Analyse WCET et OTAWA

T. Carle
Université de Toulouse, IRIT

ETR, 10 Septembre 2026





Analyse WCET et OTAWA
Contexte

Systèmes embarqués temps-réel

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

Notion de pire-temps d'exécution

- analyse de pire temps de réponse

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

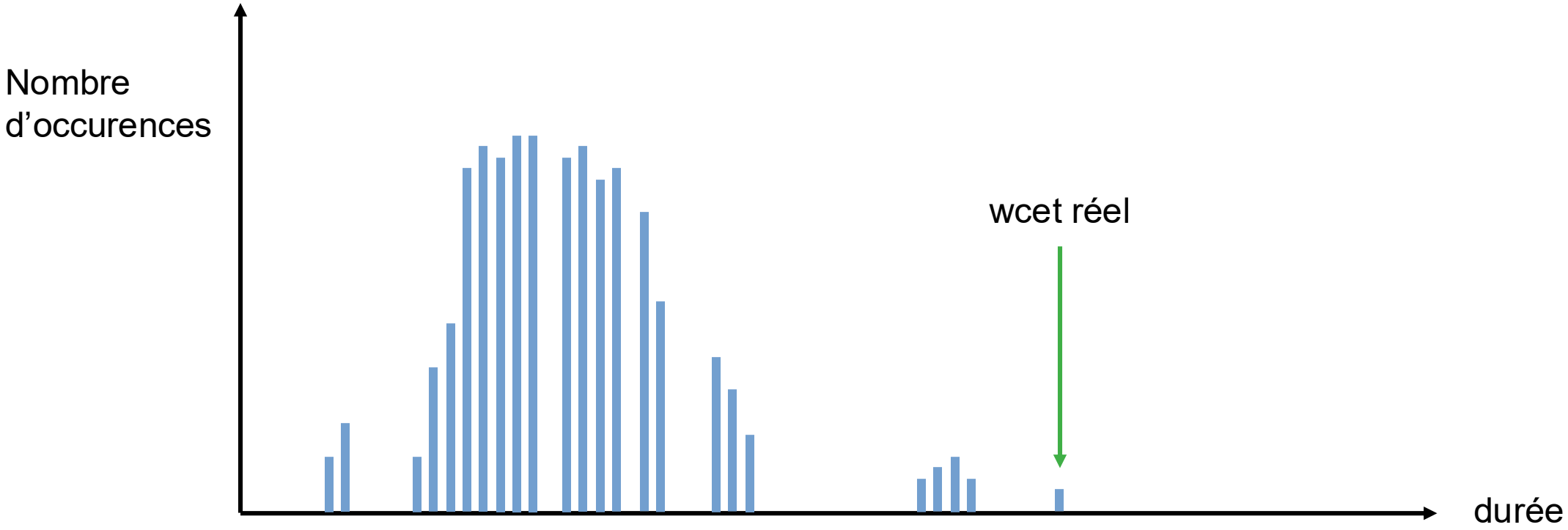
Pire temps d'exécution ?

→ **variabilité du temps d'exécution d'un programme pour une 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, caches, prédicteurs de branchement)

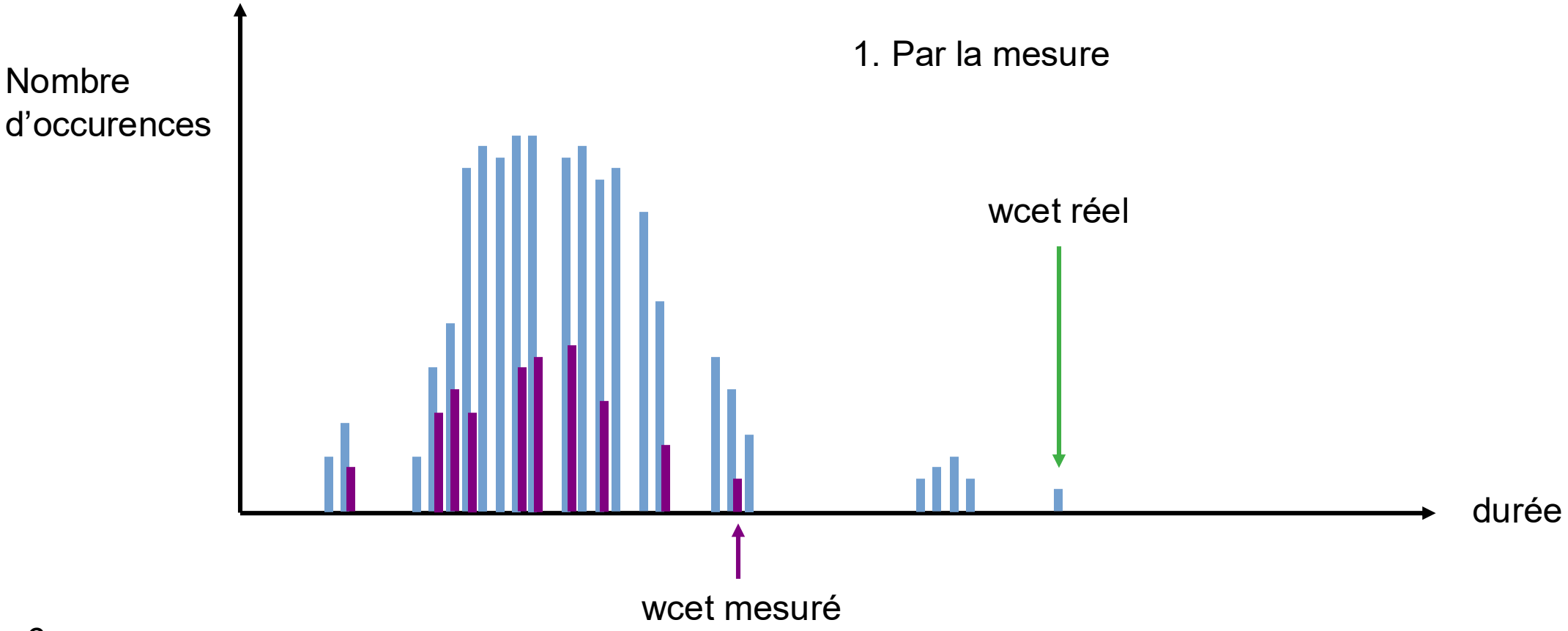
Analyse WCET et OTAWA
Introduction

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 :



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
- Non conservatif

Analyse WCET et OTAWA
Introduction

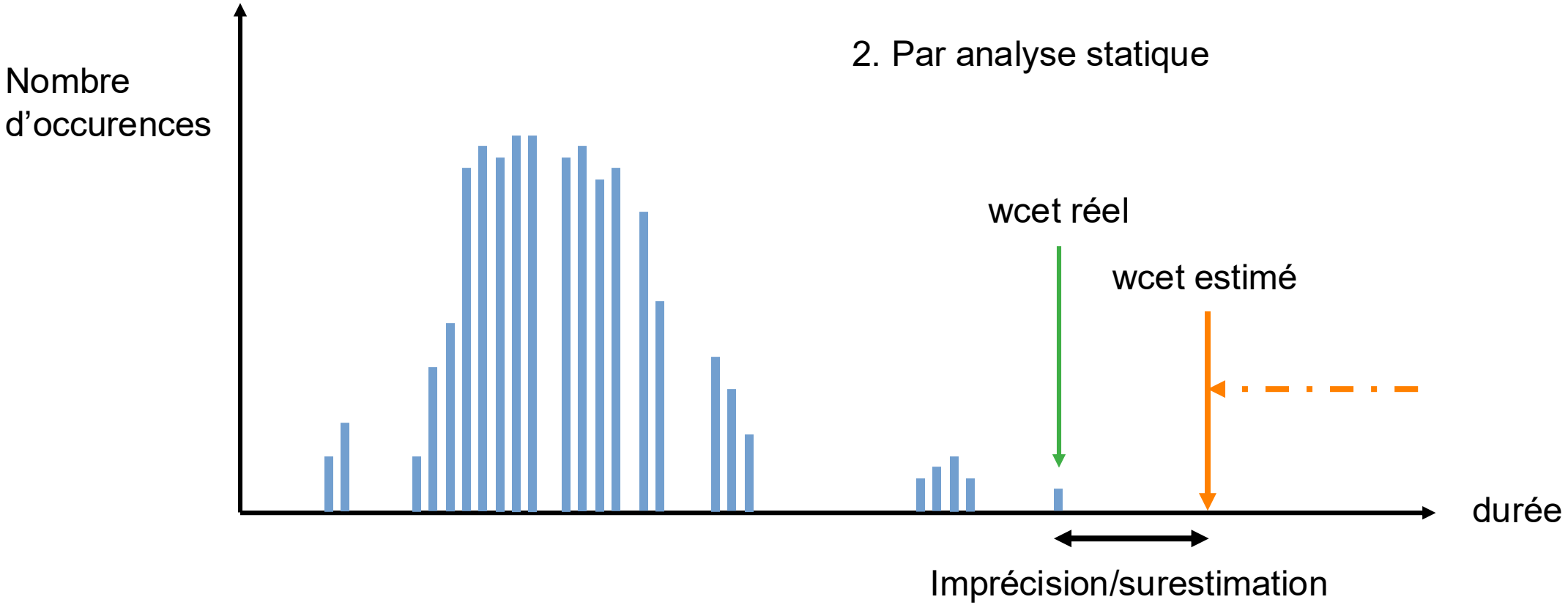
Objectif : déterminer le pire temps d'exécution d'un programme :

→ **Par la mesure**
- Relativement simple et rapide à mettre en œuvre
- Non conservatif

→ **Méthodes probabilistes**
- Campagne de mesures, puis reconstruction d'une distribution
- Garanties probabilistes

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 analyse statique**

- Modèle du programme (CFG)
- Analyses statiques conservatives (état du matériel)
- Modèle de la microarchitecture => temps des blocs de base
- Enumération implicite des chemins du CFG

- Plus compliqué à mettre en œuvre (modèle + surestimation)
- 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

