Introduction to ARM Cortex M0+ Processor

Le Yang (le.yang@canterbury.ac.nz)

Department of Electrical and Computer Engineering

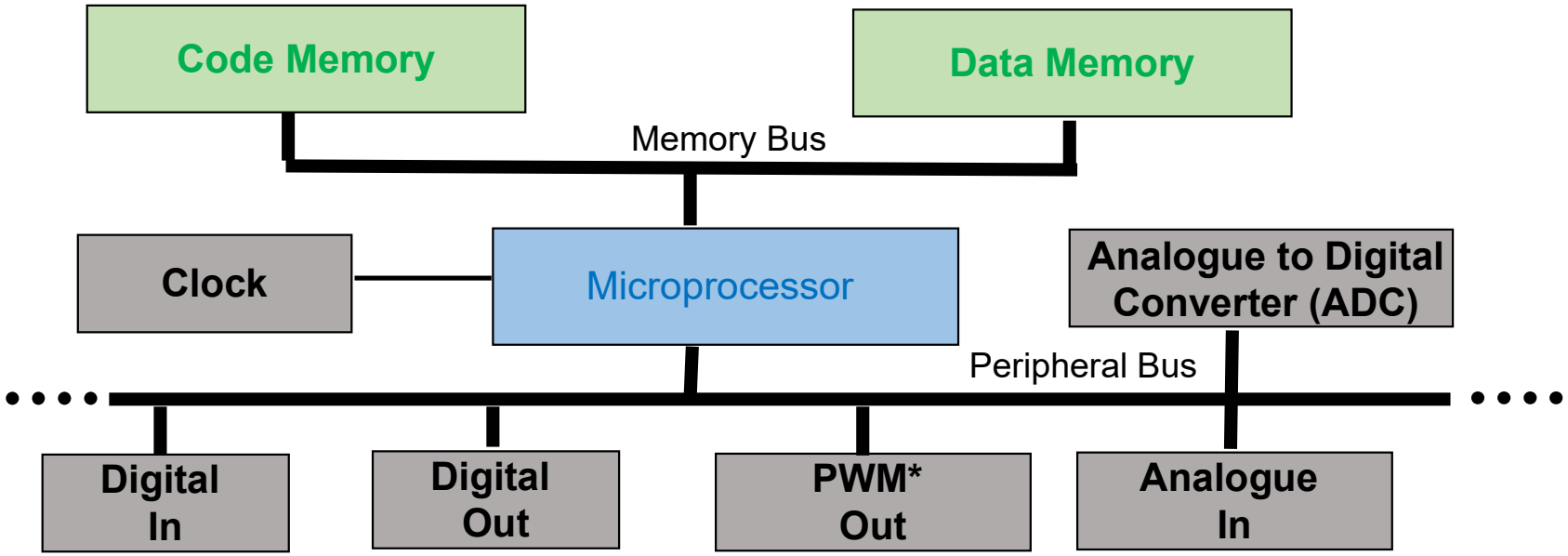
Office Hours: Link Building A510, Mondays 16:00 – 18:00

Where we're going today

- **ARM Cortex-M0+ processor**
- STM320C071x8/xB MCUs
- ARM Cortex-M0+ instruction set

Microcontroller (MCU)

