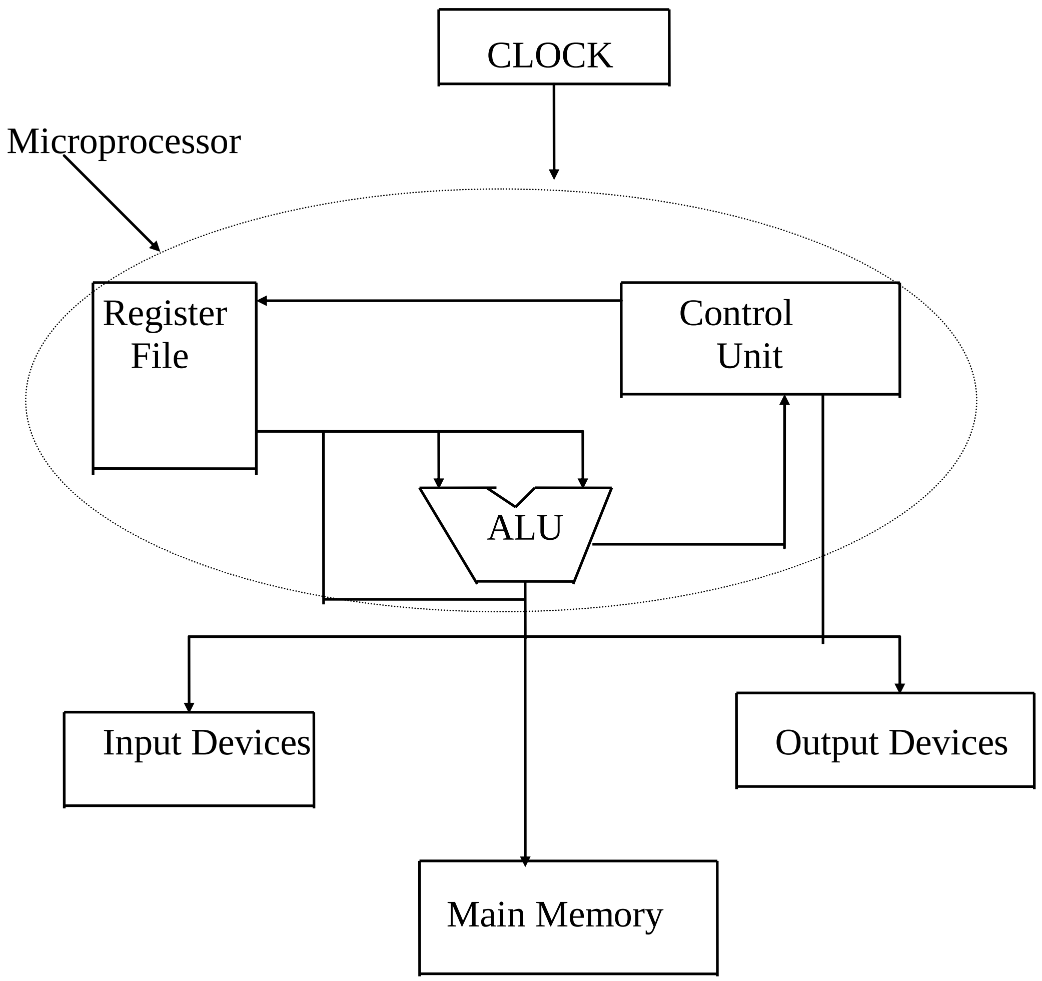
What is a computer?

A computer is a machine (hardware) usually electronic that is capable of executing a sequence of instructions from a stored program (software)

- A micro-processor based computer is called a microcomputer.
- Major components of a computer:

    1- Microprocessor (CPU)
    2- Memory (RAM and/or ROM)
    3- input/output devices
    4- Clock. (controls all devices)



Block Diagram of A Typical Microprocessor

Microprocessor

Contains the following main components:

- **ALU**
  Carries out the arithmetic functions of the computer. Also it performs the logical functions.
- **Control Unit**
  Is the "brain" of the computer. Determines which function to be done on data, decodes instructions, and controls the flow of the program
- **Register File**
  These are high speed storage locations. Hold internal data that the processor currently using while executing programs.

