x86

Gabriel Laskar <gabriel@lse.epita.fr>

http://lse.epita.fr/teaching/epita/x86a.html

LSE Security System

Outline

- x86 assembly
- 64bit support
- pagination
- multi-core
- virtualization
- project

LSE Security System

x86_64 : what's new ?

- more registers
- 64bit addresses, 64bit registers
- no more segmentation (but gdt still present)
- new features in pagination
- no Task Switch but TSS still present
- lots of thing removed, but still present (for special cases)

LSE Security System

Multiple kind of x86 registers

- General purpose registers
- Segment registers
- FLAGS
- Control & Memory registers

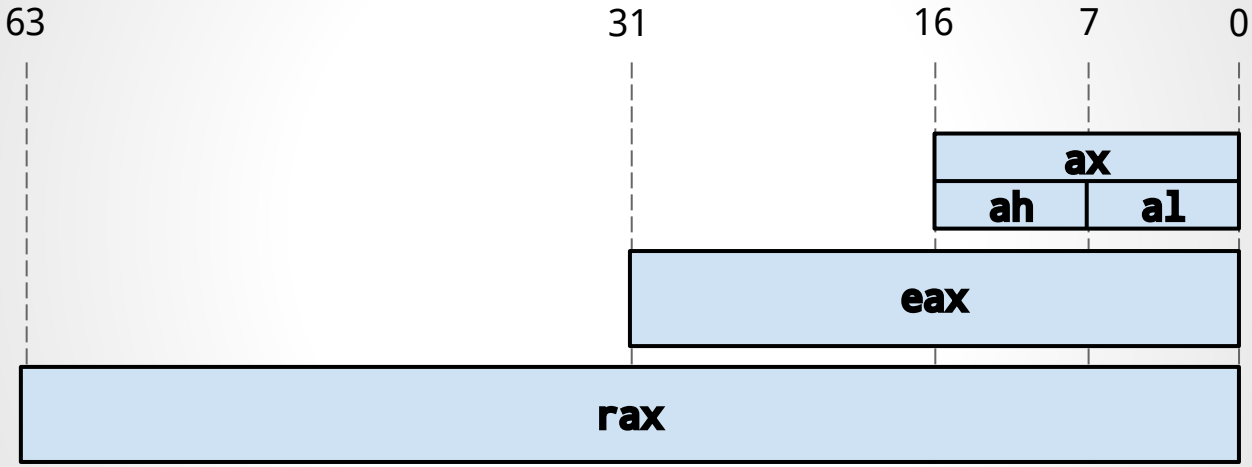
LSE Security System

General purpose registers

- `%rax`, `%rbx`, `%rcx`, `%rdx`
- `%rsi`, `%rdi`
- `%rsp`, `%rbp`
- `%rip`
- `%r8` → `%r15`

LSE Security System

Register Aliases



LSE Security System

Instruction pointer : %rip

- in x86_64, instructions can now reference data relative to %rip

```
.global main
main:
    lea string(%rip), %rdi
    call puts
    ret

.section .rodata
string:
    .ascii "hello world!"
```

LSE Security System

String manipulation

- rep prefix allow to repeat an instruction
- string instructions : movs, scas, stos

```
.global strlen
strlen:
    xor %rcx, %rcx
    not %rcx
    xor %al, %al
    cld
    repne scasb
    not %rcx
    dec %rcx
    mov %rcx, %rax
    ret
```

LSE Security System

Flags register

- flags, eflags, rflags
- pushf, popf
- contains information about execution of the last instruction

LSE Security System

Flags

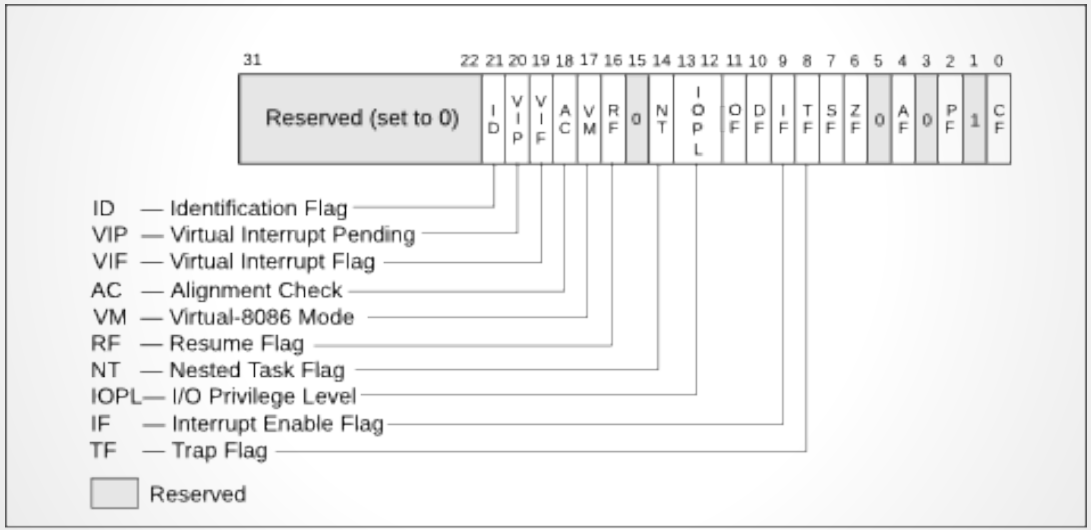


Figure 2-5. System Flags in the EFLAGS Register

Rings

- 4 rings in x86_32, only 2 rings in x86_64
- SMM mode
- other modes (virtualization)

