Extra Credit - Speculation - A Puzzle

● Graded

Student

HARRY KIM

Total Points

35 / 40 pts

Question 1

Student's answer 35 / 40 pts

Step 1 - Your computer's details

✓ **+ 7 pts** Complete - screen shot, cores, memory, and cache information visible.

+ 5 pts Mostly complete - missing only a small detail. (Usually a screenshot or L1 cache performance)

+ 0 pts Partial credit - not available on extra credit

Step 2 - Execution times

✓ **+ 8 pts** Times are clearly visible in your report

+ 0 pts No partial credit if I cannot find the execution times.

Step 3 - Explanation

+ 25 pts You've likely got it, or you've convinced me of solid reasoning as to the cause, or your experiments are very thorough. (Note: Very few students actually gave reasoning I consider to be fully correct, but solid explanations and experiments were sufficient.)

✓ **+ 20 pts** You're on the right track or your experimentation is interesting, but I'd like to have seen more correct analysis or experimentation. I also give this mark if explanations are good, but the experiments are flawed.

+ 15 pts You've got an explanation or you've done an experiment or two, but the reasoning is incomplete or not on the right track, or a deeper explanation (not just longer) or more experimentation/data is needed.

+ 8 pts It's a good start, but more was needed to earn more extra credit.

+ 0 pts There's not enough correct understanding or reasoning here to earn extra credit -- partial credit is limited on extra credit.

1 It's a start, but it has a few misconceptions in it.

2 Good

Question assigned to the following page: 1

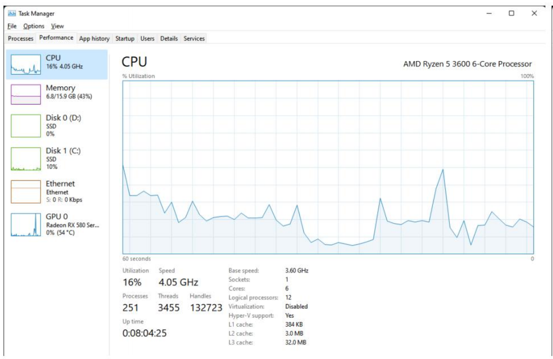
Harry Kim
u1226472

Extra Credit - Speculation - A Puzzle

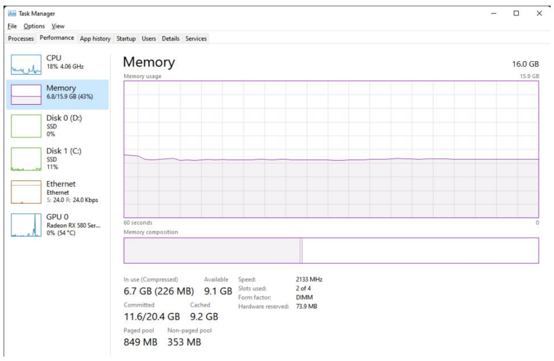
Step 1:

My pc:

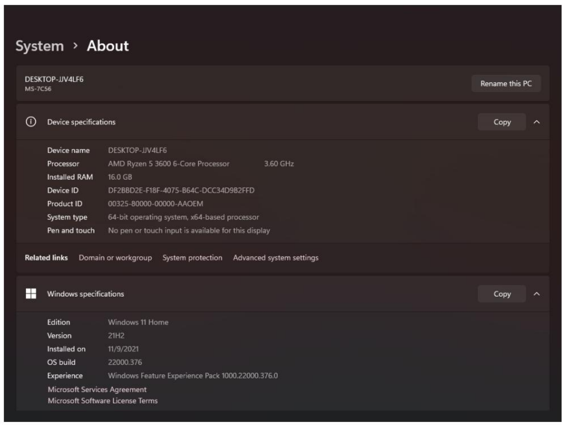
- How many cores does it have?
    - 6
- How much memory does it have, and what type and speed? (DDR3, etc.)
    - 16 GB, DDR4, 2133MHz
        - It didn't list the type of ram for some reason (maybe Windows 11), but the ram I bought for this PC is this: https://www.amazon.com/gp/product/B08B9KQDK5/ref=ppx_yo_dt_b_asin_title_o05_s02?ie=UTF8&psc=1
- What are the cache sizes and organization? Does it use separate data and instruction caches?
    - There are 3 different caches (L1, L2, L3) and they seem to use separate data and instructions.
- What OS/version are you running?
    - Windows 11 Home


