به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

برنامه‌نویسی چندهسته‌ای

گزارش کار آزمایش ۱

سید محمد حجازی حسینی

۹۷۳۳۰۲۰

زمستان ۱۴۰۰

هدف)

هدف از این آزمایش موازی‌سازی یک برنامه‌ی سریال و پایین آوردن زمان اجرا است. با استفاده از روش‌های متفاوت اینکار را انجام می‌دهیم؛ با critical block ها، با reduction و یکبار هم بدون استفاده از این‌ها و به صورت pad اضافه کردن برای جلوگیری از مشکلات cache.

سوالات مرحله اول)

به جای Visual Studio در این آزمایش از VSCode با MinGW استفاده شده است. البته با VS دانشگاه نیز کار کردم و خروجی‌های آن را نیز عکس گذاشته‌ام.

سوال ۱)

قسمتی که بیشترین زمان اجرا را دارد، for loop ای است که محاسبات در آن انجام می‌شود.

```
// Work Loop, do some work by looping VERYBIG times
for (j = 0; j<VERYBIG; j++)
{
    // increment check sum
    sum += 1;
    // Calculate first arithmetic series
    sumx = 0.0;
    for (k = 0; k<j; k++)
        sumx = sumx + (double)k;
    // Calculate second arithmetic series
    sumy = 0.0;
    for (k = j; k>0; k--)
        sumy = sumy + (double)k;
    if (sumx > 0.0)total = total + 1.0 / sqrt(sumx);
    if (sumy > 0.0)total = total + 1.0 / sqrt(sumy);
}
```

سوال ۲)

برای تکرار کل برنامه گذاشته شده است تا چندین بار زمان اجرا بررسی شود. علت آن این است که ممکن است در یک اجرا به دلایلی محاسبات زودتر یا دیرتر انجام شوند و در زمان اجرا تاثیرگذار باشند؛ مثلا CPU علاوه بر اجرای برنامه ما کار دیگری نیز انجام دهد. با تکرار این برنامه حالت‌های خاص مشخص می‌شوند و می‌توانیم با میانگین‌گیری نتیجه معقول‌تری داشته باشیم.

سوال ۳)

در حالت Debug اطلاعاتی مربوط به debug نیز در فایل‌های کامپایل شده وجود دارد در حالی که در حالت Release این اطلاعات وجود ندارد و کد کامپایل شده برای اجرا بهینه‌سازی شده است.

سوال ۴)

از روش تجزیه task parallelism و الگوی loop parallelism می‌توان استفاده کرد.

مرحله‌ی ۲)

۱ و ۲)

کد را به اول main اضافه کرده‌ام. در این آزمایش به جای استفاده از Visual Studio از VSCode استفاده کرده که با استفاده از کامپایلر MinGW فرآیند کامپایل و اجرای کد را انجام می‌دهد. برای اینکه از openmp استفاده شود باید از option مربوط به آن یعنی fopenmp- استفاده کنیم.

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: gcc.exe build active file",
            "command": "C:\\MinGW\\bin\\gcc.exe",
            "args": [
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe",
                "${file}"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": "build",
            "detail": "Task generated by Debugger."
        }
    ]
}
```

```
C lab1.c ×   {} tasks.json
1 > code > C lab1.c > main(void)
#include <omp.h>

const long int VERYBIG = 50000;
// ****************************************************************
int main(void)
{
    #ifndef _OPENMP
        printf("OpenMP is not supported, sorry!\n");
        getchar();
        return 0;
    #endif

    int i;
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL          Filter (e.g. text, !exclude)
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Warning: Debuggee TargetArchitecture not detected, assuming x86_64.
=cmd-param-changed,param="pagination",value="off"
=cmd-param-changed,param="args",value="2>CON 1>CON <CON"
[New Thread 6440.0x3540]
[New Thread 6440.0x2fe0]
Loaded 'C:\Windows\SysWOW64\kernel32.dll'. Symbols loaded.
Loaded 'C:\Windows\SysWOW64\KernelBase.dll'. Symbols loaded.
Loaded 'C:\Windows\SysWOW64\msvcrt.dll'. Symbols loaded.
OpenMP is not supported, sorry!
The program 'c:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code\lab1.exe' has exited with code 0 (0x00000000).
>
```

