Syllabus Content:

1.1.1 Number representation

- show understanding of the basis of different number systems and use the binary, denary and hexadecimal number system
- convert a number from one number system to another
- express a positive or negative integer in two's complement form
- show understanding of, and be able to represent, character data in its internal binary form depending on the character set used (Candidates will not be expected to memorise any particular character codes but must be familiar with ASCII and Unicode.)
- express a denary number in Binary Coded Decimal (BCD) and vice versa
- describe practical applications where BCD is used

Binary, Denary & Hexadecimal

The binary system on computers uses combinations of 0s and 1s.

In everyday life, we use numbers based on combinations of the digits between 0 and 9.

This counting system is known as decimal, denary or base 10.

(0 1 2 3 4 5 6 7 8 9)10

A number **base** indicates how many digits are available within a numerical system. Denary is known as **base 10** because there are ten choices of digits between 0 and 9.

For binary numbers there are only two possible digits available:

(0 or 1)₂

The binary system is also known as **base 2**.

All denary numbers have a binary equivalent and it is possible to **convert** between denary and binary.



Place values

Denary place values

Using the **denary** system, **6432** reads as six thousand, four hundred and thirty two. One way to break it down is as:

- **six** thousands
- four hundreds
- three tens
- two ones

Each number has a **place value** which could be put into columns. Each column is a power of ten in the base 10 system:

Thousands	Hundreds	Tens	Ones
1000s	100s	10s	1s
(10 [°])	(10 [°])	(10ˈ)	(10 [°])
6	4	3	2

Or think of it as:

 $(\mathbf{6} \times 1000) + (\mathbf{4} \times 100) + (\mathbf{3} \times 10) + (\mathbf{2} \times 1) = \mathbf{6432}$

Binary place values

You can also break a **binary** number down into place-value columns, but each column is a power of two instead of a power of ten.

For example, take a binary number like **1001**. The columns are arranged in multiples of 2 with the binary number written below:

Eights	Fours	Twos	Ones
8s	4s	2s	1s
(2³)	(2ຶ)	(2΄)	(2 [°])
1	0	0	1

By looking at the place values, we can calculate the equivalent denary number. That is: $(1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 8 + 0 + 0 + 1$



$$(1 \times 8) + (0 \times 4) + (0 \times 2) + (1 \times 1) = 8 + 1 = 9$$



Convert From Binary to Decimal	Powers of 2
conterent on onlary to been har	0
	$2^{\circ} = 1$
128 64 32 16 8 4 2 1 weight	$2^1 = 2$
0 0 1 0 0 1 1 0	$2^2 = 4$
	$2^3 = 8$
32 + 4 + 2 = 38 decimal	$2^4 = 16$
	$2^5 = 32$
128 64 32 16 8 4 2 1 weight	$2^6 = 64$
120 04 02 10 0 4 2 1 Weight	$2^{7} - 100$
1 1 0 0 1 1 0 1	$2_{0} = 128$
128 + 64 + 8 + 4 + 1 = 205 decimal	2° = 256

Converting binary to denary

To calculate a large **binary** number like **10101000** we need more place values of multiples of 2.

- $2^7 = 128$
- $2^6_{5} = 64$
- $2^5 = 32$
- $2^4 = 16$
- $2^3 = 8$
- $2^2 = 4$ • $2^1 = 2$
- $2^{\circ} = 2$
- $2^0 = 1$

In denary the sum is calculated as:

 $(1x2^7) + (0 x 2^6) + (1 x 2^5) + (0 x 2^4) + (1 x 2^3) + (0 x 2^2) + (0 x 2^1) + (0 x 2^0) = 168$ (1 x 128) + (0 x 64) + (1 x 32) + (0 x 16) + (1 x 8) + (0 x 4) + (0 x 2) + (0 x 1) =128 + 32 + 8 = 168

Converting denary to binary: Method 1

There are two methods for converting a <u>denary</u> (base 10) number to <u>binary</u> (base 2). This is method one.



Divide by two and use the remainder

Divide the starting number by 2. If it divides evenly, the binary digit is 0. If it does not - if there is a remainder - the binary digit is 1.

Play

A method of converting a denary number to binary

Worked example: Denary number 83

- 1. 83 ÷ 2 = 41 remainder 1
- 2. 41 ÷ 2 = 20 remainder 1
- 3. 20 ÷ 2 = 10 remainder 0
- 4. **10** ÷ 2 = 5 remainder **0**
- 5. **5** ÷ 2 = 2 remainder **1**
- 6. 2 ÷ 2 = 1 remainder 0
- 7. $1 \div 2 = 0$ remainder **1**

Put the remainders in reverse order to get the final number: 1010011.

64	32	16	8	4	2	1
1	0	1	0	0	1	1

To check that this is right, convert the binary back to denary: $(1 \times 64) + (0 \times 32) + (1 \times 16) + (0 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1) = 83$

Worked example: Denary number 122

- 1. 122 ÷ 2 = 61 remainder 0
- 2. 61 ÷ 2 = 30 remainder 1
- 3. 30 ÷ 2 = 15 remainder 0
- 4. **15** ÷ 2 = 7 remainder **1**
- 5. **7** ÷ 2 = 3 remainder **1**
- 6. 3 ÷ 2 = 1 remainder 1
- 7. $\mathbf{1} \div \mathbf{2} = \mathbf{0}$ remainder $\mathbf{1}$

Put the remainders in **reverse** order to get the final number: **1111010**.



(Paper 1. Sec 1.1.1) Number representation

128	64	32	16	8	4	2	1
0	1	1	1	1	0	1	0

To check that this is right, convert the binary back to denary:

$$(1 \times 64) + (1 \times 32) + (1 \times 16) + (1 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1) = 122$$

The binary representation of an even number always ends in 0 and an odd number in 1.

Converting denary to binary: Method 2

There are two methods for converting a <u>denary</u> (base 10) number to <u>binary</u> (base 2). This is method two. Take off the biggest 2ⁿ value you can

Remove the 2^n numbers from the main number and mark up the equivalent 2^n column with a 1. Work through the remainders until you reach zero. When you reach zero, stop and complete the final columns with 0s.



A method of converting a denary number to binary



Worked example: Denary number 84

First set up the columns of base 2 numbers. Then look for the highest 2ⁿnumber that goes into 84.

- 1. Set up the columns of base 2 numbers
- 2. Find the highest 2ⁿ number that goes into 84. The highest 2ⁿ number is 26 = 64
- 3. 84 64 = 20. Find the highest 2^n number that goes into 20. The highest 2^n number is 24 = 16
- 4. 20 16 = 4. Find the highest 2^n number that goes into 4. The highest 2^n number is 22 = 4
- 5. 4 4 = 0
- 6. Mark up the columns of base 2 numbers with a 1 where the number has been the highest 2ⁿnumber, or with a 0:

64	32	16	8	4	2	1
1	0	1	0	1	0	0

Result: 84 in denary is equivalent to 1010100 in binary.

To check that this is right, convert the binary back to denary: $(1 \times 64) + (0 \times 32) + (1 \times 16) + (0 \times 8) + (1 \times 4) + (0 \times 2) + (0 \times 1) = 84$

Binary combinations

These tables show how many binary combinations are available for each bit size.

