24 Principes de fonctionnemment des ordinateurs: Architecture, notions d'assembleur.

Prérequis : composants constitutifs, chemins de donnée, ...

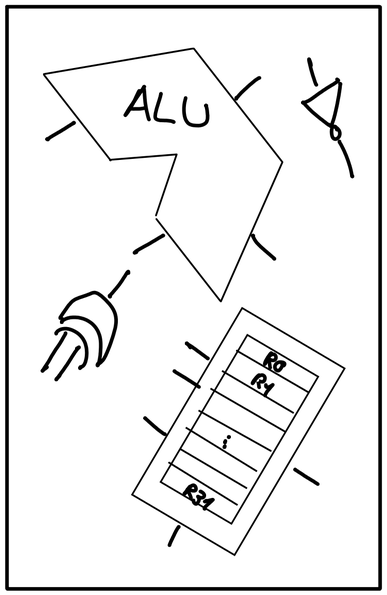
Motivation Un aperçu du fonctionnement d'un processeur est essentiel pour en saisir les limites, et pouvoir produire des programmes adaptés en conséquence.

(I) Architecture (interne, ou micro-architecture)

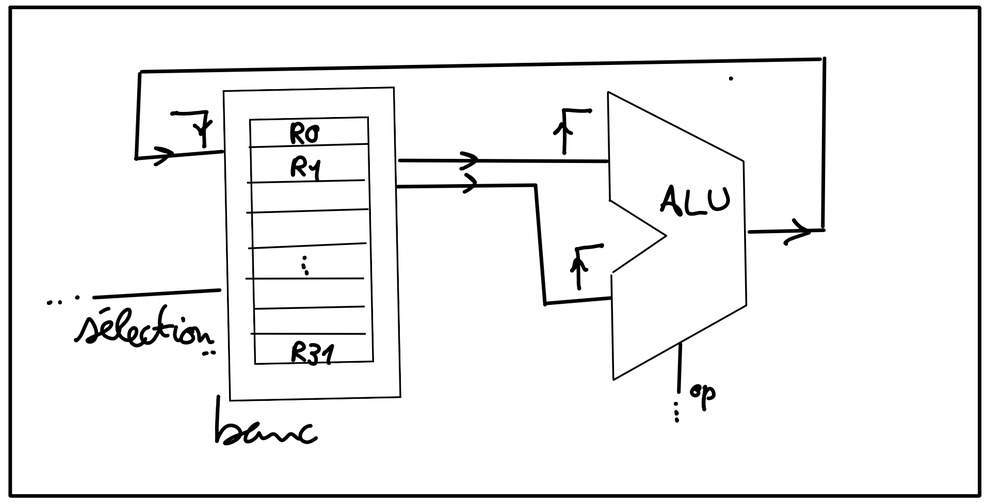
1. Composants

On dispose d'un certain nombre de composants réalisant une variété d'opérations :
- enregistrement d'état (mémoire, banc de registres, ...) ;
- arithmétique (ALU) ;
- traitement/génération signal (horloge)

Il s'agit maintenant de rassembler ces composants pour obtenir un système qui évolue étape par étape en exécutant une tâche programmée.



observation clé : les composants agissent différemment à des moments donnés d'un cycle d'horloge. ex : signal en créneau, ALU avec input en ↑ et output en ↓

