# Lecture 6: Instruction Set Architecture and the 80x86

Professor Randy H. Katz

Computer Science 252

Spring 1996

# Review From Last Time

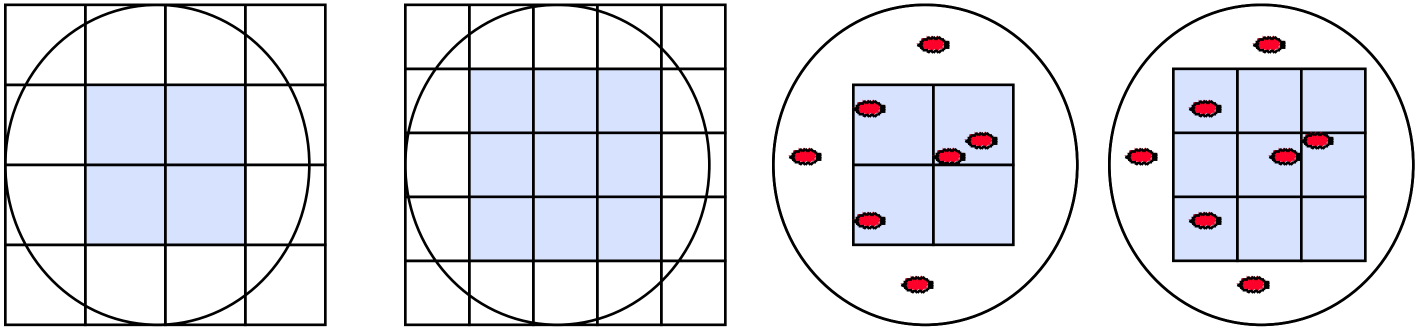
- Given sales a function of performance relative to competition, tremendous investment in improving product as reported by performance summary
- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- If not good benchmarks and summary, then choice between improving product for real programs vs. improving product to get more sales=> sales almost always wins
- Time is the measure of computer performance!
- What about cost?

# Review: Integrated Circuits Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} * \text{Die yield}}$$

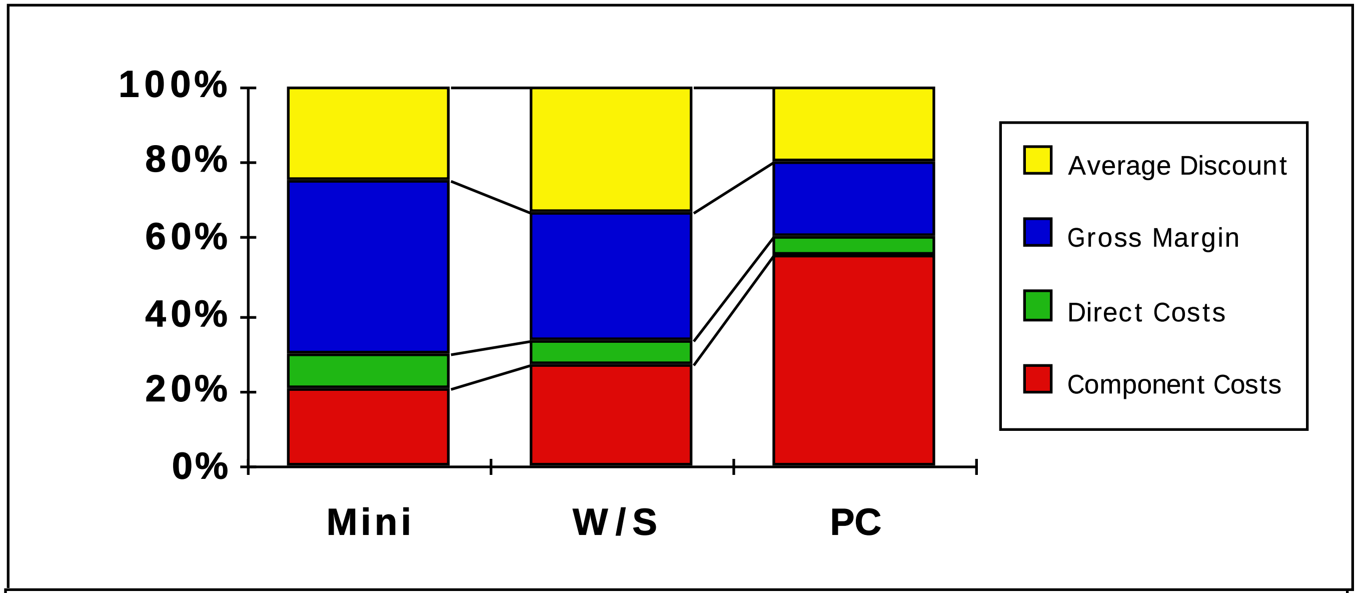
$$\text{Dies per wafer} = \frac{\pi * (\text{Wafer\_diam} / 2)^2}{\text{Die Area}} - \frac{\pi * \text{Wafer\_diam}}{\sqrt{2 * \text{Die Area}}} - \text{Test dies}$$