Task Manager
File Options View
Processes Performance App history Startup Users Details Services
CPU 16% 4.05 GHz
Memory 6.8/15.9 GB (43%)
Disk 0 (D:) SSD 0%
Disk 1 (C:) SSD 10%
Ethernet Ethernet S: 0 R: 0 Kbps
GPU 0 Radeon RX 580 Ser... 0% (54 °C)
CPU
AMD Ryzen 5 3600 6-Core Processor
% Utilization
100%
60 seconds
0

Utilization	16%
Speed	4.05 GHz
Processes	251
Threads	3455
Handles	132723
Up time	0:08:04:25
Base speed:	3.60 GHz
Sockets:	1
Cores:	6
Logical processors:	12
Virtualization:	Disabled
Hyper-V support:	Yes
L1 cache:	384 KB
L2 cache:	3.0 MB
L3 cache:	32.0 MB



In use (Compressed)	6.7 GB (226 MB)
Available	9.1 GB
Committed	11.6/20.4 GB
Cached	9.2 GB
Paged pool	849 MB
Non-paged pool	353 MB
Speed:	2133 MHz
Slots used:	2 of 4
Form factor:	DIMM
Hardware reserved:	73.9 MB



Device name	DESKTOP-JJV4LF6
Processor	AMD Ryzen 5 3600 6-Core Processor 3.60 GHz
Installed RAM	16.0 GB
Device ID	DF2BBD2E-F18F-4075-B64C-DCC34D9B2FFD
Product ID	00325-80000-00000-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display
Edition	Windows 11 Home
Version	21H2
Installed on	11/9/2021
OS build	22000.376
Experience	Windows Feature Experience Pack 1000.22000.376.0

Harry Kim
u1226472

Step 2:

Option 1:
- The loop takes 5.6080058 nanoseconds per loop iteration on average.

Option 2:
- The loop takes 5.5801816 nanoseconds per loop iteration on average.

Option 3:
- The loop takes 5.5860453 nanoseconds per loop iteration on average.

Step 3:

1. In step #2, you made a change that alters the execution time of the program. Inside of the timed code, examine the loop and the 'if' statement inside of the loop. These both execute the same number of times (in both runs), but the pattern of 'if' statement executions varies. Describe this variation. How does the 'if' statement behave differently between the options/runs? (Examine the array contents to answer this.)
   - The if statement in the first option does a bunch of computations only on the first half of the array (which holds a billion values).
   - The if statement in the second option alternates between doing a bunch of computations and just setting an int to the index number for each element in the array.
   - The if statement in the first option either does a bunch of computations or just sets an int to the index number randomly for each element in the array.

2. In general, what aspect of speculation in the CPU is likely to account for the difference in execution times? Give a justification for your answer, and make sure it aligns with your answer to #1 above. (Note: I do not expect this to be related to caches.)
   - Most likely the type of computations the program is asking the computer to perform. In the if statement it does a variety of computations instead of just one like it is currently doing in the else statement. This would make sense as doing a variety of different computations all at once as opposed to alternating or switching randomly is likely to impact the performance of the computer.

3. Now, get specific. Give the different execution times from step #2, and label them (to make it clear which time is associated with which version of the code). Identify a possible cause for the difference in times -- explain why the longer times are slower or why the shorter times are faster. Make a claim, be specific, and try to find evidence for your claim.
   - The times I got for step #2 can be found above under **Step #2**.
   - In the if statement it does a variety of computations instead of just one like it is currently doing in the else statement. As shown with the times I got, it makes sense that option 1 would take the longest since we are doing a bunch of

Question assigned to the following page: 1

different computations with the first if statement all in a row for the first half of the billion array elements until we switch over to the else statement which only sets the variable x to the variable i. The computer's performance is impacted since it can't predict the frequency of the instructions since there are so many of them being cycled through in that single if statement. This would also explain why I got similar times for option 2 and 3 which are both faster than option 1. In options 2 and 3, the if and else statements are blended together giving the computer an instruction they can remember through doing the same thing frequently (the else statement) as opposed to doing all the different calculations for the first half of the billion elements in the array all at once. By this logic, it would make sense that option 2 would be the fastest since it always alternates between the if statement and the else statement, and my data shows that I was getting that option 2 was faster than option 3 on a consistent basis.