

АРХИТЕКТУРА РАЧУНАРА

ДРУГИ ПРОЈЕКТНИ ЗАДАТАК

ОПТИМИЗАЦИЈА АЛГОРИТМА

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

Аутор: Гордан Летић

Верзија: 1.0

Датум: 27.01.2023.

Садржај:

Увод	3
Поступак и услови тестирања	3
Поређење времена извршавања	4
Сликовити приказ оптимизације употребом SIMD програмирања	6
Провјера резултата различитих варијанти програма	7
Преглед просјечних вриједности и варијанси времена извршавања	8
Графички приказ резултата	9
Закључак	11

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 C програмском језику. Свако од рјешења је тестирано на неколико различитих примјера улазних података, те је извршена провјера о валидности резултата сваког од њих.

ПОСТУПАК И УСЛОВИ ТЕСТИРАЊА

Све реализације програма су тесиране помоћу *shell* скрипте која покреће дати програм задати број пута. Сви програми су компајлирани са -O3 оптимизацијом.

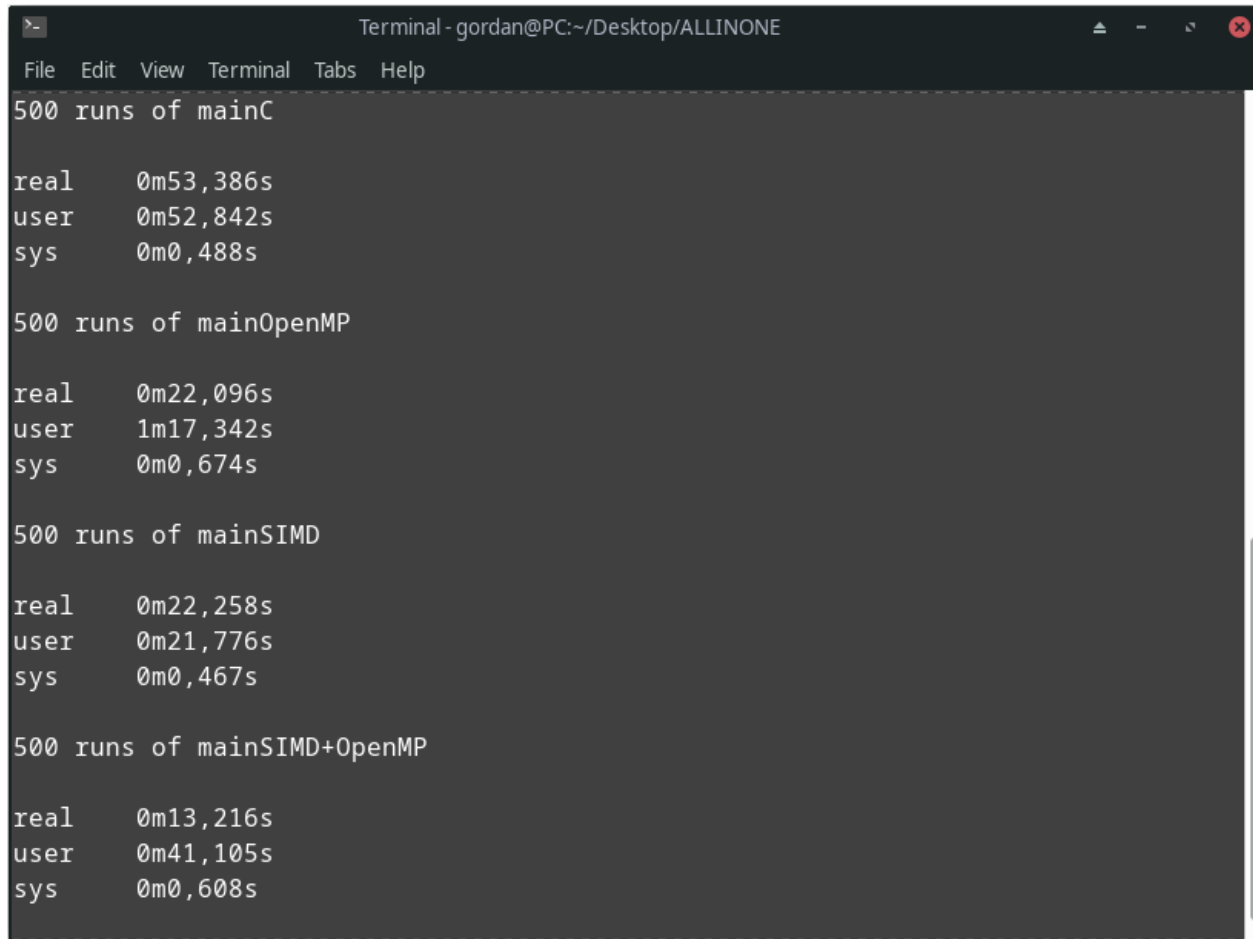
```
1  echo "500 runs of main.c"
2  time for i in {1..500}; do ./main input.bin output.bin; done
3  echo
4
5
```

