ECE/CS 5780-6780 : Embedded Systems Design

Lect. 02: ARM Processors: Architecture

Pierre-Emmanuel Gaillardon

Department of Electrical and Computer Engineering – University of Utah

Spring 2017

Salt Lake City, UT, USA

IMFlash TechTalks

TECHTalks

Win $200 and an opportunity to interview with IM Flash. Create a 5 minute presentation on the topic listed at right. Apply by Jan. 25. If your presentation is selected you will present and could win $200

- Feb. 1, 2017 – at IM Flash

"Where do we go from here?
- How does Moore's law apply beyond 20 nm?"

All TECHTalks are at 6:00 PM

To apply and for more info, go to

http://imflash.com/ustar

Submit presentation outlines by Jan 25 to Jake LaMarr, jlamarr@imflash.com

IM Flash, 4000 N Flash Drive, Lehi, UT 84043

FLASH
an intel, micron venture

Objectives for Today!

- **Get familiar with the ARM M processor architecture**
- Understand the architecture of a microcontroller
- Acquire the fundamentals of the hardware/software interface
- Acquire the fundamentals for sensing and controlling the physical world
- Learn modeling techniques for embedded system design
- Understand the usage of several peripherals through labs
- Design a complete embedded system – from specs to realization
  - **Realization of a PCB**
  - Behavioral modeling of the system
  - Complete SW realization

Homework

- Usage of L298N:
  - What is the max. current per channel?
  - What is the stall current of the motor?
  - What is the impact?
  - Sense resistor: Value? Package? Power ratings?
- Encoders:
  - What is the minimum voltage for the encoders?
  - What are the logic levels?
  - Can I connect that to my MCU?
- I^2C:
  - What about the pull-ups?

Electrostatic Discharge (ESD)

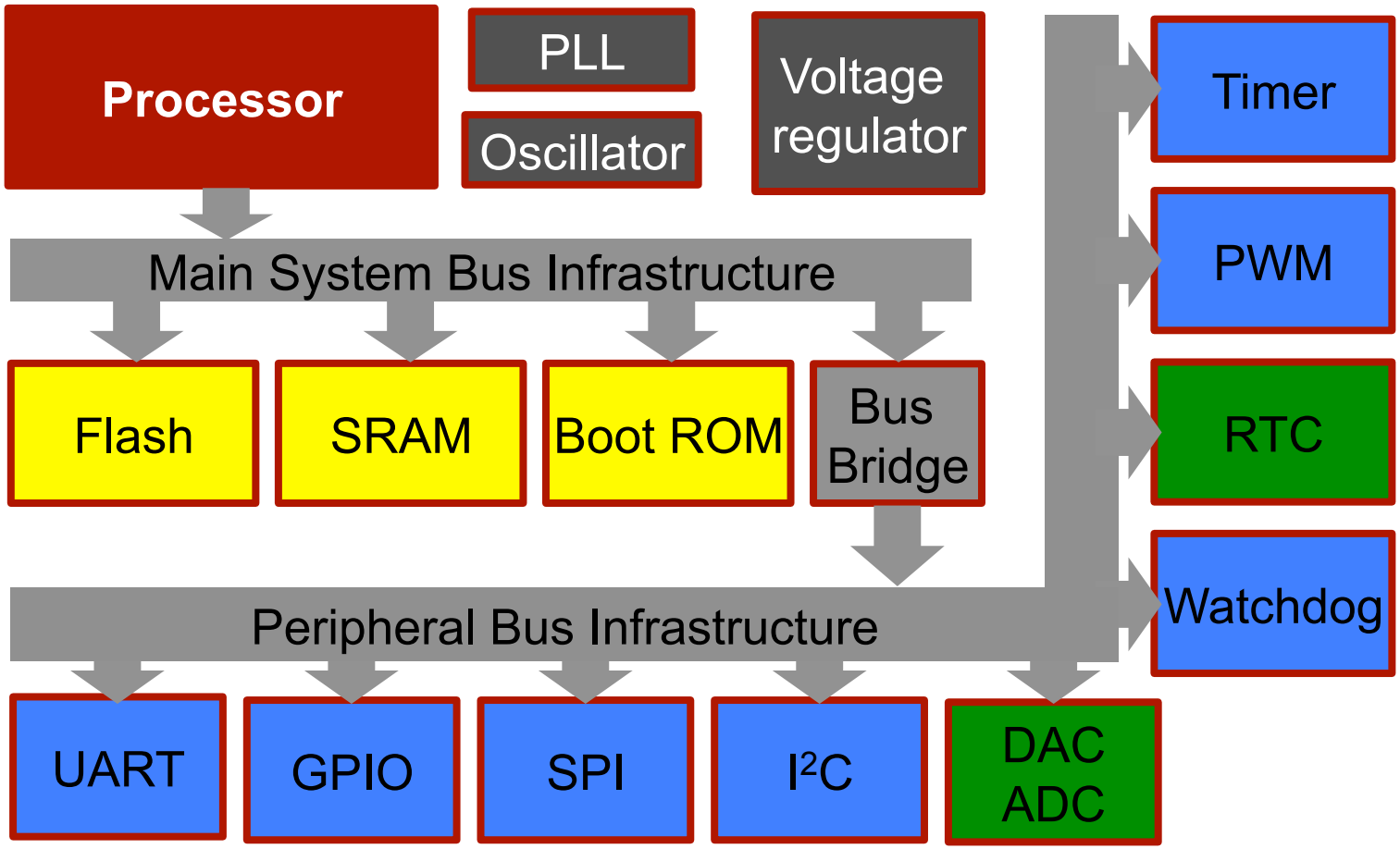
- What is its origin?
- Why does it impact electronics?
- What can we do to prevent them?
  - Circuit design
  - Careful manipulation
- What do you have to do in labs and at home?

