Generalidades de las interrupciones

Sistemas con Microprocesadores

Mg.Ing. Esteban Volentini (evolentini@herrera.unt.edu.ar)

http://www.microprocesadores.unt.edu.ar/procesadores

Cronograma

Actividad	Inicio	Descripción	Completa
Presentación	24/08	Presentación y reglamento	✔
Tema 1	24/08	Estructura de las computadoras	✔
Tema 2	26/08	Proyecto con un microcontrolador	✔
Tema 3	31/08	Descripción funcional del ISA RV32IM	✔
Tema 4	07/09	Programación en lenguaje ensamblador	✔
Tema 5	18/09	Diseño del camino de datos de un procesador	✔
Tema 6	30/09	Diseño de la unidad de control del procesador	✔
Tema 7	05/10	Tecnologias y sistema de memoria	✔
Tema 8	09/10	Dispositivos de entrada/salida	✔
Tema 9	19/10	Entradas/salidas digitales (GPIO)	✔
Tema 10	28/10	Generalidades de las interrupciones	←
Tema 11	04/11	Interrupciones y excepciones en RISC-V	
Tema 12	13/10	Manejo de señales con temporizadores	
Tema 13	18/11	Transferencias de datos	
Proyectos	30/11	Seminarios de proyectos	

Temario

- Interrupciones en un procesador
- Enmascaramiento y latencia
- Manejo de múltiples fuentes
  - Conexión, Identificación y Prioridades
- Espera pasiva, NMI y RESET
- Excepciones y traps
- Niveles de privilegio
- Llamadas al sistema operativo

Entrada/salida por encuestas

- En el tema anterior se interactuó con dispositivos de entrada/salida
- En el caso particular de las teclas se implementó un sistema de encuestas
  - Un lazo lee el valor de una entrada
  - Si se cumplen ciertas condiciones se ejecuta una acción
- Sin embargo las salidas tambien pueden necesitar un mecanismo similar
  - Se debe refrescar la pantalla multiplexada
- Este patrón se denomina polling o encuestas

Algo mejor que las encuestas

- El mecanismo de encuestas es simple pero tiene algunos problemas
  - Cuando el evento es poco frecuente se utiliza mucho procesamiento con un bajo rendimiento
  - Cuando los tiempos tiene mucha diferencia de magnitud y los márgenes de error son acotados es muy difícil cumplir con los requisitos temporales
- En este caso sería más eficiente implementar un sistemas que permita suspender el programa en curso y atender el evento

