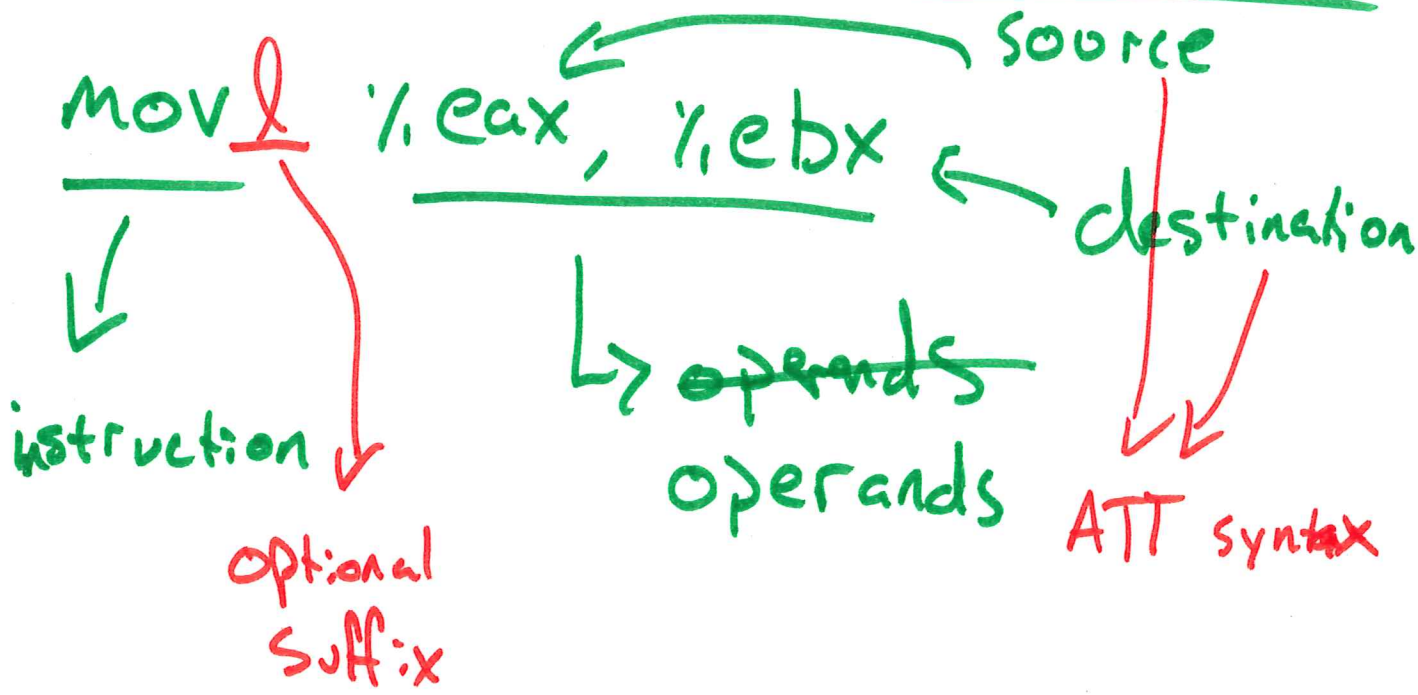
X86 history

- Lots of other ISAs
- RISC vs CISC
- 8086 -> one of the first single chip microprocessors
  - -> 80286 -> 80386 (i386)
    - -> i486 .... x86
      - Pentium 3 ('98) (P6)
        - -> Pentium 4
          - ⋮
          - Core 2 arch.

- Subset of x86 instructions
  - Data movemnt
  - Arithmetic + logical
  - Control flow



- only covering ATRT syntax (GAS)
  - Intel syntax -> ~~drop~~ drops suffixes
    -> switches operand ordering

