

## Chapter 2: x86 Processor Architecture

Kip Irvine

(c) Pearson Education, 2015. All rights reserved. You may modify and copy this slide show for your personal use, or for use in the classroom, as long as this copyright statement, the author's name, and the title are not changed.

## Chapter Overview

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-bit Processors
- Components of an IA-32 Microcomputer
- Input-Output System

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

3

## General Concepts

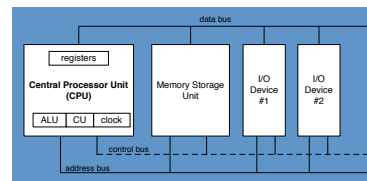
- Basic microcomputer design
- Instruction execution cycle
- Reading from memory
- How programs run

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

4

## Basic Microcomputer Design

- clock synchronizes CPU operations
- control unit (CU) coordinates sequence of execution steps
- ALU performs arithmetic and bitwise processing

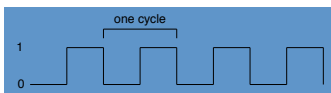


Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

5

## Clock

- synchronizes all CPU and BUS operations
- machine (clock) cycle measures time of a single operation
- clock is used to trigger events



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

6

## What's Next

- General Concepts
- **IA-32 Processor Architecture**
- IA-32 Memory Management
- 64-Bit Processors
- Components of an IA-32 Microcomputer
- Input-Output System

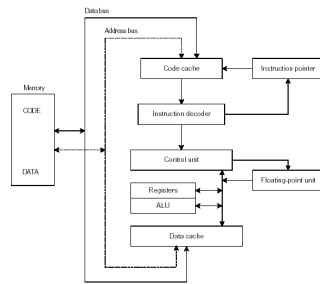
Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

7

## Instruction Execution Cycle

Figure 2-2 Simplified Pentium CPU Block Diagram.

- Fetch
- Decode
- Fetch operands
- Execute
- Store output



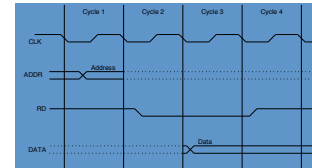
Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

8

## Reading from Memory

Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:

1. Place the address of the value you want to read on the address bus.
2. Assert (changing the value of) the processor's RD (read) pin.
3. Wait one clock cycle for the memory chips to respond.
4. Copy the data from the data bus into the destination operand



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

9

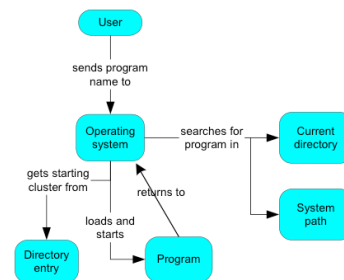
## Cache Memory

- High-speed expensive static RAM both inside and outside the CPU.
  - Level-1 cache: inside the CPU
  - Level-2 cache: outside the CPU
- Cache hit: when data to be read is already in cache memory
- Cache miss: when data to be read is not in cache memory.

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

10

## How a Program Runs



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

11

## Multitasking

- OS can run multiple programs at the same time
- Multiple threads of execution within the same program
- Scheduler utility assigns a given amount of CPU time to each running program
- Rapid switching of tasks
  - gives illusion that all programs are running at once
  - the processor must support task switching
- Use as basis for server virtualization

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

12

## IA-32 Processor Architecture

- Modes of operation
- Basic execution environment
- Floating-point unit
- Intel Microprocessor history

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

13

## Modes of Operation

- Protected mode (programs given separate memory areas)
  - native mode (Windows, Linux)
- Real-address mode (implements programming environment of Intel 8086 processor)
  - native MS-DOS
- System management mode (provides OS)
  - power management, system security, diagnostics

- Virtual-8086 mode (server virtualization)
  - hybrid of Protected
  - each program has its own 8086 computer

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

14

## Basic Execution Environment

- Addressable memory
- General-purpose registers
- Index and base registers
- Specialized register uses
- Status flags
- Floating-point, MMX, XMM registers

