MI01 – Automne 2011

IA32 – Extensions SIMD

Stéphane Bonnet

Poste : 52 56
Courriel : stephane.bonnet@utc.fr

Plan du cours

- Types de calculateurs
- Extension MMX
  - Types de données
  - Registres
  - Jeu d'instructions
- Un exemple : *chroma-key*

Types de calculateurs

Types de calculateurs

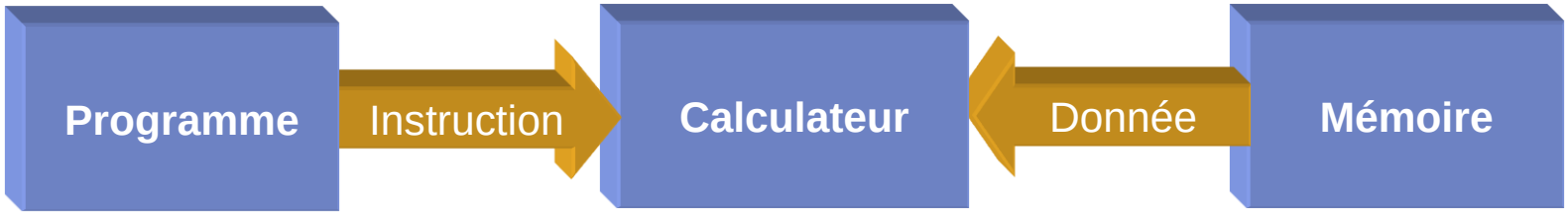
- Un ordinateur (séquentiel ou parallèle) opère en exécutant des instructions sur des données
  - Un flot d'instructions (algorithme) indique au processeur ce qu'il doit faire à chaque instant.
  - Un flot de données (les entrées de l'algorithme) est affecté par ces instructions.

Types de calculateurs

- Classification des machines parallèles (Michael Flynn)
- 4 classes :
  - Single Instruction Stream, Single Data Stream (**SISD**)
  - Multiple Instruction Streams, Single Data Stream (**MISD**)
  - Single Instruction Stream, Multiple Data Streams (**SIMD**)
  - Multiple Instruction Streams, Multiple Data Streams (**MIMD**)

Calculateurs SISD

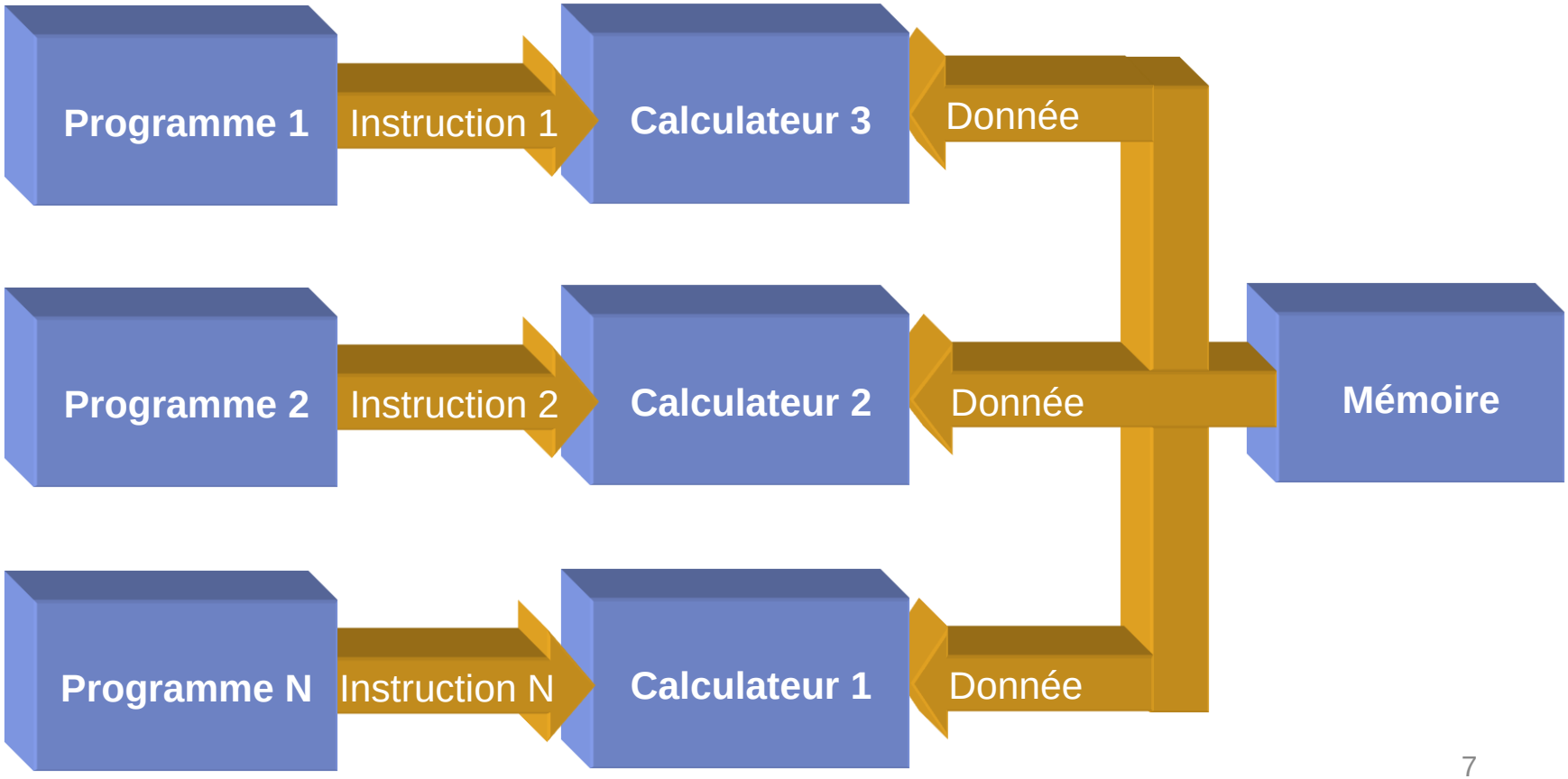
- Un seul processeur
  - Flot d'instructions unique
  - Flot de données unique



