MACHINE REPRESENTATION & DATA (CODES) IN COMPUTER SYSTEMS

Olawale J. Omotosho
Babcock University,
Computer Science Department
Ilishan-Remo, Ogun State, Nigeri
Tel: +2348034951089
Email: ojomotosho1@yahoo.com

Samson O. Ogunlere
Babcock University,
Computer Science Department
Ilishan-Remo, Ogun State, Nigeria
Tel: +2348067622845
Email: ogunlere@yahoo.com

Oludele Awodele
Babcock University,
Computer Science Department
Ilishan-Remo, Ogun State, Nigeria
Tel: +2348033378761
Email: delealways@yahoo.com

Abstract

Computer compatibility is an essential requirement when computer applications are being considered in any given environment. Compatibility is a design issue that affects both software and hardware components of computers. This paper examines how best to code computer data/characters in such a way that compatibility issue can be eliminated. The resulting efforts have brought about the development of a Coding Format and the logical assignment of binary codes to different computer characters, using the American Standard Code for International Interchange (ASCII) as a guide.

Key words—Compatibility, Coding format, Number systems, Data/Character.

1. Introduction

Communication between two locations, people and/or things (sender & receiver) is a two-way traffic for effective results. That is, the sender must send messages to the receiver in language(s) the receiver can understand before effective communication can be said to have taken place. Communication between man and computer is not an exception to this requirement. Hence, the need to use a language understandable by the computer is required. What then is the language of computers? In order to answer this question, knowledge of what computers are and what they are made of is required to be able to develop the language with which to address them.

Computers are two-state machines. That is, the components of which computers are made from are bistable elements and logic gates. These elements are either in one stable state or the other and this behaviour is best described by the binary number system. It is therefore clear that any communication mode required between man and computers must take into account the binary number system, which is a string of 1s and 0s. It is also important to point out here that the man's world is not binary but decimal. A communication gap is therefore, created between these two worlds (computer & man) as shown in the block diagram of Fig 1.1 unless the decimal world is converted to binary world through the communication channel, (mc) and binary world is converted into decimal world through the communication channel (cm).

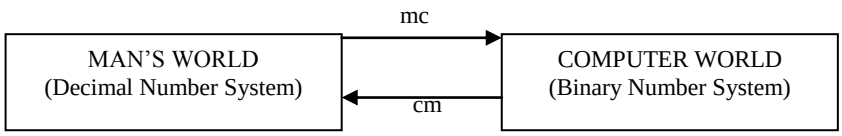


Fig 1.1: Communication Between Man & Computer/Computer & Man

Different languages, which are easily understood by man have been developed and more are still being investigated to make communication with computer as close to 'man-to-man' communication as possible. While such languages are developed from human considerations, interpreter of such languages into binary format (strings of 1s and 0s) are equally being developed. This string of 1s and 0s is called the machine language, the language computers understand. Therefore, the ABC of a computer language is 1s and 0s. Hence, all data into computers can only be represented by strings of 1s and 0s.

The combinations of these 1s and 0s which represent different characters are called binary codes. It is therefore required that every character that will be sent to a computer must be so coded. The word character here denotes any data/information that is being communicated to a computer. For an instance, all the alphabets (upper & lower cases), punctuation marks, special and mathematical symbols, numerical and non-numerical symbols including all other non-printable symbols such as commands, control, etc are all classified as computer characters and they must all be binary coded.

2. Codes & Coding Format

Codes can be broadly divided into two classes as alphanumeric and Non-alphanumeric. The classification cannot be exhausted but are limited to the currently and popularly implemented ones in electronic industries especially for digital designs and fabrications of digital equipment. Such classification is presented in Fig 2.1.

Furthermore, codes can be used for other reasons such as for security purposes, for processing numerical information. The choice of codes depends on the function or purpose the codes have to serve. Some codes are suitable for arithmetic operations such as Binary Coded Decimal. Others may lend themselves to production of hardware devices of higher efficiency like the Gray Codes. There are some that are specifically meant to detect and/or correct errors, especially in transmitted data, such as Parity Codes. The paper is primarily concerned about the Alphanumeric Codes.

In Alphanumeric Codes, after knowing the total number of characters to be coded, symbols of the same number must be provided so that each of the character could be assigned a unique symbol. These symbols are called Codes. Because these codes are meant to be used by computers, these symbols can be derived from binary number system as earlier mentioned.

It is discovered that the concept of matrix can be of immense advantage in creating as many cells as there are number of characters to be coded. For an example, an [r×c] Matrix will generate (r×c) cells. Mathematically expressed as,

$$(r \times c) = N \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

where N = Number of characters to be coded.
r = Number of rows of the matrix and
c = Number of columns of the matrix

Therefore, an (r×c) Matrix can adequately be used to assign codes to all the characters to be coded. Hence, this matrix is referred to as the Coding Format which provides logical arrangement for the assignment of codes to the characters.

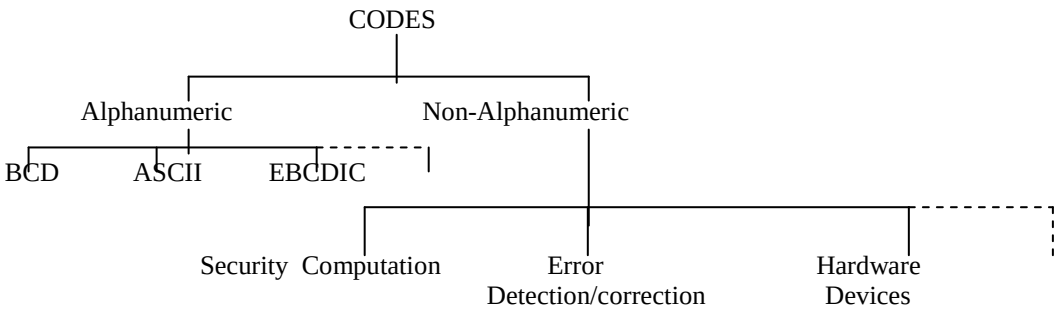


Fig 2.1: Line Diagram of Code Classification

In general more than one matrix will satisfy equation (2.1). At the worst, the matrix conjugate, (c×r) matrix will certainly be available in addition to (r×c) matrix. The total number of matrices that satisfies equation (2.1) is empirically derived in Table-2.1.

