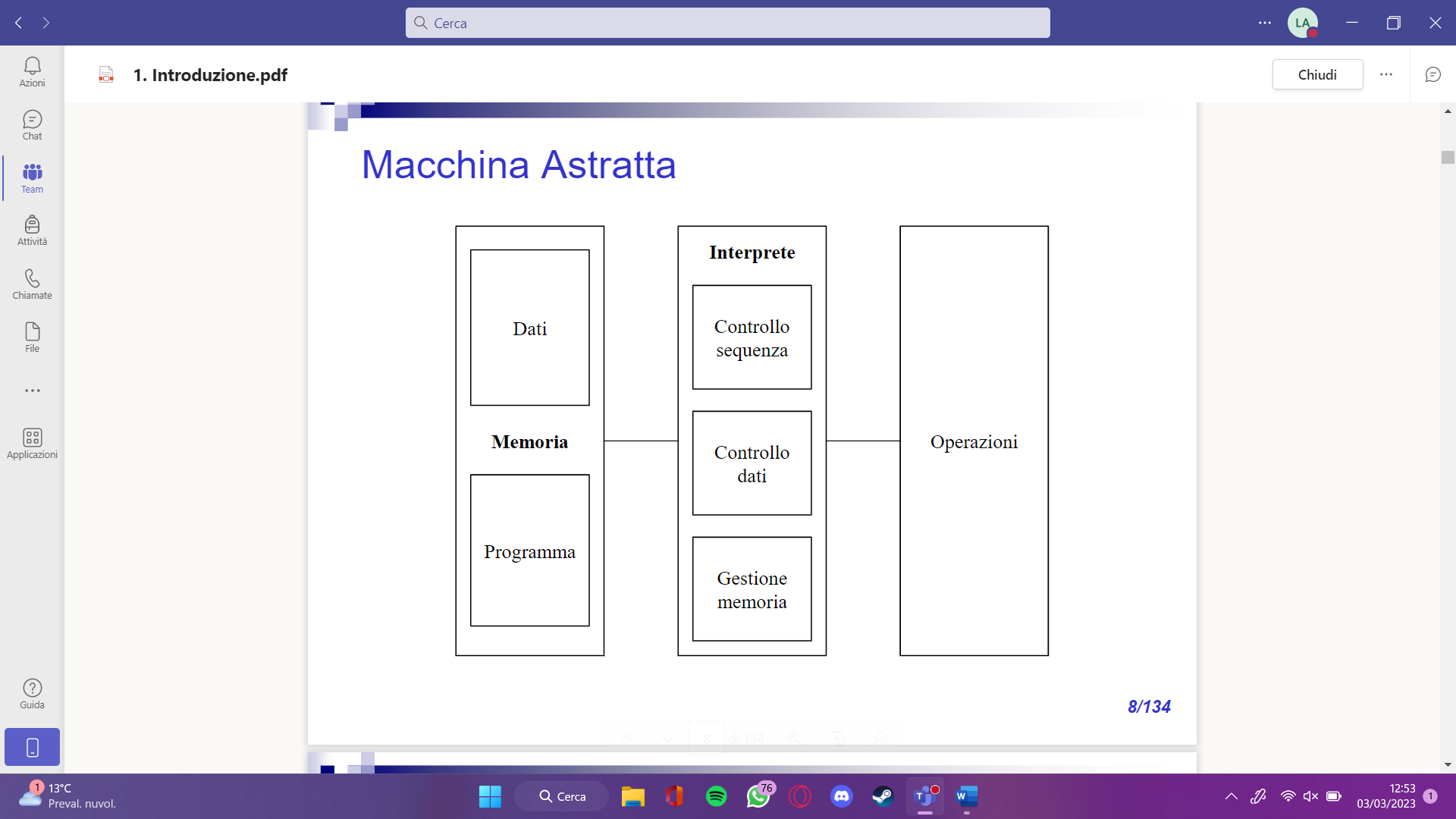
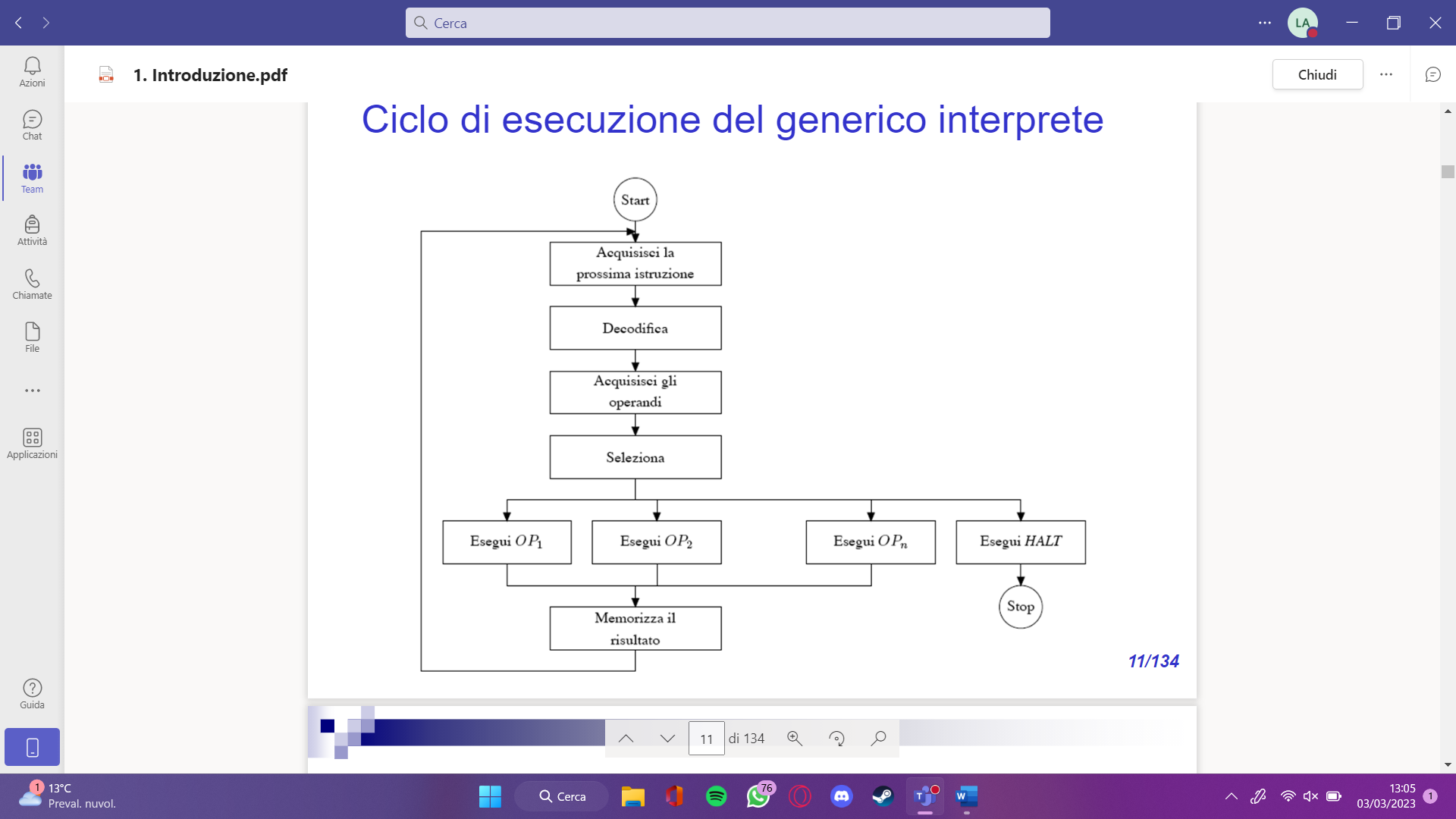
**rimo giorno linguaggi:  
-Lingue naturali**: Hanno analogie e le somiglianze derivanti dalla genealogia  
 Es: Cioccolata = ciocolata, insalta = salata, treno =tren.  
**-Linguaggi di programmazione**: è difficile conoscere un gran numero in modo approfondito  
 -E’ possibile conoscere a fondo i meccanismi che ispirano il progetto e **l’implementazione.** -E’ difficile conoscerne un gran numero in modo approfondito.  
**Macchina Astratte e implementazione dei linguaggi di programmazione**  
Qual è il significato dell’espressione: **implementazione di un linguaggio di programmazione?**  
Il concetto è strettamente collegato a quello delle **macchine astratte**  
**Calcolatore**: macchina fisica e inoltre:   
- Consente di eseguire algoritmi formalizzati(tradotti) perché siano “comprensibili” dall’esecutore.  
 - **Formalizzazione**: codifica di algoritmi in un certo linguaggio (**L**) definito da una specifica sintassi.  
 - La sintassi di **L** permette di utilizzare i costrutti per comporre programmi in **L**.   
- Un programma in L è una sequenza di istruzioni del linguaggio **L**.  
**Macchina astratta**: Non è ben definita e avviene l’astrazione del concetto di calcolatore fisico (è come il calcolatore ma senza hardware associato (astratta)). -solo software.   
**Macchina astratta per L:**-Un insieme di algoritmi e strutture dati che permettono di eseguire programmi scritti in **L**.  
-Questa macchina è denotata come: **ML.**  