One bit

		Binary number
		Place value 1
Denary	0	0
number	1	1

Maximum binary number = 1

Maximum denary number = 1

Binary combinations = 2



Two bit

		Binary	pattern
		Place value 2	Place value 1
	0	0	0
iary ibei	1	0	1
Den	2	1	0
	3	1	1

Maximum binary number = 11 Maximum denary number = 3 Binary combinations = 4

Three bit

Maximum binary number = 111

Maximum denary number = 7

Binary combinations = 8

		Binary pattern			
		Place Place Place value value value			
	-	4	2	-	
	0	0	0	0	
F	1	0	0	1	
nbe	<u>ë</u> 2	0	1	0	
nu	3	0	1	1	
<u>S</u>	4	1	0	0	
ena	5	1	0	1	
	6	1	1	0	
	7	1	1	1	

Hexadecimal Number System:

Fortunately, large binary numbers can be made much more compact—and hence easier to work with—if represented in base-16, the so-called hexadecimal number system. You may wonder: Binary numbers would also be more compact if represented in base-10—why not just convert them to decimal? The answer, as you will soon see, is that converting between binary and hexadecimal is exceedingly easy—much easier than converting between binary and decimal.

The Hexadecimal Number System

The base 16 hexadecimal has 16 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F). Note that the single hexadecimal symbol A is equivalent to the decimal number 10, the single



symbol B is equivalent to the decimal number 11, and so forth, with the symbol F being equivalent to the decimal number 15.

Just as with decimal notation or binary notation, we again write a number as a string of symbols, but now each symbol is one of the 16 possible hexadecimal digits (0 through F). To interpret a hexadecimal number, we multiply each digit by the power of **16** associated with that digit's position.

For example, consider the hexadecimal number 1A9B. Indicating the values associated with the positions of the symbols, this number is illustrated as:



The one main disadvantage of binary numbers is that the binary string equivalent of a large decimal base -10 number, can be quite long. When working with large digital systems, such as computers, it is common to find binary numbers consisting of 8, 16 and even 32 digits which makes it difficult to both read and write without producing errors especially when working with lots of 16 or 32-bit binary numbers. One common way of overcoming this problem is to arrange the binary numbers into groups or sets of four bits (4-bits). These groups of 4-bits use another type of numbering system also commonly used in computer and digital systems called Hexadecimal Numbers.

REPRESENTING INTERGERS AS HEXADECIMAL NUMBERS:

The base 16 notational system for representing real numbers. The digits used to represent numbers using hexadecimal notation are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

"H" denotes hex prefix.

Examples:

(i) $28_{16} = 28_{H} = 2 \times 16^{1} + 8 \times 16^{0} = 40$

= 32 + 8 = 40

- (ii) 2 F16 = 2 F_H = 2 × 16 +15 × 1= 47
- (iii) BC1216 = BC12_H = $11 \times 16^{3} + 12 \times 16^{2} + 1 \times 16^{1} + 2 \times 16^{0} = 48146$



Hexadecimal Numbers in Computing

There are two ways in which hex makes life easier.

- The first is that it can be used to write down very large integers in a compact form.
- For example, (**A D 4 5**)₁₆ is shorter than its decimal equivalent (44357)₁₀ and as values increase the difference in length becomes even more pronounced.

Converting Binary Numbers to Hexadecimal Numbers.

Let's assume we have a binary number of: 01010111 The binary number is 01010111 We will break number into 4 bits each as

0101 0111

Then we will start with the right side 4 bits Starting from extreme right number for 0101

for 0111

0X2³+1X2²+0X2¹+1X2⁰

0X2³+1X2²+1X2¹+1X2⁰

0X8+1X4+0X2+1X1

0X8+1X4+1X2+1X1

0+4+0+1=5

0+4+2+1=7

5

So Hexadecimal number is 57

Converting Hexadecimal Numbers to Binary Numbers

7

To convert a hexadecimal number to a binary number, we reverse the above procedure. We separate every digit of hexadecimal number and find its equivalent binary number and then we write it together.



Example 1.2.4

To convert the hexadecimal number 9F216 to binary, each hex digit is converted into binary form.

9 F 2 ₁₆ = (1001 1111 0010)₂ 9 =1001 F=1111 2=0010

So Binary equivalent of Hexadecimal number is: 9F2= 100111110010

Problems 1.2.6

Convert hexadecimal 2BF9 to its binary equivalent. Convert binary 110011100001 to its hexadecimal equivalent. (Below is working area)

Converting a Hexadecimal Number to a (Denary) Decimal Number

To convert a hexadecimal number to a decimal number, write the hexadecimal number as a sum of powers of 16. For example, considering the hexadecimal number 1A9B above, we convert this to decimal as:



Converting a (Denary) Decimal Number into Hexadecimal Number

The easiest way to convert from decimal to hexadecimal is to use the same division algorithm that you used to convert from decimal to binary, but repeatedly dividing by 16 instead of by 2. As before, we keep track of the remainders, and the sequence of remainders forms the hexadecimal representation.

For example, to convert the decimal number **746** to hexadecimal, we proceed as follows:

 Remainder

 16 | 746

 | 46
 10 =
 ▲

 | 2
 14 =
 E

 | 0
 | 2

We read the number as last is first and first is last. **2EA**

So, the decimal number 746 = 2EA in hexadecimal



Conversion of –Ve Denary number to Binary:

What is -65_{10} in binary?

Two's complement allows us to represent signed negative values in binary, so here is an introductory demonstration on how to convert a negative decimal value to its negative equivalent in binary using two's complement.

Step 1: Convert 65d to binary. Ignore the sign for now. Use the absolute value. The absolute value of -65 is 65.

65 --> 01000001 binary Step 2: Convert 01000001 to its one's complement.

01000001 --> 10111110

Step 3: Convert 10111110b to its two's complement by adding 1 to the one's complement.

10111110 + 1 ------10111111 <---- Two's complement

10111111b is -65 in binary. We know this it true because if we add 01000001 (+65) to 1011111b (-65) and *ignore the carry bit*, the sum is 0, which is what we obtain if we add +65 + (-65) = 0.

01000001 +65 + 10111111 -65 10000000b 0 denary

Ignore the carry bit for now. What matters is that original number of bits (D7-D0) are all 0.

We will examine signed binary values in more detail later. For now, understand the difference between one's complement and two's complement and practice converting between them.



One's Complement

If all bits in a byte are inverted by changing each **1** to **0** and each **0** to **1**, we have formed the one's complement of the number.

Original	One's Complement	
10011001	-> 01100110	
1000001	-> 01111110	
11110000 -	-> 00001111	
11111111 -	-> 0000000	
0000000	-> 11111111	Converting to one's complement.

And that is all there is to it! One's complement is useful for forming the two's complement of a number.

Two's Complement (Binary Additive Inverse)

The two's complement is a method for representing positive and negative integer values in binary. The useful part of two's complement is that it automatically includes the sign bit. Rule: To form the two's complement, add 1 to the one's complement.

Step 1: Begin with the original binary value

10011001 Original byte

Step 2: Find the one's complement

01100110 One's complement

Step 3: Add 1 to the one's complement

01100110 One's complement + 1 Add 1

01100111 <--- Two's complement



Two's Complement

First, find the one's complement of a value, and then add 1 to it.



USE OF HEXADECIMAL NUMBER IN COMPUTER REGISTERS AND

MAIN MEMORY:

Computers are comprised of chips, registers, transistors, resistors, processors, traces, and all kinds of things. To get the binary bits from one place to the next, software programmers convert binary to hex and move hex values around. In reality, the computer is still shoving 1's and 0's along the traces to the chips.

There are two important aspects to the beauty of using Hexadecimal with computers: First, it can represent 16-bit words in only four Hex digits, or 8-bit bytes in just two; thus,

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



by using a numeration with more symbols, it is both easier to work with (saving paper and screen space) and makes it possible to understand some of the vast streams of data inside a computer merely by looking at the Hex output. This is why programs such as DEBUG, use only Hexadecimal to display the actual Binary bytes of a Memory Dump rather than a huge number of ones and zeros!

The second aspect is closely related: Whenever it is necessary to convert the Hex representation back into the actual Binary bits, the process is simple enough to be done in your own mind.

