VERIFICA 3D

1. Pipeline:
   "La pipeline è una tecnica di parallelismo a livello di istruzioni che permette l'esecuzione sovrapposta di più istruzioni diverse. Il ciclo di esecuzione viene suddiviso in 5 fasi:

- IF (Instruction Fetch): preleva l'istruzione dalla memoria
- ID (Instruction Decode): decodifica l'istruzione e legge gli operandi
- EX (Execute): esegue l'operazione
- MEM (Memory): accede alla memoria se necessario
- WB (Write Back): scrive il risultato nei registri

Ad esempio, mentre un'istruzione è in fase EX, la successiva può essere in fase ID e una terza in fase IF, aumentando significativamente il throughput del processore.

I principali problemi sono:

- Dipendenze dai dati: quando un'istruzione necessita del risultato di quella precedente
- Dipendenze di controllo: salti condizionati che modificano il flusso di esecuzione
- Conflitti strutturali: più istruzioni necessitano della stessa risorsa

Per mitigare questi problemi si usano tecniche come:

- Forward: anticipazione dei risultati
- Branch prediction: previsione dei salti
- Pipeline stall: inserimento di cicli di attesa

Un esempio pratico:

```
ADD R1,R2,R3  ; R1 = R2 + R3
SUB R4,R1,R5  ; R4 = R1 - R5
```

La SUB deve attendere il risultato della ADD -> dipendenza dai dati"

2. CISC vs RISC:
   "Le architetture CISC (Complex Instruction Set Computing) e RISC (Reduced Instruction Set Computing) rappresentano due approcci diversi alla progettazione dei processori:

CISC:

- Molte istruzioni complesse
- Formati di istruzione variabili
- Istruzioni multi-ciclo
- Minor numero di registri
- Maggiore densità di codice
  Esempio: x86

RISC:

- Poche istruzioni semplici
- Formato fisso delle istruzioni
- Istruzioni single-cycle
- Molti registri
- Ottimizzato per pipeline
  Esempio: ARM

I processori moderni usano un approccio ibrido perché:

- Mantengono compatibilità con software legacy (CISC)
- Internamente decodificano in micro-operazioni RISC
- Sfruttano pipeline e parallelismo
- Ottimizzano prestazioni ed efficienza energetica

Ad esempio, i processori x86 moderni:

1. Decodificano istruzioni CISC complesse
2. Le convertono in micro-op RISC
3. Le eseguono su core RISC ottimizzati
4. Mantengono compatibilità x86"
5. Indirizzamento x86:
   "I metodi di indirizzamento x86 definiscono come accedere agli operandi. I principali sono:
6. Immediato:

```
MOV AX,5  ; valore immediato 5 in AX
```

2. Registro diretto:

```
MOV AX,BX  ; copia BX in AX
```

3. Memoria diretta:

```
MOV AX,[1000h]  ; carica in AX il contenuto della cella 1000h
```

4. Registro indiretto:

```
MOV AX,[BX]  ; usa BX come puntatore
```

5. Base + Index + Displacement:

```
MOV AX,[BX+SI+10]  ; BX=base, SI=index, 10=displacement
```

L'indirizzo effettivo si calcola come:
Effective Address = Base Address + Index + Displacement

Questo permette:

- Accesso flessibile agli array
- Gestione stack e heap
- Indirizzamento strutture dati
- Ottimizzazione accesso memoria"

4. Esercizio Assembly:
   "Analizziamo il codice:

```
MOV AX,5    ; Inizializza AX=5
MOV BX,3    ; Inizializza BX=3
MOV CX,0    ; Inizializza contatore CX=0

start:      ; Etichetta inizio ciclo
ADD AX,BX   ; Somma BX ad AX
DEC CX      ; Decrementa CX
CMP CX,-5   ; Confronta CX con -5
JNE start   ; Salta a start se CX≠-5
```

Il programma:

1. Somma il valore 3 (BX) al valore iniziale 5 (AX) per 5 volte
2. Usa CX come contatore che va da 0 a -5
3. Il ciclo viene eseguito 5 volte

Calcolo valore finale AX:

- Valore iniziale: 5
- 5 iterazioni sommando 3: 5 + (3 × 5) = 20

Quindi:

- AX finale = 20
- Ciclo eseguito 5 volte
- CX finale = -5"