SECTION 6 RELATED TOPICS

Logic Gates

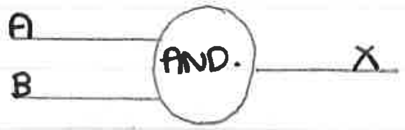
These are electronic circuits designed to perform a single operation. They use the binary system where:-

0 represents **off**
1 represents **on**

Logic gates produce a certain output depending on the input given to them. This input is in the form of an electrical impulse referred to as binary digits or Bits. There are 3 basic elements of logic gates which make up the Adding Circuit in the computer.

AND gate

2 or more pieces of input, 1 output so A and B give X

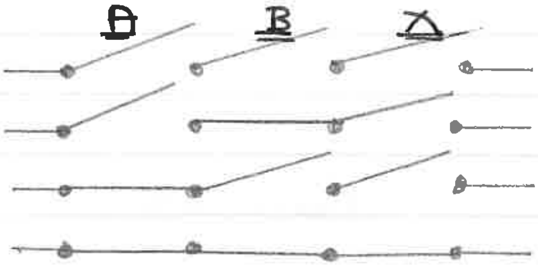


X= A AND B, $X = A \cdot B$, X= AB.

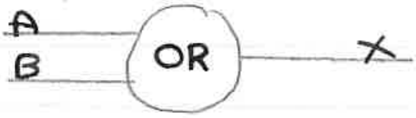
Truth Table

INPUT		OUTPUT
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Circuit



OR gate



X= A OR B, $X = A + B$

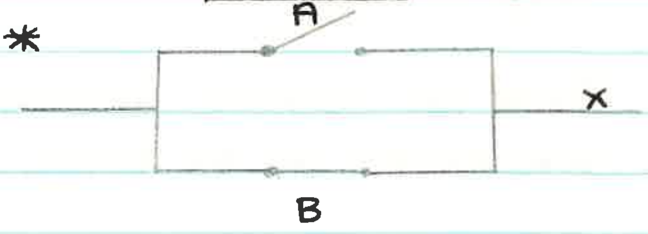
Andrew Vumuls

Truth Table

	Input		Output
	A	B	X
	0	0	0
*	0	1	1
	1	0	1
	1	1	1

Circuit

*



NOT gate

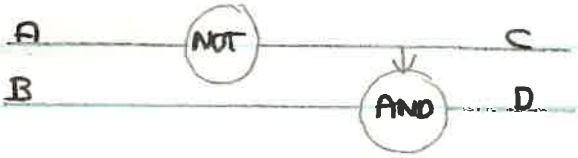
X = NOT A , X = $\overline{A}$

Truth Table

Input	Output
A	X
0	1
1	0

Simple Circuit

eg



Truth Table

Input		Output	
A	B	C	D
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

Homework

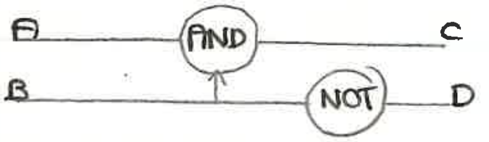
①

Input		Output	
B	B	C	D
0	0	1	1 ✓
0	1	1	1 ✓
1	0	0	0 ✓
1	1	0	1 ✓

✓

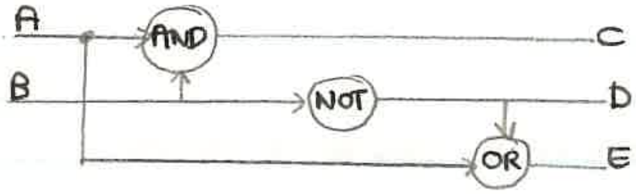
Andrew Uurnuls

②



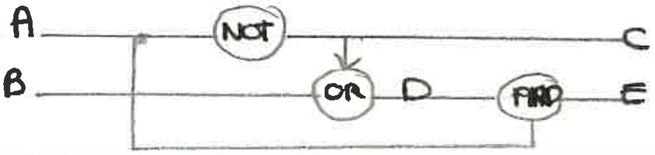
Input		Output		
A	B	C	D	
0	0	0	1	✓
0	1	0	0	✓
1	0	0	1	✓
1	1	1	0	✓

③



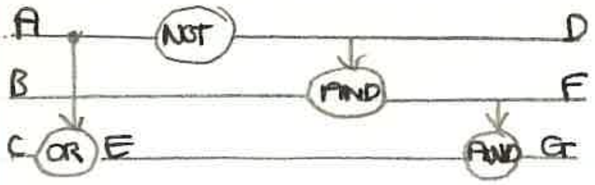
A	B	C	D	E	
0	0	0	1	1	✓
0	1	0	0	0	✓
1	0	0	1	1	✓
1	1	1	0	1	✓

④



A	B	C	D	E	
0	0	1	1	0	✓
0	1	1	1	0	✓
1	0	0	0	0	✓
1	1	0	1	1	✓

⑤



A	B	C	D	E	F	G	
0	0	0	1	0	0	0	✓
0	0	1	1	1	0	0	✓
0	1	0	1	0	1	0	✓
1	0	0	0	1	0	0	✓
0	1	1	1	1	1	1	✓
1	1	0	0	1	0	0	✓
1	0	1	0	1	0	0	✓
1	1	1	0	1	0	0	✓

$\frac{14}{14}$

Excellent

merit mark

The NAND gate p137 letts

This is a combination of an and and a not gate its symbol

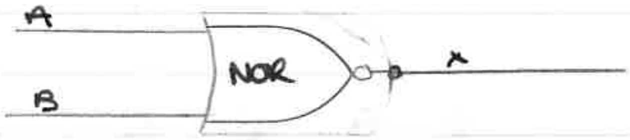
is

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

and has a truth table as on the left .

The NOR gate.

A combination of OR and NOT.



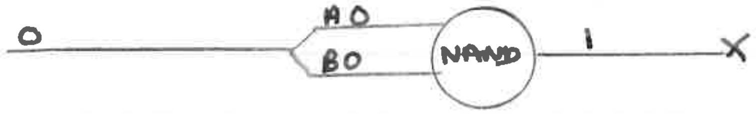
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

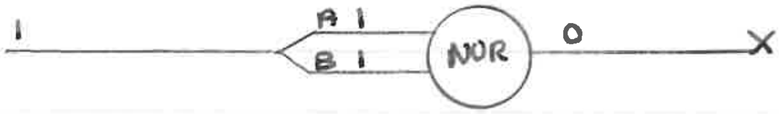
These two gates are important because

1. They use fewer electronic components than separate gates, and are therefore cheaper

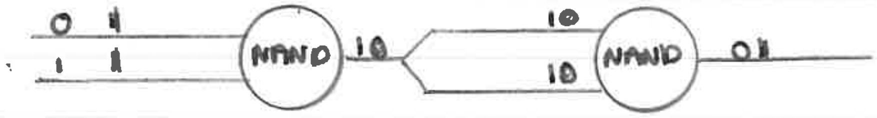
2 Both Nand and Nor gates can be used to represent not gates and so only these two need be used in any circuit.

Use of NAND and NOR as NOT gates





An AND gate can be substituted ~~to~~ by two NAND gates.

