1. Găsiţi şi descrieţi o altă arhitectură de sistem de calcul decât arhitectura von Neumann. Care sunt avantajele şi dezavantajele arhitecturii găsite

O altă arhitectură de sistem de calcul este **Arhitectura Harvard** .Aceasta are diferite cai de [stocare](https://hmn.wiki/ro/Computer_storage" \o "Stocare pe computer)  și de semnal pentru [instrucțiuni](https://hmn.wiki/ro/Machine_code" \o "Codul mașinii) și [date](https://hmn.wiki/ro/Data) . Acesta contrastează cu [arhitectura von Neumann](https://hmn.wiki/ro/Von_Neumann_architecture" \o "Arhitectura Von Neumann) , în care instrucțiunile și datele programului împărtășesc aceeași memorie și căi.

Termenul provine de la computerul bazat pe releu [Harvard Mark I](https://hmn.wiki/ro/Harvard_Mark_I) , care stoca instrucțiuni pe [bandă perforată](https://hmn.wiki/ro/Punched_tape" \o "Bandă perforată) (lată de 24 de biți) și date în contoare [electro-mecanice](https://hmn.wiki/ro/Electro-mechanical).

Avantaje:

* Performanță sporită: Arhitectura Harvard utilizează două seturi separate de memorie pentru date și instrucțiuni, ceea ce permite paralelism mai mare și o potențială creștere a performanței.
* Securitate îmbunătățită: Datorită separării memoriei pentru date și instrucțiuni, există mai puține șanse pentru exploit-uri care să permită codului să modifice datele în mod neautorizat.
* Eficiență în anumite aplicații: Arhitectura Harvard poate fi eficientă în aplicații specializate, cum ar fi microcontrolerele, în care instrucțiunile și datele sunt citite în mod frecvent și în paralel.

Dezavantaje:

* Complexitate crescută: Implementarea arhitecturii Harvard poate fi mai complexă decât cea a arhitecturii Von Neumann, ceea ce poate duce la costuri și complexitate suplimentare.
* Limitări în general-purpose computing: Arhitectura Harvard poate fi mai puțin eficientă în calculul general decât arhitectura Von Neumann, deoarece instrucțiunile și datele trebuie să fie gestionate separat.
* Costuri mai mari: Hardware-ul necesar pentru a implementa arhitectura Harvard poate fi mai scump din cauza memoriei separate pentru instrucțiuni și date.

1. Daţi un exemplu de localizare a unui fişier într-un sistem de directoare.

EXEMPLU🡪 C:\Utilizatori\NumeUtilizator\Documente\Imagine.jpg

În acest exemplu "C:" reprezintă unitatea de disc principală pe un sistem Windows. "Utilizatori" este numele directorului principal, "NumeUtilizator" este numele utilizatorului și "Documente" este un subdirector în care se află fișierul "Imagine.jpg".

1. Căutaţi pe Internet o adresa care ofera informatii despre limbajul C/C++

<https://www.w3schools.com/cpp/cpp_intro.asp>

1. Nominalizaţi un sistem încapsulat (embedded) care se programează in C++, altul in C.

Un exemplu de sistem încapsulat (embedded) programat în C ar putea fi un microcontroller AVR (cum ar fi cele folosite în platforma Arduino) care utilizează în principal limbajul C pentru dezvoltarea software-ului. Acesta este un sistem încapsulat de dimensiuni reduse care poate fi programat pentru a controla diverse dispozitive sau pentru a realiza proiecte de electronica

În ceea ce privește programarea în C++, un exemplu ar putea fi dezvoltarea software-ului pentru sisteme încapsulate bazate pe platforme mai puternice, precum Raspberry Pi. Deși Raspberry Pi nu este un sistem încapsulat tradițional, este un mini-computer care poate fi folosit pentru o varietate de aplicații încapsulate. Dezvoltatorii pot utiliza C++ pentru a crea aplicații și programe care rulează pe Raspberry Pi, făcând uz de avantajele acestui limbaj, cum ar fi programarea orientată pe obiect și gestionarea mai eficientă a memoriei.