Outline

- Pentium family history
- Pentium processor details
- Pentium registers
  - Data
  - Pointer and index
  - Control
  - Segment
- Real mode memory architecture
- Protected mode memory architecture
  - Segment registers
  - Segment descriptors
  - Segment descriptor tables
  - Segmentation models
- Mixed-mode operation
- Default segment registers used
Pentium Family

• Intel introduced microprocessors in 1969
  * 4-bit microprocessor 4004
  * 8-bit microprocessors
    » 8080
    » 8085
  * 16-bit processors
    » 8086 introduced in 1979
      – 20-bit address bus, 16-bit data bus
    » 8088 is a less expensive version
      – Uses 8-bit data bus
    » Can address up to 4 segments of 64 KB
    » Referred to as the real mode
Pentium Family (cont’d)

* 80186
  » A faster version of 8086
  » 16-bit data bus and 20-bit address bus
  » Improved instruction set

* 80286 was introduced in 1982
  » 24-bit address bus
  » 16 MB address space
  » Enhanced with memory protection capabilities
  » Introduced protected mode
    – Segmentation in protected mode is different from the real mode
  » Backwards compatible
Pentium Family (cont’d)

* 80386 was introduced 1985
   » First 32-bit processor
   » 32-bit data bus and 32-bit address bus
   » 4 GB address space
   » Segmentation can be turned off (flat model)
   » Introduced paging

* 80486 was introduced 1989
   » Improved version of 386
   » Combined coprocessor functions for performing floating-point arithmetic
   » Added parallel execution capability to instruction decode and execution units
      – Achieves scalar execution of 1 instruction/clock
   » Later versions introduced energy savings for laptops
Pentium Family (cont’d)

* Pentium (80586) was introduced in 1993
  » Similar to 486 but with 64-bit data bus
  » Wider internal datapaths
    – 128- and 256-bit wide
  » Added second execution pipeline
    – Superscalar performance
    – Two instructions/clock
  » Doubled on-chip L1 cache
    – 8 KB data
    – 8 KB instruction
  » Added branch prediction
Pentium Family (cont’d)

* Pentium Pro was introduced in 1995
  » Three-way superscalar
    – 3 instructions/clock
  » 36-bit address bus
    – 64 GB address space
  » Introduced dynamic execution
    – Out-of-order execution
    – Speculative execution
  » In addition to the L1 cache
    – Has 256 KB L2 cache
Pentium Family (cont’d)

* Pentium II was introduced in 1997
  » Introduced multimedia (MMX) instructions
  » Doubled on-chip L1 cache
    – 16 KB data
    – 16 KB instruction
  » Introduced comprehensive power management features
    – Sleep
    – Deep sleep
  » In addition to the L1 cache
    – Has 256 KB L2 cache

* Pentium III, Pentium IV,…
Pentium Family (cont’d)

* Itanium processor
  » RISC design
    – Previous designs were CISC
  » 64-bit processor
  » Uses 64-bit address bus
  » 128-bit data bus
  » Introduced several advanced features
    – Speculative execution
    – Predication to eliminate branches
    – Branch prediction
Pentium Processor

Clock
- CLK

Initialization
- Reset
- Init

Interrupts
- INTR
- NMI

Bus arbitration
- BOFF#
- BREQ
- HOLD
- HLDA

Cache control
- KEN#
- WB/WT#

Address parity
- AP
- APCHK#

Data parity
- DP0-DP7
- PCHK#
- PEN#

Address bus A3-A31

Byte enable BE0#-BE7#

Data bus D0-D63

BRDY#

M/IO#

W/R#

CACHE#

LOCK#

D/C#

Bus cycle definition

Chapter 7: Page 10

Pentium Processor (cont’d)

• Data bus (D0 – D 63)
  * 64-bit data bus

• Address bus (A3 – A31)
  * Only 29 lines
    » No A0-A2 (due to 8-byte wide data bus)

• Byte enable (BE0# - BE7#)
  * Identifies the set of bytes to read or write
    » BE0# : least significant byte (D0 – D7)
    » BE1# : next byte (D8 – D15)
    » …
    » BE7# : most significant byte (D56 – D63)
  * Any combination of bytes can be specified
Pentium Processor (cont’d)

- **Data parity (DP0 – DP7)**
  - Even parity for 8 bytes of data
    - DP0 : D0 – D7
    - DP1 : D8 – D15
    - ...
    - DP7 : D56 – D63

- **Parity check (PCHK#)**
  - Indicates the parity check result on data read
  - Parity is checked only for valid bytes
    - Indicated by BE# signals
Pentium Processor (cont’d)

• Parity enable (PEN#)
  * Determines whether parity check should be used
• Address parity (AP)
  * Bad address parity during inquire cycles
• Memory/IO (M/IO#)
  * Defines bus cycle: memory or I/O
• Write/Read (W/R#)
  * Distinguishes between write and read cycles
• Data/Code (D/C#)
  * Distinguishes between data and code
Pentium Processor (cont’d)

