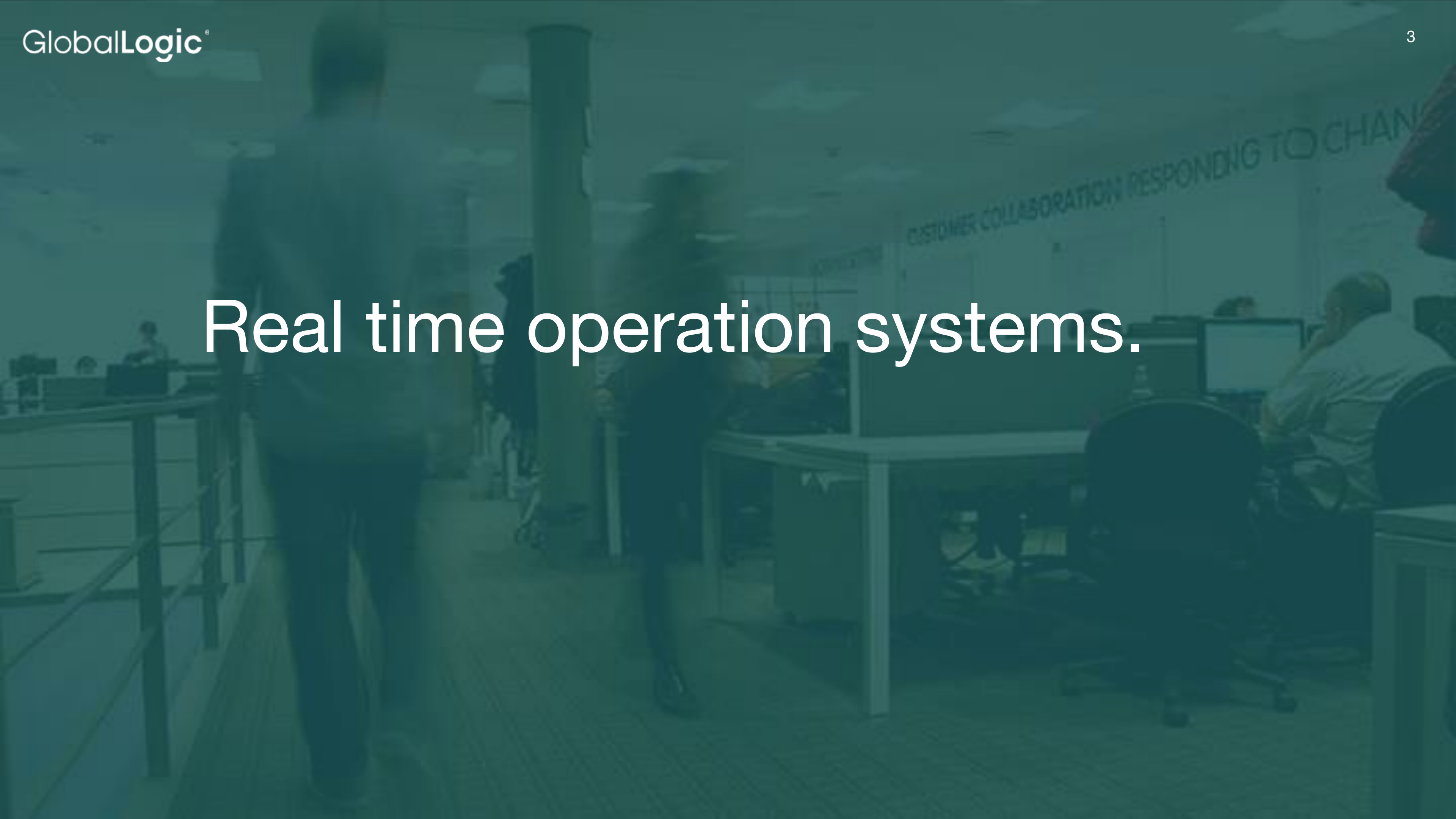
GlobalLogic®

Real time operation systems

Agenda

1. Real time operation systems.
2. Thread scheduling.
3. Context switch.
4. Interrupt latency.



Real time operation systems.

What is RTOS

A real-time operating system (**RTOS**) is an operating system (OS) intended to serve real-time applications that process data as it comes in, typically without buffer delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter increments of time.

RTOS

- A key characteristic of an RTOS is the level of its consistency concerning the amount of time it takes to accept and complete an application's task; the variability is 'jitter’.
- A 'hard' real-time operating system has less jitter than a 'soft' real-time operating system. The chief design goal is not high throughput, but rather a guarantee of a soft or hard performance category.
- An RTOS that can usually or generally meet a deadline is a soft real-time OS, but if it can meet a deadline deterministically it is a hard real-time OS.

Why we need RTOS

- For automotive, medical devices, spaceships, trains, drones, robots,
- All of this device need to guarantee reaction in a real time

Why we need RTOS

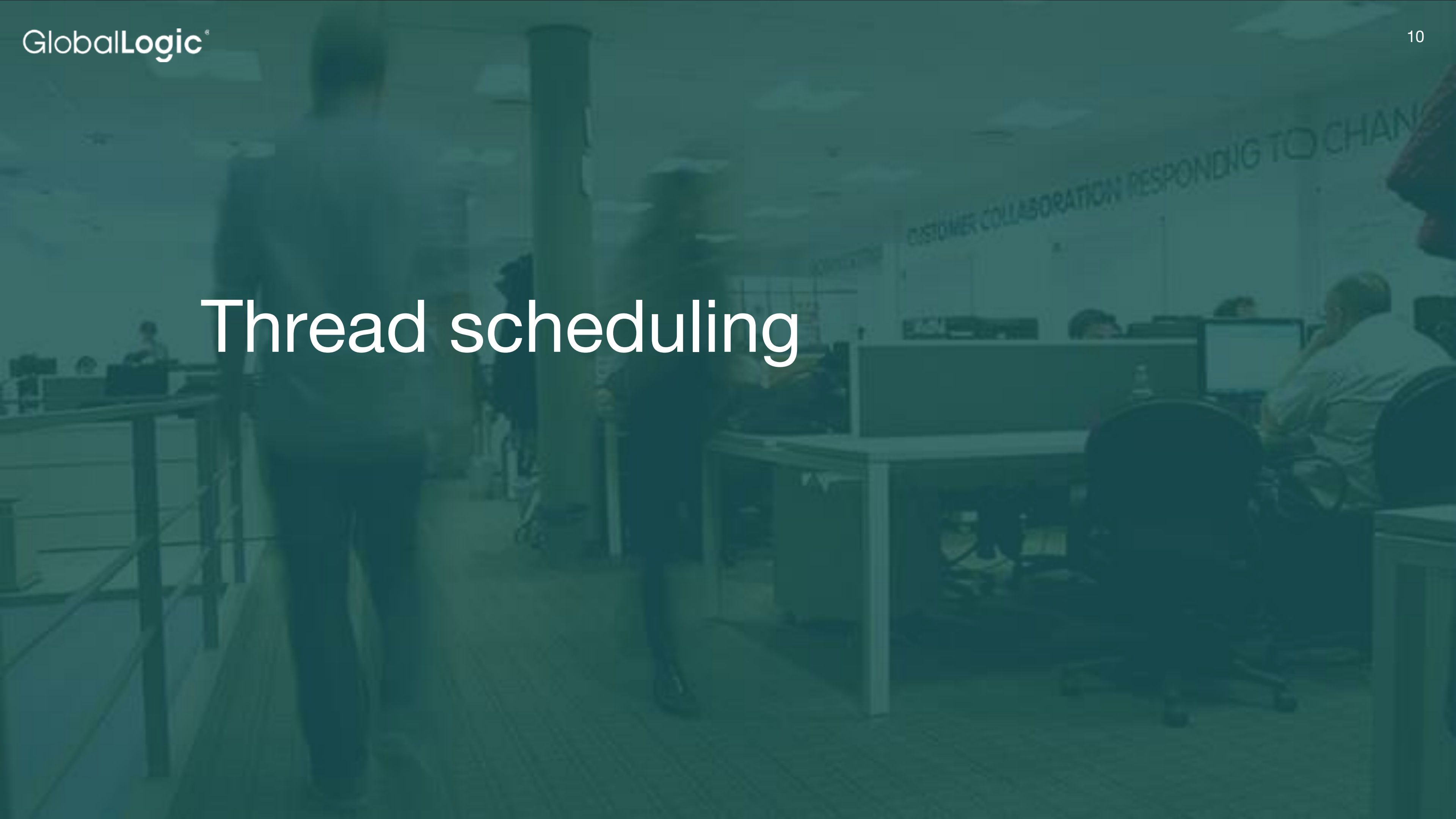
- **Small latency**: It is real-time after all!
- **Determinism**: Again, it is real-time. You need to know how long things take to process to make sure deadlines are met.
- **Structured Software**: With an RTOS, you are able divide and conquer in a structure manner. It's straight-forward to add additional components into the application.
- **Scalability**: An RTOS must be able to scale from a simple application to a complex one with stacks, drivers, file systems, etc.
- **Offload development**: An RTOS manages many aspects of the system which allows a developer to focus on their application. For example an RTOS, along with scheduling, generally handles power management, interrupt table management, memory management, exception handling, etc.

Terms

- **Interrupt Service Routine (ISR)**: Thread initiated by a hardware interrupt. An ISR is assert and it runs to completion. ISRs all share the same stack.
- **Tasks**: Thread that can block while waiting for an event to occur. Tasks are traditionally long living threads (as opposed to ISRs which run to completion). Each task has it's own stack which allows it to be long living.
- **Idle**: Lowest priority thread that only runs when no other thread is ready to execute. Generally Idle is just a special task with the lowest possible priority.

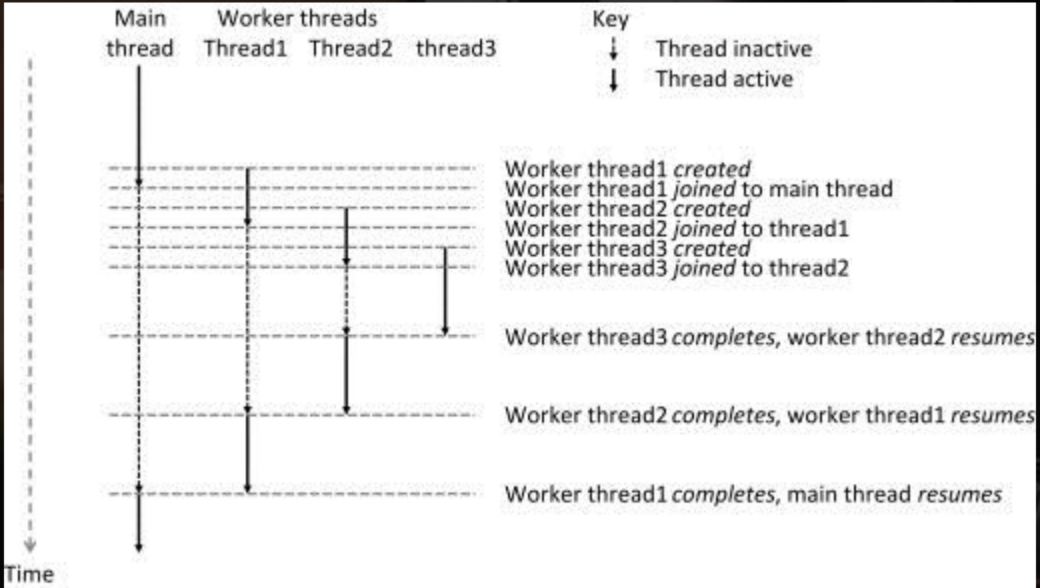
Components

- **Scheduler**: Preemptive scheduler that guarantees the highest priority thread it running.
- **Communication Mechanism**: Semaphores, Message Queues, Queues, etc.
- **Critical Region Mechanisms**: Mutexes, Gates, Locks, etc.
- **Timing Services**: Clocks, Timers, etc.
- **Memory Management**: Variable-size heaps, fixed-size heaps, etc.



Thread scheduling

Thread scheduling



Thread scheduling

Preemptive Scheduling: This is the most common type of RTOS scheduler. With a preemptive scheduler, a running thread continues until it either

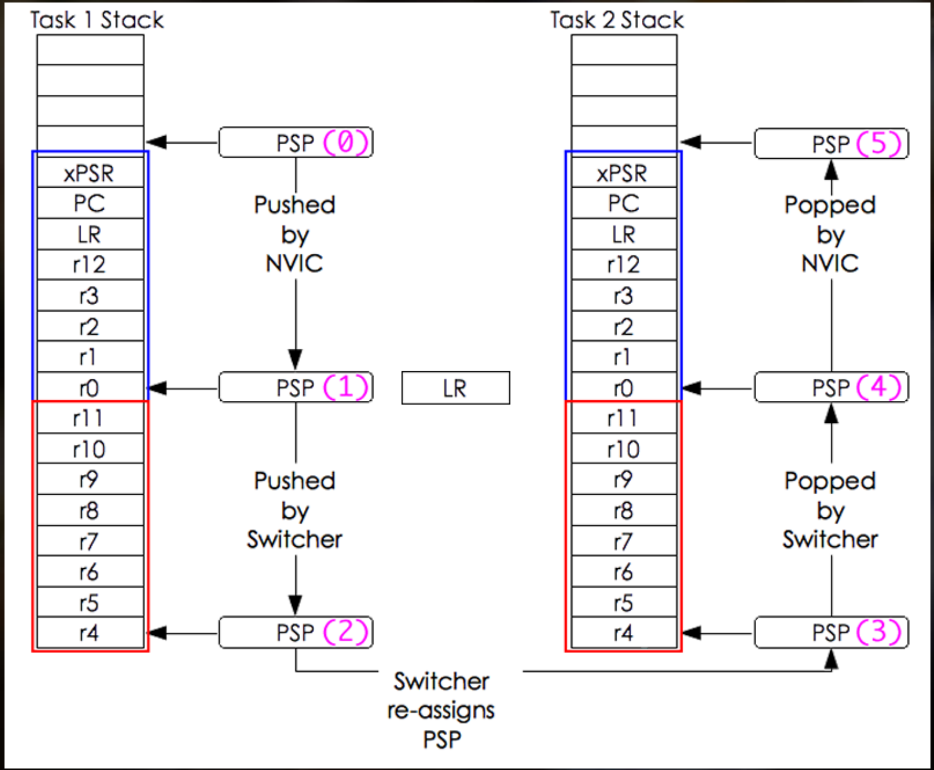
- finishes (e.g. an ISR completes)
- a higher priority thread becomes ready (in this case the higher priority thread preempts the lower priority thread)
- the thread gives up the processor while waiting for a resource (e.g. a task calls sleep()).

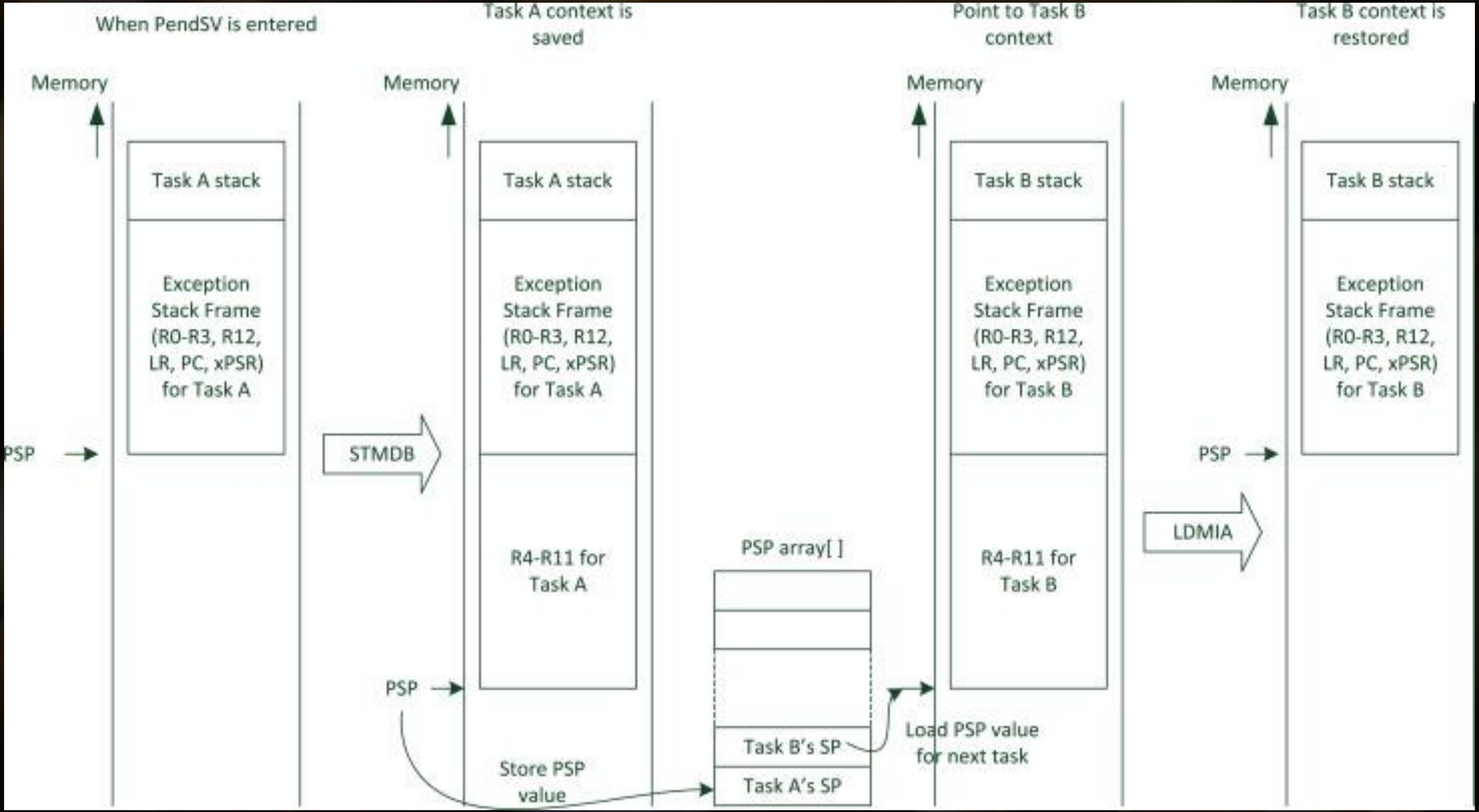
Thread scheduling

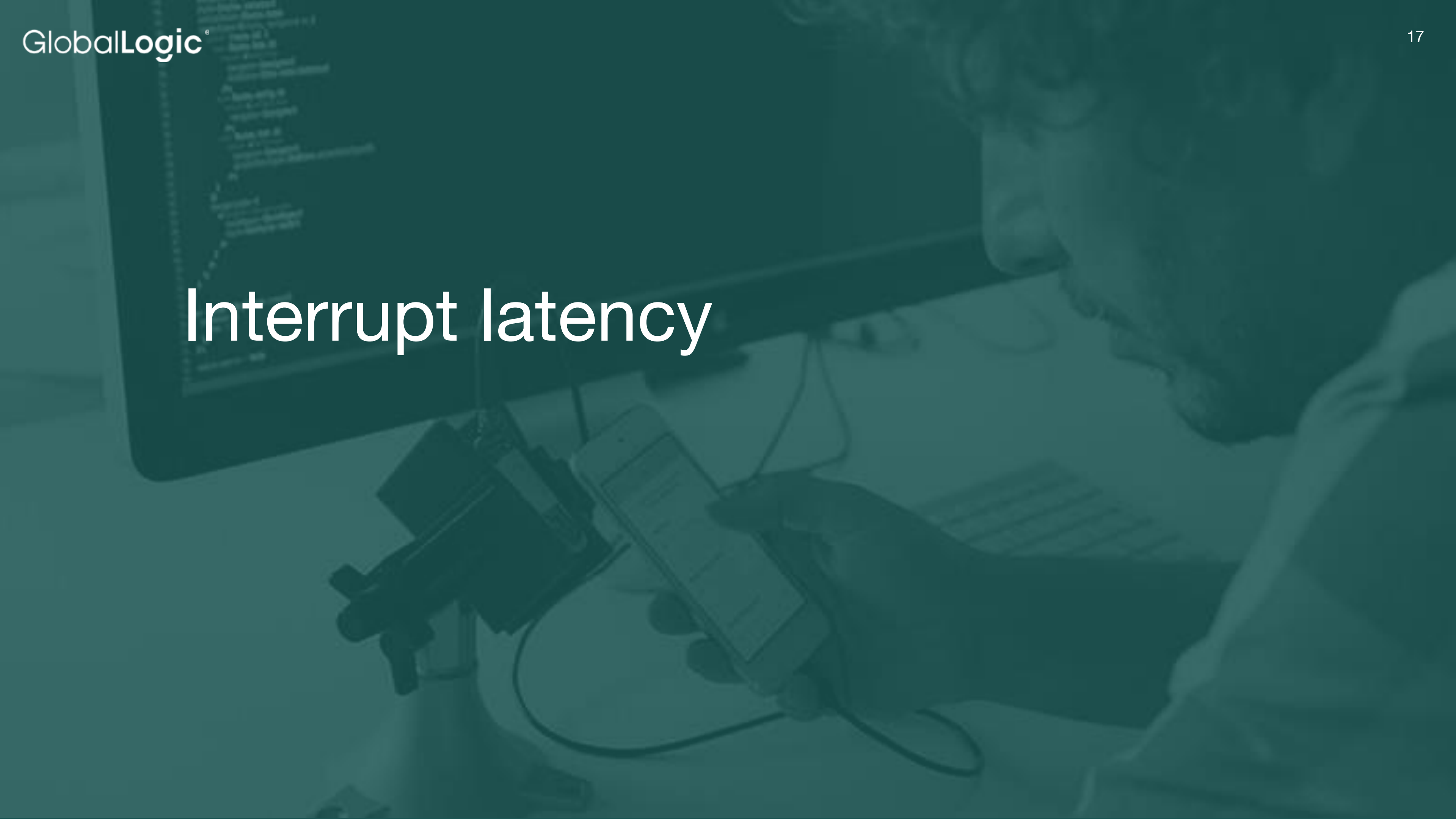
- Time-slice Scheduling: This type of scheduling guarantees that each thread is given a slot to execute. This type of scheduling is generally not conducive to real-time application.

Context switch

Context switch

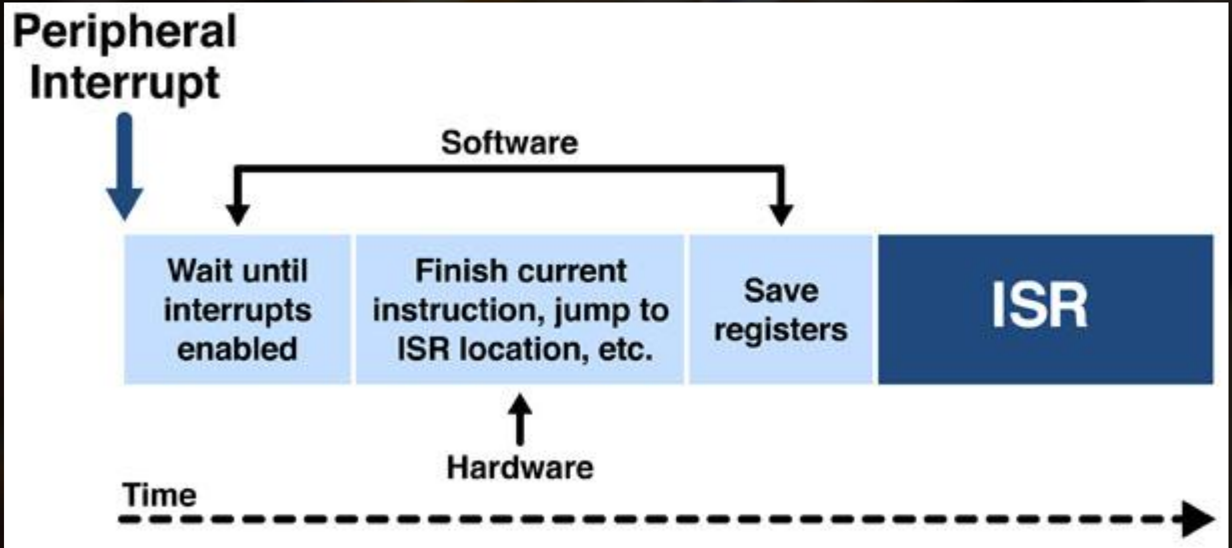



When PendSV is entered
Task A context is saved
Point to Task B context
Task B context is restored
Memory
Task A stack
Exception Stack Frame (R0-R3, R12, LR, PC, xPSR) for Task A
PSP
STMDB
R4-R11 for Task A
PSP array[ ]
Task B's SP
Task A's SP
Store PSP value
Load PSP value for next task
Task B stack
Exception Stack Frame (R0-R3, R12, LR, PC, xPSR) for Task B
R4-R11 for Task B
LDMIA



Interrupt latency

Interrupt latency



Thank you


GlobalLogic®