TABLE-2.1: Possible Coding Matrices

$N = 2^n$		POSSIBLE MATRICES		Total Number of Matrices, M
Value of 'n'	Value of N	Main	Conjugate	
1	2	1×2	2×1	2
2	4	1×4, 2×2	4×1	3
3	8	1×8, 2×4	8×1, 4×2	4
4	16	1×16, 2×8, 4×4	16×1, 8×2	5
5	32	1×32, 2×16, 4×8	32×1, 16×2, 8×4	6
6	64	1×64, 2×32, 4×16, 8×8	64×1, 32×2, 16×4	7
Total Number of Matrices, M = n +1				

2.1 Binary Coding of Alphanumeric Codes

Binary coding involves an orderly arrangement of the strings of 1s and 0s such that no two codes will represent the same character and/or no one character will have two different codes. In order to be able to identify the number of 1s and 0s that will make up a string for a character, the total number of possible characters that will be required to send to a computer is determined as in Table-2.2.

TABLE-2.2: Total Number of Possible Characters to be Coded

S/N	CLASS OF CHARACTERS	CHARACTERS	NUMBER OF CHARACTERS
1	Control/Non-Printable Symbols	NULL (Null), SOH (Start of Heading), DEL (delete), BELL (Bell), EOT (End of Transmission), etc	32
2	Punctuations & Special Symbols	Space, !, “, #, &, $, %, (, *, +, -, /,), ., :, ; etc	16
3	Numerical Symbols	0,1,2,3,4,5,6,7,8,9	10
4	Maths & Special Symbols	<, =, >, ×, ±, ≥, ?, ≤, ≠, @, etc	7
5	Alphabets (upper & lower cases)	A,B,C,D,……X,Y,Z, a,b,c,d,……x,y,z &	52
6	Other symbols not covered by items 1-5	{, [, ^, ‘,], }, etc	10
Total Number of Characters			**128**

The total number of possible characters to be send to computers is 128 in the case presented in Table-2.2. If these characters are to be represented by different distinct symbols other than the combination of 1s and 0s, there will be as many symbols as there are characters. This means that the electronic circuit required to achieve this possibility, or to realize these characters will have as many states as the number of characters to be transferred to the computers. Such a hardware will be difficult to design and fabricate. It is for the same reason that decimal number system is not suitable for computer architecture.

Combinations of 1s and 0s, otherwise referred to as bits are therefore used to represent these characters instead of distinct symbols and the number of bits required for one character is dependent on the total possible characters the computers are intended to handle. Let us examine certain scenarios as follows:
For a 2-bit code, the total maximum possible characters is $2^2 = 4$ characters.
For a 3-bit code, the total maximum possible characters is $2^3 = 8$ characters.
For a 4-bit code, the total maximum possible characters is $2^4 = 16$ characters.
Thus, for a n-bit code, the total maximum possible characters is 2^n characters.

Therefore, equation (2.2) is the valid expression that can be used to determine the number of bits required to code a given set of characters.

$$2^n = N \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

where n = number of bits and
N = maximum number of character to be coded as defined previously.

Equations (2.1) & (2.2) are sufficient to derive appropriate matrix or number of bits that can be used to fully describe codes for a given number of characters or vice-visa. From equation (2.2), the number of bits, ‘n’ required to code the 128 characters identified in Table-2.2 is 7. The relative position of 1s and 0s is a matter of choice of the computer designers or manufacturers. However, if different computer manufacturers adopt different relative positions of these bits, different codes for the same character will result and computers from one manufacturer will not be able to communicate accurately with computers from another manufacturer. Such a situation is referred to as incompatibility of one computer with another. Therefore, for compatibility and universality, the combination of these bits must be standardized. Hence, acceptable conventions are established and these are the ones used on computer machines. The most popular amongst these conventions are the **American Standard Code for Information Interchange** (ASCII, pronounced ‘As Key’) and the **Extended Binary Coded Decimal Interchange Code** (EBCDIC, pronounced ‘Eb-See-Dick’). In addition, there is also the **Extended American Standard Code for Information Interchange** (EASCII)

2.1.1 American Standard Code for Information Interchange, ASCII

This is a 7-bit binary code with the relative positions of the bits defined by a matrix of [16×8] or [8×16]. A matrix is defined by the number of rows (r) and the number of columns (c) it has. That is, a matrix [16×8] has 16 rows and 8

columns. These two matrices shall be examined separately with all possible configurations to determine which one is now known as ASCII.

Matrix [16×8]

Each cell of this matrix can be labelled serial along its rows (backward & forward) or along its columns (backward & forward) in two different ways (zig-zag & continuous) to make a total of sixteen possibilities as presented in Table-2.3. In order that each cell of the matrix be fully described, the rows and columns of the matrix are labelled, usually using binary numbers. The combinations of the labels of these rows and columns in the order of row/column or column/row must agree with any other mode/style of numbering the same cell. Where this condition is not met, such matrix numbering or option is regarded as invalid and therefore discarded.

TABLE-2.3a: Different Options of Numbering Matrix Cells

Numbering Style	Forward Numbering	Backward Numbering
Zig-zag (Row)	2 possibilities	2 possibilities
Zig-zag (Column)	2 possibilities	2 possibilities
Continuous (Row)	2 possibilities	2 possibilities
Continuous (Column)	2 possibilities	2 possibilities
For Row Consideration = 8 options For Column Consideration = 8 options **Total number of options = 16**		

In order to illustrate the contents of Table-2.3a, the numbering of the cells of a [4×4] matrix is examined as depicted in Table-2.3b.

