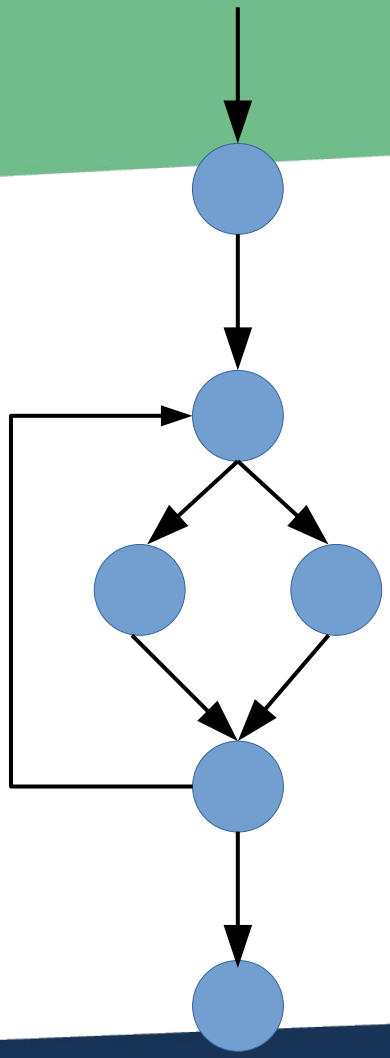
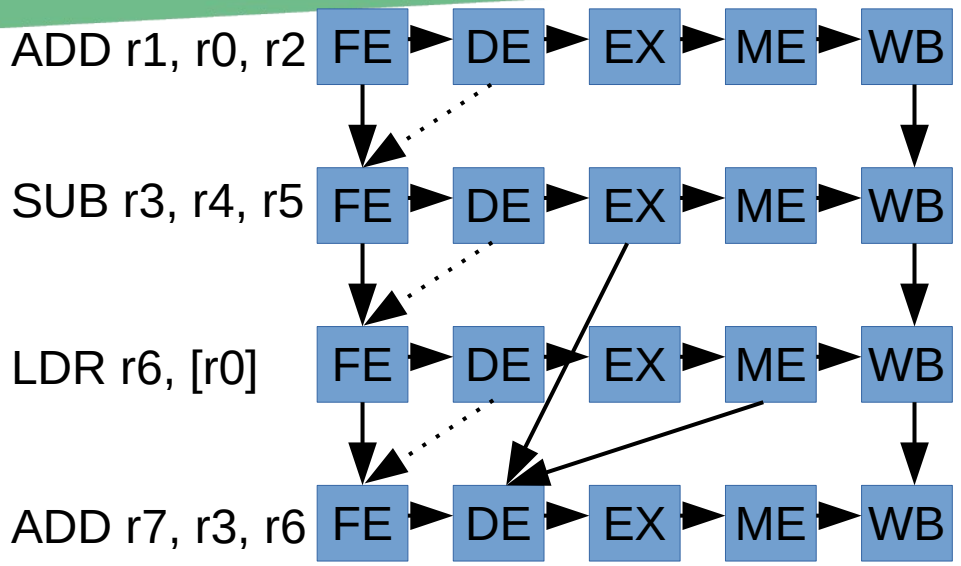
Analyse WCET et OTAWA

```
int main(){
        int j= 0, k=0;
        int* t = 0x100000 ;
        for(int i =0 ; i<16 ; i++)
                if(t[i]<100){
                        j++;
                }else{
                        k++ ;
                }
        return 0;
}
```





WCET = 12000 cycles

Analyse WCET et OTAWA
Contexte

Systèmes embarqués temps-réel

- contraintes dures (e.g. échéances)
- systèmes critiques

La connaissance du pire temps d'exécution des programmes embarqués est nécessaire pour la validation temporelle

- analyse de pire temps de réponse
- construction d'un ordonnancement statique valide

Analyse WCET et OTAWA
Introduction

Pire temps d'exécution ?

→ **variabilité du temps d'exécution d'un programme**

- Liée à l'environnement (valeurs des capteurs, modes d'opération)
- Liée à la structure du programme (branches non équilibrées, boucles avec nombre d'itérations variable)

Analyse WCET et OTAWA
Introduction

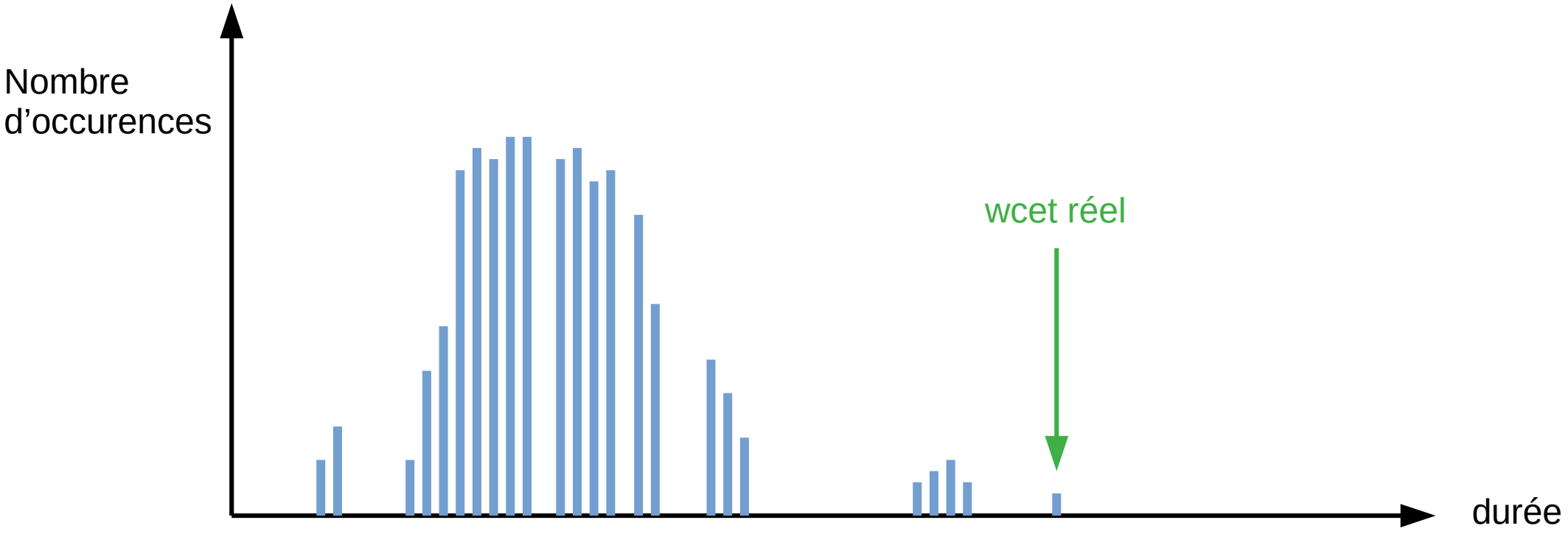
Pire temps d'exécution ?

→ **variabilité du temps d'exécution d'un programme sur un « processeur » cible**

- Liée à l'environnement (valeurs des capteurs, modes d'opération)
- Liée à la structure du programme (branches non équilibrées, boucles avec nombre d'itérations variable)
- Liée au matériel (instructions à latence variable, état du matériel - caches, prédicteurs de branchement)

Analyse WCET et OTAWA
Introduction

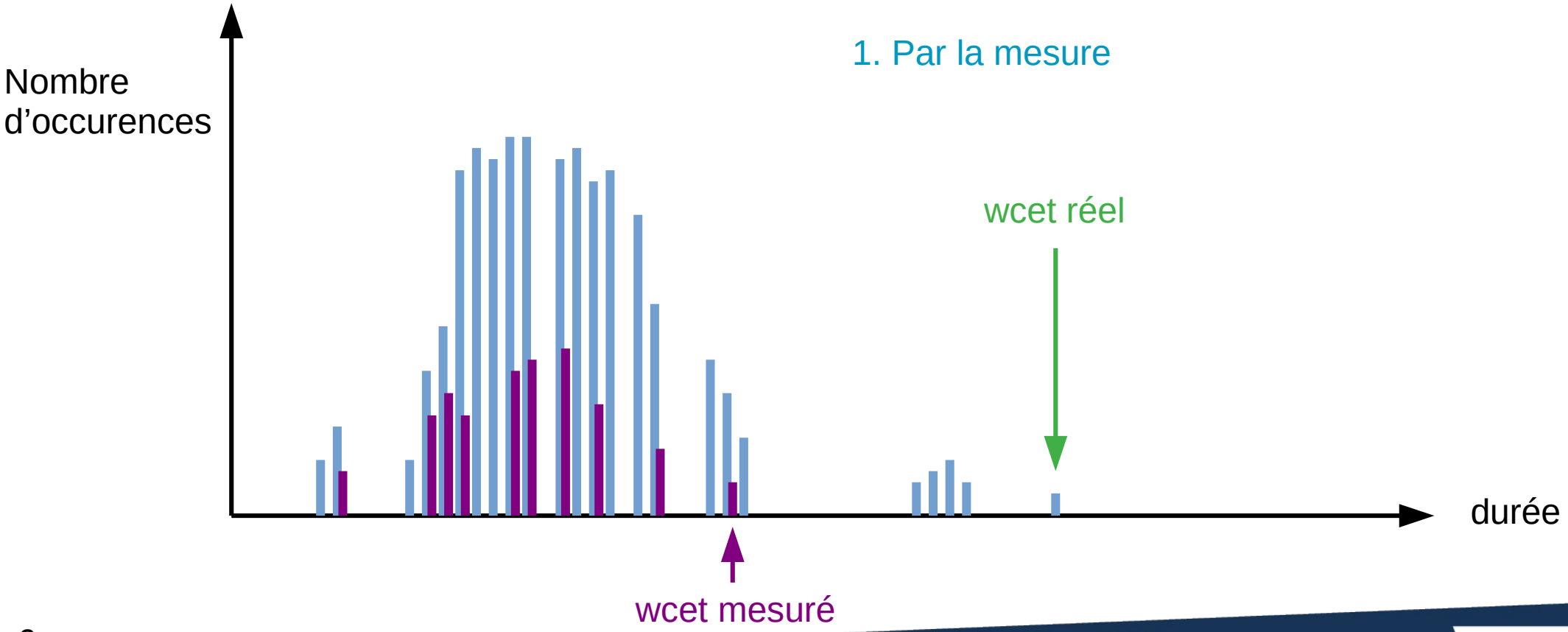
Objectif : déterminer le pire temps d'exécution d'un programme :



Analyse WCET et OTAWA
Introduction

Objectif : déterminer le pire temps d'exécution d'un programme :

1. Par la mesure



Analyse WCET et OTAWA
Introduction

Objectif : déterminer le pire temps d'exécution d'un programme :

→ **Par la mesure**

- Relativement simple et rapide à mettre en œuvre
- Impossible de couvrir tous les cas : difficile de prouver que la couverture est suffisante → certification difficile

Analyse WCET et OTAWA
Introduction

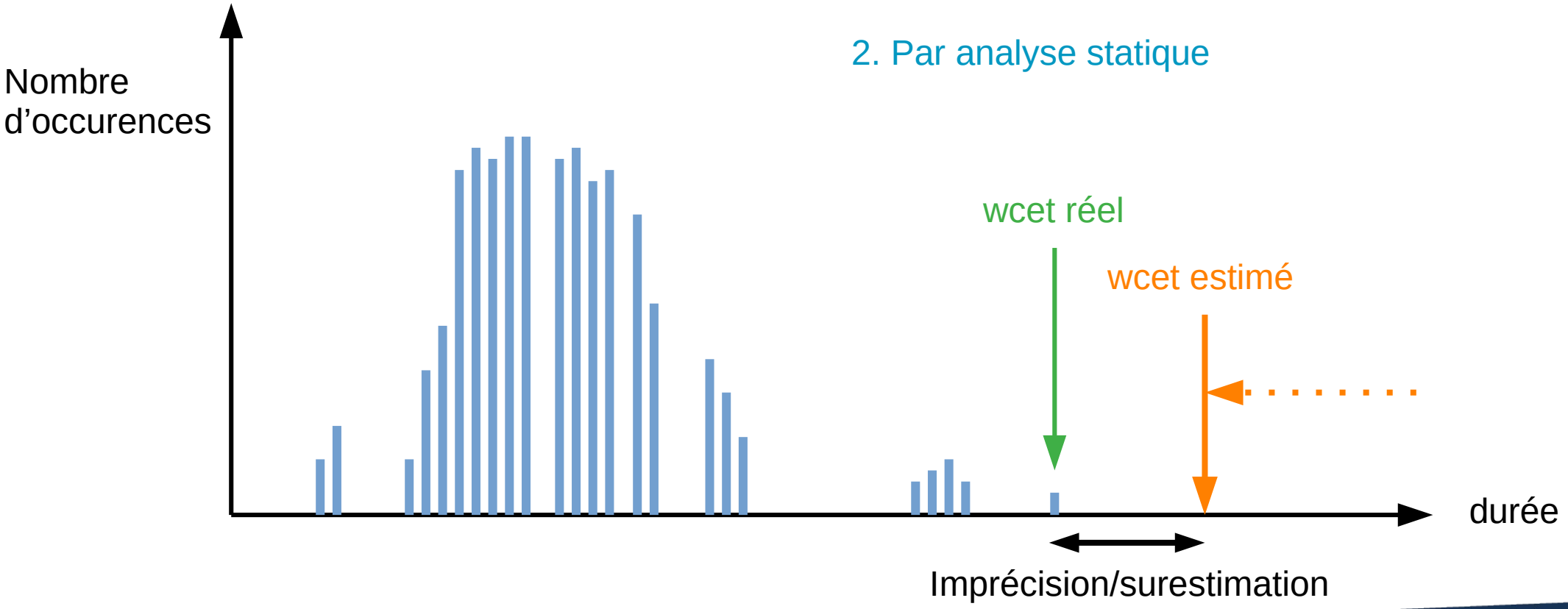
Objectif : déterminer le pire temps d'exécution d'un programme :

→ **Par la mesure**
- Relativement simple et rapide à mettre en œuvre
- Impossible de couvrir tous les cas : difficile de prouver que la couverture est suffisante → certification difficile

→ **Méthodes probabilistes**
- Campagne de mesures, puis reconstruction d'une distribution
- Obtentions de pWCETs i.e. couples (WCET, proba de dépassement)

Analyse WCET et OTAWA
Introduction

Objectif : déterminer le pire temps d'exécution d'un programme :



Analyse WCET et OTAWA
Introduction

Objectif : déterminer le pire temps d'exécution d'un programme :

→ **Par la mesure**
- Relativement simple et rapide à mettre en œuvre
- Impossible de couvrir tous les cas → difficile de prouver que la couverture est suffisante

→ **Par analyse statique**
- Définir un modèle temporel du fonctionnement du matériel
- Découper le code en morceaux simples (blocs de base) et (sur-)estimer leur durée d'exécution
- Surestimer les propriétés du flot d'exécution par analyse statique
- Combiner ces informations et chercher le temps maximum possible
- Plus compliqué à mettre en œuvre (modèle + surestimation)
- Garantie que le wcet est surestimé

Analyse WCET et OTAWA
Introduction

→ **Par analyse statique**

- Définir un modèle temporel du fonctionnement du matériel
  → Pipeline, caches, prédicteurs de branchement, mémoire, etc.
- Découper le code en morceaux simples (blocs de base) et (sur-)estimer leur durée d'exécution par analyse statique
  → CFG, analyses de cache, ExeGraphs
- Surestimer les propriétés du flot d'exécution par analyse statique
  → Bornes de boucles, chemins infaisables, etc.
- Combiner ces informations et chercher le temps maximum possible
  → IPET (Implicit Path Enumeration Technique)
- Plus compliqué à mettre en œuvre
- Garantie que le wcet est surestimé

Analyse WCET et OTAWA
Introduction

Objectif : déterminer le pire temps d'exécution d'un programme :

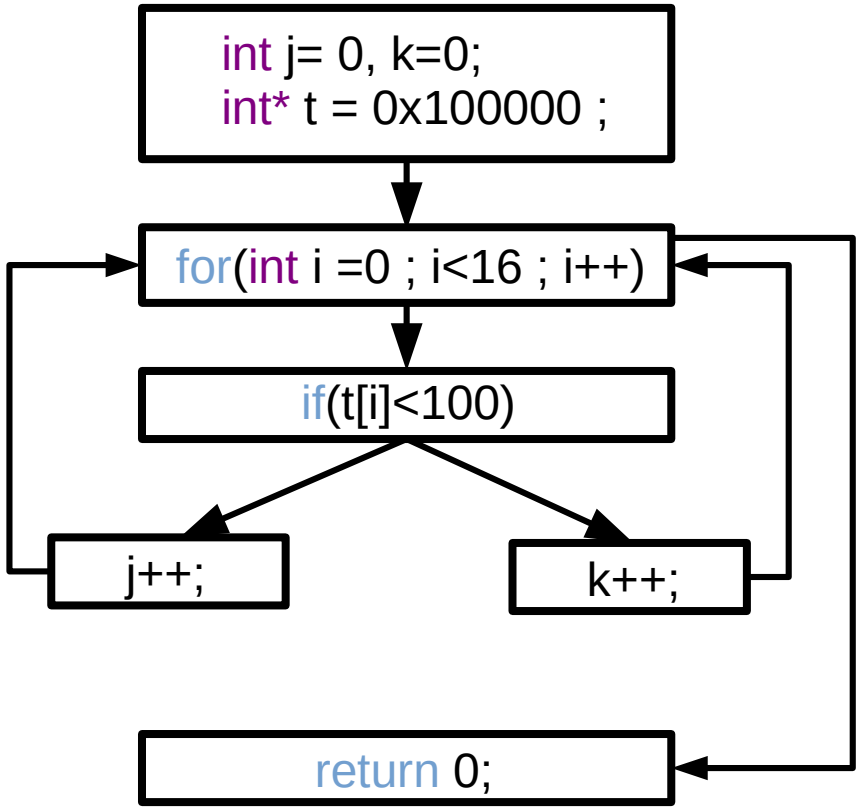
→ **Méthodes hybrides**

- Découper le code en morceaux simples (blocs de base) et mesurer leur durée d'exécution
- Surestimer les propriétés du flot d'exécution par analyse statique
- Combiner ces informations et chercher le temps maximum possible
- Le meilleur (ou le pire) des deux mondes

Analyse WCET et OTAWA
Méthode IPET : les grandes lignes

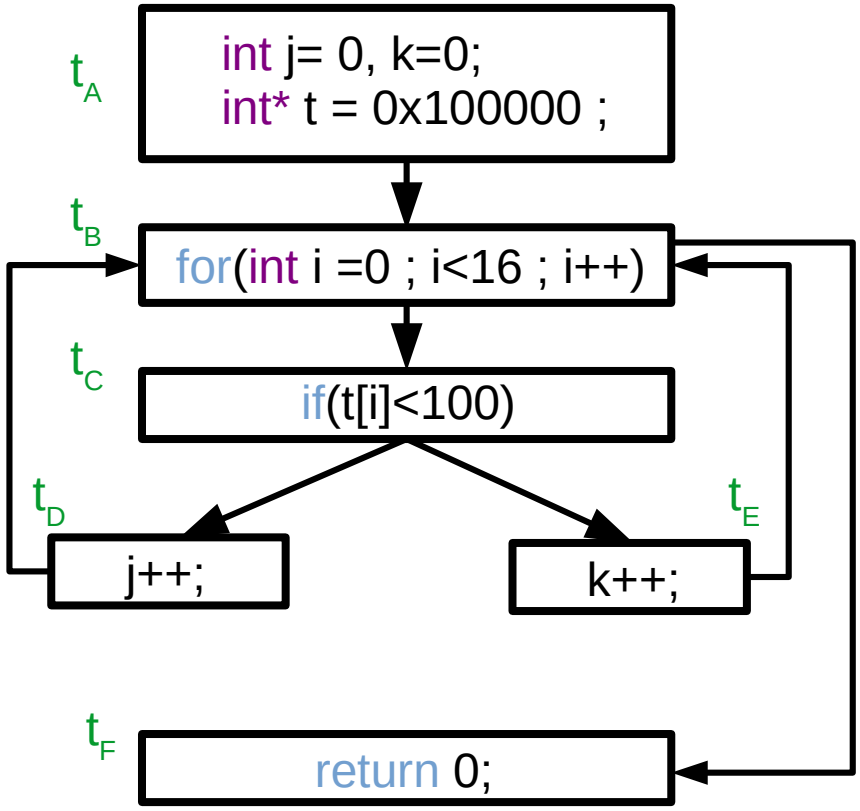
→ **Obtenir un graphe de flot de contrôle**
- Code source vs binaire
- Les nœuds sont des blocs de base
- les arcs définissent le flot de contrôle

```
int main(){
        int j= 0, k=0;
        int* t = 0x100000 ;
        for(int i =0 ; i<16 ; i++)
                if(t[i]<100){
                        j++;
                }else{
                        k++ ;
                }
        return 0;
}
```



Analyse WCET et OTAWA
Méthode IPET : les grandes lignes

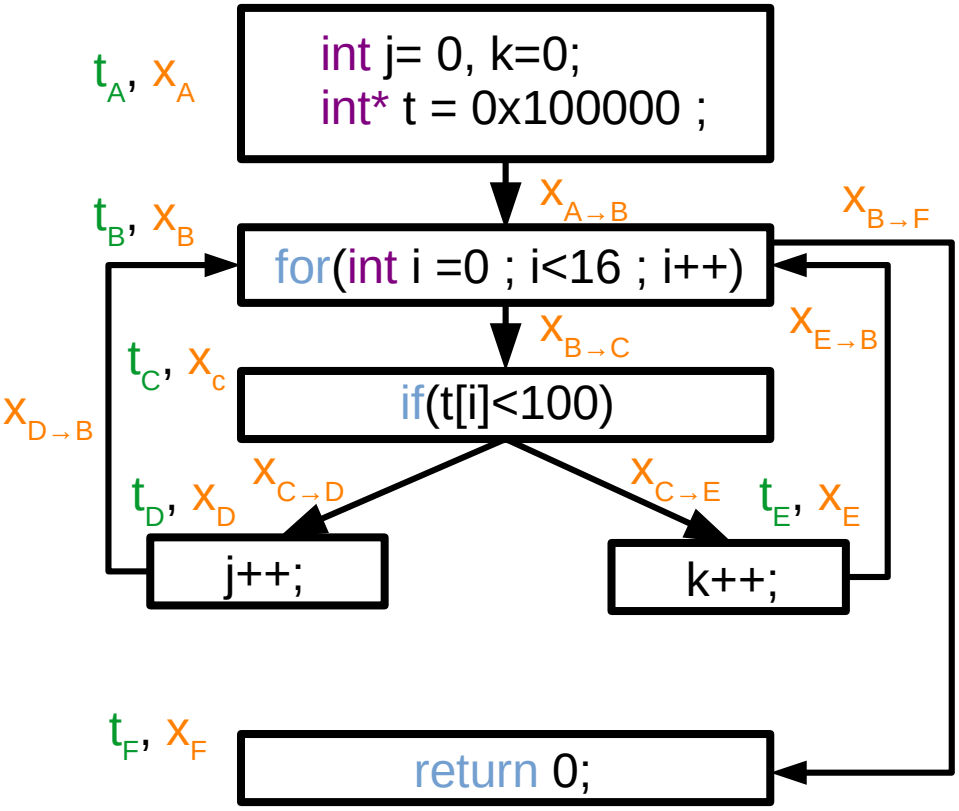
→ **Associer une durée t à chaque bloc**
- Déterminée par analyse
(- Dans le cas Hybride, déterminée par des mesures)



Analyse WCET et OTAWA
Méthode IPET : les grandes lignes

→ **Associer une durée t à chaque bloc**
- Déterminée par analyse
(- Dans le cas Hybride, déterminée par des mesures)

→ **Associer des variables d'occurence x**
- A chaque bloc
- A chaque arc



Analyse WCET et OTAWA

Méthode IPET : les grandes lignes

→ **Associer une durée t à chaque bloc**
- Déterminée par analyse
(- Dans le cas Hybride, déterminée par des mesures)

→ **Associer des variables d'occurence x**
- A chaque bloc
- A chaque arc

→ **Lier les variables par des contraintes ILP**
- Formulation du flot
- Bornes de boucle

$x_A = 1 ; x_{A \to B} = x_A$

$x_B = x_{A \to B} + x_{D \to B} + x_{E \to B}$

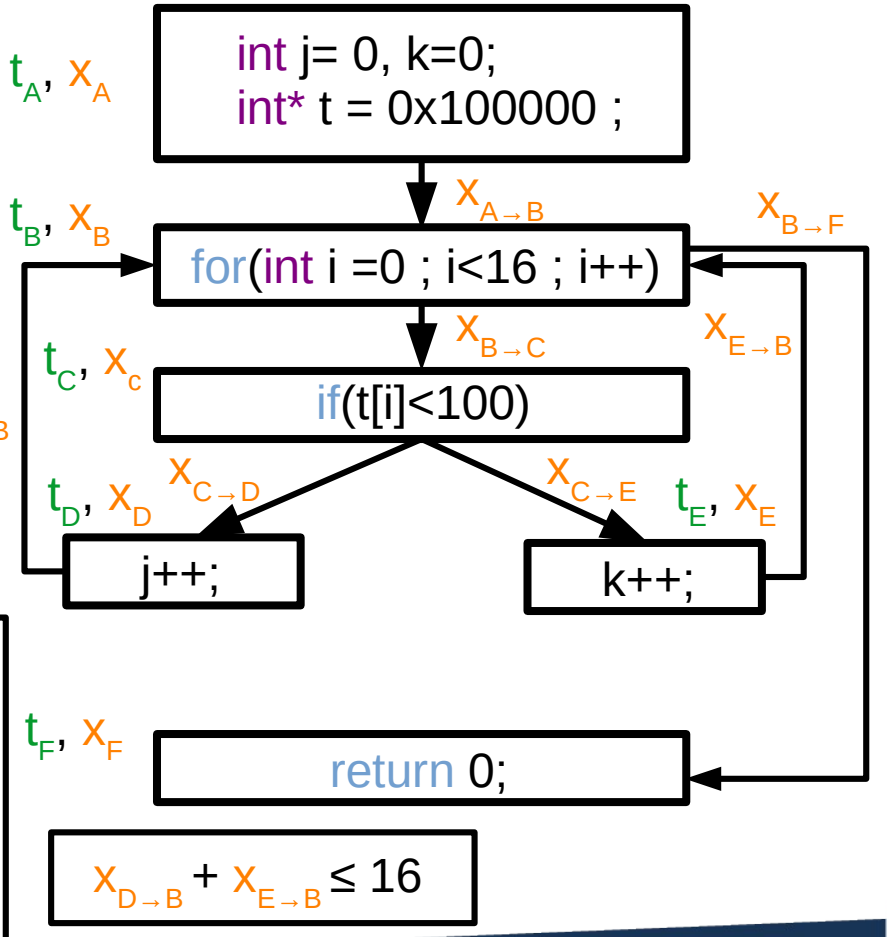
$x_{B \to C} + x_{B \to F} = x_B ; x_C = x_{B \to C}$

