Chapter 4

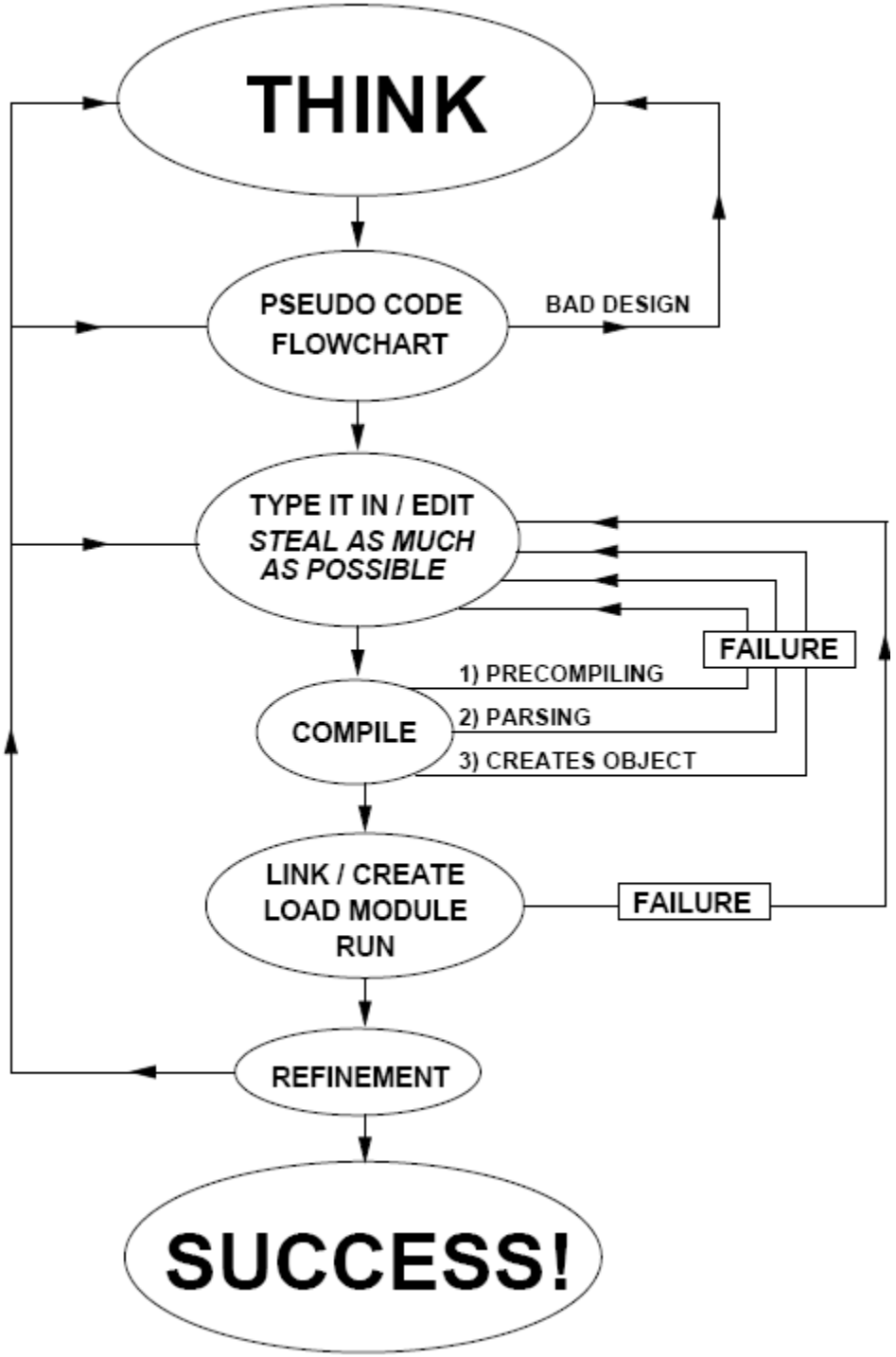
Programming and the Program Execution Process

Outline

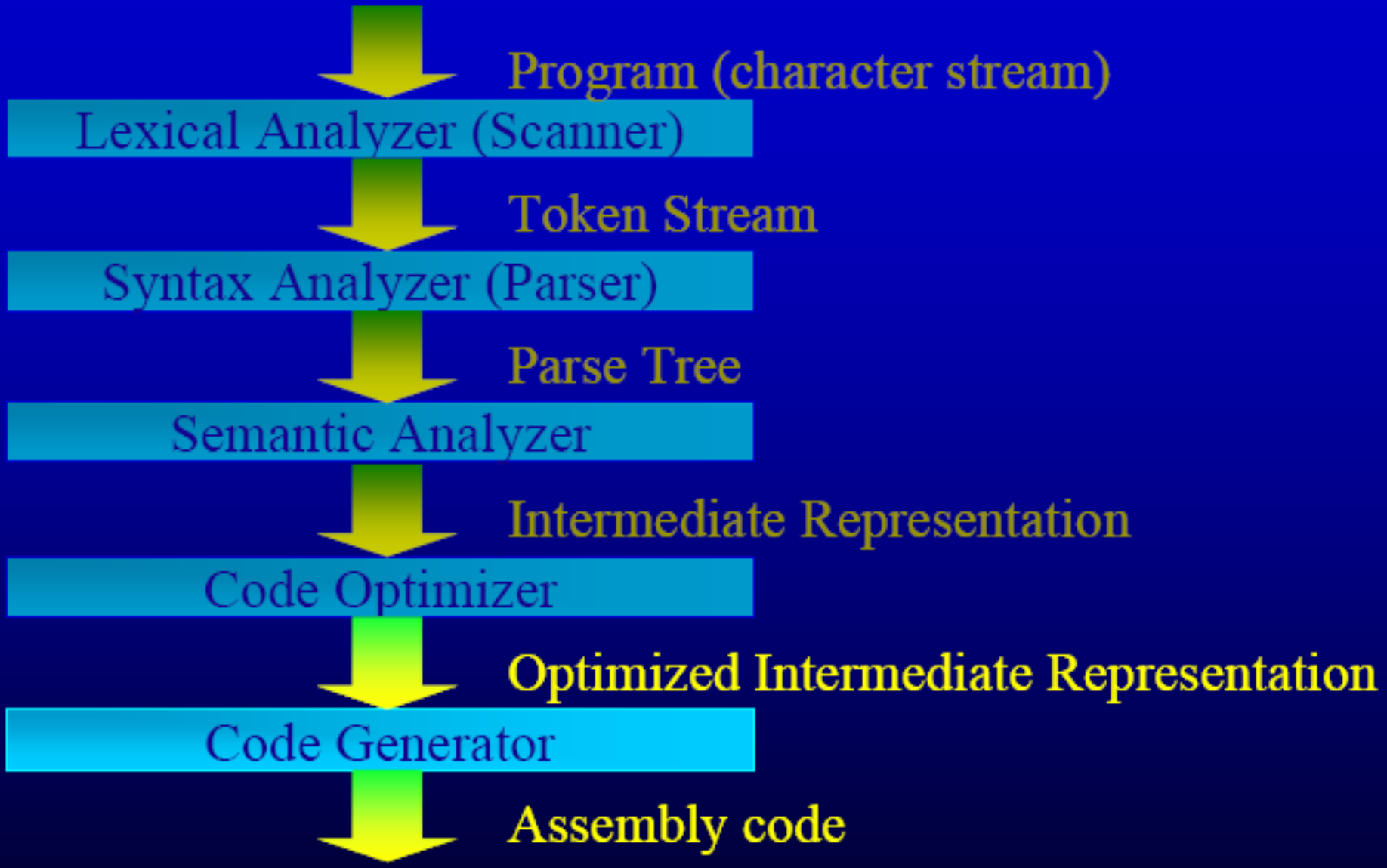
- Computer Programs and the program development Environment (procedure)
- Program execution process
- The Operating System
  - Functions and components
- Programming languages

Computer program development Process

This model of editing, preprocessing, parsing, object-code generation, linking, loading and running is followed by all computer code-development environments.