Composta da :  
- **Una memoria** per immagazzinare dati e programmi  
- **Un interprete** per interpretare ed eseguire le istruzioni contenuti nei programmi.  
**Linguaggio Macchina**:  
Se abbiamo una macchina astratta **ML**, il linguaggio **L** “compreso” dall’interprete di **ML** è detto linguaggio macchina di **ML.**  
**Macchina Astratta:**  
**Interprete:**Se siamo in presenza di un **L**, l’interprete è un programma che si occupa del controllo della sequenza delle istruzioni del linguaggio, controllo dei dati e gestire la  
memoria, es: va a prendere i valori delle variabili dalla memoria.  
**Operazioni dell’interprete:  
1. Operazioni per elaborazione dei dati primitivi:**- Numeri interi, reali.  
- Operazioni aritmetiche.  
**2. Operazioni e le strutture dati per il controllo della sequenza di esecuzione:**- Gestiscono il flusso di controllo delle istruzioni.  
- Strutture dati per memorizzare l’indirizzo della prossima istruzione  
- Operazioni per manipolare le strutture dati, Es: Calcolo indirizzo prossima istruzione.  
**3. Operazioni e strutture dati per il controllo del trasferimento dei dati:**- Gestisco il trasferimento dei dati dalla memoria all’interprete e viceversa (Sia dati che indirizzi).  
Es: recupero gli operandi  
- Possono far uso di strutture dati ausiliarie  
 es: pila.  
**Operazioni e strutture dati per la gestione della memoria:**-allocazione della memoria per i dati e programmi.  
  