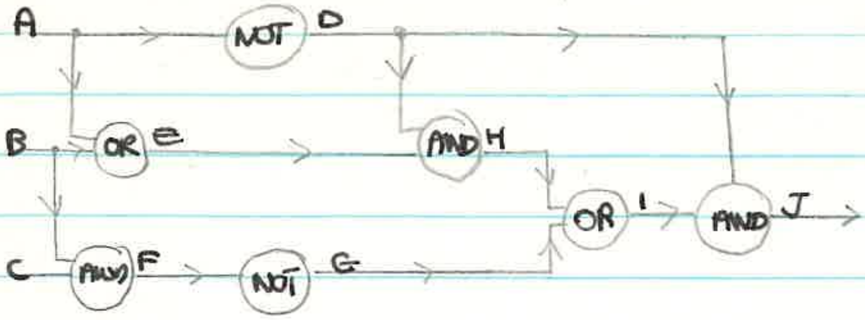


An OR gate can be substituted by two NOR gates

1) <u>June 1983 Paper I Section B Question 12:</u> (January 1983)

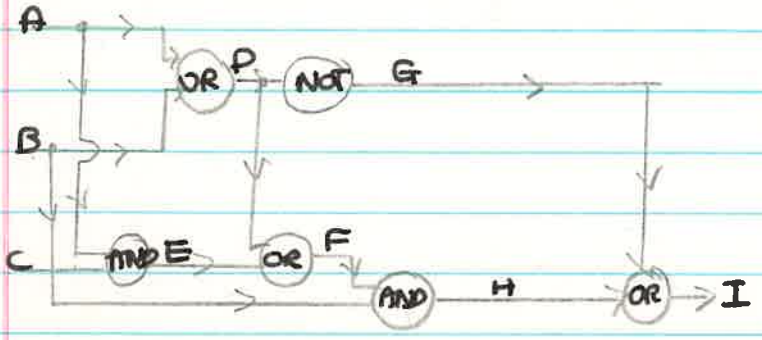
2) <u>June 1980 Paper I Section A Question 10.</u>

Andrew Vinuls Homework

1) 

INPUT			OUTPUT						
A	B	C	D	E	F	G	H	I	J
0	0	0	1	0	0	1	0	1	1
0	0	1	1	0	0	1	0	1	1
0	1	0	1	1	0	1	1	1	1
1	0	0	0	1	0	1	0	1	0
0	1	1	1	1	1	0	1	1	1
1	1	0	0	1	0	1	0	1	0
1	0	1	0	1	0	1	0	1	0
1	1	1	0	1	1	0	0	0	0

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

2) 

INPUT			OUTPUT					
A	B	C	P	E	F	G	H	I
0	0	0	0	0	0	1	0	1
0	0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	1	1
1	0	0	1	0	1	0	0	0
0	1	1	1	0	1	0	1	1
1	1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0	0
1	1	1	1	1	1	0	1	1

✓ ✓ ✓ ✓ ✓ ✓

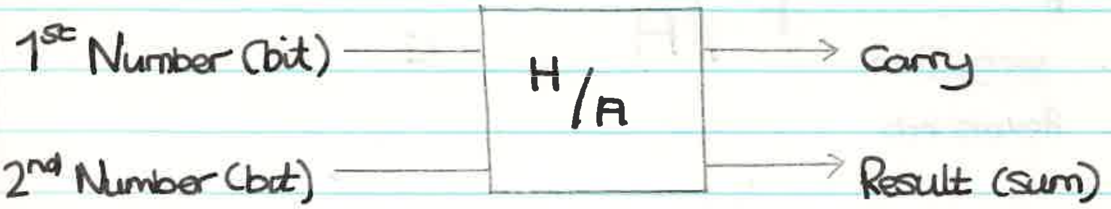
13/13

good

7.11.83

Half-Adder

This actually shows the logic gates involved in an addition circuit within the computer concerned with arithmetic operations



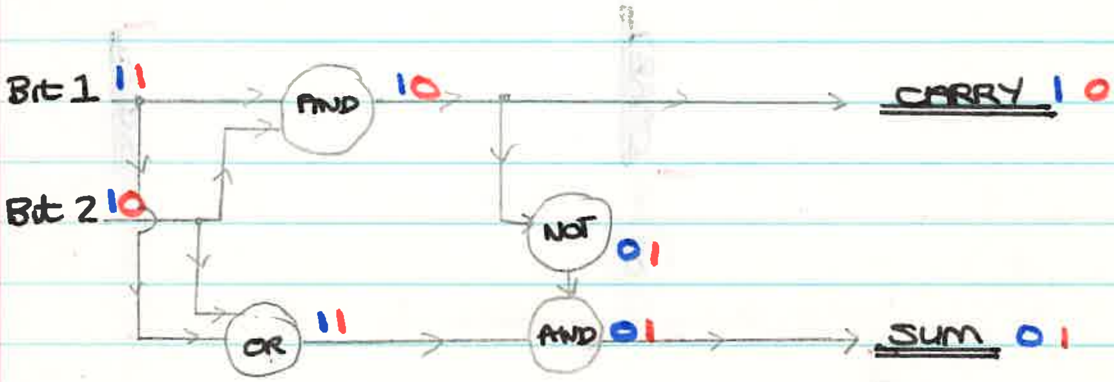
e.g.