- Modèle le plus utilisé (Modèle de Von Neumann).
- Algorithmes **séquentiels.**

Calculateurs MISD

- N processeurs
  - N flots d'instructions
  - Flot de données unique

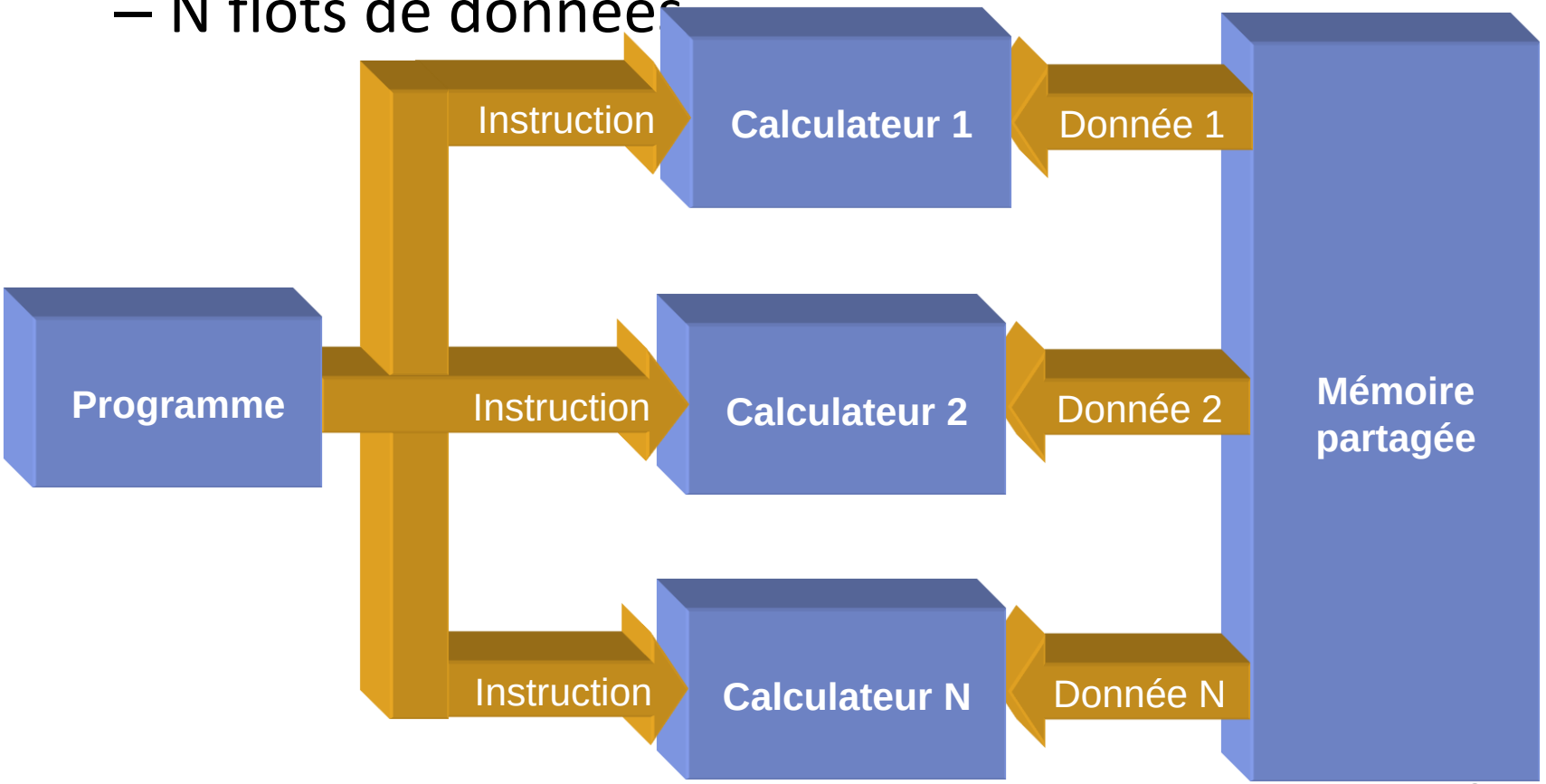


Calculateurs MISD

- Traitement simultané d'une donnée unique
- Parallélisme : chaque processeur (ou unité) réalise un traitement différent
- Algorithmes de classification (reconnaissance de formes) par exemple.

