Advanced Micro-controllers - ARM

Agenda

- Q & A
- Memory/Cache
- Toolchains
- C - Compilation and Execution
- Process

Q & A

1. When to use volatile in embedded applications?
    - Common case
        - If a variable is used in a loop and it is getting modified in an ISR (out of current execution context), then such variable should be declared volatile.
        - Example:

```
// global variable
volatile int flag = 0;
```

```
// main()
while(flag == 0)
    ;
```

```
// isr()
flag = 1;
```

- Special case
  - To declare memory-mapped registers. Typically done by manufacturers.
  - In Cortex architecture these registers are usually declared in struct.
  - Example:

```
// from LPC17xx.h (NXP semiconductors)
typedef struct
{
__I  uint32_t I2STAT;
__IO uint32_t I2DAT;
__O  uint32_t I2CONCLR;
// ...
} LPC_I2C_TypeDef;

#define LPC_I2C0_BASE 0x4001C000
#define LPC_I2C0  ((LPC_I2C_TypeDef*)LPC_I2C0_BASE)
```

  - Here, __I, __O, __IO are defined as macros (core_cm3.h).

```
#define __IO    volatile
#define __O     volatile
#define __I     const volatile
```

2. What is process?
   - Process is Program in Execution.

Toolchain

- Chain of tools required for development (compilation, linking, debugging, etc).
  - ar, cpp, as, cc, ld, gcc, size, nm, gdb, objdump, readelf, etc.
- ARM toolchain path in STM32CubeIDE:
  - /opt/st/stm32cubeide_1.12.1/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for-stm32.10.3-2021.10.linux64_1.0.200.202301161003/tools/
    - share/
      - doc/ --> toolchain docs
    - bin/
      - arm-none-eabi-as, arm-none-eabi-gcc, arm-none-eabi-ld, arm-none-eabi-objdump, etc.
    - lib/
      - *.a --> statically linked libraries e.g. libgcc.a, ...
    - arm-none-eabi/
      - bin/ --> gcc, as, ld, ...
      - lib/ --> libc.a, libm.a, ...
      - include/ --> stdio.h, math.h, ...
      - share/ --> docs
- ARM Cross-compilation toolchains
  - arm-none-eabi-
    - This is used for bare metal (with no OS) ARM development.
  - arm-linux-gnueabi-
    - This is used for ARM based Linux application development.
    - The floating point calculations are emulated in software.
  - arm-linux-gnueabihf-
    - This is used for ARM based Linux application development.
    - The floating point calculations are in hardware.
    - hf = hard-float

Native compilation vs Cross compilation

- Compiling/building programs for the current CPU architecture is referred as "Native compilation".

    - Example: on PC (x86), compiling for PC (x86) programs.
    - Example: on RPi (arm), compiling for RPi (arm) programs.
- Compiling/building programs for the other (than current) CPU architecture is referred as "Cross compilation".
    - Example; on PC (x86), compiling for STM32 (arm) programs.