

INTRODUCTION TO IA-32

Jo, Heeseung

Evolutionary design

- Starting in 1978 with 8086
- Added more features as time goes on
- Still support old features, although obsolete
- Totally dominate computer market

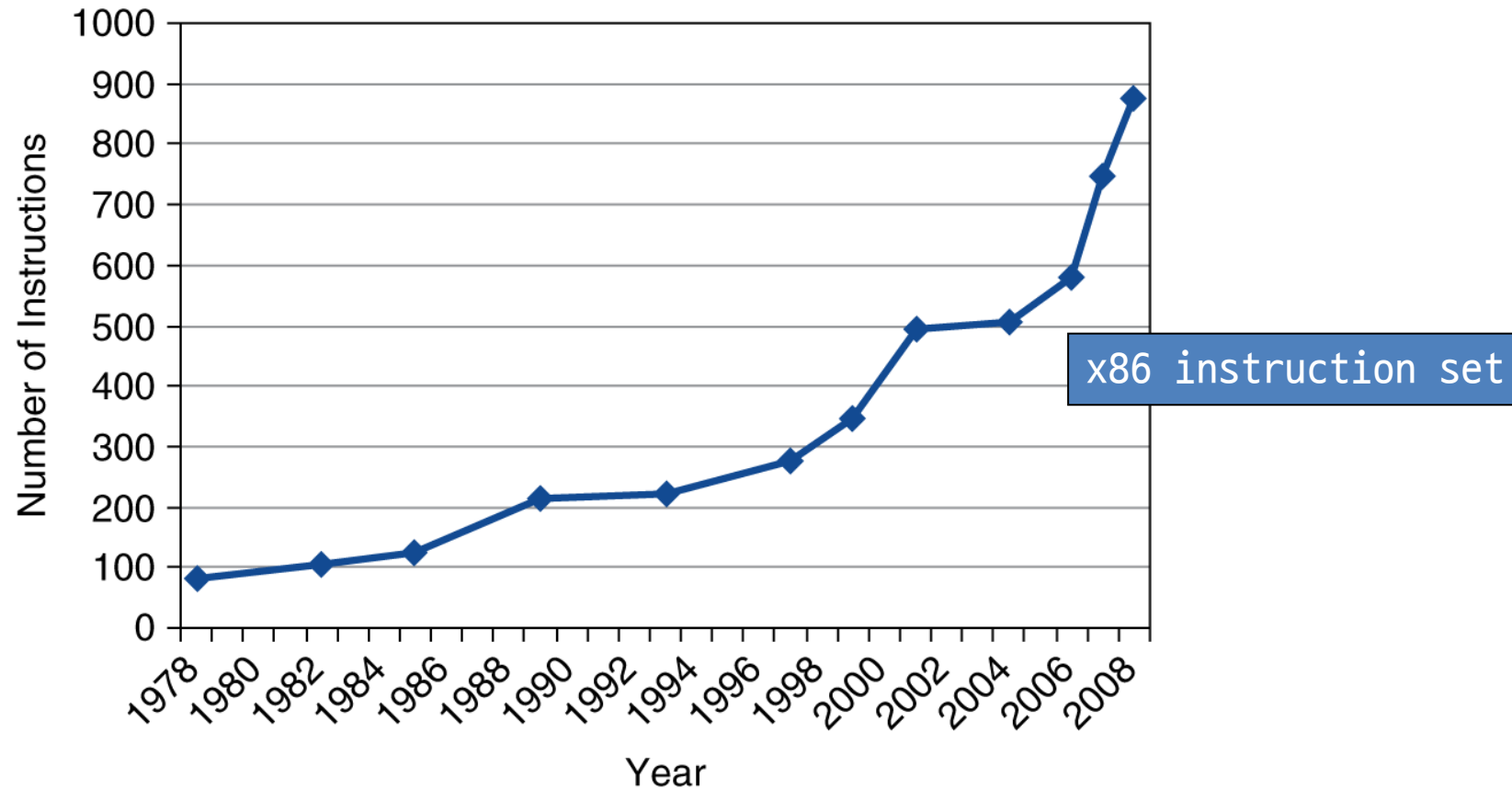
Complex Instruction Set Computer (CISC)

- Many different instructions with many different formats
- Hard to match performance of Reduced Instruction Set Computers (RISC)
- But, Intel has done just that!

