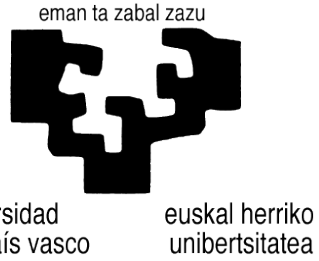
UPV / EHU



Programación Concurrente en Linux

Eventos e interrupciones

OCW OpenCourseWare

UPV / EHU

Contenido

1. Interacción en un programa: entrada/ salida
2. Eventos en la ejecución del programa: interrupciones y excepciones
3. Tratamiento de eventos por el sistema operativo.

OCW OpenCourseWare

UPV / EHU

1. Interacción en un programa: entrada/salida

OCW OpenCourseWare

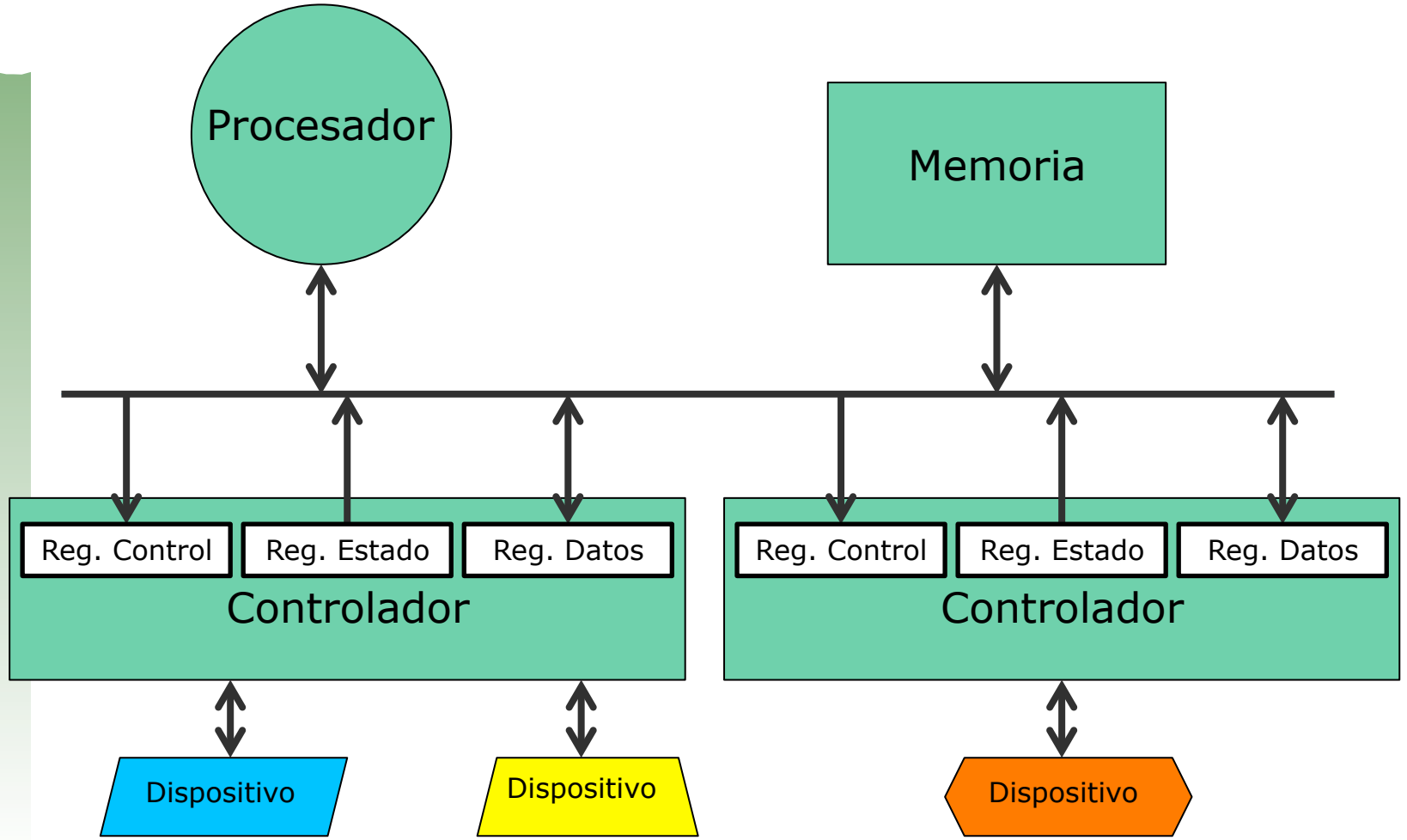
Ejecución de un programa

- Básicamente el programa ejecuta un flujo de instrucciones máquina
  - La mayoría acceden a registros del procesador o a direcciones de memoria
- Algunas de estas instrucciones acceden a los dispositivos controlados por el sistema operativo
  - Por ejemplo, para entrada/salida
- También suceden eventos asíncronos a la ejecución del programa
  - Por ejemplo, una interrupción del reloj

