ARCOS Group

uc3m | Universidad Carlos III de Madrid

Lesson 4 (III)
The processor

Computer Structure

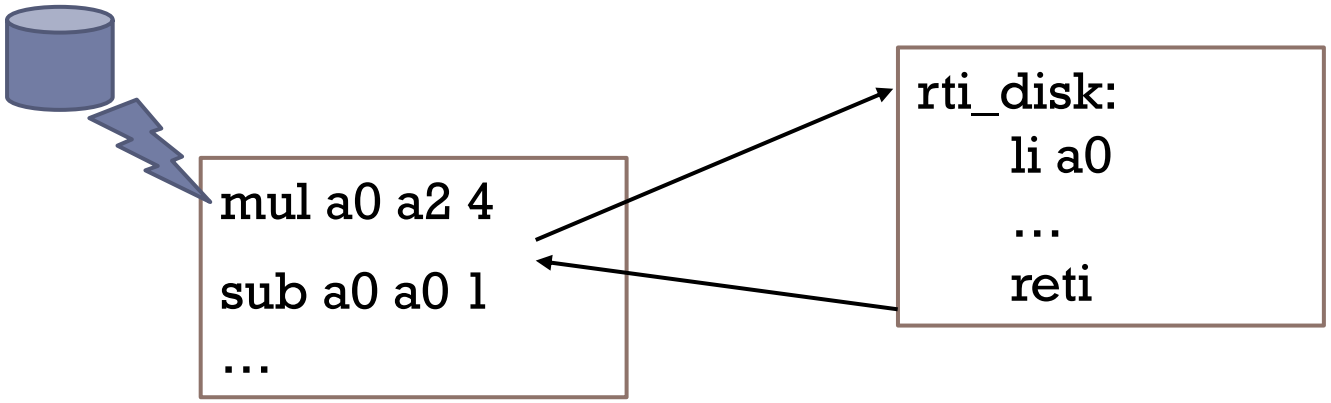
Bachelor in Computer Science and Engineering

Contents

Modes of execution

- It is indicated by a bit in the status register (U)
- At least 2 modes:
  - User Mode
    - The processor cannot execute privileged instructions (e.g.: I/O instructions, interrupt enable instructions, ...)
    - If a user process executes a privileged instruction, an interruption (exception) occurs
  - Kernel Mode
    - Reserved to the operating system
    - The processor can execute the entire repertoire of instructions

Interrupts

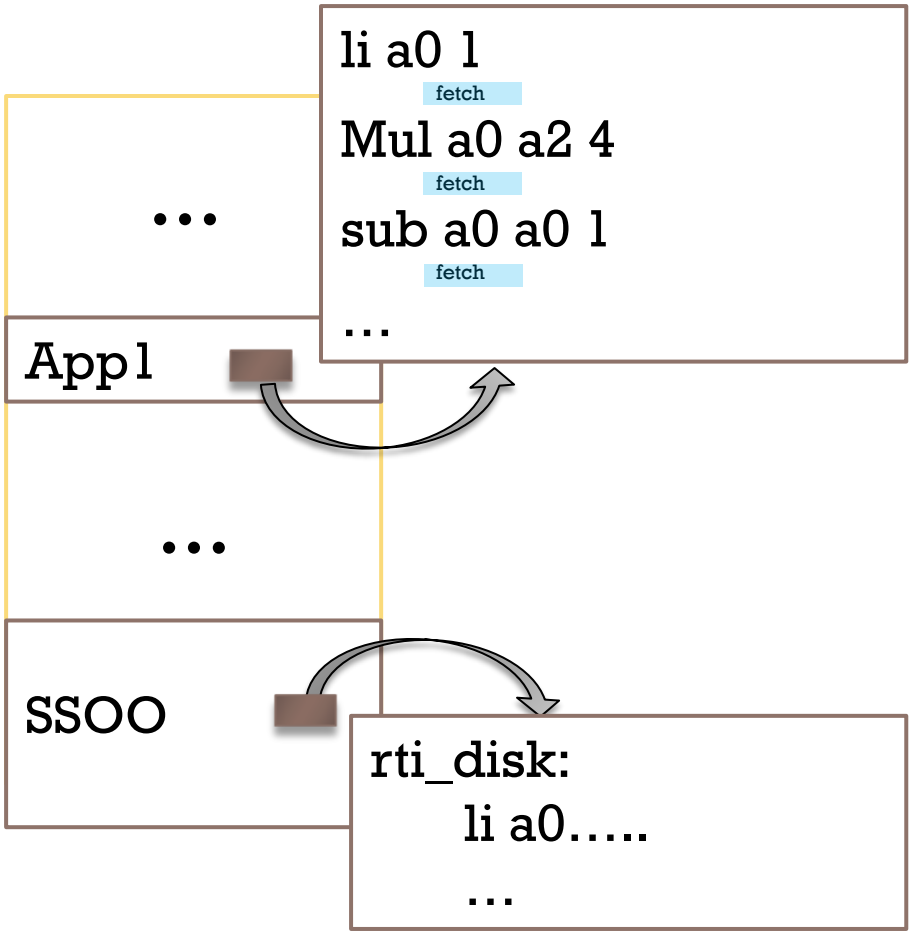


- Signal that arrives to the control unit (CU) and breaks the normal execution sequence:
  - The current program is stopped and the execution is transferred to another program that attend the interruption (ISR)
  - When the ISR ends, the execution of the interrupted program is resumed.
- Example of causes:
  - When a peripheral requests the attention of the processor,
  - When an error occurs in the execution of the instruction,
  - Etc.

Classification of interruptions

- Synchronous hardware exceptions
  - When an error occurs in the execution of the instruction: Division by zero, access to an illegal memory position, illegal instruction, etc.
- Asynchronous hardware exceptions
  - Faults or errors in hardware not related to current instruction: printer without paper, power failure, etc.
- External interruptions
  - When a peripheral (or CPU) requests the attention of the CPU: Peripherals, clock interruption
- System calls
  - Request an operating system service
    - Special machine instructions that generate an interruption to activate the operating system

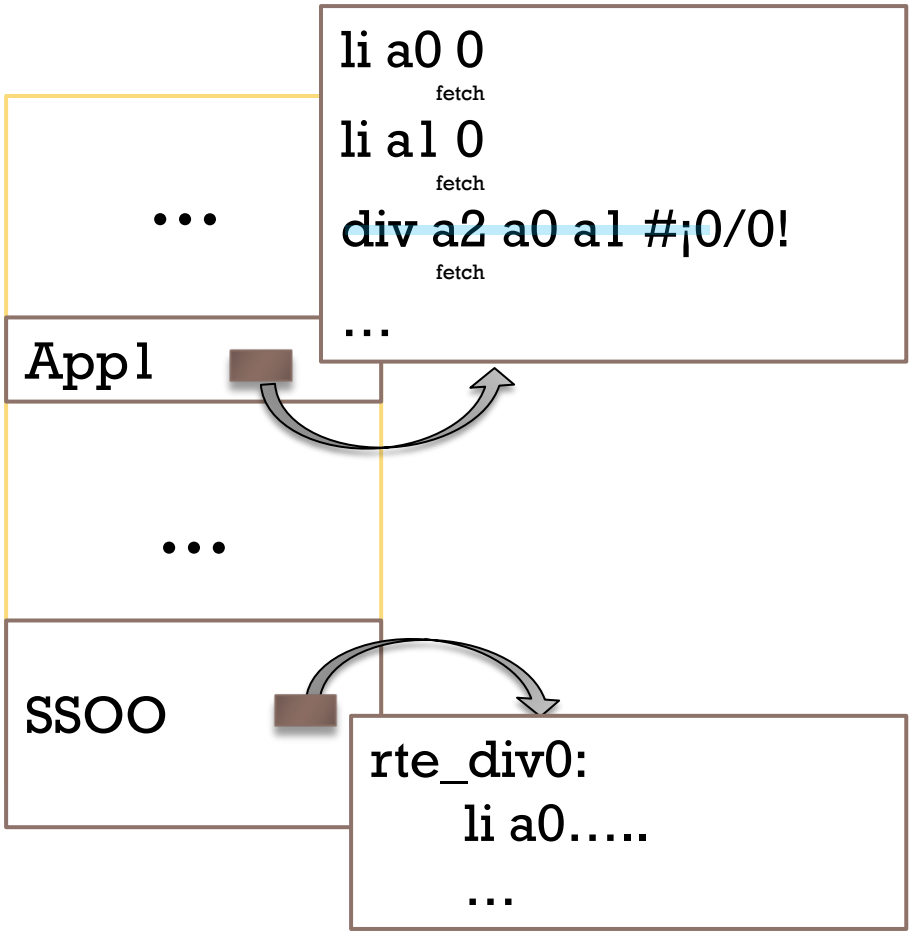
Asynchronous Hardware Exceptions and External Interrupts



- They cause an unscheduled sequence break
  - **Before doing the fetch cycle, first see if there is any pending interruption, and if so...**
  - ...Bifurcation to subroutine of the O.S. that treats it
- It then restores the status and returns control to the interrupted program.
- **Asynchronous cause to the execution of the current program**
  - Peripheral care
  - Etc.

