Analyse statique

Opérations sur la structure d’un programme

Analyse statique

- Pourquoi ?
  - Améliorer les performances
  - Trouver des problèmes
- Quand/Comment ?
  - Avant la compilation (sur du source)
  - Pendant la compilation (source, ou IR…)
  - Sur un programme binaire (ou bytecode...)

Plan

- Blocs de base (Basic Blocks)
- Graphe de flot de contrôle (CFG)
- Opérations courantes de CFG
- Construction de CFG

Blocs de base

- Définition :
  - Séquence d’instructions contiguë
  - Branchements entrants uniquement sur la 1ère instruction
  - Branchement sortant uniquement sur la dernière instruction

Blocs de base

Exemple (code fictif):

```
00 :    MOV RAX, 0x00
01 :    MOV RBX, 0x42
02 :    CMP RCX, 0x45
03 :    JNE 06
04 :    ADD RAX, RBX
05 :    ADD RBX, RCX
06 :    MOV RDI, 0x55
07 :    SUB RDI, RCX
```

Blocs de base

Exemple (code fictif):

```
00 :    MOV RAX, 0x00
01 :    MOV RBX, 0x42
02 :    CMP RCX, 0x45
03 :    JNE 06
04 :    ADD RAX, RBX
05 :    ADD RBX, RCX
06 :    MOV RDI, 0x55
07 :    SUB RDI, RCX
```

Blocs de base

Exemple (code fictif):

```
00 :    MOV RAX, 0x00
01 :    MOV RBX, 0x42
02 :    CMP RCX, 0x45
03 :    JNE 06
04 :    ADD RAX, RBX
05 :    ADD RBX, RCX
06 :    MOV RDI, 0x55
07 :    SUB RDI, RCX
```

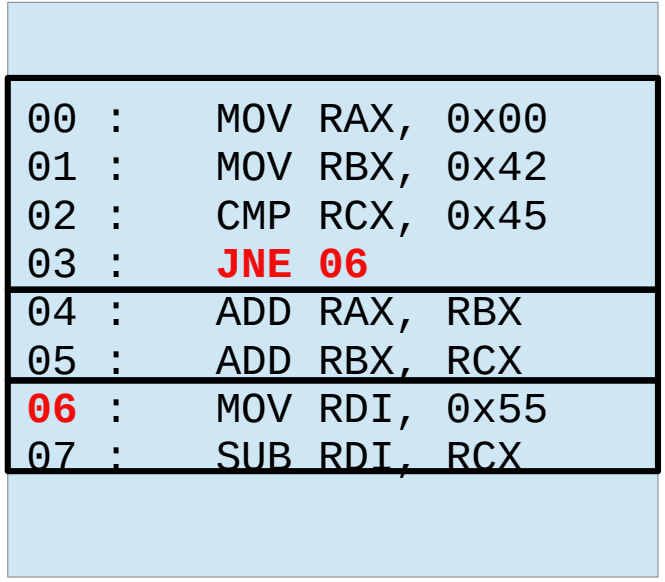
Blocs de base

Exemple (code fictif):

```
00 :    MOV RAX, 0x00
01 :    MOV RBX, 0x42
02 :    CMP RCX, 0x45
03 :    JNE 06
04 :    ADD RAX, RBX
05 :    ADD RBX, RCX
06 :    MOV RDI, 0x55
07 :    SUB RDI, RCX
```

Blocs de base

Exemple (code fictif):



Graphe de flot de contrôle

- Définition :
  - Graphe dont les nœuds sont des blocs de base
  - Arc n1 → n2 ssi on peut passer directement de n1 à n2
  - Si besoin, nœuds virtuels « entrée » et « sortie ».

