався чисельний експеримент**

Для демонстрації обчислювальної техніки використано системну інформацію ноутбука Mac OS та програму “MacCPUID”.

Системна інформація:

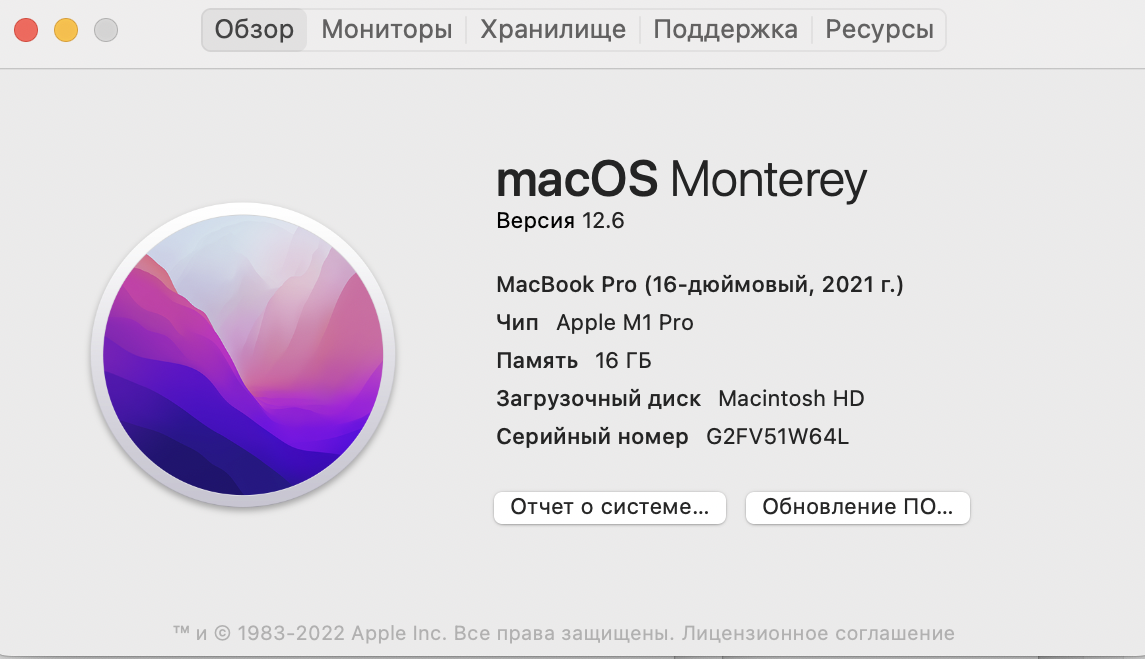
****

Рисунок 1.1.1 - Характеристики процесора

