Lecture 2
part 2 : NUMBERING SYSTEM

prepared by Eng.shada Abduladeem

1's Complement Subtraction

- Using 1's complement numbers, subtracting numbers is also easy.
- For example, suppose we wish to subtract $+(00000001)_2$ from $+(00001100)_2$.
- Let's compute $(12)_{10} - (1)_{10}$.
  - $(12)_{10} = +(00001100)_2 = 00001100_2$ in 1's comp.
  - $-(1)_{10} = -(00000001)_2 = 11111110_2$ in 1's comp.

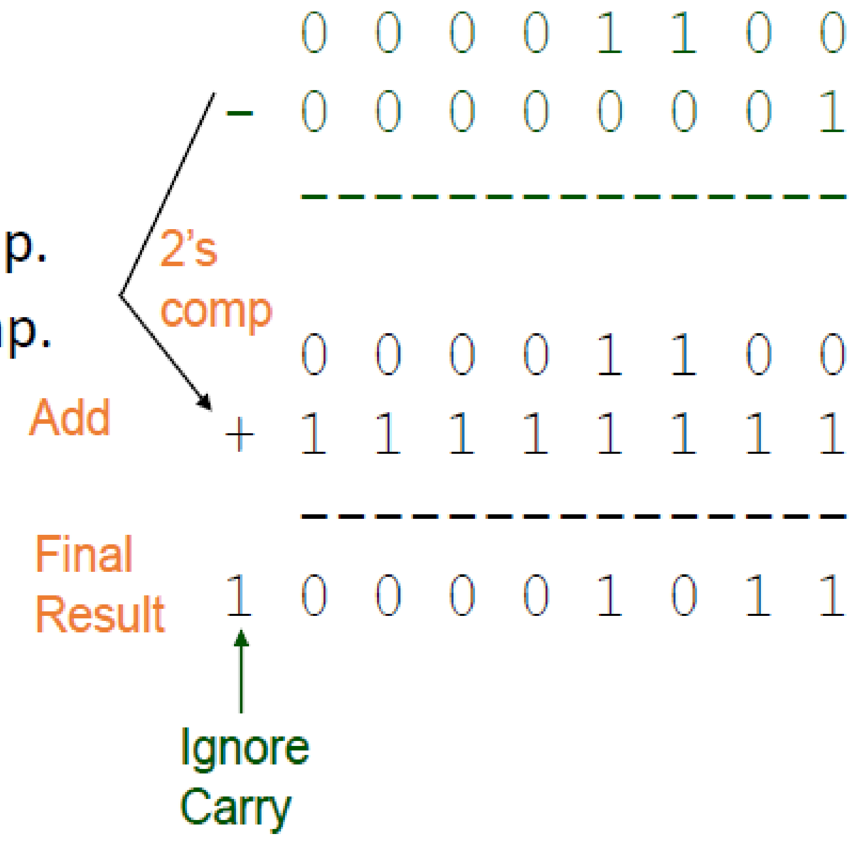
Step 1: Take 1's complement of 2nd operand
Step 2: Add binary numbers
Step 3: Add carry to low order bit

```
                    0 0 0 0 1 1 0 0
                  – 0 0 0 0 0 0 0 1
                    ---------------
      1's
      comp
                    0 0 0 0 1 1 0 0
      Add         + 1 1 1 1 1 1 1 0
                    ---------------
                  1 0 0 0 0 1 0 1 0
      Add carry   └──────────────→ 1
                    ---------------
      Final Result  0 0 0 0 1 0 1 1
```

2's Complement Subtraction

- Using 2's complement numbers, follow steps for subtraction
- For example, suppose we wish to subtract $+(00000001)_2$ from $+(00001100)_2$.
- Let's compute $(12)_{10} - (1)_{10}$.
  - $(12)_{10} = +(00001100)_2 = 00001100_2$ in 2's comp.
  - $-(1)_{10} = -(00000001)_2 = 11111111_2$ in 2's comp.

