Alternative Approaches

To

Concurrency

Outline

* **Concurrency Basics**
  - -- Granularity
  - -- SIMD/MIMD
  - -- Supercomputers vs. Multi
  - -- Data Flow vs. Control Flow

* **Data Flow Basics**
  - -- Fire when ready
  - -- Irregular parallelism
  - -- Instances
  - -- Example programs

* **Single instruction stream**
  - -- SIMD (Vectors, Arrays)
  - -- VLIW (now EPIc)
  - -- DAE
  - -- HPS

* **MP Basics**
  - -- Metrics: Speedup, Redundancy, Efficiency
  - -- Amdahl's Law
  - -- Cache Coherency (Consistency)
  - -- Interconnection Networks
    (cost, latency, contention)

* **NOT Single instruction stream**
  - -- cm* (NUMA)
  - -- HEP (today, SMT)
  - -- Hypercube
  - -- Target-triggered (the MOV instruction)
  - -- CMP
  - -- Tiling the plane
    - x early: nonVon, BVM, CM-1
    - x today: TRIPS, Cell, Niagra, RAW, Wavescalar

Granularity of Concurrency

* Intra-Instruction (Pipelining)

* Parallel Instructions (SIMD, VLIW)

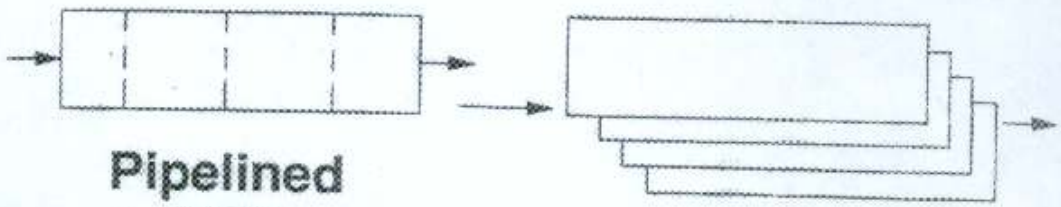
* Tightly-coupled MP

* Loosely-coupled MP

SIMD/MIMD

SISD	The Typical Pentium-Pro, for example
MISD	
SIMD	Array Processor, Vector Processor
MIMD	Multiprocessor

and, Note:

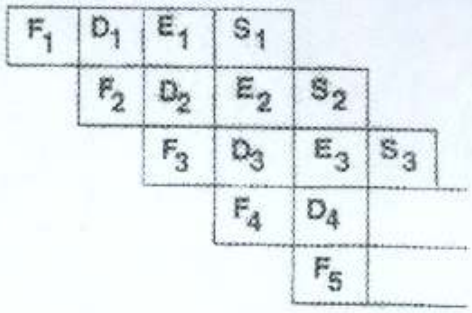


Pipelined SISD

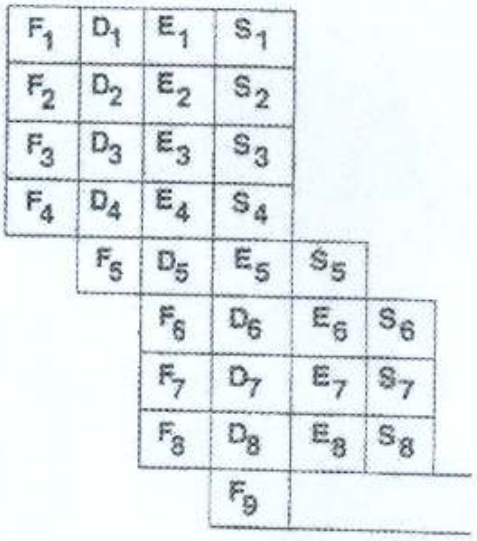
SIMD

Pipelining

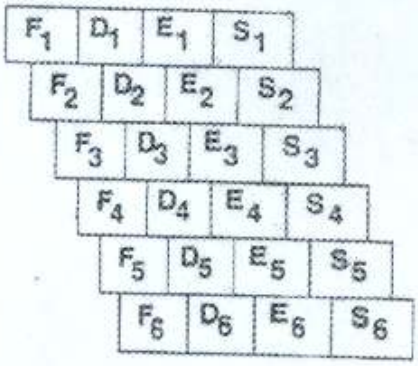
Pipelined:



Superscalar:



Superpipelined:



One Supercomputer

vs.

