9.1.11

Assembly

CSCI 2160

* Bytes
- byte = 8 bits 2^8
- word = 2 bytes 2^{16}
- double word = 2 words = 4 bytes 2^{32}
- quad word = 4 words = 8 bytes 2^{64}

* Signed integers
- MSB is the sign bit
- use two's compliment - first represent positive value, flip the bits & add 1
- -37 = 0DBh
- -97 → 01100001 → 10011110 + 1 = 10011111 = 9Fh
- -1 → 00000001 → 11111110 +1 = 1111 1111 = 0FFh
- in Assembly, you must write a 0 before a character in hex
- D7h as unsigned = 215
- D7h as character = Ö
- D7h as signed = 28h is flipped = 28h+1h = 29h = -41
- 83h as unsigned = 128 + 3 = 131
- 83h as signed = 7Ch + 1 = 70h = -125
- 73h as unsigned = 112 + 3 = 115
- 73h as signed = 115
- 80h as unsigned = 128
- 80h as signed = -128
- you can always represent one more negative value than positive in a signed binary number
- 8000h as signed = -32768
- 8000h as unsigned = 32768
- -125 as byte = 01111101 + 1 = 01111110 → 11111110 → 0FDh ✗ 83

-125 as byte = 01111101 → 10000010 + 1 = 10000011 → 83h ✓
bits flip bits add 1

* Symbols
- byte - db
- word - dw
- double word - dd
- quad word - dq
- answer byte
- byte answer; Java
- int value = 35; Java
- value dd 35 Assembly
or value dword 35 Assembly
- Java → Assembly
- short tax = 13; → tax dw 13h
- long value = 19; → value qw 19h
- int number = -20 → number dd -20
- byte answer = 'B'; → answer db 'B'
(write lined up)

```
.data
tax     word   13
value   qword  19
number  dword  -20
answer  byte   'B'      (you can use ' or ")
```

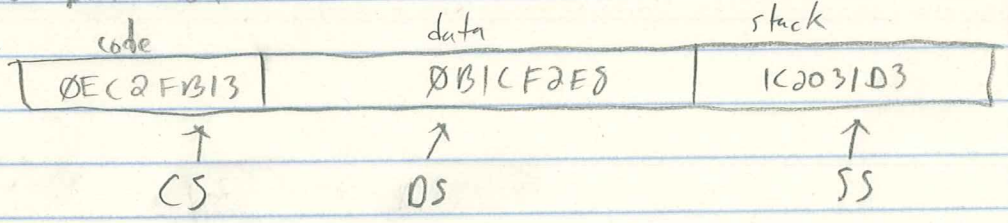
- store values in memory in reverse byte order
- Assembly assumes decimal if there is no trailing letter; use d to specify decimal, use h to specify hex
- 13d → (hex) 000Dh → (reverse byte) 0D00h
- -20d as dword → (hex) FFFFFFECh → (flip bits +1) 00000014h → (reverse byte) 14000000h
'B' as byte → (hex) 42h → (reverse) 42h

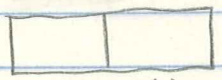
Statement	hex	reverse byte
- A word 22 →	0016h →	1600h
- B word 19 →	0013h →	1300h
- C dword 37 →	00000025h →	25000000h

- data values are stored sequentially
- strings are not stored in reverse byte order

* Registers

- RAX, RBX, RCX, RDX (workhorse registers) (32-bits)
- CS, DS, SS (32 bit)
- flat segment model - can use all 32 bits to represent an address
- 32 bits can store 4 GB of addresses
- 2^{40} - terabyte
- 2^{50} - petabyte
- 2^{60} - exabyte
- 2^{70} - zettabyte
- 2^{80} - yottabyte
- lower RAX is EAX, lower RBX is EBX, so on
- descriptor table





AX : 16 bits
AL+AH : 8 bits
EAX : 32 bits