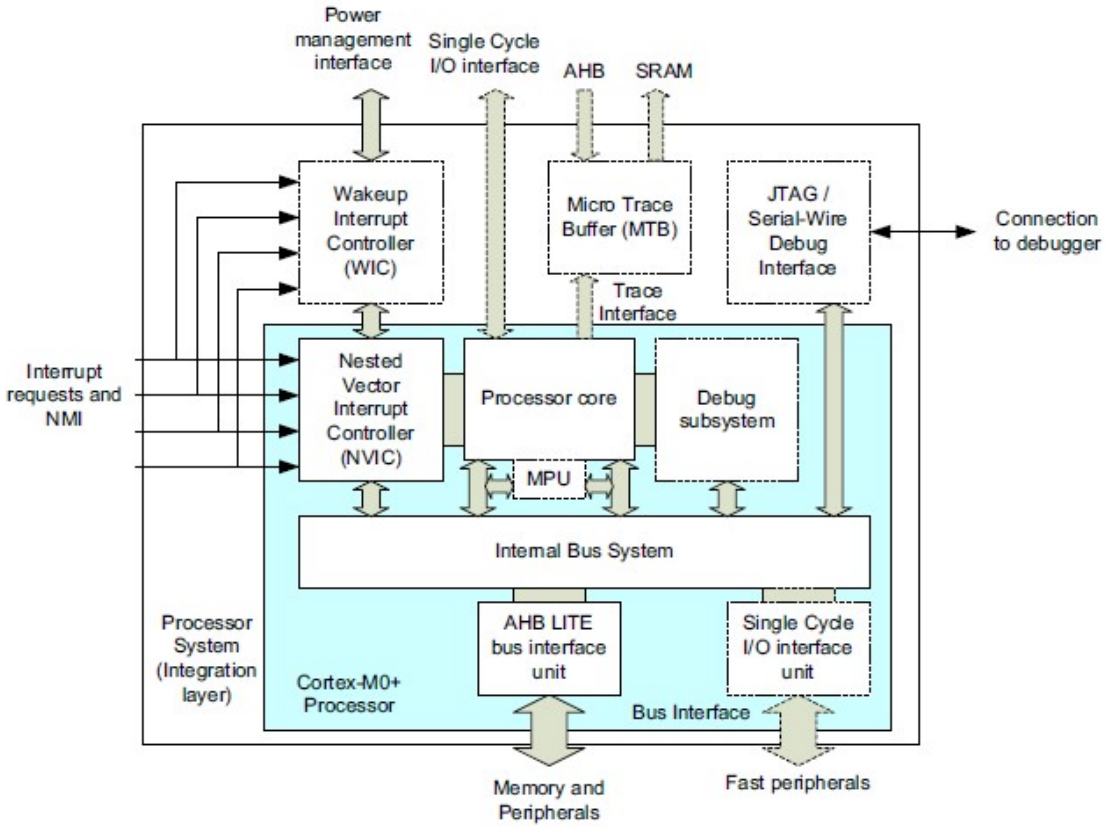
- Microcontroller = Microprocessor (MPU) + Memory + Peripherals
- Microcomputer = microcontroller on a single silicon chip
  - Of which 99% are



