Assembly

- equal-sign directive
  COUNT = 10
  COUNT is replaced, works as a constant, compiler substitutes that value at assemble time
- $ - current value of location counter
  wVal word 10, 20, 18, -7, 55 — 00AC
  wValSize = ($ - wVal) / # of bytes (2) — 00B6

  mov ecx, wValSize
  (use ecx for looping)
- equ directive - equates
  xVal equ [ebp + 4]
  equ is used more than = or $
- movsx - copy & extend sign — mov eax, wVal — FFFFFFFF
  movzx - copy & extend zero — mov eax, wVal — 00 00 FF FF

  wVal word -1

- cbw - convert byte to word — AL → AX
  cwd - convert word to double word — AX → DX:AX pair
  cwde - convert word to double word extended — AX → EAX
  cdq - convert double word to quad word — EAX → EDX:EAX pair

  works only for ax/eax registers
- EFLAGS - 32 bits
  status - contained in lower 16 bit

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Assembly

0 - carry flag
2 - parity flag
6 - zero flag
7 - sign flag
11 - overflow flag

carry - set when there is a carry out on an operation
parity - set to keep number of 1 bits odd for odd parity machine
sign - set to 1 when result is negative
zero - set to 1 when result is 0
overflow - result to 2 positives becoming negative

```
add ax, wVal
jo  - jump overflow
jc  - jump carry
```

- inc [operand] → value is increased by 1 (doesn't touch flags)
  inc [ebx] → invalid! must know size
  inc word ptr [ebx] - valid

- neg [operand] → replaces operand with 2's compliment (w flags set)
  neg ax

- add bVal, bVal bVal = 127 bVal = 1
  CF=0, OF=1

  add -128 + -128
  CF=1, OF=1
  handle overflow by jo

Assembly

- EFL represents the EFLAGS
  EFL will turn red when it is changed
- ESP - stack pointer

- neg does not just flip bits, it performs a 2's compliment