Generalities

Microprocessor vs. Microcontroller?

- Microprocessor:
  - A chip device containing processor(s)
  - Designed primarily to handle computational tasks
  - Typically need to access memory and peripherals
- Microcontroller:
  - A chip device containing processor(s)
  - Designed to handle control and computational tasks
  - This chip typically groups in the same package a processor core, a memory system and a number of peripherals

What is inside a microcontroller?



Programming an Embedded System

- Who have been doing programming on a PC?
- Forget about it! Things are very different because:
  - Very small memory footprint (16KB of flash, 8KB of SRAM)
  - Simple user interface (few buttons, LEDs)
  - No file systems
  - No operating systems (or low memory footprints RTOS)
  - You still need a PC to do the software development and need tools to transfer the developed program code to the target.

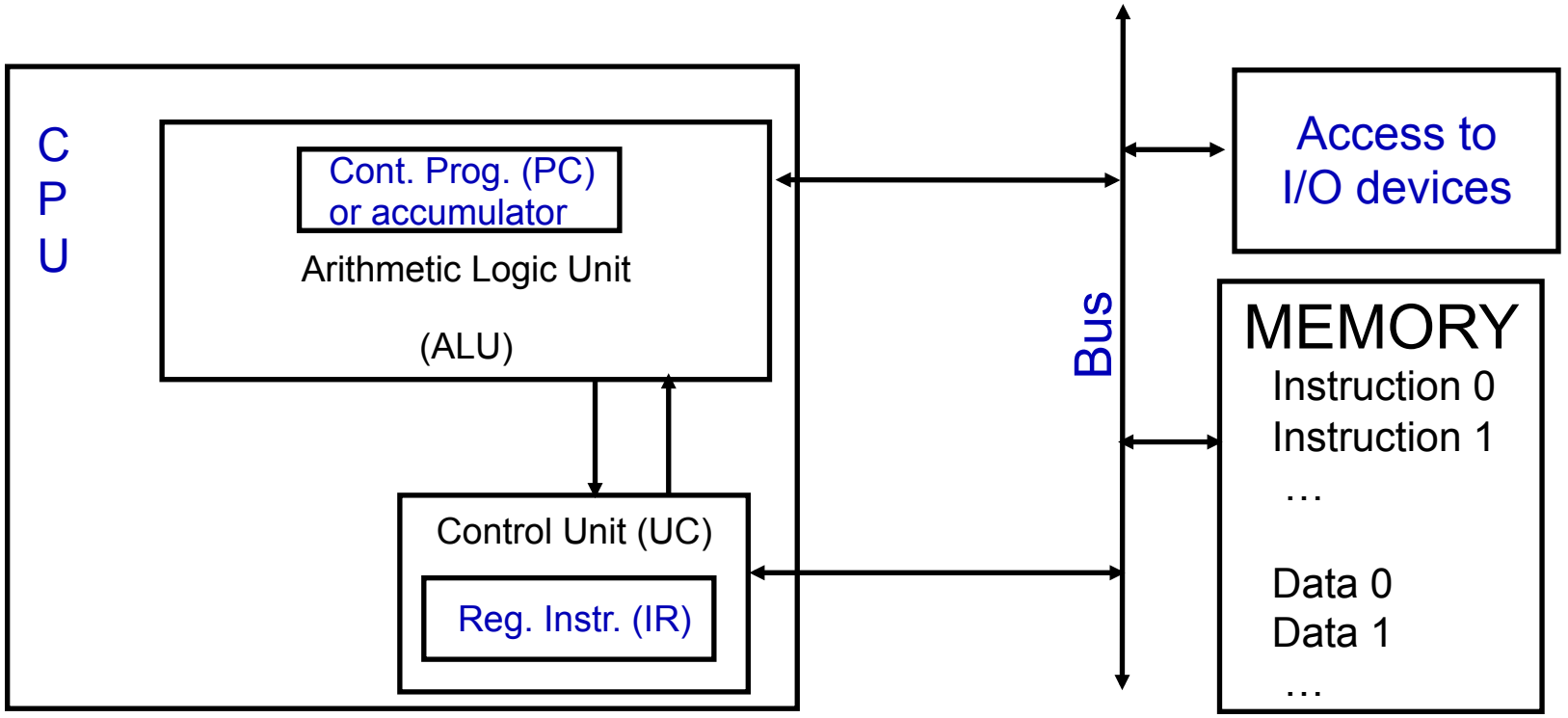
Need to know the architecture and master the toolchain

