Lecture 1: Introduction

COS 375 / ELE 375

Computer Architecture and Organization

Princeton University
Fall 2015

Prof. David August

The Usual Suspects

Me: Prof. David August, 221 CS Building, august@
Office Hours: M/W after class and by appointment

TAs: Bochao Wang, E-Quad C-319B, bochaow@
Office Hours: Th/F 10:30-11:30AM

Debajit Bhattacharya, E-Quad C-319D, dbhattac@
Office Hours: T/Th 3-4PM

Hansen Zhang, 241 CS Building, hansenz@
Office Hours: M/W 3-4PM

Course Objectives

- Enable you to design and build a computer
  - Get a lump of matter to do your bidding
  - Understand and evaluate tradeoffs in design
- Appreciate theory vs. practice
  - Become a better implementer of algorithms
- Learn assembly/machine language programming
  - Essential for OS and Compiler Work
  - Essential for understanding processor design
- Understand modern Computer Organization
  - High-level languages -> execution on physical material
- Help you to revolutionize computing
  - Discuss the latest research
  - Contribute some of our own?

Course Topics

1. Performance Evaluation
- Measures of performance
- Benchmarks and metrics

2. Instruction Set Architecture
- Instruction formats & semantics
- Addressing modes

3. Machine Arithmetic
- ALU design
- Integer multiplication & division
- Floating-point arithmetic

4. Processor Design
- Datapath design
- Instruction exec. & sequencing
- Hardwired & microcode control
- Pipelining

5. Hardware Design Languages
- Design with a Verilog
- Modeling and simulation

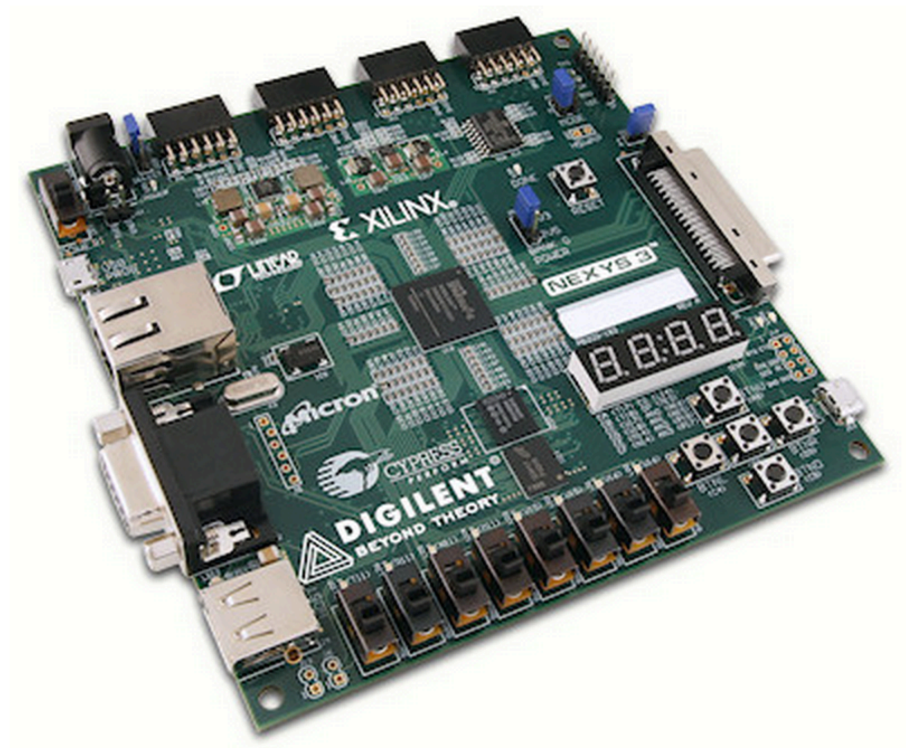
6. Memory Hierarchy
- Cache design & evaluation
- Virtual addressing
- Performance evaluation

7. Input/Output
- Types of I/O devices
- Device access and interface
- Device control
- I/O performance

8. Multiprocessor
- Interconnection networks
- Programming issues

The Course Project

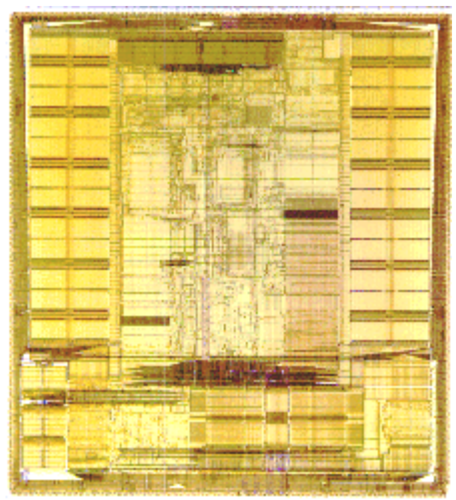
- Groups of 4 students – Sign up in September
- Work can be done anywhere
- Project consists of two parts
  - ARM-based Instructions Set Simulator (in C)
  - Implementation of processor onto an FPGA (in Verilog)



Pick a number 1,2,3

1

2



3

If the random number is the picture of a "processor", then we have a quiz.

Quizzes/Homework

Quizzes

- Chance quiz at the beginning of one class each week
- Not intended as a scare tactic – liberally graded
- Helps assess progress of class
- Just one question usually

Homework

- 4 homework sets
- Questions resemble exam questions

Exams

- Exams cover concepts presented in the lecture material, homework assignments, and/or required readings
- Sheet of paper allowed

Midterm Exam

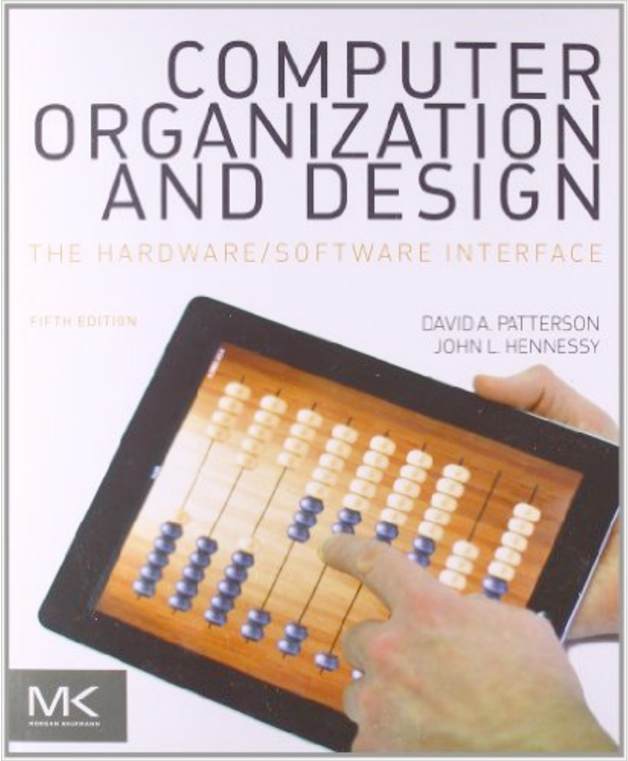
- Wednesday before Fall Break
- In class

Final Exam

- The final exam will be cumulative, three hours in length
- Time/Place determined by the Registrar

Required Reading

- *Computer Organization and Design: The Hardware/Software Interface*
  Fifth Edition
  David Patterson and John Hennessy



- Course Web Page – Off of CS page
  - Lecture Notes
  - Project Material
  - Homework Assignments
  - Course Announcements

Participation

Negatives

- Class disruptions (snoring, reading newspaper, etc.)
- Mistreatment of TAs

Positives

- Contribute questions and comments to class
- Participate in discussions
- Feedback
- Stop by office hours to introduce yourself

Grading

Project	40%
Midterm	15%
Final	25%
Quizzes	EXTRA CREDIT
Participation	5%
Homework	15% (best 3 of 4)

Who Am I?

At Princeton (Computer Science, 1999-Present):

- Professor
- Compiler and computer architecture research
- Liberty Research Group

Education (Ph.D. in 2000):

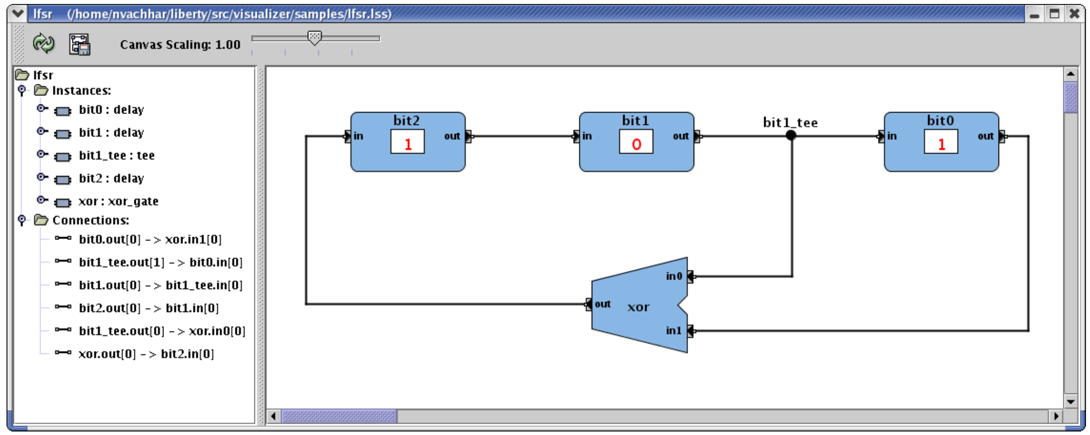
- Ph.D. Electrical Engineering from University of Illinois
- Thesis Topic: Predicate Optimization
- The IMPACT Compiler Research Group

Our Pledge to You

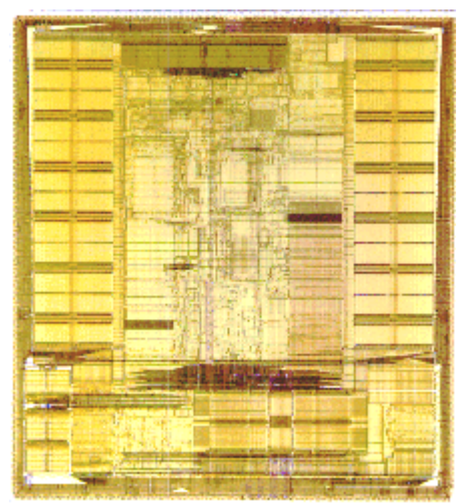
- Lectures only as long as necessary!
- Quick response to questions and issues
- Reasonable late policy
  - Up to 3 days late for any single assignment without penalty
  - Up to 7 days late total across all assignments
  - Contact me prior to deadline for special circumstances
- Fast turn-around on grading

END OF ADMINISTRATIVE STUFF

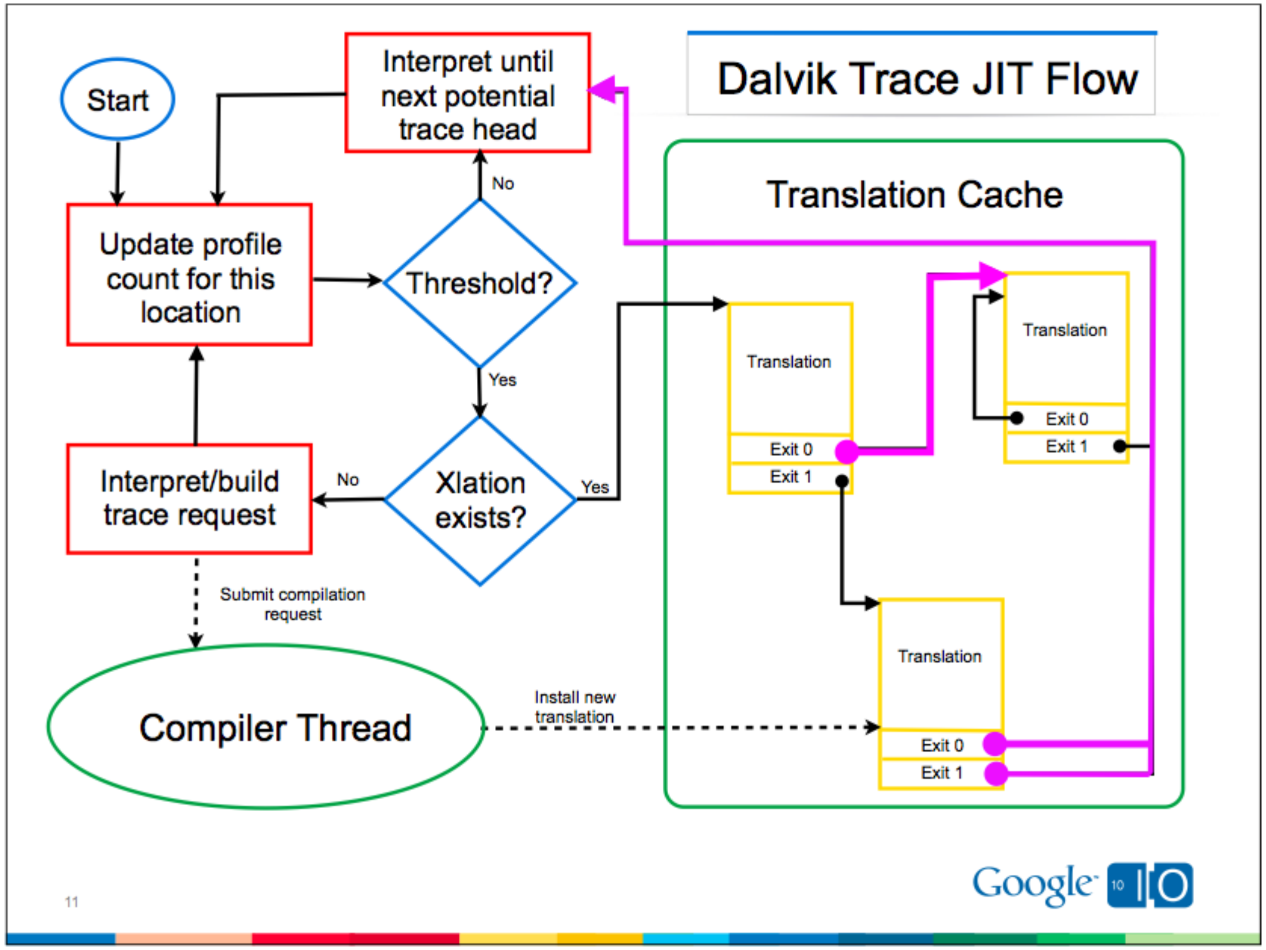
Why Computer Architecture?




```
module toplevel(clock,reset);
  input clock;
  input reset;

  reg flop1;
  reg flop2;

  always @ (posedge reset or posedge clock)
    if (reset)
      begin
        flop1 <= 0;
        flop2 <= 1;
      end
    else
      begin
        flop1 <= flop2;
        flop2 <= flop1;
      end
endmodule
```