* PWM: Pulse Width Modulation

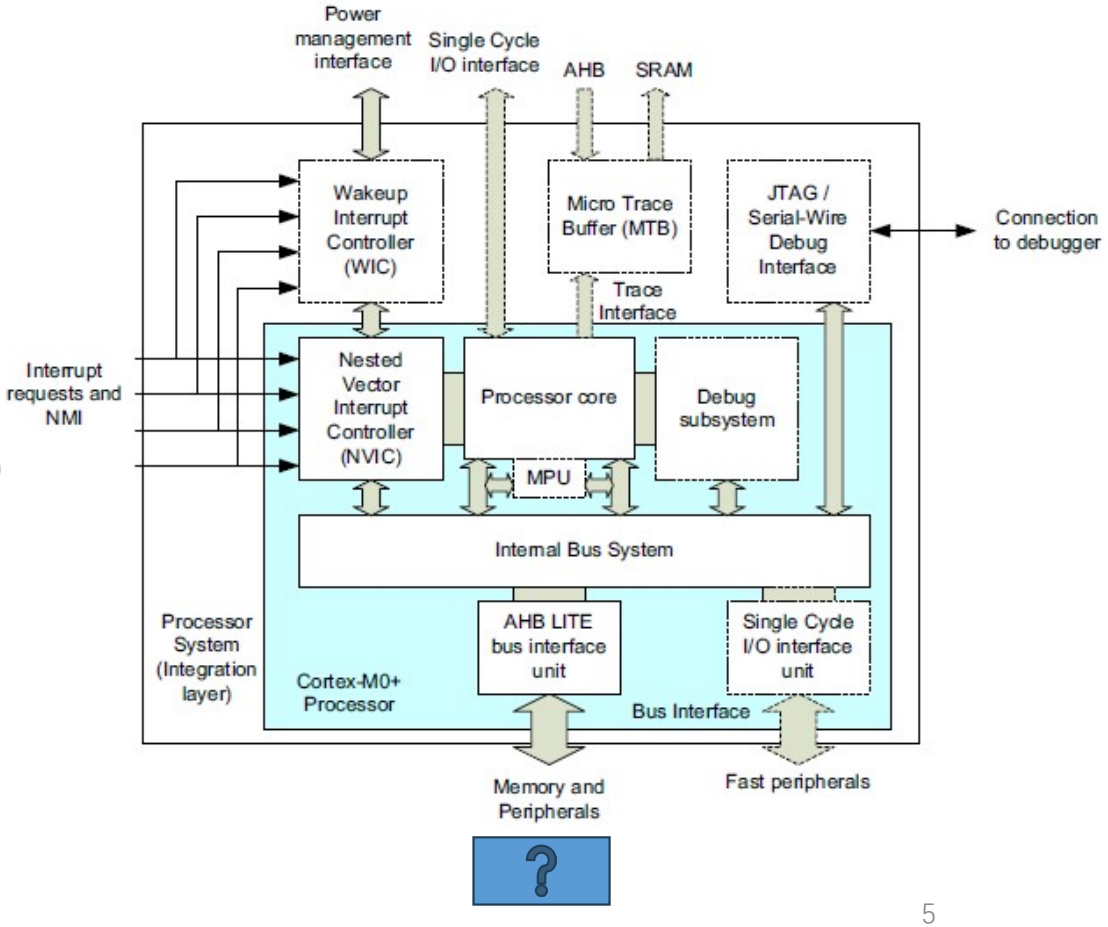
ARM Cortex-M0+ Processor (1)

- 32-bit MPU introduced in 2012
  - ARMv6-M architecture
- Two-stage pipeline
  - Fetch + pre-decode, decode + execute
- Thumb instruction set architecture (ISA)
  - Most instructions are 16-bit
- Von Neumann structure
  - Shared pathway/bus for program and data
  - Pathway/bus competition



ARM Cortex-M0+ Processor (2)

- Built-in interrupt controller
  - Nested vector interrupt controller (NVIC)
  - Interrupt prioritization and masking
  - 4 programmable priority levels + NMI
- Low-power support
  - 9.8 uW/MHz (90 nm semiconductor process)
  - Sleep and deep sleep modes
  - Wait for Input (WFI)/Wait for Event (WFE)
  - Wake-up Interrupt Controller (WIC)
- Debug: JTAG/SWD



ARM and RISC

- ARM: Advanced RISC Machine
  - First developed by Acorn Computers in the mid 80's based on RISC concept originated at Stanford & Berkeley
  - Now a separate company licensing ARM cores to STMicroelectronics, Texas Instruments, Atmel, Motorola …

- RISC vs. CISC
  - RISC: Reduced Instruction Set Computer
    - One instruction for one operation, executed within one clock cycle
    - Larger code size but less complicated hardware (advantage)
  - CISC: Complex Instruction Set Computer
    - One instruction for multiple operations, run over several clock cycles
    - Shorter code length but complex (control) hardware

Pioneers in RISC Computers

- 2022 Charles Stark Draper Prize for Engineering
  - Conceptualization, prototyping and benchmarking in 1980s
    - **David Patterson**, UC Berkeley
    - **John Hennessy**, Stanford
  - Commercialization
    - **Stephen Furber** and **Sophie Wilson**, Acorn Computers
    - Acorn/Advanced RISC machine (ARM)

https://www.nae.edu/266390/RISC-Chip-Innovators-Receive-the-2022-Charles-Stark-Draper-Prize-for-Engineering

