Introduction

COE 301

Computer Organization

Dr. Muhamed Mudawar

Computer Engineering Department

King Fahd University of Petroleum and Minerals

Presentation Outline

- ❖ **Welcome to COE 301**
- ❖ Assembly-, Machine-, and High-Level Languages
- ❖ Classes of Computers
- ❖ Programmer's View of a Computer System

Welcome to COE 301

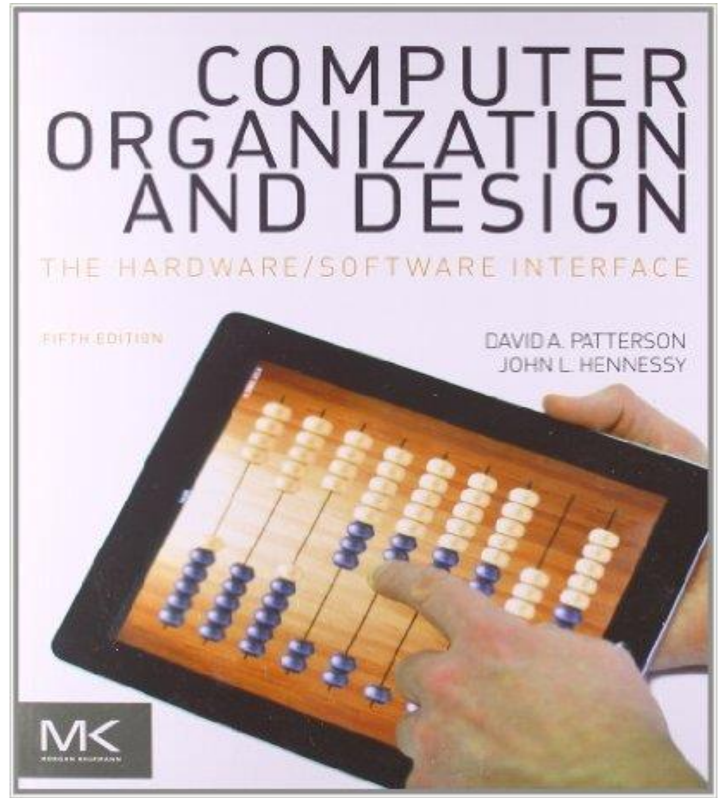
- Instructor: Dr. Muhamed F. Mudawar
- Office: Building 22, Room 410-2
- Office Phone: 4642
- Schedule and Office Hours:
  - http://faculty.kfupm.edu.sa/coe/mudawar/schedule/
- Course Web Page:
  - http://faculty.kfupm.edu.sa/coe/mudawar/coe301/
- Email:
  - mudawar@kfupm.edu.sa

Which Textbook will be Used?

- Computer Organization & Design:

  The Hardware/Software Interface

  - Fifth Edition, 2013
  - David Patterson and John Hennessy
  - Morgan Kaufmann



- Read the textbook in addition to slides

Grading Policy

❖ Quizzes	10%
❖ MIPS Programming	10%
❖ Lab Work	15%
❖ CPU Design Project	15%
❖ Midterm Exam	25%
❖ Final Exam	25%

❖ No makeup will be given for missing exam or quiz

Software Tools

- MIPS Simulators
  - MARS: MIPS Assembly and Runtime Simulator
    - Runs MIPS-32 assembly language programs
    - Website: http://courses.missouristate.edu/KenVollmar/MARS/
  - SPIM
    - Also Runs MIPS-32 assembly language programs
    - Website: http://www.cs.wisc.edu/~larus/spim.html
- CPU Design and Simulation Tool
  - Logisim
    - Educational tool for designing and simulating CPUs
    - Website: http://ozark.hendrix.edu/~burch/logisim/

Course Learning Outcomes

- Towards the end of this course, you should be able to …
  - Describe the instruction set architecture of a processor
  - Analyze, write, and test assembly language programs
  - Describe organization/operation of integer & floating-point units
  - Design the datapath and control of a single-cycle CPU
  - Design the datapath/control of a pipelined CPU & handle hazards
  - Describe the organization/operation of memory and caches
  - Analyze the performance of processors and caches
- Required Background
  - Ability to program confidently in Java or C
  - Ability to design a combinational and sequential circuit

