Computer Architecture Research and Future Microprocessors

Where do we go from here?

Yale N. Patt
The University of Texas at Austin

ISCA-33
Boston, Massachusetts
June 19, 2006

...in Two Parts

* ***Computer Architecture Research: Some criticisms and my comments***

* ***Tomorrow's (circa 2014) Microprocessor***

Computer Architecture Research

-- Is it Dead? (Nothing left to do)

-- Does it Need Revitalization?

-- Is it a Scientific Discipline?

A Scientific Discipline

* *Repeatability of Experiments*
  - *-- Use of web sites*
  - *-- Who said to publish it?*

* *What About*
  - *-- Simplescalar*
  - *-- Liberty*
  - *-- RAMP*
  - *-- Etc.*

* *Quoting original sources*
  - *-- Cache memory?*
  - *-- Predicated execution*
  - *-- 2-bit counter*

Naysayers

* *Go for the low-hanging fruit*

  --Easy to get great improvement
  (The hard problems still remain)

* *CAD can't verify*

  --Make simpler machines
  (Make better CAD tools)

* *Law of Diminishing Returns*

  Performance does not track resources applied
  (Let's look more closely)

* *We don't need anything faster*

  1986: MIPS R2000, 1996: Pentium Pro, 2006: ??
  (Check out the marketplace)

* *Processor can't keep up*

  --Memory Wall
  (Algorithm, compiler, microcode)

* *Our experiment proves it can't be done*

  1.85 IPC, for example
  (Let's look more closely)

Moore's Law:

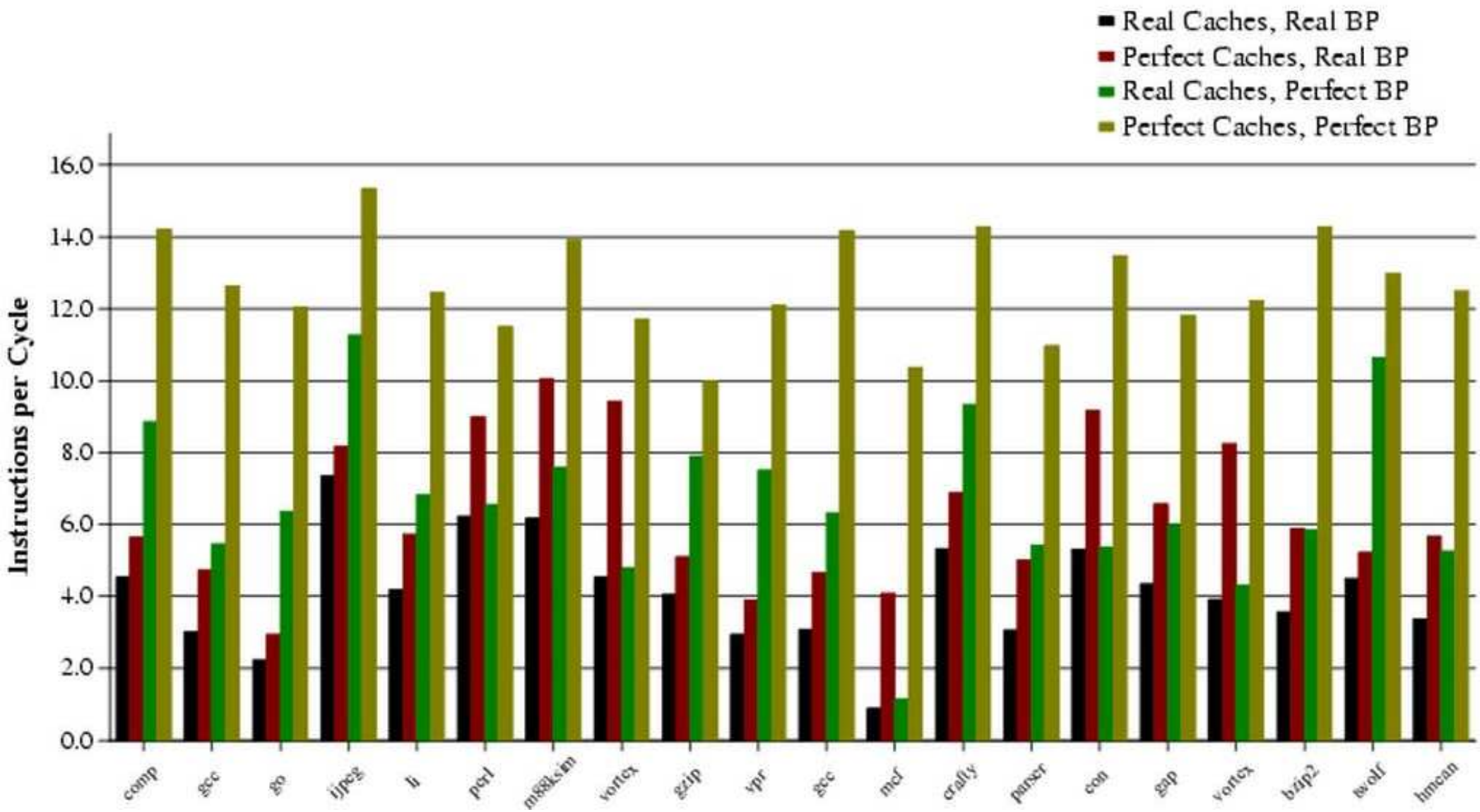
(a) Physics?

