Block 02

Architecture of MCU, instruction set, GPIO

2023

EmLab @ FI MUNI

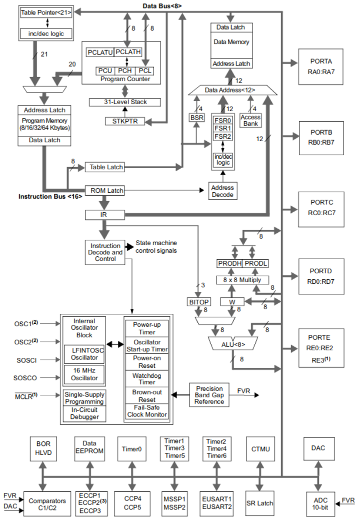
Financováno Evropskou unií
NextGenerationEU

Národní plán obnovy

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

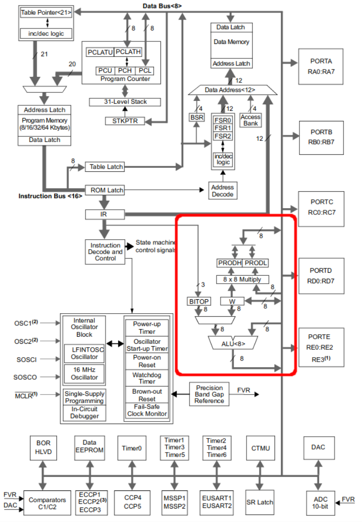
Seminar on Digital System Architecture

PIC18 core



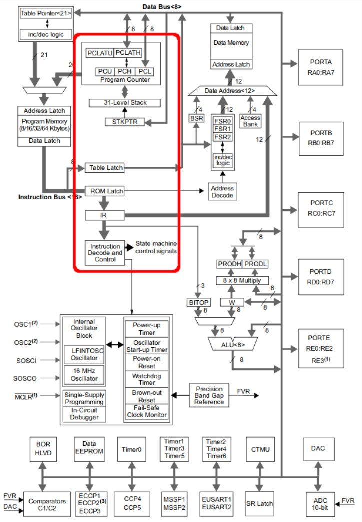
- this will be a high level overview
- detailed description in datasheet
- most of following informations can be generalized for other MCUs

ALU



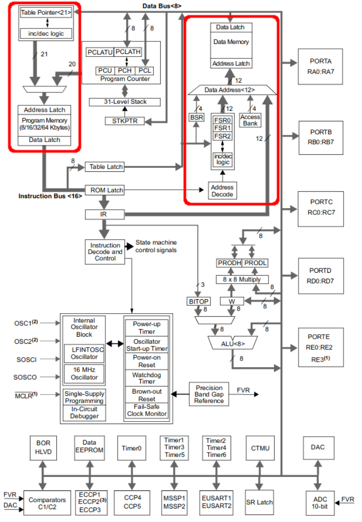
- the "core" that does most of the computational work
- consists of:
  - ALU - arithmetic-logic unit
  - WREG - working register
  - STATUS register
  - 8x8 multiplier

Control, program counter, stack



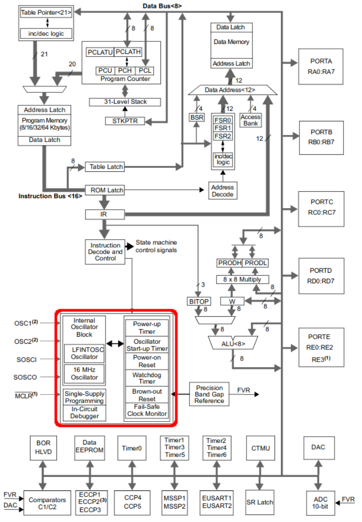
- control:
  - controls the whole CPU operation
  - it is what makes the CPU an automata
- program counter:
  - stores address of current instruction to execute
  - auto increments
  - tighly coupled to stack
- stack:
  - normal stack, just like in CPU
  - designed to only hold return address for PC
  - can insert own data
  - 31 bytes

Program and data memory



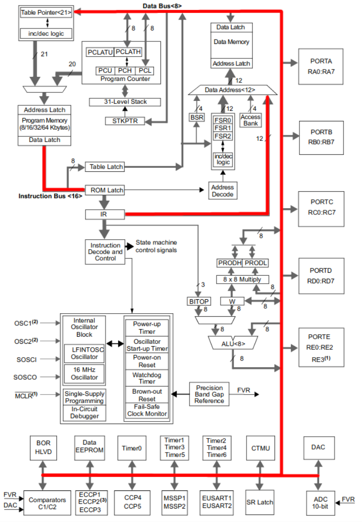
- program memory
  - flash based
  - byte reads
  - block writes and erases (64 bytes)
- data memory
  - static RAM
  - byte operations (read, write)
  - GPR vs SFR
  - EEPROM for nonvolatile storage (not memory mapped)
- addressing quite complex
- each have their own address space

