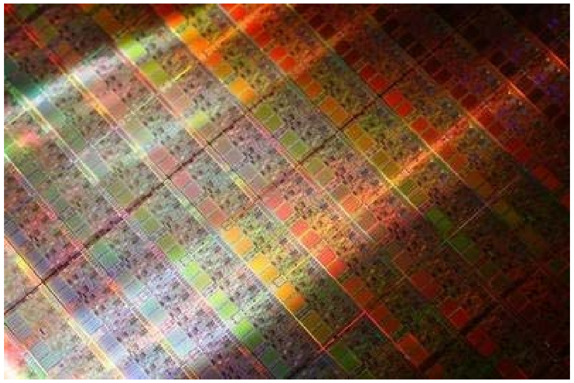
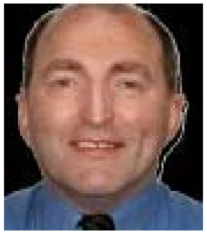
Computer Systems Fundamentals

Term One

Introduction



Bob Cherry



Leigh Travis

Course Logistics

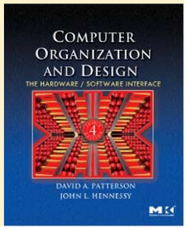
- Lecturers
  - Bob Cherry — Unit Leader — (r.cherry@mmu.ac.uk JD E 150)
  - Leigh Travis — (l.travis@mmu.ac.uk JD E 136)
- Course Page
  - http://moodle.mmu.ac.uk/course/view.php?id=9617
  - Visit regularly (at least weekly) for updates and announcements!
- Lectures — 1 hour per week
  - All Saints — Manchester Lecture Theatre — Monday 12pm
  - John Dalton Main building — JD C0.14 — Monday 4pm
- Labs — 2 hours per week

- Textbook (Required)

David A. Patterson and John L. Hennessy,

Computer Organization and Design: The Hardware/Software Interface,

4th Edition, Morgan Kaufmann, October 2008



Evaluation and Grading

- 2 assignments – Mass mark is $\geq$ 40% average (overall) for both assignments
- Term 1 One assignment : Digital Logic / Assembly language programming
  - Test – week 6 25%
  - Assembly Language Program 25%
- Term 2 One assessment : Mathematics
  - 50%

Scope of Course

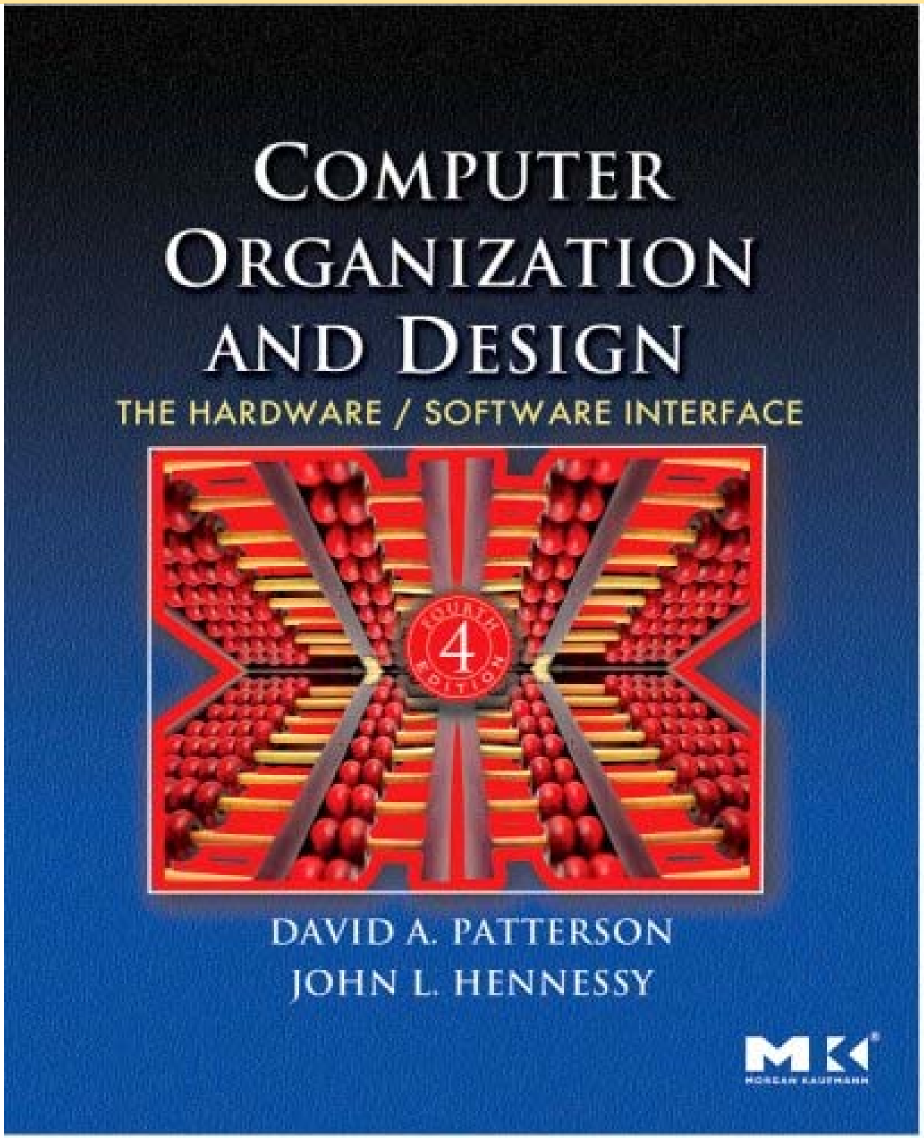
Lecture Topics (term 1)