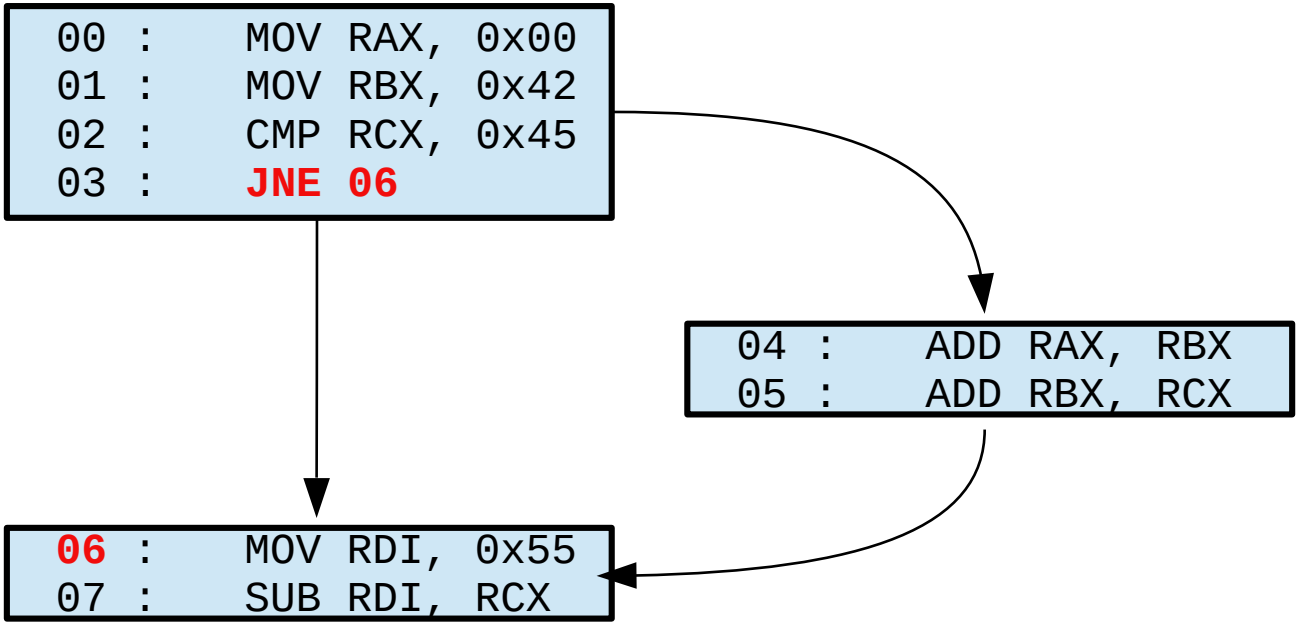
Graphe de flot de contrôle

Exemple (code fictif):

```
00 :    MOV RAX, 0x00
01 :    MOV RBX, 0x42
02 :    CMP RCX, 0x45
03 :    JNE 06
04 :    ADD RAX, RBX
05 :    ADD RBX, RCX
06 :    MOV RDI, 0x55
07 :    SUB RDI, RCX
```

