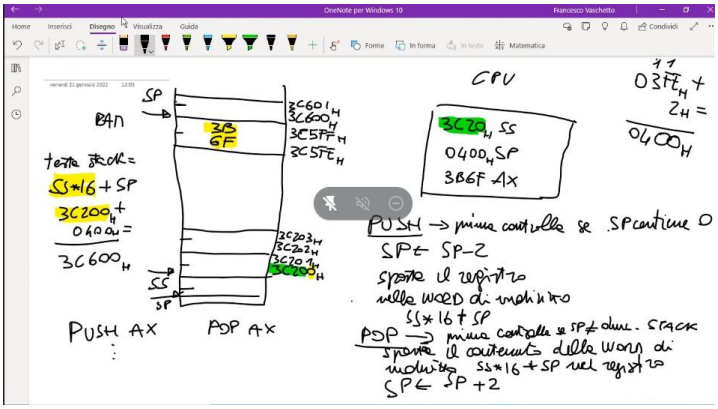
APPUNTI SISTEMI

STACK --> si trova in memoria centrale, nello specifico nella RAM, la modalità di accesso è indiretta; senza di esso non possiamo fare la chiamata alle funzioni in un programma, serve per salvare il contenuto dei registri temporaneamente. Tramite lo stack vengono inoltre passati i valori dei parametri formali. La struttura dati ha una politica di gestione **LIFO** (Last In First Out --> la prima cosa che metto è l'ultima che vado a prendere, viceversa l'ultima cosa che metto è la prima che vado a prendere->come nella lavastoviglie, il primo piatto messo è l'ultimo che vado a prendere). Quando voglio mettere un dato dello stack, non lo possono mettere dove voglio, ma lo posso mettere solo nella "testa dello stack" tramite una "PUSH" (inserzione) e lo si toglie tramite una "POP" (estrazione) - -> per fare una POP bisogna aver fatto almeno una PUSH. La POP è speculare alla PUSH.

Per gestire uno stack si usano 2 registri: **SS** (**S**tack **S**egment), contenente l'indirizzo di inizio del Segmento Stack diviso per 16 (successivamente scritto in numero binario); **SP** (**S**tack **P**ointer), inizializzato alla dimensione dello stack decisa/caricata dal programmatore (in binario) e contiene l'offset dell'ultima word salvata sullo stack.

*TESTA STACK = (SS*16) + SP (il valore verrà poi scritto ovviamente in binario)*



STACK UNDERFLOW --> quando si prova a fare una POP ma non ci sono dati. Se la dimensione di SP ha la stessa dimensione dello stack e facciamo una POP avviene un errore di stack underflow. Quando SP vale 0 significa che lo stack è pieno.

In un sistema a microprocessore Intel 8086 la dimensione massima per uno stack era di 64 KB.

Un programma con le funzioni è più lento, però a livello di occupazione di memoria è più efficiente, in quanto ci sono meno stringhe di codice.

All'inizio di ogni riga (ogni riga corrisponde ad un'istruzione) ha una serie di cifre, che definisce l'offset (distanza misurata in byte) dalla prima istruzione.