Arquitectura de Microprocesadores
Carrera de Especialización en Sistemas Embebidos
Universidad de Buenos Aires

Preguntas orientadoras

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

Los diferentes perfiles de familias, se basan principalmente en sus distintas funcionalidades, capacidades y prestaciones.

Las familias Cortex-A esta diseñadas principalmente para aplicaciones de alto rendimiento utilizadas en sistemas operativos para embebidos.
Los Cortex-R se utilizan en sistemas de tiempo real cuando se necesita sistemas baja latencia y alto procesamiento.
Los Cortex-M se utilizan para sistemas embebidos compactos que pueden correr grandes códigos y soporta programación en C.
Los Cortex-M0 son los que tienen menos funcionalidades, menos memorias, etc son mas baratos y se usan para aplicaciones donde requieran pocas funcionalidades, bajo consumo, recurso del procesador.
Los Cortex M3 al Cortex-M7 son los mas caros, son los que tienen mas funcionalidades, mas performance, mas memoria, se utilizan en microcontroladores, etc.

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

Las diferencias de los CORTEX-M:

Cortex M : Comparación arquitecturas

ARM Cortex-M	SysTick Timer	Bit-banding	Memory Protection Unit (MPU)	Tightly-Coupled Memory (TCM)	CPU cache	Memory architecture	ARM architecture
Cortex-M0[1]	Optional*	Optional[9]	No	No	No[10]	Von Neumann	ARMv6-M
Cortex-M0+[2]	Optional*	Optional[9]	Optional (8)	No	No	Von Neumann	ARMv6-M
Cortex-M1[3]	Optional	Optional	No	Optional	No	Von Neumann	ARMv6-M
Cortex-M3[4]	Yes	Optional*	Optional (8)	No	No	Harvard	ARMv7-M
Cortex-M4[5]	Yes	Optional*	Optional (8)	No	Possible[11]	Harvard	ARMv7E-M
Cortex-M7	Yes	No	Optional (8 or 16)	Optional	Optional	Harvard	ARMv7E-M

Observando la columna de SysTickTimer, vemos que en los CORTEX.-M3-M4 lo tienen incorporados:

Esta opción lo que permite, es brindar interrupciones cada un tiempo seteado, lo que nos da la posibilidad de generar interrupción para algún código que lo requiera.

La columna de Memory Protection Unit: Sirve para proteger la unidad de memoria. Lo que obliga pedir permisos para poder tener acceso a en ella. También dar prioridades.

La columna de Memory architecture vemos la Von Neumann y Harvard

Von Neumann: esta tiene un bus para comunicar con la memoria(datos) y usa el mismo bus para acceder a las instrucciones del codigo.

Hardvard: esta tiene 2 bus, uno para la memoria(datos) y otro bus para las instrucciones.

Conclusión la arquitectura Hardvard es mas rápida, consume más y es más cara que la Von Neumann.

Ultima columna es se puede observar cuales CORTEX usan ARMv6-M y cuales ARMv7-M

ARMv6-M usa las instrucciones THUMB(instrucciones de 16 bit) y algunas de THUMB-2(incorpora una instrucción condicional y son de 32bit.)

ARMv7-M usa las instrucciones THUMB-2. Los ARMv7E-M usan las instrucciones THUMB-2 y ademas esta preparado para procesamientos de señales digitales(DSP=permite ejecutar instrucciones en menos ciclos de reloj).

Los CORTEX M0 y M0+ tienen instrucciones con resolucion de 32 bit. Pero M3 y M4 tiene mas instrucciones de multiplicación en hardware con resolución de 64 bit.

Los CORTEX M0 y M0+ no tienen matemática Saturada y los CORTEX-M3 y M4 tiene. El resultado de la operación satura sobre una variable cuando sobrepasa el máximo valor.

Los CORTEX-M4 tienen unidad de punto flotante en su hardware, esto hace que las operaciones sean mucho mas rápidas.

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

Permite mayor densidad de código porque implementa instrucciones de 16 Bits que ocupan menos memoria que las instrucciones de 32 Bits. El set de instrucciones de Thumb, son de 16 bit y son un subconjunto de las instrucciones de 32 bit de ARM. Con la introducción del set de instrucciones de Thumb 2, ahora es posible manejar el procesamiento en un solo estado de operación (no es necesario alternar entre los dos estados). Instrucciones Thumb 2 (una de las características más importantes), utiliza en forma conjunta instrucciones de 32 y 16 bits logrando alta densidad de código, alta eficiencia y potente además de fácil de usar.

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

En la arquitectura load-store(cargar y almacenar), como lo dice su nombre deben cargarse(load) un dato de la memoria en un registro, luego utilizar las instrucciones que necesita el código para procesarlo y luego almacenar(store) en memoria. Por ejemplo, para incrementar un valor de datos almacenado en SRAM, el procesador necesita usar una instrucción para leer los datos de SRAM y colocarlos en un registro dentro del procesador, una segunda instrucción para incrementar el valor del registro y luego una tercera. instrucciones para volver a escribir el valor en la memoria. Los detalles de los registros dentro de los procesadores se conocen comúnmente como modelo de programador. Esta arquitectura son ademas, las encargadas de ejecutar las instrucciones de acceso a la memoria RAM, tanto para lectura como escritura.

