Introduction

Lecture 1

Welcome

to the *Microprocessor Architecture* engineering class

You will learn

- how hardware works
- how to actually build your own hardware device
- the Rust programming Language

We expect

- to come to class
- ask a lot of questions

Team

Our team

Lectures

- Alexandru Radovici

Labs

- Irina Niță
- Irina Bradu
- Teodor Dicu
- Andrei Zamfir
- Dănuț Aldea
- Teodora Miu

Outline

Lectures

- 12 lectures
- 1 Q&A lecture for the project

Labs

- 12 labs

Project

- Build a hardware device running software written in Rust
- The cost for the hardware is around 150 RON
- Presented at PM Fair during the last week of the semester

Grading

Part	Description	Points
Lecture tests	You will have a test at every class with subjects from the previous class.	2p
Lab	Your work at every lab will be graded.	2p
Project	You will have to design and implement a hardware device. Grading will be done for the documentation, hardware design and software development.	5p
Exam	You will have to take an exam during the session.	2p
Total	*You will need at least 4.5 points to pass the subject.*	**11p**

Subjects

Theory

- How a microprocessor works
- How the ARM Cortex-M processor works
- Using digital signals to control devices
- Using analog signals to read data from sensors
- How interrupts work
- How asynchronous programming works (async/await)
- How embedded operating systems work

Practical

- How to use the Raspberry Pi Pico
  - Affordable
  - Powerful processor
  - Good documentation
- How to program in Rust
  - Memory Safe
  - *Java-like features, without Java's penalties*
  - Defines an embedded standard interface *embedded-hal*

Apollo Guidance Computer

We choose to go to the moon

John F. Kennedy, Rice University, 1961

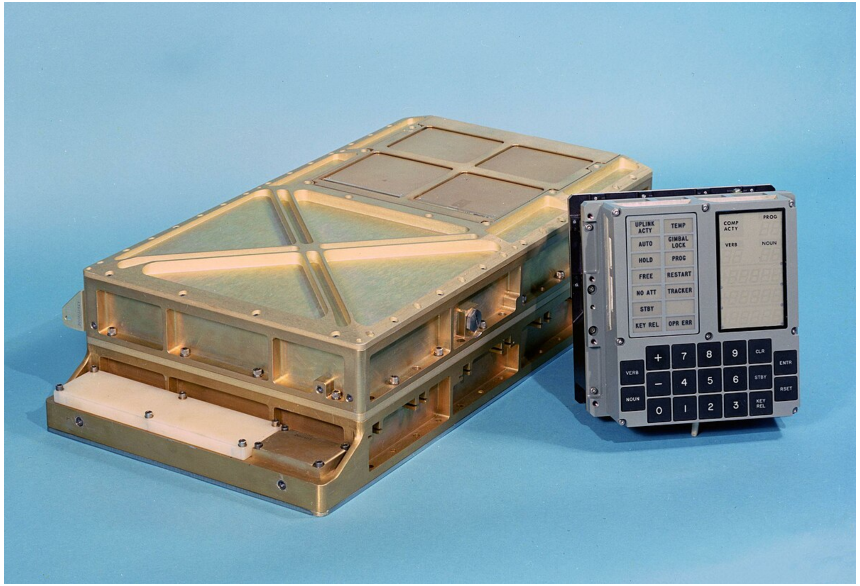
in this decade and do the other things, ***not because they are easy, but because they are hard****, because* ***that goal will serve to organize and measure the best of our energies and skills****, because that challenge is one that we are willing to accept, one we are unwilling to postpone, and one which we intend to win, and the others, too.*