Слика 1. Примјер садржаја *shell* скрипте за мјерење времена извршавања

Алгоритми су извршавани на процесору Intel i7-4510U, на оперативном систему Manjaro Linux (Arch дистрибуција). Дати процесор има 2 физичке језгре и 4 логичке (Hyper Threading), а основна фреквенција рада је 2.00GHz, а може да иде и до 3.10GHz при већем оптерећењу.

ПОРЕЂЕЊЕ ВРЕМЕНА ИЗВРШАВАЊА

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.



```
Terminal - gordan@PC:~/Desktop/ALLINONE
File Edit View Terminal Tabs Help

500 runs of mainC

real    0m53,386s
user    0m52,842s
sys     0m0,488s

500 runs of mainOpenMP

real    0m22,096s
user    1m17,342s
sys     0m0,674s

500 runs of mainSIMD

real    0m22,258s
user    0m21,776s
sys     0m0,467s

500 runs of mainSIMD+OpenMP

real    0m13,216s
user    0m41,105s
sys     0m0,608s
```

Слика 2. Поређење извршавања: У улазном фајлу 100 опсега од 1 до 1000

```

Terminal - gordan@PC:~/Desktop/ALLINONE
File Edit View Terminal Tabs Help
500 runs of mainC nMP.sh

real    2m36,826s
user    2m36,219s
sys     0m0,464s

500 runs of mainOpenMP

real    0m58,868s
user    3m44,080s
sys     0m0,665s

500 runs of mainSIMD

real    1m3,732s
user    1m3,209s
sys     0m0,454s

500 runs of mainSIMD+OpenMP

real    0m31,709s
user    1m53,962s
sys     0m0,799s

```

Слика 3. Поређење извршавања: У улазном фајлу 300 опсега од 1 до 1000

Са слика се јасно види разлика у времену извршавања различитих варијанти програма. Оно што је занимљиво је то да програм реализован употребом OpenMp-а и програм реализован употребом SIMD програмирања имају поприлично слична времена извршавања. Разлог за то је што, у OpenMP варијанти, процесор на коме је тестиран програм посједује четири thread-а, а сама оптимизација употребом SIMD програмирања смањује број итерација приближно четири пута. У случају да је био у питању процесор са више језгара за очекивати је да OpenMP варијанта програма има боље вријеме извршавања. Такође видимо да комбинацијом ова два приступа добијамо најбоље вријеме извршавања програма.

СЛИКОВИТИ ПРИКАЗ ОПТИМИЗАЦИЈЕ УПОТРЕБОМ SIMD ПРОГРАМИРАЊА

Ради бољег разумјевања погледајмо примјер употребе SIMD програмирања

Узмимо за примјер да је број 15 број који провјеравамо да ли је прост.