(b) Process Technology?

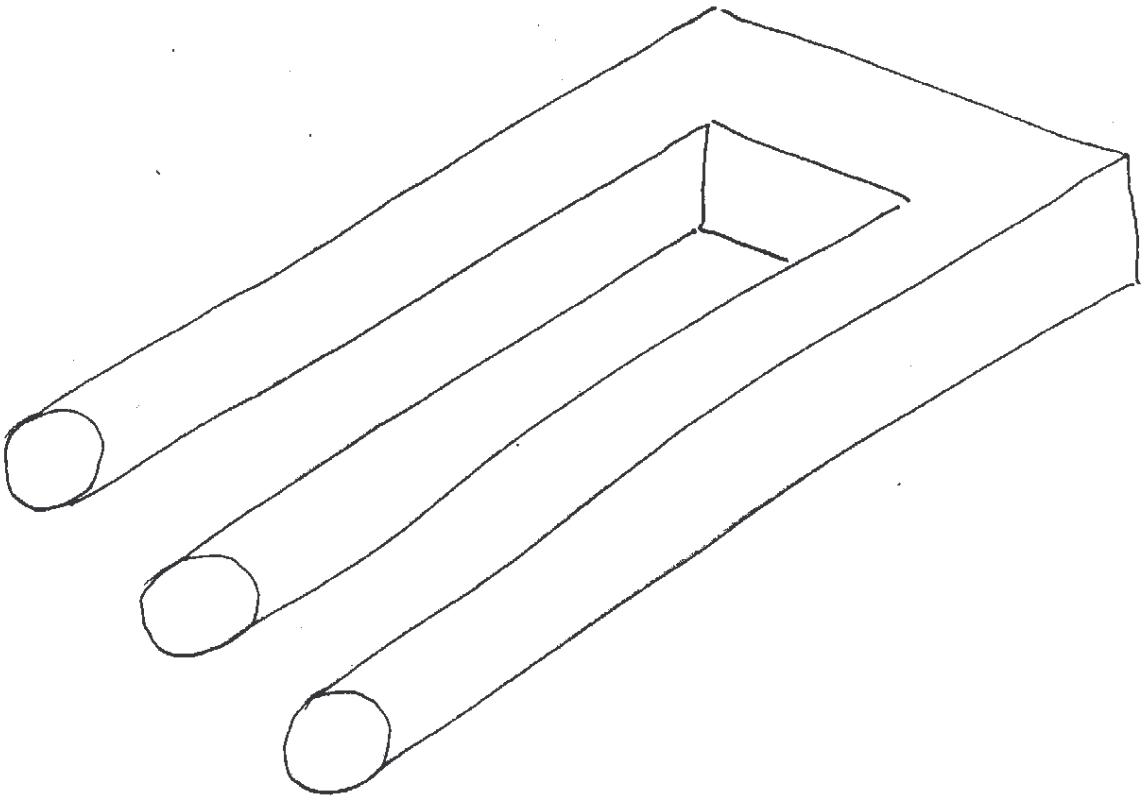
(c) Microarchitecture?

(d) Psychology?

Potential of Improving Caches and BP



Think Outside the Box



Why Revitalization?

