Embedded Systems

Lecture 2.

Hardware Software Architecture and Software Dev.

© Lothar Thiele

Computer Engineering and Networks Laboratory

Michele Magno

D-ITET center for project based learning

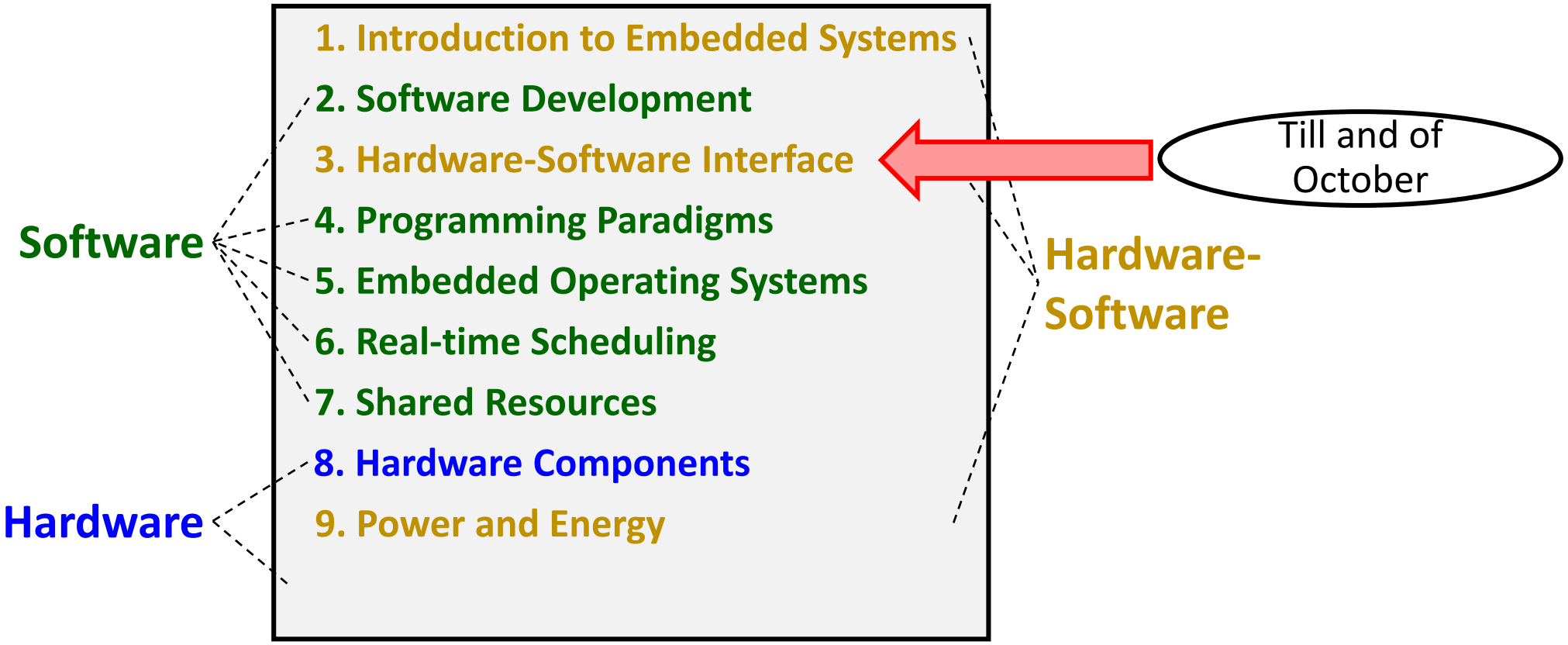
ETH

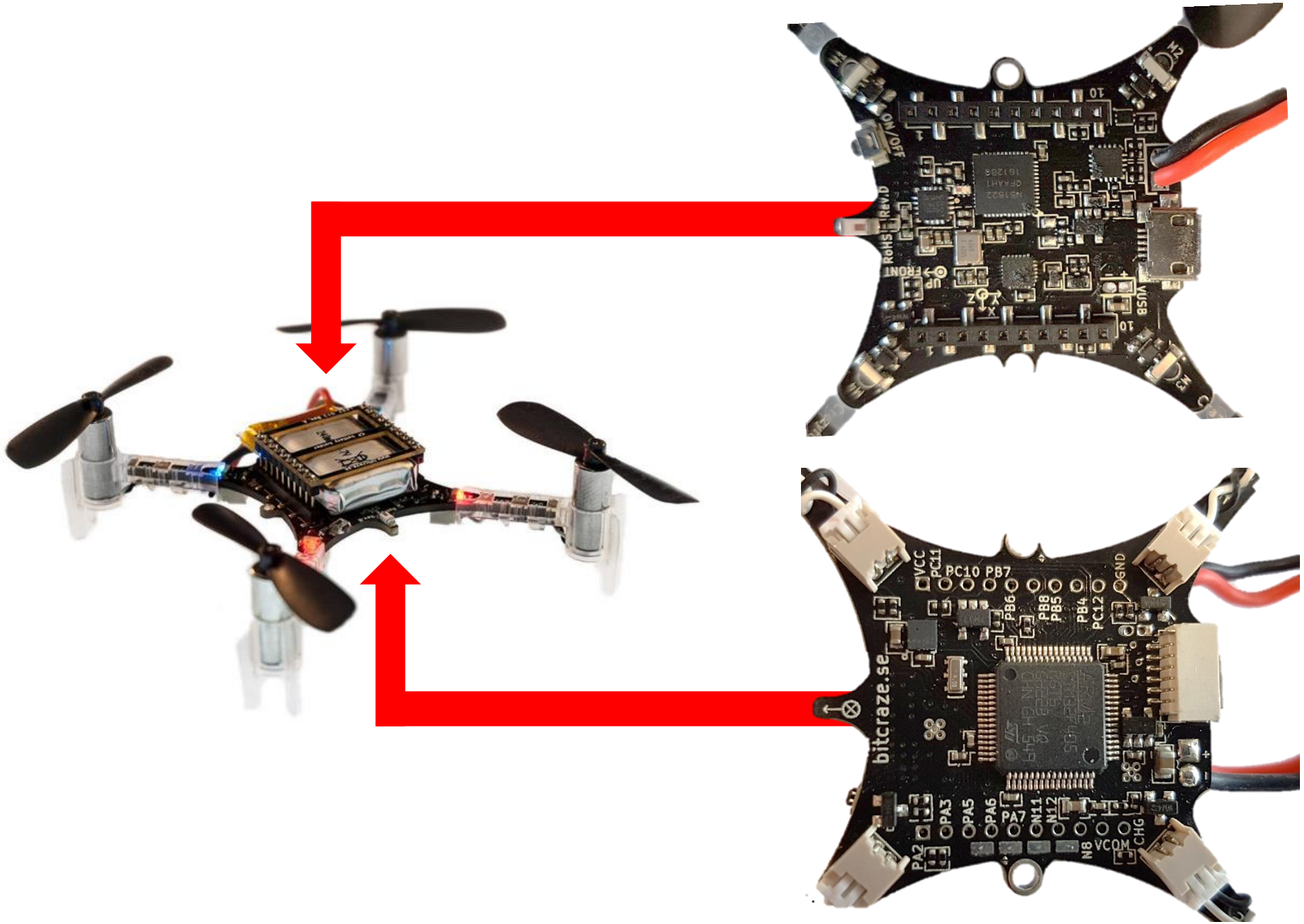
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Source the Slides or adaptation from:
Embedded Systems. P. Marwedel

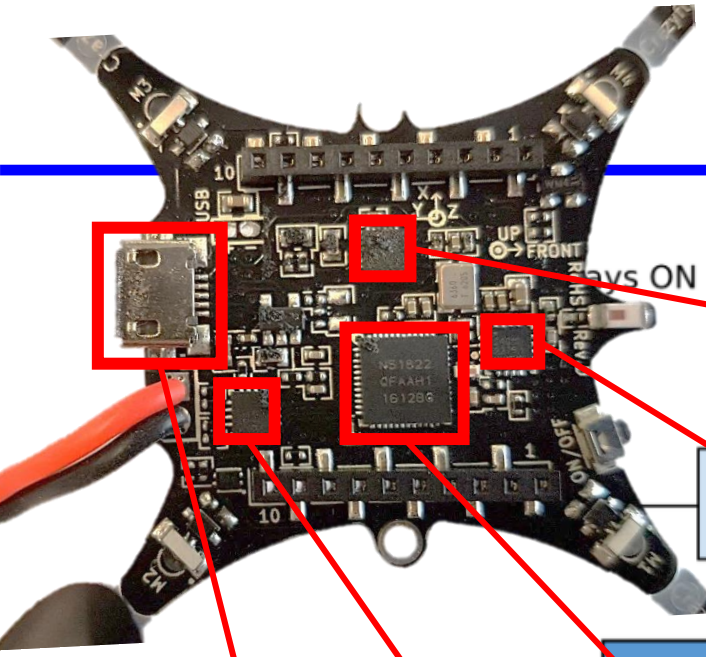
Do you Remember ?

Where we are …


1. Introduction to Embedded Systems
2. Software Development
3. Hardware-Software Interface
4. Programming Paradigms
5. Embedded Operating Systems
6. Real-time Scheduling
7. Shared Resources
8. Hardware Components
9. Power and Energy
Software
Hardware
Hardware-Software
Till and of October


ON/OFF
RoHS
REV.D
FRONT
VUSB
bitcraze.se
VCC
PC11
PC10 PB7
GND
PB6
PB8
PB5
PB4
PC12
PA2
PA3
PA5
PA6
PA7
CN11
CN12
N8
VCOM
CHG

High-Level Physical View



ays ON power domain

Power switched by nRF51 (VCC)

RF power amplifier

10DOF IMU
- 3-axis accelerometer
- 3-axis gyro
- 3-axis magnetomer
- Pressure sensor

