(19) RÉPUBLIQUE FRANÇAISE

INSTITUT NATIONAL
DE LA PROPRIÉTÉ INDUSTRIELLE

PARIS

(11) **N° de publication :** **2 805 363**
(à n'utiliser que pour les commandes de reproduction)

(21) **N° d'enregistrement national :** **00 12788**

(51) Int Cl[7] : **G 06 F 9/30,** G 06 F 12/06, G 11 C 8/00

(12) DEMANDE DE BREVET D'INVENTION A1

(22) **Date de dépôt :** 06.10.00.

(30) **Priorité :** 22.02.00 US 09510010.

(43) **Date de mise à la disposition du public de la demande :** 24.08.01 Bulletin 01/34.

(56) **Liste des documents cités dans le rapport de recherche préliminaire :** *Ce dernier n'a pas été établi à la date de publication de la demande.*

(60) **Références à d'autres documents nationaux apparentés :**

(71) **Demandeur(s) :** *HEWLETT PACKARD COMPANY* — US et *INTEL CORPORATION* — US.

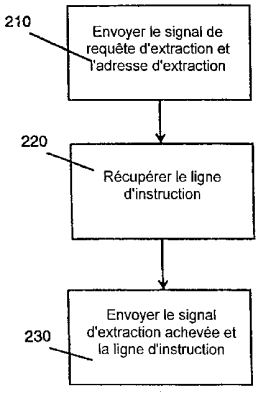
(72) **Inventeur(s) :** DUA ANNUJ, MCCORMICK JR JAMES E, UNDY STEPHEN, ARNOLD BARRY J, BROCKMANN RUSSEL C, KUBICEK DAVID CARL et STOUT JAMES CURTIS.

(73) **Titulaire(s) :**

(74) **Mandataire(s) :** REGIMBEAU.

(54) PROCEDE ET APPAREIL POUR EXTRAIRE DES INSTRUCTIONS DU SOUS-SYSTEME DE MEMOIRE D'UN PROCESSEUR A ARCHITECTURE MIXTE DANS UN MOTEUR D'EMULATION DE MATERIEL.

(57) Un procédé et un appareil pour connecter un processeur capable de traiter des instructions de plusieurs types de jeu d'instructions. Plus particulièrement, un moteur (40) sensible à des instructions natives d'extraction d'un sous-système de mémoire (20) (tel qu'un moteur d'extraction EM (40)) est connecté à un moteur (30) qui traite des instructions émulées (tels qu'un moteur x86 (30)). Ceci est réalisé en utilisant un protocole à échange d'informations, de sorte que le moteur x86 (30) envoie un signal de requête d'extraction explicite (110) au moteur d'extraction EM (40) avec une adresse d'extraction (120). Le moteur d'extraction EM (40) accède ensuite au sous-système de mémoire (20) et récupère une ligne d'instruction (150) pour décodage et exécution ultérieurs. Le moteur d'extraction EM (40) envoie cette ligne d'instruction (150) au moteur x86 (30) avec un signal d'achèvement d'extraction explicite (140).



FR 2 805 363 - A1

Un procédé et un appareil pour extraire des instructions du sous-système de mémoire d'un processeur à architecture mixte dans un moteur d'émulation de matériel.

Le domaine technique concerne des systèmes informatiques numériques et des instructions d'extraction. Plus particulièrement, il concerne des procédés et un appareil pour extraire des instructions d'une mémoire d'ordinateur dans une architecture mixte.

Dans le domaine de l'architecture informatique, une puce unique peut traiter des instructions provenant de jeux d'instructions multiples. Dans ces architectures mixtes, le matériel du processeur est conçu et optimisé pour exécuter les instructions d'un jeu d'instructions, appelé généralement le jeu d'instructions natives, tout en émulant d'autres jeux d'instructions en traduisant les instructions émulées en opérations comprises par le matériel natif. Par exemple, l'architecture IA-64 supporte deux jeux d'instructions - le jeu d'instructions de longueur variable IA-32 (ou x86) et le jeu d'instructions de mode avancé (EM) de longueur fixe. Pendant l'exécution du jeu d'instructions IA-32, l'unité centrale est dite se trouver dans le mode IA-32. Pendant l'exécution des instructions EM, l'unité centrale est dite se trouver dans le mode EM. Les instructions natives EM sont exécutées par le matériel d'exécution principal de l'unité centrale dans le mode EM. Cependant, les instructions IA-32 de longueur variable sont traitées par le moteur IA-32 (ou x86) et décomposées en instructions natives de mode EM pour être exécutées dans le pipeline de noyau de la machine. Dans le mode x86, il est souhaitable de récupérer les instructions du sous-système de mémoire IA-64 dans un moteur x86. Afin de réaliser cela, le moteur d'exécution x86 doit être connecté au pipeline EM, parce que le sous-système de mémoire est étroitement couplé au pipeline EM. Le support matériel x86 existe principalement pour supporter un héritage logiciel. Pour cette raison, il est souhaitable que le moteur x86 ne ralentisse pas le traitement des instructions natives dans le pipeline EM.

Les procédés existants d'extraction d'instructions, tels que les procédés mis en œuvre auparavant dans l'architecture IA-64, utilisent deux pipelines - un pipeline EM et un pipeline x86 - pour traiter les instructions. Dans ces procédés, le moteur x86 envoie simplement une adresse d'extraction au

moteur d'extraction EM qui accède au sous-système de mémoire et renvoie une ligne d'instruction qui est déposée dans une file d'attente de macro-instructions (MIQ) du moteur x86. Bien que les deux pipelines soient synchronisés pour traiter le même ensemble d'adresses, ils fonctionnent indépendamment de sorte que le moteur x86 envoie une nouvelle adresse d'extraction à chaque cycle d'horloge et que le moteur d'extraction EM récupère une nouvelle ligne d'instruction à chaque cycle d'horloge.

Lors de blocages de pipeline (du fait, par exemple, d'une absence d'informations dans la mémoire cache), les pipelines peuvent ne plus être synchronisés. Cela est dû au fait que la transmission des informations entre ces pipelines nécessite un cycle d'horloge complet, compte tenu de la séparation physique entre le moteur x86 et le moteur d'extraction EM. Dans le cas d'un blocage, il n'est pas possible de signaler le blocage au moteur x86 dans le même cycle que celui au cours duquel le moteur d'extraction le voit. C'est-à-dire que le moteur x86 ne remarque le blocage dans le pipeline EM qu'au moins un cycle d'horloge après son apparition. Pendant ce temps, le pipeline x86 continue d'amener l'adresse d'extraction comme si aucun blocage ne s'était produit. Le pipeline x86 et le pipeline EM se désynchronisent et traitent des instructions différentes dans des étages de pipeline correspondants. Il faut des moyens de récupération de blocage compliqués pour de nouveau synchroniser les pipelines.

