Principes de fonctionnement des ordinateurs: Architecture, notions d'assembleur

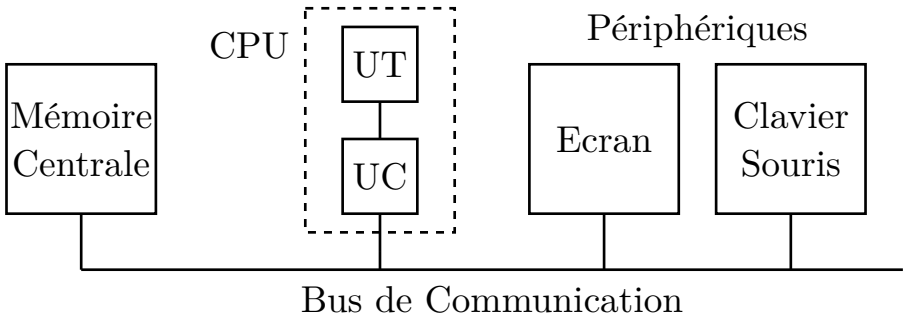
I. Principes de fonctionnement [NSIT] [NSIP]

Définition 1 **Un Programme** est une suite d'instructions.

Définition 2 **Von Neumann** [TOR 23] décrit en 1945 la structure d'un ordinateur qui possède :

- une **mémoire centrale** contenant instructions *et* données
- une **unité de contrôle** qui coordonne l'exécution de l'instruction.
- une **unité de traitement** fait les calculs nécessaires à l'instruction.

Schémas 3 [MOS]

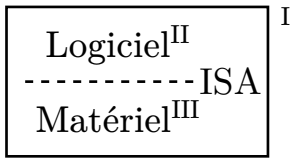


Définition 4 **CPU** (pour *Central Processing Unit* en anglais) est l'ensemble formé par l'unité de contrôle et l'unité de traitement.

Définition 5 **Le compteur ordinal** (PC pour *Program Counter* en anglais) est l'adresse de l'instruction courante que le CPU exécute.

Exemples 6 De nombreux CPU ont été produits au cours de l'histoire, par exemple les CPU **pentium Intel**.

Schéma 7



II. Notions d'assembleur: architecture externe [PAT]

Définition 8 **Le jeu d'instructions** (ISA pour *Instruction Set Architecture* en anglais) est l'interface entre le logiciel et le matériel. Il défini, entre autre, un ensemble d'opérations élémentaires, les instructions machine.

Exemple 9 Les jeux d'instructions RISC-V et ARM sont de la famille **RISC** (*Reduced Instruction Set Computer*). Le jeu d'instructions x86 est de la famille **CISC** (*Complex Instruction Set Computer*).

Remarque 10 **x86 et ARM** sont respectivement dominants pour les processeurs d'ordinateurs de bureau et ceux de *smartphones*.

Définition 11 **Un langage d'assemblage** d'un jeu d'instructions est une représentation lisible pour un humain des instructions machine. L'assembleur est le programme qui traduit le langage d'assemblage en langage machine. L'opération inverse est le désassemblage.

Définition 12 **La compilation** est la transformation d'un programme écrit dans un langage de haut niveau comme C en langage d'assemblage.

Définition 13 **Le langage d'assemblage** manipule trois types de données qui sont les :

- valeurs des **registres**: petites cellules mémoire d'accès rapide
- valeurs stockés dans la **mémoire** centrale
- **immédiats**, des constantes connues à la compilation

Définition 14 **Le format d'instruction** spécifie la manière de décoder une instruction machine en binaire en identifiant différents champs comme le type d'opération ou les opérandes.