- Logic Gates
- Combinational Logic
- Boolean Simplification
- ALU
- Fetch Execute Cycle
- Number Representation

then it's over to Dr. Bob

Lecture Topics (term 2)

- Geometric mathematics



What you will Learn in ***Term 1***

- How are programs written in high level languages (C or Java)

and translated into the language of the hardware

- How hardware ***executes*** the resulting program

- What is the ***interface*** between software and hardware

- How software ***instructs*** hardware to perform the needed function

Don't Forget …

Ask Questions in class!

The Computer Revolution

Makes novel applications feasible

- Computers in automobiles
- Cellular communication network ‘phones
- Human genome project
- World Wide Web
- Search Engines

Computers are pervasive

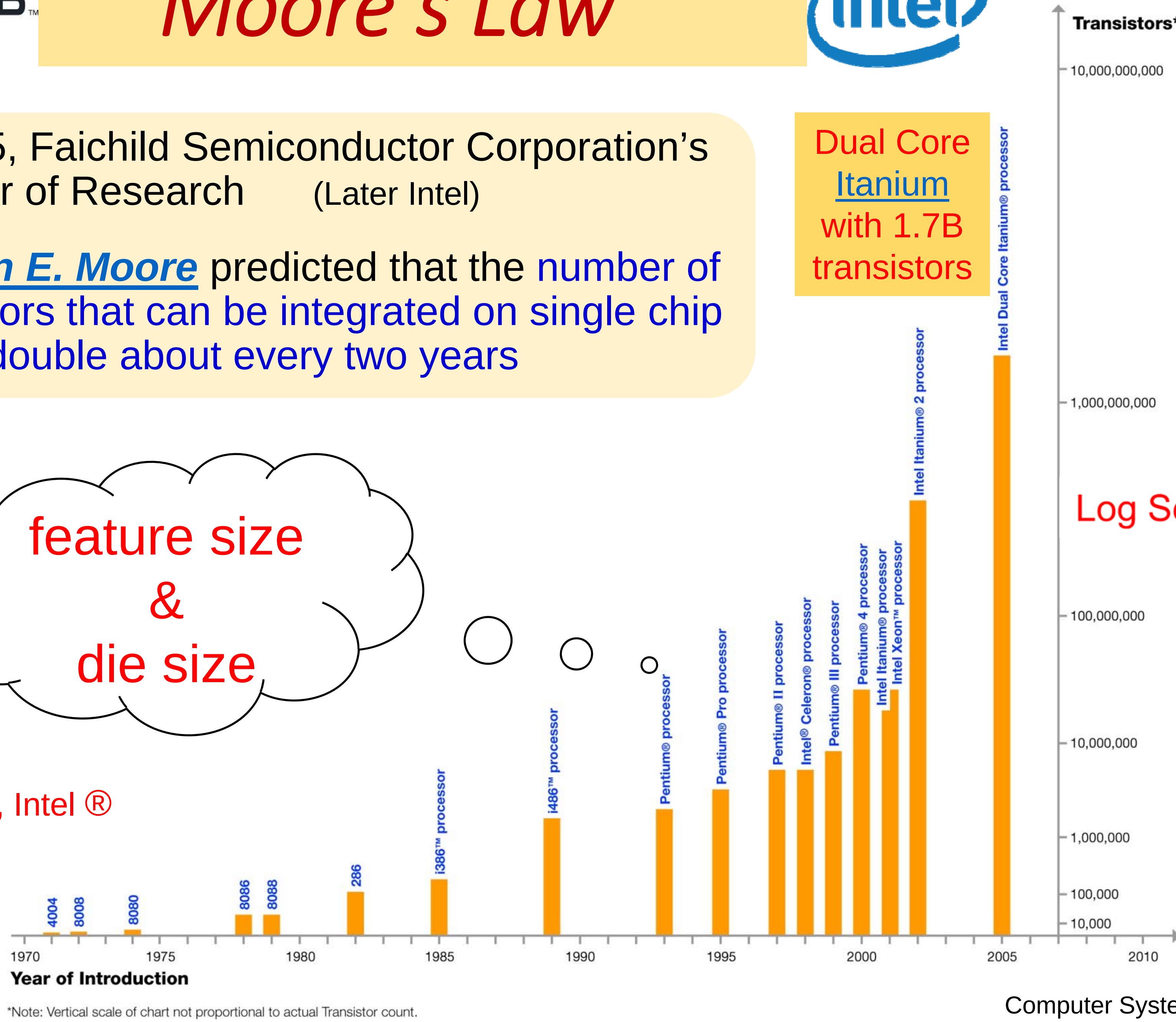
Progress in computer technology

- Underpinned by Moore’s Law

Moore's Law

- In 1965, Faichild Semiconductor Corporation's Director of Research (Later Intel)

  Gordon E. Moore predicted that the number of transistors that can be integrated on single chip would double about every two years

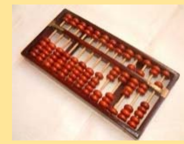


Courtesy, Intel ®

0. Computational Systems

Origins of the Computer System

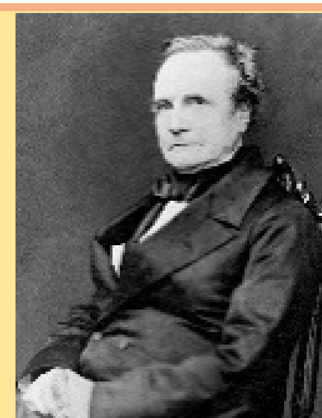
Abacus (Sudan) 2700-2300 BC

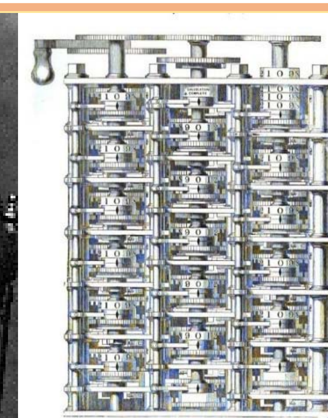


John NAPIER (Scotland) 1550 – 4 April 1617

Charles BABBAGE (UK) 26 December 1791 –18 October 1871

- Difference Engine, **Analytical Engine ....**
  - Programming by Augusta, **Ada KING,** Countess of **Lovelace**, (*née* **Byron)** (1815–1852) UK

