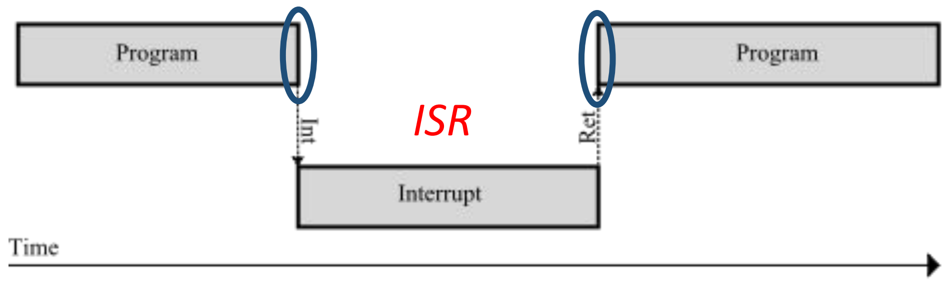
Interrupt

- Normal execution of a program may be preempted if some device requires urgent service.
- The device raises an *interrupt* signal, which is a request for service by the processor. The processor provides the requested service by executing a program called an *interrupt-service routine (ISR)*.
- Its state must be saved in the memory before servicing the interrupt request. The contents of the PC, the contents of the general-purpose registers, and some control information are saved,generally.
- When the ISR is completed, the state of the processor is restored.



slido

Join at slido.com
#736698

ⓘ Start presenting to display the joining instructions on this slide.

Microprocessors

Tuba Ayhan

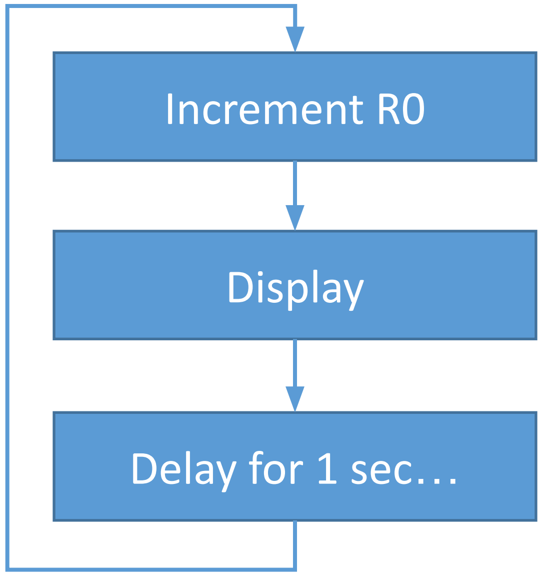
MEF University

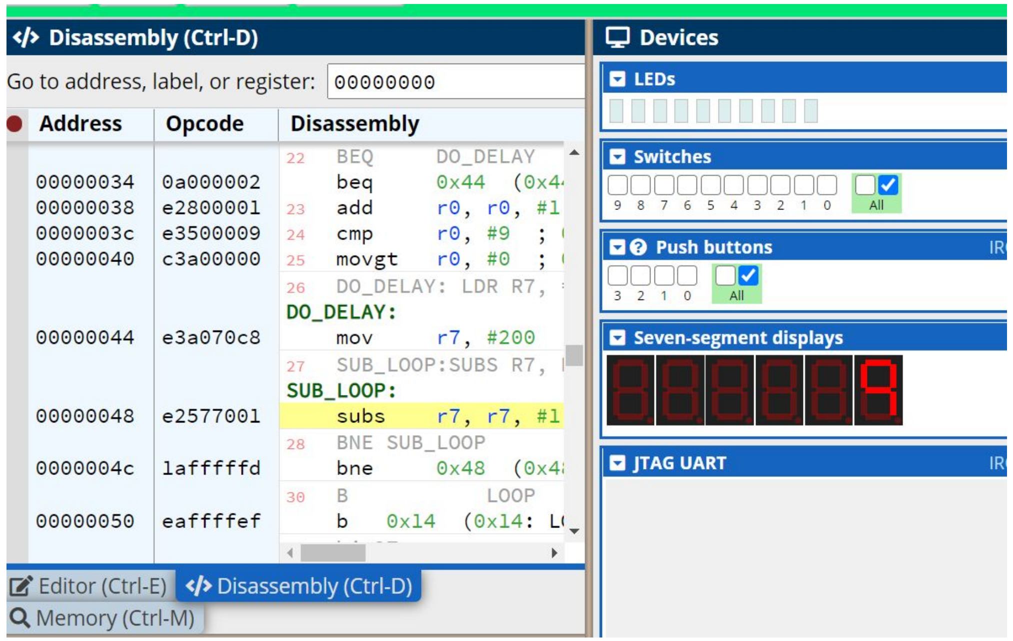
Interrupts

Computer Organization and Embedded Systems, Hamacher et. al

NOTE: Following section contains general interrupt concept.

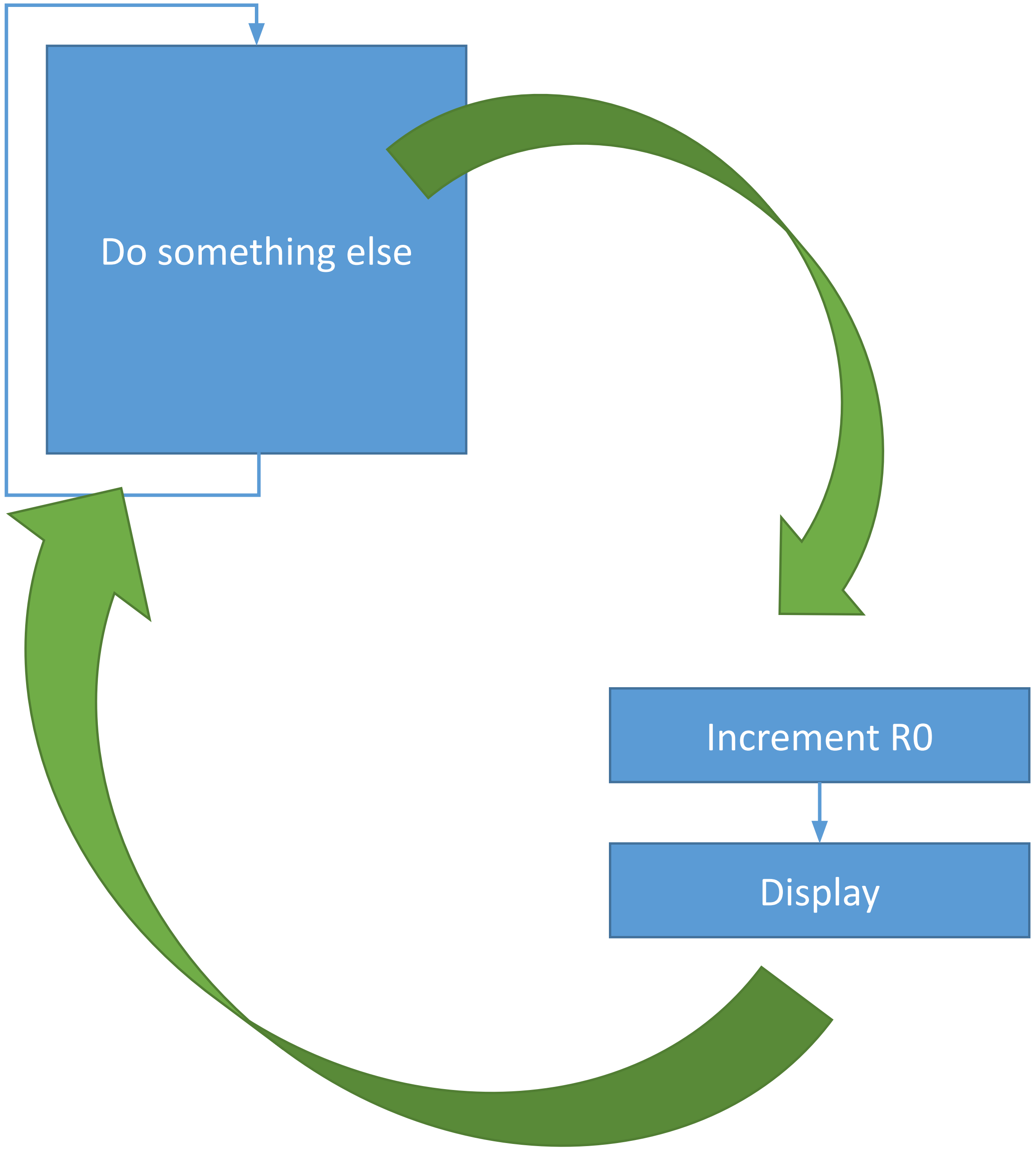
Interrupt handling in ARM will be covered in the next section.

