



香港中文大學
計算機科學與工程學系

CSC 3420: Computer Systems Architecture

Introduction

Spring, 2009

Lee Kin Hong

Course Information

Lecturer : Lee Kin Hong (李健康) (SHB 1017; ext - 8435)

Tutors :

David Lam	林智輝	cflam
Peter Lo	羅良友	lylo
Ricky Wong	黃家駿	kcwong
Harry Yu	俞海樂	hlyu

Textbook : Patterson and Hennessy, “Computer Organisation & Design, The Hardware/Software Interface,” Morgan Kaufmann, third edition

Grading System : (No Assignment & mid-term tests)

Project:

Phase 1	(Assembly Language Application Prog. & Assembler)	12 %
Phase 2	(Single cycle CPU)	10 %
Phase 3	(Multi-cycle & pipelined CPU)	10 %
4 open book quizzes with late submissions allowed (7% each)		28 %
Final Examination :		40 %

To pass this course, you must get more than **50% of the overall** marks and score **not less than 35 % in the final** examination.

Rules & Disciplines

Rules & disciplines :

Be punctual for the class, try to minimise your disturbance if you are late. I may reject students who come in 15 minutes after the scheduled time.

Observe the university rules on plagiarism

<http://www.cuhk.edu.hk/policy/academichonesty/>

If you bring a mobile phone or a pager, you must switch it off. **5% will be deducted from your overall marks** for each time you disturb the class.

Acknowledgement : The lecture notes were adapted from the original presentations created by Professors Patterson and Hennessy.

- **Rapidly changing field:**
 - vacuum tube -> transistor -> IC -> VLSI
 - doubling every 1.5 years:
 - processor speed (Due to advances in technology and organisation)*
 - Memory Capacity (Due to advances in technology)*
- **Things you'll be learning:**
 - how computers work, a basic foundation
 - how to analyse their performance (or how they are misquoted!)
 - issues affecting modern processors (caches, pipelines, synchronisation)
- **Why learn this stuff?**
 - you want to build hardware & software people use (need performance)
 - you need to make a purchasing decision or offer “expert” advice
 - you want to call yourself a “computer scientist/engineer”

- In-depth understanding of the **inner-workings of modern computers**, their evolution, and **trade-offs** present at the hardware/software boundary.
 - Insight into **fast/slow operations** that are **easy/hard to implement**
 - Making application more efficient
- **Experience with the *design process*** in the context of a complex hardware design. Lessons learned will also **useful for large scale software design**
 - Functional Spec → Control & Datapath → Physical Implementation
- **Computer Designers' (Scientists') "Intellectual toolbox."**

What is a computer?

- **Components:**
 - **Processor (microprocessor)**
 - **Memory (DRAM, SRAM, Hard disk, CDROM)**
 - **Input (mouse, keyboard)**
 - **Output (display, printer)**
 - **Network (Internet, Wi Fi)**

- **Our primary focus: the processor (datapath and control)**
 - **implemented using millions of transistors**
 - **impossible to understand by looking at each transistor**
 - **We need ...**

Abstraction

- Delving into the depths reveals more information
- An abstraction **omits unneeded detail**, helps us to cope with complexity
- *What are some of the details that appear in these familiar abstractions?*

High-level
language program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

C compiler

Assembly
language program
(for MIPS)

```
swap:
  muli $2, $5, 4
  add $2, $4, $2
  lw $15, 0($2)
  lw $16, 4($2)
  sw $16, 0($2)
  sw $15, 4($2)
  jr $31
```

Assembler

Binary machine
language program (for
MIPS)

```
000000001010000010000000000000011000
0000000010001110000110000001000001
1000110001100010000000000000000000
1000110011110010000000000000000100
1010110011110010000000000000000000
1010110001100010000000000000000100
000000111110000000000000000001000
```

Computer Design

Instruction Set Design

- Machine Language
- Compiler View
- Computer Architecture
- Instruction Set Processor

- Building Architect

Computer Hardware Design

- Machine Implementation
- Logic Designer's View
- Processor Architecture
- Computer Organisation

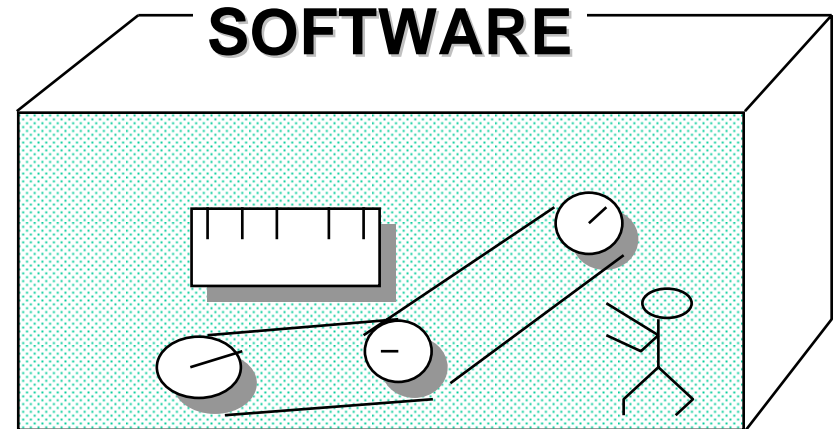
- Construction Engineer

Instruction Set Architecture

- ... the attributes of a [computing] system as seen by the programmer, i.e. the conceptual structure and functional behaviour, as distinct from the organisation of the data flows and controls the logic design, and the physical implementation.

