CS271 WK2
— readings —

4.1, 4.2, 4.5

- Opcodes can have 0, 1, 2, 3 Operands
- Operands can be:
  - → Immediate: literal
  - → register: from register
  - → Memory: references a memory location.
- When doing "MOV AL, var1," var 1 will be moved as an address, whereas "MOV AL, [var1]" will dereference var1.
- Restrictions of the MOV opcode
  - → source, dest must be same size
  - → No mem address in both src and dest
  - → Instruction ptr (RIP, EIP, IP) cannot be the destination
  - → Moving things to partial registers (for eax) aL, al, ax, messes with the value of eax, visa versa
- Moving 16 bit numbers to 32 bit registers:

```
count is a 16-bit number, needs to
go into ~~ecx~~ ecx ~~bof~~ register:

MOV ecx, 0 ; ecx: 00000000h
MOV cx, count ;
```

↳ USE MOVZX (mov zero extended)

- MOVSX: MOV sign extended
  - → moves a smaller register/mem to a larger register, filling the rest with the "highest" (location) bit:
  - → 10110010 mov'd into 16-bit reg would be 1111111110110010
- Working with flags SAHF, LAHF
  - → SAHF saves EFLAGS to AH. This includes Sign, Zero, Aux. Carry, Parity, Carry.
  - → LAHF loads AH into EFLAGS
- XCHG: a cheap exchange opcode
  - → swaps the two operands values
- Direct Offset Operands
  - → Define an array: name TYPE 1,2,3...
  - → MOV will return first byte...
  - → MOV, reg, [name+1] returns second
  - → [name+1] produces an "effective address"

NOTE: adding 1 works fore byte-long variables only - adapt for different types

- ADDING, SUBTRACTING
  - → INC, DEC increment and decrement
  - → ADD, SUB add and subtract
  - → NEG converts a number to negative using 2's complement
- Parity Flag
  - → set when !odd # of 1 bits in a result.

- Sign Flag
→ represents the High bit (sign-determining)
→ 1 if negative
- Loop instruction
→ AKA "loop according to ECX"
→ Decrements ECx
→ "Quits" loop behavior when ECX = 0.
→ nested loops need to save ECx to variable before inside loop starts, restoring it when inside loop is done.

6.3