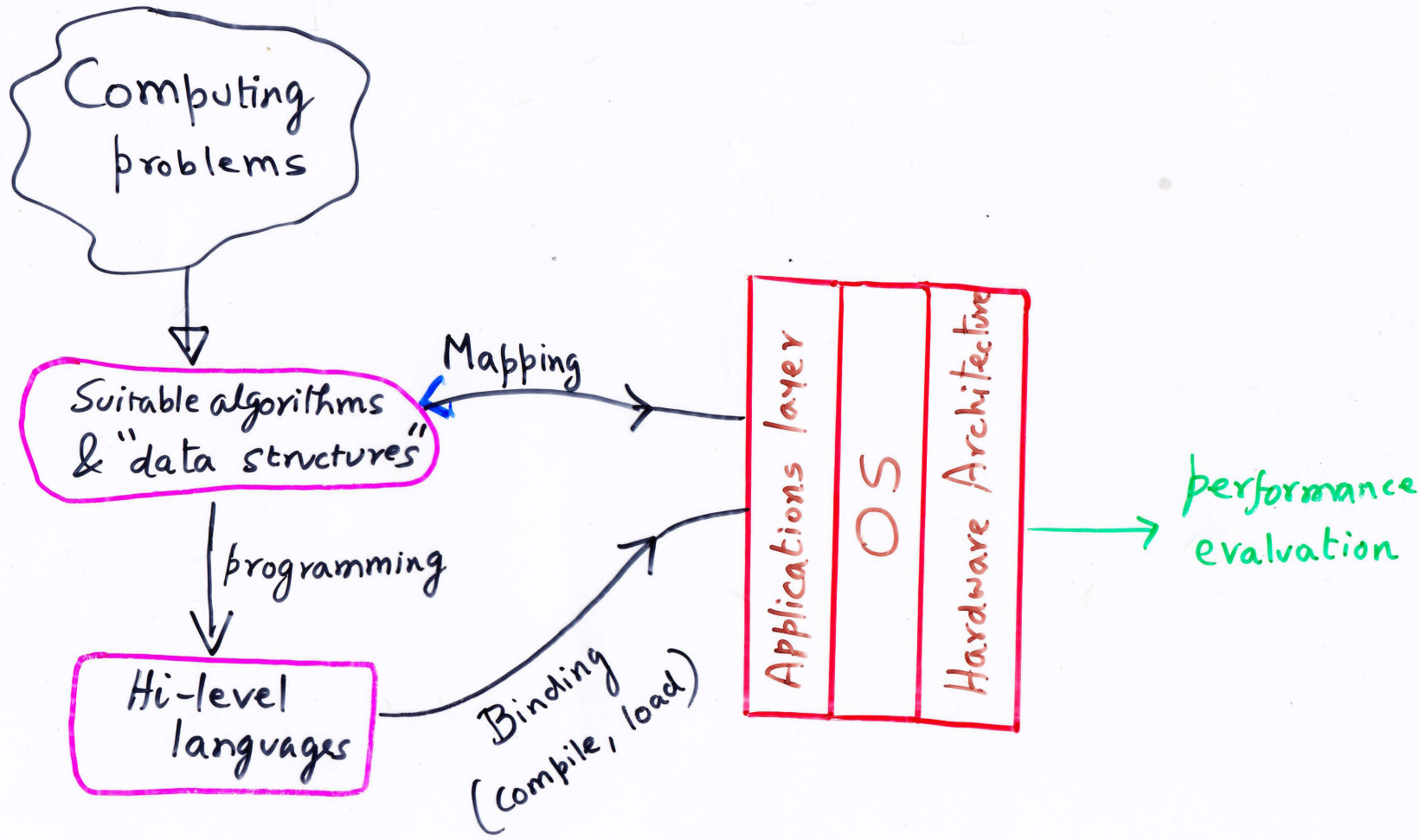


Fig 1.1. Elements of a modern computer system

OS

↳ mapping function

- bidirectional (algm struct ↔ h/w)

- mapping efficiency
  ↳ prog'ing

- mapping implies & includes
  - processor scheduling
  - memory maps
  - interproc. comm/.

} architecture dependent functions

Optimal mapping ?

