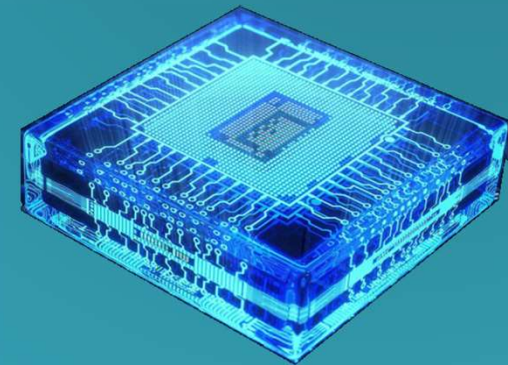




Microprocessors and Assembly language

Isfahan University of Technology (IUT)



Introduction to computing

Dr. Hamidreza Hakim
hakim@iut.ac.ir

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



Topics

- Number system
- Digital primer
- Internal organization of computers
 - The different parts of a computer
 - I/O
 - Memory
 - CPU
 - Connecting the different parts
 - Connecting memory to CPU
 - Connecting I/Os to CPU
 - How computers work

- **Decimal Numbers**
- **Binary Number system**
- **Hexadecimal Number system**

NUMBER SYSTEM

Decimal Numbers

➤ In a positional number system

- Each digit in the number is associated with a power of 10.
- i.e. 3245 = 3 thousands, 2 hundreds, 4 tens and 5 ones.
- OR, $3245 = 3 \times (10^3) + 2 \times (10^2) + 4 \times (10^1) + 5 \times (10^0)$

➤ In a positional number system

- A number b is selected as the **base**
- Symbols are assigned to **numbers between 0 and $b-1$**
 - In decimal system symbols are 0,1,2,3,4,5,6,7,8,9.
 - The base ten is represented as 10

Binary Number System

The base is two

There are only two digits, 0 and 1.

i.e. binary string 11010 represent the number:

$$1X(2^4)+1X(2^3)+0X(2^2)+1X(2^1)+0X(2^0) = 26$$

Base two is represented in binary as 2

Why Hexadecimal?

- We have decimal and binary.

Why Hex?

➤ Binary:

- Numbers written in binary tend to be long and difficult to express
- 16 bits are needed to represent a word in 8086-based computer.

➤ Decimal:

- Difficult to convert into binary.
Thus a third number became necessary

Hexadecimal Number System

**Table 1: Base 16
Number System**

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

- The base is **sixteen**
- There are total sixteen digits:
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- i.e. hex digit **4A** represent the number:
 - $4 \times (16^1) + A \times (16^0) = 74$
 - Base sixteen is represented in hex as 16
- Conversion between binary and hex is easy
- Because Sixteen is $= 2^4$, So each hex digit corresponds to a **unique four-bit number**.(nibble)

Decimal, Binary and Hex Numbers

DEC	HEX	BIN	DEC	HEX	BIN	DEC	HEX	BIN
0	00	00000000	43	2B	00101011	86	56	01010110
1	01	00000001	44	2C	00101100	87	57	01010111
2	02	00000010	45	2D	00101101	88	58	01011000
3	03	00000011	46	2E	00101110	89	59	01011001
4	04	00000100	47	2F	00101111	90	5A	01011010
5	05	00000101	48	30	00110000	91	5B	01011011
6	06	00000110	49	31	00110001	92	5C	01011100
7	07	00000111	50	32	00110010	93	5D	01011101
8	08	00001000	51	33	00110011	94	5E	01011110
9	09	00001001	52	34	00110100	95	5F	01011111
10	0A	00001010	53	35	00110101	96	60	01100000
11	0B	00001011	54	36	00110110	97	61	01100001
12	0C	00001100	55	37	00110111	98	62	01100010
13	0D	00001101	56	38	00111000	99	63	01100011
14	0E	00001110	57	39	00111001	100	64	01100100
15	0F	00001111	58	3A	00111010	101	65	01100101
16	10	00010000	59	3B	00111011	102	66	01100110
17	11	00010001	60	3C	00111100	103	67	01100111
18	12	00010010	61	3D	00111101	104	68	01101000
19	13	00010011	62	3E	00111110	105	69	01101001
20	14	00010100	63	3F	00111111	106	6A	01101010
21	15	00010101	64	40	01000000	107	6B	01101011
22	16	00010110	65	41	01000001	108	6C	01101100
23	17	00010111	66	42	01000010	109	6D	01101101
24	18	00011000	67	43	01000011	110	6E	01101110
25	19	00011001	68	44	01000100	111	6F	01101111
26	1A	00011010	69	45	01000101	112	70	01110000
27	1B	00011011	70	46	01000110	113	71	01110001
28	1C	00011100	71	47	01000111	114	72	01110010
29	1D	00011101	72	48	01001000	115	73	01110011
30	1E	00011110	73	49	01001001	116	74	01110100
31	1F	00011111	74	4A	01001010	117	75	01110101
32	20	00100000	75	4B	01001011	118	76	01110110
33	21	00100001	76	4C	01001100	119	77	01110111
34	22	00100010	77	4D	01001101	120	78	01111000
35	23	00100011	78	4E	01001110	121	79	01111001
36	24	00100100	79	4F	01001111	122	7A	01111010
37	25	00100101	80	50	01010000	123	7B	01111011
38	26	00100110	81	51	01010001	124	7C	01111100
39	27	00100111	82	52	01010010	125	7D	01111101
40	28	00101000	83	53	01010011	126	7E	01111110
41	29	00101001	84	54	01010100	127	7F	01111111
42	2A	00101010	85	55	01010101			

Conversion Between Number Systems

Binary to decimal

Decimal to Binary

Hexadecimal to Decimal

Decimal to Hex

Hex to Binary

Binary to Hex


Decimal to Binary

- Convert the Decimal number 25 in Binary.

Given a Decimal Number : **25**₁₀

Divide it with 2 until the quotient becomes zero

2	25		
2	12	- 1	(remainder of 25/2)
2	6	- 0	(remainder of 12/2)
2	3	- 0	(remainder of 6/2)
2	1	- 1	(remainder of 3/2)
2	0	- 1	(remainder of 1/2)



Place the remainders in the reverse order to reach
the equivalent Binary number : **11001**₂

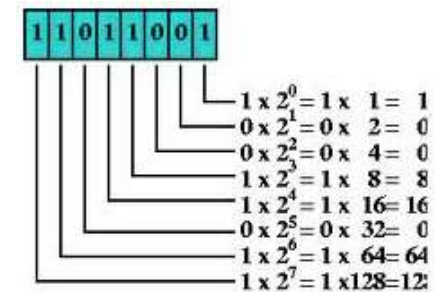
Binary to Decimal

- Convert the binary number 1111011 in decimal.

$$= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

$$= 64 + 32 + 16 + 8 + 2 + 1$$

$$= 123$$



$$1 + 8 + 16 + 64 + 128 = 217$$

- Convert the binary numbers into decimal numbers

- 11
- 101
- 1111
- 110111011

Hexadecimal to Decimal

➤ **Convert hex number 589 into decimal number.**

$$=5*(16^2)+8*(16^1)+9*(16^0)$$

$$- = 1280+128+9$$

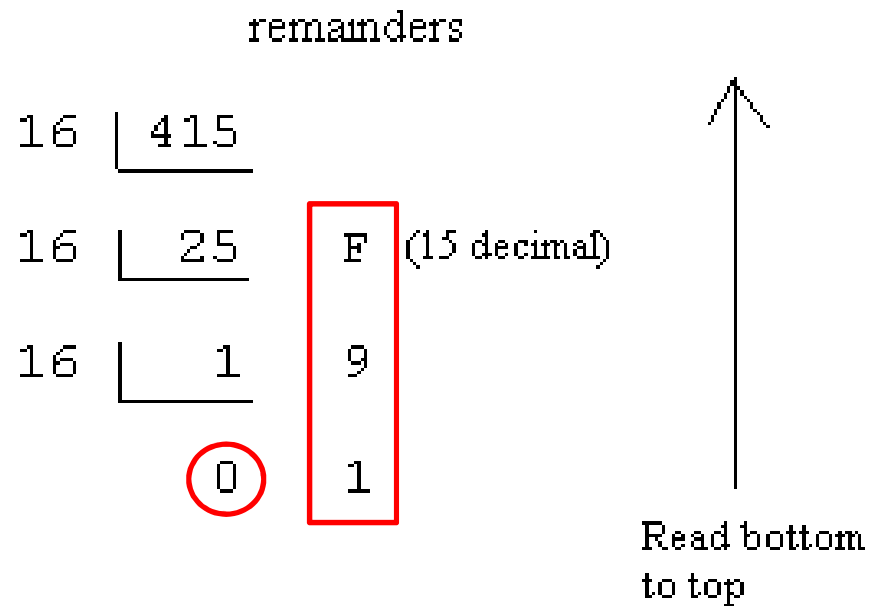
$$- = 1417$$

➤ **Convert the following hex numbers into decimal numbers**

- **A0**
- **8F**
- **1531**
- **FA8**

Decimal to Hex

➤ Convert 415 to Hex



Result: 415 in decimal is 19F in hexadecimal

Decimal to Hex

- **Convert 330 to Hex**

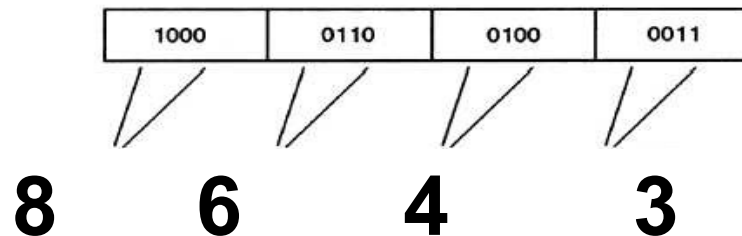
Hex to Binary

➤ Convert 39A2h to Binary

Hexadecimal	3	9	A	2
Binary	0011	1001	1010	0010

Binary to Hex

- Convert the Binary number 1000011001000011 into Hex



- Convert the Binary number 1001001001000100110101101111 into Hex

The binary number: 0001 0010 0100 1000 1001 1010 1101 1111

The hexadecimal number: 1 2 5 8 9 A D F

Conversion Task: Binary and Hex to Decimal

» **1110b**

» **100101011101b**

» **46Ah**

» **FAE2Ch**

Conversion Task: Decimal and Hex into Binary

» **97**

» **627**

» **A2Ch**

» **B34Dh**

Conversion Task: Decimal and Binary to Hex

» **921**

» **6120**

» **10101**

» **1001011b**

» **1001010110101110b**

OPERATIONS

Binary Addition

Binary addition table

\oplus	0	1
0	0	1
1	1	10

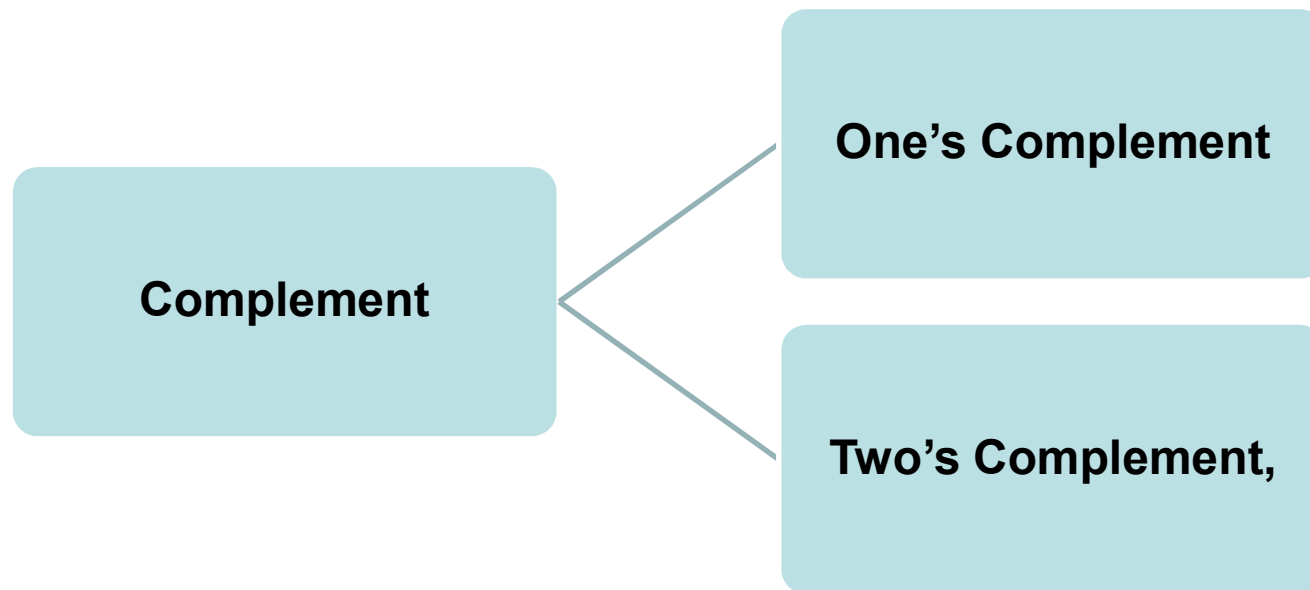
	<i>Binary</i>	<i>Decimal</i>
	1101	13
+	<u>1001</u>	<u>9</u>
	10110	22

Task: Binary Addition

➤ **Solve the following:**

- **$100101b + 10111b$**
- **$100111101b + 10001111001b$**

Complement



One's Complement

- Ones complement is obtained by **complementing each bit.**
i.e. replacing 0 by a 1 and 1 by a 0.
- **To find the one's complement of 5 or 0000000000000101**
[16bit representation of 101]
1111111111111010 [complementing each bit]

Two's Complement

- **To get Two's Complement,**

Add 1 to the one's complement of a number

Thus, Two's complement of 5 is

1111111111111010 [one's complement of 5]

+1

1111111111111011 [Two's complement of 5]

Subtraction: Binary

Subtraction: Binary

- 2's complement
- there is no separate circuitry for subtractors.
- Instead, adders are used in conjunction with 2's complement circuitry to perform subtraction
- $x-y \rightarrow x+(\text{the 2's complement of } y)$

Take the 2's complement of 10011101.