**Traduzione foto:** Acquisisce la prossima istruzione -> Decodifica(cerca di capire che tipo di istruzione è), in base all’istruzione acquisisce degli operandi e successivamente seleziona il tipo di operazione e la esegue, dopo memorizza il dato e ritorna all’inizio del ciclo, non ha termine fino a quando il programma è terminato(END).  
  
 **Realizzazione della macchina astratta:  
1.** Una **ML** per essere utilizzata deve prima o poi utilizzare qualche dispositivo fisico.  
Realizzazione “fisica” in hardware:  
 - Algoritmi di **ML** realizzati mediante dispositivi fisici.  
**2.** Realizzazioni che usano livelli intermedi tra **ML** e dispositivo fisico.  
- **Simulazione mediante software.**  
-Emulazione mediante firmware(microprogrammi in linguaggi di basso livello).  
**Firmware:** Hardware programmabile(eprom).(ne software ne hardware)  
**REALIZZAZIONE MEDIANTE SOFTWARE:**  
- Algoritmi e strutture dati di **ML** realizzati in un altro linguaggio **L’** già implementato.  
- **ML** è realizzata mediante programmi **L’** che simulano le funzionalità di **ML**.  
- **ML** è realizzata attraverso la macchina **M’L’.**  
- **M’L’** è detta macchina ospite denotata con **MoLo.**- L’implementazione di **L** sulla macchina ospite avviene mediante una qualunque “**traduzione”** di **L** in **Lo.**A seconda della traduzione si parla di:  
- **Implementazione interpretativa.**- **Implementazione compilativa.**  
**FUNZIONI PARZIALI:  
Una funzone parziale**  
  
  
è una corrispondenza tra elementi dell’insieme A e quelli dell’insieme B.  
- **Parziale**: La corrispondenza può essere non definita per qualche elemento di A.  
- Dato a A, può esistere un corrispondente in B,denotato con f(a) altrimenti non esiste.  
I programmi definiscono funzioni parziali:  
read(x);  
if x==1 then print(x) else   
while (x<>1) do skip;  
  
  
**Definizione di interprete:**  
Un programma scritto in linguaggio **L** si può vedere come una funzione parziale:   
  
Possiamo definire la seguente definizione di interprete di **L** in **Lo:**



Non vi è traduzione esplicita dei programmi scritti in **L,** solo il procedimento di decodifica.  
L’interprete per eseguire l’istruzione **I** di **L,** le fa corrispondere ad un insieme di istruzioni in **Lo.** Tale decodifica non è una traduzione esplicita poiché il codice a **i** è eseguito direttamente e non prodotto in uscita.

L’interprete funziona bene se preso l’ingresso del programma è in grado di restituirmi l’output.  
arriva fino a pag 19 a scrivere e studiare. **DEFINIZIONE DI COMPILATORE**  
Un compilatore da **L** a **Lo** è un programma che realizza la funzione:



Traduzione esplicita dei programmi scritti in **L** in programmi scritti in **Lo**  
  
  
Per eseguire **P** su Input, bisogna eseguire CL,Lo con **P** come input.  
Si avrà come risultato un programma compilato PC scritto in **Lo,** che sarà eseguito su **MoLo** con il dato in ingresso Input.  
**Interpretazione vs Compilazione:  
-Interpretazione  
Scarsa efficienza:**  
 -decodifica costrutti a tempo di esecuzione.  
 -decodifica di uno stesso comando ripetuto.  
**Maggiore flessibilità:**  
 -debugging più semplice.  
**-Compilazione:  
Maggiore efficienza:** -decodifica di un’istruzione fatta una sola volta indipendentemente da quante volte è eseguita.  
**Minore flessibilità:** -perdita di informazioni rispetto al programma sorgente(difficile tracciare errori e dunque maggiori difficoltà nel debugging).



**GERARCHIE DI MACCHINE ASTRATTE**Pensiamo ad una gerarchia di macchine astratte come: **ML0,ML1,ML2,…MLn.  
MLi** è implementata sfruttando il linguaggio della macchina sottostante **MLi-1.  
MLi** fornisce a sua volta il proprio linguaggio alla macchina sovrastante **MLi+1.**Indipendenza fra livelli:Modifiche interne alla funzionalità di un livello non hanno influenza sugli altri livelli.  
**Es gerarchia di macchine astratte:**