Parallelism → any stage i.e., design, prog., compile, run

Compiler support → preprocessor
↳ precompiler
↳ parallelizing compiler

Table 1.1 Five Generations of Electronic Computers

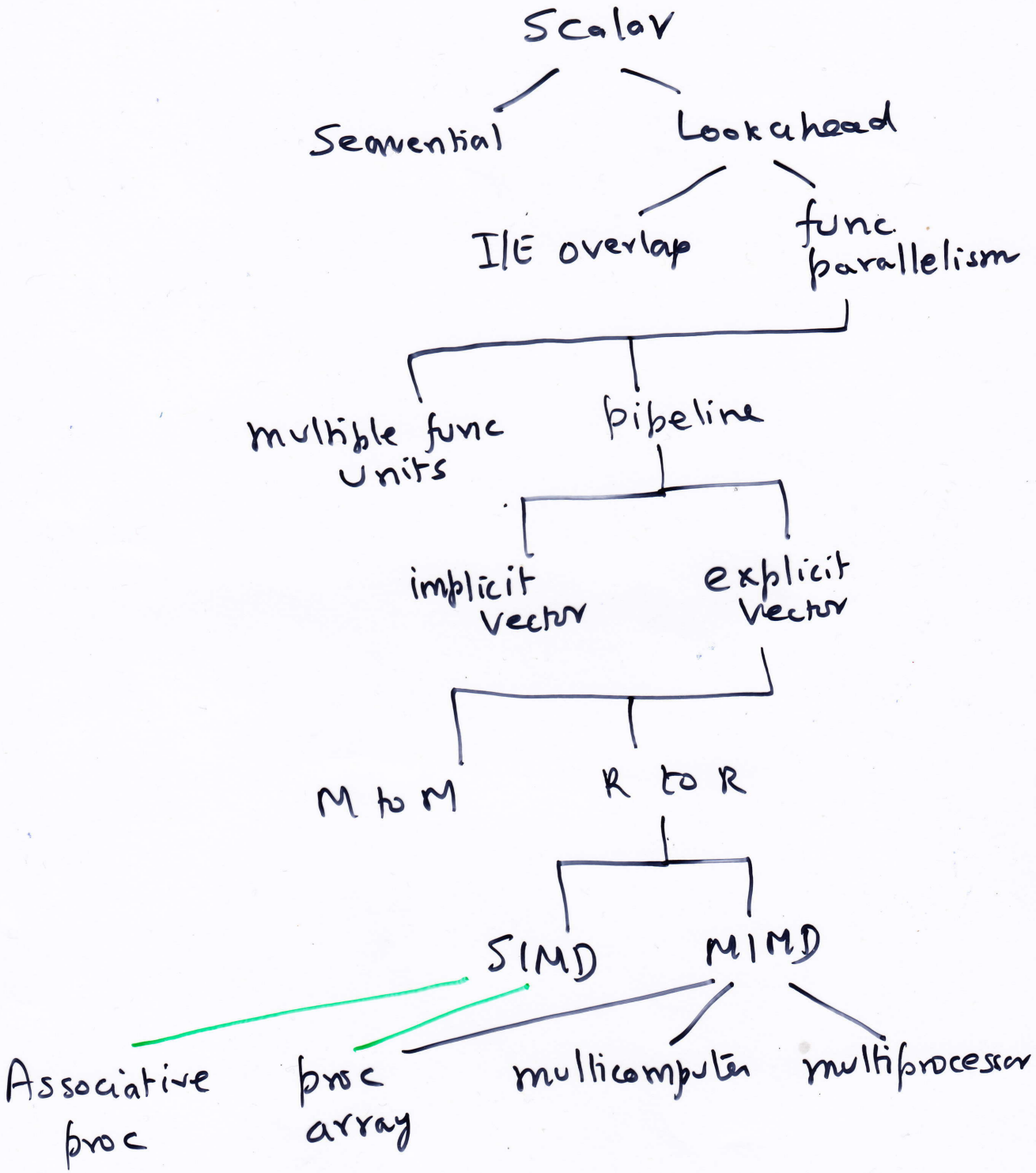
Generation	Technology and Architecture	Software and Applications	Representative Systems
First (1945–54)	Vacuum tubes and relay memories, CPU driven by PC and accumulator, fixed-point arithmetic.	Machine/assembly languages, single user, no subroutine linkage, programmed I/O using CPU.	ENIAC, Princeton IAS, IBM 701.
Second (1955–64)	Discrete transistors and core memories, floating-point arithmetic, I/O processors, multiplexed memory access.	HLL used with compilers, subroutine libraries, batch processing monitor.	IBM 7090, CDC 1604, Univac LARC.
Third (1965–74)	Integrated circuits (SSI/-MSI), microprogramming, pipelining, cache, and lookahead processors.	Multiprogramming and time-sharing OS, multiuser applications.	IBM 360/370, CDC 6600, TI-ASC, PDP-8.
Fourth (1975–90)	LSI/VLSI and semiconductor memory, multiprocessors, vector supercomputers, multicomputers.	Multiprocessor OS, languages, compilers, and environments for parallel processing.	VAX 9000, Cray X-MP, IBM 3090, BBN TC2000.
Fifth (1991–present)	ULSI/VHSIC processors, memory, and switches, high-density packaging, scalable architectures.	Massively parallel processing, grand challenge applications, heterogeneous processing.	Fujitsu VPP500, Cray/MPP, TMC/CM-5, Intel Paragon.

Table 1.2 Performance Factors Versus System Attributes

System Attributes	Performance Factors				
	Instr. Count, I_c	Average Cycles per Instruction, CPI			Processor Cycle Time, τ
		Processor Cycles per Instruction, p	Memory References per Instruction, m	Memory-Access Latency, k	
Instruction-set Architecture	X	X			
Compiler Technology	X	X	X		
Processor Implementation and Control		X			X
Cache and Memory Hierarchy				X	X

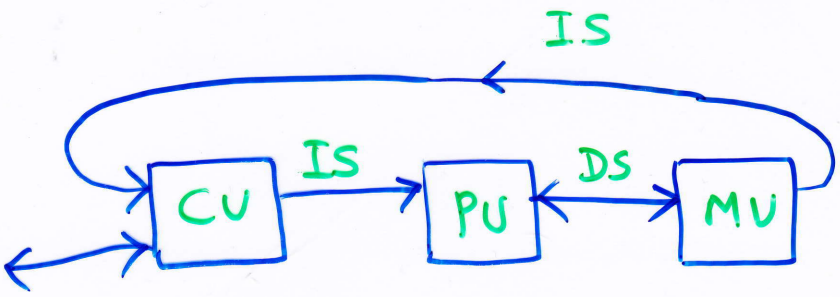
Evolution of Computer Architecture

(fig 1.2, pg: 10)

