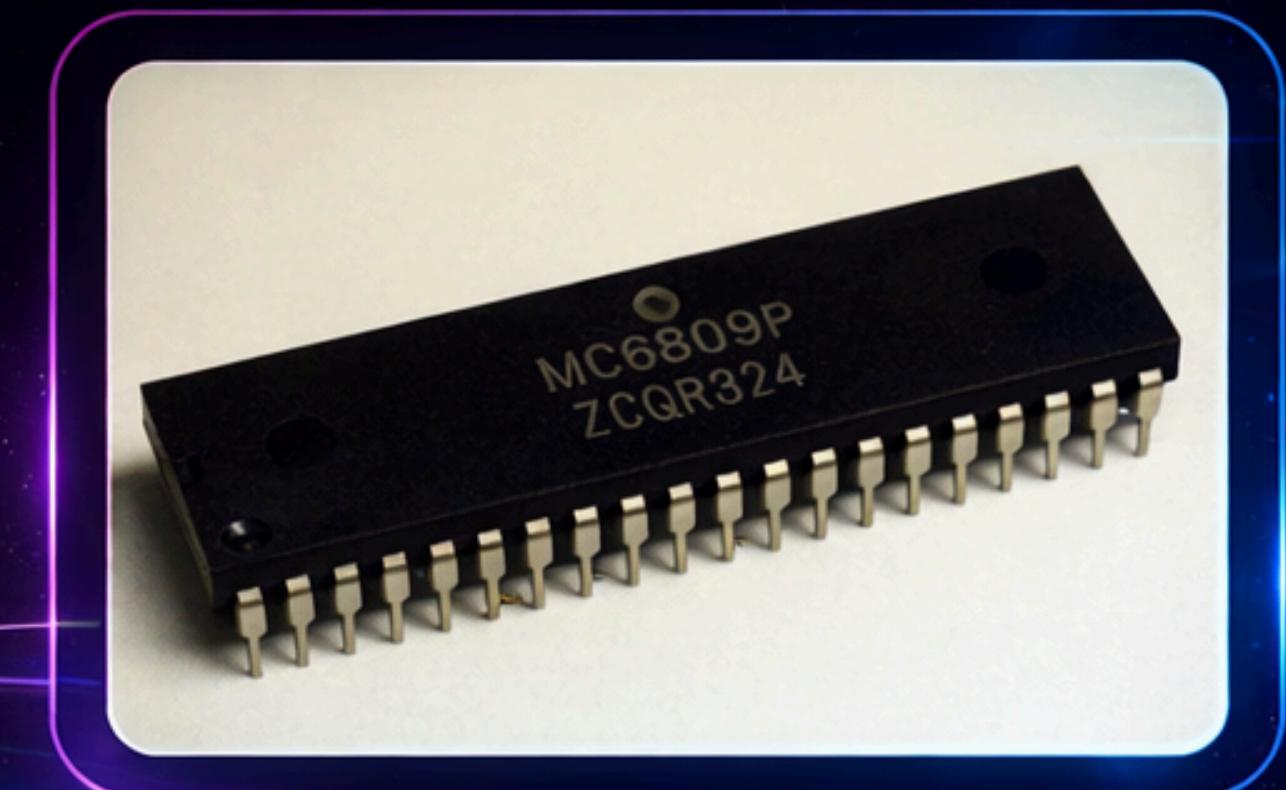


# Motorola 6809

Ce projet consiste à créer un simulateur du microprocesseur Motorola 6809, un processeur 8/16 bits, permettant d'émuler son jeu d'instructions et de déboguer des programmes.