Entrada/salida

- Los dispositivos de entrada/salida son muy heterogéneos:
  - Velocidad
  - Representación de los datos
  - Protocolos
  - Operaciones
  - Unidad de transferencia (bloques, caracteres…)
  - Tipos de errores
  - Modo de tratar la E/S
- Homogeneizar el acceso a los dispositivos:
  - los *controladores de E/S* como interfaz hardware unificada

Entrada/salida
Interfaz hardware



Entrada/salida
Interfaz hardware

UPV / EHU

- Elementos de la interfaz:
  - Espacio de direcciones de E/S, que puede ser
    - Memory-mapped
    - Independiente del de memoria
  - Operaciones de E/S mediante instrucciones máquina
    - Memory-mapped: LOAD/STORE
    - Espacios independientes: IN/OUT

Entrada/salida
Modos de operación

- Encuesta
  1. Espera activa sobre Registro de Estado
  2. Acceso a Registro de Datos
- Interrupciones
  - El dispositivo cuenta con una línea de interrupción
  - Cuando se activa la interrupción, se ejecuta la Rutina de Servicio que gestiona la E/S:
    1. Comprobación sobre Registro de Estado
    2. Acceso a Registro de Datos
- Acceso Directo a Memoria (DMA)
  - Los dispositivos de bloques no involucran a la CPU en el acceso a cada byte
    1. Se programa la operación de DMA
    2. Se ordena su inicio sobre un Registro de Control
    3. El fin de la operación se anuncia mediante una interrupción
    4. La Rutina de Servicio a la Interrupción de DMA comprueba sobre un Registro de Estado.

UPV / EHU

Gestión de la entrada/salida
Manejadores de dispositivos (drivers)

- *Driver*: código que monopoliza el acceso al dispositivo.
- El resto del sistema operativo es independiente del dispositivo.
- Un modelo de entrada/salida: *cliente-servidor*
- Las rutinas de E/S son clientes del driver.

Gestión de la entrada/salida
Interfaz del sistema operativo

UPV / EHU

- El sistema operativo ofrece una interfaz unificada para las operaciones de E/S: las *llamadas al sistema.*
- En Linux:
  - *open(), close(), read(), write()*
- El programador de aplicaciones no tiene que estar pendiente de cómo se implementa la rutina de E/S o el driver del dispositivo concreto.
- Las funciones de biblioteca (por ejemplo de C: *printf(), scanf()*…) usan estas llamadas al sistema.

UPV / EHU

2. Eventos en la ejecución del programa: interrupciones y excepciones

OCW OpenCourseWare

UPV / EHU

Interrupciones y excepciones

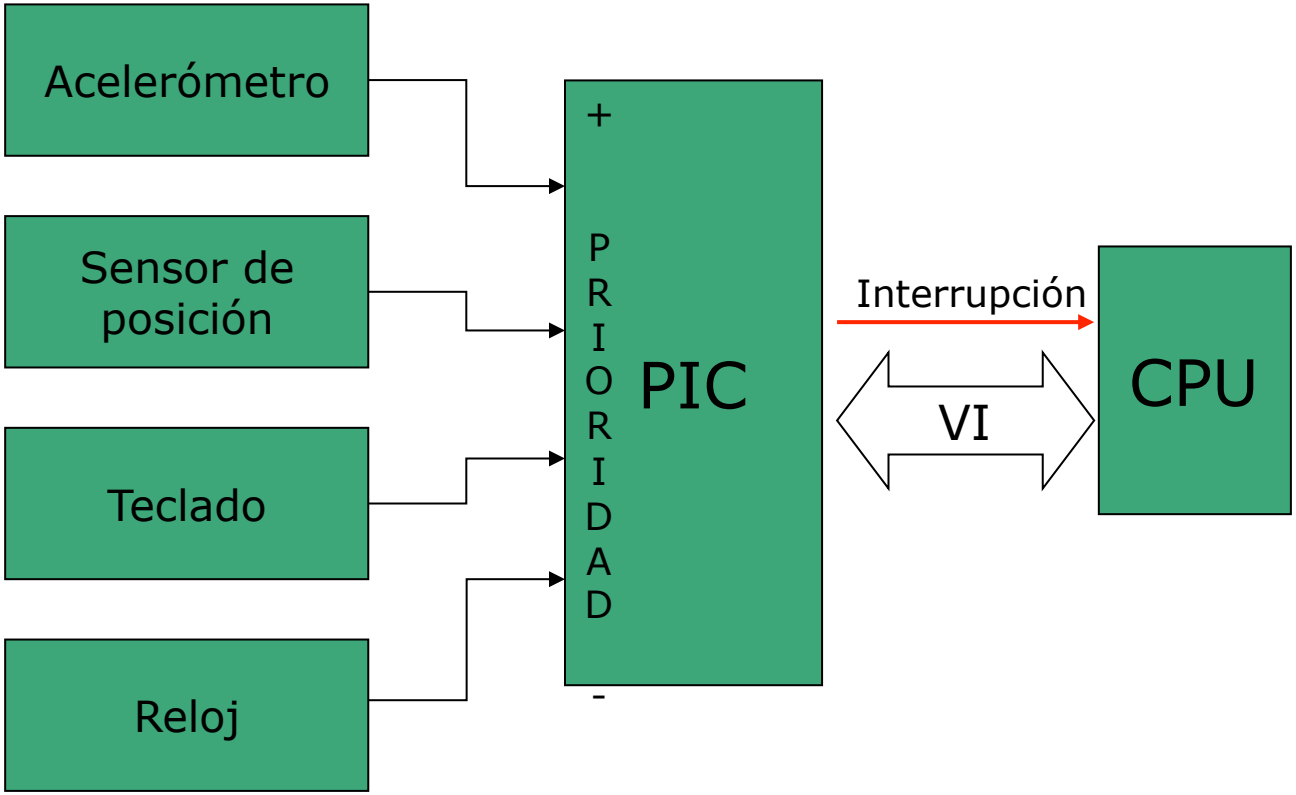
- Excepción:
  - *Evento que, producido durante la ejecución de un programa, provoca que el procesador ejecute un código especial para tratarla.*
  - También llamadas interrupciones internas.
  - Ejemplo: intento de dividir por cero.
- Interrupción:
  - *Condición asíncrona provocada por un dispositivo externo.*
  - También llamadas excepciones asíncronas.
  - Se tratan de la misma forma que las excepciones.
  - Ejemplos: reloj, teclado, DMA…