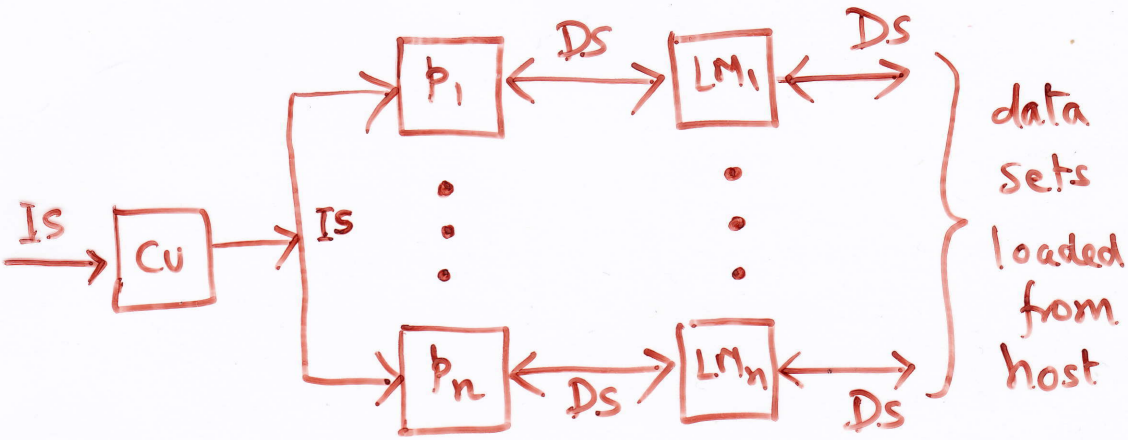


Flynn's Classification

SISD

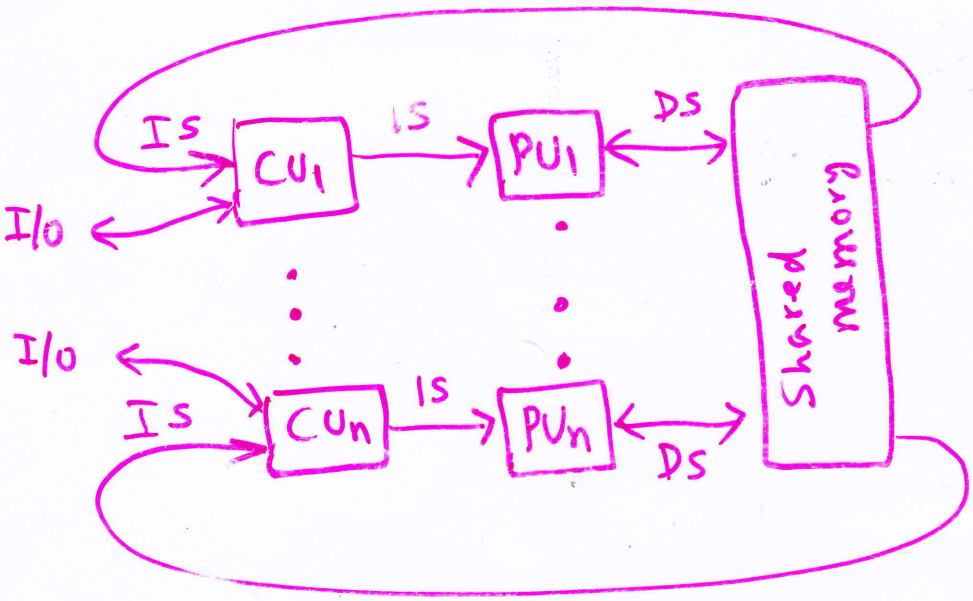


SIMD



(distributed memory)