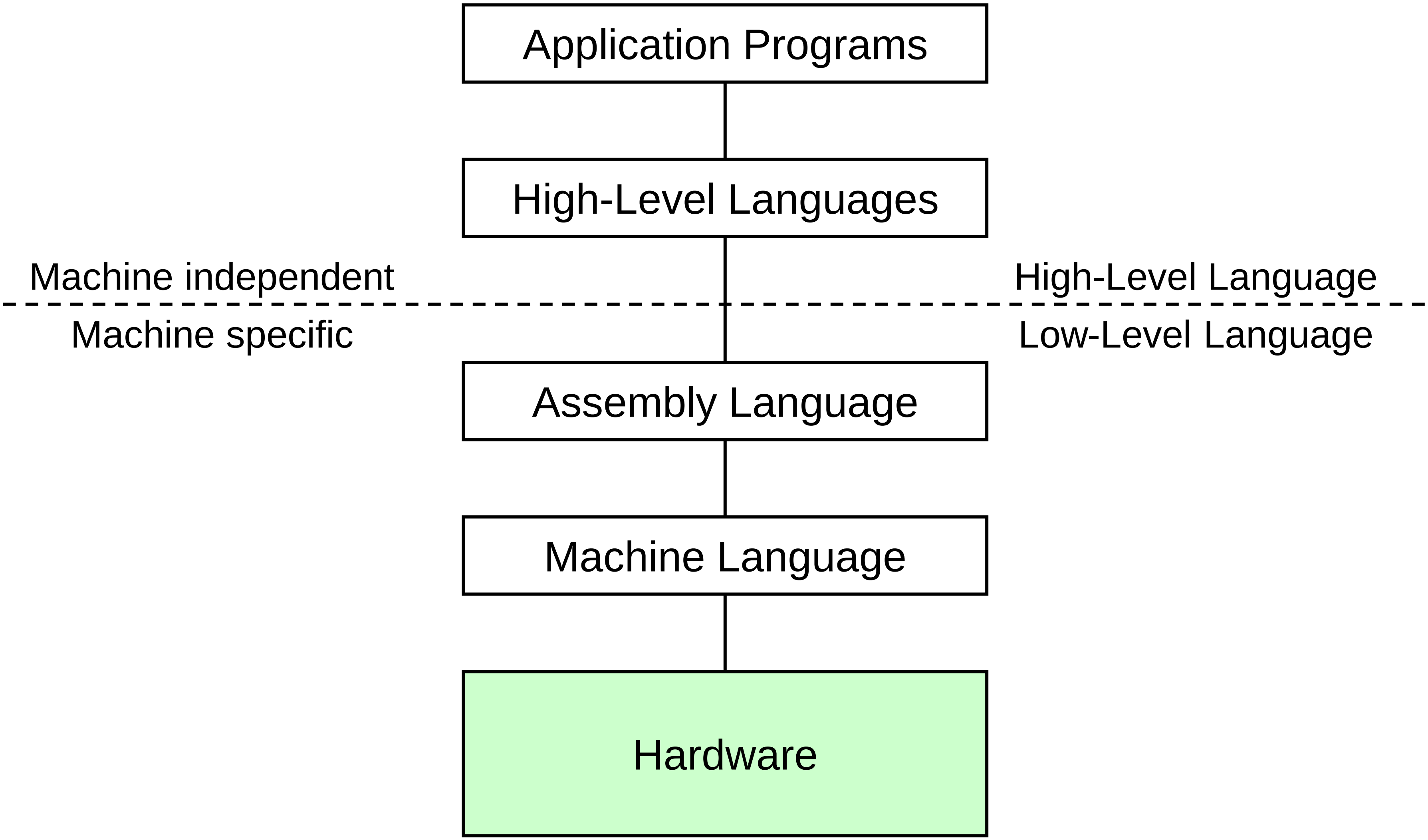
Next . . .

- Welcome to COE 301
- **Assembly-, Machine-, and High-Level Languages**
- Classes of Computers
- Programmer's View of a Computer System

Some Important Questions to Ask

- What is Assembly Language?
- What is Machine Language?
- How is Assembly related to a high-level language?
- Why Learn Assembly Language?
- What is an Assembler, Linker, and Debugger?

