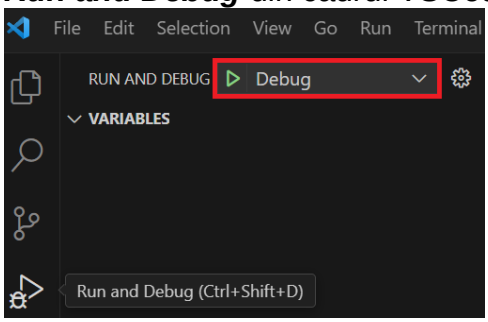
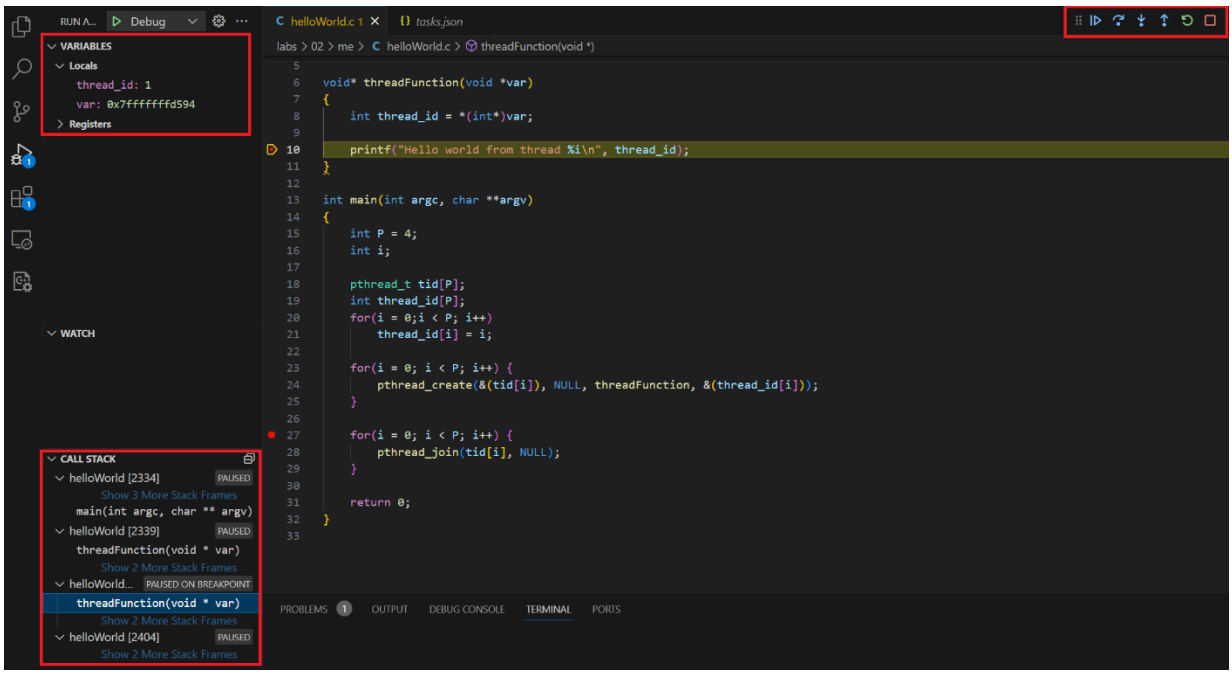
Laborator 02

Exerciții

1. Compilați și rulați codul din **helloWorld.c** .
2. Compilați programul folosind modul debug, din interfață.
   - Pentru alegerea configurației de debug este necesar să accesați secțiunea ***Run and Debug*** din cadrul VSCode
   - 

   - Configurațiile de **Run and Debug** pot fi modificate prin intermediul fișierelor **launch.json** și **tasks.json** din directorul **.vscode**. Pentru ca VSCode să le recunoască, acesta trebuie să fie deschis în directorul părinte al directorului **.vscode**. Pentru mai multe informații, accesați documentația oficială.
   - Setați un breakpoint.
   - Rulați până la breakpoint apoi mergeți prin cod pas cu pas.
   - Vizualizați thread-urile.



3. Scrieți un program **useAllCPU.c** care folosește 100% din toate core-urile disponibile.

- Programul trebuie să ruleze permanent, sau minim câteva minute.
- Vizualizați folosirea core-urilor în task manager (sau htop pe Linux).
- Aveți grijă să **nu** compilați cu **-O3** (modificați .vscode/tasks.json). Este posibil ca acest mod să șteargă tot codul.
- Nu trebuie să folosiți mult RAM, în schimb puteți face multe calcule (sqrt), pe aceeași variabilă.

4. Paralelizați adunarea a doi vectori din **addVectors-par.c** .
   - **N** numărul de elemente (argument 1).
   - **printLevel** controlează câte elemente sunt printate (argument 2).
   - **P** numărul de thread-uri (argument 3).
   - Argumentele de rulare a programului prin VSCode pot fi modificate din .vscode/tasks.json .
   - **NReps** reprezintă de câte ori executăm adunarea vectorilor. Îl folosim să controlăm viteza de execuție deoarece o singura adunare chiar și a unor vectori mari poate să fie extrem de rapidă.
   - for-ul cu **NReps** trebuie să fie copiat fără modificare în interiorul thread-urilor.
5. Rulați programul **addVectors-par** paralelizat cu număr diferit de thread-uri. Vrem să determinăm cum scalează.
   - Pentru a măsura timpul de execuție puteți să deschideți un terminal nou în Visual Studio Code apăsând pe + și scriind **time addVectors-par N 0 P**
   - Vom rula fără afișare, adică **printLevel=0**, deoarece afișarea în sine poate dura extrem de mult și ne va ascunde scalabilitatea.
   - Timpul de execuție cu un singur thread trebuie să fie de peste 10 secunde, altfel timpii sunt prea mici pentru a fi relevanți. Setați **N** și **NReps** corespunzător.
   - Testați cu **P=1**, **P=2** sau **P=4**. La **P** (număr thread-uri) mare nu vom observa scalabilitate.
     - Dacă **P>numCoreReal** (fară hyperthreading) un program va scala foarte puțin, uneori deloc.
     - Dacă **P>numCoreHyperThread** în general un program nu va mai scala, ba chiar va crește timpul de execuție (What is Hyper-Threading).

Exercițiile de la 1 la 5 sunt **obligatorii**. Conceptele explorate sunt esențiale pentru obținerea notei **minime** de promovare.

Vă recomandăm, pentru a crește șansele de a obține o notă cât mai mare să explorați și următoarele exerciții:

6. Scrieți un program care face adunarea a două matrici.
7. Faceți o copie a programului anterior și paralelizați.
8. Scrieți un program care să caute un subșir într-un șir de caractere.
9. Faceți o copie a programului anterior și paralelizați.
   - Mare grijă la cazul în care subșirul căutat este împărțit când setați de care caractere se ocupă fiecare thread.