Solution:

	10011101	binary number
	01100010	1's complement
+	<u>1</u>	
	01100011	2's complement

Subtraction: Binary

➤ Subtract number 1001b from 0111b

- In Unit's column $1b - 1b = 0b$
- In next column, $0 < 1$ so, we must **borrow 1** from the third (right to left) column. Thus, $10b - 1b = ?$
 - Binary numbers are 0,1
 - $10b/2d - 1b/d = 1b$
- In next column, $10b - 1b - 1b$ (previously lend one) = $10b/2d - 1b/d - 1b/d = 0b$

– thus, if we combine it all,

1001b

-0111b

=====

0010b

Task: Binary Subtraction

➤ **Solve the following:**

- **11011b-10110b**
- **10000101b-111011b**

Hex Addition

Solution:

$$\begin{array}{r} 23D9 \\ + \quad 94BE \\ \hline B897 \end{array}$$

LSD: $9 + 14 = 23$

$1 + 13 + 11 = 25$

$1 + 3 + 4 = 8$

MSD: $2 + 9 = B$

$23 - 16 = 7$ with a carry

$25 - 16 = 9$ with a carry

Note: Carry

**Table 1: Base 16
Number System**

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hex Addition Table

Table for Hexadecimal Addition

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Task: Hex Addition

➤ **Solve the following:**

- **B23CDh + 17912h**
- **FEFFh + FBCADh**

Subtraction: Hex

- In subtracting two hex numbers,
- if the second digit is greater than the first, borrow 16 from the preceding digit

Perform hex subtraction: $59F - 2B8$.

Solution:

$$\begin{array}{r} 59F \\ - 2B8 \\ \hline 2E7 \end{array}$$

LSD: 8 from 15 = 7
11 from 25 ($9 + 16$) = 14 (E)
2 from 4 ($5 - 1$) = 2

Table 1: Base 16
Number System

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Task: Hex Subtraction

➤ **Solve the following:**

- **5FC12h-3ABD1h**
- **F001Ch-1FF3Fh**

Integer Representation in computer

- Hardware of a computer **restricts the size** of a number can be stored.

lsb = least significant bit [**Right most bit or bit 0**] (Important: Note first bit is named bit 0)

msb = most significant bit [**left most bit or bit 15**]

- Unsigned integers:** Non-negative integer

The largest unsigned integer of a **byte**= **11111111b** or **255d** or **FFh**

The largest unsigned integer of a **word**= **1111111111111111b** or **65535d** or **FFFFh**

*** if LSB of an integer is **1** then its **ODD** . However, if it is **0** then **Even**

- Signed Integer:** Can be **positive** or **negative** number

The **MSB** is reserved for the **sign**

MSB =1 [Negative]

MSB =0 [Positive]

*** Negative integers are stored in a computer using **TWO's complement**

Observation

- If we add two's complement of 5 with 5, we get

1111111111111011

0000000000000101

=====

1000000000000000

We get total 17bits here, however, word bits can hold only 16bits.

Thus the carried out **MSB 1** is lost and final result is 0

5 +(-5) also results **zero**.

Task: 8/16bit Representation

(Important: default representation in this course is 8-bit)

- Find the two's complement of the two's complement of 5.

Show how the following decimal integer would be represented in

a) 8 bits

b) 16bits. [express the result in hex]

-97

-120

-40000

-128

65536

Task

- Give 16-bit representation of each of the following integers and express the result in hex:

234

-16

31634

-32216

Subtraction as Two's Complement Addition

- If AX contains 5ABCh and BX contains 21FCh, find AX-BX

21FCh = 0010000111111100

5ABCh = 0101101010111100

21FCh = 1101111000000011 [one's complement]

+1

=====

10011100011000000 = 38C0h [1 is carried and lost]

ASCII CODE

ASCII Code



In the 1960s a standard representation called ASCII



(American Standard Code for Information Interchange)



The ASCII system uses a total of 7 bits

ASCII Code

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

- Important Notes

- digits 0 through 9: to **convert** ASCII to decimal by masking off the “3”
- the only bit that is **different between** the uppercase “A” and lowercase “a” is bit 5

ASCII in detail

- ASCII code system Uses **7bits** = 2^7 = **128** to code each character.
 - Codes from **32 to 126 = 95** characters are only **printable**.
- **** to display the character **A** on the screen, a program sends the ASCII code **41h** to the screen.

How save “RG 2z” in 8 bit memory?

Address	Character	ASCII Code	Content

Decimal Hex Char				Decimal Hex Char				Decimal Hex Char				Decimal Hex Char			
0	0		[NULL]	32	20		[SPACE]	64	40	@	96	60			
1	1		[START OF HEADING]	33	21			65	41	A	97	61	a	b	
2	2		[START OF TEXT]	34	22			66	42	B	98	62			
3	3		[END OF TEXT]	35	23	#		67	43	C	99	63	c		
4	4		[END OF TRANSMISSION]	36	24			68	44	D	100	64			
5	5		[ENQUIRY]	37	25	%		69	45	E	101	65	e	f	
6	6		[ACKNOWLEDGE]	38	26			70	46	F	102	66	f		
7	7		[BELL]	39	27			71	47	G	103	67	g		
8	8		[BACKSPACE]	40	28			72	48	H	104	68	h	i	
9	9		[HORIZONTAL TAB]	41	29			73	49	I	105	69			
10	A		[LINE FEED]	42	2A	*		74	4A	J	106	6A	j		
11	B		[VERTICAL TAB]	43	2B			75	4B	K	107	6B	k		
12	C		[FORM FEED]	44	2C			76	4C	L	108	6C	l		
13	D		[CARRIAGE RETURN]	45	2D	-		77	4D	M	109	6D	m	n	
14	E		[SHIFT OUT]	46	2E			78	4E	N	110	6E			
15	F		[SHIFT IN]	47	2F			79	4F	O	111	6F	o		
16			[DATA LINK ESCAPE]	48	30			80	50		112	70			
17	1		[DEVICE CONTROL 1]	49	31			81	51	Q	113	71	q		
18	2		[DEVICE CONTROL 2]	50	32			82	52	R	114	72	r		
19	3		[DEVICE CONTROL 3]	51	33			83	53	S	115	73	s		
20	4		[DEVICE CONTROL 4]	52	34			84	54	T	116	74	t		
21	5		[NEGATIVE ACKNOWLEDGE]	53	35			85	55	U	117	75	u		
22	6		[SYNCHRONOUS IDLE]	54	36			86	56	V	118	76	v		
23	7		[END OF TRANSMISSION BLOCK]	55	37			87	57	W	119	77	w		
24	8		[CANCEL]	56	38			88	58	X	120	78	x		
25	9		[END OF MEDIUM]	57	39			89	59	Y	121	79	y		
26	A		[SUBSTITUTE]	58	3A			90	5A	Z	122	7A	z		
27	10		[ESCAPE]	59	3B			91	5B	[123	7B	[
28	11		[FILE SEPARATOR]	60	3C			92	5C	[124	7C	[
29	1D		[GROUP SEPARATOR]	61	3D	=		93	5D	[125	7D	[
30	1E		[RECORD SEPARATOR]	62	3E			94	5E	^	126	7E	^		
31	1F		[UNIT SEPARATOR]	63	3F			95	5F	_	127	7F	_	[DEL]	

Address	Character	ASCII Code	Content
0	R	52	0101 0010
1	G	47	0100 0111
2	Space	20	0010 0000
3	2	32	0011 0010
4	z	7A	0111 1010

Task: String Processing in a Computer

- If we want to print “**Hello World**” on the screen, How the memory will look like ?
- [**Hint:** Find ASCII code, content of string from the ASCII table and order them in a address sequence].

DIGITAL PRIMER



DIGITAL PRIMER

- Binary logic

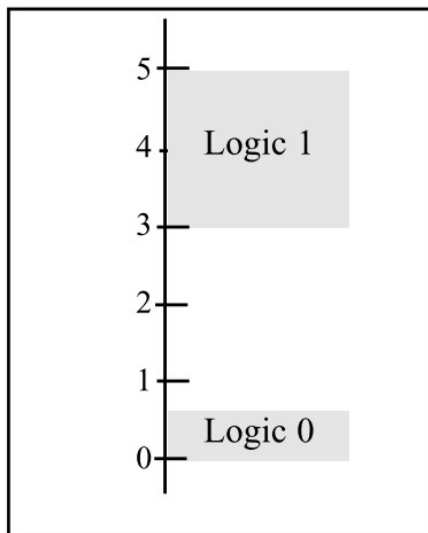
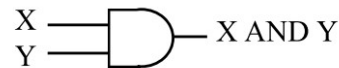


Figure 2. Binary Signals

Logical AND Function

Inputs		Output
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

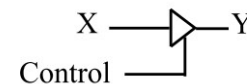


Logical OR Function

Inputs		Output
X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1



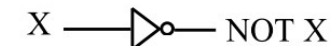
Buffer



used to isolate or amplify the signal.

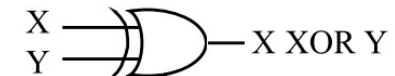
Logical Inverter

Input	Output
X	NOT X
0	1
1	0



Logical XOR Function

Inputs		Output
X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

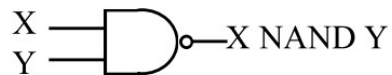


used to compare two bits

DIGITAL PRIMER

Logical NAND Function

Inputs	Output
X Y	X NAND Y
0 0	1
0 1	1
1 0	1
1 1	0

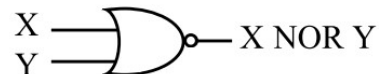


Any circuit using only NAND and NOR gates
(easy and inexpensive)

Notice in NAND,
if any input is 0, the output is 1.

Logical NOR Function

Inputs	Output
X Y	X NOR Y
0 0	1
0 1	0
1 0	0
1 1	0



Notice in NOR,
if any input is 1, the output is 0

Logic design using gates

- Add Two digit input

	<i>Carry</i>	<i>Sum</i>
$0 + 0 =$	0	0
$0 + 1 =$	0	1
$1 + 0 =$	0	1
$1 + 1 =$	1	0

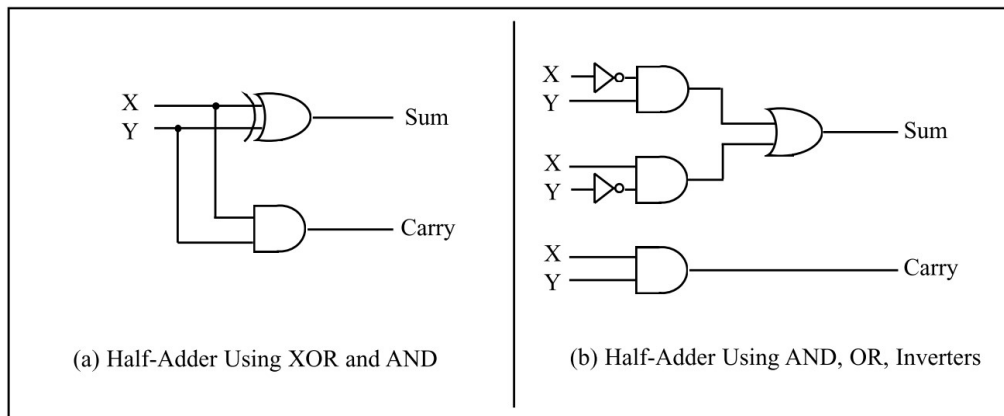


Figure 3. Two Implementations of a Half-Adder

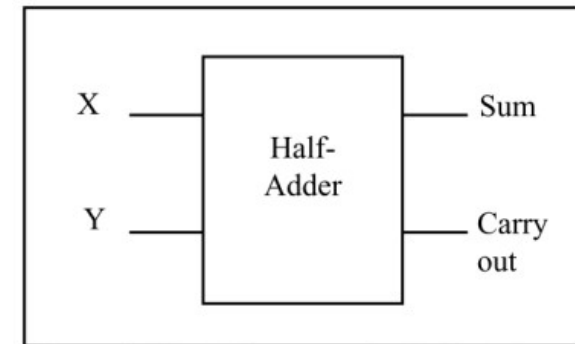
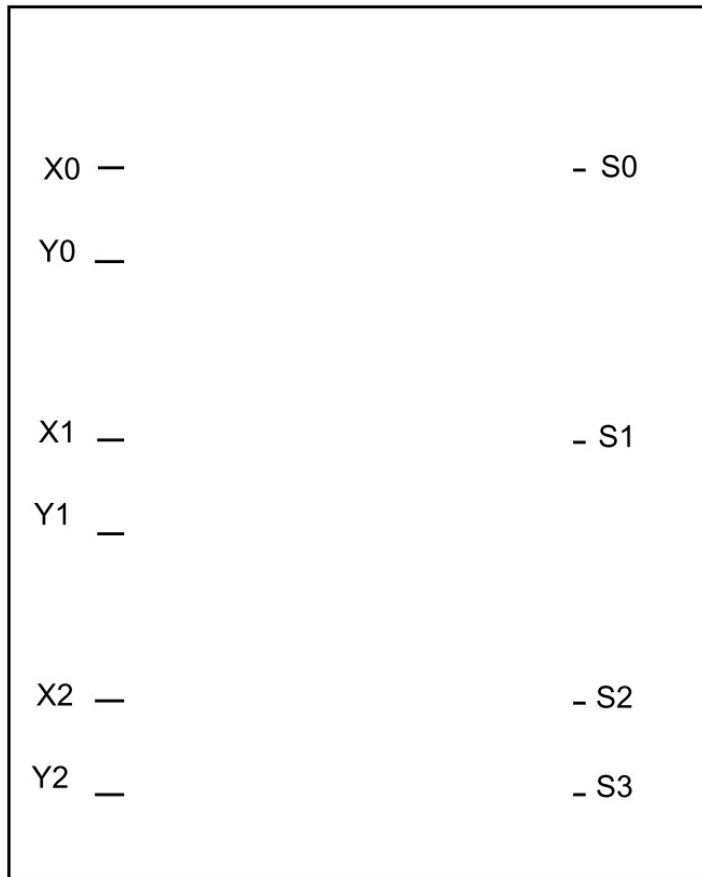


Figure 4. Block Diagram of a Half-Adder

Question

- 3-Bit Adder?