AGC

August 1966

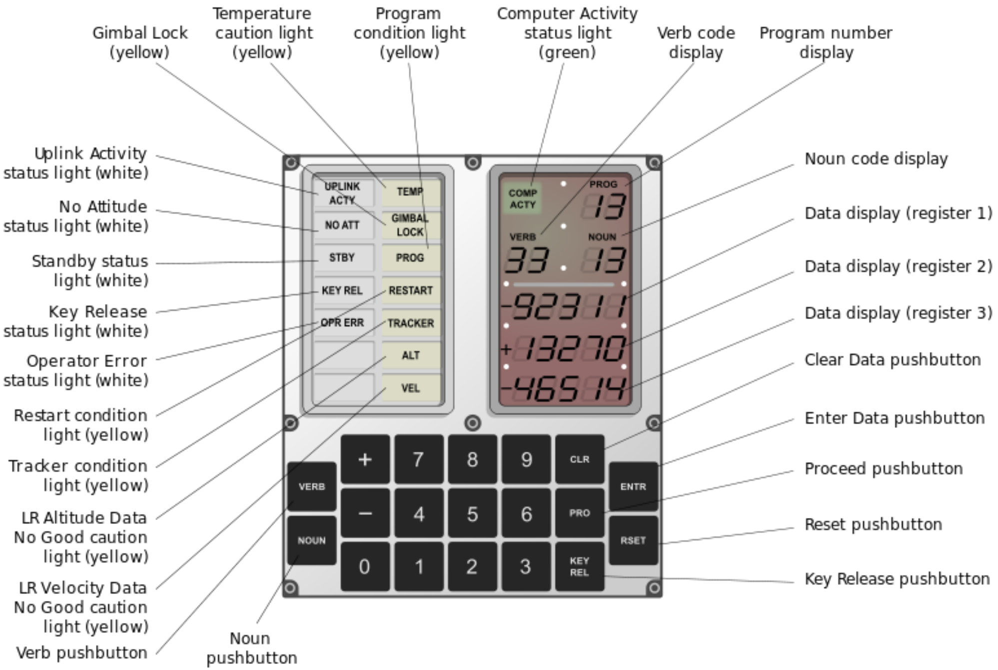
Frequency	2.048 MHz
World Length	15 + 1 bit
RAM	4096 B
Storage	72 KB
Software API	AGC Assembly Language

This landed the *moon eagle*.



DSKY

Display and keyboard



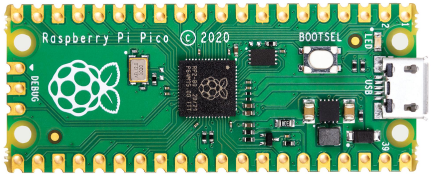
Simulator

What is a microprocessor?

Microcontroller (MCU)

Integrated in embedded systems for certain tasks

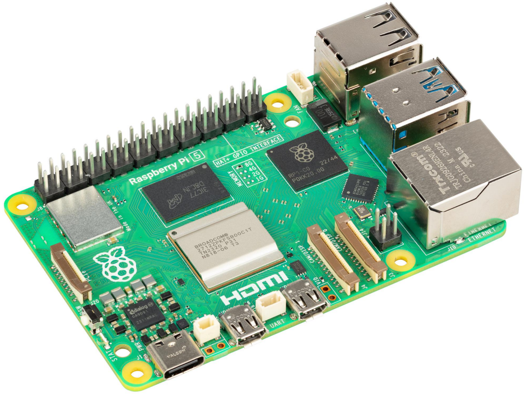
- low operating frequency (MHz)
- a lot of I/O ports
- controls hardware
- does not require an Operating System
- costs \$0.1 - \$25
- annual demand is billions



Microprocessor (CPU)