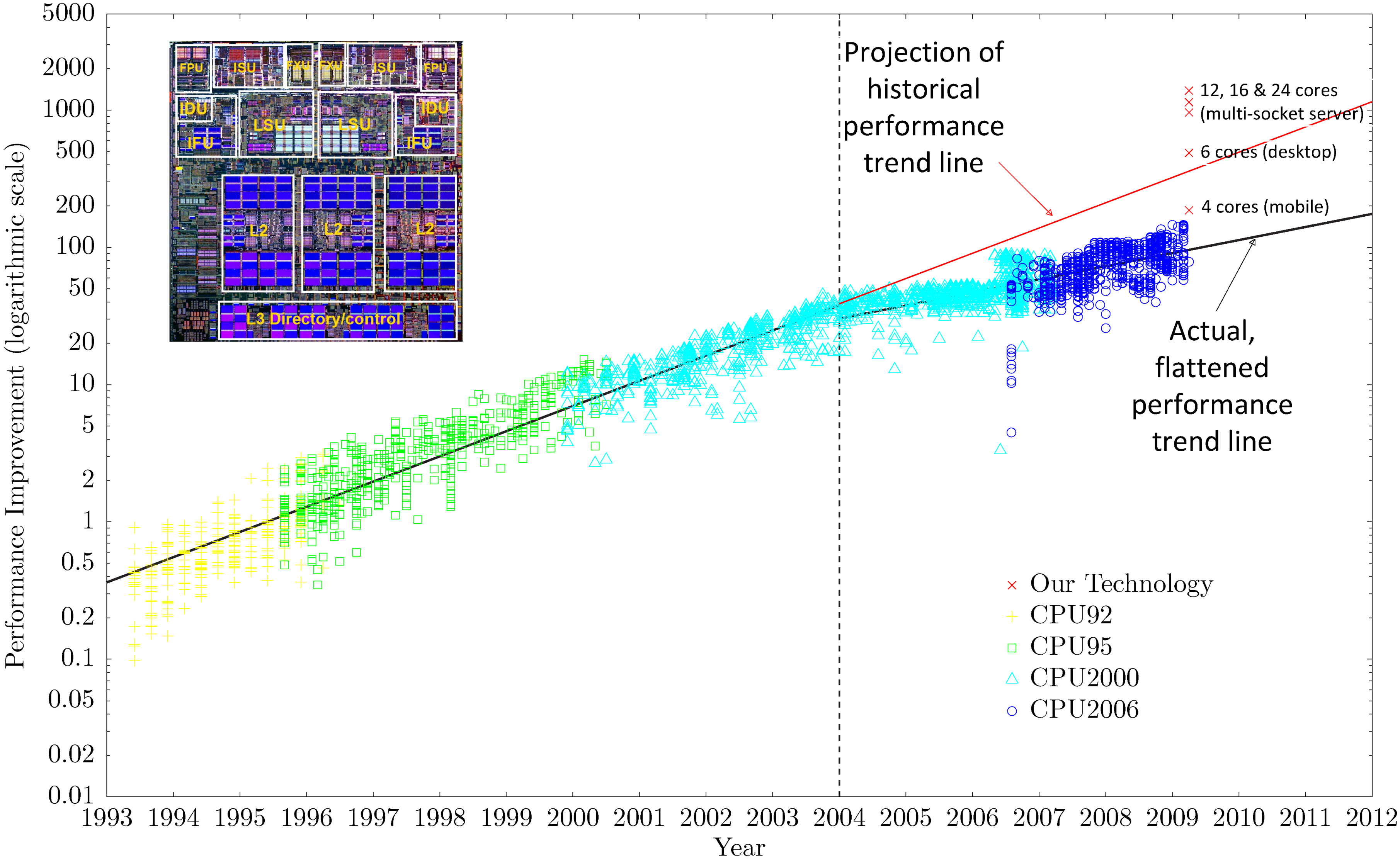


Why Computer Architecture?

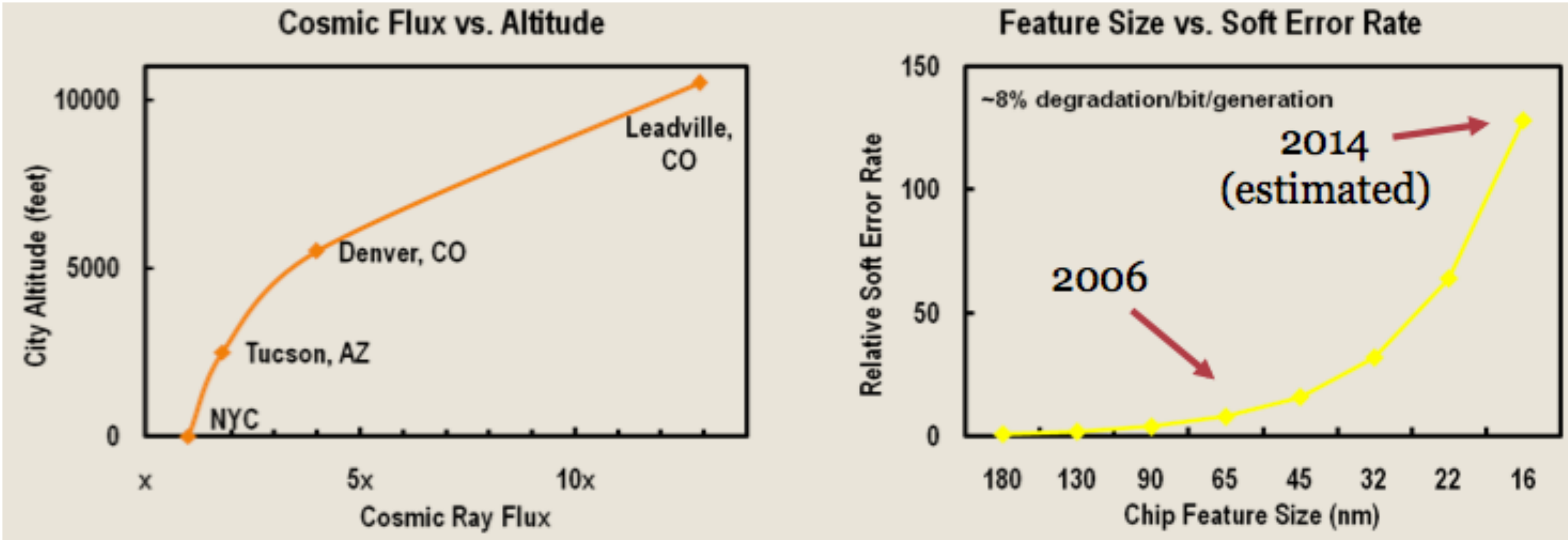


Why Computer Architecture?

Is computer architecture dead?



Why Computer Architecture?



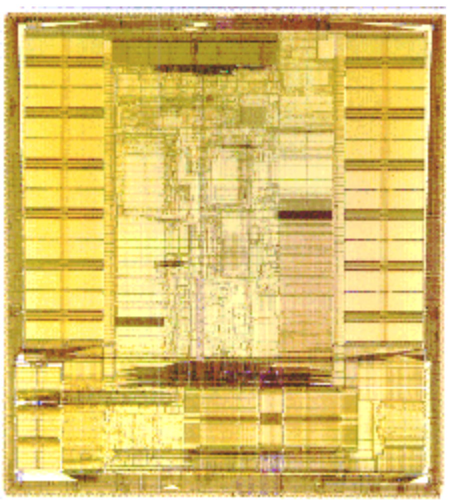
Princeton Research on Fault Tolerance wins CGO Test of Time Award

February 2, 2015

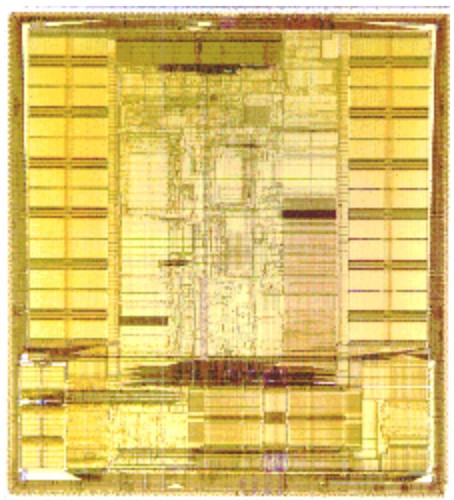
Every year, the International Symposium on Code Generation and Optimization (CGO) recognizes the paper appearing 10 years earlier that is judged to have had the most impact on the field over the intervening decade. This year at CGO 2015, the paper entitled "SWIFT: Software Implemented Fault Tolerance" by George A. Reis, Jonathan Chang, Neil Vachharajani, Ram Rangan, and David I. August won the award. The paper originally appeared at CGO 2005 and also won the best paper award that year at the conference. Congratulations to Princeton's Liberty Research Group for winning this prestigious award!

It's a Wednesday: Pick a number 1,2,3

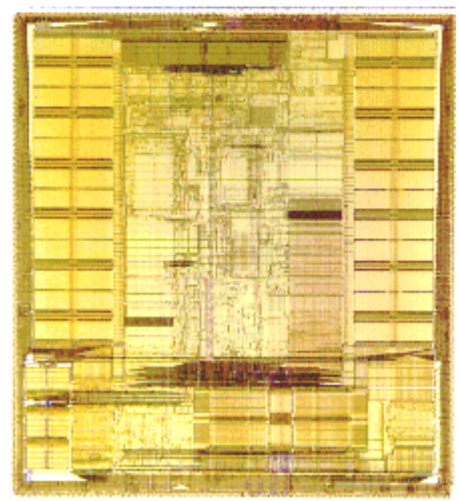
1



2



3



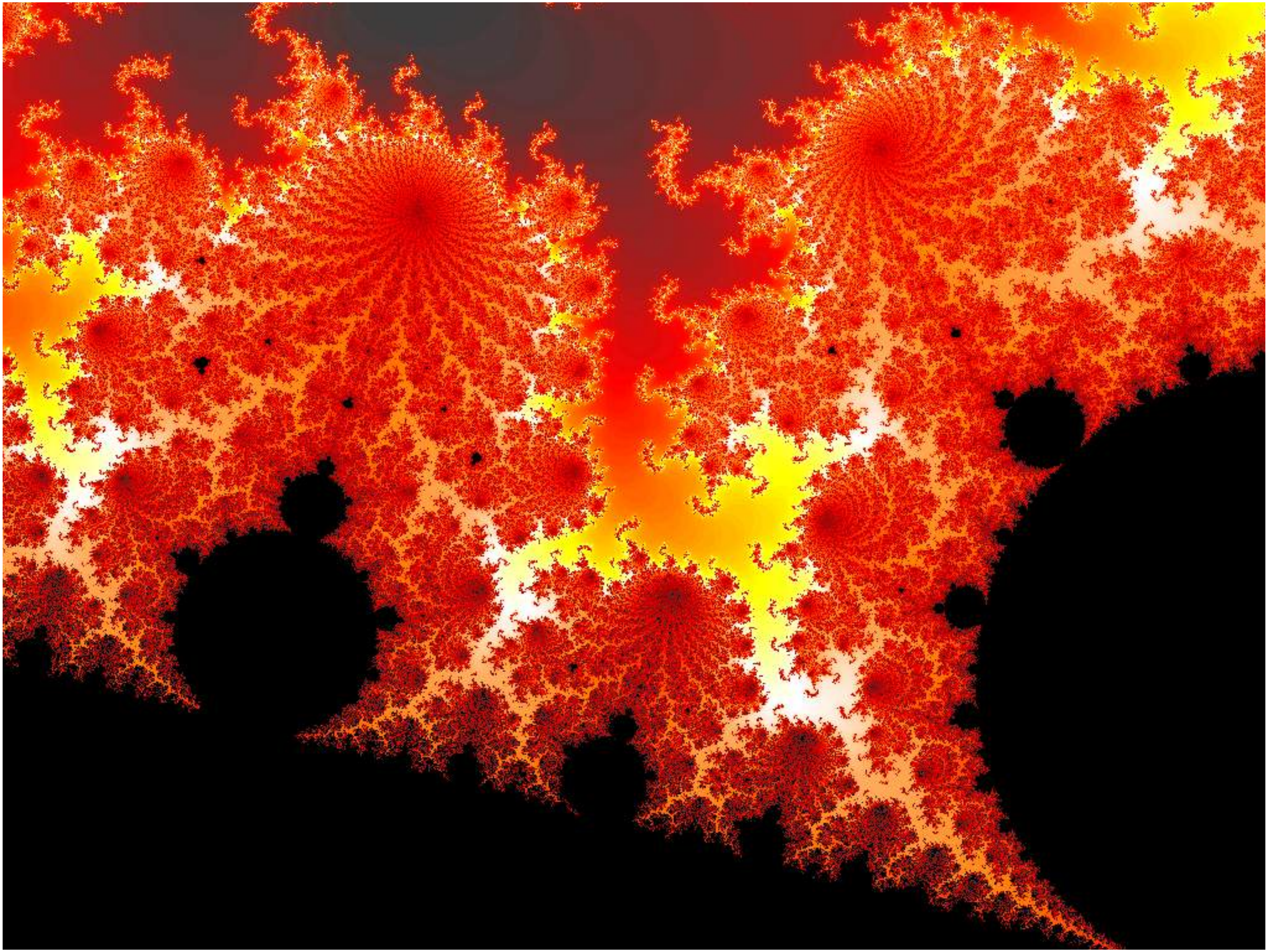
Quiz 0: Background (use index cards)

Front:

1. Full name and Email Address above the red line
2. Major/UG or G/Year (immediately below the red line)
3. Area (G: Research Area/UG: Interests)
4. Briefly describe any C/C++ experience.
5. In which programming languages are you fluent?
6. What is a bit?

Back:

1. What is an instruction cache?
2. What is the difference between a sequential and a combinational circuit?
3. What is a MUX?
4. Using AND, OR, and NOT gates design an XOR gate.



What is a Computer?

Computers haven't changed in over 50 years!

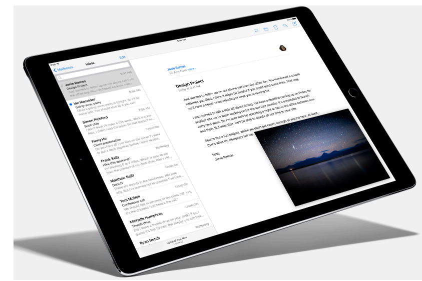
- Universal Turing Machine Equivalence
- Given enough time/memory, nothing new

Computers have undergone enormous changes!

- New applications enabled
- New form factors

Computers process information

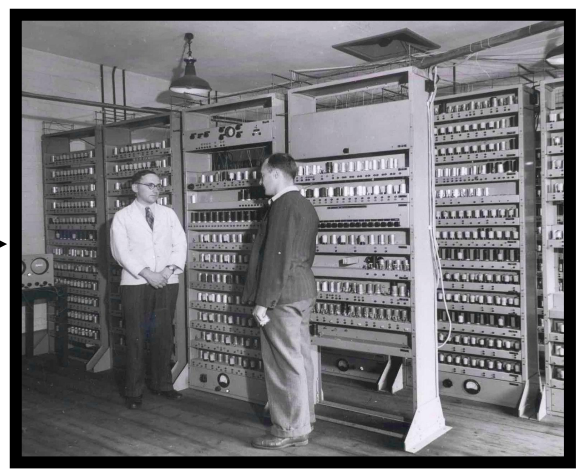
- Input/Output
- State
- Computation



"Thin. Light. Epic."

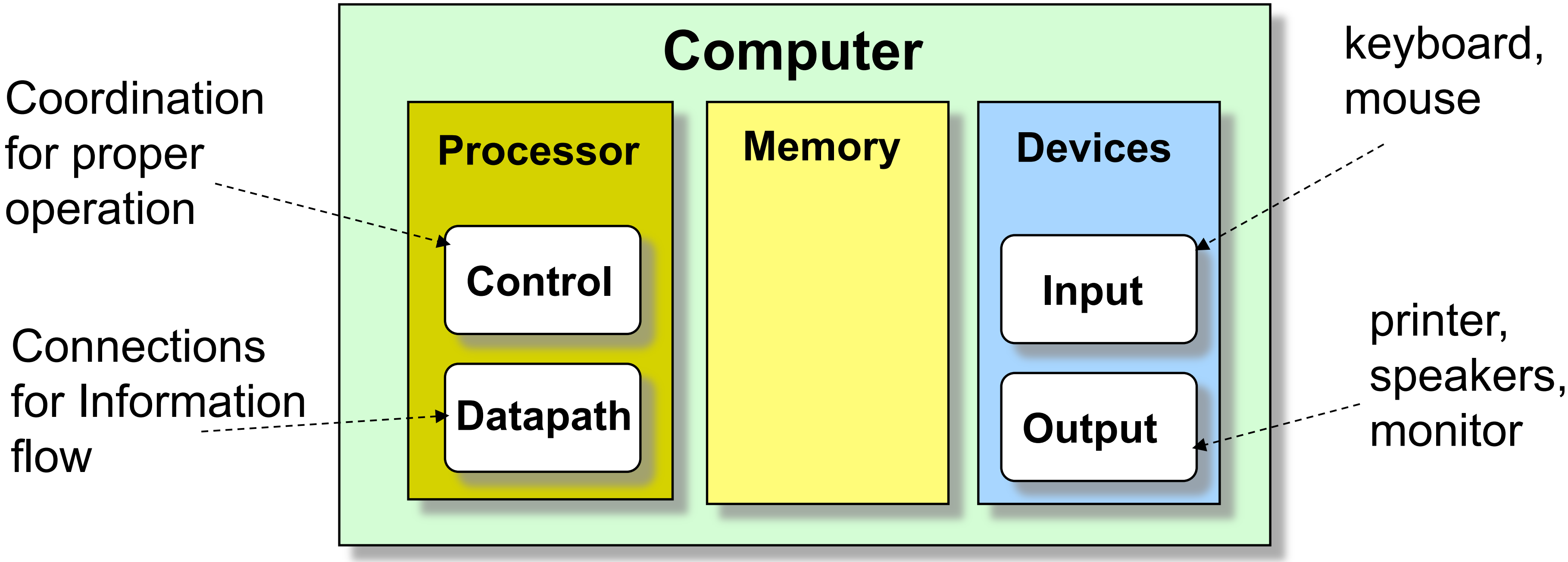
EDSAC

Universal Turing Machine



"Dude, your get'n a…"

Modern Computer Organization



- Where does the hard disk go?
- Computer system design
  - Enable applications (speed, reliability, efficiency)
  - Reduce cost (die size, technology, time-to-market)
- The Key: Manage Complexity!