Amdahl, Blaaw, and Brooks, 1964

- Organisation of Programmable Storage
- Data Types & Data Structures:
Encodings & Representations
- Instruction Formats
- Instruction (or Operation Code) Set
- Modes of Addressing and Accessing Data
Items and Instructions
- Exceptional Conditions



- **A very important abstraction**
 - interface between hardware and low-level software
 - standardises instructions, machine language bit patterns, etc.
 - advantage: *different implementations of the same architecture*
 - disadvantage: *sometimes prevents using new innovations*
- **Modern instruction set architectures:**
 - 80x86/Pentium/K6, PowerPC, DEC Alpha, MIPS, SPARC, HP
- ***True or False: Binary compatibility is extraordinarily important?***

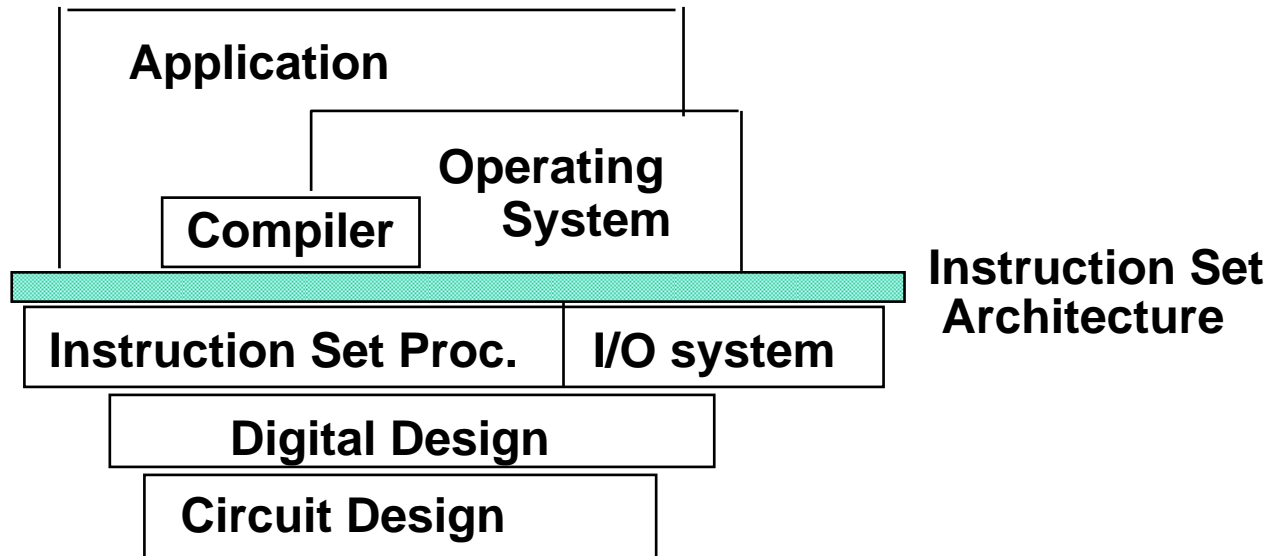
Logic Designer's View

FUs & Interconnect

- **Capabilities & Performance Characteristics of Principal Functional Units**
e.g. **Registers, ALU, Shifters, Logic Units, etc.**
- **Ways in which these components are interconnected**
- **Nature of information flows between components**
- **Logic and means by which such information flow is controlled.**
- **Choreography [舞蹈 (動作) 編排] of FUs to realise the ISA**
- **Best summarised using **Register Transfer Level Description****