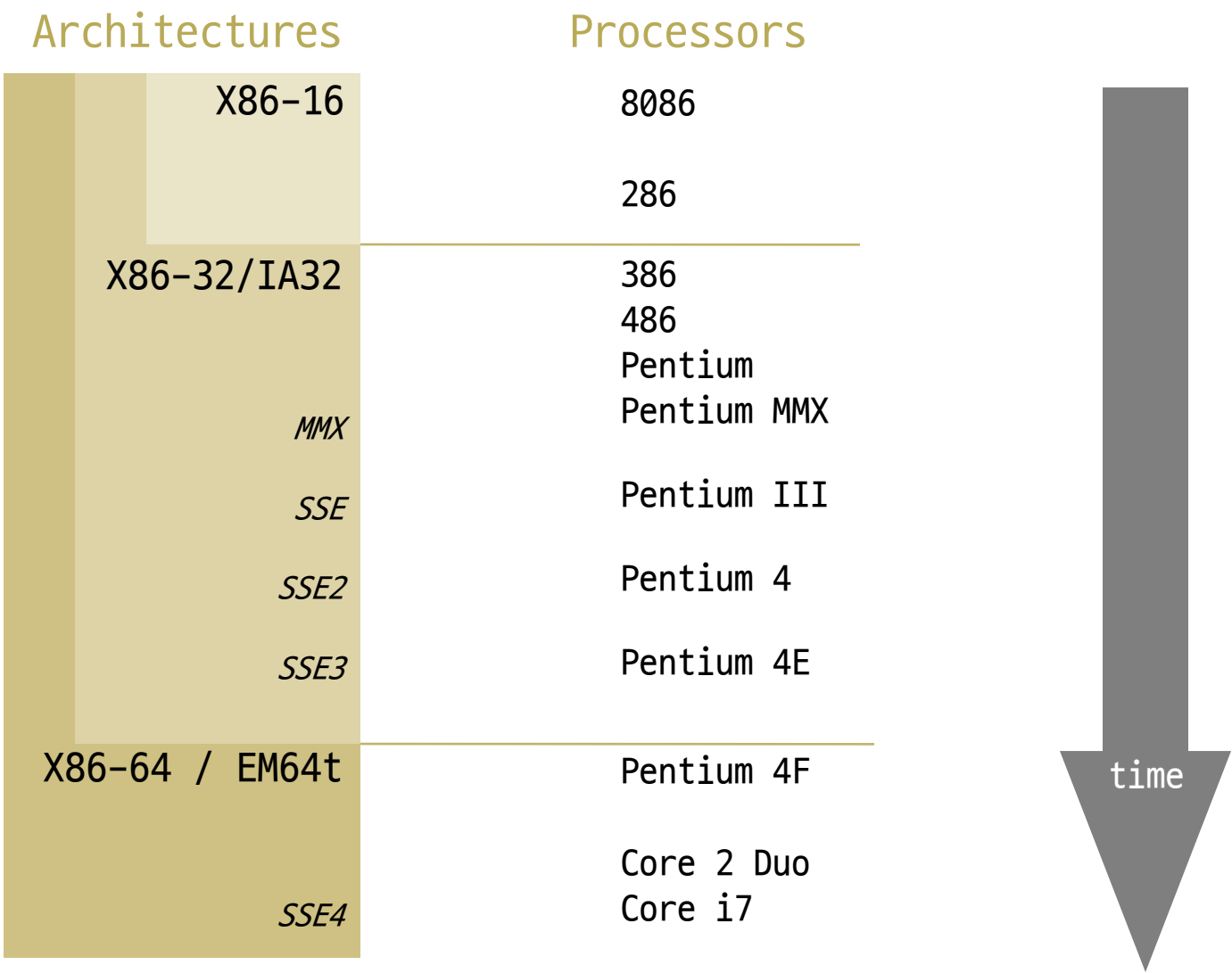
Intel's Backward Compatibility

Instruction set doesn't change

But they do accrete more instructions



Intel x86 Processors: Overview



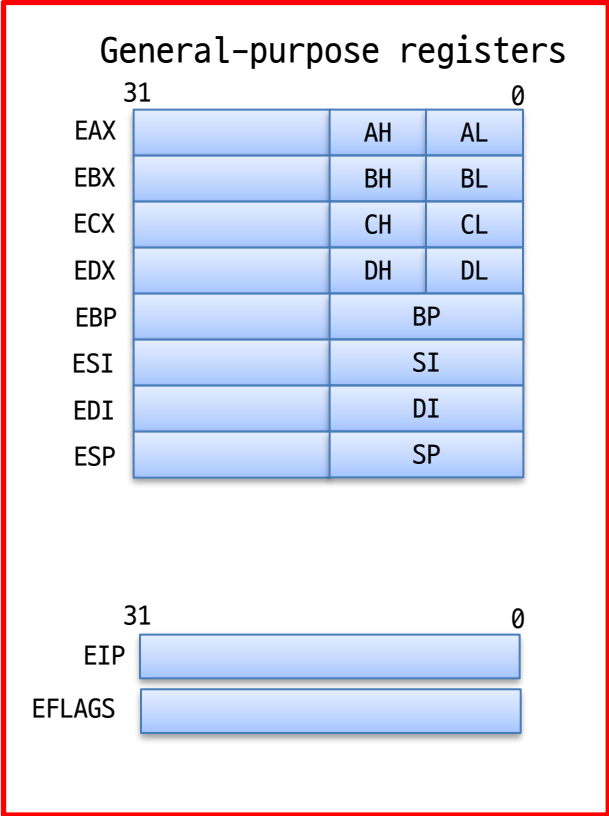
IA: often redefined as latest Intel architecture

Intel x86 Evolution: Milestones

<i>Name</i>	<i>Date</i>	<i>Transistors</i>	<i>MHz</i>
8086	1978	29K	5-10
• First 16-bit processor (Basis for IBM PC & DOS) • 1MB address space			
386	1985	275K	16-33
• First 32-bit processor, referred to as IA32 • Added "flat addressing" • Capable of running Unix • 32-bit Linux/gcc uses no instructions introduced in later models			
Pentium 4F	2004	125M	2800-3800
• First 64-bit processor, referred to as x86-64			
Core i7	2008	731M	2667-3333

Basic Execution Environment

Application Programming Registers



Segment registers

	15	0
CS	seg. selector	
DS	seg. selector	
SS	seg. selector	
ES	seg. selector	
FS	seg. selector	
GS	seg. selector	

Control registers

	31	0
CR0		
CR1		
CR2		
CR3		
CR4		

System Table Registers

	47	16	15	0
GDTR	linear base address			table limit
IDTR	linear base address			table limit

System Segment Registers

	15	0
TR	seg. selector	
LDTR	seg. selector	

Integer Registers (IA32)

				Origin (mostly obsolete)
general purpose	%eax	%ax	%ah %al	<i>accumulate</i>
	%ecx	%cx	%ch %cl	<i>counter</i>
	%edx	%dx	%dh %dl	<i>data</i>
	%ebx	%bx	%bh %bl	<i>base</i>
	%esi	%si		<i>source index</i>
	%edi	%di		<i>destination index</i>
	%esp	%sp		<i>stack pointer</i>
	%ebp	%bp		<i>base pointer</i>
16-bit virtual registers (backwards compatibility)				

