Context switching when

When the OS receives a fault

1. suspends the execution of the running thread

2. terminate the thread

When the OS receives a System Clock Interrupt or a System Call Trap (I/O request)

3. suspends the execution of the running thread

4. saves its execution context

5. changes the thread's state to ready (timeout) or waiting (I/O request)

6. elects a new thread from the ones in the ready state

7. changes its state to running

8. restores its execution context

9. resumes its execution

When the OS receives any other I/O interrupt

1. executes the I/O operation

2. switches the thread, that was waiting for that I/O operation, into the ready state

3. resumes the execution of the current program

➡ **For each thread, the OS needs to keep track of its state (ready, running, waiting) and its execution context (registers, stack, heap and so on)**

TCB (Thread Control Block)

Data structure to record thread information

- Tid (thread id)

- State (as either running, ready, waiting)

- Registers (including eip and esp)

- Pointer to a Process Control Block (coming next week)

- User (forthcoming lecture on user space)