MIMD



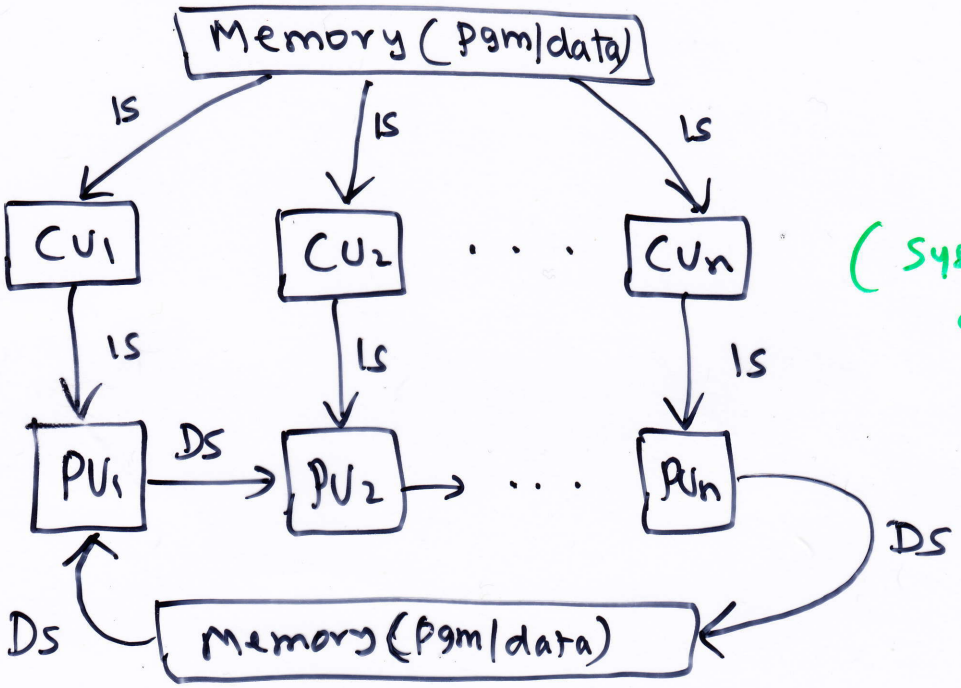
MIMD → general purpose computations

SIMD & MISD → special purpose

MISD → SIMD → MIMD

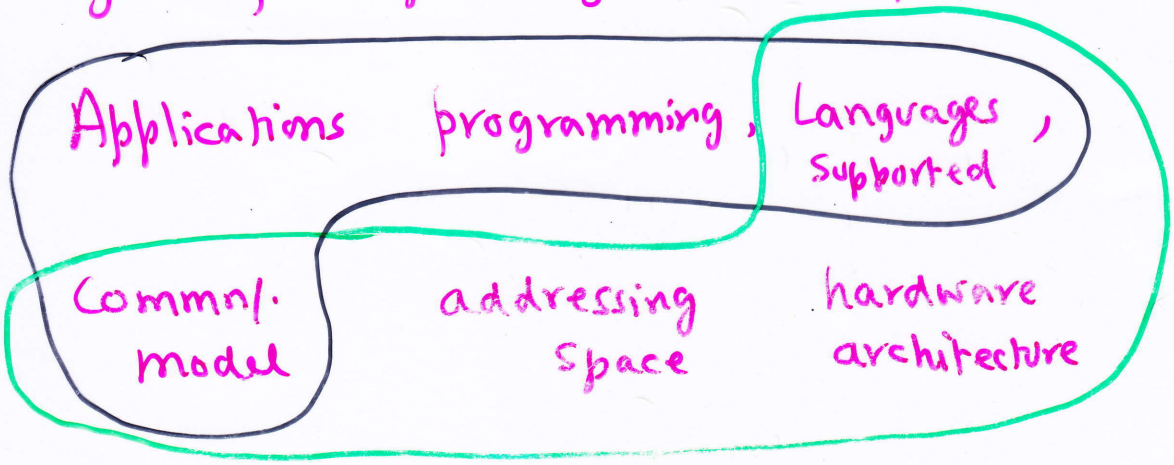
least popular → most popular

MISD:



(Systolic arrays)

Layers of computer system development



—— : machine independent

—— : machine dependent

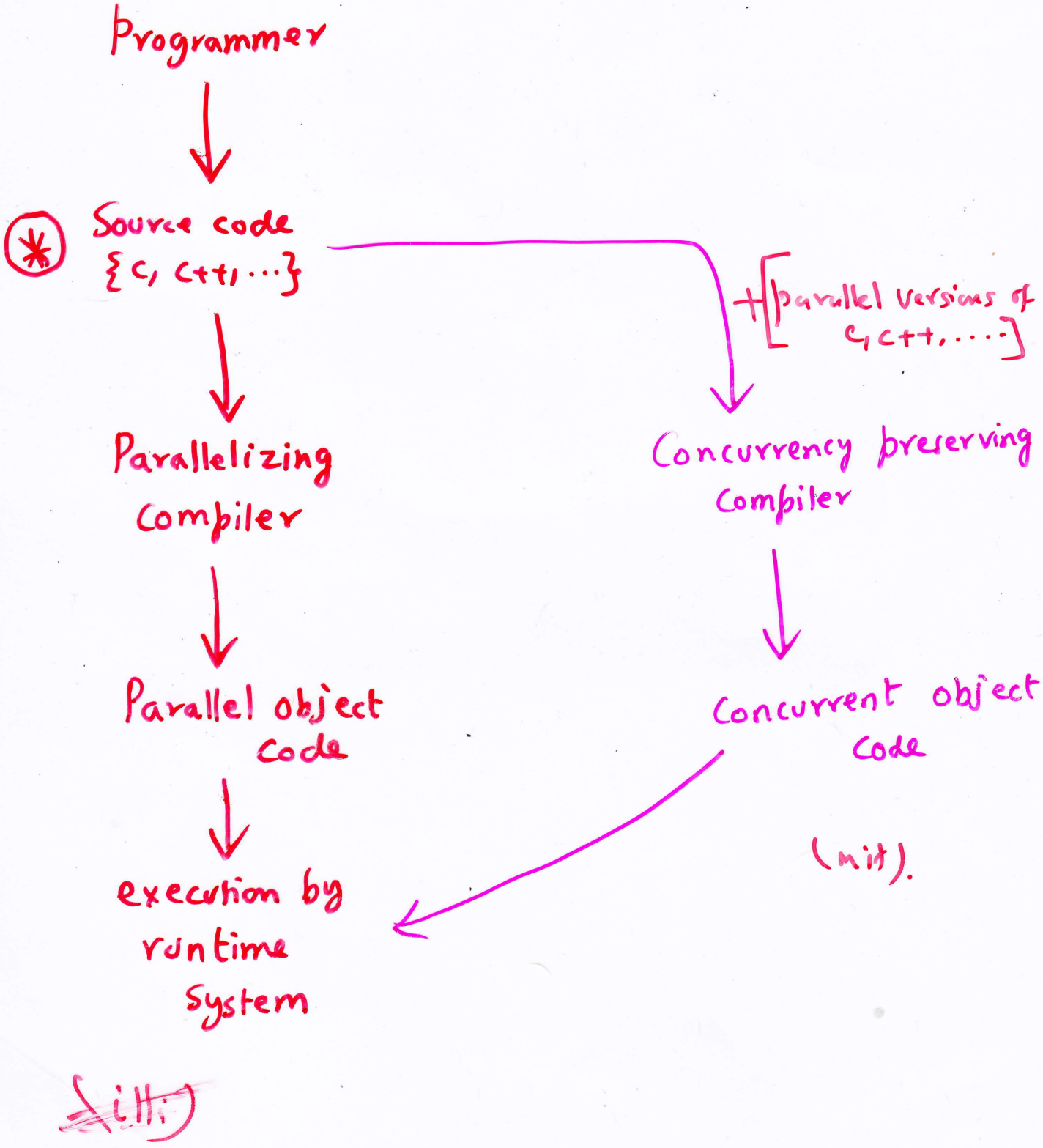
Table 1.1 Five Generations of Electronic Computers