Example in WepSIM: int, syscall, exception...

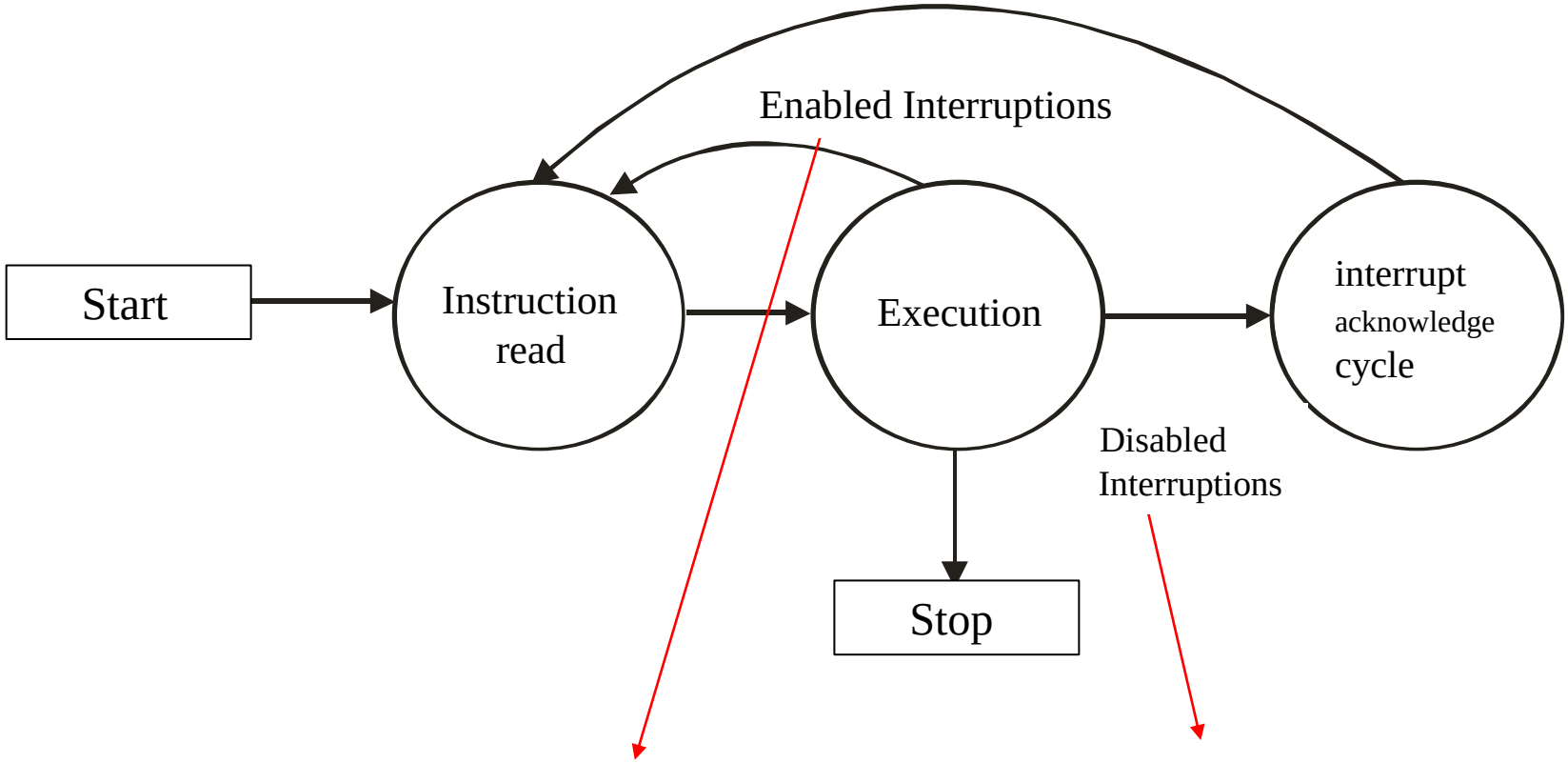
Synchronous hardware exceptions



- They cause an unscheduled sequence break
  - **Within the microprogram of the ongoing instruction…**
  - …Bifurcation to subroutine of the O.S. that treats it
- It restores the status and returns control to the interrupted program **or ends its execution**
- **Synchronous cause to the execution of the current program**
  - Division between zero
  - Etc.

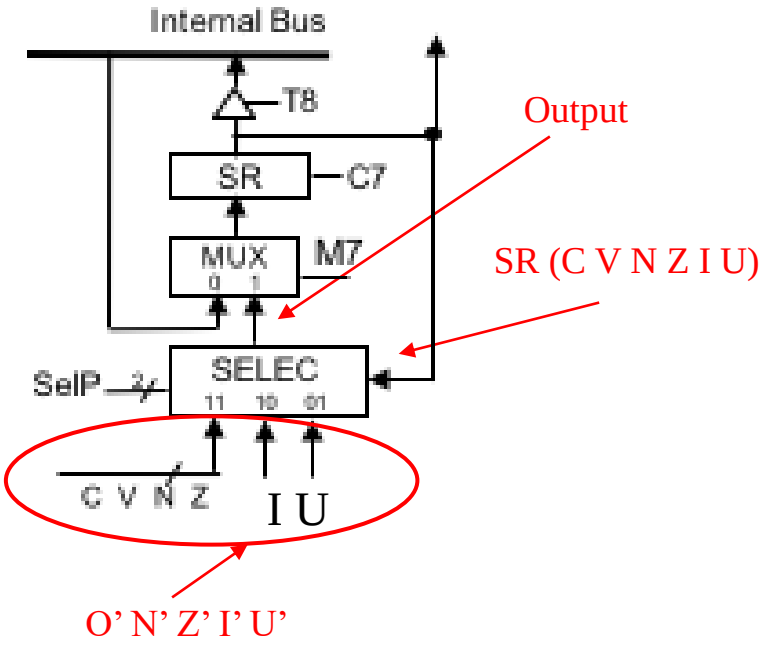
Example in WepSIM: int, syscall, exception…

Activation of the status register



It is indicated by a bit located in the status register (I)

Activation of the status register



SELEC operation:

if (SelP1 = 1 AND SelP0 == 1)
Output = C' V' N' Z' I U

if (SelP1 == 1 AND SelP0 ==0)
Output = C V N Z I' U

if (SelP1 == 0 AND SelP0 == 1)
Output = C V N Z I U'

Interrupt Acknowledge Cycle (IAC)

- It is a microcode before the fetch cycle
  - It handles the asynchronous interrupts
- General structure of the IAC:
  1. Checks if an interruption signal is activated.
  2. If it is activated:
     1. Saves the program counter and status register
        - Equivalent to "push pc, push sr"
     2. Switches from user mode to kernel mode
        - Equivalent to "SR.U = 0"
     3. Obtains the address of the Interrupt Service Routine (ISR)
        - Equivalent to "isr_addr = Vector_interrupts[id_interrupt]"
     4. Store the address obtained in the program counter (this way the following instruction will be the first one for the treatment routine)
        - Equivalent to "PC = isr_addr"

