Preguntas orientadoras

Describa brevemente los diferentes perfiles de familias de microprocesadores/microcontroladores de ARM. Explique alguna de sus diferencias características.

Cortex A:

Son procesadores de alto rendimiento optimizados para aplicaciones que emplean un sistema operativo de propósito general en sistemas embebidos de alta performance.

Su denominación "A" proviene de *Application*, se pueden encontrar en dispositivos como celulares o tables.

Se destaca la optimización para ejecutar diversas aplicaciones al mismo tiempo a costa de una disminución del tiempo de respuesta de las mismas, aspecto secundario en dispositivo de usuario.

Cortex R:

Son procesadores orientados a sistemas de tiempo real donde prima la necesidad de implementar soluciones con requerimientos temporales estrictos.

En los sistemas de tiempo real la respuesta a eventos o estímulos debe ser en un tiempo acotado y preestablecido comportándose de forma determinística.

Su denominación "R" proviene de *Real Time*, se pueden encontrar en sistemas críticos, como dispositivos médicos o sistemas de operación automovilísticos.

Cortex M:

Son procesadores orientados a dispositivos de consumo masivo y sistemas embebidos compactos. Son procesadores de uso general diseñados para alta densidad de código y con gran cantidad de periféricos. Su denominación "M" proviene de *Microcontrollers*, se pueden encontrar en dispositivos como celulares o tables.

Cortex M

2. ¿Por qué se dice que el set de instrucciones Thumb permite mayor densidad de código? Explique

El set de instrucciones Thumb está compuesto por instrucciones de 16 bits con funciones específicas. Se dice que permite mayor densidad de código en comparación de un set de instrucciones de 32 bits exclusivamente. En muchas operaciones simples no resulta necesario emplear muchas instrucciones, un set de 16 bits es suficiente y permite optimizar la cantidad de código que se debe almacenar.

En la familia Corte-M3/4 incorpora el set de instrucciones Thumb-2 que permite instrucciones de 16 y 32 bits mejorado la optimización desde el punto de vista de la densidad del código, eficiencia y performance.

3. ¿Qué entiende por arquitectura load-store? ¿Qué tipo de instrucciones no posee este tipo de arquitectura?

Una arquitectura *load-store* significa que para realizar una modificación de un dato almacenado debe cargarse previamente en un registro, procesarse y luego volver a escribirse en memoria utilizando una serie de operaciones separadas.

Por ejemplo, para incrementar un valor almacenado en una memoria SRAM, el procesador necesita usar una instrucción para leer el datos desde la SRAM y colocarlo en un registro interno, una segunda instrucción para incrementar el valor del registro y finalmente, una tercera instrucción para escribir el valor modificado en la posición de memoria donde se encontraba.

Dentro del set de instrucciones del dispositivo no existen instrucciones que permitan directamente modificar una valores del mapa memoria.

4. ¿Cómo es el mapa de memoria de la familia?

En la familia de procesadores Cortex-M el mapa de memoria esta compuesto por un ancho de palabra de 32 bits y puede contener hasta 4 Gbytes de espacio, limite impuesto por la cantidad de direcciones posibles que se pueden acceder con un 32 bits de direccionamiento ($2^{32} = 4.29 Gbits$).

El mapa de memoria se encuentra particionado en diferentes secciones donde se alojan (*"mapean"*) todos los componentes del sistema, como memoria flash, memoria SRAM, periféricos, etc. Exceptuando restricciones especificas sobre algunas zonas de memoria es posible acceder a todos los registros mediante un mismo bus de datos y set de instrucciones, característica que permite acceder fácilmente al contenido o registro del dispositivo empleando punteros en lenguaje C.

La partición del mapa de memoria está determinada por el fabricante de cada microntrolador que contiene un procesador ARM y debe consultarse en la hoja de datos del dispositivo.

6. Describa los diferentes modos de privilegio y operación del Cortex M, sus relaciones y como se conmuta de uno al otro. Describa un ejemplo en el que se pasa del modo privilegiado a no privilegiado y nuevamente a privilegiado

La familia de procesadores Cortex-M3/4 poseen dos modos de operación, modo privilegiado y no privilegiado, también llamado modo usuario.

El modo de funcionamiento privilegiado se caracteriza porque permite el acceso a todo el mapa de memoria por parte de la/s aplicación/es del usuario. En contraposición, en el modo de operación de usuario existen áreas determinadas de memoria protegidas que no son posibles acceder durante la ejecución del programa.

Una aplicación típica del uso de los modos de usuario, se da al emplear un sistema operativo. Es deseable que las tareas o aplicación del usuario no accedan a los componentes del sistema operativo ubicandolo en un área protegida de memoria solo accesible en el modo de privilegiado.

7. ¿Qué se entiende por modelo de registros ortogonal? Dé un ejemplo

8. ¿Qué ventajas presenta el uso de intrucciones de ejecución condicional (IT)? Dé un ejemplo

9. Describa brevemente las excepciones más prioritarias (reset, NMI, Hardfault).

10. Describa las funciones principales de la pila. ¿Cómo resuelve la arquitectura el llamado a funciones y su retorno?

11. Describa la secuencia de reset del microprocesador.

12. ¿Qué entiende por "core peripherals"? ¿Qué diferencia existe entre estos y el resto de los periféricos?

13. ¿Cómo se implementan las prioridades de las interrupciones? Dé un ejemplo

14. ¿Qué es el CMSIS? ¿Qué función cumple? ¿Quién lo provee? ¿Qué ventajas aporta?

15. Cuando ocurre una interrupción, asumiendo que está habilitada ¿Cómo opera el microprocesador para atender a la subrutina correspondiente? Explique con un ejemplo

17. ¿Cómo cambia la operación de stacking al utilizar la unidad de punto flotante?

16. Explique las características avanzadas de atención a interrupciones: tail chaining y late arrival.

17. ¿Qué es el systick? ¿Por qué puede afirmarse que su implementación favorece la portabilidad de los sistemas operativos embebidos?

18. ¿Qué funciones cumple la unidad de protección de memoria (MPU)?

La unidad *Memory Protection Unit*(MPU) consiste de un área de memoria que integra mecanismos de protección de forma que no sea accesible en todos los modos de ejecución del código de usuario.

La inclusión de la unidad MPU es una característica opcional en la familia de procesadores Cortex-M3 y M4, el fabricante del microcontrolador decide si está disponible o no. En el caso de contar con la MPU, el desarrollador posee la capacidad de dividir el espacio de memoria en diferentes regiones y definir permisos

de acceso para cada una. Cuando se produce un intento de acceso prohibido se genera una excepción (fault exception) que puede disparar la ejecución de acciones de análisis, advertencia o correctivas.

Un escenario de uso muy común resulta en sistemas embebidos que implementan un sistemas operativo, usualmente se aloja en un área de memoria con privilegios que no pueda ser accedida por las otras tareas del usuario. Otro escenario de uso es la creación de regiones de memoria de solo lectura, que permitan prevenir borrados accidentales de memoria SRAM o sobreescritura de código de instrucciones.

19. ¿Cuántas regiones pueden configurarse como máximo? ¿Qué ocurre en caso de haber solapamientos de las regiones? ¿Qué ocurre con las zonas de memoria no cubiertas por las regiones definidas?

20. ¿Para qué se suele utilizar la excepción PendSV? ¿Cómo se relaciona su uso con el resto de las excepciones? Dé un ejemplo

21. ¿Para qué se suele utilizar la excepción SVC? Expliquelo dentro de un marco de un sistema operativo embebido.