- Le meilleur (ou le pire) des deux mondes

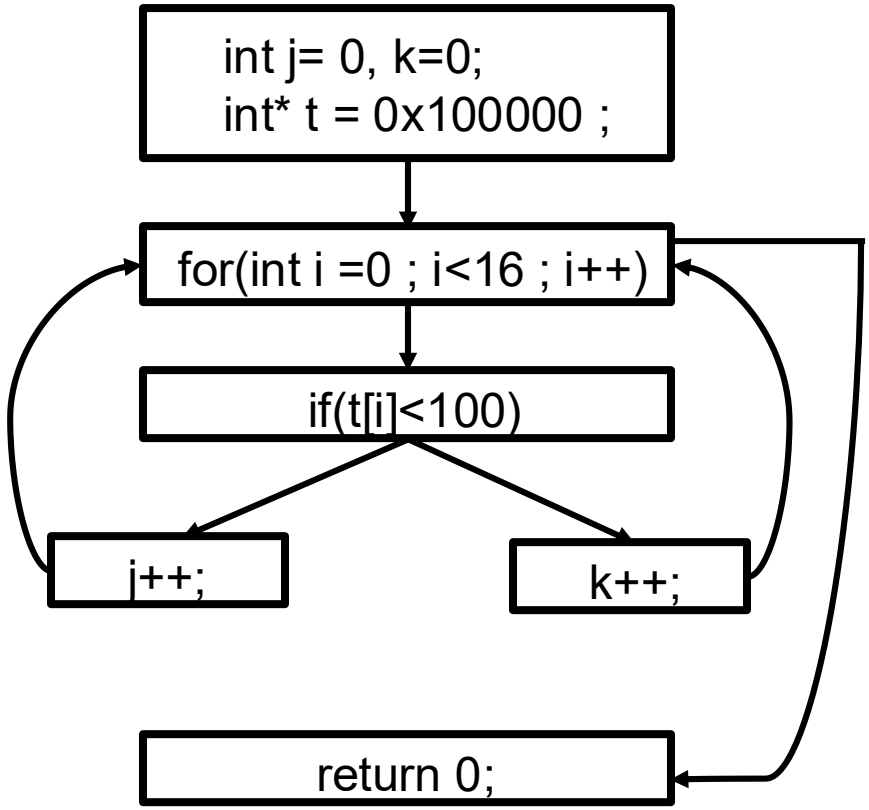
Analyse WCET et OTAWA

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;
}
```

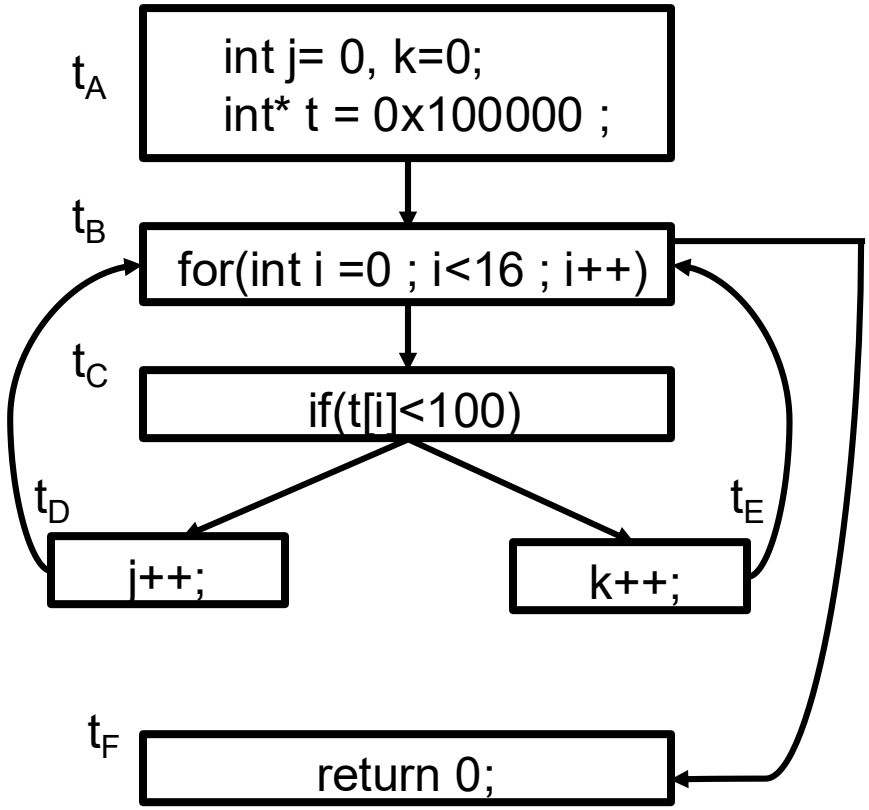


Analyse WCET et OTAWA

Méthode IPET : les grandes lignes

→ **Associer une durée t à chaque bloc**

- Déterminée par analyse



Analyse WCET et OTAWA

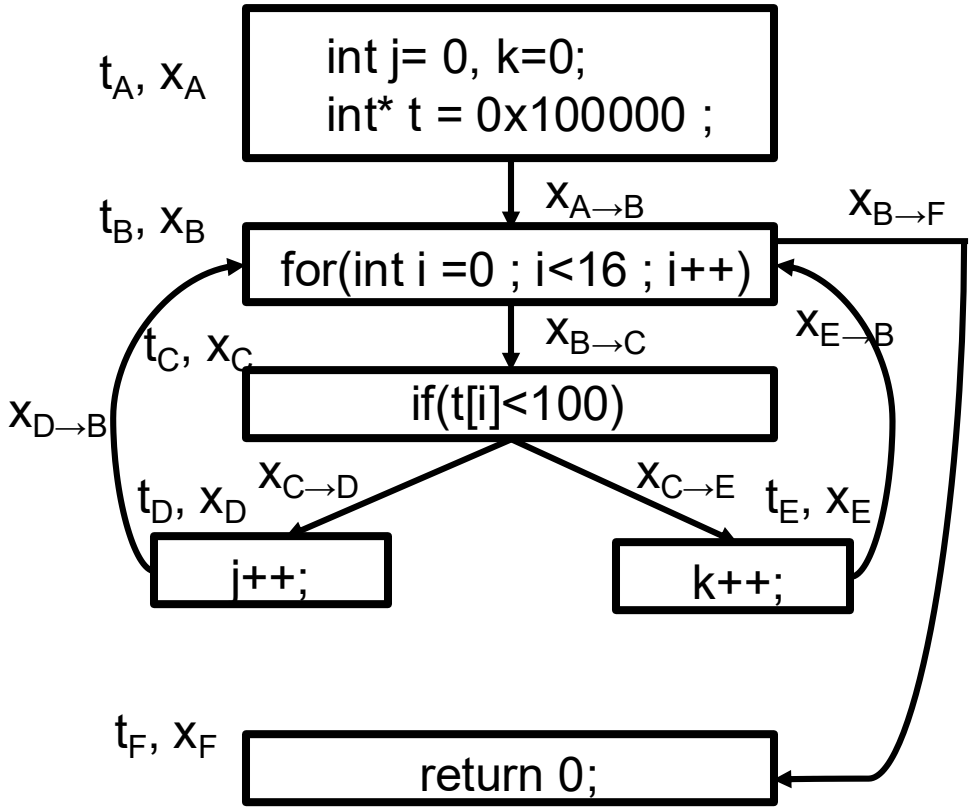
Méthode IPET : les grandes lignes

→ **Associer une durée t à chaque bloc**

- Déterminée par analyse

→ **Associer des variables d'occurence x**

- A chaque bloc
- A chaque arc



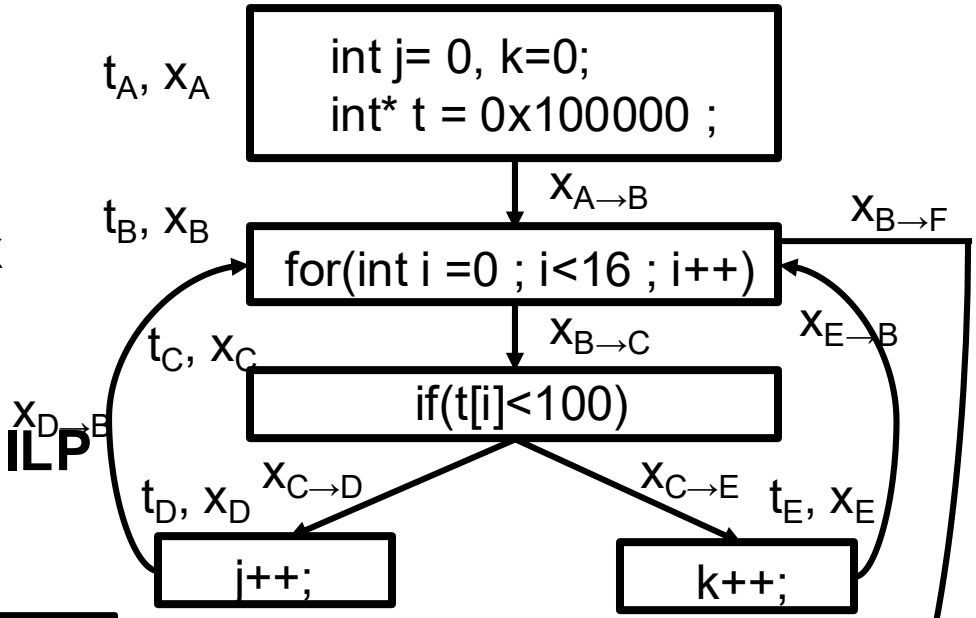
Analyse WCET et OTAWA

Méthode IPET : les grandes lignes

→ **Associer une durée t à chaque bloc**
- Déterminée par analyse

→ **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



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

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

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

$x_{D\to B} + x_{E\to B} \leq 16$

Analyse WCET et OTAWA

Méthode IPET : les grandes lignes

→ **Associer une durée t à chaque bloc**
- Déterminée par analyse

→ **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 :**

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

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

