Processeurs haute performance

Pascal Sainrat

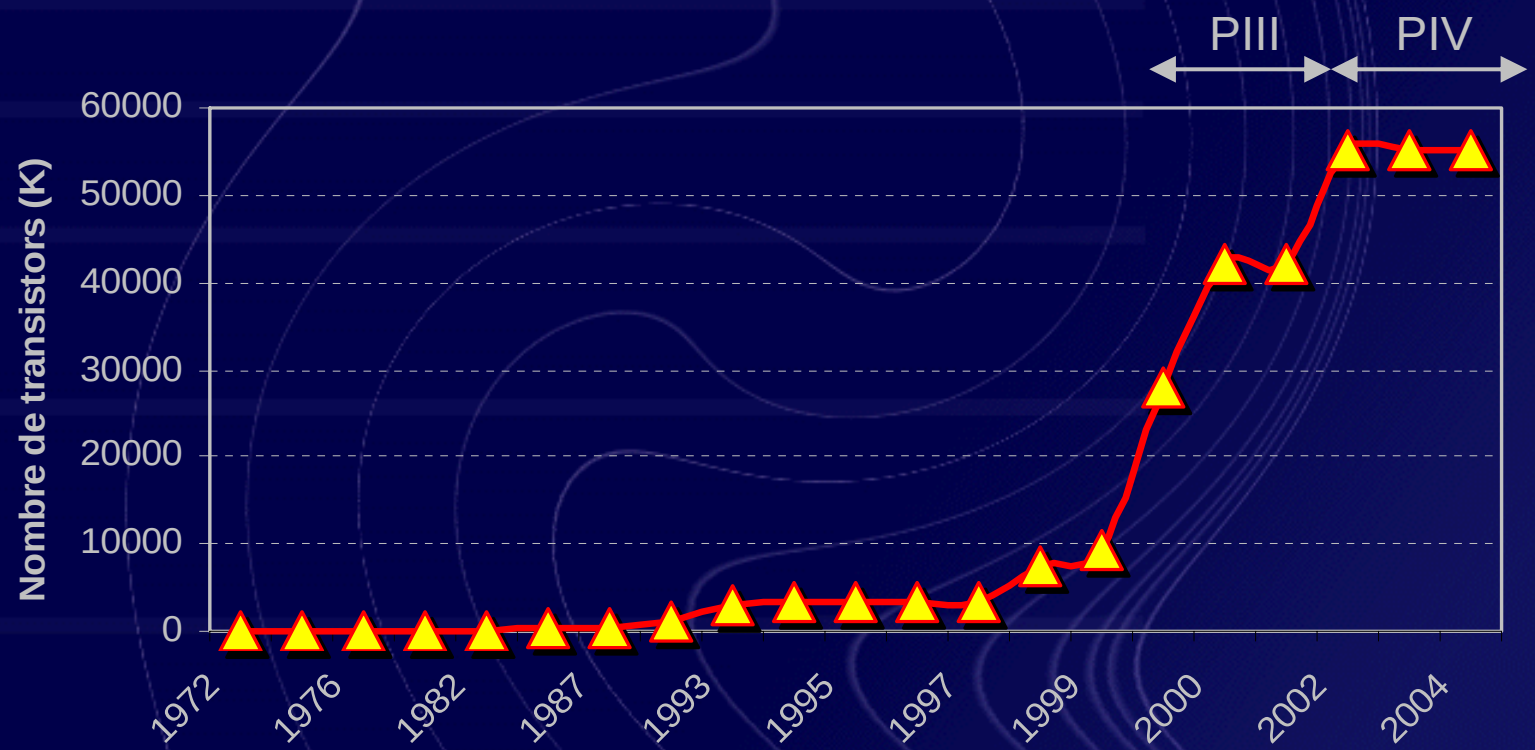
ACACES'2005 - 24 au 30 juillet - Hipeac

- Maurice Wilkes
- David August, Princeton University, Simulation
- Doug Burger, University of Texas at Austin, Tiled architectures
- Jose Duato, University of Valencia, High-speed interconnection networks
- Josh Fisher, HP, High performance embedded computing
- Rajiv Gupta, University of Arizona, Compilation for embedded processors
- Avi Mendelson, Intel, Israel Low power
- Trevor Mudge, University of Michigan, Memory systems and their implementation technologies
- Mike O'Boyle, University of Edinburgh, Adaptive & feedback driven compilation
- Yale Patt, University of Texas at Austin, Advanced microarchitecture
- Jim Smith, University of Wisconsin-Madison, Virtual machines
- Per Stenström, Universtiy of Chalmers, Chip multiprocessors
- Olivier Temam, INRIA Futurs, France Simulation
- Ayal Zaks, IBM, Optimizations in GCC

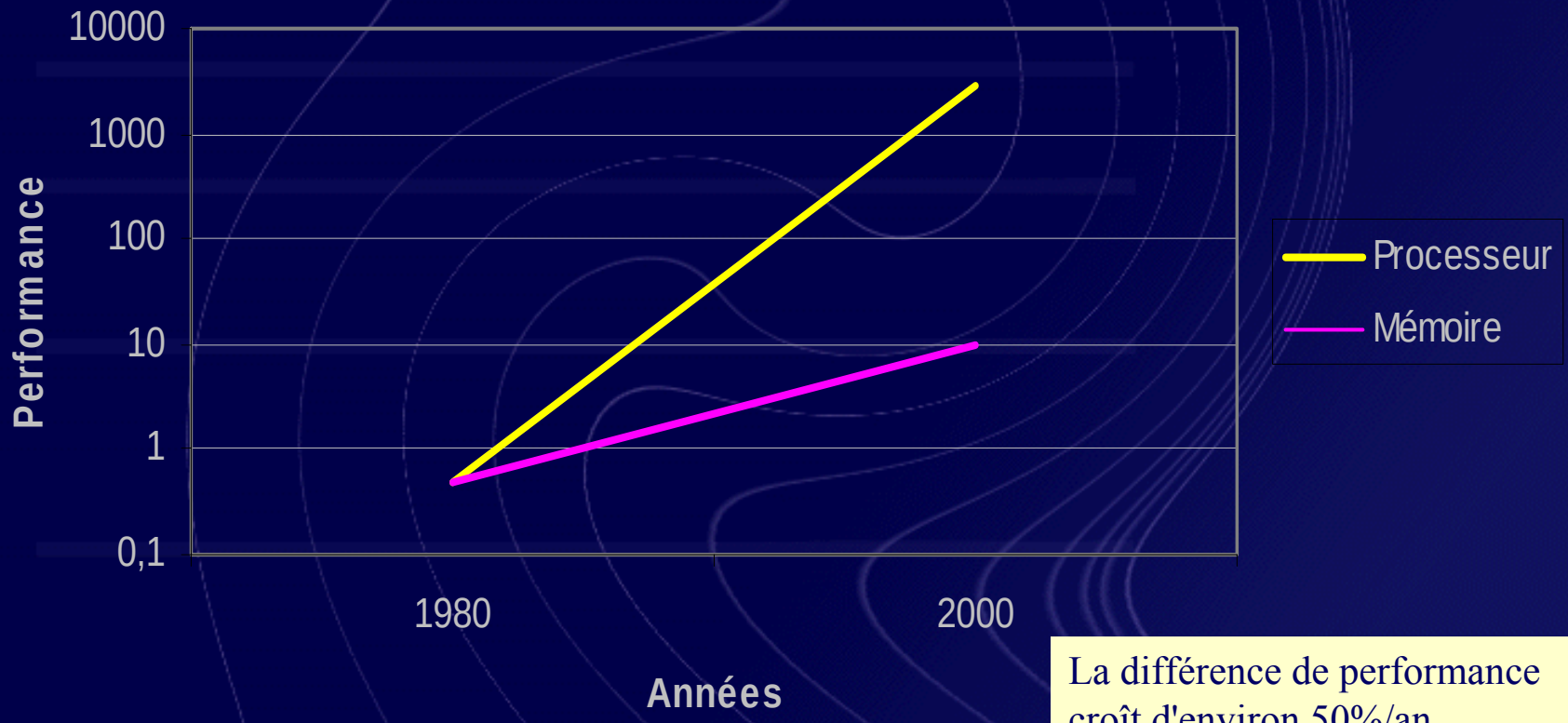
http://www.hipeac.net/summerschool/

- **Important dates**
  - Application: May 15, 2005
  - Notification: June 1, 2005 (number of participants limited)
  - Registration: June 15, 2005
  - Posters: July 1, 2005
- **Certificate**
  - Participants receive a certificate of attendance which can be accepted in PhD programs.
- Poster session for students – Excursion à Tivoli – Diner de gala
- 990 € - possibilité de bourse pour les étudiants

Roadmap : 1 milliard de transistors



- Le processeur accède à la mémoire pour charger des instructions et charger des données.



