Etude des Architectures des Microprocesseurs DEC 21164, IBM POWER2 et MIPS R8000

André Seznec, Yann Mével

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Etude des Architectures des Microprocesseurs DEC 21164, IBM POWER2 et MIPS R8000

André Seznec, Yann Mével

N° 2553

Juin 1995

PROGRAMME 1

ISSN 0249-6399

INRIA
RENNES

Etude des Architectures des Microprocesseurs DEC 21164, IBM POWER2 et MIPS R8000

André Seznec, Yann Mével

Programme 1 — Architectures parallèles, bases de données, réseaux et systèmes distribués
Projet CAPS

Rapport de recherche n° 2553 — Juin 1995 — 137 pages

Résumé : Aujourd'hui, les microprocesseurs sont utilisés dans un grand nombre de systèmes matériels : stations de travail, multiprocesseurs, systèmes temps réels... L'évolution est extrêmement rapide dans ce domaine au niveau de l'intégration sur le circuit intégré, au niveau de la fréquence, au niveau architecture et au niveau du logiciel.

Nous étudions dans ce rapport les processeurs DEC 21164, l'IBM Power2 et le MIPS R8000. Effectuée dans le cadre d'une activité de veille technologique, cette étude met en avant les choix faits par les constructeurs et compare les diverses mises en œuvre.

Mots-clé : Veille technologique, architecture de microprocesseurs, DEC 21164, IBM POWER2, MIPS R8000.

(Abstract: pto)

Ce travail a été partiellement soutenu par la DRET (convention DRET-INRIA n° 94-2595 A/DRET), ainsi que par le GDR Architecture des Machines Nouvelles.

Unité de recherche INRIA Rennes

An Architecture Study of the Microprocessors DEC 21164, IBM POWER2 and MIPS R8000

Abstract: The DEC 21164, the IBM POWER2 and the MIPS R8000 are recent microprocessors. In this report, we present in an uniform and synthetic form, the informations which are available on these three microprocessors. We try to point out the convergence of all designs on some features and some divergences on others features.

Key-words: Technological watch, architecture of the microprocessors, DEC 21164, IBM POWER2, MIPS R8000

Table des matières

Table des figures

Liste des tableaux

Introduction

Cette étude comparative est la quatrième réalisée au sein de l'équipe CAPS[1] de l'IRISA. Elle s'inscrit dans une activité de veille technologique et de diffusion d'informations sur les microprocesseurs. Aujourd'hui, les microprocesseurs sont utilisés dans un grand nombre de systèmes matériels : stations de travail, multiprocesseurs, systèmes temps réels... L'évolution est extrêmement rapide dans ce domaine que ce soit au niveau de l'intégration, de la fréquence de fonctionnement, de l'émergence de nouvelles architectures ou du logiciel (compilateurs optimiseurs, exploitation de la localité des données). Aussi, cette activité de veille technologique permet d'orienter les axes de recherche du projet en architecture de processeurs, tout en observant les grandes tendances de cette fin du $XX^{ième}$ siècle.

Le rapport précédent comparait une architecture CISC (Intel Pentium) à une architecture RISC (PowerPC 601). Ces processeurs sont essentiellement destinés au marché des PCs et des stations de travail bas de gamme. Dans ce rapport, nous étudions des processeurs visant un tout autre marché, plus haut de gamme et donc aux performances bien plus élevées en particulier pour les applications en calcul scientifique.

Les 3 processeurs retenus, le DEC 21164, le Power2 et le MIPS R8000 utilisent des approches différentes. La première approche privilégie haute fréquence interne et simplicité de séquencement, alors que l'IBM Power2 et le MIPS R8000 ont des fréquences nettement moins élevées mais mettent en œuvre des mécanismes de séquencement et d'exécution en parallèle nettement plus élaborés. Bien entendu, ces trois processeurs sont superscalaires !

Le DECchip 21164 a été annoncé le 7 septembre 1994, les premiers échantillons devant être livrés en octobre 1994 alors que la production en volume est prévue pour le premier trimestre 1995. Ce processeur est destiné aux systèmes scientifiques, pour des serveurs haut de gamme et au marché des stations de travail.

Le Power2 a été annoncé à la même époque que le PowerPC 601 (septembre 1993). Il vient améliorer la compétitivité de son prédécesseur dans le domaine des stations de travail. Il équipe aussi les systèmes parallèles IBM SP2.

Le MIPS R8000 est commercialisé par Toshiba. Il vise clairement le marché des applications scientifiques et techniques (stations de travail graphiques 3-D), les serveurs de base de données, ainsi que le domaine des multiprocesseurs. Il équipe les machines de SGI[2].

Certains prédécesseurs de ces microprocesseurs (DEC 21064, IBM Power et MIPS R4000) ont déjà fait l'objet de nos études précédentes ([1], [2], [3]). Aussi, pourrons nous étendre la compa-

1. Compilation, Architectures Parallèles et Systèmes
2. Silicon Graphics Inc.

raison de cette nouvelle génération à ces processeurs et mettre ainsi en évidence l'évolution des architectures.

Chapitre 1

Jeu d'instructions

Avertissement : cet ouvrage concerne l'étude du DEC 21164 , du MIPS R8000 et du Power 2. Dans la mesure où leur prédécesseurs ont déjà fait l'objet d'une étude dans les rapports [2] et [1] et que ces processeurs assurent une compatibilité binaire (ascendante pour le R8000 et le Power2, complète pour le 21164), il existe de nombreuses similitudes au niveau de leurs jeux d'instructions. C'est pourquoi, nous renvoyons le lecteur ayant déjà étudié les documents précédemment cités à l'annexe A qui présente succinctement les différences entre les deux générations de processeurs. Le chapitre qui suit reprend de manière plus complète l'étude du jeu d'instructions de chacun des processeurs.

1.1 Introduction

Les jeux d'instructions des trois microprocesseurs sont RISC. Le concept RISC a été introduit au début des années 80 comme une alternative aux architectures dites CISC (pour *Complex Instruction Set Computer*). À la fin des années 70, la phase de recherche d'une instruction en mémoire durait plusieurs cycles. De ce fait, réduire le nombre des instructions en les codant plus densément permettait de meilleures performances. Les conséquences ont été un nombre important de format d'instructions, avec beaucoup d'instructions spécialisées, de nombreux modes d'adressage et des opérations effectuées sur des registres ou la mémoire. Un certain nombre de problèmes se posèrent alors :

- Les compilateurs n'arrivaient pas à utiliser toute la puissance des instructions. Il devenait dès lors inutile de les coder aussi densément.
- L'accroissement de la complexité du jeu d'instructions ne fut pas sans conséquences sur la complexité de l'architecture et du temps de cycle.
- Les progrès technologiques permettant des temps d'accès plus rapides et l'utilisation de mémoire locale au processeur, ou cache, ne justifiaient plus ce type d'architecture.
- Les jeux d'instructions CISC se prêtent mal aux pipelines.

Le concept RISC [4] (pour *Reduced Instruction Set Computer*) s'est naturellement imposé. Par opposition à l'architecture précédente, ce concept privilégie un jeu d'instructions réduit réalisant des opérations "simples" sur des registres uniquement et un format fixe d'instructions naturellement

alignées sur des frontières de mots. Ceci a conduit à une architecture plus simple avec des processus de compilation mieux adaptés.

Les 3 jeux d'instructions présentent un certains nombre de similitudes :

- Mots d'instructions d'une seule taille : 32 bits et alignés en mémoire sur des frontières de mot (adresse 0 modulo 4).

- Architecture *load/store* : toute opération est exécutée entre les registres et écrit son résultat dans un registre. Aucune opération d'accès mémoire autre que des instructions de *load* et de *store* n'est permise par le jeu d'instructions.

MIPS a défini le jeu d'instructions MIPS IV comme un sur-ensemble du jeu d'instructions MIPS III (MIPS R4000). Ce nouveau jeu d'instructions est implémenté sur le R8000 et sera aussi utilisé sur le R10000. Il comprend un nouvel ensemble d'instructions de multiplication-addition flottantes, un nouveau mode d'adressage indexé pour les opérations flottantes, plus des instructions de préchargement ainsi que des déplacements conditionnels.

Le jeu d'instructions du DEC 21164 est rigoureusement identique à celui de son prédécesseur le DEC 21064. Il comprend 160 instructions et est assez semblable à celui des autres architectures RISC, à l'exception notable qu'il ne possède pas d'instructions de transfert mémoire manipulant des données de 8 ou 16 bits. De plus le jeu d'instructions défini par l'architecture Alpha inclut des instructions permettant d'accéder à une librairie de routines, *Privileged Architecture Library*, utilisée pour le traitement des exceptions ainsi que pour assurer certaines opérations systèmes.

Le jeu d'instructions du Power2 reprend lui aussi la totalité du jeu d'instructions de son prédécesseur l'IBM Power. Il introduit de plus deux nouvelles instructions de transfert de données sur 128 bits, une instruction de racine carrée émulée par logiciel sur la version précédente, et des instructions de conversion entier-flottant.

Dans ce chapitre, nous étudions les différents types de données manipulées, les modes d'adressage possibles, les formats d'instructions disponibles ainsi que les catégories d'instructions sur les trois microprocesseurs.

1.2 Architecture 32 ou 64 bits

Différence essentielle entre ces 3 microprocesseurs, le Power2 est le seul des processeurs étudiés à être encore d'architecture 32 bits alors que le MIPS R8000 et le DEC 21164 sont tous deux de type 64 bits. Le calcul des adresses est donc effectué sur 32 bits sur le Power2 et sur 64 bits sur les autres architectures. Le principal atout d'une architecture 64 bits, est qu'elle peut adresser 2^{64} octets de mémoire, soit 4 milliards de fois plus qu'une architecture 32 bits. Ceci commence à trouver sa justification avec l'émergence des applications multimédias, la CAO de haut de gamme, la simulation intensive ou encore les serveurs de données et l'arrivée sur le marché de machines avec une mémoire physique de plusieurs gigaoctets. L'utilisation d'un espace virtuel d'adressage de très grande taille par rapport à la mémoire physique simplifie d'autre part le système d'allocation des adresses virtuelles et physiques pour les processus.

Les processeurs MIPS R8000 et DEC 21164[1] n'utilisent cependant pas la totalité des 64 bits du bus d'adresse (voir chapitre 6.3.2).

1. Comme l'ensemble des processeurs de type 64 bits actuellement sur le marché.

1.3 Types de données

Le choix des concepteurs du 21164 est résolument orienté vers les performances en absence d'instructions d'accès à la mémoire manipulant des octets ou des mots de 16 bits. Un support à ce type de données est fourni par le biais d'instructions de manipulation. Nous reviendrons ultérieurement sur cette particularité.
Le MIPS R8000 offre dans son jeu d'instructions la possibilité de manipuler tous les types de données, du simple octet jusqu'au mot de 64 bits. Le Power2 offre de plus des instructions de transfert de données de 128 bits vers deux registres flottants adjacents.

Il est à noter que l'architecture des machines MIPS et DEC impose la manipulation de données alignées en mémoire alors que l'architecture Power a su s'affranchir de cette contrainte. Le DEC et le MIPS ont un rangement par défaut des données dans l'ordre *little-endian*[2], alors que l'IBM les range dans l'ordre *big-endian*[3] .

1.4 Modes d'adressage

Le tableau 1.1 récapitule les divers modes d'adressage disponibles sur les processeurs. Les trois processeurs étudiés offrent les adressages absolu, indirect et basé.

Modes d'adressage	DEC 21164	MIPS R8000	Power2
absolu	R31 + immédiat	R0 + immédiat	R0 + immédiat
indirect	registre + 0	registre + 0	registre + 0
basé	registre + immédiat	registre + immédiat	registre + immédiat
indexé	-	registre + registre[a]	registre + registre
préincrémenté	-	-	registre + immédiat registre + registre

TAB. 1.1 - *Les modes d'adressage*

[a] Ne concerne que les opérations flottantes

Chacune des architectures possède un registre (soit R0, soit R31) câblé à zéro et utilisé comme opérande source, qui permet de définir trois possibilités d'adressage (absolu, indirect et basé). Certains processeurs RISC incluent également des formes d'adressage *registre + registre* pour les accès à la mémoire de données entières et flottantes. Le DEC 21164 n'a pas implémenté directement cet adressage (2 instructions consécutives sont nécessaires pour l'émuler).
Ceci était également le cas sur le MIPS R4000. Cette lacune est comblée dans le jeu d'instructions MIPS IV. L'utilisation d'un pipeline peu commun sur le R8000 (confère chapitre 3.2.2) a incité les concepteurs à introduire ce type d'adressage lors des accès aux données flottantes dans le but de réduire le besoin de précalculer des adresses (adressage particulièrement utile lors d'accès à des tableaux).
Le jeu d'instructions Power2 offre aussi l'adressage indexé avec de plus une option de mise à jour

2. L'octet de poids faible est rangé à l'adresse la plus basse
3. L'octet de poids fort est rangé à l'adresse la plus basse

implicite du registre de base (préincrémentation). Cette technique permet de gagner une instruction à chaque accès comme l'illustre l'exemple ci-dessous.

Exemple 1

Sans préincrémentation :

```
R1 ← R1 + 4        ; Remise à jour du registre
                   ; permettant l'adressage.
LD R1, R2          ; L'élément suivant est accédé
                   ; et placé dans R2.
```

Avec préincrémentation :

```
LDU R1 + 4, R2     ; La mise à jour de R1 est effectuée.
```

Le chargement d'une donnée qui utilise la préincrémentation produit deux résultats : la donnée lue et le registre d'adresse mis à jour (pour le prochain accès). Pour pouvoir être mise en œuvre, la technique de la préincrémentation pour les lectures d'entiers nécessite donc la présence d'un port d'écriture supplémentaire sur le banc de registres entiers. La mise en œuvre de cette technique uniquement sur les instructions de lecture/écriture flottantes et les instructions d'écritures entières aurait, par contre été immédiate puisque lors de telles opérations, le banc de registres entiers n'est pas accédé en écriture.

Les immédiats pour les 3 microprocesseurs sont des valeurs signées de 16 bits.

À remarquer les choix effectués sur les trois architectures :

- sur le DEC 21164, les accès mémoire font au maximum deux accès en lecture sur le banc de registres (cas d'une écriture mémoire) ou un accès en lecture et un accès en écriture, soit la nécessité de disposer de 3 ports (2 en lecture et 1 en écriture) sur le banc de registres pour chacun des pipelines d'exécution entier.

- sur le MIPS, idem que précédemment avec la petite astuce sur les flottants. Effectivement, l'ajout de l'adressage indexé uniquement sur les données flottantes évite un port supplémentaire sur le banc de registres entiers.

- sur les architectures Power et PowerPC, possibilité d'avoir 3 lectures plus 1 écriture ou 2 lectures plus 2 écritures sur les registres entiers : ce qui impose 5 ports.

1.5 Formats des instructions

Pour les 3 jeux d'instructions, la philosophie RISC a été utilisée. Le nombre de formats d'instructions est donc limité afin de faciliter le décodage. Toutes les instructions sont alignées en mémoire sur des frontières de mot de 32 bits.

On remarquera dans les 3 jeux d'instructions les similitudes suivantes :

- Utilisation de champs fixes pour le codage de l'opération et des registres opérandes : ceci permet un décodage rapide en parallèle avec la lecture systématique des opérandes en registres (lecture par anticipation).

– 32 registres entiers et 32 registres flottants : 5 bits sont donc nécessaires pour coder les numéros de registres adressés par l'instruction.

1.5.1 Formats des instructions du DEC 21164

Sur le DEC 21164 cinq formats d'instructions de base sont utilisés :

– format dit des instructions mémoire[4],

– format des instructions de branchement,

– format des instructions d'opérations entières,

– format des instructions d'opérations flottantes,

– format des instructions du PALcode.

Le format de l'instruction diffère selon son type, cependant le code définissant le type de l'opération réalisée est toujours codé sur les 6 bits de poids fort 31 à 26.

Format des instructions mémoire

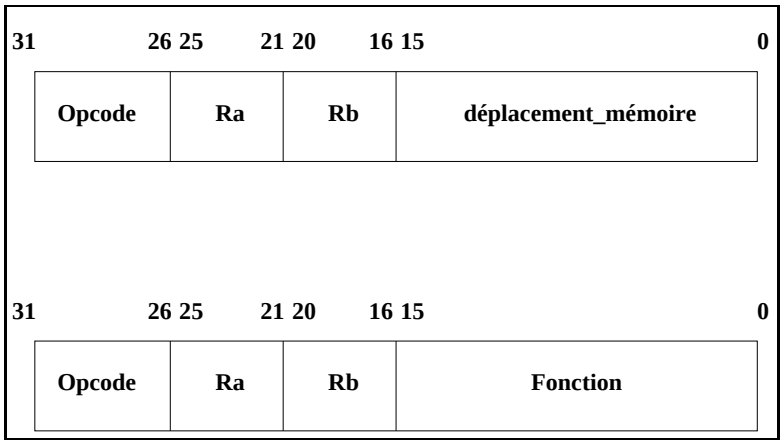


FIG. 1.1 - *Format des instructions mémoire du DEC 21164*

Ce format (figure 1.1) est utilisé pour tous les transferts entre les registres et la mémoire, pour charger une adresse virtuelle et pour effectuer des branchements inconditionnels. Le champ de déplacement mémoire est codé en octets sur 16 bits, il est signé et ajouté au contenu de *Rb* pour former une adresse virtuelle (les dépassements de capacité lors du calcul sont ignorés dans ce cas).

Le deuxième type de format implémente un champ *fonction* qui désigne un ensemble d'instructions diverses telles :

– l'instruction *Memory Barrier*,

– l'instruction *Prefetch* définie par l'architecture Alpha,

4. Ce format est également utilisé pour d'autres instructions

– des appels à des routines du Palcode,

– ...

Format des instructions de branchement

Ce format (figure 1.2) est utilisé pour les branchements conditionnels et pour les sauts relatifs (pour calculer l'adresse du saut, une valeur est ajoutée à l'ancienne valeur du compteur ordinal). Le déplacement est en fait de 23 bits signé, deux zéros étant concaténés au champ déplacement : les instructions sont alignées sur les mots de 32 bits.

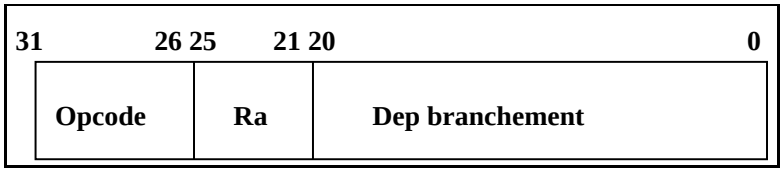


FIG. 1.2 - *Format des instructions de branchement du DEC 21164*

Formats des instructions d'opérations entières

Deux formats pour des opérations entières de registre à registre sont utilisés. Ils sont représentés figure 1.3.

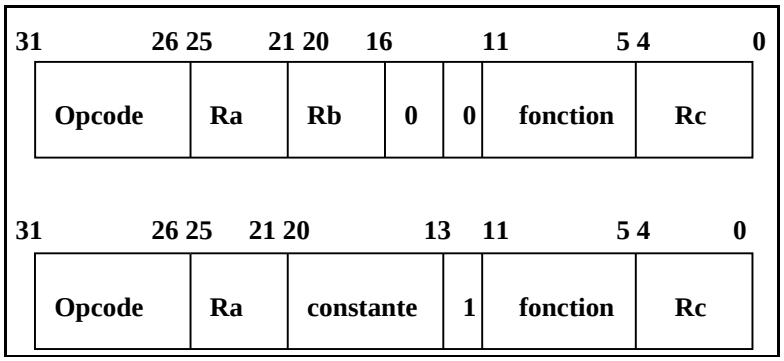


FIG. 1.3 - *Formats des instructions entières du DEC 21164*

Le bit 12 spécifie le type d'opération réalisée :

– bit 12 = 0 (cas du premier format d'instruction) : opération entre deux registres sources *Ra* et *Rb* et un registre destination *Rc*.

– bit 12 = 1 : opération entre un registre source *Ra* et une constante positive codée sur 8 bits (étendue pour le besoin du calcul à 64 bits).

Dans les deux cas un champ de 7 bits complète le champ *opcode*.

Format des instructions d'opérations flottantes

Le format des opérations flottantes est illustré figure 1.4.
À noter que dans le cas des instructions de conversion, *Fb* spécifie le registre source alors que *Fa* désigne impérativement le registre *F31*.

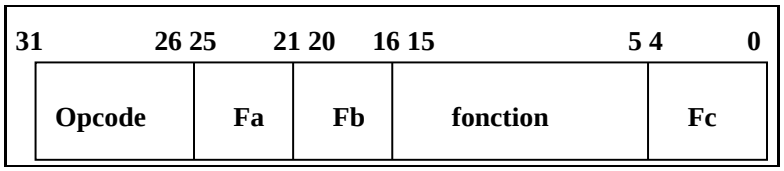


FIG. 1.4 - *Format des instructions flottantes du DEC 21164*

Format des instructions de PALcode

Ce format est utilisé pour appeler des fonctions du PALcode, sa représentation est fournie par la figure 1.5. Il contient 6 bits de code opération et un champ de 26 bits pour spécifier la fonction appelée.

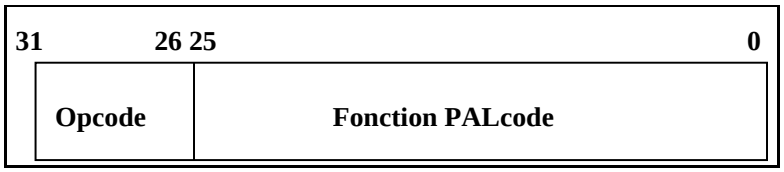


FIG. 1.5 - *Format des instructions du PALcode du DEC 21164*

Les opérandes source et destination pour ce type d'instructions sont fournies par des registres fixés au moment de la définition de l'instruction.

Le PALcode, *Privileged Architecture Library Code*, est un ensemble de procédures spécifiques à un système d'exploitation. Ces procédures fournissent des primitives pour la gestion de la mémoire, pour traiter les exceptions et les interruptions, et interviennent également lors des changements de contextes. Elles peuvent être appelées aussi bien par le matériel (traitement des exceptions) que par logiciel, et sont exécutées dans un mode spécifique appelé le mode PAL (voir chapitre 6.5.1). Huit registres, appelés *PAL shadow registers* sont destinés à entretenir des copies locales dans ce mode.

1.5.2 Format des instructions du MIPS R8000

Trois formats d'instructions sont utilisés :

– Instructions de type immédiat (I-Type) (figure 1.6).

  Ce format a trois usages : il est utilisé pour les instructions de saut et de branchement conditionnel, pour certaines instructions arithmétiques, ainsi que pour les instructions de transfert de données.

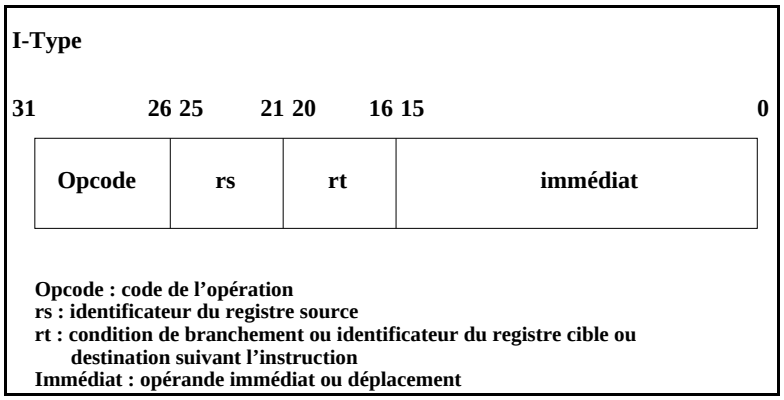


FIG. 1.6 - *Format I-type du R8000*

Le R8000 ne possède pas de codes condition pour les instructions entières. Lorsqu'un saut ou un branchement conditionnel doit être effectué, la comparaison a lieu dans l'instruction de saut : les deux registres désignés par *rt* et *rs* sont comparés et le résultat sert de condition au saut à l'adresse cible.

Une autre utilisation du format I-type concerne les instructions arithmétiques et logiques à deux opérandes : le registre désigné par *rs* et la constante (immédiat) sont les opérandes sources, le registre désigné par *rt* est la destination.

La dernière utilisation de ce format concerne les instructions de transfert de données (*load/store*). Le champ immédiat contient le déplacement alors que le champ *rs* désigne le registre de base de l'adresse de l'opérande à transférer. Le champ *rt* représente, pour sa part, le registre source ou destination suivant le sens de la transaction.

– Instructions de type saut (J-Type) (figure 1.7).

Ce format est utilisé pour les sauts et les branchements sans condition. La cible ne contient que 26 bits : après un décalage à gauche de 2 positions, on la concatène aux 4 bits de poids fort du compteur ordinal pour obtenir les 32 bits de l'adresse absolue. Lors d'une instruction *Jump and Link*, l'adresse de retour est automatiquement placée dans le registre R31.

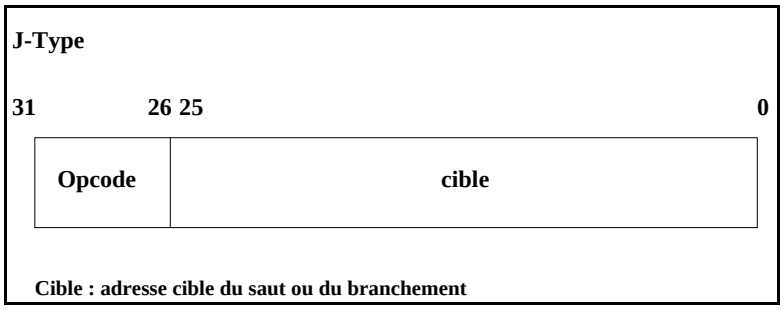


FIG. 1.7 - *Format J-type du R8000*

– Instructions de type registre (R-type) (figure 1.8).

La principale utilisation de ce format concerne les instructions arithmétiques à trois ou quatre opérandes puisque le MIPS R8000 inclut un nouveau type d'instruction de multiplication-addition flottante. *rs* et *rt* sont les identificateurs des registres sources alors que *rd* désigne le registre destination. Dans le cas d'une instruction de multiplication-addition, *fr, fd, fs* sont des registres flottants source, alors que *fd* désigne le registre destination.

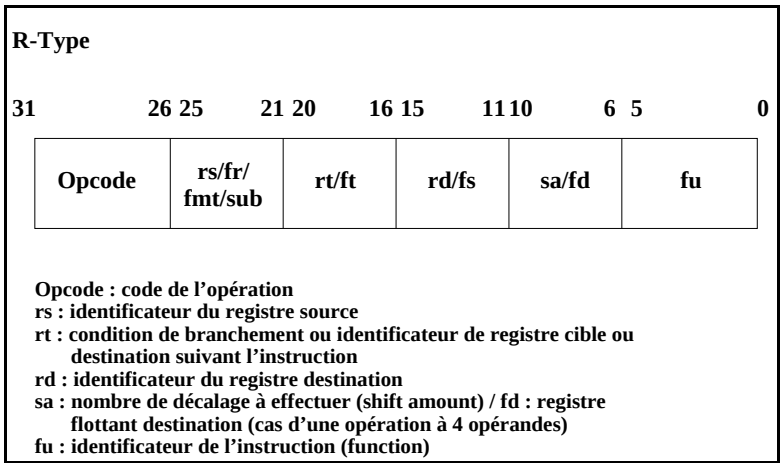


FIG. 1.8 - *Format R-type du R8000*

Il existe d'autres utilisations de ce format, comme les appels au système qui déclenchent une routine sous le contrôle du manipulateur d'exceptions ou bien les instructions déclenchant des routines d'exception après avoir testé une condition.

À ces trois types de base, ont été rajoutés 3 formats supplémentaires d'instructions :

– le type *CI-type* (Floating-point condition-code I-type),

– le type *CR-type* (Floating-point condition-code R-type),

– le type *RC-type* (Register to floating-point condition code).

1.5.3 Formats des instructions du Power2

L'architecture Power compte un grand nombre de formats d'instructions, onze en tout. On peut cependant distinguer 5 formats principaux :

– Instruction de type *registre-registre* (R-R, figure 1.9). Ce format s'applique essentiellement aux opérations arithmétiques dont les deux opérandes sources sont situées respectivement dans les registres *Rs1* et *Rs2*, le résultat de l'opération figurant dans le registre destination *Rd*. Les champs Op et Opx constituent le code opération et son extension. Le bit *Rc* permet, quand il est validé, la mise à jour des codes condition.

– Instruction de type *registre-immédiat* (R-I, figure 1.10). Ce format a le même usage que le format précédent. Au lieu de provenir d'un registre, la deuxième opérande source est une valeur immédiate, codée sur 16 bits en complément à deux.

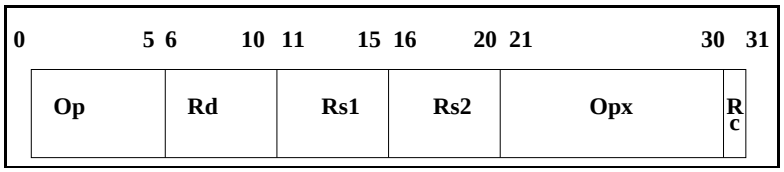


FIG. 1.9 - *Instruction du type registre-registre*

Remarque : les deux formats précédents s'appliquent aux instructions mémoires. Dans ce cas l'opération arithmétique effectuée correspond au calcul d'une adresse.

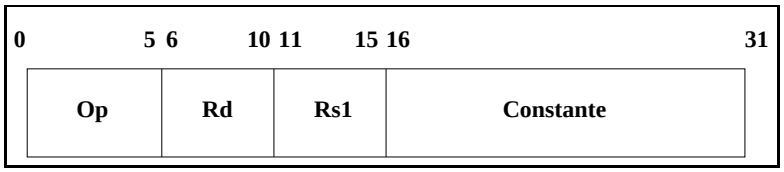


FIG. 1.10 - *Instruction du type registre-immédiat*

– Instruction de type *branchement* (figure 1.11). Ce format s'applique aux branchements conditionnels qui font suite à l'évaluation d'une condition.
Le Power2 utilise 4 codes condition qui sont inférieur, supérieur, égal et débordement. Un registre spécial, appelé le *Condition Register*, sert à mémoriser les résultats des opérations de comparaison et les codes conditions implicites (8 codes condition implicites), plus 6 champs résultats de 4 bits de comparaison explicites. Ceux-ci permettent l'évaluation à l'avance de plusieurs conditions de branchements.
Le champ *BI* sélectionne la condition de branchement (ce champ de 5 bits permet l'adressage d'un bit dans le *Condition Register*). Le champ *AA* spécifie le calcul de l'adresse cible (saut à une adresse absolue indiquée par le champ *BD* ou à une adresse relative), et le champ *LK* indique si l'adresse de l'instruction qui suit le branchement doit être placée dans le registre de lien (retour de procédures).

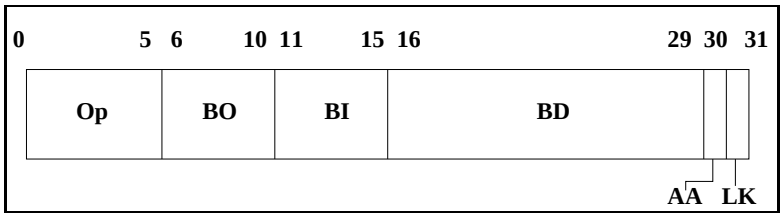


FIG. 1.11 - *Instruction du type branchement*

La condition d'un branchement conditionnel peut être établie par une opération entière située en amont du saut dans le flot d'instructions, mais elle peut aussi dépendre de la valeur du *Count Register* (ce registre peut de plus être décrémenté à chaque saut). Ce type d'instruction permet une gestion particulièrement simple des boucles. On séquence le même corps tant que le compteur de boucle (*Count Register*) n'est pas nul. Ceci permet de découpler le séquencement d'une boucle de son exécution sans avoir recours à un mécanisme de prédiction dynamique de branchement.

– Instructions de saut et appel de procédure (J/C)

Pour les branchements inconditionnels il existe la possibilité d'utiliser la constante comme une adresse absolue.

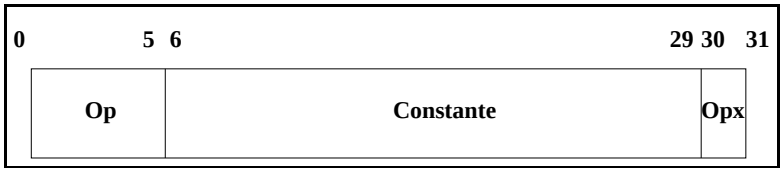


FIG. 1.12 - *Instruction de saut et appel de procédure*

Le Power2 offre les deux possibilités de saut avec ou sans sauvegarde de l'adresse de retour. Le saut avec sauvegarde de l'adresse de retour sert pour les appels de procédure. Cependant, à la différence des jeux d'instructions RISC plus classiques (DEC et MIPS), le Power2 utilise un registre spécial *Link register* pour sauvegarder l'adresse de retour (adresse de l'instruction qui suit le saut).

– Le format arithmétique en virgule flottante est particulier avec des instructions à quatre opérandes pour s'adapter à l'opérateur de multiplication-addition.

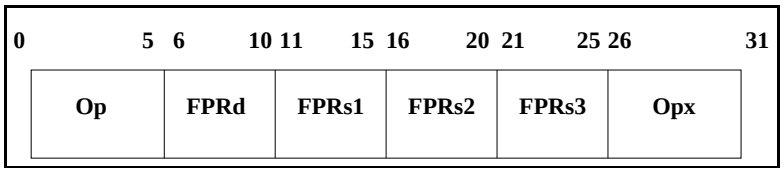


FIG. 1.13 - *Instruction des opérations arithmétiques flottantes*

1.6 Instructions

1.6.1 Les instructions d'accès à la mémoire

Les modes d'adressage ont été vus au chapitre 1.1, nous ne détaillerons donc ici que quelques points particuliers.

Instructions de synchronisation

Les 3 processeurs comprennent dans leur jeu d'instructions des instructions de synchronisation utilisées lors des accès à la mémoire. Ces instructions offrent à l'utilisateur la possibilité de mettre en œuvre des primitives de synchronisation telles que: *test-and-set*, sémaphores, *bit-level locks*, séquenceurs et compteurs d'évènements. L'exemple 2 illustre une séquence de code exécutée de manière atomique en utilisant les instructions *Load Linked* et *Store Conditional* définies sur le R4000 et inclues dans le jeu d'instructions MIPS IV.

Exemple 2 : *mise en œuvre d'un sémaphore*

```
L1 :
        LLD     T1, (T0);     Chargement de la valeur dans T1
        ADD     T2, T1, 1;    Incrémentation
        SCD     T2, (T0);     Mise à jour de la mémoire
        BEQ     T2, 0, L1;    Test du sémaphore et si
                              valeur nulle saut en L1
```

Sur le DEC 21164, ces instructions sont appelées *Load-locked* et *Store-conditional*, un exemple de leur utilisation est également donné au chapitre 10 (page 111).

Le Power2 offre des instructions similaires : *lwarx* (*Load Word and Reserve Indexed*) et *stwcx* (*Store Word Conditional Indexed*). La première instruction permet de lire un mot en mémoire tout en effectuant une *réservation* à la même adresse. L'instruction suivante teste si une réservation existe à une certaine adresse et met à jour la mémoire si c'est le cas.

Lecture/écriture d'octets et de mots non alignés

DEC 21164 Une différence essentielle entre le jeu d'instructions DEC Alpha et les autres jeux d'instructions RISC est l'absence, sur le DEC, d'instructions de transfert de données concernant les octets ou les mots de 16 bits. On dispose pour manipuler ces données d'instructions particulières d'extraction, d'insertion et de masquage, et des instructions de chargement de mots de 32 et 64 bits. Ainsi pour charger un octet ou un mot depuis la mémoire, le mot de 32 ou 64 bits le contenant est d'abord chargé, l'octet ou le mot désiré est ensuite extrait de la donnée chargée.

De même, pour écrire un octet dans la mémoire, il faut d'abord charger depuis la mémoire le mot de 32 ou 64 bits le contenant, y insérer l'octet ou le mot concerné et charger dans la mémoire le mot de 32 ou 64 bits modifié. L'exemple 3 représente une séquence d'instructions permettant d'écrire un octet en mémoire.

Exemple 3 : *Exemple d'écriture d'un octet en mémoire sur le DEC Alpha*

```
(1)   LDL     R1, 0(R9)
(2)   INSBL   R5, #1, R3
(3)   MSKBL   R1, #1, R1
(4)   BIS     R3, R1, R1
(5)   STL     R1, 0(R9)
```

L'instruction (1) charge le mot de 32 bits dont l'adresse est le contenu de R9, l'instruction (2) insère l'octet concerné dans les huit bits de poids faible (*INSBL= Insert Byte Low*, *#1* spécifie un décalage) et met les autres à zéro. L'instruction (3) masque ensuite le registre contenant le mot de 32 bits en mettant à 1 les huit bits de poids faible du registre R1, l'instruction (4) effectue un ou logique entre le registre R3 qui contient l'octet à écrire et R1. L'octet concerné étant inséré dans le registre R1, son contenu est écrit en mémoire par l'instruction STL.

Cette orientation a été volontairement choisie par DEC pour éviter d'alourdir inutilement le jeu d'instructions par des instructions peu utilisées. En effet, DEC affirme que les instructions portant sur les octets ou les mots ne représentent que 8% des instructions et qu'en les supprimant,

la pénalité engendrée n'est pas très importante : cette option nous semble une erreur manifeste ; la séquence nécessaire pour ranger un octet, illustrée par l'exemple 3, est séquentielle et nécessite un minimum de 5 cycles pour être séquencée sur le 21164 ! Avec l'élargissement du séquencement superscalaire, le coût relatif d'une écriture d'octet devient considérable par rapport à l'écriture d'un mot.

Des instructions d'extraction sont fournies pour le traitement et le transfert de données non alignées. Ceci est particulièrement important, car DEC implémente un traducteur et un interpréteur pour les clients VAX afin de faire la transition et des accès à des données non alignées peuvent subsister dans ces codes traduits. Les instructions LDQ-U et STQ-U permettent de charger des données en ignorant sur un load ou un store les trois bits de poids faible de l'adresse : ainsi par des séquences judicieuses de LDQ-U ou STQ-U successives ainsi que des instructions d'extraction, les données non alignées peuvent être accédées.

Contrairement aux deux autres processeurs, le DEC offre un support matériel uniquement au mode d'organisation des octets *little-endian*.

MIPS R8000 Le MIPS R8000 dispose de 8 instructions spéciales utilisées pour lire ou écrire des mots ou double-mots non alignés en mémoire sur des frontières de 4 ou 8 octets. Ces instructions se répartissent en 4 paires conformément au tableau 1.2. L'adressage de données non alignées induit un cycle supplémentaire du fait de l'exécution de 2 instructions.

LWL	- Load Word Left	SWL	- Store Word Left
LWR	- Load Word Right	SWR	- Store Word Right
LDL	- Load Doubleword Left	SDL	- Store Doubleword Left
LDR	- Load Doubleword Right	SDR	- Store Doubleword Right

TAB. 1.2 - *Instructions d'accès à des données non alignées*

L'instruction *Load Word Left* lit la partie haute d'un mot non aligné en mémoire et l'inscrit dans la partie gauche du registre destination, alors que l'instructions *Load Word Right* traite la partie basse. De-même dans le cas des écritures.

Instructions de transfert de 128 bits sur le Power2

L'architecture Power2 inclut deux nouvelles instructions qui devraient lui permettre d'accroître ses performances flottantes. *Load Quad* et *Store Quad* supportent toutes les formes d'adressage pour les références mémoires double précision : indexée, déplacement, avec ou sans mise à jour. Les chargements de quadruple mot (128 bits) déplacent deux opérandes de la mémoire vers deux registres flottants adjacents.

L'exemple 4 illustre le bénéfice apporté par ces nouvelles instructions utilisées en synergie avec la technique de *loop unrolling* (déroulement de boucles) appliquée par les compilateurs.