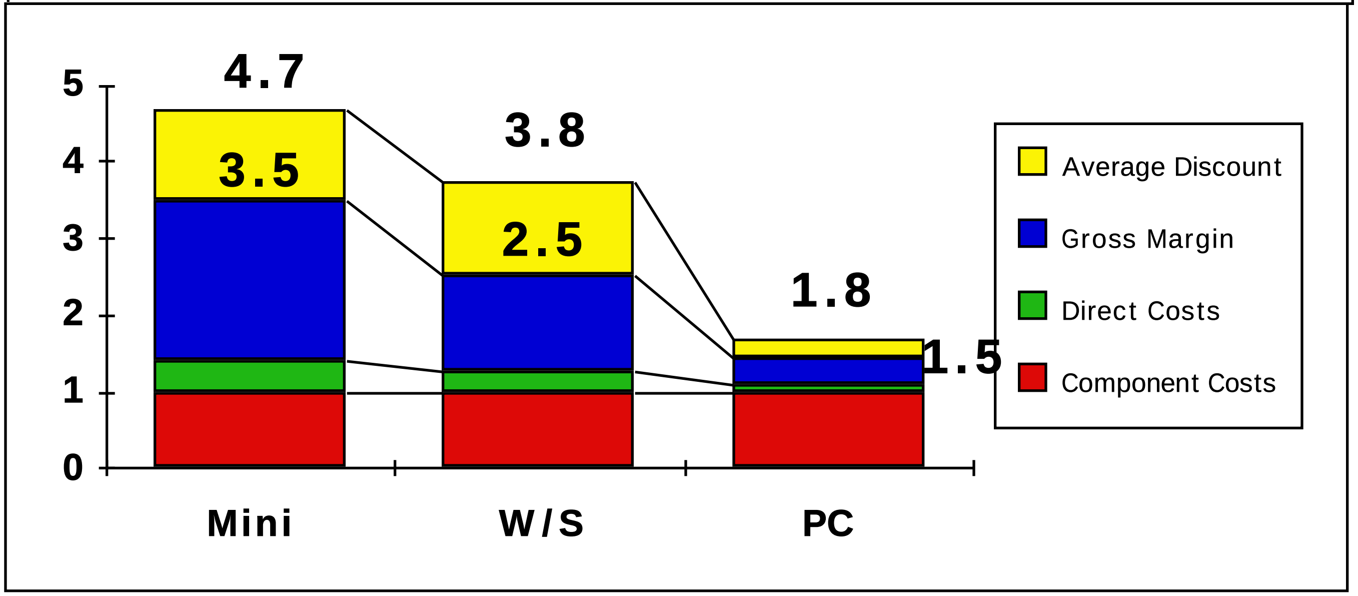


$$\text{Die Yield} = \text{Wafer yield} * \left\{1 + \frac{\text{Defects\_per\_unit\_area} * \text{Die\_Area}}{\alpha}\right\}^{-\alpha}$$

**Die Cost is goes roughly with area$^4$**

# Review From Last Time
# Price vs. Cost





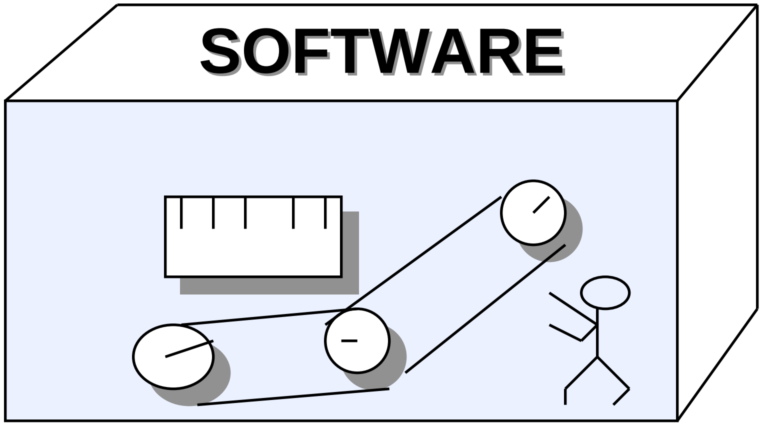
# Today: Instruction Set Architecture

- 1950s to 1960s: Computer Architecture Course Computer Arithmetic
- 1970 to mid 1980s:  Computer Architecture Course Instruction Set Design, especially ISA appropriate for compilers
- 1990s: Computer Architecture Course Design of CPU, memory system, I/O system, Multiprocessors

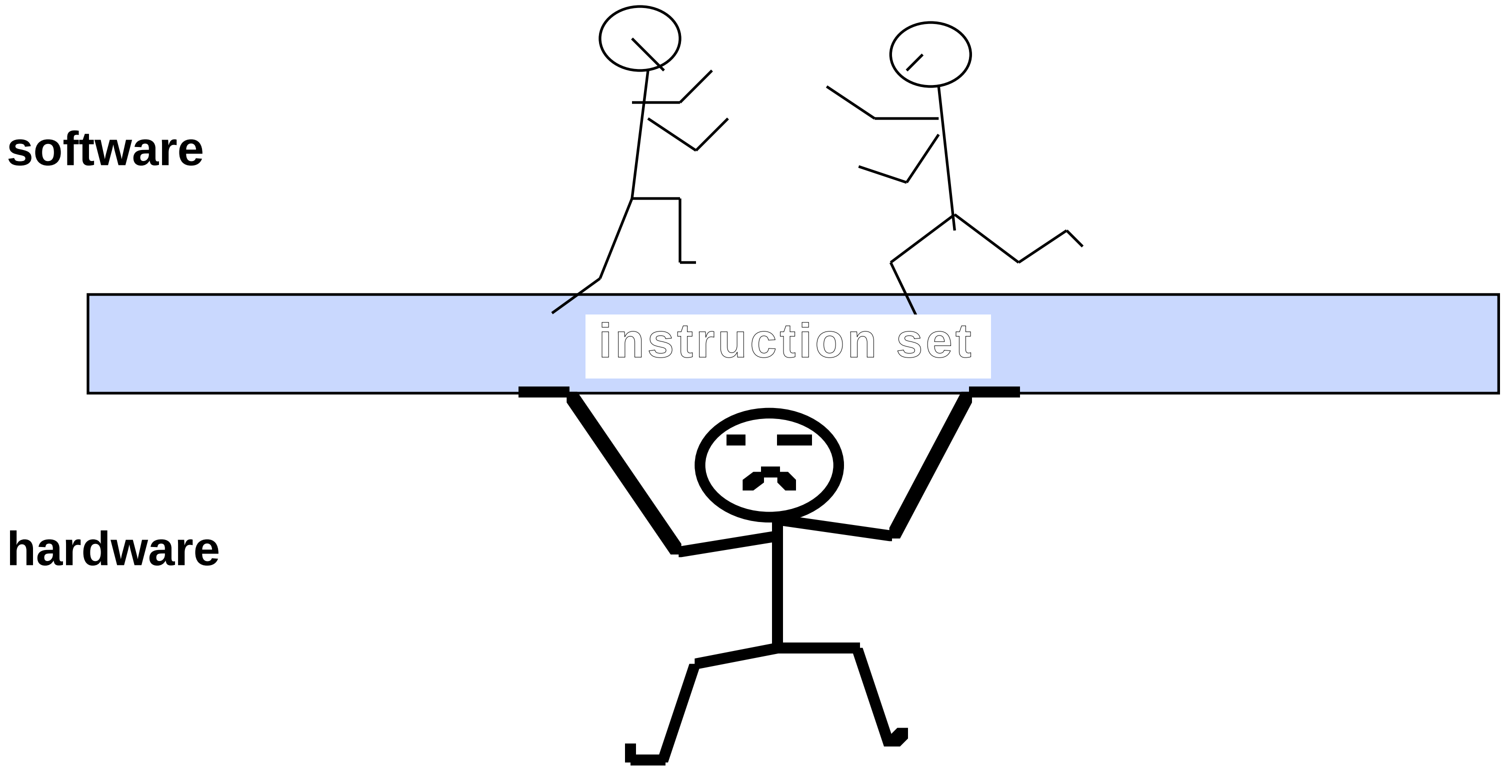
# Computer Architecture?