"The Multi"

(...Except Even Supercomputers have adopted the multi approach)

$1 * 2^n$ $2^k * 2^{n-k}$ $2^n * 1$

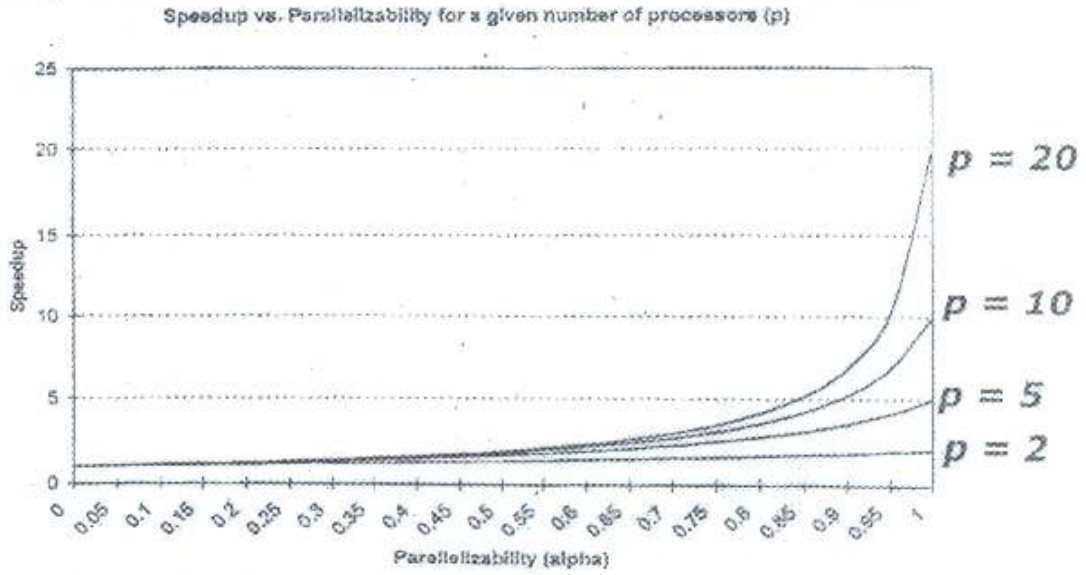
Why do we care?

- *-- Economic Answer*
- *-- Strategic Answer*
- *-- Scientific Answer*

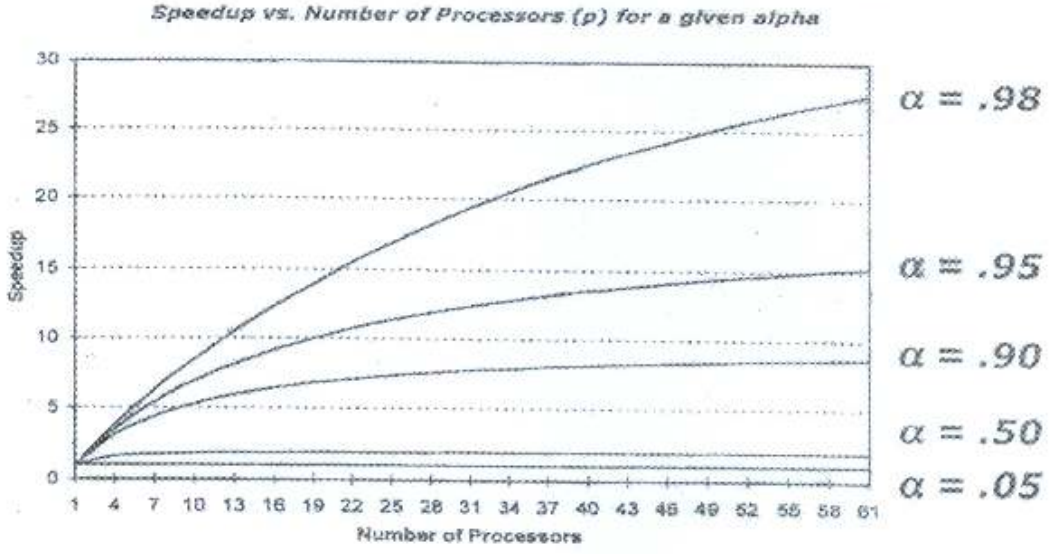
Scalability

Amdahl's Law

* Speed-up as a function of the parallelizability (α) of the application



* Speed-up of an application as we add more and more processors (p)