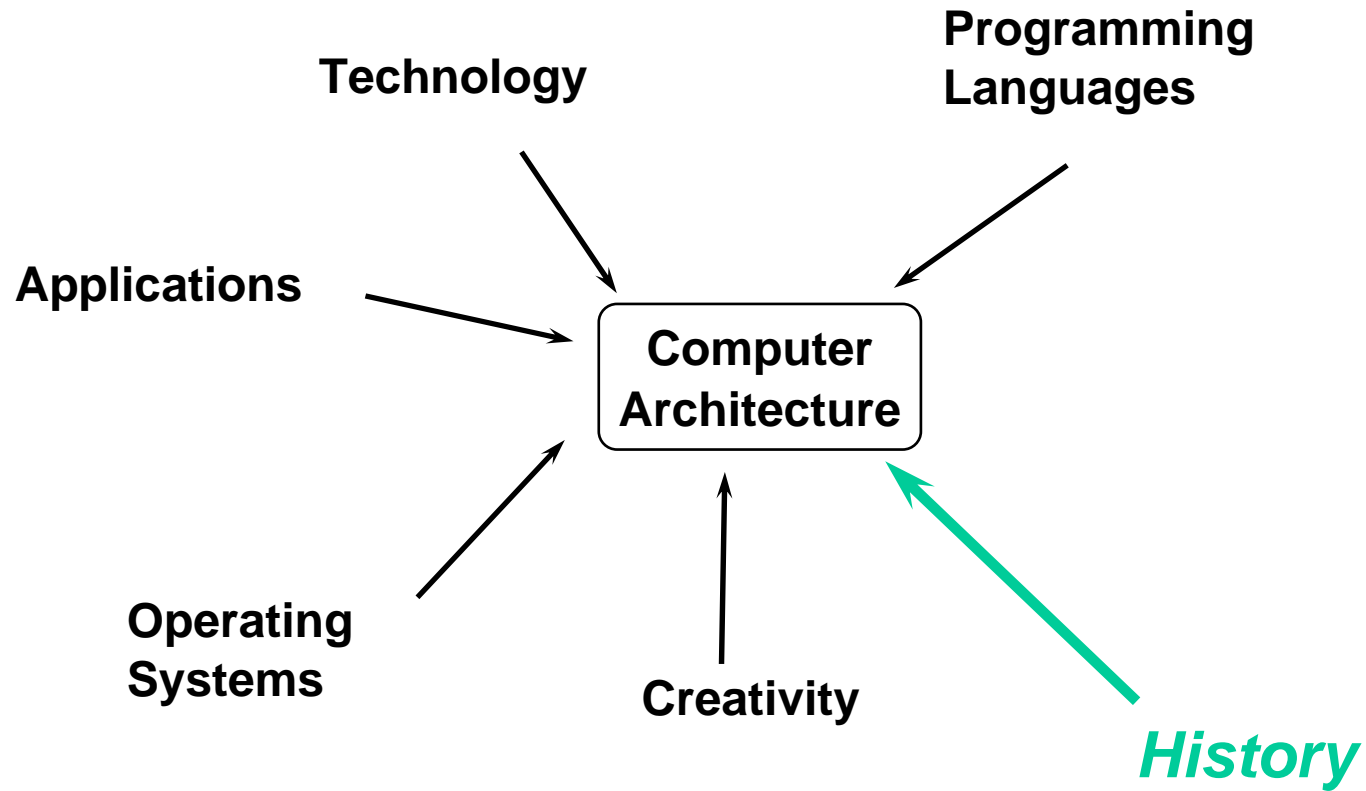
What is "Computer Architecture?"