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

**Amdahl, Blaaw, and Brooks, 1964**



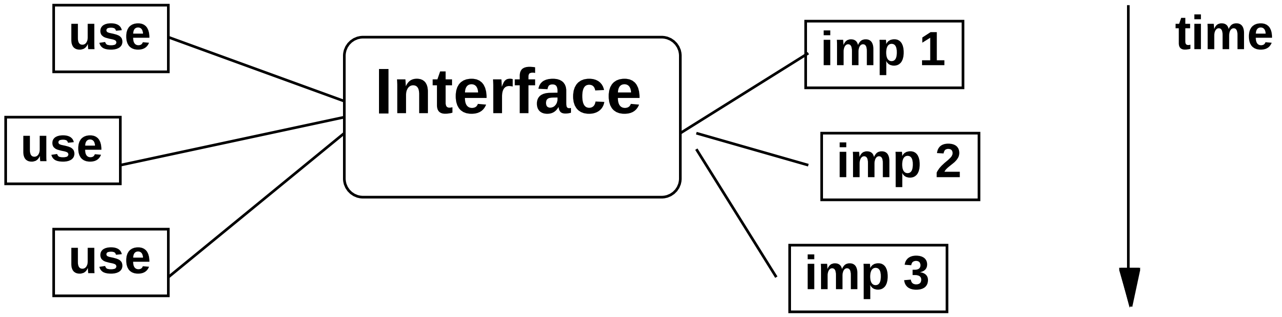
# Towards Evaluation of ISA and Organization



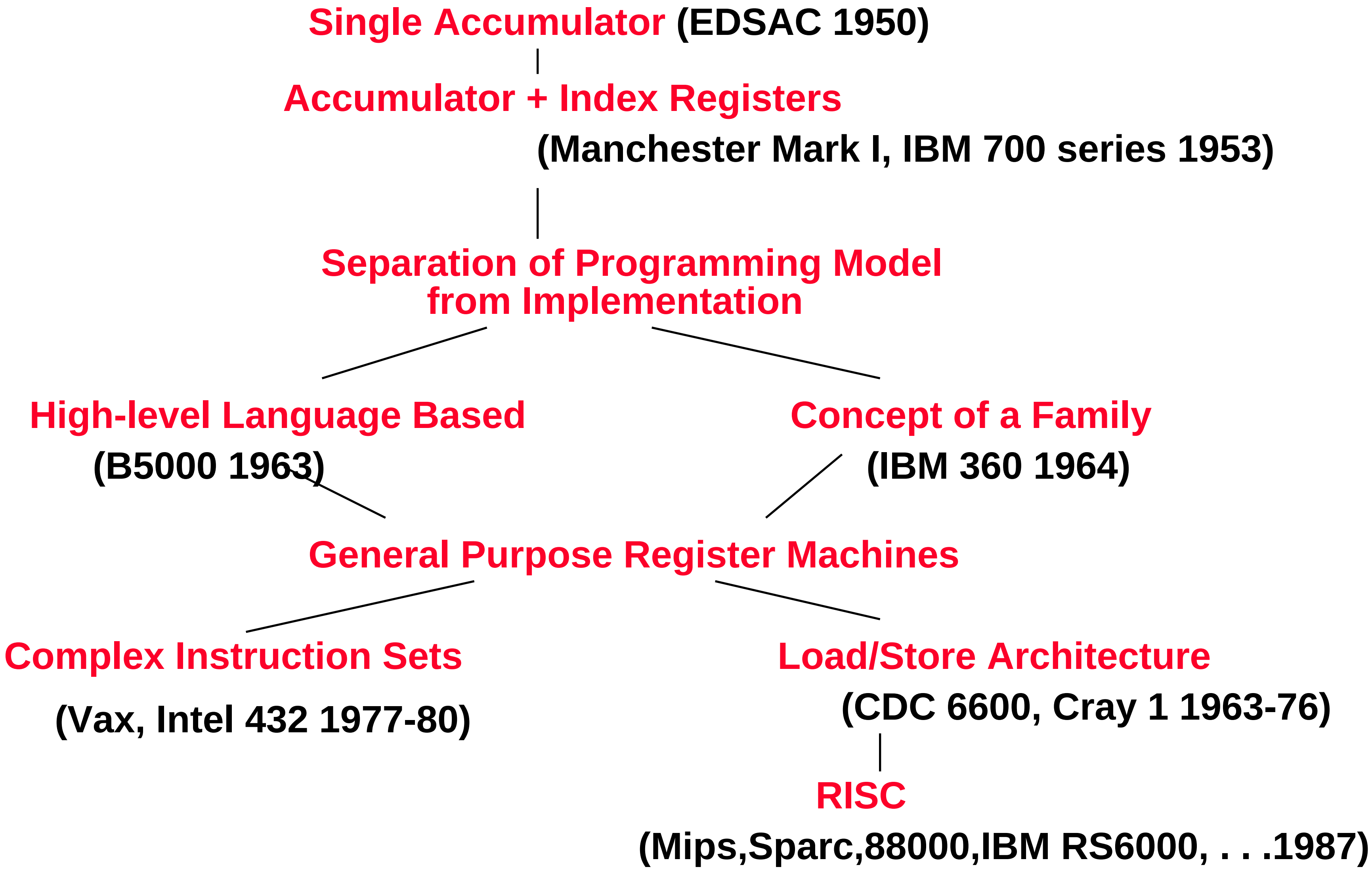
# Interface Design

A good interface:

- Lasts through many implementations (portability, compatability)
- Is used in many differeny ways (generality)
- Provides convenient  functionality to higher levels
- Permits an efficient implementation at lower levels



# Evolution of Instruction Sets



Single Accumulator (EDSAC 1950)

Accumulator + Index Registers

(Manchester Mark I, IBM 700 series 1953)

Separation of Programming Model
from Implementation

High-level Language Based

(B5000 1963)

Concept of a Family

(IBM 360 1964)

General Purpose Register Machines

Complex Instruction Sets

(Vax, Intel 432 1977-80)

Load/Store Architecture

(CDC 6600, Cray 1 1963-76)

RISC

(Mips,Sparc,88000,IBM RS6000, . . .1987)

# Evolution of Instruction Sets

- Major advances in computer architecture are typically associated with landmark instruction set designs
  - Ex: Stack vs GPR (System 360)
- Design decisions must take into account:
  - technology
  - machine organization
  - programming langauges
  - compiler technology
  - operating systems
- And they in turn influence these

# Design Space of ISA

## Five Primary Dimensions

| | |
|---|---|
| • Number of explicit operands | ( 0, 1, 2, 3 ) |
| • Operand Storage | Where besides memory? |
| • Effective Address | How is memory location specified? |
| • Type & Size of Operands | byte, int, float, vector, . . . |
| | How is it specified? |
| • Operations | add, sub, mul, . . . |
| | How is it specifed? |

## Other Aspects

| | |
|---|---|
| • Successor | How is it specified? |
| • Conditions | How are they determined? |
| • Encodings | Fixed or variable? Wide? |
| • Parallelism | |

# ISA Metrics

Aesthetics:

- Orthogonality
  - No special registers, few special cases, all operand modes available with any data type or instruction type
- Completeness
  - Support for a wide range of operations and target applications
- Regularity
  - No overloading for the meanings of instruction fields
- Streamlined
  - Resource needs easily determined

Ease of compilation (programming?)

