Microcontroller Prog. Basics Department of Instrumentation Engineering

| Lecture 2 |

Numbering and Coding Systems

A. Introduction

- ***Digital systems*** are built from circuits that process **binary digits - 0**s and **1**s, yet many real-life problems are based on numbers. Therefore we have to establish a correspondence between binary digits processed by digital circuits and real-life numbers, events and conditions.
- Since the binary number system is universally employed in digital computer, it is useful and necessary to understand the general properties of number systems and the methods of conversion from one to another.

A1. Basic Terms

Number System : a *code system* representing *quantity*

Base (Radix) : the base or radix of a number system refers to the *number of basic symbols* used.

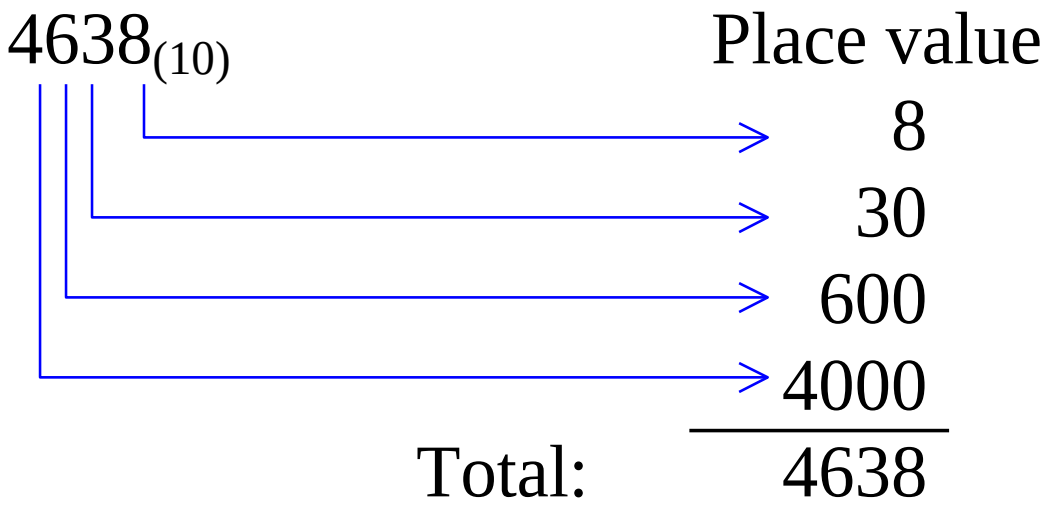
Digit : *a single basic symbol* used in a number system

Bit : bit is the abbreviated form of ***bi***nary digi***t***

Byte : an *entity* formed by combining *8 bits*

A2. Position Notation

Consider the decimal number *4638*:



Each ***digit*** carries different ***place value*** that affects the ***total value*** of a number.

B. Definition of a Number System

For a given number system, a quantity **N** with base **R** can be expressed as :