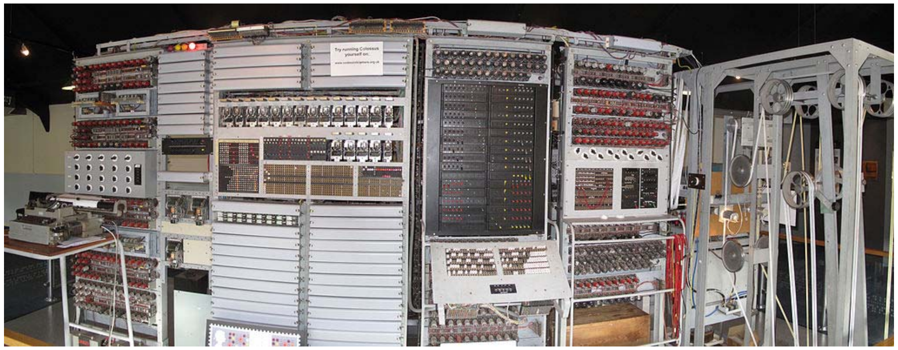


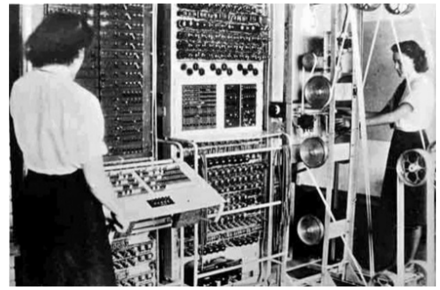


Adding machine Burroughs Corporation (now Unisys) (USA) 1888

Colossus computer

(UK) Tommy Flowers, 1943



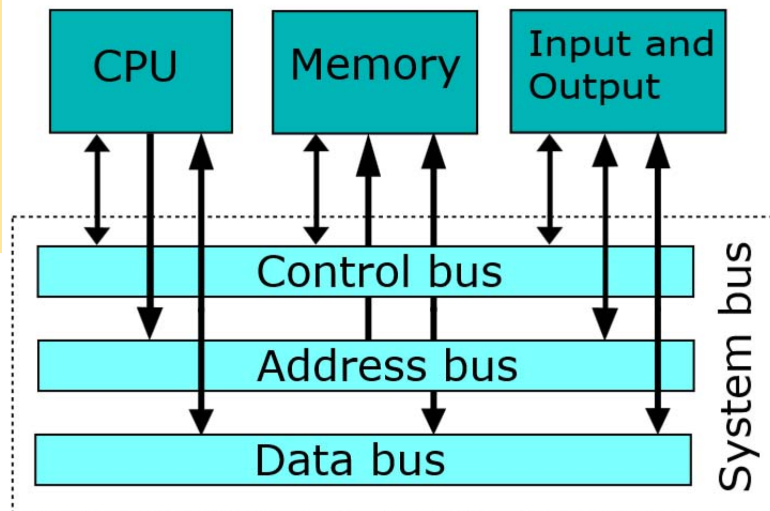


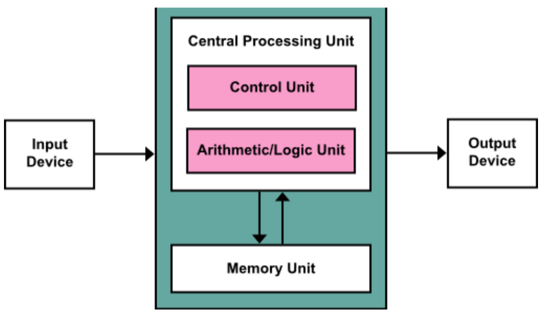
John von Neumann

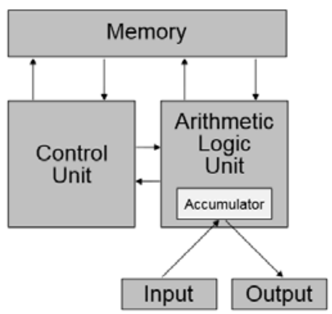
(**Hun**, **USA**) December 28, 1903 – February 8,

Von Neumann architecture

Von Neumann architecture, also known as the **Von Neumann model** and **Princeton architecture**, is a computer architecture based on that described in 1945 by the mathematician and physicist John von Neumann and others in the *First Draft of a Report on the EDVAC*.[1

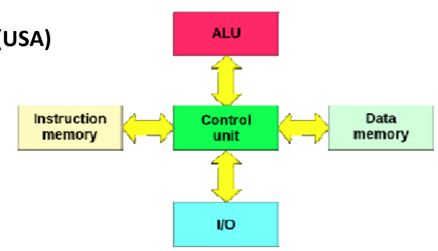






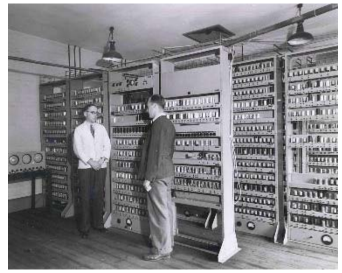
Harvard architecture

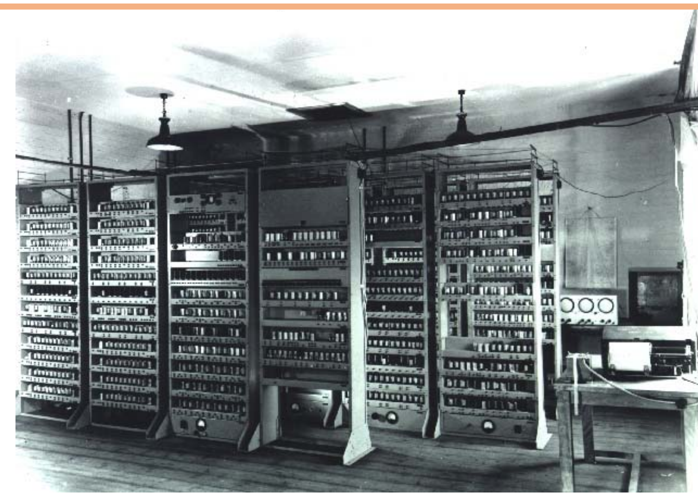
