# Microprocessor Architectures (also memory arch)

Session 2.1

PH435

# Main things discussed here

von Neumann (Princeton) v/s Harvard Architecture[R.S.]

Address space

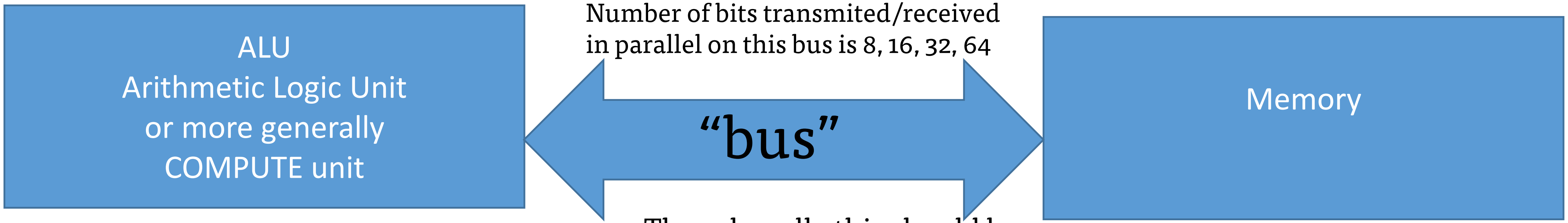
Examples of Microprocessors: architectures, instruction sets (CISC, RISC)
Intel, Motorola PowerPC
Microchip
ARM-Cortex Mx
Snapdragon (Qualcomm)

Limits of a “Turing” machine –
Clock speed limit (Dennard Scaling)
Neuromorphic computing

Quantum computing?

# Microprocessors

Big jump in digital computing comes by functionally separating the ‘compute’ and ‘memory’ functionality into two separate blocks



Number of bits transmited/received in parallel on this bus is 8, 16, 32, 64

Though really this should be called a “highway”: it has hardware wires (‘lanes’) on which multiple bits of a number are TX/RX with sync

Compare: FPGA is collection of gates state machines, memory ‘amorphously’ distributed throughout the system in registers

# John von Neumann (Princeton Univ)

Designed the ‘von Neumann architecture’

Architecture → layout of ALU, Memory and interconnections

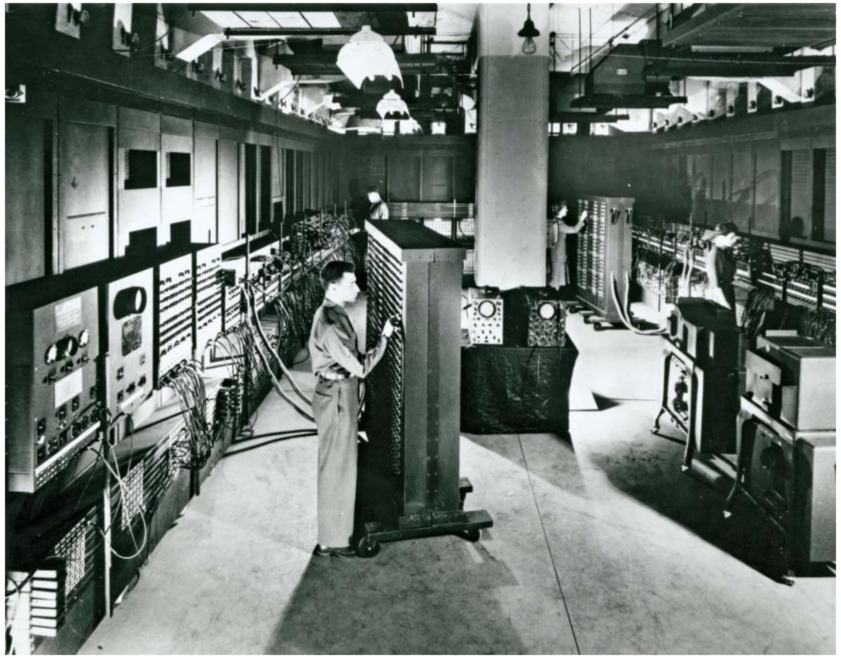
Made for a specific project: the first ‘Electronic Computer’ ENIAC ~ 1943 . Meant to solve trajectories of bombs

