ETH login ID:
(Please print in capital letters)

Full name:

263-2300: How to Write Fast Numerical Code

ETH Computer Science, Spring 2015
Midterm Exam
Wednesday, April 15, 2015

Instructions

- Make sure that your exam is not missing any sheets, then write your full name and login ID on the front.
- No extra sheets are allowed.
- The exam has a maximum score of 100 points.
- No books, notes, calculators, laptops, cell phones, or other electronic devices are allowed.

Problem 1 (16 = 1+1+5+4+5)	
Problem 2 (18 = 3+15)	
Problem 3 (18 = 2+6+10)	
Problem 4 (12 = 2 + 2 + 2 + 2 + 2 + 2)	
Problem 5 (12)	
Problem 6 (12 = 6 + 6)	
Problem 7 (12 = 2 + 5 + 5)	
Total (100)	

Problem 1 $(16 = 1 + 1 + 5 + 4 + 5)$

We consider a 128 byte data cache that is 2-way associative and can hold 4 doubles in every cache line. A double is assumed to require 8 bytes.

For the below `C` code we assume a cold cache. Further, we consider an array `A` of 32 doubles that is cache aligned (that is, `A[0]` is loaded into the first slot of a cache line in the first set). All other variables are held in registers. The code is parameterized by positive integers `m` and `n` that satisfy `m * n = 32` (i.e., if you know one you know the other).

```c
int i, j;
double A[32], t = 0;
for(i = 0; i < m; i++)
  for(j = 0; j < n; j++)
    t += A[j * m + i];
```

Answer the following:

1. How many doubles can the cache hold? **16**
2. How many sets does the cache have? **2**
3. For `m = 1`:
   (a) Determine the miss rate. $\mathbf{\frac{1}{4}}$
   (b) What kind of misses occur? **Compulsory**.
   (c) What kind of locality does the code have with respect to accesses of `A` and this cache? **Spatial locality**.
4. For `m = 2`:
   (a) Determine the miss rate. $\mathbf{\frac{1}{2}}$
   (b) What kind of misses occur? **Compulsory & conflict**.
5. For `m = 16`:
   (a) Determine the miss rate. $\mathbf{\frac{1}{4}}$
   (b) What kind of misses occur? **Compulsory**.
   (c) What kind of locality does the code have with respect to accesses of `A` and this cache? **Spatial locality**.

Problem 2 (18 = 3 + 15 points)

Consider the following code, which computes the Cholesky decomposition of a hermitian positive definite matrix A ($N \times N$).

```
void cholesky(float **A, float **L, int N){
  int i,j,k;
  float temp;

  for(j = 0; j < N; j++){
    temp = A[j][j];
    for(k = 0; k < j; k++){
      temp = temp - L[j][k]*L[j][k];
    }
    L[j][j] = sqrt(temp);
    for(i = j+1; i < N; i++){
      temp = A[i][j];
      for(k = 0; k < j; k++){
        temp = temp - L[i][k]*L[j][k];
      }
      L[i][j] = temp/L[j][j];
    }
  }
}
```

1. Define a detailed floating point cost measure $C(N)$ for the function `cholesky`. Ignore integer operations.

   Solution:

   $C(N) = \{add(N), mul(N), div(N), sqrt(N)\}$

2. Compute the cost $C(N)$ as just defined.

   Solution:

$$\begin{aligned} add(N) &= \frac{N^3}{6} + \mathcal{O}(N^2) \\ mul(N) &= \frac{N^3}{6} + \mathcal{O}(N^2) \\ div(N) &= \frac{N^2}{2} + \mathcal{O}(N) \\ sqrt(N) &= N \end{aligned}$$

Notes: Lower-order terms (and only those) may be expressed using big-O notation. This means: as the final result something like $3n + O(\log(n))$ would be ok but $O(n)$ is not.

The following formulas may be helpful:

- $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} = \frac{n^2}{2} + O(n)$
- $\sum_{i=0}^{n-1} i^2 = \frac{(n-1)n(2n-1)}{6} = \frac{n^3}{3} + O(n^2)$

