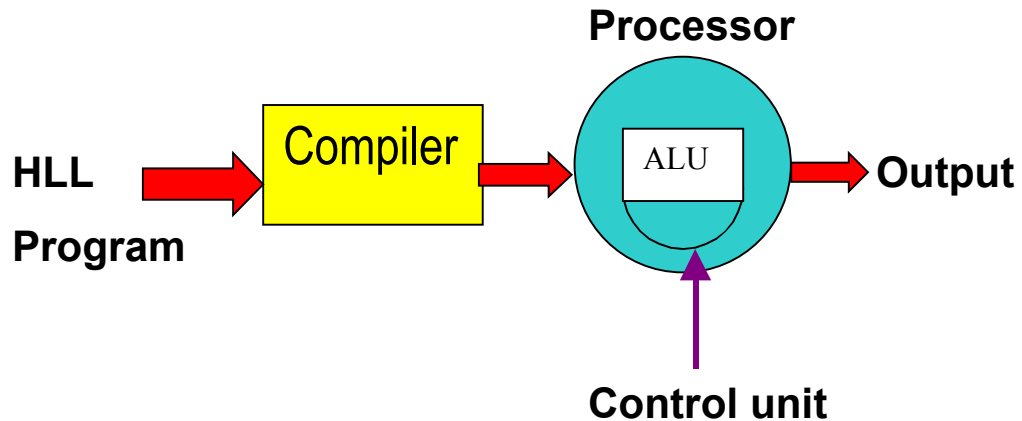


Two Design Choices



1. Either the control unit can be smart, i.e. it can delay instruction phases to avoid hazards. Processor cost increases.
2. The compiler can be smart, i.e. produce optimized codes either by inserting NOPs or by rearranging instructions. The cost of the compiler goes up.

Instruction Reorganization by Compiler

To avoid data hazards, the control unit can insert bubbles.

As an alternative, the compiler can use **NOP** instructions.

Example: Compute **a: = b + c; d:= e + f**

(a, b, c, d, e, f are stored in the memory)

LW R1, b	LW R1, b
LW R2, c	LW R2, c
ADD R3, R1, R2	NOP
SW a, R3	NOP
LW R1, e	ADD R3, R1, R2
LW R2, f	NOP
SUB R3, R1, R2	SW a, R3
SW d, R3	LW R1, e
	LW R2, f
	NOP
	NOP
	SUB R3, R1, R2
	NOP
	SW d, R3

Original code

Code generated by a smart compiler

Instruction Reorganization by Compiler

The compiler can further speedup by reorganizing the instruction stream and **minimizing the no of NOP's**.

Example: **Compute** **a:= b + c; d:= e + f**

LW R1,b	LW R1,b
LW R2,c	LW R2,c
ADD R3, R1, R2	LW R4, e
SW a, R3	LW R5, f
LW R1, e	ADD R3,R1,R2
LW R2,f	NOP
SUB R3, R1, R2	SW a, R3
SW d, R3	SUB R6, R5, R4
	NOP
	SW d, R6
	NOP

Original code

Code reorganized by a smart compiler

(Control unit remains unchanged)

*Note the **reassignment** of registers*

Intel IA-32 Architecture

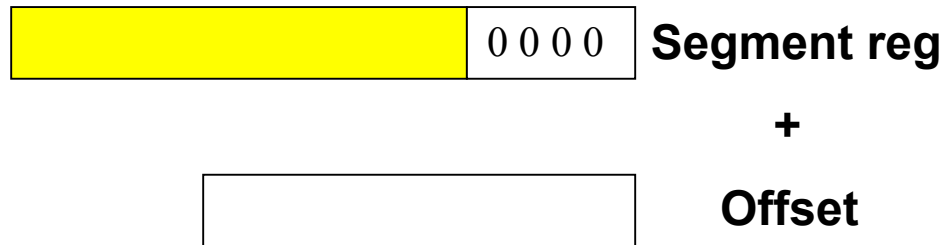
Study the 8086 architecture first.

