Assembly Language for x86 Processors

6th Edition

Kip Irvine

Chapter 2: x86 Processor Architecture

Slides prepared by the author

Revision date: 2/15/2010

Chapter Overview

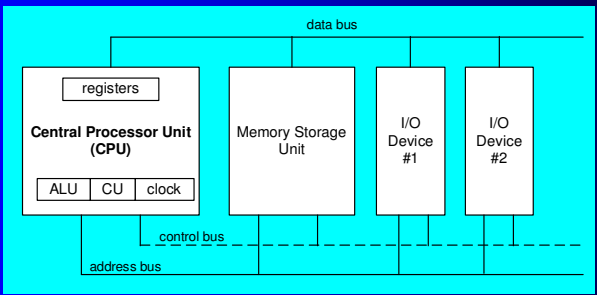
- **General Concepts**
- IA-32 Processor Architecture
- IA-32 Memory Management
- Components of an IA-32 Microcomputer
- Input-Output System

General Concepts

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

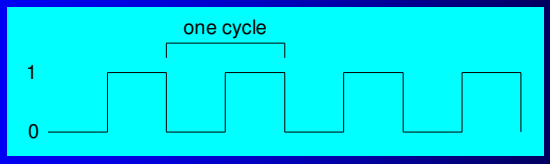
Basic Microcomputer Design

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



Clock

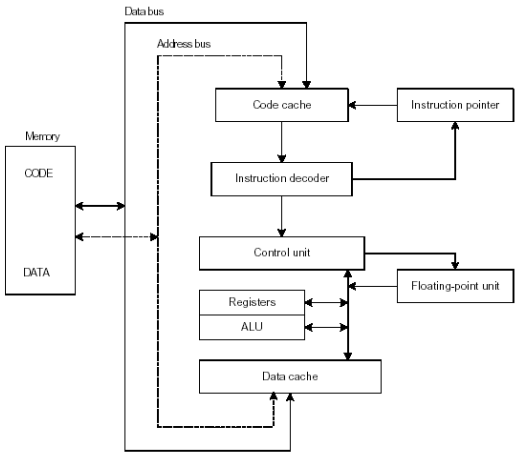
- synchronizes all CPU and BUS operations
- machine (clock) cycle measures time of a single operation
- clock is used to trigger events



Instruction Execution Cycle

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

Figure 2–2 Simplified Pentium CPU Block Diagram.

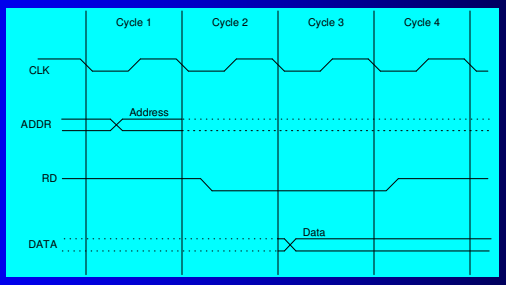


Instruction Execution Cycle

- Fetch – CU gets next instruction from the instruction queue and increments instruction pointer (IP)
- Decode – CU determines the instruction's function and the instruction's input operands are sent to the ALU; signals are sent to ALU to tell it what to do
- Fetch Operands – if operand is located in memory, CU issues a "read" to retrieve the data and put it in internal registers
- Execute – ALU performs instruction and sends output to registers or memory
- Store output operand – if operand is in memory, CU issues a "write" operation

Reading from Memory (clock – GHz)

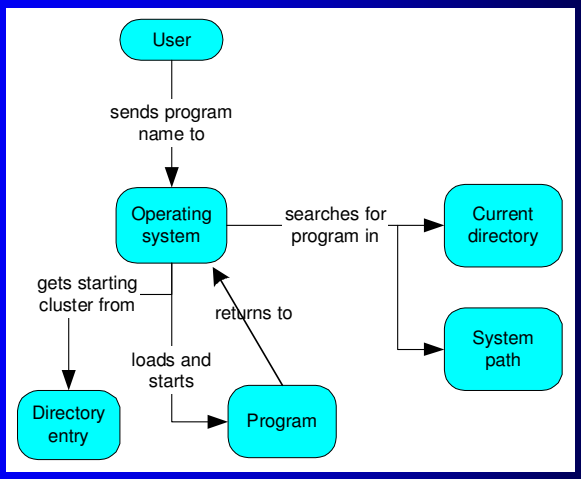
- Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:
  - address placed on address bus
  - Read Line (RD) set low
  - CPU waits one cycle for memory to respond
  - Read Line (RD) goes to 1, indicating that the data is on the data bus



