

# Projet de Semestre

Dylan Frei

h e p i a

Haute école du paysage, d'ingénierie  
et d'architecture de Genève

**Hes·SO** GENÈVE

Haute Ecole Spécialisée  
de Suisse occidentale



# INTRODUCTION

---

- Projet de Semestre en deux parties
- Custom Embedded Linux
- HEPIA-RISC & QEMU



# Custom Embedded Linux

## INTRODUCTION

- SoC Nvidia
- Source : Documentation officielle
- Client : ANTS AI Systems
  - Custom Carrier Board
- Objectif : Investigation

# TABLE OF CONTENTS

## 01

### **Jetson Linux Boot Flow**

Composants et firmwares  
du processus de boot

## 02

### **Modifications requises**

Liste des modifications  
pour carrier board  
custom

## 03

### **Customisation du système**

Investigation de divers  
changements

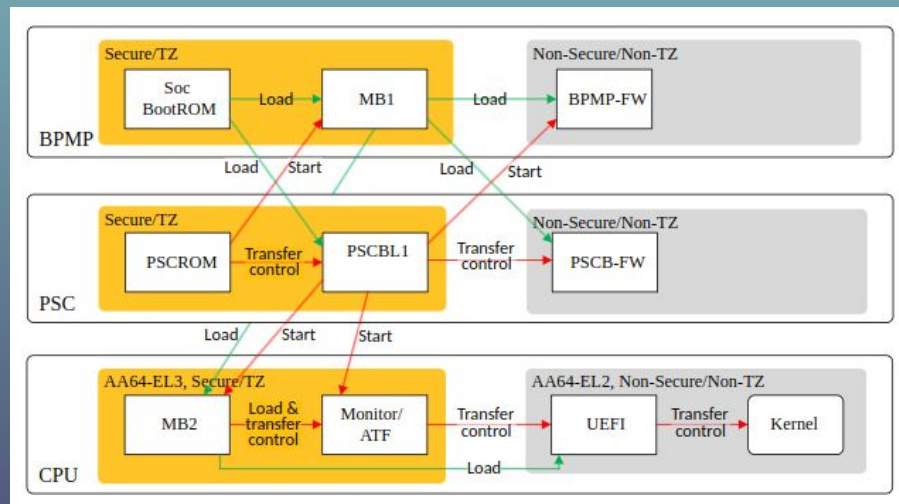
## 04

### **Problèmes**

Problèmes rencontrés  
pendant le projet

# 01

## Jetson Linux Boot Flow



# 02

## Modifications requises

- Kernel DTB
- MB1 Configuration
  - BCT sur le media de boot
  - Construite à partir du dts
- MB2 Configuration
  - Présence ou non d'EEPROM
- Flashing Configuration



# 03 Customisation du système

- Génération d'un rootfs Ubuntu 22.04
  - Autre distribution possible
  - Binaires ARM
  - Copie des composants Jetpack désirés
  - Configuration bootloader valide
- Compilation du Kernel
  - Chaîne de compilation par Nvidia
  - Configuration : defconfig
- Extlinux
  - Kernel
  - Dtb
  - Initrd
  - Arguments de boot

# 04

## PROBLÈMES

---

- Documentation “trop” complète
- Changement de version entre le début et la fin du projet
- Limitations d’extlinux
- Fin du projet anticipée



# Conclusion

- Architecture logicielle & quelques modifications investiguées
- Beaucoup de travail reste à fournir
- SoC Nvidia présentent une alternative intéressante dans les systèmes distribués



# Etude de QEMU pour HEPIA-RISC

## INTRODUCTION

- Cours d'Architecture des Ordinateurs de 1<sup>ère</sup> année
- Architecture RISC
- Source : code source QEMU et architecture similaire
- Préparation à l'implémentation d'un émulateur
  - QEMU vs from scratch