$$N_{(R)} = d_n R^n + d_{n-1} R^{n-1} + \ldots + d_2 R^2 + d_1 R^1 + d_0 R^0$$

where d_n is the digit of the corresponding position

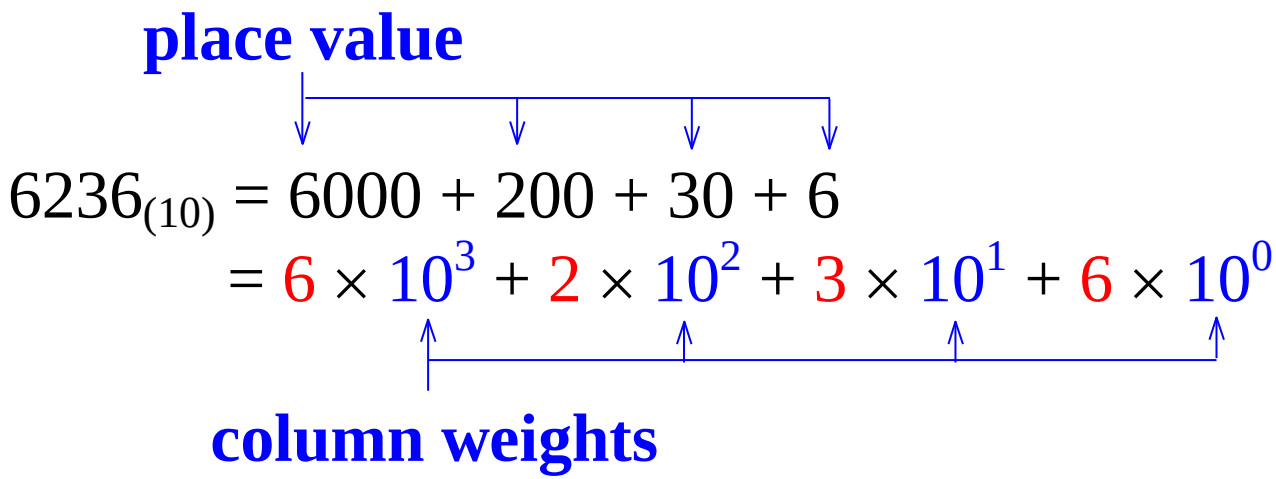
Example: **Decimal System**

base = 10; ie. 10 different digits or symbols

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

B. Definition of a Number System

Digits in different positions of a decimal number represent different value



B1. Binary Number System

base = 2

2 symbols : **0** and **1**

$$N_{(2)} = d_n 2^n + d_{n-1} 2^{n-1} + d_{n-2} 2^{n-2} + \ \ldots \ + d_2 2^2 + d_1 2^1 + d_0 2^0$$

Example 1: Determine the value of the binary number 11001.

$$11001_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

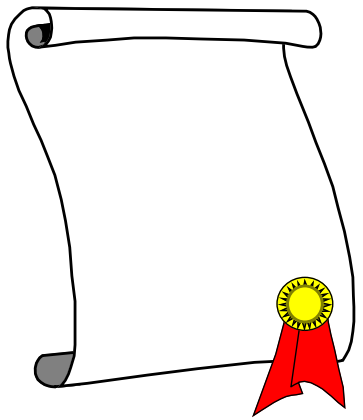
$$= 16 + 8 + 0 + 0 + 1$$

$$= \mathbf{25_{(10)}}$$ **… converted to decimal equivalent**

B1. Binary Number System

Classwork 1

Convert the following binary numbers into decimal:
(a) $1001100_{(2)}$; (b) $1101011_{(2)}$



B2. Octal Number System

base = 8

8 symbols : **0,1,2,3,4,5,6** and **7**

$$N_{(8)} = d_n 8^n + d_{n-1} 8^{n-1} + d_{n-2} 8^{n-2} + \ \ldots \ + d_2 8^2 + d_1 8^1 + d_0 8^0$$

Example 2: Convert $1357_{(8)}$ into decimal number.

$$1357_{(8)} = 1 \times 8^3 + 3 \times 8^2 + 5 \times 8^1 + 7 \times 8^0$$

$$= 512 + 192 + 40 + 7$$

$$= \mathbf{751_{(10)}}$$

B2. Octal Number System

Classwork 2

Convert the following octal numbers to decimal equivalent: (a) $76543_{(8)}$; (b) $666666_{(8)}$

B3. Hexadecimal Number System

base = 16

16 symbols : **0,1,2,3,4,5,6,7,8,9,A,B,C,D,E** and **F**

$$N_{(16)} = d_n 16^n + d_{n-1} 16^{n-1} + \ \ldots \ + d_2 16^2 + d_1 16^1 + d_0 16^0$$

Example 3: Convert $1357_{(16)}$ into decimal number.

$$1357_{(16)} = 1 \times 16^3 + 3 \times 16^2 + 5 \times 16^1 + 7 \times 16^0$$

$$= 4096 + 768 + 80 + 7$$

$$= \mathbf{4951}_{\mathbf{(10)}}$$

B3. Hexadecimal Number System

Classwork 3

Convert the following hexadecimal numbers to decimal:
(a) $ABCD_{(16)}$; (b) $F4240_{(16)}$

C. Convert Decimal Number to Other Radix

Divide the decimal integer progressively by the required radix, writing down the ***remainder*** after each division, the remainders taken in ***reverse order*** form the number of required radix.

