

## ***Computer Organization and Architecture***

---

Department of Computer Science  
University of Maine

## ***Low - Level Programming***

---

"How do I get this thing to do what I want it to do????"

Since computers were invented, we've been trying to figure out how to talk to them

"I hate this machine. It never does what I want it to do, it does only what I tell it to do!"

## ***Physical Communication Mechanisms***

---

- Cables and patch boards (1940's)  
Toggle switches  
Punch cards and paper tape (1950's)  
Teletypewriter (1950's)  
Keyboard and CRT (Glass Teletype)
- Numerous other devices introduced in recent years  
Mice  
Touchscreens  
Light pens  
Voice  
Tabletop computers...

## ***What Language Does a Computer Speak?***

---

- **There are several levels of abstraction involved in communicating with a computer**

Application (spreadsheet, word processor, game, etc.)

-- is created by a High-Level Programming Language

- which generates Assembly Language

- which is then assembled into Object Code

- which is then linked into Machine Language

- **This is not the bottom of the ladder**

## ***A Task***

---

- **Get the computer to display "Hello, World" on the monitor**
- **Here it is in a few programming languages:**

-Basic

```
10 PRINT "Hello, World"
```

-C

```
#include <stdio.h>
main()
{
    printf("Hello, World!\n");
}
```

- Pascal

```
Program Hello(Input, Output);
begin
    writeln("Hello, World!");
end.
-C++
#include <iostream.h>
int main()
{
    cout << "Hello, World" << endl;
    return 0;
}
```

---

```

- Java
class Hello {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
-HTML/Javascript
<html>
<body>
<script language="javascript">
document.write("Hello World!");
</script>
</body>
</html>

```

## Intel 8086 Assembler

---

```

dosseg
STACK SEGMENT PARA STACK 'STACK'
db 100h dup ?
STACK ENDS

DATA SEGMENT PARA PUBLIC 'DATA'
msg db 'Hello,World!',0dh,0ah
msgLen EQU $-msg
DATA ENDS

CODE SEGMENT PARA PUBLIC 'CODE'
hello PROC near
start:
    mov ax, 4000h
    mov bx, 1
    mov dx, offset msg
    mov cx, msgLen
    int 21h
    mov ax, 4c00h
    int 21h
hello endp
CODE ENDS
end start

```

## Machine Language!

---

```

11101001000100110000000001010100011
01000011010010111001100100000011010
01011100110010000001100001001000000
11011010110010101110011011100110110
00010110010100001101000010101011010
00000100110111010000000110000000111
0011010010000110111000000000001001
1001100110101000001

```

## Slightly more comprehensible...

---

```

11101001 00010011 00000000 01010100 01101000
01101001 01110011 00100000 01101001 01110011
00100000 01100001 00100000 01101101 01100101
01110011 01110011 01100001 01100101 00001101
00001010 10110100 00001001 10111010 00000011
00000001 11001101 00100001 10111000 00000000
01001100 11001101 01000001

```

## Better yet!

---

```

E9 13 00 54 68 69 73 20 69 73 20 61
20 6D 65 73 73 61 67 65 0D 0A B4 09
BA 03 01 CD 21 B8 00 4C CD 21

```

## Another version

---

```

; Hello, World Version 2 - Write directly to screen
jmp start
msg db 'Hello, World'
msgLen EQU 12
attr EQU 79h

start:
    mov ax, 0b800h
    mov es, ax
    mov cx, msgLen
    mov si, offset msg

; calculate the address needed to display the message in the middle
; of the screen. The screen is 80x25 and each line occupies
; 160 bytes because each character has an attribute byte
    mov di, 12 * 160 + 68
    mov ah, attr

L1:
    mov al, [si]
    mov es:[di], ax
    inc si
    inc di
    inc di
    loop L1
done:
    mov ax, 4c00h
    int 21h

```

### **The Nature of Computation**

- Ultimately every problem that is solvable by a computer must be expressed as a string of 0's and 1's
- The nature of computation was understood by mathematicians long before we had computers (1930's)
  - Turing Machines
  - Church's Lambda Calculus
  - Recursive Function Theory

### **Turing Machine**

- has an infinite tape divided into cells
- has a single read/write head
- each cell can contain 0 or 1 (or be blank)
- can only move left, move right, read a single cell or write a single cell

### **Church's Thesis**

- Anything that is computable can be computed by a Turing Machine

### **Gödel's Incompleteness Theorem**

- Any consistent system that is powerful enough to express arithmetic is incomplete
- There are true statements within the system that cannot be proven to be true
- Can be extended via Church's thesis to computation

### **Ramifications of Gödel's Incompleteness Theorem and Church's Thesis**

- All non-trivial computer languages are equal in power
- One cannot write a computer program to determine properties of other computer programs (Is it a virus? Will it ever stop running? ...)
- There is truth outside computation
- There are uncomputable problems

### **Computer Architecture**

- Baer: "The design of the integrated system which provides a useful tool to the programmer"
- Hayes: "The study of the structure, behavior and design of computers"
- Abd-Alla:
  - "The design of the system specification at a general or subsystem level"
- Foster: "The art of designing a machine that will be a pleasure to work with"
- Hennessy and Patterson:
  - "The interface between the hardware and the lowest level software"

### **Common themes in definitions**

- Design / structure
- Art
- System
- Tool for programmer and application
- Interface

### **Stallings Definition**

- Computer "architecture" refers to the set of attributes of a computer system that are visible to a programmer
- Those attributes have a direct impact on the execution of a program
  - Instruction sets
  - Data representations
  - Addressing
  - I/O
- Example: Is there a multiply instruction?

