Some Computer Organizations and Their Effectiveness

Michael J Flynn

IEEE Transactions on Computers. Vol. c-21, No.9, September 1972

Introduction

Attempts to codify a computer have been from three points of view

- Automata Theoritic or microscopic
- Individual Problem Oriented
- Global or Statistical

Microscopic View:

- Relationships are described exhaustively
- All parameters are considered irrespective of their importance in the problem

Individual Poblem Oriented:

- Compare Computer Organizations based on their relative performances in a peculiar environment
- Such comparisons are limited because of their ad hoc nature

Global or Statistical Comparisons:

- Made on basis of statistical tabulations of relative performances on various jobs or mixture of jobs

Drawback: The analysis is of little consequence in the system as the base premise on which they were based has been changed

Objective of this paper:

To reexamine the principal interactions within a processor system so as to generate a more "macroscopic" view, yet without reference to a particular environment

Limitations:

1. I/O problems are not treated as a limiting resource
2. No assessment of particular instruction set
3. In spite of the fact that either (1) or (2) will dominate the assessment we will use the notion of efficiency

Classification: Forms of Computing Systems

Stream : sequence of items (instructions or data)

Based on the magnitude of interactions of the instructions and data streams, machines are classified as

1. SISD - Single Instruction, Single Data Stream
2. SIMD - Single Instruction, Multiple Data Stream
3. MISD - Multiple Instruction, Single Data Stream
4. MIMD - Multiple Instruction, Multiple Data Stream

Let us formalize the stream view of Computing in order to obtain a better insight.

Consider a generalized system model consisting of a server and a requestor.

Requestor - a Program
Server – a Resource
In a stream of Requests, a Program at one stage can be a Resource to the Previous stage.

Let P be a Program – a request for service.
P specifies sub requests: R_1, R_2, R_3,..... R_n
Each sub request in turn has its own subrequest resembling a tree.

Any request R_i is bifurcated as R = $\{f_i^l, f_i^v\}$
f_i^l- logical role of a Requestor
f_i^v- physical role of a Server

Logical Role of Requestor:

- Define a result given an argument
- Deine precendence among tasks defined by P
- Transition between P and the next levels looked as logical role of R_i, while actual physical service is at the leaves of the tree
- Logical Role is defined by the term f_i^l

Service Role of Request:

- Service role is defined by f_i^v
- It defines the heirarchy of subrequests terminating in a primitive physical service resource.

In exploring the Multiple-Stream organizations, we explore about the two considerations,

1. Latency of Interstream Communications
2. Possibilities of high computational bandwidth within a stream.

Interstream Communications:

Two aspects of communication: In matrix representation,

1. *(O X S)* - Operational resource accessing Storage – useful for MIMD organizations
2. *(S X S)* – Storage accessing Storage – useful for SIMD organizations

There is also a *(O X O)* matrix which is useful for MISD organization.

Alternate format of the matrix is the connection matrix

An entry in the matrix is the reciprocal of normalized access time t_{ii} / t_{ij}

Zero entry when there is no communication

Stream Inertia:

- It's the concept of pipelining multiple instructions in a single instruction stream
- The number of instructions simultaneously processed is reffered to as “Confluence Factor” (J)
- Performance = $J / L \,.\, \Delta t$

  $L \,.\, \Delta t$ – Average execution time

System Classification:

In order to summarize, a system may be classified according to,

1. The number of instruction and data streams
2. Appropriate Communication or Connection matrices
3. Stream Inertia (J) and Instruction Execution Latency (L).

Effectiveness in Performing the Computing Process

Resolution of Entropy: (using Table Algorithm)
In a computing system, let N1 words of p bits each be input, N2 words of p bits each be output.

If each of input bit affects each of the output bits, the entropy of each output bit is N1 * p.

There are 2^{N1*p} combinations and N2 * p bits for each entry. Hence the total size is N2 * p * 2^{N1*p}

As a minimum the output bits could be independent of the input bits in which cases, the entrophy Q is just N2 * p

Thus, (p * N2) <= Q <= N2 * p * 2^{N1*p}

SISD and Stream Inertia

In SISD organization there arises situations such as
1. Direct Composition – an instruction requires an unavailable data
2. Indirect Composition - an address calculation is incomplete

Anticipation Factor (N):
When instruction “i” determines a data and “i+1” instruction requires that data, then there is a latency of (L-1) * Δt is required.

If we consider the serial latency time as L * Δt, then we consider Anticipation Factor N as the number of instructions between the one determining the data and the one using it.

For no turbulance,
N >= J / L

Delay due to Turbulance:

$$\text{delay} = \left(L - \frac{NL}{J}\right)\Delta t, \qquad \text{for } N < \frac{J}{L}$$

$$= 0, \qquad \text{for } N \geq \frac{J}{L}$$

If there are M instructions with a probabilty of turbulance causing instruction p,

The total time to execute these instructions would be,

$$T = \left[\frac{L}{J}\left[M(1-p)\right] + 1 + pM\left(L - \frac{NL}{J}\right)\right]\Delta t.$$