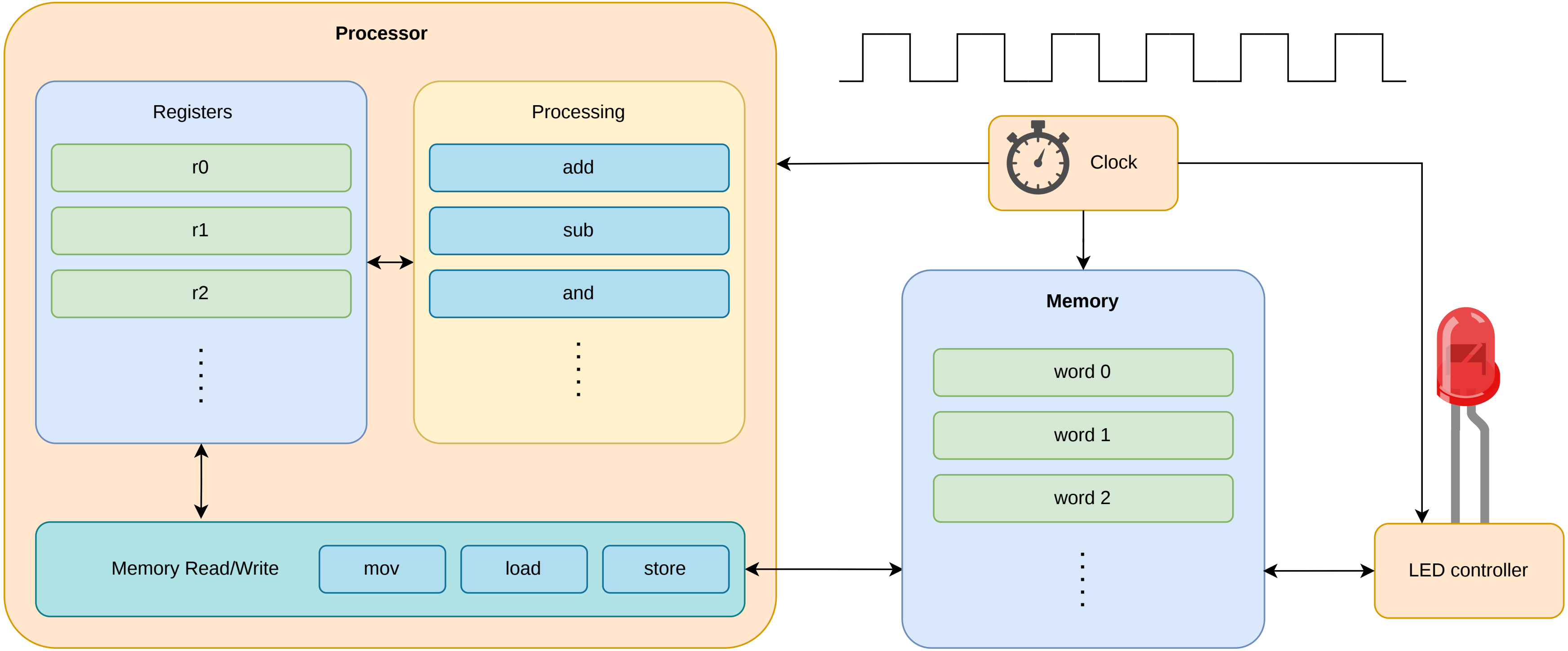
General purpose, for PC & workstations

- high operating frequency (GHz)
- limited number of I/O ports
- usually requires an Operating System
- costs \$75 - \$500
- annual demand is tens of millions



