COMPUTER AND COMMUNICATION TECHNOLOGY

DATA REPRESENTATION

BINARY NUMBER SYSTEM

BIT

- A binary digit is called a bit.
- It is usually expressed as 0 or 1, the two numbers of binary numbering system.
- A bit is a smallest unit of information a computer can use.
- A 16 bit computer would process a series of 16 bits such as 0100111101011000 in one go, repeating the process thousands or millions of times per second.
- Reading a series of bits is very difficult and to make the process easier they are often displayed in a group of 4 bits 0100 1111 0101 1000.
- This grouping is quite interesting in that a group of four bits can be replaced by a single hexadecimal digit, two groups of 4 bits can be replaced by 2 hexadecimal digits and 4 hexadecimal digits are required to replace 16 bits.

Binary	0100	1111	0101	1000
hexadecimal	4	F	9	8

BYTE

- A group of 8 bits is called a byte.
- With 8 bits, there are 256 possible decimal combinations.
- 1 byte can store one alphabetical letter, single digit or single character symbol such as #.
- Large number of bytes can be expressed in kilobytes and megabytes.

KILOBYTE

- The value of a kilobyte is 1024.
- It is worked out as 2^{10}.
- Normally a kilo refers to a thousand but in computing a kilobyte is equal to 1024.

MEGABYTE

- Likewise 1024 kilobytes is referred to as a megabyte.
- Normally a Mega refers to a million.
- In computing a megabyte is 1,048,576 bits.
- It is worked out as 2^{20}.

- A byte of memory can normally hold:

  i. a single alphabet letter (upper or lower case).
  ii. A single number 1 – 9.
  iii. A symbol (, _, +, $, #, > etc.
  iv. A further 127 alternate characters. These could be letters used in foreign languages, lines to produce a box etc.

NUMBER SYSTEM

- The everyday number system we use is denary or decimal number system.
- In computing three number system are commonly used , binary, hexadecimal and to a lesser extent octal.
- In denary, binary, octal and hexadecimal systems, the value of any digit depend on its position within the number i.e. which column it is in

How do Numbering Systems Work

- To understand this we will examine the denary system in more detail.
- Because you are used to the denary system and because it is very easy to multiply, 10 100 etc, you calculate the number in your head.
- Let us use the number 256 as an example.
- The calculation is automatically done in the following:
  - The most important calculation to do is to work out the position value for that system.
  - The position value is based on the powers of the number system base value.

Power of the base	10^4	10^3	10^2	10^1	10^0
calculation	10x10x10x10	10x10x10	10x10	10	1
=position value	10,000	1000	100	10	1

- Write down the position value for the number system you are using so for denary we would write
- Position value 10,000 1000 100 10 1.
- Underneath the correct value, write down your number
- Position value 10,000 1000 100 10 1
- Enter number 2 5 6
- The calculation that is done is
- Position value 10,000 1000 100 10 1
- Enter number 2 5 6
- Required calculation 2x100 5x10 6x1
- This equals 200 50 6
- Add the three results 200 + 50 + 6 = 256

- You can convert any number to denary system using this calculation.
- Ensure you use the positional number of the system you are using.
- A byte of memory can store number in the range of 0 to 255.
- Denary number system uses numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 thus 10 numbers.
- Hence base 10
- Adding 1 to 9 we must introduce another column to the left i.e. 10.

Binary Number System (base 2)

- Uses a 0 and a 1
- Hence base 2.
- Binary numbering is the numbering system that is used by computers.
- The positional values are shown below.

- Power of the base 2^4 `2^3 2^2 2^1 2^0
- Positional values 16 8 4 2 1

- A byte of memory can store a number in the range of 00000000 to 11111111.
- Numbers are often displayed in groups of 4s, as follows, to make them easier to read: 0000 0000 to 1111 1111.