La différence de performance croît d'environ 50%/an.

Architecture

- **Parallélisme bit**
- **Pipeline**
- **Caches**
- **Superscalaire**
- **Exécution non ordonnée**
- **Spéculation**
- **VLIW / EPIC**
- **Hyperthreading**

Latence versus débit

Avion	Toulouse/ Autrans	Vitesse	Passagers	Débit (p*km/h)
train	8 heures	68,75 km/h	200	13750
voiture	5 heures	110 km/h	4	440

Formule magique

$$\text{Temps d'exécution} = \frac{\text{Nombre d'instructions} * \text{Temps de cycle}}{\text{Nombre d'instructions par cycle}}$$

	Instructions	Instructions/cycle	Temps de cycle	Temps d'exécution
68030	1	0,19	60	31
SPARC	1,2	0,77	60	9,3

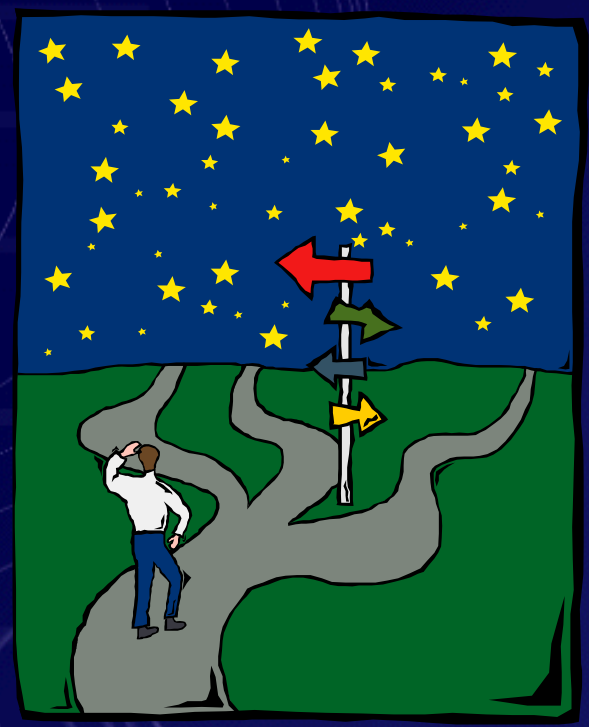
Diminuer le temps de cycle

$$\text{Temps d'exécution} = \frac{\text{Nombre d'instructions} * \text{Temps de cycle}}{\text{Nombre d'instructions par cycle}}$$

- Technologie
- Organisation
  - Simplification du jeu d'instructions (codage)
  - Augmentation du nombre d'étages du pipeline
    - ☹ Dépendances de données
    - ☹ Dépendances de contrôle

Dépendances de contrôle
→ Prédiction de branchement

- Prédiction qui prend en compte le comportement passé des branchements
- Prédiction dès le premier cycle
  - Prédiction de la présence d'un branchement
  - Prédiction du type du branchement
  - Prédiction de l'adresse cible
  - Prédiction de la direction pour les branchements conditionnels
- Vérification lors de l'exécution du branchement et annulation si mauvaise prédiction



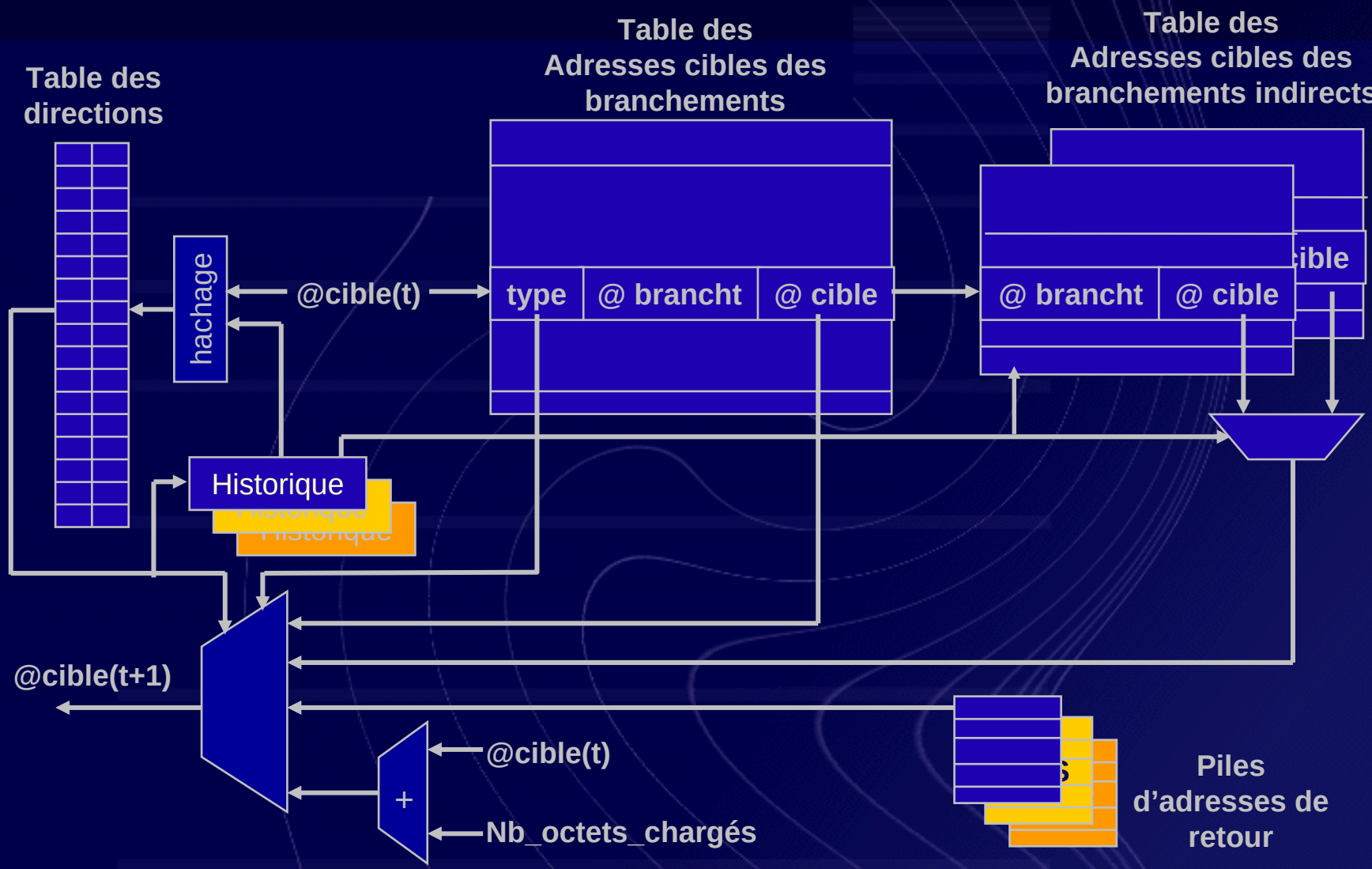
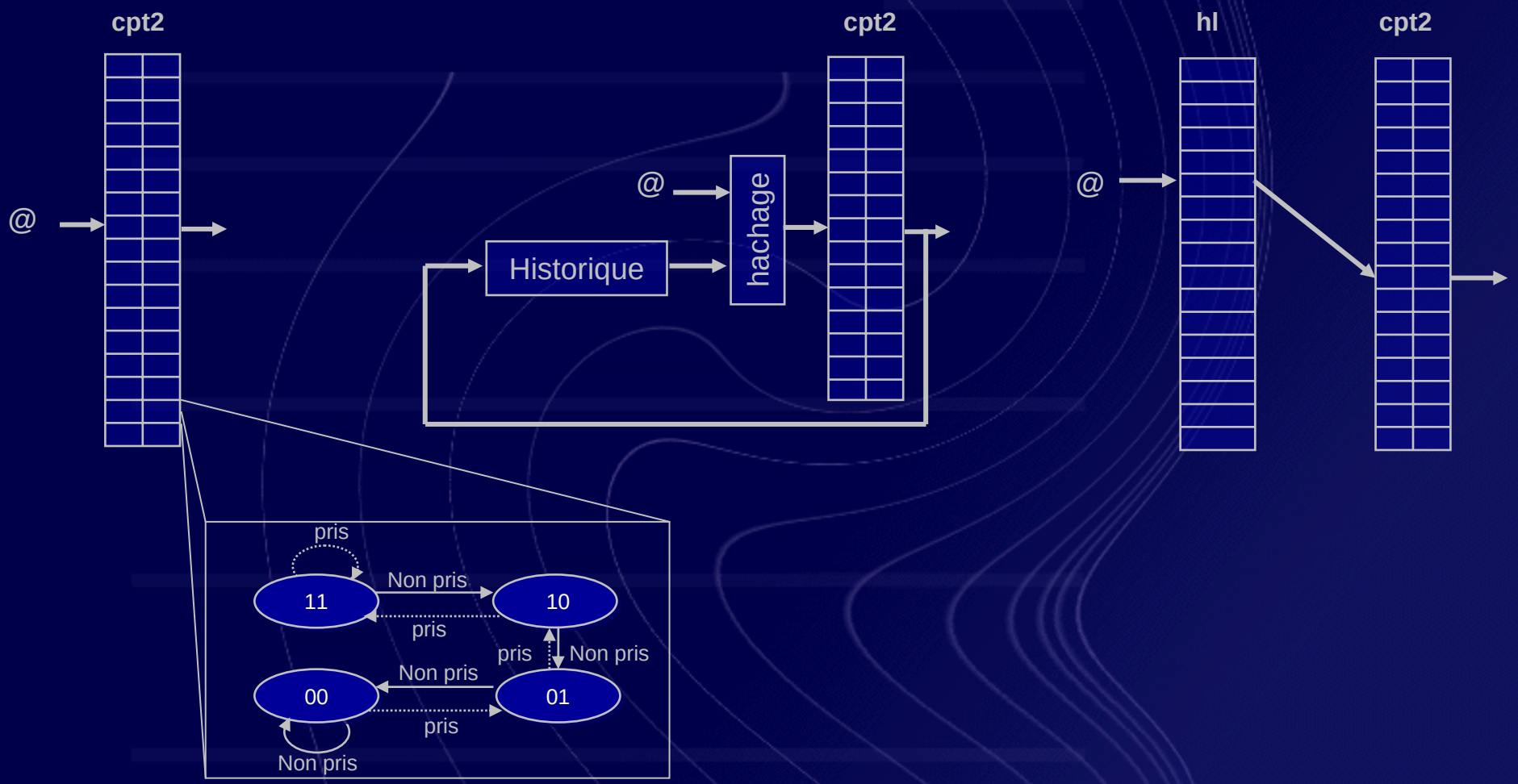


