Notes for 9/21/09

Moore's Law (1971) – transistor density doubles every 18 months

ILP – instruction level parallelism – superscalar processors execute multiple instructions simultaneously; this can only take you so far

TLP – thread level parallelism – e.g. multiples cores; the new focus of e.g. Intel

In 1971, feature size was 10 microns
In 2000, the Pentium 3 was at 130 nanometers
In 2004, the Pentium 4 was at 65 nanometers
In 2007, 45 nanometers
In 2009, 32 nanometers
Visible light is 400-700 nanometers.
Future: 22 nanometers – end of planar bulk CMOS (complementary metal-oxide-semiconductor)
Problem: 16 nanometers – quantum behavior

Static vs. dynamic linking: Files are compiled/assembled into *.o files and *.so files.

Static linking will bind and resolve symbols among the *.o files by the linker at compile time.

Dynamic linking will bind and resolve symbols among your executable and *.so files while running the program. In the ELF object format, you have a GOT (global offset table) and a PLT (procedure linkage table) to facilitate this. Each linking unit has at least one of these, and a function can calculate the location of these by subtracting the function's address from an immediate offset determined at compile time. In other words, a function always knows where its GOT and PLT are relative to itself.

The GOT maps symbols into their absolute addresses in memory. For regular data, this mapping is created at link time. For function calls, this mapping is created lazily at call time by the PLT.

The PLT table contains executable code. When we call a dynamically linked function for the first time, the GOT entry for the function will jump us into the PLT. The PLT will jump to the dynamic linker, which will resolve the address for the dynamically linked function and update the respective GOT entry with its address so that, instead of jumping into the PLT again, we can jump right into the function proper.

Amdahl's Law

$$\mathit{Overall\,speedup} = \frac{1}{(1-f)+\frac{f}{s}}$$

Note: *f* is fraction of execution time sped up, and *s* is speedup. So if you speed up 10% of the program execution time by 2x, then...

$$\mathit{Overall\,speedup} = \frac{1}{(1-0.1)+\frac{0.1}{2}} \approx 1.05$$

To find maximum speedup, take limit as *s* goes to infinity.