Generation	Technology and Architecture	Software and Applications	Representative Systems
First (1945–54)	Vacuum tubes and relay memories, CPU driven by PC and accumulator, fixed-point arithmetic.	Machine/assembly languages, single user, no subroutine linkage, programmed I/O using CPU.	ENIAC, Princeton IAS, IBM 701.
Second (1955–64)	Discrete transistors and core memories, floating-point arithmetic, I/O processors, multiplexed memory access.	HLL used with compilers, subroutine libraries, batch processing monitor.	IBM 7090, CDC 1604, Univac LARC.
Third (1965–74)	Integrated circuits (SSI/-MSI), microprogramming, pipelining, cache, and lookahead processors.	Multiprogramming and time-sharing OS, multiuser applications.	IBM 360/370, CDC 6600, TI-ASC, PDP-8.
Fourth (1975–90)	LSI/VLSI and semiconductor memory, multiprocessors, vector supercomputers, multicomputers.	Multiprocessor OS, languages, compilers, and environments for parallel processing.	VAX 9000, Cray X-MP, IBM 3090, BBN TC2000.
Fifth (1991–present)	ULSI/VHSIC processors, memory, and switches, high-density packaging, scalable architectures.	Massively parallel processing, grand challenge applications, heterogeneous processing.	Fujitsu VPP500, Cray/MPP, TMC/CM-5, Intel Paragon.

Table 1.2 Performance Factors Versus System Attributes

System Attributes	Performance Factors				
	Instr. Count, I_c	Average Cycles per Instruction, CPI			Processor Cycle Time, τ
		Processor Cycles per Instruction, p	Memory References per Instruction, m	Memory-Access Latency, k	
Instruction-set Architecture	X	X			
Compiler Technology	X	X	X		
Processor Implementation and Control		X			X
Cache and Memory Hierarchy				X	X

