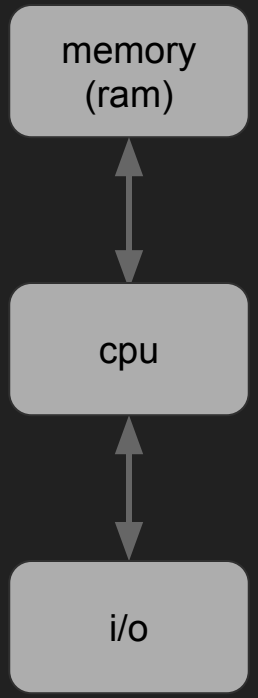
embedded architectures, htm 2024

Nathan Perry (kerb = npry)

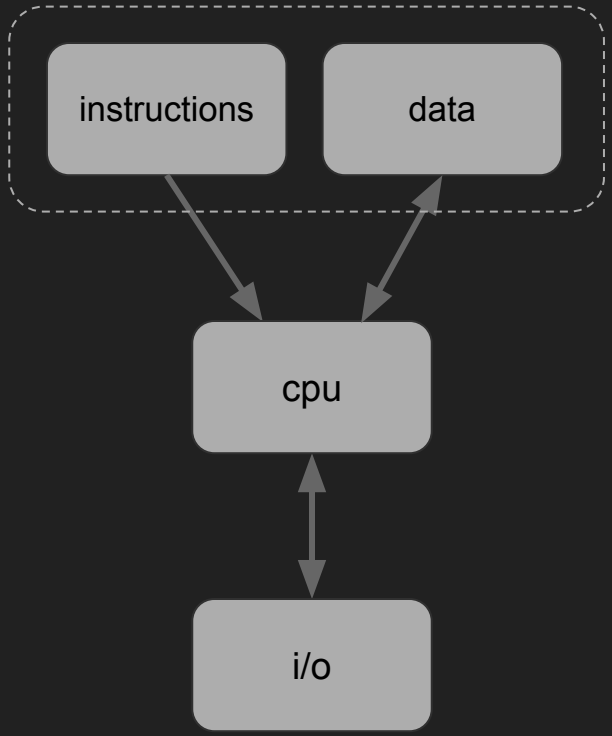
topics

- what is in a computer?
- what is in a microcontroller?
- how can we extend that functionality? tradeoffs?
  - RP2040 PIOs
  - FPGA & CPLDs
    - hardware description languages
    - open toolchains for hardware synthesis

what is in a computer?



what is in a computer?



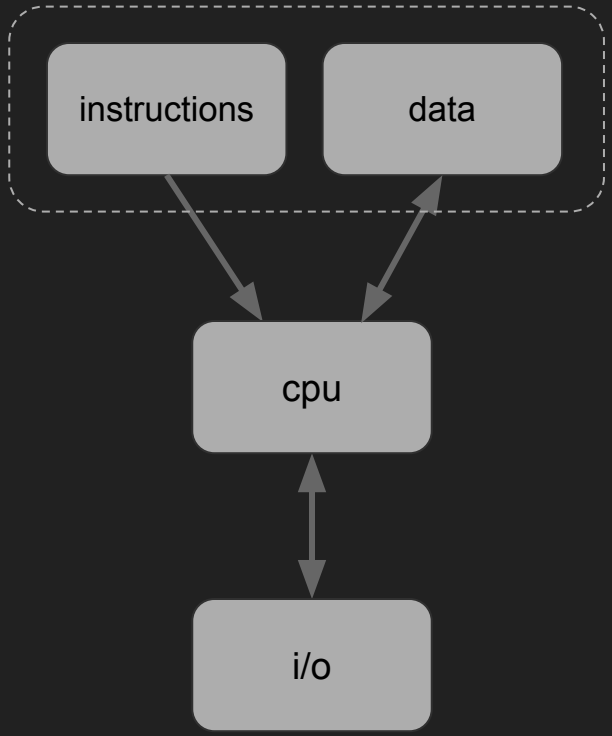
what is an instruction?

