



A10.4 Interrupts

Operating systems, software and translators · A level · OCR H446 1.2.1,
AQA 7517 4.6.1.4, Eduqas A500QS 2.6 · about 40 min

Name _____

Class _____

Date _____

What this lesson is about

Polling versus interrupts, priorities, the stack and interrupt service routines in the fetch-decode-execute cycle.

- Polling
- Interrupts
- Interrupts and the fetch-decode-execute cycle
- A worked trace
- Polling or interrupts?

Questions

5 marks in all

1. When does the processor check for an interrupt? [1 mark]

- ☐ A At the end of each fetch-decode-execute cycle
- ☐ B Only when the program has finished
- ☐ C In the middle of executing each instruction
- ☐ D Once every second

2. Put these steps in order for an interrupt of higher priority than the current task. [1 mark]

Number the lines 1 to 5 to put them in the right order.

- ___ The saved register values are popped off the stack and the task resumes
- ___ The contents of the registers, including the PC, are pushed onto the stack
- ___ The processor finishes the current fetch-decode-execute cycle
- ___ The interrupt service routine runs
- ___ The address of the interrupt service routine is loaded into the PC

3. Why are the saved register values kept on a stack? [1 mark]

- ☐ A An ISR can itself be interrupted, and the routines must be resumed in reverse order
- ☐ B A stack is the fastest kind of memory
- ☐ C The stack is stored in the BIOS
- ☐ D A stack lets any saved value be removed first

4. A low-priority printer interrupt arrives while a high-priority power-failure ISR is running. What happens? [1 mark]

- ☐ A It waits until the power-failure ISR has finished
- ☐ B It interrupts the power-failure ISR
- ☐ C It is deleted
- ☐ D Both routines run at the same time on one core

5. Which are advantages of interrupts over polling?

[1 mark]

Tick every answer that is true.

- ☐ A No processor time is wasted checking devices that have nothing to report
- ☐ B Urgent events are dealt with at the end of the current cycle
- ☐ C They are simpler to program
- ☐ D They need no stack

The task: nested interrupts

Simulate the processor from the worked trace, with nesting. The inputs are: - `MAIN_LENGTH`, the number of instructions in the main program (6); main has priority 0; - `ISR_LENGTH`, the number of instructions in every ISR (2); - `arrivals`, a dictionary from a cycle number to a list of (name, priority) interrupts that arrive during that cycle. Priorities are positive integers; higher is more important. Keep the running routine's name, priority and next instruction number (starting at "main", 0, 0), a `stack` list and a `pending` list. Number cycles from 0. Each cycle: 1. print `cycle <c>: <routine> <instruction>` and move on to the next instruction; 2. if the routine has now run all its instructions: if it is main, stop the program; otherwise pop the routine it interrupted off the stack and print `return to <routine> at <instruction>`; 3. add this cycle's arrivals to `pending`; if the highest-priority pending interrupt has a priority greater than the running routine's, remove it from `pending`, push the running routine onto the stack, print `interrupt <name>: saved <routine> at <instruction>`, and start that ISR at instruction 0. Build every line from your variables. The robot stays still.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

MAIN_LENGTH = 6
ISR_LENGTH = 2
arrivals = {1: [("printer", 1)], 2: [("power", 3)], 3: [("keyboard", 1)], 4: [("timer", 2)]}

stack = []
pending = []
```

Plan your program here, then type it in and press Run.



Do it on the robot

www.bugbotlab.com/learn/a10-4-interrupts/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Trace the task's data by hand before running it. At which cycle does the keyboard interrupt finally start, and why so late?
2. Change the rule to "greater than or equal to". What changes in the trace, and why is that a bad idea?
3. How deep does the stack get? Add a line that prints the largest stack size, and invent arrivals that make it deeper.