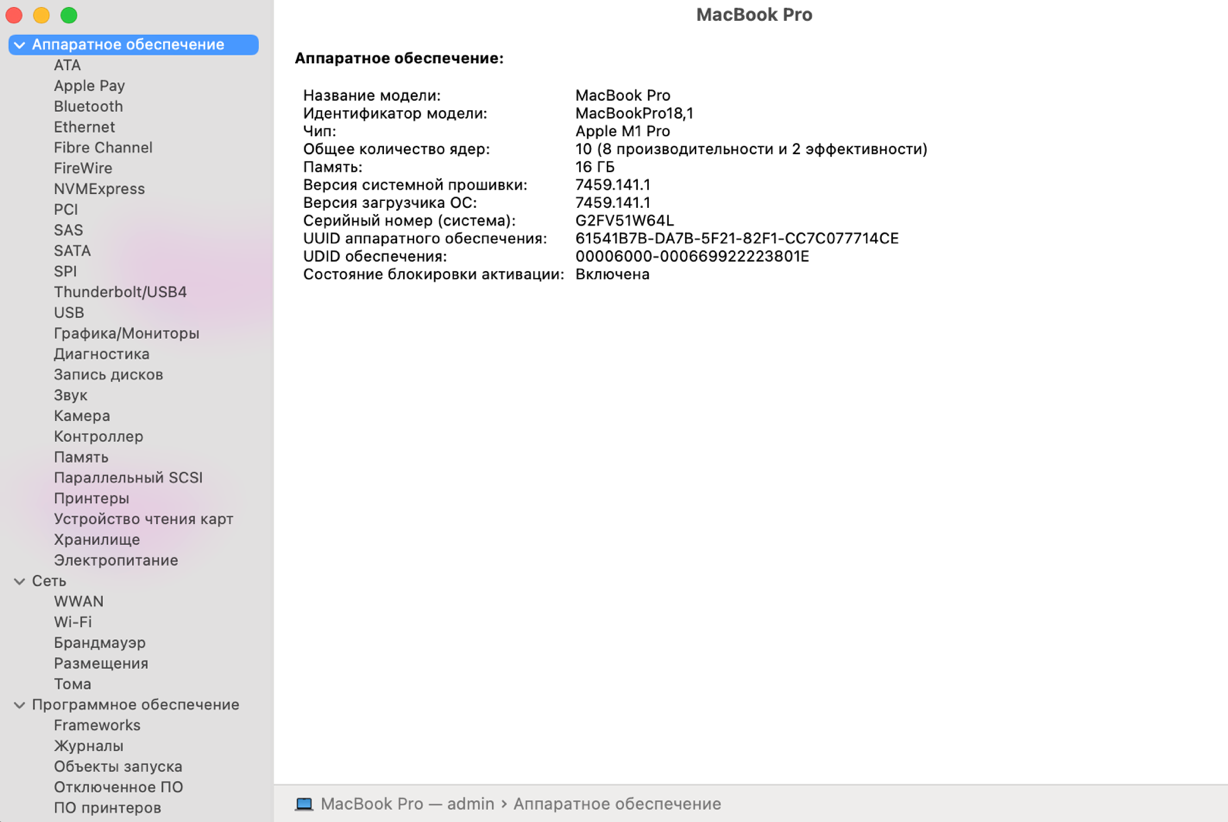
****

Рисунок 1.1.2 - Характеристики процесора

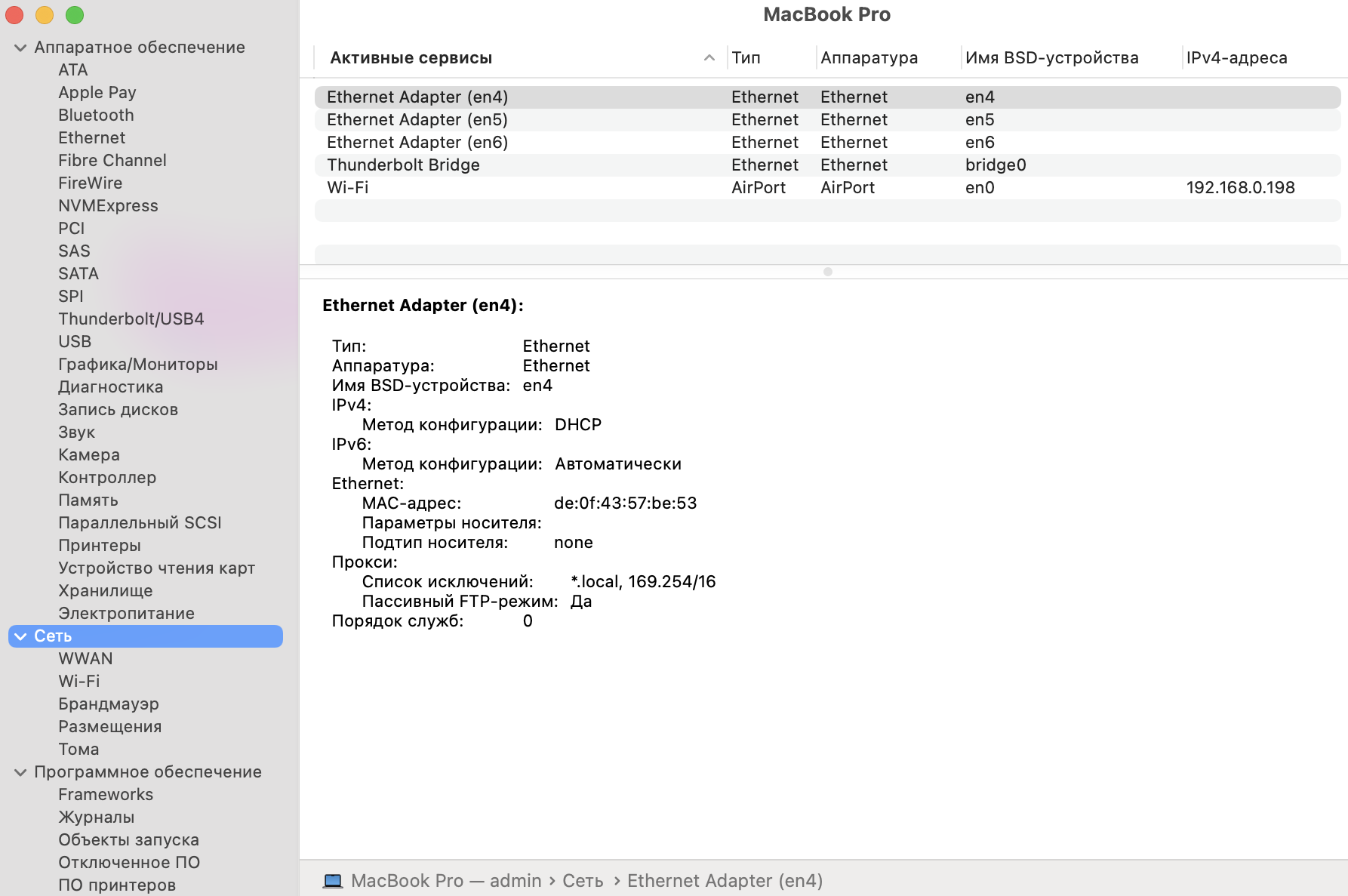
****

Рисунок 1.1.3 - Характеристики процесора

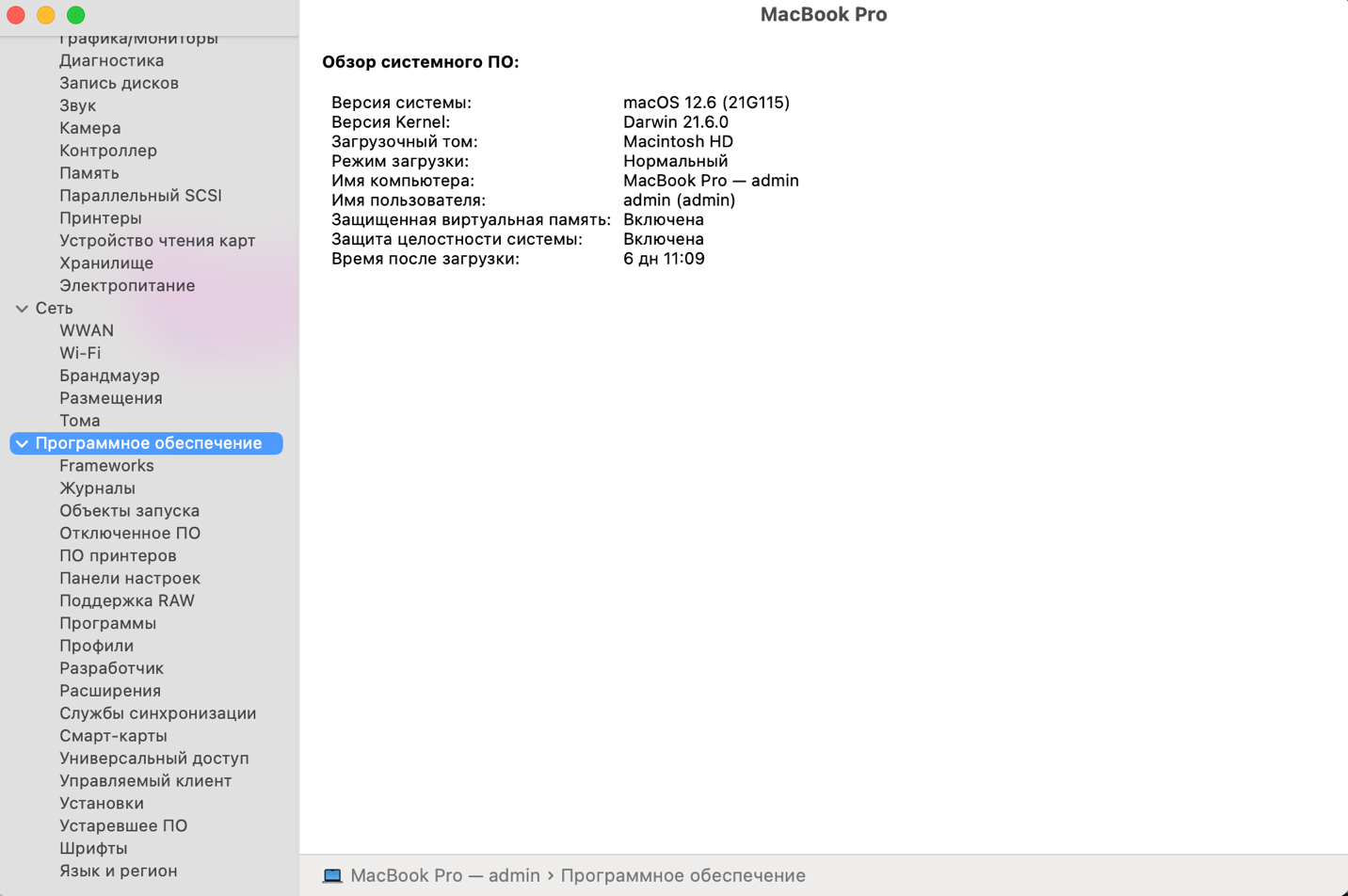
****

Рисунок 1.1.3 - Характеристики процесора

MacCPUID:

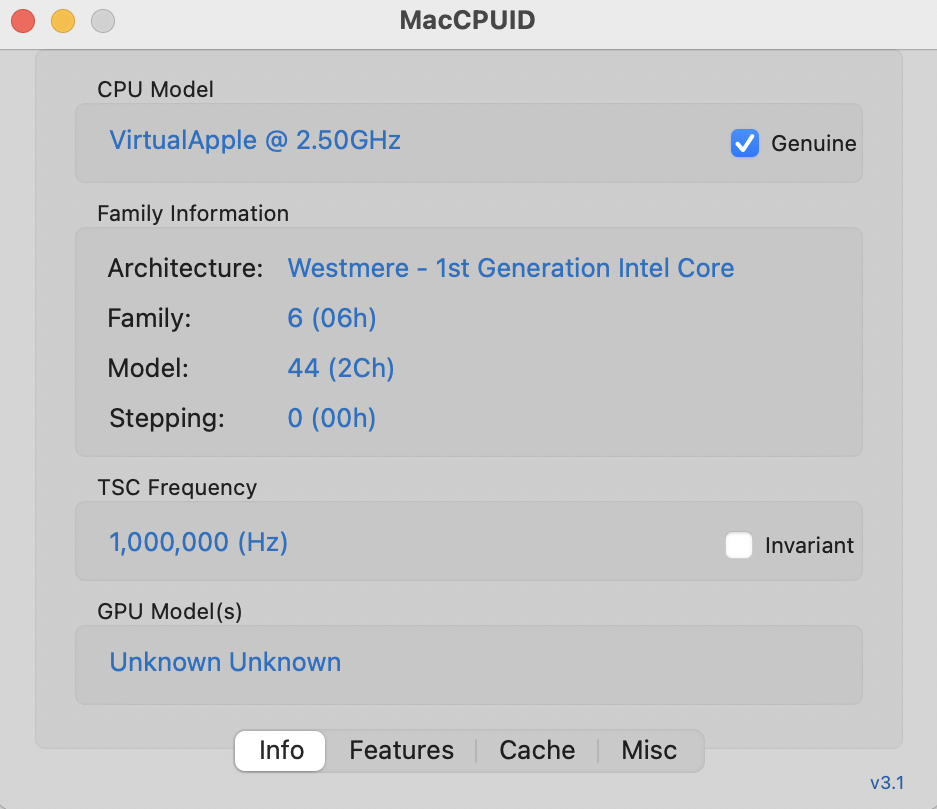
****

Рисунок 1.2.1 - Характеристики процесора

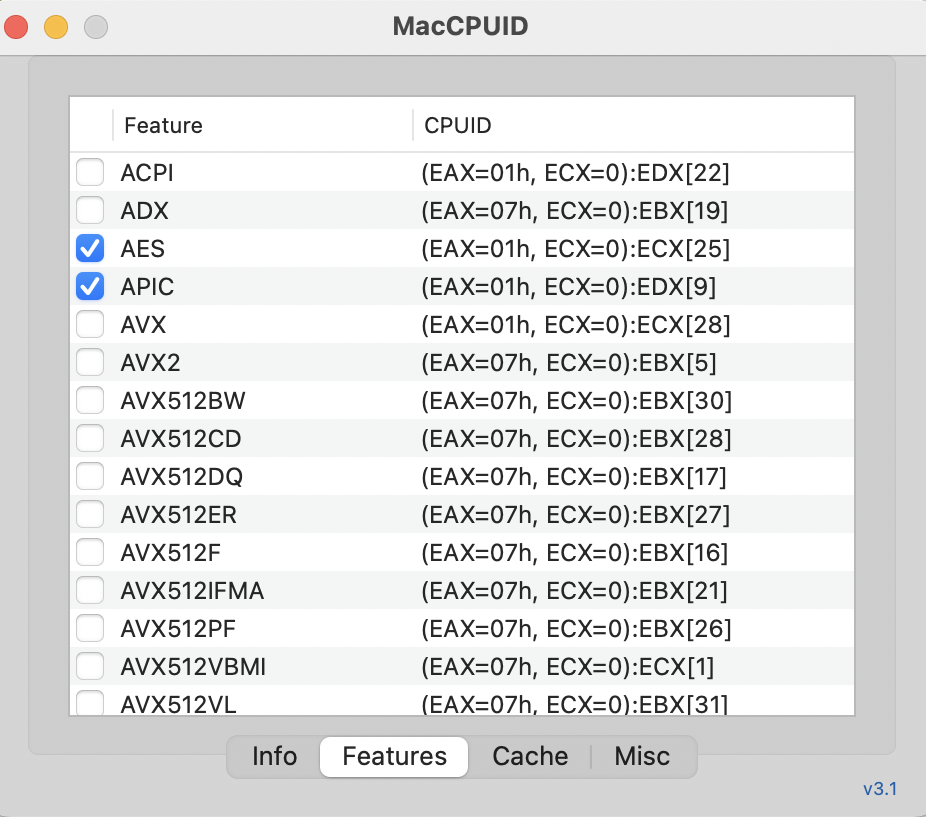
****

Рисунок 1.2.2 - Характеристики процесора

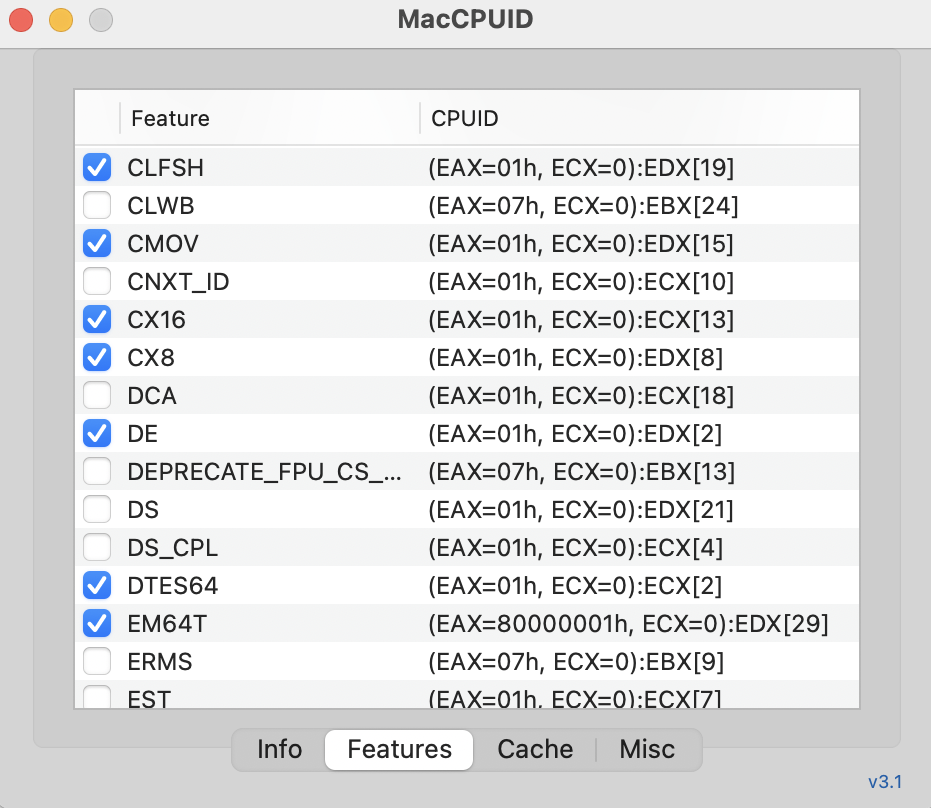
****

Рисунок 1.2.3 - Характеристики процесора

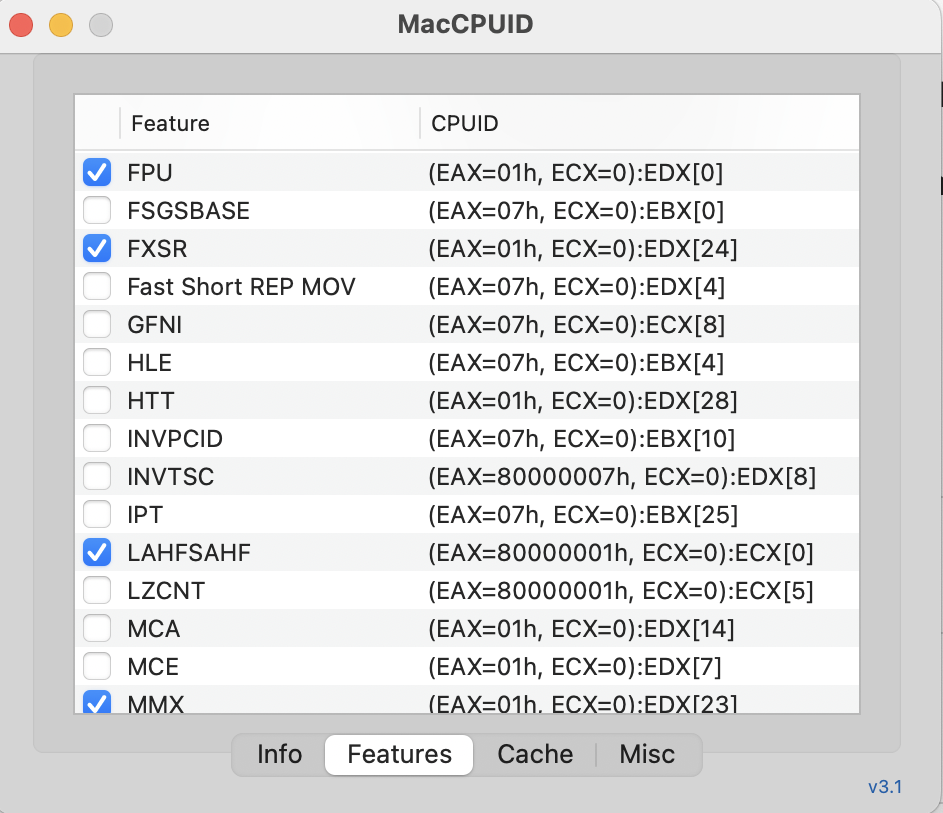
****

Рисунок 1.2.4 - Характеристики процесора

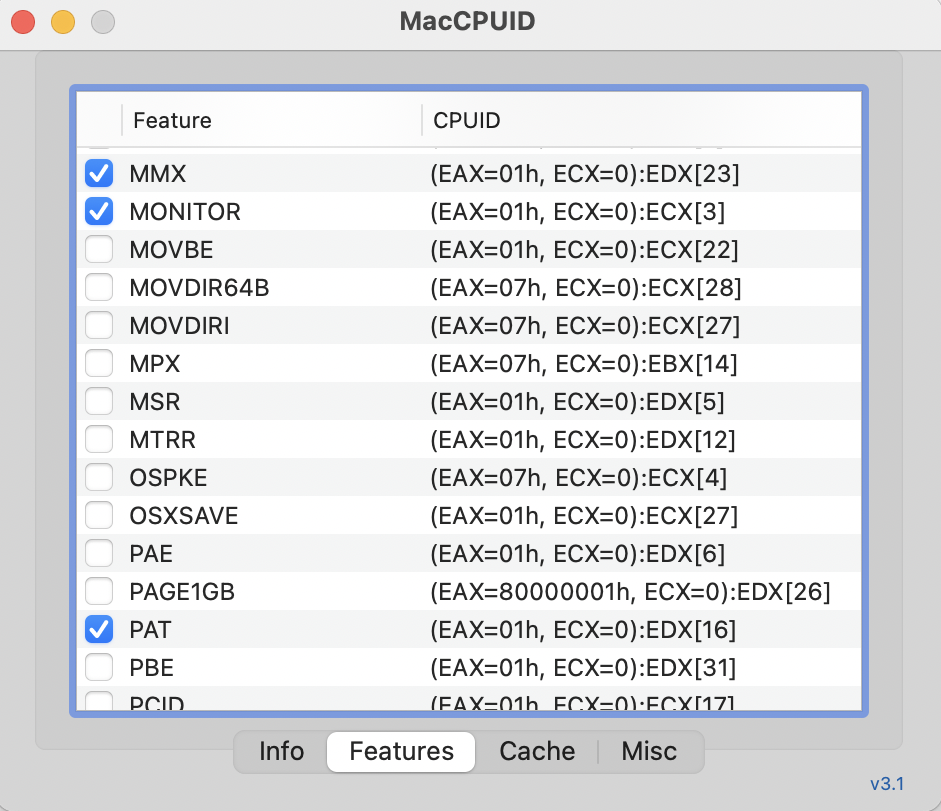
****

Рисунок 1.2.5 - Характеристики процесора