https://godbolt.org

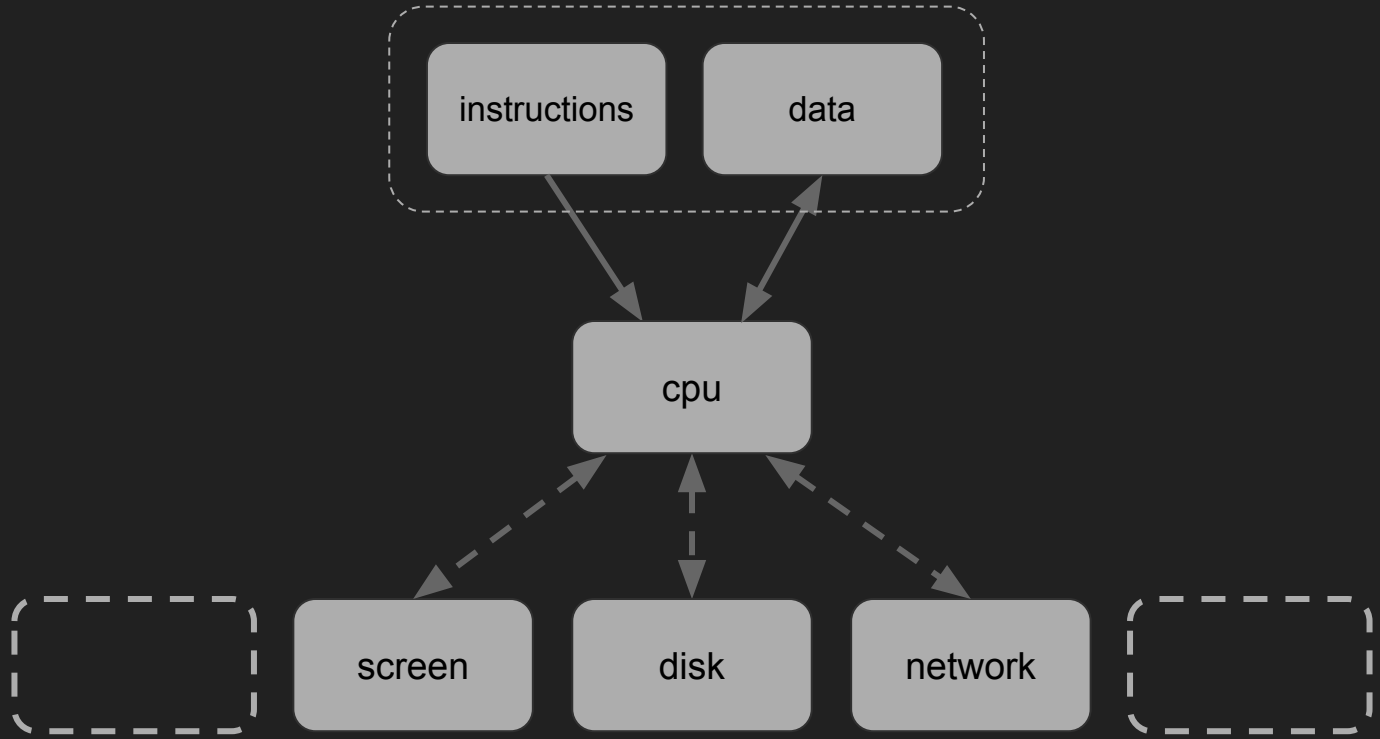
```
static int x = 0;


void loop() {
    x = x * x + 1;
}
```

```
loop():
        str     fp, [sp, #-4]!
        add     fp, sp, #0
        ldr     r3, .L2
        ldr     r3, [r3]
        ldr     r2, .L2
        ldr     r2, [r2]
        mul     r3, r2, r3
        add     r3, r3, #1
        ldr     r2, .L2
        str     r3, [r2]
        nop
        add     sp, fp, #0
        ldr     fp, [sp], #4
        bx      lr
.L2:
        .word   x
```

what is in a computer?

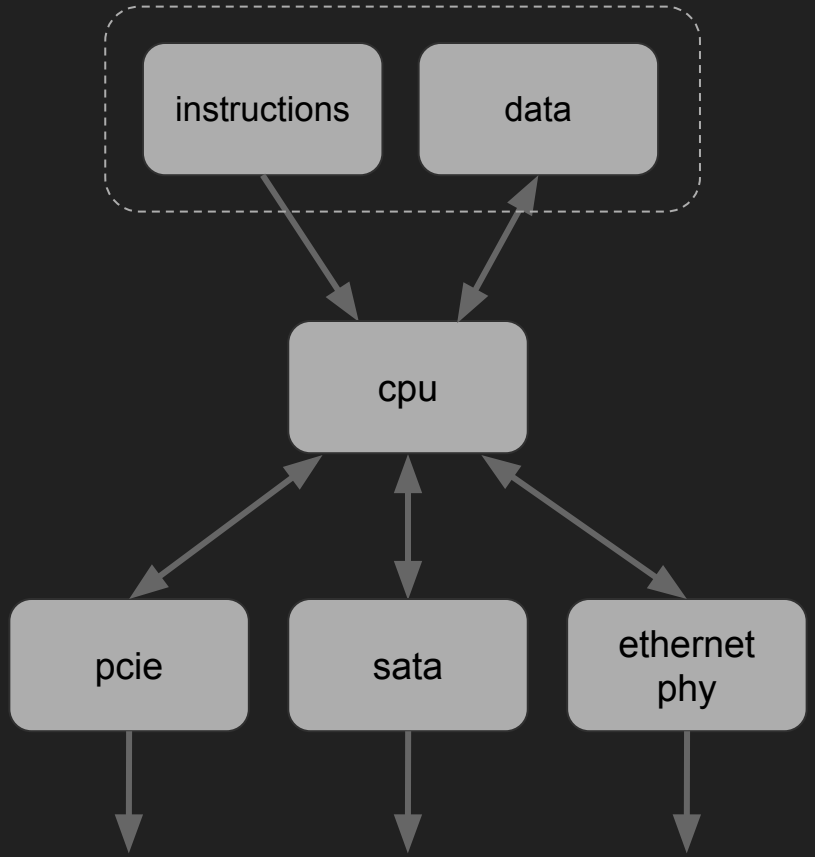


what is io?