Abstraction

- Separate implementation from specification
  - INTERFACE: specify the provided services.
  - IMPLEMENTATION: provide code or HW for operations.
  - CLIENT: code or HW that uses services.
- Examples: ADTs

- Principle of least privilege

The Living Daylights: Bond and Saunders are in a house waiting for General Koskov to defect. Bond is preparing to shoot a sniper.

Bond: What's your escape route?
Saunders: Sorry old man. Section 26 paragraph 5, that information is on a need-to-know basis only. I'm sure you'll understand.

Intuition

Client

Interface
- universal remote
- volume
- change channel
- adjust picture
- decode NTSC, PAL signals

Implementation
- cathode ray tube
- electron gun
- Sony Wega 36XBR250
- 241 pounds, $2,699

Intuition

Client

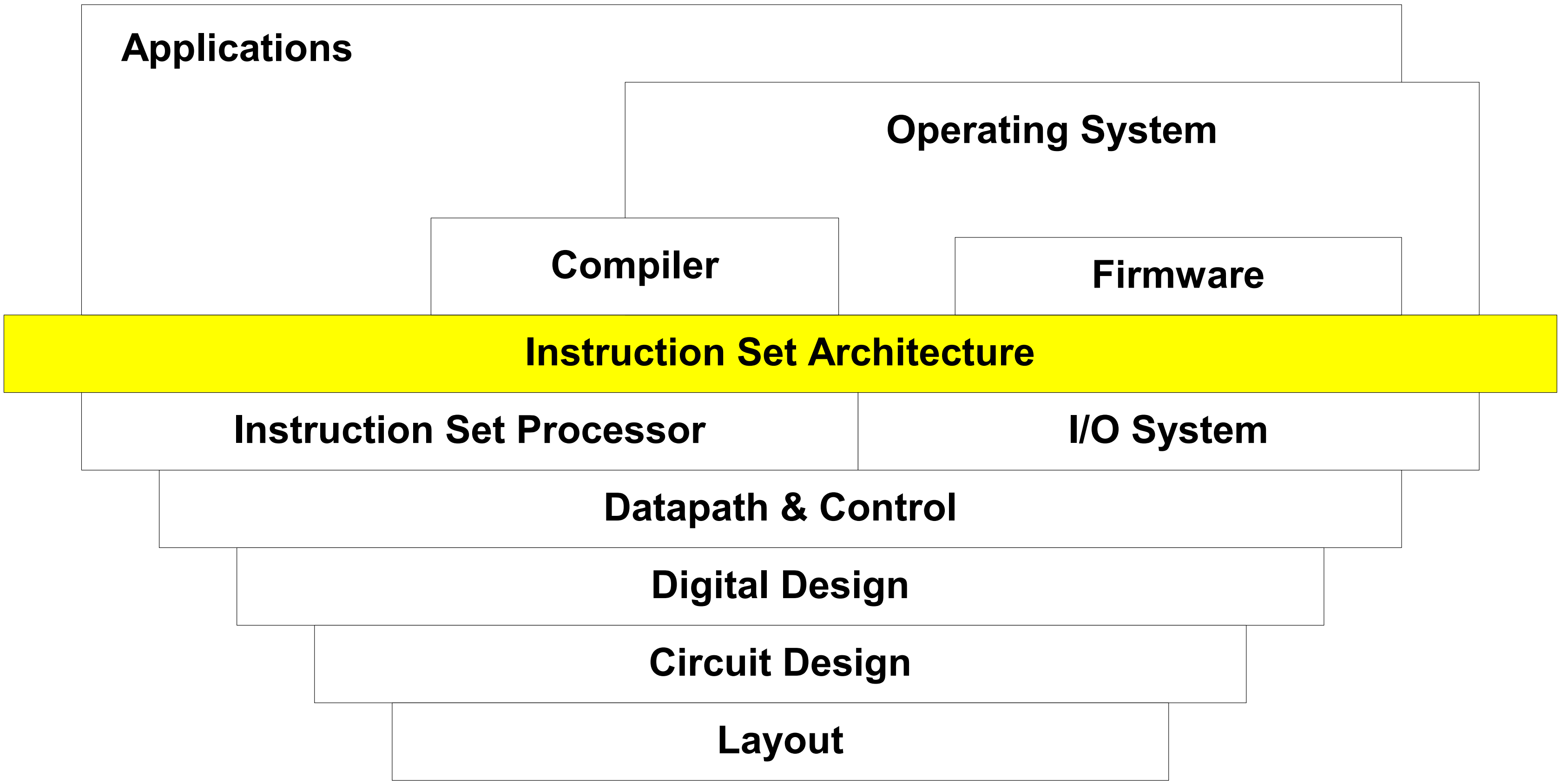
Interface
- universal remote
- volume
- change channel
- adjust picture
- decode NTSC, PAL signals

Implementation
- gas plasma monitor
- Pioneer PDP-502MX
- wall mountable
- 4 inches deep
- $19,995

Can substitute better implementation without changing client!

Interfaces in Computer Systems

Software



Hardware

Hello World

The Hello World Algorithm:

1. Emit "Hello World"
2. Terminate

Java Program

```
/* Hello World in Java */

public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Hello World

The Hello World Algorithm:

1. Emit “Hello World”
2. Terminate

C Program

```
/*
 * Good programs have meaningful comments
 */
#include <stdio.h>

int main()
{
  printf("Hello World!\n");
  return 0;
}
```

Hello World

```c
/*
 * Good programs have meaningful comments
 */
#include <stdio.h>

int main()
{
  printf("Hello World!\n");
  return 0;
}
```

C Program → GNU C Compiler → x86 Assembly Language

```
        .file   "hello.c"
        .section        .rodata
.LC0:
        .string "Hello World!\n"
        .text
.globl main
        .type   main,@function
main:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $8, %esp
        andl    $-16, %esp
        movl    $0, %eax
        subl    %eax, %esp
        subl    $12, %esp
        pushl   $.LC0
        call    printf
        addl    $16, %esp
        movl    $0, %eax
        leave
        ret
.Lfe1:
        .size   main,.Lfe1-main
        .ident  "GCC: (GNU) 3.2.2 20030222 (Red Hat Linux 3.2.2-5)"
```

A compiler is a program that takes a program written in a source language and translates into a functionally equivalent target language

Hello World

```c
/*
 * Good programs have meaningful comments
 */
#include <stdio.h>

int main()
{
  printf("Hello World!\n");
  return 0;
}
```

C Program → GNU C Compiler → IA-64 Assembly Language

GCC

```
        .file   "hello.c"
        .pred.safe_across_calls p1-p5,p16-p63
        .section        .rodata.str1.8,"ams",@progbits,1
        .align 8
.LC0:
        stringz "Hello World!\n"
.text
        .align 16
        .global main#
        .proc main#
main:
        .prologue 12, 33
        .save ar.pfs, r34
        alloc r34 = ar.pfs, 0, 3, 1, 0
        addl r35 = @ltoff(.LC0), gp
        .save rp, r33
        mov r33 = b0
        ;;
        .body
        ld8 r35 = [r35]
        br.call.sptk.many b0 = printf#
        ;;
        mov r8 = r0
        mov ar.pfs = r34
        mov b0 = r33
        br.ret.sptk.many b0
        .endp main#
        .ident  "GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.2 2.96-112.7.2)"
```

Hello World

IA-64 Assembly Language

```
        .file   "hello.c"
        .pred.safe_across_calls p1-p5,p16-p63
        .section        .rodata.str1.8,"ams",@progbits,1
        .align 8
.LC0:
        stringz "Hello World!\n"
.text
        .align 16
        .global main#
        .proc main#
main:
        .prologue 12, 33
        .save ar.pfs, r34
        alloc r34 = ar.pfs, 0, 3, 1, 0
        addl r35 = @ltoff(.LC0), gp
        .save rp, r33
        mov r33 = b0
        ;;
        .body
        ld8 r35 = [r35]
        br.call.sptk.many b0 = printf#
        ;;
        mov r8 = r0
        mov ar.pfs = r34
        mov b0 = r33
        br.ret.sptk.many b0
        .endp main#
        .ident  "GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.2 2.96-112.7.2)"
```

Hello World

```
        .file   "hello.c"
        .pred.safe_across_calls p1-p5,p16-p63
        .section        .rodata.str1.8,"ams",@progbits,1
        .align 8
.LC0:
        stringz "Hello World!\n"
.text
        .align 16
        .global main#
        .proc main#
main:
        .prologue 12, 33
        .save ar.pfs, r34
        alloc r34 = ar.pfs, 0, 3, 1, 0
        addl r35 = @ltoff(.LC0), gp
        .save rp, r33
        mov r33 = b0
        ;;
        .body
        ld8 r35 = [r35]
        br.call.sptk.many b0 = printf#
        ;;
        mov r8 = r0
        mov ar.pfs = r34
        mov b0 = r33
        br.ret.sptk.many b0
        .endp main#
        .ident  "GCC: (GNU) 2.96 20000731 (Red Hat Linux 7.2 2.96-112.7.2)"
```

IA-64 Assembly → GNU C Assembler → IA-64 Binary

0111111101000101...

$ objdump a.out

```
00000000: 7f45 4c46 0201 0100 0000 0000 0000
...
00000260: 5002 0000 0000 0000 006c 6962 632e 736f  P........libc.so
00000270: 2e36 2e31 0070 7269 6e74 6600 5f5f 6c69  .6.1.printf.__li
00000280: 6263 5f73 7461 7274 5f6d 6169 6e00 474c  bc_start_main.GL
00000290: 4942 435f 322e 3200 0000 0200 0200 0000  IBC_2.2.........
...
00000860: 4865 6c6c 6f20 576f 726c 6421 0d00 0000  Hello World!....
...
4000000000000690 <main>:
4000000000000690:       00 10 15 08 80 05       [MII]       alloc r34=ar.pfs,5,4,0
4000000000000696:       30 02 30 00 42 20                   mov r35=r12
400000000000069c:       04 00 c4 00                         mov r33=b0
40000000000006a0:       0a 20 81 03 00 24       [MMI]       addl r36=96,r1;;
40000000000006a6:       40 02 90 30 20 00                   ld8 r36=[r36]
40000000000006ac:       04 08 00 84                         mov r32=r1
40000000000006b0:       1d 00 00 00 01 00       [MFB]       nop.m 0x0
40000000000006b6:       00 00 00 02 00 00                   nop.f 0x0
40000000000006bc:       b8 fd ff 58                         br.call.sptk.many b0=4000000000000460;;
40000000000006c0:       00 08 00 40 00 21       [MII]       mov r1=r32
40000000000006c6:       80 00 00 00 42 00                   mov r8=r0
40000000000006cc:       20 02 aa 00                         mov.i ar.pfs=r34
40000000000006d0:       00 00 00 00 01 00       [MII]       nop.m 0x0
40000000000006d6:       00 08 05 80 03 80                   mov b0=r33
40000000000006dc:       01 18 01 84                         mov r12=r35
40000000000006e0:       1d 00 00 00 01 00       [MFB]       nop.m 0x0
40000000000006e6:       00 00 00 02 00 80                   nop.f 0x0
40000000000006ec:       08 00 84 00                         br.ret.sptk.many b0;;
```

The Instruction Set Architecture

"The vocabulary of commands"

- Defined by the Architecture (x86)
- Implemented by the Machine (Intel Core i7, 4 GHz)
- An Abstraction Layer: The Hardware/Software Interface
- Architecture has longevity over implementation
- Example:

add r1 = r2 + r3 (assembly)

001 001 010 011 (binary)

Opcode (verb) Operands (nouns)

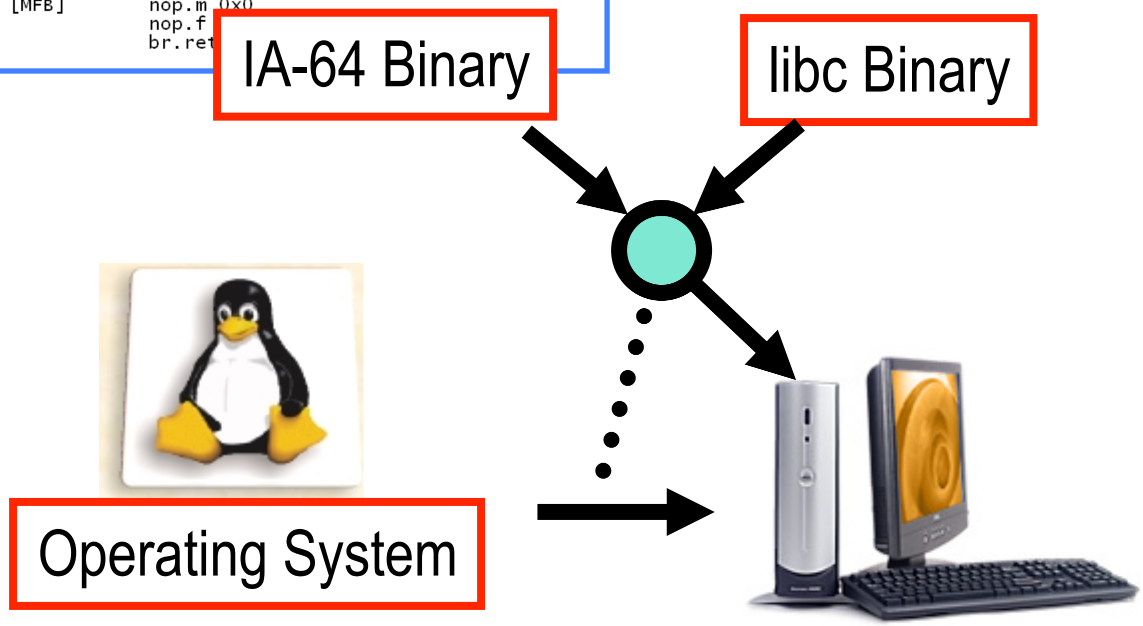
Hello World

IA-64 Binary (objdump)

```
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF.........

...

00000260: 5002 0000 0000 0000 006c 6962 632e 736f  P........libc.so
00000270: 2e36 2e31 0070 7269 6e74 6600 5f5f 6c69  .6.1.printf.__li
00000280: 6263 5f73 7461 7274 5f6d 6169 6e00 474c  bc_start_main.GL
00000290: 4942 435f 322e 3200 0000 0200 0200 0000  IBC_2.2.........

...

00000860: 4865 6c6c 6f20 576f 726c 6421 0d00 0000  Hello World!....

...

