АРХИТЕКТУРА РАЧУНАРА ПРОЈЕКТНИ ЗАДАТАК 1.2 – Асемблерски програм за обраду података

ТЕМА: Линеарна регресија

Аутор: Новица Тепић

Верзија: 1.0

Датум: 22.12.2022.

Садржај

Увод

Конкретан проблем рачунања параметара линеарне регресије ријешен је рачунањем параметара за линеарну регресију. С обзиром да је у задатку тражено да се уради верзија са стандардним инструкцијским скупом и са одређеним инструкцијским скупом (у мом случају AVX), и једна и друга верзија су направљене и тестиране. Поред тога, програм је реализован и у C програмском језику, гдје сам испробао различите оптимизације, које ће бити наведене касније у документу. Сва времена извршавања и њихова поређења су адекватно извршена и документована. Укратко, идеја је била да се број елеманта који ће се израчунавати у сумама као и сами елементи учитају из фајла, те да се након тога они они искористе да бисмо израчунали параметре и уписали их у излазни фајл, гдје су и улазни и излазни фајл наведени као аргументи командне линије.

Поступак и услови тестирања

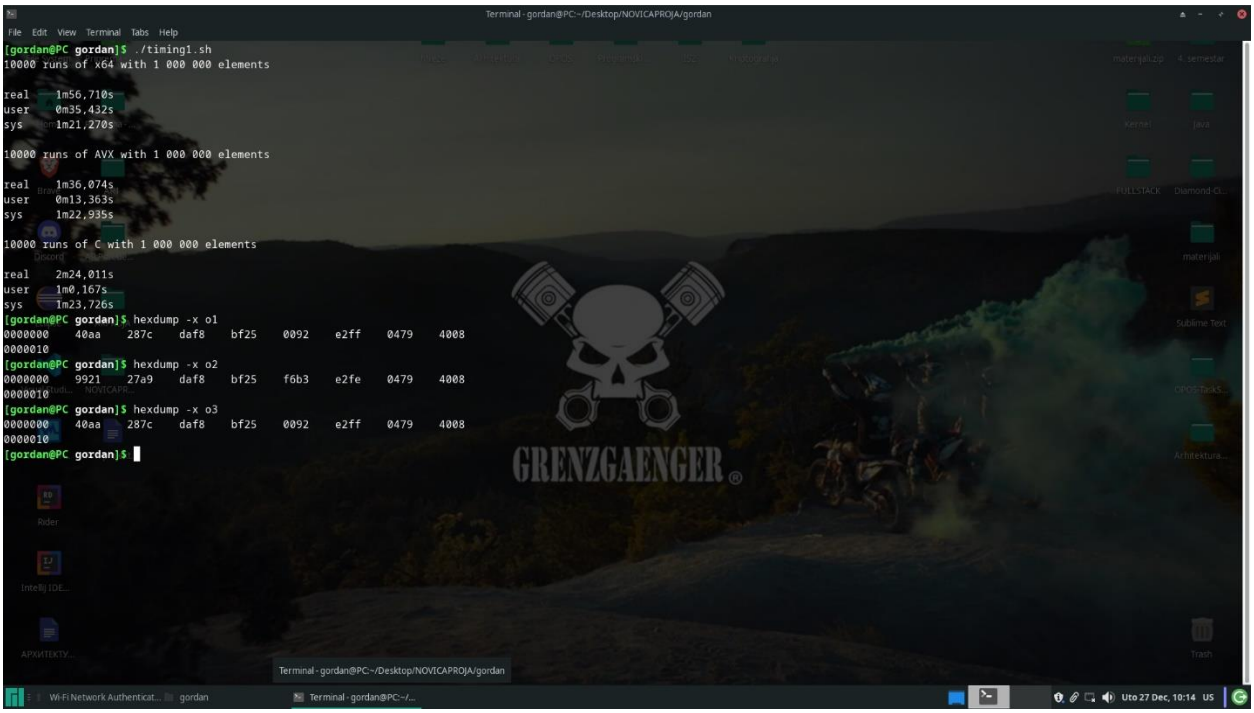
Тестирање компајлерских оптимизација је извршено користећи виртуалну машину на којој је инсталиран оперативни систем Ubuntu 22.04.1 LTS, гдје сам машини додијелио четири логичка језгра, иако их имам осам. Разлог за то је што ми виртуална машина (у овом случају VirtualBox) није дозвољавала већи број језгара, те сам добио и упозорење да би перформансе биле значајно деградиране. Процесор који сам користио је AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx 2.10GHz.

