Lecture 02
Microcontrollers

CE346 – Microcontroller System Design

Branden Ghena – Fall 2025

Some slides borrowed from:
Josiah Hester (Northwestern), Prabal Dutta (UC Berkeley)

Class Updates

- No “official” lab this Friday

- Personal Software Setup will be posted tonight
  - At some point
  - Gotta finish updating it

- Office hours should start sometime next week
  - If you need help with the software setup, always feel free to reach out on Piazza!

Today's Goals

- Explore microcontrollers
  - How do they compare to computers?
  - Purpose, capabilities, tradeoffs
- Explore the microcontroller(s) on the Microbit
- Describe history and state-of-the-art for microcontrollers

Outline

- **Microcontrollers and Computers**
- Microcontroller Components
  - Processor
  - Memory
  - Peripherals
  - Pins & Other stuff
- Microbit microcontroller
- Microcontroller History

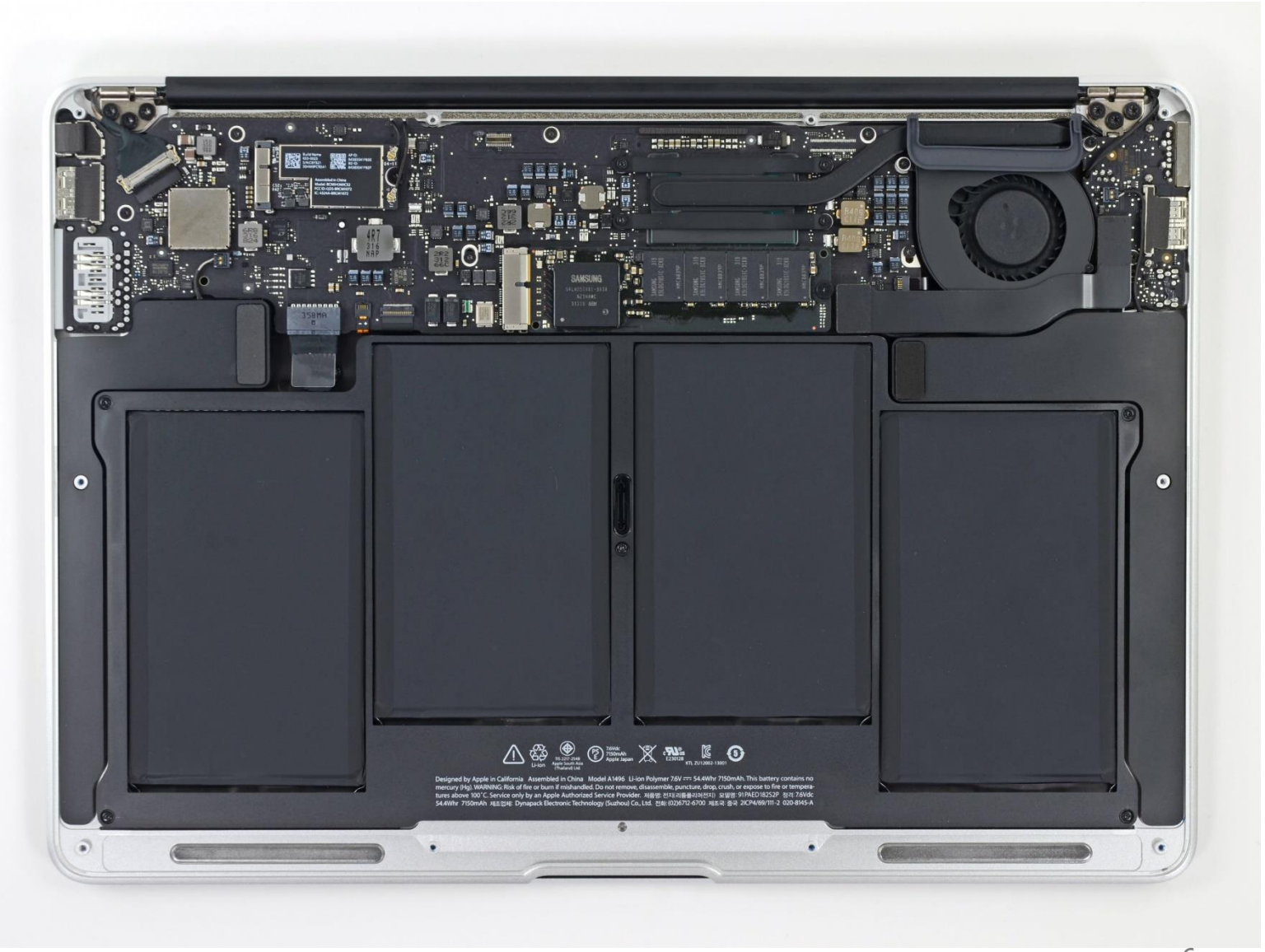
What's inside a computer?

- Macbook Air (circa 2013, 2nd gen)
  - Modern computers are mostly similar
  - Intel Core i5 processor
  - 128 GB Flash storage
  - 4 GB DDR3 RAM
  - 1440x900 pixel display
- Picked because there is a teardown I like of an embedded device from the same year

https://www.ifixit.com/Teardown/MacBook+Air+13-Inch+Mid+2013+Teardown/15042

Unscrewing the bottom cover

- Top half is the motherboard
  - Holds and connects all the parts of the computer
- Bottom half are battery packs



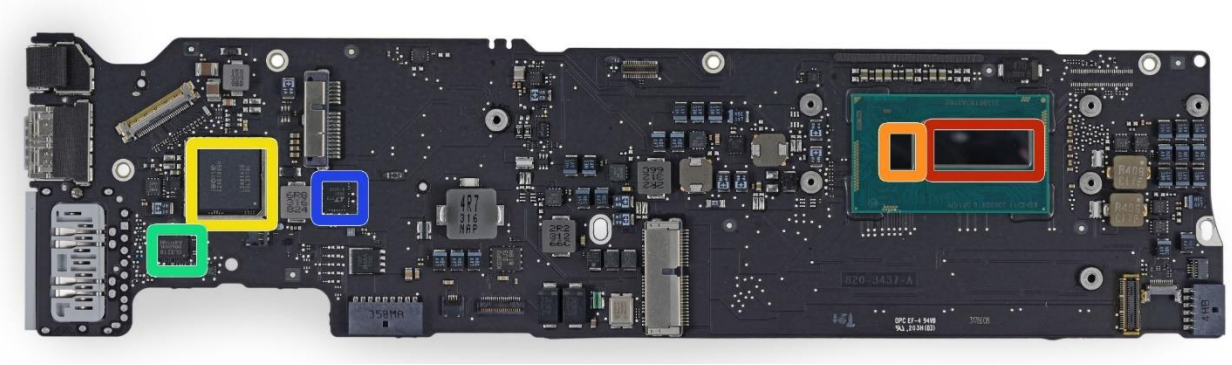
Non-volatile long-term storage: SSD

- The SSD is a module that connects through PCIe

- In modern laptops the SSD is often soldered directly onto the motherboard

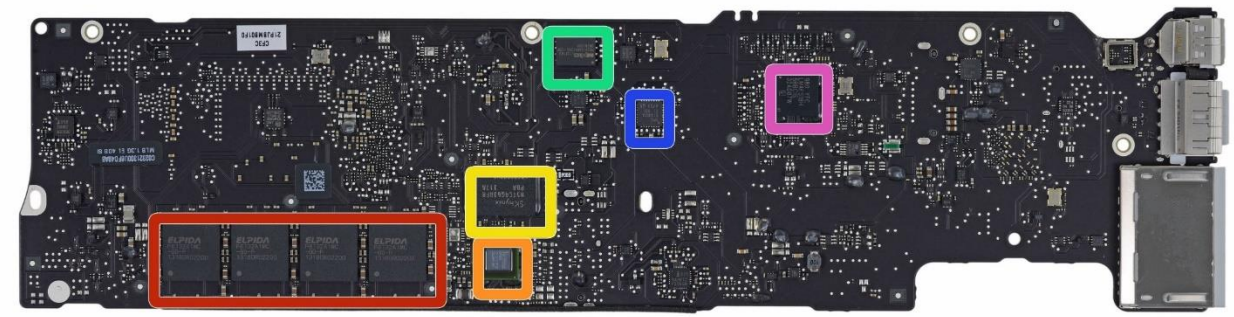
Investigating the motherboard

Front



- Red (front, right)
  - Intel core i5 processor
- Red (bottom, left)
  - Elpida LPDDR3 RAM, 4 GB

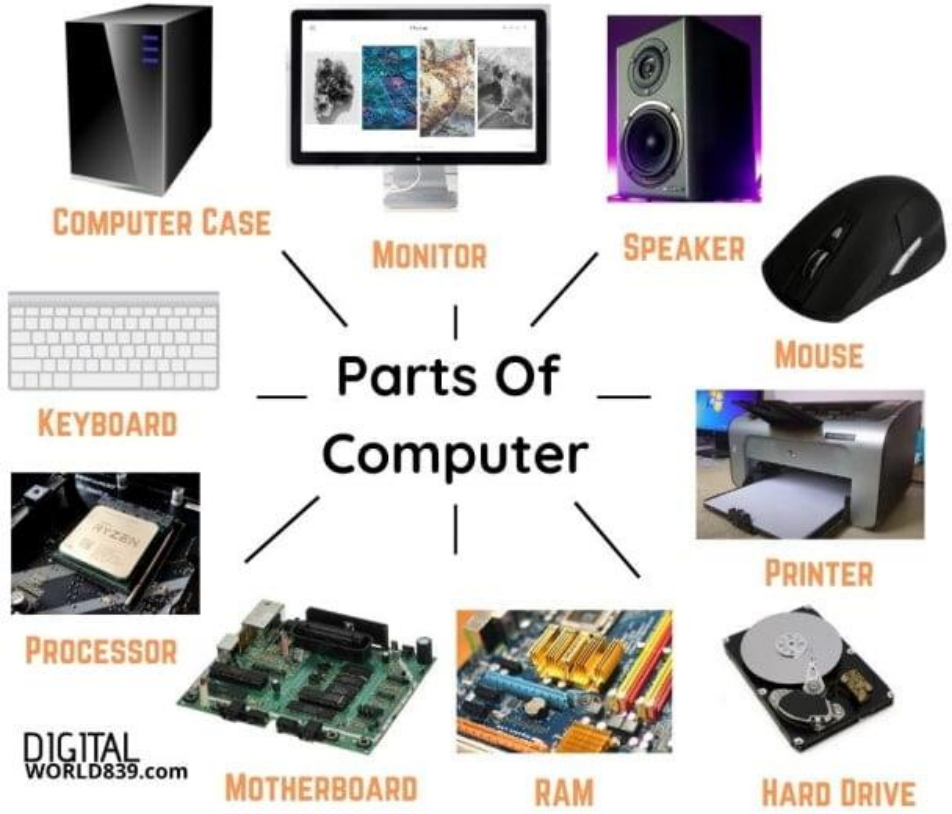
Back



- Other stuff are mostly radios, USB/Thunderbolt controllers, and power supply

Generalizing computer design

- Computers *usually* need
  - Processor
  - Memory (RAM)
  - Storage (Flash/SSD)
  - External communication
    - USB, Thunderbolt, SATA, HDMI, WiFi
  - Power management
    - Maybe batteries and charging
  - Something to connect it all: motherboard