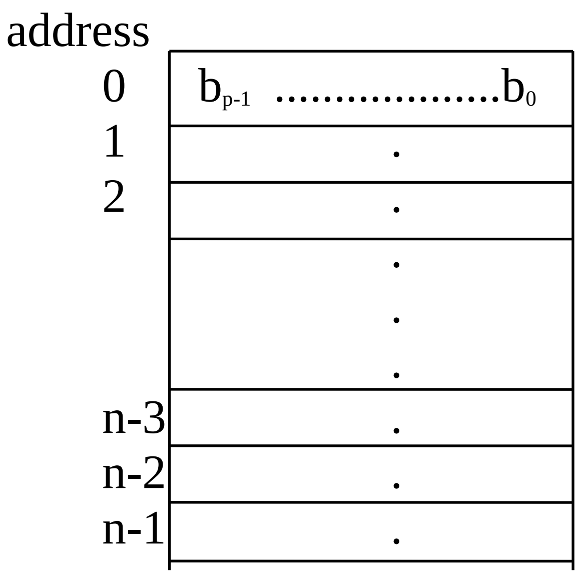
Two kinds of registers exist:

1- **Special purpose registers**
Used by CPU. Most important are the program counter (PC), and the processor status register.

2- **General purpose registers**
Used by programmers

Main Memory

A linear list of memory cells. Each cell holds a data word. A word could be a byte (eg. VAX), 2 bytes, 4 bytes .....



memory unit of n cells

- All memories share two organizational features:-
  1) Each information unit is the same size.

2) An information unit has a numbered address associated with it by which it can be uniquely referenced.

A memory cell is characterized by two things:-

1) An address
2) Content

- The basic memory cell on many computers including the VAX holds one byte (8 bits). Such machines are called byte-addressable. Other machines have larger storage units.
- Components of the computer are connected by Buses.

A bus in the simplest form is a set of wires that used to carry information in the form of electrical signals between CPU and memory and CPU and I/O

- There are 3 buses in the system
  - Data Bus
  - Address Bus
  - Control (signal) Bus
- VAX has a 32-bit address bus and a 32-bit data bus

Computer Architecture

- The architecture of a computer system is the **user-visible** interface: The structure and operation of the system as seen by the programmer. It includes
  - The instruction set the computer can obey
  - The ways in which the instructions can specify the locations of data to be processed
  - The type and representation of data
  - The format in which the instructions are stored in memory
- Most modern computers follow the ***Von Neuman Architecture***: This architecture is based on the stored program concept. The program to be executed along with the data on which it operates are stored in a memory device attached to the processor.

Instruction Set Architectures

- A computer's hardware has a small, fixed number of operations, or instructions that it can perform. This is called the instruction set of the machine. It is defined by the manufacturer.

- We will be concerned with the following issues:
  - <u>Operand storage in CPU</u>
    Where are operands kept other than in memory?
  - <u>Number of explicit operands named per instruction</u>
    How many operands are named explicitly in a typical instruction?
  - <u>Operand location (addressing modes)</u>

Can any ALU instruction operand be located in memory or must some or all of the operands be in internal storage in the CPU? If an operand is located in memory, how is the memory location specified?

- Operations

  What operations are provided in the instruction set?

- Type and size of operands

  What is the type and size of each operand and how is it specified?

Instruction Set

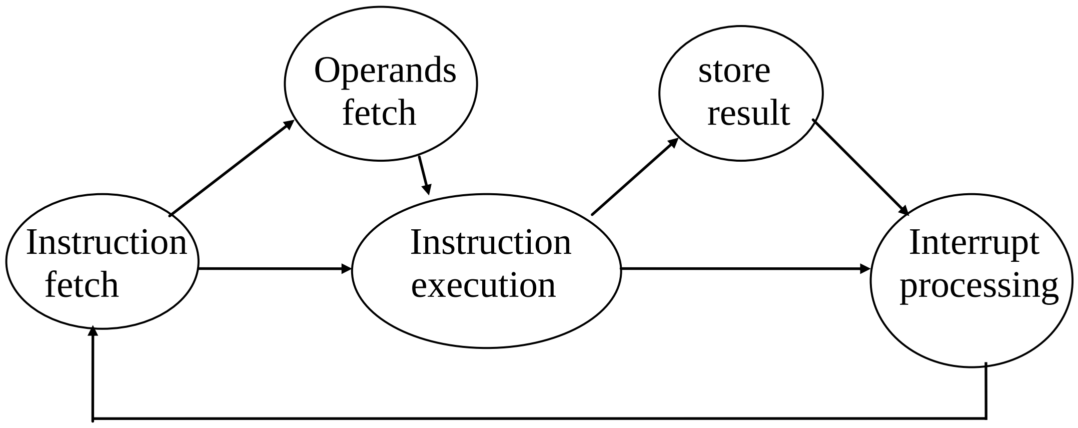
- The instruction set of a microprocessor is the collection of all the machine instructions that the processor can obey.
- Types of instructions
  - Data movements
  - Arithmetic/ logic
  - Control (modify the execution order)

