АРХИТЕКТУРА РАЧУНАРА

ЗАДАТАК 2.2 - ОПТИМИЗАЦИЈА АЛГОРИТМА

ТЕМА: Израчунавање укупне количине простих бројева у опсезима из специфицираног скупа опсега

Аутор: Даријо Прерад

Датум: 23.02.2023.

Садржај:

[Увод 3](#_TOC_250006)

[Покретање програма и тестирање 3](#_TOC_250005)

[Времена извршавања алгоритма 5](#_TOC_250004)

[Примјер оптимизације кориштењем ,,SIMD“ приступа 7](#_TOC_250003)

[Провјера резултата различитих варијанти програма 8](#_TOC_250002)

Преглед просјечних вриједности и варијанси времена извршавања 9

[Графички приказ резултата 10](#_TOC_250001)

[Закључак 10](#_TOC_250000)

# УВОД

Пројектни задатак на тему оптимизације алгоритма реализован је над алгоритмом за израчунавање укупне количине простих бројева у опсезима из специфицираног скупа опсега. За оптимизацију алгоритма сам изабрао SIMD програмирање и паралелизацију на вишејезгреном процесору употребом OpenMP-а. Основни програм (без оптимизације), програм употребом SIMD програмирања, програм употребом OpenMP-а, као и комбинација оба приступа су реализовани у ,,С“ програмском језику. Додатна оптимизација је извршена у посљедњем рјешењу гдје је искориштена и идеја мемоизације која знатно побољшава перформансе у ситуацији када се програм извршава над истим (сличним) низом бројева. Свако од рјешења је тестирано на неколико различитих примјера улазних података, те је извршена провјера о валидности резултата сваког од њих.

# ПОКРЕТАЊЕ ПРОГРАМА И ТЕСТИРАЊЕ

Све реализације програма су тесиране помоћу ,,*shell“* скрипте која покреће дати програм задати број пута. Сви програми су компајлирани са -О3 оптимизацијом. Приликом компајлирања различитих верзија програма, кориштене су сљедеће команде:

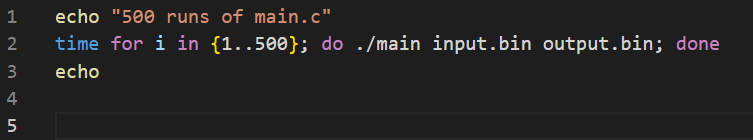
gcc -O3 -o Original Original.c  
gcc -O3 -fopenmp -o WithOMP WithOMP.c  
gcc -O3 -msse2 -o WithSIMD WithSIMD.c  
gcc -O3 -msse2 -fopenmp -o CombinedApproach CombinedApproach.c

Улазни фајлови су креирани кориштењем посебног помоћног програма ,,Potprogram“. Он је омогућио аутоматско креирање различитих скупова опсега бројева (за сваки скуп оквирни опсег се налази у називу) и програм је тестиран над свим скуповима.

input\_0-100.bin  
input\_0-1000.bin

input\_0-10000.bin

input\_0-100000.bin



*Слика 1. ,,shell“ скрипта кориштена за мјерење времена извршавања*

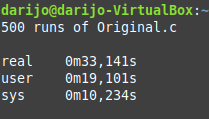
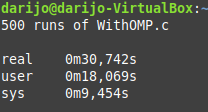
Алгоритми су извршавани на процесору ,,AMD Ryzen 7 3700U“ на оперативном систему ,,Linux Mint“ УНУТАР ВИРТУЕЛНЕ МАШИНЕ. Дати процесор има 4 физичка језгра а основна фреквенција рада је 2.3GHz.  
Виртуелној машини су додијељена два језгра и два гигабајта радне меморије.