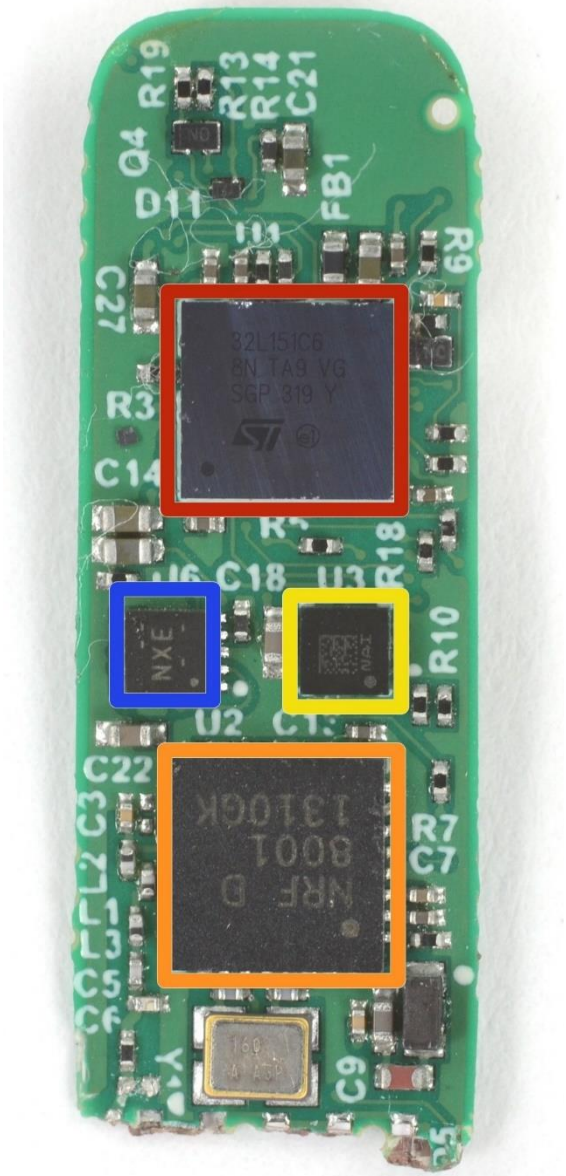
What's inside a Fitbit?

- Fitbit flex (circa 2013)
- Features
  - Counts your steps
  - Reports via Bluetooth Low Energy
  - Lights up some LEDs based on your goals
  - Vibrates when its battery is low

https://www.ifixit.com/Teardown/Fitbit+Flex+Teardown/16050

Fitbit teardown

Fitbit circuit board front



- Red (top)
  - STMicro 32L151C6 Microcontroller
- Blue (left)
  - TI BQ24040 Battery Charger
- Yellow (right)
  - STMicro LIS2DH Accelerometer
- Orange (bottom)
  - Nordic nRF8001 Bluetooth Low Energy Radio

The back is uninteresting

Fitbit as a computer

- Computers *usually* need
  - Processor
  - Memory (RAM)
  - Storage (Flash/SSD)
  - External communication
    - USB, Thunderbolt, SATA, HDMI, WiFi
  - Power management
    - Maybe batteries and charging
  - Something to connect it all: motherboard

- Microcontroller
- Bluetooth radio
- Vibratory motor
- Battery and power management
- Circuit board

Sidebar: you could make a Fitbit yourself

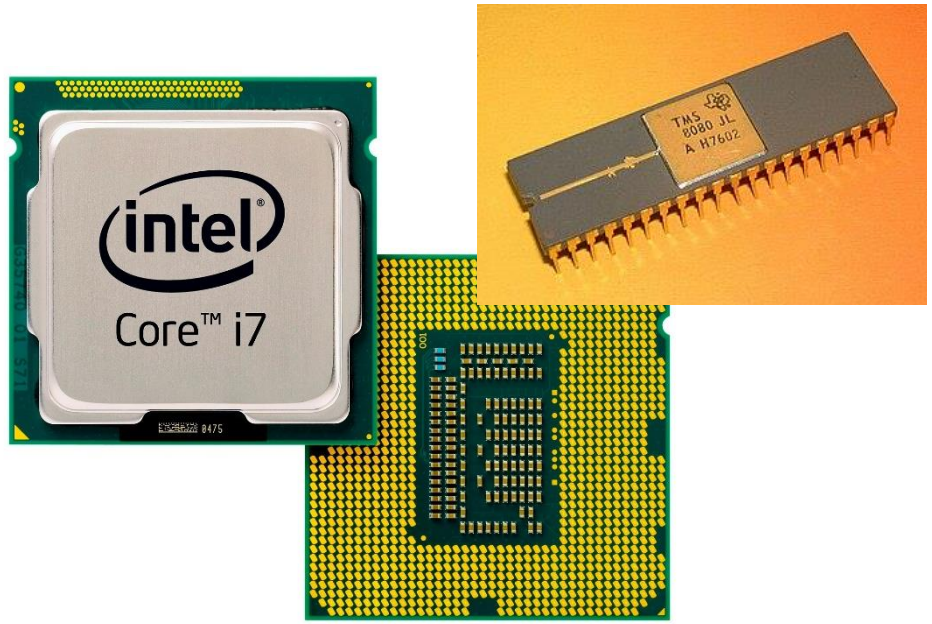
- All those parts are commercially available, so you could make your own Fitbit if you wanted to

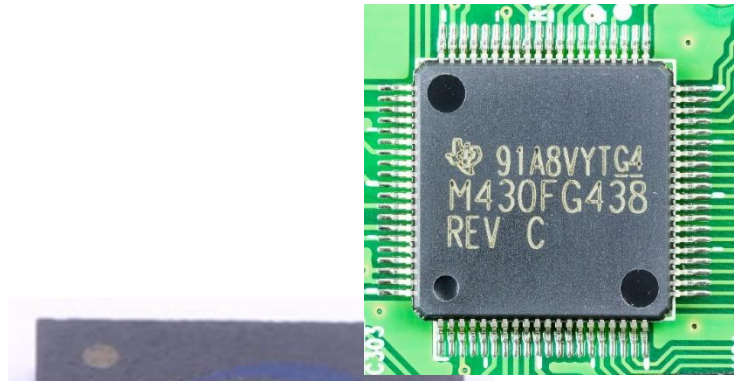
1. Make a circuit board
2. Buy those parts and solder them on
3. Write some embedded software
4. Make a plastic case
5. … build a big software backend for everything
6. You've got a Fitbit!

Categories of computer chips

- Microprocessor (MPU or Processor)
  - CPU
  - Most memory and peripherals are external

- Microcontroller (MCU)
  - CPU + Memory + Peripherals
  - Peripherals: separate hardware units within chip
    - Often I/O interfaces
  - Essentially, a microcontroller is almost an entire computer within a single chip





Break + Question

- Why do laptops/desktops use external memory chips instead of building it into the processor (like a microcontroller)?

Break + Question

- Why do laptops/desktops use external memory chips instead of building it into the processor (like a microcontroller)?
  - Flexibility
    - If it's built into the processor, you're limited to whatever they picked
  - Fabrication issues
    - DRAM and CMOS gates are somewhat different manufacturing processes (affects yield and costs)
    - DRAM takes up a lot of space (multiple chips for many GBs of RAM)
  - Some systems are doing this
    - Apple M4 includes separate RAM silicon dies in the same processor package

Modern Chips

- System on a Chip (SoC)
  - Processor with extensive peripherals
  - Example: Radios or ML Accelerators
  - Essentially multiple chips combined into one

- Evolution of increased complexity over time
  - Basically everything is an SoC these days