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



Load and Execute Process

- OS searches for program's filename in disk directory, or in predetermined list (*paths*)
- OS retrieves basic info about program from disk directory (file size, physical location)
- OS determines free space in memory and loads the program; keeps data in *descriptor* file, and may adjust addresses within the program
- OS begins execution of program. Now the program is called a *process*. OS assigns a process ID number.
- OS tracks execution of program and responds to system resource requests from the program.

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.

What's Next

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

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); programs cannot address outside area they are assigned
- Real-address mode
  - native MS-DOS; dangerous; direct access to HW
- System management mode
  - power management, system security, diagnostics

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

Basic Execution Environment

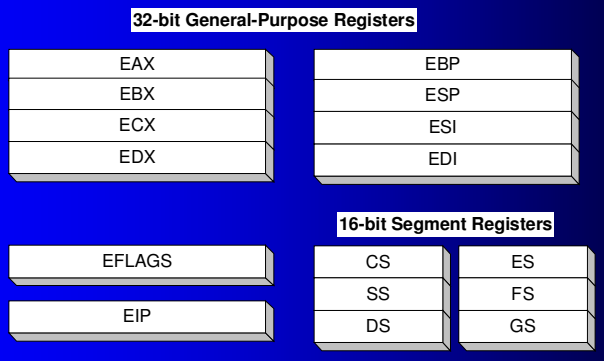
- Addressable memory
- General-purpose registers
- Index and base registers
- Specialized register uses
- Status flags
- Floating-point, MMX, XMM registers

Addressable Memory

- Protected mode
  - Linear up to about 4 GB
  - 32-bit address
  - P6 processors can go to 64 GB
- Real-address and Virtual-8086 modes
  - 1 MB space
  - 20-bit address

General-Purpose Registers

Named storage locations inside the CPU, optimized for speed.

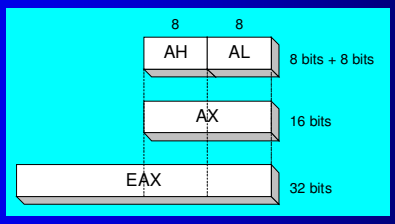


The alphabet

- A = Accumulator
- B = Base
- C = Counter or Code
- D = Data
- E = Extended
- F = Flag
- I = Index or Instruction
- H = High
- L = Low
- S = Segment or Stack
- P = Pointer
- X = Register

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

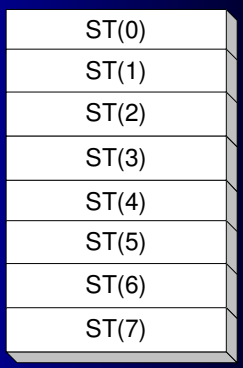
- Carry -CF
  - unsigned arithmetic out of range
- Overflow - OF
  - signed arithmetic out of range
- Sign - SF
  - result is negative
- Zero - ZF
  - result is zero
- Auxiliary Carry - AF
  - carry from bit 3 to bit 4
- Parity - PF
  - sum of 1 bits is an even number
- Trap – TF
  - Causes processor to execute in single steps
- Interrupt – IF
  - Disable external interrupts
- Direction – DF
  - Used in string operations

Flag Register Bits

	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
				O	D	I	T	S	Z		A		P		C

Floating-Point, MMX, XMM Registers

- MM = Multimedia
- 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



Intel Microprocessor History

- Intel 8086, 80286
- IA-32 processor family
- P6 processor family
- CISC and RISC

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)

