1 Installazione e setup dei simulatori per Windows e macOS

1.1 Premessa fondamentale: architetture e compatibilità

Prima di procedere all'installazione e al setup dei simulatori è necessario fare una premessa importante che riguarda l'ecosistema nel quale vengono eseguiti.

Questi due simulatori architetturali (Gem5 e MarssX86) nascono e vengono sviluppati nativamente per sistemi Unix-like.

- **macOS** è un sistema certificato Unix. "Parla la stessa lingua" di Linux, quindi può eseguire molti di questi programmi nativamente (direttamente dal terminale), con il solo aiuto di alcune librerie esterne.
- **Windows** ha un'architettura completamente diversa, per cui non può eseguire nativamente questi programmi. Per poterli eseguire è quindi necessario utilizzare delle soluzioni di **virtualizzazione o sottosistemi** per creare l'ambiente Unix necessario.

Bisogna anche considerare che, anche avendo un sistema Linux (o Unix), c'è il problema delle librerie software necessarie per compilare il codice (il "Time Gap").

- **Gem5** è un software **moderno e adattivo**, che viene continuamente sviluppato e aggiornato. Funziona senza problemi sui sistemi operativi moderni: su macOS lo si compila direttamente mentre su Windows si usa WSL con un'installazione Linux recente (Ubuntu 20.04/22.04).
- **MarssX86** è un progetto "legacy" (molto vecchio, fermo ad 2012 circa). Richiede compilatori (GCC vecchi) e librerie che sono state rimosse dai sistemi operativi moderni da anni.

Pertanto su Windows e macOS **non è possibile installarli** come semplici programmi `.exe` o `.app`, ed è necessario utilizzare delle soluzioni di **virtualizzazione o sottosistemi** per creare l'ambiente Linux necessario.

1.2 Installazione e setup del simulatore Gem5

Per quanto riguarda l'installazione del simulatore Gem5, la soluzione varia leggermente a seconda del sistema operativo utilizzato.

1.2.1 Installazione e setup di Gem5 su Windows

Per installare il simulatore su Windows la soluzione migliore è **WSL** (Windows Subsystem for Linux). Questo sottosistema permette di avere un terminale Ubuntu vero e proprio integrato in Windows. Dunque, l'installaziojne si compone di diversi passaggi:

- ***Passo 1*: attivazione di WSL**
  1. Aprire la **PowerShell** come amministratore;
  2. Digitare `wsl --install` e premere invio;
  3. Attendere il download e, una volta completato, riavvare il computer;
  4. Al riavvio, seguire le istruzioni a schermo per creare *username* e *password* Ubuntu.

- ***Passo 2*: installazione delle dipendenze**
  Nel terminale di Ubuntu appena aperto incollare questi comandi uno alla volta:

```
//Aggiorna i pacchetti
sudo apt update && sudo apt upgrade -y
```

```
//Installa compilatori, Python e librerie necessarie a Gem5
sudo apt install build-essential git m4 scons zlib1g zlib1g-dev
   libprotobuf-dev protobuf-compiler libprotoc-dev libgoogle-perftools-
   dev python3-dev python-is-python3 libboost-all-dev pkg-config -y
```

- ***Passo 3*: scaricare e compilare Gem5**

```
//Clonare il repository del progetto nel proprio workspace
git clone https://github.com/gem5/gem5.git
```

```
//Entra nella cartella
cd workspace/gem5
```

```
//Avviare la compilazione (ci vorranno dai 10 ai 40 minuti)
//Questo comando usa tutti i core della tua CPU per fare prima
scons build/ARM/gem5.opt -j $(nproc)
```

Note: se si vuole simulare RISC-V, bisogna sostituire `ARM` con `RISCV` nel comando `scons`.

1.2.2 Installazione e setup di Gem5 su macOS