Assembly Language for x86 Processors, 7th Edition
Kip Irvine

Chapter 2: IA-32 Processor Architecture

(c) Pearson Education, 2006-2016. 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
★ 32-bit x86 Processors
  • 64-bit x86 Processors
  • Components of a Typical x86 Computer
★ Input-Output System
Basic Microcomputer Design

- **ALU** performs arithmetic and bitwise processing
- **clock** synchronizes CPU operations (cycles per second)
  - machine (clock) cycle measures time of a single operation
- **control unit (CU)** coordinates the sequence of execution steps
- **Bus** – electronic wires that transfer info from one area to another
- **Registers** – storage locations used for processing data
  - MOV AX, 0123h (AX is a register)

![Diagram of Basic Microcomputer Design](image-url)
Basic Microcomputer Design

- **Bus Interface Unit (BIU)** - Responsible for communicating with the world outside the CPU.
  - Generates memory and instruction addresses, and transfers that information to the EU for processing.
- **Execution Unit** - Receives instructions from the BIU, processes the instructions and stores results in the general purpose registers.
Instruction Execution Cycle

Fetch > Decode > Fetch operands > Execute > Store output

Example:

0000h - MOV AX, 1234h  =  B8 34 12
ADD AX, 1000h  =  05 00 10

The memory starting at 0000h = B8 34 12 05 00 10

1. The processor will fetch B8 and increment IP by 1 to 0001h
2. B8 is translated and the processor recognizes that it needs 2 more
   bytes which are to be transferred to the AX register
   a) The processor retrieves the value pointed to by CS:[IP] = 0001h
      this value is moved to AL; IP is incremented by 1 to 0002h
   b) The processor retrieves the value pointed to by CS:[IP] = 0002h,
      and this value is moved to AH; IP is incremented by 1 to 0003h
   c) The processor realizes that it is finished, and returns to the initial
      fetch (Step #1)

Continue Example

Instruction Execution Cycle

Fetch > Decode > Fetch operands > Execute > Store output

Example:

\[
\begin{align*}
0000h & \quad \text{MOV AX, \text{1234h}} & = & \quad \text{B8 34 12} \\
& \quad \text{ADD AX, \text{1000h}} & = & \quad \text{05 00 10}
\end{align*}
\]

The memory starting at 0000h = B8 34 12 05 00 10

3. The processor will fetch the value at (CS:[IP] = 0003h) which is 05h, and increment IP.

4. 05h is translated and found to be an add instruction which requires 2 more bytes which will determine the amount to add to AX

   a) The processor retrieves the value pointed to by (CS:[IP] = 0004h); IP is incremented by 1 to 0005h

   b) The processor retrieves the value pointed to by (CS:[IP] = 0005h); IP is incremented by 1 to 0006h

   c) The processor realizes that it has all values, and performs the addition returning the value in AX, and being finished with this command, it returns to the initial fetch (Step #1)
Reading from Memory RAM

- Reading from memory is much slower than manipulating data within the CPU
- Multiple machine cycles are required when reading from memory.

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.
Multitasking

- OS can run multiple programs at the same time, and multiple threads of execution within the same program.
- CPU can only process one instruction at a time
- 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
★ 32-bit x86 Processors
  • 64-bit x86 Processors
  • Components of a Typical x86 Computer
★ Input-Output System
Modes of Operation

IA-32 Processors have three primary modes of operation

- Real-address mode
  - native MS-DOS
  - 1 MB space
  - 20-bit address (16-bit registers (segmented))

- Protected mode
  - native mode (Windows, Linux)
  - 4 GB space
  - 32-bit address

- System management mode
  - power management, system security, diagnostics

- Virtual-8086 mode
  - hybrid of Protected
  - each program has its own 8086 computer (Real Mode)
Real-Address mode

- 1 MB RAM maximum addressable memory
- Application programs can access any area of memory
- Single tasking (with Interrupts)
- Supported by MS-DOS operating system
- Memory divided into 64K Segments

<table>
<thead>
<tr>
<th>Segment</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000:0000h to 0000:FFFFh</td>
<td></td>
</tr>
<tr>
<td>0001:0000h to 0001:FFFFh</td>
<td></td>
</tr>
</tbody>
</table>
Protected Mode

- 4 GB addressable RAM (00000000 to FFFFFFFFFFFh)
- Each program is assigned a memory partition which is protected from other programs
- Designed for multitasking & supported by Linux & MS-Windows
- Segment descriptor tables
  - Identifies locations of individual program segments
- Program structure (More information in Chapter 11)
  - code, data, and stack areas
  - CS, DS, SS segment descriptors
  - global descriptor table (GDT)
Handling Memory