Exemple 15 **Le RISC-V** est un langage d'assemblage où les instructions sont codées sur 32 bits et les 7 derniers bits précisent le type d'opération. Différentes opérations utilisent différents formats:

Schema 16 **Formats d'instructions** [PAT Cover]

R	func7	rs2	rs1	f3	rd	opcode
I	imm[11:0]		rs1	f3	rd	opcode
S	imm[11:5]	rs2	rs1	f3	imm[4:0]	opcode
SB	imm [12|10:5]	rs2	rs1	f3	imm [4:1|11]	opcode
U	imm[31:12]				rd	opcode
UJ	imm[20|10:1|11|19:12]				rd	opcode

A. Opérations arithmétiques (R, I)

Exemple 17 **L'instruction** `add x9, x20, x21` signifie en RISC-V que lorsque cette instruction sera exécutée, le registre 9 recevra la somme des valeurs des registres 20 et 21.

Le RISC-V encode cette instruction avec le format de type R :

Champ	funct7	rs1	rs2	funct3	rd	opcode
Taille	7 bits	5 bits	5 bits	3 bits	5 bits	7 bits
Contenu	0	21	20	0	9	51

B. Transfert de données (S, U, I)

Exemple 18 **L'instruction lw** `x9, 32(x22)` permet de charger dans le registre x9 le contenu de l'adresse mémoire (x22 + 32).

Exemple 19 **Stocker une constante** de 32 bits dans un registre se fait à l'aide de deux instructions:

- **lui** (*load upper immediate*) stocke les 20 bits de poids fort
- **addi** (*add immediate*) ajoute les 12 bits de poids faible

C. Branchements (SB)

Exemple 20 **L'instruction beq** `x7, x9, loop` fait avancer le pointeur d'instructions jusqu'au label `loop` si les valeurs de `x7` et `x9` sont égales.

D. Fonctions (UJ)

DEV

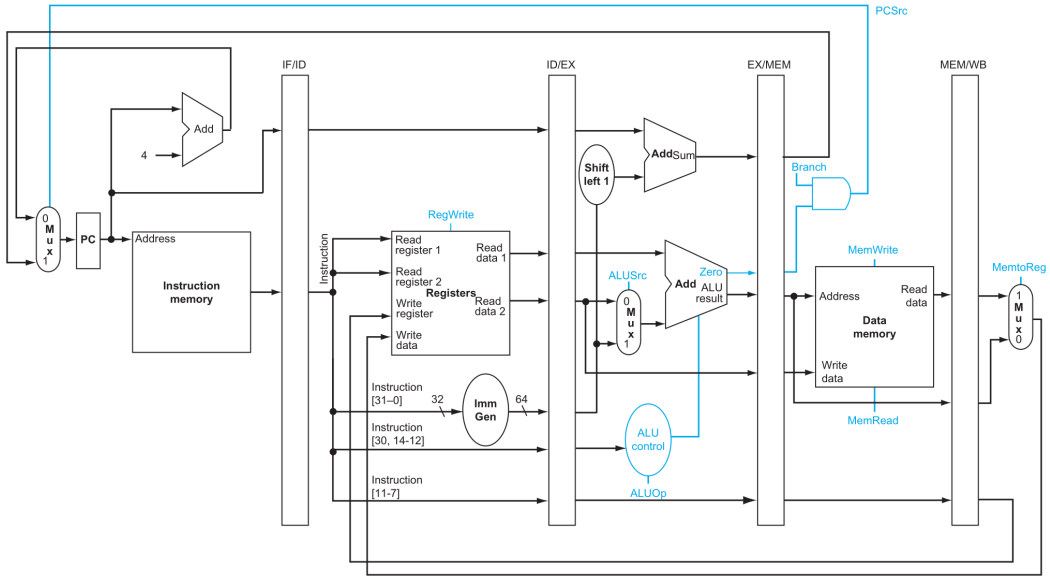
Remarque 21 **Les appels de fonctions** respectent les conventions d'appel et RISC-V en définit sa propre convention d'appel. [PAT Chap.1 p.108 et 113]

Remarque 22 En RISC-V les instructions machine sont codées sur un mot 32 bits mais ce n'est pas le cas en x86 par exemple.

[PAT Chap.2.5 2.8]

III. Micro-Architectures: architecture interne [PAT p.297]

Schéma 23 [PAT Fig4.48 p.312]



A. Circuits Combinatoires et Séquentiels

Définition 24 **L'ALU** (*Arithmethic and Logic Unit*) est le composant qui implémente l'UT. Il effectue des opérations comme l'addition d'entiers ou la conjonction logique. Il est construit à partir de composants combinatoires de base comme les portes logiques ou plus complexes comme les multiplexeurs.