OCW OpenCourseWare

Hardware para interrupciones

- En un sistema puede haber diferentes fuentes de interrupción, con diferentes prioridades:
  - Sensores
  - Reloj
  - Dispositivos de E/S
- Al procesador del sistema le llega una única línea de interrupción.
- Controlador Programable de Interrupciones (PIC):
  - Prioriza y selecciona las interrupciones.

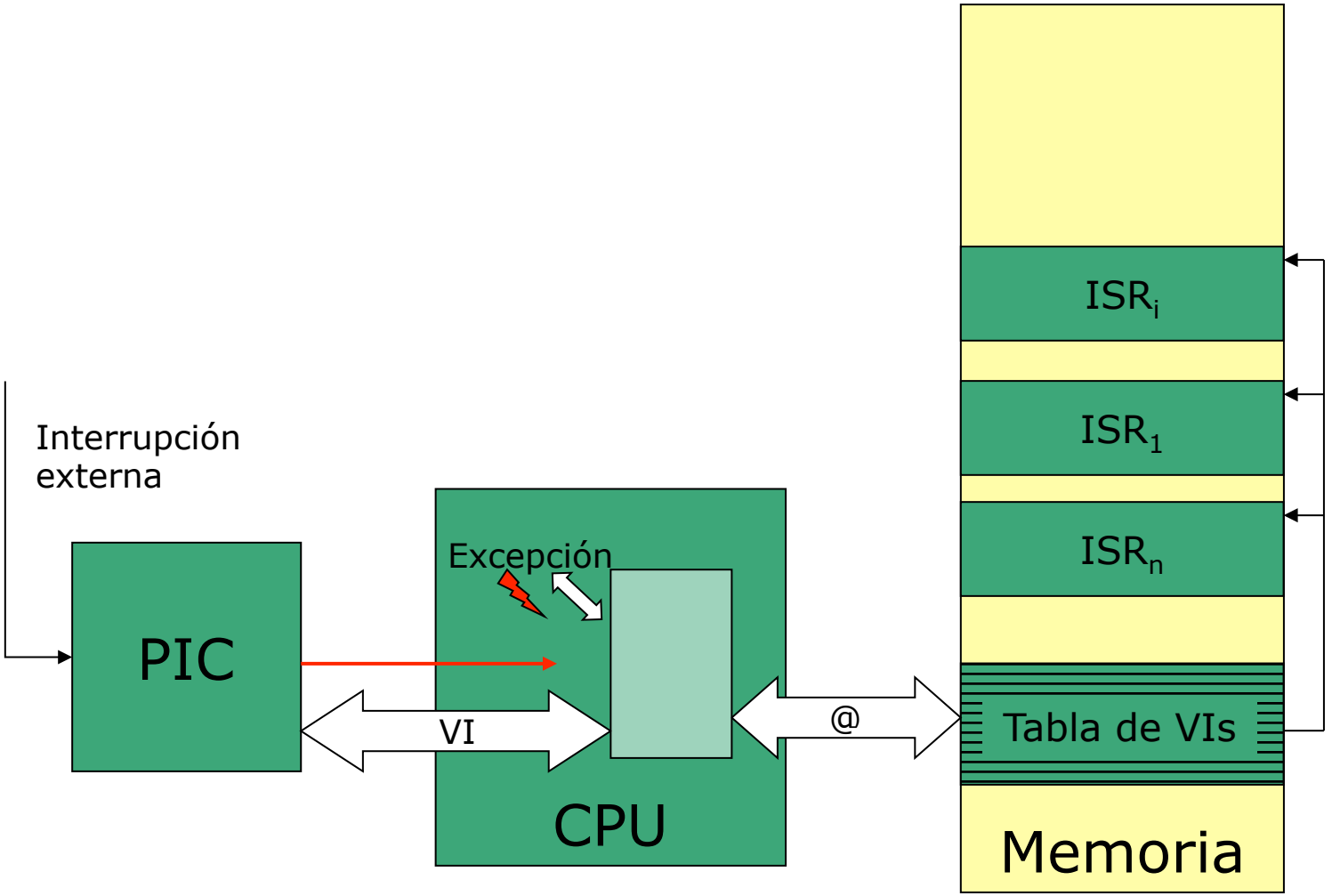
Controladores de interrupciones (PIC)



