Advanced Micro-controllers

Agenda

- Revision - ARM Cortex-M Architecture
- ARM Assembly Language
  - Logical operations: MVN, AND, ORR, EOR, BIC, TST
  - Decision control: Conditional Execution (ARM), Conditional Branching, If-Then instruction
  - ~~Iteration/Loop~~
  - ~~Barrel shifter: Instructions and Inline~~
  - ~~Global variables~~
  - ~~Arrays~~
  - ~~Functions~~

ARM Cortex-M Architecture

NVIC registers

- ISER --> Interrupt Set Enable Register
  - To enable Interrupt in NVIC
  - When enabled NVIC will receive interrupt from peripheral and send it to core
  - One bit for each peripheral
    - LPC1768 -- 35 peripherals -- 35 bits -- ISER0 (32-bit) + ISER1 (3-bits)
    - STM32F407 -- 82 peripherals -- 82 bits -- ISER0 (32-bit) + ISER1 (32-bits) + ISER2 (18-bits)
  - CMSIS helper function: NVIC_EnableIRQ(nIRQ);
- ICER --> Interrupt Clear Enable Register
  - To disable Interrupt in NVIC
  - When disabled NVIC will not receive interrupt from peripheral
  - One bit for each peripheral
    - LPC1768 -- 35 peripherals -- 35 bits -- ICER0 (32-bit) + ICER1 (3-bits)
    - STM32F407 -- 82 peripherals -- 82 bits -- ICER0 (32-bit) + ICER1 (32-bits) + ICER2 (18-bits)

    - CMSIS helper function: NVIC_DisableIRQ(nIRQ);
- ISPR --> Interrupt Set Pending Register
    - When Interrupt is received from peripheral, corresponding bit in ISPR is set.
    - When Interrupt is sent to core and core begin execution of handler, this is bit auto cleared.
    - One bit for each peripheral
        - LPC1768 -- 35 peripherals -- 35 bits -- ISPR0 (32-bit) + ISPR1 (3-bits)
        - STM32F407 -- 82 peripherals -- 82 bits -- ISPR0 (32-bit) + ISPR1 (32-bits) + ISPR2 (18-bits)
    - Before Interrupt is sent to core, it can be cleared using ICPR Register. If cleared, Interrupt will not be sent to core.
- ICPR --> Interrupt Clear Pending Register
    - Before Interrupt is sent to core, it can be cleared using ICPR Register. If cleared, Interrupt will not be sent to core.
    - One bit for each peripheral
        - LPC1768 -- 35 peripherals -- 35 bits -- ICPR0 (32-bit) + ICPR1 (3-bits)
        - STM32F407 -- 82 peripherals -- 82 bits -- ICPR0 (32-bit) + ICPR1 (32-bits) + ICPR2 (18-bits)
- IABR --> Interrupt Active Bit Register
    - When core begin execution of handler, corresponding bit in IABR is set. When handler execution is completed, that is cleared.
    - This register is read-only.
    - One bit for each peripheral
        - LPC1768 -- 35 peripherals -- 35 bits -- IABR0 (32-bit) + IABR1 (3-bits)
        - STM32F407 -- 82 peripherals -- 82 bits -- IABR0 (32-bit) + IABR1 (32-bits) + IABR2 (18-bits)
- STIR --> Software Triggered Interrupt Register.
    - When Interrupt number is written into this register, it emulates interrupt from the device.
- VTOR --> Vector Table Offset Register
    - Usually vector table starts from address 0x00000000 (in most controller).
    - Cortex-M3/M4 allows to keep vector table in other memory locations in flash or RAM.
    - In that case, VTOR represents base address of vector table.

Interrupt Priority

- Three exceptions have fixed priority.
    - Reset (-3)
    - NMI (-2)
    - Hard Fault (-1)

- Rest of interrupts/exceptions priority is programmable.
- Cortex-M architecture allows max 8-bit priority.
- Higher number of priority bits will increase complexity of internal circuit and number of transisters.
- Manufactures may choose to implement 3-bit, 4-bit, 5-bit or more priorities as per design.
    - LPC1768 -- 5 bit priority (35 peripherals)
    - STM32F407 -- 4 bit priority (82 peripherals)
- IPR register in NVIC stores priority of each interrupt. For each peripheral, 8 bits are reserved.
    - LPC1768 -- 35 peripherals -- 35 x 8 bits = 35 bytes -- IPR0, IPR1, ..., IPR8.
    - STM32F407 -- 82 peripherals -- 82 x 8 bits = 82 bytes -- IPR0, IPR1, ..., IPR20.
- 5 bit priority -- 7 6 5 4 3 x x x
- 4 bit priority -- 7 6 5 4 x x x x
- Interrupt priority bit are further divided into two parts
    - Preemption priority / Group priority
        - Higher Preemption priority Interrupt can preempt/pause execution of lower Preemption priority Interrupt ISR.
        - Equal or lower Preemption priority Interrupt can never preempt/pause execution of current Preemption priority Interrupt ISR.
    - Sub priority
        - If multiple Interrupts with same Preemption priority are pending, then higher sub-priority Interrupt's ISR is executed first.
    - Number of bits in Preemption and Sub-priority can be configured by programmer in AIRCR register. Example: LPC1768 5-bit priority
        - If PRIGROUP in AIRCR is set to 4, then bits 7-5 (3 bits) are used for Preemption priority and bits 4-3 (2 bits) are used Sub-priority.
        - This can be done using CMSIS helper function NVIC_SetGrouping(4);
- Interrupt priority for each Interrupt can be configured in corresponding IPR register. This can be done using CMSIS helper function NVIC_SetPriority(nIRQ, priority);

Cortex-M3 Assembly Language Programming

Linker Script

- Written in .ld file and contains Instructions for Linker.
- It contains addresses of Flash and SRAM.

```
MEMORY
{
    FLASH (rx) : ORIGIN = 0x00000000, LENGTH = 0x00010000
    SRAM (rwx) : ORIGIN = 0x20000000, LENGTH = 0x00002000
}
```

- It contains layout of executable file and addresses at which code/data will be kept.

```
SECTIONS
{
    .text : {
        * (.vectors);
        * (.text);
        etext = .;
    } > FLASH

    .data : {
        sdata = .;
        * (.data);
        edata = .;
    } > SRAM AT> FLASH

    .bss : {
        sbss = .;
        * (.bss);
        ebss = .;
    } > SRAM

    .rodata : {
        * (.rodata);
    } > FLASH
}
```

Basic code

- Execution start from reset handler i.e. _start
- _start:
  - initialize data section (load from flash into sram)
  - clear bss section
  - call main
- main:
  - user defined code
  - return from function