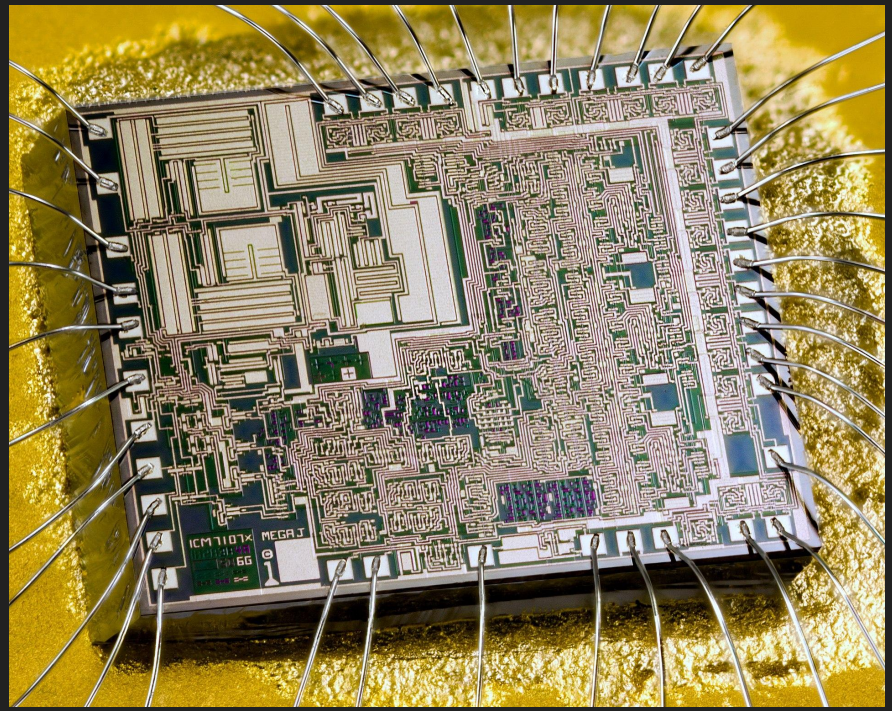


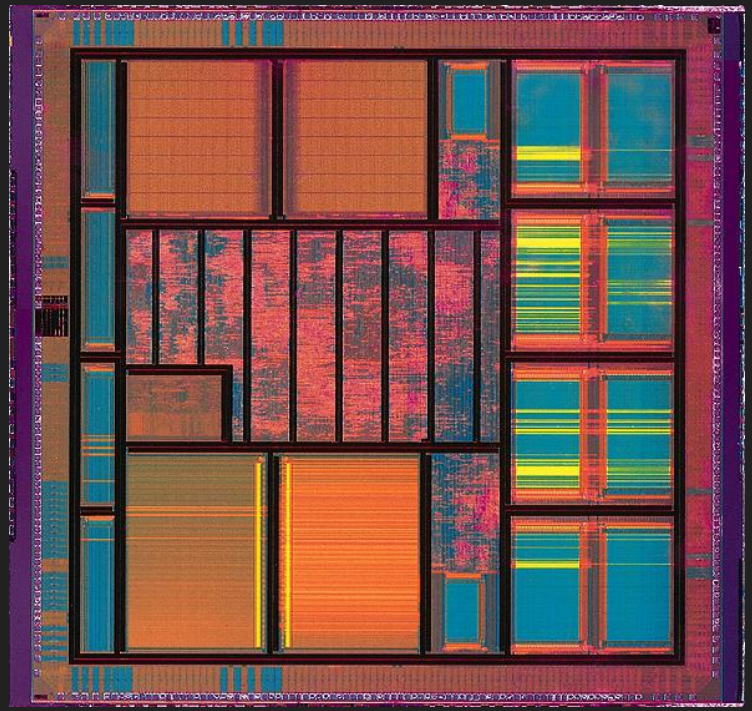
what is io?

"peripherals"

