Intel Embedded Processor

Yann-Hang Lee

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ 85287

yhlee@asu.edu
(480) 727-7507
Current Processor Design

- Moore’s law continues to hold true, transistor counts doubling every 18 months
  - But can no longer rely upon increasing clock rates and instruction-level parallelism to meet computing performance demands
- How to best exploit ever-increasing on-chip transistor counts?
  - Multi- & many-core (MC) devices are new technology wave
  - exploiting explicit parallelism in the new devices
- Power constraints
Intel Processors

- **X86 32/64 architecture**
  - Pentium microprocessor was the first x86 superscalar CPU

- **Processors for**
  - Server (Xeon), desktop (Core i3/i5/i7), mobile (Core i3/i5/i7), and embedded (Atom)
  - All of them support hypervisor (VM)

- **Differences**
  - CPUs, memory, and interconnection bandwidth
  - reliability (quality of dies) and form factor
  - power and thermal requirements

- **Uses available clock cycles and power, not to push up higher clock speeds and energy needs**
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)
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 → (paging) physical space
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.
Programmer’s model

Basic Program Execution Registers

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

FPU Registers

- Eight 80-bit Registers
- Floating-Point Data Registers

- 16 bits
- Control Register
- Status Register
- Tag Register
- Opcode Register (11-bits)
- 48 bits
- FPU Instruction Pointer Register
- FPU Data (Operand) Pointer Register

Address Space

- $2^{32} - 1$

*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.
## General-Purpose Registers

### 32-bit General-Purpose Registers

- EAX – accumulator
- EBX
- ECX – loop counter
- EDX
- ESP – stack pointer
- ESI, EDI – index registers
- EBP – extended frame pointer

### 16-bit Segment Registers

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

---

Computer Science and Engineering, ASU
Accessing Parts of Registers

- **Compatibility**
  - 8080 (A, B, C, D, H, L)
  - 8086 (Ax, BX, CX, DX, SI, DI, SP, BP)
- Use 8-bit name, 16-bit name, or 32-bit name

<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>

8 bits + 8 bits

16 bits

32 bits

Computer Science and Engineering, ASU
Stack and Procedure Call

- **CALL and RET**
  - Near calls – to local procedures within the currently running program or task.
  - Far calls – to operating system procedures or procedures in a different task.

- **ENTER and LEAVE**
  - Creates a stack frame compatible with the scope rules.

- **A full-descending stack**
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
  - Restriction of procedure entry-points.
  - Restriction of instruction set.

<table>
<thead>
<tr>
<th>Access</th>
<th>Limit</th>
<th>Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>0018</td>
<td>0000002A</td>
</tr>
<tr>
<td>DS</td>
<td>0010</td>
<td>000001B6</td>
</tr>
<tr>
<td>IP</td>
<td>0008</td>
<td>00002CD3</td>
</tr>
</tbody>
</table>

Logical addresses

Local Descriptor Table

LDTR register

Linear address space
Privilege Levels

- 4 privilege levels to improve the reliability of operating systems
- Gate to control calls to modules operating at higher privilege segments – call gate descriptor
- Each privilege level has its own stack.
  - Save registers and parameters in the stack of called procedure
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**
Page Translation

- 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.1</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.2</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.3</td>
</tr>
<tr>
<td>18</td>
<td>#MC</td>
<td>Machine Check</td>
<td>Error codes (if any) and source are model dependent.4</td>
</tr>
<tr>
<td>19</td>
<td>#XM</td>
<td>SIMD Floating-Point Exception</td>
<td>SIMD Floating-Point Instruction5</td>
</tr>
<tr>
<td>20-31</td>
<td>Reserved</td>
<td>Maskable Interrupts</td>
<td>External interrupt from INTR pin or INT n instruction.</td>
</tr>
<tr>
<td>32-255</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Interrupt and APIC

- **Interrupt in 8086**
  - Two pins: NMI and INTR
  - *Interrupt Acknowledge Cycle* to fetch the interrupt vector number from 8259

- **APIC**
  - In Pentium and P6 processors
  - Receives interrupts and send to core for handling
  - APIC bus: bi-directional data signals (APICD[1:0]) and clock (APICCLK)
  - Inter-processor interrupt messages for multi-processor systems
  - Static and dynamic (based on the priority of executing tasks) distribution
Interrupt Handling

- IO APIC delivers interrupt message to local APIC
  - Programmable vector number for each interrupt source
- Implied priority based on vector number
  - local APIC determines when to service the interrupt relative to the other activities of the processor
  - priority = vector / 16
- Locate gate from IDT
  - Far call to the handler
  - (SS, ESP), EFLAGS, CS, EIP, and Error code are saved in stack
Hardware Initialization and Reset

- **Reset processor state**
  - EIP=0000FFF0H, CS=F000H(segment) and FFFF0000H (base)
  - Disable paging, cache, and in real-address mode

- **Execute the first instruction at physical address FFFFFFFF0H.**
  - The EPROM containing the software initialization code or BIOS should be located at the upper memory space (including this address)
  - Run in real-mode, invalidate the TLBs, set up a GDT for selector 0x08 (code) and 0x10 (data), switch to protected mode
  - Start other components on motherboard (FPU, APIC, southbridge, etc.)
System Architecture

- **Chipset**
  - North Bridge
  - South Bridge
  - Firmware Hub

- **Various chipsets available from Intel to meet performance requirements**

- **FSB, DMI/Hub interface**

- **System control hub (SCH) – GMCH and ICH are merged into one chip**
Northbridge

- **Memory Controller Hub (MCH)**
  - Interfaces between the CPU and the rest of the system
    - Memory
    - AGP Bus
    - Hublink Bus

- **Graphics and Memory Controller Hub (GMCH)**
  - Includes integrated graphics accelerator
    - Supports Direct AGP - fully integrated graphics engine or AGP 2.0 (with AGP slot)
    - Support for analog video, Digital Video Out (DVO) and Display Data Channel (DDC)

- **Design requirement:**
  - Fast burst data transfer – throughput and latency
Southbridge

- The South Bridge or I/O Controller Hub (ICH)
  - Interfaces to I/O devices
    - PCI Bus
    - IDE
    - USB
    - LPC bus to Firmware Hub and Super I/O (Legacy I/O)

- The Firmware Hub (FWH)
  - Stores BIOS code/data in 512KB or 1MB flash memory
  - Random number generator
  - Can be reprogrammed in place