School of Mechanical & Aerospace Engineering

Design, Machine, Control and Intelligence

MA4832

Data Representation

Xie Ming, PhD (France)

http://personal.ntu.edu.sg/mmxie

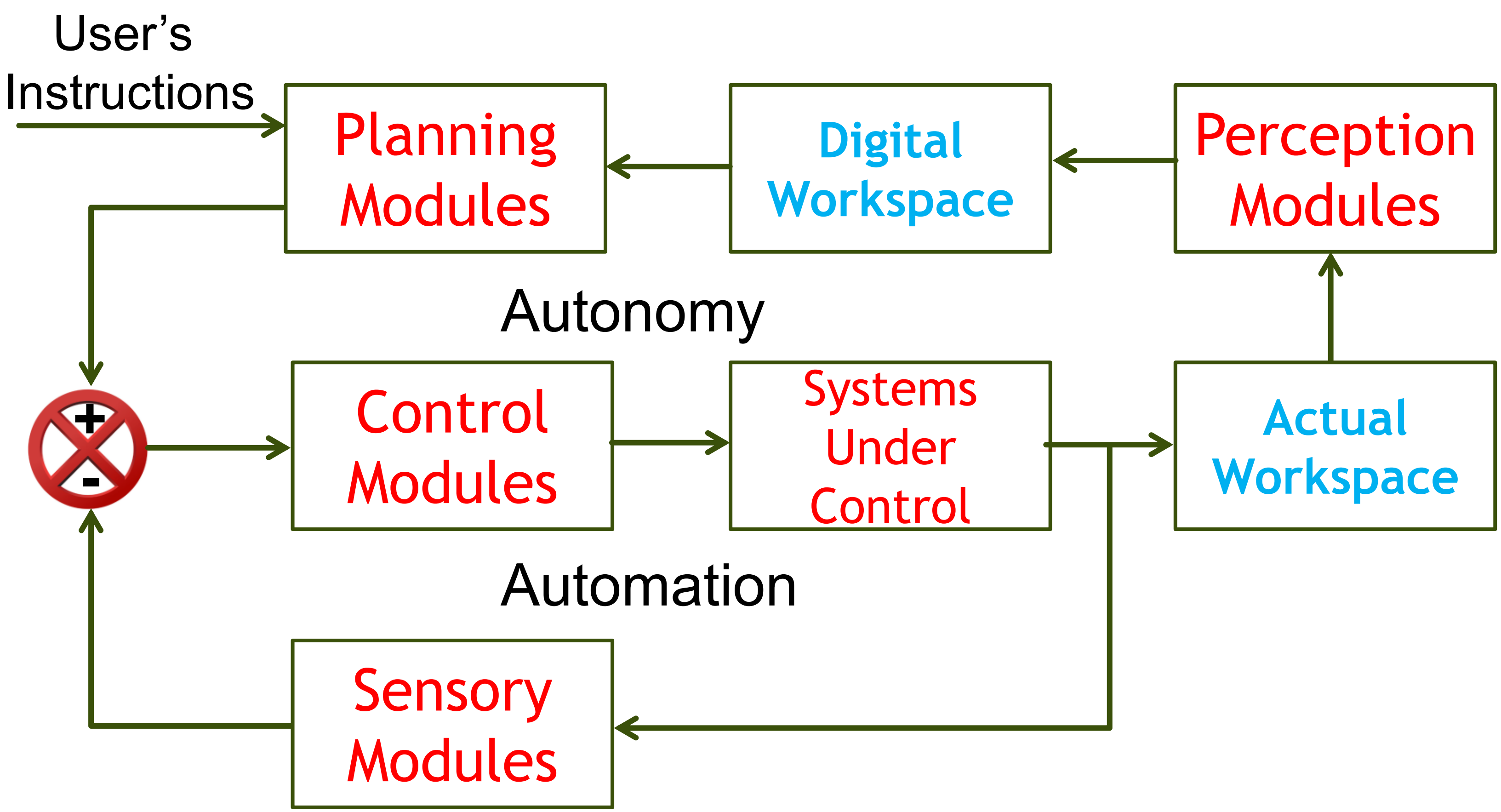
(Learning, Teaching) <o> (Research, Innovation) <o> (Leadership, Service)

Remember your mission as MAE graduates …

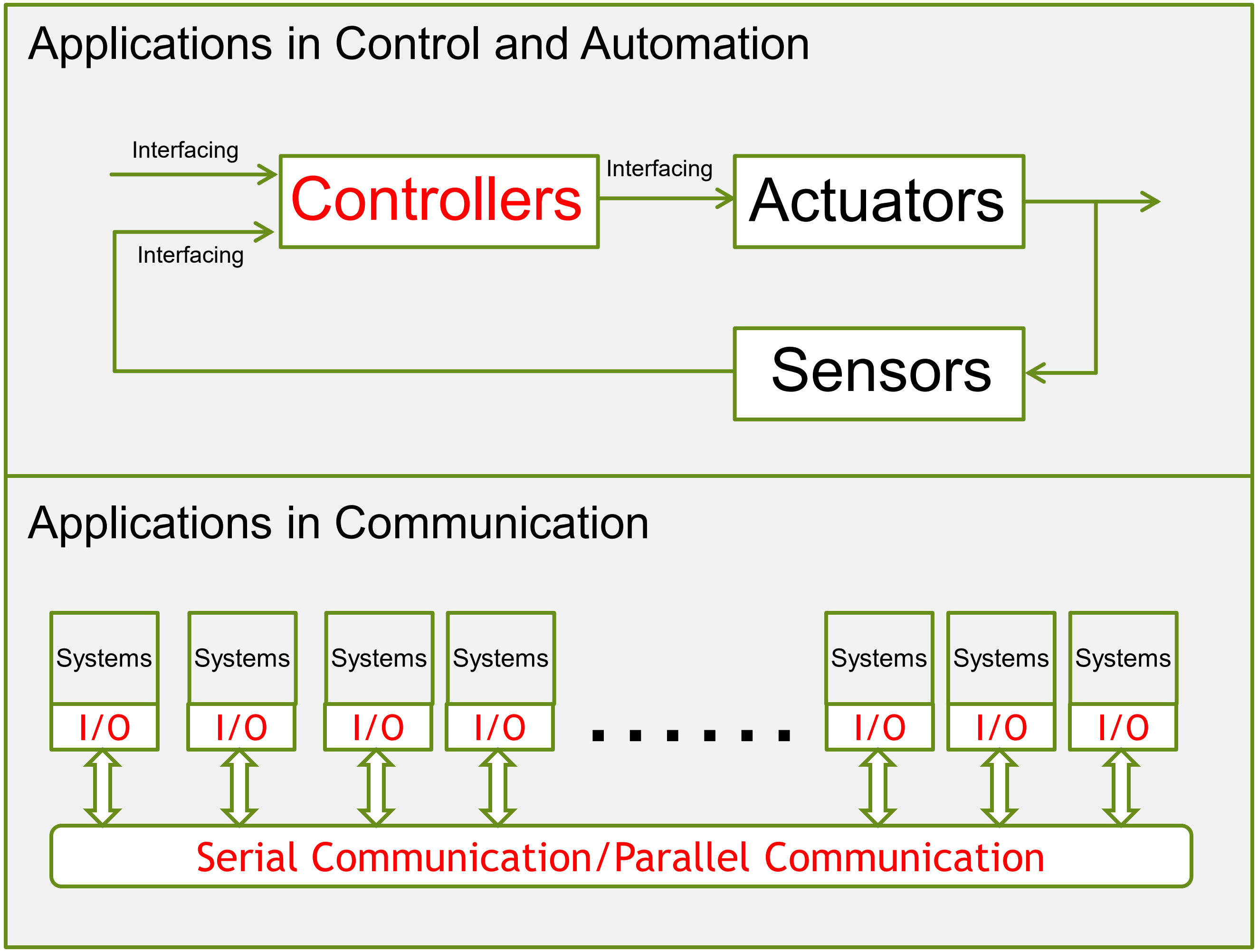
- You are here to grow your knowledge and skills so as to be able to design machines with controllable behaviors and hopefully in some intelligent ways.

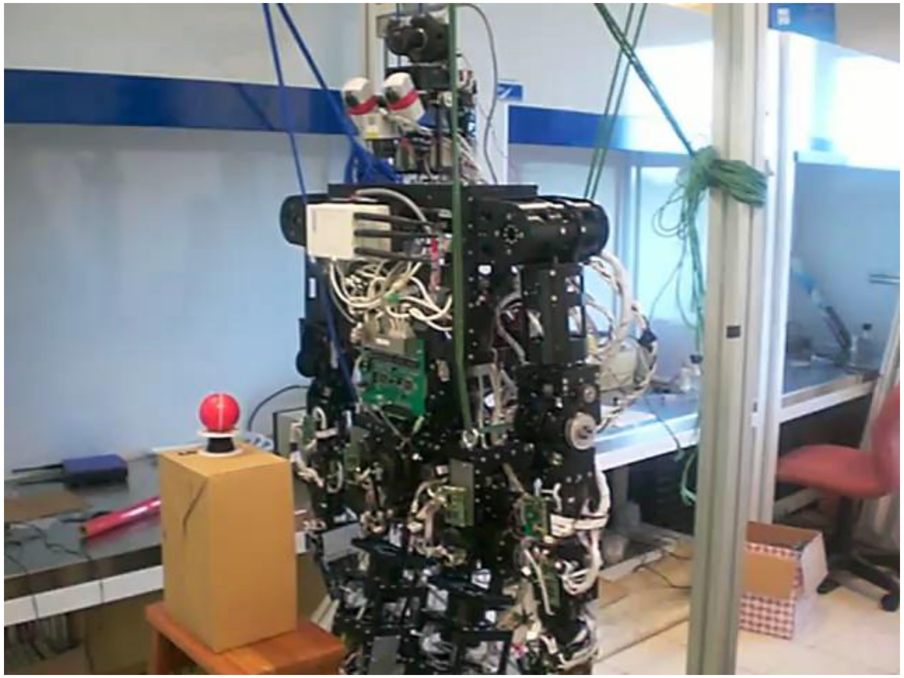
How to fulfill your mission?

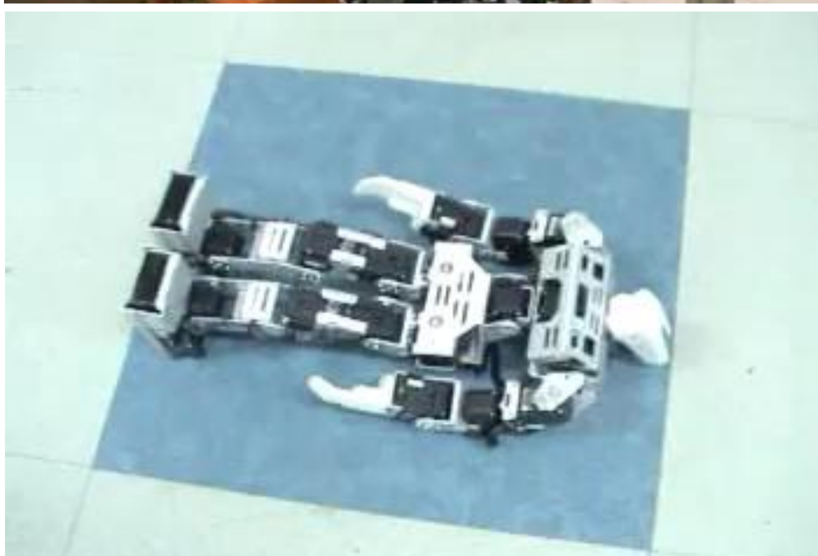
- To apply learnt knowledge and skills into the implementation of the following universal blueprint underlying all the intelligent machines or systems.

