Memory mapped IO vs IO mapped IO

- MM: IO device registers are mapped to same address space as of memory.
- IM: IO device registers are mapped to separate IO address space.
- MM: Same instructions for IO as of memory e.g. in ARM LDR/STR.
- IM: Different instructions for IO and memory e.g. in x86 MOV of memory and IN/OUT for IO.
- MM: Differention in IO addresses and memory addresses is just range of addresses.
- IM: Special control signal is used to differetiate IO or memory bus. e.g. IO/M' in x86.

Instruction Pipeline vs Instruction Queue

Instruction Pipeline

- 2-stage pipeline (e.g. AVR)
    - Fetch
    - Execute
- 3-stage pipeline (e.g. ARM7, ARM-CM3)
    - Fetch
    - Decode
    - Execute

Pipeline Hazards

- Control hazards
- Data hazards
- Structural hazards

Word

- bit, byte, word
    - bit -- minimal storage -- 1 & 0.
    - byte -- unit of storage for memory (memory width) -- 8 bits
    - word -- amount of data that can be processed by CPU at a time.
        - It is same as ALU bits.
        - word size = size of registers
        - 8051, AVR -- 8-bit processors -- word size=8 bits
        - 80186 -- 16-bit processor -- word size=16 bits
            - Intel/MS consider word = 16 bits
        - ARM -- 32-bit processor -- word size=32 bits
            - word = 32 bits

            - half-word = 16 bits
            - byte = 8 bits
        - Data bus size is same as word size (for CPU).

ISA vs Core

- ISA: Instruction execution engine
    - Usually instruction set remains same for ISA.
- Core: ISA + Buses + Caches + MMU/MPU
    - Multiple cores per ISA.

ISA

- v1, v2, v3 -- Outdated
- ARMv4: System mode, Halfword & Byte support
    - e.g. StrongARM (Intel)
- ARMv4T: Thumb state
    - e.g. ARM7, ARM9
    - ARM state
        - Each instruction is of 4 bytes.
        - PC increments by 4 bytes.
        - Lower code density.
    - Thumb state
        - Each instruction is of 2 bytes.
        - PC increments by 2 bytes.
        - Thumb instructions are internally converted into ARM instructions using Thumb decoder.
        - Pros
            - Higher code density.
            - Code density is increased by 70%.
        - Cons/Limitations
            - Not all functionalities are available in Thumb state (e.g. related to exception handling).
            - Not all registers are accessible in all instructions (R0-R7 are accessible in most instructions).
            - Conditional execution of all instructions is not supported.
- ARM v5TE:
    - e.g. FreeScale, ARM10, ARM9, ...
    - Enhanced DSP instructions

        - Saturated Math
        - Multiply & Accumulate
        - Count Leading Zeros

- ARM v5TEJ:
    - Jazzelle core
        - C/C++ code --> Compiler --> Machine code (arch specific) --> Executed by processor
        - Java code --> Compiler --> Byte code --> JVM (JIT) --> Machine code (arch specific) --> Executed by processor
        - Byte code --> Each instruction is of one byte.
        - Jazzelle core execute Java Byte code into hardware itself.
    - Jazzelle state
        - To run java code ARM should be in Jazzelle state.
        - In this case, each instruction is of 1 byte.
        - PC is incremented by 4. Four instructions are fetched at a time.
- ARMv6
    - e.g. ARM11, ...
    - SIMD: Single Instruction Multiple Data
        - e.g. QADD8 r0, r1, r2
    - Multi-processing: Multi-core
    - Unaliagned memory accesss: The data can be accessed from RAM even if it is not word aligned (i.e. not multiple of 4).
    - Thumb-2:
        - Mixed instruction set: Thumb + ARM
        - Most of instructions are of 16-bits, but few are 32-bits.
        - It can execute in Thumb state (no need of ARM state).
        - Exception handling is possible in Thumb-2.
        - Few instructions are added for Conditional execution (IF-THEN), ...
- ARMv7
    - It is called as ARM-Cortex.
    - Three profiles
        - v7A - CortexA - Application (usually OS/Linux installed)
        - v7R - CortexR - Realtime (usually RTOS installed)
        - v7M - CortexM - Microcontroller (usually no OS - bare metal)
    - Application profile
        - e.g. Cortex-A8, Cortex-A9, ...
        - Similar to traditional ARM (i.e. ARM9, ...)
        - MMU support.
        - Typically used with OS/Linux installed on it.
        - Has ARM as well as Thumb state.
        - 7 modes and 37 registers.

        - Complex and Costly.
    - Microcontroller profile
        - e.g. Cortex-M0 (v6), Cortex-M3, Cortex-M4.
        - Different ARM architecture (than traditional).
        - Typically used for bare metal. Can install minimal OS or RTOS.
        - Has only Thumb state (with Thumb-2 instruction set).
        - 2 modes and 17 + status registers.
        - Simplified and Cheaper.
- ARMv8
    - 64-bit ARM architecture

```c
#include <stdio.h>
struct test {
    int a; // 4
    char b; // 1
    short c; // 2
    float d; // 4
    char e; // 1
};
int main() {
    printf("%d\n", sizeof(struct test));
    return 0;
}
```

ARM7 architecture

- ARM7TDMI
    - T: Thumb (v4T)
    - D: On chip debug (single step execution on processor)
    - M: Enhanced multiplier (64-bit result)
    - I: Embedded ICE (In Circuit Emulation -- to get state of registers)

Programmer's model

- Registers
- Modes
- Exceptions

Modes

- Privileged modes
    - Supervisor
    - IRQ
    - FIQ
    - Undef
    - Abort

    - System
- Non-Privileged mode
    - User
- Privileged mode
    - All (relevant) registers are accessible.
    - All instructions are allowed.
    - Exception handling is possible.
    - Usually system software/OS runs in these modes.
- Non-Privileged mode
    - Few registers are not accessible (e.g. CPSR).
    - Few instructions are not allowed (e.g. MSR, MRS).
    - Usually user program runs in this mode.