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

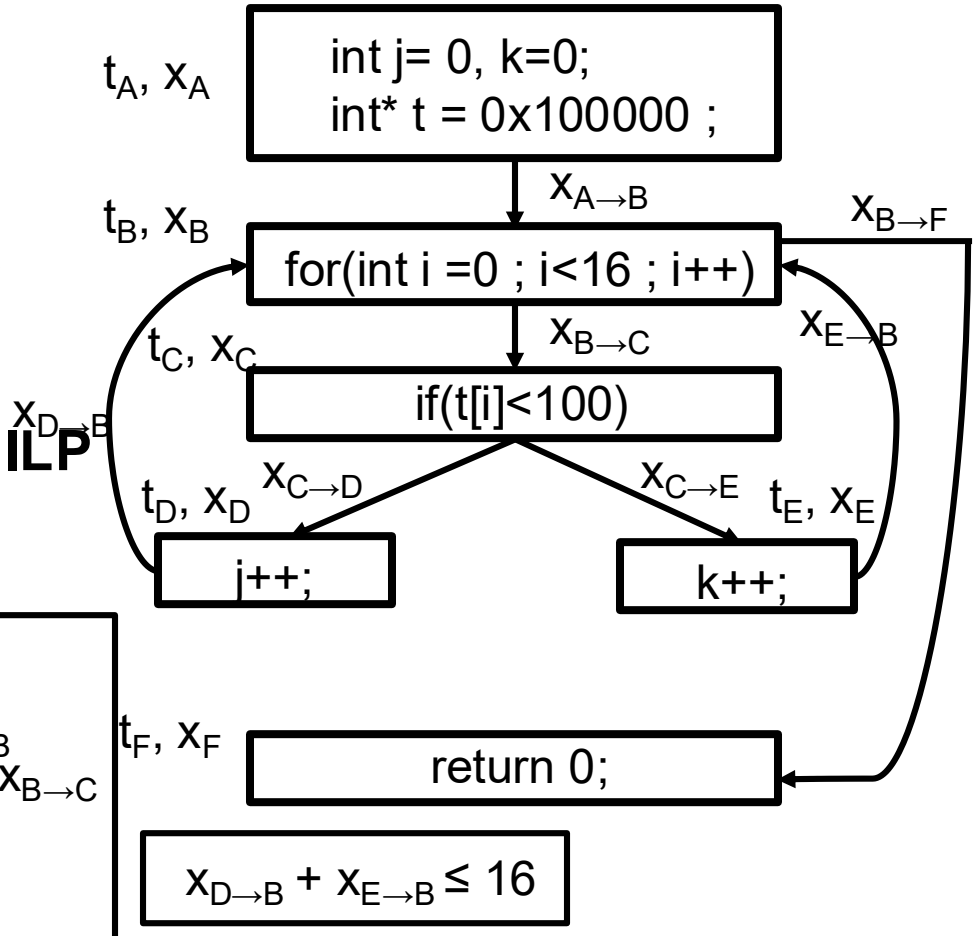
$x_{B\to C} + x_{B\to F} = x_B ; x_C = x_{B\to C}$

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

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

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

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



Analyse WCET et OTAWA

Et les t_i ?

→ **Analyse microarchitecturale**

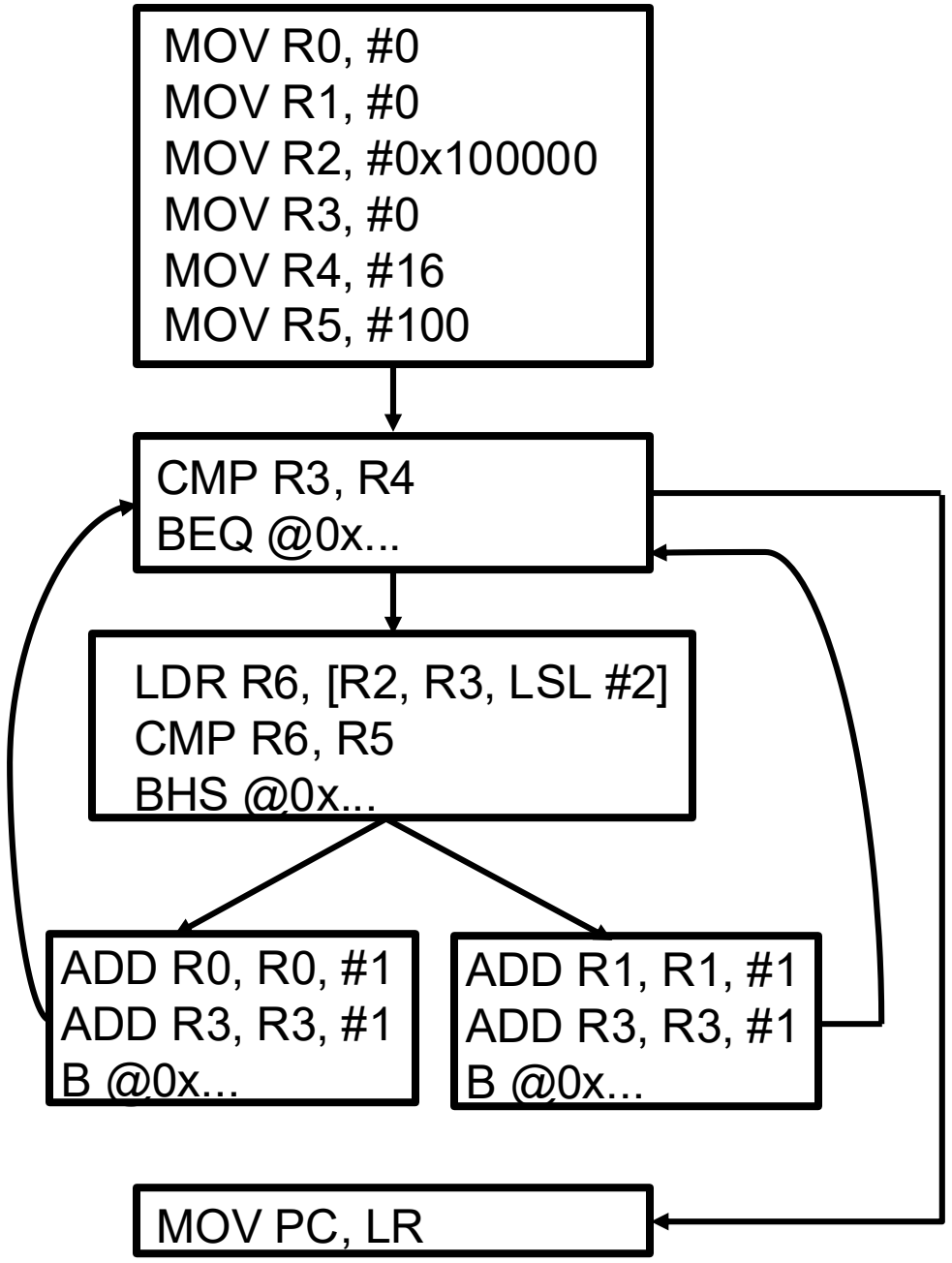
- Modélisation des éléments micro-architecturaux
- Travail sur le fichier binaire correspondant à l'exécutable

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

Analyse WCET et OTAWA

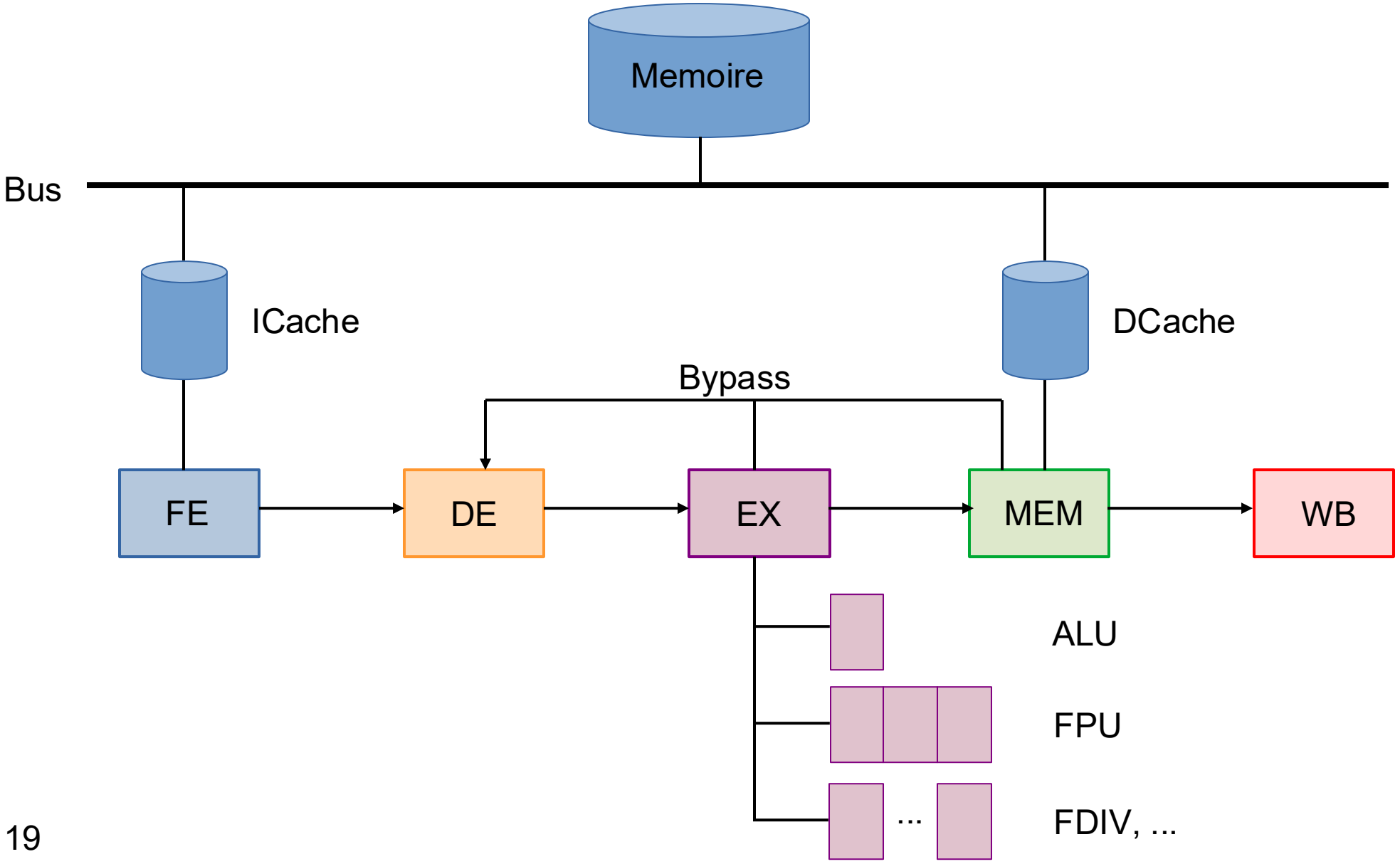
Et les t_i ?

→ **Analyse microarchitecturale**



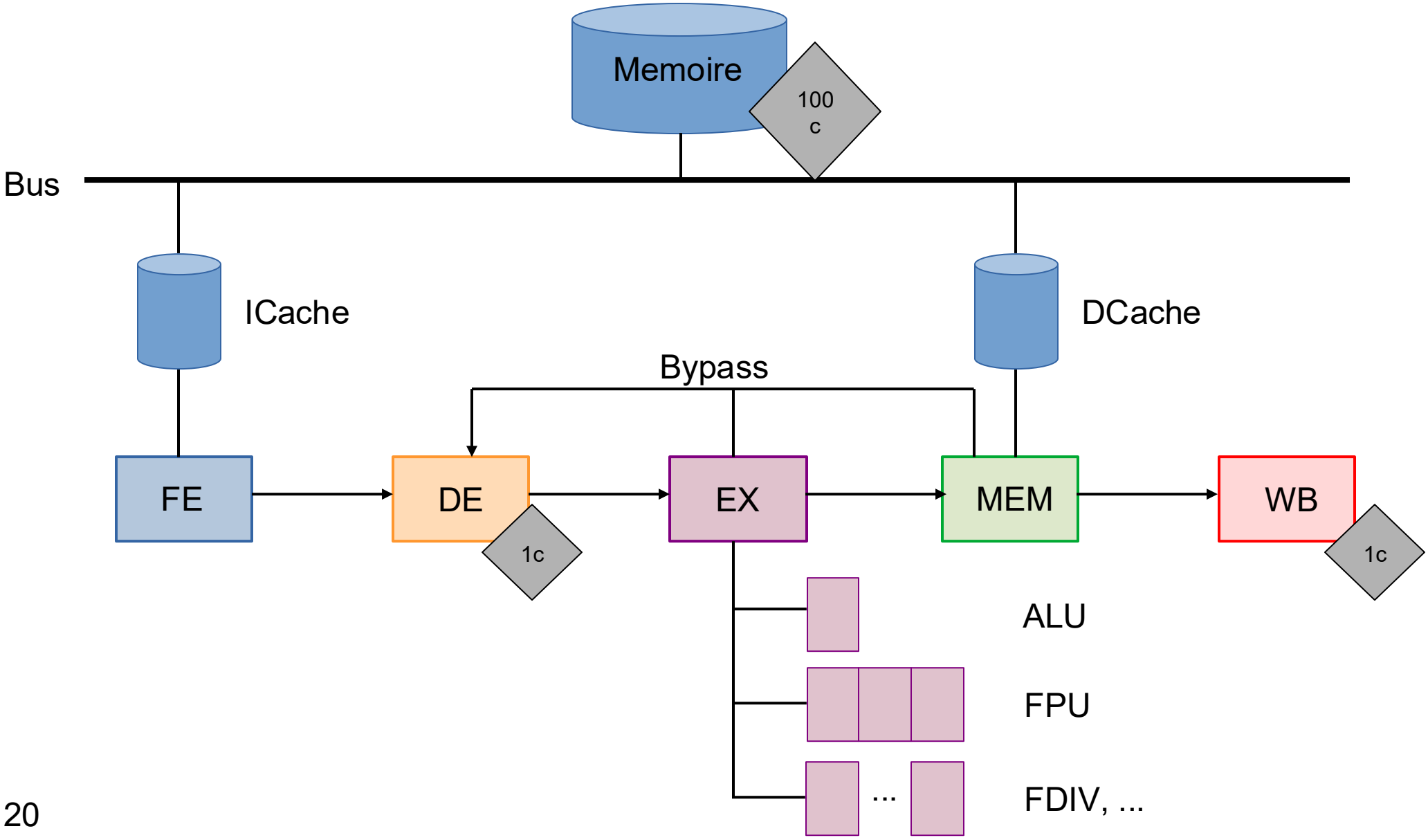
Analyse WCET et OTAWA

Modéliser l'architecture



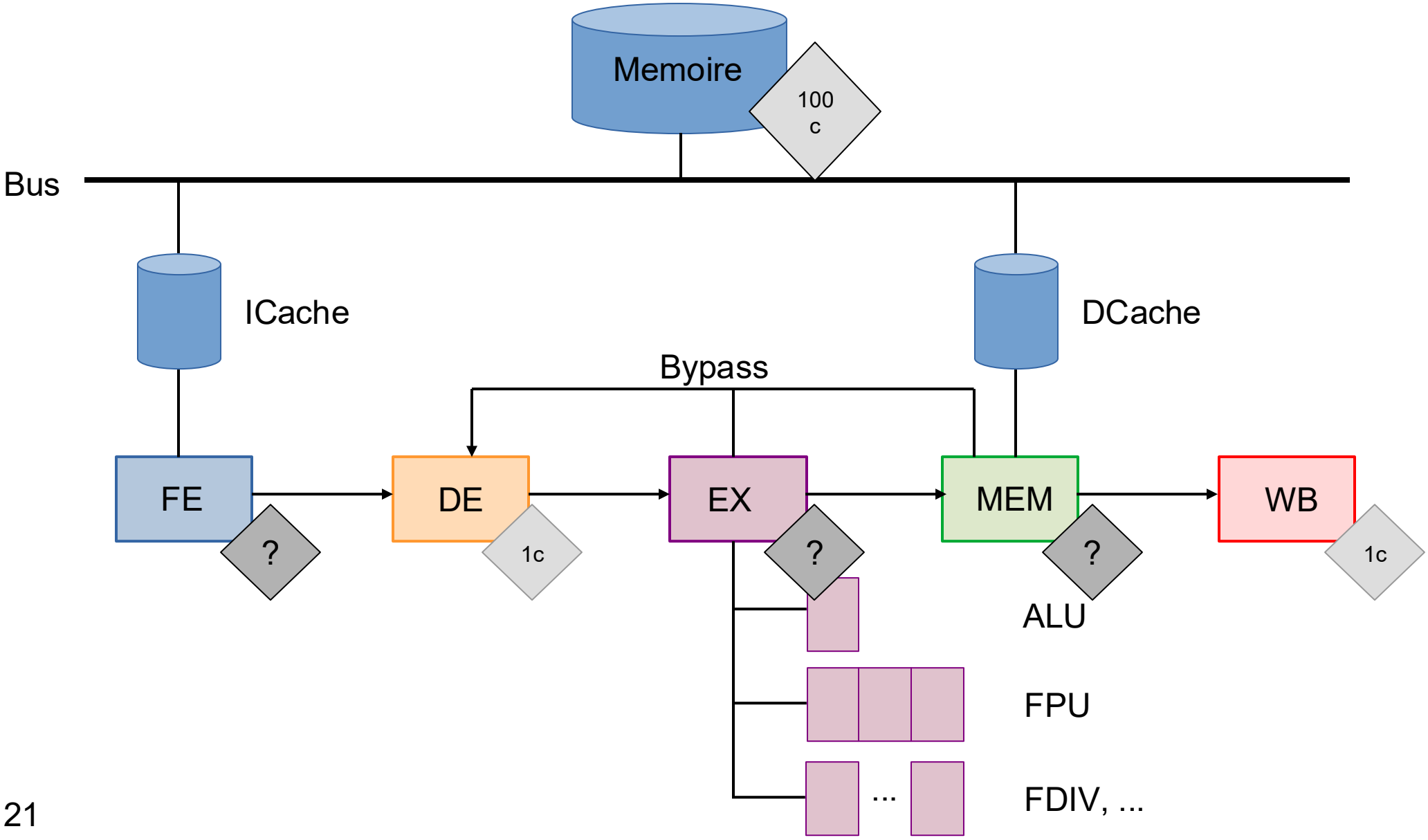
Analyse WCET et OTAWA

Modéliser l'architecture



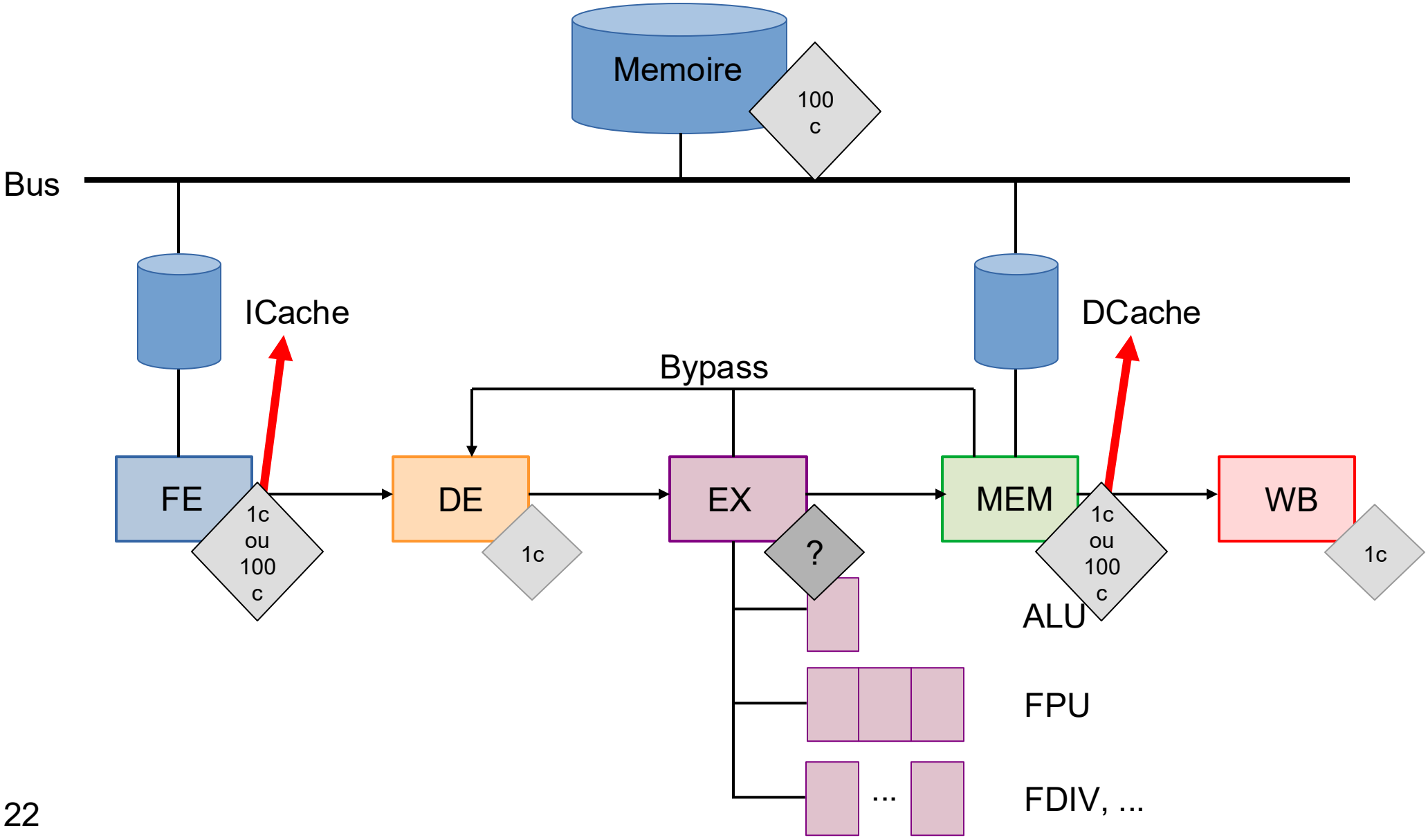
Analyse WCET et OTAWA

Modéliser l'architecture



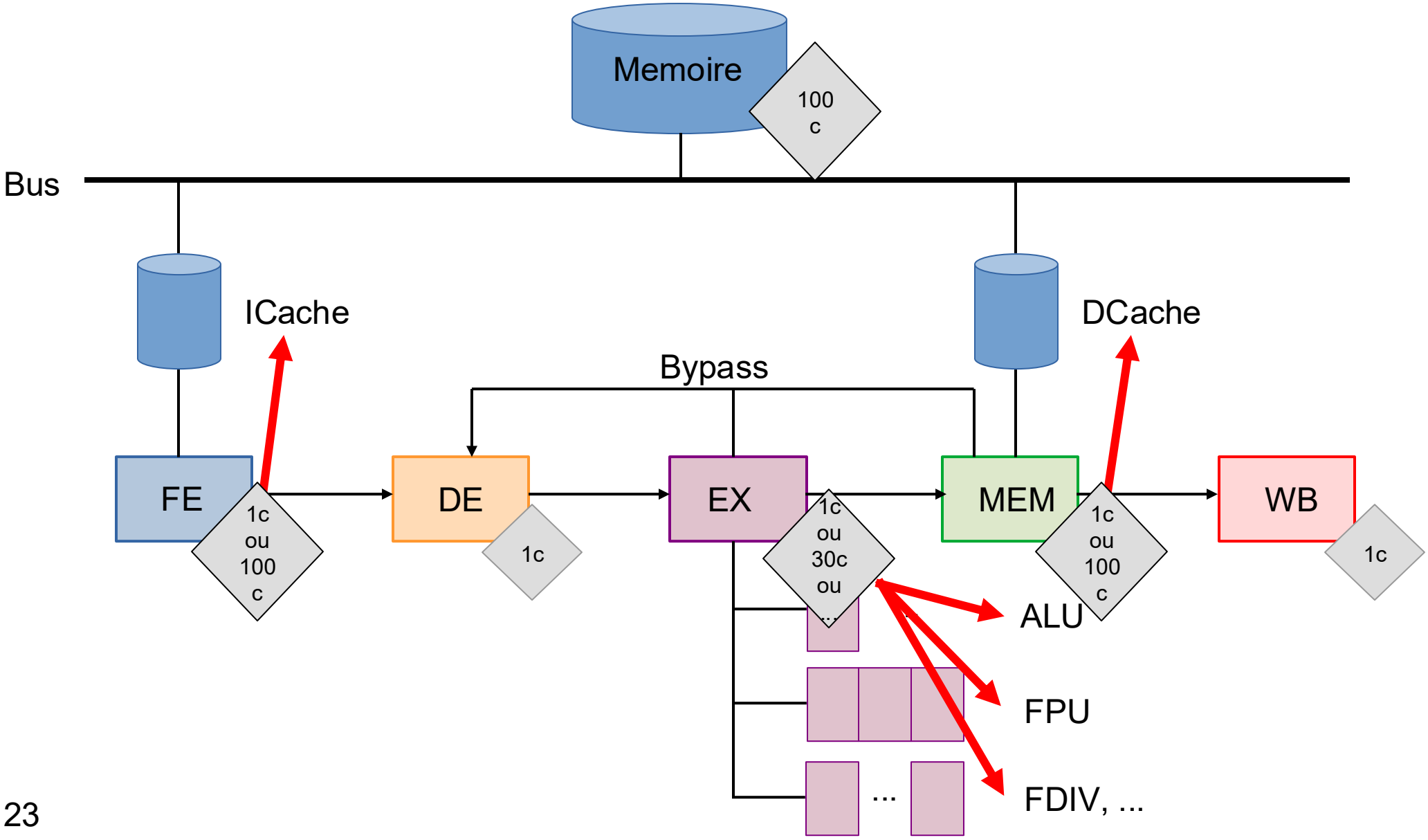
Analyse WCET et OTAWA

Modéliser l'architecture



Analyse WCET et OTAWA

Modéliser l'architecture



Analyse WCET et OTAWA

Et les t_i ?

→ **Analyse microarchitecturale**

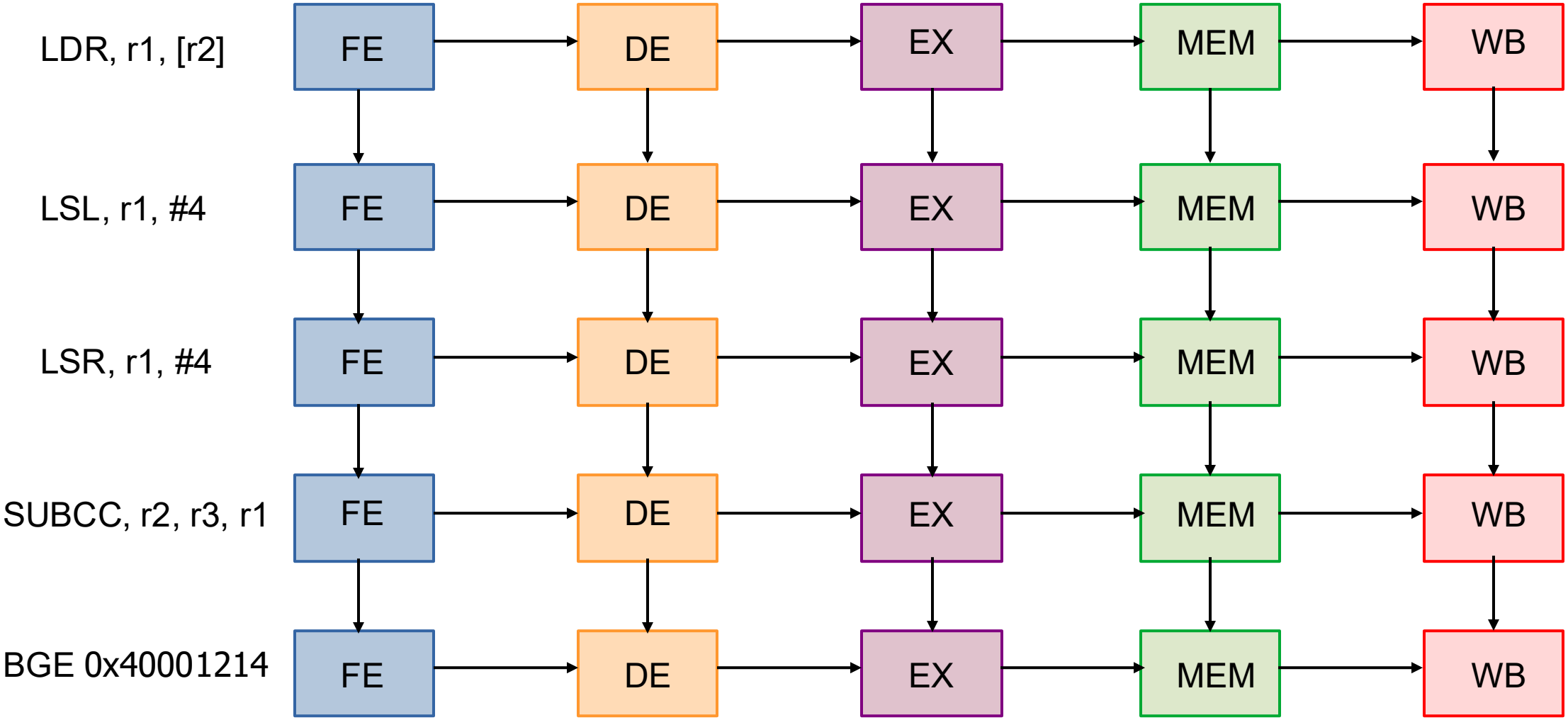
- Modélisation des éléments micro-architecturaux