```
.vscode > {} tasks.json > [ ] tasks > {} 0 > [ ] args
1  {
2      "version": "2.0.0",
3      "tasks": [
4          {
5              "type": "cppbuild",
6              "label": "C/C++: gcc.exe build active file",
7              "command": "C:\\MinGW\\bin\\gcc.exe",
8              "args": [
9                  "-fopenmp",
10                 "-o",
11                 "${fileDirname}\\${fileBasenameNoExtension}.exe",
12                 "${file}"
13             ],
14             "options": {
15                 "cwd": "${fileDirname}"
16             },
17             "problemMatcher": [
18                 "$gcc"
19             ],
20             "group": "build",
21             "detail": "Task generated by Debugger."
22         }
23     ]
24 }
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                    Filter (e.g. text, !exclude)
Loaded 'C:\MinGW\bin\libgcc_s_dw2-1.dll'. Symbols loaded.
Loaded 'C:\MinGW\bin\pthreadGC-3.dll'. Symbols loaded.
Serial Timings for 50000 iterations
Time Elapsed: 7.563000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.456000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.377000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.403000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.361000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.378000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.301000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.504000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.692000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.742000 Secs, Total = 30.656747, Check Sum = 50000
The program 'c:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code\lab1.exe' has exited with code 0 (0x00000000).
```

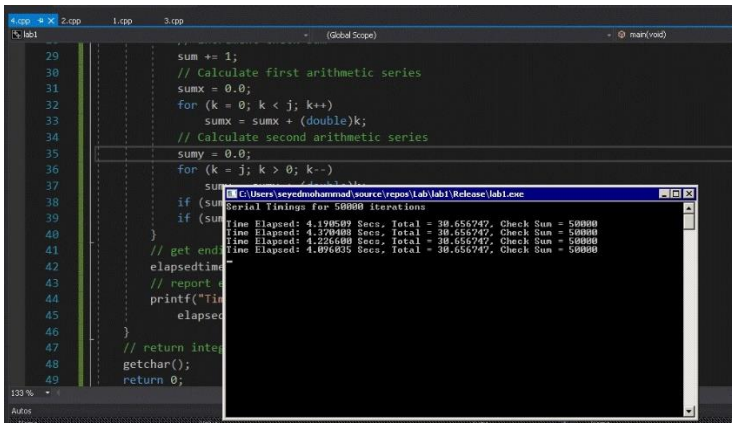
با تعریف کردن یک task در VSCode می‌توان این argument را اضافه کرد و به این ترتیب دیگر به مشکلی نخواهیم داشت. البته من از یک extension به اسم code runner استفاده می‌کنم که همین کار را در نهایت انجام می‌دهد.

۳ و ۴)

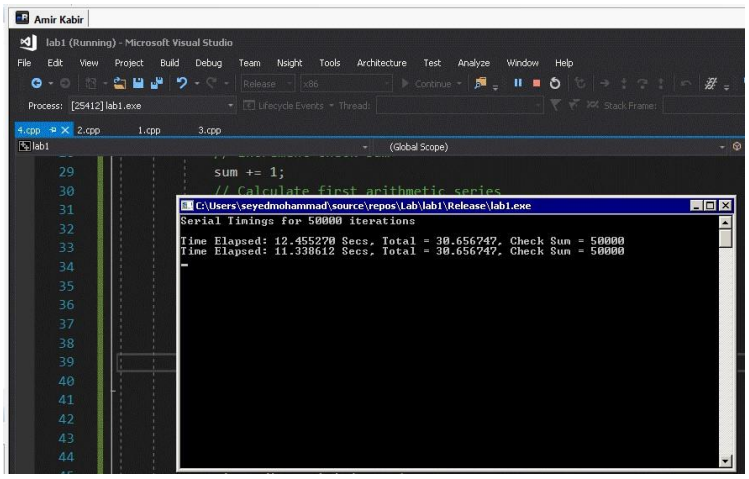
در حالت سریال:

```
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session> cd "c:
\" ; if ($?) { gcc -fopenmp lab1.c -o lab1 } ; if ($?) { .\lab1 }
Serial Timings for 50000 iterations

Time Elapsed: 7.349000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.344000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.253000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.333000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.305000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.281000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.297000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.488000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.348000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 7.477000 Secs, Total = 30.656747, Check Sum = 50000
end
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code>
```

روی سرور دانشگاه به صورت سریال با Optimization O2:



روی سرور دانشگاه به صورت سریال با Optimization disabled:



در حالت اضافه کردن parallel به قبل از for اصلی:

```
// Work Loop, do some work by looping VERYBIG times
#pragma omp parallel for
for (j = 0; j<VERYBIG; j++)
{
```

```
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code> co
\1\code\" ; if ($?) { gcc -fopenmp lab1.c -o lab1 } ; if ($?) { .\lab1 }
Serial Timings for 50000 iterations