I2C

Push button

nRF51822
- 16MHz Cortex-M0
- 16kB RAM, 256KB Flash
- BLE and NRF radio

UART

STM32F405
- 168MHz Cortex-M4
- 196kB RAM, 1MB Flash

PWM

Motor driver

I2C

SPI/I2C/GPIO/PWM

Wkup/OW/GPIO

Power supplies and battery charger

+5V

Charge/VBAT/VCC

Expansion port

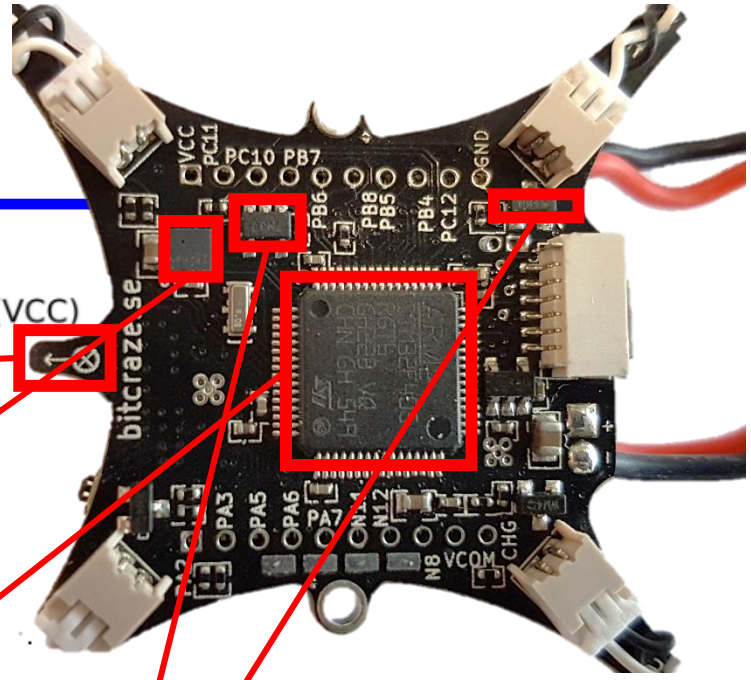
EEPROM

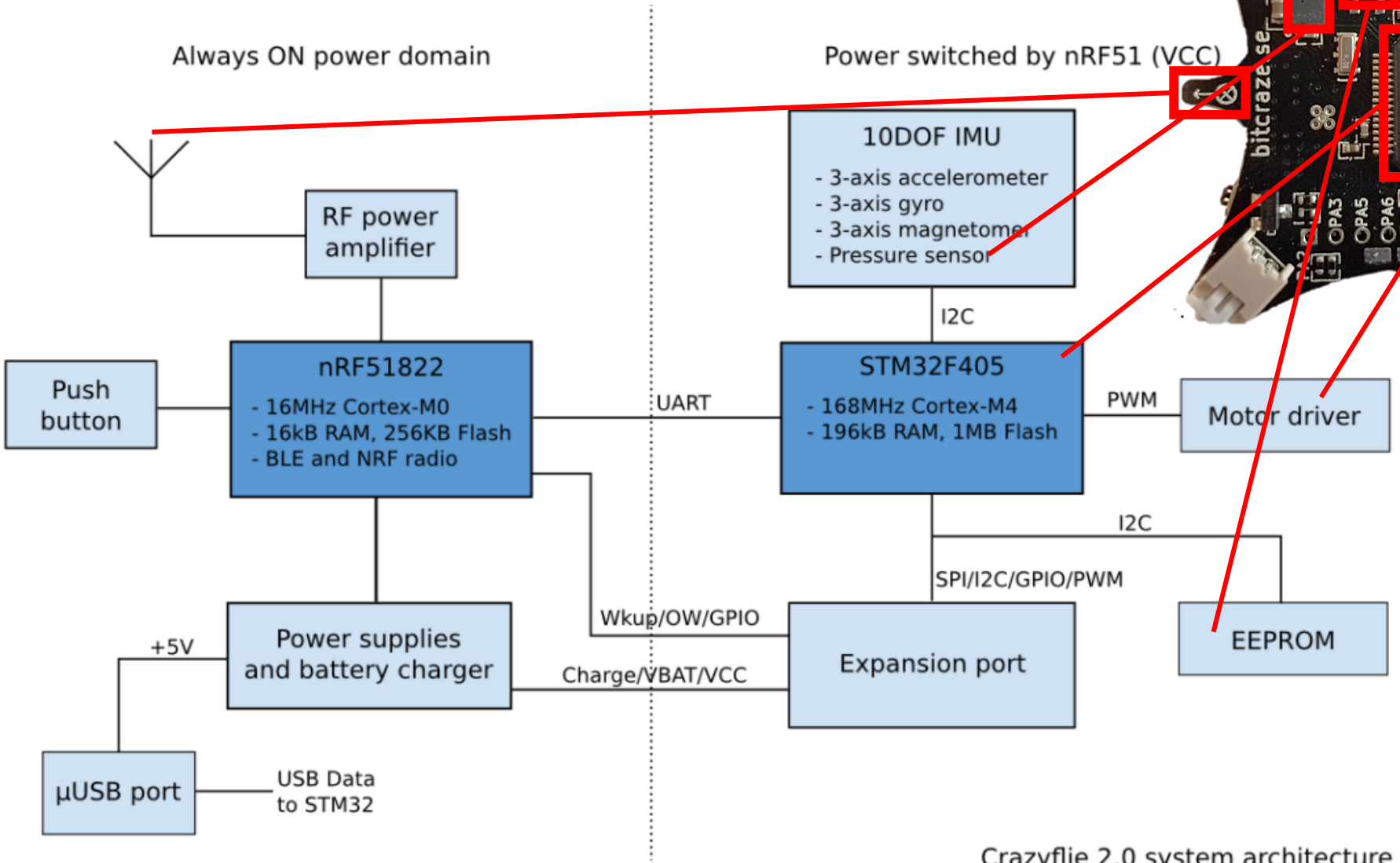
µUSB port

USB Data to STM32

Crazyflie 2.0 system architecture

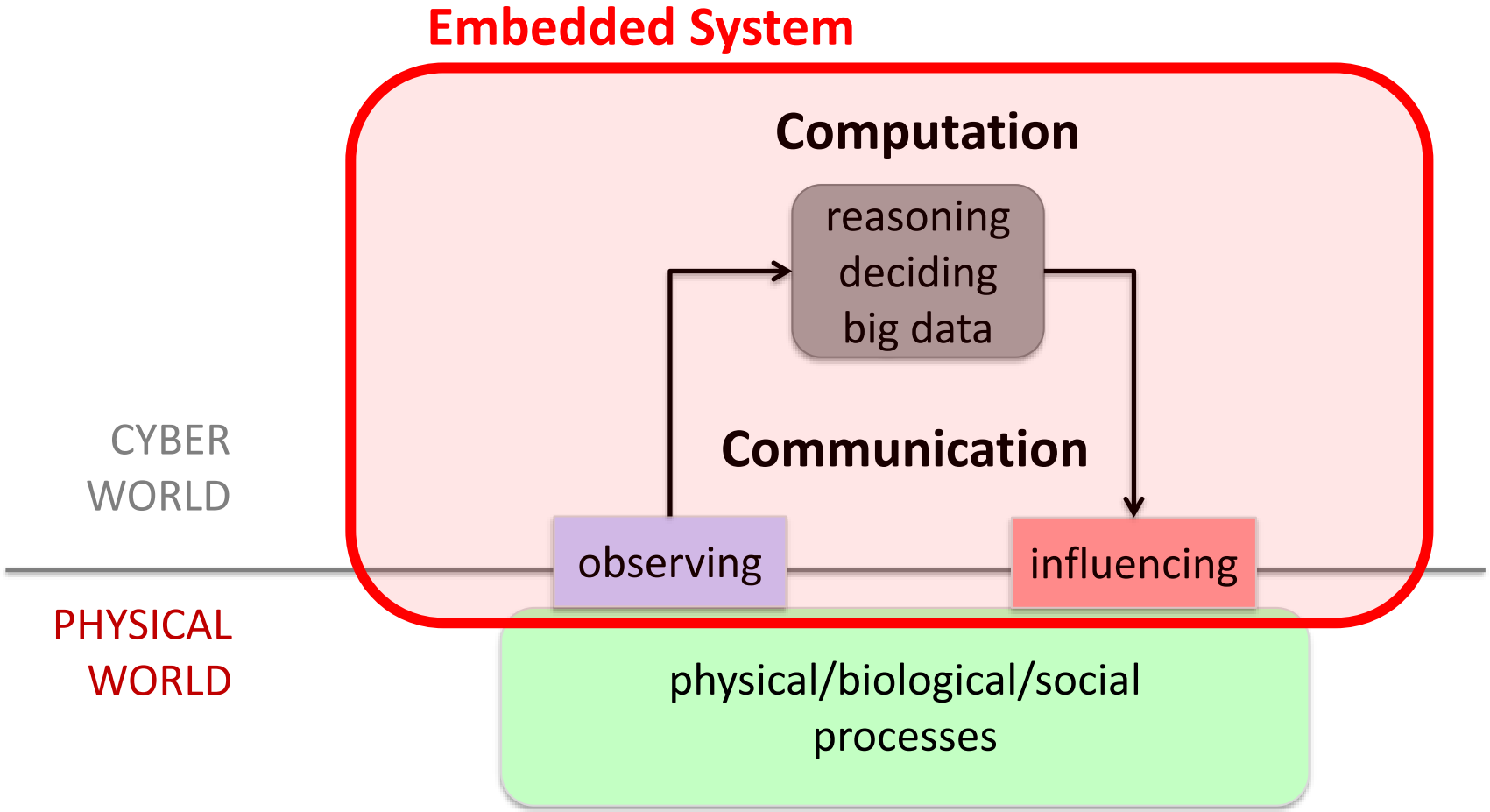
High-Level Physical View





Crazyflie 2.0 system architecture

Remember? Embedded System Overview

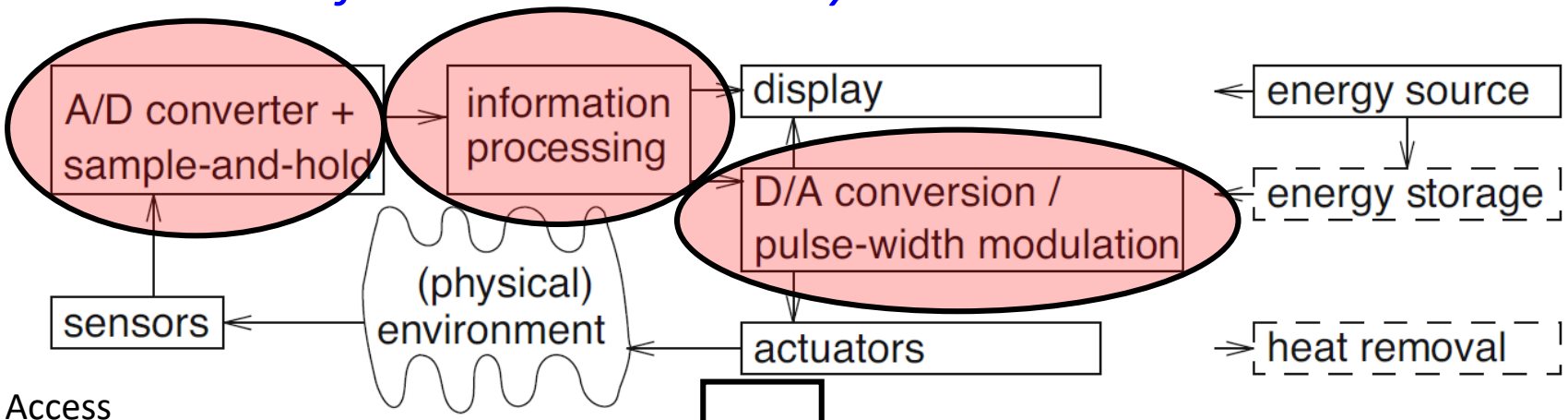


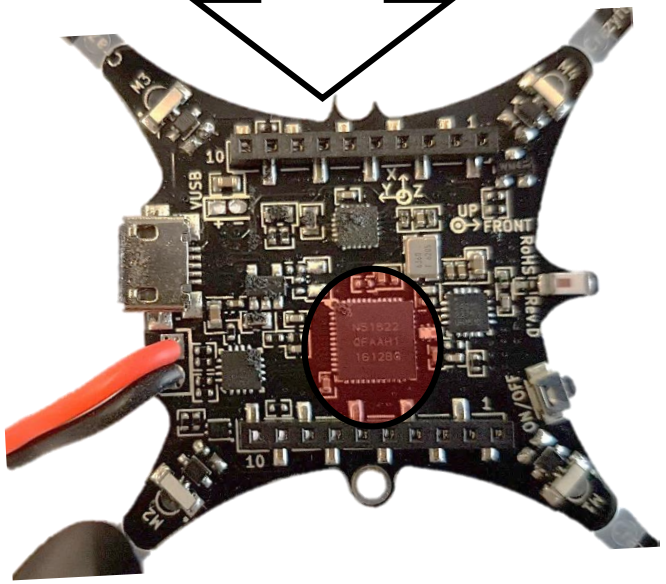
Use feedback to influence the dynamics of the physical world by taking smart decisions in the cyber world

What you will learn In next 4 weeks.

Hardware-Software Architecture and Interfaces in Embedded Systems

- ***Processing Unit***
- *Storage*
  - SRAM / DRAM / Flash
  - Memory Map
- *Input and Output*
  - UART Protocol
  - Memory Mapped Device Access
  - SPI Protocol
  - ***Sensors Interface Analog Digital Conversion***
  - ***Digital Analog Converter (DAC)***
- *Interrupts*
- *Clocks and Timers*
  - Clocks
  - Watchdog Timer
  - System Tick
  - Timer and PWM





What type of Processing unit?

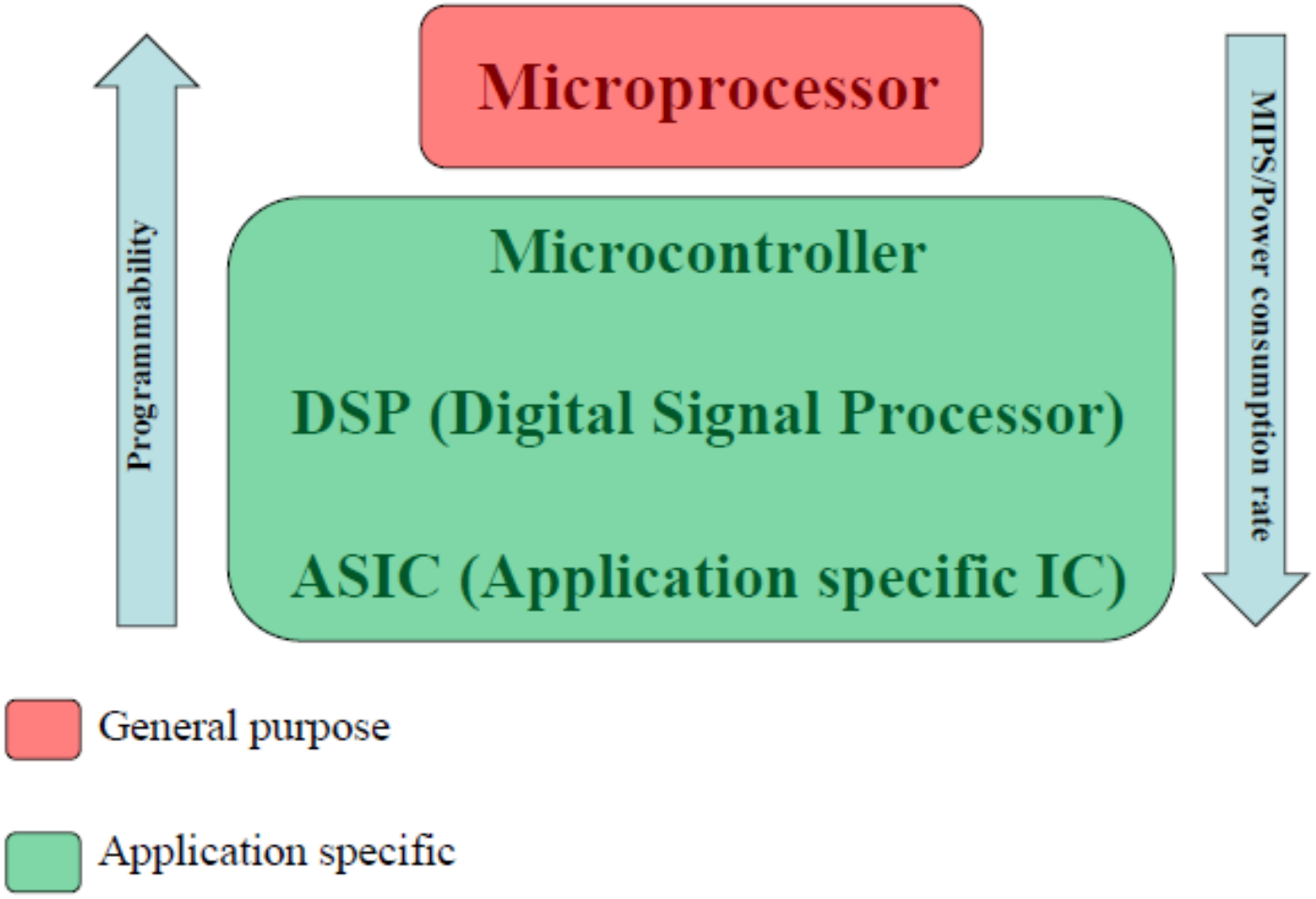
- **Application-specific integrated circuits (ASICs)**
  - Energy efficient
  - Low flexibility, designed for a specific task-application high cost to market
- **Digital Signal Processor**
  - Specialized Microprocessor (i.e. to audio streams, video stream etc.)
  - Cost-energy effective
- **Microcontroller**
  - Low cost and fully integrated solutions including control
  - Toward general purpose but still general purpose
- **Microprocessors**
  - General purpose (i.e. Intel Processors)
  - High performance
- **SoC/GPU/Others**
  - Can include all the above architecture with also GPUs.
- **FPGA**
  - Field Programmable Gate Array

FPGA drone

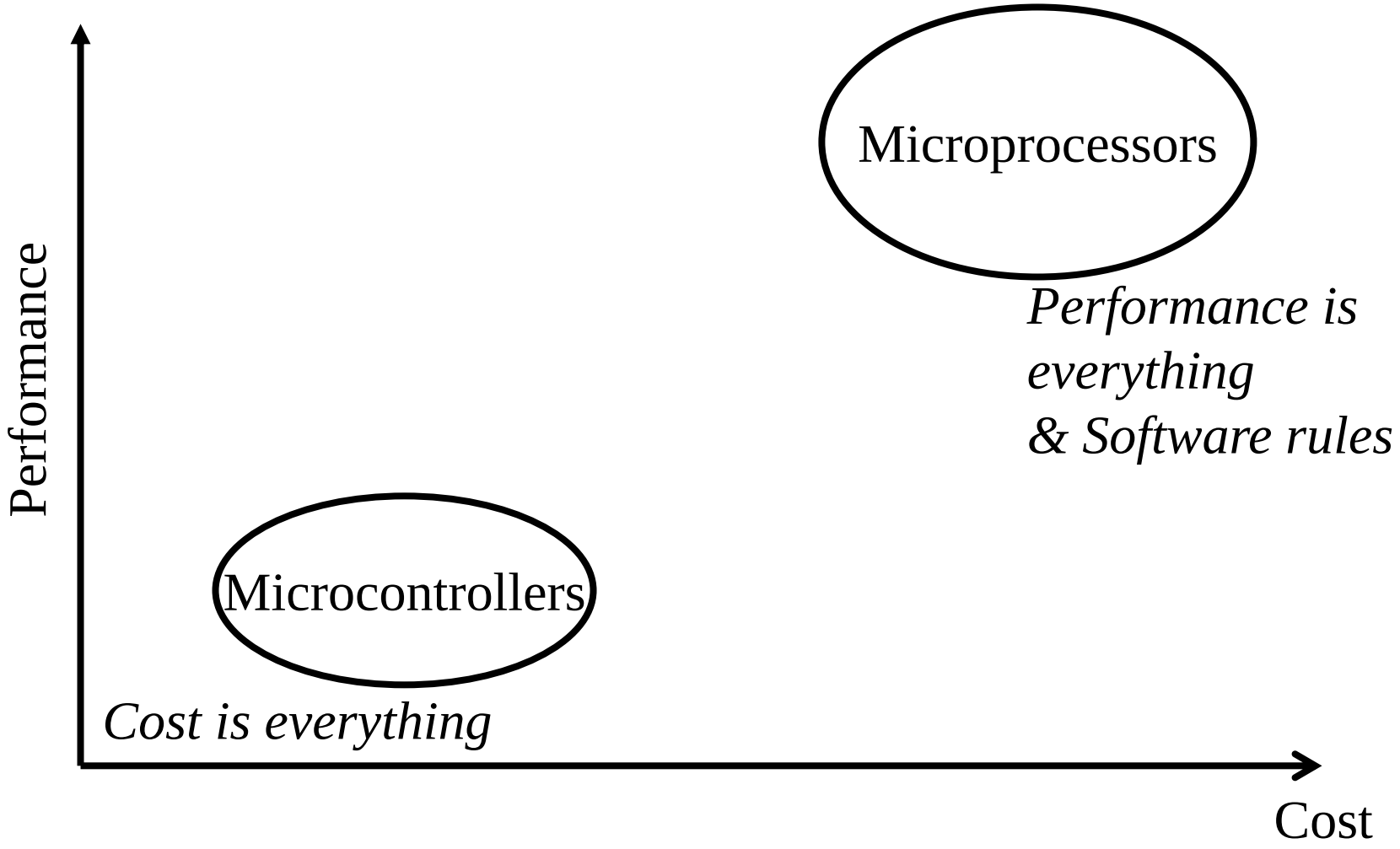
GPU drone

MCU drone

Digital Electronic Integrated circuit



The Processor Design Space



Microcontrollers

What is a microcontroller ?

