CHAPTER 2: X86 PROCESSOR ARCHITECTURE

Chapter Overview

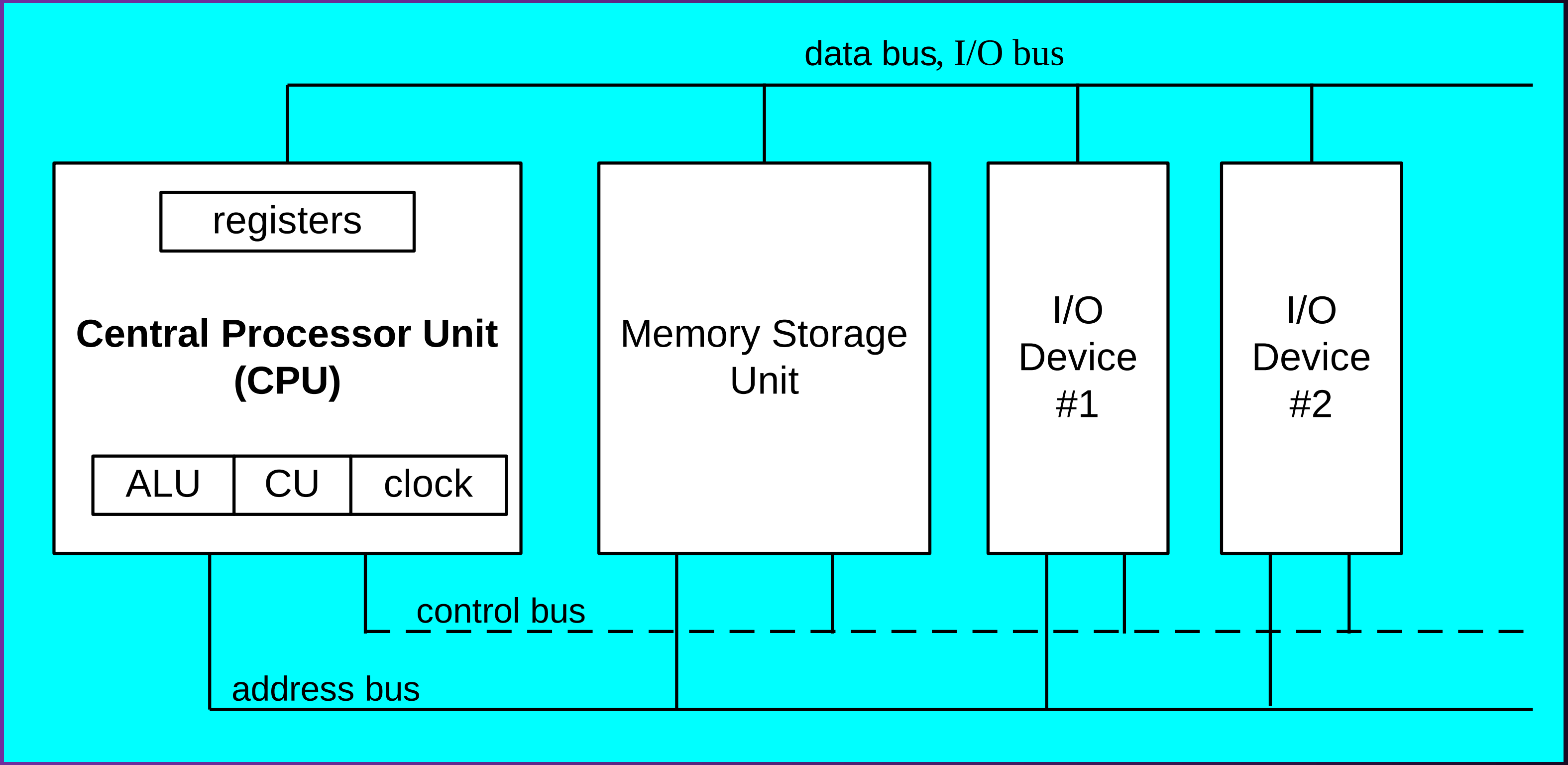
- **General Concepts**
- 32-Bit x86 Processors
- 64-Bit x86-64 Processors
- Components of a Typical x86 Computer
- Input-Output System