Calculateurs SIMD

- Un seul processeur
  - Flot d'instructions unique
  - N flots de données

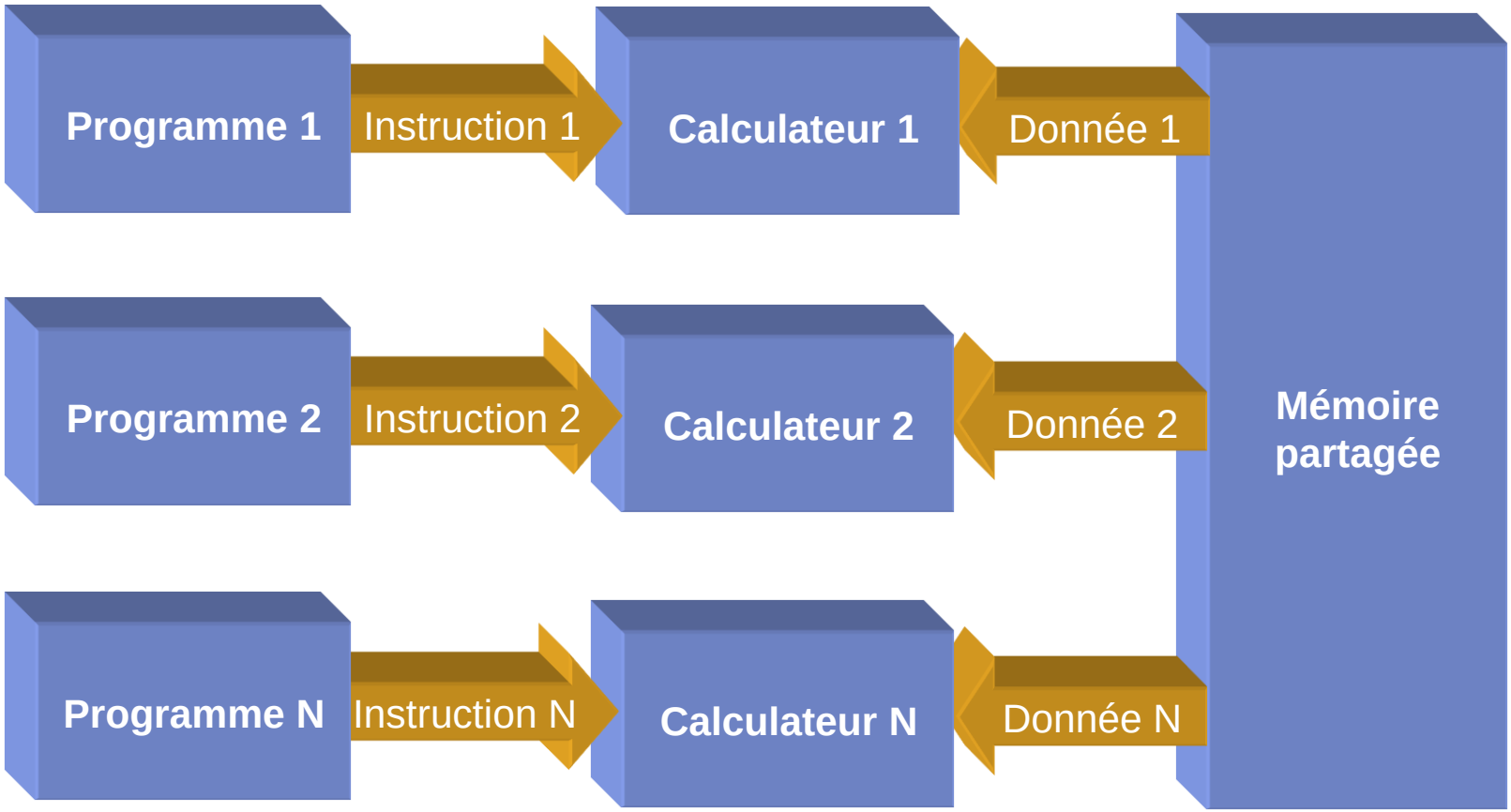


Calculateurs SIMD

- Traitement **synchrone** simultané de N données
- Les processeurs réalisent tous le même traitement.
- Plus général que *MISD* (plus grand nombre de problèmes adaptés)
- Problèmes divisibles en sous problèmes identiques solubles en même temps par le même programme (approche « *divide and conquer* »)

Calculateurs MIMD

- N processeurs
  - N flots d'instructions
  - N flots de données



Calculateurs MIMD

- Traitement **asynchrone** simultané de N données par N programmes
- Chaque processeur réalise un traitement possiblement différent
- Classe la plus générale
- Problème : les algorithmes asynchrones sont difficiles à concevoir, analyser et implémenter.
- Exemple : machines multiprocesseurs, calculateurs distribués…

Extension MMX (MultiMedia eXtensions)

MMX

- Processeurs de la famille Intel x86 de type *SISD*.
- On utilise de plus en plus d'applications multimédia, qui doivent traiter des sons, des images, des vidéos, ou générer des affichages complexes (jeux 3D…).
- Idée : inclure des instructions spécifiques pour accélérer l'exécution de ces applications.

MMX

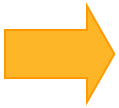
- Beaucoup d'algorithmes utilisés dans ces applications se décomposent facilement en sous problèmes parallèles.
- Caractéristiques communes :
  - Données entières de petite taille (pixels 8 bits, échantillons sonores 16 bits…)
  - Petites boucles répétées de nombreuses fois
  - Utilisation intensive de multiplications puis addition des résultats (ex : produit de convolution en traitement du signal)

MMX

- MMX (*MultiMedia eXtensions*) offre un ensemble d'instructions entières destinées au traitement de ces algorithmes :
  - architecture SIMD
  - 57 nouvelles instructions
  - 8 registres spécifiques de 64 bits
  - 4 nouveaux types de données.
- Ces instructions sont apparues pour la première fois en 1997 (Pentium MMX).

MMX – Types de données

- Comment faire travailler un calculateur *SISD* sur plusieurs données en même temps dans ajouter d'autres processeurs ?



Représenter des données différentes en les regroupant ensemble : le processeur peut réaliser le même traitement sur chaque donnée du même ensemble, mais la donnée de chaque instruction reste unique.

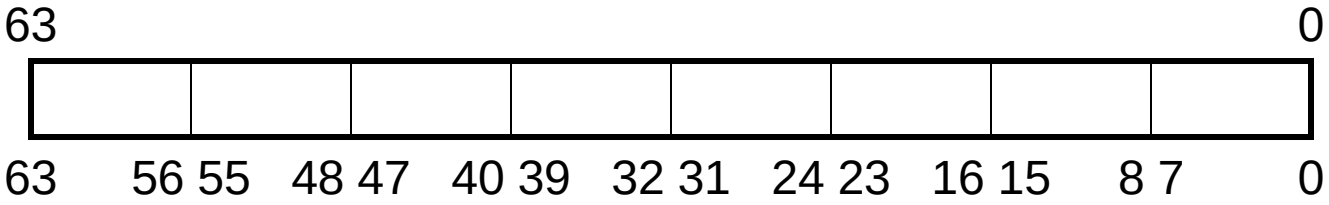
Ceci revient à « simuler » une architecture *SIMD* sur un calculateur *SISD*.