Anatomy of a Compiler



Program

- **The behaviour of the computer is controlled by a set of step-by-step instructions called program.**
- **Every thing interesting or useful about a computer behaviour results from its program rather than the hardware it carries it.**
- **The computer processes information by executing a program stored in memory.**
- **The execution takes place with in the CPU and is controlled by the CU.**

Program Execution

- Executing a program requires the CPU to examine each program instruction in memory and send out the command signals required to perform each instruction.
- Although instructions are normally performed consecutively (sequencing), they can be skipped (branching) or repeated (looping) under program control. (Structured Programming)
- During execution, data can be entered by the operator (user), or from a saved file.
- After processing, the program output can be displayed or printed as a result.

Example…

- The sequence of instructions could be as follows :- (Sequencing)
    - Get out of bed
    - Have breakfast
    - Get dressed
    - Get into car
    - Drive to work
    - End of program

Example: Finding the square root of a number *x*

(May use Branching and Looping)

1. Start with a guess, *g*
2. If *g*g* is close enough to *x*, then g is a good approximation of the square root of *x*, *jump to step 5*
3. Otherwise, create a new guess by averaging *g* and *x/g*.

   i.e., $g_{new} = (g_{old} + x/g_{old})/2$
4. Using this new guess, go back to step 2
5. Stop

You can easily code this algorithm, compile the source, and run the program

When you double click on an icon to run a program, here is what happens:

1. The program, which is stored inside the hard disk drive, is transferred to the RAM memory.
2. The CPU, using a circuit called memory controller, loads the program data from the RAM memory. A program is a series of instructions to the CPU.
3. The data, now inside the CPU, is processed.
4. What happens next will depend on the program. The CPU could continue to load and execute the program or could do something with the processed data, like displaying something on the screen.

The sequence of CPU steps can be expressed in pseudocode:

loop

fetch the instruction pointed by (the value in) IP

advance the instruction pointer (IP)

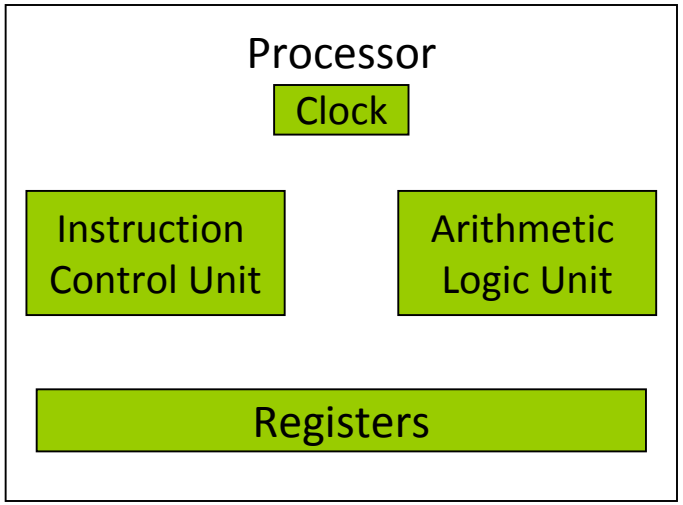
decode the instruction

if memory operand needed, read value from memory

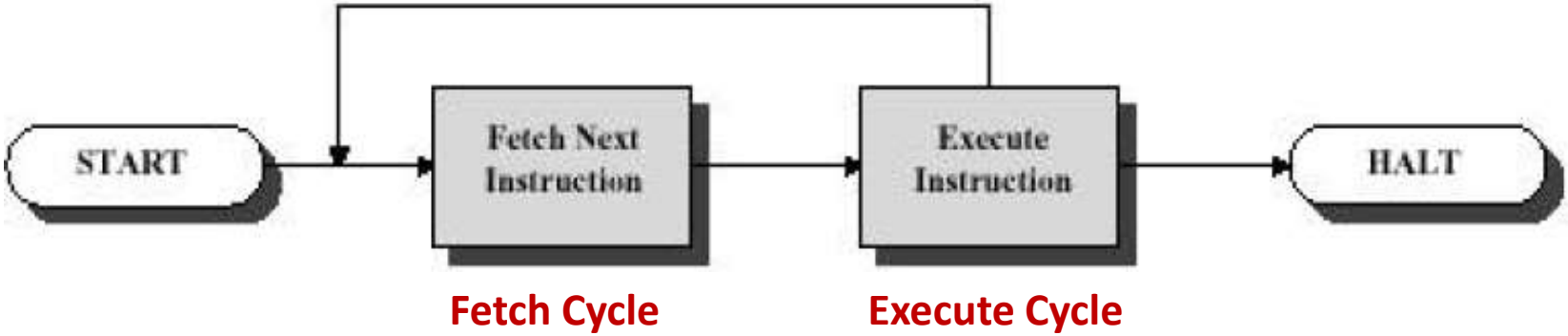
execute the instruction

if result is memory operand, write result to memory

continue loop



- Processing required for a single instruction is called an instruction cycle (Fetch-Execute Cycle), and can be viewed as shown below: **2 Steps**

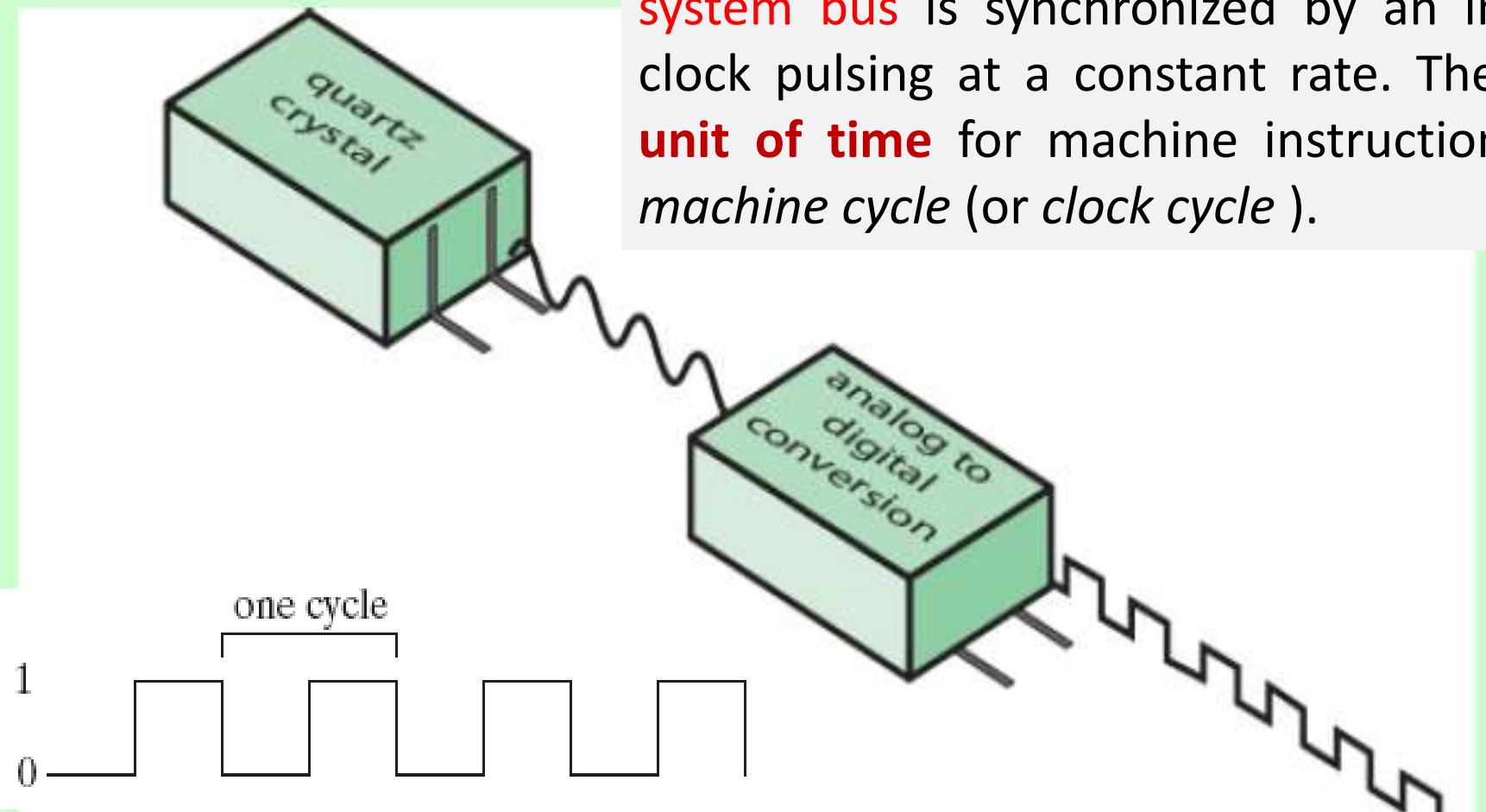


- Fetch – CPU(CU) reads an instruction from a location in memory and decodes the instruction (determine what it means)
  - Program counter (PC/Instruction Pointer) register keeps track of which instruction executes next
  - Normally, CPU increments PC after each fetch
  - Fetched instruction is loaded into the instruction register (IR)