Some General Refresh about Processors

Microprocessor Architectures

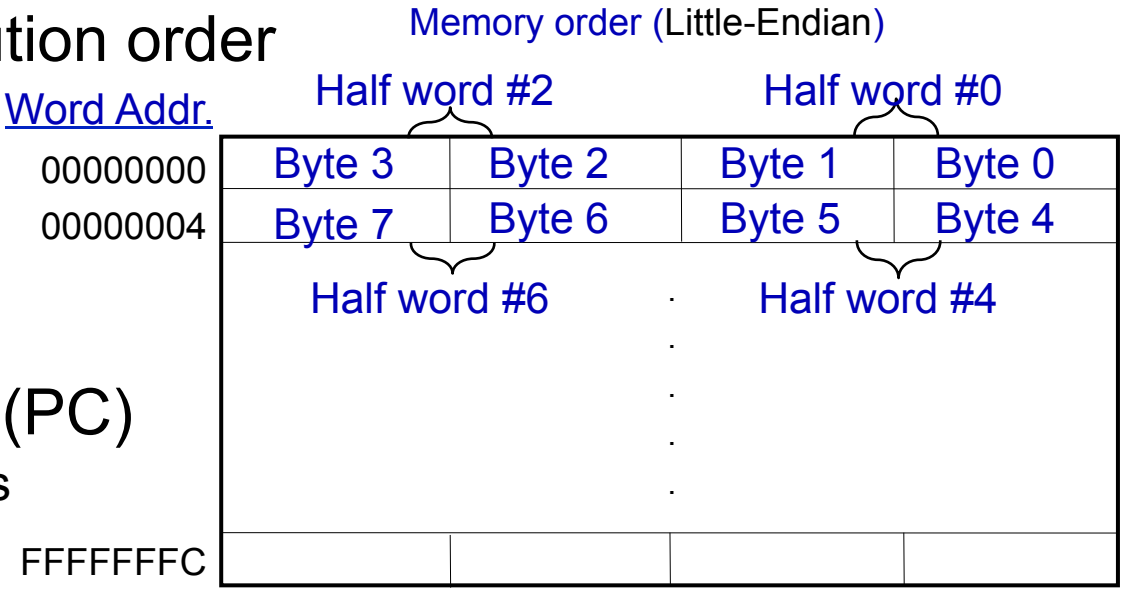
- A microprossor reads each instruction of a program stored in memory in sequence, to perform the basic arithmetical, logical, and input/output operations of the system

Von Neumann Architecture [1945]



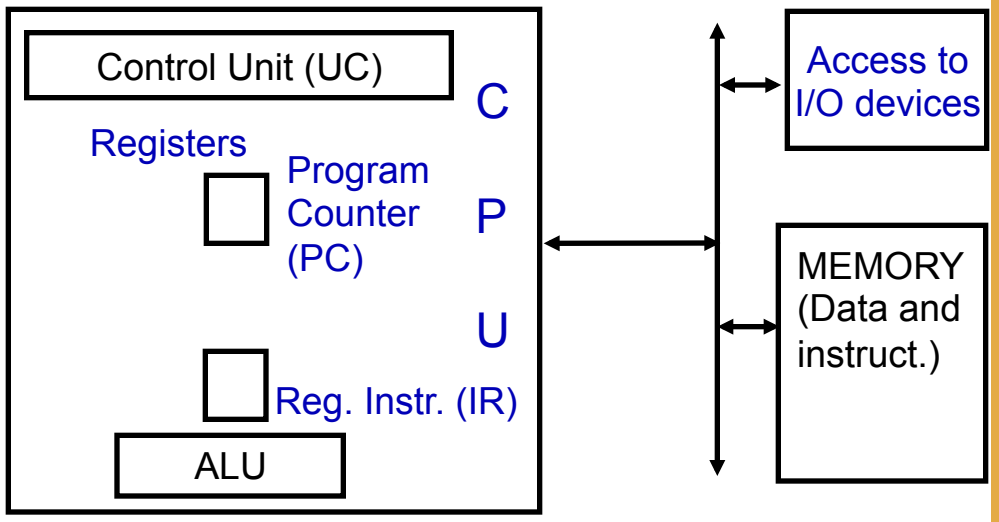
Main features of Von Neumann architecture

- It is based on the concept of a memory stored program
  - The memory content is organized in words, all of the same size (e.g., 32 bits)
  - Words in memory follow a linear order (i.e., addresses), with endianness
- Two main memory contents, but no explicit distinction between them
  - Instructions: program that controls what the microprocessor has to do
  - Data: pieces of information that the program processes and generates
- The instructions execution order is sequential (0, 1, …)
  - Determined by their addresses order in the memory
- The Program Counter (PC) keeps next instruction address to execute



Main features of Von Neumann architecture

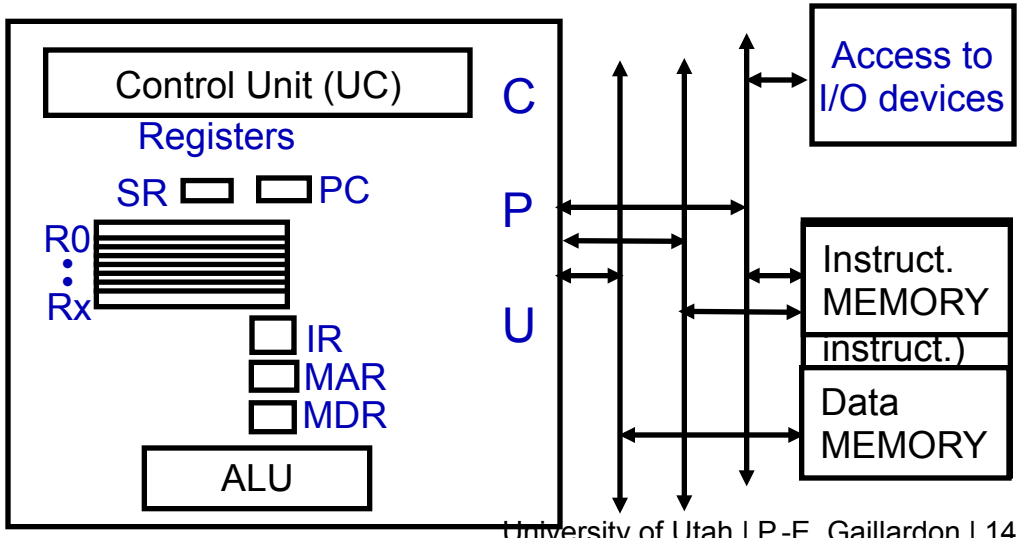
- Five phases in program instruction execution
  1. Search instruction in memory *(Fetch)* and compute next instruct. address
  2. Decode instruction by the CPU
  3. Search of instruction operands (in memory or registers)
  4. Execution of the instruction (in ALU if an arithmetic one)
  5. Write results of operation (in memory or registers)



