Solución de Preguntas Orientadoras

1. Describa brevemente los diferentes perfiles de familias de microprocesadores/microcontroladores de ARM explique alguna de las diferentes características.

Entre los perfiles de la arquitectura ARM, se tienen cortex A, cortex R y cortex M.
Cortex A.- Orientado a aplicaciones de con sistemas operativos de propósito general. Optimizado para aplicaciones multitarea.
Cortex R.- Orientado a sistemas de tiempo real.
Cortex M.- Orientado al desarrollo de sistemas embebidos.

Cortex M

1. Describa brevemente las diferencias entre las familias de procesadores Cortex M0, M3 y M4.

Cortex M0.- Para implementaciones con mínimos requerimientos de procesamiento. La arquitectura de memoria es Von Neumann. Arquitectura ARMv6-M.
Cortex M3.- La arquitectura de memoria es Harvard. Arquitectura ARMv7-M.
Cortex M4.- La arquitectura de memoria es Harvard. Arquitectura ARMv7E-M. Implementación en modo opcional de la FPU (unidad de punto flotante). Presenta sets de instrucciones orientadas a aplicaciones de procesamiento digital de señales.

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

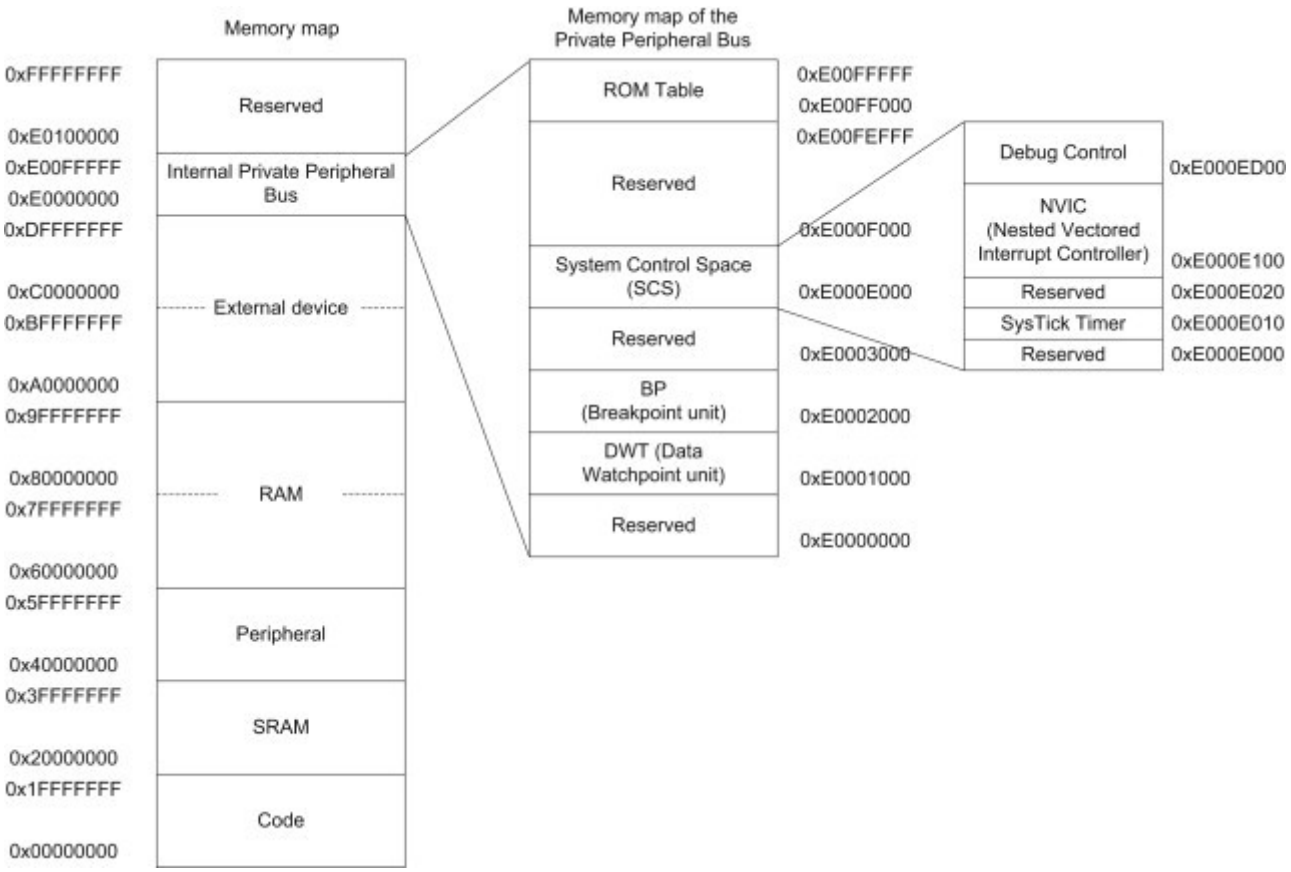
Densidad de código se refiere a cuantas instrucciones son necesarias para determinada tarea, y el espacio que ocupan estas. Por ejemplo, cuando una instrucción ocupa menos espacio para determinadas tareas, se dice que tiene mayor densidad de código; ya que se puede realizar mayor cantidad de actividades con el mismo espacio de memoria.
Para el caso de instrucciones Thumb, están alineados con 16 bits, ofreciendo una mayor densidad de código al reducir los requisitos de memoria. Adicionalmente las instrucciones Thumb-2 es un superconjunto del grupo de instrucciones Thumb (16 bits), con otras instrucciones de 16 y 32 bits.
En Thumb-2 no es necesario cambiar el procesador entre el estado Thumb y ARM. Se pueden mezclar instrucciones de 32 bits con instrucciones de 16 bits sin cambiar de estado, obteniendo un alto rendimiento sin complejidad adicional.