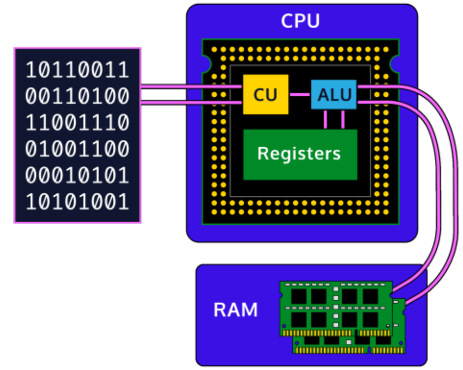


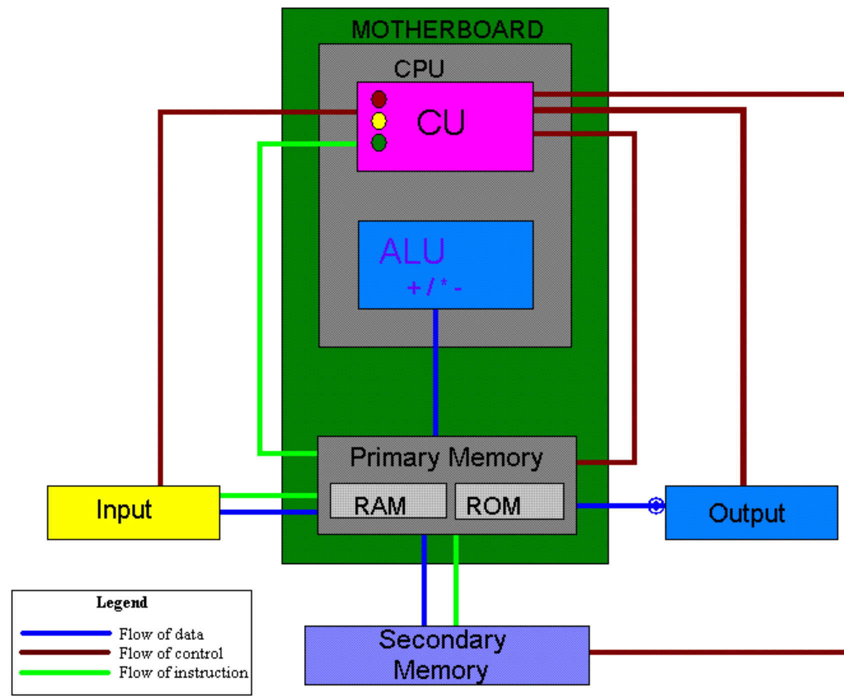
Why to study?


Applications in Control and Automation
Interfacing
Interfacing
Interfacing
Controllers
Actuators
Sensors
Applications in Communication
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Systems
I/O
Serial Communication/Parallel Communication









Block diagram of Computer with sub-units of CPU

Created by: MUHAMMAD KAMRAN KHAN

What to learn?

- Programming
- Interfacing

Machine Brain

1. ARM's Architecture
2. ARM's Memories
3. ARM's Data Representation
4. ARM's Programming
5. ARM's Data Input/Output
6. ARM's Data Processing

Microprocessor Systems

Input

1. Digital Device Interface
2. Analogue Device Interface
3. Asynchronous Communication
4. Synchronous Communication
5. Digital Motion Sensor Interface

Output

1. Digital Device Interface
2. Asynchronous Communication
3. Synchronous Communication
4. Digital Actuator Interface
5. Digital Motor Interface

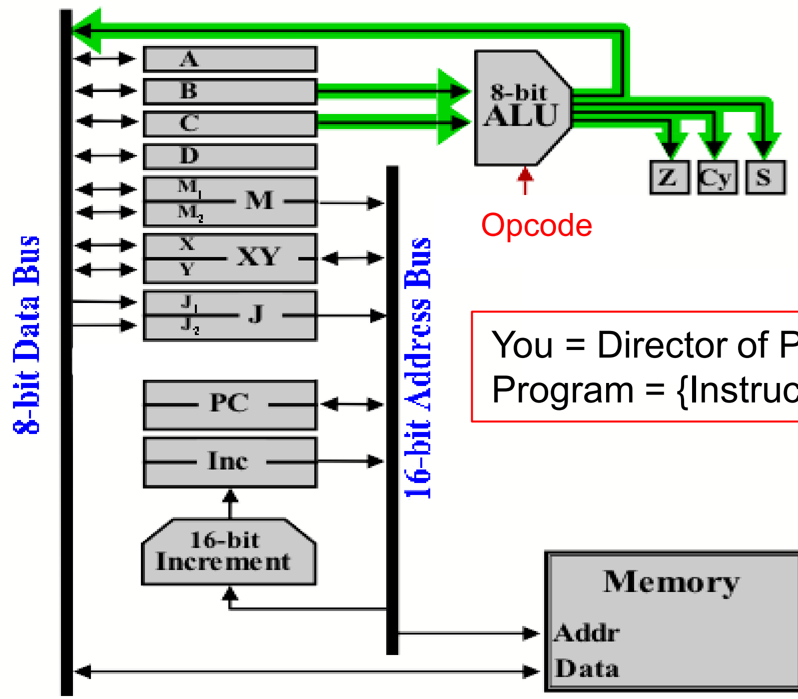
What is your role?

- Data = {Values, Symbols, Addresses, Instructions}
- Instructions = {Op Code + Addresses + Value/Symbol}

Applications/OS	
Languages	C++, Java
ISA	EOR r3,r2,r1 BEQ Table
Microarchitecture	
Gates	
Transistors	

Algorithms or Solutions

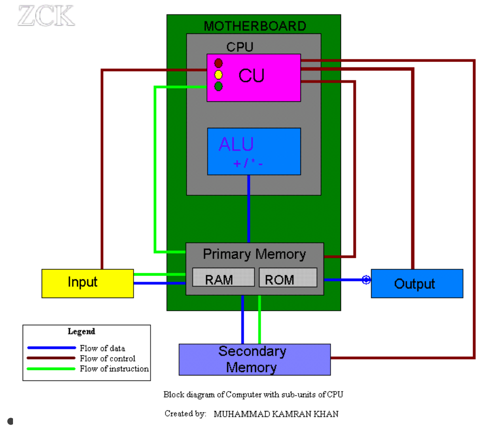
Problems to be solved



You = Director of Program
Program = {Instructions}

Memory = {Address + Data/Instruction}

How to learn?



- To understand data flows inside a microcontroller.
- To translate your solutions into data flows.
- To pay attention to <memory address> and <memory data>.
  - Memory Address: Address Label/Name and Address Value.
  - Memory Data: Data Label/Name and Data Value.

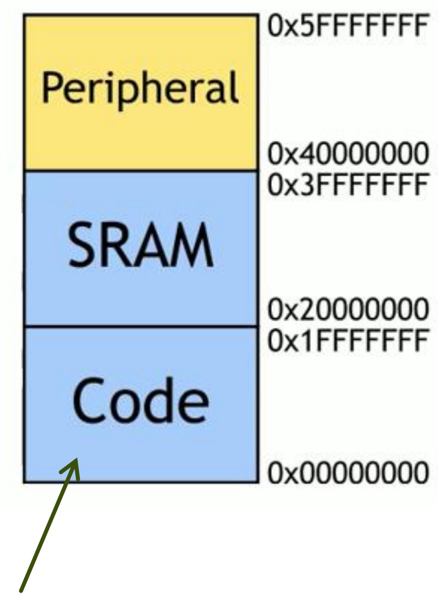
Memory Content

Data	Memory Address

- To pay attention to <memory address> and <memory code>.
  - Memory Address: Address Label/Name and Address Value.
  - Memory Code: Code Label/Name and Code Value.

Memory Content

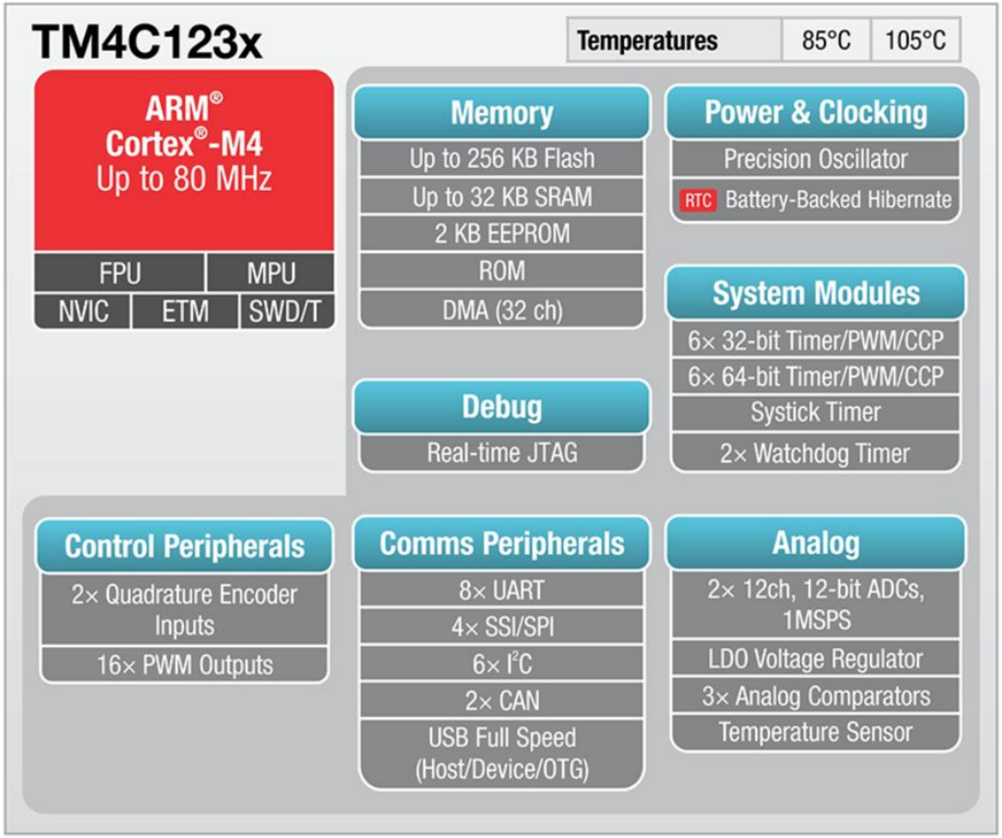
Instruction	Memory Address



Series of Instructions

Example of Using I/O Modules

- Configure Control Registers
- Clear/Monitor Status Registers
- Read/Write Data Registers
- Instructions:
  - MOV <address of destination>, <source of value>
  - LDR <address of destination>, <source of value>
  - STR <source of value>, <address of destination>

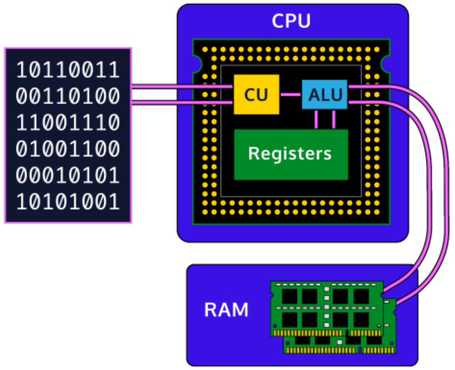



```
MOV   R0, #0x11
MOV   R1, #2560
MVN   R2, #4
MOVW  R3, #0xC0DE
MOVT  R3, #0xFEED
MOV   R4, R1
```

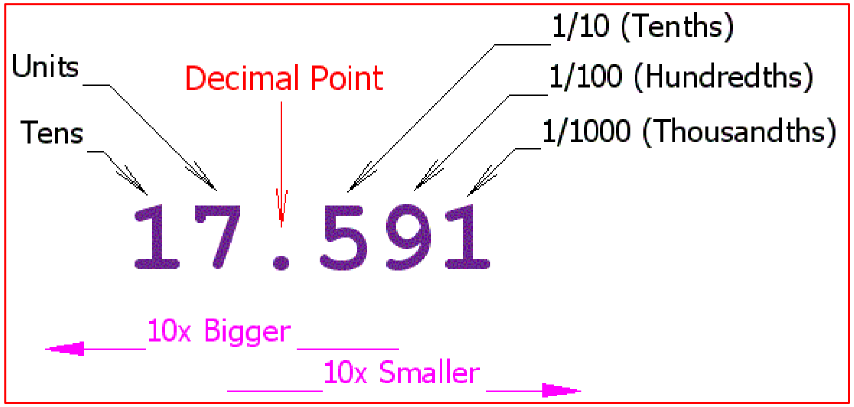
Word = 2 Bytes

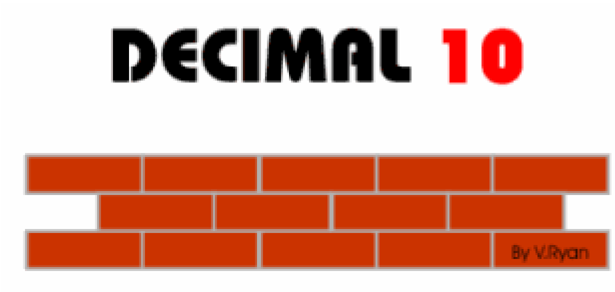
Outline

- How to store data?
- How to store instructions?