Encadré par :  
**Hicham Ben Alia**

Présenté par :  
**Hafsa Barbey & Fadila Chritt**

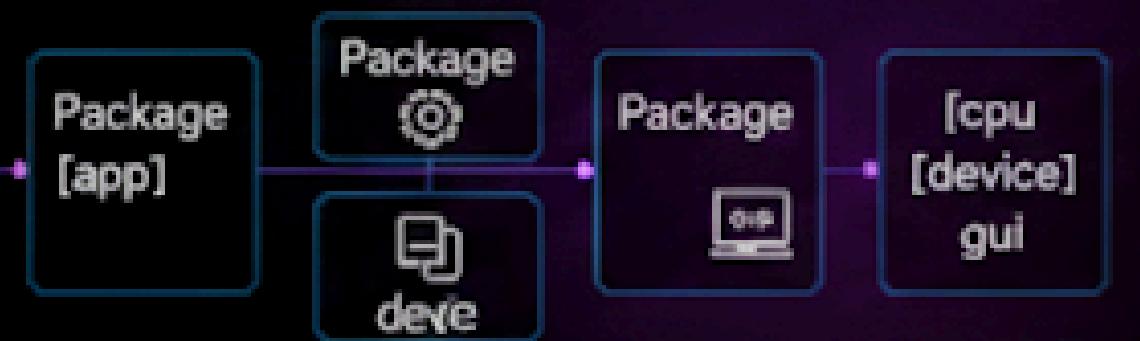
# PROJET DU MODULE: SIMULATOR M6809

## PLAN DE LA PRÉSENTATION

### 1. Introduction

- Historique
- Définition du Simulateur