Exemple 4 : *Bénéfice des instructions de référence mémoire sur une boucle du benchmark Linpack*

```
DO 100 i=1,n                          loop : lfd   fpr2,8(r11)          ; a(i,j)
  a(i,j) = a(i,j) + a(i,k) * atemp           lfdu  fpr4,8(r12)          ; a(i,k)
ENDDO                                        fma   fpr6,fpr2,fpr4,fpr8  ; fpr8 = atemp
                                             stfdu fpr6,8(r11)          ; a(i,j)
                                             bct   loop
Loop unrolling:
                                      loop : lfd   fpr2,8(r11)          ; a(i,j)
                                             lfd   fpr3,16(r11)         ; a(i+1,j)
DO 100 i=1,n,2                               ——
  a(i,j) = a(i,j) + a(i,k) * atemp           lfd   fpr4,8(r12)          ; a(i+k)
  a(i+1,j) = a(i+1,j) + a(i+1,k) * atemp     lfdu  fpr5,16(r12)         ; a(i+1,k)
ENDDO                                        ——
                                             fma   fpr6,fpr2,fpr4,fpr8  ; fpr8 = atemp
                                             fma   fpr7,fpr3,fpr5,fpr8
                                             ——
                                             stfd  fpr6,8(r11)          ; a(i,j)
                                             stfdu fpr7,16(r11)         ; a(i+1,j)
                                             bct   loop
```

La première boucle met en évidence 3 opérations de référence mémoire, et l'opération de calcul effectuée. Elle montre les limites de performances pour un code exécuté par exemple sur le Power1. La deuxième boucle représente le même code après déroulement de la boucle avec un pas de 2. Ce pseudo-assembleur montre 3 paires de références mémoire. Chaque paire invoque 2 accès mémoire liés à deux registres flottants adjacents. Les nouvelles instructions mémoires sur des quadruple-mot remplaceront avantageusement chacune de ces paires. De plus, on exploite au mieux le parallélisme entre les opérateurs en séquençant deux calculs par cycle. Ainsi, en à peu près le même nombre de cycles, chacune des itérations doublera les performances de la boucle précédente.

Instructions de manipulation de chaînes

L'architecture Power (et PowerPC) est la seule dans le domaine des processeurs RISC à mettre en œuvre des instructions de manipulation de chaînes de caractères (*string*) qui permettent un codage simple de la routine *string copy (strcpy)* du langage C (voir rapport PowerPC 601 [3]).

À noter cependant l'instruction *Compare Byte* du jeux d'instructions DEC qui effectue la comparaison de 8 octets en parallèle. Cette instruction, efficace lors des tests de chaînes de caractères impose cependant l'alignement des données.

1.6.2 Instructions arithmétiques et logiques de l'unité entière

Instructions arithmétiques et logiques entières du DEC Alpha

L'ensemble des instructions entières est assez classique par rapport à celui des autres microprocesseurs. Quelques instructions présentent cependant des particularités. Les instructions de multiplication, addition et soustraction sont fournies à la fois en 64 bits et 32 bits. Ces instructions offrent une option qui permet d'indiquer si la détection des *overflows* est ou non désirée. Il n'y a pas d'instruction de division entière. Pour effectuer les divisions par les constantes connues à la compilation, on utilise les optimisations classiques (décalages, etc...). Pour une division par une variable, une routine spécialisée est appelée.

Des instructions *conditional-move* sont mises en œuvre sur le DEC Alpha. Elles permettent de copier un registre source dans un registre destination si un troisième registre satisfait une condition. Ces instructions permettent de mettre en œuvre un cas fréquent sans que cela puisse engendrer une rupture de séquence dans le pipeline. En effet, une instruction *conditional-move* est équivalente à une instruction de test et branchement suivie d'une instruction *move*. Dans ce dernier cas, une prédiction de branchement devrait être effectuée avec les risques d'erreurs de prédiction que cela comporte et donc les risques d'annulation d'instructions déjà séquencées. De plus, le fait de mettre en œuvre en une seule instruction ce qui se fait habituellement en deux permet d'accélérer le processus.

Les séquences suivantes illustrent l'utilisation de ces instructions sur un exemple simple (calcul du minimum de deux nombres).

La séquence d'instructions nécessaire au calcul du *min (A,B)* sans instructions *conditional-move* (R4000 par exemple) est présentée à l'exemple 5.

Exemple 5

```
        LD      R1, A
        LD      R2, B
        SUB     R3, R1, R2  ; R3 ← A − B
        BGEz    R3, Etiq1   ; Branchement à Etiq1 si R3 < 0
        MV      R2, R1
Etiq1   ...
        ST      R1, A
```

Sur le DEC Alpha, des instructions et surtout une rupture de séquencement sont évitées suite à l'utilisation de l'instruction *conditional-move* illustrée par l'exemple 6

Exemple 6 : *séquence de code sur le DEC Alpha*

```
LD          R1, A
LD          R2, B
SUB         R3, R1, R2  ; R3 ← R1 − R2
CMOVGT      R3, R2, R1  ; si R3 > 0 alors R1 ← R2
ST          R1, A
```

Instructions arithmétiques et logiques entières du R8000

Il existe quatre différents types d'instructions arithmétiques et logiques qui utilisent deux formats différents (*R-type* et *I-type*) :

- instructions arithmétiques immédiates : les deux opérandes sources sont un registre et une constante (16 bits).

- instructions arithmétiques à trois opérandes : les deux opérandes sources sont contenues dans des registres.

- instructions de décalage : le nombre de décalages à effectuer est contenu soit dans le champ *sa* soit dans le champ *rs* du format R-type ; le résultat est chargé dans le registre désigné par *rd*.

- instructions de multiplication et de division : ces instructions sont particulières par le nombre élevé de cycles qu'elles nécessitent ainsi que par l'ordonnancement particulier qu'elles requièrent pour être exécutées sans génération d'exceptions.

Des instructions *conditional-move* ont été introduites. Ces instructions utilisent comme condition, un code condition flottant ou la comparaison d'un registre par rapport à la valeur zéro (voir l'exemple 6 qui illustre leur utilisation).

Le jeu d'instructions du R8000 offre la compatibilité ascendante par rapport au R4000 et R3000. Il est donc possible d'exécuter des opérations avec des opérandes de 32 bits en même temps que des opérations avec des opérandes de 64 bits. On peut même étendre des données de 32 bits à 64 bits en conservant le signe ou tronquer des données de 64 bits en 32 bits. Ceci permet de profiter de l'architecture 64 bits sans perdre le lien avec les programmes 32 bits déjà existants.

Instructions arithmétiques et logiques entières du Power2

En plus des opérations classiques, l'architecture Power compte un grand nombre d'instructions de rotation et de décalage (52). Toutes ces instructions sont construites à partir de deux opérations de base de rotation et de masquage.

Deux instructions coexistent pour chaque type d'opération, l'une mettant à jour les codes conditions et l'autre non.

1.6.3 Instructions de contrôle

L'architecture Power et, désormais, l'architecture MIPS IV (pour les flottants) mettent en œuvre des codes conditions. La mise en œuvre de codes conditions produit à chaque opération une série de résultats permettant de savoir si le résultat de l'opération est positif, négatif... Ces résultats servent notamment dans le cas des branchements. Sur le R8000, seules les opérations flottantes affectent ces codes conditions. Un des avantages des codes conditions est qu'il n'est pas nécessaire d'effectuer explicitement des comparaisons.

Sur le DEC Alpha, il a été préféré les tests et comparaisons classiques.

Instructions de transfert de contrôle du MIPS R8000

Ces instructions changent la valeur du compteur ordinal. Le branchement peut être relatif au compteur ordinal ou bien être absolu. Lorsque le saut est relatif, les 26 bits immédiats sont décalés deux fois à gauche puis concatènés aux bits de poids fort du compteur ordinal pour obtenir l'adresse absolue de la cible. Lorsque le saut est absolu, l'adresse cible est fournie par le contenu d'un registre.

Avatar de la compatibilité binaire, l'architecture MIPS met en œuvre la technique du branchement retardé. C'est à dire que tous les sauts et branchements sont définis par l'architecture avec un retard de une instruction (*branch delay slot*) qui est toujours exécutée. Un *branch delay slot* ne doit pas lui même contenir une instruction de branchement ou de saut, sans quoi le résultat de l'opération est indéfini.

Seules les opérations flottantes affectent les codes conditions. Ces codes conditions ont permis de définir quatre nouvelles instructions de branchement. Deux de ces instructions s'affichent comme des extensions de l'architecture MIPS I (branchement sur codes conditions flottants vrai ou faux) alors que les deux autres sont plutôt des extensions de l'architecture MIPS III. Il s'agit des instructions *Branch likely*, qui permettent au compilateur d'insérer dans le *delay slot* une instruction issue du branchement et qui peut être annulée si le branchement n'est pas pris.

Comme ses prédécesseurs, le R8000 procure un test et un branchement en une seule instruction. Les différents tests effectués dans les instructions de comparaison sont faits par rapport à zéro ($= 0, < 0, > 0, \leq 0$), ceci implique souvent une instruction supplémentaire de soustraction pour la comparaison de deux registres excepté le cas où l'on teste directement l'égalité de 2 registres.

On peut remarquer aussi que l'architecture MIPS fournit une instruction d'appel à des procédures qui permet d'effectuer simultanément le saut à la procédure appelée et la sauvegarde de l'adresse de retour (adresse de l'instruction qui suit le delay slot). L'exemple ci-dessous illustre son utilisation. L'utilisation de l'instruction plus classique *jr* permet de revenir naturellement au programme appelant.

Exemple 7

```
A :   ....
      ....
      jal B      ; Appel à la procédure B, et sauvegarde implicite
                 ; de l'adresse de retour dans le registre R31
      ....       ; Instruction de delay slot
      ....       ; Adresse du retour

B :   ....
      ....
      jr (R31)   ; Retour au programme A
```

Instructions de transfert de contrôle du DEC 21164

Le DEC 21164 ne met pas en œuvre de codes conditions mais peut exécuter une comparaison et un branchement en une seule instruction.

Le même ensemble de comparaisons est utilisé dans les tests des instructions de branchement que dans ceux des instructions *conditional-move*, c'est-à-dire toutes les comparaisons d'une valeur ou d'un registre par rapport à zéro. Cependant, à la différence de l'architecture MIPS, l'Alpha ne

possède pas d'instruction permettant de tester l'égalité de deux registres entre eux puis d'effectuer un branchement suivant le résultat.

Instructions de transfert de contrôle du Power2

Les branchements inconditionnels ont les formats *R-R* et *R-I*. À la différence de l'architecture MIPS, l'architecture Power dispose d'un registre spécial *Link Register* qui lui permet de sauvegarder l'adresse de retour. Les instructions logiques sur les codes conditions telles :*CRAND, CROR, CREQV, ...*, sont spécifiques à IBM.

L'architecture Power offre aussi quelques instructions non usuelles. Ce sont les instructions *BCC* et *BCCL* qui effectuent un branchement conditionnel en fonction du contenu du registre de comptage (*CTR*). Elles sont destinées à optimiser les boucles.

1.6.4 Instructions flottantes

Le MIPS R8000 et le Power2 mettent en œuvre des instructions de multiplication-addition. Les concepteurs du DEC Alpha affirment que ces instructions sont peu générales ; néanmoins elles sont très bien utilisées par les compilateurs. Ce type d'instructions permet d'atteindre une performance de 2 instructions flottantes par cycle et par opérateur flottant sur de nombreux noyaux de calculs (par exemple BLAS 3 [5]). Ce type d'instructions induit par contre un plus grand nombre de ports d'accès sur le banc de registres flottants (3 accès en lecture et un accès en écriture).

Instructions flottantes du MIPS R8000

Le jeu d'instructions du coprocesseur flottant est classique. Les transferts de données s'effectuent entre les registres de l'unité flottante et soit l'unité entière soit la mémoire. Seuls des mots ou des double mots alignés peuvent être transférés.

L'une des grandes nouveautés de ce jeu d'instructions est l'introduction de 4 instructions de multiplication-addition flottantes :

- *MADD (multiply-add)* ;
- *MSUB (multiply-subb)* ;
- *NMADD (negative multiply-add)* ;
- *NMSUB (negative multiply-subb).*

Ces 4 instructions permettent à deux calculs flottants d'être exécutés en une seule instruction. Le résultat intermédiaire n'est pas arrondi avant d'être additionné ou soustrait à une troisième opérande. Bien que cela ne soit pas défini par la norme IEEE 754, le résultat final, qui lui est arrondi selon le mode spécifié par l'instruction, gagne en précision.

Contrairement à son prédécesseur, le R8000 met en œuvre des codes conditions sur les opérations flottantes. Le registre de statut flottant a été étendu à cet effet et compte sur cette architecture 3 bits qui lui permettent de définir 8 codes conditions. L'architecture MIPS IV comprend aussi une nouvelle instruction de comparaison qui affecte les codes conditions. La comparaison est effectuée entre deux registres flottants et le résultat est sauvé dans le code condition spécifié par l'instruction. Cependant cette instruction supporte un certain nombre de restrictions du point de

vue de son séquencement. Effectivement, bien que son résultat soit immédiatement disponible sur l'unité flottante, un délai de un cycle[5] est nécessaire pour propager le code condition au reste du processeur. Les conséquences pour le séquencement du code compilé sont qu'une instruction de comparaison doit être immédiatement suivie par une instruction indépendante entière ou une instruction dépendante flottante, mais ne peut pas être immédiatement suivie par une instruction dépendante affectant une opération entière.

Instructions flottantes du DEC 21164

Le jeu d'instructions flottant du DEC fournit les opérations arithmétiques de base ainsi que des instructions de conversion entre les deux formats supportés (IEEE et VAX).

Le jeu d'instructions comprend le sous-ensemble des instructions *conditional-move* sur les données flottantes.

Le DEC 21164, est le seul des trois processeurs à ne pas disposer d'instruction de racine carrée.

Instructions flottantes du Power2

Le jeu d'instructions du Power2 inclut une instruction de racine carrée qui était émulée par l'appel à une routine spécialisée sur le Power1. L'apport de cette nouvelle instruction permet d'économiser la moitié des cycles par rapport à la version logicielle.

Le jeu d'instructions du Power2 introduit aussi deux instructions de conversion d'une valeur flottante vers une valeur entière.

Rappelons enfin que le jeu d'instructions du Power comporte des instructions de multiplication-addition. Ces instructions sont séquencées en un cycle et apporte, comme dans le cas du R8000, un gain très net de performances sur le calcul flottant.

Comme dans le cas des instructions entières, deux types d'instructions cohabitent pour la mise à jour des codes conditions.

1.6.5 Instructions particulières

Instructions particulières du MIPS R8000

L'une des particularités de l'approche de MIPS est qu'elle fournit un certain niveau d'abstraction de l'architecture en considérant certaines fonctions spécifiques comme des coprocesseurs avec des bancs de registres séparés des autres unités d'exécution. La fonction de contrôle du système des processeurs MIPS est ainsi assurée par le *Coprocesseur 0*. Le coprocesseur de contrôle du système fournit une aide au processeur pour la gestion de la mémoire (MMU) ainsi que pour celle des exceptions. Les instructions de contrôle nécessaires sont spécifiques au processeur et s'exécutent en mode noyau ou superviseur. On peut citer à titre d'exemple :

- les instructions *Data Cache Tag Read/Write* qui permettent de lire et d'écrire une étiquette du cache de données ;
- les instructions *Doubleword Move From/To System Control Coprocessor* qui permettent de lire ou d'écrire l'un des registres du Coprocesseur 0.

5. Conséquence de la compatibilité assurée avec le MIPS R2000

- l'instruction *Probe TLB For Matching Entry* qui permet de tester la table de traduction d'adresse (voir chapitre 6.4.2) dans le but d'une éventuelle correspondance avec une adresse et un numéro de processus.

- les instructions *Read/Write Indexed TLB entry* qui permettent de lire ou d'écrire une entrée de la table de traduction d'adresses.

- l'instruction *Exception Return.*

De plus, dans le but d'aider le développeur à respecter certaines restrictions inhérentes à l'architecture MIPS, le jeu d'instructions du MIPS R8000 inclut une instruction spécifique à une architecture superscalaire qui est *super-scalar no-operation* (SSNOP). L'insertion de cette instruction dans le code provoque une interruption de l'émission de plusieurs instructions simultanément et assure qu'aucune instruction qui suit le SSNOP n'est émise dans le même cycle.

Le jeu d'instructions du R8000 comprend aussi deux nouvelles instructions (*PREF* et *PFETCH*) qui anticipent le chargement des données dans le cache primaire. Elles masquent ainsi le temps d'accès au sous-système mémoire et garantissent un accès plus rapide (*cache hit*) aux données quand elles sont sollicitées au cours de l'exécution. Ce type d'instruction peut être utilisé dans le cas des boucles où l'on est sûr qu'un certain nombre d'itérations vont être effectuées. On anticipe ainsi sur le chargement des opérandes pour les futures itérations.

Remarque : en général ce type d'instruction s'appuie sur un mécanisme matériel (tampon) destiné à recevoir les données préchargées afin de ne pas polluer le cache avec des données qui ne seraient pas utilisées. Ce type de mécanisme ne semble pas avoir été mis en œuvre sur le R8000.

Instructions particulières du DEC Alpha

Le jeu d'instructions du DEC Alpha comprend un certain nombre d'instructions dédiées au support du système d'exploitation.

- *CALLPAL :* cette instruction permet d'accéder à la librairie PAL. L'accès au PALcode n'est pas autorisé tant que toutes les instructions lancées précédemment n'ont pas la garantie de s'achever sans exception.

- *TRAPB :* cette instruction permet d'avoir une gestion précise des exceptions. En effet, elle assure que l'instruction qui suit l'instruction TRAPB n'est lancée que lorsque toutes les instructions lancées avant l'instruction TRAPB se sont exécutées sans déclencher d'exception.

- *FETCH :* cette instruction est similaire à celle développée dans le cas du R8000, et permet de précharger des blocs de données de 512 octets.

- *MEMORY BARRIER :* cette instruction garantit qu'une opération de *load* ou de *store* n'accède pas à la donnée en mémoire tant que toutes les opérations précédentes de même type ne sont pas achevées. Elle est essentiellement utilisée dans le cas de systèmes multiprocesseurs. En l'absence de cette instruction les différentes opérations de lecture/écriture accédant à des données différentes peuvent s'exécuter dans le désordre.

- *RC, RS :* respectivement *Read and Clear* et *Read and set.* Ces instructions sont destinées à préserver l'atomicité des instructions VAX dans le cas de portage d'applications vers du

code Alpha. Elles ne devraient vraisemblablement pas être implémentées dans les prochaines versions.

De plus, le jeu d'instructions du DEC Alpha comprend un ensemble d'instructions (privilégiées et non privilégiées) destinées à supporter les systèmes d'exploitation Alpha VMS et OSF/1. On se reportera à [6] pour plus de détails.

Instructions particulières du Power2

L'architecture Power compte elle aussi un certain nombre d'instructions particulières destinées au support du système d'exploitation. On peut citer à titre d'exemple :

- l'instruction *Supervisor Call* qui appelle le système d'exploitation.
- des instructions de gestion de registres de segments (*Move To/From Segment Register*).
- des instructions de contrôle du cache, du TLB, etc...

L'architecture Power compte également des instructions de synchronisation lors des accès à la mémoire.

- *dcs* (appelée *synchronize* sur le PowerPC). Cette instruction est similaire à l'instruction *Memory Barrier* définie par l'architecture DEC.
- *eieio*: *Enforce In-order Execution of I/O*. Cette instruction est similaire à l'instruction précédente mais est plus adaptée dans un contexte d'entrées/sorties.

1.7 Conclusion

Les jeux d'instructions du DEC 21164 et du MIPS R8000 sont très proches l'un de l'autre et sont typiques d'un jeu d'instructions RISC. Le Power2 s'éloigne un peu de la conception traditionnelle RISC de par la richesse de son jeu d'instructions et la multiplicité des formats de données.

On peut noter de plus quelques points :

- L'absence d'adressage indexé sur le DEC 21164 semble incongru, mais limite le nombre de ports utilisés sur le banc de registres par une instruction. L'introduction de ce mode sur le MIPS R8000 devrait contribuer à améliorer ses performances flottantes.
- L'absence d'adressage pré- ou post-incrémenté sur deux des microprocesseurs étudiés est difficile à comprendre sur des processeurs dont l'un des domaines d'application privilégié est le calcul scientifique : en particulier pour l'accès aux flottants, 2 lectures et une écriture sur le banc de registres entiers suffisent.
- L'absence d'instructions d'accès mémoire sur les octets et les mots de 16 bits sur le DEC Alpha : un choix résolument orienté vers la performance crête, mais certainement un handicap majeur pour de larges domaines d'applications.
- Les instructions de multiplication-addition flottantes sur le R8000 et le Power2.

- Les instructions de type *conditional-move* du DEC Alpha ou du MIPS R8000 permettent d'éviter certains branchements qui sont sans doute les moins prédictibles.
- L'instruction FETCH du MIPS R8000 et du DEC 21164 vont permettre la mise en œuvre de préchargement logiciel des données dans le cache.

Chapitre 2

Architecture : synoptique

2.1 Introduction

Ce chapitre présente globalement la répartition des diverses fonctionnalités sur l'ensemble des composants et introduit certaines notations qui seront reprises au long de cette étude. Nous présentons de plus quelques caractéristiques technologiques.

Le MIPS R8000 est le premier processeur MIPS à mettre en œuvre une architecture superscalaire. Jusqu'à quatre instructions peuvent être séquencées en parallèle à chaque cycle grâce à ses deux unités flottantes, ses deux unités entières et ses deux unités *Load/Store*.

Le Power2 est une extension du Power1. Il reprend le même partitionnement des composants, le même pipeline, les mêmes bus et la même structure mémoire. Les principaux changements concernent l'élargissement des bus, le doublement des unités fonctionnelles (sauf bien entendu, l'unité de séquencement) et la taille des éléments mémoires. Cette nouvelle architecture peut traiter jusqu'à six instructions par cycle. Cependant l'accroissement du nombre de ces unités (ainsi que la capacité des éléments mémoires), et surtout l'ensemble de leur gestion introduit un coût en silicium non négligeable. C'est pourquoi ces deux processeurs sont des multi-composants (trois composants principaux pour le MIPS R8000, huit pour le Power2).

Le DEC 21164 est aussi superscalaire. Avec ses deux unités entières et ses deux unités flottantes, il peut lancer jusqu'à quatre instructions par cycle. À noter également sa fréquence de fonctionnement particulièrement élevée, le synchronisme de lancement des instructions et l'intégration sur un seul composant du processeur, de deux caches primaires et d'un cache secondaire.

2.2 Partitionnement en composants et unités fonctionnelles

2.2.1 Le Power2

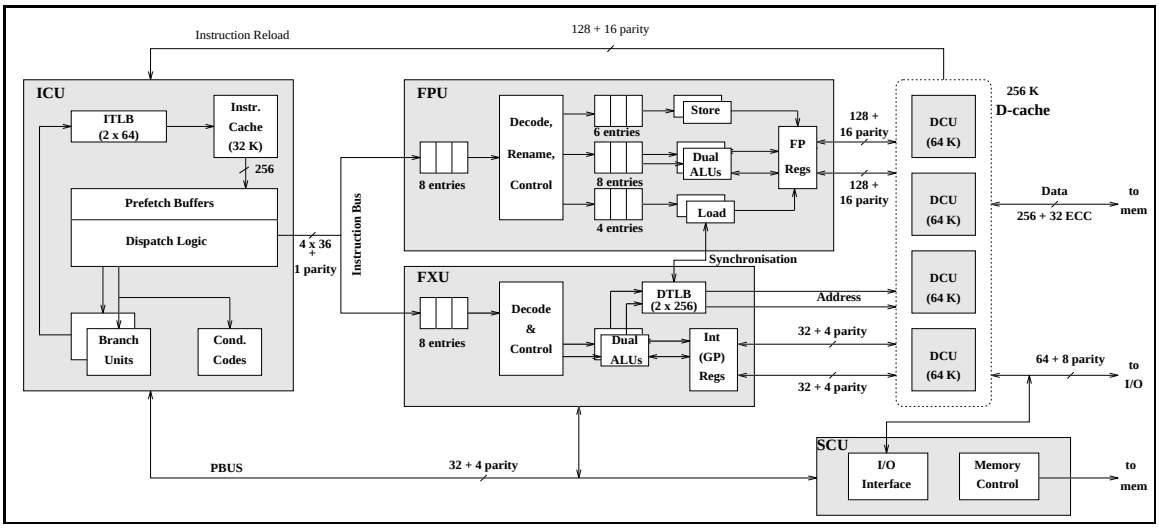


FIG. 2.1 - *Synoptique du POWER 2*

Le Power2 est bâti autour de la nouvelle implémentation de l'architecture multichip Power [1]. Comme on le voit sur la figure 2.1, le processeur est constitué de huit composants partitionnés de manière similaire au Power1.

- *L'Instruction Cache Unit (ICU)* gère le chargement des instructions ainsi que leur émission vers les unités entières et flottantes. Cette unité exécute également les instructions de branchement ainsi que celles portant sur le registre de conditions.

- *La Fixed-Point Unit (FXU)* ou unité entière. Cette unité exécute tous les calculs d'adresses, les opérations arithmétiques entières ainsi que les opérations logiques.

- *La Floating-Point Unit (FPU)* ou unité flottante. Comme son nom l'indique, cette unité est destinée à réaliser l'ensemble des calculs flottants. Elle dispose par ailleurs d'unités d'accès à la mémoire qui lui permettent d'exécuter jusqu'à 2 instructions de lecture et d'écriture par cycle.

- *La Data-Cache unit (DCU)* est un cache de données associatif par ensemble à 4 voies constitué de 4 circuits identiques.
  Ces quatre circuits alimentent l'ensemble des unités précédemment décrites par l'intermédiaire de bus, à raison de deux mots vers la FXU, de deux quadruples-mots (alignés) vers la FPU, et un bus d'instructions de la largeur d'un quadruple-mot (128 bits alignés) vers l'ICU. Deux configurations de cache sont possibles : une configuration de 256 Ko conduit à l'utilisation d'un bus mémoire d'une largeur de huit mots (configuration représentée sur la figure 2.1) alors qu'une configuration à moindre coût de 128 Ko (2 composants DCU au lieu de 4) disposera d'un bus de la largeur d'un quadruple-mot. Un quatrième type de bus d'une largeur de deux mots est destiné au système d'entrées/sorties. Ce bus permet des accès de type DMA

à la mémoire.
Pour fiabiliser les échanges entre les divers composants du circuit, un protocole de détection et de correction d'erreurs de type ECC est implémenté sur l'ensemble de ces bus (détection de deux erreurs, correction d'une seule).

– *La Storage Control Unit (SCU)* est la dernière unité représentée sur cette figure. Son rôle est de gérer les communications entre le processeur et les autres éléments du système (unités d'entrées/sorties, mémoire principale). Elle communique avec l'*ICU* et la *FXU* par l'intermédiaire du *Pbus* et avec les autres éléments par le bus système. Cette unité gère également les interruptions externes ainsi qu'un contrôleur de performances (voir chapitre 8.3).

2.2.2 Le MIPS R8000

La figure 2.2 reprend l'implantation physique du MIPS R8000. Comme dans le cas du Power2, nous avons schématisé le partitionnement des composants de manière à faire ressortir son aspect multi-chip.

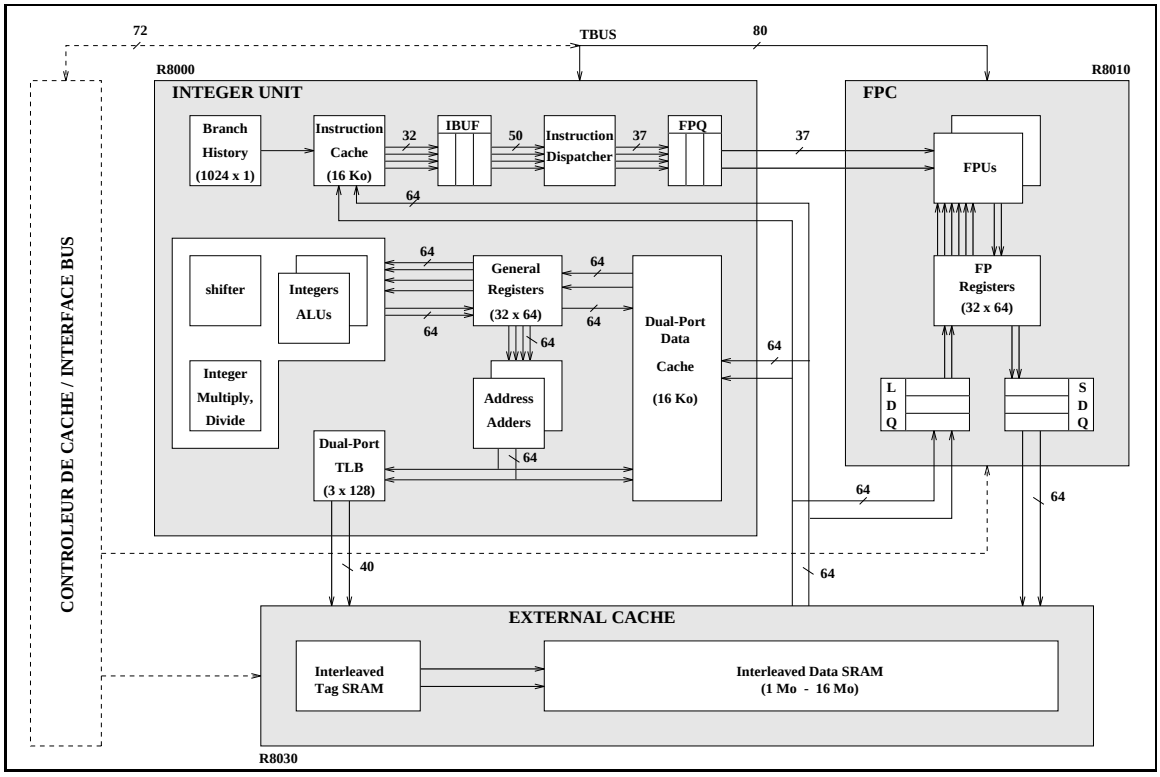


FIG. 2.2 - *Synoptique du MIPS R8000 (plus schématisation du contrôleur de cache)*

Le MIPS R8000 est organisé autour de trois unités logiques. Tout d'abord l'*Integer Unit* ou unité entière. Cette unité gère le chargement et l'émission des instructions et l'ensemble des opérations arithmétiques entières, ce qui sous entend aussi la génération des adresses. Ce composant apparait sous le nom de R8000. Il constitue par la même l'élément le plus complexe de cette architecture.

Autour de ce composant viennent s'ajouter une unité flottante (composant R8010) et un grand cache externe lui-même multi-composant appelé *streaming cache*.

Pour la mise en œuvre matérielle d'un système, il faut de plus un "contrôleur de cache", en fait une unité d'interface avec le reste du système. Ce contrôleur n'est pas fourni dans le jeu de composants proposé par MIPS (c'est pourquoi nous l'avons schématisé en pointillés), mais les fonctionnalités qu'il doit remplir sont définies. À charge pour chaque intégrateur de mettre en œuvre le contrôleur de cache le plus adapté à son application.

Les échanges entre l'opérateur flottant et l'unité entière se font par l'intermédiaire du *TBUS* d'une largeur de 80 bits. Les 8 bits de poids fort sont spécialement dédiés au contrôle des transferts entre l'IU et la FPU. Les échanges avec le "contrôleur de cache" n'utilisent donc que 72 bits du TBUS.

2.2.3 Le DEC 21164

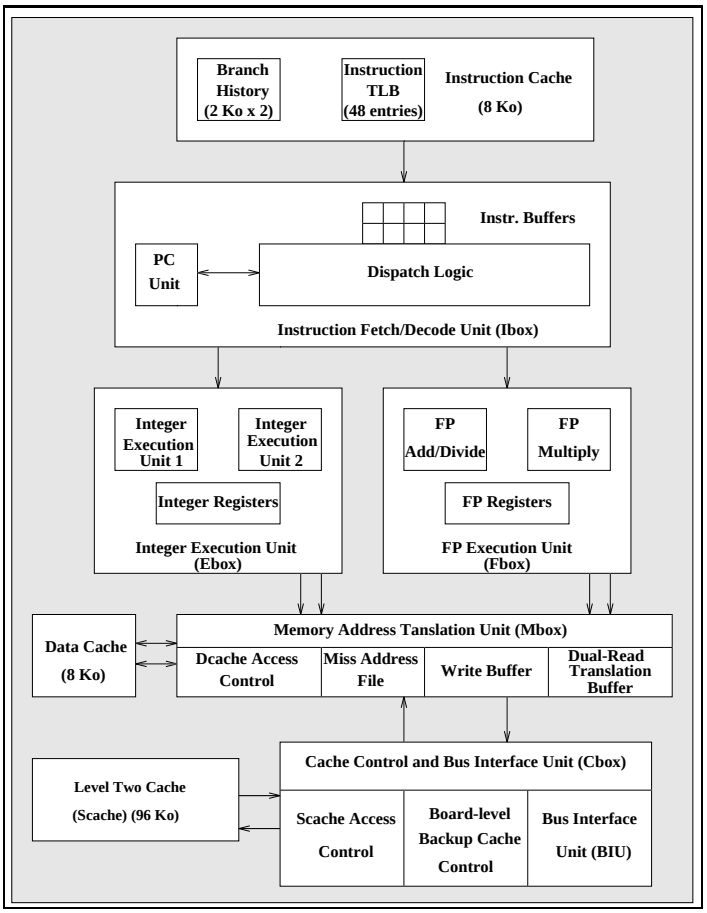


FIG. 2.3 - *Synoptique du DEC 21164*

À la différence des processeurs présentés précédemment, le DEC 21164 est un composant monolithique. Les performances du DEC 21164 améliorent celles du DEC 21064 à travers la mise en œuvre de diverses modifications telles que l'augmentation de la fréquence de l'horloge et de la

taille des caches (gain technologique), l'accroissement du nombre d'instructions émis par cycle et la diminution du temps de latence des pipelines.

Le 21164 est constitué de 5 unités fonctionnelles auxquelles viennent s'ajouter deux niveaux de cache :

- L'*Instruction Fetch and Decode Unit* (*Ibox*) gère et émet les instructions vers l'*Ebox*, la *Mbox* et la *Fbox* (voir ce qui suit). Cette unité intervient directement dans le chargement des instructions et joue un rôle essentiel dans la gestion des interruptions et des exceptions.
- La *Memory Address Translation Unit* (*Mbox*) gère toutes les opérations de lecture/écriture vers la mémoire.
- Le *Cache Control and Bus Interface Unit* (*Cbox*) traite tous les accès émis par la Mbox et implémente toutes les fonctions d'interface entre la mémoire et le reste du système. La Cbox contrôle le cache secondaire (*Scache*) ainsi que le *Board-level Backup Cache (Bcache)* qui constitue un troisième niveau de cache, optionnel et externe au composant.
- L'*Integer Execution Unit* (*Ebox*) traite comme son nom l'indique l'ensemble des calculs arithmétiques entiers.
- La *Floating-Point Unit* (*Fbox*) est responsable de l'exécution des opérations flottantes.
- Le DEC 21164 implémente également trois types de cache :
  - un cache de données primaire (*Dcache*) ;
  - un cache d'instructions primaire (*Icache*) ;
  - un cache secondaire mixte (*Scache*).

  Le 21164 fournit de plus le contrôle nécessaire au rajout d'un troisième niveau de cache externe (*Bcache*).

2.3 Technologie

Nous abordons ici brièvement l'implémentation technologique de chacun des trois processeurs à travers la technologie employée, la surface occupée et la puissance dissipée. Vue l'évolution technologique particulièrement rapide dans le domaine des microprocesseurs, il est important de préciser que ce chapitre a été rédigé en novembre 1994 et qu'il représente notre connaissance des produits à cet instant.

La fréquence d'horloge

Le DEC 21164 a privilégié un temps de cycle court avec une horloge interne à 266 MHz qui atteindra les 300 MHz. Maîtriser ce type de fréquence sur une carte, dans des systèmes "relativement" bon marché (stations de travail, etc...) est aujourd'hui impossible. Les concepteurs ont donc introduit une seconde horloge, l'horloge système. Cette horloge est la seule visible des composants externes au processeur, elle sera un multiple de l'horloge interne ; de 3 à 15 fois dans le cas du DEC ; ce multiple est choisi par l'intégrateur.

Les deux autres microprocesseurs ont une horloge beaucoup plus lente : le Power2 affiche une fréquence de 71,5 MHz qui est à peine plus rapide que l'horloge du Power1 (62,5 MHz) et le MIPS R8000 fonctionne avec une fréquence interne de 75 MHz. Ceci peut surprendre par rapport à son prédécesseur le R4000 qui utilisait la technique du *superpipeline* séquencé à 100 MHz et atteint en 1994 200 MHz. Cependant, le R8000 est un processeur superscalaire de degré 4 et, bien que l'accroissement de sa complexité ait entrainé un temps de cycle plus long, il reste technologiquement supérieur.

L'intégration

Malgré une technologie CMOS à 0,45 micron avec quatre couches de métal, les 23 millions de transistors du Power2 se répartissent sur huit composants. La plupart des liaisons inter-composants sont fournies par l'intermédiaire d'un conditionnement céramique *MCM* (*MultiChip Module*) qui augmente la vitesse de l'horloge de 20 % (selon [7]) et minimise la surface globale (1217 mm^2). Le tableau 2.1 détaille la mise en œuvre physique pour chacune des unités constituant ce processeur. L'annexe B fournit quelques indications sur le conditionnement *multichip* du Power2.

TAB. 2.1 - *Répartition physique des transistors sur le POWER2*

Composant	Transistors (en milliers)		Taille	Broches
	Logique	Mémoire	(mm x mm)	I/O
FXU	583	848	12.7 x 12.7	473
FPU	1001	315	12.7 x 12.7	504
ICU	547	2277	12.7 x 12.7	464
DCU (x4)	1117	16000	12.7 x 12.7	366
SCU	349	-	9.4 x 9.4	276
MCM (total)	3597	19440	1217 mm^2	512

Le MIPS R8000 quant à lui utilise une technologie à 0,7 micron avec trois couches de métal, ce qui représente une aire totale d'à peu près 298 mm^2 pour 3,4 millions de transistors. Ces chiffres ne prennent en compte que deux composants dans la mesure où aucune donnée n'est disponible sur le contrôleur de cache, le *streaming cache*, etc...

Le DEC utilise une densité d'intégration proche de celle du Power2 avec une technologie à 0,5 micron qui lui permet de placer ses 9,3 millions de transistors sur une surface de 298 mm^2 identique à celle du MIPS.

Puissance dissipée

La puissance dissipée par un composant est liée à son degré d'intégration et à sa fréquence d'horloge. Le Power2 qui réunit sous un même conditionnement 8 composants dissipe à peu près 65 watts ce qui représente la valeur la plus importante des trois microprocesseurs étudiés. Le MIPS R8000 et le DEC, alimentés tous les deux sous 3,3 V, dissipent respectivement 30 W (pour deux composants) et 40 watts.

Le coût

TAB. 2.2 - *Estimation du coût de fabrication des microprocesseurs*

	POWER 2	MIPS R8000	DEC 21164
Coût	\$ 1050 pour les huit composants \$ 540 pour l'ICU, la FXU et FPU	\$ 850	\$ 430 (300 MHz)

De nombreux critères interviennent sur le coût de fabrication d'un composant tels que la technologie employée, le conditionnement, les tests, ... Il s'avère que le caractère multichip d'un processeur allourdit de manière non négligeable son coût à cause des communications inter-composants qui nécessitent un grand nombre de broches (bus et signaux de contrôle). Le tableau 2.2 présente les coûts estimés de fabrication pour chacun des processeurs (selon le modèle développé dans *Microprocessor Report* [8], [9], [10], [11]). Bien entendu, ce tableau est purement théorique et ne tient pas compte du contexte dans lequel ces processeurs seront implémentés, ni du coût de développement de ces processeurs.

Chapitre 3

Séquencement et exécution des instructions

3.1 Introduction

Avec l'allongement des pipelines et la multiplication des instructions émises en parallèle, l'unité responsable de les séquencer est devenue l'un des éléments essentiels de l'architecture d'un microprocesseur. Son rôle est de décoder les instructions et de lancer leur exécution à travers l'ensemble des unités fonctionnelles mises à sa disposition. Elle a par conséquent la responsabilité de fournir à l'utilisateur le résultat attendu par l'exécution séquentielle de son programme, et assure pour cela la gestion des nombreux mécanismes matériels lui permettant d'optimiser la durée de son exécution. Elle a aussi à sa charge la prise en compte des diverses interruptions qui provoquent des ruptures de séquence dans l'exécution de ce programme.

Cet élément est particulièment déterminant dans le cas d'une architecture superscalaire où, à chaque cycle, de multiples instructions sont exécutées simultanément, ce qui accroît considérablement la complexité des mécanismes mis en place. Il est de plus capital de fournir un flot continu et suffisant d'instructions pour bénéficier pleinement des possibilités offertes par ce type d'architecture.

C'est l'ensemble de ces mécanimes que nous allons aborder dans ce chapitre. Dans un premier temps nous étudierons les pipelines de chacun des processeurs, puis nous reprendrons de manière plus détaillée l'ensemble des mécanismes de décodage et d'émission des instructions ainsi que les solutions apportées à la résolution des conflits inhérents à ce type d'architecture.

3.2 Partitionnement du pipeline

Le pipeline est une implémentation technique qui permet le recouvrement de l'exécution de plusieurs instructions. Il constitue un concept majeur mis en œuvre dans les processeurs pour accélèrer l'exécution des programmes. Il est caractérisé par sa profondeur (nombre d'étages le constituant), son temps de cycle (durée requise par l'étage le plus lent) et, dans le cas des processeurs superscalaires, par le nombre d'instructions qu'il est susceptible de traiter simultanément. Le pipeline augmente le débit d'exécution des instructions en exploitant le recouvrement des différentes phases de leur exécution, mais ne diminue pas le temps de traitement individuel de chacune.

