1. Describa brevemente las diferencias entre las familias de procesadores Cortex M0, M3 y

M4.

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

El set de instrucciones de los procesadores Cortex-M se llama Thumb y está basado en la tecnología Thumb-2. Soporta una mezcla entre instrucciones de 16 y 32 bits.

En la familia de procesadores ARM7 y ARM9, se debe cambiar de estado si se desea ejecutar cálculos complejos o un gran número de operaciones condicionales. Los procesadores Cortex-M, en cambio, pueden mezclar operaciones de 16 y 32 bits sin cambiar de estado. Esto permite mayor densidad de código y performance sin complejidad extra.

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

tipo de arquitectura?

La arquitectura Load-Store implica que para realizar una operación, se debe leer los datos de la memoria y escribirlos en registros del procesador. Una vez que los datos se encuentran en registros, allí se los puede procesar y de ser necesario volcar el resultado nuevamente a la memoria. Cada uno de los pasos descriptos se realizara utilizando una operación independiente.

Una representación de alto nivel puede observarse en el siguiente diagrama:

<<crear daigrama Load-Store>>

Load-Store, se diferencia de la arquitectura Register-Memory en que esta ultima puede combinar en una operación datos almacenados en memoria y registros.

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

Los procesadores Cortex-M utilizan direccionamiento de memoria de 32 bits. Esto resulta en un espacio de memoria de 4 GB. Los datos e instrucciones comparten el mismo espacio de direcciones.

Los 4 GB de espacio de memoria se subdivide en las siguientes regiones:

Code: 512 MB para el codigo del programa. Incluye la tabla de vectores.

SRAM: Normalmente utilizada para conectar SRAM (usualmente on-chip)

PERIPHERALS: Normalmente utilizada para conectar perifericos on-chip

RAM: Puede almacenar codigo y datos de programa.

DEVICES: Contiene dos slots de 512MB (1 GB total). Se utiliza para conectar periféricos off-chip

SYSTEM: Contiene varias partes

Internal Private Peripheral Bus (PPB): Se utiliza para acceder a componentes del sistema tales como NVIC, Systick, etc y a componentes de debug. En la mayoría de los casos esta memoria solo puede ser accedida por código que se ejecute en modo privilegiado.

External Private Peripheral Bus: Se incluye para que el proveedor pueda agregar componentes de propios. Este espacio de memoria solo puede ser accedido por código ejecutándose en modo privilegiado.

Vendor-Specific Area:

Resto de la memoria: Se utiliza para componentes específicos del proveedor. Muchas veces no se utiliza.

5. ¿Qué ventajas presenta el uso de los “shadowed pointers” del PSP y el MSP?

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 priviligiado y nuevamente a privilegiado.

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

8. ¿Qué ventajas presenta el uso de instrucciones 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.

La secuencia de reset del microprocesador puede resumirse en los siguientes pasos:

1 – Despues del reset el contador de programa (PC) se actualiza con la direccion 0x00000000.

2 – El procesador lee el valor de la dirección 0x00000000 y lo carga en el MSP (Main Stack Pointer).

3 – El procesador lee la direccion del reset handler de la direccion 0x00000004

4 – El procesador “salta” a la direccion de memoria del reset handler y comienza a ejecutar las instrucciones

5 – Se invoca a la funcion main del programa con el codigo del usuario

Reset handler: Es una funcion normal escrita en C o Assembly que se encarga de inicializar el procesador y los perifericos. Ejemplo, configura el reloj. Inicializa el stack space, etc.

<< Agregar diagrama >>

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?

CMSIS es una iniciativa de ARM para proveer librerías y una API standard para la programación de los procesadores de la familia Cortex-M. Esto permite reutilizar código e incrementar su portabilidad

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?

Los procesadores Cortex-M poseen un timer integrado llamado SysTick (System Tick) que genera una excepción en intervalos regulares de tiempo.

En el contexto de un sistema operativo, se utiliza al Systick Timer para el manejo de tareas y cambio de contexto. Fuera de los sistemas operativos, se utiliza a este componente para implementar interrupciones, generar delays o medir tiempos.

El temporizador SysTick favorece la portabilidad de los sistemas operativos ya que es una característica que todos los procesadores Cortex-M poseen. Esto hace que esa porción del código del sistema operativo, pueda funcionar en todos los procesadores de la arquitectura sin modificaciones.

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

El MPU es un dispositivo programable que se utiliza para definir permisos de acceso a memoria y atributos de memoria para las distintas regiones de memoria.

Los procesadores Cortex-M3 y Cortex-M4 soportan hasta 8 regiones de memoria. Cada una de ellas tiene su direccion inicial, tamanos y configuraciones.

La MPU se utiliza para hacer a los sistemas embebidos mas robustos y seguros:

- Previenen que las tareas de aplicación corrompan la memoria de stack o de datos utilizada por otras tareas y por el kernel del sistema operativo.

- Previenen que tareas no privilegiadas accedan a ciertos perifericos que son criticos para la solidez o la seguridad del sistemas

- Se puede al espacio SRAM o RAM como no ejecutable para prevenir ataques de inyeccion de codigo.

- Se puede definir atributos de memoria como el cacheability.

Si un acceso a memoria viola los permisos definidos por la MPU, el acceso es bloqueado y se genera una excepcion.