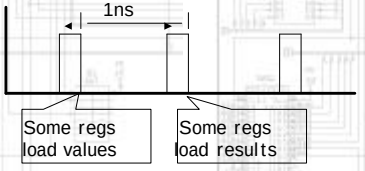
Fundamental Constrains

- We can transport information as:
  - Electromagnetic waves @:
    - C = 300000 km/sec in free space
    - C_{copper} = 0.99 C in copper
    - C_{gates} = 0.001 C passing through transistors (gates)
- For a given clock rate f this puts a limit on:
  - The physical separation of registers
    - The size of the system

Consider a Processor P

Clock rate f = 1GHz = 1000MHz = 10^9 c.p.s

Period $\tau = 1/f = 1/10^9 = 10^{-9}$ sec = 1ns

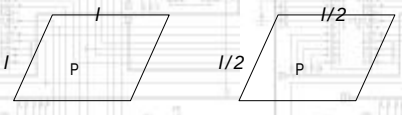


- Clearly, the maximum distance d between source and destination registers is:
  - $d = c \times \tau = 3 \times 10^{10} \times 10^{-9} = 30cm$

Heat and Speed

- This is easy to accomplish within one chip, but for chip-to-chip and pcb-to-pcb, e.g. memory to processor, it is difficult
- So, fast systems muct be very compact
- But every transistor generates heat which must be drawn off or the machine overheats
- To see how heat and speed interact, consider a processor P on area $A = l \times l$ running at clock rate f_0 and producing W watts of heat
- To speed it up, we need to half its size

Heat and Speed



Clock rate $f = f_0$

Heat intensity $H_0 = \frac{W}{A}$

Clock rate $f = 2 \times f_0$

Heat intensity $H = \frac{W}{A/4} = 4H_0$

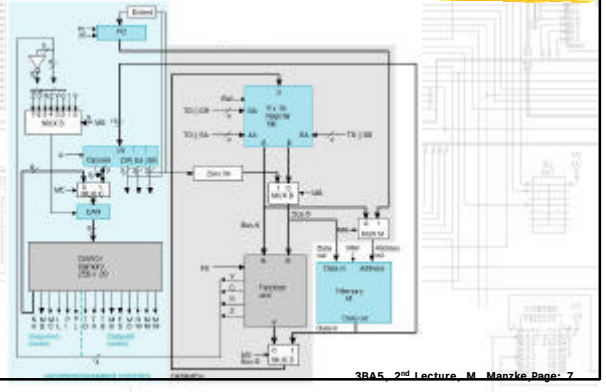
Conclusion

- There has to be an engineering compromise between *f* and *H*, and, after this has been reached, the only way to improve performance is to build larger processors and to ensure that more of their transistors are doing useful work in each clock period
  - i.e. **to use parallelism**
- This has resulted in generations of increasingly larger and more parallel architectures.

Taxonomy of Computer Architecture

- It is helpful to have a general framework in which to view the broad range of computer systems
- Flynn's taxonomy has proven to be the most useful though it is not the most sophisticated
- It divides the system into the following constituents:
  - MM: Main (primary) memory
  - CU: Control Unit
    - fetches the raw instruction code, decodes it and from this generates control signals for the PU
  - PU: Processor Unit
    - fetches operands, circulates them through the data path and stores the result

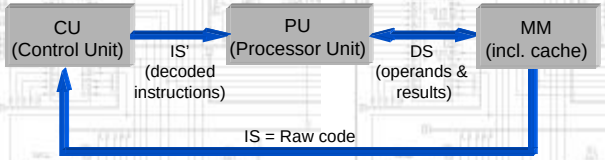
MM, PU and CU



Flynn's Taxonomy

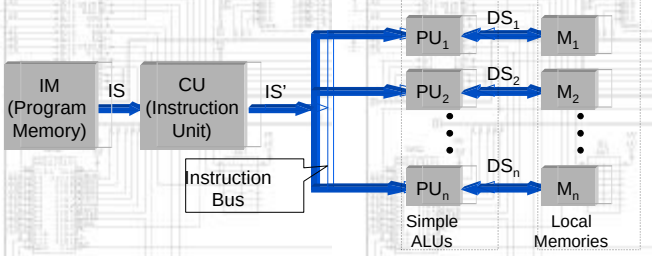
- Flynn focuses on the number of streams of:
  - Instructions (I)
  - Data (D)
    - which are 'in the same phase of execution at the most constrained component of the organisation'
- It is sufficient to distinguish single (S) from multiple (M) streams, leading to four distinct categories

SISD Single Instruction, Single Data (Von Newmann Architecture)



This schematic view emphasises the sepearate IS and DS, giving them separate pathways to MM. In practice, only a single pathway is built, leading to the so-called von Neumann bottleneck.

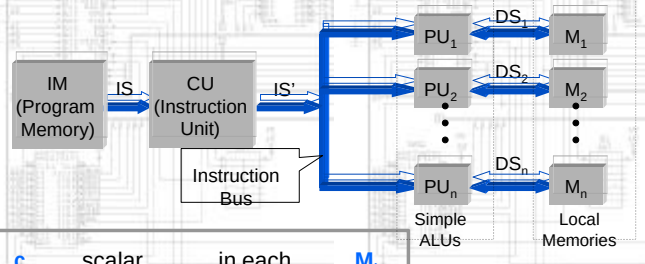
SIMD Single Instruction , Multiple Data (Array Processor)



A single instruction is applied to different data simultaneously – many separate PUs are invoked by a single CU.

