x86 Assembly Language: Live Coding Exercises

CMSC 313
Sections 01, 02

Challenge 1

```
eax = ebx + ecx
```

2

Challenge 1-Answer

```
mov     eax, ebx
add     eax, ecx
```

3

Challenge 2

```
// foo, fum and fee are in memory
// also allocate space for them
// in the data section

int foo = 0, fum = 0, fee = 0;

fee = foo + fum;
```

4

Challenge 2-Answer

```
SECTION .data
foo: dd 0
fum: dd 0
fee: dd 0

SECTION .text
mov     eax, [foo]
add     eax, [fum]
mov     [fee], eax
```

5

Challenge 3

```
// array "buf" is in memory
// allocate space for it in the
// .bss section

char buf[256];

edi = &buf[0];  // address of buf
*edi++ = 'a';
*edi = '-';
*++edi = 'z';
```

6

Challenge 3-Answer

```
SECTION .bss
buf: resb 256

SECTION .text
mov     edi, buf          ; edi=&buf[0]
mov     byte [edi], 'a'   ; *edi = 'a'
inc     edi               ; edi++
mov     byte [edi], '-'
inc     edi
mov     byte [edi], 'z'
```

7

Challenge 4

```
if (eax < 0)
{
    eax++;
}
```

8

Challenge 4-Answer

```
        cmp     eax, 0
        jge     done
        inc     eax
done:
```

9

Challenge 5

```
if (eax > 0) {  /* eax is signed */
    eax++;
} else {
    eax = 0;
}
```

10

Challenge 5-Answer

```
        cmp     eax, 0
        jle     else
        inc     eax     ; "then" block
        jmp     done
else:
        mov     eax, 0 ;or "xor eax,eax"
done:
```

11

Challenge 6

```
ebx = 0;
while (eax > 0) {  /* eax is signed */
    ebx += eax;
    eax--;
}
```

12

Challenge 6-Answer

```
        mov     ebx, 0
L1_top:
        cmp     eax, 0
L1_X:   jle     done ; (ZF=1 or SF<>OF)
        add     ebx, eax
        dec     eax
        jmp     L1_top ; or L1_X?
done:
```

13

Challenge 6b

```
ebx = 0;
do {
    ebx += eax;
    eax--;
} while (eax > 0); /* eax is signed */
```

14

Challenge 6b-Answer

```
        mov     ebx, 0
L1_top:
        add     ebx, eax
        dec     eax
        cmp     eax, 0  ; optional!
        jg      L1_top

        ; continue instructions here
```

15

A Bigger Example

- Now, let's look at toupper.asm in detail… here

16

Challenge 7

```
if (eax IS NEGATIVE AND ODD) {
    eax *= 2;
} else {
    eax /= 2;
}
```

17

Challenge 7-Answer

```
        cmp     eax, 0
        jge     else
        mov     ebx, eax
        and     ebx, 1
                ;OR "test eax, 1"
        jz      else
        sal     eax   ; "then" block
        jmp     done
else:
        sar     eax
done:
```

18

Challenge 7b

```
if (eax IS NEGATIVE OR ODD) {
    eax *= 2;
} else {
    eax /= 2;
}
```

19

Challenge 7b-Answer 1

```
        Jcc(IF NOT(NEG OR ODD)) else




        sal     eax
        jmp     done
else:
        sar     eax
done:
```

20

Challenge 7b-Answer 1

```
        Jcc(IF NOT(NEG) AND NOT(ODD)) else




        sal     eax
        jmp     done
else:
        sar     eax
done:
```

21

Challenge 7b-Answer 1

```
        Jcc(IF NOT(NEG)) els-jmp


els-jmp:
        Jcc(IF NOT(ODD)) else


        sal     eax
        jmp     done
else:
        sar     eax
done:
```

Challenge 7b-Answer 1

```
        Jcc(IF NOT(NEG)) els-jmp

        ????
els-jmp:
        Jcc(IF NOT(ODD)) else


        sal     eax
        jmp     done
else:
        sar     eax
done:
```

Challenge 7b-Answer 1

```
        Jcc(IF NOT(NEG)) els-jmp

        JMP     not-else
els-jmp:
        Jcc(IF NOT(ODD)) else
not-else:

        sal     eax
        jmp     done
else:
        sar     eax
done:
```

Challenge 7b-Answer 1

```
        Jcc(IF NOT(NEG)) els-jmp

        JMP     then
els-jmp:
        Jcc(IF NOT(ODD)) else
then:

        sal     eax
        jmp     done
else:
        sar     eax
done:
```

25

Challenge 7b-Answer 1

```
        cmp     eax, 0
        jge     els-jmp
        jmp     then
els-jmp:
        Jcc(IF NOT(ODD)) else
then:

        sal     eax
        jmp     done
else:
        sar     eax
done:
```

26

Challenge 7b-Answer 1

```
        cmp     eax, 0
        jge     els-jmp
        jmp     then
els-jmp:
        test    eax, 1
        jz      else
then:
        sal     eax
        jmp     done
else:
        sar     eax
done:
```

27

Challenge 7b-Answer 2

```
        cmp     eax, 0
        jlt     then
        test    eax, 1
        jnz     then
        jmp     else
then:
        sal     eax
        jmp     done
else:
        sar     eax
done:
```

28

Challenge 7b-Answer 2

```
        cmp     eax, 0
        jlt     then
        test    eax, 1
        jnz     then
else:
        sar     eax
        jmp     done
then:
        sal     eax
done:
```

29

Challenge 8

```
/* eax is signed */
for (eax = 0; eax < ebx; eax++) {
    ebx -= eax;
    if (ebx < 0)
        break;
}
```

30

Challenge 8-Answer

```
        mov     eax, 0
loop:   cmp     eax, ebx
        jge     loop_end
        sub     ebx, eax
            ; don't have to test again
        jlt     done
        inc     eax
        jmp     loop
loop_end:
```

31

Challenge 9

You have a sequence of chars (bytes) at location "array", i.e.:

array: resb 64

which you wish to treat as a 8x8 array, in row-major order. You are given the requested row and column in EBX and ECX. Write a single "MOV" instruction to fetch that array element into AL

32

Challenge 9-Answer

```
mov     eax, [array + ebx*8 + ecx]

; technically, [ecx + ebx*8 + array],
; but NASM rearranges
```

33