АРХИТЕКТУРА РАЧУНАРА ПРОЈЕКТНИ ЗАДАТАК 2.2 – Оптимизација алгоритма

ТЕМА: Линеарна регресија

Аутор: Новица Тепић

Верзија: 1.0

Датум: 25.1.2023.

Садржај

[Увод 3](#_Toc127030480)

[Поступак и услови тестирања 3](#_Toc127030481)

[Поређење времена извршавања 4](#_Toc127030482)

[Објашњење логике свих програма 8](#_Toc127030483)

[Просјечне вриједности и варијансе времена извршавања програма 9](#_Toc127030484)

[Графички приказ резултата 11](#_Toc127030485)

[Закључак 14](#_Toc127030486)

# Увод

За оптимизацију алгоритма кориштен је AVX инструкцијски скуп, као и паралелизација на вишејезгреном процесору употребом OpenMP. Четири програма су реализована у C програмском језику (програм без оптимизација, AVX, OpenMP као и „обједињени“ програм који садржи AVX инструкције и OpenMP.

# Поступак и услови тестирања

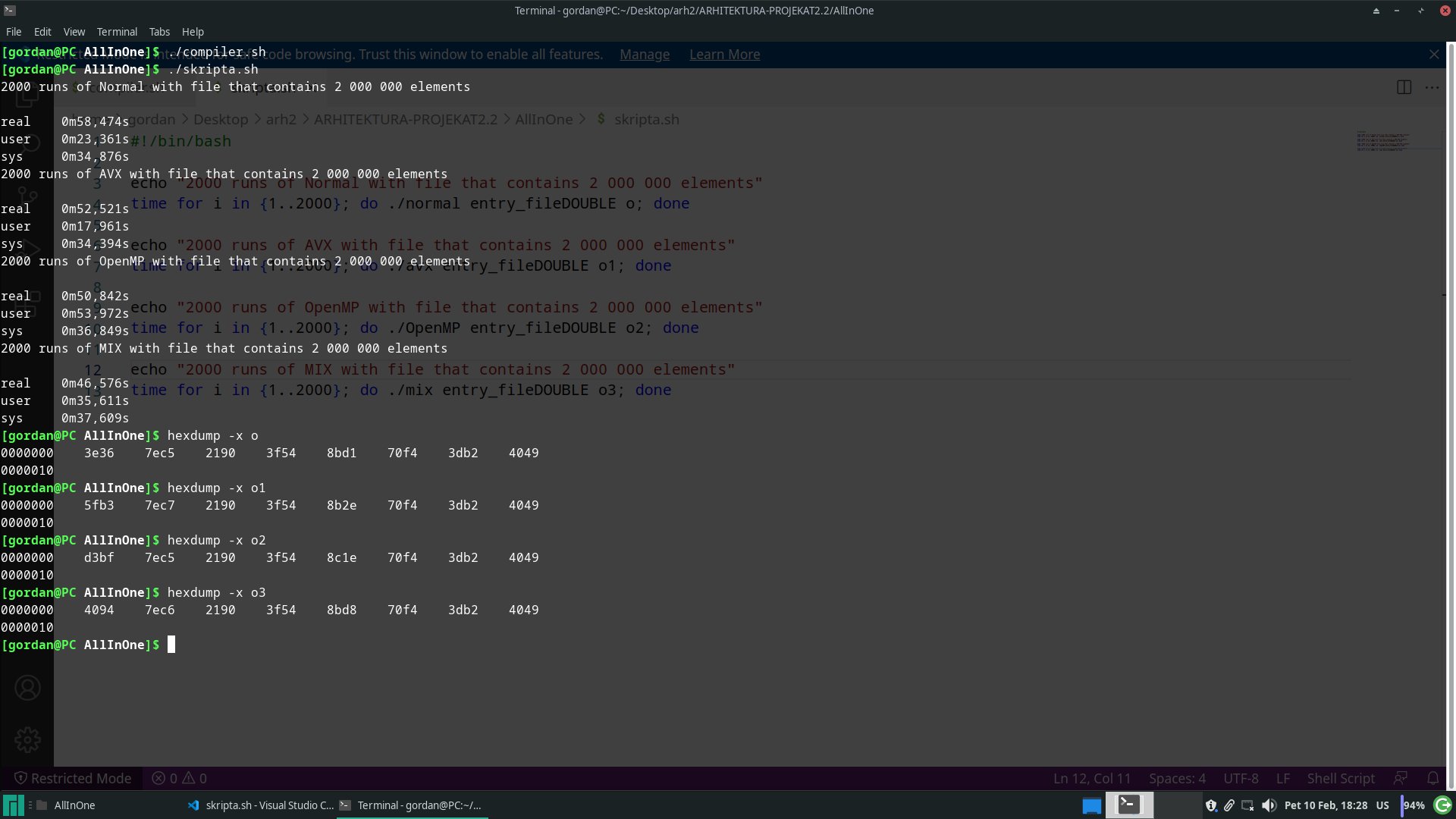
Кориштене су двије скрипте:

-прва скрипта (compiler.sh) је кориштена за компајлирање и сви програми су компајлери са –О0 оптимизацијом, с тим да је за AVX кориштен –mavx2 flag, за OpenMP је кориштен –fopenmp flag, а за MIX варијанту су кориштени и –mavx2 и –fopenmp flag.

-друга скрипта (skripta.sh) је кориштена за мјерење времена сва четири програма те њен садржај зависи од величине елемената (што је више елемената, мањи je број извршавања да се не би предуго чекало на крај извршавања програма, а да се јасно виде резултати).

Процесор који је кориштен је Intel i7-4510U (два физичка језгра и четири логичка језгра, а основна фреквенција рада је 2.00Ghz, док при већем оптерећењу иде и до 3.10GHz), а оперативни систем је Manjaro Linux (Arch дистрибуција).

# Поређење времена извршавања



1-2000 извршавања, фајл који има 2 000 000 елемената

Вриједност за o фајл (без оптимизација):

A = 0.001228705509

B = 50.482008094222367494177

Вриједност за o1 фајл (оптимизације помоћу AVX):

A = 0.0012287055097796983

B = 50.4820080942212

Вриједност за o2 фајл (OpenMP):

A = 0.0012287055097577185

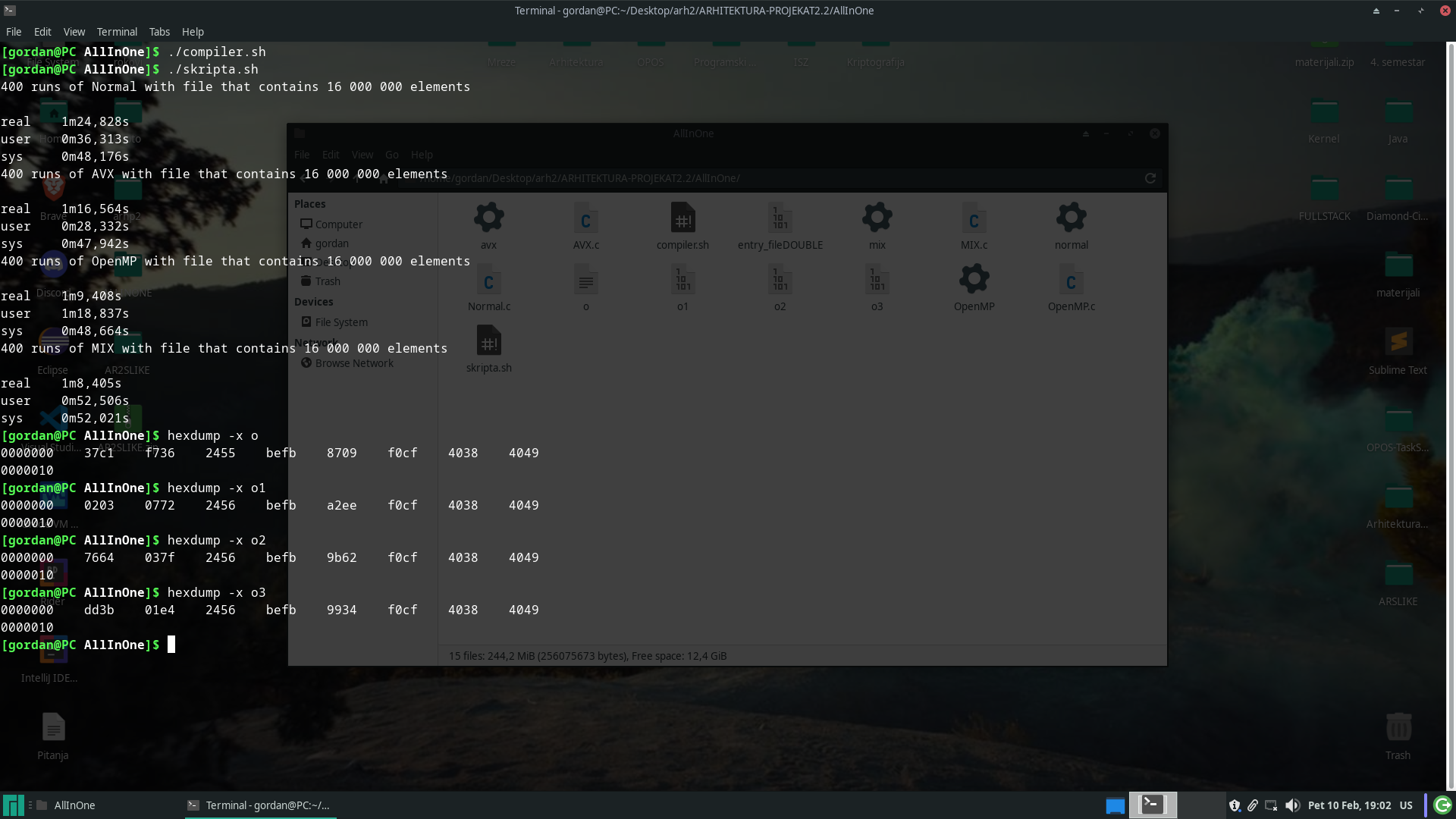
B = 50.482008094222905

Вриједност за o3 фајл (OpenMP+AVX):