The IBM-AT

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

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

64-bit Processors

- Intel64
  - 64-bit linear address space
  - Intel: Pentium Extreme, Xeon, Celeron D, Pendium 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

Intel Technologies

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

Intel Processor Families

Currently Used:

- Pentium & Celeron – dual core
- Core 2 Duo - 2 processor cores
- Core 2 Quad - 4 processor cores
- Core i7 – 4 processor cores

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)

What's Next

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

IA-32 Memory Management

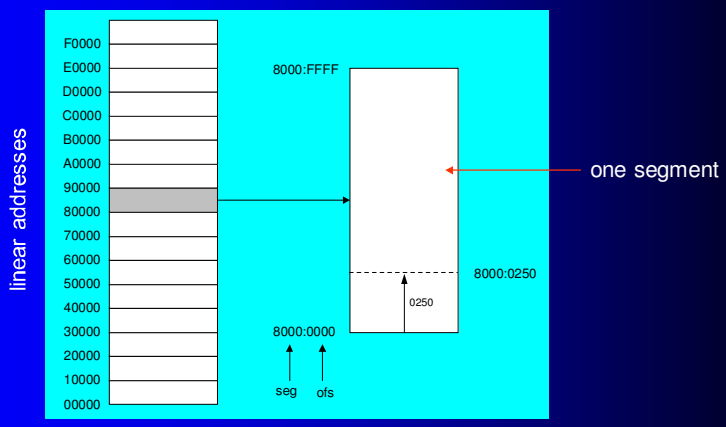
- Real-address mode
- Calculating linear addresses
- Protected mode
- Multi-segment model
- Paging