(USA)



Electronic Delay Storage Automatic Calculator (EDSAC)

an early British computer.[1] Inspired by John von Neumann's seminal *First Draft of a Report on the EDVAC*, the machine was constructed by Maurice Wilkes and his team at the University of Cambridge Mathematical Laboratory in England.





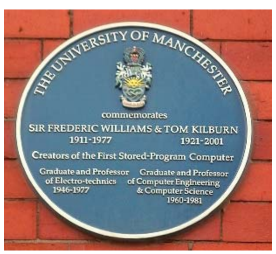
ENIAC Electronic Numerical Integrator And Computer

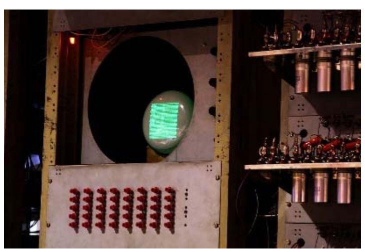
(USA) 1946

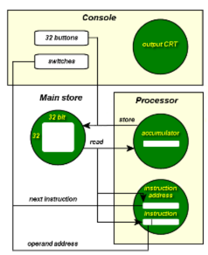
When ENIAC was announced in 1946, it was heralded in the press as a "Giant Brain."
It had a speed on the order of one thousand (10^3) times faster than that of electro-mechanical machines; this computational power, coupled with general-purpose programmability, excited scientists and industrialists alike.