C1. Convert Decimal to Binary

Example 4: Convert $43_{(10)}$ to binary

2	43	
2	21	1
2	10	1
2	5	0
2	2	1
	1	0

$43_{(10)} = \mathbf{101011_{(2)}}$

C1. Convert Decimal to Binary

Classwork 4

Convert the following decimal numbers to binary:

(a) $100_{(10)}$; (b) $1357_{(10)}$

C2. Convert Decimal to Octal

Example 5: Convert $2345_{(10)}$ into octal

8)	2345		
8)	293	____	1
8)	36	____	5
	4	____	4

$2345_{(10)} = \mathbf{4451}_{(8)}$

C2. Convert Decimal to Octal

Classwork 5

Convert the following decimal numbers to octal:

(a) $63440_{(10)}$; (b) $999999_{(10)}$

C3. Convert Decimal to Hexadecimal

Example 6: Convert $12345_{(10)}$ into hexadecimal

$$
\begin{array}{rrl}
16 \,)\, 1234 & & \\
16 \,)\, 771 & \text{—} & 9 \\
16 \,)\, 48 & \text{—} & 3 \\
3 & \text{—} & 0
\end{array}
$$

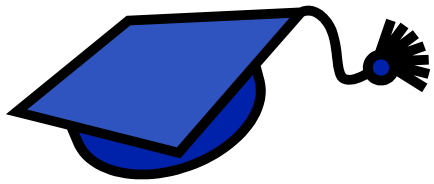
$12345_{(10)} = \mathbf{3039}_{\mathbf{(16)}}$

C3. Convert Decimal to Hexadecimal

Classwork 6

Convert the following decimal numbers to hexadecimal:

(a) $63440_{(10)}$; (b) $999999_{(10)}$



D. Binary System

Most digital computers store and manipulate data as **patterns of binary states**. The advantages of binary system over digital system with more than two states are:

a. Binary system has only **2 possible states**, either 0 and 1. They can easily be implemented by **physical devices** with 2 **stable** states; e.g. a switch with **On/Off** states, a transistor with **Saturation/Cutoff** states.

D. Binary System

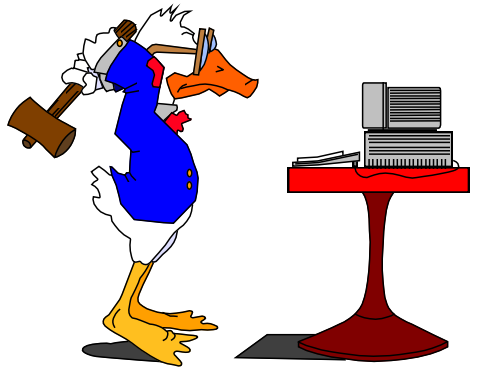
b. Practically, a **multi-state** device will be **expensive** and **sensitive to noise**. For binary device, a distorted pulse can easily be regenerated and the stability of the two possible states can easily be kept.

c. **Boolean Algebra** can be applied in binary system design.

d. Arithmetic and logical operations are simple both in hardware and software construction

D. Binary System

It should be mentioned that the binary number system is a convenient, but not the only way in manipulating the patterns of binary data. In some applications, such as **programming** and **I/O processes**, the ***octal*** and ***hexadecimal*** systems are also adopted because of their simplicity.



