The Pentium Architecture

October 9

CSC201 Section 002

Fall, 2000

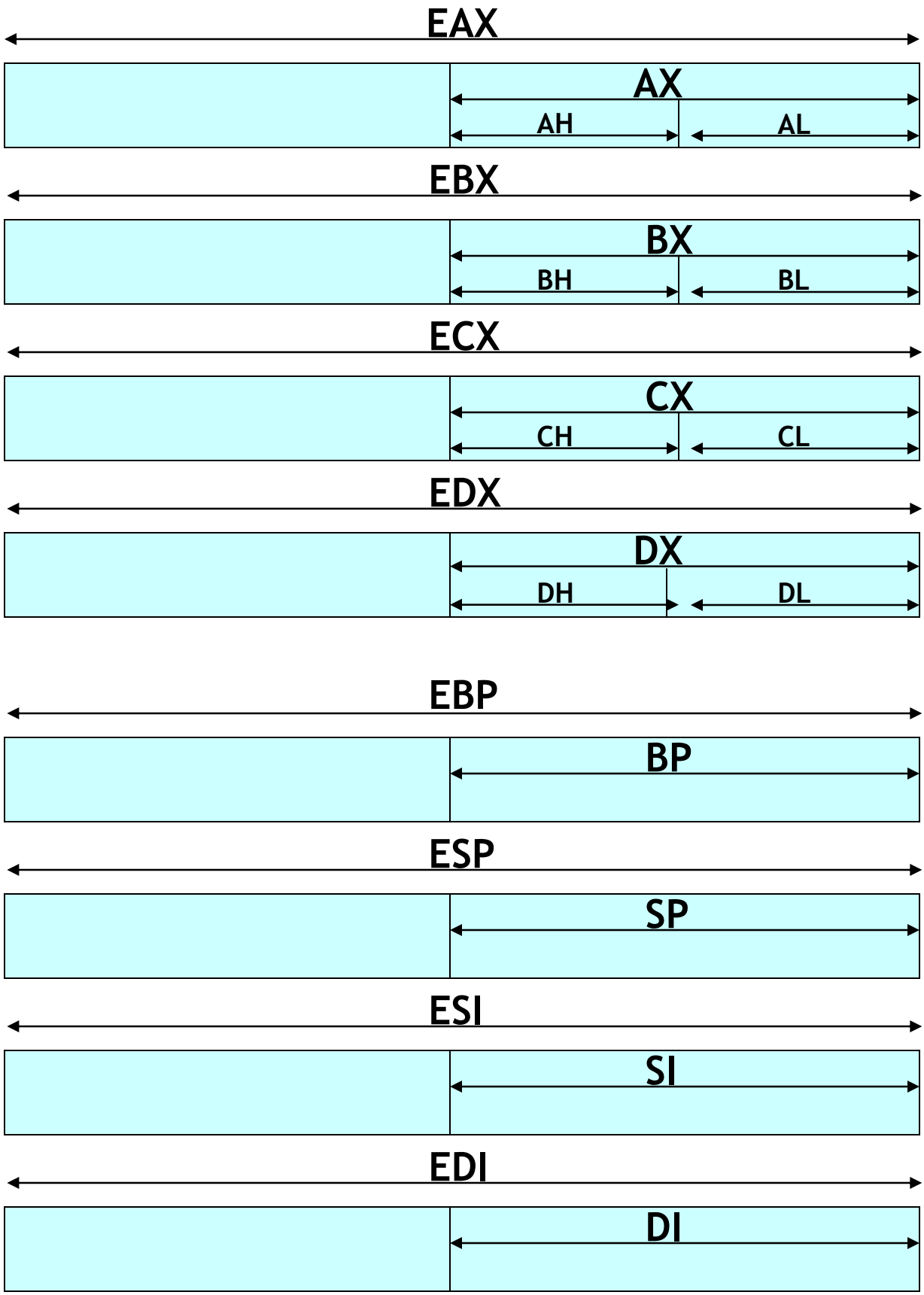
Basics

- Almost 350 different instructions!
- Backwards compatibility with the first 8086 processor
- Two-address instruction format
- Operands can be registers or memory
  - Not a load-store architecture

Basics (cont.)

- Register sizes: 8-bit, 16-bit, and 32-bit
- Very flexible addressing modes
  - Permitted instruction<-->address mode combinations irregular (lots of special cases, hard to learn!)
- Condition codes (such as ZF and SF)
  - Kept in a special register (EFLAGS)
  - Effect by each instruction is somewhat complex, irregular

Registers



Registers

- 4 main 32-bit general purpose registers
  - EAX, EBX, ECX, EDX
- Goal for register use: minimize the need to read from /write to memory
  - Would be a lot easier with more GPRs!
- Each GPR can be split, can refer to the least significant 16 bits
  - AX, BX, CX, DX
- Each 16-bit register can be split, can refer to upper and lower half
  - AH, AL, BH, BL, CH, CL, DH, CL
- 4 other 32-bit general purpose registers
  - EBP, ESP, ESI, EDI

Registers (Cont.)

- Each of these can be split, can refer to least significant 16 bits
  - BP, SP, SI, DI
- Specific instructions may limit which registers can be used for which purpose
  - In some cases, the register is implicit
  - Makes the instruction encoding and hardware implementation more efficient
  - Makes programming more difficult!
- EFLAGS is a 32-bit special purpose register
- (Segment registers not used: CS, DS, ES, FS, GS, SS)

Pentium Memory Model

- "Flat" model: every address is 32-bits
  - Any of 4 billion bytes of memory can be specified
- "Segmented" model: each address is 16 bits
  - Used by early x86 processors to reduce instruction size
  - 2^{16} = 64KB of memory; way too small!
- Add to a segment register (base address) to specify a 64KB region in memory
  - Can access much larger overall memory
  - Only 64KB for a specific segment register value

Segmented Memory Address Computation

16-bit offset (in instruction)

\+ 20-bit segment register

= 20-bit effective address

Addressing Mode Examples

- Immediate:

```
mov     eax, 0000ffa3h
```

- Register:

```
mov     eax, ecx
```

- Memory Direct:

```
mov     eax, var1
```

- Register Indirect:

```
lea ebx, var1
mov     eax, [ebx]
```

- (Alternatively:)

```
lea ebx, var1
mov     eax, dword ptr [ebx]
```

- ("Byte ptr" also possible)

Examples (Cont.)

- Base Displacement:

```
lea     ebx, myarray
mov     eax, 16[ebx]
```

- (Alternatively:)

```
lea     ebx, myarray
mov     eax, [ebx+16]
```

- Based-Indexed:

```
lea     ebx, myarray
mov     ecx, 16
mov     eax, [ecx][ebx]
```

- Memory Indirect:
  - (none)
- PC Relative:

```
jmp     label_not_too_far_away
```