MMX – Types de données

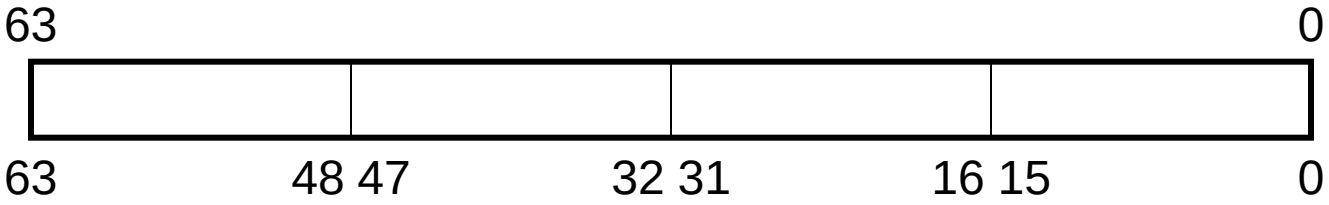
- Regroupement de données de taille identiques par paquets (*Packed data types*).
- La taille d'un paquet est toujours de 64 bits.
- Quatre types :
  - Paquet d'octets (*Packed Bytes*)
  - Paquets de mots (*Packed Words*)
  - Paquets de double mots (*Packed Double words*)
  - Quadruple mot (*Quad word*)

MMX – Types de données

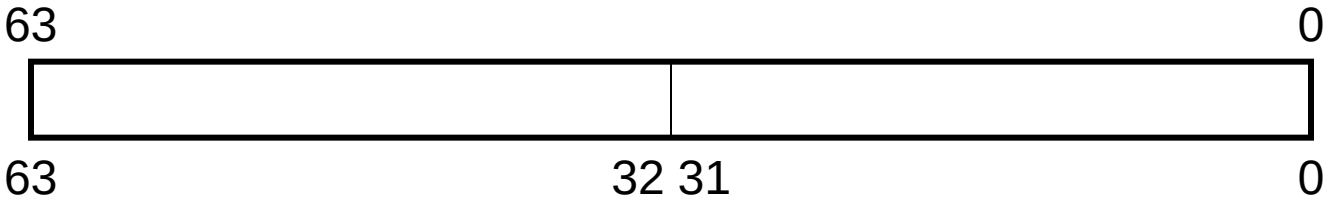
- Packed Bytes (8 éléments de 8 bits)



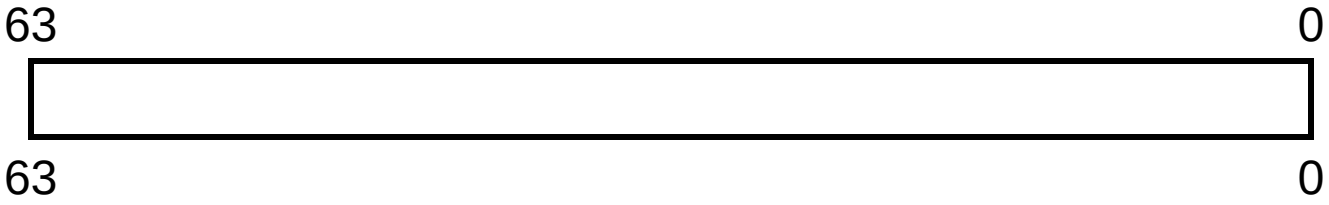
- Packed Words (4 éléments de 16 bits)



- Packed Double words (2 éléments de 32 bits)



- Quad word (1 élément de 64 bits)



MMX – Types de données

- Ces types contiennent l'ensemble des données qu'une instruction MMX doit traiter en parallèle.

➡ Le degré de parallélisme dépend du type des données à traiter :

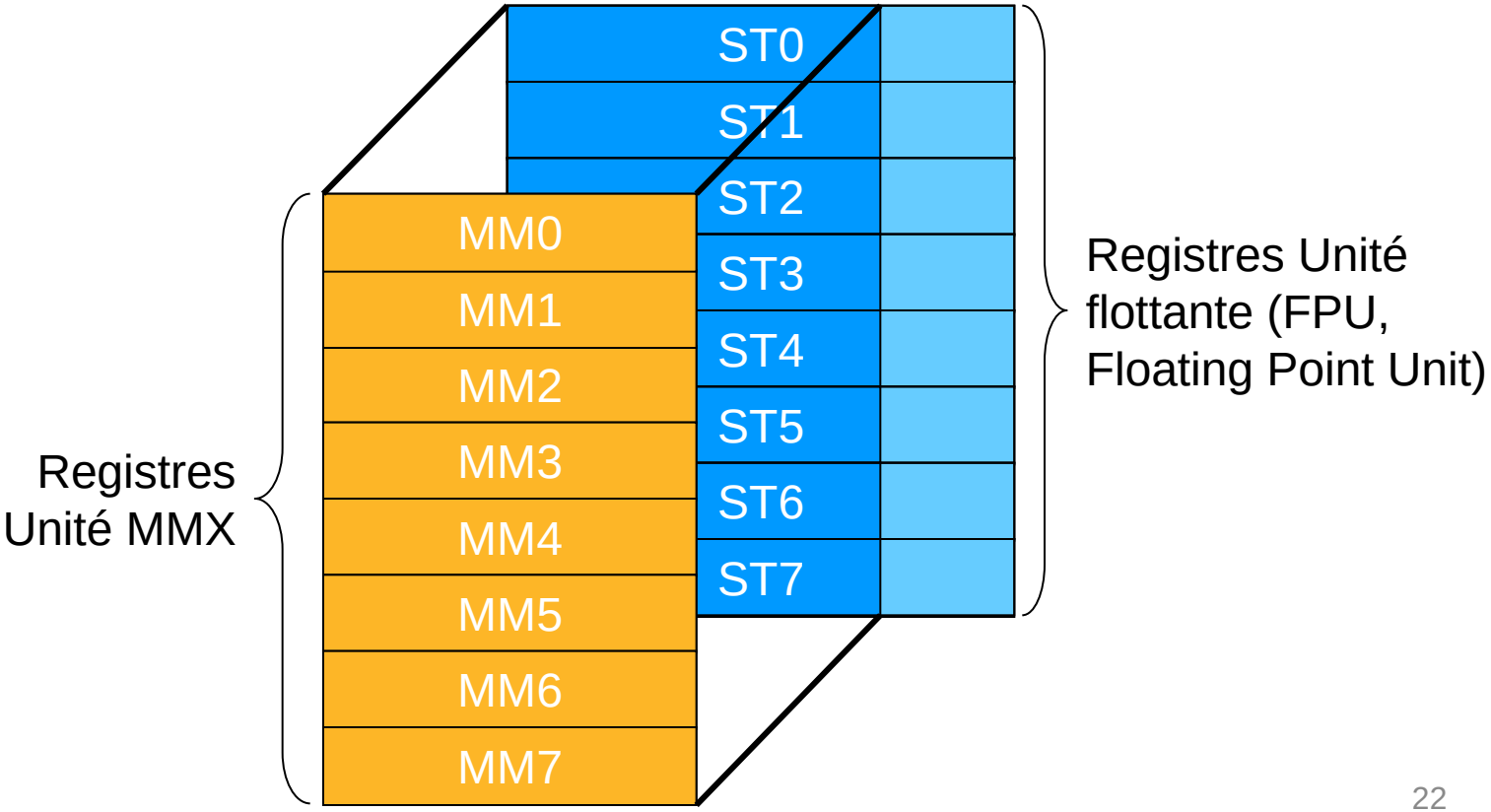
  - 8 quand on traite des octets
  - 4 quand on traite des mots
  - 2 quand on traite des double mots
  - 1 quand on traite des quadruples mots.