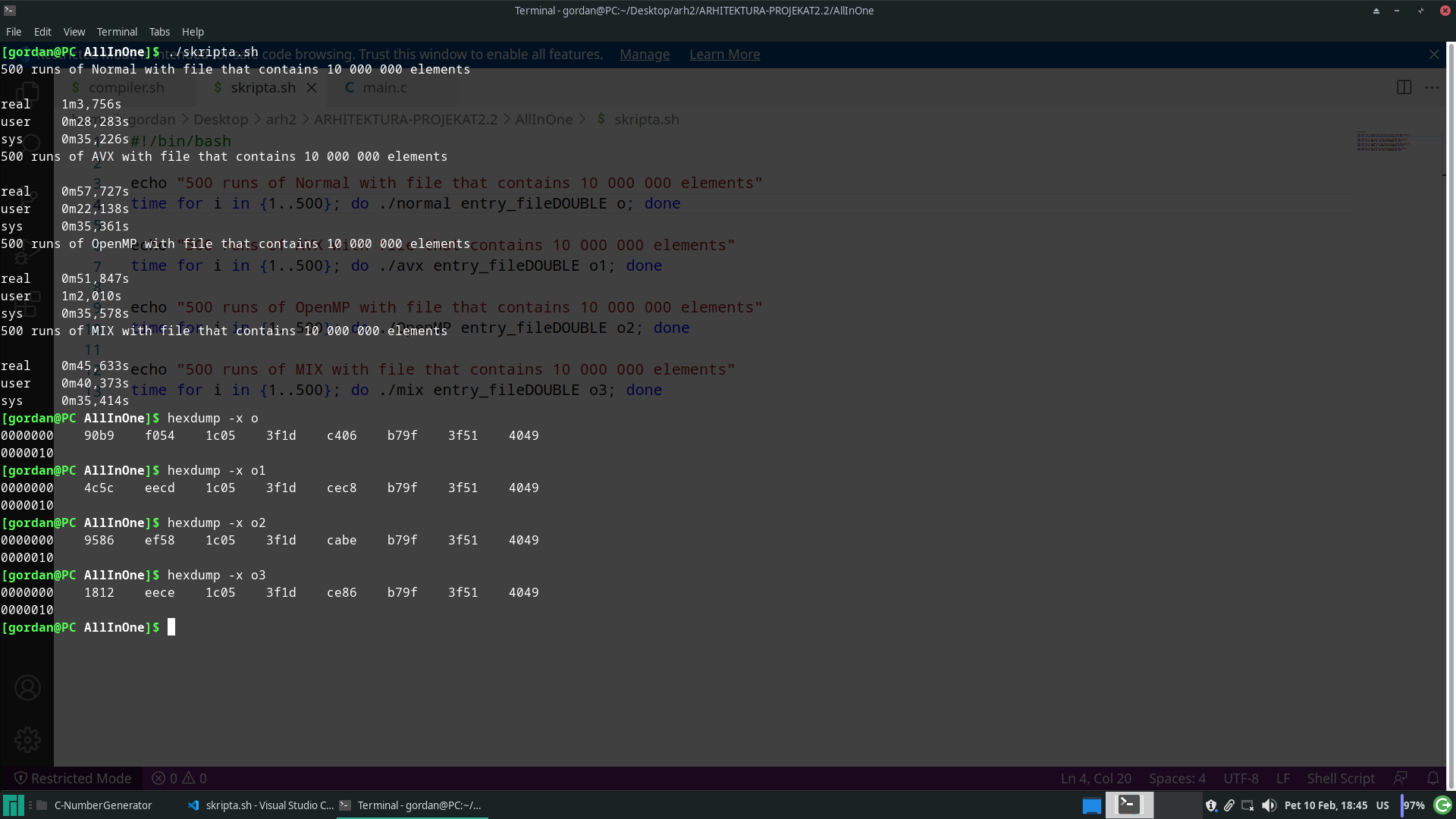
A = 0.00122870550976376

B = 50.48200809422241

Аналогно вриједи и за остале резултате (може се видјети и на основу хексадецималних бројева).

