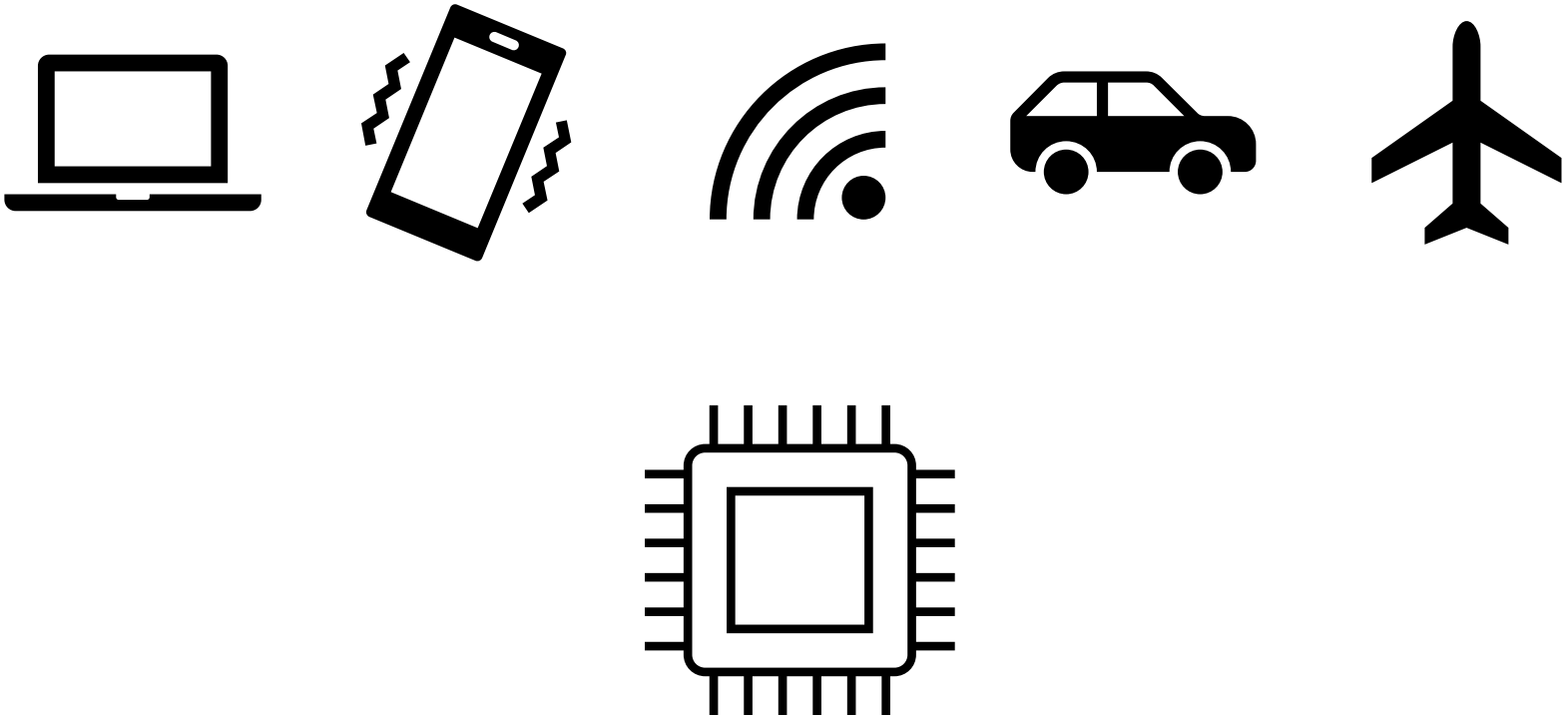
Garantir l'isolation microarchitecturale des processeurs

Directeurs de thèse:
Jean-Louis Lanet → Christophe Bidan
Jacques Fournier
Encadrant:
Ronan Lashermes

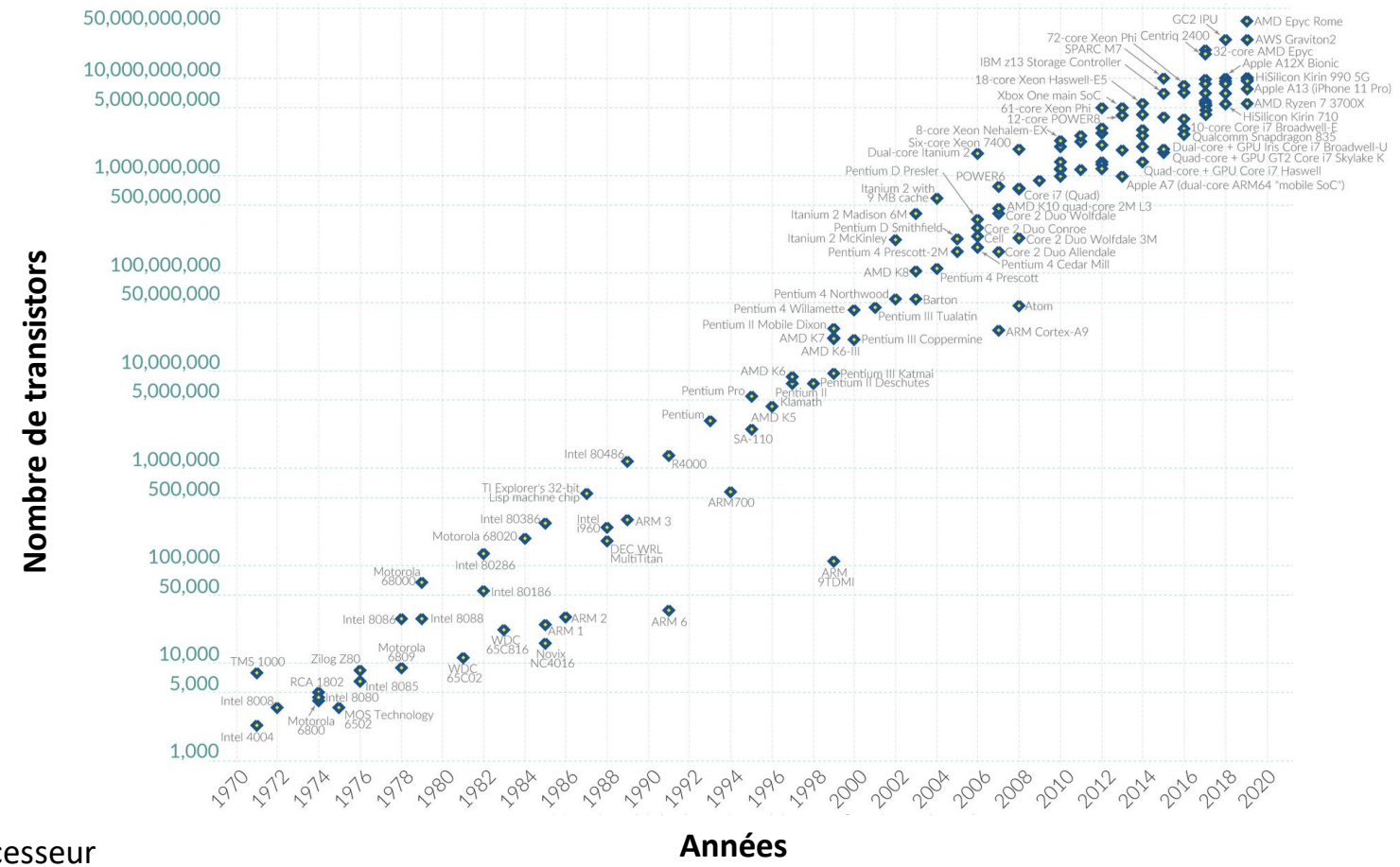
Soutenance de thèse de
Mathieu Escouteloup

Equipe CIDRE
Inria, CentraleSupélec, Univ Rennes 1, CNRS, IRISA

Une histoire de performances ...



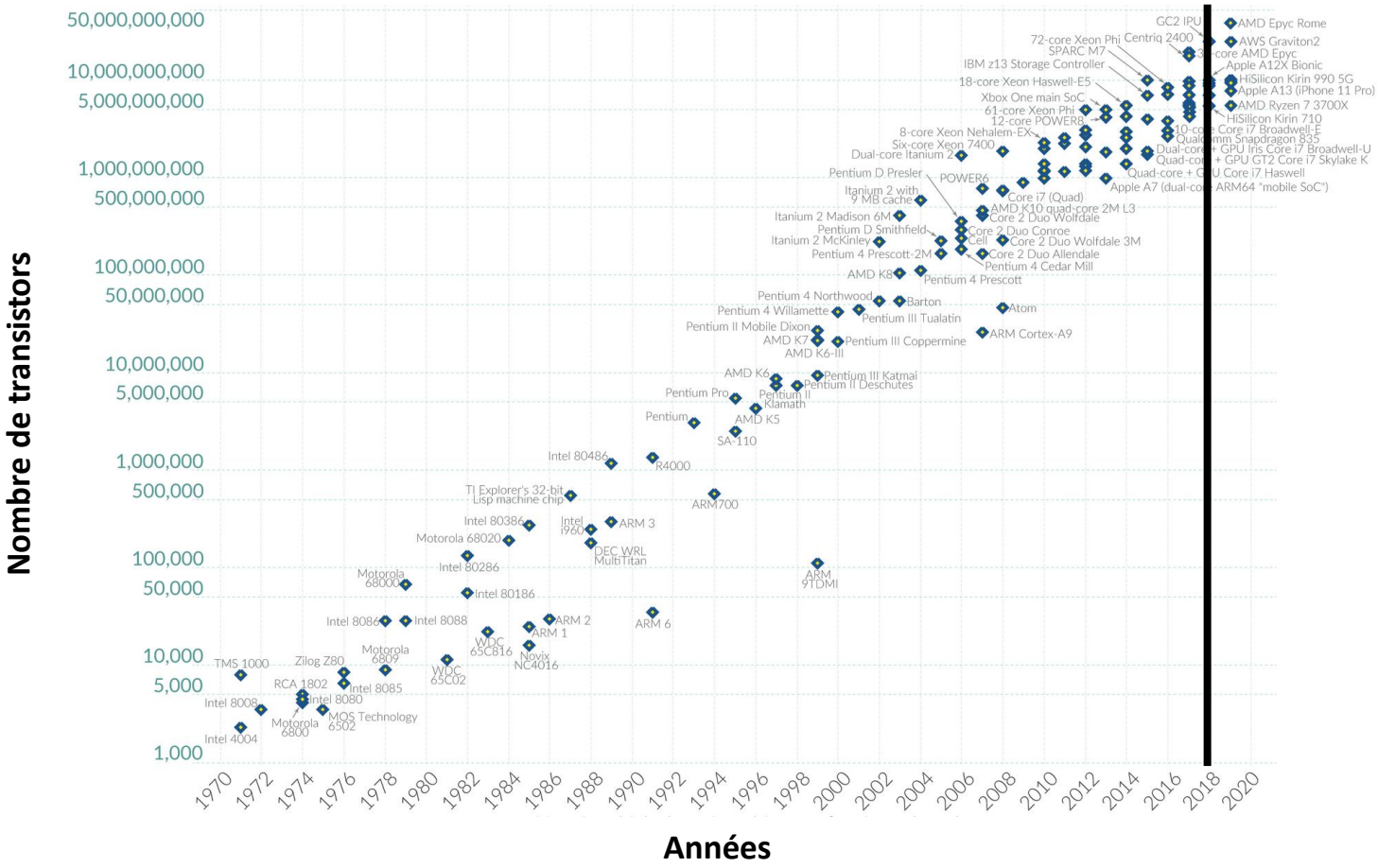
Une histoire de performances ...



- Processeur

… Et de vulnérabilités.

Découverte de Spectre & Meltdown.



Sommaire

Inria

1.

1. Fonctionnement des attaques
2. Solutions existantes et limites

3. Partage microarchitectural
4. Modification de l'ISA: les domes
5. Gestion des ressources partagées
6. Évaluation des implémentations

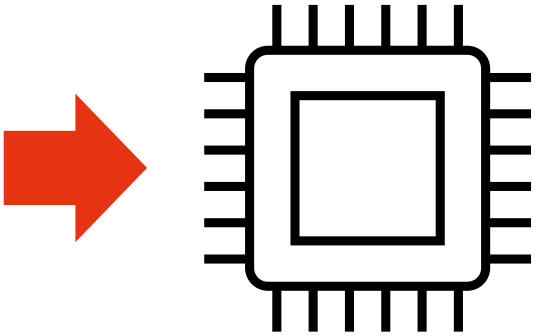
7. Synthèse et perspectives

Fonctionnement des processeurs : Principe général

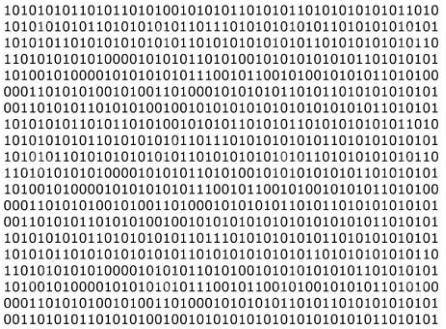
Logiciel

```
pgcd:
  addi sp, sp, -8
  lw a4, 0(sp)
  lw a5, 4(sp)

egal:
  bne a4, a5, superieur
  sw a4, 0(sp)
  addi sp, sp, 4
  jalr zero, ra, 0

superieur:
  blt a5, a4, inferieur
  sub a6, a5, a4
  sw a4, 0(sp)
  sw a6, 4(sp)
  addi sp, sp, 8
  j pgcd

inferieur:
  sub a6, a4, a5
  sw a5, 0(sp)
  sw a6, 4(sp)
  addi sp, sp, 8
  j pgcd
```

Processeur

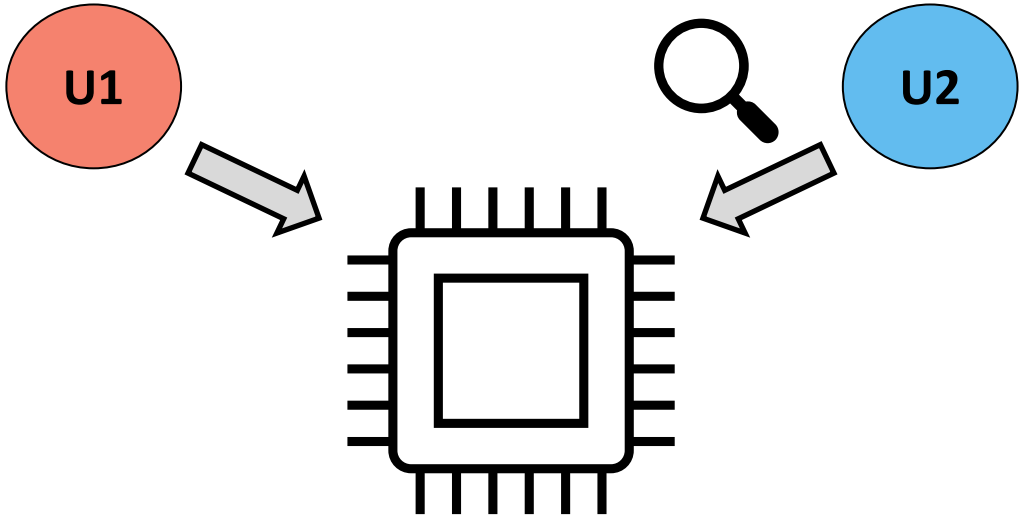


Données



A partir d'un programme, le processeur exécute successivement les opérations nécessaires.

Modèle d'attaquant



Utilisateur: une partie du logiciel ayant des données secrètes.

Modèle d'attaquant:

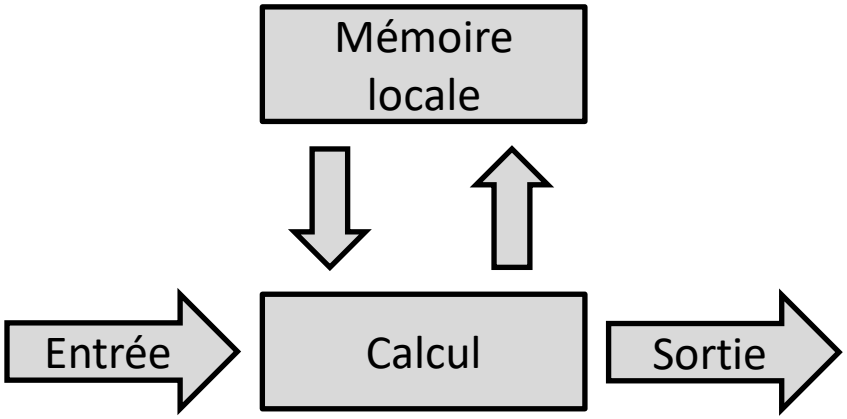
- Deux utilisateurs U1 et U2 s'exécutent sur un même processeur.
- U2 cherche à récupérer des informations de U1.
- U2 peut effectuer des mesures de temps.