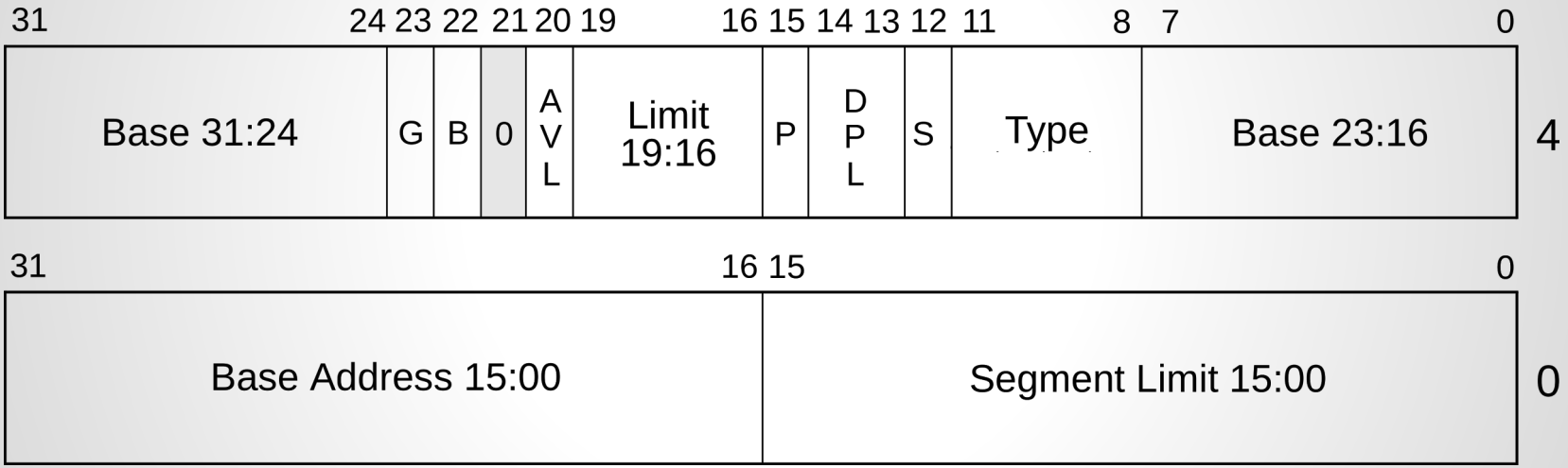
LSE Security System

Memory Registers

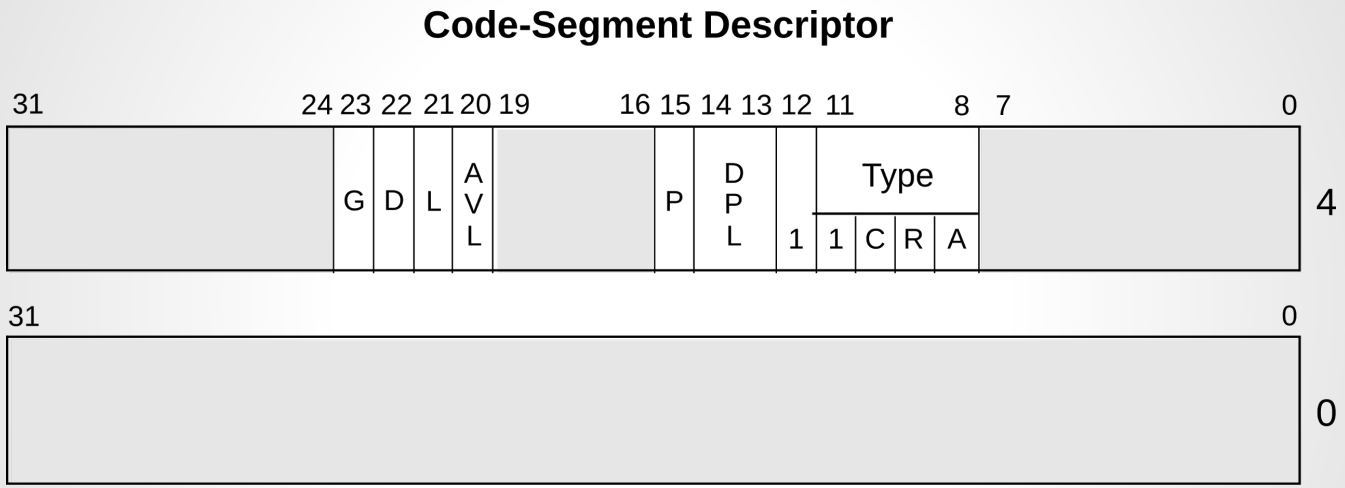
- gdtr
- idtr
- ltr

LSE Security System

GDT entries


31
24
23
22
21
20
19
16
15
14
13
12
11
8
7
0
Base 31:24
G
B
0
A V L
Limit 19:16
P
D P L
S
Type
Base 23:16
4
31
16
15
0
Base Address 15:00
Segment Limit 15:00
0