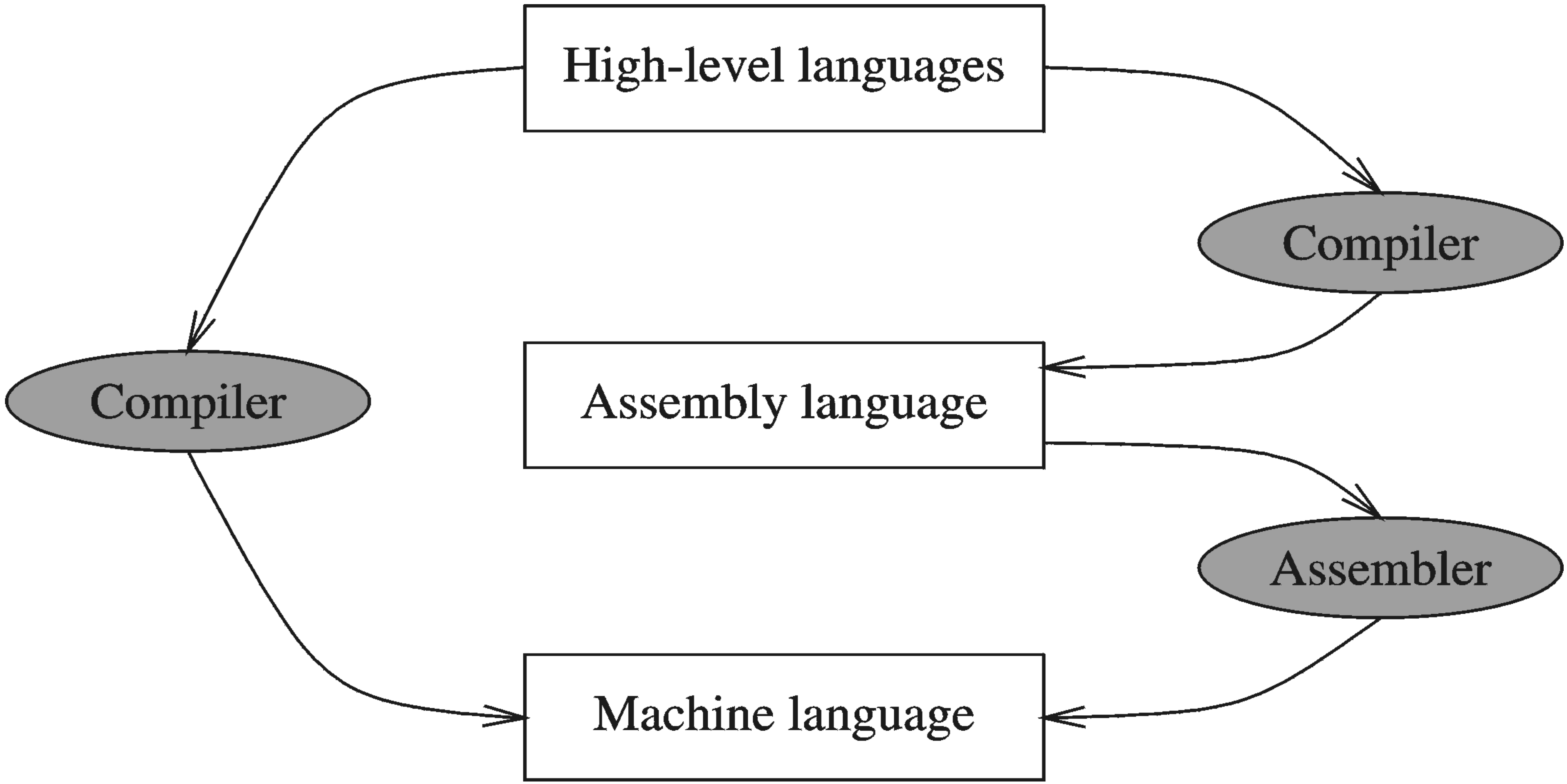
A Hierarchy of Languages



Assembly and Machine Language

- Machine language
  - Native to a processor: executed directly by hardware
  - Instructions consist of binary code: 1s and 0s
- Assembly language
  - Slightly higher-level language
  - Readability of instructions is better than machine language
  - One-to-one correspondence with machine language instructions
- Assemblers translate assembly to machine code
- Compilers translate high-level programs to machine code
  - Either directly, or
  - Indirectly via an assembler

Compiler and Assembler



Translating Languages

Program (C Language):

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

Compiler

MIPS Assembly Language:

```
sll $2,$5, 2
add $2,$4,$2
lw  $15,0($2)
lw  $16,4($2)
sw  $16,0($2)
sw  $15,4($2)
jr  $31
```

Assembler

MIPS Machine Language:

```
00051080
00821020
8C620000
8CF20004
ACF20000
AC620004
03E00008
```

A statement in a high-level language is translated typically into several machine-level instructions

Advantages of High-Level Languages

- Program development is faster
  - High-level statements: fewer instructions to code
- Program maintenance is easier
  - For the same above reasons
- Programs are portable
  - Contain few machine-dependent details
    - Can be used with little or no modifications on different machines
  - Compiler translates to the target machine language
  - However, Assembly language programs are not portable