$x_{C \to D} + x_{C \to E} = x_C$

$x_{D \to B} = x_D = x_{C \to D}$

$x_{E \to B} = x_E = x_{C \to E}$

$x_F = x_{B \to F} = 1$



Analyse WCET et OTAWA
Méthode IPET : les grandes lignes

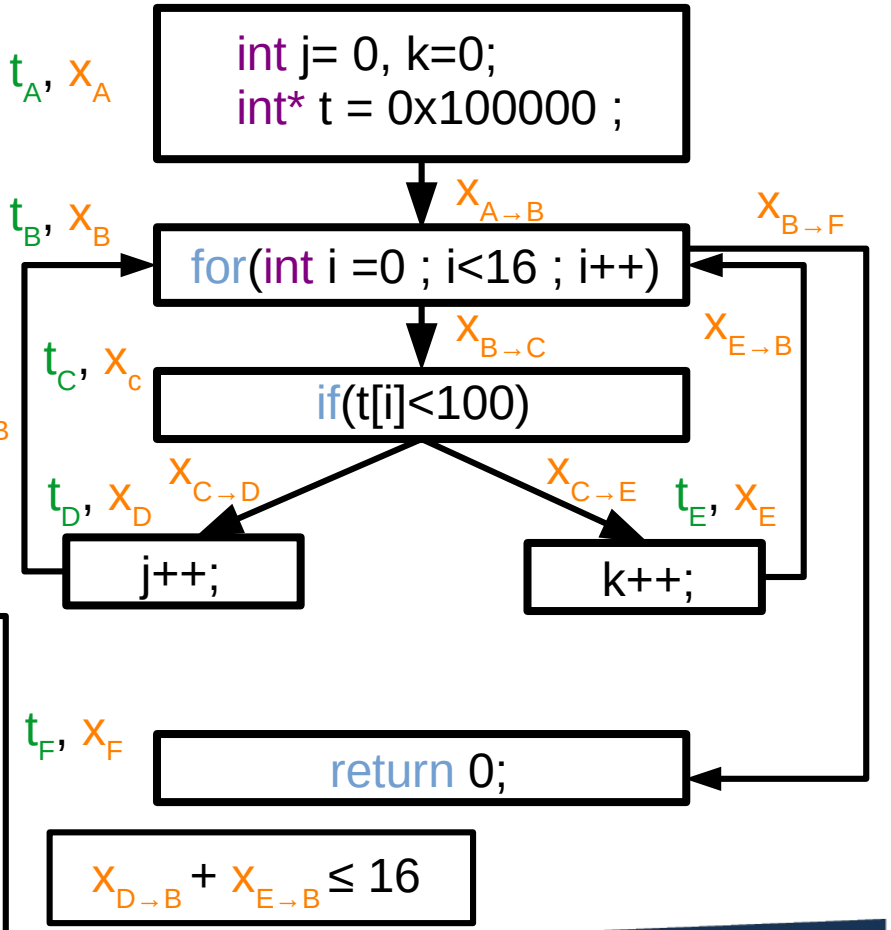
→ **Associer une durée t à chaque bloc**
- Déterminée par analyse
(- Dans le cas Hybride, déterminée par des mesures)

→ **Associer des variables d'occurence x**
- A chaque bloc
- A chaque arc

→ **Lier les variables par des contraintes ILP**
- Formulation du flot
- Bornes de boucle

→ **Maximiser :**

$$WCET = MAX(SUM_i(t_i * x_i))$$



$x_A = 1 ; x_{A \to B} = x_A$
$x_B = x_{A \to B} + x_{D \to B} + x_{E \to B}$
$x_{B \to C} + x_{B \to F} = x_B ; x_C = x_{B \to C}$
$x_{C \to D} + x_{C \to E} = x_C$
$x_{D \to B} = x_D = x_{C \to D}$
$x_{E \to B} = x_E = x_{C \to E}$
$x_F = x_{B \to F} = 1$

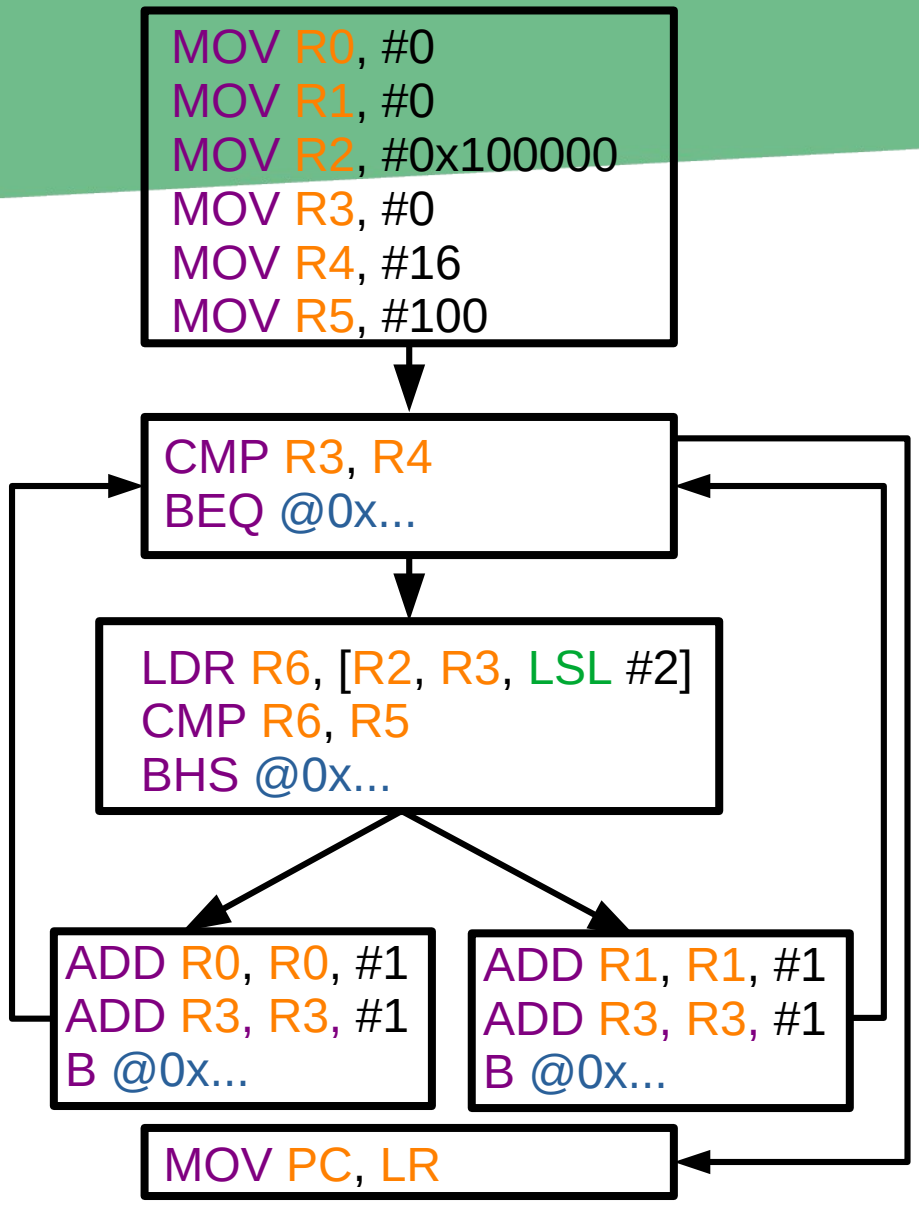
Analyse WCET et OTAWA
Et les t_i ?

→ **Analyse microarchitecturale**
- Modélisation des éléments micro-architecturaux
- Travail sur le fichier binaire correspondant à l'exécutable
  → Plus précis, et certitude d'analyser ce qui va être exécuté

Analyse WCET et OTAWA

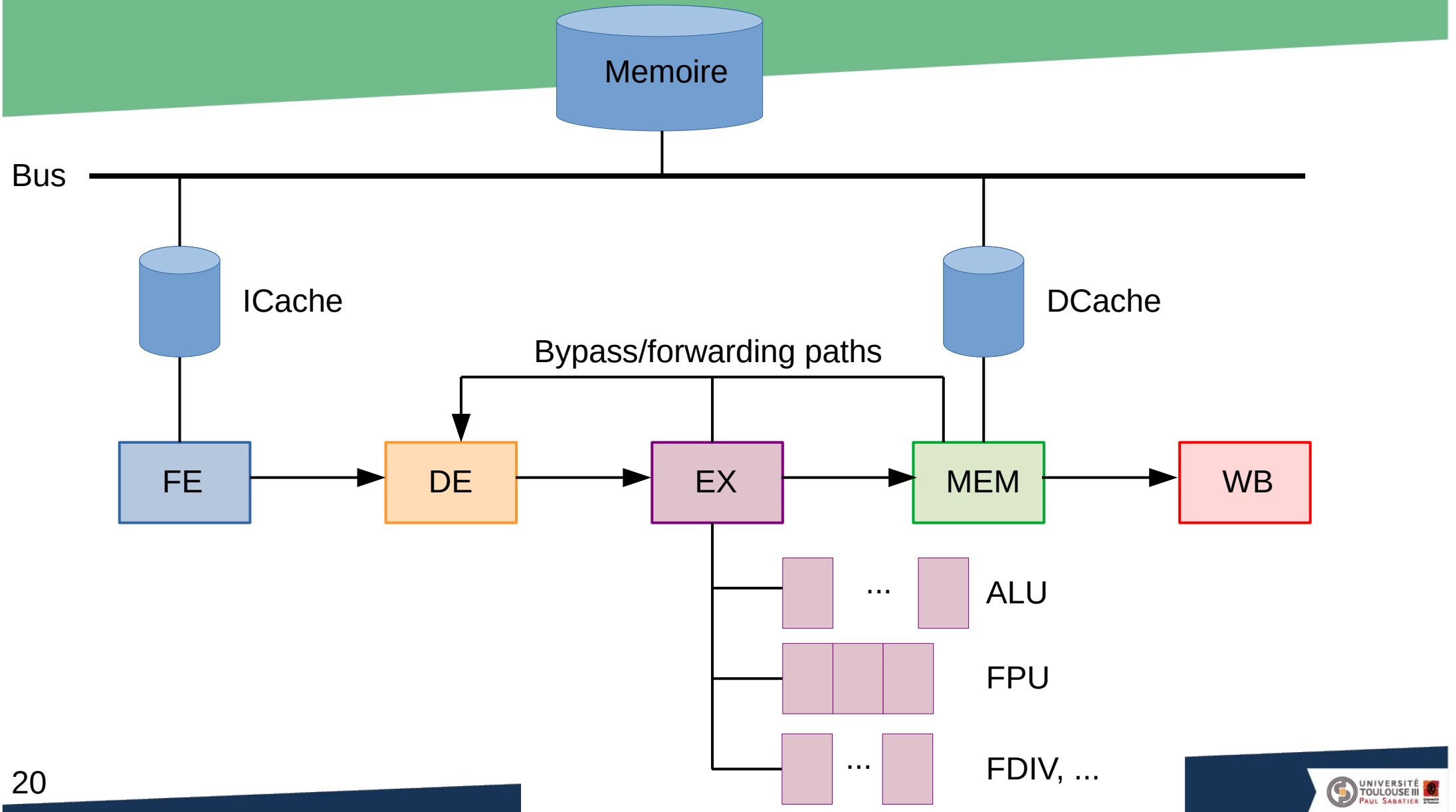
Et les t_i ?

→ **Analyse microarchitecturale**



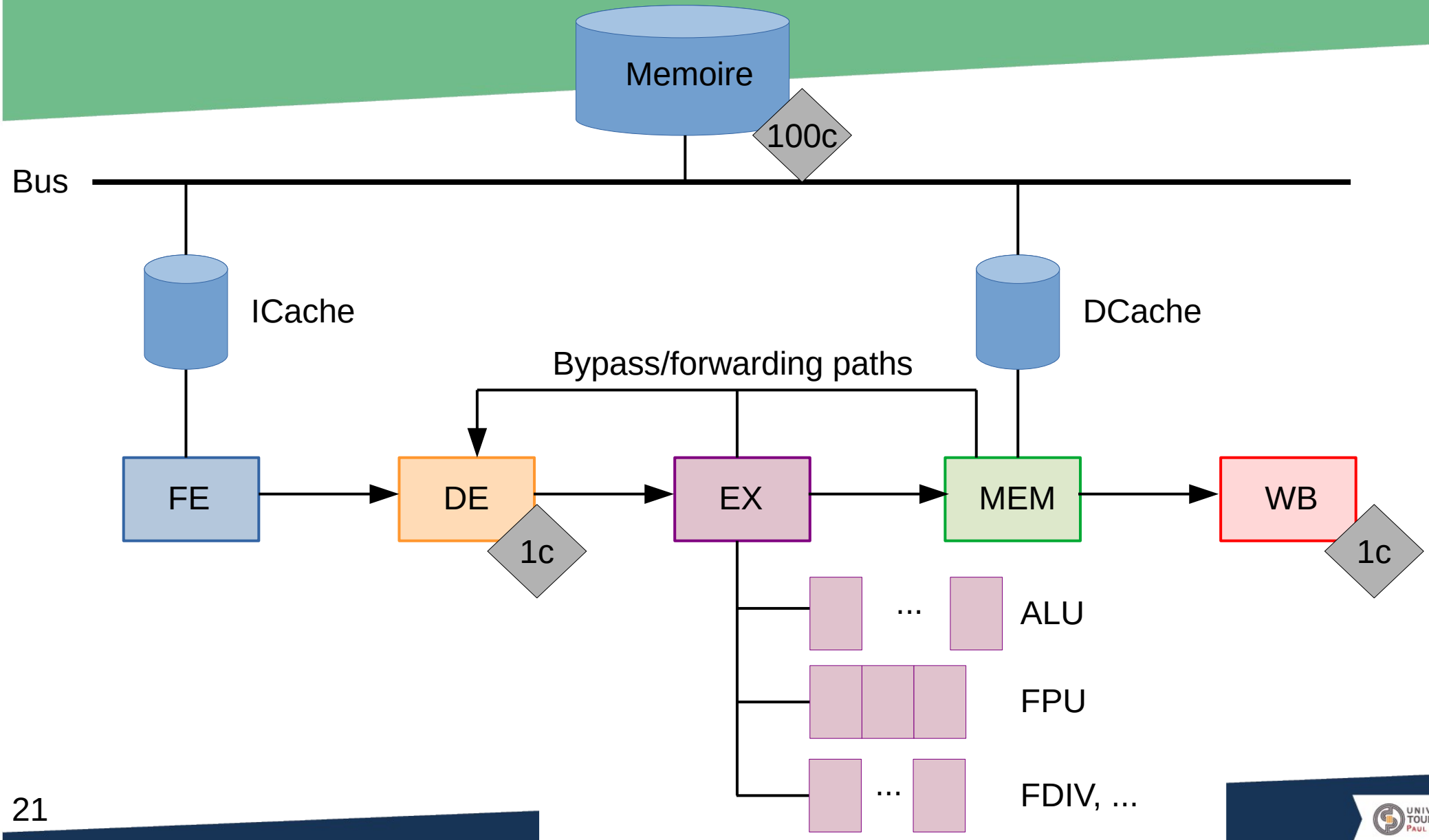
Analyse WCET et OTAWA

Modéliser l'architecture



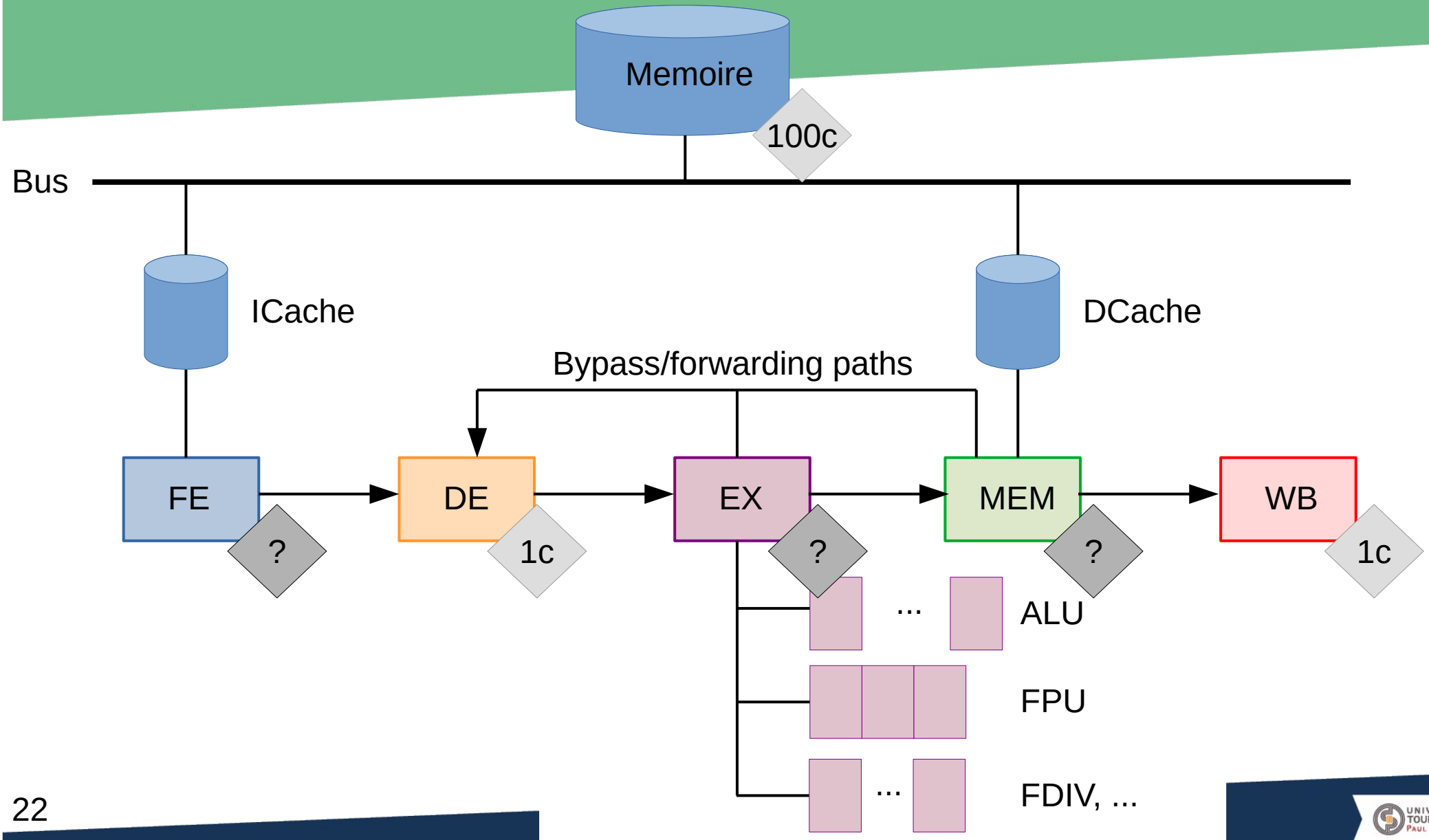
Analyse WCET et OTAWA

Modéliser l'architecture



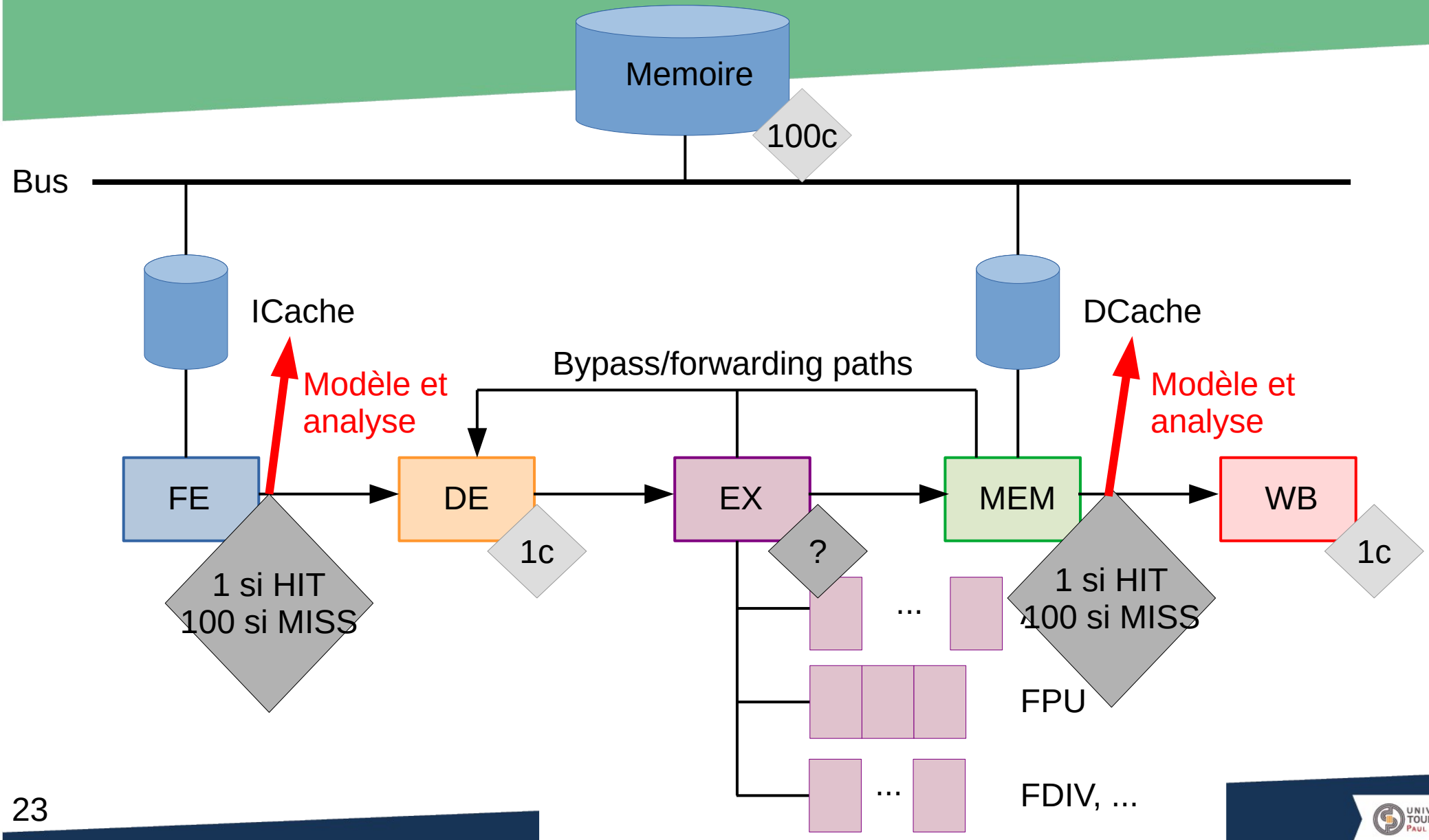
Analyse WCET et OTAWA

Modéliser l'architecture



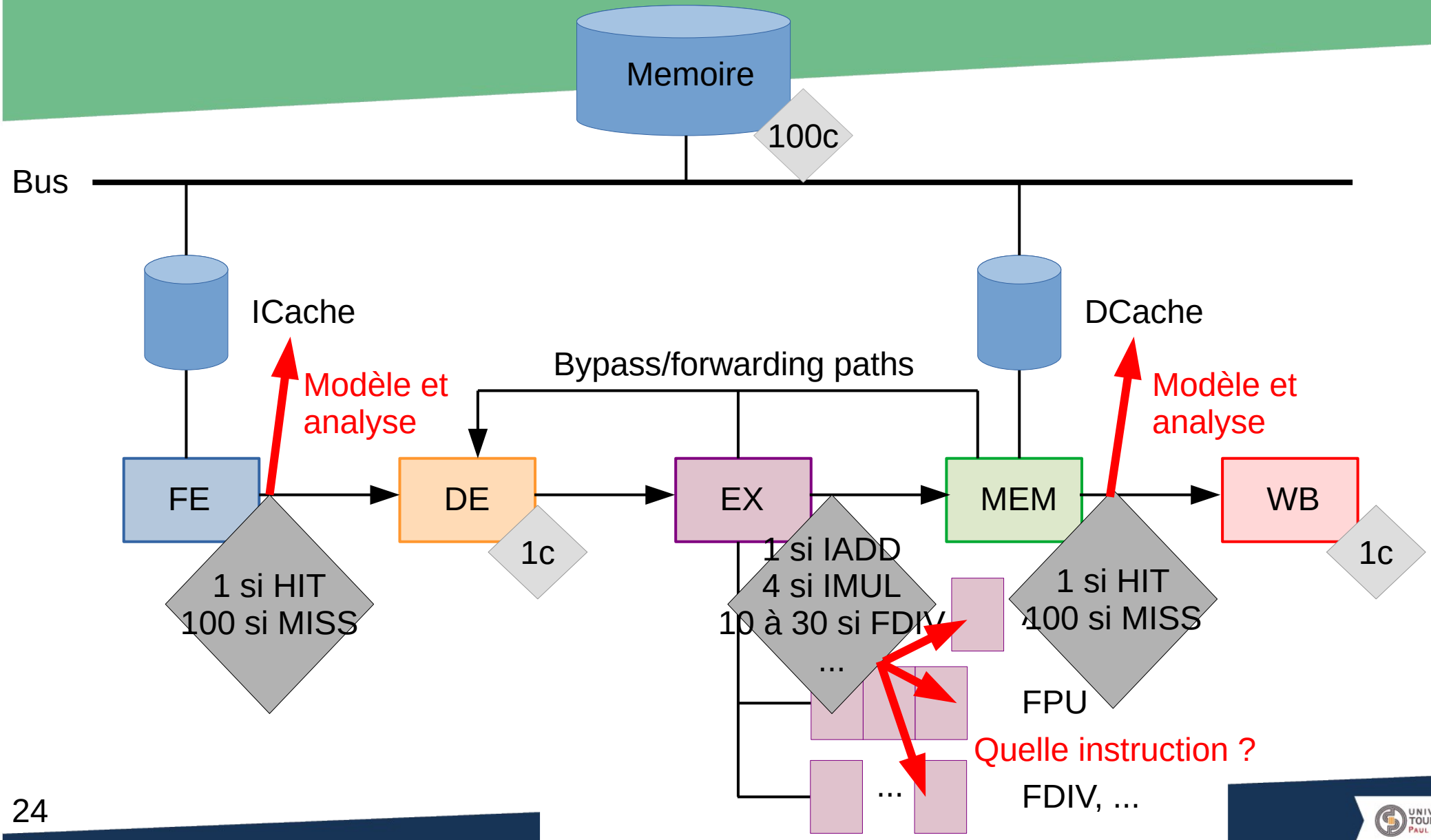
Analyse WCET et OTAWA

Modéliser l'architecture



Analyse WCET et OTAWA

Modéliser l'architecture



Analyse WCET et OTAWA
Et les t_i ?

