

26.3 Merit and Distinction

By Adam Chu



# Task 2 - Discuss Different applications of number systems

The main number systems used inside computer programming are Hexadecimal (Base 16), Octal (Base 8) & Binary (Base 2).

These are used for different things, Binary is the base language that a computer can understand due to the CPU being a mass of transistors which can have two states 1 = On or 0 = Off. All instructions have to be converted to Binary when they are compiled (C and its variants) or interpreted (JavaScript). Binary has the main flaw that it’s hard to read and gets very long due to its low base count. For example RGB colour codes use 1 byte per channel which is 010 -25510 (25510 = 111111112), this gets exponentially bigger for example with RGB colour codes you would need 24 characters (without another set for Alpha (With Alpha it would be 32 Characters)).

Binary is used when creating Raster Images as the way they work is they have a grid of individual blocks of colours called Pixels, this is good at appropriating data and reacts well to lossy compression like turning Raw / DNG to JPG without losing noticeable information (unlike text). The issue with Binary and Raster Information is that when the image is scaled up, the amount of information does not change causing degradation of quality in the image itself.

Binary is also used to store sound files inside of a computer as the computer does ‘Sampling’ to get a digital signal. Computers cannot natively interpret analogue signals they have to measure the Amplitude of the Sound file at multiple times per second (Hertz) this allows the computer to replicate the sound wave. Again the more amount of bits used allows for more dynamic range in the volume of the sound wave, which results in better fidelity and larger volume.

The idea behind the Hexadecimal number system is to shrink binary numbers by a factor of 4 (0 - 15 ->, 0 - F). This is ideal as most people cannot mentally workout long binary strings and operations whereas they could work out shorter Hexadecimal operations, so the Hardware & Software Designers usually think in and work with Hexadecimal. For example using the colour codes from before most of the time they are remembered in Hex as in FFFFFF16 is white and 00000016 is Black. Another benefit to working inside Hexadecimal (and an issue with Octal) is that most sets of data are grouped into Bytes or 8 Bits due to the grouping for Hexadecimal you can shrink 1 byte from 8 bits into 2 Characters. When compared to Octal this is worse as most computer systems use multiples of 16, 16/32/64. Whereas in Octal you would need an extra digit for 0 due to it 88being

Octal is another short hand that allows the user to map 0111 (7) to 0000 (0), using the numerical symbols of (0 - 7). This is helpful as it shortens Binary threefold, and was used similarly to how Hex is used today because the Bit-Width was 24-Bit and 32-Bit which are 3 and 4 Octets respectively. This allowed the programmer to calculate large numbers and operations much easier than using binary, an example of a CPU that used 24-Bits for the ALU is called the eZ80 microprocessor. This was important when it was used natively and still is used inside Linux, MacOS & Android when assigning file permissions with 0 - 7 representing different parts of the metadata for each item / directory.

|  |  |
| --- | --- |
| 0 | No Permissions |
| 1 | Execute Permission |
| 2 | Write |
| 3 | Execute and Write Permission |
| 4 | Read Permission |
| 5 | Read and Execute Permission |
| 6 | Read and Write Permission |
| 7 | All Permissions |

But again the only reason that Octal is being used in these programs, is the fact that they do not want to re-write the system to add proper support for Hexadecimal a more compact system, this could be because of the Open Source Nature of it and the lack of funding for changing all versions of Linux to support this change.

Some RISC languages allow for the use of Hexadecimal inputs to replace the normal TXT format as the OPCODE and OPERAND both take up 4 bits inside of one byte. Whilst using Hexadecimal it allows you code much faster than using Binary: as 1001 0001 (9 1) as using two characters to represent this. This instruction is 9 -> Input / Output and 1 which Selects Input from the two functions. Whilst Using the Text can be considered quite fast to code it is still much larger than using Hex INP.

Hexadecimal is also used for IPv6 due to it being 128 Bits Long when converted into Hexadecimal it becomes 32 Characters in length, this allows IPv6 to have more potential IP Addresses whilst being legible. This is also true for Errors inside of a IDE, usually they will use Hexadecimal to condense the list of errors and to collate them like the first characters in Hex are related to the cause of error.