A ***Microcontroller*** is a small CPU with many support devices built into the chip

- Self Contained (CPU, Memory, I/O)
- Application or Task Specific (Not a general-purpose computer)
- Appropriately scaled for the job
- Small power consumption
- Low costs ($0.50 to $5.00.)

Market overview

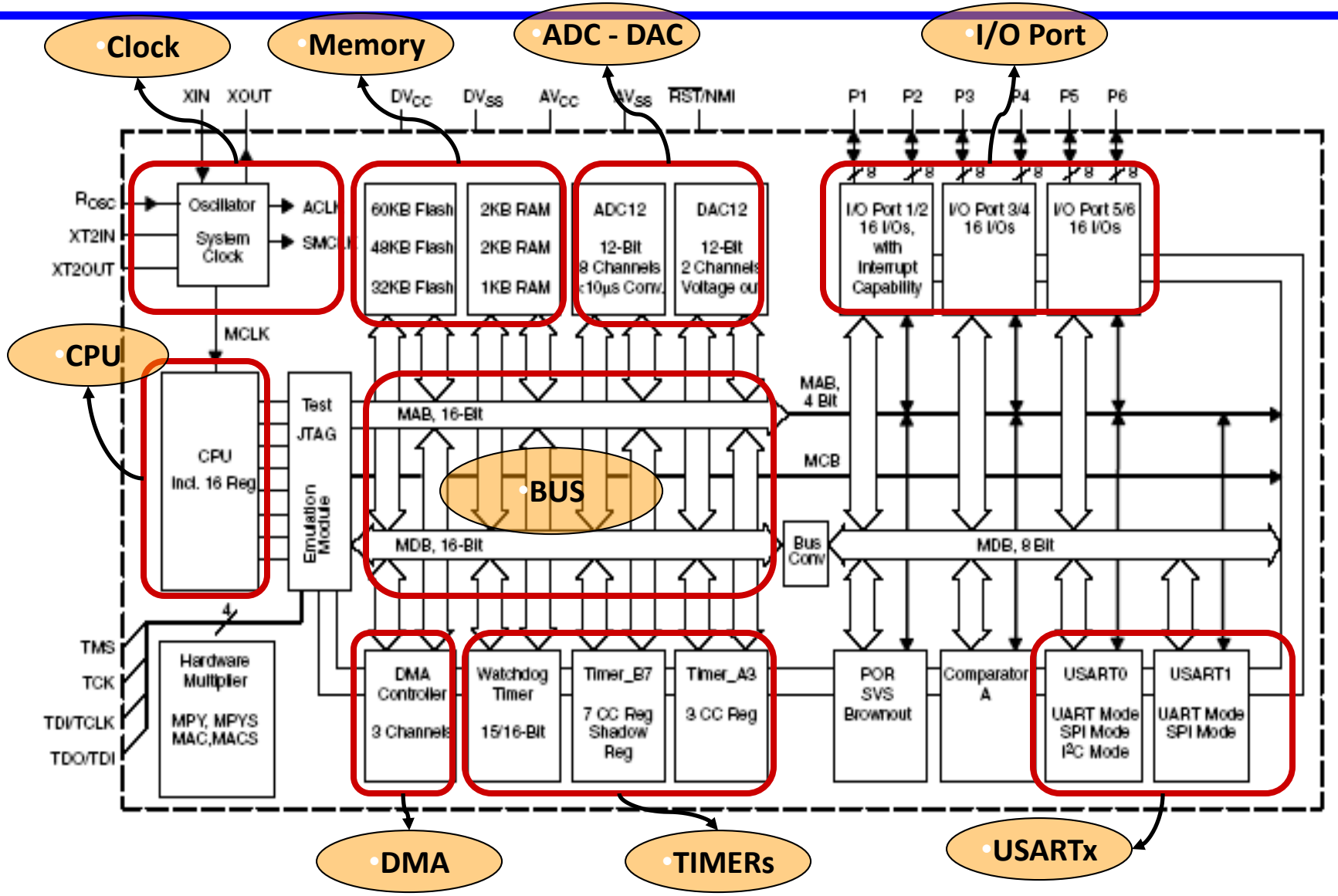
Microchip Technology™ Delivers 10 Billionth PIC® Microcontroller to Samsung Electronics Co.

2011, Sept 15th

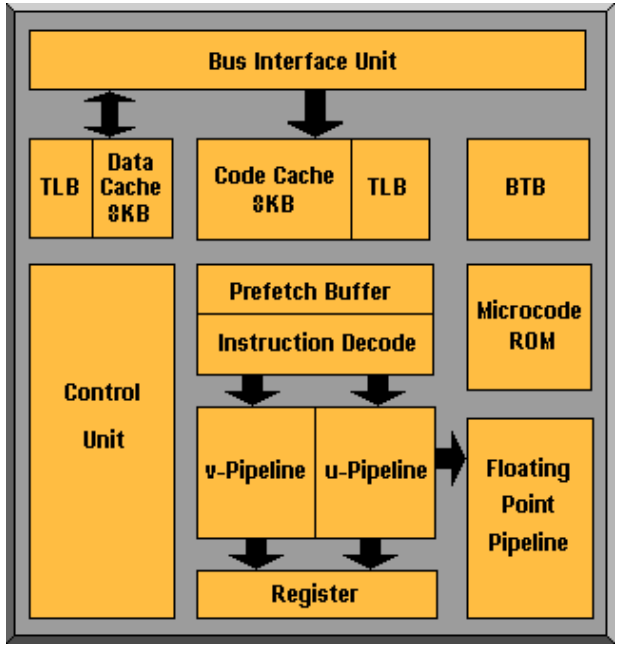
approximately 10 months after delivering its 9 billionth

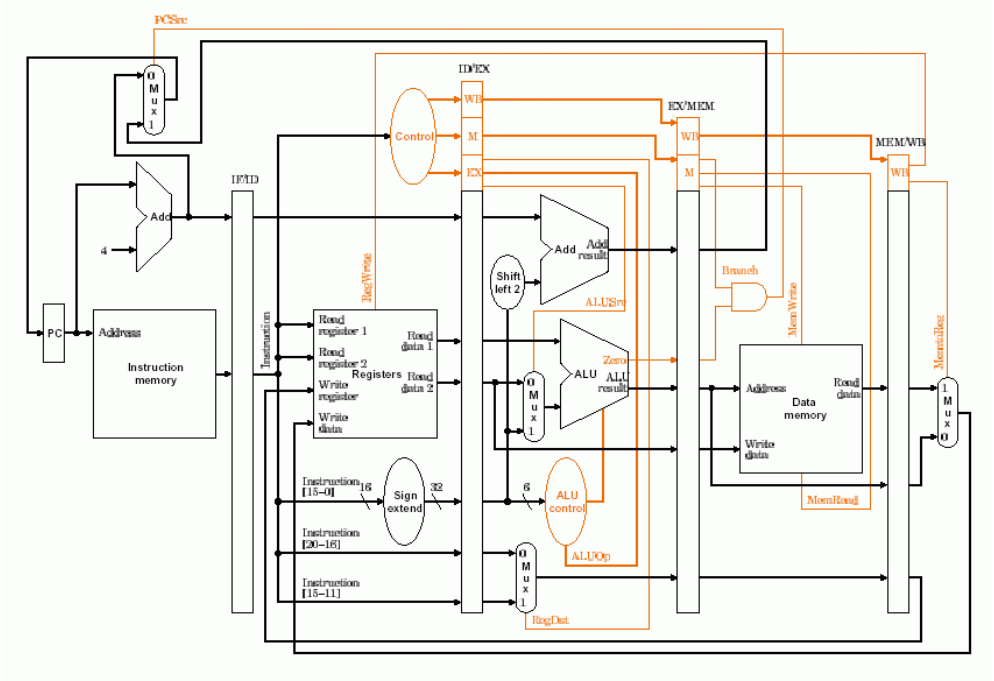
This accounts only the microprocessors of a single manufacturer

Example of MCU Architecture



CPU – Central Processing Unit



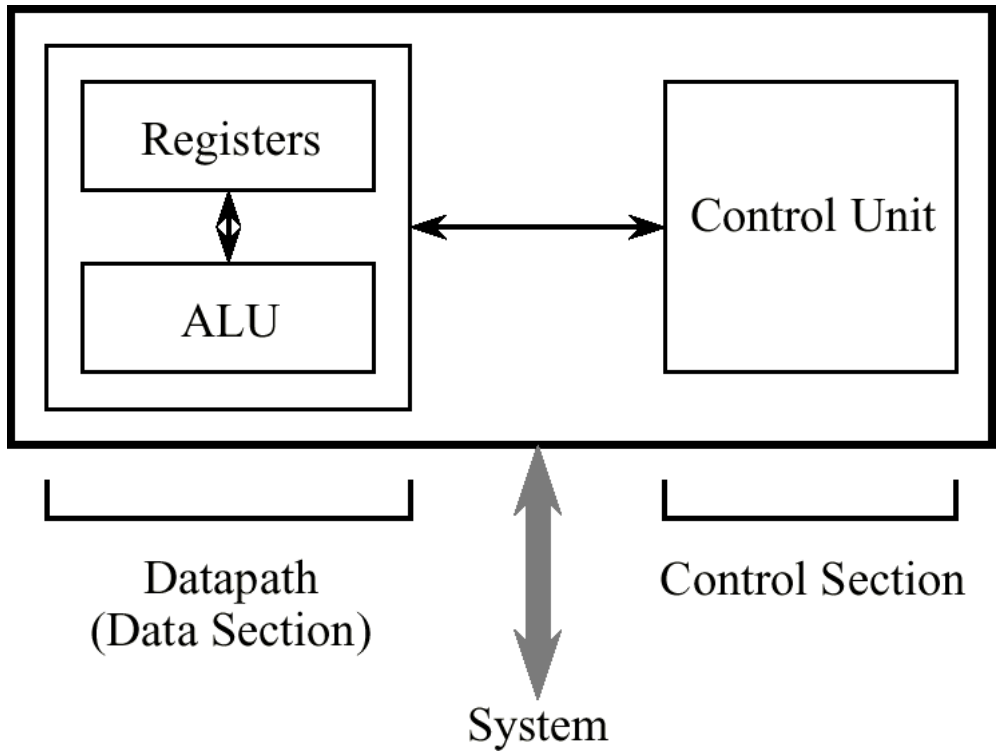


Characteristic

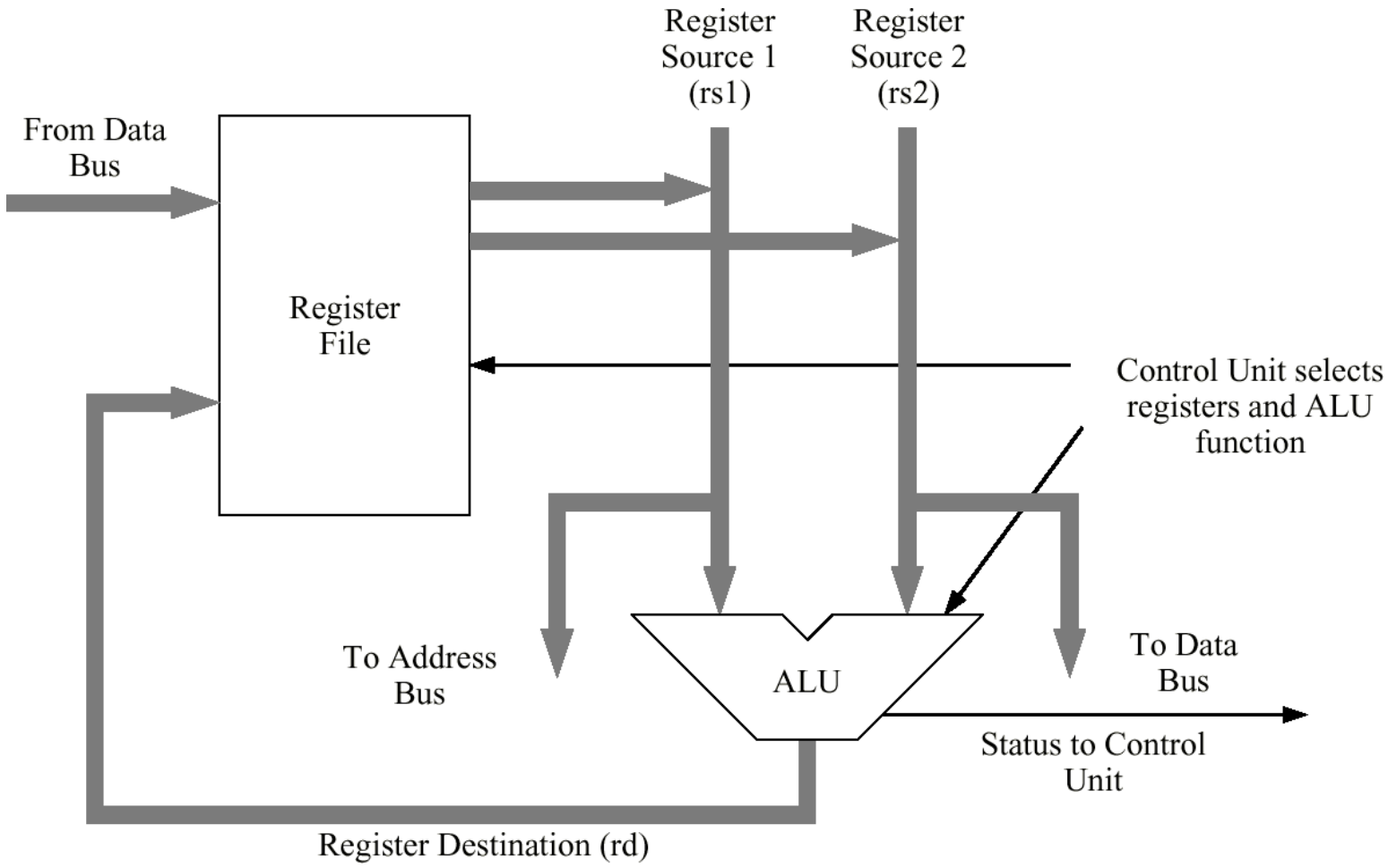
- **Instruction set**
  - CISC Complex Instruction Set Computing (Intel x86 family; Motorola 680x0 Family)
  - RISC Reduced Instruction Set computer (AIM Power PC, ARM family, ATMEL AVR Family)
- **Architecture (respect integer operand maximum dimension)**
  - 8 bit (Intel 8051, Motorola 6800, ATMEL AVR)
  - 16 bit (Intel 8088, Motorola 68000, TI MSP430)
  - 32 bit (x86 family, Motorola 680x0 Family, Power PC, **ARM**)
  - 64 bit (x86-64 family, Power PC)

Abstract View of a CPU

The CPU consists of a data section containing registers and an ALU (Arithmetic and Logic Unit), and a control section, which **interprets instructions** and effects register transfers. The data section is also known as the datapath.



An Example Datapath



•The datapath usually consists of a collection of registers known as the register file and the arithmetic and logic unit (ALU).