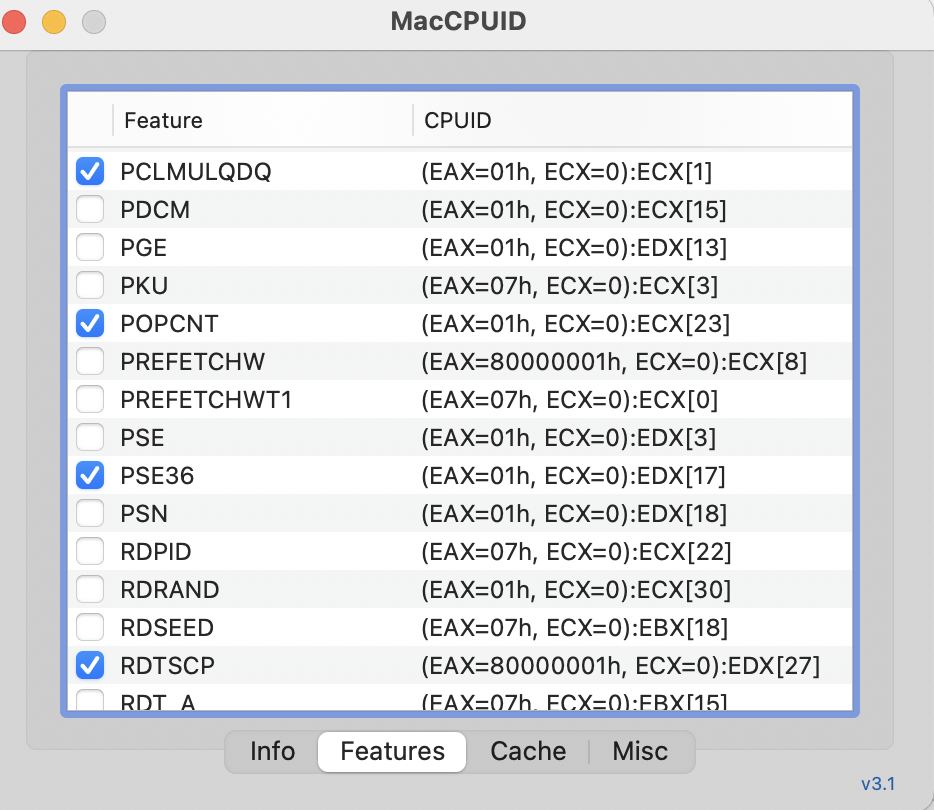
****

Рисунок 1.2.6 - Характеристики процесора

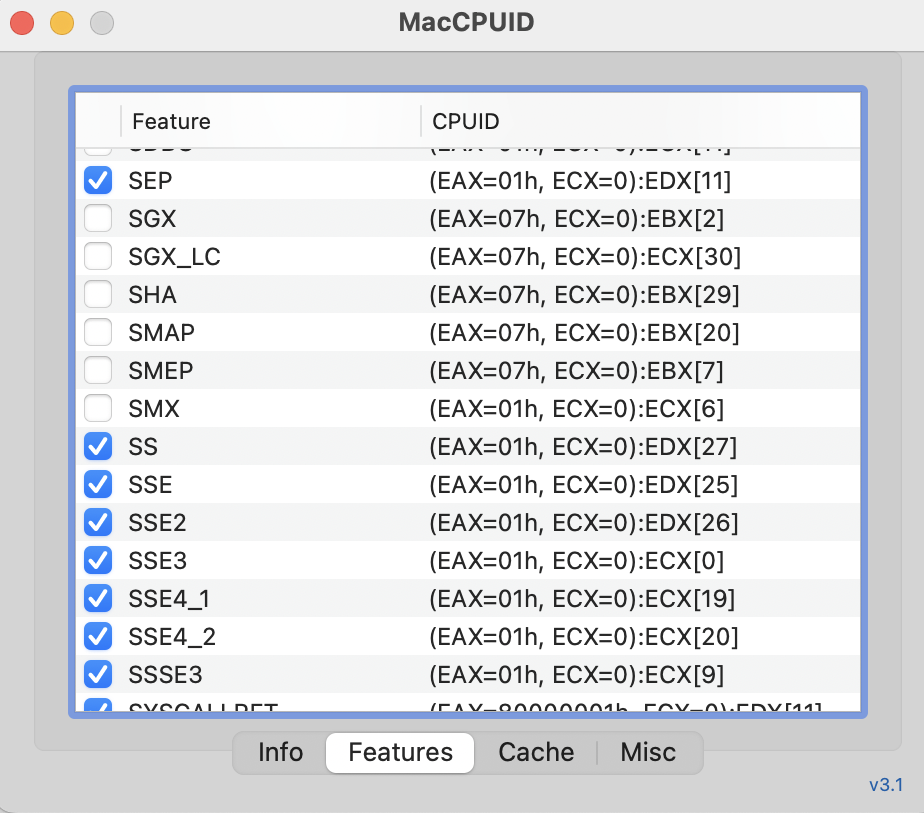
****

Рисунок 1.2.7 - Характеристики процесора

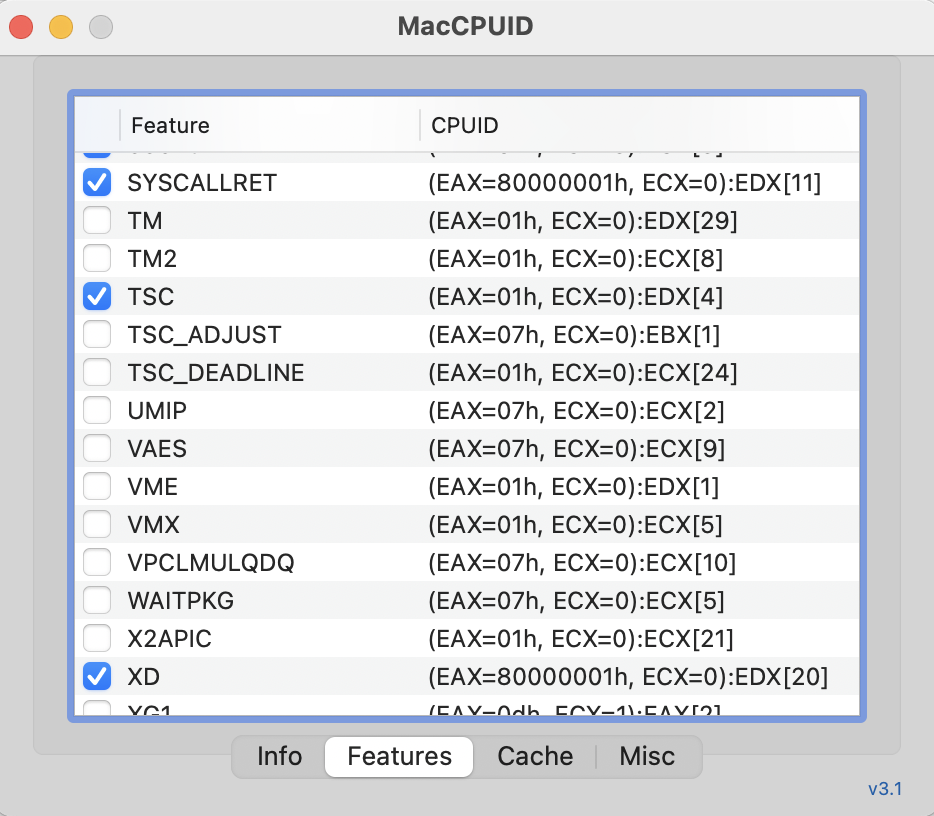


Рисунок 1.2.8 - Характеристики процесора

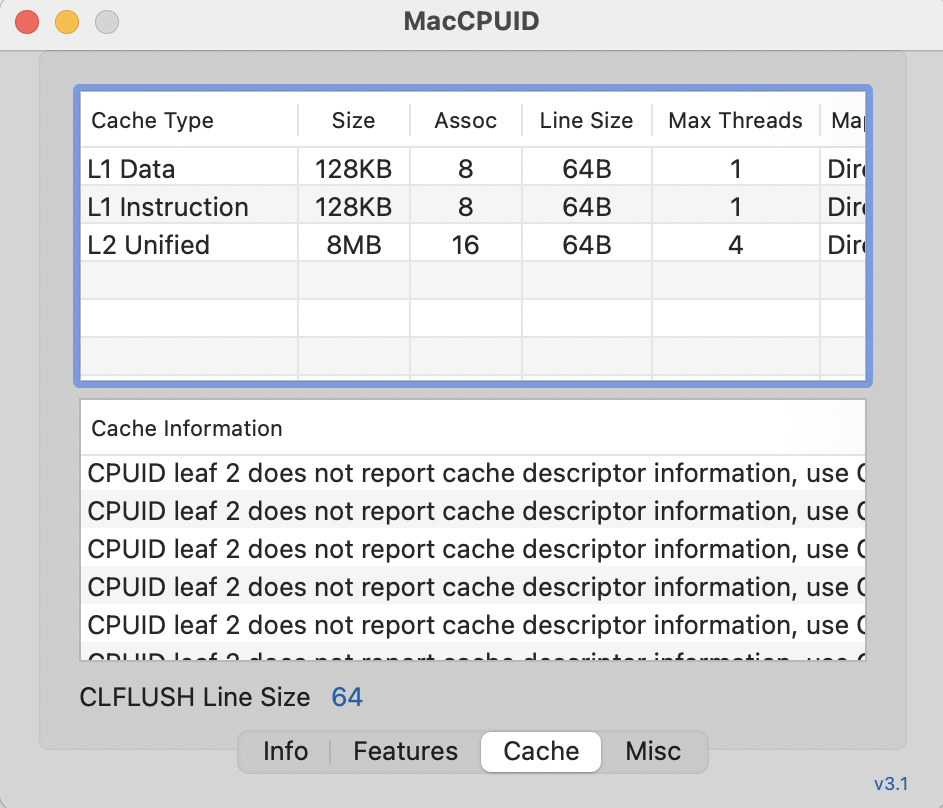
****

Рисунок 1.2.9 - Характеристики процесора

****

Рисунок 1.2.10 - Характеристики процесора