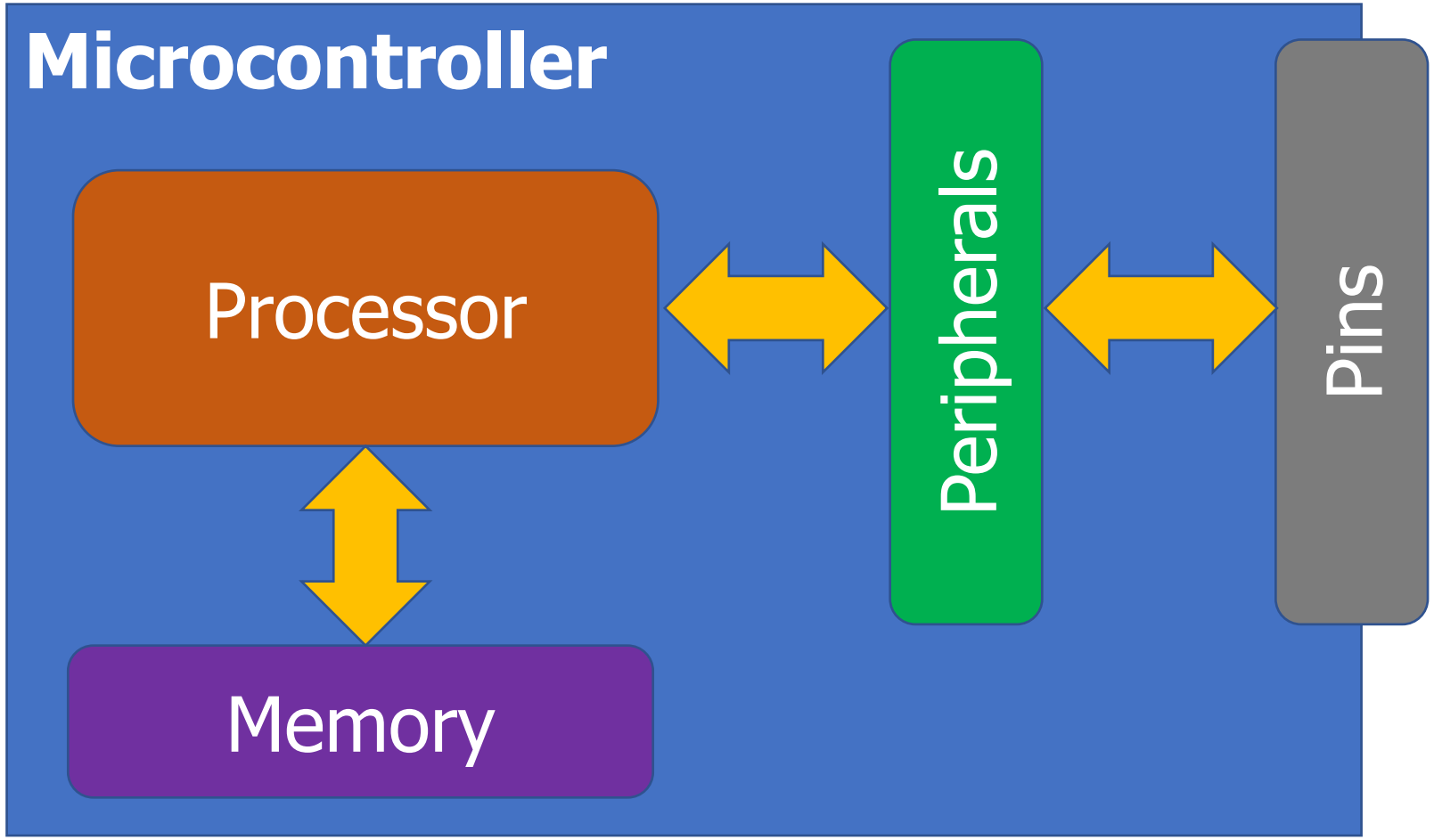
  - What we call Microcontrollers
    are still mostly self-contained and targeted and simpler, cheaper devices

  - What we call Microprocessors
    are more expensive, more performant, and usually require external memory

Outline

- Microcontrollers and Computers
- **Microcontroller Components**
  - **Processor**
  - Memory
  - Peripherals
  - Pins & Other stuff
- Microbit microcontroller
- Microcontroller History

Generic microcontroller block diagram



Processor

- Microcontroller needs to execute software instructions

- Probably just one single-core processor
  - Newest chips might have multiple (we'll come back to that)
  - Often a fairly basic processor like you'd learn in CE361 (Comp Arch)
  - Usually running a 1-100 MHz clock speeds
    - Compare to 3000-5000 MHz for modern desktop processors

- A point of consideration for microcontroller processors is what architecture they are

Background: Instruction Set Architecture

- ISA defines the actual instructions as well as the model for how the computer interacts with memory
  - ISA also includes the native word size (8, 16, 32, or 64-bit)
    - Defines size of registers and pointers
    - Also defines the limit of memory available to a system

- Does the ISA for a microcontroller matter if you're not programming in assembly? (and I promise we won't)

Background: Instruction Set Architecture

- ISA defines the actual instructions as well as the model for how the computer interacts with memory
  - ISA also includes the native word size (8, 16, 32, or 64-bit)
    - Defines size of registers and pointers
    - Also defines the limit of memory available to a system

- Does the ISA for a microcontroller matter if you're not programming in assembly? (and I promise we won't)
  - Yes
  - Differences in instruction efficiency and amount are important
  - Differences in compiler support are VERY important

ARM and RISC-V are common microcontroller architectures

- Common architectures
  - ARM (usually 32-bit) – also used in all smartphones, new Apple hardware
  - RISC-V (32-bit) – newer, free, open-source architecture

- Previously common
  - Various custom architectures created by companies
    - 8-bit or 16-bit on older systems

  - Still found in some microcontrollers
    - Original ESP32 used a 32-bit Xtensa architecture
    - New ESP32s mostly use a 32-bit RISC-V architecture

Other processor choices

- Some operations may or may not be supported depending on the processor
  - Multiplication/Division
  - Floating-point
  - Atomic operations
  - Vector / DSP operations

ARM Cortex M Series

- ARM processors commonly used in microcontrollers
- A number of options for increasing capabilities
- In practice:
  - Cortex-M0+ for low-end systems
  - Cortex-M7 for high capability systems
  - Cortex-M23 and M33 are newer replacements for those

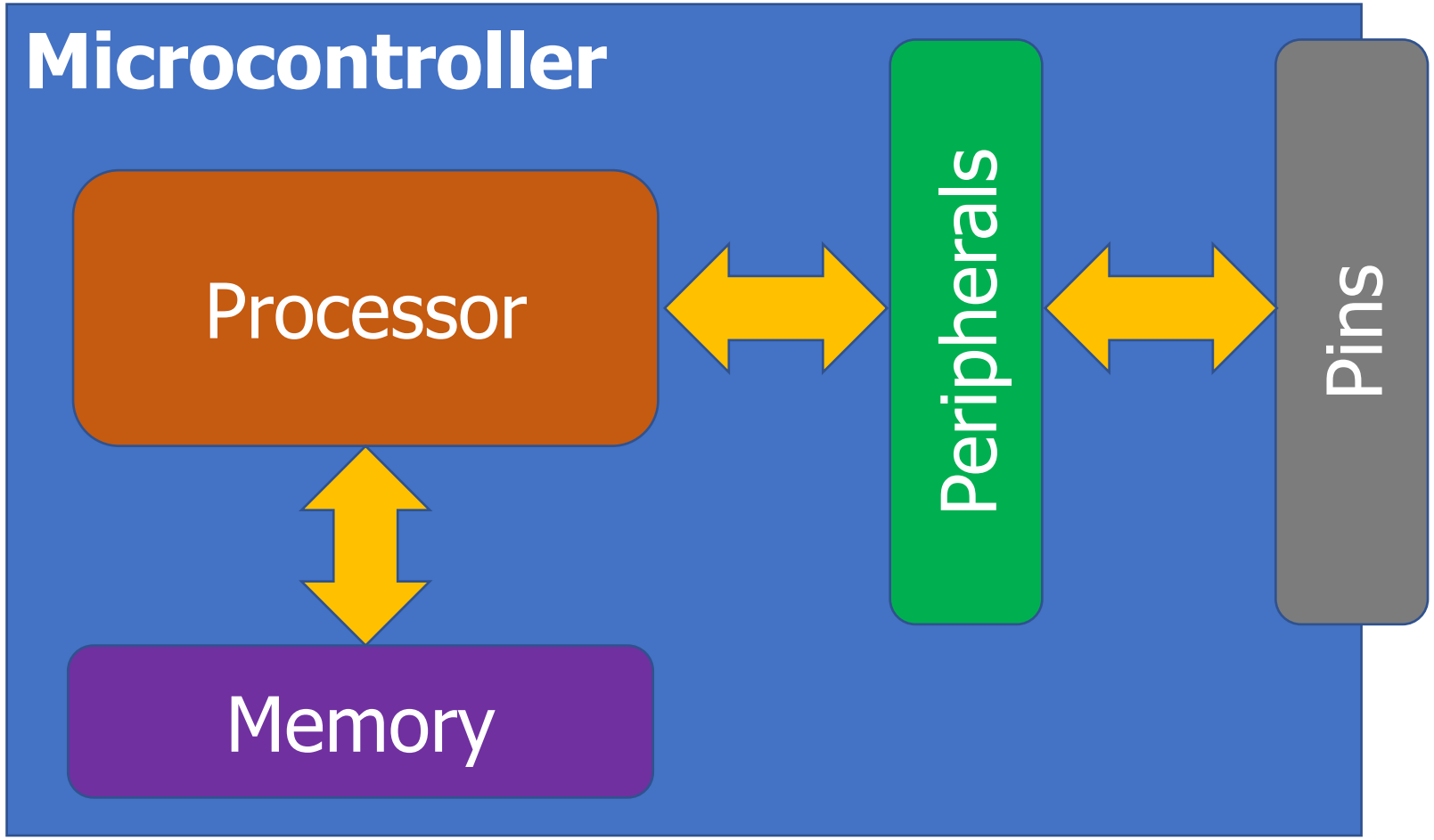
Arm Core	Cortex M0	Cortex M0+	Cortex M3	Cortex M4	Cortex M7	Cortex M23	Cortex M33
ARM architecture	ARMv6-M	ARMv6-M	ARMv7-M	ARMv7E-M	ARMv7E-M	ARMv8-M	ARMv8-M
Computer architecture	Von Neumann	Von Neumann	Harvard	Harvard	Harvard	Von Neumann	Harvard
Instruction pipeline	3 stages	2 stages	3 stages	3 stages	6 stages	2 stages	3 stages
Multiply instructions 32×32 = 32-bit result	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multiply instructions 32×32 = 64-bit result	No	No	Yes	Yes	Yes	No	Yes
Divide instructions 32/32 = 32-bit quotient	No	No	Yes	Yes	Yes	Yes	Yes
Saturated math instructions	No	No	Some	Yes	Yes	No	Yes
DSP instructions	No	No	No	Yes	Yes	No	Optional
Single-Precision (SP) floating-point instructions	No	No	No	Optional	Optional	No	Optional
Double-Precision (DP) floating-point instructions	No	No	No	No	Optional	No	No

https://en.wikipedia.org/wiki/ARM_Cortex-M

Outline

- Microcontrollers and Computers
- **Microcontroller Components**
  - Processor
  - **Memory**
  - Peripherals
  - Pins & Other stuff
- Microbit microcontroller
- Microcontroller History

