1. Advantages
   1. Low level access to machine
   2. Total control over the CPU
   3. Faster code
2. Disadvantages
   1. Lack of library routines
   2. Increased risk of subtle bugs
   3. Non-portable code
3. ax = Accumulator
4. bx = base register
   1. cx = count register
   2. dx = data register
   3. ip = instruction pointer
      1. (holds address of next instruction)
   4. sp = stack pointer
      1. (points to top of the stack)
   5. bp = base pointer
5. C = A + B
   1. mov ax,[A] ;accumulate result in the accumulator
   2. add ax,[A]
   3. mov [c],ax
6. A = -(A+1)
   1. Inc word[A]
   2. Neg word[A]
   3. ***OR***
   4. Mov ax,[A]
   5. Inc ax
   6. Neg ax
   7. Mov [A], ax
7. A = A – B
   1. Mov ax,[A]
   2. Sub ax,[B]
   3. Mov [C],ax
   4. ***OR***
   5. Mov ax,[B]
   6. Sub[A],ax
8. C = A+B+B-42
   1. Mov ax,[A]
   2. Add ax,[B]
   3. Add ax,[B]
   4. Sub ax,42
   5. Mov [C],ax
9. 2bit arethemenit
   1. 01010101+10101011 = 000000
      1. cf = 1
      2. of = 0
      3. sf = 0
      4. zf = 1
   2. 01111111 + 01111111 = 11111110
      1. cf = 0
      2. of = 1
      3. sf = 1
      4. zf = 0
      5. 127+127 = 254
      6. 127+127 = -2?!
   3. 11111111+11111111=11111110
      1. cf = 1
      2. of = 0 (top bit as 1 and overflow was a 1, no overflow)
      3. sf = 1
      4. zf = 0
      5. unsigned
         1. 255+255 = 254
      6. signed
         1. -1+-1=-2
   4. 10101010+10110001 = 01011011
      1. cf = 1
      2. of = 1
      3. sf = 0
      4. zf = 0
10. Hex to binary!
    1. Each hex is 4 bits
    2. C 4 2 B base 16 to binary
       1. C = 1100
       2. 4 = 0100
       3. 2 = 0010
       4. B = 1011
11. 31 base 9 to base 3
    1. 3 = 10
    2. 1 = 01
    3. 1001 base 3 (I don’t fully understand why)
12. 1111111111111101 2 to base 10.
    1. Add 0000000000000010 + 1
       1. = **3** total
13. just do the math know what the binary is.
14. KNOW Truth tables
15. -2 to 8-bit
    1. -2 = 00000010
    2. sign/mag = 10000010
    3. 1’s comp = 11111101
    4. 2’s comp form = 11111101 + 1 = 11111110
16. Express **1000000100101010010011112**
    1. Octal is just in groups of 3
       1. 0 2 2 5 1 1 7
          1. Add leading zeroes to make sure it’s the right size
17. Stack grows downward in assembly
    1. Sp = 00Fo
    2. top of stack = 7300
    3. ip = 0ABC
18. pop off value and put it on ip
    1. then ip = 2046
    2. sp is then 01fc
19. al = 11010010b ,cf = 0
    1. shl al,1
       1. = 10100100 cf = 1
    2. shr al,1
       1. = 01101001 cf = 0
    3. sar al,cl (cl=2)
       1. (Sign doesn’t change) 11110100 cf = 1
    4. rcl al,cl (cl=3) (rotate carry left)
       1. 10010011
20. logic instructions
    1. clear bits using AND
       1. and al,0111 1110b (7Ehex)
    2. or bx,0101 0101 0101 0101 (0 leaves them alone)
    3. xor dl, 0000 0001b (01h) (xor will flip it the bit)
    4. NOT byte[BYTE1]
21. Shl ax,3
22. Sar bx,2 (signed is sar)
23. 1. **While23**: (because it’s question 23)
    2. Cmp ax,1
    3. Jng endwhile23
    4. Sar ax,1
    5. Jmp while23
    6. **Endwhile23**:
24. 1. toUpper:
    2. cmp al,’a’
    3. jnae out (jump not above or equal)
    4. cmp al,’z’
    5. jnbe out (jump not below or equal to out)
    6. and al,0DFh (1101 1111b) ;forced upper
    7. out:
    8. ret
25. (cx is what loop uses)
    1. mov cx,100
    2. **top25**:
    3. inc ax
    4. loop top25
26. 3. h = i-j\*k/m
       1. multiplication and division in order from left to right
       2. mov ax,[j]
       3. imul word[k]
       4. idiv word[m] (presume no overflow)
       5. sub ax,[i]
       6. neg ax
       7. mov [h],ax