- Execute - CPU executes the instruction
  - May involve several operations
  - May utilize previously changed state of CPU
  - General categories:
    - CPU-Memory: Data may be transferred from CPU to memory or vice-versa
    - CPU-IO: Data may be transferred between CPU and an I/O module
    - Data Processing: CPU (ALU) may perform some arithmetic or logic operation on the data
    - Control: An instruction may specify that the sequence of execution be altered

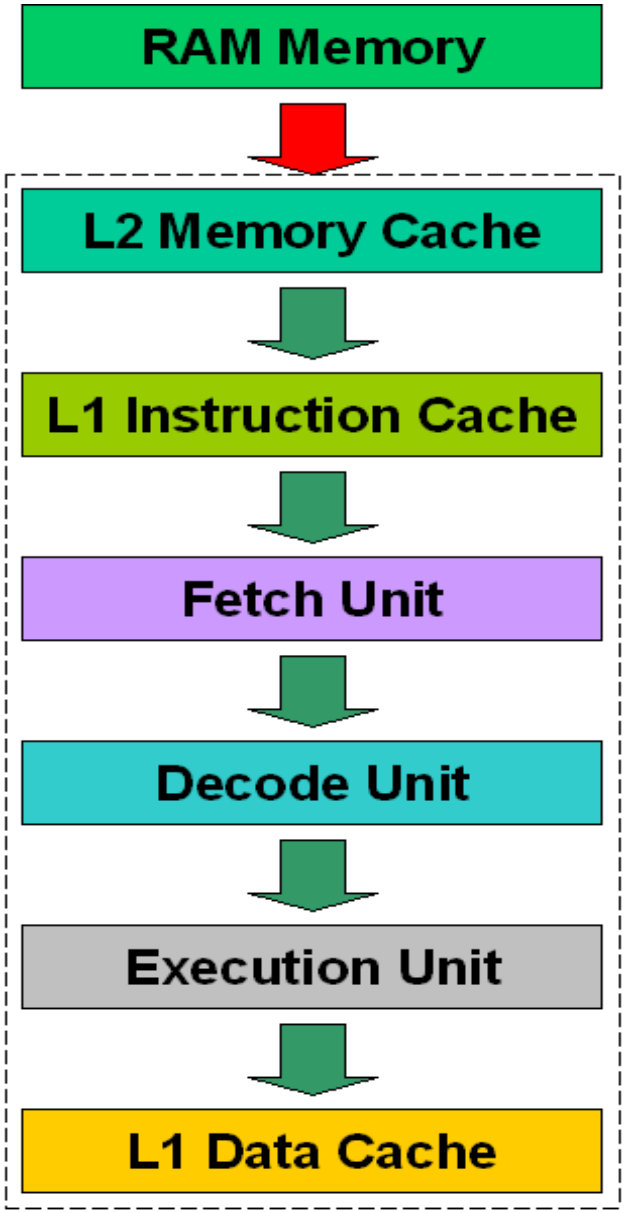
System Clock

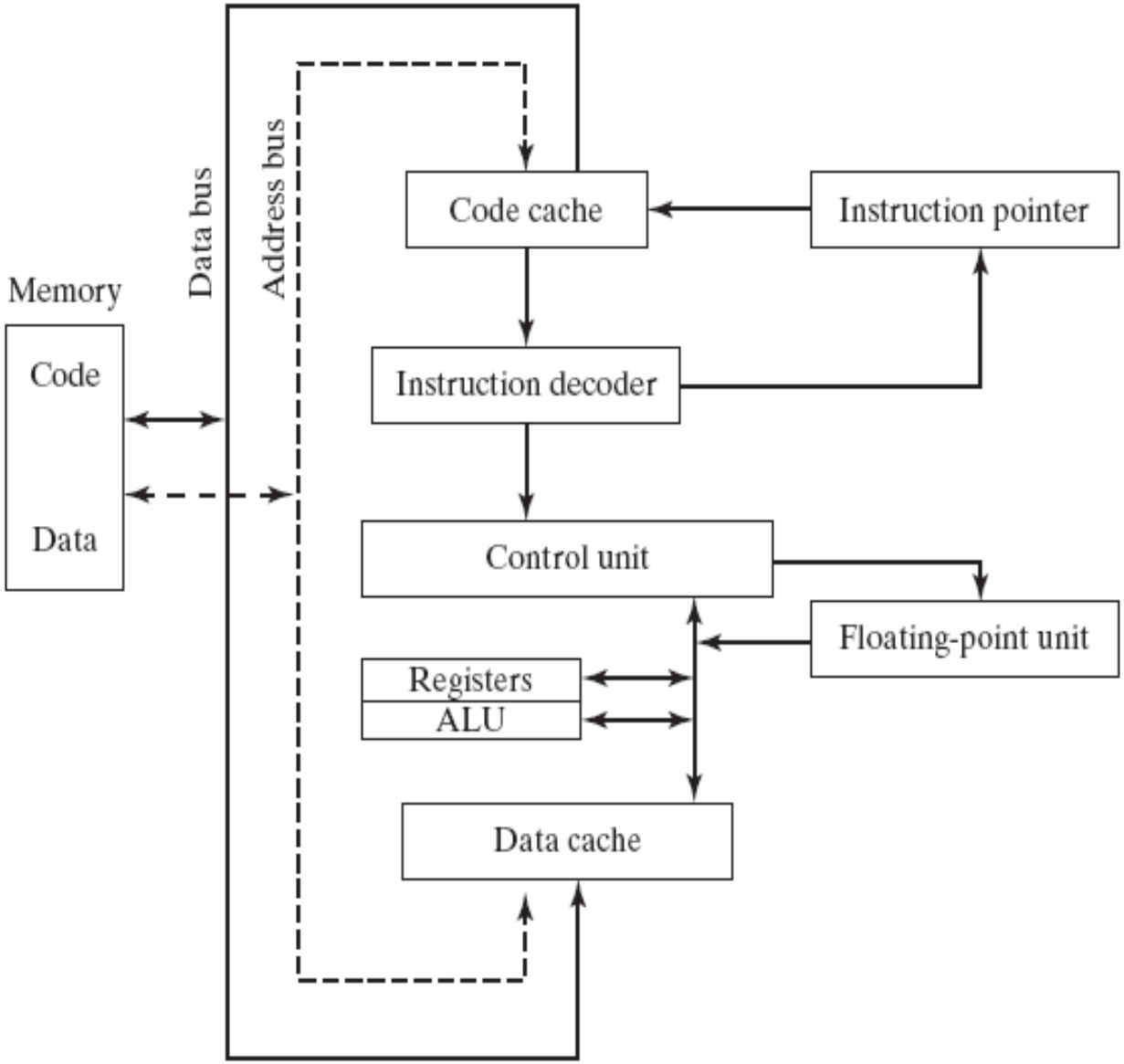
Each operation involving the CPU and the system bus is synchronized by an internal clock pulsing at a constant rate. The **basic unit of time** for machine instructions is a *machine cycle* (or *clock cycle*).



How much time it takes to execute an instruction that takes ten cpu cycles (10 clock ticks) in a 4 ghz processor?

Simplified block diagrams of a modern CPU





CPU Cont...