Table des directions
Table des Adresses cibles des branchements
Table des Adresses cibles des branchements indirects
hachage
@cible(t)
type
@ brancht
@ cible
@ brancht
@ cible
Historique
@cible(t+1)
@cible(t)
+
Nb_octets_chargés
Piles d'adresses de retour

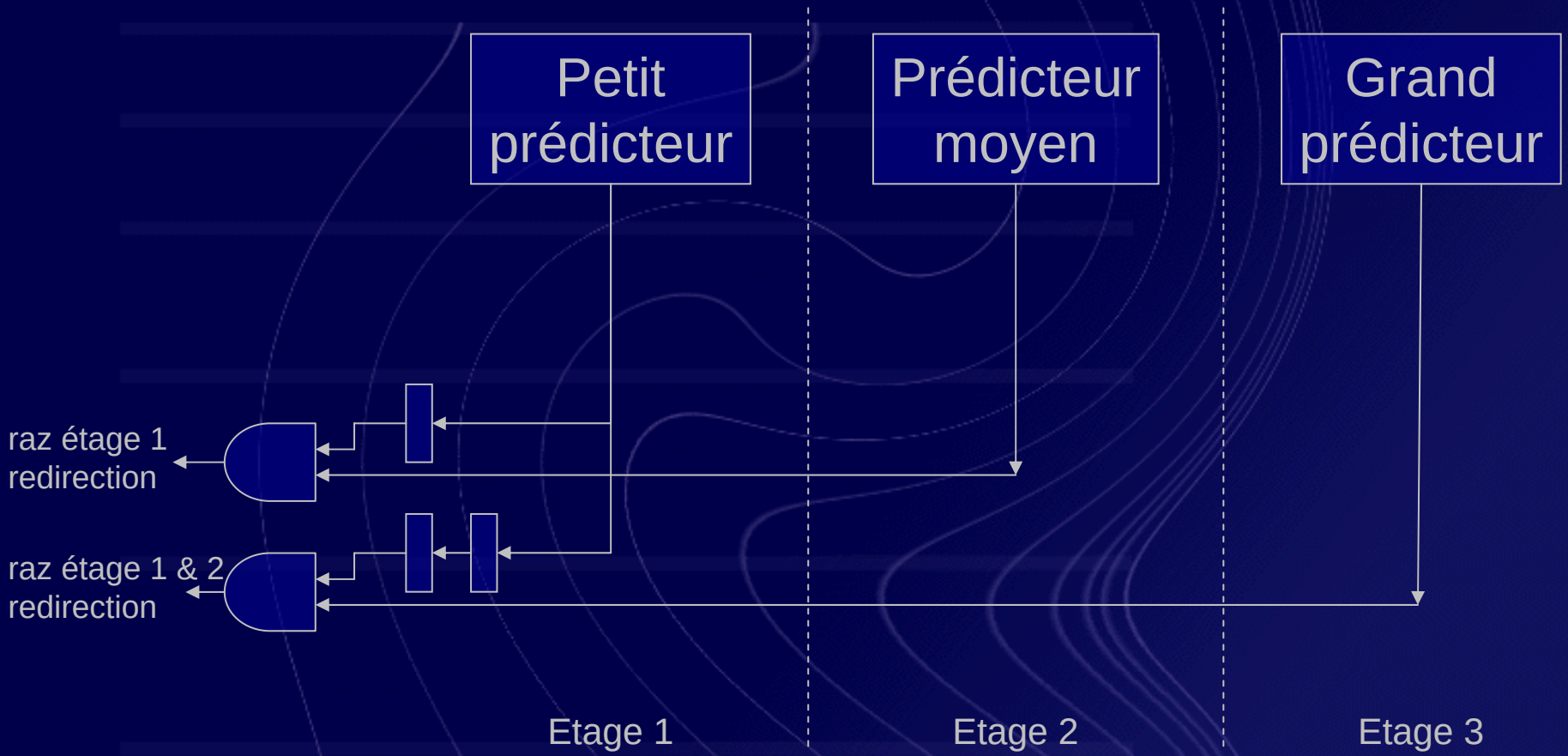
Prédiction de la direction



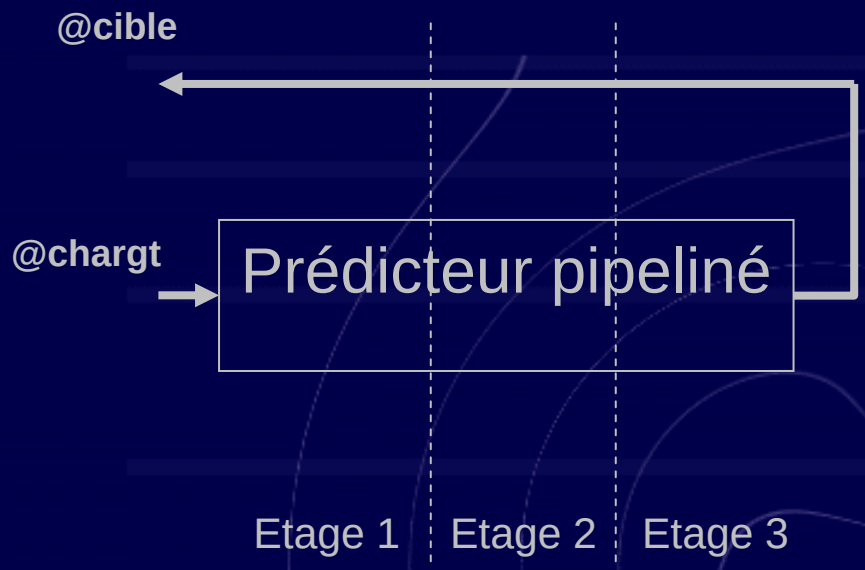
Prédiction de branchement

- Augmentation de la longueur du pipeline
  - Diminution du temps d'un étage
  - Améliorer la qualité = augmenter la latence de la prédiction
- Prédiction hiérarchique
- Prédiction pipelinée et en avance
- Prédiction hybride

