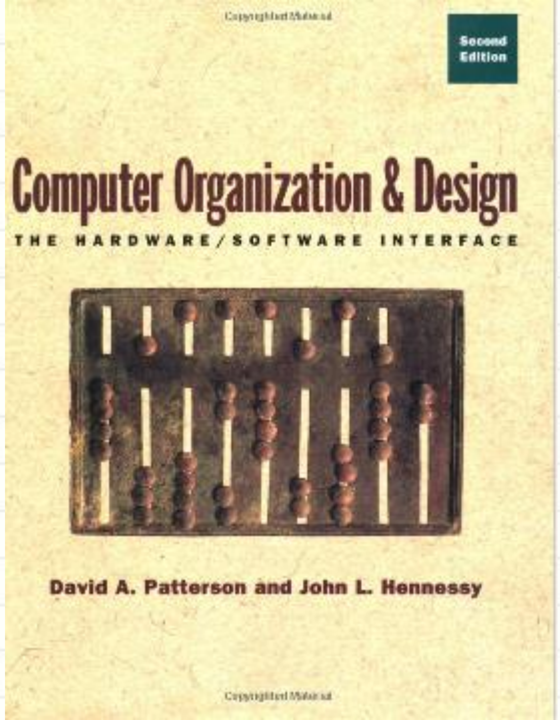
COL718 Architecture of High Performance Systems

Lecture : 1

What is the Course About?

- Coverage
  - Overview of Basic Computer Architecture
  - Pipelining
  - Cache in full glory
  - Parallel Architecture
  - Heterogeneous Architecture
  - Emerging Architectures



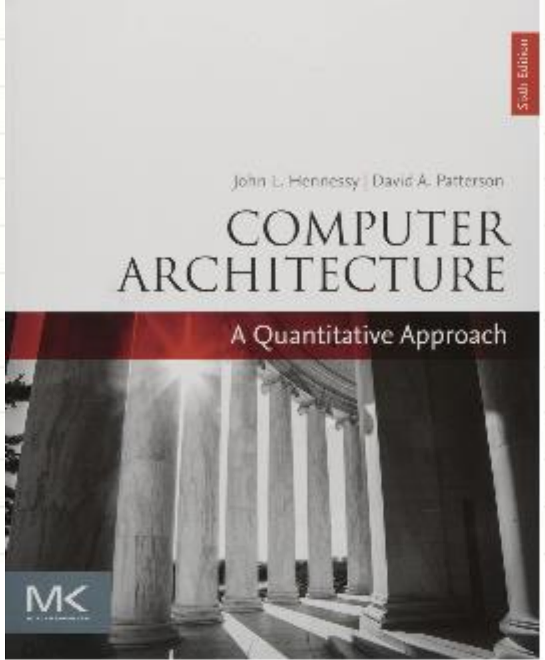
- Assumption
  - You have done
    - Computer Organization and Design The Hardware Software Interface by David A. Patterson John L. Hennessy
      - Understood and revised

How will we do this course?

- Grading Policy
  - Lab Assignments : 25% [including [[un]announced] Quiz and Viva]
  - Minor (30%)
  - Major (45%)
    - Pass: 35 [Minimum of 10 from Labs]
    - reMinor and reMajor will be "harder"
- No Late submissions
  - Except genuine medical conditions
    - IIT Hospital certifies
    - All Labs to be done individually
      - Marks allocated only if you are able to answer in VIVA
- Unfair Means
    - Minimum 2 Grade penalty or DISCO

Logistics?

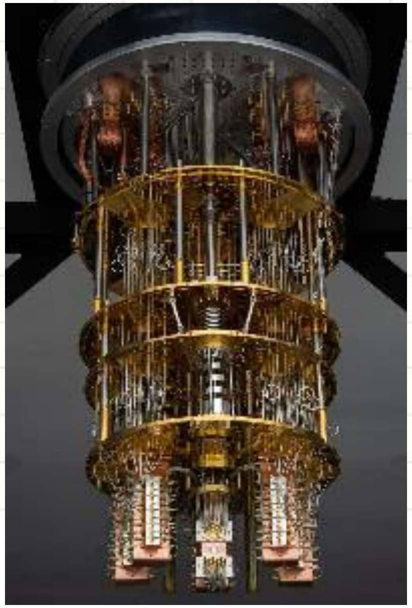
- Credits: 4 (3-0-2)
- Timing
  - Slot B: Mon, Tue : 930 to 11 AM Blk IV LT 2
  - **NO cell phone in class**
    - You will leave the class [no apologies required]
      - Two hits and automatic deregistration from Course
- You need to come to class and mark attendance
  - One grade penalty for attendance below 75%
- No Slides
  - Moodle and Course Website
    - Topics from Book will be enumerated
- Book
  - Computer Architecture: A Quantitative Approach
    John L. Hennessy David A. Patterson
  - Papers



What is Computer Architecture?

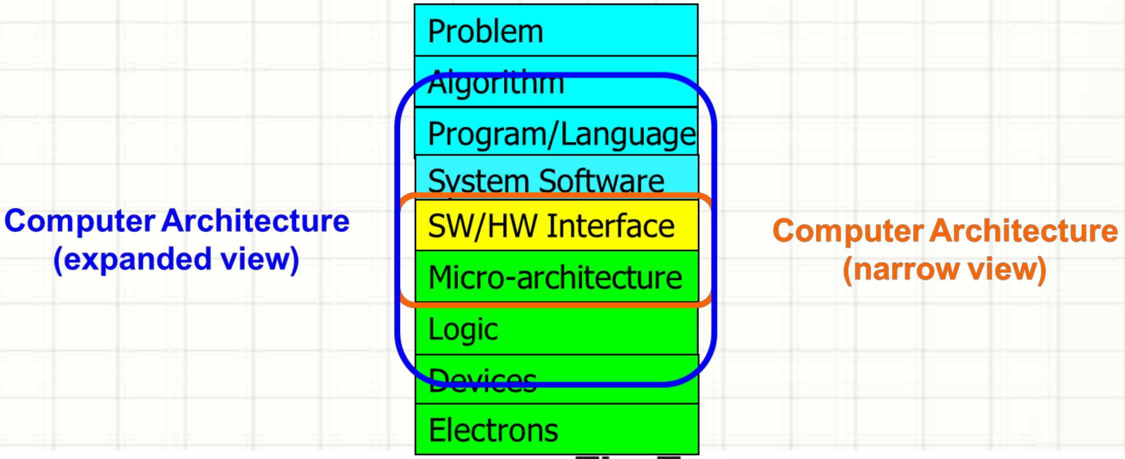
- The science and art of designing computing platforms
  - hardware, interface, system SW, and programming model
- Objective
  - Achieve a set of design goals
    - Performance
    - Performance/watt
    - etc





The Computer Engineering Hierarchy

- Plenty of Room
  - Top: Leiserson et. al.
    - "There’s plenty of room at the Top: What will drive computer performance after Moore’s law?", Science, 2020
  - Bottom: Feynmann
    - "There's Plenty of Room at the Bottom: An Invitation to Enter a New Field of Physics”, a lecture given at Caltech, 1959