General Concepts

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

Basic Microcomputer Design

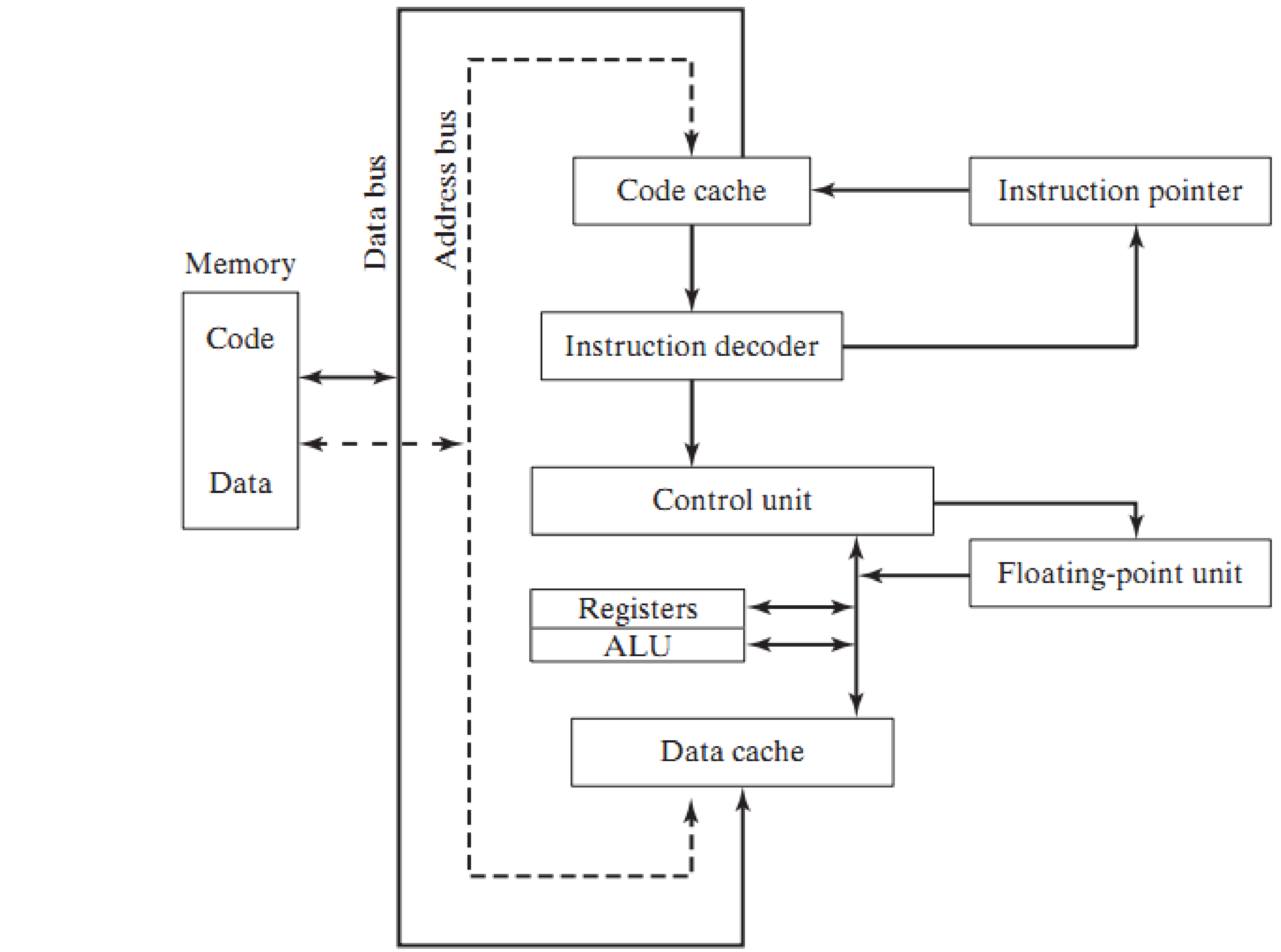
- ALU performs arithmetic and bitwise processing
- clock synchronizes CPU operations
- control unit (CU) coordinates sequence of execution steps



Instruction Execution Cycle

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

Figure 2–2 Simplified CPU Block Diagram.



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.