Тестирање асемблерских програма и C програма са циљем показивања резултата првог дијела пројектног задатка извршено је на Arch Linux оперативном систему, гдје је кориштен процесор Intel i7 4510U.

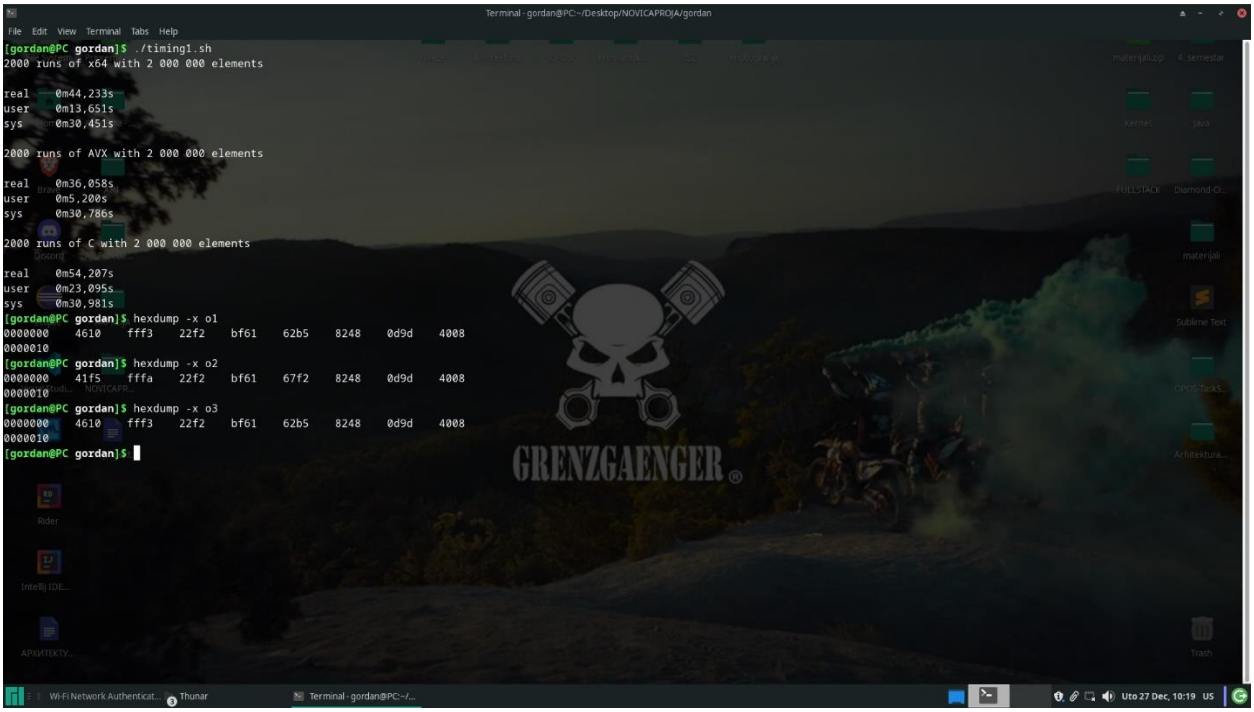
Да би се покренуло тестирање, кориштена је shell скрипта са moodle сајта предмета, коју сам модификовао на различите начине тако да су се програми извршавали различит број пута и са различитом количином улазних података.