MMX - Registres

- Intel a ajouté un jeu de 8 registres 64 bits, utilisés par les instructions MMX pour traiter ces types de données spécifiques.
- Chacun de ces registres peut stocker n'importe quel type de données MMX. Ils sont nommés MM0 à MM7

MMX - Registres

- Ressource partagée avec l'unité de calcul flottant
- MMX et FPU ne peuvent être utilisés simultanément

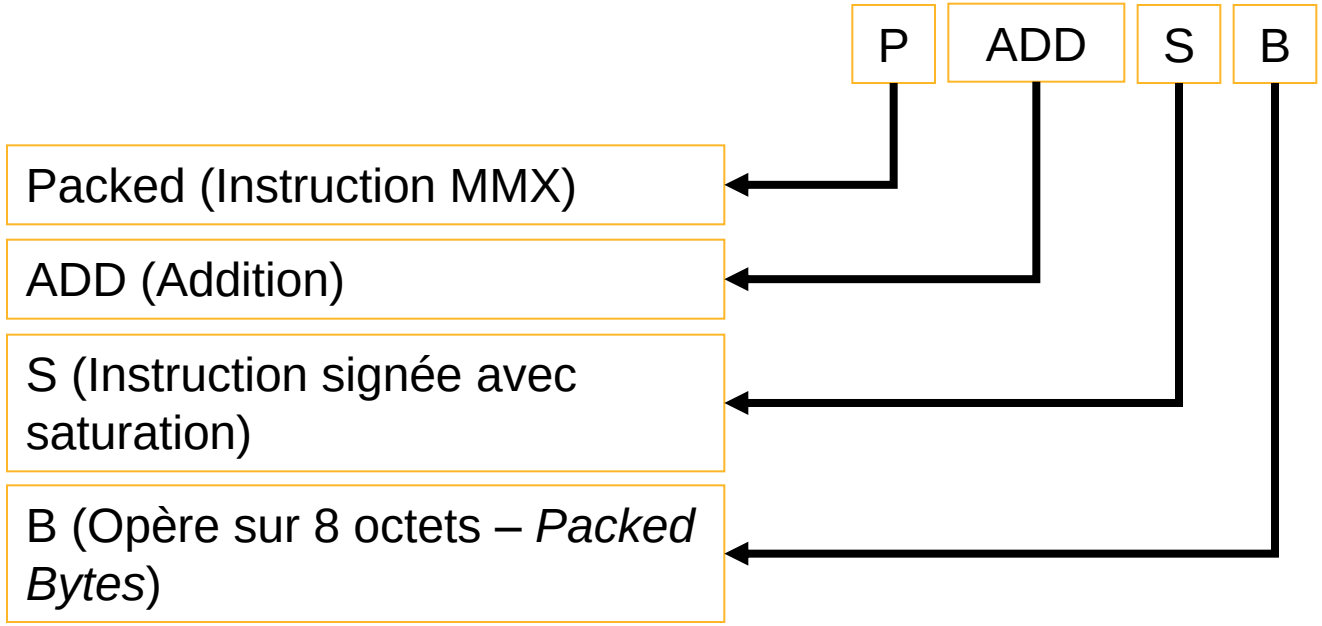


MMX - Instructions

- MMX ajoute 57 instructions dédiées au jeu d'instruction des processeurs Intel x86.
- Elles travaillent sur les types de données et registres spécifiques à MMX.
- La syntaxe générale d'une instruction MMX est formée des éléments suivants :
  - Préfixe P (Packed – Indique une instruction MMX)
  - Opération réalisée (ADD, OR, SUB…)
  - Suffixe :
    - US pour saturation signée (*Unsigned Saturation*)
    - S pour saturation non signée (*Signed Saturation*)
    - B, W, D, Q pour le type de données traité (*Packed Byte*, *Packed Word*, *Packed Double word*, *Quad word*).

MMX - Instructions

- Exemple : PADDSB



Cette instruction réalise l'addition de 8 octets non signés pris deux à deux avec saturation

MMX - Instructions

- Comme pour les instructions générales, l'assembleur attend la destination du calcul comme premier opérande, la source comme seconde

PADDSW mm0, mm1

- Les opérandes peuvent être, en fonction de l'instruction :
  - Un registre MMX, noté **mm**
  - Un registre général, noté **r**
  - Un emplacement mémoire 32 bits, noté **m32**
  - Un emplacement mémoire 64 bits, noté **m64**.
- Les opérandes mémoire sont définis par les modes d'adressage habituels :

PADDSW mm0, [EBX + ESI * 2 + 4]

- Un registre MMX ne peut jamais être utilisé comme registre d'adresse.

