CS 382
Fall 2024

Computer Architecture
and Organization

Homework 4

Requirement (READ FIRST!)

You have to **type** the solutions. **Handwritten homework will not be graded and will receive zero credit.** You can annotate on this document directly (you might need to know how to insert a picture into a PDF file); or you can submit a separate PDF, but with solutions clearly marked with question numbers.

1 (20 points)

Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3.2 GHz clock rate and a CPI of 1.5. P2 has a 2.0 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.3.

1.1 (7 points)

Calculate the performance of each processor expressed in instructions per second.

1.2 (7 points)

If each of these processors executes a program in 10 seconds, calculate the number of cycles and the number of instructions.

1.3 (6 points)

We are trying to reduce the execution time of P2 by 30%, but this leads to an increase of 20% in the CPI. What should the clock rate be to obtain this reduction in execution time?

2 (20 points)

Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3 for the corresponding classes, and P2 with a clock rate of 3 GHz and CPIs of 2, 2, 2, and 2. Given a program with a dynamic instruction count of 1.0e6 instructions divided into classes as follows: 30% class A, 20% class B, 30% class C, and 20% class D:

2.1 (7 points)

What is the total CPI for each implementation?

2.2 (7 points)

Calculate the clock cycles required in both cases.

2.3 (6 points)

Which is faster: P1 or P2?

3 (20 points)

In this exercise we look at memory locality properties of matrix computation. The following code is written in C, where *elements within the same row of a matrix are stored contiguously.* You can assume that undefined symbols are primitive types, *i.e.,* `int`, `double` *etc.*

```
for( int j = 1; j < n-1; j++) {
    for(int i = 1; i < m-1; i++) {
        Anew[j][i] = 0.25 * (A[j][i+1] + A[j][i-1] + A[j-1][i] + A[j+1][i]);
        err = max(err, fabs(Anew[j][i] - A[j][i]);
    }
}
```

3.1 (7 points)

Which variables exhibit temporal locality?

3.2 (7 points)

Which variables exhibit spatial locality? Locality is affected by both the reference order and data layout.

3.3 (6 points)

Would the program be slower or faster if the outer loop iterated over `i` and the inner loop iterated over `j`? Why?

4 (20 points)

Consider a **byte addressing** architecture with 64-bit memory addresses.

4.1 (5 points)

Which bits of the address would be used in the tag, index and offset in a direct-mapped cache with 512 1-word blocks?

4.2 (5 points)

Which bits of the address would be used in the tag, index and offset in a direct-mapped cache with 64 8-word blocks?

4.3 (5 points)

What is the ratio of bits used for storing data to total bits stored in the cache in each of the above cases?

4.4 (5 points)

Which bits of the address would be used in the tag, index and offset in a two-way set associative cache with 1-word blocks and a total capacity of 512 words.

5 (20 points)

Given a byte-addressed memory with 12-bit addresses, and an attached four-way set associate cache with a total of 16 one-word blocks, make a table showing the final state of the cache after accessing the following memory addresses. The table should include the tags and data that will be stored in cache at the end of all insertions. Data should be shown using the corresponding addresses in main memory. Use the least-recently-used rule to evict from cache as needed.

The sequence of addresses accessed is:

`0x143, 0xc4a, 0x22b, 0x42f, 0x492, 0x2a2, 0x3ba, 0xb2d`