Logic design using gates

- Add three digit input (full-adder)
- Two half-adders can be combined to form an adder that can add three input digits

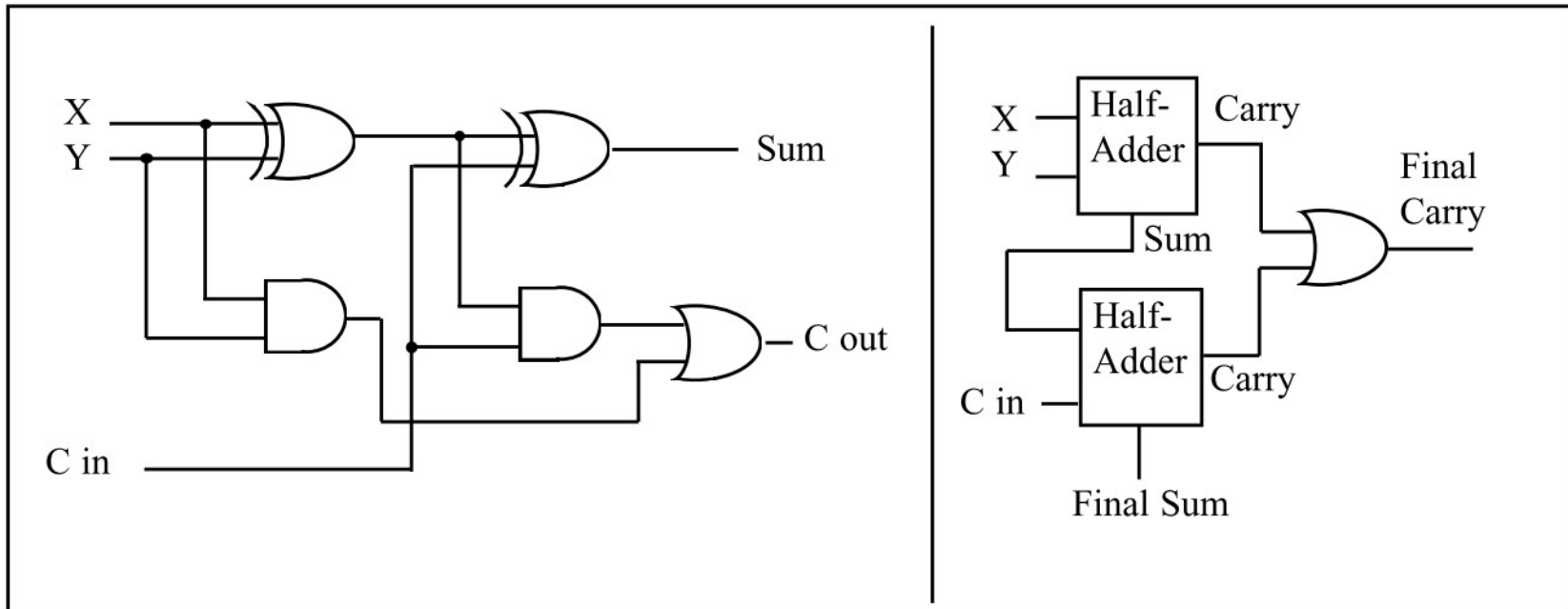
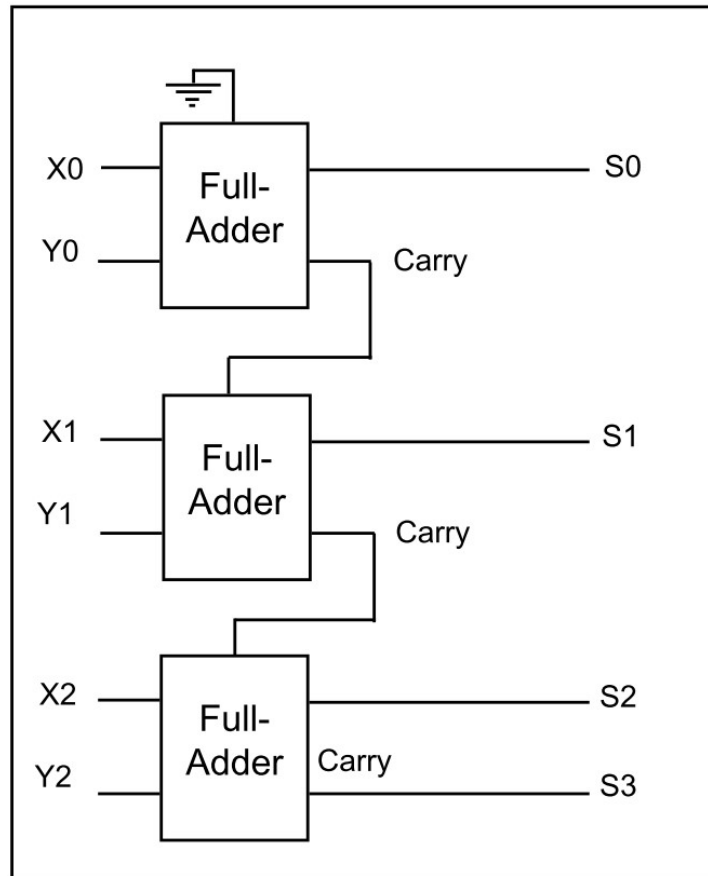


Figure 5. Full-Adder Built from a Half-Adder

Logic design using gates

- 3-Bit Adder Using Three Full Adders



Logic design using gates

- Decoders

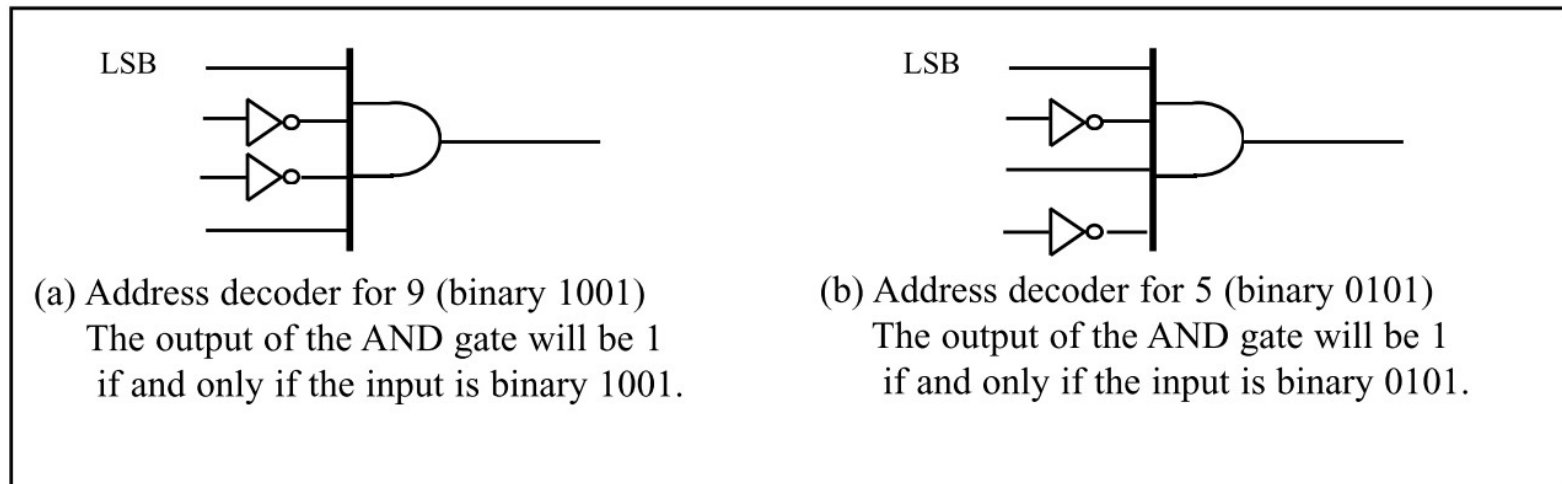


Figure 7. Address Decoders

Logic design using gates

- Flip-flops
- used to store data
- A D-FF holds the data as long as the power is on

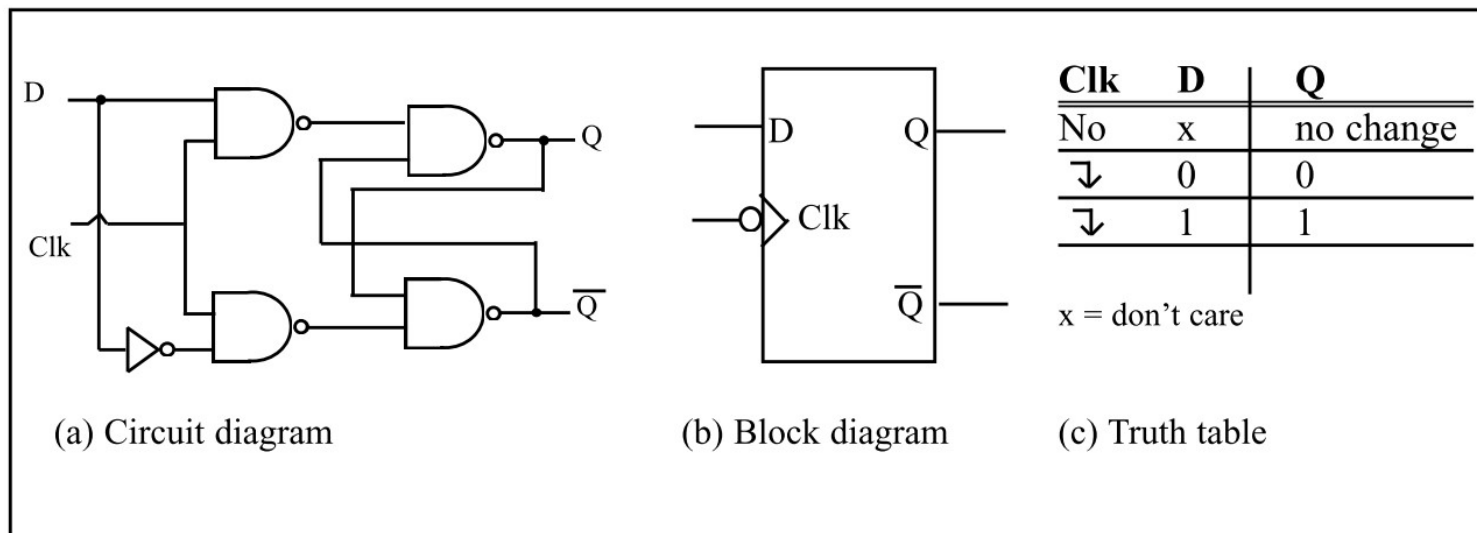


Figure 8. D Flip-Flops

SEMICONDUCTOR MEMORY



Internal organization of computers

- CPU
- Memory
- I/O
 - Input
 - E.g. Keyboard, Mouse, Sensor
 - Output
 - E.g. LCD, printer, hands of a robot

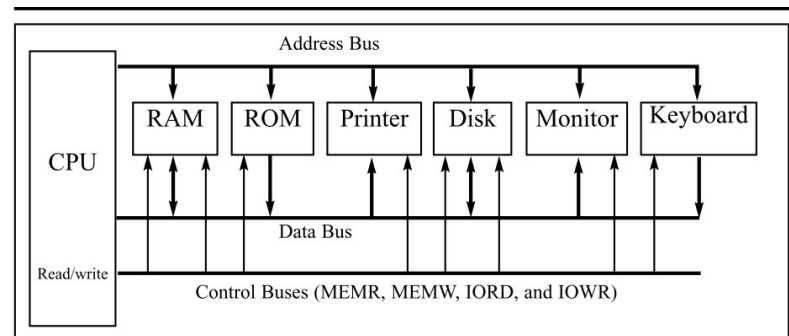
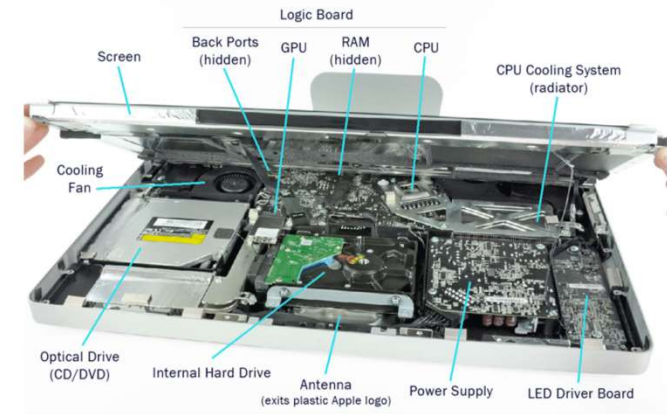


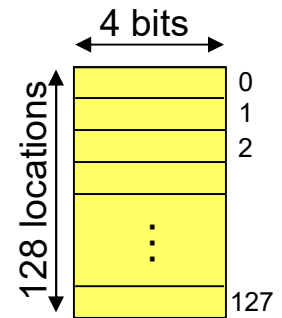
Figure 9. Internal Organization of a Computer

Memory

- Everything that can store, retain, and recall information.
 - E.g. hard disk, a piece of paper, etc.

Memory characteristics

- **Capacity**
 - The number of **bits** that a memory can store.
 - E.g. 128 Kbits, 256 Mbits
- **Organization**
 - How the locations are organized
 - E.g. a 128 x 4 memory has 128 locations, 4 bits each
- **Access time**
 - How long it takes to get data from memory



Example 12

A given memory chip has 12 address pins and 4 data pins. Find:
(a) the organization, and (b) the capacity.

