Microprocessors and Digital Systems

Semester 2, 2022

Dr Mark Broadmeadow

Tarang Janawalkar

Contents

1 Microcontroller Fundamentals

1.1 Computer

Definition 1.1 (Computer)**.** A computer is a digital electronic machine that can be programmed to carry out sequences of arithmetic or logical operations (computation) automatically.

Definition 1.2 (Control unit)**.** The control unit interprets the instructions and decides what actions to take.

Definition 1.3 (Arithmetic logic unit)**.** The arithmetic logic unit (ALU) performs computations required by the control unit. This store data in memory, stores program instructions in memory

microprocessor doesnt include external storage, inputs, outputs,
mircocontroller, integrated circuit, single chip that contains a CPU, memory, peripherals and is a standalone system.
on a desktop motherboard, the CPU is the microprocessor and the memory, and other peripherals are connected via other busses on the motherboard.
The QUTy uses a microcontroller called ATtiny1626, that are within a family of microcontrollers called AVRs.

1.2 ATtiny1626 Microcontroller

The ATtiny1626 microcontroller has the following features:

- CPU: AVR Core (AVRxt variant)
- Memory:
  - Flash memory (16KB) used to store program instructions in memory
  - SRAM used (2KB) to store data in memory
  - EEPROM (256B)
- Peripherals, i.e., inputs/outputs. These are implemented in hardware (part of the chip) in order to offload complexity

1.2.1 Flash Memory

- Non-volatile — memory is not lost when power is removed
- Inexpensive
- Slower than SRAM
- Can only erase in large chunks
- Typically used to store programme data
- Generally read-only. Programmed via an external tool, which is loaded once and remains static during the lifetime of the program
- Writing is slow

1.2.2 SRAM

- Volatile — memory is lost when power is removed
- Expensive
- Faster than flash memory and is used to store variables and temporary data
- Can access individual bytes (large chunk erases are not required)

1.2.3 EEPROM

- Older technology
- Expensive
- Non-volatile
- Can erase individual bytes

1.3 AVR Core

Definition 1.4 (Computer programme)**.** A computer programme is a sequence or set of instructions in a programming language for a computer to execute.

The main function of the AVR Core Central Processing Unit (CPU) is to ensure correct program execution. The CPU must, therefore, be able to access memories, perform calculations, control peripherals, and handle interrupts. Some key characteristics of the AVR Core are:

- 8-bit Reduced Instruction Set Computer (RISC)
- 32 working registers (r0 to r31)
- Program Counter (PC) — location in memory where the program is stored
- Status Register — modified by the ALU and stores key information from the ALU (i.e., whether a result is negative)
- Stack Pointer — temporary data that doesn't fit into the registers
- 8-bit core — all data, registers, and operations, operate within 8-bits

1.4 Programme Execution

At the time of reset PC = 0.

1. Fetch instruction (from memory)
2. Decode instruction (decode binary instruction)
3. Execute instruction:
    - Execute an operation

- Store data in data memory, the ALU, a register, or update the stack pointer

4. Store result
5. Update PC (move to next instruction or if instruction is longer than 1, add 2. Or move to different location in program arbitrary address k)

1.5 Instructions

- The CPU understands and can execute a limited set of instructions — ~88 unique instructions for the ATtiny1626
- Instructions are encoded in programme memory as opcodes. Most instructions are two bytes long, but some instructions are four bytes long
- The AVR Instruction Set Manual describes all of the available instructions, and how they are translated into opcodes
- Instructions fall loosely into five categories:
  - Arithmetic and logic — ALU
  - Change of flow — jumping to different sections of the code or making decisions
  - Data transfer — moving data in/out of registers, into the data space, or into RAM
  - Bit and bit-test — looking at data in registers (which bits are set or not set)
  - Control — changing what the CPU is doing

ldi - load immediate takes constat value and stores it in a register
data must be in a register to be used in an instruction
can only access r16-r31

1.6 Interacting with memory and peripherals

- The CPU interacts with both memory and peripherals via the data space
- From the perspective of the CPU, the data space is single large array of locations that can be read from, or written to, using an address
- We control peripherals by reading from, and writing to, their registers
- Each peripheral register is assigned a unique address in the data space
- When a peripheral is accessed in this manner we refer to it as being memory mapped, as we access them as if they were normal memory
- Different devices, peripherals and memory can be included in a memory map (and sometimes a device can be accessed at multiple different addresses)

1.7 Memory map

1.8 Assembly code

- The opcodes that get placed into programme memory are what we call machine code (i.e. code the machine operates on directly)
- We tend not to write machine code directly as it is:
- Not human readable
- Prone to errors (swapping a single bit can completely change the operation)
- Instead we can write instructions directly in assembly code
- We use instruction mnemonics to specify each instruction
- You have already seen some of these: ldi, add, sts, jmp …
- An assembler takes the assembly code and translates it into opcodes that can be loaded into programme memory