# TABLE OF CONTENTS

**01**

**Architecture**

**HEPIA-RISC**

Instructions, Registres &  
Programmation

**02**

**QEMU**

Représentation d'un CPU  
et d'une machine

**03**

**Tiny Code Generator**

Compilateur JIT et  
implémentation des  
instructions

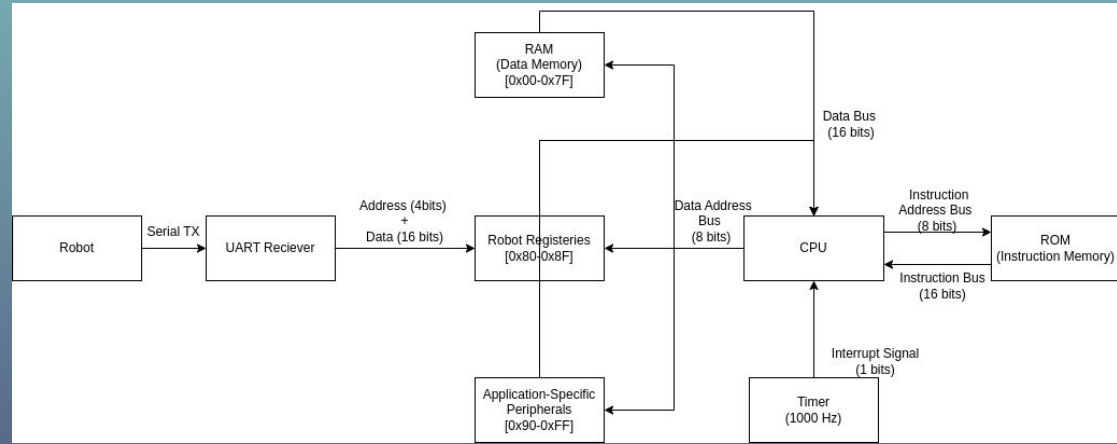
**04**

**Etude de viabilité**

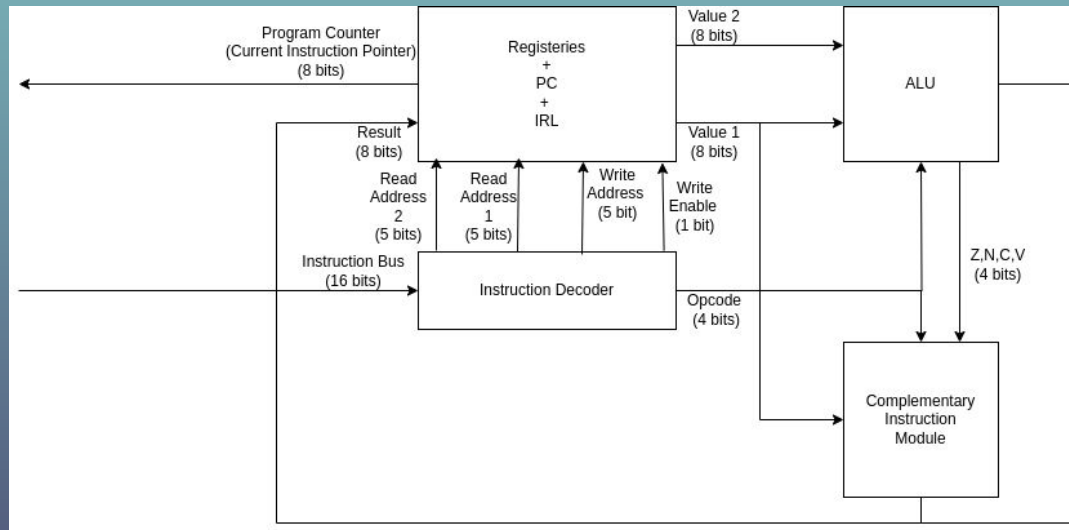
Valeur ajoutée et  
potentiels problème de  
l'intégration

## 01

## Architecture HEPIA-RISC (1/2)



# 01 Architecture HEPIA-RISC (2/2)



# 02

## QEMU

- Implémentation exemple : avr32
- Emulation d'un CPU et d'une "machine"
- La machine intègre :
  - Un microcontrôleur = CPU+ROM
  - Des périphériques (RAM, UART, ...)
- Le CPU
  - Possède des registres internes
  - Emule un PC