Memory

- Semiconductors



The pictures are copied from <http://www.wikipedia.org/>

- Non-semiconductors

read-only memory



Semiconductor memories

- ROM
 - Mask ROM
 - PROM (Programmable ROM)
 - EPROM (Erasable PROM)
 - EEPROM
(Electronic Erasable PROM)
 - Flash EPROM
- RAM
 - (SRAM) Static RAM
 - (DRAM) Dynamic RAM
 - (NV-RAM) Nonvolatile RAM

read-only memory

Memory\ROM\Mask ROM

- Programmed by the IC manufacturer
 - Fix
 - Network operating systems and server operating systems
 - Videogame cartridges



Memory\ROM\PROM (Programmable ROM)

- OTP (One-Time Programmable)
 - You can program it only once
 - burning ROM!! (one fuse for each bit)

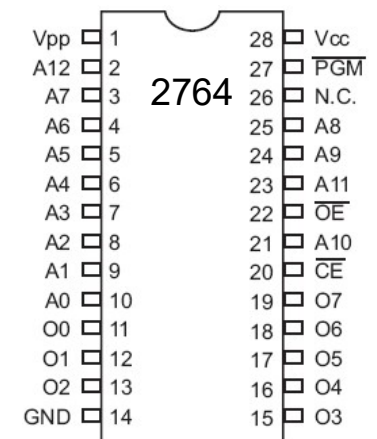
Memory\ROM\EPROM (Erasable Programmable ROM)

- **UV-EPROM**
 - You can shine **ultraviolet** (UV) radiation to **erase** it
 - Erasing takes up to **20 minutes**
 - The entire contents of ROM are erased
 - **27XX**



Table 0-5: Some UV-EPROM Chips

Part #	Capacity	Org.	Access	Pins	V _{pp}
2716	16K	2K × 8	450 ns	24	25 V
2732	32K	4K × 8	450 ns	24	25 V
2732A-20	32K	4K × 8	200 ns	24	21 V
27C32-1	32K	4K × 8	450 ns	24	12.5 V CMOS
2764-20	64K	8K × 8	200 ns	28	21 V
2764A-20	64K	8K × 8	200 ns	28	12.5 V
27C64-12	64K	8K × 8	120 ns	28	12.5 V CMOS



Memory\ROM\EPROM (Erasable Programmable ROM)

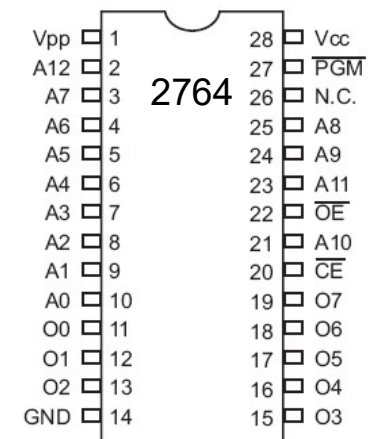
Example 14

For ROM chip 27128, find the number of data and address pins.



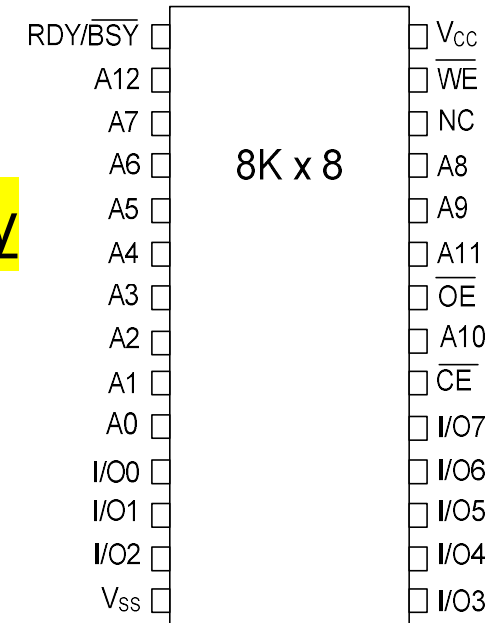
Table 0-5: Some UV-EPROM Chips

Part #	Capacity	Org.	Access	Pins	V _{pp}
2716	16K	2K × 8	450 ns	24	25 V
2732	32K	4K × 8	450 ns	24	25 V
2732A-20	32K	4K × 8	200 ns	24	21 V
27C32-1	32K	4K × 8	450 ns	24	12.5 V CMOS
2764-20	64K	8K × 8	200 ns	28	21 V
2764A-20	64K	8K × 8	200 ns	28	12.5 V
27C64-12	64K	8K × 8	120 ns	28	12.5 V CMOS



Memory\ROM\EEPROM (Electrically Erasable Programmable ROM)

- Erased Electrically
 - Erased instantly
 - Each byte can be erased separately



Part No.	Capacity	Org.	Speed	Pins	V _{pp}
2816A-25	16K	2K × 8	250 ns	24	5 V
2864A	64K	8K × 8	250 ns	28	5 V
28C64A-25	64K	8K × 8	250 ns	28	5 V CMOS
28C256-15	256K	32K × 8	150 ns	28	5 V
28C256-25	256K	32K × 8	250 ns	28	5 V CMOS

Memory\ROM\Flash ROM

- Erased in a Flash
- the entire device is **erased at once**
- **High Speed vs Disk(millisecond)**
- (Different with EEPROM and EPROM?)

Part No.	Capacity	Org.	Speed	Pins	V _{PP}
28F256-20	256K	32K × 8	200 ns	32	12 V CMOS
28F010-15	1024K	128K × 8	150 ns	32	12 V CMOS
28F020-15	2048K	256K × 8	150 ns	32	12 V CMOS

Semiconductor memories

- ROM

- Mask ROM
- PROM (Programmable ROM)
- EPROM (Erasable PROM)
- EEPROM (Electronic Erasable PROM)
- Flash EPROM

- RAM

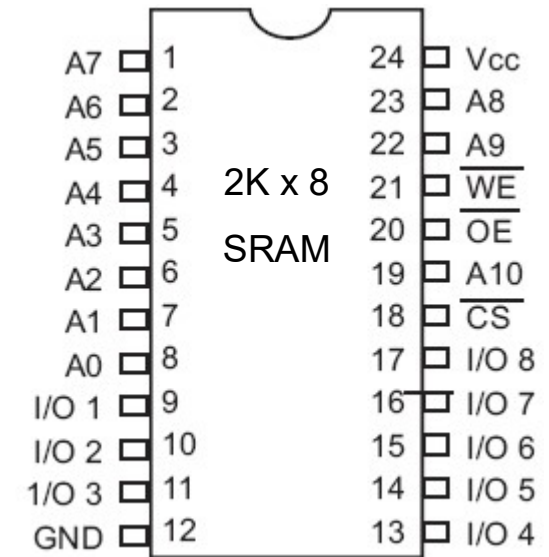
- (Static RAM) SRAM
- (Dynamic RAM) DRAM
- Nonvolatile) NV-RAM
- (RAM

random access memory

cutting off the power to the IC results in the loss of data.(volatile memory)

Memory\RAM\SRAM (Static RAM)

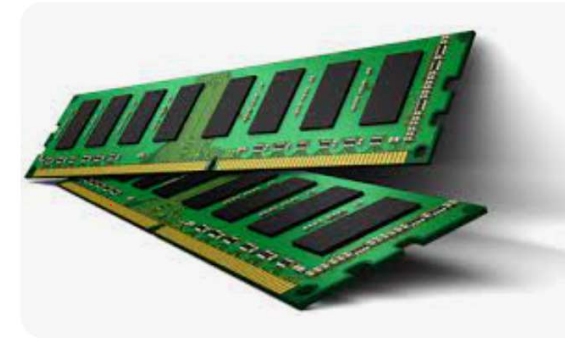
- Made of **flip-flops** (several Transistors)
- Against other RAMs
 - Advantages:
 - **Faster**
 - No need for refreshing
 - Disadvantages:
 - **High power consumption**
 - Expensive



- Made of SRAM,+Battery,+power control circuitry
- Advantages:
 - Very fast
 - Infinite program/erase cycle
 - Non-volatile(ten years!!!)
- Disadvantage:
 - Expensive

Memory\RAM\DRAM (Dynamic RAM)

- Made of capacitors
- Advantages:
 - Less power consumption
 - Cheaper
 - High capacity(High Density capacitor vs. flip flop)
 - Smaller
- Disadvantages:
 - Refresh needed



DRAM in Iran Video

Packaging issue in DRAM

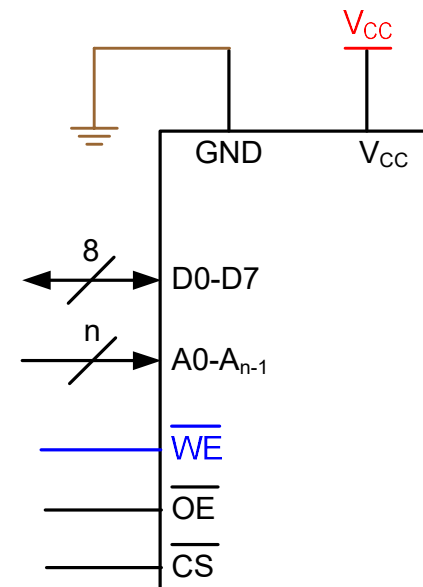
- 64Kbit (64 x 1) need 16+1+(vcc+..) pin
- Instead
 - To access a bit of data from DRAM, both row and column addresses must be provided.
 - For this concept to work, there must be a 2-by-1 multiplexer outside the DRAM circuitry and a demultiplexer inside every DRAM chip

Internal parts of computers\CPU

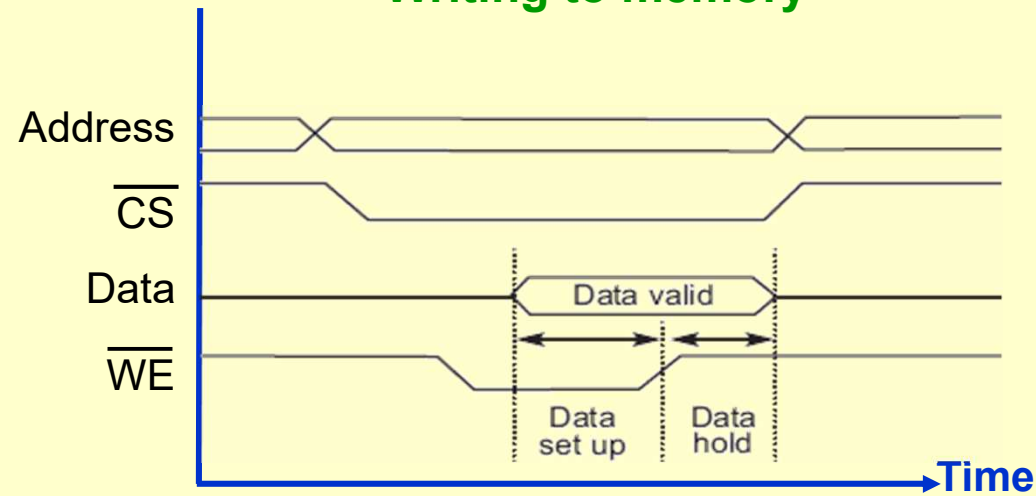
- Tasks:
 - It should execute instructions
 - It should recall the instructions one after another and execute them

Connecting memory to CPU

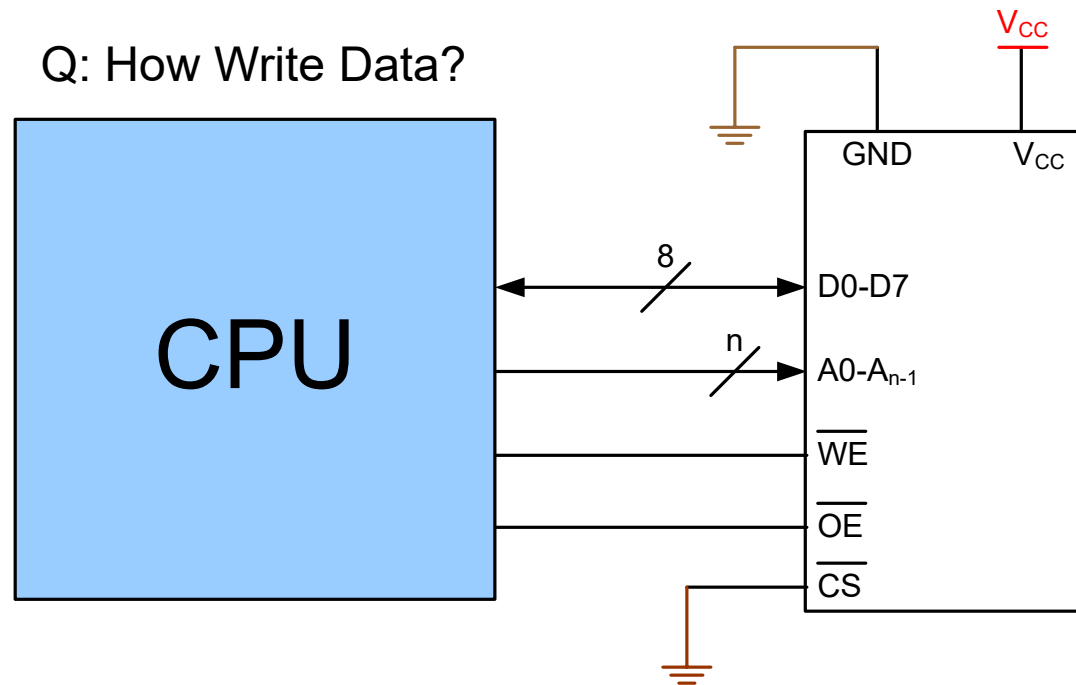
- Memory pin out
- Q: What is OE CS?