Nous présentons ici la structure générale des pipelines implantés dans les 3 processeurs. Ceux-ci offrent un éventail intéressant des diverses mises en œuvre que l'on trouve actuellement.

3.2.1 Le DEC 21164

Pour atteindre une horloge très élevée, le pipeline du DEC 21164 a été divisé en un nombre important d'étages. La technique dite du *superpipeline*[1] a été appliquée. C'est à dire que les étages critiques ont été divisés en deux, voire en trois.

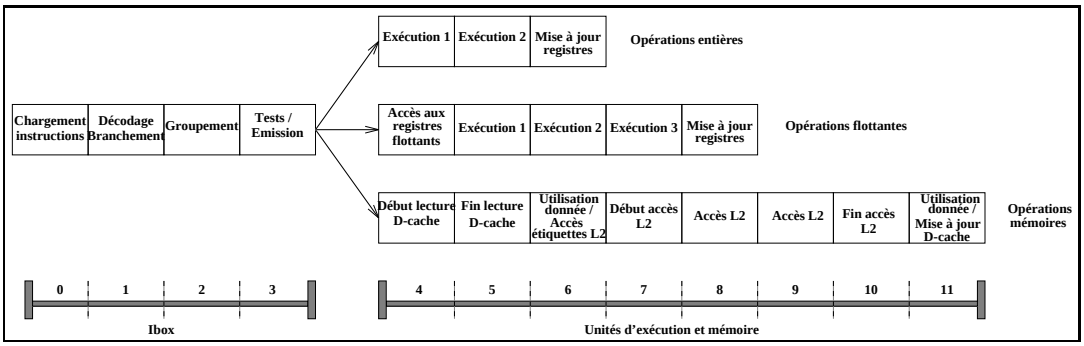


FIG. 3.1 - *Pipelines du DEC 21164*

Le pipeline du DEC est relativement semblable à celui de son prédécesseur le 21064 (voir [2]). Un pipeline de sept étages est utilisé pour les opérations entières et les instructions de load/store, et un pipeline de neuf étages pour les opérations flottantes. Les quatres premiers étages sont exécutés par l'unité de chargement et de décodage des instructions, l'*Ibox*. Cette unité, comme nous le verrons plus loin, intervient directement dans le chargement des instructions à partir du cache d'instructions (ou *Icache*) et dans leur décodage. Elle joue également un rôle essentiel dans la gestion des interruptions et des exceptions. Ces quatre étages sont communs à toutes les instructions. La suite du traitement est différenciée selon le type de l'instruction, celui-ci entrainant l'exécution dans l'une des trois unités : l'*Ebox* (l'unité entière), la *Fbox* (l'unité flottante) et la *Mbox* (unité de gestion des accès à la mémoire) (voir figure 2.3). Une distinction est faite au niveau des divers étages du pipeline. Les quatre premiers étages sont dits *statiques* dans la mesure où des instructions peuvent rester bloquées à ce niveau en raison de conflits de ressources, de dépendances de données, etc... Alors, que les étages suivants sont dits *dynamiques*, c'est à dire qu'à chaque cycle, l'exécution de toute instruction avance d'une étape et qu'aucun gel du pipeline ne peut empêcher la progression d'une instruction dans ces étages.

La figure 3.1 reprend l'exécution des instructions dans les 3 types de pipelines présents sur le DEC. L'étage d'émission a la responsabilité de garantir que tous les conflits de ressources sont résolus avant de permettre à l'instruction de continuer dans les étages dynamiques. Quatre instructions peuvent être lancées à la fois. Le seul moyen de stopper une instruction après sa sortie des étages statiques est une condition d'abandon (exception, interruption, prédiction de branchement erronée). Les trois étages nécessaires au décodage et à l'émission des instructions témoignent de la complexité de la gestion des conflits à la fréquence de fonctionnement imposée par le constructeur.

1. Notion introduite par MIPS pour son R4000

3.2.2 Le MIPS R8000

Le pipeline entier du R8000 diffère largement des structures de pipeline jusqu'ici employées par MIPS pour le R4000 (voir [2]) et le R3000 (voir [1]). Comme on le voit sur la figure 3.2, le MIPS R8000 utilise un pipeline à cinq étages :

1. *Fetch* : accès au cache d'instructions et au cache de prédiction de branchement.
2. *Decode* : décodage et émission des instructions. C'est à ce niveau que les données sont lues dans le banc de registres et que les tests de dépendance sont effectués.
3. *Address* : génération des adresses requises par les accès mémoires.
4. *Execute* : accès au cache de données et à la table de traduction d'adresses (TLB), exécution des opérations arithmétiques, résolution des branchements et gestion des exceptions.
5. *Writeback* : mise à jour du banc de registres.

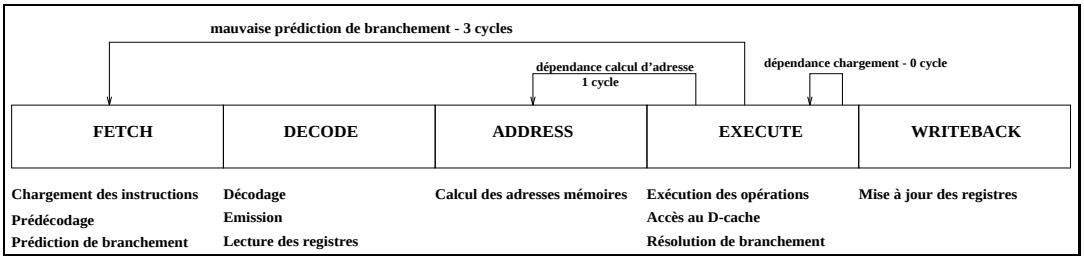


FIG. 3.2 - *Pipeline entier du MIPS R8000*

L'unité entière dispose de deux générateurs d'adresses et de deux ALUs qui peuvent être activés simultanément. Jusqu'à quatre opérations peuvent avoir lieu simultanément au sein de cette unité.

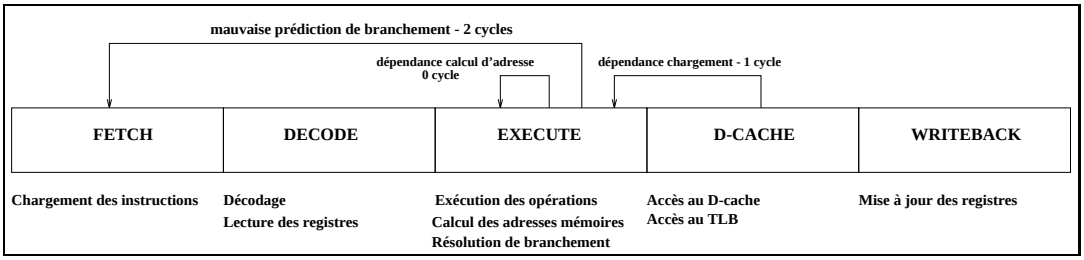


FIG. 3.3 - *Pipeline entier du MIPS R3000*

L'organisation de ce pipeline diffère des pipelines RISC traditionnels tel celui du MIPS R3000 (confére figure 3.3) ou des premiers Sparc. Sur un processeur scalaire tel que le R3000, le compilateur trouve en général une instruction indépendante du *load* à insérer entre le *load* et son utilisation. En utilisant cette structure de pipeline, dans le cas du MIPS R8000, le compilateur aurait dû insérer de 4 à 7 instructions indépendantes pour conserver le plein usage du pipeline. Cette situation a été jugée peu probable par les concepteurs qui ont préféré inverser les étages de calcul d'adresse et d'exécution (comme dans le Pentium).

Retarder l'étage d'exécution dans le pipeline, crée par contre une nouvelle pénalité lorsqu'une instruction de *load* ou de *store* requiert pour le calcul d'adresse des opérandes calculées par des instructions amonts ; afin déviter toute pénalité, le compilateur se doit d'insérer de 4 à 7 instructions indépendantes entre le calcul des opérandes d'adressage et le load/store. Selon les concepteurs de MIPS, les dépendances de *load-use* apparaissent plus fréquemment dans les programmes que les dépendances de calcul d'adresse, particulièrement dans les codes entiers avec beaucoup de branchements. Le nouveau mode d'adressage *registre + registre*, pour les opérations de load/store flottantes devrait permettre de réduire le besoin de précalculer les adresses.

Un deuxième inconvénient à cette forme de pipeline est qu'elle ajoute un cycle de pénalité sur les branchements mal prédits, le résultat du branchement n'étant connu qu'au quatrième cycle. Dans le cas d'une mauvaise prédiction, trois cycles sont alors nécessaires pour recharger le pipeline (soit une perte de 12 instructions).

Des simulations réalisées au sein du projet CAPS [12] ont montré qu'un simple réordonnancement du code assembleur ne permet pas de tirer partie de cette particularité du pipeline. Une refonte complète du processus de génération de code est nécessaire pour obtenir des performances plus élevées sur cette structure de pipeline par rapport à la structure traditionnelle.

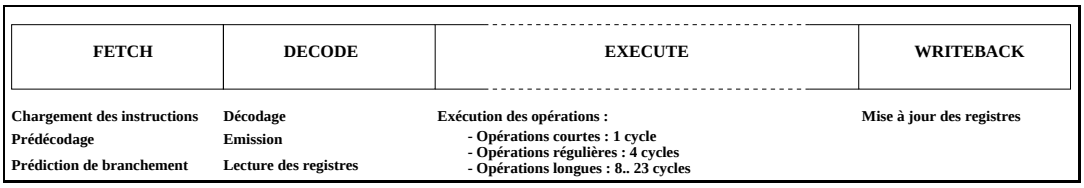


FIG. 3.4 - *Pipeline flottant du MIPS R8000*

Le pipeline flottant du R8000 (figure 3.4) est totalement découplé du pipeline entier. Seuls les étages de chargement et de décodage sont communs. Ce découplage permet de masquer le temps d'accès au cache externe qui s'effectue en 5 cycles et la latence des opérations flottantes. Le R8010 FPU implémente trois types d'opérations arithmétiques appelés *court, régulier* et *long*. Pour exécuter chacun de ces trois types de base, un certain nombre de cycles est nécessaire au sein de chacune des unités fonctionnelles. Chaque unité d'exécution est complètement pipelinée et peut recevoir une nouvelle opération à chaque cycle (voir chapitre 4.3 pour plus de détails).

3.2.3 Le Power2

Le pipeline entier est décomposé en 5 étages (6 dans le cas des opérations mémoires). L'étage supplémentaire par rapport à un pipeline RISC traditionnel correspond à la phase de prédécodage des instructions qui permet de séparer les instructions entières et flottantes des instructions portant sur le registre de condition et des instructions de branchement. Cette particularité permet d'anticiper les branchements et de réduire les cycles d'attente induits.

Le Power2 a conservé les mêmes pipelines que ceux implémentés sur le Power1 (voir [1]). Cependant les unités d'exécution ajoutées à cette nouvelle architecture lui permettent de traiter simultanément un plus grand nombre d'instructions. La figure 3.5 détaille les pipelines de cette architecture.

La phase *IF* (pour *Instruction Fetch*) effectue le chargement de 8 instructions à partir du cache d'instructions dans l'un des deux tampons d'instruction mis en œuvre à cet effet sur le Power2.

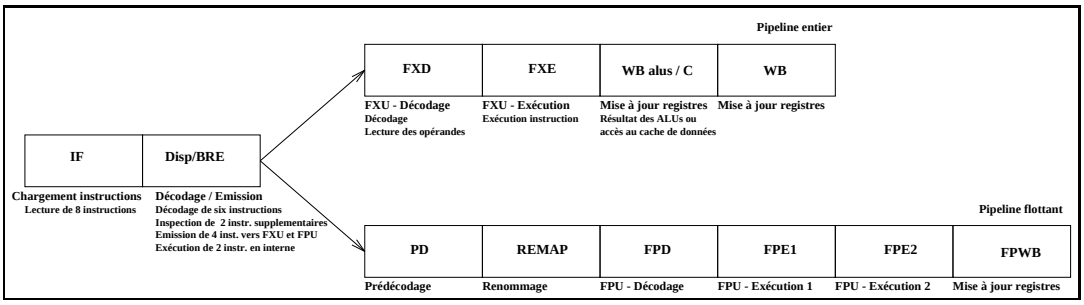


FIG. 3.5 - *Pipelines du POWER2*

L'étage suivant (*Disp/BRE*) traite six instructions simultanément. Il effectue un prédécodage partiel des instructions tout en séparant les instructions entières et flottantes des instructions traitées en interne (dans l'ICU). Deux unités de décodage supplémentaires permettent d'inspecter la septième et huitième instruction pour détecter un éventuel branchement. Cet étage se termine par l'émission possible de 4 instructions vers les unités entière et flottante et par l'exécution de deux instructions en interne (soit deux branchements, soit un branchement et une instruction portant sur le registre de condition).

L'étage d'émission ne traite absolument pas les dépendances de données qui sont gérées directement par les unités fonctionnelles réceptrices des instructions. Cette différence avec les autres processeurs qui offrent une gestion centralisée permet d'augmenter le parallèlisme des instructions.

Contrairement au Power1, les unités entières ne prennent sur l'Ibus que les instructions qui les concernent. Par ailleurs pour absorber le flot d'instructions, la taille des buffers implantés sur les unités d'exécution a été systématiquement doublée. À noter également le *bypass* de l'étage d'accès à la mémoire pour les opérations entières de registre à registre qui permet d'économiser un cycle.

3.3 Mécanismes de chargement et de séquencement des instructions

3.3.1 Introduction

Ces mécanismes sont particulièrement importants dans le cas des architectures superscalaires :

- Vu le nombre d'instructions traitées par cycle, il est nécessaire de maintenir un flot soutenu d'accès au cache d'instructions. Ce flot doit être suffisant pour exploiter au mieux la totalité des ressources mises en œuvre. Cela pose le problème du chargement des instructions ainsi que l'anticipation des branchements qui viennent rompre le flot séquentiel d'instructions.

- Une deuxième difficulté sur les architectures superscalaires, concerne la gestion des dépendances de données. Le compilateur peut permettre une exploitation optimale des ressources grâce à l'utilisation de techniques de réordonnancement de code dès lors que la séquence exécutée est prévisible à la compilation ; mais une mise en œuvre matérielle de la gestion des conflits et des dépendances de données est nécessaire pour assurer une exécution correcte des séquences s'enchainant dynamiquement à l'exécution.

3.3.2 Le DEC 21164

Le DEC 21164 présente l'architecture la plus simple en ce qui concerne le décodage et l'émission des instructions. Cette fonction est assurée par l'*Ibox* (pour *Instruction Fetch and Decode Unit*). La fréquence d'horloge très élevée du DEC 21164 limite la complexité des actions qui peuvent être exécutées en un seul cycle. Ceci conduit à des choix qui limitent les possibilités de lancer les instructions en parallèles :

- groupe d'instructions alignées ;
- lancement dans l'ordre.

L'Ibox assure le chargement et le lancement de quatre instructions en parallèle (quand ceci est possible). Pour cela, cette unité dispose de mécanismes chargés de l'alimenter : un extracteur d'instructions, un tampon de préchargement à quatre entrées appelé tampon de pré-remplissage et deux tampons d'instructions destinés à accueillir les données allant être émises.

La figure 3.6 schématise cycle par cycle le fonctionnement des étages de l'Ibox :

Etage 1 - Lecture du cache d'instructions et de l'étiquette associée : 4 instructions alignées sur un quadruple-mot sont lues en parallèle.

Etage 2 - Les 4 instructions sont placées dans l'un des deux tampons d'instructions. La validité de leur étiquette est testée. Dans le même temps la présence d'un éventuel branchement dans le groupe est testée afin de déterminer au plus tôt la prochaine adresse de chargement. Le multiplexeur permet de sélectionner l'un ou l'autre des tampons dès que le tampon courant est vide.

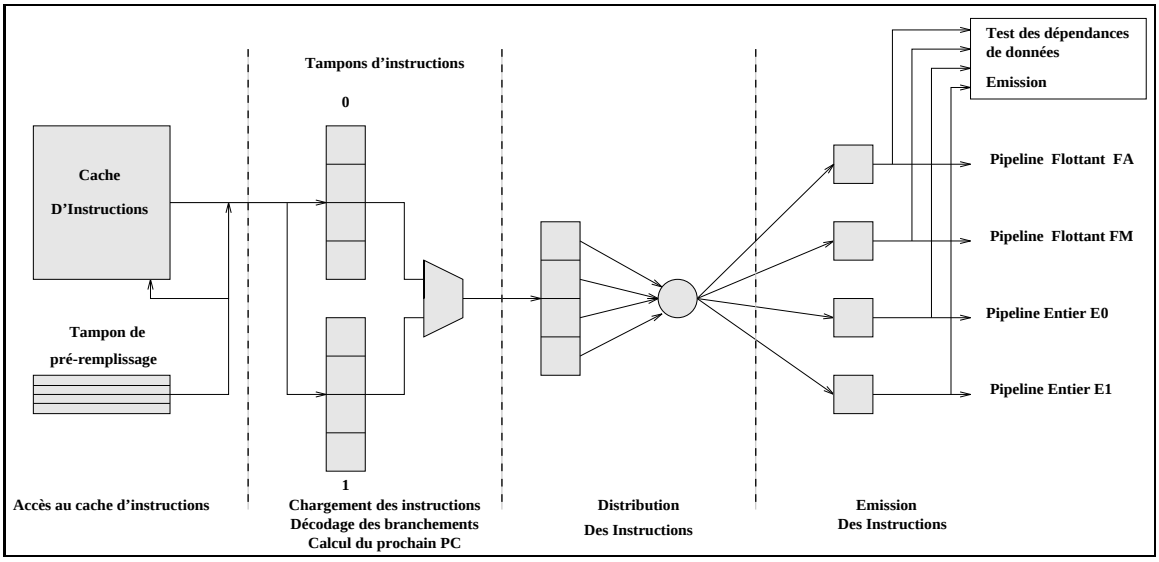


FIG. 3.6 - *Pipeline d'émission des instructions du DEC 21164*

Etage 3 - L'Ibox décode les quatre instructions en parallèle. Un mécanisme appelé *slotting function* détecte et résoud tous les conflits statiques de ressources. Cette fonction regarde vers lequel des

quatre pipelines (E0 : pipeline entier, E1 : second pipeline entier, FA : pipeline flottant, FM : pipeline flottant de multiplication, nous reviendrons plus loin sur les fonctionnalités de chacun des pipelines) les instructions seront émises en fonction de diverses règles d'ordonnancement. Elle suit l'algorithme suivant :

En commençant à partir de la première instruction valide du groupe (adresse la plus basse). Si il s'agit d'une instruction qui peut être émise dans l'un des deux pipelines entiers, la fonction l'assigne à E0, sauf si l'une des règles suivantes est vérifiée, elle l'assigne alors à E1 :

- *E0 n'est pas libre et E1 l'est,*
- *la prochaine instruction du groupe ne peut être émise que dans E0.*

Si l'instruction courante est l'une de celles qui peuvent être émise dans FA ou FM, l'assigner alors à FA à moins que FA ne soit pas libre. Si l'instruction ne peut être émise que dans FA (respectivement FM), elle doit être assignée à FA (respectivement FM).
Le pipeline sélectionné pour cette opération est alors marqué comme pris et le processus reprend avec la prochaine instruction séquentielle. Si une instruction ne peut pas être allouée car le pipeline d'exécution qu'elle pourrait utiliser est déjà pris, alors le processus s'arrête, de même si ses opérandes ne sont pas disponibles.

L'Ibox ne gère que des groupes d'instructions naturellement alignées et n'émet pas les instructions dans le désordre. Si une instruction ne dispose pas des ressources qu'elle requiert, elle bloque le flot même si une instruction postérieure pourrait être émise. Si seules les trois premières instructions sont émises dans le même cycle, la quatrième sera émise au cycle suivant car l'extracteur qui fournit les données ne passe pas au tampon suivant tant que le tampon courant n'est pas totalement vidé. Le compilateur peut jouer un rôle particulièrement important à ce niveau pour l'obtention de performance en insérant des instructions de type *NOP (No Operation)* dans le flot d'instructions.

Etage 4 - Test des opérandes de chaque instruction pour voir si elles sont valides et disponibles. Vérification des conflits de données de type WAW. À la fin de cet étage, toutes les opérandes sources doivent être disponibles pour que les instructions soient émises vers leur pipeline respectif.

À noter qu'un blocage dans l'un des étages (2, 3, 4) bloque tous les étages amonts.

Nous verrons dans le chapitre 5 les mécanismes mis en œuvre pour minimiser les défauts de cache d'instructions et les pénalités liées à ces défauts.

3.3.3 Le MIPS R8000

Le MIPS R8000 privilégie également l'émission de plusieurs instructions en parallèle, mais à la différence du DEC 21164, des mécanismes matériels plus complexes sont mis en œuvre au détriment d'une fréquence interne plus basse.

Le chargement des instructions constitue un chemin critique pour ce type d'architecture grande consommatrice d'instructions. Les instructions sont chargées au rythme de 4 instructions de 32 bits alignées en mémoire à chaque cycle (du moins tant que le tampon d'instructions destinataire n'est pas plein). Elles sont prédécodées avant d'être placées dans un tampon d'instructions à six entrées (soit 24 instructions). Le but du prédécodage est de réduire la durée de l'étage de décodage du

pipeline. Dix-huit bits de caractérisation sont ajoutés aux 32 bits d'instructions originaux. Ces bits sont utilisés au cours des deux premiers étages du pipeline, après quoi la plupart sont abandonnés.

Suite à ce prédécodage, un mécanisme appelé le *X-BAR* détermine quelles instructions émettre en fonction des ressources disponibles. Ce traitement appelé *Resource Modeling* s'appuit en fait sur une technique de *scoreboarding*. Le X-BAR contrôle le statut de chaque unité d'exécution et détermine les interdépendances entre les quatre premières instructions disponibles (sans tenir compte de leur alignement). Les instructions sont toujours émises dans l'ordre et une instruction ne disposant pas des ressources qu'elle nécessite bloque les instructions postérieures. La figure 3.7 illustre sur quatre cycles le fonctionnement de cette logique.

1. Au premier cycle, les instructions *A, B, C, D* sont présentées à la logique d'émission. Dans l'exemple présenté, seules *A* et *B* sont envoyées vers les unités fonctionnelles d'exécution. Simultanément *E* et *F* sont lues à partir du premier étage de la queue, passent à travers le multiplexeur et sont stockées dans l'unité d'émission de manière à ce qu'au prochain cycle la logique d'émission dispose à nouveau de 4 instructions à émettre.

2. Le pointeur désigne *C* comme la première instruction à émettre (émission dans l'ordre). *C, D, E* sont alors envoyées vers l'étage d'exécution, libérant de la place dans le registre de dispatch pour les instructions *G, H, I.*

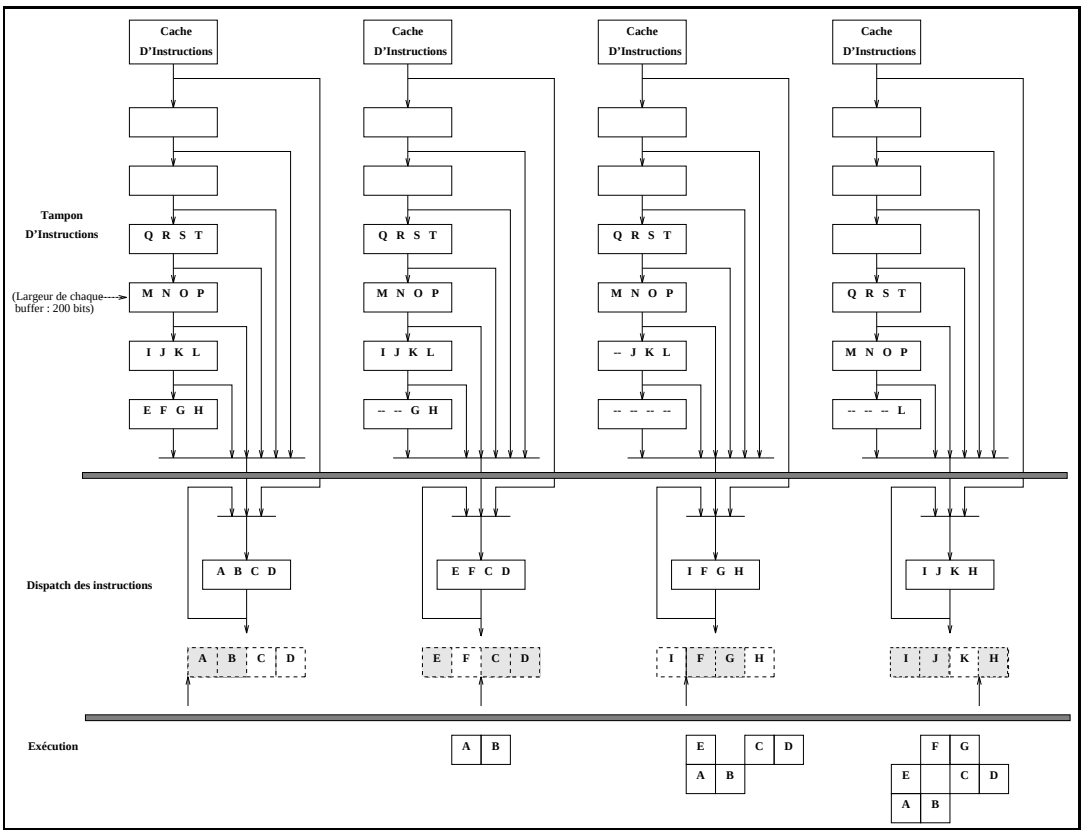


FIG. 3.7 - *Mécanisme d'émission des instructions du MIPS R8000*

3. De ce fait le premier étage de la queue se trouve alors vide. En même temps que la logique d'émission continue d'envoyer les instructions (F et G) vers les unités d'exécution, le tampon d'instruction agit comme un registre à décalage de manière à combler le premier étage de la file et à libérer ainsi une entrée de celle-ci.

4. Au quatrième cycle, les instructions *H, I, J* sont envoyées vers les unités fonctionnelles alors que les instructions *L, M, N* viendont remplir les emplacements laissées par celles-ci. Il est à noter que nous n'avons pas schématisé sur notre schéma l'arrivée permanente de nouvelles instructions en queue de file.

Quatre instructions parmi deux instructions entières, deux opérations mémoires, 4 instructions flottantes peuvent être émises par cycle. Le fonctionnement des différentes unités fonctionnelles sera décrit plus loin. Alors que les opérations entières et les générations d'adresse sont exécutées immédiatement, les instructions flottantes sont rangées dans une file d'attente appelée *Floating-Point Queue* (*FPQ*). Cette file, de 15 entrées, évite l'encombrement du mécanisme de dispatch par des instructions flottantes en attente d'opérandes et découple l'unité flottante de l'unité entière, ceci permet de masquer la latence d'accès au cache. La gestion des dépendances internes à l'unité flottante est traitée au sein même de celle-ci.

N.B. : pour les instructions flottantes, la disponibilité de ressources dont nous parlons plus haut est la place dans la *floating-point queue* et non la disponibilité des opérandes.

Le MIPS R8000 présente une unité de décodage plus élaborée que celle du DEC. La structure de son tampon d'instructions lui permet de ne pas tenir compte de l'alignement des données et évite ainsi de recourir au compilateur pour grouper les instructions en insérant des *NOPs* qui augmentent le taux de défauts dans le cache[2]. De plus, le découplage des instructions entières et flottantes offre la possibilité d'un séquencement dans le désordre et permet de masquer le temps d'accès au cache externe, optimisant ainsi l'usage de celui-ci.

3.3.4 Le Power2

Le Power2 présente l'unité de lancement des instructions la plus complexe des trois processeurs étudiés avec l'émission possible de six instructions à la fois. Comme dans l'architecture Power, le Power2 répartit l'exécution des instructions sur trois composants : l'ICU (unité de contrôle), la FXU (unité entière) et la FPU (unité flottante). L'ICU gère le chargement des instructions, exécute les branchements et les instructions manipulant le registre de conditions, et enfin, émet les instructions entières et flottantes vers la FXU et la FPU.

Pour charger les instructions, l'ICU doit générer les adresses, les traduire, et accéder au cache. Il est organisé de manière à délivrer huit mots à chaque cycle. Six bits de prédécodage sont calculés par l'ICU lorsque l'instruction est chargée dans le cache. Ces bits indiquent le type de l'instruction (flottante, entière, branchement ou code condition), et, comme dans le cas du MIPS R8000, assistent l'opération d'émission des instructions. Contrairement à la première implémentation du Power, où le cache d'instructions n'était pas accessible lors du chargement d'une ligne, le Power2 permet des accès permanents par le biais d'un *tampon de rechargement du cache* qui contient une ligne d'information.

2. On est loin du pipeline entièrement géré par logiciel du MIPS R3000

La structure du cache lui permet de ne pas tenir compte de l'alignement des instructions et permet des accès à travers des frontières de lignes dès lors qu'elles appartiennent à la même page (comme sur le Pentium).

Le Power2 implémente deux tampons d'instructions remplis à partir du cache par un mécanisme de préchargement des données. Le tampon séquentiel (16 instructions) conserve les instructions préchargées à partir du flot d'instructions en séquence du PC courant. La logique de chargement essaye de maintenir au minimum six instructions dans ce tampon. Le second tampon, appelé *tampon cible* est destiné à recevoir les données préchargées à partir d'une adresse de branchement. La taille de ce tampon est de huit instructions. Quand l'ICU rencontre un branchement conditionnel dépendant d'une instruction dont l'exécution n'est pas encore terminée (le branchement est alors dit non résolu), la logique de chargement utilise le tampon séquentiel pour charger les données à partir du flot d'instructions séquentielles, et utilise le tampon cible pour précharger des données provenant du branchement. L'ICU émet les instructions de l'un ou l'autre tampon suivant deux modes distincts de fonctionnement appelés *mode d'émission séquentielle* et *mode d'émission cible* selon l'origine des données émises. En mode d'émission séquentielle, l'ICU décode 6 instructions. Le décodage permet d'identifier le type de chacune des instructions. L'ICU peut alors exécuter deux instructions en interne (soit deux branchements, soit un branchement et une instruction portant sur le registre de conditions) et en émettre 4 vers les unités entière et flottante. Les six instructions sont évaluées en parallèle par la logique d'émission. De plus, 2 instructions supplémentaires sont inspectées pour la détection des branchements. Si un branchement est détecté dans la fenêtre des 8 instructions inspectées et si l'adresse de la cible est disponible, le tampon cible est alors préchargé.

En mode d'émission cible, l'ICU décode et n'émet que quatre instructions à la fois. De plus aucune d'entre elles ne doit être une instruction traitée en interne (branchement, manipulation de code conditions). Dans le cas contraire, le contenu du tampon cible est alors vidé vers le tampon séquentiel après résolution du branchement.

L'ICU envoie les instructions en séquence de manière conditionnelle dans le cas d'un branchement non résolu. Quand le branchement est résolu, les instructions envoyées de manière conditionnelle sont exécutées (le branchement n'est pas pris), ou abandonnées (le branchement est pris). Un seul niveau d'instructions conditionnelles peut être émis.

La logique d'émission présente sur l'ICU ne détermine pas les dépendances de données entre les instructions. Quatre instructions peuvent être émises dans l'ordre vers les unités entière et flottante sur le bus d'instructions. À chaque instruction, est associé un bit de validité et trois bits de prédécodage. Le bit de validité de l'instruction peut-être modifié à tout moment selon que l'instruction ait été émise de manière conditionnelle ou qu'une interruption soit apparue. Les unités entière et flottante disposent toutes les deux d'un tampon d'instructions FIFO à 8 entrées. Ces tampons permettent de désynchroniser les unités de calcul de l'unité de séquencement. Chacun de ces tampons peut accepter jusqu'à 4 instructions par cycle.

3.3.5 Les conflits de contrôle

La gestion des conflits de contrôle est vitale pour une architecture et est étroitement liée aux mécanismes de chargement des instructions. Les branchements constituent généralement de 15 à 30 % des instructions.

Lors d'un branchement conditionnel, le résultat de la condition de saut est évalué généralement au niveau de l'étage d'exécution, soit relativement tard dans le pipeline. Si ce résultat n'est pas

anticipé pour le chargement des instructions, c'est 4 à 5 cycles qui peuvent être perdus, soit dans le cas des processeurs superscalaires une vingtaine d'instructions.

Les trois processeurs étudiés mettent en œuvre une logique d'anticipation de branchement et font appel à des techniques différentes.

Logique de prédiction sur le MIPS R8000 Le tampon d'instructions doit charger 4 instructions à chaque cycle pour maintenir un taux d'émission de 4 instructions par cycle. Le MIPS R8000 implémente pour cela un cache d'adresses de branchement qui est amené à travailler en conjonction avec le cache d'instructions. La mise en œuvre d'un cache d'adresses de branchement résulte d'un compromis entre la taille qu'il occupe sur la puce et l'efficacité dont il doit faire preuve pour prédire les ruptures de séquences. Sur le MIPS R8000, il est organisé comme une table de 1024 entrées, une entrée pour chaque ligne de 4 mots du cache d'instructions. Il implémente un schéma de prédiction à un bit. Le schéma 3.8 reprend son organisation et son fonctionnement.

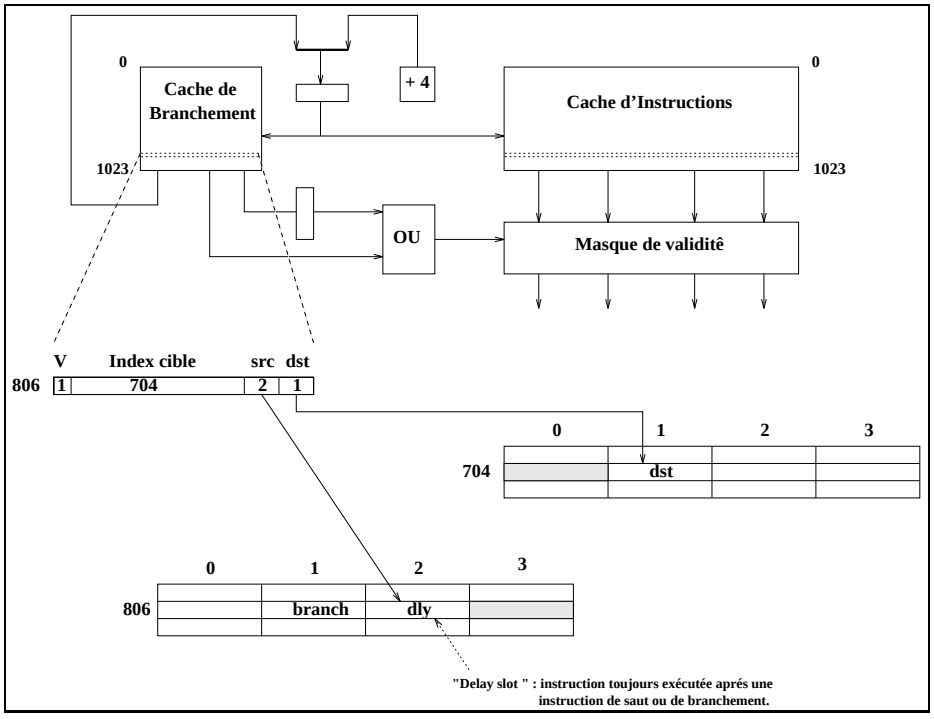


FIG. 3.8 - *Cache d'adresses de branchement du MIPS R8000*

Le cache d'instructions et le cache d'adresses de branchement sont accédés en parallèle lors de l'étage de chargement du pipeline. Chaque entrée du cache d'adresses de branchement a un bit de prédiction associé. Si celui-ci est valide, il indique une discontinuité (un saut ou un branchement) dans la ligne du cache d'instruction à laquelle il est associé. Dès lors les 10 bits du champ d'adresse cible sont directement chargés dans le PC (au lieu d'incrémenter celui-ci de 4 instructions). Les deux champs supplémentaires (*src* et *dst*) précisent les positions de la dernière instruction dans la ligne source et la première instruction dans la ligne destination.

À noter que pour maintenir la compatibilité binaire avec les générations précédentes, le MIPS R8000 exécute systématiquement l'instruction qui suit le saut pendant que l'adresse cible est chargée.

La validité de la prédiction n'est pas connue avant l'étage d'exécution du pipeline (soit 3 cycles après le chargement des instructions dans le cas du MIPS R8000). Dans le cas d'une mauvaise prédiction, 3 cycles sont perdus.

Ce mécanisme ne permet pas la prédiction de plusieurs branchements dans la même ligne d'instructions mais permet d'emboîter plusieurs prédictions consécutives. En cas de présence de deux branchements successifs dans la même ligne, les performances risquent d'être dégradées. Aussi, le compilateur doit s'efforcer d'insérer dans la même ligne de quatre mots des séquences de code exécutables inconditionnellement.

Logique de prédiction sur le DEC 21164 Le DEC 21164 implémente lui aussi une table de prédiction de branchements mais basée sur une prédiction à deux bits selon l'algorithme de Smith. Cette table est accédée en parallèle avec le cache d'instructions au niveau de l'étage de chargement du pipeline. Deux bits sont associés à chaque quadruple-mot. Comme on l'a vu précédemment, les instructions sont décodées au niveau de l'étage suivant et l'adresse cible d'un éventuel branchement est connu à la fin de cet étage. Si les bits d'historiques, conformément à l'algorithme rappelé figure 3.9, indiquent que le branchement doit être pris, le chargement est alors redirigé vers cette nouvelle adresse.

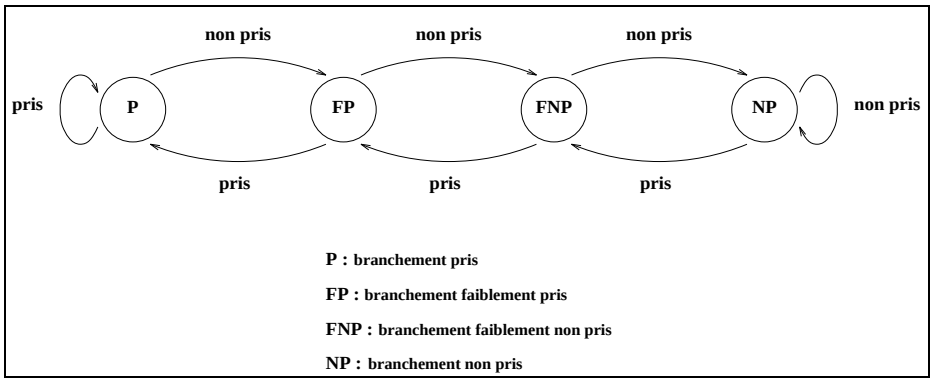


FIG. 3.9 - *Algorithme de prédiction de branchement du DEC 21964*

Dans la mesure où il faut un cycle pour calculer l'adresse cible, une *bulle* (c'est à dire un cycle sans instruction) est systématiquement insérée dans le pipeline après tout branchement. Cependant, quand des instructions restent bloquées dans les étages avals de l'Ibox, le DEC permet l'*écrasement* de cette bulle par des instructions postérieures.

Le DEC 21164 supporte plusieurs niveaux d'anticipation de branchement (jusqu'à 6).

Les prédictions sont testées par l'Ebox qui met à jour la table de branchement. Cette évaluation intervient relativement tard dans le pipeline, au niveau du deuxième étage d'exécution, ce qui introduit une pénalité de 5 cycles pour vider le pipeline et reprendre le chargement. Cependant, d'après les concepteurs, cela devrait arriver dans moins de 15 % des cas grâce à l'importante table de prédiction de branchement (2 K-entrées, une par instruction) implémentées. De plus, lors du

chargement du cache d'instructions, les bits d'historiques ne sont pas initialisés ce qui permet de conserver la trace des branchements même après un défaut de cache.

Le DEC 21164 met également en œuvre une pile, gérée comme une queue circulaire de 12 entrées, utilisée pour les retours de procédure. Cette pile est sollicitée lors du décodage du code opération de l'instruction. Chacune des entrées de cette pile sauvegarde un index du cache d'instructions qui permettra de prédire une adresse de retour. Ce mécanisme qui n'est pas mis en œuvre sur les autres architectures donne une grande sûreté de la prédiction de l'adresse de retour. Une erreur de prédiction provoquera une exception et sera résolue de la même manière qu'une mauvaise prédiction de branchement. Sont également prédis à partir de cette pile, les changements du mode PAL vers le mode utilisateur et vice-versa.

Logique de prédiction sur le Power2 Le Power2 n'utilise pas de prédiction dynamique de branchement. Il implémente deux unités de décodage partiel, dédiées aux branchements, et séparées des unités entière et flottante. Leur rôle est de détecter les instructions de branchement dans le tampon d'instructions en avance sur le flot d'exécution de façon à anticiper le calcul de l'adresse cible ainsi que la direction du branchement. Les accès au cache d'instructions sont ainsi découplés de l'exécution et les branchements inconditionnels peuvent n'introduire aucun retard. Huit instructions en séquence sont inspectées. Les unités de décodage partiel alimentent les unités de branchement responsables des calculs d'adresse. Dès que l'adresse cible est disponible, les instructions sont préchargées dans un tampon spécialement réservé à cet usage. L'ICU dispose aussi d'un mode d'émission des instructions conditionnel utilisé lorqu'il est nécessaire de calculer une condition. Grâce à l'ensemble de ces mécanismes, les branchements non pris n'introduisent aucun retard et les branchements pris entraînent un retard d'au maximum un cycle (quand les conditions de branchement sont connues à temps).