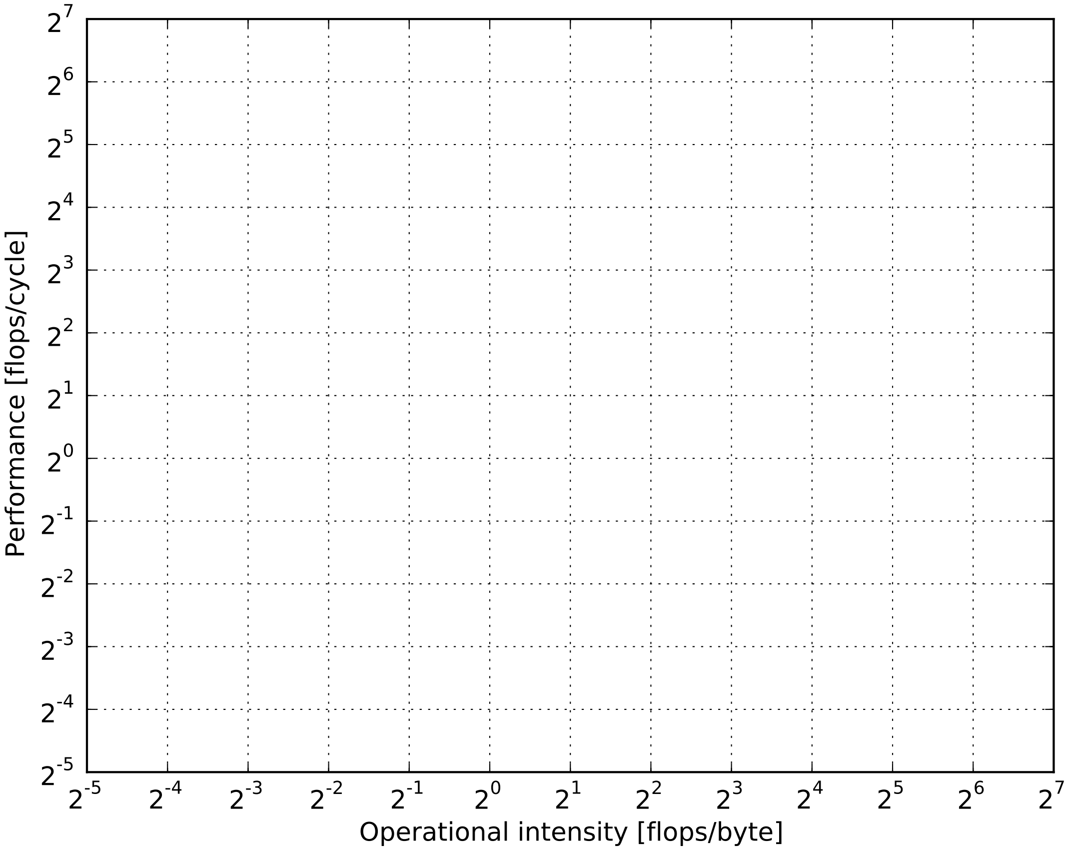
Problem 3 (18 = 2 + 6 + 10 points)

Assume you are using a system with the following features:

- A CPU that can issue 2 single precision multiplications and 2 single precision additions/subtractions per cycle.
- The interconnection between CPU and main memory (size 16 GB) has a maximal bandwidth of 8 bytes/cycle.
- The last level cache is write-allocate/write-back, direct mapped, has size 8 MB and block size of 64 bytes.

Answer the following two questions:

1. Draw the roofline plot for this system:



2. Consider the following code where all the entries in matrix m are initialized between 0 and 1:

```
1 void compute(float m[64]) {
2     int i;
3
4     for(i = 1; i < 64; i++) {
5         m[i-1] = (1 - m[i-1]) * m[i];
6         m[i] = (1 - m[i]) * m[i-1];
7     }
8 }
```

Assume a cold cache, that the operators are left associative (expressions are evaluated from left to right), and that a float takes 4 bytes. Now compute,

(a) The operational intensity of this code (ignore write-backs).

Solution:

There are 63 iterations, each iteration performs 4 flops, so $W(N) = 4 * 63 = 252$ flops. All entries in matrix are read once so $Q(N) = 64 * 4 = 256$ bytes are loaded into cache. This gives

$$I(N) = 252/256 \approx 1f/b$$

(b) An upper bound (as tight as possible) for performance on the specified system.

You are allowed to make minor approximations. Show your work.

Solution:

Since $I(N) > \frac{1}{2}$, the code is compute bound. However it is not possible to achieve peak performance of 4 f/c due to dependencies in the computation. For i-th iteration, the assignment to $m[i-1]$ at line 5 depends (except when i = 1) on $m[i-1]$ which was computed at line 6 in $(i-1)$-th iteration. Similarly, assignment to $m[i]$ at line 6 depends on $m[i-1]$ computed previously at line 5. Because of these dependencies, $(i+1)$-th iteration cannot be interleaved with i-th iteration. For i-th iteration, two substitutions can be issued in parallel. The multiplication at line 5 can only be issued after substitution at line 5 is complete. Similarly, multiplication at line 6 can only be issued after previous multiplication at line 5 is completed. The number of cycles required for this can be either 3 or 1.5 depending on the processor. We accept both solutions, thus the upper bound for performance is $\frac{4}{3}$ $(\frac{4}{1.5})$ f/c.