Co-ordination of *levels of abstraction*

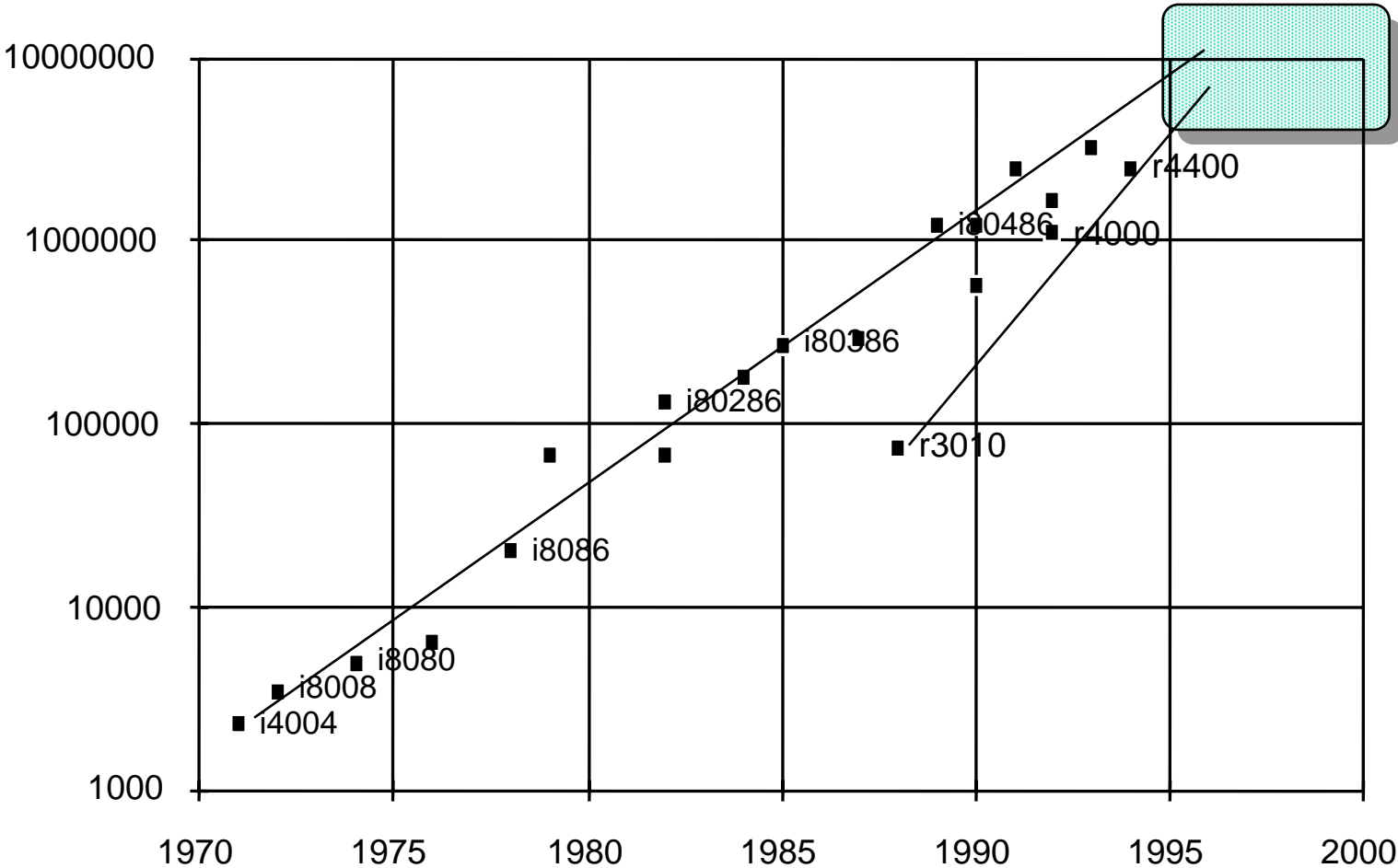


Under a set of rapidly changing *Forces*

Forces on Computer Architecture

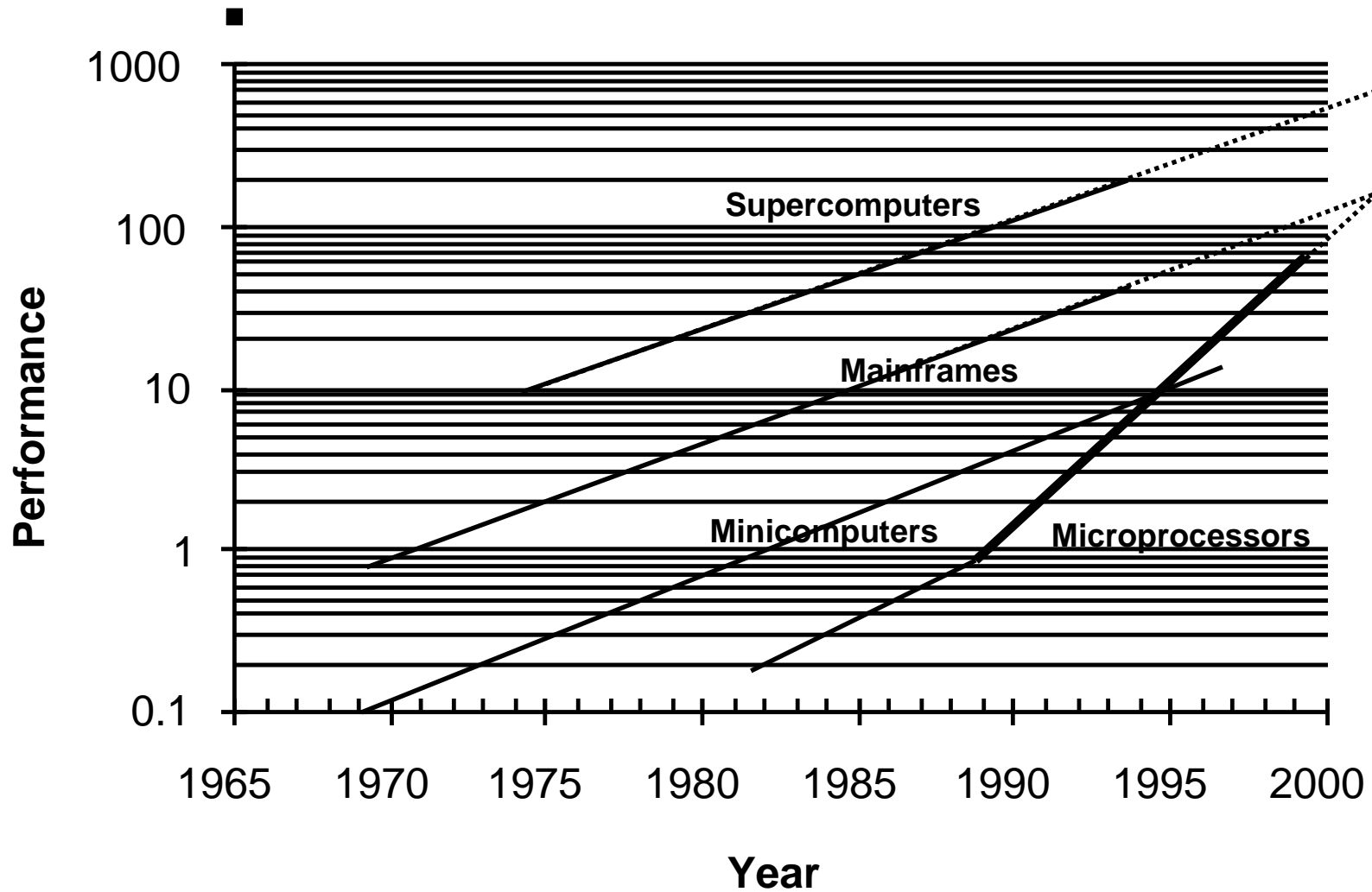


Technology: Microprocessor Logic Density

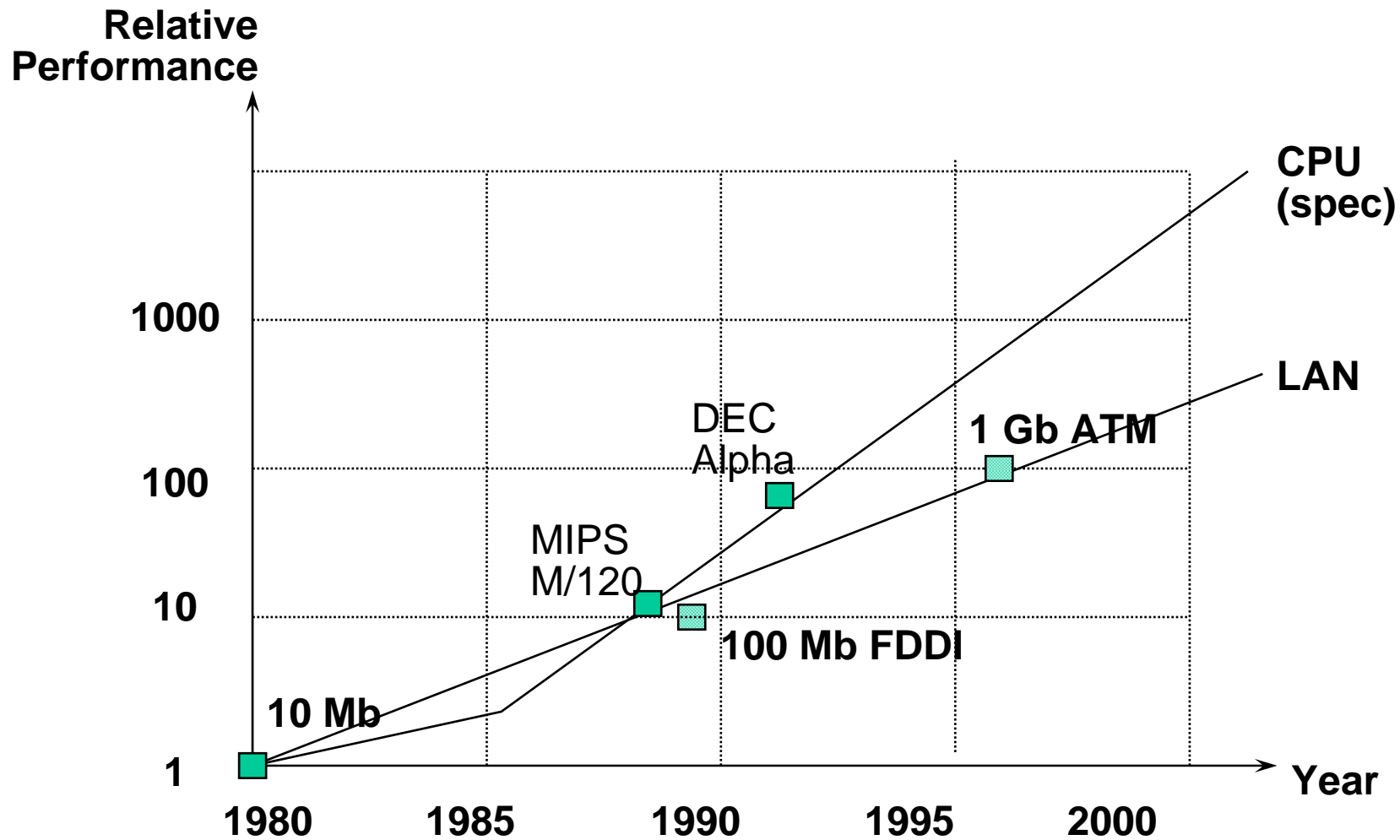