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

15

## Addressable Memory

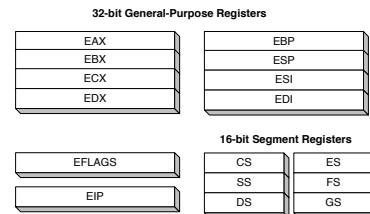
- Protected mode
  - 4 GB
  - 32-bit address (4,294,967,295)
- Real-address and Virtual-8086 modes
  - 1 MB space
  - 20-bit address (1,048,575)
  - In protected mode running multiple programs, each program has its own 1 MB memory area

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

16

## General-Purpose Registers

Named storage locations inside the CPU, optimized for speed

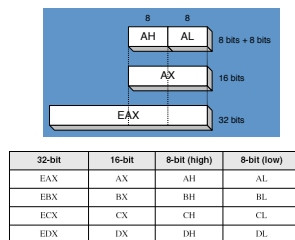


Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

17

## Accessing Parts of Registers

- Use 8-bit name, 16-bit name, or 32-bit name
- Applies to EAX, EBX, ECX, and EDX



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

18

## Index and Base Registers

- Some registers have only a 16-bit name for their lower half:

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

19

## Some Specialized Register Uses (1 of 2)

- General-Purpose
  - EAX – accumulator
  - ECX – loop counter
  - ESP – stack pointer
  - ESI, EDI – index registers
  - EBP – extended frame pointer (stack)
- Segment
  - CS – code segment
  - DS – data segment
  - SS – stack segment
  - ES, FS, GS – additional segments

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

20

## Some Specialized Register Uses (2 of 2)

- EIP – instruction pointer
  - Contains address of next instruction to be executed
- EFLAGS
  - status and control flags
  - each flag is a single binary bit
  - A flag is set when it equals 1; it is clear (or reset) when it equals 0

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

21

## Status Flags

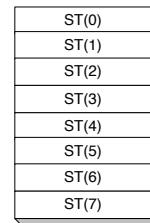
- Carry
  - unsigned arithmetic out of range
- Overflow
  - signed arithmetic out of range
- Sign
  - result is negative
- Zero
  - result is zero
- Auxiliary Carry
  - carry from bit 3 to bit 4
- Parity
  - sum of 1 bits is an even number

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

22

## Floating-Point, MMX, XMM Registers

- Eight 80-bit floating-point data registers
  - ST(0), ST(1), . . . , ST(7)
  - arranged in a stack
  - used for all floating-point arithmetic
- Eight 64-bit MMX registers
- Eight 128-bit XMM registers for single-instruction multiple-data (SIMD) operations



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

23

## Intel Microprocessor History

- Intel 8086, 80286
- IA-32 processor family
- P64 processor family
- CISC and RISC

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

24

## Early Intel Microprocessors

- Intel 8080
  - 64K addressable RAM
  - 8-bit registers
  - CP/M operating system
  - S-100 BUS architecture
  - 8-inch floppy disks!
- Intel 8086/8088
  - IBM-PC Used 8088
  - 1 MB addressable RAM
  - 16-bit registers
  - 16-bit data bus (8-bit for 8088)
  - separate floating-point unit (8087)

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

25

## The IBM-AT (Advance Technology)

- Intel 80286
  - 16 MB addressable RAM
  - Protected memory
  - several times faster than 8086
  - introduced IDE (Integrated Drive Electronics) bus architecture
  - 80287 floating point unit



Invine, Kip R. Assembly Language  
for x86 Processors 6/e, 2010.

26

## Intel IA-32 Family

- Intel386
  - 4 GB addressable RAM, 32-bit registers, paging (virtual memory)
- Intel486
  - instruction pipelining
- Pentium
  - superscalar, 32-bit address bus, 64-bit internal data path

Invine, Kip R. Assembly Language  
for x86 Processors 6/e, 2010.

27

## 64-bit Processors

- Intel64
  - 64-bit linear address space
  - Intel: Pentium Extreme, Xeon, Celeron D, Pentium D, Core 2, and Core i7
- IA-32e Mode
  - Compatibility mode for legacy 16- and 32-bit applications
  - 64-bit Mode uses 64-bit addresses and operands

Invine, Kip R. Assembly Language  
for x86 Processors 6/e, 2010.

28

## Intel Technologies

- HyperThreading technology
  - two tasks execute on a single processor at the same time
- Multi Core processing
  - multiple processor cores in the same IC package
  - each processor has its own resources and communication path with the bus

Invine, Kip R. Assembly Language  
for x86 Processors 6/e, 2010.