Variations of baseline Von Neumann architecture

- Two alternatives in the instruction set of a CPU
  - **Complex instruction set computer** (CISC): design strategy to bridge the semantic gap through single instructions that execute several low-level operations (e.g., load from memory, arithmetic operat., and memory store).
    - Examples: Intel x86, Motorola 68x, VAX, System/360, or PDP-11
  - **Reduced instruction set computer** (RISC): design strategy based on simple instructions to provide higher performance through much faster execution of each instruction, i.e., load-store architectures
    - Examples: ARM, MIPS, Atmel AVR, Power, SPARC or AMD 29k.

- Harvard architecture
  - Architectures with physically separated storage and signal buses for instruct., data and I/O
    - Examples: ARM , MIPS, Blackfin, PowerPC



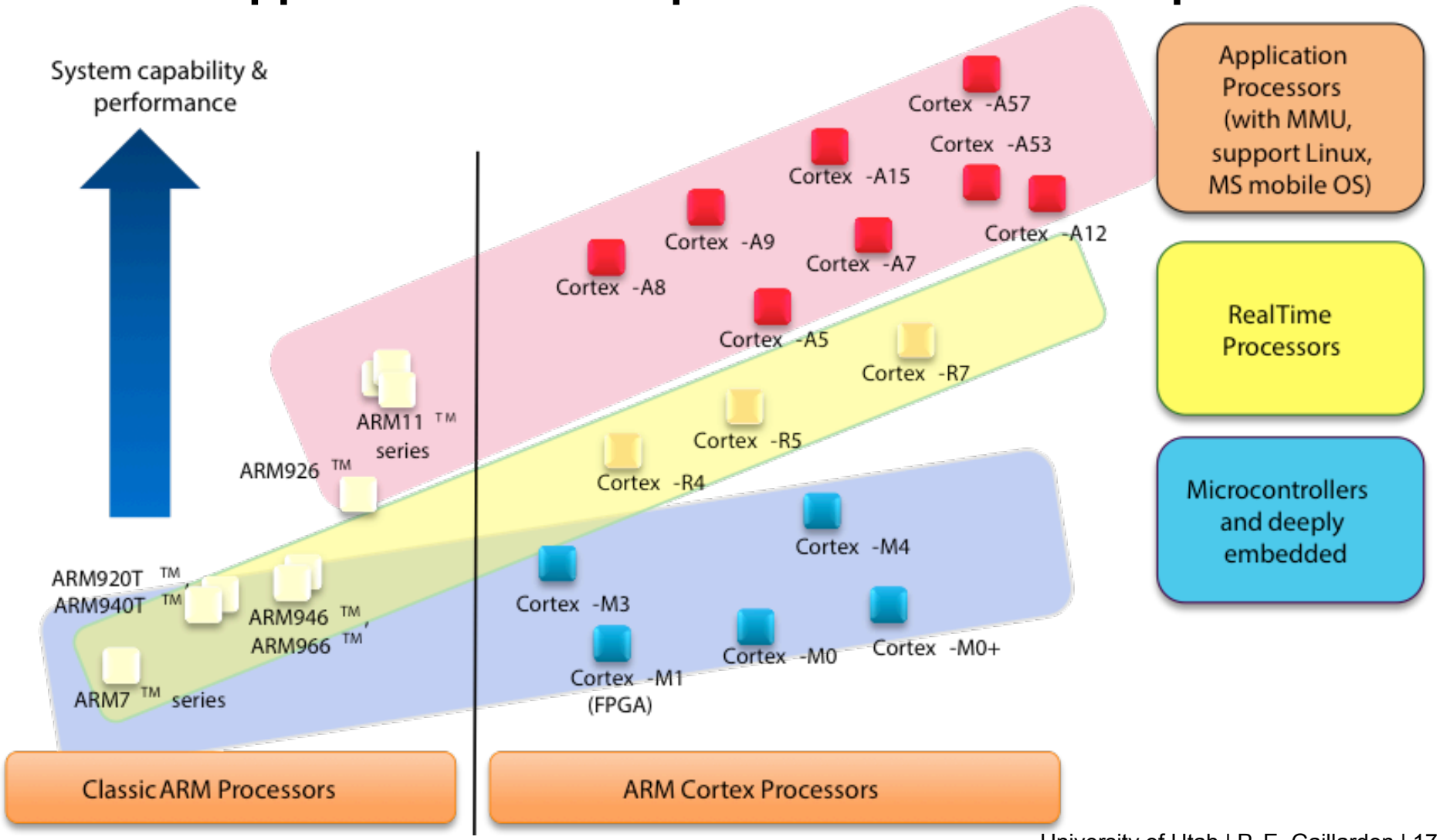
ARM Processors: Generalities

ARM Business Model

- Advanced RISC Machines Ltd (ARM) was officially founded in 1990
- Reduced Instruction Set Computers (RISC) for energy-efficient microprogrammed computing
  - Instructions with minimum semantic (move data, add/sub/mult. 2 numbers, ...)
  - ARM2 [1986], probably simplest 32-bit core ever: 30K transistors, no cache
  - Large range of ARM designs: always basic RISC architecture + extensions
- Business model:
  - Licensing ARM designs to semiconductor partners who fabricate and sell to customers specialized products