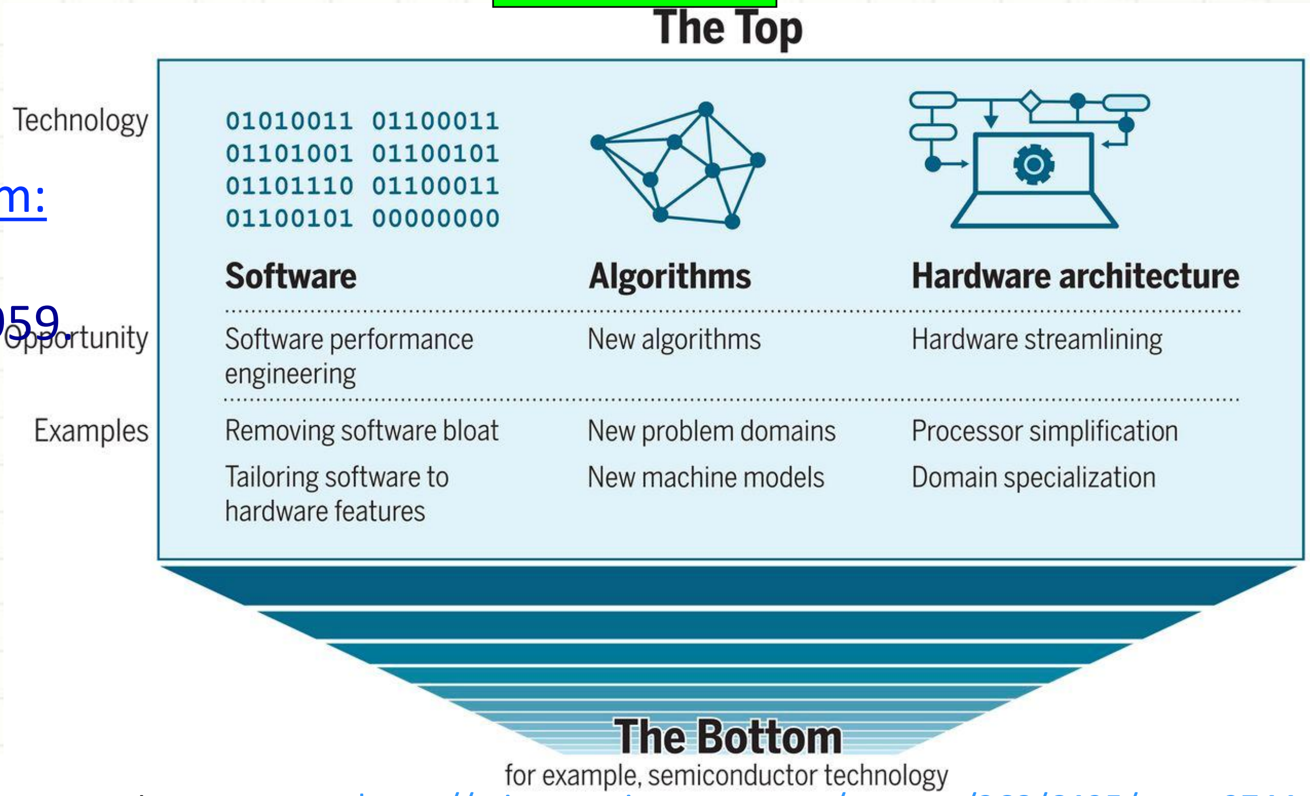


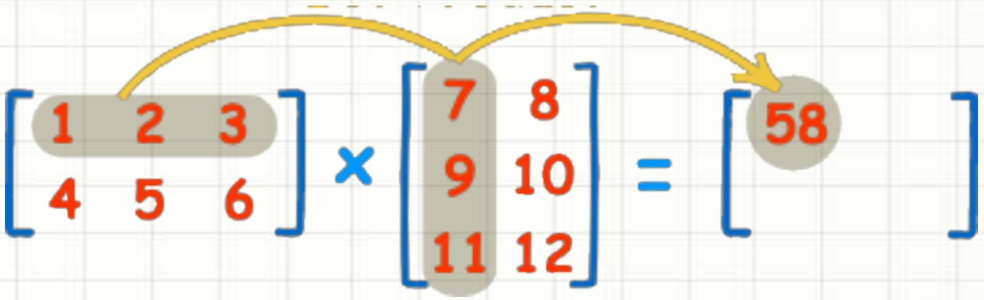
Image source: https://science.sciencemag.org/content/368/6495/eaam9744

Indian Institute of Technology Delhi

Motivating Example

- Matrix Multiplication

```
for i in xrange(4096):
  for j in xrange(4096):
    for k in xrange(4096):
      C[i][j] += A[i][k] *
B[k][j]
```

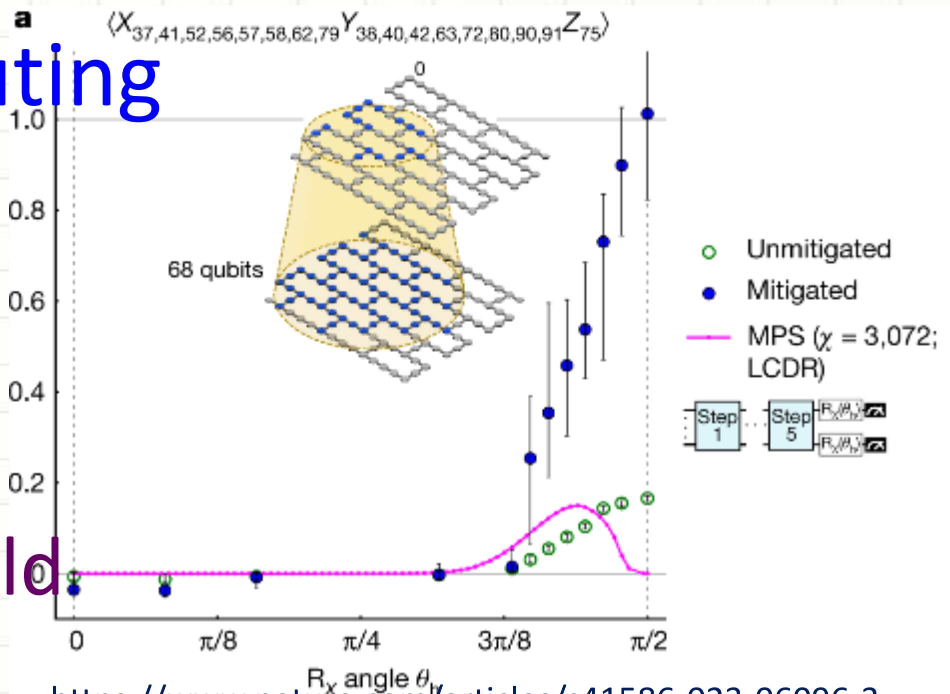


Implementation	Running time (s)	Absolute speedup
Python	25,552.48	1x
Java	2,372.68	11x
C	542.67	47x
Parallel loops	69.80	366x
Parallel divide and conquer	3.80	6,727x
plus vectorization	1.10	23,224x
plus AVX intrinsics	0.41	62,806x

Leiserson+, "There's plenty of room at the Top: What will drive computer performance after Moore's law?", Science, 2020

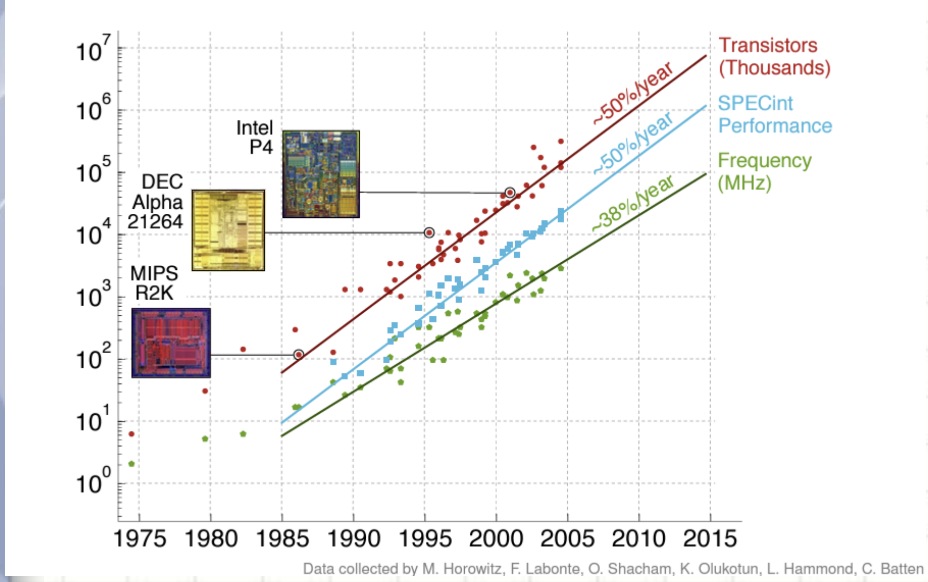
Motivating Example

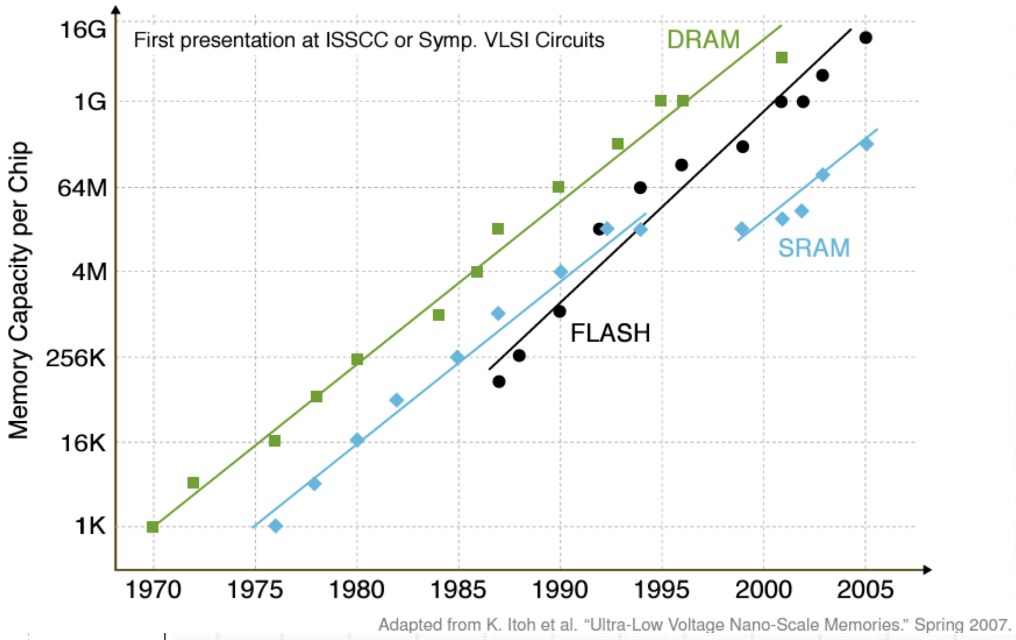
- Fault Tolerance in Quantum Computing
- Utility before FT
  - [Evidence for the utility of quantum computing before fault tolerance](https://www.nature.com/articles/s41586-023-06096-3)
  - Time evolution of a 2D transverse-field Ising mode
    - 127 qubit system
    - Demonstrated that a quantum computer can outperform a classical computer.