What's Next

- General Concepts
- **32-Bit x86 Processors**
- 64-Bit x86-64 Processors
- Components of a Typical Computer
- Input-Output System

IA-32 Processor Architecture

- Modes of operation
- Basic execution environment
- x86 Memory Management

Modes of Operation

- Protected mode
  - native mode (Windows, Linux)
- Real-address mode
  - native MS-DOS
- System management mode
  - power management, system security, diagnostics

- Virtual-8086 mode
  - hybrid of Protected
  - each program has its own 8086 computer

Basic Execution Environment

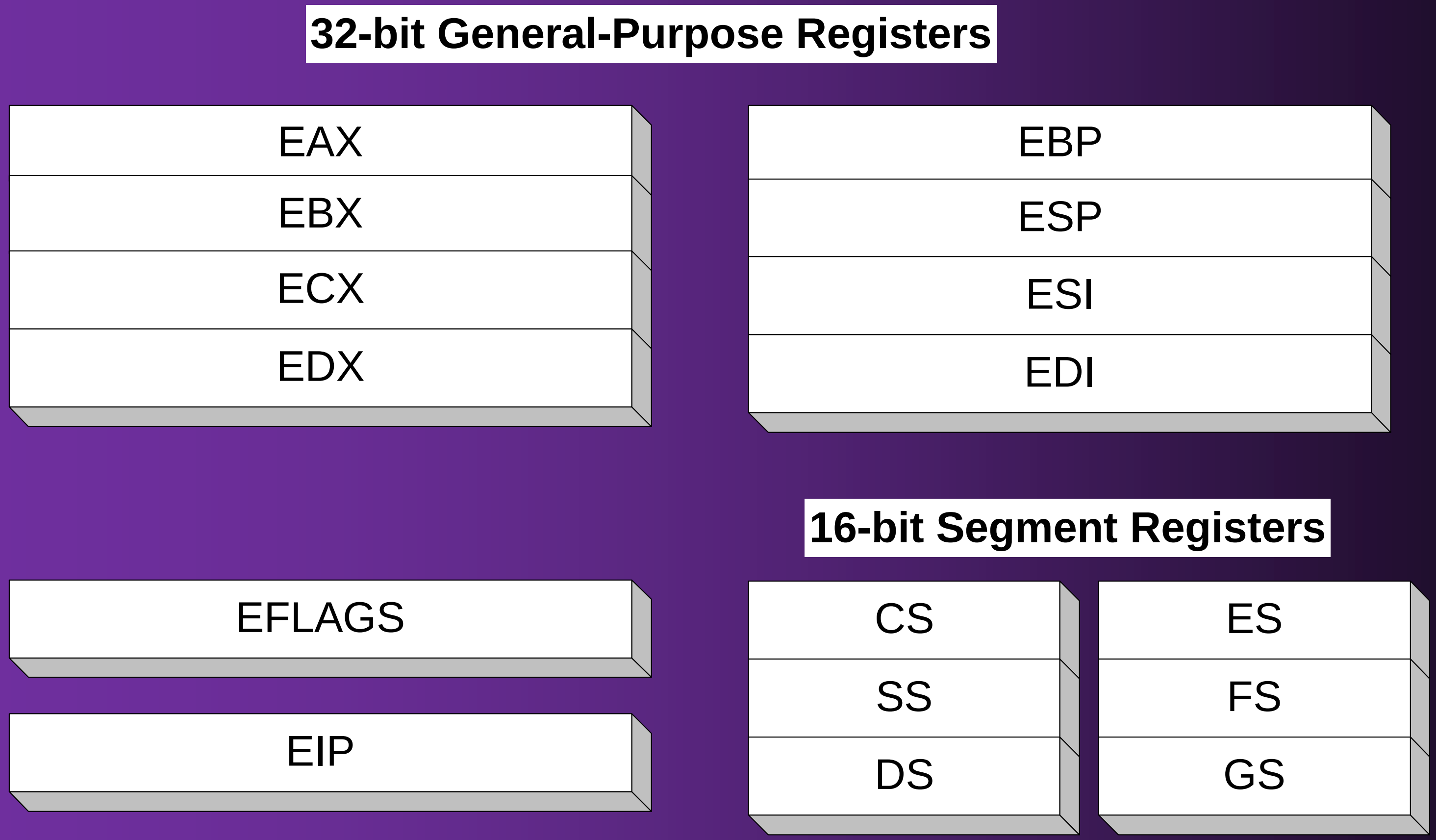
- Addressable memory
- General-purpose registers
- Index and base registers
- Specialized register uses
- Status flags
- Floating-point, MMX, XMM registers

Addressable Memory

- Protected mode
  - 4 GB
  - 32-bit address
- Real-address and Virtual-8086 modes
  - 1 MB space
  - 20-bit address