Software para manejar interrupciones

- En la CPU, las interrupciones pueden
  - Inhibirse (todas ellas)
  - Enmascararse (selectivamente), excepto las No Enmascarables
- Tanto interrupciones como excepciones generan una dirección de entrada a una tabla que contiene las direcciones de comienzo de las *Rutinas de Servicio* (ISR).
  - Atención a la terminología: a esta dirección también se le suele llamar "vector de interrupción".

Manejo de interrupciones
Esquema de direccionamiento de la ISR



Mecanismo de ejecución de una interrupción

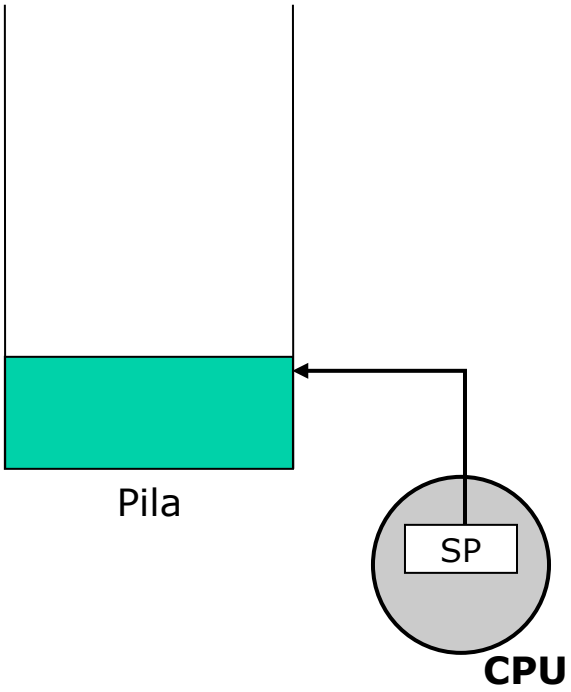
UPV / EHU

- La interrupción/excepción requiere su propio contexto de ejecución (pila).
- Debe guardarse el contexto del programa que se está ejecutando para recuperarlo tras la ejecución.

