Intel IA32 OS Support
-*Refresh*

Tore Brox-Larsen

We need to …

- Allow OS to control system
- Allow OS to interact with I/O
- Allow other parts to request OS services
- Protect OS from other parts of system
- Protect user processes from each other
- Protect I/O devices from access by other parts
- Key resources
  - CPU, Memory, I/O devices

Areas of OS Support

- Memory management
- Protection of SW modules
- Multitasking
- Exception & interrupt handling
- Multiprocessing
- Cache management
- Hardware resource & power management
- Debugging, performance monitoring

Fundamental Concepts

- Privilege levels
- HW-level exceptional control flow
  - Interrupts
  - Traps

Privilege Levels

- An fundamental concept for protection
- Intel offers four levels of privilege (0—3), at any instant in time, the processor is logically at exactly one level
- Many OS'es use only the highest (0), and the lowest (3) levels
  - Classic UNIX position. Portability reasons
- High privilege
  - You can do anything
  - Used for the OS kernel
- Low privilege
  - You can do less
  - Used for application programs
  - Used to protect other modules and the kernel from the application programs
- Privilege levels are utilized by other system mechanisms
- *How about privilege levels when executing more than one instruction at any time?*

Protecting Instructions

- Privileged instructions are legal only when processor is at the corresponding privilege level
- If the processor is not at a sufficiently high privilege level, the instruction faults, and the OS is invoked and notified that an illegal instruction was issued