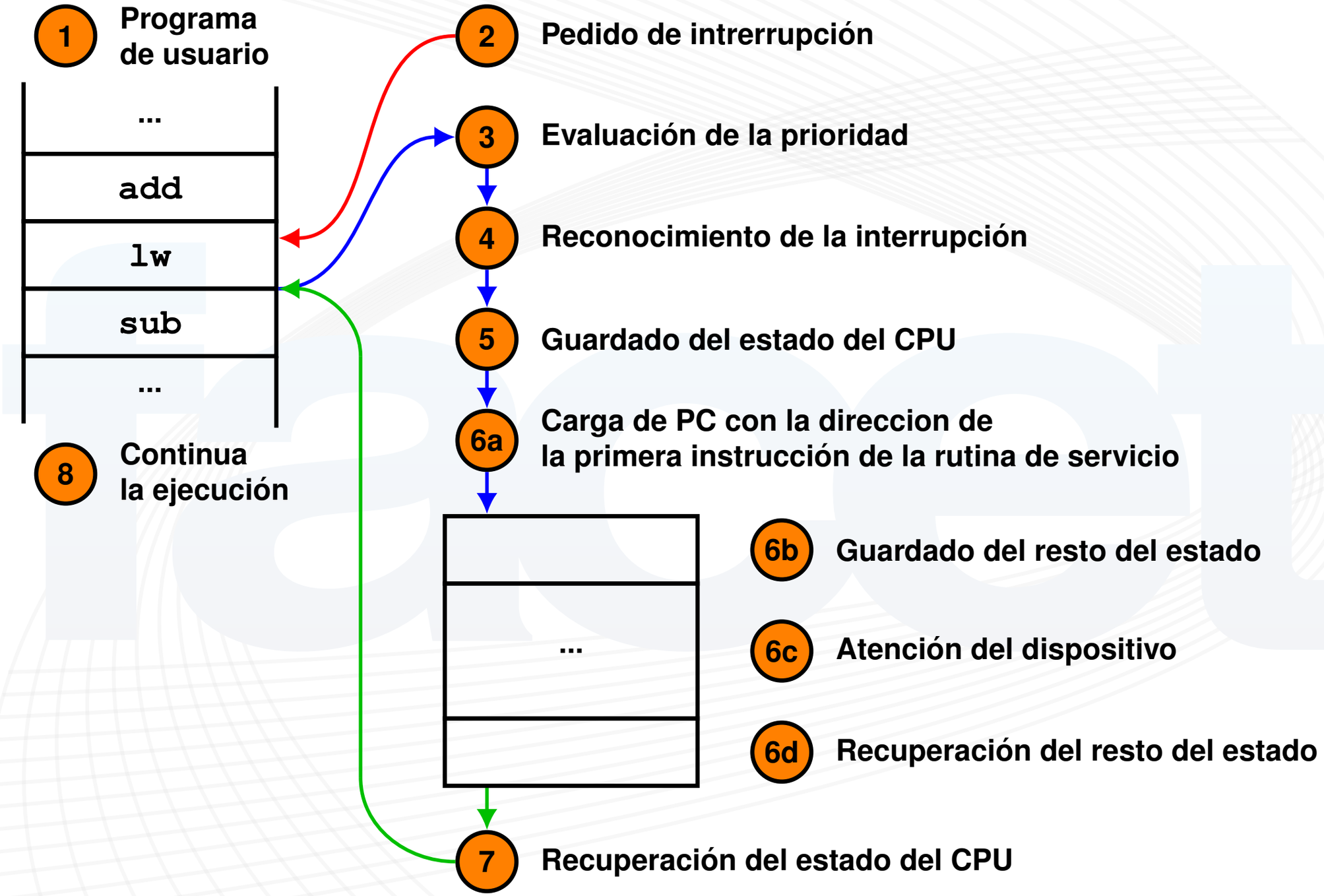
Interrupciones en la vida real

1) Un estudiante está leyendo
2) Lo llaman por teléfono
3) Decide si atenderá
4) Indica que atenderá
5) Pone un señalador
6) Atiende el teléfono
7) Abre en el señalador
8) Retoma el estudio

Interrupciones en un procesador

1) El procesador ejecuta un programa
2) Llega un pedido al control por la linea /INT
3) Si la prioridad del pedido es mayor que la del programa
4) El control genera una señal de reconocimiento /INTA
5) Guarda el valor actual del PC y ciertos registros
6) Atiende la interrupción
   a) Carga el PC con la dirección de la rutina de servicio
   b) Ejecuta una rutina que atiende el pedido de interrupción
7) Al terminar recupera el valor del PC y ciertos registros
8) Continua la ejecución del programa original

Punto de vista del programador



Conclusion del mecanismo

- Ahora el evento se informa, por una linea de entrada directamente al control del procesador
- El control altera el flujo normal del programa en curso y ejecuta un programa alternativo
- Es como llamar a una subrutina por hardware, pero no se puede predecir cuando se la llamará:
  - Pero la dirección de inicio de la subrutina tiene que estar definida de antemano
  - La subrutina debe conservar los valores de todos los registros porque el programa principal no puede hacerlo
- Es equivalente a un sistema de notificaciones

Requerimientos del dispositivo

- El dispositivo que genera la interrupción debe tener la capacidad de mantener el pedido hasta que sea atendido
  - Se necesita un elemento de memoria interno
- Se suelen utilizar bits de un registro de estado para generar el pedido de interrupción
  - Si cualquier de esos bits está activo entonces se activa la linea de pedido de interrupción por parte del dispositivo
- Es habitual tambien que el dispositivo incluya un bit para habilitar el pedido de interrupción
  - Aunque los bits de registro de esta estén activos la linea de pedido de interrupción permanece inactiva
  - Muchas veces se habla de enmascarar en lugar de habilitar

Enmascaramiento

- Las interrupciones se pueden enmascarar, es decir ignorar durante un tiempo
  - En general los pedidos quedan pendientes hasta ser atendidos
  - A veces se puede borrar un pedido sin atenderlo
- En general una interrupción se puede enmascarar de más de una forma
  - En el registro de control del dispositivo (individual)
  - En un registro de control del procesador (global)
- En algunos contexto se habla de habilitar o deshabilitar las interrupciones en lugar de enmascararlas

Registros de control y estado del CPU

- Todos los procesadores deben almacenar información sobre su estado de ejecución actual
  - Por ejemplo si esta ejecutando un programa normal o atendiendo una interrupción
- Tambien se pueden configurar algunas características del comportamiento del procesador
  - Por ejemplo si enmascara las interrupciones
- Para esto los procesadores tienen unos registros especiales de control y estado

Latencia

- Se llama latencia al tiempo que transcurre entre la generación del pedido y la atención del mismo
  - En la latencia se cuenta tanto el tiempo que necesita el procesador para ejecutar la rutina de servicio como las esperas por enmascaramiento
- El momento en que se atiende el pedido de interrupción puede ser un poco difuso
  - El inicio de la rutina de servicio
  - El final de la rutina de servicio
  - El momento exacto en el que se resuelve el pedido del dispositivo