Un autre problème lié au blocage avec les procédés de traitement d'instruction existants est qu'il peut ne pas y avoir suffisamment de place pour écrire une ligne d'instruction renvoyée dans la file d'attente de macro-instructions. C'est-à-dire que les procédés et les dispositifs existants peuvent tenter d'écrire une nouvelle ligne d'instruction dans la file d'attente de macro-instructions bien que la file d'attente de macro-instructions soit remplie d'entrées non traitées. Un procédé de l'art antérieur introduit un nouveau blocage pour récupérer après ce dépassement de capacité de la file d'attente de macro-instructions. La détection et la signalisation de ce nouveau blocage sont lourdes et, combinées aux blocages précédents liés à l'extraction, nécessitent un matériel de gestion compliqué.

Des moyens sont nécessaires pour réaliser l'interface avec le matériel d'une unité centrale qui traite à la fois des instructions natives et des instructions émulées. En particulier, un procédé est nécessaire pour récupérer les instructions d'une architecture à jeu d'instructions de la mémoire d'une architecture à jeu d'instructions natives différentes tout en évitant les problèmes liés aux blocages de pipeline et les complexités inhérentes au système de pipeline synchrone double.

Un procédé d'interfaçage du matériel dans un processeur capable de mettre en œuvre plusieurs jeux d'instructions, par exemple un jeu d'instructions natives et un jeu d'instructions émulées, est décrit. En particulier, un moteur responsable de l'extraction des instructions natives d'un sous-système de mémoire est connecté à un moteur qui traite les instructions émulées. Cela est réalisé en utilisant un protocole à échange d'informations, conçu de sorte que le moteur x86 envoie un signal de requête d'extraction explicite au moteur d'extraction EM avec une adresse d'extraction. Le moteur d'extraction EM accède ensuite au sous-système de mémoire et récupère une ligne d'instruction afin de la décoder et de l'exécuter ultérieurement. Le moteur d'extraction EM envoie cette ligne d'instruction au moteur x86 avec un signal d'achèvement d'extraction explicite. Le moteur d'extraction EM comprend également une file d'attente d'adresses d'extraction capable de contenir les adresses d'extraction avant qu'elles ne soient traitées par le moteur d'extraction EM. Les requêtes d'extraction sont traitées de telle sorte que plusieurs requêtes d'extraction puissent être en attente au même instant. Si une requête d'extraction en attente est annulée du fait d'un vidage du pipeline, alors la file d'attente d'adresses d'extraction est réinitialisée et les requêtes d'extraction en attente sont annulées. Le système empêche également les blocages liés aux macro-instructions en utilisant un pointeur d'écriture spéculatif qui contrôle l'émission de requêtes d'extraction, empêchant de ce fait un dépassement de capacité de la file d'attente de macro-instructions.

Un système informatique capable de traiter les instructions de plusieurs jeux d'instructions et un moteur qui extrait les instructions natives d'un sous-système de mémoire (tel qu'un moteur d'extraction EM) et un moteur qui traite les instructions émulées (tel qu'un moteur x86) sont décrits. Le moteur d'extraction EM comporte une file d'attente d'adresses d'extraction. L'inter-

face entre le moteur d'extraction EM, le sous-système de mémoire et le moteur x86 utilise un protocole à échange d'informations. Le moteur x86 envoie un signal de requête d'extraction explicite au moteur d'extraction EM avec une adresse d'extraction. Le moteur d'extraction EM accède ensuite au sous-système de mémoire et récupère une ligne d'instruction. Le moteur d'extraction EM envoie cette ligne d'instruction au moteur x86 avec un signal d'achèvement d'extraction explicite. Le moteur d'extraction EM comprend également une file d'attente d'adresses d'extraction capable de contenir les adresses d'extraction avant qu'elles ne soient traitées par le moteur d'extraction EM. Les requêtes d'extraction sont traitées de sorte que plusieurs requêtes d'extraction puissent être en attente au même instant. Si une requête d'extraction en attente est annulée du fait d'un vidage du pipeline, alors la file d'attente d'adresses d'extraction est mise à zéro et les requêtes d'extraction en attente sont annulées. Le système empêche également les blocages liés aux macro-instructions par l'utilisation d'un pointeur d'écriture spéculatif pour contrôler l'émission d'instructions d'extraction, empêchant, de ce fait, un dépassement de la capacité de la file d'attente de macro-instructions.

- La Figure 1 est un organigramme montrant un procédé d'extraction d'instructions.
- La Figure 2 est un schéma fonctionnel montrant le fonctionnement du système.
- La Figure 3 est un schéma fonctionnel plus détaillé montrant le pipeline.
- La Figure 4A est un graphique montrant la progression des requêtes d'extraction à travers le pipeline EM.
- La Figure 4B est un graphique montrant la progression des requêtes d'extraction à travers le pipeline x86.
- La Figure 5A est un graphique montrant la progression des requêtes d'extraction à travers le pipeline EM des procédés existants.
- La Figure 5B est un graphique montrant la progression des requêtes d'extraction telle que suivie par le pipeline x86 des procédés existants.
- La Figure 6 est un schéma fonctionnel de la file d'attente de macro-instructions.

Le système améliore l'interfaçage entre le matériel d'un processeur qui met en œuvre à la fois un jeu d'instructions natives et un jeu d'instructions émulées en remplaçant le mécanisme à blocage contrôlé synchrone par un protocole d'extraction à échange d'informations. Un spécialiste de l'art reconnaîtra que le système peut être utilisé par tout moteur qui tente d'émuler une architecture à jeu d'instructions en utilisant une autre architecture à jeu d'instructions. À titre d'illustration uniquement et non à titre de limitation, le mode de réalisation du système est montré comme une interface entre un moteur x86 30 et un sous-système de mémoire IA-64 20. Le sous-système de mémoire 20 comprend tout appareil qui peut être utilisé pour mémoriser des octets d'instruction, y compris un système de mémoire cache, une mémoire principale et toute autre mémoire utilisée par le système.

La Figure 1 montre un organigramme du procédé du système. Lorsque le moteur x86 30 a besoin de nouvelles instructions à traiter, il envoie, en 210, un "signal de requête d'extraction" 110 au moteur d'extraction 40. Le moteur x86 envoie également, en 210, une "adresse d'extraction" 120 au moteur d'extraction 40. L'adresse d'extraction 120 est l'information utilisée par le moteur d'extraction 40 pour récupérer une ligne d'instruction 130 du sous-système de mémoire 20. Dans un mode de réalisation, le signal de requête d'extraction 110 est envoyé en même temps que l'adresse d'extraction 120. Le moteur d'extraction 40 accède au sous-système de mémoire 20 et récupère, en 220, une ligne d'instruction 130. À moins qu'une requête d'extraction en attente 120 ne soit annulée du fait d'un vidage de pipeline, le moteur d'extraction 40 renvoie alors, en 230, une indication "extraction achevée" 140 au moteur x86 30. L'indication d'achèvement d'extraction 140 termine la procédure d'échange d'informations entre le moteur d'extraction 40 et le moteur x86 30, en indiquant que la requête d'extraction a été exécutée avec succès. Avec l'indication d'achèvement d'extraction 140, le moteur d'extraction 40 envoie également, en 230, une ligne d'instruction 150 au moteur x86 30.