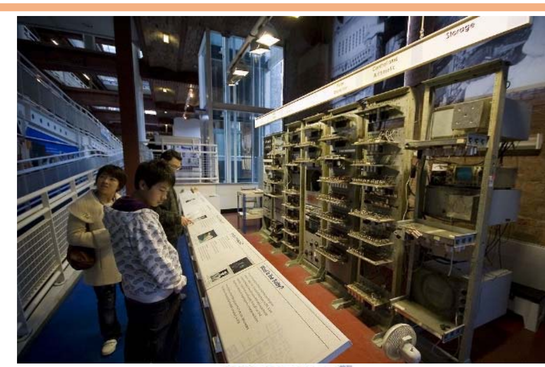
Manchester Small-Scale Experimental Machine (SSEM)

nicknamed **Baby**, was the world's first stored-program computer (Manchester University, UK) 1948









SSEM's instruction set[27]

Binary code	Original notation	Modern mnemonic	Operation
000	S, Cl	JMP S	Jump to the instruction at the address obtained from the specified memory address S[a] (absolute unconditional jump)
100	Add S, Cl	JRP S	Jump to the instruction at the program counter plus (+) the relative value obtained from the specified memory address S[a] (relative unconditional jump)
010	-S, C	LDN S	Take the number from the specified memory address S, negate it, and load it into the accumulator
110	c, S	STO S	Store the number in the accumulator to the specified memory address S
001 or 101[b]	SUB S	SUB S	Subtract the number at the specified memory address S from the value in accumulator, and store the result in the accumulator
011	Test	CMP	Skip next instruction if the accumulator contains a negative value
111	Stop	STP	Stop

The SSEM's three bit instruction set allowed a maximum of eight (2^3) different instructions

LEO I

(Lyons Electronic Office I) (UK, 1951).

The first computer used for commercial business applications

The Digital Equipment Corporation **PDP-11**

(DEC, USA) a series of 16-bit minicomputers sold by Digital Equipment Corporation (DEC) from 1970 into the 1990s

DEC VAX

(DEC, USA) an instruction set architecture (ISA), developed by Digital Equipment Corporation (DEC) in the mid-1970s. The VAX-11/780, introduced on October 25, 1977, was the first of a range of popular and influential computers implementing that architecture.

FERRANTI

computer systems

GEC Computers

International Computers Limited,

or **ICL**, was a large British computer hardware, computer software and computer services company that operated from 1968 until 2002.

1. Digital Logic Gates

Aims

- to introduce the lowest level processing circuits in digital computers
- to show that the behaviour of a logic gate is specified by its truth table
- to show how logic gates are combined on a chip
- to show how logic gates can be combined to produce useful circuits such as multiplexors and comparators

1.0 Introduction

- Digital computers store and process ***binary*** data

  binary data is represented by ***1***'s and ***0***'s

- The basis of all processing is the ***logic gate***

  the **logic gate** is a ***circuit*** that implements a **logical function**

- Each **gate** is represented by its own **symbol** and the symbols can be connected together to construct circuit diagrams

- The behaviour of a logic gate can be explained in words,

  with each gate behaving exactly the same given the same set of inputs

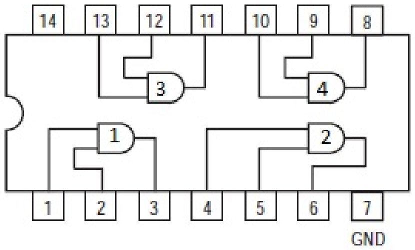
- **Truth tables** can be used to reason about the behaviour of gates and circuits

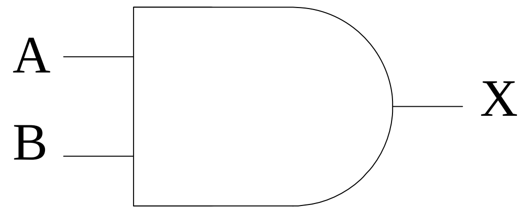
- There are <u>six basic logic gates</u> : ***AND***, ***OR***, ***NOT***, ***NAND***, ***NOR***, ***XOR***