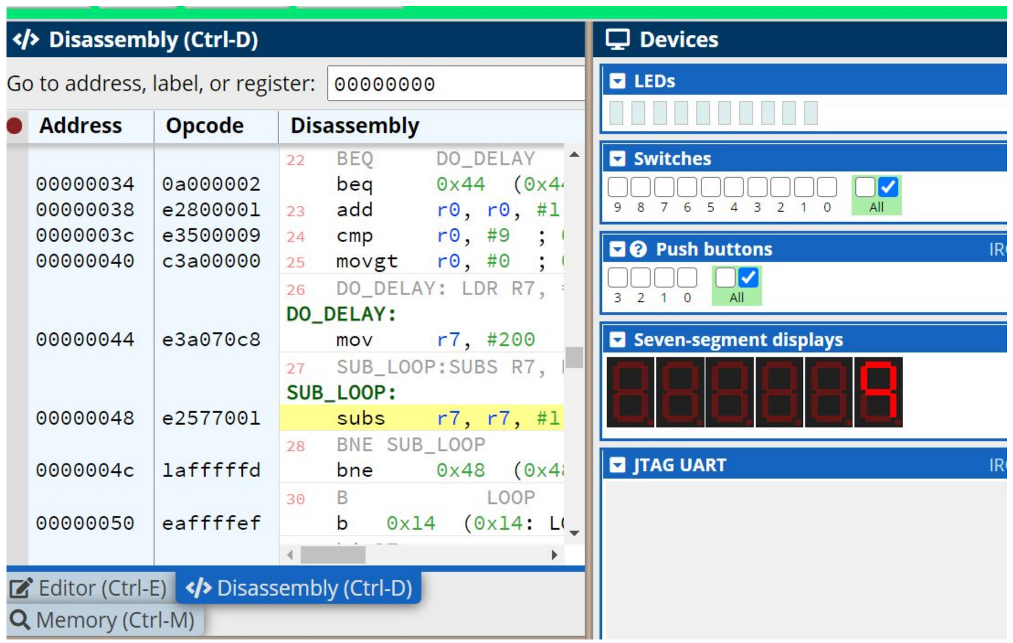




Go to address, label, or register: 00000000

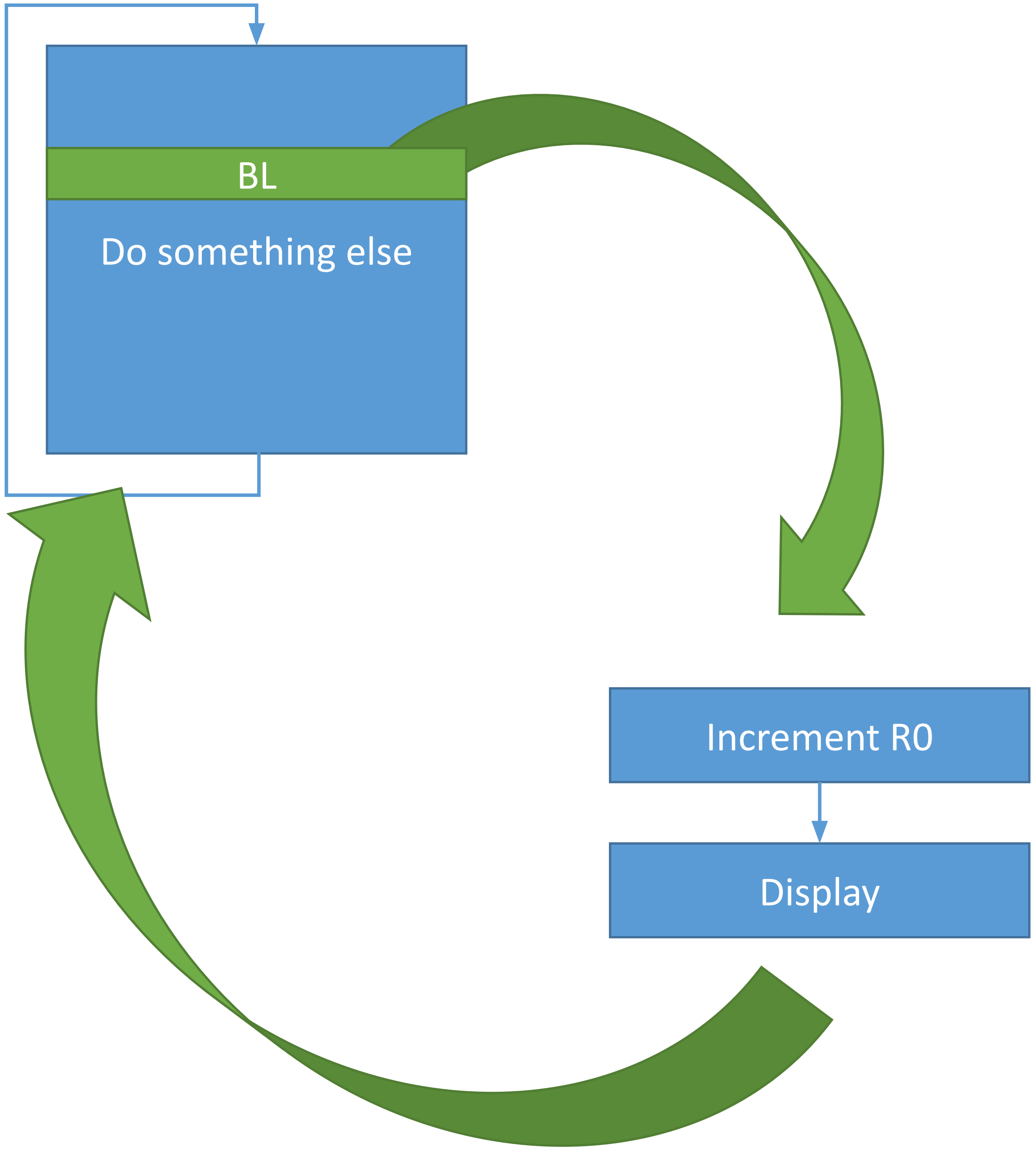
```
Address   Opcode    Disassembly
                       BEQ      DO_DELAY
00000034  0a000002     beq      0x44    (0x4
00000038  e2800001     add      r0, r0, #1
0000003c  e3500009     cmp      r0, #9   ;
00000040  c3a00000     movgt    r0, #0   ;
                       DO_DELAY: LDR R7,
                    DO_DELAY:
00000044  e3a070c8     mov      r7, #200
                       SUB_LOOP:SUBS R7,
                    SUB_LOOP:
00000048  e2577001     subs     r7, r7, #1
                       BNE  SUB_LOOP
0000004c  1afffffd     bne      0x48    (0x4
                       B            LOOP
00000050  eaffffef     b    0x14   (0x14: L
```