* *Pipelining*
* *Separate I and D caches*
* *Wide Issue*
* *Aggressive Branch Prediction*
* *Speculative Execution*
* *Out-of-order execution, in-order retirement*
* *Trace Cache*
* *SMT*
* *SSMT (aka helper threads)*
* *CMP*
* *MLP*

* *(or, I could look at the compiler's contributions)*

A Few Specifics

* ***HPS – expanded on Tomasulo***

* ***SMT – expanded on Burton***

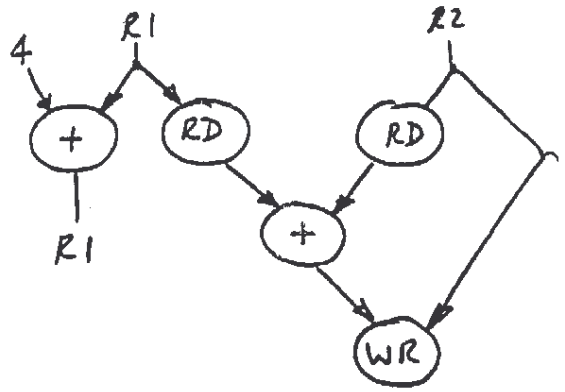
* ***Perceptron predictor – expanded on Widrow/Rosenblatt/etc.***

HPS

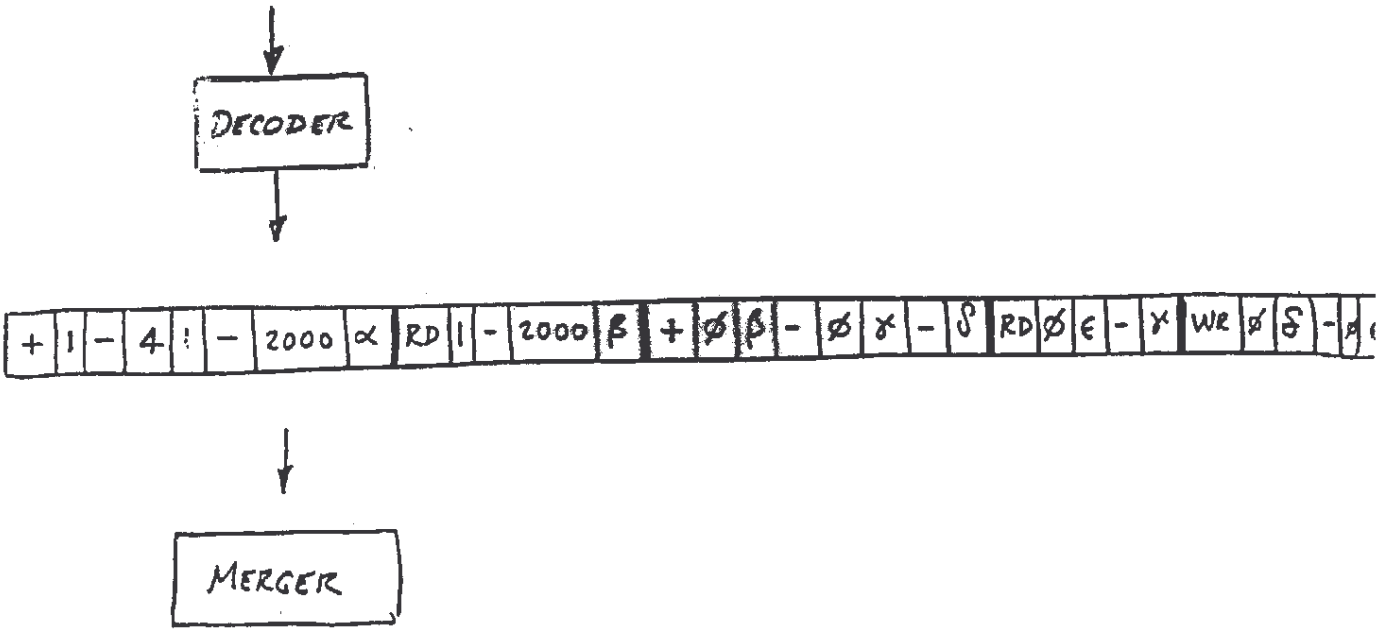
(Restricted Data Flow)