1_2

1_2 +

0 - RESULT

CARRY - 1



e.g.2

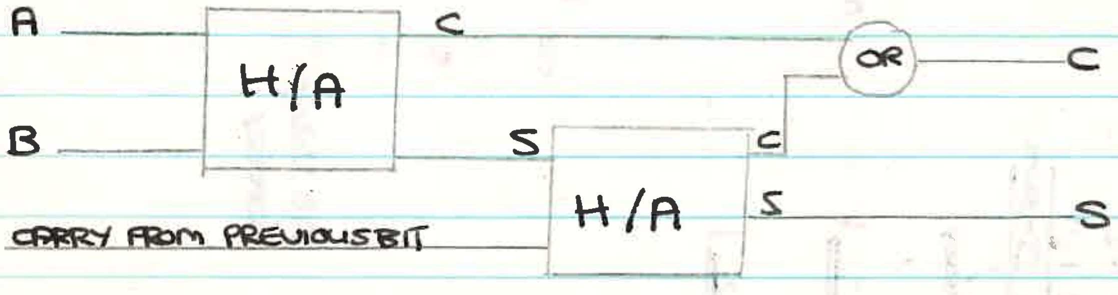
1_2

0_2 +

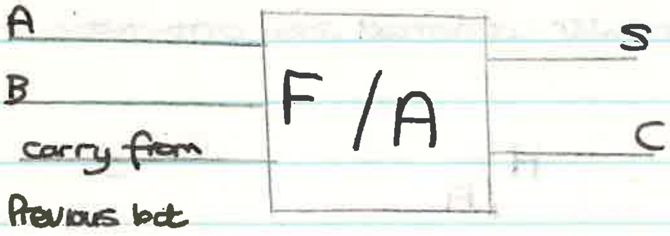
1 - RESULT

0-CARRY

Full-Adder



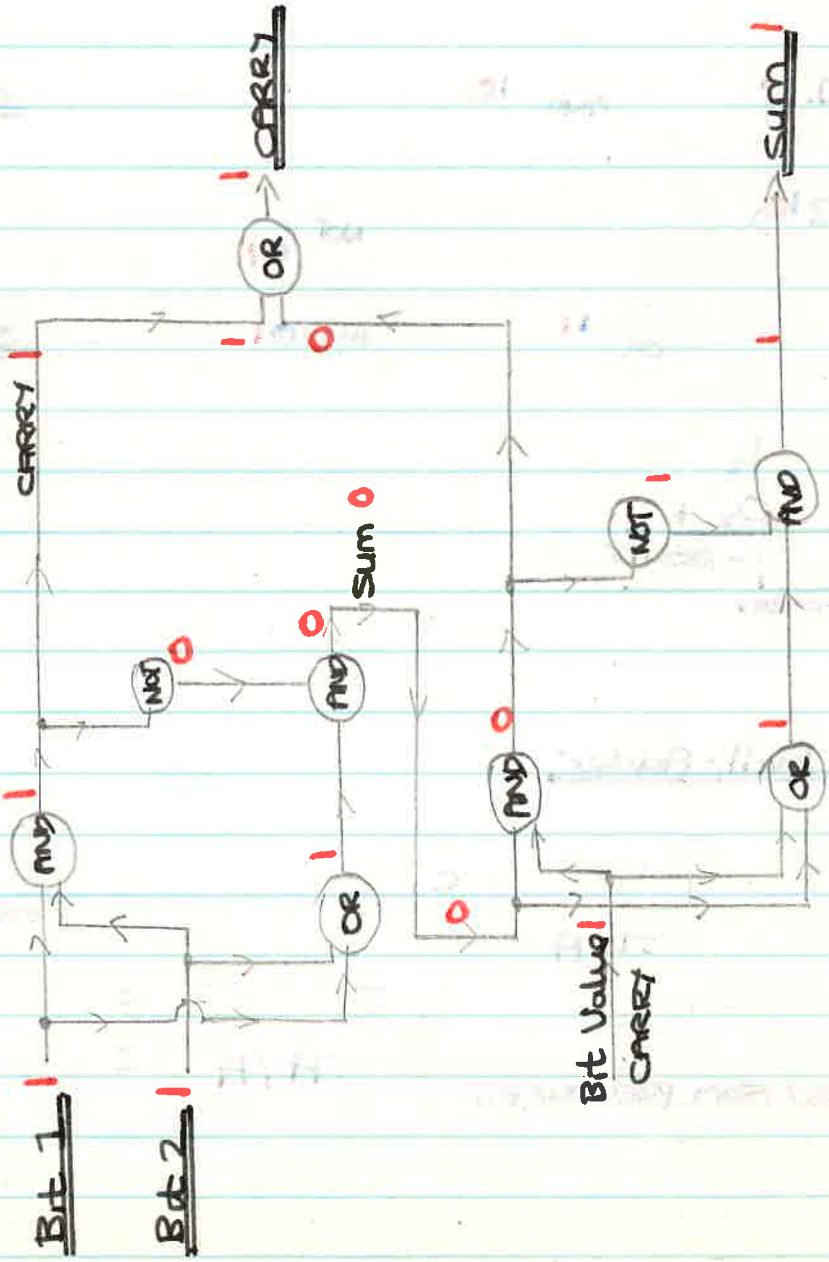
Full-Adder



Full-Adder

Full Adder

e.g $1\ 1_2$ +
$1\ 1_2$
Sum-1 | 1 0 - Sum
CARRY

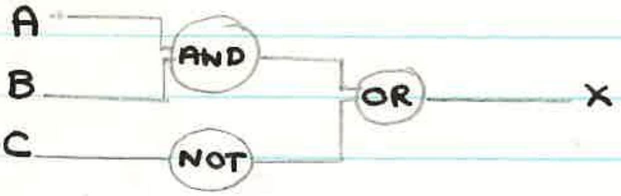


Andrew Vurnuts

Design of Circuits

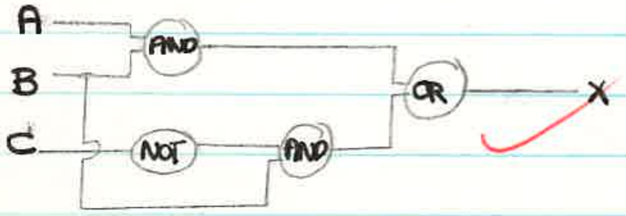
e.g

$$X = A \cdot B + \bar{C}$$



Exercise

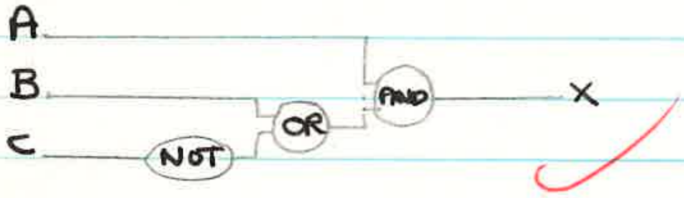
① $X = A \cdot B + \bar{C} \cdot B$



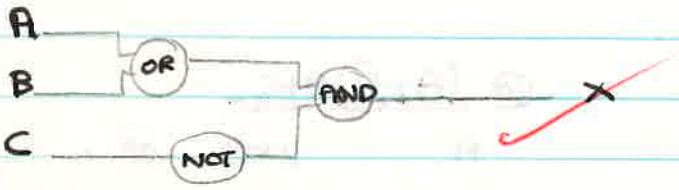
② $X = A_{\uparrow}(B + \bar{C})$

implied and

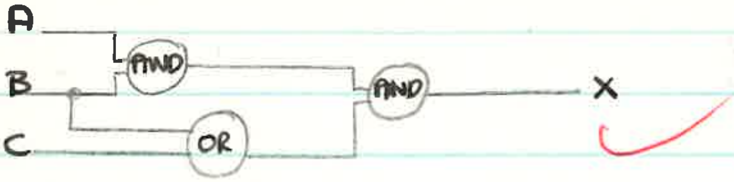
gate



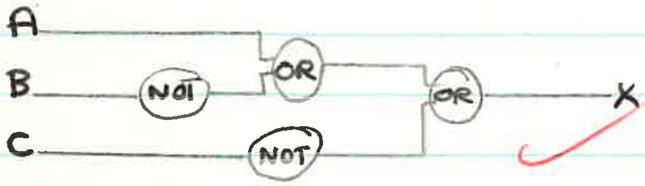
③ $X = (A+B)\bar{C}$



④ $(A \cdot B)(B+C)$

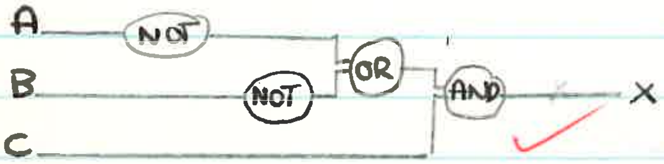


⑤ $(A+\bar{B})+\bar{C}$

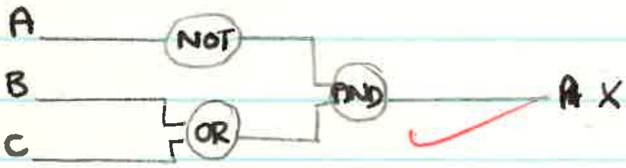


$\frac{5}{5}$ good

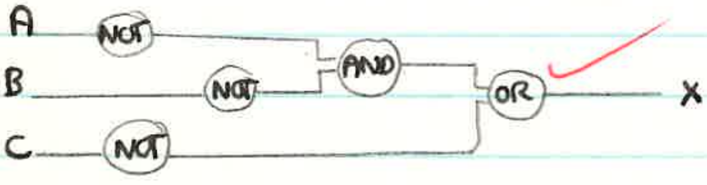
⑥ $\bar{A}+\bar{B} \cdot C$



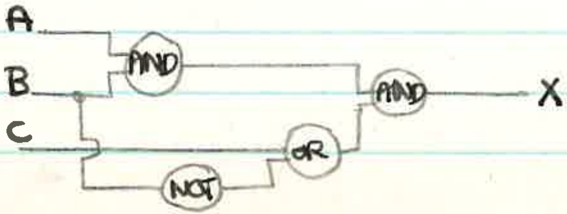
⑦ $(B+C) \cdot \bar{A}$



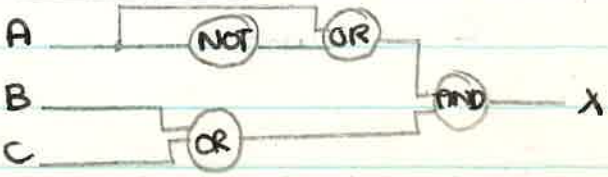
⑧ $(\bar{A} \cdot \bar{B})+\bar{C}$



⑨ $(A \cdot B)(\bar{B}+C)$



⑩ $(A+\bar{A})B+C$



Computer Arithmetic

Two State Code

We use the base 10 or denary number system for our arithmetic, probably because we have got 10 fingers, however binary or base 2 is much better for computer applications.

In base 2 the only symbols used are 0 or 1 (these are binary digits or bits). These symbols can easily be used in circuit design where they represent switches inside the computer either being ON or OFF. This is known as a two state code

Representation of Numbers

a) Base 10

HUNDREDS (100)	TENS (10)	UNITS (1)	— COLUMN HEADINGS
1	3	7_{10}	

$(1 \times 100) + (3 \times 10) + (7 \times 1) = \underline{137}_{10}$

b) Base 2

Column headings are generally given by :-

$x^4 \quad x^3 \quad x^2 \quad x^1 \quad x^0$

where x is the number of the base

$x=2$	32	16	8	4	2	1		
e.g			1	0	1	1_2	=	11_{10}

Change to base 10

	32	16	8	4	2	1		
1		1	1	0	1	1_2	=	$\underline{27}_{10}$ ✓

	32	16	8	4	2	1	
2,		1	1	0	1	0_2 =	26_{10} ✓
3,	1	1	1	0	0	1_2 =	57_{10} ✓
4,	1	0	1	1	0	1_2 =	45_{10} ✓
5,		1	0	0	0	1_2 =	17_{10} ✓
6,	1	0	1	0	1	1_2 =	43_{10} ✓
7,	1	1	0	0	1	1_2 =	51_{10} ✓
8,		1	0	0	1	0_2 =	18_{10} ✓
a,		1	1	1	1	1_2 =	31_{10} ✓