Optimisation : sauvegarde d'informations

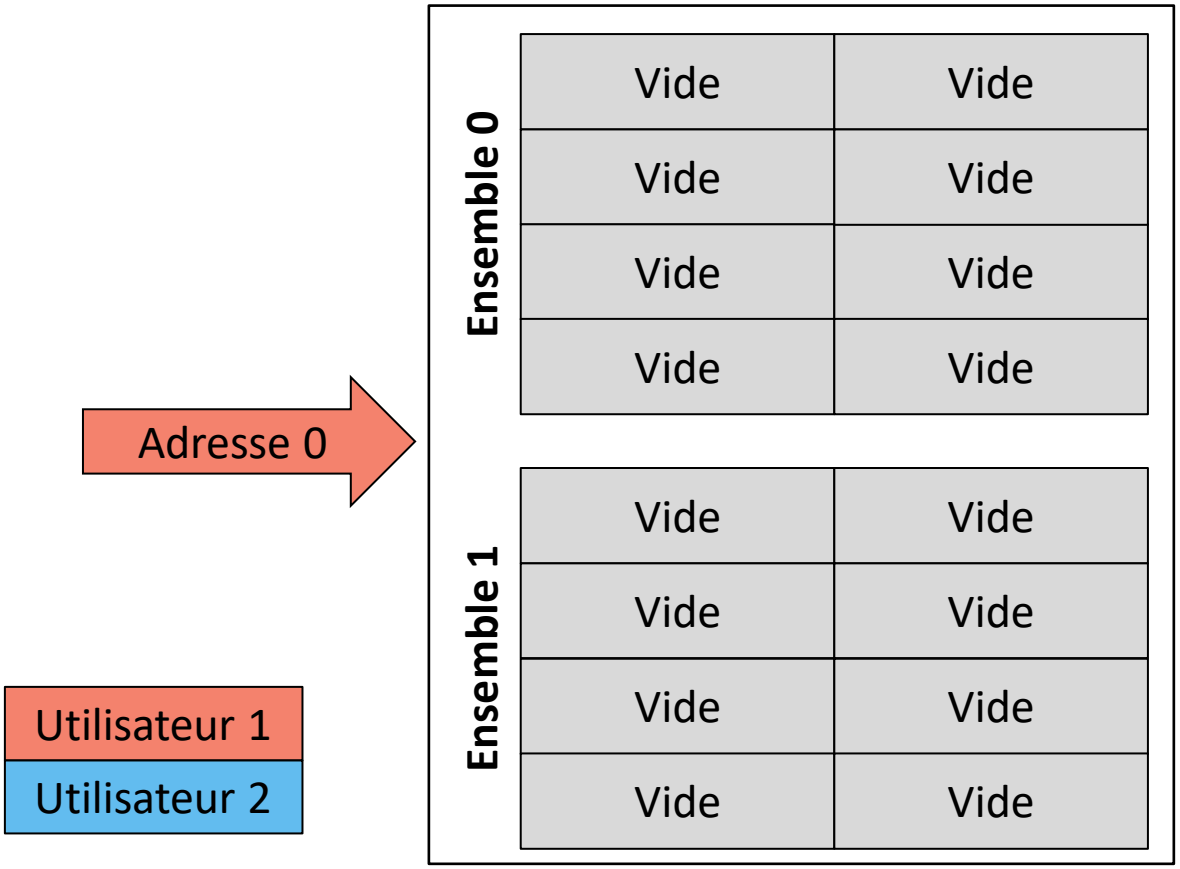
Caractéristiques des programmes:

- Instructions réutilisées (*e.g.* fonctions),
- Données réutilisées (*e.g.* variables),
- Flot non-sérialisé (*e.g.* condition).

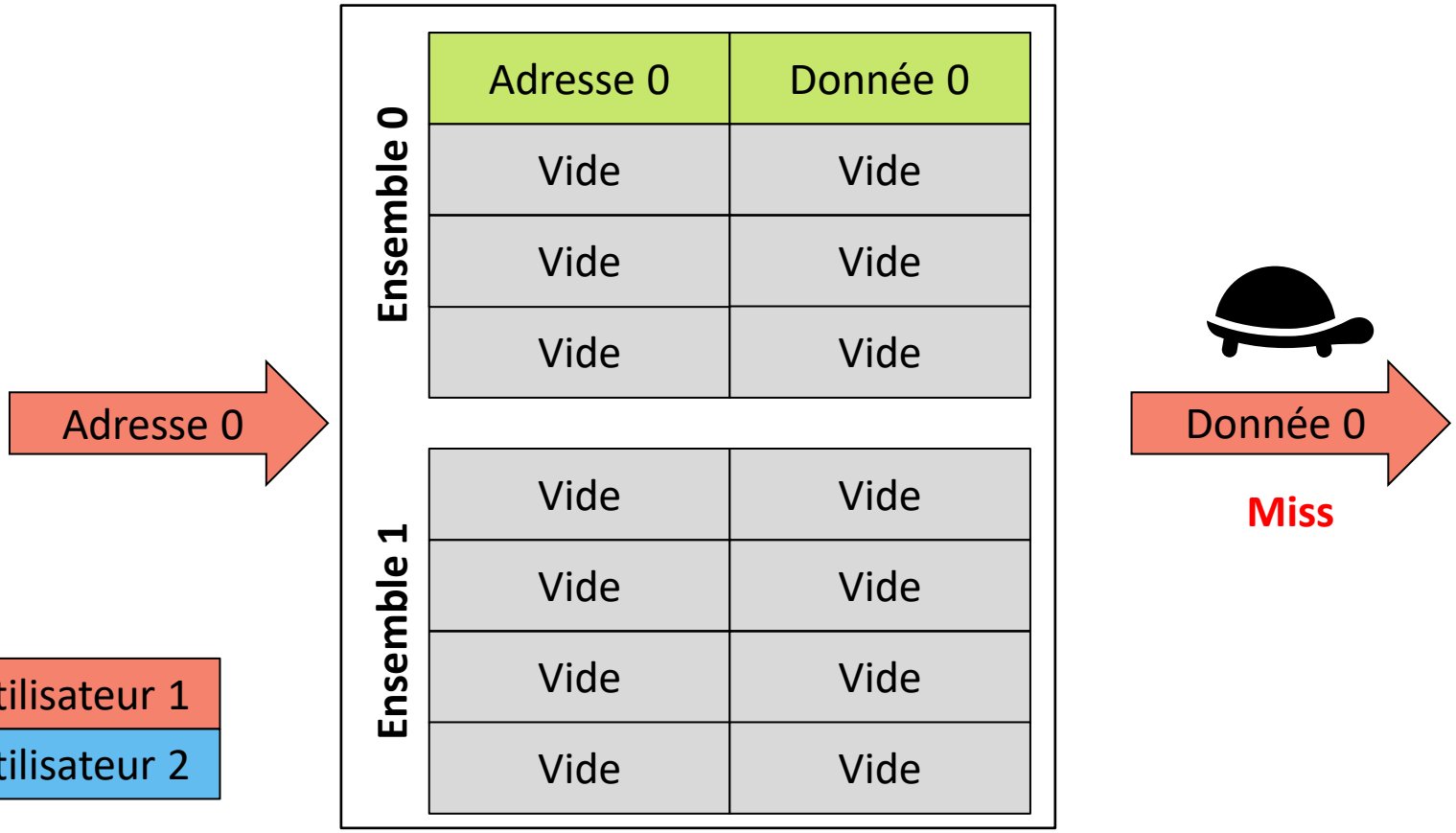
Organisation du processeur:



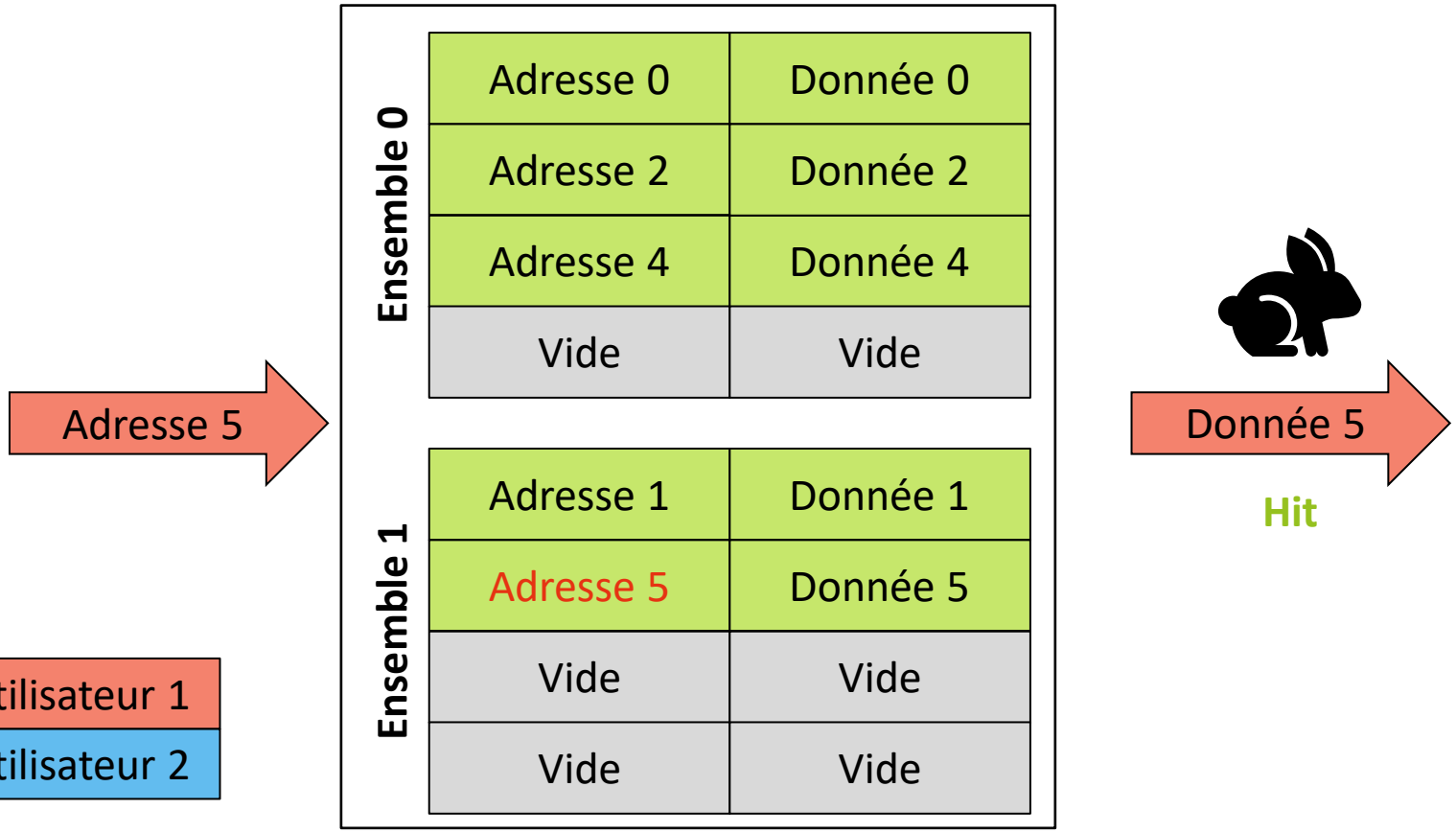
Attaque sur un cache : Accès classique (miss)



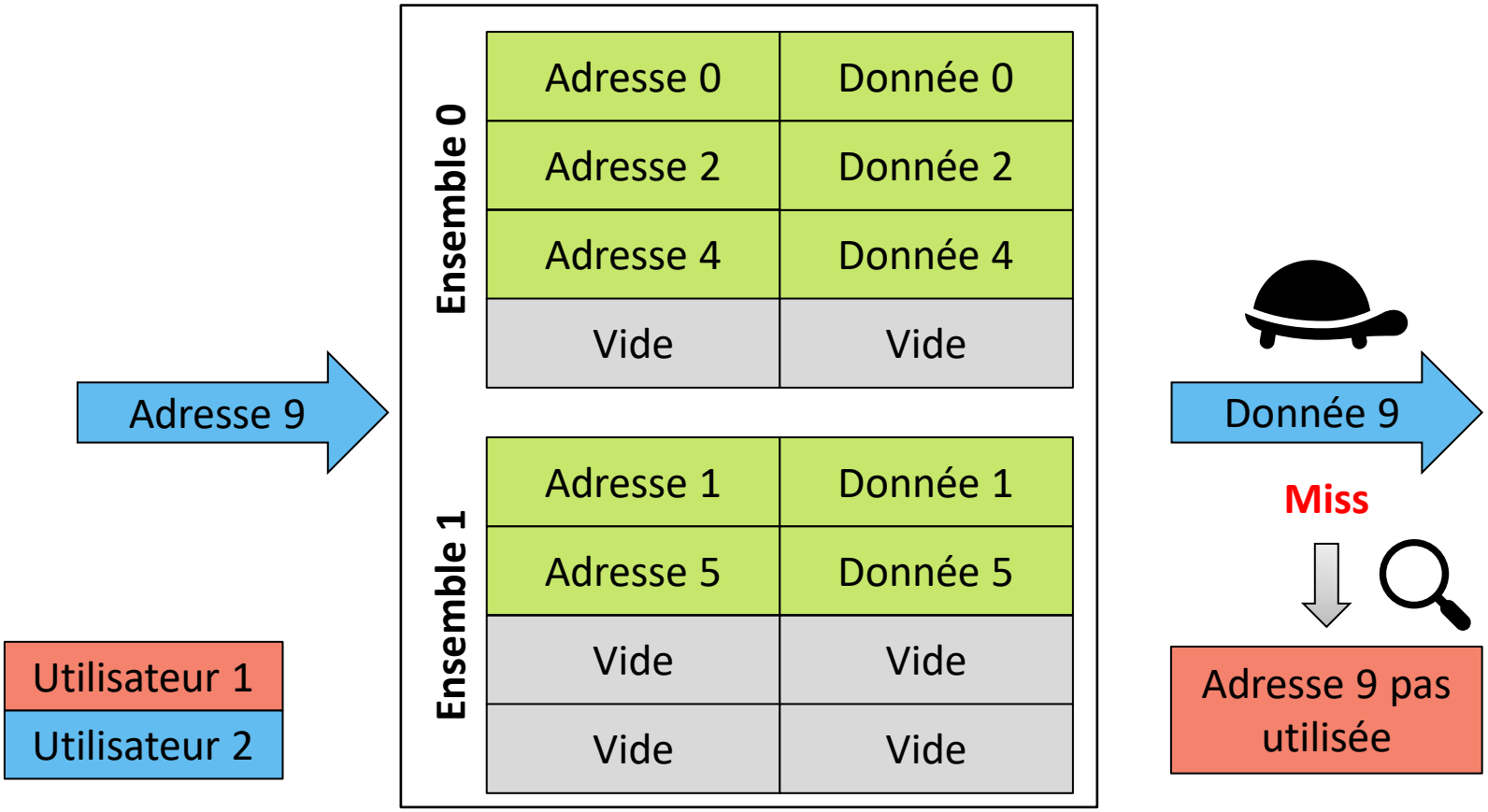
Attaque sur un cache : Accès classique (miss)