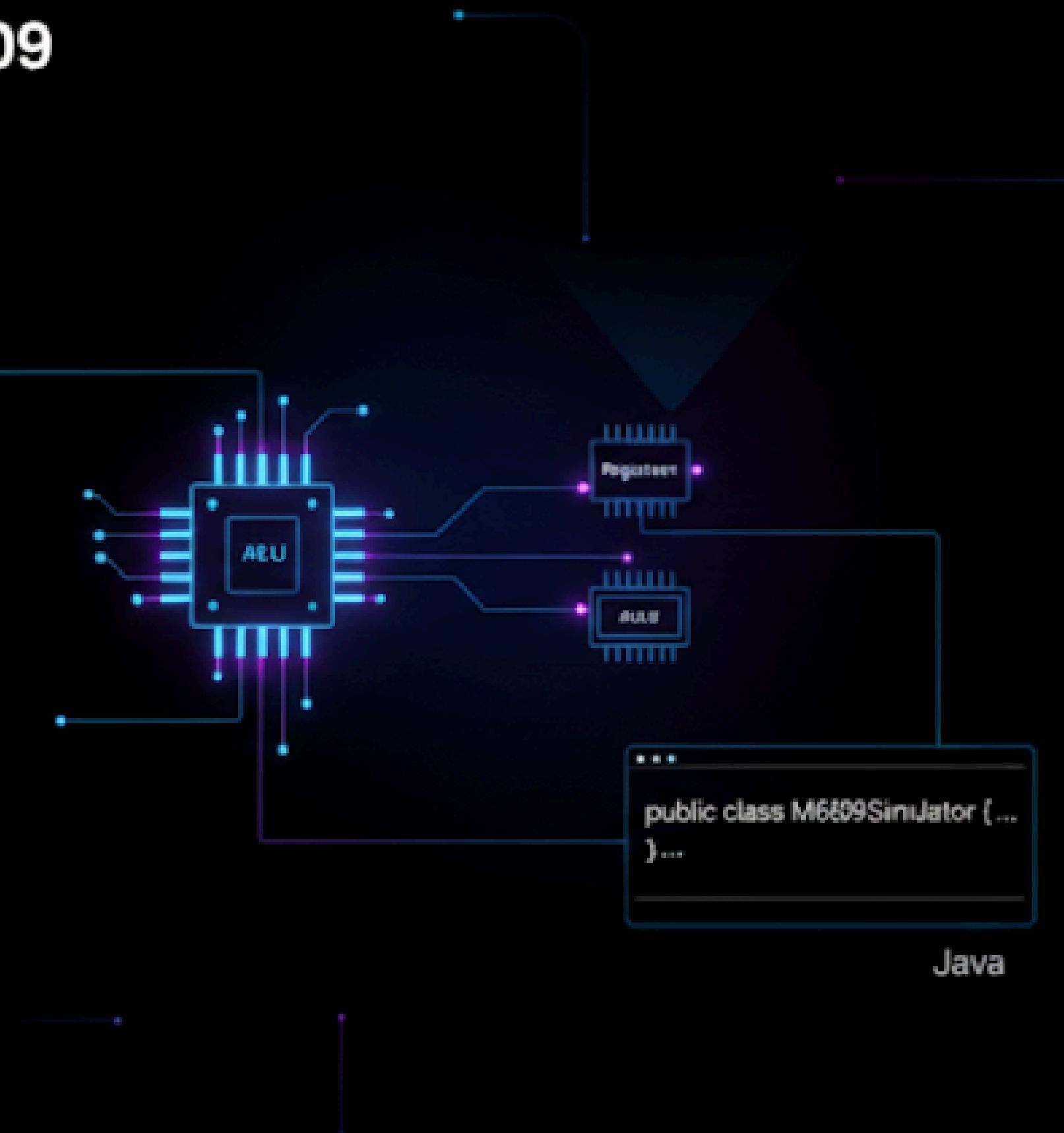
### 2. Architecture Logicielle



### 3. Détails de l'Interface

### 4. Tests & Execution

### 5. Conclusion & Perspectives



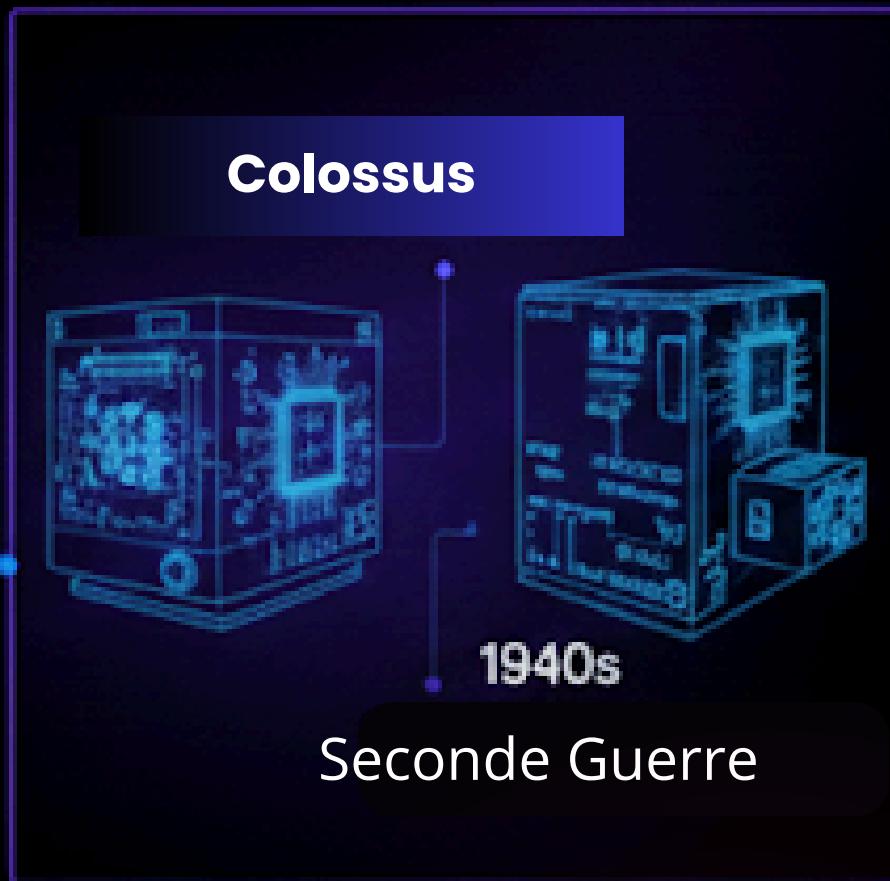
# INTRODUCTION

## Historique: de la naissance des ordinateurs