GDT entries in x86_64

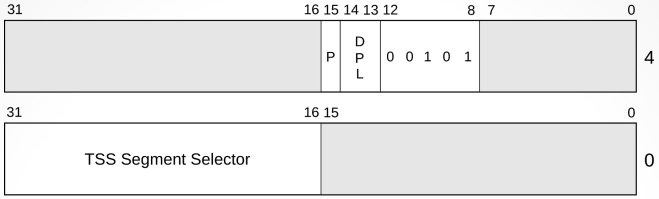


A	Accessed
AVL	Available to Sys. Programmer's
C	Conforming
D	Default
DPL	Descriptor Privilege Level
L	64-Bit Flag
G	Granularity
R	Readable
P	Present

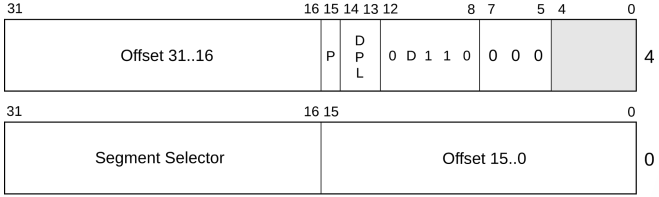
LSE Security System

IDT entries

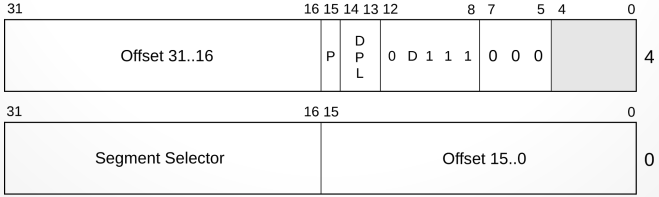
Task Gate



Interrupt Gate



Trap Gate



DPL	Descriptor Privilege Level
Offset	Offset to procedure entry point
P	Segment Present flag
Selector	Segment Selector for destination code segment
D	Size of gate: 1 = 32 bits; 0 = 16 bits

Reserved

Segment selectors

- Tied to gdt entries
- 2 parts, public part and shadowed part
- provide basic permissions on zones
- each segment selector describe memory access for some instructions

LSE Security System

Descriptions of segment selectors

- cs : access to code (%rip, call, ret ...)
- ss : access to stack data (%rsp, push, pop)
- ds : access to memory and %rdi
- es : access to %rsi
- fs : user-defined
- gs : user-defined

LSE Security System

Thread local storage

- `%fs`, `%gs` can be used to implement TLS variables.
- One page mapped, and referenced by segment selector

LSE Security System

Control registers

- cr0 : system control flags
- cr2 : page fault linear address
- cr3 : address space address
- cr4 : architecture extensions
- cr8 : Task Priority Register

LSE Security System

Control Registers

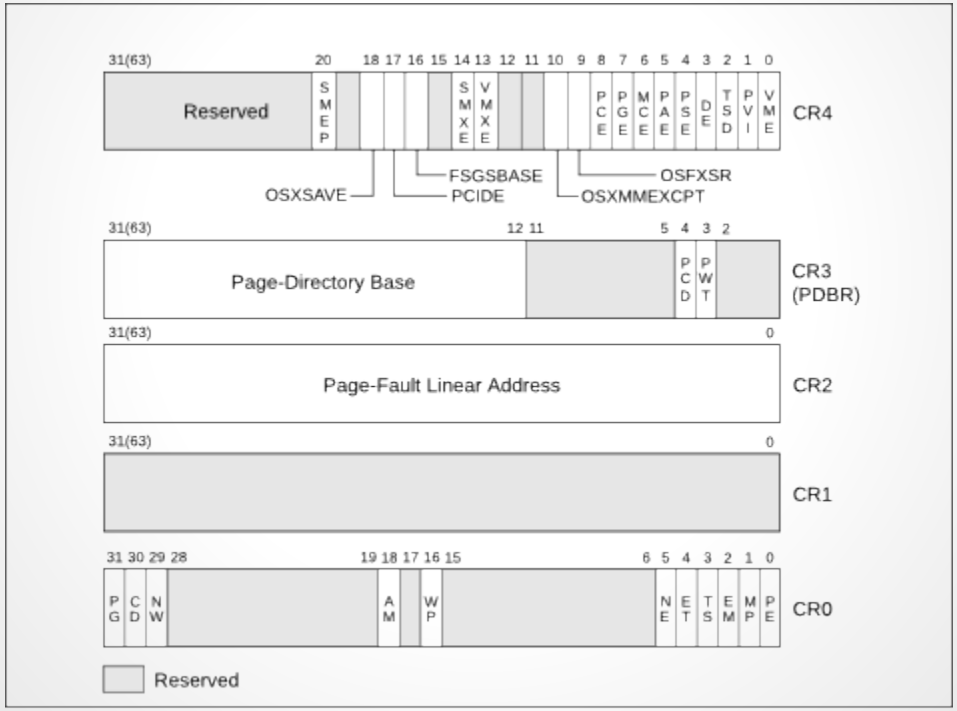


Figure 2-7. Control Registers

LSE Security System

%cr0

- Paging (bit 31)
- Cache Disable (bit 30)
- Not Write-through (bit)
- Alignment Mask (bit 18)
- Write Protect (bit 16)
- Numeric Error (bit 5)
- Extension Type (bit 4)
- Task Switched (bit 3)
- Emulation (bit 2)
- Monitor Coprocessor (bit 1)
- Protection Enable (bit 0)

LSE Security System

%cr4

- Virtual-8086 Mode Extensions (bit 0 of CR4)
- Protected-Mode Virtual Interrupts (bit 1 of CR4)
- Time Stamp Disable (bit 2 of CR4)
- Debugging Extensions (bit 3 of CR4)
- Page Size Extensions (bit 4 of CR4)
- Physical Address Extension (bit 5 of CR4)
- Machine-Check Enable (bit 6 of CR4)
- Page Global Enable (bit 7 of CR4)
- Performance-Monitoring Counter Enable (bit 8 of CR4)
- Operating System Support for FXSAVE and FXRSTOR instructions (bit 9 of CR4)
- Operating System Support for Unmasked SIMD Floating-Point Exceptions (bit 10 of CR4)
- VMX-Enable Bit (bit 13 of CR4)
- SMX-Enable Bit (bit 14 of CR4)
- FSGSBASE-Enable Bit (bit 16 of CR4)
- PCID-Enable Bit (bit 17 of CR4)
- XSAVE and Processor Extended States-Enable Bit (bit 18 of CR4)
- SMEP-Enable Bit (bit 20 of CR4)

LSE Security System

Debug registers

- support for debugging
- exceptions
- eflags register
- debug registers (`%dr0`-`%dr3`, `%dr6`, `%dr7`)

LSE Security System

Machine Specific registers

- overview
- rdmsr
- wrmsr

LSE Security System

Calling Conventions

- Lots of different ways to call a function
- here we focus on linux

http://stackoverflow.com/questions/2535989/what-are-the-calling-conventions-for-unix-linux-system-calls-on-x86-64

LSE Security System

x86_32 : calling functions

- on x86_32 :
  - arguments on the stack, in reverse order
  - return value in %eax
  - %eax, %ecx, %edx saved by caller
  - stack must be 16-byte aligned

LSE Security System

x86_32 : syscalls

- %ecx, %edx, %edi and %ebp
- instruction `int $0x80`
- The number of the syscall has to be passed in register %eax
- %eax contains the result of the system-call

LSE Security System

x86_64 : calling functions

- If the class is MEMORY, pass the argument on the stack.
- If the class is INTEGER, the next available register of the sequence %rdi, %rsi, %rdx, %rcx, %r8 and %r9 is used

LSE Security System

x86_64 : syscalls

- %rdi, %rsi, %rdx, %r10, %r8 and %r9
- The kernel destroys registers %rcx and % r11.
- instruction syscall
- The number of the syscall has to be passed in register %rax
- %rax contains the result of the system-call

LSE Security System

Pagination

- multiple modes (32bit, 32bit pae, 64bit)
- table format
- TLB
- mirroring
- permissions
- initialization
- COW, swaping, shared memory

LSE Security System

Pagination

Table 4-1. Properties of Different Paging Modes

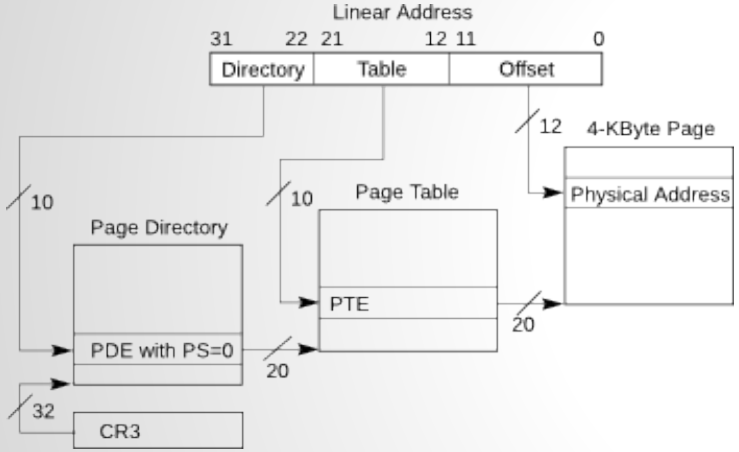
Paging Mode	PG in CR0	PAE in CR4	LME in IA32_EFER	Lin.-Addr. Width	Phys.-Addr. Width[1]	Page Sizes	Supports Execute-Disable?	Supports PCIDs?
None	0	N/A	N/A	32	32	N/A	No	No
32-bit	1	0	0[2]	32	Up to 40[3]	4 KB 4 MB[4]	No	No
PAE	1	1	0	32	Up to 52	4 KB 2 MB	Yes[5]	No
IA-32e	1	1	1	48	Up to 52	4 KB 2 MB 1 GB[6]	Yes[5]	Yes[7]

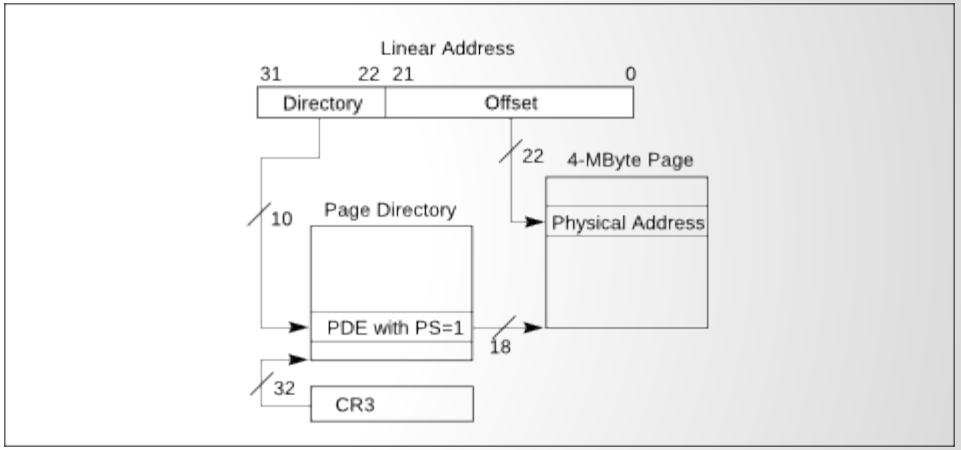
NOTES:

1. The physical-address width is always bounded by MAXPHYADDR; see Section 4.1.4.
2. The processor ensures that IA32_EFER.LME must be 0 if CR0.PG = 1 and CR4.PAE = 0.
3. 32-bit paging supports physical-address widths of more than 32 bits only for 4-MByte pages and only if the PSE-36 mechanism is supported; see Section 4.1.4 and Section 4.3.
4. 4-MByte pages are used with 32-bit paging only if CR4.PSE = 1; see Section 4.3.
5. Execute-disable access rights are applied only if IA32_EFER.NXE = 1; see Section 4.6.
6. Not all processors that support IA-32e paging support 1-GByte pages; see Section 4.1.4.
7. PCIDs are used only if CR4.PCIDE = 1; see Section 4.10.1.

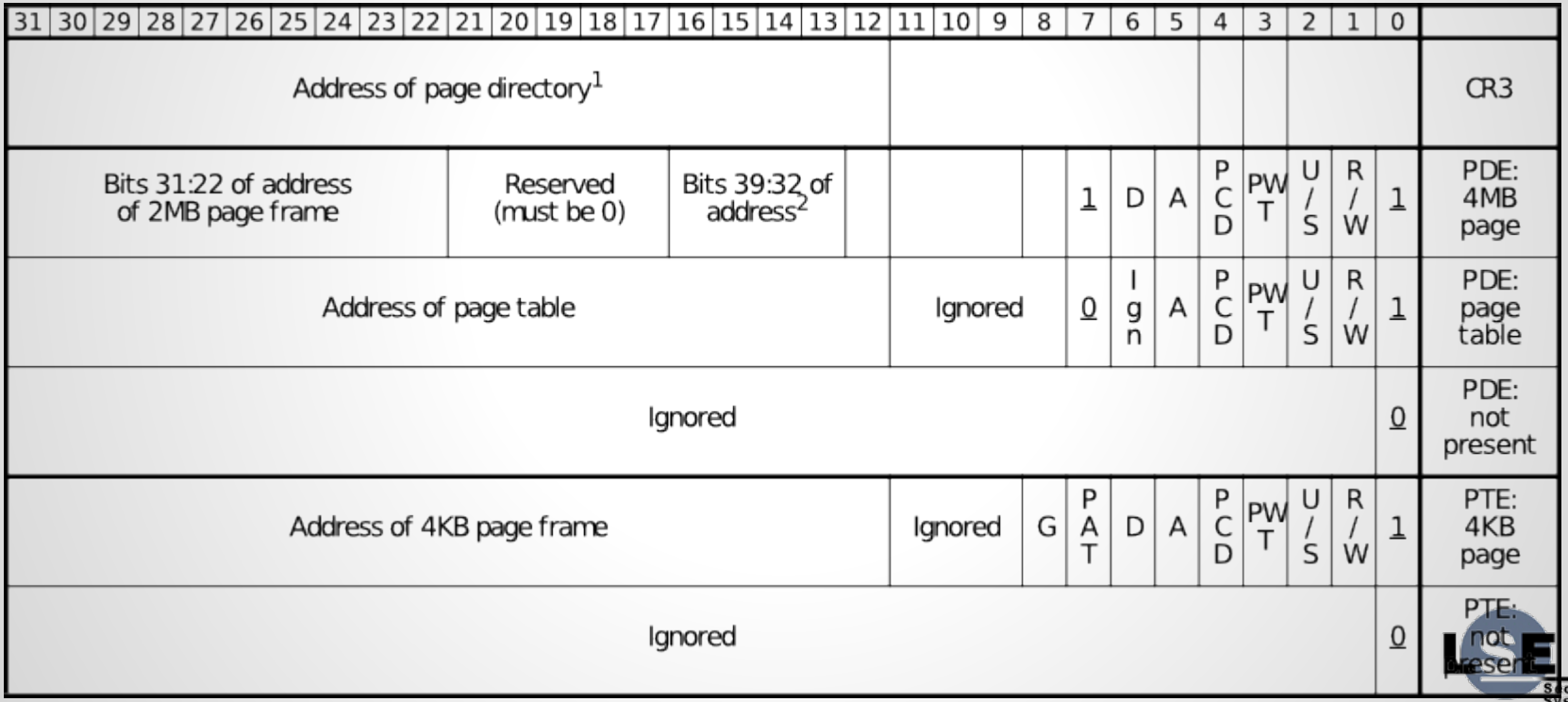
LSE Security System

x86 pagination



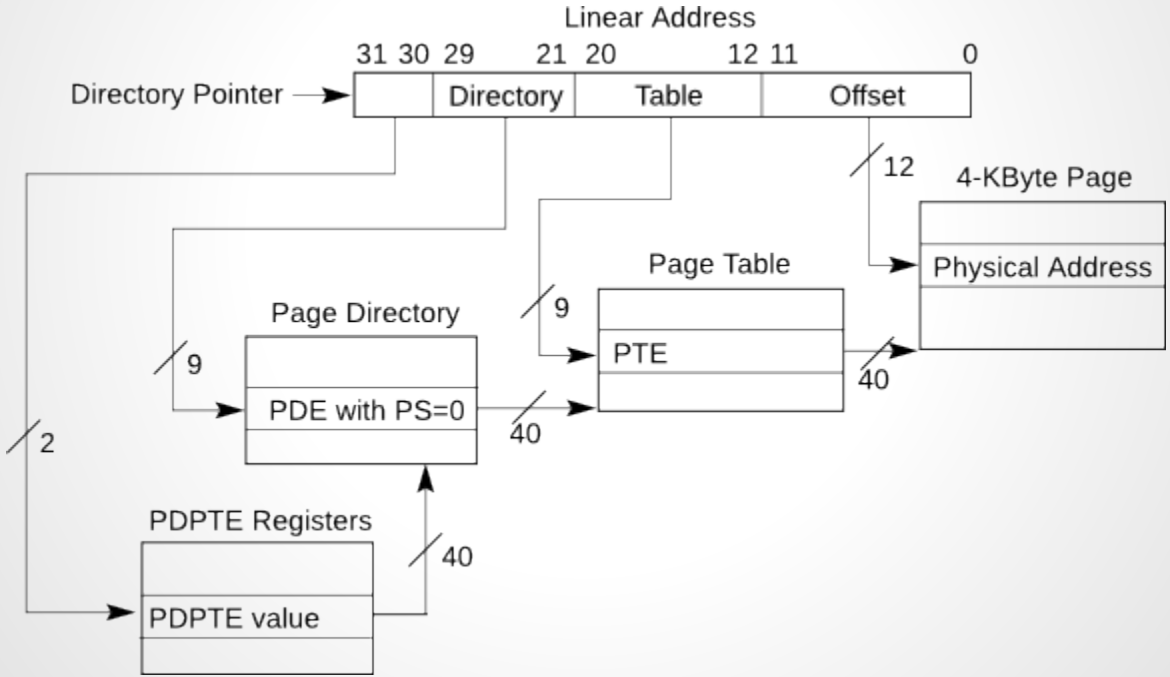


x86 structures

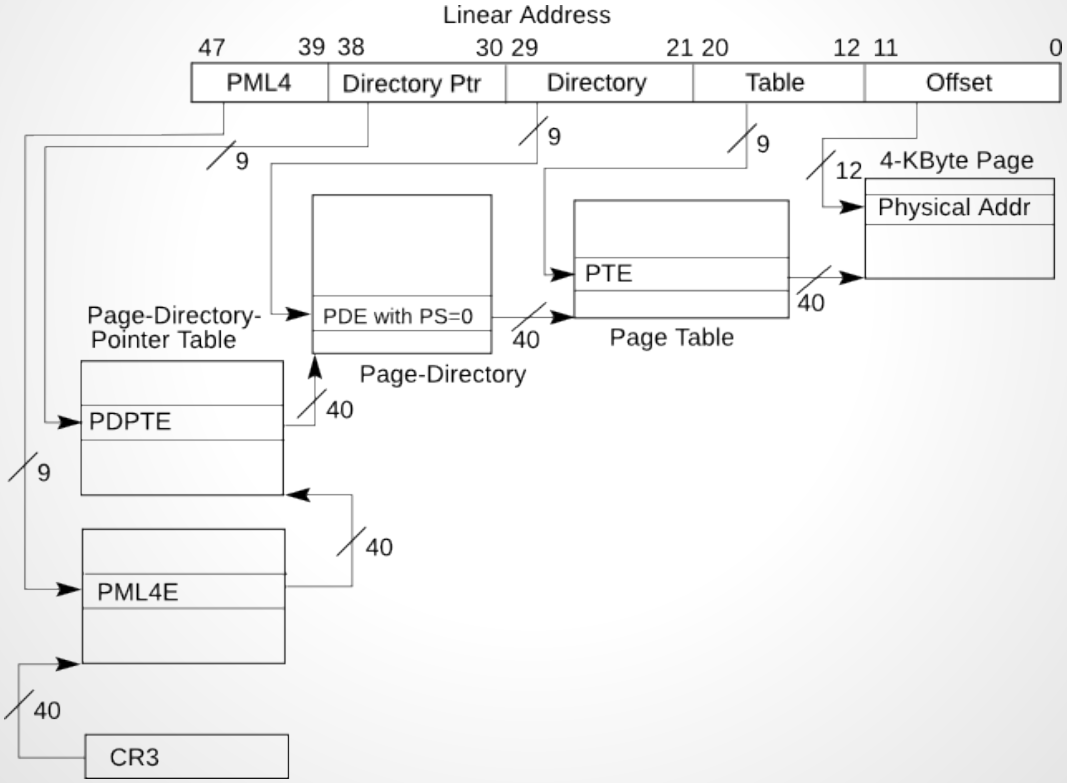


31–12	11–9	8	7	6	5	4	3	2	1	0	
Address of page directory[1]											CR3
Bits 31:22 of address of 2MB page frame; Reserved (must be 0); Bits 39:32 of address[2]			1	D	A	PCD	PWT	U/S	R/W	1	PDE: 4MB page
Address of page table	Ignored		0	Ign	A	PCD	PWT	U/S	R/W	1	PDE: page table
Ignored										0	PDE: not present
Address of 4KB page frame	Ignored	G	PAT	D	A	PCD	PWT	U/S	R/W	1	PTE: 4KB page
Ignored										0	PTE: not present

