

Overview of the 80x86 Family

1978	8086 (A ddressable M emory: 1 M)
1982	80286 (A M: 16M, V irtual M emory: 1G)
1985	80386 (A M: 4G, V M: 64Terabytes; 32-bits)
1989	80486 (as above; faster)
1993	Pentium (as above; more powerful)
2001	Itanium (64-bits)

Why Assembly Language

- Speed
 - assembly language programs much faster
 - (potentially) 5 to 10 times faster
- Space
 - assembly language programs often the smallest
 - space considerations not ignorable - even today

Why Assembly Language

- Capability
 - you can do things that are difficult or impossible in HLLs
 - e.g. I/O devices
- Knowledge
 - write better programs, even in HLLs
 - understand how things really happen

Binary Numbers

- 1 bit (0 or 1)
- 1 nibble: (four binary digits or bits)
 - min: 0000 (0h) max: 1111 (0Fh/15)
- 1 byte: (8 bits/2 nibbles)
 - min: 00000000 (0h) max: 11111111 (0FFh/255)
- 1 word: (16 bits)
 - min: 0000 0000 0000 0000 (0h)
 - max: 1111 1111 1111 1111 (0FFFFh/65,536)
 - register size

x86 Registers

- place in the CPU where a number is stored and manipulated
- 3 sizes
 - 8-bit, 16-bit, 32-bit
- 4 types
 - general purpose
 - segment
 - index
 - stack

General Purpose Registers

- 16-bit/32-bit registers, located on-chip
- all arithmetic and logical operations occur in regs
- (E)AX - accumulator register
- (E)BX - base register
- (E)CX - count register
- (E)DX - data register
- Others: IP, Flags
- faster than memory (on-chip)

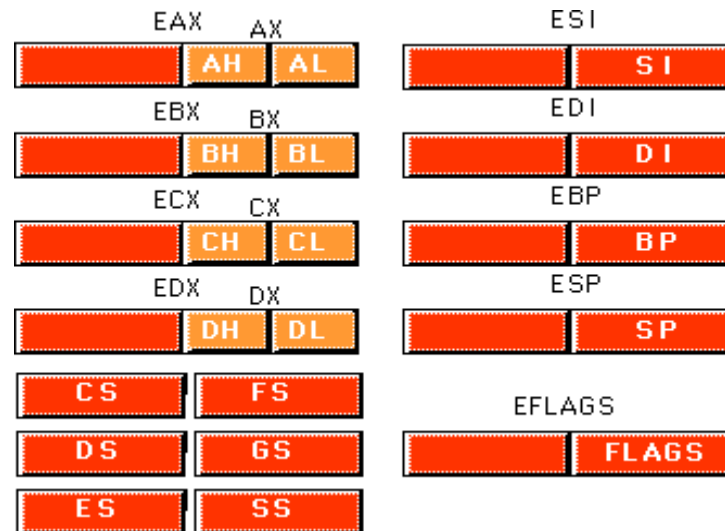
Index Register

- also called pointer registers
- mainly used for string instructions
- SI - source index
- DI - destination index
- IP - instruction pointer
- also: ESI and EDI
- BX can also be used to index strings
- IP can't be manipulated