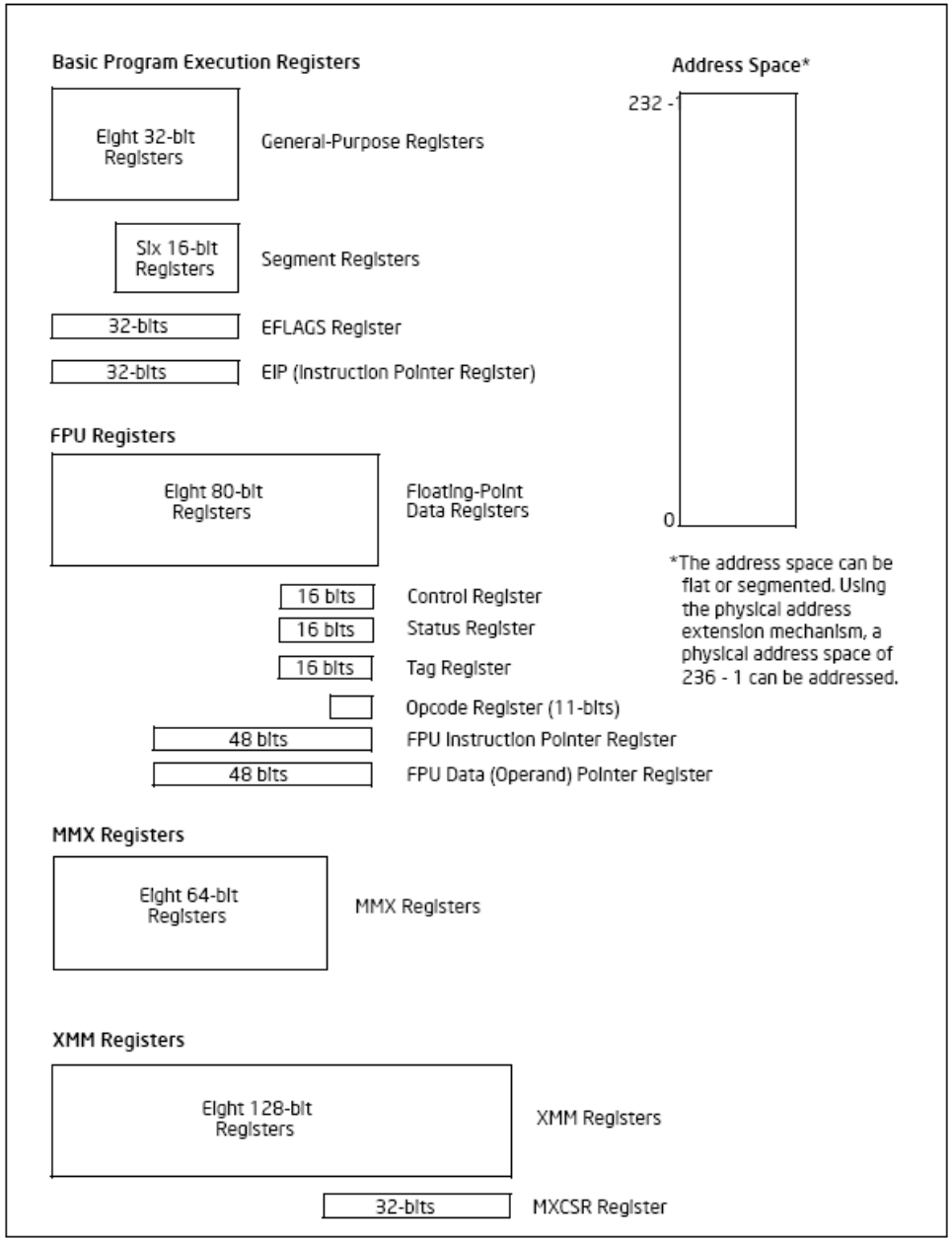
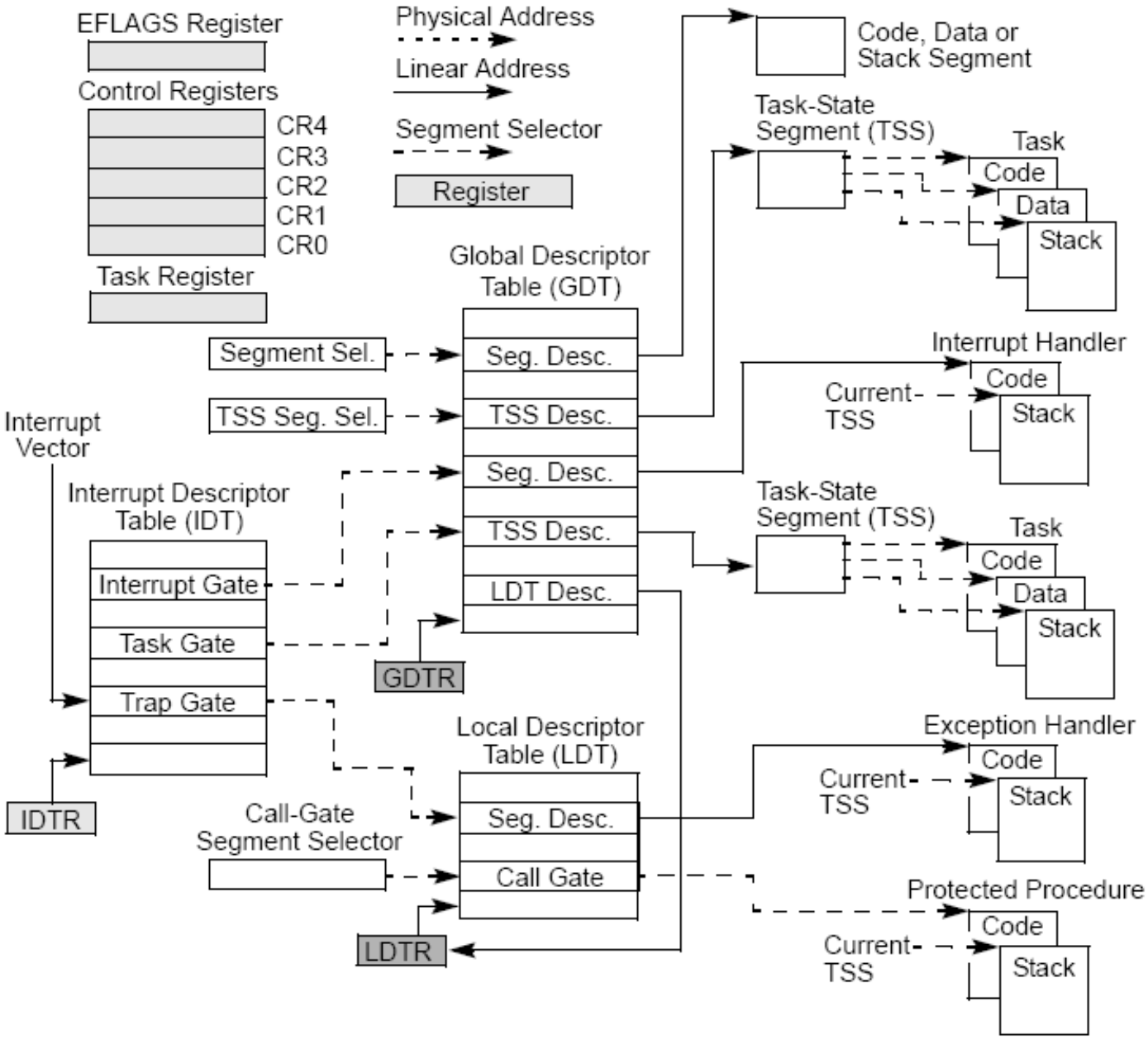