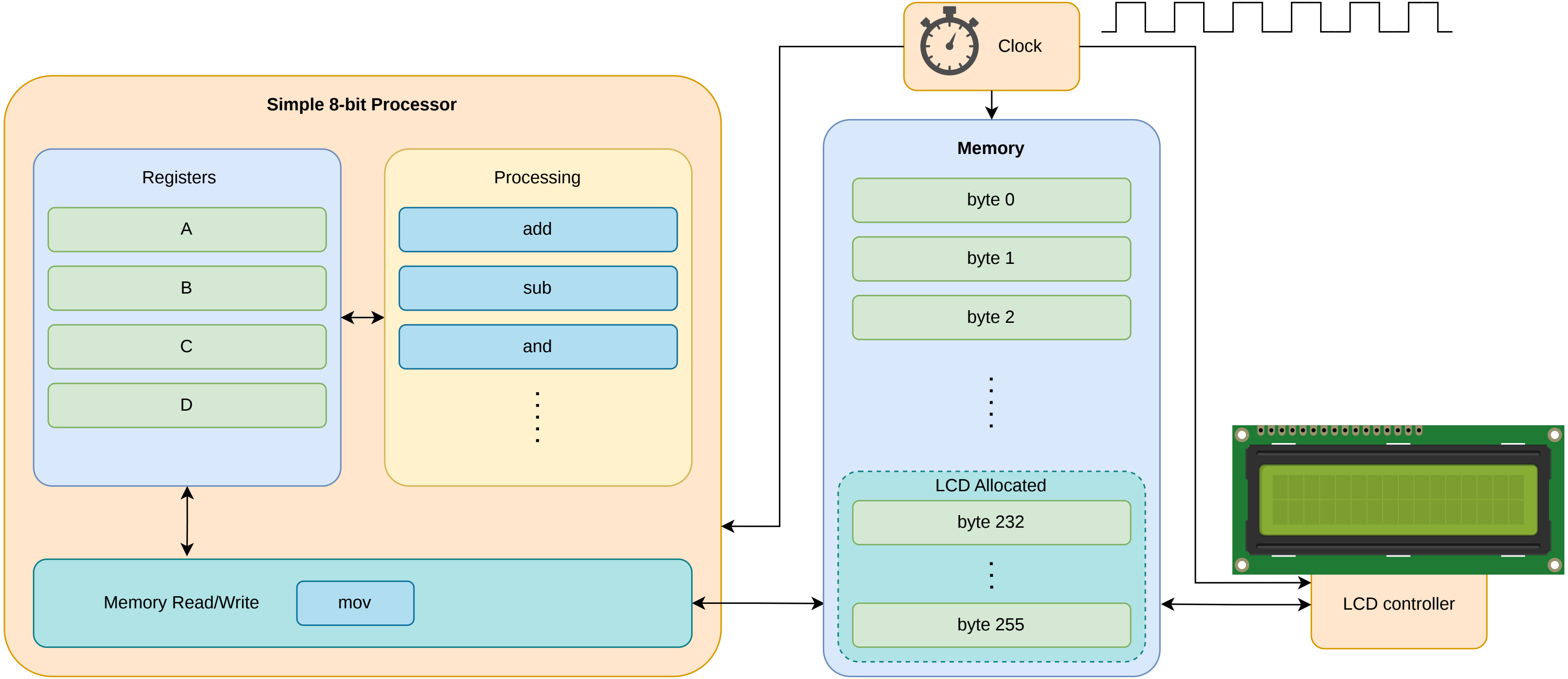
How a microprocessor (MCU) works

This is a simple processor



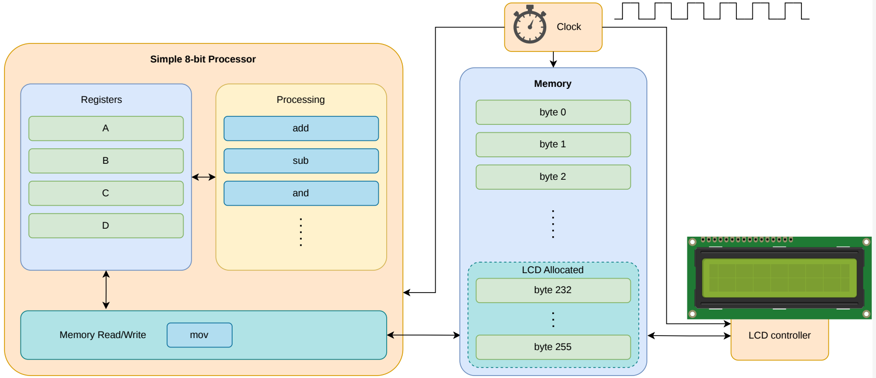
8 bit processor

a simple 8 bit processor with a text display



Programming

in Rust




```rust
use eight_bit_processor::print;

static hello: &str = "Hello World!";

#[start]
fn start() {
    print(hello);
}
```

Assembly

```asm
    JMP start
hello: DB "Hello World!" ; Variable
       DB 0  ; String terminator
start:
    MOV C, hello    ; Point to var
    MOV D, 232  ; Point to output
    CALL print
         HLT              ; Stop execution
print:        ; print(C:*from, D:*to)
    PUSH A
    PUSH B
    MOV B, 0
.loop:
    MOV A, [C]  ; Get char from var
    MOV [D], A  ; Write to output
    INC C
    INC D
    CMP B, [C]  ; Check if end
    JNZ .loop ; jump if not

    POP B
    POP A
    RET
```

Demo

a working example for the previous code

Start

Real Word Microcontrollers

Intel / AVR / PIC / TriCore / ARM Cortex-M / RISC-V rv32i(a)mc

Bibliography

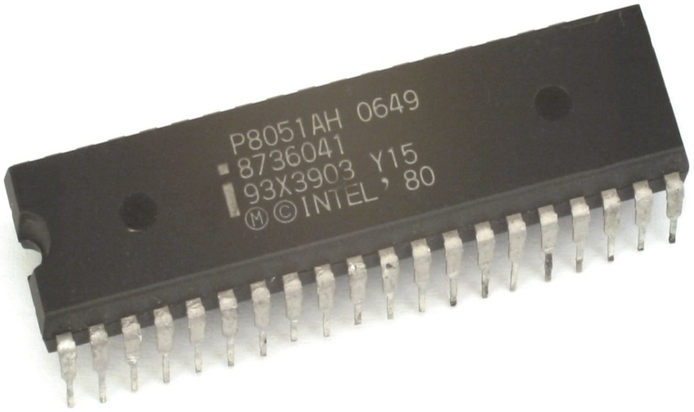
for this section

Joseph Yiu, *The Definitive Guide to ARM® Cortex®-M0 and Cortex-M0+ Processors, 2nd Edition*