4000000000000690 <main>:
4000000000000690:       00 10 15 08 80 05       [MII]       alloc r34=ar.pfs,5,4,0
4000000000000696:       30 02 30 00 42 20                   mov r35=r12
400000000000069c:       04 00 c4 00                         mov r33=b0
40000000000006a0:       0a 20 81 03 00 24       [MMI]       addl r36=96,r1;;
40000000000006a6:       40 02 90 30 20 00                   ld8 r36=[r36]
40000000000006ac:       04 08 00 84                         mov r32=r1
40000000000006b0:       1d 00 00 00 01 00       [MFB]       nop.m 0x0
40000000000006b6:       00 00 00 02 00 00                   nop.f 0x0
40000000000006bc:       b8 fd ff 58                         br.call.sptk.many b0=4000000000000460;;
40000000000006c0:       00 08 00 40 00 21       [MII]       mov r1=r32
40000000000006c6:       80 00 00 00 42 00                   mov r8=r0
40000000000006cc:       20 02 aa 00                         mov.i ar.pfs=r34
40000000000006d0:       00 00 00 00 01 00       [MII]       nop.m 0x0
40000000000006d6:       00 08 05 80 03 80                   mov b0=r33
40000000000006dc:       01 18 01 84                         mov r12=r35
40000000000006e0:       1d 00 00 00 01 00       [MFB]       nop.m 0x0
40000000000006e6:       00 00 00 02 00 80                   nop.f 0x0
40000000000006ec:       08 00 84 00                         br.ret.sptk.many b0;;
```

Hello World in Action

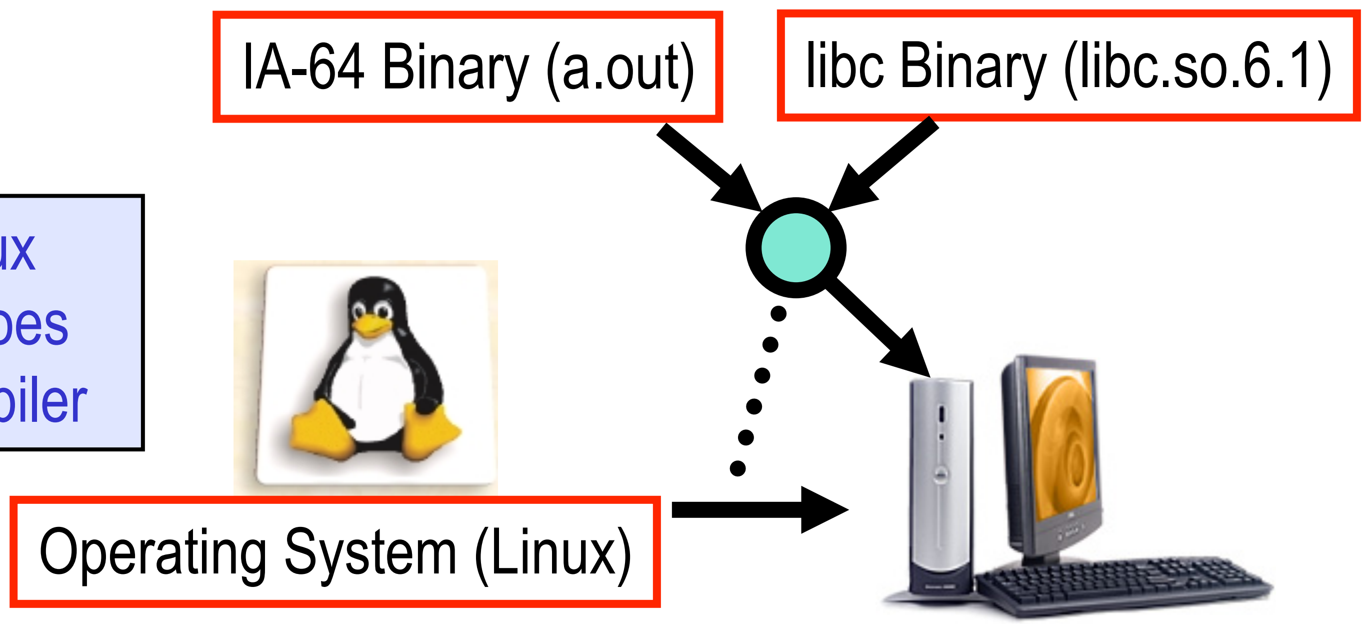
```
00000000: 7f45 4c46 0201 0100 0000 0000 0000 0000  .ELF............

...

00000260: 5002 0000 0000 0000 006c 6962 632e 736f  P........libc.so
00000270: 2e36 2e31 0070 7269 6e74 6600 5f5f 6c69  .6.1.printf.__li
00000280: 6263 5f73 7461 7274 5f6d 6169 6e00 474c  bc_start_main.GL
00000290: 4942 435f 322e 3200 0000 0200 0200 0000  IBC_2.2.........

...

00000860: 4865 6c6c 6f20 576f 726c 6421 0d00 0000  Hello World!....

...