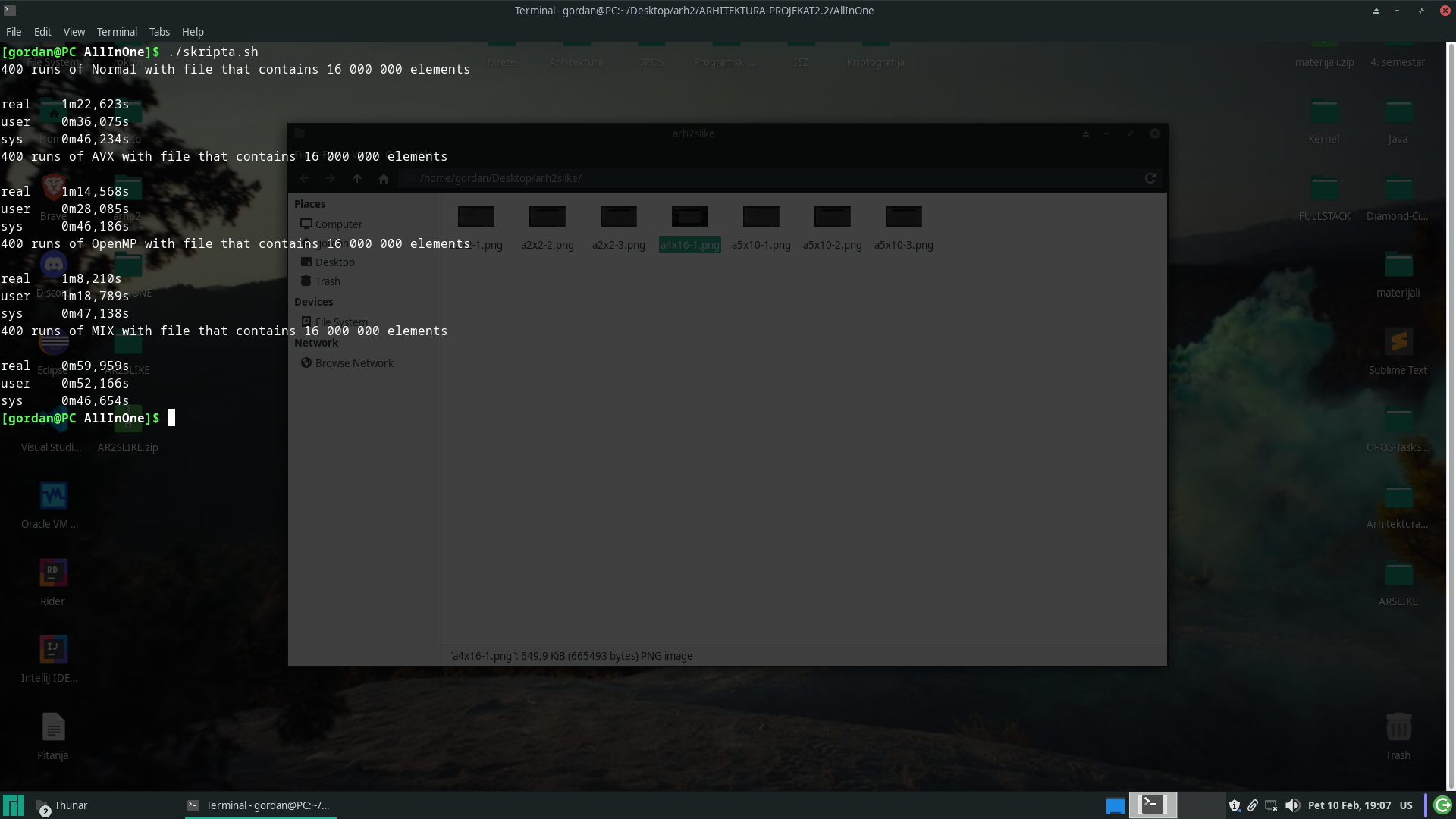


2-400 извршавања, фајл који има 16 милиона елемената



3-500 извршавања, фајл који има 10 000 000 елемената

Треба да напоменем да на другом приложеном screenshot-u вријеме није било одговарајуће за MIX варијанту, па сам извршио програм још пар пута и добио боља времена, што ће бити приложено касније у табели али и помоћу одговарајућег screenshot-a:



Приказани су и резултати извршавања при чему o фајл има резултате основног програма без оптимизација, o1 резултате гдје су кориштене AVX инструкције, o2 резултате са OpenMP варијантом, а o3 резултате са комбинованим приступом, тј. OpenMP и AVX варијантом.

Као што се види, постоје мала одступања у резултатима али су она незнатна (односе се на мјеста са десне стране зареза, као и у асемблерском програму на првом пројектном задатку).

Очигледно је да су убрзања постигнута на основу измјерених времена.

# Објашњење логике свих програма

Што се тиче програма без оптимизација, мислим да је логика и више него јасна, израчунају се суме у петљи и користе се формуле за рачунање параметара a и b, те се ти параметри смјештају у фајл.