La Figure 2 montre un système informatique 10, comportant un sous-système de mémoire 20, un moteur x86 30 et une extrémité avant du pipeline de noyau EM, appelé moteur d'extraction EM 40. Une file d'attente d'adresses d'extraction 50 se trouve dans le moteur d'extraction 40. Le moteur d'extraction 40 réalise l'interface avec le sous-système de mémoire 20

pour extraire des instructions (des instructions EM ou IA-32) afin de les exécuter. Dans le mode IA-32, le moteur d'extraction 40 délivre des instructions IA-32 au moteur x86 30 pour un traitement supplémentaire.

Le procédé à échange d'informations expliqué ci-dessus et sur la Figure 1 est montré mis en œuvre dans le système 10. Le signal de requête d'extraction 110 est envoyé, en 210, du moteur x86 30 au moteur d'extraction 40. Au même instant, une adresse d'extraction 120 est également envoyée, en 210, du moteur x86 30 au moteur d'extraction 40. Le moteur d'extraction 40 accède ensuite au sous-système de mémoire 20 et récupère, en 220, une ligne d'instruction 130. À moins qu'une requête d'extraction en attente ne soit annulée du fait d'un vidage du pipeline, le moteur d'extraction 40 envoie ensuite, en 230, un signal d'achèvement d'extraction 140 au moteur x86 30, indiquant que la requête d'extraction a été exécutée avec succès. Avec l'indication d'achèvement d'extraction 140, le moteur d'extraction 40 envoie, en 230, la ligne d'instruction 150 au moteur x86 30.

Dans un mode de réalisation, 16 octets séquentiels d'instructions sont demandés par le moteur d'extraction 40 en envoyant un signal de requête d'extraction 110. Dans ce mode de réalisation, le moteur x86 30 envoie une adresse d'extraction 120 de 28 bits alignée sur 16 octets au moteur d'extraction 40 et envoie, au même instant, le signal de requête d'extraction 110. Dans ce mode de réalisation, le moteur d'extraction 40 accède, en 220, au sous-système de mémoire 20 et renvoie, en 230, les 16 octets de données d'instruction 150 au moteur x86 30 avec une indication "extraction achevée" 140.

La Figure 3 montre un schéma plus détaillé d'un mode de réalisation du système informatique 10. Le moteur d'extraction natif 40 (dans cet exemple, un moteur d'extraction EM) et le moteur d'émulation 30 (dans cet exemple, un moteur x86) peuvent être imaginés comme deux pipelines parallèles - c'est-à-dire, le pipeline EM 40 et le pipeline x86 30. Le moteur d'extraction 40 extrait des lignes d'instructions 130 du sous-système de mémoire 20 du processeur IA-64 pour qu'elles soient exécutées dans le moteur x86 30. À cette fin, le moteur x86 30 demande une ligne d'instruction au moteur d'extraction (pipeline EM) 40. Dans le mode de réalisation montré, le pipeline EM 40 comporte trois étages - IIP 320, IPG 330 et ROT 340. Le pipeline

EM 40 utilise également une file d'attente d'adresses d'extraction 50 qui se trouve dans l'étage IIP 320. La file d'attente d'adresses d'extraction 50 est utilisée pour mémoriser les adresses d'extraction 120 envoyées par le moteur x86 30 lorsque le pipeline EM 40 est bloqué. La file d'attente d'adresses d'extraction peut ainsi mémoriser les requêtes d'extraction lorsqu'un accès au sous-système de mémoire est effectué pour une ligne particulière d'instructions. Dans le mode de réalisation montré, la file d'attente d'adresses d'extraction 50 peut contenir jusqu'à trois adresses d'extraction distinctes 120 alors qu'elles sont traitées par le pipeline EM 40. Le pipeline x86 30 comporte trois étages correspondants - BT1 350, BT2 360, BT3 370. Le pipeline x86 30 comporte également sa propre file d'attente d'adresses x86 60 qui recouvre les étages respectifs du pipeline x86 30. Les entrées de la file d'attente d'adresses x86 60 correspondent aux entrées de la file d'attente d'adresses d'extraction 50 du pipeline EM 40.

Un groupe de fonctions logiques, appelé logique d'avance du pipeline 90, est appliqué aux adresses 120 dans les étages respectifs de la file d'attente d'adresses x86 60 pour assurer l'avance des adresses d'extraction 120 le long du pipeline x86 30. La logique d'avance du pipeline 90 est conçue pour déplacer l'adresse d'extraction en suspens la plus ancienne 120 vers l'étage BT3 370. Une fois que chaque étage du pipeline x86, BT1 350, BT2 360, BT3 370, contient des adresses valides 120 correspondant aux trois requêtes d'extraction en attente dans le pipeline EM 30, les adresses 120 sont avancées à chaque fois qu'une ligne d'instruction 150 est renvoyée pour la requête en suspens la plus ancienne 120. Pour effectuer cela, la logique d'avance de pipeline 90 utilise une version retardée de l'indication d'achèvement d'extraction 142 pour faire avancer les adresses 120 le long du pipeline x86 30. Comme montré, l'indication d'achèvement d'extraction 140 est envoyée de l'étage BT3 370 à une bascule 72 dans l'étage d'alignement (ALN) 380 du pipeline x86 30.

La sortie de la bascule 72 est le signal d'achèvement d'extraction retardé 142 qui est ensuite utilisé par la logique d'avance du pipeline 90 et qui contrôle les signaux de requête d'extraction 110. À chaque étage 320, 330, 340 du pipeline EM 40, des fonctions logiques EM 80 agissent pour extraire une ligne d'instruction 150 du sous-système de mémoire 20. Lorsqu'une ligne

d'instruction 150 est renvoyée au moteur x86 30, elle correspond à l'adresse 120 de l'étage BT3 370, qui représente la requête d'extraction non satisfaite la plus ancienne. Les informations d'instruction sont mémorisées et placées dans une file d'attente de macro-instructions 70, un cycle d'horloge plus tard que dans l'étage ALN 380 du moteur x86 30.

On se rendra compte que plusieurs requêtes d'extraction peuvent être en attente ou "en cours" au même instant. Les Figures 4A et 4B montrent la progression des requêtes d'extraction 120 à travers les étages respectifs du pipeline EM 40 et du pipeline x86 30 pour des cycles d'horloge successifs. Le côté gauche des Figures 4A et 4B montre des cycles d'horloge successifs, tels que t, t+1, t+2, t+3, t+4, ... t+n. Le haut des Figures 4A et 4B montre trois étages 320, 330, 340 du pipeline EM 40 et trois étages 350, 360, 370 du pipeline x86 30, respectivement. La Figure 4A montre, pour des cycles d'horloge successifs, l'emplacement des requêtes d'extraction 120 dans les étages respectifs 320, 330, 340 du pipeline EM 40. La Figure 4B montre l'emplacement correspondant des adresses d'extraction 120 dans les trois étages respectives 350, 360, 370 de la file d'attente d'adresses x86 60. De plus, un étage de pipeline ALN 380 suit l'étage de pipeline BT3 370 du moteur x86 30. La file d'attente de macro-instructions 70 est écrite dans cette étage 380. Des exemples d'adresses d'extraction 120 émises successivement sont représentés par les lettres "A, B, C, D, etc." Certains moments (par exemple, 511 à 544 sur les Figures 4A, 4B, 5A, et 5B) sont indiqués et correspondent aux points-clés exposés ci-dessous, qui résument les opérations à certains moments.