- Instruction execution takes place in discrete steps
  - —Fetch, decode, load and store, arithmetic or logical
  - —Usually require multiple clock cycles per instruction
- Pipelining → simultaneous execution of instructions

CU starts execution of next instruction while other instructions are still being processed in other parts of the CPU.

CPU Cont…

Processor speed depends on:

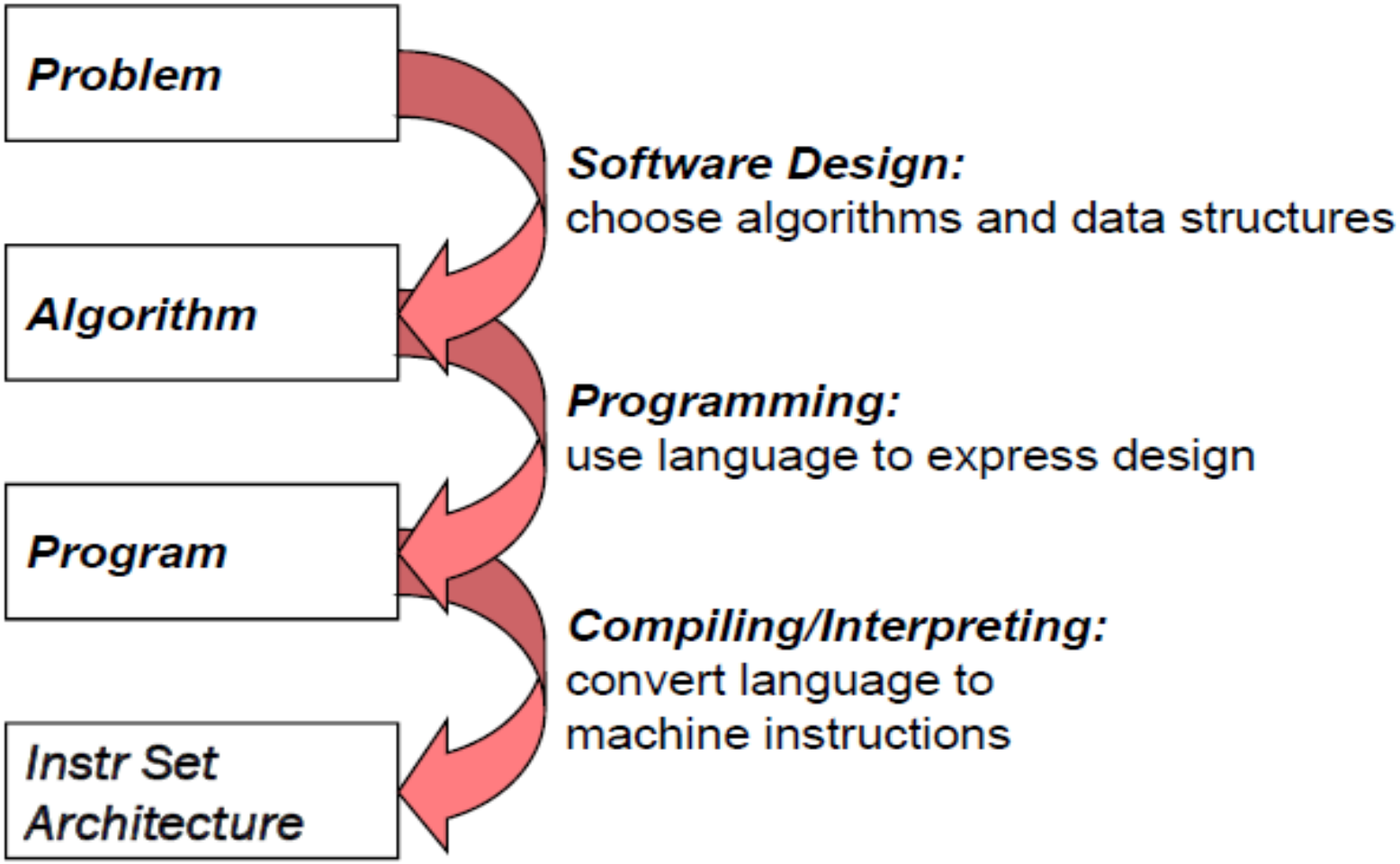
- Internal Clock Speed
- Type of Instruction Set
- Processor Implementation
- Compiler Design (efficient binary executable)
- Cache and Memory Hierarchy
- etc…

Reading Assignment: CISC & RISC, MIPS & MFLOPS

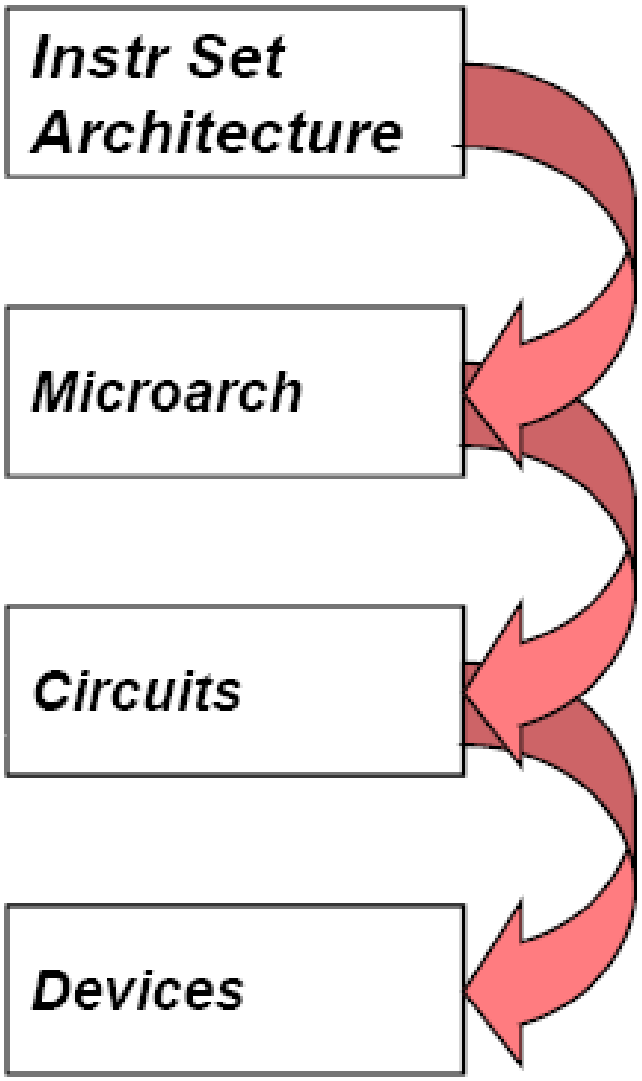
Transformations Between Layers

How do we solve a problem using a computer?

A systematic sequence of transformations between layers of abstraction.



Deeper and Deeper…



Processor Design:
choose structures to implement ISA

Logic/Circuit Design:
gates and low-level circuits to implement components

Process Engineering & Fabrication:
develop and manufacture lowest-level components

Descriptions of Each Level

Problem Statement
- stated using "natural language"
- may be ambiguous, imprecise

Algorithm
- step-by-step procedure, guaranteed to finish
- definiteness, effective computability, finiteness

Program
- express the algorithm using a computer language
- high-level language, low-level language

Instruction Set Architecture (ISA)
- specifies the set of instructions the computer can perform
- data types, addressing mode

Descriptions of Each Level (cont.)