4000000000000690 <main>:
4000000000000690:       00 10 15 08 80 05      [MII]       alloc r34=ar.pfs,5,4,0
4000000000000696:       30 02 30 00 42 20                  mov r35=r12
400000000000069c:       04 00 c4 00                        mov r33=b0
40000000000006a0:       0a 20 81 03 00 24      [MMI]       addl r36=96,r1;;
40000000000006a6:       40 02 90 30 20 00                  ld8 r36=[r36]
40000000000006ac:       04 08 00 84                        mov r32=r1
40000000000006b0:       1d 00 00 00 01 00      [MFB]       nop.m 0x0
40000000000006b6:       00 00 00 02 00 00                  nop.f 0x0
40000000000006bc:       b8 fd ff 58                        br.call.sptk.many b0=4000000000000460;;
40000000000006c0:       00 08 00 40 00 21      [MII]       mov r1=r32
40000000000006c6:       80 00 00 00 42 00                  mov r8=r0
40000000000006cc:       20 02 aa 00                        mov.i ar.pfs=r34
40000000000006d0:       00 00 00 00 01 00      [MII]       nop.m 0x0
40000000000006d6:       00 08 05 80 03 80                  mov b0=r33
40000000000006dc:       01 18 01 84                        mov r12=r35
40000000000006e0:       1d 00 00 00 01 00      [MFB]       nop.m 0x0
40000000000006e6:       00 00 00 02 00 80                  nop.f
40000000000006ec:       08 00 84 00                        br.ret
```



Hello World in Action

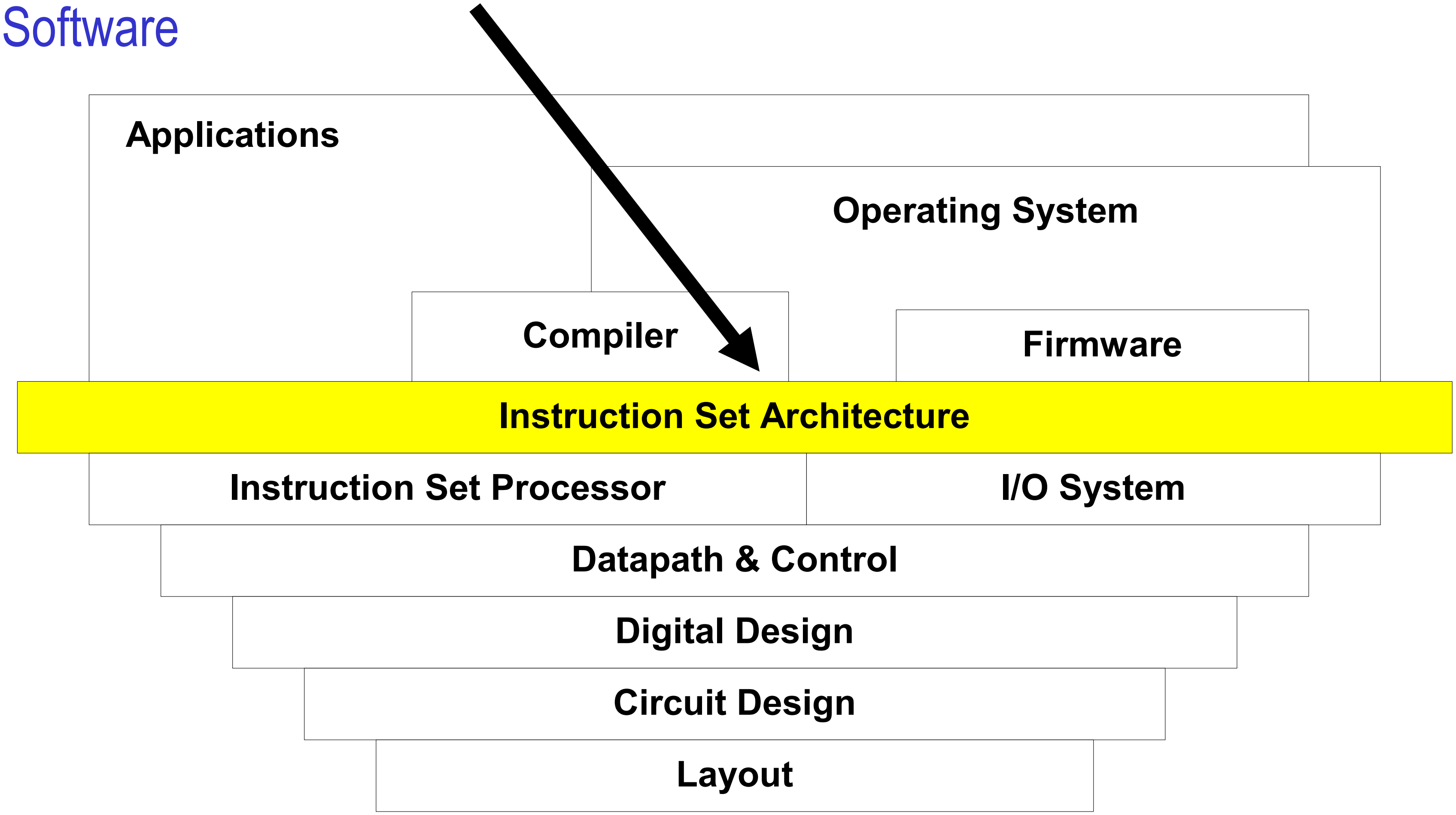
```
$ strace a.out
execve("/u/loc/august/hello/a.out", ["a.out"], [/* 112 vars */]) = 0
...
open("/lib/libc.so.6.1", O_RDONLY)       = 3
...
write(1, "Hello World!\n", 13)           = 13
...
exit(0)                                  = ?
```

99.5% of Linux source code goes through a compiler



Interfaces in Computer Systems

Software



Hardware

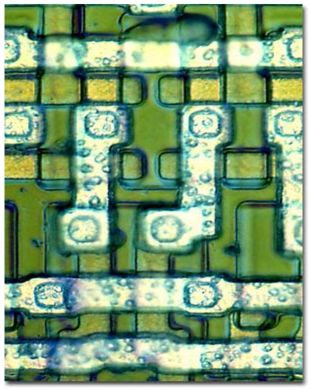
Physical Principles of Information