Par exemple, au premier cycle d'horloge t, un signal de requête d'extraction 110_A est envoyé pour la première adresse d'extraction A. Une adresse d'extraction A se trouve dans l'étage IIP 320 du pipeline EM 40, où le pipeline EM 40 reçoit l'adresse d'extraction 120 avec le signal de requête d'extraction 110. Au second cycle d'horloge t+1, un signal de requête d'extraction 110_B est envoyé pour la seconde adresse d'extraction B, tandis que le sous-système de mémoire 20 est préparé pour l'extraction de la première adresse A. A se trouve dans l'étage IPG 330 et B se trouve dans l'étage IIP 320. Au troisième cycle d'horloge t+2, 511, un signal de requête d'extraction 110_C est envoyé pour la troisième adresse d'extraction C et le

sous-système de mémoire 20 est préparé pour l'extraction de la seconde adresse B. Pour la première adresse d'extraction A, un accès au sous-système de mémoire 20 est effectué, la ligne d'instruction 130 est reçue par le moteur d'extraction 40 et la ligne d'instruction 150 est envoyée au moteur x86 30. Dans le pipeline EM 40, A se trouve dans l'étage ROT 340, B se trouve dans l'étage IPG 330 et C se trouve dans l'étage IIP 320. Dans le cycle d'horloge suivant t+3, 512, la ligne d'octets d'instruction 150 pour l'adresse A est écrite dans la file d'attente de macro-instructions 70 dans l'étage ALN 380 du pipeline x86 30, comme montré sur la Figure 4B. En l'absence de blocage de pipeline, cette progression continue avec une nouvelle requête et une nouvelle ligne d'instruction écrite dans la file d'attente de macro-instructions 70.

Dans l'exemple montré sur les Figures 4A et 4B, le moteur d'extraction 40 est incapable d'exécuter la requête d'extraction correspondant à l'adresse B dans le cycle t+3, 512, 522, du fait d'une absence d'informations dans la mémoire cache. À cause de cela, aucune indication d'achèvement d'extraction 140 n'est renvoyée au moteur x86 30 et le pipeline EM 40 se bloque à l'instant t+3, 512. C'est-à-dire que l'adresse D est bloquée dans l'étage IIP 320, l'adresse C est bloquée dans l'étage IPG 330 et l'adresse B est bloquée dans l'étage ROT 340. Cette situation continue jusqu'à ce que la machine ait pu interroger tous les niveaux de hiérarchie de mémoire 20 nécessaires pour récupérer la ligne d'instruction 130 correspondant à l'adresse B. De manière significative, les entrées dans la file d'attente d'adresses x86 60 reflètent l'emplacement des adresses 120 dans le pipeline EM 40 aussi longtemps que le blocage persiste. C'est-à-dire que l'adresse D reste bloquée dans l'étage BT1 340, l'adresse C reste bloquée dans l'étage BT2 350 et l'adresse B reste bloquée dans l'étage BT3 360. Cela est dû au fait que la logique d'avance du pipeline 90 détermine qu'il y a actuellement trois requêtes d'extraction non satisfaites dans le pipeline EM 40 (pour les adresses D, C et B) et désactive l'avance d'adresses 120 dans le pipeline x86 30. De plus, les nouvelles requêtes d'extraction sont supprimées. Lorsque la ligne d'instruction 150 est finalement renvoyée pour l'adresse B (montré dans un cycle d'horloge t+4 513, 523 à des fins de simplicité sur les Figures 4A et 4B), le moteur d'extraction EM 40 envoie la ligne d'instruction au moteur x86 30 avec l'indication d'achèvement d'extraction 140. Une ligne d'instruction pour

l'adresse B est ensuite écrite dans la file d'attente de macro-instructions 70 pendant le cycle t+5, 524. Au même instant, la logique d'avance du pipeline x86 90 échantillonne la version retardée de l'indication d'achèvement d'extraction 142 (qui avait été envoyée pour l'adresse B dans le cycle t+4) et fait avancer les adresses le long du pipeline x86 30. Une nouvelle requête d'extraction est émise pour l'adresse E et la progression continue. Pour l'exemple montré sur les Figures 4A et 4B, le blocage est montré comme ne durant qu'un cycle d'horloge. Dans la pratique, il peut être plus long, il est fonction du temps nécessaire pour exécuter la requête d'extraction.

Le point-clé suivant résume la progression dans le pipeline EM de la Figure 4A à certains moments :

511 - l'indication d'achèvement d'extraction 140 et la ligne d'instruction 150 sont renvoyées pour l'adresse A.

512 - l'indication d'achèvement d'extraction 140 n'est pas renvoyée pour l'adresse B.

- le pipeline EM 40 se bloque.

513 - l'indication d'achèvement d'extraction 140 et la ligne d'instruction 150 sont renvoyées pour l'adresse B

- le blocage du pipeline EM est supprimé.

Un point-clé correspondant résume ces points du pipeline x86 30 de la Figure 4B à certains moments :

521 - l'indication d'achèvement d'extraction 140 et la ligne d'instruction 150 sont renvoyées pour l'adresse A.

522 - la ligne d'instruction 150 pour l'adresse A est écrite dans la file d'attente de macro-instructions 70.

- l'indication d'achèvement d'extraction 140 n'est pas renvoyée pour l'adresse B.

- le pipeline x86 30 se bloque.

523 - l'indication d'achèvement d'extraction 140 et la ligne d'instruction 150 sont renvoyées pour l'adresse B.

524 - la ligne d'instruction pour l'adresse B est écrite dans la file d'attente de macro-instructions 70.

- le pipeline x86 30 avance.

La progression des instructions par le procédé, telle que montrée sur les Figures 4A et 4B, contraste avec les problèmes rencontrés par les procédés existants. Les Figures 5A et 5B illustrent le problème de blocage du pipeline dans les procédés de fonctionnement existants en suivant les entrées dans les étages du moteur d'extraction 40 et dans le moteur x86 30. Les procédés existants n'utilisent pas de mémoire tampon, telle que la file d'attente d'adresses d'extraction 50, de sorte que les Figures 5A et 5B font simplement référence aux étages dans lesquels le moteur d'extraction 40 traite les requêtes et les étages dans lesquels le moteur x86 30 suppose que les requêtes sont traitées. C'est-à-dire que le moteur x86 30 tente de suivre la progression des instructions à travers les étages du moteur d'extraction 40, mais n'y arrive pas toujours.

La Figure 5A montre la progression des adresses d'extraction 120 à travers les divers étages 320, 330, 340 du moteur d'extraction 40. La Figure 5B montre la progression de ces mêmes adresses d'extraction 120 à travers les étages respectifs 340, 350, 360 du moteur x86 30. De plus, un étage ALN du pipeline 380 suit l'étage de pipeline BT3 370 du moteur x86 30. La file d'attente de macro-instructions 70 est écrite dans cet étage 380. Le traitement des instructions est effectué dans le pipeline EM 40, mais le moteur x86 30 tente de suivre la progression des requêtes d'extraction 120 à travers les étages du moteur d'extraction 40. Tant qu'aucun blocage ne survient, le moteur x86 30 suit avec succès la progression des instructions, comme montré pendant les instants t, t+1, et t+2 des Figures 5A et 5B.