→ **Analyse microarchitecturale**

- Modélisation des éléments micro-architecturaux
- Travail sur le fichier binaire correspondant à l'exécutable

→ Plus précis, et certitude d'analyser ce qui va être exécuté

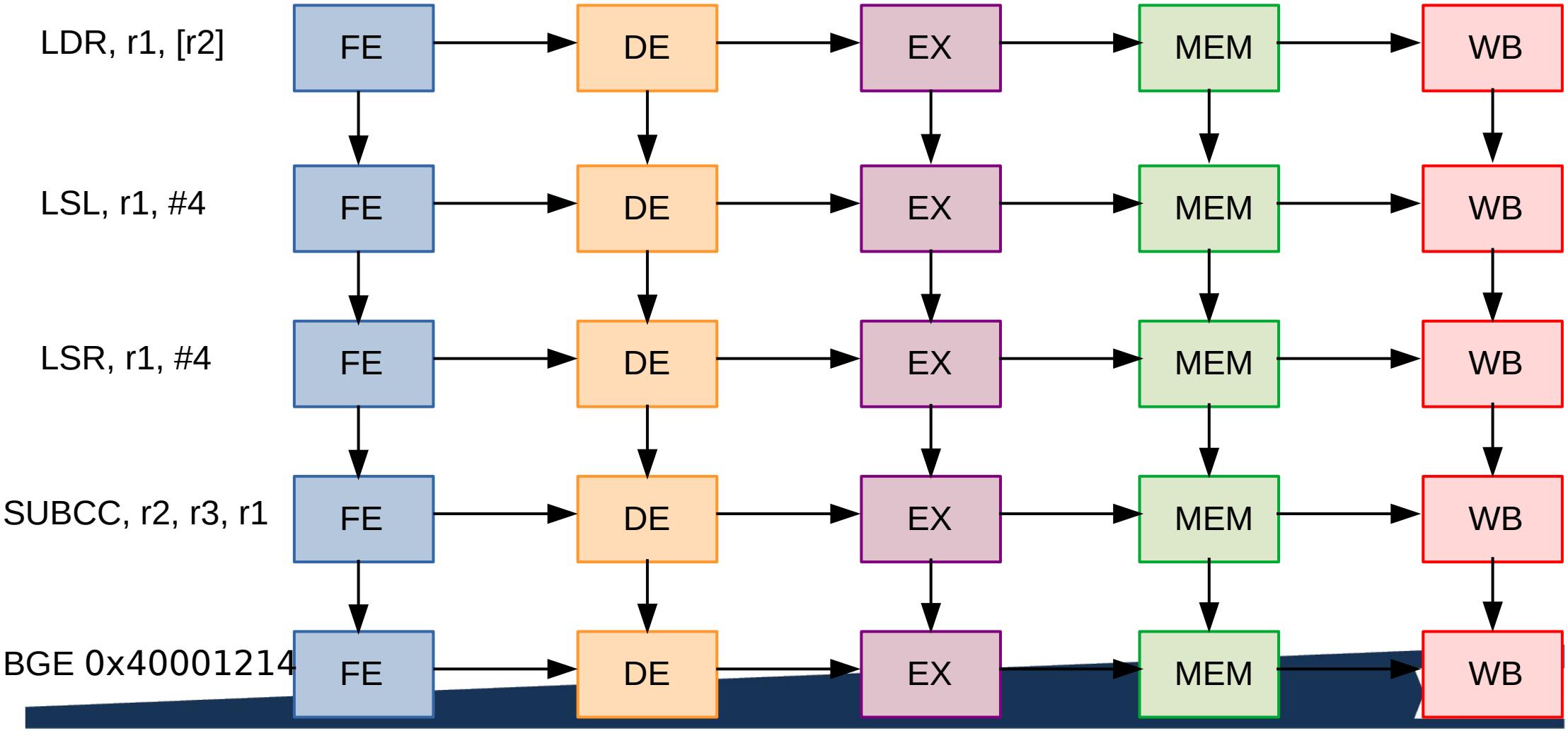
- Déterminer la durée maximale de traversée du pipeline de la séquence d'instructions correspondant au bloc de base considéré

→ Le modèle du pipeline est l'élément central de l'analyse microarchitecturale

Analyse WCET et OTAWA

Modéliser le pipeline : EXEGraphs

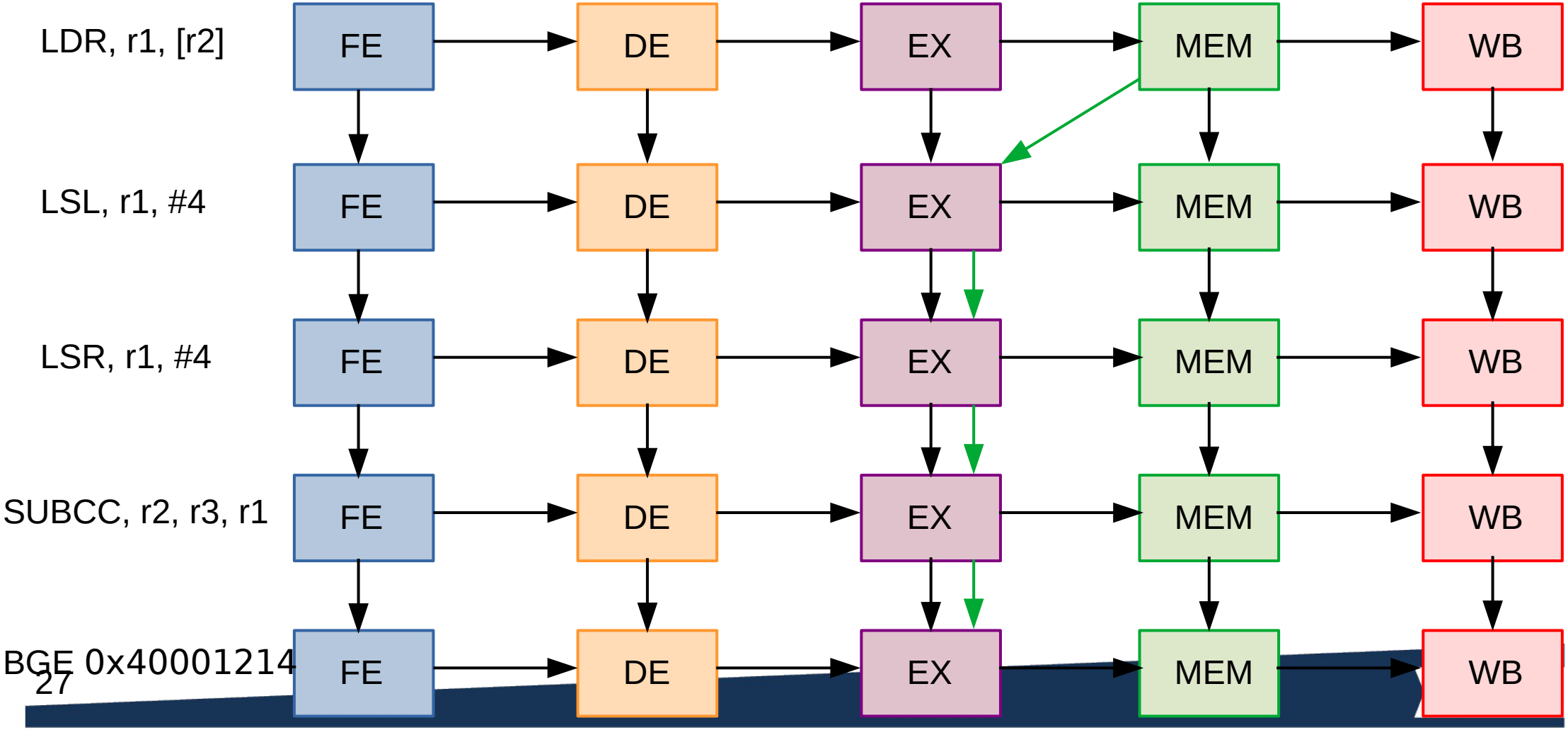
Contraintes structurelles du pipeline



Analyse WCET et OTAWA

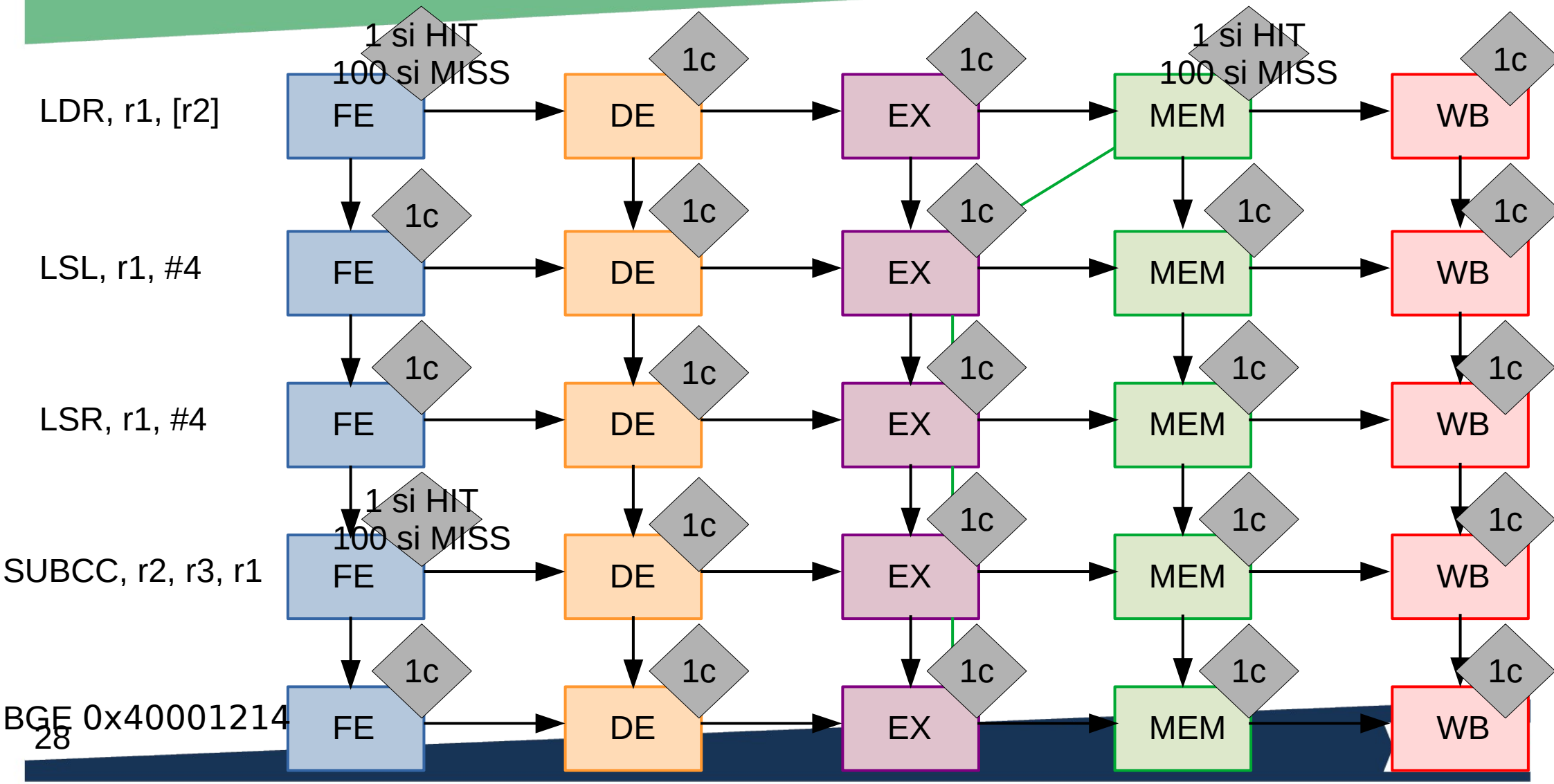
Modéliser le pipeline

Contraintes liées au code



Analyse WCET et OTAWA

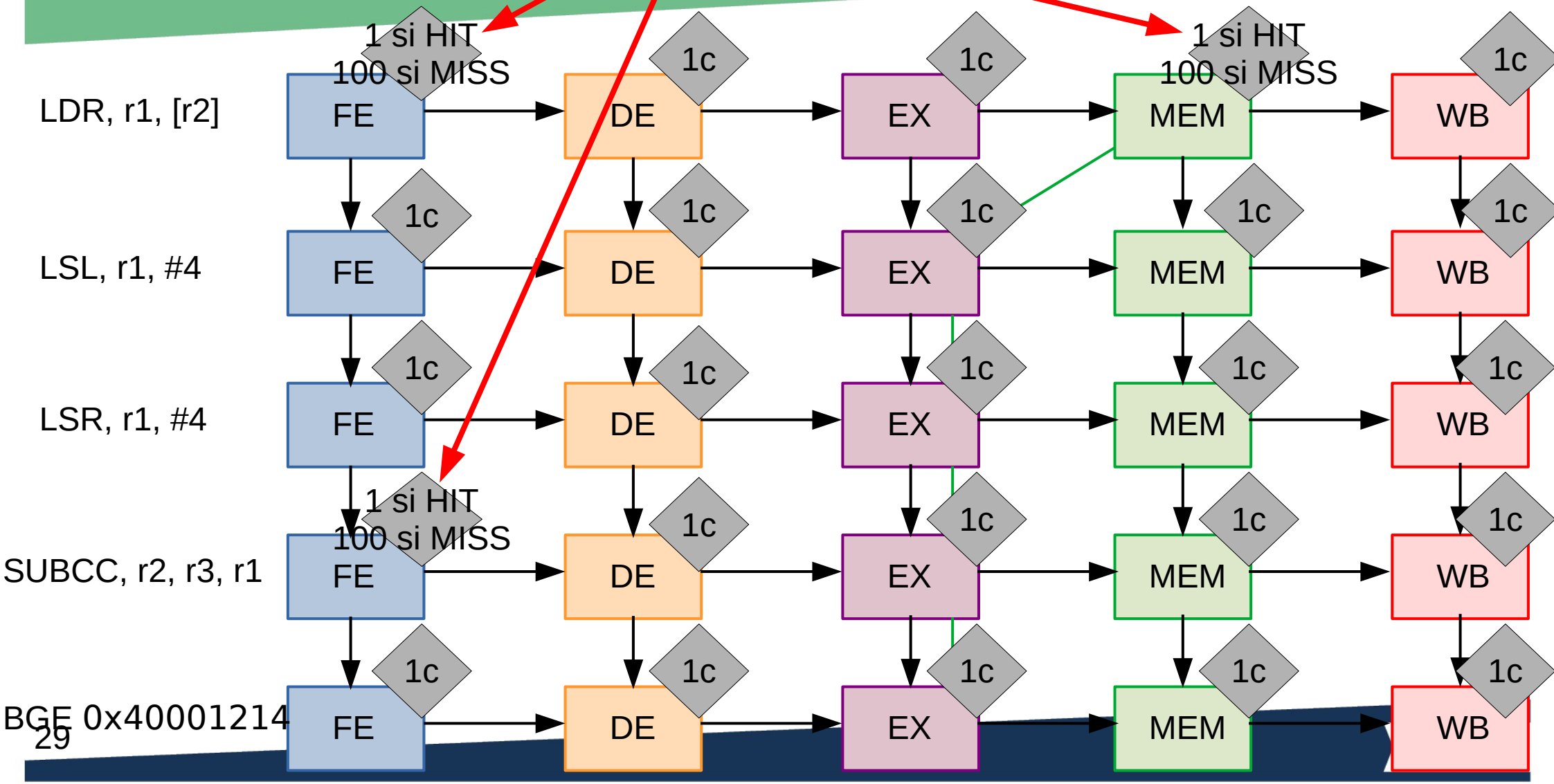
Modéliser le pipeline



Analyse WCET et OTAWA

Modéliser le pipeline

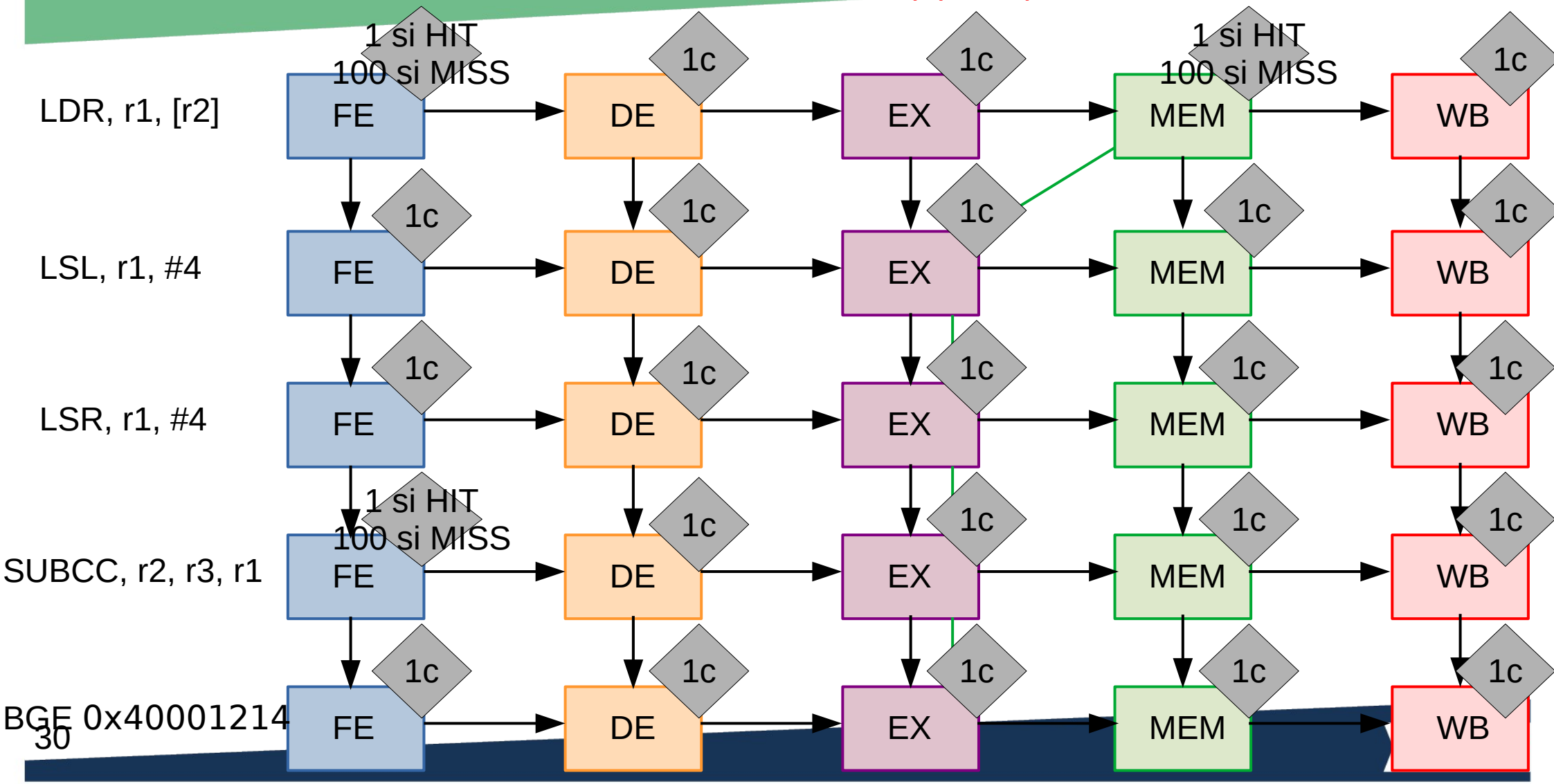
Evénements : 1 exegraph par combinaison
→ ici : 8 exegraphs pour représenter le BB



Analyse WCET et OTAWA

Modéliser le pipeline

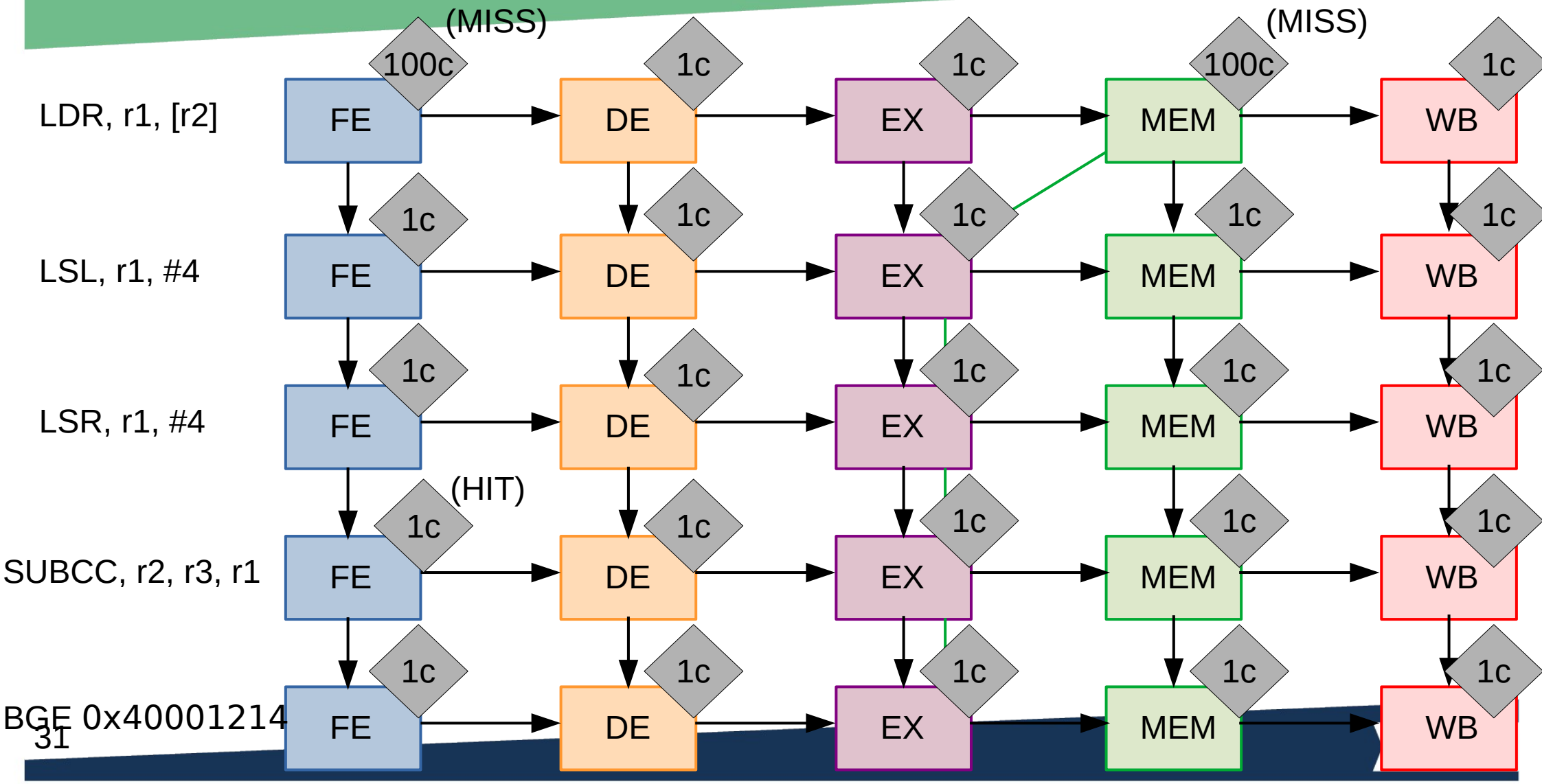
Evénements : autres types liés à
- des instructions à latence variable (e.g. FP)
- la prédiction de branchement (n'agit pas au niveau pipeline)



Analyse WCET et OTAWA

Modéliser le pipeline

Calcul du temps du BB pour cette combinaison d'événements :

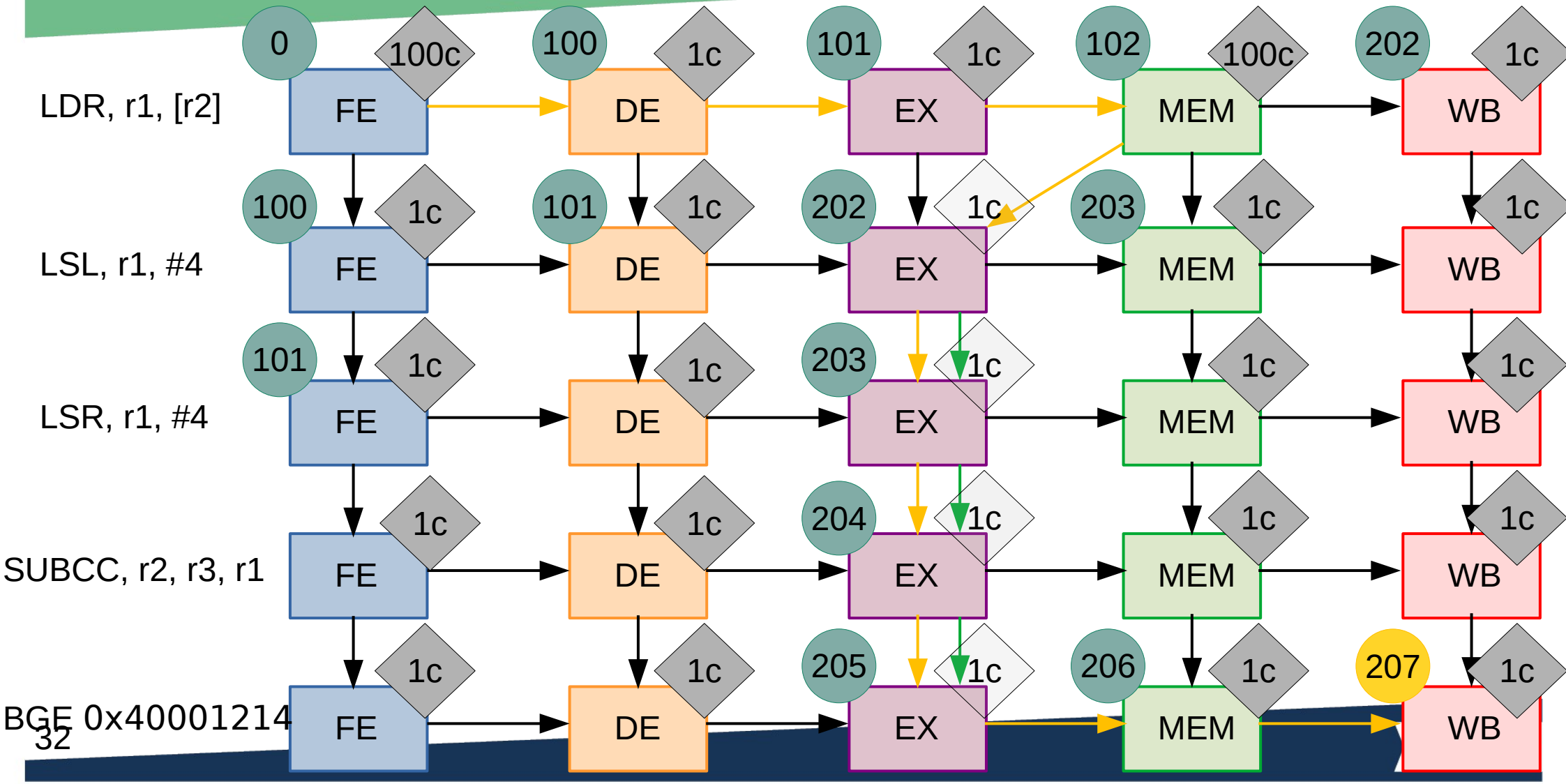


Analyse WCET et OTAWA

Modéliser le pipeline

Calcul du temps du BB pour cette combinaison d'événements :

→ Chemin critique



Analyse WCET et OTAWA
Quelques remarques

→ **1 temps de BB par combinaison possible d'événement**

- Pour le même BB, plusieurs $t_{i,c}$
- Possibilité d'introduire plusieurs variables $x_{i,c}$ pour le même BB i, afin de correspondre aux différents $t_{i,c}$ possibles de ce bloc

→ Capacité de « tendre » le WCET, à condition de pouvoir borner le nombre d'occurrences des événements, dans IPET