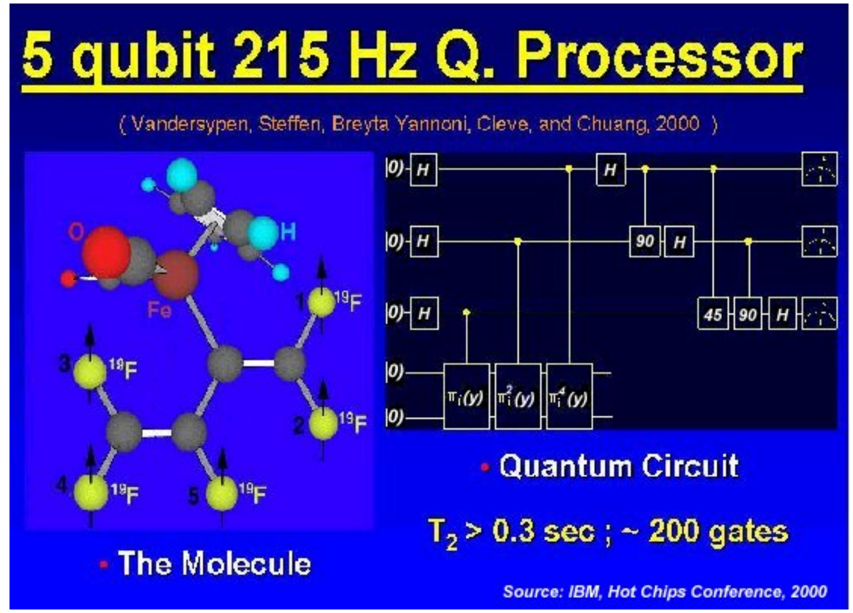
- Fredkin-Toffoli axioms.

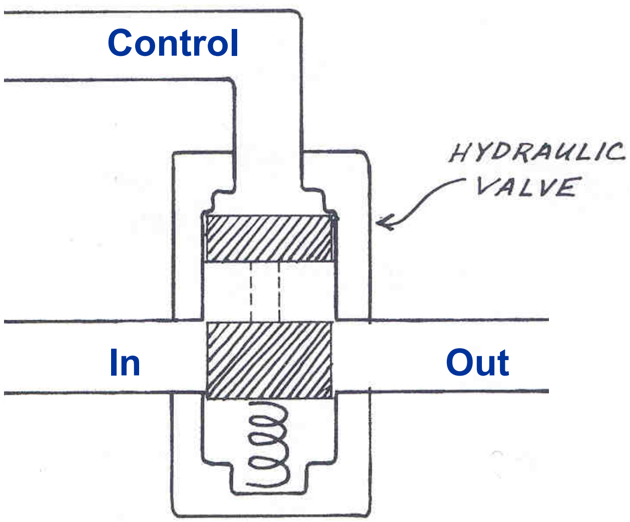
E. F. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219--253, 1982.

- *The speed of propagation of information is bounded.*
  - Speed of light
  - No action at a distance – causal effects propagate thru local interactions
- *The amount of information which can be encoded in the state of a finite system is bounded.*
  - Bounded by thermodynamical/quantum-mechanical considerations
- *It is possible to construct macroscopic, dissipative physical devices which perform in a recognizable and reliable way the logical functions AND, OR, and FAN-OUT.*

Physical Devices

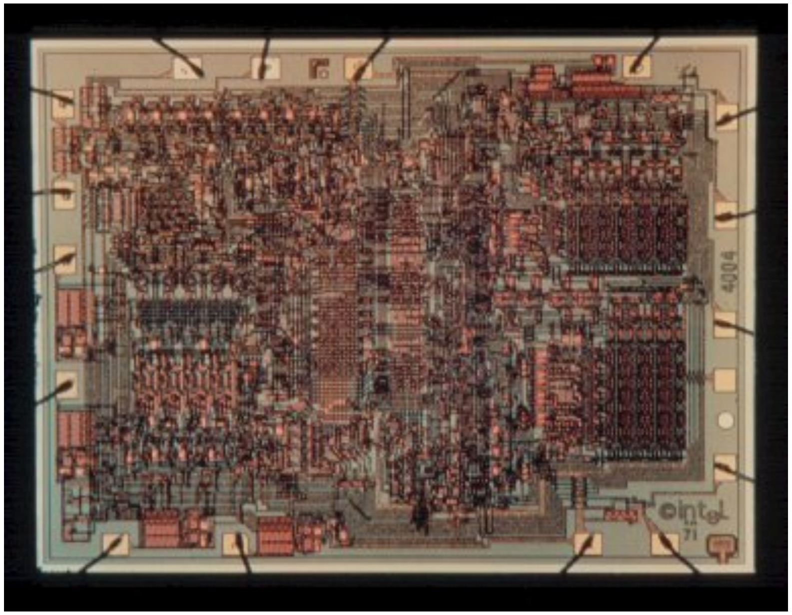


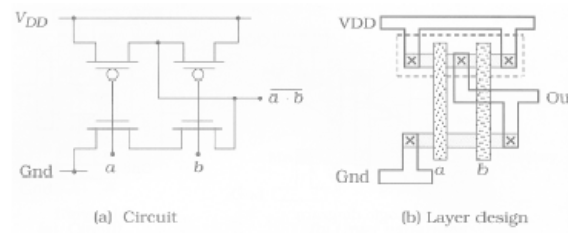


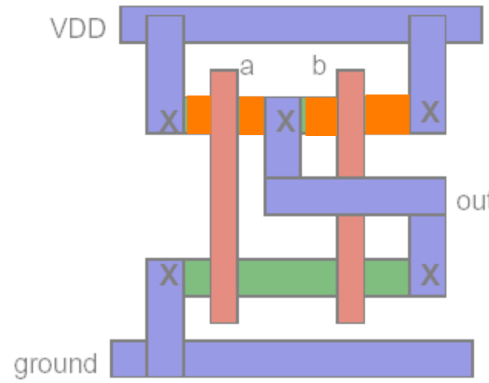


As we will see later, choice of device is critical!!

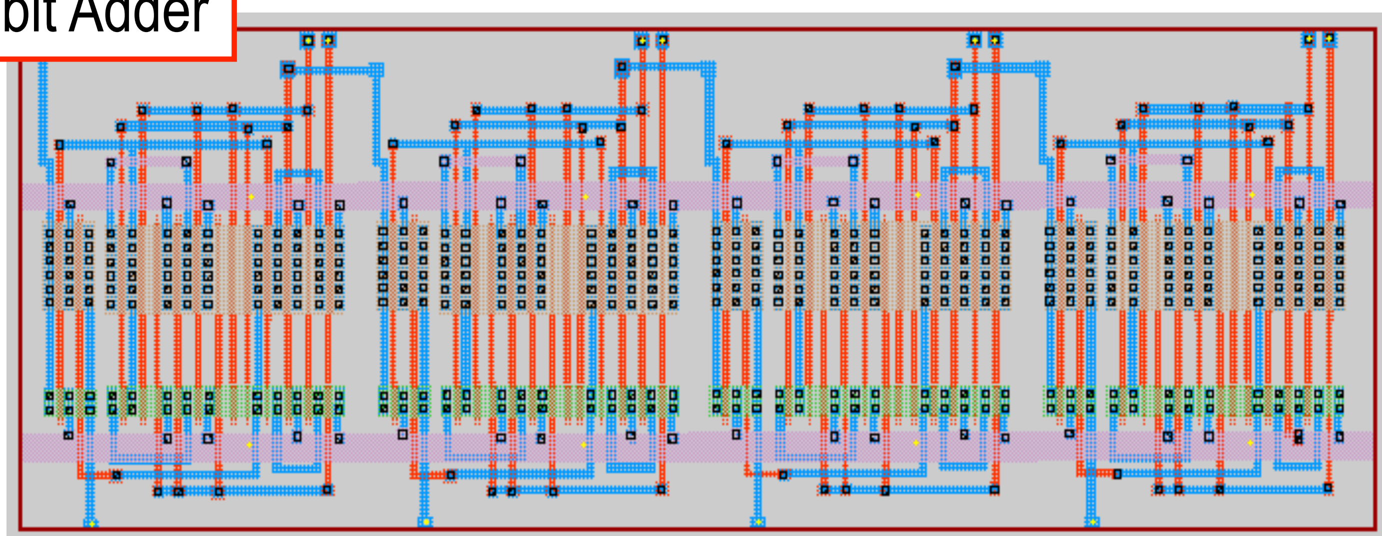
Layout/Circuit Design



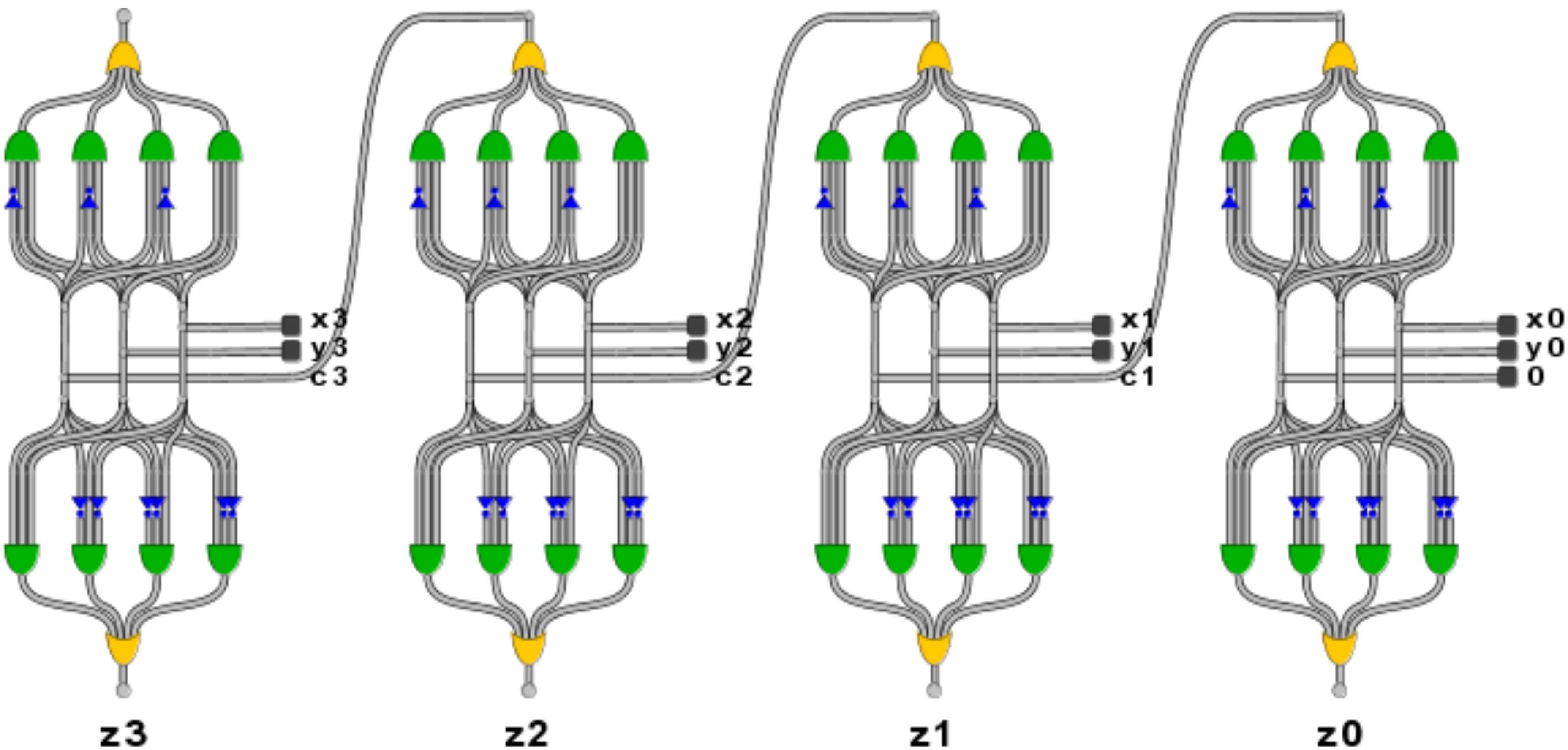




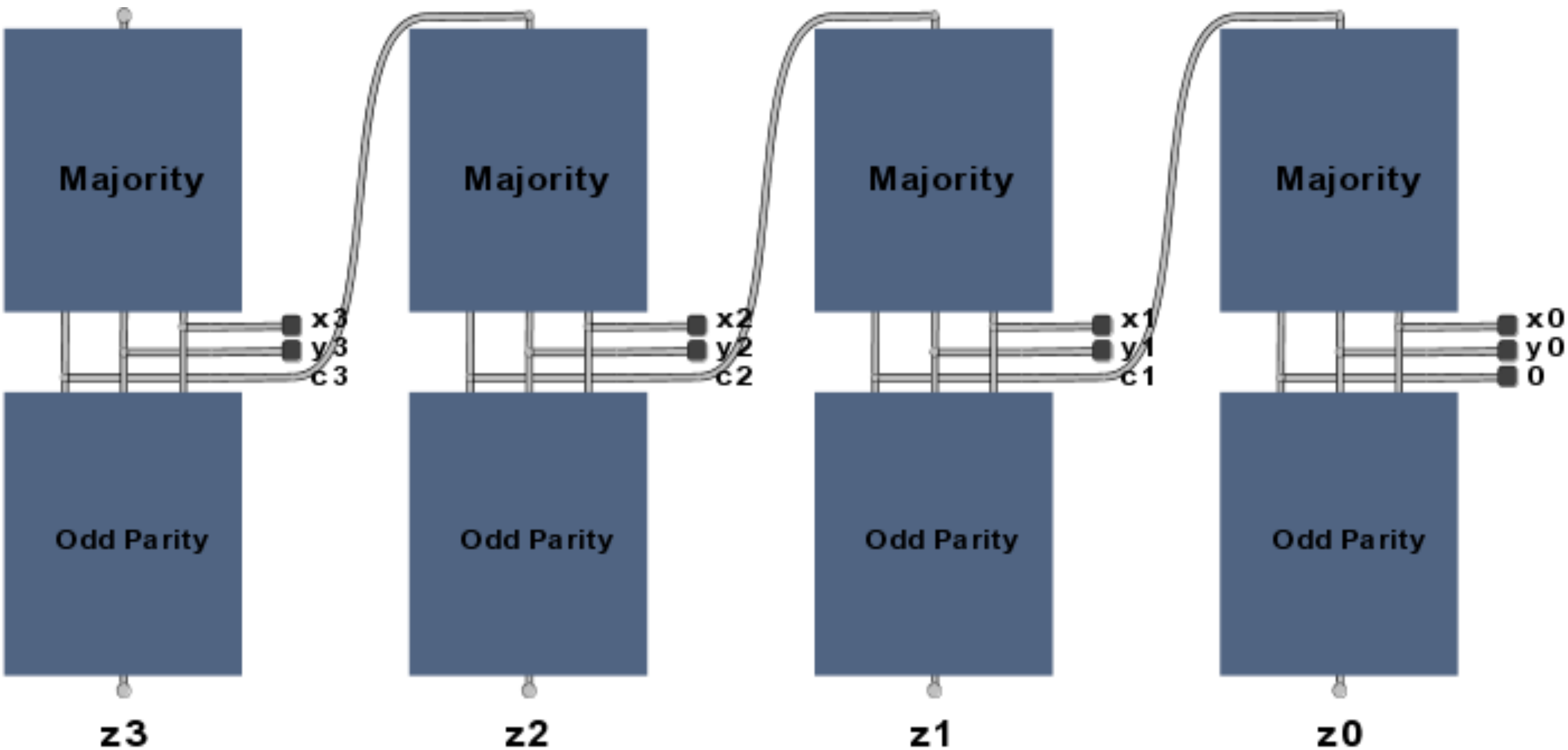
4-bit Adder



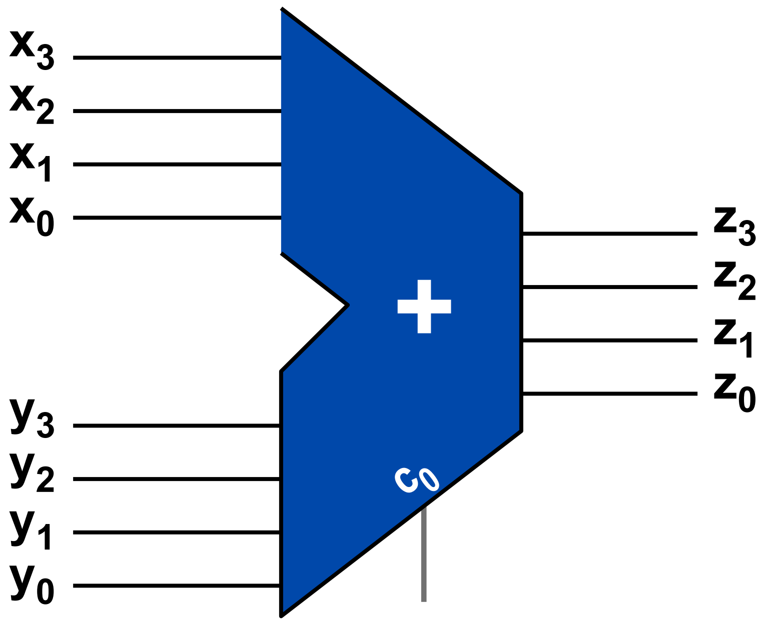
Digital Design
4-bit Adder



Digital Design
4-bit Adder

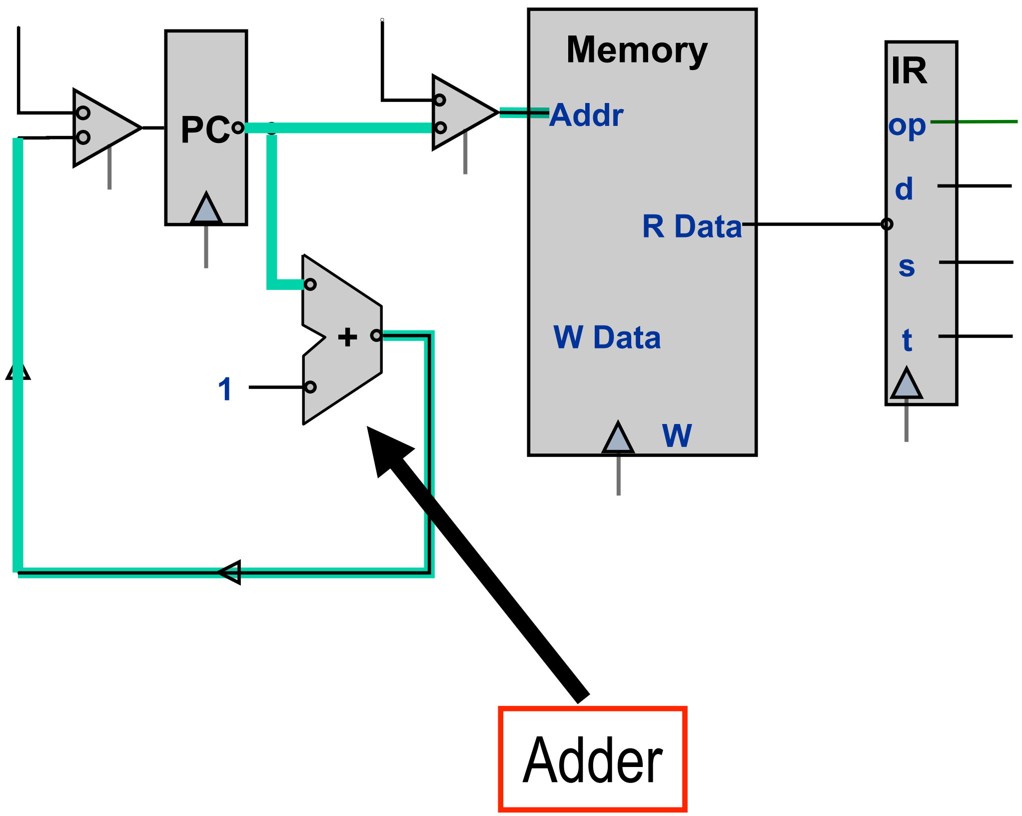


4-bit Adder

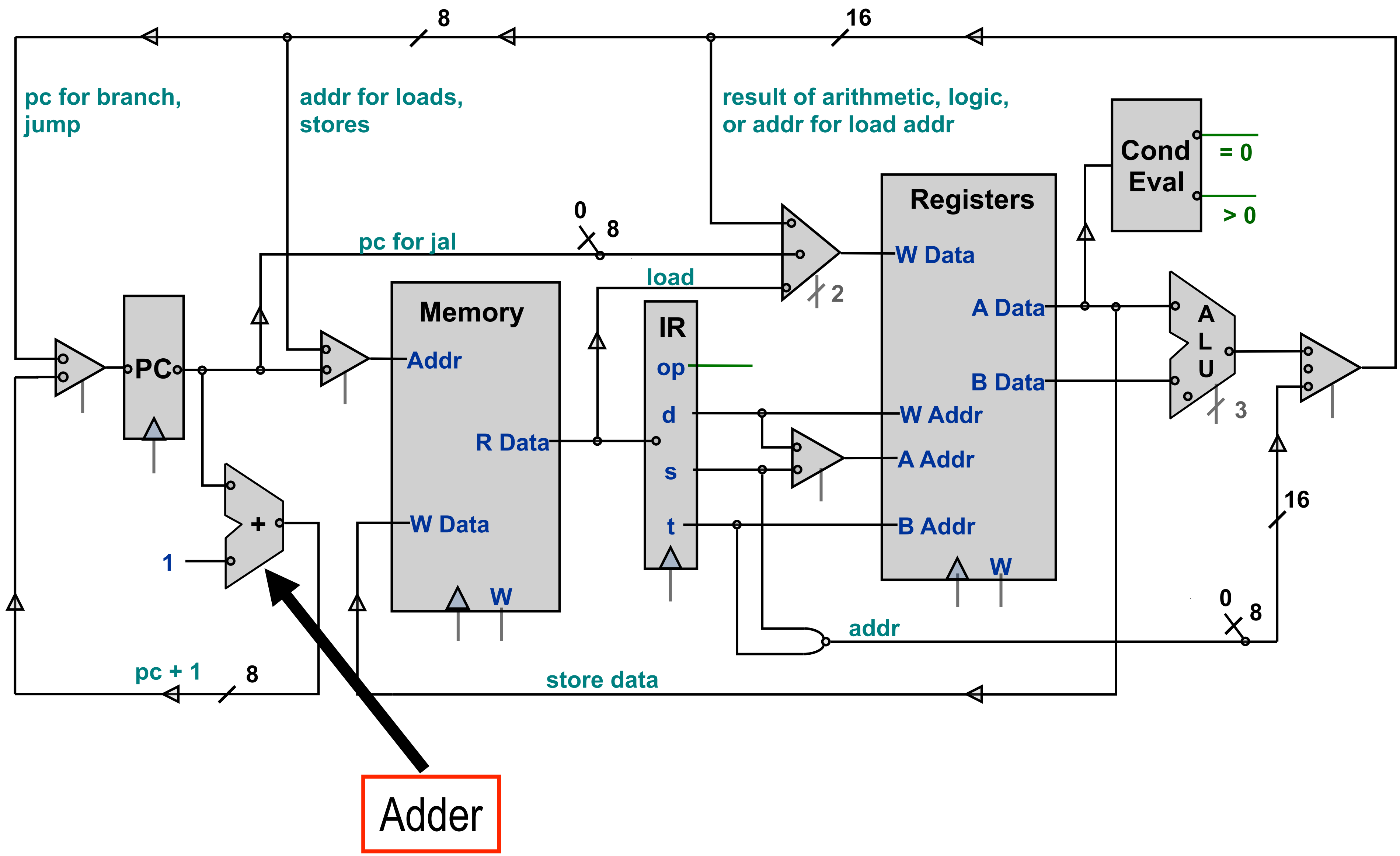


Datapath

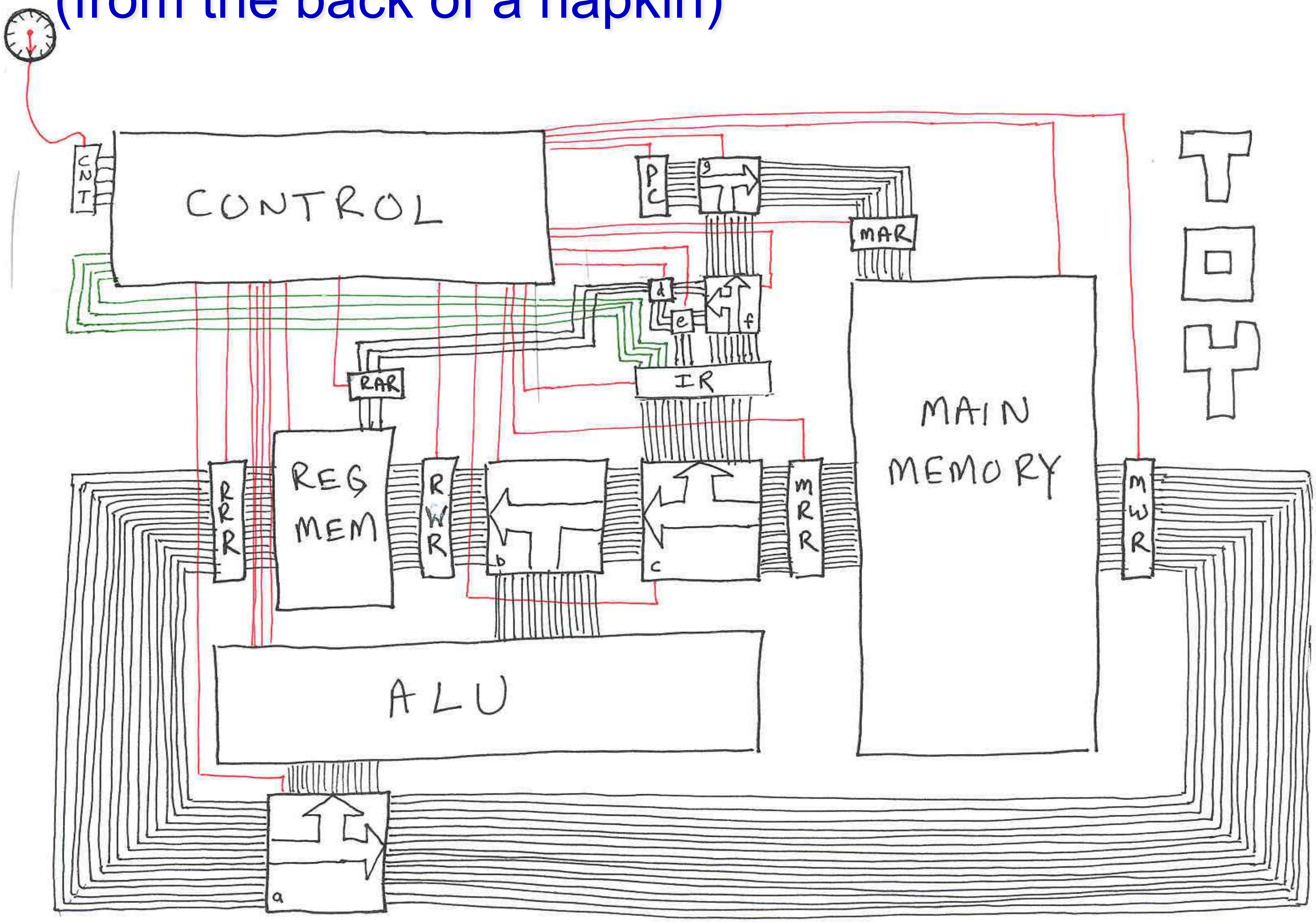
Datapath is a conduit for information flow through the processor



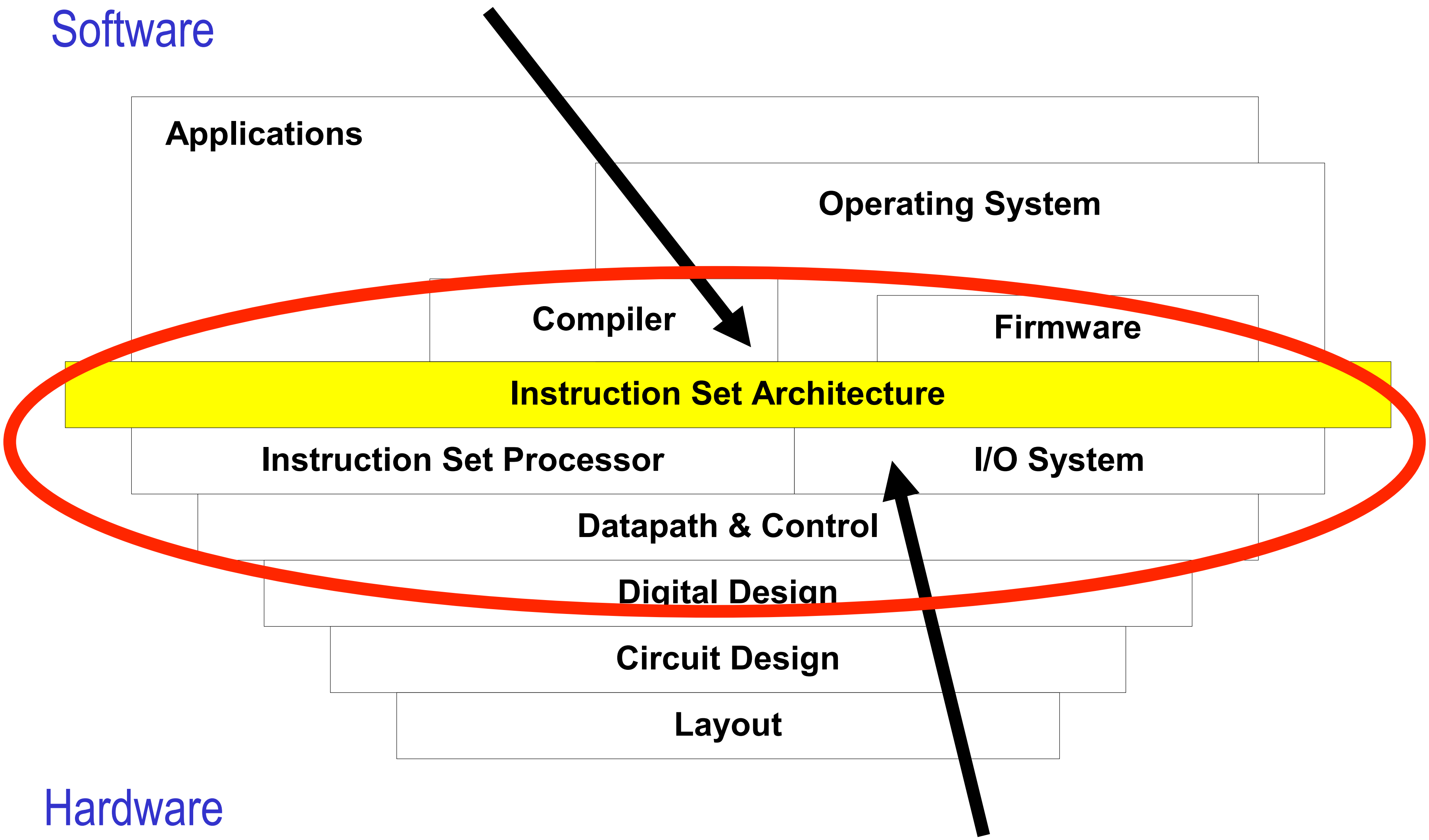
Complete Datapath



Control
(from the back of a napkin)



The Hardware/Software Interface



The Instruction Set Architecture

"The vocabulary of commands"

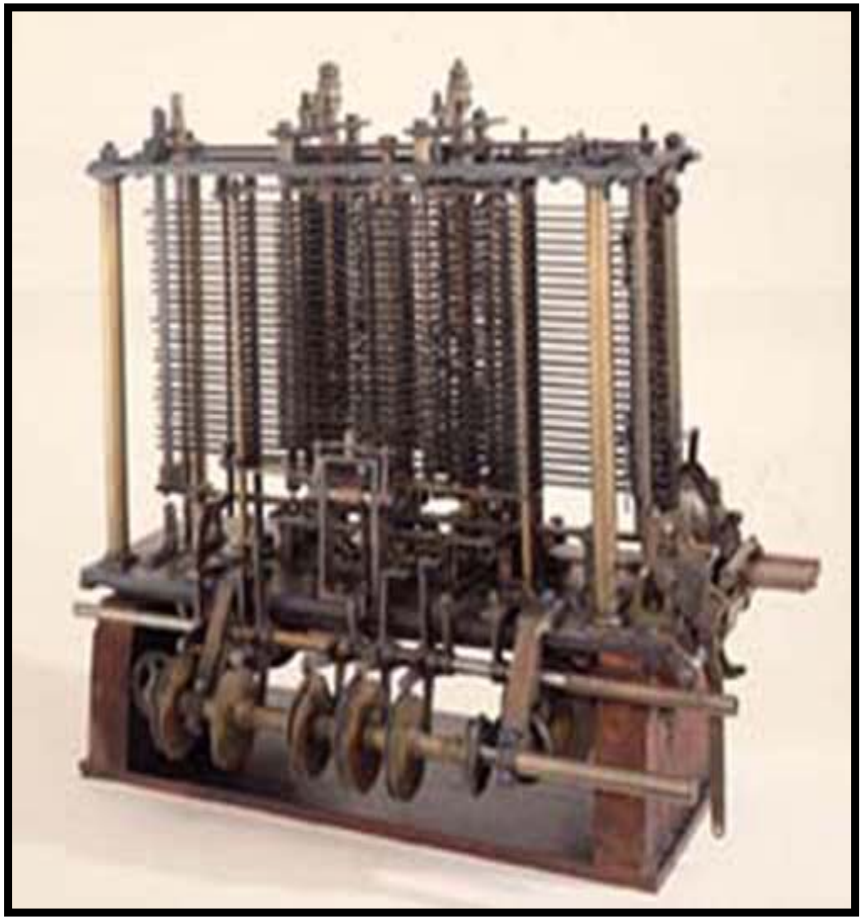
- Defined by the Architecture (x86)
- Implemented by the Machine (Pentium 4, 3.06 GHz)
- An Abstraction Layer: The Hardware/Software Interface
- Architecture has longevity over implementation
- Example:

add r1 = r2 + r3 (assembly)

001 001 010 011 (binary)

