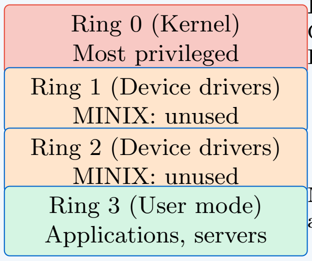
Intel 80386 Baseline Architecture (1985)

Protected Mode, Paging, and Segmentation - Foundation for MINIX

Protection

Protected Mode (Ring 0-3)

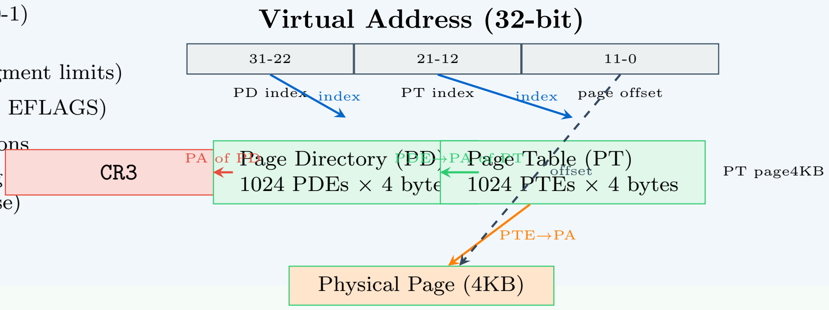


Privilege Levels (CPL):
CPL in CS register (bits 0-1)
Protection checks on:

- Memory access (segment limits)
- I/O ports (IOPL in EFLAGS)
- Privileged instructions

MINIX uses 2 levels: Ring
and Ring 3 (everything else)

Two-Level Paging (4KB pages)



Memory Management

Control Registers

CR0

CR0.PE (bit 0): Protected mode Enable
CR0.WP (bit 16): Write Protect (486+)
CR0.PG (bit 31): PaGing enable
Writing PG=1: Enables paging (requires PE=1)

CR2

Page Fault Linear Address
On #PF exception, CR2 holds faulting virtual address
Kernel reads CR2 to determine which page faulted

CR3

Page Directory Base Register (PDBR)
Bits 31-12: Physical address of PD page
Bits 11-0: Reserved (must be 0)
Writing CR3: Flushes TLB (all non-global entries)

PTE/PDE Structure (32-bit)

Page Table Entry / Page Directory Entry

31-12	11-9	8	7	6	5	4	3	2	1	0
Physical page frame	OS	G	PS	D	A	N	WT	U	RW	P

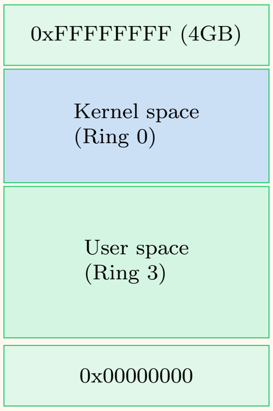
Bit 0 (P): Present (1=valid mapping, 0=page fault)
Bit 1 (RW): Read/Write (0=read-only, 1=read-write)
Bit 2 (U): User/Supervisor (0=Ring 0 only, 1=Ring 3 accessible)
Bit 3 (WT): Write-Through (cache policy)
Bit 4 (N/PCD): No cache / Cache Disable
Bit 5 (A): Accessed (set by CPU on read/write)
Bit 6 (D): Dirty (set by CPU on write, PTEs only)
Bit 7 (PS): Page Size (PDEs: 0=4KB PT, 1=4MB page; not on 386)
Bit 8 (G): Global (don't flush on CR3 write; Pentium Pro+)
Bits 9-11: Available for OS use (MINI
Bits 31-12: Physical page frame numb

Segmentation & Instructions

Segmentation

Segment Registers: CS, DS, SS, ES, FS, GS
Each contains 16-bit **selector**:
Bits 15-3: Index into GDT/LDT
Bit 2: Table (0=GDT, 1=LDT)
Bits 1-0: RPL (Requested Privilege Level)
Segment Descriptor (8 bytes in GDT/IDT):
Base: 32-bit linear address
Limit: 20-bit size (granularity: bytes or 4KB)
Type: Code/Data/System
DPL: Descriptor Privilege Level (0-3)
Present bit
Logical → Linear translation:
1. Load selector into segment register
2. CPU loads descriptor from GDT/LDT (cached)
3. Linear addr = Segment base + Offset
4. If paging enabled: Linear → Physical via page tables

386 Memory Model

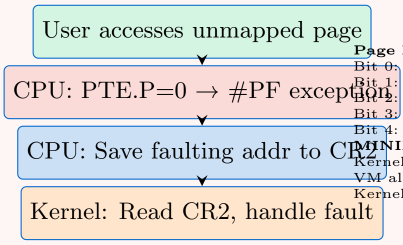


Typi

Privileged (Ring 0 only):
`LGDT/LIDT` - Load GDT/IDT register
`LTR` - Load Task Register
`LLDT` - Load LDT
`CLI/STI` - Clear/Set Interrupt Flag
`HLT` - Halt until interrupt
`INVLPG` - Invalidate TLB entry (single page)
`MOV CR0/CR2/CR3` - Control register access
Protection checks:
`LAR` - Load Access Rights
`LSL` - Load Segment Limit
`VERR/VERW` - Verify Read/Write
`ARPL` - Adjust RPL
System calls:
`INT n` - Software interrupt (200 cycles)
`IRET` - Return from interrupt

Exception Handling

Page Fault Flow (#PF Exception)



User accesses unmapped page
CPU: PTE.P=0 → #PF exception
CPU: Save faulting addr to CR2
Kernel: Read CR2, handle fault

Page Fault Error Code (pushed on stack):
Bit 0: P (0=not present, 1=protection violation)
Bit 1: W/R (0=read, 1=write)
Bit 2: U/S (0=kernel, 1=user mode)
Bit 3: RSVD (reserved bit violation)
Bit 4: I/D (instruction fetch)
MINIX handling:
Kernel sends IPC to VM server with CR2 address
VM allocates page, maps it, returns
Kernel restores user context, retries instruction

Purpose: Run real-mode 8086 code in protected environment
Activation: Set EFLAGS.VM (bit 17)
Features:

- Maps 1MB address space (0x00000-0xFFFFF)
- Segments behave like real mode (base = seg << 4)
- I/O and interrupts can be virtualized
- Page-level protection still enforced

MINIX usage: Not used (no DOS/BIOS compatibility needed)

386 Key Innovations (1985)

Protected Mode: 4 privilege rings (Ring 0-3) with hardware-enforced protection checks
Paging: Two-level page tables, 4KB pages, demand paging, copy-on-write support
32-bit Architecture: 32-bit registers (EAX, EBX, etc.), 4GB virtual address space
Virtual 8086 Mode: Run real-mode code in protected environment (for DOS compatibility)
Foundation for MINIX: All modern OS features (process isolation, virtual memory, privilege separation)

`pte.h:78-170` MMU structure, PTE bits
`pte.h:238-266` PG_V, PG_RW, PG_U flags
`specialreg.h:35-50` CR0, CR2, CR3 bits

`pmap.h:100-107` Page directory structure
`pmap.h:145-171` pmap structure
`protect.c:98-107` Segment setup