Cependant, les Figures 5A et 5B montrent de nouveau l'effet d'un blocage à l'instant t+3 532, 542. Pendant le cycle suivant t+4 533, une requête d'extraction B reste dans l'étage ROT 340, une requête d'extraction C reste dans l'étage IPG 330 et une requête d'extraction D reste dans l'étage IIP 320 du moteur d'extraction 40, comme montré sur la Figure 5A. Cependant, le moteur x86 30 n'a pas connaissance du blocage pendant le cycle où il survient, ainsi il fait avancer les adresses d'extraction 120 dans le pipeline x86, comme montré à l'instant t+4 543 sur la Figure 5B. À l'instant t+4 543, le moteur x86 30 a connaissance du blocage et ainsi répète son entrée à l'instant t+5 544.

Sur la Figure 5A, le blocage est montré comme étant levé à l'instant t+4 533 dans le pipeline EM 40 et le moteur d'extraction 40 fait avancer l'adresse d'extraction 120 d'un étage. De nouveau, cependant, le moteur x86 30 n'a pas connaissance immédiatement de la suppression du blocage. Il n'a connaissance de la suppression du blocage qu'à l'instant t+5 544, c'est-à-dire lorsque finalement il retrouve la progression vraie des instructions dans le moteur d'extraction 40, comme montré sur la Figure 5B. Cela est différent du système décrit dans le présent document qui utilise une file d'attente d'adresses d'extraction 50 dans le moteur d'extraction 40 et une file d'attente d'adresses x86 60 dans le moteur x86 30 et qui garantit que le moteur x86 30 suit avec précision la progression des requêtes d'extraction 120 à travers le moteur d'extraction 40.

Le point-clé suivant résume certains événements et illustre les problèmes du pipeline EM dans les procédés existants à certains instants :

- la ligne d'instruction 150 pour l'adresse A est renvoyée.

532 - le pipeline EM 40 se bloque (par exemple, du fait d'une absence d'informations dans la mémoire cache) pour l'adresse B.

533 - le blocage du pipeline EM est levé.

- la ligne d'instruction 150 pour l'adresse B est renvoyée.

Un point-clé correspondant résume le fonctionnement du pipeline x86 à certains instants :

541 - la ligne d'instruction 150 pour l'adresse A est écrite dans une file d'attente de macro-instructions 70.

- le blocage du pipeline EM n'est pas perçu par le pipeline x86 30.

542 - le blocage du pipeline EM est perçu par le pipeline x86 30.

- le pipeline x86 30 n'est plus synchronisé avec le pipeline EM 40.
- une mémoire tampon de récupération pour l'adresse B est nécessaire.

543 - la suppression du blocage du pipeline EM est perçue par le pipeline x86 30.

- le pipeline x86 30 n'avance pas.

544 - la suppression du blocage du pipeline EM est perçue par le pipeline x86 30.

- une ligne d'instruction 150 ppour l'adresse B est écrite dans la file d'attente de macro-instructions 70.

Le protocole à échange d'informations, décrit ci-dessus, est conçu pour alléger les mécanismes de récupération complexes qui sont nécessaires dans les procédés précédents pour maintenir deux pipelines indépendants qui sont connectés en synchronisation. La procédure d'échange d'informations est, par définition, indépendante du temps entre l'envoi d'une requête d'extraction et son exécution. Cela rend la mise en œuvre assez directe et relativement facile à vérifier.

Le moteur x86 30 envoie jusqu'à 3 requêtes d'extraction 120 avant d'attendre la ligne d'instruction pour la requête correspondant à la première requête d'extraction 120_1. En l'absence de blocage lié à l'extraction d'extrémité avant (par exemple, du fait de l'absence d'informations dans la mémoire cache ou d'un échec de TLB), les données pour la requête de la première adresse 120_1 sont renvoyées pendant le cycle où la requête d'extraction pour la troisième adresse d'extraction 120_3 est émise. Ainsi, de nouvelles requêtes peuvent encore être introduites dans le pipeline et une nouvelle requête d'extraction est effectuée à chaque cycle d'horloge. Dans le cas d'un blocage d'extraction d'extrémité avant, l'indication d'achèvement d'extraction 140 ne sera pas affirmée tant que la condition de blocage ne sera pas levée et que les données ne seront pas disponibles. Le moteur d'extraction 40 peut mémoriser jusqu'à trois adresses d'extraction dans la file d'attente d'adresses d'extraction 50 et traiter les données correspondantes d'une manière premier-entré-premier-sorti (FIFO). Ainsi, une requête d'extraction pour la quatrième adresse d'extraction 120_4 ne sera pas exécutée par le moteur x86 30 tant qu'une indication d'achèvement d'extraction 140 ne sera pas connue comme ayant été reçue de la requête en suspens la plus ancienne dans le cycle précédent. En mémorisant jusqu'à trois requêtes en attente, la logique de requête d'extraction peut utiliser une version retardée d'un cycle d'horloge de l'indication d'achèvement d'extraction 142 provenant du moteur d'extraction 40 pour lancer la nouvelle requête. Cela allège le pression de la synchronisation sur le signal d'achèvement d'extraction 140 venant du moteur d'extraction 40 tout en maintenant encore les deux pipelines 30, 40 synchronisés. Dans le cas d'un vidage du pipeline, les files d'attente d'extraction 50 sont vidées et toutes les requêtes d'extraction 120 en cours ou en attente sont annulées. Parce que les pipelines 30, 40 sont synchronisés, il n'est pas

nécessaire de vider de manière sélective les différents étages du pipeline comme cela était nécessaire pour les mises en œuvre précédentes. De plus, parce que le moteur x86 30 est conçu pour garantir qu'il n'y a jamais plus de trois adresses d'extraction 120 en cours ou en attente (c'est-à-dire, demandées mais pas encore renvoyées) à un instant quelconque donné, la file d'attente d'adresses d'extraction 50 n'est jamais en dépassement.

La file d'attente de macro-instructions 70 montrée dans le mode de réalisation de la Figure 3 est une file d'attente à découplage. C'est-à-dire que l'écriture des lignes d'instructions 150 dans la file d'attente de macro-instructions 70 est effectuée à un débit différent de leur lecture. Pour cette raison, il est nécessaire de s'assurer que la file d'attente de macro-instructions 70 n'est pas en dépassement. La Figure 6 montre un schéma fonctionnel d'une file d'attente de macro-instructions 70. Afin d'éviter que la file d'attente de macro-instructions 70 soit en dépassement, un premier mode de réalisation utilise quatre pointeurs distincts 410, 420, 430, 440 vers la file d'attente de macro-instructions 70 pour contrôler le flot d'informations vers la file d'attente de macro-instructions 70. Dans le mode de réalisation montré, la file d'attente de macro-instructions 70 comporte 8 entrées d'une largeur de 16 octets. Un pointeur d'écriture 410 identifie l'entrée où la ligne d'instruction est écrite et avance vers l'entrée suivante à chaque écriture. Un pointeur de lecture 420 identifie l'entrée à lire dans la file d'attente de macro-instructions 70 par le moteur x86 30. Les 16 octets de données des entrées de la file d'attente de macro-instructions 70 peuvent contenir plusieurs instructions. Le moteur x86 30 pourrait ne lire qu'une partie d'une entrée à la fois. Par conséquent, le pointeur de lecture 420 n'avance que lorsque les 16 octets de l'entrée sont lus.