• Cacheability (CACHE#)
  * Read cycle: indicates internal cacheability
  * Write cycle: burst write-back

• Bus lock (LOCK#)
  * Used in read-modify-write cycle
  * Useful in implementing semaphores

• Interrupt (INTR)
  * External interrupt signal

• Nonmaskable interrupt (NMI)
  * External NMI signal
Pentium Processor (cont’d)

• Clock (CLK)
  * System clock signal

• Bus ready (BRDY#)
  * Used to extend the bus cycle
    » Introduces wait states

• Bus request (BREQ)
  * Used in bus arbitration

• Backoff (BOFF#)
  * Aborts all pending bus cycles and floats the bus
  * Useful to resolve deadlock between two bus masters
Pentium Processor (cont’d)

- **Bus hold (HOLD)**
  - Completes outstanding bus cycles and floats bus
  - Asserts HLDA to give control of bus to another master

- **Bus hold acknowledge (HLDA)**
  - Indicates the Pentium has given control to another local master
  - Pentium continues execution from its internal caches

- **Cache enable (KEN#)**
  - If asserted, the current cycle is transformed into cache line fill
Pentium Processor (cont’d)

- Write-back/Write-through (WB/WT#)
  * Determines the cache write policy to be used

- Reset (RESET)
  * Resets the processor
  * Starts execution at FFFFFFF0H
  * Invalidates all internal caches

- Initialization (INIT)
  * Similar to RESET but internal caches and FP registers are not flushed
  * After powerup, use RESET (not INIT)
Pentium Registers

- Four 32-bit registers can be used as
  * Four 32-bit register (EAX, EBX, ECX, EDX)
  * Four 16-bit register (AX, BX, CX, DX)
  * Eight 8-bit register (AH, AL, BH, BL, CH, CL, DH, DL)

- Some registers have special use
  * ECX for count in loop instructions
Pentium Registers (cont’d)

• Two index registers
  * 16- or 32-bit registers
  * Used in string instructions
    » Source (SI) and
destination (DI)
  * Can be used as general-
purpose data registers

• Two pointer registers
  * 16- or 32-bit registers
  * Used exclusively to
  maintain the stack
Pentium Registers (cont’d)

Flags register

EFLAGS

Status flags
CF = Carry flag
PF = Parity flag
AF = Auxiliary carry flag
ZF = Zero flag
SF = Sign flag
OF = Overflow flag

Control flags
DF = Direction flag

System flags
TF = Trap flag
IF = Interrupt flag
IOPL = I/O privilege level
NT = Nested task
RF = Resume flag
VM = Virtual 8086 mode
AC = Alignment check
VIF = Virtual interrupt flag
VIP = Virtual interrupt pending
ID = ID flag

Instruction pointer

EIP  IP
Pentium Registers (cont’d)

• Control registers
  * (E)IP
    » Program counter
  * (E) FLAGS
    » Status flags
      – Record status information about the result of the last arithmetic/logical instruction
    » Direction flag
      – Forward/backward direction for data copy
    » System flags
      – IF : interrupt enable
      – TF : Trap flag (useful in single-stepping)
Pentium Registers (cont’d)

• Segment register
  * Six 16-bit registers
  * Support segmented memory architecture
  * At any time, only six segments are accessible
  * Segments contain distinct contents
    » Code
    » Data
    » Stack

<table>
<thead>
<tr>
<th></th>
<th>Code segment</th>
<th>Data segment</th>
<th>Stack segment</th>
<th>Extra segment</th>
<th>Extra segment</th>
<th>Extra segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Real Mode Architecture

- Pentium supports two modes
  - Real mode
    » Uses 16-bit addresses
    » Runs 8086 programs
    » Pentium acts as a faster 8086
  - Protected mode
    » 32-bit mode
    » Native mode of Pentium
    » Supports segmentation and paging
Real Mode Architecture (cont’d)

- Segmented organization
  - 16-bit wide segments
  - Two components
    » Base (16 bits)
    » Offset (16 bits)

- Two-component specification is called \textit{logical address}
  - Also called \textit{effective address}

- 20-bit \textit{physical address}
Real Mode Architecture (cont’d)

- Conversion from logical to physical addresses

11000 (add 0 to base)  +  450 (offset)  
\[11450\] (physical address)
Real Mode Architecture (cont’d)

Two logical addresses map to the same physical address
Real Mode Architecture (cont’d)

• Programs can access up to six segments at any time
  • Two of these are for
    * Data
    * Code
  • Another segment is typically used for
    * Stack
  • Other segments can be used for
    * data, code,..
Real Mode Architecture (cont’d)

(a) Adjacent  (b) Disjoint  (c) Partially overlapped  (d) Fully overlapped
Protected Mode Architecture

- Supports sophisticated segmentation
- Segment unit translates 32-bit logical address to 32-bit linear address
- Paging unit translates 32-bit linear address to 32-bit physical address
  - If no paging is used
    - Linear address = physical address
Protected Mode Architecture (cont’d)

