Kernel entry and exit:

Privileged mode

CS.privilege == 0

int, sysenter, page faults, device interrupts, exceptions

User mode

CS.privilege == 3

iret, sysexit

Interruption stack frame

CS

flags

instruction pointer

[ESP]

after

before

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

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

Interrupt stack frame

CS

flags

stack pointer

SS

[ESP]

after

before

TSS: Task State Segment

SS0

ESP0

stack top

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:

**PDE** — Page Directory Entry

- 20 bits
- target page physical address
- 3-bits unused
- global (ignored)
- super page
- reserved (set to 0)
- accessed
- cache disabled
- write-through
- user privilege
- writable
- valid/present

**PTE** — Page Table Entry

- 20 bits
- target page physical address
- 3-bits unused
- global
- page table attribute index
- dirty
- accessed
- cache disabled
- write-through
- user privilege
- writable
- valid/present

Page fault error code

- page fault error code
- reserved

Page tables icon used in lecture:

- Page directory index
- page table index
- data page physical location
- [CR3]

- 4GB virtual address space, sparsely populated
- 4MB data page
- 4kB, 1024 entries
- page directory physical location
- page table physical location
- superpage physical location

- 4kB, 1024 entries
- 4kB data page
- 4MB virtual address region, sparsely populated
- 4MB virtual address
- 31
- 0
- virtual address
- 10-bit
- 10-bit
- 12-bit
- data byte offset
- target page physical address
- present
- write fault
- fault while [CS].privilege == 3
- dirty
- accessed
- cache disabled
- write-through
- user privilege
- writable
- valid/present
- PDE reserved bits fault
Implementing the kernel's address space:

The kernel page tables are initially defined in the master kernel page directory; their PDEs are copied to the active page directory on demand.

Removal of a 4kB kernel page mapping is visible in all address spaces due to the shared kernel page tables.

Virtual addresses for the page tables and page directory:

4kB, 1024 entries

4kB, 1024 entries

Page directory, page table 1

Page table 3

Page table 2

Virtual addresses for the page tables and page directory:
Legacy devices:

<table>
<thead>
<tr>
<th>Device</th>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyboard</td>
<td>i8042</td>
<td>Keyboard controller</td>
</tr>
<tr>
<td>PIT</td>
<td>i8253</td>
<td>Programmable Interval Timer</td>
</tr>
<tr>
<td>RTC</td>
<td>MC146818</td>
<td>Real Time Clock</td>
</tr>
<tr>
<td>UART</td>
<td>8250</td>
<td>Legacy serial port controller</td>
</tr>
<tr>
<td>XT-PIC</td>
<td>i8259</td>
<td>Legacy Programmable Interrupt Controller</td>
</tr>
</tbody>
</table>

**IRQ**

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

User defined interrupts (external interrupts or \texttt{INT} \textit{n} instruction)

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