For example, **FAD7 hex is 1111 1010 1101 0111 (F=1111, A=1010, D=1101, 7=0111)** in Binary. The reason one might wish to do this is in order to work with "logical" (AND, OR or XOR) or "bit-oriented" instructions (Bit tests, etc.) which may make it easier (at times) for a programmer to comprehend.

For example, if you wanted to logically AND the hex number FAD7 with D37E, you might have a difficult time without first changing these numbers into Binary. If you jot them out in Binary on scratch paper, the task

will be much easier:

FAD7(hex)	1111	1010	1101	0111
D37E(hex)	1101	0011	0111	1110
ANDing gives	1101	0010	0101	0110
Answer (in hex)	D	2	5	6

Converting Between Bases

To convert from denary to hexadecimal, it is recommended to just convert the number to binary first, and then use the simple method above to convert from binary to hexadecimal.





BCD Binary Coded Decimals:

In computing and electronic systems, **binary-coded decimal** (**BCD**) is a class of binary encodings of decimal numbers where each decimal digit is represented by a fixed number of bits, usually four or eight. Special bit patterns are sometimes used for a sign or for other indications (e.g., error or overflow).

BCD was used in many early <u>decimal computers</u>, and is implemented in the instruction set of machines such as the <u>IBM System/360</u> series and its descendants and <u>Digital's VAX</u>. Although BCD *per se* is not as widely used as in the past and is no longer implemented in computers' instruction sets^[dubious - discuss], decimal <u>fixed-point</u> and <u>floating-point</u> formats are still important and continue to be used in financial, commercial, and industrial computing

As most computers deal with data in 8-bit <u>bytes</u>, it is possible to use one of the following methods to encode a BCD number:

- **Unpacked**: each numeral is encoded into one byte, with four bits representing the numeral and the remaining bits having no significance.
- Packed: two numerals are encoded into a single byte, with one numeral in the least significant <u>nibble</u> (bits 0 through 3) and the other numeral in the most significant nibble
 - The Denary number 8 5 0 3 could be represented by one BCD digit per byte
 - 00001000 00000101 00000000 000000011 (Unpacked)
 - Denary Number 8 5 0 3 represented by One BCD per nibble

1000 0101 0000 0011 (Packed as 1000010100000011)

e.g. 398602 in BCD

Answer: 3 = 0011 9 = 1001 8 = 1000 6 = 0110 0 = 0000 2 = 0010

So 398602 = 001110011000011000000010 (in BCD)

Note: All the zeros are essential otherwise you can't read it back.

But do not get confused, *binary coded decimal* is not the same as hexadecimal. Whereas a 4-bit hexadecimal number is valid up to F_{16} representing binary 1111₂, (decimal 15), binary coded decimal numbers stop at 9 binary 1001₂



Application

The BIOS in many personal computers stores the date and time in BCD because the MC6818 real-time clock chip used in the original IBM PC AT motherboard provided the time encoded in BCD. This form is easily converted into ASCII for display.

There are a number of applications where BCD can be used. The obvious type of application is where denary digits are to be displayed, for instance on the screen of a calculator or in a digital time display.

A somewhat unexpected application is for the representation of currency values. When a currency value is written in a format such as \$300.25 it is as a fixed-point decimal number (ignoring the dollar sign). It might be expected that such values would be stored as real numbers but this cannot be done accurately.

IBM and BCD IBM used the terms binary-coded decimal and BCD for six-bit alphameric codes that represented numbers, upper-case letters and special characters. Some variation of BCD was used in most early IBM computers, including the IBM 1620, IBM 1400 series, and non-Decimal Architecture members of the IBM 700/7000 series. With the introduction of System/360, IBM replaced BCD with 8-bit EBCDIC

ASCII code:

If text is to be stored in a computer it is necessary to have a coding scheme that provides a unique binary code for each distinct individual component item of the text. Such a code is referred to as a character code.

The scheme which has been used for the longest time is the ASCII (American Standard Code for Information Interchange) coding scheme. This is an internationally agreed standard. There are some variations on ASCII coding schemes but the major one is the 7-bit code. It is customary to present the codes in a table for which a number of different designs have been used.

The full table shows the 27 (128) different codes available for a 7-bit code. You should not try to remember any of the ind ividua I codes but the re are certain aspects of the coding scheme which you need to understand.