Opcode (verb) Operands (nouns)

Computer Pre-history



Charles Babbage

- Analytical Engine – “programmable”
- Started in 1834
- Babbage Never Finished

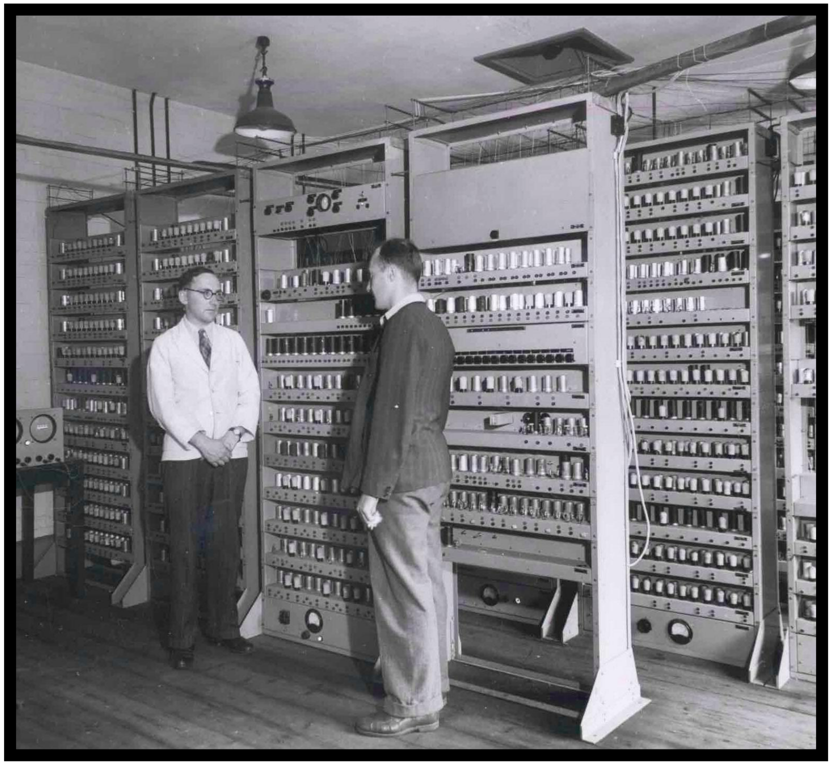
Computer History

ENIAC

Eckert and Mauchly

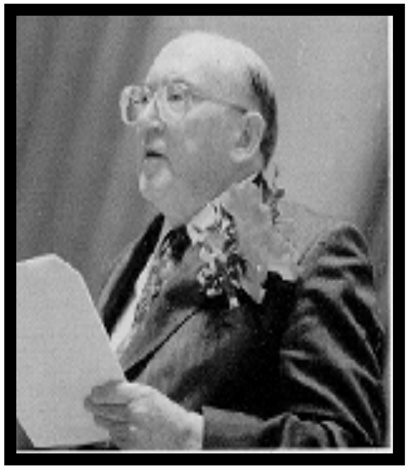
- 1st working programmable electronic computer (1946)
- 18,000 Vacuum tubes
- 1,800 instructions/sec
- 3,000 ft^3

Computer History



EDSAC 1 (1949)

Maurice Wilkes



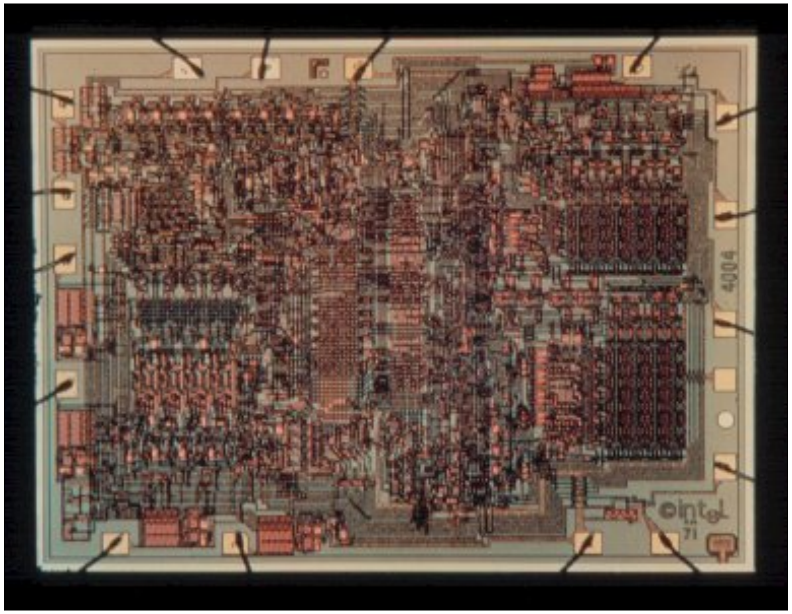
1st stored program computer

650 instructions/sec

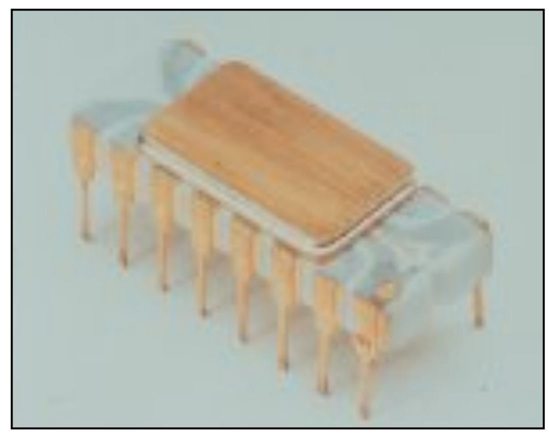
1,400 ft^3

Slice of History – Intel Processors

Intel 4004 Die Photo

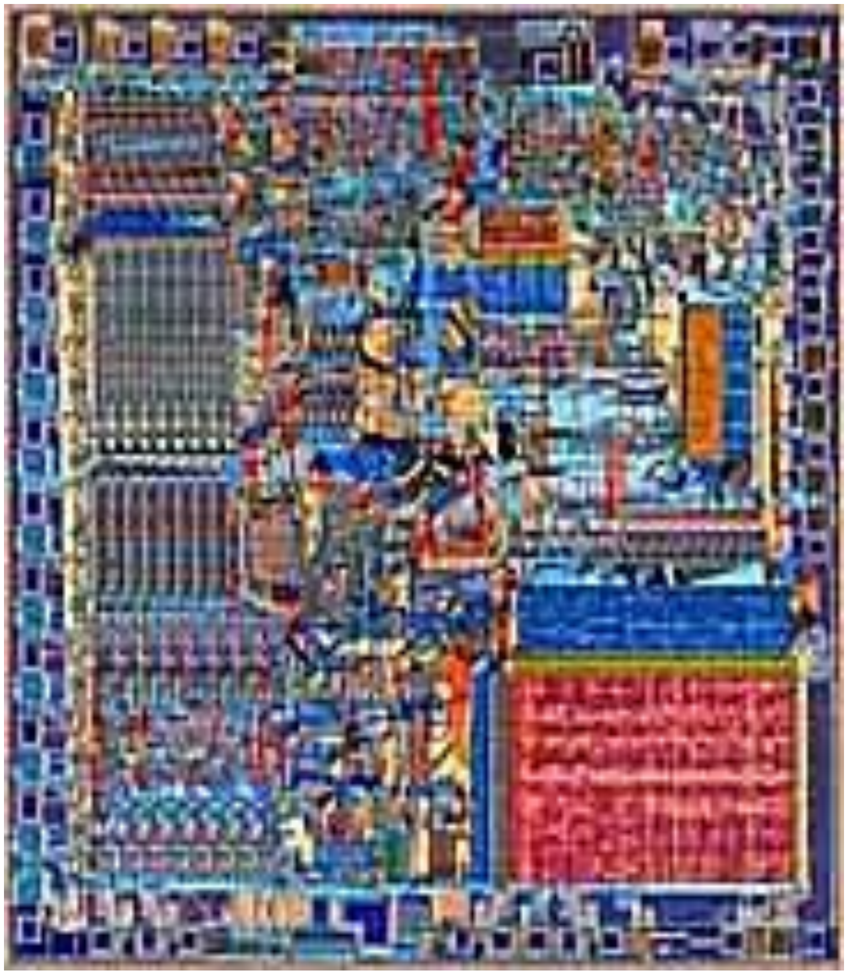


- First microprocessor
- Introduced in 1970
- 2,250 transistors
- 12 mm^2
- 108 KHz
- 2 Designers



Slice of History – Intel Processors

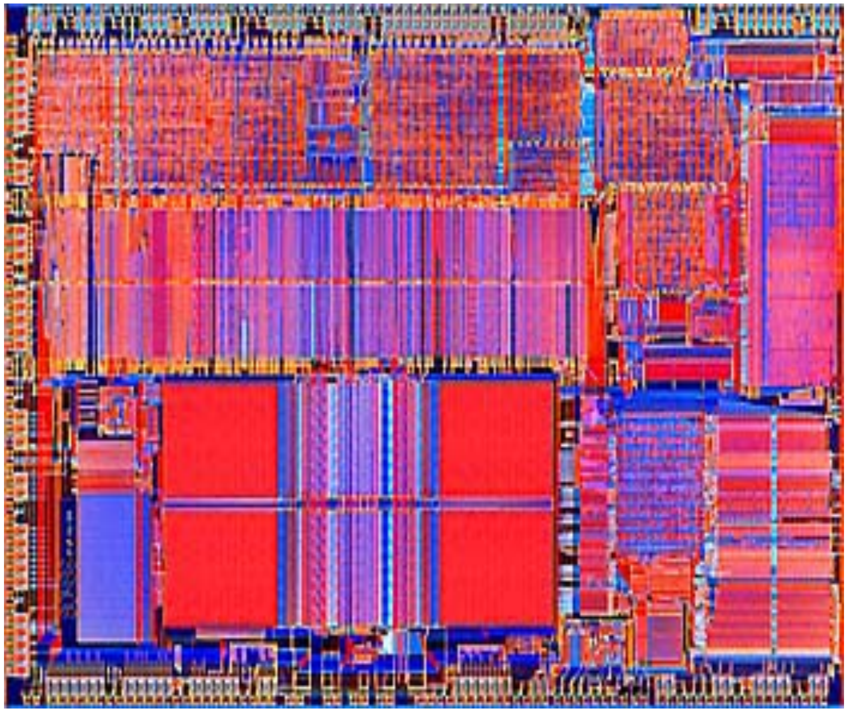
Intel 8086 Die Scan



- Basic x86 architecture
- Introduced in 1979
- 29,000 transistors
- 33 mm^2
- 5 MHz

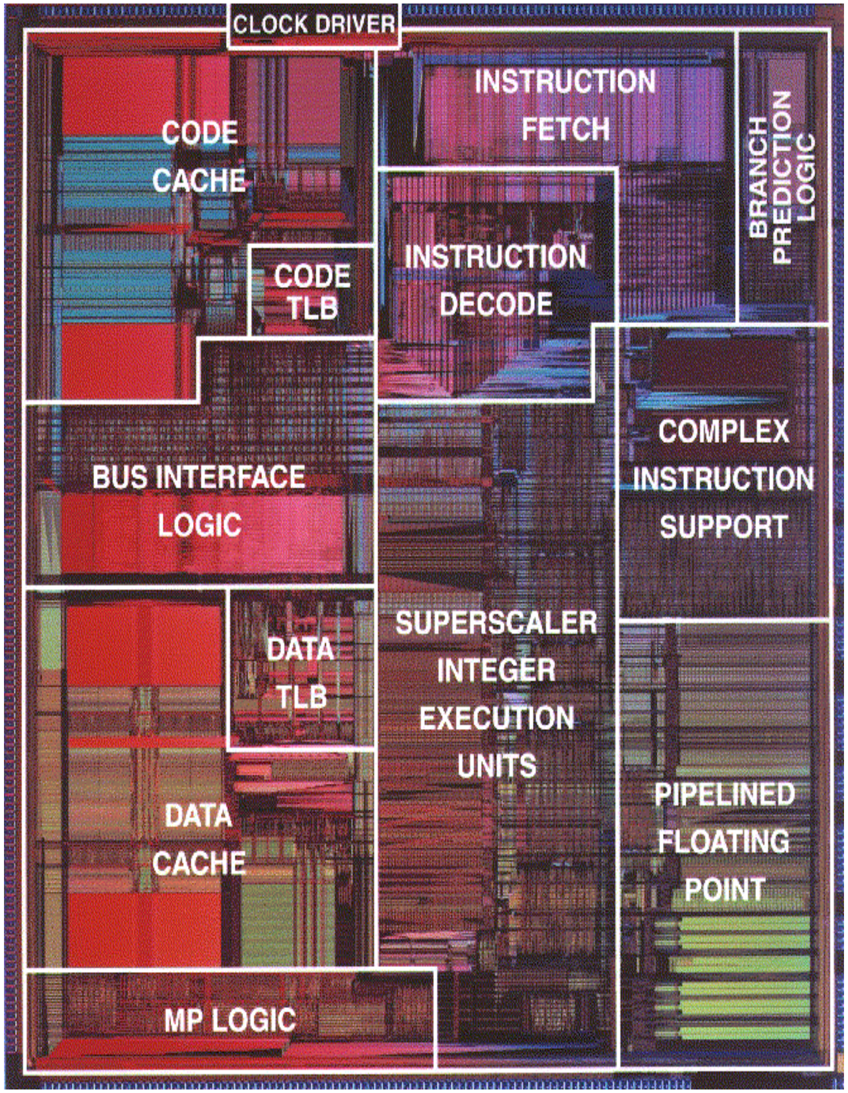
Slice of History – Intel Processors

Intel 80486 Die Scan



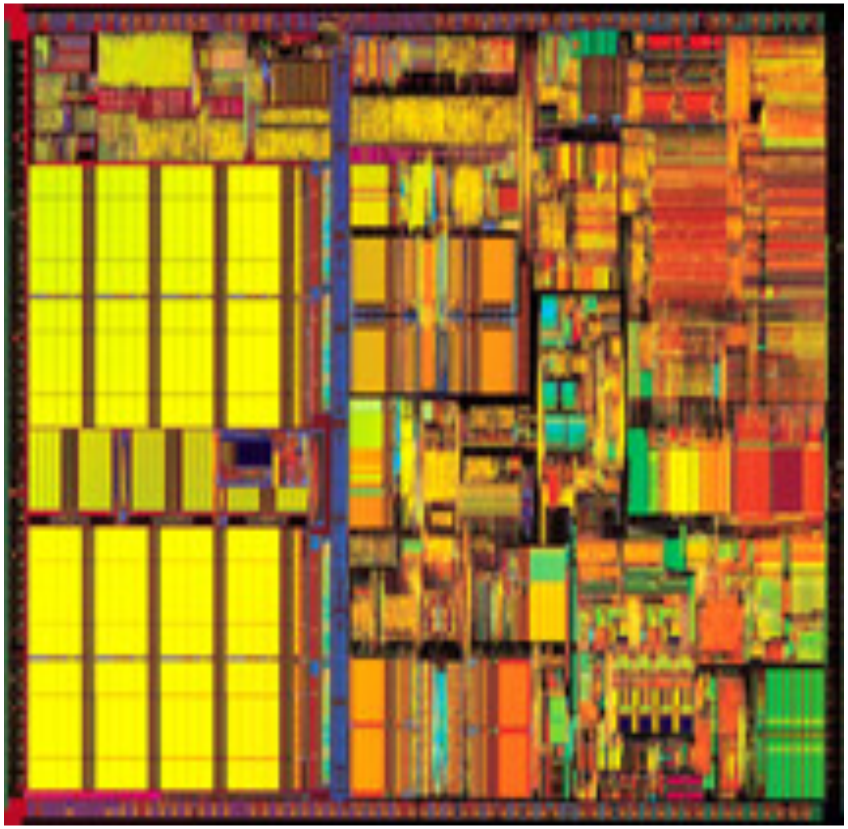
- 1st pipelined x86 implementation
- Introduced in 1989
- 1,200,000 transistors
- 81 mm^2
- 25 MHz

Slice of History – Intel Processors
Pentium Die Photo



- 1st superscalar x86
- Introduced in 1993
- 3,100,000 transistors
- 296 mm^2
- 60 MHz

Slice of History – Intel Processors
Pentium III

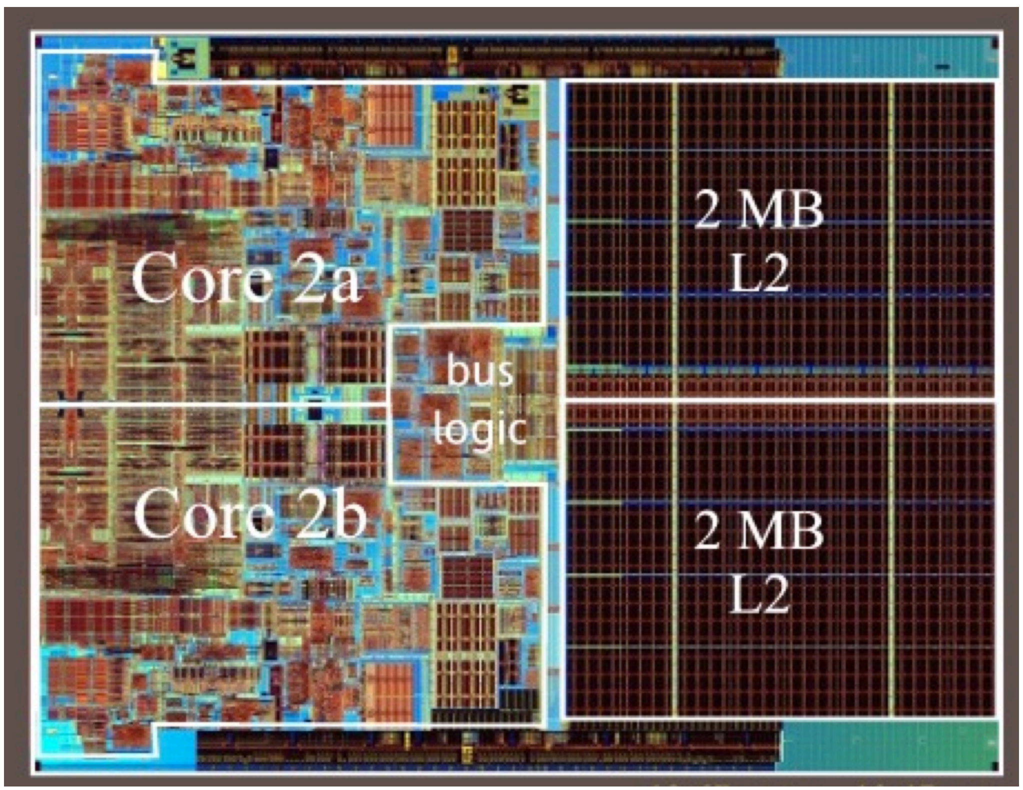


- 9,5000,000 transistors
- 125 mm^2
- 450 MHz
- Introduced in 1999

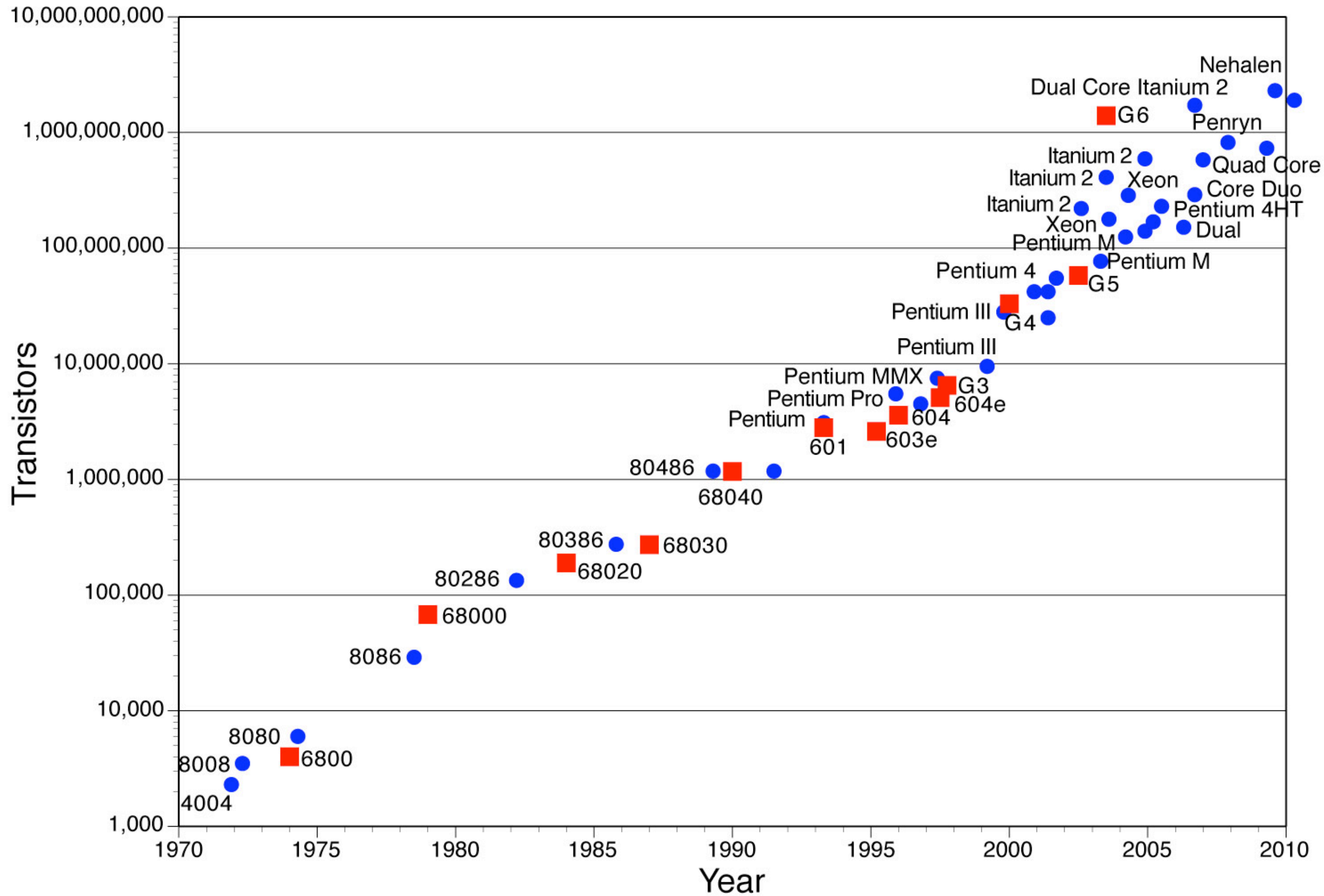
Slice of History – Intel Processors

Core 2 Duo (Merom)

- 293,000,000 transistors
- 143 mm^2
- 1.6 GHz - 3.16 GHz
- Introduced in 2006



Slice of History – Intel Processors

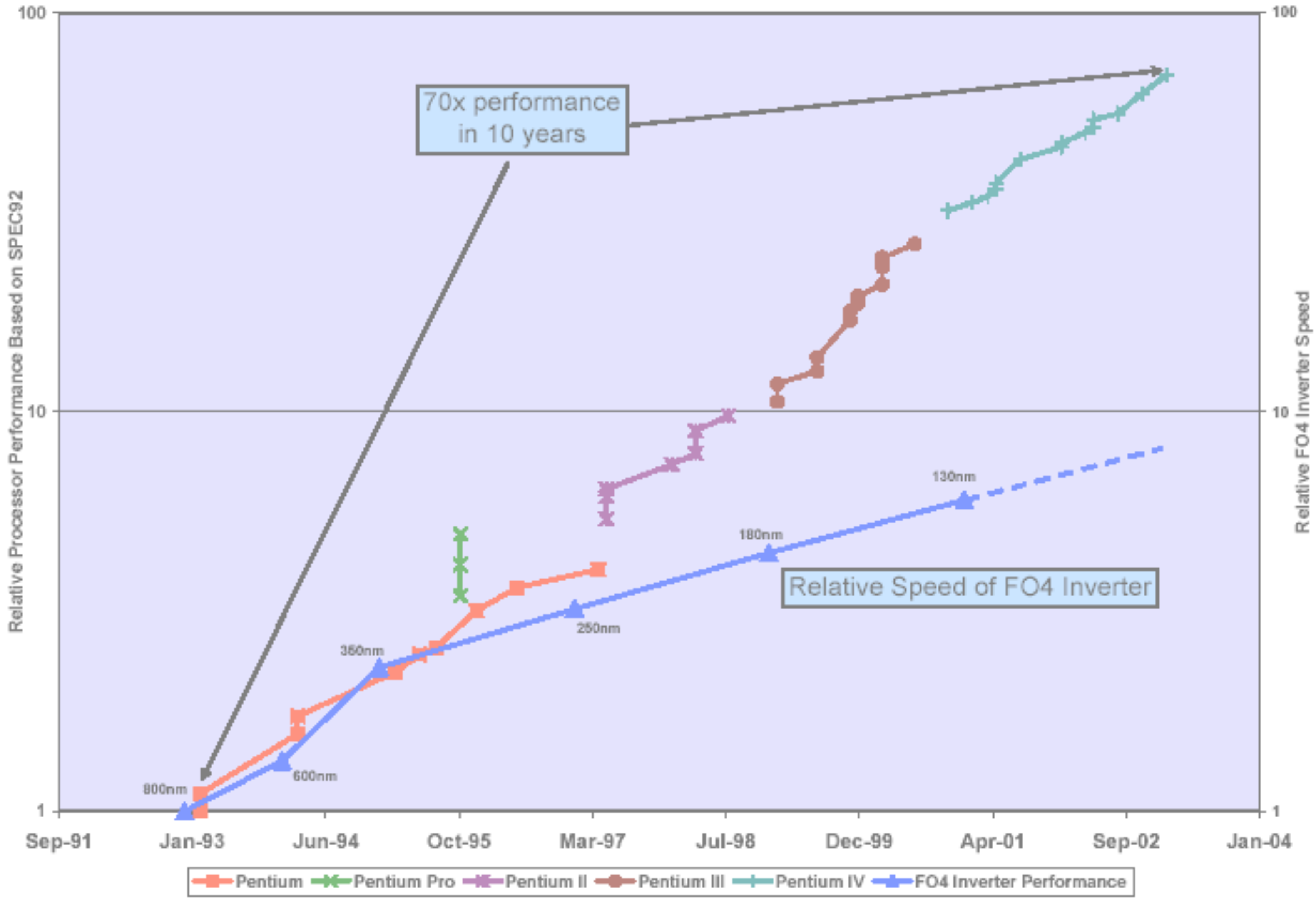