Memory: 4x every 3 years

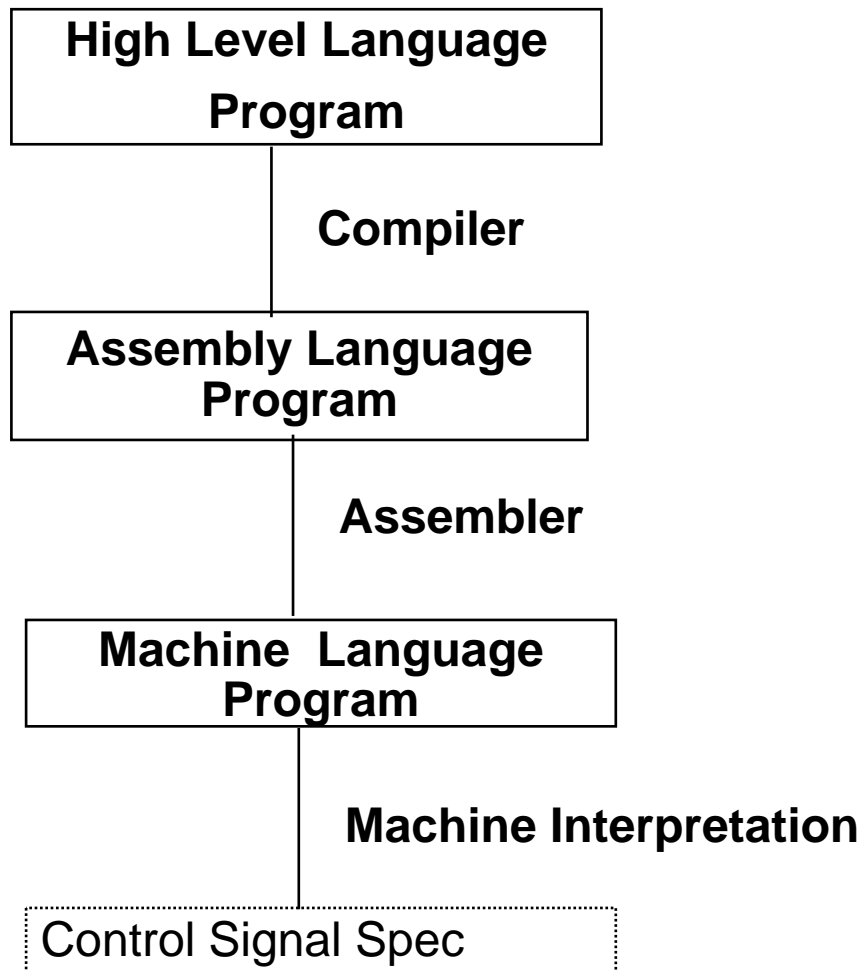
Performance Trends



CPU and LAN Performance



Levels of Representation



```
temp = v[k];  
v[k] = v[k+1];  
v[k+1] = temp;
```

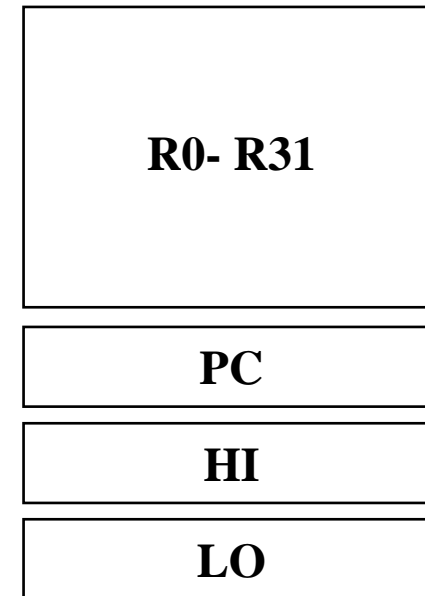
```
lw    $15, 0($2)  
lw    $16, 4($2)  