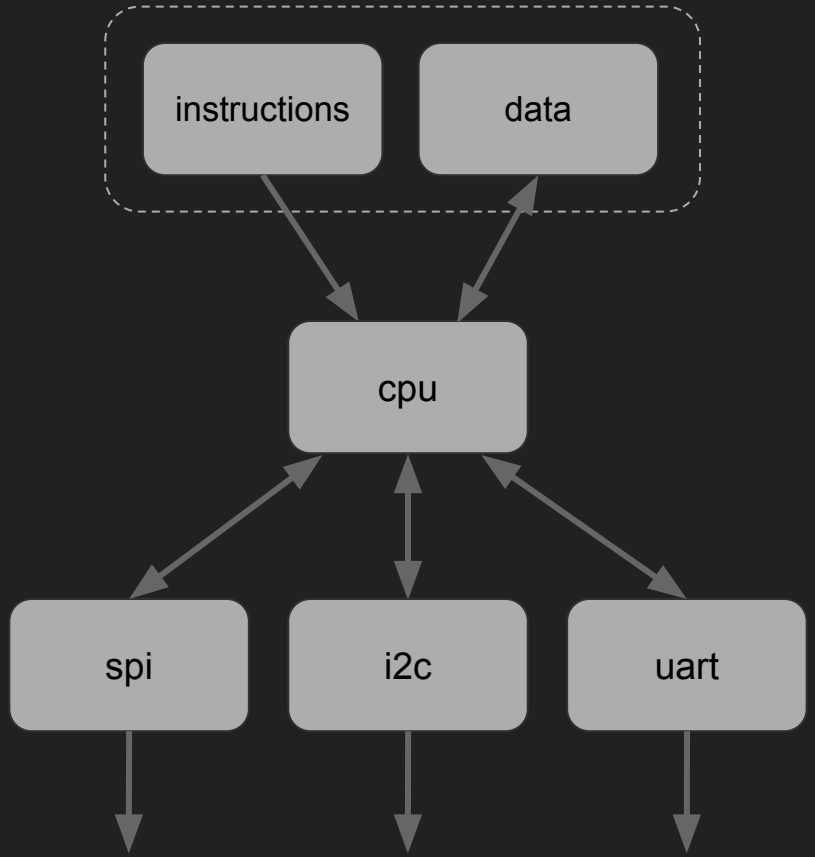


what are we really talking about

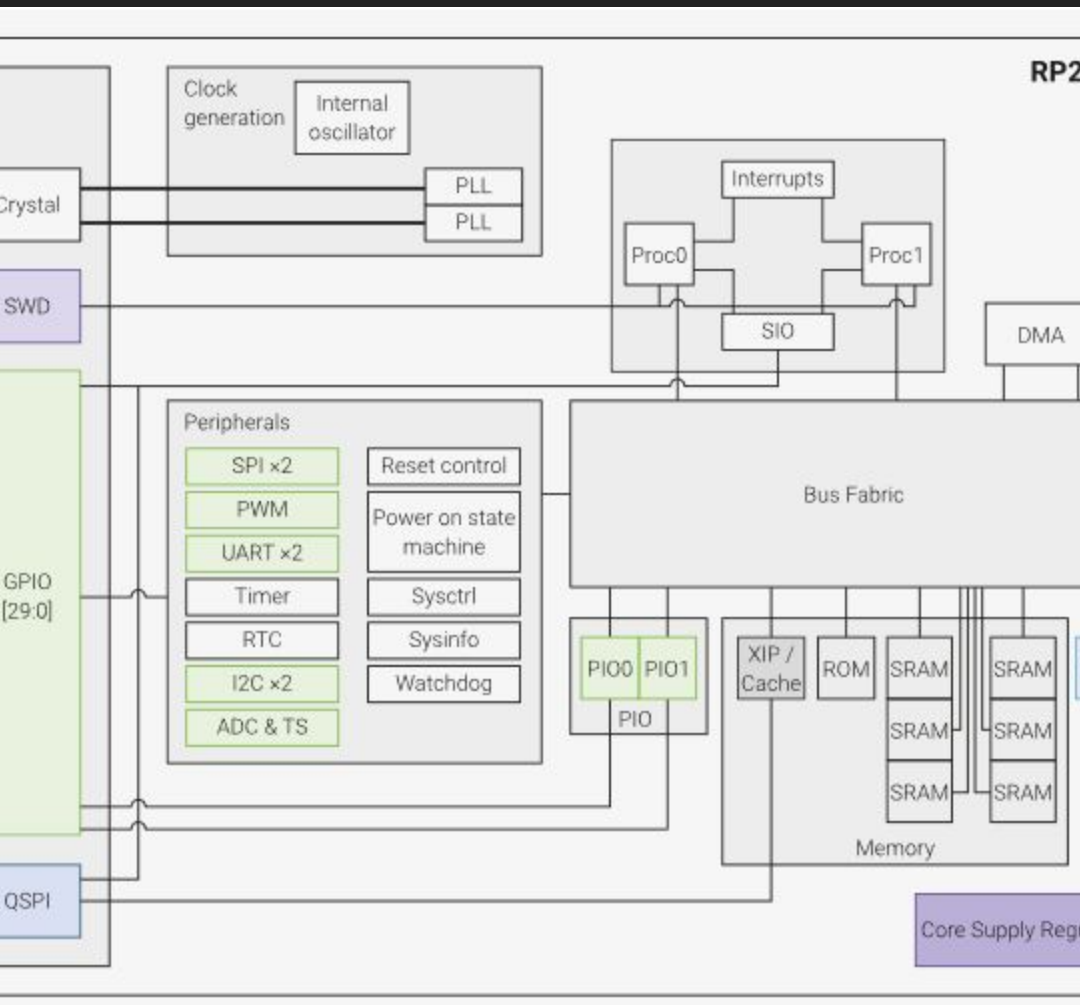


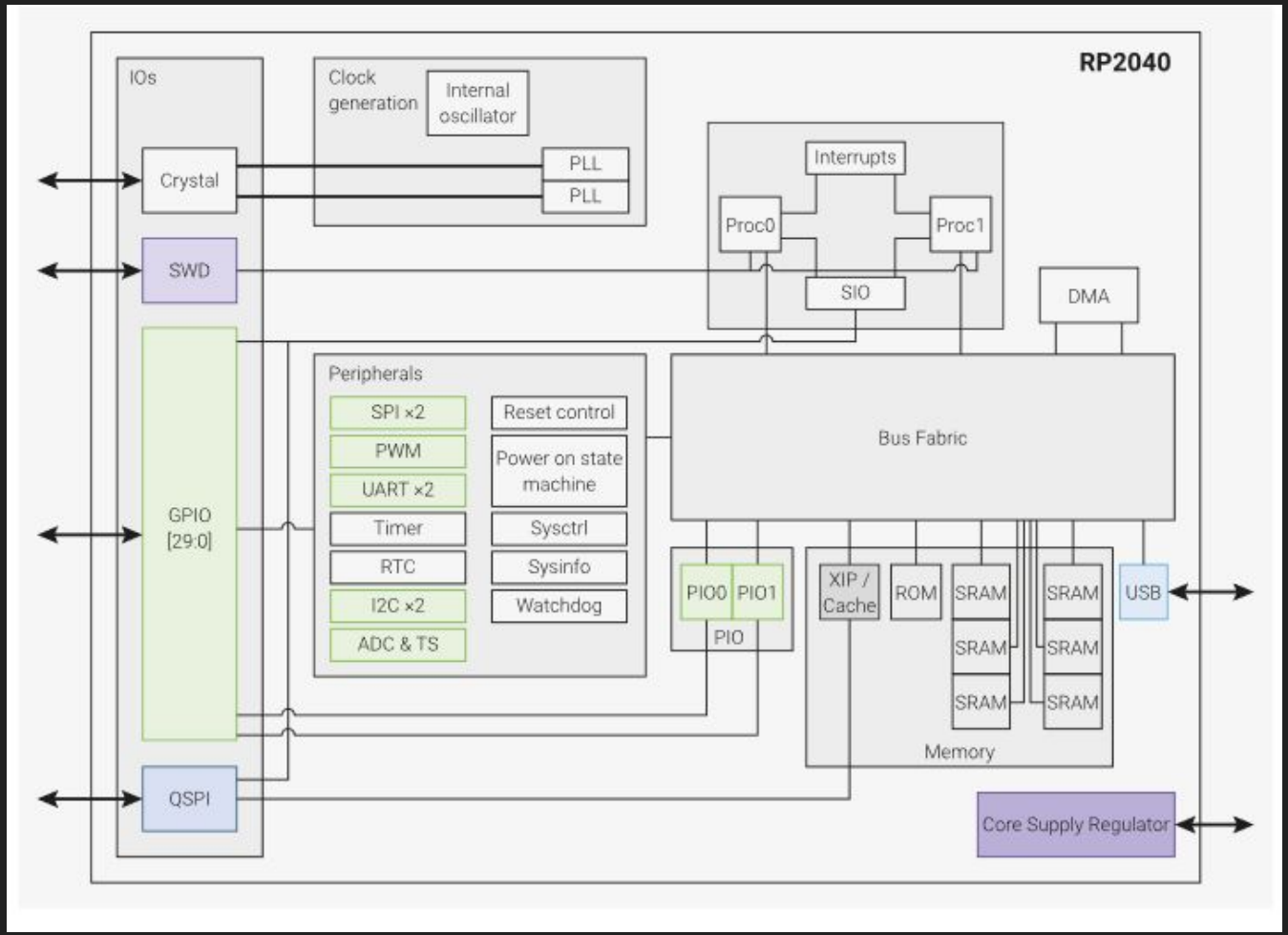


microcontrollers

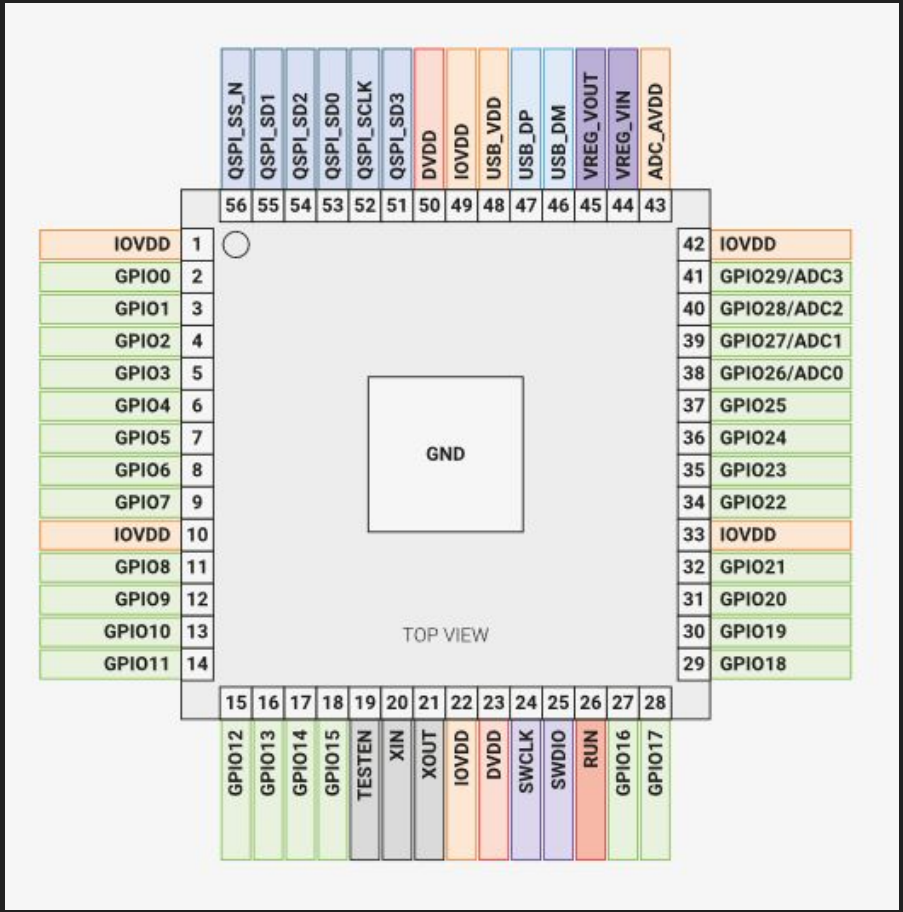


peripherals

