Chapitre 00 : Introduction

Du NAND au Tetris — Construire un ordinateur de zéro

🎯 Accroche : Le Mystère de l'Ordinateur

Que se passe-t-il quand vous tapez une lettre sur votre clavier ?

1. Vos doigts appuient sur une touche physique
2. Un signal électrique est envoyé
3. Ce signal est transformé en code numérique
4. Le processeur le détecte et l'interprète
5. Un programme décide quoi faire
6. Des pixels s'allument sur votre écran

Entre votre doigt et le pixel

Le Problème de la "Boîte Noire"

L'ordinateur est une boîte noire. Nous tapons, et la magie opère.

Combien de développeurs savent vraiment :

- Comment le processeur exécute leur code ?
- Pourquoi certaines opérations sont rapides et d'autres lentes ?
- Ce qui se passe quand on écrit `x = 5` ?
- Comment une image apparaît à l'écran ?

> ⚠️ **Le problème**
>
> Impossible d'optimiser ce qu'on ne comprend pas

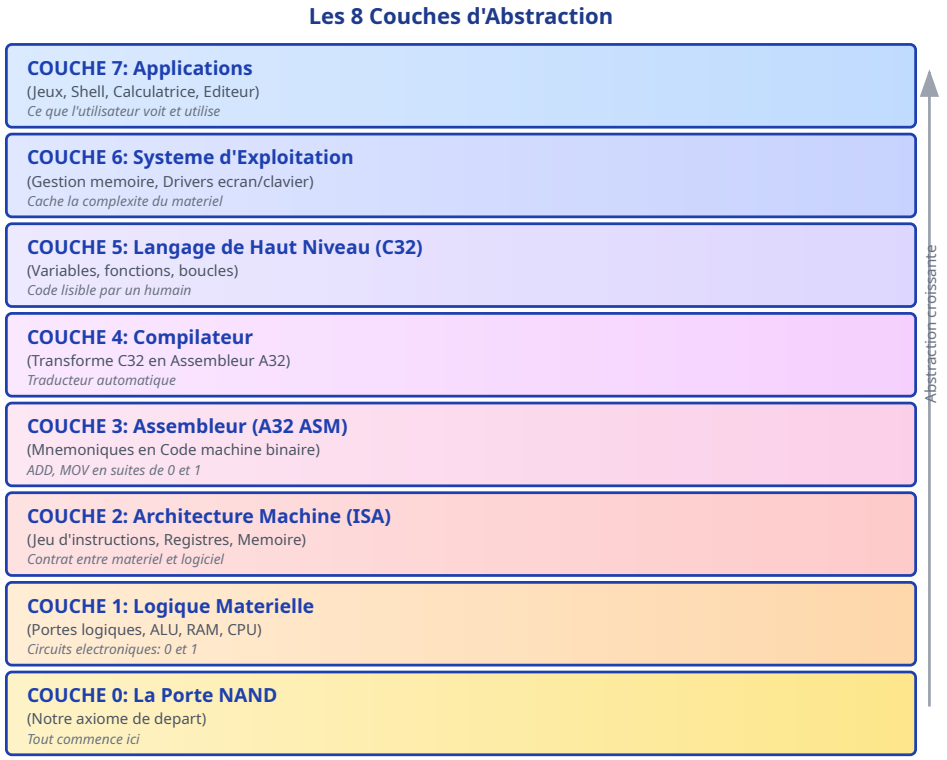
Notre Mission : Briser l'Abstraction

Nous allons descendre au niveau le plus bas — **la porte logique** — et remonter couche par couche.

À la fin, quand vous verrez du code s'exécuter, vous saurez **exactement** ce qui se passe.