Ejecución de interrupciones. Ejemplo

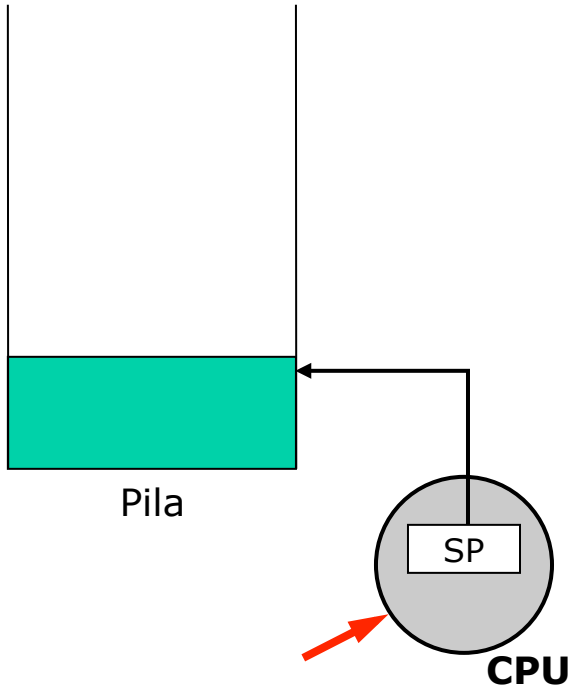
Ejecución de interrupciones. Ejemplo

1. Se está ejecutando una tarea.

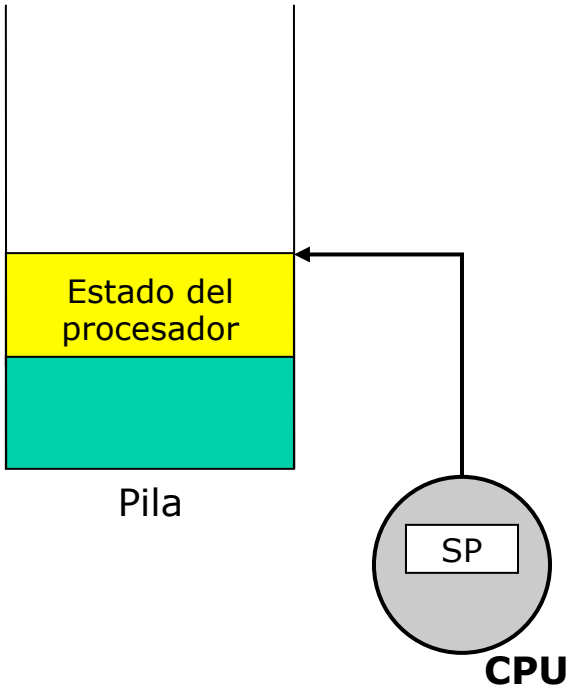


Ejecución de interrupciones. Ejemplo

1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.

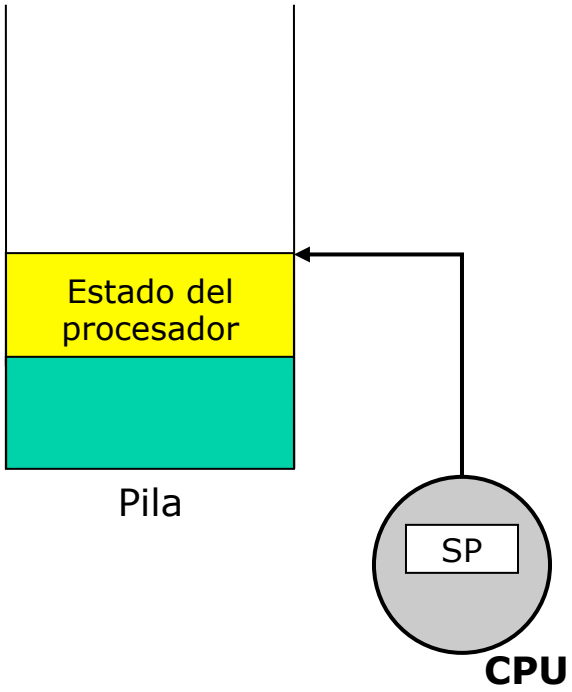


Ejecución de interrupciones. Ejemplo



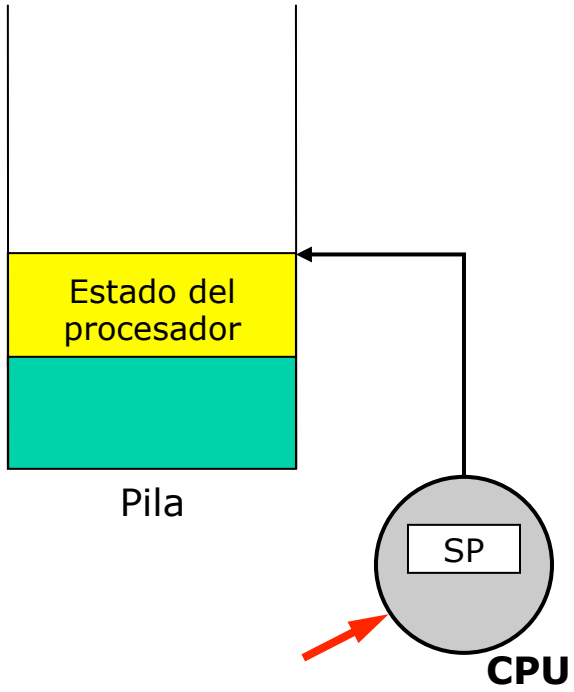
1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.

