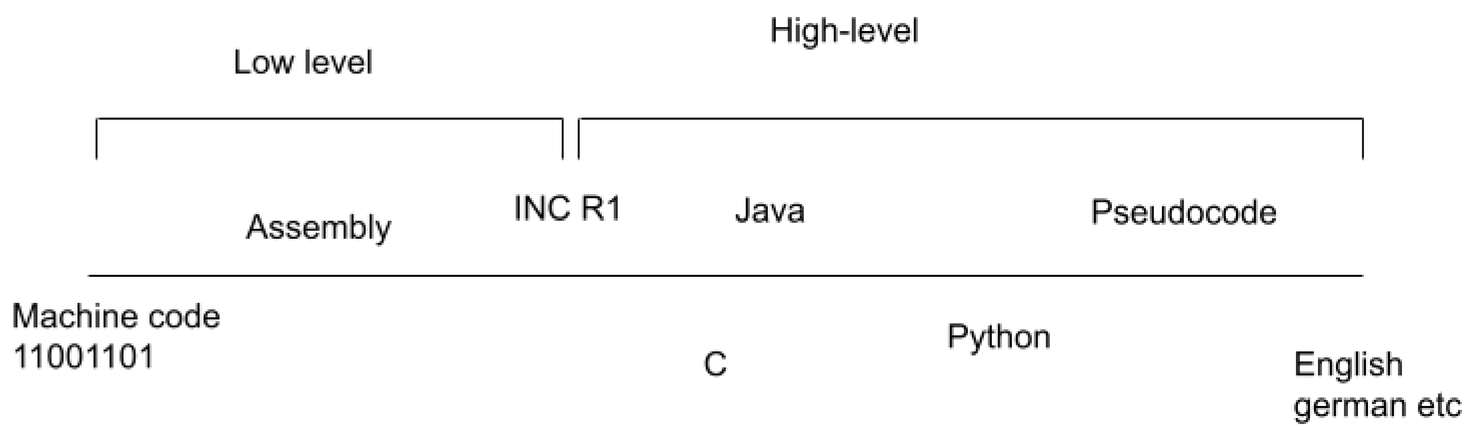
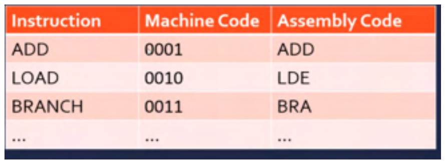
- To build a software model, you need to work out what connects aspects of the real life problem, and make assumptions and simplification
- The model needs to be accurate enough to produce realistic results, but not so detailed that it takes too long to run.
- Creating software model saves tie and money - tests can be repeatedly run, and factors like safety are not issues (such as simulating wars and nuclear reactors)

Machine, assembly & High-level programming languages

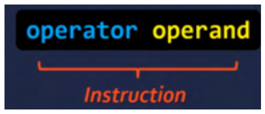


Low level

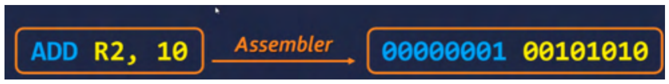
- Machine code has the least abstraction; processors execute it directly
- Assemble and high-level code must be translated to machine code
- Machine code is specific to a processor/family of processors (not portable) and corresponds to its instruction set



Instruction	Machine Code	Assembly Code
ADD	0001	ADD
LOAD	0010	LDE
BRANCH	0011	BRA
...	...	...



- Assembly languages are also low level, but more abstracted
- They use simple mnemonics
- They are also specific to hardware and have a *mostly* one to one relationship with machine code



*as always hex can be used to shorten binary

High-level languages

- Most like ordinary languages, so are far easier to use (still strict syntax)
- They are portable, meaning they can be executed on many computers i.e you don't need to write hardware specific code, like low level
- However some high-level languages are more abstract than others
- Less abstract: pointers, strong types, enums, stacks, templates etc

- More abstract languages leave more work to be done at runtime, so are much slower to execute. With less abstract, you can optimise more

Translators (Assembler, compiler & Interpreter)

… a translator is a program that converts code from one language into equivalent code written in another
- All code needs to be translated to machine code

Assembler

Assembly -> Machine
- Uses the processors instruction set to convert the instructions to the machine code equivalent (produces an object file)
- Optimisation will occur, eg subroutine subprogram calls as if they were inline, calculating values of constants etc

Compiler

High level -> Machine
- A compiler scene through the whole code, and translates it all into machine code
- A compiled program can be directly execute (a binary file is produced)
- After translation, the compiler & source code are no longer needed
    - You can distribute your program without the source code
- Error messages are only shown after scanning the whole code
    - Can be awkward and time consuming for debugging