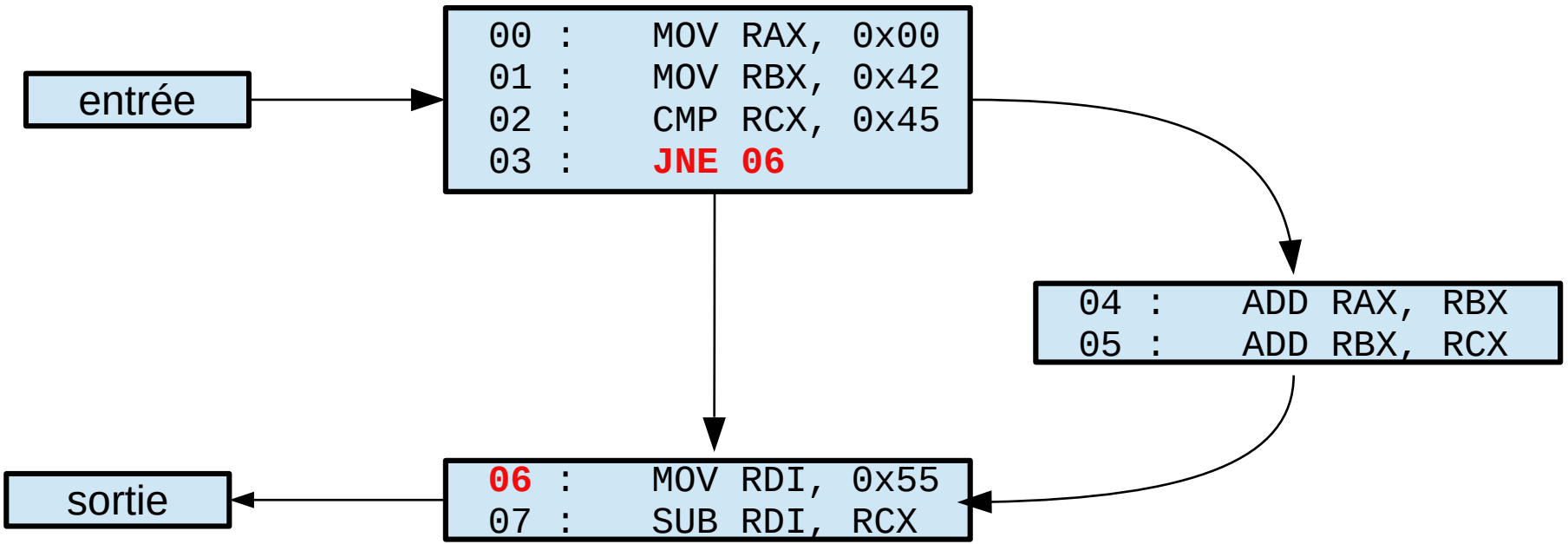
Graphe de flot de contrôle

Exemple (code fictif):



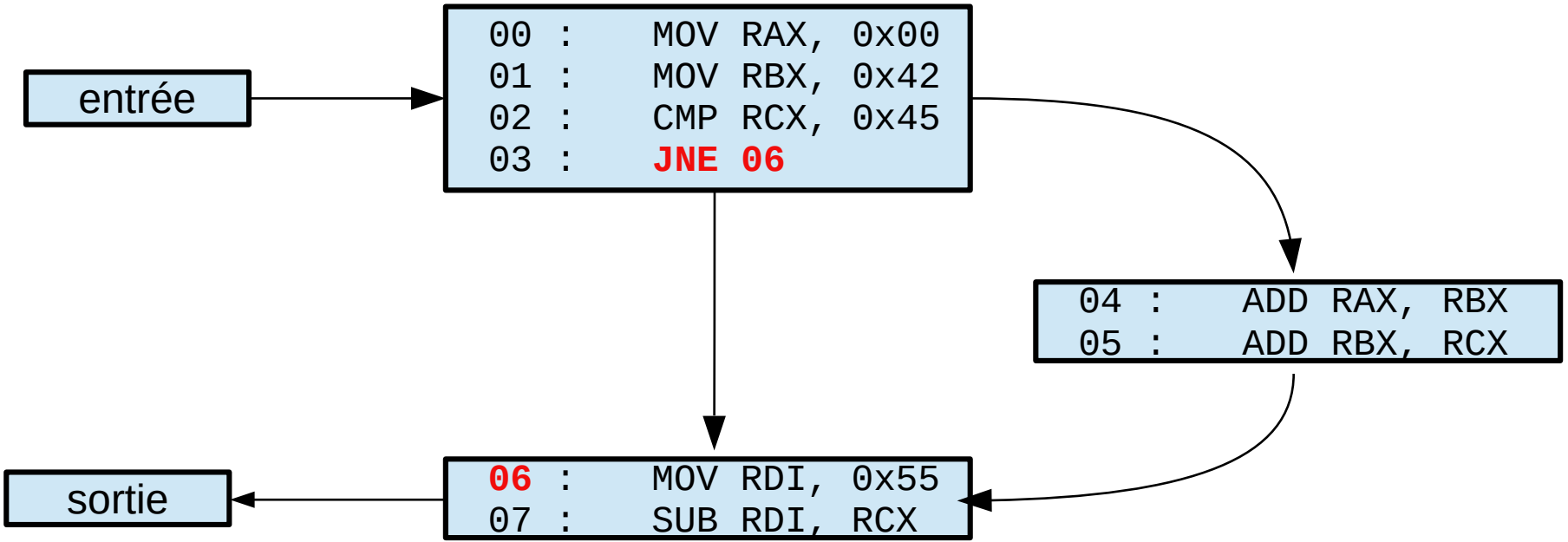
Graphe de flot de contrôle

Exemple (code fictif):