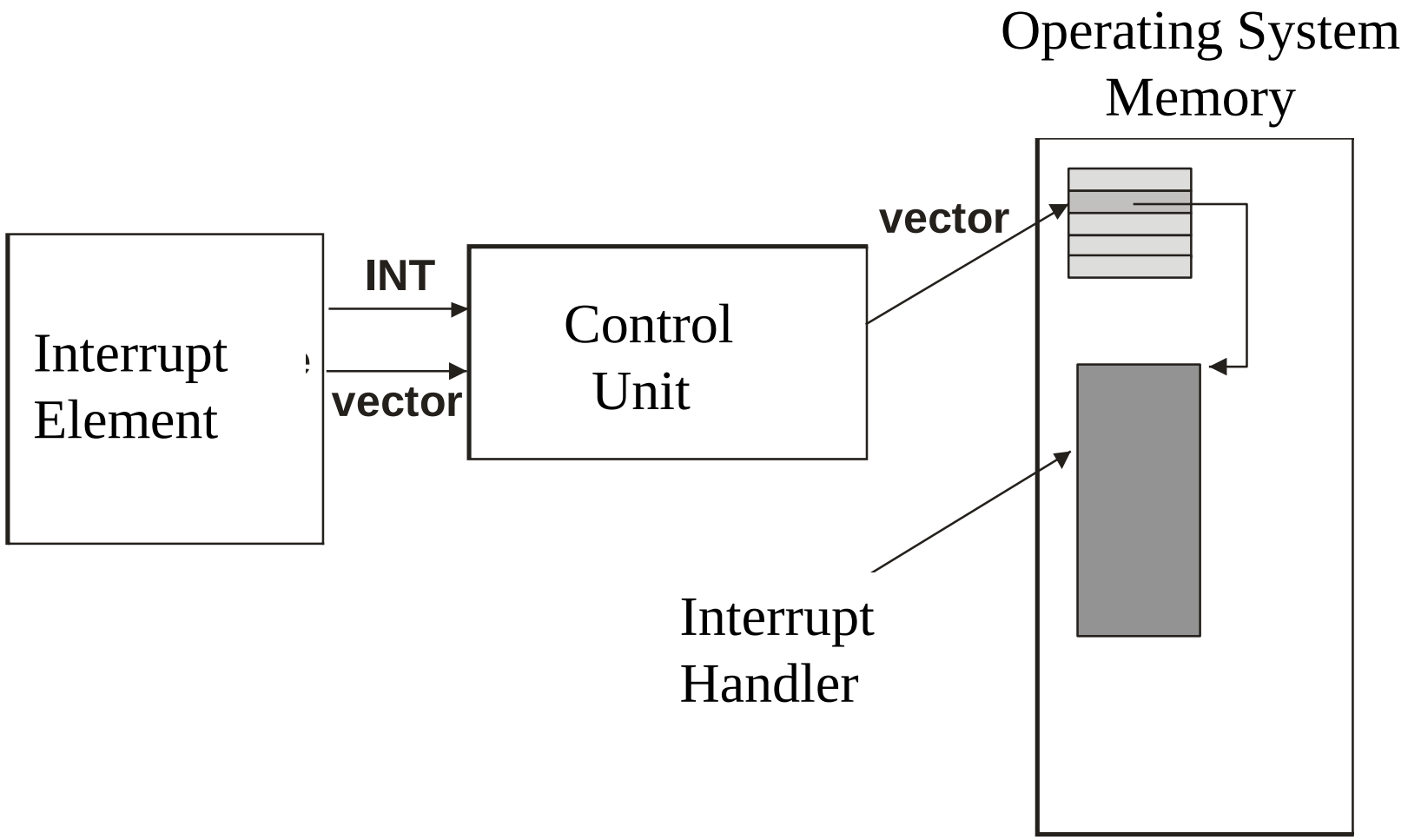
Example in WepSIM: int, syscall, exception…

Interrupt service routine (ISR)

- It is part of the operating system code
  - There is one ISR for each interruption that may occur
- General structure of the ISR:
  1. Saves the rest of the processor registers (if required)
  2. Service the interrupt
  3. Restores processor registers saved in (2)
  4. Executes a special machine instruction: RETI
     - Resets the status register of the interrupted program (by setting the processor mode back to user mode).
     - Resets the program counter (so that the next instruction is that of the interrupted program).

Example in WepSIM: int, syscall, exception…

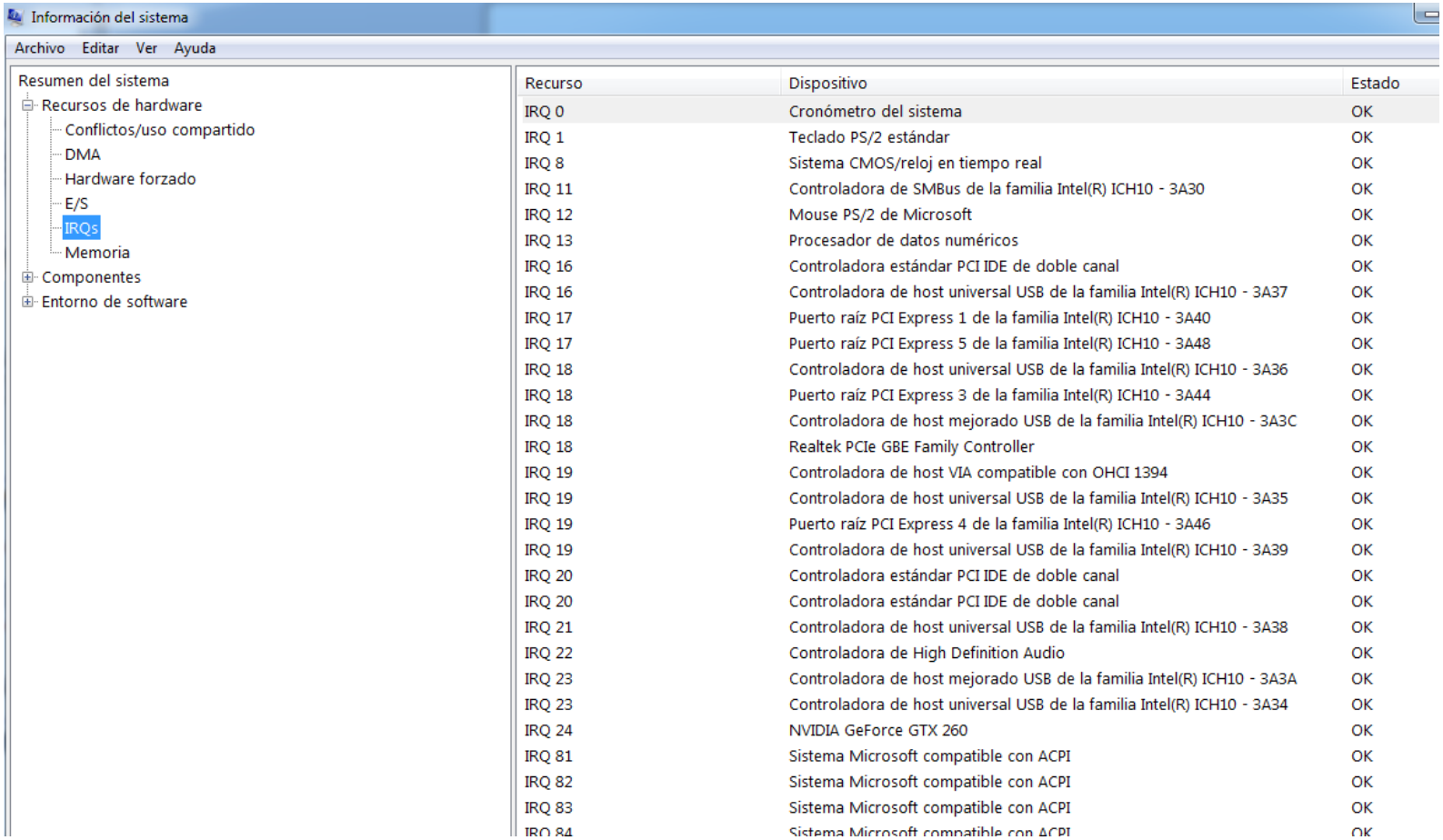
Vector interrupts



Vector interrupts

- The interrupting element supplies the interrupt vector
- This vector is an index in a table containing the address of the interrupt handdler routine.
- The Control Unit reads the content of this entry and loads the value into the PC
- Each operating system fills this table with the addresses of each of the treatment routines, which are dependent on each operating system.