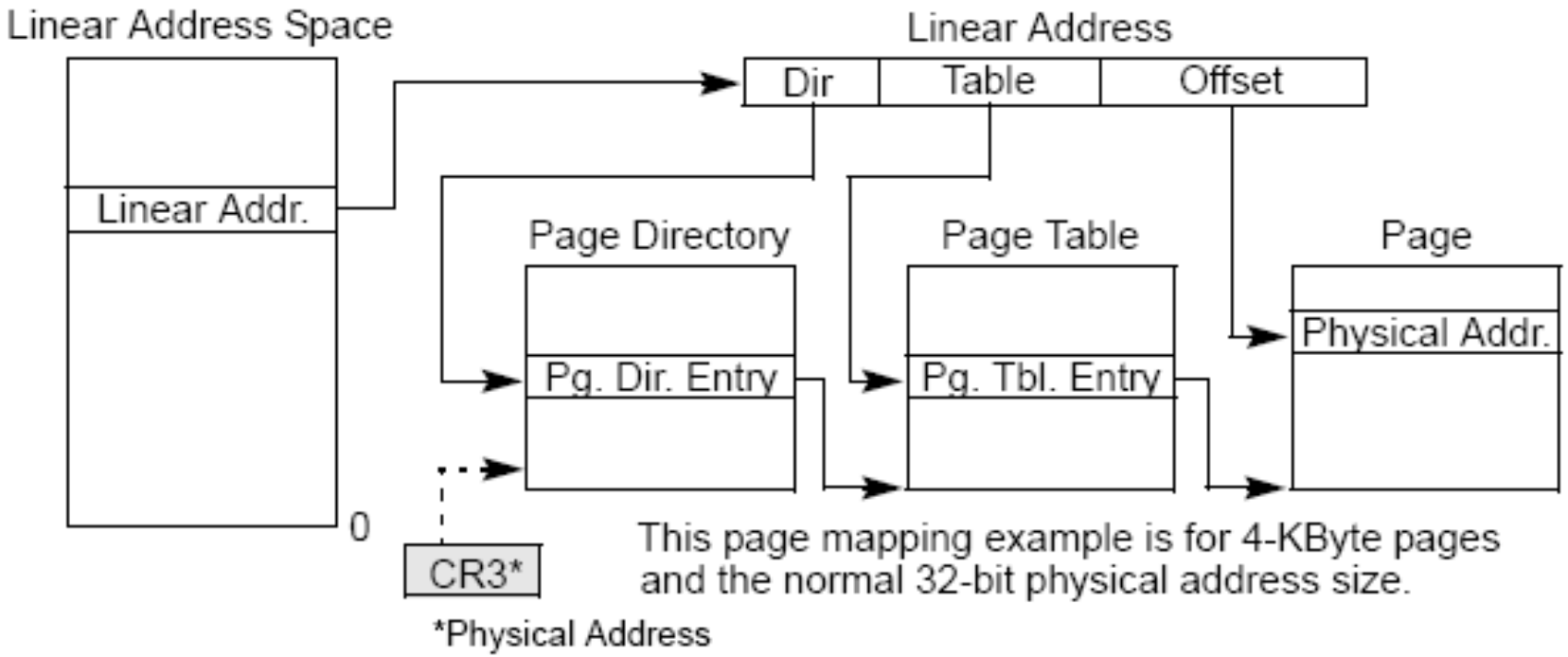