Prédiction au premier cycle
→ Prédiction hiérarchique



Prédiction pipelinée et en avance

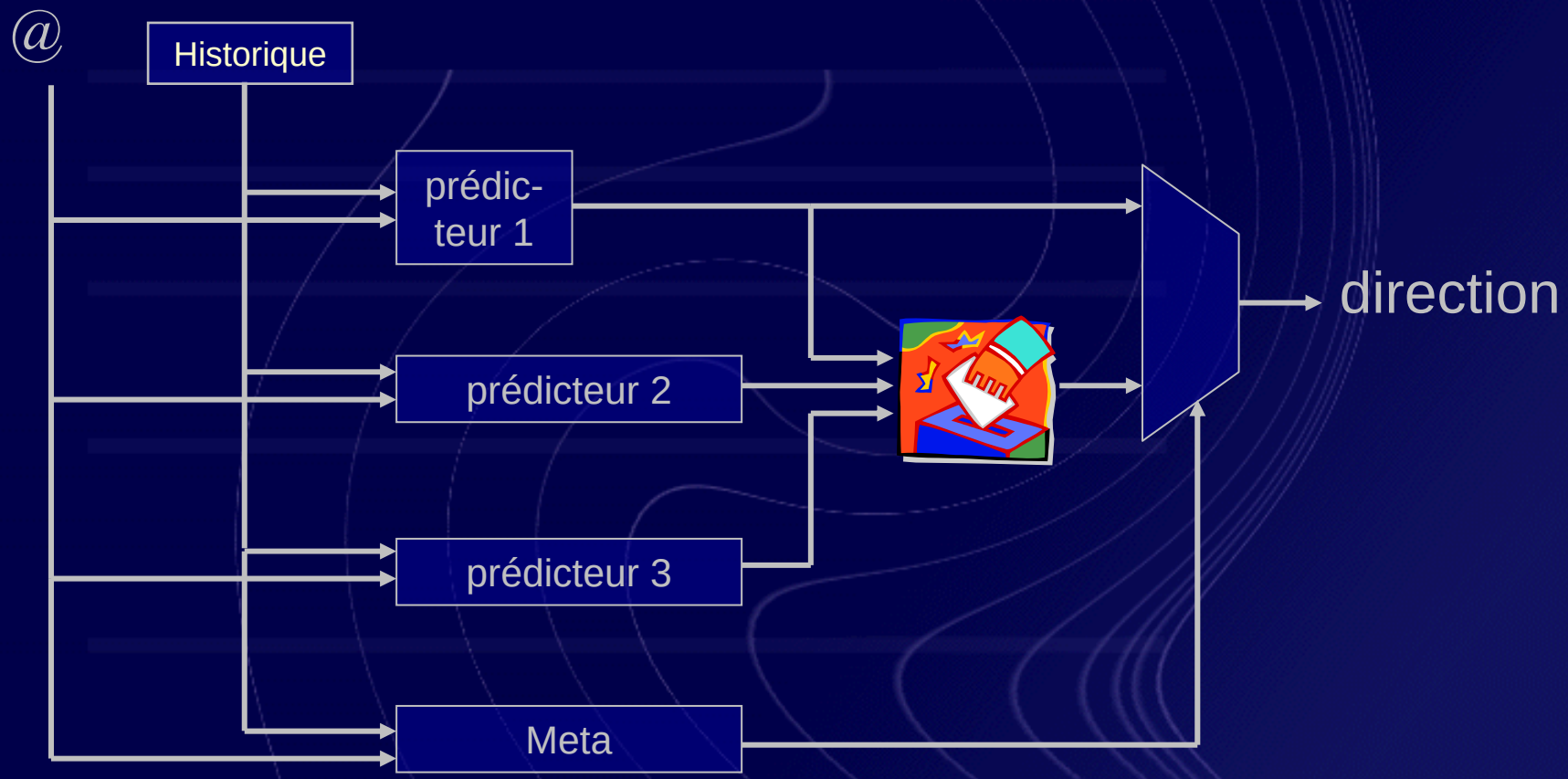


Début prédiction
pour le cycle i+3

Adresse cible
prédite au cycle i+3

Multiple-Block Ahead Branch Prediction

Améliorer la prédiction
→ Prédicteur hybride



Design Tradeoffs for the Alpha EV8 Conditional Branch Predictor

Instructions conditionnelles

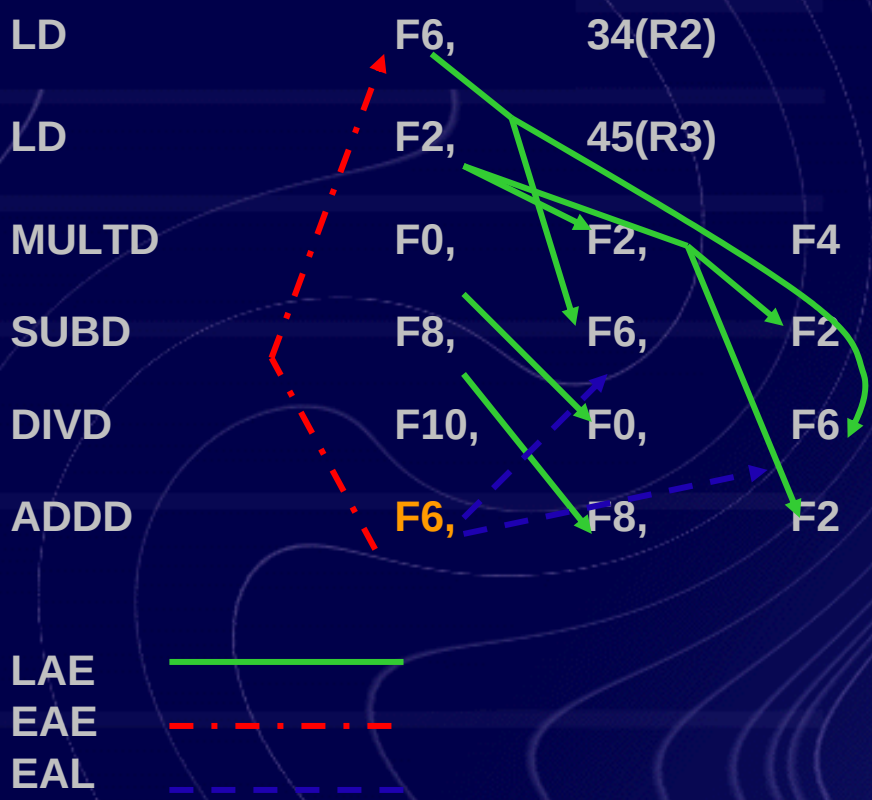
- Une instruction se réfère à une condition qui est évaluée au cours de l'exécution de l'instruction si la condition est vraie, l'instruction est exécutée sinon elle continue comme si c'était un NOP.
- Déja utilisé pour les transferts registre/registre
  - Mips, SPARC, POWERPC, alpha, p6
- Exemple:

  If (A=0) {S=T}

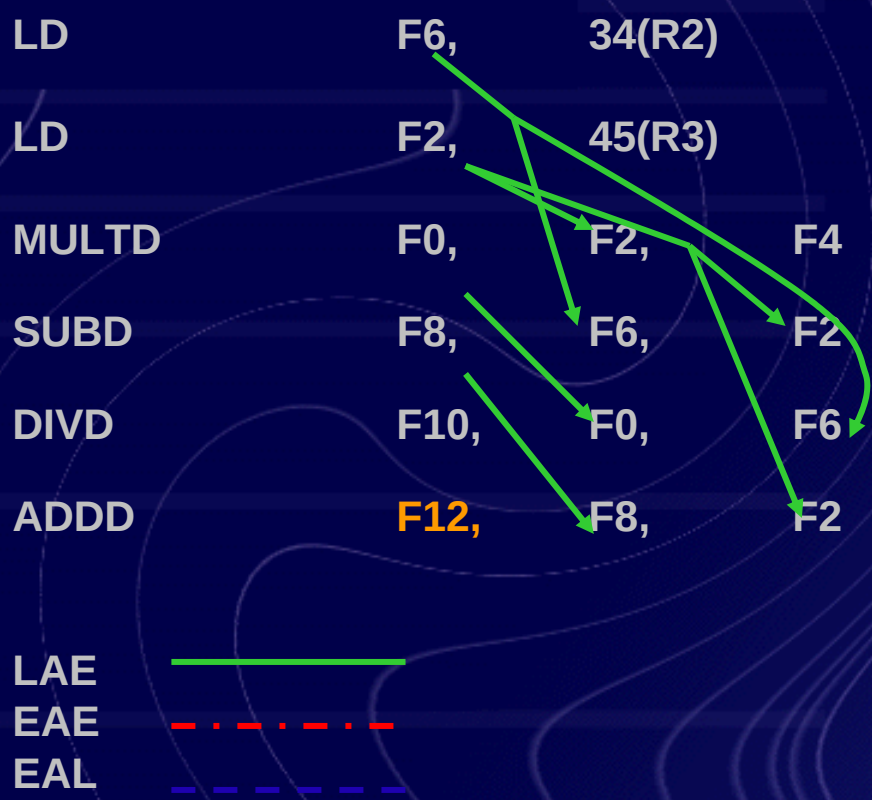
Code normal	instruction conditionnelle équivalente
BEQZ r1,I	CMOVZ r2,r3,r1
MOV r2,r3	