How can you Convert from Binary to Denary

- For example convert 1011 binary to denary.
- Position value 16 8 4 2 1
- Enter number 1 0 1 1
- Required calculation 8x1 4x0 2x1 1x1
- This equals 8 0 2 1
- Add the three results 8 + 0 + 2 + 1 = 11.
- Therefore $1011_2 = 11_{10}$

Hexadecimal Numbering System (Base 16)

- Uses numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- Hence 16 numbers and base 16.
- By using letters A – F , a single digit requires a single position (column)
- Power of the base 16^4 16^3 16^2 16^1 16^0
- Positional value 65536 4096 256 16 1
- A byte of memory can store the number in the range of 00 to FF.
- Hexadecimal number are of displayed in the group of two to make them easier to read.
- For example 10 AF 3C 9F.

Octal (Base 8)

- Uses numbers 0, 1, 2, 3, 4, 5, 6, 7,
- Thus 8 numbers
- Hence base 8.
- A byte of memory can store an octal number in the range of 0 to 377.

- Power of the base 8^4 8^3 8^2 8^1
- Position value 4096 512 64 1

Binary Addition

0 + 0 = 0,   0 + 1 = 1,   1 + 0 = 1,   1 + 1 = 10.

Consider the following:

1 1 0	1 1	1 0 0	1 1	1 0 1 0	1 1 1 1 1
+ 0 0 1	+ 1 0	+1 0 1	+0 1	+0 1 1 1	+ 0 1 0 1 1
1 1 1	1 0 1	1 0 0 1	1 0 0	1 0 0 0 1	1 0 10 1 0

Exercise:

Perform the following binary additions

a. 1 0 0 1	b. 1 1 1 0	c. 1 0 1 0 1	d. 1 1 0 1 1 0
+1 1 0 0	+ 1 1 0 1	+ 0 0 1 1 1	+ 1 1 1 0 1 1

Subtracting Using Complement

- Subtraction in any number system can be done through the use of a complement.
- A complement is a number that is used to represent the negative of a number.
- When two numbers are subtracted, the complement of a subtrahend can be added to a minuend to obtain the difference.
- When this method is used, the addition will produce a higher order (left most) one in the result (a “carry”) which must be dropped.
- This is how a computer performs subtraction.

- To understand the complements consider a mechanical register such as mileage indicator being rotated backwards.
- A five digit register approaching and passing through 0 would read as follows
- 00005
- 00004
- 00003
- 00002
- 00001
- 99999
- 99998
- 99997
- Etc

- It should be clear that the number 99998 correspond to -2.
- Further if we add
- 00005
- +99998
- 100003
- And ignore carry to the left, we have effectively formed an operation of subtraction 5 – 2 = 3.
- The number 99998 is called a tens complement of 2.
- The tens complement of any decimal number may be formed by subtracting each digit from the number 9, then adding 1 to the least significant digit of the number formed.
- In the example above:

1. First each digit of subtrahend was subtracted from 9 (this preliminary value is called nines complement of the subtrahend

9	9	9	9	9
-0	-0	-0	-0	-2
9	9	9	9	8

2. We now add 1 to the nines complement to get the tens complement.

9	9	9	9	7
+				1
9	9	9	9	8

3. The tens complement was added to the minuend giving 100003. The leading carry was dropped, effectively performing subtraction 00005 – 00002 = 00003.

```
  0 0 0 0 5
+ 9 9 9 9 8
1 0 0 0 0 3
```

- Another example consider 4589 – 322.

1. First compute the nines complement

9	9	9	9
-0	-3	-2	-2
9	6	7	7

2. Add 1 to nines complement

   9 6 7 7

   + 1

   9 6 7 8

3. Add tens complement of subtrahend to minuend giving

   4 5 8 9

   +9 6 7 8

   1 4 2 6 7

   Drop the leading 1 and the answer is 4267.

   Exercise:

   Try the following problems:

   a. 5086 – 2993 b. 8391 – 255.

Binary Subtraction

- We use complement method to compute subtraction in binary.
- Steps:
  1. Compute 1s complement – subtract each digit of subtrahend from 1. A short cut for doing this is simply to reverse the digits.
  2. Add 1 to the 1s complement of the subtrahend to get 2s complement.
  3. Add 2s complement of subtrahend to minuend and drop the higher order 1. This is your difference.

- Example : compute 11010101_2 - 1001011_2

1.