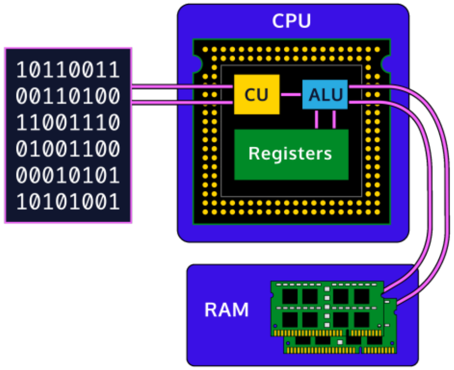
- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- Representation of Floating-point Numbers



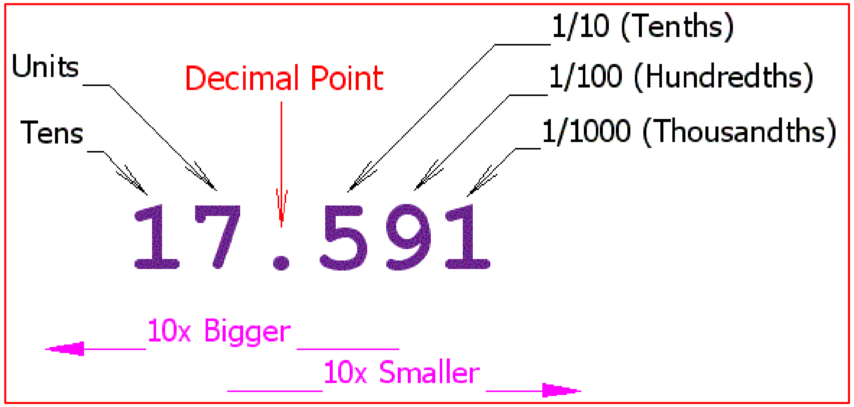


Outline

- How to store data?
- How to store instructions?



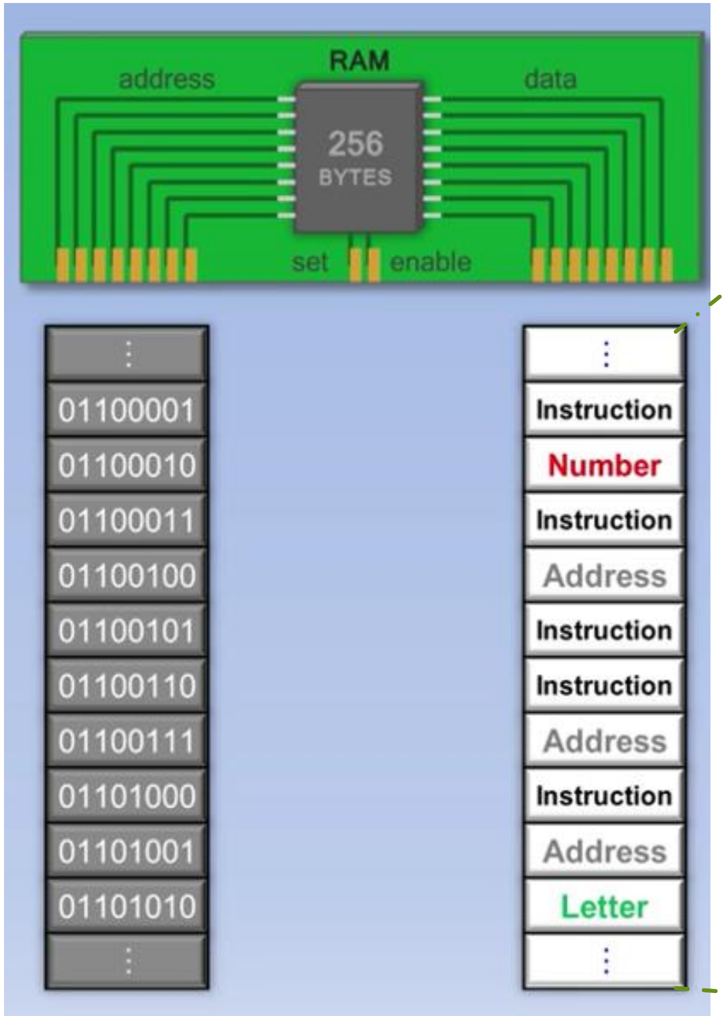
- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- Representation of Floating-point Numbers

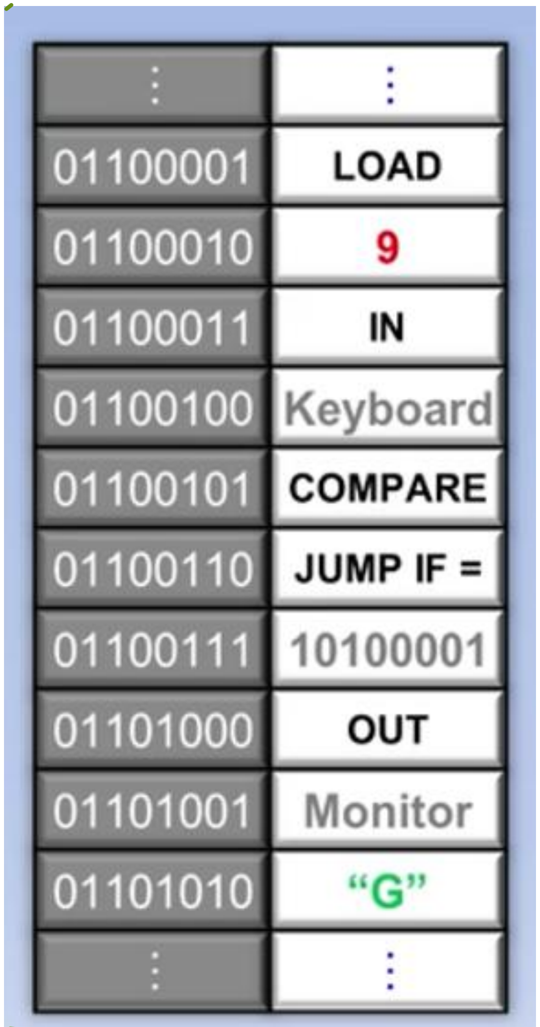


History of Binary Number System …

- Gottfried Wilhelm Leibniz (1646-1716) is the self-proclaimed inventor of the binary system and is considered as such by most historians of mathematics and/or mathematicians. However, systems related to binary numbers have appeared earlier in multiple cultures including ancient Egypt, China, and India.
- Modern methods of writing decimals were invented less than 500 years ago. However, the use of decimals in various forms can be traced back thousands of years. The Babylonians used a number system based on 60 and extended it to deal with numbers less than 1. Some use of decimals was also made in ancient China, medieval Arabia and in Renaissance Europe.
- Who invented hexadecimal notation? Swedish American engineer John Williams Nystrom developed the hexadecimal notation system in 1859.
- The eight digits of the octal system are 0, 1, 2, 3, 4, 5, and 7. The octal system was first discovered in 1716 by Emanuel Swedenborg, although linguists speculate that it was discovered by the Proto-Indo-Europeans, a group of Neolithic people ancestors to the Indo-European people of the Bronze Age.

Example of Using Binary Numbers …





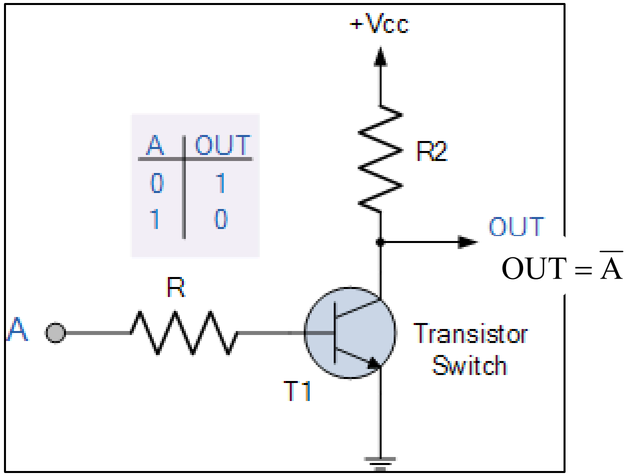
Digital computers are binary number systems

Binary Values

- 1 (Logic High)
- 0 (Logic Low)

Implementation

- +5 V (Logic High)
- 0 V (Logic Low)



Format of Binary Numbers

- A number consists of a series of digits.
- A digit has its value and its position in the series.
- For example, 11 1001 1100 0111

MSB LSB

Scale

2^{Position}

One digit in the left is two times the digit in the right.

Format of Binary Numbers (continued)

One digit in the left is two times the digit in the right.

- Hence, we can represent binary numbers in the following way:

$$V = D_{n-1}D_{n-2}{}^{\circ\circ\circ}D_1D_0$$

- in which:

$$D_i \in [0,1], i = 0,1,2, \ldots$$ (or i = -1, -2, …)

$$V = D_{n-1}2^{n-1} + D_{n-2}2^{n-2} + \cdots + D_1 2^1 + D_0 2^0$$

Example (i = 0, 1, 2, …)

One digit in the left is two times the digit in the right.

Binary Numbers	Corresponding Decimal Values
0 0 0 0 1 1 1 1	8 + 4 + 2 + 1 = 15
1 1 1 1 0 0 0 0	128 + 64 + 32 + 16 = 240
1 1 0 0 1 1 0 0	128 + 64 + 8 + 4 = 204
0 0 1 1 0 0 1 1	32 + 16 + 2 + 1 = 51

Example (i = -1, -2, …)

One digit in the left is two times the digit in the right.

Binary Numbers	Corresponding Decimal Values
0. 0 0 0 0 1 1 1 1	0.0313 + 0.0156 + 0.0078 + 0.0039
0. 1 1 1 1 0 0 0 0	0.5 + 0.25 + 0.125 + 0.0625
0. 1 1 0 0 1 1 0 0	0.5 + 0.25 + 0.0313 + 0.0156
0. 0 0 1 1 0 0 1 1	0.125 + 0.0625 + 0.0078 + 0.0039

Binary Addition

- Input: $$V_1 = A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \cdots + A_1 2^1 + A_0 2^0$$

$$V_2 = B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \cdots + B_1 2^1 + B_0 2^0$$

- Output:

$$V_3 = C_{n-1}2^{n-1} + C_{n-2}2^{n-2} + \cdots + C_1 2^1 + C_0 2^0$$

- Rules:
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 0 with a carry of 1

	1	1	1				(carry)
		1	0	1_2	=	5_{10}	(augend)
+			1	1_2	=	3_{10}	(addend)
	1	$^{1}0$	$^{1}0$	$^{1}0_2$	=	8_{10}	

Example of Binary Addition

One digit in the left is two times the digit in the right.

- Augend:

  1 1 1 1 0 0 0 0

- Addend:

  1 1 0 0 1 1 0 0

- Result:

	1 1 0 0 0 0 0 0 0	Carry
	1 1 1 1 0 0 0 0	Augend
	+ 1 1 0 0 1 1 0 0	Addend
carry	1 1 0 1 1 1 1 0 0	

Binary Subtraction

One digit in the left is two times the digit in the right.

- Input: $$V_1 = A_{n-1}2^{n-1} + A_{n-2}2^{n-2} + \cdots + A_1 2^1 + A_0 2^0$$

$$V_2 = B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \cdots + B_1 2^1 + B_0 2^0$$

- Output:

$$V_3 = C_{n-1}2^{n-1} + C_{n-2}2^{n-2} + \cdots + C_1 2^1 + C_0 2^0$$

- Rules:
  - 0 - 0 = 0
  - 0 - 1 = 1 with a borrow of 1
  - 1 - 0 = 1
  - 1 - 1 = 0

	1	$^{1}0$	1_2	=	5_{10} **(minuend)**
-		1	1_2	=	3_{10} **(subtrahend)**
-	1				**(borrow)**
	0	1	0_2	=	2_{10}

Example of Binary Subtraction

One digit in the left is two times the digit in the right.

- Minuend:

  1 1 1 1 0 0 0 0

- Subtrahend:

  1 1 0 0 1 1 0 0

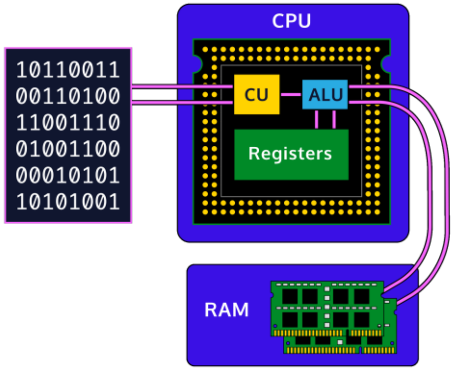
- Result:

0 0 0 0 1 1 0 0 0	Borrow
1 1 1 1 0 0 0 0	Minuend
- 1 1 0 0 1 1 0 0	Subtrahend
0 0 0 1 0 0 1 0 0	

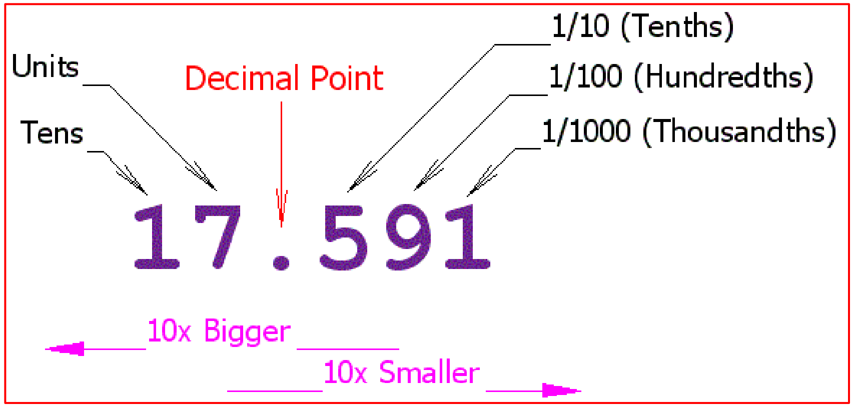
borrow

Outline

- How to store data?
- How to store instructions?