Table-2.3b: Possibilities of Numbering of [4×4] Matrix

Forward	Forward	Forward	Forward	Forward Numbering – 1 (Continuous Along Columns) **Option-1**
Backward	Backward	Backward	Backward	Backward Numbering – 2 (Continuous Along Columns) **Option-2**
0 7 8 15 1 6 9 14 2 5 10 13 3 4 11 12	15 8 7 0 14 9 6 1 13 10 5 2 12 11 4 3	3 4 11 12 2 5 10 13 1 6 9 14 0 7 8 15	12 11 4 3 13 10 5 2 14 9 6 1 15 8 7 0	Forward Numbering – 1 (Continuous Along Columns) $1_1 = 2_2$. $2_1 = 1_2$. $3_1 = 4_2$. $4_1 = 3_2$.
1	2	3	4	
15 8 7 0	0 7 8 15	12 11 4 3	3 4 11 12	Backward Numbering – 2

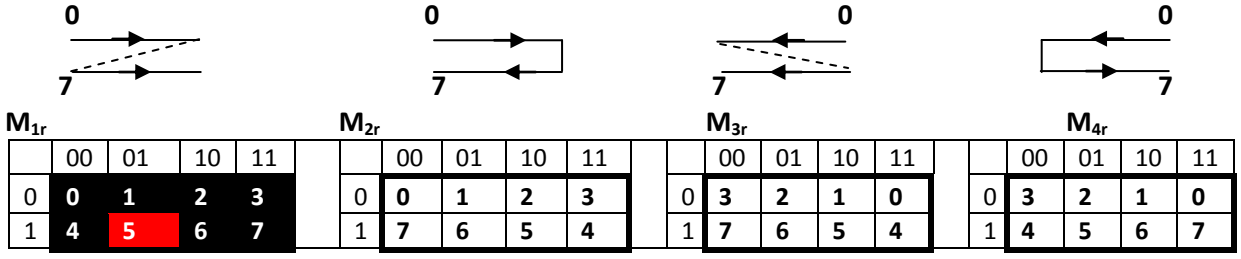
NOTE: From this analysis, there are only four (4) distinct possibilities in numbering the cells. Hence, two (2) possibilities per type of numbering (Forward & Backward) is required.

14	9	6	1		1	6	9	14		13	10	5	2		2	5	10	13	(Continuous Along Columns)
13	10	5	2		2	5	10	13		14	9	6	1		1	6	9	14	$1_2 = 2_1$.
12	11	4	3		3	4	11	12		15	8	7	0		0	7	8	15	$2_2 = 1_1$.
1					2					3					4				$3_2 = 4_1$.
																			$4_2 = 3_1$.

A [2×4] matrix will be used to illustrate this principle here for the economy of space and clarity.

(i) Matrix Row Consideration

On the first row of the four set of matrix, the matrix cells are numbered, starting with zero, as per the sketch illustrated above the matrix set.



	M_{1r}				M_{2r}				M_{3r}				M_{4r}			
	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0	0	1	2	3	0	1	2	3	3	2	1	0	3	2	1	0
1	4	5	6	7	7	6	5	4	7	6	5	4	4	5	6	7

Fig 2.2a: Four Options of Labelling 2×4 Matrix Cells along Rows

If in Fig 2.2a, the matrix cells are identified by the ***column & row*** labels in that order, cell 5 of (M_{1r}) will be labelled as = 01 & 1 = 011. Hence, Fig 2.2a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.2b.

	$(c\text{-}r)_1$				$(c\text{-}r)_2$				$(c\text{-}r)_3$				$(c\text{-}r)_4$			
	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0	000	010	100	110	000	010	100	110	000	010	100	110	000	010	100	110
1	001	011	101	111	001	011	101	111	001	011	101	111	001	011	101	111

Fig 2.2b: Each 2×4 Matrix Cell Identified by the Column & Row Labels

If in Fig 2.2a, the matrix cells are identified by the ***row & column*** labels in that order, cell 5 of (M_{1r}) will be labelled as = 1 & 01 = 101. Hence, Fig 2.2a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.2c.

	$(r\text{-}c)_1$				$(r\text{-}c)$				$(r\text{-}c)_3$				$(r\text{-}c)_4$			
	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0	000	001	010	011	000	001	010	011	000	001	010	011	000	001	010	011
1	100	101	110	111	100	101	110	111	100	101	110	111	100	101	110	111

Fig 2.2c: Each 2×4 Matrix Cell Identified by the Row & Column Labels

In Fig 2.2d, the contents of the matrix cells in Fig 2.2b are converted into decimal numbers.
For an example, cell 011 of Fig 2.2b = 3 in Fig 2.2d.

	$(c\text{-}r)_{11}$				$(c\text{-}r)_{22}$				$(c\text{-}r)_{33}$				$(c\text{-}r)_{44}$			
	00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
0	0	2	4	6	0	2	4	6	0	2	4	6	0	2	4	6
1	1	3	5	7	1	3	5	7	1	3	5	7	1	3	5	7

Fig 2.2d: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.2b

Similarly in Fig 2.2e, the contents of the matrix cells in Fig 2.2c are converted into decimal numbers. For an example, cell 101 of Fig 2.2c = 5 in Fig 2.2e.

$(r\text{-}c)_{11}$

	00	01	10	11
0	0	1	2	3
1	4	5	6	7

$(r\text{-}c)_{22}$

	00	01	10	11
0	0	1	2	3
1	4	5	6	7

$(r\text{-}c)_{33}$

	00	01	10	11
0	0	1	2	3
1	4	5	6	7

$(r\text{-}c)_{44}$

	00	01	10	11
0	0	1	2	3
1	4	5	6	7

Fig 2.2e: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.2c

Comparing Fig 2.2a with Figs 2.2d & 2.2e, the matrix labelling that agrees with one another is identified. These matrices are as marked in the appropriate figures as Fig 2.2a M_{1r} &
Fig 2.2e$(r\text{-}c)_{11}$.

This same process is repeated for column consideration as follows:

(ii) <u>Matrix Column Consideration</u>

On the first row of the four set of matrix, the matrix cells are numbered, starting with zero, as per the sketch illustrated above the matrix set.