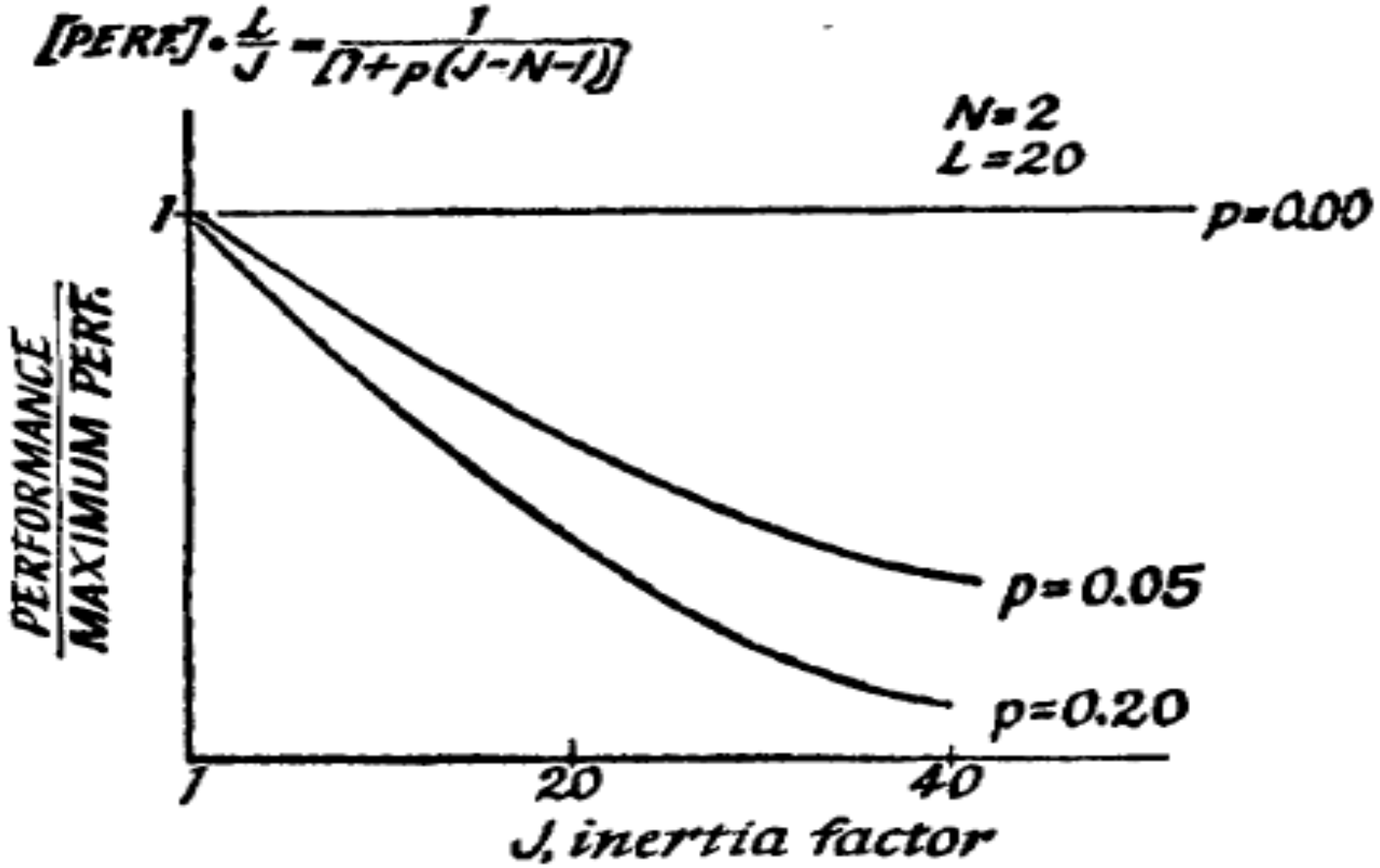
$$\text{perf.} = \frac{M}{T} = \frac{M}{\left[\frac{L}{J}[M(1-p)] + pM\left(L - \frac{NL}{J}\right) + 1\right]\Delta t}$$

$$= \frac{1}{\frac{L \cdot \Delta t}{J}\left[(1-p) + p(J-N) + \frac{J}{L \cdot M}\right]} .$$

The last term in the denominator drops out as M becomes large. Then

$$\text{perf.} = \frac{J}{L\Delta t} \cdot \frac{1}{[1 + p(J - N - 1)]} .$$

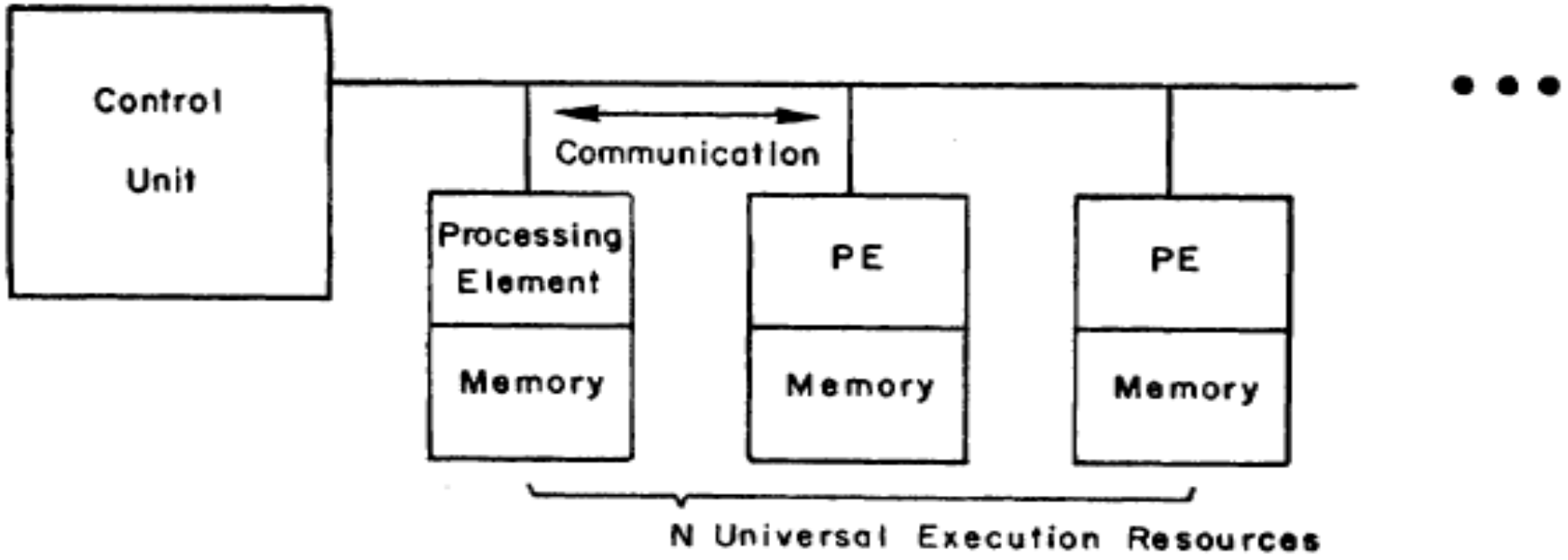
Stream Inertia and effects of branch:



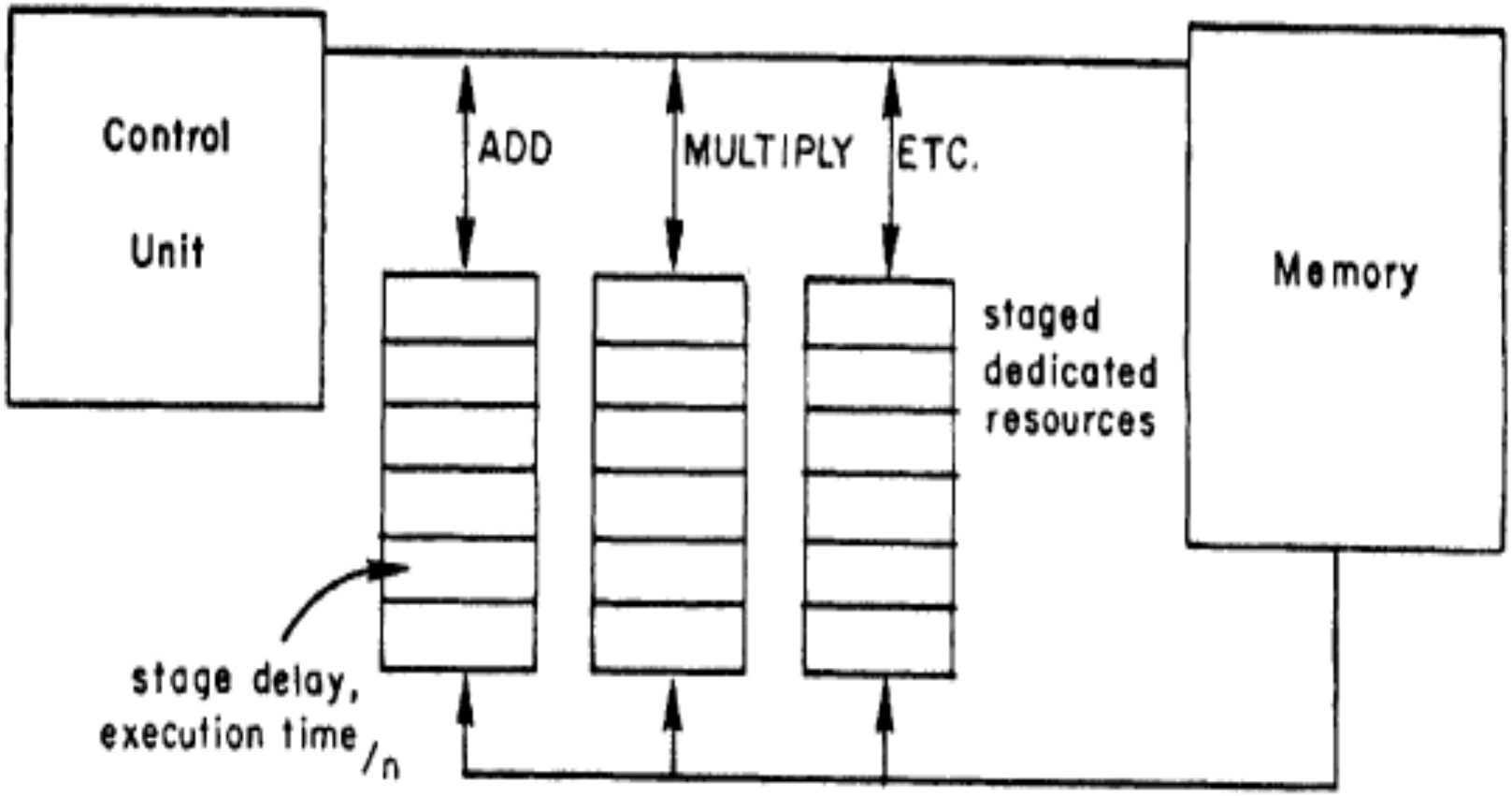
SIMD and its Effectiveness

SIMD is of three types:

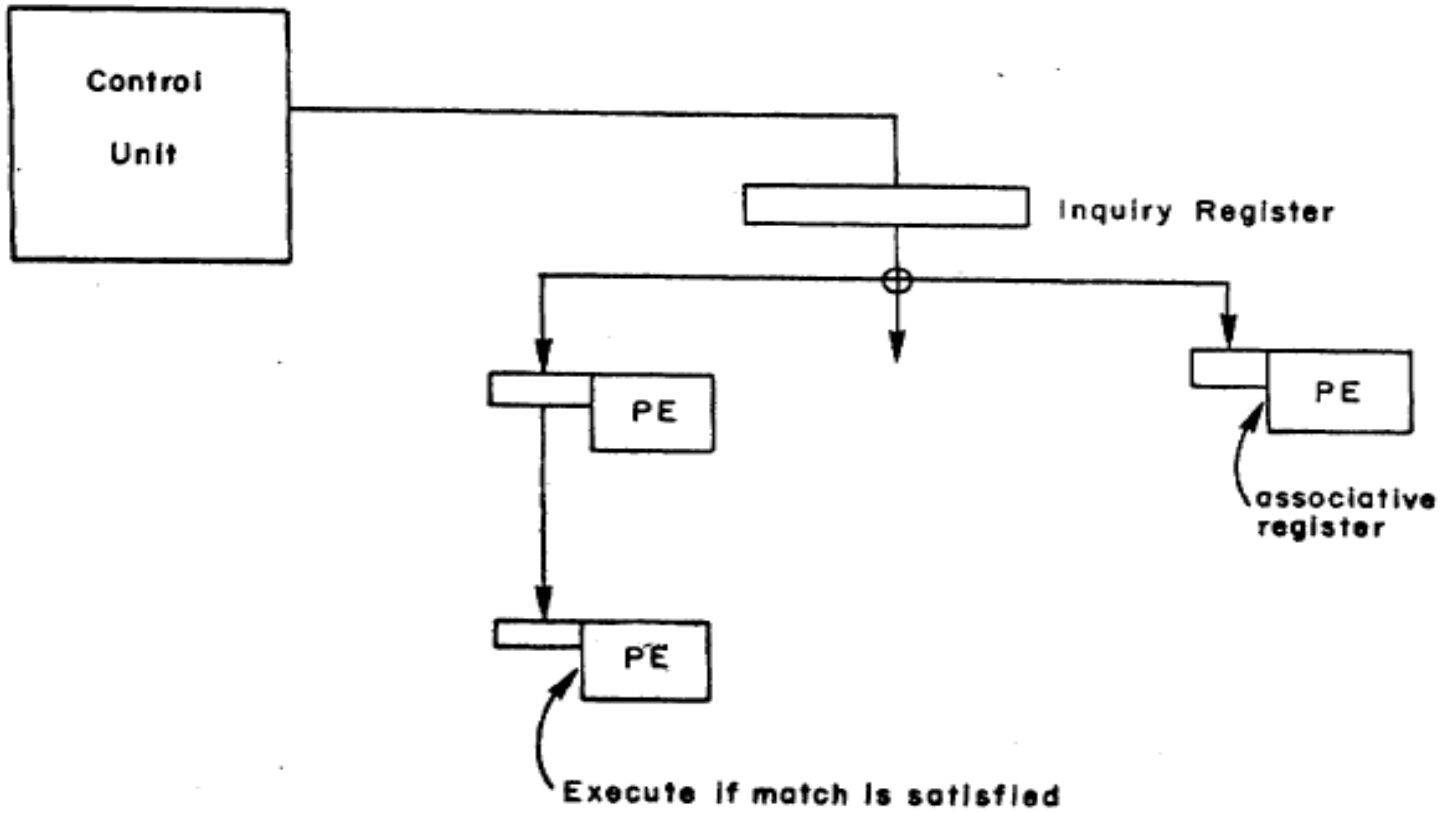
1. Array Processor – one control unit and 'm' directly connected processors – each element is independent.



2. Pipelined Processor – same as array processor but each processor has specific operation – each unit accepts a pair of operands every Δt time unit.



3. Associative Processor – Processing elements not directly addressed – a processing element is activated when a match relation is satisfied between a input register and characteristic data contained in each of the processing element.



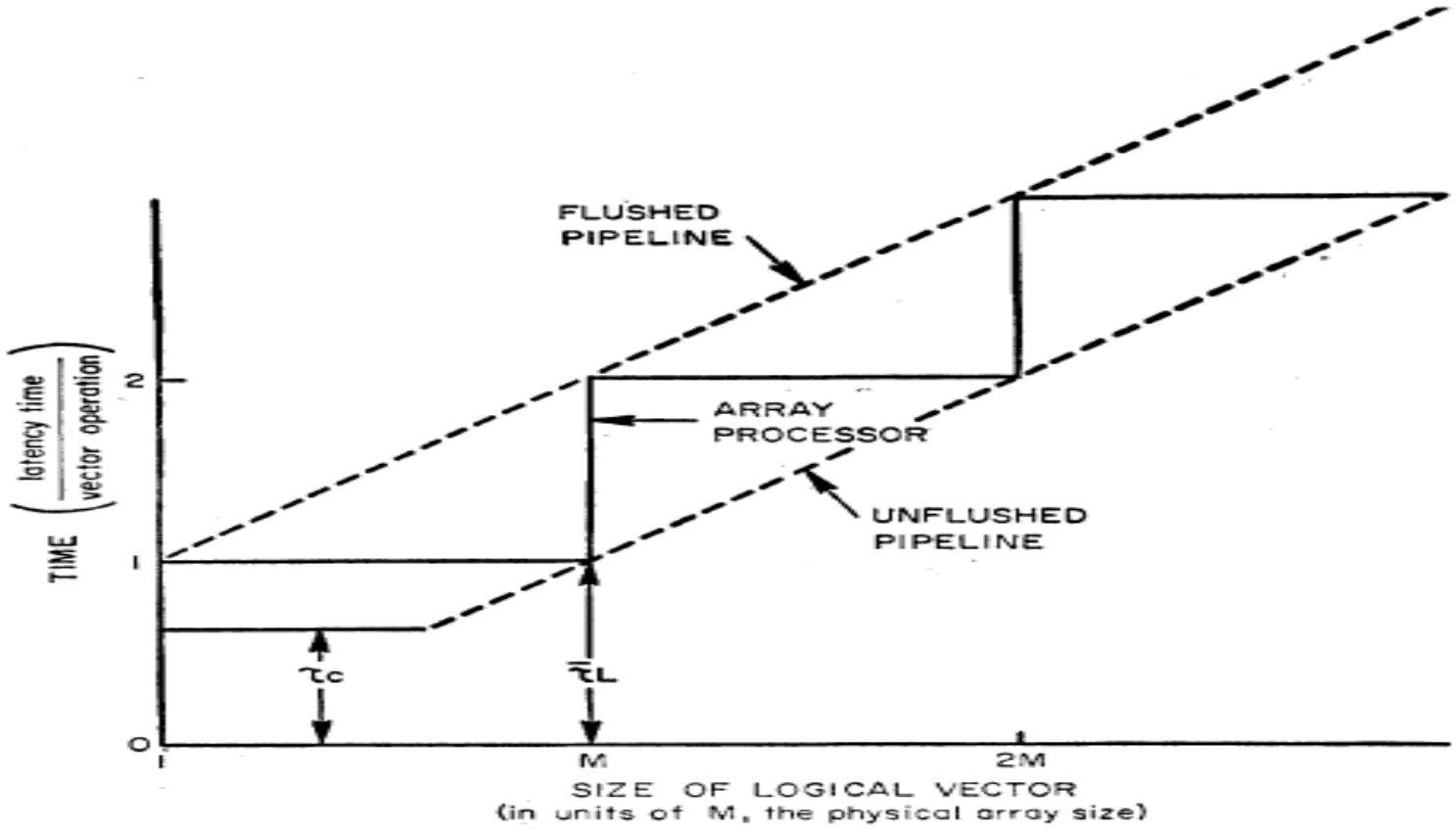
Difficulties of SIMD Organization:

1. Communications between processing elements
2. Vector Fitting: Matching of size between logical vector to be performed and the size of physical array which will process the vector.
3. Sequential Code: Housekeeping and Bookkeeping operations
4. Degradation due to Branching: When a branching occurs, the instructions that have already started execution will have to be flushed out if the branch is taken.
5. Performance proportional to $\log_2 m$ rather than linear due to the above factors.

Fitting Problem:

Given a source vector of size m, in an array processor of size M, the performance is degraded if m is not divided among M processors.

If m is large there is no significant degradation



If there are M data streams,
SISD instruction takes M.L.Δt time units
SIMD instruction takes L.Δt time units (ignoring the overlap)

In order to achieve this 1/M bound, the problem should be divided into M code segments.
We find that, the time to execute a block of N* instruction, where there is equal probability to take the branch or not,

$$T = L \cdot \sum_i N_{i,0} + L \cdot \sum_i N_{i,1} \cdot 2 + L \cdot \sum_i N_{i,2} \cdot 4 + \cdots + L \cdot \sum N_{i,j} \cdot 2^j$$

$$T = L \cdot \sum_j \sum_i N_{i,j} 2^j$$

Where,

"j" is the level of nesting of a branch and $N_{i,j}$ is the number of source instructions in the primary or branch path for ith branch at level j

$$\text{perf.} = \frac{I}{T} = \frac{N^*}{L \cdot \sum_j \sum_i N_{i,j} 2^j} .$$

If factor $\sum_i N_{i,j} / N^*$ is the probability of encountering a source instruction

$$P_j = \sum N_{i,j}/N^* \leq 1.$$

Minsky's Conjecture.

$$\text{perf.}_{\text{SIMD}} \approx \log_2 M$$

Since the performance is logarithmic and not linear, there is a performance degradation.
If we consider this degradation to be due to branching,

Let us find the P_j and performance when it is equally likely to be at level 1,2,3.... $\log_2 M$

Since no further degradation occurs beyond this level
$P_1 = P_2 = P_j = P_{[\log M] - 1}$

$$P_j = \frac{1}{[\log_2 M]} \Bigg|_{j=0}^{j=[\log_2 M]-1}$$

$$\text{perf.} = \frac{1}{L} \cdot \frac{1}{\sum_j p_j 2^j} .$$

Relative Performance with SISD overlapped processor,

$$\text{perf. relative} = \frac{M}{\sum_j P_j 2^j}$$

Thus SIMD organization is M times faster,

$$\text{perf. relative} = \frac{M}{\sum_j \frac{2^j}{[\log_2 M]}}$$

Ignoring the effect of non integral values of $\log_2 M$

$$\text{perf. relative} = \frac{M}{\sum_j \frac{2^j}{[\log_2 M]}}$$

for large M,

$$\text{perf. relative} \approx \frac{\log_2 M}{2}$$

If we assume that,

$$P_j = 2^{-j}$$

Then,

$$\text{perf. relative} \approx \frac{M}{\log_2 M}$$

Referring to the Vector Fitting Problem, we can interpret the number of data streams when comparing a SIMD processor to a pipelined processor.

Pipelined Processors categories:
There are two main types of Pipelined Processors,
1. ***Flushed:*** Until a vector operation is completed, the control unit does not issue the next operation

2. ***Unflushed:*** Next vector is issued as soon as the last element of the current operation is issued.

MIMD and Its Effectiveness:

It is of at least two types.

1. True Multiprocessors: Many independent SI processors share storage at some level for cooperative execution of a multitask program
2. Shared Resource Multiprocessor: Skeleton processors are arranged to share system resource.

Problems in MIMD:

1. Communication / Composition overhead
2. Cost increases linearly with additional processors, while performance increases at a lower rate.
3. Providing a method for dynamic reconfiguration.