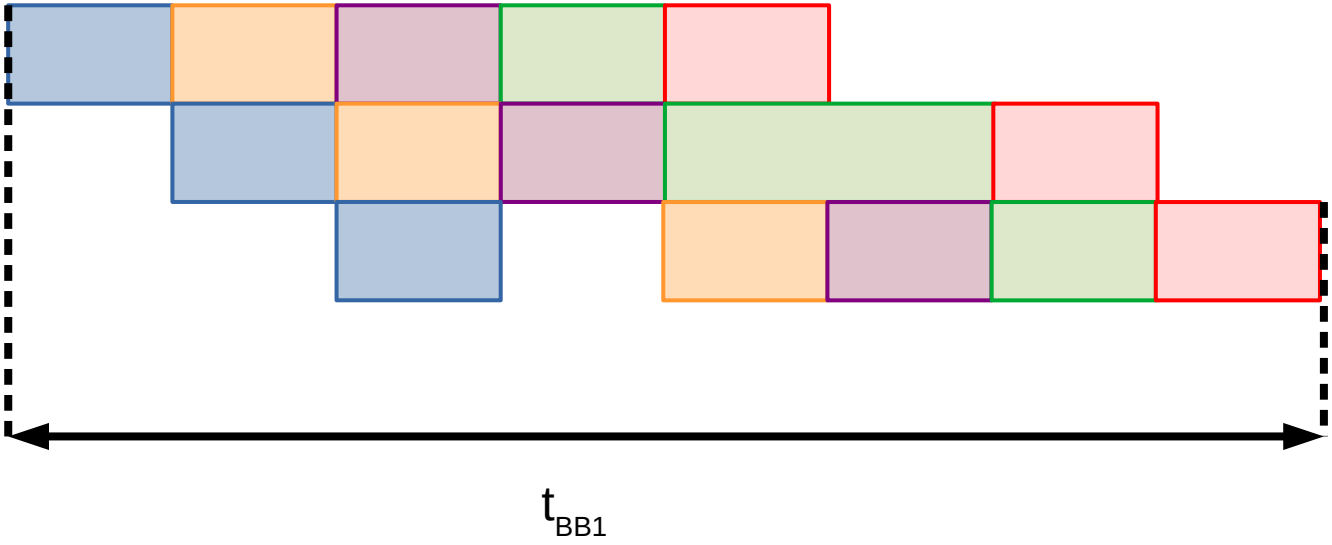
- Résultats des analyses statiques (e.g. l'un des LDR du BB fait un miss toutes les 5 exécutions du BB, le fetch de la 1ere instruction du BB ne fait un miss que la 1ere fois que le BB est exécuté, etc)

→ Trouver un compromis entre nombre de variables et précision du WCET

→ Propriétés d'amortissement du pipeline : plusieurs combinaisons d'événements peuvent mener au même temps

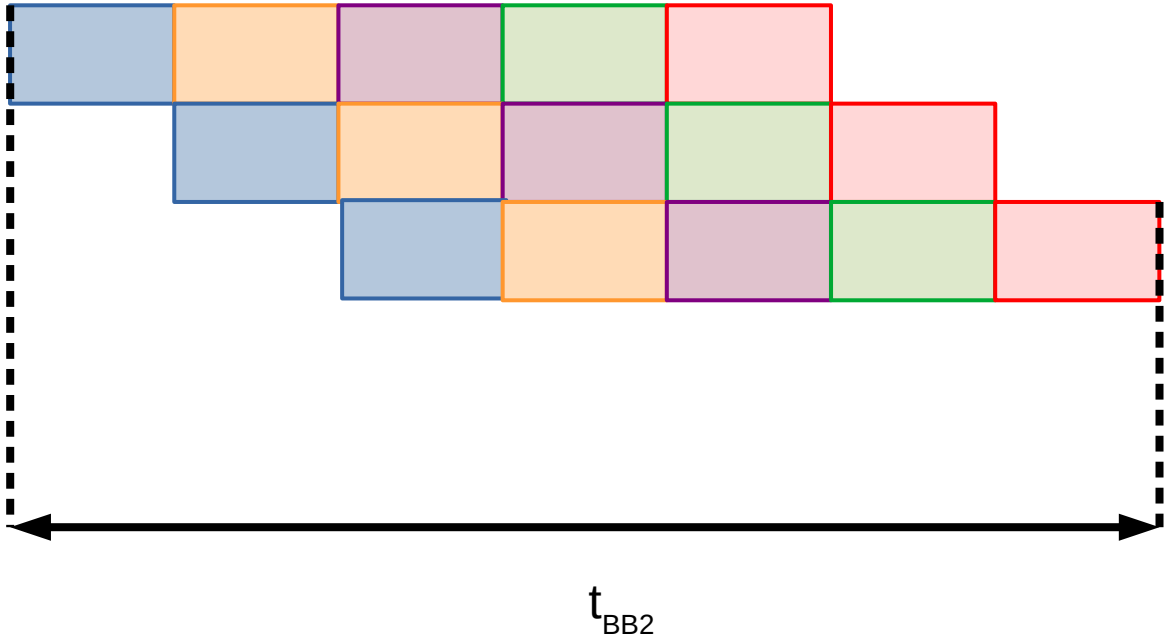
Analyse WCET et OTAWA
Quelques remarques

→ **Effet du pipeline entre 2 BBs consécutifs**



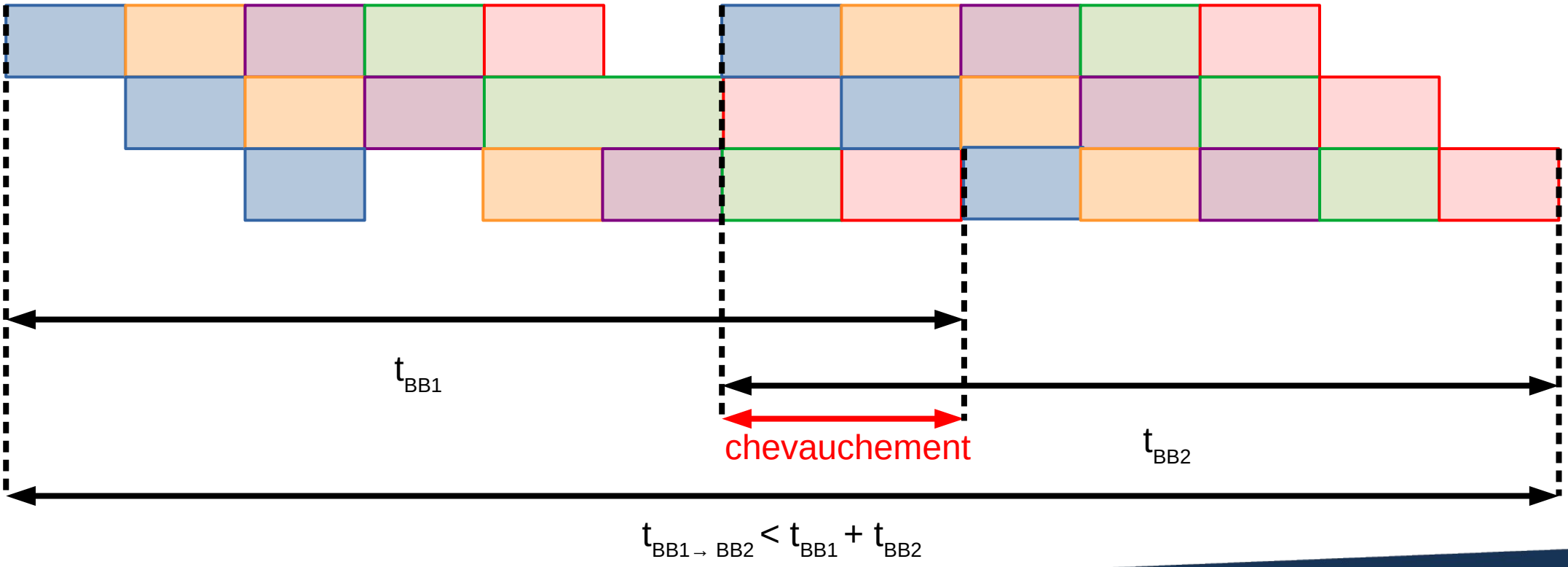
Analyse WCET et OTAWA
Quelques remarques

→ **Effet du pipeline entre 2 BBs consécutifs**



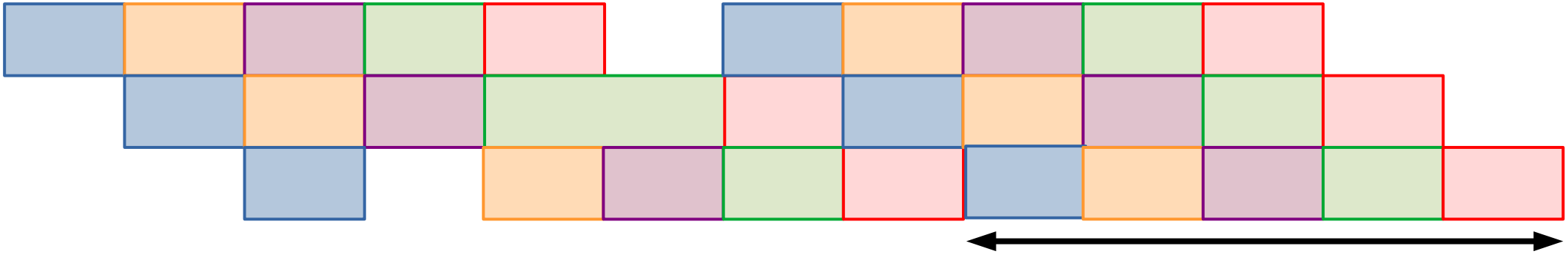
Analyse WCET et OTAWA
Quelques remarques

→ **Effet du pipeline entre 2 BBs consécutifs**



Analyse WCET et OTAWA
Quelques remarques

→ **Effet du pipeline entre 2 BBs consécutifs**



- On ne travaille en fait pas avec des t_{BB} (resp. x_{BB}) pour chaque BB, mais avec des $t_{BB1 \rightarrow BB2}$ (resp. $x_{BB1 \rightarrow BB2}$) pour chaque arc BB1→BB2 dans le CFG.

→ Dans l'exegraph de BB1→BB2 on rajoute à BB2 un prologue composé des instructions de BB1 pour estimer la durée sans le chevauchement.

Analyse WCET et OTAWA
Et les analyses statiques ?

→ **Nécessaires**
- Bornes de boucle

→ **Pour tendre le WCET**
- Cache d'instructions
- Cache de données
- Prédiction de branchement
- Chemins infaisables

→ **Interprétation abstraite au niveau du CFG**

~ Interpréteur du programme, qui n'émule pas le calcul pour un seul jeu d'entrées, mais qui sur-approxime l'ensemble des « configurations » atteignables pour toutes les exécutions possibles (e.g. à chaque point du programme :
- **toutes les valeurs qu'une variable pourrait avoir**
- **tous les états dans lequel le cache de données pourrait se trouver)**

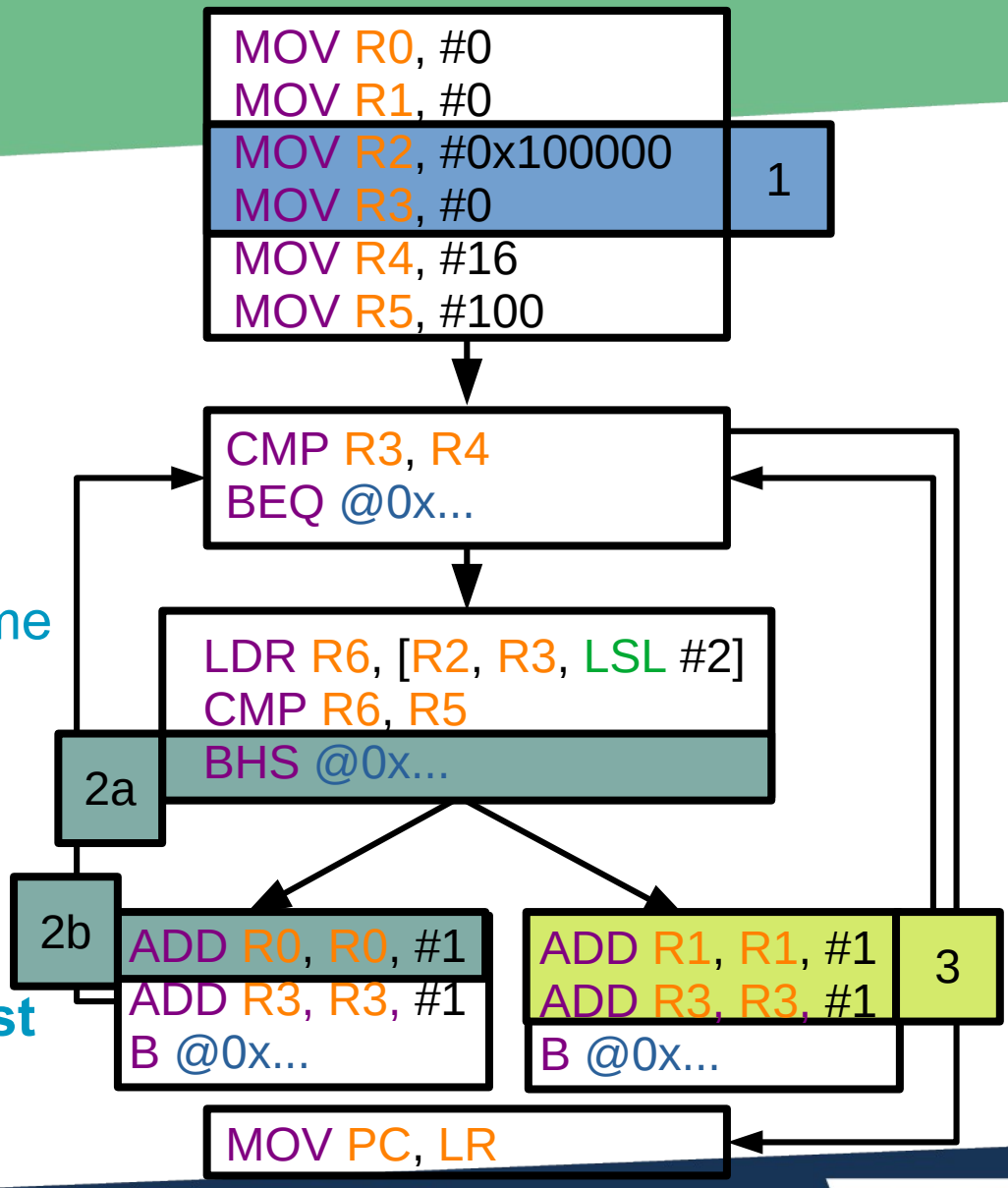
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

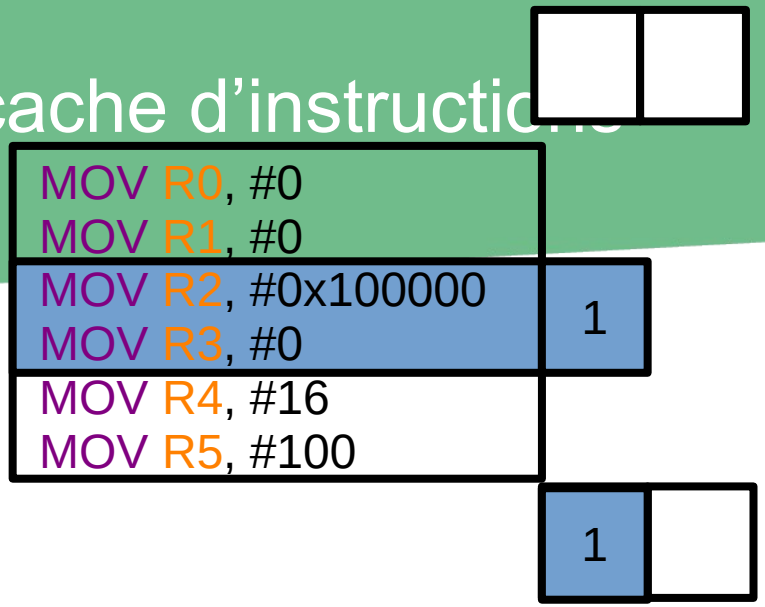
→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions



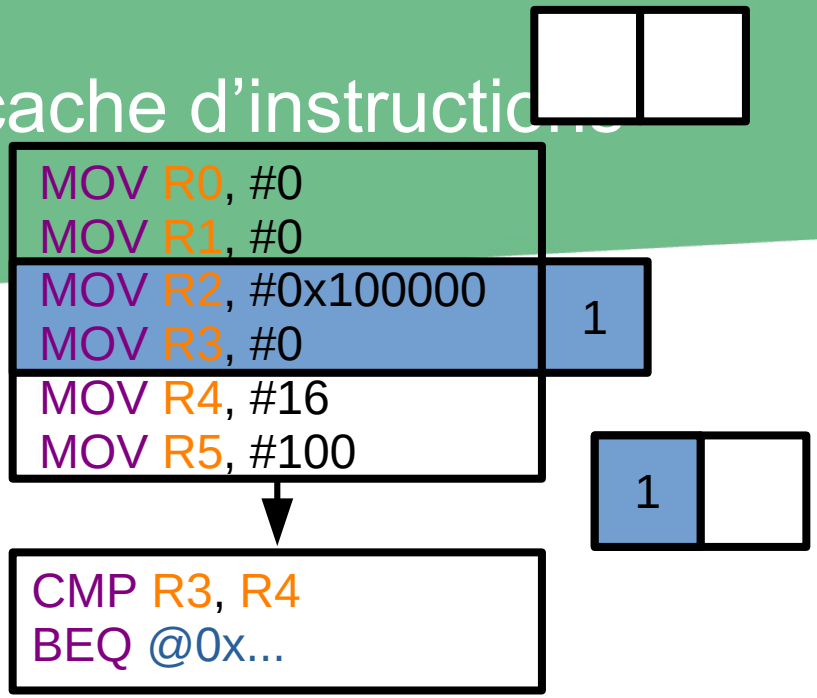
→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe

Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions



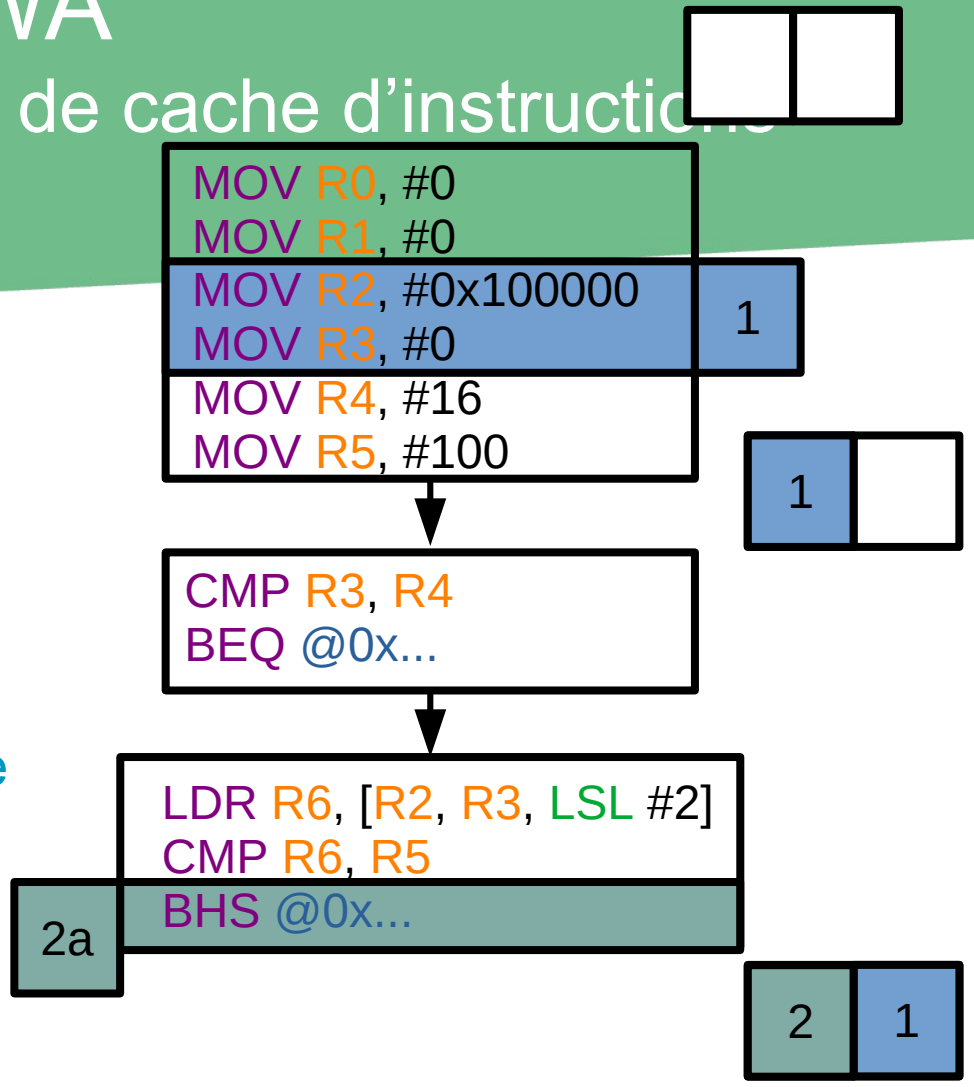
→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe

Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions



→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe

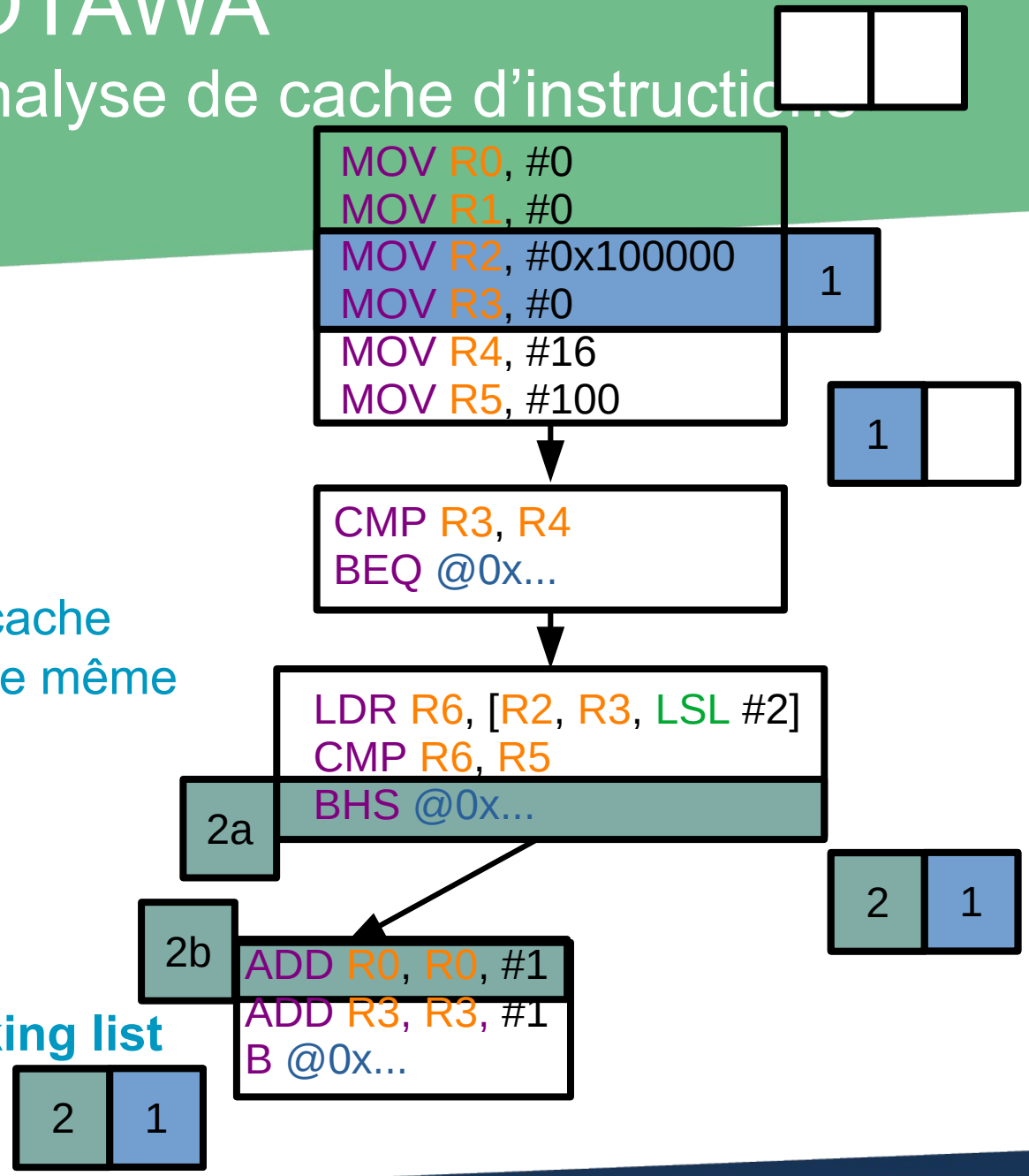
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



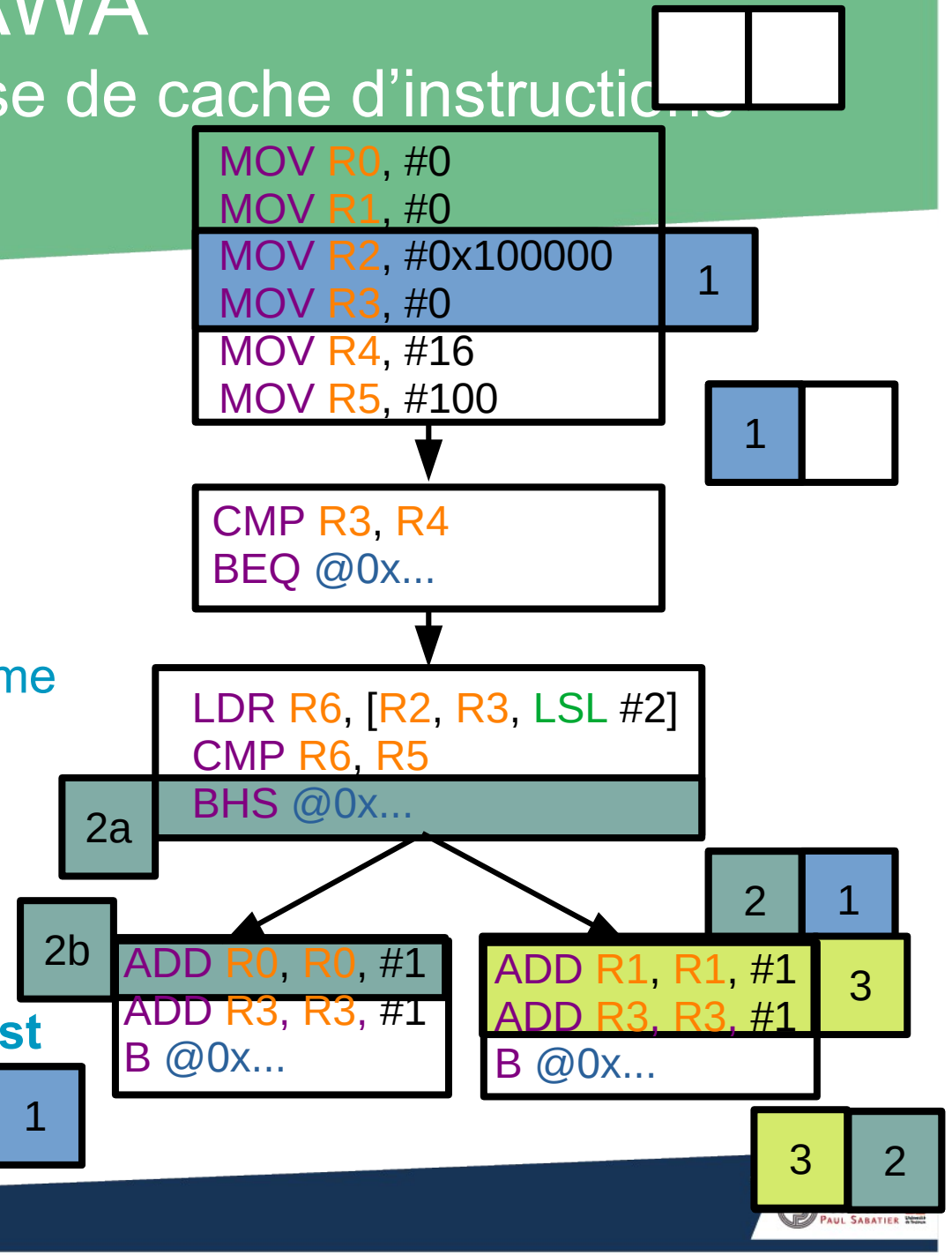
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