M_{1c}

	00	01	10	11
0	0	2	4	6
1	1	3	5	7

M_{2c}

	00	01	10	11
0	0	3	4	7
1	1	2	5	6

M_{3c}

	00	01	10	11
0	1	3	5	7
1	0	2	4	6

M_{4c}

	00	01	10	11
0	1	2	5	6
1	0	3	4	7

Fig 2.3a: Four Options of Labelling 2×4 Matrix Cells along Columns

If in Fig 2.3a, the matrix cells are identified by the ***column & row*** labels in that order, cell 5 of (M_{1c}) will be labelled as = 10 & 1 = 101. Hence, Fig 2.3a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.3b.

$(c\text{-}r)_1$

	00	01	10	11
0	000	010	100	110
1	001	011	101	111

$(c\text{-}r)_2$

	00	01	10	11
0	000	010	100	110
1	001	011	101	111

$(c\text{-}r)_3$

	00	01	10	11
0	000	010	100	110
1	001	011	101	111

$(c\text{-}r)_4$

	00	01	10	11
0	000	010	100	110
1	001	011	101	111

Fig 2.3b: Each 2×4 Matrix Cell Identified by the Column & Row Labels

If in Fig 2.3a, the matrix cells are identified by the ***row & column*** labels in that order, cell 5 of (M_{1c}) will be labelled as = 1 & 10 = 110. Hence, Fig 2.2a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.2c.

$(r\text{-}c)_1$

	00	01	10	11
0	000	001	010	011
1	100	101	110	111

$(r\text{-}c)_1$

	00	01	10	11
0	000	001	010	011
1	100	101	110	111

$(r\text{-}c)_1$

	00	01	10	11
0	000	001	010	011
1	100	101	110	111

$(r\text{-}c)_1$

	00	01	10	11
0	000	001	010	011
1	100	101	110	111

Fig 2.3c: Each 2×4 Matrix Cell Identified by the Row & Column Labels

In Fig 2.3d, the contents of the matrix cells in Fig 2.3b are converted into decimal numbers.
For an example, cell 101 of Fig 2.3b = 5 in Fig 2.2d.

$(c\text{-}r)_{11}$

	00	01	10	11
0	0	2	4	6

$(c\text{-}r)_{22}$

	00	01	10	11
0	0	2	4	6

$(c\text{-}r)_{33}$

	00	01	10	11
0	0	2	4	6

$(c\text{-}r)_{44}$

	00	01	10	11
0	0	2	4	6

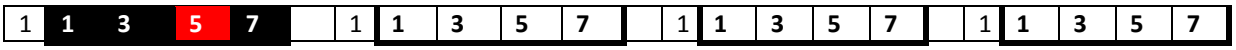


Fig 2.3d: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.3b

Similarly in Fig 2.3e, the contents of the matrix cells in Fig 2.3c are converted into decimal numbers. For an example, cell 110 of Fig 2.3c = 6 in Fig 2.2e.

$(r\text{-}c)_{11}$					$(r\text{-}c)_{22}$					$(r\text{-}c)_{33}$					$(r\text{-}c)_{44}$				
	00	01	10	11		00	01	10	11		00	01	10	11		00	01	10	11
0	**0**	**1**	**2**	**3**	0	**0**	**1**	**2**	**3**	0	**0**	**1**	**2**	**3**	0	**0**	**1**	**2**	**3**
1	**4**	**5**	**6**	**7**	1	**4**	**5**	**6**	**7**	1	**4**	**5**	**6**	**7**	1	**4**	**5**	**6**	**7**

Fig 2.3e: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.3c

Comparing Fig 2.3a with Figs 2.3d & 2.3e, the matrix labelling that agrees with one another is identified. These matrices are as marked in the appropriate figures as Fig 2.3a M_{1c} &
Fig 2.3e$(c\text{-}r)_{11}$.

From the above analysis, there are only two distinct options that are consistent and logically and methodically assignable without ambiguity out of identified sixteen possibilities of Table 2.3a. Both options are from zig-zag configuration with one labelled along the column and the other along row. Hence, in the case of [16×8] Matrix the two possible options are presented in Table-2.4a & Table-2.4b.

Table-2.4a: [16×8] Matrix: The cells are labelled in Decimal Numbers in Zig-Zag Along Row

B_{10}	B_{16}	B_{10}	0	1	2	3	4	5	6	7
		B_{16}	0	1	2	3	4	5	6	7
		B_2	000	001	010	011	100	101	110	111
0	0	0000	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1	0001	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
2	2	0010	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
3	3	0011	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
4	4	0100	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**
5	5	0101	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
6	6	0110	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**
7	7	0111	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
8	8	1000	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**
9	9	1001	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
10	A	1010	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**
11	B	1011	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
12	C	1100	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**
13	D	1101	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
14	E	1110	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
15	F	1111	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**

The reading order is (r,c). Examples: Cell 22 = 0010 110, Cell 99 = 1100 011

Table-2.4b: [16×8] Matrix: The cells are labelled in Decimal Numbers in Zig-Zag Along Column

B_{10}	B_{16}	B_{10}	0	1	2	3	4	5	6	7
		B_{16}	0	1	2	3	4	5	6	7
		B_2	000	001	010	011	100	101	110	111
0	0	0000	**0**	**16**	**32**	**48**	**64**	**80**	**96**	**112**
1	1	0001	**1**	**17**	**33**	**49**	**65**	**81**	**97**	**113**
2	2	0010	**2**	**18**	**34**	**50**	**66**	**82**	**98**	**114**
3	3	0011	**3**	**19**	**35**	**51**	**67**	**83**	**99**	**115**
4	4	0100	**4**	**20**	**36**	**52**	**68**	**84**	**100**	**116**