- Convertit une dépendance de contrôle en une dépendance de données (résolution plus tard dans le pipeline)
- Généralisé dans l'Itanium, dans l'ARM, …

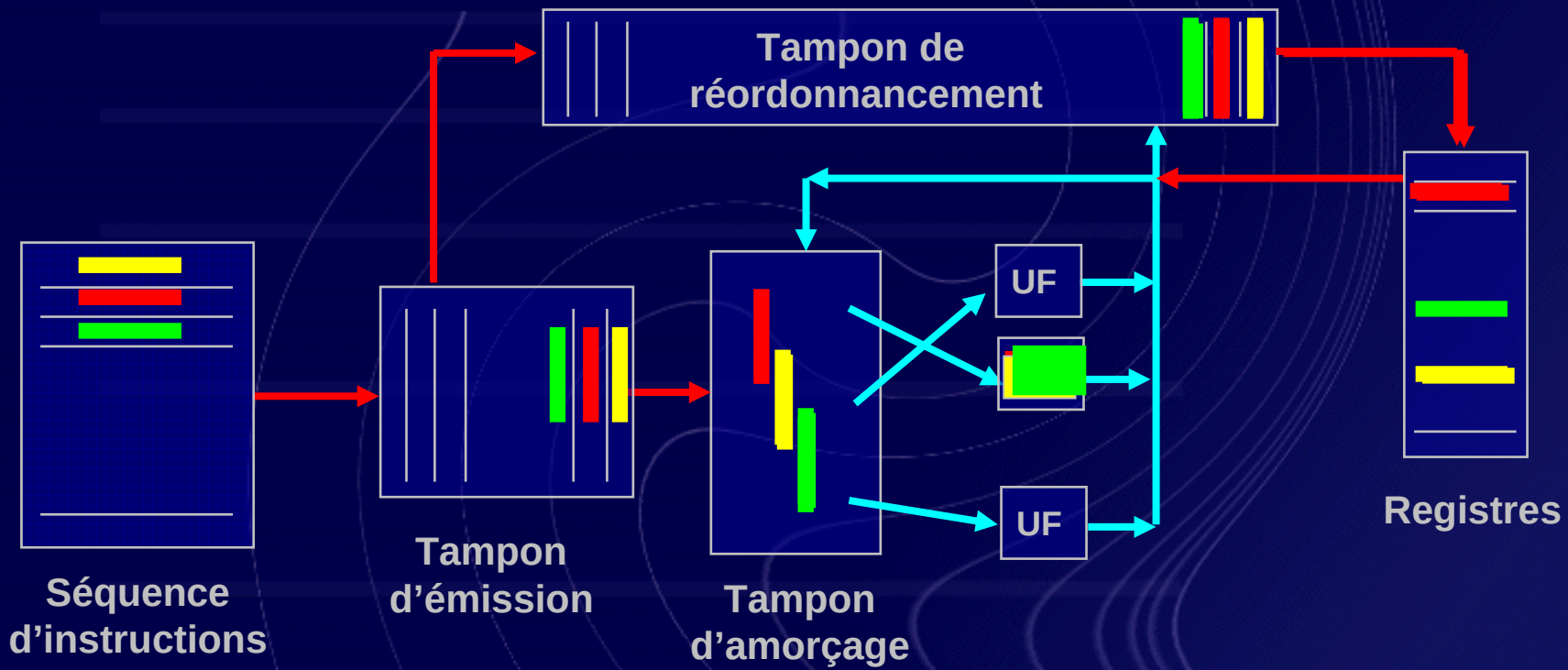
Dépendances de données



Renommage


LD F6, 34(R2)
LD F2, 45(R3)
MULTD F0, F2, F4
SUBD F8, F6, F2
DIVD F10, F0, F6
ADDD F12, F8, F2
LAE
EAE
EAL

Renommage & ordonnancement dynamique



Exécution spéculative: 4 étapes (1)

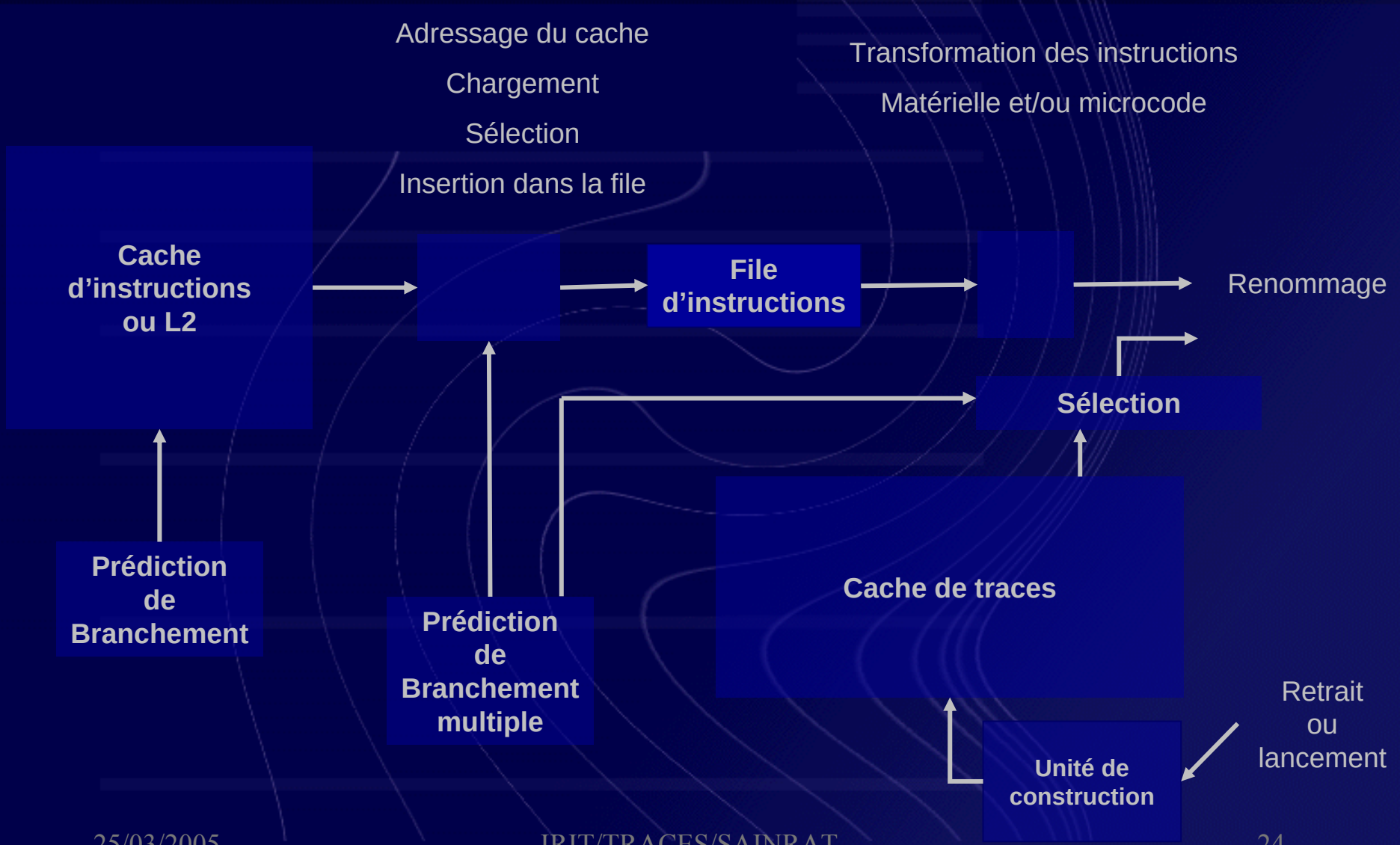
- Lancement ou distribution
  - si SR libre et une entrée du ROB libre sinon suspension car le lancement doit se faire dans l'ordre
  - Les opérandes sont lus dans les registres ou dans le ROB s'ils sont prêts sinon n° d'entrée du ROB pour les retrouver
  - Le n° d'entrée du ROB est transmis à la SR pour mettre à jour après exécution
- Exécution
  - Tant qu'un des deux opérandes est absent, surveiller le CDB (n° entrée ROB)
  - Exécuter si l'UF est libre une des instructions prêtes