10,		1	0	0	0	0_2 =	16_{10} ✓ $\frac{10}{10}$ - me

c) Base 8 - Octal

	x^3	x^2	x^1	x^0
	512	64	8	1
eg		1	2	4_8

Digits 0 ⟶ 7

$(1\times64)+(2\times8)+(4\times1) = 84_{10}$

Change to Base 10

	512	64	8	1			64	8	1	
1)		1	4	2_2 =	98_{10} ✓	4)	4	7	3_8 =	59_{10} ✓
2)			4	5_8 =	$3\cancel{6}7_{10}$ ✗	5)	1	0	6_8 =	70_{10} ✓
3)		2	1	6_8 =	142_{10} ✓					

$\frac{4}{5}$ - me

d) Base 16 - Hexadecimal

x^2	x^1	x^0
256	16	1
1	2	A_{16}

Digits 0 → 9 A → F

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

10, 11, 12, 13, 14, 15

$(1 \times 256) + (2 \times 16) + (10 \times 1) = 298_{10}$

Change to Base 10

	256	16	1			256	16	1	
1)		2	A_{16}	$= 42_{10}$ ✓	4)	2	4	D_{16}	$= 589_{10}$ ✓
2)	1	1	B_{16}	$= 283_{10}$ ✓	5)	2	1	G_{16}	$= 534_{10}$ ✓
3)	1	3	F_{16}	$= 319_{10}$ ✓					

5 [illegible]

5

Conversion between bases

1) Binary to Octal

Take the binary number and divide into groups of three from the right handside

e.g 101100_2

101_2 100_2

5_{10} 4_{10} $= 54_8$

e.g i) 111001_2

7_{10} 1_{10} $= 71_8$

ii) $100\ 101_2$

100_2 101_2

4_{10} 5_{10} $= 45_8$

iii) 1010111_2

$1 \quad 010_2 \quad 111_2$

$1_{10} \quad 2_{10} \quad 7_{10} \quad = \underline{\underline{127}}_8$

2) Binary to hexadecimal

Take binary number and divide into groups of 4 from the right hand side

e.g 1010110110_2

$10_2 \quad 1011_2 \quad 0110_2$

$2_{10} \quad 11_{10} = B_{16} \quad 6_{10} \quad = \underline{\underline{2B6}}_{16}$

e.gs i) 10001101_2

$1000_2 \quad 1101_2$

$8_{10} \quad 13_{10} = D_{16} \quad = \underline{\underline{8D}}_{16}$

ii) 1010100111_2

$10_2 \quad 1010_2 \quad 0111_2$

$2 \quad 10_{10} = A_{16} \quad 7_{10} \quad = \underline{\underline{2A7}}_{16}$

iii) 1110001010_2

$11_2 \quad 1000_2 \quad 1010_2$

$3_{10} \quad 8_{10} \quad 10_{10} = A_{16} \quad = \underline{\underline{38A}}_{16}$

3) Octal to Binary

64	8	1
3	0	4_8

$= \underline{196}_{10}$

$11_2 \quad 000_2 \quad 100_2$

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

$= \underline{196}_{10}$

4) Hex to binary

e.g $1 \quad A \quad C_{16}$

$00011010\,1100_2$

$\underline{\underline{256}} \quad \underline{\underline{16}} \quad \underline{\underline{1}}$

$1 \quad A \quad C_{16} = \underline{\underline{428}}_{10}$

256 128 64 32 16 8 4 2 1

$110101100 = \underline{\underline{428}}_{10}$

5) Octal to hex (+ v.v)