ASCII TABLE

0 0 0 0 0 0 0 96 60 1100000 10 1 1 1 1 155 101 2 2 10 2 110000 10000 110000 14 a 2 2 10 2 155AF OT TEXT 50 32 110010 64 1100001 143 c 4 4 100 4 EMD OF TRANSMISSION 52 34 110100 65 5 101 66 1100011 145 e 6 6 110 6 ACXMONEDGE 54 36 110110 65 5 1001 66 1100011 46 110010 144 7 8 10000 10 BACKSRACE 56 38 11100 71 9 105 66 1100011 151 1 11 8 10101 10 MACKSRACE 59 38 <td< th=""><th>Decimal</th><th>Hexadecimal</th><th>Binary</th><th>Octal</th><th>Char</th><th>Decimal</th><th>Hexadecimal</th><th>Binary</th><th>Octal</th><th>Char</th><th>Decimal</th><th>Hexadecimal</th><th>Binary</th><th>Octal</th><th>Char</th></td<>	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
1 1 1 1 1 1 1 97 61 110000 141 # 3 3 11 3 BRD OF TEXT 50 32 110001 62 2 98 62 110001 142 b 3 3 11 3 BRD OF TEXT 51 33 11001 65 2 98 62 110001 142 b 4 4 100 4 100 64 11000 64 4 100 65 11001 142 b 6 111 7 IRLL 55 37 11011 66 6 1001 144 e 7 111 7 IRLL 55 37 11011 67 7 103 67 11001 144 p 9 9 1001 11 Inverse 56 38 111001 73 104 68 110001 150 i 11001 151 i 11001 152 i 11010 1101 153 i 110101 153 i 110101 153 i 110101 153 i 110101 153 i <	0	0	0	0	[MULL]	48	30	110000	60	0	96	60	1100000	140	1.0
2 10 2 START OF TEXT1 50 32 110010 62 2 98 62 110010 143 b 4 4 100 4 HEND OF TRAINSESSON 52 34 110010 163 3 99 63 110010 144 d 5 101 5 100 64 110010 144 d 6 6 110 6 (ACWOWLEDGE) 54 35 110110 66 1002 66 110010 145 e 7 71 7 71 7 110 7 110 7 110 7 110 7 110 140 68 110100 72 106 6A 110100 151 1 110 141 100 151 11000 151 1 11000 151 1 10010 151 1 10010 151 1 10010 151 100000	1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	-18
3 3 11 3 END OF TEXT 51 33 110011 613 8 99 63 1100011 143 c 5 5 101 5 [HOCMET] 53 35 110101 65 5 100 64 110010 145 e 6 6 101 6 (HOCMETEXT) 55 37 110110 65 5 101 65 110010 145 e 7 111 7 (BEL) 55 37 110111 67 7 103 67 110011 143 g 9 9 1001 11 PREXENCES 56 38 111001 72 104 68 110100 155 i 112 C 1001 13 (PERTAL, TAN) 59 38 11010 74 106 60 11010 74 106 60 11010 74 107 68 11010 155 i 11010 75 107 68 110101 155 m 115 11010 155 i 11010 155 i 11010 155 i 11010 155 i 11010 155	2	2	10	2	(START OF TEXT)	50	32	110010	6.2	2	98	62	1100010	142	b
4 4 100 4 100 64 11000 64 110000 1144 6 5 5 101 6 1100 65 11000 65 110010 155 1001 65 110010 156 110010 66 1002 66 1100100 110010 1100100	3	3	11	3	(END OF TEXT)	51	33	110011	63	3	99	63	1100011	143	C
5 5 101 5 101010 65 10010 145 e 7 7 111 7 1821 55 37 110110 66 6 11000110 146 f 9 9 10011 11 100200781 78 38 110000 70 8 104 66 1100100 11 9 9 1001 11 1002200781 78 39 111001 72 106 66 1101010 151 1 10 A 1001 12 147 148 11100 72 106 66 1101010 151 1 112 C 1100 14 1604770077 61 30 111100 75 = 109 60 101010 155 n 15 F 1111 17 18770077 61 37 111107 7 111 66 110111155 n 110001 112 110001 112 110001 112 110001 1111017 <td< td=""><td>4</td><td>4</td><td>100</td><td>4</td><td>[END OF TRANSMISSION]</td><td>52</td><td>34</td><td>110100</td><td>6-4</td><td>4</td><td>100</td><td>64</td><td>1100100</td><td>144</td><td>d</td></td<>	4	4	100	4	[END OF TRANSMISSION]	52	34	110100	6-4	4	100	64	1100100	144	d
66 61 110 6 MCNOWLEDGE[54 36 11011 66 6 102 66 110011 114 7 8 8 1000 10 BACKSARCE] 56 38 111001 70 8 104 68 110010 151 i 10 A 1010 12 LUREFEED 58 38 111001 72 106 6A 1101010 152 j 12 C 1100 14 FORM FEED 60 3C 111107 7 113 66 1101010 153 k 13 D 1101 15 (EARMGE RETWRI) 61 30 11107 7 111 66 110110 156 110110 156 110110 156 110110 156 110111 157 111 1000 160 100110 157 111 1000 160 100110 150 100110 150	5	5	101	5	(ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
7 111 7 111 7 111 67 7 103 67 110011 117 9 9 9 1001 11 μ (0825MCE) 56 38 11000 70 8 104 68 110100 155 h 10 A 1010 12 μ (MFEE) 58 33 11010 72 : 106 66 110101 152 i 11 8 1011 13 μ (MFEC) 59 38 11010 73 : 107 68 110101 155 i 12 C 1100 16 β (MMFEC) 60 32 11110 75 : 109 65 1101101 155 n 13 D 11001 15 μ (MARTAR) 63 37 11111 77 111001 11000 1101 155 n 110001 100 11010 11000 11010 11000 11010 11000 11000 11000 11000 11000 110000	6	6	110	6	(ACKNOWLEDGE)	54	36	110110	66	6	102	66	1100110	146	f
8 1000 100 BACKSPACE/ INTERCATAR 56 38 111007 70 8 104 68 1101000 150 n 10 A 1010 12 AUME FEED 58 33 111001 72 1 106 6A 110101 153 k 12 C 1100 14 FRAMAGE FETDRI 60 3C 111100 74 108 6C 110101 155 k 111110 155 k 107 68 110101 155 m 111110 15 k 10011 155 m 111110 76 110 66 1101101 155 m 11111 156 111110 156 1101111 156 111110 156 111110 156 111110 156 111110 156 111110 156 11111000 154 11111000 154 11111000 154 11111000 154 111110000 154 <t< td=""><td>7</td><td>7</td><td>111</td><td>7</td><td>(BELL)</td><td>55</td><td>37</td><td>110111</td><td>67</td><td>7</td><td>103</td><td>67</td><td>1100111</td><td>147</td><td>g</td></t<>	7	7	111	7	(BELL)	55	37	110111	67	7	103	67	1100111	147	g
9 9 1001 11 <i>IDORALOWING TRUE</i> 57 39 111001 71 9 105 69 110100 151 i 11 8 1001 12 <i>IME FEED</i> 58 3A 11010 73 : 106 6A 110100 152 i 12 C 1100 14 <i>IPERFULT TWI</i> 61 3D 111101 73 : 106 6B 1101100 154 i 11111 107 6B 1101100 154 i 10110 155 m 1111 17 1111 6C 110110 155 m 1111 1000 100 1000 100 11111 77 111 6E 110110 11111 77 1111000 11110 11111 77 1111000 11111 77 1110000 116 n 1111000 116 1111000 116 1111000 116 11110000 116 1111000	8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
10 A 1010 12 [JME FEED] 58 3A 111010 72 i 106 6A 110101 152 j 11 B 1011 13 [JPRFCALTAN] 59 38 111010 72 i 106 6A 110100 152 j 12 C 1100 15 [JRRARLAGE RETAR] 60 3C 111100 74 i 108 6C 110110 155 n 14 E 1110 15 [JRRARLAK ESCAPE] 64 40 1000000 100 111 111 77 111 6E 110101 157 0 16 10 10000 20 [DRVAE CONTROL] 65 41 1000001 100 111 111 77 1110001 1110001 1110001 1110001 1110001 1110001 1110001 1110001 1110001 1110001 1110001 1110001 1110001 111001 1110010 1110010 1110010 1110010 1110010 11110010 11100100 1110010 </td <td>9</td> <td>9</td> <td>1001</td> <td>11</td> <td>(HORIZONTAL TAB)</td> <td>57</td> <td>39</td> <td>111001</td> <td>71</td> <td>9</td> <td>105</td> <td>69</td> <td>1101001</td> <td>151</td> <td>1</td>	9	9	1001	11	(HORIZONTAL TAB)	57	39	111001	71	9	105	69	1101001	151	1