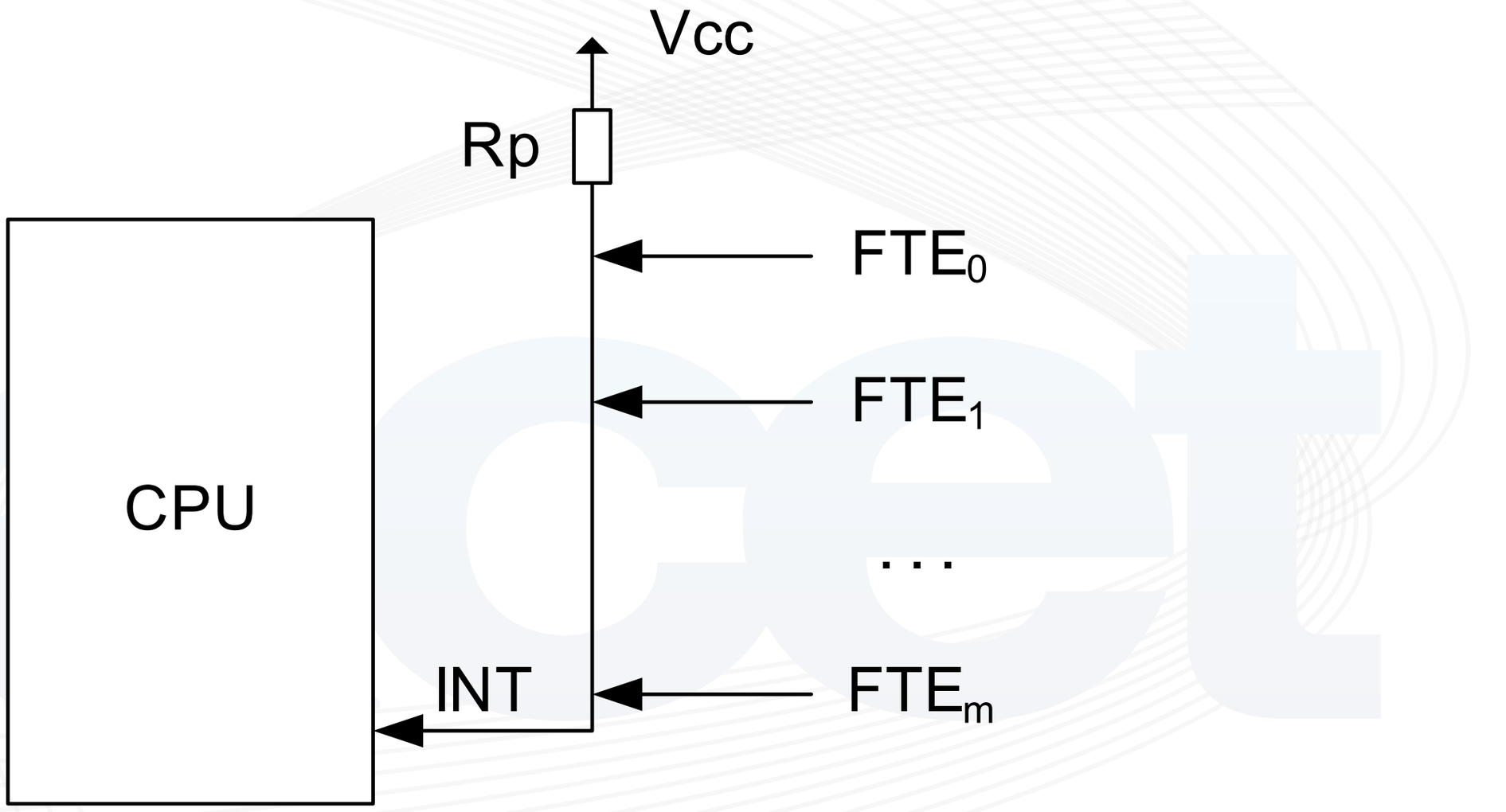
Resumen de las interrupciones

- Interrupciones
  - Causada por eventos externos
  - Asíncronas con la ejecución del programa
  - Reclaman un servicio
- Transparentes al programa principal
  - No tiene forma de saber cuándo llega
  - No puede prepararse para ello
- Es como llamar a una subrutina pero por hardware
- Las interrupciones se pueden enmascarar
  - Quedan pendientes hasta que sean atendidas
  - Se llama latencia al tiempo entre el pedido y la atención

