Arquitectura de Microprocesadores
Carrera de Especialización en Sistemas Embebidos
Universidad de Buenos Aires
Alumno: Ing Hernán Gomez Molino

Preguntas orientadoras

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

La arquitectura ARM cuenta con tres familias: Cortex A, Cortex R y Cortex M.
La familia Cortex A (Application) está orientada a la implementación de sistemas operativos en sistemas embebidos de alta performance. Un ejemplo de estos podrían ser teléfonos celulares.
La familia Cortex R (Real time)está orientada a la implementación de sistemas operativos real time, donde es preciso mantener baja latencia en los procesos. Ejemplos de uso podrían ser todo tipo de procesos críticos, como computadoras automotrices, sistemas médicos, etc.
La familia Cotex M (Microcontrollers) engloba a los microcontroladores que abarcan una gama amplia de rendimiento. Incluye desde los de bajo consumo y menor poder de procesamiento (M0), hasta los de mayores prestaciones (unidad de punto flotante o unidad de protección de memoria, por ejemplo). Su uso se puede ver en todo tipo de sistemas embebidos.

Cortex M

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

La principal diferencia es que el Cortex M0 tiene arquitectura Von Neumann (ARM V6) mientras que M3 y M4 tienen arquitectura Harvard (ARM V7).
Muchas de las características son opcionales por lo que dependen del fabricante, pero en general las diferencias mas salientes son:
SysTickTimer – lo tienen todos los M3 y M4, pero es opcional para M0.
Memory protection unit – Es opcional para M3 y M4, pero no se incluye en M0 (si es opcional para M0+).
Las extensiones de división por hardware y matemática saturada no se encuentran disponibles para M0 y si para M3 y M4. Y las de DSP y FPU se incluyen en M4 (FPU opcional).

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

El set de instrucciones Thumb permite realizar las mismas operaciones que la instrucción original (en las instrucciones soportadas) pero con formato de 16 bits. Al hacerlo por hardware, no afecta la performance, pero consume menos memoria. A esto se refieren como mayor densidad de código.

Arquitectura de Microprocesadores
Carrera de Especialización en Sistemas Embebidos
Universidad de Buenos Aires
Alumno: Ing Hernán Gomez Molino

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

Se le dice load-store a las arquitecturas que realizan las instrucciones lógicas y aritméticas solo sobre registros como operandos, tanto de entrada como de salida, obligando al programador a transferir los datos de memoria a los registros de entrada (load) y el registro de salida a memoria (store). No posee instrucciones que operen desde o hacia memoria directamente.

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

La familia Cortex tiene un mapa de memoria plano direccionable de 4 GB.
Está dividido en 6 zonas. La zona de direcciones más baja es la zona de código, de 500 MB; seguida de la zona de RAM estática, también de 500 MB ; le sigue la zona de RAM de 1 GB; a continuación una zona de 1 GB para dispositivos y los 500 MB más altos son para sistema o específicos para el fabricante. En esta última zona es donde se encuentran las facilidades para debug, NVIC, etc.

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

En la implementación de OSs (o RTOSs) mejora la robustez, ya que si una aplicación tiene problemas de pila, el kernel y las otras aplicaciones no se ven afectadas.
Un OS puede usar la protección de memoria (MPU) para definir un área de memoria para la pila de una aplicación. En caso de salirse de este área (overflow) se producirá una excepción de manejo de memoria y evitará la sobrescritura de la memoria.

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.

Cortex M tiene dos estados de operación:

- Estado Debug: cuando esta detenido (halt) y tiene un debugger conectado.
- Estado Thumb: cuando está ejecutando código Thumb (normal). En este, hay dos modos de operación:
  - Modo Handler: cuando se está ejecutando un handler de excepción, o una rutina de atención de interrupción. En este modo el procesador tiene nivel privilegiado de acceso.
  - Modo Thread: Cuando se ejecuta código normal de aplicaciones. Puede o no tener nivel de acceso privilegiado, según se configure en el registro de control.

Un ejemplo podría ser:
el scheduler de un sistema operativo no cooperativo (preemptive) se ejecuta en modo privilegiado y le cede el procesador a una tarea común (no

privilegiada). Al vencer su tiempo, se produce una interrupción que en su rutina de atención le devuelve el control al scheduler.

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

Se le llama ortogonal al modelo de registros en los que pueden ser aplicadas todas las instrucciones del set sin distinción de cual de estos se trate. También se le llama así a su set de instrucciones.
Un ejemplo podría ser la instrucción ADD que suma un operando al registro destino, que puede ser cualquiera de ellos.

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

La ejecución condicional permite crear bloques de hasta 4 instrucciones que se ejecutarán o no dependiendo de las condiciones (flags) de la operación anterior al bloque. El bloque se identifica con una instrucción compuesta por una **I** al inicio (IF) y siguiendo con **T** (then) o **E** (else) dependiendo si la instrucción debe ejecutarse al cumplirse o no la condición, que se escribe inmediatamente después del bloque descripto. Se pueden especificar hasta 4 letras (T o E) despues de la I combinadas de la manera que se desee. Por ejemplo:

```
CMP R1, #1    @ en base a esta comparación
ITEET eq,     @ se ejecutarán de la sig. manera:
“instr #1”    @ inst #1 si es igual
“instr #2”    @ inst #2 si no es igual
“instr #3”    @ inst #3 si no es igual
“instr #4”    @ inst #4 si es igual
```

Esta característica puede ahorrar muchas comparaciones y mejorar el rendimiento del sistema.