Understanding Different Types of Processors

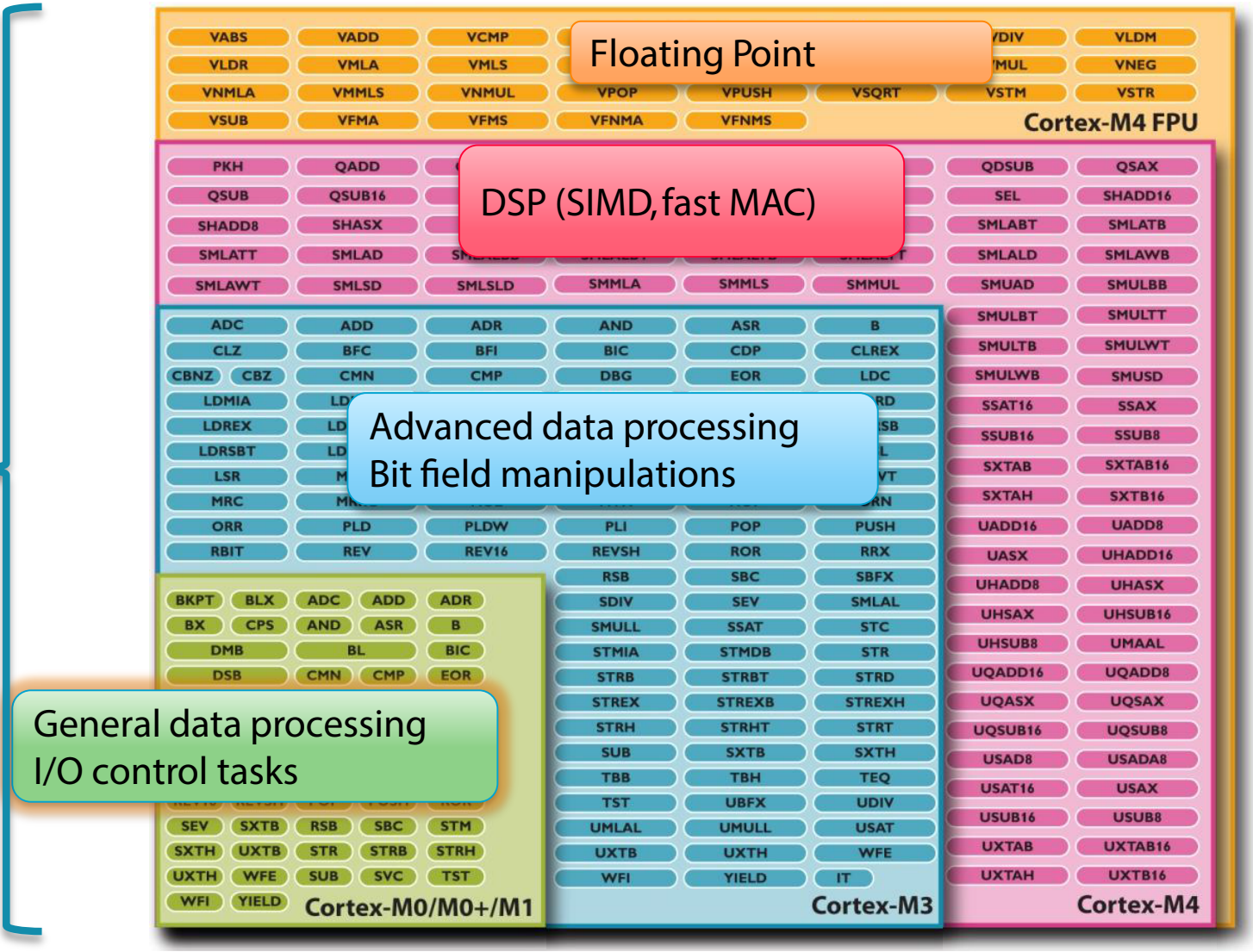
Different applications and requirements = Different processors