UC Inventor of RISC Embedded Processors

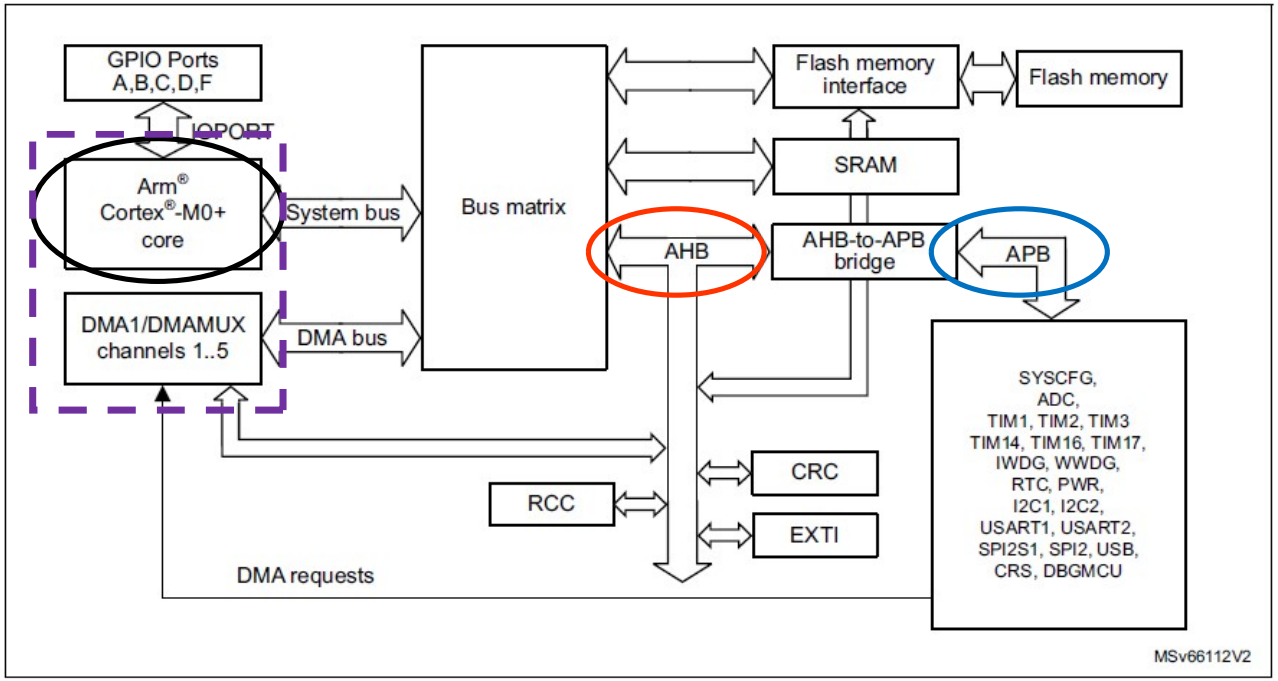
- **David Jagger**
  - B.S. from UC, Computer Science, 1987
  - M.S. from UC, Computer Science, 1991
    - **Thesis title**: A performance study of the Acorn RISC machine
  - Computer Scientist, ARM, 1992-2000
    - Designer of ARM7, ARM10
    - Inventor of **Thumb** architecture
  - James Clerk Maxwell Medal from IEEE, 2019

https://www.canterbury.ac.nz/alumni/our-alumni/notable-alumni/david-jaggar/

Where we're going today

- ARM Cortex-M0+ processor
- **STM320C071x8/xB MCUs**
- ARM Cortex-M0+ instruction set

STM32C071x8/xB MCU Architecture



FLASH: 128 KB

SRAM: 36 KB

RCC: RESET & Clock Control

CRC: Cyclic Redundancy Check

EXTI: Extended Interrupt & Event Controller

- AHB: Advanced High-Performance Bus
  - Better back-to-back access performance
- APB: Advanced Peripheral Bus
  - Legacy bus