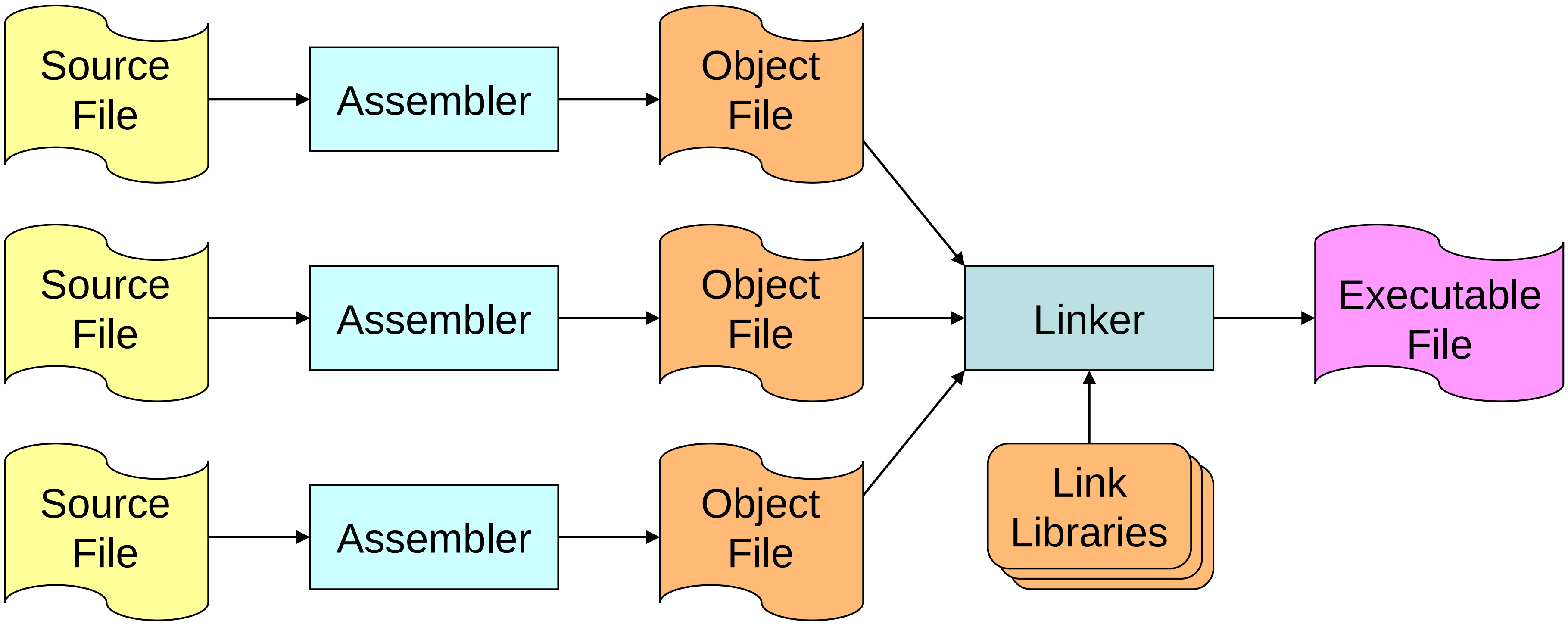
Why Learn Assembly Language?

- Many reasons:
  - Accessibility to system hardware
  - Space and time efficiency
  - Writing a compiler for a high-level language
- Accessibility to system hardware
  - Assembly Language is useful for implementing system software
  - Also useful for small embedded system applications
- Programming in Assembly Language is harder
  - Requires deep understanding of the processor architecture
  - However, it is very rewarding to system software designers
  - Adds a new perspective on how programs run on real processors