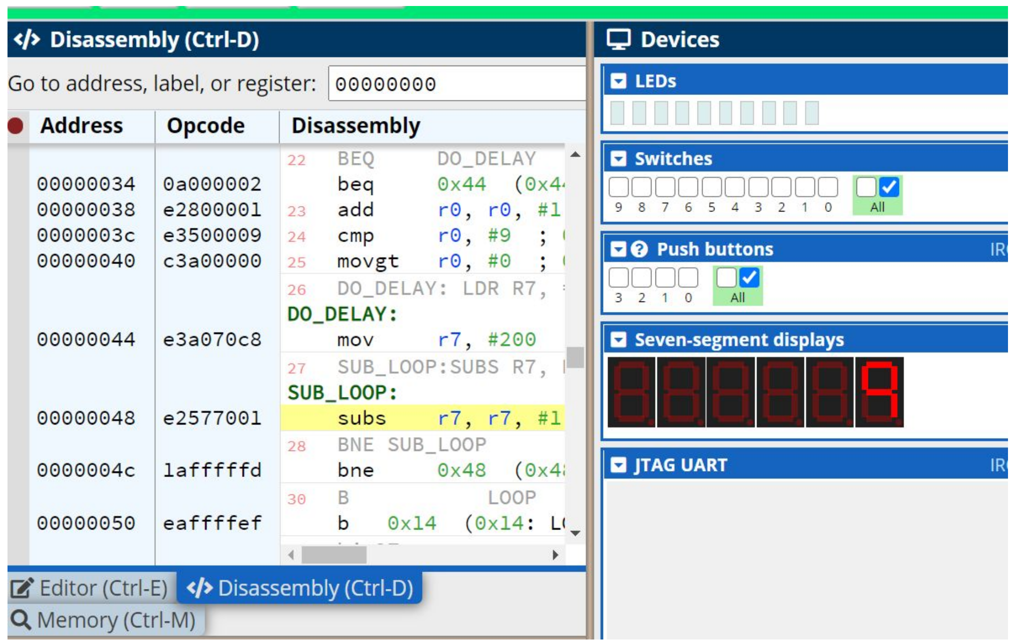




Go to address, label, or register: 00000000

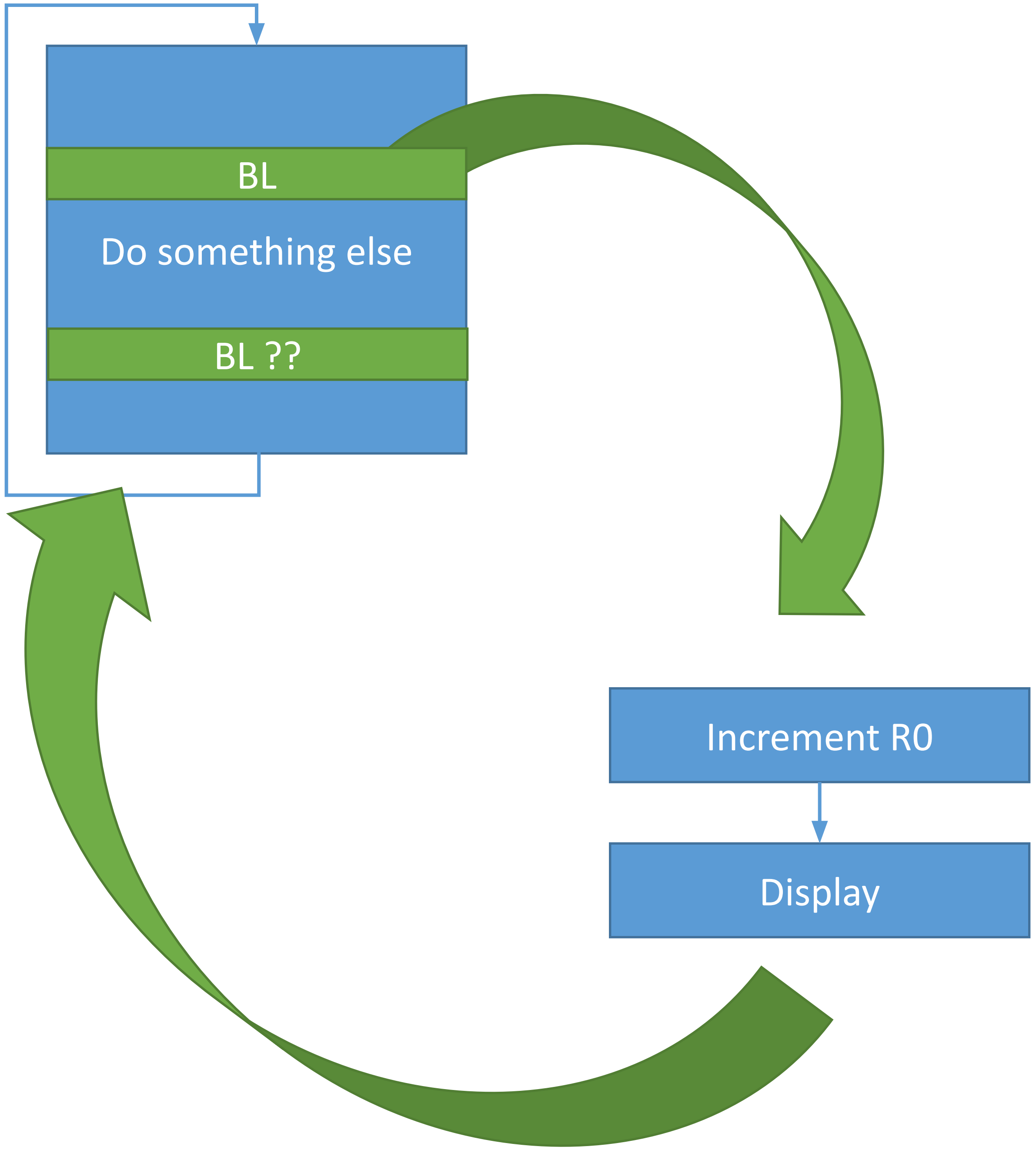
```
Address   Opcode    Disassembly
                       BEQ      DO_DELAY
00000034  0a000002     beq      0x44  (0x4
00000038  e2800001     add      r0, r0, #1
0000003c  e3500009     cmp      r0, #9   ;
00000040  c3a00000     movgt    r0, #0   ;
                       DO_DELAY: LDR R7,
                    DO_DELAY:
00000044  e3a070c8     mov      r7, #200
                       SUB_LOOP:SUBS R7,
                    SUB_LOOP:
00000048  e2577001     subs     r7, r7, #1
                       BNE SUB_LOOP
0000004c  1affffd      bne      0x48  (0x4
                       B           LOOP
00000050  eaffffef     b   0x14  (0x14: L
```

