

A decorative graphic element consisting of a blue gradient shape that starts as a thin arc on the left and expands into a larger, darker blue triangular area pointing towards the bottom right corner of the slide.

**Week 2**

# Chapter 2: x86 Processor Architecture

Class 4

# Chapter Overview

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

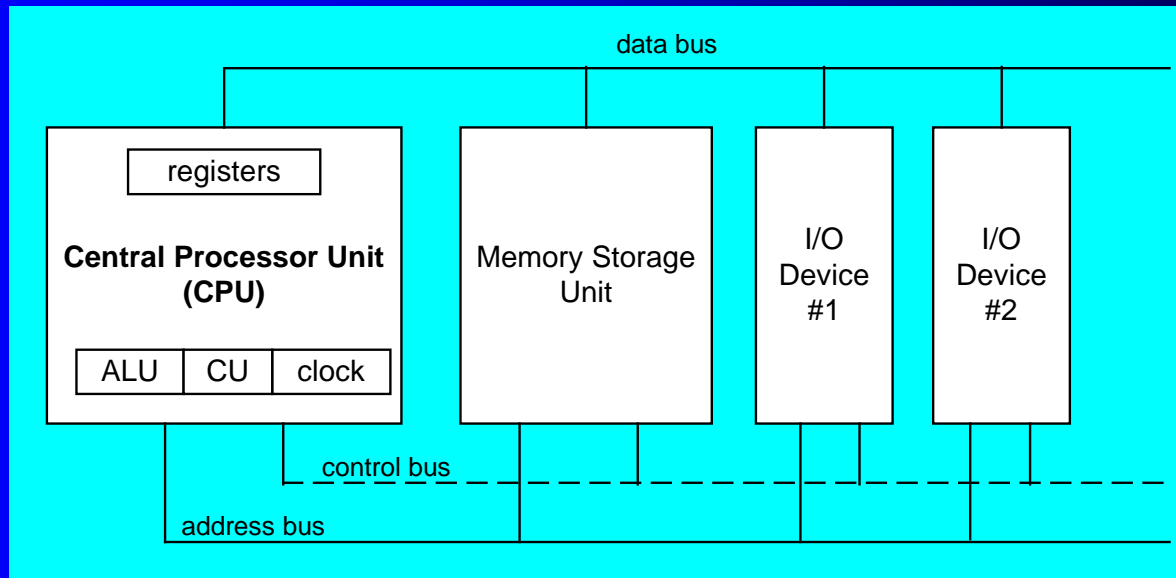
# General Concepts

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

Book's Page No 33

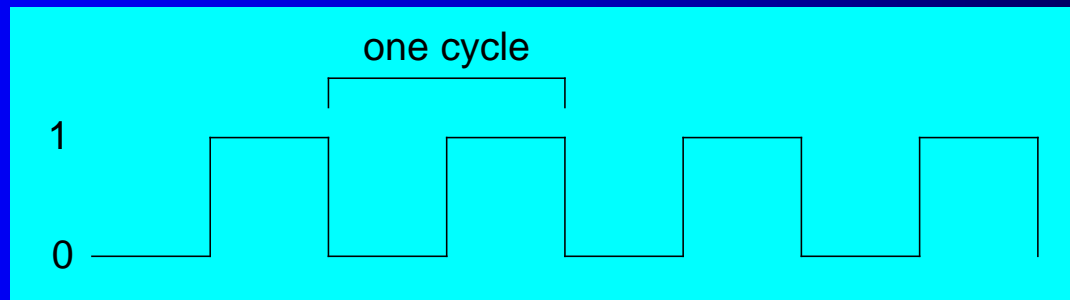
# Basic Microcomputer Design

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



# Clock

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