Поређење времена извршавања

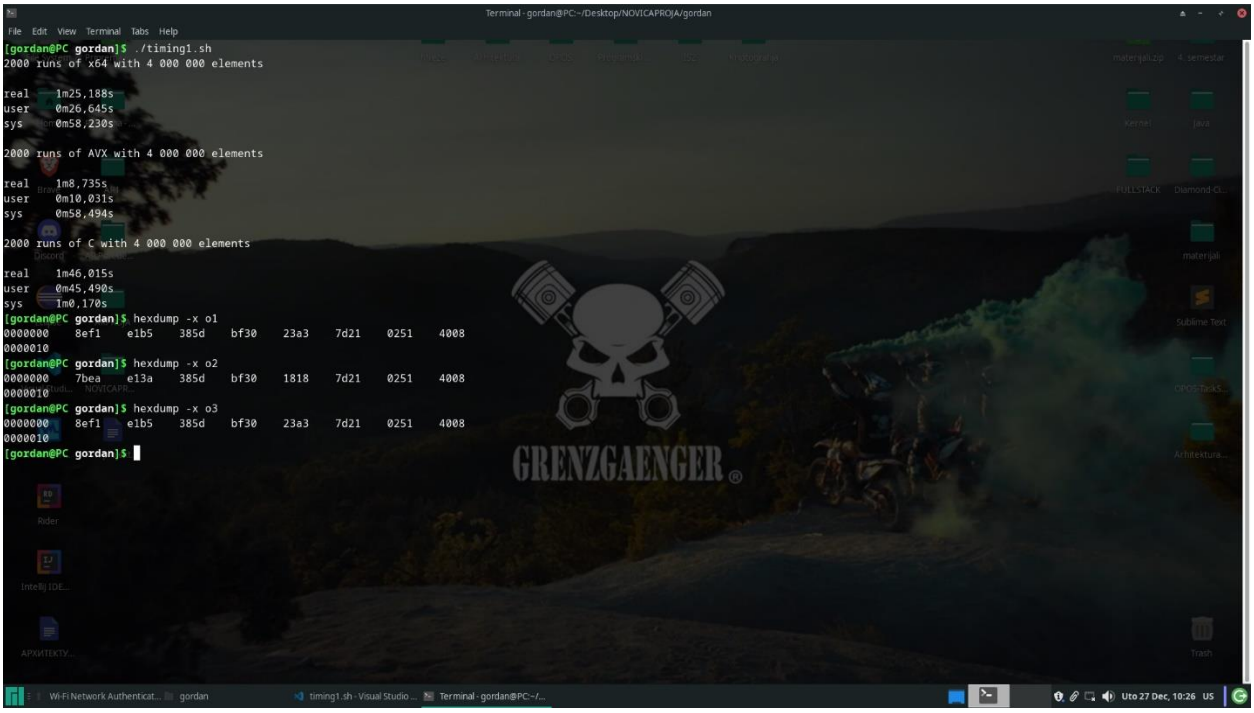
Поређење времена извршавања ћу представити помоћу screenshot-ова, на којима се јасно виде разлике у мјерењима извршавања, као и о томе да битних одступања у резултатима извршавања. Поред тога, приложио сам и документоване податке за float, о чијим проблемима ћу причати у закључку овог документа.



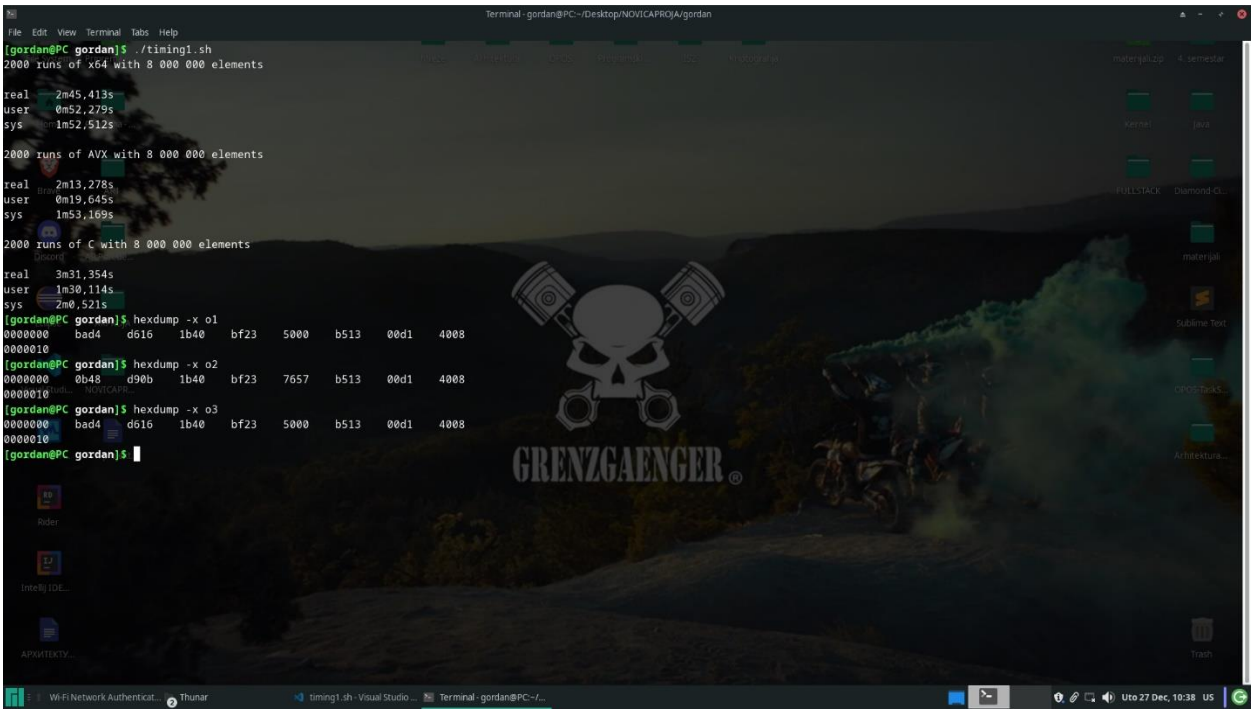
1-10 000 извршавања са 1 000 000 елемената