3. Consider the following code where *alpha* is initialized between 0 and 1:

```
void  compute(float A[4096][4096], float alpha) {
    int i,j;

    for(i = 0; i < 4096; i++)
        for(j=0; j < 4096; j++)
            A[i][j] = alpha*A[i][i] + (1 - alpha)*A[j][j];
}
```

Assume a cold cache, that the operators are left associative (expressions are evaluated from left to right), and that a float takes 4 bytes. Now compute,

(a) The operational intensity of this code (ignore write-backs).

Solution:

The size of the matrix is $2^{12} \times 2^{12} \times 4 = 2^{26} = 64MB$. Since the size of cache is $8MB$ only, it can accommodate only $\frac{1}{8}$-th of the matrix. The diagonal operands $A[j][j]$ ($A[i][i]$) are accessed in stride of 4097. Since the cache is direct mapped, $\approx \frac{2^{17}}{2^8} = 512$ blocks are used for $A[j][j]$ ($A[i][i]$). This results in cache miss for each access to $A[j][j]$. Access to $A[i][i]$ will be cache miss after ≈ 512 iterations of j-loop. Note that $A[j][j]$ ($A[i][i]$) can also be replaced by $A[i][j]$ (vice versa). For simplicity, we ignore misses for $A[i][j]$, $A[i][i]$, thus at least 64 bytes needs to be loaded at every iteration for performing 4 flops. Thus,

$$I(N) \approx \frac{4}{64} = \frac{1}{16} f/b$$

(b) An upper bound (as tight as possible) for performance on the specified system.

You are allowed to make minor approximations. Show your work.

Solution:

Since $I(N) < \frac{1}{2}$, the computation is memory bound and the peak performance has upper bound of $\frac{1}{16} \times \beta = \frac{1}{16} \times 8 = \frac{1}{2}$ f/c.

Problem 4 ($12 = 2 + 2 + 2 + 2 + 2 + 2$)

Mark the following statements as true (T) or false (F). Explanations are not needed. Wrong answers give negative points but you cannot get less than 0 points for this problem. You can leave questions unanswered.

- ☐ Assume a program runs N many floating point adds and N many floating point mults and that the gaps for the two instructions are respectively g_1 and g_2 cycles/issue. Then assuming a warm cache scenario where the data set fits in cache and that accesses to the cache have a negligible cost the achievable peak performance can always be estimated as $\pi = \frac{1}{g_1} + \frac{1}{g_2}$ flops/cycle.
- ☐ A direct mapped cache with parameters $(\text{number of sets}, \text{associativity}, \text{block size}) = (S, 1, B)$ always produces twice as many conflict misses as a 2-way set associative cache with parameters $(S/2, 2, B)$.
- ☐ Data prefetching can increase operational intensity.
- ☐ Every TLB miss will also cause a cache miss.
- ☐ Every cache miss will also cause a TLB miss.
- ☐ If two algorithms solve the same problem in the same time, they have the same performance.

Solution: All statements are false.

Problem 5 (12)

Associative caches were designed to reduce conflict misses. However, increasing associativity (while maintaining the cache size) does not guarantee to achieve this in all cases. Consider a cache C_1 with $(\text{number of sets}, \text{associativity}, \text{block size}) = (S, 1, 8)$, i.e., the block size is one double. A second cache C_2 has the same size with parameters $(S/2, 2, 8)$. Both have LRU replacement and are empty.

Consider an array `a` of $2S$ doubles that is cache-aligned (i.e., `a[0]` is mapped to the first block of either cache). Provide an access sequence (of a length that you can choose) to this array such that on C_1 fewer misses occur than on C_2.

Hint: It helps to draw the caches.

Solution: Two possible sequences are $0; \frac{S}{2}; \frac{3S}{2}; 0$ and $0; \frac{S}{2}; S; 0; \frac{S}{2}$.

Problem 6 (12 = 6 + 6)

In this problem we consider a computer with a fully associative cache of size γ (measured in doubles; one double is 8 bytes) and three algorithms for which the flop count W and lower bounds for the minimal memory traffic Q (in doubles) are known:

MMM: Matrix multiplication of $N \times N$ matrices with $W(N) = 2N^3$, $Q(N) \geq \frac{N^3}{2\sqrt{2\gamma}}$ doubles.

FFT: A variant of an N-point fast Fourier transform (N a power of 2) with $W(N) = 2N \log_2 N$, $Q(N) \geq \frac{2N \log_2 N}{\log_2 \gamma}$ doubles.

CG: A conjugate gradient method that solves a system of linear equations over a two-dimensional grid of size $N \times N$ in T iterations with $W(N,T) = 20N^2T$, $Q(N,T) \geq 6N^2T$ doubles.

