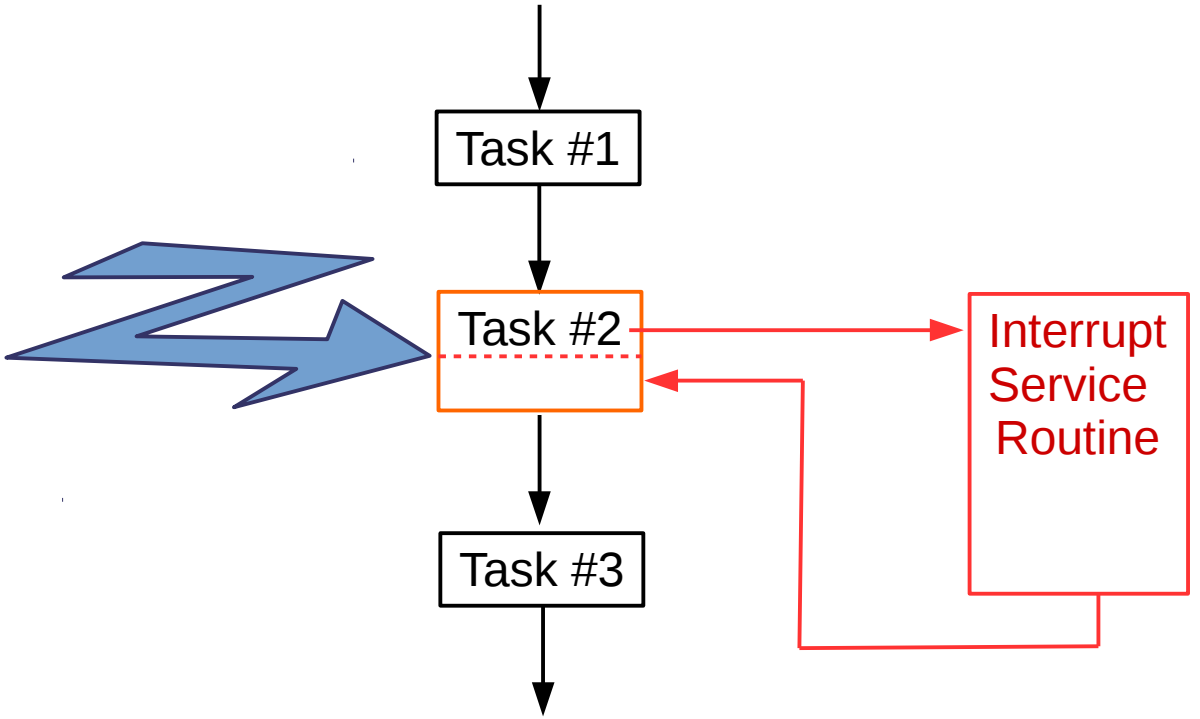
Interrupts

- Interrupt is a CPU mechanism to respond for specific internal or external events
- Internal events may occur from software requested command
- External events may occur from signal received from input/output devices
- Interrupts can be random (asynchronous) or interval (synchronous)
- When interrupt occurs, CPU pauses current task and switches to interrupt service routine
- After servicing interrupt, CPU resumes paused task



Interrupts

Interrupt Categories

Hardware Interrupt

Hardware interrupts are typically generated by hardware and asynchronously occurring. Hardware interrupts can be issued by:

- I/O devices (keyboard, network adapter, etc.)
- Interval timers
- Other CPUs (on multiprocessor systems)

Software Interrupt

- Software interrupt is produced by CPU while processing instructions.
- Usually caused by ***int*** (for 32/64-bit mode) or ***syscall*** (for 64-bit mode) instructions