Figure 3-1. IA-32 Basic Execution Environment for Non-64-bit Modes


EFLAGS Register
Physical Address
Linear Address
Segment Selector
Register
Control Registers
CR4
CR3
CR2
CR1
CR0
Task Register
Global Descriptor Table (GDT)
Code, Data or Stack Segment
Task-State Segment (TSS)
Task
Code
Data
Stack
Segment Sel.
Seg. Desc.
TSS Seg. Sel.
TSS Desc.
Seg. Desc.
TSS Desc.
LDT Desc.
Interrupt Handler
Code
Stack
Current-TSS
Interrupt Vector
Interrupt Descriptor Table (IDT)
Interrupt Gate
Task Gate
Trap Gate
GDTR
IDTR
Local Descriptor Table (LDT)
Call-Gate Segment Selector
Seg. Desc.
Call Gate
LDTR
Exception Handler
Protected Procedure

System Level Registers & Data Structures (ii)



Example Privileged Registers

- Flag register, EFLAG
  - Processor state indicator bits
- Control registers
  - CR0 System control flags
  - CR2 Page fault linear address
  - CR3 Page directory base
  - CR4 Extensions
- Instruction pointer register, EIP

Example System Instructions

Table 2-2. Summary of System Instructions

Instruction	Description	Useful to Application?	Protected from Application?
LLDT	Load LDT Register	No	Yes
SLDT	Store LDT Register	No	No
LGDT	Load GDT Register	No	Yes
SGDT	Store GDT Register	No	No
LTR	Load Task Register	No	Yes
STR	Store Task Register	No	No