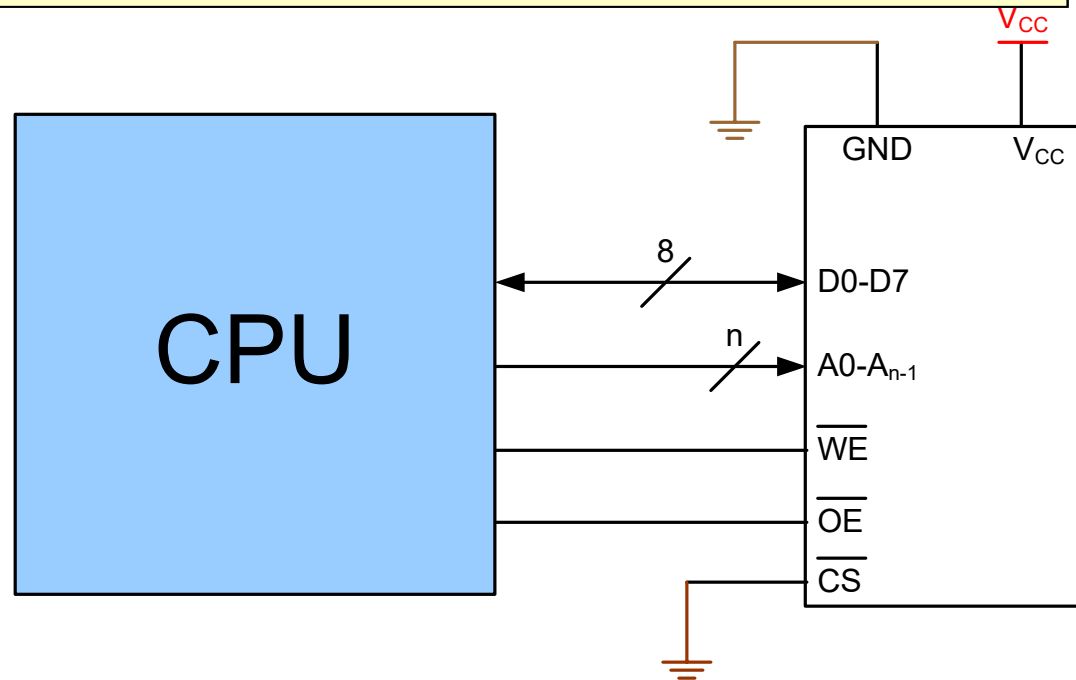
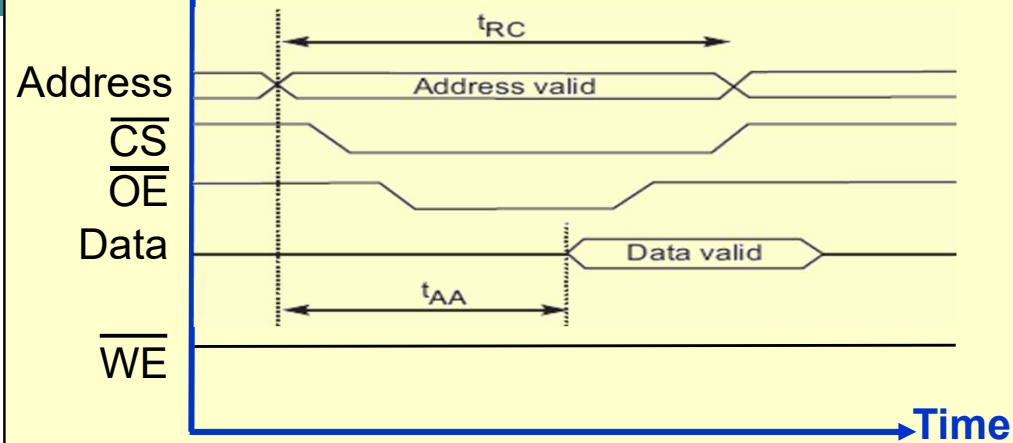
Writing to memory



Q: How Write Data?

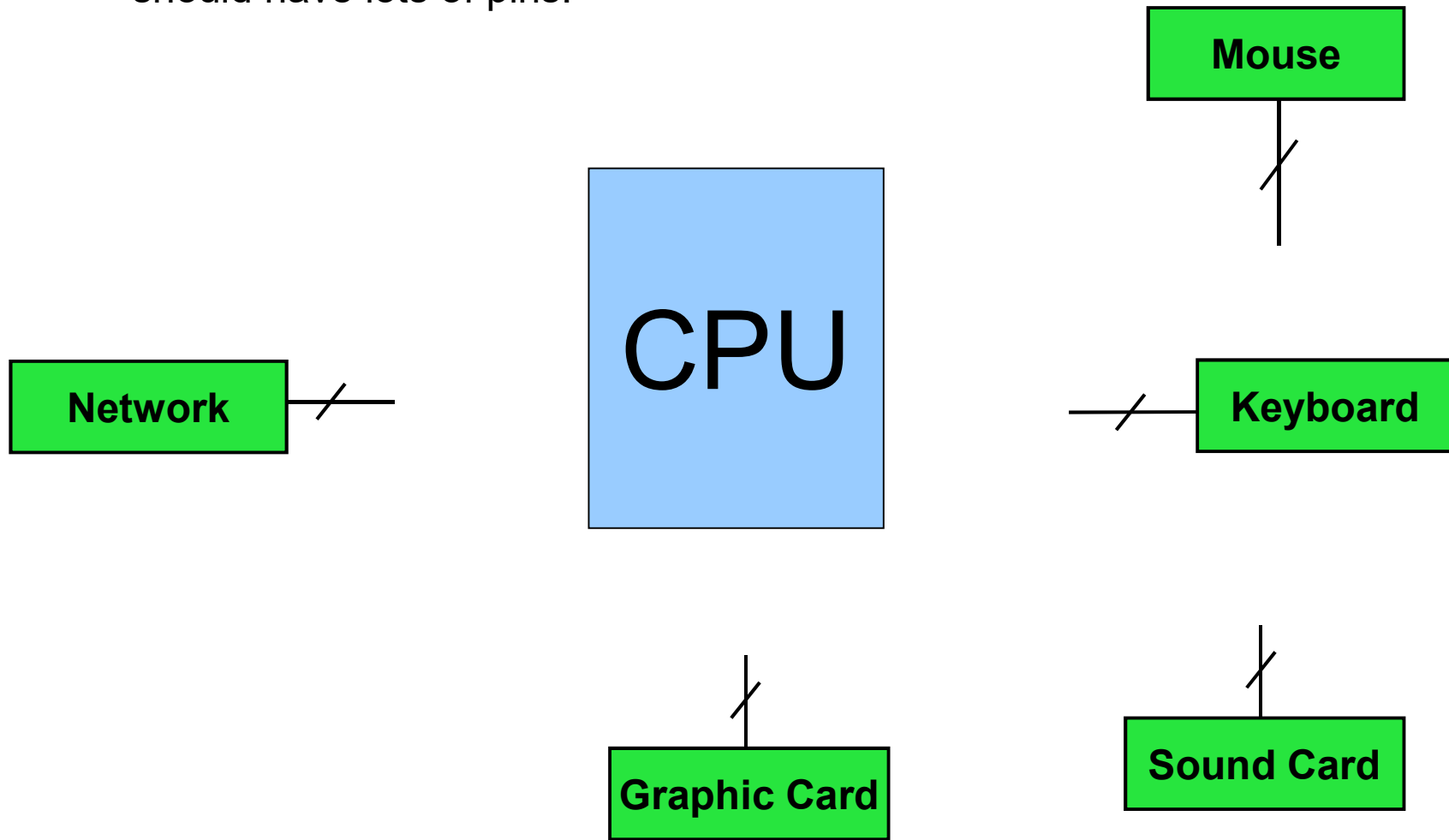


Reading from memory



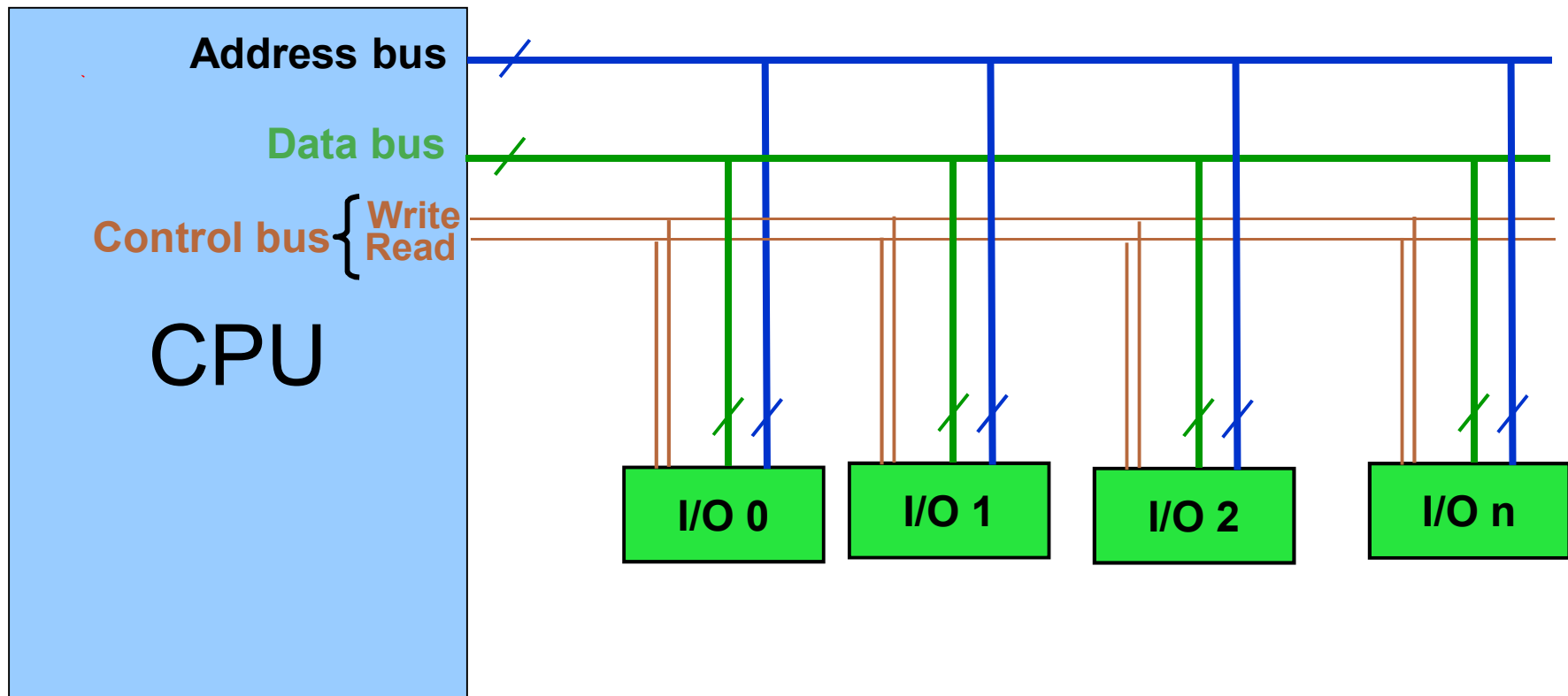
Connecting I/Os to CPU

- Q:How Connect All I/O to CPU?
 - should have lots of pins!

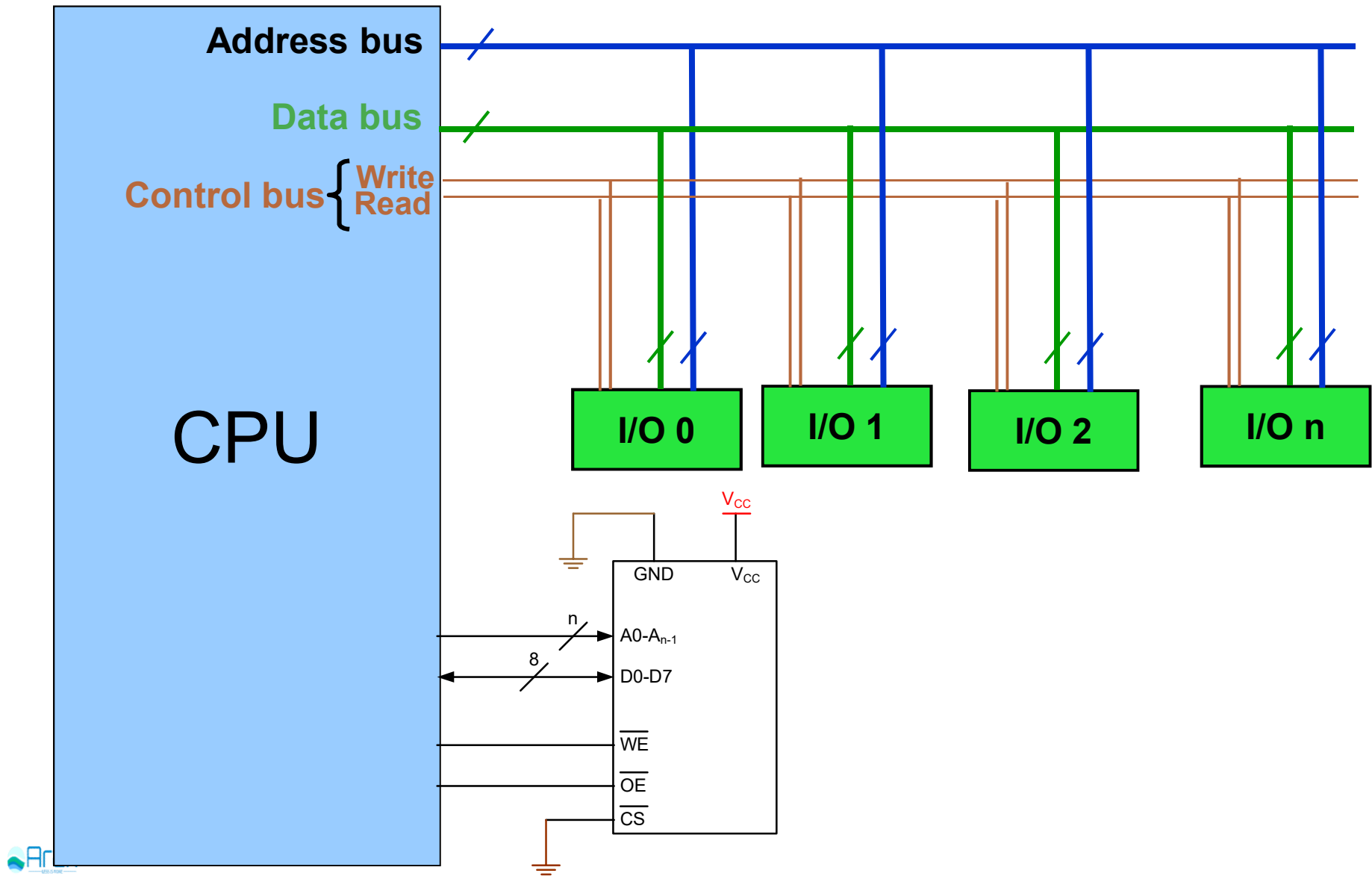


Connecting I/Os to CPU using bus

Bus Solution.

