VM Case Study: x86

- Topics
  - X86 (lots of helpful hardware)
- Learning Objectives:
  - Describe the x86 MMU support
  - Discuss the tradeoffs between hardware and software managed VM
  - Explain why the current x86 processors have so many different modes of operation (e.g., real, protected, etc).
  - Explain how to achieve flat addressing on a segmented architecture.
  - Discuss the tradeoffs in memory sharing flexibility between segmented and non-segmented architectures.

Intel x86 VM System

- The Intel x86 Virtual Memory Architecture is a reflection of many major revisions that have occurred over various generations of Intel microprocessors.
  - The changes between these revisions were substantial enough that learning x386 addressing is a lot like learning at least three different virtual memory systems!
  - We will study the three addressing models in the order in which they were introduced.
  - This also roughly matches the level of complexity of the early systems.

X86 Historical Summary

	Introduced	N Instr	Phys. Mem	VAS size
8008	1972	66	16 KB	None
8080	1974	111	64 KB	None
8086	1978	133	1 MB	None
8088	1981	133	1 MB	None
80286	1982	169	16 MB	16 MB
80386	1985	187	4 GB	64 TB
80486	1989	193	4 GB	64 TB
Pentium	1993	198	4 GB	64 TB
Pentium Pro	1995	234	64 GB	64 TB
Pentium 4	2004		1 TB	16 EB
Core i7	2008		16 TB	16 EB

The Intel 8080

- 8 bit arithmetic
- 16 bit addressing
- 3 registers
  - An 8-bit accumulator
  - Two 8-bit address registers, called H and L
- The address registers could be used together to form a 16-bit address.
- Memory was addressed in bytes, so the number of bytes that could be addressed was 2^{16} = 65536 (64K) bytes.

The Intel 8086: Real Mode

- The memory addressing model of the Intel 8086 is used by PC programs that run in real mode.
  - Memory is addressed by taking a 16-bit segment base, shifting left by four bits, and adding a 16-bit index.
  - This yields a 20-bit address, for up to 1 megabyte of physical memory.
  - The size of the index registers limits the segment size to 64K bytes.
- Windows systems provide an emulator for 8086 memory addressing inside a "DOS box."
- The Intel 8088, which was used in the first IBM PC, was the low cost version of the 8086, and had an identical architecture to the 8086.

