Advanced Micro-controllers

Agenda

- ARM Cortex-M Architecture
  - Registers
  - States
  - Modes
  - Access Levels
  - Status, Control and Mask registers
  - Exceptions and Interrupts
  - Exception handling
  - NVIC behaviour

ARM Cortex-M Architecture

States

- No ARM state
- Thumb state and Debug state
- Debug state -- Step by step debug execution
- Thumb state -- Regular execution
  - xPSR: T=1 (Thumb state)

Modes

- Handler mode and Thread mode
- Handler mode --
  - Exception/Interrupt handler (except Reset Handler) --> Priv access level
- Thread mode --
  - Bare metal

        - Reset Handler + main/user code --> Priv access level
    - Embedded OS
        - Reset Handler + main code + OS kernel --> Priv access level
        - User tasks/programs --> Non-Priv access level

Access Levels

- Privileged and Non-Privileged access levels
- Privileged level
    - Handlers + Reset Handler + main code + OS kernel
- Non-Privileged level
    - User tasks (only in Embedded OS env)
- Non-Privileged level Limitations
    - Control & Mask registers not accessible
    - NVIC registers not accessible
    - Memory regions not accessible
    - Few instructions not allowed
- controlled by CONTROL[bit 0]
    - 0 --> Privileged level
    - 1 --> Non-Privileged level

Registers

- r0 to r12 --> general purpose registers
- r15 --> program counter
    - address of next instruction to be fetched.
    - program counter is incremented by 2 or 4 bytes (after each instruction fetch)
- r14 --> link register
    - Return address during function call.
    - EXC_RETURN (special value) during exception.
- r13 --> stack pointer
    - only register that can be banked in/out.

    - msp = main stack pointer
    - psp = program stack pointer
    - Bare metal programming
        - Only MSP is used.
    - Embedded OS programming
        - Handlers + main code + OS kernel --> MSP is used
        - User tasks --> PSP is used
    - controlled by CONTROL[bit 1].
        - 0 = MSP
        - 1 = PSP

Status, Control and Mask registers

- CONTROL register
    - bit0 --> Access Level --> 0=Priv, 1=Non-Priv
    - bit1 --> Stack Pointer --> 0=MSP, 1=PSP
- PRIMASK: Disable interrupts and exceptions
    - bit0 --> 1=Disable, 0=Enable
- FAULTMASK: Disable interrupts, exceptions, and faults
    - bit0 --> 1=Disable, 0=Enable
- BASEPRI: Disable interrupts/exceptions of equal or lower than given priority
    - Number of bits depends on manufacturer (max 8 bits)
- xPSR register -- Status register
    - bit0-9 --> Interrupt/Exception number (when handler is executing)
    - bit16-19 --> GE bits (SIMD instructions)
    - bit24 --> T bit (T=1 Thumb state)
    - bit28-31 --> NZCV ALU flags
    - bit27 -- Q bit (Saturated Maths)
    - bit10-15,25-26 -- IT bits (If-Then instruction bits)
- xPSR is accessible in program via APSR, EPSR, IPSR registers.
    - APSR - Application PSR - NZCV, Q, GE
    - EPSR - Execution PSR - IT, T

    - IPSR - Interrupt PSR - Interrupt/Exception Number

Exceptions, Faults and Interrupts

- Exception/Interrupt vector table
    - 0 -- Initial address for MSP
    - 1-15 -- Exceptions and Faults
    - 16-255 -- Peripheral Interrupts e.g. Timer, ExtInt, Uart, ADC, SPI, I2C, CAN, ...
        - LPC1768 has 35 Peripherals
        - STM32F407 has 82 Peripherals
- Exceptions and Faults
    - 1 - Reset (Priority = -3)
    - 2 - NMI (Priority = -2)
    - 3 - Hard Fault (Priority = -1)
    - 4 - MemManage Fault
    - 5 - Bus Fault
    - 6 - Usage Fault
    - 11 - SVC (Software Interrupt)
    - 12 - Debug Monitor
    - 14 - PendSV
    - 15 - SysTick Timer
- The EVT stores addresses of the handlers.

APCS (ARM function calling convention)

- APCS --> ARM Procedure Calling Standard
    - Function arguments: r0-r3
    - Function return value: r0

Exception handling

- When exception occures, following steps done in core...
    - Stacking

    - Vector Fetch
    - Register update
- Stacking
    - Values of 8 registers are pushed on the stack.
        - r0-r3, r12, pc, lr, xpsr
    - Stack grows downwards (new SP = cur SP - 32).
        - cur SP -->
        - xpsr (push 2)
        - pc (push 1)
        - lr (push 8)
        - r12 (push 7)
        - r3 (push 6)
        - r2 (push 5)
        - r1 (push 4)
        - r0 (push 3)
        - new SP -->
- Vector Fetch
    - Get handler address from vector table
    - This is done parallel to stacking due to Harvard architecture.
        - Separate data memory and bus --> SRAM
        - Separate program memory and bus --> Flash
- Register update
    - New SP = SP - 32
    - New PC = Handler address
    - New LR = EXC_RETURN
    - New xPSR = Exception/Interrupt Number
    - Few NVIC registers
- EXC_RETURN
    - Special value stored in LR when Exception occurrs.
    - Special value 0xFFFFFFFX (Higher 28-bits = 1)
        - bit0 - Return Processor state (T) -- always 1.

        - bit1 - Reserved -- always 0.
        - bit2 - Return Stack -- 0=MSP, 1=PSP.
        - bit3 - Return Mode -- 0=Handler, 1=Thread.
    - Before Exception: Main code or Embedded OS kernel
        - bit 3210 = 1001 i.e. LR=0xFFFFFFF9
        - Return back to Thread mode with MSP.
    - Before Exception: Handler (Lower Priority Interrupt/Exception)
        - bit 3210 = 0001 i.e. LR=0xFFFFFFF1
        - Return back to Handler mode with MSP.
    - Before Exception: User Task (in Embedded OS)
        - bit 3210 = 1101 i.e. LR=0xFFFFFFFD
        - Return back to Thread mode with PSP.
- Now handler Execution begins (after stacking + vector fetch + register update).
    - Handler is always executed in HANDLER mode with PRIVILEGED access level.
    - When EXC_RETURN value is loaded in PC, Exception return sequence is initiated.
        - BX lr @ here LR EXC_RETURN is copied into PC
        - POP {pc} @ if lr is pushed on stack and then popped in pc
        - LDR pc, =0xFFFFFFFX @const special value loaded in PC
- When Exception returns, following steps done by core...
    - Unstacking
        - r0-r3, r12, pc, lr, xPSR are restored from stack.
        - PC will be now address of next instruction.
    - Register update
        - New SP = SP + 32
        - Few NVIC registers

NVIC registers

- Refer slides

Interrupt Priority

- To be continued tomorrow