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

Esta particionada en regiones, su capacidad es de 4gb. Los sectores se ubican: sistema, external device, external RAM, periféricos, SDRAM, código, componentes internos del procesador, etc.

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

El shadowed pointer es porque usa 2 stack pointers SP para funciones de OS Kernel y manejadores de interrupciones como MSP(main stack pointer) y PSP(process stack pointer) y otro para las tareas que se ejecutan en las aplicaciones SP y PSP.
Cuando inicia el procesador lo hace utilizando el SP apuntando MSP, para cambiar de MSP a PSP se hace con un bit de control.
Toda variable declarada en una función se guarda en el stack. Entonces se puede acceder a funciones de interrupciones el SP. Cuando se trabaja en aplicaciones de bare metal, el PSP se puede ignorar y se trabaja simplemente con el MSP. Pero,cuando se implementan soluciones que utilizan un RTOS o un OS Kernel, el uso del "shadowed pointer" es muy práctico, ya que permite separar los SP del modo thread priviligiado (OS Kernel) y del modo handler (excepciones) por un lado y los del modo thread usuario por otro lado. De esta forma, si llegase a fallar el PSP, el MSP seguiría funcionando.

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.

Los procesadores Cortex M3/M4 tiene dos niveles de privilegio y dos modos de operación:

Handler: al ejecutar un controlador de excepciones como un Servicio de Interrupción Rutina (ISR). Cuando está en modo Handler, el procesador siempre tiene privilegios nivel de acceso.

Thread: cuando se ejecuta el código de aplicación normal, el procesador puede estar en un nivel de acceso privilegiado o en un nivel de acceso no privilegiado. Esto está controlado por un registro especial llamado "CONTROL". NO ES POSIBLE regresar al nivel privilegiado por software desde nivel no privilegiado en modo usuario, solo es posible desde el modo handler.

Los niveles de privilegio sirven para protejer los accesos a regiones críticas de memoria para evitar un posible problema o daño. Pero los manejadores de excepciones sólo pueden hacerlo en estado privilegiado. Es el Handler de una interrupción quién puede regresar al modo privilegiado.

Un ejemplo podria ser cuando se pasa desde un modo privilegiado a no privilegiado y de regreso es el usado por las llamadas(SVC), utilizando el registro de control al sistema de un Sistema Operativo.

Un ejemplo podria ser cuando se pasas del nivel privilegiado en modo thread al nivel no privilegiado en modo thread (ejemplo SVC), luego cuando se produce una excepcion o interrupcion estamos en modo Handler, de aquí se puede modificar el registro control y poder pasar al nivel privilegiado del modo thread.

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

Por ejemplo el conjunto de instrucciones del 8086/8088, la mayoría de estas instrucciones están disponibles en el modo de 32 bits, ellas simplemente operaban en registros y valores de 32 bits (EAX, EBX, etc) en vez de 16 bits (AX, BX, etc). Estas estaban asociados a su función(era mas intuitivo).

Ahora en particular no hay nombre, sino números, que la llamamos arquitectura ortogonal, es decir toda operación o conjunto de instrucción es ortogonal cuando se puede utilizar cualquier modo de direccionamiento en cualquier instrucción. Esto hace que el procesamiento sea más complejo pero aporta una mayor facilidad de programación.

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).

La familia Cortex M posee un controlador de interrupciones programable denominado “nested vectored interrupt controller” (NVIC). Formalmente, el NVIC puede manejar hasta 255 excepciones (de las cuales las interrupciones son un caso particular):
• Reset (IRQ 1)
• Excepciones del core (IRQ2-15)
• Hasta 240 fuentes de interrupción externas (IRQ16-255)
Las primeras tres fuentes de IRQ tienen prioridades fijas y el resto pueden programarse hasta en 128 niveles de prioridad (no necesariamente implementados).
Las tres primeras prioridades son:
Reset: es la excepción con mas alta prioridad en el vector de interruciones y corresponde a las propias de la arquitectura ARM. Es decir, esta excepción está definida por la propia arquitectura. Esta excepción hace que el microcontrolador se reinicie incondicionalmente.

NMI(Non-Maskable Interrupt), esta interrupcion o excepcion puede ser generada desde un periferico o desde fuentes externas. NMI es usualmente generado desde perifericos como el watchdog timer or Brown-Out Detector (BOD). El resto de las excepciones son desde el nucleo del procesador. Pueden haber interrupciones tambien generadas usando software.

Hard Fault: todas las condiciones de falla o errores seran interrupciones, si el contolador de errores correspondiente no esta habilitado. No hay necesidad de habilitar el controlador o manejador de Hard Fault. Esto esta siempre esta habilitado y tieneuna prioridad fija de -1. El nombre por default de controlador de excepcion es Hard Fault(definido por CMSIS-Core) es HardFault_Handler.