Ease of implementation

Scalability

# Basic ISA Classes

## Accumulator:

| | | |
|---|---|---|
| 1 address | add A | acc ← acc + mem[A] |
| 1+x address | addx A | acc ← acc + mem[A + x] |

## Stack:

| | | |
|---|---|---|
| 0 address | add | tos ← tos + next |

## General Purpose Register:

| | | |
|---|---|---|
| 2 address | add A B | EA(A) ← EA(A) + EA(B) |
| 3 address | add A B C | EA(A) ← EA(B) + EA(C) |

## Load/Store:

| | | |
|---|---|---|
| 3 address | add Ra Rb Rc | Ra ← Rb + Rc |
| | load Ra Rb | Ra ← mem[Rb] |
| | store Ra Rb | mem[Rb] ← Ra |

# Stack Machines

- **Instruction set:**

  **+, -, *, /, . . .**

  **push A, pop A**

- **Example: a*b - (a+c*b)**

  **push a**

  **push b**

  *****

  **push a**

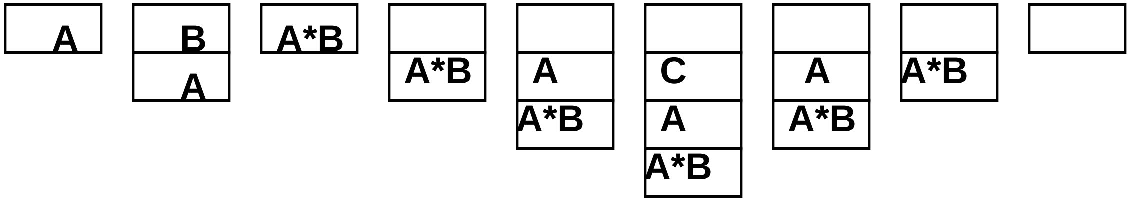
  **push c**

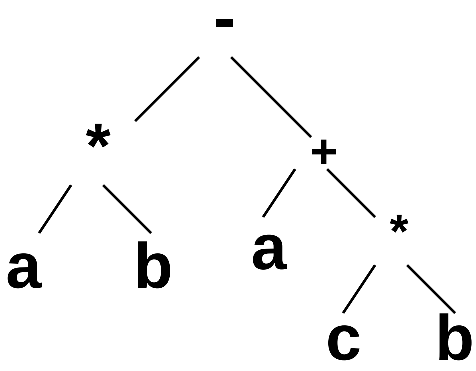
  **push b**

  *****

  **+**

  **-**





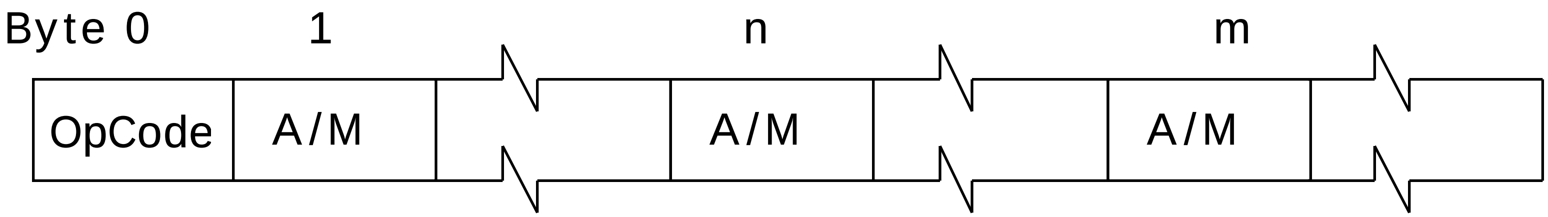
# The Case Against Stacks

- Performance is derived from the existence of several fast registers, not from the way they are organized
- Data does not always “surface” when needed
  - Constants, repeated operands, common subexpressions

  so TOP and Swap instructions are required
- Code density is about equal to that of GPR instruction sets
  - Registers have short addresses
  - Keep things in registers and reuse them
- Slightly simpler to write a poor compiler, but not an optimizing compiler

# VAX-11

## Variable format, 2 and 3 address instruction



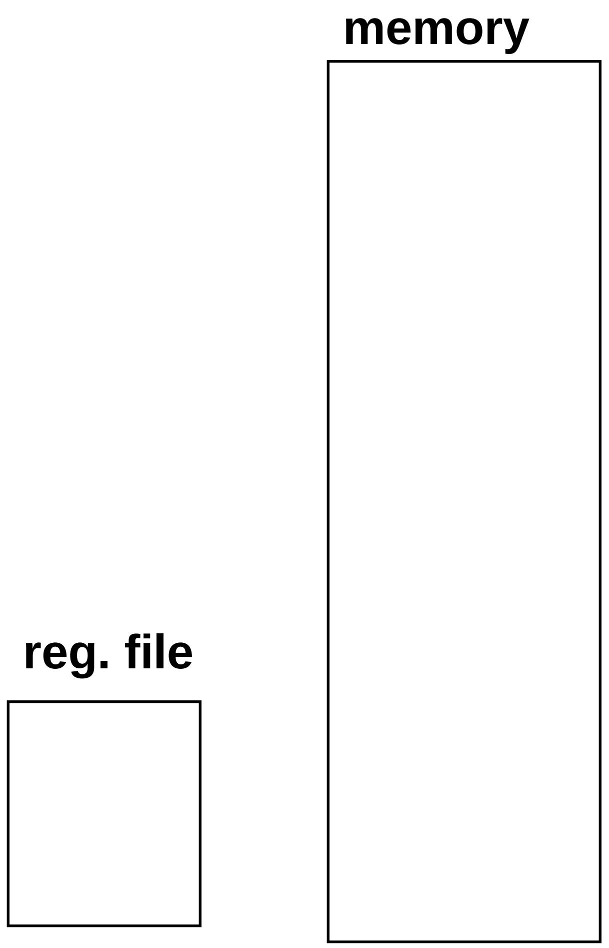
Byte 0 1 n m

OpCode A/M A/M A/M

- 32-bit word size, 16 GPR (four reserved)
- Rich set of addressing modes (apply to any operand)
- Rich set of operations
  - bit field, stack, call, case, loop, string, poly, system
- Rich set of data types (B, W, L, Q, O, F, D, G, H)
- Condition codes