Step 1: Take 2's complement of 2nd operand
Step 2: Add binary numbers
Step 3: Ignore carry bit



```
      0 0 0 0 1 1 0 0
    - 0 0 0 0 0 0 0 1
      ---------------

      0 0 0 0 1 1 0 0
    + 1 1 1 1 1 1 1 1
      ---------------
    1 0 0 0 0 1 0 1 1
```

Fractions

- Binary to decimal

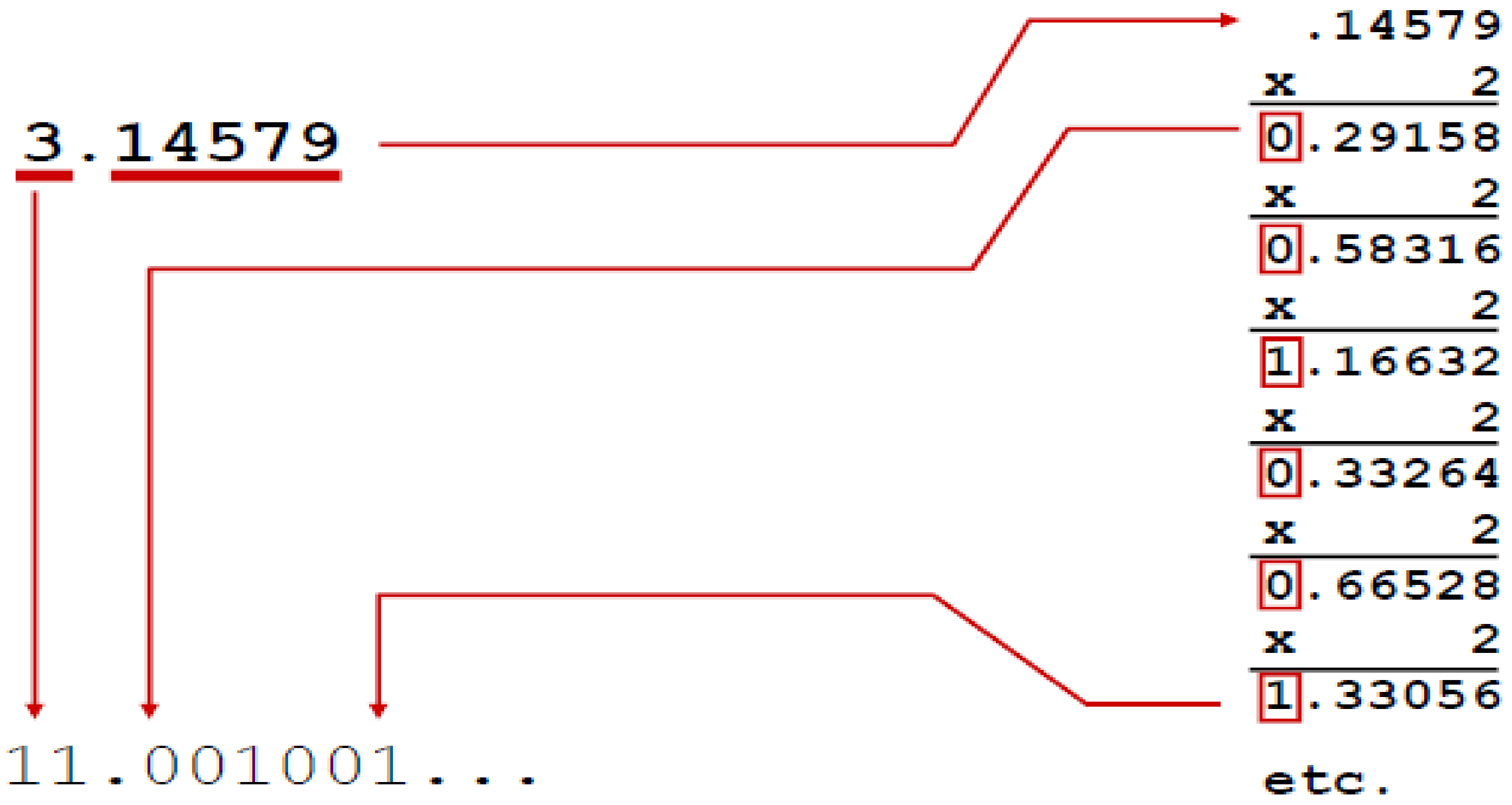
```
10.1011 =>    1 x 2^-4 = 0.0625
              1 x 2^-3 = 0.125
              0 x 2^-2 = 0.0
              1 x 2^-1 = 0.5
              0 x 2^0  = 0.0
              1 x 2^1  = 2.0
                        ________
                         2.6875
```