Exécution spéculative: 4 étapes (2)

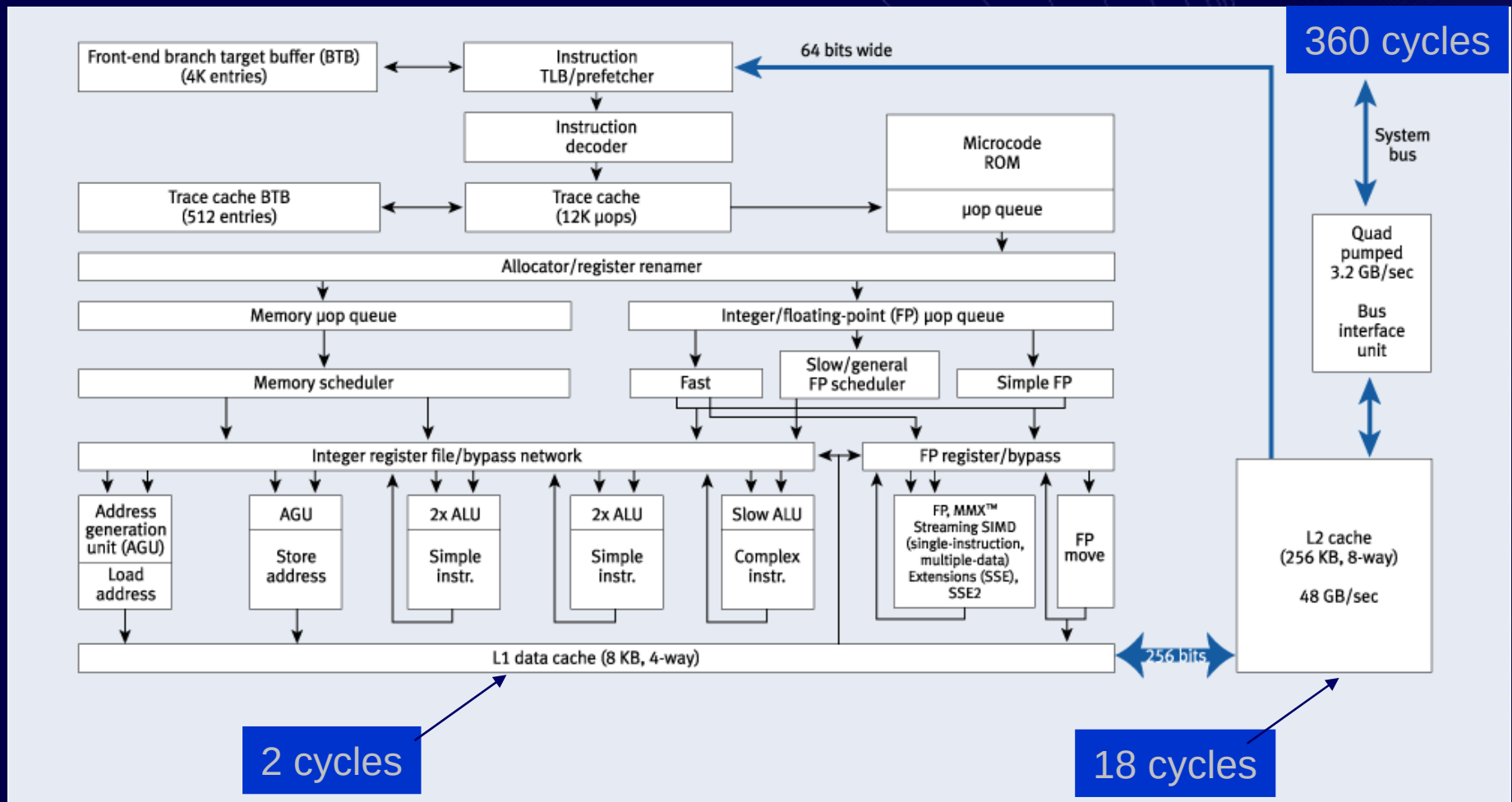
- Ecriture du résultat
  - Résultat envoyé sur le CDB avec le n° d'entrée du ROB, va dans le ROB et les SR
- Garantie
  - Lorsqu'une instruction autre qu'un branchement mal prédit arrive en tête du ROB, elle est retirée du ROB libérant ainsi une entrée.
  - Le registre (ou l'emplacement mémoire) est mis à jour.
  - Si branchement mal prédit, le ROB est nettoyé et le chargement des instructions repart avec la bonne instruction.
  - Le ROB est une file circulaire.

Quelques problèmes

- Nombre de registres physiques
  - Temps d'accès
    - Clusters, hiérarchie, …
- Lectures mémoire spéculatives
  - écritures non spéculatives
    - Prédiction de dépendances
- Complexité de l'issue
  - pre-scheduling
  - trace processor
  - clusters


Adressage du cache
Chargement
Sélection
Insertion dans la file
Transformation des instructions
Matérielle et/ou microcode
Cache d'instructions ou L2
File d'instructions
Renommage
Sélection
Prédiction de Branchement
Prédiction de Branchement multiple
Cache de traces
Retrait ou lancement
Unité de construction