MP vs. Multicomputer Network

* Shared memory vs. Message passing

* Easier for software, or easier for hardware

* No free lunch
  - Cache Consistency
  - Memory Contention

A Unit of Computation:

The Data Flow Node

OR,

*	R	ARG1	R	ARG2	Dest. Of Result

The Operation
(In Larger Granularity Systems,
"The Compound Function")

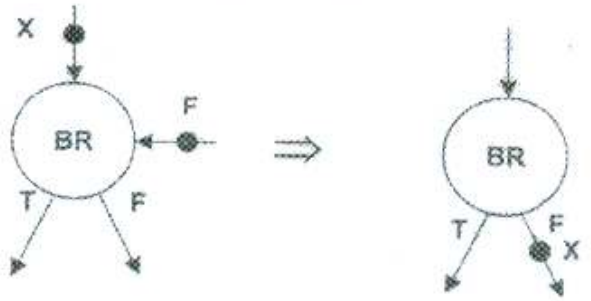
Fires
When
Ready

The Firing Rule:

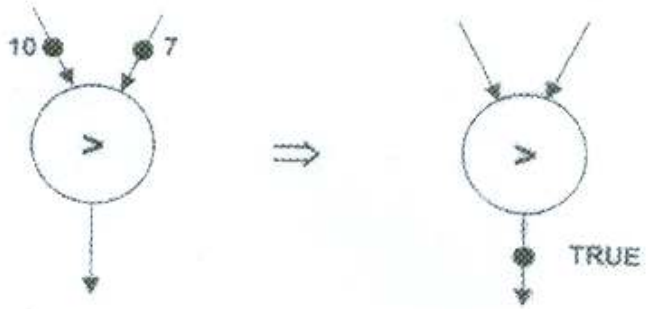
When all Inputs Have Tokens

(Note: Safe vs. Queues)