$b_1 b_0 . b_{-1} b_{-2} b_{-3} b_{-4}$

$$1 \times 2^{-4} = 0.0625$$
$$1 \times 2^{-3} = 0.125$$
$$0 \times 2^{-2} = 0.0$$
$$1 \times 2^{-1} = 0.5$$
$$0 \times 2^{0} = 0.0$$
$$1 \times 2^{1} = 2.0$$
$$2.6875$$

Fractions

- Decimal to binary



Conversion from Octal to Decimal

$$
\begin{aligned}
(431.65)_8 &= 4*8^2 + 3*8^1 + 1*8^0 + 6*8^{-1} + 5*8^{-2} \\
&= 4*64 + 3*8 + 1 + 6/8 + 5/64 \\
&= (281.828125)_{10}
\end{aligned}
$$

Octal and Hexadecimal Numbers

- Binary to Octal

(10	110	001	101	011	.	111	$100)_2$
(2	6	1	5	3	.	7	$4)_8$

- Binary to Hexadecimal

(10	1100	0110	1011	.	1111	$00)_2$
(2	C	6	B	.	F	$0)_8$

- $(10110001101011.111100)_2 = (26153.74)_8 = (2C6B.F0)_{16}$

Octal and Hexadecimal Numbers

- Octal to Binary

$$\begin{array}{cccccc} (6 & 7 & 3 & . & 1 & 2)_8 \\ (110 & 111 & 011 & . & 001 & 010)_2 \end{array}$$

- Hexadecimal to Binary

$$\begin{array}{ccccc} (3 & 0 & 6 & . & D)_{16} \\ (0011 & 0000 & 0110 & . & 1101)_2 \end{array}$$

Binary Addition

- Binary addition is very simple.
- This is best shown in an example of adding two binary numbers…

```
    1  1  1  1  1  1 ←———— carries
       1  1  1  1  0  1
+         1  0  1  1  1
-----------------------
    1  0  1  0  1  0  0
```

Binary Multiplication

- Binary multiplication is much the same as decimal multiplication, except that the multiplication operations are much simpler…

```
                  1   0   1   1   1
X                     1   0   1   0
-------------------------------------
                  0   0   0   0   0
              1   0   1   1   1
          0   0   0   0   0
      1   0   1   1   1
-------------------------------------
      1   1   1   0   0   1   1   0
```

Binary Division

25/5

```
                 1  0  1
         ________________
1  0  1 | 1  1  0  0  1
          1  0  1
          _______________
          0  0  1  0
             0  0  0
          _______________
                1  0  1
                1  0  1
          _______________
                0  0  0
```

Alphanumeric Data

There are different standards for representing letters (alpha) and numbers

- ASCII – American standard code for information interchange
- EBCDIC – Extended binary-coded decimal interchange code
- Unicode

ASCII Features

- 7-bit code
- 8th bit is unused
- $2^7 = 128$ codes
- Two general types of codes:
  - 95 are “Graphic” codes (displayable on a console)
  - 33 are “Control” codes (control features of the console or communications channel)

ASCII Chart

	000	001	010	011	100	101	110	111
0000	NULL	DLE		0	@	P	`	p
0001	SOH	DC1	[illegible]	1	A	Q	a	q
0010	STX	DC2	[illegible]	2	B	R	b	r
0011	ETX	DC	[illegible]	[illegible]	[illegible]	S	c	s
0100	EDT	DC	[illegible]	[illegible]	[illegible]	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
011	BEL	ETB	'	7	G	W	g	w
100	BS	CAN	(	8	H	X	h	x
100	HT	EM	)	9	I	Y	i	y
101	LF	SUB	*	:	J	Z	j	z
[illegible]	[illegible]	[illegible]	+	;	K	[	k	{
[illegible]	[illegible]	[illegible]	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