D1. Binary to Octal Conversion

As 8 is the third power of 2, we ***group the binary bits in threes***, starting at the **binary point**, then convert each group of three bits to its octal equivalent (**0s are added if necessary**)

Binary	Octal	Binary	Octal
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

D1. Binary to Octal Conversion

Example 8

Convert $110011101_{(2)}$ to octal

$110011101_{(2)} = \underline{110}\ \underline{011}\ \underline{101}_{(2)} = \mathbf{635_{(8)}}$

Classwork 8

Convert the following binary numbers to octal:

(a) $101010110000_{(2)}$; (b) $1110111011000101_{(2)}$

D2. Octal to Binary Conversion

It is the reverse of the binary to octal conversion, we merely convert one octal digit at a time to its binary equivalent.

Example 9

Convert $456_{(8)}$ to binary

$456_{(8)} = \underline{100}\ \underline{101}\ \underline{110} = \mathbf{100101110_{(2)}}$

Classwork 9

Convert the following octal numbers to binary:
(a) $7654_{(8)}$; (b) $12345_{(8)}$

D3. Binary to Hexadecimal Conversion

As 16 is the fourth power of 2, we ***group the binary bits in fours***, starting at the **binary point**, then convert each group of four bits to its hexadecimal equivalent (**0s are added if necessary**).

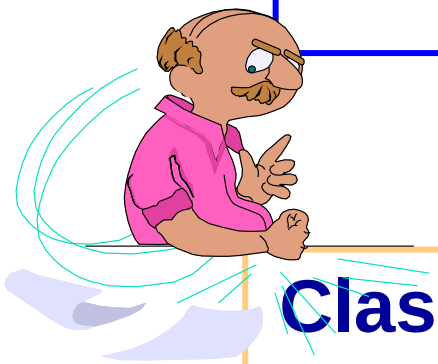
Binary	Hexadecimal	Binary	Hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

D3. Binary to Hexadecimal Conversion

Example 10

Convert $110001101111_{(2)}$ to hexadecimal

$110001101111_{(2)} = \underline{\mathbf{1100}}\ \underline{\mathbf{0110}}\ \underline{\mathbf{1111}} = \mathbf{C6F_{(16)}}$



Classwork 10

Convert the following binary numbers to hexadecimal:
(a) $101010110000_{(2)}$; (b) $110111011000101_{(2)}$

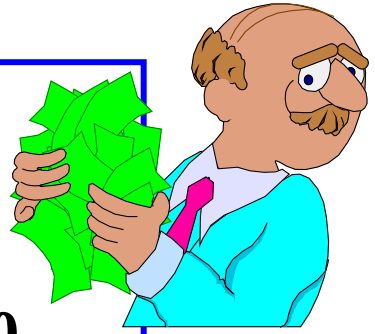
D4. Hexadecimal to Binary Conversion

It is the reverse of the binary to hexadecimal conversion, we merely convert one hexadecimal digit at a time to its binary equivalent.

Example 11

Convert $ABC_{(16)}$ to binary

$ABC_{(16)} = \underline{1010}\ \underline{1011}\ \underline{1100}_{(2)} = \mathbf{101010111100_{(2)}}$



Classwork 11

Convert the following hexadecimal numbers to binary:
(a) $CDEF_{(16)}$; (b) $12345_{(16)}$

E. Coding Systems

E1. Computer Words

- A word stored in a computer's memory can contain either of two kinds of information – ***instruction*** or ***data***
- Data can be numerical or character information that is processed by a program
- Instruction are executed by the CPU (Central Processing Unit) to complete special task

E2. Data / Information Representation

There are different types of information to be processed within computer. The most common types are:

- Numerical value
- Text
- Picture

When numbers, letters, or words are represented by a special group of symbols, we say that they are being encoded, and the group of symbols is called a ***code***.

E2. Data / Information Representation

For different types of information, there are different codes to represent them.