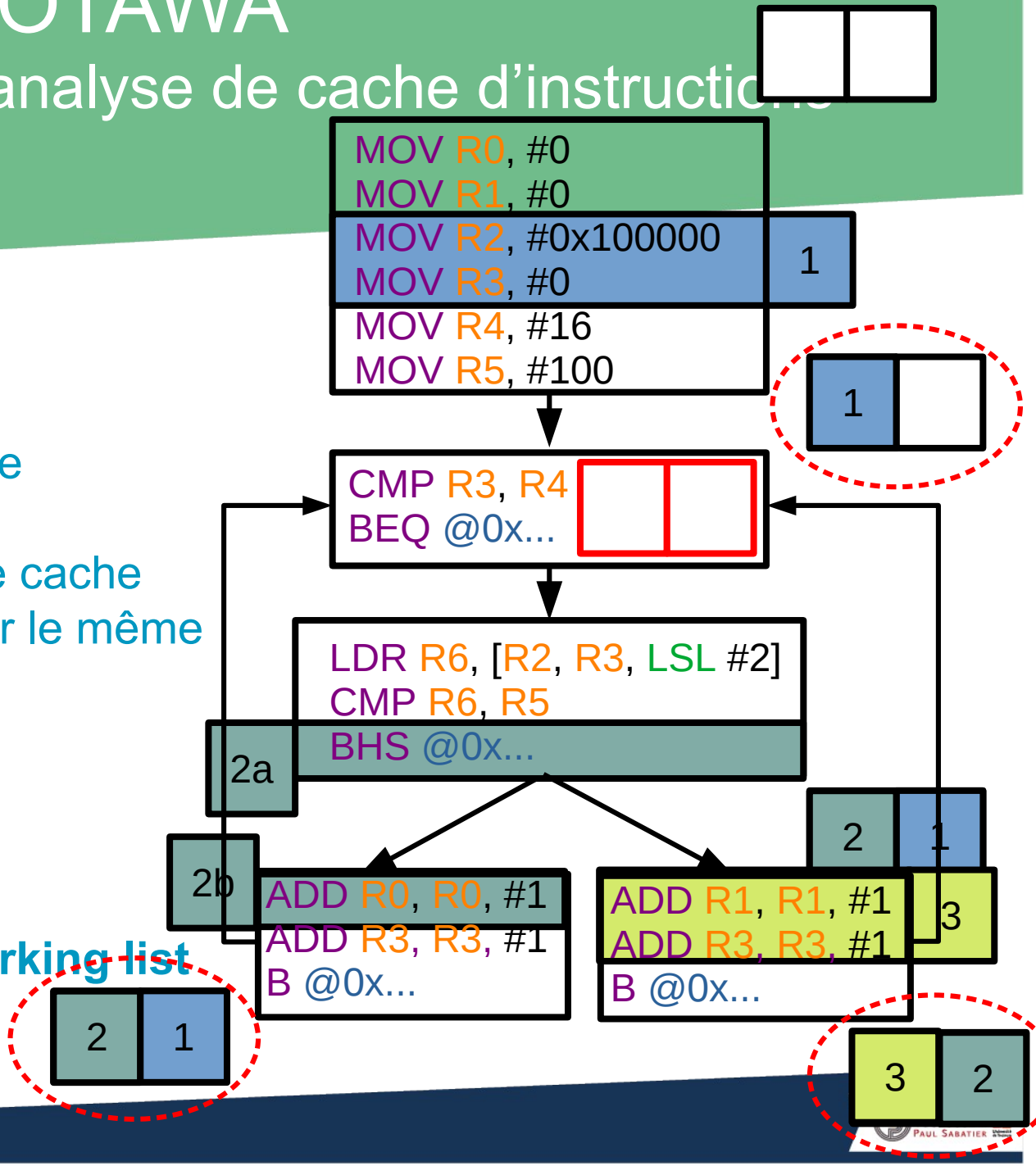
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



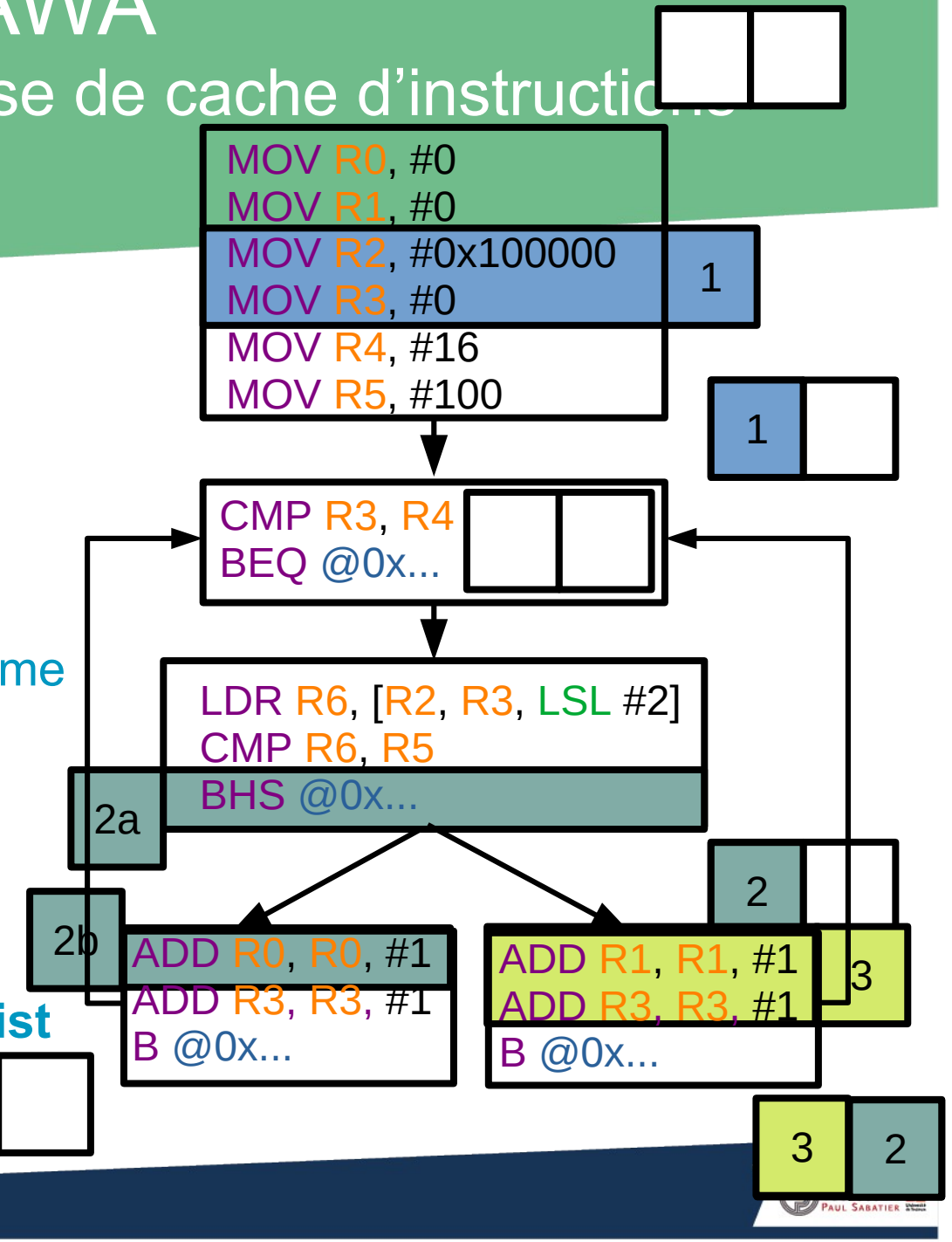
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



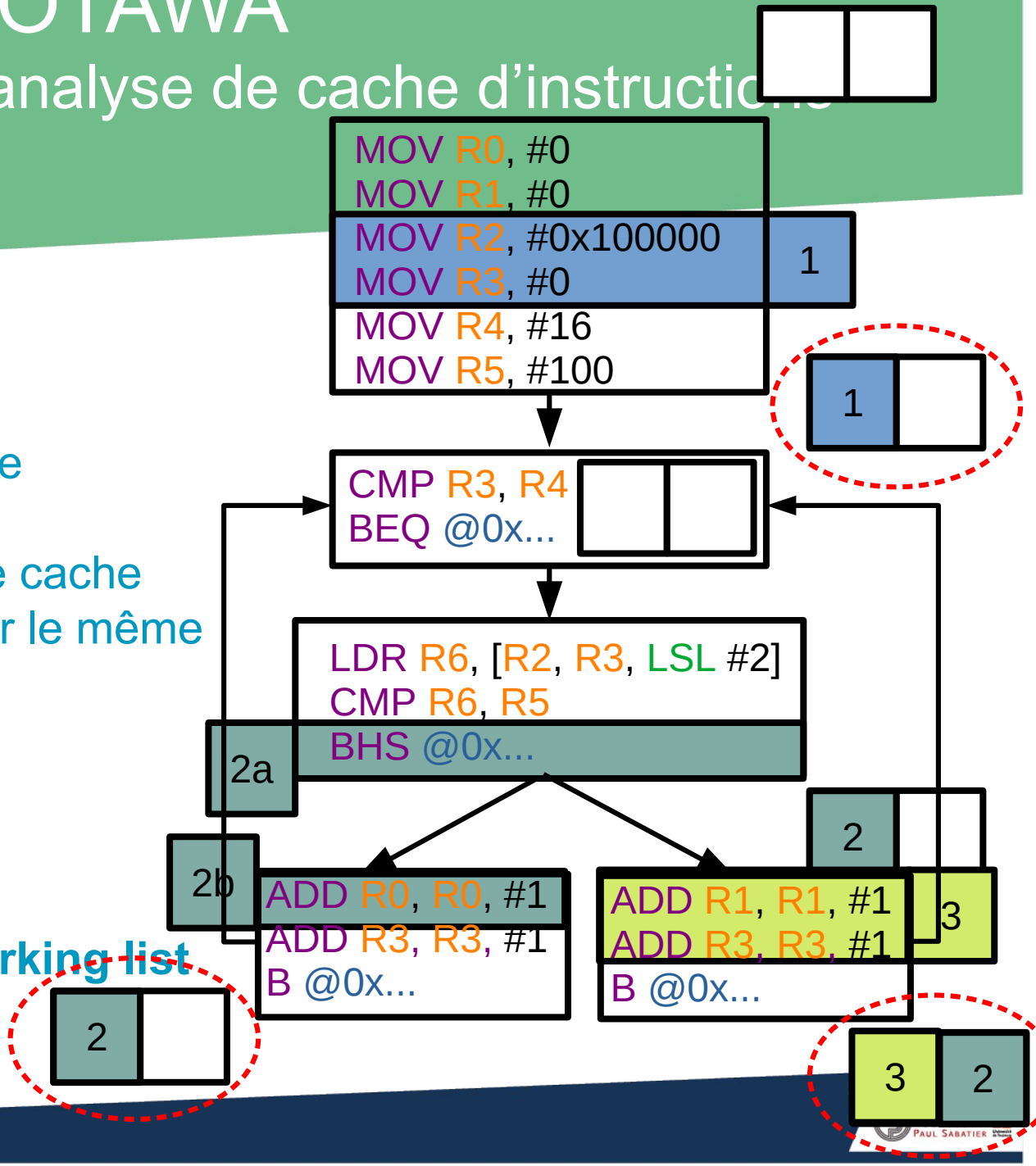
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



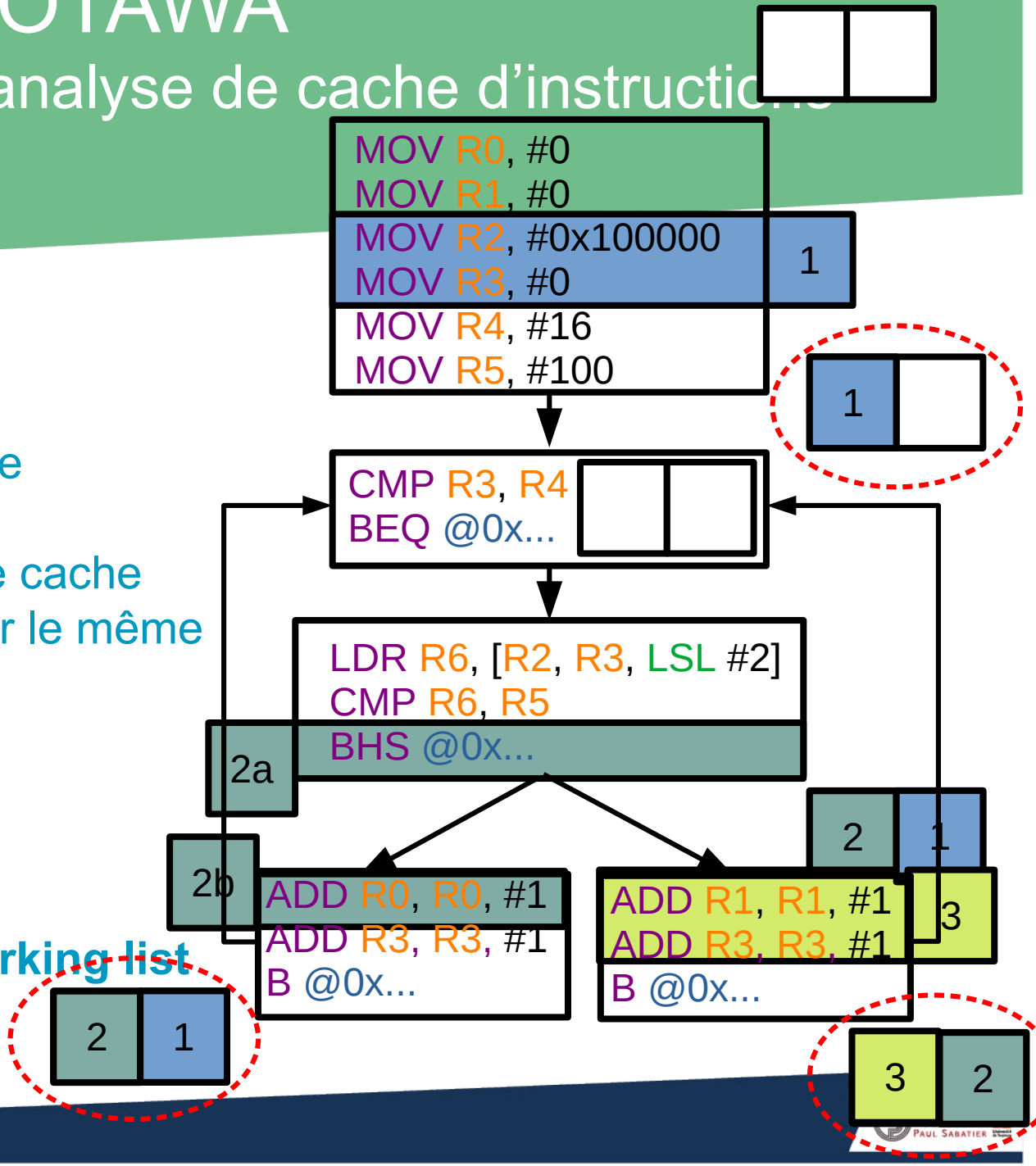
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



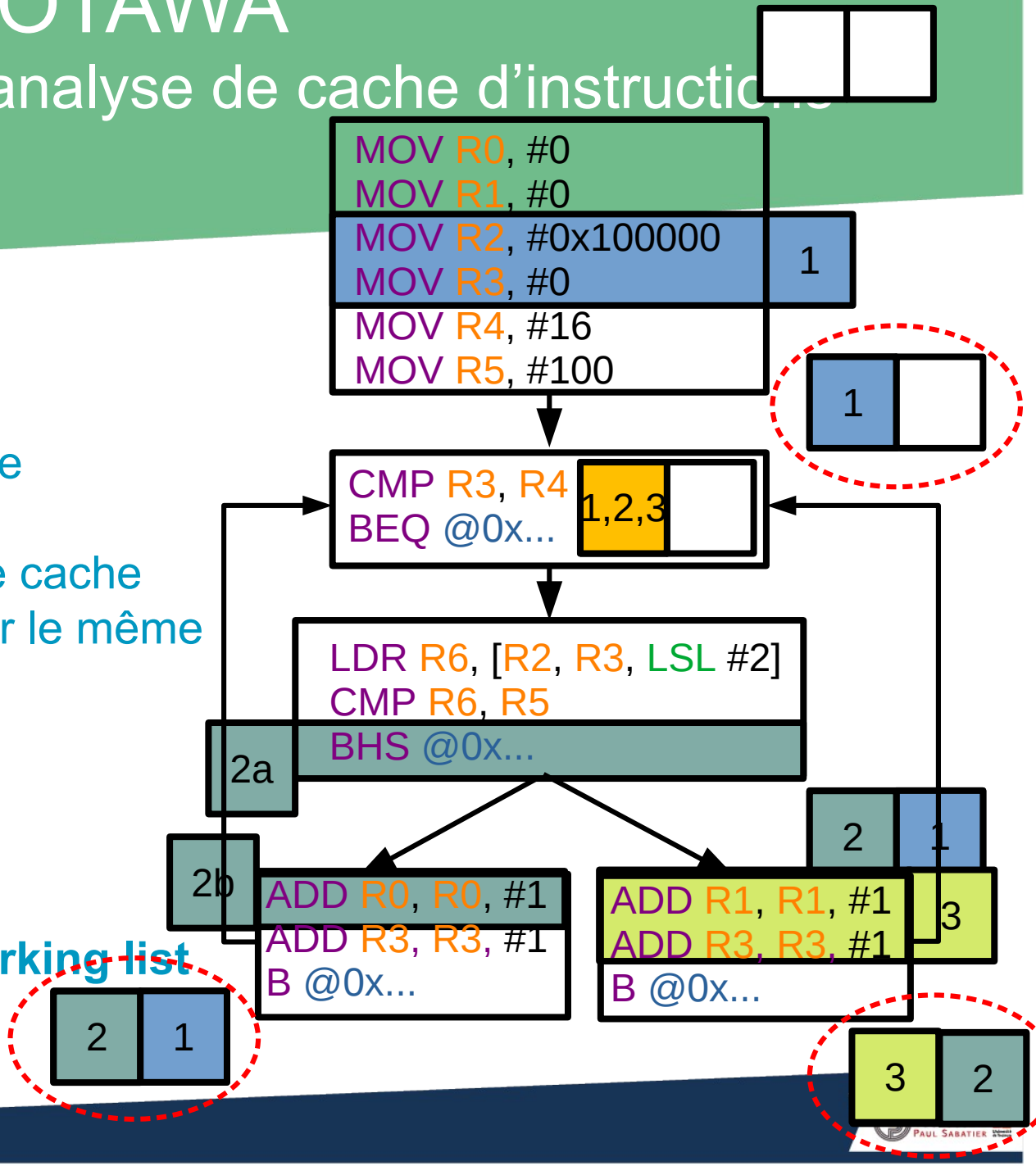
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**
- Recherche de point fixe



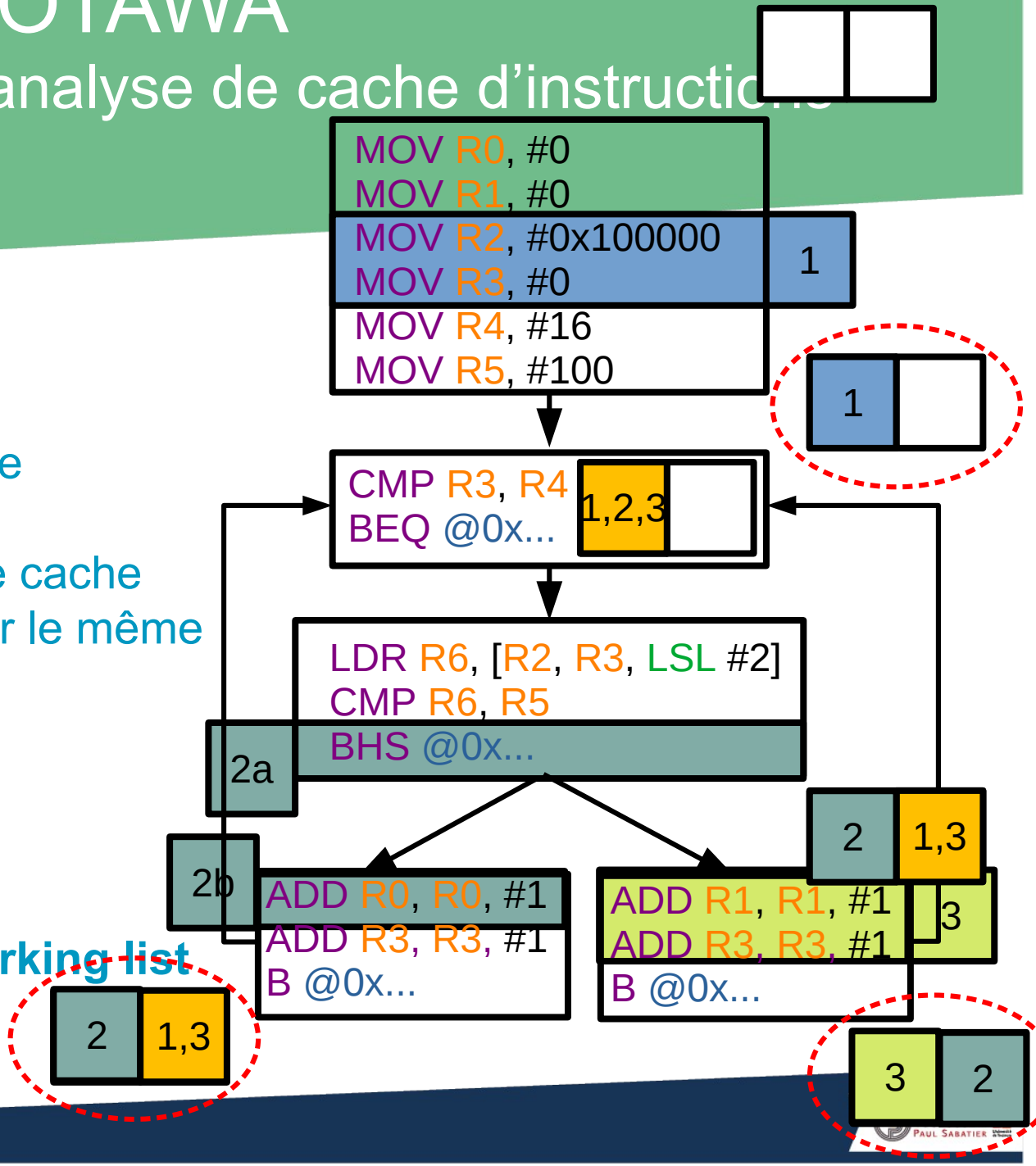
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Exemple :**
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**
- MUST
- MAY
- PERSISTENCE

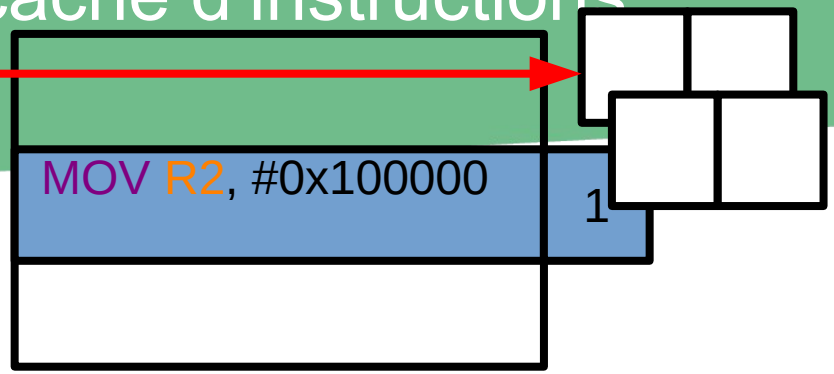
→ **Parcours du graphe par working list**
- Recherche de point fixe



Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

Absent du MAY : ALWAYS MISS
→ Exegraph : FE = 100c



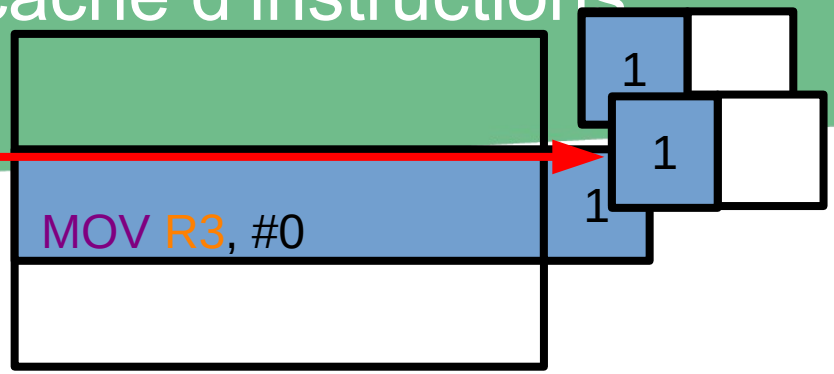
→ **Et après ?**

- En utilisant les résultats des analyses MAY et MUST, on classifie les accès mémoire :
  - Always HIT
  - Always MISS
  - Not Classified
- En utilisant les résultats de l'analyse PERSISTENCE, on obtient des informations pour borner des variables dans IPET

Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

Présent dans le MUST : ALWAYS HIT
→ Exegraph : FE = 1c

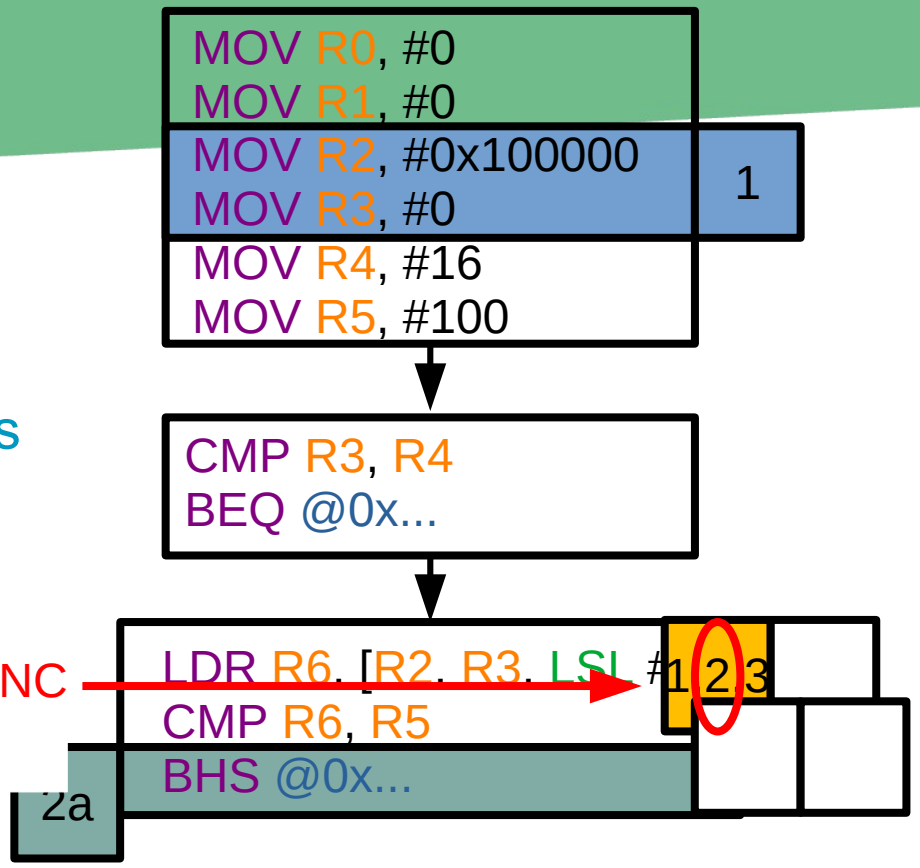


→ **Et après ?**

- En utilisant les résultats des analyses MAY et MUST, on classifie les accès mémoire :
  - Always HIT
  - Always MISS
  - Not Classified
- En utilisant les résultats de l'analyse PERSISTENCE, on obtient des informations pour borner des variables dans IPET

Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions



→ **Et après ?**

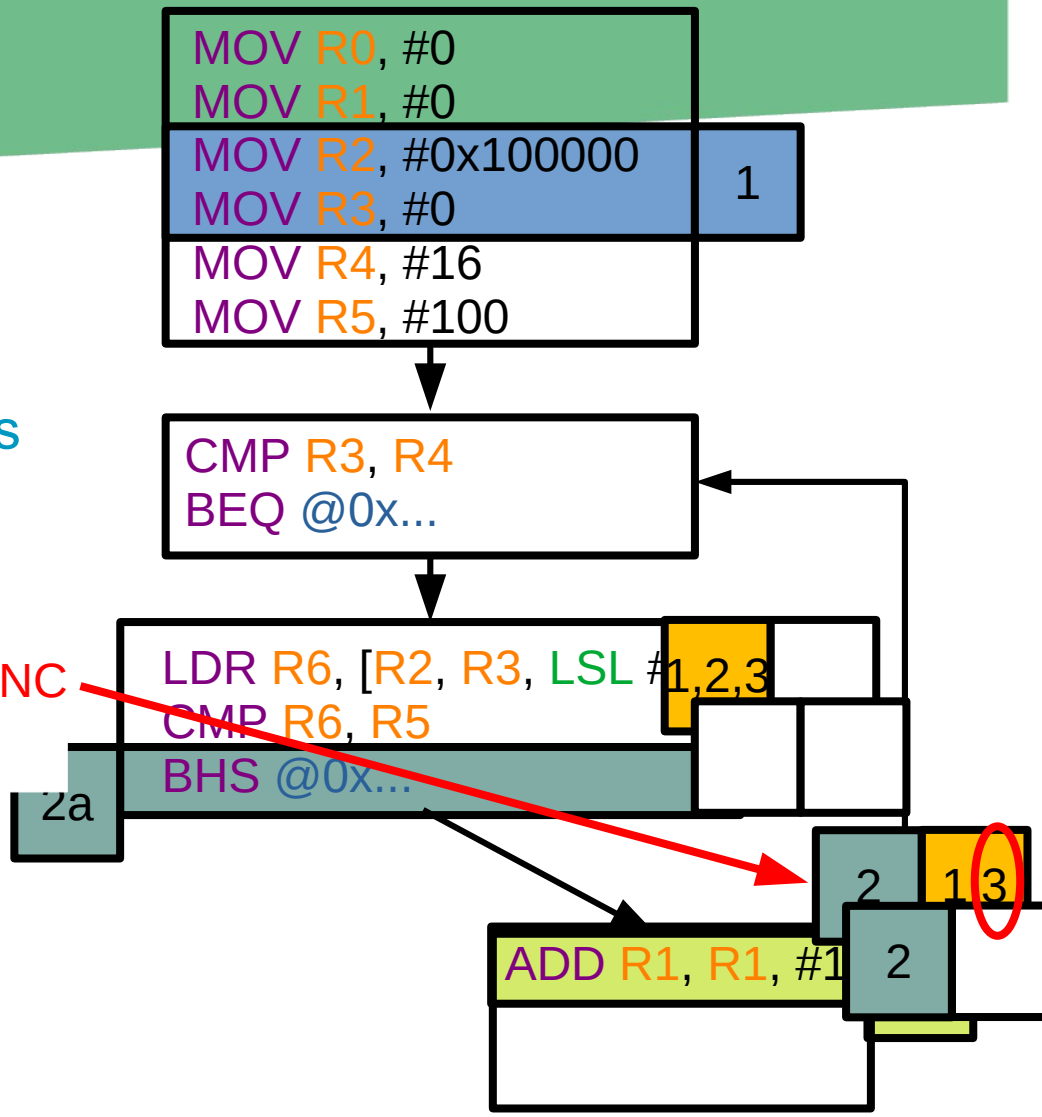
- En utilisant les résultats des analyses MAY et MUST, on classifie les accès mémoire :
  - Always HIT
  - Absent du MUST + présent dans MAY : NC
  - → Exegraph : FE a un événement
- En utilisant les résultats de l'analyse PERSISTENCE, on obtient des informations pour borner des variables dans IPET

Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Et après ?**

- En utilisant les résultats des analyses MAY et MUST, on classifie les accès mémoire :
  - Always HIT
  - Absent du MUST + présent dans MAY : NC
    → Exegraph : FE a un événement
- En utilisant les résultats de l'analyse PERSISTENCE, on obtient des informations pour borner des variables dans IPET



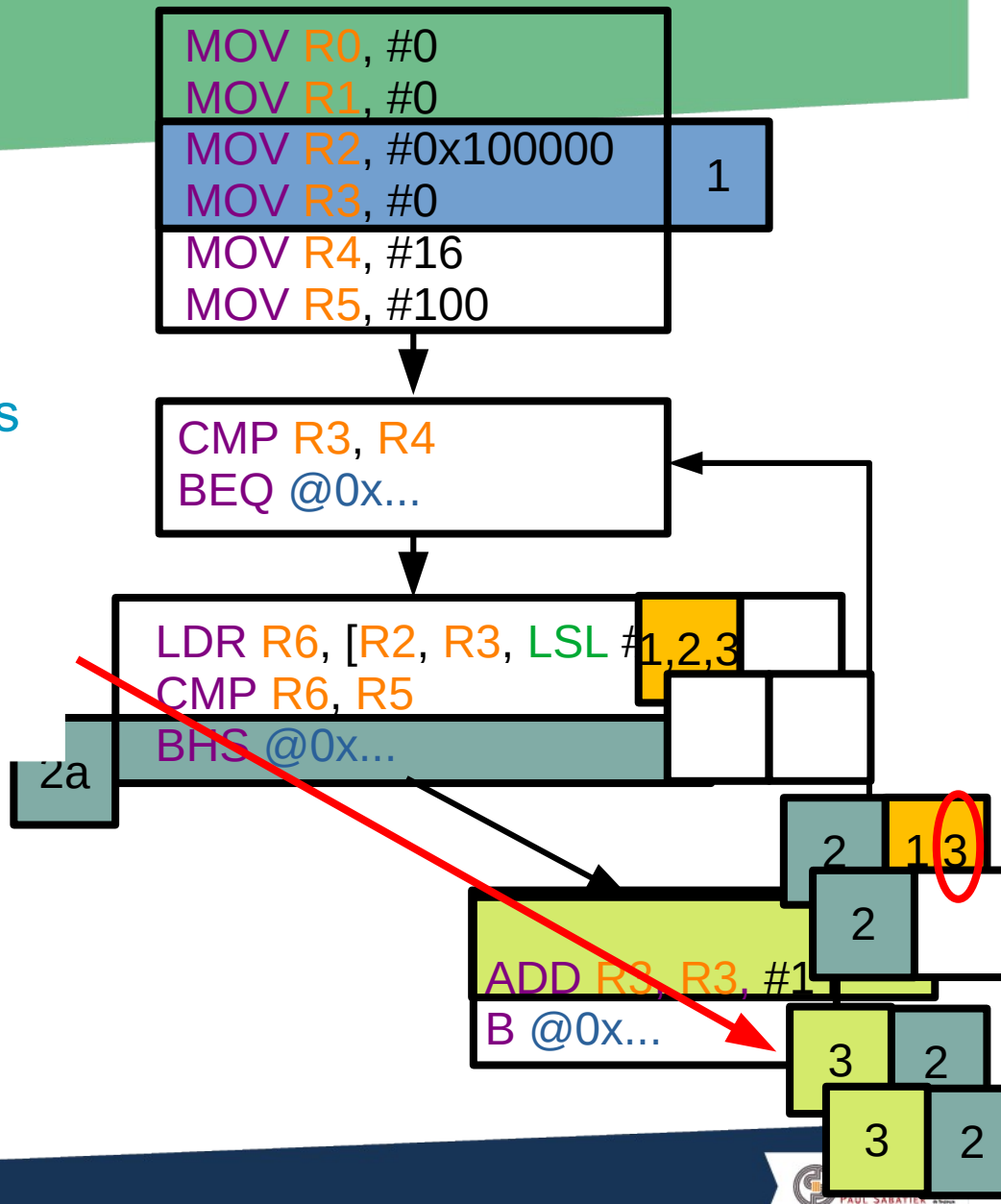
Analyse WCET et OTAWA

Illustration : un mot sur l'analyse de cache d'instructions

→ **Et après ?**

- En utilisant les résultats des analyses MAY et MUST, on classifie les accès mémoire :
  - Always HIT
    Présent dans le Must => Always Hit

- En utilisant les résultats de l'analyse PERSISTENCE, on obtient des informations pour borner des variables dans IPET



Analyse WCET et OTAWA

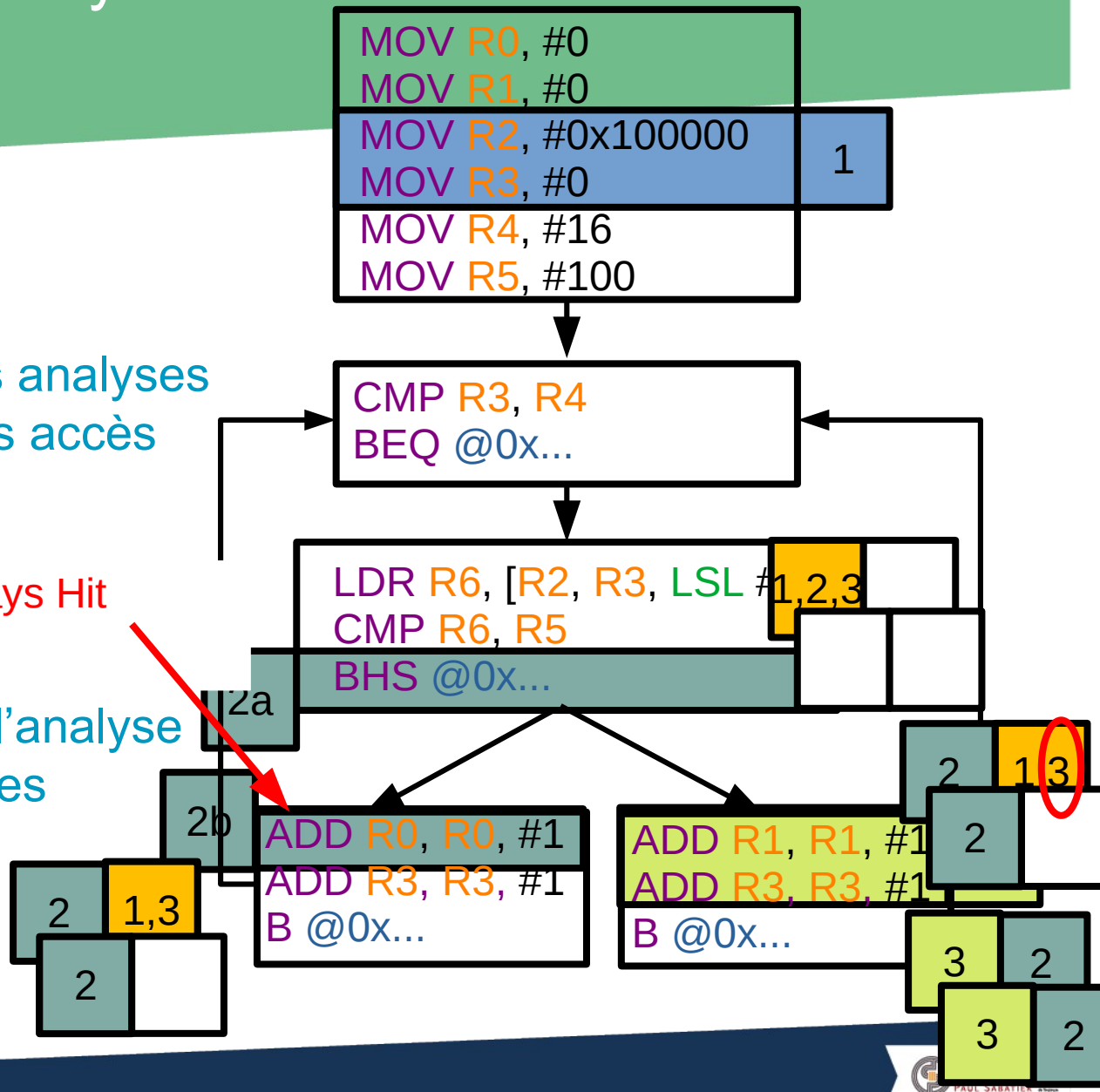
Illustration : un mot sur l'analyse de cache d'instructions

→ **Et après ?**

- En utilisant les résultats des analyses MAY et MUST, on classifie les accès mémoire :
  - Always HIT

    Présent dans le Must => Always Hit

- En utilisant les résultats de l'analyse PERSISTENCE, on obtient des informations pour borner des variables dans IPET



Analyse WCET et OTAWA
OTAWA - OWCET

→ **Framework OTAWA**

- Un analyseur WCET (Owcet) incluant des analyses State-of-the-Art et des modèles d'architectures embarquées (basées sur ISA ARM, PowerPC, Aurix TriCore, Kalray, Sparc/Leon, RiscV – et bientôt GPU Nvidia Pascal !)
- Des outils d'analyse de code (désassembleurs, analyses statiques)
- Des bibliothèques pour développer ses propres analyses
- Open source

WCET ET ORDONNANCEMENT : PROBLEMES OUVERTS

Cache Related Preemption Delays

→ **Multithread/multiprocess avec préemptions**

- Modèle classique e.g. Liu & Layland tâches périodiques

→ Critère suffisant d'ordonnançabilité pour RM (exact pour EDF)

→ Hypothèse : préemption « 0 coût » (a.k.a. coût de préemption dans les Ci)

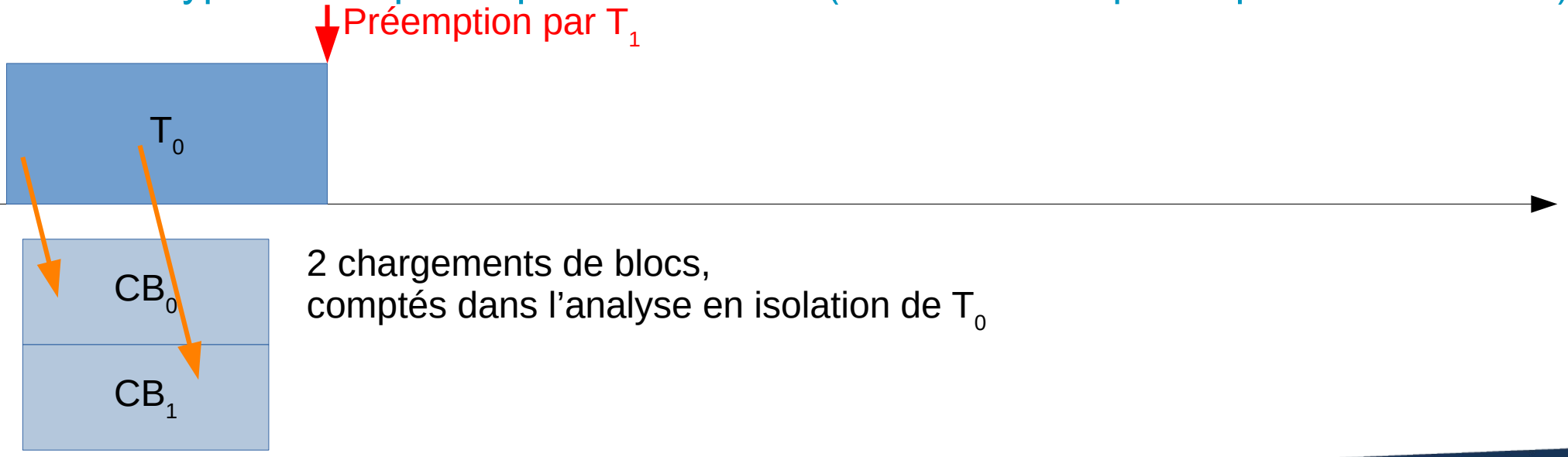
Cache Related Preemption Delays

→ **Multithread/multiprocess avec préemptions**

- Modèle classique e.g. Liu & Layland tâches périodiques

→ Critère suffisant d'ordonnançabilité pour RM (exact pour EDF)

→ Hypothèse : preemption « 0 coût » (a.k.a. coût de preemption dans les Ci)



2 chargements de blocs,
comptés dans l'analyse en isolation de T_0

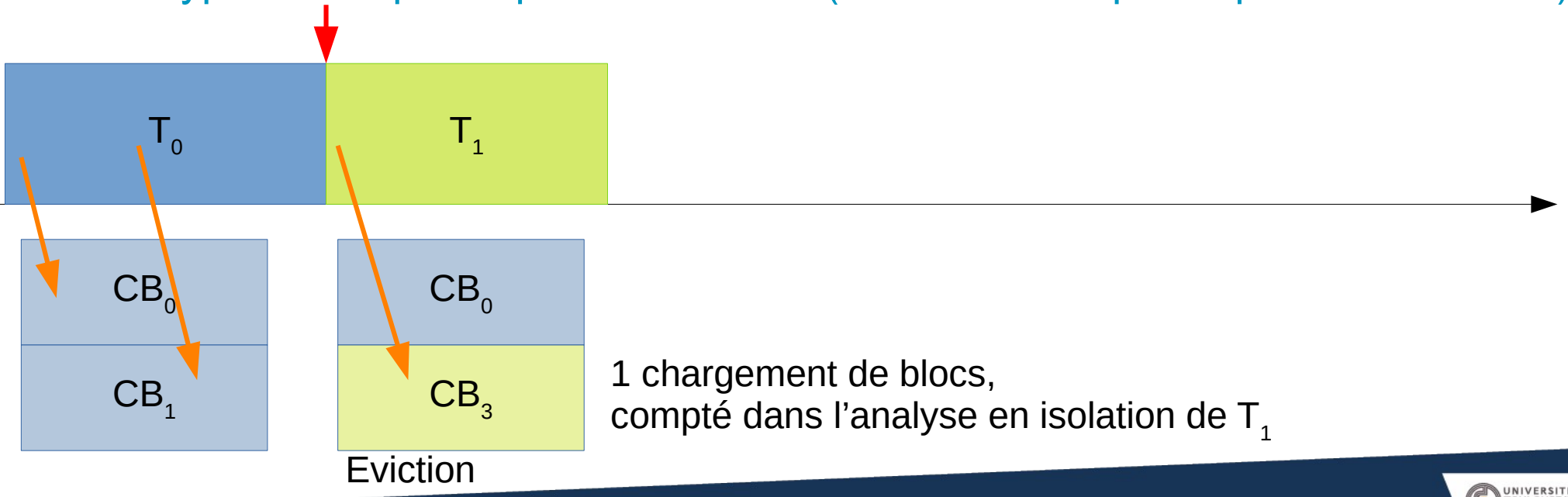
Cache Related Preemption Delays

→ **Multithread/multiprocess avec préemptions**

- Modèle classique e.g. Liu & Layland tâches périodiques

→ Critère suffisant d'ordonnançabilité pour RM (exact pour EDF)

→ Hypothèse : preemption « 0 coût » (a.k.a. coût de preemption dans les Ci)



1 chargement de blocs,
compté dans l'analyse en isolation de T_1

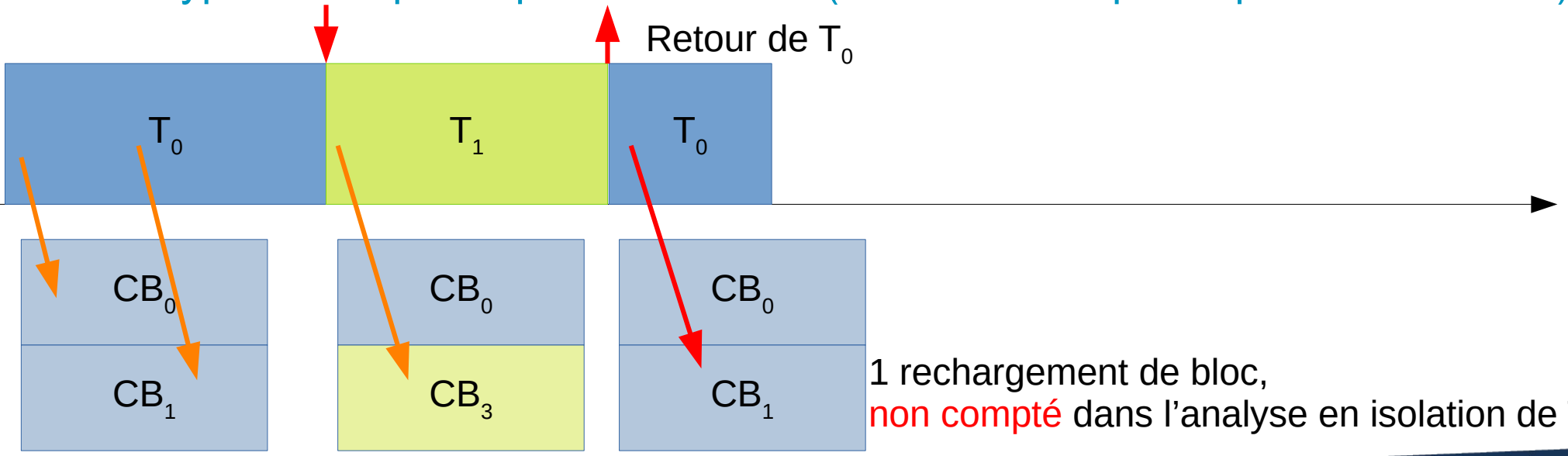
Cache Related Preemption Delays

→ **Multithread/multiprocess avec préemptions**

- Modèle classique e.g. Liu & Layland tâches périodiques

→ Critère suffisant d'ordonnançabilité pour RM (exact pour EDF)

→ Hypothèse : preemption « 0 coût » (a.k.a. coût de preemption dans les Ci)



1 rechargement de bloc,
non compté dans l'analyse en isolation de T_0

Cache Related Preemption Delays

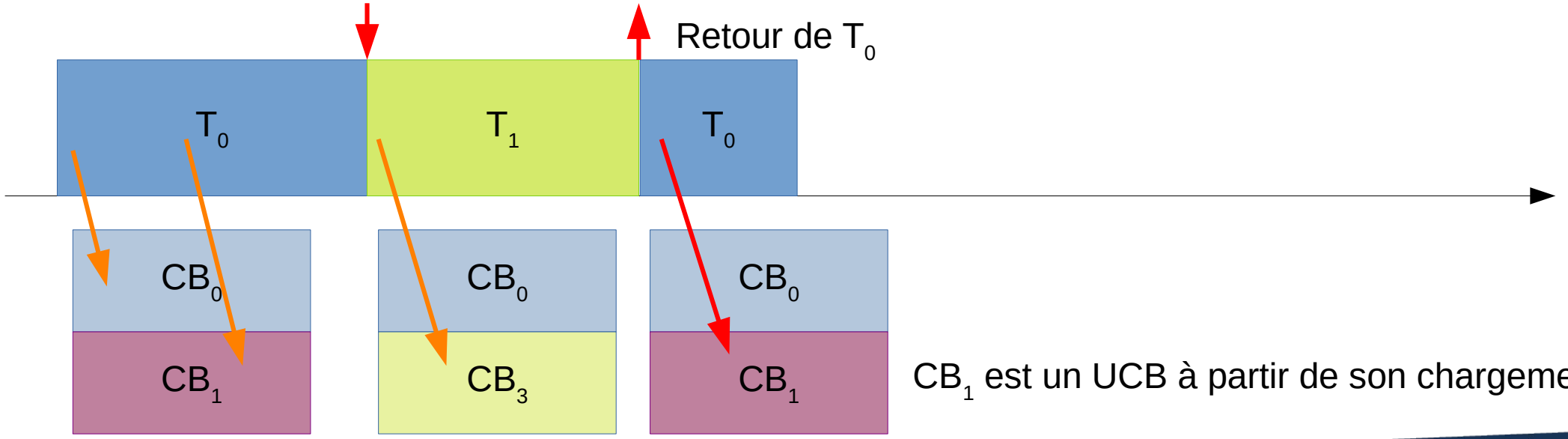
→ **Multithread/multiprocess avec préemptions**

- Pour tout point de la tâche, par interprétation abstraite :
  - Calcul des blocs (potentiellement) évincés dont la tâche aura (potentiellement) besoin à nouveau : UCB
    → Bloc présent dans MAY à un point donné, et ultérieurement lors d'un accès
  - Calcul des blocs (potentiellement) chargés par la tâche préemptrice: ECB
    → Bloc présent dans MAY à n'importe quel point de la tâche