To change from base 8 to base 16 (and v.v) you must go through an intermediate stage in binary (base 2)

eg i) $2 \quad 0 \quad B_{16}$

$= 10_2\, 0000_2\, 1011_2$

1000001011_2

$1_2\, 000_2\, 001_2\, 011_2$

$1 \quad 0 \quad 1 \quad 3 = \underline{1013}_8$

ii) $1 \quad 2 \quad 3_8$

$001_2\, 010_2\, 011_2$

$= 1010011_2$

$101_2\, 0011_2$

$5_{10} \quad 3_{10} = \underline{\underline{53}}_8$

6) Denary to Binary

$53_{10} = 110101_2$

$2 \,|\, 53$
$2 \,|\, 26 \quad 1$
$2 \,|\, 13 \quad 0$
$2 \,|\, 6 \quad 1$
$2 \,|\, 3 \quad 0$
$2 \,|\, 1 \quad 1$
$2 \,|\, 0 \quad 1$

$= 110101_2$

Binary Arithmetic

16 8 4 2 1

1) $1011_2 \qquad 11_{10}$
$\ {}_1 1 {}_1 0 {}_1 1_2 + \qquad + 5_{10}$
$\ 10000_2 \qquad 16_{10}$

2) 16 8 4 2 1
$1111_2 \qquad 15_{10}$
${}_1 1 {}_1 1 {}_1 1_2 + \qquad 7_{10}$
$10110_2 \qquad 22_{10}$

3) $\overset{0}{\cancel{1}}\overset{2}{\cancel{0}}11_2 \qquad 11_{10}$
$\ 101_2 - \qquad 5_{10}$
$\ 110_2 \qquad 6_{10}$

4) $\overset{0}{\cancel{1}}\overset{2}{0}\overset{0}{\cancel{1}}\overset{2}{0}_2 \qquad 10_{10}$
$\ 101_2 \qquad 5_{10}$
$\ 101_2 \qquad 5_{10}$

Homework

1)

16	8	4	2	1
1	1	0	1	0_2

$= 26_{10}$

2)

16	8	4	2	1
1	0	0	0	1_2

$= 17_{10}$

3) $\underline{16}\ \ \underline{8}\ \ \underline{4}\ \underline{2}\ \ \underline{1}$

$1\ \ 0\ \ 10\ \ 1_2 = \underline{\underline{21}}_{10}$

4) $1\ \ 0\ \ 1\ 1\ \ 1_2 = \underline{\underline{23}}_{10}$

5) $1\ \ 0\ \ 0\ \ \ 100 = \underline{\underline{36}}_{10}$

6) $\underline{\underline{64}}\ \ \underline{\underline{8}}\ \ \underline{\underline{1}}$

$1\ \ \ 3\ \ \ 2_8 = \underline{\underline{90}}_{10}$

7) $1\ \ \ 2\ \ \ 1_8 = \underline{\underline{81}}_{10}$

8) $2\ \ \ 1\ \ \ 3_6 = \underline{\underline{139}}_{10}$

9) $\underline{\underline{256}}\ \ \ \underline{\underline{16}}\ \ \ \underline{1}$

$2\ \ \ D_{16} = \underline{\underline{45}}_{10}$

10) $3\ \ C_{16} = \underline{\underline{60}}_{10}$

11) $24_{10} = 11000_2$

12) $36_{10} = 100100_2$

13) $45_{10} = 101101_2$

14) $57_{10} = 111001_2$

15) $63_{10} = 111111_2$

16) $101_2 +$ $\qquad$ $5_{10}\ +$

$\underline{\ \ 10_2}$ $\qquad$ $\underline{2_{10}}$

$\underline{111_2}$ $\qquad$ $\underline{7_{10}}$

17) $$\begin{array}{r} 1101_2 + \\ 11_2 \\ \hline 10000_2 \end{array} \quad = \quad \begin{array}{r} 13_{10} + \\ 3_{10} \\ \hline 16_{10} \end{array}$$

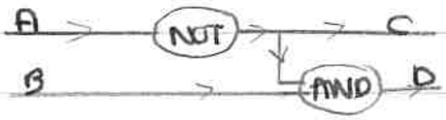
18) $$\begin{array}{r} 1001_2 \\ 101_2 + \\ \hline 1110 \end{array} \qquad \begin{array}{r} 9_{10} + \\ 5_{10} \\ \hline 14_{10} \end{array}$$

19) $$\begin{array}{r} 101_2 \\ 1_2 - \end{array} \qquad \begin{array}{r} \cdot 101_2 + \\ 111_2 \\ \hline \cancel{1}100_2 \end{array} \quad = \underline{\underline{100_2}}$$

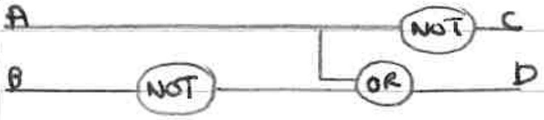
$$\begin{array}{r} 5_{10} \\ -1_{10} \\ \hline = 4_{10} \end{array} \qquad \underline{\underline{100_2}}$$

20) $$\begin{array}{r} 1011_2 \\ 10_2 - \end{array} \quad = \quad \begin{array}{r} 1011_2 + \\ 1101_2 \end{array} \qquad \begin{array}{r} 1011_2 + \\ 1110_2 \\ \hline 11001_2 \end{array}$$

$$\begin{array}{r} 11_{10} - \\ 2_{10} \\ \hline 9_{10} \end{array} \qquad \underline{\underline{11001_2}}$$

21) 

A	B	C	D
0	0	1	0
0	1	1	1
1	0	0	0
1	1	0	0

22) 

A	B	C	D
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	1

Convert the following numbers to the base 10 (denary)

11010_2 2. 10001_2 3. 10101_2 4. 10111_2 5. 100100_2

132_8 7. 121_8 8. 213_8 9. $2D_{16}$ 10. $3C_{16}$

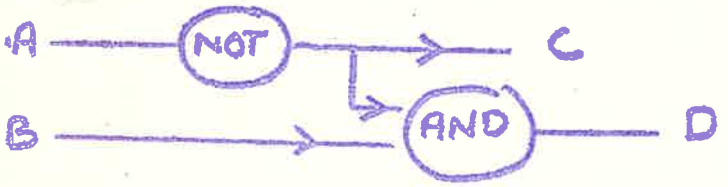
Convert the following to binary, base 2.

24_{10} 12. 36_{10} 13. 45_{10} 14. 57_{10} 15. 63_{10}

Following are binary addition and subtraction

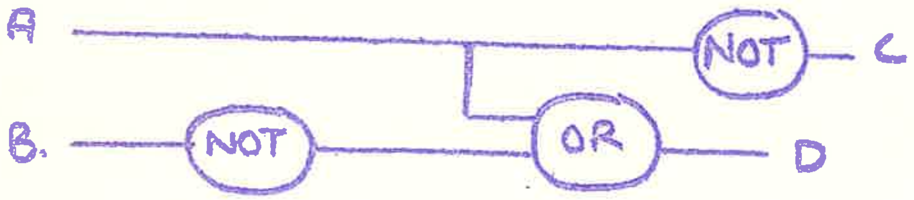
	17.	18.	19.	20.
101	1101	1001	101	1011
10 +	11 +	101 +	1 -	10 -

Complete the truth tables for the following logic circuits



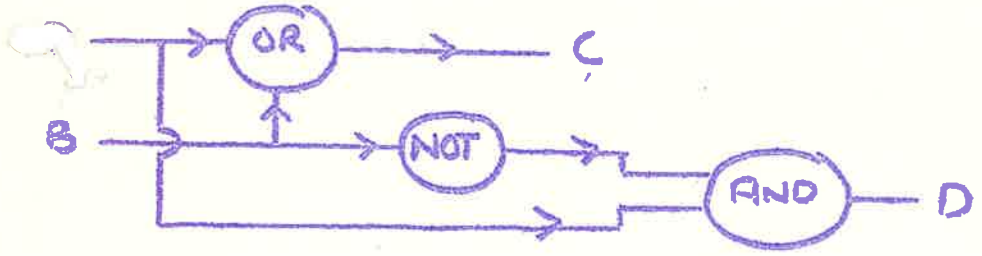
A	B	C	D
0	0		
0	1		
1	0		
1	1		

2.