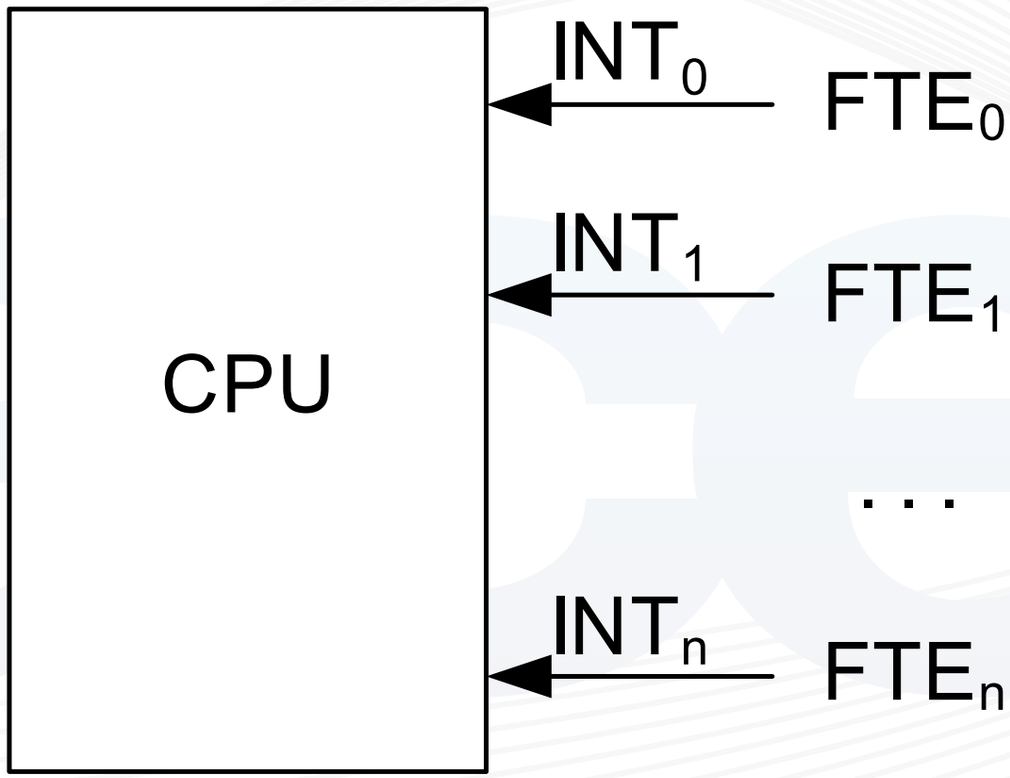
Implementación real en procesadores

- Pero suele haber varias fuentes de interrupciones, lo que implica resolver las siguientes cuestiones:
  - Conexión externa
    - ¿Cómo se conectan al CPU?
  - Identificación
    - ¿Cómo se identifica quién generó el pedido de interrupción?
  - Simultáneas
    - ¿Qué pasa cuando llegan varios pedidos de interrupciones de diferentes dispositivos al mismo tiempo?
  - Anidamiento
    - ¿Que pasa cuando aparece un nuevo pedido de interrupción mientras todavía se atiende un pedido anterior?

