**Computer Organizations**

- **Central Processing Unit (CPU)**
  - ALU
  - FPU
- **Control Unit**
  - Registers
  - Cache
  - Memory
  - Display Ethernet Disk
  - Printer
- **I/O Bus**
  - MBus

**Storage Hierarchy**
- **Registers**
  - Fastest storage (as fast as CPU cycle time), but often very few (<128)
- **Caches**
  - "Small" but faster than main memory with 1 to 3 levels (1K-4Mbytes)
- **Memory**
  - Fairly fast (200ns) and quite large (1-1000Mbytes)
  - An array of cells made of dynamic random-access memory (DRAM), each cell is usually a byte and has an address. Most machines operate most efficiently on one data type called a word, which are typically composed of several cells, e.g., 4 bytes in 1 word. Address size may be unrelated to the amount of allocatable memory. Each cell is usually a byte and has an address, e.g., 100000001-000000001.
  - Registers (as fast as CPU cycle time, but other very few (<125))

**Compilation to Machine Code**
- **Compiler**
  - Converts each assembly language instruction into a bit pattern that hardware understands
  - **X = a + b**
  - **ld a, %r1**
  - **ld b, %r2**
  - **add %r1, %r2, %r3**
  - **st %r3, x**

**Machine Language**
- **Macro language** is the bit patterns that specify CPU instructions
- Understanding machine languages helps design a better instruction set and register processor
- Learn how to write very fast code, when — and only when — it's necessary
- Understand processor core mechanisms
- Understand how compilers do and how to implement code generators
- Understanding how operating systems implement security
- Build intuition about the cost of high-level functionality
- Machine language is the bit patterns that specify CPU instructions

**Compilation to Machine Code**

**Machine Language**

**Storage Hierarchy**
Instructions are composed of
  - opcode — specifies function to be performed
  - operands — data that is operated on

Most machines have only a few formats:
- 0, 1, 2, 3-operand instruction format

T ypical 0, 1, 2-operand instruction format:

Instructions are composed of

Instruction Execution

CPU's algorithm for executing a program:

```
PC <- memory location of the 1st instruction
while ( PC != lastInstructionLocation ) {
  execute ( MEM[ PC ] );
}
```

Each machine instruction has several phases:

- Fetch — Instruction fetch, increment PC
- Decode — Instruction decode
- Operand Fetch — Fetch registers
- Execute — Instruction execution
- Store — Store results

CPU's algorithm for executing a program: