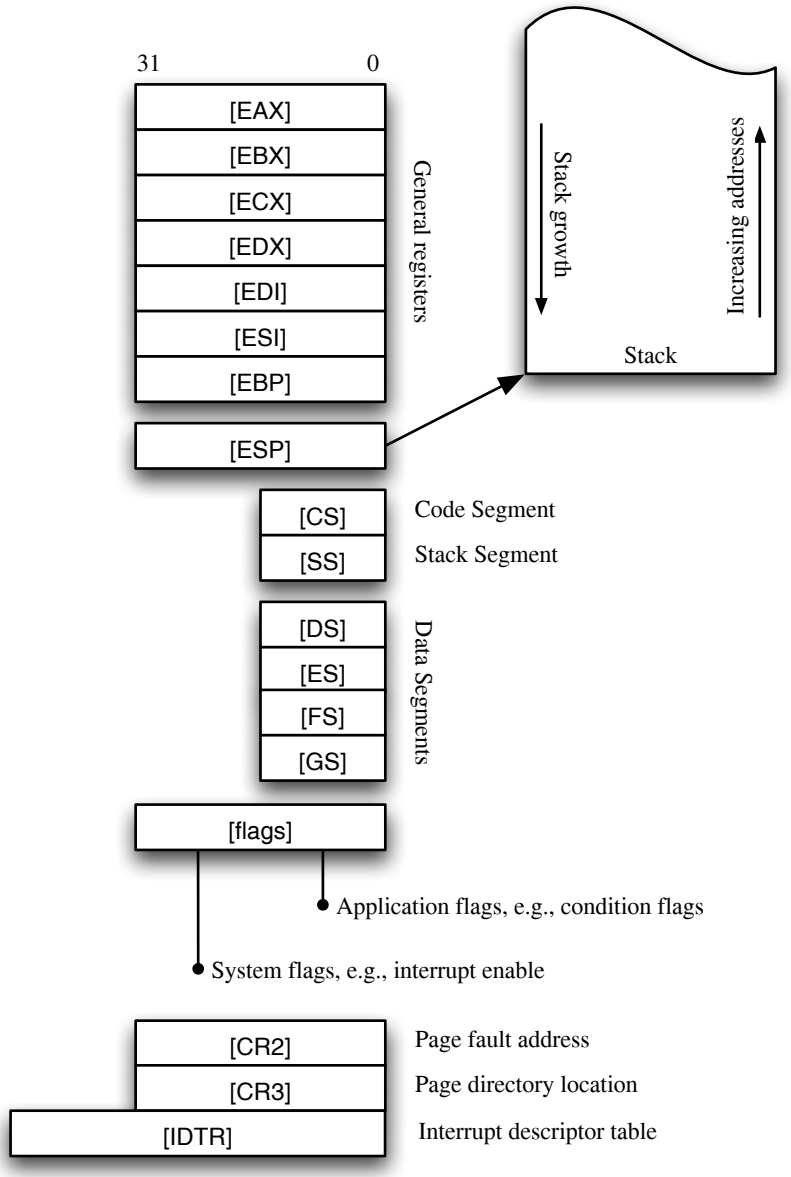
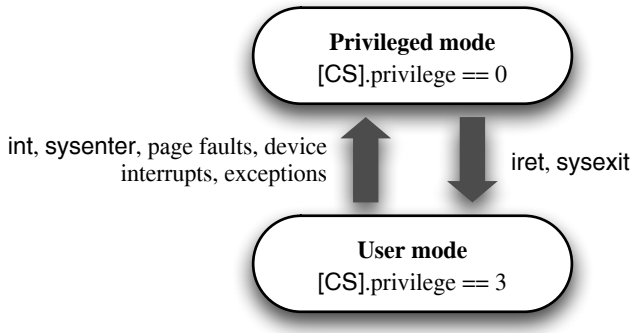
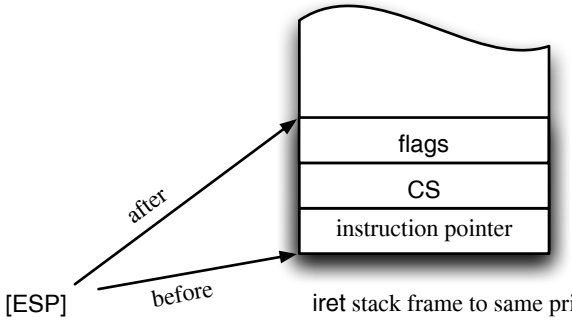
IA32 Condensed for Kernel Developers



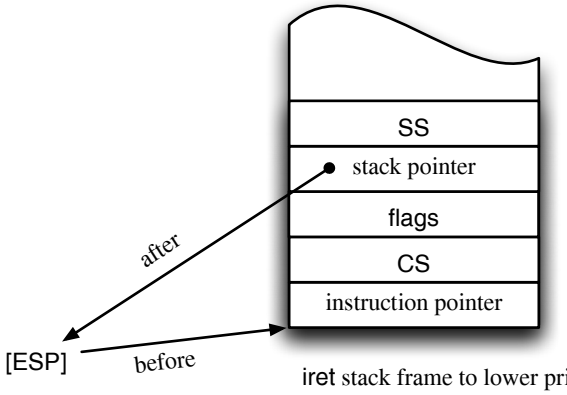
Operation	AT&T syntax	Intel syntax
[EBX] := [EAX]	movl %eax, %ebx	mov ebx, eax
[EAX] := 4	movl $4, %eax	mov eax, 4
*([EBX]) := [EAX]	movl %eax, 0(%ebx)	mov [ebx], eax
*([EBX]) := [EAX][7:0]	movb %al, 0(%ebx)	mov byte ptr [ebx], al
*([EBX] + 4) := [EAX]	movl %eax, 4(%ebx)	mov [ebx+4], eax
[EAX] := [EAX] + [EBX]	addl %ebx, %eax	add eax, ebx
Install [EAX] as the page directory	movl %eax, %cr3	mov cr3, eax
Push [EAX] on the stack	pushl %eax	push eax
Pop the stack into [ECX]	popl %ecx	pop ecx
Raise software interrupt 80h	int $0x80	int 0x80
Return from interrupt	iret	iret
Enable [flags].interrupts	sti	sti
Disable [flags].interrupts	cli	cli
Push [flags] on the stack	pushf	pushf
Pop the stack into [flags]	popf	popf
Call with link to function printf()	call printf	call printf
Call to absolute address	call *%eax	call eax
Return from function (address on stack)	ret	ret
Fast system call	sysenter	sysenter
Fast return from system call	sysexit	sysexit

Kernel entry and exit:

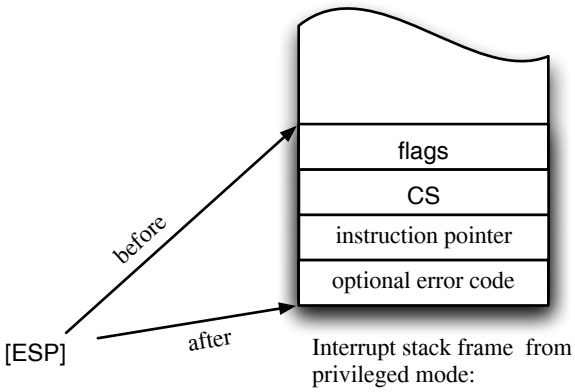




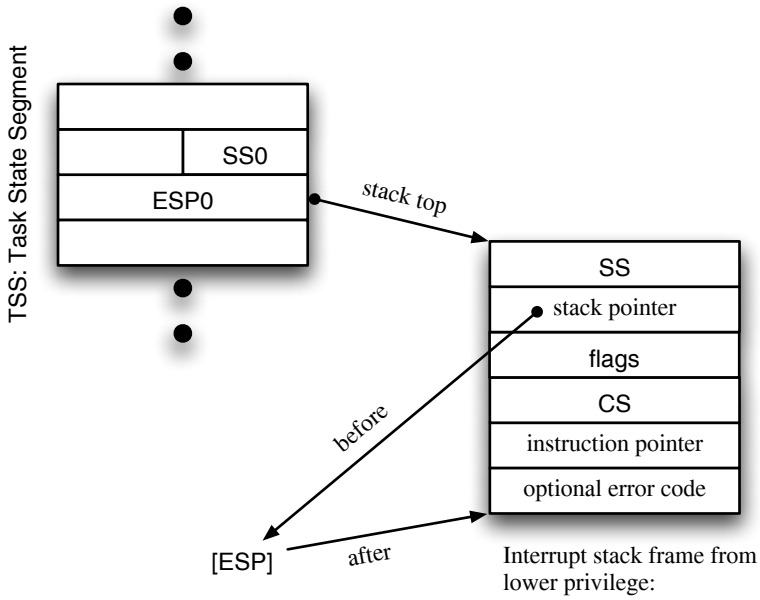
iret stack frame to same privilege:
CS.privilege == [CS].privilege



iret stack frame to lower privilege:
CS.privilege > [CS].privilege



Interrupt stack frame from privileged mode:
CS.privilege == [CS].privilege



Interrupt stack frame from lower privilege:
CS.privilege > [CS].privilege

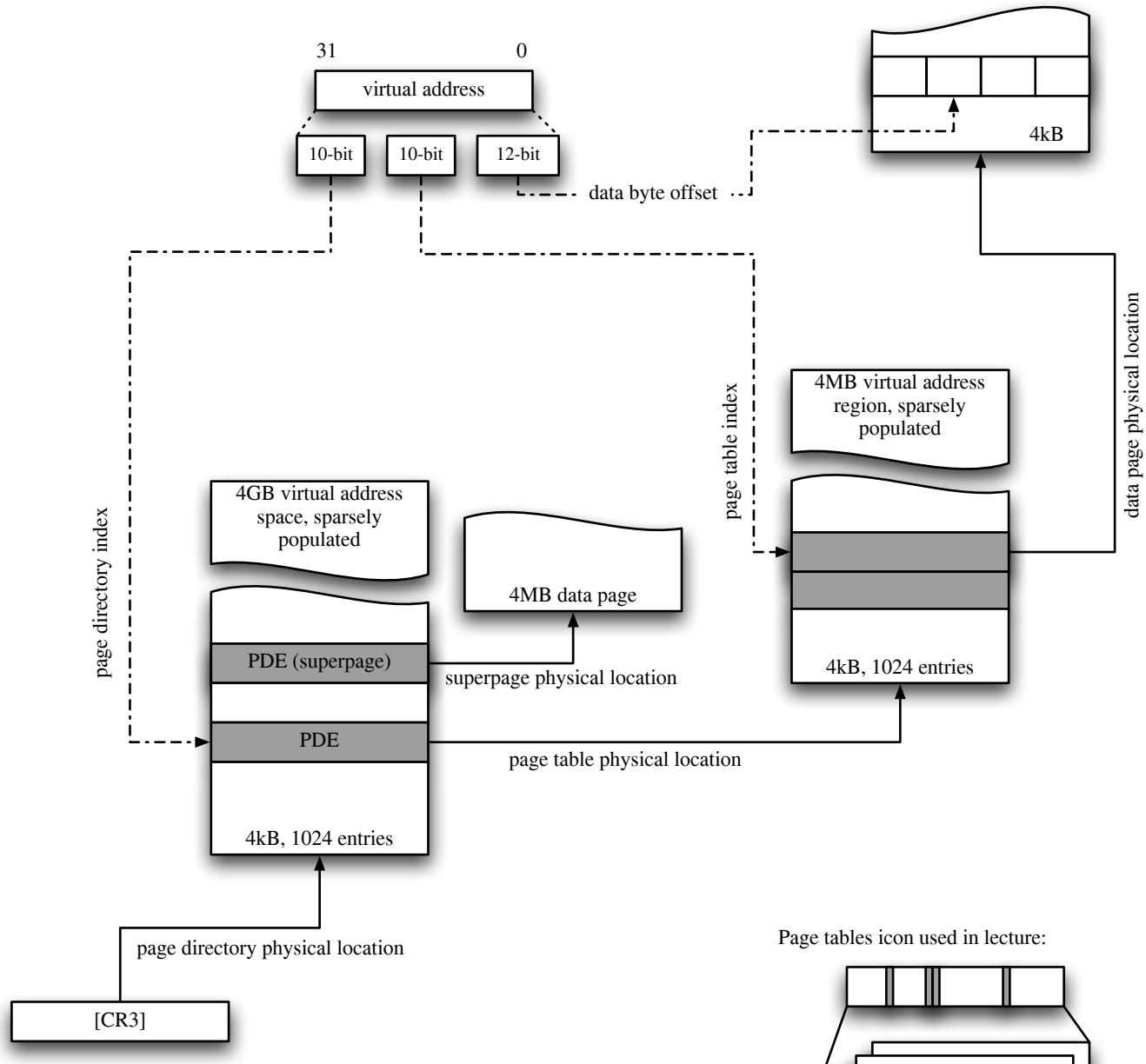
sysenter configuration:

[MSR sysenter CS]	CS segment selector
[MSR sysenter EIP]	Instruction pointer
[MSR sysenter ESP]	Stack pointer

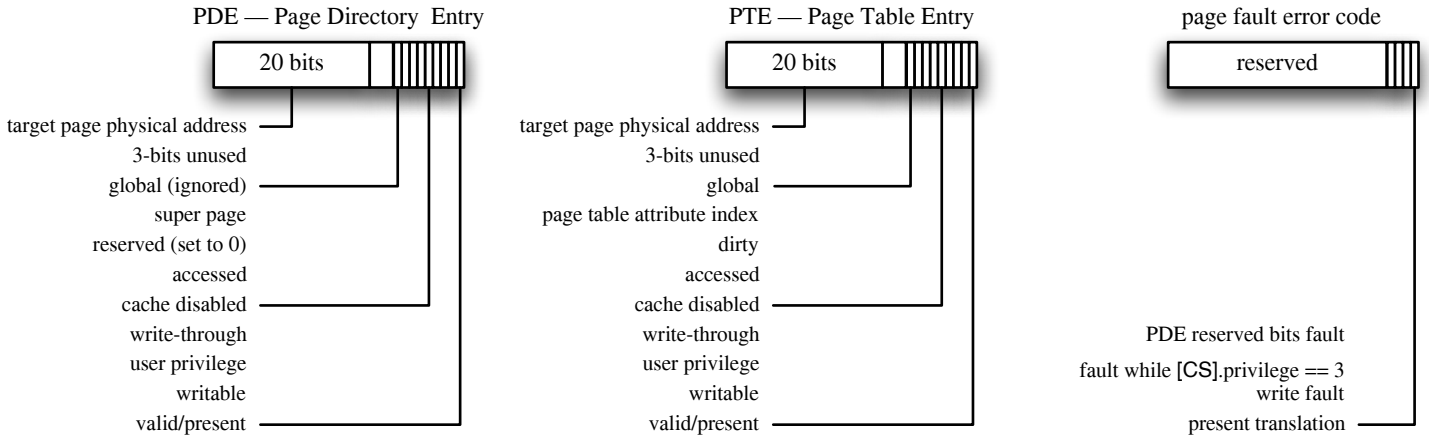
sysexit configuration:

[MSR sysenter CS]	CS segment selector base
[EDX]	Instruction pointer
[ECX]	Stack pointer

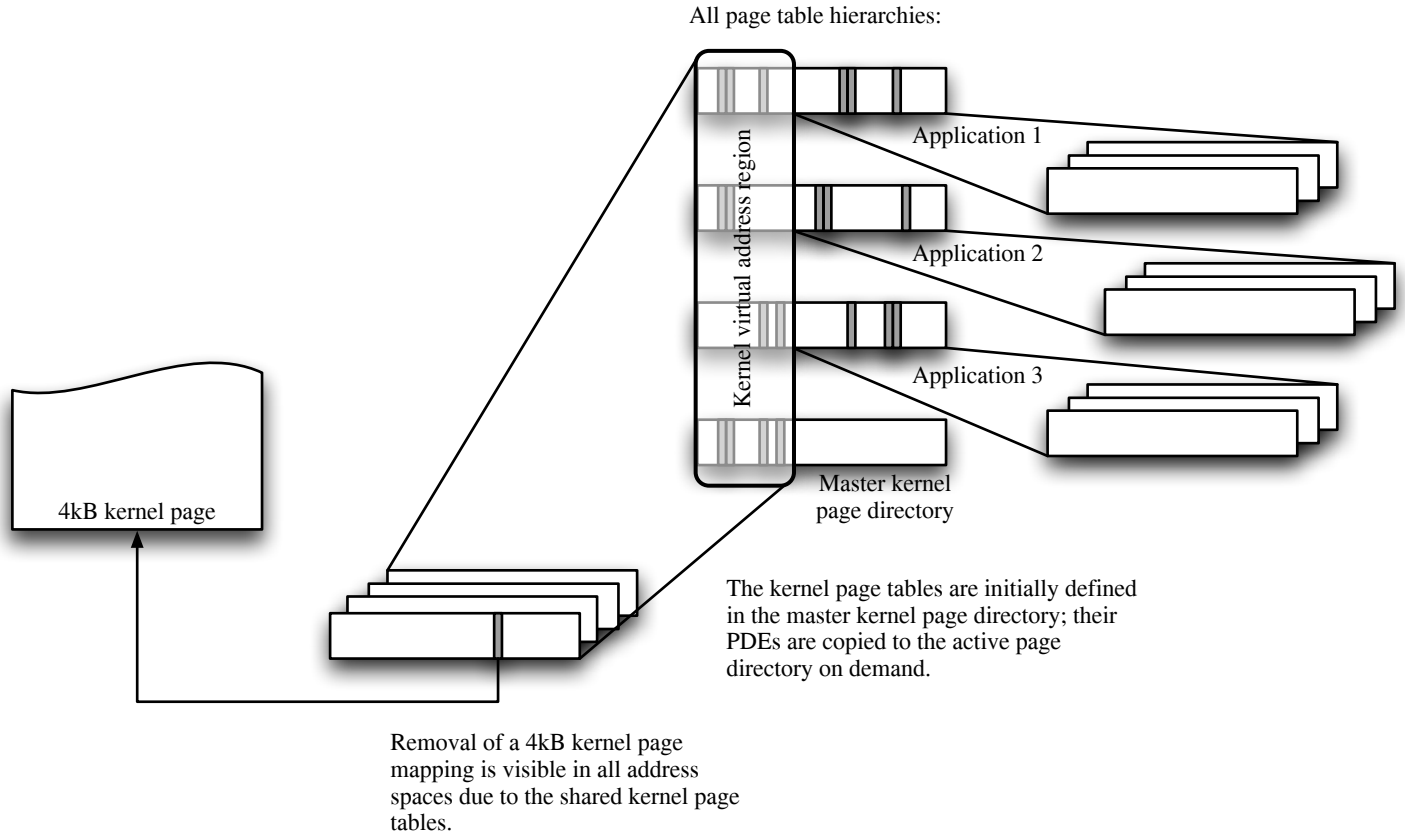
Virtual to physical address translation:



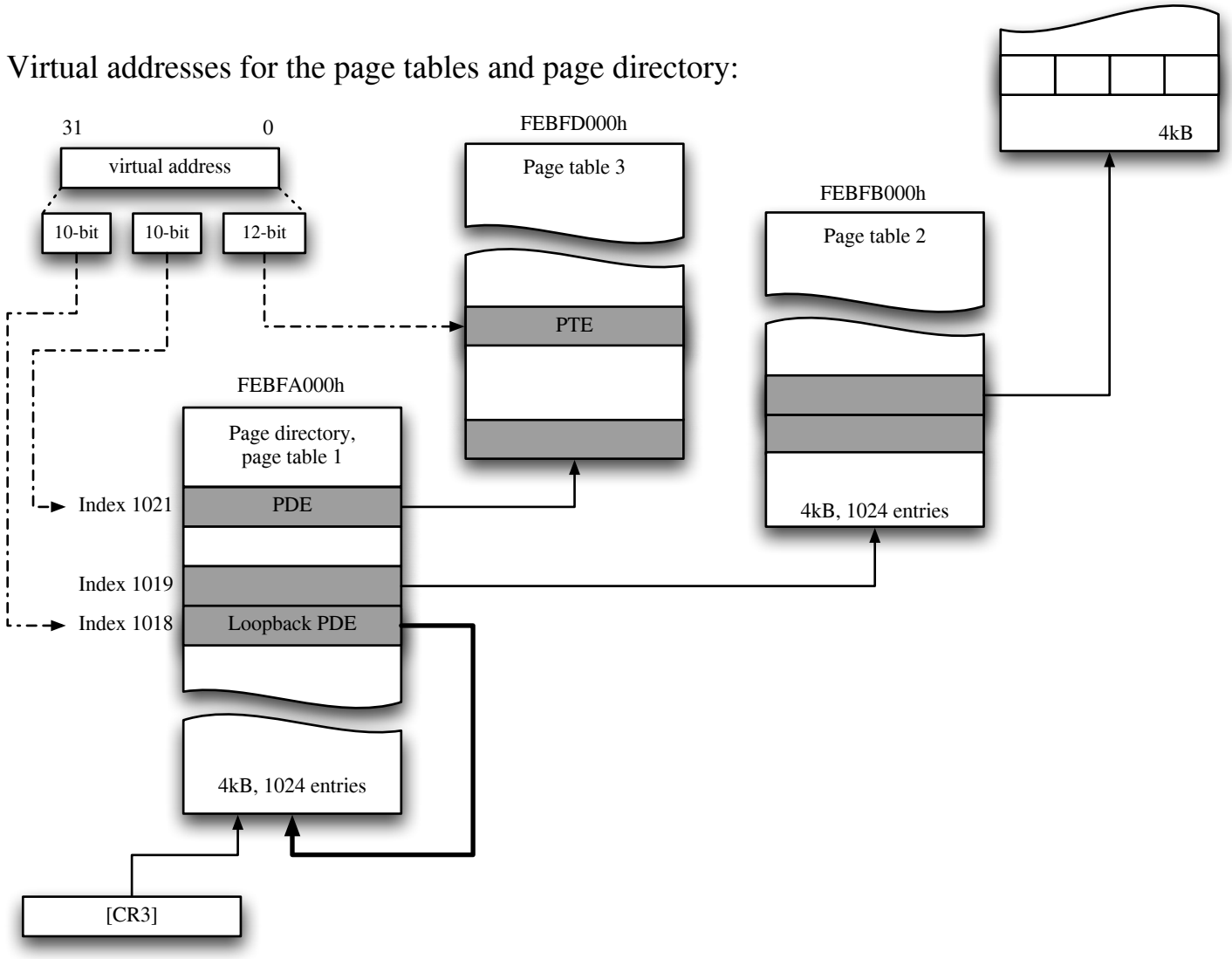
Page tables icon used in lecture:



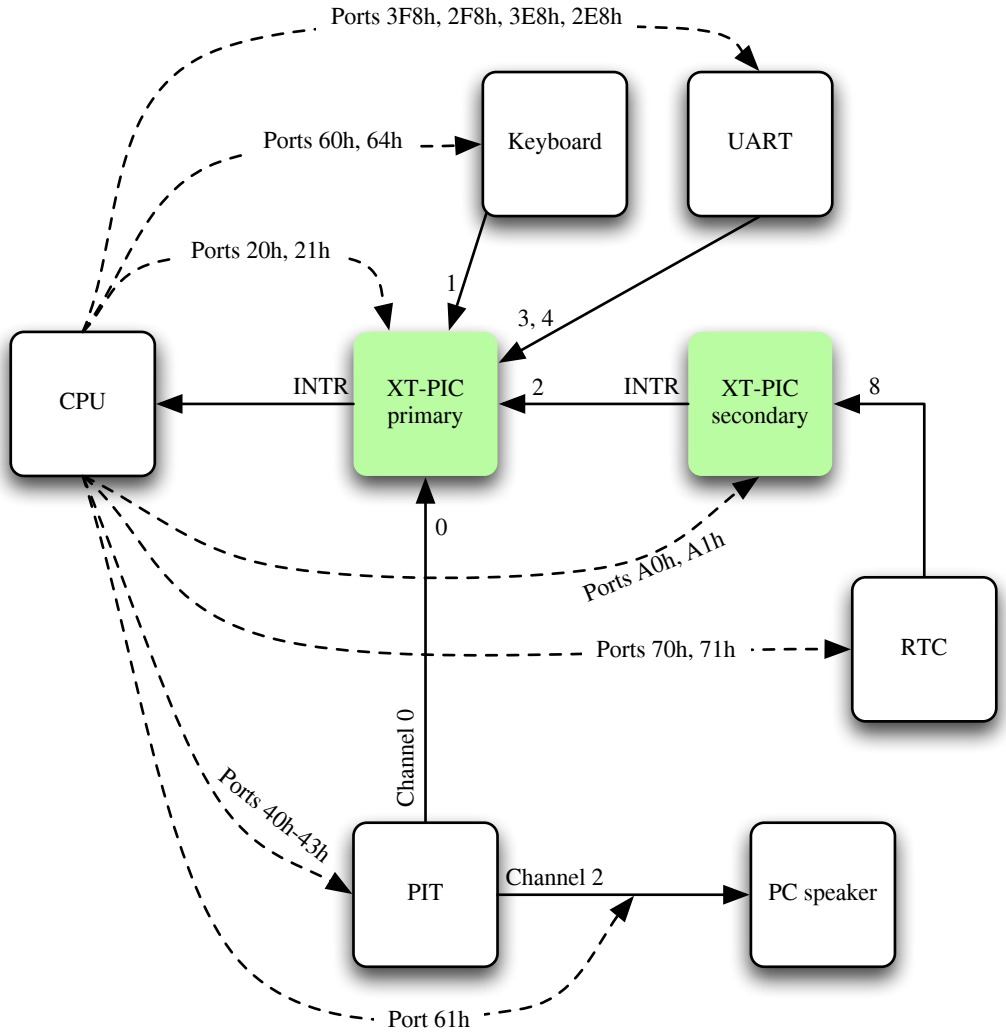
Implementing the kernel's address space:



Virtual addresses for the page tables and page directory:

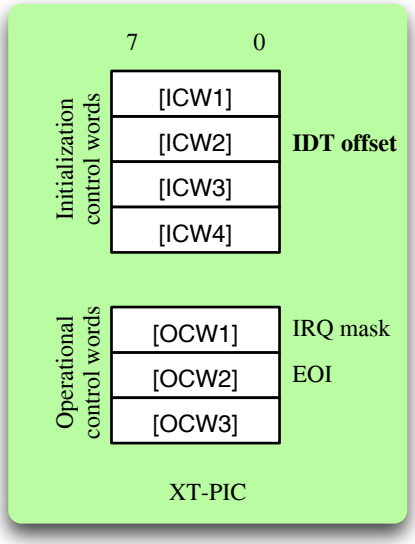


Legacy devices:

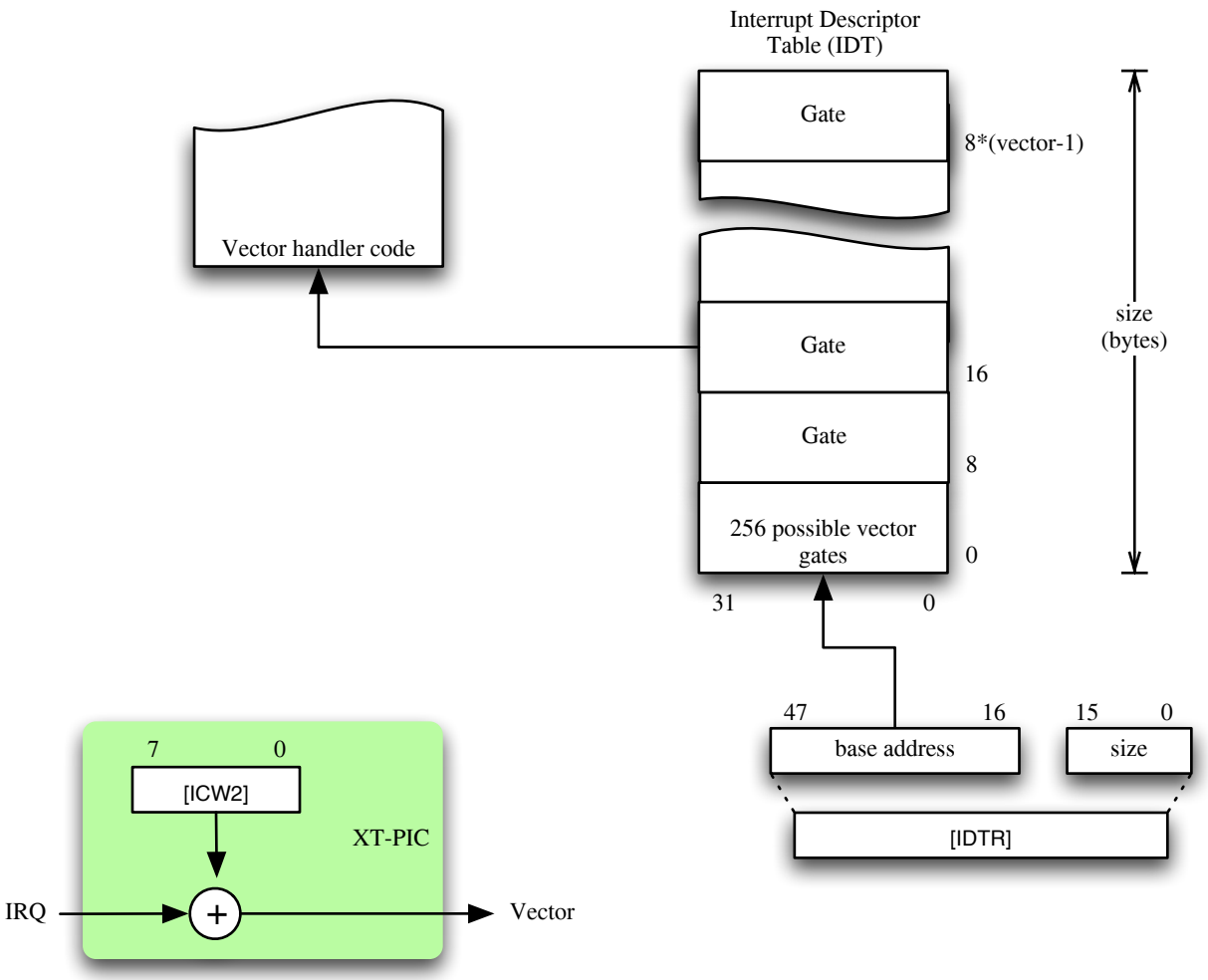


Device	Model	Description
Keyboard	i8042	Keyboard controller
PIT	i8253	Programmable Interval Timer
RTC	MC146818	Real Time Clock
UART	8250	Legacy serial port controller
XT-PIC	i8259	Legacy Programmable Interrupt Controller

IRQ	Device
0	Timer (channel 0 output of the PIT)
1	Keyboard
2	Cascaded to slave XT-PIC INTR output
3	Serial ports 2 and 4
4	Serial ports 1 and 3
5	Parallel port 2
6	Floppy disk
7	Parallel port 1
8	Real time clock (RTC)
9	
10	
11	
12	PS/2 mouse
13	
14	Primary IDE controller
15	Secondary IDE controller



Vectors:



Vector	Error Code	Function
0		Divide error
1		Reserved
2		NMI interrupt (nonmaskable external interrupt)
3		Breakpoint (INT 3)
4		Overflow
5		Bound range exceeded
6		Invalid opcode (and UD2 instruction)
7		Device not available (also WAIT/FWAIT instructions)
8	Yes (zero)	Double fault
9		Coprocessor
10	Yes	Invalid TSS
11	Yes	Segment not present
12	Yes	Stack-segment fault
13	Yes	General protection
14	Yes	Page fault
15		Reserved
16		x87 FPU floating-point (also WAIT/FWAIT instructions)
17	Yes (zero)	Alignment check
18		Machine check
19		SIMD floating-point exception
20-31		Reserved
32-255		User defined interrupts (external interrupts or INT *n* instruction)

Inlined assembler examples:

```
INLINE void ia32_cpuid( word_t index,
    word_t* eax, word_t* ebx, word_t* ecx, word_t* edx)
{
    __asm__ ("cpuid"
        : "=a" (*eax), "=b" (*ebx), "=c" (*ecx), "=d" (*edx)
        : "a" (index)
        );
}
```

```
INLINE u64_t ia32_rdtsc(void)
{
    u64_t value;
    __asm__ __volatile__ ("rdtsc" : "=A"(value));
   return value;
}
```

```
INLINE void ia32_wrmsr(const u32_t reg, const u64_t val)
{
    __asm__ __volatile__ ("wrmsr"  :  : "A"(val), "c"(reg));
}
```

General form: __asm__ __volatile__ (*assembler* : *outputs* : *inputs* : *clobbers*);

Constraints:

m	General memory operand
r	General purpose register
i	Constant integer value known at compile time
A	Specifies [eax] and [edx] for a 64-bit integer, with [edx] holding the most significant bits.
a	[eax]
b	[ebx]
c	[ecx]
d	[edx]
D	[edi]
S	[esi]
N	Constant in range 0 to 255 for the out instruction.
0, 1, 2, ... 9	Specify an additional constraint for the operand matching this position.

Constraint modifiers:

=	Write-only operand
+	A read and write operand