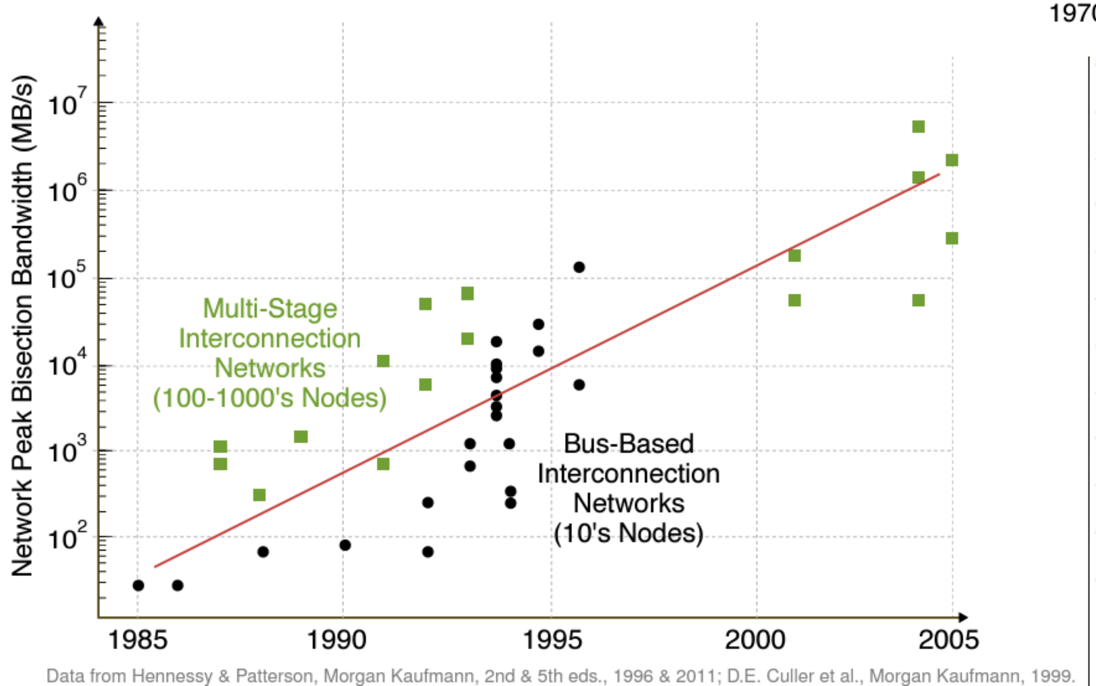


https://www.nature.com/articles/s41586-023-06096-3

Scaling: Processor, Memory and Network



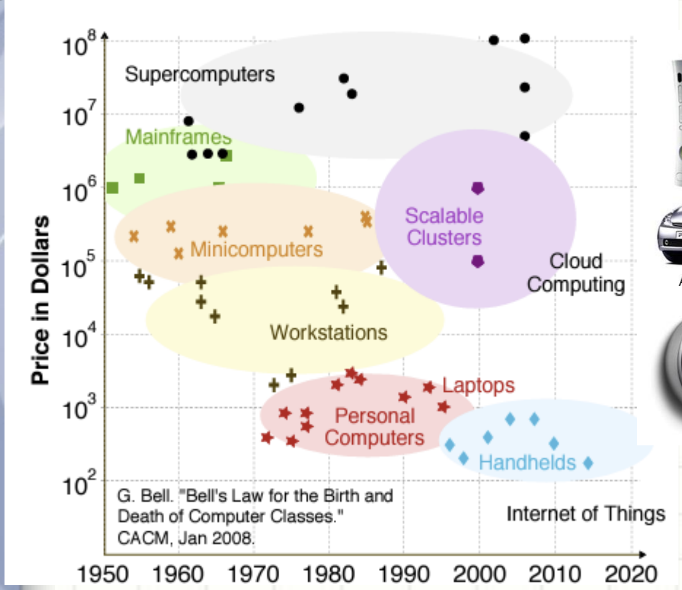


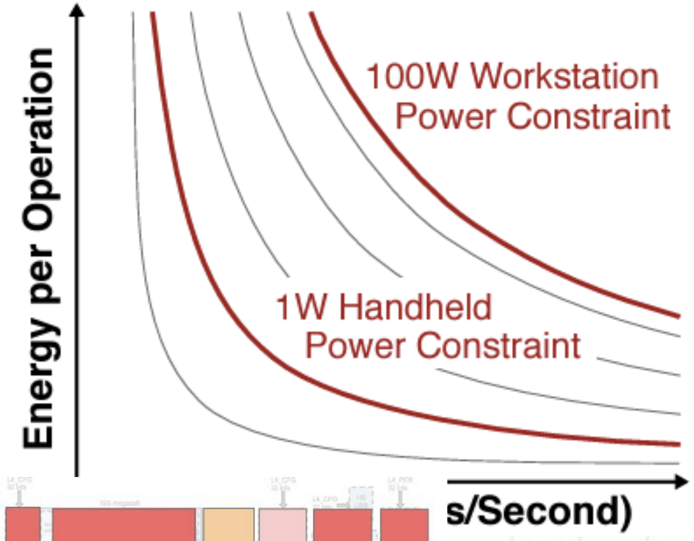


What drives CA today

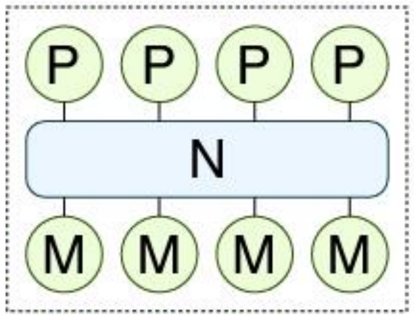
- Diversity in applications
  - Cloud and IoT
- Energy & power constrained systems
- Multiple cores
- Heterogeneous systems-on-chip
- Technology scaling challenges
  - New emerging compute, storage, and communication device technologies

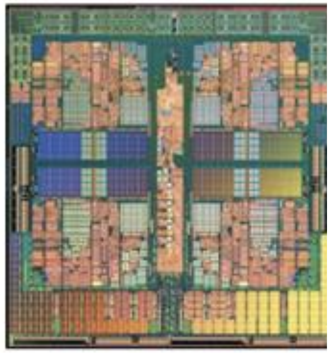
Some Trends

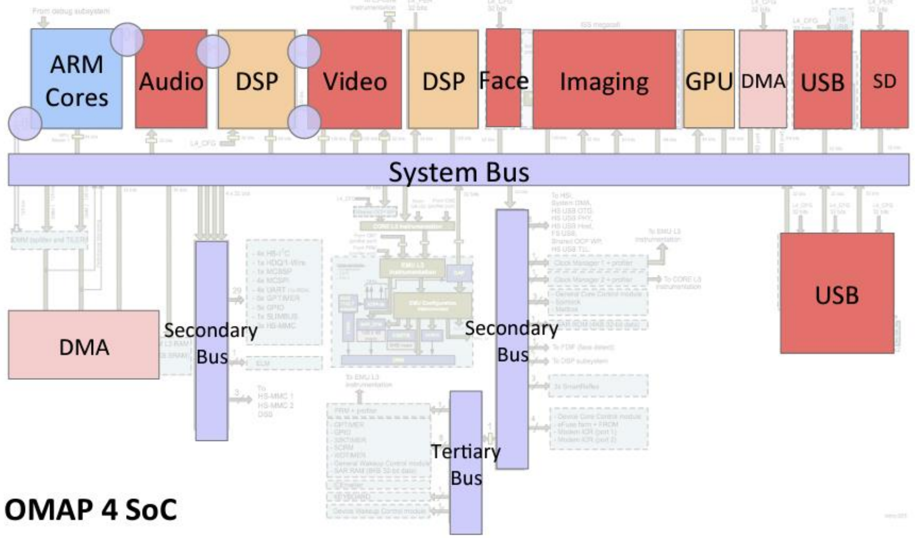


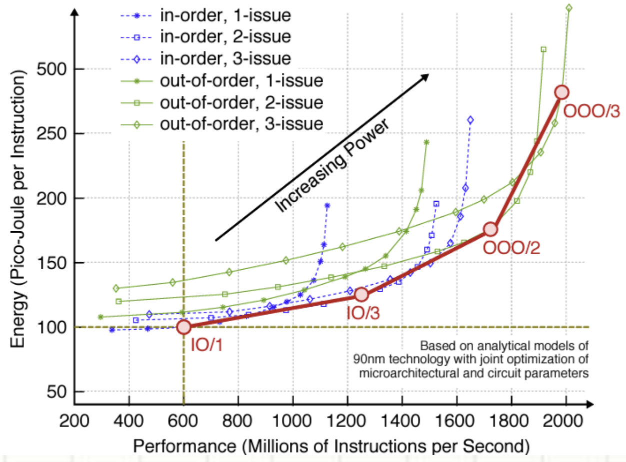


Vertical MOSFETs
Graphene
Carbon Nanotubes
Nanorelays
Quantum Computing
Molecular Computing
Memristers
Phase-Change Mem
Spintronics
3D Integration
Nanophotonics







