Microcontroller Programming and Interfacing

Contents

- ARM Microcontroller
- ARM Cortex-A
    - Architecture
- ARM Cortex-M
    - Architecture
    - Assembly language
    - Protocols
- RISC-V
    - Architecture
- C Programming -- Target board

Pre-requisites

- Embedded C Programming
    - Bitwise operators
    - Structures
    - Functions
    - Makefile
    - Volatile
    - Debugging (gdb)
- Computer Fundamentals
    - Processor Architecture
    - Interrupts & Polling

Evaluation

- Theory -- CCEE -- 40 marks (MCQ based)
- Lab -- C Programming -- STM32F407G-DISC1 -- 40 marks
- Internal -- 20 marks

Reading

- Mazidi -- AVR Microcontroller
    - Chapter 0 -- Introduction to Embedded
    - Chapter 1, 2 & 3 -- Concepts
    - Protocols -- RS232, SPI, I2C
- Slos -- ARM System Developer's Guide (Cortex-A)
- Yiu -- Definitive guide to Cortex M3/M4
- STM32 User manual
- STM32F407G-DISC1 User manual

uP vs uC

- uP: general purpose computing machine/chip.
- uC: dedicated processing chip.
- uP: contains CPU [ALU, registers], cache and/or MMU.
- uC: contains CPU, RAM, ROM, pheripherals (timers, gpio, adc, dac, spi, i2c, can, etc.) on same chip.
- uP: usually works on high clock speed and hence higher power cosumption.
- uC: usually works on lower clock speed and hence lower power cosumption.
- uP: requires more space and cost.
- uC: requires less space and cost.
- uP: system design is flexible -- designer can choose required amount of RAM, ROM & peripherals.
- uC: cannot customize a controller -- but can choose from variety of controllers as per need.

Von Neumann vs Harvard

- V: addr & data bus is common for program memory (ROM) and data memory (RAM).
- H: seperate addr & data buses for program memory (ROM) and data memory (RAM).
- V: instruction and data cannot be fetched simultaneously.
- H: instruction and data can be fetched simultaneously.
- V: op code and operands are not fetched in single cycle. So slower execution.
- H: op code and operands are fetched in single cycle. So faster execution.
- V: unified Cache (i.e. common cache for instruction & data).
- H: seperate cache of instruction & data.
- V: e.g. x86 arch
- H: e.g. AVR

RISC vs CISC

- High level languages --> Compiler --> Low level languages
- CISC arch --> Complex instruction set
  - High level language constructs (if-else, loops, ...) --> Assembly language
  - Simplified compiler code generation
  - Complex instructions (macro-instructions) --> While execution, divided into micro-instructions
  - Complex execution engine --> More number logic gates
- RISC arch --> Reduced instruction set
  - Minimal instruction set (micro-instructions).
  - Simplified execution engine --> Less number logic gates
  - Complex compiler code generation

RISC vs CISC

- R: Reduced Instruction Set Computing
- C: Complex Instruction Set Computing
- R: less number of instructions
- C: more number of instructions
- R: all instructions are micro-instructions -- cpu doesn't further divide it into smaller units.
- C: some instructions are macro-instructions -- cpu further divides it into smaller units and then execute.
- R: most of the instructions are executed in single cpu cycle.
- C: needs variable number of cycles from 1-6.
- R: instruction width is fixed i.e. most of the instructions are of same size.

- C: instruction width is variable.
- R: large number of general purpose registers.
- C: less number of general purpose registers.
- R: follows load/store pattern i.e. data must be loaded into CPU regrs from RAM before manipulating it.
- C: some instructions can directly manipulate data present in main memory.
- R: compiler design more complicated; but more efficient execution.
- C: Becauze of macro-instructions, compiler design (for high level language) is easier.
- R: typically uses instruction pipeline -- instruction level parallelism.
- C: typically uses instruction queue -- fetch next instruction.
- R: Examples - AVR, ARM, ...
- C: Examples - 8086, 8051, ...

Memory mapped IO vs IO mapped IO

- M: IO devices (SFRs) are mapped as memory addresses (in memory addr space).
- I: IO devices (SFRs) are mapped as seperate IO addresses (in diff addr space).
- M: IO regrs are accessed with same instructions as of memory e.g. MOV, LD/ST.
- I: IO regrs are accessed with special instructions e.g. IN/OUT.
- M: There is no differentiation betn mem locations and IO regrs -- except addrs.
- I: Mem locs and IO regrs are differentiated by different buses or control signal e.g. IO/M'
- M: Example: ARM.
- I: Example: x86 arch.