Time Elapsed: 88.675000 Secs, Total = 1.#INF00, Check Sum = 47275
Time Elapsed: 74.877000 Secs, Total = 1.#INF00, Check Sum = 47580
Time Elapsed: 87.822000 Secs, Total = 1.#INF00, Check Sum = 47062
```

دلیل تفاوت آن این است که متغیرهایی که بیرون از block اصلی تعریف شده‌اند به صورت shared می‌شوند که دچار حالت مسابقه می‌شود و در روند اجرای برنامه تاثیرگذار است. چون k هم shared است، ممکن است که برنامه حتی به پایان نرسد.

۵ و ۶)

```
    total = 0.0;
    // Work Loop, do some work by looping VERYBIG times
    #pragma omp parallel for private(k, sumx, sumy)
    for (j = 0; j<VERYBIG; j++)
    {
        // increment check sum
```

```
\1\code\" ; if ($?) { gcc -fopenmp lab1.c -o lab1 } ; if ($?) { .\lab1 }
Serial Timings for 50000 iterations

Time Elapsed: 1.945000 Secs, Total = 30.642286, Check Sum = 49983
Time Elapsed: 1.924000 Secs, Total = 30.636800, Check Sum = 49977
Time Elapsed: 1.923000 Secs, Total = 30.615169, Check Sum = 49973
Time Elapsed: 1.894000 Secs, Total = 30.609865, Check Sum = 49980
Time Elapsed: 1.947000 Secs, Total = 30.618347, Check Sum = 49984
Time Elapsed: 1.892000 Secs, Total = 30.634618, Check Sum = 49981
Time Elapsed: 1.901000 Secs, Total = 30.637686, Check Sum = 49983
Time Elapsed: 1.932000 Secs, Total = 30.644252, Check Sum = 49983
Time Elapsed: 1.893000 Secs, Total = 30.627094, Check Sum = 49985
Time Elapsed: 1.924000 Secs, Total = 30.584406, Check Sum = 49979
end
```

۷)

```c
#pragma omp parallel for private(k, sumx, sumy)
for (j = 0; j<VERYBIG; j++)
{
    // increment check sum
    #pragma omp critical
        sum += 1;
    // Calculate first arithmetic series
    sumx = 0.0;
    for (k = 0; k<j; k++)
        sumx = sumx + (double)k;
    // Calculate second arithmetic series
    sumy = 0.0;
    for (k = j; k>0; k--)
        sumy = sumy + (double)k;
    if (sumx > 0.0){
        #pragma omp critical
            total = total + 1.0 / sqrt(sumx);
    }
    if (sumy > 0.0){
        #pragma omp critical
            total = total + 1.0 / sqrt(sumy);
    }
}
```

```
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code> cd "
\1\code\" ; if ($?) { gcc -fopenmp lab1.c -o lab1 } ; if ($?) { .\lab1 }
Serial Timings for 50000 iterations

Time Elapsed: 1.995000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.906000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.909000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.917000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.933000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.899000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.956000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.950000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.926000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.884000 Secs, Total = 30.656747, Check Sum = 50000
end
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code>
```

اجرا روی سرور دانشگاه:

```
C:\Users\seyedmohammad\source\repos\Lab\lab1\Release\lab1.exe
Serial Timings for 50000 iterations

Time Elapsed: 2.947593 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.410910 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.615809 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.525841 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.732951 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.139267 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.709382 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.560152 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.690937 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.474568 Secs, Total = 30.656747, Check Sum = 50000
```

۸ و ۹)

```
total = 0.0;
// Work Loop, do some work by looping VERYBIG times
#pragma omp parallel for private(k, sumx, sumy) reduction(+:sum, total)
for (j = 0; j<VERYBIG; j++)
{
    // increment check sum
    sum += 1;
    // Calculate first arithmetic series
    sumx = 0.0;
    for (k = 0; k<j; k++)
        sumx = sumx + (double)k;
    // Calculate second arithmetic series
    sumy = 0.0;
    for (k = j; k>0; k--)
        sumy = sumy + (double)k;
    if (sumx > 0.0)total = total + 1.0 / sqrt(sumx);
    if (sumy > 0.0)total = total + 1.0 / sqrt(sumy);
}
// get ending time and use it to determine elapsed time
```

```
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code
\1\code\" ; if ($?) { gcc -fopenmp lab1.c -o lab1 } ; if ($?) { .\lab
Serial Timings for 50000 iterations

Time Elapsed: 1.991000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.917000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.918000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.886000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.893000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.895000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.899000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.917000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.879000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.889000 Secs, Total = 30.656747, Check Sum = 50000
end
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code
```

زمان اجرا تغییری نمی‌کند.

اجرای این کد روی سرور دانشگاه:

```
C:\Users\seyedmohammad\source\repos\Lab\lab1\Release\lab1.exe
Serial Timings for 50000 iterations

Time Elapsed: 2.630630 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.649945 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.585991 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.635950 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.579025 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.680512 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.635167 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.389696 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.461018 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 2.615186 Secs, Total = 30.656747, Check Sum = 50000
```

۱۰)

کد را تغییر می‌دهیم و از راه padding برای جلوگیری از update شدن ناخواسته در cache می‌شویم.

