Instruction Pipeline vs Instruction Queue

- Q: Temp storage for the instruction before it is executed.
- P: Multiple independent but cascaded processing stages process instructions simultaneously.
- Q: instruction is only stored not processed.
- P: each stage is doing some processing on instruction.
- Q: typically used in CISC arch like x86.
- P: typically used in RISC arch like AVR, PIC, ARM, ..

Instruction Pipeline

- Instruction pipeline contains multiple independent units that process instruction partially. All these units can execute simultaneously and thus multiple instructions can be processed parallely. This is called as "instruction level parallelism".
- Example:
  - 2-stage: Fetch --> Execute
    - AVR & PIC
  - 3-stage: Fetch --> Decode --> Execute
    - ARM7 & ARM-CM3

Pipeline Problems/Hazards

Control Hazards

- Due to jump or conditional jump, instructions already fetched into pipeline are of no use and hence pipeline need to be cleared and re-filled from new (jumped) address. Obviously next instruction is not effectively completed in single CPU cycle.
- To overcome this problem, CPU can use "branch prediction" i.e. CPU predicts whether condition in current jump instruction will be true or not. Depending on that it fetch the next instructions.
- Some advanced CPUs also use "brach speculation". In this next instructions are not only fetched but also processed and their result is stored temporarily. If prediction is correct, the result is utilized (for faster execution); otherwise the result is discarded.

Data Hazards

- For some instructions more CPU cycles are needed for the execution eg. mul. If next instruction execution depends on result of previous instruction, then pipeline is stalled (paused) until current instruction is completed. Due to this effective number of CPU cycles for execution of instruction will increase.

Structural Hazards

- Due to CPU design restrictions some instructions cannot be completed immediately. e.g. If CPU is designed to write only one result into regr and two results were produced in an instruction. In this case, pipeline is stalled to complete the current instruction.

What is AMBA?

- AMBA = Advanced Microcontroller Bus Architecture
- Defined by ARM, it's a set of protocols for internal communication in SoCs.
- AMBA versions:
    - AHB: High-performance bus for main system interconnect
    - APB: Simplified bus for low-bandwidth peripherals
    - AXI: Advanced system bus for high-end SoCs (used in Cortex-A, not Cortex-M4)
- AHB
    - Advanced High-performance Bus
    - Fast
    - Complex
    - CPU, RAM, Flash, DMA
- APB
    - Advanced Peripheral Bus
    - Slower
    - Simple
    - Low-speed peripherals (UART, SPI, I2C, GPIO, etc.)

STM32F407VG GPIO

- Multiplexed GPIO pins
    - GPIOA - GPIOI ports
    - Max 16 IO pins per port
    - Total 82 GPIO pins
- Input / Output types
    - push-pull or open drain
    - pull-up or pull-down registors
    - floating or analog input
- Configurable speed
- External interrupt
- GPIO registers
    - MODER: Input (0), Output (1), Analog (3) or Alt (2)
    - OTYPER: Push-pull (0) or Open-drain (1)
    - PUPDR: Pull-up (1), Pull-down (2) or none (0)
    - OSPEEDR: Low (0) to Very High (3)
    - IDR: Input
    - ODR: Output
    - BSRR: Bit Set/Reset
- GPIO programming steps
    - GPIO as output
        - Enable GPIO clock (AHB1ENR)
        - Select GPIO mode as output

        - Select GPIO speed
        - Set pull-up or pull-down resistor
        - Set output type (push-pull or open-drain)
        - Set or Clear pin

LED

LED

- 4 user LEDs are connected to PORTD
    - Pin 12 - Green
    - Pin 13 - Orange
    - Pin 14 - Red
    - Pin 15 - Blue
- All GPIO ports are connected to AHB1 bus
- To set GPIO Pin in input/output mode
    - MODER
        - 00 - input mode
        - 01 - output mode
    - 0 - 31, 29, 27, 25
    - 1 - 30, 28, 26, 24
- To set type of output (Push-Pull/Open drain)
    - OTYPER
        - 0 - push pull
        - 1 - open drain
    - 0 - 12, 13, 14, 15
- To set speed of GPIO pin
    - OSPEEDR
        - 00 - low speed
        - 01 - medium speed
        - 10 - High speed
        - 11 - very high speed
    - 0 - 24 to 31
- To activate pull up or pull down register
    - PUPDR
        - 00 - no pull up/ pull down
        - 01 - pull up
        - 10 - pull down
    - 0 - 24 to 31
- To read input from GPIO pin
    - IDR
- To write output on GPIO pin
    - ODR
    - bit 12 to 15 need to modify for LEDs
- To enable clock for GPIOD
    - AHB1ENR
        - 0 - disabled

        - 1- enabled
    - 1 - 3 bit