IB Computer Design - Supervision 4

Nandor Licker <nl364@cl.cam.ac.uk>

Due noon before the day of supervision

Q1 2018 Paper 5 Question 2

Q2 2017 Paper 5 Question 2

Q3 2017 Paper 5 Question 3

Q4 2014 Paper 5 Question 3

Q5 You are given 2 functions implementing matrix multiplication. `T` is a hardcoded tile size, while `N` is the size of the matrices, assumed to always be a multiple of `T`.

1. ```
   void naive_multiply(const float *a, const float *b, float *c) {
     for (int i = 0; i < N; ++i) {
       for (int j = 0; j < N; ++j) {
         float r = 0.0;
         for (int k = 0; k < N; ++k) {
           r += a[i * N + k] * b[k * N + j];
         }
         c[i * N + j] = r;
       }
     }
   }
   ```
2. ```
   void tiled_multiply(const float *a, const float *b, float *c) {
     for (int bj = 0; bj < N; bj += T) {
       for (int bk = 0; bk < N; bk += T) {
         for (int i = 0; i < N; ++i) {
           for (int j = bj; j < bj + T; ++j) {
             float r = 0.0;
             for (int k = bk; k < bk + T; ++k) {
               r += a[i * N + k] * b[k * N + j];
             }
             c[i * N + j] += r;
           }
         }
       }
     }
   }
   ```

Consider matrices of size `N` = 128. Assume that floats are 4 bytes wide and the data cache of the processor has 256 lines of 64 bytes each. Compute the number of cache hits and cache misses for both algorithms using the following cache layouts:

- Direct Mapped
- 4-way set associative, LRU
- 8-way set associative, LRU
- Fully Associative Cache, Round Robin

You may assume that the matrices are laid out consecutively in memory. Consider tile sizes of 4, 8 and 16. Count how many dirty lines must be written back to main memory. You can compute the numbers by creating a cache simulator in a programming language of your choice, but it would be a good idea to estimate at least one set-associative example by hand. You should create a plot of the hit and miss rates by cache type and tile size.

Q6 Assuming the existence of a test-and-set primitive, implement the following spinlocks in C:

- test-and-set (TAS)
- test-and-test-and-set (TATAS)

Assuming 4 threads running on 4 different cores, show the MESI bus and cache interactions which happen with and without contention. Clearly describe your assumptions about the TAS primitive.

Q7 Consider the following C code:

```
int f(int a, int b) {
  if (a & 1) {
    return g(a);
  } else {
    return g(a + b);
  }
}
int g(int x) {
  return x + 5;
}
int q(int n, int b, int *dst) {
  int arr[n];
  for (int i = 0; i < n; ++i) {
    arr[i] = f(i, b);
  }
  for (int j = 0; j < n; ++j) {
    dst[j] = arr[n - j - 1];
  }
}
```

Manually compile this code to `RISC-V`, `x86_64`, `AArch64`, `ARMv7` and `MIPS`.

Q8 Consider the following fragments of C code:

- ```
  for (int i = 0; i < n; ++i) {
    if (i < 100) {
      f(i);
    } else {
      g(i);
    }
  }
  ```
- ```
  for (int i = 0; i < n; ++i) {
    if (i & 1) {
      f(i);
    } else {
      g(i);
    }
  }
  ```

Show how a branch predictor relying on 2-bit saturating counters would behave in both scenarios. How do prediction rates improve if a branch history buffer is added?