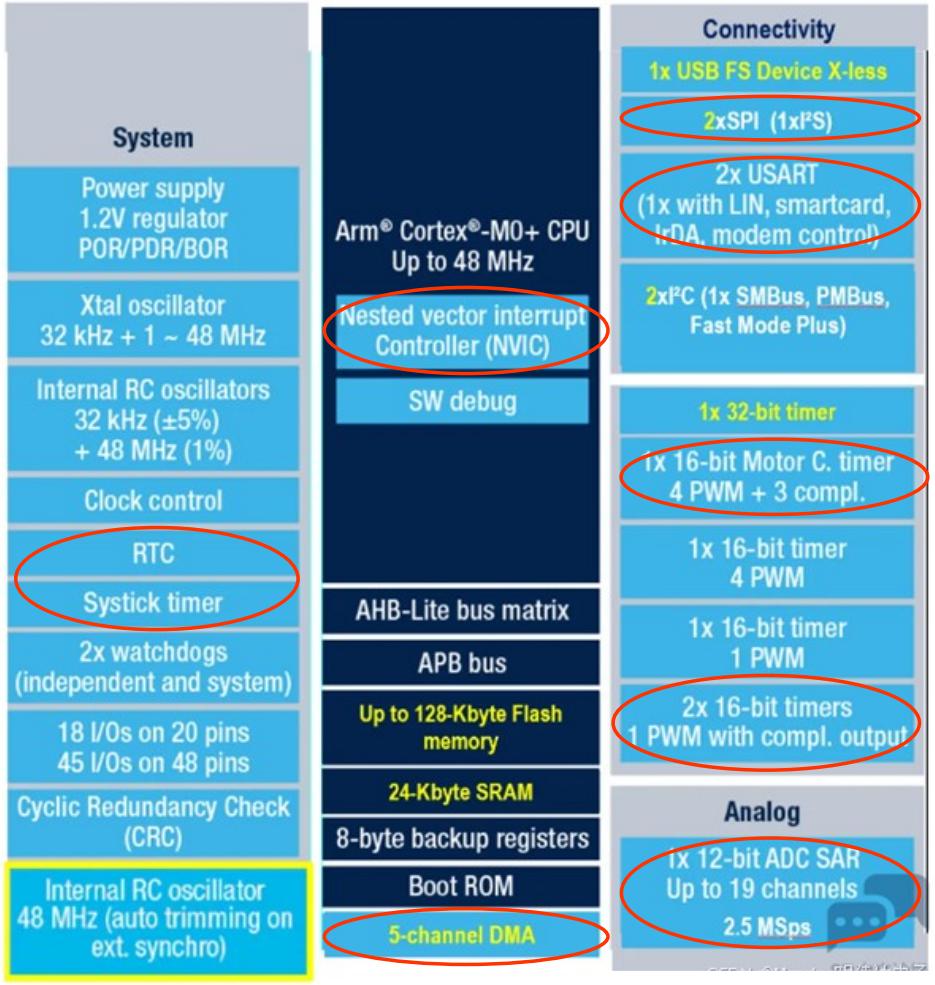
RM0490 Reference manual STM32C0 series advanced Arm®-based 32-bit MCUs.pdf