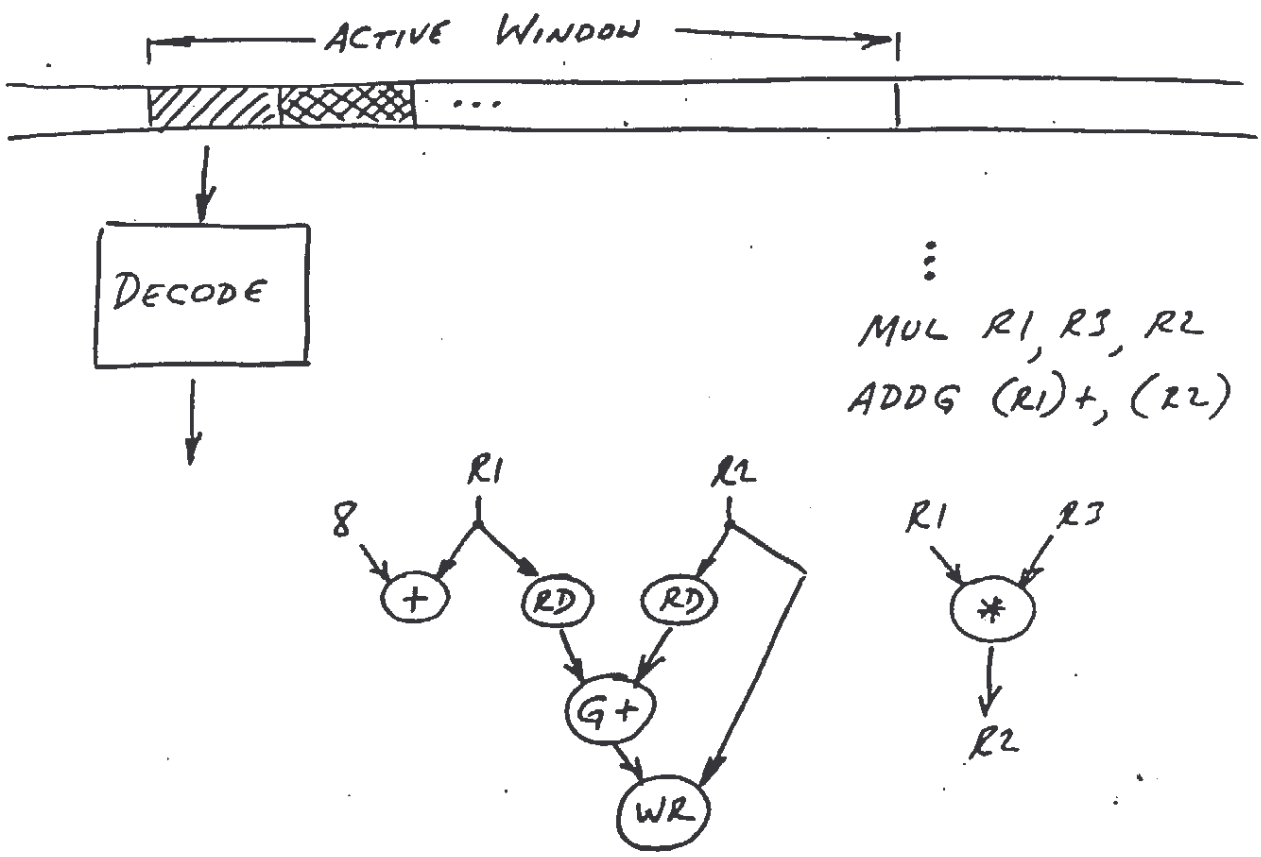
For example, the VAX instruction:

ADDL2 (R1)+, (R2)