1	1	1	1	1	1	1	1
-0	-1	-0	-0	-1	-0	-1	-1
1	0	1	1	0	1	0	0

2.

1	0	1	1	0	1	0	0
+							1
1	0	1	1	0	1	0	1

3.

1	1	0	1	0	1	0	1
+1	0	1	1	0	1	0	1
1 1	0	0	0	1	0	1	0
1	0	0	0	1	0	1	0

- Example 1 1 1 1 10 1 1_2 – 1 1 0 0 0 0 0 1_2

1. Original number 1 1 0 0 0 0 0 1

   1s complement 0 0 1 1 1 1 1 0
2. Add 1 to get the 2s complement

   0 0 1 1 1 1 1 0

   +   1

   0 0 1 1 1 1 1 1
3. Add 2s complement of the subtrahend to minuend.

   1 1 1 1 1 0 1 1

   +0 0 1 1 1 1 1 1

   1 0 0 1 1 1 0 1 0
4. Ignore the carry the answer is 0 0 1 1 1 0 1 0

   Exercise: carry out the following subtractions

   a. 1 1 1 1 1 0 1 1_2 – 1 0 1 0 1 0 1 0_2 b. 1 0 0 1 0 0_2 – 1 1 1 0 1_2

Octal Addition

- Octal addition is performed like decimal addition
-
- Example $543_8 + 121_8$
- 5 4 3
- +1 2 1
- 6 6 4
-

- Example $7652_8 + 4574_8$
- 7 6 5 2
- + 4 5 7 4
- 1 4 4 4 6
- 12/8 12/8 12/8

Exercise:

a. $5430_8 + 3241_8$ b. $6405_8 + 1234_8$

Octal Subtraction

- Example: compute $7526_8 - 3142_8$
- 7 5 2 6
- -3 1 4 2
- 4 3 6 4

- Example: compute $545_8 - 14_8$
- 5 4 5
- - 1 4
- 5 3 1

Hexadecimal Addition

- One consideration is that when the result of addition is between 10 and 15, a corresponding letter A through F must be written in the result.
-
- Example: $195_{16} + 319_{16}$
- 1 9 5
- +3 1 9
- 4 A E

- Example: $3A2_{16} + 41C_{16}$
- 3 A(10) 2
- <u>+4 1 C(12)</u>
- 7 B E
-
- Example: $DEB_{16} + 10E_{16}$
- D(13) E(14) B(11)
- <u>+1 0 E(14)</u>
- E F 9
-
- Example: $8F97_{16} – D54C_{16}$
- 8 F(15) 9 7
- <u>+ D(13) 5 4 C(12)</u>
- 1 6 4 E 3

Exercise: compute the following:

a. $BED_{16} + 2A9_{16}$ b. $DEED_{16} + BEEF_{16}$

Hexadecimal Subtraction

- Example: compute $ABED_{16} – 1FAD_{16}$
- A(10) B(11) E (14) D(13)
- <u>- 1 F(15) A(10) D(13)</u>
- 8 C 4 0
-
- Example: compute $FEED_{16} – DAF3_{16}$
- F(15) E(14) E(14) D(13)
- <u>- D(13) A(10) F(15) 3</u>
- 2 3 F A
- Exercise: compute: a. $98AE_{16} – 1FEE_{16}$ b. $B6A1_{16} – 8B12_{16}$

Number System Conversion

- *Converting a binary number to a decimal number*.

-
- Example: convert 1001_2 to decimal.

$1\ 0\ 0\ 1_2$

$1 \times 2^0 = 1 \times 1 = 1$

$0 \times 2^1 = 0 \times 2 = 0$

$0 \times 2^2 = 0 \times 4 = 0$

$1 \times 2^3 = 1 \times 8 = \underline{8}$

9_{10}

- Example: convert 1101010 to decimal
- $1\ 1\ 0\ 1\ 0\ 1\ 0_2$
- $0 \times 2^0 = 0 \times 1 = 0$
- $1 \times 2^1 = 1 \times 2 = 2$
- $0 \times 2^2 = 0 \times 4 = 0$
- $1 \times 2^3 = 1 \times 8 = 8$
- $0 \times 2^4 = 0 \times 16 = 0$
- $1 \times 2^5 = 1 \times 32 = 32$
- $1 \times 2^6 = 1 \times 64 = \underline{64}$
- 106_{10}

Exercise: convert a. 1100110_2 b. 11111001_2 to decimal.