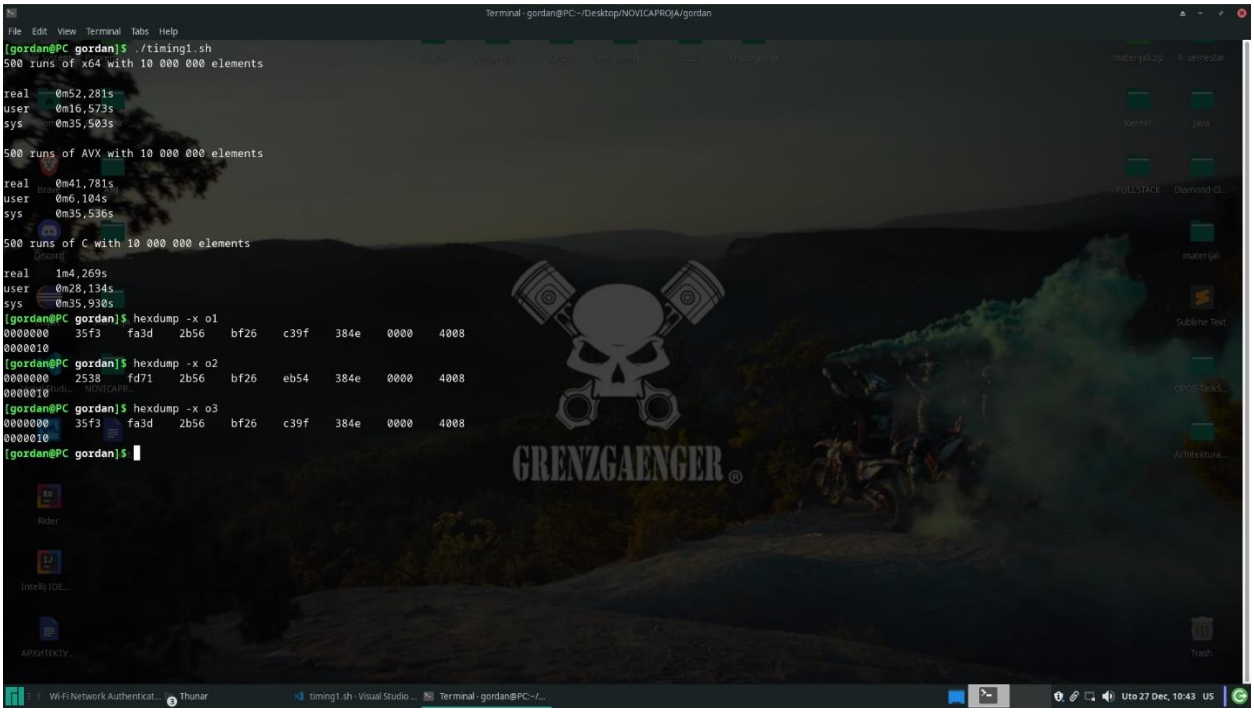
2-2 000 извршавања са 2 000 000 елемената