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

Load-Store significa que para cualquier valor que se quiera operar, primero tendrá que cargarse los valores de la memoria a los registros (Load), luego realizar la operación correspondiente y finalmente guardar los valores del registro operado en la memoria (Store).

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

El mapa de memoria es donde se detalla el espacio de memoria y las diferentes particiones que presenta. Este mapa se muestra en la siguiente figura, donde se observa la región del código del programa, SRAM, periféricos, etc.



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

En los sistemas operativas permite switchear los punteros del stack para códigos privilegiados y no privilegiados.

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.

En los modos de privilegio:
Existen 2 modos, el privilegiado y no privilegiado. Siendo la diferencia principal el acceso sin restricciones a los espacios de memoria disponibles.
En los modos de operación:
Modo Thread.- Se da cuando se ejecuta un “Exception Handler”. Tiene nivel de acceso privilegiado o no privilegiado.
Modo handler.- Se da cuando se ejecuta normalmente del código de la aplicación. Tiene nivel de acceso privilegiado.

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

Los set de instrucciones funcionan para manejar cualquier registro. No hay operaciones que funcionen para ciertos registros únicamente.
Ejemplo:
Se quiere realizar una operación de adición y guardarla en un registro.

```
ADD     R0, R1, R2      // r0 = r1 + r2
```
El modelo de registro ortogonal indica que esta suma se podría haber guardado en cualquier otro registro como R3, R5 o R7.

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

Las instrucciones IT(If Then) pueden generar de manera seguida de 1 a 4 instrucciones de ejecución condicional. En la mayoría de ocasiones las instrucciones TI ayuda a mejorar significativamente el rendimiento del código, esto comparando con las instrucciones salto condicional que necesitan normalmente 3 ciclos del procesador para volver a llenar el pipeline del procesador.
Ejemplo:

```
CMP     R0, R1      // Compara R0 con R1
ITE     EQ          //
ADDEQ   R0, 1       // Si R0 = R1 ejecuta, R0 = R0 + 1
                    // Si no es igual no realiza la operación
                    // pero igual consume ciclos de reloj
SUBNE   R0, 1       // Si R0 != R1 ejecuta, R0 = R0 -1
                    // Si es igual no realiza la operación
                    // pero igual consume ciclos de reloj
```

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

Reset.- Prioridad -3, es la prioridad mas alta . Se da cuando se inicia el sistema.
NMI (Interrupción no enmascarable).- Prioridad -2.
Hardfault.- Prioriodad -1. Se da cuando hay un fallo en el sistema y no se ha configurado alguna otra excepción de fallo.

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.

Tail chaining.- Cuando ocurre una excepción, pero el sistema ya esta atendiendo otra excepción. El procesador termina de atender la excepción actual, y en vez de volver al modo thread para luego atender a la excepción pendiente, no sale del handler y atiende inmediatamente a la excepción encolada. De esta manera el tiempo entre los manejadores de interrupciones se optimiza.

Late arrival.- Cuando ocurre una excepcion, el procesador acepta la solicitud de excepción e inicia la operación de stacking. Si antes del vector fecth se produce una excepción de mayor prioridad, el procesador atenderá la excepción de mayor prioridad.

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)?

Entre sus funciones se tienen:
- Prevenir el acceso de aplicaciones a zonas de memoria (stacks) de otras aplicaciones o del kernel.
- Prevenir el acceso de aplicaciones a periféricos sin los permisos adecuados.
- Evitar la ejecución de código desde zonas no permitidas.