  - Croisement des blocs UCB et ECB, suivant la politique du cache

Cache Related Preemption Delays

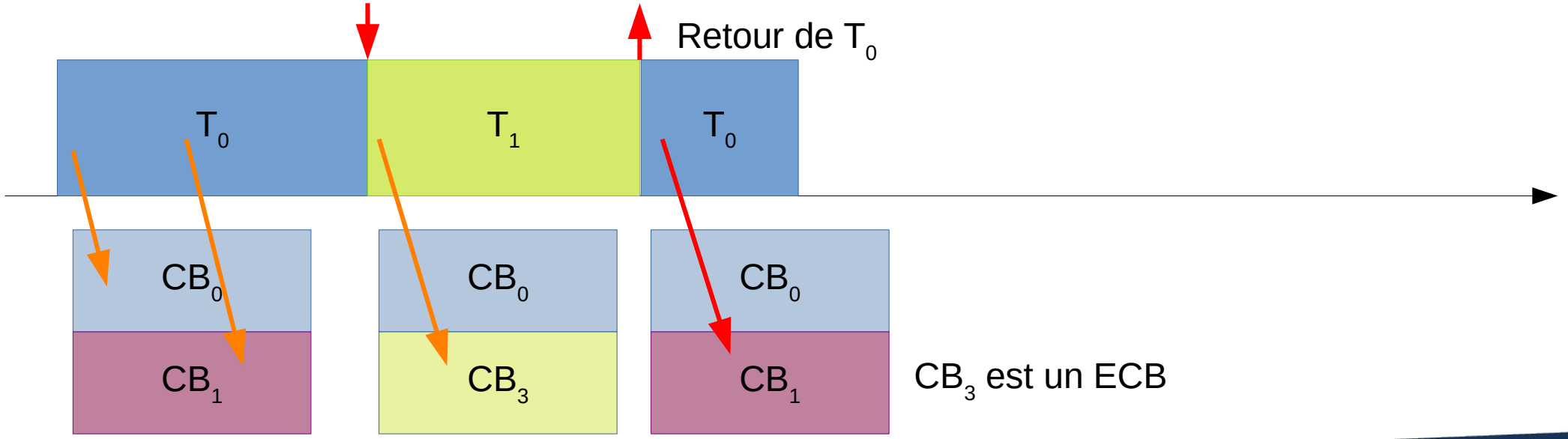
→ **Multithread/multiprocess avec préemptions**



CB_1 est un UCB à partir de son chargement

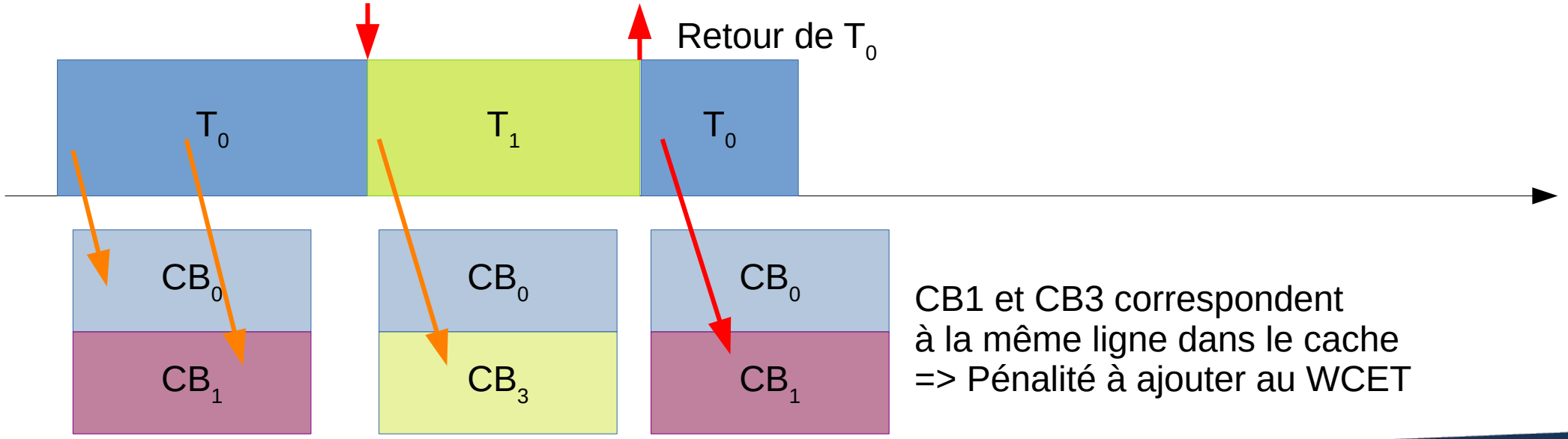
Cache Related Preemption Delays

→ **Multithread/multiprocess avec préemptions**



Cache Related Preemption Delays

→ **Multithread/multiprocess avec préemptions**



CB1 et CB3 correspondent
à la même ligne dans le cache
=> Pénalité à ajouter au WCET

Cache Related Preemption Delays

→ **Multithread/multiprocess avec préemptions**

- Sélection du maximum de pénalité sur la totalité de la tâche et ajout dans le WCET de base

  → Très imprécis => surestimation forte

- Mitigation possible : points de préemption

  → plus précis, moins flexible (équilibre statique vs dynamique)

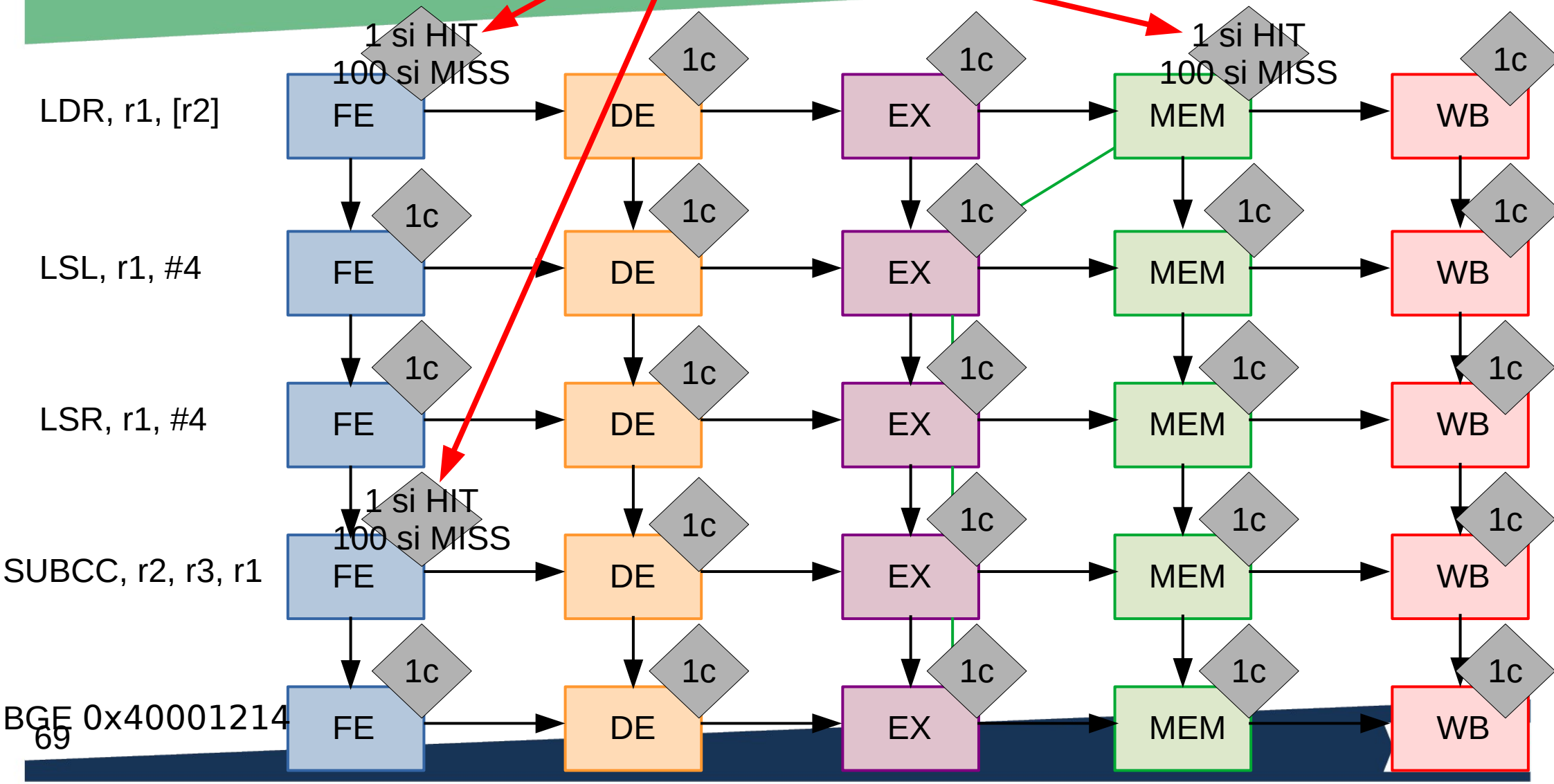
Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- Explosion combinatoire

Timing anomalies

Evénements : 1 exegraph par combinaison
→ ici : 8 exegraphs pour représenter le BB



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- Explosion combinatoire

→ Approche gloutonne : le pire cas global comme combinaison des pire cas locaux

  - ex : acces cache NC → cache miss, spéculation

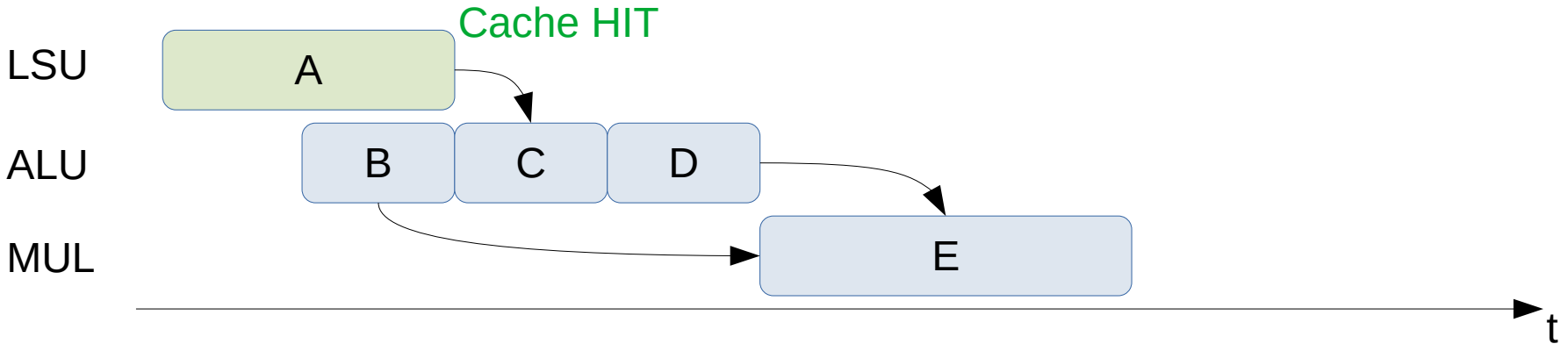
- Problème : l'approche gloutonne ne mène pas au pire cas global

  - ex : multiscalaire out-of-order

Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

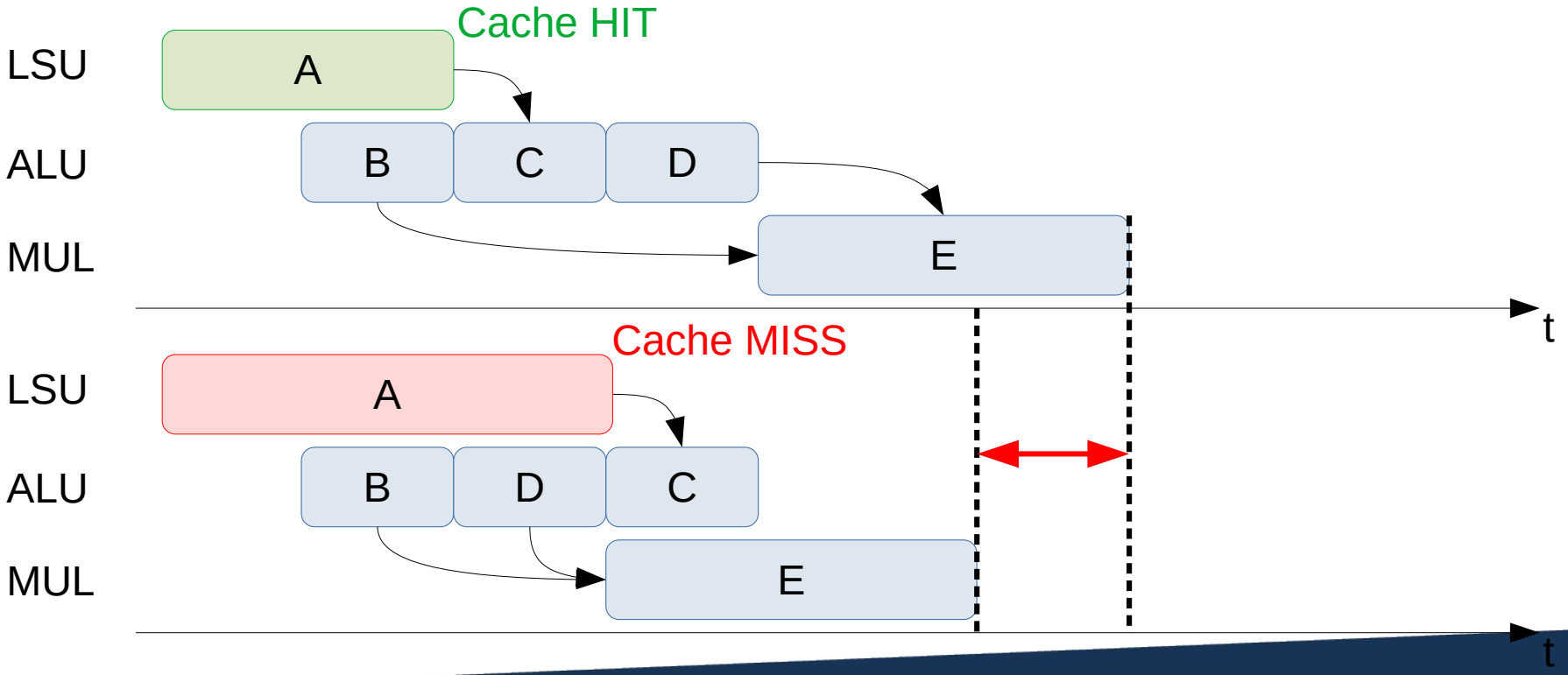
- ex : multiscalaire out-of-order



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- ex : multiscalaire out-of-order



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- Explosion combinatoire

→ Approche gloutonne : le pire cas global comme combinaison des pire cas locaux

  - ex : acces cache NC → cache miss

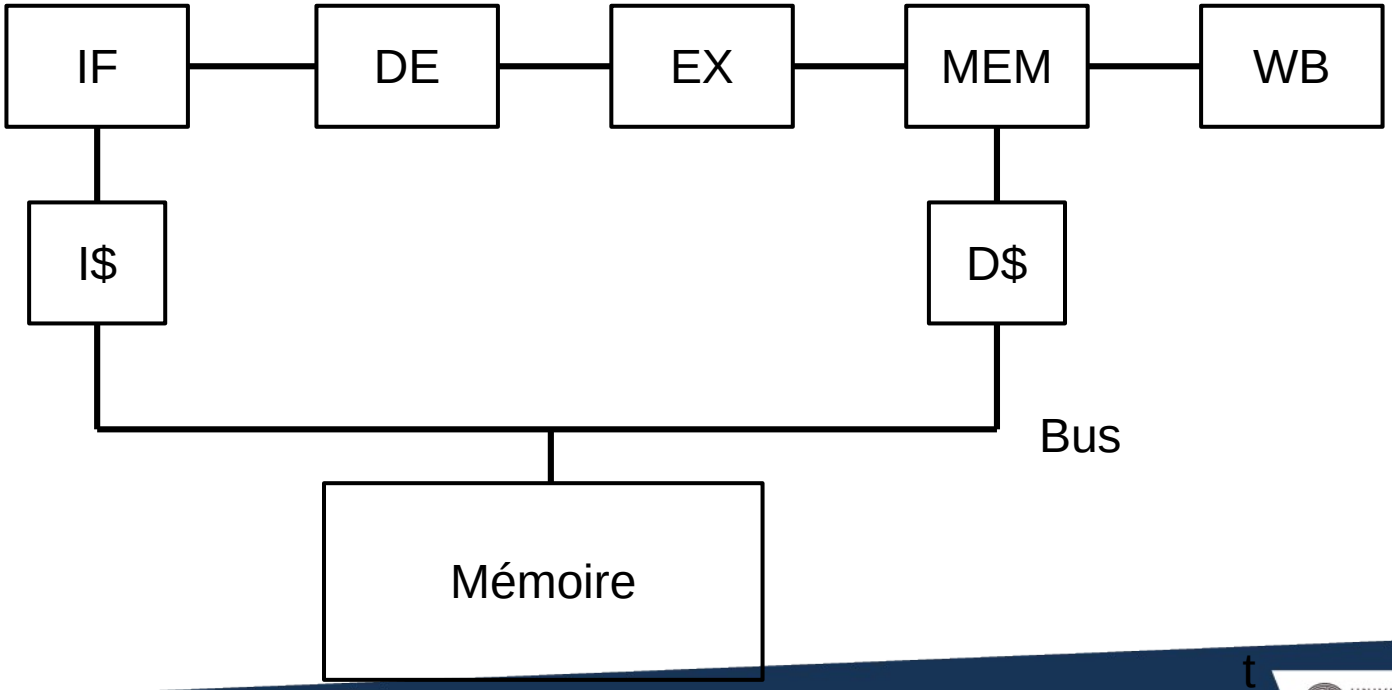
- Problème : l'approche gloutonne ne mène pas au pire cas global

  - ex : multiscalaire out-of-order
  - ex : monoscalaire in-order

Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- ex : monoscalaire in-order

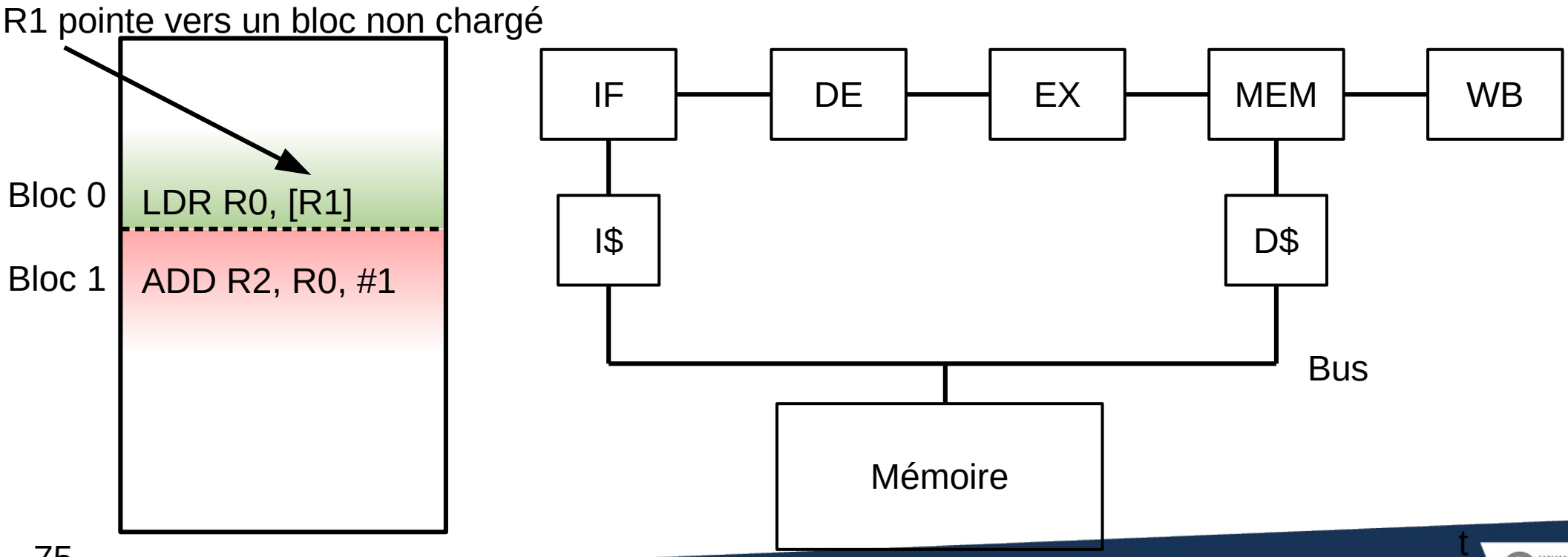


t

Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

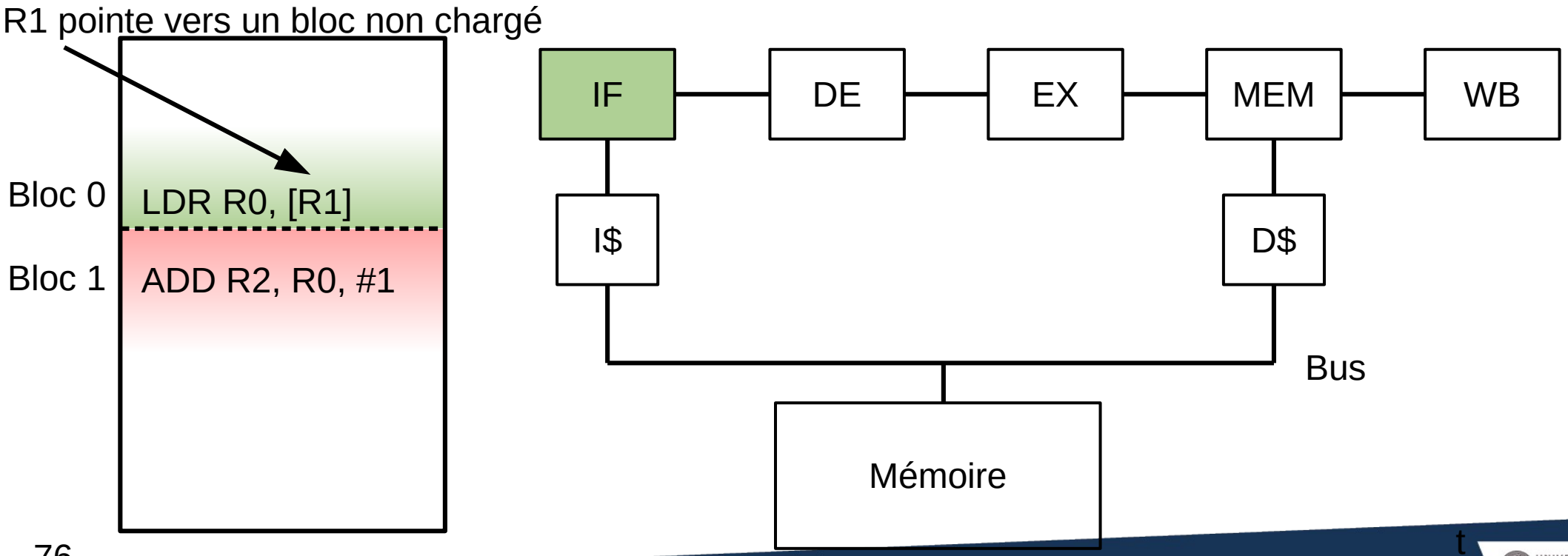
- ex : monoscalaire in-order



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

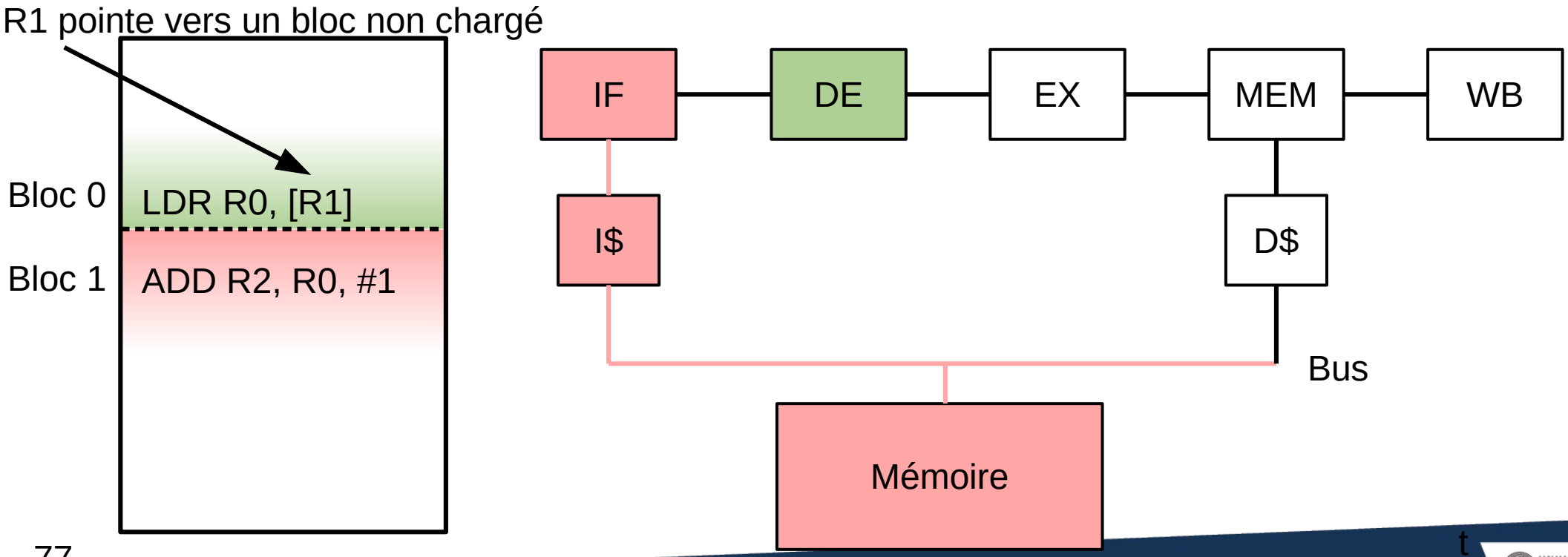
- ex : monoscalaire in-order



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

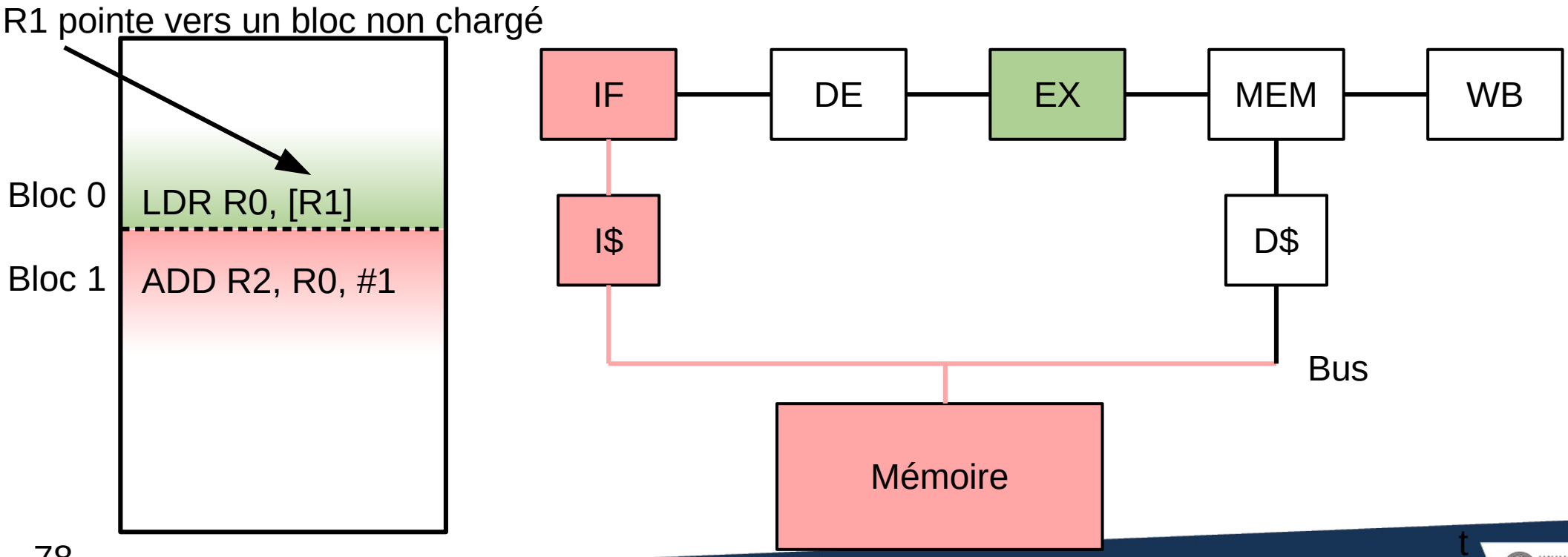
- ex : monoscalaire in-order



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

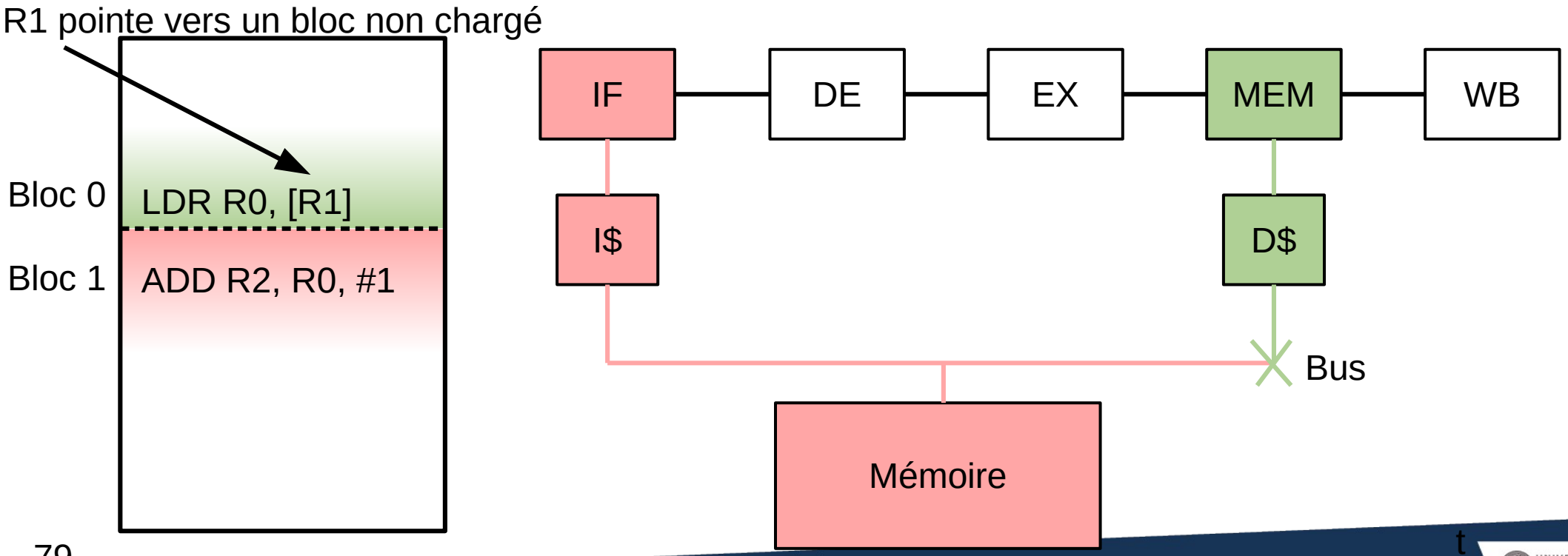
- ex : monoscalaire in-order



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- ex : monoscalaire in-order