5	5	0101	**5**	**21**	**37**	**53**	**69**	**85**	**101**	**117**
6	6	0110	**6**	**22**	**38**	**54**	**70**	**86**	**102**	**118**
7	7	0111	**7**	**23**	**39**	**55**	**71**	**87**	**103**	**119**
8	8	1000	**8**	**24**	**40**	**56**	**72**	**88**	**104**	**120**
9	9	1001	**9**	**25**	**41**	**57**	**73**	**89**	**105**	**121**
10	A	1010	**10**	**26**	**42**	**58**	**74**	**90**	**106**	**122**
11	B	1011	**11**	**27**	**43**	**59**	**75**	**91**	**107**	**123**
12	C	1100	**12**	**28**	**44**	**60**	**76**	**92**	**108**	**124**
13	D	1101	**13**	**29**	**45**	**61**	**77**	**93**	**109**	**125**
14	E	1110	**14**	**30**	**46**	**62**	**78**	**94**	**110**	**126**
15	F	1111	**15**	**31**	**47**	**63**	**79**	**95**	**111**	**127**
The reading order is (c,r). Examples: Cell 22 = 001 0110, Cell 99 = 110 0010										

Finally, Table 2.4b is preferred to Table 2.4a because the readings produced for each cell are in the same format as the ASCII codes. Therefore, Table 2.4b is the convention adopted to describe the totality of ASCII codes. However, Table 2.4a offers an alternative just as any other six forms earlier discarded is a possible option if the codes are to be assigned randomly.

Matrix [8×16]

Similarly like [16×8] Matrix, each cell of the matrix can be labelled serial along its rows (backward & forward) or along its columns (backward & forward) in two different ways (zig-zag & continuous) to make a total of sixteen possibilities. A 4×2 matrix will be used to illustrate this principle here for clarity.

(i) Matrix Row Consideration

On the first row of the four set of matrix, the matrix cells are numbered, starting with zero, as per the sketch illustrated above the matrix set.

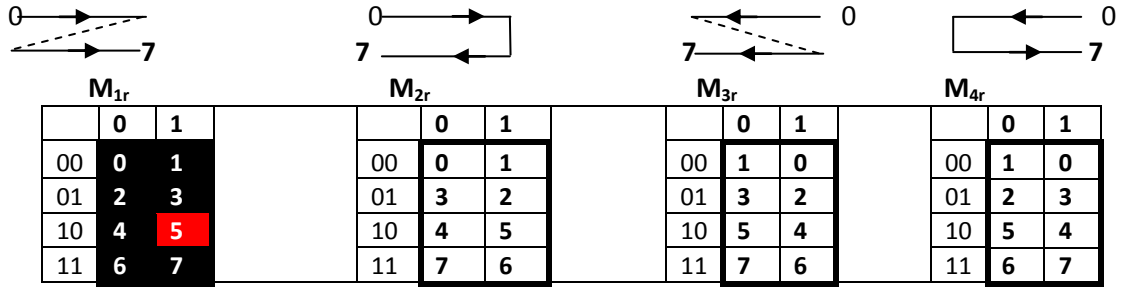


Fig 2.4a: Four Options of Labelling 4×2 Matrix Cells along Rows

If in Fig 2.4a, the matrix cells are identified by the ***column & row*** labels in that order, cell 5 of (M_{1r}) will be labelled as = 1 & 10 = 110. Hence, Fig 2.4a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.4b.

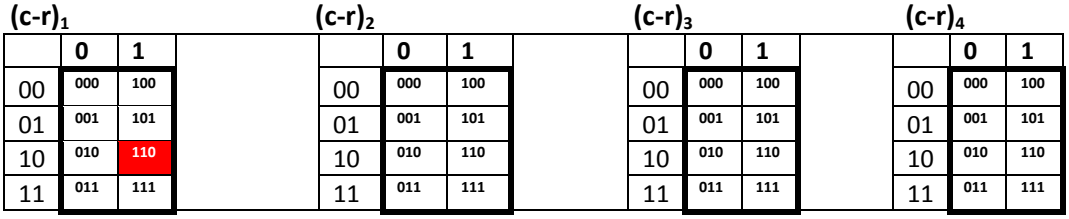
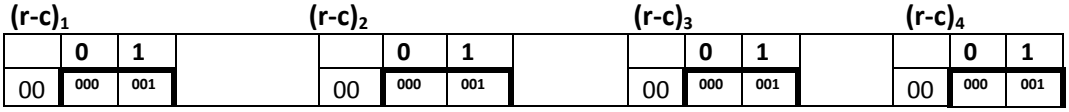


Fig 2.4b: Each 4×2 Matrix Cell Identified by the Column & Row Labels

If in Fig 2.4a, the matrix cells are identified by the ***row & column*** labels in that order, cell 5 of (M_{1r}) will be labelled as = 10 & 1 = 101. Hence, Fig 2.4a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.2c.



01	010	011		01	010	011		01	010	011		01	010	011
10	100	101		10	100	101		10	100	101		10	100	101
11	110	111		11	110	111		11	110	111		11	110	111

Fig 2.4c: Each 4×2 Matrix Cell Identified by the Row & Column Labels

In Fig 2.4d, the contents of the matrix cells in Fig 2.4b are converted into decimal numbers.
For an example, cell 011 of Fig 2.2b = 3 in Fig 2.2d.

$(c\text{-}r)_{11}$				$(c\text{-}r)_{22}$				$(c\text{-}r)_{33}$				$(c\text{-}r)_{44}$		
	0	1			0	1			0	1			0	1
00	0	4		00	0	4		00	0	4		00	0	4
01	1	5		01	1	5		01	1	5		01	1	5
10	2	6		10	2	6		10	2	6		10	2	6
11	3	7		11	3	7		11	3	7		11	3	7

Fig 2.4d: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.4b

Similarly in Fig 2.4e, the contents of the matrix cells in in Fig 2.4c are converted into decimal numbers. For an example, cell 101 of Fig 2.4c = 5 in Fig 2.4e.

$(r\text{-}c)_{11}$				$(r\text{-}c)_{22}$				$(r\text{-}c)_{33}$				$(r\text{-}c)_{44}$		
	0	1			0	1			0	1			0	1
00	0	1		00	0	1		00	0	1		00	0	1
01	2	3		01	2	3		01	2	3		01	2	3
10	4	5		10	4	5		10	4	5		10	4	5
11	6	7		11	6	7		11	6	7		11	6	7

Fig 2.4e: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.4c

Comparing Fig 2.4a with Figs 2.4d & 2.4e, the matrix labelling that agrees with one another is identified. These matrices are as marked in the appropriate figures, Fig 2.4a M_{1r} & Fig 2.4e$(r\text{-}c)_{11}$.