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	10	A	1010	12	(LME FEED)	58	3A	111010	72	1	106	6A .	1101010	152	1
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	11	8	1011	13	(VERTICAL TAB)	59	38	111011	73	1	107	68	1101011	153	k
13 0 1101 15 [CARBURGE FETWRN] 61 300 11101 75 = 109 6D 101101 155 m 15 F 1111 17 [SHHT OUT] 62 3E 111110 76 > 110 6E 110110 155 m 16 10 10000 20 [DATALINK ESCRF] 64 40 1000000 112 70 1110001 160 p 17 11 10010 21 [DEVACE CONTROL]] 65 41 1000000 100 B 114 72 1110001 160 p 18 12 10010 22 [DEVACE CONTROL]] 66 42 1000010 115 73 1110010 165 s 120 14 10310 24 [DEVACE CONTROL]] 67 43 1000010 D 116 74 1110010 155 s 110010 155 s 11010 155 117 75 1110010 156 117 75 1110010 165	12	c	1100	14	(FORM FEED)	60	3C	111100	74	<	108	6C	1101100	154	1
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	13	D	1101	15	(CARRIAGE RETURN)	61	3D	111101	75	-	109	6D	1101101	155	m
15 F 1111 17 $SMFTNI$ 63 3F 11111 7 111 6F 110111 157 0 16 10 10000 20 $[DEWGE CONTROL 2]$ 65 41 100000100 $@$ 112 70 1110001 60 p 17 11 10001 22 $[DEWGE CONTROL 2]$ 66 42 1000001 101 A 113 71 1110001 162 r 19 13 10010 22 $[DEWGE CONTROL 3]$ 66 42 100001 103 C 114 72 1110010 162 r 20 14 10100 24 $[DEWGE CONTROL 4]$ 68 44 1000101 104 D 116 74 1110101 65 v 21 15 10110 26 $[SWKOPKOWDS DOUS DOUE] 70 46 1000110 106 F 118 76 1110101 05 w 22 16 10101 26 [SWKOPKOWDS DOUE] 72 48 10001001 111 1 120 78 11110010 100 1000$	14	E	1110	16	[SHIFT OUT]	62	38	111110	76	>	110	6E	1101110	156	n
16 10 10000 20 0DMA LUXE SECAPE) 64 40 1000000 112 70 1110000 160 p 17 11 10001 21 [DEWCE CONTROL 2] 65 41 100000 101 A 113 71 1110000 161 q 18 12 10010 22 [DEWCE CONTROL 2] 66 42 100000 100 A 113 71 1110000 162 r 19 13 10011 23 [DEWCE CONTROL 3] 67 43 100000 100 D 116 73 111001 163 s 20 14 10100 26 [STMCHRONOUS INLE] 70 46 1000110 100 F 118 76 111010 165 u 21 15 1010 26 [STMCHRONOUS INLE] 71 47 1000110 112 78 111010 165 u 11010 16 u 10010 112 78 1111000 100 1100 110 110	15	F	1111	17	(SHIFT IN)	63	3F	1111111	77	2	111	6F	1101111	157	0
17 11 10001 21 [DEVICE CONTROL 1] 65 41 1000001 101 A 113 71 1110001 161 0 18 12 10010 22 [DEVICE CONTROL 2] 66 42 1000010 102 B 114 72 1110010 165 r 20 14 10100 24 [DEVICE CONTROL 4] 66 44 1000010 100 C 115 73 1110010 165 r 21 15 10110 24 [DEVICE CONTROL 4] 68 44 100010 104 D 116 74 1110100 164 t 22 16 10110 26 [SWOWROMOUS INL] 70 46 100010 100 H 120 78 111000 17 x 23 17 10111 37 [EWO OF FRANS, BUCK] 73 49 1001001110 H 120 78 1111000 17 x 24 18 11010 33 [ESUSEFTUTTE] 75	16	10	10000	20	(DATA LINK ESCAPE)	64	40	1000000	100	8	112	70	1110000	160	P
18 12 10010 22 [DEVICE CONTROL 2] 66 42 100010 102 8 114 72 1110010 162 r 19 13 10011 23 [DEVICE CONTROL 3] 67 43 100010 103 C 115 73 1110010 162 r 20 14 10100 24 (DEVICE CONTROL 3) 68 44 1000100 106 T 116 74 1110010 165 u 21 15 10101 25 [MEGATVE ACKNOWLEGCE] 69 45 100010 106 F 118 76 111010 165 u 22 19 11001 31 70 46 1000110 10 H 120 78 111001 165 v 23 17 1011 32 [SUSSTUTE] 74 4A 1001001 111 121 79 111001 70 x 111000 170 x 111000 170 x 121 76 1111000 111100 <td>17</td> <td>11</td> <td>10001</td> <td>21</td> <td>IDEVICE CONTROL 11</td> <td>65</td> <td>41</td> <td>1000001</td> <td>101</td> <td>A</td> <td>113</td> <td>71</td> <td>1110001</td> <td>161</td> <td>q</td>	17	11	10001	21	IDEVICE CONTROL 11	65	41	1000001	101	A	113	71	1110001	161	q
19 13 10011 23 (DEVICE CONTROL 3) 67 43 1000011 103 C 115 73 1110010 165 y 20 14 10100 24 (DEVICE CONTROL 4) 68 44 1000100 104 D 116 74 1110010 165 u 21 15 10110 25 (MEGATWE ACKNOWLEDGE) 69 45 1000101 106 F 118 76 1110010 165 u 22 16 10111 27 (EWG OF TAALS ELOCK) 71 47 1000111 107 G 119 77 1110011 167 w 24 18 11000 30 (CAVCEL) 72 48 1001000 110 H 120 78 1111001 172 z 25 19 11010 31 (ENO OF MEDIUN) 73 49 1001001 11 1 121 79 1111001 172 z 7A 1111001 172 z 7A 1111001 172	18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
20 14 10100 24 (DEWCE CONTROL 4) 68 44 1000100 104 D 116 74 1110100 164 t 21 15 10101 25 (MEGATIVE ACKNOWLEDGE) 69 45 1000101 105 E 117 75 1110100 164 u 23 17 10111 26 (SYNCHROMOUS TOLE) 70 46 1000101 06 F 118 76 1110101 166 u 24 18 11000 30 (CANCEL) 72 48 1001001 11 1 120 78 1110001 70 x 25 19 11001 31 [END OF MEDIUM] 73 49 1001001 111 1 121 79 1111001 70 x 26 1A 11001 34 [HLE SEMANZOR] 76 4C 1001101 114 L 123 78 1111001 77 1111001 77 1111001 77 1111001 77 1111101 76	19	13	10011	23	IDEVICE CONTROL 31	67	43	1000011	103	C	115	73	1110011	163	5
21 15 10101 25 (MEGATIVE ACKNOWLEDGE) 69 45 1000101 105 E 117 75 1110101 165 u 22 16 10110 26 (STMCNROWOUS IOLE) 70 46 1000111 107 6 118 76 1110110 165 u 24 18 11000 30 (CANCEL) 72 48 1001000 100 H 120 78 1111010 167 x 25 19 11001 31 (ENO OF MEDUM) 73 49 1001000 100 H 120 78 1111000 170 x 26 1A 11010 31 (ESCAPE) 75 48 100101011 15 122 7A 111000 177 x 22 10 111010 17 x 23 76 1111010 165 u 124 7C 111100 17 x 124 7C 1111001 17 111100 17 111100 17 111100 17	20	14	10100	24	(DEVICE CONTROL 4)	68	44	1000100	104	D	116	74	1110100	164	t
22 16 10110 26 (STNCHRONOUS TOLE) 70 46 1000110 106 F 118 76 1110110 166 v 23 17 10111 27 (ENG OF TRANS BLOCK) 71 47 1000111 107 G 119 77 1110111 167 w 24 18 11000 30 (CANCEL) 72 48 1001000110 H 120 78 1111001 70 x 25 19 11001 31 (END OF MEDRUM) 73 49 1001001 111 I 121 79 1111001 77 x 26 1A 11010 32 (SUSSTITUTE) 74 4A 1001010 112 J 122 7A 1111001 73 x 27 18 11011 33 (ESCAPE) 76 4C 100110 115 K 123 78 1111001 74 1 124 7C 1111001 74 x 1111001 75 x 1111010 175 x 11111010 175 x	21	15	10101	25	INEGATIVE ACKNOWLEDGET	69	45	1000101	105	E	117	75	1110101	165	