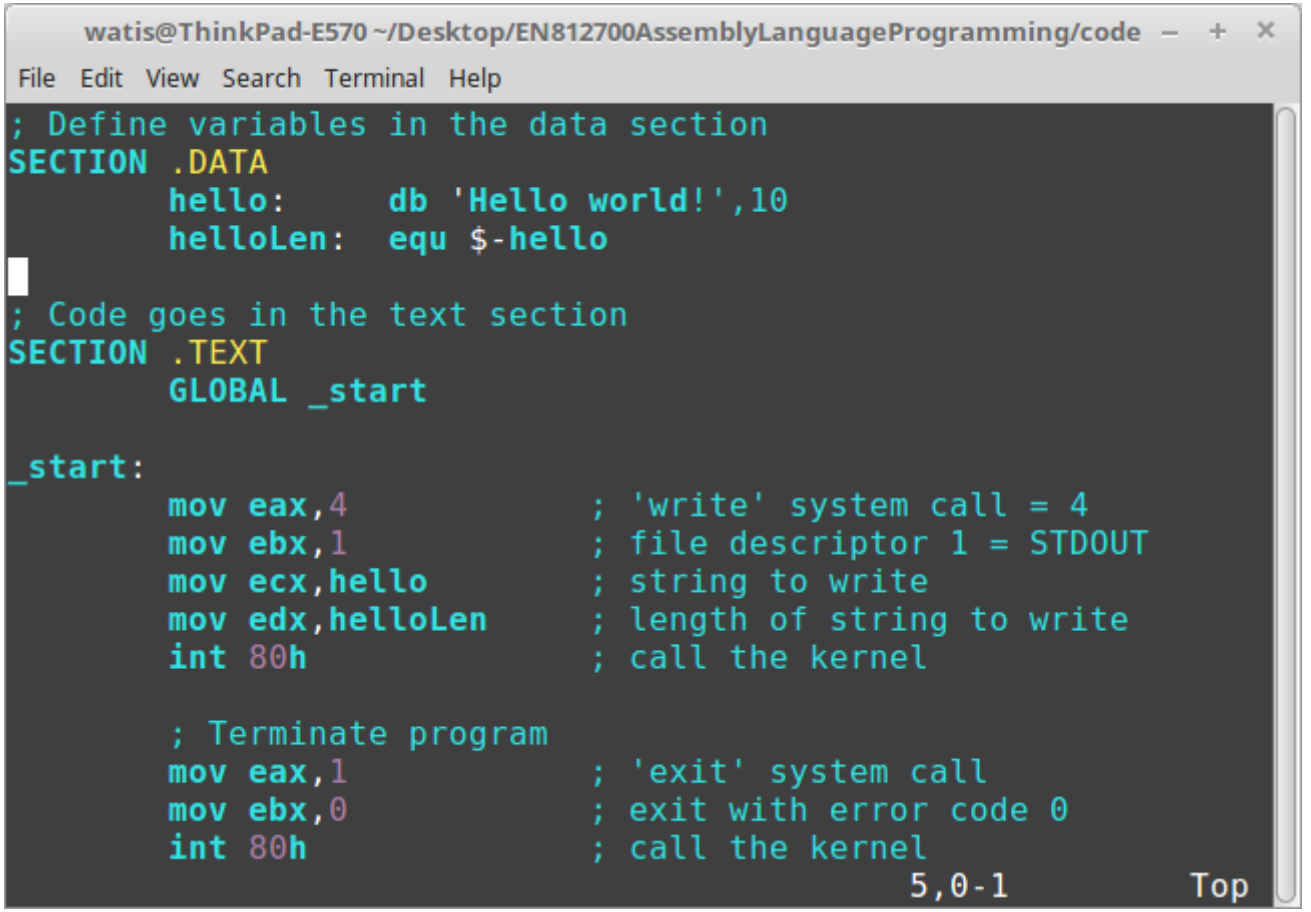
Exception

An exception is a term for an interrupt that is caused by the current process and needs attention of the kernel. Exceptions are typically divided into categories as follows:

- Faults → error occurs, but can be corrected, and continued
- Traps → instructions are executed and paused, used for debugging
- Abort → serious error occurs, process triggered error must be terminated

Interrupts

Example: int 80h



Interrupts

Interrupt Types and Levels

Interrupt Types

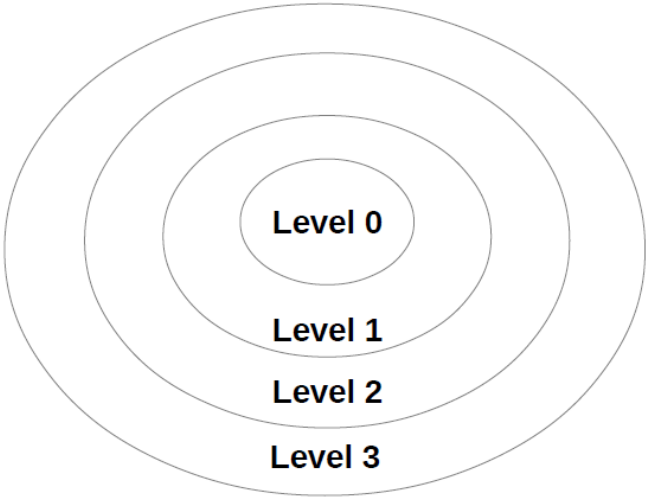
The two different types or kinds of interrupts are:

- Maskable interrupts → can be ignored or delayed services
- Non-maskable interrupts (NMI) → must be handled immediately