A	B	C	D
0	0		
0	1		
1	0		
1	1		

3.



A	B	C	D
0	0		
0	1		
1	0		
1	1		

Design circuits for the following output

4. (A OR B) AND (A AND C)

5. (A OR NOT B) AND C

The Use of Complementary Arithmetic

Ones complement

eg 1　A = 10110
　　　B = 01001　　　　10110 1C

A − B = A + Comp (1C) B

$$A + \text{Comp (1C) } B = \begin{array}{r} 10110 \\ 10110 \\ \hline 101100 \\ \hline \end{array}$$

To get the result we now ~~ignore~~ add the msB (most significant bit) to the LSB (least significant bit)

$$\text{msB}\quad \cancel{1}01100 \quad \text{LSB}$$

This is called end around carry

$$\begin{array}{lr} & 01100 \\ & 1 \\ \hline \text{RESULT} & 01101 \\ \hline \end{array}$$

CHECK

$$\begin{array}{rr} & 1\overset{2}{0}11\overset{2}{0} \\ - & \cancel{0}10\cancel{0}1 \\ \hline & 01101 \\ \hline \end{array} \quad \begin{array}{rr} = & 22_{10} \\ = & -9_{10} \\ \hline & 13_{10} \end{array}$$

eg 2　A = 101
　　　B = 011

eg 3　A = 110
　　　B = 101

Twos Complement

eg 1　A = 1101
　　　B = 0111　　　1000 (1C)　　1001 (2C)

A − B = A + Comp (2C) B

$$\begin{array}{rr} A = & 1101 \\ \text{Comp (2C) } B = & 1001\ + \\ \hline & \cancel{1}0110 \end{array}$$

To get a result ignore the msB

RESULT = 110

CHECK

$$\begin{array}{r} 8\ 4\ 2\ 0 \\ 1\cancel{1}01 \\ \cancel{0}\cancel{1}11 \\ \hline 0110 \end{array} \qquad \begin{array}{r} 13_{10} \\ -7_{10} \\ \hline 6_{10} \end{array}$$

eg 2　A = 111
　　　B = 101

eg 3　A = 1010
　　　B = 1000

The Use of Complementary Arithmatic

Ones complement

e.g 1 A = 10110

B = 01001 10110 1C

A − B = A + Comp (1C) B

$$A + \text{Comp (1C) } B = \begin{array}{r} 10110 \\ 10110 \\ \hline 101100 \\ \hline \end{array}$$

To get the result we now add the M.S.B (most signficant bit) to the L.S.B (least significant bit)

This is called

$$\begin{array}{c} 101100 \\ \nearrow \qquad\qquad \nwarrow \\ \text{M.S.B} \qquad \text{L.S.B} \end{array}$$

end around carry

$$\begin{array}{r} 01100 \\ 1 \\ \hline 01101 \\ \hline \end{array}$$

Check

$$\begin{array}{r} 1\overset{2}{0}11\overset{2}{0} \\ -{}_{1}\cancel{0}10\overset{1}{\cancel{0}}1 \\ \hline 01101 \\ \hline \end{array} \quad = \quad \begin{array}{r} 22_{10} \\ -9_{10} \\ \hline 13_{10} \\ \hline \end{array}$$

Twos Complement.

e.g 1 A = 1101

B = 0111 1000 (1C) 1001 (2C)

A − B = A + Comp (2C) B

$$\begin{array}{r} A = 1101 \\ \text{Comp(2C) } B = \underline{1001} \\ \underline{\cancel{1}0110} \end{array}$$

To get a result ignore the M.S.B

Result = $\underline{\underline{110}}$

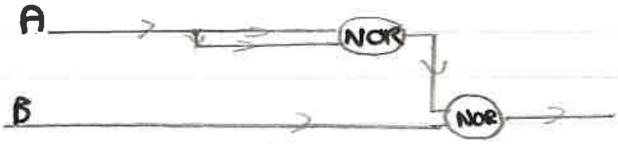
Check

$$\begin{array}{r} 1\,{}^{2}\cancel{1}\,{}^{2}\cancel{0}\,1 \\ -\ \underline{\cancel{0}_{2}\cancel{1}\,1\,1} \\ \underline{0110} \end{array} = \begin{array}{r} 13_{10} \\ \underline{-\ 7_{10}} \\ \underline{6_{10}} \end{array}$$

Page 34 Exercise 7 15th December

17. The best description of a NOT gate is;-

a) If the input bit is 1, the result is 0; if the input bit is 0, the result is 1.

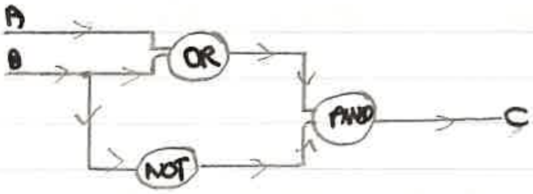
18.


A	B	C
0	1	0
1	0	1

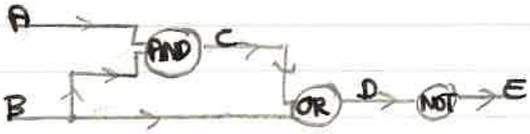
19.

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

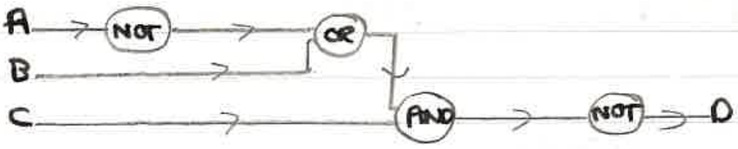
This truth table represents an OR gate

20.


A	B	C
1	0	1
0	1	0
1	1	0

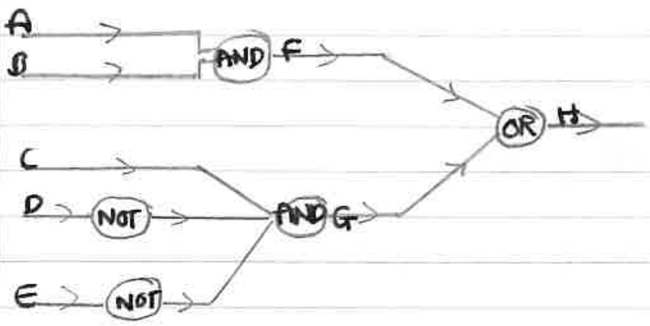
21


A	B	C	D	E
0	0	0	0	1
0	1	0	1	0
1	0	0	0	1
1	1	1	1	0

22


A	B	C	D
0	0	0	1
0	0	1	0
0	1	0	1
1	0	0	1
0	1	1	0
1	0	1	1
1	1	0	1
1	1	1	0

23



A	B	C	D	E	F	G	H
1	1	0	1	1	1	0	1
0	1	1	0	0	0	1	1
0	0	1	0	1	0	0	0
1	1	1	1	1	1	0	1
1	0	0	1	1	0	0	0
1	1	1	0	0	1	1	1

Page 202 Exercise 1

13. 35_8 = 011101_2

= 29_{10}.