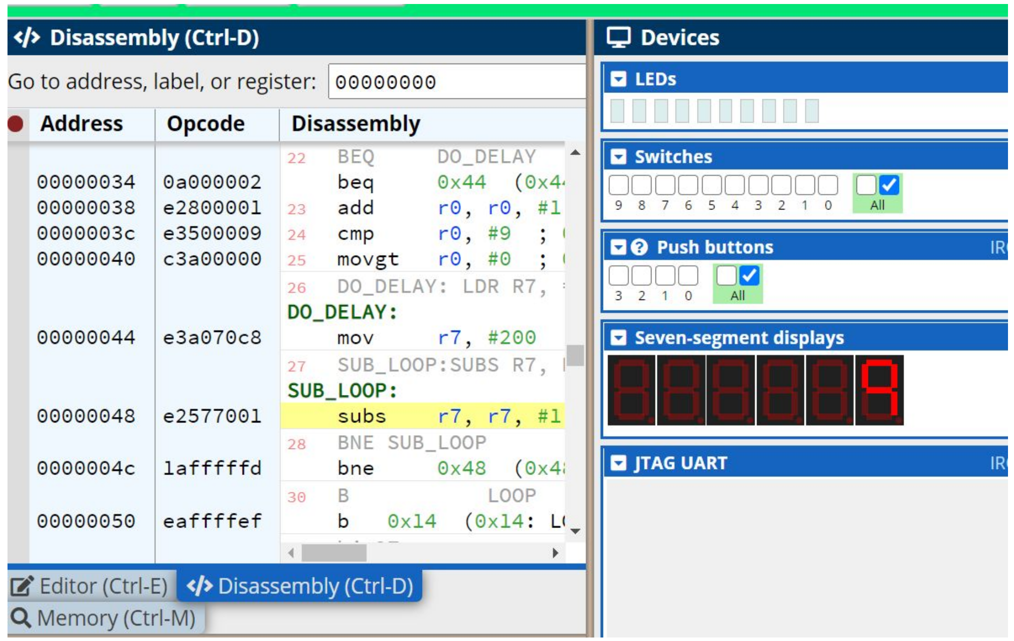




Go to address, label, or register: 00000000

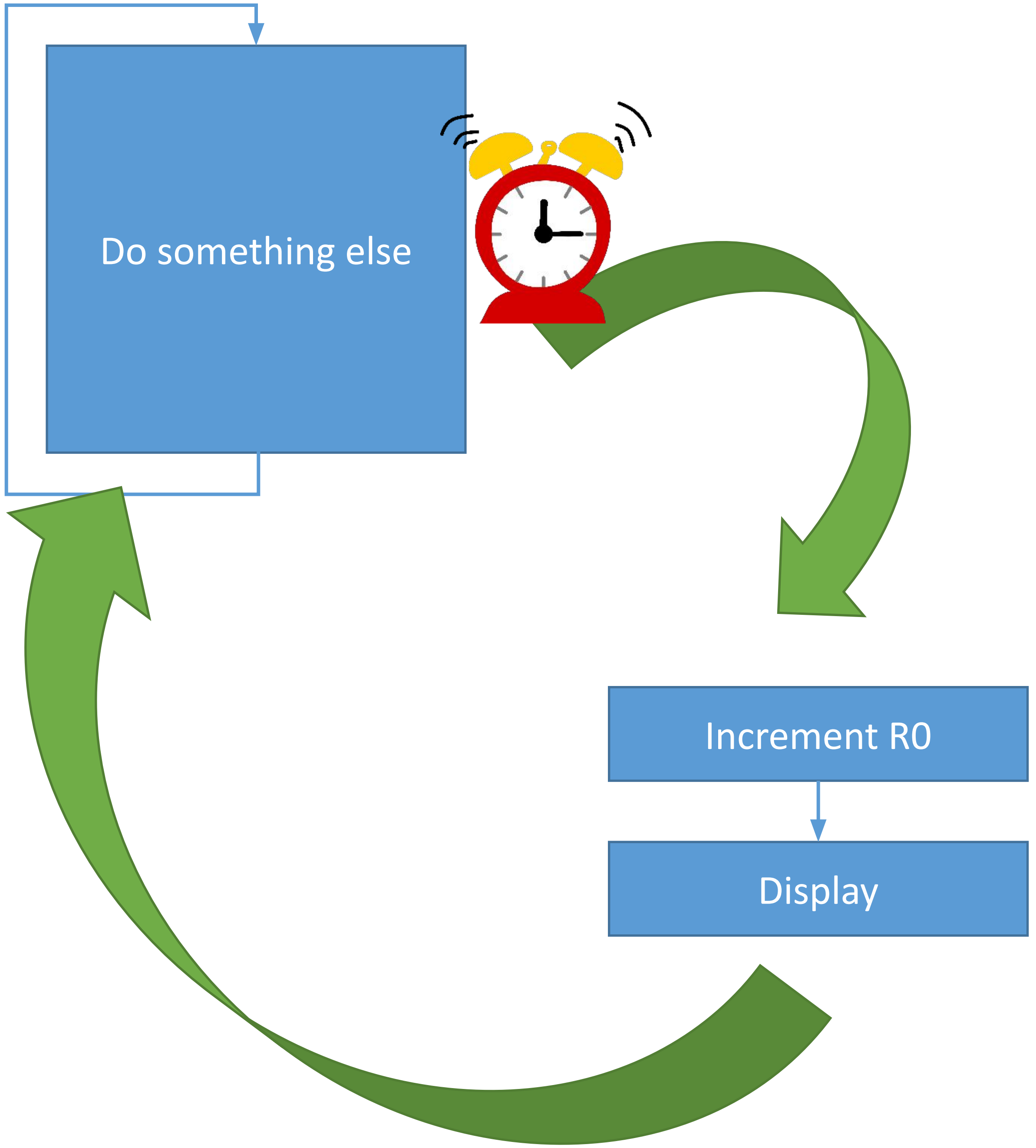
```
Address   Opcode    Disassembly
                    22   BEQ       DO_DELAY
00000034  0a000002       beq       0x44   (0x4
00000038  e2800001  23   add       r0, r0, #1
0000003c  e3500009  24   cmp       r0, #9   ;
00000040  c3a00000  25   movgt     r0, #0   ;
                    26   DO_DELAY: LDR R7,
                    DO_DELAY:
00000044  e3a070c8       mov       r7, #200
                    27   SUB_LOOP:SUBS R7,
                    SUB_LOOP:
00000048  e2577001       subs      r7, r7, #1
                    28   BNE  SUB_LOOP
0000004c  1afffffd       bne       0x48   (0x4
                    30   B              LOOP
00000050  eaffffef       b    0x14   (0x14: L
```

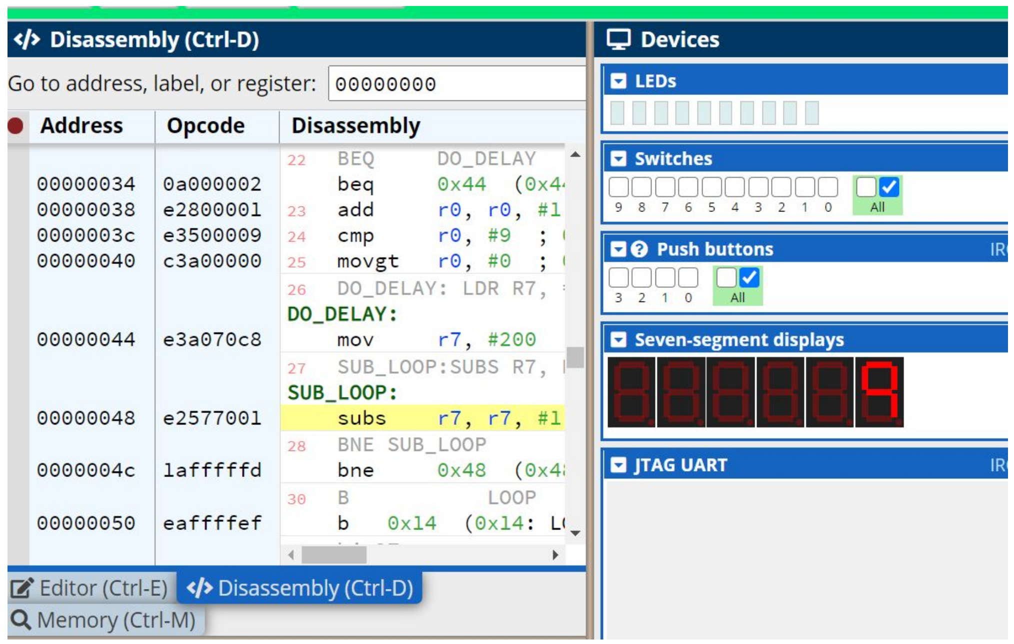




Go to address, label, or register: 00000000

```
Address   Opcode    Disassembly
                    BEQ       DO_DELAY
00000034  0a000002  beq       0x44  (0x4
00000038  e2800001  add       r0, r0, #1
0000003c  e3500009  cmp       r0, #9  ; (
00000040  c3a00000  movgt     r0, #0  ; (
                    DO_DELAY: LDR R7,
                  DO_DELAY:
00000044  e3a070c8  mov       r7, #200
                    SUB_LOOP:SUBS R7,
                  SUB_LOOP:
00000048  e2577001  subs      r7, r7, #1
                    BNE  SUB_LOOP
0000004c  1afffffd  bne       0x48  (0x4
                    B              LOOP
00000050  eaffffef  b    0x14  (0x14: L
```

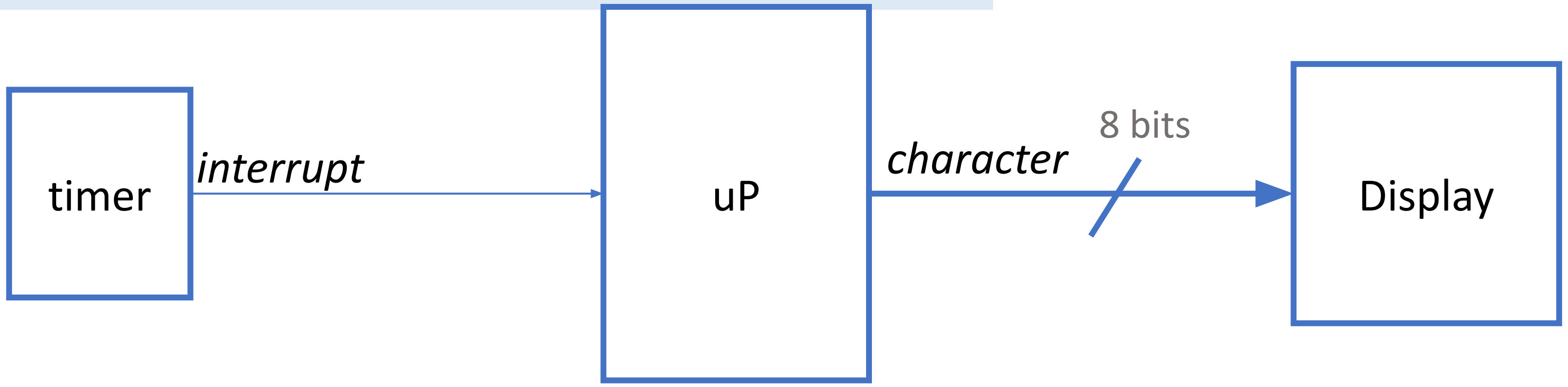


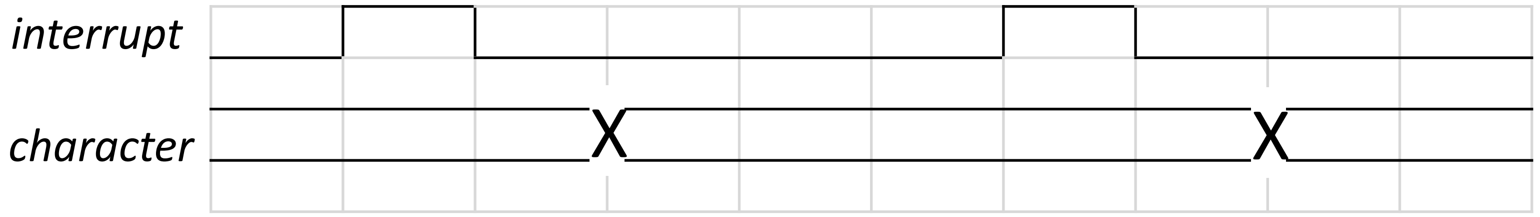


Go to address, label, or register: 00000000

```
Address   Opcode    Disassembly
                        BEQ       DO_DELAY
00000034  0a000002      beq       0x44   (0x4
00000038  e2800001      add       r0, r0, #1
0000003c  e3500009      cmp       r0, #9   ;
00000040  c3a00000      movgt     r0, #0   ;
                        DO_DELAY: LDR R7,
                    DO_DELAY:
00000044  e3a070c8      mov       r7, #200
                        SUB_LOOP:SUBS R7,
                    SUB_LOOP:
00000048  e2577001      subs      r7, r7, #1
                        BNE  SUB_LOOP
0000004c  1afffffd      bne       0x48   (0x4
                        B              LOOP
00000050  eaffffef      b    0x14   (0x14: L
```