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

Es una excepción que se dispara por software, tiene la prioridad más baja asegurándose que todas las demás de tareas de procesamiento de interrupciones se ejecuten. Es usado en operaciones de cambio de contexto en los sistemas operativos.
La excepción resuelve el problema cuando se esta ejecutando solicitud de interrupción(IRQ) antes de la excepción sysTick, evitando el retraso del cambio de contexto. Ya que si el sistema operativo decide que necesita un cambio de contexto, establece el estado pendiente de PendSV y lleva a cabo el cambio de contexto dentro de le excepción PendSV.

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

SVC(SuperVisor Call) es una excepción proporciona por ARM Cortex – M. Se puede utilizar para realizar operaciones privilegiadas. Normalmente una aplica corre en un nivel sin privilegios, por lo que los recursos de hardware y ciertas regiones regiones de memoria están protegidos, con lo que si se tiene la necesidad de estos recursos, normalmente no se podría acceder. En este caso el acceso a los recursos es posible a través de SVC, lanzando una instrucción SVC y atendiendo así la excepción SVC, teniendo así acceso privilegiado a los recursos.

ISA

1. ¿Qué son los sufijos y para qué se los utiliza? Dé un ejemplo.

Los sufijos complementan la información proporcionada por ciertas instrucciones. Al ser un sufijo esta ubicado inmediatamente después del nombre de la instrucción.
Entre los sufijos usados en el lenguaje ensamblador del cortex M se encuentran: los sufijos condicionales(EQ, NE, CS), el sufijo 'S'(actualiza el APSR) , sufijos para especificar el uso de variables a 8, 16 o 32 bits.
Ejemplo:
Utilización del sufijo condicional EQ.

```
CMP     R1, 10      // compara el valor de registro R1 con 10
                    // Actualiza Z = 1, si R1 == 10, sino Z = 0
B EQ    oper1       // Si Z = 1 salta a oper1, sino sigue con oper2:
oper2: …
oper1: …
```

2. ¿Para que se utiliza el sufijo 's'? Dé un ejemplo.

Significa que la operación a realizarse actualizará los flags de estado (actualización del APSR – *Application Program Status Register*), según el resultado de la operación. Entre los flags se tiene al flag zero (Z), flag negative (N), flag carry (C), flag overflow (V).
Ejemplo:

```
MOV     R1, 1       // R1 tiene el valor de 1
SUB S   R1, 1       //  Resta 1 a R1, R1 = R1 -1
                    // Actualiza Z = 1, si R1 == 11, sino Z = 0
BNE     oper1       // Si Z = 0 salta a oper1, sino sigue con oper2:
oper2: …
oper1: …
```

3. ¿Que utilidad tiene la implementación de aritmética saturada? Dé un ejemplo con operaciones de datos de 8 bits.

La aritmética saturada implica limitar el valor de una variable. Esto puede ser utilizado en el caso que se quiera editar una variable, ya sea sumándola con otra o amplificandola; esta operación de adición podría generar que la variable opere fuera de su rango dinámico(overflow). Es por ello de la importancia de limitar los valores a un umbral máximo y mínimo.
Ejemplo:
Sea una variable *R0 = 120:

```
LDRB    R1, [R0]        // var dentro de los valora  0 < var  < 0xFF
MUL     R1, R1, 3       //  R1 = 360, fuera del rango dinámico ERROR!
                        // Como R1 es de 8 bits el valor guardado es
```

```
                                    // R1 =104 (contando los bits menos significactivos)
USAT        R1, 8, R1               // saturamos a uint8_t R1, entonces el valor de R1 = 255
```

Al no saturar la variable se puede obtener datos no deseados para el fin especifico de alguna aplicación.

4. Describa brevemente la interfaz entre ensamblador y C ¿Cómo se reciben los argumentos de las funciones?¿Cómo se devuelve el resultado?¿Qué registros deben guardarse en la pila antes de ser modificados?.

Los argumentos en ensamblador se maneja como si recibieran 4 argumentos de una función, guardando el argumento 1 al r0 y así hasta el argumento 4 con el registro 3.
Para devolver un valor de retorno se utiliza el registro r0.

5. ¿Qué es una instrucción SIMD? ¿En que aplican y que ventajas reporta su uso? Dé un ejemplo.

SIMD(Single Instruction, Multiple Data) son instrucciones que procesan múltiples datos en una única operación. Operando al doble o 4 veces más rápido, permitiendo realizar operaciones en paralelo entre datos de 16 o bits. Estas instrucciones permiten manejar el procesamiento digital de señales digitales de manera mas eficiente en relación a tareas de tiempo real.