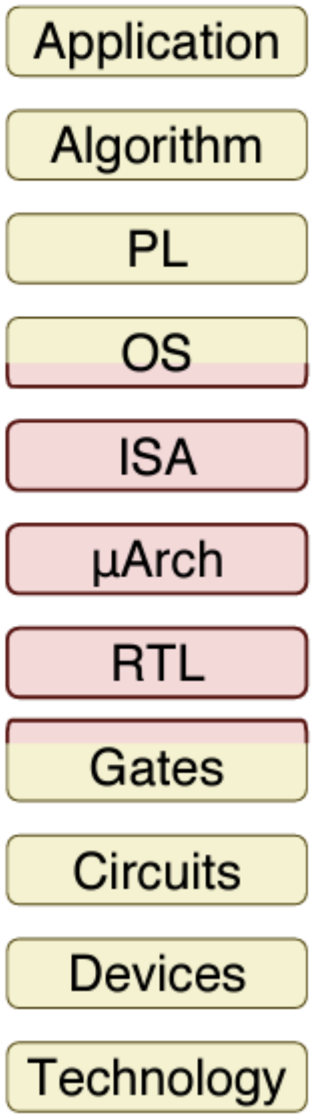


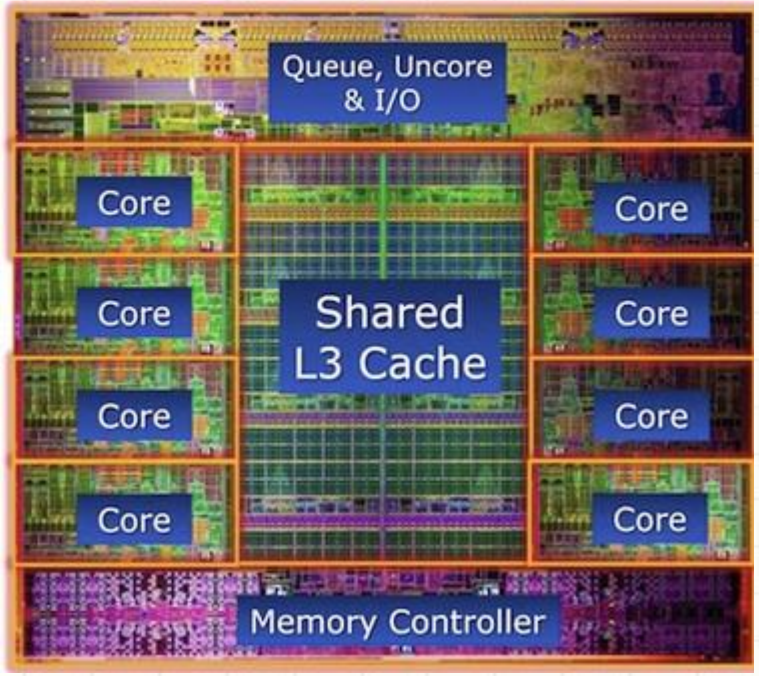
There is plenty of room both at the top and at the bottom
but **much more so**
communicate well between and optimize across
the top and the bottom

Indian Institute of Technology Delhi

Incredibly Complex

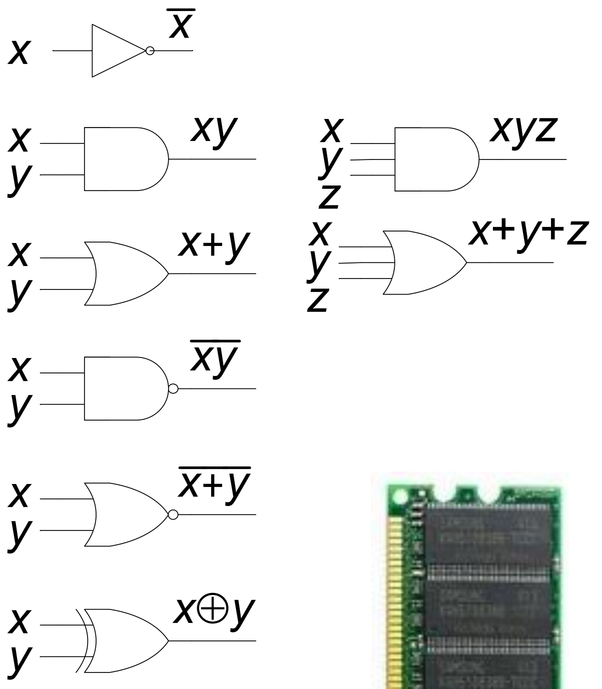
- The Design of a Modern Processor
  - Fighter Plane
    - 10 k Parts
  - Intel Sandy Bridge E
    - 2.27 Billion Transistors
- Design Philosophy
  - Modularity
  - Hierarchy
  - Encapsulation
  - Regularity
  - Extensibility
- Design Patterns
  - Control/Datapath split

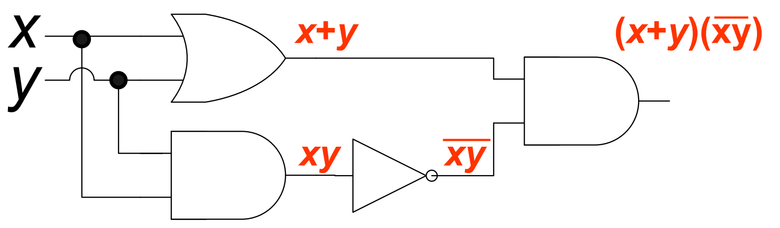


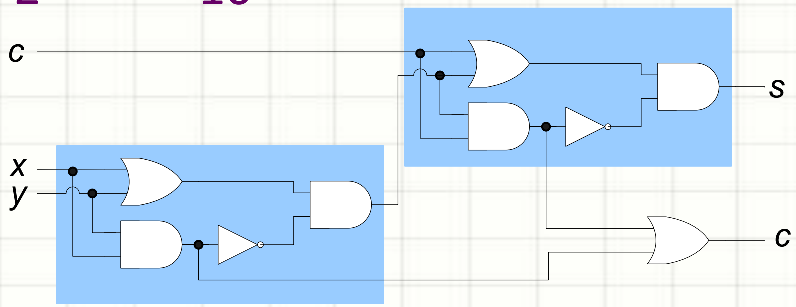


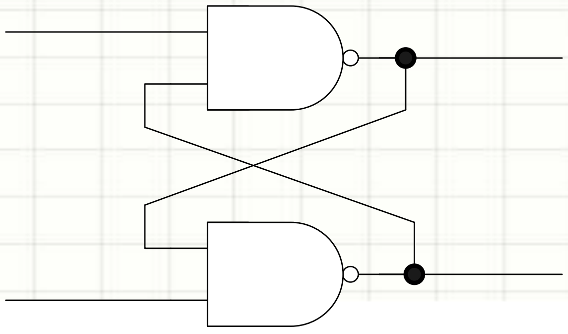
Quick Review : Digital Logic

- Boolean logic
  - Variables from {0,1}
- Truth Table
- Logic Gates
- Binary Numbers
  - $53_{10} = 00110101_2$
  - $00101001_2 = 41_{10}$
- Adder
  - Half
  - Full
- Memory
  - Flip Flops