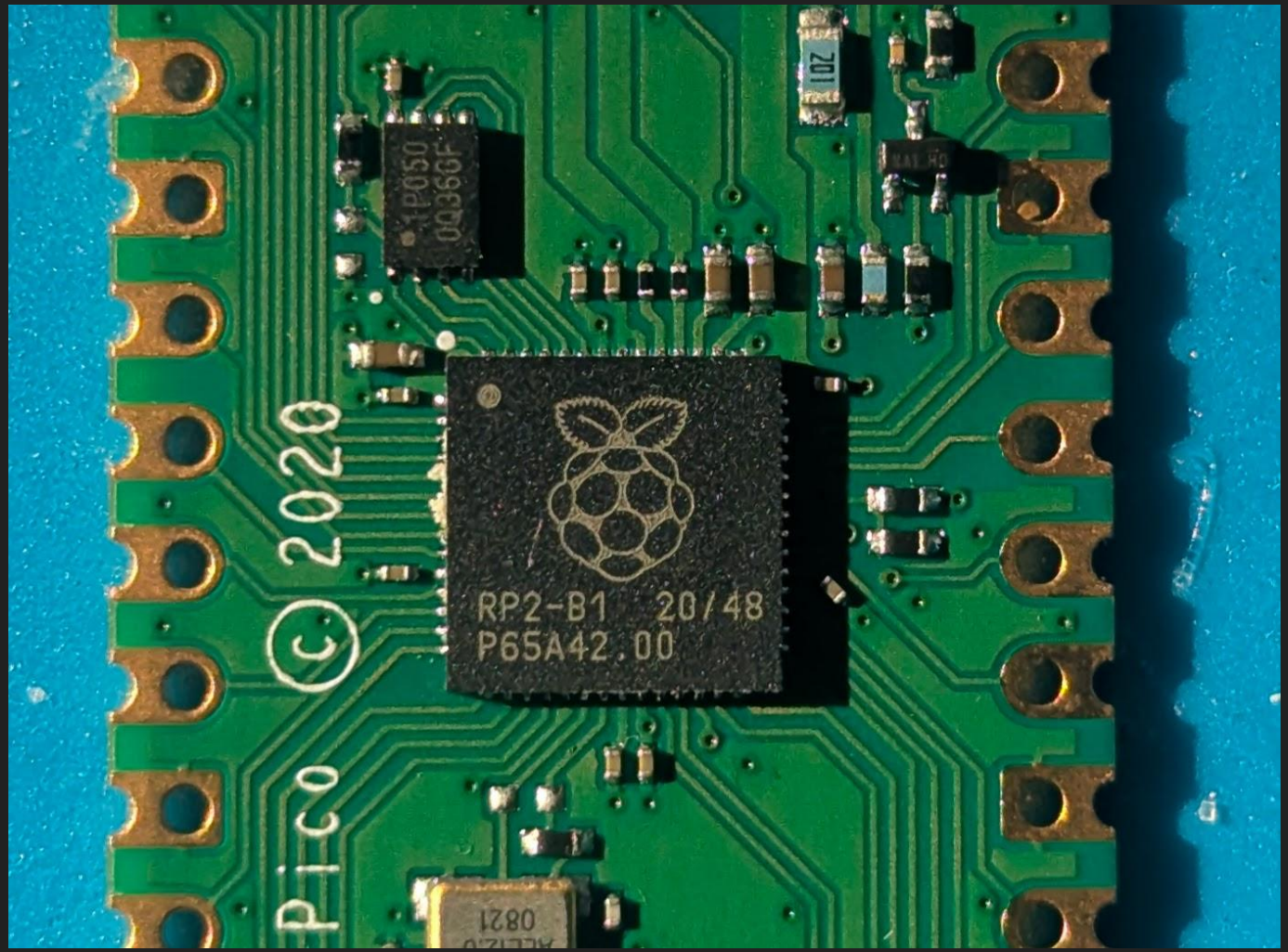

RP2
Clock generation
Internal oscillator
PLL
PLL
Crystal
SWD
Interrupts
Proc0
Proc1
SIO
DMA
Peripherals
SPI ×2
PWM
UART ×2
Timer
RTC
I2C ×2
ADC & TS
Reset control
Power on state machine
Sysctrl
