- CPSC 275: Introduction to Computer Systems

CPSC 275: Introduction to Computer Systems

Fall 2025

- Syllabus
- Schedule
- Resources
- Upload
- Solution

Homework 28

NOTE: You are not required to hand in the following exercises, but you are strongly encouraged to complete them to strengthen your understanding of the concepts covered in class.

1. Consider a function that computes the dot product of two vectors:

```
float dotprod(float x[8], float y[8])
{
   float sum = 0.0;
   int i;

   for (i = 0; i < 8; i++)
      sum += x[i] * y[i];
   return sum;
}
```

This function has good spatial locality with respect to `x` and `y`, and so we might expect it to enjoy a good number of cache hits. Unfortunately, this is not always true.

   A. Suppose that `floats` are 4 bytes, that `x` is loaded into the 32 bytes of contiguous memory starting at address 0, and that `y` starts immediately after `x` at address 32. For simplicity, suppose that a block is 16 bytes (big enough to hold four `floats`) and that the direct-mapped cache consists of two sets, for a total cache size of 32 bytes. We will assume that the variable `sum` is actually stored in a CPU register and thus does not require a memory reference. Show that each `x[i]` and `y[i]` will map to the identical cache set.
   B. Now, instead of defining `x` to be float `x[8]`, we define it to be `float x[12]`. Assuming `y` starts immediately after `x` in memory. Show that with the "padding" at the end of `x`, `x[i]` and `y[i]` now map to different sets.
   C. What fraction of the total references to `x` and `y` will be hits once we have padded array `x`?

2. A processor has a cache with a hit time of 1 ns and a miss penalty of 50 ns. If the cache hit rate is 90%, compute the average memory access time.

3. Two systems have the following characteristics:

System	Hit Time	Miss Penalty	Hit Rate
A	1 ns	40 ns	95%
B	2 ns	25 ns	98%

Which system provides a faster average memory access time?

4. A processor uses two cache levels (L1 and L2) before accessing main memory:
   - L1 hit time: 1 ns
   - L1 miss rate: 10%
   - L2 hit time: 5 ns
   - L2 miss rate: 40%
   - Main memory access time: 100 ns

   Compute the overall average memory access time. Hint: When an L1 cache miss occurs, the processor looks for the data in the next cache level (L2). If it is not found there either, the processor fetches the data from main memory.

5. **Block Matrix Multiplication**. *Blocking* a matrix works by partitioning the matrices into submatrices and then exploiting the mathematical fact that these submatrices can be manipulated just like scalars. Modify your `mmul_ijk` function to implement block matrix multiplication. Call this new function `bmmul`:

   ```
   void bmmul(double *A, double *B, double *C, int N, int b);
   ```

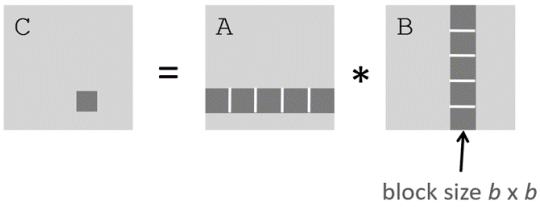
   where *b* is the block size.

   Using this function, write a C program (`block_matmul.c`) which allocates three *N*-by-*N* arrays of `doubles`, *A*, *B*, and *C*, and computes *C* = *A* * *B* blockwise. The values of *N* and *b*, the block size, must be passed to the program at the command line. Initialize the matrices using the function in Laboratory 11. Run your program with:

   ```
   $ time ./block_matmul N b
   ```

   where *N* = 2048 and *b* = 32, 64, 128, 256, 512. Measure the running time of your program. Which of the block sizes gives the fastest running time?

   The key idea of blocking is that it loads a block of the matrix *B* into the cache, uses it up, and then discards it. References to *A* enjoy good spatial locality because each row of the current block is accessed with a stride of 1. There is also a good temporal locality because the entire row of the current block is referenced *b* times in succession. References to *B* enjoy good temporal locality because the entire *b*-by-*b* block is accessed *n* times in succession. Finally, the references to *C* have good spatial locality because each element of the row of the current block is written in succession.



- **Welcome: Sean**
  - LogOut

Trinity College
HARTFORD CONNECTICUT