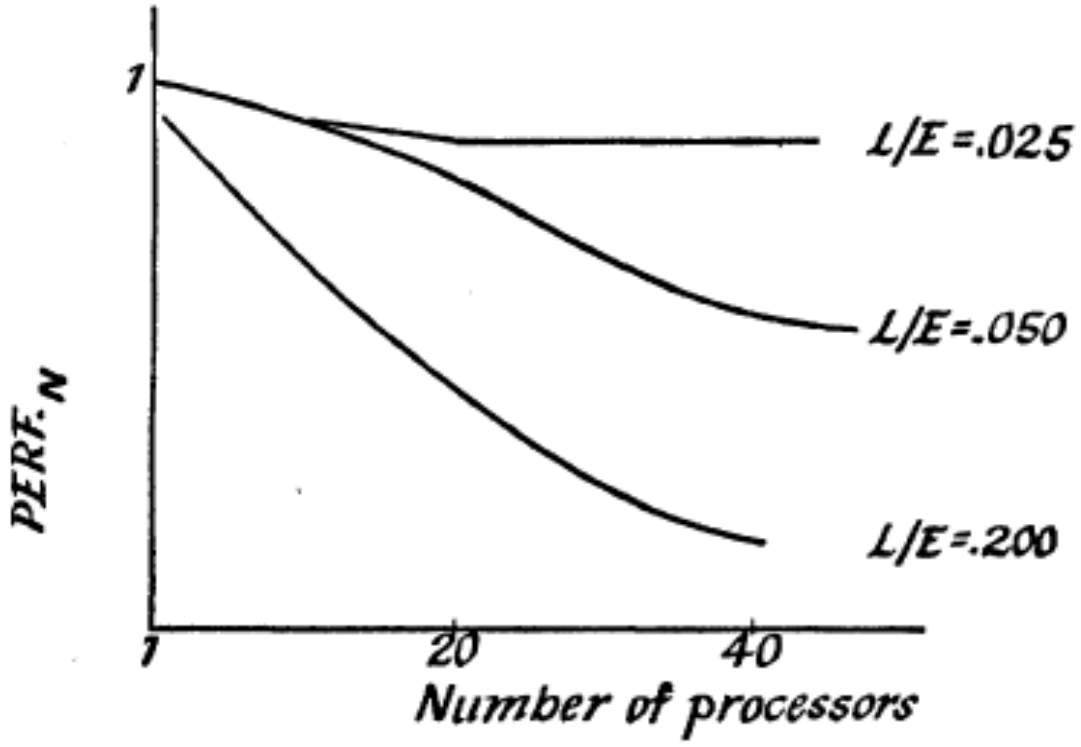
There are two other problems,
Precedence Problem and Lockout Problem

Software Lockout: In a MIMD environment,
A processor cannot access data due to interstream communication problems. It is termed as software lockout.The lockout time for jth processor is,

$$L_j = \sum_i p_{ij} T_{ij}$$

T_{ij} – is the communication Time
p_{ij} – probability of task j accessing data stream i.

MIMD Lockout is shown as,



Expected number of Locked-out Processors for a given number of processors (n)

$$\mathcal{E}\,(\text{idle}) = \sum_{i=2}^{n} \frac{(i-1)}{\left(\frac{E}{L}\right)^{i}(n-i)!} \Bigg/ \sum_{i=0}^{n} \frac{1}{\left(\frac{E}{L}\right)^{i}(n-i)!}$$

If a single processor has unit performance then n processors

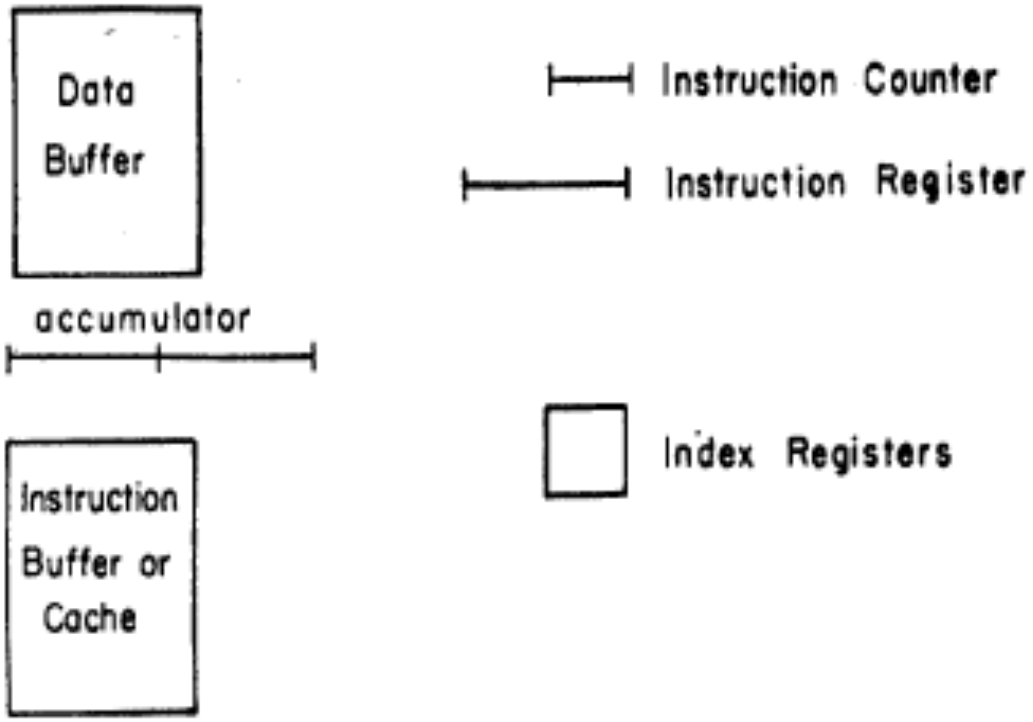
$$\text{perf.} = n - \mathcal{E}\,(\text{idle})$$