(ii) <u>Matrix Column Consideration</u>

On the first row of the four set of matrix, the matrix cells are numbered, starting with zero, as per the sketch illustrated above the matrix set.

M_{1c}				M_{2c}				M_{3c}				M_{4c}		
	0	1			0	1			0	1			0	1
00	0	4		00	0	7		00	3	7		00	3	4
01	1	5		01	1	6		01	2	6		01	2	5
10	2	6		10	2	5		10	1	5		10	1	6
11	3	7		11	3	4		11	0	4		11	0	7

Fig 2.5a: Four Options of Labelling 4×2 Matrix Cells along Columns

If in Fig 2.5a, the matrix cells are identified by the ***column & row*** labels in that order, cell 5 of (M_{1c}) will be labelled as = 1 & 01 = 101. Hence, Fig 2.5a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.5b.

$(c\text{-}r)_1$				$(c\text{-}r)_2$				$(c\text{-}r)_3$				$(c\text{-}r)_4$		
	0	1			0	1			0	1			0	1
00	000	100		00	000	100		00	000	100		00	000	100

01	001	101		01	001	101		01	001	101		01	001	101
10	010	110		10	010	110		10	010	110		10	010	110
11	011	111		11	011	111		11	011	111		11	011	111

Fig 2.5b: Each 4×2 Matrix Cell Identified by the Column & Row Labels

If in Fig 2.5a, the matrix cells are identified by the ***row & column*** labels in that order, cell 5 of (M_{1c}) will be labelled as = 01 & 1 = 011. Hence, Fig 2.5a is redrawn with each cell labelled according to this rule and this is shown in Fig 2.5c.

$(r\text{-}c)_1$				$(r\text{-}c)_2$				$(r\text{-}c)_3$				$(r\text{-}c)_4$			
	0	1			0	1			0	1			0	1	
00	000	001		00	000	100		00	000	100		00	000	100	
01	010	011		01	001	101		01	001	101		01	001	101	
10	100	101		10	010	110		10	010	110		10	010	110	
11	110	111		11	011	111		11	011	111		11	011	111	

Fig 2.5c: Each 4×2 Matrix Cell Identified by the Row & Column Labels

In Fig 2.5d, the contents of the matrix cells in Fig 2.5b are converted into decimal numbers.
For an example, cell 101 of Fig 2.3b = 5 in Fig 2.5d.

$(c\text{-}r)_{11}$				$(c\text{-}r)_{22}$				$(c\text{-}r)_{33}$				$(c\text{-}r)_{44}$			
	0	1			0	1			0	1			0	1	
00	0	4		00	0	4		00	0	4		00	0	4	
01	1	5		01	1	5		01	1	5		01	1	5	
10	2	6		10	2	6		10	2	6		10	2	6	
11	3	7		11	3	7		11	3	7		11	3	7	

Fig 2.5d: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.5b

Similarly in Fig 2.5e, the contents of the matrix cells in Fig 2.5c are converted into decimal numbers. For an example, cell 110 of Fig 2.5c = 6 in Fig 2.5e.

$(r\text{-}c)_{11}$				$(r\text{-}c)_{22}$				$(r\text{-}c)_{33}$				$(r\text{-}c)_{44}$			
	0	1			0	1			0	1			0	1	
00	0	1		00	0	1		00	0	1		00	0	1	
01	2	3		01	2	3		01	2	3		01	2	3	
10	4	5		10	4	5		10	4	5		10	4	5	
11	6	7		11	6	7		11	6	7		11	6	7	

Fig 2.5e: Each 2×4 Matrix Cell Identified by the Decimal Equivalent of Fig 2.5c

Comparing Fig 2.5a with Figs 2.5d & 2.5e, the matrix labelling that agrees with one another is identified. These matrices are as marked in the appropriate figures, Fig 2.5a M_{1c} &
Fig 2.5e$(c\text{-}r)_{11}$.

Consequent upon the above analysis, there are also only two distinct options that have satisfied the logically presented method out of identified sixteen possibilities of Table 2.3. Both options are from zig-zag configuration with one labelled along the column and the other along row. Hence, in the case of [8×16] Matrix the two possible options are presented in Tables 2.6a & 2.6b.

Table-2.6a: [8×16] Matrix: The cells are labelled in Decimal Numbers Zig-Zag Along Row

B_{10}	B_{16}																
		B_{10}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		B_{16}	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		B_2	000	000	001	001	010	010	011	011	100	100	101	101	110	110	111	111

			0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	0	000	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	1	001	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
2	2	010	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
3	3	011	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
4	4	100	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
5	5	101	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
6	6	110	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
7	7	111	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**	**120**	**121**	**122**	**123**	**124**	**125**	**126**	**127**
The reading order is (r,c). Examples: Cell 22 = 001 0110, Cell 99 = 110 0011																		

Table-2.6b: [8×16] Matrix: The cells are labelled in Decimal Numbers Zig-Zag Along Column

B_{10}	B_{16}	B_{10}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		B_{16}	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		B_2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	000	**0**	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**72**	**80**	**89**	**97**	**105**	**113**	**121**
1	1	001	**1**	**9**	**17**	**25**	**33**	**41**	**49**	**57**	**65**	**73**	**81**	**90**	**98**	**106**	**114**	**122**
2	2	010	**2**	**10**	**18**	**26**	**34**	**42**	**50**	**58**	**66**	**74**	**82**	**91**	**99**	**107**	**115**	**123**
3	3	011	**3**	**11**	**19**	**27**	**35**	**43**	**51**	**59**	**67**	**75**	**83**	**92**	**100**	**108**	**116**	**124**
4	4	100	**4**	**12**	**20**	**28**	**36**	**44**	**52**	**60**	**68**	**76**	**84**	**93**	**101**	**109**	**117**	**125**
5	5	101	**5**	**13**	**21**	**29**	**37**	**45**	**53**	**61**	**69**	**77**	**85**	**94**	**102**	**110**	**118**	**126**
6	6	110	**6**	**14**	**22**	**30**	**38**	**46**	**54**	**62**	**70**	**78**	**85**	**95**	**103**	**111**	**119**	**127**
7	7	111	**7**	**15**	**23**	**31**	**39**	**47**	**55**	**63**	**71**	**79**	**87**	**96**	**104**	**112**	**120**	**128**
The reading order is (c,r). Examples: Cell 22 = 0001110, Cell 99 =11 00010																		

Again, considering the method by which the content of each cell is read, only the form in Table 2.6a is in the same format as the codes for ASCII. Therefore, Table 2.6a is preferred and it is the same as Table 2.5b when Table 2.5b is rotated clockwise through 90^0.