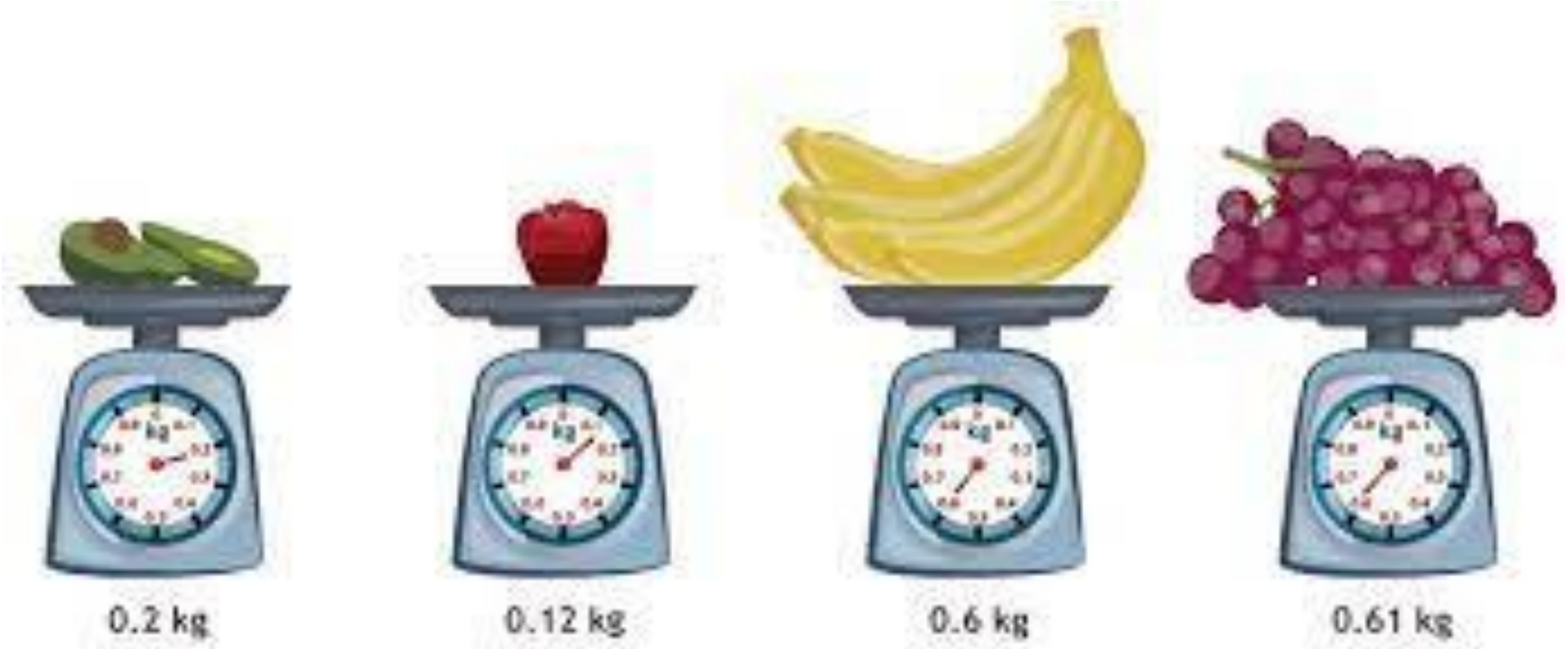
- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- Representation of Floating-point Numbers



History of Decimal Number System …

- Modern methods of writing decimals were invented less than 500 years ago. However, the use of decimals in various forms can be traced back thousands of years. The Babylonians used a number system based on 60 and extended it to deal with numbers less than 1. Some use of decimals was also made in ancient China, medieval Arabia and in Renaissance Europe.
- Gottfried Wilhelm Leibniz (1646-1716) is the self-proclaimed inventor of the binary system and is considered as such by most historians of mathematics and/or mathematicians. However, systems related to binary numbers have appeared earlier in multiple cultures including ancient Egypt, China, and India.
- Who invented hexadecimal notation? Swedish American engineer John Williams Nystrom developed the hexadecimal notation system in 1859.
- The eight digits of the octal system are 0, 1, 2, 3, 4, 5, and 7. The octal system was first discovered in 1716 by Emanuel Swedenborg, although linguists speculate that it was discovered by the Proto-Indo-Europeans, a group of Neolithic people ancestors to the Indo-European people of the Bronze Age.

Example of Using Decimal Numbers …



Format of Decimal Numbers

- A number consists of a series of digits.
- A digit has its value and its position in the series.
- For example, 1 2 3 4 0 0 0 0

Scale

10^{Position}

One digit in the left is ten times the digit in the right.

Format of Decimal Numbers (continued)

One digit in the left is ten times the digit in the right.

- Hence, we can represent decimal numbers in the following way:

$$V = D_{n-1}D_{n-2}{}^{\circ\circ\circ}D_1D_0$$

- in which:

$$D_i \in [0,1,2,3,4,5,6,7,8,9], i = 0,1,2, \dots$$ (or i = -1, -2, …)

$$V = D_{n-1}10^{n-1} + D_{n-2}10^{n-2} + \cdots + D_1 10^1 + D_0 10^0$$

Example (i = 0, 1, 2, …)

One digit in the left is ten times the digit in the right.

Decimal Numbers	Corresponding Decimal Values
0 0 0 0 2 2 3 4	2000+200+30+4
6 7 8 9 0 0 0 0	60000000+7000000+800000+90000
3 4 0 0 8 9 0 0	30000000+4000000+8000+900
0 0 8 9 0 0 3 4	800000+90000+30+4

Example (i = -1, -2, …)

One digit in the left is ten times the digit in the right.

Decimal Numbers	Corresponding Decimal Values
0. 0 0 0 0 2 2 3 4	0.00002+0.000002+0.0000003+0.00000004
0. 6 7 8 9 0 0 0 0	0.6+0.07+0.008+0.0009
0. 3 4 0 0 8 9 0 0	0.3+0.04+0.00008+0.000009
0. 0 0 8 9 0 0 3 4	0.008+0.0009+0.0000003+0.00000004

Decimal Addition

- Input: $$V_1 = A_{n-1}10^{n-1} + A_{n-2}10^{n-2} + \cdots + A_1 10^1 + A_0 10^0$$

$$V_2 = B_{n-1}10^{n-1} + B_{n-2}10^{n-2} + \cdots + B_1 10^1 + B_0 10^0$$

- Output:

$$V_3 = C_{n-1}10^{n-1} + C_{n-2}10^{n-2} + \cdots + C_1 10^1 + C_0 10^0$$

- Rules:
  - If Ai + Bi > 9, Ci = Ai + Bi - 10, with a carry of 1
  - Otherwise, Ci = Ai + Bi

$$\begin{array}{rl} \color{blue}{1\,1\;\;} & \color{blue}{\leftarrow \text{carry}} \\ 95_{10} & \\ +\quad 16_{10} & \\ \hline 111_{10} & \end{array}$$

Example of Decimal Addition

One digit in the left is ten times the digit in the right.

- Augend:

  6 7 8 9 0 0 0 0

- Addend:

  3 4 0 0 8 9 0 0

- Result:

1 1 0 0 0 0 0 0 0	Carry
6 7 8 9 0 0 0 0	Augend
+ 3 4 0 0 8 9 0 0	Addend
1 0 1 8 9 8 9 0 0	

carry

Decimal Subtraction

One digit in the left is ten times the digit in the right.

- Input:

$$V_1 = A_{n-1}10^{n-1} + A_{n-2}10^{n-2} + \cdots + A_1 10^1 + A_0 10^0$$

$$V_2 = B_{n-1}10^{n-1} + B_{n-2}10^{n-2} + \cdots + B_1 10^1 + B_0 10^0$$

- Output:

$$V_3 = C_{n-1}10^{n-1} + C_{n-2}10^{n-2} + \cdots + C_1 10^1 + C_0 10^0$$

- Rules:
  - If Ai < Bi, Ci = 10 + Ai – Bi, with a borrow of 1
  - Otherwise, Ci = Ai – Bi

$$\begin{array}{rrr} & 9 & {}^{1}5_{10} \\ - & 1 & 6_{10} \\ - & 1 & \\ \hline & 7 & 9_{10} \end{array}$$

$$\begin{array}{rcl} 95 = & 9\text{x}10^1 + 5\text{x}10^0 = & 9\text{x}10^1 + 15\text{x}10^0 \\ -16 = & -1\text{x}10^1 + -6\text{x}10^0 = & -1\text{x}10^1 + -6\text{x}10^0 \\ & \text{borrow} = & -1\text{x}10^1 \\ \hline & & 7\text{x}10^1 + 9\text{x}10^0 \end{array}$$

Example of Decimal Subtraction

One digit in the left is ten times the digit in the right.

- Minuend:

6 7 8 9 0 0 0 0

- Subtrahend:

3 4 0 0 8 9 0 0

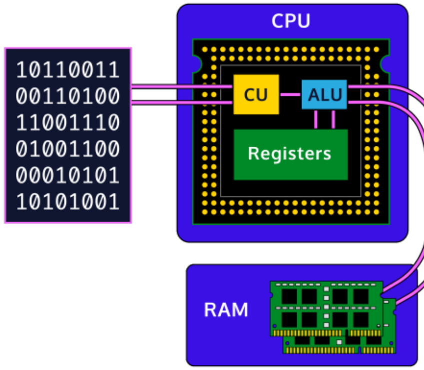
- Result:

0 0 0 0 1 1 0 0 0	Borrow
6 7 8 9 0 0 0 0	Minuend
- 3 4 0 0 8 9 0 0	Subtrahend
0 3 3 8 8 1 1 0 0	

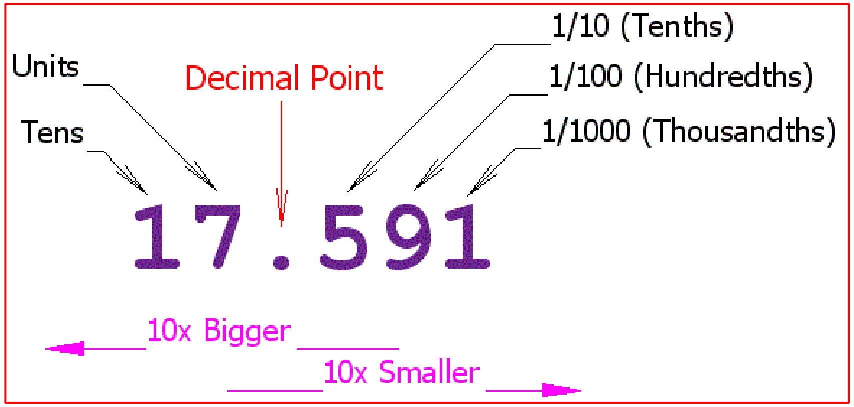
borrow

Outline

- How to store data?
- How to store instructions?