Што се тиче AVX инструкцијског скупа, омогућио сам учитавање 4 вриједности одједном (из оба низа), те су и неопходна множења и сабирање рађена помоћу одговарајућих AVX инструкција, чиме се постигло убрзање.

OpenMP је кориштен на једноставан начин, јер ми је reduction омогућио да искористим више threadova на поуздан начин јер су операције thread safe и нисам имао страха да ће доћи до нарушавања резултата.

Комбинација OpenMP и AVX инструкцијског скупа је опет користила reduction, али сада је постојао проблем са threadovima, те сам због тога онемогућио да један thread узме податке из низа (низова) другог threada тако што је сваки thread имао свој опсег који је покривао у петљи, те су остварена значајна убрзања у односу на програм без оптимизација, али и остала два програма.

# Просјечне вриједности и варијансе времена извршавања програма

Треба рећи да у првој табели имамо 2000 покретања фајла који има 4 милиона елемената:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Редни број покретања | Без оптимизација | AVX | OpenMP | MIX варијанта |
| 1. | 58,474s | 52,521s | 50,842s | 46,576s |
| 2. | 57,564s | 52,566s | 51,008s | 46,551s |
| 3. | 57,503s | 52,338s | 50,783s | 46,498s |
| ПРОСЈЕК | 57.847s | 52.475s | 50.877s | 46.541s |
| ВАРИЈАНСА | **0.295777** | **0.014583** | **0.013610333** | **0.0015863333** |

Table

У другој табели посматрамо 500 покретања фајла који има 20 милиона елемената:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Редни број покретања | Без оптимизација | AVX | OpenMP | MIX варијанта |
| 1. | 1m 3.756s | 57,727s | 51,847s | 45,633s |
| 2. | 1m 3,485s | 57,534s | 51,527s | 45,795s |
| 3. | 1m 4,127s | 57,478s | 51,584s | 45,882s |
| ПРОСЈЕК | 63.789s | 57.579s | 51.652s | 45.77s |
| ВАРИЈАНСА | **0.10387433** | **0.017064333** | **0.029136333** | **0.015969** |

Table

У трећој табели посматрамо 400 покретања фајла који има 32 милиона елемената:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Редни број покретања | Без оптимизација | AVX | OpenMP | MIX варијанта |
| 1. | 1m 24,828s | 1m 16,564s | 1m 9,408s | 1m 8,405s |
| 2. | 1m 22,623s | 1m 14,568s | 1m 8,210s | 59,959s |
| 3. | 1m 23,080s | 1m 15,329s | 1m 8,088s | 1m 0,665s |
| ПРОСЈЕК | 83.510s | 75.487s | 68.568s | 63.009s |
| ВАРИЈАНСА | **1.3543963** | **1.014727** | **0.53208133** | **21.956825** |

Table

# Графички приказ резултата

Вриједности на графику на y-оси су у секундама.