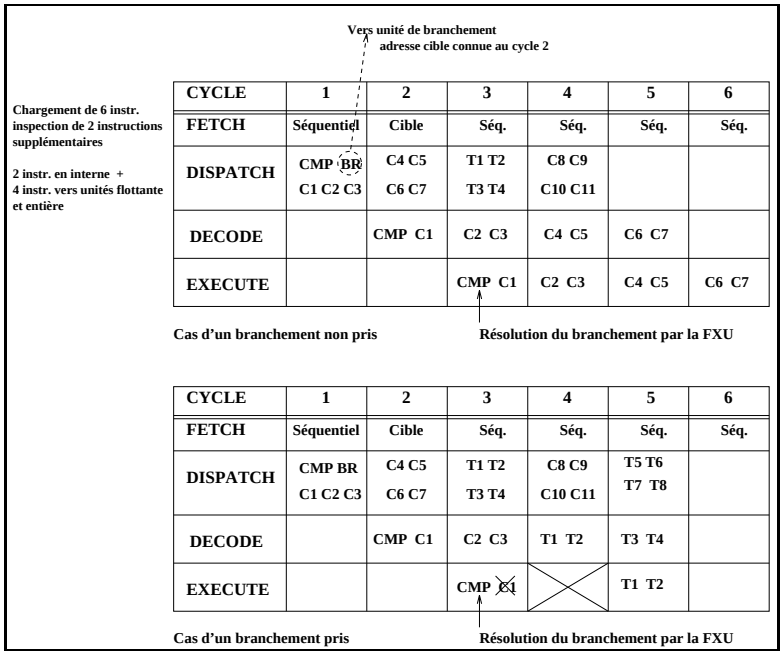


FIG. 3.10 - *Cas des branchements conditionnels sur le POWER2*

La figure 3.10 présente l'exemple d'un branchement conditionnel. Seules sont représentées les instructions qui concernent la FXU et les unités de branchements. Dans le cas d'un branchement pris, on remarque un cycle de pénalité dû à l'abandon des instructions. Cependant, cette pénalité aurait pu être évitée en insérant des instructions indépendantes entre l'instruction de comparaison et celle de branchement (ce qui représente quand même 6 instructions à insérer).

Grâce à ses deux unités de branchement, le Power2 peut exécuter deux branchements par cycle (seulement si le premier est résolu et non pris !).

3.3.6 Interblocages

Les mécanismes de gestion des interblocages ont une influence considérable sur les performances des processeurs superscalaires.

Nous présentons ici une description des différents types d'interblocages pouvant survenir sur les processeurs, avant de montrer comment ils sont résolus sur les trois processeurs étudiés.

Défauts de cache: lorsqu'une instruction ou une donnée accédée ne se trouve pas dans le cache, une condition d'interblocage est détectée (excepté sur le DEC 21164). Par contre, lors d'une traduction d'adresse, quand le descripteur de la page concernée ne se trouve pas dans le cache de traduction d'adresses (TLB), une exception est déclenchée sur le MIPS R8000 et le DEC 21164. Sur l'IBM Power2, le traitement des défauts de TLB est géré par matériel.

Conflits de ressources: les conflits de ressources surviennent lorsque plusieurs instructions tentent d'accéder simultanément à une même ressource, par exemple au même étage du pipeline d'une unité fonctionnelle.

Dépendances de données: Les dépendances de données ont pour effet de forcer l'exécution séquentielle de plusieurs instructions. Trois types de dépendances de données inter-instructions peuvent être distinguées :

- **Lecture après écriture (ou RAW).** Ce conflit se produit lorsqu'une instruction cherche à lire une donnée avant que cette dernière ait pu être écrite par une instruction lancée antérieurement. Il est à remarquer que l'utilisation des mécanismes de *bypass* (voir page 56) limite les occurences de ce type de dépendances.

- **Ecriture après lecture (ou WAR).** Ce conflit se produit lorsqu'une instruction cherche à écrire dans un registre avant que celui-ci n'ait pu être lu par une instruction antérieure. Toutefois, ce conflit reste assez rare pour les pipelines dont la phase de lecture se situe tôt (dans les premiers étages), et la phase d'écriture tard, mais il peut devenir de plus en plus fréquent avec l'avènement de l'exécution dans le désordre.

- **Ecriture après écriture (ou WAW).** Ce conflit se produit lorsqu'une instruction essaie d'écrire dans un registre alors qu'une instruction précédente n'a pu écrire son résultat dans ce même registre. Ce conflit ne survient que dans les pipelines qui écrivent à différents étages ou qui autorisent une instruction à continuer même si une instruction précédente est gelée.

- **Remarque**: une lecture après une lecture n'est pas une dépendance de données.

Ces trois types de dépendance sont aussi connus sous les noms de *vraie dépendance* , *d'anti dépendance* et de *dépendance de sortie*. L'exemple de la figure 3.11 illustre ces trois types de dépendance.

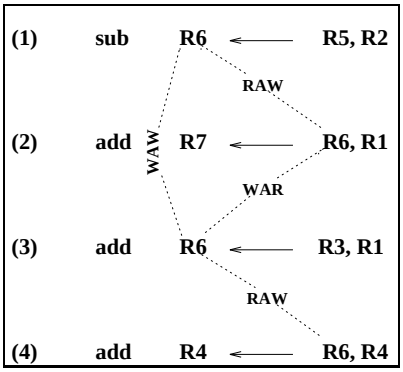


FIG. 3.11 - *Illustration des trois types de dépendance*

L'instruction (2) présente une dépendance de type RAW sur l'instruction (1). L'instruction (3), elle, présente une dépendance de type WAR par rapport à l'instruction (2) et est aussi victime d'une dépendance de type WAW par rapport à l'instruction (1).

Résolution des interblocages sur le DEC 21164

Comme on l'a vu précédemment, la détection des conflits de ressources est globale et a lieu dans les étages statiques du pipeline. Cette détection repose sur un mécanisme de scoreboard implémenté sur l'Ibox.
Cependant, il arrive que ce mécanisme ne puisse pas prévoir la disponibilité de certaines ressources au moment du cycle d'émission. Le 21164 implémente un mécanisme spécial appelé *Replay Trap* qui consiste à abandonner l'instruction au moment où elle requiert la ressource indisponible. Cette instruction est alors relancée dans le pipeline et laisse ainsi le temps à la ressource de se libérer (si ça n'est pas le cas, un *replay trap* est à nouveau provoqué). Cette situation intervient au niveau des défauts de cache de données. Lorsqu'une instruction de calcul est émise et qu'au moment où on évalue la disponibilité des opérandes il se produit un *Dcache miss*, l'opération de calcul est abandonnée et relancée.

Résolution des interblocages sur le MIPS R8000

Pour les instructions entières et les accès mémoire, le R8000 s'appuie lui aussi sur la technique du scoreboarding et sur une exécution des instructions totalement synchrone. En ce qui concerne l'unité flottante, les instructions sont émises vers la *FPQ* (voir chapitre 3.3.3). Les dépendances de données entre les instructions de chargement flottant et les opérations flottantes, sont résolues par cette file. Au niveau des écritures en mémoire, un second mécanisme, la *Store Address Queue* (SAQ), localisée sur l'unité entière, permet de séquencer les accès à la mémoire sans attendre la disponibilité des données à écrire. Les lectures peuvent ainsi être exécutées en doublant les écritures en attente avec détection des aléas.

En plus des combinaisons standards conduisant au *NOP* (par exemple instructions écrivant dans le registre R0), le jeu d'instruction du R8000 inclut une nouvelle instruction *SSNOP* pour

Super-Scalar No-Operation. Cette instruction garantit qu'il n'y aura pas d'autres instructions en séquence émises au même cycle, ceci afin de préserver certaines restrictions inhérentes à l'architecture MIPS (exemple, l'insertion nécessaire de 3 *SSNOP* (donc 3 cycles) entre n'importe quelle opération concernant la *TLB* et une opération mémoire).

Résolution des interblocages sur le Power2

La logique d'émission du Power2 ne s'occupe pas de la détection des dépendances de données, ceci afin d'augmenter le taux de parallélisme des instructions. Les instructions sont émises vers les unités entière et flottante dans des files d'instructions où divers mécanismes permettent de détecter et de résoudre les dépendances.

L'unité entière du Power2 met en œuvre une technique de scoreboarding pour détecter et résoudre les dépendances de données.

Comme son prédécesseur le Power1, le Power2 implémente la technique du renommage de registres dans l'unité flottante. Cette approche permet de résoudre les conflits de registres de type WAR et WAW grâce à l'utilisation d'un nombre de registres physiques supérieur au nombre de registres logiques vu par le compilateur et le jeu d'instructions. Le nombre de registres physiques a été augmenté de 40 sur le Power1 à 54 sur le Power2 par rapport aux 32 registres logiques définis par l'architecture. L'unité flottante implémente aussi un deuxième mécanisme qui permet de sérialiser les opérations de calculs flottants avec les opérations d'écriture mémoire. Ce mécanisme permet d'optimiser le parallélisme entre ces deux types d'instructions en détectant les dépendances d'exécution (résolution des conflits de type RAW et WAW). Les instructions d'écriture et d'arithmétique flottante sont réparties respectivement dans deux files différentes (voir chapitre 4.4). Un *champ de comptage des écritures* ou *STC* (pour *Store Count Field*) associé à la file des instructions arithmétiques indique le nombre d'instructions d'écriture présentes dans la file et devant s'exécuter avant une opération arithmétique. Un banc de comparateurs teste l'opérande cible de l'instruction arithmétique avec les opérandes sources des instructions dans la file d'écriture. Cette information, utilisée en conjonction avec le STC, détermine si le pipeline peut exécuter l'instruction. Par ailleurs, ce mécanisme est aussi utilisé pour déterminer quand les instructions d'écriture doivent commencer. Le fait que le Power2 puisse lancer deux opérations d'écriture et de calcul par cycle complique d'autant le fonctionnement. Cependant, ce mécanisme exploite au maximum le parallélisme en ne synchronisant les unités que sur les vraies dépendances de données.

Mécanisme de *bypass*

Sur les trois microprocesseurs étudiés, un mécanisme de chaînage des instructions, appelé mécanisme de *bypass* (court-circuit) est mis en œuvre : le résultat d'une opération ou d'un chargement dans un registre peut être utilisé comme opérande par une instruction postérieure dès sa sortie de l'unité fonctionnelle et sans attendre son écriture dans le registre destination. Ces mécanismes permettent de réduire les dépendances entre les unités en minimisant les délais, ils jouent ainsi un rôle essentiel pour l'obtention de performances.

3.4 Exceptions et interruptions

Les exceptions surviennent lors d'erreurs de fonctionnement telles que :

- débordement arithmétique (overflow),
- erreur de bus,
- tentative d'utilisation d'instructions réservées,
- erreur d'adressage,
- défaut sur le cache de traduction d'adresses,
- défaut de pages en mémoire,
- ..

De manière générale, il s'agit d' "accidents" dans l'exécution. Certaines exceptions doivent être ignorées de l'utilisateur (par exemple un défaut de page ne doit pas changer le résultat final d'une application) et d'autres provoquer un traitement spécifique (par exemple, en cas de débordement arithmétique, appel à une routine spécialisée).

Les interruptions sont une autre forme de rupture de séquence. Contrairement à la catégorie précédente, les interruptions peuvent survenir de manière asynchrone (par exemple réinitialisation du système, interruption externe, ...) et sans aucune relation avec les instructions en cours d'exécution.[3]

Gestion précise ou gestion imprécise?

On parle de gestion précise des exceptions si l'état de la machine qui résulterait de l'exécution séquentielle de toutes les instructions antérieures à l'instruction provoquant l'exception peut être reconstruit dans tous les cas. Il est alors possible après le traitement de l'exception de reprendre l'exécution à l'instruction même ayant provoqué l'exception.

On parle de gestion imprécise des exceptions dans le cas contraire, c'est à dire s'il n'est pas toujours possible de restaurer un état cohérent de la machine permettant de reprendre l'exécution à l'instruction ayant provoquée l'exception.

Certaines exceptions supportent dans la plupart des cas une gestion imprécise car elles entraînent le déroulement total d'un processus sans reprise derrière l'instruction ayant provoqué l'exception (erreur de bus, instructions interdites, débordement arithmétique, ..), d'autres au contraire ne supportent pas cette gestion (le résultat d'une application ne doit pas dépendre de l'occurence ou non de tel ou tel défaut de page).

La gestion précise des exceptions est difficile à réaliser sur un processeur superscalaire possédant des unités d'exécution indépendantes et de latence de traversées distinctes. En effet, plusieurs instructions étant lancées dans un même cycle et dans des unités différentes, il devient très difficile de savoir à quelle instruction se rapporte l'exception et de bloquer le pipeline en conséquence.

3. Il existe une certaine confusion au niveau de ces deux termes dans la mesure où les constructeurs les utilisent pour désigner des choses différentes.

Exceptions sur le Power2 : La mise en œuvre d'une gestion précise des exceptions accroît la complexité du contrôle dans le pipeline. Aussi, sur le Power2, seules les exceptions entières sont gérées de façon précise. Le haut degré de parallélisme des instructions voile un peu le concept de compteur de programme dans la mesure où dans un processeur superscalaire très asynchrone tel que le Power2, chaque unité fonctionnelle, et à l'intérieur de celle-ci, chaque étage de pipeline, a sa propre vue de l'instruction courante. Pour mettre en place un mode de gestion précis des interruptions, un compteur de programme de "validation" est implémenté et est utilisé par les mécanismes de traitement d'interruptions : les instructions antérieures à celle désignée par le compteur sont exécutées (c'est à dire ont écrit leur résultat), et les instructions postérieures peuvent être annulées. De plus, le Power2 implémente un système de priorités des exceptions qui privilégie les instructions le plus tôt dans le flot (donc le plus loin dans le pipeline). Par exemple, une exception déterminée sur un chemin prédit aura une priorité minimale. Les exceptions flottantes et les interruptions externes sont gérées de façon imprécise dans la mesure où l'ICU termine l'exécution de toutes les instructions en attente avant de les prendre en compte. Cependant pour le déboguage, un mode d'exception précis est disponible bien qu'il dégrade les performances.

Exceptions sur le DEC 21164 : Les concepteurs du DEC ont défini deux types de rupture de séquence : les *exceptions*, gérées de manière imprécise et les *non-exceptions* gérées de façon précise. Les exceptions sont servies dans un mode spécifique à l'architecture Alpha, appelé le mode *PAL*. Ce mode donne accès à un ensemble de primitives systèmes exécutées de manière ininterruptible (voir page 103).
Lors du traitement des exceptions, les instructions en cours dans le pipeline terminent leur exécution avant que l'exception soit traitée par appel à une routine du Palcode. On peut citer en exemple le cas des interruptions. Le 21164 supporte trois sources d'interruption :

- matérielle, par le biais de 7 broches d'entrées :
  - *irq_h[0 :3]*, sont des entrées classiques d'interruptions d'entrées/sorties ;
  - 3 entrées supplémentaires sont dédiées à des interruptions de type défaut-alimentation, arrêt-système, test du système.
- logicielle. Ces interruptions sont internes et concernent des interruptions logicielles ainsi que celle fournie par les compteurs de performances (voir chapitre 8).
- interruptions asynchrones. Ce type d'interruption permet à un processus de prendre en compte un événement au cours de son exécution. Si le processus a défini une routine de traitement relative à cet évènement, le processus est alors interrompu et la routine exécutée.

Quand le processeur reçoit une requête d'interruption, le pipeline est vidé des instructions qu'il contient jusqu'au point où celles-ci ne pourront plus générer de nouvelles exceptions (les écritures en attente sont également effectuées). L'adresse de redémarrage est alors sauvegardé dans un registre interne dédié aux traitements des exceptions et le processeur entre en PALmode afin d'exécuter la séquence de code relative à l'interruption.

Le cas des non-exceptions ne nécessite pas de terminer l'exécution des instructions en cours de traitement. Celles-ci sont tout simplement abandonnées et le pipeline redémarre directement à une nouvelle adresse. À titre d'exemple on peut citer les cas de mauvaises prédictions de branchement,

les *replay traps* (voir page 55), ainsi que les défauts de cache de données (qui peuvent aussi dans certain cas être traités par un gel de l'émission des instructions).

Les exceptions arithmétiques (qui sont inclues dans les non-exceptions), constituent un cas particulier au fonctionnement précédemment cité dans la mesure où le processeur peut exécuter plusieurs instructions après celle ayant provoquée une erreur. L'adresse de retour sera alors celle de la dernière instruction exécutée.

Le DEC 21164 implémente aussi comme son prédécesseur l'instruction *Trap Barrier* qui permet de faire une gestion précise des exceptions. Le lancement de l'instruction suivante n'est autorisée que quand le matériel garantit qu'aucune des instructions précédentes n'a causé d'exception, en instaurant des délais (*stall*), si cela est nécessaire. *Il n'y a pas de mode spécifique garantissant le traitement précis des exceptions.*

Exceptions sur le MIPS R8000 : Le MIPS R8000 implémente une gestion précise des exceptions pour les instructions entières. Les exceptions issues du pipeline entier telles que codes d'opérations illégaux, adresses mémoires invalides, ou gestion de la TLB sont traitées de manière précise. Les exceptions issues du pipeline flottant sont traitées de manière imprécise. Du fait du découplage des deux unités, une instruction flottante qui provoque une exception, écrit une valeur appropriée de substitution (par exemple NaN) dans le banc de registres, et valide sa broche d'interruption (*FPINTR_*) à destination de l'unité entière, puis elle continue l'exécution des instructions suivantes. L'interruption est prise en compte par un registre dédié à cet usage (*Cause Register*), et un traitement spécifique est généré par l'unité entière. Cependant, pour des raisons de compatibilité avec le R3000, le processeur supporte aussi un mode de traitement des exceptions précis où le pipeline entier est gelé à chaque exécution d'instructions flottantes. Bien entendu ce mode dégrade énormément les performances.

Le R8000 supporte donc des interruptions générées en interne mais il peut aussi en recevoir de sources externes. À la différence du DEC, le R8000 n'implémente pas de broches spécifiques à cet usage. Ces interruptions sont prises en compte par le contrôleur de cache et le statut de son registre d'interruption est transféré au R8000 via le Tbus. L'interruption est alors traitée par l'exécution de la routine appropriée.

3.5 Unités entières et de calcul d'adresse

Ce chapitre décrit les unités de calcul entier et d'adresses. À noter la philosophie différente d'implémentation des 3 processeurs :

- Le Power2 sépare l'unité de gestion des branchements de l'unité entière et d'accès à la mémoire.

- Le MIPS R8000 sépare les unités d'accès à la mémoire de l'unité entière. Il implémente deux unités arithmétiques entières et deux unités d'adressages.

- Le DEC 21164 met en œuvre deux unités entières relativement identiques mais seule une est à même de gérer les branchements ainsi que les opérations d'écriture vers la mémoire.

3.5.1 Le DEC 21164

Constitution L'unité entière (*Ebox*) est organisée autour de deux pipelines d'exécution de 64 bits (appelés E0 et E1) qui incluent les éléments suivants :

- deux additionneurs,
- deux unités d'opérateurs booléens,
- une unité de décalage,
- une logique de manipulation d'octets (elle permet de manipuler des données de 8 ou 16 bits. Consulter le chapitre 1.6.1 pour plus de détails).
- un multiplieur entier.

Ces deux pipelines ne sont pas tout à fait identiques. Les deux gèrent les opérations arithmétiques de base ainsi que les opérations de lecture de la mémoire, mais seule l'unité E0 dispose d'un multiplieur et d'une unité de décalage alors que l'unité E1 gére les branchements et les opérations d'écriture vers la mémoire. D'où, des restrictions au niveau de l'émission simultanée de certaines instructions arithmétiques.

À cette structure d'opérateurs, vient s'ajouter un banc de 40 registres de 64 bits qui contient les 32 registres définis par l'architecture Alpha et 8 autres registres appelés *PAL shadow registers* destinés à entretenir des copies locales en mode PAL. Pour fournir les opérandes aux deux pipelines, le banc de registres dispose de 4 ports de lecture. Deux ports d'écriture sont destinés à recevoir les résultats des calculs effectués et servent aussi lors des opérations de chargement en provenance de la mémoire.
Les circuits de *bypass* sont implémentés, ce qui permet au résultat de n'importe quelle opération entière d'être disponible au plus tôt pour toute autre instruction consécutive. Seule exception, la multiplication entière (déjà longue, de 8 à 16 cycles) qui n'est pas à même de recevoir ses opérandes directement des autres unités fonctionnelles. La figure 3.12 schématise les deux pipelines d'exécution du DEC.

Fonctionnement Le 21164 peut associer par deux la plupart des opérations entières, cependant, du fait de l'unicité de certains opérateurs, il ne pourra pas émettre dans le même cycle deux instructions de décalage, deux multiplications entières ou encore deux branchements. Pour effectuer au mieux le séquencement des instructions dans les deux pipelines entiers, une fonction de regroupement a été définie, basée sur un certain nombre de règles mises en place au niveau de chaque classe d'instructions (confère chapitre 3.3.2).

Il est à noter aussi un certain nombre de restrictions lors des opérations d'accès à la mémoire[4] :

- Une instruction de chargement ne peut pas être simultanément émise avec une instruction d'écriture du fait de la structure du cache (voir chapitre 5.5).
- Une instruction de chargement ne peut pas être émise dans le deuxième cycle qui suit l'émission d'une instruction d'écriture (voir chapitre 5.5).

4. Pour plus de détails sur les règles d'émission, consulter [13]

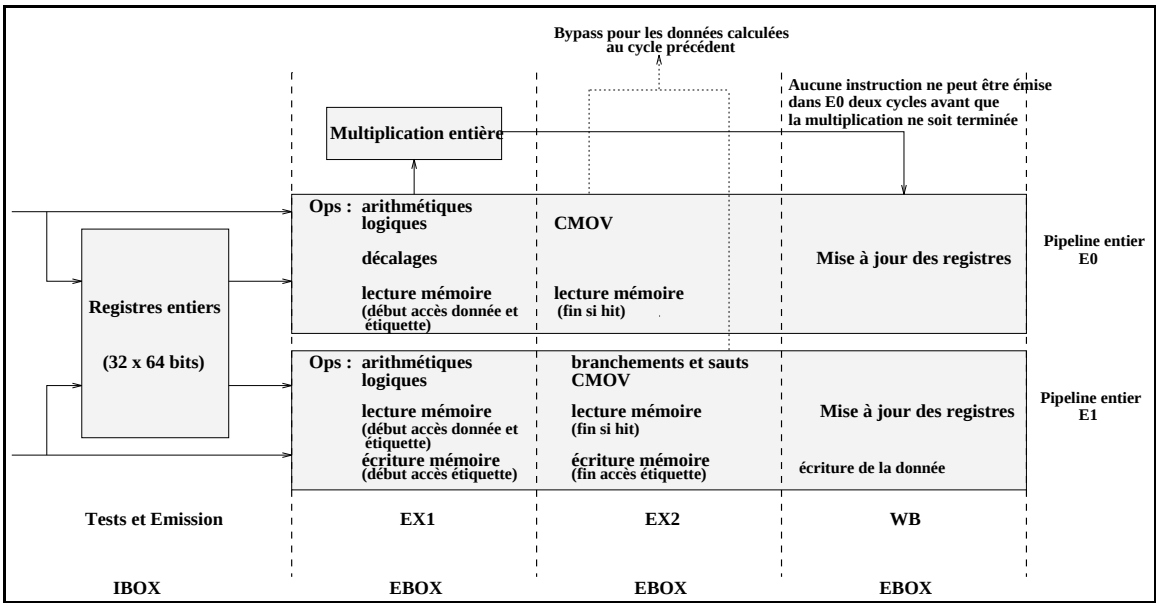


FIG. 3.12 - *Pipelines entiers du DEC 21164*

Toutes les opérations entières du DEC 21164 sont exécutées en un cycle (à l'exception de la multiplication). Presques toutes sont effectuées au niveau du premier étage d'exécution du pipeline. Seules les opérations de "déplacement" conditionnel de données (*CMOV*) et les branchements sont exécutés au cours du cycle suivant. Ces instructions pourraient être traitées au même niveau que les autres mais cette astuce permet d'améliorer le débit du pipeline. Une instruction de comparaison et un branchement conditionnel dépendant ou un *conditional-move* dépendant peuvent être séquencés au même cycle. L'exemple ci-dessous illustre ce fonctionnement :

Exemple 8

```
CMP    R1, R2, R3;
BEQ    R3, label;
```

La comparaison s'effectue au niveau du premier étage d'exécution et son résultat est utilisé par l'instruction de branchement conditionnel exécuté au cycle suivant. Sans cette accommodation, ces instructions seraient émises l'une après l'autre, réduisant ainsi le débit du pipeline (en encombrant de plus l'étage d'émission d'un cycle supplémentaire). Le fait de retarder l'évaluation du branchement à l'étage 5 du pipeline étend la pénalité d'un cycle en cas de mauvaise prédiction. Cependant, selon les concepteurs du DEC, cette pénalité n'intervient que lors de mauvaises prédictions ce qui devrait arriver dans moins de 15 % des cas grâce à la table de prédiction (confère le chapitre 3.3.5). Si le branchement était effectué dès le premier étage d'exécution, il serait retardé d'un cycle du fait de la dépendance de donnée au cas où on ne pourrait trouver une instructions entière indépendante à insérer avant.

	21164	21064
Opérations entières	1 cycle	1 cycle
Décalage et opérations sur les octets	1 cycle	2 cycles
Comparaison avec branchement	0 cycle	1 cycle
Hit cache primaire	2 cycles	3 cycles
Multiplication entière	8 - 16 cycles	19 - 23 cycles
Mauvaise prédiction de branchement	5 cycles	4 cycles

TAB. 3.1 - *Amélioration des latences d'opérations entre le DEC 21164 et le 21064*

L'unité entière a été améliorée par rapport à celle de son prédécesseur, le DEC 21064, où les opérations de décalage et de manipulation d'octets étaient effectuées au cours du deuxième étage d'exécution du pipeline. Le tableau 3.1 montre les progrès réalisés (en cycles) sur les latences des principales opérations entre le 21064 et 21164.

3.5.2 Le MIPS R8000

Constitution La figure 3.13 détaille l'unité entière du MIPS R8000. Celle-ci est constituée de deux unités arithmétiques et logiques de 64 bits, d'une unité de décalage, d'une unité de multiplication-division et de deux générateurs d'adresses. L'architecture MIPS offre aussi à l'utilisateur 32 registres généraux (GPR) de 64 bits. Le banc de registres dispose de 9 ports en lecture et de 4 ports en écriture dont l'utilisation se répartie comme suit :

- 4 ports de lecture sont dédiés à l'usage des ALUs (opérandes).
- 4 ports servent à la génération des adresses lors des opérations de référence mémoire entière ou flottante.
- Le neuvième port est spécialement dédié aux écritures dans le cache de données. Son existence permet de réduire le partage des ports et de ce fait, il contribue à la résolution des conflits de ressources. Cependant, une seule écriture dans le cache est possible par cycle.
- Les 4 ports d'écriture sont destinés à recevoir les résultats des opérations effectuées ainsi que les données acheminées lors des opérations de lecture.

Fonctionnement Toutes les opérations entières sont effectuées de manière synchrone au niveau de l'étage d'exécution du pipeline (cycle 4). Le MIPS R8000 permet l'exécution de 4 instructions entières par cycle : deux opérations arithmétiques et deux calculs d'adresses. L'unité de multiplication-division n'est pas pipelinée ce qui empêche de lancer une opération à chaque cycle. La latence de la division peut varier de 21 à 73 cycles suivant les opérandes. La latence de la multiplication est de 4 (32 bits) ou 6 (64 bits) cycles. L'unité de multiplication-division utilise deux registres spéciaux pour stocker les parties haute et basse du résultat. Deux instructions spécifiques permettent de transférer le contenu de ces registres spéciaux dans les registres généraux. Ceci introduit des restrictions dans le séquencement de ces opérations dans la mesure où aucune instruction de multiplication-division

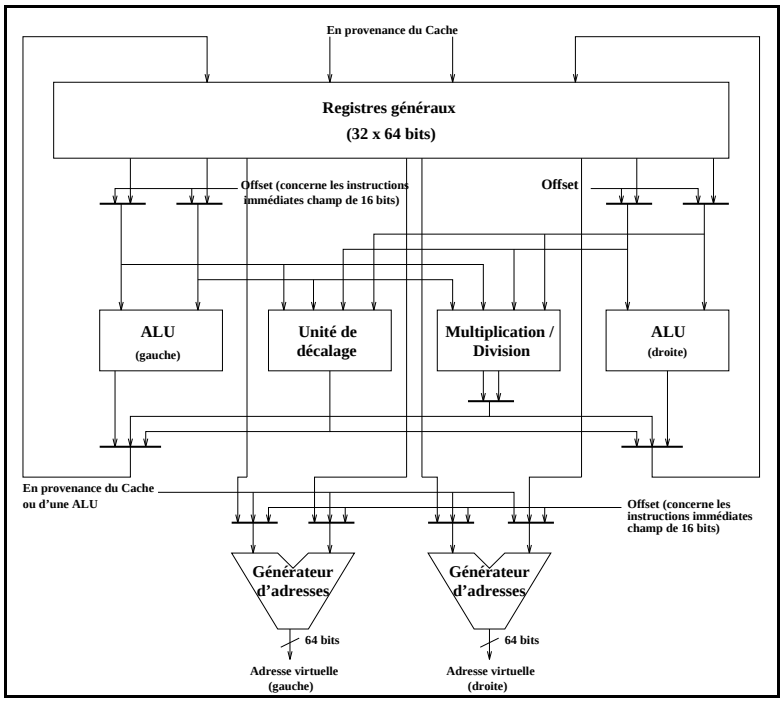


FIG. 3.13 - *Unité entière du MIPS R8000*

ne peut être séquencée parmi les deux instructions qui suivent la lecture des registres spéciaux (compatibilité avec le R3000).

3.5.3 Le Power2

Constitution L'unité d'exécution entière du Power2 est organisée autour d'un tampon d'instructions, d'une unité de décodage et de deux unités de contrôle. Elle dispose aussi de deux unités arithmétiques et logiques qui lui permettent d'effectuer les calculs. Ces deux unités regroupent un certains nombre d'opérateurs :

- *L'unité d'exécution EX0* est constituée d'un additionneur à deux entrées et d'une unité de décalage.
- La deuxième (*EX1*) inclut un additionneur à trois entrées, une unité de décalage ainsi qu'une unité de multiplication-division (voir page 64).

D'autre part, rappelons les deux unités de branchements et l'unité de calcul sur les codes conditions, toutes trois implémentées sur l'ICU.

L'architecture Power définit également 32 registres généraux de 32 bits (*GPR*) implantés sur la FXU. Afin d'implémenter la totalité des ports qui accèdent au banc de registres sans que celui-ci soit trop volumineux, deux copies identiques des GPR existent au sein du processeur, alimentant chacune des unités d'exécution. Ceci permet de disposer de 7 ports en lecture et de 4 ports en écriture répartis de la façon suivante :

- Deux ports de lecture pour les opérandes d'EX0, 3 pour les opérandes d'EX1 (à cause de l'additionneur à trois entrées).

– Les sixième et septième ports sont destinés aux données à écrire dans le cache de données.

– Les 4 ports d'écriture reçoivent les résultats des ALUs et des opérations de lecture. Rappelons que les opérations *load with update* produisent deux écritures, la donnée lue et l'adresse préincrémentée. Par ailleurs, le pipeline entier du Power2 permet le *bypass* de l'étage d'accès au cache dans le cas des opérations entières. Aussi, une instruction de lecture mémoire qui avait un cycle d'avance dans le pipeline entier peut se finir en même temps qu'une instruction *registre-registre*. La présence de deux ports d'écriture par pipeline permet de résoudre cette situation.

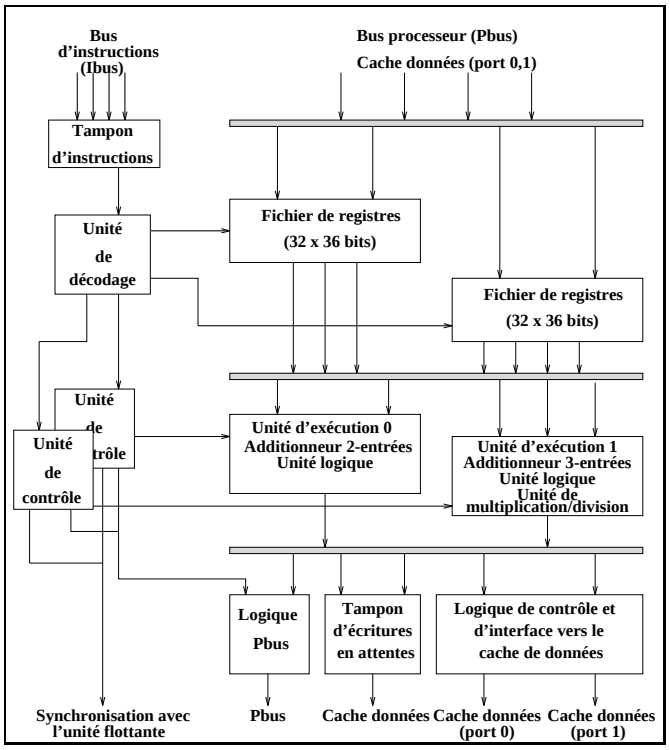


FIG. 3.14 - *Unité arithmétique entière du POWER2*

Fonctionnement Les unités présentes sur l'ICU disposent des instructions émises en interne par celle-ci. Elles sont donc exécutées directement après leur émission en un cycle. Les unités de branchements permettent, dès qu'il y a une instruction de saut, de calculer au plus tôt l'adresse à partir de laquelle on préchargera les instructions cibles. L'autre unité reçoit les instructions logiques portant sur le registre de conditions. Ce registre est organisé en 8 champs de 4 bits destinés à recevoir les résultats des instructions de branchement conditionnel. À ce niveau, deux instructions peuvent être exécutées simultanément à chaque cycle, soit deux branchements, soit un branchement et une instruction manipulant les codes conditions.

La FXU est chargée d'exécuter toutes les instructions arithmétiques ainsi que celles ayant trait aux références mémoires. L'ICU peut émettre jusqu'à 4 instructions entières sur l'Ibus. Un tampon

de huit entrées réceptionne les instructions et les émet vers l'unité de décodage. Cette unité peut traiter deux instructions à la fois et contrôle :

- la lecture des registres généraux,
- l'extension des immédiats,
- résoud les dépendances (mécanisme de scoreboard et de bypass),
- et émet les instructions vers les deux unités d'exécution (*EX0* et *EX1*).

Durant le cycle de décodage, les registres généraux sont lus. Au cycle suivant, les unités de décodage émettent les instructions vers les unités d'exécution. Les deux unités de contrôle d'exécution gèrent l'arrivée des opérandes, interviennent dans l'exécution des opérations de lecture/écriture et enfin, contrôlent l'écriture des résultats de chacune des ALUs qu'elles gouvernent dans le banc de registres. La figure 3.14 reprend les principaux éléments de l'unité arithmétique du Power2 [5].

L'unité entière inclut un support pour l'exécution parallèle d'opérations d'additions dépendantes (comme le TI SuperSparc par exemple). Toutes les opérations de multiplication s'effectuent en deux cycles (3 à 5 cycles étaient nécessaires sur le Power1). La division nécessite de 13 à 14 cycles suivant les opérandes contre 19-23 sur le Power1.

Au niveau de l'étage de décodage, la FXU teste les dépendances de données entre les deux instructions. Du fait de l'exécution dans l'ordre, elle peut être amenée à les exécuter séquentiellement. L'additionneur à 3 entrées implémenté dans EX1 permet d'améliorer les performances en exécutant en parallèle deux additions dépendantes. La séquence de code de l'exemple 9 nécessiterait d'être exécutée séquentiellement du fait de la dépendance de type RAW. Au lieu d'attendre le résultat de la première addition, l'additionneur à trois entrées va permettre de calculer directement (*R1 + R2 + R4*) pendant que l'autre addition s'effectue en parallèle.

Exemple 9

$$R1 + R2 \rightarrow R3;$$
$$R3 + R4 \rightarrow R5;$$

À noter que ces deux additions peuvent être des calculs d'adresses dans des opérations de load/store.

3.6 Conclusion

L'implémentation du DEC 21164 privilégie une fréquence d'horloge très élevée qui impose de nombreuses restrictions sur les instructions séquençables en parallèle (en particulier, leur alignement). L'absence de tampon d'instructions en entrée des unités fonctionnelles ne permet pas de découpler les exécutions entre instructions entière et flottante et son mécanisme d'émission des instructions interdit une exécution dans le désordre.

5. Les unités implémentées sur l'ICU ne sont pas représentées

Notons enfin, que la phase de séquencement des instructions a été pipelinée sur 3 cycles (soit 10 ns à 300 MHz). Ceci est à rapprocher des phases de décodage des deux autres processeurs étudiés : un cycle (13 ns) sur le MIPS R8000 et deux cycles (soit 28 ns dans le cas des instructions entières) sur le Power2 (les trois processeurs utilisent également des bits de prédécodage). Cette répartition est imposée par la fréquence de fonctionnement du processeur, la gestion des conflits restant inférieure à celle du MIPS et du Power.

Le MIPS R8000 bénéficie d'une structure de séquencement plus efficace que celle du DEC. Sa logique d'émission a toujours le choix entre quatre instructions consécutives sans restrictions d'alignement. La présence d'un tampon pour les instructions flottantes permet, de plus, un découplage des exécutions qui permet de masquer la latence du cache externe. En inversant les étages d'exécution et d'accès mémoire, le MIPS R8000 supprime la pénalité du *load-use* fréquente dans de nombreux codes, mais ajoute un cycle de pénalité dans le cas des *address-use* et des branchements. Pour les codes flottants, un nouveau mode d'adressage a été implémenté de manière à réduire cette pénalité.

Sur le DEC 21164, comme sur le MIPS R8000, la direction et l'adresse des branchements ne sont connues que très tard dans le pipeline (respectivement au cycle 5 et au cycle 4). Ceci explique la nécessité d'utiliser des mécanismes matériels de prédiction de branchement.

Le Power2 a le mécanisme d'émission des instructions le plus complexe. Avec l'émission possible de six instructions par cycle et sa multitude d'unités fonctionnelles, les architectes du Power2 ont privilégié un haut degré de parallélisme d'exécution plutôt que le temps de cycle. Les mécanismes mis en place tels que tampons d'instructions, renommage de registres, et autres ..., reflétent la complexité de cette architecture pour permettre un taux optimal d'exécution simultanée d'instructions, même dans le désordre. L'implémentation des mécanismes dédiés aux traitements des branchements (tampon cible, unités de décodage, unités de branchement, mode d'émission conditionnel) témoignent de la nécessité d'anticiper l'exécution du code et de la difficulté, sur ce type d'architecture, de garder occupé l'ensemble des unités fonctionnelles. L'absence de prédiction de branchements sur le Power2 est sans doute due au traitement précoce des branchements dans le pipeline. Cependant, sur les branchements conditionnels (autre que gestion de boucles), les performances du Power2 peuvent être sévèrement affectées par l'absence de prédiction.

À partir de l'étude de ces trois microprocesseurs, quelques remarques s'imposent. L'utilisation des tampons d'instructions présente un grand intérêt :

- Ils permettent de découpler l'exécution au sein des diverses unités fonctionnelles augmentant ainsi le parallélisme utilisable entre les instructions (cas des trois tampons *arithmetic queue, store queue* et *load queue* implémentés sur l'unité flottante du Power2).

- Ils constituent des endroits où les instructions peuvent attendre la résolution des conflits de ressources sans bloquer le mécanisme d'émission (exemple de la *Floating-Point Queue* sur l'unité entière du MIPS R8000).

- Ils permettent une exécution des instructions dans le désordre en favorisant l'exécution d'instructions par rapport à celles requérant des ressources non disponibles (exemple de la file d'instructions du PowerPC 601 (voir [3])).

– Ils garantissent l'alimentation des diverses unités en pré-chargeant des instructions bien au-delà de l'instruction courante exécutée (tampons de préchargement des instructions du Power2, du MIPS R8000, du DEC).

L'utilisation du séquencement superscalaire exacerbe le besoin de séquencer en avance. Les mécanismes de séquencement sont aujourd'hui essentiels pour la performance. On notera l'importance accrue des mécanismes de gestion de branchements.

3.7 Récapitulatif

Nous reprenons dans les tableaux 3.2 et 3.3 quelques valeurs essentielles de l'architecture des processeurs étudiés. En ce qui concerne l'exécution des instructions flottantes et d'accès à la mémoire, nous expliciterons ces valeurs dans les chapitres respectifs.

