Vy-An Phan

26159888

cs195-15-aii

**Assignment 1**

1. Summation timing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sum, print | Sum, no print | Hardcoded, print | Hardcoded, no print |
| With O2 | 18976 | 2258 | 19282 | 2175 |
| Without O2 | 78731 | 78417 | 83374 | 80239 |

2. Pointer updating

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Print | Avg. cycles/step | Hardcoded, print | Avg. cycles/step |
| With O2 | 5355454 | 5.227 | 5333970 | 5.206 |
| Without O2 | 12185552 | 11.893 | 11989353 | 11.702 |

\*excluded "without printing" because then optimization causes it not to execute

a) This pointer chasing benchmark measures the time between requesting and obtaining a value from memory by

b) An int array of length N is \_\_\_ bytes in memory on this computer (x64)

Explain how this pointer chasing benchmark measures the time between

requesting and obtaining a value from memory (i.e. the memory latency).

(2 points)

\_ How many bytes in memory is an int array of length N on the computer

you are running on? (1 point)

\_ Generate a plot showing the relationship between the average number

of cycles to execute a step (vertical axis) and the size in bytes of the

array (horizontal axis). Vary the size of the array from 32 kilobytes to 8

megabytes. (32 points)

\_ Discuss the shape of the curve. In particular, explain what the sudden

slope changes in the curve correspond to. (4 points)

\_ In class we discussed how modern out-of-order processors can execute

many instructions in parallel through the use of pipelining. Thus even if

a single instruction takes t nanoseconds to complete, ten instructions will

likely take much less than 10t nanoseconds to complete. Does this affect

our benchmark? That is, if it takes t cycles to run c steps in our pointer

chasing benchmark, how can we be sure that a single step is actually

completing in t

c cycles and that the steps aren’t simply being pipelined?

(2 points)

\_ BONUS: We would ideally like the index, i, to eventually traverse through

the entire array. That is, we would like i to take on all the values between

0 and N eventually. However not every array A allows this. Write a

function that when given an array A, computes how many unique values

i actually takes on. Explain how the code works. (16 points)

5