- Travail sur le fichier binaire correspondant à l'exécutable
  → Plus précis, et certitude d'analyser ce qui va être exécuté

- ti => temps de traversée du pipeline par le bloc de base
  → Pipeline = élément central de l'analyse microarchitecturale

Analyse WCET et OTAWA

Modéliser le pipeline : EXEGraphs

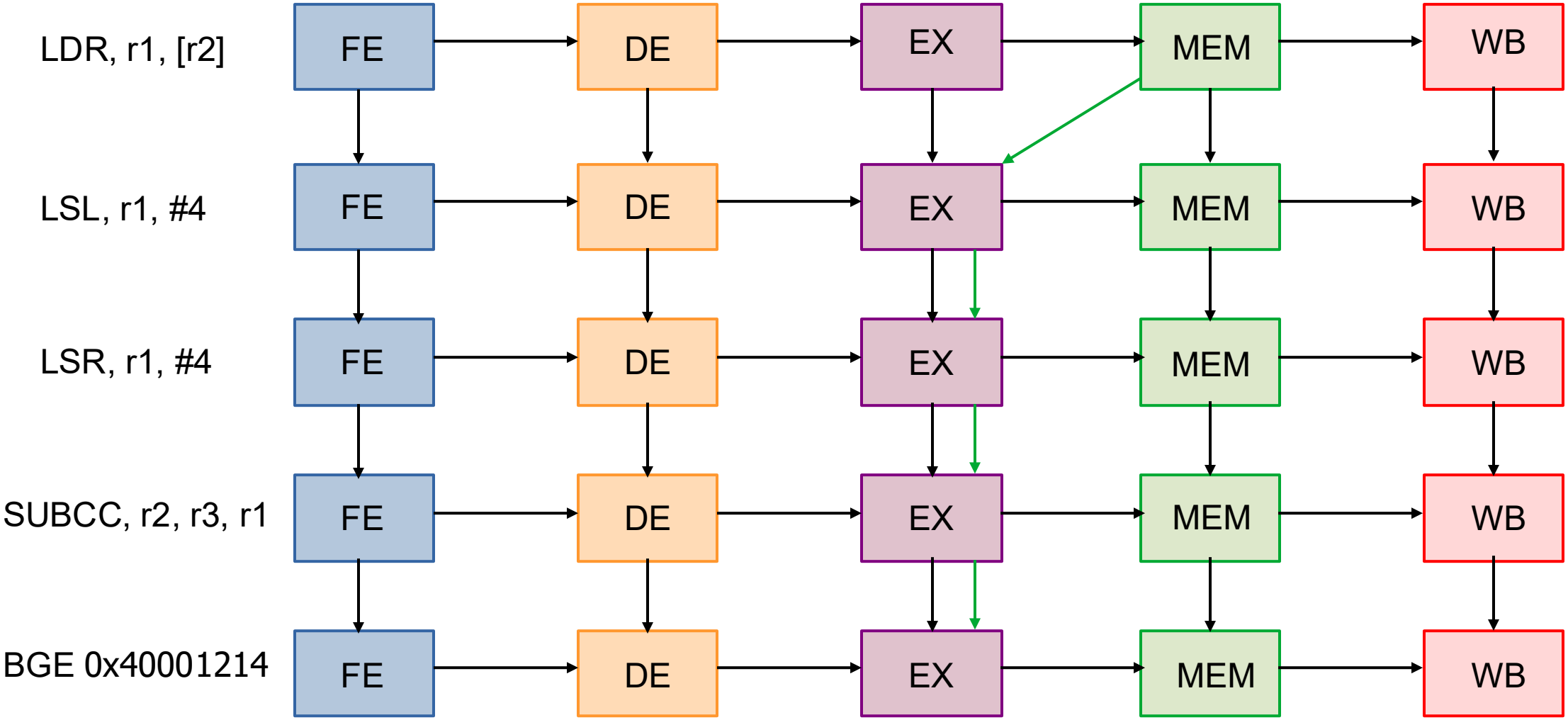
Contraintes structurelles du pipeline



Analyse WCET et OTAWA

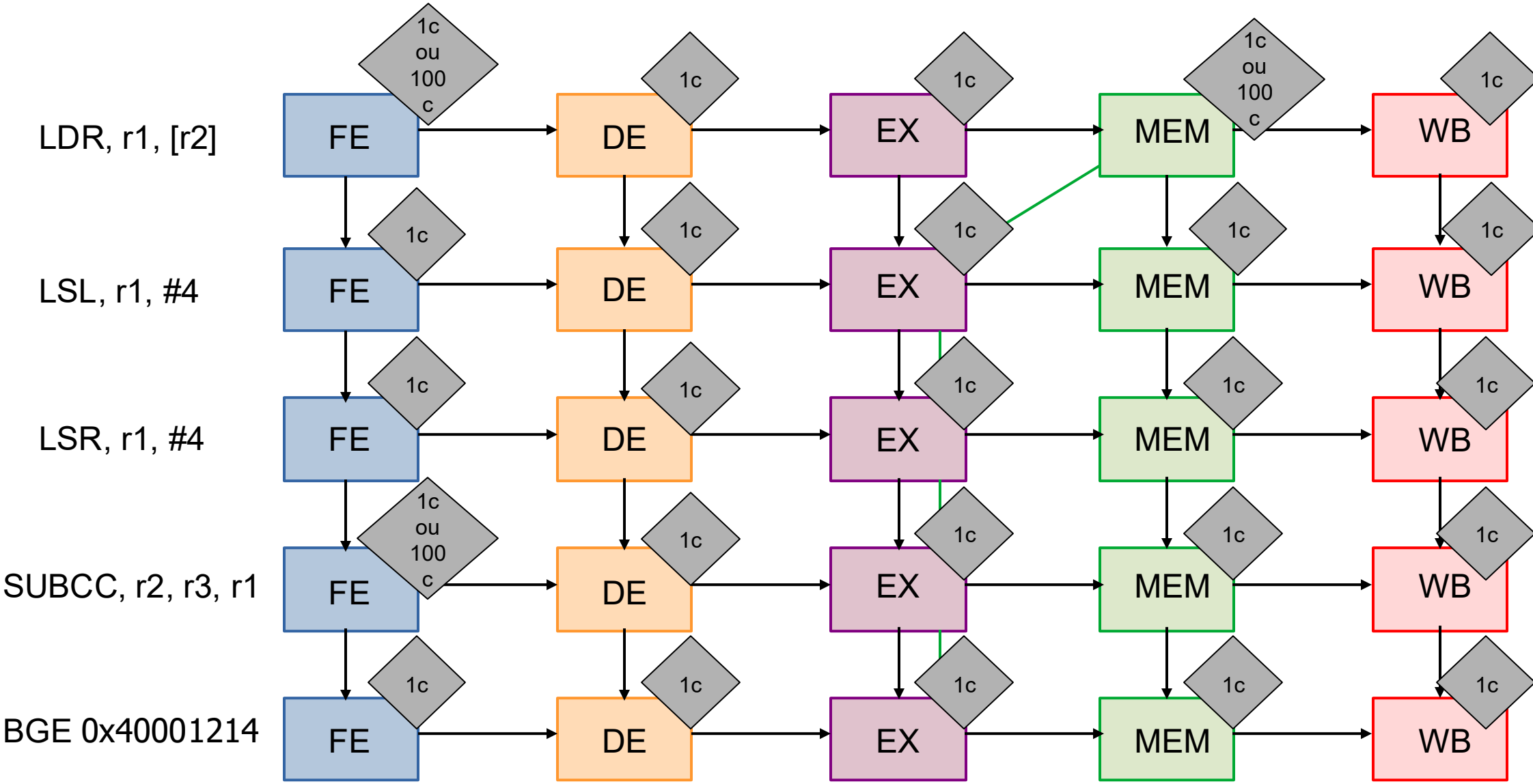
Modéliser le pipeline

Contraintes liées au code



Analyse WCET et OTAWA

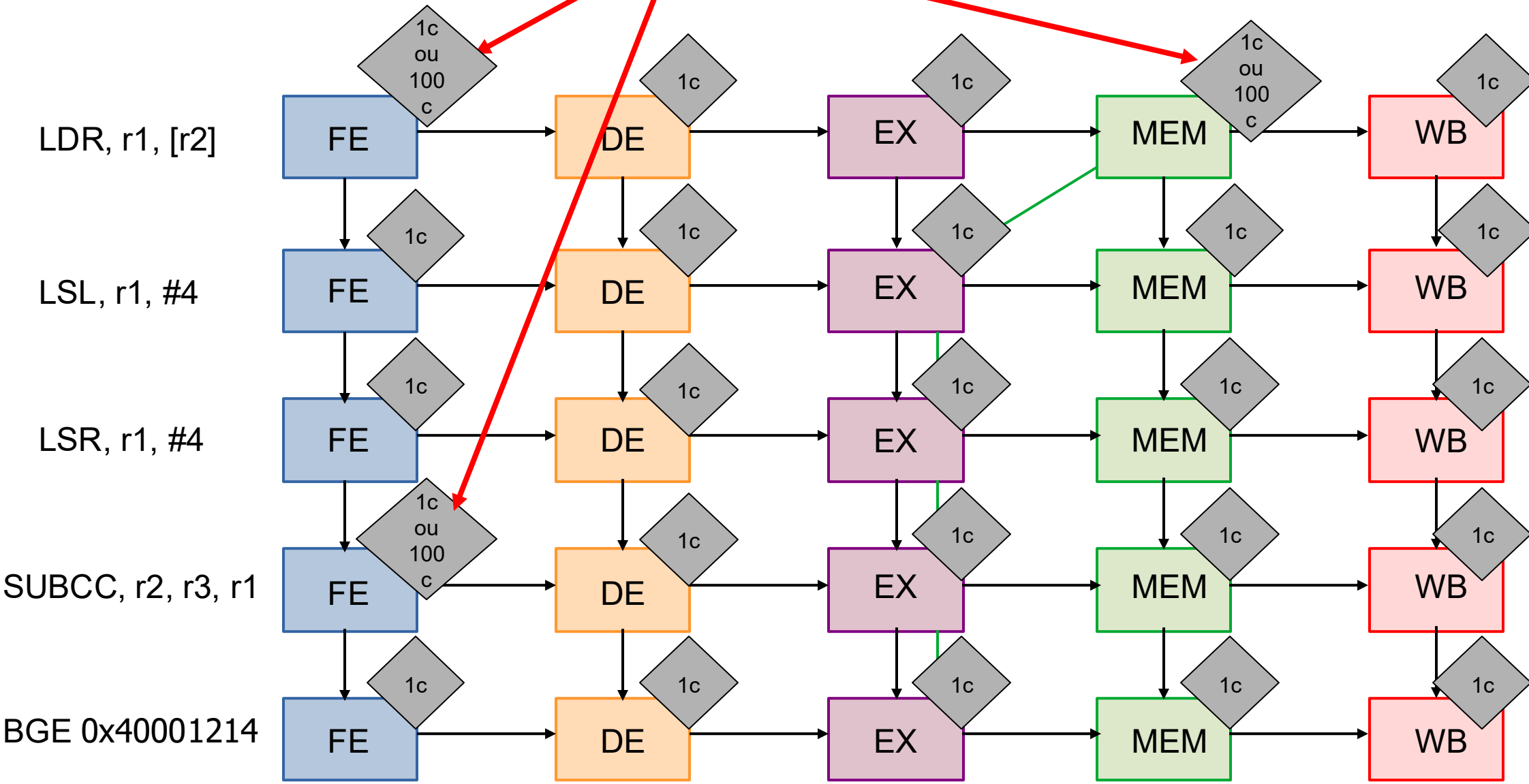
Modéliser le pipeline



Analyse WCET et OTAWA

Modéliser le pipeline

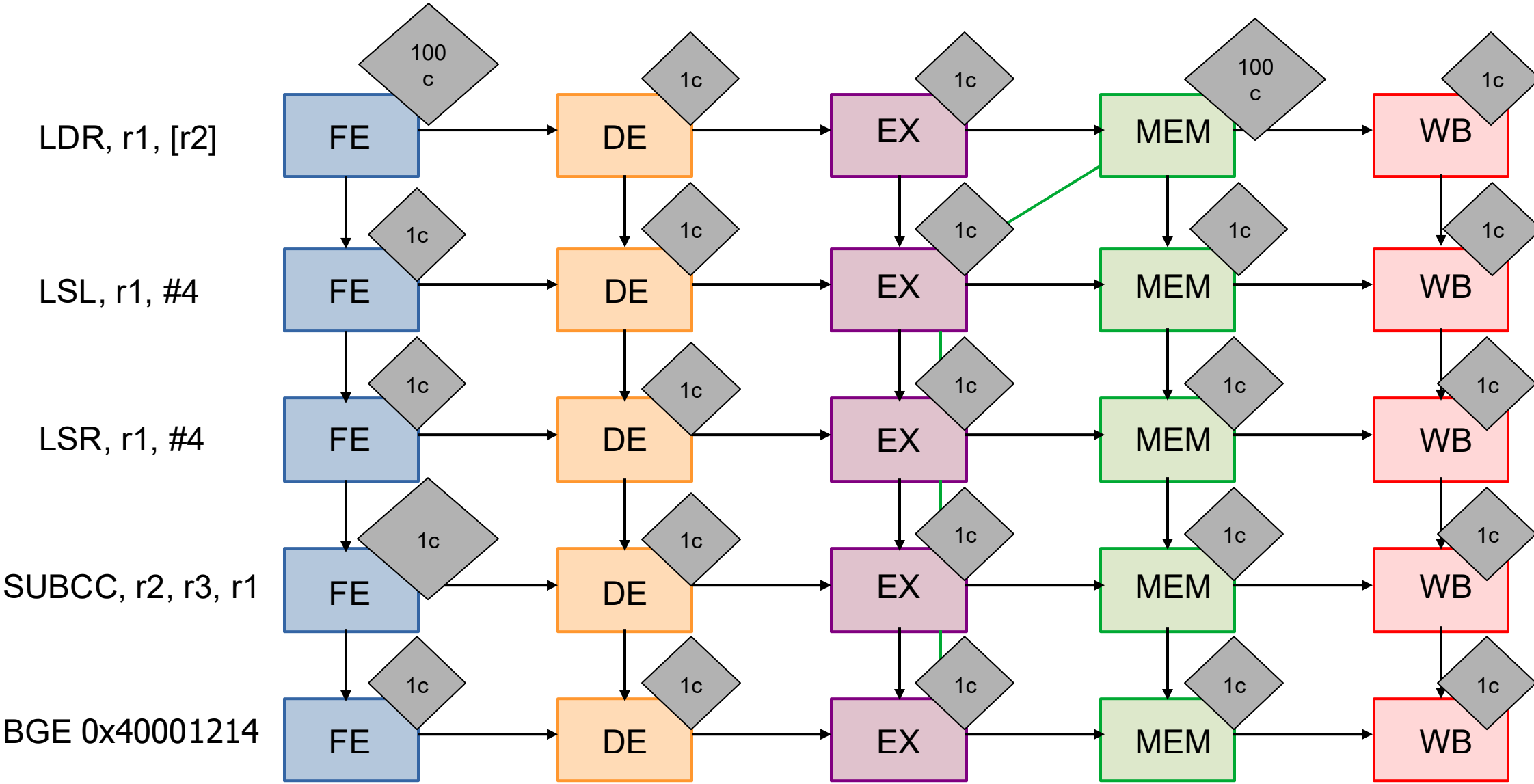
Evénements : 1 exegraph par combinaison
→ ici : 8 exegraphs pour représenter le BB



Analyse WCET et OTAWA

Modéliser le pipeline

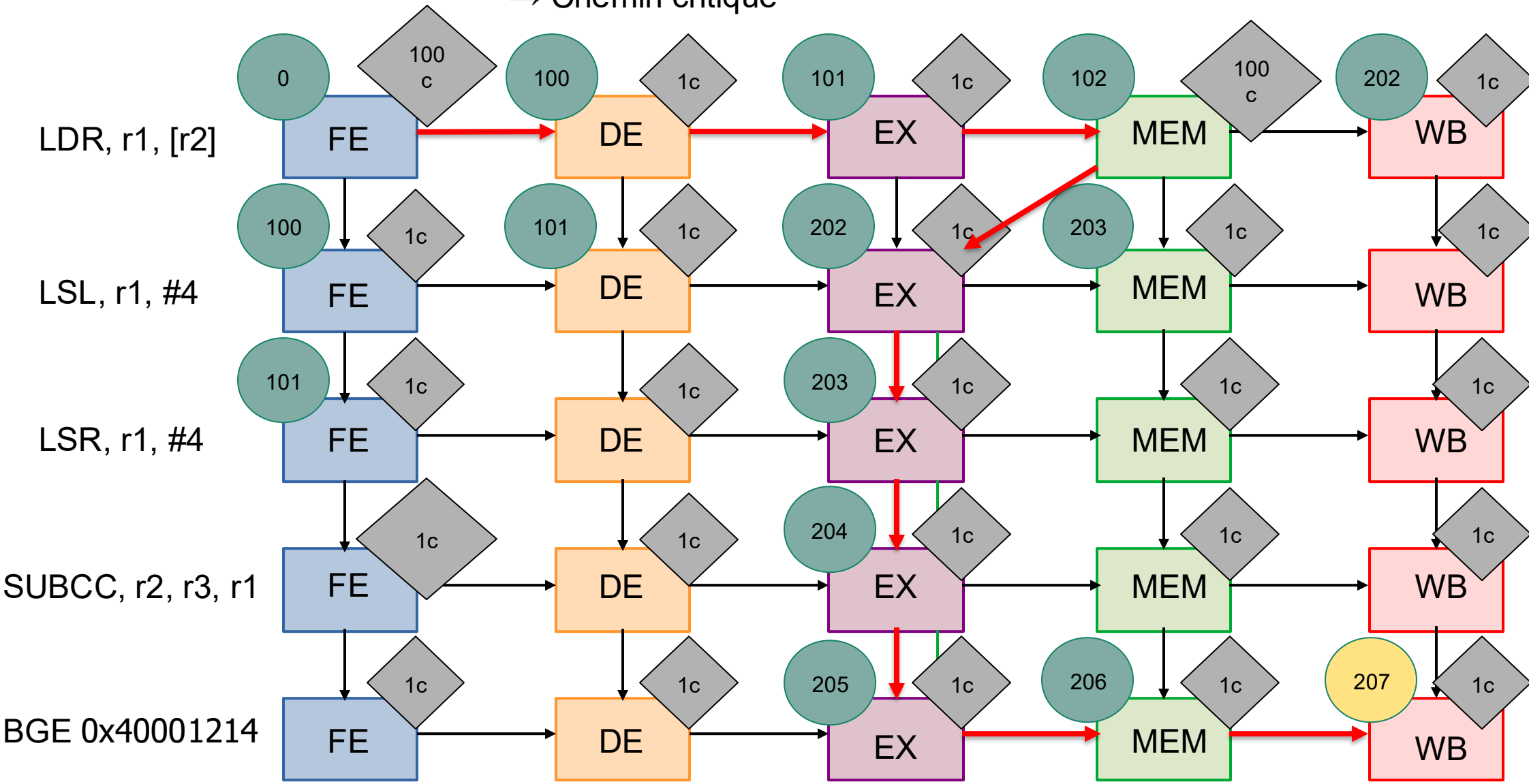
Calcul du temps du BB pour cette combinaison d'événements :



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 $t_{i,c}$
  => plusieurs variables $x_{i,c}$ pour le même BB
- Résultats des analyses statiques (e.g. l'un des LDR du BB fait un miss toutes les 5 exécutions du BB)

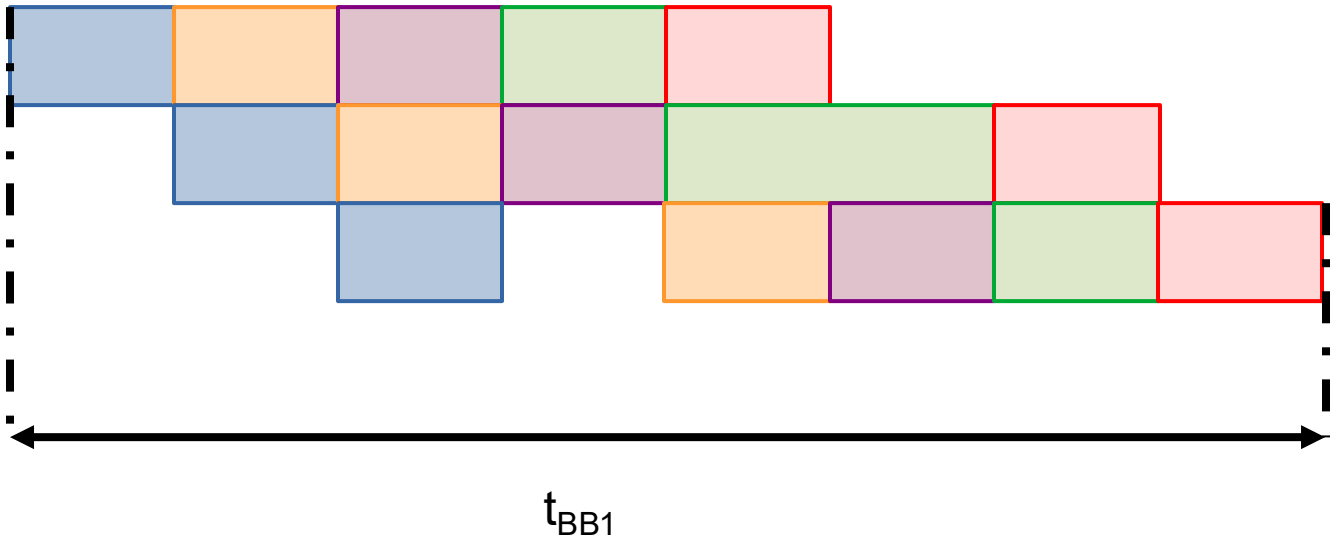
→ Compromis entre nombre de variables et précision du WCET

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

Analyse WCET et OTAWA