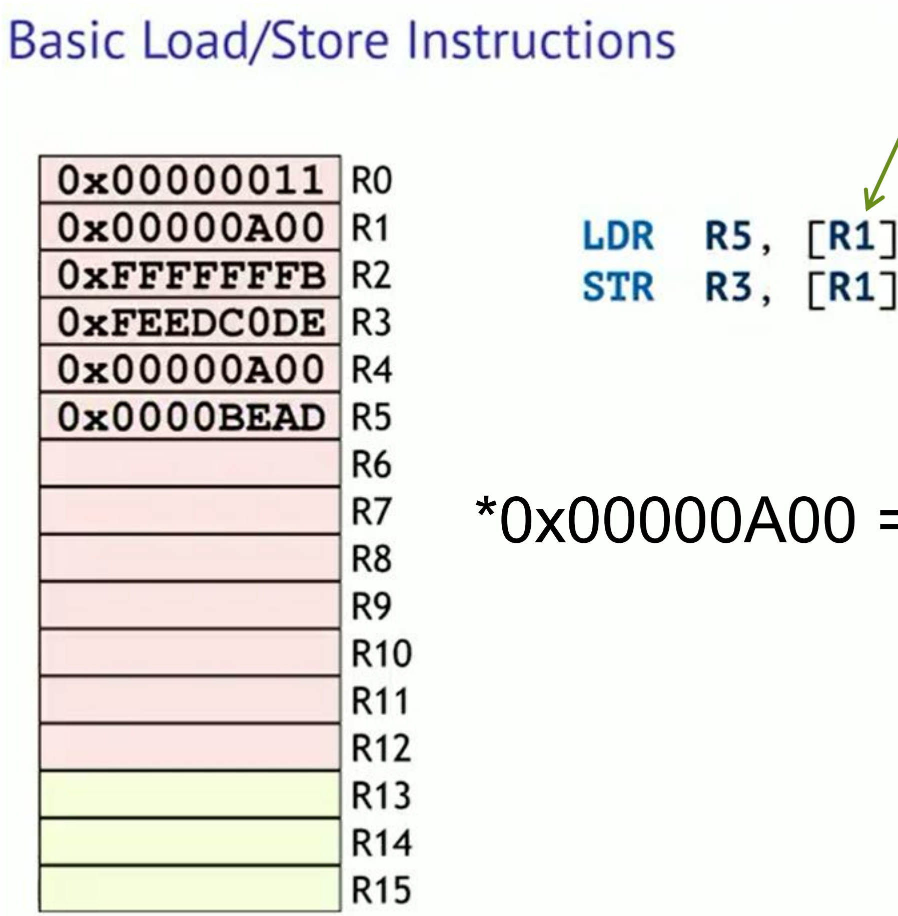
- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- Representation of Floating-point Numbers



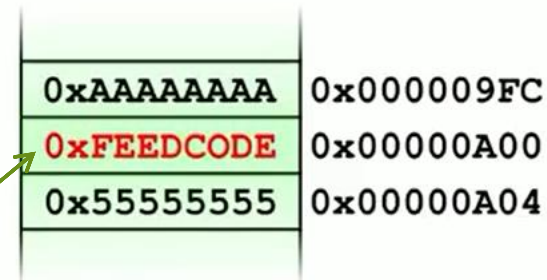
History of Hexadecimal Number System …

- Who invented hexadecimal notation? Swedish American engineer John Williams Nystrom developed the hexadecimal notation system in 1859.
- Modern methods of writing decimals were invented less than 500 years ago. However, the use of decimals in various forms can be traced back thousands of years. The Babylonians used a number system based on 60 and extended it to deal with numbers less than 1. Some use of decimals was also made in ancient China, medieval Arabia and in Renaissance Europe.
- Gottfried Wilhelm Leibniz (1646-1716) is the self-proclaimed inventor of the binary system and is considered as such by most historians of mathematics and/or mathematicians. However, systems related to binary numbers have appeared earlier in multiple cultures including ancient Egypt, China, and India.
- The eight digits of the octal system are 0, 1, 2, 3, 4, 5, and 7. The octal system was first discovered in 1716 by Emanuel Swedenborg, although linguists speculate that it was discovered by the Proto-Indo-Europeans, a group of Neolithic people ancestors to the Indo-European people of the Bronze Age.

Example of Using Hexadecimal Numbers …



1. Use the memory value as an address
2. Get the value from the address



*0x00000A00 = 0xFEEDC0DE

Address

Two-dimensional Space

1. Memory Data
2. Memory Address
3. Memory Label

Bit

Format of Hexadecimal Numbers

- A number consists of a series of digits.
- A digit has its value and its position in the series.
- For example, A B C D 5 6 7 8

Scale

16^{Position}

One digit in the left is sixteen times the digit in the right.

Format of Hexadecimal Numbers (continued)

One digit in the left is sixteen times the digit in the right.

- Hence, we can represent hexadecimal numbers in the following way:

$$V = D_{n-1}D_{n-2}{}^{\circ\circ\circ}D_1 D_0$$

- in which:

$$D_i \in [0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F], i = 0,1,2, \ldots$$

(or i = -1, -2, …)

$$V = D_{n-1}16^{n-1} + D_{n-2}16^{n-2} + \cdots + D_1 16^1 + D_0 16^0$$

Example (i = 0, 1, 2, …)

One digit in the left is sixteen times the digit in the right.

Hexadecimal Numbers	Corresponding Values
0 0 0 0 5 6 8 8	20480+1536+128+8
A B C D 0 0 0 0	2684400000+184549376+12582912+851968
C D 0 0 7 8 0 0	3221200000+218103808+28672+2048
0 0 7 8 0 0 C D	7340032+524288+192+13

Example (i = -1, -2, …)

One digit in the left is sixteen times the digit in the right.

Hexadecimal Numbers	Corresponding Decimal Values
0. 0 0 0 0 5 6 8 8	0.0000047684+0.00000035763+0.000000029802+0.0000000018626
0. A B C D 0 0 0 0	0.625+ 0.043+ 0.0029+0.00019836
0. C D 0 0 7 8 0 0	0.75+ 0.0508+0.0000066757+0.00000047684
0. 0 0 7 8 0 0 C D	0.0017+0.00012207+0.000000044703+0.0000000030268

Hexadecimal Addition

- Input: $V_1 = A_{n-1}16^{n-1} + A_{n-2}16^{n-2} + \cdots + A_1 16^1 + A_0 16^0$

$$V_2 = B_{n-1}16^{n-1} + B_{n-2}16^{n-2} + \cdots + B_1 16^1 + B_0 16^0$$

- Output:

$$V_3 = C_{n-1}16^{n-1} + C_{n-2}16^{n-2} + \cdots + C_1 16^1 + C_0 16^0$$

- Rules:
  - If Ai + Bi > F, Ci = Ai + Bi - 16, with a carry of 1
  - Otherwise, Ci = Ai + Bi

1 0 1 1 0	Carry
F 0 B A	Augend
+ E 9 A D	Addend
--------------	
1 D A 6 7	Sum

Carry

Example of Hexadecimal Addition

One digit in the left is sixteen times the digit in the right.

- Augend:

  A B C D 0 0 0 0

- Addend:

  0 0 7 8 0 0 C D

- Result:

0 0 1 1 0 0 0 0 0	Carry
A B C D 0 0 0 0	Augend
+ 0 0 7 8 0 0 C D	Addend
0 A C 4 5 0 0 C D	

carry

Hexadecimal Subtraction

One digit in the left is sixteen times the digit in the right.

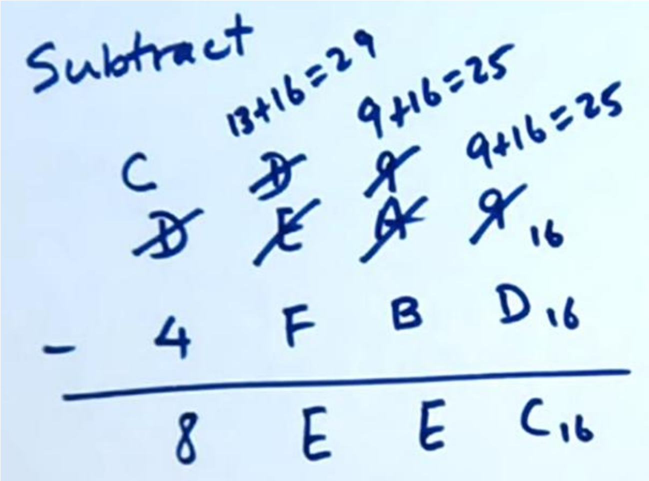
- Input: $V_1 = A_{n-1}16^{n-1} + A_{n-2}16^{n-2} + \cdots + A_1 16^1 + A_0 16^0$

$$V_2 = B_{n-1}16^{n-1} + B_{n-2}16^{n-2} + \cdots + B_1 16^1 + B_0 16^0$$

- Output:

$$V_3 = C_{n-1}16^{n-1} + C_{n-2}16^{n-2} + \cdots + C_1 16^1 + C_0 16^0$$

- Rules:
  - If Ai < Bi, Ci = 16 + Ai – Bi, with a borrow of 1
  - Otherwise, Ci = Ai – Bi



Example of Hexadecimal Subtraction

One digit in the left is sixteen times the digit in the right.

- Minuend:

  6 7 8 9 0 0 0 0

- Subtraction:

  3 4 0 0 8 9 0 0

- Result:

	0 0 0 0 1 1 0 0 0	Borrow
	6 7 8 9 0 0 0 0	Minuend
	- 3 4 0 0 8 9 0 0	Subtrahend
borrow	0 3 3 8 8 7 7 0 0	

Representation of Symbols (ASCII Code)

```
MOV    r0, #0x42; move a 'B' into register r0
```

Hexadecimal Codes

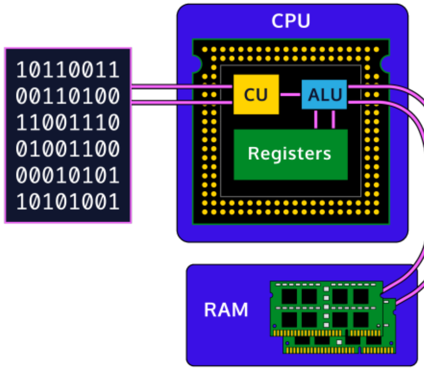
MOST SIGNIFICANT BITS

LEAST SIGNIFICANT BITS

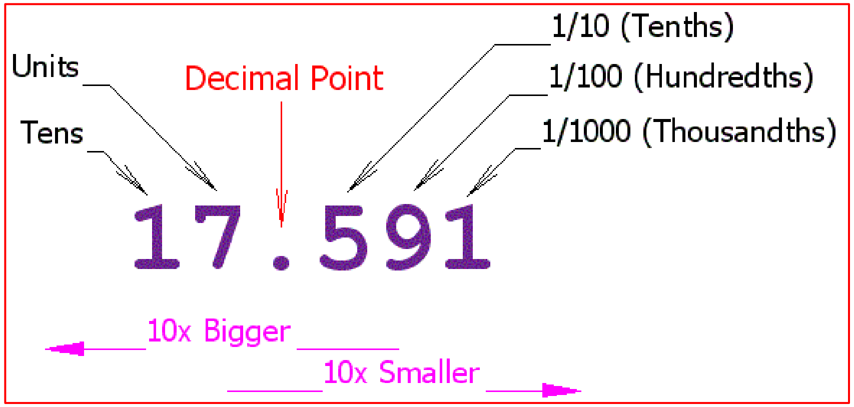
	0	1	2	3	4	5	6	7
0	Null	Data Link Escape	Space	0	@	P	`	p
1	Start of Heading	Device Control 1	!	1	A	Q	a	q
2	Start of Text	Device Control 2	“	2	B	R	b	r
3	End of Text	Device Control 3	#	3	C	S	c	s
4	End of Transmit	Device Control 4	$	4	D	T	d	t
5	Enquiry	Neg Acknowledge	%	5	E	U	e	u
6	Acknowledge	Synchronous Idle	&	6	F	V	f	v
7	Bell	End of Trans Block	’	7	G	W	g	w
8	Backspace	Cancel	(	8	H	X	h	x
9	Horizontal Tab	End of Medium	)	9	I	Y	i	y
A	Line Feed	Substitute	*	:	J	Z	j	z
B	Vertical Tab	Escape	+	;	K	[	k	{
C	Form Feed	File Separator	,	<	L	\	l	|
D	Carriage Return	Group Separator	-	=	M	]	m	}
E	Shift Out	Record Separator	.	>	N	^	n	~
F	Shift In	Unit Separator	/	?	O	_	o	Delete

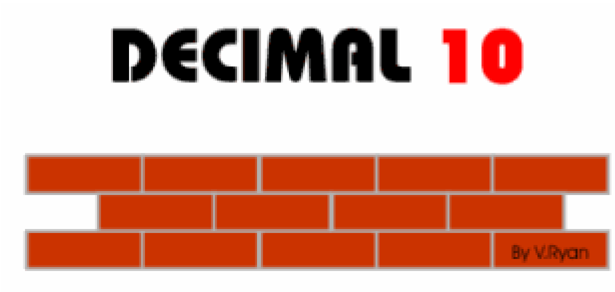
Outline

- How to store data?
- How to store instructions?



- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- Representation of Floating-point Numbers





Conversion Look-up Table

Hex	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

Hex	Binary	Decimal
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

From Binary to Hexadecimal

- Rules:
  - 1. One digit of hexadecimal corresponds to four digits of binary
  - 2. Group every four digits of binary together and replace it with the corresponding digit of hexadecimal.

- Example:
  - Binary Input: 1 1 1 1 1 1 1 0 1 1 0 0 1 0 0 0
  - Hexadecimal Output:
    - F = 1 1 1 1 E = 1 1 1 0 C = 1 1 0 0 8 = 1 0 0 0
    - Hexadecimal = 0xFEC8

From Hexadecimal to Binary

- Rules:
  - 1. One digit of hexadecimal corresponds to four digits of binary
  - 2. Expand every digit of hexadecimal into a series of four digits of binary and replace it with the corresponding four digit of binary.