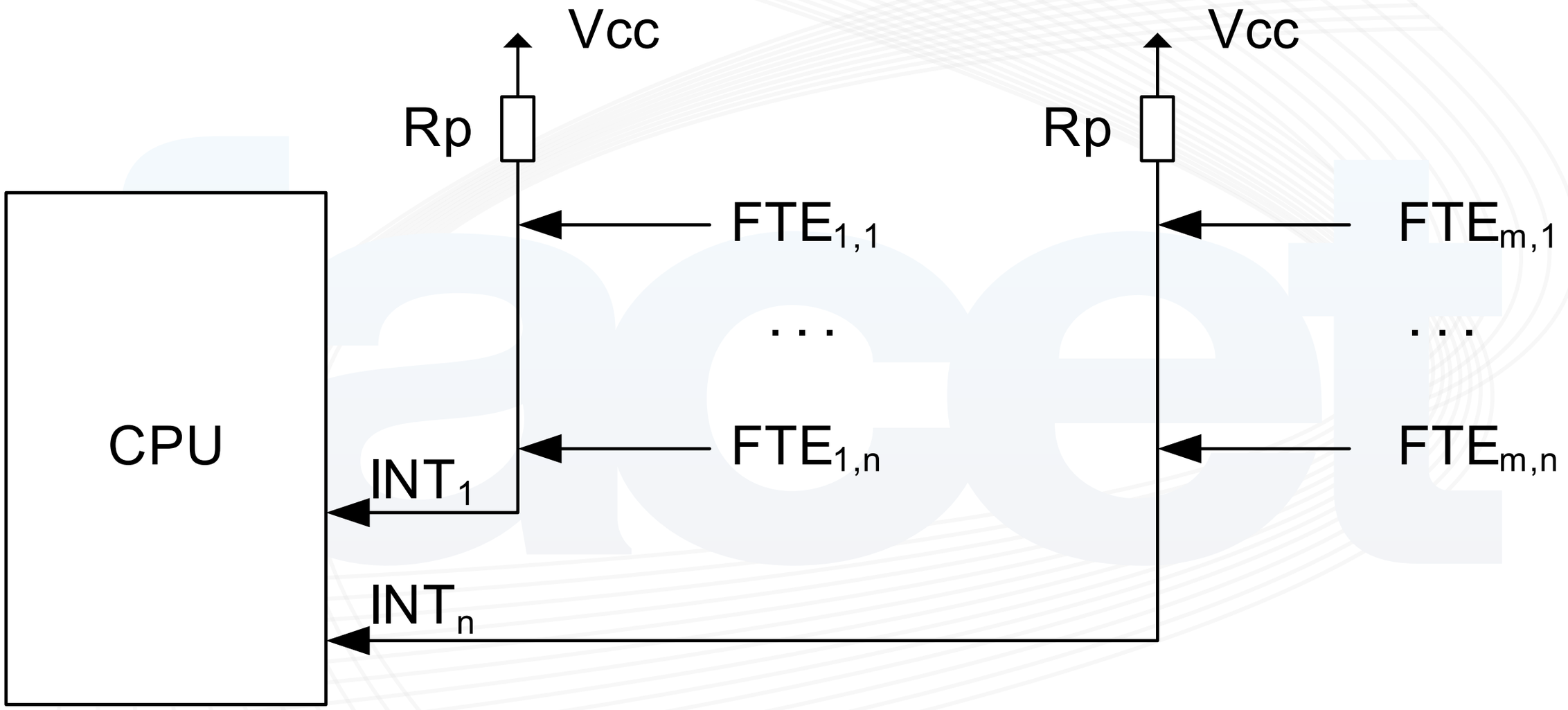
Conexión a colector abierto



Múltiples lineas de interrupciones

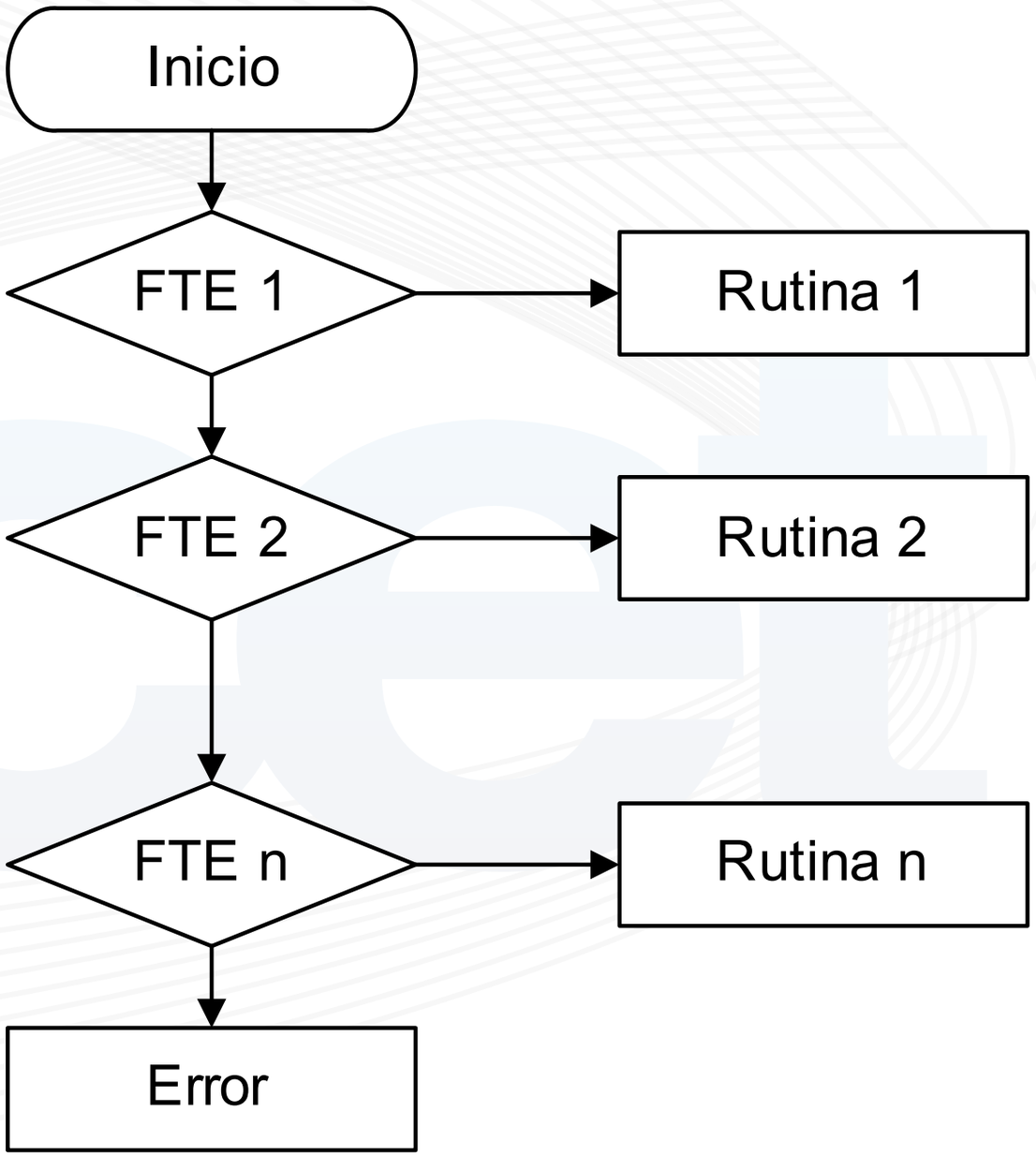


Múltiples lineas a colector abierto



Identificación por software (polling)

- Se puede leer el registro de estado de cada uno de los dispositivos para verificar si están activos los bits que generan un pedido de interrupción



Identificación en múltiples líneas

- Cada línea se identifica con un número
- Se puede guardar ese identificador de la causa en un registro especial
- Se puede utilizar la identificación por software evaluando ese registro
- Tambien se puede saltar directamente a cada rutina utilizando una tabla de saltos (vector de interrupciones)

Pedidos simultáneos de interrupciones

- Son simultáneas las que ocurren en el intervalo de una instrucción
- También son simultáneas las que quedaron sin atender de una ronda previa
- Se resuelve por prioridades
- El sistema de prioridades debería ser equitativo: que nadie quede sin atender
  - Es decir que todos los dispositivos fueron atendidos antes del tiempo máximo que cada uno puede esperar

Prioridades en pedidos simultáneos

- Software (Polling)
  - El que se encuesta primero es prioritario
  - Permite esquemas elaborados más elaborados
    - Por ejemplo Round Robin con dos prioridades
- Hardware
  - Cada línea tiene su prioridad
  - Se las puede configurar por software
  - También permite esquemas más elaborados