# ВРЕМЕНА ИЗВРШАВАЊА АЛГОРИТМА

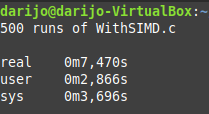
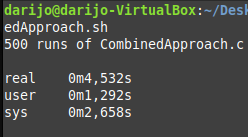
Поређење времена извршавања је представљено у виду screenshot-ова, на којима се јасно виде разлике у временима извршавања над различитим реализацијама алгоритама, као и различитих улазних података. За сваки улаз мјерења си извршена више пута а коначни резултати представљају просјечне вриједности. Прије мјерења резултата извршено је загријавање тј. вишеструко покретање програма без мјерења.

Провјеравамо времена извршавања свих алгоритама:  
-сваки програм се покреће 500 пута  
-улазни подаци (ПРВИ СЛУЧАЈ): 100 различитих опсега   
 (почетни опсег 0 - 100)

Добијена времена:

- ,,Original.c” – без оптимизација - ,,WithOMP.c” – користећи ,,OMP”паралелизацију 

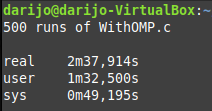
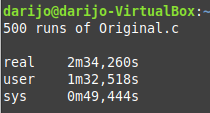
* ,,WithSIMD.c” – користећи - ,,CombinedApproach.c” – ,,SIMD“ приступ комбинација два приступа

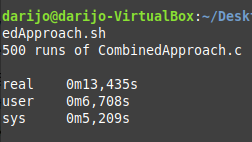
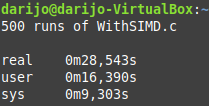
Провјеравамо времена извршавања свих алгоритама:  
-сваки програм се покреће 500 пута  
-улазни подаци (ДРУГИ СЛУЧАЈ): 100 различитих опсега   
 (почетни опсег 0 - 1000)

Добијена времена:

- ,,Original.c” – без оптимизација - ,,WithOMP.c” – користећи ,,OMP”паралелизацију



* ,,WithSIMD.c” – користећи - ,,CombinedApproach.c” – ,,SIMD“ приступ комбинација два приступа



Закључак: што је већи улазни фајл и што је више улазних података, то је уштеда на времену значајнија и уочљивија. У случају 10 000 улазних опсега, најбоље вријеме које остварује комбиновани алгоритам је 1мин37с. што је значајан напредак у односу на верзију алгоритма без оптимизације којој треба 20+ минута за извршавање.

# ПРИМЈЕР ОПТИМИЗАЦИЈЕ КОРИШТЕЊЕМ ,,SIMD” ПРИСТУПА

Примјер ,,SIMD“ приступа у пракси:

Посматрамо да ли је број 15 прост. Умјесто да га у свакој итерацији дијелимо са сљедећим бројем, идући од 2 до 15/2, ми остварујемо побољшање перформанси тако што у једној итерацији провјеравамо дјељивост са 4 боја. У свакој наредној итерацији узимамо сљедећа 4 броја итд.

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | 15 | 15 | 15 |

-тражимо остатак при дијељењу са:

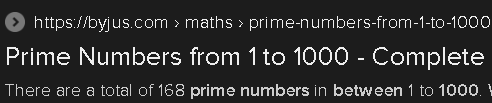
|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 3 | 4 | 5 |

-резултат:

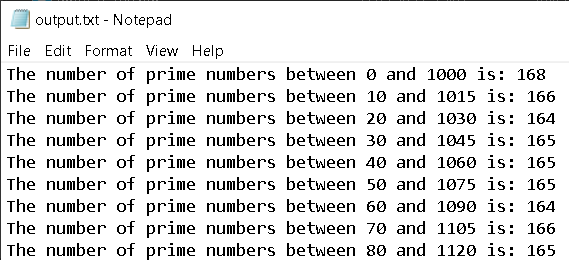
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 3 | 0 |

Након тога, можемо једноставно провјерити да ли имамо иједну нулу у резултантном вектору, ако имамо број није прост јер је очигледно дјељив са неким бројем из горњег вектора.