Microarchitecture

- detailed organization of a processor implementation
- different implementations of a single ISA

Logic Circuits

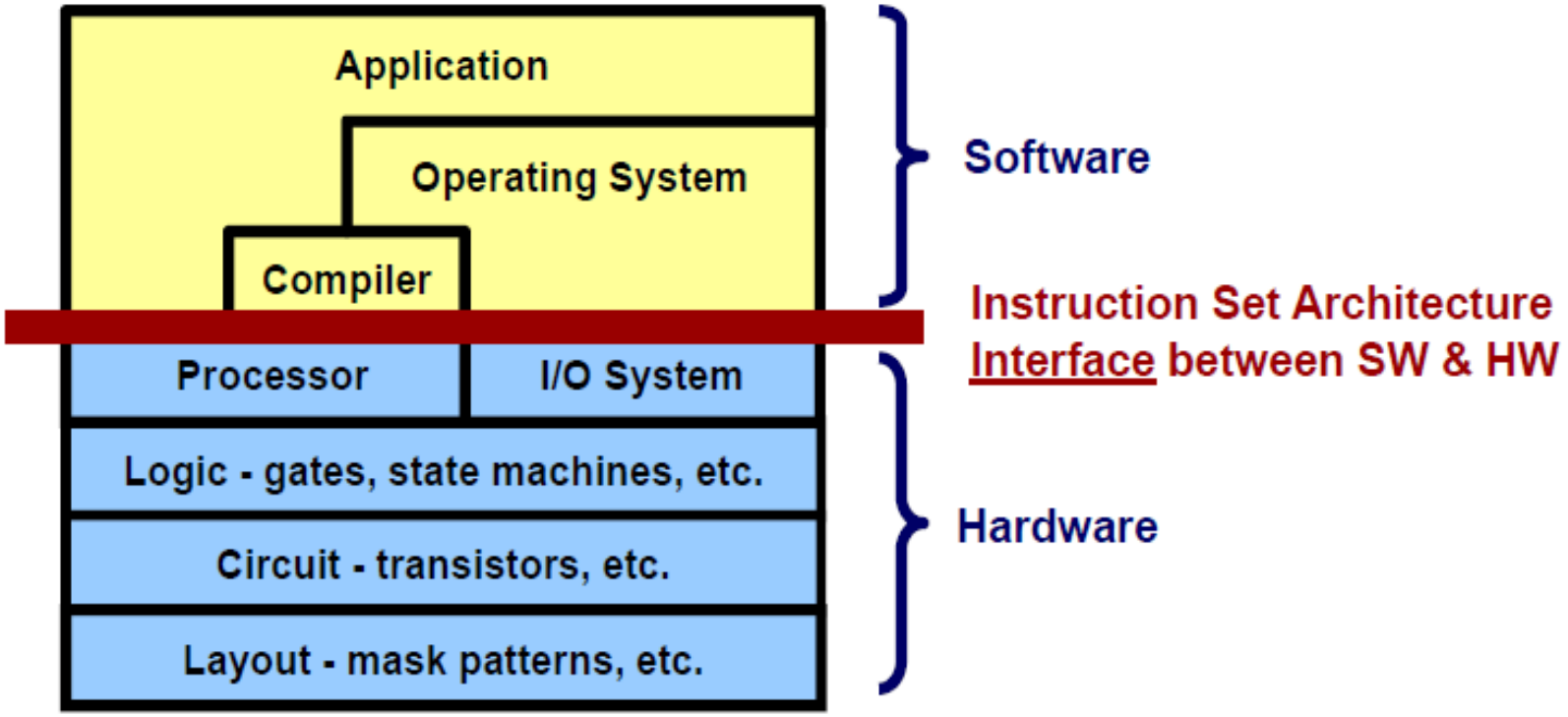
- combine basic operations to realize microarchitecture
- many different ways to implement a single function (e.g., addition)

Devices

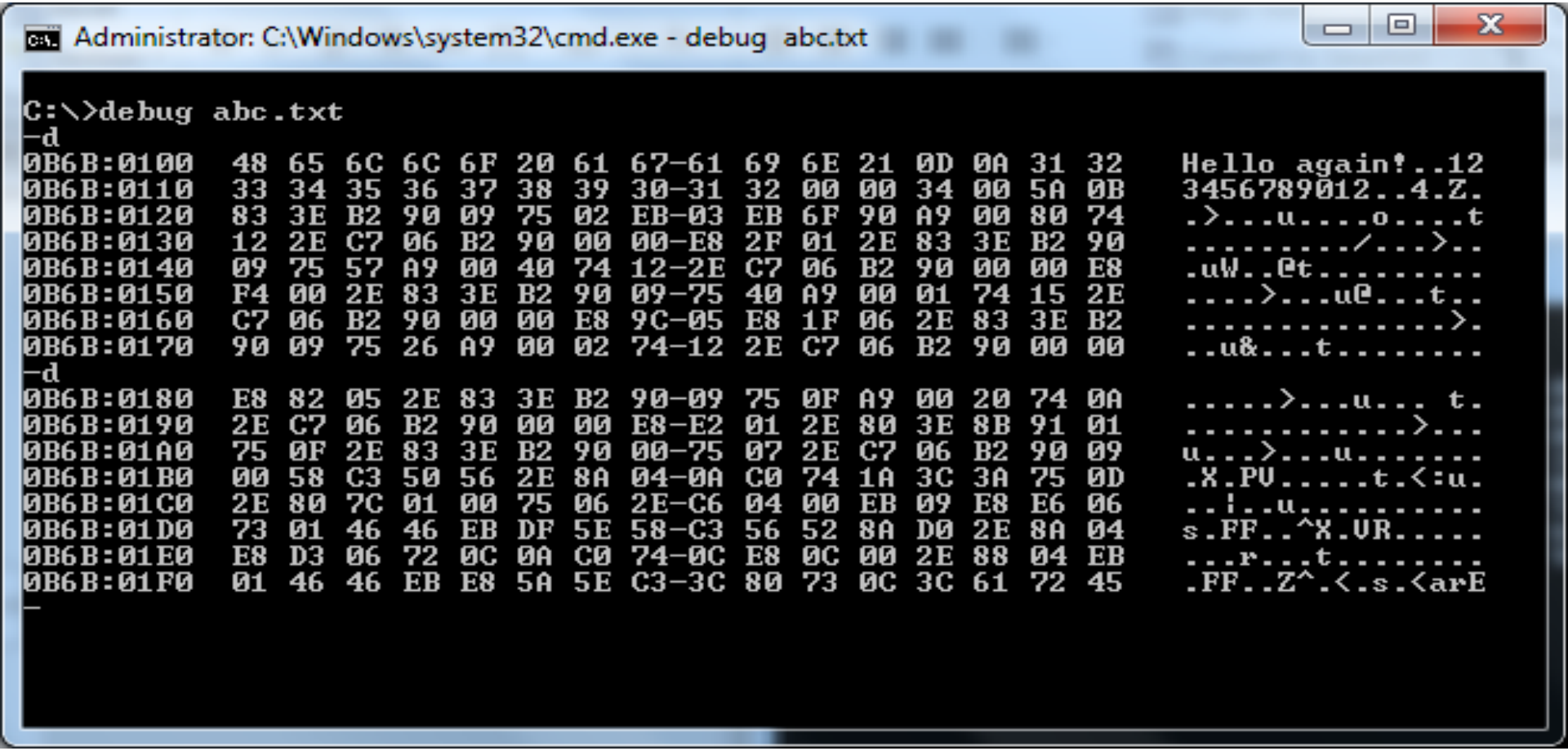
- properties of materials, manufacturability

Instruction Set Architecture (ISA) - The Hardware-Software Interface

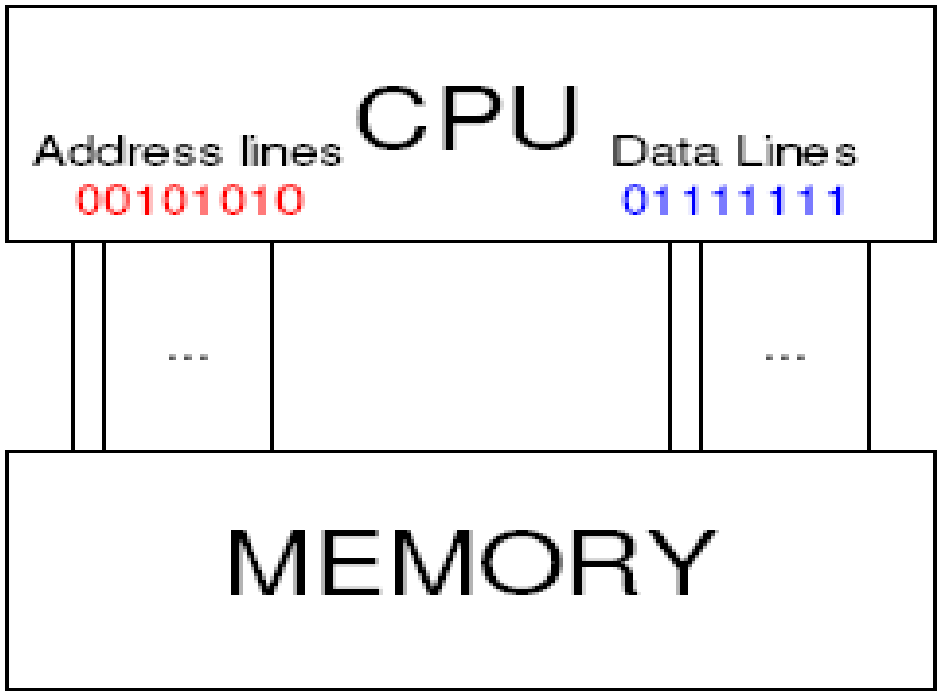
- The most important abstraction of computer design