Quelques remarques

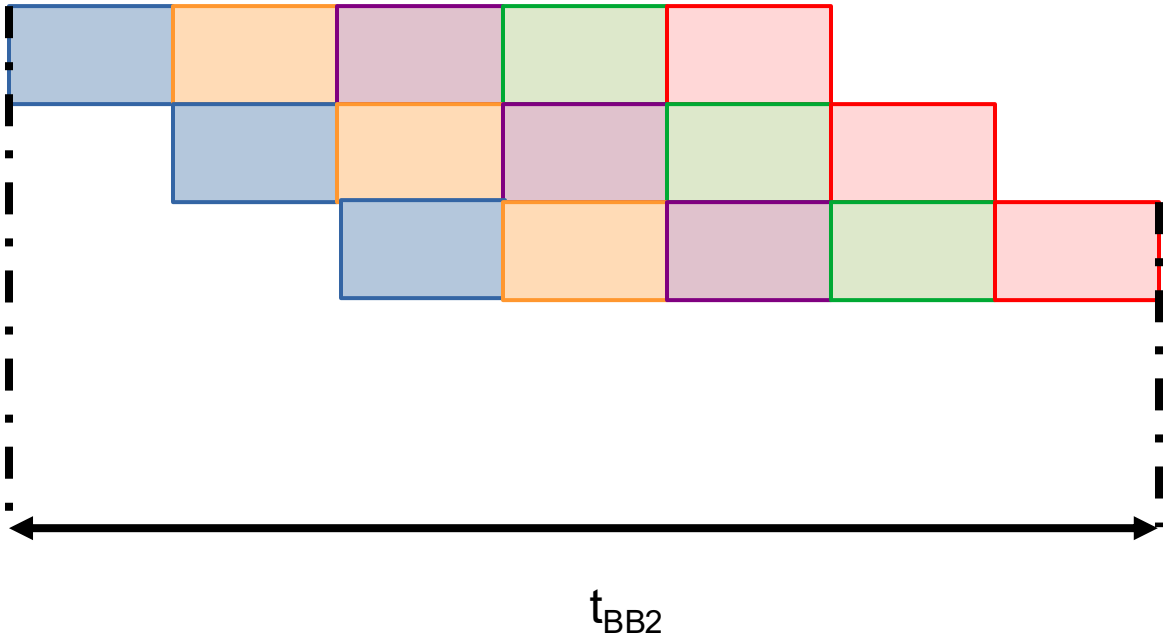
→ **Effet du pipeline entre 2 BBs consécutifs**



Analyse WCET et OTAWA

Quelques remarques

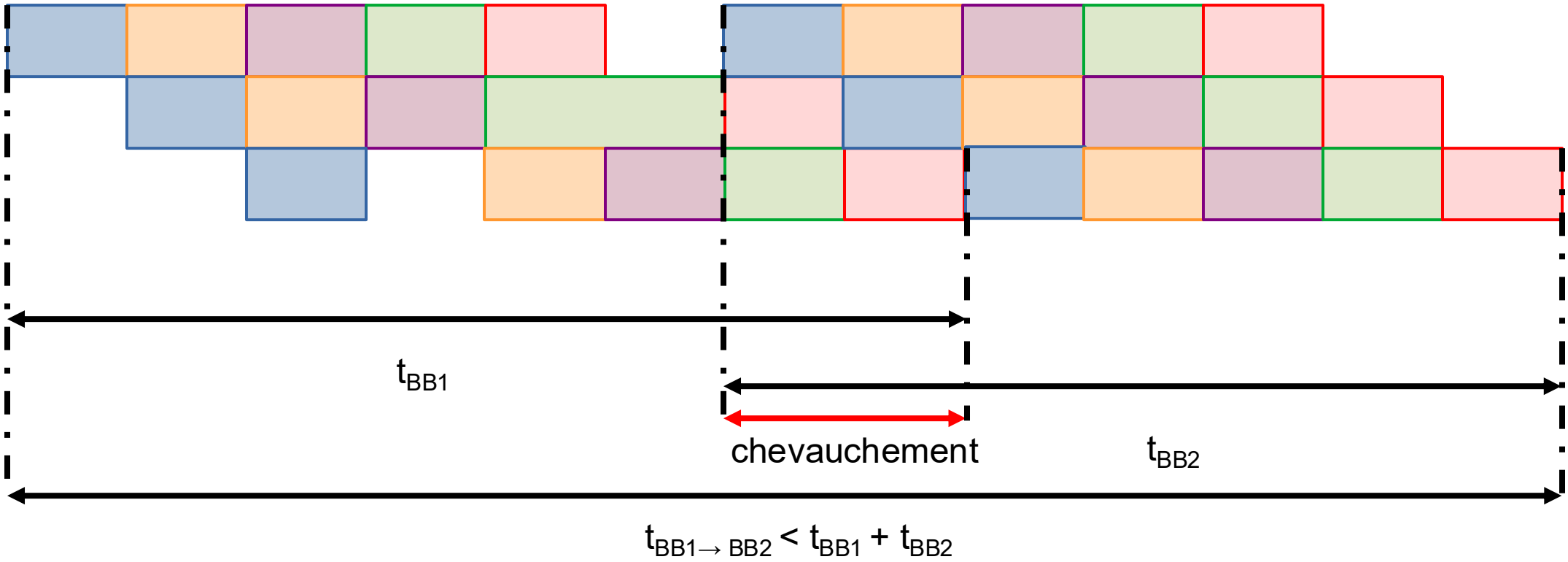
→ **Effet du pipeline entre 2 BBs consécutifs**



Analyse WCET et OTAWA

Quelques remarques

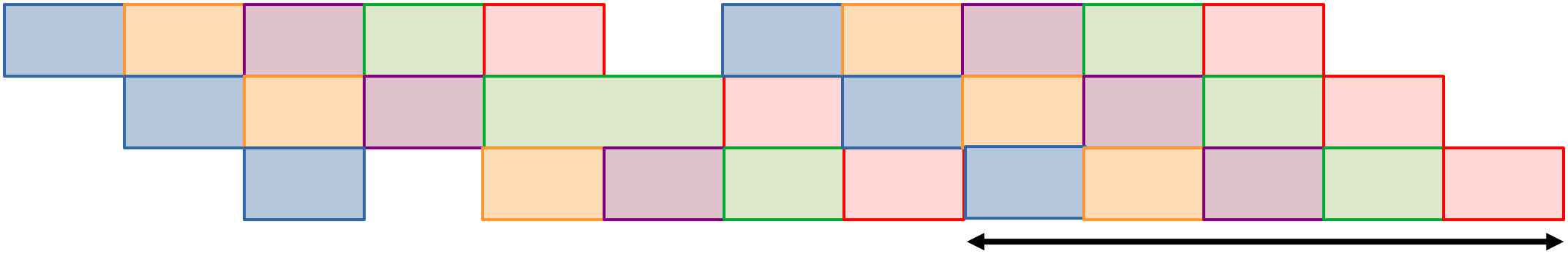
→ **Effet du pipeline entre 2 BBs consécutifs**



Analyse WCET et OTAWA

Quelques remarques

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



- On met les temps sur les arcs BB1→BB2 dans le CFG

→ On calcule les exegraphs de BB1→BB2

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 sur-approxime l'ensemble des « configurations » atteignables pour toutes les exécutions possibles (e.g. à chaque point du programme :

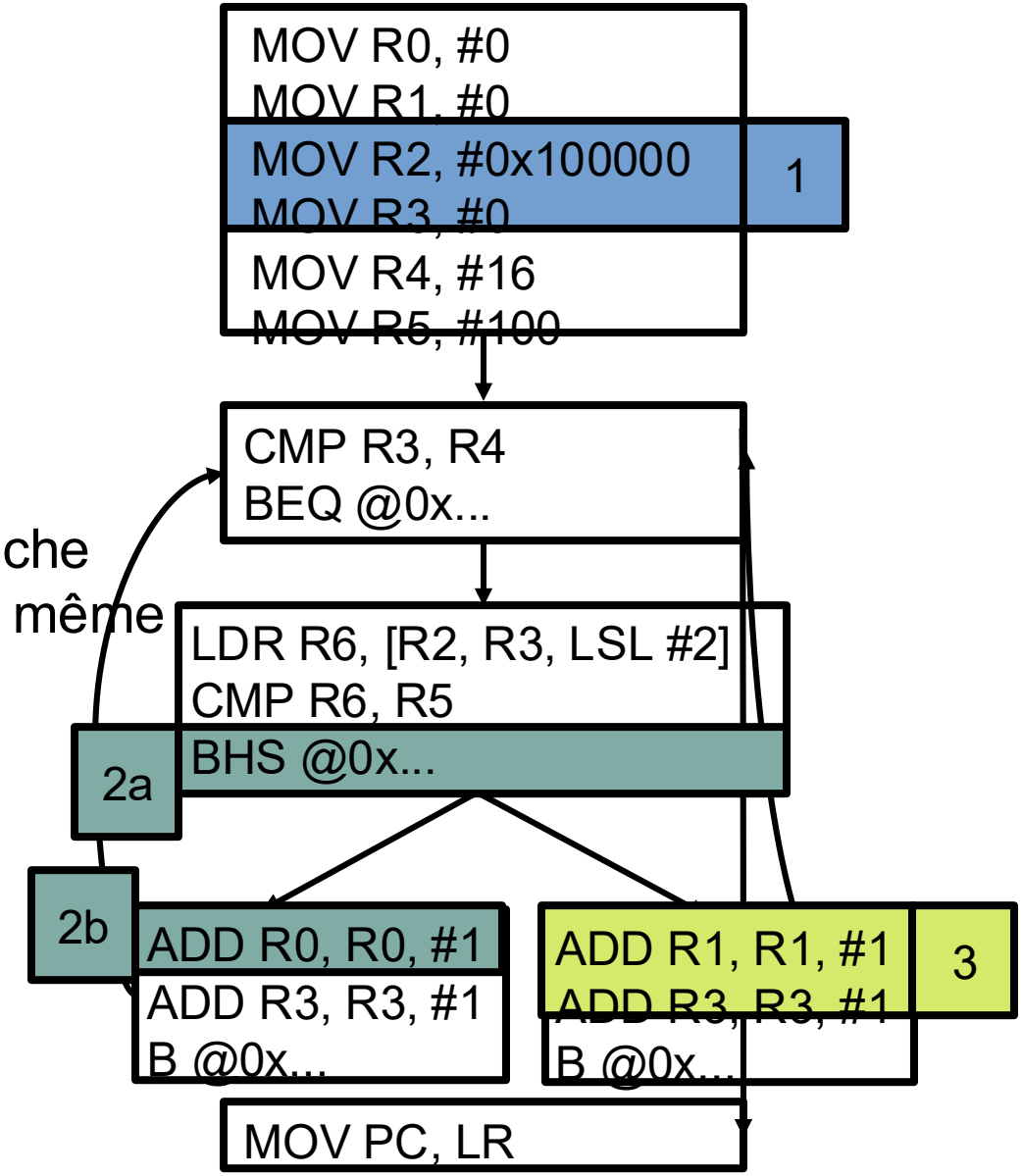
- **toutes les valeurs qu'une variable pourrait avoir**
- **tous les états dans lequel le cache de données pourrait se trouver)**

Analyse WCET et OTAWA

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

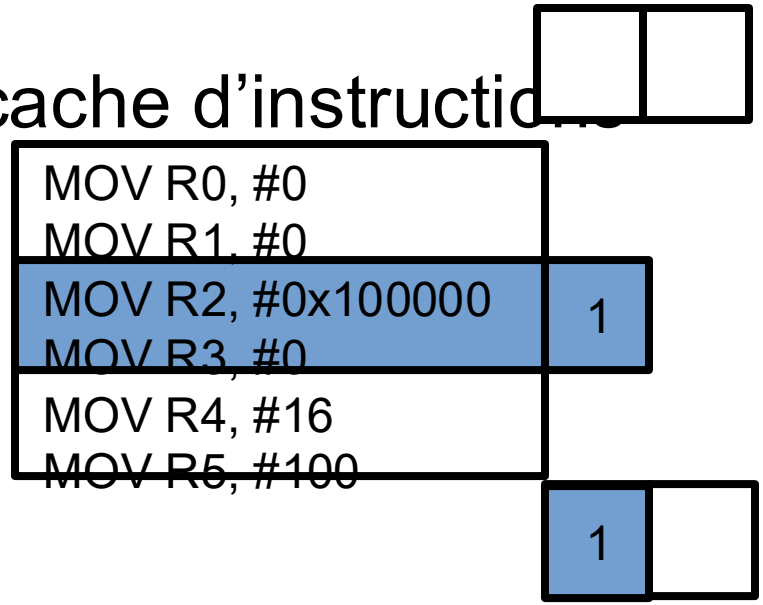
→ **Exemple :**

- 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



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**

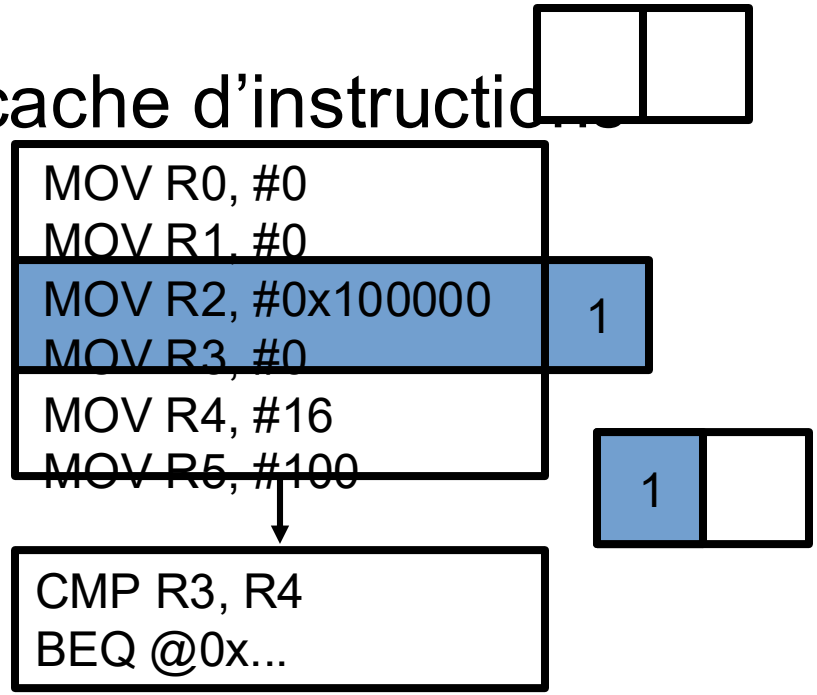
- MUST
- MAY
- PERSISTENCE

→ **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**