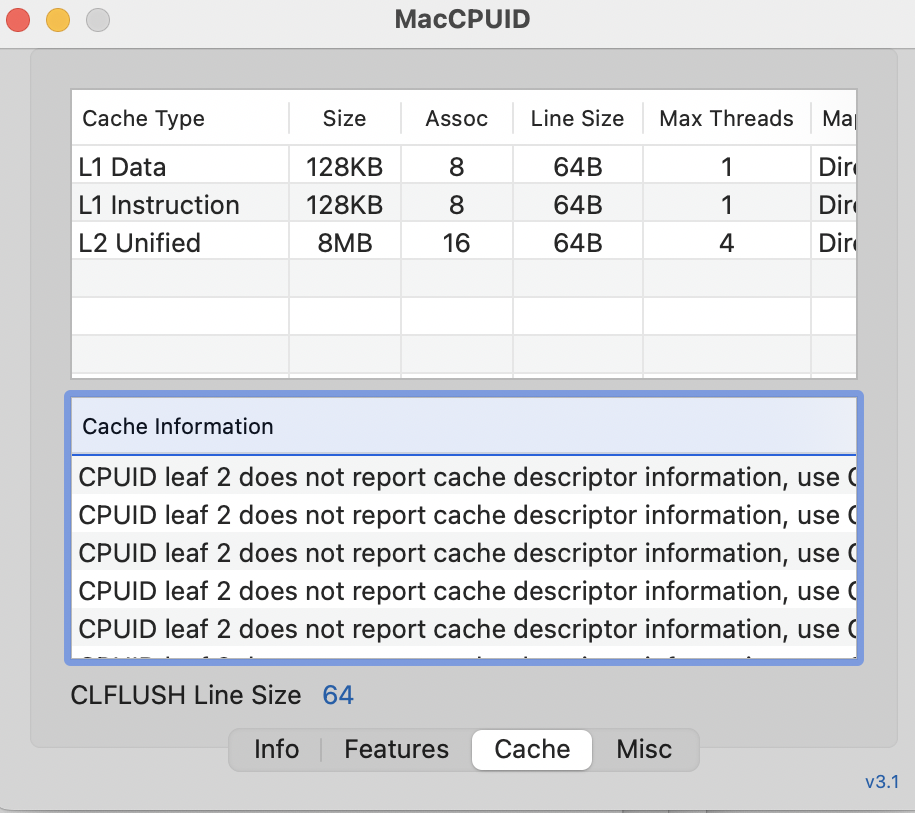
****

Рисунок 1.2.11 - Характеристики процесора

****

Рисунок 1.2.12 - Характеристики процесора

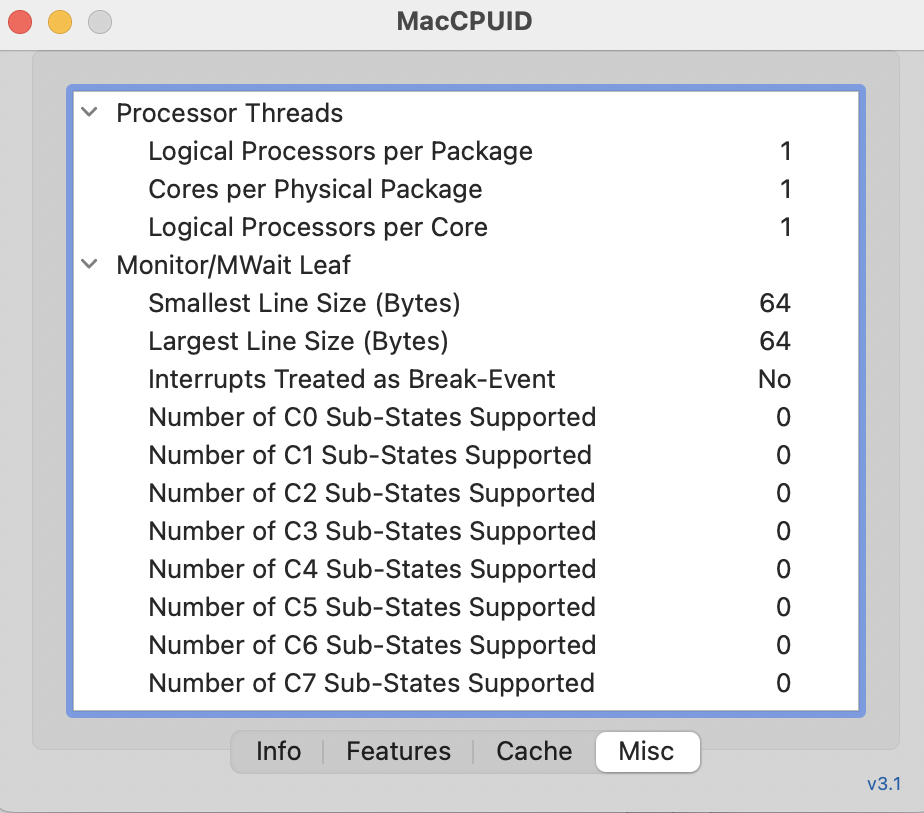
****

Рисунок 1.2.13 - Характеристики процесора

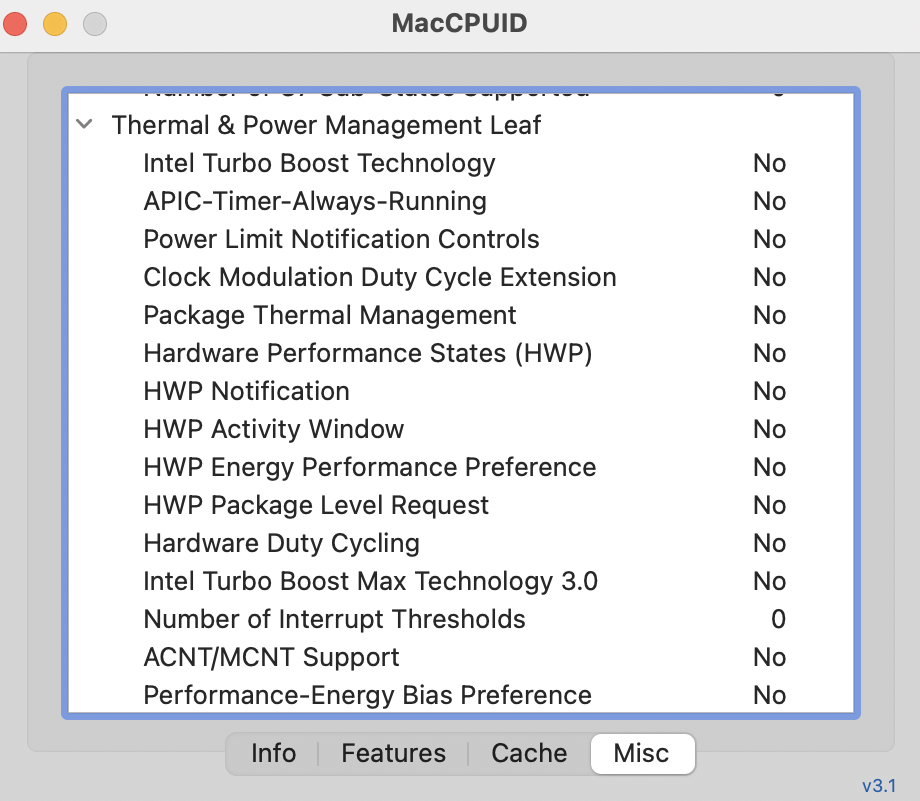
****

Рисунок 1.2.14 - Характеристики процесора

**Cкріни про те, що перемноження відбувається вірно на прикладі невеликих матриць**

Дві матриці можна перемножити між собою тоді й тільки тоді, коли кількість стовпців першої матриці дорівнює кількості рядків другої матриці. При множенні матриці вектор отримуємо вертикальну матрицю в один стовпець, а при множенні вектора на матрицю отримуємо вектор (матрицю в один рядок). Немає сенсу перевіряти працездатність та правильність виконання для кожного окремого типу даних (int, float, double), тому перевірка буде проходити через int оголошення матриць.