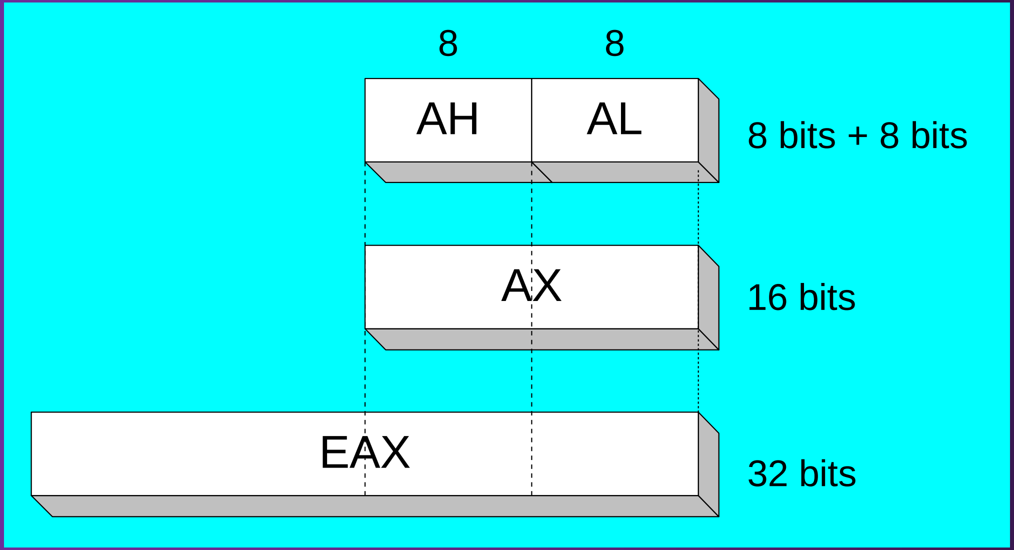
General-Purpose Registers

Named storage locations inside the CPU, optimized for speed.



Accessing Parts of Registers

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



32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

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

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

Some Specialized Register Uses (2 of 2)

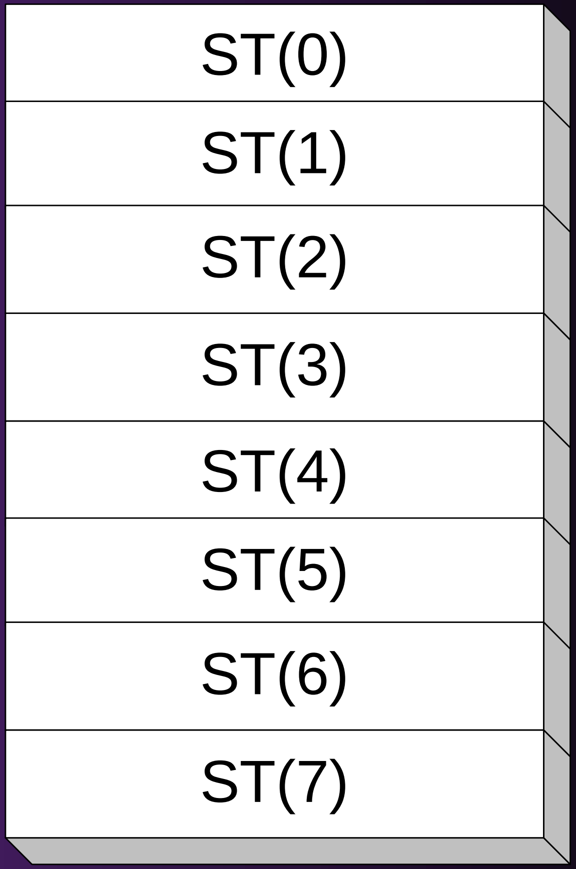
- EIP – instruction pointer
- EFLAGS
  - status and control flags
  - each flag is a single binary bit