- Chapter 1 - *Introduction*
- Chapter 2 - *Technical Overview*

Intel

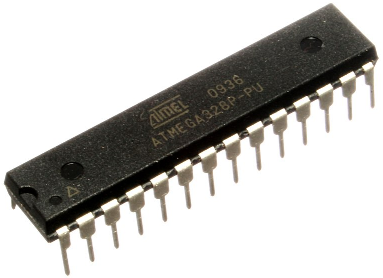
Vendor	Intel
ISA	8051, 8051
Word	8 bit
Frequency	a few MHz
Storage	?
Variants	*8048, 8051*



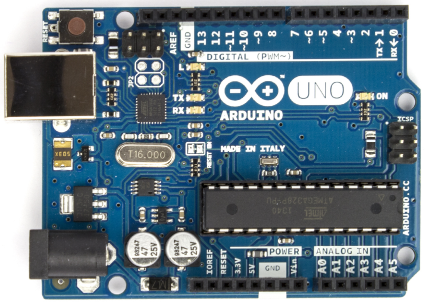
AVR

probably *Alf and Vegard's RISC processor*

Authors	Alf-Egil Bogen and Vegard Wollan
Vendor	Microchip (*Atmel*)
ISA	AVR
Word	8 bit
Frequency	1 - 20 MHz
Storage	4 - 256 KB
Variants	*ATmega*, *ATtiny*



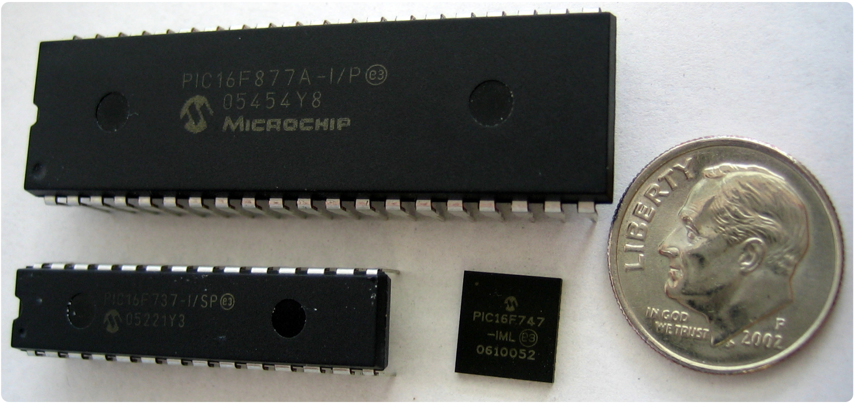
Board



PIC

Peripheral Interface Controller / Programmable Intelligent Computer

Vendor	Microchip
ISA	PIC
Word	8 - 32
Frequency	1 - 20 MHz
Storage	256 B - 64 KB
Variants	*PIC10, PIC12, PIC16, PIC18, PIC24, PIC32*



TriCore

Vendor	Infineon
ISA	AURIX32
Word	32 bit
Frequency	hundreds of MHz
Storage	a few MB
Variants	*TC2xx*, *TC3xx*, *TC4xx*

ARM Cortex-M

Advanced RISC Machine

Vendor	Qualcomm, NXP, Nordic Semiconductor, Broadcom, Raspberry Pi
ISA	ARMv6-M (Thumb and some Thumb-2) ARMv7-M (Thumb and Thumb-2)
Word	32
Frequency	1 - 900 MHz
Storage	up to a few MB
Variants	*M0, M0+, M3, M4, M7, M33*

RISC-V rv32i(a)mc

Fifth generation of RISC ISA

Authors	University of California, Berkeley
Vendor	Espressif System
ISA	rv32i(a)mc
Word	32 bit
Frequency	1 - 200 MHz
Storage	4 - 256 KB
Variants	*rv32imc*, *rv32iamc*

RP2040

ARM Cortex-M0+, built by Raspberry Pi

Bibliography

for this section

Raspberry Pi Ltd, *RP2040 Datasheet*

- Chapter 1 - *Introduction*
- Chapter 2 - *System Description*
    - Section 2.1 - *Bus Fabric*