STM32C071x8/xB MCUs

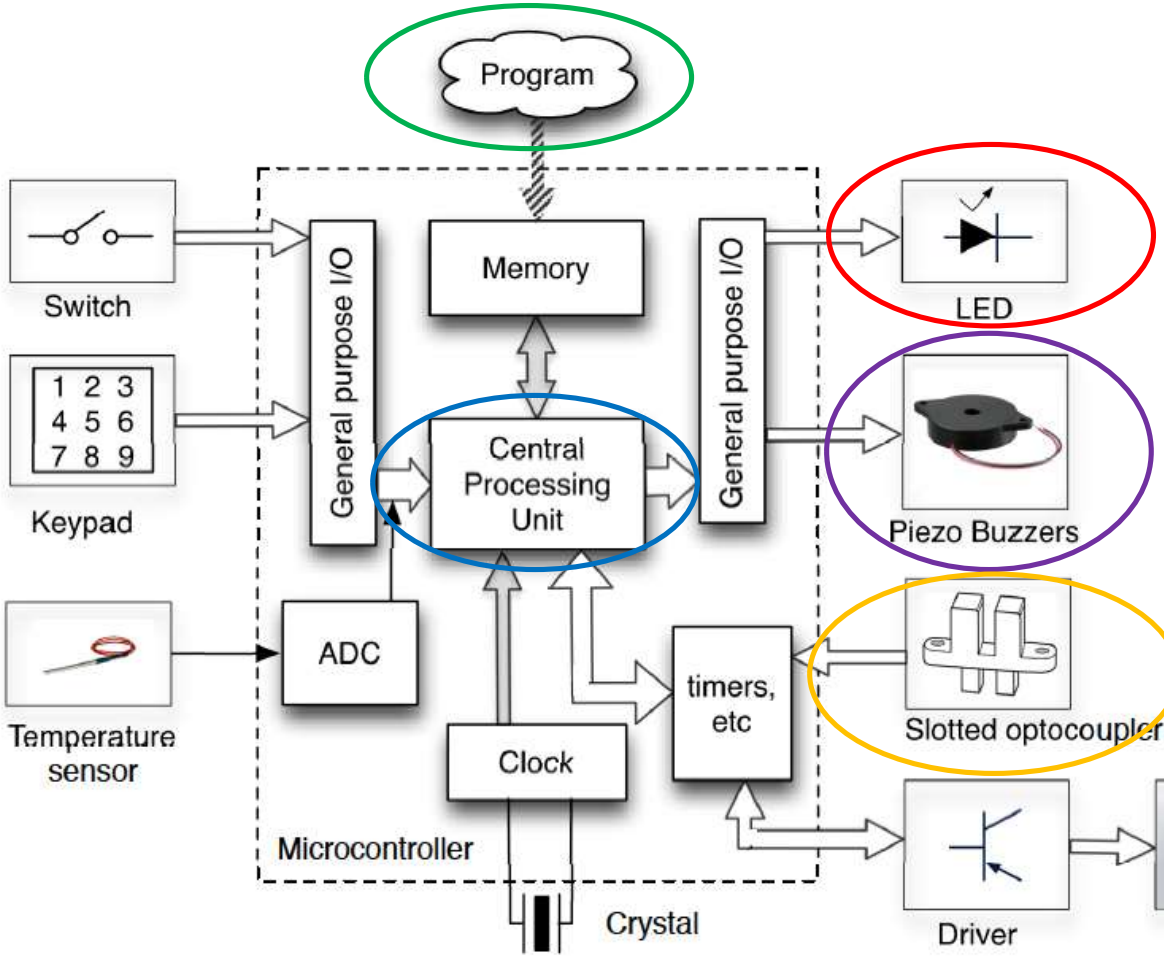
- Based on ARM Cortex-M0+ CPU
  - **STM32C071RBT6**
  - STM32C071CB
  - STM32C071KB
  - STM32C071GBU6

- Circles: modules relevant to this course

- Homework: find out the definition of each acronym in the diagram



Typical Application Scenario



Program: Code + Data

Central Processing Unit (CPU)

- ARM Cortex-M0+ processor

LED: Lights Emitting Diode

- Semiconducter light source for signalling

Piezo buzzer

- Audio signalling device to indicate button pressing (click, ring or beep)

Optocoupler: light emitter + receiver

- Signal detection in electrical noise …

Where we're going today

- ARM Cortex-M0+ processor
- STM320C071x8/xB MCUs
- **ARM Cortex-M0+ instruction set**

Example of Cortex-M0+ Instructions

Table 3-1 Cortex-M0+ instructions

Mnemonic	Operands	Brief description	Flags	Page
ADC{S}	{Rd,} Rn, Rm	Add with Carry	N,Z,C,V	page 3-20
ADD{S}	{Rd,} Rn, <Rm|#imm>	Add	N,Z,C,V	page 3-20
ADR	Rd, label	PC-relative Address to Register	-	page 3-12
AND{S}	{Rd,} Rn, Rm	Bitwise AND	N,Z	page 3-20
ASRS	{Rd,} Rm, <Rs|#imm>	Arithmetic Shift Right	N,Z,C	page 3-24
B{cc}	label	Branch {conditionally}	-	page 3-34

