

DEVICE DRIVERS AND INTERRUPTS SERVICE MECHANISM

Lesson-10: Context, context Switching and interrupt latency

1. Context

An embedded system executes:

- multiple tasks (processes). An operating system facilitates this
- perform multiple actions or functions due to multiple sources of the interrupts. An interrupt service mechanism in the system facilitates this

- The multitasking and multiple ISRs execute even though there is only one processor by first saving the one program context and retrieving another program context

Current program's program counter, status word, registers, and other program-contexts

- Before executing new instructions of the new function, the current program's program counter are saved.
- Also status word, registers, and other program-contexts are saved, if not done automatically by the processor.
- Because the program counter, status word register and other registers are needed by the newly called function.

Program Counter- a part of the context of presently running program

- Getting an address (pointer) from where the new function begins , and loading that address into the program counter_and then executing the called function's instructions will change the running program at the CPU to a new program
- Program Counter- a part of the context of presently running program

Context

- A context of a program **must include** program counter as well as the program status word, stack pointer and **may include** the processor registers. The context may be at a register set or at the separate memory block for the stack.
- A register set or memory block can hold context information

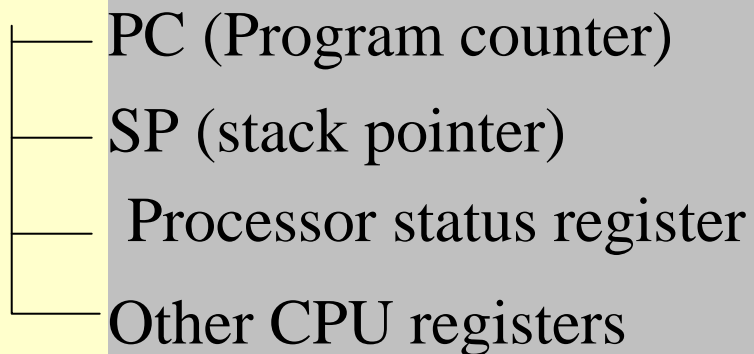
Context

- Present CPU state, which means registers and
- May include ID of the process interrupted and
- Function's local variables.

What should constitute the context?

- Depends on the processor of the system or operating system supervising the program.

Current Program Context



A diagram showing a vertical list of four items: PC (Program counter), SP (stack pointer), Processor status register, and Other CPU registers. To the left of this list is a vertical line with four horizontal tick marks, each aligned with one of the items, indicating they are part of a single set or context.

- PC (Program counter)
- SP (stack pointer)
- Processor status register
- Other CPU registers



A rectangular box with a red border containing the text "Current Program context".

Current Program
context

An Example

- In an 8051 program, the program counter alone constitute the context for the processor interrupt service and multiple functions call mechanism.
- In 68HC11 program, the program counter, and CPU registers constitute the context for the processor interrupt service and multiple functions call mechanism.