Instruction Execution

- A computer is controlled by a program, which is a sequence of instructions. Each instruction specifies a single operation to be performed by the CPU

  Eg.

  ADD R7, ALPHA
- Instructions are stored in memory along with the data on which they operate.
- The processor **FETCHES** and instruction from memory, **INTERPRETS** what function is it to be performed, and **EXECUTES** the function on its operands (data)
- This is called the ***instruction fetch/execute cycle***



Instruction Cycle

All instructions are composed of two components

1. An operation code (OPCODE) that specifies the function to be performed.
2. One or more operand specifiers that describe the locations of the information units on which the operation is performed.

- A computer by itself can't do anything. It needs software. A sequence of instructions called a program, which makes the computer work.
- Microprocessors by themselves only react to patterns of electrical signals.
- These patterns comprise what we call a ***machine language***, which consists of the binary patterns of 0's and 1's that represent the signals the CPU understand.
- Machine language is machine dependent and varies from model to model of computers.

eg.

    0000 0100 0110 0111 1001 111 000 0100 1100 0000

on the VAX means

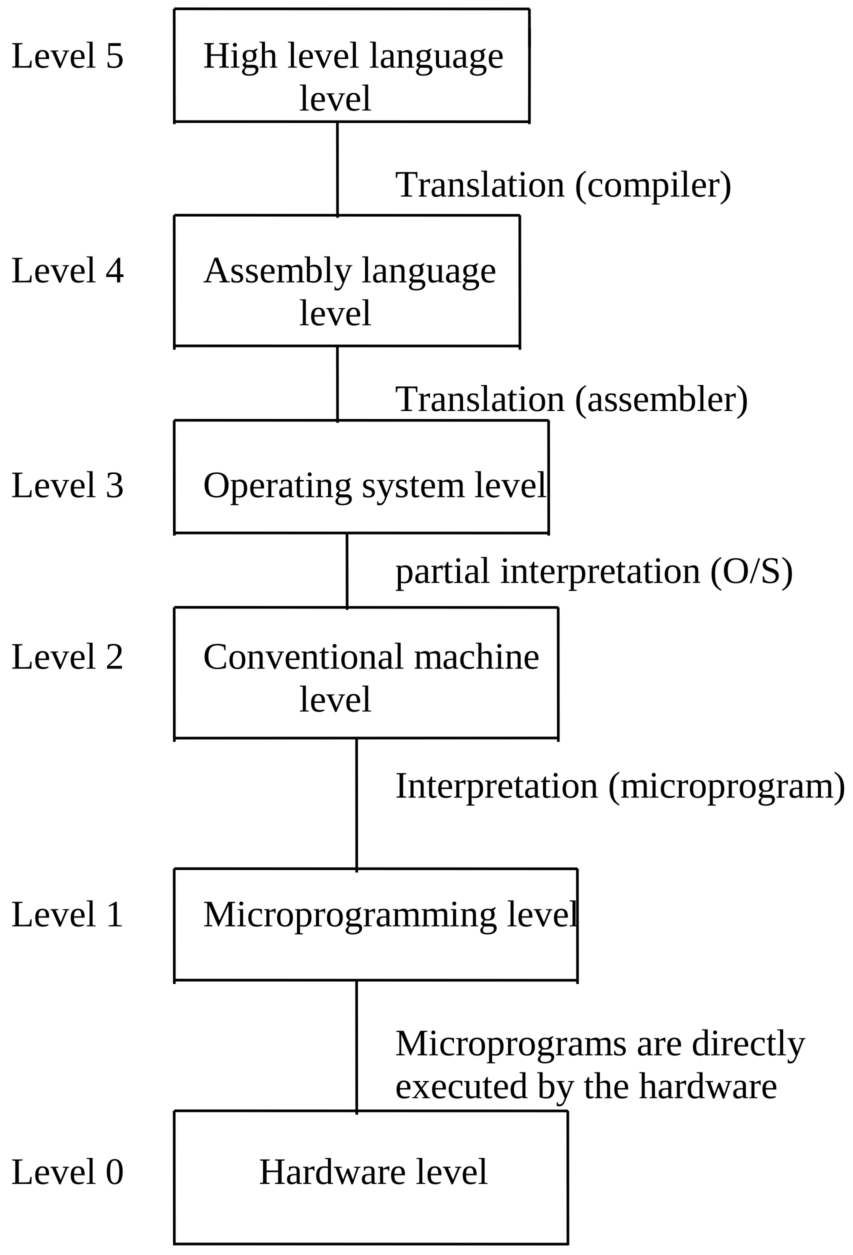
    add the integer 6 to the integer stored at memory location 467

