CS203P Lab 1

January 23, 2024

Question 1.

Compile the attached program *ast.cpp* using the commands *g++ -O0 -o astO0.out ast.cpp* and *g++ -O3 -o astO3.out ast.cpp*. Compare the sizes of the resulting binaries and their runtimes.

Question 2.

Add a statement to print the value of the sum (at the end of the computation) in the code. Compile and compare the runtimes.

Question 3.

Interchange i and j when accessing the array and compile with *-O0*. Run each version of the code at least ten times to calculate the average runtime. Compare the runtimes between the two versions, change the value of N, and repeat the experiment. Finally, plot a graph with N on the x-axis and Time on the y-axis.

Question 4.

Implement a program to store 10,000 elements using two methods. In Version 1, employ a static array: *int arr[10000]*. In Version 2, utilize a linked list. Fill each structure with random numbers, then traverse the structures to measure and compare only the traversal times.

Question 5.

Explain the following code snippet i.e. add comments to each line of code.

```
uint32_t function_x(uint64_t i)
{
  i = i - ((i >> 1) & 0x5555555555555555);
  i = (i & 0x3333333333333333) + ((i >> 2) & 0x3333333333333333);
  i = (i + (i >> 4)) & 0x0f0f0f0f0f0f0f0f;
  i = i + (i >> 8);
  i = i + (i >> 16);
  i = i + (i >> 32);
  return (uint32_t)i;
}
```

`https://www.chessprogramming.org/Population_Count`

Question 6.

Compare the runtimes.

```cpp
// compile with -msse4.2
#include <iostream>
#include <vector>
#include <nmmintrin.h>
#include <chrono>
#include <random>

uint32_t function_x(uint32_t i) {
  // Fill this up from the previous question
}

uint32_t hw_popcount(uint32_t i) {
  return _mm_popcnt_u32(i);
}

int main() {
  std::vector<uint32_t> numbers(100000);
  std::mt19937 rng;
  rng.seed(std::random_device()());
  std::uniform_int_distribution<uint32_t> dist;

  for (auto& num : numbers) {
    num = dist(rng);
  }

  auto start = std::chrono::high_resolution_clock::now();
  for (uint32_t num : numbers) {
    popcount(num);
  }
  auto end = std::chrono::high_resolution_clock::now();
  std::chrono::duration<double, std::micro> custom_duration = end - start;
  std::cout << "function_x: " << custom_duration.count() << " microseconds\n";

  start = std::chrono::high_resolution_clock::now();
  for (uint32_t num : numbers) {
    hw_popcount(num);
  }
  end = std::chrono::high_resolution_clock::now();
  std::chrono::duration<double, std::micro> hw_duration = end - start;
  std::cout << "Hardware popcount duration: " << hw_duration.count() << " microseconds\n";
  return 0;
}
```