Most significant bit

Least significant bit

“Hello, world” Example

		Binary		Hexadecimal		Decimal
H	=	01001000	=	48	=	72
e	=	01100101	=	65	=	101
l	=	01101100	=	6C	=	108
l	=	01101100	=	6C	=	108
o	=	01101111	=	6F	=	111
,	=	00101100	=	2C	=	44
	=	00100000	=	20	=	32
w	=	01110111	=	77	=	119
o	=	01100111	=	67	=	103
r	=	01110010	=	72	=	114
l	=	01101100	=	6C	=	108
d	=	01100100	=	64	=	100

Unicode : 1 up to 4 byte

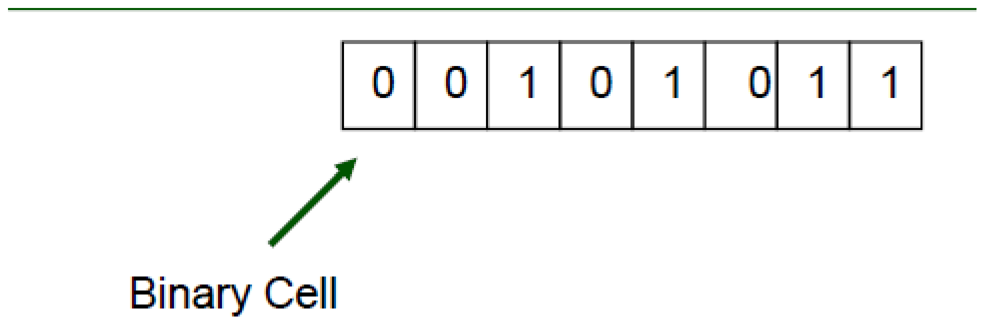
Code point types :

- UTF-8
- UTF -16
- UTF -32

U+1F600 :	😀	U+263A :	☺️	U+1F61F :	😟	U+1F61E :	😞
U+1F603 :	😃	U+1F61A :	😚	U+1F641 :	🙁	U+1F648 :	🙈
U+1F604 :	😄	U+1F619 :	😙	U+2639 :	☹️	U+1F649 :	🙉
U+1F601 :	😁	U+1F60B :	😋	U+1F62E :	😮	U+1F64A :	🙊
U+1F606 :	😆	U+1F61B :	😛	U+1F62F :	😯	U+1F44B :	👋
U+1F605 :	😅	U+1F61C :	😜	U+1F632 :	😲	U+1F91A :	🤚
U+1F923 :	🤣	U+1F92A :	🤪	U+1F633 :	😳	U+1F590 :	🖐️
U+1F602 :	😂	U+1F61D :	😝	U+1F626 :	😦	U+270B :	✋
U+1F642 :	🙂	U+1F911 :	🤑	U+1F627 :	😧	U+1F596 :	🖖
U+1F643 :	🙃	U+1F917 :	🤗	U+1F628 :	😨	U+1F44C :	👌
U+1F609 :	😉	U+1F92D :	🤭	U+1F630 :	😰	U+270C :	✌️
U+1F60A :	😊	U+1F92B :	🤫	U+1F625 :	😥	U+1F91E :	🤞
U+1F607 :	😇	U+1F914 :	🤔	U+1F622 :	😢	U+1F91F :	🤟
U+1F60D :	😍	U+1F60E :	😎	U+1F62D :	😭	U+1F918 :	🤘
U+1F929 :	🤩	U+1F913 :	🤓	U+1F631 :	😱	U+1F919 :	🤙
U+1F618 :	😘	U+1F9D0 :	🧐	U+1F616 :	😖	U+1F44D :	👍
U+1F617 :	😗	U+1F615 :	😕	U+1F623 :	😣	U+1F44E :	👎

Binary Data Storage

- **Binary cells store individual bits of data**
- **Multiple cells form a register.**