# ПРОВЈЕРА РЕЗУЛТАТА РАЗЛИЧИТИХ ВАРИЈАНТИ ПРОГРАМА



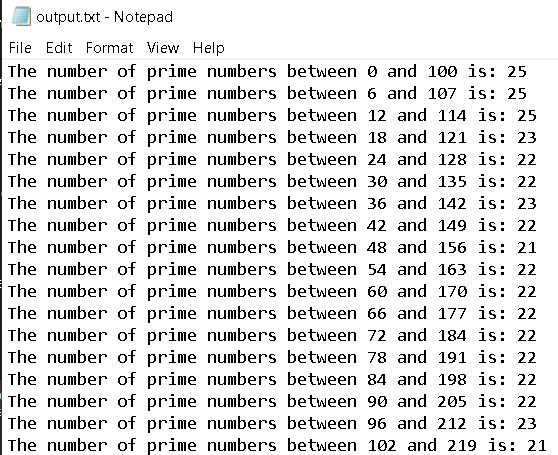
*Прости бројеви од 0 до 1000*



*Излаз генерисан од стране програма*



*Прости бројеви од 1 до 100*



*Излаз генерисан од стране програм*

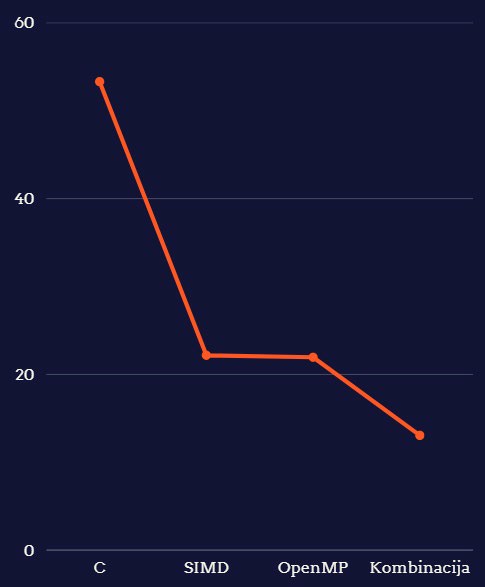
**ПРЕГЛЕД ПРОСЈЕЧНИХ ВРИЈЕДНОСТИ И ВАРИЈАНСИ ВРЕМЕНА ИЗВРШАВАЊА ПРОГРАМА**

Напомена: Мјерење извршено над почетним опсегом од 0 до 1000.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| РЕДНИ БР.  ПОКРЕТАЊА | БР.  ОПСЕГА | С  ПРОГРАМ | SIMD  ПРОГРАМ | OpenMP  ПРОГРАМ | КОМБИНАЦИЈА |
| 1 | 100 | 2m34s | 0m28s | 2m33s | 0m13s |
| 2 | 100 | 2m36s | 0m33s | 2m40s | 0m12s |
| 3 | 100 | 2m28s | 0m28s | 2m32s | 0m14s |
| ПРОСЈЕК | - | 2m36s | 0m29.6s | 2m35s | 0m12s |
| ВАРИЈАНСА | - | 0,01396 | 0,03017 | 0,02194 | 0,03816 |

# ГРАФИЧКИ ПРИКАЗ РЕЗУЛТАТА

## Напомена: Вриједности на графику су у секундама.



*Графички приказ 100 опсег*

**ЗАКЉУЧАК**

Након извршених свих неопходних тестирања над различитим скуповима улазних података, долазимо до закључка да оптимизације попут употребе ,,SIMD“ програмирања или ,,OpenMP“ представљају врло моћан алат помоћу којег можемо постићи значајна побољшања перформанси. Нарочито добре перформансе можемо постићи комбинацијом ове двије методе чиме можемо уштедити и сате времена чекања при обради врло велике количине података. Такође, ако се осврнемо на резултате мјерења, закључујемо да је ријеч о врло стабилном рјешењу које даје поприлично конзистентне резултате.