au microprocesseurs

Premiers Ordinateurs



Ordinateurs et Guerre

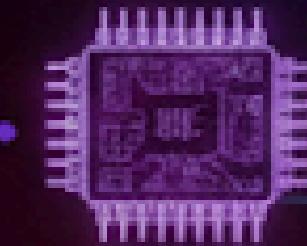


Guerre froide

Avancées Technologiques

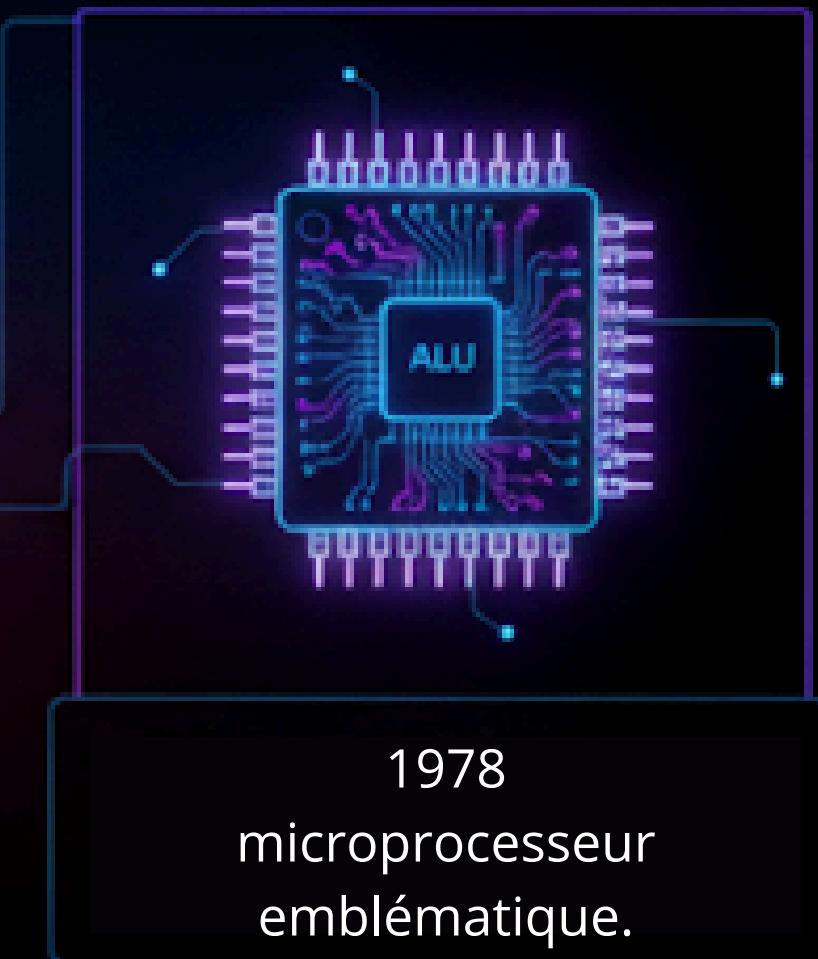


Invention du  
Transistor  
(1947)



Circuits  
Intégrés  
(1960s)