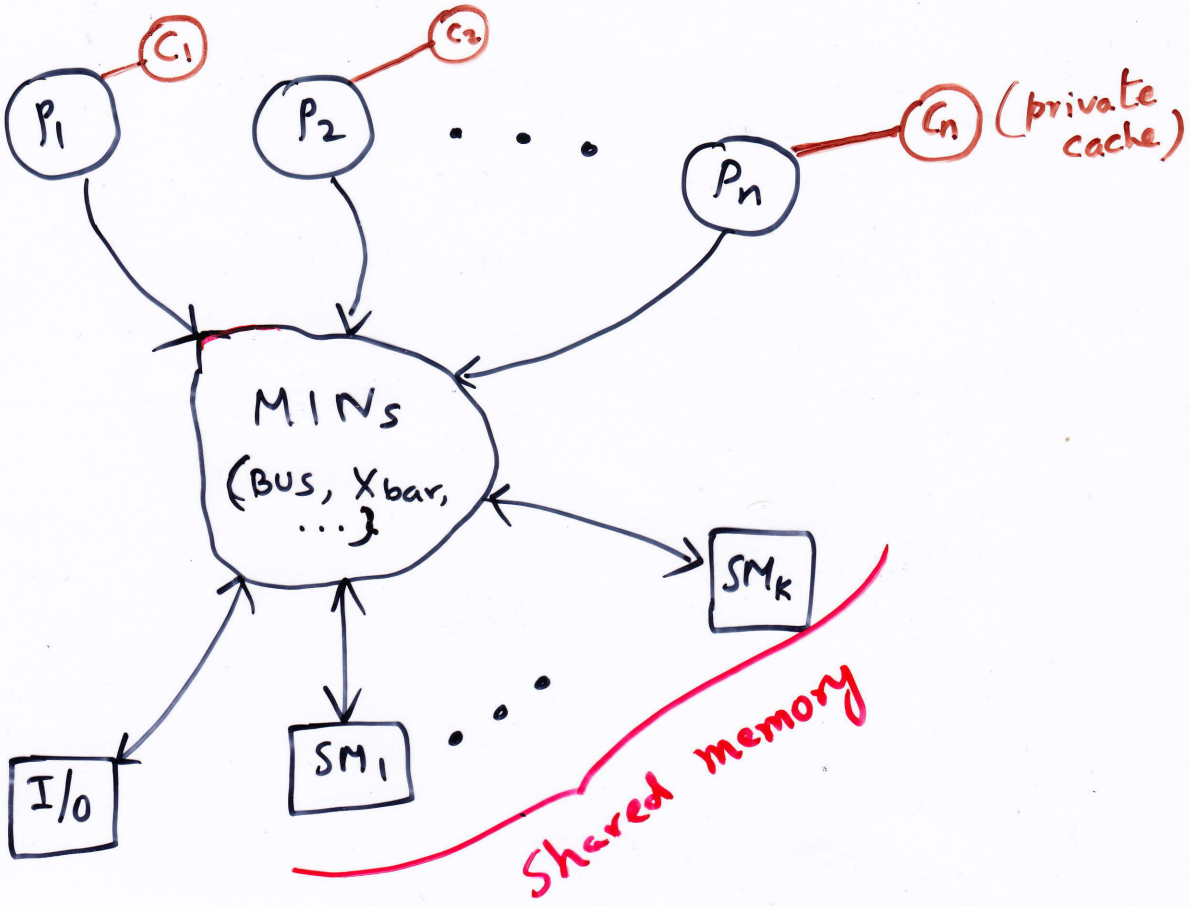
Two Approaches to parallel programming



(*) Implicit parallelism → Sequential languages

Explicit parallelism → Concurrent dialects of C, C++, Pascal....

UMA Multiprocessor Model



Example (1.2)

```
1.  DO 10 I=1,N
2.  A(I) = B(I)+C(I)
3.  10 CONTINUE
4.  SUM=0
5.  DO 20 J=1,N
6.      SUM = SUM +A(J)
7.  20 CONTINUE
```

(L = N/M)

```
Doall k=1,M
    Do 10 I=L(k-1)+1, KL
        A(I)= B(I)+C(I)
10  CONTINUE
    SUM(K) = 0
    DO 20 J=1,L
        SUM(K)= SUM(k)+
            A(L(k-1)+J)
20  CONTINUE
Endall
```

Sequential processing: 2N cycles → I : N, J : N

(L_2, L_4, L_6 → 1 m/c cycle to execute

L_1, L_3, L_5, L_7 → ignored for analysis)

Multiprocessing

- Divide the looping into M sections : $L = (N/M)$
- Doall declares that all M sections are executed by M processors in parallel

I loop : L cycles

J loop : M partial sums in L cycles

2L cycles are consumed to produce M partial sums. We need to merge these M to get the final sum.

$(N = 2^{20})$

- l-level binary adder : $\log_2 M = l$

each pair : k cycles ⇒ $(k+1)l$ cycles

$$\Rightarrow 2L + (k+1)l = \left(2\frac{N}{M} + (k+1)\log_2 M\right)$$

$$\text{Speed-up} = \frac{2N}{2 \cdot L + (k+1)\log_2 M}$$

where $L = (N/M)$

($k = 200$ cycles, $M = 256$)

$N = 2^{20} \Rightarrow$

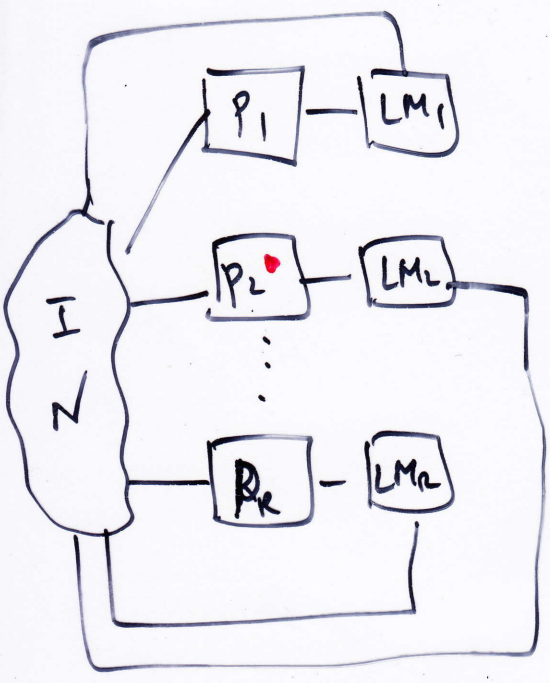
$$\text{Speed up} = \frac{2^{21}}{(2^{13} + 1608)}$$

$\simeq$ 83.6%.

NUMA MODEL

Shared mem system → access times vary with the location of a word.