23 17 10111 27 ENG OF TRAMS. BLOCK! 71 47 1000111 107 G 119 77 1110111 167 w 24 18 11000 30 [CANCEL] 72 48 1001000110 H 120 78 1111000170 x 25 19 11001 31 [ESCOP MEDRUM] 73 49 1001001111 1 120 78 1111001170 x 26 1A 11010 32 [SUBSTRUTE] 74 4A 1001010112 122 7A 1111001172 z 27 18 11010 34 [ESCAPE] 75 48 100101113 K 123 78 1111001 72 z 28 1C 1100 34 [PLE SEMAATOR] 76 4C 1001101 15 122 70 1111101 75 3 30 1E 11110 36 [RECORD SEPAATOR] 78 4E 1001111 10 125 70 1111101 76 - 127 </td <td>22</td> <td>16</td> <td>10110</td> <td>26</td> <td>(SYNCHRONOUS IDLE)</td> <td>70</td> <td>46</td> <td>1000110</td> <td>106</td> <td>F</td> <td>118</td> <td>76</td> <td>1110110</td> <td>166</td> <td>v</td>	22	16	10110	26	(SYNCHRONOUS IDLE)	70	46	1000110	106	F	118	76	1110110	166	v
24 18 11000 30 [CANCEL] 72 48 1001000 10 H 120 78 1111000 170 x 25 19 11001 31 [END OF MEDUNH] 73 49 1001001 111 121 79 1111001 171 y 26 1A 11010 31 [END OF MEDUNH] 73 49 1001001 111 121 79 1111001 171 y 27 18 11011 33 [ESCAPE] 75 48 100101 113 K 123 78 1111001 77 y 28 1C 11100 34 [FLE SEPARATOR] 77 4D 1001101 15 M 125 7D 1111101 75 } 30 1E 11110 36 [RECORD SEPARATOR] 78 4E 1001110 16 N 126 7E 1111101 75 } 1111101 75 1111101 7F 1111101 7F 11111101 7F 1111110 7F	23	17	10111	27	IENG OF TRANS. BLOCK!	71	47	1000111	107	G	119	77	1110111	167	W
25 19 11001 31 END OF MEDIUM! 73 49 1001001 111 1 121 79 1111001 171 y 26 1A 11010 32 [SUSSTITUTE] 74 4A 100101 112 j 122 7A 1111001 171 y 27 18 11011 33 [ESCAPE] 75 48 1001101 113 K 123 76 111100 172 z 28 1C 11100 34 [PLE SEPARATOR] 76 4C 1001100 114 L 124 7C 111100 174 [29 1D 11101 35 [GACUP SEPARATOR] 77 4D 1001101 115 M 125 7D 1111101 175 j 30 1E 11110 36 [RECORD SEPARATOR] 78 4E 1001101 116 N 126 7E 111110 176 - 31 1F 11113 (WOR SEPARATOR) 79 4F 1001101 117 0 127 7F 1111101 176 - 32 20 100001 42 * 82 52 101001 122	24	18	11000	30	(CANCEL)	72	48	1001000	110	H	120	78	1111000	170	×
26 1A 11010 32 (SUBSTRUTE) 74 4A 1001010 112 J 122 7A 1111010 172 z 27 18 11011 33 (ESCAPE) 75 48 100110 113 K 123 78 111101 173 { 28 1C 11100 34 (FLE SEMANTOR) 76 4C 1001100 14 L 124 7C 111100 174 { 29 1D 11100 34 (FLE SEMANTOR) 77 4D 100110 14 L 124 7C 111100 174 { 30 1E 1110 36 (RECOR) SEMANTOR) 78 4E 100110 16 N 125 7D 111110 176 - 31 1F 11111 37 (UWT SEMANTOR) 78 4E 100100 120 P 111110 176 4 181 51 1010001 121 Q 127 7F 1111111 177 IDELJ	25	19	11001	31	(END OF MEDIUM)	73	49	1001001	111	1	121	79	1111001	171	v
27 1B 11011 33 [ESCAPE] 75 4B 1001011 113 K 123 7B 1111011 173 { 28 1C 11100 34 (PLE SERAANOR) 76 4C 1001010 114 L 124 7C 111100 174 29 1D 1110 35 (EACUP SERAANOR) 77 4D 1001101 115 M 125 7D 111100 174 30 1E 11110 36 (EACORD SERAANOR) 77 4D 1001101 115 M 125 7D 1111101 175 ~ 31 1F 11111 37 (UMT SEBARATOR) 79 4F 1001111 117 0 127 7F 1111110 176 ~ 32 20 100000 40 (SA4CE) 80 50 1010001 120 P 127 7F 1111111 177 (DEL) 33 21 100010 42 * 82 52 101001 122 R 101001 125 0 34 22 100101 45 % 85 55 1010101 125 0 38 26 <td>26</td> <td>14</td> <td>11010</td> <td>32</td> <td>(SUBSTITUTE)</td> <td>74</td> <td>44</td> <td>1001010</td> <td>112</td> <td>1</td> <td>122</td> <td>74</td> <td>1111010</td> <td>172</td> <td>2</td>	26	14	11010	32	(SUBSTITUTE)	74	44	1001010	112	1	122	74	1111010	172	2
28 1C 11100 34 (FLE SEPARATOR) 76 4C 1001100 114 L 124 7C 1111100 174 [29 1D 11101 35 (GROUP SEPARATOR) 77 4D 1001101 115 M 125 7D 1111100 174 [30 1E 11110 36 (RECORD SEPARATOR) 78 4E 1001101 115 M 126 7E 1111100 17 - 31 1F 11111 37 (UWT SEPARATOR) 79 4F 1001101 111 17 0 127 7F 111110 17 (DEL) 32 20 100000 40 (SPACE) 80 50 1010000 120 P 127 7F 1111111 177 (DEL) 33 21 100001 41 1 81 51 101000 121 Q 127 7F 1111111 177 (DEL) 34 22 100010 45 83 53	27	18	11011	33	IESCAPEI	75	48	1001011	113	K	123	7B	1111011	173	1
29 1D 11101 35 (GROUP SEPARATOR) 77 4D 1001101 115 M 125 7D 111101 175) 30 1E 1110 36 (RECORD SEPARATOR) 78 4E 1001101 116 N 126 7E 1111101 176 - 31 1F 11111 37 (UWT SEPARATOR) 79 4F 1001111 117 0 127 7F 1111101 76 - 32 20 100000 40 (SAACE) 80 50 1010000 120 P 127 7F 1111111 177 (DEL) 33 21 100010 41 1 81 51 1010001 122 R 35 23 100011 43 # 83 53 1010101 123 S 36 24 100100 44 \$ 85 55 1010101 125 U 38 26 100100 44 \$ 87 57 1010111 127 W	28	10	11100	34	IFILE SEPARATORI	76	4C	1001100	114	L	124	7C	1111100	174	1
30 1E 11110 36 (RECORD SEPARATOR) 78 4E 1001110 116 N 126 7E 111110 176 31 1F 11111 37 (UWT SEPARATOR) 79 4F 1001111 117 0 127 7F 1111110 176 - 32 20 100000 40 (SPACE) 80 50 1010000 120 P 33 21 100001 41 81 51 1010001 12 Q 34 22 100010 42 * 82 52 1010010 122 R 35 23 100010 44 5 84 54 1010100 123 S 36 24 100100 44 5 85 55 1010101 126 V 38 26 100101 45 % 85 55 1010101 127 W 41 29 101001 51 1 89 59 1011001 131 <	29	10	11101	35	IGROUP SEPARATORI	77	4D	1001101	115	M	125	70	11111101	175	3
31 1F 11111 37 (UMT SEPARATOR) 79 4F 1001111 117 0 127 7F 1111111 177 (DEL) 32 20 100000 40 (SMACE) 80 50 1010000 120 P 33 21 100001 41 1 81 51 1010001 120 P 34 22 100010 42 * 82 52 1010010 123 S 35 23 100011 45 84 54 1010100 124 T 37 25 100100 44 5 85 55 1010101 125 U 38 26 10010 46 6 86 56 1010101 26 V 40 28 101000 50 1 89 59 1011001 130 X 41 29 101001 51 1 89 59 1011001 130 X 42 2A 101000	30	16	11110	36	IRECORD SEPARATORI	78	4F	1001110	116	N	126	TE	1111110	176	2