Motorola 6809



# Définition: Du Rèel au Virtuel

1.Motorola 6809 Physique



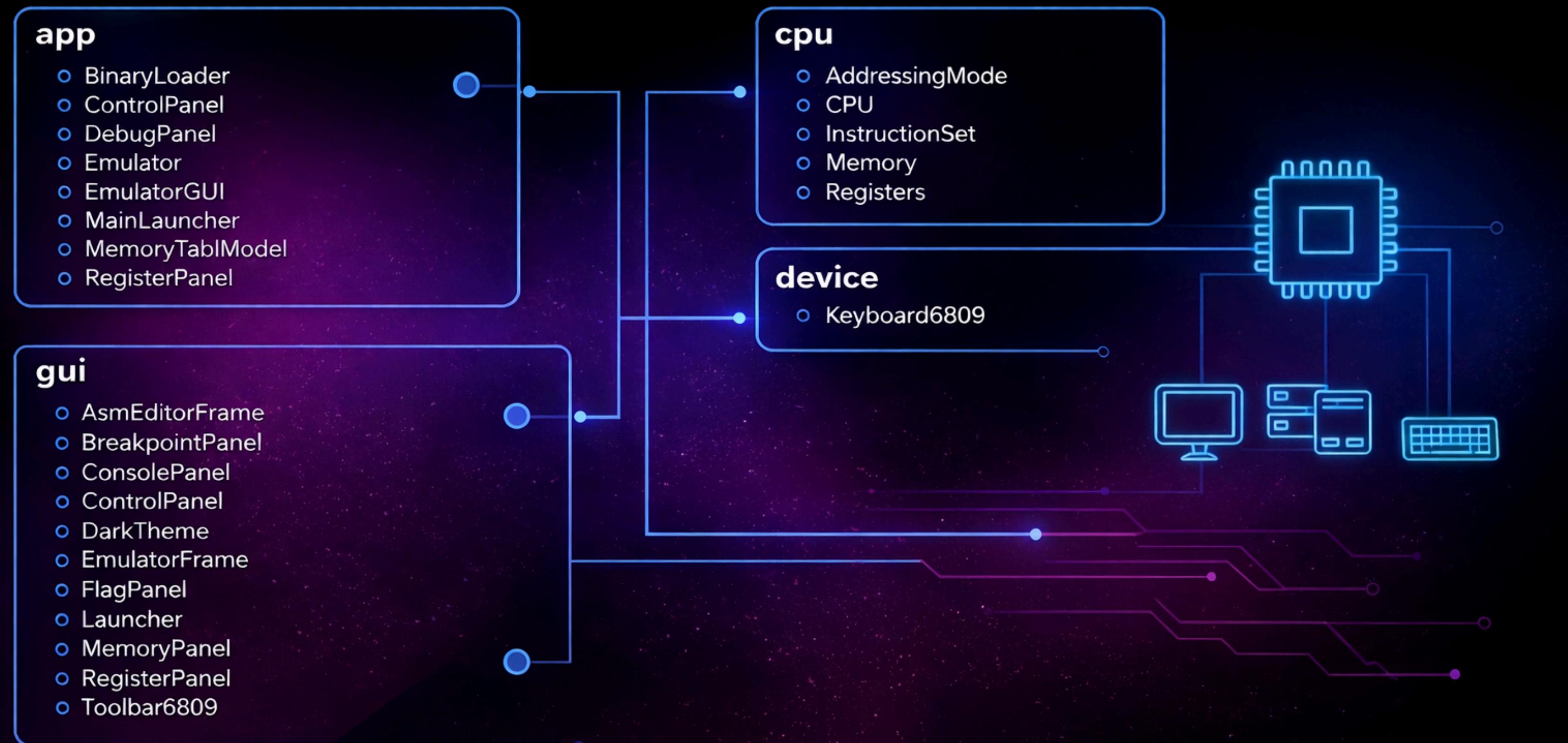
microprocesseur puissant et flexible inventé en 1978, utilisé dans les systèmes embarqués et pour l'enseignement, avec un jeu d'instructions avancé et plusieurs modes d'adressage.

