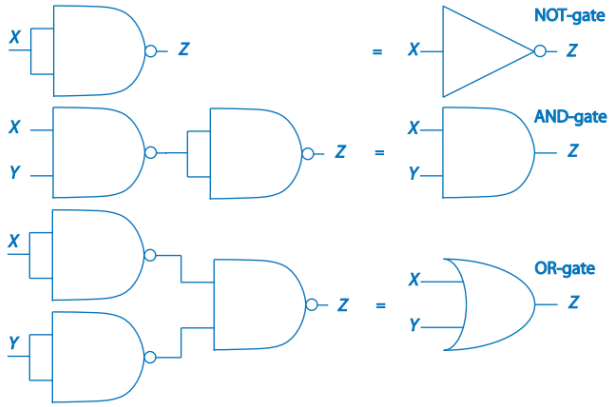
CT Revision

1 What is computer hardware

Three phases of circuit design:

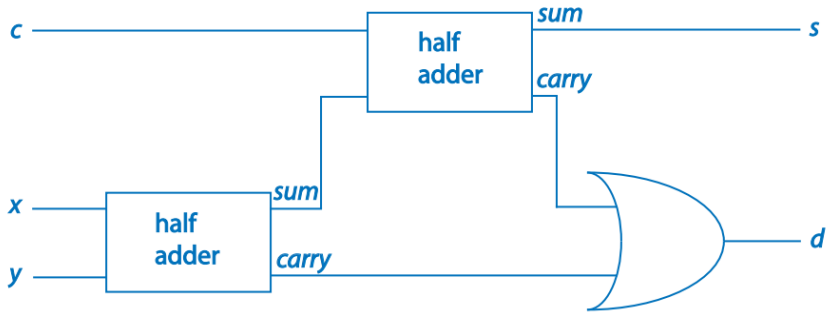
- Functional specification - what the IC is supposed to do
- Register transfer level - Design of chips and links
- Mapped to physical layout - RTL design made appropriate for silicon

NAND gates are functionally complete, as can be shown below



1.1 Adders

Half adders and full adders are capable of doing binary arithmetic, the advantage of a full adder is that it can take in a carry, allowing them to be chained to process more bits



Chaining can work as so, with n bit strings $X_1X_2...X_n$ and $Y_1Y_2...Y_n$

- A half adder adds the first bits of each string, output sum
- The carry is fed into a full adder, along with the next two bits of each string, output sum
- This carry is fed into another full adder, along with the next two bits of each string, output sum
- This keeps going until you reach the nth bit, output sum and carry

1.2 Microarchitecture

Components in a CPU microarchitecture:

- Datapath - Data processing operations
- Control - Tells everything what to do
- Cache - Used to store regularly accessed memory items

1.3 Buses

Types of buses

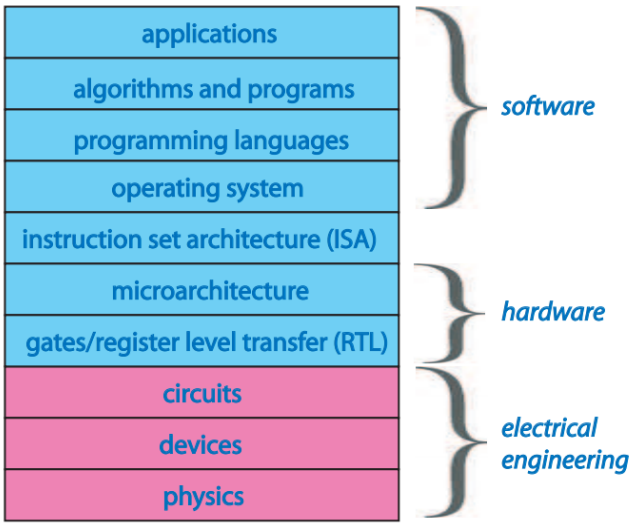
- Data bus - Contents of memory locations between the processor and main memory
- Address Bus - Addresses of locations in main memory

Bus Width - Number of parallel wires in a bus
The width of a the **data bus** determines word size
The width fo the **address bus** determines size of addressable memory

1.4 Processor cycle

- Instruction fetch - Give instruction address, get instruction back
- Instruction decode - Interpret the stored instruction
- Operand fetch - Give address of required data, get data back
- Execute instruction - Perform necessary actions, may include a write back substage

1.5 Layers of abstraction



1.6 Processor electronics

Northbridge - Fast communications amongst critical components
Southbridge - Slow communications amongst non critical components (like peripherals)

Components of an integrated circuit - Transistors interconnected by microscopic wires

The current trend in microprocessor design is to produce multi core processors, which can give better computational improvements.