2. Context Switching

New program executes with new context of called routine

Save current
routine context on
stack and load new
routine context

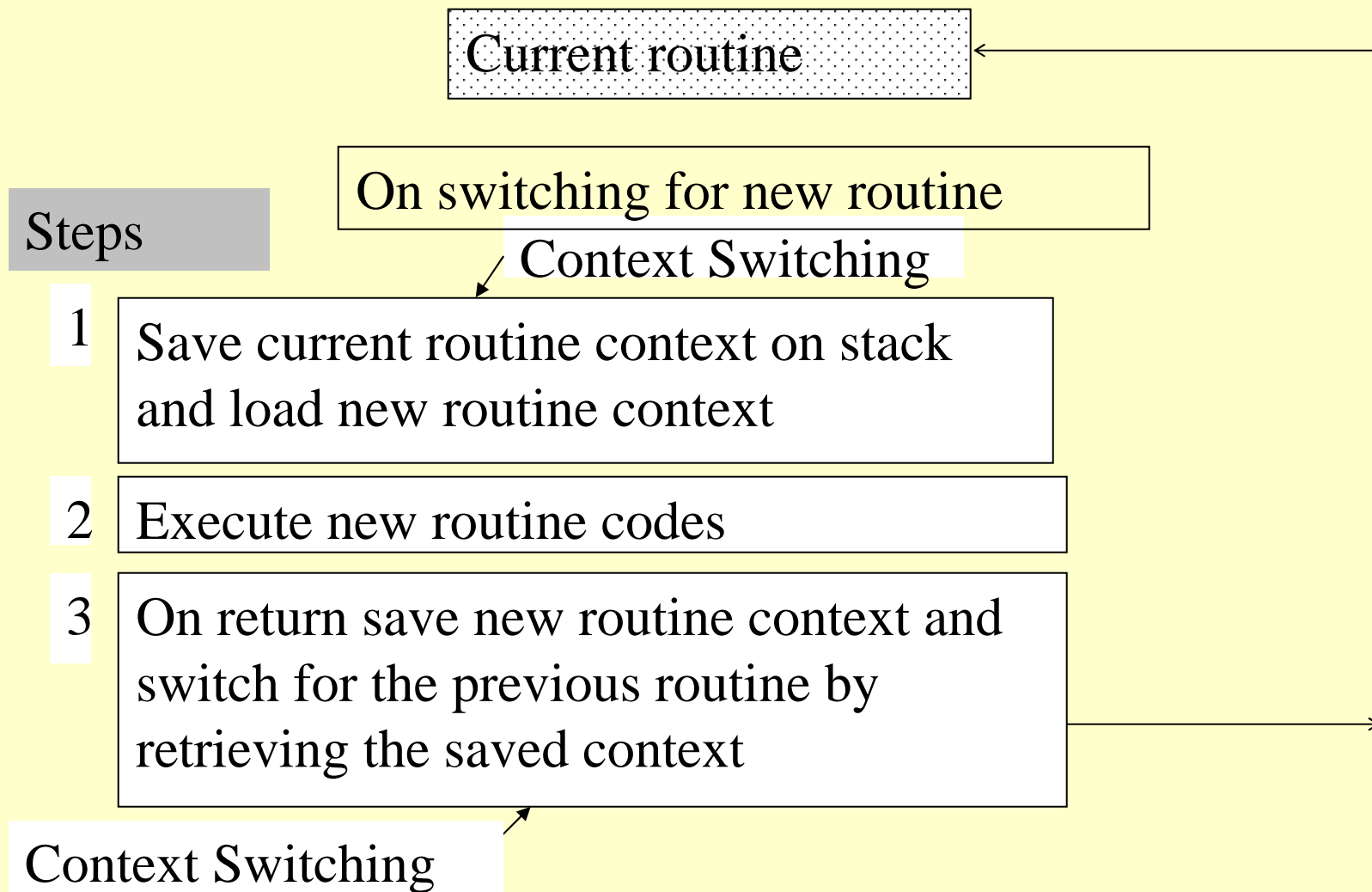
Context Switching

A diagram illustrating the context switching process. A light blue rectangular box at the top contains the text 'Save current routine context on stack and load new routine context'. A dashed arrow points from the bottom center of this box to a light blue rectangular box at the bottom containing the text 'Context Switching'.