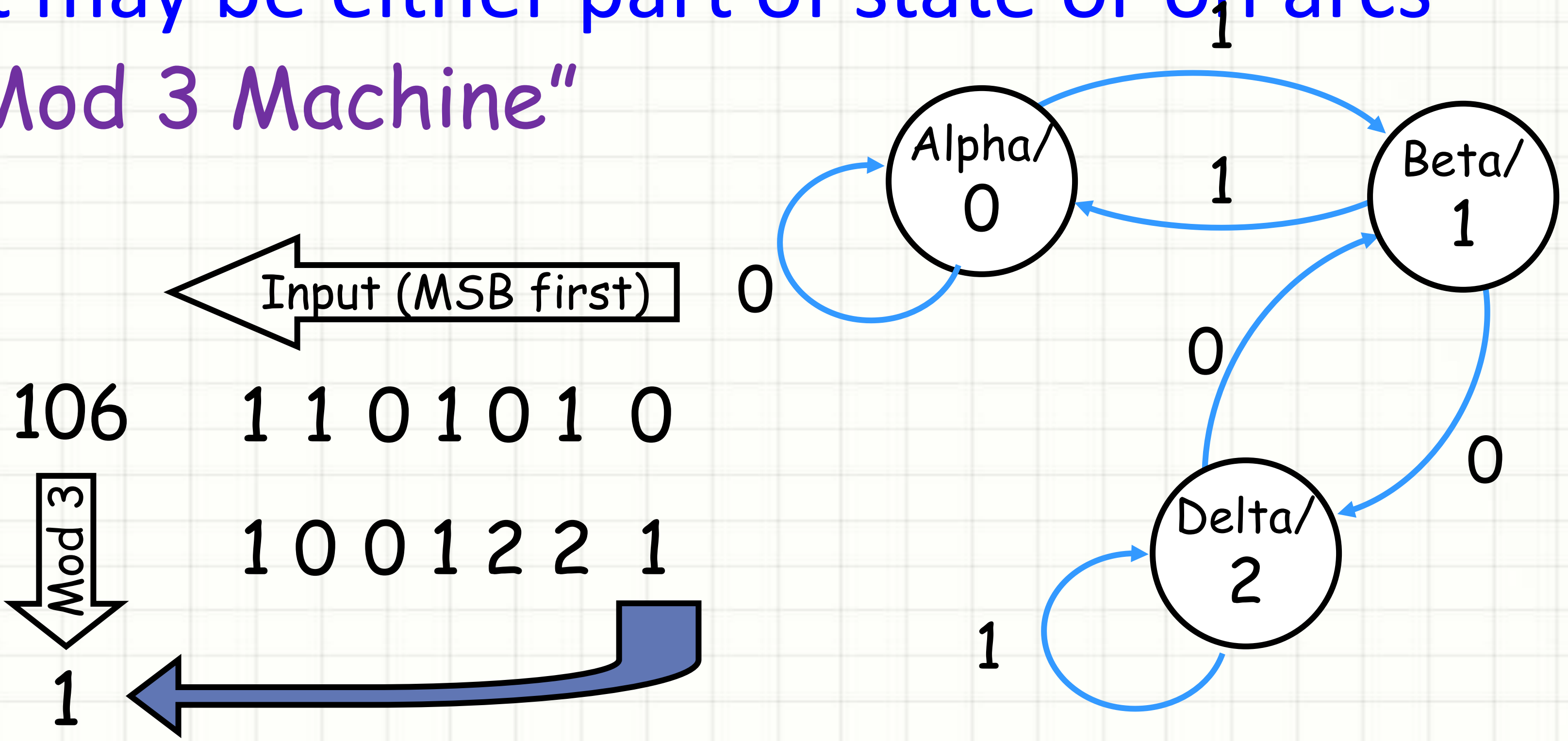




Finite State Machines:

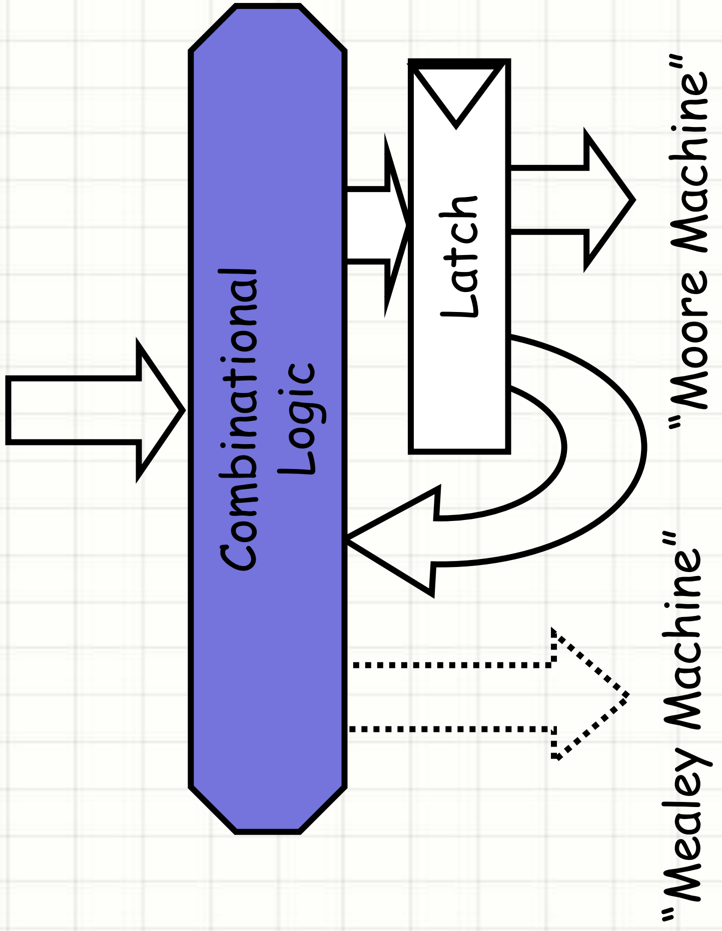
- System state is explicit in representation
- Transitions between states represented as arrows with inputs on arcs.
- Output may be either part of state or on arcs

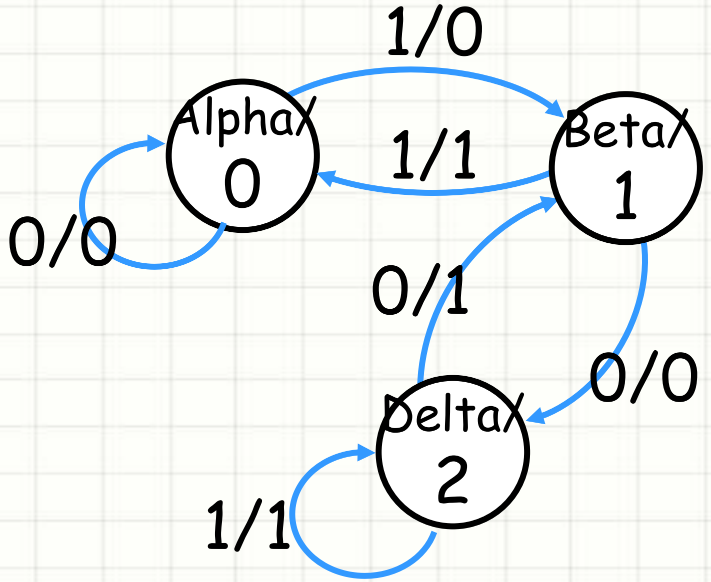
"Mod 3 Machine"



Indian Institute of Technology Delhi

Implementation as Comb logic + Latch

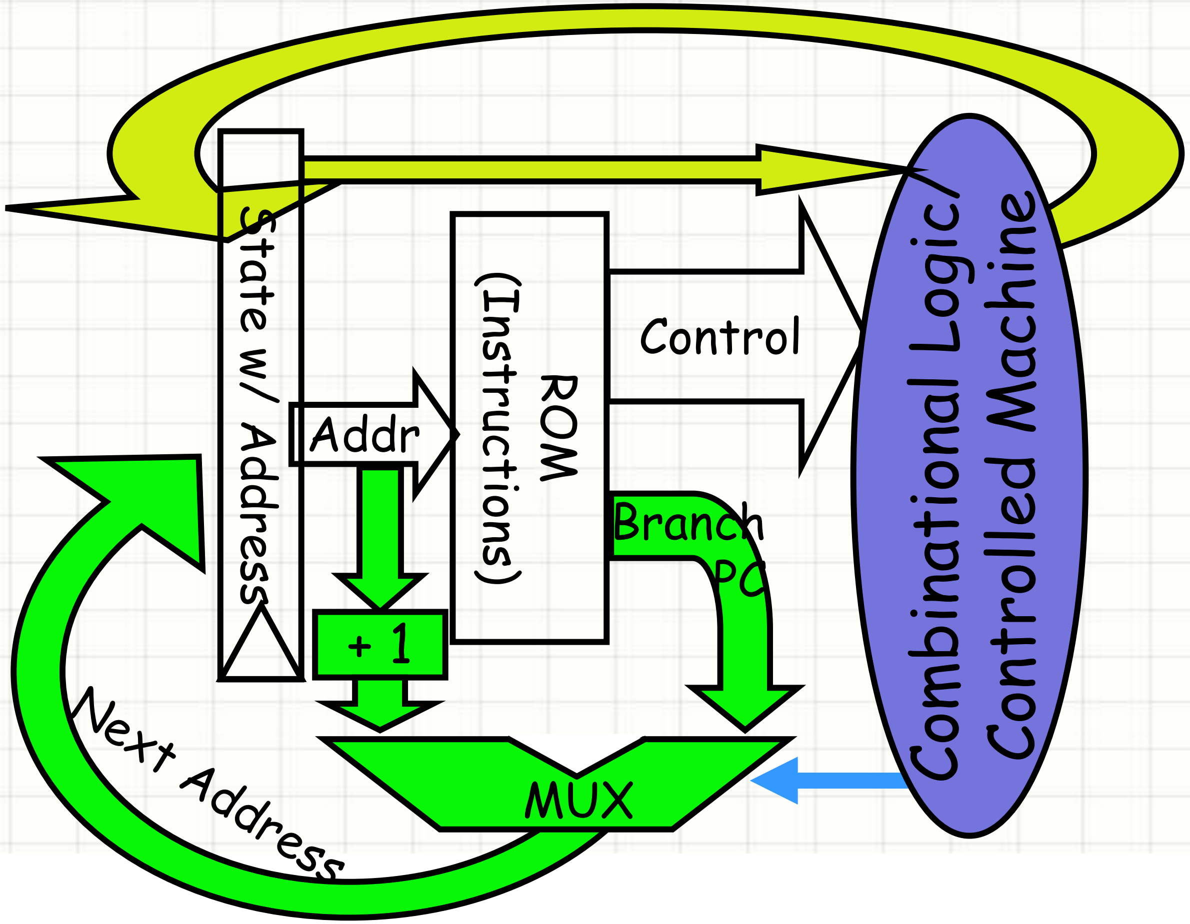




Input	$State_{old}$	$State_{new}$	Div
0	00	00	0
0	01	10	0
0	10	01	1
1	00	01	0
1	01	00	1
1	10	10	1