3-2 000 извршавања са 4 000 000 елемената



4-2 000 извршавања са 8 000 000 елемената



5-500 извршавања са 10 000 000 елемената

Поређење времена извршавања – Оптимизације gcc компајлера

За поређење времена извршавање приложићу релевантне screenshot-ове. На њима се виде које су компајлерске оптимизације коришћене, као и број извршавања програма и времена за које су се одговарајући програми извршили. У закључку документа ћу се детаљније осврнути на резултате поређења времена извршавања.

```
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$ ./timing.sh
10 runs of O0 with file that contains 5 000 000 elements

real    0m0,817s
user    0m0,253s
sys     0m0,564s
10 runs of O1 with file that contains 5 000 000 elements

real    0m0,595s
user    0m0,088s
sys     0m0,508s
10 runs of O2 with file that contains 5 000 000 elements

real    0m0,588s
user    0m0,082s
sys     0m0,505s
10 runs of O3 with file that contains 5 000 000 elements

real    0m0,590s
user    0m0,078s
sys     0m0,512s
10 runs of Ofast with file that contains 5 000 000 elements

real    0m0,586s
user    0m0,072s
sys     0m0,515s
10 runs of Ofastmavx with file that contains 5 000 000 elements

real    0m0,576s
user    0m0,071s
sys     0m0,505s
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$
```

Примјер бр. 1

```
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$ ./timing.sh
100 runs of O0 with file that contains 10 000 000 elements

real    0m13,856s
user    0m5,574s
sys     0m8,292s
100 runs of O1 with file that contains 10 000 000 elements

real    0m9,248s
user    0m1,280s
sys     0m7,978s
100 runs of O2 with file that contains 10 000 000 elements

real    0m9,260s
user    0m1,302s
sys     0m7,969s
100 runs of Ofast with file that contains 10 000 000 elements

real    0m9,088s
user    0m1,081s
sys     0m8,019s
100 runs of Ofastmavx with file that contains 10 000 000 elements

real    0m9,008s
user    0m1,051s
sys     0m7,969s
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$
```

Примјер бр. 2

```
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$ ./timing.sh
50 runs of O0 with file that contains 20 000 000 elements

real    0m14,393s
user    0m5,504s
sys     0m8,892s
50 runs of O1 with file that contains 20 000 000 elements

real    0m10,002s
user    0m1,395s
sys     0m8,609s
50 runs of O2 with file that contains 20 000 000 elements

real    0m9,927s
user    0m1,366s
sys     0m8,567s
50 runs of O3 with file that contains 20 000 000 elements

real    0m10,003s
user    0m1,350s
sys     0m8,656s
50 runs of Ofast with file that contains 20 000 000 elements

real    0m9,719s
user    0m1,123s
sys     0m8,601s
50 runs of Ofastmavx with file that contains 20 000 000 elements

real    0m9,708s
user    0m1,046s
sys     0m8,667s
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$
```