sw    $16, 0($2)  
sw    $15, 4($2)
```

MIPS R3000 Instruction Set Architecture

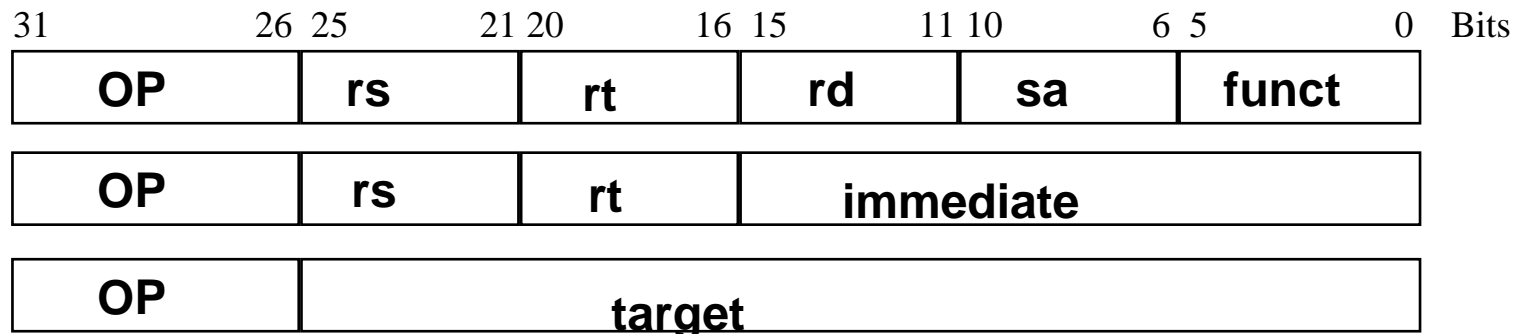
- **Instruction Categories**

- **Load/Store**
- **Computational**
- **Jump and Branch**
- **Floating Point**
 - coprocessor
- **Memory Management**
- **Special**