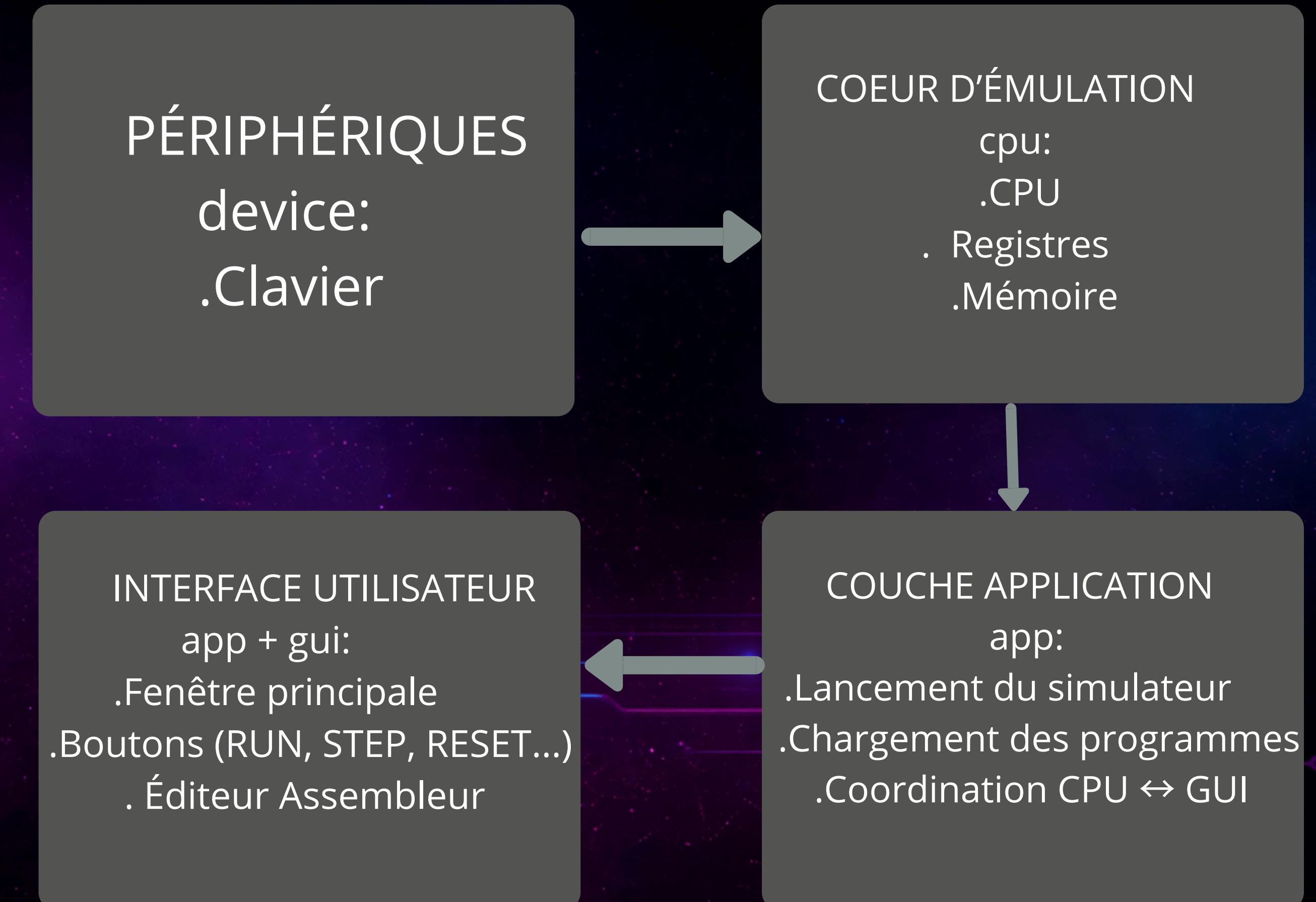
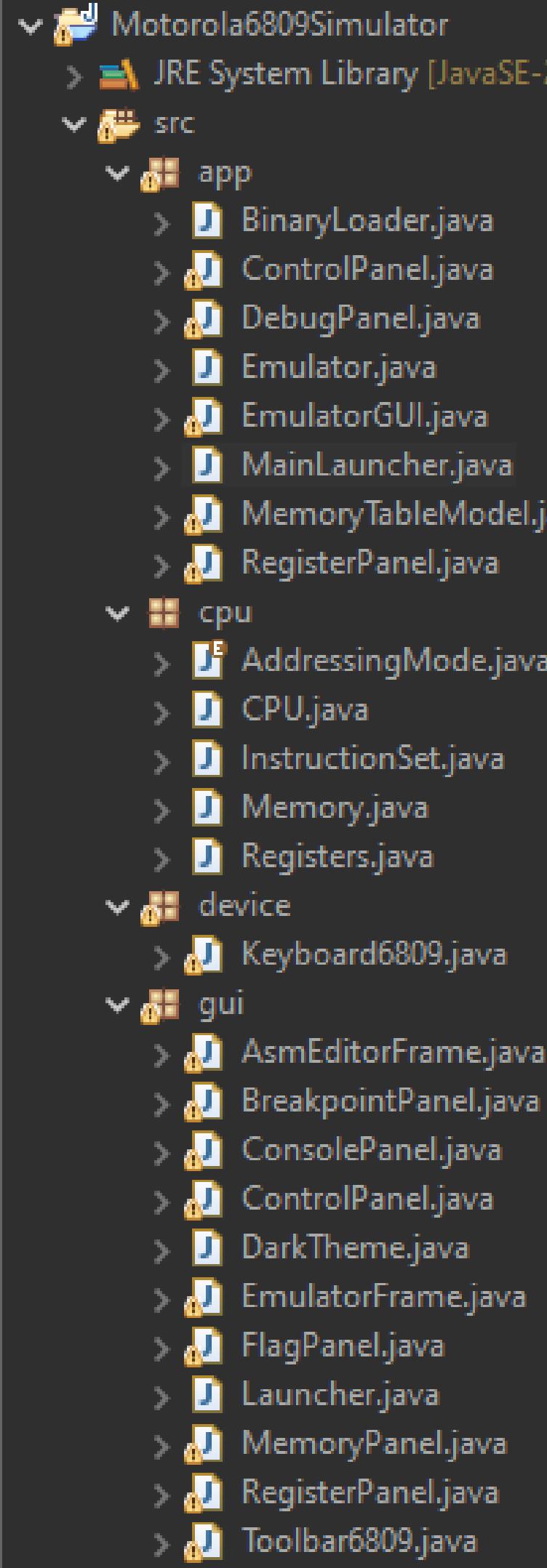
2.Notre Simulateur Logiciel