A snapshot of memory address and content (data)

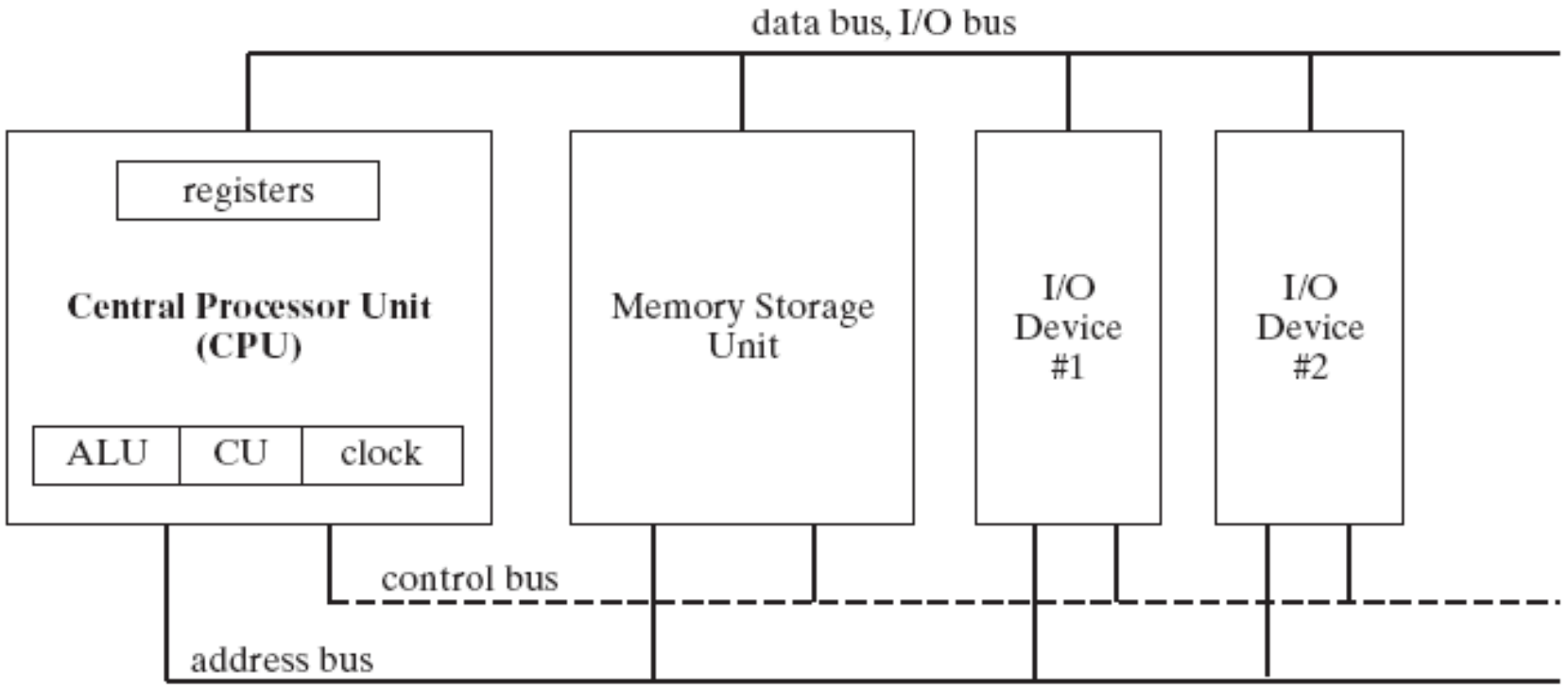


Memory makes data available if CPU issues address on the address bus



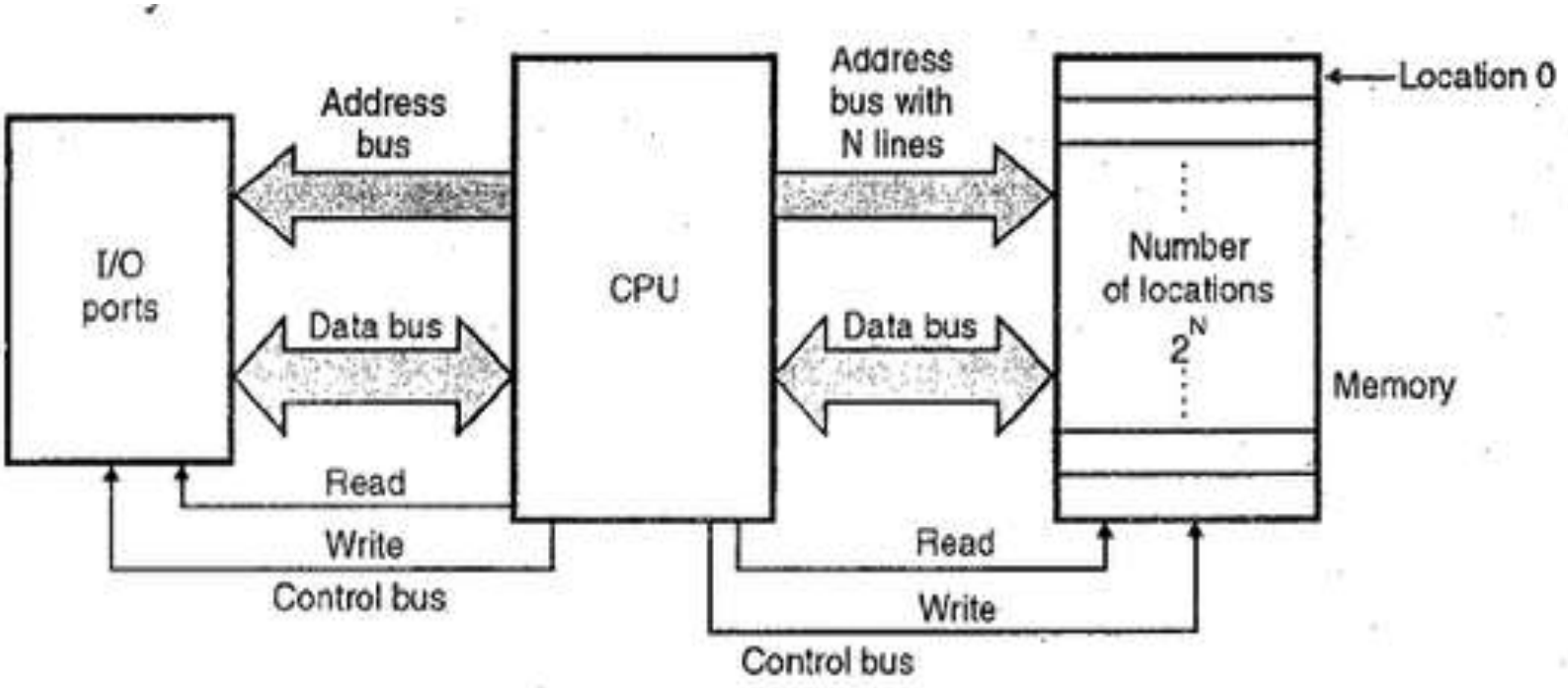
Memory Location 42 contains the number 127

Block Diagram of a Microcomputer.