На рисунку 2 нижче наведено приклад виконання операцій (виведено на консоль): множення матриці на вектор, множення вектора на матрицю, множення вектора на матрицю (швидшим способом). Надамо, для перевірки, n значення 3 – у нас буде квадратна матриця розміром 3 на 3 та вектор розміром 1 на 3.

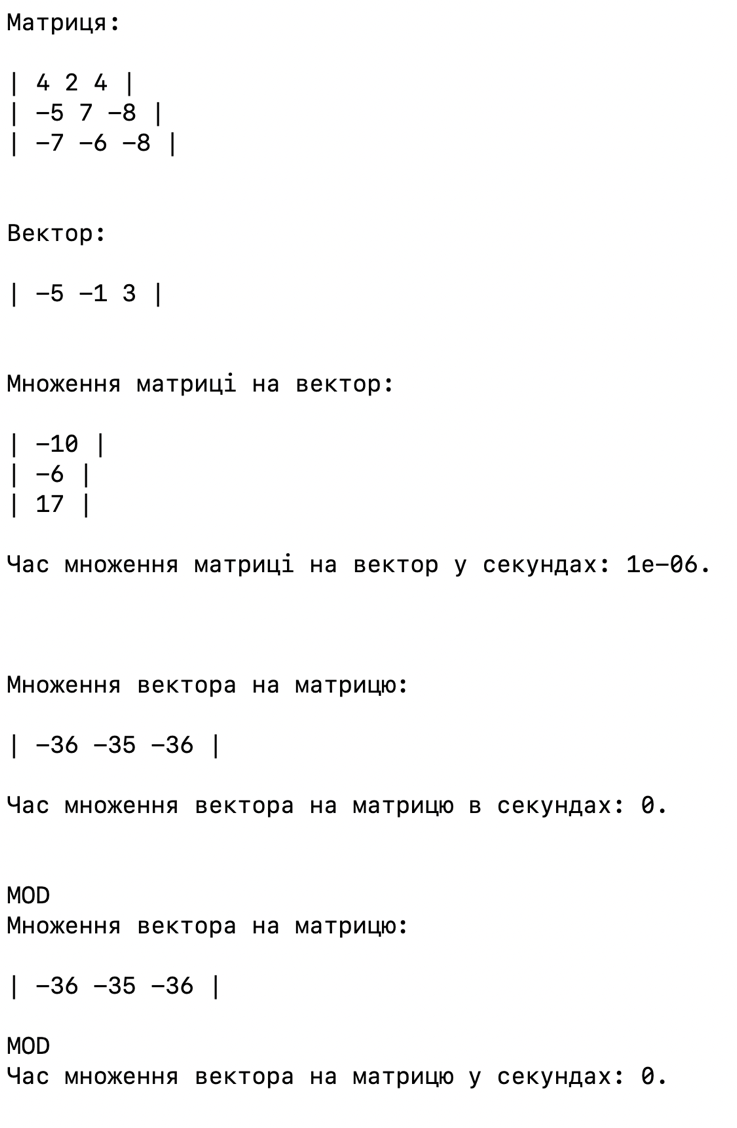


Рисунок 2 Приклад заміру часу для вектора та матриці типу int розміром 3 та 3 на 3 відповідно

**Створити таблиці, куди занести результати засічок часу**

Уже відомо, що розрахунки матриць правильні, тому будемо виводити лише час. Надамо N значення 1000, будемо виконувати операції: множення матриці на вектор, множення вектора на матрицю, множення вектора на матрицю (швидко) – поки N менше або дорівнює 22000, кожного разу в кінці циклу додаємо до N крок - 1000. Виконуємо операції з трьома типами даних: int, float, double - окремо. Усі розрахунки часу визначені в секундах і вказані нижче в таблиці 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Кількість eлементів N** | **Час виконання на операцію множення** | | | | | | | | |
| **M\*V** | | | **V\*M** | | | **V\*M(MOD)** | | |
| **int** | **float** | **double** | **int** | **float** | **double** | **int** | **float** | **double** |
| 1000 | 0,003837 | 0,00617 | 0,006305 | 0,013946 | 0,010457 | 0,014938 | 0,002847 | 0,00241 | 0,002193 |
| 2000 | 0,008976 | 0,014605 | 0,01427 | 0,038697 | 0,039142 | 0,038925 | 0,006557 | 0,006351 | 0,007288 |
| 3000 | 0,01905 | 0,0312 | 0,032696 | 0,089406 | 0,087962 | 0,087783 | 0,015774 | 0,015119 | 0,015215 |
| 4000 | 0,033891 | 0,055807 | 0,057871 | 0,161385 | 0,160383 | 0,224364 | 0,025812 | 0,025386 | 0,026044 |
| 5000 | 0,053894 | 0,088177 | 0,088343 | 0,283018 | 0,272176 | 0,251084 | 0,03954 | 0,040225 | 0,042655 |
| 6000 | 0,075603 | 0,126348 | 0,125725 | 0,390935 | 0,444348 | 0,35482 | 0,05691 | 0,056085 | 0,059829 |
| 7000 | 0,108728 | 0,168534 | 0,173607 | 0,518237 | 0,617258 | 0,514266 | 0,078475 | 0,075774 | 0,083592 |
| 8000 | 0,133256 | 0,220721 | 0,222446 | 0,710356 | 1,07334 | 0,709486 | 0,100181 | 0,09989 | 0,106761 |
| 9000 | 0,169694 | 0,28107 | 0,283912 | 0,894732 | 1,42439 | 0,892628 | 0,125861 | 0,126678 | 0,13589 |
| 10000 | 0,205609 | 0,346484 | 0,345153 | 1,13765 | 1,80311 | 1,10855 | 0,155692 | 0,154501 | 0,165543 |
| 11000 | 0,251293 | 0,415937 | 0,419125 | 1,46314 | 2,2657 | 1,44069 | 0,186524 | 0,185897 | 0,206033 |
| 12000 | 0,295982 | 0,493814 | 0,495079 | 1,77433 | 2,32259 | 1,797 | 0,222938 | 0,226885 | 0,237644 |
| 13000 | 0,353344 | 0,601393 | 0,583834 | 2,04697 | 2,65673 | 2,05524 | 0,261095 | 0,259915 | 0,278427 |
| 14000 | 0,406838 | 0,676064 | 0,673402 | 2,48166 | 3,27135 | 2,57716 | 0,302048 | 0,305046 | 0,322561 |
| 15000 | 0,460884 | 0,771333 | 0,779847 | 2,66586 | 4,18373 | 2,66419 | 0,346256 | 0,346602 | 0,370881 |
| 16000 | 0,524376 | 0,877601 | 0,879357 | 3,31517 | 5,24573 | 3,00133 | 0,395482 | 0,394508 | 0,422571 |
| 17000 | 0,598597 | 0,997056 | 0,995494 | 3,79438 | 6,1538 | 3,62831 | 0,446962 | 0,446856 | 0,474922 |
| 18000 | 0,663118 | 1,1116 | 1,11567 | 4,54364 | 6,83354 | 4,25764 | 0,502622 | 0,502528 | 0,53232 |
| 19000 | 0,73854 | 1,23705 | 1,24697 | 5,1782 | 7,27366 | 4,81837 | 0,560807 | 0,56136 | 0,594614 |
| 20000 | 0,818146 | 1,37577 | 1,37386 | 6,08845 | 7,72103 | 5,15996 | 0,624196 | 0,623692 | 0,656162 |
| 21000 | 0,908708 | 1,51611 | 1,5214 | 6,97051 | 7,78562 | 5,84658 | 0,687847 | 0,689461 | 0,724639 |
| 22000 | 0,986219 | 1,65864 | 1,6806 | 8,139 | 8,70276 | 6,46457 | 0,752683 | 0,753389 | 0,793926 |