(I)



- Collection of all local memories forms a global address space accessible by all the processors

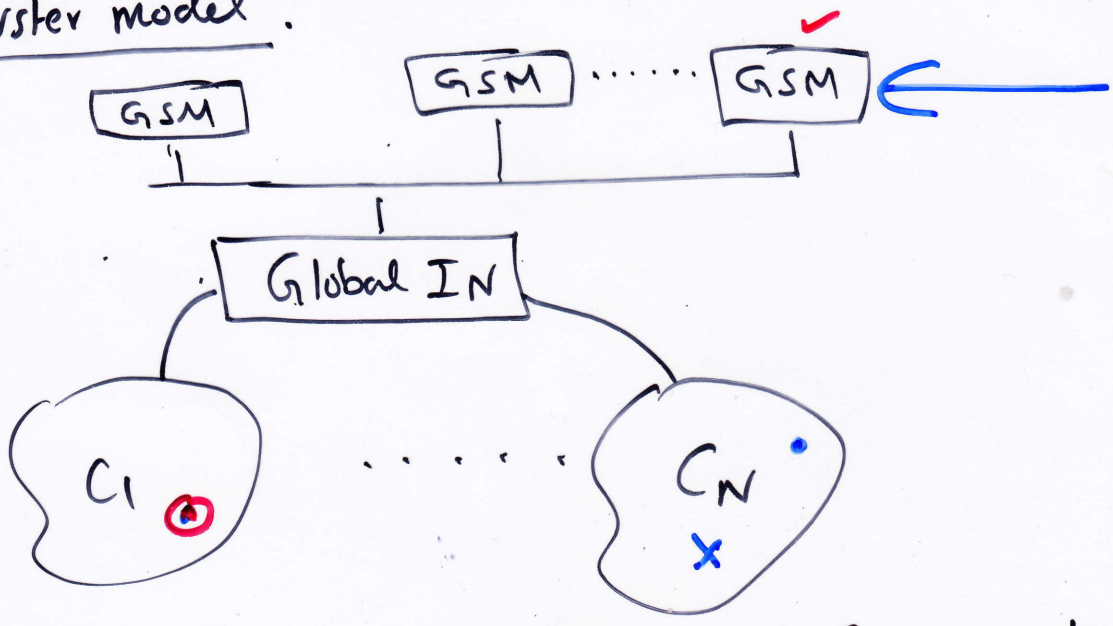
$t(LM) < t(GM)$ → true with the current day R.C architectures!

modern day → RCA

$t(LM) < t(GM) < t(\text{local disk})$

Shared local mem. model

(II) Cluster model:



Each C_i → Shared mem model or C_i → can be a UMA

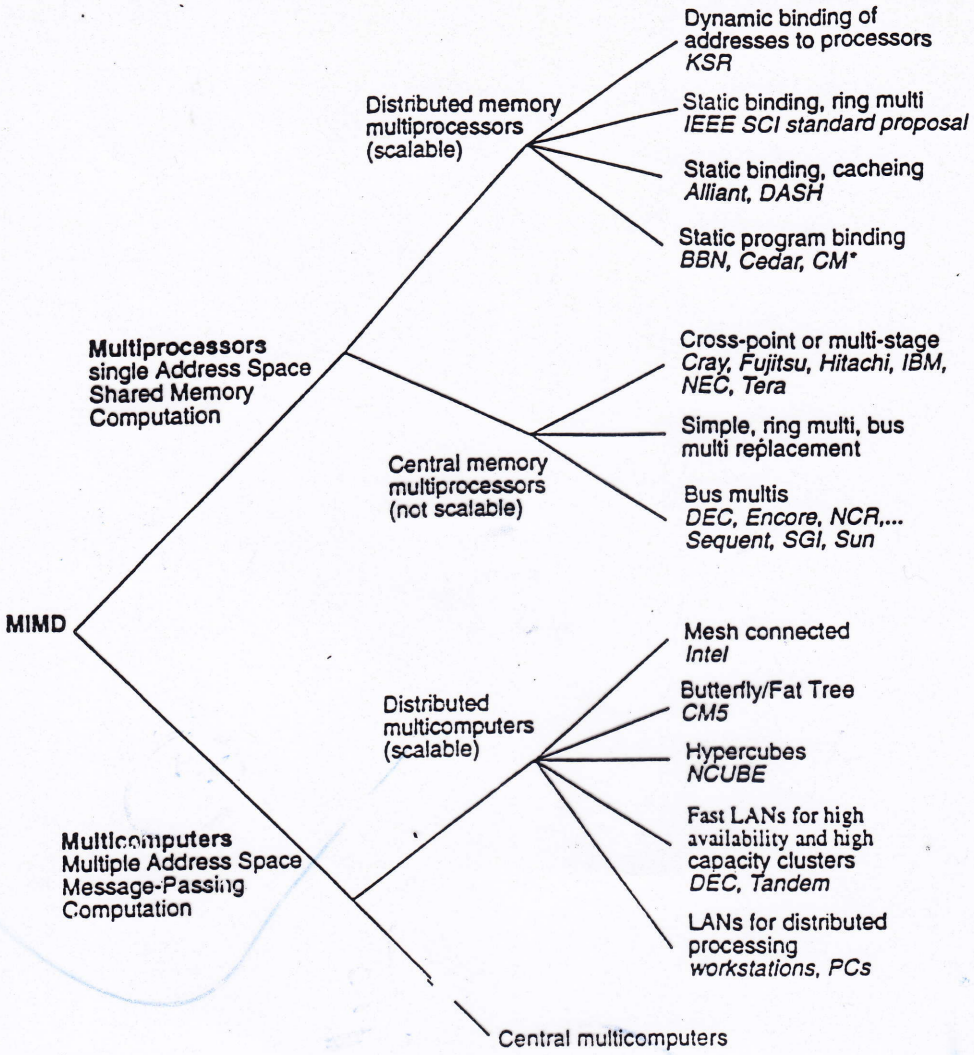


Figure 1.10 Bell's taxonomy of MIMD computers. (Courtesy of Gordon Bell; reprinted with permission from the *Communications of ACM*, August 1992)