- *Converting decimal number to binary using remainder method.*
  1. Divide the decimal number by the base (in case of binary divide by 2)
  2. Indicate the remainder to the right.
  3. Continue dividing each quotient (indicating the remainder) until the divide operation produces a zero quotient.

- Example: convert 99_{10} to binary.
- r
- 2 99 1
- 2 49 1
- 2 24 0
- 2 12 0
- 2 6 0
- 2 3 1
- 2 1 1
- 0

The answer is bottom up 1100011_2

- Example convert 13_{10} to binary
- R
- 2 13 1
- 2 6 0
- 2 3 1
- 2 1 1
- 0

$13_{10} = 1101_2$

Exercise: convert the following numbers to binary: a. 49_{10} b. 21_{10}

- *Converting octal number to decimal number*
- Example: convert 367_8 to decimal
- 3 6 7
- $7 \times 8^0 = 7 \times 1 = 7$
- $6 \times 8^1 = 6 \times 8 = 48$
- $3 \times 8^2 = 3 \times 64 = \underline{192}$
- 247_{10}

- Example: convert 1601_8 to decimal.
- 1 6 0 1
- $1 \times 8^0 = 1 \times 1 = 1$
- $0 \times 8^1 = 0 \times 8 = 0$
- $6 \times 8^2 = 6 \times 64 = 384$
- $1 \times 8^3 = 1 \times 512 = \underline{512}$
- 897_{10}

Exercise: convert the following numbers to decimal: a. 536_8 b. 1163_8

- *Converting decimal number to octal number.*
  1. Divide the decimal number by the base (in this case of octal, divide by 8).
  2. Indicate the remainder to the right.
  3. Continue dividing into each quotient (and indicating the remainder) until the division operation produces a zero quotient.
- The base 8 number is the numeric remainder reading from the last division to the first.
-
- Example: convert 465_{10} to octal.
- r
- 8 | 465 | 1
- 8 | 58 | 4
- 8 | 7 | 7
- | 0 |
- $465_{10} = 741_8$

- Example: convert 2548_{10} to octal.
- r
- 8 | 2548 | 4
- 8 | 318 | 6
- 8 | 39 | 7
- 8 | 4 | 4
- | 0 |
- $2548_{10} = 4764_8$

Exercise: convert the following decimal numbers to octal: a. 3002 b. 6512.

- *Converting hexadecimal number to decimal number.*
-
- Example: convert $20B3_{16}$ to decimal
- 2 0 B 3
- $3 \times 16^0 = 3 \times 1 = 3$
- $B(11) \times 16^1 = 11 \times 16 = 176$
- $0 \times 16^2 = 0 \times 256 = 0$
- $2 \times 16^3 = 2 \times 4096 = \underline{8192}$
- 8371_{10}
-
- Example: convert $12AE5_{16}$ to decimal.
- 1 2 A(10) E(14) 5
- $5 \times 16^0 = 5 \times 1 = 5$
- $14 \times 16^1 = 14 \times 16 = 224$
- $10 \times 16^2 = 10 \times 256 = 2560$
- $2 \times 16^3 = 2 \times 4096 = 8192$
- $1 \times 16^4 = 1 \times 65536 = \underline{65536}$
- 76517_{10}

Exercise: Convert the following numbers to decimal: a. $243F_{16}$ b. $BEEF_{16}$

- *Converting decimal number to hexadecimal*
-
- Example convert 9263_{10} to hexadecimal
- r
- 16 | 9263 | F
- 16 | 578 | 2
- 16 | 36 | 4
- 16 | 2 | 2
- 0

- $9263_{10} = 242F_{16}$
- Example; convert 4259_{10} to hexadecimal.

		r
16	4259	3
16	266	A
16	16	0
16	1	1
	0	

- $4259_{10} = 10A3_{16}$

Exercise: convert the following numbers to hexadecimal: a. 69498_{10} b. 114267_{10}