Ejecución de interrupciones. Ejemplo



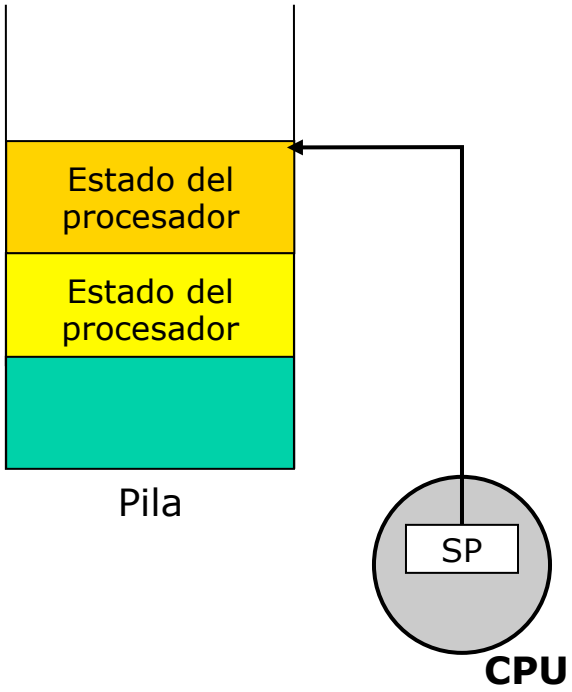
1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.

Ejecución de interrupciones. Ejemplo



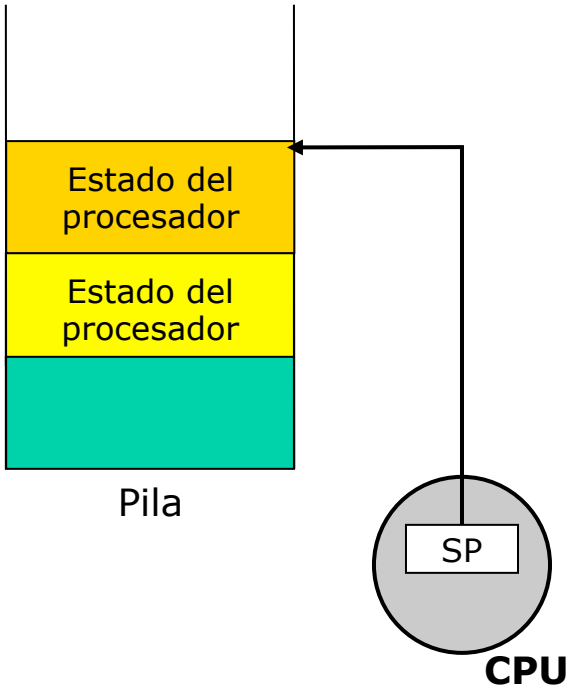
1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.
5. Se produce una interrupción de prioridad alta.

Ejecución de interrupciones. Ejemplo



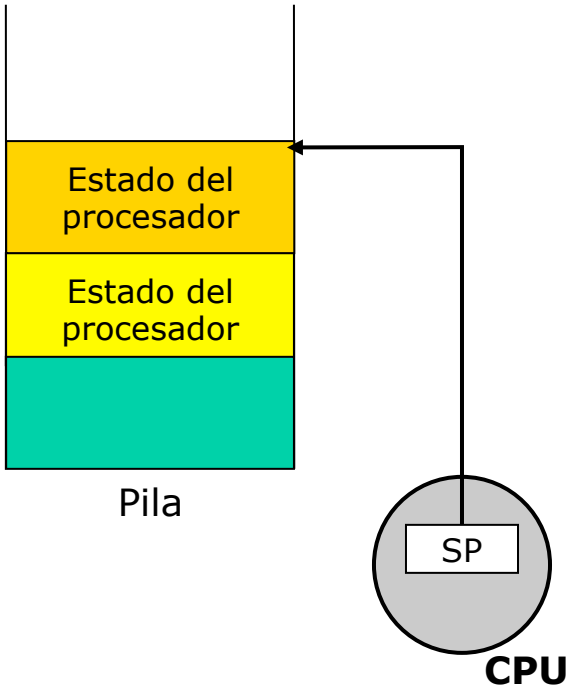
1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.
5. Se produce una interrupción de prioridad alta.
6. Se guarda el estado de procesador.

Ejecución de interrupciones. Ejemplo



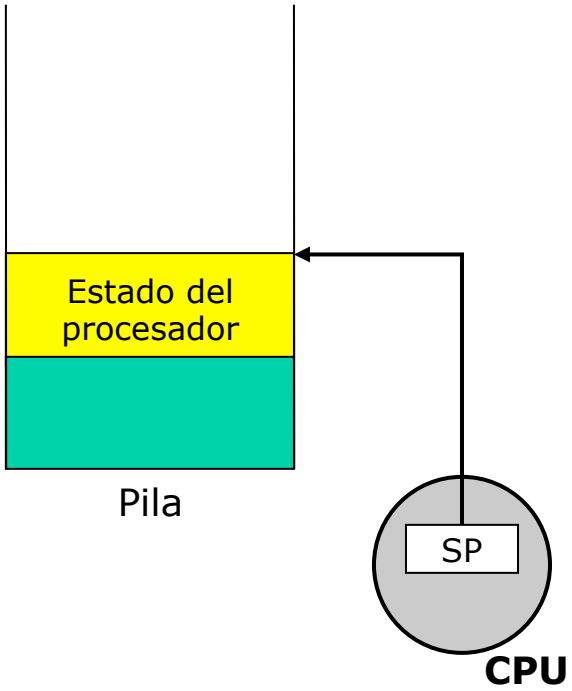
1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.
5. Se produce una interrupción de prioridad alta.
6. Se guarda el estado de procesador.
7. Se ejecuta la rutina de servicio ISR_H.