General-Purpose Registers

EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP

EAX: Accumulator for operands and results data

EBX: Pointer to data in the DS segment

ECX: Counter for string and loop operations

EDX: I/O pointer

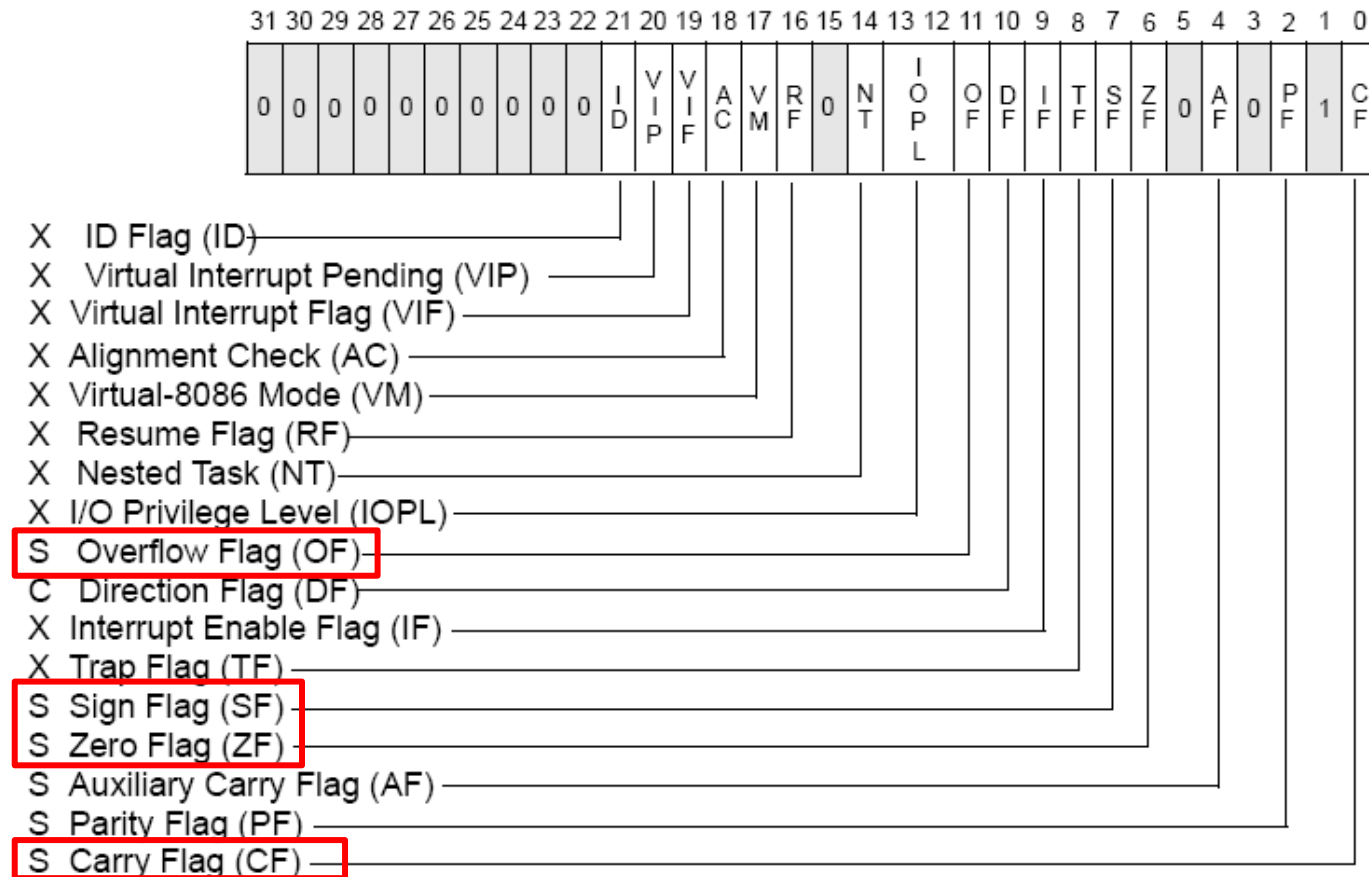
ESI: Pointer to data in the segment pointed to by the DS register; Source pointer for string operations

EDI: Pointer to data in the segment pointed to by the ES register; Destination pointer for string operations

ESP: Stack pointer (in the SS segment)

EBP: Pointer to data on the stack (in the SS segment)

EFLAGS Register (1)



S Indicates a Status Flag
C Indicates a Control Flag
X Indicates a System Flag

Reserved bit positions. DO NOT USE.
Always set to values previously read.

EFLAGS Register (2)

Status flags

- CF (Carry):** set if an arithmetic operation generates a carry or a borrow; indicates an overflow condition for unsigned-integer arithmetic
- ZF (Zero):** set if the result is zero
- SF (Sign):** set equal to the most-significant bit of the result
- OF (Overflow):** set if the integer result is too large a positive number or too small a negative number to fit in the destination operand; indicates an overflow condition for signed-integer arithmetic
- PF (Parity):** set if the least-significant byte of the result contains an even number of 1 bits
- AF (Adjust):** set if an arithmetic operation generates a carry or a borrow out of bit 3 of the result; used in binary-coded decimal (BCD) arithmetic
- DF (Direction):** setting the DF causes the string instructions to auto-decrement; set and cleared by STD/CLD instructions