Interrupts

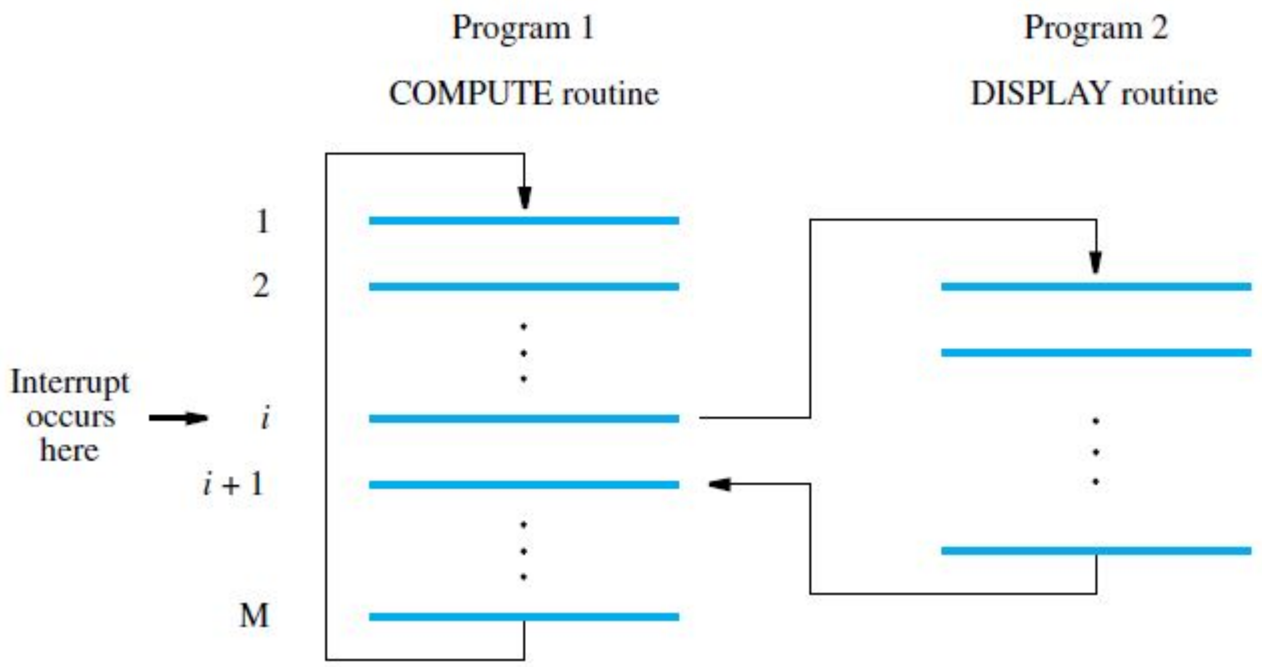




- Consider a task that requires continuous extensive computations to be performed and the results to be displayed on a display device, every ten seconds.
- The ten-second intervals is determined by a timer circuit.
- The timer circuit raise an interrupt request once every ten seconds.
- In response, the processor displays the latest results.

Interrupts

- The processor continuously executes the COMPUTE routine.
- DISPLAY routine sends the latest results to the display device.
- When uP receives an interrupt request from the timer, it suspends the execution of the COMPUTE routine and executes the DISPLAY routine.
- Upon completion of the DISPLAY routine, the processor resumes the execution of the COMPUTE routine.
- A **Return-from-interrupt** instruction at the end of the interrupt-service routine reloads the PC from that temporary storage location. (The return address must be saved.)



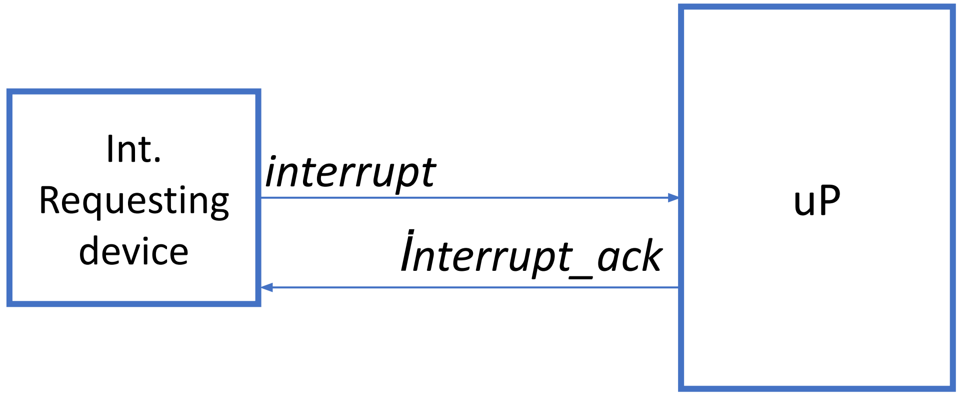
slido

What may be the potential problems with the interrupts?

ⓘ Start presenting to display the poll results on this slide.

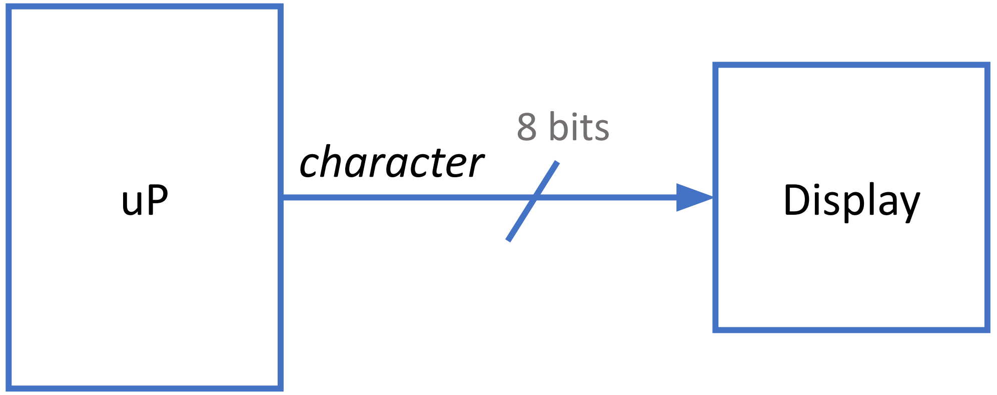
Interrupts

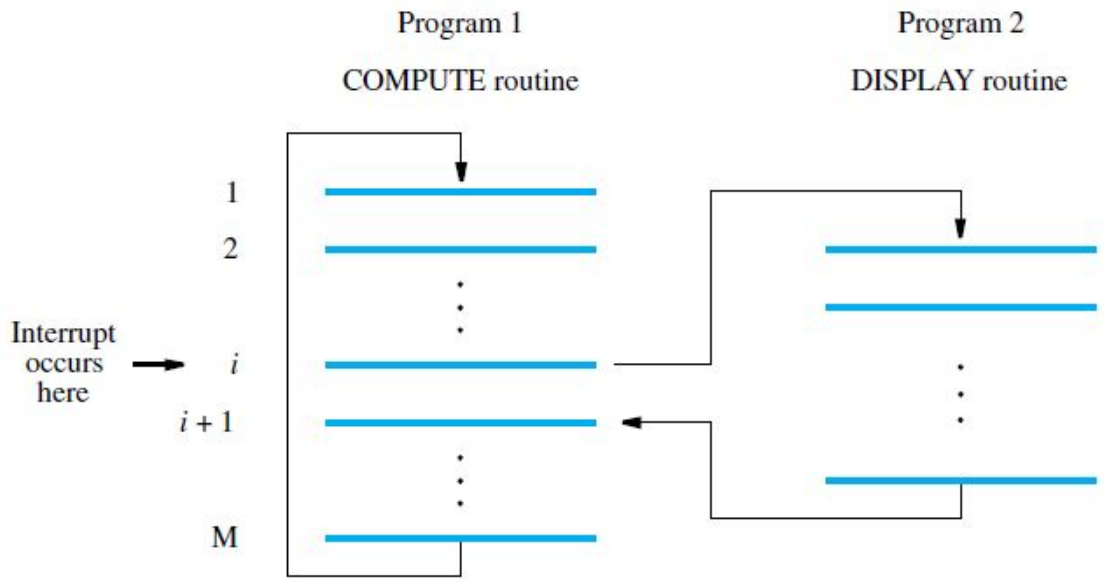
- **interrupt acknowledge signal**:
- The processor must inform the device that its request has been recognized so that it may remove its interrupt-request signal.
- It is sent through the interconnection network.