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

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



x86 Memory Management

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

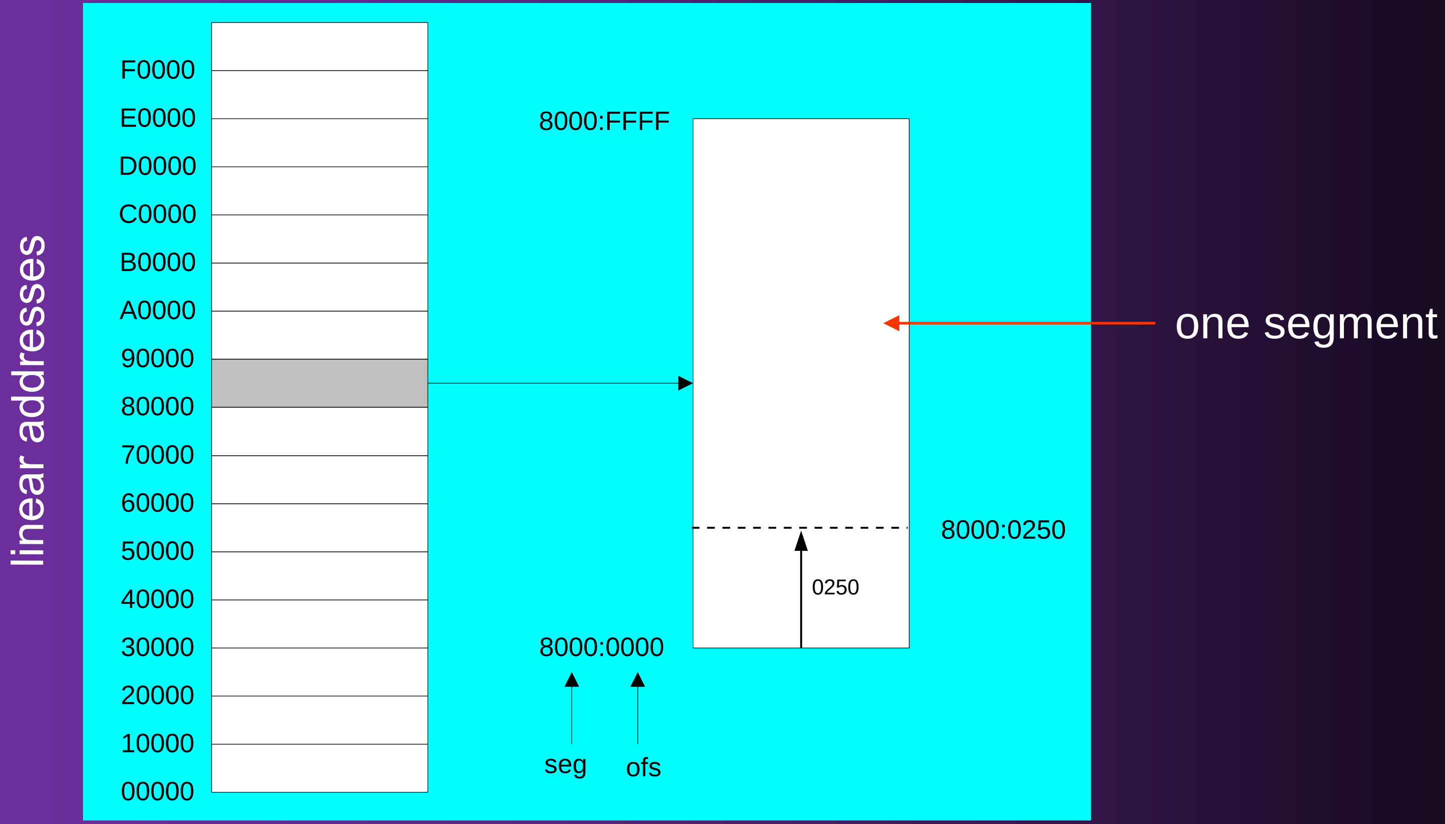
- (see Chapter 11)

Real-Address mode

- 1 MB RAM maximum addressable
- Application programs can access any area of memory
- Single tasking
- Supported by MS-DOS operating system

Segmented Memory

Segmented memory addressing: absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset



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
```

Protected Mode

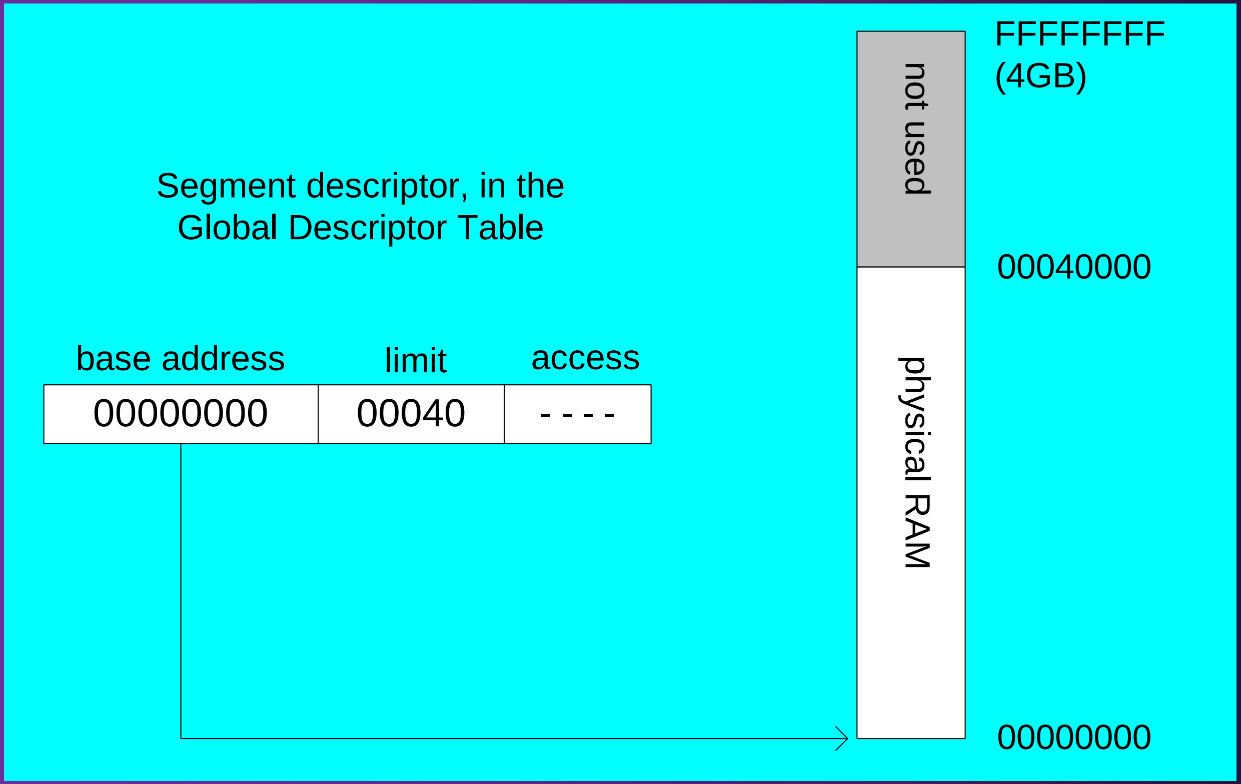
- 4 GB addressable RAM
  - (00000000 to FFFFFFFFh)
- Each program assigned a memory partition which is protected from other programs
- Designed for multitasking
- Supported by Linux & MS-Windows

Protected mode (2 of 2)

- Segment descriptor tables
- Program structure
  - code, data, and stack areas
  - CS, DS, SS segment descriptors
  - global descriptor table (GDT)
- MASM Programs use the Microsoft flat memory model

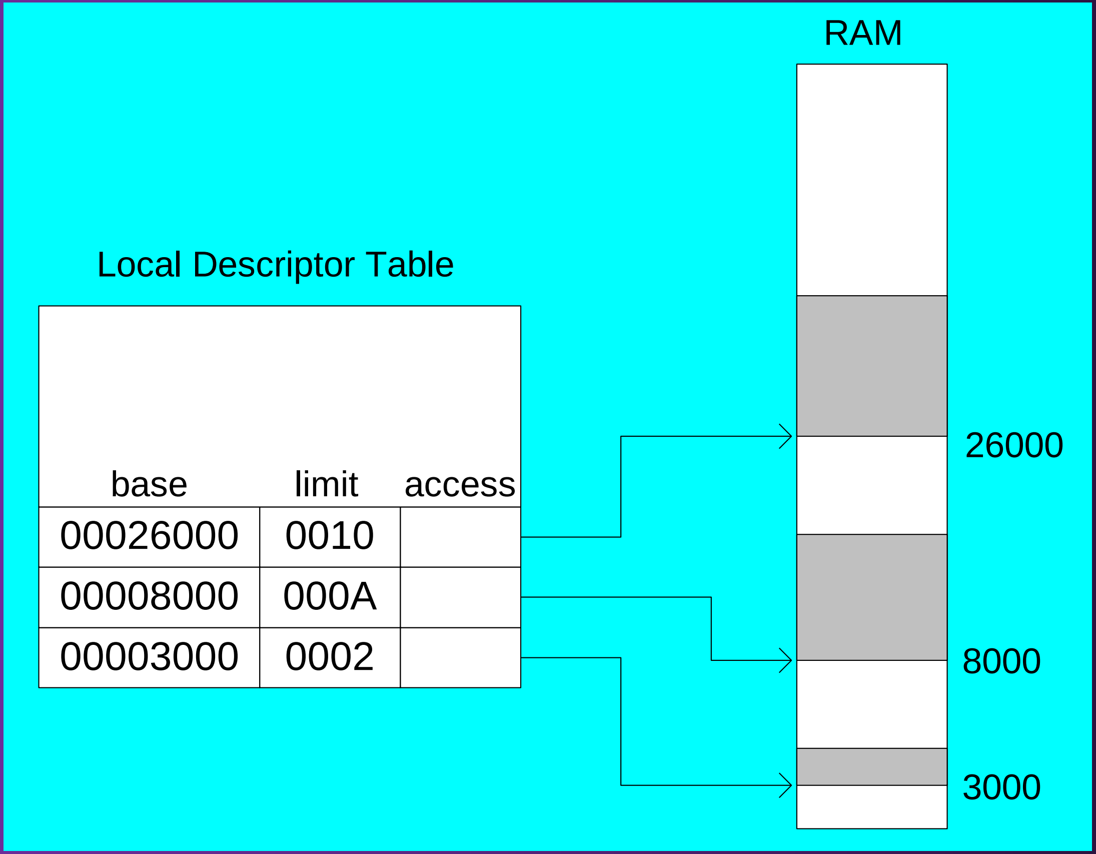
Flat Segment Model

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



Multi-Segment Model

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



Local Descriptor Table

base	limit	access
00026000	0010	
00008000	000A	
00003000	0002	

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