Privilege Levels

Privilege level defines how deep the interrupt can access hardware

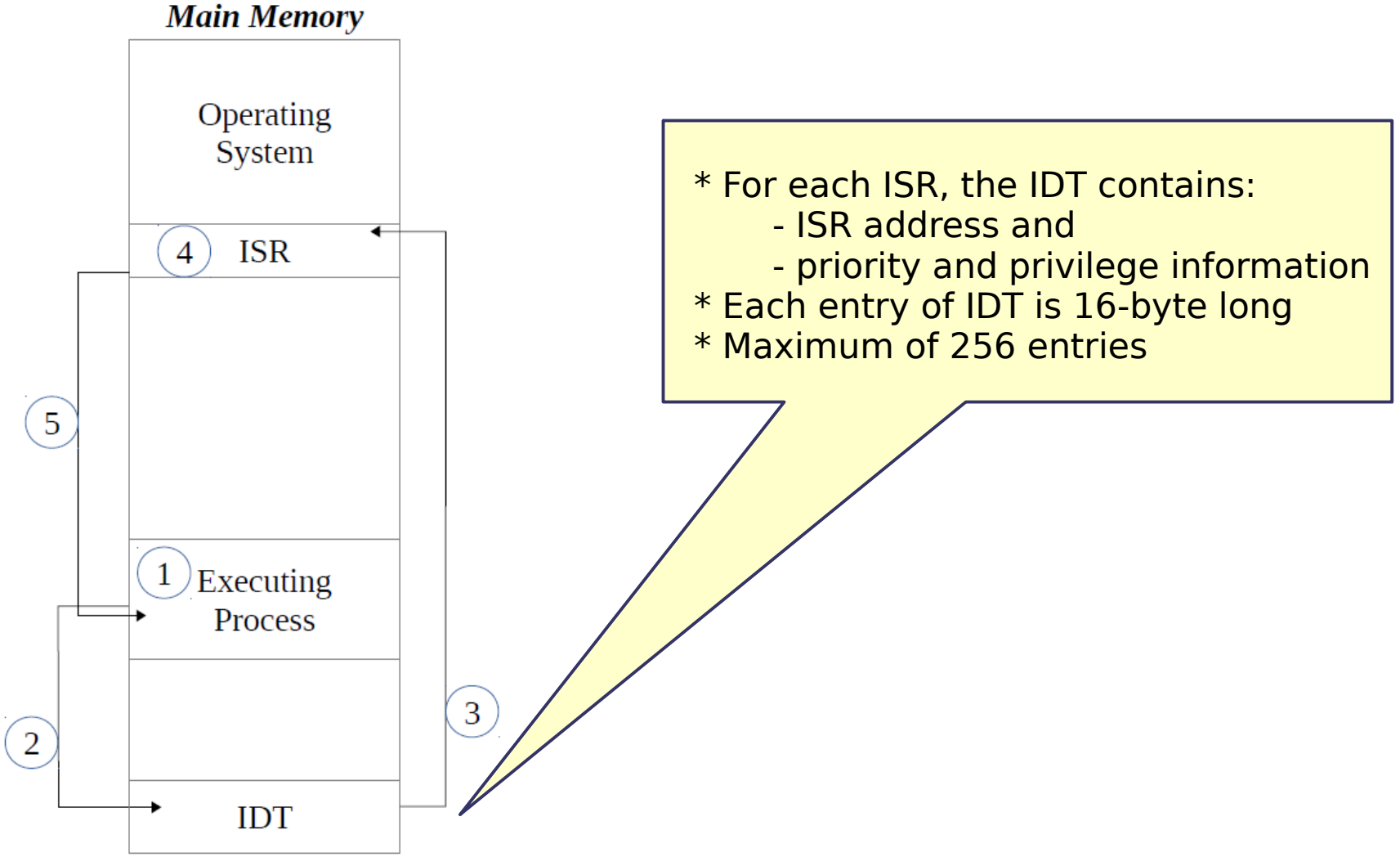
Level	Description
Level 0	Full access to all hardware resources (no restrictions). Used by only the lowest level OS functions.
Level 1	Somewhat restricted access to hardware resources. Used by library routines and software that interacts with hardware.
Level 2	More restricted access to hardware resources. Used by library routines and software that has limited access to some hardware.
Level 3	No direct access to hardware resources. Application programs run at this level.



Interrupts

Interrupt Processing Steps

1. Suspension → Current task (program) is suspended (paused)
2. Obtain ISR address → ISR address is located in Interrupt Descripter Table (IDT)
3. Jump to ISR → Save status of all registers used by suspended program
4. Execute ISR
5. Resume → Return to and continue execution of suspended program



Interrupts

Detail Process

1. Execution of current program is suspended
   - save **rip** and **rFlags** registers to stack
2. Obtain starting address of Interrupt Service Routine (ISR)
   - interrupt number multiplied by 16
   - used as offset into Interrupt Descriptor Table (IDT)
   - obtains ISR address (for that interrupt) from IDT
3. Jump to Interrupt Service Routine
   - set rip to address from IDT
4. Interrupt Service Routine Executes
   - save context (i.e., any additional registers altered)
   - process interrupt (specific to interrupt generated)
   - schedule any later data processing activities
5. Interrupted Process Resumption
   - resume scheduling based on OS scheduler
   - restore context
   - perform iret (to pop rFlags and rip registers)