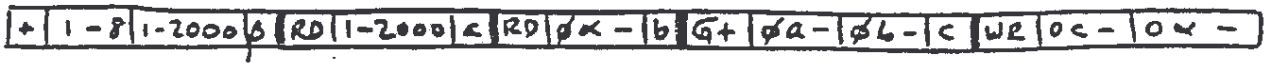
VAX Instruction





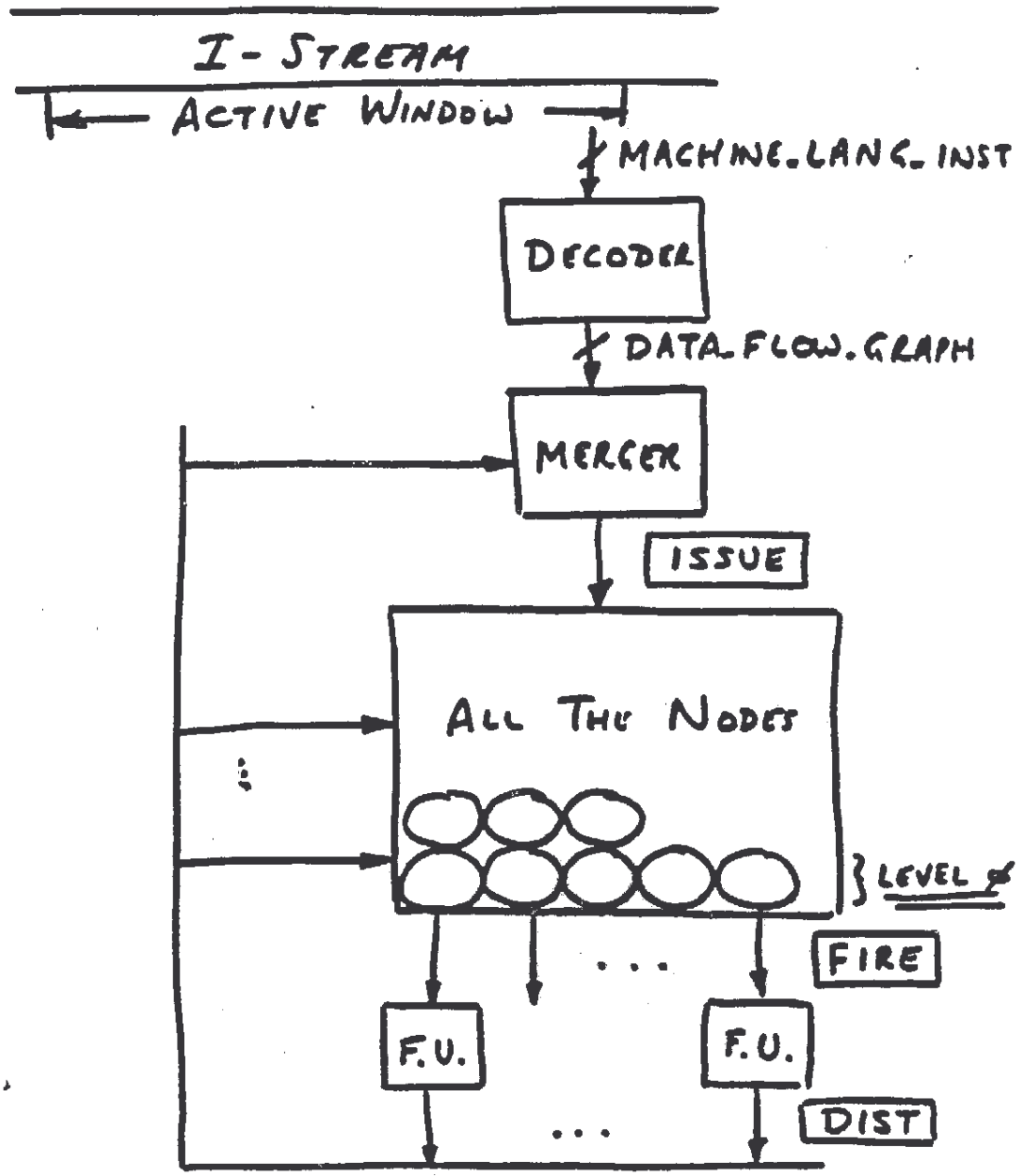
à la Tomasulo

R1	1	—	2000
R2	0	α	—
R3	1	—	100



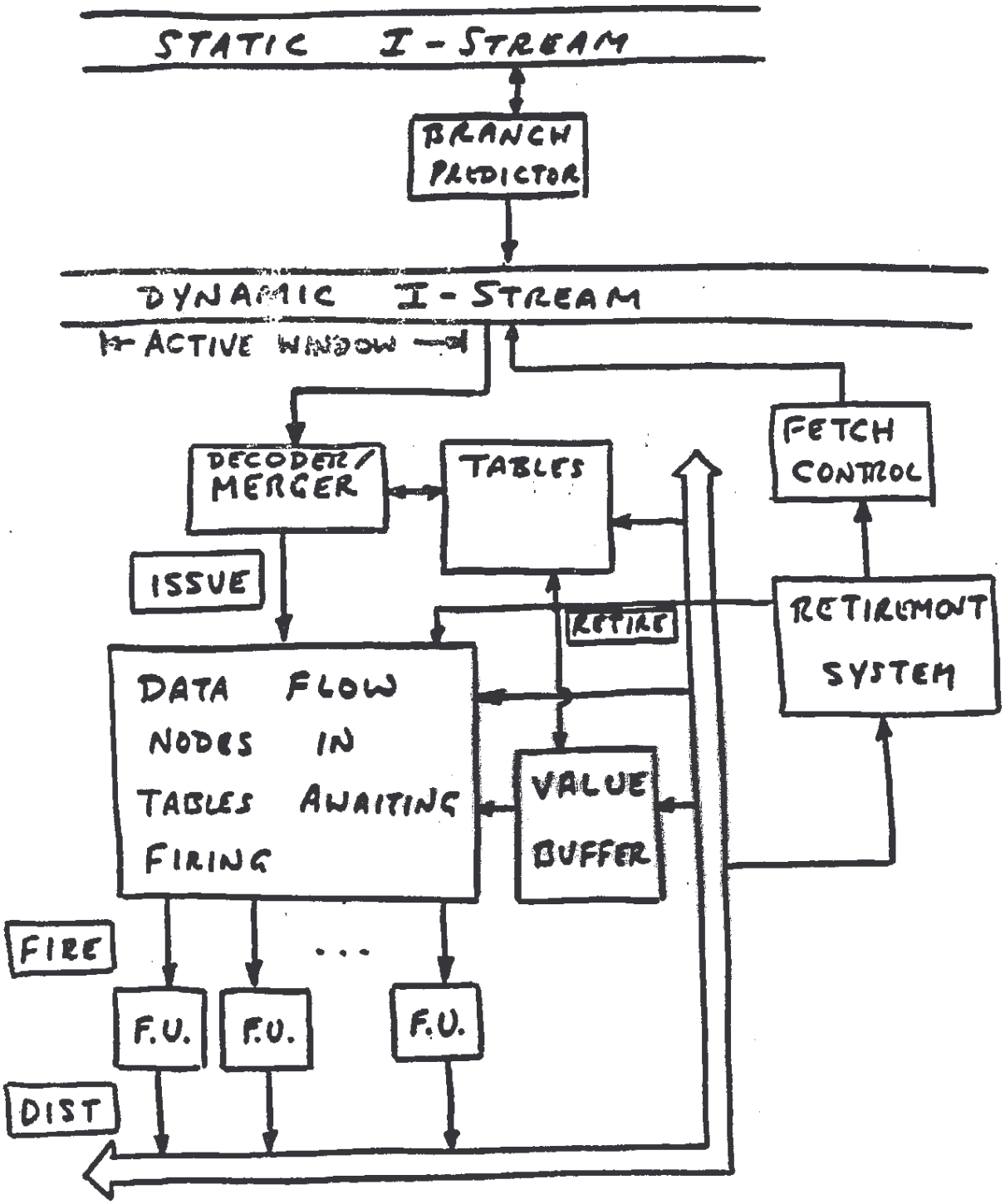
R1	φ	β	—
R2	φ	α	—
R3	1	—	100

HPS - What Is It ?



* Restricted Data Flow

HPS


STATIC I-STREAM
BRANCH PREDICTOR
DYNAMIC I-STREAM
ACTIVE WINDOW
DECODER/ MERGER
TABLES
FETCH CONTROL
ISSUE
RETIRE
RETIREMENT SYSTEM
DATA FLOW NODES IN TABLES AWAITING FIRING
VALUE BUFFER
FIRE
F.U.
F.U.
F.U.
DIST

Fundamentals

* ***Overriding consideration:***

  What does it cost?
  What is the benefit?