8086 Example

16-bit arithmetic

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

15 7 0

data registers

16-bit index

SI	Src ndx
DI	Dst ndx
BP	Base ndx
SP	Stack ndx

15 0

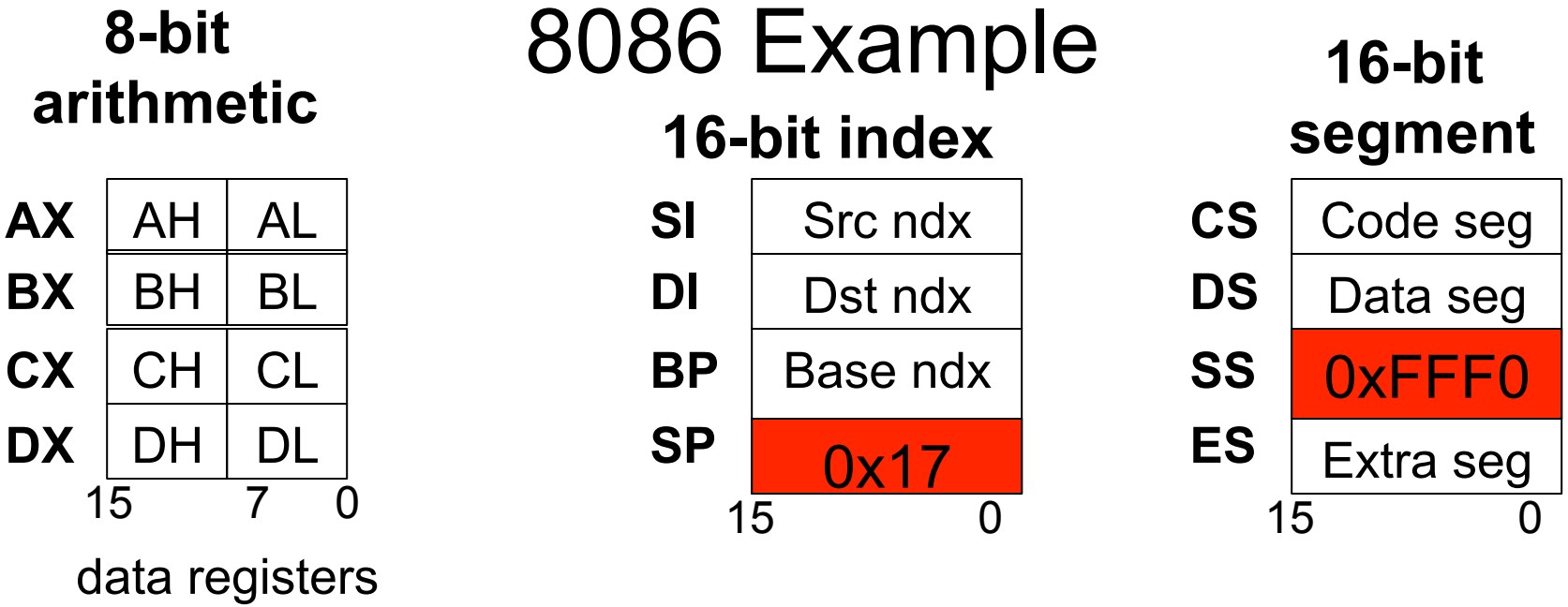
16-bit segment

CS	Code seg
DS	Data seg
SS	Stack seg
ES	Extra seg

15 0

- Let's say that Register SS contains 0xFFF0
- And that Register SP contains: 0x17
- What address is referenced by the instruction push AX?

8086 Example



- Let's say that Register SS contains 0xFFF0
- And that Register SP contains: 0x17
- What address is referenced by the instruction push XA?
  - Shift segment base left by four: 0xFFF0 -> 0xFFF00
  - Add index to segment address: 0xFFF00 + 0x17 = 0xFFF17

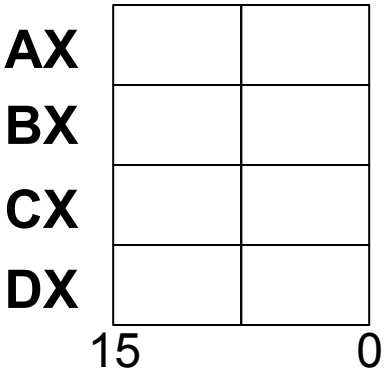
20286 Segmented Virtual Memory

- Introduced virtual memory in the x86 family.
- Added a level of "indirection" or address mapping between the segment number and the address of the segment.
- This was achieved through a number of architectural changes: termed protected mode.
  - **The interpretation of the contents of the segment registers changed.**
  - The 20286 supports two segment descriptor tables.
  - Segment registers contain a segment selector.
    - The TI bit (bit 2) selects one of two segment descriptor tables.
    - The segment descriptor index indexes into that table, which contains a segment descriptor
    - The segment descriptor holds all the information needed to access a segment, including base address, size, and other attributes.
    - The RPL field (bits 0 and 1) is used to implement the 80286 protection model

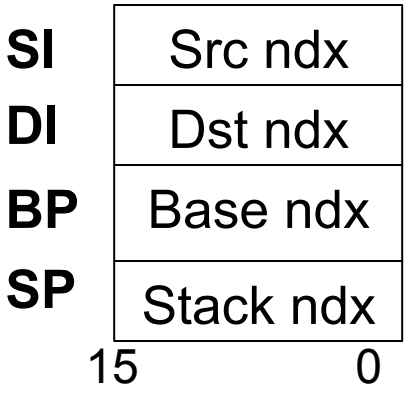
Segment descriptor index 15 .. 3	TI 2	RPL 1-0

80286 Registers

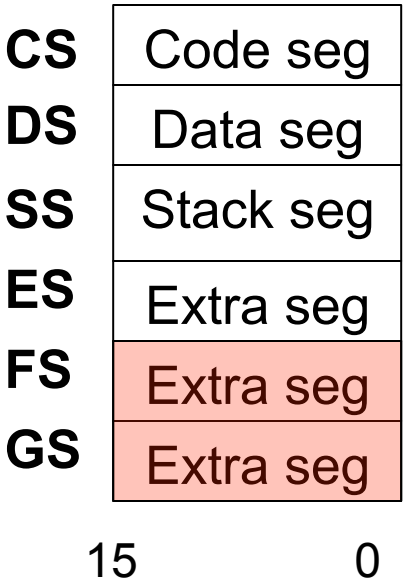
16-bit arithmetic



16-bit index



16-bit segment



80286 Example

- Register ES contains 0x0018
- What address is referenced by the instruction:

```
mov word ptr es:[0x6e70], esp
```

ES | 0000000000011000

Base	Size	Attributes
0	1000	No access
100000	3000	Rx, present, …
400000	8000	Rw, present …
480000	8000	Gate, AVL
…		