Pedidos anidados de interrupciones

- Durante la atención de una interrupción llega un nuevo pedido de interrupción
  - Se puede suspender la rutina en curso y atender el pedido
  - Se puede posponer el pedido hasta terminar la rutina en curso
- El diseñador evalúa su uso en función de las necesidades
  - Usar solo en caso necesario, es complejo
- Se utilizan prioridades para definir el comportamiento
  - Sin embargo estas prioridades son independientes de las que se utilizan para definir en caso de pedidos simultáneos

Un esquema de prioridades flexible

- Las prioridades para la simultaneidad son de una jerarquía menor que las usadas para el anidamiento
  - Si una interrupción es lo suficientemente importante para anidarse con otra, entonces es lógico pensar que tambien debe atenderse antes si las dos ocurren simultáneamente
  - Sin embargo que una interrupción se atienda antes que otra no implica que necesariamente deba anidase si ocurren en diferentes tiempos
- Una solución moderna a este problema consiste en utilizar un valor numérico que se divide en dos campos
  - Para resolver los pedidos simultáneos se evalua el valor completo
  - Pero para resolver anidamientos solo se evalúa el campo formado por los bits mas significativos del valor, y de denomina grupo o nivel

Respuesta rápida o Tail-chaining

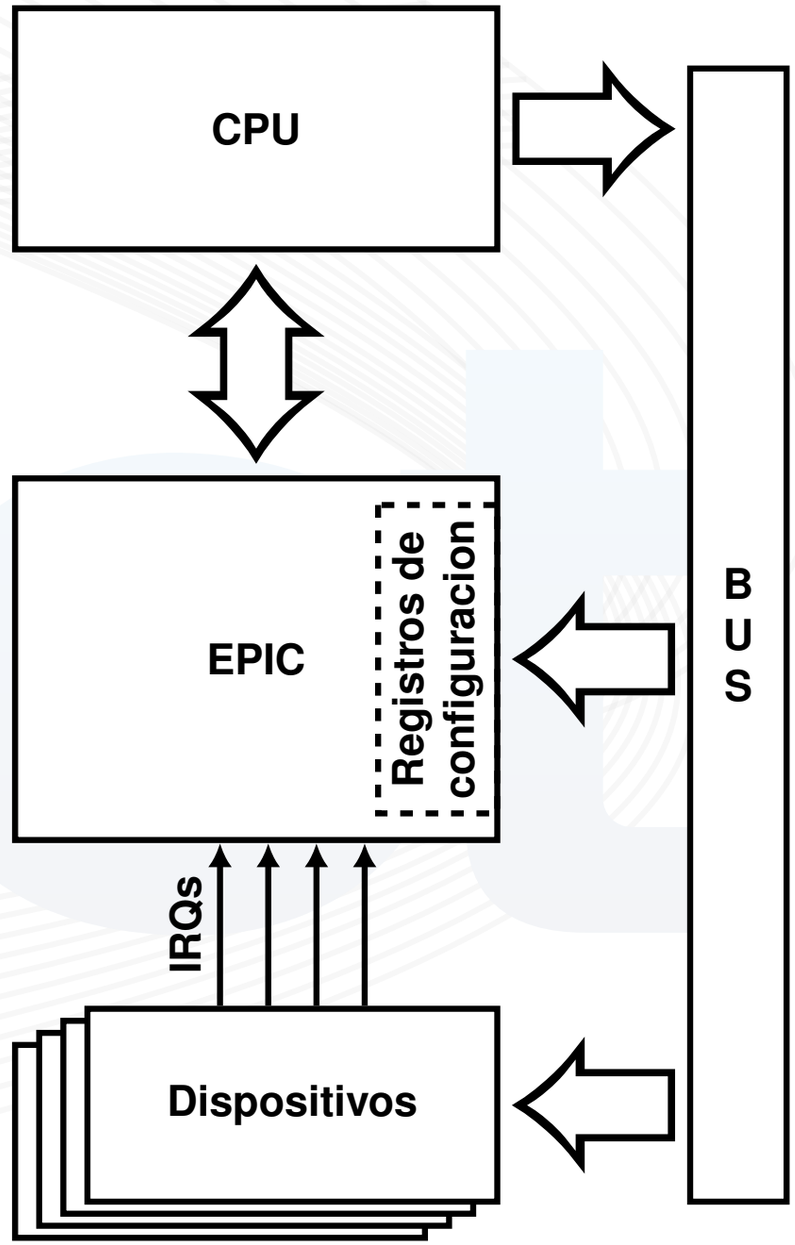
- Tradicionalmente al terminar una interrupción se recupera el estado y se retorna al programa principal
  - Si al retornar hay una interrupción pendiente, entonces se comienza la atención es esta interrupción en ese mismo instante
  - Como resultado se recupera el estado para, inmediatamente, volver a almacenar ese mismo estado sin cambios
- El tail-chain consiste en evaluar si hay una segunda interrupción pendiente al terminar la rutina de servicio y antes recuperar el estado del programa principal
  - En este caso se ejecuta la rutina de la interrupción pendiente justo a continuación de la rutina que acaba de terminar