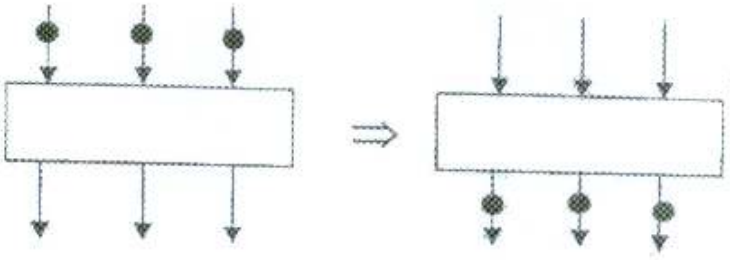
*Conditional



*Relational

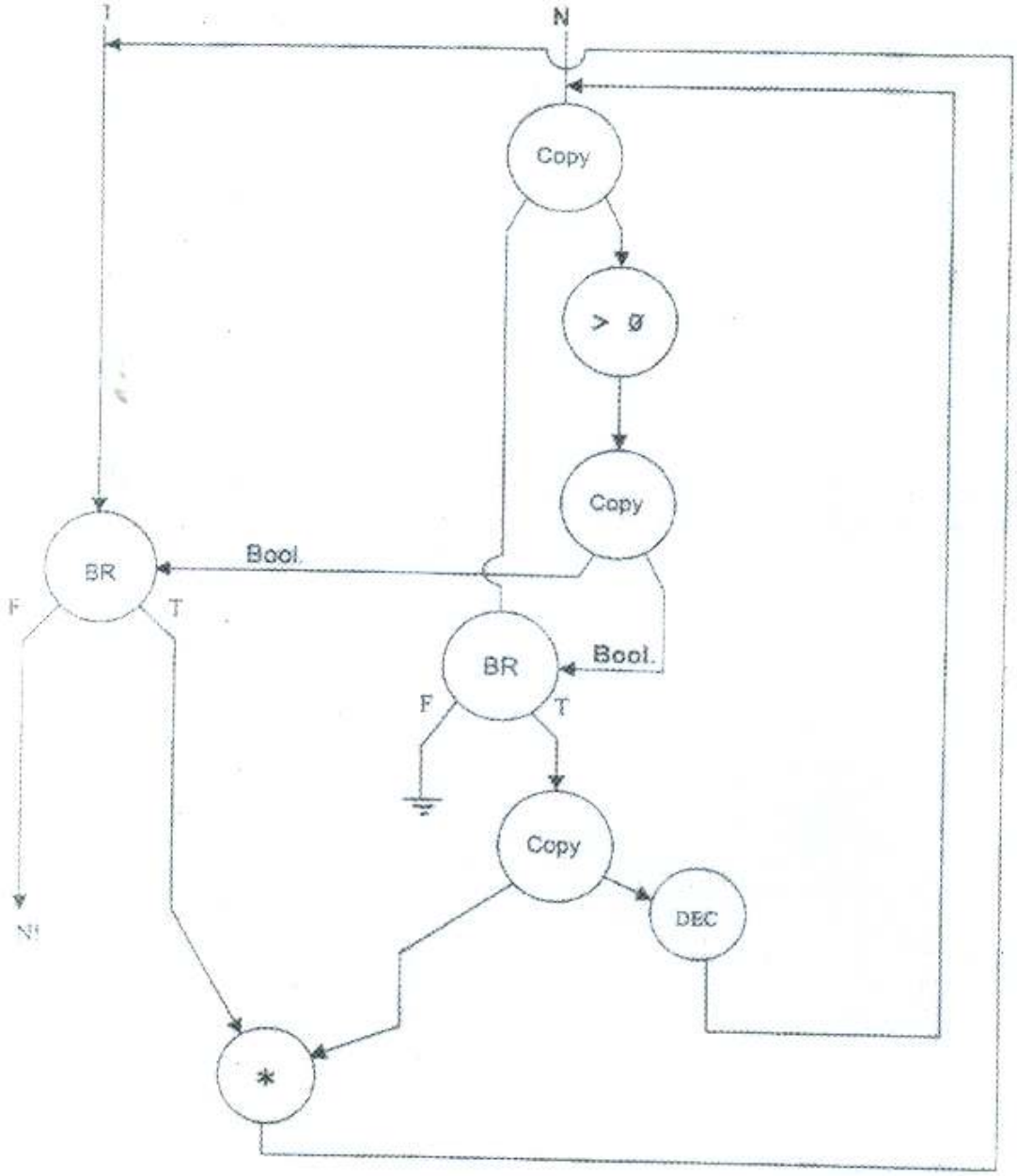


*Barrier Synch



An Example Data Flow Program:

Factorial (Done, Iteratively)

