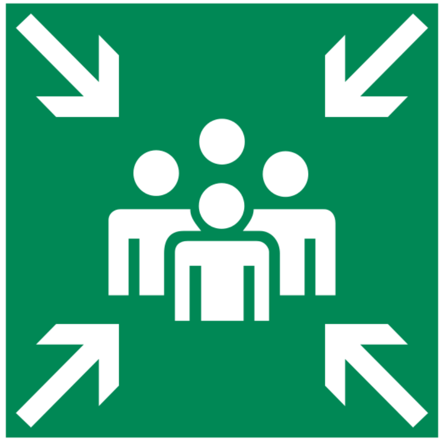
Topic V17

Polling, Interruptions,
and Watchdog Pooling
Readings: (Section 4.9)

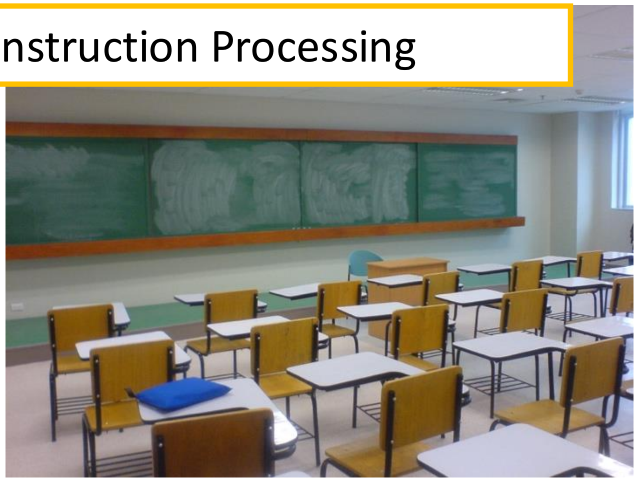
Exceptions and Interrupts

Where?



Exceptions and Interrupts

Normal Instruction Processing

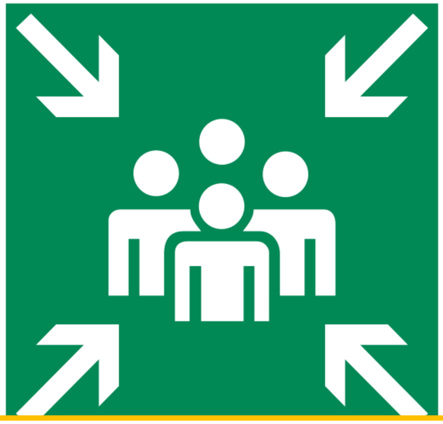


Leave routine cycle

Where?

Interruption

External Event



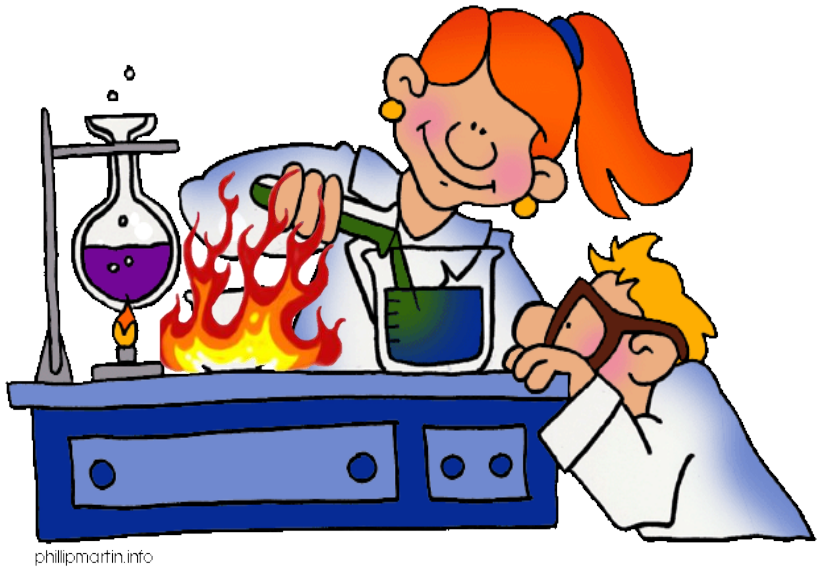
Go to Exception Handler

Exceptions and Interrupts

Exception

Interrupt

External Source



Exception and Interrupts

Exceptions:

Handle internal events to the processor:

```
li      s0, 0x0007
lw      s1, 0(s0)      # unaligned access (odd address)
```

unaligned access

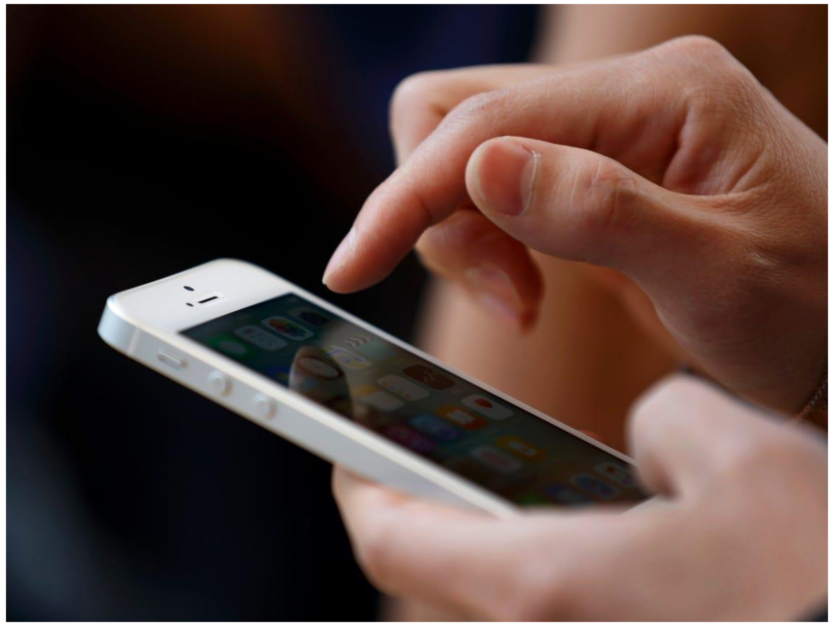
```
lw      s0, 0(zero)    # segmentation fault
```

access to wrong segment

Exceptions and Interrupts

Interruptions:

handle external events:



I/O Event

When exception occurs

What should the processor do when an exception occurs?

1. **Save the address of the instruction that caused the exception.**

   Where?

   In a special register called the User Exception Program Counter (uepc).

2. **Jump to a specified address** (Address stored in the User Trap Handler base address (utvec) register in RISC-V)**.**

   What code should be there?

   OS code to handle exceptions.

Things that the OS needs to know to handle an exception

1. **The reason for the exception**

    There are two ways to report the reason to the OS

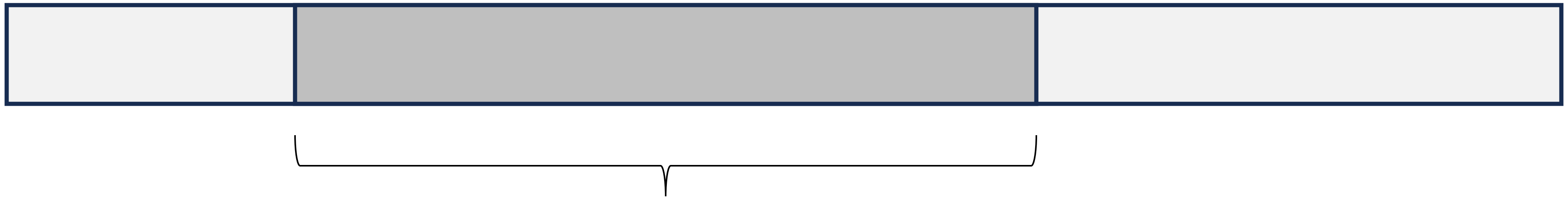
2. **Which instruction caused the exception**

    Instruction address is in UEPC

UEPC = User-level Exception Program Counter

Cause Register × Interrupt Vectors

Cause Register: a special register encodes the reason



Vectored Interrupts

utvec:

Contains the address of the handler.

Word aligned ⇒ LSB should always be 0.

LSB == 1 ⇒ it is in vectored mode:

Make LSB=0: it is the address of a table of pointers to exception handlers depending on its code.

- For all synchronous exceptions (caused by instruction execution):
  - PC ← BASE field
- For interrupts (using vector mode):
  - PC ← BASE field + 4 × causeNumber

Example:

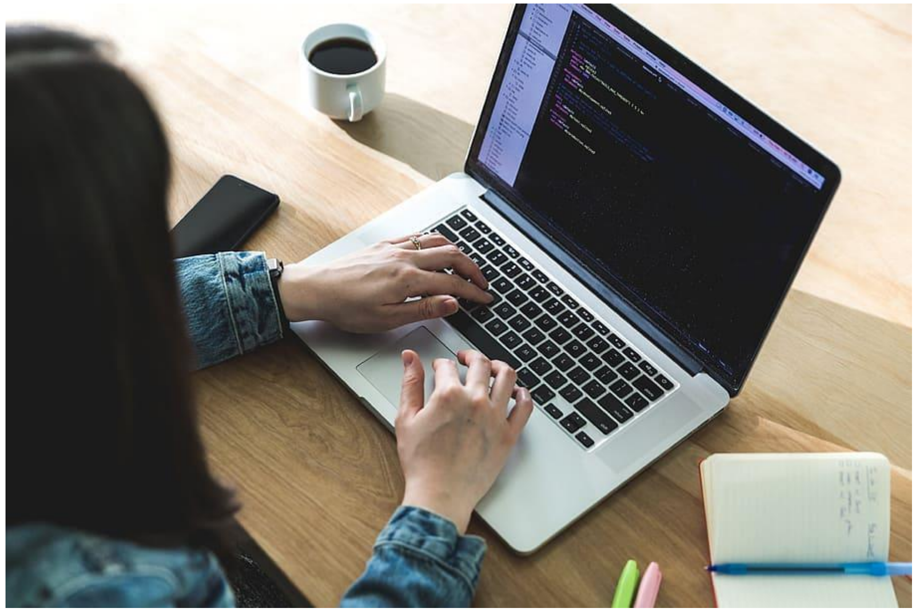
A user-mode timer interrupt:

PC ⟵ BASE+0x1c.