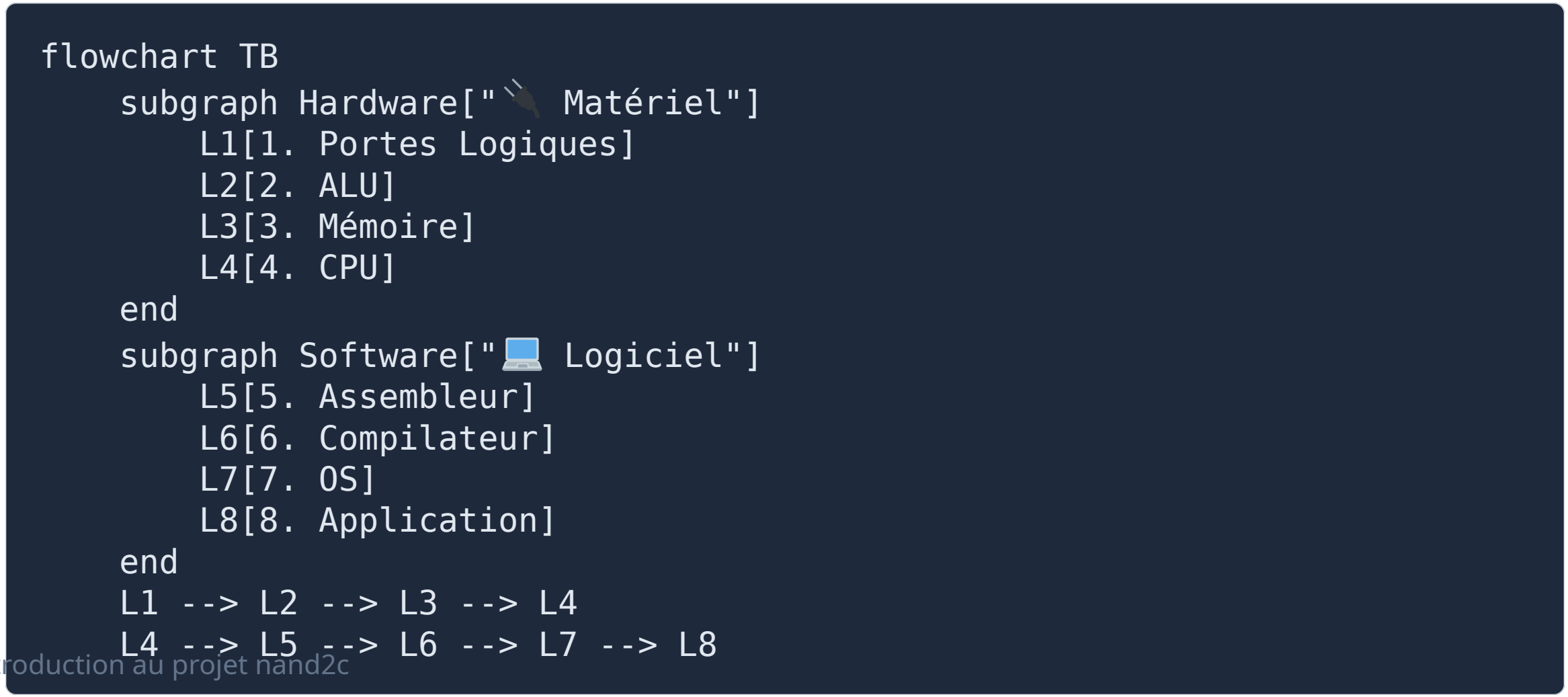
> **Philosophie du cours**
>
> Ce n'est plus de la magie — c'est de l'ingénierie que vous maîtrisez.

Les 8 Couches d'Abstraction



Du transistor à l'application — 8 niveaux d'abstraction

Détail des Couches



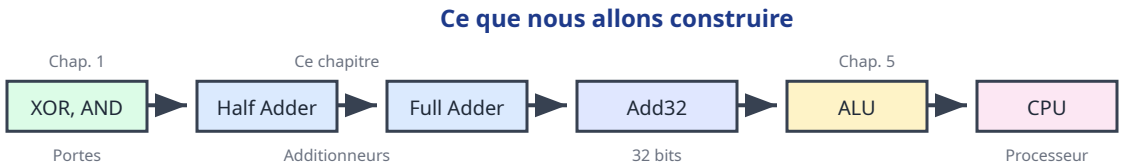
La Beauté de l'Abstraction

Chaque couche a une propriété remarquable :

Elle n'a besoin de connaître que la couche juste en dessous.

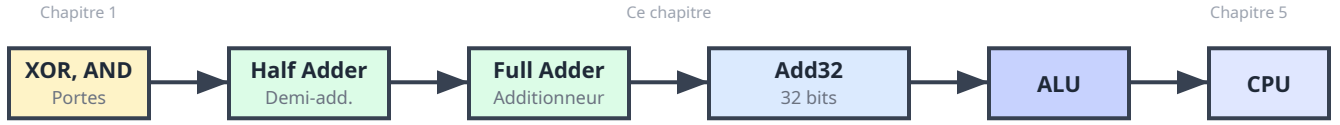
- Le programmeur C32 n'a pas besoin de savoir comment fonctionne l'ALU
- L'ALU n'a pas besoin de savoir qu'elle sera utilisée pour un jeu
- La porte NAND ne "sait" pas qu'elle fait partie d'un ordinateur
- Cette ignorance permet la **complexité maîtrisée**