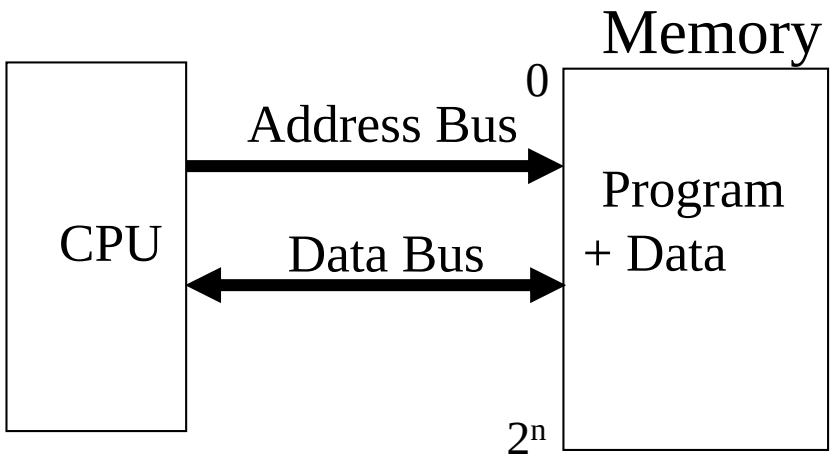
Memory System Architectures

- Two types of information are found in a typical program code:
  - Instruction codes for execution
  - Data that is used by the instruction codes

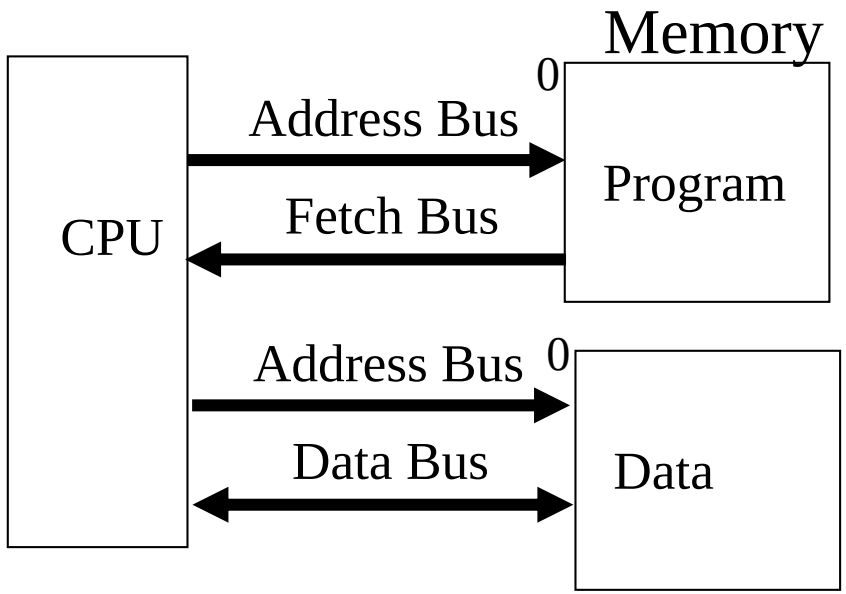
The **instruction cycle** (also known as the **fetch–decode–execute cycle)**

- Two classes of memory systems designed to store the information:
  - von Neumann architecture
  - Harvard architecture

CPU Architectures



Von Neumann Architecture

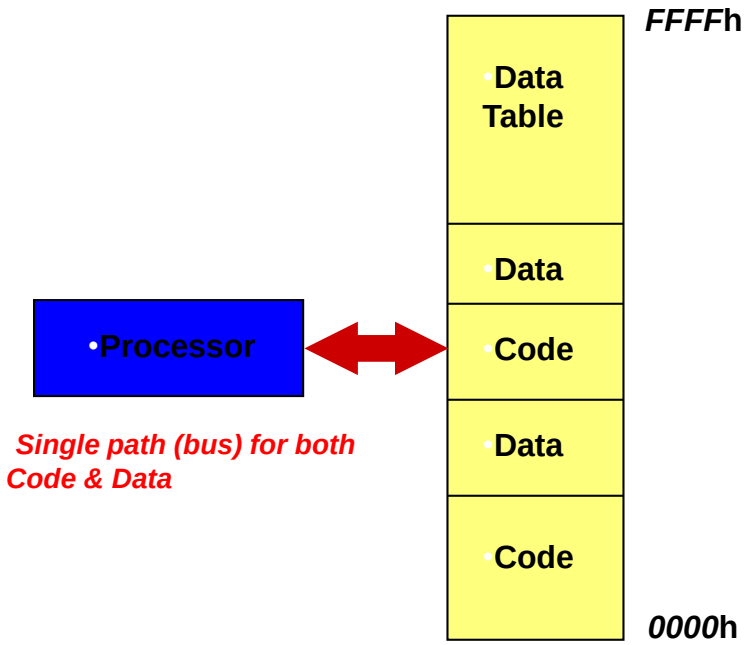


Harvard Architecture

von Neumann Architecture

The von Neumann architecture utilizes only one memory bus for both instruction fetching and data access

- simplifies the hardware and glue logic design
- code and data located in the same address space



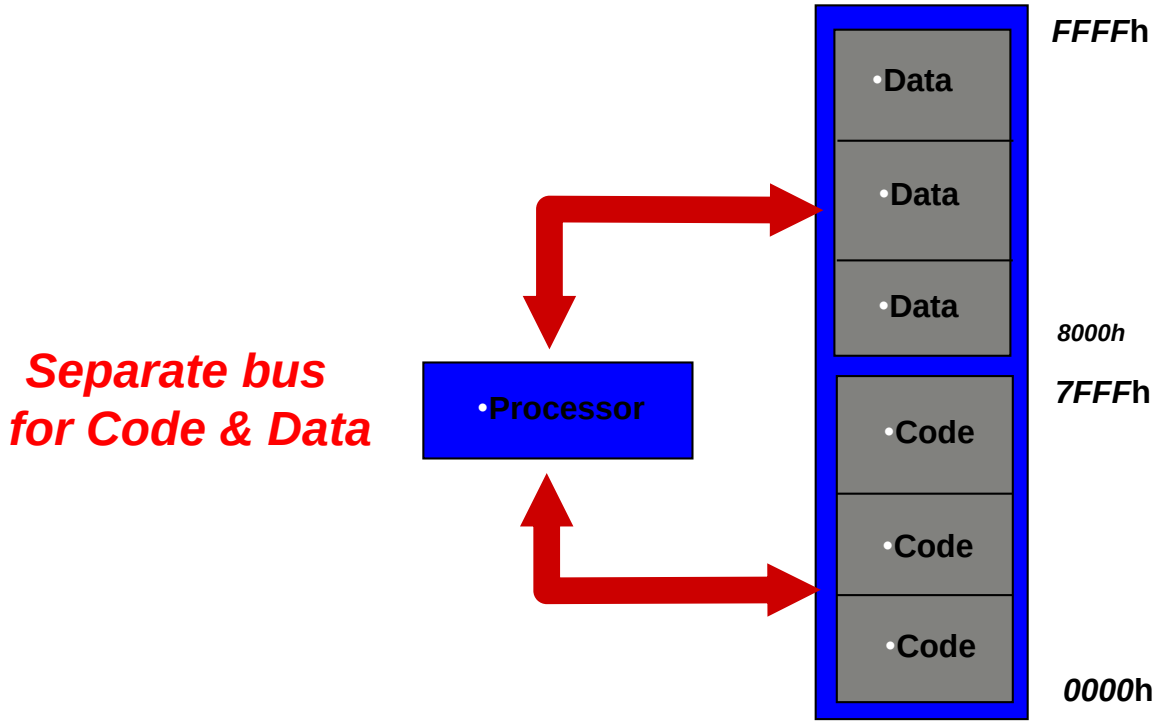
von Neumann Features

- Single memory interface bus
  - simplifies the hardware and glue logic design
- More efficient use of memory
  - code and data can reside in the same physical memory chip
- More flexible programming style
  - e.g., can include self-modified code
- But data may overwrite code (e.g. due to program bug)
  - need memory protection (e.g. hardware-based MPU)
- Bottleneck in code and data transfer
  - only one memory bus for both data and code fetching

Harvard Architecture

The Harvard architecture utilizes separate instruction bus and data bus

- code and data may still share the same memory space

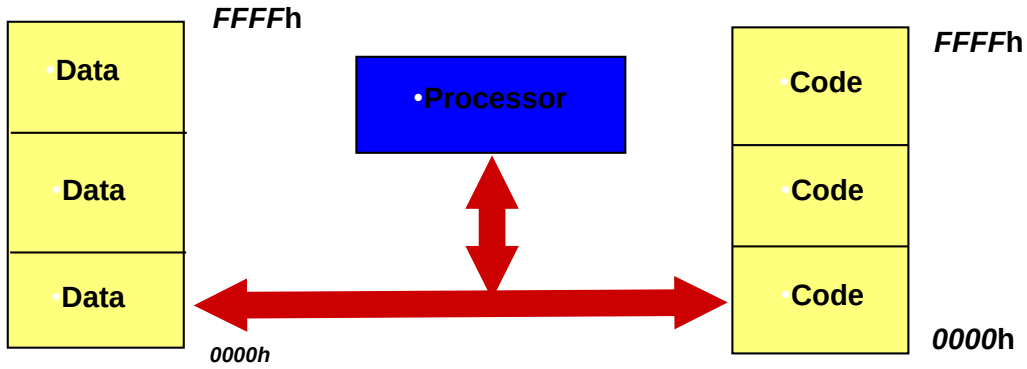


Harvard Features

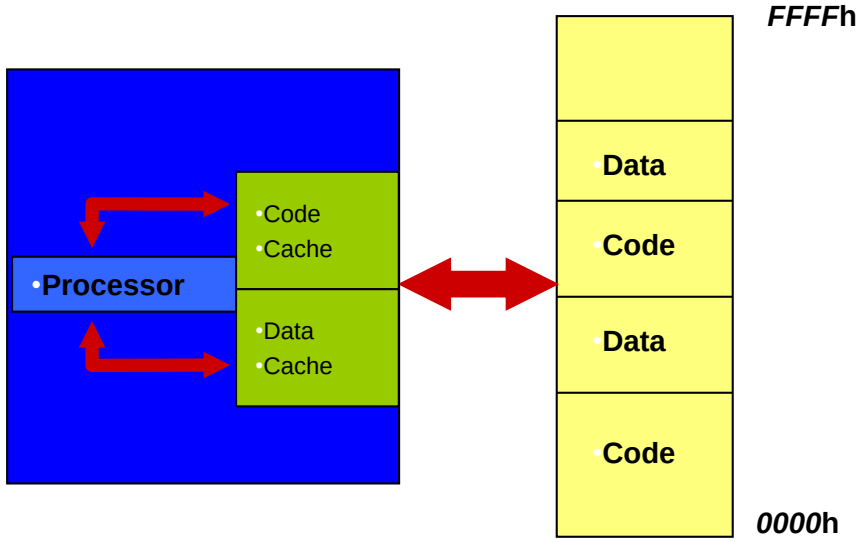
- Separate instruction and data buses
  - allow code and data access at the same time which gives improved performance
  - provide better support for instruction pipeline operations and shorter instruction execution time
  - allow different sizes of data and instructions to be used which results in more flexibility
  - do not incur any code corruption by data which makes the operations more robust
- But more sophisticated hardware glue logic is required to support multiple interface buses

Architecture Variations

Independent data and code memory but with one shared bus (e.g. 8051)



Two separate internal bus for code & data (e.g. ARM9)



von Neumann vs. Harvard

- Harvard allows two simultaneous memory fetches.
- Most DSPs use Harvard architecture for streaming data:
  - greater memory bandwidth;
  - more predictable bandwidth.
- Harvard cannot deal with self-modified code
  - Code that alters its own instructions while it is executing (We don't see it in this course ☺)

von Neumann Architecture an example

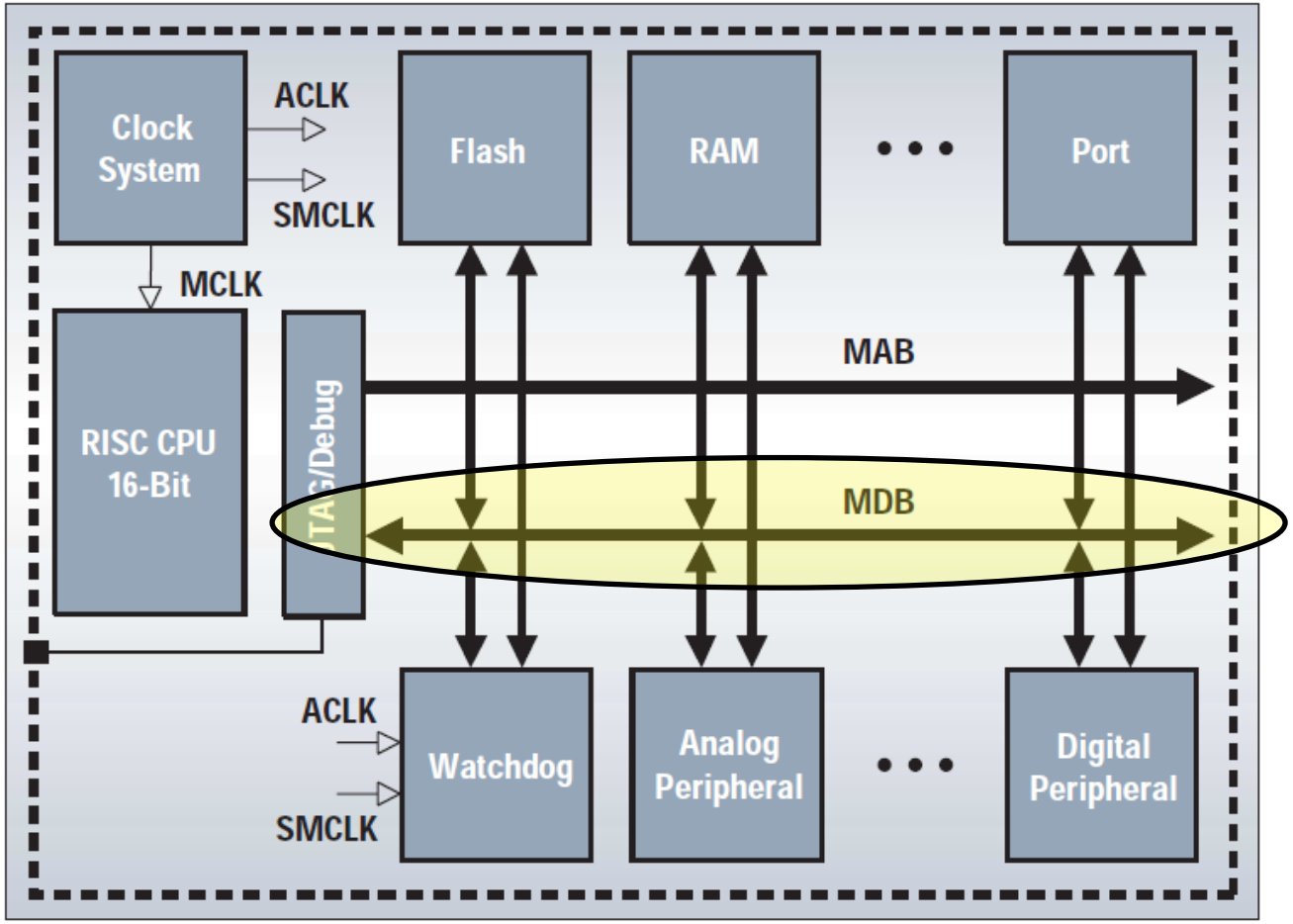
MSP430

Texas Instruments

von-Neumann architecture

All program, data memory and peripherals share a common bus structure.

Consistent CPU instructions and addressing modes are used.