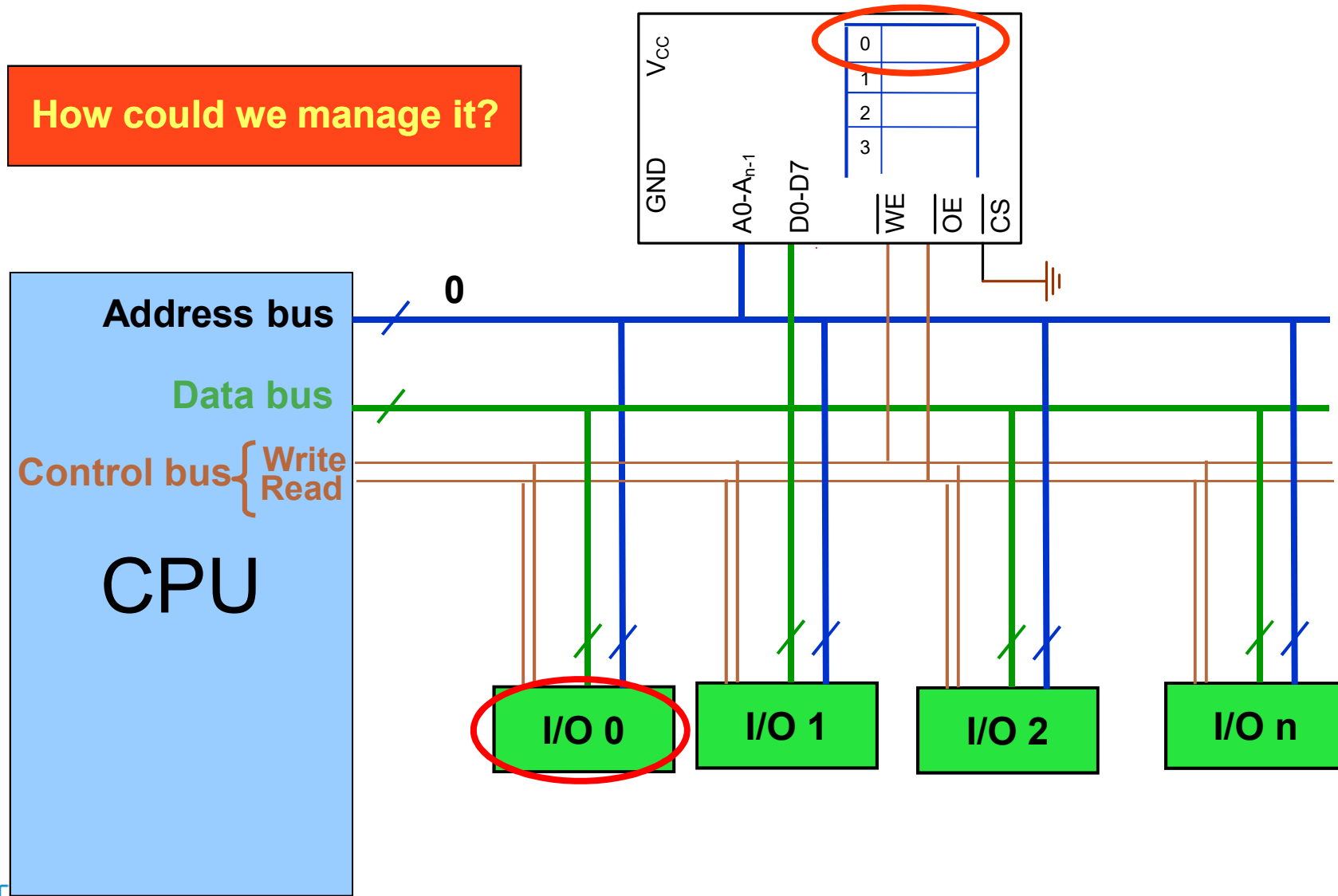


Connecting I/Os and Memory to CPU



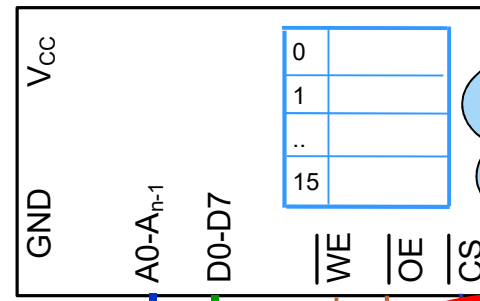
Connecting I/Os and memory to CPU using bus

How could we manage it?



Connecting I/Os and Memory to CPU using bus (Memory Mapped I/O)

How could we make the logic circuit?



The logic circuit enables CS when address is between 0 and 15

Logic circuit

Address bus

Solution

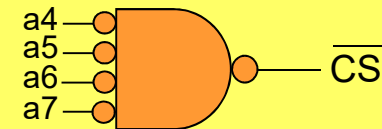
1. Write the address range in binary
2. Separate the fixed part of address
3. Using a NAND, design a logic circuit whose output activates when the fixed address is given to it.

From address 0 →

a7	a6	a5	a4	a3	a2	a1	a0
0	0	0	0	0	0	0	0

To address 15 →

a7	a6	a5	a4	a3	a2	a1	a0
0	0	0	0	1	1	1	1



Another example for address decoder

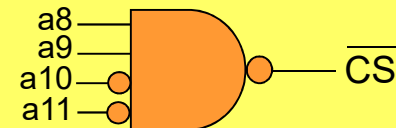
- Design an address decoder for address of 300H to 3FFH.

Solution

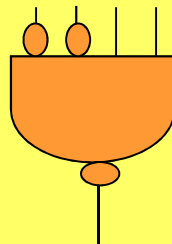
1. Write the address range in binary
2. Separate the **fixed part** of address
3. Design the logic circuit.

From address 300H →
To address 3FFH →

a11	a10	a9	a8	a7	a6	a5	a4	a3	a2	a1	a0
0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1



An easy way of
designing



Decoder 74LS138

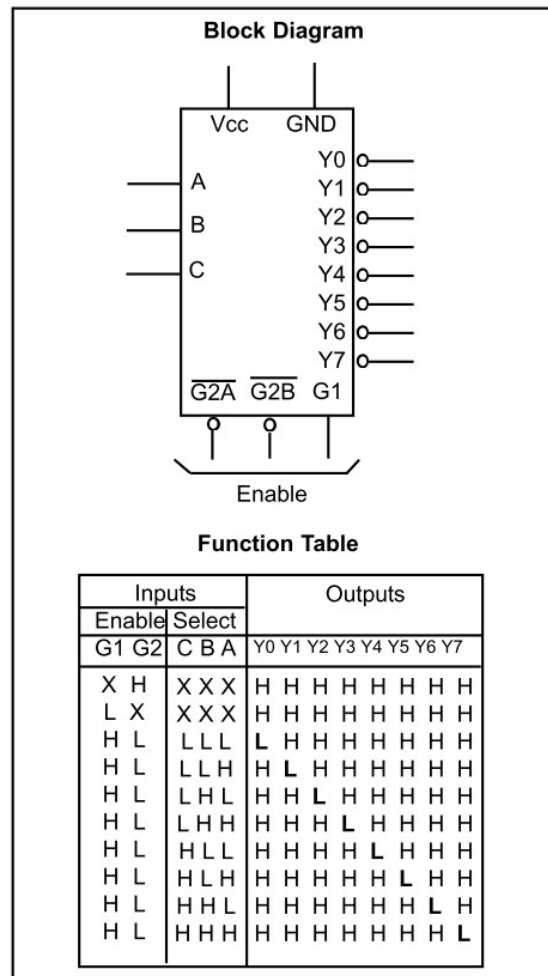


Figure 17. 74LS138 Decoder

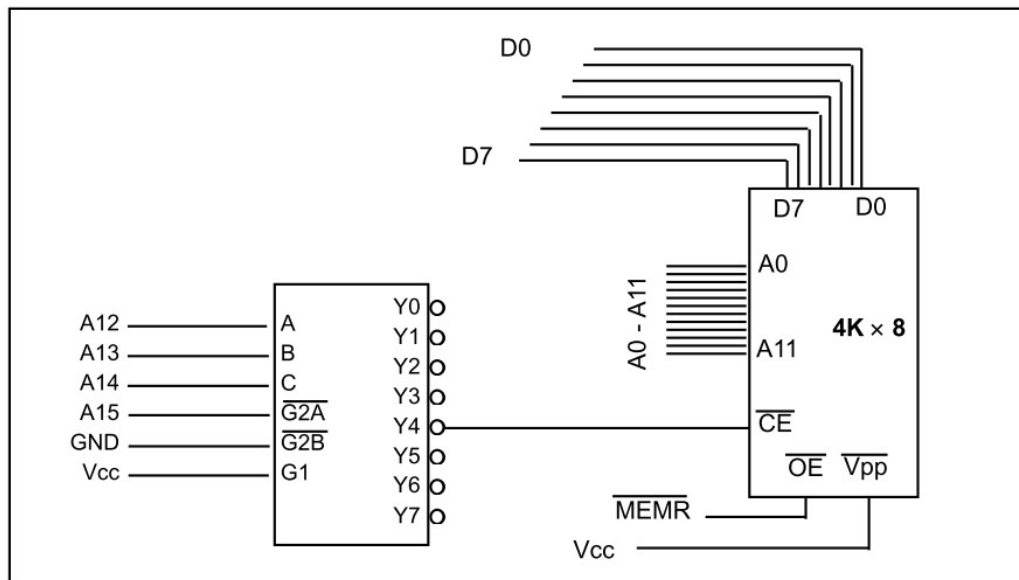


Figure 18. Using 74LS138 as Decoder

Example 16

Looking at the design in Figure 18, find the address range for the following:
(a) Y4, (b) Y2, and (c) Y7.

(a) The address range for Y4 is calculated as follows.

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

The above shows that the range for Y4 is 4000H to 4FFFH. In Figure 18, notice that A15 must be 0 for the decoder to be activated. Y4 will be selected when A14 A13 A12 = 100 (4 in binary). The remaining A11–A0 will be 0 for the lowest address and 1 for the highest address.

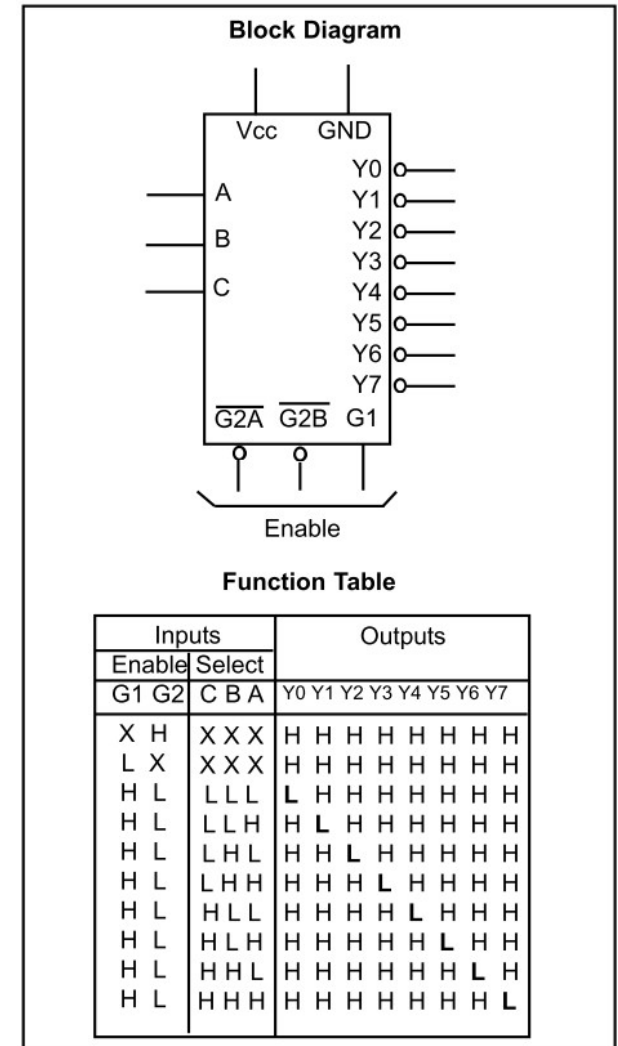
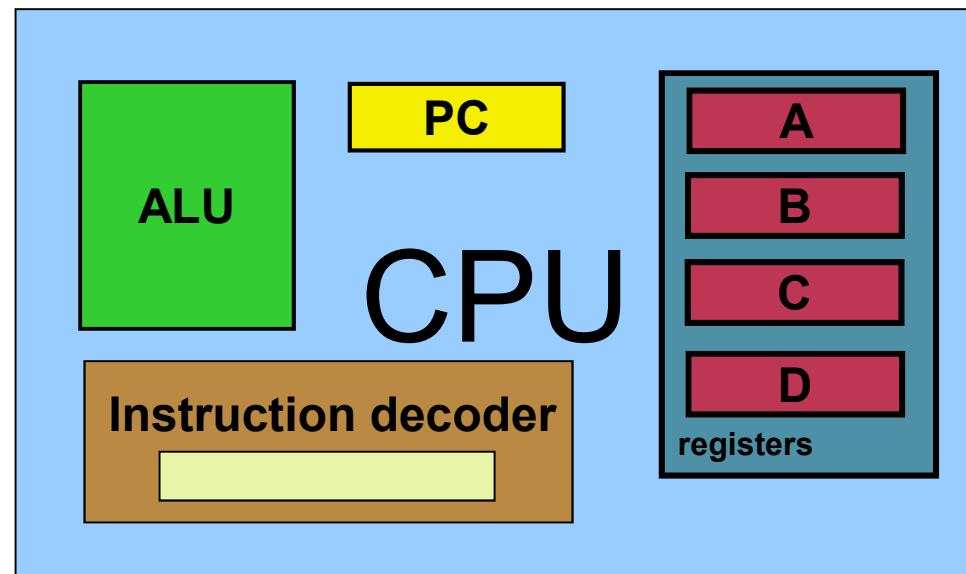


Figure 17. 74LS138 Decoder

Inside the CPU

- PC (Program Counter for next Instruction)
- Instruction decoder
- ALU (Arithmetic Logic Unit)
- Registers