- Example:
  - Hexadecimal Input: 0xFE89AB56
  - Binary Output:
    - F = 1 1 1 1 E = 1 1 1 0 8 = 1 0 0 0 9 = 1 0 0 1
    - A = 1 0 1 0 B = 1 0 1 1 5 = 0 1 0 1 6 = 0 1 1 0
    - Binary = 1 1 1 1 1 1 1 0 1 0 0 0 1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0

From Decimal to Binary

- Decimal Input:

$$V_{decimal} = D_{n-1}10^{n-1} + D_{n-2}10^{n-2} + \cdots + D_1 10^1 + D_0 10^0$$

- Binary Output:

$$V_{binary} = B_{n-1}2^{n-1} + B_{n-2}2^{n-2} + \cdots + B_1 2^1 + B_0 2^0$$

- Process (integer part):
  - Divide the input by 2. The remainder is the first digit of binary output.
  - Divide the quotient by 2. The remainder is the second digit of binary output.
  - Continue the process until the quotient becomes zero.

(Do multiplication for fractional part)

Example of Decimal to Binary Conversion: Integer Part

- Input: 57 in decimal

1. 57 / 2 = 28, remainder = 1 (binary number will end with 1)
2. 28 / 2 = 14, remainder = 0
3. 14 / 2 = 7, remainder = 0
4. 7 / 2 = 3, remainder = 1
5. 3 / 2 = 1, remainder = 1
6. 1 / 2 = <u>0</u>, remainder = 1 (binary number will start with 1)

Therefore, collecting the remainders, $57_{10} = 111001_2$

Note order

- Output: 1 1 1 0 0 1 in binary

Example of Decimal to Binary Conversion: Fractional Part

- Input: 0.375 in decimal

$$
\begin{aligned}
0.375 \times 2 &= 0.75 = 0.75 \text{ with carry } = 0 \\
0.75 \times 2 &= 1.50 = 0.50 \text{ with carry } = 1 \\
0.50 \times 2 &= 1.00 = 0.00 \text{ with carry } = 1
\end{aligned}
$$

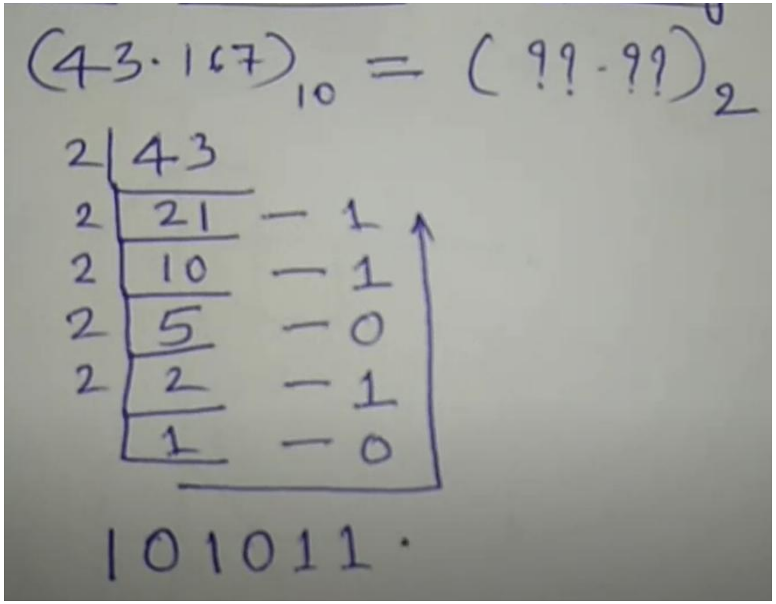
Therefore, 0.375 = 0.011

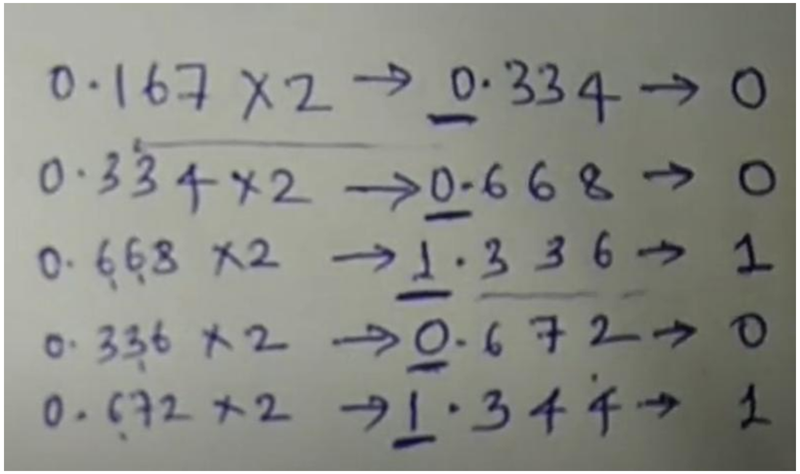
Note order

- Output: 0.011 in binary

One More Example

- Input: 43.167 in decimal





- Output: 101011.00101 in binary

From Decimal to Hexadecimal

- Decimal Input:

$$V_{decimal} = D_{n-1}10^{n-1} + D_{n-2}10^{n-2} + \cdots + D_1 10^1 + D_0 10^0$$

- Hexadecimal Output:

$$V_{hexadecimal} = H_{n-1}16^{n-1} + H_{n-2}16^{n-2} + \cdots + H_1 16^1 + H_0 16^0$$

- Process (integer part):
  - Divide the input by 16. The remainder is the first digit of hex output.
  - Divide the quotient by 16. The remainder is the second digit of hex output.
  - Continue the process until the quotient becomes zero.

(Do multiplication for fractional part)

Example of Decimal to Hexadecimal Conversion: Integer Part

- Input: 35243 in decimal

$$
\begin{array}{rlll}
35243 / 16 & = 2202, & \text{remainder} = 11 \rightarrow \text{B} & \text{(hex number will end with B)} \\
2202 / 16 & = 137, & \text{remainder} = 10 \rightarrow \text{A} & \\
137 / 16 & = 8, & \text{remainder} = 9 & \\
8 / 16 & = \underline{0}, & \text{remainder} = 8 & \text{(hex number will start with 8)}
\end{array}
$$

Therefore, collecting the remainders, $35243_{10} = 89AB_{16}$

Check: $(8 \times 16^3) + (9 \times 16^2) + (10 \times 16^1) + (11 \times 16^0) = 35243$ (dec)

- Output: 89AB in hexadecimal

Example of Decimal to Hexadecimal Conversion: Fractional Part

- Input: 0.375 in decimal

0.375x16 = 6.0 with carry = 6

- Output: 0.6 in hexadecimal

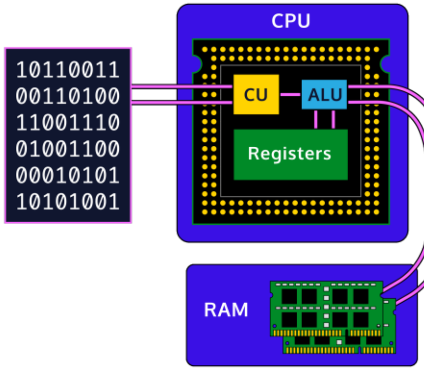
- Input: 0.987 in decimal

0.987x16 = 15.792 with carry = 15 → F
0.792x16 = 12.672 with carry = 12 → C
0.672x16 = 10.752 with carry = 10 → A
0.752x16 = 12.032 with carry = 12 → C

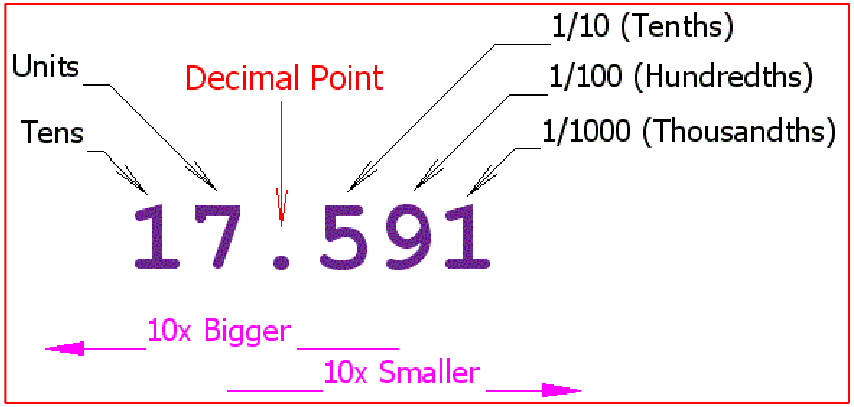
- Output: 0.FCAC in hexadecimal

Outline

- How to store data?
- How to store instructions?



- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- Representation of Floating-point Numbers



Example of Using Integers …

How many players? How many balls? …

Binary-coded Decimal of Integers

- Rules
  - Each digit of decimal is represented by four digits of binary.
  - 0 = 0000, 1 = 0001, 2 = 0010, 3 = 0011, 4 = 0100, 5 = 0101, 6 = 0110
  - 7 = 0111, 8 = 1000, 9 = 1001.
- Examples:
  - Decimal(9879) = Binary(1001 1000 0111 1001)
  - Decimal (12345) = Binary(0001 0010 0011 0100 0101)

Signed Magnitude Format of Integers

- Rules:
  - Allocate the most-significant-bit (MSB) to represent the signs.
  - Allocate the remaining bits to represent the values.

- Examples:
  - Binary with Four Bits: max = 0 111 (-> 7), min = 1 111 (-> -7)
  - Binary with Eight Bits: max = 0 1111111 (-> 127), min = 1 1111111 (-> -127)
  - Binary with Sixteen Bits:
    - max = 0 111 1111 1111 1111 (-> 32767), min = 1 111 1111 1111 1111 (-> -32767)
  - Binary with Thirty-two Bits:
    - max = 0 111 1111 1111 1111 1111 1111 1111 1111 (-> 2,147,483,647)
    - min = 1 111 1111 1111 1111 1111 1111 1111 1111 (-> - 2,147,483,647)

Could we further improve this representation?

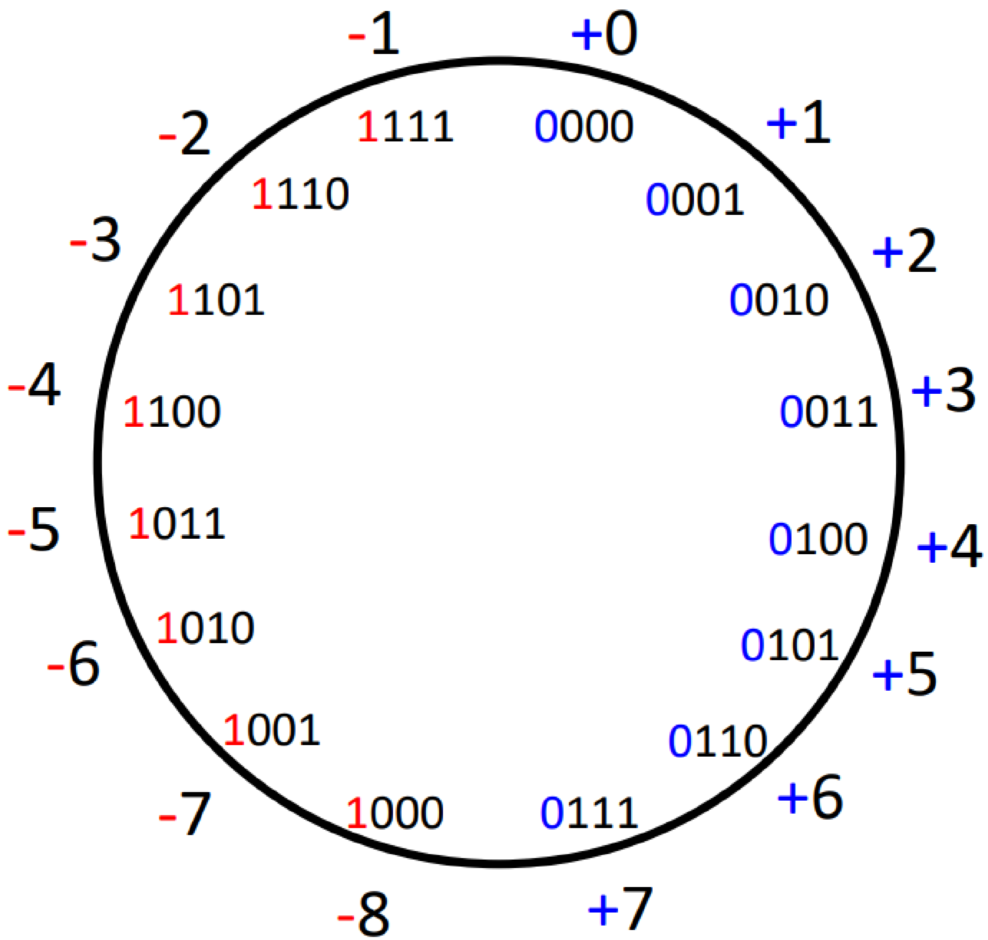
Two's Complement Format of Integers

- Rules:
  - For a binary number of N bits, the positive numbers are represented by the bits from 0 (LSB) to N-2 and bit N-1 (MSB) is zero.
  - For a binary number of N bits, the negative numbers are represented by the two's complements of their positive numbers.
  - A two's complement is equal to bit-wise complement plus 1.