MMX - Instructions

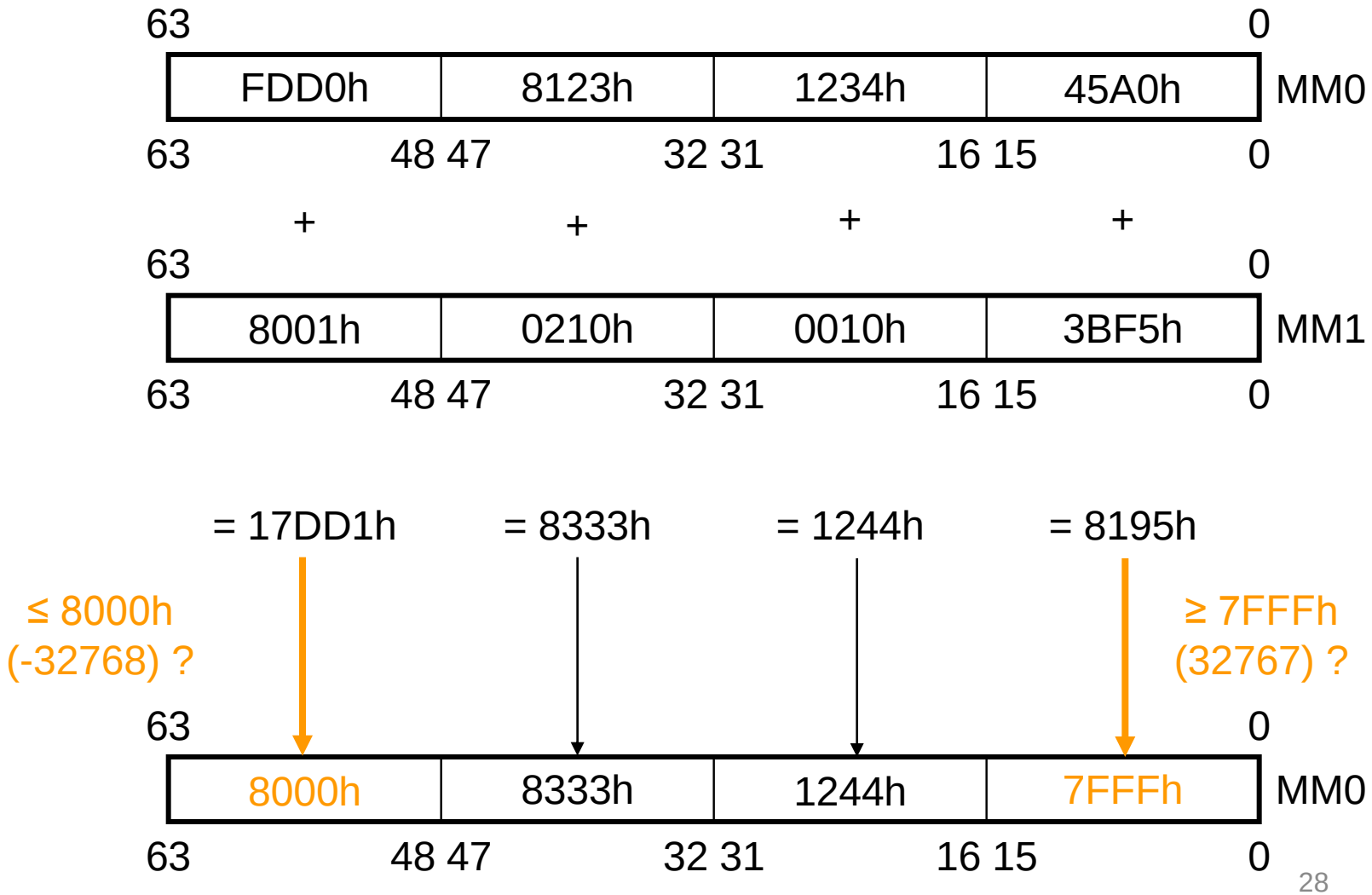
- On divise les instructions MMX en 7 familles :
  - Transferts de données
  - Conversions de types de données
  - Logiques
  - Arithmétiques
  - Comparaisons
  - Décalages
  - Contrôle de l'état MMX.
- On peut encore diviser ces instructions en instructions saturées (*saturated*) ou non saturées (*wrap-around*)

MMX - Instructions

- **Mode saturé** : limite la valeur du résultat à la valeur maximale (ou minimale) représentable.
- **Mode wrap-around** : pas de limitation (fonctionnement « habituel » d'un processeur), mais la retenue éventuelle n'est pas stockée.