Each **b**inary dig**it** is known as a bit, and its combination are:

4-bit	nibble
8-bit	byte
16-bit	word
32-bit	long word, double word
64-bit	quad word

E2.1 Integer

- Usually coded in ***straight binary code***
- Arbitrary numbers can be represented with the digits 0 and 1, the minus sign (if any)
- However, in computer, multiple of 8 bits are commonly used
- ***8-bit*** represent 0 to 255 for unsigned, –128 to +127 for signed number
- ***16-bit*** represent 0 - 65535 for unsigned, –32768 to +32767 for signed number
- ***Sign-magnitude*** or ***Two's complement*** can be used to represent signed number

E2.2 Binary-Coded-Decimal (BCD)

- ***Straight binary code*** is used within digital system, while ***decimal code*** is used in real world in nature. Hence, conversion between decimal and binary code has to be carried out often
- ***BCD*** is used with each digit of a decimal number is represented by its binary equivalent
- Since a decimal digit can be as large as 9, four bits are required to code each digit

E2.2 Binary-Coded-Decimal (BCD)

Example 12

Convert 928_{10} to BCD code.

9	2	8
↓	↓	↓
1001	0010	1000

$928_{10} = \mathbf{100100101000_{BCD}}$

E2.2 Binary-Coded-Decimal (BCD)

Example 13

Convert $0001\ 0111\ 0011_{BCD}$ to decimal code.

0001	0111	0011
↓	↓	↓
1	7	3

$000101110011_{BCD} = \mathbf{173_{10}}$

E2.3 Alphanumeric Codes

- Computer must be able to handle non-numerical information.
- Sometimes refer as ***text***, ***character***, ***character string*** or ***text string***
- For every letter, "**a**" to "**z**", upper and lower case, numeric digits, symbols, punctuation marks and control characters, they are represented by a 7-bit code
- The most common code system used is ***ASCII*** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)

E2.3 Alphanumeric Codes

ASCII Table

Least significant nibble		Most significant nibble							
		0	1	2	3	4	5	6	7
		0000	0001	0010	0011	0100	0101	0110	0111
0	0000	NUL	DLE	SP	0	@	P	‘	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	“	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	’	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL

E.g. ASCII code of “A” = 41H and “a” = 61H

E2.4 Parity Method for Error Detection

- Electrical noise will cause transmission errors when information is sent from a place to another place.
- Use parity bit is one of the simplest error detection method and most widely used in computer system.
- Parity bit is an extra bit that is attached to a code group that is being transferred from one location to another. The parity bit is made either 1 or 0.
- Two different scheme:
  - Even-parity : The value of parity bit is chosen to give an even nos. of 1s in the code group (including the parity bit).

E2.4 Parity Method for Error Detection

- Even-parity:

  1 1000 0011

  ↑

  added parity bit = 1 to make even nos. of 1s, otherwise parity bit = 0

- Odd-parity : The value of parity bit is chosen to give an odd nos. of 1s in the code group (including the parity bit).

  1 1001 0011

  ↑

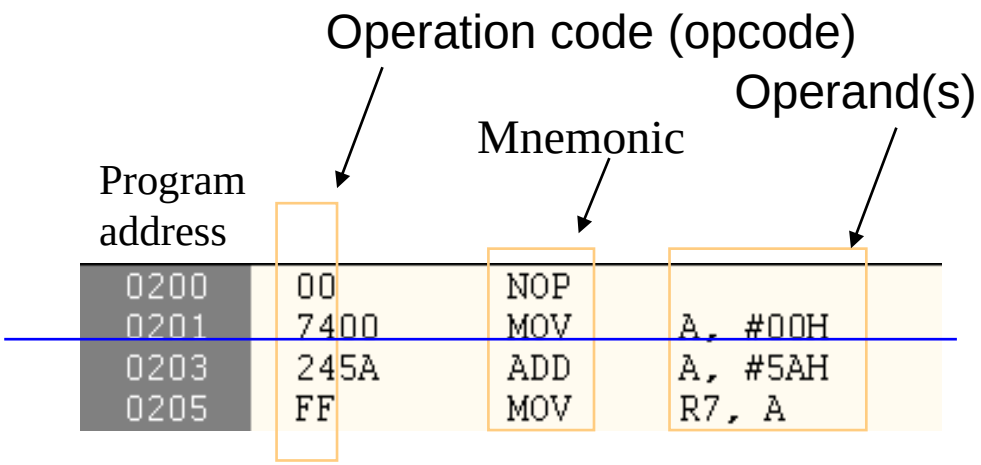
  added parity bit = 1 to make odd nos. of 1s, otherwise parity bit = 0