Interrupts in Windows



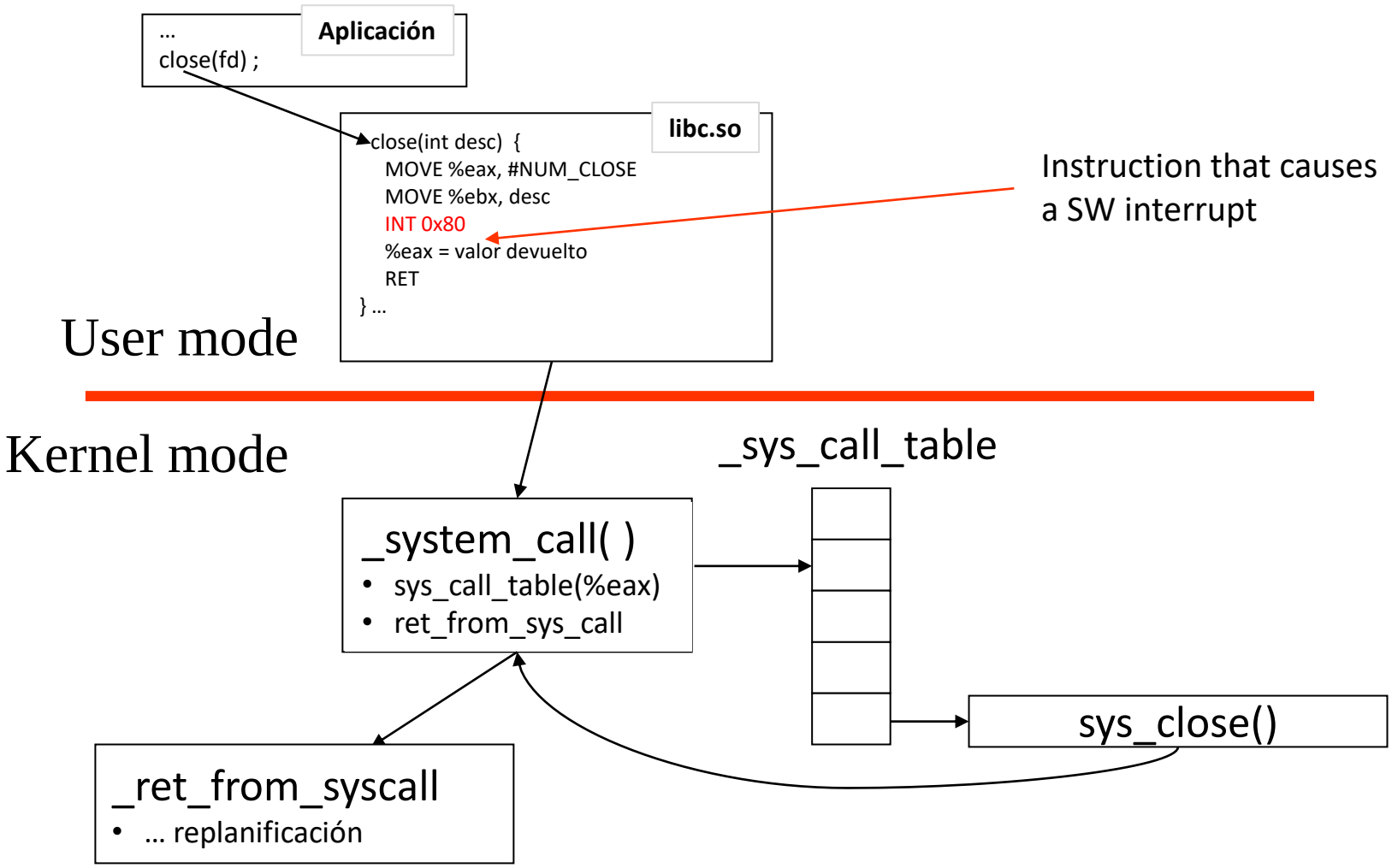
Recurso	Dispositivo	Estado
IRQ 0	Cronómetro del sistema	OK
IRQ 1	Teclado PS/2 estándar	OK
IRQ 8	Sistema CMOS/reloj en tiempo real	OK
IRQ 11	Controladora de SMBus de la familia Intel(R) ICH10 - 3A30	OK
IRQ 12	Mouse PS/2 de Microsoft	OK
IRQ 13	Procesador de datos numéricos	OK
IRQ 16	Controladora estándar PCI IDE de doble canal	OK
IRQ 16	Controladora de host universal USB de la familia Intel(R) ICH10 - 3A37	OK
IRQ 17	Puerto raíz PCI Express 1 de la familia Intel(R) ICH10 - 3A40	OK
IRQ 17	Puerto raíz PCI Express 5 de la familia Intel(R) ICH10 - 3A48	OK
IRQ 18	Controladora de host universal USB de la familia Intel(R) ICH10 - 3A36	OK
IRQ 18	Puerto raíz PCI Express 3 de la familia Intel(R) ICH10 - 3A44	OK
IRQ 18	Controladora de host mejorado USB de la familia Intel(R) ICH10 - 3A3C	OK
IRQ 18	Realtek PCIe GBE Family Controller	OK
IRQ 19	Controladora de host VIA compatible con OHCI 1394	OK
IRQ 19	Controladora de host universal USB de la familia Intel(R) ICH10 - 3A35	OK
IRQ 19	Puerto raíz PCI Express 4 de la familia Intel(R) ICH10 - 3A46	OK
IRQ 19	Controladora de host universal USB de la familia Intel(R) ICH10 - 3A39	OK
IRQ 20	Controladora estándar PCI IDE de doble canal	OK
IRQ 20	Controladora estándar PCI IDE de doble canal	OK
IRQ 21	Controladora de host universal USB de la familia Intel(R) ICH10 - 3A38	OK
IRQ 22	Controladora de High Definition Audio	OK
IRQ 23	Controladora de host mejorado USB de la familia Intel(R) ICH10 - 3A3A	OK
IRQ 23	Controladora de host universal USB de la familia Intel(R) ICH10 - 3A34	OK
IRQ 24	NVIDIA GeForce GTX 260	OK
IRQ 81	Sistema Microsoft compatible con ACPI	OK
IRQ 82	Sistema Microsoft compatible con ACPI	OK
IRQ 83	Sistema Microsoft compatible con ACPI	OK
IRQ 84	Sistema Microsoft compatible con ACPI	OK

Software Interrupts. System calls and operating systems

- The system call mechanism is the one that allows user programs to request the services offered by the operating system
  - Load programs into memory for execution
  - Access to peripheral devices
  - Etc.
- Similar to the system calls offered by the CREATOR simulator
  - WepSIM examples show how system calls are internally implemented.

Example in WepSIM: int, syscall, exception…

Software interrupts
System calls (example: Linux)



Clock interrupts and operating system

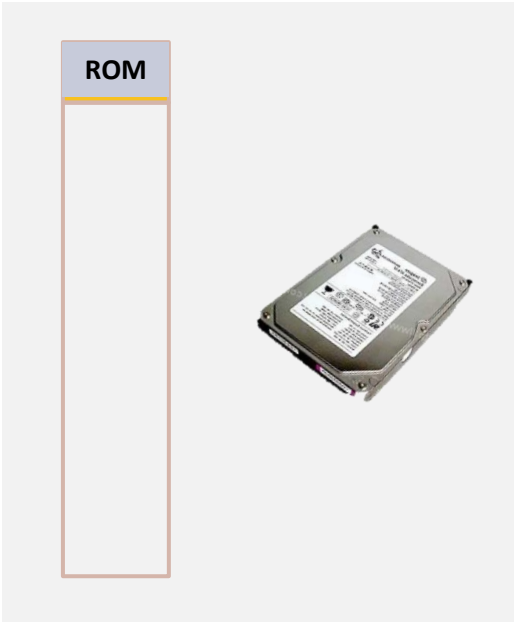
- The signal that governs the execution of machine instructions is divided by a frequency divider to generate an external interruption every certain time interval (a few milliseconds)
- These clock interruptions or ticks are periodic interruptions that allow the operating system to come in and run periodically, preventing a user program from monopolizing the CPU
  - Allows to alternate the execution of various programs on a system given the appearance of simultaneous execution
  - Each time a clock interruption arrives, the program is suspended and the operating system that runs the scheduler is skipped to decide the next program to run

Example in WepSIM: clock usage...