The totality of this analysis has therefore, shown that there are two formats that fully describe the codes for 128 characters in the form that satisfy the American Standard Code for Information Interchange. It can therefore, be generalized that in any matrix, labelling in a zig-zag manner along the longer side will invariably generate the best fit, if ASCII code format is accepted as the best practice. The full ASCII codes are presented in Tables 2.7a & 2.7b.

Table-2.7a: [16×8] Matrix Describing ASCII Codes (Zig-Zag along Column)

B_{10}	B_{16}	B_{10}	0	1	2	3	4	5	6	7
		B_{16}	0	1	2	3	4	5	6	7
		B_2	000	001	010	011	100	101	110	111
0	0	0000	**NUL**	**DLE**	**SP**	**0**	**@**	**P**	**‘**	**p**
1	1	0001	**SOH**	**DC1**	**!**	**1**	**A**	**Q**	**a**	**q**
2	2	0010	**STX**	**DC2**	**“**	**2**	**B**	**R**	**b**	**r**
3	3	0011	**ETX**	**DC3**	**#**	**3**	**C**	**S**	**c**	**s**
4	4	0100	**EOT**	**DC4**	**$**	**4**	**D**	**T**	**d**	**t**
5	5	0101	**ENG**	**NAK**	**%**	**5**	**E**	**U**	**e**	**u**
6	6	0110	**ACK**	**SYN**	**&**	**6**	**F**	**V**	**f**	**v**
7	7	0111	**BEL**	**ETB**	**‘**	**7**	**G**	**W**	**g**	**w**
8	8	1000	**BS**	**CAN**	**(**	**8**	**H**	**X**	**h**	**x**
9	9	1001	**HT**	**EM**	**)**	**9**	**I**	**Y**	**i**	**y**
10	A	1010	**LE**	**SUB**	*****	**:**	**J**	**Z**	**J**	**z**
11	B	1011	**VT**	**ESC**	**+**	**;**	**K**	**[**	**K**	**{**
12	C	1100	**FF**	**FS**	**,**	**<**	**L**	****	**L**	**I**
13	D	1101	**CR**	**GS**	**-**	**=**	**M**	**]**	**m**	**}**
14	E	1110	**SO**	**RS**	**.**	**>**	**N**	**^**	**N**	**~**
15	F	1111	**SI**	**US**	**/**	**?**	**O**	**_**	**O**	**DEL**
The reading order is Column followed by Row										

Table-2.7b: [8×16] Matrix: Describing ASCII Codes (Zig-Zag along Row)

B_{10}	B_{16}	B_{10}	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		B_{16}	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		B_2	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	000	NUL	SOH	STX	ETX	EOT	ENG	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	1	001	DEL	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	2	010	SP	!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	3	011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	4	100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	5	101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	6	110	‘	a	b	C	d	e	f	G	h	i	j	k	l	m	n	o
7	7	111	p	q	r	S	t	u	v	w	x	y	z	{	l	}	~	DEL

The reading order is Row followed by Column

Control characters occupy the first 32 cells of the matrix while the upper and lower cases of the alphabets logically occupy the last four columns in the case of [16×8] Matrix or the last three rows of [8×16] Matrix. The numerals are logically placed in the columns just before the columns the alphabets occupy. The remaining characters are distributed among the rest of the cells at random.

Each of the cells can be identified by the following numbers which can be called codes as follows:

a. **Binary Number Format**: Each cell is read according to the rules of the matrix employed.
Examples:
The cell containing ‘A’ in [8×16] Matrix (Table 2.6a [65] or Table 2.7b) = 100 0001_2 = 65_{10}.
The cell containing ‘A’ in [16×8] Matrix (Table 2.4b [65] or Table 2.7a) = 100 0001_2 = 65_{10}.
NOTE: The binary equivalent of 65_{10} is 1000001_2.

b. **Decimal Number Format**: The decimal number of each cell is read from the appropriate Table.
Examples:
The cell containing ‘P’ in [8×16] Matrix (Table 2.7b or Table 2.6a [80]) = 101 0000_2 = 80_{10}.
The cell containing ‘P’ in [16×8] Matrix (Table 2.7a or Table 2.4b [80]) = 101 0000_2 = 80_{10}.
NOTE: The binary equivalent of 80_{10} is 1010000_2.

c. **Hexadecimal Number Format**: The decimal number of each cell is converted to its Hex equivalent.
Examples:
The cell containing ‘P’ in [8×16] Matrix (Table 2.7a or Table 2.4b [80]) = 101 0000_2
101 0000_2 = **0**101 0000 = 50_{16}.
The cell containing ‘P’ in [16×8] Matrix (Table 2.7a or Table 2.4b [80]) = 101 0000_2
101 0000_2 = **0**101 0000 = 50_{16}.

The binary number format is what is well known as the American Standard Code for Information Interchange (ASCII). There are other matrices that can be employed such as [32×4], [64×2], [128×1] and their conjugates. The popular choice of [16×8] may not be out of consideration for conciseness and space economy. In addition, the decision is guided by the following equation

$$n_c + n_r = n \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.3)$$

Where n_c = number of bits in the column.
n_r = number of bits in the row.
n = number of bits required to code the characters as previously defined.

For ASCII Codes, all possible Coding Format Matrices and their respective numbers of bits are calculated as shown in Table-2.8 as an example of the application of the theory here developed. The two conventionally accepted formats are shown on the black background.