Segment descriptor table 0

80286 Example

- Register ES contains 0x0018
- What address is referenced by the instruction:

```
mov word ptr es:[0x6e70], esp
```

ES 0000000000011000

Index TIRPL

Base	Size	Attributes
0	1000	No access
100000	3000	Rx, present, …
400000	8000	Rw, present …
480000	8000	Gate, AVL
…		

Segment descriptor table 0

486E70

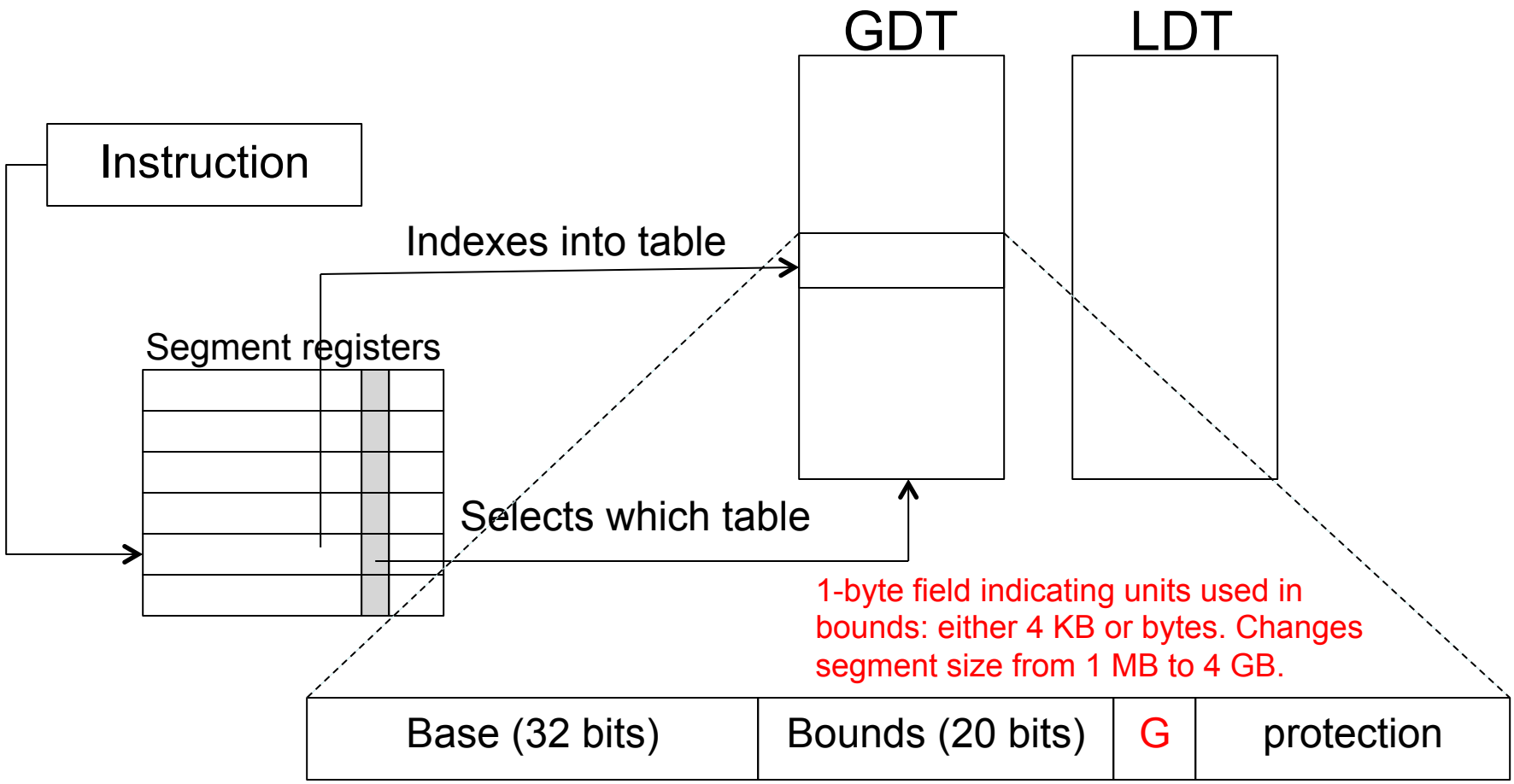
80386: Paged Virtual Memory

- Introduced paged virtual memory to the x86 family.
- Extends the 80286 architecture.
  - Two additional segment registers.
  - Segment register selected depending on the instruction, the addressing mode, and an optional prefix which explicitly selects a register.
  - Segment descriptors are similar to the80 286.
  - There are two types of descriptor tables:
    - Global Descriptor Table (GDT): normally used for operating system segments.
    - Local Descriptor Table (LDT): normally used for user mode segments. Typically one per process/task (although Windows uses a single LDT for all applications).
  - As in the 80286, the TI bit selects the table.