Нпр. `__m128i Xmm0`:

15	15	15	15
----	----	----	----

`__m128i Xmm1`:

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

`Xmm0 mod Xmm1`:

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

Итерацијом кроз резултат провјеравамо да ли је остатак при дјељењу једнак нули, уколико јесте инкрементујемо бројач. Овакав поступак понављамо кроз до $n/2$ елементата. Уколико је бројач једнак нули број је прост. У овом случају сљедећа итерација ће изгледати овако:

`Xmm0`:

15	15	15	15
----	----	----	----

`Xmm1`:

6	7	X	X
---	---	---	---

`Xmm0 mod Xmm1`:

3	1	X	X
---	---	---	---

У овом случају нас вриједности поља означених са X не занимају.

Као што видимо на овај начин смо ефикасно умјесто четири итерације за сваки од елемената (2, 3, 4, 5) извршили само једну итерацију чиме добијамо знатно убрзање. Вриједност бројача ће бити 2 што значи да број није прост.

ПРОВЈЕРА РЕЗУЛТАТА РАЗЛИЧИТИХ ВАРИЈАНТИ ПРОГРАМА

```

Terminal - gordan@PC:~/Desktop/ALLINONE
File Edit View Terminal Tabs Help

[gordan@PC ALLINONE]$ sh ./main input.bin output.bin
[gordan@PC ALLINONE]$ hexdump -x output.bin
00000000    04cd    0000
00000004

[gordan@PC ALLINONE]$ ./mainASM input.bin output.bin
[gordan@PC ALLINONE]$ hexdump -x output.bin
00000000    04cd    0000
00000004

[gordan@PC ALLINONE]$ ./mainSSE input.bin output.bin
[gordan@PC ALLINONE]$ hexdump -x output.bin
00000000    04cd    0000
00000004

[gordan@PC ALLINONE]$ ./mainSIMD input.bin output.bin
[gordan@PC ALLINONE]$ hexdump -x output.bin
00000000    04cd    0000
00000004

[gordan@PC ALLINONE]$ ./mainOpenMP input.bin output.bin
[gordan@PC ALLINONE]$ hexdump -x output.bin
00000000    04cd    0000
00000004

[gordan@PC ALLINONE]$ ./mainSIMD+OpenMP input.bin output.bin
[gordan@PC ALLINONE]$ hexdump -x output.bin
00000000    04cd    0000
00000004

[gordan@PC ALLINONE]$

```

Слика 4. Поређење резултата: У улазном фајлу један опсег од 1 до 10000

There are **1229** prime numbers between 1 and 10,000. They are given here below.

>> <https://jalu.ch/coding/primers/list>

Prime Numbers List – jalu.ch

About featured snippets Feedback

Enter hex number

04cd 16

Convert Reset Swap

Decimal number

1229 10

Слика 5. Потврда рјешења

ПРЕГЛЕД ПРОСЈЕЧНИХ ВРИЈЕДНОСТИ И ВАРИЈАНСИ ВРЕМЕНА ИЗВРШАВАЊА ПРОГРАМА

Напомена: Ради лакшег праћења сви опсези су од 1 до 1000.

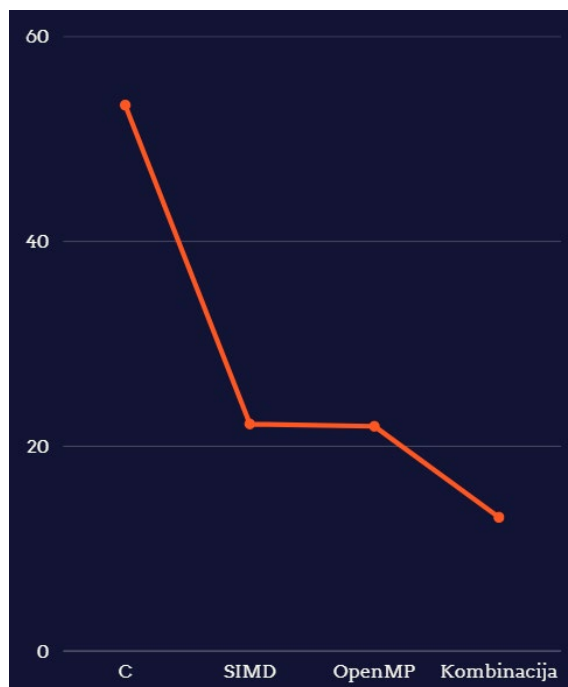
РЕДНИ БР. ПОКРЕТАЊА	БР. ОПСЕГА	С ПРОГРАМ	SIMD ПРОГРАМ	OpenMP ПРОГРАМ	КОМБИНАЦИЈА
1	100	53,140s	21,927s	21,745s	12,787s
2	100	53,395s	22,324s	22,000s	13,185s
3	100	53,386s	22,258s	22,096s	13,216
ПРОСЈЕК	-	53,307s	22,169	21,947	13,06
ВАРИЈАНСА	-	0,01396	0,03017	0,02194	0,03816