**Real Mode**
Determining Logical Memory Address

- **CS** = 1234h
- **IP** = 0005h

\[ \text{CS}\left[\text{IP}\right] \text{ Combines to locate the next instruction} \]

Memory Location:
\[
\begin{array}{c}
12340h \\
0005h \\
\end{array}
\]
\[12345h = \text{Physical Memory Address}\]

**Protected Mode**
Determining Logical Memory Address

- **CS** = 0008h
- **IP** = 0005h

\[ \text{CS} = \text{Lookup Segment} \]

Memory Location:
\[
\begin{array}{c}
12340h \\
0005h \\
\end{array}
\]
\[12345h = \text{Physical Memory Address}\]
General-Purpose Registers

Named storage locations inside the CPU, optimized for speed.

### 32-bit General-Purpose Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAX</td>
<td></td>
</tr>
<tr>
<td>EBX</td>
<td></td>
</tr>
<tr>
<td>ECX</td>
<td></td>
</tr>
<tr>
<td>EDX</td>
<td></td>
</tr>
<tr>
<td>EBP</td>
<td></td>
</tr>
<tr>
<td>ESP</td>
<td></td>
</tr>
<tr>
<td>ESI</td>
<td></td>
</tr>
<tr>
<td>EDI</td>
<td></td>
</tr>
</tbody>
</table>

### 16-bit Segment Registers

<table>
<thead>
<tr>
<th>Segment</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td></td>
</tr>
<tr>
<td>DS</td>
<td></td>
</tr>
<tr>
<td>ES</td>
<td></td>
</tr>
<tr>
<td>FS</td>
<td></td>
</tr>
<tr>
<td>GS</td>
<td></td>
</tr>
</tbody>
</table>
Accessing Parts of Registers

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

<table>
<thead>
<tr>
<th>32-bit</th>
<th>16-bit</th>
<th>8-bit (high)</th>
<th>8-bit (low)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAX</td>
<td>AX</td>
<td>AH</td>
<td>AL</td>
</tr>
<tr>
<td>EBX</td>
<td>BX</td>
<td>BH</td>
<td>BL</td>
</tr>
<tr>
<td>ECX</td>
<td>CX</td>
<td>CH</td>
<td>CL</td>
</tr>
<tr>
<td>EDX</td>
<td>DX</td>
<td>DH</td>
<td>DL</td>
</tr>
</tbody>
</table>

Index and Base Registers

- Have only a 16-bit name for their lower half:

<table>
<thead>
<tr>
<th>32-bit</th>
<th>16-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESI</td>
<td>SI</td>
</tr>
<tr>
<td>EDI</td>
<td>DI</td>
</tr>
<tr>
<td>EBP</td>
<td>BP</td>
</tr>
<tr>
<td>ESP</td>
<td>SP</td>
</tr>
</tbody>
</table>

Some Specialized Register Uses

- **General-Purpose Registers**
  - EAX – accumulator
  - ECX – loop counter
  - ESP – stack pointer
  - ESI, EDI – index registers (ESI = Source, EDI = Destination)
  - EBP – extended frame pointer (stack)

- **Segment Registers**
  - CS – code segment
  - DS – data segment
  - SS – stack segment
  - ES, FS, GS - additional segments

- **EIP – instruction pointer**

- **EFLAGS**
  - status and control flags
  - each flag is a single binary bit

Control Flags

- Controls CPU Operation
  - Examples: Direction Flag, and Interrupt Flags

Status Flags

- Carry (CF)
  - unsigned arithmetic out of range
- Overflow (OF)
  - signed arithmetic out of range
- Sign (SF)
  - result is negative (if signed)
- Zero (ZF)
  - result is zero
- Auxiliary Carry
  - carry from bit 3 to bit 4 (special math)
- Parity
  - sum of bits is an even number
What's Next

★ General Concepts
★ 32-bit x86 Processors
• 64-bit x86 Processors
• Components of a Typical x86 Computer
★ Input-Output System
64-bit Processors

- Backward Compatible with x86 instruction set.
- 64-bit Addresses available
  - Currently uses 48-bit Physical Address space
  - Supports 256 Terabytes of RAM
- 64-bit Registers.
  - 64-bit Registers are proceeded by R (Example RAX, RBX)
- 16 General Purpose Registers
  - 8 More than 32-bit Mode (R8, R9, R10, etc…)
- 64-bit Status Flag (RFLAGS)
  - Share the same lower 32-bits as in 32-bit mode
  - Upper 32-bits are unused
- 64-bit Instruction Pointer (RIP)
What's Next