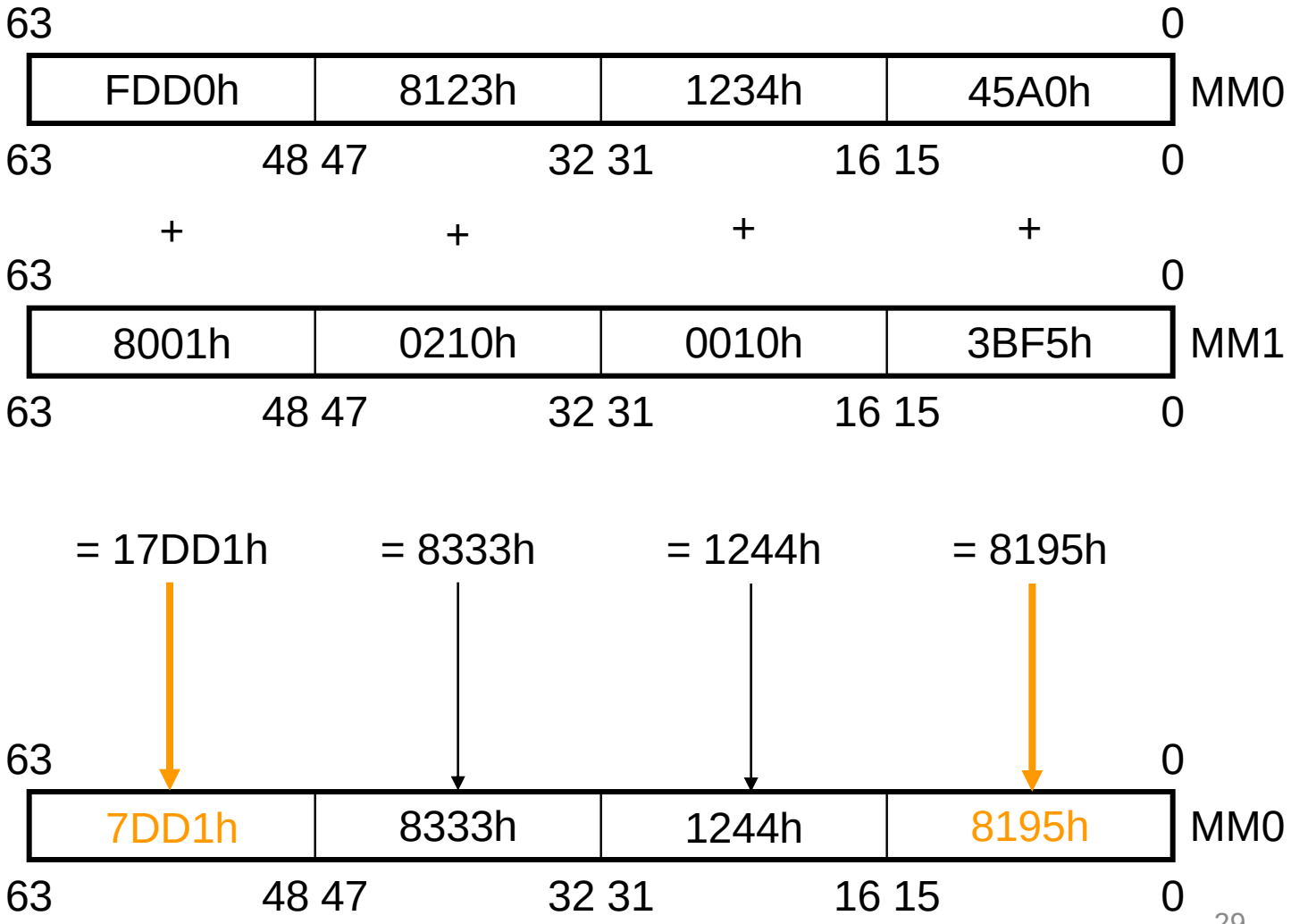
MMX – Saturation

- Exemple : PADDSW MM0, MM1



MMX – Wrap-around

- Exemple : PADDW MM0, MM1



MMX – Jeu d'instructions

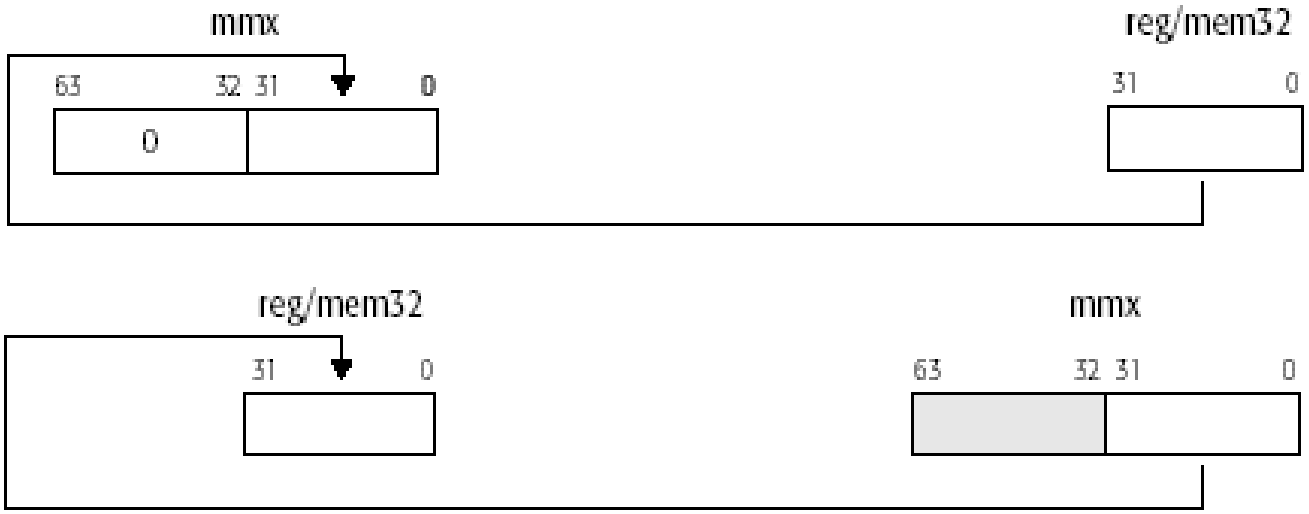
MMX – Jeu d'instructions

- Classes d'instructions
  - Transferts de données
  - Conversions de données
  - Opérations logiques
  - Opérations arithmétiques
  - Décalages
  - Comparaisons
- Aucune instruction n'affecte le registre des drapeaux (*EFLAGS*).

MMX – Transferts de données

MOVD mm, r/m32
MOVD r/m32, mm

MOVe Double word – Ces instructions copient 32 bits depuis l'opérande source dans l'opérande destination.