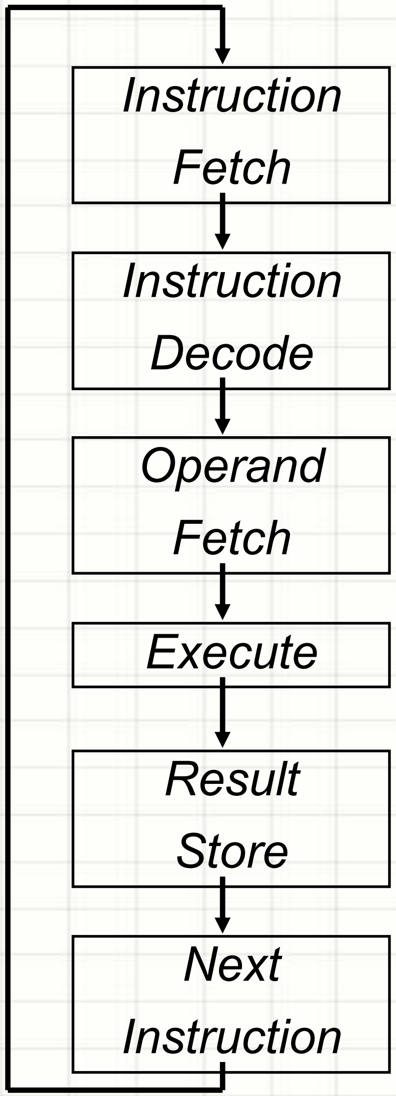
Microprogrammed Controllers

- State machine in which part of state is a "micro-pc".
  - Explicit circuitry for incrementing or changing PC
- Includes a ROM with "microinstructions".
  - Controlled logic implements at least branches and jumps



Instruction	Branch
0: forw 35	xxx
1: b_no_obstacles	000
2: back 10	xxx
3: rotate 90	xxx
4: goto	001

Fundamental Execution Cycle



Obtain instruction from program storage

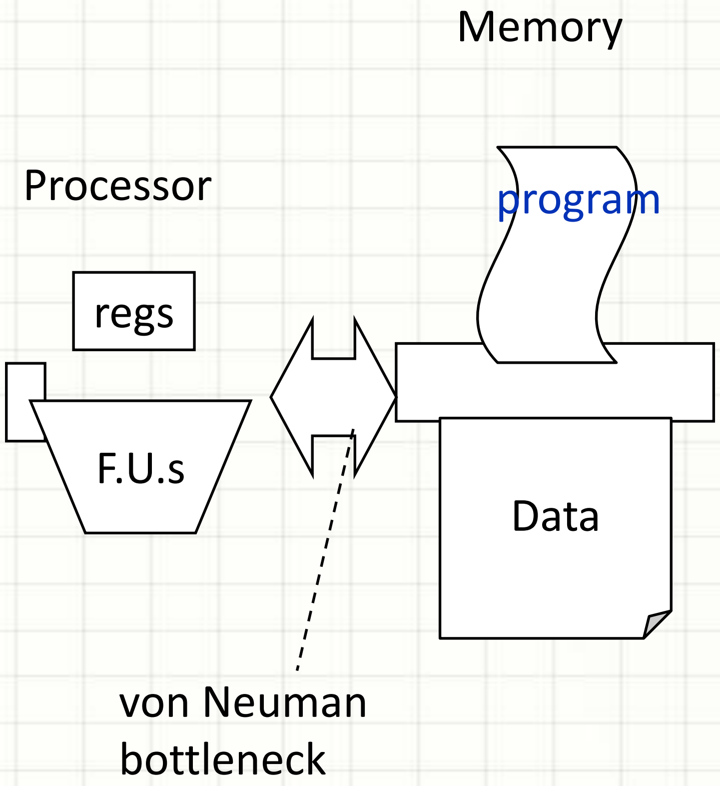
Determine required actions and instruction size

Locate and obtain operand data

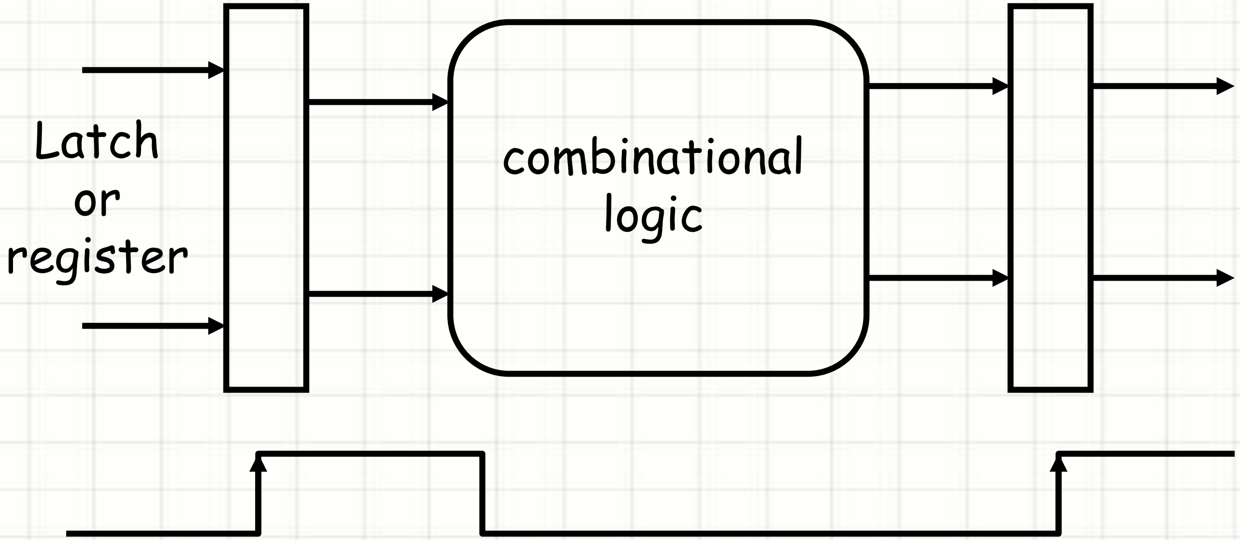
Compute result value or status

Deposit results in storage for later use

Determine successor instruction



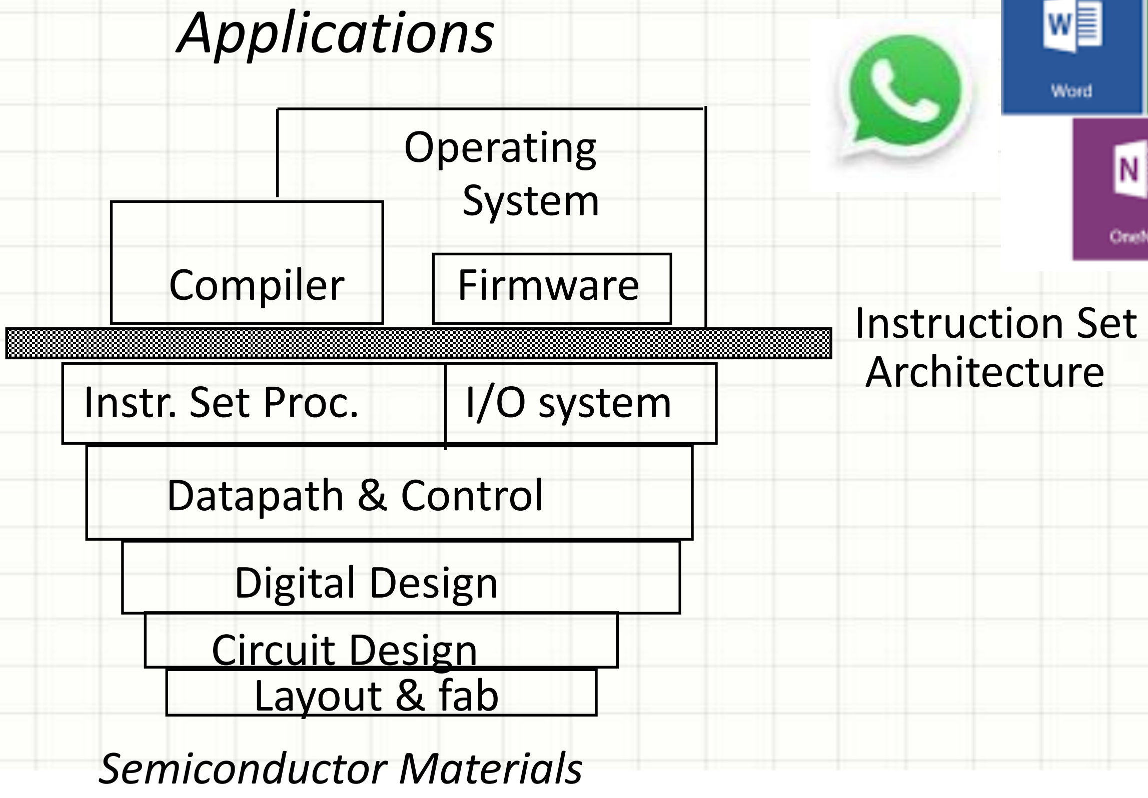
What's a Clock Cycle?

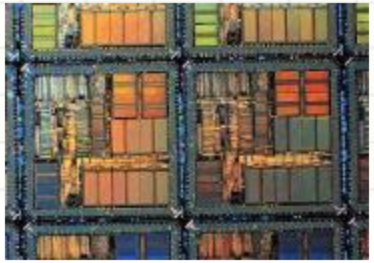


- 10 levels of gates
  - clock propagation, wire lengths, drivers

What is "Computer Architecture"?