* ***Global View***

  --Global vs. Local transformations

* ***Microarchitecture view***

  --The three ingredients to performance

* ***Physical view***

  --Partitioning
  --Power consumption
  --Wire delay

Problem

Algorithm

Program

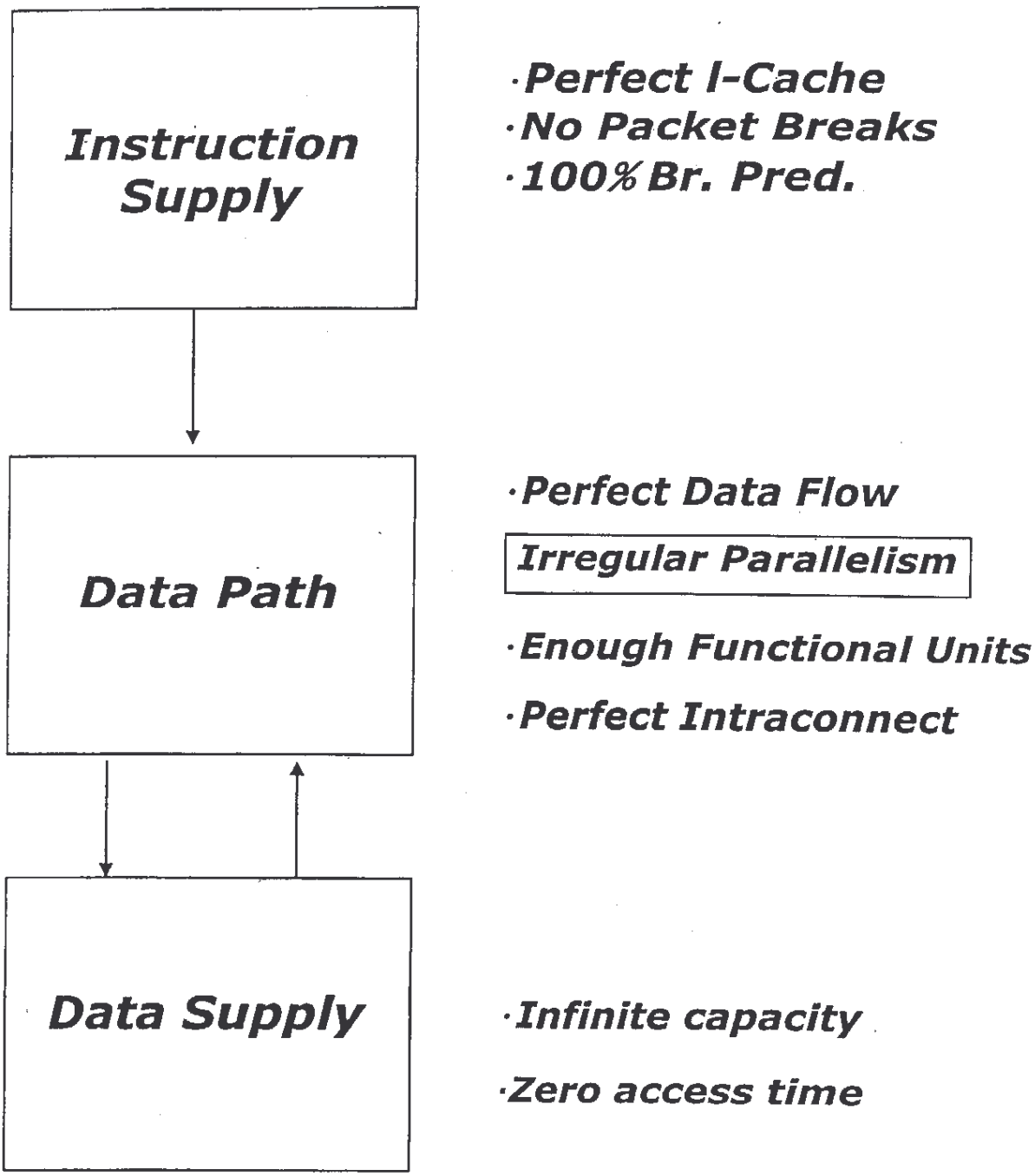
Instruction Set Architecture (ISA)

Microarchitecture

Circuits

Electrons

Microarchitecture (The Requirement)



"Tomorrow's" Microprocessor

* *Wholistic approach*
  --The transformation hierarchy revisited

* *X + Superscalar*

* *Compiler/μarch symbiosis*
  --multiple levels of cache
  --Block-structured ISA
  --part by compiler, part by μarch
  --fast track/slow track