32 20 100000 40 (SPACE) 80 50 1010000 120 P 33 21 100001 41 1 81 51 1010001 121 Q 34 22 100010 42 - 82 52 1010101 123 R 35 23 100101 43 # 83 53 1010101 123 S 36 24 100100 44 \$ 84 54 1010100 124 T 37 25 100101 45 \$ 86 56 1010101 125 U 38 26 100100 46 4 86 56 1010101 126 V 39 27 100111 47 87 57 1010101 127 W 40 28 101000 50 6 88 58 1011000 130 X 41 29 101001 51 9 89 59 1011001 131 Y 42 2A 101010 52 90 SA 1011010 132 Z 43 28 101101 53 + 91	31	1F	11111	37	IUWT SEPARATORI	79	45	1001111	117	0	127	7F	11111111	177	IDELL
33 21 100001 41 1 81 51 1010001 121 Q 34 22 100010 42 - 82 52 1010010 122 R 35 23 100011 43 # 83 53 1010011 123 S 36 24 100100 44 \$ 84 54 1010100 124 T 37 25 100101 45 % 85 55 1010101 125 U 38 26 100100 46 6 86 56 1010101 126 V 39 27 100111 47 * 87 57 1010111 127 W 40 28 101000 50 6 88 58 1011001 130 X 41 29 101001 51 J 89 59 1011001 131 Y 42 2A 101010 52 * 90 SA 1011010 132 Z 43 2B 101010 53 + 91 58 1011011 133 I 44 2C 101100 54 -	32	20	100000	40	(SPACE)	80	50	1010000	120	P	1.222	- 22			
34 22 100010 42 82 52 1010010 122 R 35 23 100011 43 83 53 1010010 123 S 36 24 100100 44 5 84 54 1010100 124 T 37 25 100101 45 85 55 1010101 125 U 38 26 100110 46 6 86 56 1010101 127 W 40 28 101000 50 6 88 58 1011000 130 X 41 29 1010151 89 59 1011001 131 Y 42 2A 101010 52 90 5A 1011010 132 Z 43 28 101010 53 + 91 58 1011101 133 I 44 2C 101100 54 . 92 5C 1011100 134 I 45 2D 101105 5 . 93 5D 1011101 135 I 46 2E 101110 56 . 94 5E 1011110 136 I <td>33</td> <td>21</td> <td>100001</td> <td>41</td> <td></td> <td>81</td> <td>51</td> <td>1010001</td> <td>121</td> <td>0</td> <td></td> <td></td> <td></td> <td></td> <td></td>	33	21	100001	41		81	51	1010001	121	0					
35 23 100011 43 # 83 53 1010011 123 S 36 24 100100 44 \$ 84 54 1010100 124 T 37 25 100101 45 % 85 55 1010101 125 U 38 26 100101 45 % 86 56 1010101 125 U 39 27 100111 47 * 87 57 1010101 126 V 40 28 101000 50 6 88 58 1011000 130 X 41 29 101001 51 1 89 59 1011001 131 Y 42 2A 101010 52 * 90 SA 1011001 131 Y 42 2B 101010 54 . 91 58 1011001 132 Z 43 2B 10100 54 . 92 5C 1011001 134 Y 45 2D 101105 . 93 50 1011101 135 1 46 2E 101110 56 .	34	22	100010	42	2	82	52	1010010	122	R					
36 24 100100 44 \$ 84 54 1010100 124 T 37 25 100101 45 % 85 55 1010101 125 U 38 26 100110 46 4 86 56 1010101 125 U 39 27 100111 47 87 57 1010101 127 W 40 28 101000 50 6 88 58 1011000 130 X 41 29 101001 51 9 89 59 1011001 131 Y 42 2A 101010 52 90 SA 1011010 132 Z 43 28 101010 53 + 91 58 1011101 133 1 44 2C 101100 54 . 92 5C 1011101 133 1 45 2D 10110 55 . 93 50 1011101 135 1 46 2E 101110 56 . 94 5E 10111101 136 1 47 2F 101111 57 / 95 5F <td>35</td> <td>23</td> <td>100011</td> <td>43</td> <td></td> <td>83</td> <td>53</td> <td>1010011</td> <td>123</td> <td>5</td> <td></td> <td></td> <td></td> <td></td> <td></td>	35	23	100011	43		83	53	1010011	123	5					
37 25 100101 45 % 85 55 1010101 125 U 38 26 100110 46 6 86 56 1010101 125 V 39 27 100111 47 87 S7 1010111 127 W 40 28 101000 50 6 88 58 1011001 130 X 41 29 101001 51 1 89 59 1011001 131 Y 42 2A 101010 52 90 SA 1011010 132 Z 43 2B 101010 53 + 91 58 1011001 133 I 44 2C 101100 54 . 92 5C 1011100 134 V 45 2D 101105 55 . 93 50 1011101 135 1 46 2E 101110 56 . 94 5E 1011110 136 . 47 2F 101111 57 / 95 5F 1011111 137	36	24	100100	44	6	84	54	1010100	124	T					
38 26 100110 46 6 86 56 1010110 126 V 39 27 100111 47 87 57 1010111 127 W 40 28 101000 50 6 88 58 101100 130 X 41 29 101001 51 1 89 59 101100 130 X 42 2A 101010 52 90 5A 101100 132 Z 43 28 101011 53 + 91 58 1011011 133 [44 2C 101100 54 . 92 5C 1011100 134 1 45 2D 101105 5 . 93 5D 1011101 135] 46 2E 101110 56 . 94 5E 1011110 136 ^ 47 2F 101111 57 / 95 5F 1011111 137	37	25	100101	45		85	55	1010101	125	U.					
39 27 100111 47 * 87 57 1010111 127 W 40 28 101000 50 6 88 58 1011000 130 X 41 29 101001 51 1 89 59 1011001 131 Y 42 2A 101010 52 * 90 5A 1011010 132 Z 43 2B 101100 54 . 91 58 1011010 133 F 44 2C 101100 54 . 92 5C 1011100 134 1 45 2D 101105 - 93 50 1011101 135 1 46 2E 101110 56 . 94 5E 1011110 136 4 47 2F 101111 57 / 95 5F 1011111 137 1	38	26	100110	46	6	86	56	1010110	1126	V					
40 28 101000 50 6 88 58 1011000 130 X 41 29 101001 51 1 89 59 1011001 131 Y 42 2A 101010 52 90 5A 1011010 132 Z 43 2B 101100 54 91 58 1011101 133 I 44 2C 101100 54 92 5C 1011100 134 Y 45 2D 10110 55 93 50 1011101 135 I 46 2E 101110 56 94 5E 1011110 136 A 47 2F 101111 57 / 95 5F 1011111 137	39	27	100111	47	-	87	57	1010111	127	ŵ					
41 29 101001 S1) 89 59 1011001 131 Y 42 2A 101010 52 90 SA 1011010 132 Z 43 2B 101011 53 + 91 58 1011011 133 [44 2C 101100 54 . 92 5C 1011100 134 \ 45 2D 101101 55 . 93 5D 1011101 135] 46 2E 101110 56 . 94 5E 1011110 136 ^ 47 2F 101111 57 / 95 5F 1011110 136 ^	40	28	101000	50	1	88	5.8	1011000	130	×					
42 2A 101010 52 90 5A 1011010 132 Z 43 2B 101011 53 + 91 5B 1011011 133 [44 2C 101100 54 - 92 5C 1011100 134 \ 45 2D 101101 55 - 93 50 1011101 135] 46 2E 101110 56 - 94 5E 1011110 136 ^ 47 2F 101111 57 / 95 5F 1011110 137 _	41	29	101001	51	i	89	59	1011001	131	Y					
43 28 101011 53 + 91 58 1011011 133 [44 2C 101100 54 . 92 5C 101101 133 [45 2D 101101 55 . 93 5D 1011101 135] 46 2E 101110 56 . 94 5E 1011110 136 _ 47 2F 101111 57 / 95 5F 1011111 137	42	24	101010	52	-	90	SA	1011010	132	7					
44 2C 101100 54 92 5C 1011100 134 1 45 2D 101101 55 93 5O 1011101 135 1 46 2E 101110 56 94 5E 1011110 136 1 47 2F 101111 57 7 95 5F 1011111 137	43	28	101011	53		91	58	1011011	133	E.					
45 2D 101101 55 93 50 1011101 135 1 46 2E 101110 56 94 5E 1011101 136 1 47 2F 101111 57 7 95 5F 1011101 137	44	20	101100	54		92	SC	1011100	134	1					
46 2E 101110 56 . 94 5E 101110 136 ^ 47 2F 101111 57 / 95 5F 1011110 137	45	20	101101	55	÷	93	50	1011101	135	1					
47 2F 101111 57 / 95 5F 1011111 137	46	2E	101110	56		94	SE	1011110	136	2					
	47	2F	101111	57	1	95	SE	1011111	137						