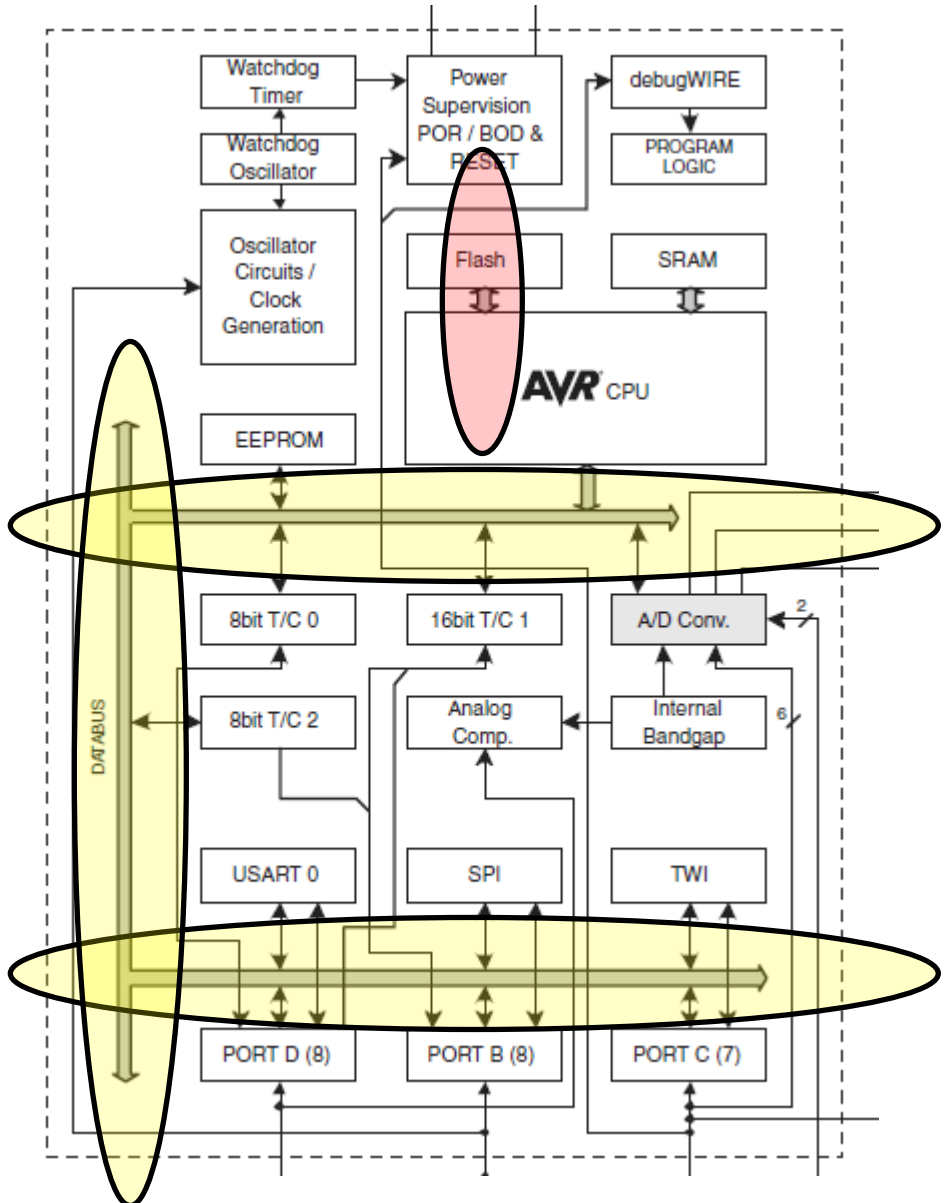


Harvard Architecture an example

- **AVR (atmel)**
- Harvard architecture

- **Separate memories and buses for program and data**.

- Instructions in the program memory are executed with a single level pipelining.



Processor Size

- Processor size is described in terms of ‘bits’ (e.g. an 8-bit or 32-bit processor)
  - corresponds to the data size that can be manipulated at a time by the processor
  - typically reflected in the size of the processor (internal) data path and register bank
- An 8-bit processor can only manipulate one byte of data at a time, while a 32-bit processor can handle one 32-bit double word sized data at a time even though the data content may only be of single byte size

Performance Metrics

- How we compare and classify CPU and microcontrollers?
  - Performance Metrics NOT easy to define and mostly application depended.

Eletrical:

- Power Consumptions
- Voltage Supply
- Noise Immunity
- Sensitivity

Goal: best *tradeoff*
power consumptions ***Vs*** performances

Computation:

- Clock Speed
- **MIPS (instructions per sec)**
- **Latency**
  - Lateness of the response
  - Lag between the begin and the end of the computation
- Throughput
  - Tasks per second
  - Byte per second

Power as a Design Constraint

- Why worry about power?
  - Battery life in portable and mobile platforms
  - Power consumption in desktops, server farms
    - Cooling costs, packaging costs, reliability, timing
    - Power density: 30 W/cm2 in Alpha 21364 (3x of typical hot plate)

Where does power go in CMOS?

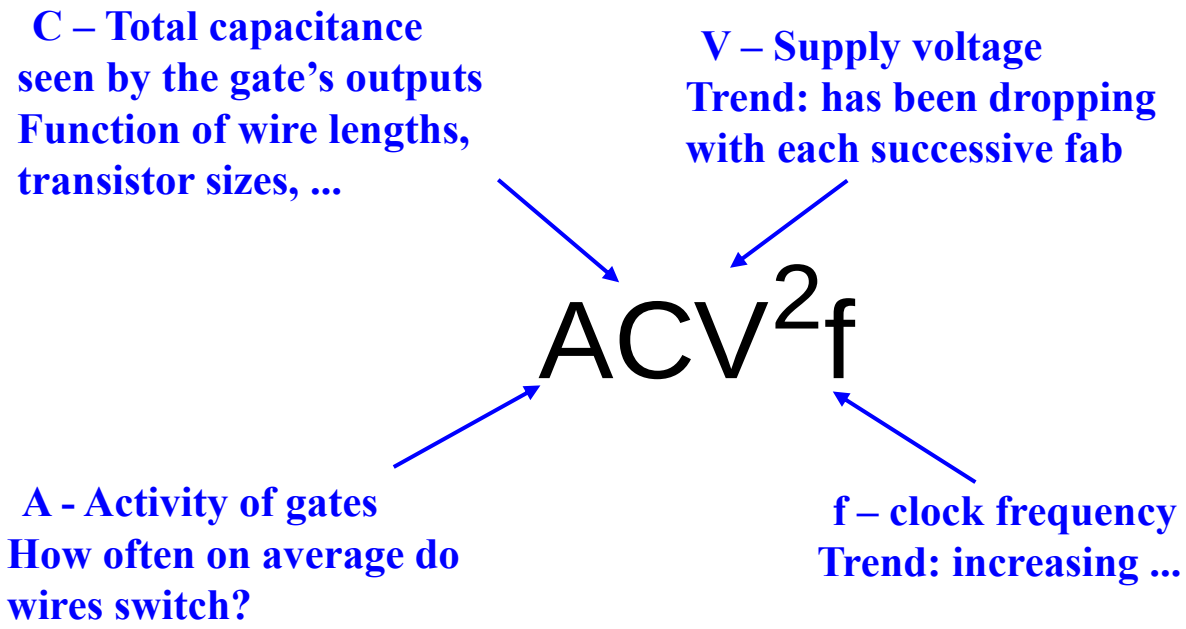
Dynamic power consumption

Power due to short-circuit current during transition

Power due to leakage current

$$P = ACV^2f + \tau AVI_{short}f + VI_{leak}$$

Dynamic Power Consumption



Reducing Dynamic Power

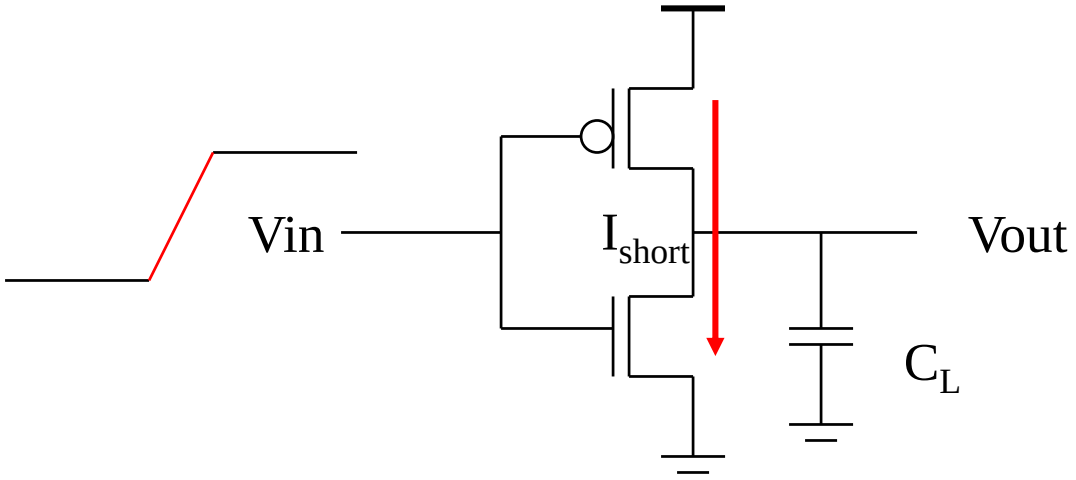
Reducing V has quadratic effect; Limits?

Lower C - shrink structures, shorten wires

Reduce switching activity - Turn off unused parts or use design techniques to minimize number of transitions

Short-circuit Power Consumption

$$\tau A V I_{short} f$$



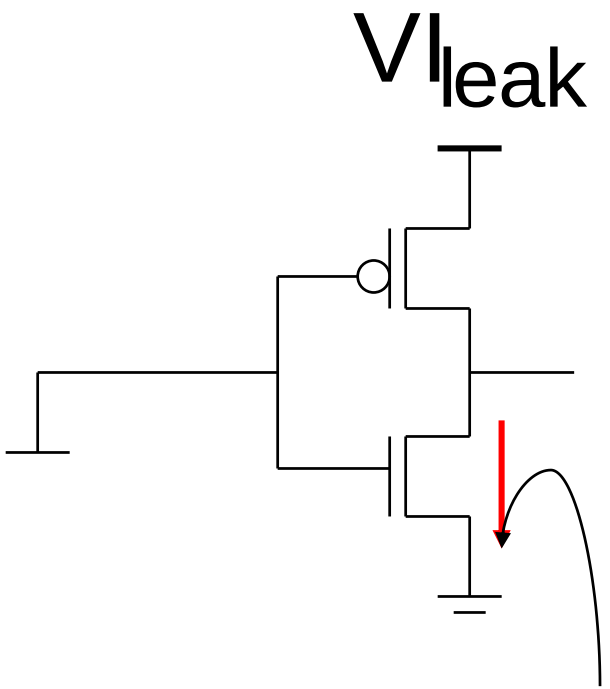
Finite slope of the input signal causes a direct current path between V_{DD} and GND for a short period of time during switching when both the NMOS and PMOS transistors are conducting

Reducing Short-circuit

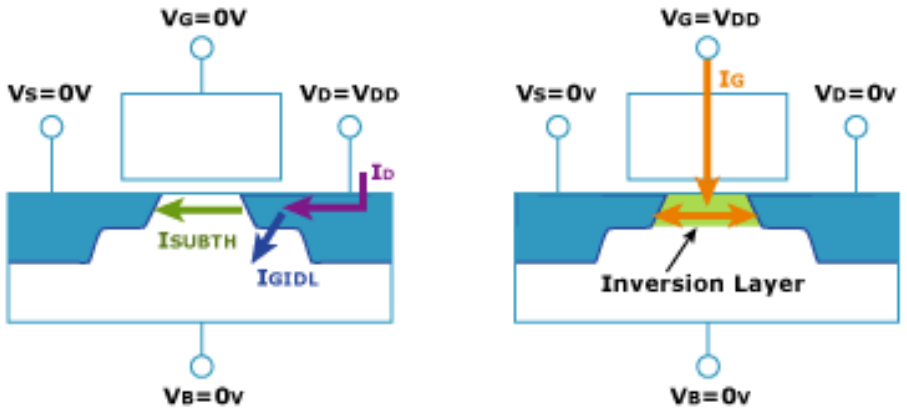
Lower the supply voltage V

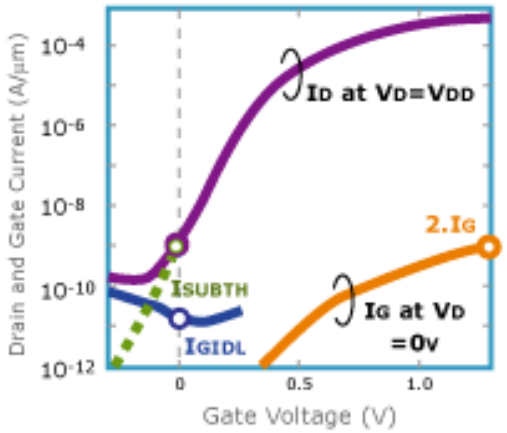
Slope engineering – match the rise/fall time of the input and output signals

Leakage Power



Sub-threshold current

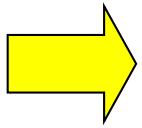




Sub-threshold current grows exponentially with increases in temperature and decreases in Vt

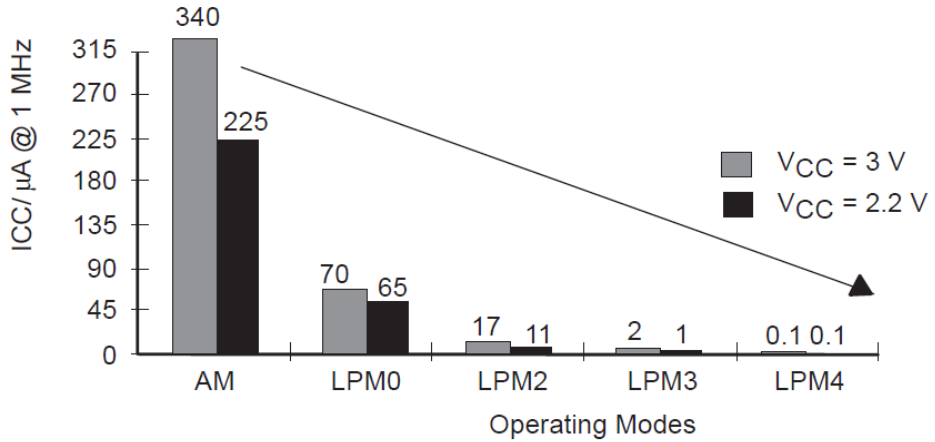
How can we reduce power consumption?

- Dynamic power consumption
  - Reduce the rate of charge/discharge of highly loaded nodes
  - Reduce spurious switching (glitches)
  - Reduce switching in idle states (clock gating)
  - Decrease frequency
  - Decrease voltage (and frequency)
- Static power Consumption
  - Smaller area (!)
  - Reduce device leakage through power gating
  - Reduce device leakage through body biasing
  - Use higher-threshold transistors when possible



Power performance tradeoffs!

Operating Modes



- **Assembler Code Example:**

```
bis.w    #CPUOFF,SR      ; LPM0
```

- **C Code Example:**

```
_BIS_SR (CPUOFF);        // LPM0
```

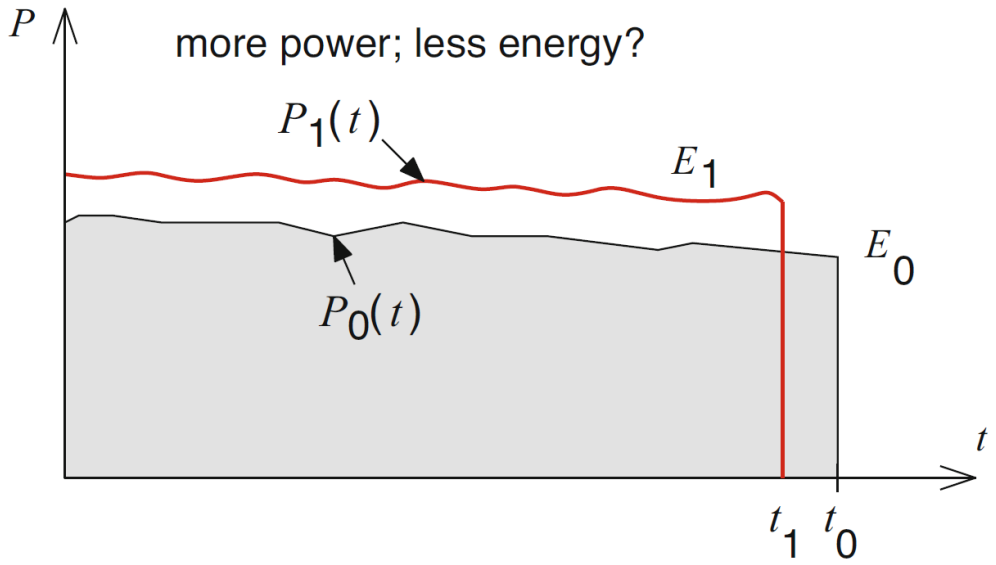
SCG1	SCG0	OSCOFF	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, all enabled clocks are active
0	0	0	1	LPM0	CPU, MCLK are disabled SMCLK , ACLK are active
0	1	0	1	LPM1	CPU, MCLK, DCO osc. are disabled DC generator is disabled if the DCO is not used for MCLK or SMCLK in active mode SMCLK , ACLK are active
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator remains enabled ACLK is active
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO osc. are disabled DC generator disabled ACLK is active
1	1	1	1	LPM4	CPU and all clocks disabled

Why *Ultra-low Power* Is so Important

- Longer battery life
- Smaller products
- Simpler power supplies
- Less EMI simplifies PCB
- *Permanent* battery
- Reduced liability

What about the energy?