11101_2 = $1D_{16}$

14a) 216_8 = ii)010001110_2

b) 67_{10} = $\overset{64}{1}$ $\overset{8}{0}$ $\overset{1}{3}_8$ = 103_8

c) 1010101_2 = ~~v) non of these~~. = 85_{10} iii) 85_{10}.

15a) Base 8 is used because

~~D. There are 8 bits in a byte~~ A. 8 is a power of 2

b)
$$\begin{array}{r} 111010_2 + \\ 11010_2 \\ \hline 1010100_2 \end{array}$$

B 1010100_2

c)
$$\begin{array}{r} 10000_2 \\ -\ 1_2 \end{array} = \begin{array}{r} 10000_2 + \\ +11111_2 \\ \cancel{1}011111 \end{array} = 01111$$

$$\begin{array}{r} 16_{10} \\ -\ 1_{10} \\ \hline =\ 15_{10} \end{array}$$

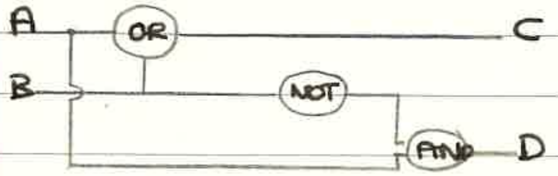
1111_2

d) 001010_2
2C $110101_2 + 1_2$

$$\begin{array}{r} 110101_2 \\ 1_2 \\ \hline 110110_2 \end{array}$$

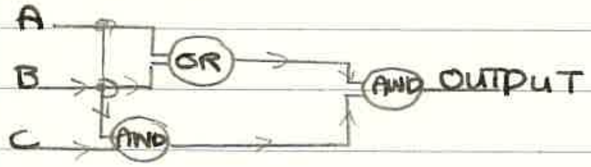
D) 110110_2

23)

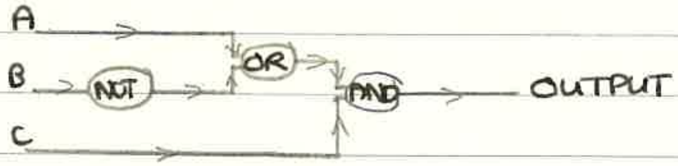


A	B	C	D
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	0

24) (A OR B) AND (A AND C)



25) (A OR NOT B) AND C



Binary Fractions

12·1·84

Column Headings

$x=2$	x^3	x^2	x^1	x^0	x^{-1}	x^{-2}	x^{-3}
	8	4	2	1	$1/2$	$1/4$	$1/8$
eg	1	0	1	1 ·	1	0	1_2

↑ Binary Point

$= 11 \, {}^5/8_{10}$

$= \underline{11 \cdot 625}_{10}$

N.B $\frac{1}{2} = 0{\cdot}5$ $\frac{1}{4} = 0{\cdot}25$ $\frac{1}{8} = 0{\cdot}125$ $\frac{1}{16} = 0{\cdot}0625$

Representation of Numbers

To represent characters bits are arranged in a certain pattern so that each character has it's own unique pattern. A normal character set of 64 would require 6 bits to represent it ($2^6 = 64$) This group of bits which come together to form a character set is called a byte (here a 6 bit byte)

ASC II code

This stands for the American Standard Code for Information Interchange.

In this code the letter 'A' is represented by the binary number 1000001_2 (65_{10})

Thus the word "HELLO" would be represented by

$H = 72_{10} = 01001000_2 = 48_{16}$
$E = 69_{10} = 01000101_2 = 45_{16}$
$L = 76_{10} = 01001100_2 = 4C_{16}$
$L = 76_{10} = 01001100_2 = 4C_{16}$
$O = 79_{10} = 01001111_2 = 4F_{16}$

Representation of numbers

1) Binary Coded Decimal (B·C·D)

e.g 8 4 2 1 weighted B·C·D code, so called because the decimal digits are formed from the binary code using the standard 8421 column headings.

$2 \quad 0 \quad 9 \quad 4_{10}$ (use groups of 4 bits)

$= 0010\,0000\,1001\,0100_2$

Decoding from binary to decimal in B·C·D is much easier than standard binary

e.g 000111000101_2

$1 \quad 12 \quad 5_{10}$

$= 1125_{10}$

B·C·D is often used in simple machines such as calculators which require fast conversion for the decimal display. However it is less efficient because it requires more bits for number storage

2) Negative Numbers

a) Sign and magnitude (sign and modulus)

Using an 8 bit byte the largest positive Integer we could represent would be 255. However with the sign and modulus method using the same size byte would represent numbers in the range -127 to +127. The first bit indicates the sign e.g 64 32 16 8 4 2 1

sign → 0 1 0 0 0 0 0 0 = +64

bit → 1 1 0 0 0 0 0 0 = −64

b) **Twos Complement Method**

e.g $+5 = 101_2$

$-5 =$ 2C $= 011_2$ using 3 bit byte

011_2 = what negative number?

= 2C $= 101_2 =$ positive value $= +5$

$011_2 = -5$ (using twos complementation)

Quick Method

eg $+6 = 0110_2$ (using a 4 bit byte)

$-6 =$ ~~0101~~

1010 start from the right hand side copy up to the first 1 and including it, and then reverse the remaining digits.

e.g What would be the bit pattern that represents the number − 113 using 8 bits.

+ 113 = 64 32 16 8 4 2 1

$0\ 1\ 1\ 1\ 0\ 0\ 0\ 1_2$

− 113 = 10001111

e.g 10101_2 = what negative number

2C $= 01011_2 = +11$

$10101_2 =$ ~~2for~~ ~~=2t~~$_{10}$ −11

Storage of Numbers

The central memory contains certain locations which are of a fixed size. This is called the Word Length. The contents of such a location are known as the Word e.g ~~00~~00100000 is an eight ~~to~~ bit word representing the number 32_{10}