```
int i, n, mainSum;
long int j, k;
double sumx, sumy, mainTotal;
double starttime, elapsedtime;
// ----------------------------------------------------------------------
// Output a start message
printf("Serial Timings for %d iterations\n\n", VERYBIG);
// repeat experiment several times
for (i = 0; i<10; i++)
{
    // get starting time56 x CHAPTER 3 PARALLEL STUDIO XE FOR THE IMPATIENT
    starttime = omp_get_wtime();
    // reset check sum & running total
    long int sum[NUM_THREADS][PAD] = {0};
    double total[NUM_THREADS][PAD] = {0.0};
    // Work Loop, do some work by looping VERYBIG times
    omp_set_num_threads(NUM_THREADS);
    #pragma omp parallel for private(k, sumx, sumy)
    for (j = 0; j<VERYBIG; j++)
    {
        int id;
        id = omp_get_thread_num();
        // increment check sum
        sum[id][0] += 1;
        // Calculate first arithmetic series
        sumx = 0.0;
        for (k = 0; k<j; k++)
            sumx = sumx + (double)k;
        // Calculate second arithmetic series
        sumy = 0.0;
        for (k = j; k>0; k--)
            sumy = sumy + (double)k;
        if (sumx > 0.0)total[id][0] = total[id][0] + 1.0 / sqrt(sumx);
        if (sumy > 0.0)total[id][0] = total[id][0] + 1.0 / sqrt(sumy);
    }

    for (n = 0, mainTotal = 0; n<NUM_THREADS; n++) mainTotal += total[n][0];
    for (n = 0, mainSum = 0; n<NUM_THREADS; n++) mainSum += sum[n][0];
    // get ending time and use it to determine elapsed time
    elapsedtime = omp_get_wtime() - starttime;
    // report elapsed time
    printf("Time Elapsed: %f Secs, Total = %lf, Check Sum = %ld\n",
        elapsedtime, mainTotal, mainSum);
```

نتیجه:

```
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code> cd "c
\1\code\" ; if ($?) { gcc -fopenmp lab1-3.c -o lab1-3 } ; if ($?) { .\lab1-3
Serial Timings for 50000 iterations

Time Elapsed: 1.934000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.867000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.901000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.878000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.880000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.899000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.896000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.867000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.898000 Secs, Total = 30.656747, Check Sum = 50000
Time Elapsed: 1.912000 Secs, Total = 30.656747, Check Sum = 50000
PS C:\Users\Mohammad\Desktop\Multicore Programming\Lab\Session\1\code>
```

سوالات مرحله ۲)

سوال ۱)

برابر با تعداد core ها است. در سیستم من که core i7 هست، ۸ core وجود دارد.

```
31        // Work Loop, do some work by looping VERYBIG times
32        #pragma omp parallel for private(k, sumx, sumy) reduction(+:sum, total)
33        for (j = 0; j<VERYBIG; j++)
34        {
35            printf("Number of threads: %d\n", omp_get_num_threads());
36            // increment check sum
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: Cod

```
Number of threads: 8
```

سوال ۲)

بله می‌توان اینکار را کرد؛ البته برای total ابتدا باید 1 / sqrt را محاسبه و سپس از atomic block استفاده کنیم تا جمع بزنیم؛ زیرا نمی‌توان محاسبه‌ی آن را در atomic انجام داد.

سوال ۳)

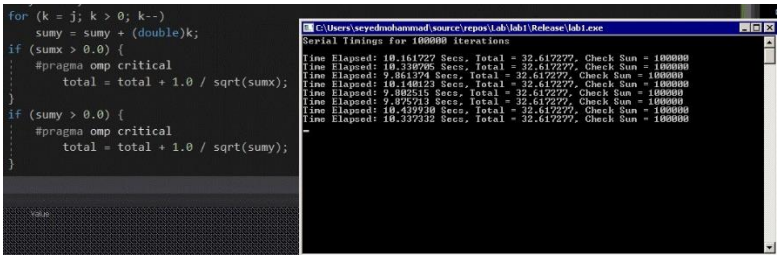
برای max threads در اجرا برای 50k، 60k و 90k روی کامپیوتر شخصی بار تفاوتی نمی‌کنند.