15		0	
AH	AL	AX primary acc.	
BH	BL		BX arithmetic
CH	CL		
DH	DL		CX loop count
		DX multiply & divide	
		SI source index	
		DI destination Index	
		BP base pointer	
		SP stack pointer	

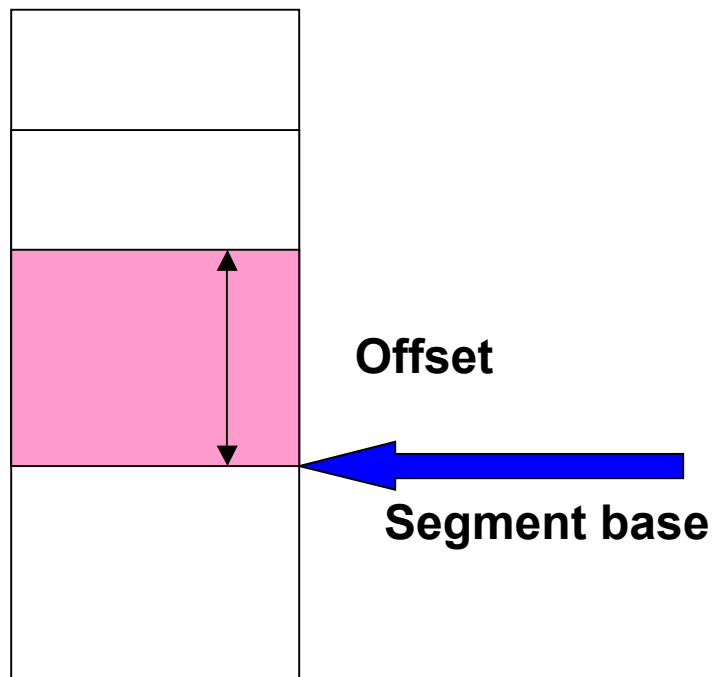
SEGMENT REGISTERS		CS code segment
		DS data segment
		SS stack segment
		ES extra segment

	IP instruction pointer
	PSW program status wd

Intel 8086 Addressing Style



Generates a **20-bit** memory address



Sample Instructions

Accumulator type **AX:= AX + BX**
 AX:= AX + M

String operations **Move a char string from
 one location to another
 (Uses SI and DI)**

Starting from 80386, x86 became a true 32-bit architecture. Its instruction set broadly defines the IA-32 architecture.

Architecture vs Organization

CPU architecture refers to the programmer's view, i.e. primarily the instruction set. **The same architecture can be implemented in many different ways. True for IA-32 too.**

Pentium Family

CISC architecture – executes **IA-32** instruction set

Year	Processor	Clock	L1 cache KB	L2 cache KB
1993	Pentium	60 MHz		
1995	Pentium Pro	100-200 MHz	8+8	256+1024
1998	Pentium II (MMX added)	233-450 MHz	16+16	256 + 512
1999	Pentium II Xeon	400-450 MHz	16+16	512 + 2048
1999	Celeron	500-900 MHz	16+16	128
1999	Pentium III (70 new MMX instructions)	450-1100 MHz	16+16	256+512
2000	Pentium III Xeon	700-900 MHz	16+16	1024+2048
2001	Pentium 4 (NetBurst)	1.3 – 2.1 GHz	8+8	256+512

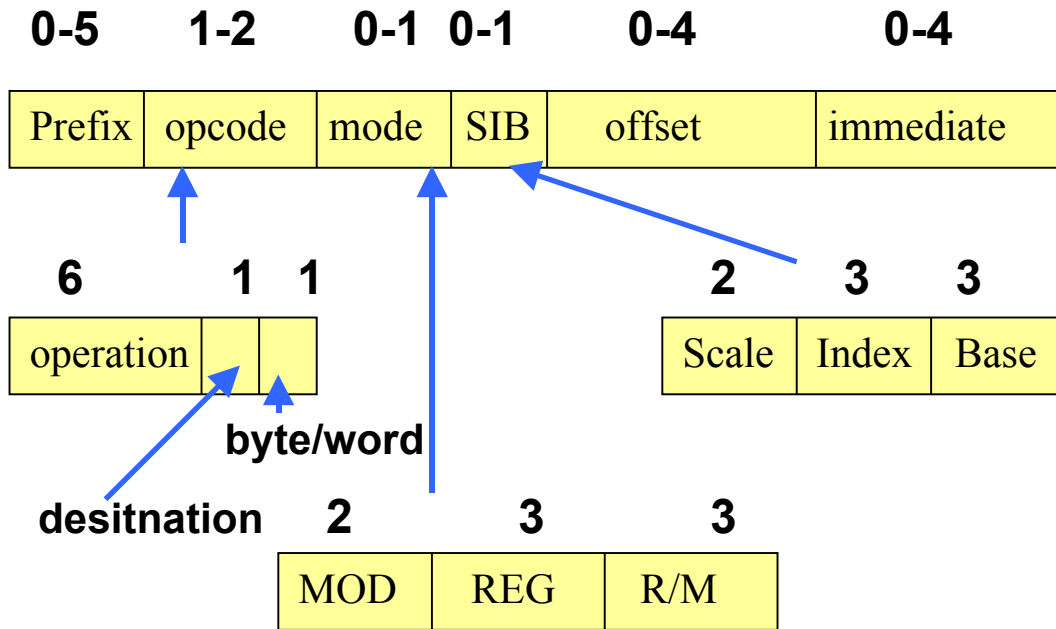
All Intel processors are backward compatible.