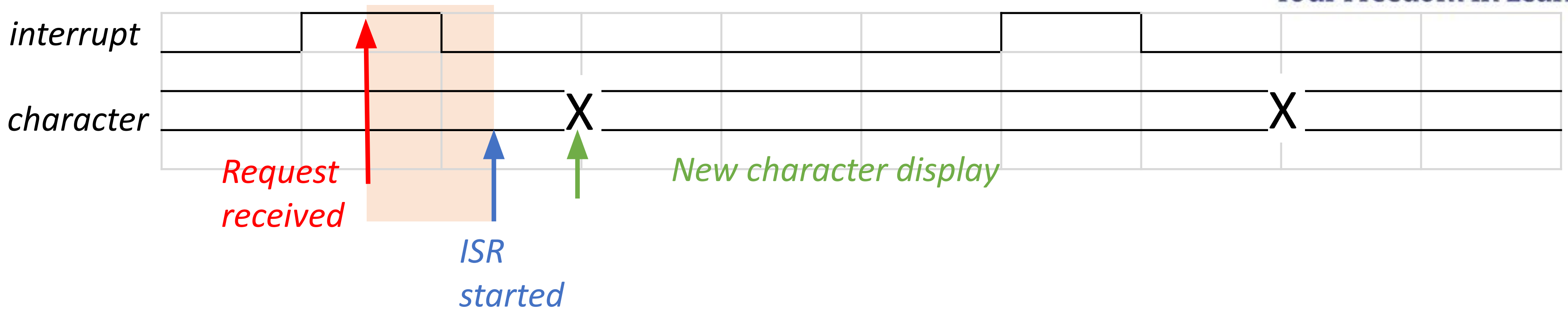
Interrupts

- **interrupt service routine (ISR)**:
- Display routine in this example.
- It resembles a subroutine performing DISPLAY.
- ISR may not have any relation to the portion of the program being executed at the time the interrupt request is received.
- Before starting execution of the ISR status information and contents of processor registers must be saved.





Interrupt latency



- is the delay between the time an interrupt request is received and the start of execution of the ISR.
- Most modern processors **save only the minimum amount of information** needed to maintain the integrity of program execution. Because:
- Saving and restoring registers involves memory transfers the total execution time increases. This represents **execution overhead**.

Enabling and Disabling Interrupts

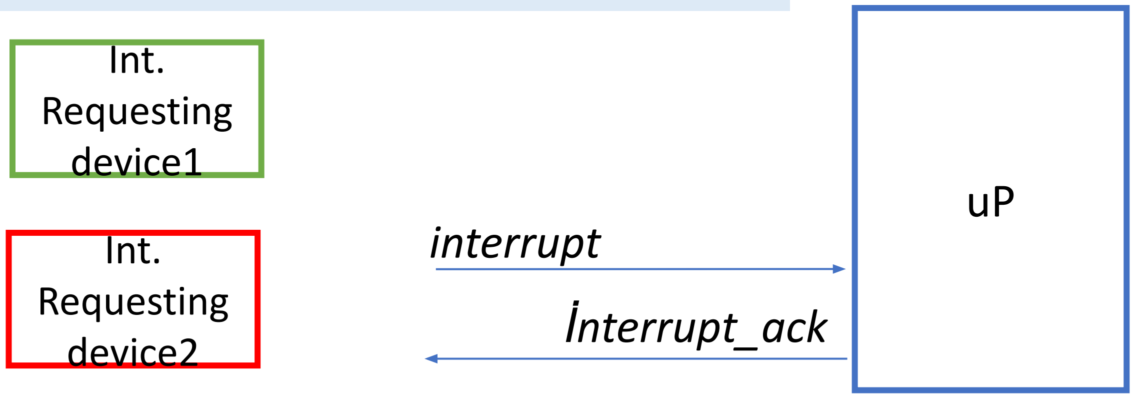
- One bit of the register be assigned for enabling/disabling interrupts.
  - Usually IE or ID
  - The programmer can set or clear IE/ID to cause the desired action.
  - The processor (usually) automatically disable interrupts before starting the execution of the interrupt-service routine

Handling an interrupt request

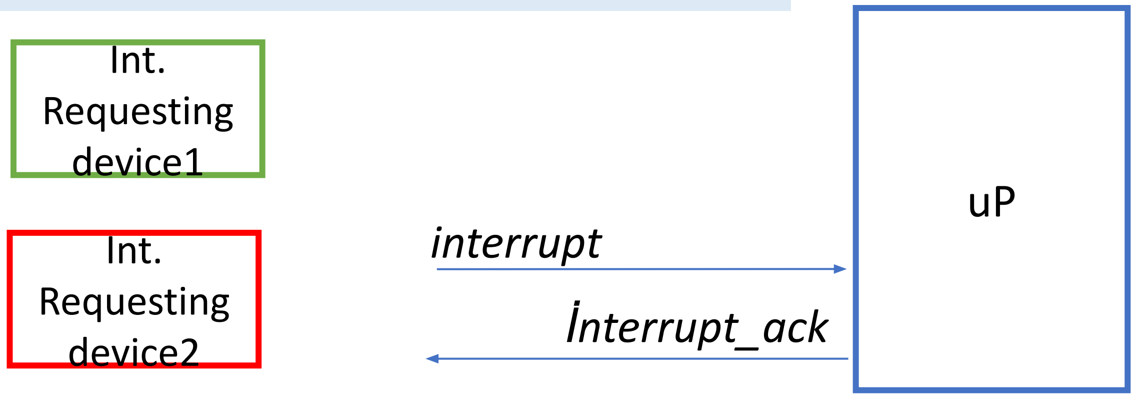
1. The device raises an interrupt request.
2. The processor interrupts the program currently being executed and saves the contents of the PC and PS registers.
3. *Interrupts are disabled by clearing/setting the IE/ID bit in the status register*.
4. The device is informed that its request has been recognized, and in response, it deactivates the interrupt-request signal. The action requested by the interrupt is performed by the ISR.
5. Upon completion of the interrupt-service routine, the saved contents of the PC and status registers are restored.

Interrupts may or may not be enabled.