- Examples:
  - 4 bit binary of integer 6 = 0 1 1 0
  - 4 bit binary of integer -6 = 1 0 0 1 + 1 = 1 0 1 0
  - 8 bit binary of integer 102 = 0 1 1 0 0 1 1 0
  - 8 bit binary of integer -102 = 1 0 0 1 1 0 0 1 + 1 = 1 0 0 1 1 0 1 0

An Illustration with 4-Bit Binary

- 8 positive numbers
- 8 negative numbers
- +8 = 1000
- -8 = 0111 + 1 = 1000
- +8 = -8
- So, we ignore +8



Why to use two's complement format?

Addition

- Addition not dependent on the signs of operand
- No need to compare magnitudes to determine sign of result

Subtraction

- Subtraction is treated as an addition
- Add the negative of the subtrahend to the minuend

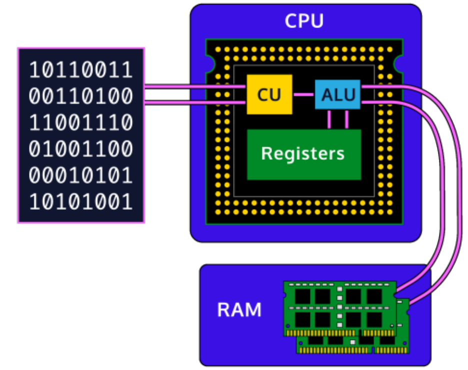
Simple implementation

- Adder unit
- Negation circuit unit

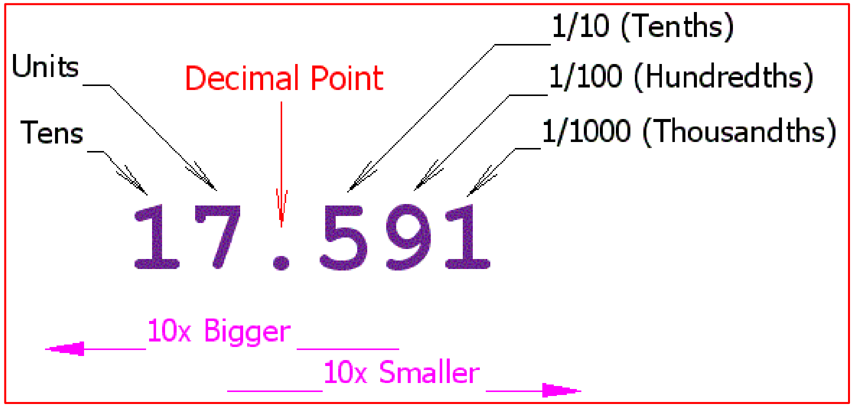
The simpler addition/subtraction scheme makes two's-complement the most common choice for integer number systems within digital systems

Outline

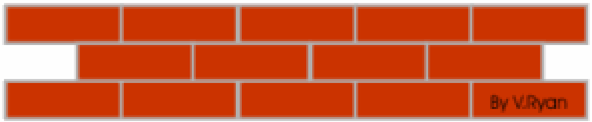
- How to store data?
- How to store instructions?



- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- **Representation of Floating-point Numbers**

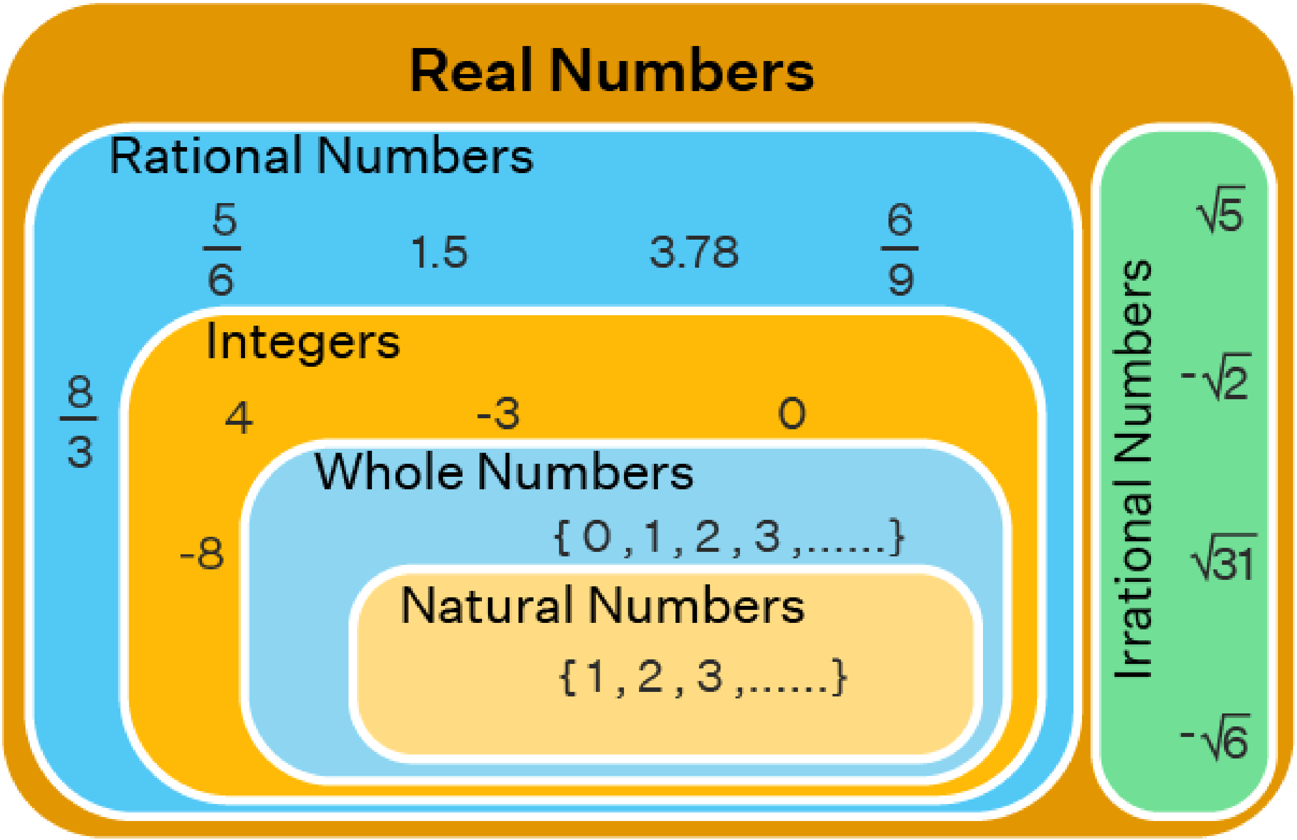


DECIMAL 10

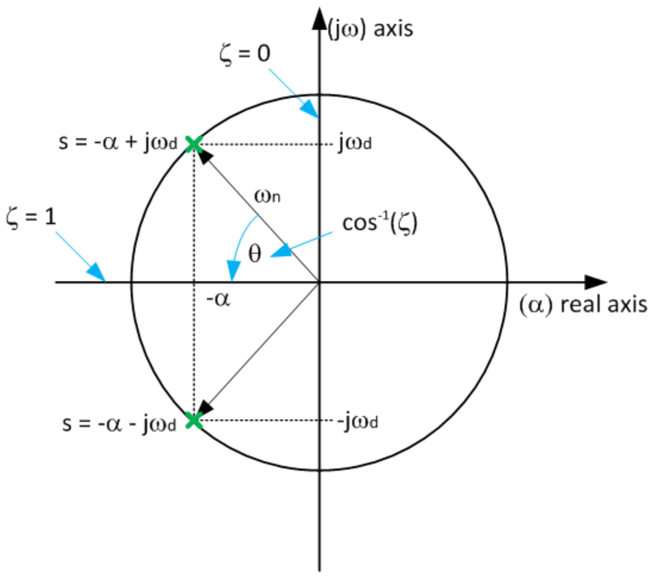


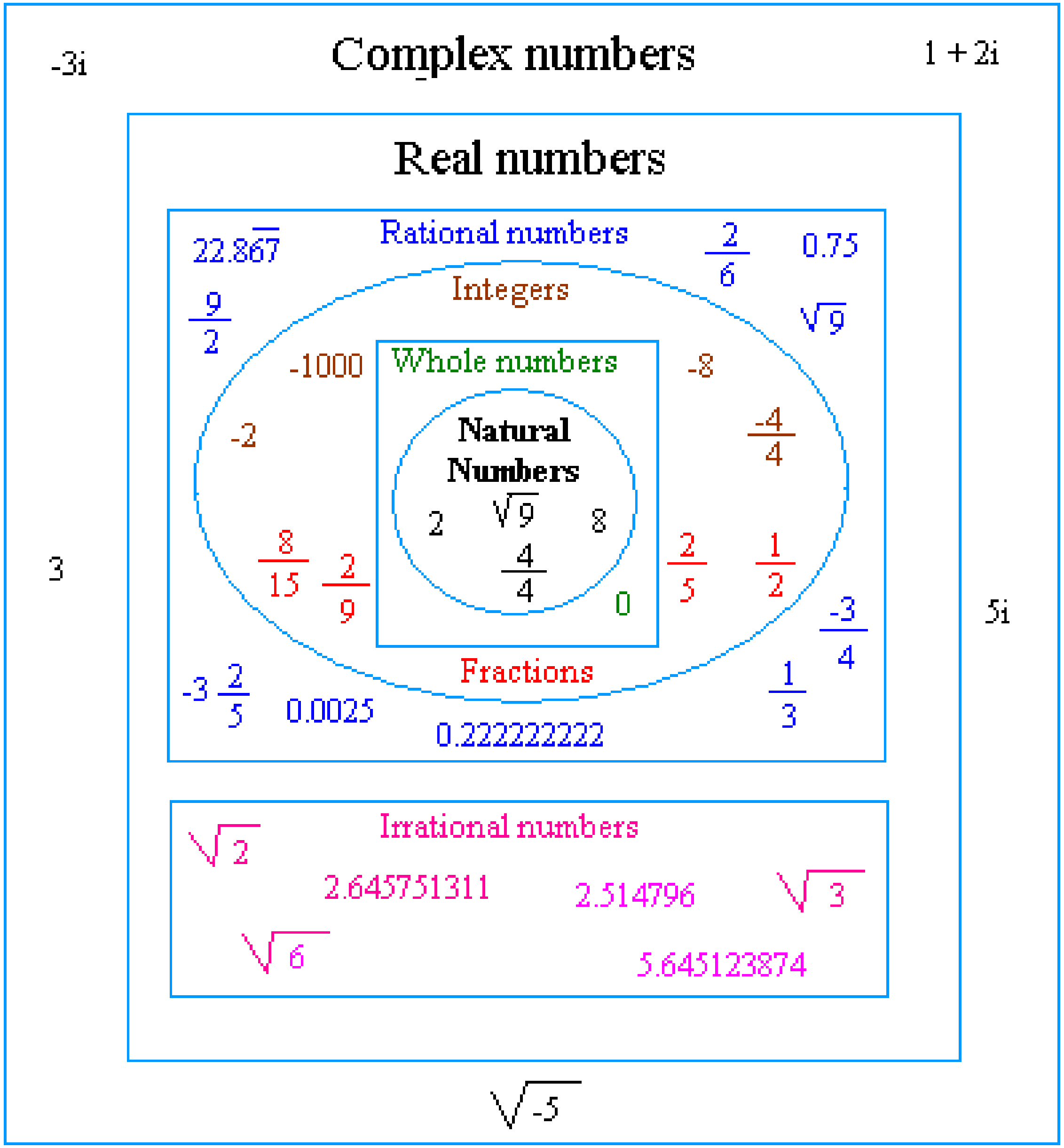
Example of Using Real Numbers …

Real Numbers Chart



Complex Numbers?





Binary-coded Decimal of Floating-point Numbers

- Rules
  - Each digit of decimal is represented by four digits of binary.
  - 0 = 0000, 1 = 0001, 2 = 0010, 3 = 0011, 4 = 0100, 5 = 0101, 6 = 0110
  - 7 = 0111, 8 = 1000, 9 = 1001.
- Examples:
  - Decimal(9879.34) = Binary(1001 1000 0111 1001).Binary(0011 0100)
  - Decimal (12345.5) = Binary(0001 0010 0011 0100 0101).Binary(0101)

Could we have a better format of representation?

Exponent-Mantissa Format of Floating-point Numbers

- Using the 32-bit ANSI/IEEE 754 Standard Format

(S) (1 bit)	Exponent (E) (8 bits)	Mantissa (M) or Fraction (23 bits)

Sign Bit (S)
0 denotes positive number, 1 denotes negative number

Exponent (E)
Represents both positive and negative exponents. A bias value of 127_{10} must be added to all exponents, regardless of sign.

Mantissa or Fraction (M)
Represents the leading significant bits in the number. It is stored in the normalized form.