RP2040

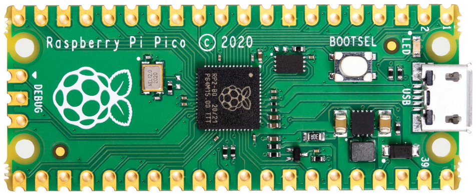
the MCU

Vendor	Raspberry PI
Variant	ARM Cortex-M0+
ISA	ARMv6-M (Thumb and some Thumb-2)
Cores	2
Word	32 bit
Frequency	up to 133 MHz
RAM	264 KB
Storage	N/A (external only)

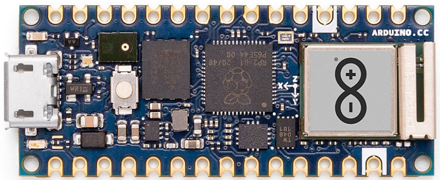
Boards

that use RP2040

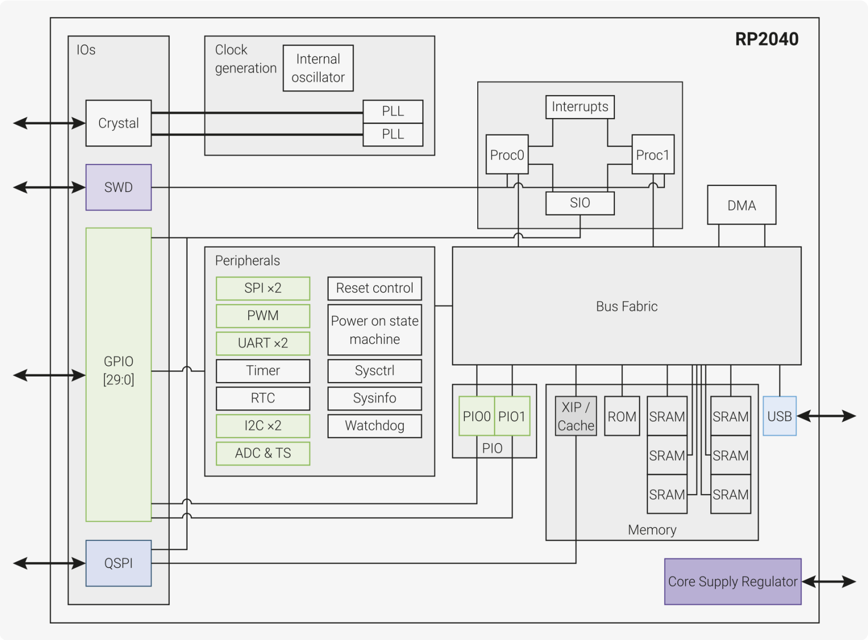
Raspberry Pi Pico (W)



