Review From Last Time:

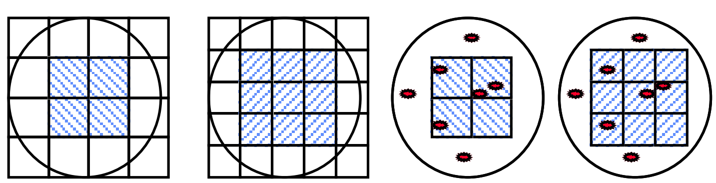
- Given sales a function of performance relative to competition, tremendous investment in improving product as reported by performance summary
- Good products created when have:
  - Good benchmarks
  - Good ways to summarize performance
- If not good benchmarks and summary, then choice between improving 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 From Last Time: Integrated Circuits Costs

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

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

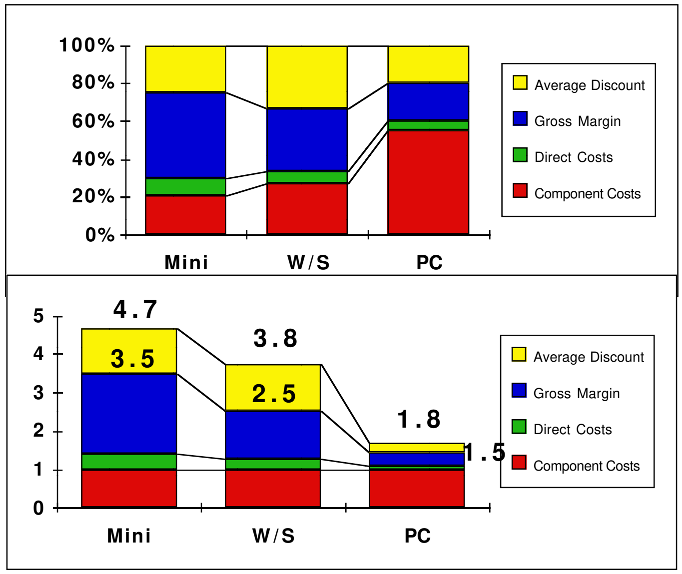
$$\text{Dies per wafer} = \frac{\pi * (\text{Wafer\_diam} / 2)^2}{\text{Die Area}} - \frac{\pi * \text{Wafer\_diam}}{\sqrt{2 * \text{Die Area}}} - \text{Test dies}$$



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

Die Cost is goes roughly with the cube of the area.

Review From Last Time:Price v. Cost:



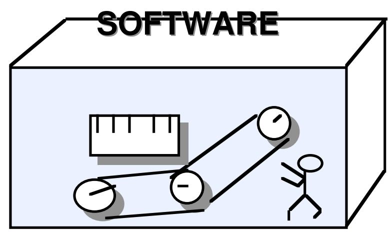
Today: Instruction Set Architecture

- **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



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)

© Culler,Patterson 1994

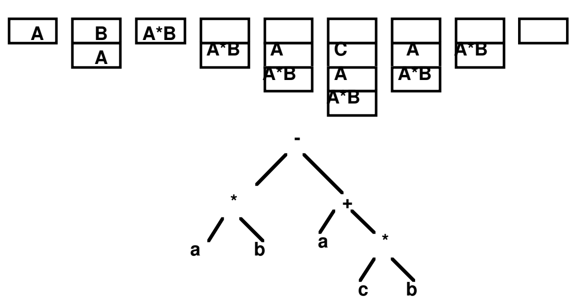
Basic ISA Classes

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
  *****
  +
  -



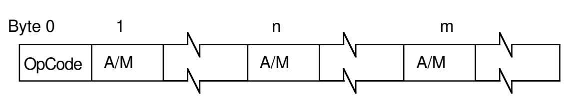
The Case Against Stacks

- 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
- Slightly simpler to write a poor compiler, but not an optimizing compiler

VAX-11

Variable format, 2 and 3 address instruction



° 32-bit word size, 16 GPR (four reserved)

° Rich set of addressing modes (apply to any operand)

° Rich set of operations

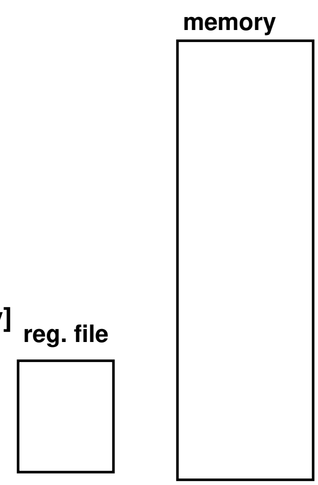
(bit field, stack, call, case, loop, string, poly, system)

° Rich set of data types (B, W, L, Q, O, F, D, G, H)