"Grove giveth and Gates taketh away." - Bob Metcalfe (inventor of Ethernet)

Other Technology Trends

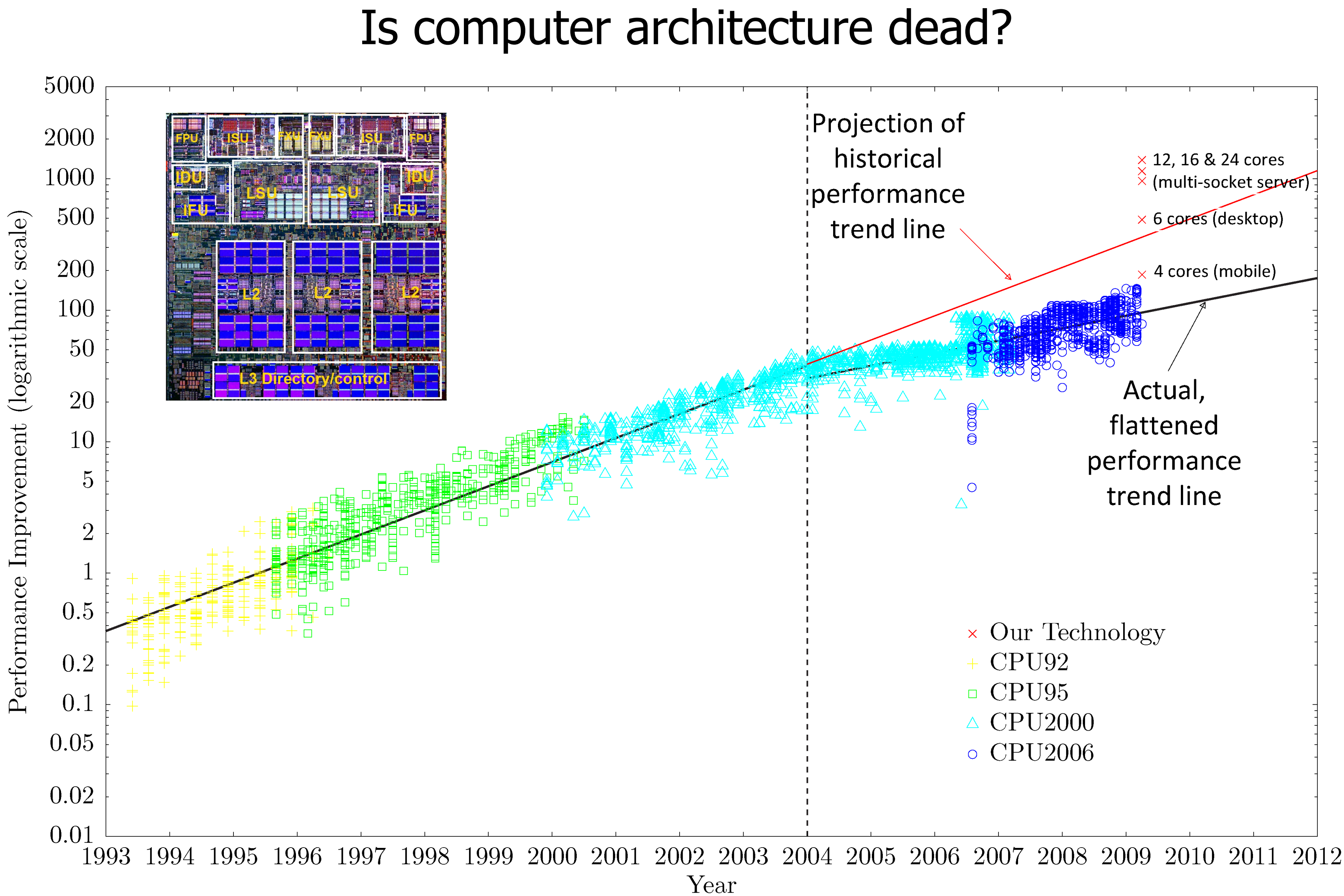
- Processor
  - Logic Capacity: ~30% increase per year
  - Clock Rate: ~20% increase per year
- Memory
  - DRAM Capacity: ~60% increase per year
  - Memory Speed: ~10% increase per year
  - Cost per Bit: ~25% decrease per year
- Disk
  - Capacity: ~60% increase per year

Trends…



70x performance in 10 years
Relative Speed of FO4 Inverter
Relative Processor Performance Based on SPEC92
Relative FO4 Inverter Speed
100
10
1
800nm
600nm
350nm
250nm
180nm
130nm
Sep-91
Jan-93
Jun-94
Oct-95
Mar-97
Jul-98
Dec-99
Apr-01
Sep-02
Jan-04
Pentium
Pentium Pro
Pentium II
Pentium III
Pentium IV
FO4 Inverter Performance

Trends?


Is computer architecture dead?
5000
2000
1000
500
200
100
50
20
10
5
2
1
0.5
0.2
0.1
0.05
0.02
0.01
Performance Improvement (logarithmic scale)
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
Year
FPU
ISU
FXU
IDU
LSU
IFU
L2
L3 Directory/control
Projection of historical performance trend line
12, 16 & 24 cores (multi-socket server)
6 cores (desktop)
4 cores (mobile)
Actual, flattened performance trend line
Our Technology
CPU92
CPU95
CPU2000
CPU2006

One More Trend…

- Intel 4004 (Thousands of transistors)
  - Number of Designers: 2

- Intel Core i7 (Billions of transistors)
  - Number of Designers: ~1500

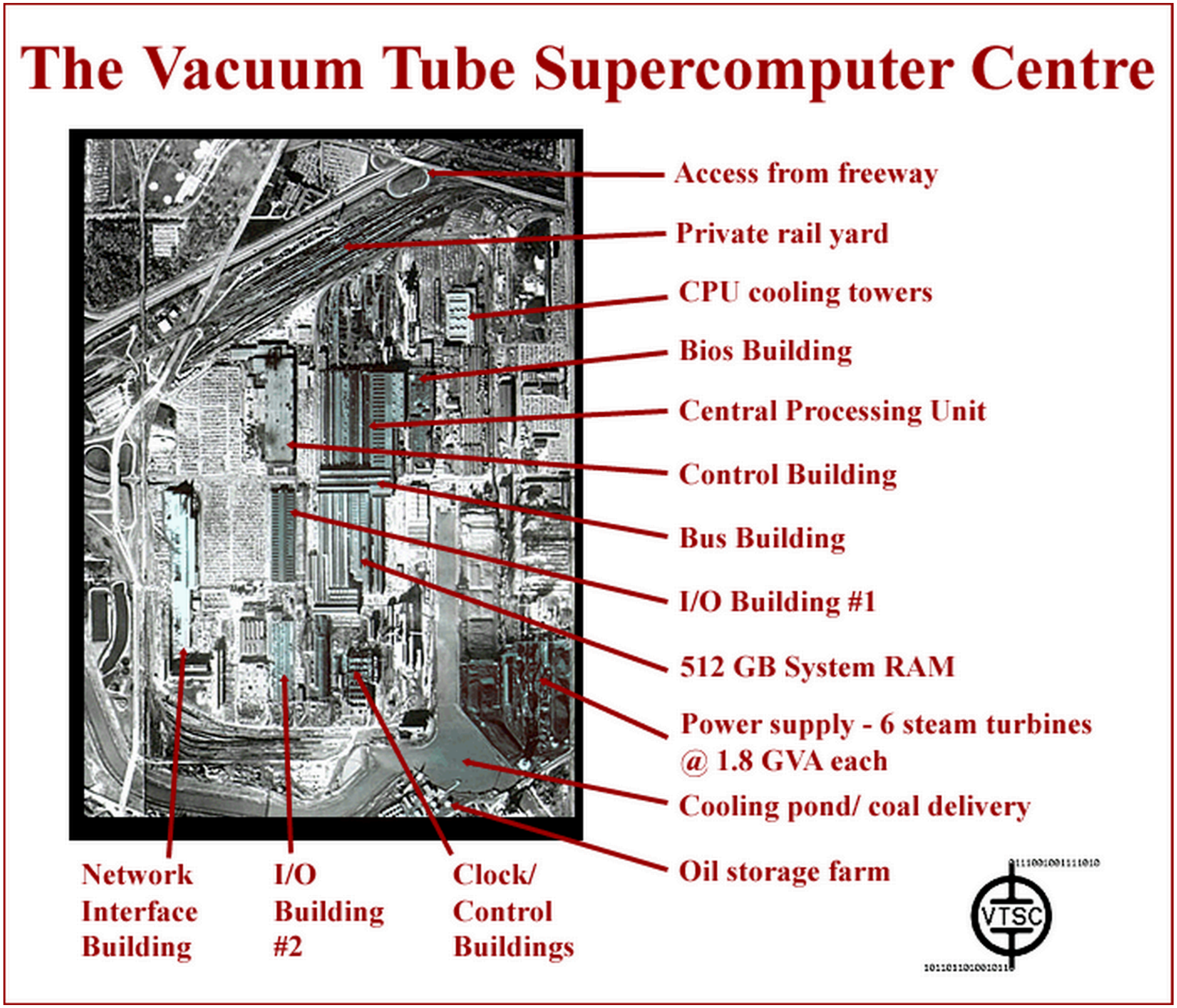
- How does Intel manage so many designers on one project?
  - Architects
  - Microarchitects
  - Circuit designers
  - Validation
  - Software

- This trend is exponential. What does this mean?

Summary

- Read Chapter 1 in H&P
- Abstraction in HW and SW to manage complexity
- ISA defines the Hardware Software Interface
- Technology influences implementations…

Poor choice of device technology:



http://www.ominous-valve.com/vtsc.html