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

- Reset: es la excepción con la prioridad más alta existente y su ocurrencia implica el reset total del core (nivel de prioridad -3).
- NMI (Non-Maskable Interrupt): es la siguiente en prioridad y responde a una linea de entrada y no es deshabilitable (nivel de prioridad -2).
- HardFault: Ocurre bajo ciertas condiciones erróneas de sistema. Por ejemplo errores de bus, violación de protección de memoria, etc (nivel de prioridad -1).

Arquitectura de Microprocesadores
Carrera de Especialización en Sistemas Embebidos
Universidad de Buenos Aires
Alumno: Ing Hernán Gomez Molino

A continuación viene las excepciones programables con nivel 0 o superior (descendente).

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

La pila es una área de memoria LIFO cuya función principal de la pila es la conservación del contexto de ejecución ante cambios.
El llamado a funciones se logra almacenando los parámetros y el valor de retorno en los registros (calling convention).

11. Describa la secuencia de reset del microprocesador.

Después de un reset el procesador lee de las dos primeras palabras de la memoria la dirección del vector de reset. Luego lee de las dos primeras palabras de este vector el valor del main stack pointer (MSP) y el valor del PC (program counter), al cargar estos el procesador queda ubicado para ejecutar el reset handler.

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

Son lo periféricos que forman parte del procesador (por ejemplo el NVIC). La diferencia es que son esenciales para el funcionamiento del procesador y se encuentran mapeados en el área de memoria perteneciente al sistema.

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

Las prioridades, por default, bajan su prioridad a medida que suben su número de identificación . La IRQ_0 es la de mayor prioridad. El programador puede cambiar el nivel de prioridad escribiendo en los registros de prioridad de interrupción. El nivel seteado en este registro tiene precedencia sobre el nivel por default (hardware).

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

CMSIS significa Cortex Microcontroller Software Interface Standard y es una biblioteca de Hardware Abstraction Level (HAL), provista por ARM. Tiene la ventaja que su implementación es obligatoria para todos los fabricantes y por lo tanto asegura un nivel de portabilidad entre fabricantes.

Arquitectura de Microprocesadores
Carrera de Especialización en Sistemas Embebidos
Universidad de Buenos Aires
Alumno: Ing Hernán Gomez Molino

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?

En los controladores M4F (con FPU) existe un banco de registros adicional que se utilizan con la FPU. Para evitar que el cambio de contexto sea muy lento se incorpora el modo lazy stacking, que evita guardar en la pila estos registros si el handler de la interrupción o la tarea interrumpida no usan la FPU.

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

Tail chaining significa que si existiera más de una interrupción pendiente, se ejecutarán una a continuación de la otra sin volver al contexto interrumpido entre handlers.

Late arrival significa que si durante la fase de entrada de una interrupción (stacking y fetch) se dispara otra interrupción de prioridad más alta, se ejecutará esta última primero y luego la de menor prioridad.

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

SysTick es un timer que genera un pedido de interrupción cíclico. Esto lo usa el OS para los cambios de contexto. En bare metal se puede usar para temporización o disparar rutinas cíclicas.

Al basarse en este timer, un OS puede utilizarse en toda la familia Cortex. Por ello se dice que favorece su portabilidad.

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

La MPU proporciona todos los mecanismos necesarios para implementar la protección de memoria. Esto significa que se divide la memoria en segmento y se restringe su acceso según prioridades.

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?

Se pueden configurar 8 regiones. Ante el solapamiento de regiones, los atributos del área solapada corresponderán a los de la región de número más alto. Las áreas no incluidas en ninguna región componen el área de background , que puede ser accedida por las rutinas privilegiadas y las no privilegiadas a las que se les haya seteado un bit específico del registro de control.

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 utiliza el OS cuando surge una interrupción de menor prioridad que la que se está atendiendo. Antes de volver al contexto interrumpido se ejecutan las interrupciones pendientes.

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

SVC es un pedido de interrupción por software que se suele utilizar para pasar de una rutina no privilegiada a una privilegiada. Como no puede hacerse directamente, se llama al handler (modo handler) y este cambia al contexto privilegiado.

ISA

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

Los sufijos son modificadores a las instrucciones y se utilizan para especificar la forma en que deben ejecutarse.
Por ejemplo, MOVH realiza lo mismo que la instrucción MOV pero solo para dos bytes (halfword).

2. ¿Para qué se utiliza el sufijo ‘s’? Dé un ejemplo

Se utiliza para indicarle al procesador que debe actualizar los flags de acuerdo a la operación.
Por ejemplo para comparar puedo restar utilizando SUBS (sub + sufijo s) para luego chequear el flag Z (cero), que si esta seteado indica que los operandos eran iguales (resta igual a 0).

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

Su utilidad es evitar que una señal muestreada tenga un overflow al exceder los valores limite y esto provoque que una señal de valores altos se convierta

instantáneamente en una de valor bajo o negativo, o viceversa. Si bien una señal saturada esta distorsionada, es muy preferible a una señal con overflow. Por ejemplo, si una señal entera no signada de 8 bits llega al valor 255 y se le suma uno más pasa a poner todos sus bits en 0 (y carry en 1) por que esta tratando de escribir el valor 256. la señal máxima se convierte en 0 repentinamente al sumarle 1.

4. Describa brevemente la interfaz entre assembler 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?

En el programa en C se declaran las funciones en assembler como externas. En el código asm declara la función con “.global”.

Los argumentos se almacenan en los registros en el orden que se declararon antes del llamado y el retorno se aloja en R0 (calling convention). deben resguardarse en la pila los registros con número superior a 3 que se utilizarán. Los registros del 0 al 3 se colocan en la pila automáticamente.

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