برای تعداد دو نخ و 50000 بار روی کامپیوتر شخصی نیز آزمایش شد و تفاوتی نکردند.

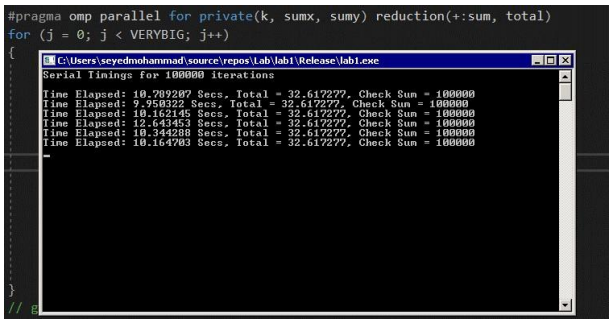
برای max threads 100 و 1000 نیز روی کامپیوتر شخصی تفاوتی نکردند.

روی سرور دانشگاه نیز برای 100k نیز اجرا کردم:

حالت 100k critical با دو thread:



حالت 100k reduction با دو thread:



1k reduction با دو thread:

```
omp_set_num_threads(2);
#pragma omp parallel for private(k, sumx, sumy) reduction(+:sum, total)
    sumy = 0.0;
    for (k = j; k > 0; k--)
```

```
C:\Users\seyedmohammad\source\repos\Lab\lab1\Release\lab1.exe
Serial Timings for 1000 iterations

Time Elapsed: 0.002849 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001859 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001868 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001893 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001872 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001846 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001875 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001841 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001844 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001847 Secs, Total = 19.590488, Check Sum = 1000
```

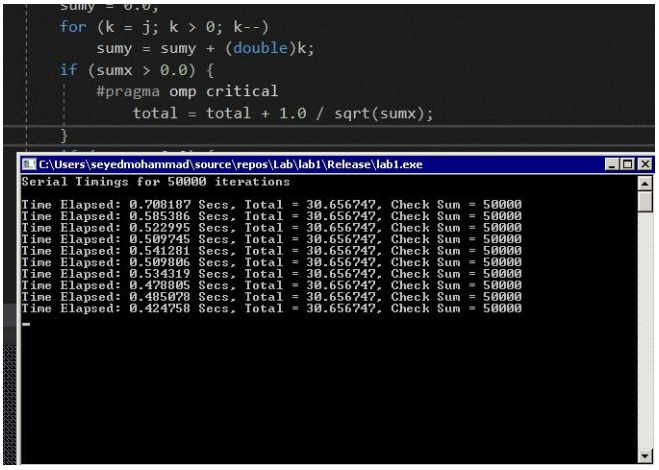
1k critical با دو thread:

```
    if (sumx > 0.0) {
        #pragma omp critical
            total = total + 1.0 / sqrt(sumx);
    }
    if (sumy > 0.0) {
        #pragma omp critical
            total = total + 1.0 / sqrt(sumy);
```

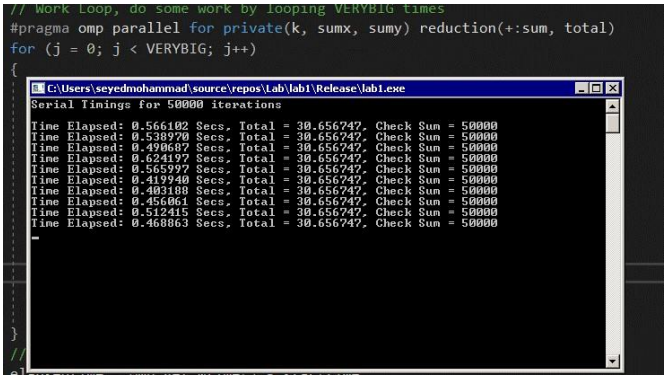
```
C:\Users\seyedmohammad\source\repos\Lab\lab1\Release\lab1.exe
Serial Timings for 1000 iterations

Time Elapsed: 0.003260 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002013 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002010 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002009 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002046 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.001981 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002055 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002040 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002019 Secs, Total = 19.590488, Check Sum = 1000
Time Elapsed: 0.002063 Secs, Total = 19.590488, Check Sum = 1000
```

برای 1k روی سرور دانشگاه کمی متفاوت است، reduction کمی سریع‌تر است.

50k critical با max threads:



50k reduction با max threads:



خروجی‌ها آنچنان فرقی ندارند.