Exemple : Pentium 4

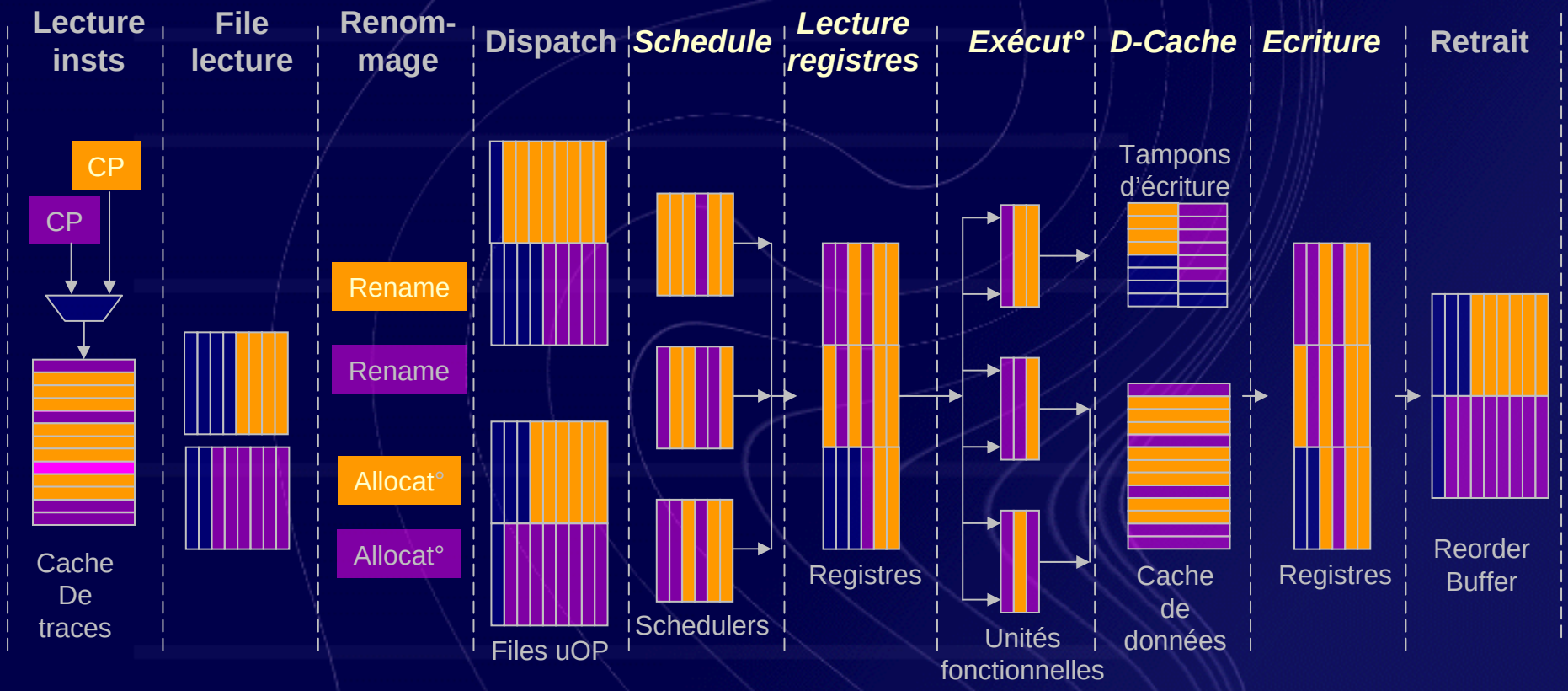


Simultaneous multithreading

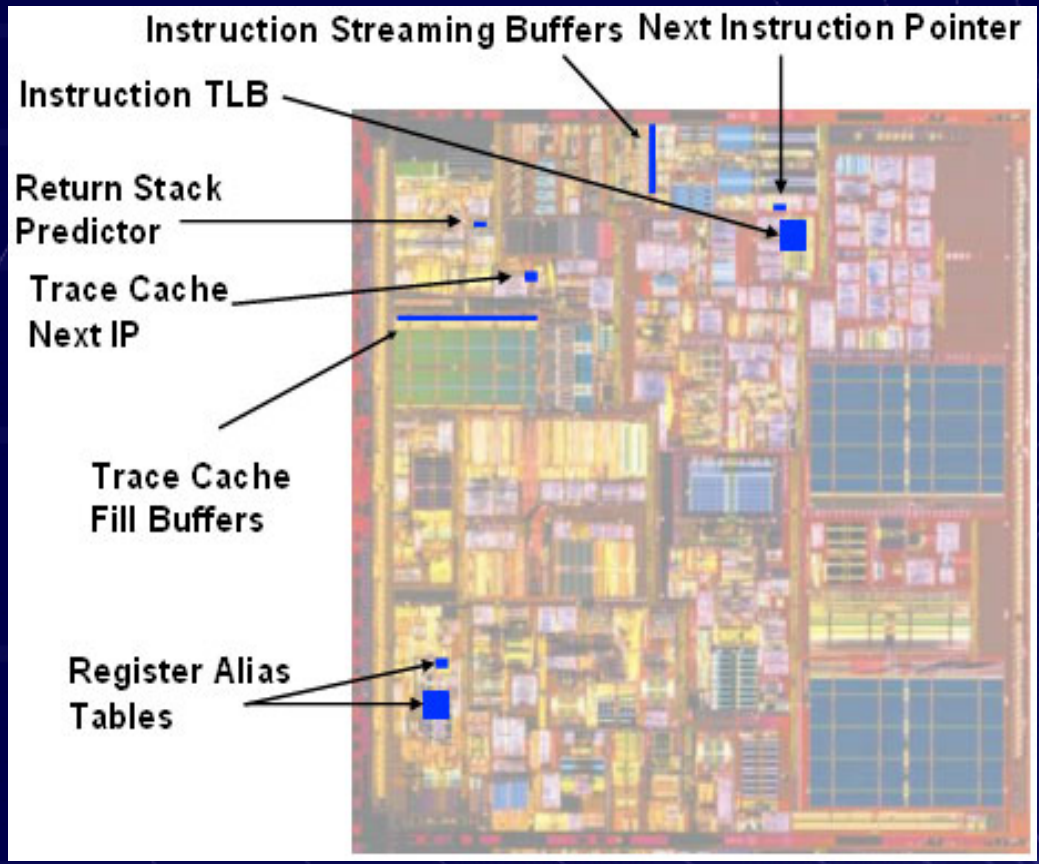
- But 1 : tolérer de longues latences.
  - Quand un thread est bloqué, l'autre peut continuer son exécution et profiter de certaines ressources qui sont inoccupées
- But 2 : mieux utiliser les ressources présentes
  - Augmenter le nombre d'instructions qui peuvent s'exécuter en parallèle.

Simultaneous multithreading

Exemple de l'hyperthreading dans le Pentium 4



Puce du P4 hyperthreadé



Mise en œuvre de l'hyperthreading

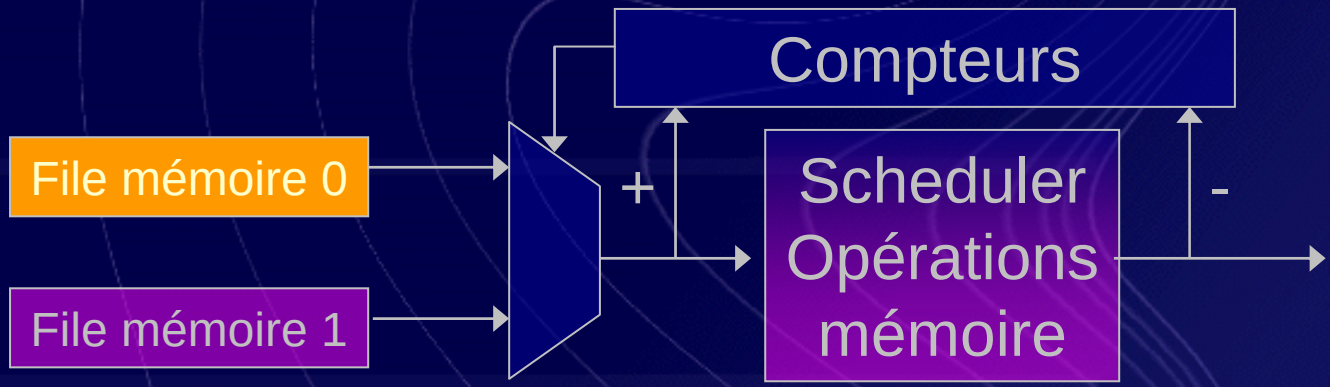
- Structures doublées
  - PC
  - Table de renommage
  - Constructeur de traces
  - Pile d'adresses de retour
  - APIC
- Partage des structures
  - Statique, dynamique avec maximum, dynamique
- Politique de chargement des instructions
  - ☹ Statique : alternance
  - ☺ Dynamique : icount
- Vidage des instructions bloquées d'un thread ?

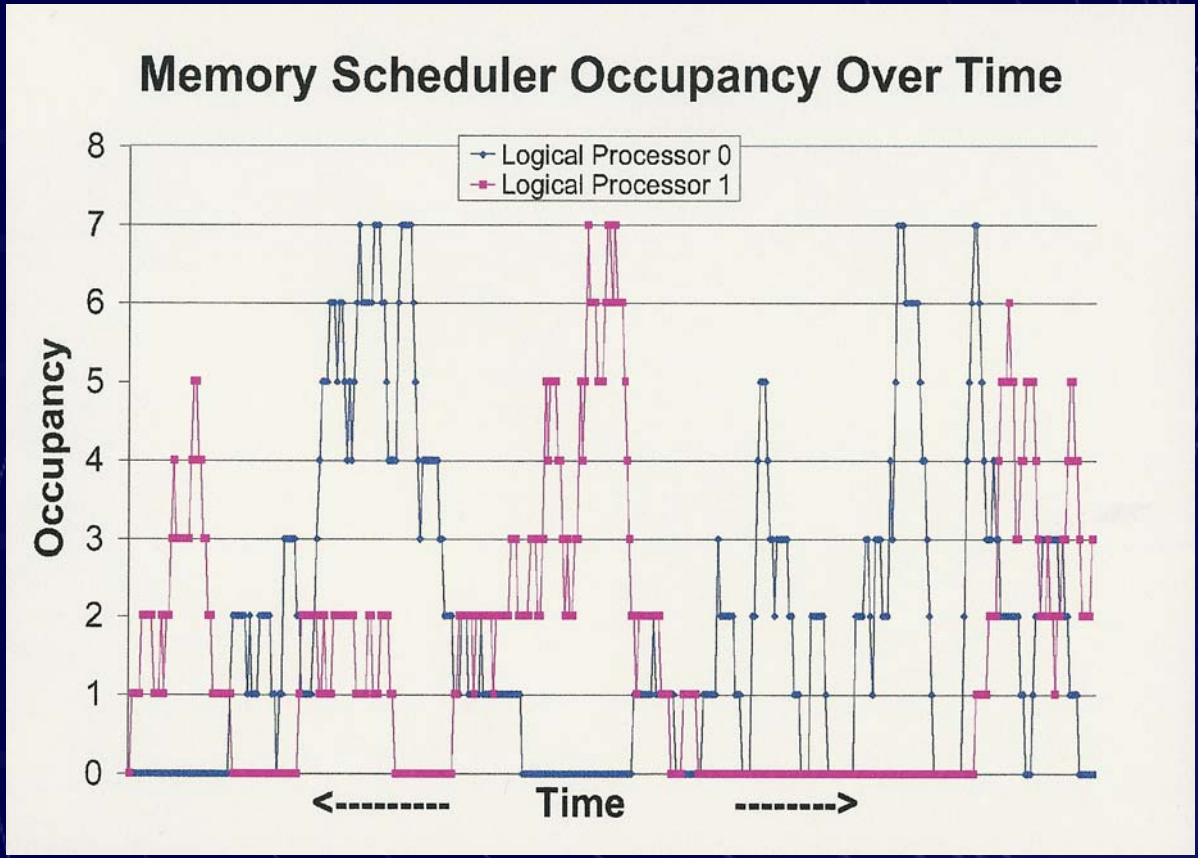
Partage statique

- Files entre les grandes phases du pipeline (fetch, dispatch, retire)
  - les instructions sont susceptibles d'y rester longtemps.
  - Non prédictibles
  - Très utilisées
- Exemple : les tampons d'écritures
- Moyen d'éviter un blocage sans pour autant avoir une réelle politique de scheduling entre threads