- *Converting binary to hexadecimal or vice versa*
- Four binary digits are equivalent to one hexadecimal digit
- Divide the binary number into groups of 4 starting from the right.
- If the left most group has less than 4 digits, put necessary number of leading zeros on the left.
- For each group, write equivalent hexadecimal digit
-
- Example: convert 1101001101110111_2
- 1011 0011 0111 0111
- D 3 7 7 (Hex)
-
- Example: convert 101101111_2 to hexadecimal
- 0001 0110 1111
- 1 6 F (Hex)
-
- Example: convert $1BE9_{16}$ to binary.
- Hex 1 B E 9
- Binary 001 1011 1110 1001

- Example: convert $B0A_{16}$ to binary
- Hex B 0 A
- Binary 1011 0000 1010

- *Converting from octal to binary or from binary to octal*.
- Three binary digits are equivalent to one octal digit.
- To convert from binary to octal, divide the binary number into group of three digits starting from the right.
- If the left most group has less than three digits, put necessary leading zeros.
- For each group of three bits, write a single corresponding octal digit.
-
- Example; convert 1101001101110111_2 to octal.
- Binary 001 101 001 101 110 111
- Octal 1 5 1 5 6 7
-
- Example: convert 10110111_2 to octal.
- Binary 101 101 111
- Octal 5 5 7
-
- Example: convert 1764_8 to binary.
- Octal 1 7 6 4
- Binary 001 111 110 0100
-
- Example: convert 731_8 to binary.
- Octal 7 3 1
- Binary 111 011 001

ASCII

- ASCII represents American standard code for information interchange.
- This is a character encoding scheme originally based on the English alphabet.
- ASCII codes represent text in computers, communication equipment and other devices that use text.

- Most modern characters schemes are based on ASCII, though they support many additional characters.
- ASCII include the definition of 128 characters (many now obsolete) that affect how text or space are processed and 95 printable characters including the space.

Table of ASCII codes

EBCDIC

- EBCDIC stands for extended binary coded decimal interchange code
- It is an 8 bit character encoding used mainly on IBM mainframe and IBM midrange computer operating system.
- It descends from the codes used with punch cards and the corresponding six bit binary coded decimal code, used with most of the IBM computer peripherals of late 1950s and 1960s.
- It is also employed on various non IBM platforms such as Fujistu.-Siemens BS2000/OSD/ix and the Unisys MCP.
- It was created to extend the existing binary coded decimal (BCD) interchange code.
- While IBM was a proponent of ASCII, it did not have time to prepare for ASCII peripherals.

UNICODE

- Unicode provides a unique number for every character.
- It developed because of limitations of existing text encoding schemes such as ASCII.
- ASCII can accommodate up to 256 characters because it uses only a byte for storage of information.
- ASCII was sufficient when only English characters were encoded.
- If you consider characters for all other languages in the world, ASCII does not have capacity to accommodate them.
- Hence UNICODE comes in to solve this problem.
- Unicode provides a unique number for every character:
    - No matter what the platform
    - No matter what the program
    - No matter what the language

- Before UNICODE was invented, there were hundreds of different of encoding schemes for assigning these numbers.
- The encoding schemes also conflicted with each other.
- That is two encoding schemes can use the same number for two different characters or use different numbers for the same character.
- UNICODE uses up to four bytes for storage of these numbers that represent the characters.