L'exécution des instructions dans le moteur x86 30 passe par plusieurs étages avant que l'instruction soit finalement retirée. C'est-à-dire qu'une instruction peut avoir été lue dans la file d'attente de macro-instructions 70, mais rester dans le moteur x86 30 pendant un certain temps avant d'être retirée. Il peut être souhaitable qu'une ré-écriture des entrées de la file d'attente de macro-instructions 70 ne soit pas exécutée tant qu'une instruction n'a pas été retirée du moteur x86 30. Le pointeur de lecture 420 peut avancer avant que l'instruction ait été retirée. Par conséquent, dans un mode de réalisation, la file d'attente de macro-instructions 70 comprend un pointeur

de retrait 430 qui indique qu'une instruction a été retirée par le moteur x86 30 et qu'une écriture peut être effectuée dans cette entrée de manière sure. Le pointeur de retrait 430 sera en retard par rapport au pointeur de lecture 420.

Dans un premier mode de réalisation, le système empêche l'émission de nouvelles requêtes d'extraction 120 par le moteur x86 30 si la file d'attente de macro-instructions 70 est pleine. Ceci est réalisé en comparant les pointeurs de file d'attente de macro-instructions afin de garantir qu'aucune entrée n'est pas ré-écrite avant le temps souhaité. Le pointeur d'écriture 410 ne doit pas écrire dans les entrées qui n'ont pas été lues, comme indiqué par le pointeur de lecture 420. Également, il peut être souhaitable d'empêcher la ré-écriture des entrées qui ont été lues mais non retirées, comme indiqué par le pointeur de retrait 430.

Comme noté, cependant, le système peut comprendre de multiples étages de requêtes en attente (par exemple, IIP, IPG, ROT) et la file d'attente d'adresses d'extraction 50 peut comporter des entrées multiples. Par exemple, le mode de réalisation montré sur la Figure 3 a trois entrées dans la file d'attente d'adresses d'extraction 50 et trois requêtes d'extraction 120 peuvent être en cours ou en attente au même instant. C'est un vrai défi d'empêcher un dépassement de la file d'attente de macro-instructions 70, parce que les pointeurs de file d'attente de macro-instructions doivent tenir compte des requêtes d'extraction en attente 120 afin de garantir qu'il reste suffisamment de place dans la file d'attente de macro-instructions 70 pour gérer les requêtes en attente. Un premier mode de réalisation du système comprend un quatrième pointeur, appelé pointeur d'écriture spéculatif 440, pour garantir que la file d'attente de macro-instructions 70 n'est pas en dépassement du fait des requêtes d'extraction en attente 120. Le pointeur d'écriture spéculatif 440 avance chaque fois qu'une requête d'extraction 120 est émise par le moteur x86 30. De cette manière, il tient compte des requêtes d'extraction en attente 120 et assure une espace suffisant dans la file d'attente de macro-instructions 70 pour l'écriture des lignes d'instruction renvoyées 150. Le pointeur d'écriture spéculatif 440 peut être comparé au pointeur de retrait 430 et garantit qu'aucune requête d'extraction 120 n'est émise avant que le pointeur de retrait 430 ait avancé et laissé une entrée vide dans la file d'attente de macro-instructions 70.

Par exemple, si la file d'attente de macro-instructions 70 a 8 entrées (0 - 7), le pointeur d'écriture 410 peut pointer vers l'entrée 3, en tant qu'entrée où la ligne d'instruction suivante sera écrite. Le pointeur de lecture 420 peut pointer vers l'entrée 2, entrée où la ligne suivante d'instruction sera lue. Le pointeur de retrait 440 peut pointer vers l'entrée 1, entrée de laquelle l'instruction la plus récente a été retirée dans le moteur x86 30. Si trois requêtes d'extraction 120 sont déjà dans le moteur d'extraction 40, alors le pointeur d'écriture spéculatif pointera vers l'entrée 6, laissant de la place dans la file d'attente de macro-instructions 70 pour les lignes d'instruction 130 qui sont renvoyées pour ces requêtes 120. Parce que le pointeur d'écriture spéculatif 440 ne peut pas dépasser le pointeur de retrait 430, le système peut être configuré de sorte qu'aucune requête d'extraction 120 ne soit émise lorsque le pointeur d'écriture spéculatif 440 rattrape le pointeur de retrait 430.

Bien que le système et le procédé aient été décrits en détail avec référence à certains de ses modes de réalisation, des variantes sont possibles. Par exemple, bien que les valeurs de certaines données, les tailles des pipelines, le nombre de requêtes d'extraction en attente, les cycles d'horloge et certaines autres informations spécifiques aient été donnés à titre d'exemples, ces exemples étaient uniquement illustratifs et non limitatifs. Le système et le procédé peuvent être mis en œuvre sous d'autres formes spécifiques sans s'écarter de l'esprit ou des attributs essentiels de ceux-ci. Bien que les exemples présentés fassent référence spécifiquement à l'architecture IA-64 et au moteur d'extraction EM et au moteur x86 en tant que systèmes natif et d'émulation, ceux-ci sont présentés à titre d'illustration uniquement et non à titre de limitation. Le procédé peut être mis en œuvre sur n'importe quel type d'architecture capable d'utiliser plusieurs types d'architecture de jeu d'instructions. Il est souhaitable que le mode de réalisation décrit dans le présent document soit considéré à tous égards comme illustratif, non restrictif et que référence soit faite aux revendications jointes pour déterminer l'étendue de l'invention.

REVENDICATIONS

1. Procédé pour connecter un moteur d'émulation (30) à un moteur d'extraction (40) qui extrait des instructions d'une architecture à jeu d'instructions émulées (ISA) d'un sous-système de mémoire (20), caractérisé en ce qu'il comprend :

— l'envoi d'un signal de requête d'extraction (110) au moteur d'extraction (40) ;
— l'envoi d'une adresse d'extraction (120) au moteur d'extraction (40) ;
— la récupération d'une ligne d'instruction (150) du sous-système de mémoire (20) ;
— l'envoi d'un signal d'achèvement d'extraction (140) par le moteur d'extraction (40) au moteur d'émulation (30) ; et
— l'envoi de la ligne d'instruction au moteur d'émulation (30).

2. Procédé selon la revendication 1, caractérisé en ce qu'il comprend, de plus, la mémorisation de l'adresse d'extraction (120) dans une file d'attente d'adresses d'extraction (50), dans le moteur d'extraction (40), et en ce qu'une pluralité de requêtes d'extraction peuvent être traitées au même instant.

3. Procédé selon les revendications 1 ou 2, caractérisé en ce qu'il comprend, de plus, l'envoi de la ligne d'instruction (150) à une file d'attente de macro-instructions (70), dans le moteur d'émulation (30).