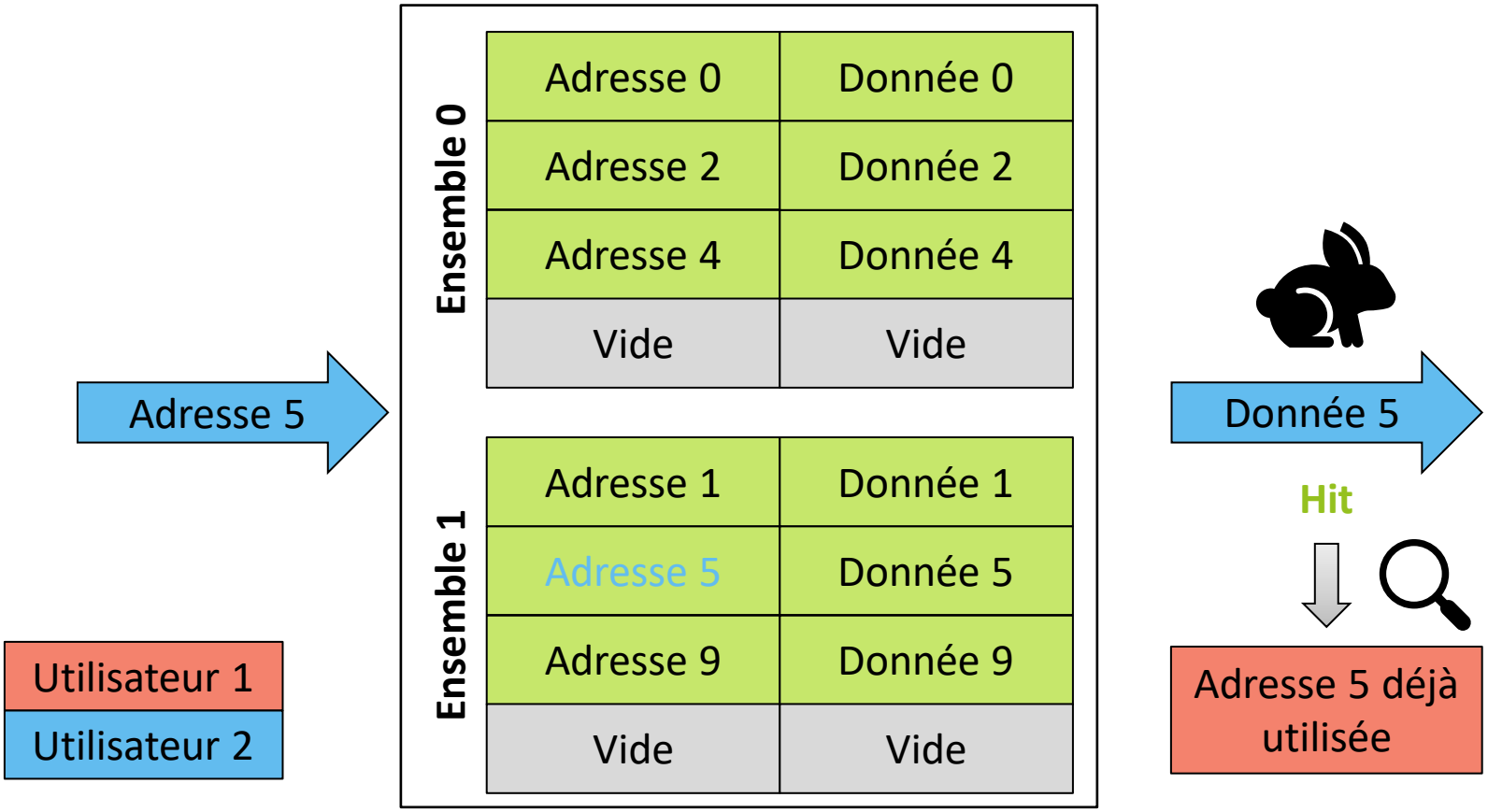
Attaque sur un cache: Accès classique (hit)



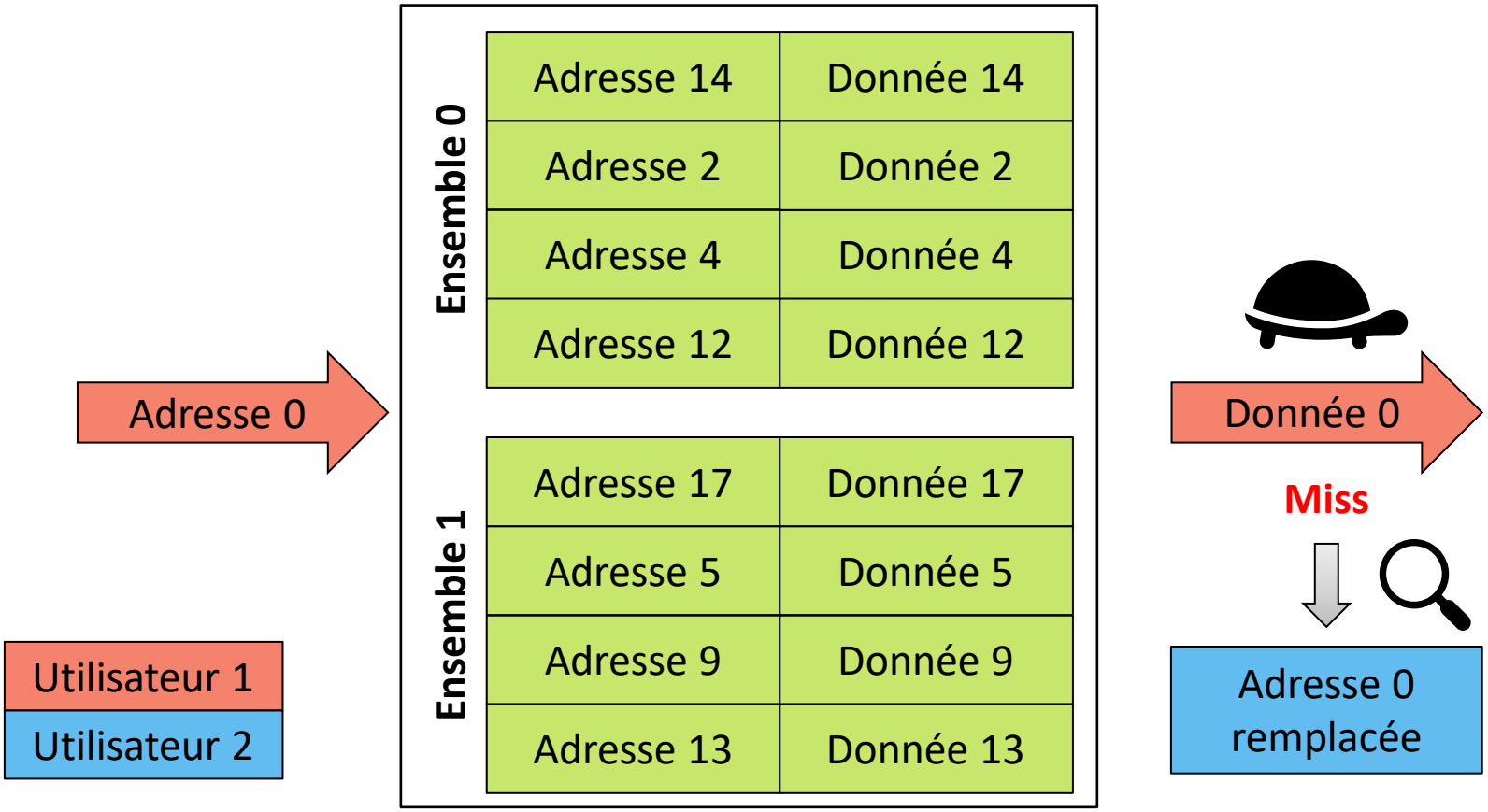
Attaque sur un cache : Détection d'un miss



Attaque sur un cache : Détection d'un hit



Attaque sur un cache : Détection d'un remplacement

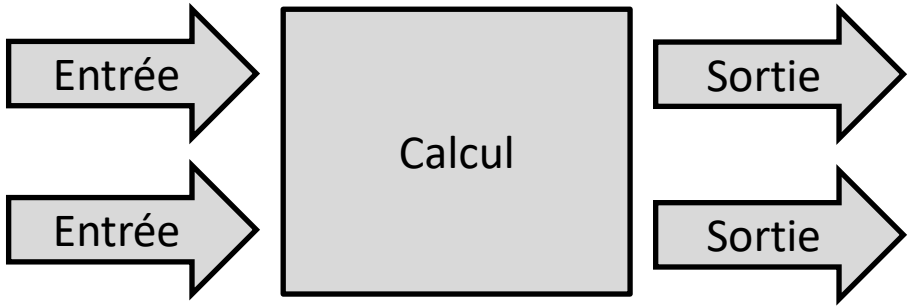


Optimisation : parallélisme

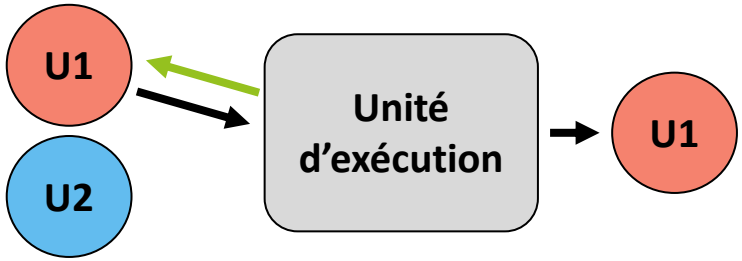
Caractéristiques des programmes :

- Indépendances entre opérations (*e.g.* opérandes différents).
- Indépendances entre programmes.

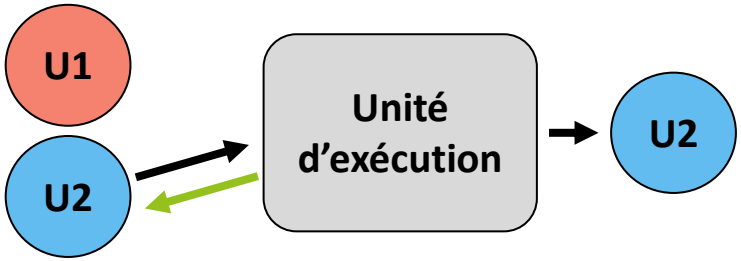
Organisation du processeur :



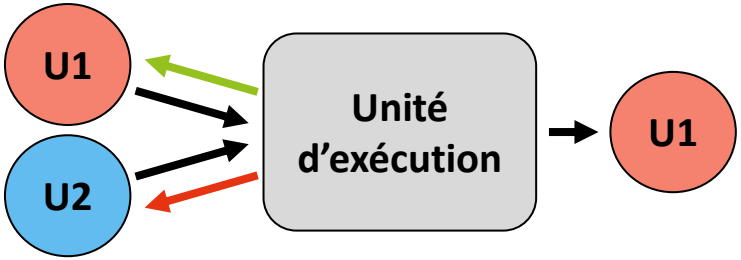
Attaque sur une unité d'exécution : Détection d'utilisation



U1 est servi.
Pas de demande de U2.



Pas de demande de U1.
U2 est servi.



U1 est servi.
U2 doit attendre.

→ U2 détecte l'utilisation par U1

2.

1. Fonctionnement des attaques
2. Solutions existantes et limites

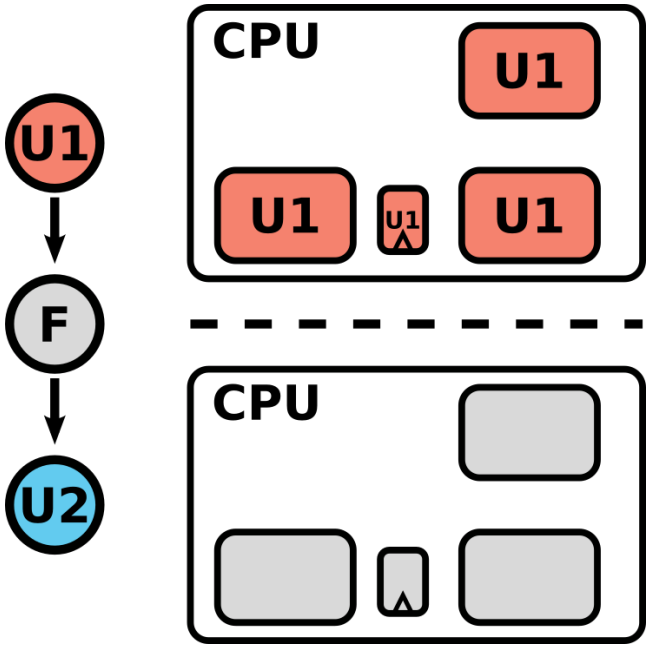
3. Partage microarchitectural
4. Modification de l'ISA: les domes
5. Gestion des ressources partagées
6. Évaluation des implémentations

7. Synthèse et perspectives

Gestion des ressources partagées

Effacement des traces (ou *flush)* : après utilisation, le système supprime les états persistants. [Bour2019, Wist2021]

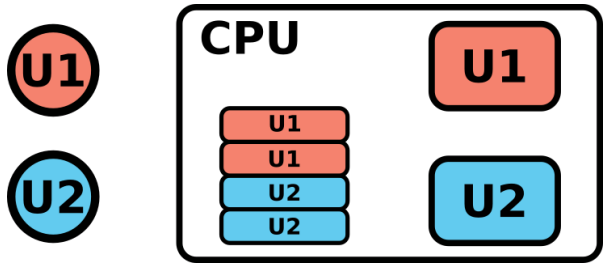
Différentes implémentations : écrasement des données ou instructions dédiées.



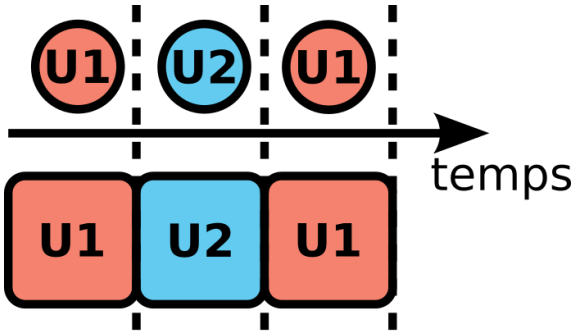
[Wist2021] Wistoff, Nils et al. "*Microarchitectural Timing Channels and their Prevention on an Open-Source 64-bit RISC-V Core*", DATE 2021
[Bour2019] Bourgeat, Thomas et al. "*MI6: Secure enclaves in a speculative out-of-order processor*", MICRO 2019

Gestion des ressources partagées

Partitionnement spatial : séparation des exécutions. [Ferr2017]



Partitionnement temporel : séparation du temps d'utilisation. [Ferr2017, Town2019]



[Ferr2017] Ferraiulo, Andrew et al. "*Full-processor timing channel protection with applications to secure hardware compartments*", 2017
[Town2019] Townley, Daniel et al. "*SMT-COP: Defeating side-channel attacks on execution units in SMT processors*", PACT 2019

Gestion du temps et des évènements

Modification des évènements microarchitecturaux :

- Supprimer les caches hits.
- Générer des caches misses. [Ojha2021]

Ajout d'indéterminisme :

- Randomiser le fonctionnement de mécanismes. [Wang2007]

Altération des mesures de temps :

- Réduire la précision des horloges. [GeJi2019]
- Restreindre les mesures de temps.

[Ojha2021] Ojha, Divya et Dwarkadas, Sandhya "*TimeCache: using time to eliminate cache side channels when sharing software*", ISCA 2021
[Wang2007] Wang, Zhenghong and Lee, Ruby B. "*New cache designs for thwarting software cache-based side channel attacks*", ISCA 2007
[GeJi2019] Ge, Jingquan et al. "*AdapTimer: Hardware/Software Collaborative Timer Resistant to Flush-Based Cache Attacks on ARM-FPGA Embedded SoC*", ICCD 2019

Limites des solutions existantes

Gestion des ressources partagées :

- Considération d'une ressource ou d'une implémentation.
- Efficace si toutes les ressources sont considérées. [GeQi2016]
- Peut altérer les performances (échelle macroscopique).

Gestion du temps et des évènements :

- Masque seulement certains problèmes.
- Efficace si toutes les informations sont masquées.
- Comment garantir que les informations sont inaccessibles ?

Comment généraliser l'isolation des exécutions ?
A quelle échelle l'appliquer ?

[GeQi2016] Ge, Qian et al. "*Your Processor Leaks Information-and There's Nothing You Can Do About It*", 2016

3.