- Energy is $E = \int P dt$
  - Where P is the power consumption in a dt Time.
- Example we have 2 tasks with different power P_o and time t_o
  - Energy for a specific task 0 is $\mathrm{E_o} = \int_0^{t_0} P_o\ (t)\ dt$
  - Energy for a specific task 1 is $\mathrm{E_1} = \int_0^{t_1} P_1\ (t)\ dt$



Parallel Architectures

- Assuming we have a CPU working with a task that takes t time to execute at the frequency f

- Moving toward executing β operations in parallel
  - Extend the time for each operation by a factor of β
  - We can reduce the original frequency of fact β $\Longrightarrow f' = \frac{f}{\beta} \Longrightarrow$ **Lower power and eventually lower voltage**
    - to execute at the same time
  - Or we can run the task with the some frequency but for less time $t' = \frac{t}{\beta} \Longrightarrow E' = \frac{E}{\beta}$ **Saving Energy**

Table 2 Basic parameters of CPUs and GPUs

Processor Type	**CPU**		**GPU**	
Version	Intel-E5-2650 v1	Intel-E5-2695 v3	Nvidia Tesla K-40	Nvidia Tesla V-100
Core number	8 cores, 16 threads	18 cores, 36 threads	2880	5120*
Clock rate	2.0GHz	2.3GHz	0.875GHz	1.5GHz
Memory	32GB	32GB	12GB	16GB*
Compute ability	---	---	3.5	7.0
Year of production	2011	2016	2013	2017

Liu, J., Hu, F. Q., & Li, X. (2018). Performance Comparison on Parallel CPU and GPU Algorithms for Unified Gas-Kinetic Scheme. *arXiv preprint arXiv:1810.08137*.

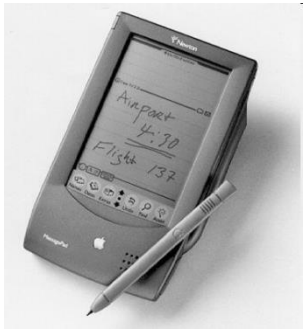
ARM- Processors

The ARM Processor Architecture

- ARM stands for “Advanced RISC Machine”
  - based on Reduced Instruction Set Computer (RISC) architecture
    - trading simpler hardware circuitry with software complexity (and size)
    - but latest ARM processors utilize more than 100 instructions

A Bit of ARM History

- Originally conceived to be a processor for the desktop system (Acorn®)
  - now entrenched in embedded markets
- First well-known product:
  - Apple®'s Newton™ PDA (1993)
    based on an ARM6™ core
- Significant breakthrough:
  - Apple®'s iPod® (2001)
    based on an ARM7™ core



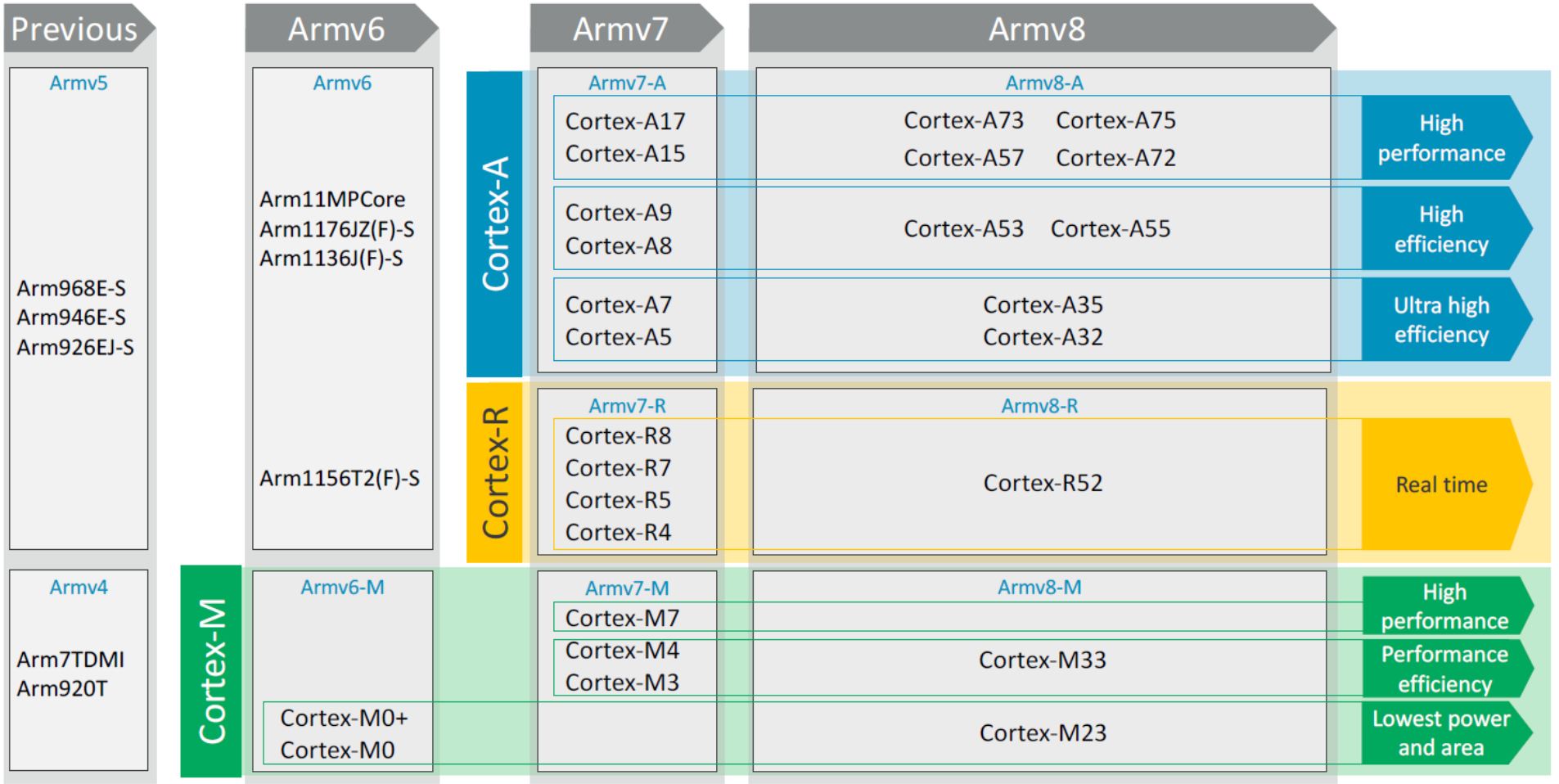
The Microprocessor Market

- In 2007,
  - 13 billion microprocessors were shipped
  - 3 billion were embedded processors based on the ARM architecture
  - 150 million were for the PC, notebook, and workstation

- By February 2008,
  - 10 billion ARM-based processors have been produced

- Only in 2017
  - 20 Billions ARM-Based processors have been produced!

ARM Processors Families

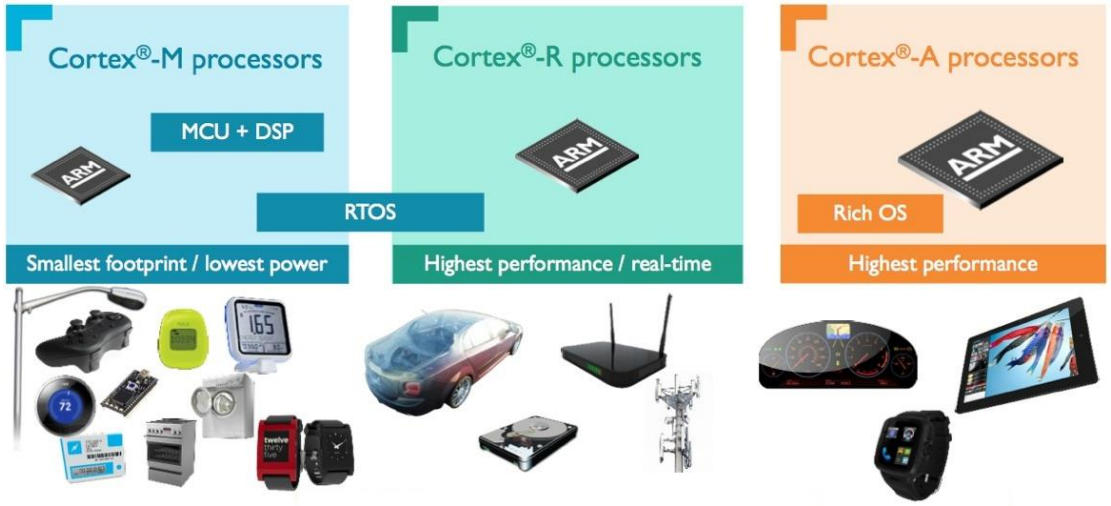


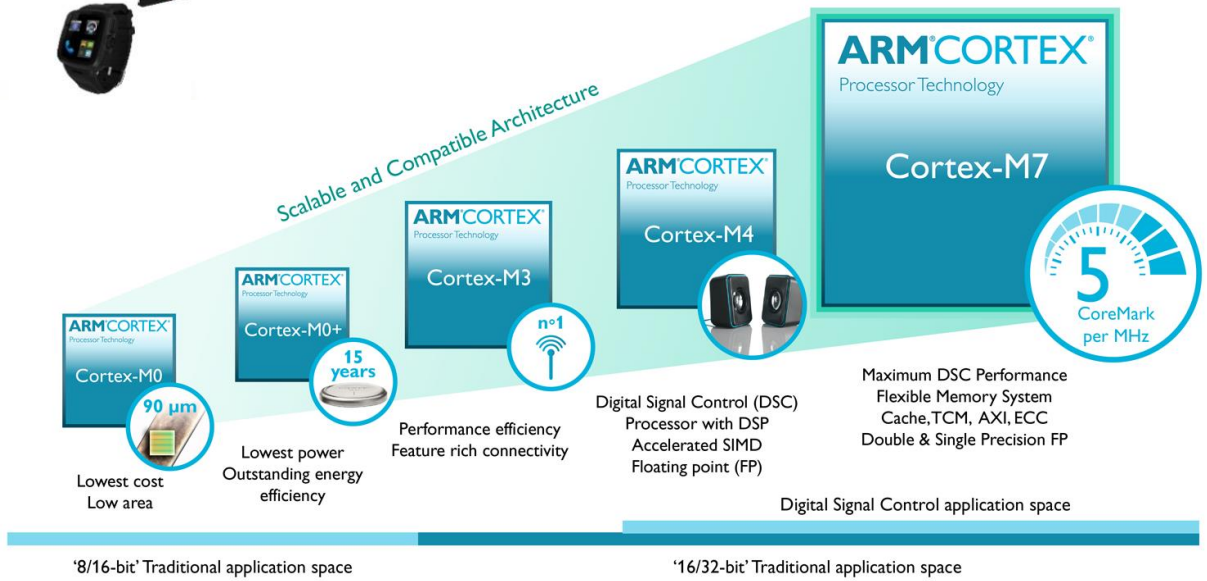
arm

ARM Processors Architectures (2)

- Key attributes: Implementation size, performance, and very low power.
- Architectures types:
  - **ARMv4T** architecture introduced the 16-bit Thumb® instruction set alongside the 32-bit ARM instruction set.
  - **ARMv5TEJ** architecture introduced arithmetic support for digital signal processing (DSP) algorithms.
  - **ARMv6** architecture introduced an array of new features including the **Single Instruction Multiple Data (SIMD) operations.**
  - **ARMv7** architecture implementsThumb-2 technology.
    - **Cortex-A** implements a virtual memory system architecture based on an MMU, an optional NEON processing unit for multimedia applications and advanced hardware Floating Point.
    - **Cortex-R** – implements a protected memory system architecture based on an MPU (memory protection unit).
    - **Cortex-M** – Microcontroller profile designed for fast interrupt processing.
  - **ARMv8** implementing 64bit instruction set

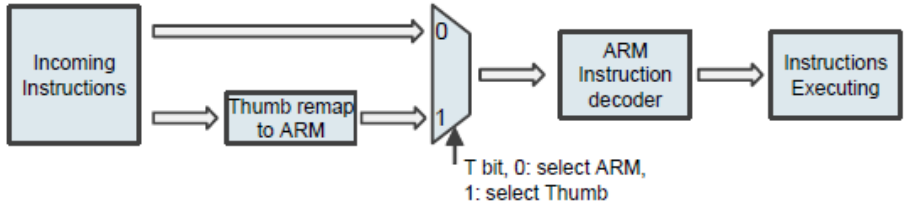
ARM Processors Families

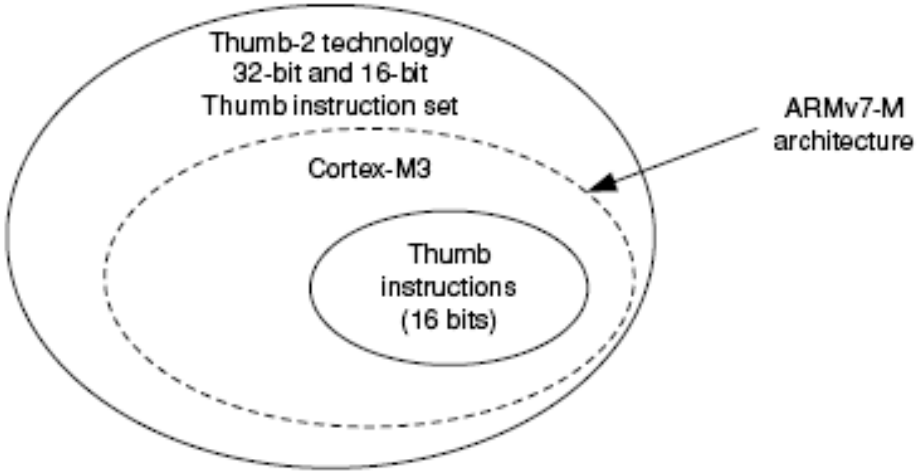




Thumb-2 Instrution Set

- Mixes 16 and 32 bits instructions
  - Enhancements: eg division, bit-field operators, wrt traditional ARMv4T
  - No need to mode switch, can be mixed freely
- **Not** backwards binary compatible
  - But porting is «easy»





Cortex-M4 Processor Overview

- Cortex-M4 Processor
  - Introduced in 2010
  - Designed with a large variety of highly efficient signal processing features
  - Features extended single-cycle multiply accumulate instructions, optimized SIMD arithmetic, saturating arithmetic and an optional Floating Point Unit (**When is an Cortex-M4F!).**
- High Performance Efficiency
  - 1.25 DMIPS/MHz (Dhrystone Million Instructions Per Second / MHz) at the order of µWatts / MHz
- Low Power Consumption
  - Longer battery life – especially critical in mobile products
- Enhanced Determinism
  - The critical tasks and interrupt routines can be served quickly in a known number of cycles

Cortex-M4 Processor Features

- 32-bit Reduced **Instruction Set Computing (RISC) processor**
- Harvard architecture
  - Separated data bus and instruction bus
- Instruction set
  - Include the entire Thumb®-1 (16-bit) and Thumb®-2 (16/ 32-bit) instruction sets
- 3-stage + branch speculation pipeline
- Performance efficiency
  - 1.25 – 1.95 DMIPS/MHz (Dhrystone Million Instructions Per Second / MHz)
- Supported Interrupts
  - Non-maskable Interrupt (NMI) + 1 to 240 physical interrupts
  - 8 to 256 interrupt priority levels