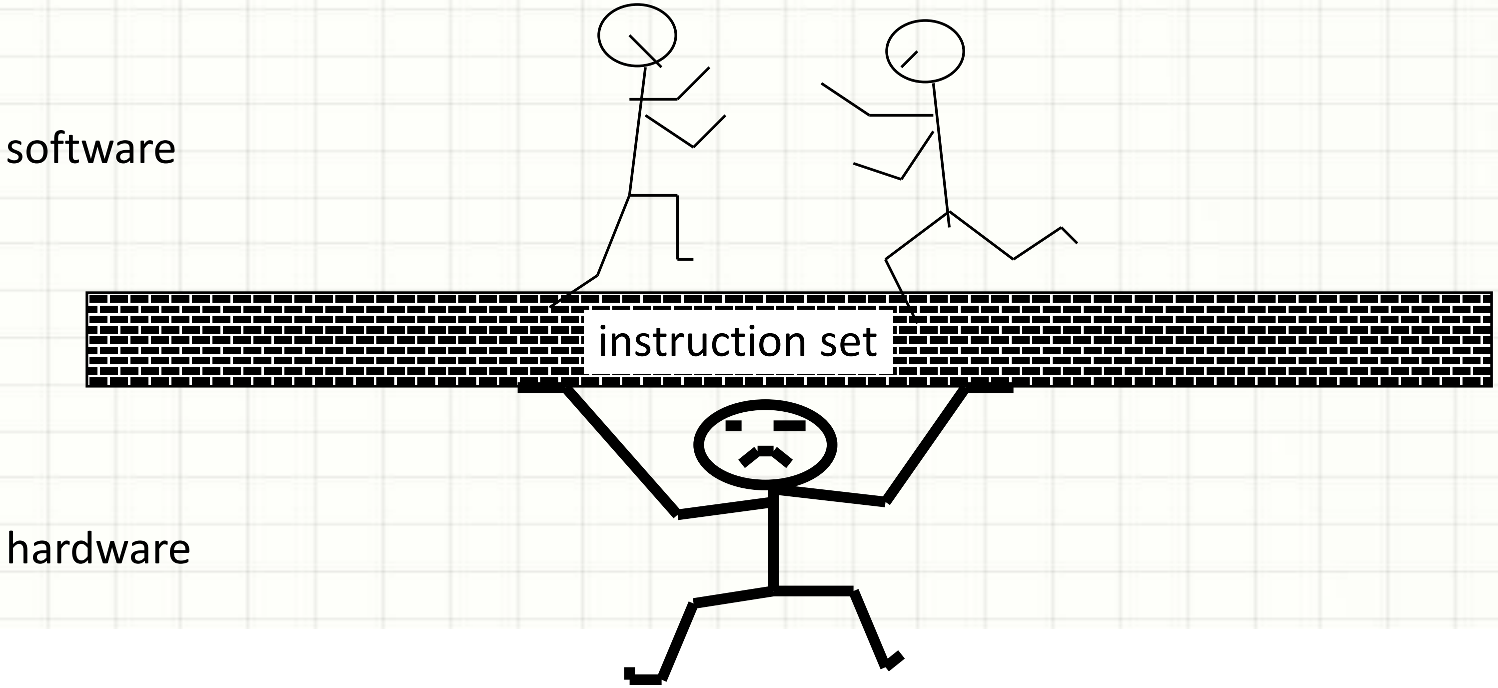
- Coordination of many *levels of abstraction*
- Under a rapidly *changing set of forces*
- Design, Measurement, *and* Evaluation



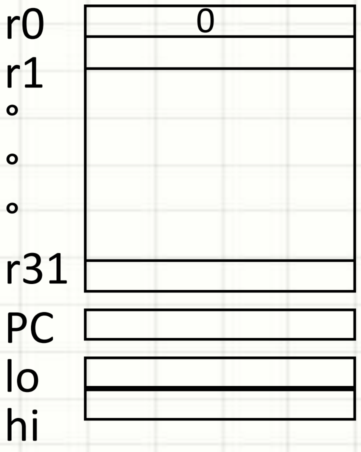


The Instruction Set: a Critical Interface

- Properties of a good abstraction
  - Lasts through many generations (portability)
  - Used in many different ways (generality)
  - Provides convenient functionality to higher levels
  - Permits an efficient implementation at lower levels



Example: MIPS R3000



Programmable storage

- 2^32 x bytes
- 31 x 32-bit GPRs (R0=0)
- 32 x 32-bit FP regs (paired DP)
- HI, LO, PC

Data types ?

Format ?

Addressing Modes?

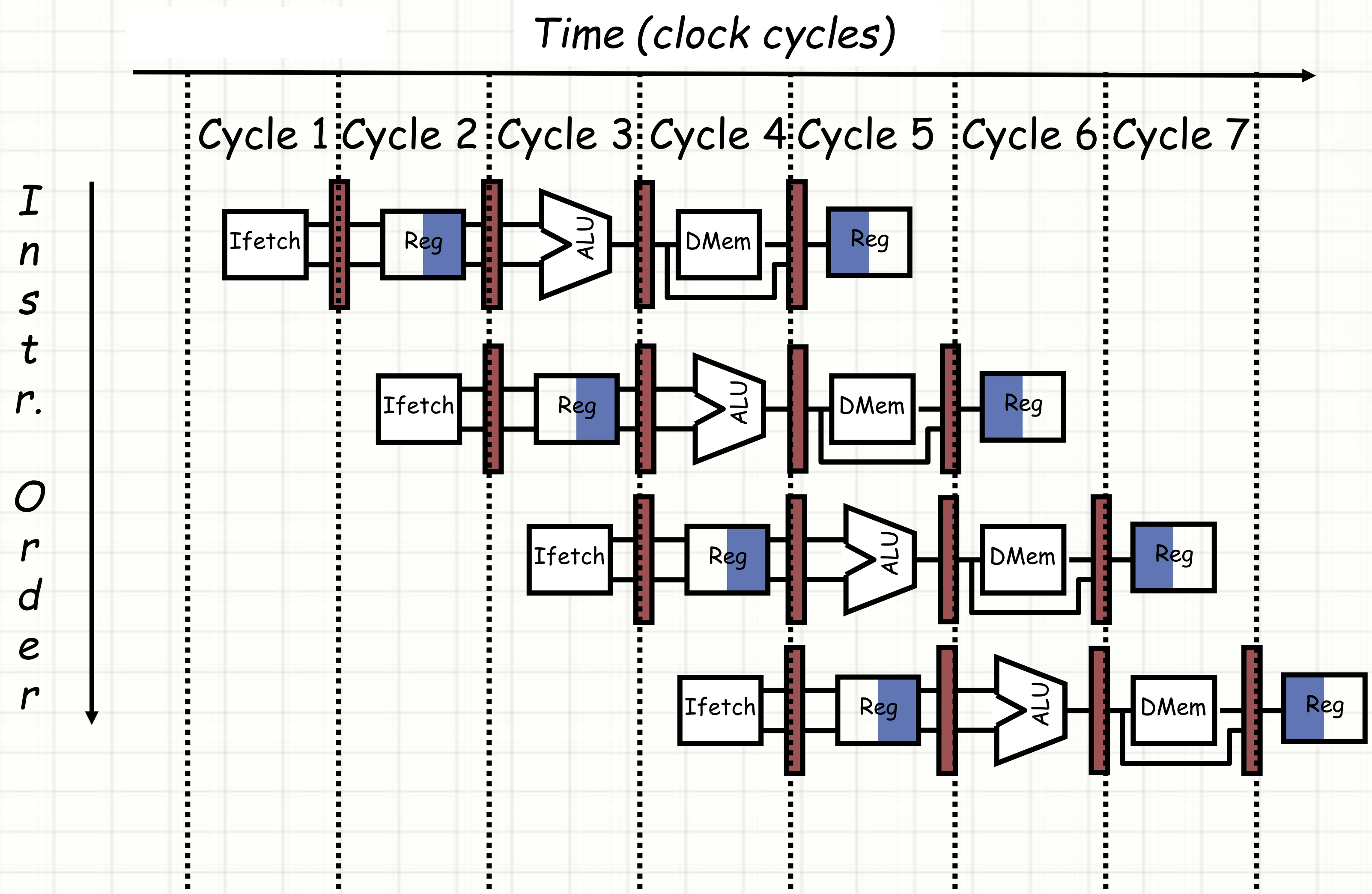
Arithmetic logical
- Add, AddU, Sub, SubU, And, Or, Xor, Nor, SLT, SLTU,
- AddI, AddIU, SLTI, SLTIU, AndI, OrI, XorI, *LUI*
- SLL, SRL, SRA, SLLV, SRLV, SRAV

