Chapter 7

- execution mode (why? how? where?) (98)
  - privilege and protection
  - software and hardware
  - how-to switch between modes

- microcode (99)
  - "macro" code may execute many smaller micro instructions
  - may add overhead but faster than CPU

- branch prediction (112)
  - prevent pipeline stalls
  - may use prediction, do both in parallel, need an "undo"
  - architectures use a scoreboard, determine dependencies and allow parallel executions
  - branches are taken 60% of the time
  - as a programmer, improve performance by adjusting branch conditions

Chapter 8

- high level languages (Java/C) (116)
  - many-to-one compilation
  - hardware independent
  - general purpose
  - application oriented
  - abstractions
- low level languages
  - one-to-one compilation
  - hardware specific
  - special/specific/system oriented
    (every proc has an assembly language - although most are similar)

- sample assembly translations
  - if statements
  - if-else statements
  - for loops (loop unrolling and moving a goto via early branch)
  - while loops
  - functions, function calls,
  - functions/calls with arguments in registers (126)

Chapter 17

- parallelism and pipelining (279)
- types of parallelism (280)
  - microscopic vs macroscopic

   -symmetric vs asymmetric
   -fine-grain vs coarse-grain
   -explicit vs implicit
-Flynn Classification (283)
   -SISD (conventional)
   -SIMD (vector/graphics processors)
   -MIMD (SMP vs AMP)
-Challenges to performance (289)
   -communication, coordination, and contention
   -bottlenecks in memory, I/O, OS, and resources
-memory hierarchy and caching problems (leads to locking)
-most processes are limited by I/O not CPU
-speedup - T1 / TN (time on one proc over time on N procs)
-speedup as a function of number of procs, ideal vs real
-large # of procs not great for general computing
-programmer issues (294)
   -must prevent caching issues through locks
   -explicit parallelism often requires complex code
   -symmetric parallelism is easier
     -1 instruction set
     -can pass a job to any processor with no consideration
   -globals are still cumbersome compared to conventional hardware
-redundant parallelism (295)
-distributed computing, clusters (295)