1. Fonctionnement des attaques
2. Solutions existantes et limites

3. Partage microarchitectural
4. Modification de l'ISA: les domes
5. Gestion des ressources partagées
6. Évaluation des implémentations

7. Synthèse et perspectives

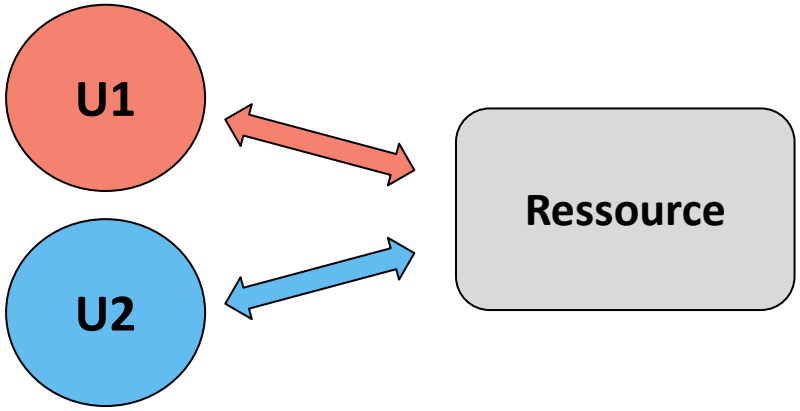
Un partage excessif d'informations

Ressource	Information perçue	Information déduite
Mémoire cache	Accès mémoire rapide (hit)	Utilisation d'une donnée
Mémoire cache	Accès mémoire lent (miss)	Non-utilisation d'une donnée
Mémoire cache	Accès mémoire lent (miss)	Remplacement d'une donnée
Unité d'exécution	Exécution ralentie	Type d'opération en cours
Contrôleur mémoire	Accès mémoire ralenti	Accès mémoire en cours
BTB	Saut rapide	Saut déjà effectué
BHT	Branchement rapide	Sens d'un branchement
…	…	…

BTB: *Branch Target Buffer*
BHT: *Branch History Table*

Partage microarchitectural

Définition : mécanisme matériel permettant à différentes exécutions logicielles d'accéder à une même ressource ou donnée.



Type d'informations partagées :

- **État persistant :** information mémorisée et qui reste.
- **État transitoire :** information temporaire non-mémorisée.

Intérêts :

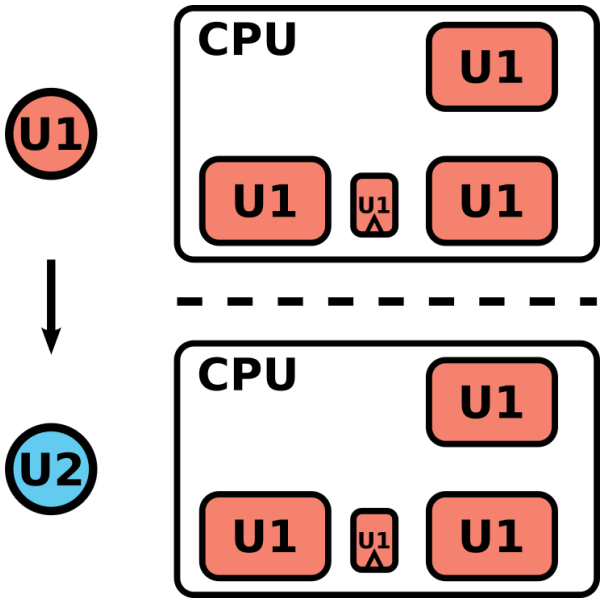
- Mise en commun d'informations.
- Réduction du nombre de ressources.

Partage temporel : état persistant

Partage temporel : possibilité pour plusieurs exécutions d'accéder à une même ressource à des moments différents.

Implémentations :

- Mémoires caches,
- Tables de prédiction,
- Machines d'états ...

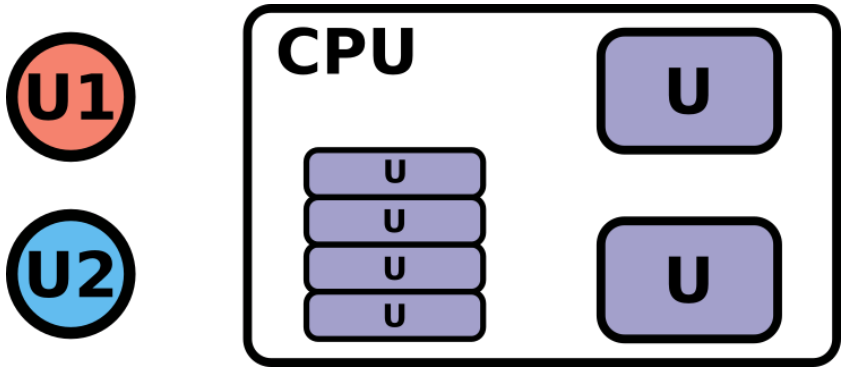


Partage spatial: état persistant

Partage spatial : possibilité pour plusieurs exécutions d'accéder à une même ressource simultanément.

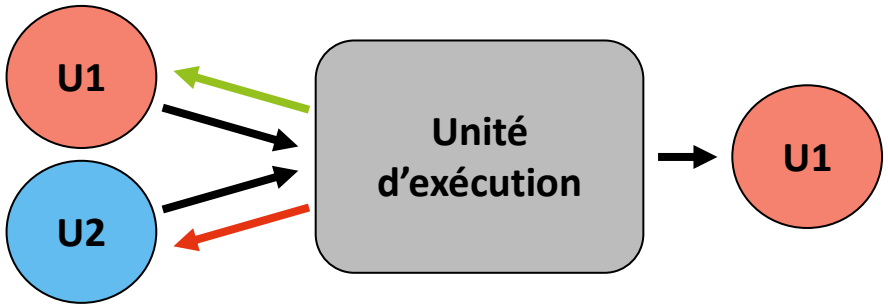
Implémentations :

- Mémoires caches,
- BTB,
- Machines d'états ...



Partage spatial: état transitoire

Contention dynamique : utilisation d'une ressource par un utilisateur menant à son indisponibilité pour les autres utilisateurs.



Implémentations :

- Unités d'exécution,
- Contrôleurs mémoires …

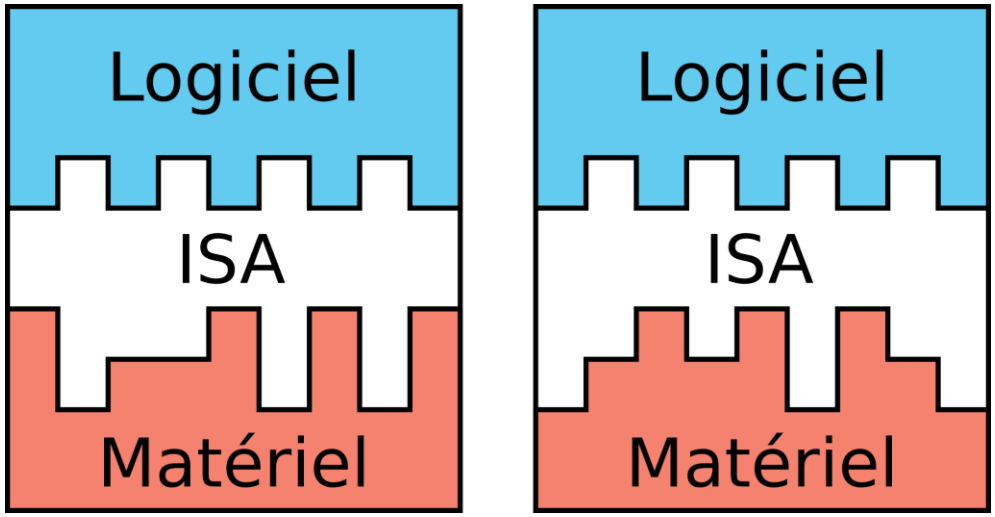
4.

1. Fonctionnement des attaques
2. Solutions existantes et limites

3. Partage microarchitectural
4. Modification de l'ISA: les domes
5. Gestion des ressources partagées
6. Évaluation des implémentations

7. Synthèse et perspectives

Rôle de l'ISA



1. **Quelle partie du système connaît les contraintes ?**
2. **Quelle partie du système peut assurer l'isolation ?**
3. **Comment leur permettre d'échanger des informations ?**

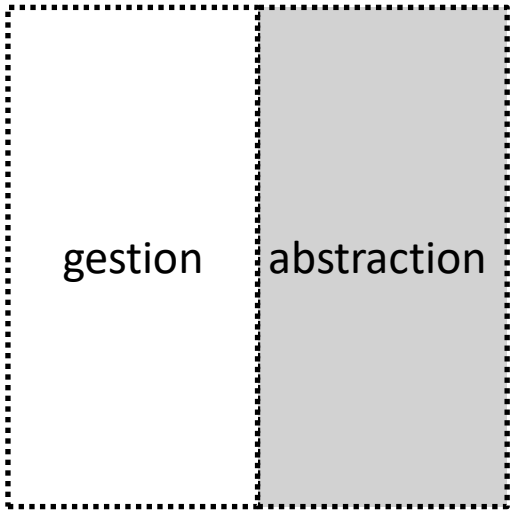
Retpoline: Turner, Paul "*Retpoline: A software construct for preventing branch-target injection*", 2018

ISA: *Instruction Set Architecture* ou jeu d'instructions

Stratégies d'abstraction

Gestion de la microarchitecture :

→ Instruction/ mécanisme dédié pour chaque ressource.
\+ Efficace en logiciel : contrôle fin du matériel.
± Surtout logiciel impacté.
− Logiciel dépendant du processeur ciblé.



Abstraction de la microarchitecture :

→ Envoi d'informations génériques du logiciel vers le matériel.
\+ Sécurité matérielle complètement déléguée au matériel.
\+ Logiciel indépendant du processeur ciblé.
− Logiciel et matériel impactés.

Formes des modifications

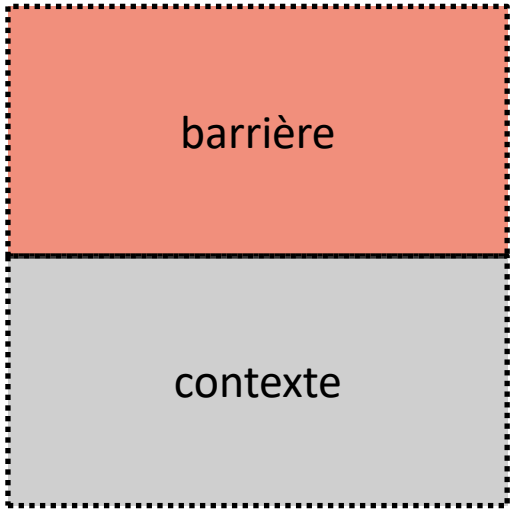
Barrière temporelle:

→ Indique une rupture dans le temps.

\+ Gestion du partage temporel.

\+ Approche fonctionnelle : une instruction déclenche une opération.

− Pas de gestion du partage spatial.



Contextualisation:

→ Associe un contexte d'exécution à chaque donnée ou instruction.

\+ Gestion du partage temporel.

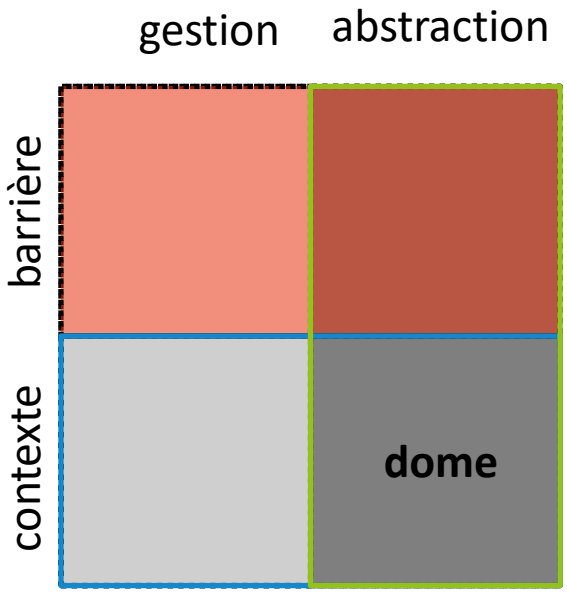
\+ Gestion du partage spatial.

− Approche contextuelle : une instruction influence d'autres instructions.

Les domes

Nos contraintes :

1. Considérer l'ensemble du problème d'isolation.
2. Conserver le rôle d'abstraction de l'ISA.
3. Intégrer une notion de domaine de sécurité standard.
4. Supporter tout type d'implémentation.



Description :

- Contexte d'exécution pour la sécurité.
- Gestion par le logiciel et application par le matériel.
- Extension de l'ISA RISC-V.

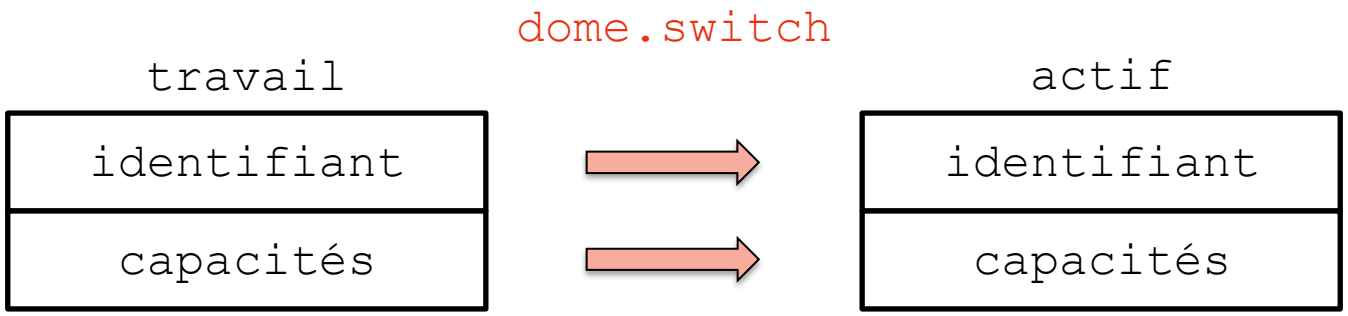
Les domes

Nouveaux registres dédiés (CSR) :

- **Identifiant** : un nombre unique pour chaque dome.
- **Capacité** : indication sur les besoins du domes.

Nouvelle instruction `dome.switch` :

- Indique un changement de dome actif.
- Déclenche les opérations au niveau du matériel.
- **Succès** = un nouveau dome peut être exécuté.



5.

1. Fonctionnement des attaques
2. Solutions existantes et limites

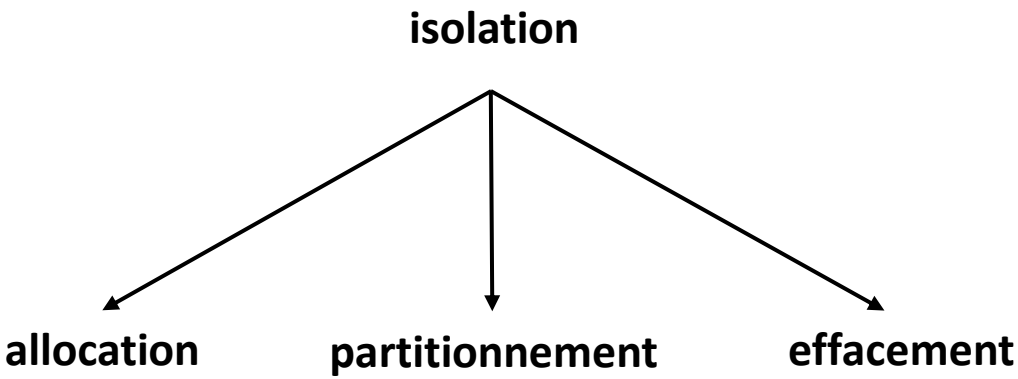
3. Partage microarchitectural
4. Modification de l'ISA: les domes
5. Gestion des ressources partagées
6. Évaluation des implémentations

7. Synthèse et perspectives

Ressources partagées et isolation

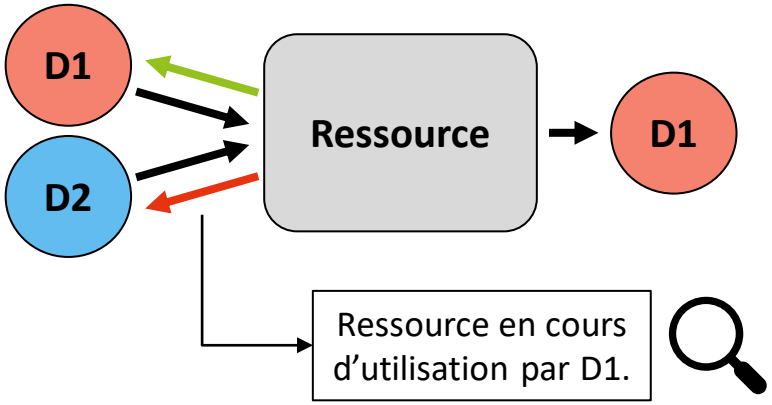
<u>Objectif :</u> ressource partagée isolée.
Les seules informations qu'un domaine de sécurité peut extraire d'une ressource partagée sont celles propres à ce domaine ou la disponibilité statique de la ressource.