Transfer of Information

- Data input at keyboard
- Shifted into place
- Stored in memory

NOTE: Data input in ASCII

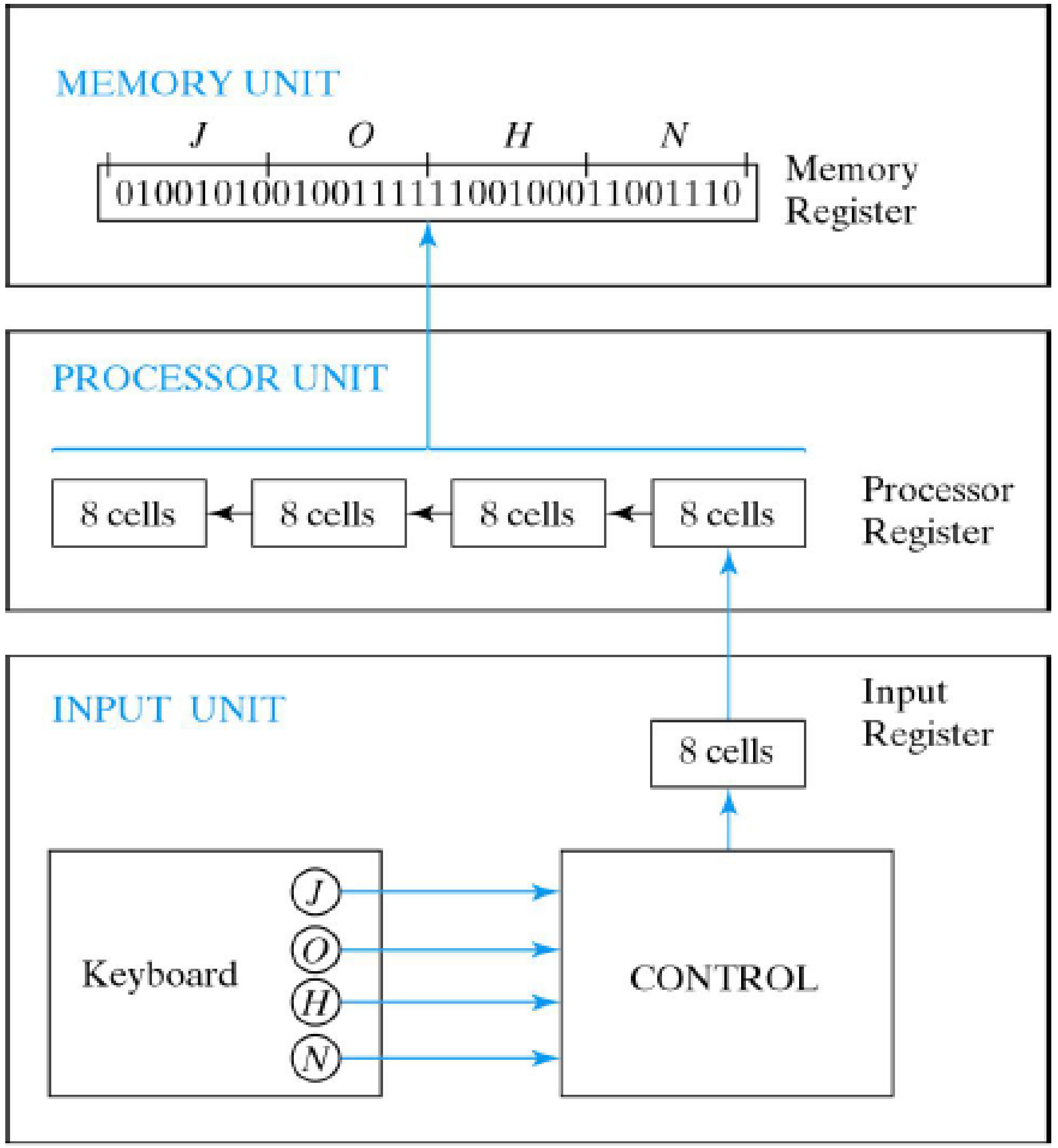


Fig. 1-1 Transfer of information with registers