Sysinfo
Watchdog
Bus Fabric
GPIO [29:0]
PIO0
PIO1
PIO
XIP / Cache
ROM
SRAM
SRAM
SRAM
SRAM
SRAM
SRAM
Memory
QSPI
Core Supply Regu



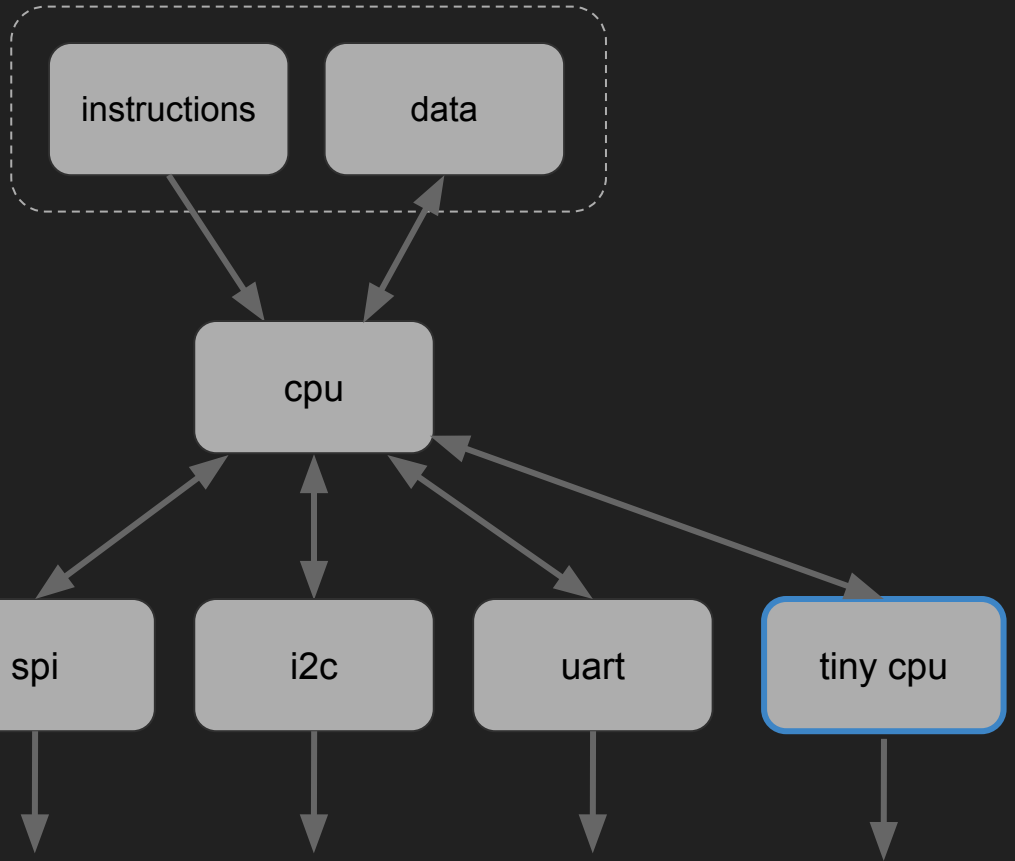
https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf



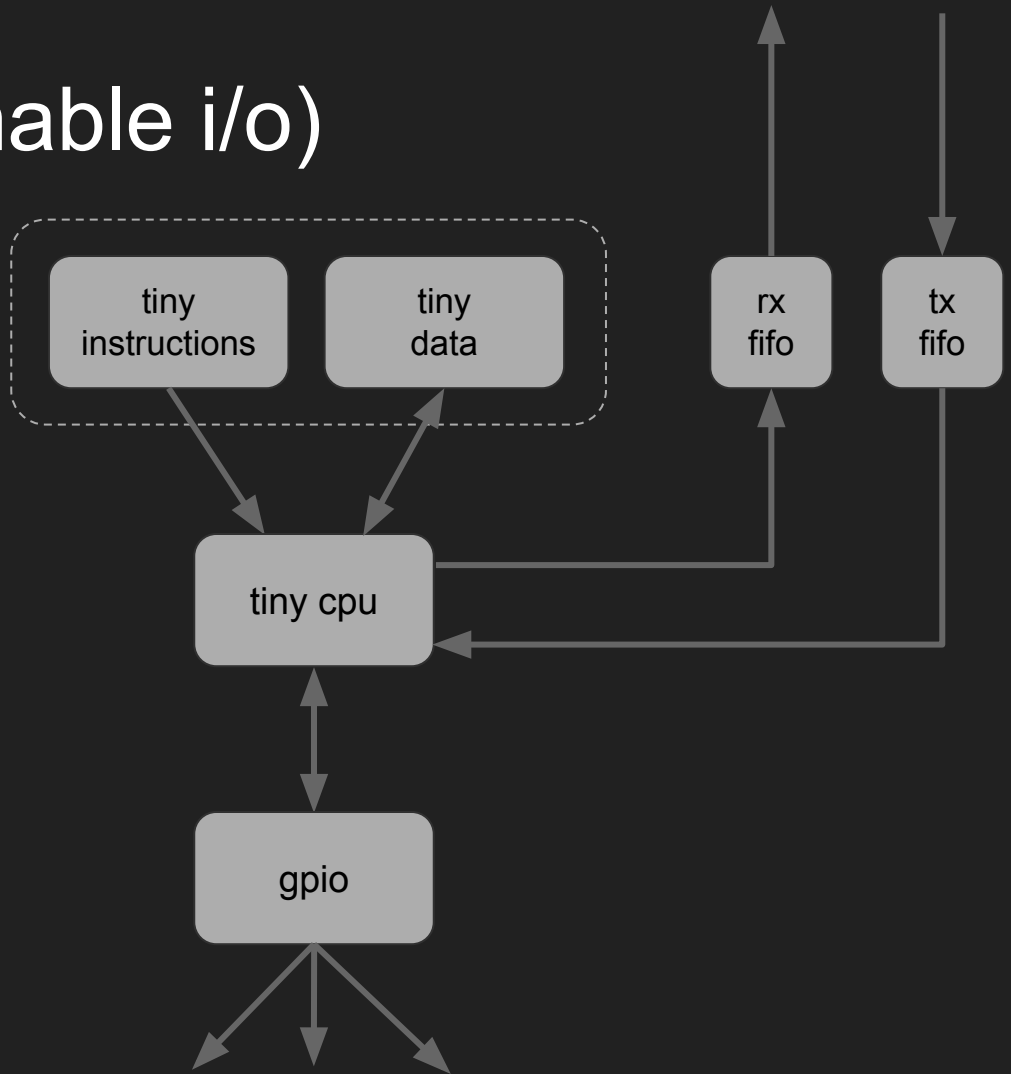
https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf


Pico © 2020
RP2-B1 20/48
P65A42.00
0821

pio (programmable i/o)



pio (programmable i/o)



pio programming

rp2040 datasheet ch.3

```
in      mov     .side_set
out     irq     .wrap, .wrap_target
push    wait    gpio / pin
pull    jmp     pindirs
set             [delay]
                x, y
```

pio example: square wave

```
set pins, 1
set pins, 0
```

how fast? system clock by default (150MHz), need to configure divider to slow down

circuitpython?

https://learn.adafruit.com/intro-to-rp2040-pio-with-circuitpython/overview

pio example: in circuitpython

https://circuitpython.org/downloads

```
prog = adafruit_pioasm.assemble("""
    set pins, 1
    set pins, 0
""")
rp2pio.StateMachine(prog,
    frequency=2000,
    first_set_pin=board.LED,
)
```

square wave extensions

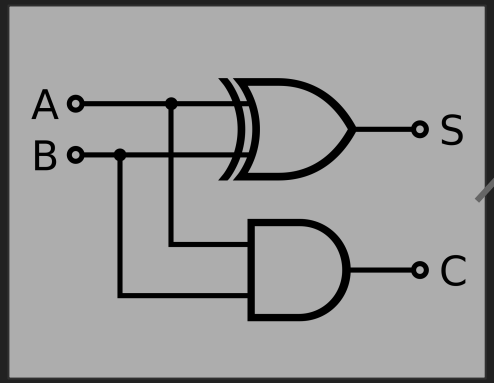
```
set pins, 0
.wrap_target
set pins, 1
set pins, 0
.wrap
```

square wave extensions

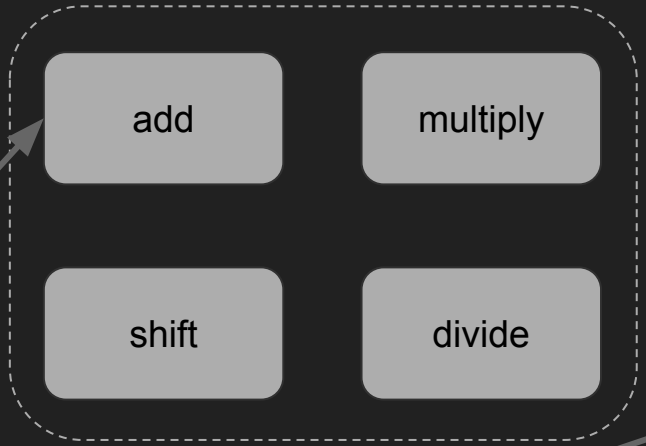
```
set pins, 0
.wrap_target
set pins, 1 [8]
set pins, 0 [1]
.wrap
```

how are silicon ICs designed?