Cortex-M4 Processor Features

- Supports Sleep Modes
  - Up to 240 Wake-up Interrupts
  - Integrated WFI (Wait For Interrupt) and WFE (Wait For Event) Instructions and Sleep On Exit capability (to be covered in more detail later)
  - Sleep & Deep Sleep Signals
  - Optional Retention Mode with ARM Power Management Kit
- **Enhanced Instructions**
  - Hardware Divide (2-12 Cycles)
  - **Single-Cycle 16, 32-bit MAC (Multiply and Accumulator), Single-cycle dual 16-bit MAC**
  - **8, 16-bit SIMD arithmetic**
- Debug
  - Optional JTAG & Serial-Wire Debug (SWD) Ports
  - Up to 8 Breakpoints and 4 Watchpoints
- Memory Protection Unit (MPU)
  - Optional 8 region MPU with sub regions and background region

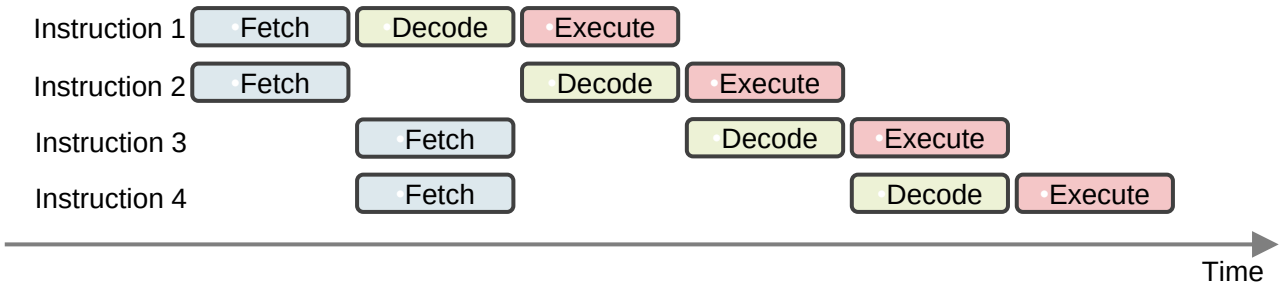
Cortex-M4 Processor Features

- Cortex-M4 processor is designed to meet the challenges of low dynamic power constraints while retaining light footprints
  - 180 nm ultra low power process –157 µW/MHz
  - 90 nm low power process – 33 µW/MHz
  - 40 nm G process – 8 µW/MHz

ARM Cortex-M4 Implementation Data			
Process	180ULL (7-track, typical 1.8v, 25C)	90LP (7-track, typical 1.2v, 25C)	40G 9-track, typical 0.9v, 25C)
Dynamic Power	**157 µW/MHz**	**33 µW/MHz**	**8 µW/MHz**
Floorplanned Area	0.56 mm^2	0.17 mm^2	0.04 mm^2

Cortex-M4 Block Diagram

- Processor core
  - Contains internal registers, the ALU, data path, and some control logic
  - Registers include sixteen 32-bit registers for both general and special usage
- Processor pipeline stages
  - Three-stage pipeline: fetch, decode, and execution
  - **Some instructions may take multiple cycles to execute**, in which case the pipeline will be stalled
  - The pipeline will be flushed if a branch instruction is executed
  - **Up to two instructions can be fetched in one transfer (16-bit instructions)**

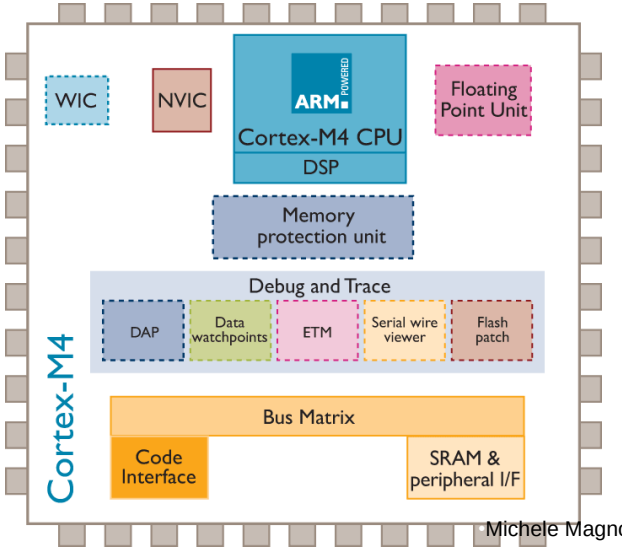


Embedded ARM Cortex Processors (M4)

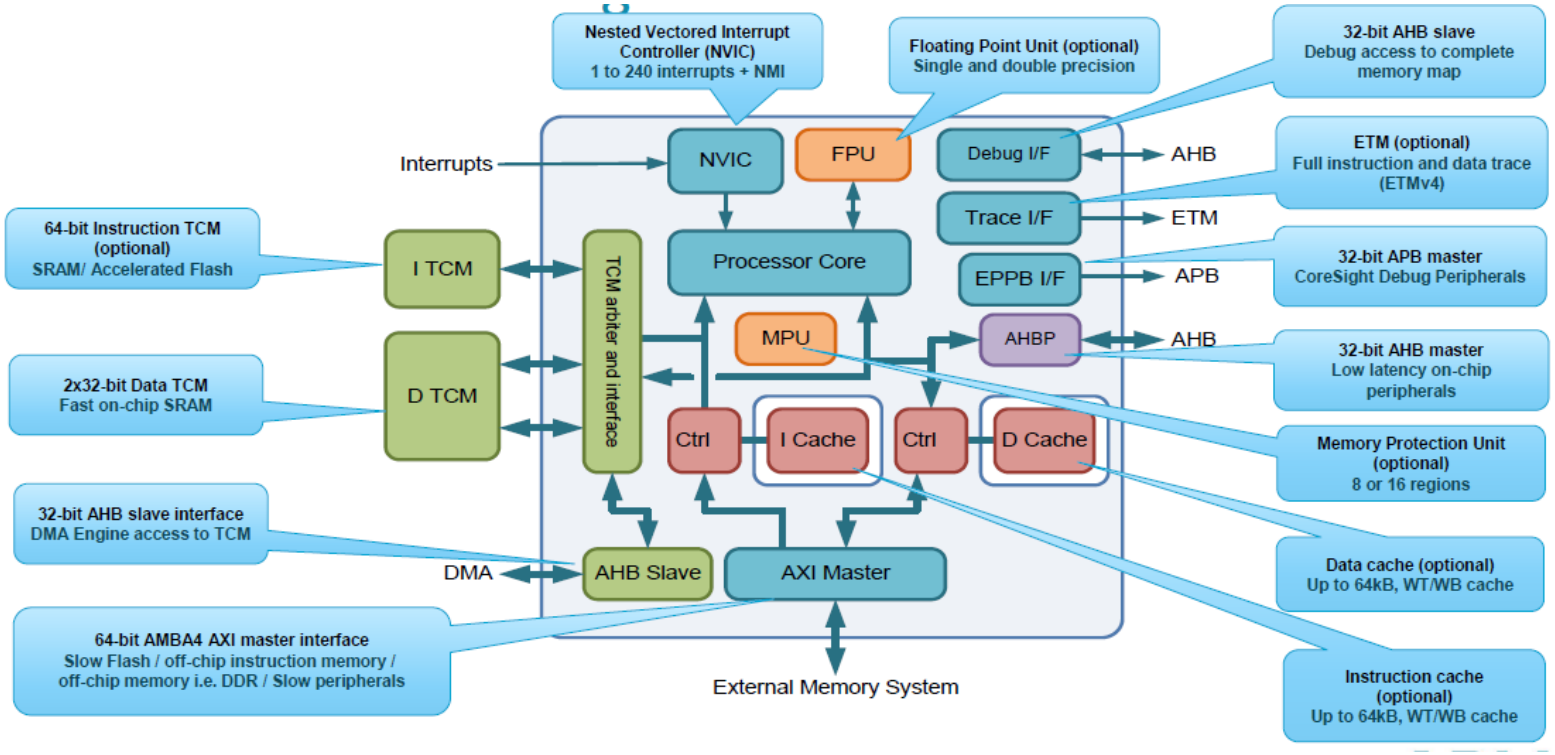
Cortex M4

Embedded processor for DSP with FPU (M4F)

ISA Support	Thumb® / Thumb-2
DSP Extensions	Single cycle 16/32-bit MAC Single cycle dual 16-bit MAC 8/16-bit SIMD arithmetic Hardware Divide (2-12 Cycles)
Floating Point Unit	Single precision floating point unit IEEE 754 compliant
Pipeline	3-stage + branch speculation
Performance Efficiency	3.40 CoreMark/MHz*
Performance Efficiency	Without FPU: 1.25 / 1.52 / 1.91 DMIPS/MHz** With FPU: 1.27 / 1.55 / 1.95 DMIPS/MHz**
Memory Protection	Optional 8 region MPU with sub regions and background region
Interrupts	Non-maskable Interrupt (NMI) + 1 to 240 physical interrupts
Interrupt Priority Levels	8 to 256 priority levels
Wake-up Interrupt Controller	Up to 240 Wake-up Interrupts
Sleep Modes	Integrated WFI and WFE Instructions and Sleep On Exit capability. Sleep & Deep Sleep Signals. Optional Retention Mode with ARM Power Management Kit
Bit Manipulation	Integrated Instructions & Bit Banding
Debug	Optional JTAG & Serial-Wire Debug Ports. Up to 8 Breakpoints and 4 Watchpoints.



Cortex M7

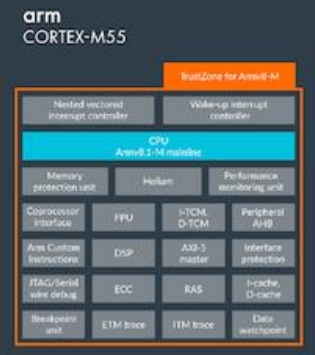


2x Perf of M4

Future Processors: What are you expecting?

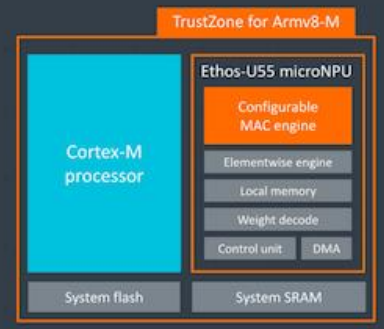
Cortex-M55: The Most AI-capable Cortex-M Processor

- ✓ First CPU based on Arm Helium technology
  - Energy-efficient and configurable with vector processing capabilities
  - Delivers up to 5x DSP performance and up to 15x ML performance*
  - Versatile capability for both classical ML and NN inference
- ✓ Advanced memory interfaces for fast access to ML data and weights
- ✓ Arm TrustZone security, accelerating the route to PSA Certified

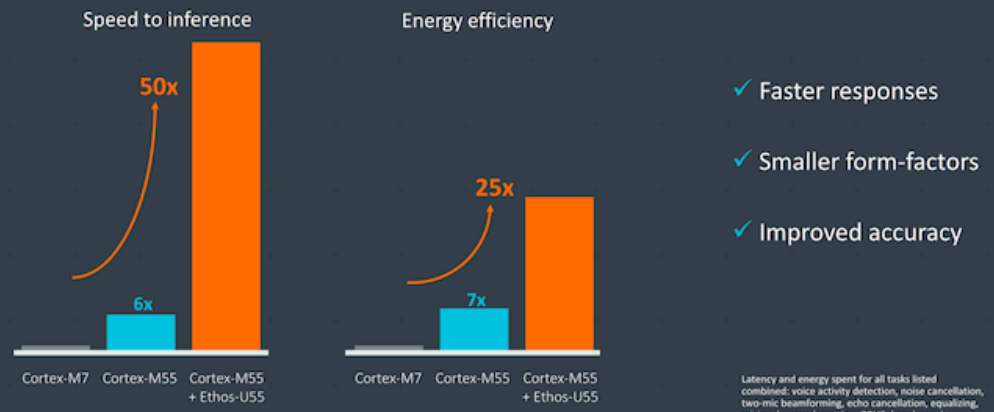


Ethos-U55: The First microNPU for Cortex-M

- ✓ Highest efficiency and small memory footprint
- ✓ 32, 64, 128, or 256 unit multiply-accumulate (MAC) engine
- ✓ Weight decoder and DMA for on-the-fly weight decompression
- ✓ Tooling available for offline optimization
- ✓ Works with a range of Cortex-M processors:
  - Cortex-M55
  - Cortex-M7
  - Cortex-M33
  - Cortex-M4

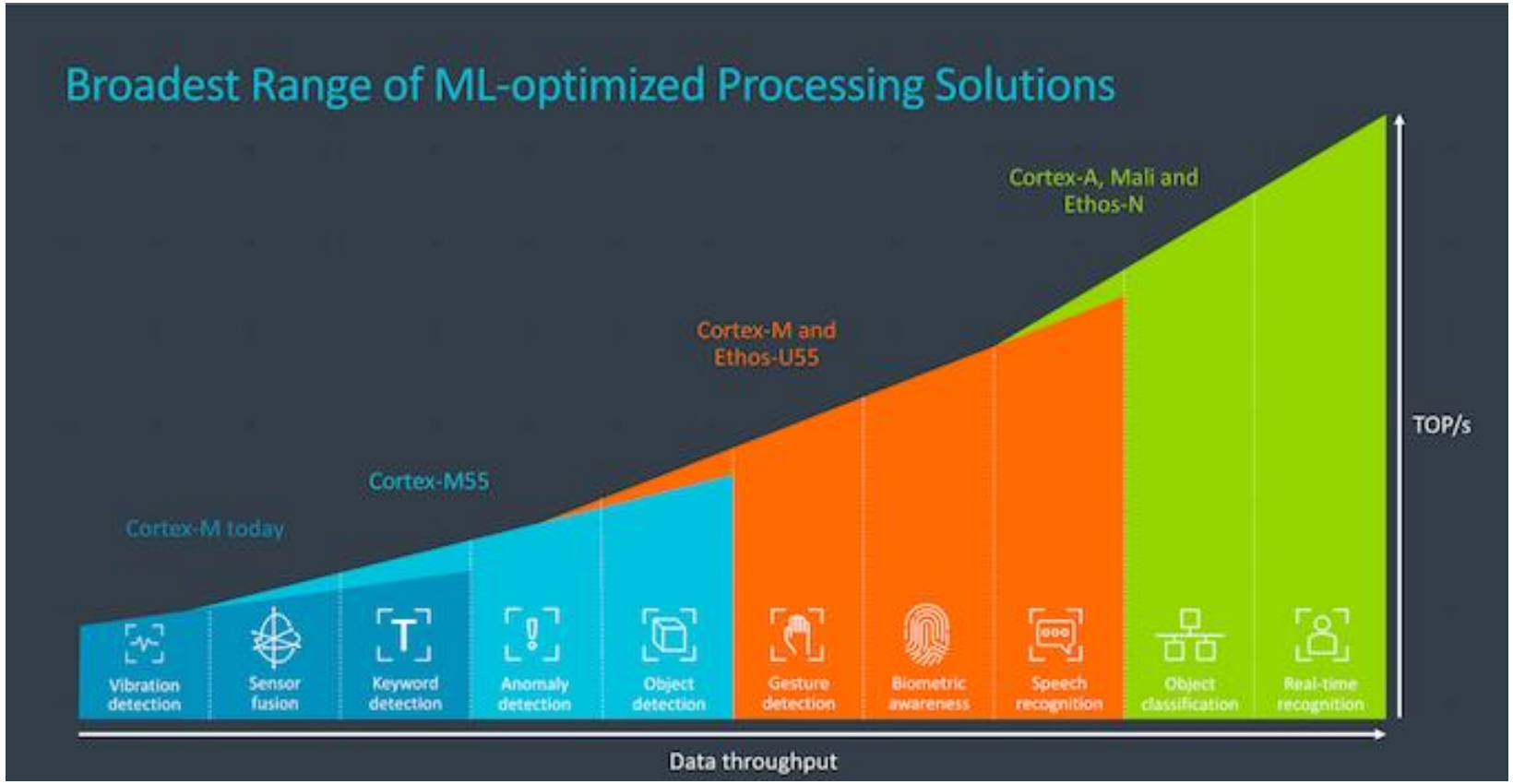


Example: Typical ML Workload for a Voice Assistant



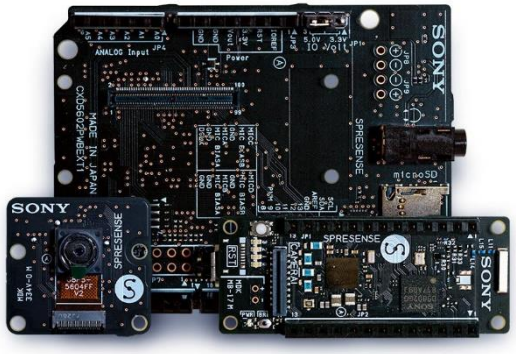
- ✓ Faster responses
- ✓ Smaller form-factors
- ✓ Improved accuracy

Where they will be applied?