logiciel développé en Java, offrant un environnement d'émulation pour apprendre, tester et développer des programmes pour le microprocesseur 6809.

# Architecture Logicielle





```
public void step() {
    if (halted || waitingForInterrupt)
        return;

    if (irqPending) {
        handleIRQ();
        irqPending = false;
        return;
    }

    if (breakpoints.contains(reg.PC & 0xFFFF))
        return;

    int opcode = fetchByte();
    InstructionSet.Instruction
    inst = iset.get(opcode);

    if (inst == null && opcode == 0x10)
        inst = iset.get(0x100 | fetchByte());
    else if (inst == null && opcode == 0x11)
        inst = iset.get(0x1100 | fetchByte());

    if (inst == null) {
        System.err.printf("Opcode inconnu %02X @ %04X%n",
                          opcode, (reg.PC - 1) & 0xFFFF);
        halted = true;
        return;
    }

    inst.execute(this);
    instructionsExecuted++;
}
```

# CPU \_ Cycle d'exécution

## 1/FETCH:

Lit l'opcode à l'adresse PC.

## 2/DECODE:

Identifie l'instruction correspondante  
à l'opcode.

## 3/EXECUTE:

Exécute l'opération de l'instruction.

## 4/Mise à jour du PC:

Met à jour PC pour pointer vers  
l'instruction suivante.