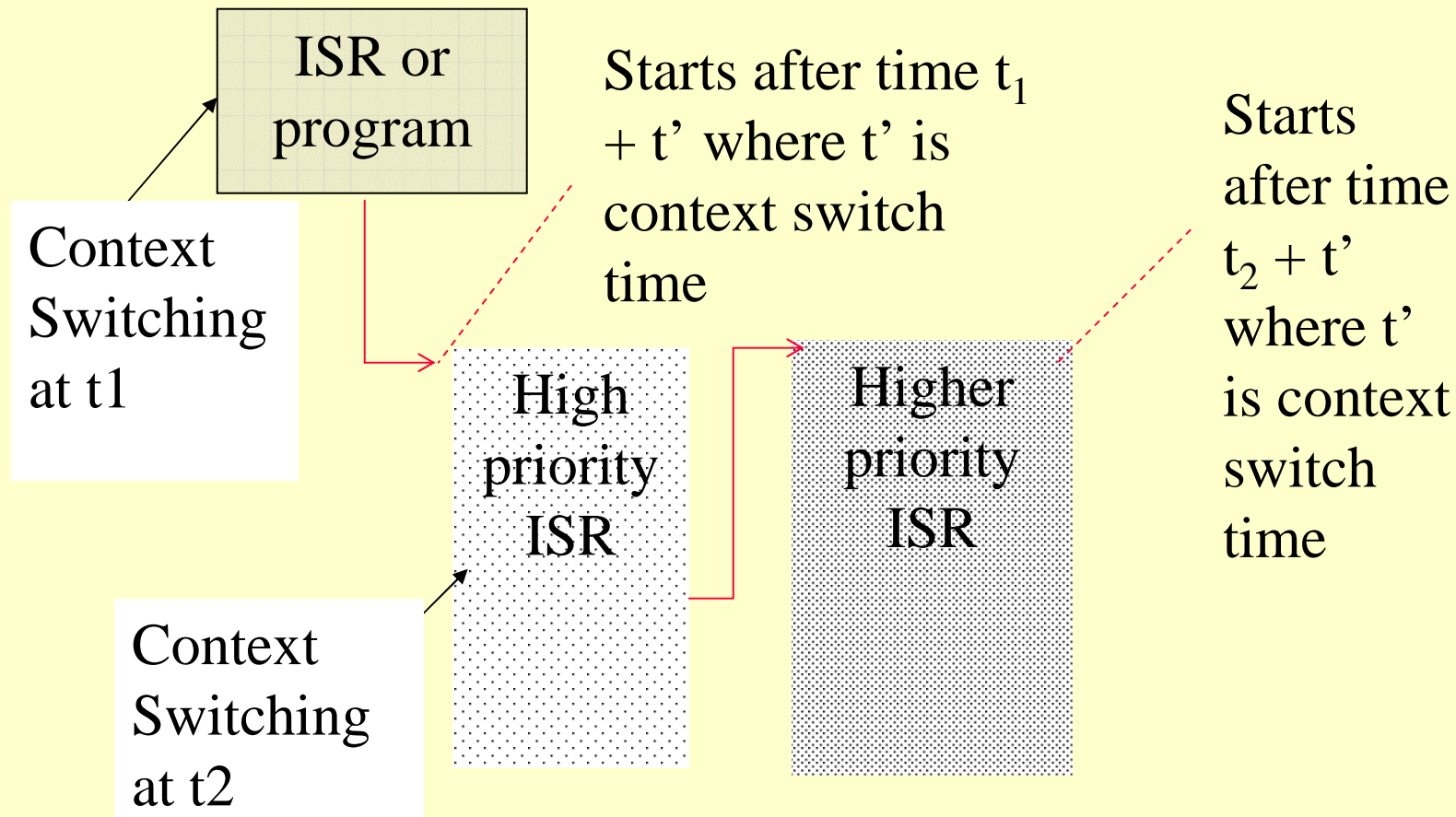
Context Switching on Interrupts

- Context switching means saving the context of interrupted routine (or function) or task and retrieving or loading the new context of the called routine or task to be executed next.

Context switching for new routine and another switch on return routine



Context switching in case of multiple interrupts



Fast Context Switching in ARM7 on Interrupt

- The interrupt mask (disable) flags set [Disable low priority interrupts.]
- Program Counter (PC) saved
- Current Program Status Register, CPSR copies into one for the saved program status register (SPSR), and CPSR stores the new status (interrupt source data or information).

- PC gets the new value as per the interrupt source from the vector table.

On return from the ISR, the context switching back to previous context

- (i) Program Counter (PC) retrieves.
- (ii) The corresponding SPSR copies back into CPSR.
- (iii) The interrupt mask (disable) flags are reset. [Enable again the earlier disabled low priority interrupts.]

Operating system

- OS provides for memory blocks to be used as stack frames if internal stack frames are not available at the processor being used in the system.

3. Classification of Processors Interrupt Service Mechanism from Context Saving Angle

8051

- 8051 interrupt-service mechanism is such that on occurrence of an interrupt service, the processor pushes the processor registers PCH (program counter higher byte) and PCL (program counter lower byte) on to the memory stack.
- The 8051 family processors do not save the context of the program (other than the absolutely essential program counter) and a context can save only by using the specific set of instructions for that. For example, using Push instructions at the ISR.

Advantage of Saving PC only in 8051

- It speeds up the start of ISR and returns from ISR but at a cost.
- The onus of context saving is on the programmer in case the context (SP and CPU registers other than PCL and PCH) to be modified on service of or on function calls during execution of the remaining ISR instructions.

68HC11 interrupt mechanism

- Processor registers save on to the stack whenever an interrupt service occurs. These are in the order of PCL, PCH, IYL, IYH, IXL, IXH, ACCA, ACCB, and CCR.
- 68HC11 thus does automatically save the processor context of the program without being so any instructed in the user program.
- As context saving takes processor time, it slows a little at the start of ISR and returns from ISR but at the a great advantage that the onus of context saving is not there on the programmer and there is no risk in case the context modifies on service or function calls

ARM7 interrupt mechanism

- ARM7 provides a mechanism for fast context switching between the two tasks, one current and one at the stack.

Summary

We learnt

- Each running-program has a *context* at an instant.
- Context means a CPU state (program counter, stack pointer(s), registers and program state (variables that should not be modified by another routine). The context must be saved on a *call* to another ISR or *task* or routine

We learnt :

- Context-Switching
- 8051, 68HC11 and ARM7 processors provide for different mechanisms for the context saving and switching .

End of Lesson 10 of Chapter 4