4. Procédé selon les revendications 1, 2 ou 3, caractérisé en ce qu'il comprend, de plus :

— la détermination du fait que la file d'attente de macro-instructions (70) est pleine ou sera remplie par une ou plusieurs lignes d'instruction (150) renvoyées par une ou plusieurs requêtes d'extraction en attente ; et
— dans le cas où la file d'attente de macro-instructions (70) est pleine ou sera remplie par une ou plusieurs lignes d'instruction (150) renvoyées par une ou plusieurs requêtes d'extraction en attente, le fait d'attendre que la file d'attente de macro-instructions (70) ne soit plus pleine et ne puisse pas être remplie, pour envoyer un nouveau signal de requête d'extraction.

5. Procédé selon les revendications 1, 2, 3, 4 ou 5, caractérisé en ce qu'il comprend, de plus, l'utilisation d'un pointeur d'écriture spéculatif (440) pour déterminer si la file d'attente de macro-instructions (70) est pleine ou sera remplie par une ou plusieurs lignes d'instruction (150) renvoyée par une ou plusieurs requêtes d'extraction en attente en comparant un certain nombre de requêtes d'extraction en attente avec un certain nombre d'entrées ouvertes dans la file d'attente de macro-instructions (70).

6. Système informatique à architectures multiples (10) capable de mettre en œuvre une architecture à jeu d'instructions natives et une architecture à jeu d'instructions émulées, caractérisé en ce qu'il comprend :

— un sous-système de mémoire (20) d'une architecture à jeu d'instructions natives ;
— un moteur d'extraction (40) de l'architecture à jeu d'instructions natives, ledit moteur d'extraction (40) étant connecté électriquement au sous-système de mémoire (20) de l'architecture à jeu d'instructions natives ; et
— un moteur (30) d'une architecture à jeu d'instructions émulées, ledit moteur (30) d'une architecture à jeu d'instructions émulées étant connecté électriquement au moteur d'extraction (40) de l'architecture à jeu d'instructions natives ;

caractérisé en ce que le moteur d'extraction (40) de l'architecture à jeu d'instructions natives est capable d'accéder au sous-système de mémoire (20) de l'architecture à jeu d'instructions natives et est connecté au moteur (30) de l'architecture à jeu d'instructions émulées en utilisant un protocole à échange d'informations.

7. Système informatique selon la revendication 6, caractérisé en ce que le moteur (30) de l'architecture à jeu d'instructions émulées demande une ligne d'instruction (150) et en ce que le moteur d'extraction (40) envoie une ligne d'instruction (150) au moteur (30) de l'architecture à jeu d'instructions émulées.

8. Système informatique selon les revendications 6 ou 7, caractérisé en ce que le moteur (30) de l'architecture à jeu d'instructions émulées envoie un signal de requête d'extraction (110) et une adresse d'extraction (120) au moteur d'extraction (40) et en ce que le moteur d'extraction (40) envoie un

signal d'achèvement d'extraction (140) et la ligne d'instruction (150) au moteur (30) de l'architecture à jeu d'instructions émulées.

9. Système informatique selon les revendications 6, 7 ou 8, caractérisé en ce que le moteur d'extraction (40) comprend une file d'attente d'adresses d'extraction (50) capable de mémoriser une adresse d'extraction (120) et en ce que le moteur (30) de l'architecture à jeu d'instructions émulées comprend, de plus, une file d'attente de macro-instructions (70) qui reçoit la ligne d'instruction (150) du moteur d'extraction (40).

10. Système informatique selon les revendications 6, 7, 8 ou 9, caractérisé en ce qu'il comprend, de plus, un pointeur d'écriture spéculatif (440) qui empêche le dépassement de la file d'attente de macro-instructions (70) du fait d'une ou de plusieurs requêtes d'extraction en attente, lequel pointeur d'écriture spéculatif (440) peut être utilisé pour contrôler l'envoi d'une requête d'extraction.

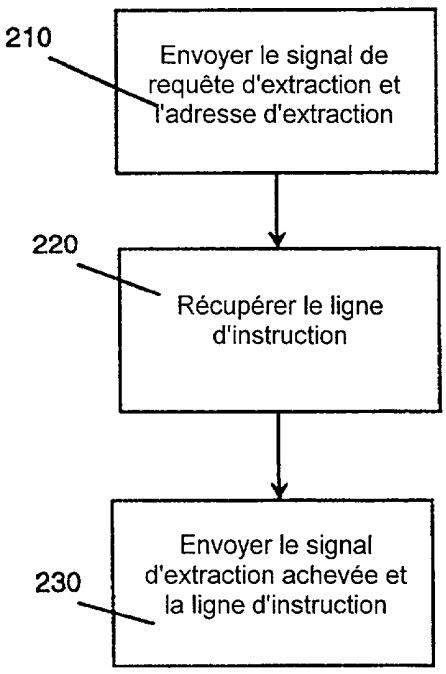

210
Envoyer le signal de requête d'extraction et l'adresse d'extraction
220
Récupérer le ligne d'instruction
230
Envoyer le signal d'extraction achevée et la ligne d'instruction

Fig. 1

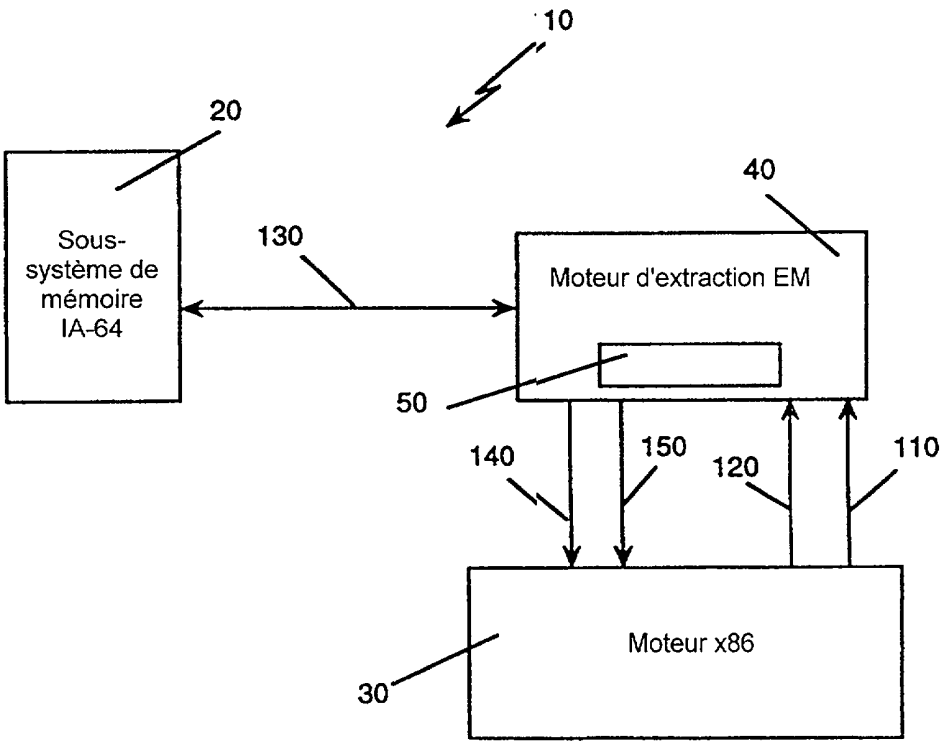

10
20
Sous-
système de
mémoire
IA-64
130
40
Moteur d'extraction EM
50
140
150
120
110
Moteur x86
30