Handling Multiple Devices



Handling Multiple Devices



1. How can the processor determine which device is requesting an interrupt?

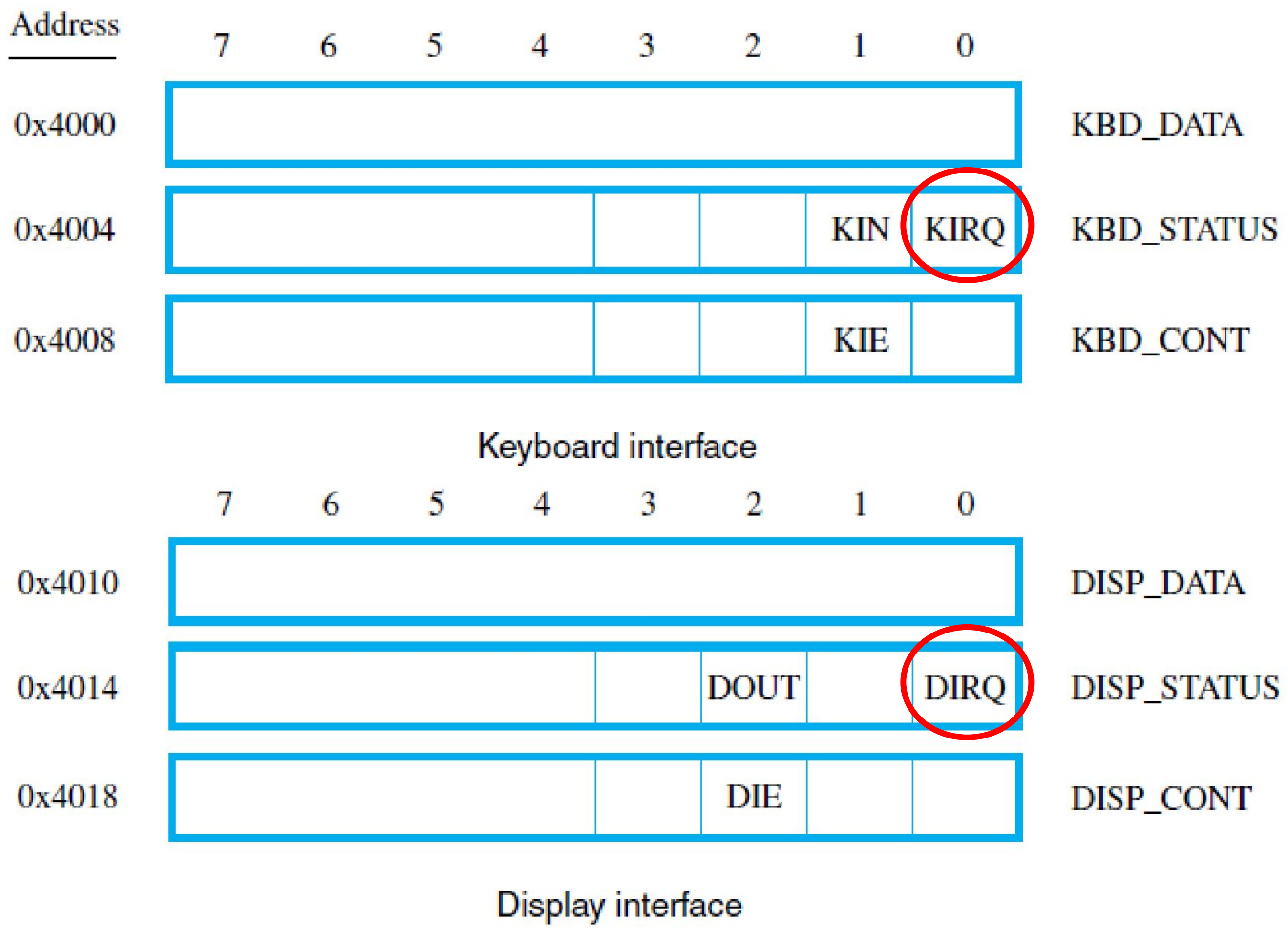
2. If they require different ISRs, how can the processor obtain the correct ISR address?

3. Should a device be allowed to interrupt the processor while another interrupt is being serviced?

4. How should two or more simultaneous interrupt requests be handled?

Handling Multiple Devices

- A device's Interrupt ReQuest (IRQ) is available in its status register.
- Identify the device requesting the interrupt:
- The ISR polls all I/O devices in the system.
  - ☺️ The simplest way
  - ☹️ Time spent interrogating the IRQ bits of devices that may not be requesting any service

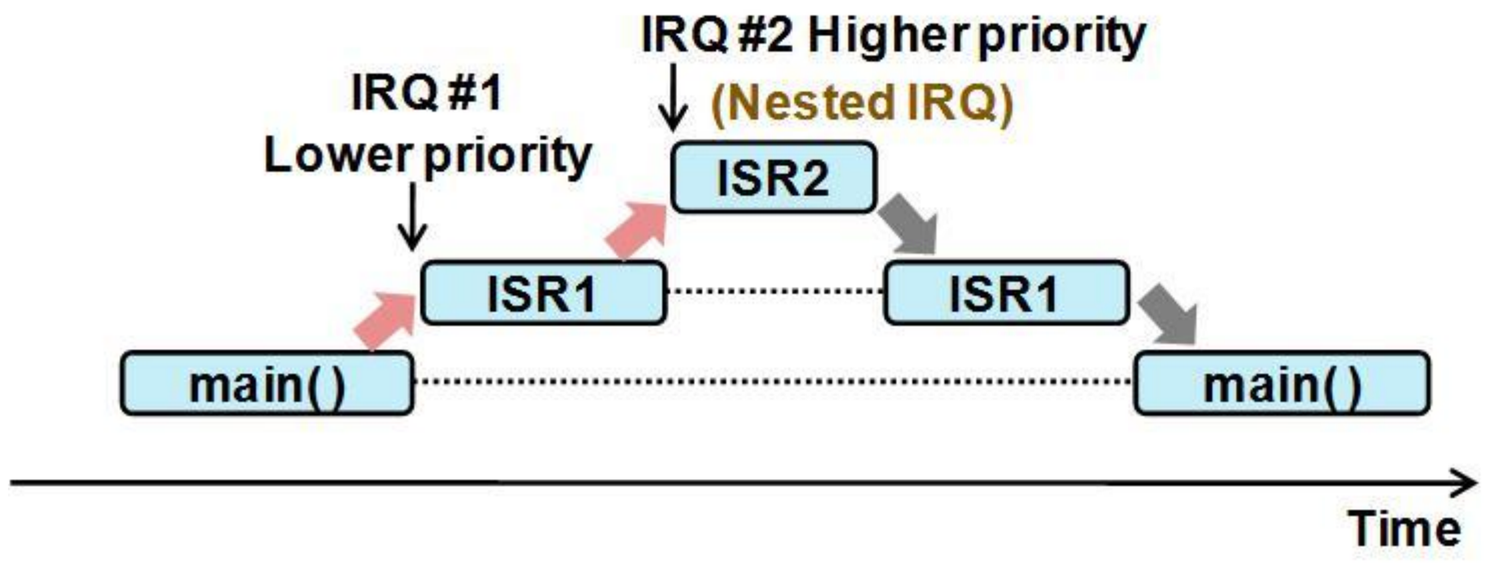


Vectored Interrupts

- A device requesting an interrupt may identify itself directly to the processor:
  - it has its own interrupt-request signal
  - it sends a special code to the processor through the interconnection network
- The processor's circuits determine the memory address of the required interrupt-service routine.
- Typically, **an area in the memory** holds the addresses of interrupt-service routines: **interrupt vectors**.
- The **interrupt vector table** is –usually, in the lowest-address range. The interrupt-service routines may be located anywhere in the memory.

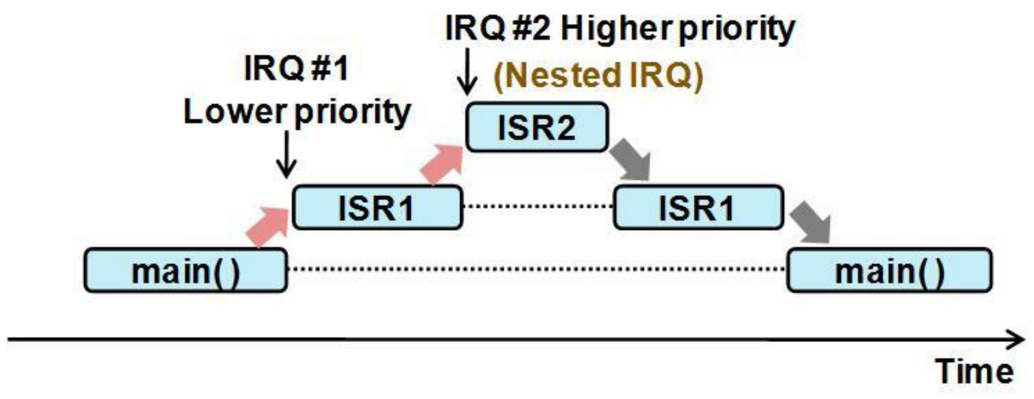
0x00	B ISR1
0x04	B ISR2
0xFA0	ISR1 instructions
0xFF0	ISR1 instructions

Interrupt Nesting



Interrupt Nesting

- I/O devices should be organized in a priority structure: **multiple-level priority organization**
- An interrupt request from a high-priority device should be accepted while the processor is servicing a request from a lower-priority device
- Assign a priority level to the processor that can be changed under program control



- During execution of an ISR,
  - The priority of the processor is raised to that of the device
  - Interrupts from devices that have the same or lower level of priority are disabled
  - Interrupt requests from higher-priority devices will continue to be accepted.