Instruction set of the Cortex-M family

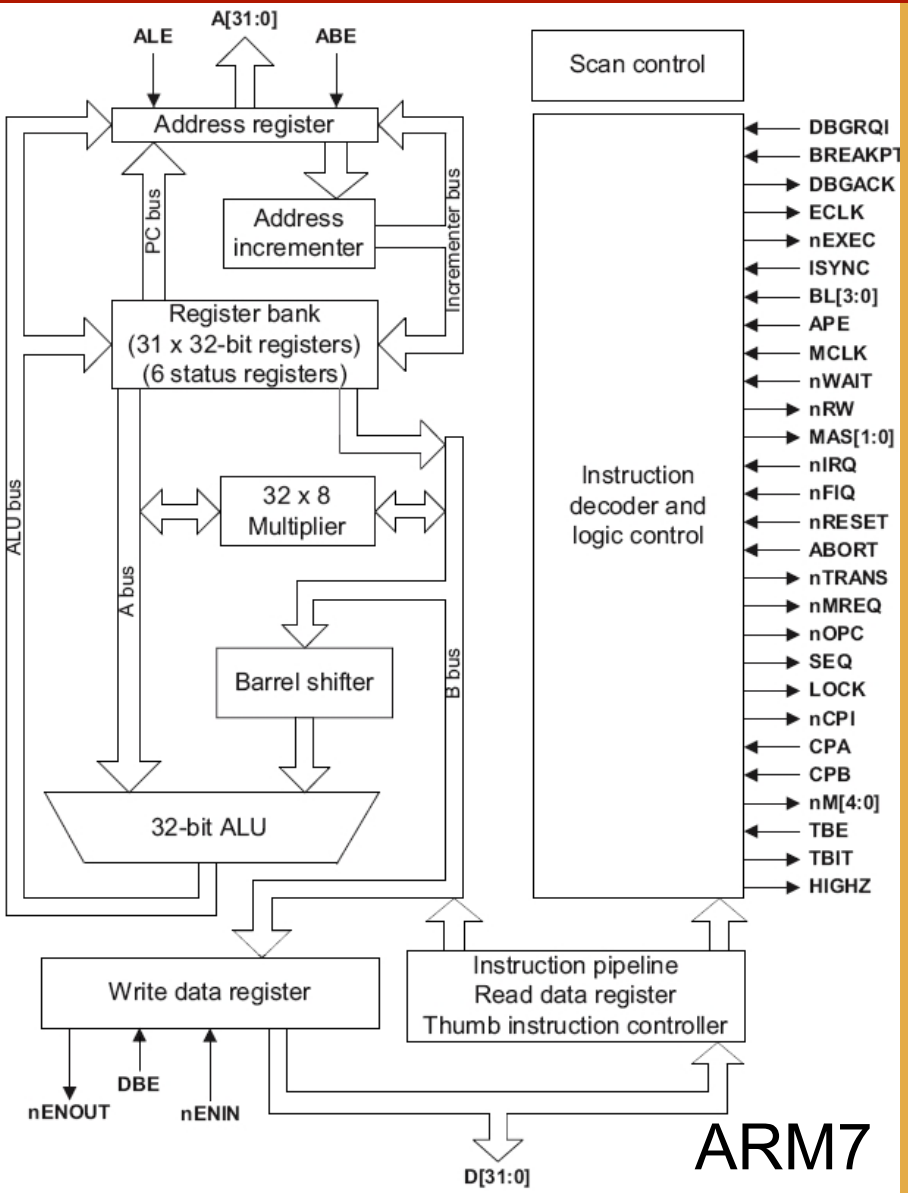
ARMv7 -M Architecture

ARMv6 -M Architecture



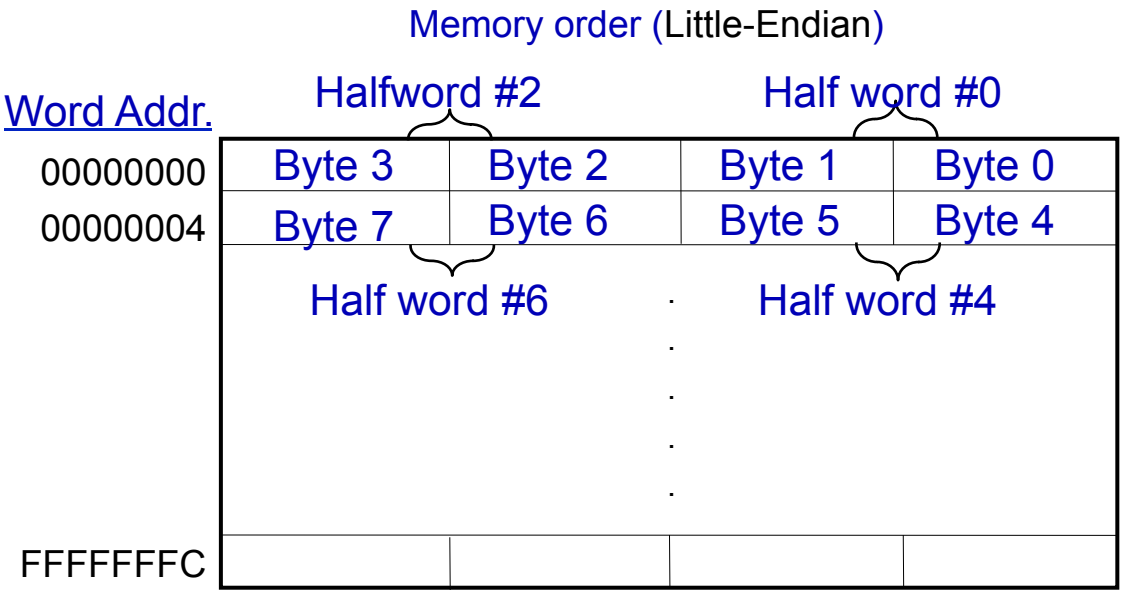
ARM processor architectures

- 32-bit RISC processor (48MHz)
  - Register and data paths = 32 bits
- Three-stage pipeline
  - fetch/decode/execute
- Thumb instruction set
  - Most instructions are in 16bit
  - Large code density