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- ex : monoscalaire in-order

→ incertitude si le fetch du bloc rouge est NC

→ Un miss du I$ dans le passé peut avoir chargé le bloc rouge et mener à une meilleure situation ici

→ incertitude si des évènements précédents ne permettent pas de savoir si le fetch du bloc rouge commence avant l'opération mémoire du bloc vert

t

Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- 2 solutions :
  - calcul exhaustif des possibilités, malgré l'explosion combinatoire des cas
  - conception de processeurs garantis sans anomalie temporelle

Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- Accélérer les analyses pour permettre le calcul exhaustif des combinaisons

→ XDD : compactage des états possibles du pipeline

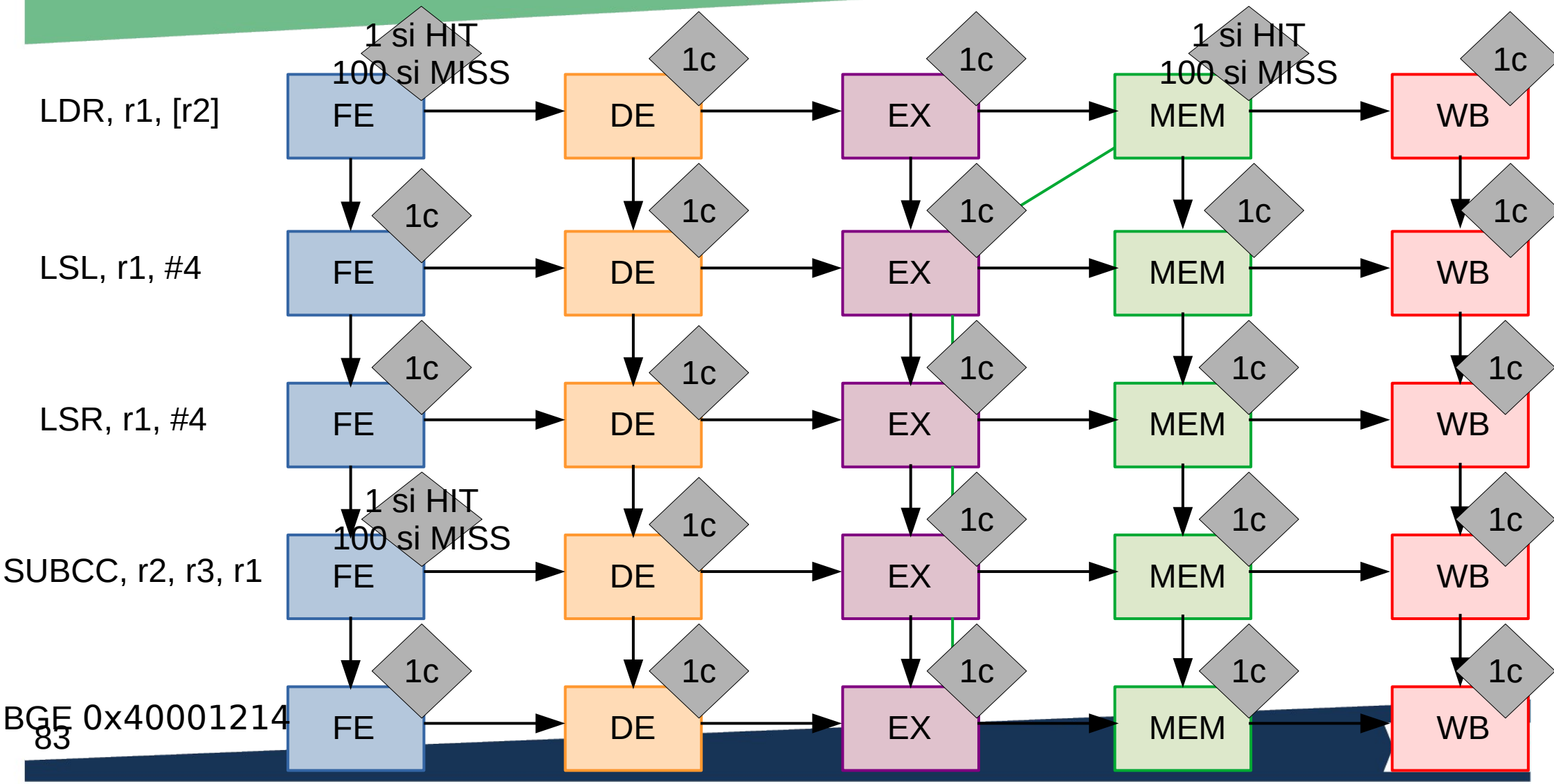
- Croiser les combinaisons d'évènements et les temps d'exécution

→ Arbre binaire

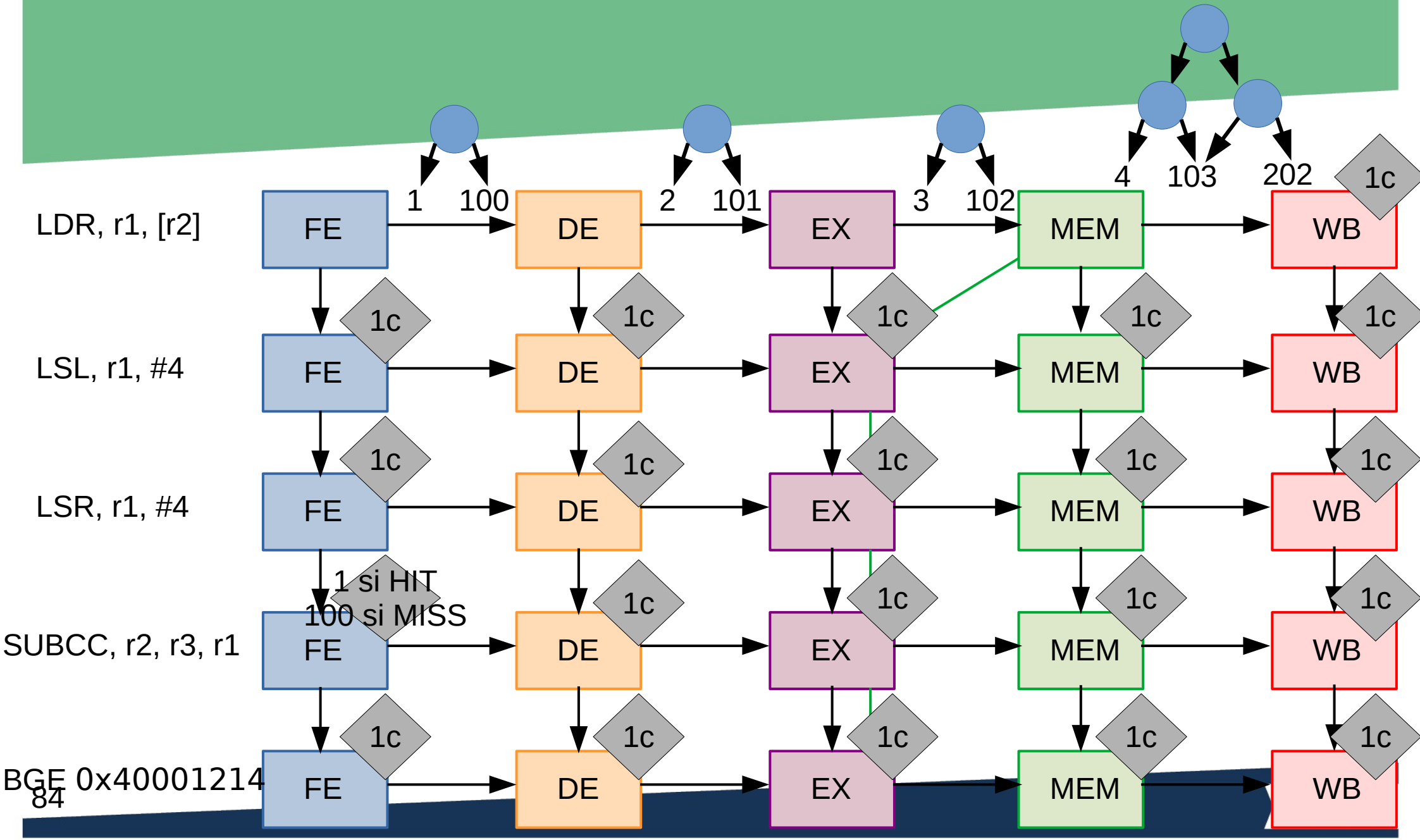
- Exploiter :
  - Propriétés d'amortissement des pipelines
  - Évènements qui s'annulent 2 à 2

→ Plusieurs combinaisons d'évènements mènent en fait au même temps

Timing anomalies



Timing anomalies



Timing anomalies

→ **Combinaisons d'évènements dans l'analyse microarchitecturale**

- Accélérer les calculs pour permettre le calcul exhaustif des combinaisons

→ XDD : compactage des états possibles du pipeline

- Concevoir des processeurs sans anomalie temporelle

→ VLIW : e.g. PATMOS (+ interconnect TDM), Kalray K1-K3 (+ scratchpads)

→ Restrictions sur spéculation et mémoire : e.g. SIC, Vicuna

→ Exotique : e.g. PRET (multithread matériel)

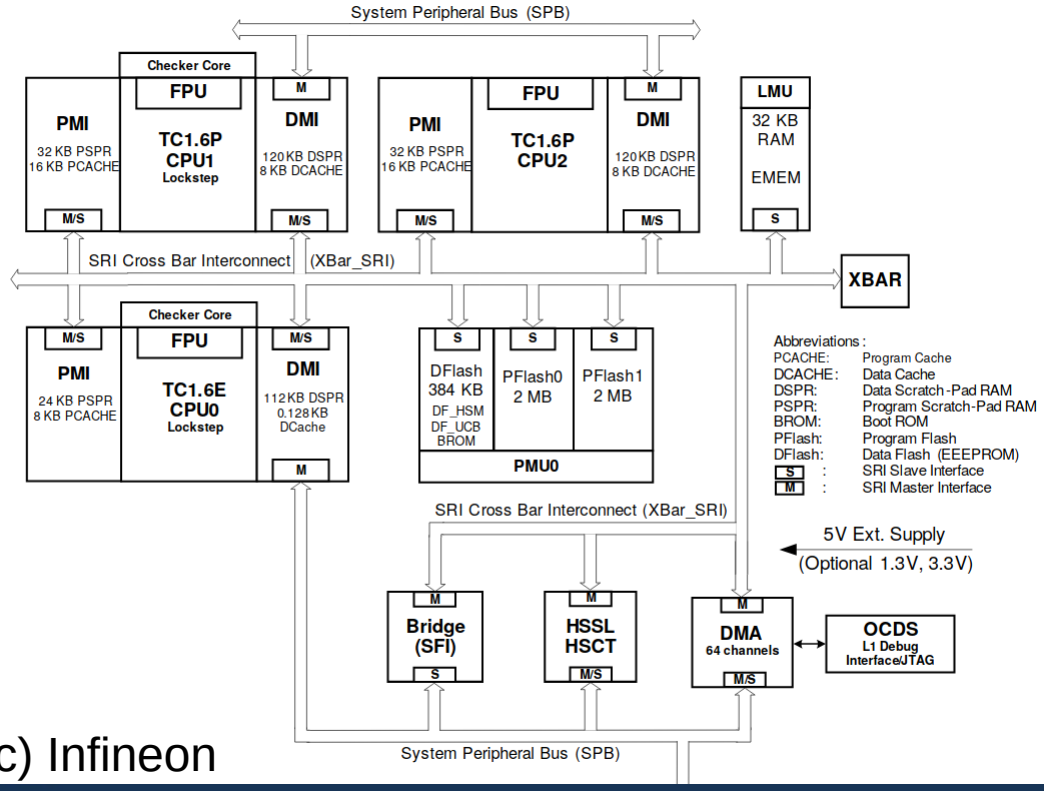
Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

- indépendance des coeurs → exécution en isolation
- MAIS partage des composants mémoire, en particulier bus

Multicoeur et interférences

→ Exécution sur multicoeurs à mémoire partagée

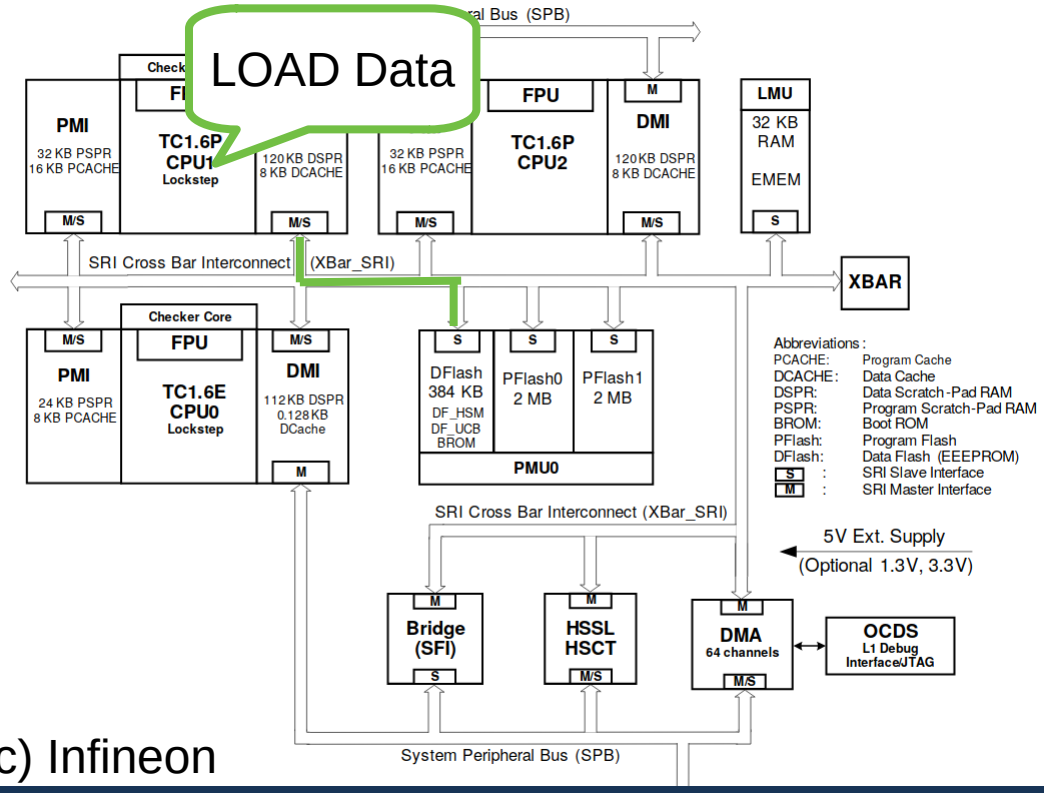


Infineon TriCore TC275

Infineon website (c) Infineon

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

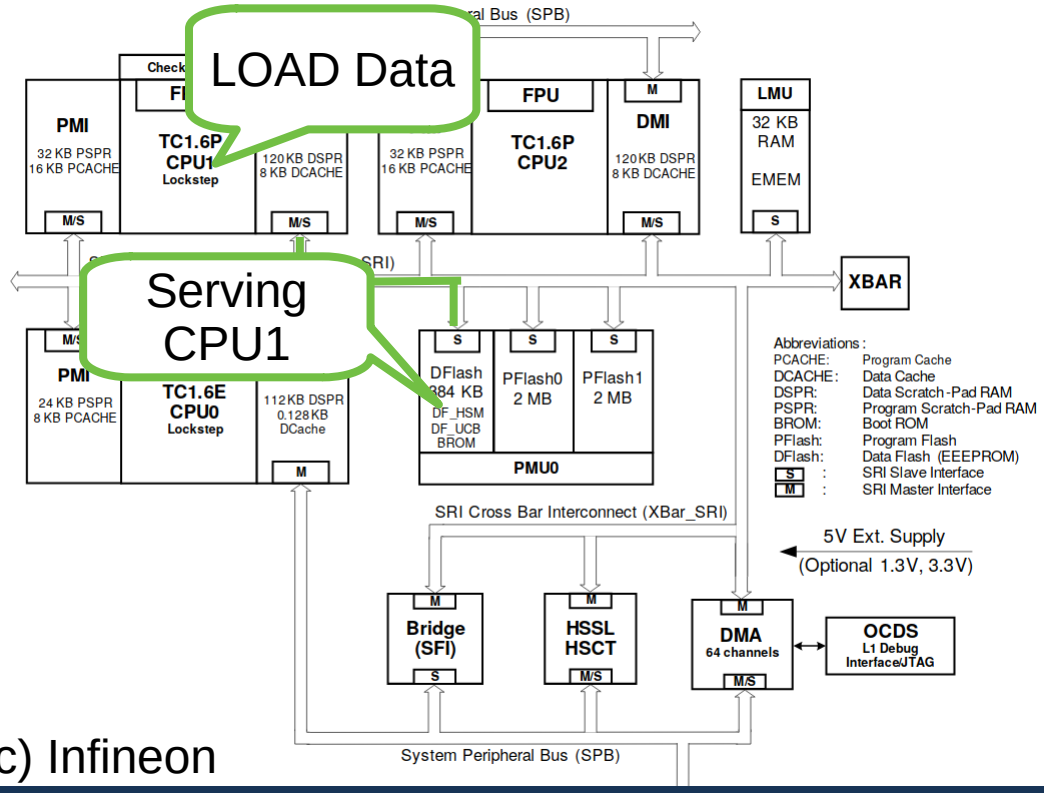


Infineon TriCore TC275

Infineon website (c) Infineon

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

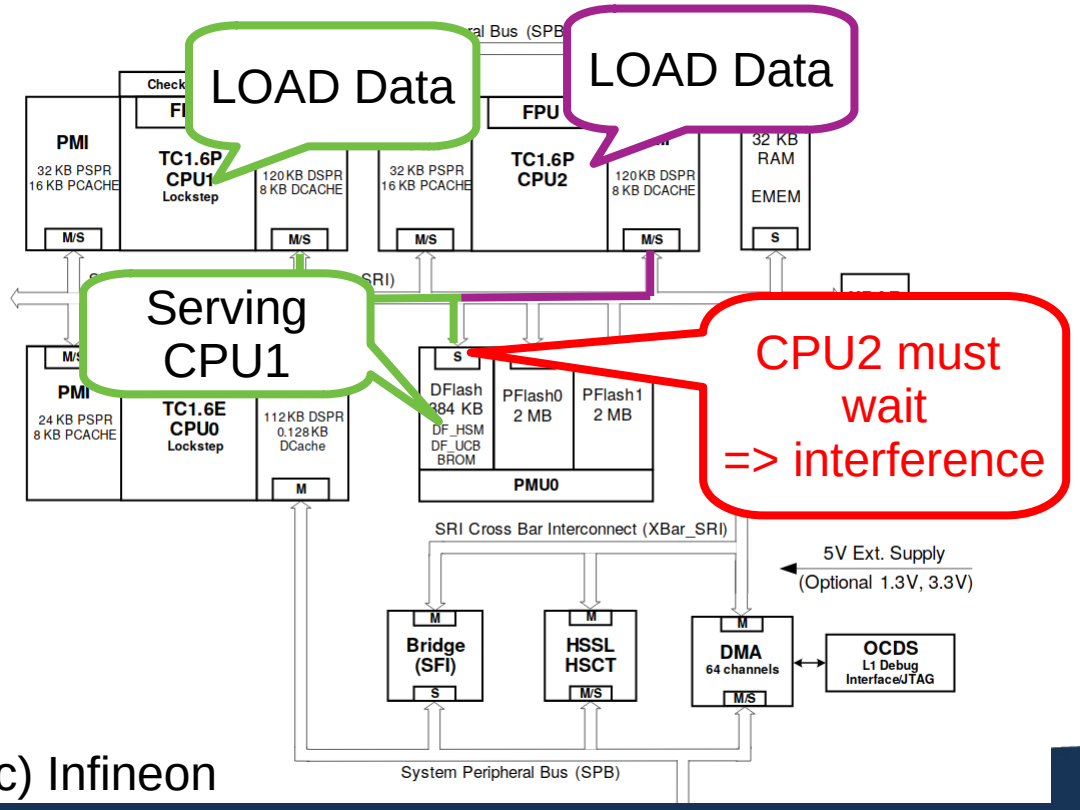


Infineon TriCore TC275

Infineon website (c) Infineon

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**



Infineon TriCore TC275

Infineon website (c) Infineon

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

- indépendance des coeurs → exécution en isolation
- MAIS partage des composants mémoire, en particulier bus

→ Analyse WCET en isolation insuffisante à cause du phénomène d'interférence (~ généralisation du problème préemptif)

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

- solutions analytiques :
  - adaptation de WCRT sur traces d'exécution

    → Explosion combinatoire
  - RT-calculus (adaptation de network calculus)

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

- solutions compilation : AER/REW

→ tâches écrites en trois morceaux :

- Acquisition : copie du code et des données depuis la mémoire vers le coeur
- Execution : exécution depuis la mémoire locale du coeur
- Restitution : copie des résultats vers la mémoire partagée

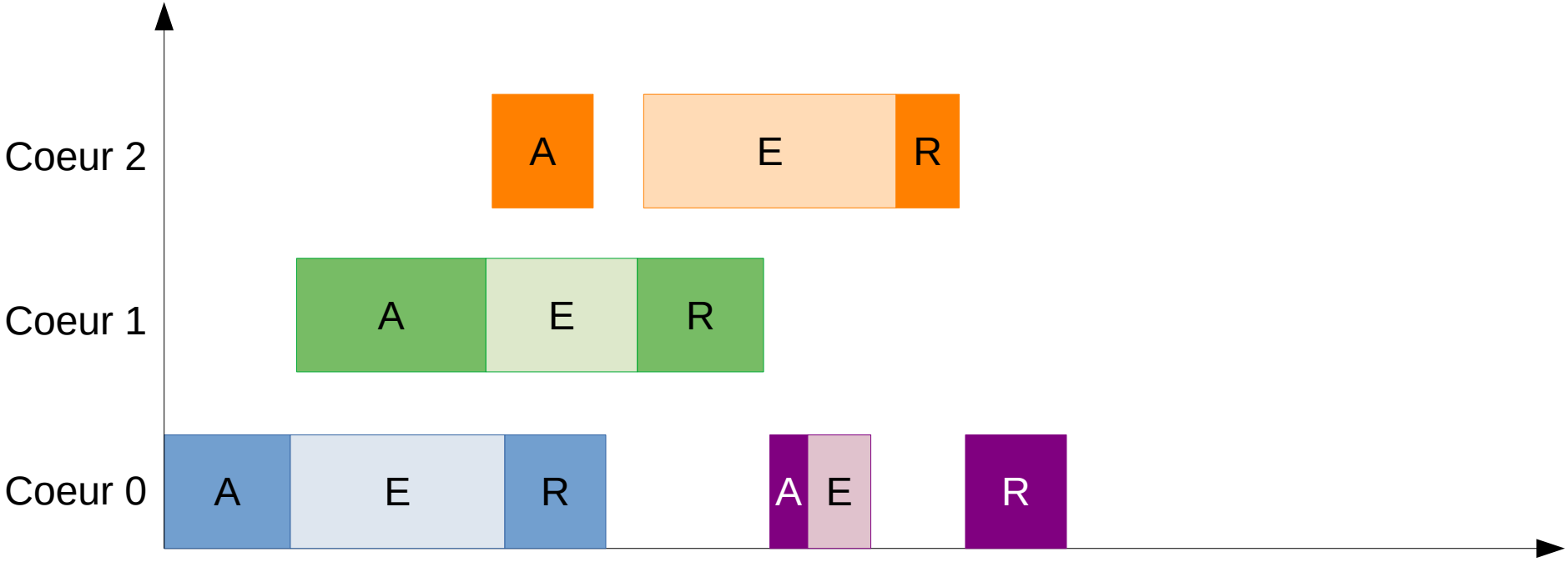
→ Seules les phases A et R sollicitent l'interconnect

→ Construction d'un ordonnancement statique dans lequel les phases A et R n'arrivent jamais en même temps

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

- solutions compilation : AER/REW



Architectures hétérogènes

→ **System-on-Chip embarquant des calculateurs de différentes natures**
- e.g. CPU+GPU, CPU+FPGA

→ **Prise en compte de nouveaux modèles d'exécution matérielle**
- GPU : modèle SIMT
- Systolic arrays