Generic microcontroller block diagram



Multiple types of memory are needed

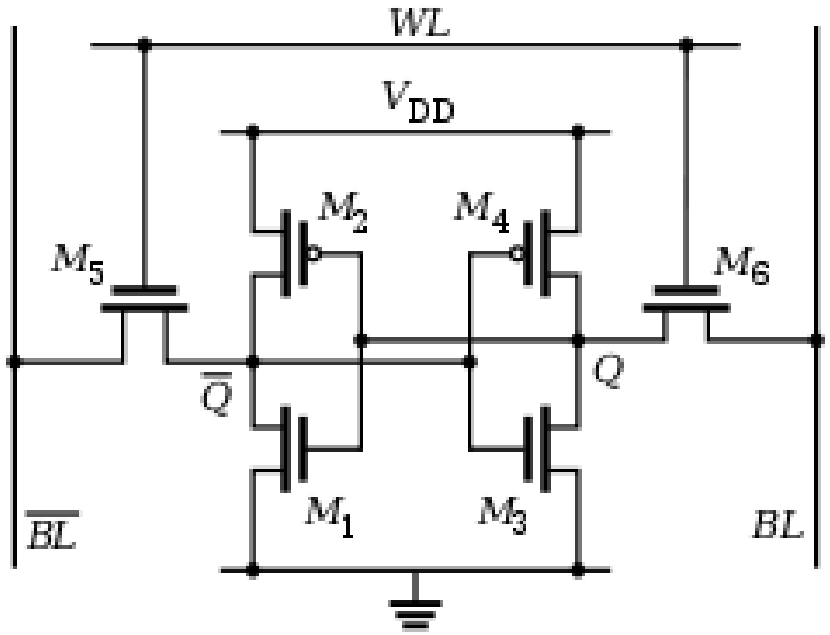
- Systems often have two types of memory
  - Volatile memory
    - Cheap and fast
    - Loses values without power
  - Non-volatile memory
    - Keeps values without power
    - Slower to access and modify
- On normal computers, these are RAM and Disk
  - Usually DRAM and Flash SSDs

Volatile memory: SRAM

- Same design as registers in traditional computer systems
  - No need to refresh it -> lower energy costs

- Embedded systems use SRAM for all of their memory (no DRAM)

- Tradeoffs:
  - Way less energy cost
  - Faster to access
  - More expensive and physically larger
    - So we have less of it in total



Non-Volatile memory: Flash (usually)

- Same design as SSDs and Flash drives

- Memory is stored with no energy costs
  - Reading is lowish energy and quick
  - Writing is high energy and very slow
    - Writing also has to occur in blocks (e.g. 512 bytes)

- Concern: Flash has a limited lifetime ~10,000 writes
  - Only writing to Flash when you load code? Essentially forever
  - Recording data to Flash every second? 2.8 hours
  - Result: most embedded systems don’t modify their own Flash memory

SRAM (RAM) versus Flash (ROM) memory

- Size and time difference between non-volatile and volatile memory is *significantly* reduced from traditional computing
  - Non-volatile store 2-10x larger than volatile
  - Single-cycle RAM. Maybe two-cycles to Flash (to read)

- Major difference: energy and writability
  - SRAM is low-energy to read and write (no refresh needed)
  - Flash is lowish-energy to read, but very high energy to write

- Use by software
  - Stack/Heap/Data sections (variables) in SRAM
  - Code goes in Flash (also const variables)

Memory design choices

- How much memory can we fit without making the microcontroller too high cost or too high power?
  - The answer has slowly increased over time
  - 20 years ago: 2 KB RAM – 32 KB Flash
  - Today: 256 KB RAM – 1024 KB Flash

- That's enough memory that you *usually* don't need to think about it
  - But not enough memory to *never* think about it

Differences from memory in standard computers

- How to provide memory safety?
  - Usually no virtual memory (all addresses are real addresses)
  - Nothing stops arbitrary memory overwriting
  - Modern systems: Memory Protection Unit
    - Specify a small number of ranges and permissions

- Hierarchical structure is not enforced
  - Same address space for RAM and Flash (very different from traditional)
  - Run instructions right inside of Flash
  - Keep variables in RAM (or `const` variables in Flash)

What about caches?

- Often no caches

- Caches are about improving performance
  - But embedded systems are not often performance-constrained
  - AND caches can make timing unreliable

- They exist in some high-end microcontrollers
  - Often instruction caches only
  - Often disabled by default

Break + Open Question

- How little memory (RAM and/or ROM) can a computer have and still be useful?

Break + Open Question

- How little memory (RAM and/or ROM) can a computer have and still be useful?
  - If we want to run C code, we need a stack and a global data section
  - Function calls may require local variables to be placed in the stack

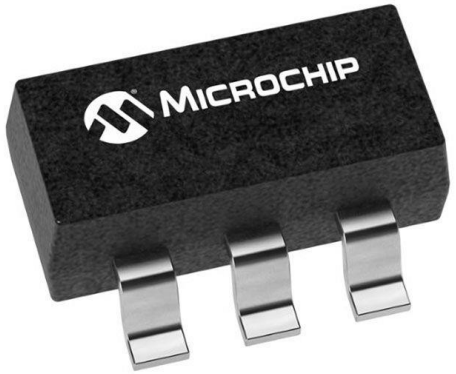
  - To use less memory, we'd need to write applications in registers
  - Most variables could *probably* fit in registers?
  - So the limit could get *very* low
    - Enough ROM space for some useful code + a tiny bit of RAM

Fall 2025: least memory

- How little memory might a microcontroller have?

- PIC10F200
  - 384 Bytes of ROM,
  - 16 Bytes of RAM

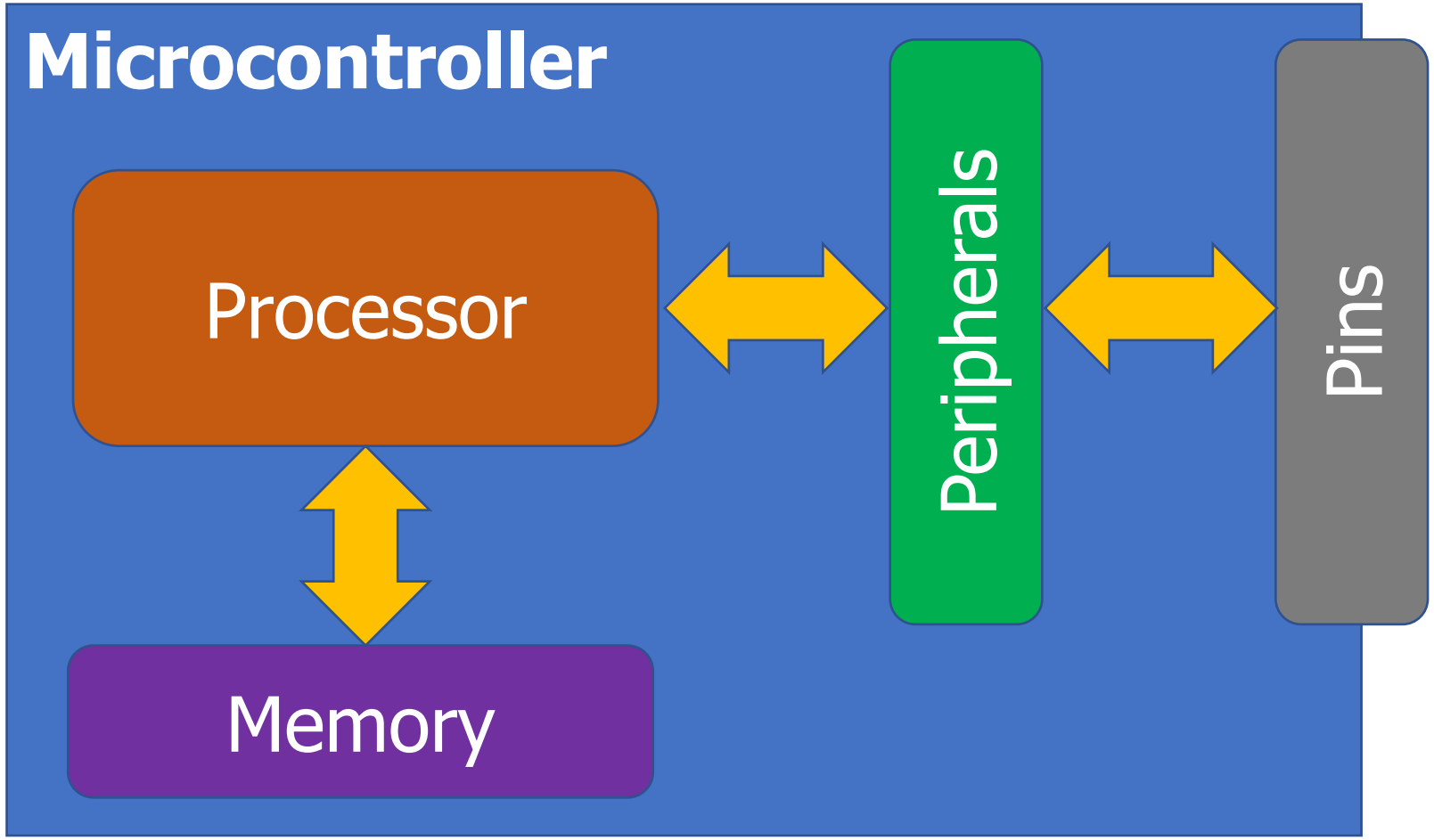
  - Still actively produced and for sale on Digikey
    - $0.58 a piece



Outline

- Microcontrollers and Computers
- **Microcontroller Components**
  - Processor
  - Memory
  - **Peripherals**
  - Pins & Other stuff
- Microbit microcontroller
- Microcontroller History

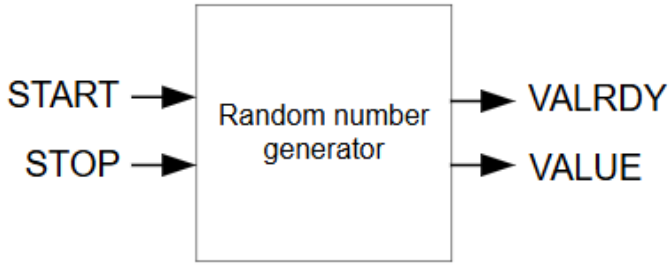
Generic microcontroller block diagram



Peripherals

- Hardware modules that perform some action
  - Specifically, modules within the microcontroller chip
  - NOT a separate chip, sensor, or inpu

- Examples
  - Random Number Generator (RNG)
    - Processor can request a random number
    - The RNG peripheral does some kind of actions to generate one
    - The RNG provides the random number to the processor



What kinds of peripherals exist?