1. Compute for all three algorithms upper bounds on the operational intensity $I(N)$ or $I(N,T)$ (unit: flops/byte).

   Solution:

   MMM:

$$I(N) \leq \frac{W(N)}{8Q(N)} = 0.5\sqrt{2\gamma} \text{ flops/byte} = \hat{I}.$$

   FFT:

$$I(N) \leq \frac{W(N)}{8Q(N)} = \frac{\log_2 \gamma}{8} \text{ flops/byte} = \hat{I}.$$

   CG:

$$I(N,T) \leq \frac{W(N,T)}{8Q(N,T)} = \frac{5}{12} \text{ flops/byte} = \hat{I}.$$

2. We continue with assume a system that the computer has a peak performance of $\pi = 4$ flops/cycle and a memory bandwidth of $\beta = 8$ bytes per cycle. Determine, separately for all three cases, the cache sizes γ (again measured in doubles, and a power of 2) for which the computation is memory bound:

   Solution: A computation is memory bound if and only if $I(N) < \frac{\pi}{\beta}$.

   MMM:

$$\hat{I} < \frac{\pi}{\beta} = \frac{1}{2} \text{ flops/byte} \iff \gamma < 0.5 \text{ double}$$

   This means that the only way to ensure that MMM is always memory bound is to eliminate the cache.

   FFT:

$$\hat{I} < \frac{\pi}{\beta} = \frac{1}{2} \text{ flops/byte} \iff \gamma < 2^4 \text{ double}$$

   CG:

$$\hat{I} < \frac{\pi}{\beta} = \frac{1}{2} \text{ flops/byte} \implies \text{The computation is always memory bound.}$$

Problem 7 (12 = 2 + 5 + 5)

Assume a CPU with the following parameters:

- Frequency $f = 5$ GHz
- One cache (L1) with instant access (i.e., no latency, infinite bandwidth)
- Main memory with access bandwidth of β doubles/cycle and a latency of $\ell_{\mathrm{RAM}} = 100$ ns (time needed to have a double available for computation)
- Peak performance of 2 flops/cycle

1. Determine β (make it low enough) such that every L1 miss contributes exactly ℓ_{RAM} to the total execution time.

   Solution: We need to enforce a bandwidth capable of transferring a double every ℓ_{RAM}:

$$\beta = \frac{1}{f \cdot \ell_{\mathrm{RAM}}} = \frac{1}{500} \text{ doubles/cycle.}$$

2. Now we execute a program P on this CPU with $W(N) = 20N^2$ flops and accesses $A(N) = N^2$ doubles. If all accesses did hit the cache, P would run at the CPU's peak. However, the hit rate is 96%. What is the runtime of P (using β from the previous part)? Assume that the computation and memory accesses do not overlap.

   Solution:

$$T_1 = \frac{W(N)}{2} + 0.04 \cdot A(N) \cdot f \cdot \ell_{\mathrm{RAM}} \text{ cycles} = 10N^2 + 20N^2 = 30N^2 \text{ cycles .}$$

3. Assume the introduction of a second cache (L2) with access bandwidth of β (same as main memory) and latency 10 ns. The miss rate for this cache for program P is 0.5%. What is the speed-up obtained for P by introducing this cache?

   Solution: We provide two alternatives both considered acceptable:

   (a) Assuming that both L2's and RAM's bandwidth are the same we don't notice any significant speed-up:

$$\begin{aligned} T_2 &= \frac{W(N)}{2} + f \cdot \ell_{\mathrm{L2}} + ((0.04 - 0.005) \cdot A(N) - 1) \cdot f \cdot \ell_{\mathrm{RAM}} + 0.005 \cdot A(N) \cdot f \cdot \ell_{\mathrm{RAM}} \\ &\approx \frac{W(N)}{2} + 0.04 \cdot A(N) \cdot f \cdot \ell_{\mathrm{RAM}} \text{ cycles} = 10N^2 + 20N^2 = 30N^2 \text{ cycles} . \end{aligned}$$

   (b) Assuming that a condition similar to the one in 1 holds for the L2's bandwidth (i.e., $\beta_{\mathrm{L2}} = 1/\ell_{\mathrm{L2}}$) we obtain the following:

$$\begin{aligned} T_2 &= \frac{W(N)}{2} + (0.04 - 0.005) \cdot A(N) \cdot f \cdot \ell_{\mathrm{L2}} + 0.005 \cdot A(N) \cdot f \cdot \ell_{\mathrm{RAM}} \\ &= 10N^2 + 1.75N^2 + 2.5N^2 = 14.25N^2 \text{ cycles.} \end{aligned}$$

   Finally, we conclude that the speed-up compared to the single-cache system is

$$\frac{T_1}{T_2} = \frac{30N^2}{14.25N^2} = 2.1.$$