- Address bus width limits the amount of memory that can be installed in the computer

- Address bus width limits the amount of memory that can be installed in the computer



The three types of buses and their utility

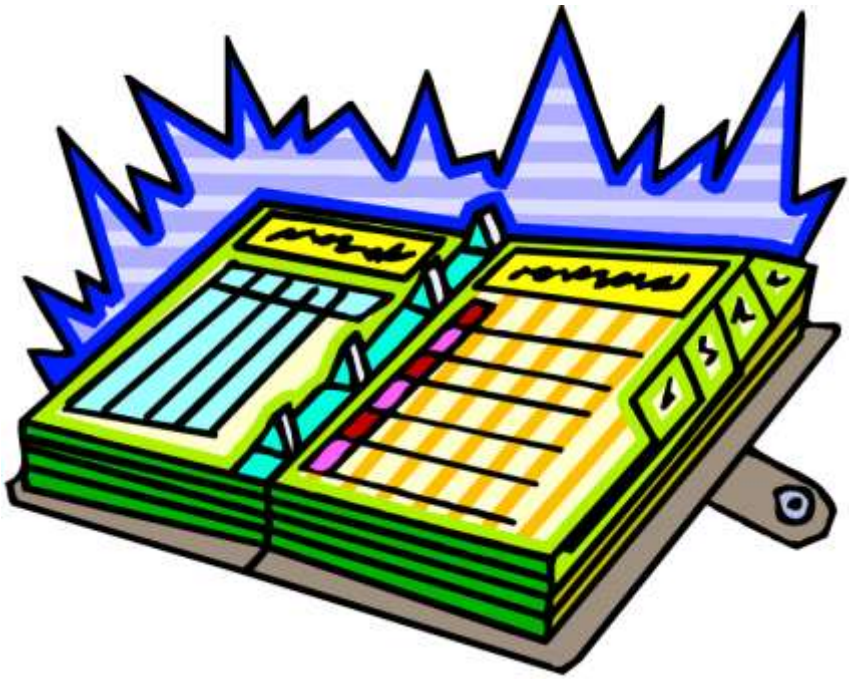
A single 1- 0 transmission is referred to as a clock cycle or bus cycle

Computer Startup

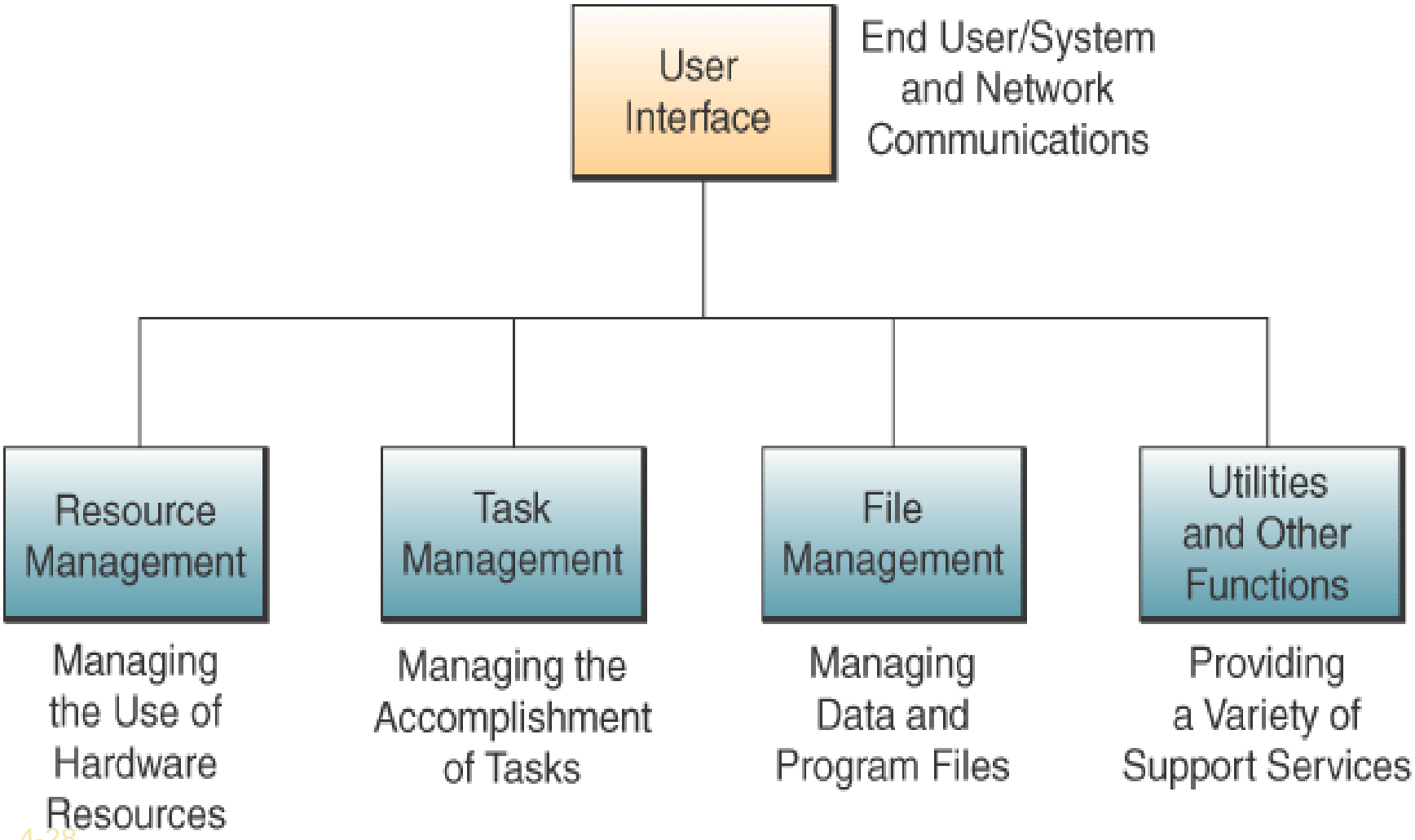
- The power is turned on
  - The Basic Input/Output System (BIOS)
    - Loads from a memory chip (ROM) and executes
    - Initializes the hardware (keyboard, disk drive, mouse, etc)
  - Then loads the operating system into memory and executes it
- The Operating System waits for user input
- The user starts a program
  - By double clicking on an icon or file
  - Or by click on Start > Program > Some Program

Operating System - Organizer

- Keep track of executing programs
  - Give them time with the CPU
  - A program gets a slice of time with the CPU
- Keep track of memory
  - Decide when to move some data to disk (virtual memory)
- Keep track of disk space
  - Decide where to store stuff
- Interface with the user
  - Accept input via keyboard and mouse
- Keep track of devices
  - USB drives, cameras, etc
- Provides networking capabilities



Operating System basic functions



Operating System

- Integrated system of programs that
  - Manages the operations of the CPU
  - Controls the input/output and storage resources and activities of the computer system
  - Provides support services as computer executes applications programs
  - Maximizes the productivity of a computer system by operating it in the most efficient manner

The Operating System and the Kernel

kernel: The operating system kernel is the part of the operating system that responds to system calls, interrupts and exceptions.

Ex. system("cls"); from C++ source, in <stdlib.h>

operating system: The operating system as a whole includes the kernel, and may include other related programs that provide services for applications.