	DEC 21164	MIPS R8000	Power2
Profondeur pipeline			
Entier Flottant	7 ≥ 9	5 ≥ 7 (4 dans le cas d'opérations courtes)	6 ≥ 8
Chargement et séquencement des instructions			
Nbre d'étages	4	2	2
Nbre d'instructions traitées par cycle	4	4	6 (+ 2 instructions supplémentaires inspectées pour d'éventuels branchements)
Gestion des conflits de contrôle	Table de prédiction à 2 bits (2 K-entrées, 1 par instruction)	Cache de branchement (1 K-entrées, 1 pour 4 instructions)	2 unités de décodage partiel (fenêtre de 8 instructions), tampon d'instructions cible, émission conditionnelle.
Gestion des interblocages	Centralisée sur l'Ibox	Scoreboarding et exécution synchrone des instructions entières. Découplage des instructions flottantes (FPQ, LDQ, SAQ et SDQ).	Décentralisée. Unité entière : scoreboarding. Unité flottante : découplage des opérations par l'intermédiaire de files, renommage de registres, champ de comptage des écritures, ...
Gestion des exceptions et interruptions	Imprécise et précises Pas de mode spécifique de gestion précise	Instructions entières : gestion précise Instructions flottantes : gestion imprécise Existence d'un mode précis sur le processeur	

TAB. 3.2 - *Récapitulatif: chargement et séquencement des instructions*

	DEC 21164	MIPS R8000	Power2
Nbre d'instructions lancées simultanément par unité			
Instructions entières	2	4 (2 opérations arithmétiques + 2 calculs d'adresse)	(2 branchements (ou 1 branchement et 1 code condition sur l'ICU)) + 2 instructions arithmétiques
Instructions flottantes	2	2 opérations dont 2 MADD	
Instructions mémoires	2 lectures ou 1 écriture (le calcul de l'adresse remplace alors une opération dans le pipeline entier)	2 lectures ou 1 lecture et une écriture en plus des opérations arithmétiques	2 lecture/écriture (ou combinaison lecture-écriture). Le calcul de l'adresse occupe alors le pipeline arithmétique.

TAB. 3.3 - *Récapitulatif: exécution des instructions*

Chapitre 4

Unité flottante

Les trois microprocesseurs étudiés affichent des performances potentielles en flottant impressionnantes (600 Mflops pour le DEC 21164, et respectivement 300 et 287 Mflops pour le MIPS R8000 et le Power2) et visent clairement le marché du calcul scientifique. SGI présentant même le MIPS R8000 comme orienté vers le calcul vectoriel.

Il y a quelques années encore, les opérateurs flottants se présentaient sous la forme de coprocesseurs. L'évolution technologique a permis de regrouper sur un même composant opérateurs entiers et flottants, comme le démontre la génération précédente des processeurs étudiés [2] ainsi que le DEC 21164. Cependant, les progrés réalisés ouvrant toujours de nouveaux horizons, le superscalaire introduit la multiplicité des opérateurs pour augmenter le parallélisme au niveau de l'exécution des instructions. Le regain de surface nécessité par ce nouveau concept ainsi que l'accroissement des éléments mémoires font que le Power2 et le MIPS R8000 mettent en œuvre des unités flottantes sous la forme de deux coprocesseurs.

Les concepteurs ont à faire un certain nombre de choix pour optimiser le rapport coût/performance de ces unités. Divers critères sont à considérer telle que le choix de la mise en œuvre (matérielle ou logicielle), l'équilibre entre entier et flottant, et enfin les caractéristiques des opérateurs (rapidité des algorithmes implémentés, multiplication-addition versus multiplieur plus additionneur). Les choix sont conditionnés par les domaines d'applications pour lesquels les processeurs sont prévus.

4.1 Norme IEEE 754

Les trois processeurs étudiés implémentent la norme IEEE *Standard for binary Floating-Point Arithmetic* (ANSI/IEEE Standard 754 - 1985). Ce standard a été défini dans le but d'améliorer la qualité du calcul flottant et la portabilité des applications. Ce standard est maintenant utilisé par tous les acteurs principaux du calcul scientifique. Il supporte deux types de format 32 et 64 bits (confère figure 4.1) et 4 modes d'arrondis (vers $+\infty$, vers $-\infty$, vers 0, au plus près).

La signification des trois champs est la même pour les formats simple et double précision. Le bit de poids fort représente le bit de signe, l'exposant est un nombre représenté en utilisant une méthode de base, et enfin la *mantisse* désigne une valeur inféreure à 1 mais dont la signification implicite est 1 plus ce champ. En d'autres termes, la représentation des nombres flottants est

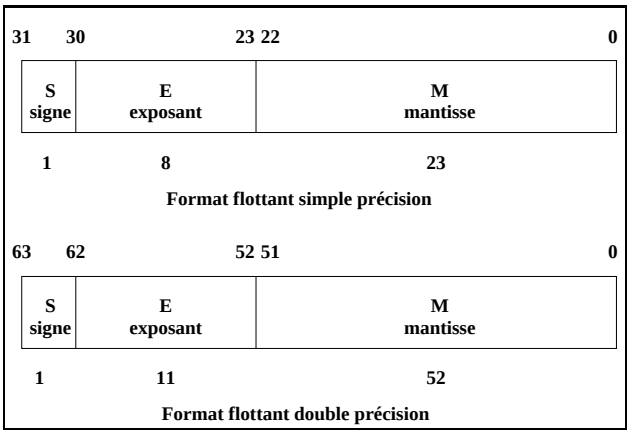


FIG. 4.1 - *Format des nombres flottants défini par la norme IEEE 754*

conforme à la formule $(-1)^S$ x *(1 + M)* x $2^{(Exposant-biais)}$ avec dans le cas des nombres simple précision : biais = 127 et double précision : biais = 1023.

Pour des raisons de compatibilité, le DEC 21164 supporte les types de données VAX F et G et fournit aussi un support limité pour le type D (voir [2] pour plus de détails sur ces formats).

Le DEC 21164 et le MIPS R8000 supportent les formats 32 et 64 bits, tandis que le Power2 effectue toutes les opérations sur 64 bits (conversion implicite en entrée, explicite en sortie).

4.2 Le DEC 21164

Constitution L'unité flottante du DEC est organisée autour de deux pipelines d'exécution :

- un pipeline de multiplication flottante (FM),
- un pipeline d'addition (FA) auquel est associé un pipeline de division flottante non pipeliné. Ce pipeline exécute aussi les branchements sur condition flottante.

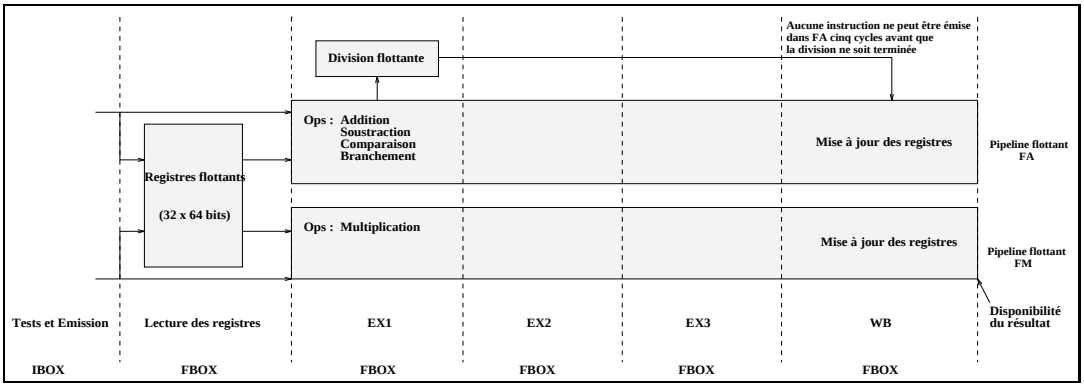


FIG. 4.2 - *Pipelines flottants du DEC 21164*

Un registre de contrôle, le *Floating-Point Control Register*, accessible en mode utilisateur est destiné au contrôle des modes d'arrondis et des informations d'exceptions. L'architecture DEC

définit un banc de 32 registres de 64 bits. Ce banc a cinq ports de lecture et quatre ports d'écriture. Quatre des ports de lecture fournissent les opérandes aux pipelines. Le cinquième est destiné aux opérations de mises à jour du cache de données. Deux des ports d'écriture servent pour les opérations de lecture flottante, les deux autres reçoivent les résultats des deux pipelines.

Fonctionnement Pour chacun des pipelines FA et FM, l'unité flottante peut accepter une instruction par cycle, à l'exclusion des divisions (opérateur non pipeliné). La plupart des opérations sont exécutées en 4 cycles (six cycles étaient nécessaires sur le DEC 21064) avec un taux d'émission d'une multiplication et d'une addition (ou opération similaire) par cycle. La latence de la division varie de 15 à 31 cycles en simple précision (19 en moyenne), et de 22 à 60 cycles en double précision (31 en moyenne).

4.3 Le MIPS R8000

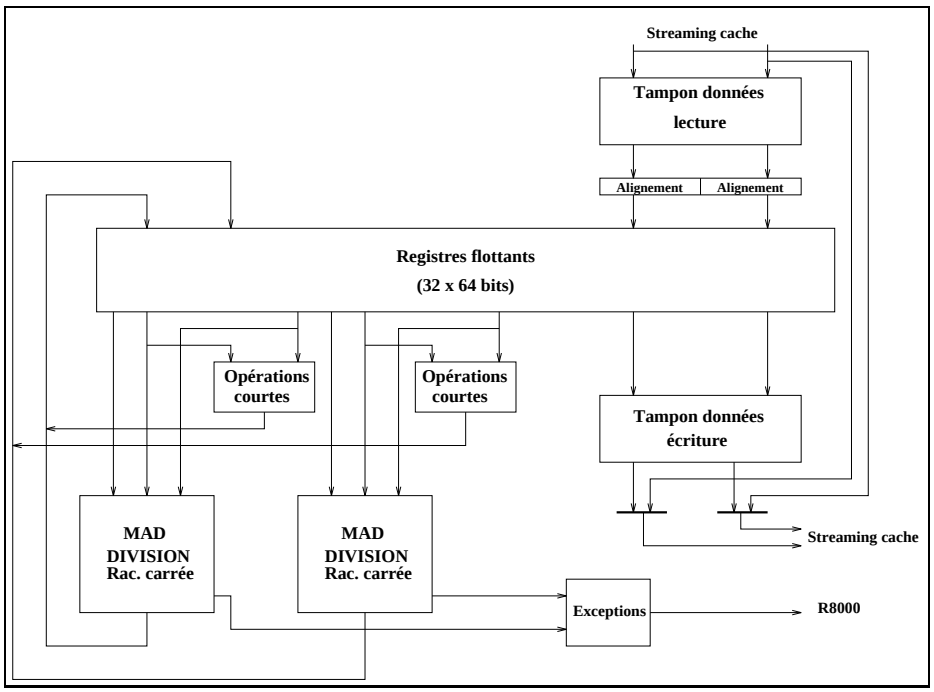


FIG. 4.3 - *Unité flottante du MIPS R8000*

Le R8010 FPU est un circuit de 591 broches conçu par Weitek et qui joue le rôle d'un coprocesseur pour le R8000 en effectuant la totalité des opérations flottantes. Il dispose pour cela de deux unités d'exécution à 3 opérandes, de 32 registres flottants de 64 bits et de deux tampons de données à 32 entrées destinés aux opérations de lecture et d'écriture. Le banc de registres contient 8 ports de lecture dont 6 sont nécessaires pour les deux unités d'exécution totalement symétriques, et 4 ports d'écriture dont deux sont utilisés pour les résultats. Les ports restant permettant deux opérations de load/store par cycle. L'unité flottante du MIPS R8000 est conforme au schéma de la figure 4.3.

Fonctionnement Jusqu'à quatre instructions flottantes peuvent être émises par l'unité entière à chaque cycle vers la *Floating Point Queue* (15 entrées). Deux instructions au maximum, sont émises par cycle, dans l'ordre, à partir de la *FPQ*, où elles ont attendues la disponibilité de leur opérandes. Cette file fonctionne donc en synergie avec les deux files de chargement de données (d'une capacité de 32 entrées) qui ont permis d'accèder les données requises par les instructions tout en masquant la latence du *Streaming Cache* (voir chapitre 5.3).

La FPU exécute trois types d'opérations de base : court, régulier et long, qui correspondent au temps d'occupation de l'unité d'exécution. Chaque unité d'exécution est complètement pipelinée et peut débuter une opération à chaque cycle (sauf pour les opérations longues qui occupent l'un des étages du pipeline pendant 3 cycles). Les opérations dites courtes (*MOV, MOVC, ABS, NEG, ...*) ont une latence de un cycle et peuvent être exécutée en parallèle avec des opérations plus longues (excepté les instructions de comparaison). Le tableau 4.1 reprend les diverses valeurs pour chacun des différents types.

La plupart des nouvelles extensions du jeu d'instructions MIPS IV du MIPS R8000 visent l'augmentation des performances flottantes. L'architecture MIPS IV inclut pour cela 4 nouvelles instructions de multiplication-addition flottantes à trois opérandes ($\pm(A \times B \pm C) \rightarrow D$) qui interviennent dans un grand nombre d'algorithmes scientifiques lors des calculs de sommes de produits. Ces instructions devraient favoriser l'exécution des codes numériques vectorisables. De-même que l'opération *Fmadd* du Power1, l'opérateur n'est pas conforme à la norme IEEE 754 car il n'effectue pas d'arrondi entre la multiplication et l'addition. Ceci permet de diminuer la latence de l'opération.

Type	Opération	Durée d'occupation de l'opérateur		Latence	
		SP	DP	SP	DP
Court	MOV, MOVF, ...	1 cycle	1 cycle	1 cycle	1 cycle
	NEG, ABS, C	1 cycle	1 cycle	1 cycle	1 cycle
Régulier	Load	1 cycle	1 cycle	5 cycles	5 cycles
	Store	1 cycle	1 cycle	n/a	n/a
Régulier	Add./soust.	1 cycle	1 cycle	4 cycles	4 cycles
	Conversion	1 cycle	1 cycle	4 cycles	4 cycles
Régulier	Multiplication	1 cycle	1 cycle	4 cycles	4 cycles
	MADD	1 cycle	1 cycle	4 cycles	4 cycles
Long	Division	11 cycles	17 cycles	14 cycles	20 cycles
	Racine carrée	11 cycles	20 cycles	14 cycles	23 cycles

TAB. 4.1 - *Caractéristiques des opérations flottantes simples et doubles précisions sur le MIPS R8000*

Les concepteurs du MIPS R8000 ont plus visé les performances de crête que l'obtention d'une latence minimale sur les opérations (confère tableau 4.1). Pour eux, la latence n'est vitale que lorsqu'une instruction nécessite le résultat d'une opération précédente, situation qui tendrait à diminuer selon eux grâce aux compilateurs modernes. L'argument semble d'autant plus acceptable que MIPS présente ce processeur comme un processeur dédié au calcul numérique. La plupart des

opérations ont une latence de 4 cycles, ce qui représente 53 ns à 75 MHz par rapport aux 28 ns de latence moyenne sur le Power2 et aux 13 ns du DEC 21164 (séquencé à 300 MHz).

4.4 Le Power2

La figure 4.4 présente les diverses unités fonctionnelles qui constituent la FPU. Elle est organisée autour d'un fichier de 54 registres physiques de 64 bits dotés de 8 ports en lecture et 4 ports en écriture. Elle contient également trois tampons d'instructions qui alimentent les deux pipelines d'exécution pour chacun des trois types d'instructions représentés sur la figure. La FPU dispose aussi d'un registre appelé le *Floating-Point Statut and Control Register* qui comme son nom l'indique, contrôle les arrondis et gère les exceptions.

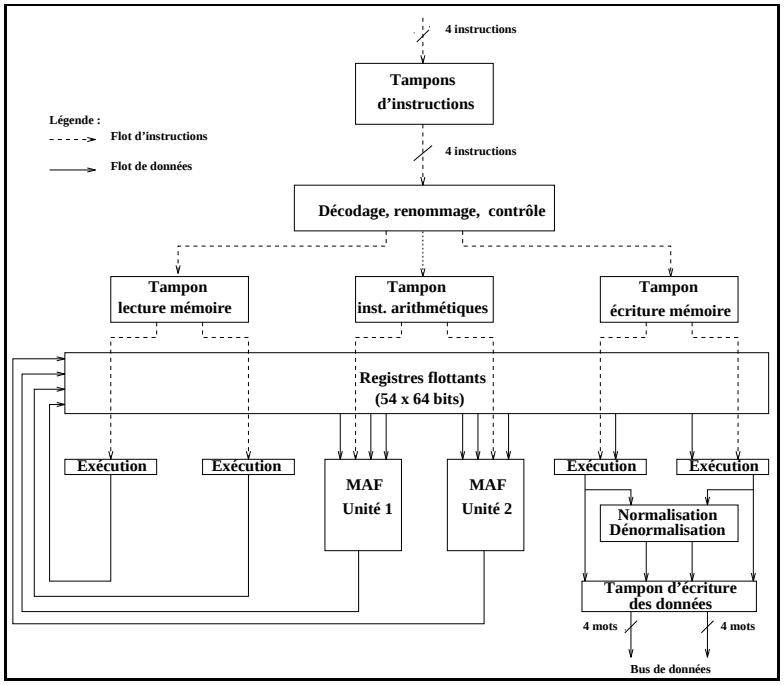


FIG. 4.4 - *Unité flottante du POWER2*

Fonctionnement La FPU est susceptible de recevoir quatre instructions par cycle de l'ICU. Ces instructions sont stockées dans un tampon à 8 entrées ou directement envoyées vers les étages de prédécodage et renommage selon la disponibilité de ceux-ci (voir chapitre 3.2.3 pour plus de détails). Les instructions sont ensuite réparties dans les tampons relatifs aux trois types d'instructions susceptibles d'être reçu : lecture d'une opérande, écriture d'une donnée, exécution d'une opération. Chacun de ces tampons alimente deux pipelines d'exécution. Les unités travaillent indépendamment l'une de l'autre, et les instructions peuvent donc s'exécuter dans le désordre si il n'y a pas de dépendance de données. Cependant, seules deux instructions de *Load/Store* peuvent avoir lieu simultanément dans la mesure où les calculs d'adresse sont effectués par l'unité entière qui ne dispose que de deux ALUs. À noter que la FPU et la FXU reçoivent les instructions de load/store flottants.

Le Power2 met en œuvre la technique du renommage de registres qui permet de résoudre les dépendances de données de type WAR et WAW.

La figure 4.5 illustre de manière simple l'intérêt d'un tel mécanisme.

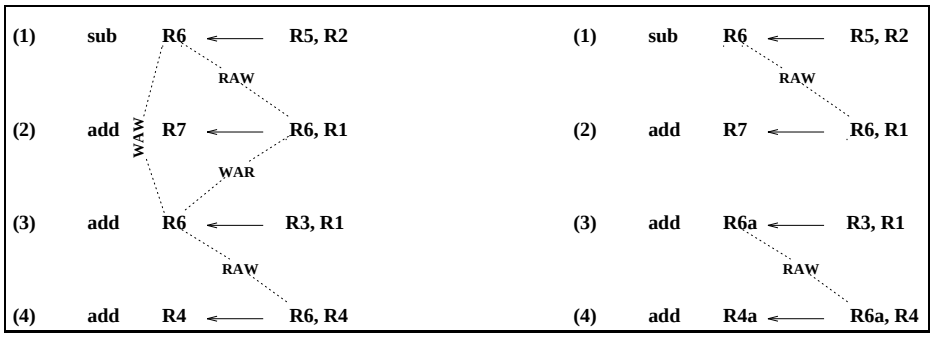


FIG. 4.5 - *Renommage de registres*

Dans le cas des boucles, ce mécanisme va permettre de manière simple le recouvrement des opérations grâce aux registres supplémentaires susceptible de stocker des résultats. Il permet également d'anticiper le chargement des opérandes d'une itération sur l'autre. En ce sens, ce mécanisme est à rapprocher du découplage des opérations flottantes sur le MIPS R8000. La FPQ associée à la file de chargement de données permet d'anticiper le chargement des opérandes de la mémoire et traite les aléas WAW sur les registres flottants quand le second *Write* est une instruction de chargement. Le mécanisme de renommage sur les registres est plus général car il traite les aléas WAW quelle que soit l'origine du *Write*.

Les unités d'exécution de la FPU sont identiques. Elles sont aussi plus autonomes que celles de l'unité entière. En effet, la FXU n'envoie une instruction vers sa deuxième unité fonctionnelle que si deux instructions sont prêtes à être émises simultanément. Sur la FPU, quand une instruction est prête à être émise et qu'une unité d'exécution est disponible, elle est envoyée. Les deux unités arithmétiques sont capables d'exécuter n'importe quelle opération flottante de registre à registre.

Le tampon d'instructions arithmétique a une structure particulière, car il doit alimenter les deux unités de calcul à partir d'une file commune. Aussi, quand une unité est occupée pendant plusieurs cycles, le tampon peut émettre des instructions si la deuxième unité est disponible (*out of order completion*). La structure de ce tampon offre également un support limité à l'émission des instructions dans le désordre (dépassement possible d'une instruction dépendante d'une opération en cours d'exécution).

L'analyse des performances du Power1 a conduit les concepteurs à une mise en œuvre matérielle de la racine carrée. Celle-ci permet désormais de gagner la moitié des cycles par rapport à l'émulation logicielle effectuée sur le Power1. L'implémentation des nouvelles instructions de conversion flottant-entier contribue aussi à améliorer les performances[1].
L'instruction de multiplication-addition, associée à l'instruction de chargement de mots de 128 bits, permet d'obtenir des performances très élevées sur les noyaux de calculs intensifs. On pourra consulter la table présentée dans [14] à titre indicatif. Le tableau 4.2 récapitule les latences des principales opérations.

1. Comment expliquer une telle bévue de la part des concepteurs du Power1 ?

Opération	Power2 Cycles (14 ns)	Power1 Cycles (16 ns)
FP Add	2	2
FP MUL	2	2
FP MUL-ADD	2	2
FP Div	17	20
FP Racine carrée	27	53

TAB. 4.2 - *Latences des opérations flottantes sur le POWER2*

À l'exception de la division et de la racine carrée, toutes les opérations sont pipelinées et peuvent être émises à chaque cycle.

Synchronisation entre unité entière et unité flottante Les unités entière et flottante reçoivent deux types d'instructions de la part de l'ICU : les instructions qui leur sont spécifiques et qu'elles exécuteront et des instructions interruptibles. Ces dernières regroupent les opérations d'accès à la mémoire (*load* et *store*) et les *fixed-point trap instructions* qui permettent de tester un ensemble de conditions sous lesquelles une exception sera déclenchée. Une synchronisation existe entre la FXU et la FPU au niveau de chacune de ces *IOP* (opération interruptible). Chaque unité peut progresser indépendamment jusqu'à ce qu'elle rencontre une IOP. Si la FPU est la première à la rencontrer, elle attendra que la FXU effectue le calcul de l'adresse (ou le test de l'exception) et lui signale qu'elle peut reprendre son exécution. Si le cas inverse se produit, la FXU traitera directement l'IOP et informera la FPU qu'elle peut continuer son exécution sans tenir compte d'une possible interruption.

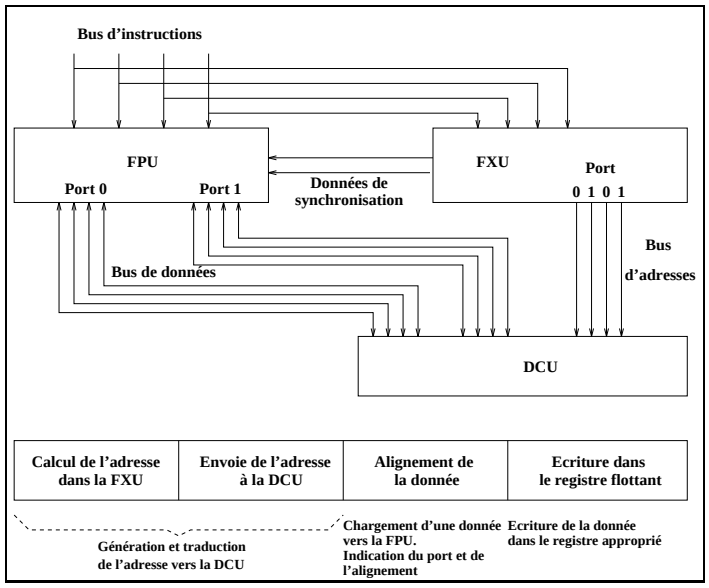


FIG. 4.6 - *Synchronisation entre la FXU et la FPU sur le POWER2*

La figure 4.6 reprend ce schéma de synchronisation sur les opérations mémoires. Lors d'une instruction de chargement flottant, le calcul de l'adresse est effectué dans la FXU au même niveau que l'étage de renommage dans la FPU. Au cycle suivant, la donnée sera lue dans le cache de données, puis placée dans le registre cible déterminé lors de l'opération de renommage.
La synchronisation entre la FXU et la FPU assure l'intégrité de l'association entre les données et les instructions et préserve des interruptions précises.

4.5 Conclusion

Ces trois microprocesseurs mettent en œuvre des techniques différentes pour permettre des performances flottantes élevées sur les applications. Le DEC 21164 avec sa haute fréquence d'horloge privilégie le taux d'émission des instructions.

Le MIPS R8000 met en œuvre un découplage des instructions flottantes. Ce découplage masque la latence du cache externe et permet d'anticiper le chargement des opérandes.

Le Power2 utilise une technique différente en implémentant un nombre de registres physiques supérieur au nombre de registres vu par le compilateur. Cette technique qui permet elle aussi de précharger les données, contribue également à la résolution des conflits de type WAR et WAW.

Les instructions de multiplication-addition ont déjà démontré leur efficacité sur le Power1. L'introduction de celles-ci dans le jeu d'instructions MIPS IV et le doublement des opérateurs sur le Power2, devraient permettre à ces architectures d'atteindre les performances annoncées. De plus, l'élargissement des chemins de données sur le Power2 devrait sans nul doute contribuer à accroître les performances à travers les accès à la mémoire de 128 bits sur les applications vectorielles accédant à des tableaux contigus en mémoire.

4.6 Récapitulatif: unités arithmétiques

Les tableaux ci-dessous reprennent les diverses latences d'opérations pour les trois processeurs étudiés. Ils mettent en avant un certain nombre de discontinuité dans les performances. Il nous a semblé intéressant de rappeler le temps de cycle de chacune des machines afin de relativiser les différences.

Processeurs	**DEC 21164**	**MIPS R8000**	**Power2**
décodage instructions	Ibox	IU	FXU et FPU
calcul adresses opérandes	Ebox	IU	FXU

Unité entière	Ebox	IU	FXU
Registres	32 x 64 bits	32 x 64 bits	32 x 36 bits
Ports (lecture/écriture)	4/2	9/4	7/4
Opérateurs entiers	2 ALUs Multiplication	2 ALUs Décalage et mult. 2 générateurs d'adr.	2 ALUs Code condition
Profondeur pipeline	7	5	6

Unité flottante	Fbox	FPC	FPU
Registres	32 x 64 bits	32 x 64 bits	54 x 64 bits (physiques) 32 x 64 bits (logiques)
Ports (lecture/écriture)	5/4	8/4	8/4 (dont 4 x 128 bits)
Opérateurs flottants	Multiplication Addition Division	2 Mul-add 2 unités op. courtes	2 Mul-add
Profondeur pipeline	≥ 9	≥ 7	≥ 8

TAB. 4.3 - *Caratéristiques des unités arithmétiques*

Processeurs / Opérations	DEC 21164 Cycles (3,33 ns)[a]	MIPS R8000 Cycles (13 ns)	Power2 Cycles (14 ns)
arithmétiques	1	1	1
Décalage	1	1	1
Multiplication	8 - 16	4 - 6	2
Division	fonction logicielle[b]	21 - 73	13 - 17

TAB. 4.4 - *Latences des opérations entières*

[a] Fréquence de 300 MHz
[b] Ou opérateur de multiplication quand division par une constante

Processeurs / Opérations	DEC 21164 Cycles (3,33 ns)[a]	MIPS R8000 Cycles (13 ns)	Power2 Cycles (14 ns)
FP opérations	4	4	4
FP MUL	4	4	2
FP MUL-ADD	non défini	4	2
FP Div (SP/DP)	19/31	14/20	17[b]
FP Racine carrée (SP/DP)	?	14/23	27

TAB. 4.5 - *Latences des opérations flottantes*

[a] Fréquence de 300 MHz
[b] Pas de simple précision

Chapitre 5

Hiérarchie mémoire

5.1 Introduction

L'objectif de performances des premiers processeurs RISC était de lancer une instruction par cycle ; sur les microprocesseurs superscalaires actuels, on souhaite atteindre de quatre à six instructions par cycle. Ces architectures posent le problème de conserver remplis les pipelines avec des instructions et des données. Ceci est de plus en plus difficile à réaliser. Les performances des CPUs doublent tous les 18 - 24 mois, alors que les temps d'accès et de cycle (respectivement 60 et 100 ns actuellement) des composants DRAM qui constituent la mémoire principale ne diminuent que de quelques pourcents chaque année. De plus, rappelons que le temps d'accès à une mémoire principale comporte la traversée de plusieurs couches de logique. Le temps d'accès serait donc plutôt de l'ordre de 150 à 300 ns. Même si le débit de la mémoire peut être très élevé, le temps d'accès est considérable par rapport aux temps de cycle des microprocesseurs étudiés : 3,33 ns pour le DEC 21164 (300 MHz), 14 ns pour le Power2 et 13 ns pour le MIPS R8000.

Dès lors, la mise en œuvre entre le processeur et la mémoire principale de mécanismes masquant la latence de cette mémoire est la clé des performances. Le principal de ces mécanismes est le cache. Plusieurs niveaux de caches sont maintenant couramment utilisés sur les processeurs. Ces derniers sont organisés en une hiérarchie mémoire représentée figure 5.1 .

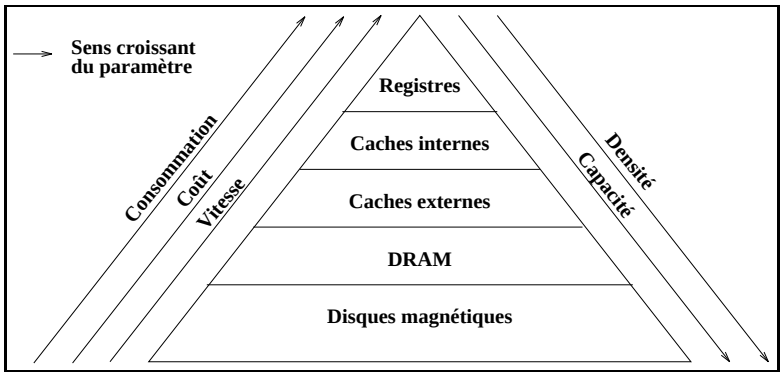


FIG. 5.1 - *Hiérarchie mémoire*

Les registres sont au sommet de la hiérarchie. Ils peuvent être lus ou écrits en un cycle sans calcul d'adresse. Les caches primaires constituent le niveau suivant de la hiérarchie. Les tailles typiques de ces caches primaires sont de 8 Ko à 256 Ko, et leur temps d'accès est en général de 2 à 3 cycles (en incluant le calcul d'adresse). Depuis que l'évolution des possibilités d'intégration a permis l'implantation de ces caches sur la même puce que le CPU, la plupart des processeurs peuvent fonctionner avec de plus un second niveau de cache externe. Le DEC 21164 est le premier processeur à intégrer sur le composant un cache secondaire, et les concepteurs conseillent d'ajouter un troisième niveau de cache externe.

Conséquence de cette hiérarchie mémoire qui prend de plus en plus d'ampleur, ces processeurs intègrent, entre chacun des niveaux de cache, d'autres mécanismes destinés à masquer le plus possible la latence du sous-système mémoire. L'émergence de ces mécanismes, et leur ingéniosité, témoigne de la difficulté de tenir constamment occupées ces architectures superscalaires, et ce malgré la relative lenteur de la mémoire.

5.2 Généralités sur les caches

Nous rappelons ici quelques définitions usuelles sur les structures de caches avant d'étudier de manière plus détaillée la hiérarchie mémoire de chacun des processeurs.

5.2.1 Placement des données

Un cache est constitué de lignes qui peuvent être organisées de diverses façons :

- Organisation à correspondance directe avec la mémoire (*direct-mapped*). Chaque bloc de la mémoire principale ne peut être chargé qu'à un seul endroit du cache. En général, la fonction de correspondance est un simple calcul de modulo. Cette organisation de cache est la plus simple.

- Organisation totalement associative (*fully associative*). Chaque bloc de la mémoire principale peut être chargé à n'importe quel endroit du cache.

- Organisation associative par ensembles (*set associative*). Le cache est divisé en ensembles d'emplacements et chaque bloc de la mémoire principale ne peut être chargé qu'à l'intérieur d'un des ensembles. Cette organisation est un compromis entre les deux organisations précédentes.

5.2.2 Stratégies de remplacement

Le chargement d'une ligne dans un cache se fait toujours au détriment d'une ligne déjà présente. Un mécanisme matériel arbitre le remplacement des lignes dans un ensemble. Sur les caches associatifs, plusieurs stratégies de remplacement sont couramment utilisées :

- La stratégie de choix aléatoire (*Random*) : la ligne est choisie de manière aléatoire parmi les lignes possibles. Cette stratégie peu coûteuse en logique est peu fiable du point de vue des performances.

– La stratégie LRU (*Least Recently Used*) : la ligne choisie est celle qui a été la plus anciennement référencée. Cette stratégie donne en général de bons résultats, mais devient assez coûteuse lorsque l'associativité croît.

Ces stratégies sont assez simples, mais pas toujours satisfaisantes. De nombreuses stratégies ont été envisagées, mais le problème reste ouvert.

5.2.3 Politique d'écriture

Les accès en lecture sur le cache sont les plus nombreux (lecture d'opérandes, accès aux instructions) et ne posent pas de réels problèmes, alors que les accès en écriture, moins nombreux, nécessitent une gestion rigoureuse de la cohérence des données au travers de la hiérarchie mémoire. Là encore, diverses politiques sont mises en œuvre lors de la modification d'une donnée :

– Écriture simultanée (*write-through*) : lorsqu'une ligne du cache est modifiée, cette ligne est écrite tout de suite dans un niveau mémoire inférieur de la hiérarchie. Cette politique garantit la cohérence entre les niveaux mais augmente le trafic sur le bus de données. Généralement, pour éviter que les écritures entravent le fonctionnement de l'UC, elles sont placées dans un tampon d'écriture ou *write-buffer* et effectuées pendant que l'UC se consacre à d'autres tâches.

– Recopie ou *write-back* : la ligne est marquée *modifiée* et elle sera recopiée en mémoire losqu'elle sera la cible d'un remplacement.

De plus, lorsque l'UC veut écrire une donnée dans une ligne absente du cache, l'une des deux options suivantes est généralement utilisée :

– écriture attribuée ou *write-allocate* : la ligne est chargée dans le cache avant d'être modifiée.

– écriture non attribuée ou *no write allocate* : la ligne est modifiée directement dans le niveau inférieur de la hiérarchie et non chargée dans le cache.

L'option *write-allocate* est généralement associée à la politique d'écriture *write-back*, et l'option *no write-allocate* à la politique d'écriture *write-through* (ce qui est cohérent dans la mesure où une écriture nécessitera sa mise à jour directement dans la mémoire, on évite ainsi des transferts supplémentaires).

5.2.4 Répartition physique des caches entre données et instructions

Les 3 microprocesseurs étudiés utilisent deux caches primaires séparés pour les données et les instructions. Chacun des caches est relié à l'UC par un bus. Cette configuration permet de réaliser au même cycle un accès à une donnée et à une instruction sans avoir recours à un cache multiport (plus coûteux à implémenter comme nous le verrons plus loin) et elle évite les interférences de référence mémoire entre les données et les instructions. Le cache secondaire, généralement moins sollicité est partagé (DEC 21164 et MIPS R8000).

5.2.5 Adressage physique ou virtuel

Un cache peut être indexé soit par l'adresse virtuelle directement fournie par le programme, soit par l'adresse physique après traduction de l'adresse virtuelle. L'adressage virtuel offre un temps d'accès au cache plus court car aucun mécanisme de traduction d'adresse ne doit être traversé avant l'indexage du cache, mais il induit des problèmes de cohérence de données entre processus.

Remarque : la terminologie n'est pas uniforme selon les constructeurs en ce qui concerne le sens des termes *secteur* et *ligne*. Nous avons conservé les notations utilisées dans [3], à savoir :

- le secteur est associé à une étiquette ;
- la ligne désigne l'unité de cohérence et de transfert.

Nous allons maintenant voir de quelle manière chacun des contructeurs a apporté une solution à l'organisation de la hiérarchie mémoire autour du CPU en répondant aux critères énoncés dans ce chapitre.

5.3 Les caches du MIPS R8000

Le MIPS R8000 met en œuvre une structure de cache particulière. Il comporte deux caches internes, d'instructions et de données, localisés sur l'unité entière. Ce cache de données n'est accédé que par les load/store entiers. Un second niveau de cache est également implémenté par le biais d'un large cache externe. Ce cache, qui alimente en données les unités flottante et entière, n'est accessible en écriture qu'à travers l'unité flottante.

5.3.1 Caches internes

Le cache d'instructions

Le cache d'instructions du MIPS R8000 a une capacité de 16 Ko et est à correspondance directe. La figure 5.2 présente son organisation ainsi que celle du cache d'étiquettes. Il est structuré en 1024 lignes de quatre instructions. La taille d'un secteur est de 32 octets, soit deux de ces lignes.

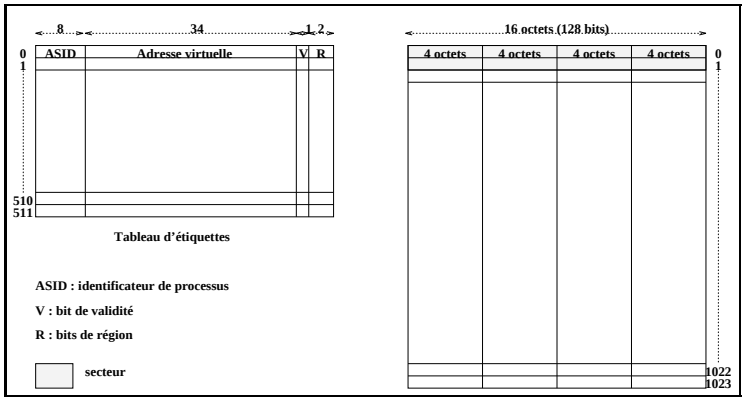


FIG. 5.2 - *Cache d'instructions du MIPS R8000*

Le cache d'instructions est adressé virtuellement. Ce choix permet d'utiliser directement l'adresse venant du programme et simplifie le mécanisme de prédiction de branchement mis en œuvre sur le MIPS R8000. La cohérence des instructions doit être assurée par logiciel. Cependant, en cas de changement de contexte, il n'est pas nécessaire de vider le cache dans la mesure ou un identificateur de processus est associé à chaque étiquette. Chaque entrée du cache d'étiquettes est constituée de 34 bits d'adresse virtuelle, de 8 bits d'identificateur d'espace d'adresse (ASID), de deux bits de régions (voir chapitre 6.5.2) et d'un bit de validité.

Le cache de données

Le cache de données du MIPS R8000 n'est accédé qu'à travers les load/store entiers. Les données flottantes sont accédées à partir du cache externe.
Le cache de données a aussi une capacité de 16 Ko et est à correspondance directe. La taille des secteurs est de 32 octets, celle des lignes de 16 octets. De plus, un bit de validité est associé à chaque mot de quatre octets (confére figure 5.3). Il supporte deux accès par cycle (soit deux lectures, soit une lecture et une écriture). Il met en œuvre une politique d'écriture simultanée (*write-through*). Ce cache est virtuellement indexé et physiquement *taggé*. La lecture du cache et des étiquettes est effectuée en parallèle avec la traduction de l'adresse virtuelle en adresse physique.

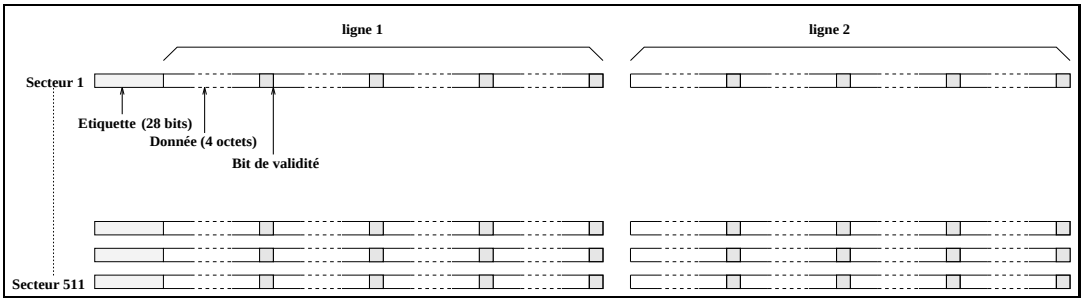


FIG. 5.3 - *Cache de données du MIPS R8000*

L'unité entière met en œuvre un mécanisme original lors des écritures. Profitant de la structure *direct-mapped* du cache, la donnée est systématiquement écrite dans le cache à l'unique place possible pendant le test de l'étiquette. Puis, si le test de l'étiquette révèle que le secteur est absent du cache, alors le (ou les) bit(s) de validité du mot écrit est (sont) mis à zéro. Ce type de fonctionnement n'est permis que parce que le cache de données est à correspondance directe et à recopie simultanée.