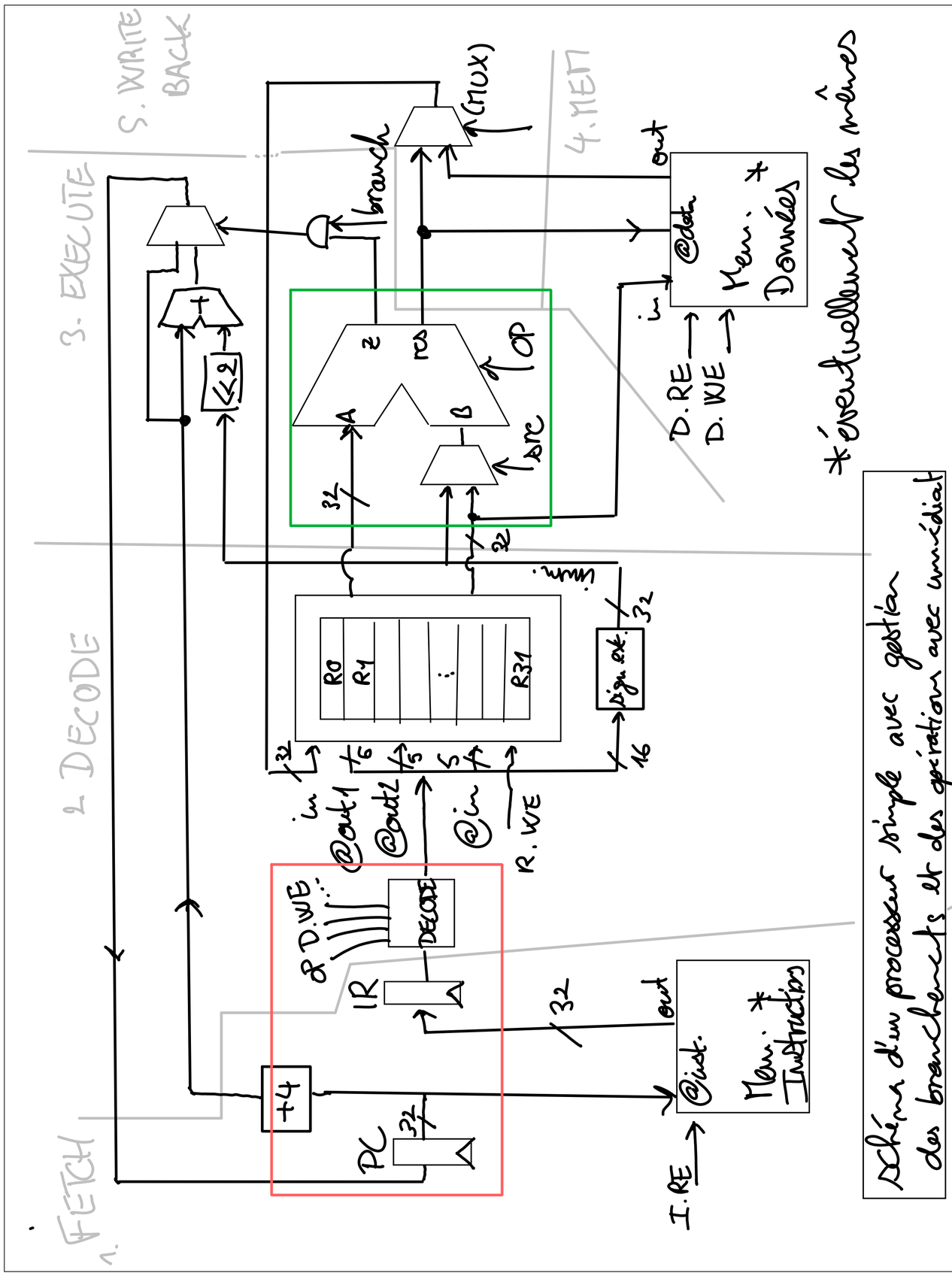


Montage simplifié de calcul sur des registres.

Cela permet de passer d'un état stable au suivant, en évitant les bouclages ; c'est la base de l'interconnexion entre les composants.

2. Agencement monocycle

On présente ci-après une micro-architecture simple mettant en évidence les modules essentiels de la mémoire, avec leurs descriptions.



Décrivons séparément les modules d'intérêt:

- **Instructions** récupère les instructions, les décode (activation en conséquences des signaux de lecture/écriture/opération ALU...) (cf. section suivante);
- **ALU**, réalise les calculs en adéquation avec les signaux perçus du décodeur.
- la mémoire est accessible en lecture et en écriture OU on parle de machine de Harvard lorsqu'on distingue mémoire des instructions et de données, sinon (le plus courant), on parle de machine de Von Neumann.
- le banc de registre est accessible en lecture 2 fois et en écriture 1 fois; ici, il est composé de 32 registres, donc adressables par 5 bit.

...

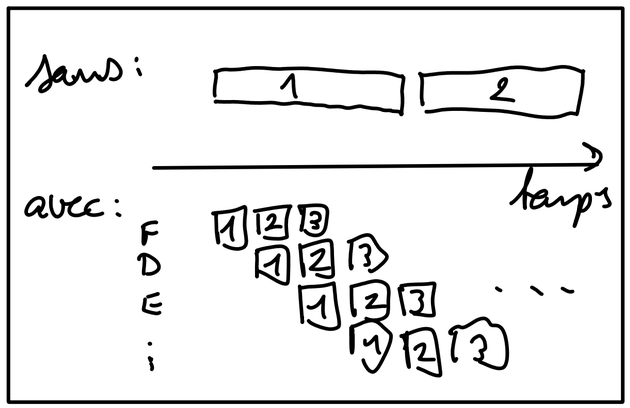
activité Exécution pas à pas sur un logiciel de simulation graphique pour observer les différents chemins de données

DEV 1 version tableau de cette exécution

3. Limitations

Une instruction doit pouvoir être réalisée entièrement avant de passer à la suivante. → chaque instruction prend un seul cycle, mais celui-ci est aussi long que le plus long chemin de donnée possible; ce qui

est très pénalisant.
Une solution est le découpage en modules pour une exécution en pipeline : cela introduit plein de défis mais limite la durée d'un cycle au chemin de donnée maximal pour un seul module.



II Jeu d'instructions (Architecture externe, ISA)

1. Présentation

Est spécifié pour chaque processeur un ensemble d'instructions valides, ainsi qu'une sémantique opérationnelle, qui abstrait la microarchitecture des composants. Par exemple :

	FTT	Opcode	func3 ...	description
add	R	0110011	0x00	rd = rs1 + rs2
xor	R	0110011	0x40	rd = rs1 ^ rs2
lw	I	0000011	0x2	rd = M[rs1 + imm][0:7]
sw	S	0100011	0x2	M[rs1 + imm][0:31] = rs2
...				

Ce tableau, extrait de la spécification de RISC-V, précise la sémantique associée à certaines instructions.
Les opcodes (et leurs ornements func..) sont les valeurs qu'utilise le décodeur pour régler les signaux.

Ces instructions machines peuvent être instruites par le programmeur par l'intermédiaire d'un assembleur ; en correspondance presque 1:1 avec l'ISA. On manipule ici directement les registres, les appels mémoire et les sauts.

le code ci-contre est un exemple de fonction écrite en RISC-V, qui pour une entrée > 0 (dans le registre a0), renvoie le n-ième nombre de Fibonacci.

```
.fibo:
    li a3, 0
    li a2, 1
.LOOP
    add a4, a3, a2
    addi a0, a0, -1
    mv a3, a2
    mv a2, a4
    bnez a0, .LOOP
    mv a0, a1
    ret
```

Fibonacci en RISC-V

activité Associer à des structures connues (en C) des équivalents assembleur, observer les résultats de compilation de fonctions simples, ...

2. Aspects de compilation

On dispose également d'un pointeur de pile. C'est le point de départ de l'exécution par fonctions.

Tous les langages compilés se réécrivent (par déf.) en assembleur. C'est un domaine de recherche entier de savoir compiler : — de manière correcte/sûre
— vers du code efficace

DEV2 Compilation d'expressions arithmétiques.