Partage dynamique avec maximum

- Pour les structures généralement peu occupées
- Ou les instructions passent peut de temps
- Exemple : les schedulers d'instructions




Memory Scheduler Occupancy Over Time
Logical Processor 0
Logical Processor 1
Occupancy
8
7
6
5
4
3
2
1
0
<----------
Time
-------->
Time

Partage dynamique total

- Pour les grandes structures
  - Quand l'occupation peut être très variable
  - Un thread ne va pas bloquer l'autre
- Exemple : les caches
  - Les threads se partagent certaines données
  - Degré d'associativité élevé pour éviter des échecs dus à des conflits (8 pour le L2 et le L3).
  - En moyenne, un cache partagé est 12% plus performant qu'un cache partitionné.
  - L'amélioration du taux de succès est entre 1 et 3,1 fois meilleur avec un cache partagé.

Performance de l'hyperthreading

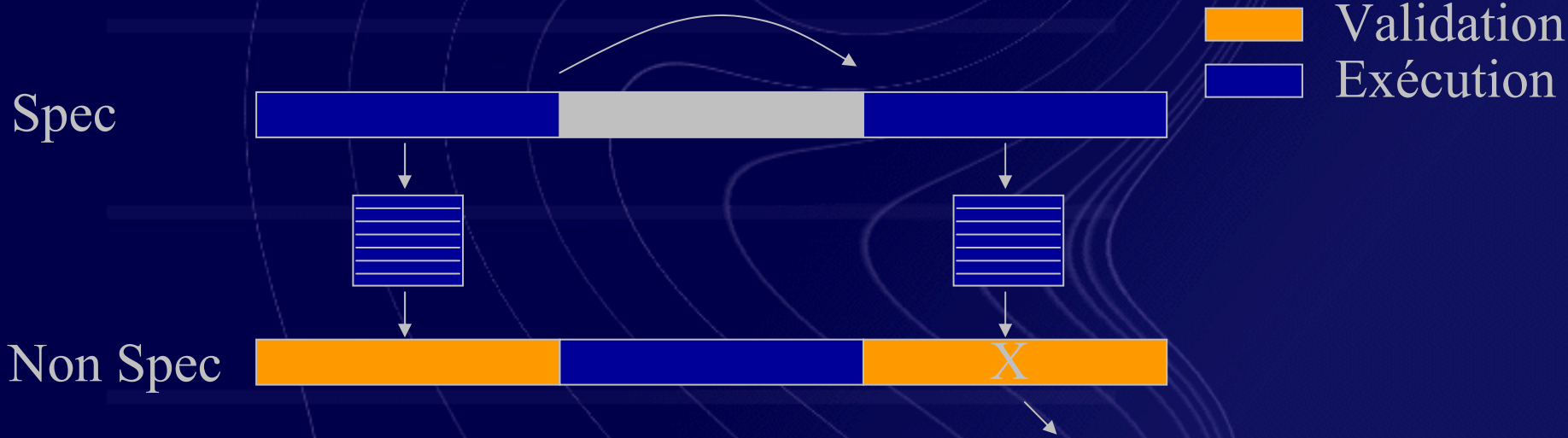
- Amélioration d'environ 20% pour une augmentation de la complexité d'environ 5%.
- Meilleur si les threads ne nécessitent majoritairement pas les mêmes ressources
  - Par exemple, un thread qui fait majoritairement du calcul entier, un autre qui fait majoritairement du calcul flottant.

Nouvelle primitive de communication

- Primitive implémentée dans le processeur pour synchroniser deux threads.
- MONITOR et MWAIT
- MONITOR définit une zone mémoire à surveiller
- MWAIT attend une écriture dans cette zone
  - MWAIT se comporte comme un NOP
    - Très peu d'énergie consommé
    - Pas de ressources utilisées

Spéculation de traces utilisant le SMT

- Traces définies à la compilation (feedback)
- Prédire les valeurs de sortie d'une trace
- Exécuter spéculativement une trace avec ces valeurs
- Pour paralléliser l'exécution de traces

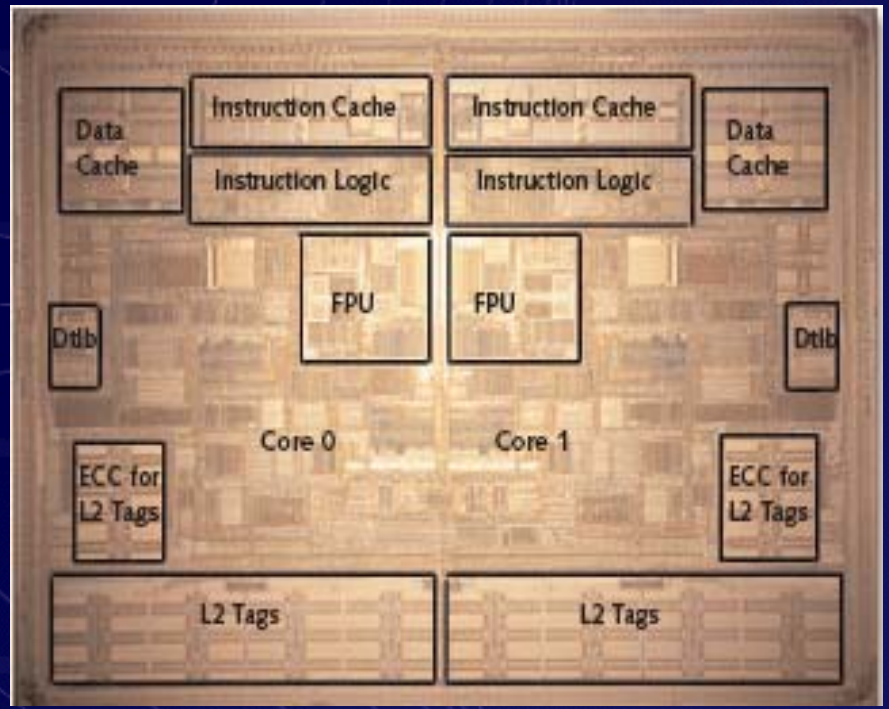


Multimedia = SIMD

- Pour augmenter la performance des applications multimédia, de nouvelles instructions sont apparues dans les jeux d'instructions.
  - Intel : MMX, SSE, SSE2, SSE3
  - Jeux, graphiques 3D, audio, video, reconnaissance de la parole…
- Alimenter les cartes graphiques !
- De plus en plus, les cartes graphiques sont capables de la même chose… voire mieux !

Multi-coeurs

- Plusieurs processeurs sur une puce
- Partage du cache L2
- Exemples :
  - UltraSparc 4 = 2 ultrasparc 3
    - Cache L2 externe partagé
  - Prochain Itanium (montecito)
    - 2 cœurs, L3: 24 MO !
    - 1,7 milliard de transistors
  - Bench (IBM)
    - 8 cœurs
  - P4
    - ?



Pour aller plus loin…

- La base (mais que la base)
  - Hennessy et Paterson – Architecture des ordinateurs, une approche quantitative
    - Attention : il y a trois éditions assez différentes. La première rentre plus dans les détails de l'exécution non ordonnée mais n'est plus très à jour pour d'autres choses.
- J. Silc, B. Robic, Th. Ungerer: Processor Architecture From Dataflow to Superscalar and Beyond. Springer-Verlag, Berlin, Heidelberg, New York 1999
- Site web d'Intel : slides d'exposés d'Intel et leur revue : Intel Technology journal
- IBM Journal on Research and Development (en ligne chez ibm.com)
- Les grandes conférences du domaine:
  - International Symposium on Computer Architecture
  - International Symposium on Microarchitecture
  - High Performance Computer Architecture
  - A venir : conférence et revue du réseau HiPEAC (hipeac.net)
- Les infos les plus récentes sur les processeurs commercialisés
  - Microprocessor Report mais attention au porte-monnaie !
  - des sites web : xbit-labs, intel secrets, Computer Architecture Home Page…