

CHAPITRE 1: rappels sur les processeurs.

1/ Introduction et objectif du cours:

L'évolution des architectures d'ordinateurs répond aux besoins croissants en CPU et en temps de calcul pour: les simulations scientifiques, prévisions météorologiques, modélisation du climat, le traitement d'images et les bases de données (big data).

Objectif: Comprendre les facteurs influençant les performances de processeurs.

2/ Architecture d'un système informatique:

Un système informatique regroupe: moyens matériels (processeur, mémoire, périphériques), moyens logiciels (système d'exploitation, compilateurs).

Il sert à répondre aux besoins des utilisateurs pour les traitements automatiques.

2.1/ Architecture Interne d'un processeur:

Processeur: unité intelligente qui sert à organiser, décoder, et exécuter les instructions d'un programme.

Unités principales:

Unité de commandes: coordonne et séquencie l'exécution des instructions d'un programme principale composante.

- compteur d'ordinal (PC): contient l'adresse de la prochaine instruction à exécuter.

- Register d'instruction (RI): stocke temporairement l'instruction en cours.

- Décodeur d'instruction: décode l'instruction et génère les signaux logiques.

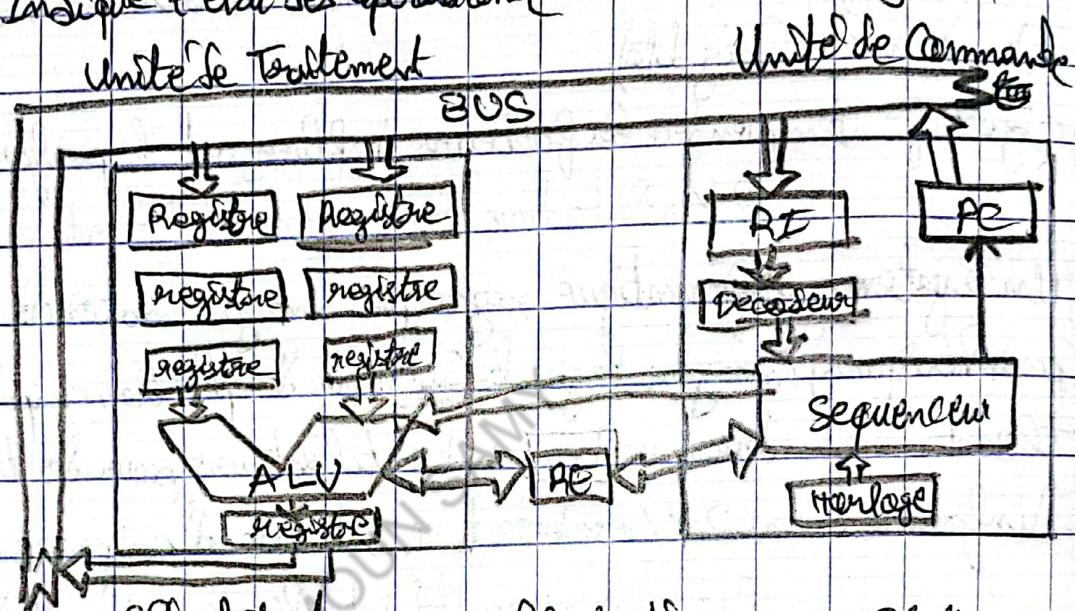
- Bloc logique de Commande (Séquenceur): gère la synchronisation et génère les signaux de commande internes/externes

• Unité de traitement: Effectue les opérations arithmétiques et logiques.
Principaux composants:

Accumulateur: Stockent les opérandes et les résultats des opérations (Register de travail)

Unité Arithmétique et Logique (UAL): Exécute les opérations et réalise les comparaisons.

Register d'état (Flag): Indique l'état des opérations (indicateurs comme zéro, négatif, retenue).



L'architecture simplifiée d'un processeur.

2.2 / Architecture externe d'un processeur:

Ce que doit connaître le programmeur:

Langage machine: Instructions en binaire propre au processeur.

Langage assembleur: équivalent au langage machine c'est une représentation mnémotechnique des instructions.

Principaux composants

Register Direct: Manipulable directement par les instructions.

Adressage mémoire: Modes d'accès aux données en mémoire (direct, indirect, ...)

Le jeu d'instruction: le format et le nombre d'instructions.

Les mécanismes de traitement des interruptions et des exceptions: lignes et priorité d'interruption, localisation du gestionnaire d'interruption

Exemple d'un processeur:

Structure générale: composé de l'unité de commande et de l'unité de traitement. Comportent des composants essentiels comme :

Registers: petites mémoires rapides pour stocker temporairement les données en cours de traitement.

Bus: circuit qui transmettent les données, les adresses et les commandes entre les composants.

- bus d'adresse: identifie les emplacements mémoire
- bus de données: transporte les données
- bus de contrôle: gère les signaux de lecture/écriture.

3/ Organisation mémoire:

Dépend de la taille des bus d'adresse et bus de données, qui influencent la puissance du processeur.

exemple: processeur 8 bits → bus de données 8 bits; 32 bits → 32 bits etc...

Caractéristiques clés :

Capacité: Quantité d'informations stockées, exprimée en octets ou moëts.

Temps d'accès: Durée pour recuperer une donnée, mesurée en nanosecondes (10^{-9} s) pour les mémoires rapides comme la RAM, ROM ou registres.

4/ Modes d'adressage de la mémoire:

Immédiat: L'oprande est directement spécifié dans l'instruction.

Direct: L'adresse mémoire est directement donnée.

Indirect: Une adresse contient l'adresse réelle des données.

par registre: Ses données sont dans un registre interne

Indirect par registre: L'adresse est calculée à partir d'un registre.

Basé: Combinaison d'un registre de base et d'un déplacement.

Relatif: L'adresse dépend du compteur de programme (PC)

(3)

5- Cycles d'exécution d'un programme :

Recherche (fetch): Lecture de l'instruction depuis la mémoire

Décodage (decode): Interprétation de l'instruction

Exécution (execute): Réalisation de l'opération

6- Indicateurs de performance des processeurs:

Temps d'exécution d'un programme : $T_{exec} = N_{inst} \times CPI \times T_{cycle}$

avec : N_{inst} : Nombre Total d'instructions.

CPI: Cycle par instruction

Tcycle : Temps d'un cycle d'héritage.

optimisation: - Réduire Tcycle.

- Réduire le nombre d'instructions ou de cycles par instruction (CISC vs RISI)

7- Amélioration des performances:

principales: Parallelisation: Exécution simultanée de plusieurs instructions (pipeline, multi-coeur). - Héritage mémoire: Optimisation des accès (cache, RAM). - optimisation des instructions: Simplification (RISI) ou enrichissement (CISC).

8- Fréquence d'héritage:

Définition: Vitesse à laquelle le processeur exécute les instructions.

$T_{cycle} = \frac{1}{f}$: plus la fréquence est élevée plus le processeur est rapide (Temps de cycle \rightarrow).

limites: - Dissipation de chaleur accrue

- Temps de propagation des signaux limitant la vitesse.

REMARQUE: Pour rendre les processeurs plus rapides, on peut augmenter la fréquence d'héritage (limite et courbe) ou introduire du parallelisme pour exécuter plusieurs instructions simultanément.

(4)

CHAPITRE 2 : Le pipeline

1/ Machine sans pipeline:

Un processeur conventionnel exécute les instructions une par une, séquentiellement. Chaque instruction passe par 3 étapes :

Recherche (fetch) : Recherche (Recuperer) l'instruction en mémoire.

Décodage (decode) : Identifier l'opération à réaliser

Exécution (execute) : Réalisation de l'opération.

Limites : à chaque étape, seuls certains composants du processeur sont utilisés, tandis que d'autres restent inactifs, ce qui réduit l'efficacité. Le temps total d'exécution est élevé, car chaque instruction attend la fin de l'autre.

2/ principe du pipeline:

Inspiré des chaînes industrielles d'assemblage, où plusieurs étapes sont effectuées en parallèle. concept : Recouper les instructions en sous-parties (étapes) et permettre leur traitement simultané.

Exemple : Dans un restaurant universitaire, une seule personne dans la chaîne (Séquentiel) est lente, mais plusieurs personnes passant en parallèle (pipeline) améliorent l'efficacité. Temps total avec pipeline : $T_p = (m+n-1) \times T_e$, où m est le nombre de personnes, n le nombre d'étapes, et T_e le temps par étape.

3/ technique du pipeline:

Fonctionnement : Le pipeline est divisé en étages correspondant aux étapes de traitement (fetch, decode, execute).

- Plusieurs instructions sont en cours simultanément, chacune d'un étage différent.

- Des registres intermédiaires stockent les résultats entre les étages pour synchronisation.



* pipeline à 5 étages (MIPS):

- IF (Instruction Fetch): Lecture de l'instruction
- ID (Instruction decode): Décodage et lecture des opérandes
- EX (Execution): Exécution de l'opération ou calcul d'un membre
- MEM (Memory): Accès au contenu en mémoire.
- WB (Write Back): Ecriture du résultat dans un registre

4/ Calcul de performances d'un pipeline:

Temps Total d'exécution:

$$\rightarrow \text{sans pipeline: } T_f = N \times R \times T_e \quad (\text{Toutes les instructions sont séquentielles})$$

$$\rightarrow \text{avec pipeline: } T_f = T_p + (N-1) \times T_e \text{ soit } T_f = T_e \times (R + N - 1)$$

Latence: Temps pour exécuter une instruction du début à la fin.

Débit: Nombre d'instructions exécutées par unité de temps: $D = \frac{1}{T_f}$

$$\text{Accélération théorique: } A = \frac{N \times R \times T_c}{(R + N - 1) T_e}$$

5/ Profondeur du pipeline:

Une architecture pipeline plus profonde (plus d'étages) permet d'exécuter plus d'instructions en parallèle.

avantages: Débit accru. Inconvénients: Gestion des alias et augmentation de la latence.

6/ Problèmes du pipeline (Alios):

① Alios structuraux: conflits d'accès aux ressources partagées, par exemple 2 instructions accédant à la même mémoire.

Solutions: Multiplier les ressources ou utiliser des suspensions.

Exemple: pendant l'exécution d'une instruction, 2 étapes du pipeline peuvent nécessiter un accès à la Mémoire au même temps.

- phase IF (Fetch): une instruction est récupérée en mémoire pour être décodée.

- phase MEM (Memory access): une autre instruction, déjà en cours d'exécution effectue un accès Mem (L/E de données)

L1 load R1, 0 (R2)
 L2 add R3, R1, R4

INST 1	IF	ID	EXE	MEM	WB
	IF	ID	EXE	MEM	WB
	IF	ID	EXE	MEM	WB
	IF	ID	EXE	MEM	WB

Resolution

IF	ID	EXE	MEM	WB
IF	ID	EXE	MEM	WB
IF	ID	EXE	MEM	WB
IF	ID	EXE	MEM	WB

② Alias de données: Une instruction dépend d'une donnée produite par une instruction précédente.

Types: RAW (Read After Write): Lecture avant que l'écriture soit terminée.

WAR (Write After Read): Ecriture dans une destination encore utilisée en lecture.

WAW (Write After Write): Deux écriture sur la même destination, (register).

Solutions: Performance des instructions.

forwarding (bypassing): Transmettre directement les résultats intermédiaires

Exemple: ADD R1, R2, R3 | SUB R4, R1, R5

RI	DI	RD	EI	ER
RI	DI	RD	EI	ER

Suspension

RI	DI	RD	EI	ER
RI	DI		RD	EI

→ Forwarding [RI DI RD EI ER]

(R1)

RI	DI	RD	EI	ER
R1	DI		RD	EI

Forwarding by bypass.

RAW:

L1: R1 = R2 + R3

IF	ID	EXE	MEM	WB
	IF	ID	EXE	MEM

L2: R4 = R1 + R5

IF	ID	EXE	MEM	WB
	IF	ID	EXE	MEM

écrasé de l'ire R1 avant qu'elle soit écrite dans R4 Pre registre

WAR: Une instruction escale de modification donnee qui une autre instruction précédente est l'entraîne de l'ire

L1: R1 = R2 + R3

IF	ID	EXE	MEM	WB
	IF	ID	EXE	MEM

L2: R2 = R4 + R5

IF	ID	EXE	MEM	WB
	IF	ID	EXE	MEM

elle modifie R4 bâtonnée de R5.

WAW:

L1: R1 = R2 + R3

IF	ID	EXE	MEM	WB(R1)
	IF	ID	EXE	MEM

L2: R1 = R4 + R5

IF	ID	EXE	MEM	WB(R1)
	IF	ID	EXE	MEM

③

⑤ Aléas de Contrôle : Causes par des instructions

de branchements ou des tests conditionnels. Le processeur ne sait pas quelle instruction ~~entre~~ doit exécuter avant d'évaluer la condition.

Solution : - Prediction de branchements : Deviner l'instruction suivante.

Suspension : pipeline stoppé jusqu'à ce que le branchements soit évalué.

7 Méthodes de résolution des aléas de contrôle :

Aléas structurels : gérer les ressources (ex : membre distincts pour instructions et données).

Aléas de données :

Forwarding : Transmettre directement les résultats calculés.

Reordonnancement : Modifier l'ordre des instructions pour éviter les conflits.

Aléas de contrôle :

- Prédiction dynamique des branchements.

- Nettoyage du pipeline en cas d'erreur de prédiction.

(4)

CHAPITRE 3 : Hiérarchie Mémoire

1/ Introduction à la mémoire :

Réfinition: La mémoire est un dispositif qui permet d'enregistrer, conserver et restituer des informations.

Types d'accès :

Séquentiel: Accès fait nécessitant de parcourir toutes les informations précédentes (ex: bandes magnétiques).

Direct: Chaque information a une adresse propre, permettant un accès immédiat (ex: mémoire centrale).

Semi-Séquentiel: Combinaison des deux (ex: disque dur). ^{direct} + ^{séquentiel}

Associatif: Les informations sont identifiées et accessibles par une clé (ex: mémoire cache).

2/ Classification des mémoires :

① **Mémoire vive (RAM):** Volatile, elle nécessite une alimentation électrique pour conserver les données.

Types statiques :

S-RAM: Rapide, chère, faible capacité (utilisée pour les caches)

Types dynamiques :

D-RAM: Plus lente, moins chère, utilisée pour la mémoire centrale.

SDRAM: Mémoire dynamique synchronisée avec l'horloge du processeur, permettant des accès ardennés et rapides.

RDRAM: Mémoire dynamique à haut débit, conçue pour des performances élevées mais nécessitant des contrôleur(s) ~~station~~ spécifiques.

② Mémoire morte (ROM): Non volatile, conserve les données même sans alimentation.

Types :

ROM: Mémoire programmée de façon irreversible lors de la fabrication, utilisée pour stocker des microprogrammes.

PROM: Mémoire programmable une seule fois après fabrication, pour des applications fixes.

EPROM: Mémoire programmable et effaçable par rayon UV, réutilisable après chaque effacement.

EEPROM: Mémoire programmable et effaçable électriquement, plus pratique et rapide que l'EPROM.

Flash: Mémoire de type EEPROM, effaçable et programmable par blocs, utilisée dans les clés USB et SSD.

3 / Evolution des performances :

Problème: Les processeurs évoluent beaucoup plus vite que les mémoires, créant un écart de performances.

Exemple: Vitesse des processeurs double tous les 1,5 ans (loi Empirique), les vitesses d'accès à la Mémoire n'évoluent pas aussi rapidement.

Impact: Les processeurs passent beaucoup de cycles d'attente pour accéder aux données en mémoire.

4 - Caractéristique d'une mémoire:

* Capacité: Nombre maximal de données qu'une mémoire peut contenir.

* Temps d'accès / latence: Temps nécessaire pour accéder à une donnée (lecture / écriture). Varie selon le type de mémoire: SRAM (rapide), DRAM, HDD (lent) ...

* Temps de cycle mémoire: Temps minimal entre deux accès consécutifs à un membre.

Temps cycle = Temps d'accès + Temps d'attente entre deux accès.

* Taux de transfert: Quantité de données transférées par seconde: $\text{Taux de transfert} = \frac{\text{Volume}}{\text{Temps}}$

* Débit: Volume total de données échangé par unité de temps.

* Besoin: Mémoire avec une ~~plus grande capacité~~, réduction des temps d'accès pour suivre les vitesses des processeurs, nécessité de maintenir des coûts raisonnables.

* Dilemme: Mémoire avec plus de capacité implique un temps d'accès plus élevé, une Mem avec un temps d'accès court est plus coûteuse.

* Compromis: Capacité, vitesse, coût.

* Solution: Ne pas baser sur un compromis, utiliser une hiérarchie mémoire.

5 Principe de la hiérarchie mémoire:

Les processeurs n'ont pas besoin d'accéder à toutes les données simultanément.

* Hiérarchie mémoire :

- Mémoires rapides, petites et courtes proches du processeur (cache, registres).

- Mémoires lentes, grandes et moins courtes éloignées du processeur (RAM, disque).

* Objectif :

- Maximiser la performance en gardant les données les plus utilisées dans les mémoires les plus rapides.

6 Mémoire cache

Problème : La Mémoire cache a une capacité limitée, et un cache MISS oblige le processeur à accéder à la mémoire principale, ce qui ralentit l'exécution.

Solutions : Utiliser une hiérarchie de cache (L1, L2, L3), appliquer le préchargement des données et exploiter les principes de la localité pour maximiser l'efficacité.

① Définition : Mémoire rapide et petite de taille qui stock temporairement des données provenant de la mémoire principale.

② Fonctionnement:

① Structure de Transfert:

Transfert pour Mém. Transfert pour Bus

Entre CPU et cache: Les Transferts se font par mots. Cela signifie que des unités élémentaires de données (quelques octets) transistent directement entre ces deux composants.

The diagram illustrates the data flow between the CPU, Cache, and Main Memory. It shows two paths: one from the CPU (labeled 'CPU') to the Cache (labeled 'Cache'), and another from the Cache to the Main Memory (labeled 'MEM PA'). The Cache is represented as a box containing 'L1 Cache' and 'L2 Cache'. Arrows indicate the direction of data transfer between these components.

Entre Cache et Mémoire centrale: Les Transferts se font par blocs (en lignes), ce qui implique qu'une portion plus grande de données est déplacée d'un coup pour optimiser les performances.

② Décomposition en blocs:

- La Mémoire cache est découpée en lignes (blocks) qui représentent les plus petites unités de données avec une étiquette (tag) unique. Ces étiquettes servent à vérifier si les données demandées par le CPU sont présentes dans le cache.

- Lors d'un transfert entre le cache et la Mémoire centrale une ligne entière est déplacée. Cela réduit le nombre d'opérations nécessaires pour des accès fréquents aux mêmes données.

③ Cache comme sous-partie de la Mémoire centrale:

La Mémoire cache est une partie plus rapide mais plus petite de la mémoire centrale. Elle sert à accélérer les performances en stockant temporairement les données les plus utilisées.

④ Principe de la Mémoire cache:

Accès prioritaire: le processeur consulte d'abord la mémoire

cache avant la mémoire principale.

Succès (hit): Si la donnée est trouvée, elle est immédiatement fournie au processeur sans accès à la Mémoire principale.

Échec (miss): En cas d'absence, la donnée est récupérée depuis la mémoire principale et stockée dans le cache avant d'être utilisée.

Taux de succès (hit rate): La fréquence des succès dépend de la taille du cache et de l'algorithme de gestion utilisée, influençant les performances globales.

7 - Technique de conception de la mémoire cache

① Problème de la Mémoire cache:

- Taille limitée: Une Mémoire cache est petite, ce qui empêche de stocker tout un programme et ses données.

- Optimisation nécessaire: Les données essentielles doivent être choisies pour maximiser le débit.

② Solution:

- Pré-chargement (pre-fetching): charger en avance les données nécessaires basé sur les principes de localité.

- Placement et identification: Organisation de la mémoire cache pour décider quelles données y placer.

③ Principe de localité:

- Localité Temporelle: Les données récemment utilisées ont une forte probabilité d'être prochainement, **ex: les boucles.**

Localité Spatiale: Les données proches en mémoire ont tendance

à être accédés successivement. Ex: les tableaux.

① Placement et organisation des blocs dans le cache:

Placement direct (Direct Mapped): chaque donnée a une position unique chaque ligne du cache unique dans le cache.

Tag	Index	Offset
-----	-------	--------

Avantages: accès rapide et technique simple.

Inconvénients: conflits de cache lorsque 2 blocs doivent occuper la même ligne, entraînant une faute d'accès!

Associativité pure (Fully associative): Une donnée peut être placée n'importe où dans le cache.

Tag	Offset
-----	--------

Avantages: flexibilité maximale, chaque bloc peut être placé n'importe où dans le cache.

Inconvénients: recherche lente, surtout avec une grande taille de cache, car tous les tags doivent être comparés en parallèle.

Associativité par ensemble (Set Associative): la Mémoire cache est divisée en ensembles. Une donnée peut être placée dans n'importe quel emplacement d'un ensemble spécifique.

Tag	Set	Offset
-----	-----	--------

Avantages: bon compromis entre flexibilité et performance, plus

Inconvénients: moins rapide que le fully associative.

Inconvénients: moins flexible que le fully associative, mais plus intenses organique.

2. Politiques de remplacement des blocs :

- Direct Mapped: pas de méthode de remplacement spécifique, car une ligne mémoire ne peut être assignée qu'à une seule ligne de cache.

- Fully associatif: trop de choix pour remplacer une ligne, nécessite une méthode de gestion.

- Sets associatif: moins de choix que le fully associatif, mais une méthode de remplacement est nécessaire.

• Méthodes de Remplacement des blocs :

- Aleatoire: simple à mettre en œuvre, mais peu efficace car des lignes très accès peuvent être supprimées.

(First In First Out) - FIFO: remplace la ligne la plus ancienne, plus efficace que l'aleatoire, mais peut ~~être~~ retirer une ligne fréquemment utilisée si elle devient trop vieille.

(Least Recently Used) - LRU: remplace la ligne la moins récemment utilisée, très efficace, mais plus complexe.

(Least Frequently Used) - LFU: remplace la ligne utilisée le moins fréquemment ou utilisée le moins. Pas simple que LRU, mais nécessite une gestion des compteurs.

(Not Most Recently Used) - NMRU: variante de LRU où la ligne la plus récemment utilisée reste dans le cache. La ligne à remplacer est choisie parmi celles non récemment utilisées.

9. Stratégies pour gérer l'accès des données entre le cache et la mémoire principale :

Write-through: écriture immédiate dans la mémoire principale et le cache à chaque modification.

Copyback (Write-back): écriture dans le cache uniquement, avec mise à jour de la mémoire principale lors du retrait de la ligne en utilisant un "dirty bit".

10. Les points clés sur l'amélioration de la performance de la Mémoire cache !

1. Mesure des performances de la cache :

- **Taux de fautes (Miss rate)** = Nombre de Miss / Nombre total d'accès.
- **Taux de succès (Hit rate)** = Nombre de Hits / Nombre total d'accès.
- **Pénalité de faute (Miss penalty)** = Nombre de cycles supplémentaires en cas de faute de cache.
- **Temps de succès (Hit time)** = Nombre de cycles pour obtenir la donnée en cas de succès.
- **Temps d'accès moyen :**
 - Temps d'accès moyen = Temps HIT + (Taux MISS × Pénalité MISS).

2. Amélioration des performances :

- **Réduire le Temps HIT** : Réduire le temps nécessaire pour obtenir une donnée lorsque le cache est un succès.
- **Réduire le Taux MISS** : Minimiser le nombre de fautes de cache.
- **Réduire la Pénalité MISS** : Minimiser le coût en cycles supplémentaires en cas de faute de cache.

3. Réduction du Taux MISS :

- **Taille de ligne plus élevée** : Augmenter la taille des lignes du cache.
- **Taille du cache plus grande** : Avoir plus de mémoire cache.
- **Plus d'associativité** : Augmenter le niveau d'associativité dans le cache pour mieux distribuer les blocs.

4. Réduction de la Pénalité MISS :

- **Caches multi-niveaux** :
 - **Niveau L1** : Cache interne au processeur, rapide, mais de petite taille (SRAM).
 - **Niveau L2** : Cache interne au processeur, plus grand que L1 mais moins rapide (DRAM).
 - **Niveau L3** : Cache externe au processeur, situé sur la carte mère, généralement plus grand et plus lent que L1 et L2.

5. Relation entre les niveaux de cache :

- **Cache inclusif** : Le contenu du cache L1 est aussi dans L2, donc la taille globale du cache est celle de L2.
- **Cache exclusif** : Le contenu de L1 et L2 est distinct, donc la taille globale du cache est la somme des tailles de L1 et L2.

6. Conclusion :

- Les performances de la mémoire cache dépendent de plusieurs facteurs :
 - Taille du cache et organisation des niveaux.
 - Méthode d'association des lignes entre mémoire centrale et cache.
 - Rapidité d'accès aux bonnes lignes dans le cache.
 - Méthode de remplacement des lignes.
 - Chargement en avance des données nécessaires.

$$1\text{THz} = 10^{12} \text{Hz}$$

$$1\text{GHz} = 10^9 \text{Hz}$$

$$1\text{MHz} = 10^6 \text{Hz}$$

$$1\text{kHz} = 10^3 \text{Hz}$$

$$1\text{Hz} = 10^{-6} \text{Hz}$$

* bit → Kilobits

$$K\text{bts} = \frac{\text{bts}}{1024}$$

* bits → Megabits

$$M\text{bts} = \frac{\text{bts}}{1024 \times 1024}$$

* bits → octets

$$\text{octets} = \frac{\text{bts}}{8}$$

$$1\text{ps} = 10^{-12} \text{s}$$

$$1\text{ns} = 10^{-9} \text{s}$$

$$1\text{μs} = 10^{-6} \text{s}$$

$$1\text{ms} = 10^{-3} \text{s}$$

①

bits ver kilobits

$$Kb = \frac{\text{bts}}{8 \times 1024}$$

bits ver megabits

$$Mb = \frac{\text{bts}}{8 \times 1024 \times 1024}$$

$$1KO \rightarrow 1024 \quad | \quad 1MO \rightarrow 1024^2 \quad | \quad 1GO, \quad | \quad (1024)^3$$

Temps d'exécution = Nins x CPI x Tcycle
Nombre x Nombre x Tcycle

$$Tcycle = \frac{1}{\text{fréquence}}$$

$$CPI = \frac{\text{nbr de cycle}}{\text{nbr Inst}}$$

$$\text{nbr mœts} = \frac{\text{Capacité}}{\text{nbr Ø}}$$

$$\text{Capacité} = 2^{\text{taille max}} \times \text{taille blocs de données}$$

②

* Calcul Temps Total d'exécution des Inst:

Sequentiel :

$$T_T = N \times T_p$$

↓

$$T_T = N \times K \times T_e$$

↓

T_e : Temps exécution d'une étape
 N: nombre d'étapes d'une inst.
 T_p : le temps d'exécution d'une inst.
 N : le Nbr d'inst à exécuter
 T_T : Temps Total

Pipeline :

$$T_T = T_p + (N-1) \times T_e \Rightarrow K \times T_e + (N-1) \times T_e$$

$$\Rightarrow T_e \times (K+N-1)$$

* Calcul de l'accélération :

$$A = \frac{T_{seq}}{T_{pip}}$$

$$= \frac{N \times K \times T_e}{(K+N-1) \times T_e} \Rightarrow \frac{N \times K}{K+N-1}$$

(3)

* Calcul du débit du pipeline:

$$D = \frac{1}{T_{\text{esui}}} \quad \text{avec } T_{\text{esui}} = \frac{T_f}{N}$$

$$D = \frac{N}{(k+N-1) \cdot T_e} \quad \Rightarrow \quad \frac{1}{\text{Temps de cycle}}$$

La latence : Somme des durées des étages -

$$T_c \times N_b$$

④

CHAP 3:

Nombre de lignes = Taille de la Mem Cache
① Mem Cache Taille d'une ligne

Nombre de mots = Taille d'une ligne
② Taille d'un mot

Nombre de blocs = Taille de la Mem centrale
dans la Mem centrale = Taille d'une ligne

Cas: Organisation direct (Object Mapped):

- Taille index: Nbr de lignes = Table index
Mem Cache = 2

- Taille de l'offset: Nbr de mots = 2^{offset}
mots

⑤

Table du Tag = Taille de l'^{ea}_{Mem} - Taille de l'adresse de l'offset

Remarque: Si on a pas la Taille de l'^{ea}_{Mem} on peut la calculer de suite de :
Nb mots contenu dans la Mem centrale = Taille de la Mem centrale / Taille d'un mot
le résultat sera sous forme $T @ \text{mem}^{\text{adr}}$

Tag	Index	Offset
-----	-------	--------

Cas: Organisation Associative Totale
(Fully Associative):

Les bits ①, ②, ③, ④ ne changent pas juste
on va pas calculer l'adresse ⑤

$\text{diff. Tag} = \text{Taille de l'offset} - \text{taille}_{\text{RAM}}$

Tag	Offset
-----	--------

Casi Organisation associative par ensemble (Set Associative):

Les bits ①, ②, ③, ④ ne changent pas juste à la place de l'adresse, on va calculer le nbr d'ensembles :

$$\text{Nbr ensemble} = \frac{\text{Nb ligne}}{\text{Nb ensemble}} = \frac{16}{2} = 8$$

7

$$\text{Accélération} = \frac{1}{(1-F_a) + \frac{F_a}{\text{Accélération (Dc)}}}$$

F_a: Fraction (Partie) du programme utilisée

Laï pour mettre les valeurs dans la Mémoire Cache:

$$\text{N° ligne} = \text{N° bloc mod}(\text{Nbr de lignes})$$

⑧