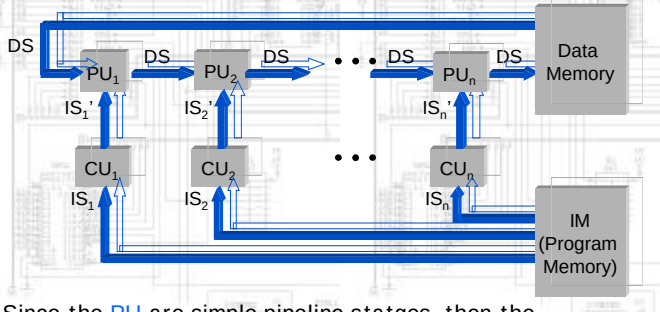
SIMD Example

$c \neg 7.3; Y \neg X + c$



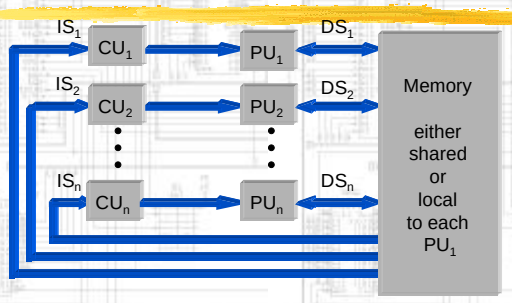
c	scalar	in each	M_i
X	vector	x_i in each	M_i
Y	vector	y_i in each	M_i

MISD Multiple Instruction, Single Data (Pipelined Vector Processor or Systolic Array)



Since the PU_i are simple pipeline statges, then the Instructions IS_i are just a couple of bits and change slowly.

MIMD Multiple Instruction , Multiple Data
(Multiprocessors, Multicomputers, Dataflow)



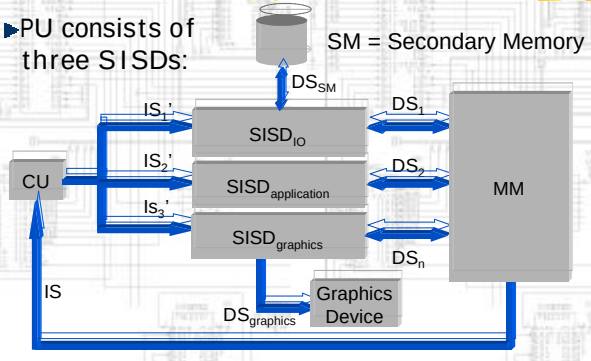
In MIMD structures in particular, interconnections networks (INs) are used to extend the address space of individual processors.

Implementations

- Note that the PU itself may be implemented by any member of this taxonomy
  - Leading to a hierarchical implementation in practice.
- Example:
  - Personal Computers (PCs) incorporate specialised SISDs for:
    - Application
    - I/O
    - Graphics

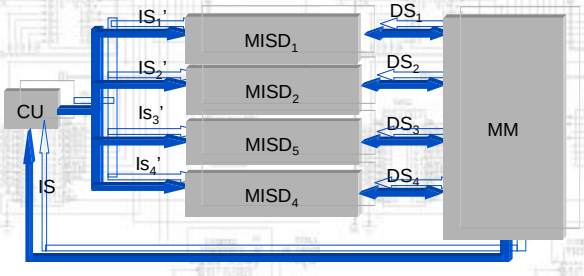
PC Example

- PU consists of three SISDs:



Vector Processor

- Vector processors analogously use multiple MISDs to implement their processor unit



Analogy – Car Manufacturing

Single Instruction

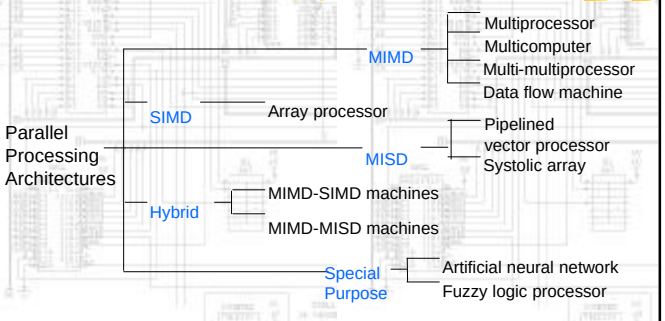
- SISD
  - One person builds a complete car, one at a time
- SIMD
  - n people build n cars, doing the same tasks concurrently (each person does all the tasks to build one complete car)

Analogy – Car Manufacturing

Multiple Instruction

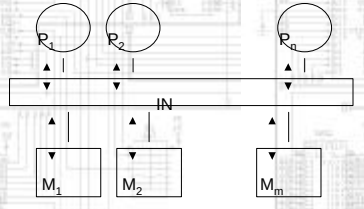
- MISD
  - A traditional assembly line – n people and each person always does the same task on the result of the previous person's work
- MIMD
  - Same as SIMD, except the people do not perform the same task at the same time. Instead, each person independently builds a complete car following their own set of instructions

Parallel Processing Architectures



MIMD Varieties

Multiprocessor-Shared Memory



MIMD Varieties

Multiprocessor - Passes Messages