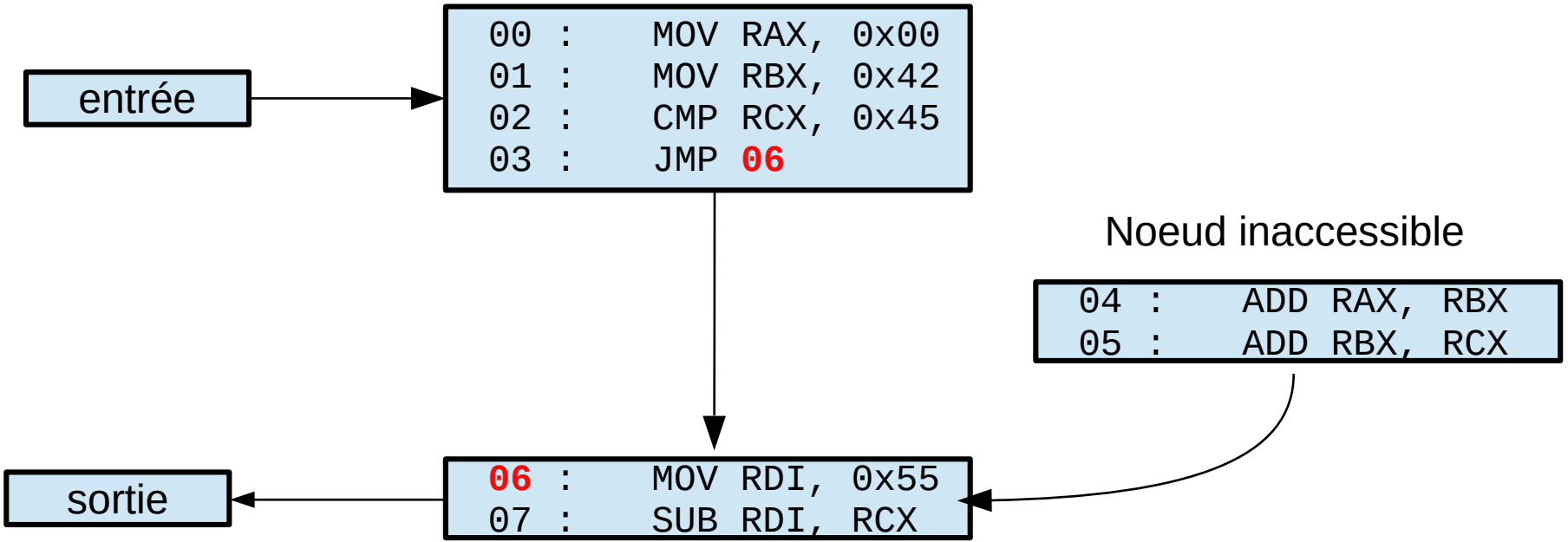
Opérations CFG : nœuds inaccessibles ?

- Problème : est ce qu'il y a des nœuds inaccessibles ?



