Performance

CS 107e with Anna Zeng

SPECIALIZED
dz
zoot
30
S-WORKS
SPECIALIZED

“Every time you run a program,
you are doing science.
The more precise your hypothesis,
the more you can be surprised.
Surprise is good!”

– Dawson Engler,
a previous CS107e instructor
& speed master

Big Ideas about Performance

- Be vigilant! Ensure correctness with test-driven development.
- The compiler and machine can do whatever they like as you can’t tell the difference.
  - Reordering, elimination of operations
  - Caches, reordering instructions, overlapping instruction execution / pipelining, cache conflicts, timing inconsistency due to instruction clustering
  - Branch prediction, speculation
- How much faster can we go? Amdahl’s Law: speeding up a subsection of the code can speed up the entire program by a rate of $\frac{1}{(1-p)+\frac{p}{s}}$

How to Optimize Code

1. Don’t optimize it. Get it right first.
   (Simple code generally runs faster too.)
2. If code runs slow, profile the code.
   (Capture your program’s baseline performance.)
3. Understand what to optimize for.
   (Speed, program size, I/O with peripherals, etc.)
4. *Iteratively* optimize.

```
struct pixel { unsigned char b, g, r, alpha; };
typedef volatile struct pixel pixel_t; static pixel_t *fb;

static void draw_pixel(unsigned char v, int x, int y) {
    pixel_t (*fb_s)[fb_get_width()] = (pixel_t (*)[fb_get_width()])fb;
    pixel_t *p = &fb_s[y][x];
    p->r = v;
    p->g = v;
    p->b = v;
    p->alpha = 0xff;
}
void write_screen(unsigned char v) {
    for (volatile unsigned i = 0; i < fb_get_height(); i++) {
        for (volatile unsigned j = 0; j < fb_get_width(); j++) {
            draw_pixel(v, j, i);
        }
    }
}
```

clear/clear.c

dma/dma_test.c