★ General Concepts
★ 32-bit x86 Processors
  • 64-bit x86 Processors
  • Components of a Typical x86 Computer
★ Input-Output System
Early Intel Microprocessors

- **Intel 8080**  (1974)
  - 64K addressable RAM, 8-bit registers

- **Intel 8086/8088**  (1978)
  - 1 MB addressable RAM
  - 16-bit registers/data bus

- **Intel 80286**  (1982)
  - 16 MB addressable RAM
  - First Protected mode Intel processor (Slide 11 & 12)

- **Intel386**  (1985)
  - 4 GB addressable RAM
  - 32-bit registers

- **Intel486**  (1989)
  - instruction pipelining
Modern Intel Processors

  - 32-bit address bus, 64-bit internal data path
- **Pentium II** (1997)
  - MMX (multimedia) instruction set
- **Pentium III** (1999)
  - SIMD (Single Instruction Multiple Data) instructions
- **Pentium 4 and Xeon** (2002)
  - Hyperthreading (Multi-Threads in parallel on multi-core)
- **Dual Core Processors (Core 2 Duo/Quad)** (2006)
  - Multiple cores (2-4), 64-bit and 32-bit versions
- **Core i7, i5, i3** (2008)
  - 2-6 processor cores, 64-bit, 8+ processing threads
- **Xeon E3, E5, E7** (2011+)
  - 4-15 processor cores, 64-bit
Intel D850MD Motherboard

- Video
- Audio chip
- PCI slots
- AGP slot
- Firmware hub
- I/O Controller
- Speaker
- Battery
- IDE drive connectors
- Memory controller hub
- Pentium 4 socket
- Dynamic RAM
- Power connector
- Diskette connector

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


Web site  Examples
Motherboard Chipset

A collection of chips designed to work together on a specific type of motherboard.

The motherboard chipset defines the capabilities of the motherboard:

- Includes a Memory Controller Hub for accessing specific types of memory (DDR, DDR2, DDR3, etc...)
- Includes I/O Controller Hub for controlling data transfer options:
  - Disk Drives (IDE, SATA, etc...)
- Supports other devices:
  - USB
  - AGP, PCI, PCI-Express
  - Audio
Video Output

- **Video controller**
  - Has a specialized processor for controlling video hardware
  - Can be on motherboard, or on expansion card
  - Data Transfer Options:
    - PCI (peripheral component interconnect) (0.1-0.5 GB/sec)
    - AGP (accelerated graphics port) (0.2-2.1 GB/sec)
    - PCI-Express (4-16 GB/sec)

- **Video memory (VRAM)**

- **Video CRT Display**
  - uses raster scanning
  - horizontal / vertical retrace

- **Digital LCD or LED monitors**
  - no raster scanning required
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
Input-Output Ports

• **USB (universal serial bus)**
  - up to 12 megabits/second
  - USB hub connects multiple devices with *enumeration*
  - supports *hot* connections

• **Parallel**
  - short cable, high speed (1 MB/sec)
  - common for printers
  - bidirectional, parallel data transfer

• **Serial**
  - one bit at a time
  - long cables, slow speed (19,200 bits/sec)

• **ATA & SATA**
  - Used to connect mass storage devices (hard drives, cdroms)
  - SATA – Serial ATA – bidirectional data transfer (100-600 MB/sec)

• **Firewire**
  - High speed external data transfer (800 MB/sec)
What's Next

⭐ General Concepts
⭐ 32-bit x86 Processors
  • 64-bit x86 Processors
  • Components of a Typical x86 Computer
⭐ Input-Output System
Levels of Input-Output

- **Level 3: Call a library function (C++, Java)**
  - easy to do; abstracted from hardware; details hidden
  - slowest performance
- **Level 2: Call an operating system function**
  - specific to one OS; device-independent
  - medium performance
- **Level 1: Call a BIOS (basic input-output system) function**
  - may produce different results on different systems
  - knowledge of hardware required
  - usually good performance
- **Level 0: Communicate directly with the hardware**
  - May not be allowed by some operating systems
Displaying a String of Characters

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

- Application Program
- OS Function
- BIOS Function
- Hardware
ASM Programming levels

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

- Irvine Library: Level 2 (More Portable)
- OS Function: Level 1
- BIOS Function: Level 0 (More Control, Speed)
- Hardware

Introduced in Chapter 5