Instruction Pointer

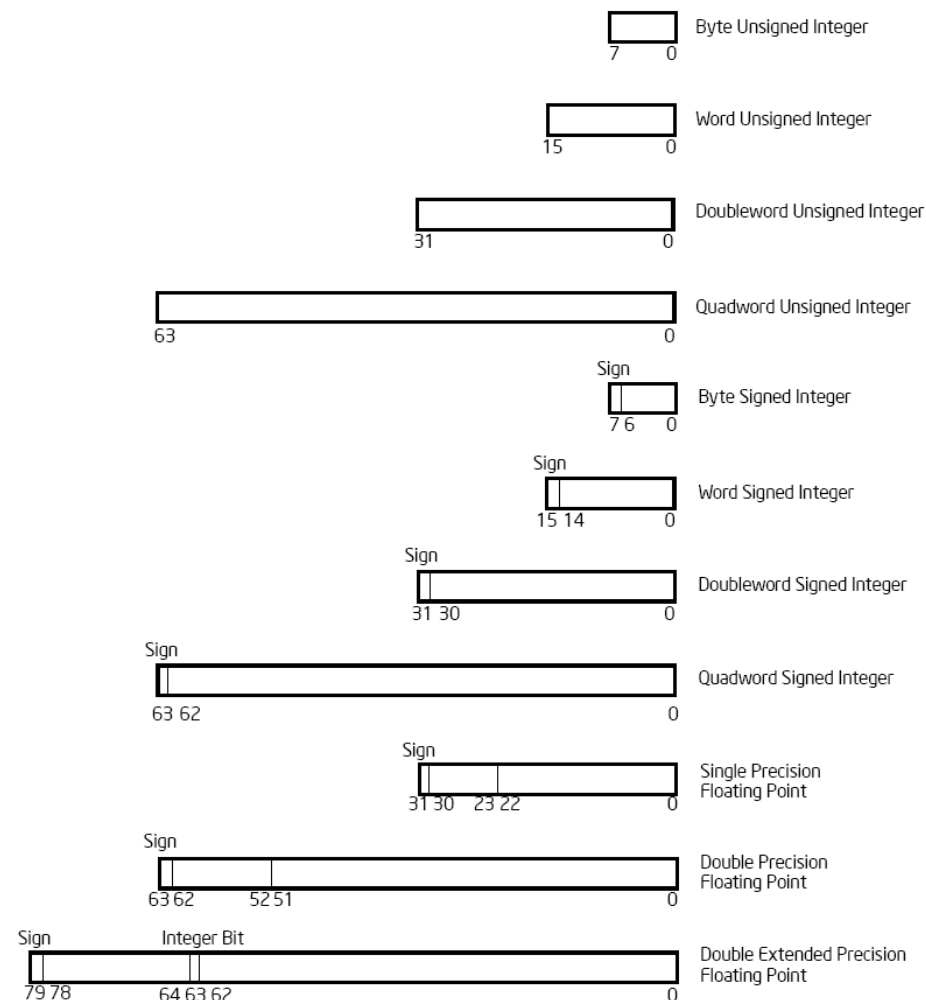
EIP Register (Program Counter, PC)

- Contains the offset in the current code segment for the next instruction to be executed
 - Advanced from one instruction boundary to the next in straightline code, or
 - Moved ahead or backwards by instructions such as JMP, Jcc, CALL, RET, and IRET
- Cannot be accessed directly by software
 - EIP is controlled implicitly by control transfer instructions, interrupts, and exceptions
- Because of instruction prefetching, an instruction address read from the bus does not match the value in the EIP register

Assembly Characteristics (1)

Minimal data types

- "Integer" data of 1, 2, 4, or 8 bytes
 - Data values
 - Addresses (untyped pointers)
- "Floating point" data of 4, 8, or 10 bytes
- No aggregate types such as arrays or structures
 - Just contiguously allocated bytes in memory
- (cf.) In IA-32, a "word" means 16-bit data



Assembly Characteristics (2)

Three primitive operations

- Perform an arithmetic/logical function on register or memory data
- Transfer data between memory and register
 - Load data from memory into register
 - Store register data into memory
- Transfer control
 - Unconditional jumps
 - Conditional branches
 - Procedure calls and returns

IA-32 Reference

Intel 64 and IA-32 Architectures Software Developer's Manual

- Volume 1: Basic Architecture
- Volume 2A, 2B: Instruction Set Reference
- Volume 3A, 3B: System Programming Guide