29

## Intel Processor Families

Currently Commonly Used:

- Pentium & Celeron – dual core
- Core 2 Duo - 2 processor cores
- Core 2 Quad - 4 processor cores
- Core i5 and i7 – 4 processor cores
- Xeon - up to 64 processor cores (for server machines)
- Intel Microprocessor Reference Guide
  - <http://www.intel.com/pressroom/kits/quickreffam.htm>

Invine, Kip R. Assembly Language  
for x86 Processors 6/e, 2010.

30

## CISC and RISC

- CISC – complex instruction set
  - large instruction set
  - high-level operations
  - requires microcode interpreter
  - examples: Intel 80x86 family
- RISC – reduced instruction set
  - simple, atomic instructions
  - small instruction set
  - directly executed by hardware
  - examples:
    - ARM (Advanced RISC Machines)
    - DEC Alpha (now Compaq)
    - Sun Sparc (now Oracle)

Invine, Kip R. Assembly Language  
for x86 Processors 6/e, 2010.

31

## What's Next

- General Concepts
- IA-32 Processor Architecture
- **IA-32 Memory Management**
- 64-Bit Processors
- Components of an IA-32 Microcomputer
- Input-Output System

## IA-32 Memory Management

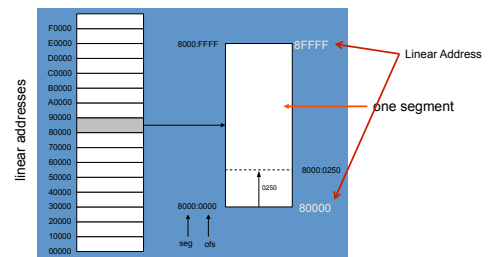
- Real-address mode
- Calculating linear addresses
- Protected mode
- Multi-segment model
- Paging

## Real-Address mode

- 1 MB RAM maximum addressable
  - 0 to FFFFF hexadecimal
- Application programs can access any area of memory
- Single tasking
- Supported by MS-DOS operating system
- 8086 processor could not hold 20-bit addresses
  - Uses segmented memory
  - All memory divided into 64KB units called segments

## Segmented Memory

- Segmented memory addressing: absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset
- Each segment begins with an address having a 0 as it's last hex digit



## Calculating Linear Addresses

- Given a segment address, multiply it by 16 (add a hexadecimal zero), and add it to the offset
- Example: convert 08F1:0100 to a linear address

Adjusted Segment value:	0 8 F 1 0
Add the offset:	0 1 0 0
Linear address:	0 9 0 1 0

## Your turn . . .

What linear address corresponds to the segment/offset address 028F:0030?

$$028F0 + 0030 = 02920$$

Always use hexadecimal notation for addresses.

## Your turn . . .

Lets go the other way: given a linear address, find the segment address

What segment addresses correspond to the linear address 28F30h?

Many different segment-offset addresses can produce the linear address 28F30h. For example:

28F0:0030, 28F3:0000, 28B0:0430, . . .

Invine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

38

## Protected Mode (1 of 2)

- 4 GB addressable RAM  
– (00000000 to FFFFFFFFh)
- Each program assigned a memory partition which is protected from other programs
- Designed for multitasking
- Supported by Linux & MS-Windows

Invine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

39

## Protected mode (2 of 2)

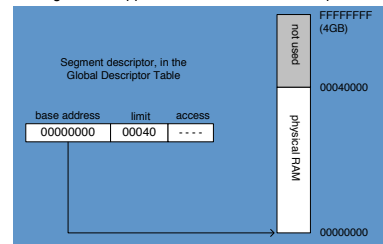
- Segment descriptor tables
  - Used to keep track of locations of individual program segments
- Program structure
  - code, data, and stack areas
  - CS, DS, SS segment descriptors
  - global descriptor table (GDT)
- MASM (Microsoft assembler ) Programs use the Microsoft **flat** memory model

Invine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

40

## Flat Segment Model

- Single global descriptor table (GDT).
- All segments mapped to entire 32-bit address space