Ejecución de interrupciones. Ejemplo



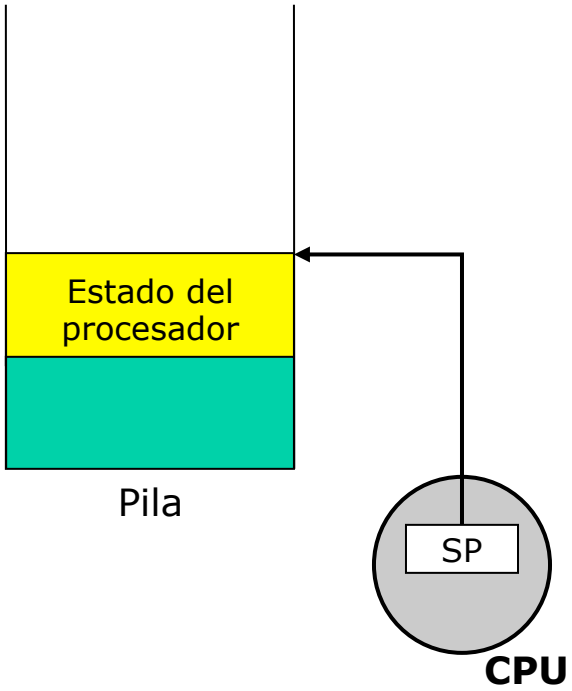
1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.
5. Se produce una interrupción de prioridad alta.
6. Se guarda el estado de procesador.
7. Se ejecuta la rutina de servicio ISR_H.
8. Termina ISR_H. Se vuelve a ISR_L.

Ejecución de interrupciones. Ejemplo



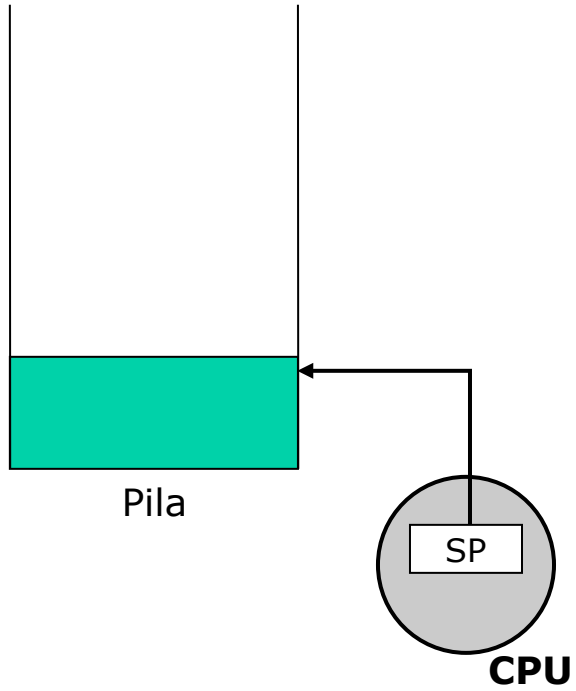
1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.
5. Se produce una interrupción de prioridad alta.
6. Se guarda el estado de procesador.
7. Se ejecuta la rutina de servicio ISR_H.
8. Termina ISR_H. Se vuelve a ISR_L.

Ejecución de interrupciones. Ejemplo



1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.
5. Se produce una interrupción de prioridad alta.
6. Se guarda el estado de procesador.
7. Se ejecuta la rutina de servicio ISR_H.
8. Termina ISR_H. Se vuelve a ISR_L.
9. Termina ISR_L. Se vuelve al punto de ejecución de la tarea.

Ejecución de interrupciones. Ejemplo



1. Se está ejecutando una tarea.
2. Se produce una interrupción de prioridad baja.
3. Se guarda el estado de procesador en la pila del programa.
4. Se ejecuta la rutina de servicio ISR_L.
5. Se produce una interrupción de prioridad alta.
6. Se guarda el estado de procesador.
7. Se ejecuta la rutina de servicio ISR_H.
8. Termina ISR_H. Se vuelve a ISR_L.
9. Termina ISR_L. Se vuelve al punto de ejecución de la tarea.