Tasks

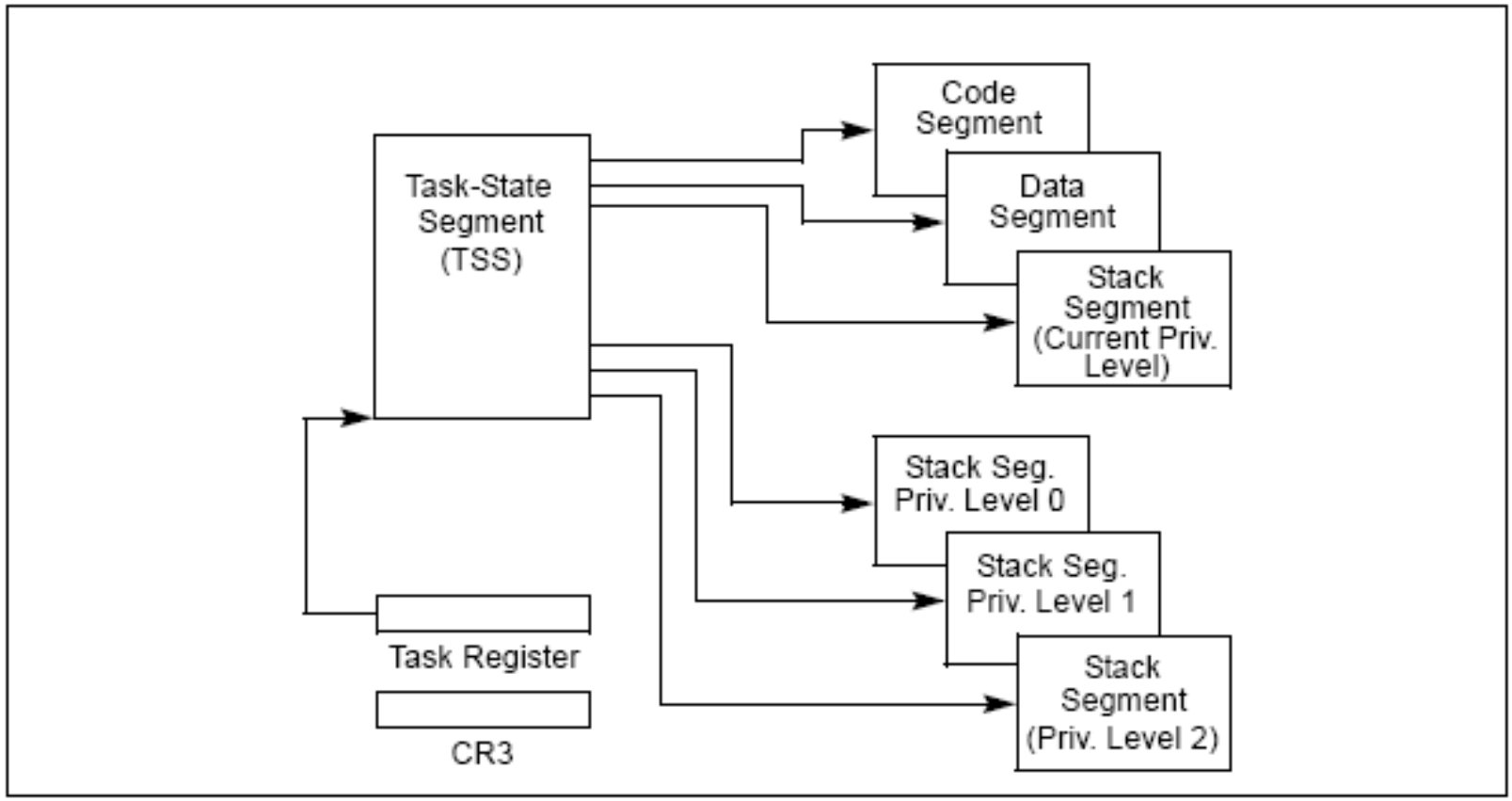


Figure 6-1. Structure of a Task

Transferring Control Across Privilege Levels

- Gates
  - Call gates
  - Trap gates
  - Interrupt gates
  - Task gates
- Sysenter/sysexit

HW Level Exceptional Control Flow *(Intel Terminology)*

- Interrupt
  - Asynchronous event, typically triggered by an I/O device
  - Pending interrupts checked (and handled) before each instruction fetch
  - Eventually resumes at next instruction
- Exception
  - Synchronous event, triggered by the processing of an instruction
- Three classes of exceptions
  - Trap
    - Intentional. Resumes at next instruction
  - Fault
    - Error conditions that may be handled by handler. Re-executes faulting instruction if doable, otherwise makes sure faulting process is aborted
  - Abort
    - Unrecoverable, fatal error. Terminates process, …possibly also system
- IA-32 supports precise interrupts

Actions taken on interrupt/exception

- Processor pushes status information (SS, EFLAGS register, CS (code segment register), and return address (EIP) onto stack
- Pushes error code (if appropriate) onto stack
- Service routine executed
- Return from interrupt instruction restores state of interrupted process

Interrupt Description Table (IDT)

- Used to locate appropriate service routine for each interrupt type
- Used similarly for interrupts and exceptions
- IDTR register specifies *base address & length* of IDT
- Intel uses *vectored interrupts*. The vector is a number provided by the interrupting unit that identifies the interrupt type. The vector (scaled by eight) is used to index into the IDT
- Each entry in IDT is an eight-byte descriptor which contains a segment selector for the handler's code segment, an offset within the code segment, and access rights information

IDT entries

- Max 256 entries
- 0—31: Architecture-defined, or reserved by Intel for future use
- 32—255: ”User defined,” i.e. OS-defined

Interrupt priorities

- Simultaneous exceptions and interrupts are prioritized in eight classes
- Highest class (1)
  - Hardware reset, machine check
- Lowest class (8)
  - Faults on executing an instruction

Masking

- IF- and RF-flags in the EFLAGS register may be used to inhibit the generation of some interrupts

Some interrupt handling instructions

- LIDT
  - Load IDTR register (32 bit) from memory. Privileged instruction (CPL = 0)
- SIDT
  - Store IDTR register (32 bit) to memory. Non-privileged instruction
- INT
  - Explicit call to any specific exception
- INTO, INT 3, BOUND
  - Allow SW exception checking. Respectively Overflow, Breakpoint, and Range
- CLI, STI
  - Clear/Set Interrupt Enable Flag
- PUSHF, POPF
  - Push/pop Flags on/off stack
- IRET
  - Return from interrupt