Trois stratégies génériques et complémentaires :



Stratégie 1 : l'allocation statique

Disponibilité dynamique :

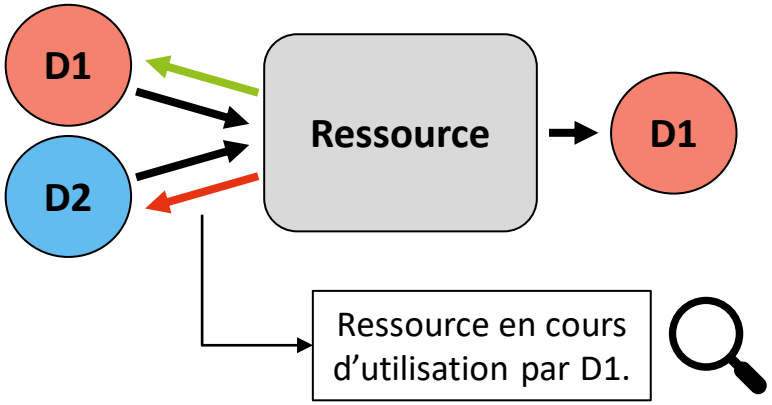


Stratégie 1 : l'allocation statique

Principe 1 : allocation statique.
Les ressources minimales nécessaires à un domaine de sécurité doivent être allouées durant la création de ce domaine et bloquées jusqu'à sa suppression.

Objectif : empêcher la contention (avec allocation exclusive).

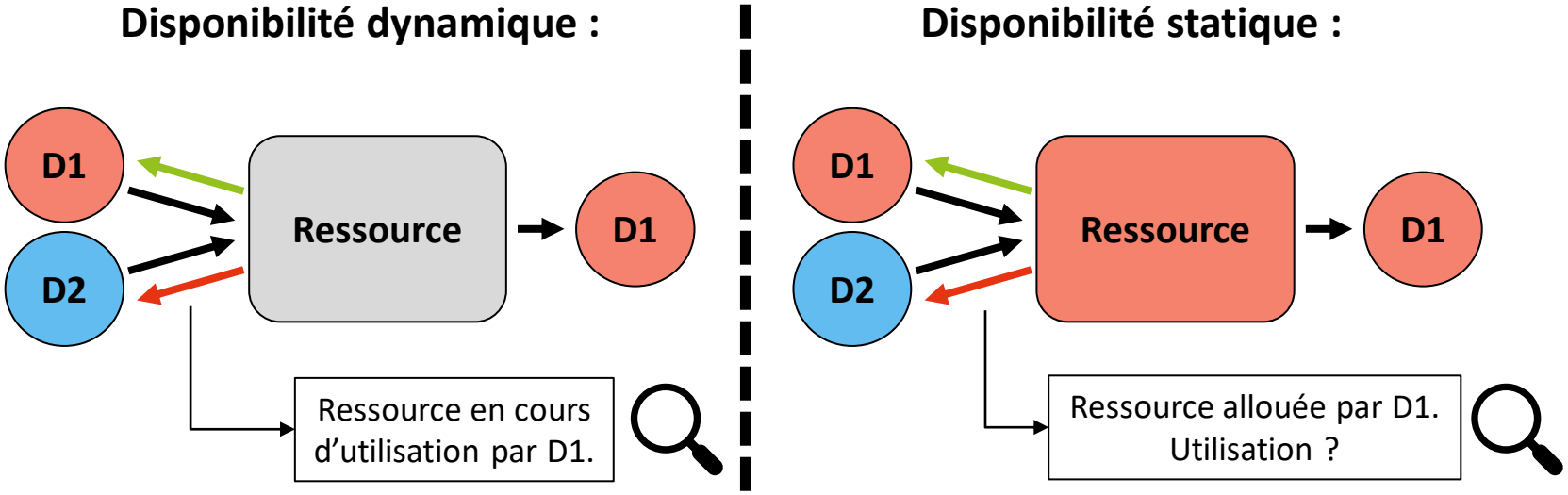
Disponibilité dynamique :



Stratégie 1 : l'allocation statique

Principe 1 : allocation statique.
Les ressources minimales nécessaires à un domaine de sécurité doivent être allouées durant la création de ce domaine et bloquées jusqu'à sa suppression.

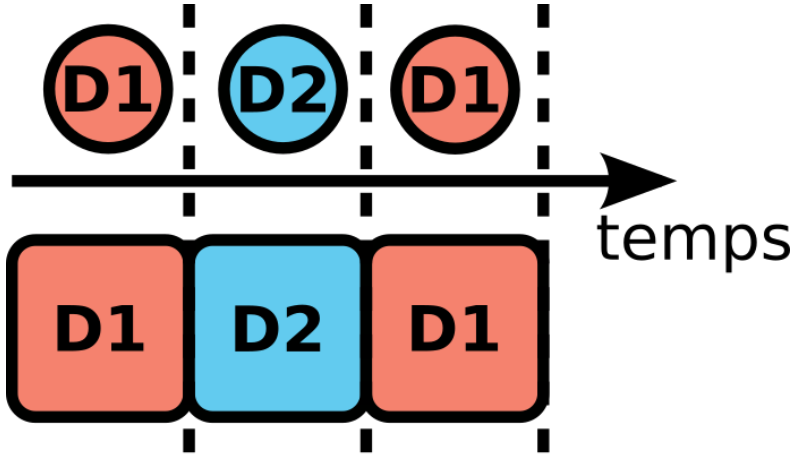
Objectif : empêcher la contention (avec allocation exclusive).



Stratégie 2 : le partitionnement

Principe 2 : séparation de la disponibilité.
Une ressource partagée doit s'assurer qu'à n'importe quel moment, sa disponibilité pour un domaine de sécurité est indépendante des autres domaines de sécurité.

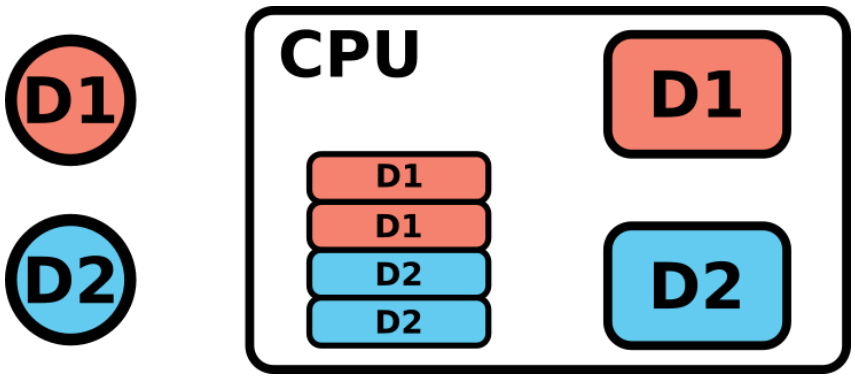
Objectif : empêcher la contention (sans allocation exclusive).



Stratégie 2 : le partitionnement

Principe 3 : partitionnement spatial.
Une ressource gérant simultanément des requêtes de plusieurs domaines de sécurité doit être capable de cloisonner chacun d'eux dans son propre compartiment.

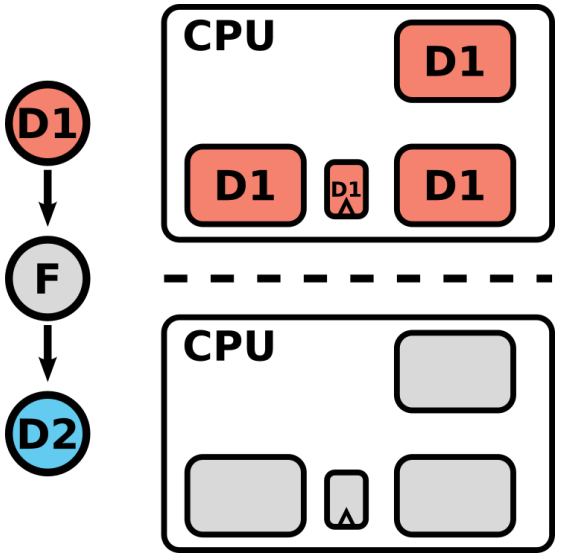
Objectif : séparer les états persistants.



Stratégie 3 : l'effacement

Principe 4 : suppression des traces.
Quand un domaine de sécurité termine son exécution, toutes ses ressources doivent être relâchées uniquement lorsque les traces persistantes ont été effacées.

Objectif : séparer les états persistants.



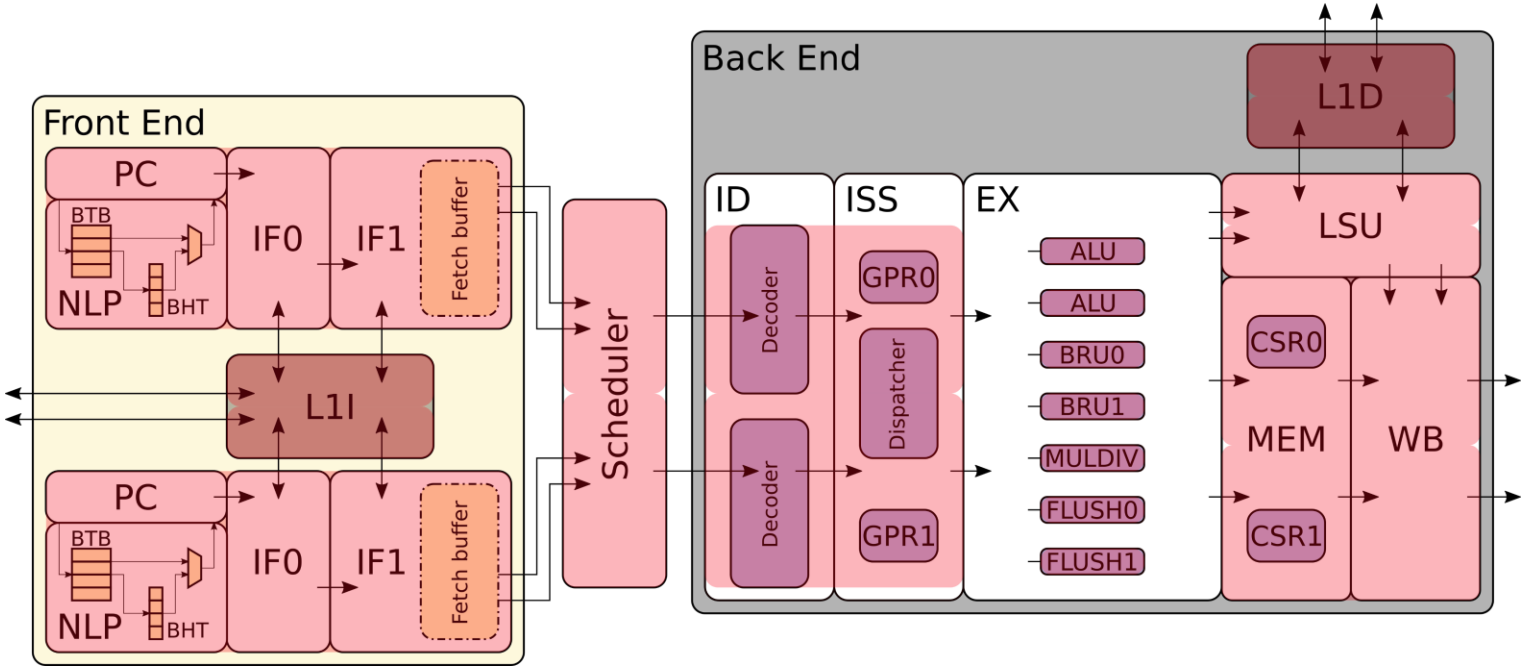
Scénario d'utilisation : logiciel

Un premier dome s'exécute:
- Il utilise des ressources,
- Prépare le prochain dome,
- Puis le lance.

Un second dome s'exécute,
- Le processeur assure l'isolation.

```
1.  # ANCIEN DOME
2.  old-app:
3.    ...

10. switch-code:
11.   csrw nextid,a0  # config
12.   dome.switch a0  # switch

13. # NOUVEAU DOME
14. new-app:
15.   ...
```

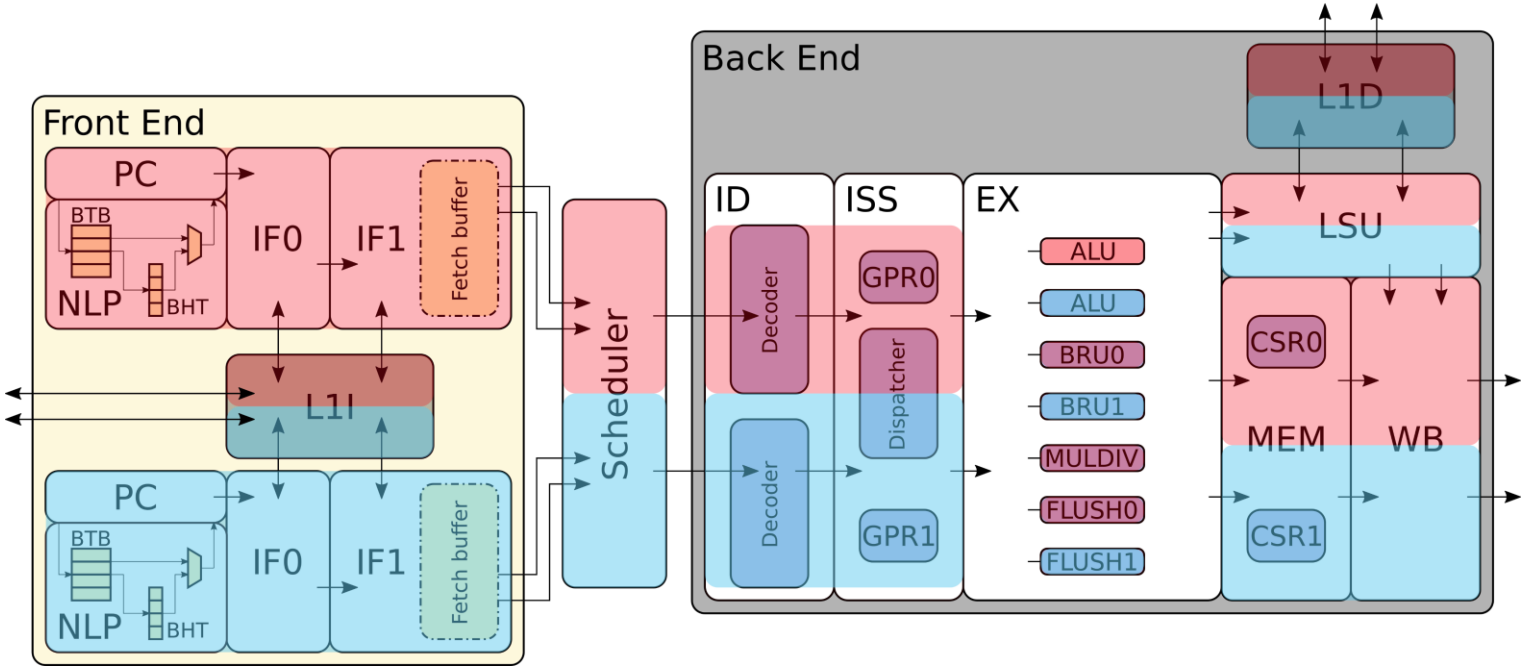
Scénario d'utilisation : matériel



Processeur Salers modifié

Principes aussi appliqués sur le coeur Aubrac: pipeline cinq étages avec partage temporel seulement

Scénario d'utilisation : matériel



Processeur Salers modifié

Principes aussi appliqués sur le coeur Aubrac: pipeline cinq étages avec partage temporel seulement

6.