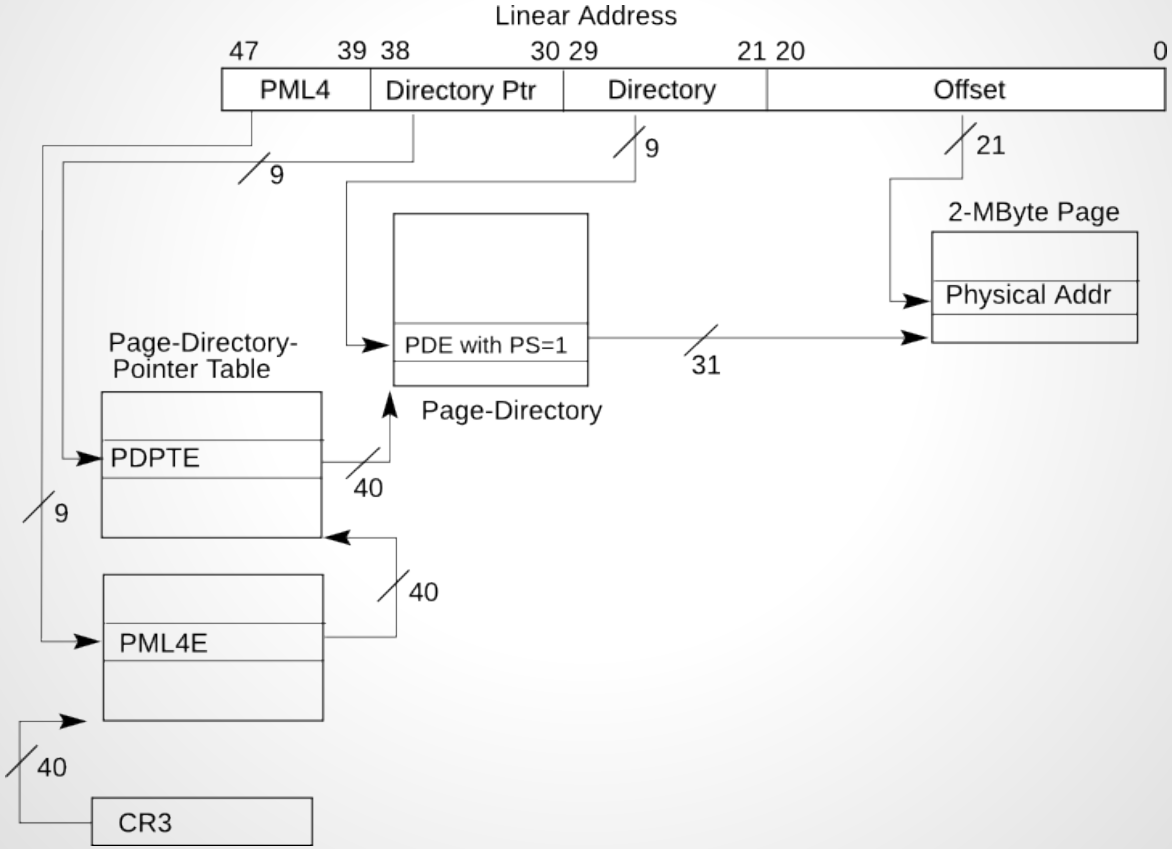
PAE


Linear Address
31 30 29 21 20 12 11 0
Directory Pointer
Directory
Table
Offset
12
4-KByte Page
Physical Address
Page Table
9
PTE
40
Page Directory
9
PDE with PS=0
40
2
PDPTE Registers
40
PDPTE value