Real-Address mode

- 1 MB RAM maximum addressable
- 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
```


Your turn . . .

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

028F0 + 0030 = 02920

Always use hexadecimal notation for addresses.

Your turn . . .

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, . . .

Protected Mode (1 of 2)

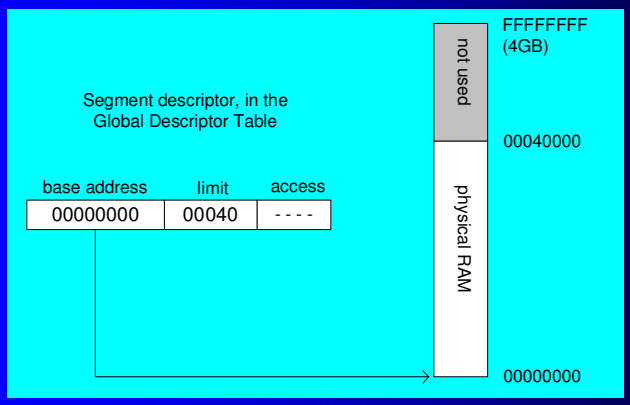
- 4 GB addressable RAM
  - (00000000 to FFFFFFFFh)
- Each program assigned a memory partition which is protected from other programs
- Designed for multitasking
- Supported by Linux & MS-Windows

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

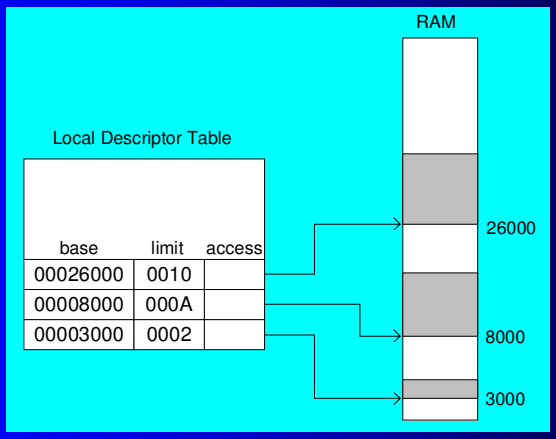
Flat Segment Model

- 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



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

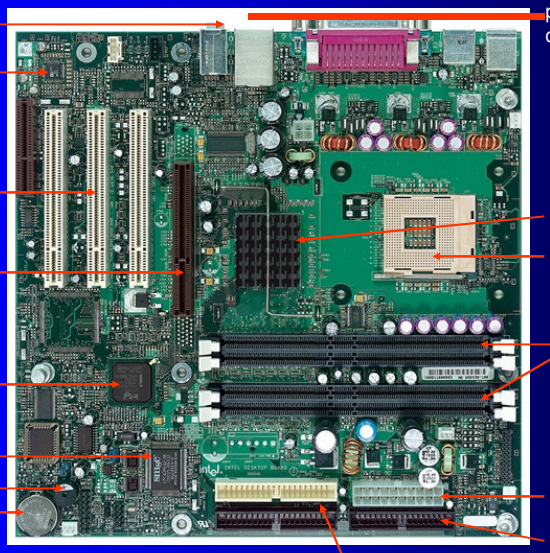
Components of an IA-32 Microcomputer

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

Motherboard

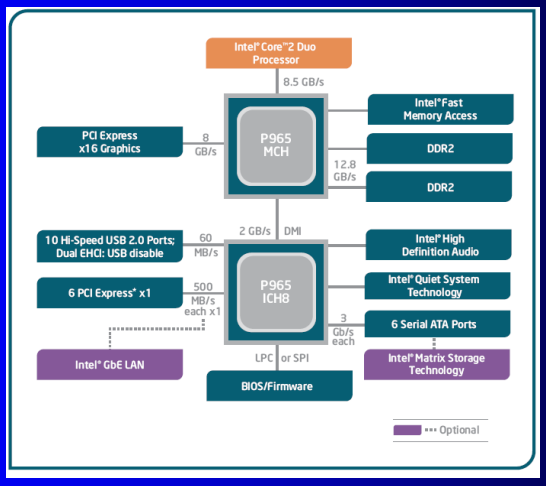
- CPU socket
- External cache memory slots
- Main memory slots
- BIOS (basic input/output system) chips
- Sound synthesizer chip (optional)
- Video controller chip (optional)
- IDE, parallel, serial, USB, video, keyboard, joystick, network, and mouse connectors
- PCI bus connectors (expansion cards)

Intel D850MD Motherboard



Source: Intel® Desktop Board D850MD/D850MV Technical Product Specification

Intel 965 Express Chipset



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

* This link may change over time.

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

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 (Intel technology brief: link address may change)

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
- Parallel
  - short cable, high speed
  - common for printers
  - bidirectional, parallel data transfer
  - Intel 8255 controller chip

Input-Output Ports (cont)

- 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

Device Interfaces

- ATA 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
- Wi-Fi (wireless Ethernet)
  - IEEE 802.11 standard, faster than Bluetooth

What's Next

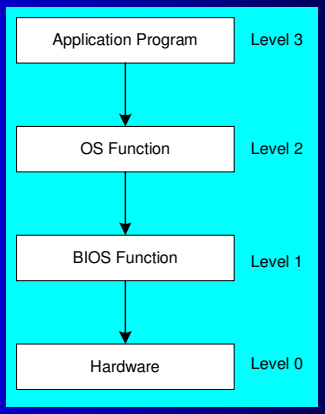
- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- Components of an IA-32 Microcomputer
- **Input-Output System**

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

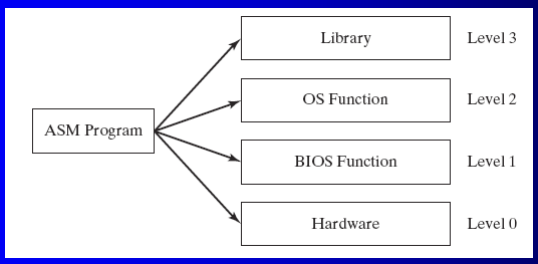
Displaying a String of Characters

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



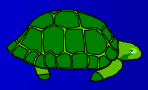
Programming levels

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



Summary

- Central Processing Unit (CPU)
- Arithmetic Logic Unit (ALU)
- Instruction execution cycle
- Multitasking
- Floating Point Unit (FPU)
- Complex Instruction Set
- Real mode and Protected mode
- Motherboard components
- Memory types
- Input/Output and access levels



42 69 6E 61 72 79

What does this say?