1. Fonctionnement des attaques
2. Solutions existantes et limites

3. Partage microarchitectural
4. Modification de l'ISA: les domes
5. Gestion des ressources partagées
6. Évaluation des implémentations

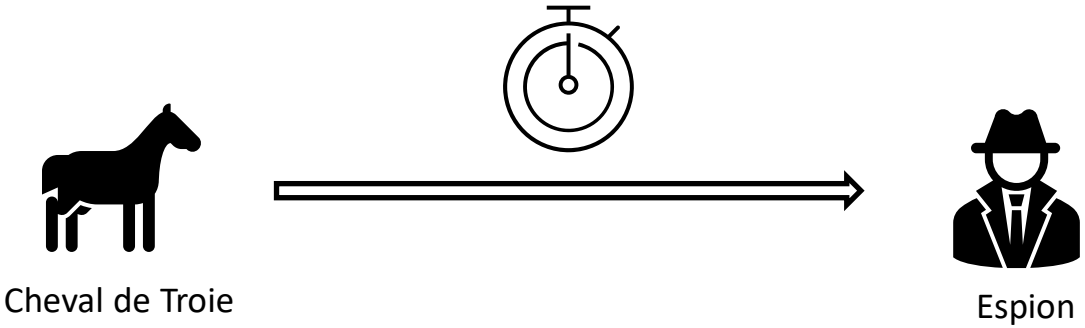
7. Synthèse et perspectives

Un suite de tests pour la sécurité : Timesecbench

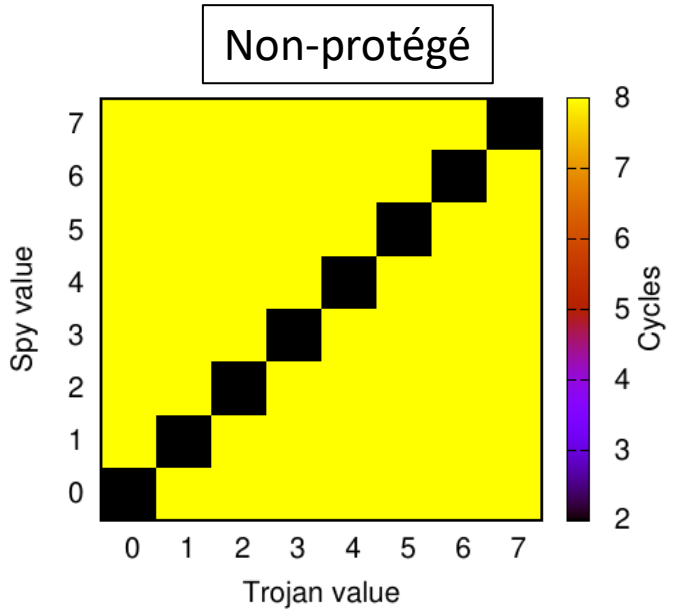
Objectifs :

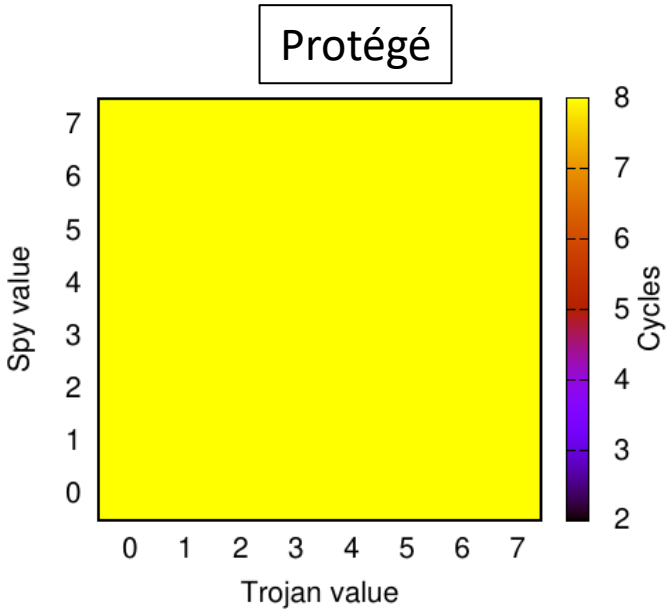
- Détecter les fuites temporelles dues aux ressources partagées.
- Disposer d'un outil de test simple d’utilisation dès la conception.
- Être indépendant de l'implémentation.
- Être indépendant du jeu d'instructions.

Scénario d'attaque retenu : canal caché



Test : L1D/ temporel





Caractéristiques du cache :

- 8 ensembles,
- *Hit* = 2 cycles,
- *Miss* = 8 cycles.

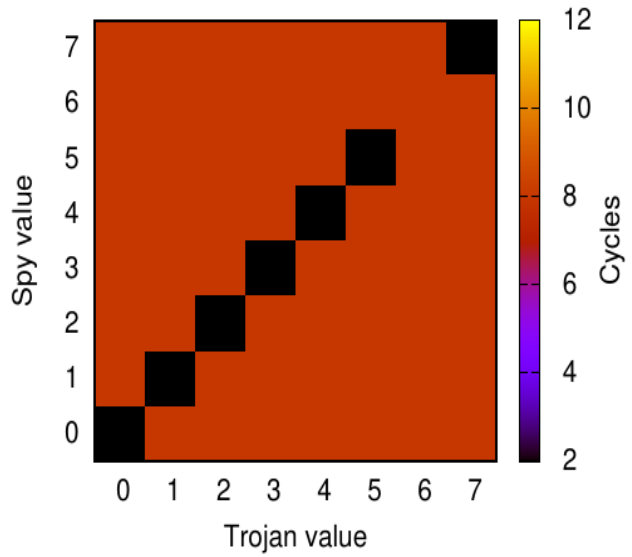
Cœur ciblé : Aubrac (pas de SMT)

Lecture du diagramme :

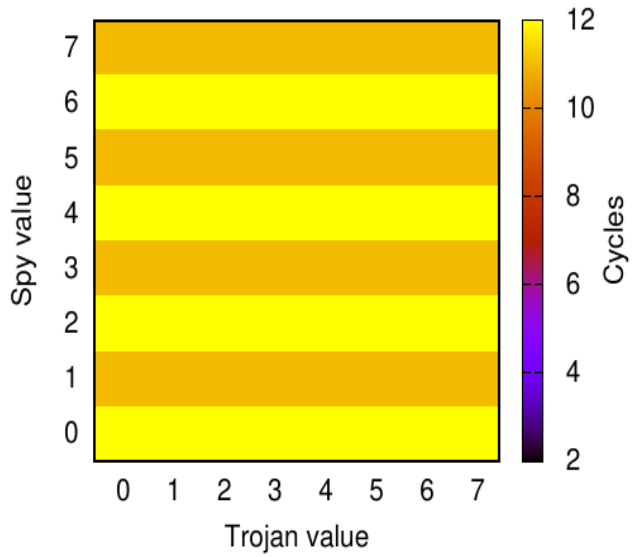
- Axe X : ensemble utilisé par le cheval de Troie.
- Axe Y : ensemble utilisé par l'espion.
- Couleur : nombre de cycles.

Test : L1D/ spatial

Non-protégé



Protégé



Caractéristiques du cache :

- 8 ensembles,
- *Hit* = 2 cycles,
- *Miss* = 8 cycles (sans dome), 11/12 cycles (avec dome).

Cœur ciblé : Salers (avec SMT)

Lecture du diagramme :

- Axe X : ensemble utilisé par le cheval de Troie.
- Axe Y : ensemble utilisé par l'espion.
- Couleur : nombre de cycles.

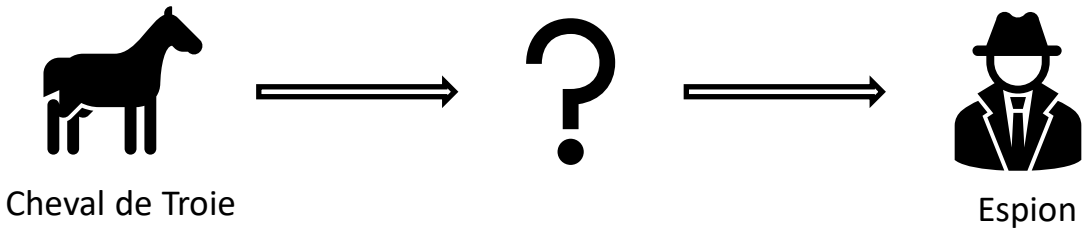
Résultats de Timesecbench

Autres ressources partagées testées :

- BTB (partage temporel),
- BHT (partage temporel),
- Mémoire cache L1I (partage temporel),
- Unité d'exécution (contention dynamique).

Bilan:

- Différents types de fuites par variations temporelles ont été détectés.
- Les principes sont efficaces sur toutes les ressources testées.



Impact sur les performances

Cache	1 kB	4 kB	1 kB	4 kB
Salers	1 thread actif		2 threads actifs	
	1,05	0,91	0,56	0,51
Dome	1,07	0,88	0,76	0,62
NLP	1,03	0,88	0,55	0,51
NLP Dome	1,05	0,86	0,75	0,61

Moyenne géométrique obtenue pour Salers avec Embench (version SMT).
Référence: cœur Aubrac / 1kB / sans dome / sans NLP

Fréquence d'horloge : pas d'impact significatif.
Suite de tests Embench :

- Pas d'impact sur Aubrac (partage temporel uniquement),
- Intérêt du SMT conservé pour Salers,
- Pertes en grande partie dues aux caches L1 scindés.

Surcoût matériel

Cache	1 kB		4 kB	
Aubrac - without SMT				
	9370 LUT	réf.	24590 LUT	×2,62
	4408 FF	réf.	8378 FF	×1,90
Dome	9651 LUT	×1,03	24530 LUT	×2,62
	4715 FF	×1,07	8703 FF	×1,97
NLP	12168 LUT	×1,30	27410 LUT	×2,93
	5267 FF	×1,19	9237 FF	×2,10
NLP Dome	12146 LUT	×1,30	26984 LUT	×2,88
	5575 FF	×1,26	9562 FF	×2,17

Cache	1 kB		4 kB	
Salers - with SMT				
	21000 LUT	réf.	46943 LUT	×2,24
	8270 FF	réf.	13676 FF	×1,65
Dome	22919 LUT	×1,09	44852 LUT	×2,14
	10889 FF	×1,32	16498 FF	×1,99
NLP	26731 LUT	×1,27	54278 LUT	×2,58
	10011 FF	×1,21	15566 FF	×1,88
NLP Dome	28167 LUT	×1,34	55977 LUT	×2,67
	12599 FF	×1,52	18188 FF	×2,20

Utilisation de ressources matérielles pour chaque version des cœurs.

7.

1. Fonctionnement des attaques
2. Solutions existantes et limites

3. Partage microarchitectural
4. Modification de l'ISA: les domes
5. Gestion des ressources partagées
6. Évaluation des implémentations

7. Synthèse et perspectives

Synthèse

Généralisation de la problématique d'isolation microarchitecturale :

- Classification des différentes attaques logicielles.
- Mise en place d'une approche globale et générique.
- Trois stratégies pour toute ressource partagée.

Principes	Où	Effet
1. Allocation statique	Unités d'exécution	Empêche la contention.
2. Séparation de la disponibilité	Contrôleur mémoire	Empêche la contention.
3. Partitionnement spatial	Mémoire cache	Empêche les fuites dues au partage spatial.
4. Effacement des traces	Mémoire cache, pipeline, BHT, BTB	Empêche les fuites dues au partage temporel.

Synthèse

Une extension de l'ISA RISC-V pour l'isolation microarchitecturale :

- Notion standard de contexte de sécurité.
- Préserve le rôle d'abstraction de l'ISA.

Un nouvel outil d'évaluation des processeurs :

- Permet la détection de fuites par variations temporelles dues aux ressources partagées.
- Capable d'éprouver plusieurs sources de fuites.

Deux implémentations de référence (Aubrac et Salers) :

- Contraintes d'isolation intégrées dès les fondements.
- Gestion du partage spatial et temporel.
- Aucune rustine logicielle ou matérielle nécessaire.

Travaux futurs et perspectives

Extension de l'isolation:

- Implémenter un processeur complexe (exécution dans le désordre).
- Prolonger l'isolation à l'extérieur du processeur.
- Intégrer des frontières pour la spéculation.

Adaptation du logiciel :

- Adapter un système d'exploitation.
- Adapter les compilateurs.

Nouveaux outils de détection des fuites :

- Cibler de nouvelles ressources partagées avec Timesecbench.
- Étude des techniques de formalisation.

Publications

- Mathieu Escouteloup et Jacques Fournier et Jean-Louis Lanet et Ronan Lashermes "*Recommendations for a radically secure ISA*", CARRV 2020
- Thomas Trouchkine, Sébanjila Kevin Bukasa, Mathieu Escouteloup, Ronan Lashermes et Guillaume Bouffard "*Electromagnetic fault injection against a complex CPU, toward new microarchitectural fault models*", JCEN 2021
- Mathieu Escouteloup, Ronan Lashermes, Jacques Fournier et Jean-Louis Lanet "*Under the dome: preventing timing hardware timing information leakage*", CARDIS 2021
- Publication dans un journal en préparation "*From static to dynamic privileges: redesign the ISA for modern constraints*"

Communications

- Mathieu Escouteloup et Jacques Fournier et Jean-Louis Lanet et Ronan Lashermes "*Microarchitecture security*", PHISIC 2019
- Mathieu Escouteloup, Ronan Lashermes, Jacques Fournier et Jean-Louis Lanet "*Preventing timing information leakages from the microarchitecture*", 2nd RISC-V Week 2021

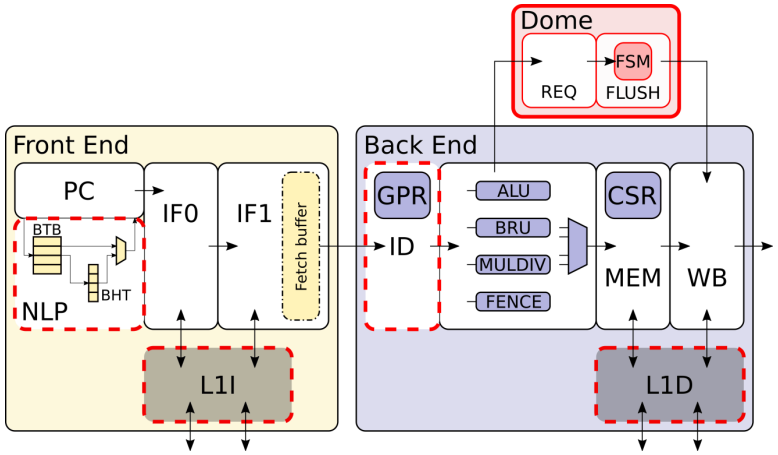
Projets

- Coeurs modifiés et évaluation: https://gitlab.inria.fr/mescoute/hsc-eval
- Timesecbench: https://gitlab.inria.fr/rlasherm/timesecbench

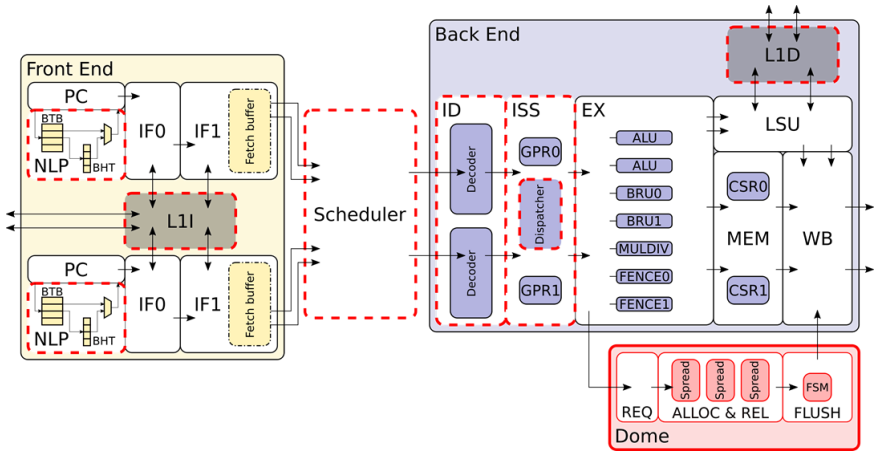
Bibliographie

- **[Wist2021]** Wistoff, Nils et al. “Microarchitectural Timing Channels and their Prevention on an Open-Source 64-bit RISC-V Core”, DATE 2021
- **[Bour2019]** Bourgeat, Thomas et al. “MI6: Secure enclaves in a speculative out-of-order processor”, MICRO 2019
- **[Ferr2017]** Ferraiulo, Andrew et al. “Full-processor timing channel protection with applications to secure hardware compartments”, 2017
- **[Town2019]** Townley, Daniel et al. “SMT-COP: Defeating side-channel attacks on execution units in SMT processors”, PACT 2019
- **[Ojha2021]** Ojha, Divya et Dwarkadas, Sandhya “TimeCache: using time to eliminate cache side channels when sharing software”, ISCA 2021
- **[Wang2007]** Wang, Zhenghong and Lee, Ruby B. “New cache designs for thwarting software cache-based side channel attacks”, ISCA 2007
- **[GeJi2019]** Ge, Jingquan et al. “AdapTimer: Hardware/Software Collaborative Timer Resistant to Flush-Based Cache Attacks on ARM-FPGA Embedded SoC”, ICCD 2019
- **[GeQi2016]** Ge, Qian et al. “Your Processor Leaks Information-and There's Nothing You Can Do About It”, 2016

Avez-vous des questions ?