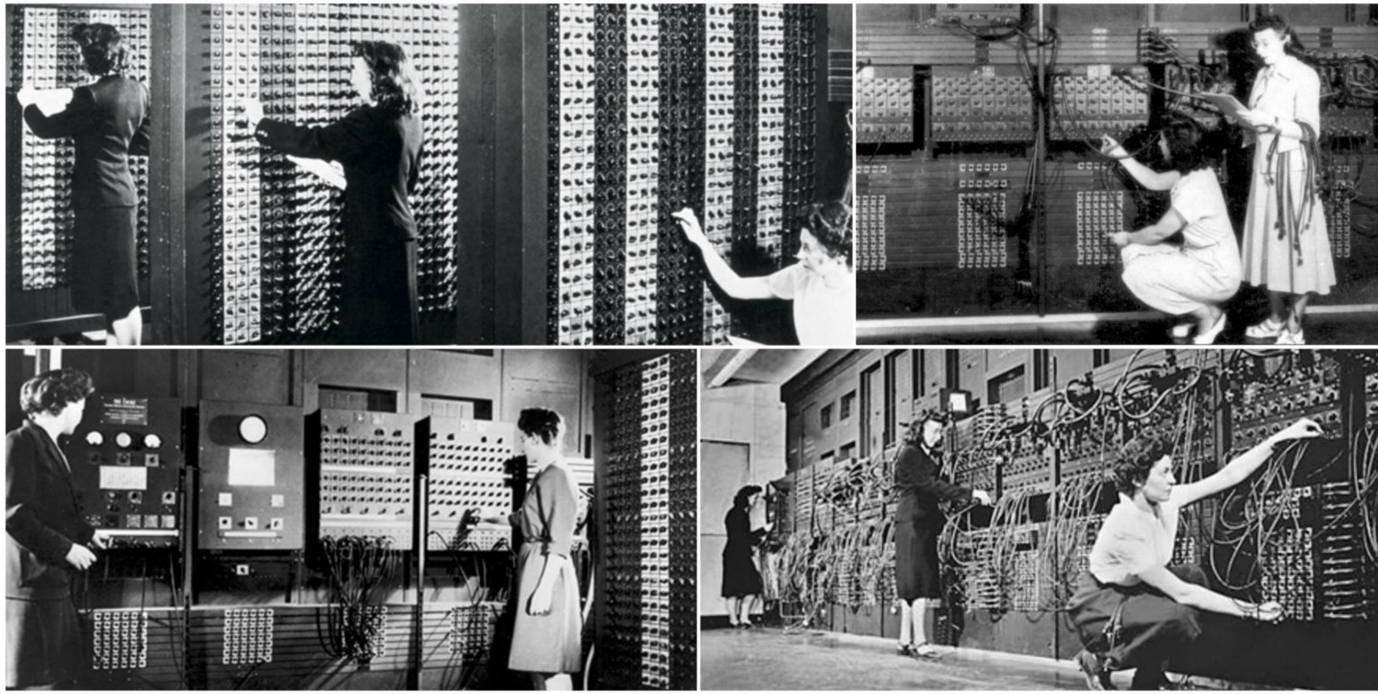
**E**lectronic **N**umerical **I**ntegrator and **C**omputer

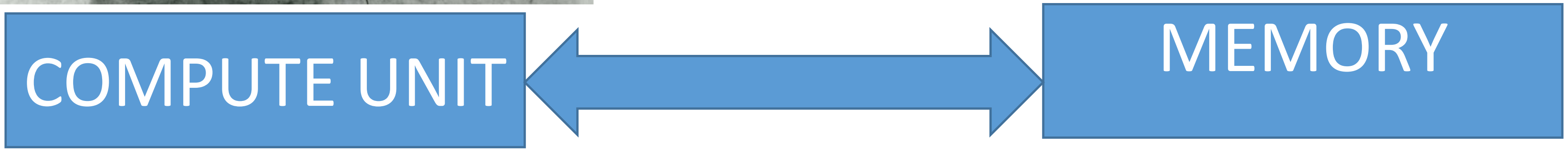
This computer occupied more space than our entire electronics lab (Room 130+132+134)

https://www.youtube.com/watch?v=bGk9W65vXNA

# ENIAC architecture







COMPUTE UNIT

MEMORY

These were in *physically* in different parts of the room!

Wires need to be connected by hand to read/write program code + data to the 'CPU'

Program and Data memory mixed & accessed on same 'bus'

# What did the ENIAC compute?

# What was revolutionary about ENIAC?

It included an instruction set that allowed you to run a program like:

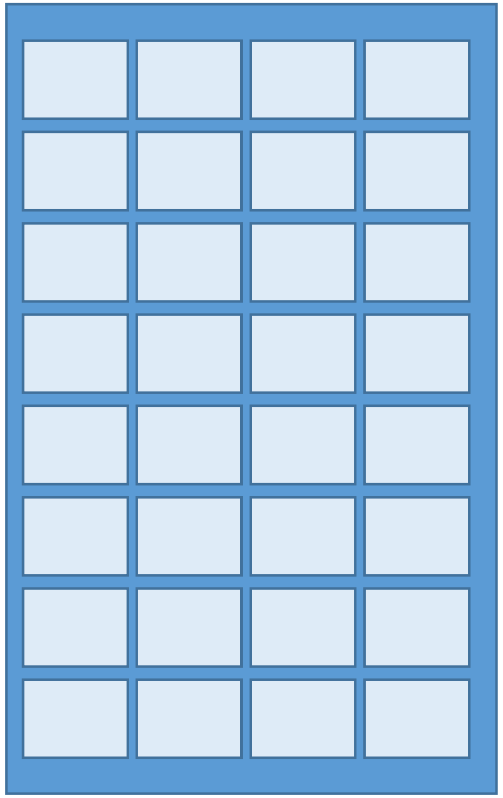
```
IF(x>5)THEN
GOTO LINE 23
```

This requires the CPU to know where in data memory the value of `x` is stored

&

Where in program memory `LINE 23` is to be accessed

# Address spaces – at what “address” in physical memory is data / program stored



□ This ‘memory’ IC has
4x8=32
D registers, each stores
a single bit

Put them in a logical
arrangement such that
‘address’ (aaa,bb) refers
to row aaa, column bb

Suppose we build this ‘memory IC’ on breadboard with 32 D registers..

# Memory is functionally and practically of different types

Functional defn

**RAM SRAM, DRAM**

Random Access Memory

Every object stored here has an address – you can pull up any object at random if you know its address

Functional defn

**Program Memory**

**ALU reads 'what to do' from here. Note: functionally program instructions (called 'opcodes') are also data!**

Functional defn

**Data Memory**

**ALU reads/writes data input and compute outputs here**

Memory

Practical defn

**ROM**

Read Only Memory

If you want to protect some data from being changed you put it here. Written once – then gate connections removed – stores values forever

Practical defn

**EPROM**

Eraseable Programmable Read Only Memory

Can be erased and re-written (i.e. gate connections can be re-arranged) - like an FPGA, but only functions as memory

Practical def

**DRAM**

**Dynamic RAM**

frequent read/write

Practical def

**Flash RAM**

**(also NV RAM)**

faster to r/w than EPROM