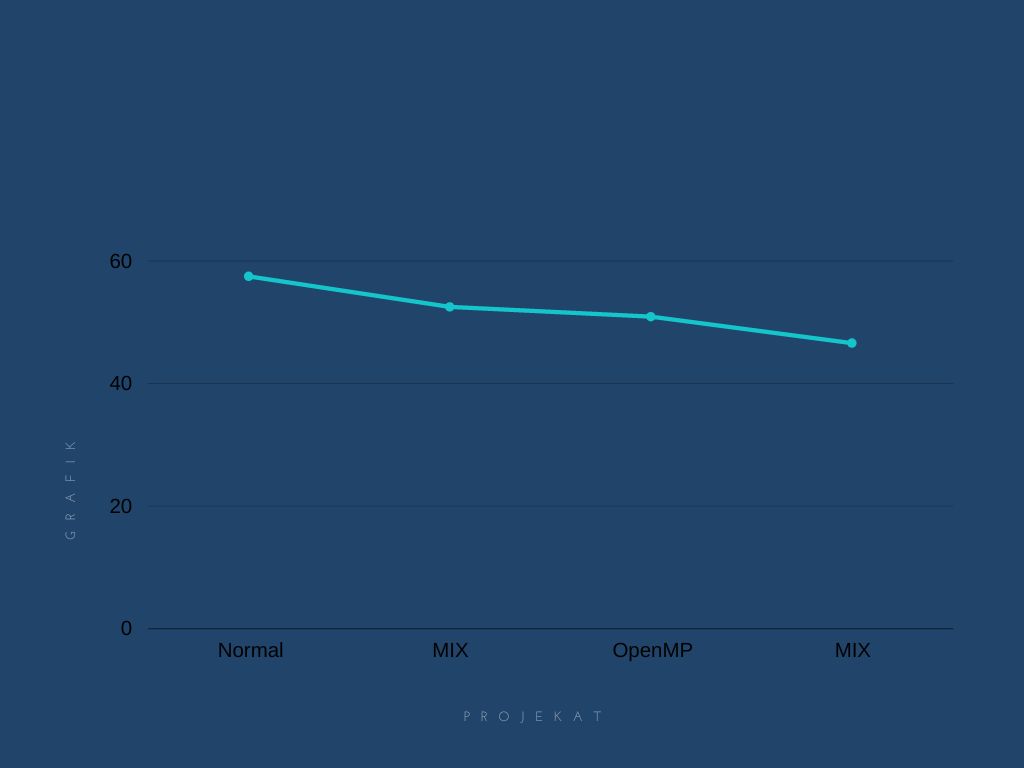


График - Просјечне вриједнсоти из table 1

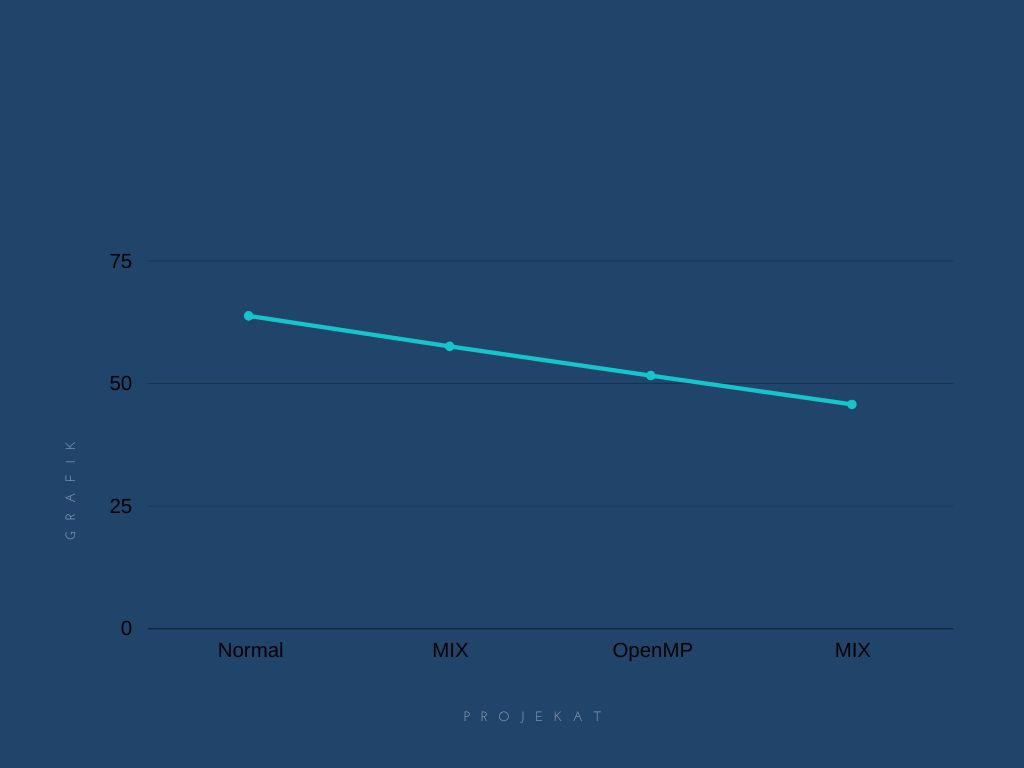


График - Просјечне вриједности из table 2

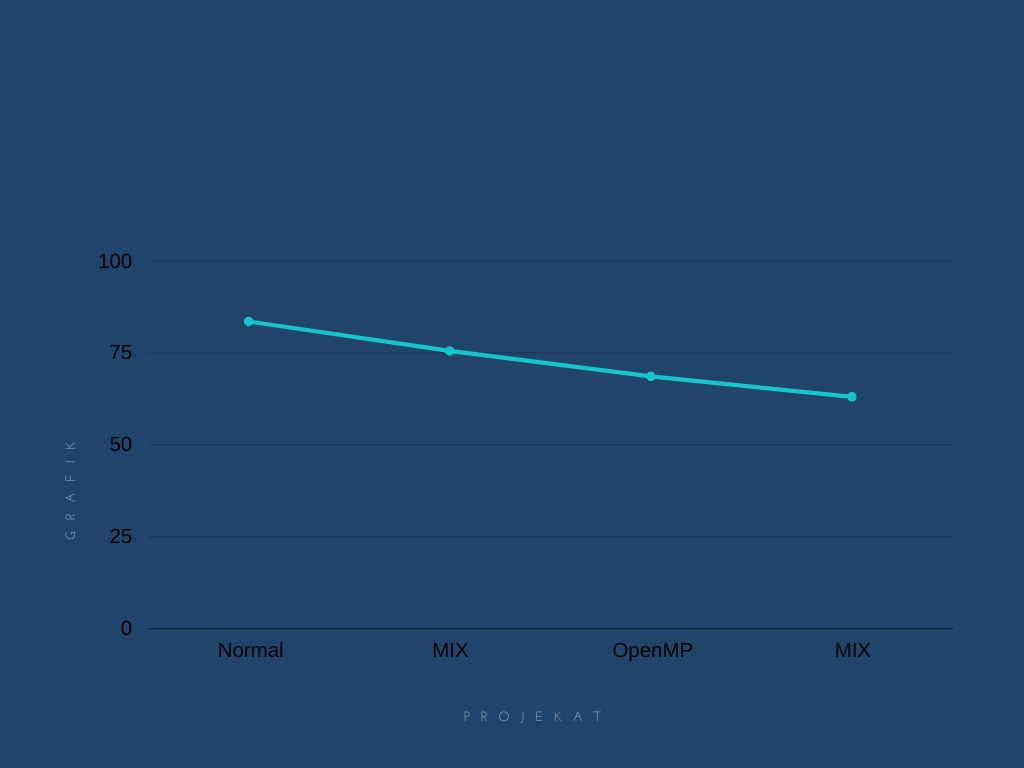


График - Просјечне вриједности из table 3

# Закључак

С обзиром да се ради о линеарној регресији, која нема превише комплексних операција осим рачунања сума, резултати су и више него задовољавајући.

Видимо и конзистентност брзина извршавања што показују табеле, али и резултати су конзистентни уз минимална одступања.

При извршавању програма са O3 оптимизацијом сам примијетио да су перформансе обрнуте, односно да програм без оптимизација има најбрже вријеме извршавања (бар на лаптопу који сам ја користио и на примјерима које сам тестирао, не мора значити да важи за сваки примјер).

Што је већи број елемената, више се показује и снага оптимизације што је логичан закључак.

При неким тестирањима (нпр. са 1 000 000 елеманата), OpenMP ми је показивао јако лоше резултате за шта нисам могао наћи одговор, али на тестирањима са 2, 4 и 16 милиона елемената (у једном низу) OpenMP се показао и више него добар.