UPV / EHU

3. Tratamiento de eventos por el sistema operativo

OCW OpenCourseWare

Tratamiento de eventos por el sistema operativo
Ejemplo: el reloj

UPV / EHU

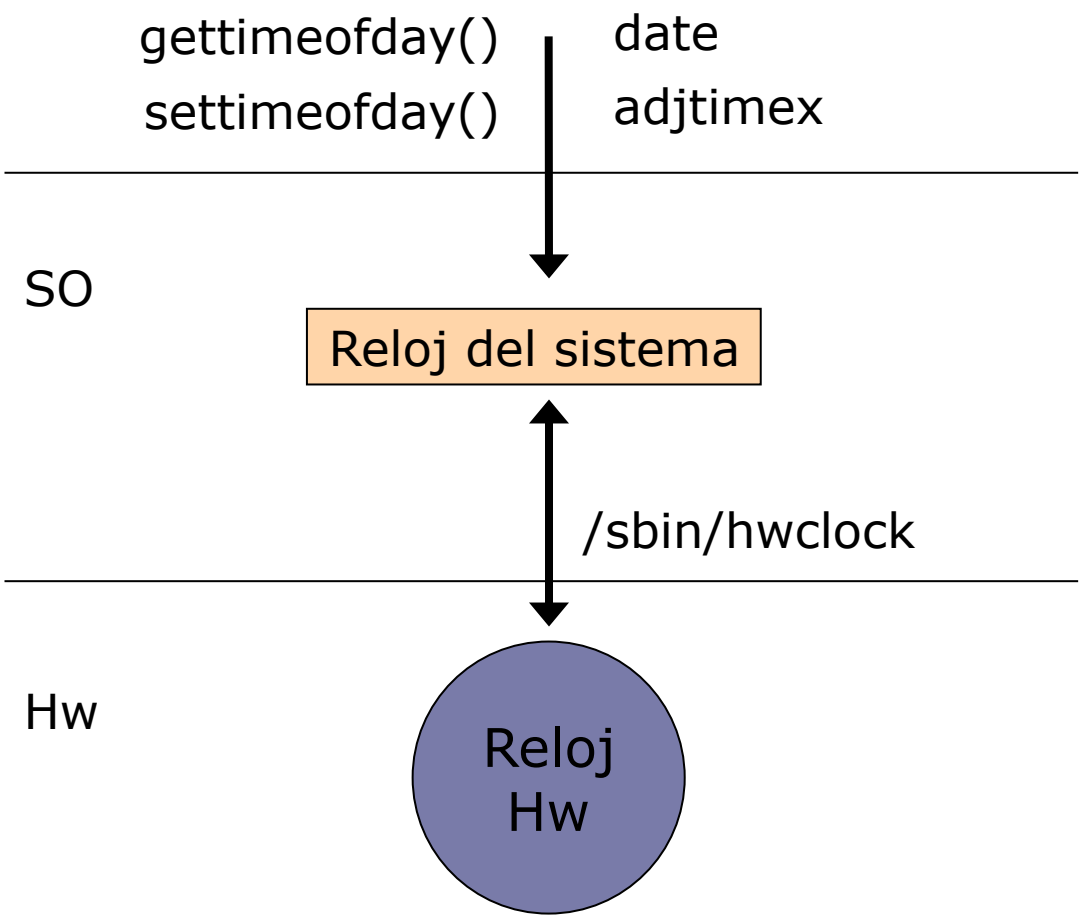
- Recordemos que el sistema operativo ofrece al programador una interfaz unificada de llamadas al sistema para tratar eventos de E/S (p.ej., *read()* en Linux).
- Para los eventos de tiempo un sistema operativo ofrece dos servicios:
  - Un reloj software
    - En Linux, *gettimeofday()*
  - Temporización (programar evento de tiempo)
    - En Linux, *alarm()*

Relojes

- Reloj de tiempo real (RTC)
  - Siempre en funcionamiento. Requiere alimentación propia.
- Reloj Hardware
  - Se carga con el RTC al arrancar el sistema.
  - Produce las interrupciones de reloj.
  - Suele ser programable (PIT, p. ej., Intel 8253).
  - Puede tener otras funciones (p. ej., refresco de la DRAM).
- Reloj del sistema
  - Es un reloj software implementado a partir de la rutina de interrupción del reloj hardware.
    - Por ejemplo, si el reloj hw interrumpe con una frecuencia *f*, cada *f* interrupciones incrementa un contador de segundos.

UPV / EHU

Relojes en Linux



Relojes en Linux
Ejemplo: cronómetro de alta resolución

```
#include <sys/time.h>

struct timeval t0, t1;


gettimeofday(&t0, NULL);

/* aquí, el código a cronometrar */

gettimeofday(&t1, NULL);

printf("Duracion: %d,%d segundos\n",
            t1.tv_sec-t0.tv_sec, t1.tv_usec-t0.tv_usec);
```