Arduino Nano RP2040 Connect



The Chip



GPIO: General Purpose Input/Output
SWD: Debug Protocol
DMA: Direct Memory Access

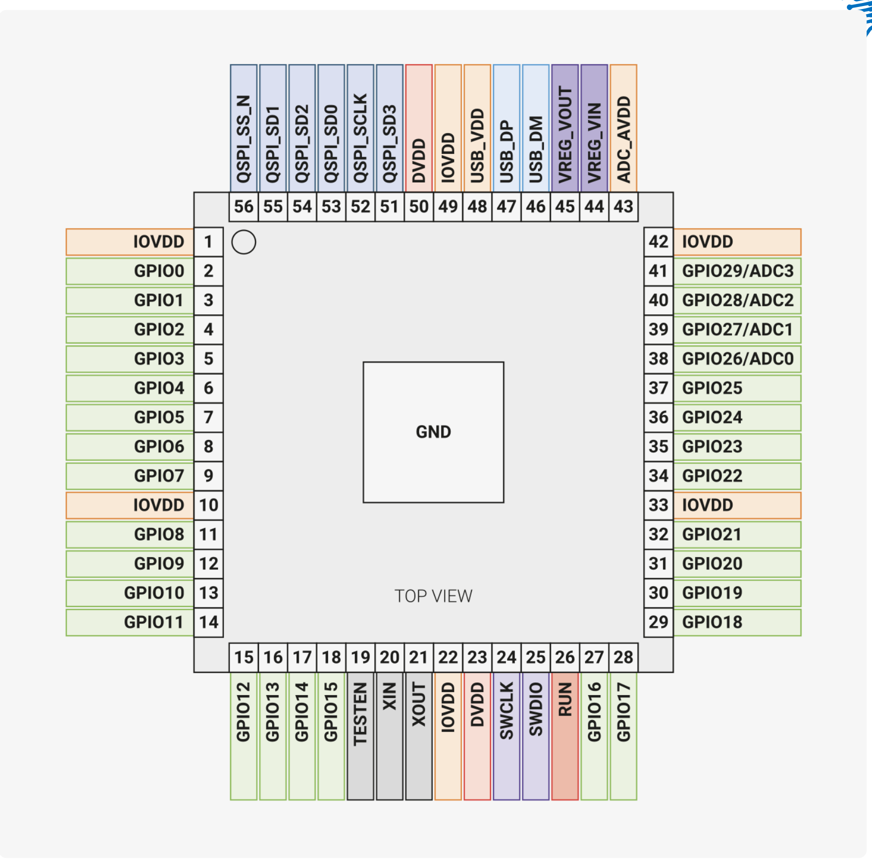
Peripherals

SIO	Single Cycle Input/Output
PWM	Pulse With Modulation
ADC	Analog to Digital Converter
(Q)SPI	(Quad) Serial Peripheral Interface
UART	Universal Async. Receiver/Transmitter
RTC	Real Time Clock
I2C	Inter-Integrated Circuit
PIO	Programmable Input/Output