El Hard Fault Status Register permite monitorear las fuentes de fallas. A continuación, se detallan los flags que es posible monitorear:

DEBUGEVT: Indica que el evento de depuración desencadena un fallo grave.

FORCED: Indica que se tomó una falla permanente debido a una falla de bus, una falla de administración de memoria o una falla de uso.

VECTBL: Indica que la falla es causada por una una falla del vector fetch(encuentra direcciones en el stack).

EXTERNAL: indica que el evento de depuración es causado por una señal externa .

VCATCH: indica que el evento de depuración es causado por una vector catch(lo usa el debugger), una función programable que permite que el procesador se detenga automáticamente cuando ingresa cierto tipo de excepción del sistema incluido el reinicio.

DWTTRAP: Indica que el evento de depuración es causado por un punto de observación.

BKPT: Indica que el evento de depuración es causado por un punto de interrupción.

HALTED: Indica que el procesador se detuvo debido a una solicitud del depurador (incluido un solo paso).

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

Las funciones principales de la pila “STACK” son para pasar datos a funciones o subrutinas, para guardar variables locales, para guardar el estado del procesador y de los registros de proposito general cuando ocurre una interrupcion. Tambien para guardar temporalmente el valor de registros previo a su reutilizacion.

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?

Capa de acceso al Core Peripherals: Definiciones de nombres, definiciones de direcciones y funciones auxiliares para acceder a los registros del núcleo y a los periféricos del núcleo. Este es específico del procesador y es proporcionado por ARM.
El resto de los periféricos, varios dispositivos del mismo proveedor pueden usar el mismo conjunto de archivos. Estos son específicos del dispositivos o es específico del proveedor y es opcional. Es decir se puede programar los periféricos directamente.

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 fue desarrollado por ARM para permitir que los proveedores de software y microcontroladores utilicen una infraestructura de software consistente para desarrollar soluciones de software para microcontroladores Cortex -M. Muchos productos de software para microcontroladores Cortex-M son compatibles con CMSIS. ARM trabajó con varios proveedores de microcontroladores, proveedores de herramientas y proveedores de soluciones de software para desarrollar CMSIS, un marco de software trabajo que cubre la mayoría de los procesadores Cortex-M y los productos de microcontrolador Cortex-M.
Tiene varias funciones entre ellas:
El controlador de interrupciones del sistema (NVIC)- Control del Systic Timer.- Drivers genéricos para los diferentes periféricos.- Proporciona una API para la implementación de sistemas operativos en tiempo real. - Funciones de acceso especial para introducción de código ensamblador. Etc.

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

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

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

Tail chaining tiene la caracteristica que cuando ocuren dos interrupciones anidades de igual prioridad….

Late arrival ocurre cuando se esta conmutanfo a una IRQ y ocurre otra de mayor prioridad….

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

Los procesadores Cortex -M tienen un pequeño temporizador integrado llamado SysTick temporizador. Está integrado como parte del NVIC y puede generar el SysTick excepción (tipo de excepción #15). El temporizador SysTick es un temporizador de decremento simple de 24 bits, y puede ejecutarse en la frecuencia de reloj del procesador o desde una frecuencia de reloj de referencia(normalmente una fuente de reloj en chip).
En los sistemas operativos modernos, se necesita una interrupción periódica para garantizar que el sistema operativo kernel puede invocar regularmente; por ejemplo, para la gestión de tareas y el cambio de contexto. Esto permite que un procesador maneje diferentes tareas en diferentes intervalos de tiempo.
El diseño del procesador también garantiza que las tareas de la aplicación se ejecuten a un nivel sin privilegios, no puede deshabilitar este temporizador; de lo contrario, estas tareas podrían desactivar el temporizador SysTick y bloquear todo el sistema.
La razón para tener el temporizador dentro del procesador es ayudar al software a transportar portabilidad. Dado que todos los procesadores Cortex-M tienen el mismo temporizador SysTick, un sistema operativo escrito para un microcontrolador Cortex-M3/M4 se puede reutilizar en otro Cortex-M3/M4 microcontroladores.
El temporizador SysTick Si no necesita un sistema operativo integrado en su aplicación, el temporizador SysTick se puede utilizado como un periférico de temporizador simple para la generación periódica de interrupciones, generación de retrasos, o medición de tiempo.

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

Sirve para proteger la unidad de memoria. Lo que obliga pedir permisos para poder tener acceso a ella, es decir evitar acceder a áreas criticas o consideradas bajo ciertas prioridades. Por parte del sistema o por el usurario.

20. ¿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?

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

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

ISA
1. ¿Qué son los sufijos y para qué se los utiliza? Dé un ejemplo
2. ¿Para qué se utiliza el sufijo ‘s’? Dé un ejemplo
3. ¿Qué utilidad tiene la implementación de instrucciones de aritmética saturada? Dé un ejemplo con operaciones con datos de 8 bits.
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?
5. ¿Qué es una instrucción SIMD? ¿En qué se aplican y que ventajas reporta su uso? Dé un ejemplo.
2