Assembly Language Programming Tools

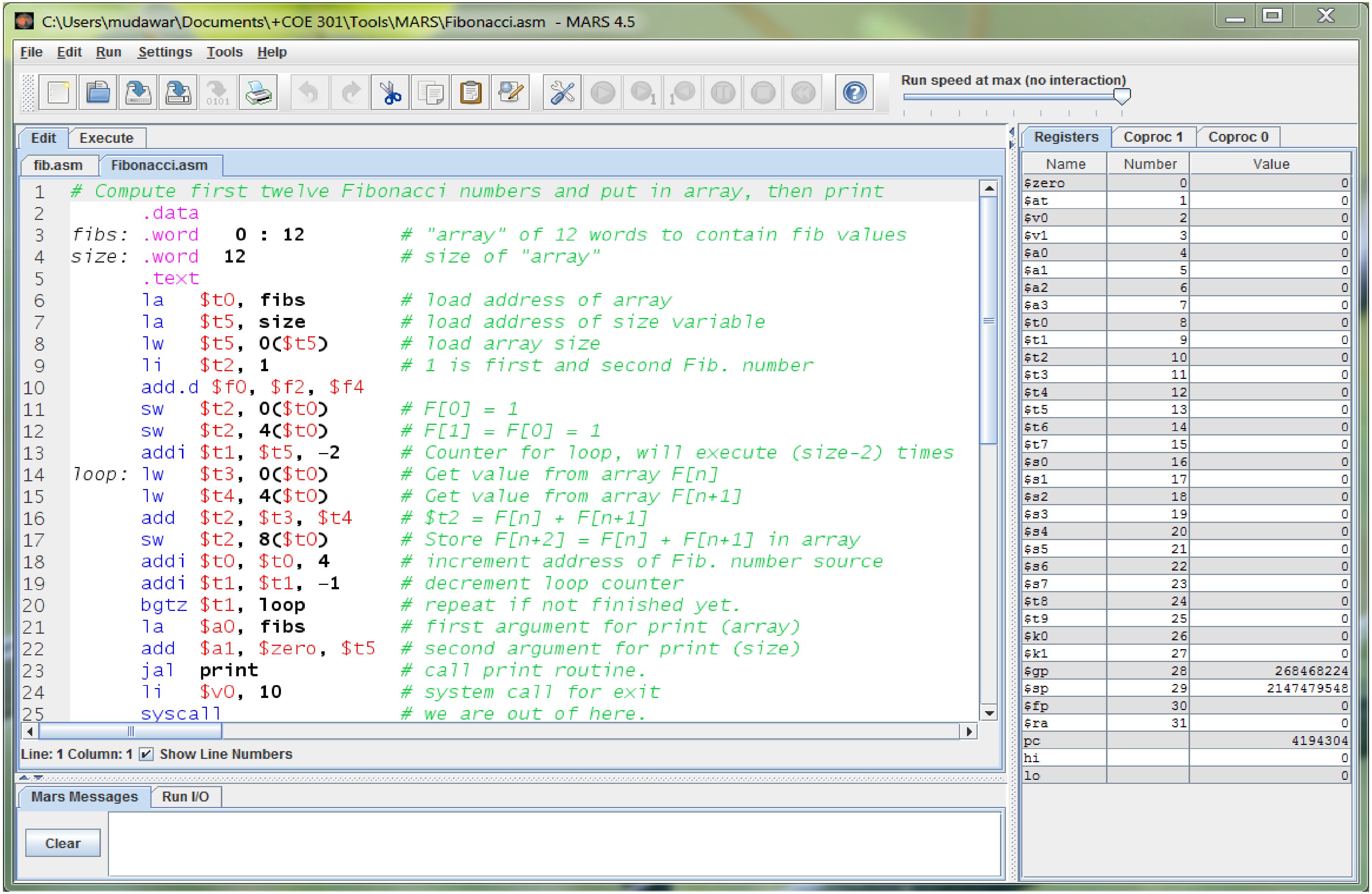
- Editor
  - Allows you to create and edit assembly language source files
- Assembler
  - Converts assembly language programs into object files
  - Object files contain the machine instructions
- Linker
  - Combines object files created by the assembler with link libraries
  - Produces a single executable program
- Debugger
  - Allows you to trace the execution of a program
  - Allows you to view machine instructions, memory, and registers

Assemble and Link Process



- A program may consist of multiple source files
- Assembler translates each source file into an object file
- Linker links all object files together and with link libraries
- The result executable file can run directly on the processor

MARS Assembler and Simulator Tool



MARS Assembler and Simulator Tool

- Simulates the execution of a MIPS program
  - No direct execution on the underlying Intel processor
- Editor with color-coded assembly syntax
  - Allows you to create and edit assembly language source files
- Assembler
  - Converts MIPS assembly language programs into object files
- Console and file input/output using system calls
- Debugger
  - Allows you to trace the execution of a program and set breakpoints
  - Allows you to view machine instructions, edit registers and memory
- Easy to use and learn assembly language programming

Next . . .

- Welcome to COE 301
- Assembly-, Machine-, and High-Level Languages
- **Classes of Computers**
- Programmer's View of a Computer System