# 03

## Tiny Code Generator (I/2)

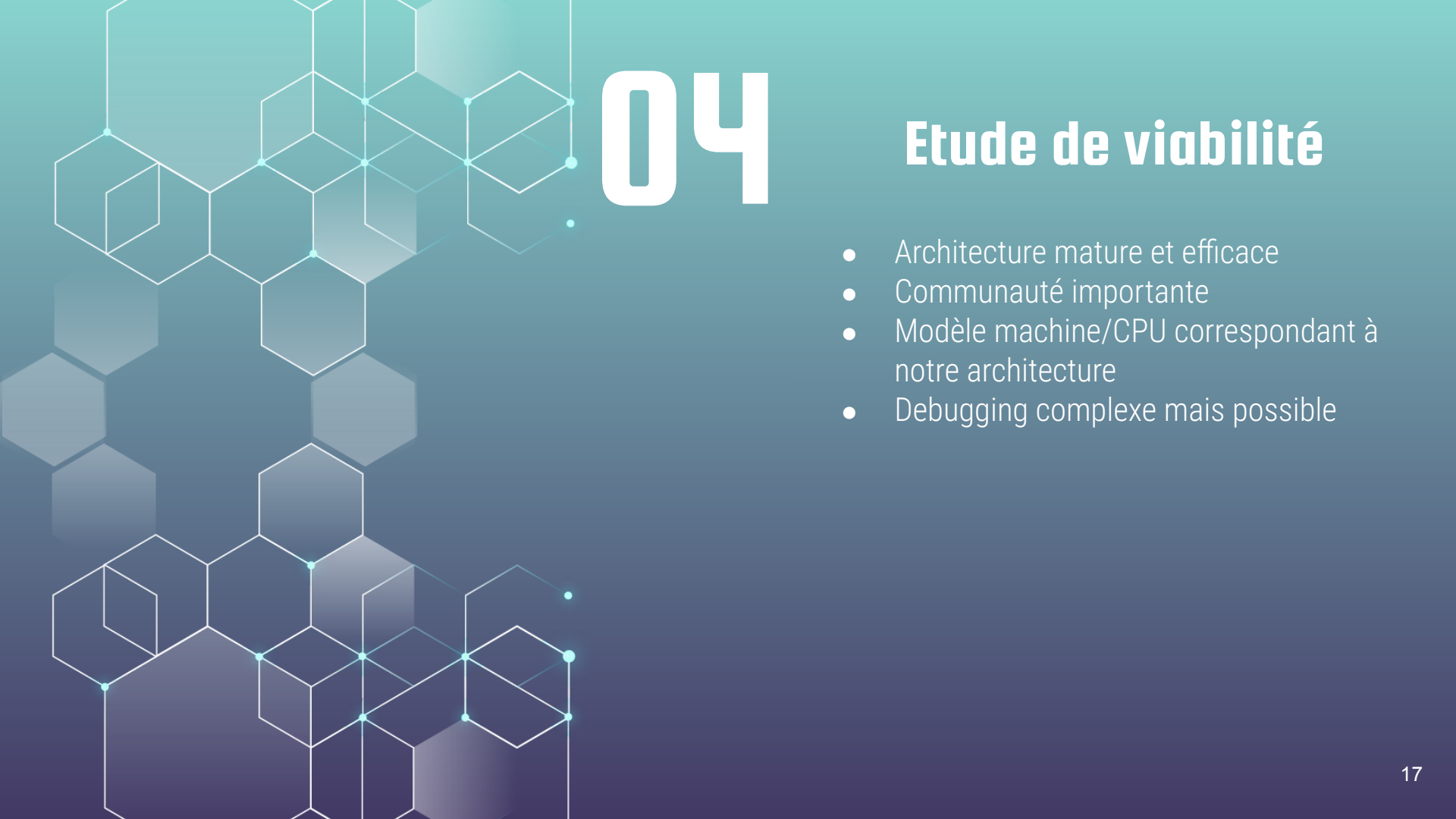
- Partie de QEMU responsable de l'émulation
- Génération d'une représentation intermédiaire
  - Pattern matching
  - Généré instruction par instruction à l'aide des handlers
  - Impossibilité de manipuler les registres internes, frontend fourni par TCG

# 03

## Tiny Code Generator (2/2)

- Block de traduction
  - Prend fin lorsqu'une instruction "branch" ou "jump" est rencontrée
  - S'exécute en une fois sur l'architecture hôte à travers une compilation "JIT"
  - Mets à jour les registres du CPU émulé
- Fonctions "Helpers"
  - Remplacent une instruction trop complexe pour être traduite directement
  - Fonction C





# 04

## Etude de viabilité

- Architecture mature et efficace
- Communauté importante
- Modèle machine/CPU correspondant à notre architecture
- Debugging complexe mais possible

# Conclusion

- L'émulation via QEMU est décidée
- Exemple concret proche de notre architecture
- L'approche est prometteuse
- L'architecture peut encore évoluer



# THANKS

Do you have any questions?  
dylan.frei@etu.hesge.com  
+41 76 432 79 05  
HEPIA

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, infographics & images by Freepik