### **Computer Organization**

- Synonymous with "architecture" in many uses and textbooks
- Organization is concerned with how features (attributes) are implemented
  - Control signals, interfaces, memory technology.
- Transparent to the programmer
  - Example: Is there a hardware multiply unit or is it done by repeated addition?

### **Architecture & Organization**

- All members of the Intel x86 family share the same basic architecture
- All members of the IBM System/370 family share the same basic architecture
- This gives code compatibility
  - At least backwards
- Organization differs between different versions

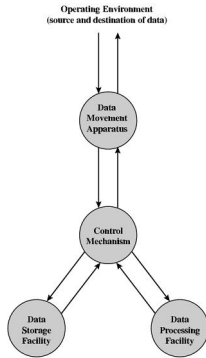
### **Structure & Function**

- Structure is the way in which components relate to each other
- Function is the operation of individual components as part of the structure

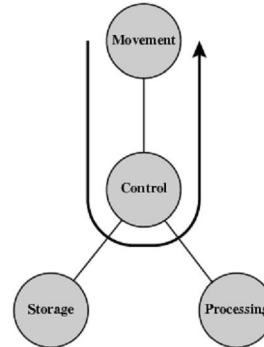
### **What is the function of a computer?**

- "Computer" used to be a job title, not a piece of equipment
- Requirements of a computer:
  - Process data
  - Store data
  - Move data between the computer and the outside world
  - Control the operation of the above

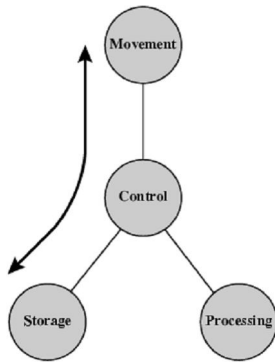
**Functional View**



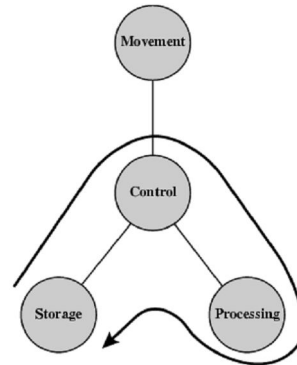
**Operations (a) Data movement**



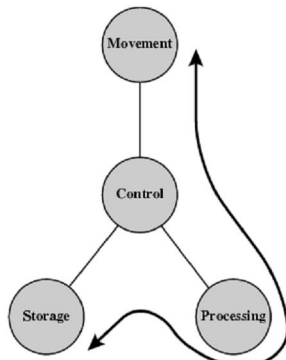
**Operations (b) Storage**



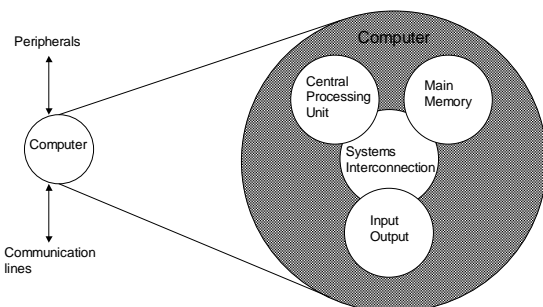
**Operation (c) Processing from/to storage**



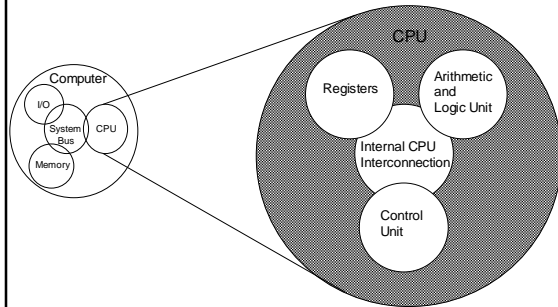
**Operation (d) Processing from storage to I/O**



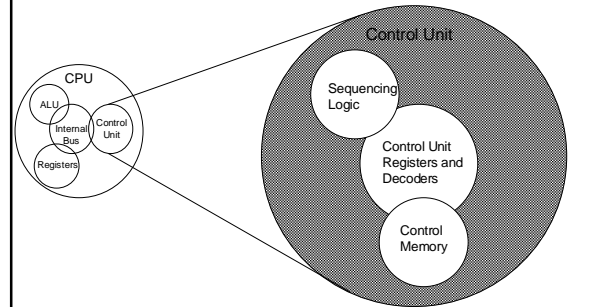
**Structure - Top Level**



### **Structure - The CPU**



### **Structure - The Control Unit**



### **Outline of the Book (1)**

- Computer Evolution and Performance
- Computer Interconnection Structures
- Internal Memory
- External Memory
- Input/Output
- Operating Systems Support
- Computer Arithmetic
- Instruction Sets

### **Outline of the Book (2)**

- CPU Structure and Function
- Reduced Instruction Set Computers
- Superscalar Processors
- Control Unit Operation
- Microprogrammed Control
- Multiprocessors and Vector Processing
- Digital Logic (Appendix)