Intel Pentium Registers

	31	16	15	0	
EAX			AH	AL	AX primary acc.
EBX			BH	BL	BX arithmetic
ECX			CH	CL	CX loop count
EDX			DH	DL	DX mult & div
ESI					SI source index
EDI					DI dest. Index
EBP					BP base pointer
ESP					SP stack pointer
SEGMENT REGISTERS					CS code segment
					DS data segment
					SS stack segment
					ES extra segment
					FS extra segment
					GS extra segment
EIP					IP instruction pointer
EFLAG					PSW program status wd

Each **memory address** is the sum of the contents of a **segment register** and **another register**.

Pentium Instruction Formats



Tells us about the operand

Complex, irregular, and suffers from the legacy of some bad irreversible design decisions made in the past.



1st operand = **register**

2nd Operand = **register** or **memory** or **immediate**

Pentium instruction formats

Two operands: One operand is a Register (REG field), and the other specified by MOD and R/M.

R/M	MOD=00	MOD=01	MOD=10	MOD=11
000	M[EAX]	M[EAX+ Offset 8]	M[EAX+ Offset 32]	EAX or AL
001	M[ECX]	M[ECX+ Offset 8]	M[ECX+ Offset 32]	ECX or CL
010	M[EDX]	M[EDX+ Offset 8]	M[EDX+ Offset 32]	EDX or DL
011	M[EBX]	M[EBX+ Offset 8]	M[EBX+ Offset 32]	EBX or BL
100	SIB	SIB with Offset 8	SIB with Offset 32	ESP or AH
101	Direct	M[EBP+ Offset 8]	M[EBP+ Offset 32]	EBP or CH

Using SIB, operand address = Base register + index register x scale factor (1/2/4) + offset.

Lack of **orthogonality** is disturbing

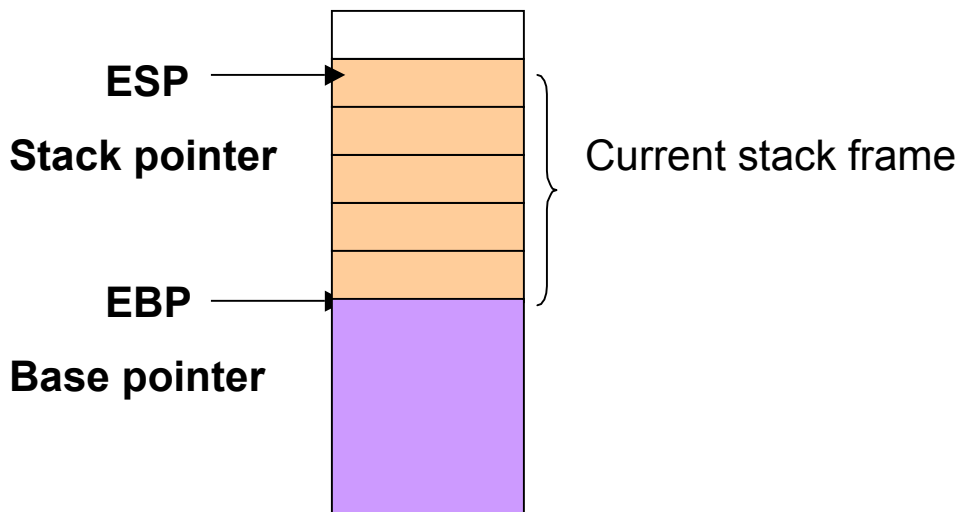
Observations about Pentium Instructions

Variable length instructions: 1 to 17 bytes. It makes the control unit ugly.