Coeur Aubrac modifié

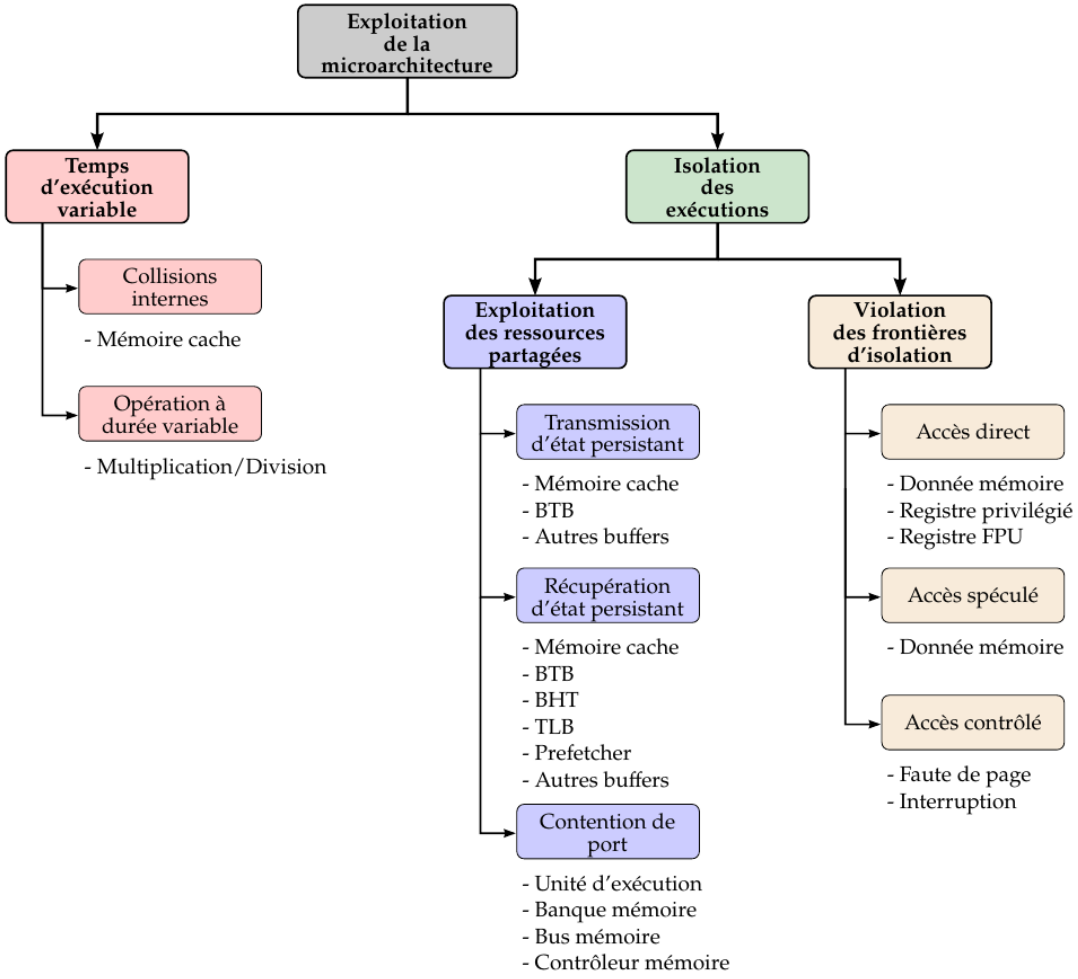


Coeur Salers modifié

8.

Diapositives supplémentaires

Classification des attaques



Récapitulatif des principes

N	Nom	Implémentation
1	Allocation statique	Tag et blocage des ressources
2	Séparation de la disponibilité	Partitionnement temporal
3	Partitionnement spatial	Répartition des ressources dupliquées
4	Suppression des traces	Effacement ou écrasement des états persistants
5	Relâchement en temps constant	Effacement ou écrasement en un temps fixé
6	Homogénéité	Implémentation de coeurs/threads équivalents
7	Plage mémoire possédée	Renforcement des frontières et de la spéculation
8	Plage mémoire dupliquée	Copie différente pour chaque domaine
9	Cohérence mémoire	Adaptation des protocoles de cohérence mémoire
10	Plage mémoire modifiable	Restriction de modification par un seul domaine

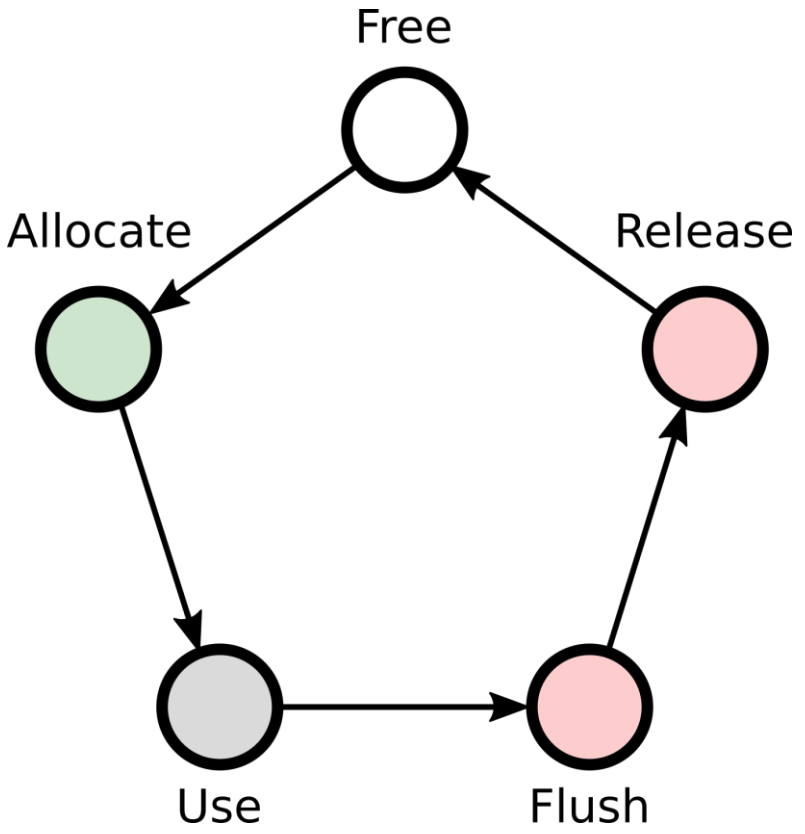
Cycle de vie d'une ressource

Différents états :

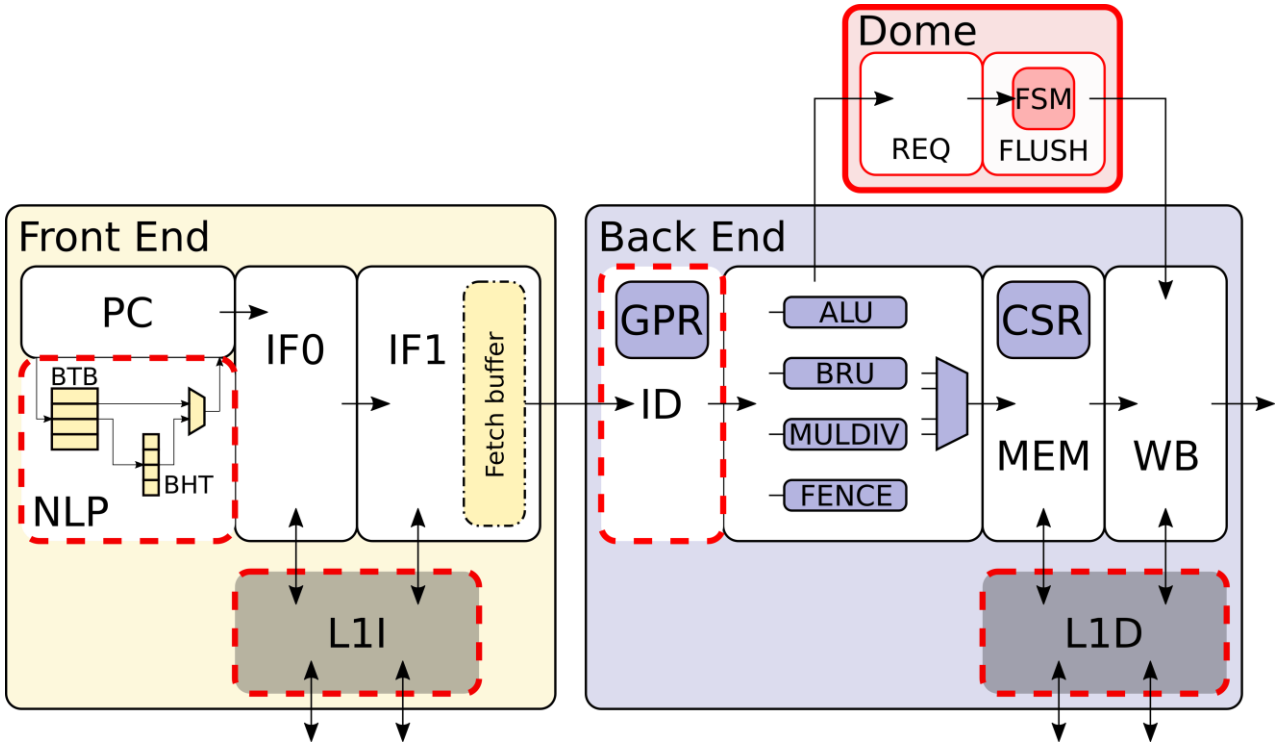
1. *Free* : allocation possible/libre,
2. *Allocate* : allocation en cours,
3. *Use* : utilisation possible,
4. *Flush* : ressource effacée,
5. *Release* : ressource libérée.

Partage temporel seulement :

- Simplification possible,
- Seuls *Use*/*Flush* utiles.



Coeur Aubrac : implémentation

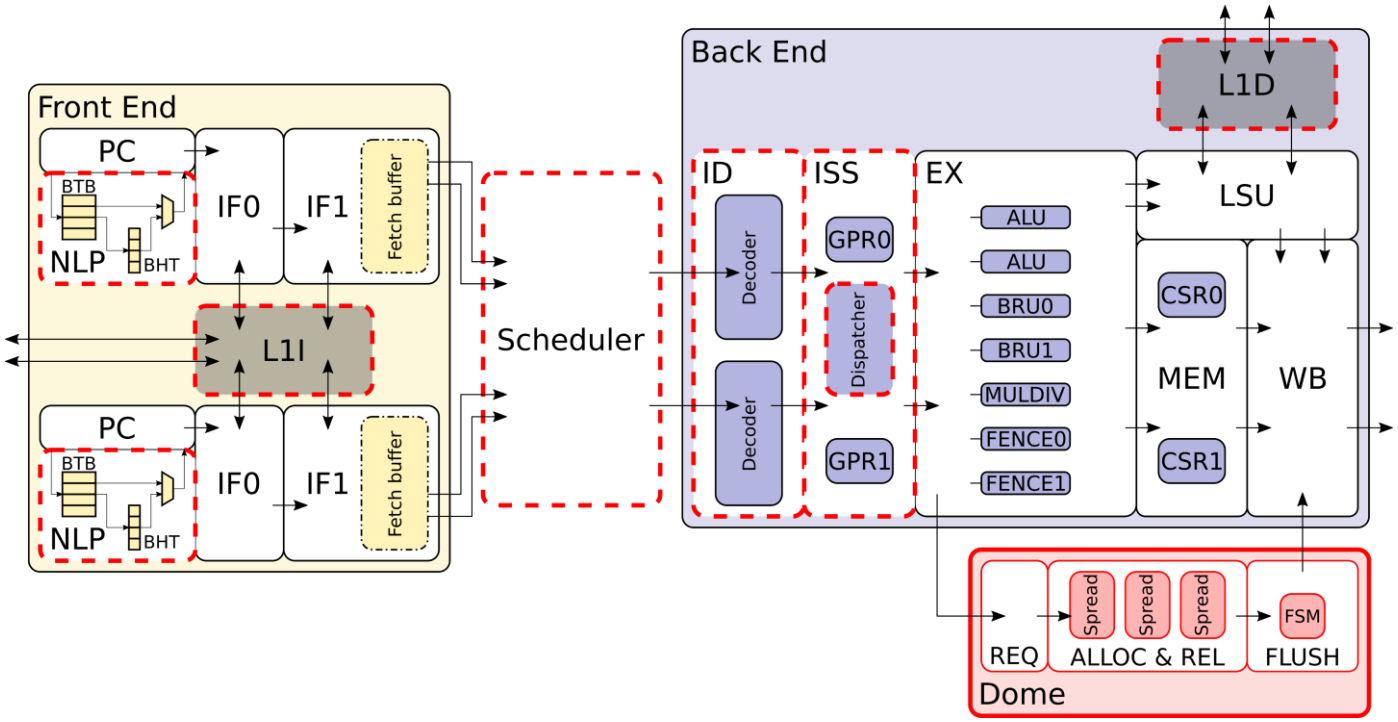


Coeur Aubrac modifié

Coeur Aubrac : application des principes

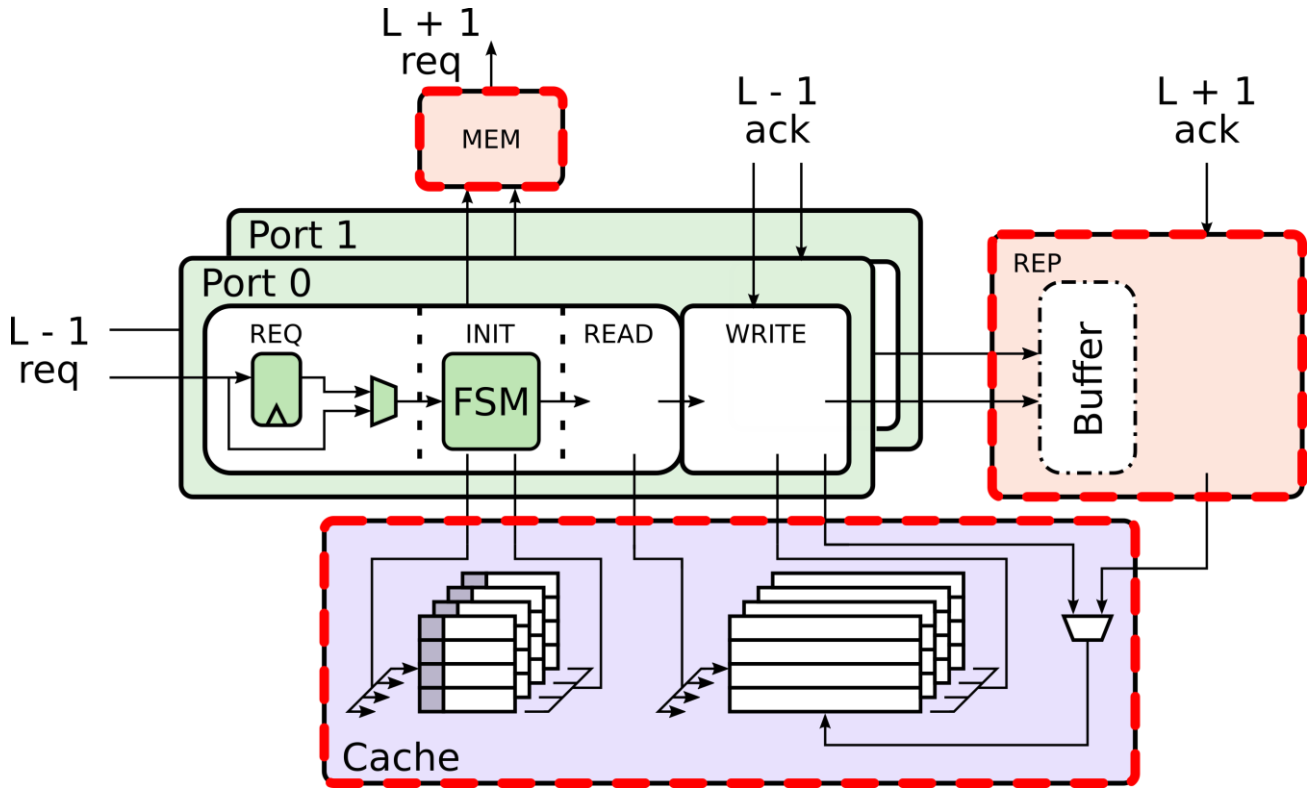
Ressource	Partage	Mécanisme
Toutes	T	Effacement