- External connections
  - Used by processor to interact with outside world
  - Examples
    - Read from buttons and switches
    - Turn on/off LEDs
    - Measure a voltage
    - Connect to external sensors chips

- Internal systems
  - Used by processor to speedup or provide some action it’s not capable of
  - Examples:
    - Request random numbers
    - Encrypt or decrypt some information
    - Check the current time

What do microcontrollers usually have?

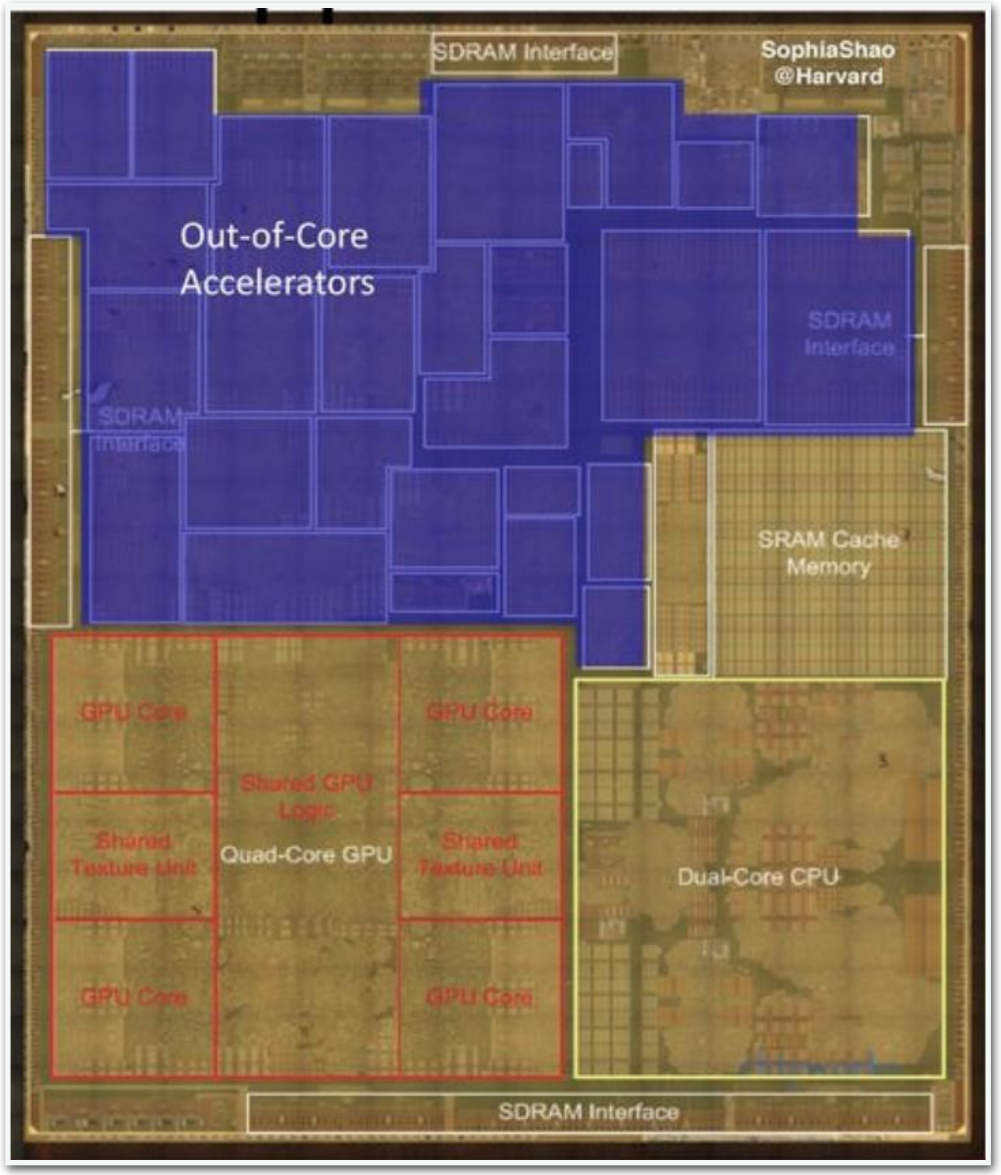
- Common examples
  - Digital inputs and outputs
  - Analog inputs and outputs
  - Messages over various communication protocols: UART, I2C, SPI
  - Set and check timers

- Less common (but not rare) examples
  - Cryptography accelerators
  - Complicated wire protocols (USB, CAN)
  - Wireless radios (BLE, 802.15.4, WiFi)

- We'll spend most of class learning various peripherals

Peripheral design choices

- More peripherals mean more capabilities
  - But also means more cost for the chip
  - Peripherals occupy silicon
  - Most peripherals will go unused for a given application…
- Flexibility tradeoffs within a given peripheral
  - One capability is easier to use
  - Many capabilities are more useful

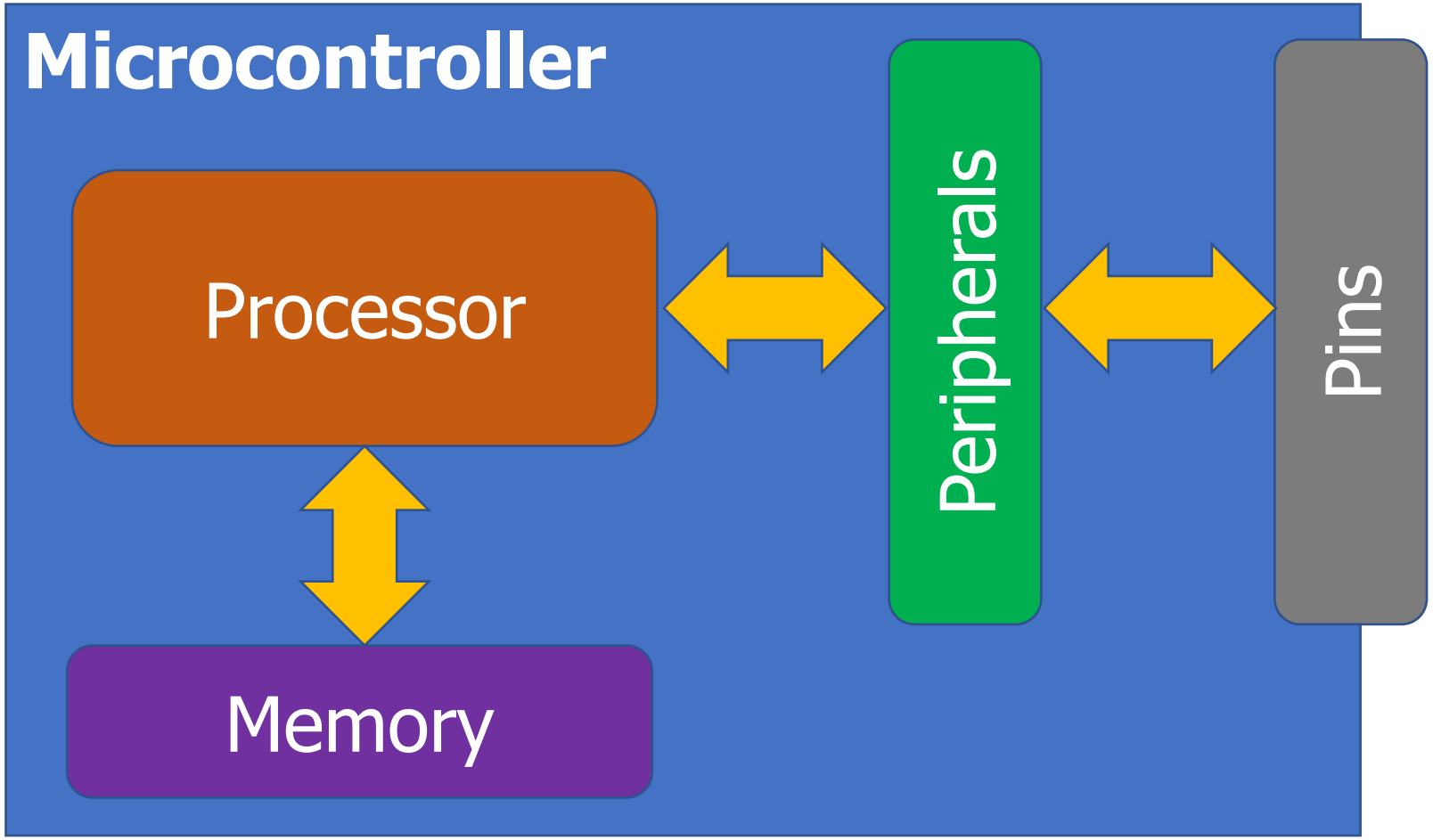


Apple A8 SoC

Outline

- Microcontrollers and Computers
- **Microcontroller Components**
  - Processor
  - Memory
  - Peripherals
  - **Pins & Other Stuff**
- Microbit microcontroller
- Microcontroller History