* *Power Awareness*

* *Multiple cycle times*
  --asynch/synch together

* *SSMT (aka helper threads)*

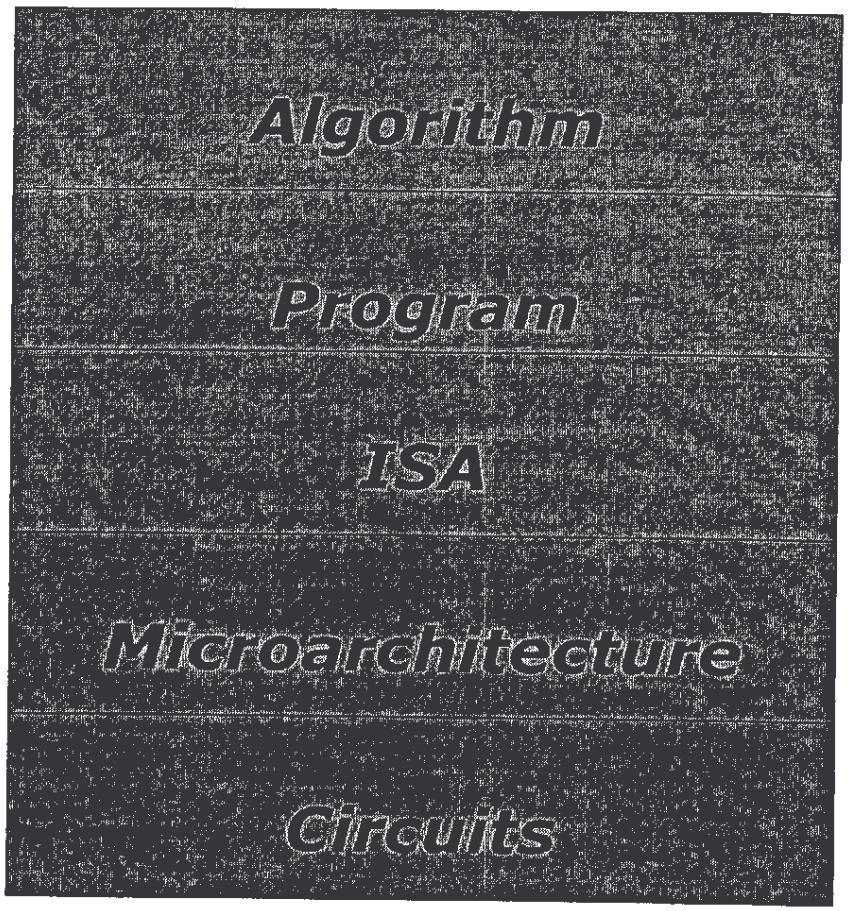
* *More Microcode*

* *Verification Hooks*

* *Internal fault tolerance*

* *Niagra X / Pentium Y*

* *Security*


Problem
Algorithm
Program
ISA
Microarchitecture
Circuits
Electrons

x + superscalar

* ***DSP + superscalar***
* ***graphics + superscalar***
* ***data base + superscalar***
* ***networks + superscalar***
* ***vectors + superscalar***

Hardware vs. Microarchitecture/Compiler

Rather than ask the Hardware to do the whole thing,

1. *Partition the problem into part compiler/ part µarch*

2. *Augment ISA to deliver compiler-generated information to the µarch*

3. *µarch manages the optimization, combining compile-time part and hardware part.*

Fast track / Slow track (Another compiler/μarch symbiosis)

1. *Implement the ISA in two parts:*

   --Fast track: those things that can go fast

   --Slow track: those things that can't

2. *Compiler knows what is done where, and compiles accordingly*

Power

***Virtual Allocate, Physical Store (UPC)**

***Retire Does Not Necessarily Mean Update**

***Partition The Cache, Don't Replicate**

***Demand Only Broadcast**

***The Refrigerator**

***No More Registers (Only Buffers)**

***Block-Structured ISA**

An observation:

> ***Computer Architecture will always be alive and healthy as long as people can dream***

Reason:

> ***Computer architecture is about the interface between what technology can provide and what the market demands***

i.e.:

> ***As process technology continues to advance, and dreamers find new uses for computers, Computer Architecture's future should be very, very bright***