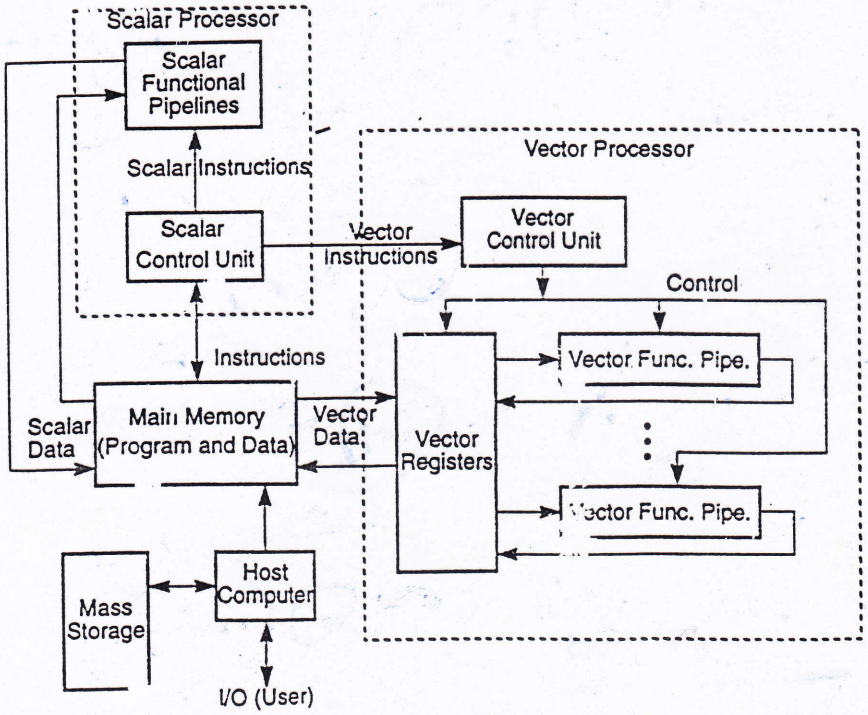
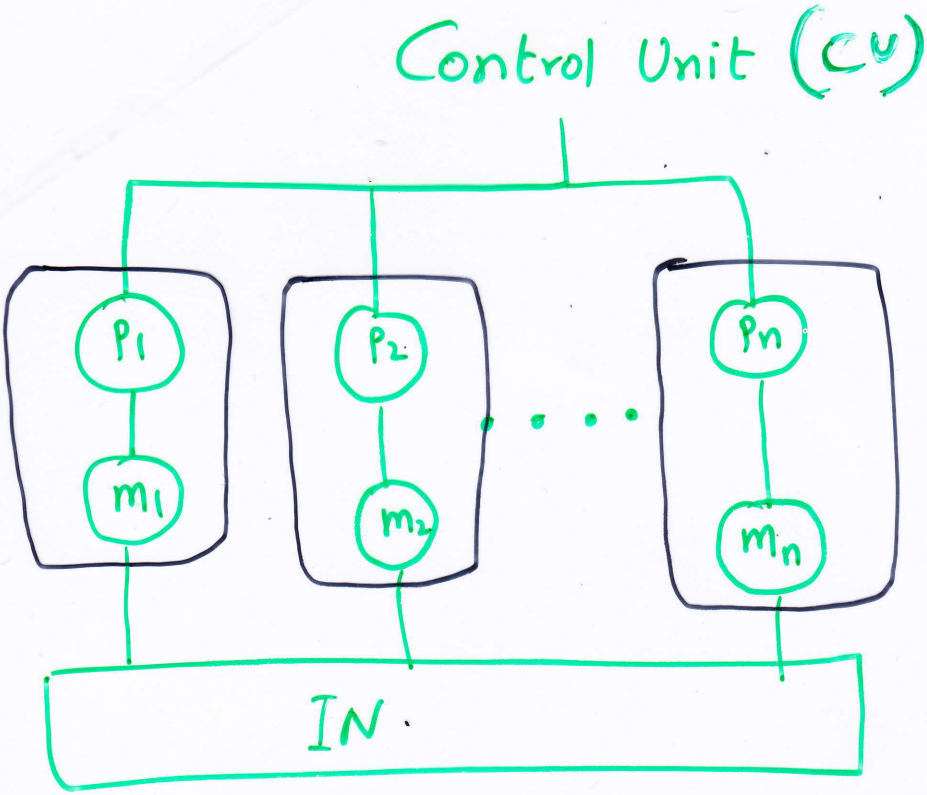


Figure 1.11 The architecture of a vector supercomputer.

SIMD Machine Model



$$M = \langle N, C, I, M, R \rangle$$

N : # of processing elements

C : Instr. directly executed by CU

I : Instr. broadcast by CU to all PEs.

M : Masking schemes

R : data-routing functions