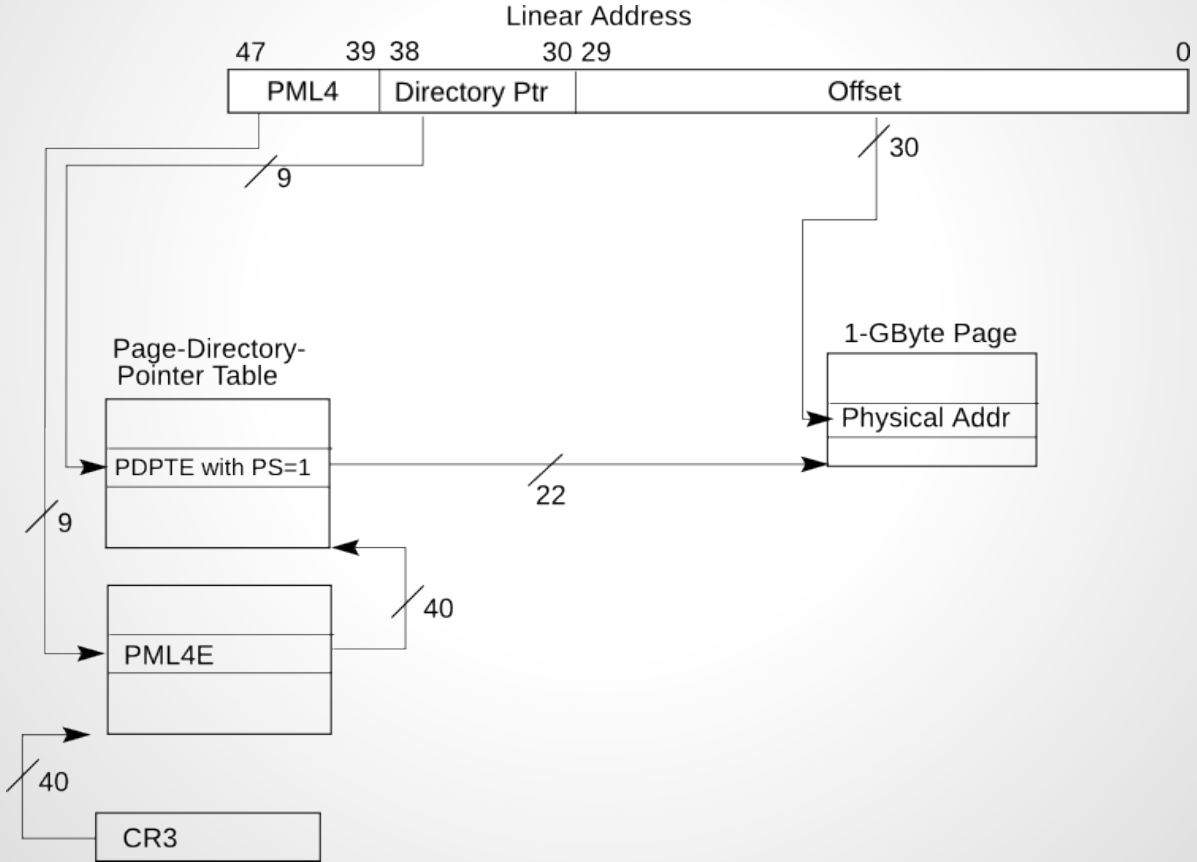
64bit pagination



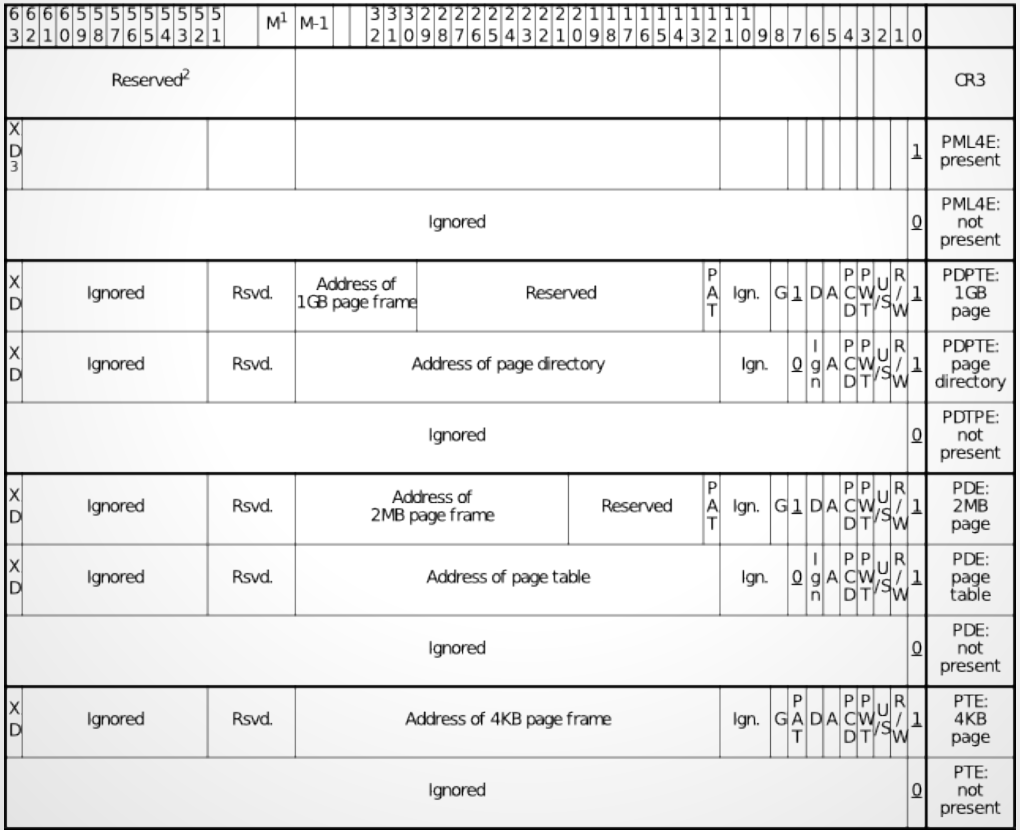
x86_64 : 2Mb Pages



x86_64 : 1Gb pages



x86_64 : structures



Page fault Handling

P	0	The fault was caused by a non-present page.
	1	The fault was caused by a page-level protection violation.
W/R	0	The access causing the fault was a read.
	1	The access causing the fault was a write.
U/S	0	A supervisor-mode access caused the fault.
	1	A user-mode access caused the fault.
RSVD	0	The fault was not caused by reserved bit violation.
	1	The fault was caused by a reserved bit set to 1 in some paging-structure entry.
I/D	0	The fault was not caused by an instruction fetch.
	1	The fault was caused by an instruction fetch.

LSE Security System

TLB

- Cache for address translations
- 2 TLB : one for data, one for instructions

LSE Security System

PAX

On x86_32, How can we enforce NX bit without the hardware support ?

LSE Security System