Concepts Introduced in Chapter 1

- why computer organization is important
- languages for programming a computer
- computer system hardware components
- layers of abstraction
General Facts

- Computers systems (hardware and software) comprise 5-10% of the GNP of the United States.
- Most historians agree that computers have led to a third revolution for civilization.
  - agricultural
  - industrial
  - information
- Progress in computer efficiency makes more applications available (automatic teller machines, laptop computers, embedded applications like in cars, etc.).
Programmer’s Concerns

- **Data**
  - store -- memory, representation
  - initialize -- input
  - update -- operations
  - read externally -- output

- **Program**
  - specify operations (do what?) -- instructions
  - specify operands (to what?) -- addressing
  - specify ordering (when?) -- control and scheduling

- **Performance**
  - measure performance -- metrics
  - improve performance -- code generation and HW implementation
  - afford performance -- cost/performance tradeoffs
Performance Issues

- In the past (1960’s and 1970’s) size was the primary constraint on computer performance since computer memories were so small.
- Sizes of data and code still very important in embedded systems
- Today, general issues that now affect performance are:
  - lay out of data to reduce the memory hierarchy and communication overhead
  - exploitation of parallelism
Levels of Computer Programs

<table>
<thead>
<tr>
<th>Program</th>
<th>Produced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Programmer</td>
</tr>
<tr>
<td>Assembly</td>
<td>Compiler</td>
</tr>
<tr>
<td>Object</td>
<td>Assembler</td>
</tr>
<tr>
<td>Executable</td>
<td>Linker</td>
</tr>
<tr>
<td>Process</td>
<td>Loader</td>
</tr>
</tbody>
</table>
Figure 1.1: C program compiled into assembly language and then assembled into binary machine language.

```c
swap(int v[], int k)
{
    int temp;
    v[k] = v[k+1];
    v[k+1] = temp;
}
```

```
#include <stdio.h>

main()
{
    int i, j, x,
    int v[],
    int k;

    v[0] = 0;
    v[1] = 1;
    k = 0;

    for (i = 0; i < 50; i++)
    {
        j = swap(v[], k);
        k = swap(v[], k);
        v[j] = v[i+1];
        v[i+1] = v[j];
    }
}
```

```
  1: 00000000
  2: 00000000
  3: 00000000
  4: 00000000
  5: 00000000
  6: 00000000
  7: 00000000
  8: 00000000
  9: 00000000
 10: 00000000
 11: 00000000
 12: 00000000
 13: 00000000
 14: 00000000
 15: 00000000
 16: 00000000
 17: 00000000
 18: 00000000
 19: 00000000
 20: 00000000
 21: 00000000
 22: 00000000
 23: 00000000
 24: 00000000
 25: 00000000
 26: 00000000
 27: 00000000
 28: 00000000
 29: 00000000
 30: 00000000
 31: 00000000
 32: 00000000
 33: 00000000
 34: 00000000
 35: 00000000
 36: 00000000
 37: 00000000
 38: 00000000
 39: 00000000
 40: 00000000
 41: 00000000
 42: 00000000
 43: 00000000
 44: 00000000
 45: 00000000
 46: 00000000
 47: 00000000
 48: 00000000
 49: 00000000
 50: 00000000
```
Advantages of Using High-Level Languages

- Allows the programmer to think in a more natural language.
- It improves the productivity of the programmer.
- It makes the programs portable from one machine to another.
- Cost of portability is often lower performance.
- System SW and HW attempt to minimize this loss at an acceptable implementation cost.
Figure 1.2: A simplified view of hardware and software as hierarchical layers, classically shown as concentric rings building up from the core of hardware to the software closest to the user.
Figure 1.3: An example of the decomposability of computer system.
Key Components of a Computer System

- input devices
  - mouse
  - keyboard
- output devices
  - screen
  - printer
- memory
  - programs
  - data
- processor
  - datapath
  - control
Figure 1.9: Inside the processor chip used on the board shown in Figure 1.8.
Layers of Abstraction

- Both software and hardware systems are designed using layers of abstraction.
- Each lower layer allows the next outer layer to interface with it without requiring knowledge of its details.
- instruction set architecture
  - interface between the hardware and the lowest level of software
- implementation
  - hardware that carries out the architecture abstraction
Types of Memory

- Main Memory
  - DRAM (Dynamic Random Access Memory)
    - volatile
    - destructive reads
    - requires refreshing
    - 8 to 16 times cheaper than cache

- Cache
  - SRAM (Static Random Access Memory)
    - nondestructive reads
    - requires more transitors
    - 8 to 16 faster than DRAM
Secondary Memory

- nonvolatile
- about 100,000 times slower than DRAM
- about 50 times less expensive than DRAM
- consists of
  - rotating platters
  - movable arm with read/write head
Computer Networks

- Allow communication between computers.
- Advantages
  - Information is exchanged at high speeds.
  - Can share I/O devices on the network.
  - Can use computers without being near them.
- Types
  - Local Area Networks (LAN)
    - connects computers within a building
    - examples include Ethernet, FDDI
  - Wide Area Networks (WAN)
    - Internet
Figure 1.13: Relative performance per unit cost of technologies used in computers over time.

<table>
<thead>
<tr>
<th>Year</th>
<th>Technology used in computers</th>
<th>Relative performance/unit cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1955</td>
<td>Vacuum tube</td>
<td>1</td>
</tr>
<tr>
<td>1965</td>
<td>Transistor</td>
<td>35</td>
</tr>
<tr>
<td>1975</td>
<td>Integrated circuit</td>
<td>900</td>
</tr>
<tr>
<td>1995</td>
<td>Very large-scale integrated circuit</td>
<td>2,400,000</td>
</tr>
</tbody>
</table>
Figure 1.14: Growth of capacity per DRAM chip over time.
VLSI Circuits

- Chips are made from silicon from which tiny areas can be transformed into:
  - conductors of electricity
  - insulators of electricity
  - switches (conductors or insulators)
- Transistors are switches.
- A VLSI circuit contains millions of conductors, insulators, and switches.
Figure 1.15: The chip manufacturing process
Figure 1.18: A Pentium Pro die.
Computer Architects

- Computer architects need to have knowledge from many disciplines.
  - Hardware
    - integrated circuit technology
    - digital design
  - Software
    - compilers
    - operating systems
Fallacies and Pitfalls

- fallacy - a commonly held misconception
  - Computers have been built in the same way too long and a new approach will be needed to get improved performance.
- pitfall - an easily made mistake
  - Ignoring the progress of hardware when planning a new machine.
Figure 1.20: Performance increase of workstations, 1987-1997.