Polling vs. Interrupts

Polling

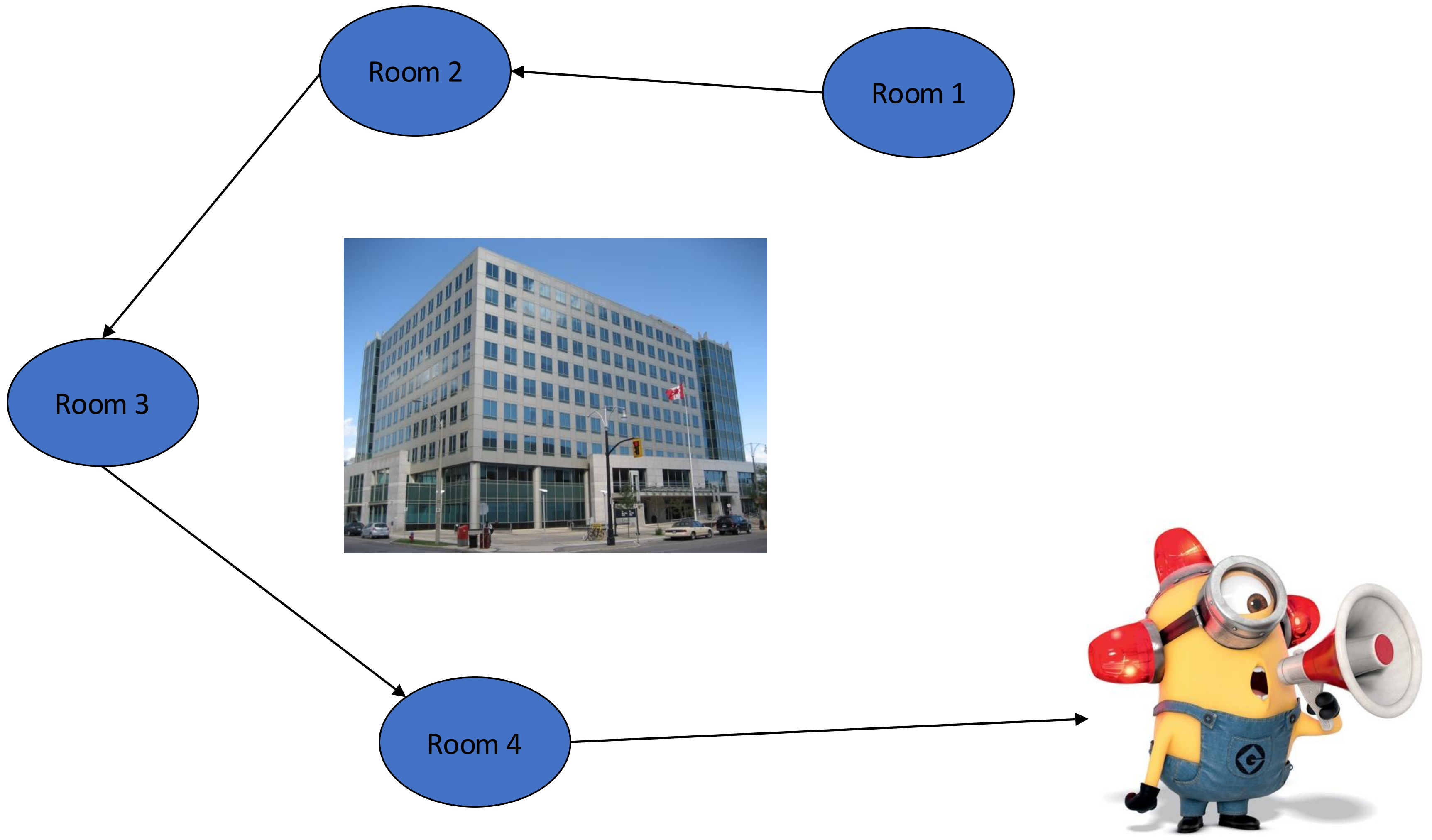
Interrupt



Polling

Interrupt

Polling



Polling

Periodically checks I/O status register

- device ready ⇒ perform operation
- error ⇒ take action

Common in small or low-performance real-time embedded systems:

- Predictable timing
- Low hardware cost

In other systems, wastes CPU time

Interrupts

When a device is ready or error occurs

- Controller interrupts CPU

Interrupt is like an exception

- But not synchronized to instruction execution
- Can invoke handler between instructions
- Cause information often identifies the interrupting device

Priority interrupts

- Devices needing more urgent attention get higher priority
- Can interrupt a handler that is dealing with a lower priority interrupt