Compiler error C25 is a series of errors generated when the compiler detects incomplete function callings and incomplete lines of code.

# Task 3 - CIDR Vs Classful Addressing

The idea behind the Classful addressing scheme is that the size of business’ networked devices would allow different IP addresses to be segregated. There are 3 main classes called A, B & C respectively which separate the 4 sections of 4 Decimal numbers, also when a Class is bought you own all potential addresses causing a lot of wasted addresses and having to pay for total number of addresses not the size you need. This was introduced in 1981 to replace the older way of Addressing which only allowed for 128 Networks to exist in total with 126 (+2 because of the broadcast ID & Network ID) devices in network as it was designed for the Military and University with higher research purposes. The older standard was:

Network ID. Blank Byte. Blank Byte. Host ID

The reason why there was blank bytes was for expansion as they wanted to allow the IPv4 system to adopted to more normal businesses without re-creating the addressing system.

Class A: has 128 Networks that could be sold to the populous in total, but they allow for 16,777,216 Hosts (-2 because of the broadcast and network id). The idea behind group A having not many potential networks is at the time (1981 - IPv4). There weren’t enough large scale businesses that actually would use a Class A, network package: for example Microsoft which opened in 1975.

Class B: Had 2 Bytes for the Network ID with 2 Bytes for the potential Hosts which allowed the amount of Networks to increase from 28 to 2 16. This had 16384 sellable networks, with each business able to hold 65,556 network attached devices, this was designed for businesses with multiple offices but not at the mega-corporation level like: Apple or Microsoft, more like Lenovo in the early days (it opened in 1981). This package was not for the average household and small business, which left it more practical to buy Multiple Class C addresses just to reduce wasted IP addresses.

Class C: is a class designed for smaller businesses and households as the Network ID is made up of 3 out of the 4 sections. 212 = 2,097,152 is with 256 devices on each network, this has the largest amount of networks to sell as they would be sold to small start-up businesses and normal households.

Classless Addressing is where the user buys multiple Classes of the smaller variety and merge them into one large (+ 2 for the broadcast and network IDs), Also it allows you to split a network into individual sub-nets which are useful when you would have extra IP addresses that would normally be wasted and you have multiple offices which you want to connect to its own LAN. Also with Classless subnets you can use VLSM (Variable Length Subnet Mask) which replaces the Classful Subnetting with Classful Subnetting all of the dividers between each subnet have to be equal more than 2 subnets if one needs > 83 Addresses. With Classful Subnetting you need 85 Addresses if you have 3 Subnets on a Class C address.

|  |  |  |
| --- | --- | --- |
| Dorchester - 40 Hosts | Decimal | Binary |
| Brief : Subnet Number 195.20.5.0/24 11000011‬.00010100.0000 | | |
| No. of host bits | 6 | N/A |
| No. of useable host addresses | 62 -2 = 60 | N/A |
| Network ID\* | 195.20.5.0/26 | 11000011.00010100.00001001.00000000 |
| Subnet Mask | 255.255.255.192 | 11111111.11111111.11111111.11000000 |
| First host | 195.20.5.1 | 11000011‬.00010100.0000 |
| Last host | 195.20.5.62 | 11000011‬.00010100.0000 |
| Broadcast Address | 195.20.5.63 | 11000011‬.00010100.0000 |

## Subnets

|  |  |  |
| --- | --- | --- |
| Bournemouth - 66 Hosts | Decimal | Binary |
| Brief : Subnet Number 195.20.5.0/24 11000011‬.00010100.0000 | | |
| No. of host bits | 7 | N/A |
| No. of useable host addresses | 127 - 2 = 125 | N/A |
| Network ID\* | 195.20.5.128/25 | 11000011.00010100.00001001.10000000 |
| Subnet Mask | 255.255.255.128 | 11111111.11111111.11111111.10000000 |
| First host | 195.20.5.129 | 11000011‬.00010100.0000 |
| Last host | 195.20.5.254 | 11000011‬.00010100.0000 |
| Broadcast Address | 195.20.5.255 | 11000011.00010100.00000101.11111111 |

|  |  |  |
| --- | --- | --- |
| Weymouth - 10 Hosts | Decimal | Binary |
| Brief : Subnet Number 195.20.5.0/24 11000011‬.00010100.0000 | | |
| No. of host bits | 4 | N/A |
| No. of useable host addresses | 15 - 2 = 13 | N/A |
| Network ID\* | 195.20.5.13/28 | 11000011.00010100.00001001.11000000 |
| Subnet Mask | 255.255.255.240 | 11111111.11111111.11111111.11110000 |
| First host | 195.20.5.193 | 11000011‬.00010100.0000 |
| Last host | 195.20.5.254 | 11000011‬.00010100.0000 |
| Broadcast Address | 195.20.5.207 | 11000011.00010100.00000101.11001111 |