and normalized performance is given by,

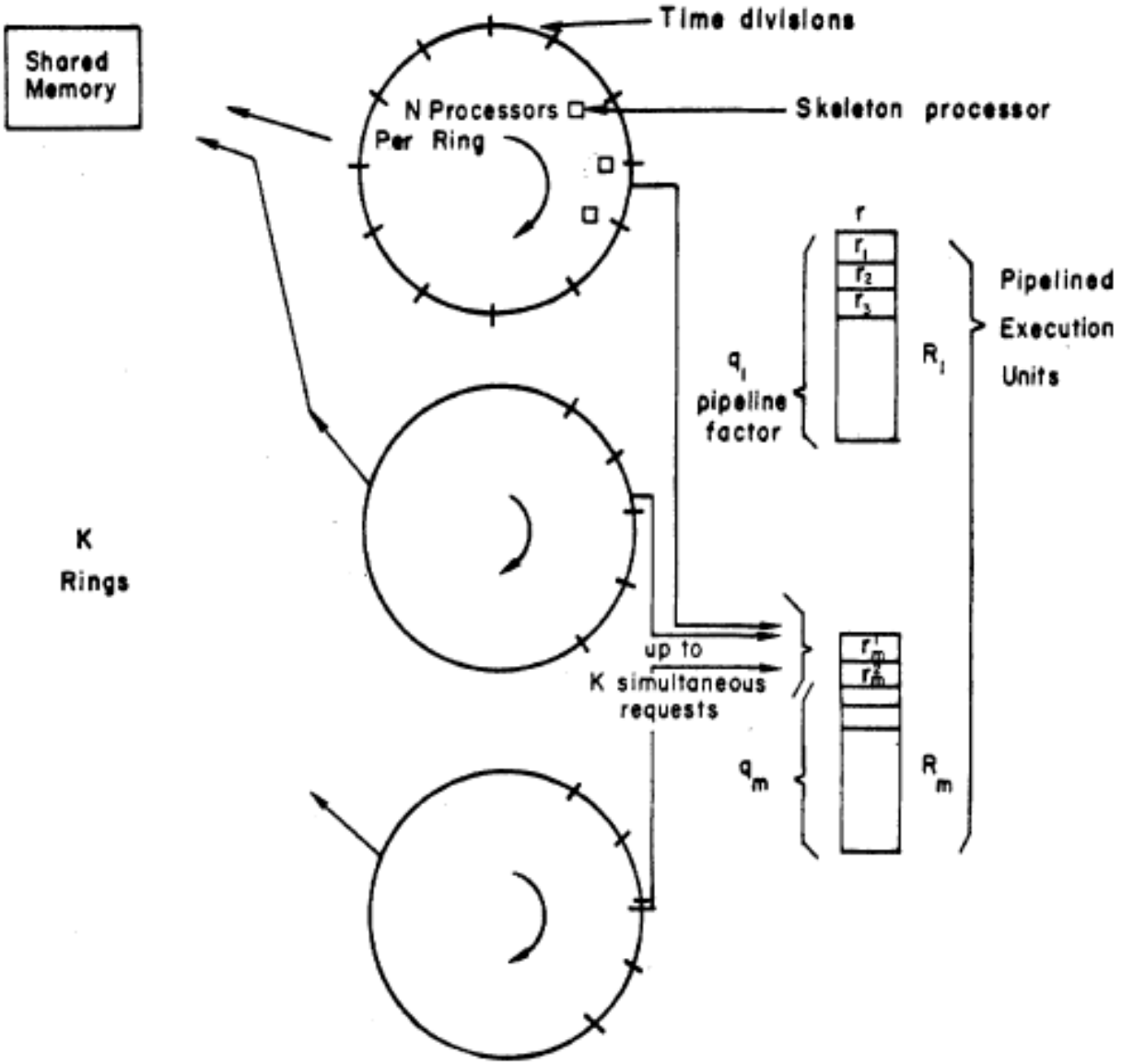
$$\text{perf.}_N = \frac{n - \mathcal{E}\,(\text{idle})}{n}$$

In order to use the componants of the SISD overlapped computer, a multiple skeleton processors sharing the execution resources is considered.