Moore's Law - Long term transistor capacity doubles every 18-24 months

1.7 Boolean Algebra

Boolean Function - A function $f : 0, 1^n \rightarrow 0, 1$
We can build any boolean function using only AND, OR, and NOT gates

1.8 Theory of IC design

Formal methods - Mathematically prove properties of designs and programs, rather than just relying on empirical testing
HDL - A language which has explicit notations for time and concurrency
von Neumann bottleneck - The limitation of the rate of data transfer between the CPU and memory, this gave rise to caches
Harvard Architecture - Memory is partitioned into data and instruction with dedicated buses for each

1.9 Memory

The cost and performance of memory is proportional to its physical distance from the CPU
DRAM - Uses a transistor/capacitor combination and is cheap and slow
SRAM - Uses a flip flop and many transistors, fast and expensive.
Cache - Memory used to store rapidly stored items
Register - On chip memory locations giving the fastest access to data

2 What is Computer Software

Algorithm - A sequence of precise instructions that can be applied to specific data items
Program - The implementation of an algorithm in a form that can be executed, or at least compiled, by a computer

Programming paradigms:

- Imperative - Statements change a programs state
- Declarative - Programs say what to do, rather than how to do it
- Data Oriented - Work with data by manipulating and searching relations
- Scripting - Automate frequently used tasks

Drivers of the evolution of programming languages:

- Productivity
- Reliability
- Security
- Execution (multi threading etc)

General properties a programming language should have

- Easy to use
- Support testing
- Inexpensive (resources)

Ubiquitous computing - The integration of computers and software into everyday objects and activities
Syntax - The rules that make a program 'legitimately written'
Semantics - The rules that govern what a program 'means'

2.1 Prolog

Prolog programs work with a list of facts and rules, then takes queries about those facts.
An example of valid syntax is:

```
grandma(X,Y):- mother (X,Z), parents(_,Z,Y)
```

3 How does hardware execute software

ISA - The interface between hardware and software, its primary component is its assembly language Examples of MIPS instructions

- lw - move value from register to memory
- addi - add a register and a value
- j - jump to a memory location

Width of a MIPS instruction - The length of the instruction
Groups of MIPS instructions

- I Type - Data transfer
- R Type - Registers
- J Type - Jumps

PCB - A data structure that the kernel uses in order to manage a process.
A PCB is used so the CPU can better handle processes, it contains:

- The process' unique ID
- The current state of the process
- CPU scheduling information such as priority

Functions of an OS

- Virtualise a machine - Provide abstractions that present clean interfaces to make the computer easier to use
- Start and stop programs - free up memory
- Manage memory - Make sure programs can run even if they request in use memory
- Handle I/O - Interrupt on Mouse movement

Data security - Ensure that memory allocated to each program is kept separate ans secure from other programs
Interrupt - Input from the user which should be handled immediately by the CPU

- The OS ensures actions are coordinated
- Ensures no programs end up being closed because a mouse is clicked

Process - A currently executing program, it consists of:

- Thread - A sequence of instructions in the context of a sequential execution
- Address Space - Memory locations to read from and write to

Multiple processes can be run on one processor, provided there is no mutual execution
Mutual Exclusion - Ensuring two threads are not in the critical selection at the same time
Critical Selection - Exclusive access to some shared resource such as memory location

Virtual memory - Moving data backwards and forwards between RAM and the hard disk as and when required
Virtual memory is required when there is insufficient RAM to store what all the running programs wish to store

Life Cycle of a process:

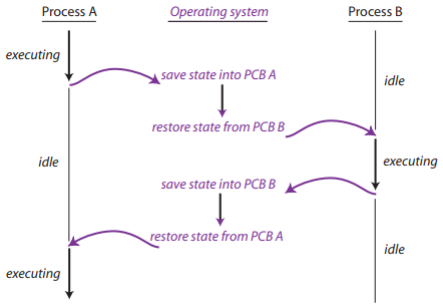
1. **new**: The process being created
2. **ready**: the process not being executed in the CPU but is ready to execute
3. **running**: the process is executing on the CPU
4. **blocked**: the process is waiting for an event

5. **exit**: the process has finished

This life cycle has the following state transitions:

- **admit**: process moved to run queue
- **dispatch**: CPU allocated to executable process
- **timeout/yield**: executing process releases access to the CPU
- **event/wait**: process waiting for an event and releases access to the CPU
- **event**: an event occurs and wakes up a process
- **release**: process terminates and releases access to the CPU and other resources

Context Switching - Where the operating system pauses one process and resumes another



4 What type of problem do we have to solve?

Key notions of a general problem:

- Computation
- Resource
- Correctness

Abstraction - Hiding information, such as to make the problem easier to solve
When developing an abstraction of a real world problem, the abstractions must mirror reality sufficiently closely so conclusions are valid for the real world problem

A decision problem D consists of:

- A set of instances I
- A subset $Y \subseteq I$ of yes instances

A **graph**, G(V,E), has a finite set V of vertices and a set E of pairs of distinct vertices where no edge is repeated.
Proper Colouring - Colouring in which any two adjacent vertices are coloured differently.
Optimal Colouring - A proper colouring that uses the smallest number of colours

Four colour theorem - Every planar graph can be properly coloured using just 4 colours
Greedy Algorithm - An algorithm that works through a list ”greedily” choosing the best thing to do
Brute force colouring - Try all possible colourings to see if it is legal or not

A search problem S consists of:

- A set of instances I
- A set of solutions J
- A binary search relation $R \subseteq I \times J$

An optimisation problem is defined as:

- A set of instances I
- For every instance $x \in I$ there is a set of feasible solutions $f(x)$
- For every instance $x \in I$ and for every feasible solution $y \in f(x)$ there is a value $v(x, y) \in \mathbb{N} = 0, 1, 2, ...$ giving the measure of the feasibility solution y for the instance x
- There is a goal which is either min or max

Independent set - A set of vertices in a graph where no two are adjacent
Maximal Independent Set - Not contained in any other independent set

5 How do we develop algorithms to solve fundamental problems?

Selection sort - Constant time complexity $O(n^2)$
Bubble sort - Worst case complexity $O(n^2)$ best case complexity $O(n)$

6 How do we know an algorithm solves a particular problem

Ways a software bug can arise:

- Programming error
- Secondary error like a compiler

Waterfall model:

- **Requirements phase** - Produce a specification of what a program is supposed to do
- **Design phase** - Compose a plan for the solution involving Algorithms, Data Structures etc
- **Implementation phase** - Write the code
- **Verification phase** - Test and debug
- **Maintenance** - Continual modification/updating

Formal Methods - Mathematically based techniques for the formal specification and verification of software and hardware systems
Software Testing - Evaluating an attribute or capability of a program or system and determine that it meets its required results
Software Metric - A measure of testing
Totally Correct - Correct with respect to some specification and terminates on every input
Partially Correct - Only correct with respect to the specification
Fundamental aspects of a successful proof of total correctness when using invariants

- Initialisation - Show that the invariant holds prior to the first iteration of the loop
- Maintenance - Show that the invariant is maintained
- Termination - Show that the invariant holds when the loop stops

Formal Specification - The study of the specification of a program's properties using a specification language defined by logic
Formal Verification - Mathematical techniques used to ensure that a design conforms to some precisely expressed notion of functional correctness

7 How can we measure how good an algorithm is?

Resources to measure about an algorithm:

- Time
- Memory

$f = O(g)$ if there exists $n_0 \in \mathbb{N}$ and $k \in Q$ such that $f(n) \leqslant k \cdot g(n)$ whenever $n \geqslant n_0$
$f = \Omega(g)$ if there exists $n_0 \in \mathbb{N}$ and $k \in Q$ such that $g(n) \leqslant k \cdot f(n)$ whenever $n \geqslant n_0$
$f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$

8 What exactly is a hard problem?

Tractable - Can be solved by an algorithm of time complexity $O(n^k)$
Efficiently checkable - Given a potential witness that some instance is a yes-instance, we can check in polynomial time whether the witness is indeed a witness
Non deterministic algorithm - An algorithm which guesses the solution, then checks if it's correct **P** - The complexity class of efficiently solvable problems
NP - The complexity class of efficiently checkable problems

A polynomial time transformation from X to Y is an algorithm α that:

- Takes an instance I of X as input and provides an instance $\alpha(I)$ of Y as output
- Is such that an instance I of X is a yes-instance iff the instance $\alpha(I)$ of Y is a yes-instance

Cook's theorem - P=NP iff $SAT \in P$