Classes of Computers

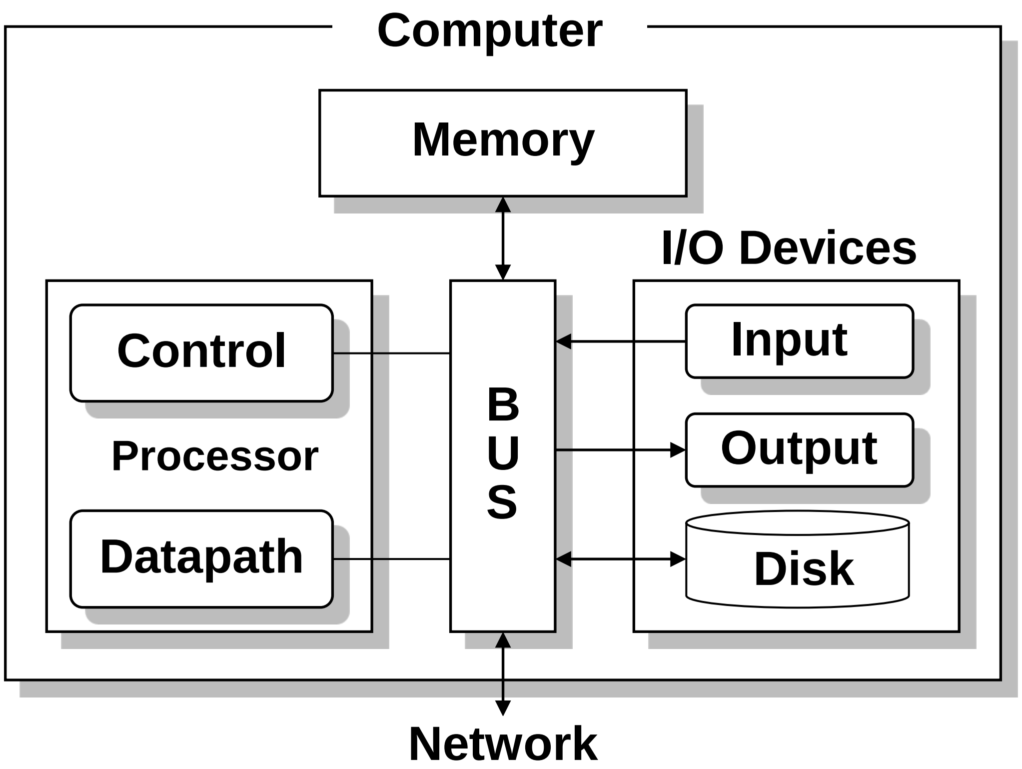
- Personal computers
  - General purpose, variety of software, subject to cost/performance
- Server computers
  - Network based, high capacity, performance, and reliability
  - Range from small servers to building sized
- Supercomputers
  - High-end scientific and engineering calculations
  - Highest capability but only a small fraction of the computer market
- Embedded computers
  - Hidden as components of systems
  - Stringent power/performance/cost constraints

Classes of Computers (cont'd)

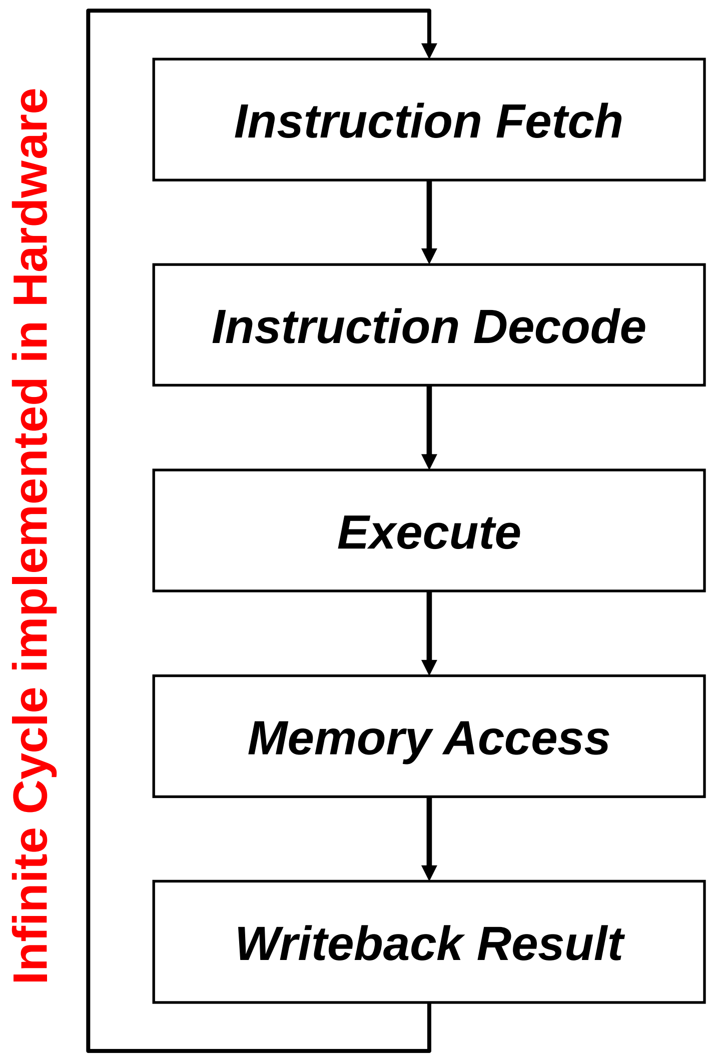
- Personal Mobile Device (PMD)
  - Battery operated
  - Connects to the Internet
  - Low price: hundreds of dollars
  - Smart phones, tablets, electronic glasses
- Cloud Computing
  - Warehouse Scale Computers (WSC)
  - Software, Platform, and Infrastructure as a Service
  - However, security concerns of storing "sensitive data" in "the cloud"
  - Examples: Amazon and Google