I used VSLM to separate the Subnets up as you have 3 un-even subnets and it was cause wastage between each network. With VSLM you can make a subnet which holds 15 IP Addresses instead of giving each office 85 Addresses INC. Broadcast and network IDs. With this current set up for the business using 2 bits for the Subnet Number we have up to 4 possible subnets which we are using 3 of 00, 01, 11, 10. This would allow for expansion as we are not using it.

## Proof

195.20.5.197 - Weymouth

11000011‬.00010100.0000

11111111.11111111.11111111.11110000

----------------------------------------------------------- Logical AND

11000011.00010100.00000101.11000000

When you take a random hosts IP address and do a logical AND against the corresponding Subnet Mask it will give you the Network ID if its from the correct Subnet.

195.20.5.50

11000011‬.00010100.0000

11111111.11111111.11111111.11000000

----------------------------------------------------------- Logical AND

11000011.00010100.00000101.00000000

Because it’s the first Subnet with an subnet address of .00000000 it would be wrong if the host ID had a logical and with any character in the last section.

195.20.5.145

11000011‬.00010100.0000

11111111.11111111.11111111.10000000

----------------------------------------------------------- Logical AND

11000011.00010100.00000101.10000000

This is the Network ID for Bournemouth as it needed 7 Bits for the Host with 1 being the subnet mask.

# Resub

|  |  |  |
| --- | --- | --- |
| Bournemouth - 66 Hosts | Decimal | Binary |
| Brief : Subnet Number 195.20.5.0/24 11000011‬.00010100.0000 | | |
| No. of host bits | 7 | N/A |
| No. of useable host addresses | 127 - 2 = 125 | N/A |
| Network ID\* | 195.20.5.0/25 | 11000011.00010100.00001001.00000000 |
| Subnet Mask | 255.255.255.128 | 11111111.11111111.11111111.10000000 |
| First host | 195.20.5.1 | 11000011‬.00010100.0000 |
| Last host | 195.20.5.126 | 11000011‬.00010100.0000 |
| Broadcast Address | 195.20.5.127 | 11000011.00010100.00000101.01111111 |

|  |  |  |
| --- | --- | --- |
| Dorchester - 40 Hosts | Decimal | Binary |
| Brief : Subnet Number 195.20.5.0/24 11000011‬.00010100.0000 | | |
| No. of host bits | 6 | N/A |
| No. of useable host addresses | 63 -2 = 61 | N/A |
| Network ID\* | 195.20.5.128/26 | 11000011.00010100.00001001.10000000 |
| Subnet Mask | 255.255.255.192 | 11111111.11111111.11111111.11000000 |
| First host | 195.20.5.129 | 11000011‬.00010100.0000 |
| Last host | 195.20.5.190 | 11000011‬.00010100.0000 |
| Broadcast Address | 195.20.5.191 | 11000011‬.00010100.0000 |
|  |  |  |
| Weymouth - 10 Hosts | Decimal | Binary |
| Brief : Subnet Number 195.20.5.0/24 11000011‬.00010100.0000 | | |
| No. of host bits | 4 | N/A |
| No. of useable host addresses | 15 - 2 = 13 | N/A |
| Network ID\* | 195.20.5.192/28 | 11000011.00010100.00001001.11000000 |
| Subnet Mask | 255.255.255.240 | 11111111.11111111.11111111.11110000 |
| First host | 195.20.5.193 | 11000011‬.00010100.0000 |
| Last host | 195.20.5.206 | 11000011‬.00010100.0000 |
| Broadcast Address | 195.20.5.207 | 11000011.00010100.00000101.11001111 |