Par ailleurs, du fait de la particularité de ce cache qui n'est accessible que par l'unité entière, les bits de validité permettent de maintenir la cohérence d'une ligne quand des données entières et flottantes résident dans la même ligne. En ayant une granularité d'un bit de validité par mot de 32 bits, le mélange de données entières et flottantes s'en trouve accomodé et évite l'invalidation de toute la ligne du cache.

Pénalités sur les défauts de cache interne

Les caches de données et d'instructions sont tous les deux remplis à partir du cache externe qui délivre 16 octets à chaque accès. Le temps de remplissage du cache de données est de sept cycles : cinq cycles pour accéder aux RAMs du cache externe et deux cycles pour écrire les 32 octets d'un

secteur.
Le remplissage du cache d'instructions prend plus de temps (11 cycles). En effet, en cas de défaut sur le cache d'instructions, l'exécution effective de l'instruction doit d'abord être confirmée par la prédiction de branchement, puis l'adresse transite par la table de traduction d'adresses avant d'être validée vers le cache externe. Lors d'un défaut sur le cache externe, la pénalité dépend de la structure de la hiérarchie implémentée dans le système, cependant il semble difficile de mettre en œuvre un système où cette pénalité serait inférieure à 50 cycles !

5.3.2 Cache externe

Les applications numériques flottantes requièrent en général un ensemble de travail, ou *working-set*, relativement volumineux. Les concepteurs de MIPS ont privilégié cet aspect ainsi que la bande passante au détriment du temps d'accès. Le *streaming cache* a une taille configurable de 4 à 16 Mo correspondant à des secteurs de 128, 256 ou 512 octets. Il est associatif par ensemble à quatre voies et il met en œuvre une politique d'écriture de type *write-back*. Les secteurs sont divisés en quatre lignes.

Ce cache joue le rôle d'un cache secondaire pour le premier niveau de cache d'instructions et de données, et constitue un cache primaire vis à vis de l'unité flottante. Aussi, l'efficacité de ce cache est particulièrement déterminante sur les performances du système.
Les concepteurs ont choisi de faire appel à une technologie récente de mémoires synchrones (c'est à dire utilisant une horloge).

Ainsi, l'utilisation de ces mémoires, de 12 ns de temps de cycle, organisées en deux bancs entrelacés, a permis d'atteindre une bande passante de 1.2 Go par seconde. Ce type d'organisation est sujet au traditionnel problème du conflit de banc qui peut provoquer une perte de 50 % de la bande passante quand deux accès mémoires adressent le même banc (ils sont alors servis séquentiellement, le deuxième port n'étant pas sollicité). Afin d'obtenir un débit élevé le plus proche possible de deux accès par cycle, un tampon d'une seule entrée mémorise une des requêtes dans le cas d'un conflit de banc. Ce mécanisme permet de résoudre les problèmes d'alignement locaux alors qu'on peut confier au compilateur la responsabilité de l'équilibrage des accès aux bancs (par exemple en déroulant deux itérations d'une boucle). Ceci illustre la synergie d'une solution matérielle et logicielle pour optimiser le fonctionnement du cache externe.

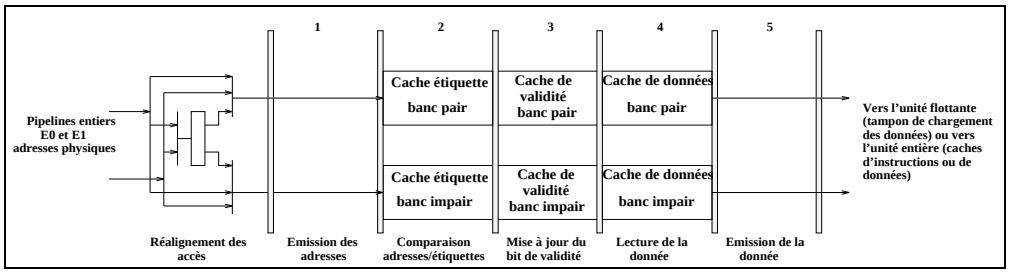


FIG. 5.4 - *Pipeline d'accès au cache secondaire du MIPS R8000*

Le cache externe est physiquement indexé et *taggé*. Il est associatif par ensemble de degré 4. Sur les caches externes, l'utilisation d'un tel degré d'associativité est irréaliste si on veut accéder en même temps aux données et aux étiquettes (problème de *pin-out*). Les concepteurs de MIPS

ont résolu ce problème en pipelinant l'accès au cache secondaire : détermination de la place dans l'ensemble, puis lecture du mot dans le banc mémoire.

L'inconvénient de cette méthode est la latence du cache secondaire du fait de l'accès séquentiel, mais le fait de pipeliner les accès conformément au schéma 5.4 permet d'obtenir deux mots par cycle. Nous reviendrons plus loin sur l'impact modéré de cet allongement de la latence d'accès au *streaming cache* sur les performances des codes flottants.

Organisation des étiquettes

Le cache d'étiquettes est dupliqué sur deux composants (pour adresser les bancs pair et impair de la mémoire). Chaque composant contient 8 K-entrées de 128 bits de large. Ces 128 bits correspondent à quatre entrées de 32 bits associées chacune à un secteur de quatre lignes. Une RAM de bits de validité (un bit par ligne) est également implémentée dans le composant.

L'annexe D présente un exemple d'organisation mémoire de ce cache secondaire.

5.4 Les caches du Power2

La hiérarchie mémoire mise en œuvre sur le Power2 est plus classique que celle du MIPS R8000. Elle est constituée d'un cache d'instructions implanté sur l'unité de chargement et d'émission des instructions et d'un large cache de données.

5.4.1 Cache d'instructions

Comme on l'a vu précédemment, l'ICU charge et émet les instructions. Elle dispose pour cela d'un cache d'instructions de 32 Ko (soit 4 fois plus que le cache du Power). Ce cache est associatif par ensemble à deux voies et met en œuvre une stratégie de remplacement de type LRU. La taille de 128 octets des secteurs du cache (32 instructions) est relativement grande par rapport aux autres architectures mais du fait de sa capacité, elle permet de conserver un nombre relativement faible d'étiquettes. Le Power2 charge huit instructions consécutives par cycle, sans contrainte d'alignement. Le cache est organisé sous la forme de deux bancs mémoires, lignes paires et lignes impaires, ce qui permet de s'affranchir de l'alignement du bloc d'instructions sur les frontières de lignes sans avoir à implémenter une mémoire cache double accès. Les instructions doivent cependant appartenir à la même page.
Le cache d'instructions du Power2 est physiquement adressé et *taggé*.

Contrairement au cache du Power1 qui était inaccessible durant le chargement d'une ligne à partir de la mémoire, le Power2 met en œuvre un tampon de rechargement qui permet d'éviter ce problème : avec une ligne de 128 octets, ce serait 8 cycles qui serait perdus sans séquencement avec un bus de 16 octets. Les instructions parviennent donc au cache par l'intermédiaire du bus d'instructions. Un prédécodage crée un champ de six bits qui permet d'identifier le type de l'instruction. Ce champ est conservé au même titre que l'instruction dans le cache au moment du chargement.

5.4.2 Caches de données

Le cache de données du Power2 est associatif par ensemble à quatre voies. Il implémente un algorithme de remplacement des données de type LRU. Deux configurations sont disponibles :

- une configuration à 256 ko (4 composants) permet d'avoir des lignes de 256 octets et un bus vers la mémoire d'une largeur de 256 bits.
- une configuration à moindre coût (2 composants) impose une capacité de 128 Ko avec des lignes de 128 octets et un bus mémoire de 128 bits.

Dans les deux cas, le chargement d'une ligne de cache est effectué en 8 cycles.
Ces composants fournissent les données aux unités à travers :

- deux bus de données d'une largeur de un mot vers l'unité entière ;
- deux bus de données d'une largeur de quatre mots chacun vers l'unité flottante ;
- un bus à destination du cache d'instructions d'une largeur de quatre mots ;
- un bus système d'entrées / sorties destiné aux accès DMA d'une largeur de deux mots.

L'ensemble de ces bus fournit une bande passante de 2,2 Go par seconde.

Ce cache implémente une politique d'écriture de type *write-back*. Il est physiquement indexé et *taggé*.

Le cache de données du Power2 permet deux accès quelconques par cycle. À la différence du MIPS R8000, qui met en œuvre une mémoire entrelacée et des mécanismes pour optimiser son efficacité, ou, au lieu d'utiliser des cellules classiques à double ports qui augmentent considérablement la surface occupée, le Power2 utilise une technique appelée le *multiport virtuel*. La technologie à 0.6 μm employée pour les cellules de RAM statiques, leur permet de réagir plus rapidement que les 14 ns de temps de cycle du processeur. Elles sont alors à même de délivrer deux résultats par cycle. Les accès sont donc effectués séquentiellement, le premier résultat est délivré en 5.6 ns et les deux sont disponibles au bout de 9.2 ns. En cas de défaut de cache sur une donnée, le cache reste quand même accessible pour les accès suivants. Un second défaut bloque cependant tout accès. Ce cache dispose de deux ports spécialement dédiés aux opérations de chargement de données et de mise à jour de la mémoire. Ces ports constituent une ressource supplémentaire qui permet de minimiser la pénalité engendrée par ce type d'opération (cache de type *write-back*).

La DCU ne constitue qu'un cache de données. Les instructions ne font que traverser cette unité, ce qui permet de bénéficier des mécanismes de détection et de correction d'erreurs dispensés par cette unité tout en minimisant le nombre de broches vers l'extérieur (voir figure 2.1).

5.5 Les caches du DEC 21164

L'organisation de la hiérarchie mémoire du DEC 21164 comprend deux caches primaires d'instructions et de données et un cache secondaire unifié. Le DEC 21164 est le premier processeur à mettre en œuvre sur le même composant un premier niveau de cache et un cache secondaire de taille moyenne. Un troisième niveau de cache optionnel peut être implémenté en dehors du composant ; les mécanismes de contrôle de ce troisième cache sont fournis sur le processeur.

5.5.1 Caches primaires

Le DEC 21164 met en œuvre un premier niveau de cache (instructions et données) de petite taille. Ce choix privilégie un temps d'accès rapide et constitue un compromis entre la mise en œuvre d'un cache double-port et l'accroissement de la surface qui en résulte. Un cache secondaire, placé sur la même puce, complète efficacement ce premier niveau de caches.

Cache d'instructions

Le cache d'instructions primaire a une capacité de 8 Ko. Il est à correspondance directe, virtuellement adressé et *taggé*. Sa cohérence avec le cache secondaire doit être maintenue par logiciel. La taille des secteurs est de 32 octets, avec un bit de validité pour 16 octets. Chaque étiquette du cache d'instructions est constituée de trois champs en plus d'une adresse virtuelle :

- Un champ de 7 bits (*ASN* pour *Address Space Number*), défini par l'architecture Alpha, qui permet d'identifier un processus et aide à maintenir la cohérence dans le cache quand des changements de contextes ont lieu.
- Un champ de 1 bit *ASM* (pour *Address Space Match*) qui permet une invalidation rapide de certaines entrées.
- Un champ également de 1 bit destiné au Palcode.

Un seul port de lecture est mis en œuvre sur ce cache et permet de délivrer quatre instructions alignées (soit 16 octets) à chaque accès. Il est à noter qu'un champ de prédécodage est associé à chacune des intructions et conservé dans le cache (cinq bits).

Cache de données

Le cache de données du DEC 21164 a une capacité de 8 Ko. Il est également à correspondance directe avec des tailles de ligne de 32 octets. Il met en œuvre une politique d'écriture simultanée, non attribuée (*write-through, no allocate*). Il est adressé virtuellement et physiquement *taggé*.
Ce cache supporte deux lectures par cycle ou une seule écriture. Il est constitué de deux bancs identiques contenant les mêmes données et assignés à chacun des deux pipelines entiers. C'est pourquoi, seule une écriture est possible à chaque cycle, car la donnée doit être écrite dans les deux bancs pour que ceux-ci restent cohérents. Cette écriture ne peut pas être accompagnée d'une lecture. De plus, la structure du pipeline (voir figure 3.1, page 42) interdit le séquencement de lecture deux cycles après le séquencement d'une écriture.

5.5.2 Cache secondaire

Le DEC 21164 est le premier processeur à mettre en œuvre sur le même composant un cache secondaire. Ce cache est un cache mixte de 96 Ko d'instructions et de données. Il est associatif par ensemble à trois voies, physiquement adressé et *taggé*, de type recopie avec allocation de l'écriture (*write-back*, *write-allocate*). Il est organisé en lignes de 32 octets, et en secteurs de 64 octets. La granularité des échanges avec la mémoire ou le cache externe est configurable : soit une ligne de 32 octets, soit un secteur de 64 octets. Ce cache est totalement pipeliné, sa bande passante est de 16 octets par cycle en lecture ou écriture vers les caches primaires, les deux pouvant être alternées sans perte de cycle.

5.5.3 Cache externe

En plus du deuxième niveau de cache, le DEC 21164 met en œuvre la logique de contrôle nécessaire pour ajouter un troisième niveau de cache externe appelé *Board-level Backup Cache* (*Bcache*).
Ce cache est un cache optionnel, à correspondance directe, physiquement adressé, *write-back* avec une politique d'écriture de type *write-allocate*. La taille des secteurs est elle aussi configurable de 32 à 64 octets et doit être identique à la taille des secteurs choisie pour le second niveau de cache. La taille de ce cache est configurable et peut être de 1, 2, 4, 8, 16, 32 ou 64 Mo. Il peut y avoir jusqu'à deux lectures en cours par cycle (en fonction de l'implémentation). La vitesse et le temps d'accès de ce cache sont paramétrables.

5.5.4 Bande passante et temps de latence

	Temps de latence (cycle)	Bande passante (octets / cycle)
Caches de premier niveau	2	16
Cache secondaire	≥ 8	16
Cache externe	≥ 12	≤ 4

TAB. 5.1 - *Bande passante et temps de latence sur le DEC 21164*

5.6 Mécanismes d'optimisation de débit et latence

Au cours de ces dix dernières années, le temps de cycle du processeur a considérablement décru par rapport au temps d'accès à la mémoire. La hiérarchie mémoire mise en œuvre et l'utilisation de mémoires caches ne suffisent pas à compenser le coût d'un défaut de cache. Dans un futur proche selon [15], un défaut de cache sur une machine superscalaire n'exécutant que 2 instructions par cycle, pourrait engendrer la perte d'une centaine d'instructions et ce malgré les techniques classiques de hiérarchie mémoire mises en œuvre. Ces faits démontrent l'importance des recherches effectuées dans ce domaine, tant du point de vue logiciel par l'intermédiaire d'instructions de préchargements (voir 1.6.5) que par la mise en œuvre de techniques matérielles. Nous présentons ici les différents mécanismes mis en œuvre sur les processeurs étudiés pour masquer tout ou partie de la latence réelle des accès aux données.

5.6.1 Le DEC 21164

Deux unités sont sollicitées lors des accès à la mémoire:

- L'unité de traduction d'adresses mémoire (ou *Mbox*). Cette unité gère toutes les opérations de lecture/écriture à travers la table de traduction d'adresses qui convertit en adresses physiques les deux adresses virtuelles qu'elle est susceptible de recevoir de l'unité entière à chaque cycle. Cette unité assure également le respect de l'ordre des accès à la mémoire.

– Le contrôleur de cache et le contrôleur du bus regroupés sous la dénomination de *Cbox* traitent tous les accès émis par la Mbox et mettent en œuvre toutes les fonctions d'interface entre la mémoire et le reste du système (en particulier la gestion du protocole de cohérence de cache). La Cbox gère le cache secondaire (*Scache*) et le cache optionnel externe.

Afin de masquer la latence des accès (voir tableau 5.1), le DEC 21164 met en œuvre entre chacun des divers niveaux de la mémoire des mécanismes d'interface.

Le cache primaire d'instruction

Le DEC 21164 utilise une technique de préchargement matérielle des instructions de manière à garantir le remplissage des tampons d'instructions (voir figure 3.6).

Le cache d'instructions fournit les données à chaque fois qu'un nouveau groupe de quatre instructions est sollicité par le pipeline de l'Ibox (soit au maximum quatre instructions alignées par cycle). Lorsque les instructions demandées ne sont pas présentes dans le cache, le cache d'instructions passe en mode de remplissage et le tampon de pré-remplissage est alors testé. Si celui-ci contient les instructions demandées, elles sont envoyées directement vers le tampon d'instructions et le cache qui est mis à jour simultanément. Sinon, le cache d'instructions est à nouveau testé dans l'attente d'un *hit* suite à l'arrivée des instructions du cache secondaire (*Scache*), voire du Bcache (*Scache miss*), voire de la mémoire (*Scache miss, Bcache miss*). Ces instructions arrivent au rythme de quatre instructions par cycle, en réponse aux requêtes de chargement formulées par l'extracteur à la Mbox et sont écrites simultanément dans le cache et le tampon d'instructions. Quand l'instruction attendue est détectée dans le cache d'instructions, celui-ci redevient alors accessible et l'extracteur cesse d'émettre ses requêtes. Un certain nombre de groupes de quatre instructions vont continuer à arriver et seront stockés dans le tampon de pré-remplissage en attente du prochain défaut (au-moins trois blocs de 32 octets en amont du point courant de l'étage d'émission du pipeline).

Le cache primaire de données

Le DEC 21164 met en œuvre entre le premier niveau de cache et le reste du système deux mécanismes duaux respectivement sollicités lors des opérations de lecture et écriture.

– Le *Miss Address File* (ou MAF pour *Fichier d'adresses manquantes*), est un tampon de six entrées sollicité lors des défauts en lecture sur le cache primaire de données. Le MAF enregistre dans une de ses entrées, l'adresse de lecture et le registre cible, afin que le séquencement ne s'arrête pas. Si le deuxième niveau de cache est disponible, il est immédiatement sollicité. Sinon, le tampon agit comme une FIFO. Sa particularité est qu'il exploite la localité spatiale des données. Si deux instructions de lecture adressent la même ligne de cache, elles sont regroupées dans la même entrée du MAF et seront servies simultanément. Ce mécanisme permet ainsi de réduire le nombre des accès vers le cache secondaire, et tire partie de la bande passante offerte par l'architecture. Le MAF a une capacité de 21 chargements en attente (quatre par entrée plus une pour la dernière). Les données sont retournées du cache secondaire vers le cache primaire, ainsi que vers le fichier de registres ou encore directement vers l'unité de calcul qui attend son opérande.

– Le *write-buffer* (ou tampon d'écriture) est un tampon de six entrées également totalement associatif. Les requêtes d'écritures accédant à la même adresse d'une ligne de 32 octets peuvent

être mixées. Le tampon d'écriture est régi par la Mbox. Celle-ci envoie une requête à la Cbox pour lui demander d'effectuer l'écriture d'une transaction en attente (ces transactions peuvent arriver au rythme d'un mot de 8 octets par cycle).

Pour des raisons de cohérence, lors d'un défaut en lecture, le *write-buffer* est également testé afin de vérifier si la donnée demandée n'est pas en cours de mise à jour.

L'une des caractéristiques de fonctionnement de ces mécanismes est qu'ils conservent un certain temps les requêtes avant de les envoyer à la Cbox, pour optimiser l'efficacité du mécanisme de regroupement. Par exemple, dans le cas des écritures, les entrées en attente peuvent n'être envoyées qu'au bout de 64 cycles.

Remarque : Ces tampons représentent une structure relativement volumineuse puisque par exemple dans le cas du tampon d'écriture, chaque entrée doit contenir un champ d'adresse physique (bits d'adresses [39 : 05]), 32 octets de données, 8 bits de masques qui permettent de définir lequel des 8 mots de 4 octets contient une donnée valide, des bits de contrôle, etc... Soit un total de plus de 300 bits par entrée. Par ailleurs, ces tampons doivent également pouvoir supporter plusieurs accès à chaque cycle (par exemple : 2 *miss* plus un *miss* en cours de traitement).

Le cache secondaire

Le cache secondaire est de type *write-back*, ce qui impose de mettre à jour la mémoire lors du remplacement d'une ligne. Afin de ne pas contraindre le renouvellement d'une ligne au délai de mise à jour de la mémoire (qui est dans ce cas relativement long puisqu'il s'agit ici d'une transaction externe au composant), le cache secondaire met en œuvre un tampon de deux entrées qui lui permet de découpler la mise à jour du sous-système mémoire en conservant les lignes évincées du cache secondaire. Ce même mécanisme est aussi utilisé dans le cas de défaut en lecture afin de laisser libre d'accès le cache secondaire. Celui-ci est alors mis à jour à partir de ce tampon dès que les données sont fournies par le sous-système mémoire (cache externe ou mémoire principale).

Le cache externe

Deux types de configurations sont disponibles sur ce cache (lui aussi *write-back*) pour gérer les défauts en lecture. Une première configuration, sans tampon d'écriture (appelé ici *victim buffer*), impose au système de mettre à jour la mémoire principale avant de demander une nouvelle ligne. La deuxième configuration permet d'écrire la ligne victime du remplacement dans ce tampon, alors que la nouvelle ligne est simultanément demandée.

5.6.2 Le MIPS R8000

L'architecture découplée du MIPS R8000 nécessite divers mécanismes de prise en charge des accès pour masquer la latence de la mémoire et assurer la cohérence des données. Le R8000 est susceptible de générer deux adresses par cycle à destination du cache secondaire. Deux types d'accès peuvent avoir lieu vers la mémoire :

- accès en lecture : ils se font directement entre l'unité entière et le *streaming cache* pour les données entières, et entre l'unité flottante et le *streaming cache* en ce qui concerne les données flottantes.

– accès en écriture : ce type d'accès se fait exclusivement entre l'unité flottante et le *streaming cache*.

Dans les deux cas, les accès se font sous le contrôle de l'unité entière qui fournit les adresses ainsi que les signaux de commande (voir chapitre 9.2 relatif au *Tbus*).

Dans un premier temps, l'unité entière effectue une lecture des étiquettes du cache secondaire qui lui permettent de déterminer dans quel ensemble la donnée est présente (dans le cas des écritures, cette phase peut avoir lieu bien avant que la donnée ne soit disponible). Cette information est retournée vers l'unité entière sous la forme de deux bits qui constitueront les bits de poids fort d'adresse (dans les composants mémoire).

Dans le cas des instructions de lecture entière, le *streaming cache* est directement adressé par l'unité entière qui lit les données à partir d'un bus de 64 bits (le bus est dupliqué sur chacun des bancs de la mémoire).

Dans le cas des instructions de lecture flottante (issues de la *Floating-Point Queue*), les lectures se font sous le contrôle de l'unité entière qui génère les signaux à destination de l'unité flottante (*Enqueue Load*) et du *streaming cache*. Les données reçues sont écrites dans la *Floating-Point Data Queue* d'une capacité de 32 données (16 banc pair, 16 banc impair).

Dans le cas des instructions d'écriture, le fonctionnement est un peu plus complexe dans la mesure où seule l'unité flottante peut écrire dans le cache secondaire. L'ensemble des adresses générées par l'unité entière est mémorisé dans la *Store Address Queue* (d'une capacité de 32 adresses) localisée sur l'unité entière. Les données à écrire, associées à ces adresses, sont placées dans une file, la *Store Data Queue* (32 données de 64 bits pour chacun des bancs pair et impair), localisée sur l'unité flottante. Les données flottantes sont directement placées dans cette file à partir des registres, les données entières transitent par le Tbus. Une fois placée dans cette file, les données sont écrites dans le *streaming cache* sous le contrôle de l'unité entière qui adresse la mémoire et qui génère les signaux de contrôle (*Validation écriture, Dequeue Store*). La mise à jour des bits de validité est effectuée par l'unité entière.

La *Store Address Queue* et la *Floating Point Queue*, ainsi que les deux files de données qui leur sont associées, sont deux mécanismes qui permettent de gérer les conflits de ressources sans arrêter le séquencement des instructions. Ces mécanismes exploitent la structure pipelinée du *streaming cache*. Ils jouent un rôle capital sur cette organisation mémoire et permettent de masquer les cinq cycles de pénalité d'accès au cache. Ils rendent également possible le dépassement des écritures par les lectures en mémoire avec détection des aléas RAW.

Interface entre Streaming cache et mémoire principale

Les accès entre le *streaming cache* et la mémoire principale se font sous le contrôle du "contrôleur de cache" qui prend possession des bus de données. Cependant, l'ensemble des signaux de contrôle pour le *streaming cache* reste généré par l'unité entière. La mise à jour de la mémoire principale, à partir de données contenues dans le *streaming cache*, s'effectue par l'intermédiaire de l'unité flottante. La ligne de données est d'abord transférée vers l'unité flottante (comme une banale lecture), puis elle transite directement du bus de lecture vers le bus d'écriture à destination de la mémoire sous le contrôle du "contrôleur de cache" (génération des signaux *Floating-Point Output enable* et *Floating-Point Bypass*). La figure 5.5 schématise cette opération.

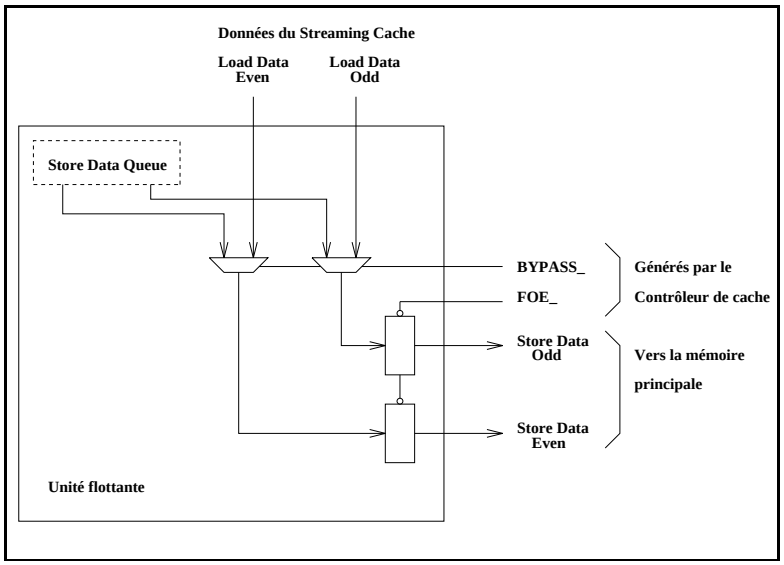


FIG. 5.5 - *Mise à jour de la mémoire principale sur le MIPS R8000*

5.6.3 Le Power2

L'unité entière et l'unité flottante offrent une structure similaire à celle du MIPS, dans la mesure où elles sont réparties sur des composants différents. Une synchronisation est nécessaire entre ces unités (voir chapitre 4.4).
L'unité entière gère les adresses à destination du cache de données. Les données sont lues dans le cache et placées sur l'un des quatre bus à destination des unités entières ou flottantes. Ce cache est non-bloquant, les données peuvent toujours être accédées même si un défaut de cache est en cours de traitement (deux défauts bloquent cependant tout accès).
Les lectures ont priorité sur les écritures.

Deux écritures (entières ou flottantes) peuvent être exécutées à chaque cycle. Chacune des unités dispose de tampons permettant de différer les accès vers le cache (*XPSQ* pour *Fixed-point Unit Pending Store Queue* d'une capacité de 2 entrées, et la *FPSQ* d'une capacité de 6 entrées[1]). Une ou deux de ces entrées peuvent être écrites dans le cache de données à chaque cycle selon la disponibilité des ports. Les écritures entières sont prioritaires sur les flottantes. En cas de débordement de la file de données flottantes, l'exécution des instructions d'écriture est stoppée. Les écritures se font toujours sous le contrôle de l'unité entière qui signale à l'unité flottante la disponilité du bus.
Les accès réels au cache ne respectent pas l'ordre de séquencement, mais les aléas RAW et WAW sont détectés à chaque accès au cache en testant les tampons *XPSQ* et *FPSQ*.

Quand une donnée est absente du cache, le renouvellement d'une ligne du cache à partir de la mémoire principale a alors lieu. Cette opération est réalisée à partir d'un troisième port sur le cache de données. Deux tampons permettent d'optimiser les performances lors des opérations de rechargement en différant la mise à jour de la mémoire.

1. La différence de capacité s'explique par le tampon d'instructions présent sur l'unité flottante et par le fait que les écritures entières sont prioritaires sur les écritures flottantes.

Par ailleurs, le Power2 offre la possibilité d'initialiser à zéro des lignes de données dans le cache. Ce mécanisme est bien plus rapide qu'une série d'écritures de données nulles et permet ainsi une initialisation rapide d'une zone de données.

5.7 Conclusion

Ces trois microprocesseurs mettent en œuvre des structures et des technologies de mémoire différentes pour satisfaire leur besoin en données. La diversité des méthodes illustre la complexité du problème posée par l'organisation efficace d'une hiérarchie mémoire.

Le Power2 implémente un cache primaire d'instructions et un large cache externe de données. L'utilisation d'une technique de multiport virtuel est une solution technologique brute par rapport aux solutions architecturales telle que l'entrelacement de la mémoire. Aussi, les limites sont claires quant à la fréquence d'horloge qui pourra être atteinte.

Le DEC 21164 met en œuvre un premier niveau de caches de petite taille permettant un cycle d'horloge très court, complété par un cache secondaire plus important. Cette organisation offre plusieurs avantages. Un premier niveau de cache de faible capacité permet de mettre en œuvre les deux ports d'accès en lecture par duplication du cache, ceci évite un accroissement de la surface trop important et privilégie un temps d'accès rapide. L'associativité du deuxième niveau de cache permettra de réduire les défauts de conflits sortant du composant.

Après un pipeline entier peu commun dans le monde des processeurs RISC, le MIPS R8000 met en œuvre une hiérarchie mémoire originale, séparant entiers et flottants. Le temps d'accès aux données flottantes est moins critique que le temps d'accès aux entiers car les tests influant sur le séquencement portent plus rarement sur des flottants, les instructions *CMOV* rendant ces tests encore plus rares. L'efficacité de cette organisation repose sur de nombreux mécanismes et accès pipelinés qui permettent de masquer la latence du cache externe.

Il est à noter que sur les trois processeurs, il a été mis en œuvre des mécanismes pour permettre de réduire les pénalités engendrées en cas de défauts sur le cache primaire. Ce type de mécanisme tend à se généraliser. La prochaine génération de processeurs aura une exécution d'instructions toujours plus dans le désordre.

Notons enfin que l'utilisation de mémoires statiques synchrones devrait se généraliser malgré les inconvénients inhérents à cette technologie récente (absence de standard, composant spécialisé pour le contrôle adapté à un type de processeur, etc...). Quoi qu'il en soit, les mémoires synchrones sont utilisées principalement dans les systèmes privilégiant la vitesse par rapport au coût et devraient équiper les prochaines génération de microprocesseurs (MIPS R10000, UltraSparc, PowerPC 620,...). En matière de standard et de coût, les fabricants s'accordent pour dire que la prochaine génération (1995-1996) sera beaucoup plus homogène, et les volumes de production augmentant, les prix baisseront.

5.8 Récapitulatif

TAB. 5.2 - *Hiérarchie mémoire : récapitulatif*

	MIPS R8000	Power2	DEC 21164
Cache d'instructions			
Capacité	16 ko	32 Ko	8 Ko
Associativité	direct-mapped	2 voies	direct-mapped
Algorithme de remplacement	-	LRU	-
Adressage des données	virtuel	physique	virtuel
Adressage des étiquettes	virtuel	physique	virtuel
Taille d'une ligne	16 octets	128 octets	32 octets
Taille d'un secteur	32 octets		32
On-chip	oui	oui	oui
Cache de données			
Capacité	16 ko (entier)	256 Ko	8 Ko
Associativité	direct-mapped	4 voies	direct-mapped
Algorithme de remplacement	-	LRU	-
Politique d'écriture	write-through	write-back	write-through no write-allocate
Adressage des données	virtuel	physique	physique
Adressage des étiquettes	virtuel (mais testé sur contenu physique)	physique	physique
Taille d'une ligne	8 octets	256/128 octets	32 octets
Taille d'un secteur	32 octets	256/128 octets	
architecture	?	double-pump techno 0.6 um	duplication du banc mémoire
Ports	2 : 2 lectures ou 1 lecture et 1 écriture	2	2 : 2 lectures ou 1 écriture
On-chip	oui	non	oui

TAB. 5.3 - *Hiérarchie mémoire : récapitulatif (suite)*

	MIPS R8000	Power2	DEC 21164
Cache secondaire			
Capacité	1 - 16 Mo	-	96 Ko
Associativité	4 voies	-	3-way
Politique d'écriture	write-back	-	write-back
	write-allocate	-	write-allocate
Adressage des données	physique	-	physique
Adressage des étiquettes	physique	-	physique
Taille d'une ligne	64, 128, 256	-	32 ou 64 octets
Taille d'un secteur	128, 256, 512	-	64
architecture	entrelacée (2 bancs)	-	
Ports	2	-	1
Nature	instructions et données	-	instructions et données
On-chip	non		oui

Chapitre 6

Support des systèmes d'exploitation

6.1 Introduction

Les trois microprocesseurs étudiés sont destinés à être utilisés dans des stations de travail ou des systèmes multiprocesseurs. Ils fournissent donc des mécanismes de gestion de la mémoire virtuelle et un support permettant de mettre en œuvre un système d'exploitation.

Les mécanismes matériels mis en œuvre sur chacune des architectures pour implémenter la mémoire virtuelle sont spécifiques à une hiérarchie mémoire donnée. Ce chapitre est donc étroitement lié au chapitre précédent.

La taille de la mémoire virtuelle est fixée par le nombre de bits d'adresse virtuelle. Dans, le cas d'un espace d'adressage virtuel linéaire (cas du MIPS R8000 et du DEC 21164), l'adresse virtuelle est identique à l'adresse calculée par l'instruction (à laquelle s'ajoute un identificateur de processus). Sur le Power2, l'espace d'adressage est segmenté ; l'adresse virtuelle est constituée d'un numéro de segment et d'un déplacement dans le segment.

6.2 Généralités

Le système d'exploitation doit gérer le système mémoire. Plusieurs processus peuvent être actifs et résidants en mémoire simultanément. Afin de permettre à ces processus de cohabiter et de partager des données, un mécanisme de mémoire virtuelle est mis en œuvre.

La mémoire virtuelle peut être implémentée de diverses manières. On distingue généralement deux types d'espace d'adressage : l'adressage linéaire et l'adressage segmenté. Dans le cas de l'adressage linéaire, une adresse de taille fixe calculée par le processeur est utilisée comme adresse virtuelle. Elle est divisée en un numéro de page et un déplacement dans cette page. Dans le cas d'un adressage segmenté, l'adresse virtuelle s'obtient par la concaténation d'un numéro de segment et le déplacement dans le segment (voir Pentium [3]). Le déplacement dans le segment est l'adresse calculée par le programme.

Certaines machines (en particulier les architectures Power et PowerPC) utilisent une approche hybride, appelée *segmentation paginée* dans laquelle chaque segment est constituée d'un certain nombre de pages. L'adresse virtuelle est composée des 28 bits de poids faible de l'adresse calculée plus un numéro de segment. Cette variante permet de s'affranchir de la contrainte d'avoir de grands

espaces d'adressage contigus, facilitant ainsi le remplacement des blocs de la mémoire principale, tout en alliant les avantages de la segmentation : protection séparée des instructions et des données, partage facilité des instructions et des données, manipulation plus simple des structures de données dont la taille varie dynamiquement.

6.3 L'espace virtuel

Sur le MIPS R8000 et le DEC 21164, une mémoire paginée est mise en œuvre. Le Power2 implémente une mémoire segmentée paginée.

6.3.1 Taille des pages

Comme son prédécesseur le R4000, le MIPS R8000 manipule des pages dont les tailles peuvent varier de 4 Ko à 16 Mo (4, 8, 16, 64 Ko, 1 Mo et 4 Mo). Chaque processus peut ainsi avoir une taille de pages spécifique (pouvant être différente entre les données et les instructions) ce qui permet une meilleure gestion de la mémoire en fonction du type de l'application. La différence considérable entre la taille minimale et maximale d'une page (rapport de 4096) traduit les efforts effectués pour optimiser le chargement des données : une application numérique écrite en Fortran où tous les tableaux sont alloués statiquement pourra utiliser une grande taille de page, tandis qu'un processus système utilisera une taille de page plus petite.

La norme DEC Alpha définit des pages d'une taille de 8 Ko. Afin de compenser la taille constante des pages et de minimiser les défauts de traduction d'adresse, le matériel permet de grouper et d'aligner les pages par 8, 64 ou 512.

Le Power2 supporte une seule taille de pages : 4 Ko.

6.3.2 Espace virtuel

MIPS R8000

L'adresse virtuelle est calculée sur 64 bits, mais seuls les 48 bits de poids faible sont utilisés pour adresser une région de l'espace mémoire. L'adressage physique est de 40 bits.
Les 48 bits d'adresse virtuelle sont étendus avec un champ de huit bits d'identification de processus (ASID pour *Adress Space Identifier*, 256 processus distincts possibles). Deux valeurs d'ASID sont utilisées par processus, une pour les instructions (IASID) et l'autre pour les données. Ceci permet d'utiliser des tailles de page différentes pour les instructions et les données.

DEC 21164

L'adresse virtuelle est calculée sur 64 bits, mais seuls les 43 bits de poids faible sont utilisés, les bits de poids fort étant juste validés. Cette configuration est identique à celle du 21064 ; DEC prévoit de passer successivement à 47, 51 puis 55 bits d'adresses virtuelles pour les générations successives. L'espace d'adressage physique a été augmenté à 40 bits (au lieu de 34 sur la version précédente). Comme sur le MIPS, l'adresse virtuelle est complétée par un identificateur de processus (*ASN* pour *Address Space Number*). Une seule valeur d'ASN est utilisée par processus. Ce champ est codé sur sept bits, permettant ainsi à 128 processus de cohabiter simultanément en mémoire.

Power2

L'espace virtuel du Power2 est identique à celui du Power1 et du PowerPC 601, c'est pourquoi nous reprendrons pour l'essentiel ce qui a déjà été dit dans [1]. L'espace virtuel du Power2 n'est pas réellement segmenté au sens habituel du terme. L'approche mise en œuvre est une extension de l'espace virtuel paginé qui reste linéaire. Chaque processus ne voit qu'une mémoire virtuelle de 4 Go, découpée en 16 segments de 256 Mo chacun. Cependant la véritable adresse virtuelle est obtenue par indirection. Les quatre bits de poids fort de l'adresse effective sur 32 bits sont utilisés comme numéro de registre de segment. Ce registre de segment contient 24 bits. L'adresse est alors étendue à (32 - 4) + 24 = 52 bits, ce qui donne un espace virtuel de 4 Peta-octets. Le contenu des registres de segments est géré par le système d'exploitation, ainsi un processus ne peut accéder qu'aux 16 segments qui lui sont alloués sur les 2^{24} segments disponibles. Cette notion d'espace virtuel étendu emprunté à Multics, permet notamment d'intégrer le système de gestion des fichiers dans l'espace virtuel. L'adressage physique du Power2 est de 32 bits.

L'approche d'IBM présente un compromis intéressant pour le partage de données ou de code. Tout en conservant l'espace virtuel linéaire paginé, deux processus partagent du code ou des données en partageant un même segment. Ce code ou ces données ne sont pas forcément à la même adresse virtuelle vue du processus puisque les quatre bits de poids fort de l'adresse de 32 bits peuvent être différents. On peut toutefois reprocher à l'architecture Power le petit nombre de segments (16) dont un processus dispose.

6.4 Traduction d'adresse

6.4.1 Mécanismes de traduction d'adresses

Pour les trois processeurs étudiés, la table des pages est *mappée* en mémoire virtuelle. La traduction s'effectue sur les bits de poids forts de l'adresse virtuelle.