56 16-bit instructions
6 32-bit instructions

NOT for heavy-duty number-crunching tasks

S: cause an instruction to update flags (Negative, Zero, Carry, oVerflow)

Rd: result register

Rn/Rm: First/second source register

cc: conditional execution (EQ, NE, GT, LT, LE, …)

Cortex-M0+ Devices_Generic Users Guide.pdf

Example Program in Assembly

- BEQ label ; Branch to label if previous operation results in equal status (Z = 1)
- ADD R0, R1, R2 ; Carry out R0 = R1 + R2 **without** affecting the flags

- What does the following code do? (refer to slide 18)

```
                CMP R0, #9
                BLE  label_LE
                ADD R1, R0, #55
                B       next
label_LE        ADD R1, R0, #48
    next        …
```

Supplementary Materials

- Fundamental data types
- ASCII table
- ARM licensing
- Evolution of ARM process architecture

Fundamental Data Types

Type Class	Machine Type	Byte size	Byte alignment	Note
Integral	Unsigned byte	1	1	Character
	Signed byte	1	1	
	Unsigned half-word	2	2	
	Signed half-word	2	2	
	Unsigned word	4	4	
	Signed word	4	4	
	Unsigned double-word	8	8	
	Signed double-word	8	8	
Floating Point	Single precision (IEEE 754)	4	4	The encoding of floating point numbers is described in [ARM ARM] chapter C2, *VFP Programmer's Model*, *§2.1.1 Single-precision format*, and *§2.1.2 Double-precision format*.
	Double precision (IEEE 754)	8	8	
Pointer	Data pointer	4	4	Pointer arithmetic should be unsigned.
	Code pointer	4	4	Bit 0 of a code pointer indicates the target instruction set type (0 ARM, 1 Thumb).

Table 1, Byte size and byte alignment of fundamental data types

ASCII Table

- American Standard Code for Information Interchange
  - Numerical representation of characters

- Code 48 – 57
  - ‘0’ – ‘9’

- Code 65 – 70
  - ‘A’ – ‘F’

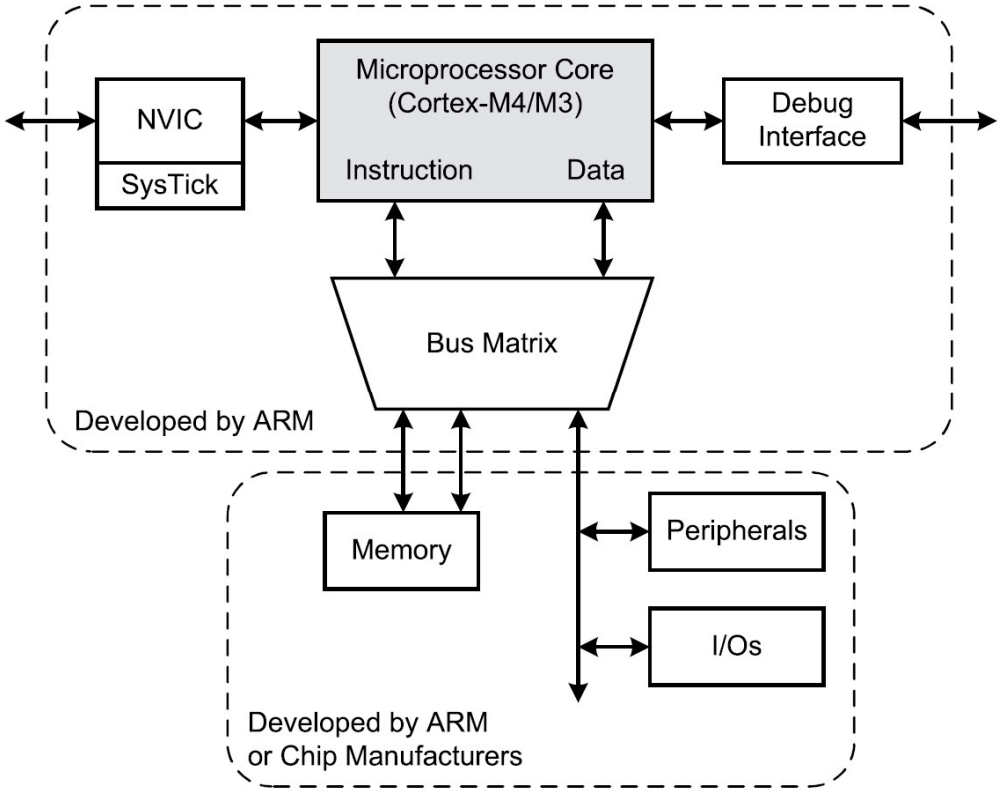
- Hint: hex numerical system has 16 symbols …