# Kinds of Addressing Modes

| | |
|---|---|
| • Register direct | Ri |
| • Immediate (literal) | v |
| • Direct (absolute) | M[v] |
| | |
| • Register indirect | M[Ri] |
| • Base+Displacement | M[Ri + v] |
| • Base+Index | M[Ri + Rj] |
| • Scaled Index | M[Ri + Rj*d + v] |
| • Autoincrement | M[Ri++] |
| • Autodecrement | M[Ri - -] |
| | |
| • Memory Indirect | M[ M[Ri] ] |
| | |
| • [Indirection Chains] | |



| | | |
|---|---|---|
| Ri | Rj | v |

# A "Typical" RISC

- 32-bit fixed format instruction (3 formats)
- 32 32-bit GPR (R0 contains zero, DP take pair)
- 3-address, reg-reg arithmetic instruction
- Single address mode for load/store:
  base + displacement
  - no indirection
- Simple branch conditions
- Delayed branch

see: SPARC, MIPS, MC88100, AMD2900, i960, i860
PARisc, DEC Alpha, Clipper,
CDC 6600, CDC 7600, Cray-1, Cray-2, Cray-3

# Example: MIPS

## Register-Register

| 31 26 | 25 21 | 20 16 | 15 11 | 10 6 | 5 0 |
|---|---|---|---|---|---|
| Op | Rs1 | Rs2 | Rd | | Opx |

## Register-Immediate

| 31 26 | 25 21 | 20 16 | 15 0 |
|---|---|---|---|
| Op | Rs1 | Rd | immediate |

## Branch

| 31 26 | 25 21 | 20 16 | 15 0 |
|---|---|---|---|
| Op | Rs1 | Rs2/Opx | immediate |

## Jump / Call

| 31 26 | 25 0 |
|---|---|
| Op | target |

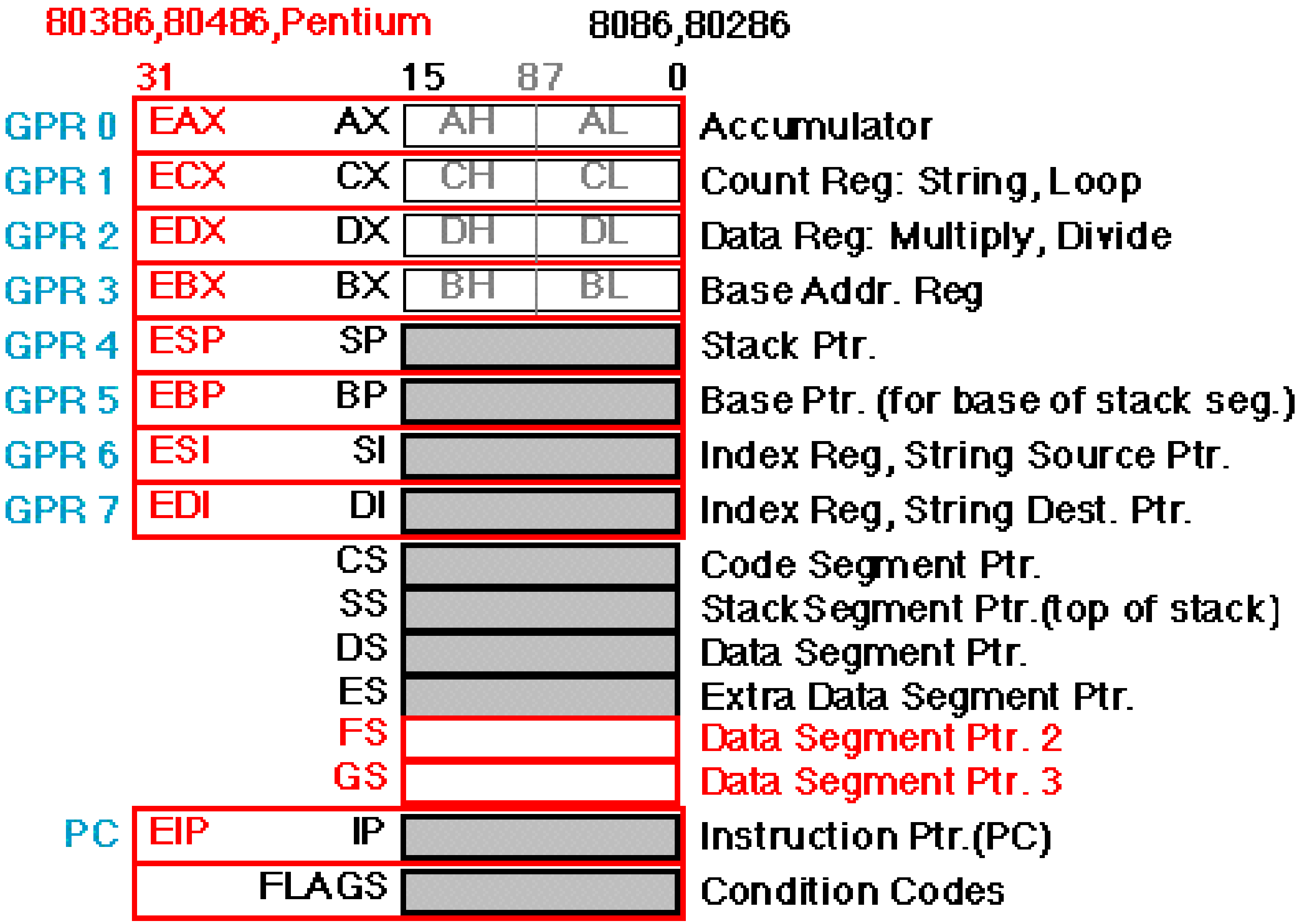
# Most Popular ISA of All Time: The Intel 80x86

- 1971: Intel invents microprocessor 4004/8008, 8080 in 1975
- 1975: Gordon Moore realized one more chance for new ISA before ISA locked in for decades
  - Hired CS people in Oregon
  - Weren’t ready in 1977 (CS people did 432 in 1980)
  - Started crash effort for 16-bit microcomputer
- 1978: 8086 dedicated registers, segmented address, 16 bit
  - 8088; 8-bit external bus version of 8086; added as after thought

# Most Popular ISA of All Time: The Intel 80x86

- 1980: IBM selects 8088 as basis for IBM PC
- 1980: 8087 floating point coprocessor:
  adds 60 instructions using hybrid stack/register scheme
- 1982: 80286 24-bit address, protection, memory mapping
- 1985: 80386 32-bit address, 32-bit GP registers, paging
- 1989: 80486 & Pentium in 1992: faster + MP few instructions