Building a Computer

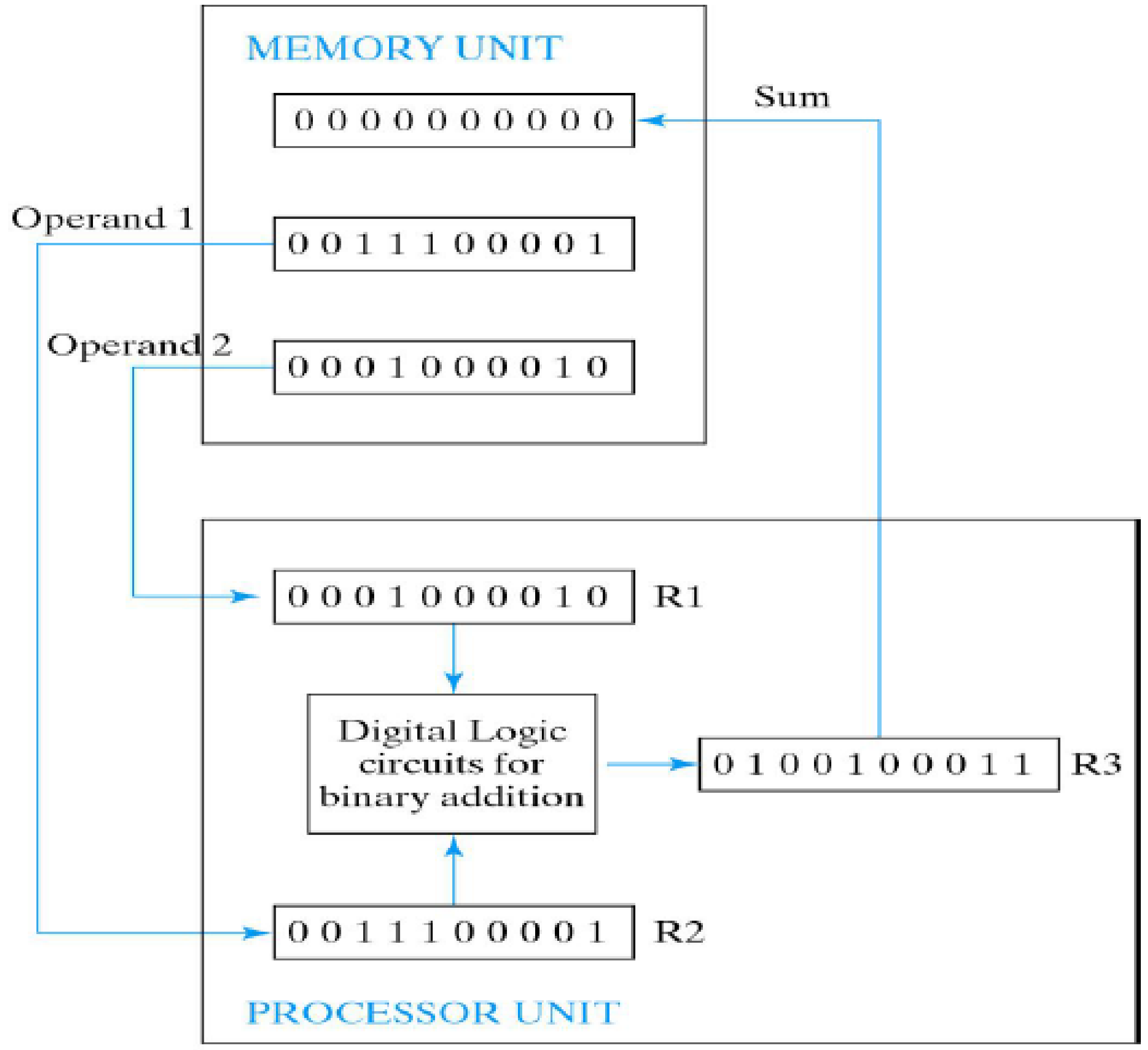


Fig. 1-2 Example of binary information processing

- We need processing
- We need storage
- We need communication

- You will learn to use and design these components.