Un ejemplo de aplicación

- Se utiliza un número de 8 bits para las prioridades
  - La mayor prioridad corresponde al valor 0xFF y la menor al valor 0x00
- Solo los 3 bits mas significativos definen el anidamiento
  - Se pueden anidar hasta 8 interrupciones como máximo
- Dispositivos iguales con rutinas de servicio de 100 T
  - Durante los primeros 10T no se pueden anidar llamados
  - DA: 0x62, DB: 0x14, DC: 0x26, DE: 0x54, DF: 0x17
- En T0 pide DC, en T15 pide DB, en T30 pide DE, T33 pide DA y en T80 pide DF
  - Dibuje una linea de tiempo con la secuencia de pedidos y atenciones

Controlador externo de interrupciones

- Se puede delegar toda la gestión de las interrupciones a un EPIC, un controlador externo de interrupciones
- El EPIC tiene suficientes lineas IRQ para todos los dispositivos y retransmite el pedido al CPU solo se cumplen las condiciones adecuadas
- Tambien es responsable de identificar que pedido se está atendiendo
- Este controlador es tambien un dispositivo que tiene sus propios registros de control y estado
- Mediante estos registros se pueden configurar la prioridad de cada línea entrante IRQ, consultar si hay un pedido pendiente, enmascararlo y hasta borrarlo



La interrupción no enmascarable

- NMI es una interrupción especial que siempre debe ser atendida
  - En general no puede ser enmascarada
  - En general tiene la mayor prioridad
- Suelen utilizarse para gestionar errores críticos de hardware
  - Errores de paridad en memoria
  - Corrupción de datos en los buses

Reset

- Cuando se alimenta el procesador todos los registros tienen valores indeterminados
  - Registros, memoria volátil y puertos de entrada/salida
- Es necesario asegurar un valor inicial de algunos registros y puertos de entrada/salida
  - Se utilizan señales de reset asincrónicas en los flip-flops
  - Se espera una cantidad de ciclos de reloj para asegurar la estabilidad en las tensiones y frecuencias de los osciladores
- Una de las tareas más importantes es asignar un valor inicial al PC para comenzar a ejecutar las rutinas de configuración del sistema (boot)

Espera pasiva

- Hasta el momento la espera de un evento se realiza en un lazo donde se encuesta una variable
- Pero si el sistema utiliza interrupciones se puede diseñar un sistema basado en eventos
- Para esto se define una instrucción WFI que suspende la ejecución del procesador hasta que ocurra una interrupción
- Esta instrucción reemplaza a todo el lazo de espera pero con un menor consumo de energía
  - Se inhibe el reloj en varios módulos del procesador

Gestión de errores en un procesador

- ¿Que ocurre con el procesador cuando ocurre un error durante la ejecución del programa?
- Lo ideal sería ejecutar una rutina que evalúe el error y tome las acciones correspondientes (no todos los errores son fatales)
- Para llamar a una rutina es necesario conocer el punto de entrada, es decir la dirección de la primera instrucción
- Como el error puede ocurrir en cualquier momento ¿a que dirección de memoria debería saltar el procesador?
- Una opción sería utilizar una dirección de memoria fija, pero esto es muy poco flexible
- Otra alternativa es utilizar un registro especial para indicarle al procesador donde se ubica la rutina para la gestión de errores

Definiciones

- Se denomina Excepción a una condición inusual que ocurre durante la ejecución de una instrucción
  - Internas (instrucción inválida)
  - Externas (acceso a memoria no implementada)
- Se denomina Trap a la transferencia de control sincrónica a la rutina de servicio encargada de gestionar una excepción o una interrupción
  - El proceso es similar al de un llamado a un subrutina, no se puede predecir cuando ocurrirá
  - Si es posible, se resuelve el problema y el programa principal continua la ejecución en forma transparente