Simultaneous Requests

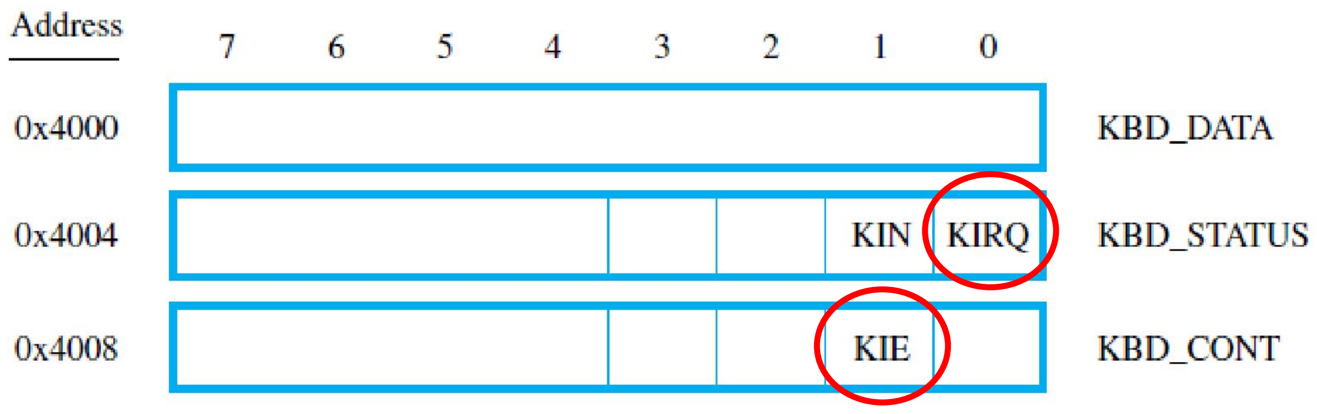
- Problem: Deciding which request to service first
- Solution 1: Polling the status registers of the I/O devices
  - Priority is determined by the order in which the devices are polled
- Solution 2: Use vectored interrupts
  - Ensure that only one device is selected to send its interrupt vector code
  - Use an appropriate interface circuit

Simultaneous Requests

- Problem: Deciding which request to service first

Interrupt controlled I/O

- Device is allowed to interrupt the processor whenever it is ready for an I/O transfer
- Control register
  - It can be accessed as an addressable location
  - It involves the interrupt-enable bit, IE
  - Simple devices - a keyboard, require little control
  - Complex devices - have a number of possible modes of operation



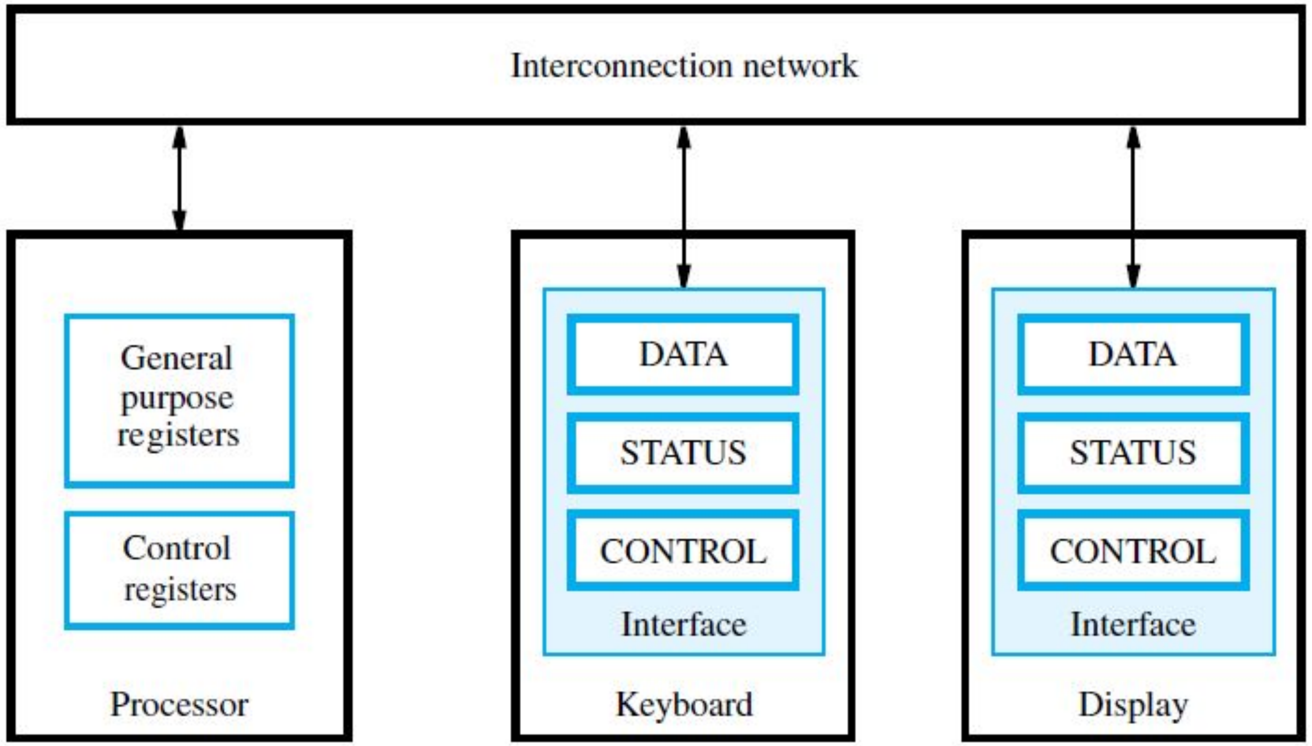
Keyboard interface

- Example: The keyboard status register includes bits KIN and KIRQ. Control register involves KIE: Keyboard interrupt enable.
  - If, KIE = 1
  - KIRQ □1, when an interrupt request has been raised, but not yet serviced.

I/O Device Interface

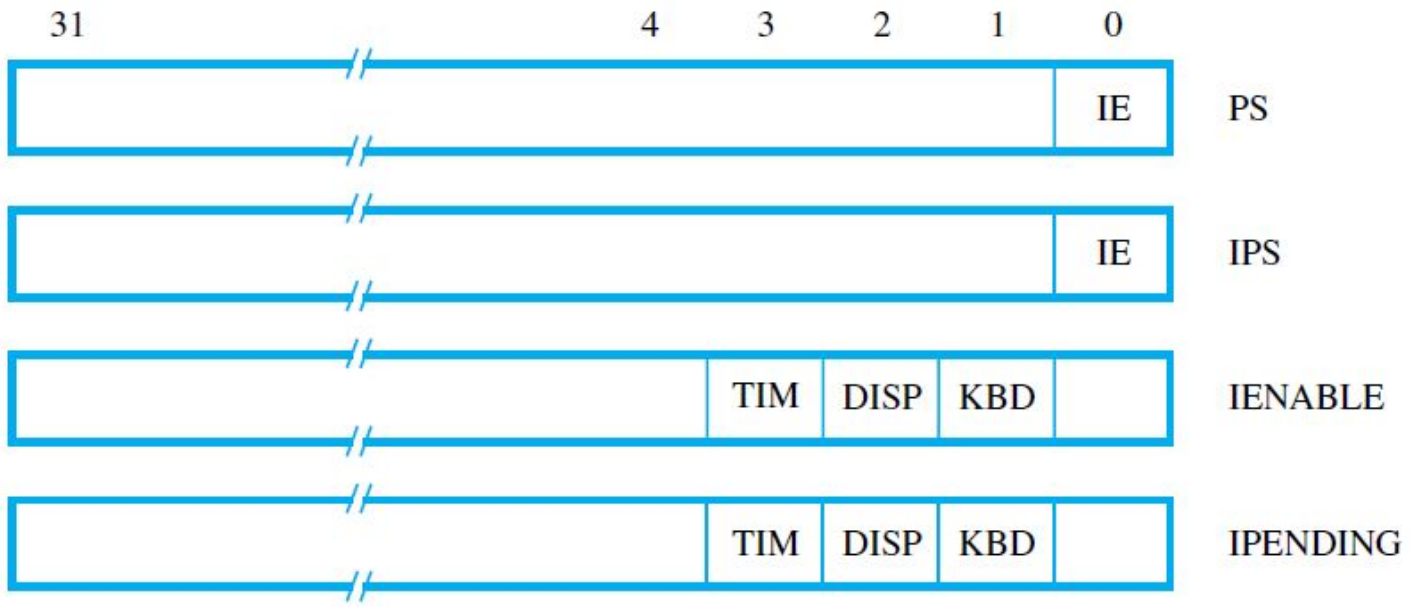
governs the operation of the device.

- Data registers: a buffer for data transfers
- Status registers: hold info about the current status of the device
- Control registers: hold info that controls the operational behavior of the device



Processor Control Registers

- PS is status register.
- IPS register is used to automatically save the contents of PS when an interrupt request is received and accepted



- IENABLE register allows the processor to selectively respond to individual I/O devices.
- IPENDING register indicates the active interrupt requests This is convenient when multiple devices may raise requests at the same time.