РЕДНИ БР. ПОКРЕТАЊА	БР. ОПСЕГА	С ПРОГРАМ	SIMD ПРОГРАМ	OpenMP ПРОГРАМ	КОМБИНАЦИЈА
1	200	1m47,366s	44,396s	40,601s	22,439s
2	200	1m45,007s	42,951s	40,387s	22,299s
3	200	1m44,976s	42,978s	40,385s	22,278s
ПРОСЈЕК	-	1m45,783s	43,441s	40,457s	22,338s
ВАРИЈАНСА	-	1,2531	0,4555	0,01027	0,00511

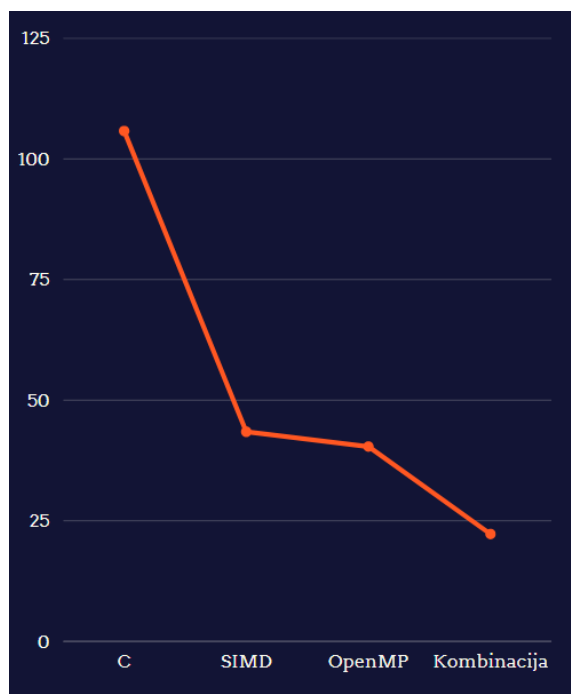
РЕДНИ БР. ПОКРЕТАЊА	БР. ОПСЕГА	С ПРОГРАМ	SIMD ПРОГРАМ	OpenMP ПРОГРАМ	КОМБИНАЦИЈА
1	300	2m37,031s	1m3,718s	58,921s	32,506s
2	300	2m36,848s	1m3,685s	58,832s	31,507s
3	300	2m36,826s	1m3,732s	58,868s	31,709s
ПРОСЈЕК	-	2m36,901s	1m3,711s	58,873s	31,907s
ВАРИЈАНСА	-	0,00844	0,00039	0,00134	0,186

