6.172 Quiz 1 Review

October 4, 2011
1. Administrivia

2. Performance Analysis
   - Basic Performance Engineering
   - C to Assembler
   - Bit Hacks

3. Parallelism
   - Multicore Programming
80 minutes, during lecture, Thursday, October 6.

**One hand-written** double-sided crib sheet (8.5"x11")

Cover materials until September 30

- Lecture Slides
- Lecture Notes
- Recitations
- Projects
Outline

1. Administrivia

2. Performance Analysis
   - Basic Performance Engineering
   - C to Assembler
   - Bit Hacks

3. Parallelism
   - Multicore Programming
- Cycles
- IPC
- Cache-references
- Cache-misses
- Branch-misses
Modifying Data

- Caching
  - Pre-Computation
  - Lazy Evaluation
- Packing/Compression
- Single instruction, multiple data (SIMD)
- Exploit Algebraic Identities
- Loop Invariant Code Motion
- Sentinel Loop Exit Test
- Loop Elimination by Unrolling
- Loop Fusion
- Eliminate Wasted Iterations
- Reordering Tests
- Inline Small Functions
- Common Subexpression Elimination
- Parallelism
/* Return 0 if the first n bytes of s1 and s2 are equal, and return 1 otherwise. */

int my_memcmp(char *a, char *b, size_t n) {
    for (size_t i = 0; i < n; i++)
        if (a[i] != b[i])
            return 1;
    return 0;
}
int my_memcmp_opt1(char *a, char *b, size_t n) {
  uint64_t *aw = (uint64_t *) a;
  uint64_t *bw = (uint64_t *) b;
  size_t iw = 0;
  for (; iw < n / sizeof(uint64_t); iw++)
    if (aw[iw] != bw[iw])
      return 1;
  size_t i = iw * sizeof(uint64_t);
  for (; i < n; i++)
    if (a[i] != b[i])
      return 1;
  return 0;
}
int my_memcmp_opt2(char *a, char *b, size_t n) {
    if (n == 0)
        return 0;
    size_t i;
    char oldanm1 = a[n-1];
    a[n-1] = !b[n-1];
    for (i = 0;; i++) {
        if (a[i] != b[i])
            break;
    }
    a[n-1] = oldanm1;
    return a[i] != b[i];
}
1. Administrivia

2. Performance Analysis
   - Basic Performance Engineering
   - C to Assembler
   - Bit Hacks

3. Parallelism
   - Multicore Programming
You should be able to read assembly code.

- Identify loops: jxx, cmp
- Identify function call: call, ret
- Arithmetic:
  - add, sub, inc, dec, imul, idiv
  - and, or, xor
  - shl, shr
Which one is faster?

cmpl %edi, %esi
movl %edi, %eax
cmovle %esi, %eax

cmpl %esi, %edi
setl %al
movzbl %al, %eax
negl %eax
movl %esi, %edx
xorl %edi, %edx
andl %edx, %eax
xorl %esi, %eax
1. Administrivia

2. Performance Analysis
   - Basic Performance Engineering
   - C to Assembler
   - Bit Hacks

3. Parallelism
   - Multicore Programming
No-Temp Swap

tmp = x;
\( x = y; \)
\( y = \text{tmp}; \)
\( x = x \text{ ^ } y; \)
\( y = x \text{ ^ } y; \)
\( x = x \text{ ^ } y; \)
Minimum of Two Integers

\[ r = (x < y) \ ? \ x : y; \]

\[ r = y \ ^ { ^ \ (x \ ^ { ^ y} \ & \ \neg(x < y))}; \]
More...

- Modular Addition
- Round up to a Power of 2
- Find Index of Least-Significant 1
- Population Count
  - Repeatedly Eliminate the Least-Significant 1
  - Table look-up
  - Parallel Divide-and-Conquer
- Queens Problem
Outline

1. Administrivia

2. Performance Analysis
   - Basic Performance Engineering
   - C to Assembler
   - Bit Hacks

3. Parallelism
   - Multicore Programming
MSI Protocol

- Each cache has a state:
  - **M**: cache block has been modified. No other caches contain this block in M or S states.
  - **S**: other caches may be sharing this block.
  - **I**: cache block is invalid (same as not there).

- Before a cache modifies a location, the hardware first invalidates all other copies.
Two logically parallel instructions access the same memory location and at least one of the instructions performs a write.

Example

```c
int x = 0;
cilk_for (int i=0, i<2, ++i) {
    x++;
} assert(x == 2);
```

Dependency Graph
Race Condition

```
int x = 0;

x++;  // B
x++;  // C

assert(x == 2);  // D
```

```
x = 0;

r1 = x;
r1++;  // B

x = r1;
```

```
r2 = x;
r2++;  // C

x = r2;
```

```
assert(x == 2);
```
What is the work?
- 24

What is the span?
- 8

What is the parallelism?
- $24 / 8 = 3$