Components of a Computer System

- Processor
  - Datapath and Control
- Memory & Storage
  - Main Memory
  - Disk Storage
- Input / Output devices
  - User-interface devices
  - Network adapters
    - For communicating with other computers
- Bus: Interconnects processor to memory and I/O
- Essentially the same components for all kinds of computers



Fetch - Execute Cycle



Fetch instruction

Compute address of next instruction

Generate control signals for instruction

Read operands from registers

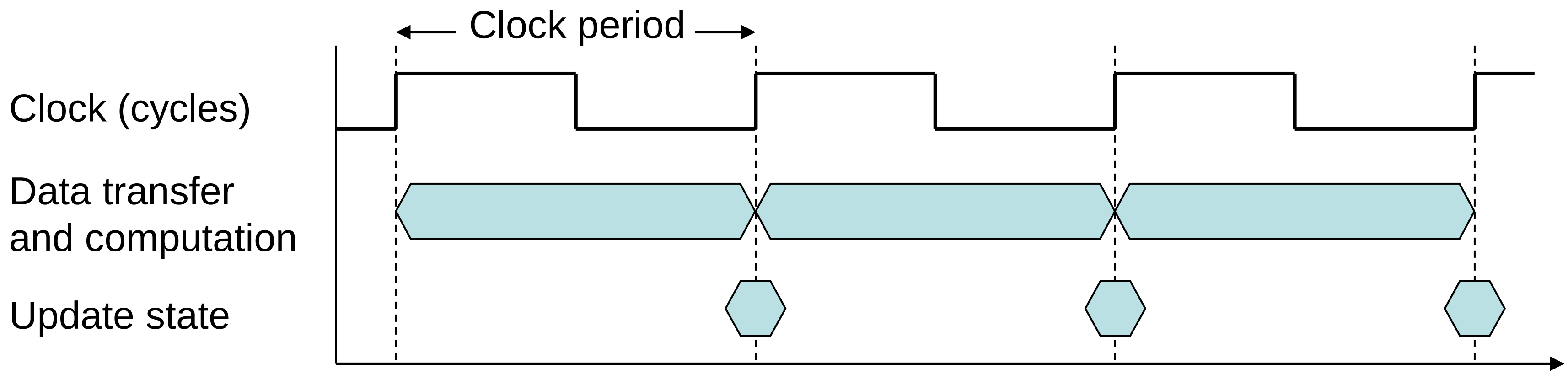
Compute result value

Read or write memory

Writeback result in a register

Clock

Operation of digital hardware is governed by a clock



- Clock period: duration of a clock cycle
  - e.g., 250 ps = 0.25 ns = 0.25 $\times 10^{-9}$ sec
- Clock frequency (rate) = 1 / clock period
  - e.g., 1/ 0.25 $\times 10^{-9}$ sec = 4.0$\times 10^{9}$ Hz = 4.0 GHz

Memory and Storage Devices

- Volatile Memory Devices

  - RAM = Random Access Memory
  - DRAM = Dynamic RAM
    - Dense but must be refreshed (typical choice for main memory)
  - SRAM: Static RAM
    - Faster but less dense than DRAM (typical choice for cache memory)
- Non-Volatile Storage Devices

  - Magnetic Disk
  - Flash Memory (Solid State Disk)
  - Optical Disk (CDROM, DVD)

Units for Storage and Memory

Decimal term	Abbreviation	Value	Binary term	Abbreviation	Value	% Larger
kilobyte	KB	10^3	kibibyte	KiB	2^{10}	2%
megabyte	MB	10^6	mebibyte	MiB	2^{20}	5%
gigabyte	GB	10^9	gibibyte	GiB	2^{30}	7%
terabyte	TB	10^{12}	tebibyte	TiB	2^{40}	10%
petabyte	PB	10^{15}	pebibyte	PiB	2^{50}	13%
exabyte	EB	10^{18}	exbibyte	EiB	2^{60}	15%
zettabyte	ZB	10^{21}	zebibyte	ZiB	2^{70}	18%
yottabyte	YB	10^{24}	yobibyte	YiB	2^{80}	21%

Size of disk storage

Value = 10^n (base 10)