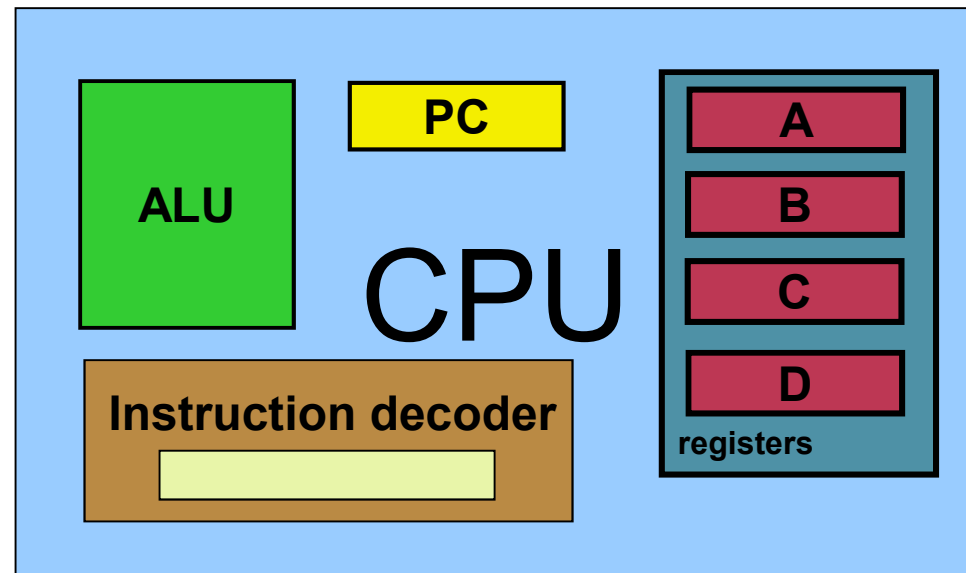
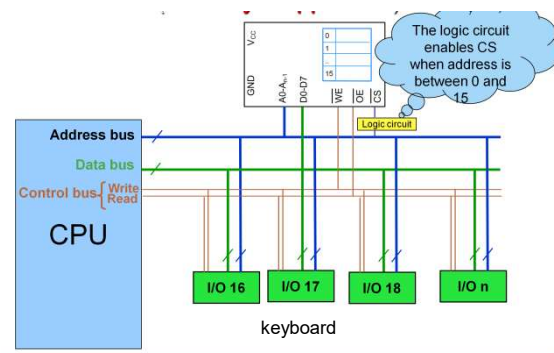


How computers work

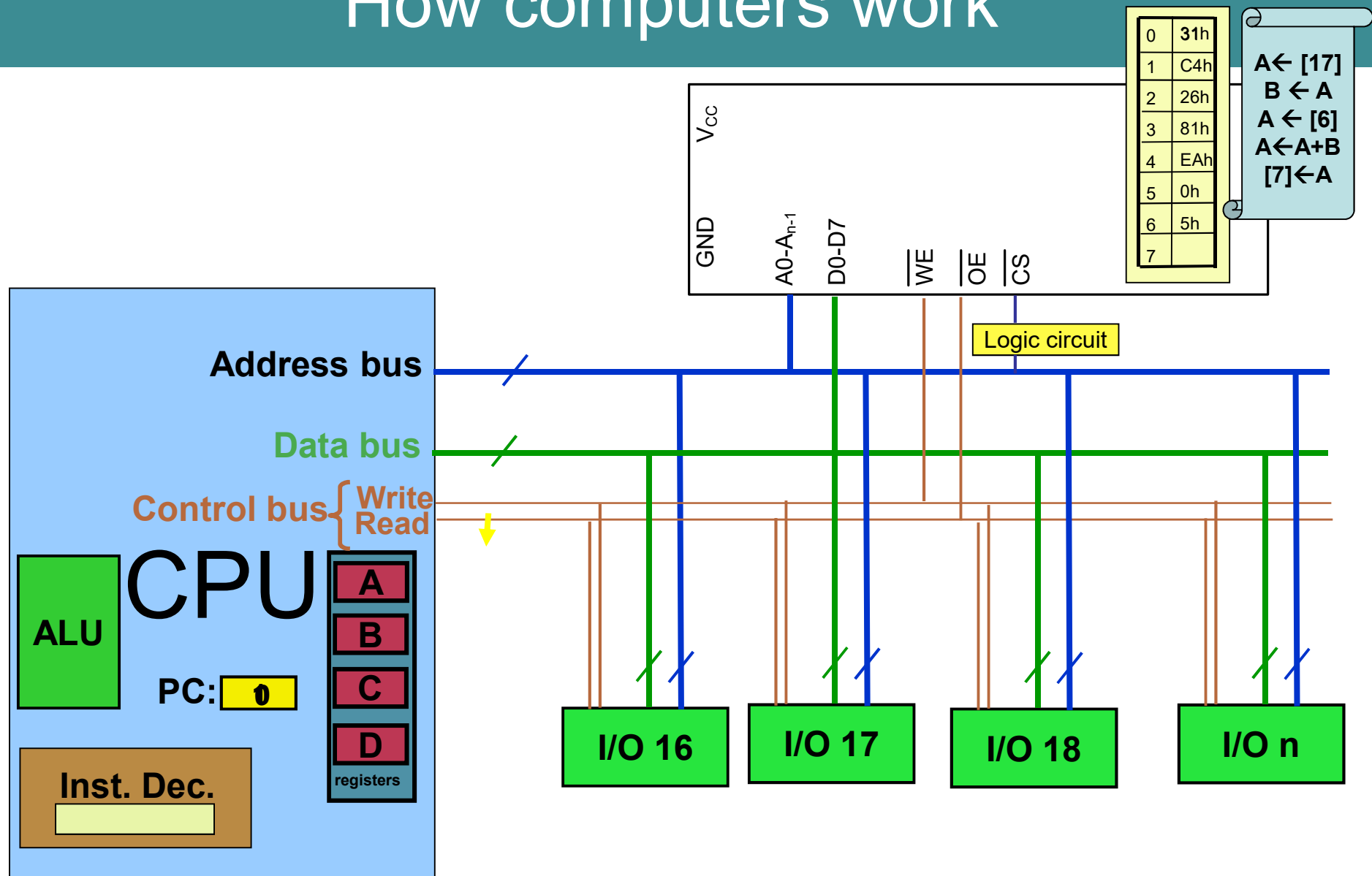
Goal

Memory([7])=I/O([17])+Memory([6])
(Just) A for external data interaction

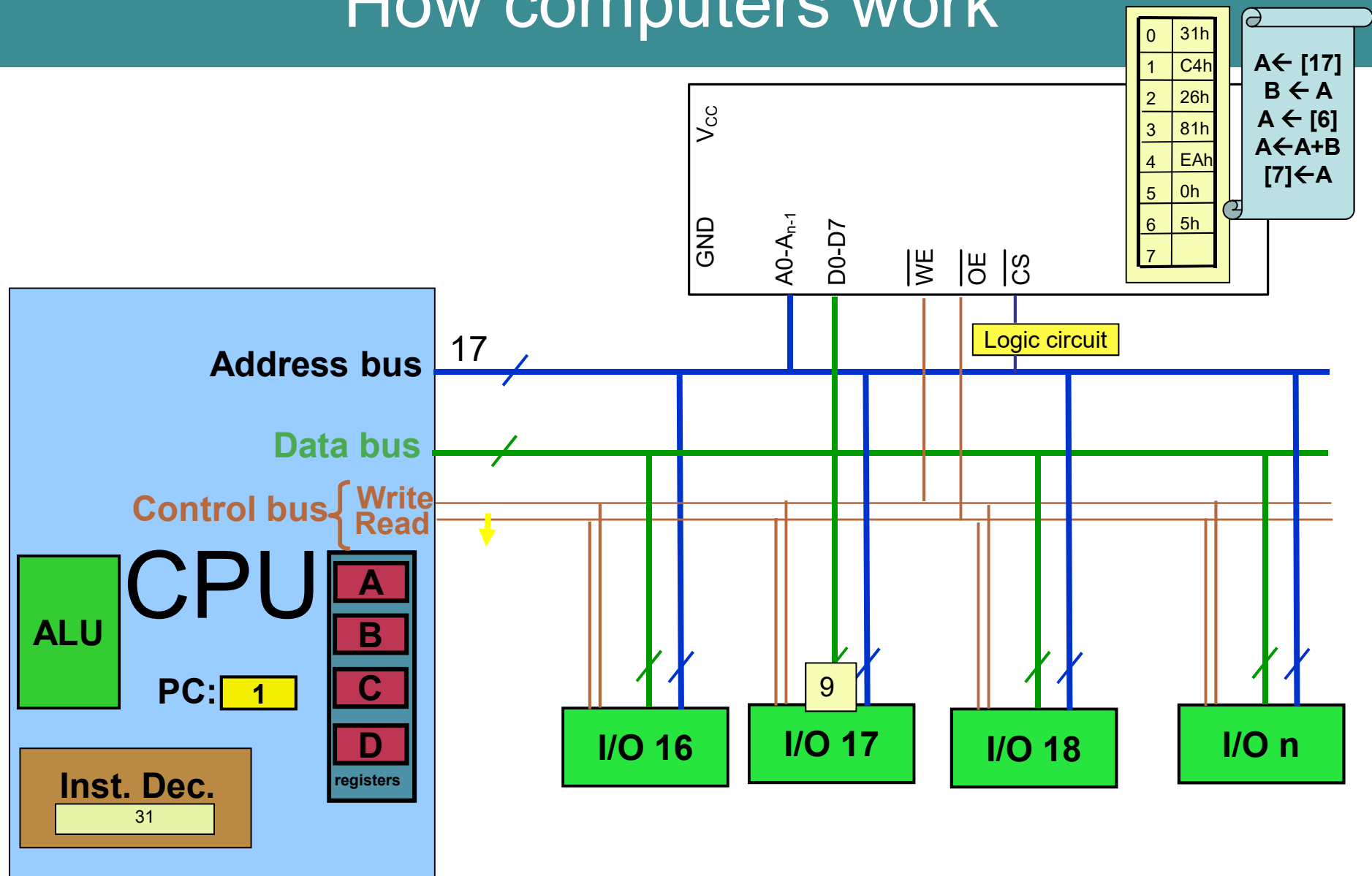
$A \leftarrow [17]$
 $B \leftarrow A$
 $A \leftarrow [6]$
 $A \leftarrow A+B$
 $[7] \leftarrow A$