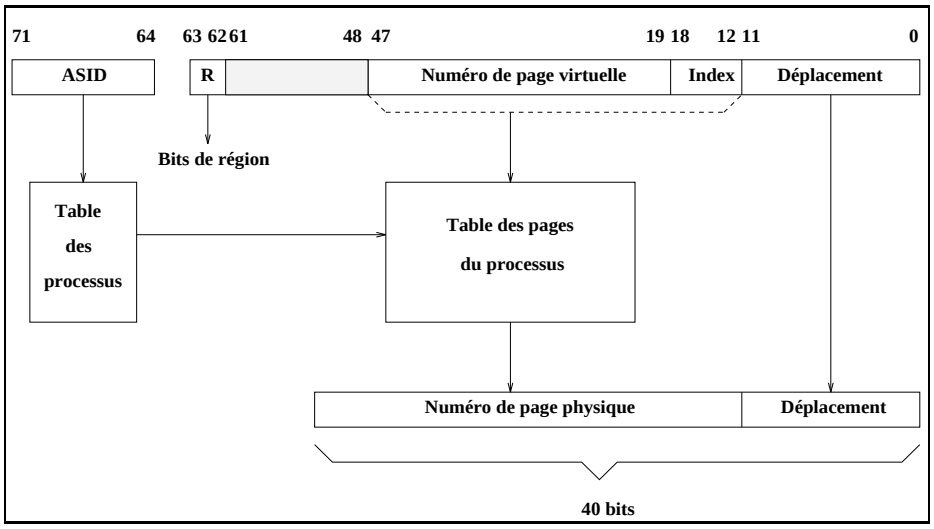


FIG. 6.1 - *Mécanisme de traduction d'adresses du MIPS R8000 (exemple d'une page de 4 Ko)*

Sur le MIPS R8000 et le DEC 21164 ces bits de poids fort sont utilisés pour indexer une table des pages contenant l'adresse physique correspondante. Le déplacement dans la page reste inchangé. La table des pages a été préalablement sélectionnée par l'identificateur de processus (l'ASID dans le cas du R8000 ou l'ASN dans le cas du DEC 21164) parmi l'ensemble des tables présentes en mémoire. Le mécanisme de traduction d'adresses utilisé pour le MIPS R8000 est présenté figure 6.1.

Le DEC 21164, comme le MIPS R8000, offre selon le mode d'exploitation (voir chapitre 6.5), des régions où l'adresse virtuelle et l'adresse physique sont confondues.

Dans le cas du Power2, une adresse effective (adresse issue du programme) est d'abord traduite en une adresse virtuelle puis en une adresse physique. Le mécanisme de traduction d'adresse du Power2 est présenté à la figure 6.2 et la table des pages est détaillé à la figure 6.3.

Les 32 bits d'adresses effectives sont convertis en une adresse virtuelle de 52 bits comme suit :

1. Utilisation des 4 bits de poids forts de l'adresse effective pour sélectionner l'un des 16 registres de segments.

2. Concaténation des 24 bits d'identification du segment fournis par le registre de segment accédé, avec les 28 bits restants de l'adresse effective.

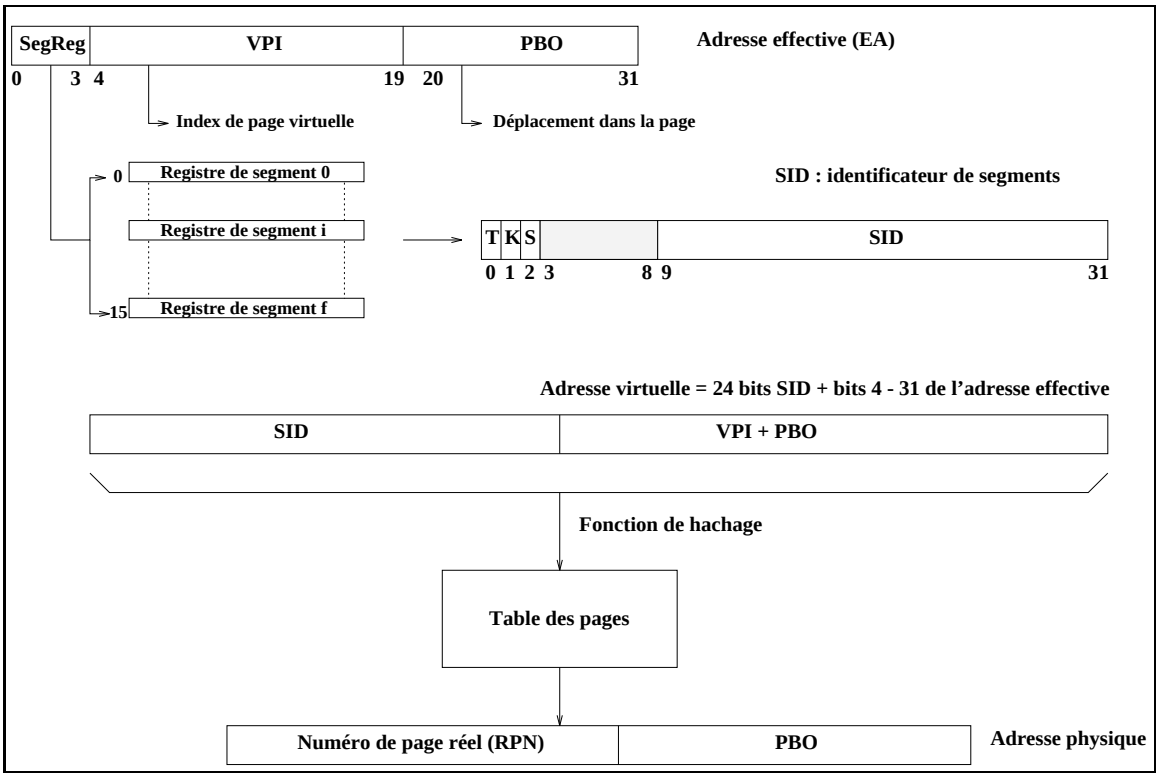


FIG. 6.2 - *Mécanisme de traduction d'adresses du POWER2*

On arrive alors à une adresse virtuelle sur 52 bits. Celle-ci est alors convertie en une adresse physique de 32 bits en utilisant une table de pages indexée par l'intermédiaire de deux fonctions de hachage. Cette table de pages contient un maximum de 2^{19} entrées. On applique une première

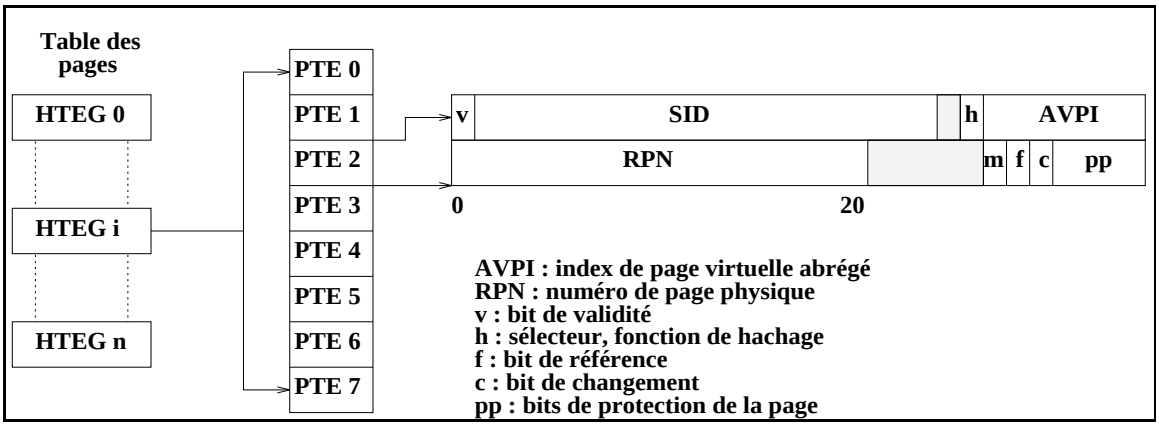


FIG. 6.3 - *Détails de la table des pages et de ses entrées*

fonction de hachage sur l'adresse virtuelle. Ceci produit un pointeur sur la première des deux entrées susceptibles de correspondre à cette adresse virtuelle. Si la première entrée n'est pas la bonne, la fonction de hachage est à nouveau appliquée et une seconde entrée est testée.
Chacune des entrées contient huit descripteurs de page qui contiennent chacun un identificateur de segment, un numéro de page virtuel abrégé que l'on teste avec le numéro de page virtuel généré. Si le test est positif, le descripteur fournit un numéro de page physique, les bits de protection relatifs à la page et des bits de référence et de changement (cette organisation est similaire à celle étudiée dans [3]).

6.4.2 Cache de traduction d'adresses

Les mécanismes de traduction d'adresse décrits précédemment sont très lourds. Pour pouvoir atteindre des performances correctes, un cache de traduction d'adresse, généralement appelé le TLB (*Translation Lookaside Buffer*), est utilisé. Suivant le même principe que les caches étudiés précédemment, le TLB est consulté à chaque fois que le processeur produit une adresse virtuelle et permet d'accélérer nettement le mécanisme.

Cache de traduction d'adresses du MIPS R8000

Le fonctionnement des caches de traduction d'adresses est étroitement lié au fonctionnement de la hiérarchie mémoire. Le cache de traduction du MIPS R8000 est implanté sur l'unité entière. Puisque le cache d'instructions est virtuellement indexé, la traduction de l'adresse est nécessaire dans ce cas uniquement lors d'un défaut de cache. Aussi, l'unité entière n'a qu'un unique TLB pour la traduction des adresses d'instructions et de données.

Le cache de données du MIPS R8000 est indexé virtuellement mais le test de validité se fait sur l'adresse physique. Le cache de traduction doit supporter deux accès par cycle, et offre à cet effet deux ports. Il est associatif par ensemble à trois voies et compte au total 384 entrées (soit 128 ensembles). Le nombre relativement important d'entrées de ce cache résulte d'un compromis entre un temps d'accès d'un cycle et la nécessité de minimiser les défauts de page. L'indexage d'un TLB associatif par ensembles où peuvent être rangé des descripteurs de page de différentes tailles est un

problème délicat (choix de l'index). Il est ici résolu simplement par l'utilisation pour chaque ASID d'une seule taille de page ; cette taille fait partie du contexte du processus.

L'algorithme de remplacement sur le TLB est basé sur un algorithme de remplacement aléatoire. On notera le changement par rapport au R4000 où le remplacement est géré par logiciel. Pour éviter que certaines pages fréquemment accédées ne soient vidées, le système d'exploitation a la possibilité de verrouiller certaines entrées (jusqu'à 4 entrées du premier banc du TLB peuvent ainsi être préservées).

Cache de traduction d'adresses du DEC 21164

Le DEC 21164 a deux caches de traductions d'adresses séparés :

- un cache de traduction de 48 entrées pour les instructions (ITB) ;
- un cache de traduction de 64 entrées pour les données (DTB).

Ces deux TLB sont totalement associatifs et mettent en œuvre un algorithme de remplacement de type *not last used* (choix aléatoire d'une ligne à remplacer parmi l'ensemble, excepté la plus récemment référencée). Chacune des entrées supporte toutes les combinaisons de granularité quant à la taille des pages (de 8 Ko à 4 Mo). La taille des TLBs présente une notable amélioration par rapport au DEC 21064 (TLB instructions et données respectivement de 12 et 32 entrées).

La gestion du remplacement est entièrement logicielle. Le système d'exploitation utilise le PAL-code pour remplir et maintenir la cohérence au sein de ces caches.

Cache de traduction d'adresses du Power2

Deux TLB distincts pour les instructions et les données sont mis en œuvre sur le Power2. Le TLB instructions est associatif par ensemble de deux voies et a 128 entrées. Le TLB données est également associatif par ensemble de deux voies et a 512 entrées.

Conséquence des mécanismes mis en œuvre par IBM (tables de pages et fonctions de hachage), la mise à jour des TLB sur les défauts est supportée par matériel (algorithme de remplacement de type *least recently used*).

6.5 Protection

Les processeurs disposent de plusieurs modes de fonctionnement. Ces modes protégent en écriture ou en lecture certaines zones mémoires et permettent à différentes tâches de s'exécuter sans interférences. Sur les processeurs MIPS R8000 et DEC 21164, la protection est réalisée à chaque accès à la mémoire, au cours de la traduction d'adresse. Cette protection se fait au niveau de la page. Sur le Power2, la protection est également effectuée au niveau de la segmentation.

6.5.1 Protection sur le DEC 21164

Le DEC 21164 possède les quatre modes d'exploitation suivants :

- le mode noyau ;
- le mode superviseur ;

– le mode exécution ;

– le mode utilisateur.

En mode utilisateur et exécution, l'usage de certaines instructions est interdit. En mode noyau et superviseur, tout est permis. Les modes noyau et superviseur sont généralement destinés à gérer le séquencement des processus, la gestion des interruptions, etc... Le système d'exploitation s'exécute dans un de ces modes et contrôle l'ensemble du fonctionnement de la machine. Le *mapping* de la mémoire est différent selon le mode noyau ou superviseur.

La protection des pages varie en fonction du mode courant. Par exemple les droits d'accès sur une page peuvent être *lecture* en mode utilisateur et *lecture - écriture* en mode superviseur ou noyau.

Les appels au mode noyau sont implantés via l'appel à une librairie, le PALcode (*Privileged Architecture Library*), servant d'intermédiaire entre le système d'exploitation et le matériel. Chaque version de Palcode est spécifique à un système d'exploitation donné. Les concepteurs de DEC justifient cette approche par le fait que certaines opérations seraient trop complexes à mettre en œuvre matériellement et ne sont pas prises en comptes par un système d'exploitation courant. Ces opérations concernent :

– l'appel explicite à des routines systèmes ;

– La gestion du TLB ;

– les routines de traitement des exceptions ;

– des primitives de synchronisation.

Ces fonctions s'exécutent dans un environnement particulier dans lequel la traduction des adresses des instructions en mémoire est invalidée (puisque ces fonctions assurent la gestion du TLB. Les données restent cependant valides) et les interruptions sont masquées. Ceci permet d'exécuter des séquences de code de manière atomique. L'un des aspects important de cette librairie est qu'elle utilise le jeu d'instructions du processeur pour réaliser l'ensemble de ses opérations. Seules quelques rares instructions pour la gestion des registres de contrôle internes ont été rajoutées. Cependant toutes tentatives d'exécution de ces instructions en dehors du mode PAL se soldent par une exception d'instruction illégale. Les fonctions PALcode sont appelées à travers l'instruction *CALL_PALL*. Un mécanisme masque alors les interruptions et autorise le mode privilégié pour les routines du PALcode. La démarche inverse est ensuite réalisée pour retrouver l'environnement initial. Des registres de contrôle et de sauvegarde de résultats temporaires sont dédiés aux opérations effectuées dans ce mode.

6.5.2 Protection sur le MIPS R8000

Le MIPS R8000 ne supporte que deux modes d'exécution : le mode noyau et le mode utilisateur. Le mode *superviseur* de son prédécesseur, le MIPS R4000, n'a pas été implémenté sur ce processeur. Un processus lancé par le système d'exploitation en mode utilisateur, ne pourra générer des adresses que dans la zone mémoire allouée à ce mode, garantissant ainsi une protection du système.

Le mode est spécifié par les deux bits de région de l'adresse virtuelle ([63 : 62]). Le mode noyau se décompose en trois modes différents auxquels correspond un espace mémoire spécifique et

l'utilisation ou non de la table de traduction d'adresses et des indentificateurs de processus pour la traduction adresse virtuelle - adresse physique. Ces trois modes sont :

- *le mode noyau virtuel 0* est un mode privé. Ce mode permet l'utilisation de la TLB et des ASID.

- *le mode noyau virtuel 1* défini comme un noyau global. Par opposition au mode précédent, les adresses virtuelles ne sont pas étendues avec les huit bits d'ASID.

- *le mode noyau physique*. Dans ce mode, les adresses virtuelles et physiques sont identiques et aucune traduction d'adresse n'est effectuée.

Le tableau 6.1 récapitule les divers modes du MIPS R8000 et l'usage de la table de traduction d'adresses et des identificateurs de processus.

Espace d'adresses virtuelles	Région [63 : 62]	Utilisation de la table de traduction	Utilisation des ASID
Mode noyau virtuel 1	11	oui	non
Mode noyau physique	10	non	non
Mode noyau virtuel 0	01	oui	oui
Mode utilisateur	00	oui	oui

TAB. 6.1 - *Modes d'utilisation du MIPS R8000*

6.5.3 Protection sur le Power2

Dans le cas de l'architecture Power, la séparation superviseur - utilisateur se fait au niveau des segments. Chaque processus utilisateur dispose de 16 segments dont deux sont réservés au noyau. Les droits d'accès sont contrôlés à deux niveaux : segments et pages. Le chargement des registres de segments, donc le contrôle de l'accès à l'espace virtuel système étendu, ne se fait qu'en mode superviseur par des instructions privilégiées. Les entrées - sorties bénéficient d'un traitement particulier avec des registres de segments spécifiques.

Une deuxième vérification est effectuée au niveau de la page adressée. Le type d'accès effectué (écriture ou lecture) y est comparé avec les droits que possède le processus sur la page accédée.

6.6 Conclusion

L'utilisation de la segmentation sur le Power2, offre au programmeur une grande souplesse d'utilisation pour le partage des données entre processus, tout en assurant une sécurité au niveau des accès à la mémoire. La manipulation des segments est à la charge du système d'exploitation, en particulier le contenu des registres de segments, modifié à chaque changement de contexte.

Les processeurs MIPS R8000 et DEC 21164 ont privilégié une approche différente en choisissant une mémoire seulement paginée. La protection se fait donc à chaque accès au niveau des pages. Les processus sont identifiés par le biais d'un champ supplémentaire codé sur huit bits dans le cas du MIPS et sept bits dans le cas du DEC. MIPS offre deux valeurs différentes d'identificateurs par processus, et permet ainsi l'utilisation de pages de différentes tailles pour les instructions et les données. À la différence de la segmentation, la pagination est entièrement gérée par le système d'exploitation et est donc transparente au programmeur.

Les tailles des zones que peuvent *mapper* les TLBs des différents processeurs, sont devenues extrèmement importantes et sembles adaptées aux besoins de plus en plus grands des applications d'aujourd'hui.

Chapitre 7

Support multiprocesseur

7.1 Introduction

Le degré de performance atteint par les microprocesseurs étudiés les rendent attrayant pour les systèmes multiprocesseurs. Le DEC 21164 et le MIPS R8000 sont conçus pour être utilisés dans des systèmes multiprocesseurs symétriques.

Les microprocesseurs MIPS R8000 et DEC 21164 mettent en œuvre des mécanismes matériels qui facilitent l'implémentation d'une mémoire partagée. Les systèmes parallèles annoncés par DEC devraient pouvoir accueillir jusqu'à six processeurs DEC 21164. Les systèmes multiprocesseurs à base de MIPS R8000, commercialisés par SGI, accueillent jusqu'à 18 processeurs et peuvent atteindre des performances crêtes de 5.4 Gflops (système *Power Challenge XL*). Un système à moindre coût intégrera six processeurs.

Dans un système multiprocesseur à mémoire partagée, il est nécessaire de disposer de protocoles de cohérence de caches qui assurent l'exclusion mutuelle en écriture et la mise à jour des différents caches, afin que chaque processeur puisse lire des données cohérentes.

Les deux microprocesseurs disposent de supports matériels pour l'implémentation de protocoles de cohérence de caches ainsi que de primitives de synchronisation qui permettent l'exécution de façon atomique d'accès à la mémoire.

Nous ne mentionnerons pas dans ce chapitre le Power2. Le Power2 apparaît dans des systèmes multiprocesseurs à mémoire distribuée, mais aucun support matériel n'est mis en œuvre sur le processeur lui-même pour faciliter ce type d'architecture.

7.2 Cohérence de cache

La figure 7.1 illustre une implantation possible d'un multiprocesseur à mémoire partagée autour d'un microprocesseur et d'un cache externe. Chaque contrôleur d'interface accède aux étiquettes dans le but de maintenir la cohérence. La duplication de ces étiquettes n'est pas obligatoire, mais elle est généralement conseillée afin de ne pas encombrer les ports réservés aux transactions du processeur par des cycles de cohérence de caches.

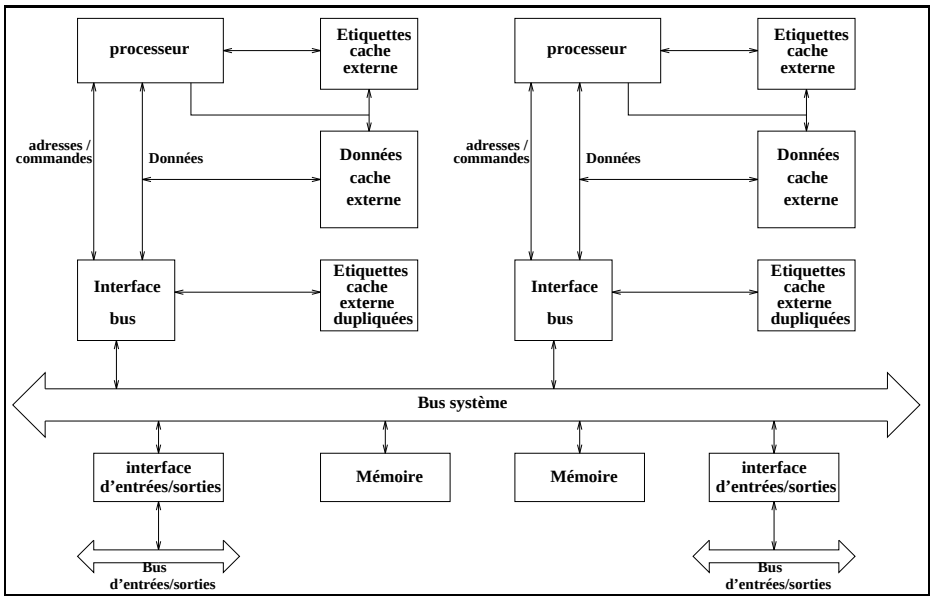


FIG. 7.1 - *Configuration typique d'un système multiprocesseur avec un cache externe*

Dans de telles configurations, la cohérence des caches est maintenue grâce à l'association de deux mécanismes :

- l'espionnage du bus (*Bus Snooping*). Chaque contrôleur de cache compare l'adresse du transfert en cours sur le bus commun avec le contenu de son, ou ses cache(s). Si l'un des caches du processeur possède une copie de la donnée transférée, une action est nécessaire (mise à jour de la donnée, changement de statut de cette donnée). Ce type de mécanisme est très répandu dans de petits systèmes parallèles (pas plus de 20 processeurs) où la mise en œuvre d'un bus partagé favorise les mécanismes de diffusion pour maintenir la cohérence.

- Le recommencement des transferts qui ont été avortés (*Bus Retry*). Il se peut suivant le protocole de cohérence utilisé, qu'un transfert soit avorté par un autre processeur qui possède une copie plus récente que la donnée transférée. Une fois la mise à jour de la mémoire effectuée, le processeur qui avait initié le transfert essaye à nouveau d'accéder à la mémoire. Ce mécanisme possède l'avantage d'être particulièrement simple à mettre en œuvre mais reste assez coûteux en temps.

De plus pour maintenir simplement la cohérence de toute la hiérarchie des caches, la propriété d'inclusion est forcée par matériel. L'ensemble des données représentées dans un niveau de cache est un sous-ensemble de l'ensemble des données présentes au niveau supérieur.

7.2.1 Cohérence de cache pour le MIPS R8000

L'un des éléments importants d'un système bâti autour du MIPS R8000, est *le contrôleur de cache* (voir figure 2.2, page 36). Ce composant n'est pas proposé par MIPS, mais ses fonctionnalités sont spécifiées. Il est chargé de l'interface entre le MIPS R8000, la mémoire principale et le reste du système. L'intervention de ce *contrôleur de cache* est requise dans les cas suivants :

- interface avec le reste du système (chargement de données à partir de la mémoire principale ou mise à jour de la mémoire à partir du cache secondaire, transaction de cohérence avec les autres processeurs) ;
- gestion de la cohérence des deux niveaux de caches (cache secondaire externe et cache interne) ;
- gestion de la cohérence du cache secondaire externe avec les autres caches du système ;

La duplication des RAM d'étiquettes (déjà présentes dans le *streaming cache*), est nécessaire pour permettre au système d'atteindre le maximum de ses performances.

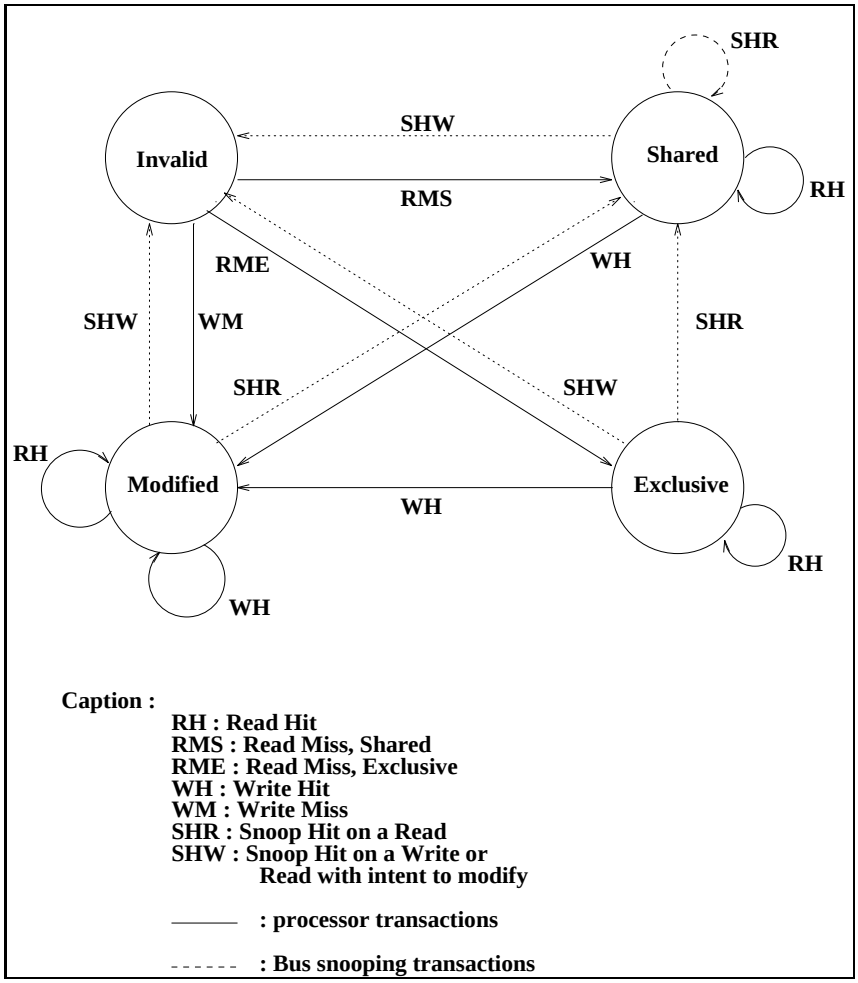


FIG. 7.2 - *Protocole MESI à invalidation sur écriture*

Le protocole de cohérence de caches employé par le MIPS R8000 est le protocole à invalidation en écriture nommé *MESI* d'après les quatre états possibles d'une ligne de cache : *Modified, Exclusive, Shared, Invalid.* Ce protocole est actuellement le plus utilisé dans des systèmes incluant des caches à recopie retardée en mémoire. Lorqu'une donnée est modifiée par un processeur, tous les caches possédant une copie de cette donnée invalident la ligne qui la contient. La figure 7.2 décrit ce protocole.

7.2.2 Cohérence de cache pour le DEC 21164

Le DEC 21164 gère la cohérence des niveaux de caches internes, mais la cohérence avec la mémoire principale du système est fournie par la logique d'interface.

- L'unité d'interface du bus et de contrôle du cache (la *Cbox*) est chargée de maintenir la propriété d'inclusion entre le cache de données et le cache secondaire.
- Si le cache externe optionnel est présent, alors la Cbox fournit aussi la logique nécessaire pour maintenir le cache secondaire comme un sous-ensemble de ce cache externe.
- Par contre l'interface avec le reste du système doit fournir au 21164 les mécanismes qui permettront de maintenir le cache externe (si il est présent) ou le cache secondaire cohérent avec la mémoire principale et les autres caches dans le système.
- Le cache d'instructions n'est pas gardé cohérent avec le reste du système. La cohérence au sein du cache d'instructions est un problème qui doit être résolu de manière logicielle. Pour mettre à jour le flot d'instructions partagé par l'ensemble des processeurs, un processeur écrit dans le flot d'instructions, effectue une instruction PALcode *IMB (I-stream Memory Barrier)* et envoie un signal d'interruption vers les autres processeurs. À la réception de ce signal, les processeurs effectuent eux aussi une instruction *IMB* (cette instruction garantit que le cache d'instructions est cohérent avec le contenu de la mémoire principale) pour charger le nouveau flot d'instructions dans le cache.

La figure 7.3 illustre la cohérence au sein de la hiérarchie mémoire à travers le mécanisme d'inclusion de ses divers niveaux.

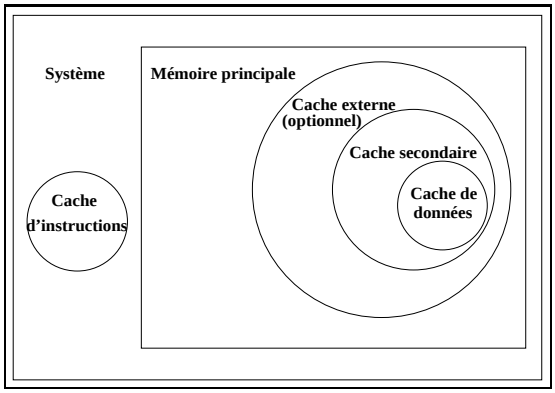


FIG. 7.3 - *Cohérence au sein de la hiérarchie mémoire*

Le DEC 21164 inclut des mécanismes matériels qui permettent de supporter divers protocoles de cohérence de cache. Ces protocoles imposent des contraintes plus ou moins fortes sur le système et se répartissent communément en deux classes :

- protocole de cohérence de cache d'invalidation sur écriture de type MESI (voir MIPS R8000) ;

- protocole de cohérence de cache par éviction du bloc (*Flush Protocol*) essentiellement destiné aux configurations à moindre coût.

Ces deux protocoles peuvent être plus ou moins performants selon la configuration matérielle mise en œuvre par le concepteur du système. Cette configuration doit comprendre un contrôleur de cache, une copie des RAM d'étiquettes internes, et une interface avec le reste du système. L'ensemble de ces mécanismes soulage le processeur de la surveillance des transactions qui se déroulent sur le bus commun.

Le protocole de cohérence de cache par éviction du bloc (ou *Flush Protocol*) ne permet pas le partage des données. Il comprend par conséquent moins d'états que le protocole d'invalidation sur écriture similaire à celui illustré pour le MIPS R8000 (figure 7.2). Ce protocole ne permet pas d'avoir plus d'un propriétaire d'une même ligne. La figure 7.4 reprend les divers états possibles d'une ligne de cache de ce protocole.

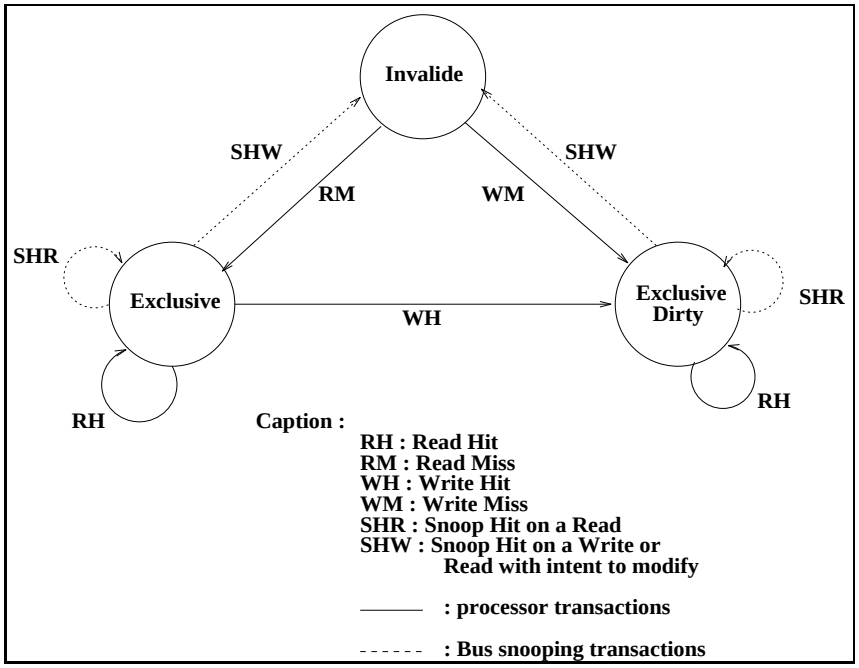


FIG. 7.4 - *Protocole de cohérence par éviction du bloc*

7.3 Support de synchronisation

Dans la plupart des systèmes monoprocesseurs actuels, l'ordre des accès à la mémoire à l'exécution est effectivement le même que celui écrit par le programmeur (ou, tout du moins généré par le compilateur). Dans un système multiprocesseur, ceci ne peut être respecté que grâce à des primitives de synchronisation. En effet, les exécutions ayant lieu en parallèle, il est impossible de connaître de manière déterministe l'ordre des accès à un emplacement donné de la mémoire partagée sans instaurer des protocoles. De plus, le partage d'une mémoire induit naturellement des problèmes d'exclusion mutuelle. Pour les résoudre, certains supports matériels tel que, par exemple, la présence d'opérations atomiques dans le jeu d'instructions des processeurs sont utiles.

7.3.1 Accès à la mémoire partagée

Pour réaliser des accès atomiques à la mémoire partagée, c'est à dire qui ne doivent pas être interrompus par des interruptions ou d'autres écritures au même emplacement, les deux microprocesseurs proposent des instructions spécifiques.

- *load-locked / store-conditional* pour le DEC 21164;
- *load-linked / store-conditional* pour le MIPS R8000.

L'exemple ci-dessous, montre une séquence permettant un accès atomique à une donnée alignée de la mémoire partagée dans un système bâti autour d'un processeur DEC.

Exemple 10

```
Ex_ :
        LDQ-L R1, x;            Chargement d'une donnée partagée dans R1.
        "modification de R1";   Traitement sur cette donnée.
        STQ-C R1, x;            Tentative de mise à jour de la mémoire. Cette
                                instruction renvoie "1" dans R1 si aucune in-
                                terruption n'a eu lieu entre LDL-L et STQ-C et
                                si aucun processeur n'a écrit à l'adresse x, sinon
                                elle renvoie "0".
        BEQ R1, défaut_écriture; Si R1=1, la mémoire a été mise à jour;
        ...                     Si R1=0 on recommence la séquence.
défaut_écriture :
        "Code d'attente pour éviter un
        rebouclage trop rapide"
        BR Ex_;
```

Il est à noter que la séquence d'instructions entre *LDQ-L* et *STQ-C* (appelée sur notre exemple *modification de R1*), ne doit pas être trop longue sans quoi des interruptions périodiques du système viendraient entraver le bon fonctionnement de cette séquence. Aussi, dans le cas d'une séquence plus longue ou d'une mise à jour d'une structure de donnée partagée, le programmeur peut définir une section critique par le biais d'une variable de verrouillage. L'exemple 11 illustre cette situation où on retrouve la séquence de base d'exclusion mutuelle dans un système multiprocesseur à mémoire partagée :

1. Acquisition de la variable d'exclusion mutuelle ;

2. Section critique (mise à jour de données partagées, etc...);

3. Relachement de la variable d'exclusion mutuelle.

Exemple 11

```
stq_c_bcle :
boucle :
        LDQ_L R1, lock_variable ;
        BLBS  R1, déjà_validé ;        Branch if register Low Bit is Set.
        OR    R1, #1, R2 ;
        STQ_C R2, lock_variable ;      Validation du bit de verrouillage
        BEQ R2, échec_stq ;
        MB ;                           Garantit la séquentialité des accès sur la mé-
                                       moire avant d'entrer dans la section critique.
        < section critique >
        MB ;                           Garantit la mise à jour de la mémoire à la sortie
                                       de la section critique.

        STQ   R31, lock_variable ;     Libération du bit de verrouillage
        ...
déjà_validé :
        "code d'attente pour laisser
        le temps à un autre proces-
        seur de sortir de la section
        critique"
        BR boucle ;
échec_stq :
        "Code d'attente pour éviter un
        rebouclage trop rapide"
        BR stq_c_bcle ;
```

7.3.2 Ordre des lectures /écritures

La mise en œuvre de systèmes multiprocesseur à mémoire partagée pose le problème de l'ordre des accès à la mémoire par des processeurs distincts. La définition de l'architecture Alpha traite ce problème. La mise en œuvre d'un multiprocesseur doit respecter un certain ordre sur les requêtes à la mémoire afin de permettre la portabilité des applications et/ou le déterminisme de certaines applications.

Pour le DEC 21164, les transactions sur une *même case X* de la mémoire doivent être séquentiellement cohérentes ; c'est à dire que les résultats d'une suite de requêtes sur la case X doivent correspondre à un quelconque ordre total respectant la condition suivante :

> *L'ordre induit sur les requêtes d'un même processeur sur la case X est le même que celui du séquencement des requêtes.*

De plus, les résultats de l'ensemble des requêtes doivent correspondre à un ordre partiel sur l'ensemble des requêtes respectants les ordres partiels d'accès à la mémoire par un même processeur induits par les instructions *Memory Barrier* et *Instruction Memory Barrier* (en gros tout accès séquencé avant une instruction *Memory Barrier* précède tout accès après cette instruction).

Ceci peut paraître relativement complexe, mais laisse une grande liberté pour l'implémentation. En particulier, l'ordre des écritures effectives en provenance d'un même processeur vers des zones mémoires distinctes n'a pas à respecter l'ordre de séquencement. Ceci permet d'avoir des distances différentes entre un processeur et les divers modules mémoires.

7.4 Conclusion

Les microprocesseurs MIPS R8000 et DEC 21164 ont été définis pour pouvoir être intégrés dans des machines parallèles à mémoire partagée. Ces deux microprocesseurs incluent dans leurs jeux d'instructions un support à l'exécution de programmes parallèles par le biais de primitives de synchronisation et d'instructions exécutées atomiquement.
Les processeurs MIPS R8000 et DEC 21164 mettent en œuvre des protocoles de cohérence de cache qui maintiennent l'intégrité des données dans de tels systèmes. On peut cependant remarquer que l'exploitation de ces microprocesseurs dans des machines parallèles nécessite un apport non négligeable de logique externe pour garantir des performances acceptables.
La prochaine génération de processeurs (MIPS R10000, Intel P6), permettra de connecter plusieurs processeurs sur un bus de grappe avec seulement la nécessité d'ajouter un composant externe d'interface avec les autres éléments du système. Une telle configuration permettra de réduire la complexité et le coût du système.
Par ailleurs, on peut espérer que les tentatives de normalisation de l'ordre des lectures/écritures sur les multiprocesseurs faites dans les définitions des architectures (DEC Alpha mais aussi SPARC 9 et Pentium) permettront l'émergence de programmes parallèles portables.

Chapitre 8

Support de mesures de performances

8.1 Introduction

Le DEC 21164 et le Power2 intègrent des mécanismes matériels de mesure de performance qui permettent aux utilisateurs d'analyser les interactions matérielles et logicielles sur des applications exécutées par ces processeurs. Deux raisons motivent ce type de mesure :

1. elles permettent d'extraire des informations importantes sur le design de futurs systèmes en identifiant des goulets d'étranglements de l'architecture ;

2. elles permettent une mise au point orientée performance de logiciels.

Par exemple, il est particulièrement utile d'identifier les opportunités de séquencement d'instructions sur une machine dans le but d'améliorer les algorithmes de génération de code d'un compilateur. Par ailleurs, beaucoup d'optimisations effectuées par les compilateurs sont basées sur des heuristiques qui peuvent être bénéfiques pour un programme mais inadaptées sur d'autres. Aussi, il est particulièrement utile d'examiner les effets d'une optimisation pour un programme particulier sur d'autres applications, afin de s'assurer que cette optimisation ne dégrade pas les performances globales. De plus, il est possible avec ce type de mise en œuvre, d'examiner les caractéristiques d'une application plus facilement qu'autrement.

Aussi le DEC 21164 et le Power2 mettent en œuvre de tels mécanismes à l'instar d'autres architectures telles le CRAY Y-MP ou le Pentium d'Intel.

8.2 Support de mesures de performances sur le DEC 21164

Le DEC 21164[1] inclut des mécanismes qui lui permettent de comptabiliser certains évènements. Trois compteurs sont fournis. Ils permettent de mesurer des évènements sélectionnés parmi les suivants :

- les émissions d'instructions ;

- les non-émissions ;

1. Avertissement : l'implémentation de ces mécanismes est spécifique au 21164

- le total des cycles ;
- la non-utilisation du pipeline ;
- les gels du pipeline ;
- les défauts de cache et les mauvaises prédictions de branchements ;
- la comptabilisation de diverses classes d'instructions.