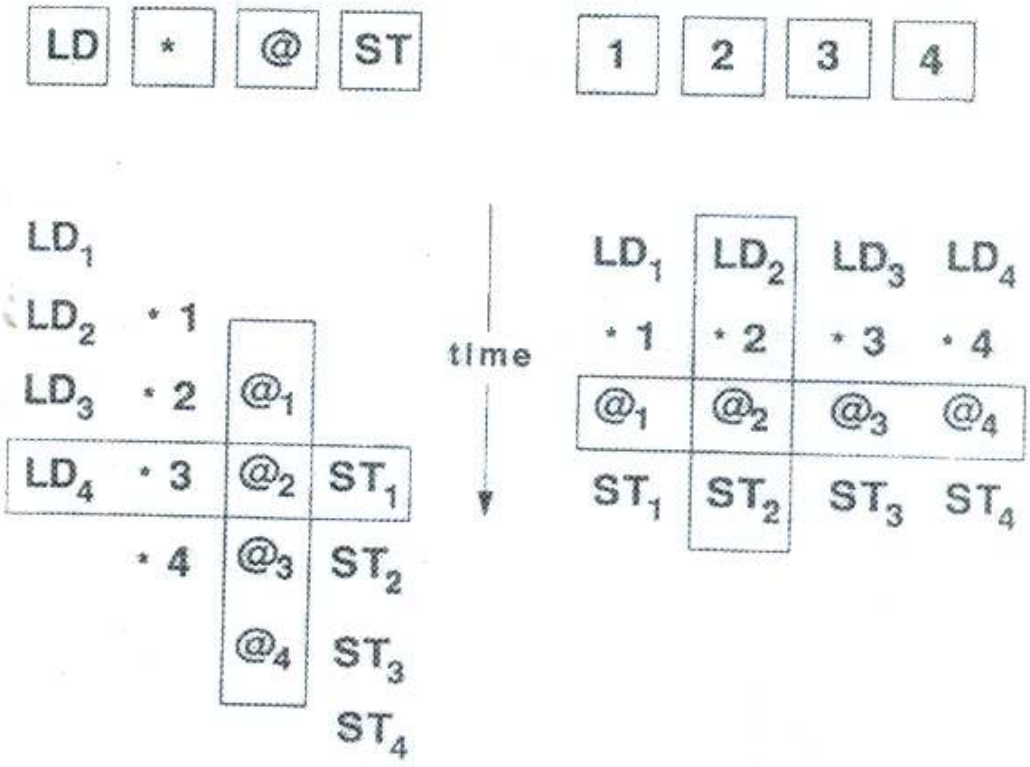


Characteristics of Data Flow

* Data Driven Execution of Instruction-level Graphical Code
  - --Nodes are Operators
  - --Arcs are I/O
* Only REAL Dependencies Constrain Processing
  - --Anti-Dependencies Don't (write-after-read)
  - --Output Dependencies Don't (write-after write)
  - --NO Sequential I-stream (No PC)
* Operations Execute Asynchronously
* Instructions Do Not Reference Memory
  (at least, memory as we understand it)
* Execution Triggered By Presence of Data
  - --Safe vs. Queues