Contents

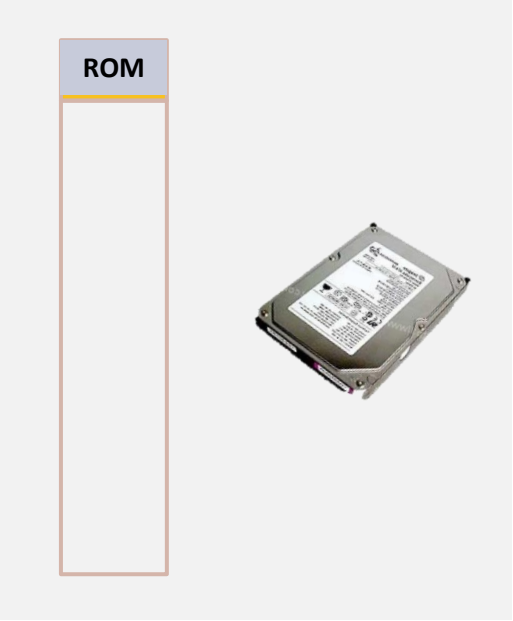
Computer booting

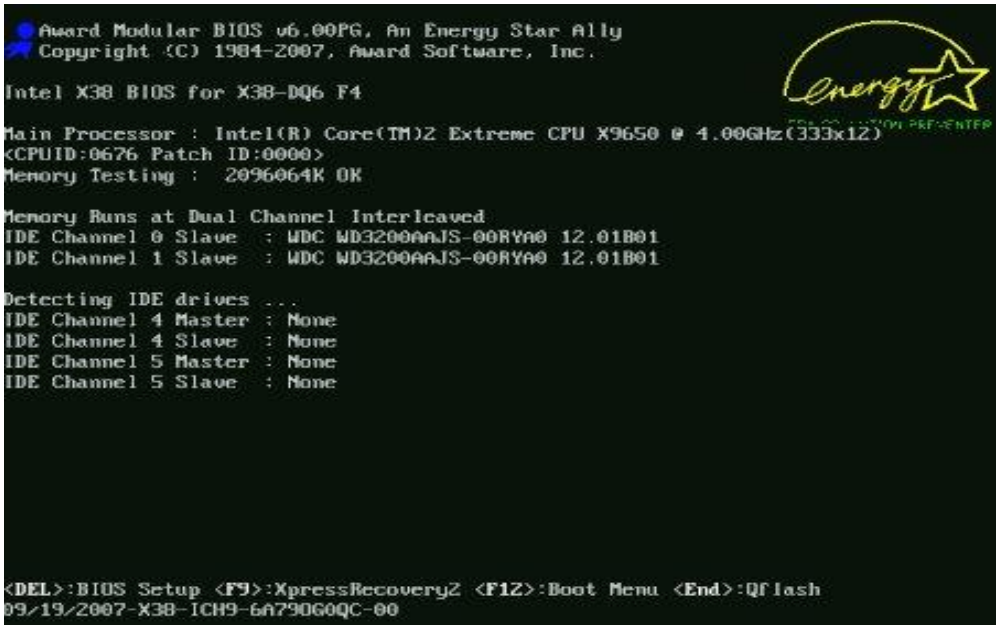
- The *Reset* loads the predefined values in registers:
  - PC ← initial address of the initialization program (in ROM memory)



Computer booting

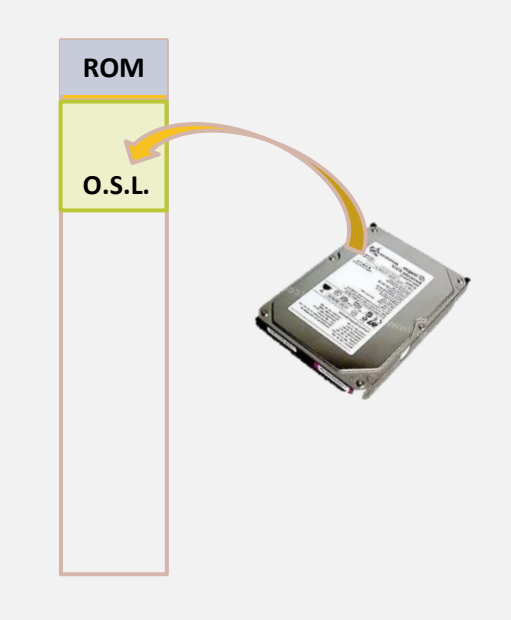
- The *Reset* loads the predefined values in registers:
  - PC ← initial address of the initialization program (in ROM memory)
- The initialization program is executed:
  - System test (POST)



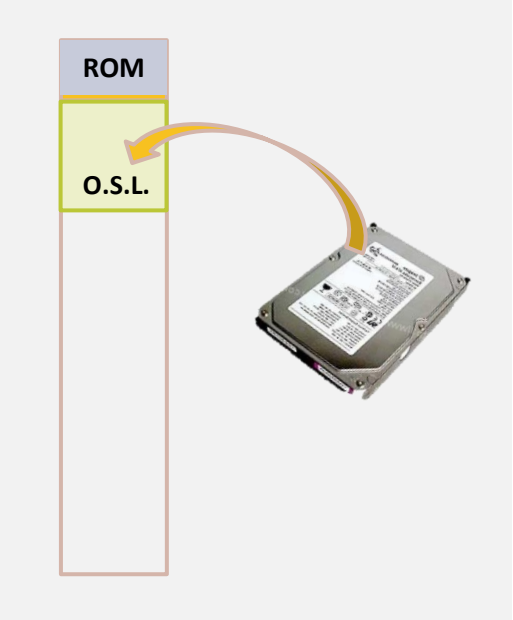


Computer booting

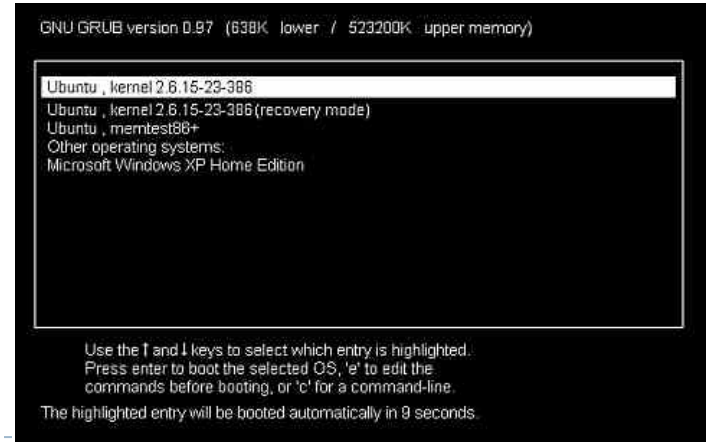
- The *Reset* loads the predefined values in registers:
  - PC ← initial address of the initialization program (in ROM memory)
- The initialization program is executed:
  - System test (POST)
  - Load into memory the operating system loader (MBR)