Interpreter

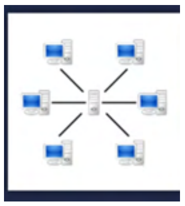
High level -> Machine
- Works line by line -> translates a line, then immediately executes it
- Every time you want to execute code, it must be translate again
- Stops as soon as it reaches an error
- Both the interpreter and the source code are needed at all times
- Slower than a compiler

Some languages are either, but most (modern) languages use both and may only translate part of the way (eg to bytecode or object code)

Networks

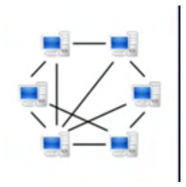
- Are connection between nodes (devices) to share resources
- A personal area network (PAN) is within the range of an individual person eg Bluetooth connection which have a range of around 10 m
- A local area network (LAN) is a network that connects devices close to each other eg same house or school
- A wide area network (WAN) is a network over a broader geographic area (possibly in several locations), eg the internet. In a WAN, some infrastructure is owned by someone else (eg an ISP)
- The internet is a wan made up of many individual LANs.

Client server network



- Every device either a client or server
- A client establishes a connection with the server over the network
- Servers can backup and store data centrally, though they can be expensive and difficult to run

Peer to peer network



- This network configuration has no central server
- Each computer is equal in responsibility and each has the ability to work as both a client and a server.

Factors affecting performance

- Latency - the delay (how fast signals travel)
- Bandwidth - max rate of data transfer (bps)
- Error rate - rate of corruption

- Wired connection are generally faster than wireless
- Bandwidth vary eg Ethernet = 10 Mbps, Wireless = 11 Mbps to 1.3 Gbps
- BUT bandwidth is shared across the network - can become congested
- Wireless range - signal degrades quickly, and signals may be blocked
- Signals at same frequency interfere (leading to data collisions)
- Bus topology = higher error rate

Network protocols and the 4 layer Model

Protocols

= sets of rules for communication

- There need to be accepted rules so devices can be compatible and reliably communicate - these are what protocols are.
- Protocols are usually developed in layers, with each being responsible for a different part of the communication process. TCP/IP has 4:
- Application layer: HTTP/S, FTP, SMTP, IMAP & POP, DNS
- Transport Layer: TCP, UDP
- Network payer: IP =All internet Protocol Suite (TCP/IP) protocols
- Link layer: Ethernet, Wifi

Layers

- Break the process down into manageable, self contained parts (DECOMPOSITION)
    - Easier to develop -> focus on one aspect
        - Easier to develop standards and make software.hardware compatible

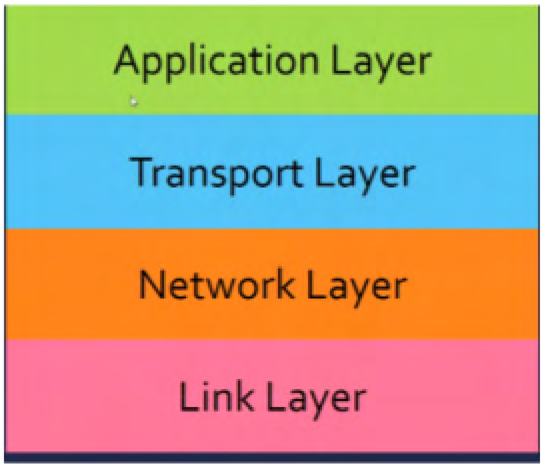
Changing one layer won't affect another, providing the input and output remain in the same format

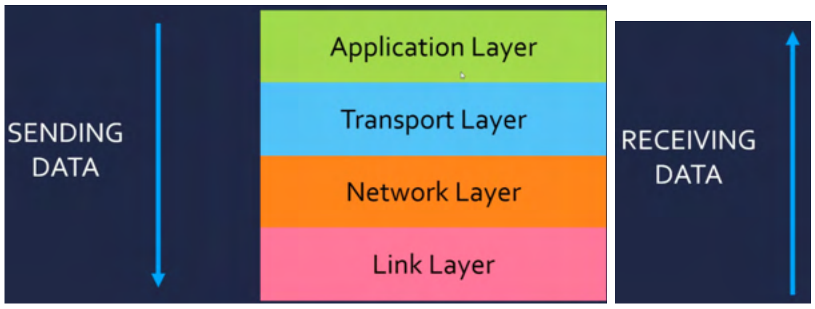
Where the network applications operate (eg web browsers, email clients)

Sets up the communication between the 2 hostel, including agreed rules (like packet size)

Addresses and packages the datm then actually routes it

Where the hardware (eg NICs) and drivers operate


Application Layer
Transport Layer
Network Layer
Link Layer


SENDING DATA
Application Layer
Transport Layer
Network Layer
Link Layer
RECEIVING DATA