SIMD

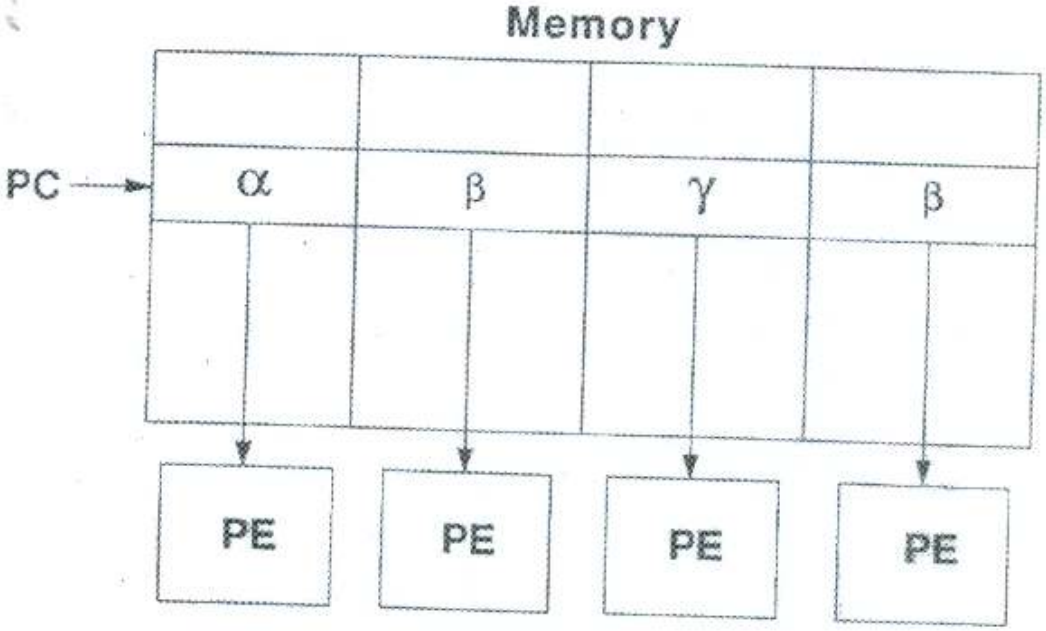
Vector Processors, Array Processors



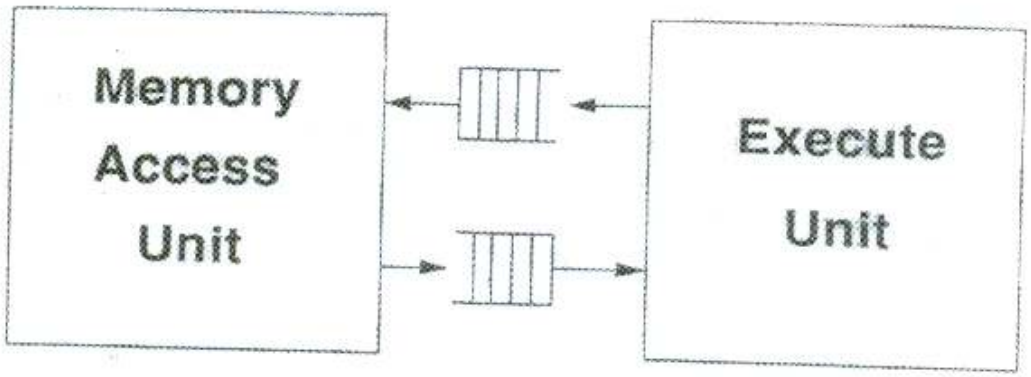
VLIW

* **Static Scheduling**
  - **Everything in lock step**
  - **Trace Scheduling**

* **Generic Model**



Early Form of Decoupled - Access/Execute