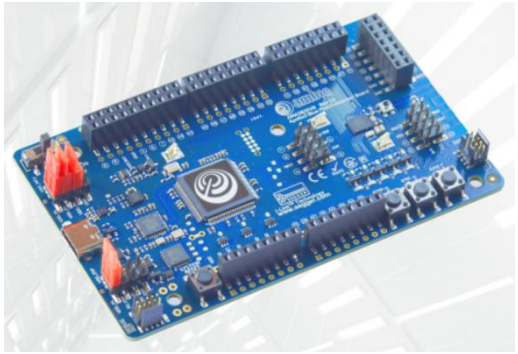
Memory and Parallel are also important

Spresense based on 6 cores Cortex-M4F



Spresense main board	
Model name	CXD5602PWBMAIN1
Size	50.0 mm x 20.6 mm
CPU	ARM® Cortex®-M4F x 6 cores
Maximum clock frequency	156 MHz
SRAM	**1.5 MB**
Flash memory	8 MB
Digital input / output	GPIO, SPI, I2C, UART, I2S
Analog input	2 ch (0.7 V range)
GNSS	GPS(L1-C/A), QZSS(L1-C/A), GLONASS(L1), WAAS, QZSS(L1-S)
Camera input	Dedicated parallel interface

Ambiq Apollo 4 Cortex-M4F



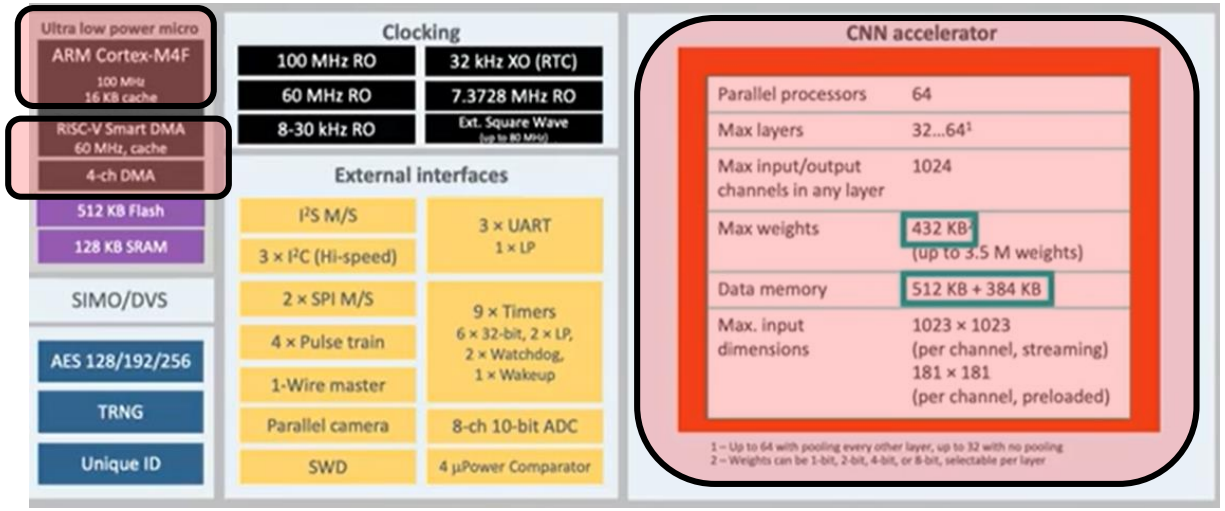
Apollo 4	
Model name	Ambiq Apollo 4(blue)
CPU	ARM® Cortex®-M4F x 1 cores
Maximum clock frequency	192 MHz
SRAM	**3.5 MB**
Flash memory	Info ND
Digital input / output	GPIO, SPI, I2C, UART, I2S
Analog input	Several option
Current Consumption	3 µA/MHz
Wireless Interface	BTLE

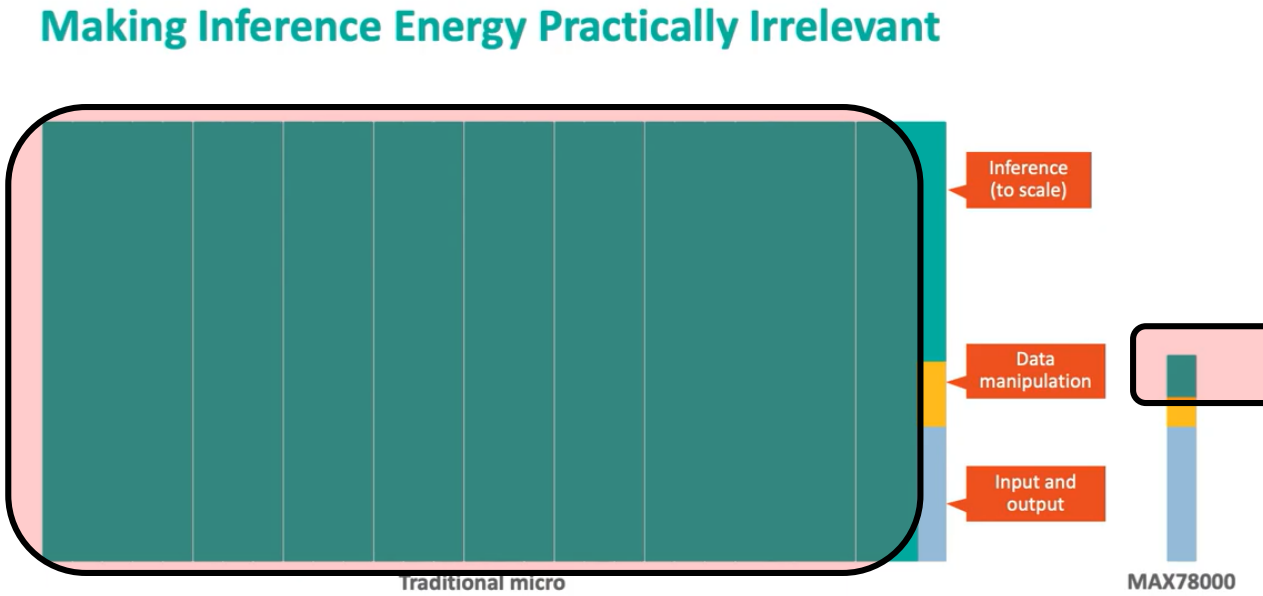
GreenWave GAP8/9 RISC-V5 Based PULP

	GAP8	GAP9
Processing power	22.65 GOPS	150.8 GOPS
Power efficiency	4.24 mW/GOP	0.33 mW/GOP
Memory		
L1	80 kB	128 kB
RAM	512 kB	**1.5 MB**
Non Volatile	None	**2 MB**
External	QSPI/ HyperBus	2x QSPI/OCTO-SPI/HyperBus/SDIO
MAX Frequency		
FC*	250	400
Cluster**	175	400
Fixed Point	8, 16, 32-bit	8, 16, 32, 64-bit***
Floating Point	None	16/16alt/32, 64-bit***
Sound Interface	2 Rx-Only I2S interfaces	3 master/slave SAI full duplex, I2S and TDM 4/8/16 ch capable
Camera Interface	8-bit CPI (Camera Parallel Interface)	8-bit CPI, 2-lane CSI-2

New generation of Embedded Processing (MAX78000 example)

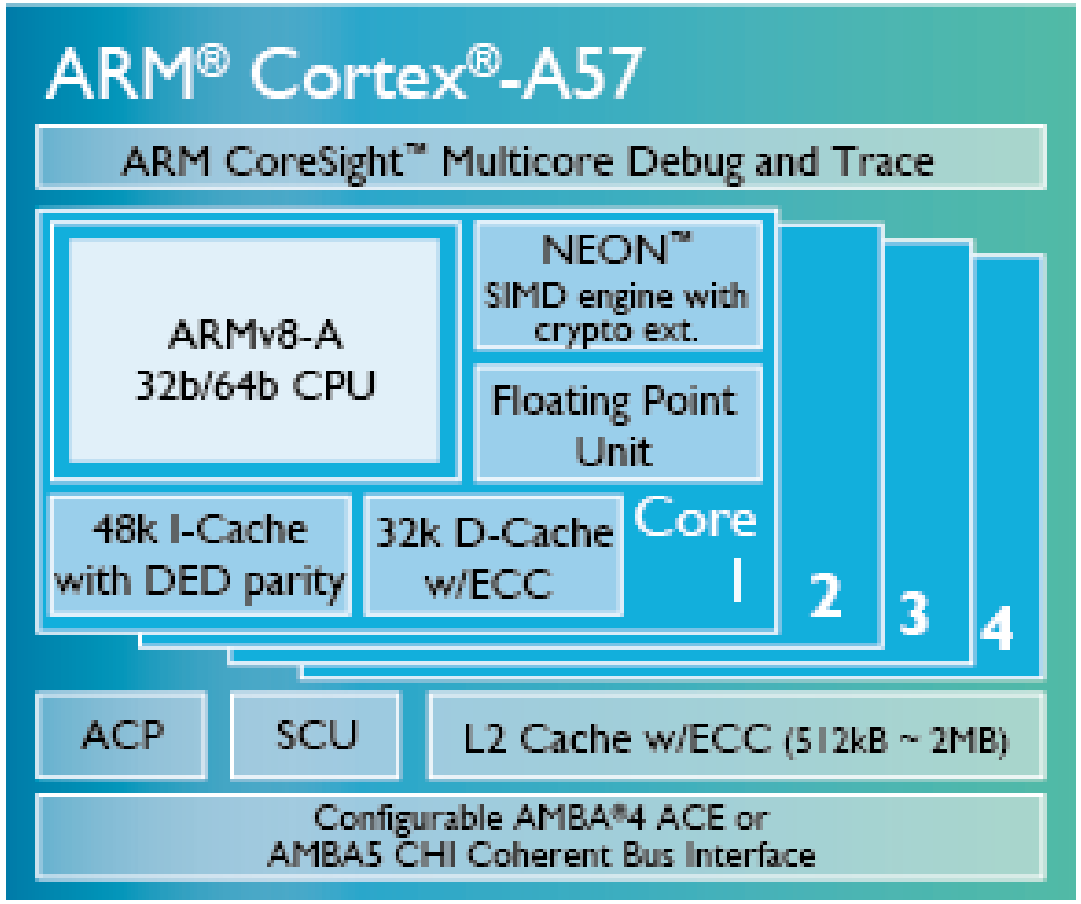
- Multi cores and Convolutional Neural Networks accelerator



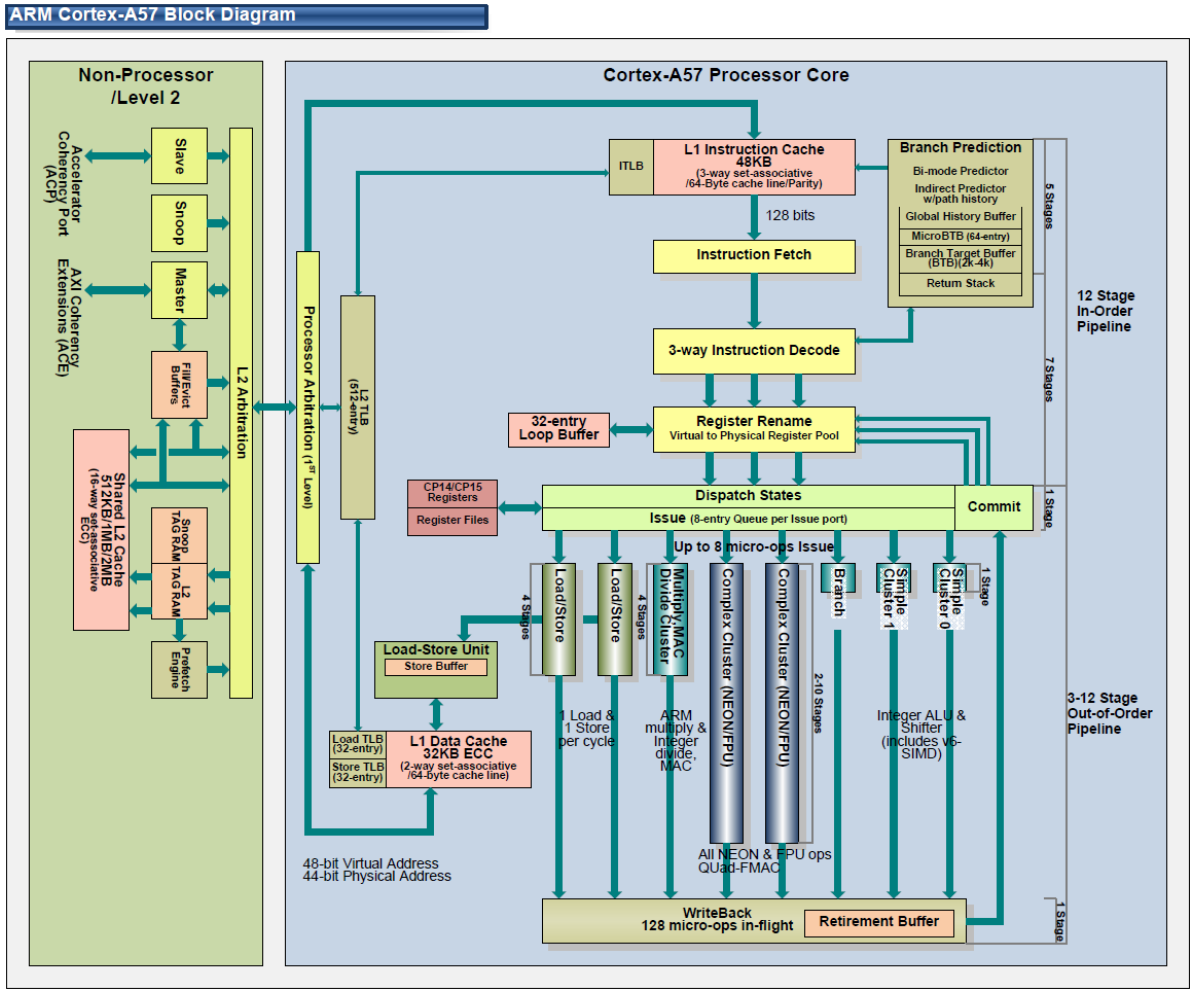


ARMv8 64bit

- Premium smartphones
- Enterprise servers
- Home server
- Wireless Infrastructure
- Digital TV



Cortex A57 Block Diagram



ARM Partnership Model

What did you Learn?

- Variety of possibility for embedded processing
- CPU has an instruction set.
  - Different architecture according to the bus
- According to the architecture we can run more fast or less fast a task
- **Energy is different than power Possible exam exercise!**
  - **Frequency affect the latency and the energy, as well as the power**
- ARM processors