1.1 Gates

AND Gate

7408

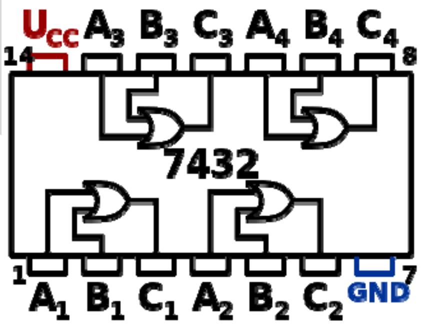


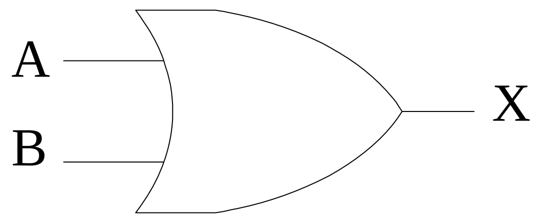


The **AND** gate outputs a 1 when all the inputs are 1 (X = 1 when both A **AND** B are 1)

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

OR Gate



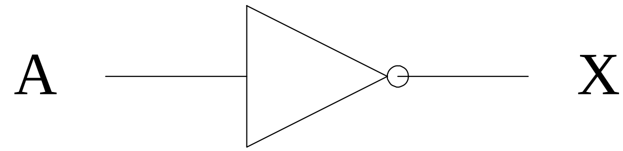


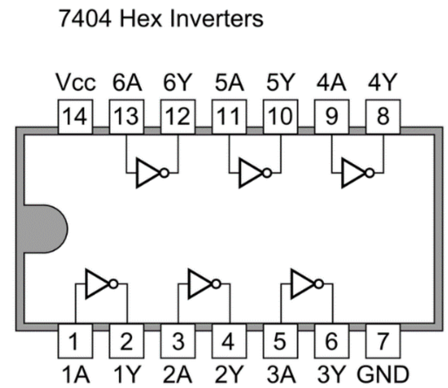
The ***OR*** gate outputs a **1** when ***either*** **OR** ***both*** of the inputs is a **1**

(**X** = **1** if **A** ***OR*** **B** = **1**, ***OR*** both).

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

NOT Gate





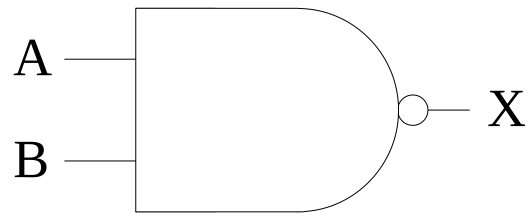
- The ***NOT*** gate outputs a **1** when the input is **0**

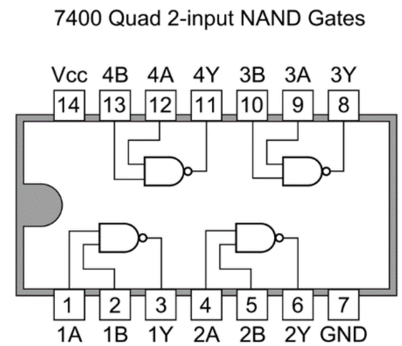
When the input is a **1** the ***NOT*** gate outputs a **0**

A	X
0	1
1	0

- The **NOT** gate is also referred to as an ***inverter***, as it inverts its input

NAND Gate





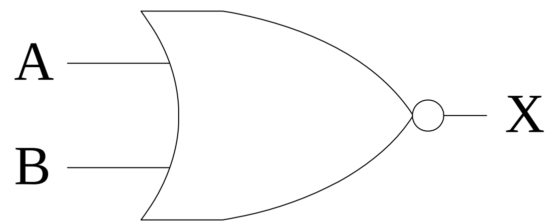
The **NAND** gate behaves like an **AND** gate whose output is then passed through a **NOT** gate

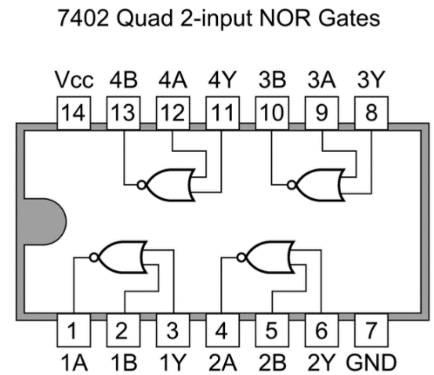
Thus **NOT AND** is shortened to give **NAND**

Therefore the truth table is the *opposite of an AND gate*

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

NOR Gate





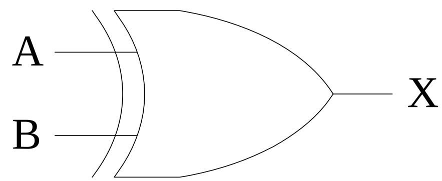
The **NOR** gate behaves like an **OR** gate whose output is then passed through a **NOT** gate