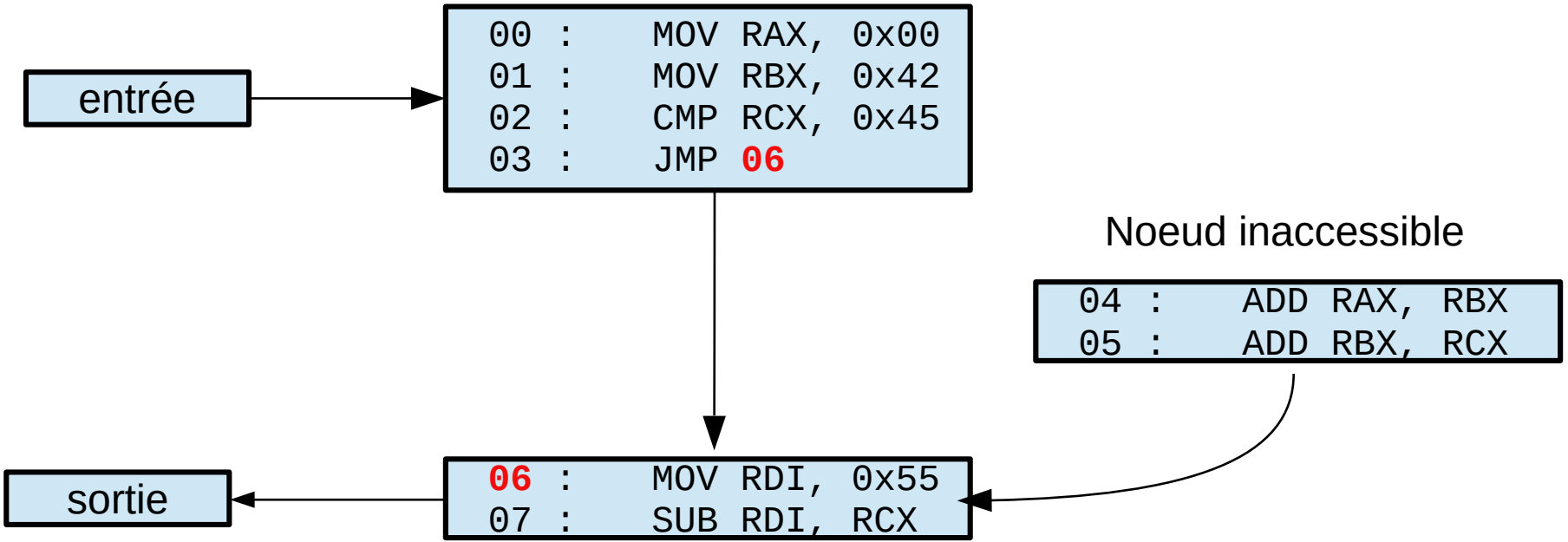
Opérations CFG : nœuds inaccessibles ?

- Problème : est ce qu'il y a des nœuds inaccessibles ?



Opérations CFG : nœuds inaccessibles ?

- Problème : est ce qu'il y a des nœuds inaccessibles ?



Opérations CFG : nœuds inaccessibles ?

- Vérifier que tout les nœuds sont accessibles :
  - Parcours récursif du graphe (depuis l’entrée)
  - On oublie pas de marquer les nœuds visités
  - A la fin on vérifie que tout le monde a été visité

Opérations CFG : analyse de flux de données

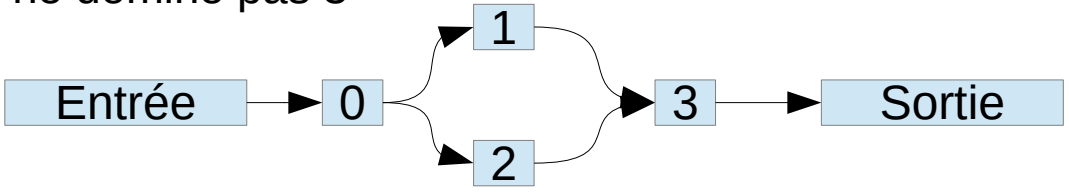
- Approche générale : algorithme de Kildall

```
Pour chaque nœud n :
    Etat[n] ← ensemble initial
Etat[entrée] ← ensemble d'entrée
Tant qu'il y a du changement :
    Pour chaque nœud n :
        EtatPred ← Union des états des prédécesseurs
        Etat[n] ← transfert(n, EtatPred)
```

- Usages : Definition, Liveness, Domination, ...
- Variantes :
  - Intersection des états prédécesseurs (au lieu d'Union)
  - Parcours en arrière

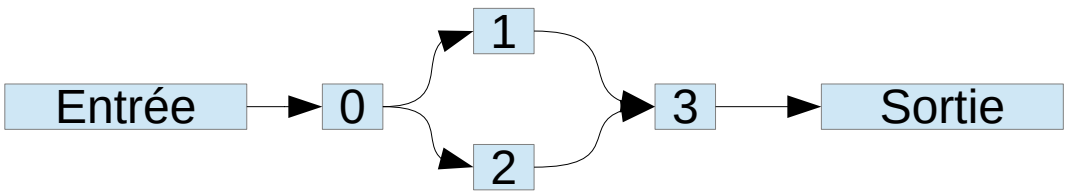
Opérations CFG : analyse de flux de données

- Application : calcul des dominateurs
- Définition : n1 domine n2 ssi tous les chemins depuis l'entrée vers n2 passent par n1.
- Exemples :
  - 0 domine 1, 2, et 3,
  - 1 ne domine pas 3
  - 2 ne domine pas 3