Generic microcontroller block diagram

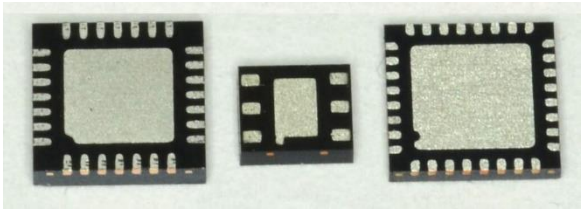


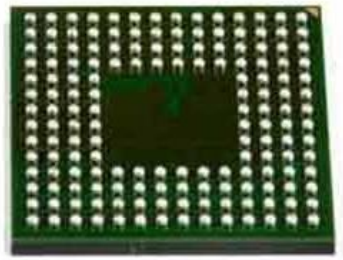
Pins

- Peripherals need to electrically connect to outside world
  - Attach to pins on the exterior of the chip

- More pins allow for more connections
  - At increased cost and size

- Pin layouts can add more pins for cheaper
  - But make soldering and debugging difficult





Internal connections to pins

- Peripherals need to be connected to external pins
  - Can any peripheral connect to any pin, or are there limited mappings?
  - Modern microcontrollers allow any-to-any connections

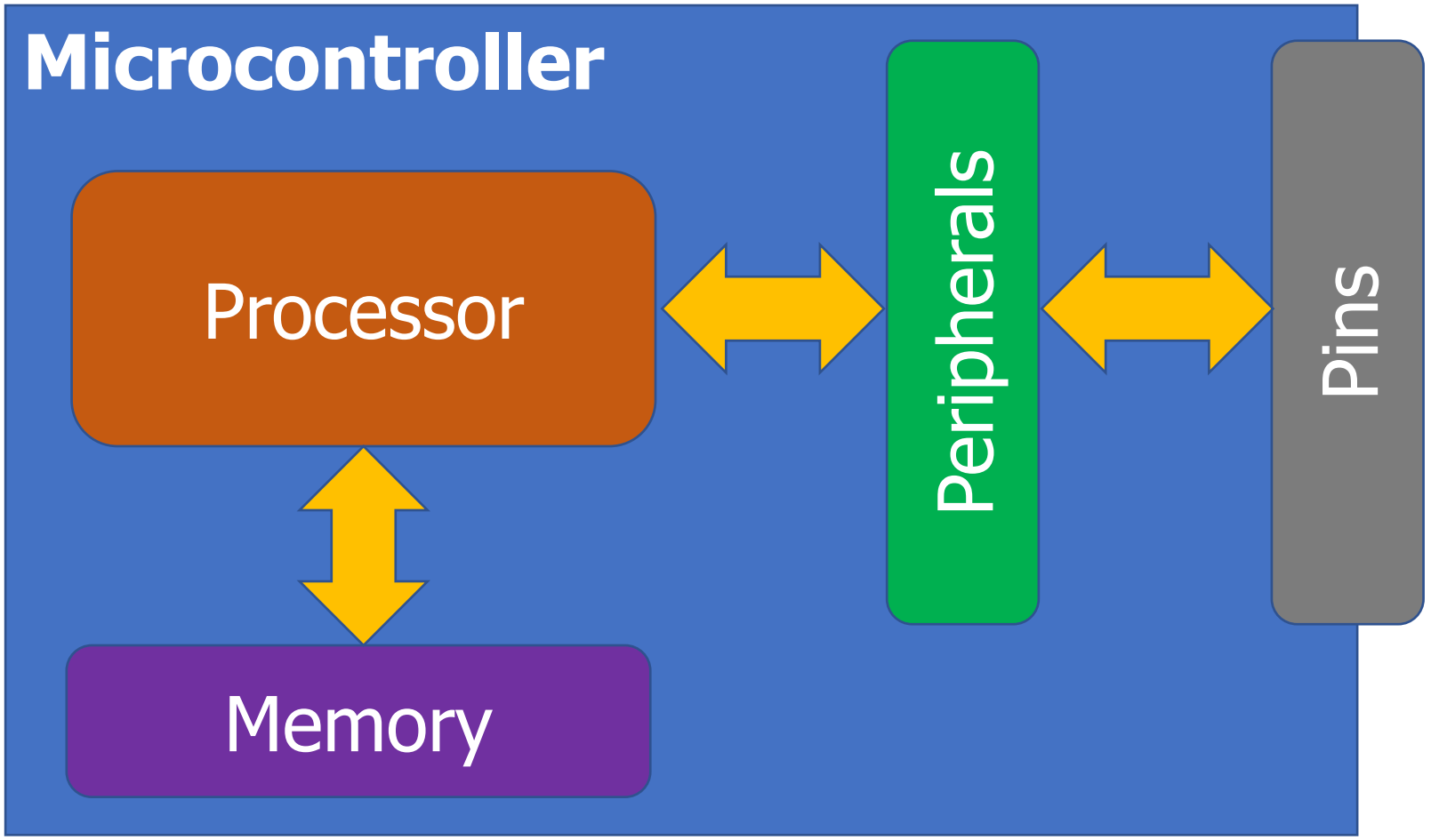
- Older MCUs had mapping tables and pin selection was more challenging

Table 3-1. 100-pin GPIO Controller Function Multiplexing (Sheet 1 of 4)

ATSAM4LC		ATSAM4LS		Pin	GPIO	Supply	GPIO Functions						
QFN	VFBGA	QFN	VFBGA				A	B	C	D	E	F	G
5	B9	5	B9	PA00	0	VDDIO							
6	B8	6	B8	PA01	1	VDDIO							
12	A7	12	A7	PA02	2	VDDIN	SCIF GCLK0	SPI NPCS0					CATB DIS
19	B3	19	B3	PA03	3	VDDIN		SPI MISO					
24	A2	24	A2	PA04	4	VDDANA	ADCIFE AD0	USART0 CLK	EIC EXTINT2	GLOC IN1			CATB SENSE0
25	A1	25	A1	PA05	5	VDDANA	ADCIFE AD1	USART0 RXD	EIC EXTINT3	GLOC IN2	ADCIFE TRIGGER		CATB SENSE1
30	C3	30	C3	PA06	6	VDDANA	DACC VOUT	USART0 RTS	EIC EXTINT1	GLOC IN0	ACIFC ACAN0		CATB SENSE2

What haven’t we talked about?

- **Power**
- **Programming**
- **Others?**
  - Clocks
  - Antennas



Powering microcontrollers

- Usually require a specific voltage
  - E.g. 5 volts, 3.3 volts, 1.8 volts
  - Must be stable and supply enough current (or MCU "browns out")
  - Noisy power supply can be an issue

- Some microcontrollers have wider ranges of acceptable voltages
  - Need to pay attention to acceptable range on I/O though

Programming microcontrollers

- JTAG (Joint Test Action Group)
  - Hardware built into the microcontroller for testing purposes
  - Can arbitrarily read/write memory
  - Can single step process too, at runtime!
    - GDB can connect to it! (sort of)

- Bootloaders
  - Software runs on the microcontroller at boot that waits a short time for someone to contact it and upload code
    - Via I/O pins
  - Convenient, but sometimes unreliable

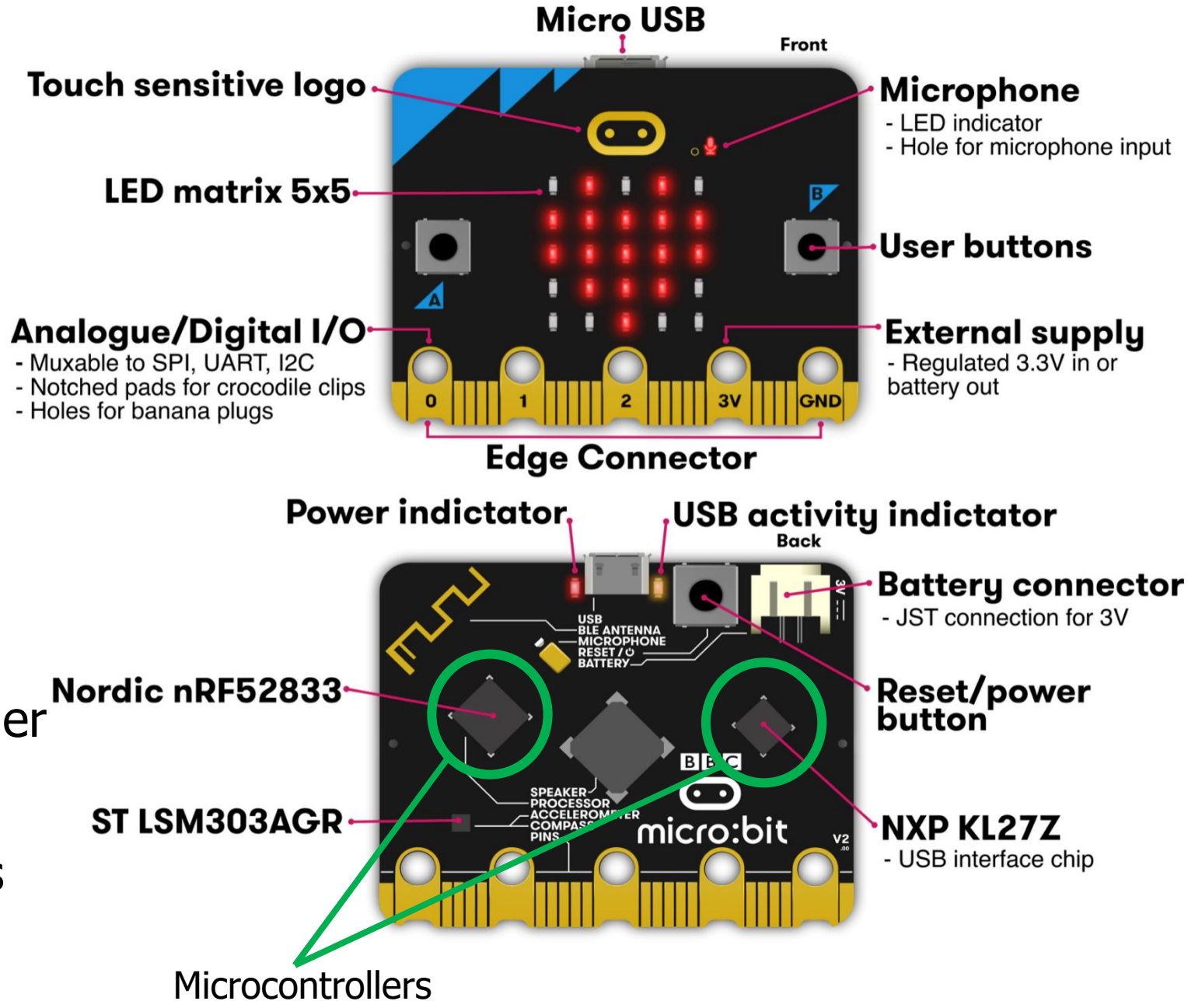
Outline

- Microcontrollers and Computers
- Microcontroller Components
  - Processor
  - Memory
  - Peripherals
  - Pins & Other stuff
- **Microbit microcontroller**
- Microcontroller History