80386 Translation

- Instruction selects a segment register.
- Segment register selects a Descriptor table.
- Segment register indexes into Descriptor table.
- Descriptor provides a base address.
- Offset is added to base address.
- Constrained by limit in descriptor (properly modified by the G-bit to indicate whether max segment size is 1 MB or 4 GB).

80386 Data Structures

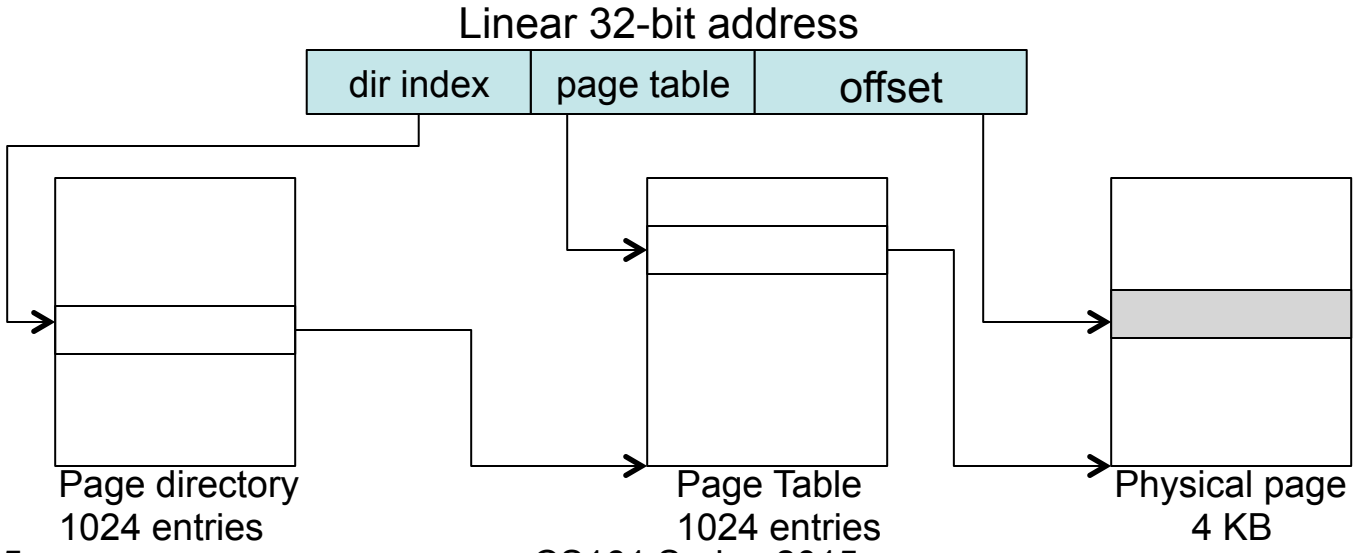


Linear Addressing on the 80386

- By setting all segment registers to reference the same 4GB segment, memory can be referenced using a "flat" 4GB address space (as opposed to a segmented one).
- (It is not coincidence that this is much like the R2000.)
- So-called flat addressing is a convention, not a processor mode.

Paging on the 80386

- The address that comes out of the previously described translation can then be fed into a paging unit.
- As with the MIPS R2000, page translation occurs (in concept) by using the virtual page number as an index into the Page Table, yielding a physical page number
- A 32-bit "linear" (e.g., non-segmented) address is made up of a 20-bit page number and a 12 bit offset.
- The page number is further divided into a 10-bit directory index and a 10-bit page number.



Parting Notes on the x86

- In the 80386, all the segments map into the same linear address space and the linear address space is paged.
- The IBM 370 also combines segmentation and paging, except that it uses a separate page table for each segment.
- How can sharing be implemented?
  - On a segment basis: two processes can share the same segment descriptor.
  - On a page basis: two segments can have page tables entries, which map the same physical page into each segment.
  - How does location of permission bits affect this?
- What about fragmentation?
  - Pages eliminate external fragmentation (allocation is done in page sized units).
  - Internal fragmentation is the same for paging (you always allocate full pages, so you waste any part of a page that is unused).
  - If page size is small, you can reduce internal fragmentation.
- Later x86 architectures
  - Look like the x86 with increased virtual and physical memory sizes.