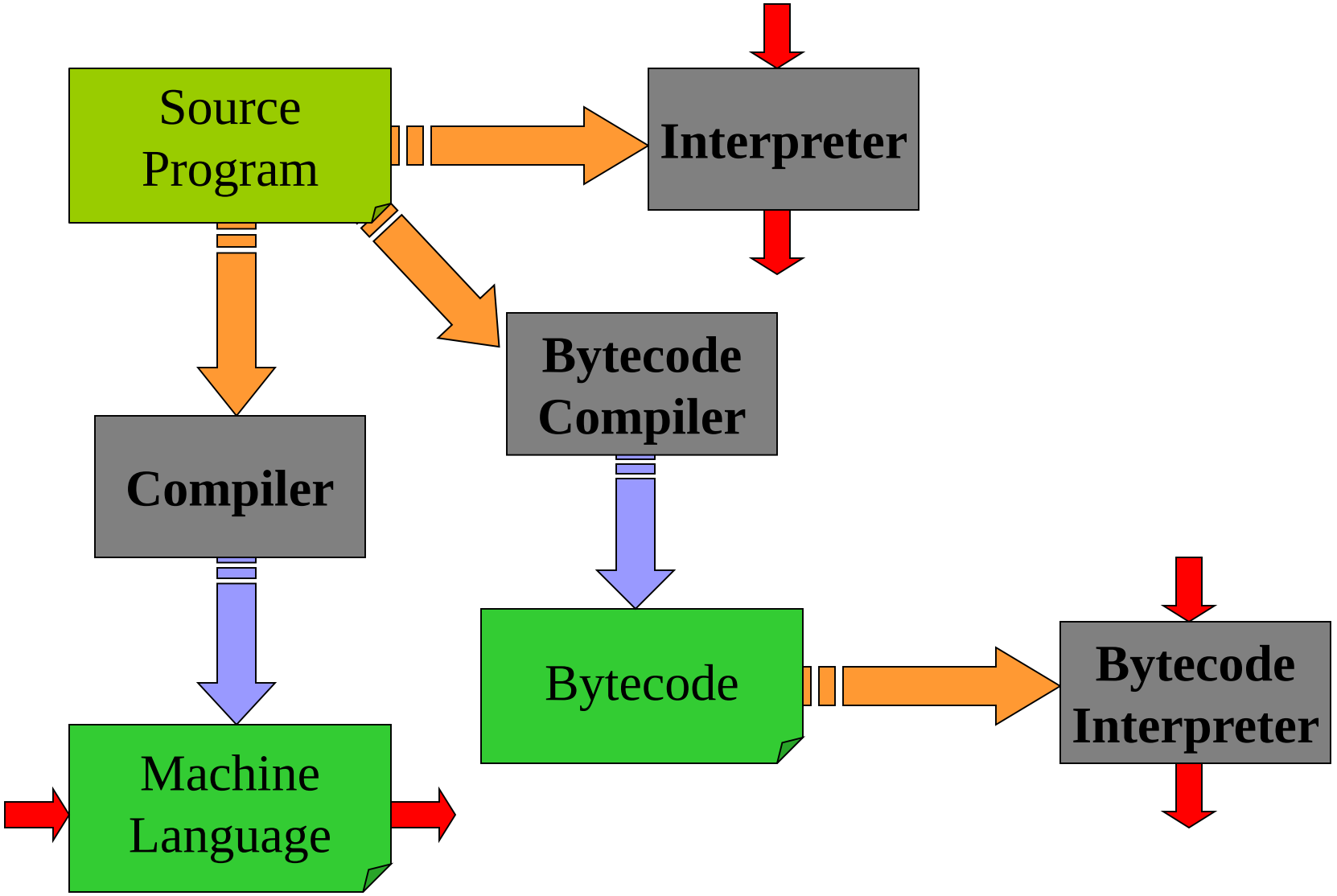
This may include things like:

- utility programs
- command interpreters
- programming libraries

Program Execution

- The operating system reads the program into memory
  - adds it to a list of programs that want to execute
- It gives it a time slice of the CPU
  - adds it to a list of programs that are executing
- It saves the current state of the CPU when it gives another program a time slice of the CPU
  - Context switching
- It can restore the state of the CPU

Programs can be executed in different ways.



Classification of programming languages

Imperative

- Procedural: C, Ada, **Pascal**, **Algol**, **FORTRAN**, . . .
- Object oriented: **Scala**, C#, Java, **Smalltalk**, **SIMULA**, . . .
- Scripting: Perl, Python, PHP, . . .

Declarative

- Functional: Haskell, SML, Lisp, Scheme, . . .
- Logic: Prolog
- Dataflow: Id, Val
- Constraint-based: spreadsheets
- Template-based: XSLT

Why are there so many languages?

- Evolution.
- Special purposes.
- Personal preference.

Assignment:

Implement Heron's algorithm in DOS.

Programming Languages

Four Levels of Programming Languages	
• **Machine Languages:** Use binary coded instructions 1010 11001 1011 11010 1100 11011	• **High-Level Languages:** Use brief statements or arithmetic notations BASIC: X = Y + Z COBOL: COMPUTE X = Y + Z
• **Assembler Languages:** Use symbolic coded instructions LOD Y ADD Z STR X	• **Fourth-Generation Languages:** Use natural and nonprocedural statements SUM THE FOLLOWING NUMBERS

Machine Languages

- First-generation languages
- All program instructions had to be written using binary codes unique to each computer
- Programmers had to know the internal operations of the specific type of CPU

Assembler Languages

- Second-generation languages
- Symbols are used to represent operation codes and storage locations
- Need language translator programs to convert the instructions into machine instructions
- Used by systems programmers (who program system software)

High-Level Languages

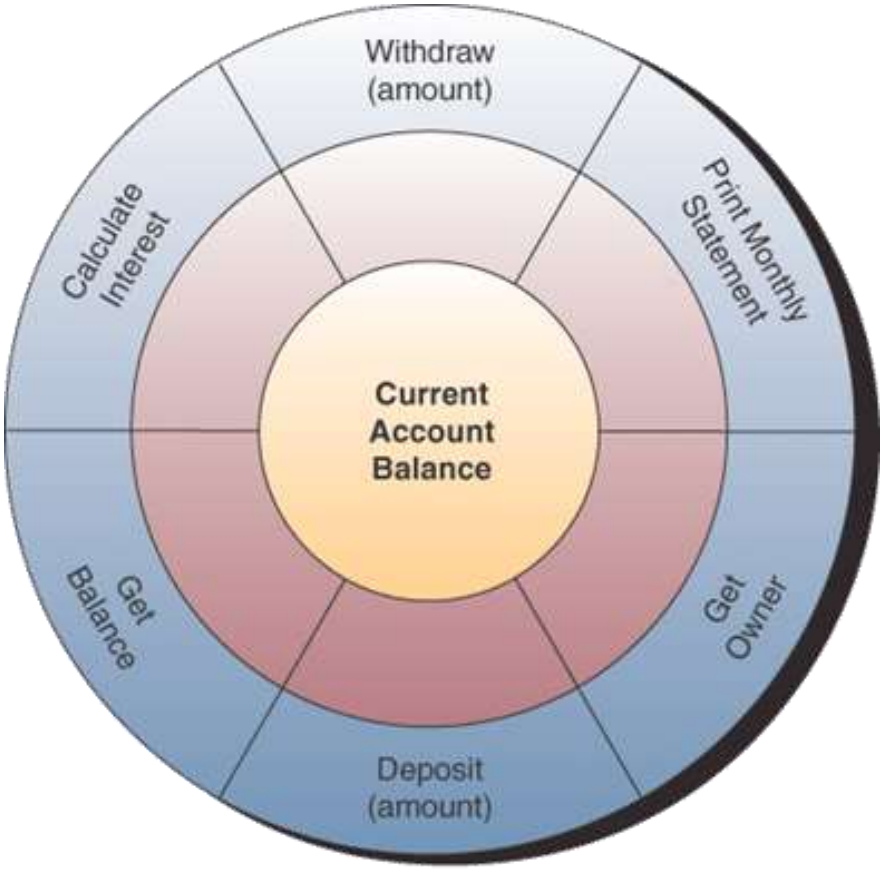
- Third-generation languages
- Instructions that use brief statements or arithmetic expressions
- Macroinstructions: each statement generates several machine instructions when translated by compilers or interpreters
- Easier to learn than assembler
- Machine independent
- Less efficient than assembler
- COBOL is an example of a High-Level Language

Fourth-Generation Languages

- Variety of programming languages that are nonprocedural and conversational
- Nonprocedural – users specify results they want while computer determines the sequence of instructions that will accomplish those results
- Natural Language – very close to English or other human language

Object-Oriented Languages

- Combine data elements and the procedures that will be performed upon them into Objects
- E.g., an object could be data about a bank account and the procedures performed on it such as interest calculations



Object-Oriented Languages

- Most widely used software development languages today
- Easier to use and more efficient for graphics-oriented user interfaces
- Reusable: can use an object from one application in another application
- E.g., Visual Basic, C++, Java

Web Languages

- HTML
  - A page description language that creates hypertext documents for the Web
- XML
  - Describes the contents of Web pages by applying identifying tags or contextual labels to the data in Web documents
- Java
  - Object-oriented programming language that is simple, secure and platform independent
  - Java applets can be executed on any computer