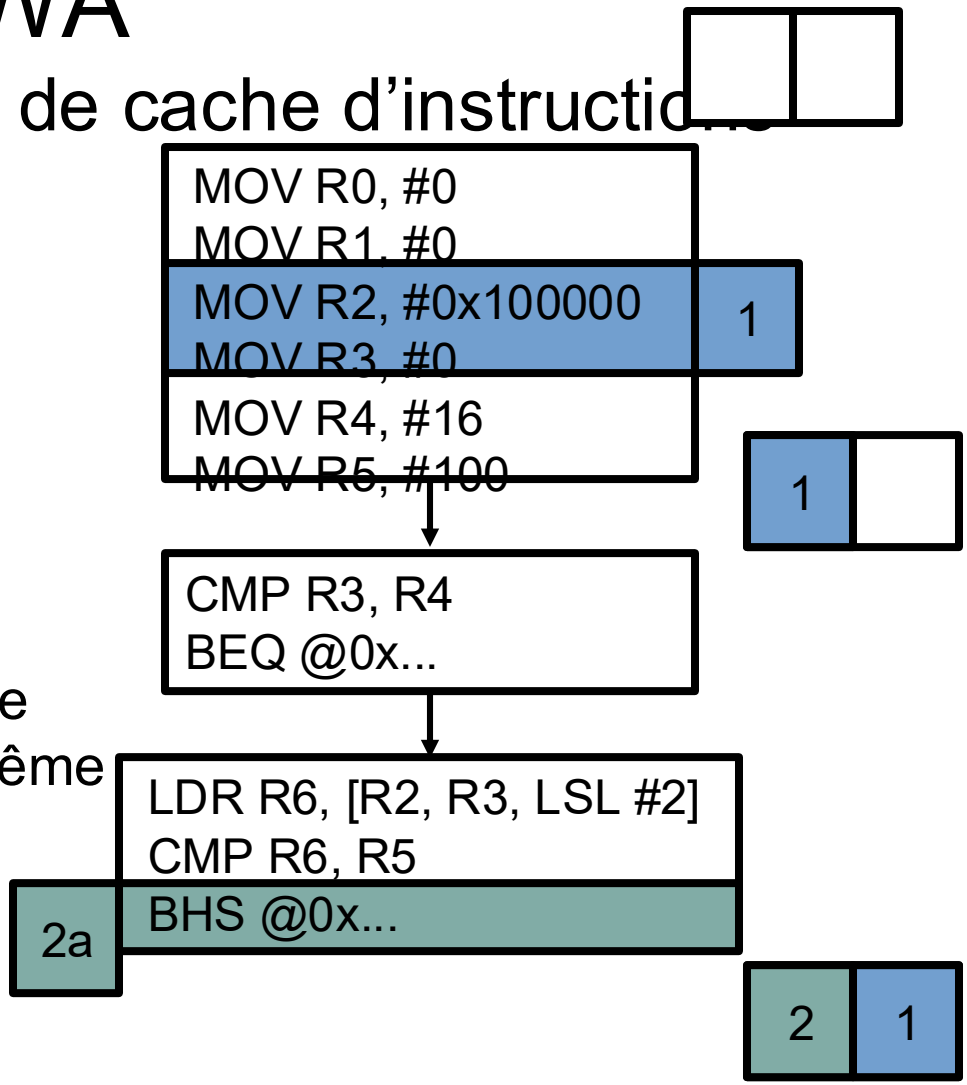
- MUST
- MAY
- PERSISTENCE

→ **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 :**

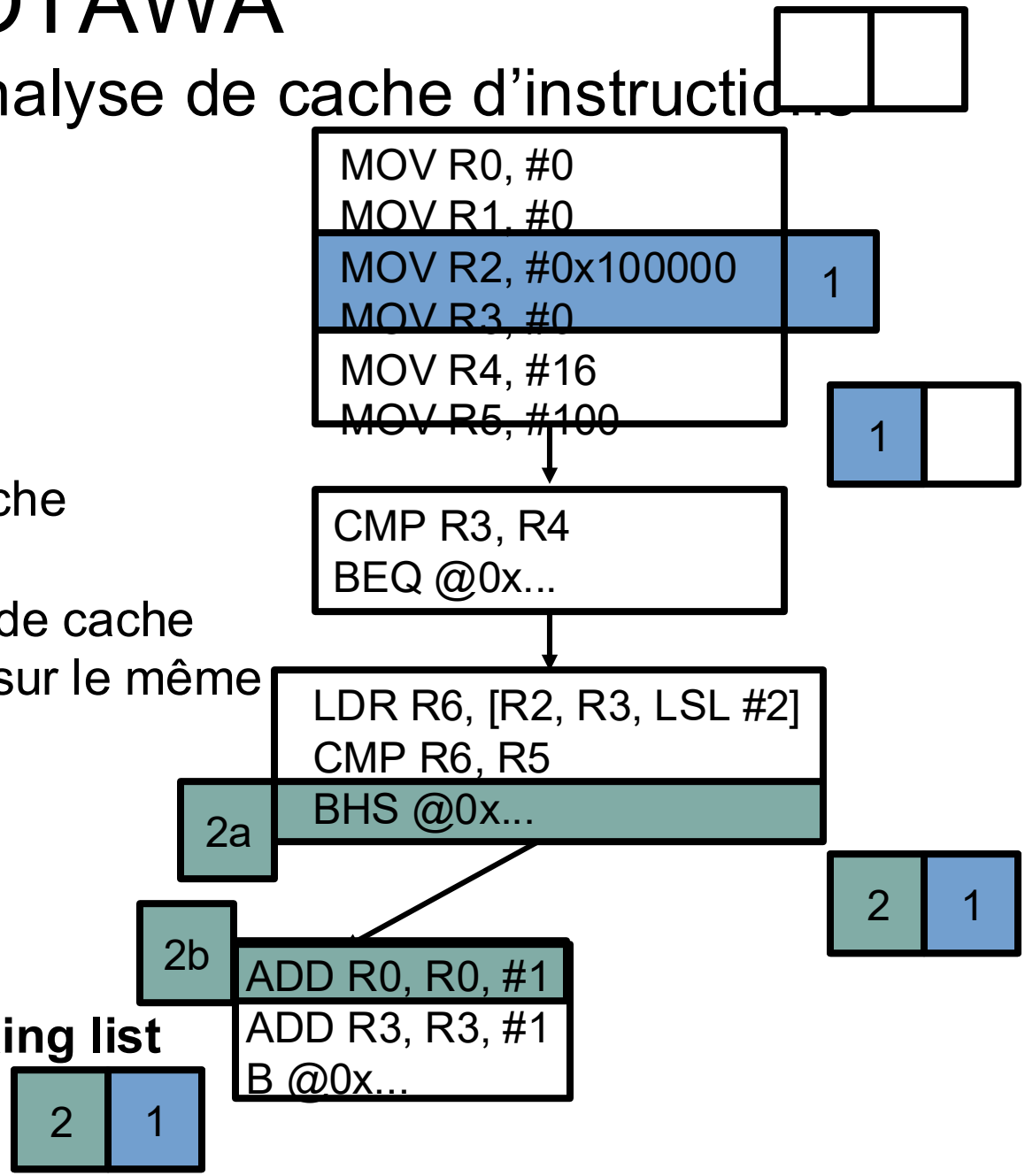
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **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 :**

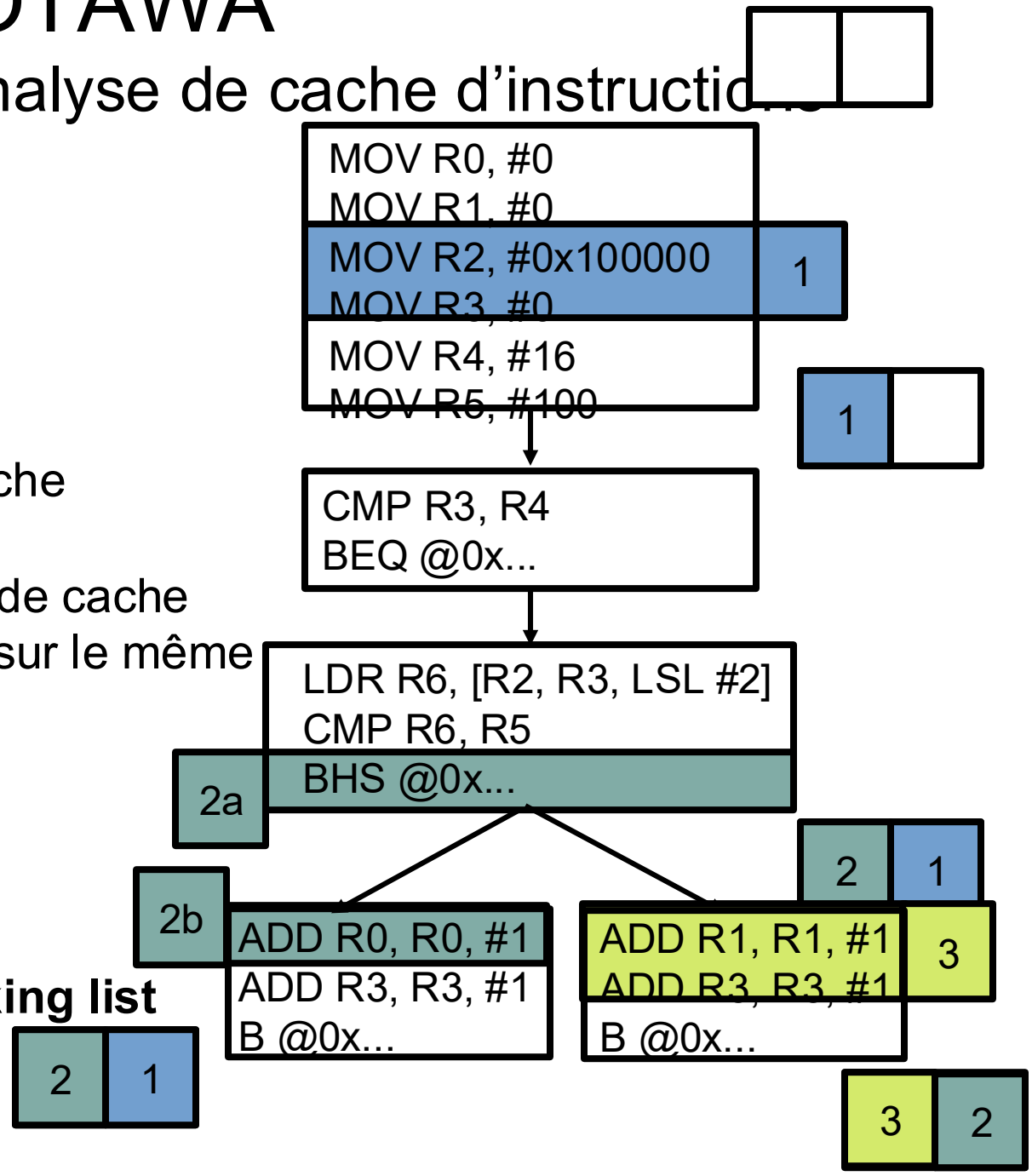
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **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 :**

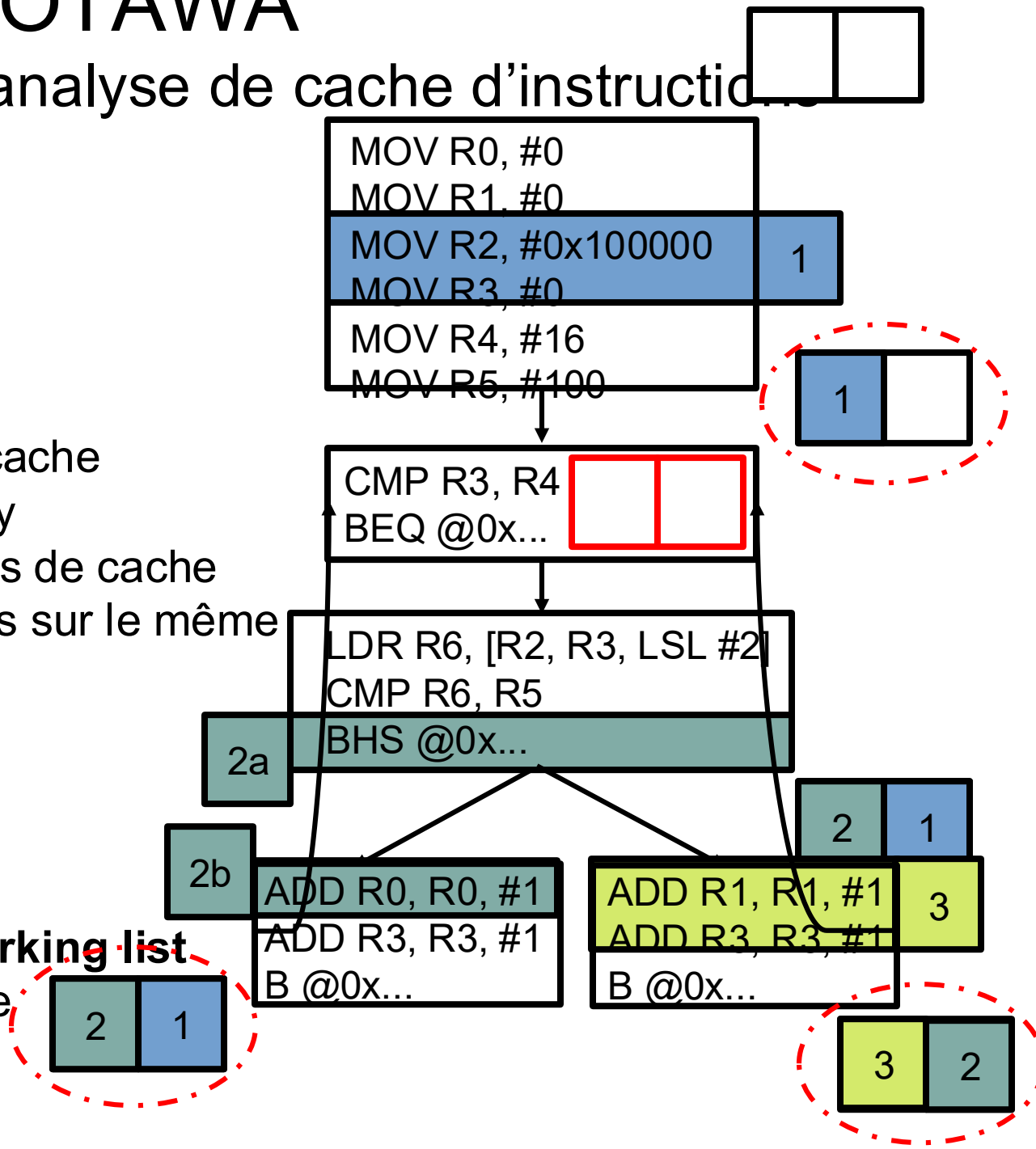
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **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 :**

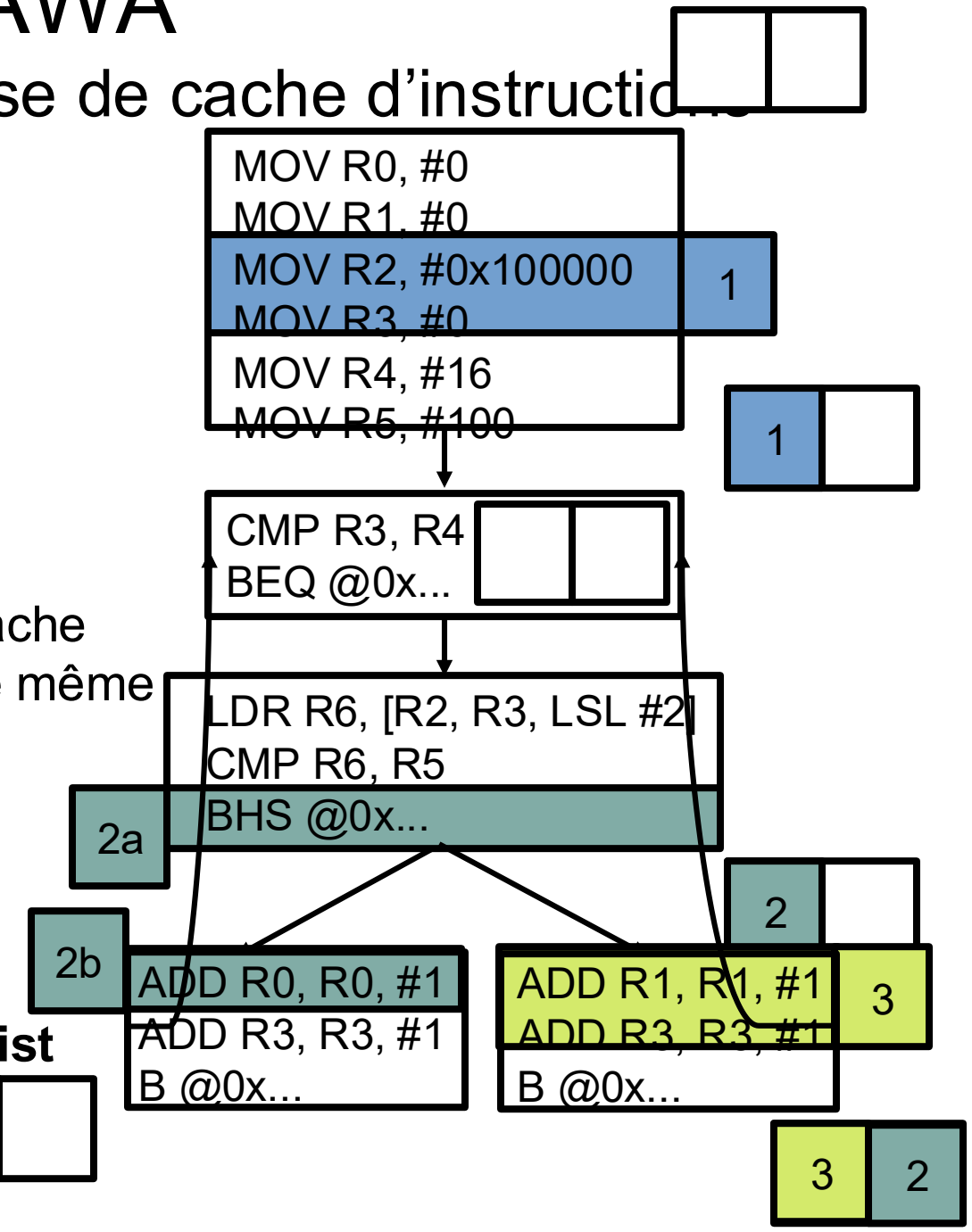
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **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 :**

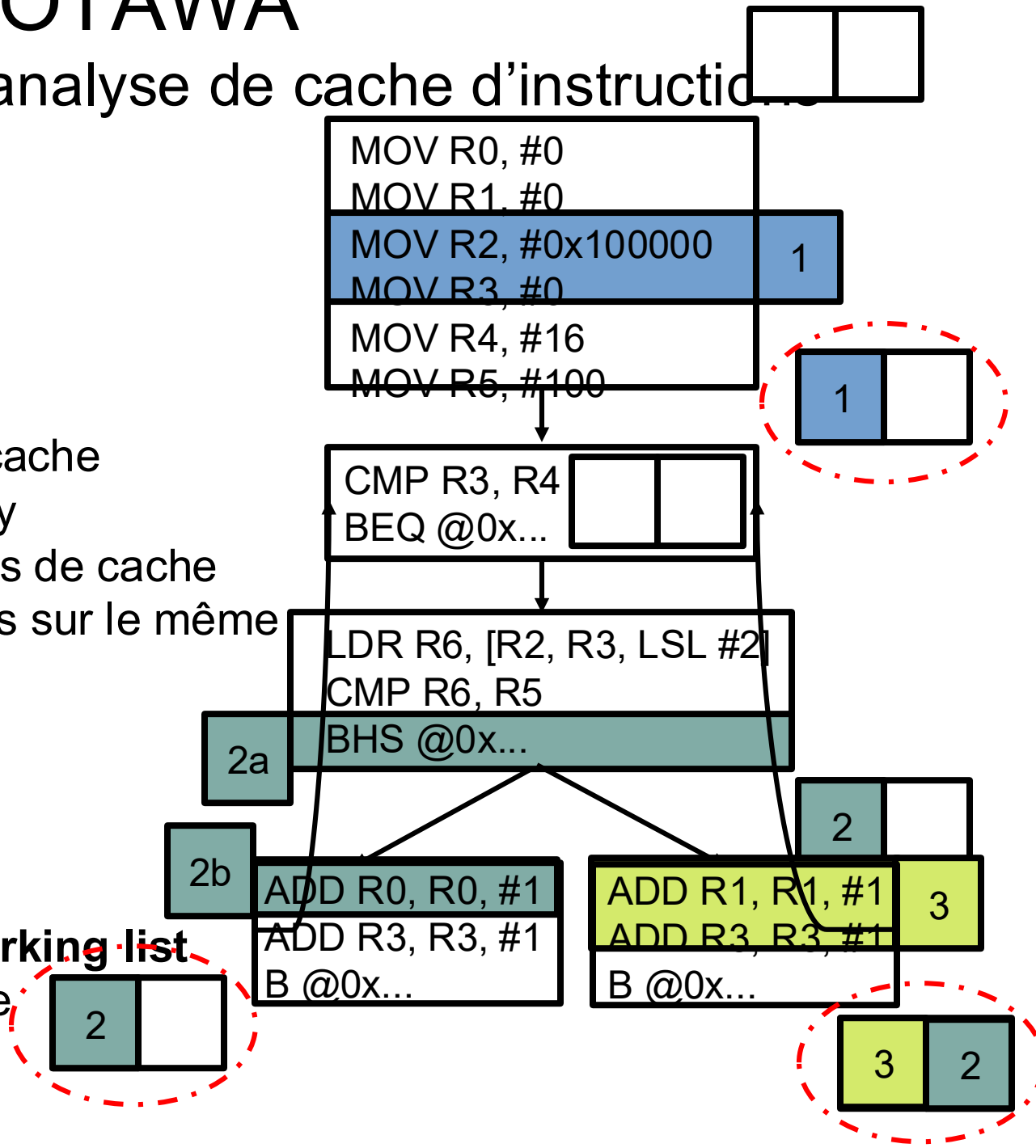
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **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 :**

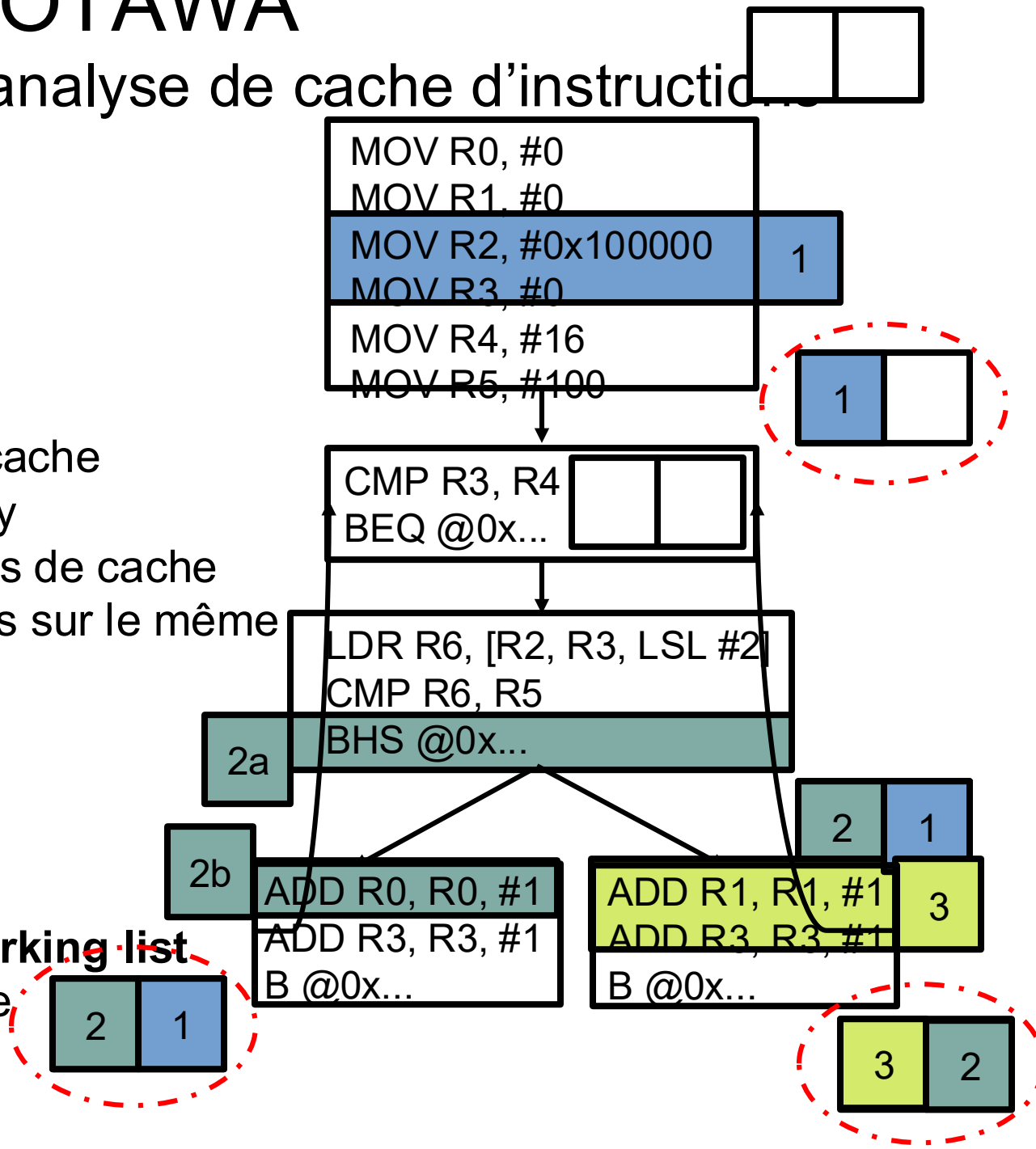
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **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 :**

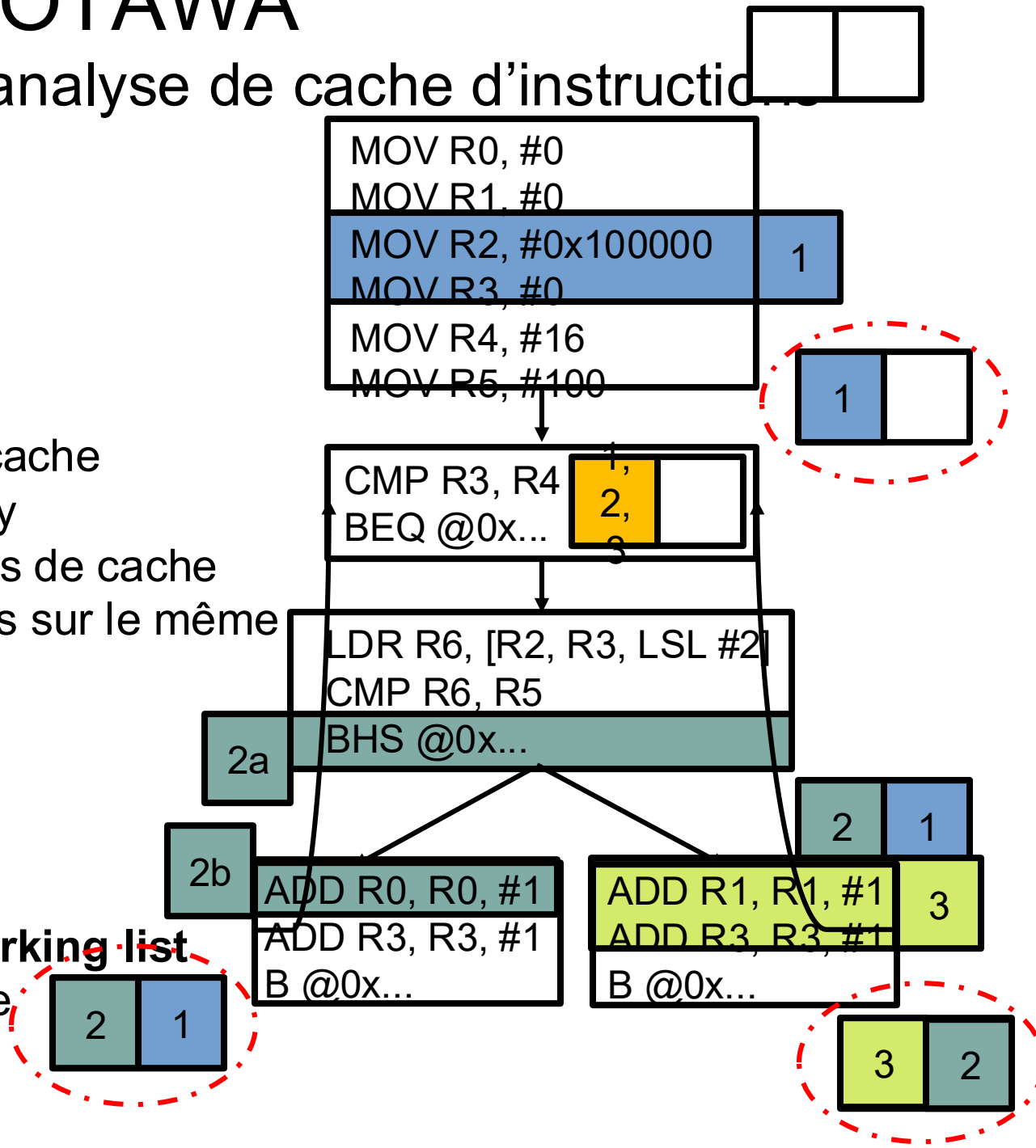
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **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 :**

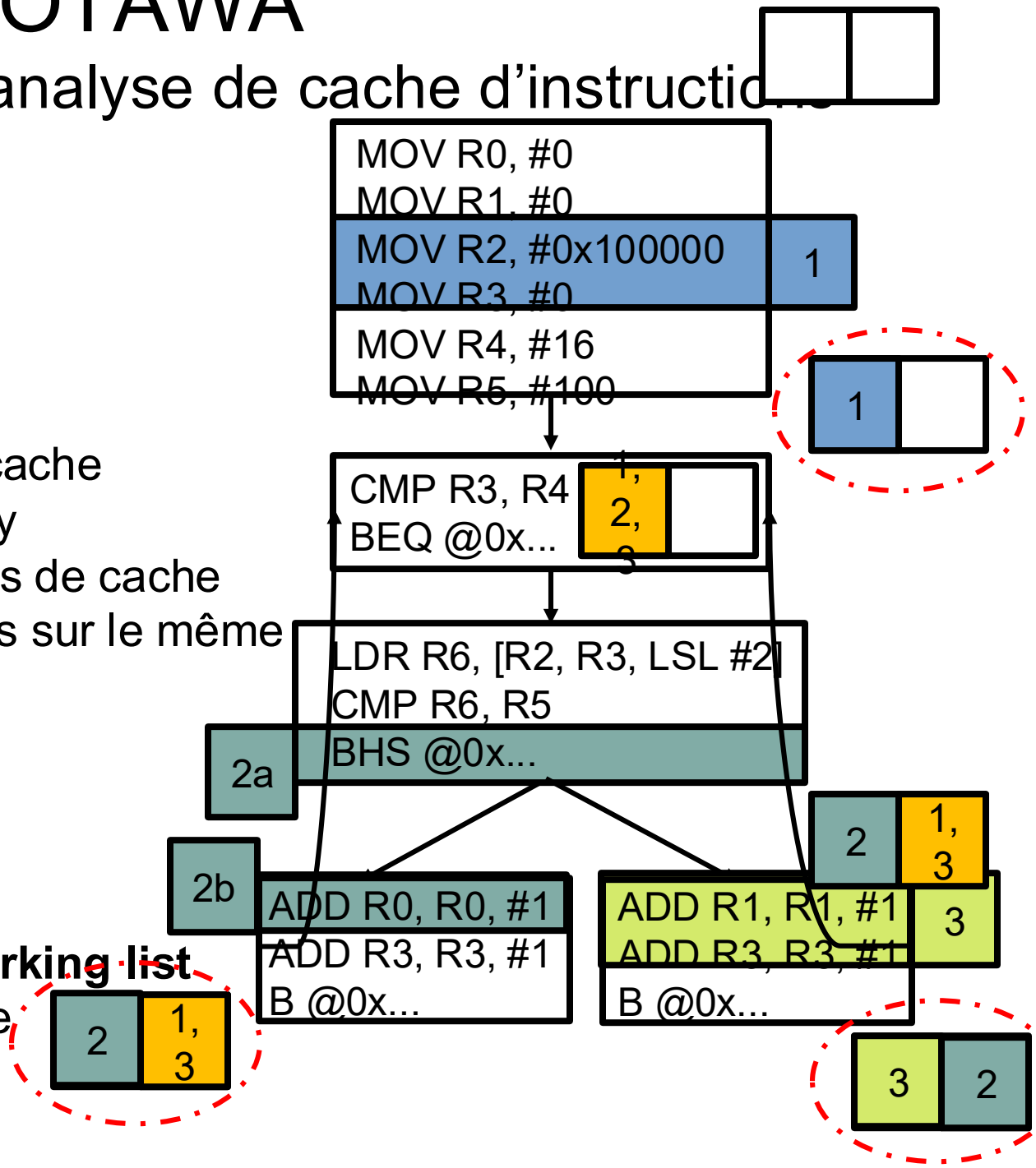
- 2-way set associative cache
- LRU replacement policy