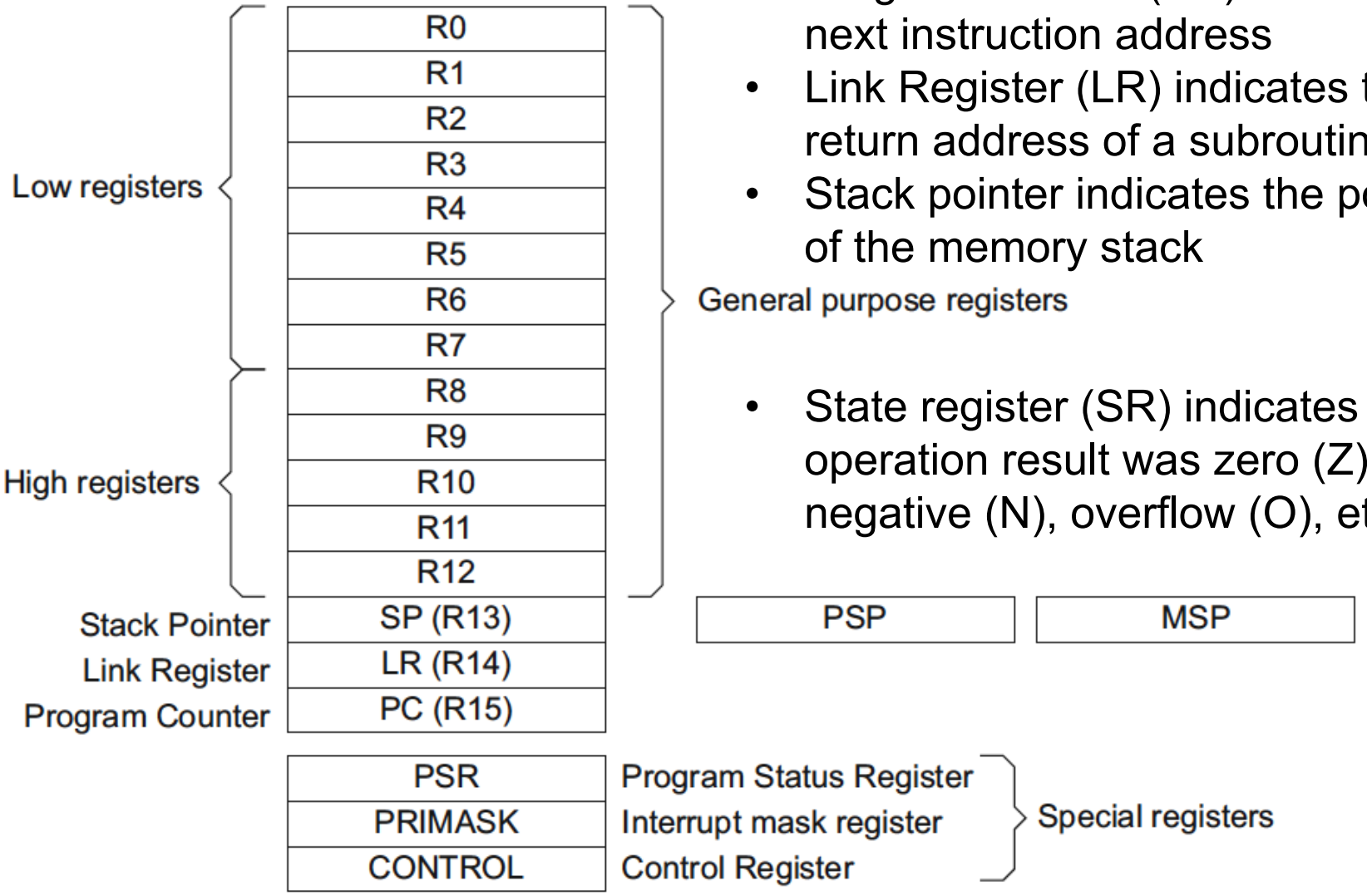
Allowed data sizes and instruction sets

- Possible data sizes in ARM:
  - **Word** means 32 bits (four bytes)
  - **Halfword** means 16 bits (two bytes)
  - **Byte** means 8 bits (one byte)
- By default: little-endian (big-endian is configurable)



- Most ARM processors implement two instruction sets
  - Regular instructions of 1 word each: 32-bit ARM Instruction Set
  - Compact instructions of halfword: 16-bit Thumb Instruction Set
- Well-known extensions of the ARM instruction set
  - Jazelle: direct execution of Java bytecode
  - NEON: acceleration for multimedia and signal processing

User / System Registers



- Program counter (PC) indicates next instruction address
- Link Register (LR) indicates the return address of a subroutine
- Stack pointer indicates the position of the memory stack

- State register (SR) indicates if last operation result was zero (Z), negative (N), overflow (O), etc.

Combined Program Status Register (*xPSR*)