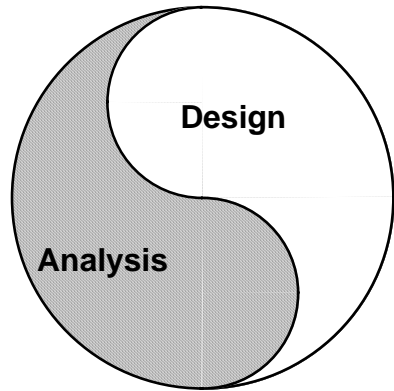
- **Registers**



Instruction Format

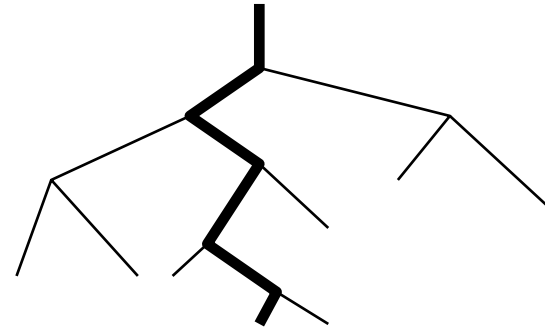


Measurement and Evaluation

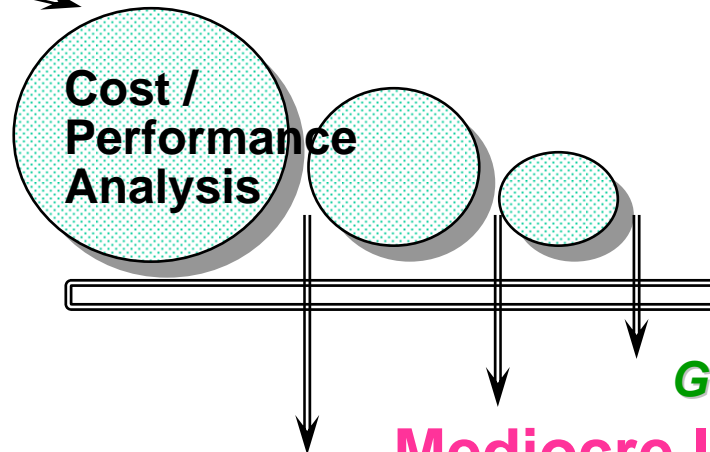
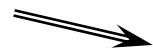


Architecture Design is an iterative process

- searching the space of possible designs
- at all levels of computer systems



Creativity



Bad Ideas

Mediocre Ideas

Good Ideas

Computer Design

Instruction Set Design

- **Compiler View**
- **Machine Language**
- **Computer Architecture**
- **Instruction Set Processor**

- **Building Architect**

Computer Hardware Design

- **Logic Designer's View**
- **Machine Implementation**
- **Processor Architecture**
- **Computer Organisation**

- **Construction Engineer**

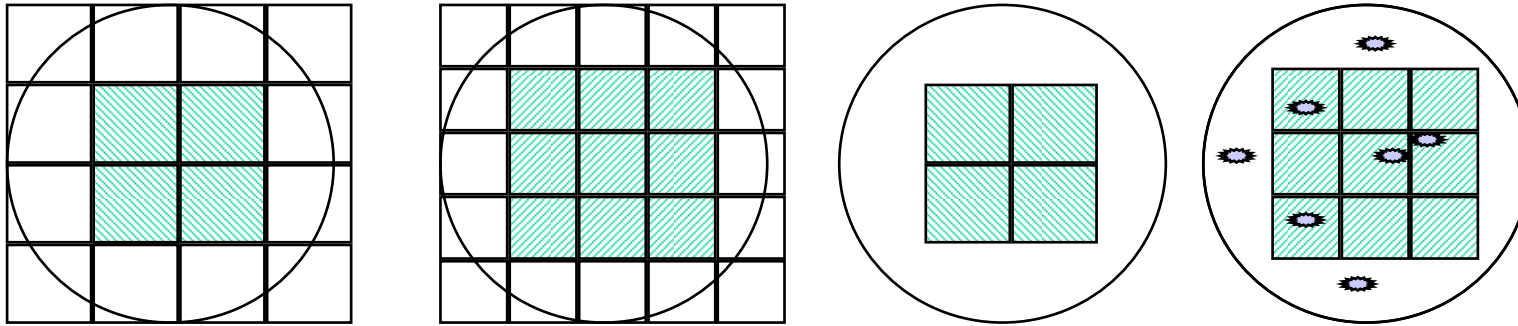
Few people design computers!

Very few design instruction sets!

Many people design computer components.

Very many people are concerned with computer function, in detail.

Integrated Circuits Costs



$$\text{Die Cost} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} * \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times (\text{Wafer diameter}/2)^2}{\text{Die Area}} - \frac{\pi \times \text{Wafer Diameter}}{\sqrt{2 \times \text{Die Area}}} - \text{Test dies per wafer}$$

No of testing dies for characteristics testing



No. of dies along the edge ~ circumference/diagonal of die

$$\text{Die yield} = \text{Wafer yield} \times \left\{ 1 + \frac{\text{Defects per unit area} \times \text{Die area}}{\alpha} \right\}^{-\alpha}$$

No of masking levels critical to die yield;
Depends on the manufacturing process

Real World Examples

<i>Chip</i>	<i>Metal layers</i>	<i>Line width</i>	<i>Wafer cost</i>	<i>Defect /cm²</i>	<i>Area mm²</i>	<i>Dies/wafer</i>	<i>Yield</i>	<i>Die Cost</i>
386DX	2	0.90	\$900	1.0	43	360	71%	\$4
486DX2	3	0.80	\$1200	1.0	81	181	54%	\$12
PowerPC 601	4	0.80	\$1700	1.3	121	115	28%	\$53
HP PA 7100	3	0.80	\$1300	1.0	196	66	27%	\$73
DEC Alpha	3	0.70	\$1500	1.2	234	53	19%	\$149
SuperSPARC	3	0.70	\$1700	1.6	256	48	13%	\$272
Pentium	3	0.80	\$1500	1.5	296	40	9%	\$417

From "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993, p. 15

Die cost are getting higher but yield is getting worst.

