Systems Assignment 1

Marcus Acord-Serventi

1. What are similarities and differences between compilers and interpreters? What are pros and cons of each respectively?
   1. Compilers and interpreters both convert high level programming languages (such as Java, Python, or C++ etc.) to code that can be understood by computers (such as binary). Because coding in a high level language is much easier for people to understand, compilers and interpreters are used by people to interact with computers in a quicker and more meaningful way. Put another way, compilers and interpreters serve as a “translator” between people (who code) and computers. However, the way in which the work are very different. One key difference is that a compiler converts the entire program into code all at once, while an interpreter converts the code line by line. This makes an interpreter easier to debug and go through code to find errors quicker as any errors are generated in real-time as the program is being coded, but also means that a compiler can execute the code quicker when run. While programming in an interpreter, it can display errors as you program. A compiler can only display all errors when it is compiled. While a compiler can execute code quicker, it typically uses more space in memory to generate an intermediate code, while an interpreter does not generate any intermediate code. One big negative of compilers is that you can’t make changes to the program without going back to the source code and recompiling (which is an issue in large programs that take a long time to compile). One big negative of interpreters is that interpreted programs can only be run on computers that have the interpreter language installed.
   2. References:
      1. <https://www.geeksforgeeks.org/difference-between-compiler-and-interpreter/>
      2. <https://www.businessinsider.in/difference-between-compiler-and-interpreter/articleshow/69523408.cms>
      3. <https://techdifferences.com/difference-between-compiler-and-interpreter.html>
2. What is caching and what is locality?
   1. Caching is also known as locality of reference. Caching basically refers to a technique to improve a computer system’s performance. Caching allows the computer to store items used frequently in faster forms of memory (so storing something in the computer’s RAM cache instead of on the hard drive, for instance). Caching serves as a temporary storage area for frequently used instructions. Computers tend to access instructions whose addresses are stored in faster memory first and instructions whose addresses are near one another both spatially and temporally (locality). There are two types of locality: Temporal locality refers to an object stored in local cache which may be needed soon or once again. One way to trigger temporal locality is to repeatedly access a memory location multiple times. Spatial locality refers to an object that is stored near to another object, which the computer may need to access soon in the future. For example, a CPU will first check to see if an object is stored in cache memory as it is very fast. If it is, the CPU will pull it from the cache in order to save time (referred to as a cache hit). If it’s not stored there, the CPU will then check system RAM, which is slower (referred to as a cache miss). If it is being accessed multiple times, that object may move from the RAM to the cache, in order to improve the CPU access times. However, the first time an object is accessed, it will not be in the cache, so the computer will need to pull it from slower memory types.
   2. References:
      1. <https://www.geeksforgeeks.org/locality-of-reference-and-cache-operation-in-cache-memory/>
      2. <https://www.d.umn.edu/~gshute/arch/caching.xhtml>
3. Why can good locality help improve caching performance?
   1. Locality is an important topic in computer science and software development. Programs with high levels of locality are more efficient and can run quicker. If a program has high locality, it will also have a high cache hit-to-miss ratio. If good locality exists, then the most frequently used instructions are stored in faster caches, which helps improve the overall performance by allowing the CPU to access the instructions stored in faster caches. If a cache miss occurs, then the computer must go further in the memory, which is much slower, in order to access the instructions. Good localiuty allows the most frequently used inbstructions to be stored in the faster memory caches and thus allow the CPU (and programs) to run faster. Two things that programmers can do to increase locality are to access frequent instructions often (thus storing it temporally) and focusing on the inner loops to minimize cache misses.
   2. References:
      1. <https://www.cs.cornell.edu/courses/cs3110/2012sp/lectures/lec25-locality/lec25.html#:~:text=On%20a%20cache%20miss%2C%20the,is%20filled%20from%20main%20memory.&text=Therefore%2C%20caches%20improve%20performance%20when,memory%20locations%20near%20previous%20requests>.
      2. <https://www.geeksforgeeks.org/computer-organization-locality-and-cache-friendly-code/>