Embedded Systems Programming

x86 Memory and Interrupt
(Module 8)

Yann-Hang Lee
Arizona State University
yhlee@asu.edu
(480) 727-7507

Summer 2014
X86 ISA Data Representations

- Little-endian byte ordering in memory
- Words, doublewords, and quadwords do not need to be aligned in memory on natural boundaries.
  - 2 memory accesses for an unaligned memory access
  - aligned accesses require only one
- Unsigned integer, signed (two's complement)
- FP, string of bits, bytes, .. etc.
- SIMD packed data
- Pointer
  - Near
  - Far (logical)
Programmer's model

Basic Program Execution Registers
- Eight 32-bit Registers
- General-Purpose Registers
- Six 16-bit Registers
- Segment Registers
- EFLAGS Register
- EIP (Instruction Pointer Register)

FPU Registers
- Eight 80-bit Registers
- Floating-Point Data Registers
- Control Register
- Status Register
- Tag Register
- Opcode Register (11-bits)
- FPU Instruction Pointer Register
- FPU Data (Operand) Pointer Register

Address Space

*The address space can be flat or segmented. Using the physical address extension mechanism, a physical address space of $2^{36} - 1$ can be addressed.
Modes of Operation

- **Protected mode (32 bits address)**
  - native mode (Windows, Linux), full features, separate memory
  - virtual-8086 mode

- **Real-address mode (20 bits address)**
  - the programming environment of the Intel 8086 processor with extensions
  - native MS-DOS

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

- **IA-32e (Intel 64 architecture)**
  - Compatibility mode - similar to 32-bit protected mode
  - 64-bit mode -
    - 16 64-bit general purpose registers
    - default address size is 64 bits and its default operand size is 32 bits.
Memory Model

- Flat memory model - a single, continuous linear address space of $2^{32}$ bytes
- Segmented model - a logical address consisting of a segment selector and an offset
- Real-address mode - for 8086, 
  - 16 segments of 64K
- Linear address space $\rightarrow$ (paging) physical space
Protected Mode Memory Management

- Use segment descriptor to protect memory accesses
- Each program has a descriptor table to map segments
  - allow shared segments
- Memory access checks
  - Limit, type, privilege level checks.
  - Restrictions of addressable domain, procedure entry-points, and instruction set.

<table>
<thead>
<tr>
<th>Access</th>
<th>Limit</th>
<th>Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Logical addresses

- SS 0018 0000002A
- DS 0010 000001B6
- IP 0008 00002CD3

Local Descriptor Table

- Linear address space
- Unused
- DRAM
- 0001A000
- 00002A00
- 00001A00
- 00003000
- LDTR register
Virtual Memory and Paging

- **Virtual memory**
  - Uses disk as part of the memory, thus allowing sum of all programs can be larger than physical memory
  - Only part of a program must be kept in memory, while the remaining parts are kept on disk.

- **The memory used by the program is divided into small units called pages (4096-byte).**
  - OS maintains page directory and page tables
  - Page translation: CPU converts the linear address into a physical address
  - Page fault: occurs when a needed page is not in memory, and the CPU interrupts the program

- **Virtual memory manager (VMM) - OS utility that manages the loading and unloading of pages**
A linear address is divided into a page directory field, page table field, and page frame offset.

The CPU uses all three to calculate the physical address.
Interrupt and Exception

- **Interrupt**
  - an asynchronous event that is typically triggered by an I/O device.

- **Exception**
  - a synchronous event that is generated when the processor detects one or more predefined conditions while executing an instruction.
  - three classes of exceptions: faults, traps, and aborts.

- **18 predefined interrupts and exceptions and 224 user defined interrupts**

- **Access handler procedures through entries in the interrupt descriptor table (IDT)**
  - A call to a handler procedure is similar to a procedure call to another protection level
Interrupt and Exception

- **Interrupt vector references**
  - an interrupt gate (interrupt enable (IF) flag in the EFLAGS register is cleared)
  - a trap gate

- **Gate contains**
  - access rights information
  - segment selector for the code segment of the handler procedure
  - an offset into the code segment to entry point of the handler procedure

<table>
<thead>
<tr>
<th>Vector No.</th>
<th>Mnemonic</th>
<th>Description</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>#DE</td>
<td>Divide Error</td>
<td>DIV and IDIV instructions.</td>
</tr>
<tr>
<td>1</td>
<td>#DB</td>
<td>Debug</td>
<td>Any code or data reference.</td>
</tr>
<tr>
<td>2</td>
<td>#NMI</td>
<td>NMI interrupt</td>
<td>Non-maskable external interrupt.</td>
</tr>
<tr>
<td>3</td>
<td>#BP</td>
<td>Breakpoint</td>
<td>INT 3 instruction.</td>
</tr>
<tr>
<td>4</td>
<td>#OF</td>
<td>Overflow</td>
<td>INTO instruction.</td>
</tr>
<tr>
<td>5</td>
<td>#BR</td>
<td>BOUND Range Exceeded</td>
<td>BOUND instruction.</td>
</tr>
<tr>
<td>6</td>
<td>#UD</td>
<td>Invalid Opcode (Undefined Opcode)</td>
<td>UD2 instruction or reserved opcode.</td>
</tr>
<tr>
<td>7</td>
<td>#NM</td>
<td>Device Not Available (No Math Coprocessor)</td>
<td>Floating-point or WAIT/FWAIT instruction.</td>
</tr>
<tr>
<td>8</td>
<td>#DF</td>
<td>Double Fault</td>
<td>Any instruction that can generate an exception, an NMI, or an INTR.</td>
</tr>
<tr>
<td>9</td>
<td>#MF</td>
<td>CoProcessor Segment Overrun (reserved)</td>
<td>Floating-point instruction.</td>
</tr>
<tr>
<td>10</td>
<td>#TS</td>
<td>Invalid TSS</td>
<td>Task switch or TSS access.</td>
</tr>
<tr>
<td>11</td>
<td>#NP</td>
<td>Segment Not Present</td>
<td>Loading segment registers or accessing system segments.</td>
</tr>
<tr>
<td>12</td>
<td>#SS</td>
<td>Stack Segment Fault</td>
<td>Stack operations and SS register loads.</td>
</tr>
<tr>
<td>13</td>
<td>#GP</td>
<td>General Protection</td>
<td>Any memory reference and other protection checks.</td>
</tr>
<tr>
<td>14</td>
<td>#PF</td>
<td>Page Fault</td>
<td>Any memory reference.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>#MF</td>
<td>Floating-Point Error (Math Fault)</td>
<td>Floating-point or WAIT/FWAIT instruction.</td>
</tr>
<tr>
<td>17</td>
<td>#AC</td>
<td>Alignment Check</td>
<td>Any data reference in memory.</td>
</tr>
<tr>
<td>18</td>
<td>#MC</td>
<td>Machine Check</td>
<td>Error codes (if any) and source are model dependent.</td>
</tr>
<tr>
<td>19</td>
<td>#XM</td>
<td>SIMD Floating-Point Exception</td>
<td>SIMD Floating-Point Instruction</td>
</tr>
<tr>
<td>20-31</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-255</td>
<td>Reserved</td>
<td>Maskable Interrupts</td>
<td>External interrupt from INTR pin or INT n instruction.</td>
</tr>
</tbody>
</table>