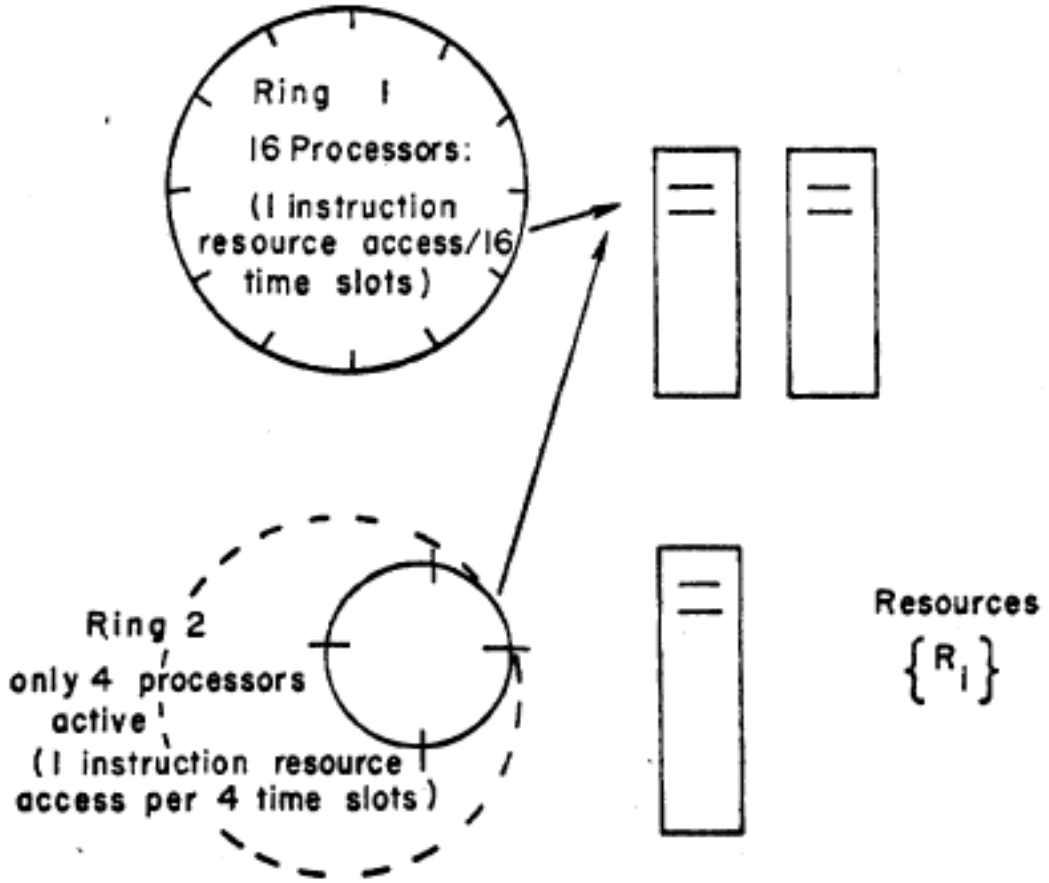
A single Skeleton processor has just enough resource to do load and store operations.



An optimal arrangement is space-time switching, which is shown as,



When the amount of parallelism is less than the number of processors available, a ring topology of the processor arrangement is followed which can be explained from the following diagram,



Some Considerations on System Resources:

Execution Resources:

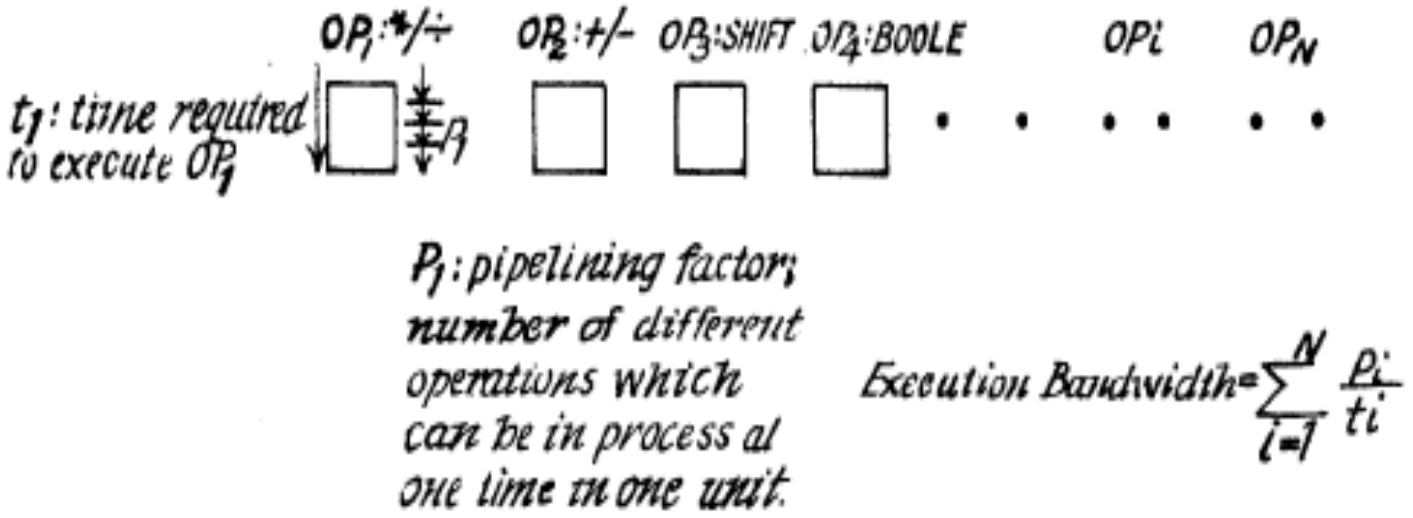
- Execution bandwidth is always greater than the maximum performance of large systems
- In SISD, Execution bandwidth is given by,

$$\text{execution bandwidth} = \sum_{i=1}^{N} \frac{P_i}{t_i}.$$

P_i – Pipelining factor for the ith unit in the SISD processor containing n different processing untis.

t_i- is the time required to fully execute the ith type of operation.

It is given by the following diagram,



Instruction Control:

A control area is responsible for finding the operand sink and source positions. The decision efficiency of this area is possible when it handles minimum number of interlocks and exceptional conditions and when it is being continually utilized.

overlapped SIMD organizations has the simplest control structure when compared to SISD and SIMD.

Primary and Secondary Storage:

Usually it is considered that a storage is most effeciently used when both the program and the data reside in the storage for the minimum time possible.

Storage cost is given by $$\text{storage cost} = \sum_i c_i \cdot s_i \cdot t_i$$

i is the level of storage heirarchy

c_i is the cost per word at that level

s_i is the average number of words the program used

t_i is the time spent at that level

Finally,
MI organizations will require both the data and the instruction to be available simultaneously at the lower levels while SIMD required only simutaneous access to M data sets and SISD has least intrinsic storage demands.
Thus in general,

$$S_1 \mid_{\text{MIMD}} \geq S_1 \mid_{\text{SIMD}} \geq S_1 \mid_{\text{SISD}}$$

This shows that MIMD and SIMD should have mre efficient program execution t_i to have optimized the use of storage resource.