E2.5 Computer Instructions

- These are the word which contain the information necessary for a computer to execute its various operations
- The format and codes for these can vary widely from computer to computer
- Instruction is consisted of two parts: ***opcode*** (operation code) and ***operand*** (data to be operated on)

E2.5 Computer Instructions

- ***Op code*** is used to specify what operation to be executed by the CPU
- ***Operand*** is used to specify which/what data to operate on and we can have zero , one and multiple operands
- Instruction can be of 1-byte, 2-byte and multi-byte in length
- E.g. For 1-byte long op code, there are $2^8 = 256$ possible operations to be carried out



Instruction Set of 8051 Microcontroller

Mnemonics, Arranged Alphabetically

MNEMONIC	DESCRIPTION	BYTES	CYCLES	FLAGS
ACALL addr11	PC + 2 → (SP); addr11 → PC	2	2	
ADD A, direct	A + (direct) → A	2	1	C OV AC
ADD A, @Ri	A + (Ri) → A	1	1	C OV AC
ADD A, #data	A + #data → A	2	1	C OV AC
ADD A, Rn	A + Rn → A	1	1	C OV AC
ADDC A, direct	A + (direct) + C → A	2	1	C OV AC
ADDC A, @Ri	A + (Ri) + C → A	1	1	C OV AC
ADDC A, #data	A + #data + C → A	2	1	C OV AC
ADDC A, Rn	A + Rn + C → A	1	1	C OV AC
AJMP addr11	Addr11 → PC	2	2	
ANL A, direct	A AND (direct) → A	2	1	
ANL A, @Ri	A AND (Ri) → A	1	1	

E2.5 Computer Instructions

Classwork 16

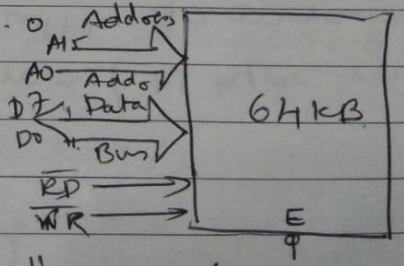
1. What information is contained in the first byte of a multi-byte instruction?
2. How many possible operations a CPU can have if it used 2 bytes for the op code?

Bit → binary digit that have value 0 or 1
Byte → 8 bit
Nibble → 4 bit
word 2 byte – 16 bit.