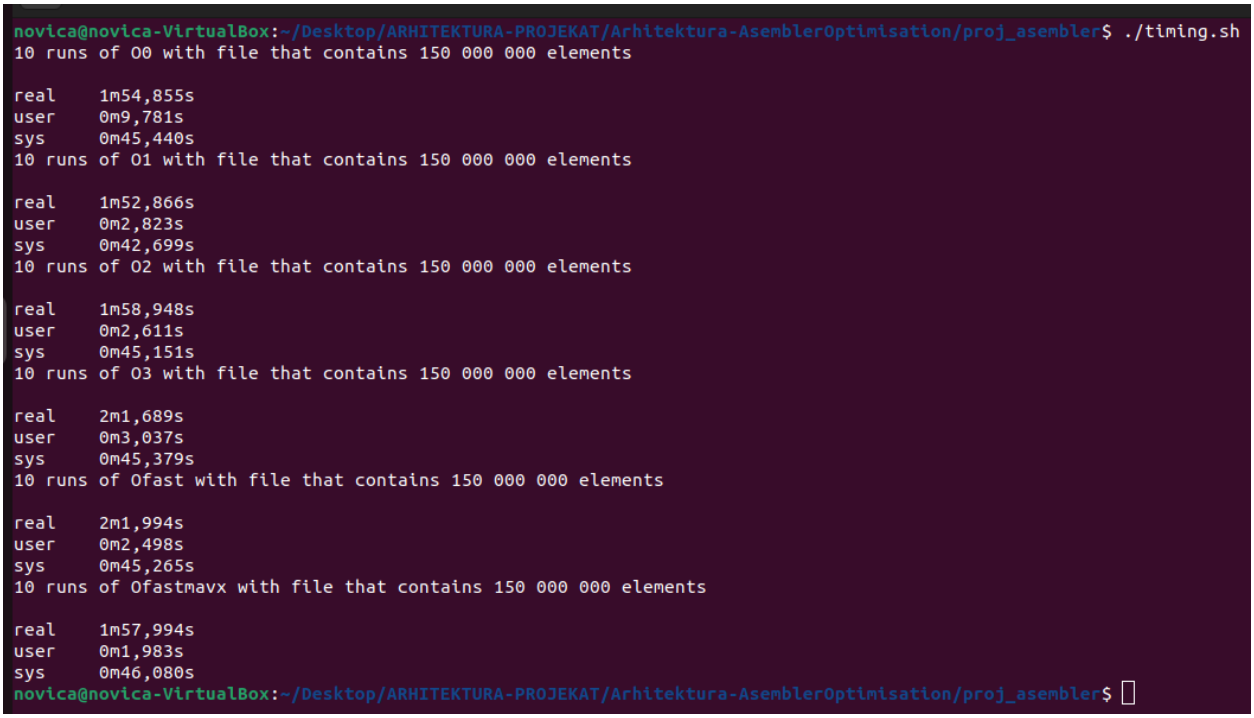
Примјер бр. 3

```
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$ ./timing.sh
10 runs of O0 with file that contains 50 000 000 elements

real    0m7,184s
user    0m2,086s
sys     0m5,095s
10 runs of O1 with file that contains 50 000 000 elements

real    0m4,955s
user    0m0,508s
sys     0m4,444s
10 runs of O2 with file that contains 50 000 000 elements

real    0m5,011s
user    0m0,486s
sys     0m4,522s
10 runs of O3 with file that contains 50 000 000 elements

real    0m5,230s
user    0m0,566s
sys     0m4,655s
10 runs of Ofast with file that contains 50 000 000 elements

real    0m4,826s
user    0m0,563s
sys     0m4,261s
10 runs of Ofastmavx with file that contains 50 000 000 elements

real    0m4,811s
user    0m0,310s
sys     0m4,501s
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$
```

Примјер бр. 4

```
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$ ./timing.sh
10 runs of O0 with file that contains 100 000 000 elements

real    0m31,154s
user    0m7,325s
sys     0m17,489s
10 runs of O1 with file that contains 100 000 000 elements

real    0m23,090s
user    0m2,048s
sys     0m16,341s
10 runs of O2 with file that contains 100 000 000 elements

real    0m22,723s
user    0m2,052s
sys     0m15,983s
10 runs of O3 with file that contains 100 000 000 elements

real    0m22,249s
user    0m1,991s
sys     0m15,776s
10 runs of Ofast with file that contains 100 000 000 elements

real    0m22,455s
user    0m1,602s
sys     0m16,125s
10 runs of Ofastmavx with file that contains 100 000 000 elements

real    0m21,324s
user    0m1,370s
sys     0m15,577s
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$
```