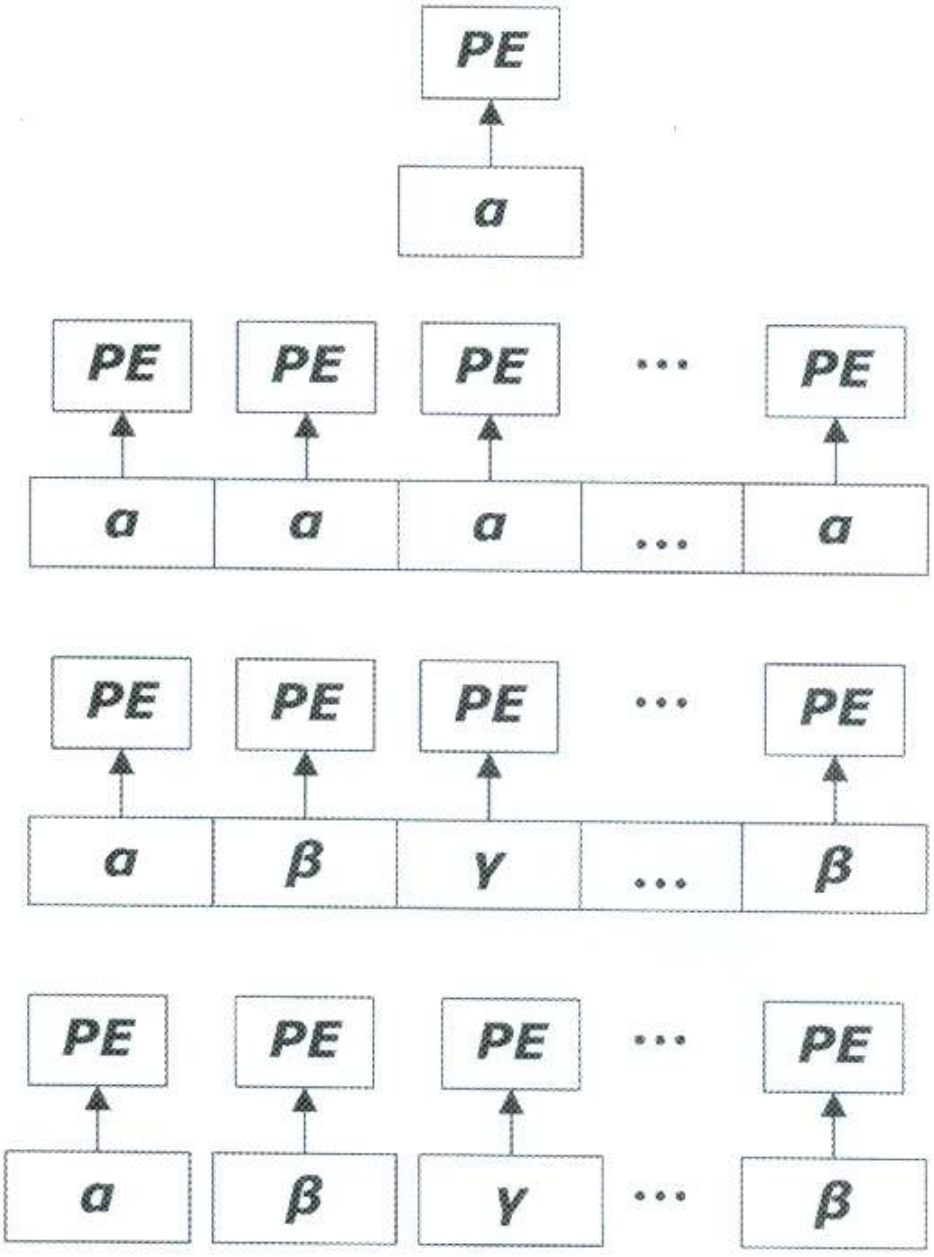
* Andrew Plezskun, Univ. of Illinois

SMA

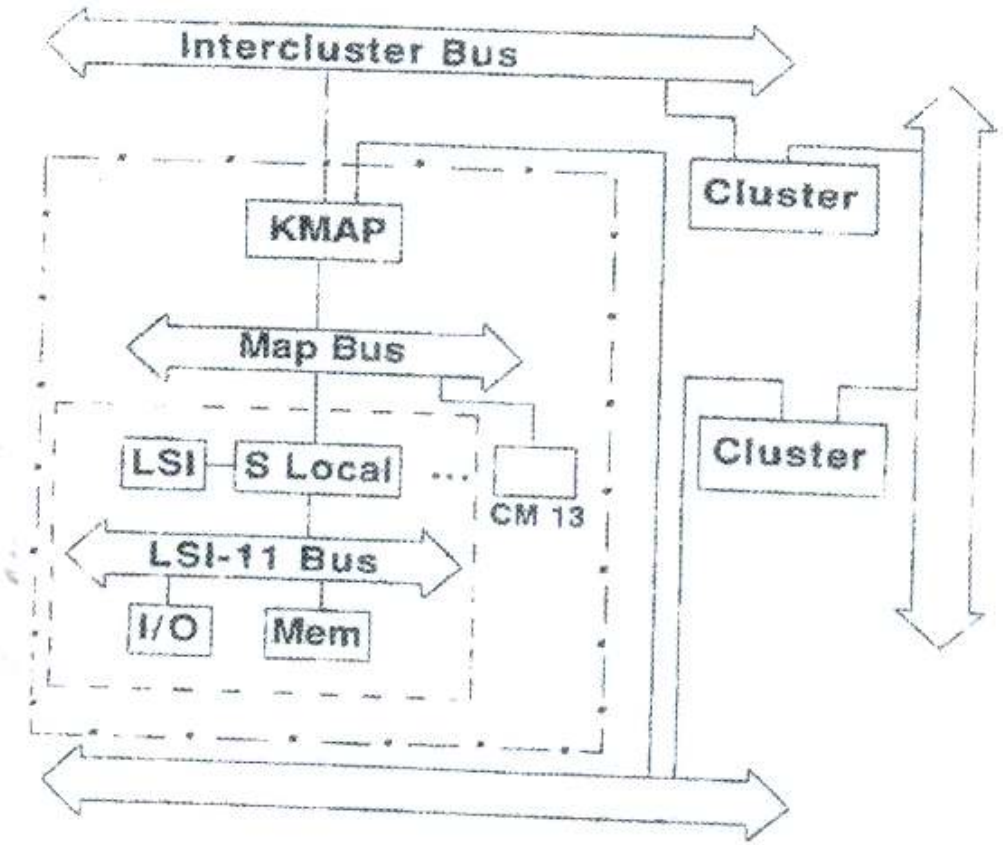
* James E. Smith, Univ of Wisconsin

DAE

HPS As Evolution

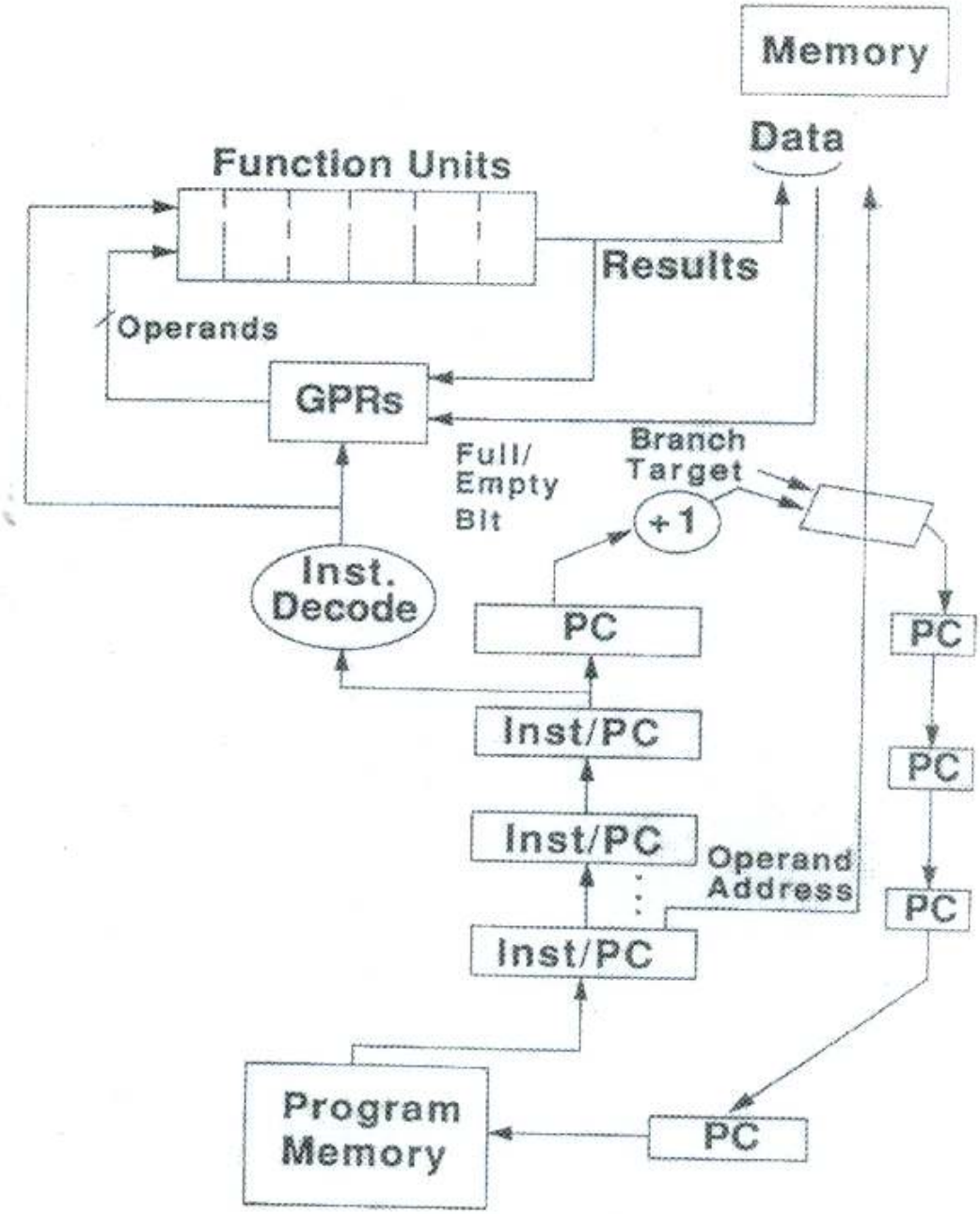


cm*



Note: ***A well-meaning student told me to get rid of this slide. cm* is old. People will think you are an old man, and not take you seriously.***

The HEP


Memory
Data
Function Units
Results
Operands
GPRs
Full/
Empty
Bit
Branch
Target
+1
Inst.
Decode
PC
PC
Inst/PC
PC
Inst/PC
Operand
Address
PC
Inst/PC
Program
Memory
PC

Cosmic Cube

(Example: k = 4)