Pins

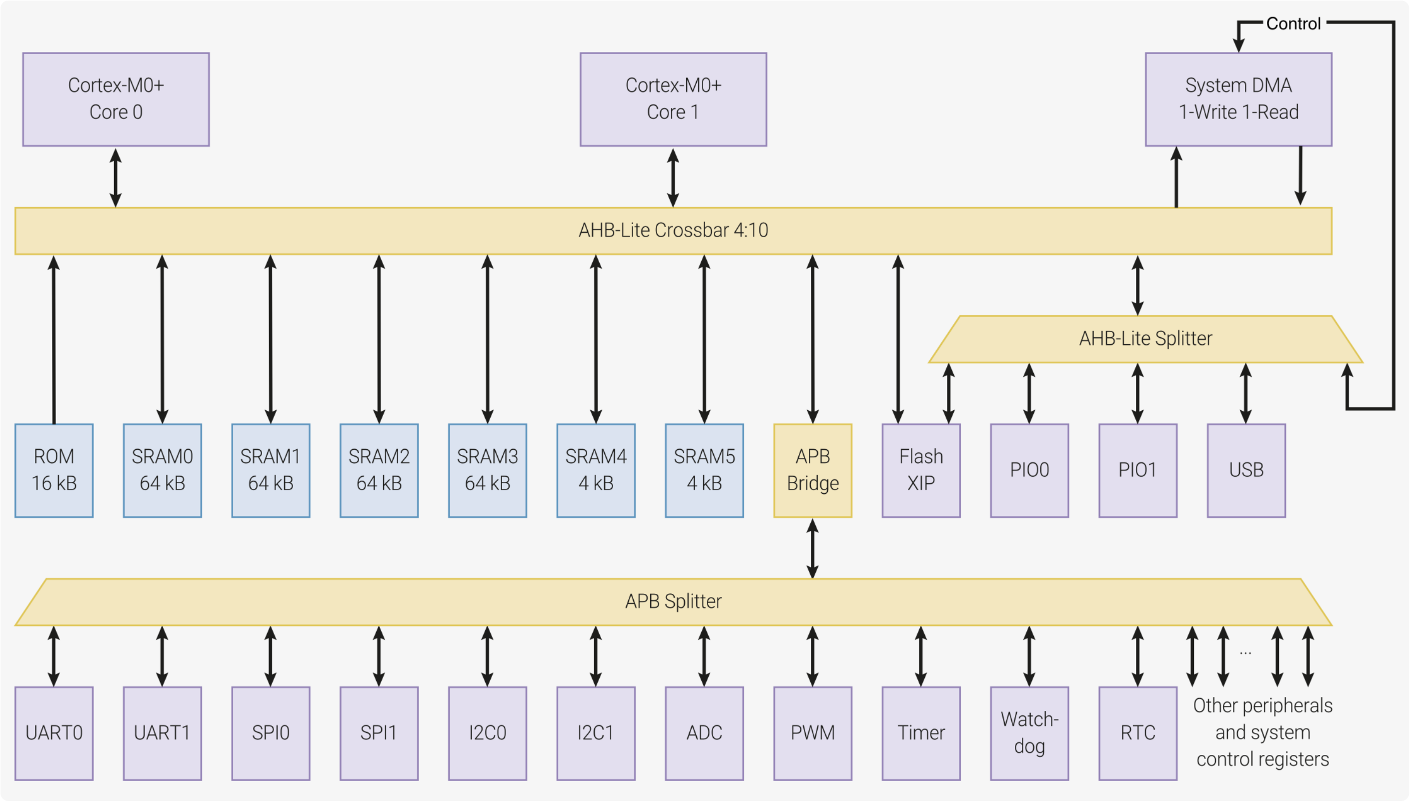
have multiple functions

	Function								
GPIO	F1	F2	F3	F4	F5	F6	F7	F8	F9
0	SPI0 RX	UART0 TX	I2C0 SDA	PWM0 A	SIO	PIO0	PIO1		USB OVCUR DET
1	SPI0 CSn	UART0 RX	I2C0 SCL	PWM0 B	SIO	PIO0	PIO1		USB VBUS DET
2	SPI0 SCK	UART0 CTS	I2C1 SDA	PWM1 A	SIO	PIO0	PIO1		USB VBUS EN
3	SPI0 TX	UART0 RTS	I2C1 SCL	PWM1 B	SIO	PIO0	PIO1		USB OVCUR DET
4	SPI0 RX	UART1 TX	I2C0 SDA	PWM2 A	SIO	PIO0	PIO1		USB VBUS DET
5	SPI0 CSn	UART1 RX	I2C0 SCL	PWM2 B	SIO	PIO0	PIO1		USB VBUS EN
6	SPI0 SCK	UART1 CTS	I2C1 SDA	PWM3 A	SIO	PIO0	PIO1		USB OVCUR DET
7	SPI0 TX	UART1 RTS	I2C1 SCL	PWM3 B	SIO	PIO0	PIO1		USB VBUS DET
8	SPI1 RX	UART1 TX	I2C0 SDA	PWM4 A	SIO	PIO0	PIO1		USB VBUS EN
9	SPI1 CSn	UART1 RX	I2C0 SCL	PWM4 B	SIO	PIO0	PIO1		USB OVCUR DET
10	SPI1 SCK	UART1 CTS	I2C1 SDA	PWM5 A	SIO	PIO0	PIO1		USB VBUS DET
11	SPI1 TX	UART1 RTS	I2C1 SCL	PWM5 B	SIO	PIO0	PIO1		USB VBUS EN
12	SPI1 RX	UART0 TX	I2C0 SDA	PWM6 A	SIO	PIO0	PIO1		USB OVCUR DET
13	SPI1 CSn	UART0 RX	I2C0 SCL	PWM6 B	SIO	PIO0	PIO1		USB VBUS DET
14	SPI1 SCK	UART0 CTS	I2C1 SDA	PWM7 A	SIO	PIO0	PIO1		USB VBUS EN
15	SPI1 TX	UART0 RTS	I2C1 SCL	PWM7 B	SIO	PIO0	PIO1		USB OVCUR DET
16	SPI0 RX	UART0 TX	I2C0 SDA	PWM0 A	SIO	PIO0	PIO1		USB VBUS DET
17	SPI0 CSn	UART0 RX	I2C0 SCL	PWM0 B	SIO	PIO0	PIO1		USB VBUS EN
18	SPI0 SCK	UART0 CTS	I2C1 SDA	PWM1 A	SIO	PIO0	PIO1		USB OVCUR DET
19	SPI0 TX	UART0 RTS	I2C1 SCL	PWM1 B	SIO	PIO0	PIO1		USB VBUS DET
20	SPI0 RX	UART1 TX	I2C0 SDA	PWM2 A	SIO	PIO0	PIO1	CLOCK GPIN0	USB VBUS EN



The Bus

that interconnects the cores with the peripherals



Conclusion

we discussed about

- How a processor functions
- Microcontrollers (MCU) / Microprocessors (CPU)
- Microcontroller architectures
- ARM Cortex-M
- RP2040