Size of memory

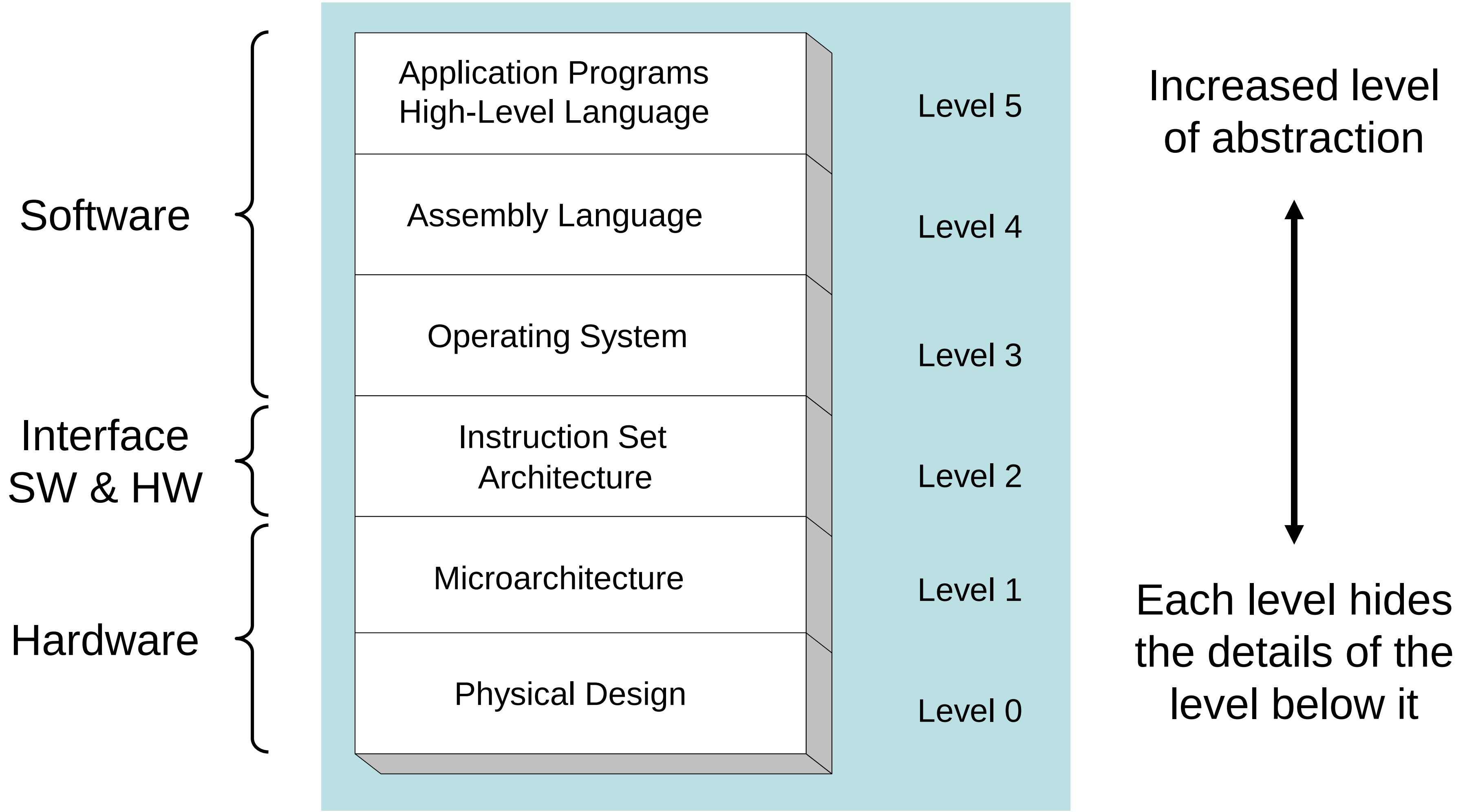
Value = 2^n (base 2)

❖ The binary terms are used to avoid the confusion with the commonly used decimal terms. The size of memory is 2^n because the memory address is an *n*-bit binary number.

Next . . .

- ❖ Welcome to COE 301
- ❖ Assembly-, Machine-, and High-Level Languages
- ❖ Classes of Computers
- ❖ **Programmer's View of a Computer System**

Programmer's View of a Computer System



Programmer's View (cont'd)

- Application Programs (Level 5)
  - Written in high-level programming languages
  - Such as Java, C++, Pascal, Visual Basic . . .
  - Programs compile into assembly language level (Level 4)
- Assembly Language (Level 4)
  - Instruction mnemonics (symbols) are used
  - Have one-to-one correspondence to machine language
  - Calls functions written at the operating system level (Level 3)
  - Programs are translated into machine language (Level 2)
- Operating System (Level 3)
  - Provides services to level 4 and 5 programs
  - Translated to run at the machine instruction level (Level 2)

Programmer's View (cont'd)

- Instruction Set Architecture (Level 2)
  - Interface between software and hardware
  - Specifies how a processor functions
  - Machine instructions, registers, and memory are exposed
  - Machine language is executed by Level 1 (microarchitecture)
- Microarchitecture (Level 1)
  - Controls the execution of machine instructions (Level 2)
  - Implemented by digital logic
- Physical Design (Level 0)
  - Implements the microarchitecture at the transistor-level
  - Physical layout of circuits on a chip