Other Costs

$$IC\ Cost = \frac{Die\ Cost + Testing\ Cost + Packaging\ Cost}{Final\ Test\ Yield}$$

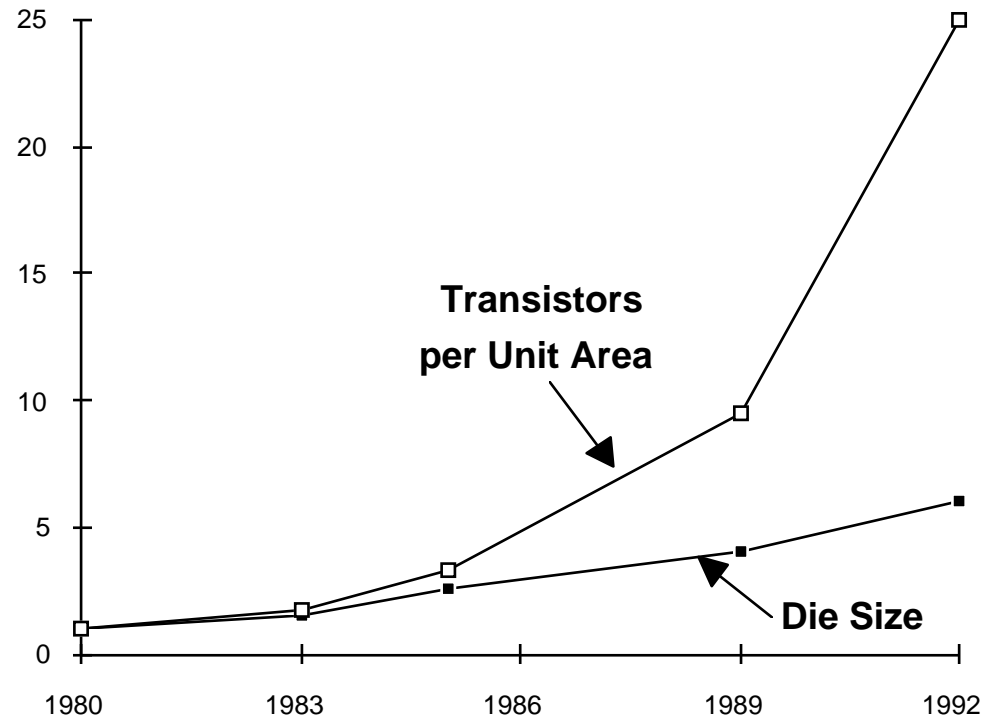
$$Cost\ of\ testing\ die = \frac{Cost\ of\ testing\ per\ hour \times Average\ die\ test\ time}{Die\ yield}$$

Packaging Cost: depends on pins, heat dissipation

Chip	Die	Package			Test & Assembly	Total Cost
		pins	type	cost		
386DX	\$4	132	QFP	\$1	\$4	\$9
486DX2	\$12	168	PGA	\$11	\$12	\$35
PowerPC 601	\$53	304	QFP	\$3	\$21	\$77
HP PA 7100	\$73	504	PGA	\$35	\$16	\$124
DEC Alpha	\$149	431	PGA	\$30	\$23	\$202
SuperSPARC	\$272	293	PGA	\$20	\$34	\$326
Pentium	\$417	273	PGA	\$19	\$37	\$473

CMOS improvements

- **Die size 2X / 3 years; Line widths halve / 7 years**



Technology Trends

	Capacity	Speed
Logic	2x in 3 years	2x in 3 years
DRAM	4x in 3 years	1.4x in 10 years
Disk	4x in 3 years	1.4x in 10 years

- **All computers consist of five components**
 - **Processor: (1) datapath and (2) control**
 - **(3) Memory**
 - **(4) Input devices and (5) Output devices**
- **Not all “Memory” are created equally**
 - **Cache: fast (expensive) memory are placed closer to the processor**
 - **Main memory: less expensive memory--we can have more**
- **Input and output (I/O) devices has the messiest organisation**
 - **Wide range of speed: graphics vs. keyboard**
 - **Wide range of requirements: speed, standard, cost ... etc.**
 - **PC industry has imposed some sort of standards**
 - **Cost is falling fast too**

Where we are heading?

- **Performance issues** (Chapter 2): *vocabulary and motivation*
- A specific **instruction set architecture** (Chapter 3)
- Arithmetic and how to **build an ALU** (Chapter 4)
- **Constructing a processor** to execute our instructions (Chapter 5)
- **Pipelining** to improve performance (Chapter 6)
- **Memory:** caches and virtual memory (Chapter 7)

- **Key to a good grade: reading!**