"1100 0000" is the code for add instruction on the VAX

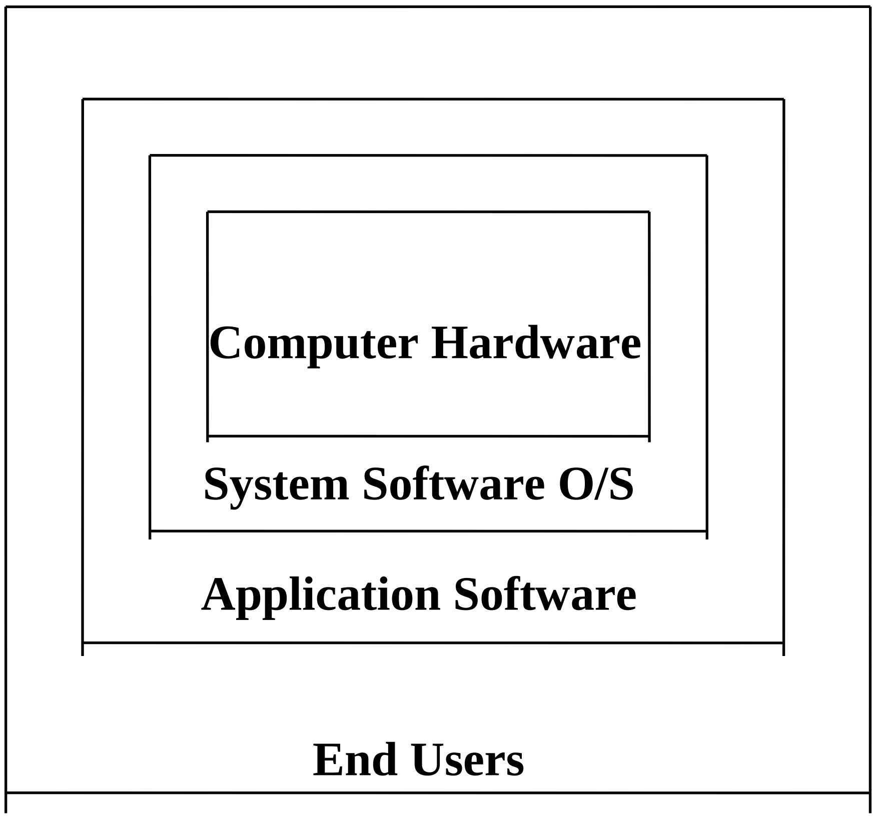
- The machine designers decide on these codes and patterns when designing the machine.
- Programming in machine language is very difficult and tedious.
- A better way is needed to write programs, so Assembly language and high-level language were introduced

Assembly Language

- Is a symbolic code that allows mnemonics for machine language instructions and symbolic names for memory locations.
  eg.
  instead of using "1100 000", we write the symbol ADD to mean the same thing
- It is a programming language in which instructions correspond closely to the individual machine instructions that are carried by a particular computer. So assembly language, like machine language, is machine dependent.
- The VAX/VMS assembly language is called MACRO
- Assembly language uses symbols to represent the operation codes while a special notation is used to represent the different addressing modes.
- A translator is used to translate the assembly code into machine code. This translator is called an **assembler**.
- The name of the assembler we will use is MACRO.