Computers store text documents, both on disk and in memory, using ASCII codes. For example, if you use Notepad in Windows OS to create a text file containing the words, "Four score and seven years ago," Notepad would use 1 byte of memory per character (including 1 byte for each space character between the words

It is worth emphasising here that these codes for numbers are exclusively for use in the context of stored, displayed or printed text. All of the other coding schemes for numbers are for internal use in a computer system and would not be used in a text.

There are some special features that make the coding scheme easy to use in certain circumstances. The first is that the codes for numbers and for letters are in sequence in each case so that, for example, if 1 is added to the code for seven the code for eight is produced.



The second is that the codes for the upper-case letters differ from the codes for the corresponding lower-case letters only in the value of bit 5. This makes conversion of upper case to lower case, or the reverse, a simple operation.

Unicode

Despite still being widely used, the ASCII codes are far from adequate for many purposes.

Unicode is an international encoding standard for use with different languages and scripts.

It works by providing a unique number for every character, this creates a consistent encoding, representation, and handling of text.

Basically Unicode is like a Universal Alphabet that covers the majority of different languages across the world, it transforms characters into numbers.

It achieves this by using character encoding, which is to assign a number to every character that can be used.

What's an example of a Unicode?

Unicode has its own special terminology. For example, a character code is referred to as a 'code point'. In any documentation there is a special way of identifying a code point. An example is U+0041 which is the code point corresponding to the alphabetic character A.

The 0041 are hexadecimal characters representing two bytes. The interesting point is that in a text where the coding has been identified as Unicode it is only necessary to use a one-byte representation for the 128 codes corresponding to ASCII. To ensure such a code cannot be misinterpreted, the codes where more than one byte is needed have restrictions applied.



(Paper 1. Sec 1.1.1) Number representation

Computer Science 9608 with Majid Tahir

0C4A	0C48	0C4C	0C4D								0C55	0C56		
دن ۵۵۹۸	80 3	ൌ ⁰⁰⁴⁰	് ‱										ൗ 0057	
്	†	ୢୖ	ိ	ိ	0	0	໑	ഉ	ഒ	۵	æ	Ъ	ത	IJ
0E4A	0E48	0E4C	0E4D	0E4E	0E4F	0E50	0E51	0E52	0E53	0E54	0E55	0E56	0E57	0E58
	DF4B	P OF4C	F4D	Ъ OF4E	5 DF4F	X 0F50	5 0F51	15 0F52	J 0F53	Z 1 0F54	년 0F55	지 0F56	0557	5 1 0F58
 104A	 1048	COJ]	၅ 1040	9 6 :	ଜ୍ଧ _{104F}	⊙ 1050	() 1051	(5 1052	25 1053	e 1054	e. 1055	ු 1056	് പാ 1057	୍ଚ 1058
0E	<u>0</u> 页	ò	天0	$\overline{\mathcal{N}}$	75	$\overline{\mathbf{n}}$	M	汐	ゐ	\star	大	亚日	풍	ঠঠ
114A	114B	114C	1140	114E	114F	1150	1151	1152	1153	1154	1155	1156	1157	1158
ቀኣ	ቋ	ß	ቀኁ			Ŧ	Æ	Æ	Þ	Æ	Ŧ	F		ቘ
124A	1248	124C	124D			1250	1251	1252	1253	1254	1255	1256		1258
6.	4.	60	ፍ	6.	<u>ኑ</u>	Т	F	T,	Т	Т	Т	T	T	2
134A	134B	134C	134D	134E	134F	1350	1351	1352	1353	1354	1355	1356	1357	1358

At its core, Unicode is like ASCII: a list of characters that people want to type into a computer. Every character gets a numeric codepoint, whether it's capital A, lowercase lambda, or "<u>man in business suit levitating</u>."

A = 65

λ = 923

So Unicode says things like, "Allright, this character exists, we assigned it an official name and a codepoint, here are its lowercase or uppercase equivalents (if any), and here's a picture of what it could look like. Font designers, it's up to you to draw this in your font if you want to."



Just like ASCII, Unicode strings (imagine "codepoint 121, codepoint 111...") have to be encoded to ones and zeros before you can store or transmit them. But unlike ASCII, Unicode has more than a million possible codepoints, so they can't possibly all fit in one byte. And unlike ASCII, there's no One True Way to encode it.

What can we do? One idea would be to **always** use, say, 3 bytes per character. That would be nice for string traversal, because the 3rd codepoint in a string would always start at the 9th byte. But it would be inefficient when it comes to storage space and bandwidth.

Instead, the most common solution is an encoding called UTF-8.

UTF-8:

UTF-8 gives you four templates to choose from: a one-byte template, a two-byte template, a three-byte template, and a four-byte template.

```
0xxxxxx
110xxxxx 10xxxxxx
1110xxxx 10xxxxxx 10xxxxxx
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

Each of those templates has some headers which are always the same (**shown here in red**) and some slots where your codepoint data can go (shown here as "x"s).

The four-byte template gives us 21 bits for our data, which would let us represent 2,097,151 different values. There are only about 128,000 codepoints right now, so UTF-8 can easily encode any Unicode codepoint for the foreseeable future.

- Unicode to represent any possible text in code form.
- Unicode is a computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.
- Developed in conjunction with the Universal Coded Character Set (UCS) standard and published as *The Unicode Standard*, the latest version of Unicode contains a repertoire of more than 128,000 characters covering 135 modern and historic scripts, as well as multiple symbol sets..



- As of June 2016, the most recent version is Unicode 9.0. The standard is maintained by the Unicode Consortium.
- Unicode's success at unifying character sets has led to its widespread and predominant use in the internationalization and localization of computer software. The standard has been implemented in many recent technologies, including modern operating systems, XML, Java (and other programming languages), and the .NET Framework

Practice Questions

Convert to one's complement:

- 1. 1010
- 2. 11110000
- 3. 10111100 11000000
- 4. 10100001

Convert to two's complement:

- 1. 1010
- 2. 11110000
- 3. 10000000
- 4. 011111111

Convert these negative decimal values to negative binary using two's

complement:

- 1. -192d
- 2. -16d
- 3. -1d
- 4. -0d





Answers

One's complement:

- 1. 0101
- 2. 00001111
- 3. 01000011 00111111
- 4. 01011110

Two's complement:

- 1. $0110(1010 \rightarrow 0101 + 1 = 0110)$
- 2. 00010000
- 3. 10000000 (Result is the same as the original value.)
- 4. 10000001

Negative decimal to negative binary:

- 1. 0100000b (192d = 1100000b -> 00111111 + 1 = 0100000b)
- 2. 11110000b (16d = 00010000b -> 11101111 + 1 = 11110000b)
- 3. 11111111b (1d = 0000001b -> 1111110 + 1 = 1111111b) Tricky? Before converting from binary to decimal, we must know ahead of time if the binary value is signed or not because a signed binary value will not convert properly using the place value chart we have seen so far. If seen by itself, 1111111b = 255d, not -1d. As a rule, assume that a binary value, such as 1111111b, is a positive integer unless context specifies otherwise. Since we are dealing with negative binary values in this problem set, then 1111111b is -1d, not 255d.
- 4. 0. There is no such thing as negative zero (-0). Nothing is always nothing and does not have a sign. We can convert anyway: 0d = 00000000 -> 11111111 + 1 = 1 00000000b. We still arrive at 0 in binary for the eight relevant bits. Ignore the ninth carry bit.

References:

http://www.bfoit.org/itp/ComputerContinuum/RobotComputer.html https://en.wikibooks.org/wiki/GCSE_Computer_Science/Binary_representation http://bssbmi.com/olevel/computer-science-2210/class-9/binary-systems/ http://www.math10.com/en/algebra/systems-of-counting/binary-system.html http://www.answers.com/Q/What are the applications of BCD Code https://en.wikipedia.org/wiki/Binary-coded_decimal https://commons.wikimedia.org/wiki/File:CPT-Numbers-Conversion.svg