DEV

Remarque 25 **Un additionneur N bits** [PAT p.A-36 à p.A-47] qui enchaîne N addittioneurs 1 bits possède un chemin critique linéaire en n. On peut optimiser ce chemin critique.

Remarque 26 **L'UC** peut être implémentée par un système séquentiel et donc une machine à états telle qu'une Machine de Moore ou de Mealy. On parle alors de micro-architecture multi-cycle.

Performance 27 **Les performances** d'une architecture peuvent être mesurées. On utilise alors différents indicateurs comme le **CPI**

(Cycles Par instruction) ou **MIPS** (Million d'Instructions Par Seconde).

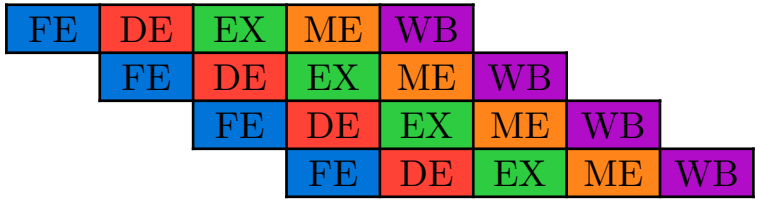
B. Micro-architecture chaînée

Définition 28 **La chaîne de traitement** (*pipeline* en anglais) est une technique d'implémentation de micro-architecture qui sépare les instructions en de multiples étapes qui peuvent s'exécuter en parallèle.

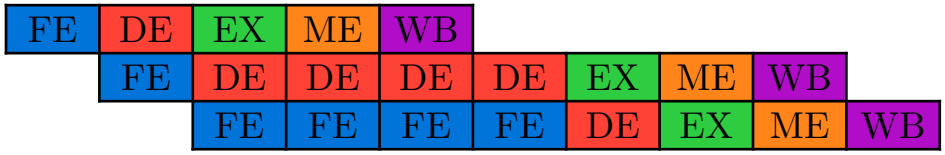
Exemple 29 **Le Pipeline sur 5 étages** [PAT 4.6, p.284] est un exemple classique avec 5 étapes: **Fetch** (récupère l'instruction) **Decode** (lis les opérandes, entre autres) **Execute , Memory et Write-Back**.

Exemple 30

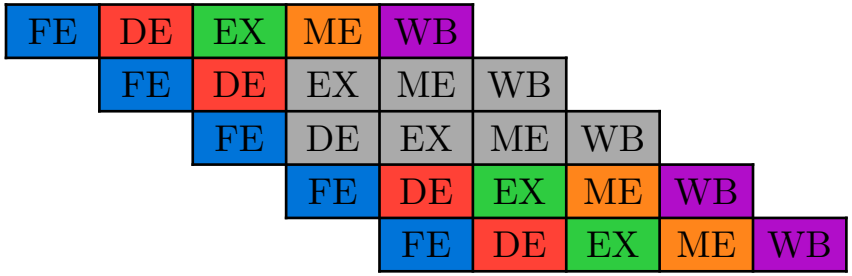
- Le pipeline en fonctionnement optimal



- Dépendance de données entre deux intructions.



- Branchement conditionnel pris



Remarque 31 **Des aléas** structurels, de données ou de contrôle rendent plus complexe le pipeline, ce qui explique la volonté d'utiliser des ISA plus simples (RISC) qui facilitent l'implémentation de ce type d'optimisation.

C. Micro-architectures Parallèles [PAT 6]

Définition 32 **La Classification de Flynn** [PAT 6.3 Fig 6.2] est une typologie des systèmes en fonction du nombre de donnnées et d'instructions simultanées.

		Données	
		Une donnée	Multiples données
Instructions	Une	SISD Ex: Intel Pentium 4	SIMD Ex: Instructions SSE du x86
	Multiples	MISD Pas d'exemple connu	MIMD Ex: Intel Core i7

Définition 33 **Le parallélisme** signifie que plusieurs données sont traitées en parallèle. Les architectures SIMD et MIMD sont donc des architectures parallèles.

Exemple 34 **SSE** (Streaming SIMD Instruction) est l'extension de x86 avec des opérations **vectorisées**. Elles sont l'application d'une instruction simple sur une collection de données.

Définition 35 **La Classification de Raina** [VIA p.10] [PAT 6.5] décrit une typologie plus précise des systèmes MIMD.

Accronyme	Catégorie	Exemple
SASM	Single Adress space, Shared Memory	Architectures multicœur
DADM	Distributed Address Space, Distributed Memory	Architectures distribuées
SADM	Single Address Space, Distributed Memory	Supercalculateurs

Principes de fonctionnement des ordinateurs: Architecture, notions d'assembleur

I. Principes de fonctionnement [NSIT] [NSIP]

1 Def Un Programme
2 Def Von Neumann [TOR 23]
3 Schémas [MOS]
4 Def CPU
5 Def Le compteur ordinal
6 Exemples
7 Schéma

II. Notions d'assembleur: architecture externe [PAT]

8 Def Le jeu d'instructions
9 Ex
10 Rem x86 et ARM
11 Def Un langage d'assemblage
12 Def La compilation
13 Def Le langage d'assemblage
14 Def Le format d'instruction
15 Ex Le RISC-V
16 Schema Formats d'instructions [PAT Cover]

III. Micro-Architectures: architecture interne [PAT p.297]

23 Schéma [PAT Fig4.48 p.312]

A. Opérations arithmétiques (R, I)

17 Ex L'instruction add

B. Transfert de données (S, U, I)

18 Ex L'instruction lw
19 Ex Stocker une constante

C. Branchements (SB)

20 Ex L'instruction beq

D. Fonctions (UJ)

21 Rem Les appels de fonctions
22 Rem

A. Circuits Combinatoires et Séquentiels

24 Def L'ALU
25 Rem Un additionneur N bits [PAT p.A-36 à p.A-47]
26 Rem L'UC
27 Performance Les performances

B. Micro-architecture chaînée

28 Def La chaîne de traitement
29 Ex Le Pipeline sur 5 étages [PAT 4.6, p.284]
30 Ex
31 Rem Des aléas

C. Micro-architectures Parallèles [PAT 6]

32 Def La Classification de Flynn [PAT 6.3 Fig 6.2]
33 Def Le parallélisme
34 Ex SSE
35 Def La Classification de Raina [VIA p.10] [PAT 6.5]

Commentaires :

- Figure en III. de plan de Yaelle. [PAT]
- Reprise du plan de Maxime Bridoux pour la partie III.
- Partie I volontairement courte avec schémas explicatif pour gagner de la place pour le grand schémas partie III.
- Partie II plutôt longue pour bien présenter le RISC-V et les différents types d'instruction. L'assembleur est important dans la leçon car « Architecture » en anglais signifie principalement l'ISA (et pas la micro architecture).

Autres développements :

- Pipeline à 5 étages
- Décode et exec d'une instruction avec le schémas partie III.
- Construction shémas partie III. : « Un processeur simple » sur le site interne. [VAH p.426]

Bibliographie

[NSIT] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen, *Numériques et Sciences Informatiques Terminale.*

[NSIP] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen, *Numériques et Sciences Informatiques 1er.*

[TOR] T. Balabonski & S. Conchon & J. Filliâtre & K. Nguyen & L. Sartre, *MP2I MPI, Informatique Cours et exercices corrigés.*

[MOS] Tanenbaum & Bos, *Modern Operating System, 5th Edition.*

[PAT] D. A. Patterson & J. L. Henessi, *Computer Organisation and Design, RISC-V Edition.*

[VIA] K. T. Vianney, *Solutions parallèles efficaces sur le modèle CGM d'une classe de problèmes issus de la programmation dynamique.*

[VAH] F. Vahid, *Digital Design.*

Paul Adam - Santiago Sara Bautista