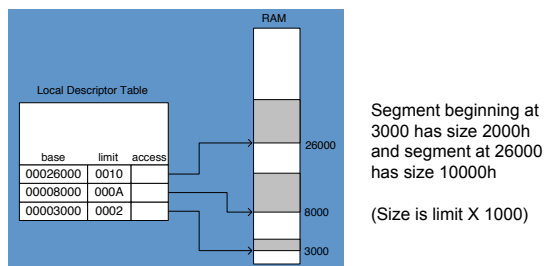
- Base Address: where segment begins
- Limit: Where the segment ends
- Access: bits to help determine how segment is used

Invine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

41

## Multi-Segment Model

- Each program has a local descriptor table (LDT)
  - holds descriptor for each segment used by the program



Invine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

42

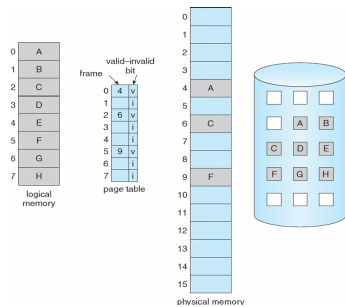
## Paging

- Supported directly by the CPU
- Divides each segment into 4096-byte blocks called **pages**
- Sum of all programs can be larger than physical memory
- Part of running program is in memory, part is on disk
- **Virtual memory manager (VMM)** – OS utility that manages the loading and unloading of pages
- **Page fault** – issued by CPU when a page must be loaded from disk

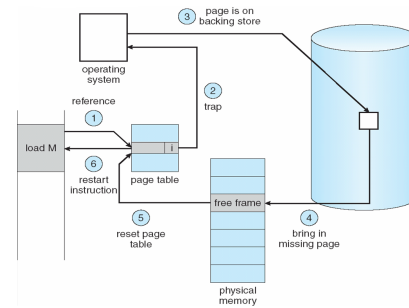
Invine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

43

### Page Table When Some Pages Are Not in Main Memory



### Steps in Handling a Page Fault



### What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- **64-Bit Processors**
- Components of an IA-32 Microcomputer
- Input-Output System

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

46

### 64-Bit Processors

- 64-Bit Operation Modes
  - Compatibility mode – can run existing 16-bit and 32-bit applications (Windows supports only 32-bit apps in this mode)
  - 64-bit mode – Windows 64 uses this
- Basic Execution Environment
  - addresses can be 64 bits (48 bits, in practice)
  - 16 64-bit general purpose registers
  - 64-bit instruction pointer named RIP

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

47

### 64-Bit General Purpose Registers

- 32-bit general purpose registers:
  - EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D, R9D, R10D, R11D, R12D, R13D, R14D, R15D
- 64-bit general purpose registers:
  - RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8, R9, R10, R11, R12, R13, R14, R15

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

48

### What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-Bit Processors
- **Components of an IA-32 Microcomputer**
- Input-Output System

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

49



## Components of an IA-32 Microcomputer

- Motherboard
- Video output
- Memory
- Input-output ports

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

50

## Motherboard

- CPU socket
- External cache memory slots
- Main memory slots
- BIOS chips
- Sound synthesizer chip (optional)
- Video controller chip (optional)
- IDE, parallel, serial, USB, video, keyboard, joystick, network, and mouse connectors
- PCI bus connectors (expansion cards)

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

51

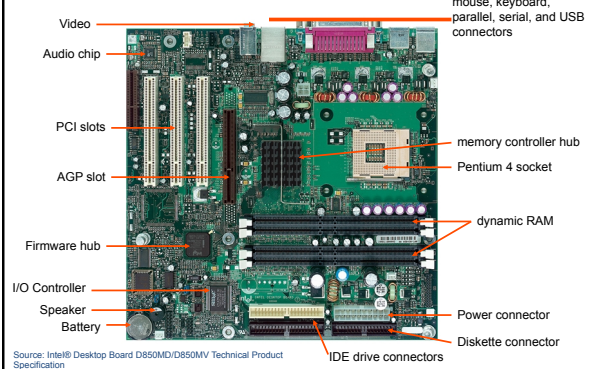
## Important Support Processors

- Floating-Point Unit (FPU)
- 8284/82C284 Clock Generator (clock)
- 8259A Programmable Interrupt Controller (PIC)
  - Handles interrupts from external devices
- 8253 Programmable Interval Timer/Counter
  - Interrupts system 18.2 times/second
  - Updates system date and clock
  - Controls the speaker
  - Refreshes RAM memory chips
- 8255 Programmable Parallel Port
  - Transfers data to and from the computer
  - Commonly used for printers as well as other i/o devices