Coeur Salers : implémentation



Coeur Salers modifié

Mémoire cache: implémentation

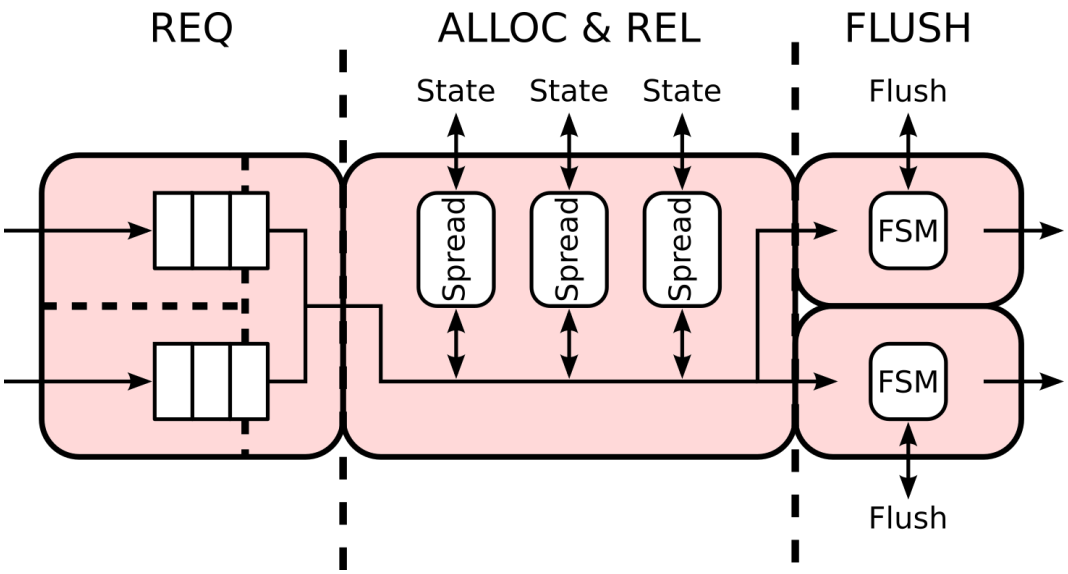


Mémoire cache modifiée

Coeur Salers : application des principes

Ressource	Partage	Mécanisme
Pipeline	T + S	Effacement + Allocation statique + Partitionnement spatial
Buffers	T + S	Effacement + Partitionnement spatial
Unité d'exécution	S	Allocation statique
Prédicteur de branchement	T	Effacement
BTB	T	Effacement
BHT	T	Effacement
Mémoire cache	T + S	Effacement + Partitionnement spatial + Partitionnement temporel
Emplacements mémoires	T + S	Effacement + Partitionnement spatial
Contrôleur	T + S	Effacement + Partitionnement spatial + Partitionnement temporel
Port mémoire	S	Partitionnement temporel

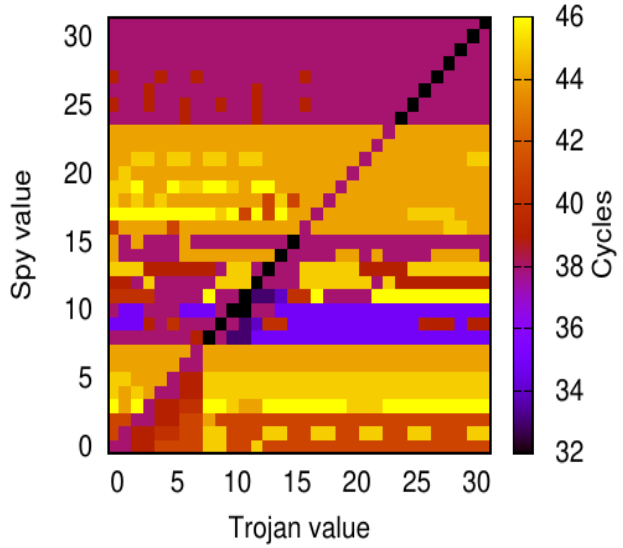
Unité de gestion des domes



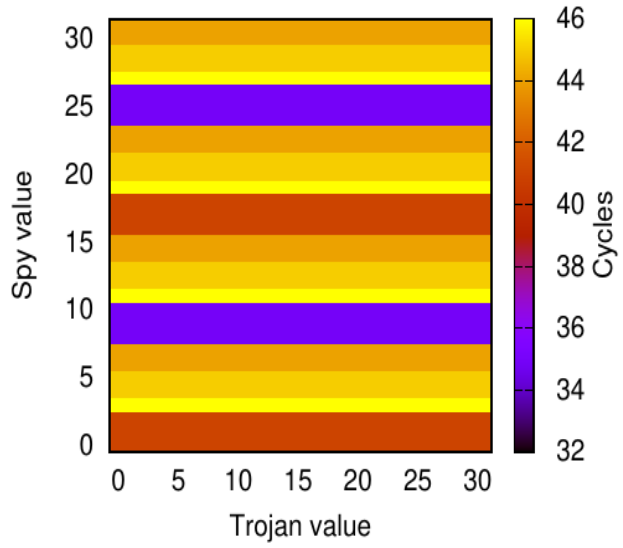
Unité de gestion des domes (Salers)

Test : BTB/ temporel

Non-protégé



Protégé



Caractéristiques du BTB :

- 32 emplacements.

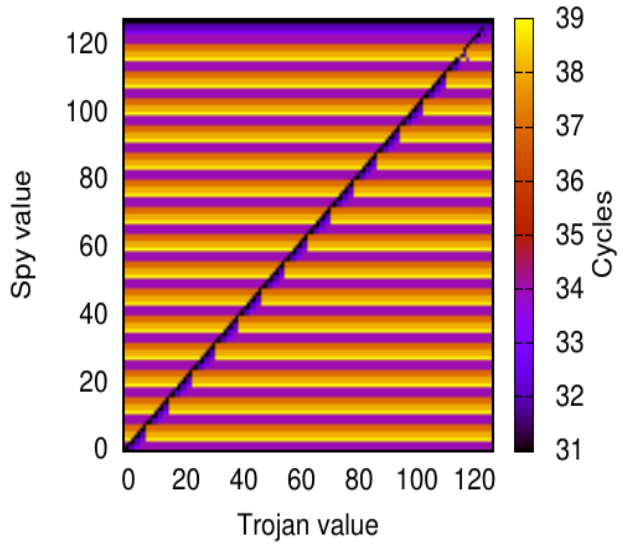
Cœur ciblé : Aubrac (sans SMT)

Lecture du diagramme :

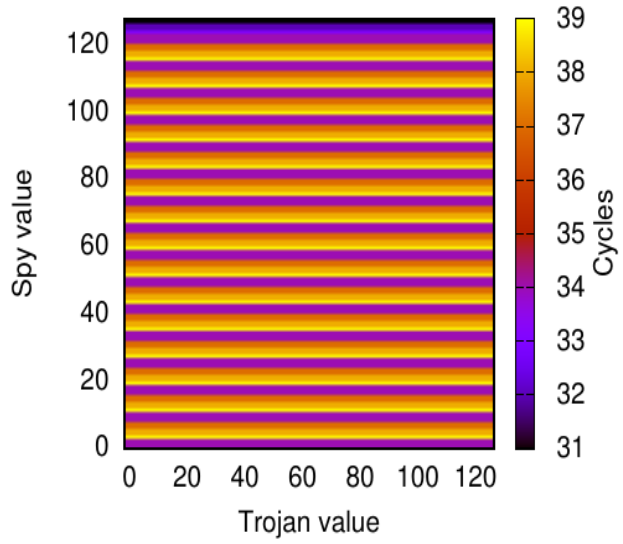
- Axe X : emplacement utilisé par le cheval de Troie.
- Axe Y : emplacement utilisé par l'espion.
- Couleur : nombre de cycles.

Test : BHT/ temporel

Non-protégé



Protégé



Caractéristiques du BHT :

- 128 compteurs.

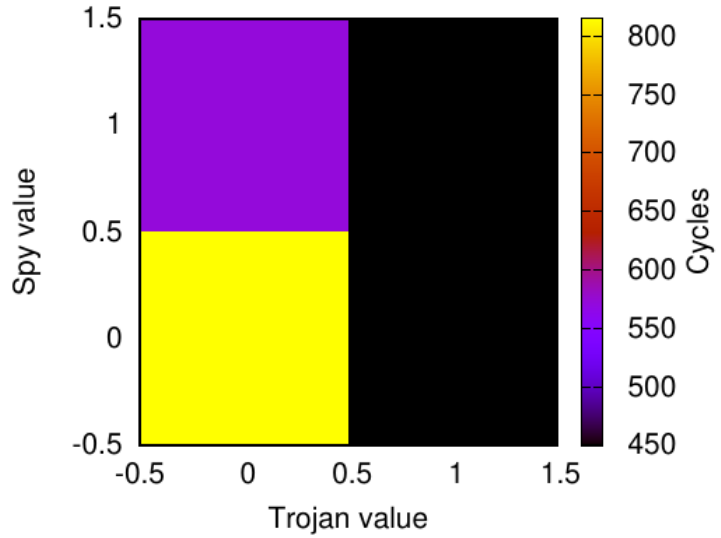
Cœur ciblé : Aubrac (sans SMT)

Lecture du diagramme :

- Axe X : compteur utilisé par le cheval de Troie.
- Axe Y : compteur utilisé par l'espion.
- Couleur : nombre de cycles.

Test : Unité d'exécution/ contention

Non-protégé



Protégé

Impossible : Un seul dome possède la ressource à la fois.

Caractéristiques de l'unité :

- Exécute les multiplications.
- Multiplications = plusieurs cycles d'horloge.
- 0 = pas utilisée, 1 = utilisée.

Cœur ciblé : Salers (avec SMT)

Lecture du diagramme :

- Axe X : utilisation par le cheval de Troie.
- Axe Y : utilisation par l'espion.
- Couleur : nombre de cycles.

Impact sur les performances

Cache	1 kB		4 kB	
Aubrac				
	4,70 ns	réf.	4,93 ns	×1,05
Dome	4,85 ns	×1,03	4,91 ns	×1,04
NLP	4,69 ns	×1,00	4,93 ns	×1,05
NLP Dome	4,76 ns	×1,01	5,05 ns	×1,07

Cache	1 kB		4 kB	
Salers				
	6,90 ns	réf.	7,48 ns	×1,08
Dome	6,82 ns	×0,99	7,50 ns	×1,09
NLP	7,57 ns	×1,10	7,69 ns	×1,11
NLP Dome	7,44 ns	×1,08	7,50 ns	×1,09

Impact des domes sur la fréquence maximale.

Impact sur les performances

Cache	1 kB	4 kB	1 kB	4 kB
	Aubrac		Salers (1 thread)	
	réf.	0,86	0,95	0,90
Dome	1,00	0,86	1,16	0,92
NLP	0,92	0,78	0,95	0,81
NLP Dome	0,92	0,78	1,07	0,82

Cache	1 kB	4 kB	1 kB	4 kB
Salers	1 thread actif		2 threads actifs	
	1,05	0,91	0,56	0,51
Dome	1,07	0,88	0,76	0,62
NLP	1,03	0,88	0,55	0,51
NLP Dome	1,05	0,86	0,75	0,61

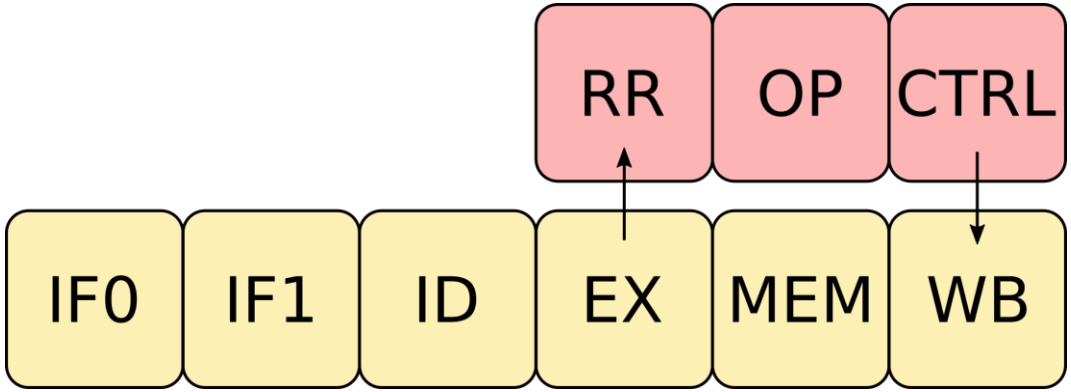
Moyennes géométriques obtenues avec Embench.

Impact sur les performances

	Non-protégé (cycles)	Protégé (cycles)	Surcoût (%)	Surcoût par `dome.switch` (cycles)
L1D (temporel)	27256	35880	+32%	67,4
L1I	57416	64264	+12%	53,5
BHT	1756202	1796774	+2%	19,8
BTB	445544	463443	+4%	35,0
L1D (spatial)	188026	134395	−29%	N/A
Contention	59250	N/A	N/A	N/A

Nombre de cycles par test de Timesecbench.

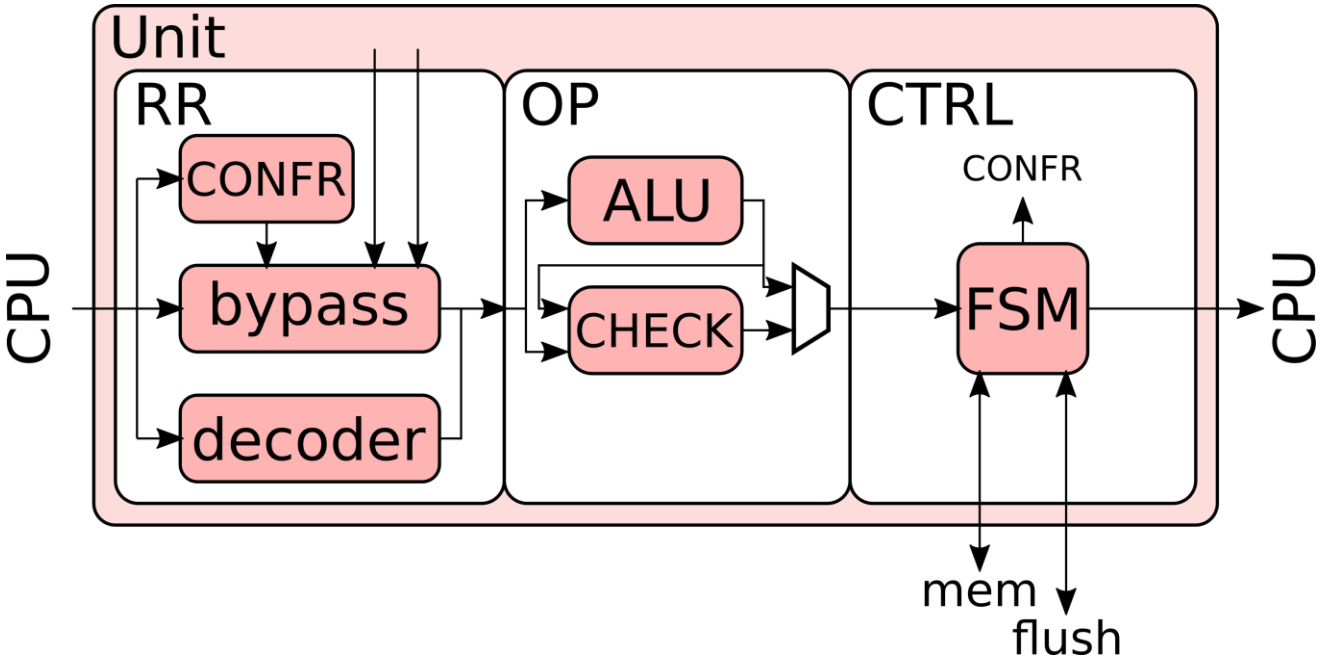
Vision du système pipeline + unite pour les domes (version dynamique)



Trois étages:

1. RR : Register Read Stage.
2. OP : Operation Stage.
3. CTRL : Control Stage

Unité pour les domes (version dynamique)



Trois étages:

1. RR : Register Read Stage.
2. OP : Operation Stage.
3. CTRL : Control Stage

Coût des domes (version dynamique)

Version	LUT	Registres (FF)
Pipeline	3927	2406
Pipeline + NLP	6602 (+68,1%)	5494 (+128,3%)
Pipeline + Domes (2C)	5258 (+33,9%)	3147 (+30,8%)
Pipeline + Domes (4C)	5560 (+41,6%)	3315 (+37,8%)
Pipeline + Domes (8C)	5735 (+46,0%)	3641 (+51,3%)
Pipeline + NLP + Domes (8C)	8565 (+118,1%)	6734 (+179,9+%)

C = Registre de configuration