ASCII	DEC	Hex	Octal	Binary
NULL	000	00	000	0000 0000
SOH	001	01	001	0000 0001
STX	002	02	002	0000 0010
ETX	003	03	003	0000 0011
EOT	004	04	004	0000 0100
ENQ	005	05	005	0000 0101
ACK	006	06	006	0000 0110
BEL	007	07	007	0000 0111
BS	008	08	010	0000 1000
HT	009	09	011	0000 1001
LF	010	0A	012	0000 1010
VT	011	0B	013	0000 1011
FF	012	0C	014	0000 1100
CR	013	0D	015	0000 1101
SO	014	0E	016	0000 1110
SI	015	0F	017	0000 1111
DLE	016	10	020	0001 0000
DC1	017	11	021	0001 0001
DC2	018	12	022	0001 0010
DC3	019	13	023	0001 0011
DC4	020	14	024	0001 0100
NAK	021	15	025	0001 0101
SYN	022	16	026	0001 0110
ETB	023	17	027	0001 0111
CAN	024	18	030	0001 1000
EM	025	19	031	0001 1001
SUB	026	1A	032	0001 1010
ESC	027	1B	033	0001 1011
FS	028	1C	034	0001 1100
GS	029	1D	035	0001 1101
RS	030	1E	036	0001 1110
US	031	1F	037	0001 1111
space	032	20	040	0010 0000
!	033	21	041	0010 0001
"	034	22	042	0010 0010
#	035	23	043	0010 0011
$	036	24	044	0010 0100
%	037	25	045	0010 0101
&	038	26	046	0010 0110
'	039	27	047	0010 0111
(	040	28	050	0010 1000
)	041	29	051	0010 1001
*	042	2A	052	0010 1010

ASCII	Dec	Hex	Octal	Binary
+	043	2B	053	0010 1011
,	044	2C	054	0010 1100
-	045	2D	055	0010 1101
.	046	2E	056	0010 1110
/	047	2F	057	0010 1111
0	048	30	060	0011 0000
1	049	31	061	0011 0001
2	050	32	062	0011 0010
3	051	33	063	0011 0011
4	052	34	064	0011 0100
5	053	35	065	0011 0101
6	054	36	066	0011 0110
7	055	37	067	0011 0111
8	056	38	070	0011 1000
9	057	39	071	0011 1001
:	058	3A	072	0011 1010
;	059	3B	073	0011 1011
a	060	3C	074	0011 1100
=	061	3D	075	0011 1101
>	062	3E	076	0011 1110
?	063	3F	077	0011 1111
@	064	40	100	0100 0000
A	065	41	101	010 00001
B	066	42	102	0100 0010
C	067	43	103	0100 0011
D	068	44	104	0100 0100
E	069	45	105	0100 0101
F	070	46	106	0100 0110
G	071	47	107	0100 0111
H	072	48	110	0100 1000
I	073	49	111	0100 1001
J	074	4A	112	0100 1010
K	075	4B	113	0100 1011
L	076	4C	114	0100 1100
M	077	4D	115	0100 1101
N	078	4E	116	0100 1110
O	079	4F	117	0100 1111
P	080	50	120	010 10000
Q	081	51	121	0101 0001
R	082	52	122	0101 0010
S	083	53	123	0101 0011
T	084	54	124	0101 0100
U	085	55	125	0101 0101

ASCII	Dec	Hex	Octal	Binary
V	086	56	126	0101 0110
W	087	57	127	0101 0111
X	088	58	130	0101 1000
Y	089	59	131	0101 1001
Z	090	5A	132	0101 1010
[	091	5B	133	0101 1011
\	092	5C	134	0101 1100
]	093	5D	135	0101 1101
^	094	5E	136	0101 1110
_	095	5F	137	0101 1111
`	096	60	140	0110 0000
a	097	61	141	0110 0001
b	098	62	142	0110 0010
c	099	63	143	0110 0011
d	100	64	144	0110 0100
e	101	65	145	0110 0101
f	102	66	146	0110 0110
g	103	67	147	0110 0111
h	104	68	150	0110 1000
i	105	69	151	0110 1001
j	106	6A	152	0110 1010
k	107	6B	153	0110 1011
l	108	6C	154	0110 1100
m	109	6D	155	0110 1101
n	110	6E	156	0110 1110
o	111	6F	157	0110 1111
p	112	70	160	0111 0000
q	113	71	161	0111 0001
r	114	72	162	0111 0010
s	115	73	163	0111 0011
t	116	74	164	0111 0100
u	117	75	165	0111 0101
v	118	76	166	0111 0110
w	119	77	167	0111 0111
x	120	78	170	0111 1000
y	121	79	171	0111 1001
z	122	7A	172	0111 1010
{	123	7B	173	0111 1011
|	124	7C	174	0111 1100
}	125	7D	175	0111 1101
~	126	7E	176	0111 1110
DEL	127	7F	177	0111 1111