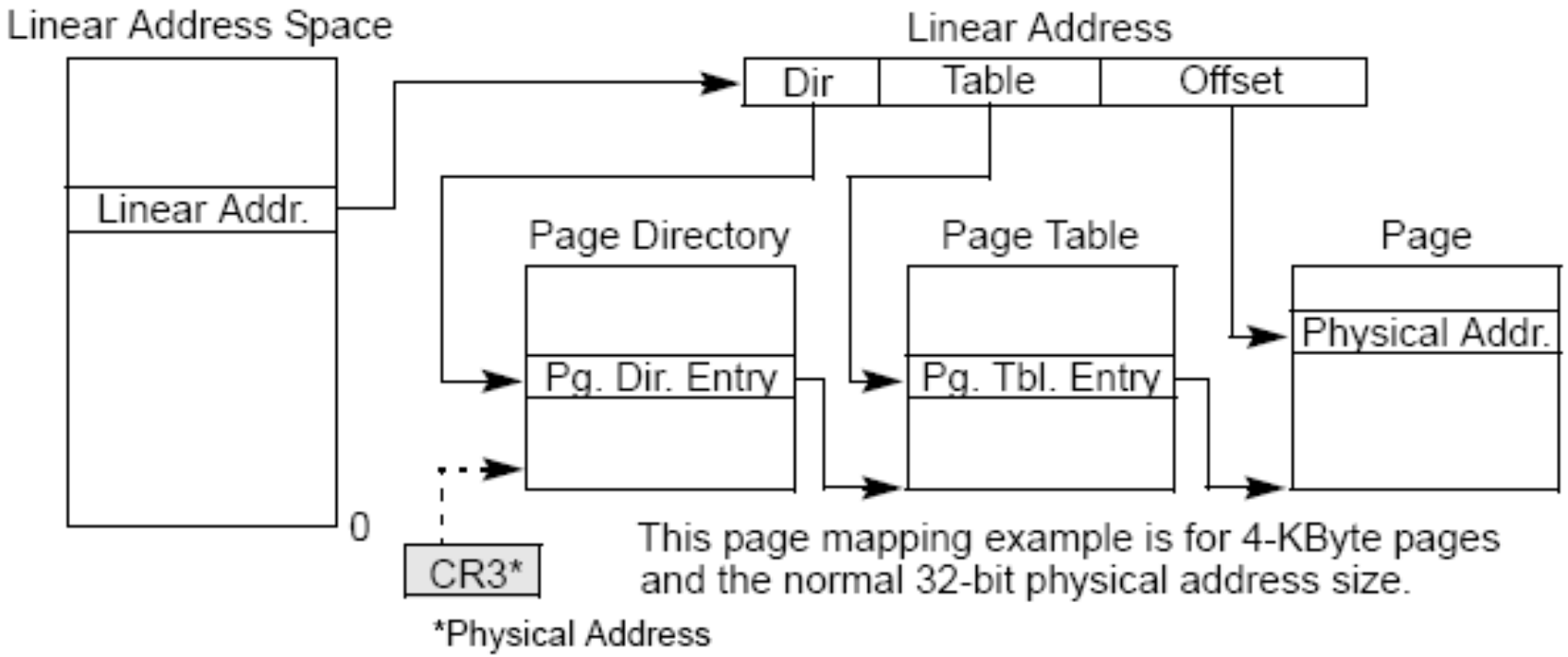
Virtual memory support (Providing and Protecting Memory)

- Segmentation and paging
- Basic flat memory model
  - Contiguous unsegmented address space
  - Set up (at least) one code and one data segment that both map entire address space
- Protected flat memory model
  - As above except segment limits set to match physical memory present
- Multi segment model
  - Each process is given its own table of segment descriptors and its own segments
- Works intimately with privilege level and access control
- *Let's skip the grimy details for now*

Virtual memory support ii

- Segment registers contain segment selectors (14 bit)
- Segment selectors index into one of two (GDT/LDT) descriptor tables, containing segment descriptors
- Segment descriptors contain page directory base address and limit, ++

System Level Registers & Data Structures (ii)



Cache support

- Cache policy set through CD and NW bits in CR3 (bits 4 and 3)
  - Write back
  - Write through
  - Cache disabled
- INVD, Invalidate internal caches
- WBINVD, Write back, and invalidate internal caches

I/O support

- See foils on I/O
- Special instructions
- Memory mapped I/O
- Interrupts
- Polling

Virtual Machine Extensions

- Support for running several virtual machines concurrently on single physical processor
- Virtual Machine Monitors (VMM)
- Guest software
- Important resurface of technology similar to IBM/VM
- *However, not a core issue on this course now*

Multiprocessor support

- Multiple processors
- Single “hyperthreaded” processor
- Single “multi-core” processor chip
- Combinations of above
- Some issues
  - Distribution of tasks among processors
  - Distribution of interrupts among processors
  - Ensuring memory consistency and cache coherence

Literature

- Randal E. Bryant and David O'Hallaron, Computer Systems: A Programmer's Perspective, Prentice-Hall, 2003. Chapter Eight
  - » http://csapp.cs.cmu.edu/
- Jerome H. Saltzer; M. Frans Kaashoek, Principles of Computer System Design – An Introduction
  - » http://ocw.mit.edu/resources/res-6-004-principles-of-computer-system-design-an-introduction-spring-2009/
- Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture
  - » http://www.intel.com/design/processor/manuals/253665.pdf
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1
  - » http://www.intel.com/design/processor/manuals/253668.pdf
- Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, Part 2
  - » http://www.intel.com/design/processor/manuals/253669.pdf