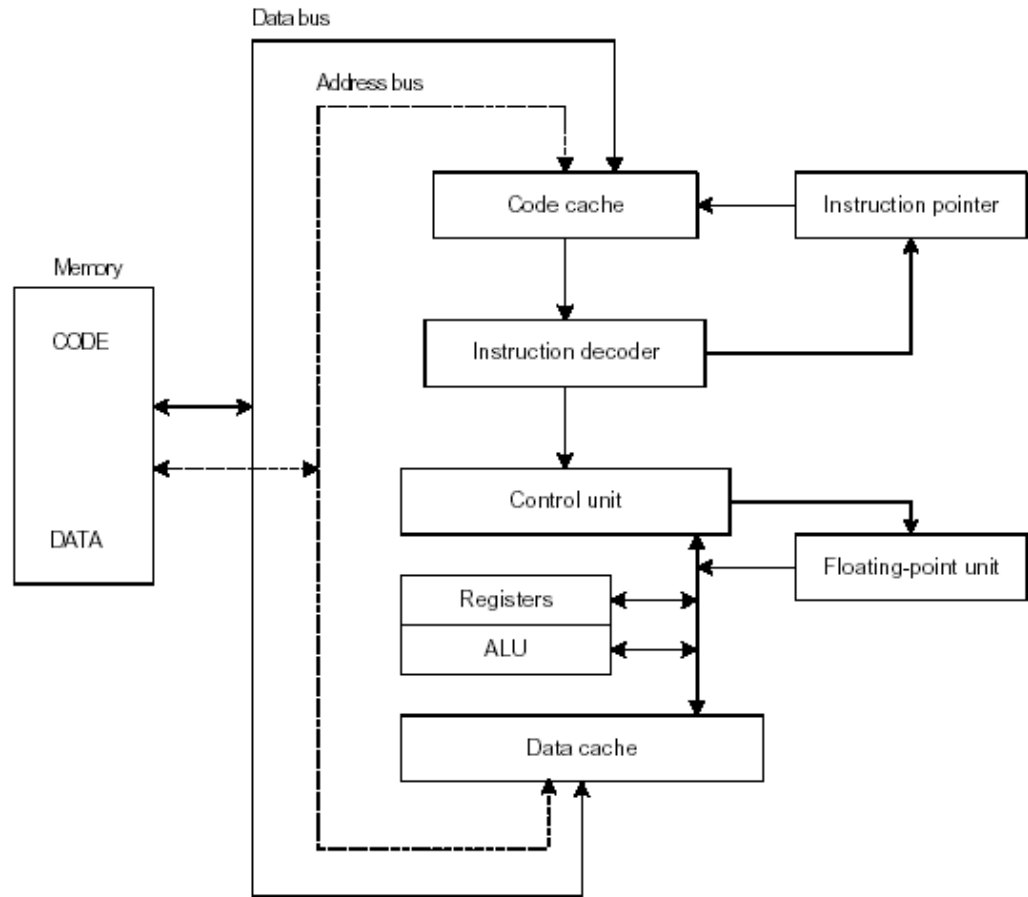


# What's Next

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

# Instruction Execution Cycle

Figure 2-2 Simplified Pentium CPU Block Diagram.



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

# Reading from Memory

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

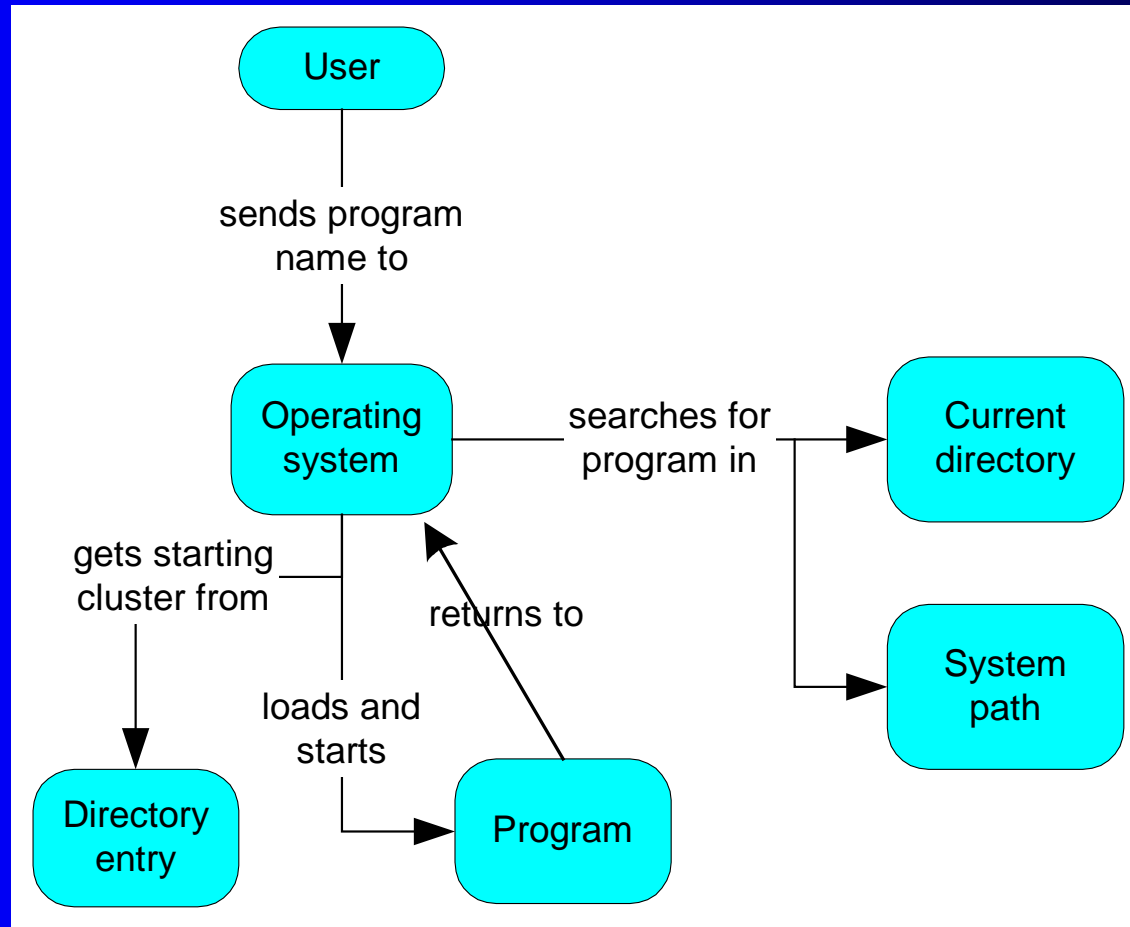
1. Place the address of the value you want to read on the address bus.
2. Assert (changing the value of) the processor's RD (read) pin.
3. Wait one clock cycle for the memory chips to respond.
4. Copy the data from the data bus into the destination operand