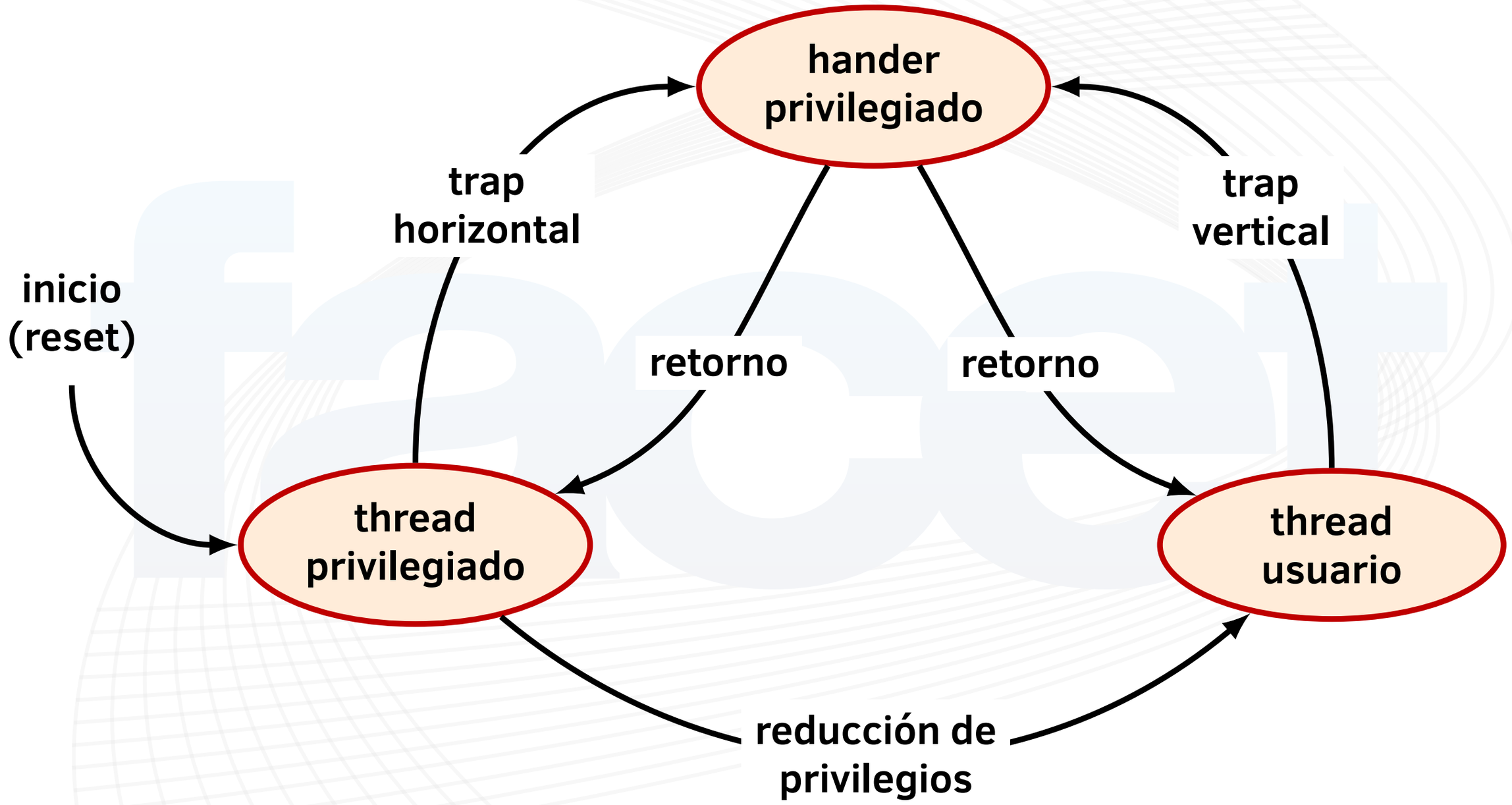
Sistema operativo

- El programa de aplicación de un sistema embebido se puede dividir en cuatro grandes secciones:
  - Rutinas de inicialización
  - Rutinas de gestión de errores
  - Rutinas para controlar entradas/salidas
  - Rutinas para resolver el problema de aplicación concreto
- Las tres primeras tienen mas relación con el entorno de ejecución que con el problema a resolver
- Se pueden separar y reutilizar para resolver diferentes problemas en el mismo entorno de ejecución
- Constituyen el núcleo de un Sistema Operativo y se convierten en un supervisor del programa de aplicación

Niveles de privilegio

- Para funcionar correctamente el programa supervisor necesita privilegios sobre el programa de usuario
- Esto le permite "defenderse" de programas maliciosos y/o errores
- El procesador inicia con el mayor nivel de privilegio lo que permite al software ejecutar cualquier operación
  - Se hace por hardware durante el reset del procesador
- Después ejecutar la rutina de inicialización y antes de iniciar el programa de aplicación se ejecuta una operación para disminuir los privilegios
  - El procesador debe tener un registro de estado donde se almacena, entre otras cosas, el nivel de privilegio actual
- Ahora el programa de aplicación está obligado a pedirle al supervisor que ejecute las operaciones restringidas

Niveles de privilegio



Terminología en los niveles de privilegio

- Se denomina thread a un programa secuencial normal (puede incluir llamados a subrutinas y lazos)
- Se denomina handler a una rutina de servicio que tiene por función atender una excepción
- Se denomina trap horizontal a un cambio de contexto que no implica la elevación de los privilegios, por ejemplo de un thread privilegiado a un handler
- Se denomina trap vertical a un cambio de contexto que eleva los privilegios, por ejemplo de un thread de usuario a un handler

Llamada al sistema

- Siempre que existen niveles de privilegio, hay una instrucción especifica para pedir servicios al sistema operativo
- Similar a un llamado a subrutina pero en realidad es una excepción voluntaria y provoca un trap vertical
  - No es necesario conocer el punto de entrada
  - Se produce una elevación de privilegios
- Como es provocada por el programa se pueden pasar argumentos por registros
  - Permite definir un conjunto de servicios que el supervisor ofrece al programa de usuario
  - Simplifican las tareas del programa de usuario
  - Permiten al supervisor compartir y gestionar los recursos

Conclusiones

- El mecanismo de interrupciones permite simplificar las esperas y mejorar la eficiencia del proceso
- El cambio de contexto, o trap, debe ser transparente para el programa de usuario
  - Hay un overhead al guardar y recuperar el estado
- Se utilizan prioridades para resolver los problemas de simultaneidad y de anidamiento
- Tambien permiten, junto con los niveles de privilegio, que el sistema operativo conviva y supervise a los programas de aplicación