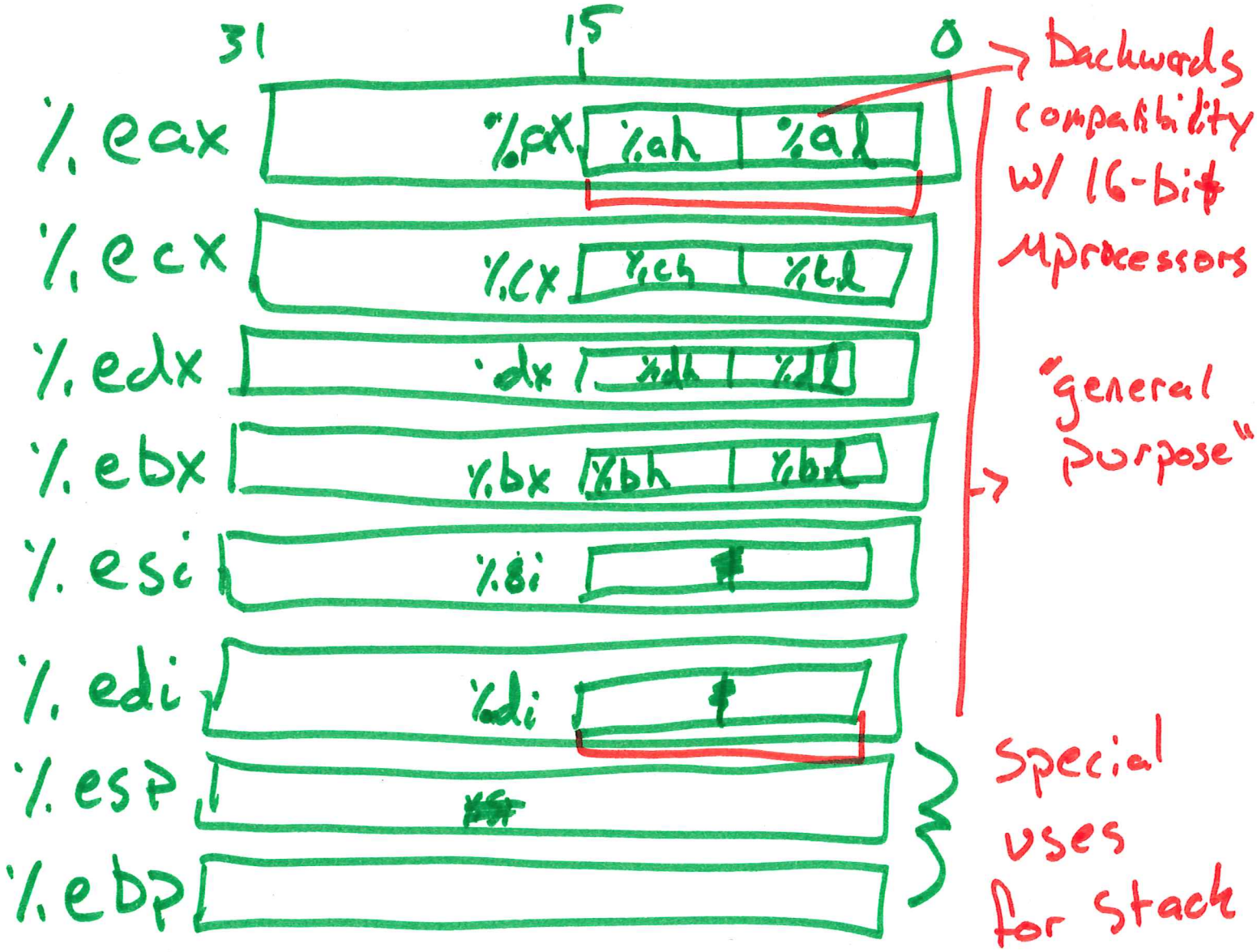
- Most common operand is registers

Registers: place hold temp. data

~~other~~ most hardware/inst. directly interacts w/ registers

---

x86 -> 8 registers -> i386



%xyz

- general purpose registers
  - can be used for anything
  - except when they can't

- x86 → "word" 16-bits (2-bytes)

(double) → "long" 32-bits (4-bytes)

→ "quad" 64-bits (8-bytes)

operand specifiers

instr op1, op2

↳ what's allowed here

- Registers → %eax (value in the register)
- Immediates → ~~0x1000~~

mov $12, %eax

$12 (this value)
$0xFF $-15

Addressing memory (value in addr 0x1000)

- Absolute addressing: 0x1000

  mov 0x1000, %eax

- register direct: (%eax) → value in address held in %eax
  (%esi)
  (%esp)

base-displacement: 8(%eax)

Imm(%R)

↳ value in address held in %eax+8

-16(%ecx)

0(%esp)

Scaled index: most general memory addressing

Imm(%R1,%R2,Imm) ← size

↓ address

%R1 + %R2·size + Imm offset

0(%eax, %ebx)I

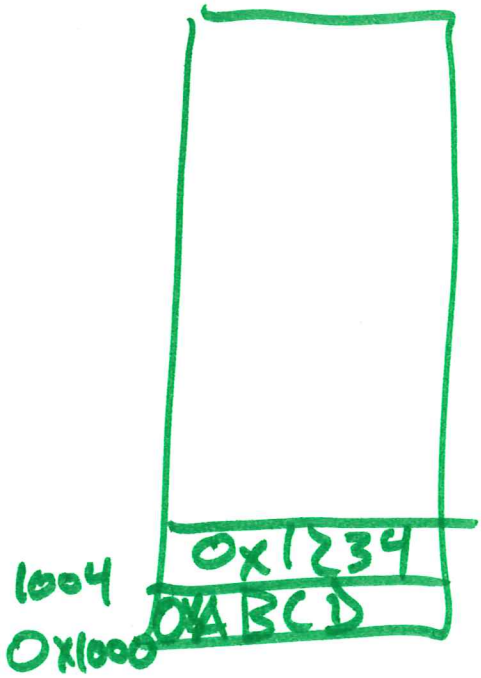
↳ value in address %eax + (%ebx·1)

0(%ebx, %edx, 4)

↳ value in address %ebx + %edx·4

8(%esp, %edx, 2)

↳ value in addr. %esp + %edx·2 + 8



mem (Abs. Addressing)

0x1000 ⟼ 0xABCD

$0x1000 ⟼ 0x1000

↳ Imm.