Table-2.8: Calculation of n_c and n_r of the Coding Format Matrices Requirements

S/N	Main Matrices Requirements	Conjugate Matrices Requirements
1.	**[32×4] matrix:**	**[4×32] matrix:**

	$2^{nr} = 32 = 2^5$. Therefore, $\mathbf{n_r = 5}$ $2^{nc} = 4 = 2^2$. Therefore, $\mathbf{n_c = 2}$	$2^{nc} = 32 = 2^5$. Therefore, $\mathbf{n_c = 5}$ $2^{nr} = 4 = 2^2$. Therefore, $\mathbf{n_r = 2}$
2.	**[64×2] matrix:** $2^{nr} = 64 = 2^6$. Therefore, $\mathbf{n_r = 6}$ $2^{nc} = 2 = 2^1$. Therefore, $\mathbf{n_c = 1}$	**[2×64] matrix:** $2^{nc} = 64 = 2^6$. Therefore, $\mathbf{n_c = 6}$ $2^{nr} = 2 = 2^1$. Therefore, $\mathbf{n_r = 1}$
3.	**For [128×1] matrix (a Row Vector):** $2^{nr} = 128 = 2^7$. Therefore, $\mathbf{n_r = 7}$ $2^{nc} = 1 = 2^0$. Therefore, $\mathbf{n_c = 0}$	**For [1×128] matrix (a Column Vector):** $2^{nc} = 128 = 2^7$. Therefore, $\mathbf{n_c = 7}$ $2^{nr} = 1 = 2^0$. Therefore, $\mathbf{n_r = 0}$
4.	**[16×8] matrix:** $2^{nr} = 16 = 2^4$. Therefore, $\mathbf{n_r = 4}$ $2^{nc} = 8 = 2^3$. Therefore, $\mathbf{n_c = 3}$	**[8×16] matrix:** $2^{nc} = 8 = 2^3$. Therefore, $\mathbf{n_c = 3}$ $2^{nr} = 16 = 2^4$. Therefore, $\mathbf{n_r = 4}$
NOTE: The number of Matrices, M = Number of bits (n) required to sufficiently assign codes to all desired characters plus One (1). That is, M = (n + 1)		

It is important to note that in all the matrices used, sixteen possibilities per matrix are possible in labelling the cells of the matrices orderly without considering the possibility of random labelling. These sixteen possibilities are due to four different sequence of numbering the matrix cells (see Table-2.3). Therefore, if all the possible matrices are investigated, the total number of all options (N_{oT}) generating codes is given by

$$N_{oT} = 16M = (n+1)16 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.4)$$

However, only two of these possibilities are valid since the numbering of the matrix cells whether backward or forward, zig-zag or continuous should give the same value/figure as the ones obtained by reading the bits on rows/columns and/or columns/rows of the matrix. Hence, the useable options (N_{ou}) can be expressed thus,

$$N_{ou} = 2M = (n+1)2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.5)$$

3. Conclusion

From the materials presented in this paper, the following salient points have been established to make the assignment of Binary Codes to different computer characters logical and scientific other than random guessing:

a) To determine the number (n) of bits required for a given number of characters, Equation (2.2) is required to be solved. Hence, the total number of possible Code Format (Matrices) to be used is (n + 1).

b) To determine the dimensions of all the possible Code Format (Matrices) using the number of bits obtained from (a), Equation (2.3) and Table-2.8 are employed.

c) To determine the most suitable Matrix for the required computer characters, many other considerations are brought to the fore; such as:
   - At least, one side of the Matrix (row/column) must be equal or greater than 10 in order to accommodate the decimal numerals (0 to 9) completely in a row or column. This is why a Square Matrix is usually not preferred to a Rectangular Matrix when the side of the Square Matrix is less than 10. Otherwise, Square Matrix would have been the first consideration for the simple reason of conciseness of space.
   - Other peculiar considerations may apply depending on other requirements that the characters to be coded demand and the preference of the designer.

d) From the analysis presented in this paper, zig-zag numbering along the longer side of the Coding Format Matrix is preferred judging from the example of ASCII Codes since Table-2.7a is preferred to Table-2.7b.

e) The placement of the characters to be coded is done logically. For an example in the case of [16×8] Matrix of ASCII, the following considerations are systematically applied:
   - The first two columns are allotted to unprintable control characters arranged alphabetically downwards or upwards.
   - The last four columns are allotted to alphabets starting from uppercase occupying the first two columns of the four columns allotted and lowercase to the last two columns also arranged alphabetically downwards or upwards.
   - The third column is allocated to other characters in such a logical sequence as dictated by the nature of the remaining characters to be coded.
   - The remaining spaces left in columns which are not fully occupied are filled logically as per the remaining characters to be coded.

f) The sequence of assignment of binary codes is read first, from the side of movement/numbering followed by the binary number of the other side. That is, if the cells of the matrix are numbered serially in a zig-zag manner along the **row**, the sequence of assignment of binary codes is: first, binary number of the **row** follow by the binary number of the **column**. On the other hand, if the cells of the matrix are numbered serially in a zig-zag manner along the **column**, the sequence of assignment of binary codes is: first, binary number of the column follow by the binary number of the **row**.

This paper has succeeded in developing a logical procedure to assign codes to computer characters in a manner that errors can be traced logically and systematically, thereby reducing downtime to repair and eliminating possible further damage to equipment under quality assurance. By this hypothesis, every computer designer can now design their computers without the fear of one not being compatible with another from another designer on the bases of data interpretation.

References

1. *Fundamentals of Digital Systems by Olawale J. Omotosho, Franco-Ola Publishers, First Edition, 2012.*
2. *Digital Systems: Principles and Applications by Ronald J. Tocci and Neal S. Widmer, Von Hoffmann Press Inc & Prentice-Hall International, Inc., Seventh Edition, 1998.*
3. *Computer Architecture and Logic Design by Thomas C. Bartee, McGraw-Hill, Inc., International Edition, 1991.*
4. *Digital Computers and Machine Representation of Data,* www.math.wsu.edu/kcooper/M300/**computer**-binary