Stack Registers

- BP - base pointer
- SP - stack pointer

Register Summary



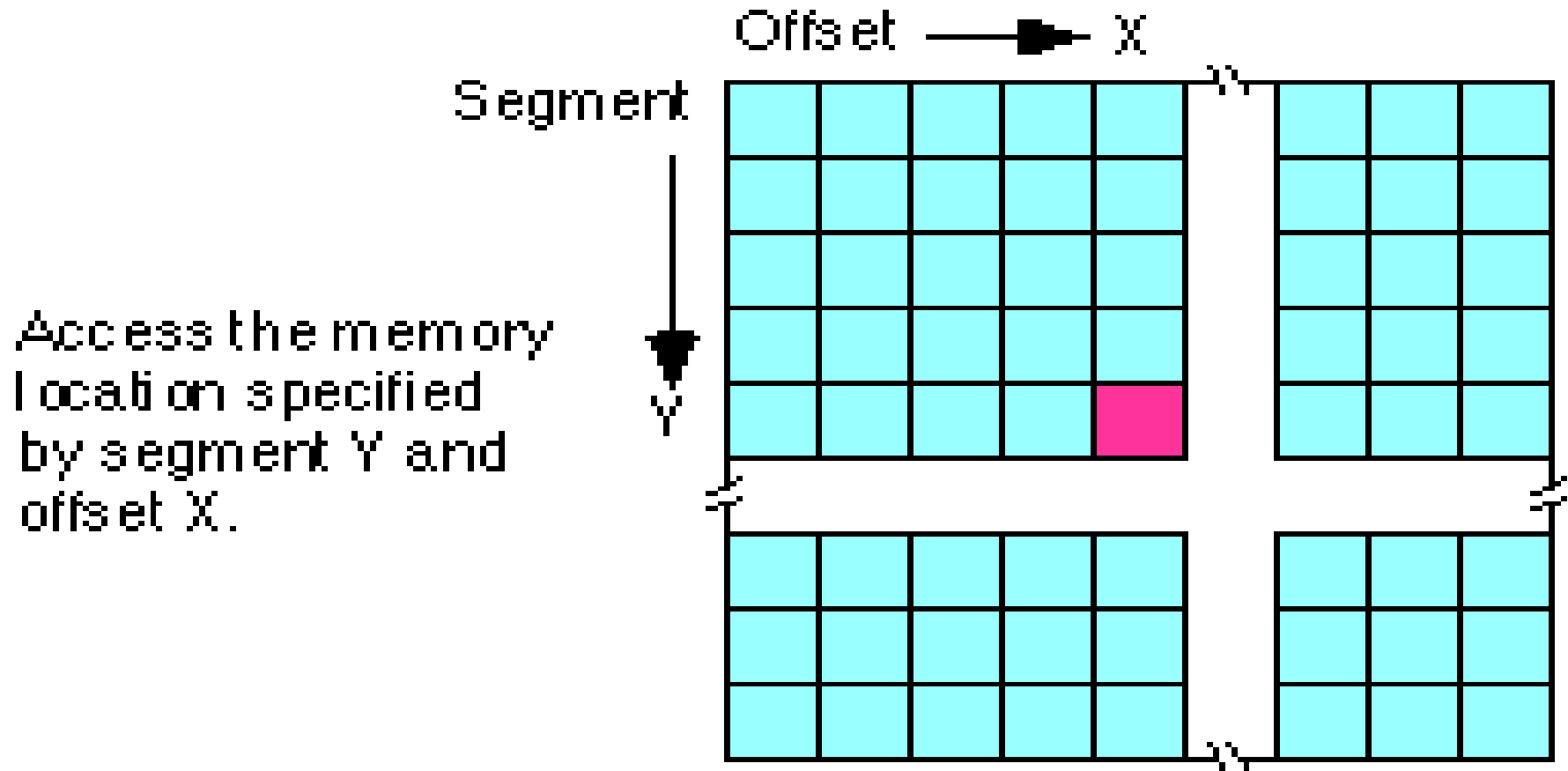
Segments and Offsets

- powerful memory management mechanism
- break programs into modules
- provide a way to easily implement OO programs
- two processes can share data

BUT ...

- many people think it's terrible
- Why?

Segments and Offsets



Segments and Offsets

- 8086 max addressable memory: 1M
 - couldn't use memory beyond 1M
 - only 16-bit regs => only address 64K
 - segmentation: 64K->1M (> 64K effort, > 256K lot of effort)
- 80286 max addressable memory: 16M
- Fixed with 80386 but ...
 - MS-Dos still uses 1976 segmentation!
 - Linux, Unix, Windows 9x 2000 NT XPALL OK

Segments and Offsets

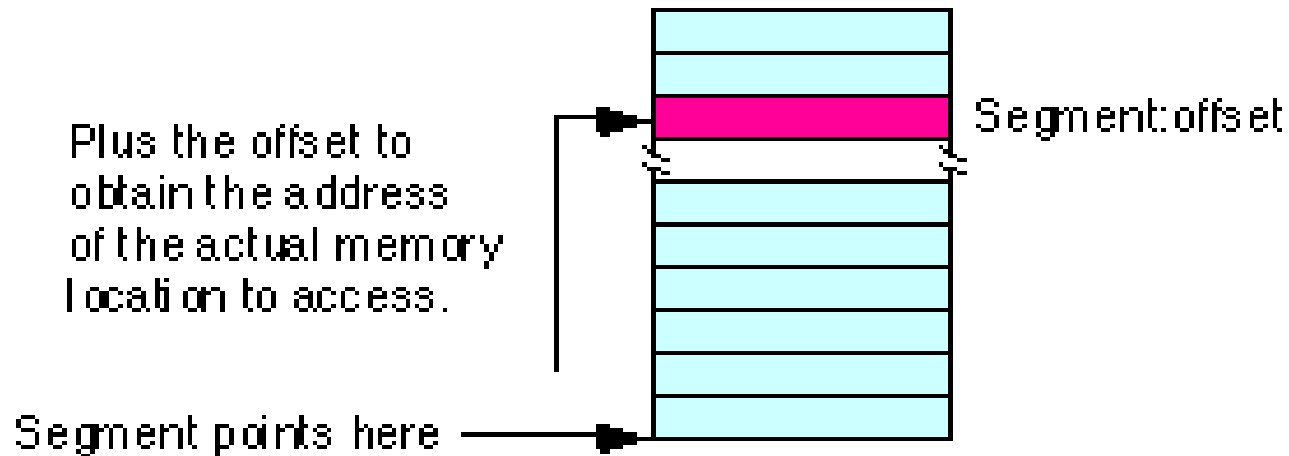
- Linear addressing
 - linear array of bytes
 - single address/index selects a particular byte
- Segment addressing
 - segments of bytes
 - segment value and offset selects a particular byte
 - segment value and offset should be independent of one another
 - offset size limits the max size of a segment

Segments and Offsets

- allows programs to have local variables
 - (also implemented via protected mode 80286)
- 8086/80286
 - segment and offset components 16-bit constants
 - max segment is 64K
- 80386/later
 - 16-bit or 32-bit components
 - 4GB segments, 32-bit offsets

Segments and Offsets

- segmented address = logical address
- actual linear address = physical address



Segments and Offsets

- 8086, 8088, 80186, 80188

Segment to Physical Address Mapping

Segment \longrightarrow **1000:1F00** \longleftarrow Offset

$$\begin{array}{r} \downarrow \\ \mathbf{10000} \\ + \mathbf{1F00} \\ \hline \mathbf{11F00} \end{array}$$

First, multiply the segment value by 10h.
Then add in the offset portion.

Their sum produces the physical address

Segments and Offsets

Segment registers

CS - Code

DS - Data

ES - Extended

SS - Stack

386+

- FS, GS

Offset registers

BX

DI

SI

BP

SP

IP

386+

any general reg can be used as
offset reg