Laborator 5 – Rezolvare

**Ex 1**. *Create a program with an array of 1 million entries with values of 1. In a loop compute the sum of all elements 1024 times. Do the same but before accessing each element invalidate it from the cache. Compute the time to do that in both cases and tell if there is any difference.*

#include <stdio.h>

#include <time.h>

#include <x86intrin.h>

// invalidate cache from CPU

static inline void clflush(volatile void \*p)

{

asm volatile("clflush %0" : "+m" (\*(volatile char \*)p));

}

int main()

{

int array[1000000];

int i, j, sum = 0;

for(i = 0; i < 1000000; i++)

{

array[i] = 1;

clflush(&array[i]); // we invalidate them from the cache

}

// First Scenario

clock\_t t;

t = clock();

for(j = 0; j < 1024; j++)

for(i = 0; i < 1000000; i++) sum += array[i];

t = clock() - t;

double time\_taken = ((double)t)/CLOCKS\_PER\_SEC; // in seconds

printf("Sum: %d\n", sum);

printf("Seconds First Scenario: %f \n", time\_taken);

// Second Scenario

sum = 0;

clock\_t y;

y = clock();

for(j = 0; j < 1024; j++)

for(i = 0; i < 1000000; i++)

{

sum += array[i];

clflush(&array[i]);

}

y = clock() - y;

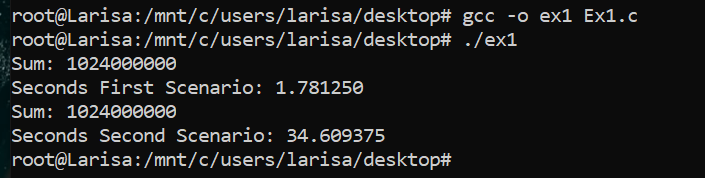
double time = ((double)y)/CLOCKS\_PER\_SEC; // in seconds

printf("Sum: %d\n", sum);

printf("Seconds Second Scenario: %f \n", time);

return 0;

}



După cum se poate remarca, durează mult mai mult să calculăm suma dacă curățăm memoria cache de fiecare dată (al doilea scenariu).

**Ex 2**. *Declare an array of char pointers like this: char \* array[256]. Allocate for each element of this array a 4096 bytes buffer, like this: array[i] = (char \*) malloc(4096). Invalidate from cache each element like array[i][1000] where i=0…255. Store a value in array[secret][1000], where secret is a value you choose. Now measure the time in cpu cycles needed to access each element like array[i][1000], print it and see if you can guess from what you have printed the value of secret.*

#include <stdio.h>

#include <stdlib.h>

#include <x86intrin.h>

// invalidate cache from CPU

static inline void clflush(volatile void \*p)

{

asm volatile("clflush %0" : "+m" (\*(volatile char \*)p));

}

int main()

{

char \*array[256];

int i, sum = 0;

for(i = 0; i < 256; i++)

array[i] = (char\*)malloc(4096);

for(i = 0; i < 256; i++)

clflush(&array[i][1000]);

array[11][1000] = 58;

unsigned int dummy;

unsigned long long t1, t2;

for(i = 0; i < 256; i++)

{

t1 = \_\_rdtscp(&dummy);

sum += array[i][1000];

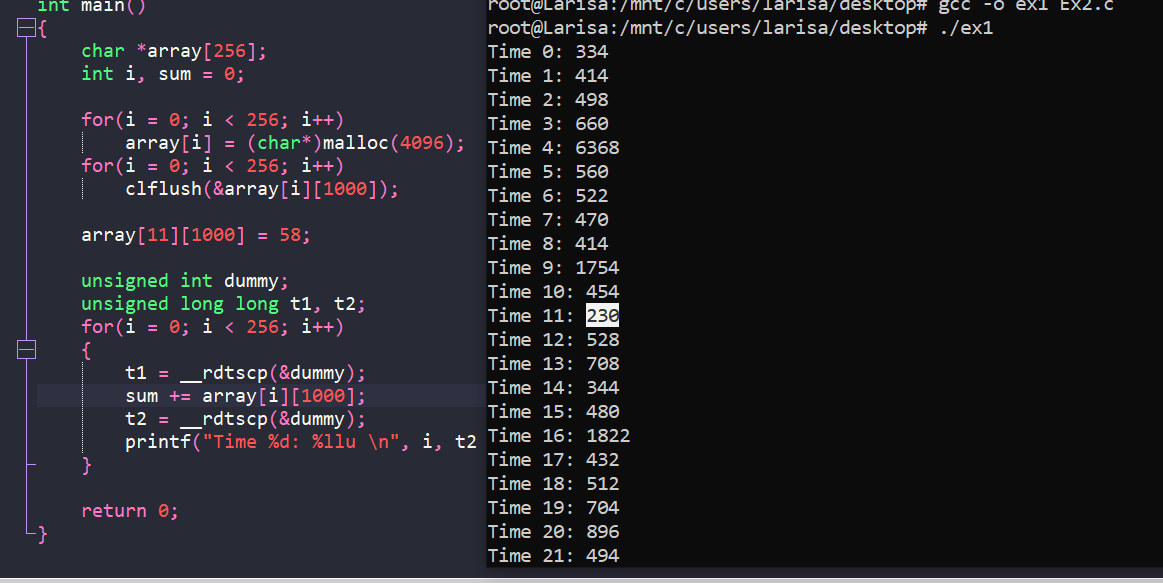
t2 = \_\_rdtscp(&dummy);

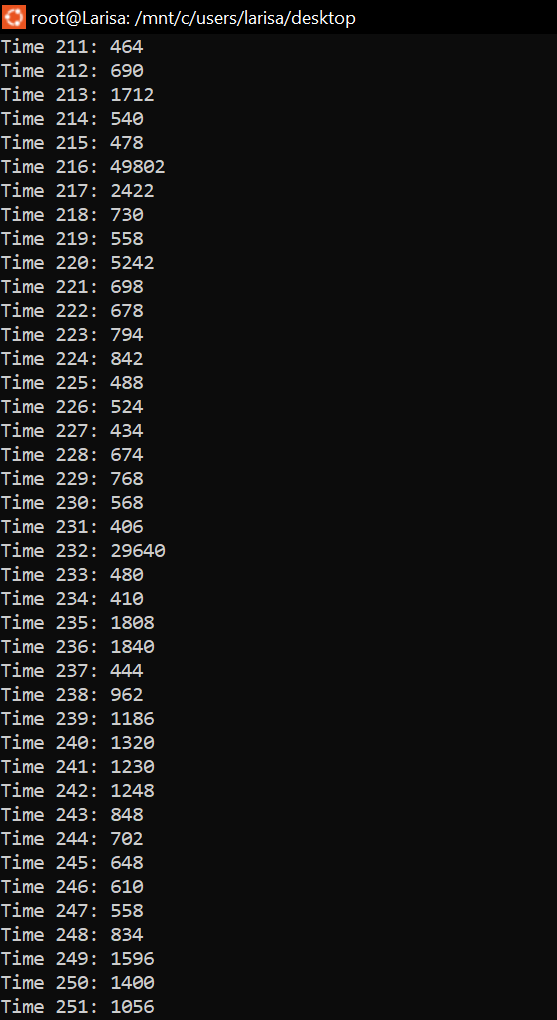
printf("Time %d: %llu \n", i, t2 - t1);

}

return 0;

}





După cum se poate remarca, timpul, în ciclii, este mai mic pentru adresa care a fost accesată anterior.

Ex 3.