What's Next

- General Concepts
- 32-Bit x86 Processors
- **64-Bit x86-64 Processors**
- Components of a Typical x86 Computer
- Input-Output System

64-Bit x86-64 Processors

- x86-64
  - 64-bit linear address space
  - Intel64: Xeon (至强), Core i5, Core i7, and Core i9 ...
  - AMD64: Opteron (皓龙), Athlon (速龙) 64, Turion (炫龙) 64 ...
- IA-32e Mode
  - Compatibility mode for legacy 16- and 32-bit applications
  - 64-bit Mode uses 64-bit addresses and operands

x86-64 Assembly Information

- x86-64: http://en.wikipedia.org/wiki/X86-64

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 status flags register named RFLAGS (only the lower 32 bits are used)
  - 64-bit instruction pointer named RIP

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

What's Next

- General Concepts
- 32-Bit x86 Processors
- 64-Bit x86-64 Processors
- **Components of a Typical Computer**
- Input-Output System

Components of an IA-32 Microcomputer

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

What's Next

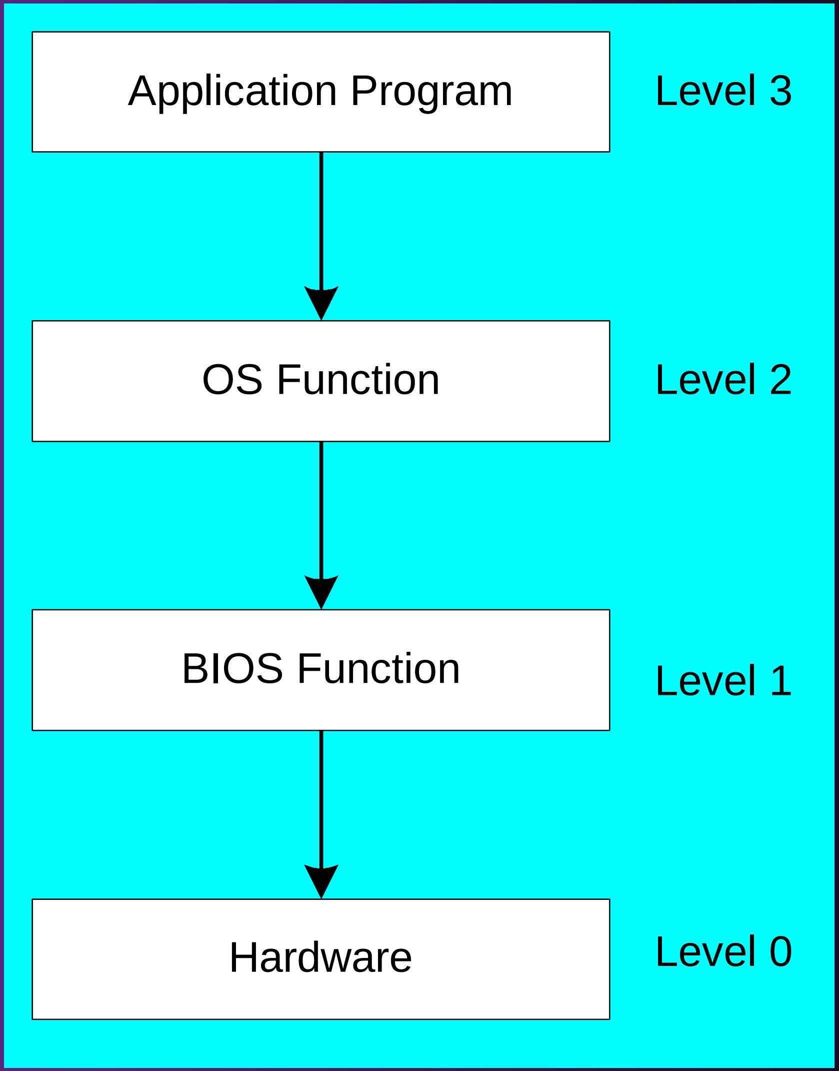
- General Concepts
- 32-Bit x86 Processors
- 64-Bit x86-64 Processors
- Components of a Typical x86 Computer
- **Input-Output System**

Levels of Input-Output

- Level 3: High-level language function
  - examples: C++, Java
  - portable, convenient, not always the fastest
- Level 2: Operating system
  - Application Programming Interface (API)
  - 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

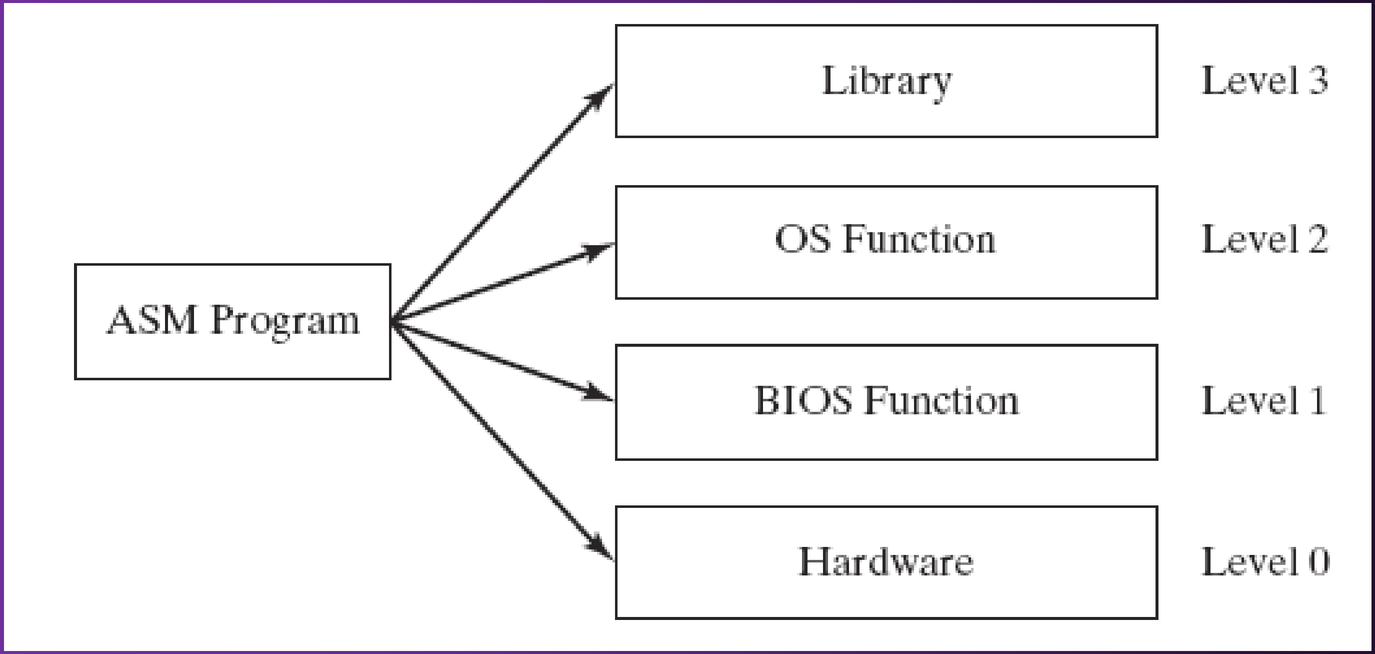
Displaying a String of Characters

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



Programming levels

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


ASM Program
Library
Level 3
OS Function
Level 2
BIOS Function
Level 1
Hardware
Level 0