Data Representation

Converting between number bases

- Bit - is a single binary digit
- Nibble 4 bits
- Byte – is 8 bits

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

What number do you think this is the binary code is for??

Add together 16+8+1

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Hexadecimal numbers

(base 16) uses letters A - F for 10 - 15.

Covert HEX A7 to Denary

A 7

10 7

1010 0111

Answer: 167

Exam question: Explain why hexadecimal notation is used.
Hexadecimal is used as shorthand for binary and uses fewer digits, so humans make fewer mistakes and find it easier to read

Exam question: Explain the reason why at least nine bits are needed to store 300 different binary patterns.

29 gives 512 patterns whereas 28 gives 256 patterns
28 does not give enough patterns whereas 29 gives more than enough patterns

Character Encoding

ASCII is a standard that defines how each alphabetical letter is represented by a unique 7 bit binary value (for example 'A' is 100 0001). There are 127 binary values for upper and lowercase letters, the digits and most punctuation.

Extended ASCII uses 8 bits to store 256 characters.

Unicode extends this character set by using more bytes to represent many more characters. (16 to 32 bits)

Binary addition

Binary addition rules:

Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Solution

```
  0  0  1  1
+
  0  0  1  1
  ------------
  0  1  1  0
  ------------
Carry 1  1
```

Overflow Error – Extra Bit!

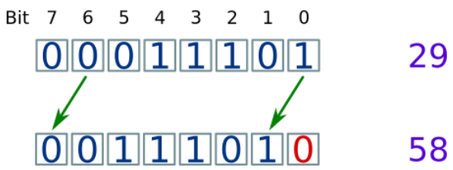
```
  1 1 1 1 1 1 1 1 1
+ 0 0 0 0 0 0 0 0 1
-------------------
1 0 0 0 0 0 0 0 0 0
  \_____________/
      8-bits
```

Binary Shift

Shifts are performed on binary patterns.

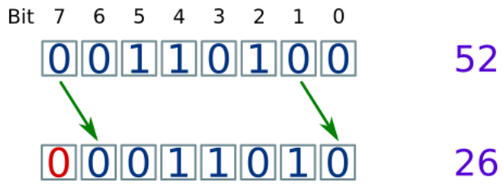
Logical shift

Logical Left Shift - doubling the value.
adding zeros on to the right of a binary value and discarding the bits on the left. Multiples the value by 2



Logical Right Shift halving the value.

adding zeros on to the left of a binary value and discarding the bits on the right. Divides the value by 2



Exam question:

A logical shift right is performed on a pattern.
An arithmetic shift right is performed on the same original pattern.

Describe the reason the results will be different.

An arithmetic shift fills from the left with a copy of the most-significant bit (MSB) whereas a logical shift fills from the left with a 0

An arithmetic shift keeps the most-significant bit (MSB) the same whereas a logical shift always fills the MSB bit with a 0

Arithmetic Shifts

Arithmetic shift left
Exactly the same as a logical shift left
Arithmetic shift right
Copying the most significant bit on to the left of a binary value and discarding the bits on the right. Divides a signed or unsigned number by 2

How to do

To perform an arithmetic shift right of *n* positions, carry out the following steps.

1. Shift each binary digit *n* positions to the right.
2. Discard the rightmost *n* bits.
3. Fill up the empty spaces on the left with a copy of the original MSB.

Worked example

Perform an arithmetic shift right of one position on the binary pattern 1011 0000. Move all the bits one position to the right. Discard the rightmost bit. Put a 1 in the empty space on the left.

Before shift	1	0	1	1	0	0	0	0
After shift	1	1	0	1	1	0	0	0

The result is 1101 1000.

2's Complement

Describe the process of converting a binary number to two's complement.

Copy/keep all the 0s from the right/LSB, up to and including the first 1, then flip the remaining bits

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

2's complement

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

Converting a denary number to binary
Worked example
Convert the denary number –22 to 8-bit binary using two's complement

128 – 22 = 106

106

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

Binary Subtraction

In arithmetic, subtraction can be done by adding a negative number.
Calculate 18 − 8, using 8-bit binary and two's complement.

Step 1 18 converted to 8-bit binary 0001 0010
Steo 2 -8 converted to two's complement 8-bit binary
1111 1000

Step 3 Addition performed:
00010010
11111000
00001010
Result of 0000 1010