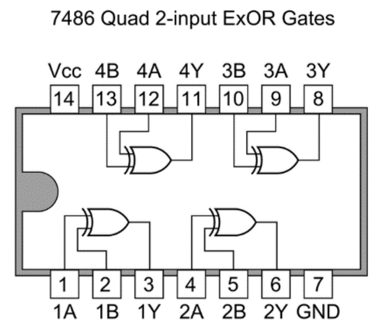
Thus **NOT OR** is shortened to give **NOR**

Therefore the truth table is the *opposite* of an OR gate

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

XOR Gate





The **XOR** gate (**exclusive OR**) outputs a **1** when ***only one* of its inputs is a 1**

(**X = 1** if **A OR B = 1**, ***but not both***)

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

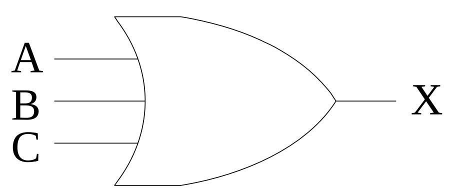
1.2 Multiple Input Gates

- Each logic gate can have only one output, but any number of inputs

(with the exception of the ***NOT*** gate which can only have **one input**)

- The number of rows in the truth table depends on the number of inputs to the gate

number of rows = 2^n (where n is the number of inputs.)



Thus, a three input ***OR*** gate would have a truth table with 2^3 (eight) rows

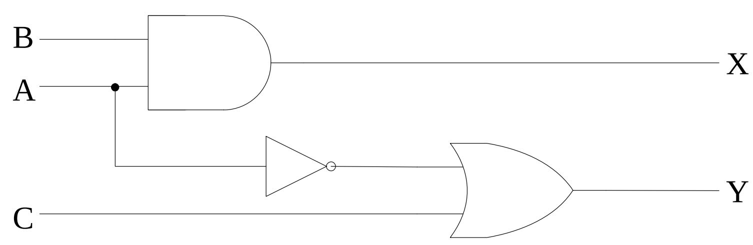
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

1.3 Simple Circuits

Truth tables

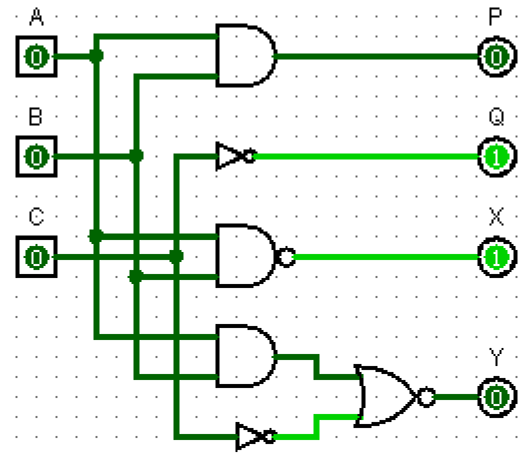
allow us to ascertain the behaviour of circuits with