Bit 0
Nibble 0000
Byte 0000 0000
word 0000 0000 0000 0000
0 0 0 0
F F F F

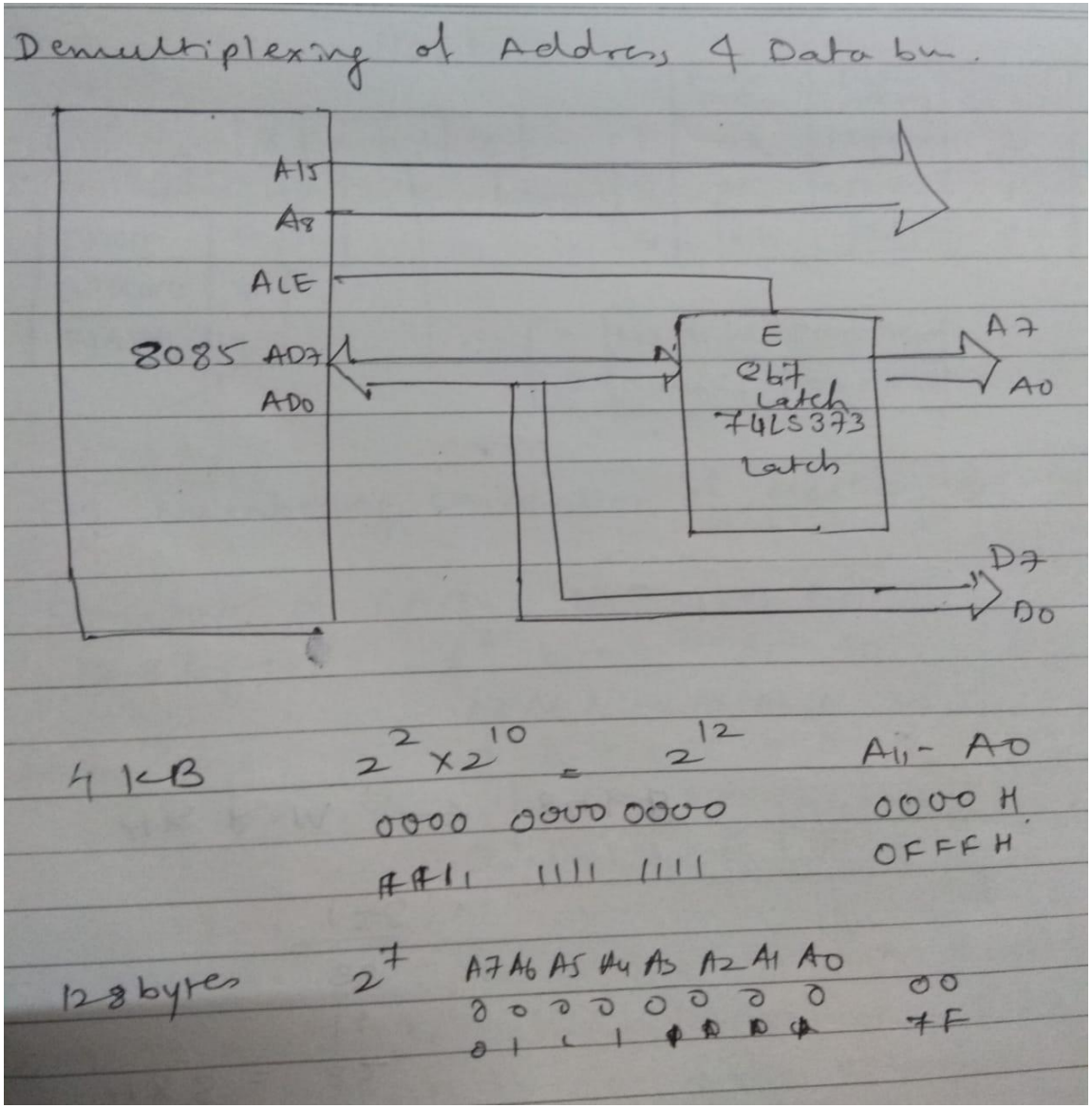
1KB – 2^{10} bytes = 1024 bytes.
eg. 2KB 64KB $2^6 \times 2^{10} = 2^{16}$.
4KB 65,536 M.L

A15 A0
0 0



1 megabyte = 2^{20} byte.
1 GB = 2^{40} bytes

2KB = $2^1 \times 2^{10} = 2^{11}$ A10 – A0


Demultiplexing of Address & Data bus.
A15
A8
ALE
8085 AD7
AD0
E
Q67
Latch
74LS373
Latch
A7
A0
D7
D0
4 KB
$2^2 \times 2^{10} = 2^{12}$
$A_{11} - A_0$
0000 0000 0000
0000 H
1111 1111 1111
0FFF H
128 bytes
2^7
A7 A6 A5 A4 A3 A2 A1 A0
00
7F

🕮 Read reference

- The 8051 Microcontroller and Embedded Systems - Using Assembly and C, Mazidi
  - Chapter 0 P.1 – P.6