Level 5
High level language level
Translation (compiler)
Level 4
Assembly language level
Translation (assembler)
Level 3
Operating system level
partial interpretation (O/S)
Level 2
Conventional machine level
Interpretation (microprogram)
Level 1
Microprogramming level
Microprograms are directly executed by the hardware
Level 0
Hardware level

Multilevel Machine



- The most important system software is the operating system, which is an integrated system of programs that manages the system resources, and provides various support services such as the computer executing the application programs of users.

Major Categories of Programming Languages:

- **Machine Language** (First-generation Language)
  Most basic level of programming languages. Uses binary coded instruction.
- **Assembly Language** (Second-generation Language)
  They are sometimes called symbolic languages, because symbols are used to represent operation codes and storage locations.
- **High-Level Languages** (Third-generation Languages)
  Use instructions called statements, that closely resemble human language or standard notation of mathematics.
- **Fourth-generation Languages**
  A variety of programming languages that are more nonprocedural and conversational. The programmers specify the result they want, while the computer determines the sequence of instructions that will accomplish these results.

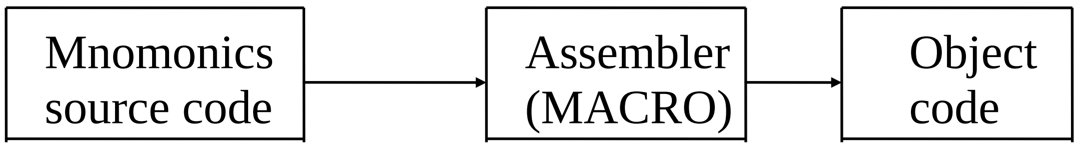
eg.

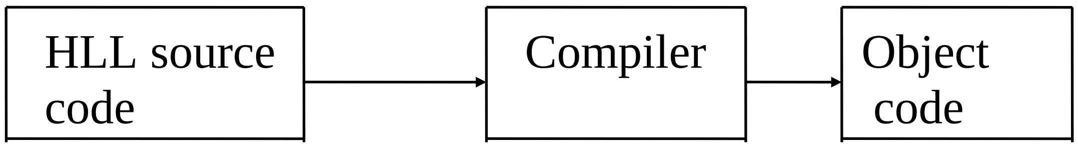
```
SUM = SUM * 2 + ALPHA/3;
```

in C

```
MULL3       SUM, #2, R6
DIVL3#3, ALPHA, R7
ADDL3       R6, R7, SUM
```

Each one of the above instructions will correspond to one machine instruction. (a pattern of 1's and 0's)





Advantages of assembly language?

- **Speed**
  - Assembly language programs are generally the fastest programs around.
  - Machine language created by compilers are usually longer and slower.
- **Space**
  Assembly language programs are often the smallest.
- **Capability**
  - You can do things in assembly which are difficult or impossible in HLLs.
  - Applications requiring access to low level machine capability are often written in assembly language
    eg. device drivers.
- **Knowledge**
  - Your knowledge of assembly language will help you write better programs, even when using HLLs.

- Assembly language requires learning the architecture of the specific machine on which assembly language is used

High-Level vs. Low-level languages

- An LLL requires long sequences of instructions to perform a complex task.

An HLL provides complex statements that perform complex jobs. ex. the for loop in C or other languages

- An LLL is machine dependent. Every microprocessor has its own assembly language. (instruction set dependent)

An HLL is not machine dependent (compilers are)
What about JAVA?

- Programmers must know the internal structure of the microprocessor, ex. number of registers, and how to use them

An HLL provides a high level of abstraction from the hardware details. Hides the internal structure of the processor.

Reduced Instruction Set Computers (RISC) vs. Complex Instruction Set computer (CISC)

Characteristics of CISC (ex. VAX)

1. Large number of instructions
2. some instructions that perform specialized tasks and are used infrequently
3. large variety of addressing modes
4. variable length instruction format
5. instructions that manipulate operand in memory

Characteristics of RISC (ex. SPARC)

1. few instructions
2. few addressing modes
3. memory access limited to load/store instructions
4. all operations done within the registers
5. fixed length instruction format
6. single cycle instruction execution