Microcomputer Architecture
- Basic Computer Organization
- The Processor
- A User’s View of Computer Systems
- Programming Language
A computer system is comprised of three main components:
- A central processing unit (CPU) or processor
- A memory unit
- Input/output (I/O) devices
Architecture
Instruction Fetch

- Fetch an instruction
  - Place the appropriate address on the address bus
  - Activate the memory read signal on the control bus to indicate to the memory unit that an instruction should be read from that location
  - Memory places the instruction on the data bus
Pipeline

- The processor acts as the controller of all actions or services provided by the system.
- **Execution cycle**
  - **Fetch**: fetch an instruction from the memory
  - **Decode**: decode the instruction (identify the instruction)
  - **Execute**: execute the instruction (perform the action specified by the instruction)
Instruction Decoding

- Identify the instruction that has been fetched from the memory.
- The machine code is coded in a specific format

```
101101 001100000000 00011 00001 00010
```

```
add  r3, r1, r2
```
Instruction Execution

- **CU (control unit)**
  - To generate the appropriate signals for the decoded instruction.

- **ALU (Arithmetic and Logic Unit)**
  - Perform arithmetic operations, ex: add, sub, …
  - Perform logical operations, ex: and, or,
Register

- A small amount of very fast computer memory used to speed the execution of computer programs by providing quick access to commonly used values.
- **PC (program counter)**
  - Store the address of the next instruction to be fetched or to executed.
System Clock

- The system clock
  - Provides the timing signal to synchronize the operations of the computer system
  - Defines the speed of the system.
A User’s View of Computer Systems
Programming Language

- High-level language
  - Independent of machine
- Assembly language
  - Highly dependent on machine
- Machine language
  - Absolute dependence on machine

<table>
<thead>
<tr>
<th>C Language</th>
<th>Assembly Language</th>
<th>Machine Language (in hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>result ++</td>
<td>inc</td>
<td>FF060A00</td>
</tr>
<tr>
<td>xx = 45</td>
<td>mov xx, 45</td>
<td>C7060C002D00</td>
</tr>
<tr>
<td>mask = mask &amp; 128</td>
<td>and mask, 128</td>
<td>80260E0080</td>
</tr>
<tr>
<td>marks = marks + 10</td>
<td>add marks, 10</td>
<td>83060F000A</td>
</tr>
</tbody>
</table>
Assembly Language

- Low-level language
  - Each instruction performs a much lower-level task compared to a high-level language instruction

- One-to-one correspondence between assembly language and machine language instructions
  - For most assembly language instructions, there is a machine language equivalent
  - Assembler translates assembly language instructions to machine language instructions

- Directly influenced by the instruction set and architecture of the processor (CPU)
Why Assembly Language

- Two main issues:
  - Efficiency
    - Space-efficiency
      - Small code size
    - Time-efficiency
      - A well-written program in assembly language runs faster
  - Accessibility to system hardware
Comparison of C and Assembly Language

![Graph: Multiplication time comparison on a 2.4GHz Pentium4 system]

- **C version**
- **Assembly language version**

<table>
<thead>
<tr>
<th>Number of calls (in millions)</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>80</td>
<td>4</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
</tr>
</tbody>
</table>

The graph shows the comparison of multiplication time between C and Assembly language versions on a 2.4GHz Pentium4 system.