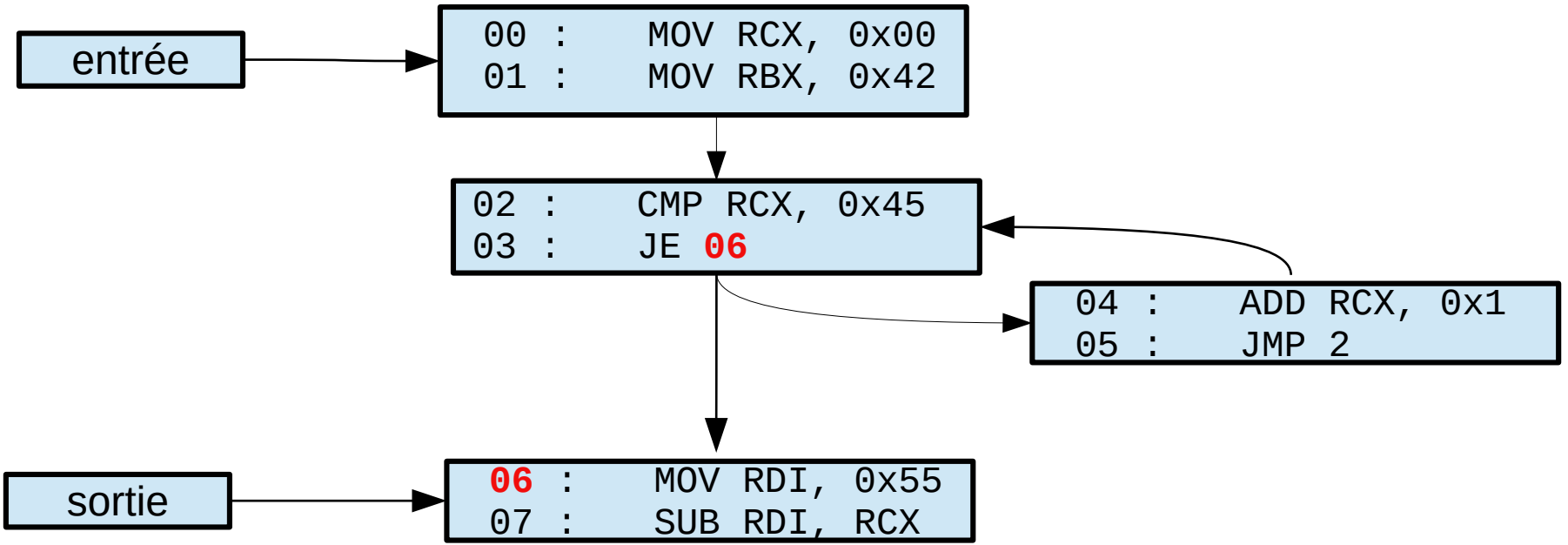
Opérations CFG : analyse de flux de données

- Algorithme de Kildall
  - Parcours en avant
  - Faire l'Intersection des états prédecesseurs
  - Transfert : Rajouter nœud courant dans le set
- Etat[entrée] = {Entrée}
- Etat[0] = {Entrée, 0}
- Etat[1] = {Entrée, 0, 1}
- Etat[2] = {Entrée, 0, 2}
- Etat[3] = {Entrée, 0, 3}