Таблиця 1 – Заміри часу виконання операцій над матрицею та векторами

**Створити графіки, на яких відображати дані з таблиць**

Графік 1 - Вимірювання часу множення матриці на вектор трьома типами даних

Графік 2 - Вимірювання часу множення вектора на матрицю трьома типами даних

Графік 3 - Вимірювання часу множення вектора на матрицю (MOD) трьома типами даних

Графік 4 - Вимірювання часу множення матриці на вектор, вектора на матрицю та вектора на матрицю (MOD) використавши тільки тип даних int

Графік 5 - Вимірювання часу множення матриці на вектор, вектора на матрицю та вектора на матрицю (MOD) використавши тільки тип даних float

Графік 6 - Вимірювання часу множення матриці на вектор, вектора на матрицю та вектора на матрицю (MOD) використавши тільки тип даних double

**Код для перевірки програми на невеликих числах, при N=3**

#include <iostream>

#include <ctime>

#include <cmath>

using namespace std;

int main()

{

srand(time(NULL));

double t1, t2, t3, t4, t5, t6, t7, t8, t9, t10;

int n = 3;

int \*\*M = new int\*[n];

for (int i=0; i<n; i++)

M[i] = new int[n];

int \*V = new int[n];

int \*MV = new int[n];

int \*VM = new int[n];

int \*VM2 = new int[n];

for (int i=0; i<n; i++)

{

for (int j=0; j<n; j++)

{

M[i][j] = rand()%19-9;

}

}

cout << "\nMатриця:\n\n";

for (int i=0; i<n; i++)

{

cout << "| ";

for (int j=0; j<n; j++)

{

cout << M[i][j];

if (j < n - 1) cout << " ";

else cout << " |" << endl;

}

}

for (int i=0; i<n; i++)

{

V[i] = rand()%19-9;

}

cout << "\n\nВектор:\n\n| ";

for (int i=0; i<n; i++)

{

cout << V[i] << " ";

}

cout << "|\n\n";

for (int i=0; i<n; i++)

{

MV[i] = 0;

}

t1= clock();

for (int i=0; i<n; i++)

{

for (int j=0; j < n; j++)

{

MV[i] += M[i][j] \* V[j];

}

}

t2 = clock();

cout << "\nМноження матриці на вектор:\n\n";

for (int i=0; i<n; i++)

{

cout << "| " << MV[i] << " |\n";

}

cout << "\nЧас множення матриці на вектор у секундах: " << (t2-t1)/CLOCKS\_PER\_SEC << ".\n\n";

for (int i=0; i < n; i++)

{

VM[i]=0;

}

t3 = clock();

for (int i=0; i < n; i++)

{

for (int j=0; j < n; j++)

{

VM[i] += M[j][i] \* V[j];

}

}

t4 = clock();

cout << "\n\nМноження вектора на матрицю:\n\n| ";

for (int i=0; i < n; i++)

{

cout << VM[i] << " ";

}

cout << "|" << endl;

cout << "\nЧас множення вектора на матрицю в секундах: " << (t4-t3)/CLOCKS\_PER\_SEC << ".\n\n";

for (int i=0; i < n; i++)

{

VM2[i]=0;

}

t5 = clock();

for (int j=0; j < n; j++)

{

for (int i=0; i < n; i++)

{

VM2[i] += M[j][i] \* V[j];

}

}

t6 = clock();

cout << "\nMOD\nМноження вектора на матрицю:\n\n| ";

for (int i=0; i < n; i++)

{

cout << VM2[i] << " ";

}

cout << "|" << endl;

cout << "\nMOD\nЧас множення вектора на матрицю у секундах: " << (t6-t5)/CLOCKS\_PER\_SEC << ".\n\n";

return 0;

}

**Код програми, при N=22000**

#include <iostream>

#include <ctime>

#include <cmath>

using namespace std;

int main()

{

srand(time(NULL));

double t1, t2, t3, t4, t5, t6, t7, t8;

int n = 22000;

float \*\*M = new float\*[n];

for (int i=0; i<n; i++)

M[i] = new float[n];

float \*V = new float[n];

float \*MV = new float[n];

float \*VM = new float[n];

float \*VM2 = new float[n];

for (long long int o=1000; o<=n; o+=1000)

{

for (int i=0; i<o; i++)

{

for (int j=0; j<o; j++)

{

M[i][j] = rand()%19-9;

}

}

for (int i=0; i<o; i++)

{

V[i] = rand()%19-9;

}

for (int i=0; i<o; i++)

{

MV[i] = 0;

}

t1= clock();

for (int i=0; i<o; i++)

{

for (int j=0; j<o; j++)

{

MV[i] += M[i][j] \* V[j];

}

}

t2 = clock();

cout << "\nЧас множення матриці на вектор у секундах: " << (t2-t1)/CLOCKS\_PER\_SEC << ". N = " << o << ".\n\n";

for (int i=0; i<o; i++)

{

VM[i]=0;

}

t3 = clock();

for (int i=0; i<o; i++)

{

for (int j=0; j<o; j++)

{

VM[i] += M[j][i] \* V[j];

}

}

t4 = clock();

cout << "\nЧас множення вектора на матрицю в секундах: " << (t4-t3)/CLOCKS\_PER\_SEC << ". N = " << o << ".\n\n";

for (int i=0; i<o; i++)

{

VM2[i]=0;

}

t5 = clock();

for (int j=0; j<o; j++)

{

for (int i=0; i<o; i++)

{

VM2[i] += M[j][i] \* V[j];

}

}

t6 = clock();

cout << "\nMOD\nЧас множення вектора на матрицю у секундах: " << (t6-t5)/CLOCKS\_PER\_SEC << ". N = " << o << ".\n\n";

}

return 0;

}

**Висновки**

Порівнявши час множення різних типів даних і різних методів множення, можна зробити такі висновки.

По-перше, найшвидшим типом даних для множення матриць є int. Різниця між float і double майже непомітна на ранніх етапах, але на більшій кількості елементів можна побачити, що float швидше ніж double.

По-друге, звичайне множення матриці на вектор майже в 10 разів швидше ніж множення вектора на матрицю. Змінивши алгоритм множення вектора на матрицю, як зазначено в завданні, можна переконатися, що множення вектора на матрицю виконується так само швидко, як множення матриці на вектор. Час множення вектора на матрицю змінився майже в 10 разів і став несуттєво відрізнятися від часу множення матриці на вектор.