° Condition codes

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

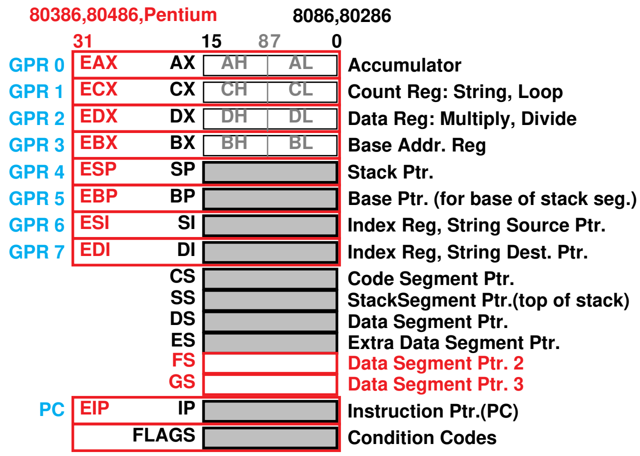
Jump / Call

31 – 26	25 – 0
Op	target

Most Popular ISA of all time: Intel 80x86

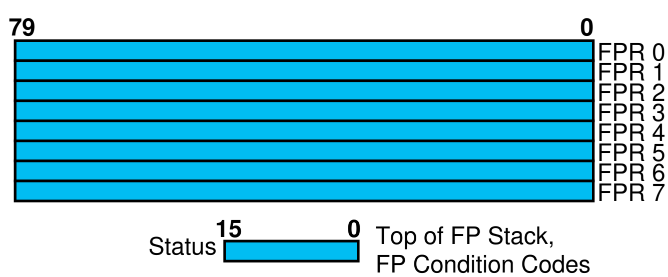
- 1971: Intel invents microprocessor 4004/8008, 8080 in 1975
- 1975: Gordon Moore realized one more chance for new ISA before ISA locked in for decades
  - hired CS people in Oregon
  - weren't ready in 1977 (CS people did 432 in 1980)
  - started crash effort for 16-bit microcomputer
- 1978: 8086 dedicated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



cs252 3.15

Intel 80x86 floating point registers



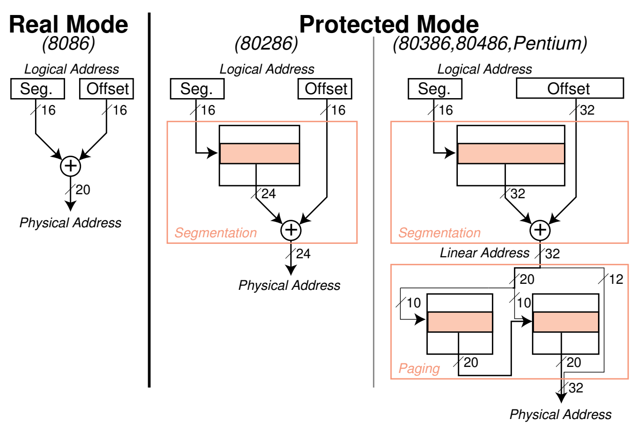
cs252 3.16

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%

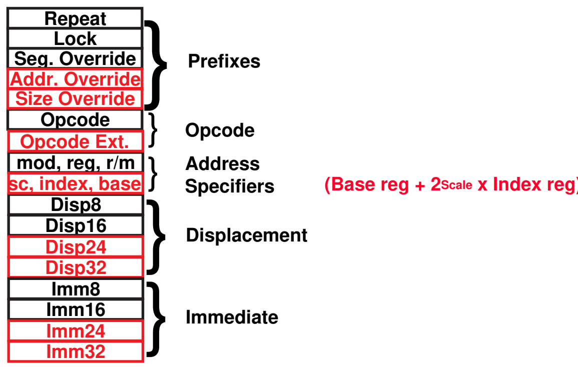
- Stack unused by Solaris compilers fo fastest execution

80x86 Addressing/Protection



80x86 Instruction Format

- 8086 in black; 80386 extensions in color



80x86 Instruction Encoding: mod, reg, r/m field

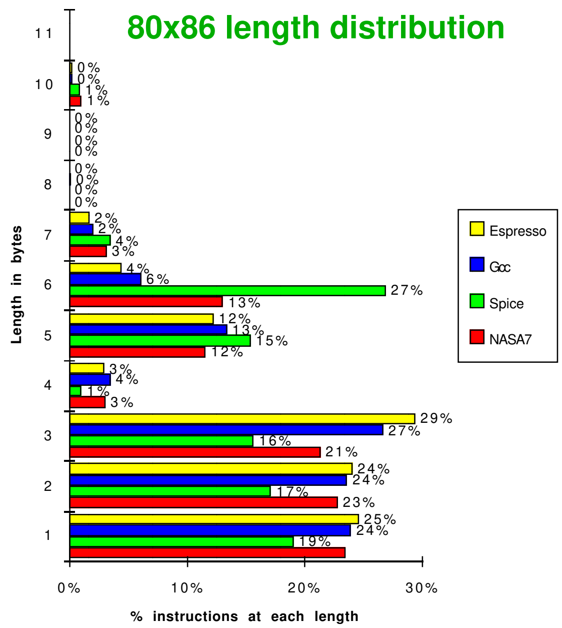
r	*w0*	*w1*		*r/m*	*mod=0*		*mod=1*		*mod=2*		*mod=3*
		16b	32b		16b	32b	16b	32b	16b	32b	
0	AL	AX	EAX	0	addr=BX+SI	=EAX	*same*	*same*	*same*	*same*	*same*
1	CL	CX	ECX	1	addr=BX+DI	=ECX	*addr*	*addr*	*addr*	*addr*	*as*
2	DL	DX	EDX	2	addr=BP+SI	=EDX	*mod=0*	*mod=0*	*mod=0*	*mod=0*	*reg*
3	BL	BX	EBX	3	addr=BP+SI	=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	"

80x86 Instruction Encoding: sc/index/base field

Sc	*Index*	*Base*
0	EAX	EAX
1	ECX	ECX
2	EDX	EDX
3	EBX	EBX
4	no index	ESP
5	EBP	if mod=0, d32 if mod$\neq$0, EBP
6	ESI	ESI
7	EDI	EDI

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%



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 compatability 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