Notre Feuille de Route



Progression du projet — du NAND au Tetris

Du NAND au CPU



Du plus simple au plus complexe : chaque composant construit sur le precedent

Comment une simple porte devient un processeur complet

L'Architecture nand2c A32

Notre ordinateur s'inspire des architectures **ARM modernes** :

Caractéristique	Hack (Original)	nand2c (Ce projet)
Architecture	16-bits	**32-bits**
Registres	2 (A et D)	**16 (R0-R15)**
Mémoire	Séparée	**Unifiée**
Instructions	Propriétaire	**RISC moderne**

Pont avec ARM

> 🔗 **ARM**
>
> Les processeurs ARM Cortex de vos smartphones utilisent aussi 16 registres (R0-R15) et une architecture RISC.

Comprendre nand2c, c'est comprendre ARM.

Les mêmes concepts s'appliquent :

- Load/Store architecture
- Registres généraux
- Flags (N, Z, C, V)
- Pipeline d'exécution

Ce que Vous Allez Apprendre

Au niveau matériel :

- Construire des portes logiques à partir de NAND
- Comment un additionneur transforme des bits en nombres
- Comment la mémoire "se souvient" des données
- Comment le CPU orchestre tout cela

Au niveau logiciel :

- Comment l'assembleur traduit en binaire
- Comment un compilateur transforme du code
- Comment un OS simplifie l'accès au matériel

Objectifs Détaillés par Chapitre

Ch.	Objectif Principal	Livrable
01	Construire toutes les portes depuis NAND	15 composants HDL
02	Créer l'ALU complète	Additionneur + ALU
03	Implémenter la mémoire	Registres + RAM
04	Comprendre l'architecture	Jeu d'instructions
05	Assembler le CPU	Processeur complet
06	Programmer en assembleur	Programme fonctionnel

Vos Outils

Outil	Rôle
`hdl_cli`	Simule vos circuits HDL
`a32_cli`	Assemble le code A32 → binaire
`c32_cli`	Compile le code C32 → assembleur
Simulateur Web	Interface visuelle pour tout
CPU Visualizer	Voir le CPU en action

Installation

```
cd seed && cargo build --release
```

Le Simulateur Web

Pour une expérience **visuelle et interactive** :

- Écrire et tester votre HDL dans le navigateur
- Voir l'état des signaux en temps réel
- Compiler et exécuter du code C et Assembleur
- Visualiser l'écran, les registres et la mémoire
- Mode pas-à-pas pour le débogage
- Export/Import de fichiers

```
cd web && npm run dev
# → http://localhost:5173
```

Le CPU Visualizer

Outil pédagogique pour comprendre le processeur :

- **Pipeline** : Les 5 étapes d'exécution s'illuminent
- **Registres** : R0-R15 avec mise en évidence des modifications
- **Flags** : N, Z, C, V animés
- **Cache** : Statistiques hits/misses

👉 Accessible via `/visualizer.html`

Plan du Cours

Chapitre	Sujet	Couche	Durée estimée
01	Logique Booléenne	Portes	2h
02	Arithmétique	ALU	2h
03	Mémoire	RAM, Registres	2h
04	Architecture	Structure CPU	2h
05	CPU	Implémentation	3h
06	Assembleur	Programmation	2h

Comment Réussir

1. **Lisez chaque chapitre en entier**

   Avant les exercices — comprenez le contexte

2. **Faites les exercices dans l'ordre**

   Chaque exercice prépare le suivant

3. **Ne regardez pas les solutions**

   Avant d'avoir vraiment essayé (30 min minimum)

Conseils Pratiques

Utilisez le simulateur

La visualisation aide énormément à comprendre

Reliez à l'ensemble

Demandez-vous toujours : "où cela s'insère-t-il ?"

Piège classique

Ne pas comprendre l'abstraction = être bloqué aux chapitres suivants

La Grande Aventure Commence

Vous êtes sur le point d'entreprendre un voyage fascinant.

Quand vous aurez terminé, vous regarderez votre ordinateur différemment.

Promesse du cours

Ce ne sera plus une boîte noire mystérieuse, mais une symphonie d'abstractions que vous pouvez comprendre, modifier, et reconstruire.

Prêt ? Passons à la première brique : la logique booléenne.

Questions ?

📚 **Références :**

- Livre Seed, Chapitre 00 - Introduction
- Simulateur Web : `npm run dev`

👉 **Prochain chapitre :** Logique Booléenne