Micro:bit v2

- Circuit board
  - Entire thing
  - a.k.a “Dev Board”
  - a.k.a PCB (Printed Circuit Board)

- Microcontroller
  - The computer on it
  - Microbit has two
    - One as a programmer for the nRF (USB->JTAG)
    - One for applications that we will use

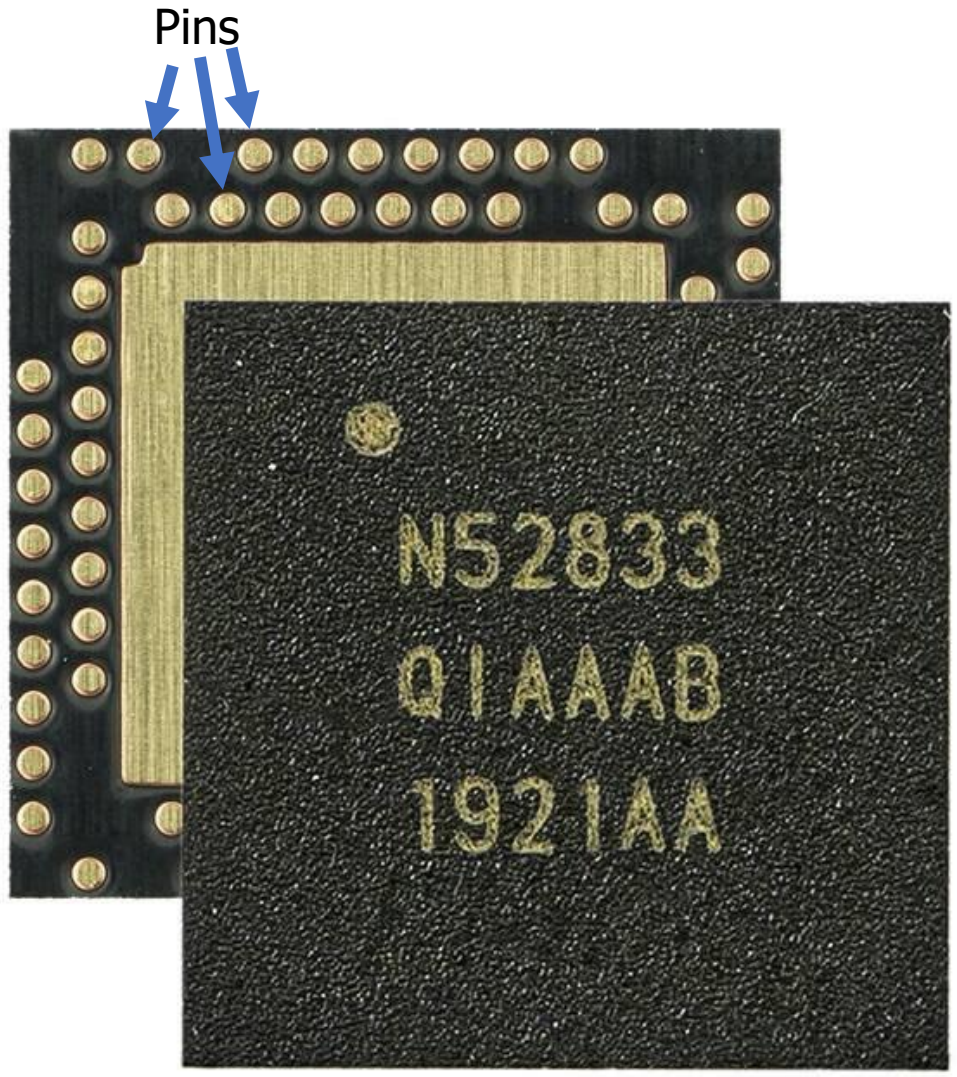


nRF52833 – Microcontroller on the Micro:bit v2

- Processor: 32-bit ARM Cortex-M4F
  - 32-bit registers and pointers, Floating-point support
  - 64 MHz clock frequency
- Memory:
  - 128 KB SRAM
  - 512 KB Flash
- Peripherals:
  - Various peripherals
    - ADC, PWM
    - I2C, UART, SPI, USB
    - RNG, 32-bit Timers, Watchdog, Temperature
    - 2.4 GHz Radio: Bluetooth Low energy / 802.15.4
- Pins:
  - 73 total pins, up to 42 I/O pins that can connect to peripherals

Pins on the nRF52833

- 73 individual pins
  - Plus a big ground pad
- Some are used for special purposes
  - Power (16)
  - External Oscillator (2)
  - Debugging (2)
  - RF Antenna (1)
  - USB (2)
  - Not connected at all (8)
- Remaining 42 pins can be used as General Purpose I/O (GPIO)
  - Digital I/O
  - Or connected to other peripherals



Accessing nRF52833 pins in software

- Pins are collected into “Ports”
  - Each Port has 32 pins

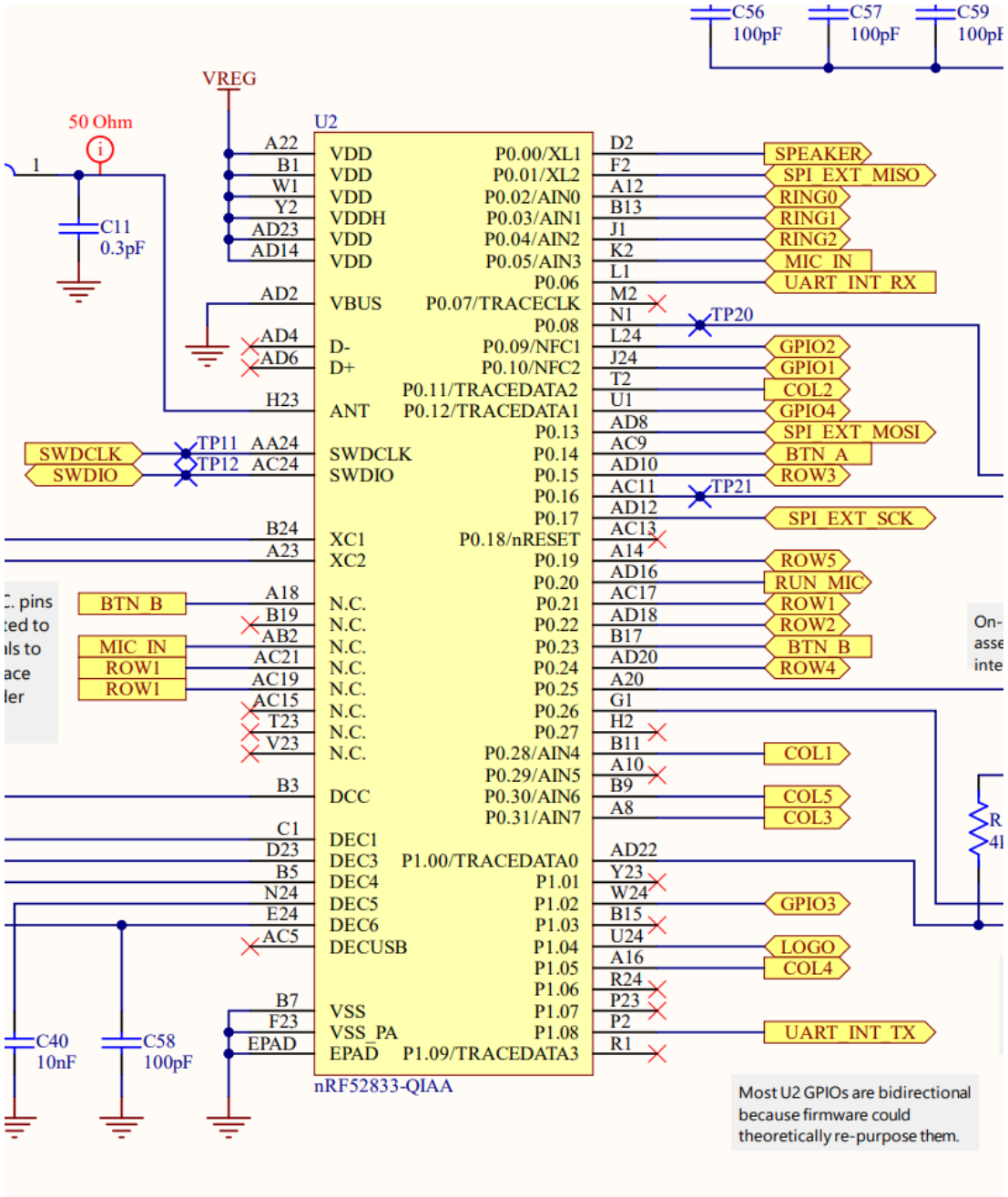
- 42 total GPIO pins
  - Port 0 – pins 0-31
  - Port 1 – pins 0-9

- Referred to as P`[port].[pin]`
  - For example: P0.20, P1.08, P0.00

Pins on the Microbit

- The Microbit uses the nRF52833, and then connects individual pins from the Microcontroller to certain parts of the board
  - Every board does this
  - The only way to know this connections is to look at the documentation