Fixed and floating point representation

So far we have looked at fixed point representation, assuming that the binary point is at the end of the number (ie integers only) We can represent real number (those with binary point) using the fixed method, so long as you know the position of the point. However the range is very limited eg in a 4 bit word;-

$$10_{\uparrow}.01_2 = \underline{2.25}_{10}$$

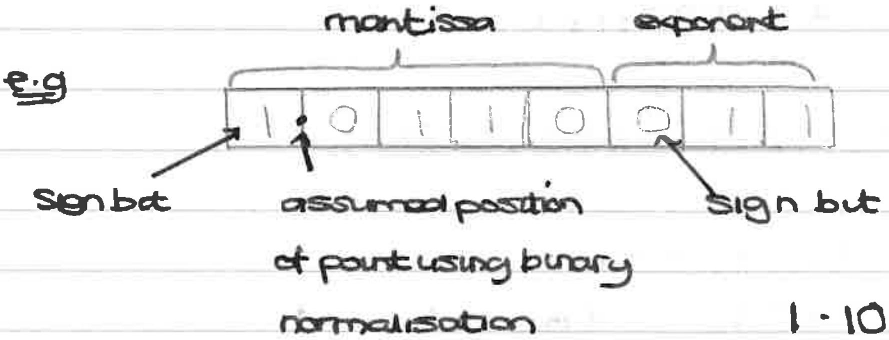
binary point

Floating point representation

i) Sign and Magnitude

The use of this method greatly increases the range of number you can represent in an 8 bit word. There are 2 parts to the representation, the first is the mantissa and the second the exponent

e.g



$$1\cdot10 \times 10_{10}^{+2}$$

$$0\cdot110 \times 10_{10}^{+3}$$

Mantissa	Exponent
10110	011
-0.0110_2	$\times 10_2^{+11}$
	$\times 10_{10}^{+3}$

$$-0.0110_2 \times 10_{10}^{+3}$$

$$= -11.0_2$$

$$-3_{10}$$

$$\overbrace{\boxed{1}\boxed{0}\boxed{1}\boxed{1}\boxed{0}}^{m}\overbrace{\boxed{1}\boxed{0}\boxed{1}}^{e}$$

Mantissa	Exponent
10110	101
$-\ 0.0110$	$-\ 01$
	$\times 10_2^{-1}$

$$-\ 0.0110_2 \times 10_{10}^{-1}$$

$$= -\ 0.00110_2$$

$$= -\frac{3}{16} = -0.1875_{10}$$

ii) Twos Complement

12 bit word

e.g.7	6 bit mantissa	6bit exponenent .
	0.00001_2	000110_2
	assumed position of point	
		$= 6_{10} \Rightarrow 10_{10}^{+6}$

6 places to right $\quad 0.00001_2 \times 10_{10}^{+6}$

$$= 10.0_2 = 2_{10}$$

e.g 2

Mantissa — Exponent

010000 — 111110_2

2C 000010_2 $+2_{10}$

$0\cdot10000$ — $111110_2 = -2_{10}$

2places to left.

$0\cdot10000_2 \quad \times 10_{10}^{-2}$

$= 0\cdot00100_2$

$= \; = +0\cdot125_{10}$

e.g 3

Mantissa — Exponent.

$1\cdot11000_2$ — 111110_2

~~0·010001~~ — 2C $= 000010 = +2_{10}$

2 places to the left — $111110 = -2_{10}$

$01\cdot11000_2$ ~~0·01000₂~~ $\times 10_{10}^{-2}$

~~= 0·0011000₂~~

~~= 1·10010 00~~

~~= 0·0625₁₀~~

$= 0\cdot0111_2$

$= \frac{7}{16}_{10}$

$= 0\cdot4375_{10}$

N.B - Leave the sign bit on, it is part of the number

e.g 4

Mantissa — Exponent.

$1\cdot11000_2$ — 000100_2

* 4places right — 2C $= 4_{10}$

$1\cdot11000_2 \quad \times 10_{10}^{+4}$

$= 11100\cdot0_2$

2C $= 00100\cdot0_2 \quad = 4_{10}$

$= 11100\cdot0_2 \quad = -4_{10}$

Floating Point

Using a 6 bit word represent

a) $+6_{10} = 0110_2$

Mantissa	Exponent
000110_2	000101_2

b) $-4_{10} = 0100_2$

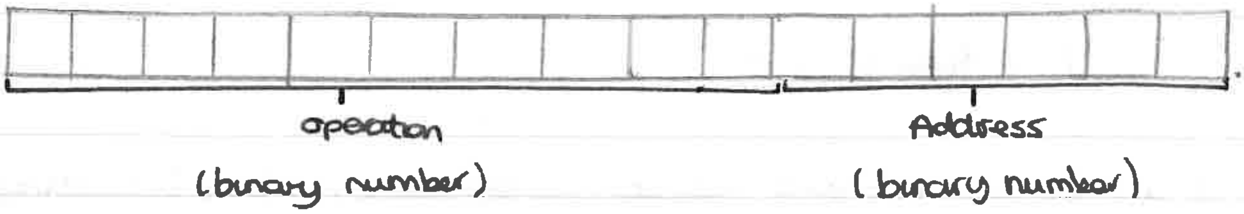
$2C = 111100_2$

Mantissa	Exponent
111100_2	000101_2

c)

1980 Paper II Qu 7

(a) Show how a 16-bit computer word can store an instruction in machine code



b) +12 in an 8 bit computer is

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

c) -12 in a 8 bit word, ones complement method.

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

d) -12, twos complement method

1	1	1	1	0	1	0	0

e) + 11·001001$_2$ in a 12 bit word.

Sign bit →

mantissa									exponent		
0	1	1	0	0	1	0	0	1	0	1	0

f) 2357$_8$ in a 12 bit word.

0	1	0	0	1	1	1	0	1	1	1	1

g) C6 in an 8 bit word

1	0	1	0	0	1	1	0

h) how the character codes for C and 9 ~~illustrated~~ could be represented as two 8-bit bytes in a 16 bit word.

C								9							
1	1	0	0	0	0	1	0	1	0	0	1	1	1	0	0

1981 Paper II Qu 4

4 a) It is sometimes necessary to represent numbers in floating point form inside a computer because bigger numbers and fractions can be represented

b) (i) The largest possible number is

e				m							
0	1	1	1	0	1	1	1	1	1	1	1

(ii) The smallest number greater than zero.

e				m							
0	0	0	0	0	0	0	0	0	0	0	1

(iii) The negative number furthest from zero

e				m							
0	1	1	1	1	1	1	1	1	1	1	1

c) 0011 00011000

= 0101 00001100

d) (i) 0011 00111010

\+ 0011 00110001

= 0011 01101011

(ii) 0001 00000001

\+ 0011 01000010

= 0010 10000100

or 0011 01000010

e) What can you say about the accuracy of answer d(ii)

The number is not very accurate because it carries on after the binary point. This cannot be represented using this system f Why is it preferable to increase the value of the smaller exponent rather than decrease the value of the larger exponent when adding two numbers in floating point form.

If you decrease the value of the ~~f~~ larger exponent, the first digit of the mantissa might become a 1, this would imply it was a negative number.

1982 Paper II

1 (a) Convert 23 to binary

32	16	8	4	2	1
0	1	0	1	1	1_2

b(i) Show how 23 in binary is represented in an eight bit byte.

0	0	0	1	0	1	1	1

(ii) Show how it's ones complement is~~b~~ stored

1	1	1	0	1	0	0	0

(iii) Show how it's twos complement is stored

1	1	1	0	1	0	0	1

d(i) 46 = 23x2 - represent 46 in an eight bit byte

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

e(i) Do the sum 46-23 using 1complement in an 8bit byte .

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

= 1 end around carry

0	0	0	1	0	1	1	0
0	0	0	1	0	1	1	1

(ii) Why could you not do the same in a 6-bit byte .

46 would appear to be a negative number because the first digit

would ~~bet~~ be "1"