Address translation

SEGMENT SELECTOR

INDEX TI RPL

OFFSET

31 0

SEGMENT descriptor

DESCRIPOR TABLE

ACCESS RIGHTS

LIMIT

BASE ADDRESS

ADDER

32-bit base address

LINEAR ADDRESS

31 0
Protected Mode Architecture (cont’d)

• Index
  * Selects a descriptor from one of two descriptor tables
    » Local
    » Global

• Table Indicator (TI)
  * Select the descriptor table to be used
    » 0 = Local descriptor table
    » 1 = Global descriptor table

• Requestor Privilege Level (RPL)
  * Privilege level to provide protected access to data
    » Smaller the RPL, higher the privilege level
### Protected Mode Architecture (cont’d)

- **Visible part**
  - Instructions to load segment selector
    - `mov, pop, lds, les, lss, lgs, lfs`

- **Invisible**
  - Automatically loaded when the visible part is loaded from a descriptor table

<table>
<thead>
<tr>
<th>Visible part</th>
<th>Invisible part</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment selector</td>
<td>Segment base address, size, access rights, etc.</td>
<td>CS</td>
</tr>
<tr>
<td>Segment selector</td>
<td>Segment base address, size, access rights, etc.</td>
<td>SS</td>
</tr>
<tr>
<td>Segment selector</td>
<td>Segment base address, size, access rights, etc.</td>
<td>DS</td>
</tr>
<tr>
<td>Segment selector</td>
<td>Segment base address, size, access rights, etc.</td>
<td>ES</td>
</tr>
<tr>
<td>Segment selector</td>
<td>Segment base address, size, access rights, etc.</td>
<td>FS</td>
</tr>
<tr>
<td>Segment selector</td>
<td>Segment base address, size, access rights, etc.</td>
<td>GS</td>
</tr>
</tbody>
</table>
### Protected Mode Architecture (cont’d)

#### Segment descriptor

<table>
<thead>
<tr>
<th>31</th>
<th>16</th>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **BASE 31:24**
- **G**
- **D**
- **B**
- **AVL**
- **LIMIT 19:16**
- **P**
- **DPL**
- **S**
- **TYPE**
- **BASE 23:16**

+4

---

Protected Mode Architecture (cont’d)

- **Base address**
  * 32-bit segment starting address

- **Granularity (G)**
  * Indicates whether the segment size is in
    » 0 = bytes, or
    » 1 = 4KB

- **Segment Limit**
  * 20-bit value specifies the segment size
    » G = 0: 1byte to 1 MB
    » G = 1: 4KB to 4GB, in increments of 4KB
Protected Mode Architecture (cont’d)

• D/B bit
  * Code segment
    » D bit: default size operands and offset value
      – D = 0: 16-bit values
      – D = 1: 32-bit values
  * Data segment
    » B bit: controls the size of the stack and stack pointer
      – B = 0: SP is used with an upper bound of FFFFH
      – B = 1: ESP is used with an upper bound of FFFFFFFFH
  * Cleared for real mode
  * Set for protected mode
Protected Mode Architecture (cont’d)

• S bit
  * Identifies whether
    » System segment, or
    » Application segment

• Descriptor privilege level (DPL)
  * Defines segment privilege level

• Type
  * Identifies type of segment
    » Data segment: read-only, read-write, …
    » Code segment: execute-only, execute/read-only, …

• P bit
  * Indicates whether the segment is present
Protected Mode Architecture (cont’d)

- Three types of segment descriptor tables
  - Global descriptor table (GDT)
    » Only one in the system
    » Contains OS code and data
    » Available to all tasks
  - Local descriptor table (LDT)
    » Several LDTs
    » Contains descriptors of a program
  - Interrupt descriptor table (IDT)
    » Used in interrupt processing
    » Details in Chapter 20
Protected Mode Architecture (cont’d)

- **Segmentation Models**
  - Pentium can turn off segmentation
  - Flat model
    - Consists of one segment of 4GB
    - E.g. used by UNIX
  - Multisegment model
    - Up to six active segments
    - Can have more than six segments
    - Descriptors must be in the descriptor table
    - A segment becomes active by loading its descriptor into one of the segment registers
Protected Mode Architecture (cont’d)
Mixed-Mode Operation

- Pentium allows mixed-mode operation
  - Possible to combine 16-bit and 32-bit operands and addresses
  - D/B bit indicates the default size
    - 0 = 16 bit mode
    - 1 = 32-bit mode
  - Pentium provides two override prefixes
    - One for operands
    - One for addresses

- Details and examples in Chapter 11
Default Segments

- Pentium uses default segments depending on the purpose of the memory reference
  - Instruction fetch
    » CS register
  - Stack operations
    » 16-bit mode: SP
    » 32-bit mode: ESP
  - Accessing data
    » DS register
    » Offset depends on the addressing mode