Jonah R. Smith

PSID: 1569818

04/23/2019

MECE 5397: Assignment 8AA

1. A hard drive has the following properties:

Rotational Speed = 10,000 rpm

Average Seek Time = 4ms

Average Sectors per Track = 800 = Sectors per Rotation

Bytes/Sector = 1024 = 1 KB

**Determine the total access time to read 32 KB of data from one surface of a hard drive:**

Time to Read = (Seek Time) + [Bytes]/[(Sectors per Rotation)\*(Rotations per Second) \* (Bytes/Sector)]

**Time = 4.24 ms**

**2. Compare 70% cache hit rate with 31% cache miss rate**

I’ll assume the same cache hit time and miss penalty used in the Memory Hierarchy lecture notes:

Cache hit time = 1 cycle

Miss Penalty = 100 cycle

Then the average access time can be found by: Cycles = Cache Hit Time + (Miss Chance) \* Miss Penalty

70% Hit Rate: Cycles = Cache Hit Time + 0.3\*Miss Penalty

31% Miss Rate: Cycles = Cache Hit Time + 0.31\*Miss Penalty

This can give us the relationship of how much slower the 31% miss penalty is than the 70% hit rate:

(Cycles for 31% Miss Rate) - (Cycles for 70% Hit Rate) = 0.01\*Miss Penalty

In our example, if the miss penalty is 100 cycles, that means we take 1 extra cycle for the 31% Miss Rate than if we had a 70% Hit Rate. Numerically, this would solve to:

70% Hit Rate: Takes 31 cycles

31% Miss Rate: Takes 32 cycles

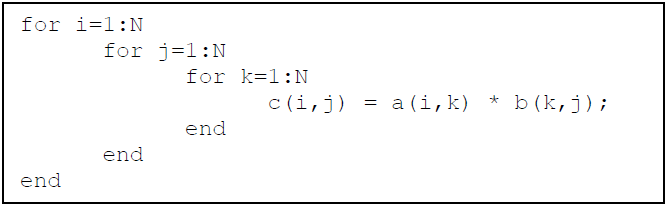
That is, having a 70% hit rate vs. a 31% miss rate, using our assumed values for cache hi time and miss penalty, would have a 3.125% decrease in required cycles.

**3. Fortran programming language uses column major ordering to store arrays. If I have a large 2D array in Fortran with individual dimensions of array exceeding the cache memory size, what would be the cache miss rate when traversing along a column and when traversing along a row (assume single cache memory level)?**

Again, we’re assuming a 1 cycle hit rate and 100 cycle miss penalty.

Well idk, I’ll do this problem later, and hopefully not submit it with this text still in here.

**4. Optimize the following piece of code:**



I shall assume that we are using MATLAB, since the indices start at 1 and not 0 for these arrays.

Notice that c(i,j) gets overwritten for every i,j by the value at k=N in the original code.

**Optimized Code:**

for j=1:N

for i=1:N

c(i,j) = a(i,N) \* b(N,j);

end

end