ГРАФИЧКИ ПРИКАЗ РЕЗУЛТАТА

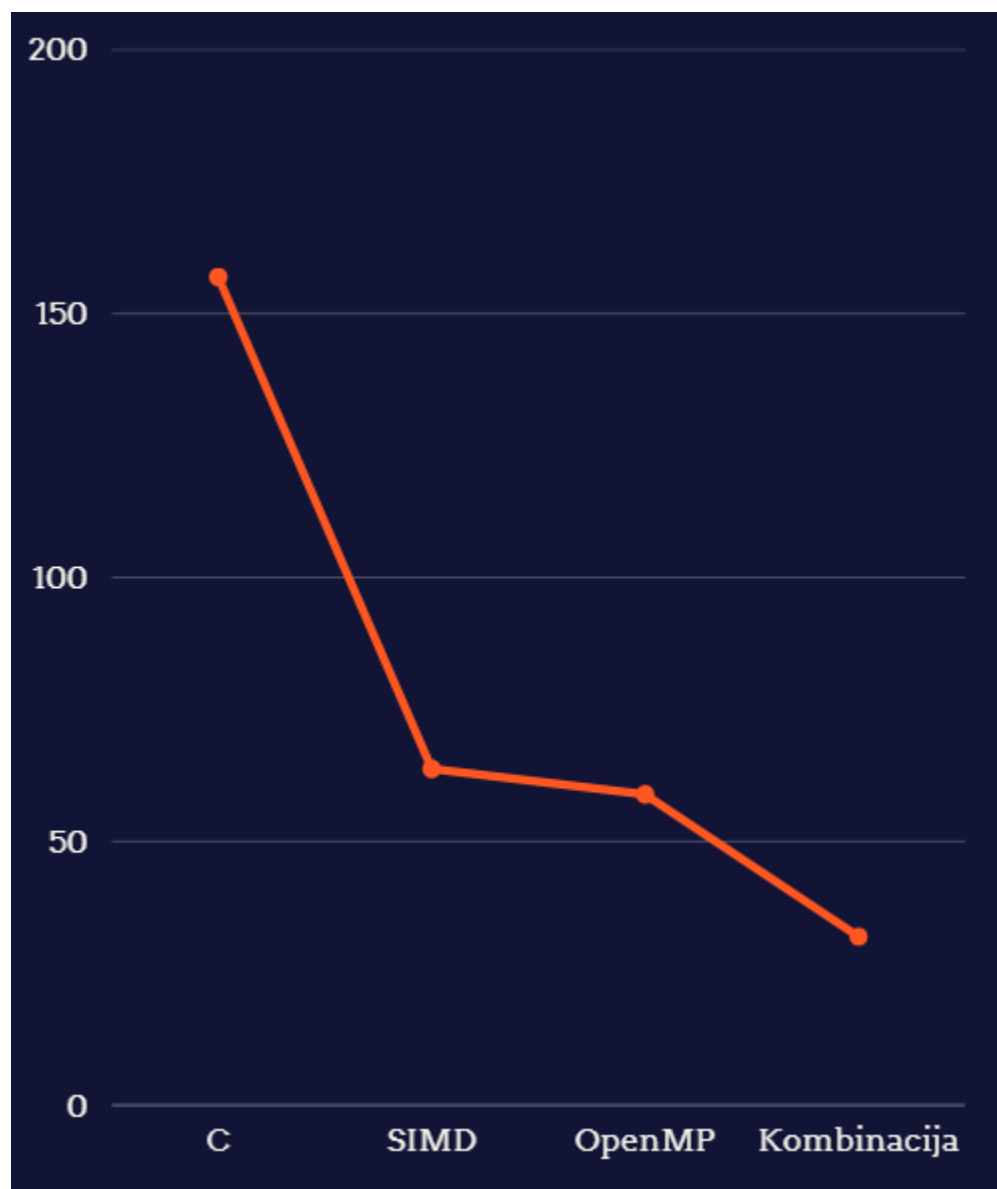
Напомена: Вриједности на графику су у секундама.



Слика 6. Графички приказ 100 опсега од 1 до 1000



Слика 7. Графички приказ 200 опсега од 1 до 1000



Слика 8. Графички приказ 300 опсега од 1 до 1000

ЗАКЉУЧАК

Након извршених свих неопходних тестирања, на неколико различитих улазних података, долазимо до закључка да оптимизације попут употребе SIMD програмирања или OpenMP-а представљају врло моћан „алат“ помоћу којег можемо постићи значајна побољшања у виду перформанси. Наручито добре перформансе можемо постићи комбинацијом ове двије методе чиме можемо уштедети 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. Овдје бих напоменуо да сам све програме извршавао са прикљученим пуњачем на лаптопу како бих постигао што конзистентније перформансе. У случају извршавања програма на батерији лаптопа, долази до значајнијих варијација у временима извршавања. Разлог за то је што процесор тада ради у моду штедње енергије, те оперативни систем не дозвољава константно максималне перформансе.