# Intel 80x86 Integer Registers

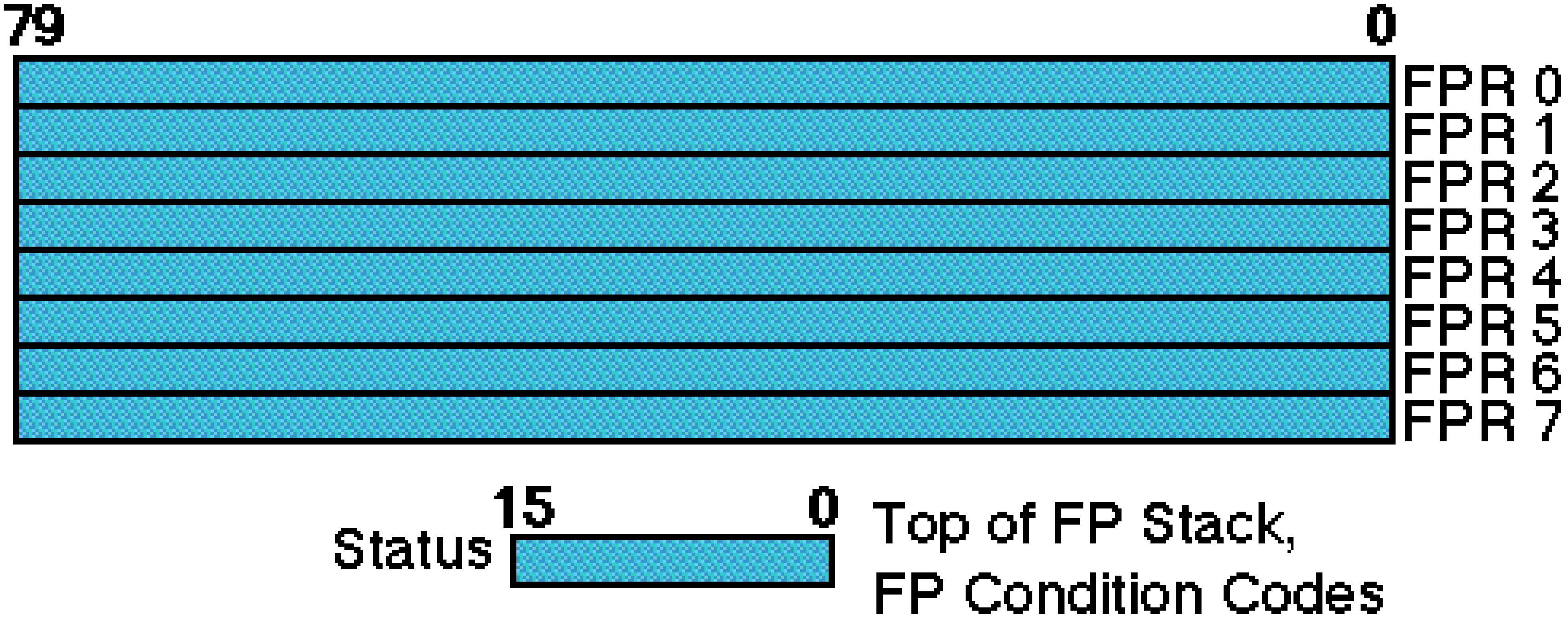


80386,80486,Pentium 8086,80286

31 15 87 0

| | | | | | |
|---|---|---|---|---|---|
| GPR 0 | EAX | AX | AH | AL | Accumulator |
| GPR 1 | ECX | CX | CH | CL | Count Reg: String, Loop |
| GPR 2 | EDX | DX | DH | DL | Data Reg: Multiply, Divide |
| GPR 3 | EBX | BX | BH | BL | Base Addr. Reg |
| GPR 4 | ESP | SP | | | Stack Ptr. |
| GPR 5 | EBP | BP | | | Base Ptr. (for base of stack seg.) |
| GPR 6 | ESI | SI | | | Index Reg, String Source Ptr. |
| GPR 7 | EDI | DI | | | Index Reg, String Dest. Ptr. |
| | | CS | | | Code Segment Ptr. |
| | | SS | | | Stack Segment Ptr.(top of stack) |
| | | DS | | | Data Segment Ptr. |
| | | ES | | | Extra Data Segment Ptr. |
| | | FS | | | Data Segment Ptr. 2 |
| | | GS | | | Data Segment Ptr. 3 |
| PC | EIP | IP | | | Instruction Ptr.(PC) |
| | | FLAGS | | | Condition Codes |

# Intel 80x86 Floating Point Registers



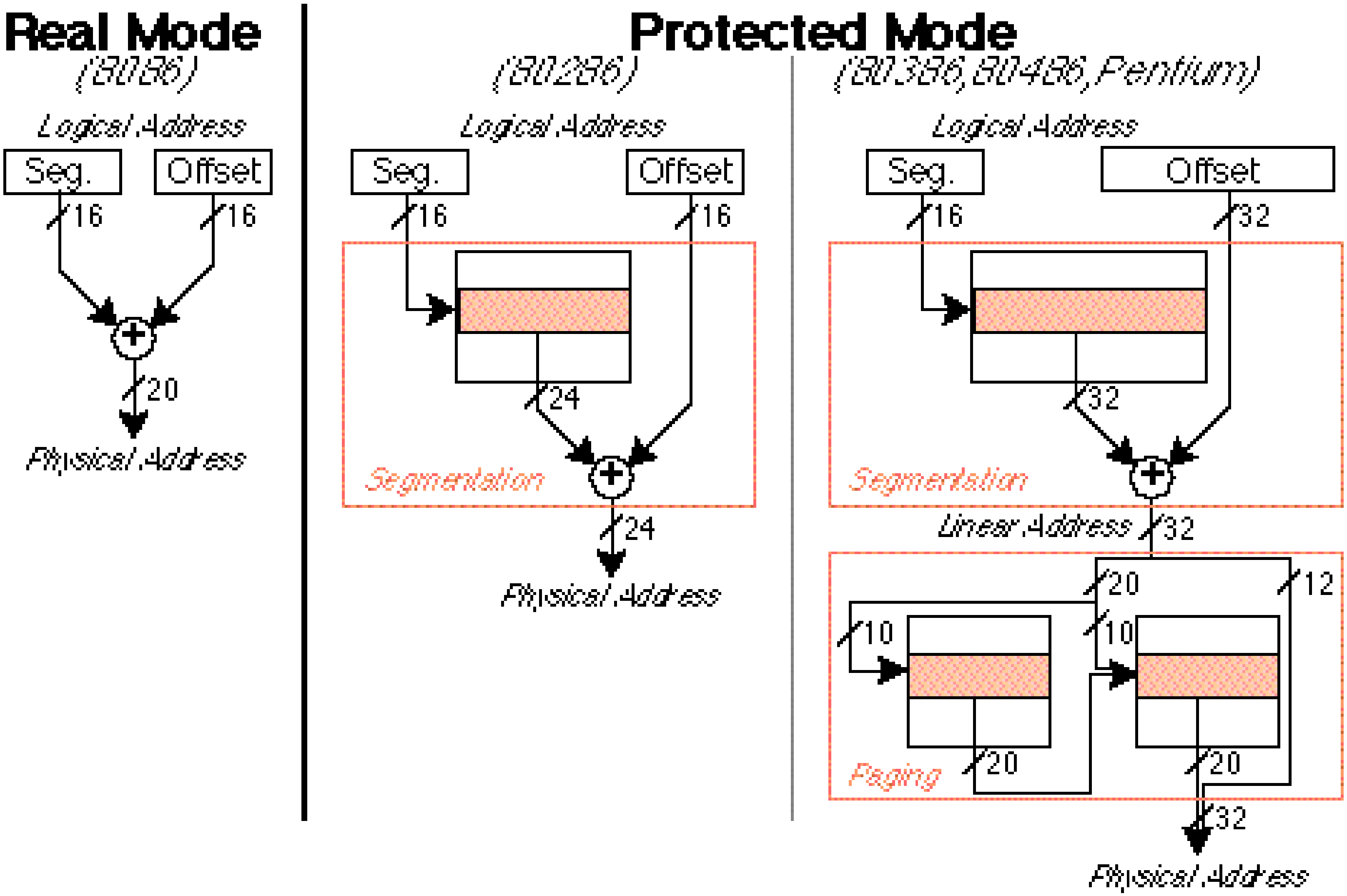
# Usage of Intel 80x86 Floating Point Registers