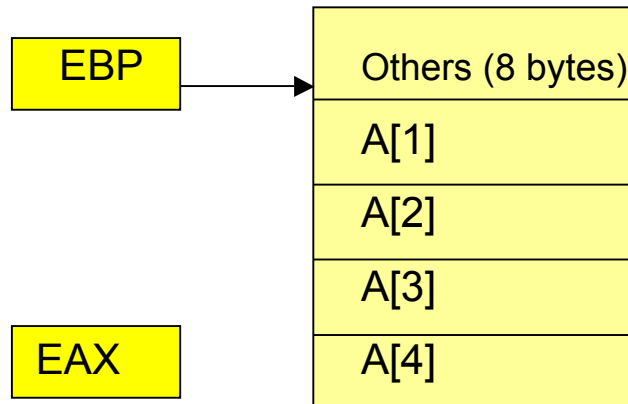
The burden of backward compatibility...

Real mode	Runs 8088 programs.
Virtual 8086 mode	Runs 8086 programs
Protected mode	Works like Pentium

Use of EBP and ESP



What is SIB?



Useful for addressing the elements of an array

Let each element be 32 bits, i.e 4 bytes.

Store the array index i in EAX.

$$\text{Address of } A[i] = \text{EBP} + 4 * \text{EAX}$$

MMX and SSE instructions

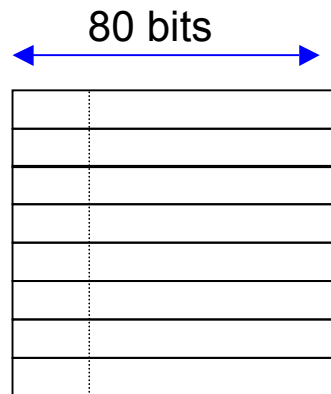
SIMD (Single Instruction Multiple Data) instructions are useful for multimedia application.

MMX = multimedia

SSE = Streaming SIMD instruction



Supports multimedia operations
Can store 8-bit colors for 8 pixels
in one MMX register and execute
SIMD instructions to accelerate
graphics applications



FP-cum-MMX

SSE instructions use a separate set of registers.

Prefix bytes

Attributes to instructions...

REP [instruction]

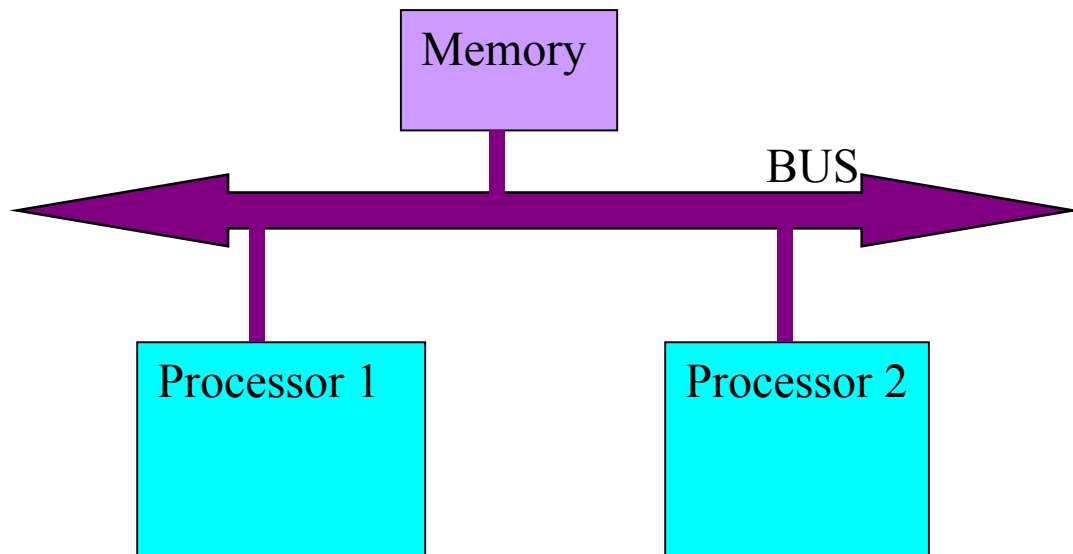
Repeat the instruction until ECX=0

REPZ [instruction]

Repeat the instruction until Z-flag is set.

LOCK [instruction]

Lock the memory bus until the instruction execution is complete.



A multiprocessor