- For example:
  - BTN_A connects to P0.14
  - BTN_B connects to P0.23



In our software library

- Header file gives names to each pin based on how the Microbit uses it
  - https://github.com/nu-ce346/nu-microbit-base/blob/main/software/boards/microbit_v2/microbit_v2.h

- You can use these names directly in your code where you need the pin number

```
// LED with microphone. Drive with high strength
#define LED_MIC NRF_GPIO_PIN_MAP(0,20)

// LED Rows. Drive high to enable LED
#define LED_ROW1 NRF_GPIO_PIN_MAP(0,21)
#define LED_ROW2 NRF_GPIO_PIN_MAP(0,22)
#define LED_ROW3 NRF_GPIO_PIN_MAP(0,15)
#define LED_ROW4 NRF_GPIO_PIN_MAP(0,24)
#define LED_ROW5 NRF_GPIO_PIN_MAP(0,19)

// LED Columns. Drive low to enable LED
#define LED_COL1 NRF_GPIO_PIN_MAP(0,28)
#define LED_COL2 NRF_GPIO_PIN_MAP(0,11)
#define LED_COL3 NRF_GPIO_PIN_MAP(0,31)
#define LED_COL4 NRF_GPIO_PIN_MAP(1, 5)
#define LED_COL5 NRF_GPIO_PIN_MAP(0,30)

// Push buttons
#define BTN_A NRF_GPIO_PIN_MAP(0,14)
#define BTN_B NRF_GPIO_PIN_MAP(0,23)

// Touch sensitive pins
#define TOUCH_LOGO  NRF_GPIO_PIN_MAP(1, 4)
#define TOUCH_RING0 NRF_GPIO_PIN_MAP(0, 2)
#define TOUCH_RING1 NRF_GPIO_PIN_MAP(0, 3)
#define TOUCH_RING2 NRF_GPIO_PIN_MAP(0, 4)
```

To the datasheet!

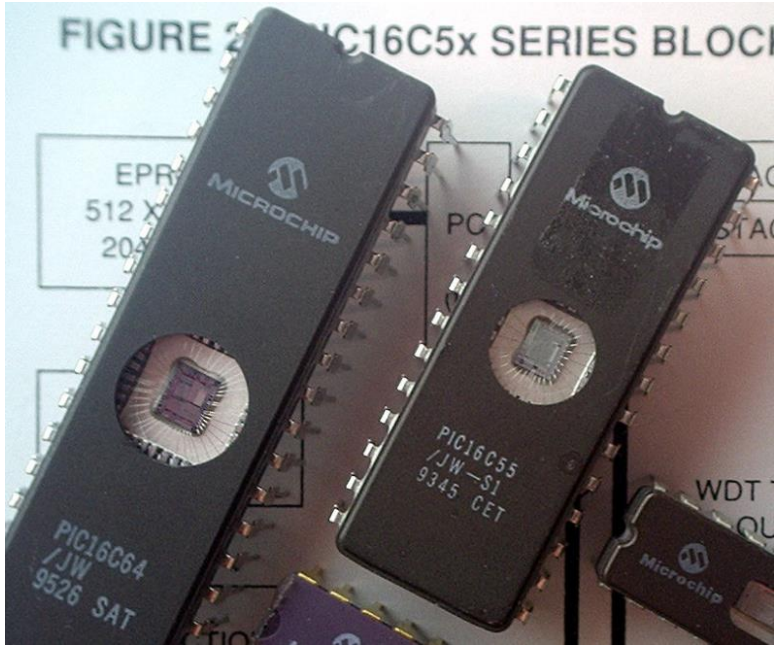
- nRF52833 Product Specification
  - Online: https://docs.nordicsemi.com/bundle/ps_nrf52833/page/keyfeatures_html5.html
  - PDF: https://docs-be.nordicsemi.com/bundle/ps_nrf52833/attach/nRF52833_PS_v1.7.pdf

Outline

- Microcontrollers and Computers
- Microcontroller Components
  - Processor
  - Memory
  - Peripherals
  - Pins & Other stuff
- Microbit microcontroller
- **Microcontroller History**

Older microcontrollers (90s-00s)

- Focus: cheap, small computer systems



- PIC
  - 8, 16, and 24-bit MIPs architecture
  - PICAXE available with Basic interpreter

- AVR
  - 8-bit custom architecture (AVR architecture)
  - Used in Arduinos
  - AT Tiny 4: $0.30 per unit
    - 4 I/O pins, 32 Bytes RAM, 512 Bytes ROM

More capable microcontrollers (00s-10s)

- Focus: diversify into extreme low power and more capable systems

- MSP430
  - 16-bit custom architecture (MSP430 architecture)
  - Capable, but also extremely low power
    - <1 μA sleep current

- STM32, Atmel SAM series (Cortex-M0, M3, M4, M4F)
  - 32-bit ARM architecture (ARMv7)
    - Leverage success of ARM on smartphones
  - Every peripheral under the sun. Plus a variety of memory choices

Modern system-on-chips (10s-20s?)

- Focus: increase memory and capability
  - Include radios in the same chip

- nRF51 and nRF52 series
  - 32-bit ARM microcontrollers (Cortex M)
  - Include 2.4 GHz radio: Bluetooth Low Energy and 802.15.4/Thread

- Others have followed this same path
  - TI CC26XX
  - Apollo3
  - STM32WL

Future microcontrollers (20s and beyond)

- Multi-core systems
  - Not for performance, but for separation of concerns
    - Run radio code on one core
    - Application code goes on the other core
  - Also allows BIG.little architecture
    - Higher performance core when interpreting data
    - Lower performance core for sampling sensors
  - nRF54 series really emphasizes this
    - Multiple BIG ARM processors and little RISC-V processors

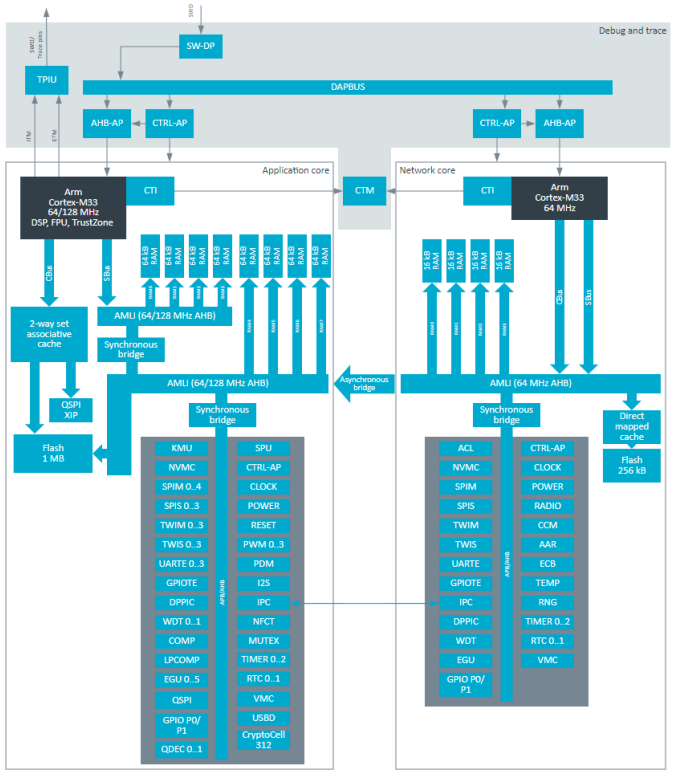
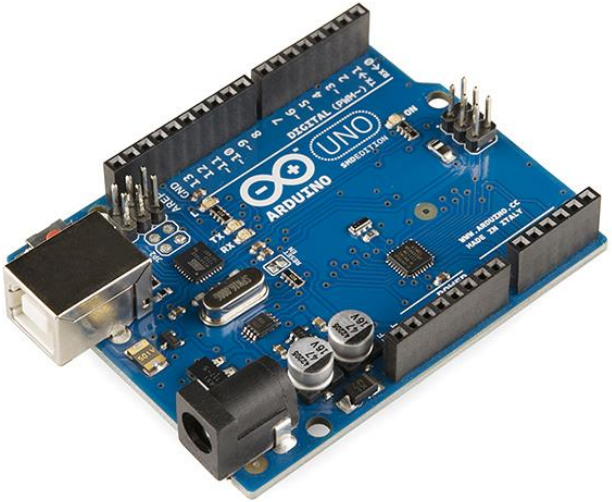


Figure 1. Simplified block diagram

Popular components rarely die

- Majority of popular older options still exist
  - Designers can trade off cost, capability, and efficiency alongside “modern features”

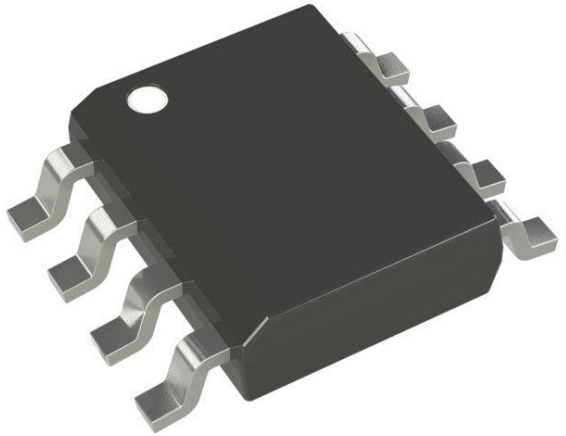


- Upgrading for an existing product is unlikely
  - Large cost, little benefit

- Example: Arduino still primarily uses AVR
  - Works just fine for its needs
  - Also releases new boards with nRF52s (Arduino Nano 33 BLE)

Fall 2025: cheapest microcontroller

- PIC16F15213
  - 16-bit custom architecture running at 32 MHz
  - 256 KB of RAM
  - 3.5 KB of Flash
  - 8 pins (5 for I/O)



  - $0.38 per microcontroller
    - Still an “active” chip being manufactured

- Can get significantly cheaper prices buying at scale

Outline

- Microcontrollers and Computers
- Microcontroller Components
  - Processor
  - Memory
  - Peripherals
  - Pins & Other stuff
- Microbit microcontroller
- Microcontroller History