| | NASA 7 | Spice |
|---|---|---|
| Stack (2nd operand ST(1)) | 0.3% | 2.0% |
| Register (2nd operand ST(i), i>1) | 23.3% | 8.3% |
| Memory | 76.3% | 89.7% |

Above are *dynamic* instruction percentages (i.e., based on counts of executed instructions)

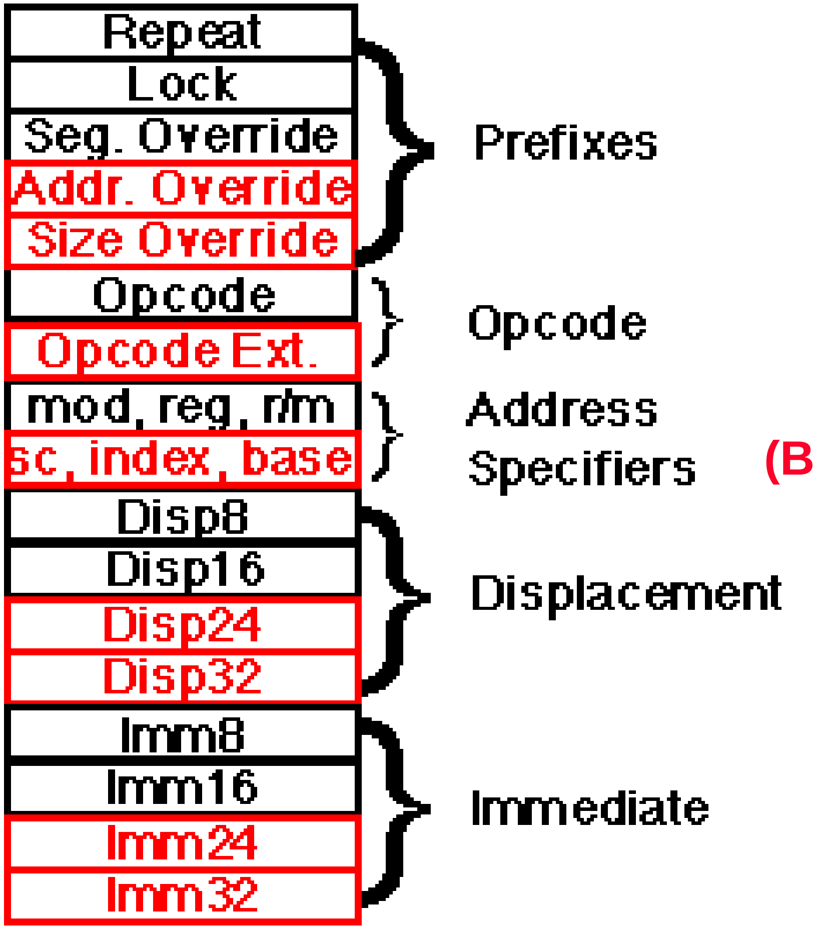
Stack unused by Solaris compilers for fastest execution

# 80x86 Addressing/Protection



# 80x86 Instruction Format

- 8086 in black; 80386 extensions in color



Repeat

Lock

Seg. Override

Addr. Override

Size Override

Prefixes

Opcode

Opcode Ext.

Opcode

mod, reg, r/m

sc, index, base

Address Specifiers

(Base reg + $2^{\text{Scale}}$ x Index reg)

Disp8

Disp16

Disp24

Disp32

Displacement

Imm8

Imm16

Imm24

Imm32

Immediate

# 80x86 Instruction Encoding: Mod, Reg, R/M Field

| r | w=0 | w=1 | | r/m | mod=0 | | mod=1 | | mod=2 | | mod=3 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 16b | 32b | | 16b | 32b | 16b | 32b | 16b | 32b | |
| 0 | AL | AX | EAX | 0 | addr=BX+SI | =EAX | *same* | *same* | *same* | *same* | *same* |
| 1 | CL | CX | ECX | 1 | addr=BX+DI | =ECX | *addr* | *addr* | *addr* | *addr* | *as* |
| 2 | DL | DX | EDX | 2 | addr=BP+SI | =EDX | *mod=0* | *mod=0* | *mod=0* | *mod=0* | *reg* |
| 3 | BL | BX | EBX | 3 | addr=BP+SI | =EBX | *+d8* | *+d8* | *+d16* | *+d32* | *field* |
| 4 | AH | SP | ESP | 4 | addr=SI | =(sib) | SI+d8 | (sib)+d8 | SI+d8 | (sib)+d32 | “ |
| 5 | CH | BP | EBP | 5 | addr=DI | =d32 | DI+d8 | EBP+d8 | DI+d16 | EBP+d32 | “ |
| 6 | DH | SI | ESI | 6 | addr=d16 | =ESI | BP+d8 | ESI+d8 | BP+d16 | ESI+d32 | “ |
| 7 | BH | DI | EDI | 7 | addr=BX | =EDI | BX+d8 | EDI+d8 | BX+d16 | EDI+d32 | “ |

w from opcode

r/m field depends on mod and machine mode

**First address specifier: Reg=3 bits, R/M=3 bits, Mod=2 bits**

# 80x86 Instruction Encoding
# Sc/Index/Base field

| | *Index* | *Base* |
|---|---|---|
| 0 | EAX | EAX |
| 1 | ECX | ECX |
| 2 | EDX | EDX |
| 3 | EBX | EBX |
| 4 | no index | ESP |
| 5 | EBP | if mod=0, d32<br>if mod≠0, EBP |
| 6 | ESI | ESI |
| 7 | EDI | EDI |