Clocks



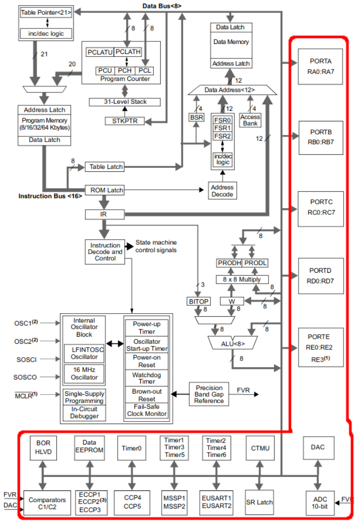
- provide all required clocks
- internal and external oscillators
- very configurable, include PLLs
- complex topic on it's own
- can clock from kHz to 64 MHz

Buses



- datapath for data, instructions
- PIC18 only has two internal buses
- 16bit for instructions
- 8bit for data

Peripherals



- the main feature of MCU
- provide external communication, timing, etc.
- each peripheral can be very complex
- controlled over memory-mapped registers
- *memory-mapped*: the registers live in data memory address space

What happens at reset

- must be well defined, MCU needs to start deterministically
- registers, peripherals usually have well defined reset state
- from reset to first instruction:
    1. registers, peripherals reset
    2. wait until clocks and such stabilize
    3. initialize program counter to 0x0 address - *reset vector*
    4. execute first instruction, located at reset vector
    5. continue execution loop as normal

PIC18 instruction set

High level overview

- instructions are the commands for our CPU
- surprisingly small set of instructions needed to implement any program
- most instructions are convenience
- PIC18 has 75 constant length instructions
- classified as RISC
- instructions can take more than 1 instruction cycle

Single instruction cycle

- each cycle divided into 4 subcycles: Q1 to Q4
- each instruction uses the four cycles differently
- usually follow *Decode, Read, Process, Write* steps
- program counter incremented every Q1
- PIC18 has two stage pipeline
    - we "prefetch" next instruction
    - we need to flush the pipeline when jump or branch occurs

Grouping

- 4 groups of instructions in PIC18
- byte-oriented - adding registers ...
- bit-oriented - flipping a single bit ...
- literal - adding a constant to a register ...
- control - calls, jumps, branches ...

Understanding/reading instructions

- all required info in datasheet

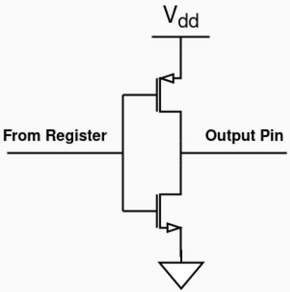
GPIO

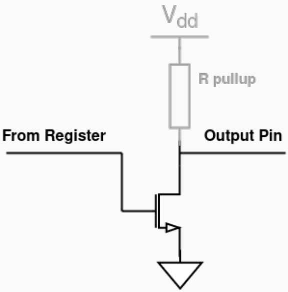
GPIO basics

- *General Purpose Input/Output*
- the most basic external communication method
- commonly found on every MCU, even MPUs
- as input:
  - detect digital state of input voltage
  - possible values - **0 or 1**
- as output:
  - set a voltage level on the pin
  - possible values - **0, 1, High-Z**

Basic model of GPIO

- input model - wired straight into register/bus
- output model - one or two switches, depending on the type
- gross over-simplification, but useful mental model
- *push-pull* vs. *open-drain* GPIO





Pins and Ports

- GPIO pins are organized into ports
- port size is usually dependent on:
    - architecture width
    - bus width
- a single GPIO peripheral usually has only single port
- writes and reads to the port are parallel
    - used in writing parallel communication

PIC18 GPIO

- each port is a separate peripheral
- 5 registers per port
- for now, we only need 2 registers:
  - TRIS - data direction
  - PORT - levels of the pin

Writing ASM

- empty ASM project ready in IS
- assignment for the rest of the lecture:
    1. set GPIO Port D as output and write 0xFF to it
    2. write 0x15 to WREG, add 0x32 to it and display it on LEDs
    3. create a subroutine that XORs content of WREG with 0xAA
    4. call this subroutine and display result on LEDs
    5. try writing a loop that loops 20 times

Homework

Mandatory

- create a knight rider effect on LEDs
- can be only a single direction
- you'll need a delay, you can utilize *NOP* instructions and loops

Optional

- draw a schematic for controlling 6 LEDs with 3 GPIO pins
- hint: GPIO pins are push-pull and LEDs are directional