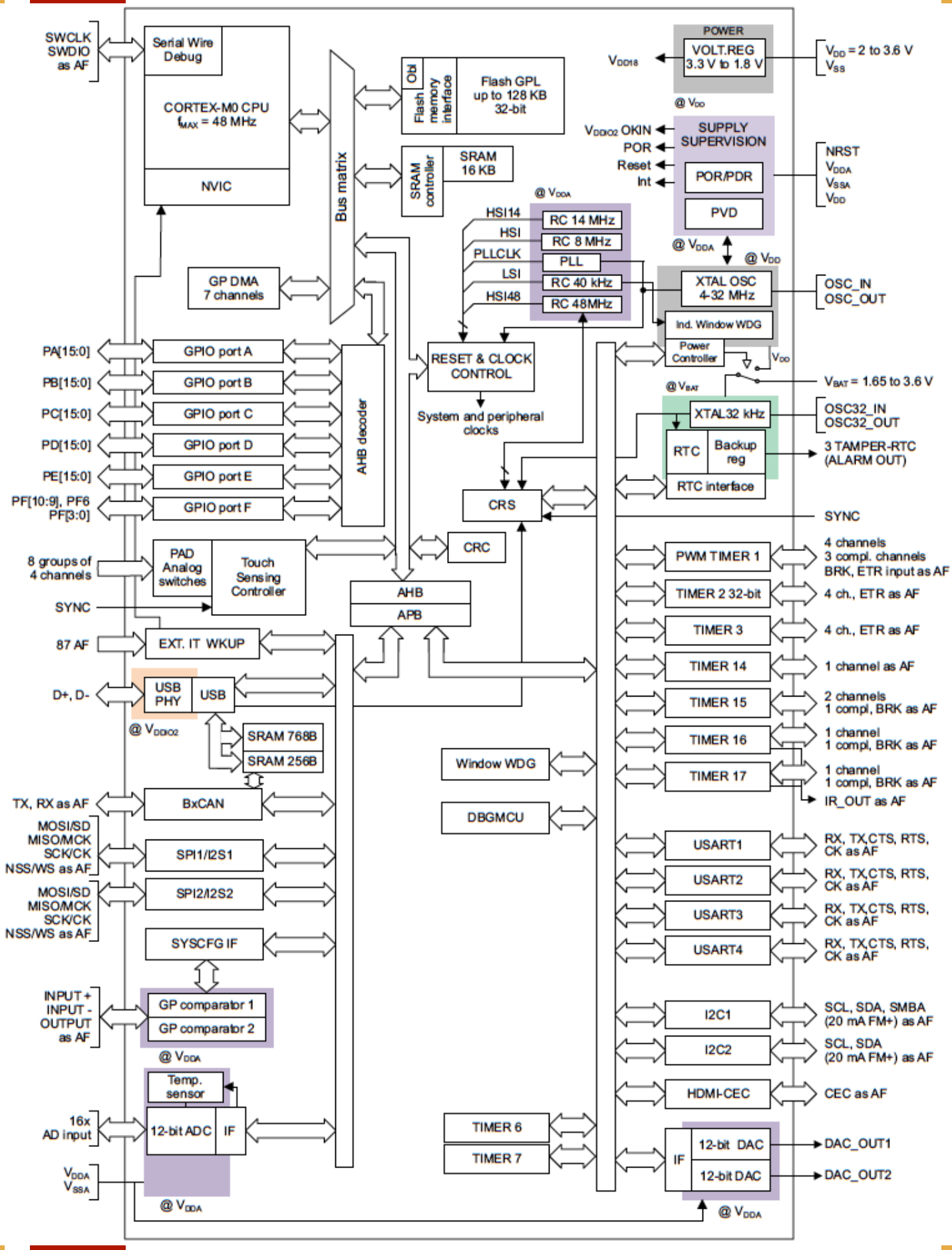
- Condition code flags (or bits)
  - N = **N**egative result from ALU
  - Z = **Z**ero result from ALU
  - C = ALU operation **C**arried out
  - V = ALU operation o**V**erflowed
- T Bit
  - T = 0: Processor in ARM state
  - **T = 1**: Processor in Thumb state
- ISR Number
  - Current executing Interrupt Service Routine

STM32F072RBT6 Microcontroller

- Are things are simple as the basic ARM core?

- What about peripherals?

They lead the game!!!


SWCLK
SWDIO
as AF
Serial Wire Debug
CORTEX-M0 CPU
f_{MAX} = 48 MHz
NVIC
Bus matrix
Obl
Flash memory interface
Flash GPL up to 128 KB 32-bit
SRAM controller
SRAM 16 KB
POWER
V_{DD18}
VOLT.REG 3.3 V to 1.8 V
V_{DD} = 2 to 3.6 V
V_{SS}
@ V_{DD}
V_{DDIO2} OKIN
POR
Reset
Int
SUPPLY SUPERVISION
POR/PDR
PVD
NRST
V_{DDA}
V_{SSA}
V_{DD}
@ V_{DDA}
@ V_{DD}
HSI14
HSI
PLLCLK
LSI
HSI48
RC 14 MHz
RC 8 MHz
PLL
RC 40 kHz
RC 48MHz
XTAL OSC 4-32 MHz
OSC_IN
OSC_OUT
Ind. Window WDG
Power Controller
V_{DD}
V_{BAT} = 1.65 to 3.6 V
@ V_{BAT}
XTAL32 kHz
OSC32_IN
OSC32_OUT
RTC
Backup reg
3 TAMPER-RTC (ALARM OUT)
RTC interface
GP DMA 7 channels
RESET & CLOCK CONTROL
System and peripheral clocks
CRS
SYNC
PA[15:0]
PB[15:0]
PC[15:0]
PD[15:0]
PE[15:0]
PF[10:9], PF6
PF[3:0]
GPIO port A
GPIO port B
GPIO port C
GPIO port D
GPIO port E
GPIO port F
AHB decoder
8 groups of 4 channels
PAD Analog switches
Touch Sensing Controller
CRC
AHB
APB
SYNC
87 AF
EXT. IT WKUP
D+, D-
USB PHY
USB
@ V_{DDIO2}
SRAM 768B
SRAM 256B
PWM TIMER 1
4 channels
3 compl. channels
BRK, ETR input as AF
TIMER 2 32-bit
4 ch., ETR as AF
TIMER 3
4 ch., ETR as AF
TIMER 14
1 channel as AF
TIMER 15
2 channels
1 compl, BRK as AF
TIMER 16
1 channel
1 compl, BRK as AF
TIMER 17
1 channel
1 compl, BRK as AF
IR_OUT as AF
Window WDG
DBGMCU
TX, RX as AF
BxCAN
MOSI/SD
MISO/MCK
SCK/CK
NSS/WS as AF
SPI1/I2S1
SPI2/I2S2
SYSCFG IF
USART1
USART2
USART3
USART4
RX, TX,CTS, RTS, CK as AF
INPUT +
INPUT -
OUTPUT
as AF
GP comparator 1
GP comparator 2
@ V_{DDA}
I2C1
I2C2
SCL, SDA, SMBA (20 mA FM+) as AF
SCL, SDA (20 mA FM+) as AF
Temp. sensor
HDMI-CEC
CEC as AF
16x AD input
12-bit ADC
IF
TIMER 6
TIMER 7
12-bit DAC
DAC_OUT1
DAC_OUT2
V_{DDA}
V_{SSA}
@ V_{DDA}
@ V_{DDA}

Thank you for your attention

Questions?

Laboratory for NanoIntegrated Systems
Department of Electrical and Computer Engineering
MEB building – University of Utah – Salt Lake City – UT – USA