Примјер бр. 5

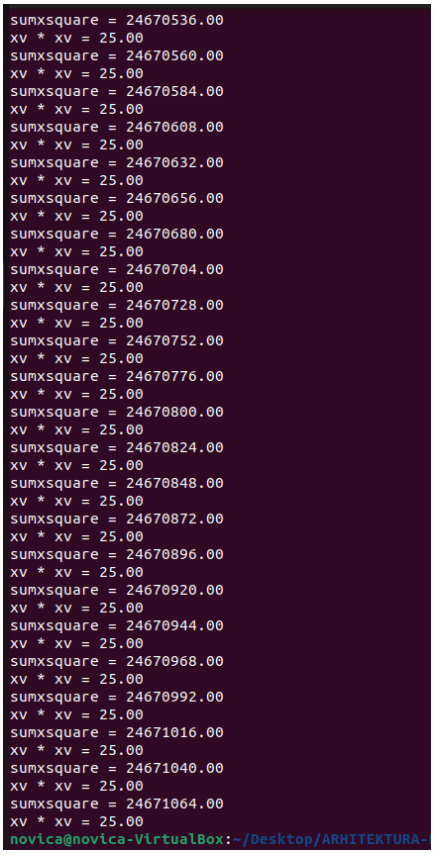

novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$ ./timing.sh
10 runs of O0 with file that contains 150 000 000 elements
real 1m54,855s
user 0m9,781s
sys 0m45,440s
10 runs of O1 with file that contains 150 000 000 elements
real 1m52,866s
user 0m2,823s
sys 0m42,699s
10 runs of O2 with file that contains 150 000 000 elements
real 1m58,948s
user 0m2,611s
sys 0m45,151s
10 runs of O3 with file that contains 150 000 000 elements
real 2m1,689s
user 0m3,037s
sys 0m45,379s
10 runs of Ofast with file that contains 150 000 000 elements
real 2m1,994s
user 0m2,498s
sys 0m45,265s
10 runs of Ofastmavx with file that contains 150 000 000 elements
real 1m57,994s
user 0m1,983s
sys 0m46,080s
novica@novica-VirtualBox:~/Desktop/ARHITEKTURA-PROJEKAT/Arhitektura-AsemblerOptimisation/proj_asembler$

Примјер бр. 6

Закључак

Као прву ставку ћу напоменути да сам првобитно реализовао асемблерске програме и C програм који је радио са floating point типовима података, али је то довело до одређених проблема. Наиме, floating point са рачунањем велике количине улазних података (као примјер можемо узети 1 000 000 улазних података гдје имамо случајне бројеве у распону од 1.0 до 100.0) доводи до тога да након одређеног броја рачунања floating point више не може адекватно рачунати податке, и при томе немамо одговарајуће резултате и прецизност. Упркос томе, документовао сам floating point и на другој слици се јасно види да резултати нису одговарајући. Да бих дошао до овог закључка, ставио сам 1 000 000 улазних података гдје сам генерисао 1 000 000 података које имају вриједност 5.0, а 1 000 000 које имају вриједност 6.0 и видио да су суме које се рачунају погрешне.

На screenshot-у испод је приказано како C програм надодаје 24.0 умјесто 25.0, иако се јасно види да је израчунато 25.0.



Након тога сам програм написао помоћу AVX инструкцијског скупа и као што је очекивано, није било проблема са сумирањем велике количине улазних података. Проблем који сам имао при оптимизацији је да се код YMM регистара не може користити shuffle, па сам то морао симулирати на свој начин. Упркос томе, убрзања су документована и програм ради како је очекивано. С програм је најспорији и при компајлирању нису кориштене никакве оптимизације. Што је већа количина улазних података и дуже вријеме извршавања, осјетиће се већа разлика између оптимизованог и неоптимизованог кода.

Као закључак се може истаћи да, иако рад са floating point подацима није довољно поуздан, доноси боље перформансе и резултате јер се може користити shuffle, а иначе код ymm регистара морамо радити непотребна копирања или помоћу помоћних промјенљивих или евентуално претварати packed double у packed single податке, па користити shuffle (што може довести до претходно наведених проблема), па враћати назад у packed double податке.

Када погледамо С компајлерске оптимизације и релевантне приложене screenshot-ове, можемо видјети којим редослиједом извршавања и које компајлерске оптимизације сам користио. Најспорије вријеме извршавања је очекивано за -O0 компајлерске оптимизације, а најбрже за Ofast, mavx2. Битно је примијетити да -O1, -O2 и -O3 компајлерске оптимизације дају готово идентично вријеме извршавања. Једино неочекивано одступање је рад са 150 000 000 елемената гдје Ofast mavx2 гдје -O1 даје најбоље резулате, а -O0 је одмах након њега по питању добрих перформанси.

Потребно је истаћи да проблем може представљати и количина улазних података са којом се ради, тако да треба да будемо опрезни и са становишта погледа отварања, копирања и генерално рада са фајловима, мада тестови не одступају од основне идеје, а то је да оптимизације значајно доприносе побољшању перфоманси и временима извршавања, како и асемблерског програма, тако и С програма компајлираних помоћу различитих компајлерских оптимизација.