Memory Access
- LB, LBU, LH, LHU, LW, LWL,LWR
- SB, SH, SW, SWL, SWR

Control
- J, JAL, JR, JALR
- BEq, BNE, BLEZ,BGTZ,BLTZ,BGEZ,BLTZAL,BGEZAL

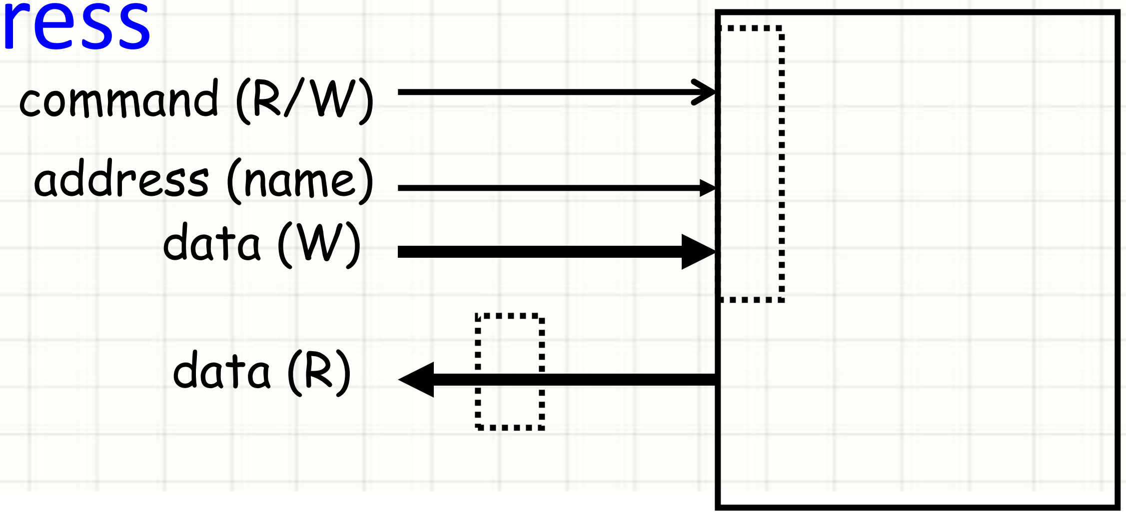
32-bit instructions on word boundary

Pipelined Instruction Execution



The Memory Abstraction

- Association of <name, value> pairs
  - typically named as byte addresses
  - often values aligned on multiples of size
- Sequence of Reads and Writes
- Write binds a value to an address
- Read of addr returns most recently written value bound to that address



Levels of the Memory Hierarchy

Capacity
Access Time
Cost

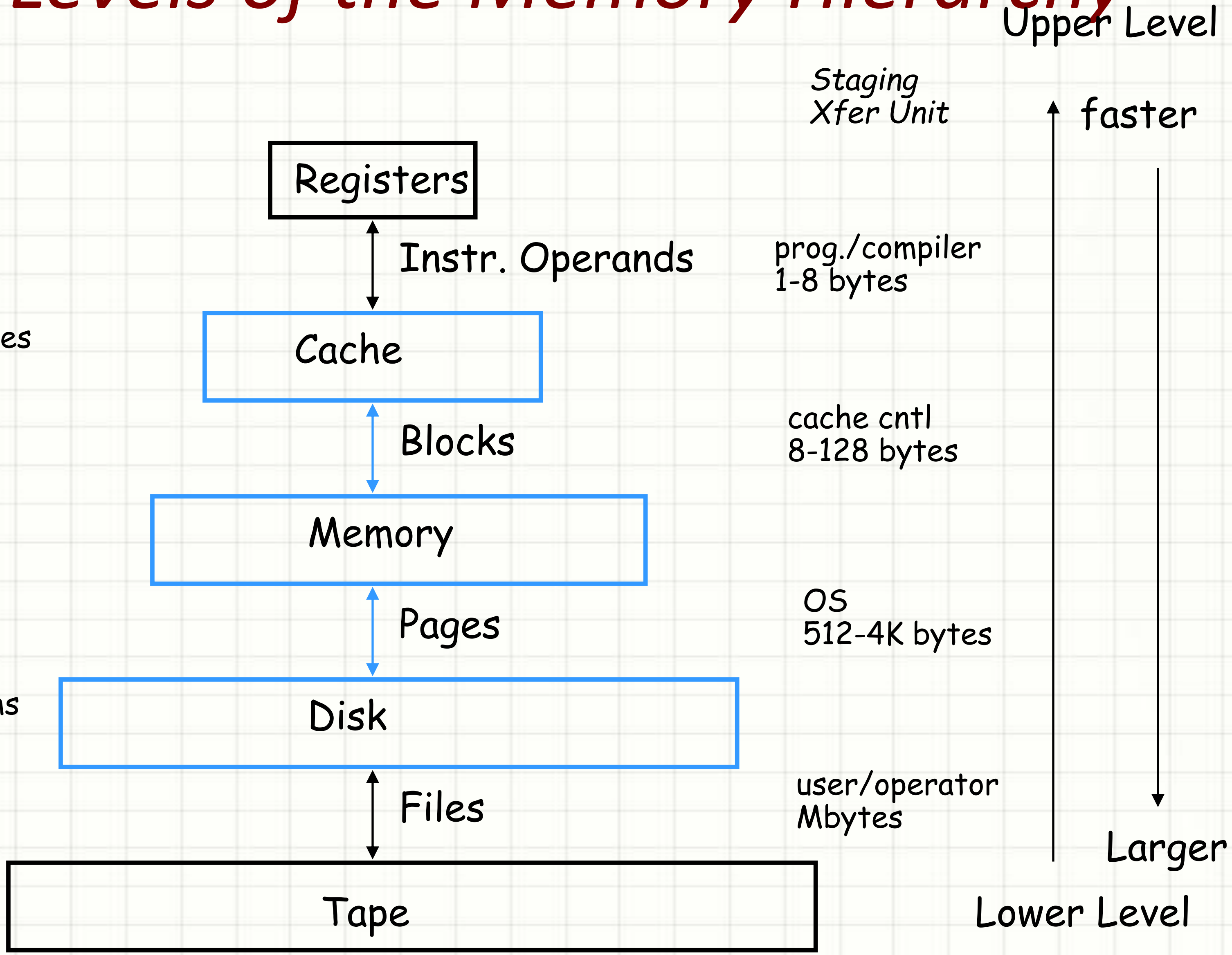
CPU Registers
100s Bytes
<< 1s ns

Cache
10s-100s K Bytes
~1 ns
$1s/ MByte

Main Memory
M Bytes
100ns- 300ns
$< 1/ MByte

Disk
10s G Bytes, 10 ms
(10,000,000 ns)
$0.001/ MByte

Tape
infinite
sec-min
$0.0014/ MByte



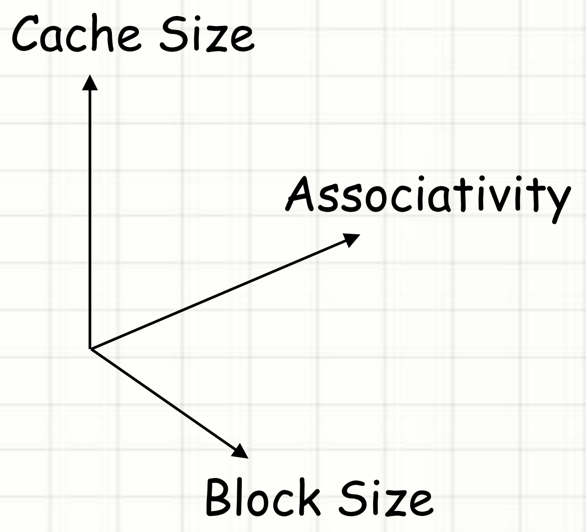
The Principle of Locality

Indian Institute of Technology Delhi

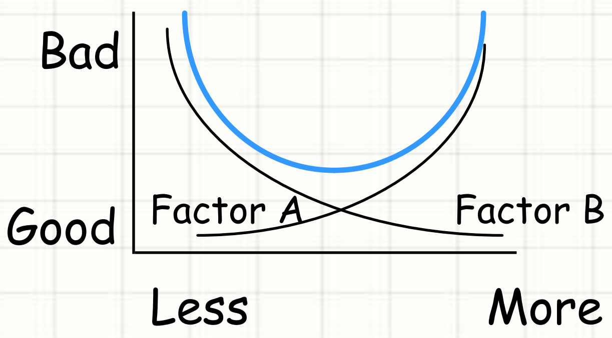
- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)
- Last 30 years, HW relied on locality for speed

The Cache Design Space

- Several interacting dimensions
  - cache size
  - block size
  - associativity
  - replacement policy
  - write-through vs write-back



- The optimal choice is a compromise
  - depends on access characteristics
    - workload
    - use (I-cache, D-cache, TLB)
  - depends on technology / cost

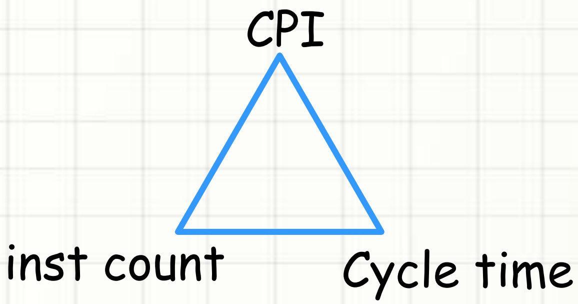


- Simplicity often wins

Indian Institute of Technology Delhi

Processor performance equation

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$



	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	(X)	
Inst. Set.	X	X	
Organization		X	X
Technology			X