Computer booting

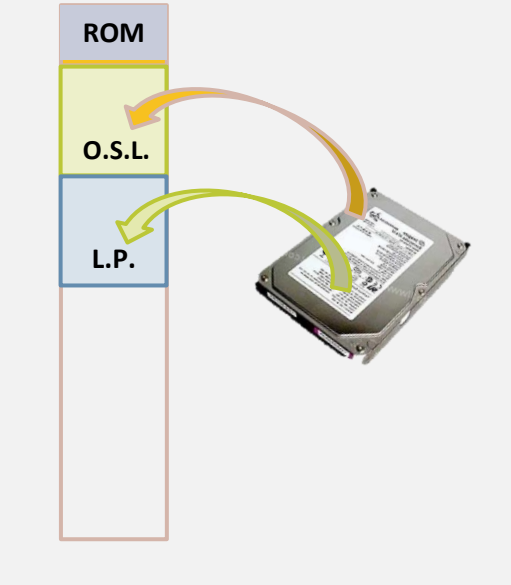


- The *Reset* loads the predefined values in registers:
  - PC ← initial address of the initialization program (in ROM memory)
- The initialization program is executed:
  - System test (POST)
  - Load into memory the operating system loader (MBR)
- The Operating System Loader is executed:
  - Sets boot options

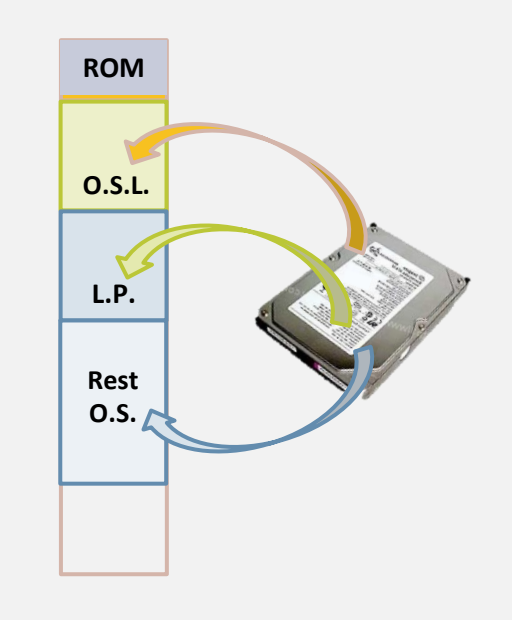


Computer booting

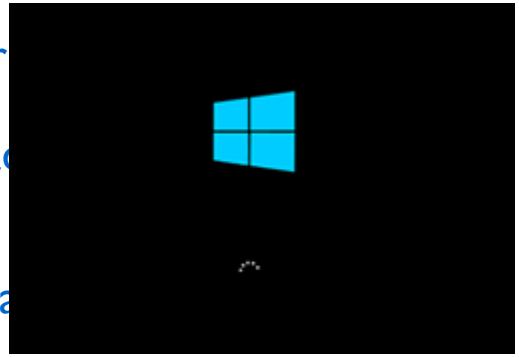
- The *Reset* loads the predefined values in registers:
  - PC ← initial address of the initialization program (in ROM memory)
- The initialization program is executed:
  - System test (POST)
  - Load into memory the operating system loader (MBR)
- The Operating System Loader is executed:
  - Sets boot options
  - Loads the loading program

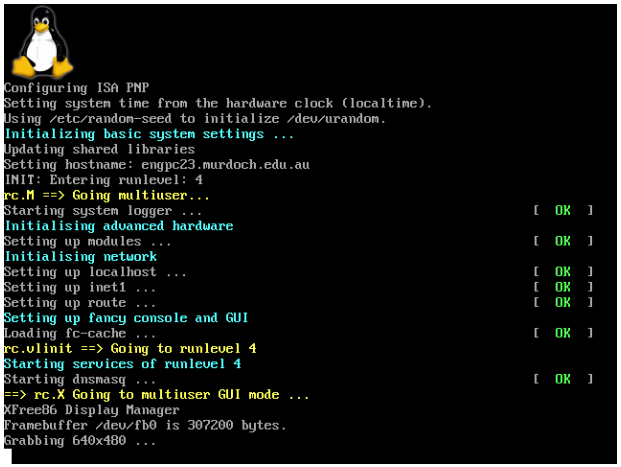


Computer booting

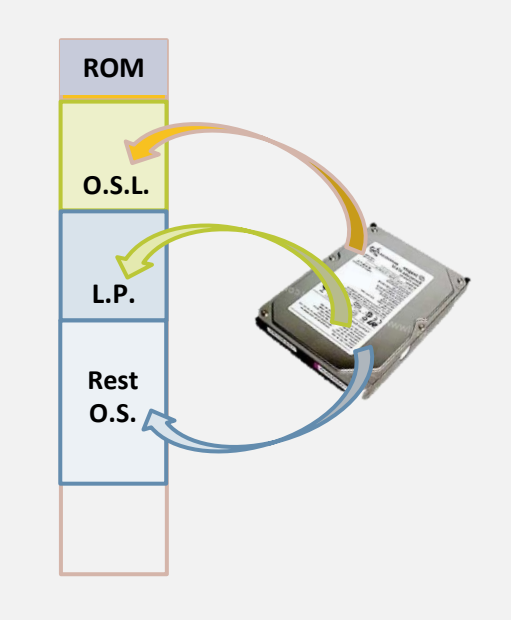


- The *Reset* loads the predefined values in registers:
  - PC ← initial address of the initialization program (in ROM memory)
- The initialization program is executed:
  - System test (POST)
  - Load into memory the operating system loader
- The Operating System Loa
  - Sets boot options
  - Loads the loading progra
- The Loading Program is executed:
  - Sets the initial state of the O.S.
  - Loads the O.S. and executed it.





Computer booting
summary



- The *Reset* loads the predefined values in registers:
  - PC ← initial address of the initialization program (in ROM memory)
- The initialization program is executed:
  - System test (POST)
  - Load into memory the operating system loader (MBR)
- The Operating System Loader is executed:
  - Sets boot options
  - Loads the loading program
- The Loading Program is executed:
  - Sets the initial state of the O.S.
  - Loads the O.S. and executed it.

Program execution time

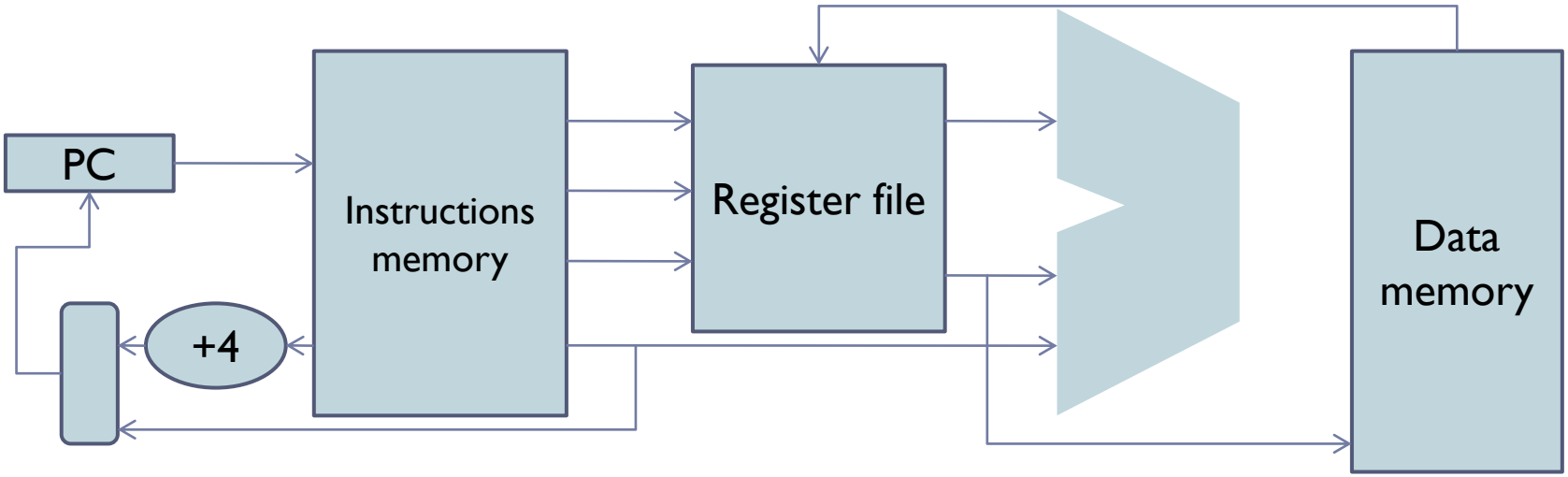
$$Time_{execution} = IN \times CPI \times t_{cycle\_CPU} + IN \times AMI \times t_{cycle\_mem}$$