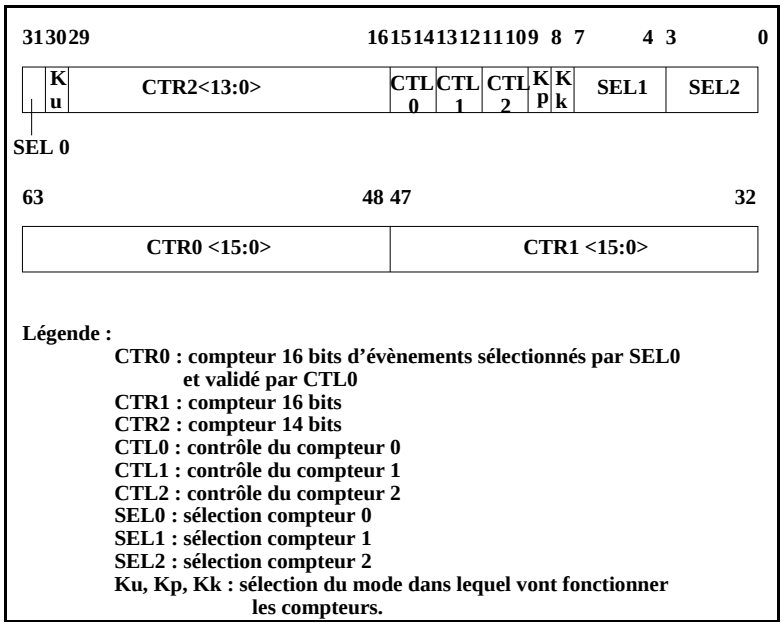


FIG. 8.1 - *Registre de mesure des performances*

De plus, le DEC 21164 fournit une broche d'entrée supplémentaire pour mesurer des évènements externes. Ces trois compteurs, ainsi que les bits de configuration, sont implémentés sous la forme d'un seul registre interne du processeur (*Internal Personnal Register*) appelé *Performance Counter Register*. Ce registre est représenté figure 8.1. Il peut être accédé dans tous les modes.

8.3 Support de mesures de performance sur le Power2

Le Power2 inclut un contrôleur de mesures de performance constitué de 22 compteurs de 32 bits et d'un registre de contrôle qui permet de sélectionner les évènement que l'on souhaite observer. Ces compteurs, ainsi que les registres peuvent être lus et écrits par l'intermédiaire d'opérations d'entrées/sorties. Cinq compteurs sont dédiés à chacune des unités de calcul entière et flottante (FXU et FPU), de chargement et d'émission des instructions (ICU) ainsi qu'à l'unité de contrôle de la mémoire (SCU). Un champ de quatre bits est réservé pour chacune de ces unités dans le registre de contrôle et permet de sélectionner cinq événements par unité parmi les 16 possibles. Pour chacune de ces unités la sélection de ces événements est indépendante des trois autres. La figure 8.2 schématise l'organisation de ce contrôleur.

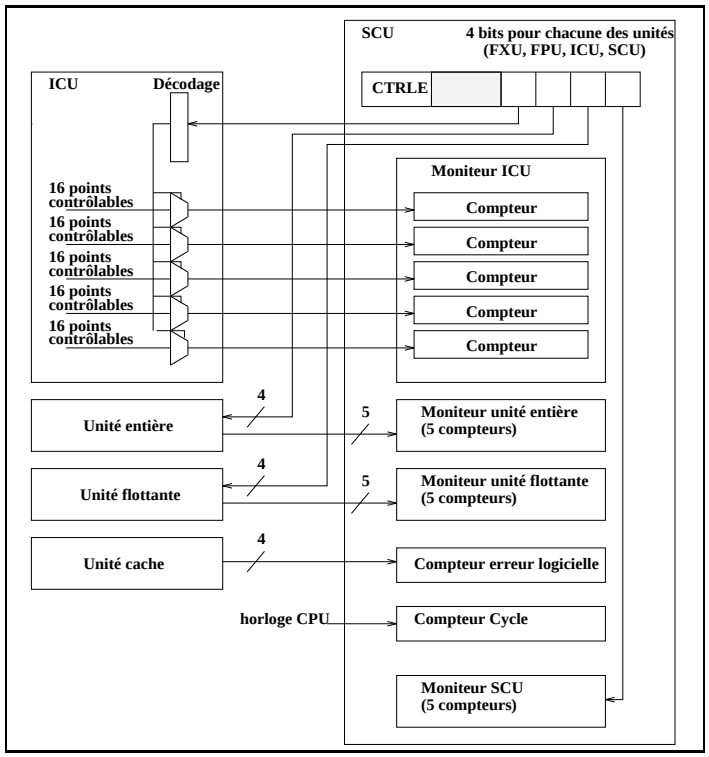


FIG. 8.2 - *Organisation du contrôleur de performance*

Jusqu'à 80 événements différents peuvent être suivis par composant, ce qui représente un total de 320 points contrôlables. Les 22 compteurs peuvent être actifs simultanéments. Seuls deux de ces compteurs ne sont pas paramétrables (le compteur d'erreur logicielle et le compteur de cycles). Par ailleurs, pour étudier plus en détails les performances mémoire et les entrées/sorties, un mode particulier a été défini, sous lequel l'unité de contrôle de la mémoire a à sa disposition les 20 compteurs. Ce mode illustre l'importance accordée par IBM à la compréhension des interactions sur la hiérarchie mémoire.

Ce contrôleur de performance permet d'examiner un processus particulier ou l'activité du noyau sans surcoût excessif. Ce mode de contrôle peut être validé à tout moment, pour chacun des processus. Le Power2 comprend un jeu de commandes destiné à acquérir de manière simple la valeur des compteurs. Deux cas peuvent se présenter :

- le temps d'exécution du programme est court. Il est alors possible de mettre à zéro le compteur, lancer le programme et récupérer les valeurs de compteurs.

- pour des programmes plus longs, où il est nécessaire de prendre en compte le dépassement de capacité des compteurs (compteurs 32 bits, à une fréquence de 66 MHz, le temps d'exécution doit être inférieur à 65 secondes), il est nécessaire de les sauver périodiquement. Aussi, ce jeu de commande permet de sauver ces compteurs à une fréquence spécifiée par l'utilisateur.

Dans ce dernier cas, le surplus engendré à chaque sauvegarde de registres est équivalent à un basculement de contexte. Il est aussi possible d'affiner les mesures en insérant directement dans le programme des appels au système à destination de registres de contrôle pour n'analyser que

certaines portions du code. L'accès au contrôleur de performances est fourni par une extension du noyau. Cette extension fournit un ensemble d'appels systèmes qui permettent de gérer les registres de contrôle et manipulent les compteurs.

8.4 Conclusion

Ce chapitre présente quelques uns des dispositifs mis en œuvre sur les processeurs pour analyser l'ensemble des interactions logicielles et matérielles. Cette méthode d'analyse présente cependant certains inconvénients :

- inconvénients liés à l'intégration technologique : problème lié à la limitation des broches et à l'utilisation d'une surface dédiée à des outils de mesure qui pourrait être utilisée plus efficacement pour accroître les performances.
- augmentation globale du coût du système du fait de l'apport de ces outils ;

Actuellement, ces mécanismes sont essentiellement utilisés pour la mise au point des systèmes et du logiciel associé (par exemple le compilateur) et ont donc un rôle limité. Une manière de rentabiliser la mise en œuvre de ces mécanismes serait d'apporter une aide au développeur par le biais d'une méthode d'optimisation des applications basée sur l'analyse des mesures de performances.

Chapitre 9

Interface bus et chemins de données

9.1 Introduction

Deux des processeurs étudiés sont multicomposants. L'échange de données et d'informations de contrôle ou de synchronisation s'effectue par l'intermédiaire d'un ensemble de bus qui permettent le fonctionnement du système. Par ailleurs, les processeurs étudiés implémentent des interfaces bus qui leur permettent la communication avec l'extérieur. Afin de fiabiliser l'échange des données, des mécanismes de détection et de correction d'erreur sont mis en œuvre.

Nous arborderons principalement dans ce chapitre l'aspect interface de communication avec l'extérieur, dans la mesure où ces processeurs sont destinés à être utilisés dans des applications requérant de nombreux échanges de données (applications graphiques, serveurs de données, etc...)

9.2 Chemins de données sur le MIPS R8000

Le MIPS R8000 est un processeur multicomposants. Pour permettre le fonctionnement du système, un certain nombre de bus permettent la communication et la synchronisation des opérations. Le *Tbus* est l'un d'entre eux, et il joue un rôle particulièrement important au sein de cette architecture.

Le Tbus

Le TBus est un bus de 80 bits à 75 MHz, utilisé pour la communication entre l'unité entière, l'unité flottante et le *contrôleur de cache*. Le possesseur et le format de ce bus change selon le type de l'opération exécutée.

Communication entre l'unité entière et l'unité flottante

Les instructions sont émises par l'unité entière vers l'unité flottante à travers le Tbus. Quatre formats de transmission sont utilisés selon la nature de l'opération à effectuer :

- *Transmission d'instructions.* Ce format est le mode normal d'utilisation du Tbus. Il comprend deux opérations flottantes arithmétiques, chacune de 28 bits de largeur et deux opérations mémoires flottantes, chacune de neuf bits de largeur (destination du registre flottant et information d'alignement), plus un bit de validité par instruction.

- *Ecriture entière en mémoire.* Ce mode concerne l'écriture d'un entier dans le *streaming cache*. Les données sont transmises sur le Tbus et placées sur les broches d'écriture de l'unité flottante. Dans ce mode, le Tbus transmet les 64 bits de donnée entière avec des informations d'alignement.

- *Transfert de données entières vers l'unité flottante.* Cette opération transfert le contenu d'un registre entier vers un registre flottant. Le format de ce mode est similaire au format précédent à l'exception des informations d'alignement remplacées ici par le numéro de registre flottant destinataire. Les 64 bits de donnée sont transmis sur les 64 bits de poids faible.

- *Transfert de données à partir de l'unité flottante vers l'unité entière.* Ce mode est similaire au précédent si ce n'est que le sens de la transaction est inversée. C'est le seul cas où l'unité flottante dirige le Tbus.

Communication entre l'unité entière et le contrôleur de cache

Lors d'opérations normales, le Tbus est utilisé pour transférer les informations entre les unités entière et flottante. Durant ce temps, le contrôleur de cache joue le rôle d'interface avec le reste du système. Quand un défaut sur le cache secondaire, ou une opération de cohérence sur le bus survient, le contrôleur de cache utilise alors le Tbus pour communiquer avec l'unité entière et les caches. Il est à noter qu'avant toute opération de mise à jour de la mémoire à partir du cache secondaire, le contrôleur de cache sollicite l'unité entière afin de s'assurer que la *Store Address Queue* ne contient pas une donnée en cours de mise à jour. Si tel est le cas, le contrôle du Tbus est alors rendu à l'unité entière afin que les données de la SAQ soient écrites dans le cache externe, avant d'être recopiées par le contrôleur de cache dans la mémoire.
L'orientation et la définition des signaux varient selon le possesseur du Tbus. Seuls les 72 bits de poids faible sont utilisés par le contrôleur de cache pour la communication.

9.3 Le Power2

9.3.1 Chemins de données sur le Power2

Le Power2 est le processeur le plus complexe du point de vue de l'émission et de l'exécution des instructions. Comme on l'a vu, chacune des unités dispose d'un certain degré d'indépendance. Pour éviter des interblocages au niveau des bus qui sont généralement des goulots d'étranglement de l'architecture, le Power2 met en œuvre une structure de bus d'instructions et de données qui lui permet d'alimenter les diverses unités fonctionnelles de manière satisfaisante. La figure 2.1 (page 35) représente l'essentiel de ces bus (nous renvoyons le lecteur à [16] pour plus de détails sur les bus internes).

Remarque : à la différence du MIPS R8000, il n'existe aucun chemin de données entre l'unité entière et flottante (par conséquent tout échange de données nécessite une écriture et une lecture dans le cache).

9.3.2 Interface bus sur le Power2

L'interface mémoire

L'interface mémoire est un bus synchrone d'adresses et de données séparées, qui permet au processeur, ainsi qu'aux composants d'entrées/sorties, d'accéder à la mémoire principale. Il constitue un bus de données de 256 bits (dans le cas d'une configuration à 256 Ko), offrant à l'utilisateur un taux de transfert supérieur à 2000 Mo/sec.

Le bus système d'entrées/sorties

Conséquence de l'évolution des applications (multimédia), l'aspect entrées/sorties du Power2 a été considérablement amélioré par rapport à son prédécesseur.

Le bus système d'entrées/sorties est un bus dédié à la communication entre les unités d'exécution et les unités d'entrées/sorties. Ce bus contient 98 signaux d'entrées/sorties répartis comme suit :

- 86 signaux sont bidirectionnels et consistent en un bus multiplexé d'adresses et de données (qui inclut huit bits de parité), un champ de contrôle de huit bits (avec un bit de parité) plus cinq bits de contrôle d'étiquettes (adresse valide, donnée valide, etc...).

- Les douze autres signaux sont unidirectionnels et sont utilisés pour l'arbitrage du bus.

Le bus système d'entrées/sorties permet des transferts en blocs de type DMA, des interruptions, etc... Il supporte jusqu'à quatre unités de contrôle d'E/S, chacune de ces unités pouvant gérer de quatre à huit canaux permettant des transferts à un taux de 80 Mo/sec. Pour permettre une telle bande passante, un ensemble de tampons servent à *bufferiser* les données, l'ensemble de ces opérations s'effectuant sous le contrôle de la SCU. Afin de masquer la latence du système mémoire (DRAM), des mécanismes de préchargement de données sont également mis en œuvre.

Caractéristiques des canaux d'entrées/sorties (*Micro Channel Architecture*)

Le Power2 peut contrôler jusqu'à 32 de ces canaux développés par IBM dans le but d'interfacer directement avec la gamme la plus vaste de produits. Ils sont donc à usage général et fournissent de hauts niveaux de performances par le biais de procédures de transfert de données en mode groupé (initialisation d'une transaction, puis transfert de plusieurs données de 32 bits) alliées à des mécanismes de détection d'erreur (parité sur les adresses et les données). Cette architecture facilite le *plug and play* en imposant aux composants ou cartes, de s'identifier à la mise sous tension en envoyant au CPU leur adresse ainsi que leur besoin en mémoire.

9.4 Interface bus sur le DEC 21164

Le DEC 21164 étant un monocomposant, nous ne développerons ici que l'ensemble des bus destinés à la communication avec le reste du système. On peut cependant remarquer que la totalité des accès au sein du DEC 21164 sont sur 128 bits.

Interface avec le système

Le 21164 met en œuvre avec le reste du système un bus de données d'une largeur de 128 bits ainsi qu'un bus d'adresses séparé. Bien que le 21164 ait un bus système similaire à celui du 21064, les protocoles sont suffisaments différents pour employer un nouvel ensemble de composants d'interface. Cet ensemble est constitué de deux types de composants, l'un destiné au routage des données, l'autre fournissant les signaux de contrôle et d'adresses. La figure 9.1 reprend le diagramme d'interface du DEC 21164 avec le reste du système et illustre les divers bus d'interface. Chacun de ces composants contient un ensemble de tampons qui permettent de *bufferiser* les transactions.

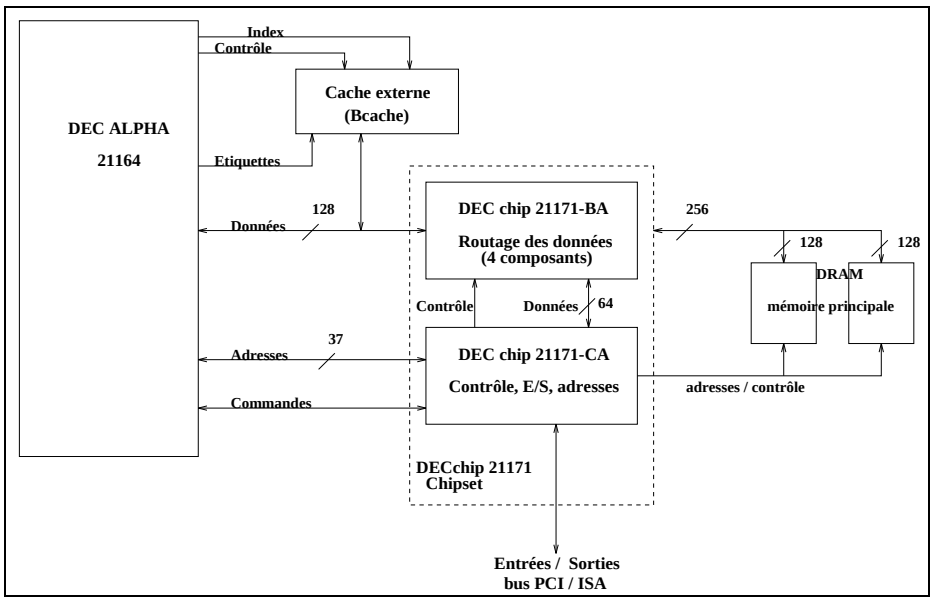


FIG. 9.1 - *Circuits d'interfaçage du DEC 21164 avec le système*

Le Chipset du DEC 21164 offre un support à diverses interfaces de bus :

- Bus local *PCI (Peripheral Component Interconnect Bus)* : trois emplacements dédiés PCI (un 64 bits, et deux 32 bits), plus un emplacement 64 bits PCI/ISA. La vitesse du bus PCI est paramétrable de 25 à 33 MHz. Une passerelle du bus PCI vers le bus ISA est également implémentée.
- bus *ISA (Industry-Standard Architecture Bus)* : deux emplacements, plus un partagé avec le bus PCI.

Divers autres accès sont également disponibles tels que ports parallèles, séries, etc...

Bus PCI (Peripheral Component Interconnect Bus)

Le bus *PCI*, développé par Intel pour les *xxx86*, s'impose désormais comme un standard. Il présente de nombreuses similitudes avec l'architecture *Micro Channel* développé par IBM : bus 32 bits, *plug and play*. Il s'est cependant plus largement répandu du fait de la politique plus libérale menée par Intel.

Le bus *PCI* fonctionne en mode 32 bits et supporte des extensions 64 bits. Ce bus gère des transferts de données à un taux de 132 Mo/sec. Du fait de l'autoconfiguration des périphériques lors de la mise sous tension, les données peuvent être envoyées directement à une adresse sans requérir d'instructions spéciales. D'où un usage immédiat dans les applications multimédias où le son et la vidéo peuvent être directement transférés en un flot ininterrompu.

9.5 Protection des données

Afin d'assurer l'intégrité des données, les processeurs mettent en œuvre divers mécanismes de détection et de correction d'erreurs.

9.5.1 Mécanismes de correction et de détection d'erreur sur le DEC 21164

Le DEC 21164 dispose de deux modes de protection de données, l'un reposant sur un code de correction d'erreur (ECC), l'autre reposant sur un bit de parité par octet. L'un ou l'autre de ces modes peuvent être validés à partir d'un registre interne de contrôle.

En mode ECC, le DEC 21164 met en œuvre aux niveaux des échanges avec le cache externe et la mémoire principale une correction d'erreur pour chaque bloc de 64 bits. Ce code est généré par le CPU (il permet la détection d'erreurs doubles et la correction d'erreur simple). Que ce soit en mode parité ou en mode ECC, une donnée incorrecte en provenance du cache externe ou de la mémoire provoque une exception.

Le DEC 21164 dispose de plus d'une protection de parité par octet sur les caches internes. Les bits de parité générés sont envoyés vers le système mémoire lors des opérations d'écriture (sur un bus spécifique), et sont testés à chaque opération de lecture. Ce mode de protection ne permet pas la correction d'erreur. En cas d'erreur de parité ou ECC, l'adresse de la donnée incorrecte est si possible sauvegardée dans un registre interne au processeur. Le système d'exploitation peut alors ré-essayer le flot d'instructions ou décider de mettre fin aux processus qui accèdent à la zone mémoire affectée.

9.5.2 Mécanismes de détection d'erreur sur le MIPS R8000

Le MIPS R8000 met en œuvre au niveau des communications entre les composants (unité entière, unité flottante et cache externe) une protection par bits de parité. Le choix de la parité paire ou impaire, ainsi que sa validation est effectuée à partir d'un registre interne. Lorqu'une erreur de parité est détectée, une interruption est générée (traitement de type imprécis).

9.5.3 Mécanismes de détection et de correction d'erreur sur le Power2

Le Power2 met en œuvre au niveau de son cache de données une détection d'erreur double et une correction d'erreur simple. Le code généré contient 32 bits de données, sept bits de test et un bit supplémentaire (*Spare bit*). Chaque mot est codé selon un code de Hamming (modifié) au moment de son écriture dans la mémoire, qui sera testé au moment de sa relecture. Dans le cas d'une erreur simple, la donnée est corrigée et le syndrome ECC (champ de 8 bit qui indique la position de l'erreur) est écrit dans un registre. Ce registre peut être lu par logiciel et permet par exemple d'isoler un défaut particulier sur la mémoire (mécanisme semblable à celui du DEC 21164). Dans le cas de la détection d'une erreur matérielle sur la mémoire, la SCU valide alors le *spare bit* qui

vient remplacer le bit défectueux.

Pour fiabiliser les communication inter-composants, le Power2 implémente également une protection par parité au niveau de chacun de ses bus internes.

Conclusion

En dépit d'un pipeline relativement profond, le pipeline du DEC 21164 est relativement proche des cinq étages classiques d'un processeur RISC, si ce n'est que la complexité du séquencement superscalaire de quatre instructions nécessite trois étages. L'un des problèmes posés par une fréquence d'horloge élevée est l'accès à la mémoire en un cycle. Un premier niveau de cache complété par un deuxième niveau de plus grande capacité apporte une solution efficace aux accès mémoires à une fréquence de 300 MHz. On notera cependant le problème de l'accès à des données de 8 ou 16 bits qui amoindrit l'efficacité de cette architecture sur les applications multimédias.

Le MIPS R8000 présente une architecture originale avec son premier niveau de cache d'instructions et de données entières et son large cache secondaire. L'architecture découplée de ce processeur à travers les mécanismes de *Floating-Point Queue* et de *Store Address Queue* constitue une nouvelle étape de l'évolution des architectures des microprocesseurs vers l'exécution dans le désordre. À l'exploitation du parallélisme entre les instructions (superscalaire), vient désormais s'ajouter l'exploitation de l'asynchronisme des unités fonctionnelles et le dépassement des écritures par les lectures sur les mémoires. L'ensemble de ces caractéristiques, ajouté aux nouvelles instructions de multiplication-addition, prédispose ce processeur au domaine des applications flottantes.

Sur le Power2 d'IBM une très grande complexité a été introduite dans les mécanismes de séquencement. Il met en œuvre la plupart des techniques évoluées d'architectures de processeurs telles que : renommage de registres, unités d'exécution découplées, exécution spéculative ainsi que dans le désordre. Le nombre de transistors réservé à la logique de gestion de cette architecture traduit la complexité d'une telle mise en œuvre. Il est cependant surprenant que le Power2 n'implémente pas de prédiction dynamique de branchement, car l'une des particularités des architectures superscalaires est d'exacerber le besoin d'anticiper l'exécution du code.
L'étude de ces processeurs met en évidence l'importance croissante des mécanismes de séquencement : cache d'adresses de branchement sur le MIPS R8000, unités de décodage et exécution spéculative des instructions sur le Power2, schéma de prédiction de branchement à deux bits sur le DEC 21164, et utilisation d'instructions *conditional-move* sur le MIPS R8000 ainsi que sur le DEC 21164.

Le Power2 est un processeur à architecture 32 bits. Actuellement, peu de programmes nécessitent l'usage effectif de plus de 32 bits d'adresse mais les architectures 64 bits devraient très rapidement s'imposer dans le domaine des grandes bases de données interactives, des applications multimédias, de la simulation numérique intensive, etc...

Les processeurs de ces prochaines années utiliseront une architecture découplée avec une organisation asynchrone superscalaire et une exécution dans le désordre. L'utilisation de mécanismes permettant d'anticiper l'exécution du code (table de prédiction, cache d'adresse de branchement, exécution spéculative, pile de retour, etc...) jouera également un rôle primordial sur ces architectures. À noter cependant que de nombreuses études montrent que le niveau de parallélisme entre les instructions est généralement de trois à quatre opérations possibles par cycle. Aussi, construire des processeurs avec un très grand nombre d'unités fonctionnelles (> 8) semble inutile.

Par ailleurs, l'émergence des applications multimédias laisse présager une évolution des architectures au niveau des unités d'exécution et des jeux d'instructions. On peut citer à titre d'exemple l'UltraSparc de Sun qui introduit dans son architecture deux unités graphiques ainsi que de nouvelles instructions (similaires aux extensions de l'HP PA-7100). Celles-ci permettent de traiter en parallèle des données de 8, 16 ou 32 bits (par exemple huit résultats de huit bits peuvent être calculés par une unique instruction). L'impact de ce nouveau type d'architecture devrait être immédiat dans le domaine des vidéoconférences et des applications audio et vidéo.

Les processeurs MIPS R8000 et Power2 sont probablement les derniers processeurs (destinés au marché haut de gamme) à avoir une fréquence de fonctionnement inférieure à 100 MHz (en dehors des marchés spécifiques à basse consommation). L'évolution technologique permet de maîtriser des fréquences de plus en plus importantes (300 MHz pour le DEC 21164).

L'évolution actuelle des architectures internes de processeurs (séquencement dans le désordre, etc...) conduit à s'interroger sur l'évolution du concept RISC. Ces architectures sont bien loin du pipeline unique de la fin des années 1980 mais en ont conservé les caractéristiques essentielles du jeu d'instructions (taille unique, architecture *load/store*). Il est à remarquer qu'à performances comparables, la complexité de mise en oeuvre d'un processeur superscalaire à jeu d'instructions RISC est nettement moins élevée que la complexité d'un processeur superscalaire CISC (4,5 millions de transistors de logique pour l'Intel P6 contre 2,3 millions pour le MIPS R10000).

Remerciements

Nous tenons à remercier François Bodin (IRISA), Sébastien Hily (IRISA), Yvon Jégou (IRISA), Philippe Joubert (IRISA), Stéphan Jourdan (IRIT) et Thierry Vauléon (IRISA) qui ont vivement contribué à la qualité de cette étude par leur Béta-lecture.

Annexe A

Jeux d'instructions

Le tableau A.1 récapitule les divers modes d'adressage disponibles sur l'ensemble des machines. Les processeurs RISC offrent généralement les 4 modes d'adressage.

Modes d'adressage	DEC 21164	MIPS R8000	Power2
absolu	R31 + immédiat	R0 + immédiat	R0 + immédiat
indirect	registre + 0	registre + 0	registre + 0
basé	registre + immédiat	registre + immédiat	registre + immédiat
indexé	-	registre + registre [a]	registre + registre
préincrémenté	non	non	oui

TAB. A.1 - *Les modes d'adressage*

[a] Ne concerne que les opérations flottantes

Le tableau A.2 récapitule certaines caratéristiques des jeux d'instructions étudiés. Pour plus de détails, se référer au chapitre 1.

Processeurs / Types d'instructions	DEC 21164	MIPS R8000	Power2
Transfert de données	Pas d'instructions manipulant les octets et les mots de 16 bits.	Nouveau mode d'adressage indexé pour les opérations flottantes.	Nouvelles instructions d'accès à la mémoire pour les données flottantes permettant le chargement de quadruple-mot (128 bits) dans deux registres flottants adjacents.
Arithmétiques et logiques	instructions *conditional-move.*	4 nouvelles instructions qui permettent le déplacement de données de registre à registre en testant un code condition ou un registre source.	2 types d'instructions coexistent, l'un affectant les codes conditions et l'autre non
Contrôle	Ne met pas en œuvre de codes conditions.	Technique du branchement retardé, possibilité de sauvegarder l'adresse de retour dans le registre R31.	Possibilité de sauvegarder l'adresse de retour dans le registre *Link register*. Gestion de boucles découplée de l'exécution.
Flottantes	Fournit les opérations de base ainsi que des instructions de conversion entre les formats VAX et IEEE. Pas d'instructions de racine carrée.	Introduction de 4 instructions de multiplication-addition flottante. Définition de 8 codes conditions et d'une nouvelle instruction de comparaison associée.	Mise en œuvre de la racine carrée émulée logiciellement sur le Power1. À noter aussi la présence des instructions de multiplication-addition.
Particulières[a]	CALLPALL, Memory Barrier, TRAPB	PREF, SSNOP	sync, instructions de manipulation de chaînes de caractères

TAB. A.2 - *Récapitulatif des jeux d'instructions*

[a] Ces instructions sont développées au chapitre 1.6.5

Annexe B

Multichip Ceramic Module

L'une des innovations du Power2 concerne sa mise en œuvre technologique. Cette annexe illustre les recherches effectuées dans le domaine de la méthodologie de conception de composants, l'intégration technologique et les divers problèmes de communication inter-composants et de conditionnement. L'évolution constante de nouveaux outils de conception est un passage obligatoire pour les constructeurs pour pouvoir élaborer de nouvelles architectures.

Nous ne développerons ici que ce qui concerne le conditionnement (*Packaging*) du Power2. C'est un aspect important du microprocesseur dans la mesure où il doit dissiper la chaleur générée par la fréquence de fonctionnement et que ses caractéristiques électriques déterminent la vitesse de transfert vers l'extérieur.

Le *Multichip Ceramic Module*, conçu et produit par IBM accroît les performances et la densité du circuit, et améliore la connectivité.

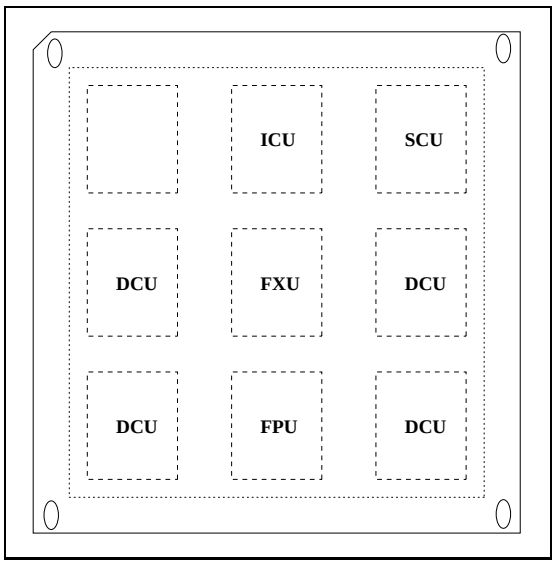


FIG. B.1 - *Multichip Ceramic Module - POWER2*

Ce *packaging* permet de décongestionner le routage sur la carte, et réduit la connectivité intercomposant tout en limitant le nombre de chemins critiques. L'utilisation de céramique offre les avantages d'un haut taux de dissipation de chaleur et permet d'ajouter plusieurs niveaux de routage. Ce type de conditionnement est déjà largement répandu dans des domaines sensibles tels ceux de l'aérospaciale ou militaire, qui acceptent son surcoût en retour de l'augmentation des performances et de la densité. Son usage commence à se généraliser dans le domaine des processeurs ultra-rapides, le Power2 étant le premier processeur de station de travail à en tirer partie. Le substrat céramique contient vingt couches de routages de signaux (44 couches au total) qui inclut plus de 90 mètres de routage.

Le module contient 736 broches dont 512 d'entrées/sorties. Les broches d'alimentation sont essentiellement placées dans les coins de manière à protéger les broches particulièrement sensibles de toute dégradation physique. De plus, cela permet de placer à l'intérieur les signaux critiques, raccourcissant ainsi au maximum leur routage. Les 8 composants et leur placement sur le module sont illustrés figure B.1. Les composants de contrôle d'entrées/sorties ainsi que la génération de l'horloge sont externes au module et sont placés sur la carte.

D'autres constructeurs pensent également tirer partie de ce conditionnement dans le futur. MIPS Technologies examine le placement du R8000 et du R8010 dans un package MCM. On peut également citer Hewlett Packard (PA-8000), Intel (Intel P6 avec un conditionnement MCM PGA à deux cavités), Hal Computer Systems (Sparc64, annoncé pour le $3^{ième}$ trimestre de l'année 95).

Annexe C

Performances

Nous n'aborderons que brièvement cet aspect dans la mesure où ce rapport repose essentiellement sur l'étude des mécanismes mis en œuvre par les constructeurs au sein de leur architecture. C'est pourquoi, aucune étude de performance n'a été menée et les chiffres repris ci-dessous sont ceux parus dans la presse. Le tableau C.1 nous permettra cependant de situer les performances de chacun.

TAB. C.1 - *Performances des microprocesseurs*

	POWER 2	MIPS R8000	DEC 21164
SPECint92	126	100	290 (266 Mhz) 330 (300 Mhz)
SPECfp92	260	310	440 (266 Mhz) 500 (300 Mhz)
MIPS	430	300	1200
MFLOP	286	300	600

Il va sans dire que ces chiffres sont à interpréter avec précautions, les performances réelles d'un processeur étant fonction du système d'exploitation, du système mémoire mis en œuvre ainsi que du système d'entrées/sorties.

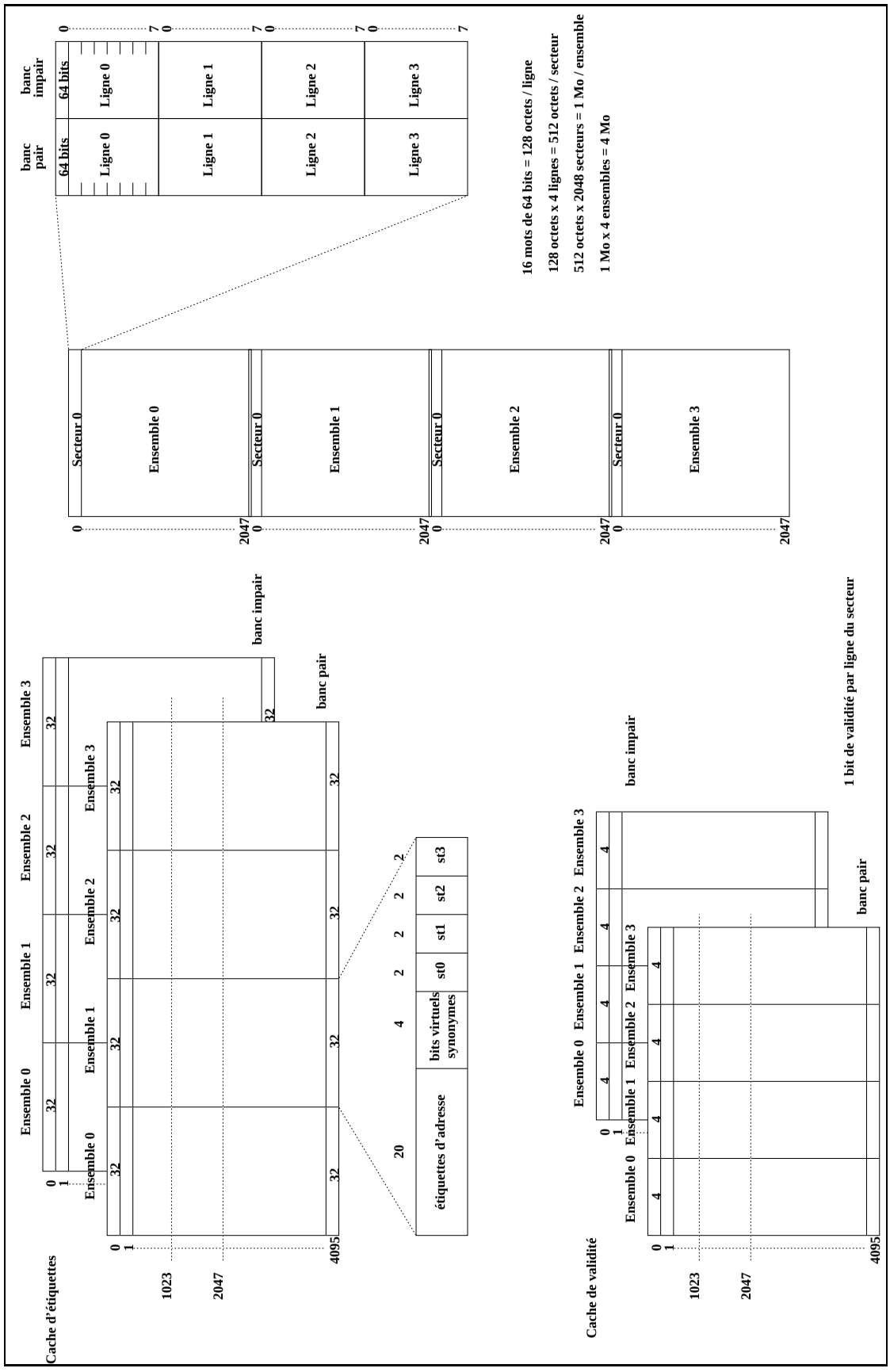
Annexe D

Organisation du cache externe du MIPS R8000

La figure D.1 illustre l'organisation physique du cache externe (ou *streaming cache*) du MIPS R8000. Nous avons schématisé les trois éléments principaux de ce cache : le tableau d'étiquettes, la RAM de validité et enfin le cache de données. L'accès à ce cache est pipeliné selon le principe décrit au chapitre 5.4 (page 85). Cette figure représente le *streaming cache* dans sa configuration minimale (4 Mo). Vingt-deux bits d'adresses sont donc nécessaires pour adresser n'importe quel octet de ce cache. Le tableau ci-dessous reprend la décomposition de cet adressage.

DATSA [1 : 0]	2 bits	Sélection d'un des 4 ensembles du cache (*Data Set Address*)
A [19 : 9]	11 bits	Sélection d'un des 2048 secteurs d'un ensemble
A [8 : 7]	2 bits	Sélectionne l'une des 4 lignes d'un secteur
A [6 : 4]	3 bits	Sélectionne un mot de 128 bits sur les 8 constituants une ligne
A [3]	1 bit	Détermine le banc accédé : pair ou impair (on accède par conséquent à un mot de 64 bits)
A [2 :0]	3 bits	Sélection de l'un des octets dans le mot de 64 bits adressé

TAB. D.1 - *Adressage du cache externe dans le cas d'une configuration à 4 Mo*



RR n ° 2553 FIG. D.1 - *Organisation du streaming cache (configuration de 4 Mo)*

Bibliographie

[1] Patrice Laporte, André Seznec. *Une étude comparative des microprocesseurs MIPS R3000, SUN SPARC VERSION 7 et IBM POWER : Architectures et Performances.* Rapport de recheche, IRISA, Février 1992.

[2] André Seznec, Anne-Marie Kermarrec et Thierry Vauléon. *Etude comparée des architectures des microprocesseurs MIPS R4000, DEC 21164 et T.I. SUPERSPARC.* Rapport de recherche 1836, IRISA, Janvier 1993.

[3] André Seznec, Thierry Vauleon. *Etude comparative des architectures des microprocesseurs Intel Pentium et PowerPC 601.* Rapport de recherche 835, IRISA, juin 1994.

[4] John L. Hennesy and David A. Patterson. *Computer Architecture A Quantitative Approach.* Morgan Kaufmann, 1990.

[5] J.J. Dongarra, J. DuCroz, S. Hammarling and I. Duff. *A set of level 3 basic linear algebra subprograms.* ACM Transactions on Mathematical Software, Vol. 16(No. 1):pp. 1–17, 1990.

[6] *Alpha Architecture Handbook - Preliminary.* Digital Equipment Corporation, Maynard, Massachusetts.

[7] Linley Gwennap. *IBM regains performance lead with POWER2.* Microprocessor Report, 7(13), Octobre 1993.

[8] Linley Gwennap and Alisa Scherer. *IC manufacturing drives CPU performance.* Microprocessor Report, septembre 1993.

[9] Linley Gwennap. *Estimating IC manufacturing costs.* Microprocessor Report, août 1993.

[10] Linley Gwennap. *Packaging influences microprocessor cost.* Microprocessor Report, septembre 1993.

[11] Linley Gwennap. *Alternative packages emerge for processors.* Microprocessor Report, octobre 1993.

[12] Nathalie Drach. *Optimisation du Pipeline sur les Processeurs Monocomposants.* Thèse de l'Université de Rennes 1, 1994.

[13] *Alpha 21164 Microprocessor Hardware Reference Manual.* Digital Equipment Corporation, preliminary edition, september 1994.

[14] L. J. Shieh, E. L. Hannon, F. P. O'Connel. *POWER2 performance on engineering/scientific applications.* PowerPC and Power2 Technical Aspects of the New IBM RISC System/6000, 1994.

[15] Norman P. Jouppi. *Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers.* Proceedings of the 17th Intl. Symp. on Computer Architecture, pages 364–373, 1990.

[16] *PowerPC and POWER2, Technical Aspects of the New IBM RISC System/6000.* IBM, first edition edition, 1994.

[17] Linley Gwennap. *Digital leads the pack with 21164.* Microprocessor Report, 12(8), septembre 1994.

[18] Linley Gwennap. *TFP designed tremendous floating point.* Microprocessor Report, 7(11), Août 1993.

[19] MIPS Technologies. *TFP Microprocessor Chip Set, Preliminary Product Information.* Technical Report, MIPS Technologies, Incorporated, October 1993.

[20] Peter Yan-Tek Hsu. *Design of the R8000 microprocessor.* MIPS Technologies, Inc., 1994.

[21] Steve Rhodes. *MIPS, TFP Microprocessor Chip Set, Users Manual.* Silicon Graphics Computer Systems, MIPS Technologies, Incorporated, 2011 N. Shoreline Blvd., revision 2.0 edition, April 1994.

[22] *PowerPC 601, RISC Microprocessor User's Manual.* Motorola, 1993.

[23] IRISA, *Journées microprocesseurs rapides*, Rennes, 10 janvier 1995.

Index

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399