multiple inputs and ***multiple outputs***

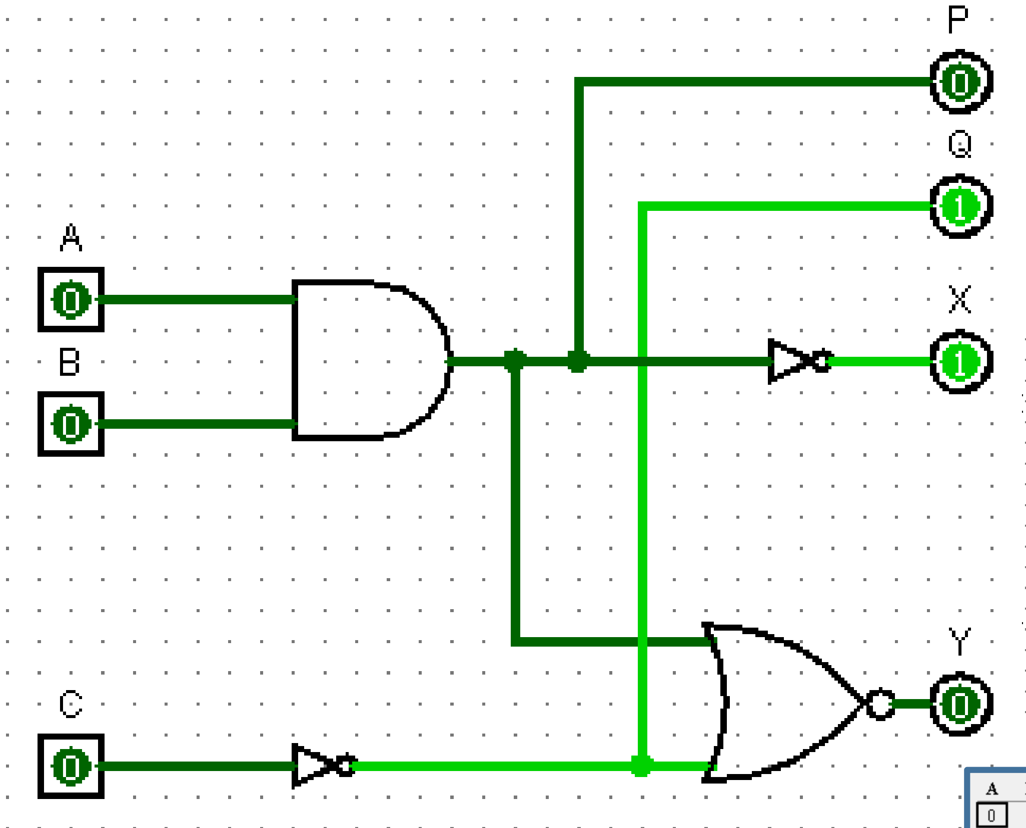


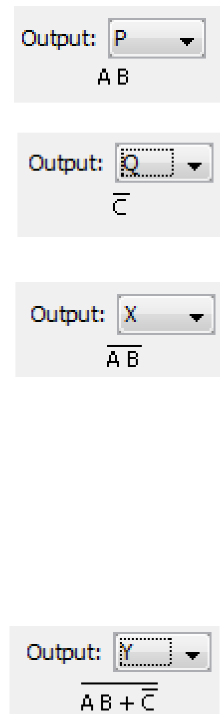
A	B	C	X	Y
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

1.4 Intermediate Results



- With more complex circuits ***intermediate results*** can be used to help construct the final truth table





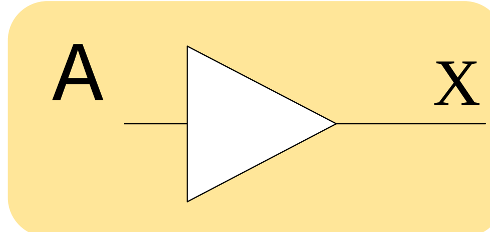
A	B	C	P	Q	X	Y
0	0	0	0	1	1	0
0	0	1	0	0	1	1
0	1	0	0	1	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	0
1	0	1	0	0	1	1
1	1	0	0	1	1	0
1	1	1	1	0	0	0

A	B	C	P	Q	X	Y
0	0	0	0	1	1	0
0	0	1	0	0	1	1
0	1	0	0	1	1	0
0	1	1	0	0	1	1
1	0	0	0	1	1	0
1	0	1	0	0	1	1
1	1	0	1	1	0	0
1	1	1	1	0	0	0

1.5 Buffer & Tri-State Gate outputs

1.5.1 Buffer

The gate to the right looks like an inverter,



- but it is not,
  ... because it *does not have a small circle at its output*

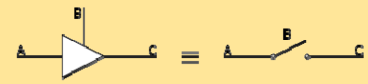
Such a gate is called a ***buffer*** because it *copies* the logic signal at its input to its output

(therefore the *buffer gate does not change the state of the data passing through it*, unlike other gates)

1.5.2 Tri-State output

A gate with a ***tri-state*** output has the special property that the output of the gate can be : -

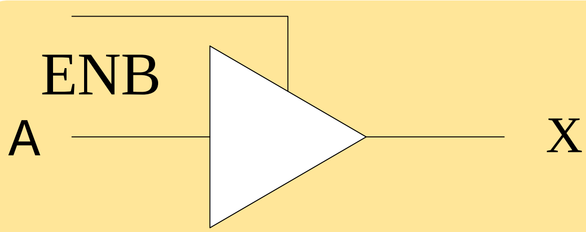
0, 1, or a ***meaningless*** *(disconnected) (third)* ***state***

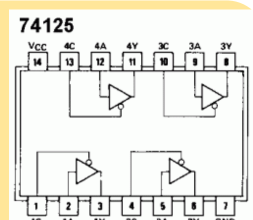


A ***tri-state*** gate can be ***any*** of the gates previously encountered

– it is not the gates logical function that is different, it is the behaviour of its output

The gate shown here is classed as a *non-inverting tri-state buffer*





ENABLE	A	Output	Description
0	0	X	Output meaningless & disconnected
0	1	X	Output meaningless & disconnected
1	0	0	Output same as Input
1	1	1	Output same as Input

All tri-state gates have a special ***ENABLE*** input

⇒ when ***ENABLE = 1***

the gate behaves normally and its output is either a *1* or a *0* depending on its input

⇒ when ***ENABLE = 0***

the *output is physically disconnected* from the gate's internal circuitry.

In this case we can say the output is meaningless

REFERENCES

Logisim

http://www.electronics-micros.com/software-hardware/logisim/