Conversion of Decimals to Exponent-Mantissa Format

- Procedure:
  - Convert Decimals to Binary Format
  - Normalize Binary Format
  - Determine Exponent
  - Determine Mantissa
- Example:

a) $+2.75_{10} = 10.11_2$

b) In normalized form: $1.011_2 \times 2^1$

c) Express in 23-bit mantissa without leading 1_2 :
M = 011 0000 0000 0000 0000 0000

d) Express 8-bit exponent (E) with added bias:
$E = 1_{10} + 127_{10} = 128_{10} = 1000\ 0000_2$

e) Express sign bit (S):
S = 0 (positive number)

(S)	Exponent (E)	Mantissa (M) or Fraction
0	1000 0000	011 0000 0000 0000 0000 0000

Example of Representing -0.0625

a) $-0.0625_{10} = -0.0001_2$

b) In normalized form: $1.0_2 \times 2^{-4}$

c) Express in 23-bit mantissa without leading 1_2 :
$M = 00000000000000000000000$

d) Express 8-bit exponent (E) with added bias:
$E = -4_{10} + 127_{10} = 123_{10} = 01111011_2$

e) Express sign bit (S):
S = 1 (negative number)

(S)	Exponent (E)	Mantissa (M) or Fraction
1	01111011	00000000000000000000000

Example of Representing -417680

a) $-417680_{10} = -110\ 0101\ 1111\ 1001\ 0000_2$

b) In normalized form: $1.100101111110010000_2 \times 2^{18}$

c) Express in 23-bit mantissa without leading 1_2 :
M = 10010111111001000000000

d) Express 8-bit exponent (E) with added bias:
$E = 18_{10} + 127_{10} = 145_{10} = 1010001_2$

e) Express sign bit (S):
S = 1 (negative number)

(S)	Exponent (E)	Mantissa (M) or Fraction
1	10010001	10010111111001000000000

Special Cases: Zero and Infinity

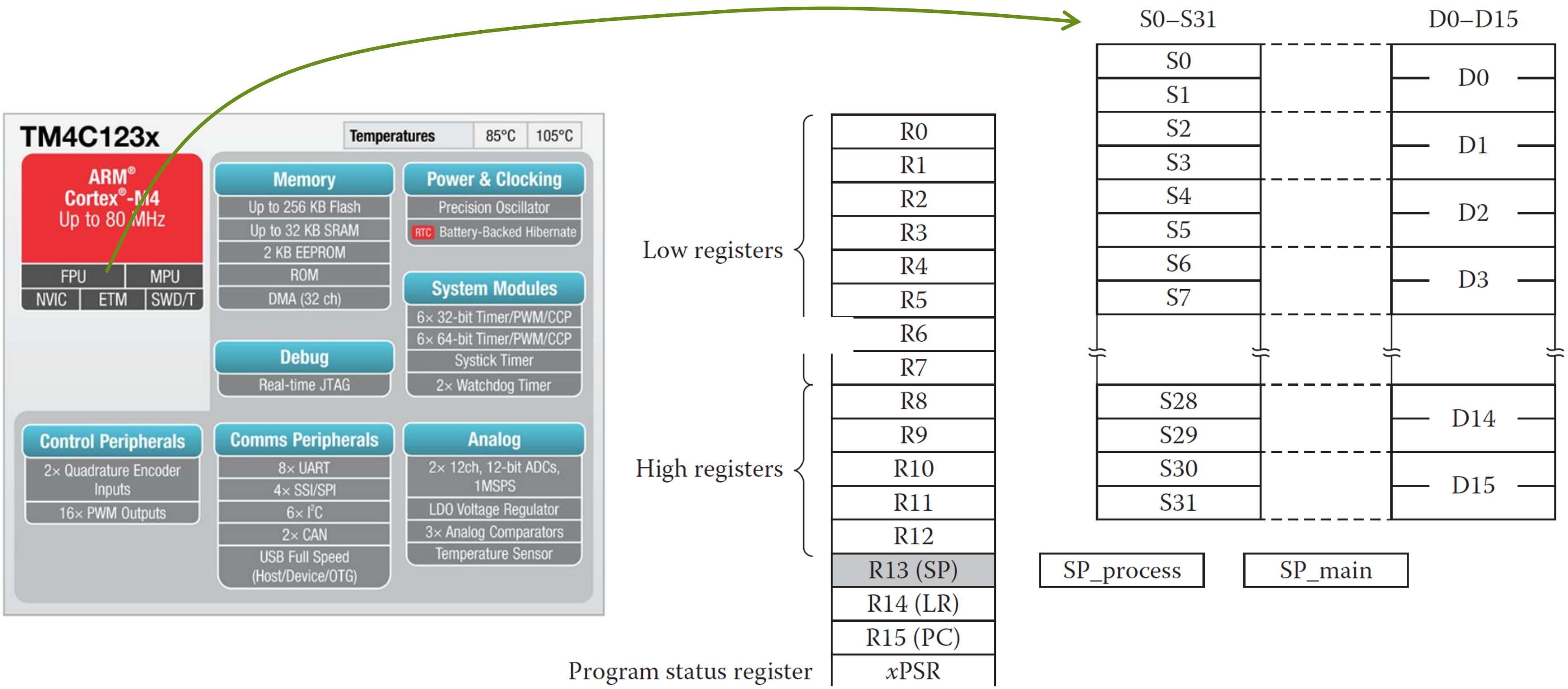
Represent +/- 0.0 in 32-bit ANSI/IEEE 754 Standard Format

(S)	Exponent (E)	Mantissa (M) or Fraction
0/1	00000000	00000000000000000000000

Represent +/- Infinity in 32-bit ANSI/IEEE 754 Standard Format

(S)	Exponent (E)	Mantissa (M) or Fraction
0/1	11111111	00000000000000000000000

Cortex M4 with Floating-point Registers



$$d[x] \Leftrightarrow \{s[(2x)+1],\ s[2x]\}$$

Load Data Into Floating-point Registers

- LDR or STR:

```
VLDR|VSTR{&lt;cond&gt;}.32 &lt;Sd&gt;, [&lt;Rn&gt;{, #+/ – &lt;imm&gt;}]
VLDR|VSTR{&lt;cond&gt;}.64 &lt;Dd&gt;, [&lt;Rn&gt;{, #+/ – &lt;imm&gt;}]
```

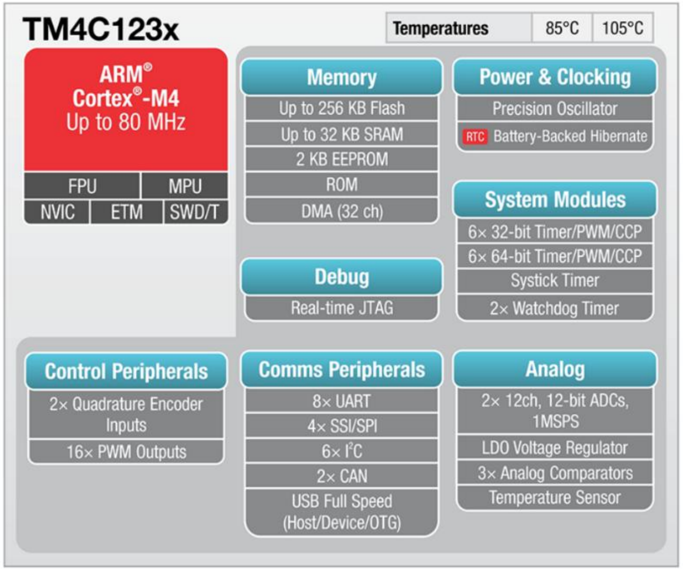
```
VLDR s5, [r6, #08]
```

- MOV:

```
VMOV{<cond>}.F32 <Sd>, <Rt>
VMOV{<cond>}.F32 <Rt>, <Sn>
```

```
VMOV s12, s13, r6, r11    ; s12 = r6, s13 = r11
```

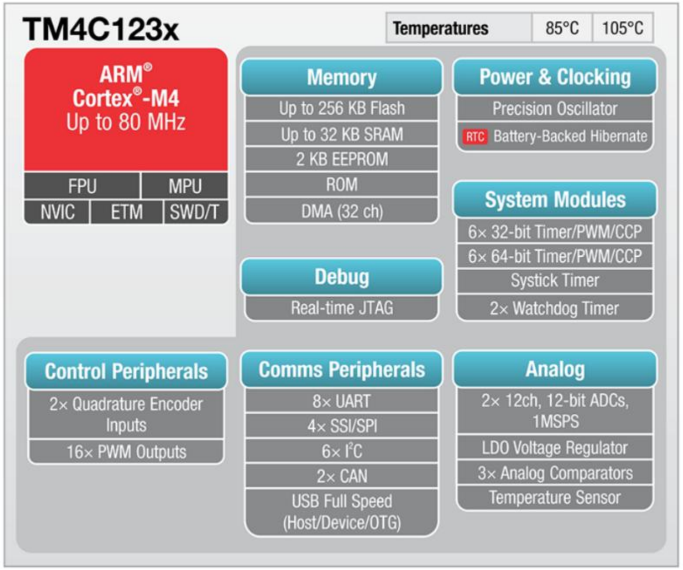
Example



; r0 = 0xE000ED88

```
LDR      r0, =0xE000ED88    ; Read-modify-write
LDR      r1, [r0]
ORR      r1, r1, #(0xF << 20) ; Enable CP10, CP11
STR      r1, [r0]
VMOV.F   s0, #0x3F800000    ; single-precision 1.0
VMOV.F   s1, s0
VADD.F   s2, s1, s0         ; 1.0+1.0=??
```

Example

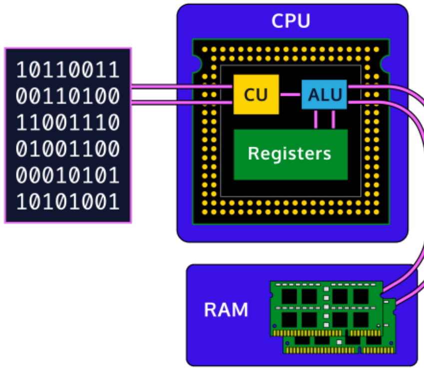


; r0 = 0xE000ED88

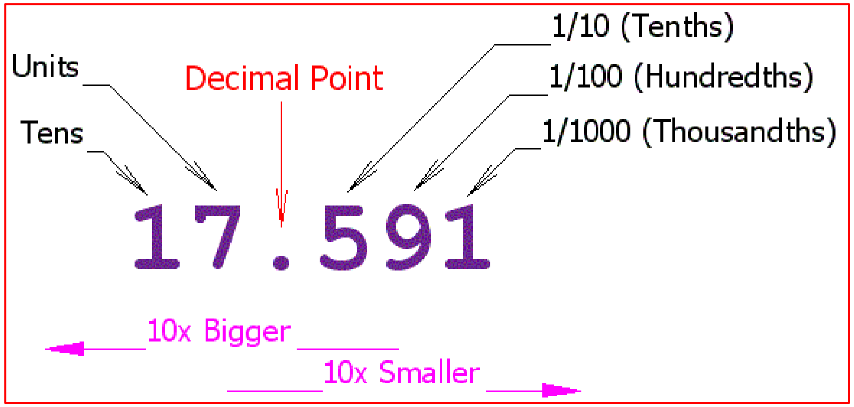
```
LDR      r0,  =0xE000ED88       ; Read-modify-write
LDR      r1,  [r0]
ORR      r1,  r1, #(0xF << 20)  ; Enable CP10, CP11
STR      r1,  [r0]
LDR      r3,  =0x3F800000       ; single precision 1.0
VMOV.F   s3,   r3               ; transfer contents from ARM to FPU
VLDR.F   s4,  =6.0221415e23     ; Avogadro's constant
VMOV.F   r4,  s4                ; transfer contents from FPU to ARM
```

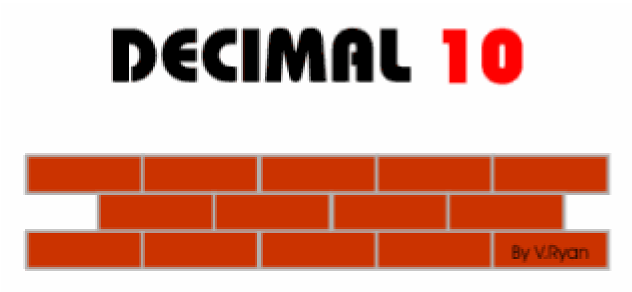
Summary

- How to store data?
- How to store instructions?



- Binary Numbers
- Decimal Numbers
- Hexadecimal Numbers
- Conversion of Numbers
- Representation of Integers
- Representation of Floating-point Numbers





“Ask not what your country can do for you – ask what you can do for your country,” - John F. Kennedy

“Do not think that you are needy – think that you are needed in the world”, - Manis Friedman

“Study will make you knowledgeable, resourceful, and hence more needed”, - Xie Ming

Thank You for Listening!