# 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.

# How a Program Runs



# IA-32 Processor Architecture

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

# 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.

## 32-bit General-Purpose Registers

EAX
EBX
ECX
EDX

EBP
ESP
ESI
EDI

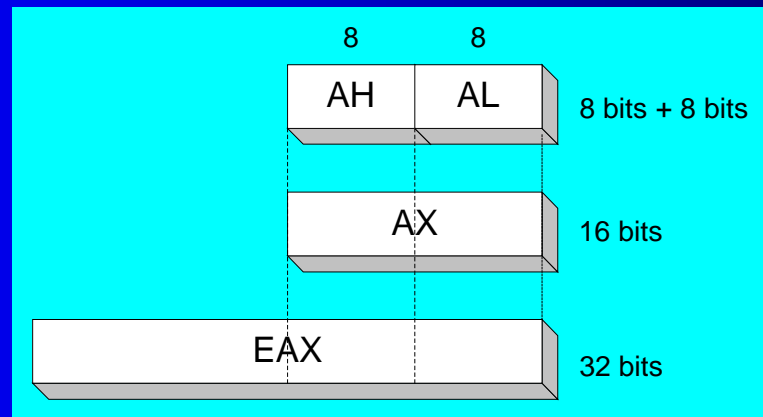
## 16-bit Segment Registers

EFLAGS
EIP

CS	ES
SS	FS
DS	GS

# 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)

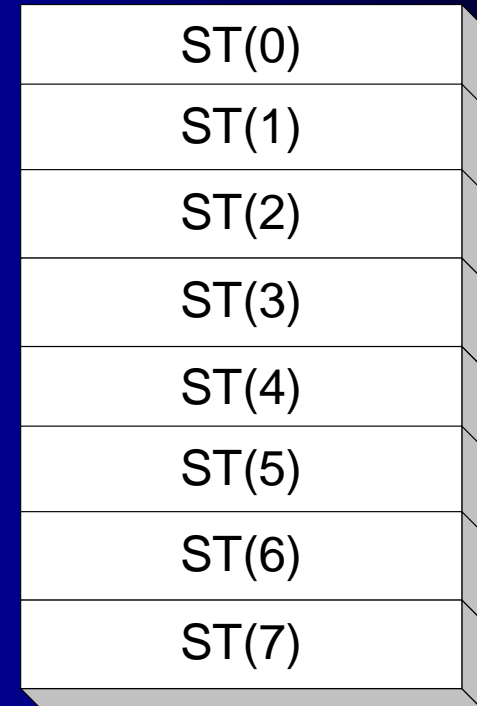
- EIP – instruction pointer
- EFLAGS
  - status and control flags
  - each flag is a single binary bit

# 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



ST(0)
ST(1)
ST(2)
ST(3)
ST(4)
ST(5)
ST(6)
ST(7)