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

52

## Intel D850MD Motherboard



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

53

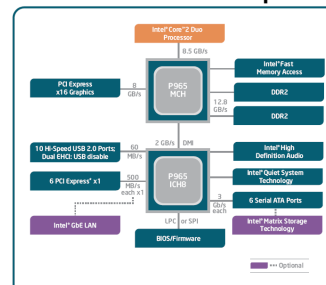
## PCI and PCI Express Bus Architecture

- Peripheral Component Interconnect (PCI) bus
  - Connecting bridge between CPU and other system devices
    - Memory, hard drives, video controllers, sound cards, and network controllers
  - Provides two-way serial connections between
    - Devices, memory, and the processor
    - Carries data in packets similar to networks
    - Widely supported by graphics controllers
    - Transfer rate about 4 Gbytes/second

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

54

## Intel 965 Express Chipset

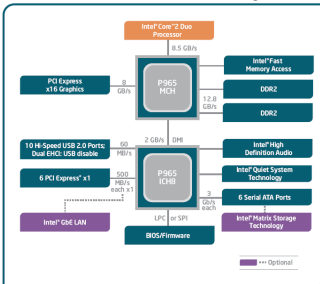


Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

55

- Collection of processor chips designed to work together on a specific type of motherboard
- P965 used in desktop PC with either a Core 2 Duo or Pentium D processor

## Intel 965 Express Chipset



- Some features of P965
  - Fast Memory Access
  - I/O Controller Hub (ICH8/R/DH)
  - Supports 10 USB ports, six PCI express slots and networking
  - High definition audio chip

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

56

## Video Output

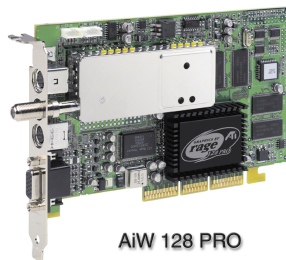
- Video controller
  - on motherboard, or on expansion card
  - AGP ([accelerated graphics port technology](#))
- Video memory (VRAM)
- Video CRT Display
  - uses raster scanning
  - horizontal retrace
  - vertical retrace
- Direct digital LCD monitors
  - no raster scanning required

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

57

## Sample Video Controller (ATI Corp.)

- 128-bit 3D graphics performance powered by RAGE™ 128 PRO
- 3D graphics performance
- Intelligent TV-Tuner with Digital VCR
- TV-ON-DEMAND™
- Interactive Program Guide
- Still image and MPEG-2 motion video capture
- Video editing
- Hardware DVD video playback
- Video output to TV or VCR



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

58

## Memory

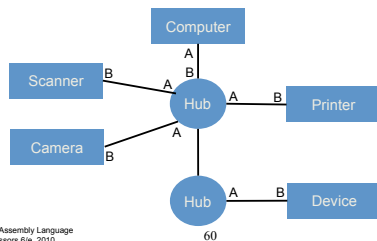
- ROM
  - read-only memory
- EPROM
  - erasable programmable read-only memory
- Dynamic RAM (DRAM)
  - inexpensive; must be refreshed constantly
- Static RAM (SRAM)
  - expensive; used for cache memory; no refresh required
- Video RAM (VRAM)
  - dual ported; optimized for constant video refresh
- CMOS RAM
  - complimentary metal-oxide semiconductor
  - system setup information
- See: [intel platform memory](http://www.intel.com/content/www/us/en/platform-memory/platform-memory.html) for latest memory configuration
  - (<http://www.intel.com/content/www/us/en/platform-memory/platform-memory.html>)

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

59

## Input-Output Ports

- USB (universal serial bus)
  - intelligent high-speed connection to devices
  - up to 12 megabits/second
  - USB hub connects multiple devices
  - enumeration*: computer queries devices
  - supports *hot* connections



Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

60

## Input-Output Ports (cont)

- Parallel
  - short cable, high speed
  - common for printers
  - bidirectional, parallel data transfer
  - Intel 8255 controller chip
- Serial
  - RS-232 serial port
  - one bit at a time
  - uses long cables and modems
  - 16550 UART (universal asynchronous receiver transmitter)
  - programmable in assembly language

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

61

## Device Interfaces

- ATA (Advanced Technology Attachment) host adapters
  - intelligent drive electronics (hard drive, CDROM)
- SATA (Serial ATA)
  - inexpensive, fast, bidirectional
- FireWire
  - high speed (800 MB/sec), many devices at once
- Bluetooth
  - small amounts of data, short distances, low power usage
- Ethernet
  - IEEE 802.3 standard, high speed wired network
- Wi-Fi (wireless Ethernet)
  - IEEE 802.11 standard, faster than Bluetooth

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

62

## What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- Components of an IA-32 Microcomputer
- **Input-Output System**

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

63

## Levels of Input-Output

- Level 3: High-level language functions
  - examples: C++, Java
  - portable, convenient, not always the fastest
- Level 2: Operating system
  - Application Programming Interface (API) library
  - Writing strings to files, reading strings from keyboard, allocating blocks of memory
  - extended capabilities, lots of details to master
- Level 1: BIOS
  - drivers that communicate directly with devices
  - OS security may prevent application-level code from working at this level

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

64

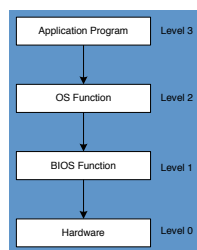
## Device Drivers

- Programs that permit the OS to communicate directly with hardware devices
- Simplifies programming by acting as translator between a hardware device and the applications or operating systems that use it
- Programmers can write the higher-level application code independently of whatever specific hardware the end-user is using

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

65

## Displaying a String of Characters



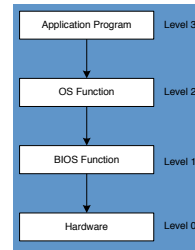
When a HLL program displays a string of characters, the following steps take place:

- Level 3: The library function calls an operating system function, passing a string pointer
- Level 2: The operating system function uses a loop to call a BIOS subroutine, passing it the ASCII code and color of each character. The operating system calls another BIOS subroutine to advance the cursor to the next position on the screen

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

66

## Displaying a String of Characters

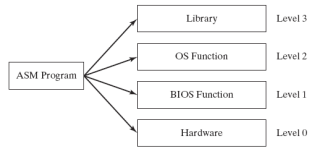


- Level 1: The BIOS subroutine receives a character, maps it to a particular system font, and sends the character to a hardware port attached to the video controller card
- Level 0: The video controller card generates timed hardware signals to the video display that control the raster scanning and displaying of pixels

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

67

## Programming levels



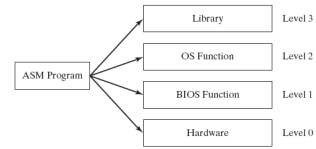
Assembly language programs can perform input-output at each of the following levels:

- Level 3: Call library functions to perform generic text I/O and file-based I/O. We would supply such a library with this class, for instance.
- Level 2: Call operating system functions to perform generic text I/O and file-based I/O. If the OS uses a graphical user interface, it has functions to display graphics in a device-independent way.

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

68

## Programming levels



Assembly language programs can perform input-output at each of the following levels:

- Level 1 : Call BIOS functions to control device-specific features such as color, graphics, sound, keyboard input, and low-level disk I/O.
- Level 0: Send and receive data from hardware ports, having absolute control over specific devices.

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

69

## Tradeoffs

- Control versus portability
  - Level 2 works on any computer running the same operating system, but is not particularly fast because each I/O call must go through several layers before it executes
  - Level 1 works on all systems having a standard BIOS, but will not produce the same result on all systems
  - Level 0 works with generic devices such as serial ports and with specific I/O devices produced by known manufacturers.
  - Not all OS permit user programs to directly access system hardware. Reserved for OS and device drivers only

Irvine, Kip R. Assembly Language for x86 Processors 6/e, 2010.

70



42 69 6E 61 72 79

What does this say?

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2015.

72