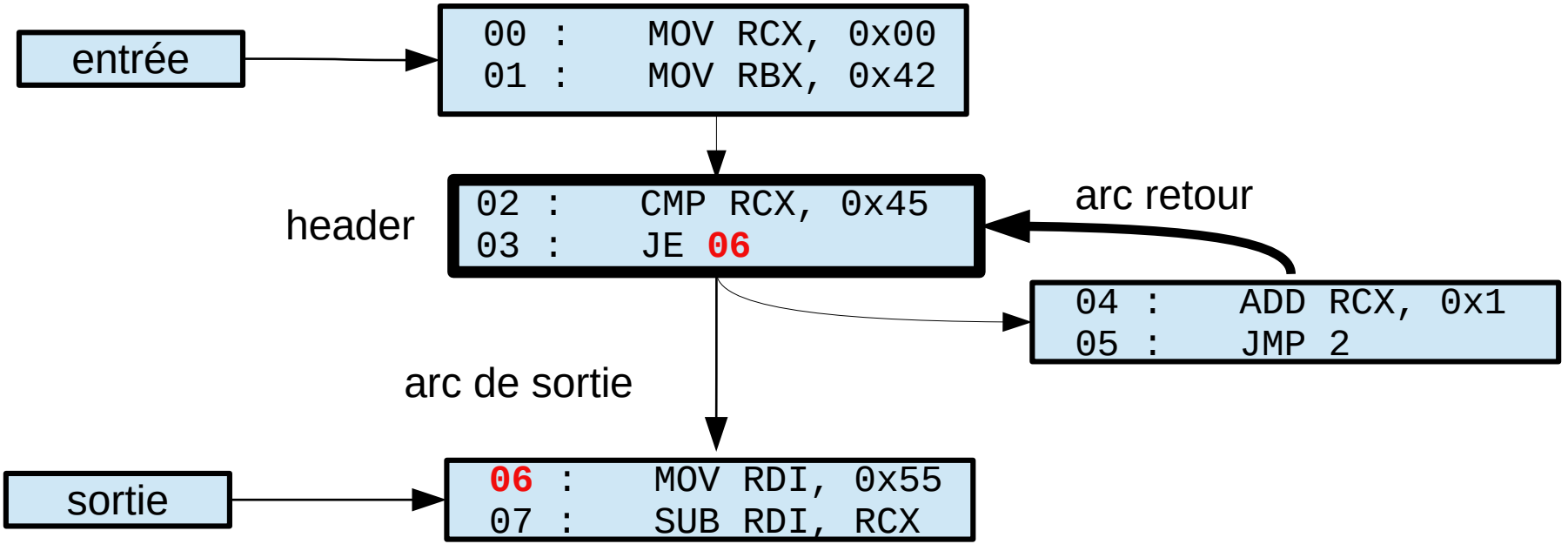
Opérations CFG : boucles

- Les boucles du programme se retrouvent dans le CFG



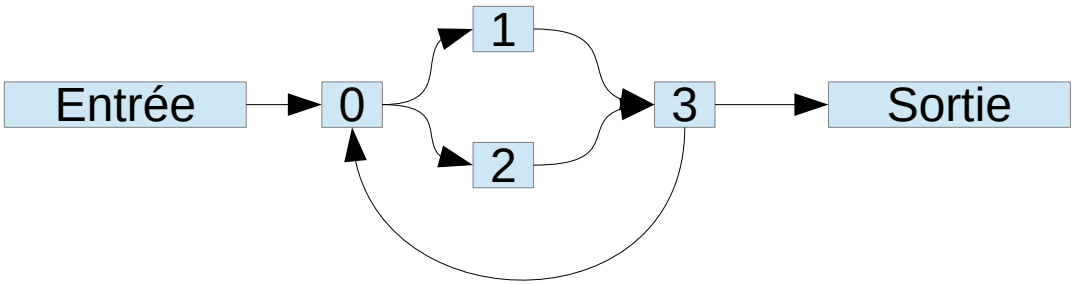
Opérations CFG : boucles

- Boucles correctement structurées : header unique
- Arc retour : la cible (header) domine la source



Opérations CFG : boucles

- Algorithme de Kildall
  - Parcours en arrière
  - Faire l'Union des états prédecesseurs
  - Transfert :
    - si predecesseur de back-edge, ajouter succeseur dans le set
    - si header, supprimer header du set
- Etat[Sortie] = {}
- Etat[3] = {0}
- Etat[2] = {0}
- Etat[1] = {0}
- Etat[0] = {}



- Permet de construire un arbre de la hiérarchie des boucles

(Re)construction de CFG

- Construction de CFG à partir d'une séquence d'instructions :
  - Initialement, chaque instruction est un bloc de base
  - Tant qu'on peut le faire, on fusionne des blocs adjacents
- A partir de quoi ?
  - Représentation intermédiaire (IR)
  - Binaire
- Reconstitution à partir de binaire : branchements indirects ?
  - Pointeurs de fonctions
  - Vtables, méthodes virtuelles en C++
  - Pointeurs de fonctions