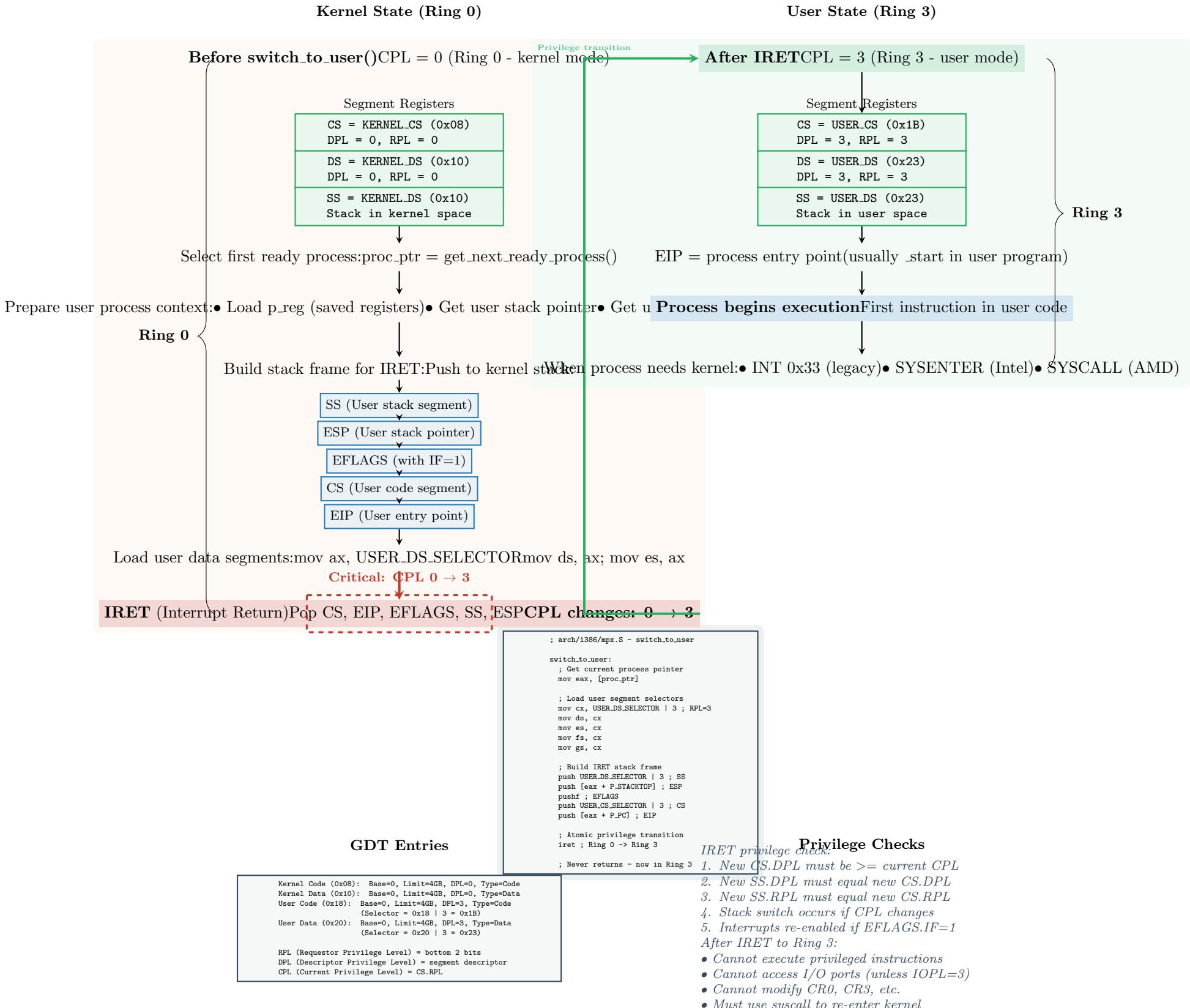# Kernel to Userspace Transition

Ring 0 → Ring 3 Privilege Level Change (switch_to_user)

Source: arch-specific code (i386: arch/i386/mpx.S, switch_to_user)

Kernel State (Ring 0)   **User State (Ring 3)**

Privilege transition

Before switch_to_user() CPL = 0 (Ring 0 - kernel mode)   **After IRET** CPL = 3 (Ring 3 - user mode)

Segment Registers   Segment Registers

```
CS = KERNEL_CS (0x08)
DPL = 0, RPL = 0
```
```
DS = KERNEL_DS (0x10)
DPL = 0, RPL = 0
```
```
SS = KERNEL_DS (0x10)
Stack in kernel space
```

```
CS = USER_CS (0x1B)
DPL = 3, RPL = 3
```
```
DS = USER_DS (0x23)
DPL = 3, RPL = 3
```
```
SS = USER_DS (0x23)
Stack in user space
```

Ring 3

Select first ready process: proc_ptr = get_next_ready_process()   EIP = process entry point (usually _start in user program)

Prepare user process context: • Load p_reg (saved registers) • Get user stack pointer • Get u **Process begins execution** First instruction in user code

Ring 0

Build stack frame for IRET: Push to kernel stack When process needs kernel: • INT 0x33 (legacy) • SYSENTER (Intel) • SYSCALL (AMD)

SS (User stack segment)

ESP (User stack pointer)

EFLAGS (with IF=1)

CS (User code segment)

EIP (User entry point)

Load user data segments: mov ax, USER_DS_SELECTOR mov ds, ax; mov es, ax

Critical: CPL 0 → 3

IRET (Interrupt Return) Pop CS, EIP, EFLAGS, SS, ESP **CPL changes: 0 → 3**

```
; arch/i386/mpx.S - switch_to_user

switch_to_user:
  ; Get current process pointer
  mov eax, [proc_ptr]

  ; Load user segment selectors
  mov cx, USER_DS_SELECTOR | 3 ; RPL=3
  mov ds, cx
  mov es, cx
  mov fs, cx
  mov gs, cx

  ; Build IRET stack frame
  push USER_DS_SELECTOR | 3 ; SS
  push [eax + P_STACKTOP] ; ESP
  pushf ; EFLAGS
  push USER_CS_SELECTOR | 3 ; CS
  push [eax + P_PC] ; EIP

  ; Atomic privilege transition
  iret ; Ring 0 -> Ring 3

  ; Never returns - now in Ring 3
```

GDT Entries

```
Kernel Code (0x08):  Base=0, Limit=4GB, DPL=0, Type=Code
Kernel Data (0x10):  Base=0, Limit=4GB, DPL=0, Type=Data
User Code (0x18):    Base=0, Limit=4GB, DPL=3, Type=Code
                     (Selector = 0x18 | 3 = 0x1B)
User Data (0x20):    Base=0, Limit=4GB, DPL=3, Type=Data
                     (Selector = 0x20 | 3 = 0x23)

RPL (Requestor Privilege Level) = bottom 2 bits
DPL (Descriptor Privilege Level) = segment descriptor
CPL (Current Privilege Level) = CS.RPL
```

Privilege Checks

IRET privilege check:
1. New CS.DPL must be >= current CPL
2. New SS.DPL must equal new CS.DPL
3. New SS.RPL must equal new CS.RPL
4. Stack switch occurs if CPL changes
5. Interrupts re-enabled if EFLAGS.IF=1
After IRET to Ring 3:
• Cannot execute privileged instructions
• Cannot access I/O ports (unless IOPL=3)
• Cannot modify CR0, CR3, etc.
• Must use syscall to re-enter kernel

Note: Each process has its own CR3 (page directory) loaded before switch_to_user(). All processes share kernel code mapped at high addresses (kernel space).