- IN — is the number of instructions of the program
- CPI — is the average number of clock cycles to execute an instruction
- t_{cycle_CPI} — is the cycle clock duration
- AMI — is the average number of memory access per instruction
- t_{cycle_mem} — is the time needed for a memory access

Factors affecting execution time

	IN	CPI	t_{cycle_CPI}	AMI	t_{cycle_mem}
Program	✓			✓	
Compiler	✓	✓		✓	
Instruction set	✓	✓	✓	✓	
Organization		✓	✓		✓
Technology			✓		✓

Model of processor based on datapath (without internal bus)



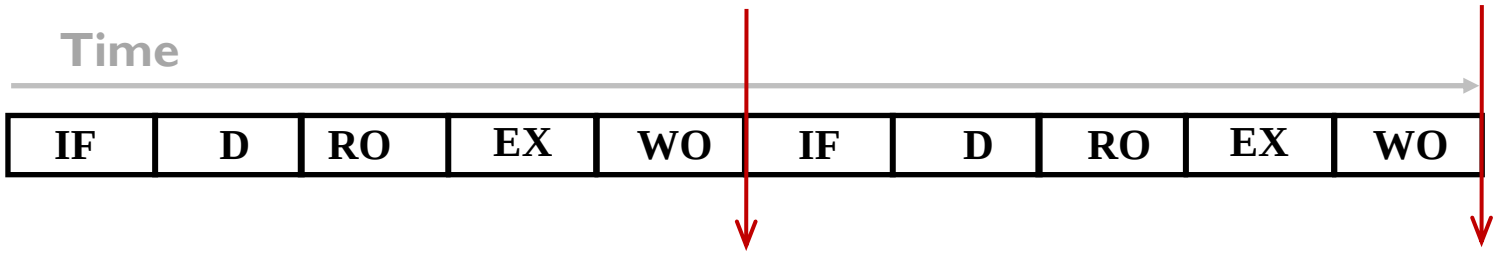
Instruction level parallelism

- Concurrent execution of several machine instructions
- Combination of elements working in parallel:
  - Pipelined processor: use pipelines in which multiple instructions are overlapped in execution
  - Superscalar processor: multiple independent instruction pipelines are used. Each pipeline can handle multiple instructions at a time
  - Multicore processor: several processors or cores in the same chip

Segmentation of instructions

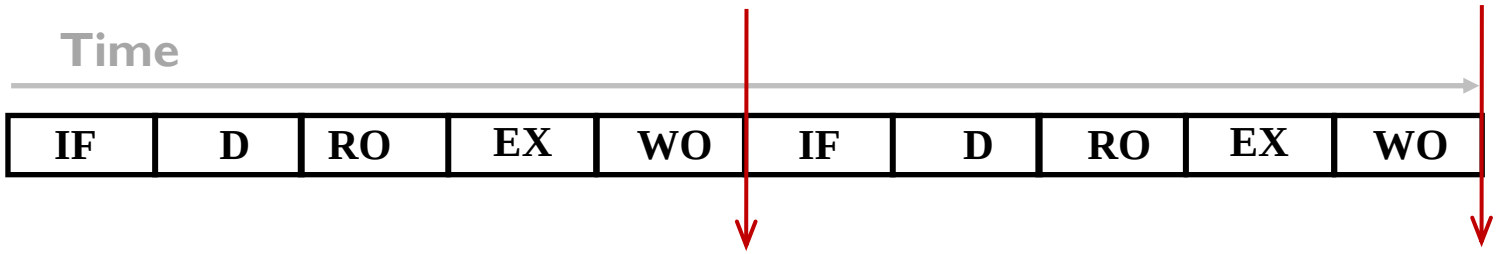
- Stages in the execution of instructions:
  - IF: Instruction fetch
  - D: Decoding
  - RO: Read operands
  - EX: Execution
  - WO: Write operands

Segmentation of instructions
without pipeline



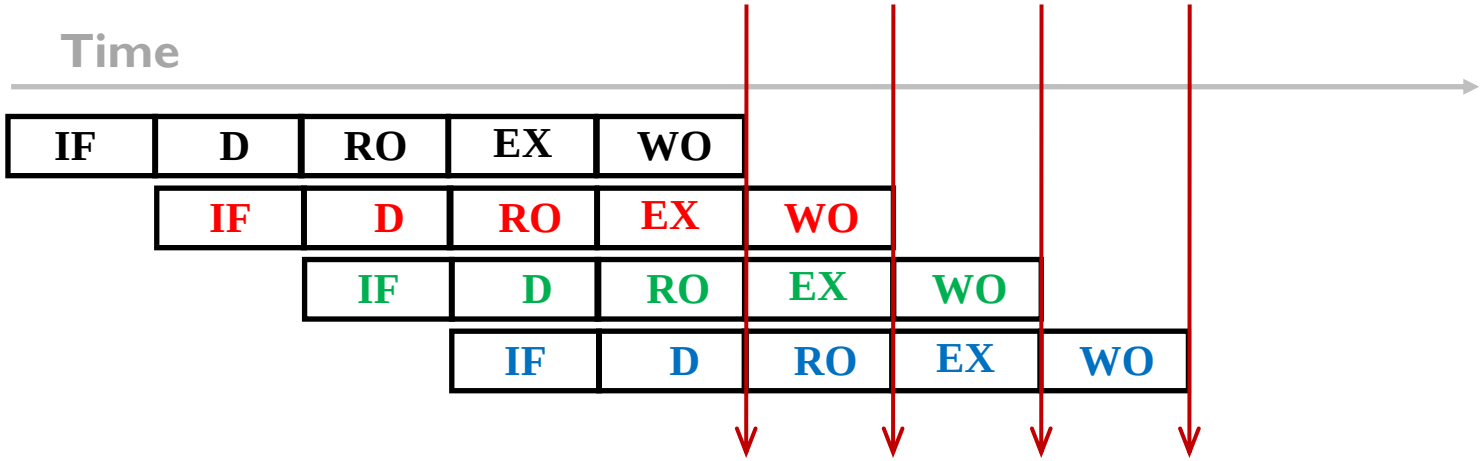
- Stages in the execution of instructions:
  - IF: Instruction fetch
  - D: Decoding
  - RO: Read operands
  - EX: Execution
  - WO: Write operands