## Base + Scaled Index Mode

**Used when:**

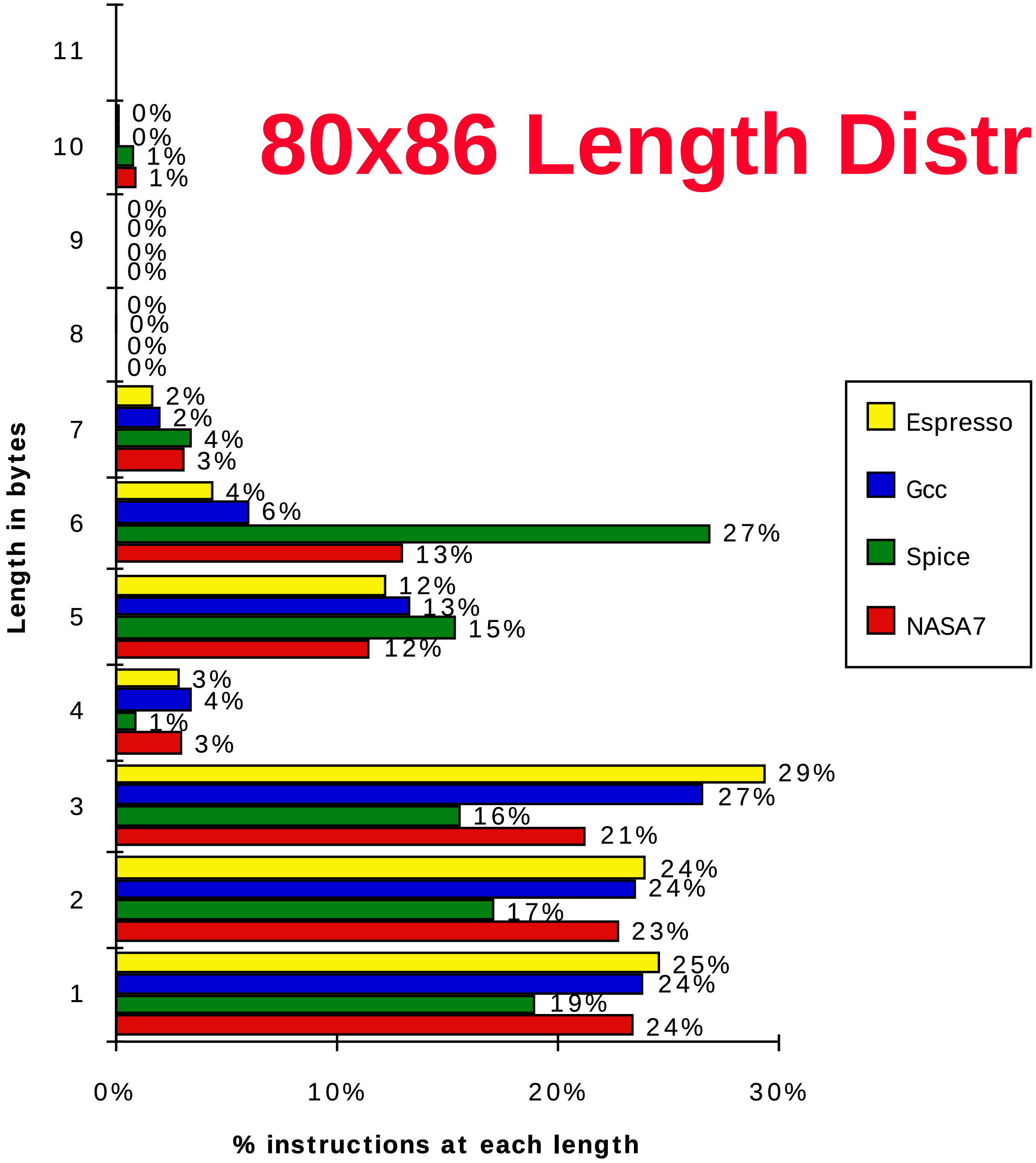
**mod = 0,1,2**
**in 32-bit mode**
**AND r/m = 4!**

**2-bit Scale Field**
**3-bit Index Field**
**3-bit Base Field**

# 80x86 Addressing Mode Usage for 32-bit Mode

| *Addressing Mode* | *Gcc* | *Espr.* | *NASA7* | *Spice* | *Avg.* |
|---|---|---|---|---|---|
| Register indirect | 10% | 10% | 6% | 2% | 7% |
| Base + 8-bit disp | 46% | 43% | 32% | 4% | 31% |
| Base + 32-bit disp | 2% | 0% | 24% | 10% | 9% |
| Indexed | 1% | 0% | 1% | 0% | 1% |
| Based indexed + 8b disp | 0% | 0% | 4% | 0% | 1% |
| Based indexed + 32b disp | 0% | 0% | 0% | 0% | 0% |
| Base + Scaled Indexed | 12% | 31% | 9% | 0% | 13% |
| Base + Scaled Index + 8b disp | 2% | 1% | 2% | 0% | 1% |
| Base + Scaled Index + 32b disp | 6% | 2% | 2% | 33% | 11% |
| 32-bit Direct | 19% | 12% | 20% | 51% | 26% |

# 80x86 Length Distribution



# Instruction Counts: 80x86 v. DLX

| *SPEC pgm* | *x86* | *DLX* | *DLX÷86* |
|---|---:|---:|---:|
| gcc | 3,771,327,742 | 3,892,063,460 | 1.03 |
| espresso | 2,216,423,413 | 2,801,294,286 | 1.26 |
| spice | 15,257,026,309 | 16,965,928,788 | 1.11 |
| nasa7 | 15,603,040,963 | 6,118,740,321 | 0.39 |

# Intel Compiler vs. Compilers YOU Can Buy

| 66 MHz Pentium Comparison | SpecInt92 | SpecFP92 |
|---|---|---|
| Intel Internal Optimizing Compiler | 64.6 | 59.7 |
| Best 486 Compiler (June 1993) | 57.6 | 39.9 |
| Typical 486 Compiler in 1990, when Intel started project | 41.0 | 32.5 |

- Integer Intel 1.1X faster, FP 1.5X faster

| 486 Comparison | SpecInt92 | SpecFP92 |
|---|---|---|
| Intel Internal Optimizing Compiler | 35.5 | 17.5 |
| Best 486 Compiler (June 1993) | 32.2 | 16.0 |
| Typical 486 Compiler in 1990, when Intel started project | 23.0 | 12.8 |

- Integer: Intel 1.1X faster, FP 1.1X faster

# Intel Summary

- Archeology: history of instruction design in a single product
  - Address size: 16 bit vs. 32-bit
  - Protection: Segmentation vs. paged
  - Temp. storage: accumulator vs. stack vs. registers
- “Golden Handcuffs” of binary compatibility affect design 20 years later, as Moore predicted
- Not too difficult to make faster, as Intel has shown
- HP/Intel announcement of common future instruction set by 2000 means end of 80x86???
- “Beauty is in the eye of the beholder”
  - At 50M/year sold, it is a beautiful business