- on considère les 3 blocs de cache 1, 2 et 3 qui sont mappés sur le même ensemble

→ **Plusieurs types d'analyse**

- MUST
- MAY
- PERSISTENCE

→ **Parcours du graphe par working list**

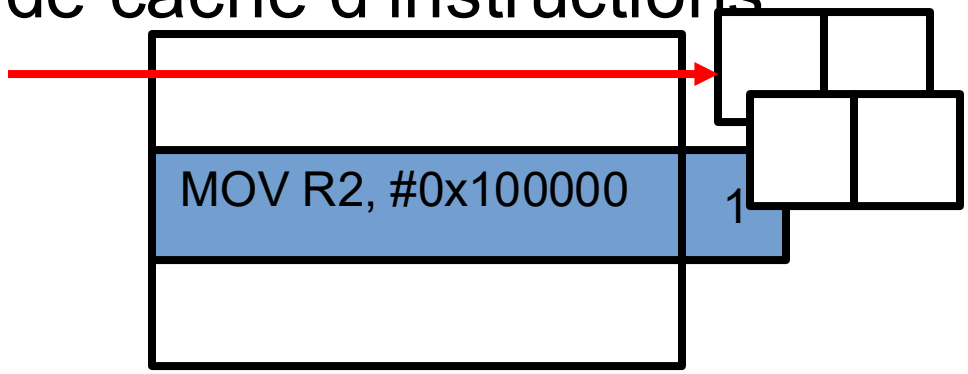
- Recherche de point fixe



Analyse WCET et OTAWA

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

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



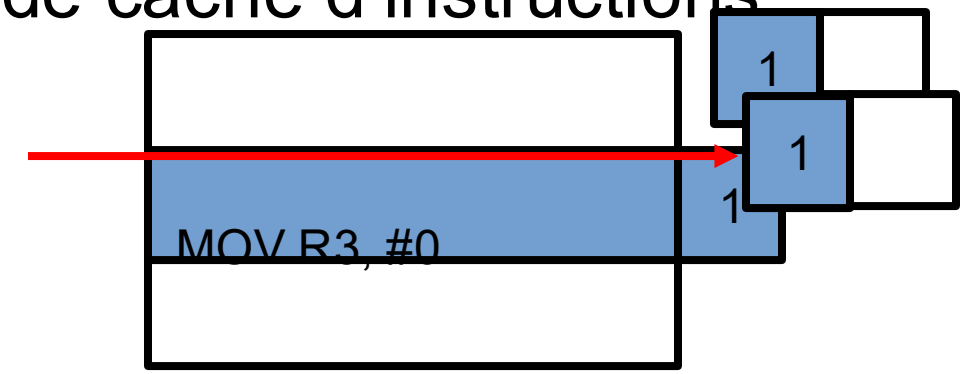
→ **Et après ?**

- 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

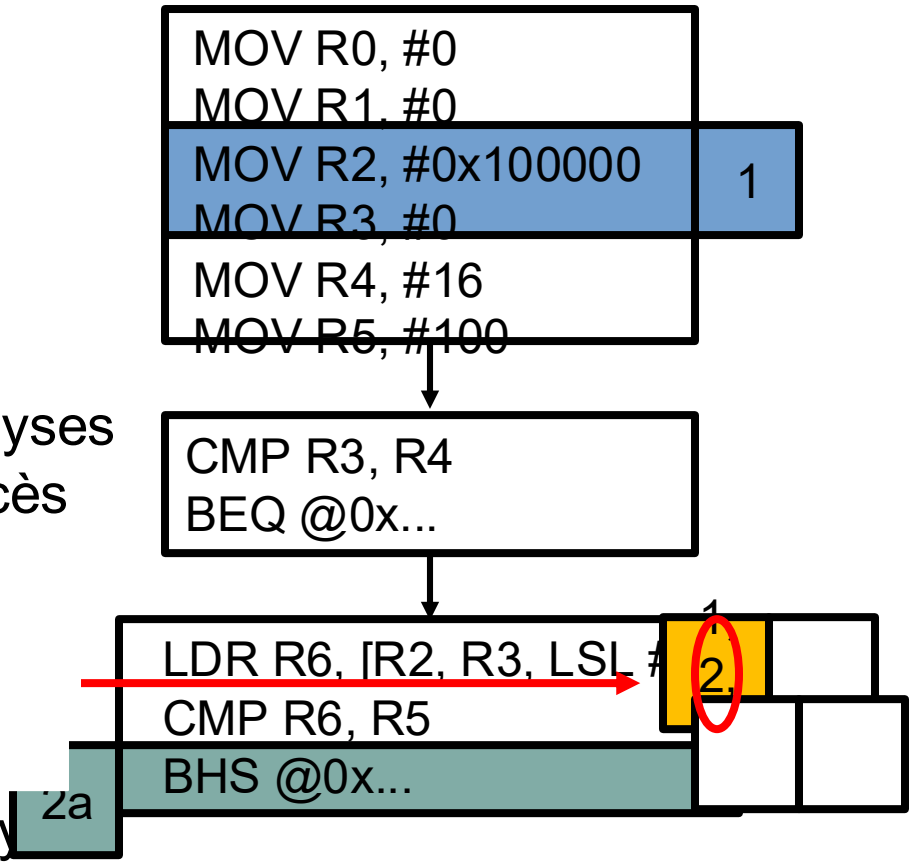


→ **Et après ?**

- 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'analy
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

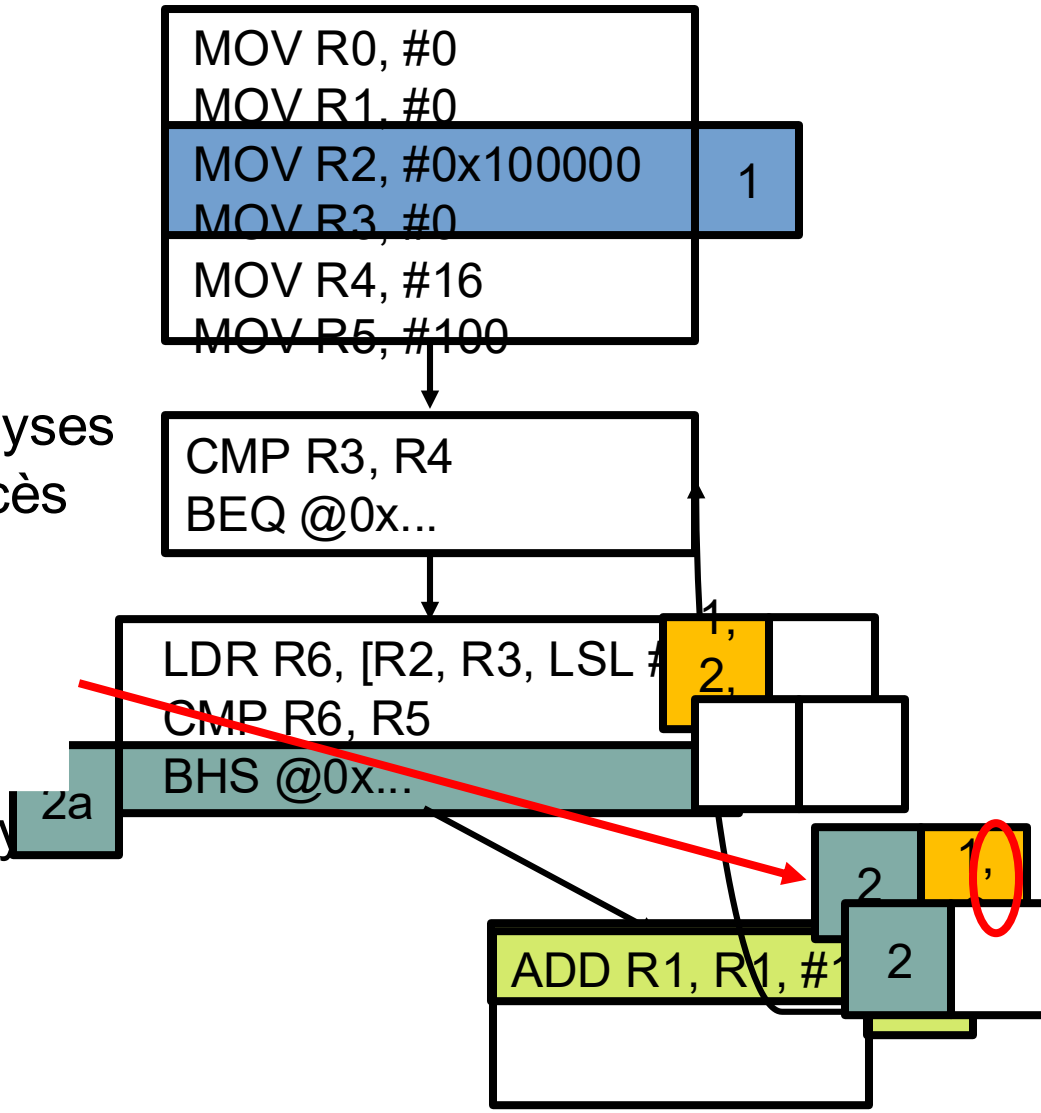
→ **Et après ?**

- 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

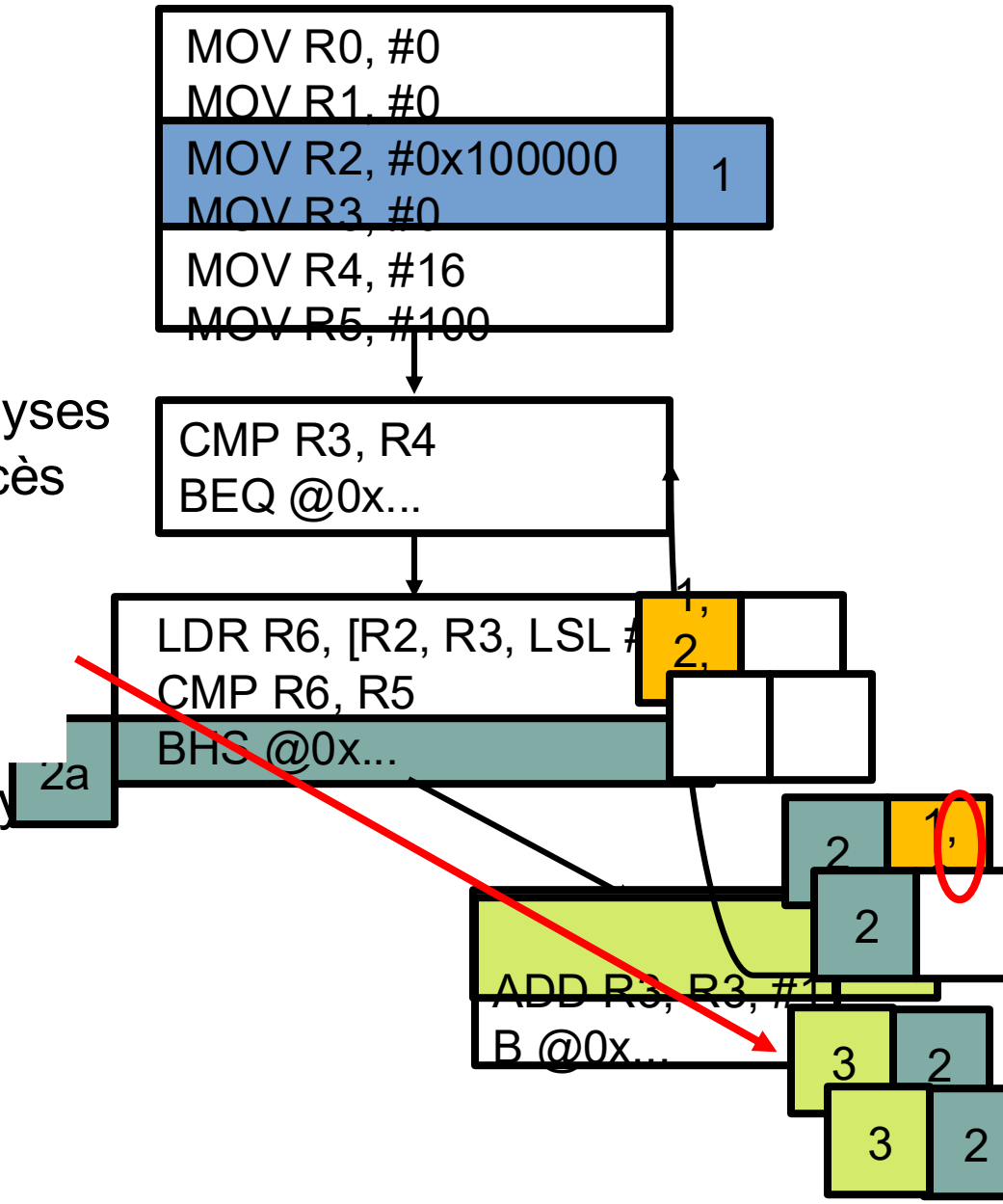
Illustration : un mot sur l'analyse de cache d'instructions

→ **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'analy PERSISTENCE, on obtient des informations pour borner des variables dans IPET



Analyse WCET et OTAWA

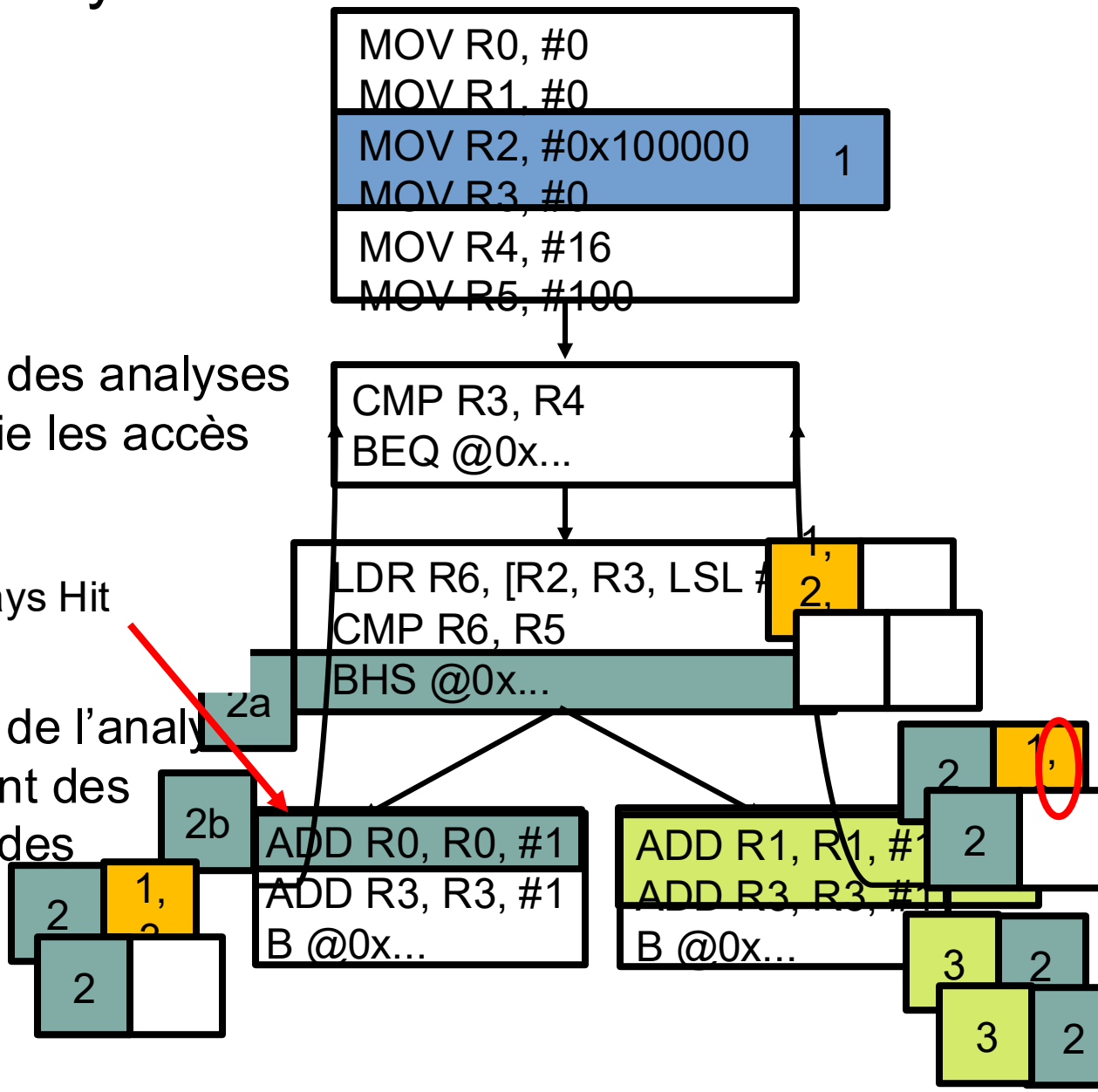
Illustration : un mot sur l'analyse de cache d'instructions

→ **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
- Des outils d'analyse de code (désassembleurs, analyses statiques)
- Des bibliothèques pour développer ses propres analyses