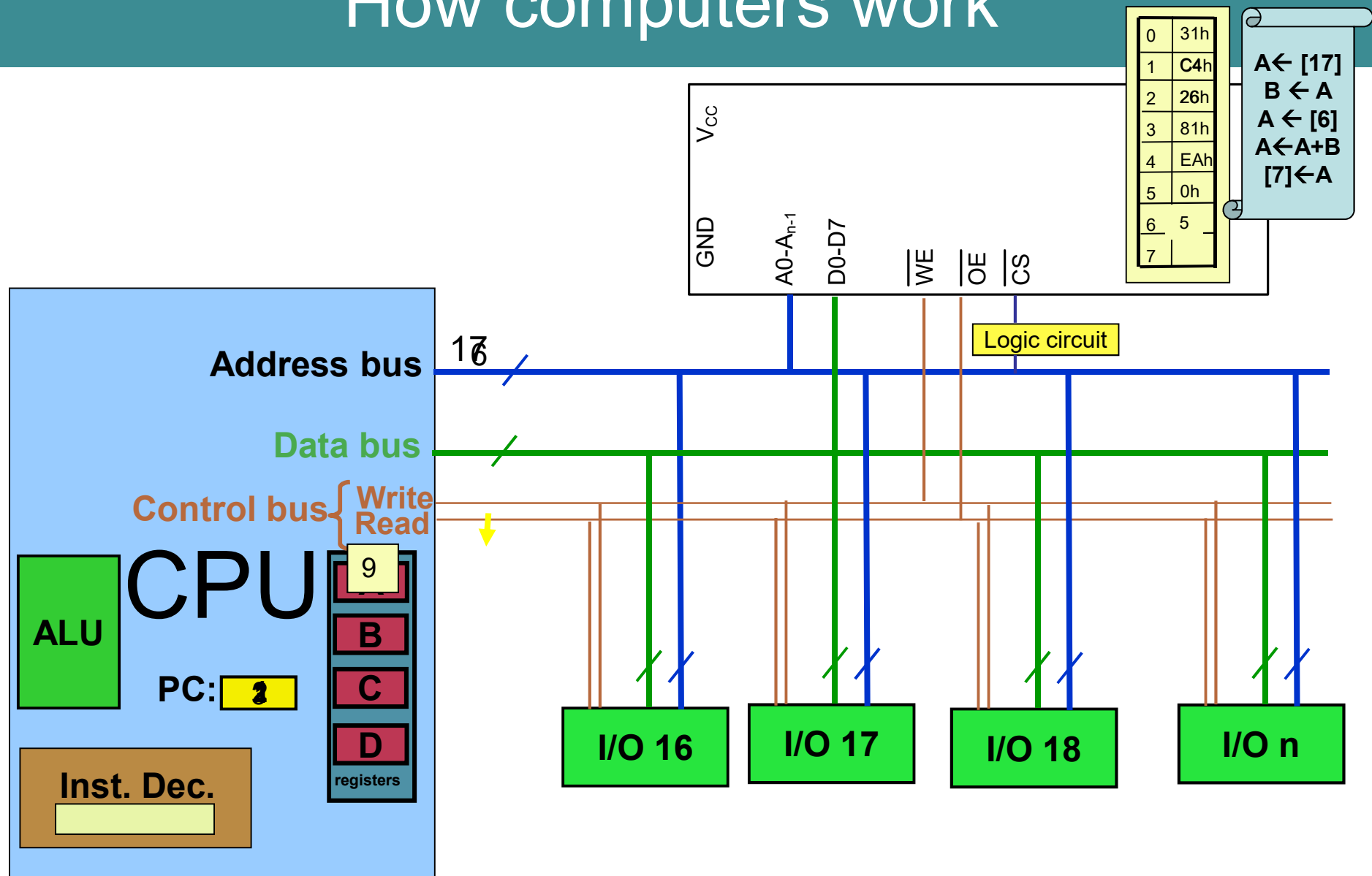
How computers work



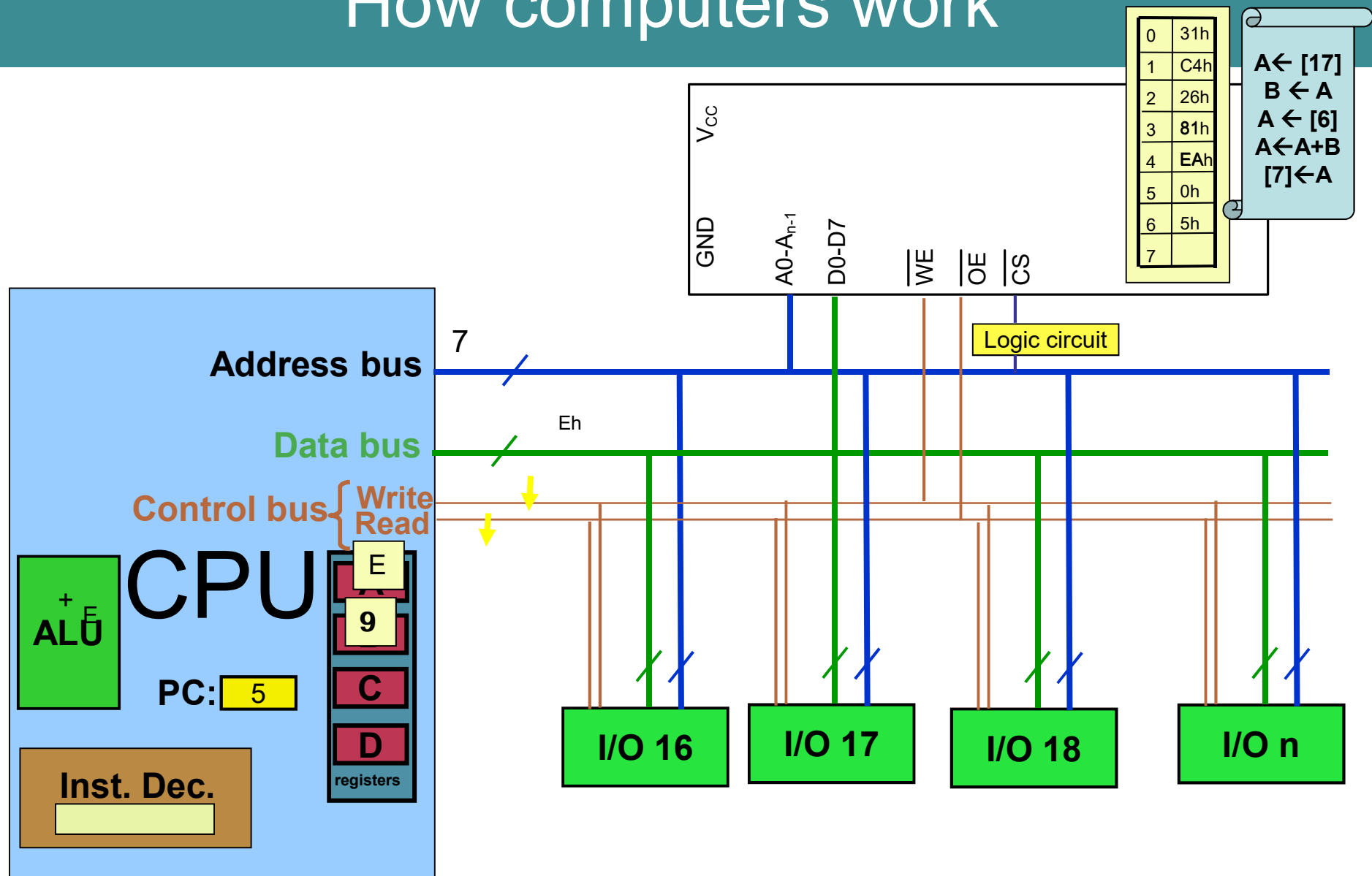
How computers work



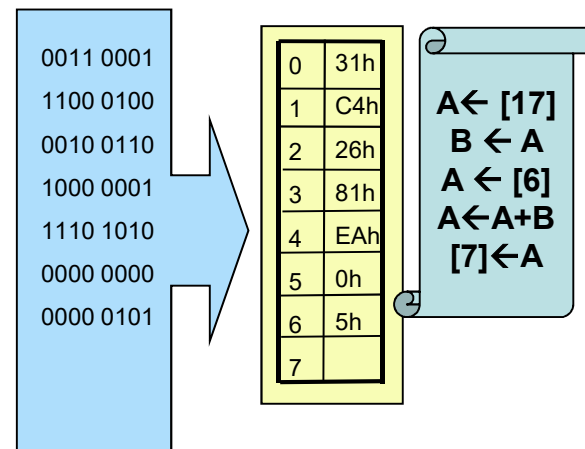
How computers work



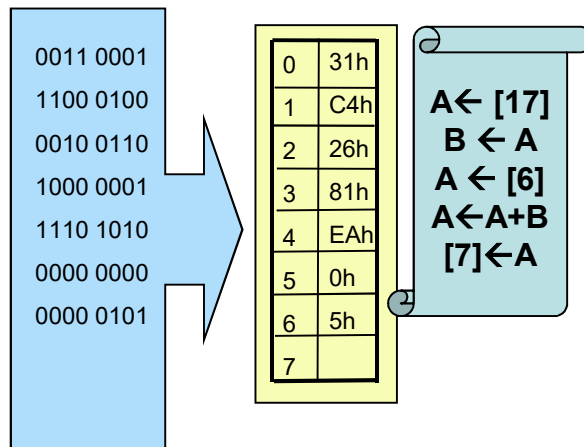
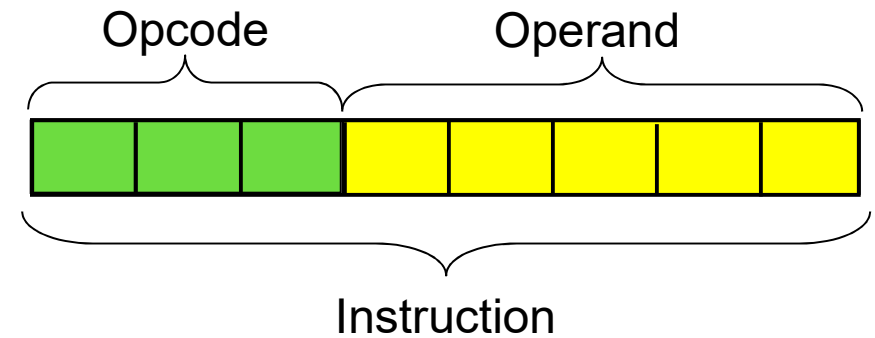
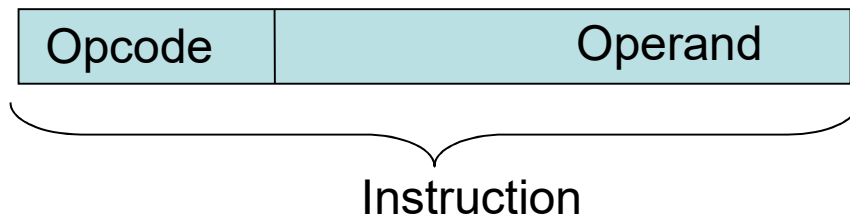
How computers work



How Instruction decoder works



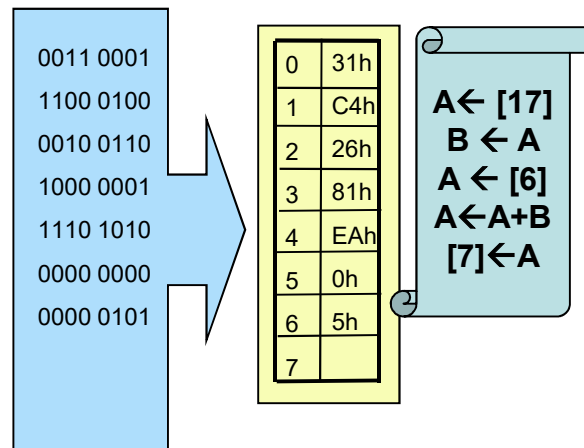
How Instruction decoder works



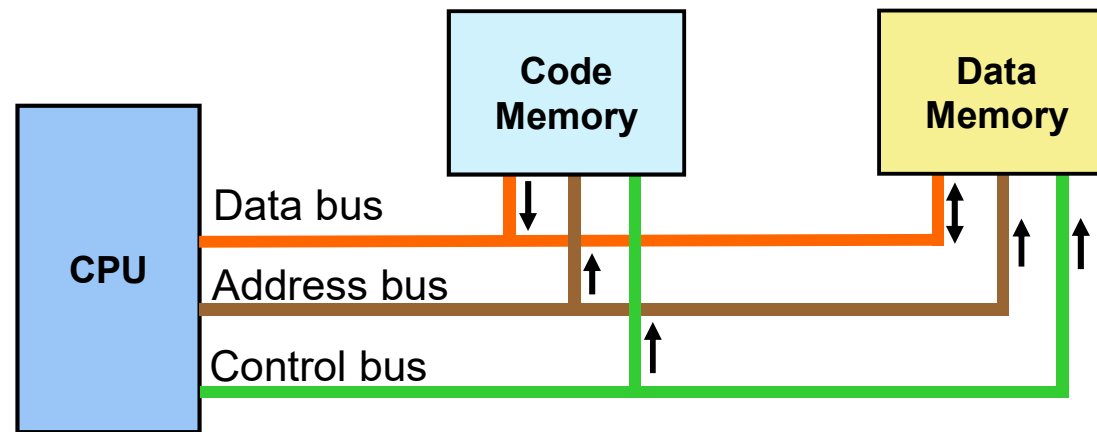
Operation Code	Meaning
000	$A \leftarrow x$
001	$A \leftarrow [x]$
010	$A \leftarrow A - \text{register}(x)$
011	$A \leftarrow A + x$
100	$A \leftarrow A + \text{register}(x)$
101	$A \leftarrow A - x$
110	$\text{Register}(x_H) \leftarrow \text{Register}(x_L)$
111	$[x] \leftarrow A$

Memory and CPU connections

- Memory includes
 - Data(what is?)
 - Codes

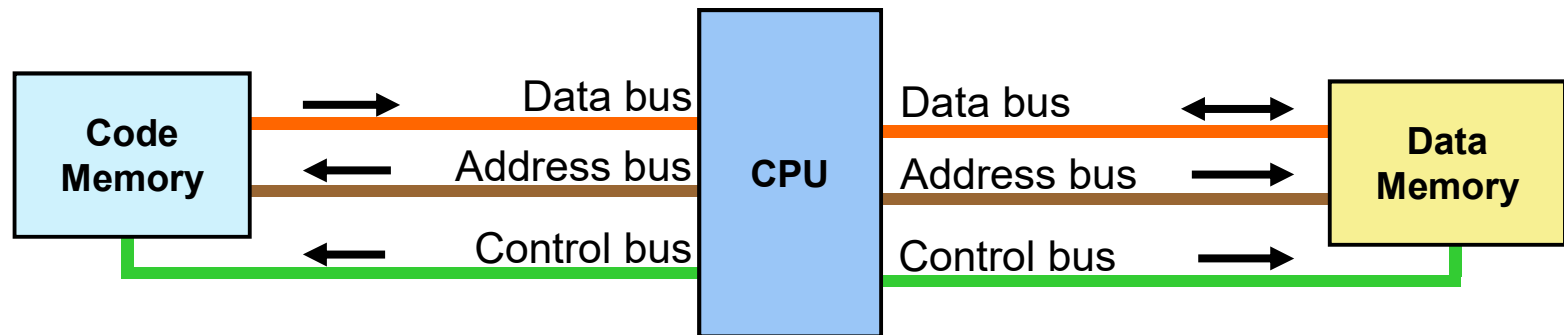


Von Neumann architecture

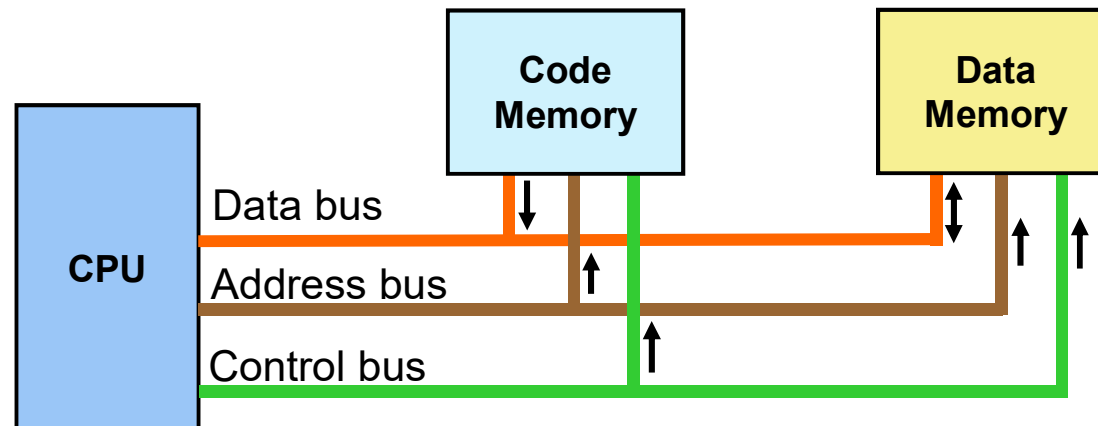


- How Access data and code *Simultaneously*?

Von Neumann vs. Harvard architecture



- Harvard architecture
- More Pin (Intel, AVR)



- Von Neumann architecture

video

- [How A CPU Works](#)
- DDR4 RAM

Video

- <https://youtu.be/Qu2njWY3Hjk>
- <https://youtu.be/efVS9l5RNYA>
- <https://youtu.be/OuFIISa73Sw>
- Videos
 - [History of Intel Processors](#)
 - [History Lost and Found - Mark I Computer](#)
 - [History Lost and Found - ENIAC](#)
 - [History Lost and Found - Transistor](#)
 - [History Lost and Found - Apple Computer](#)
 - [The Most Important Invention of the 20th Century: Transistors](#)