Fig. 2

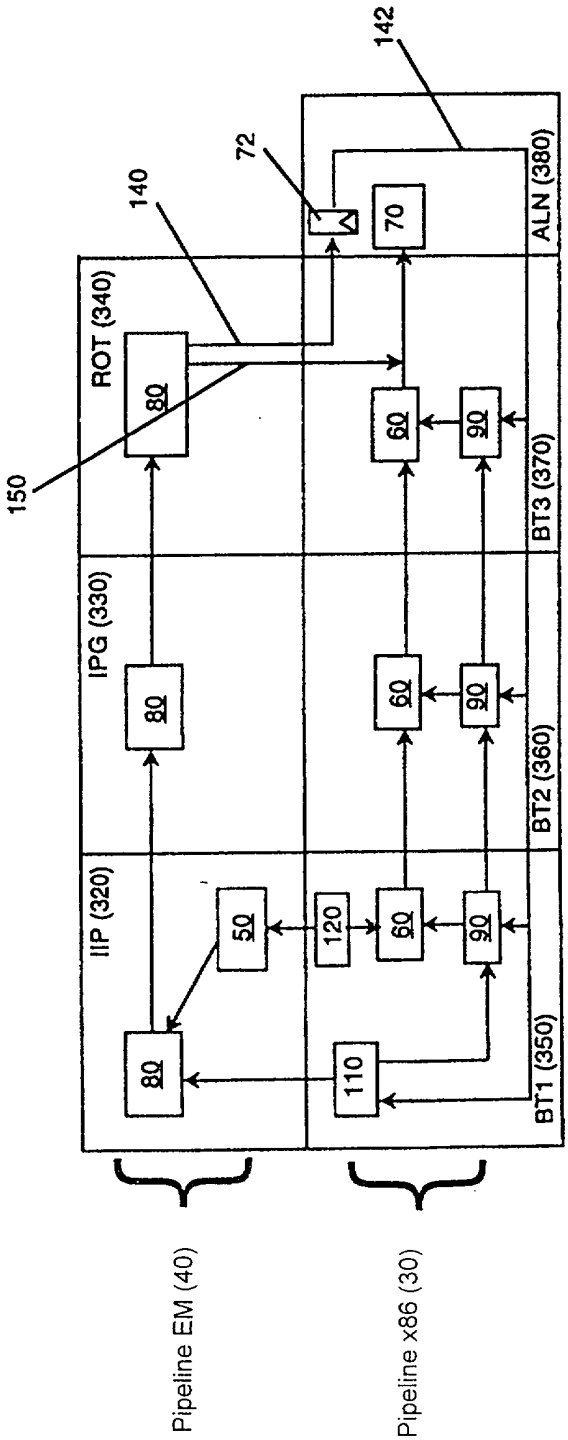

Pipeline EM (40)
Pipeline x86 (30)
IIP (320)
IPG (330)
ROT (340)
BT1 (350)
BT2 (360)
BT3 (370)
ALN (380)
80
50
120
60
90
110
70
72
140
142
150

Fig. 3

Progression de l'adresse d'extraction à travers le moteur d'extraction (40)

		IIP (320)	IPG (330)	ROT (340)
	t	A		
	t+1	B	A	
511 -	t+2	C	B	A
512 -	t+3	D	C	B
513 -	t+4	D	C	B
	t+5	E	D	C
	t+6	F	E	D

Fig. 4a

Le moteur x86 reflète la progression des adresses d'extraction (30)

		BT1 (350)	BT2 (360)	BT3 (370)	ALN (380)
	t	A			
	t+1	B	A		
521 -	t+2	C	B	A	
522 -	t+3	D	C	B	A
523 -	t+4	D	C	B	--
524 -	t+5	E	D	C	B
	t+6	F	E	D	C

Fig. 4b

Progression de l'adresse d'extraction 120 à travers le moteur d'extraction (40)

		IIP (320)	IPG (330)	ROT (340)
	t	A		
	t+1	B	A	
531 -	t+2	C	B	A
532 -	t+3	D	C	B
533 -	t+4	D	C	B
	t+5	E	D	C
	t+6	F	E	D

Fig. 5a

Le moteur x86 reflète la progression des requêtes d'extraction 120

		BT1 (350)	BT2 (360)	BT3 (370)	ALN (380)
	t	A			
	t+1	B	A		
541 -	t+2	C	B	A	
542 -	t+3	D	C	B	A
543 -	t+4	E	D	C	B
544 -	t+5	E	D	C	B
	t+6	F	E	D	C

Fig. 5b

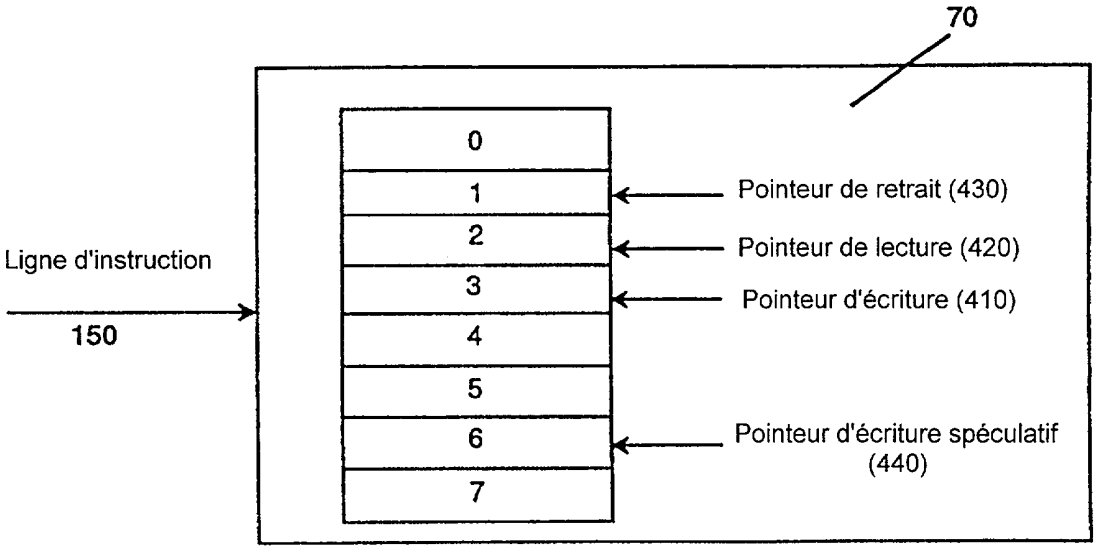

70
0
1
2
3
4
5
6
7
Pointeur de retrait (430)
Pointeur de lecture (420)
Pointeur d'écriture (410)
Pointeur d'écriture spéculatif (440)
Ligne d'instruction
150

Fig. 6