- Open source

WCET ET ORDONNANCEMENT : PROBLEMES OUVERTS

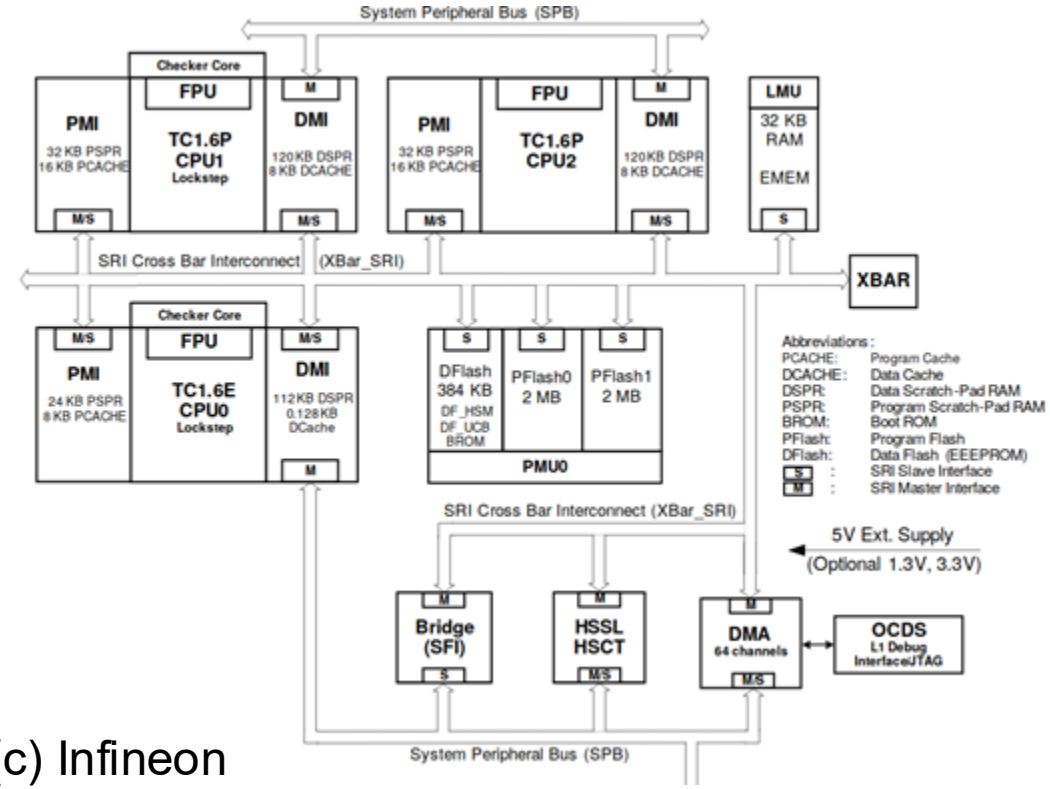
Multicoeur et interférences

→ **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**

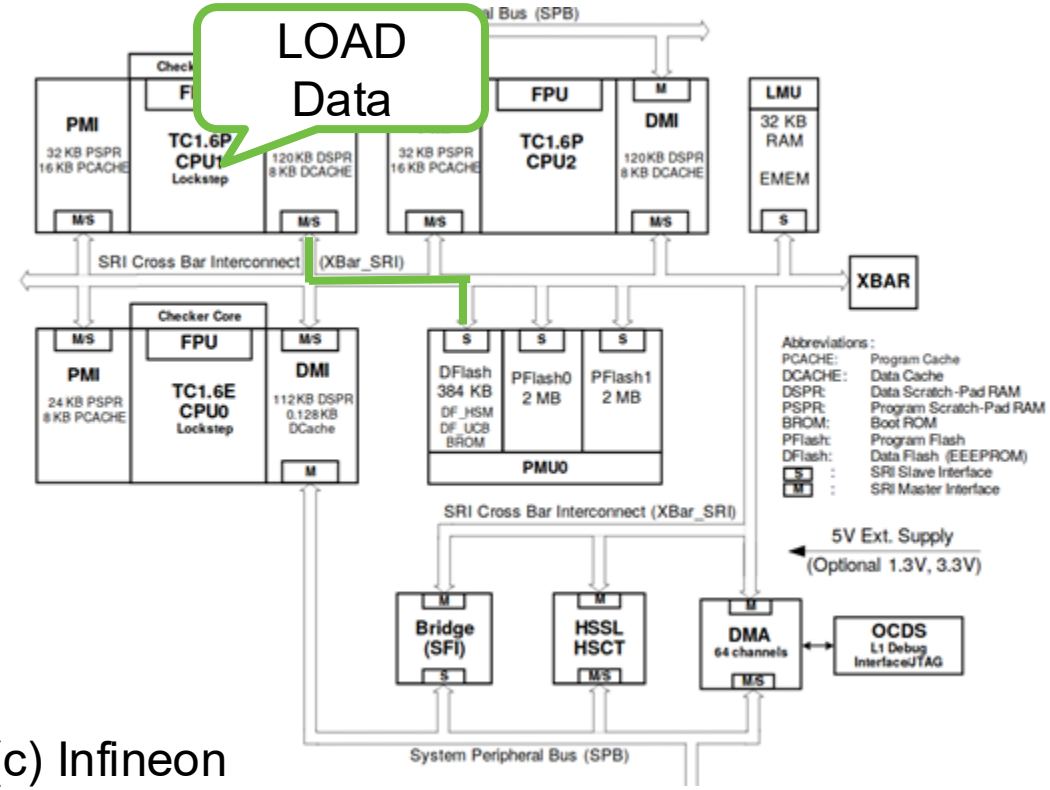


Infineon TriCore
TC275

Infineon website (c) Infineon

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

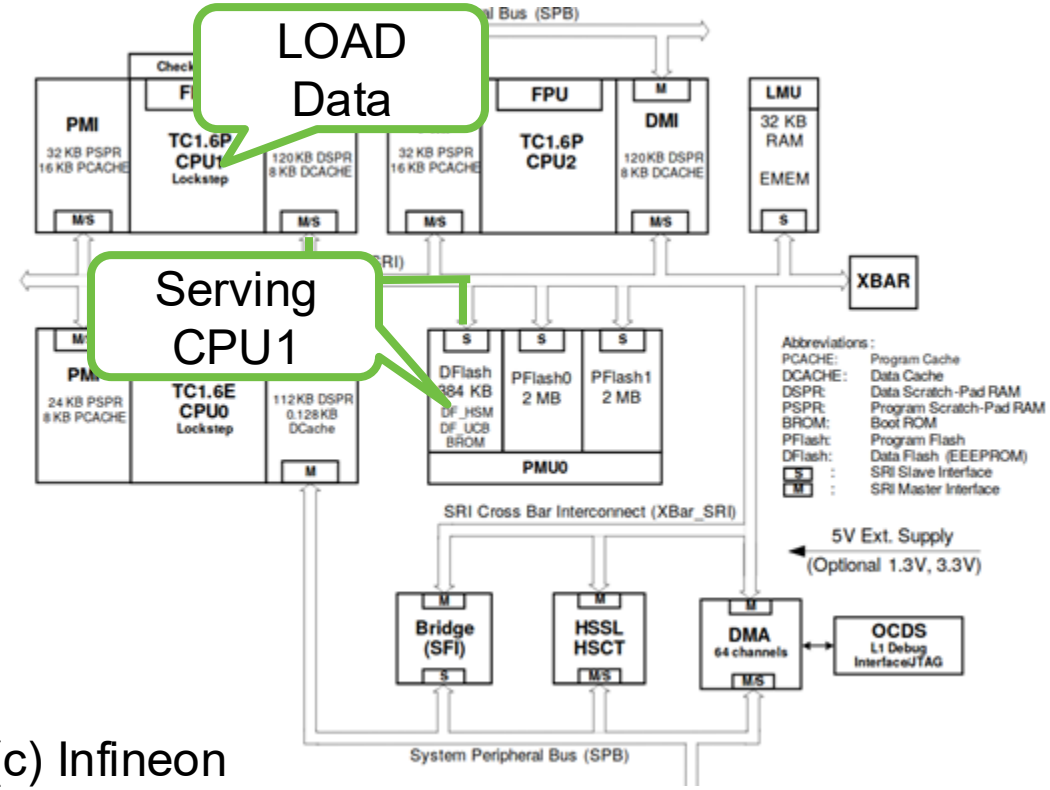


Infineon TriCore
TC275

Infineon website (c) Infineon

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

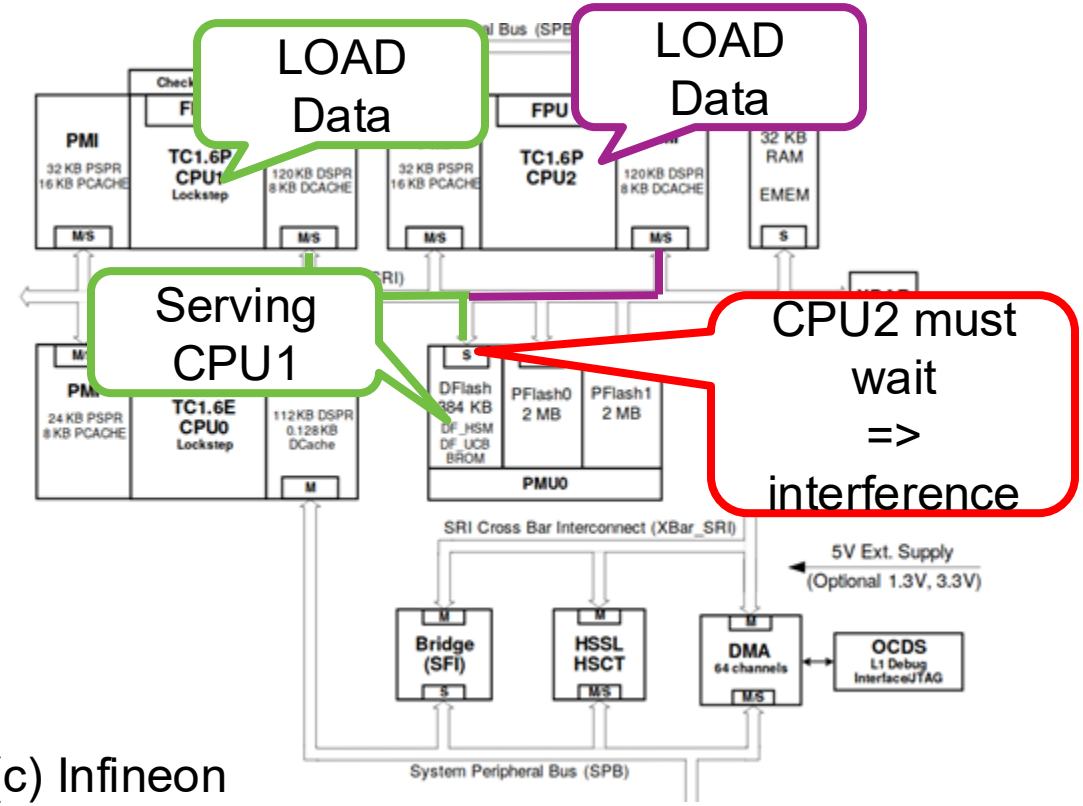


Infineon TriCore
TC275

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

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

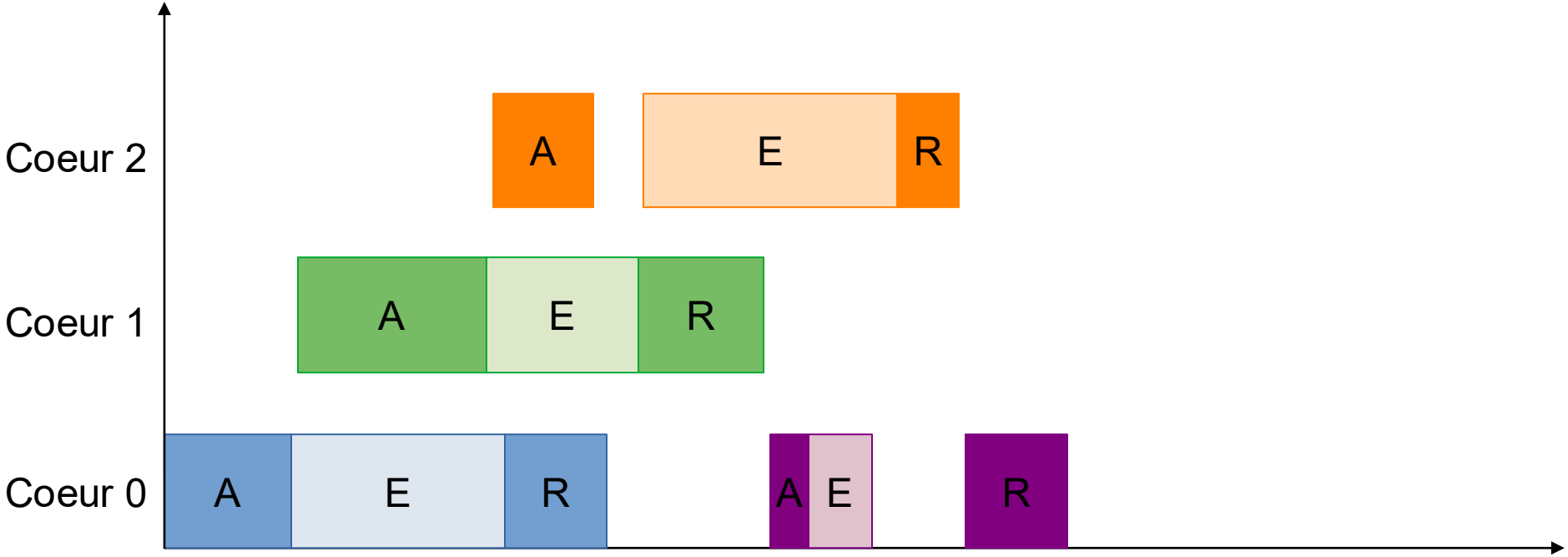
→ Seules les phases A et R sollicitent l'interconnect

→ 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**

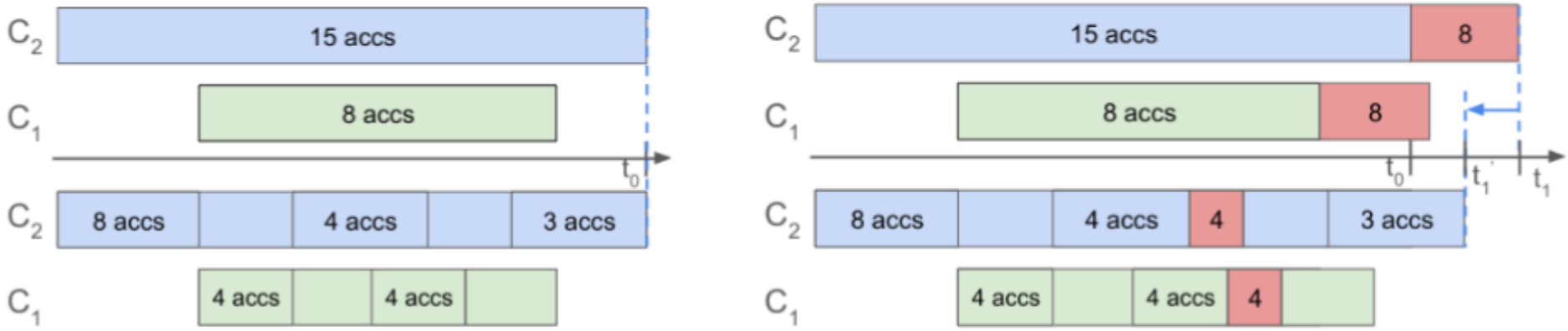
- solutions compilation : AER/REW



Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

- solutions compilation / analyse : multiphases



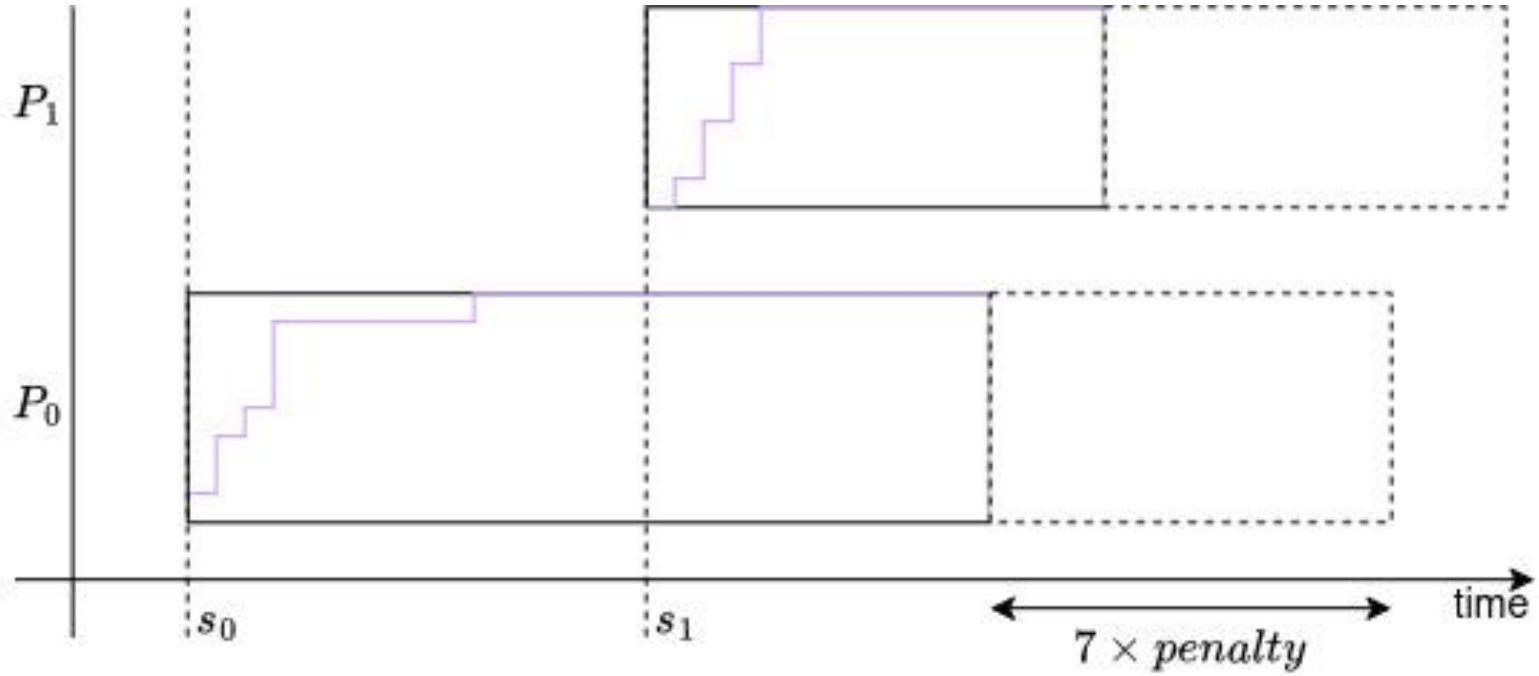
(a) Before interference analysis

(b) After interference analysis

Multicoeur et interférences

→ **Exécution sur multicoeurs à mémoire partagée**

- solutions analyse : courbes d'arrivée d'accès



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