Segmentation of instructions
without pipeline



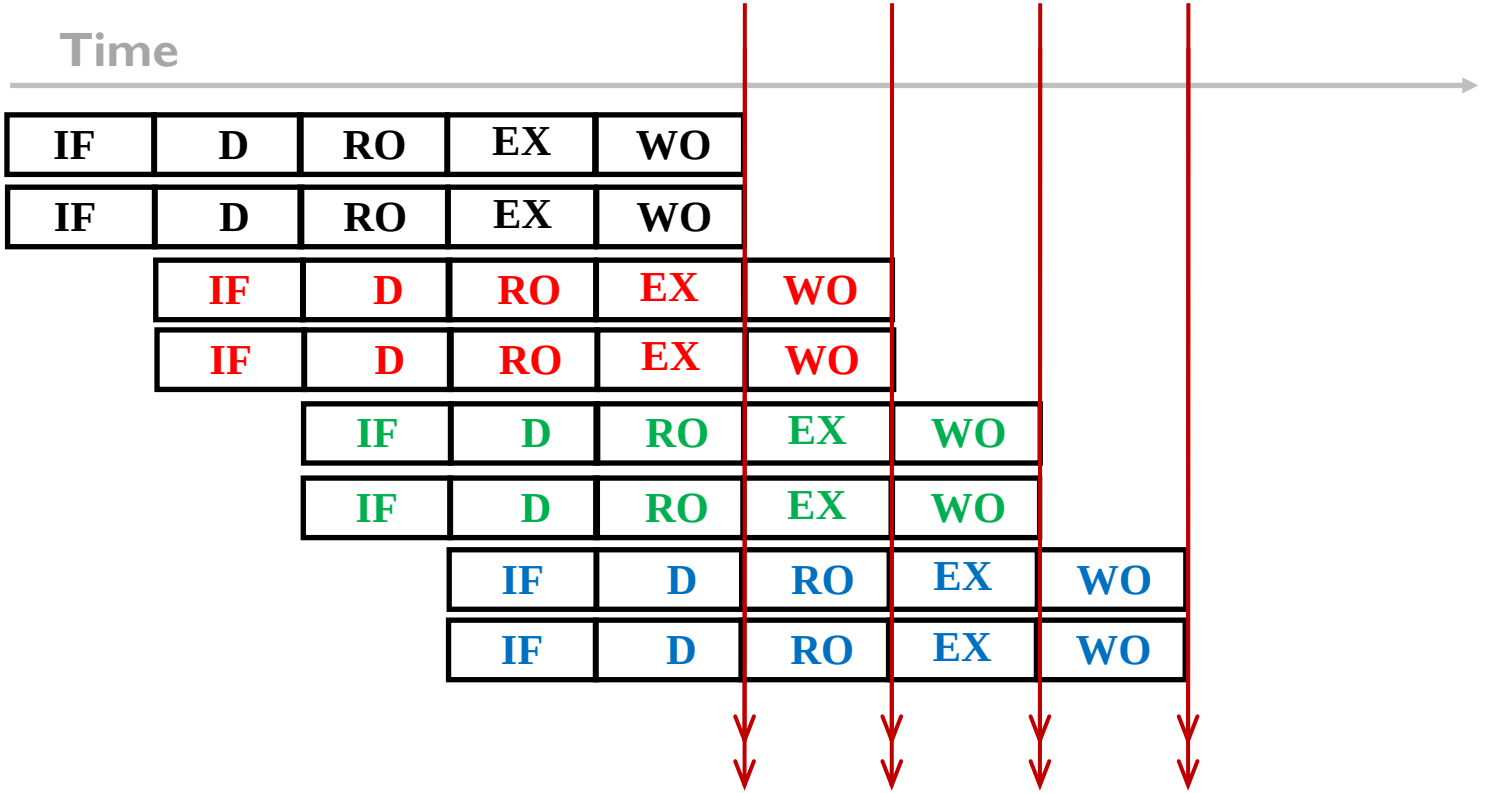
- If each phase takes N clock cycles, then:
  - One instruction takes 5*N clock cycles to be executed
  - 1/5 of instruction is issued every N clock cycles

Segmentation of instructions
with pipeline



- If each phase takes N clock cycles, then:
  - One instruction takes 5*N clock cycles to be executed
  - One instruction is issued every N clock cycles

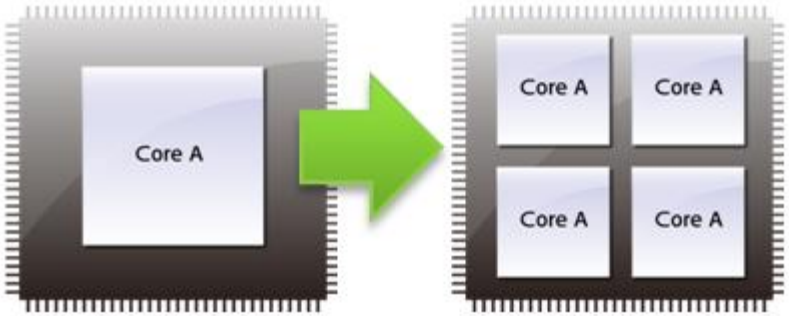
Superscalar

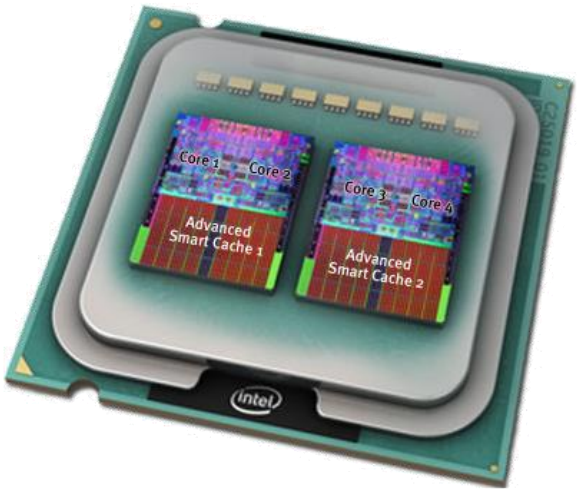


- Pipeline with several functional units in parallel

Multicore

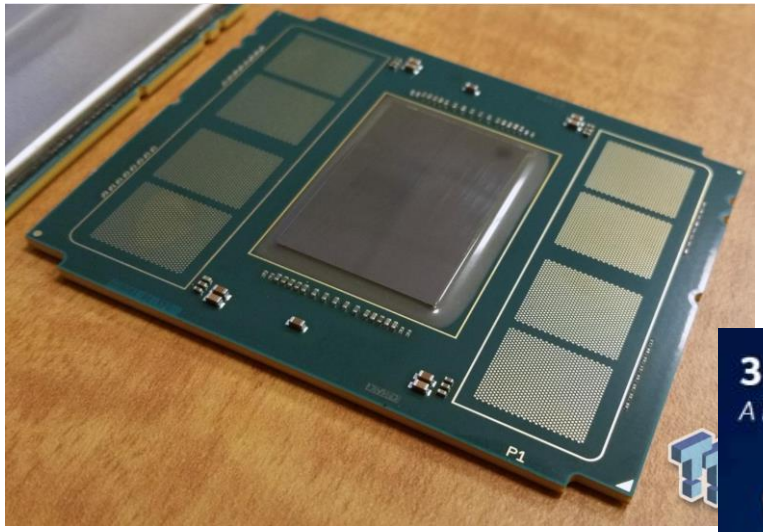
- Multiples processors in the same chip

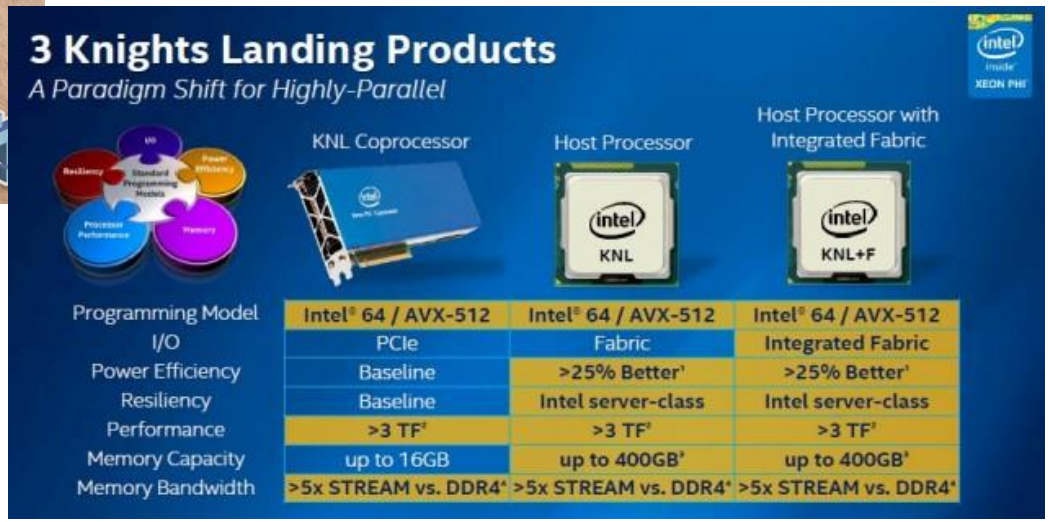




Multicore

- Multiples processors in the same chip





http://wccftech.com/intel-knights-landing-detailed-16-gb-highbandwidth-ondie-memory-384-gb-ddr4-system-memory-support-8-billion-transistors/