IMPORTANT STUFF FOR THE LAB/SESSION

LAB 2

ADD/SUB

-the operands don’t need to have the same type

-only one op can be a memory location ([a]

**- mov [a], byte [b] WRONG**

MUL

-mul BL **AX<-AL\*BL**

-mul CX **DX:AX<-AX\*CX**

-mul EBX **EDX:EAX<-EAX\*EBX**

-the explicit operand can be a reg/variable but NOT A CONSTANT

**-mul word 6 WRONG**

DIV

-div CL AL<-AX/CL, AH<-AX%CL

-div BX AX<-DX:AX:BX, DX<-DX:AX%BX

-div DWORD[a] EAX<-EDX:EAX/DWORD[a], EDX<-EDX:EAX%DWORD[a]

-the explicit operand can be a reg/variable but NOT A CONSTANT

**-div word 6 WRONG**

**!!!!**

* **-**Dividing a large number by a small number, it is possible that the result exceeds the capacity of representation. In this case the same error is triggered as in the case of a division by 0.

NEG

-neg BX **BX<-0-BX**

LAB 3

IMUL/IDIV -same as mul/div..

Text

Description automatically generated

Text

Description automatically generated

Asa se aduna cand e quadword!!!

Sau dx:ax + cx:bx!!!!

PUSH -copies the value into the stack+decrements ESP

POP-copies the value pointed by the ESP into an operand+inc ESP

**!!!!**

The MOV instruction cannot be used to copy the flags to a variable, so PUSHFD may be the best way to save the flags

Text, letter

Description automatically generated

Text, letter

Description automatically generated

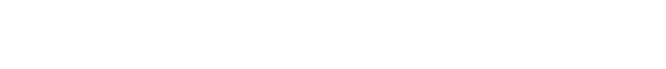
Graphical user interface, text

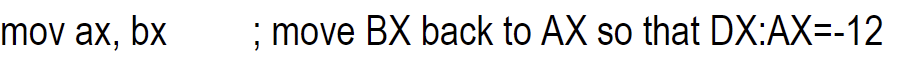
Description automatically generated

-we can do this!!

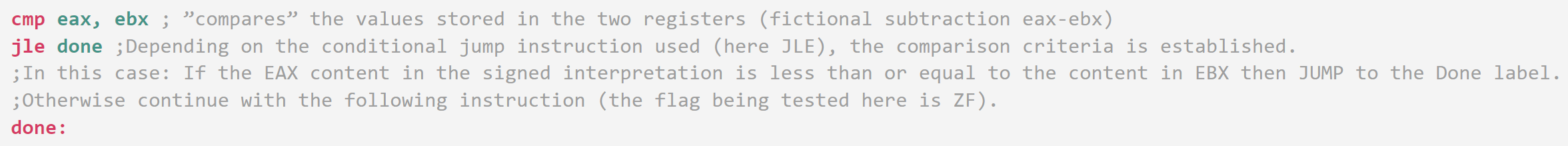
Text

Description automatically generated



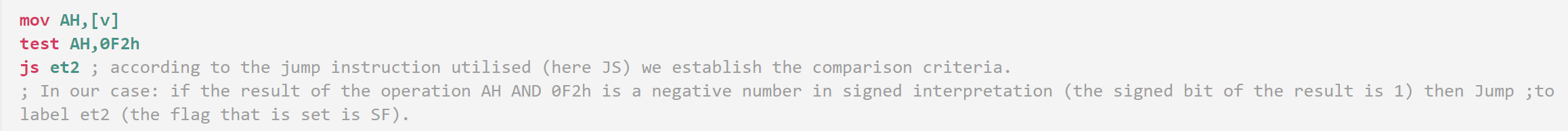


CMP



-if we compare to reg/mem operands (and the flags are set) and the comparison relationship is accordingly to the jump label, then the content from the label will be executed otherwise it will be skipped

TEST

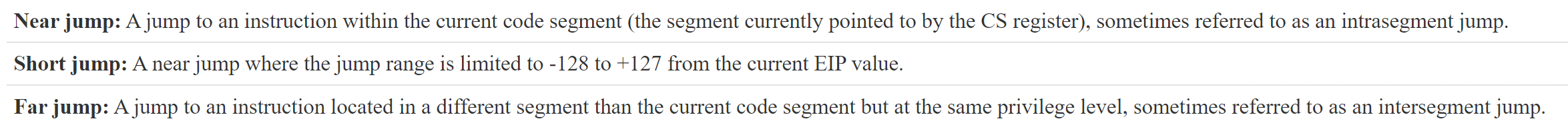


-similarly to the cmp principle, test performs the AND instruction to the operands (and modift SF,ZF, PF) and if the result is accordingly to the jump label, the code from the label will be executed

SIGNED=LESS/GREATER

UNSIGNED=BELOW/ABOVE

!!!!!!!



LAB 5