Dec	Hex	Oct	Binary	Char	Dec	Hex	Oct	Binary	Char
32	20	040	0100000	space	64	40	100	1000000	@
33	21	041	0100001	!	65	41	101	1000001	A
34	22	042	0100010	"	66	42	102	1000010	B
35	23	043	0100011	#	67	43	103	1000011	C
36	24	044	0100100	$	68	44	104	1000100	D
37	25	045	0100101	%	69	45	105	1000101	E
38	26	046	0100110	&	70	46	106	1000110	F
39	27	047	0100111	'	71	47	107	1000111	G
40	28	050	0101000	(	72	48	110	1001000	H
41	29	051	0101001	)	73	49	111	1001001	I
42	2A	052	0101010	*	74	4A	112	1001010	J
43	2B	053	0101011	+	75	4B	113	1001011	K
44	2C	054	0101100	,	76	4C	114	1001100	L
45	2D	055	0101101	-	77	4D	115	1001101	M
46	2E	056	0101110	.	78	4E	116	1001110	N
47	2F	057	0101111	/	79	4F	117	1001111	O
48	30	060	0110000	0	80	50	120	1010000	P
49	31	061	0110001	1	81	51	121	1010001	Q
50	32	062	0110010	2	82	52	122	1010010	R
51	33	063	0110011	3	83	53	123	1010011	S
52	34	064	0110100	4	84	54	124	1010100	T
53	35	065	0110101	5	85	55	125	1010101	U
54	36	066	0110110	6	86	56	126	1010110	V
55	37	067	0110111	7	87	57	127	1010111	W
56	38	070	0111000	8	88	58	130	1011000	X
57	39	071	0111001	9	89	59	131	1011001	Y

ARM Licensing (1)

- Licensee must follow ARM CPU architecture and instruction set
  - But free to implement their own peripherals (I/O ports, ADCs, Timers, SPI…)
    - Functional registers and their physical locations are not standardized
  - Assembly language programs for ARM chip can be run on any ARM chip
  - Assembly language programs for peripherals on one ARM chip may not be able to run on other ARM chips
- Two approaches for programming ARM chip peripherals
  - Use propriety device library of functions from ARM chip vendor
  - Write your own programs with customized interfaces

ARM Licensing (2)



Evolution of ARM Processor Architecture

Processor Family	ARM7TDMI	ARM9E	ARM11, Cortex-M0, Cortex-M1	Cortex-M, Cortex-R, Cortex-A
Processor Cores	ARM7TDMI, 920T, Intel StrongARM	ARM926, 946, 966, Intel XScale	ARM1136, 1176, 1156T2 Cortex-M0, Cortex-M1 (FPGA)	Cortex-M3/M4, Cortex-R4, Cortex-A8
Architecture Version	ARMv4/v4T	ARMv5/v5E	ARMv6	ARMv7

Figure 2.1: Different versions of ARM processor architecture and their evolution.