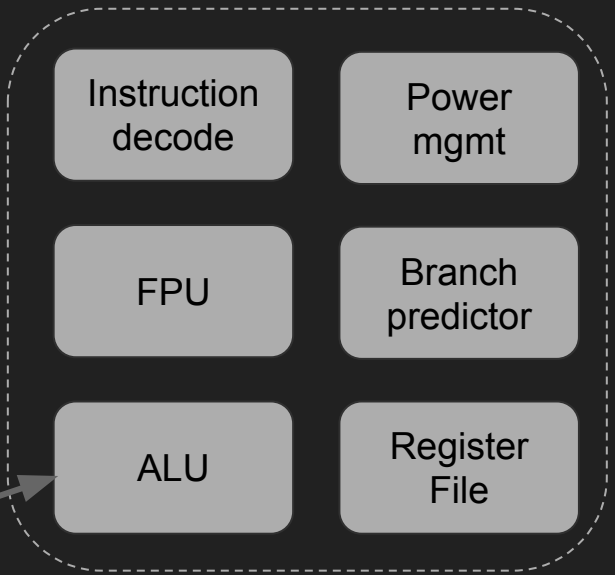
at a high level: composable functional blocks starting with transistors



half adder

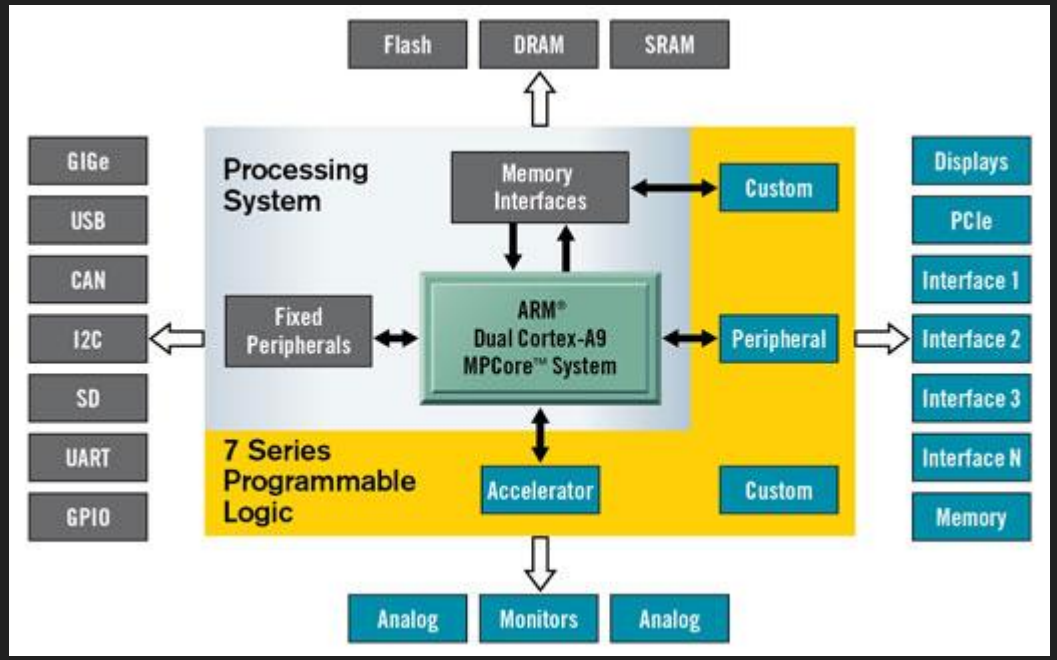


ALU

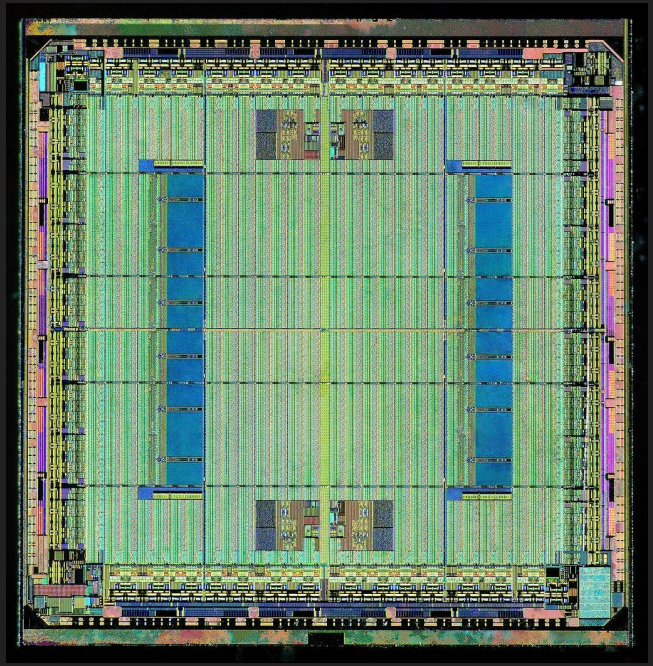


CPU

field programmable gate array (fpga)



Xilinx, CC BY-SA 3.0



BGcrain03 (wikimedia), CC BY-SA 4.0

field programmable gate array (fpga)

xilinx, altera (main vendors)

zynq 7000, tinyfpga (dev boards)

lattice ice40 – relatively low-cost

hardware description languages (HDLs)

verilog, system verilog

vhdl

generators in other programming languages:

- clash (haskell) https://clash-lang.org/
- amaranth (python) https://amaranth-lang.org

demo^

synthesis toolchains

f4pga: https://f4pga.org/

openfpga: https://github.com/lnis-uofu/OpenFPGA

icestorm: https://github.com/YosysHQ/icestorm

\+ closed vendor tools (e.g. Xilinx Vivado)