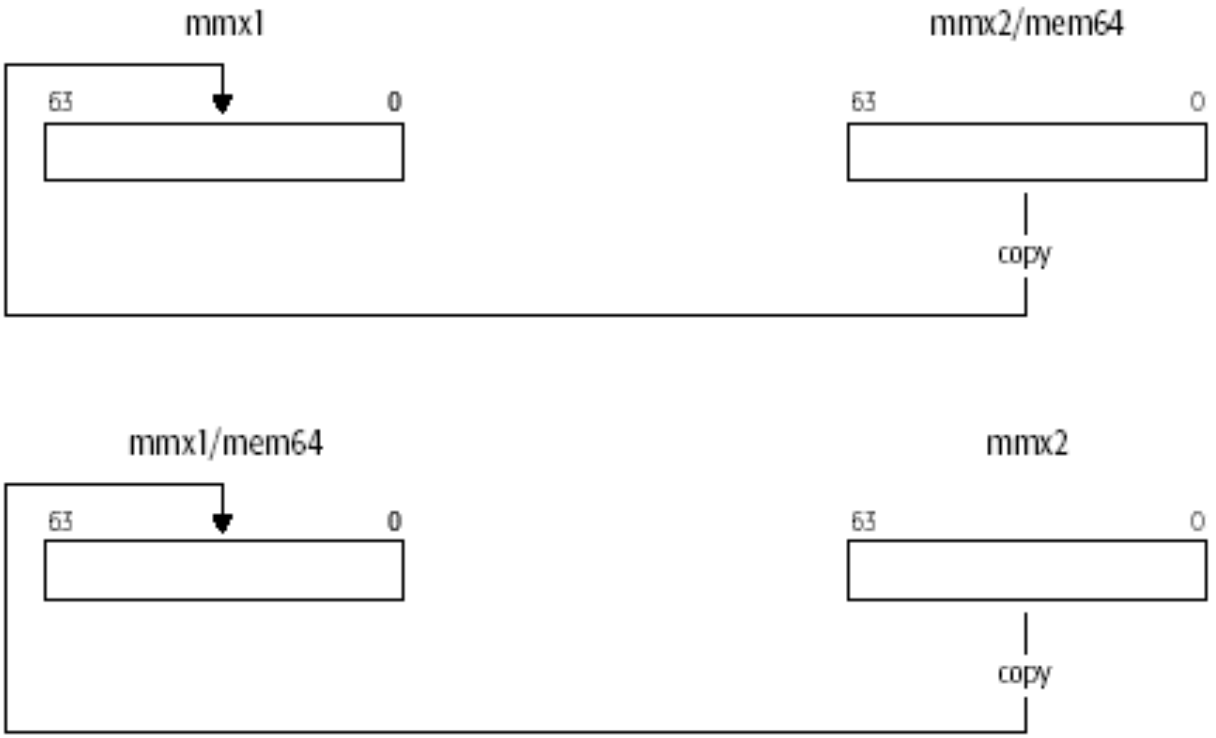


C'est la seule instruction MMX qui permet d'accéder aux registres généraux du processeur.

MMX – Transferts de données

MOVQ mm, mm/m64
MOVQ mm/m64, mm

MOVe Quad word – Ces instructions copient 64 bits depuis l'opérande source dans l'opérande destination.



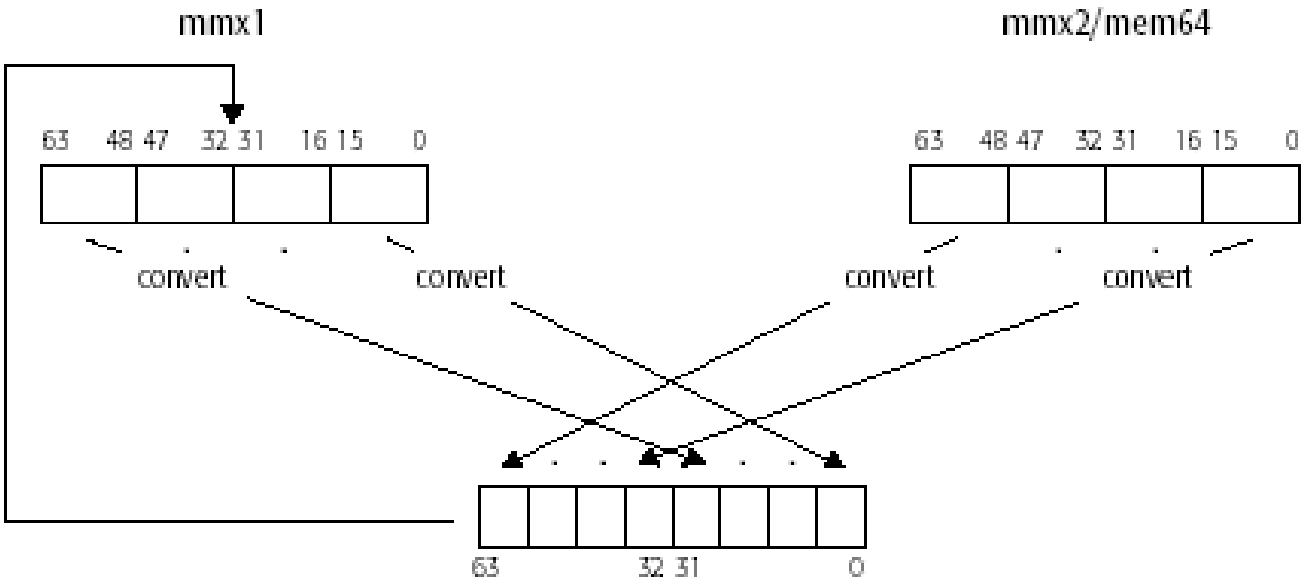
MMX – Conversions de données

- Conversion d'un type de données MMX en un autre type de données MMX. Par exemple on peut vouloir :
  - Réorganiser les éléments d'un ensemble de données pour faciliter les calculs
  - Fusionner les éléments de deux ensembles de données
  - Faire des calculs en stockant les valeurs intermédiaires sur un ensemble de données plus grandes que le résultat...
  - Simplement transformer un type en un autre type plus petit.
- Syntaxe : les deux dernières lettres indiquent le type de départ et le type d'arrivée

MMX – Conversions de données

PACKSSWB mm, mm/m64
PACKSSDW mm, mm/m64

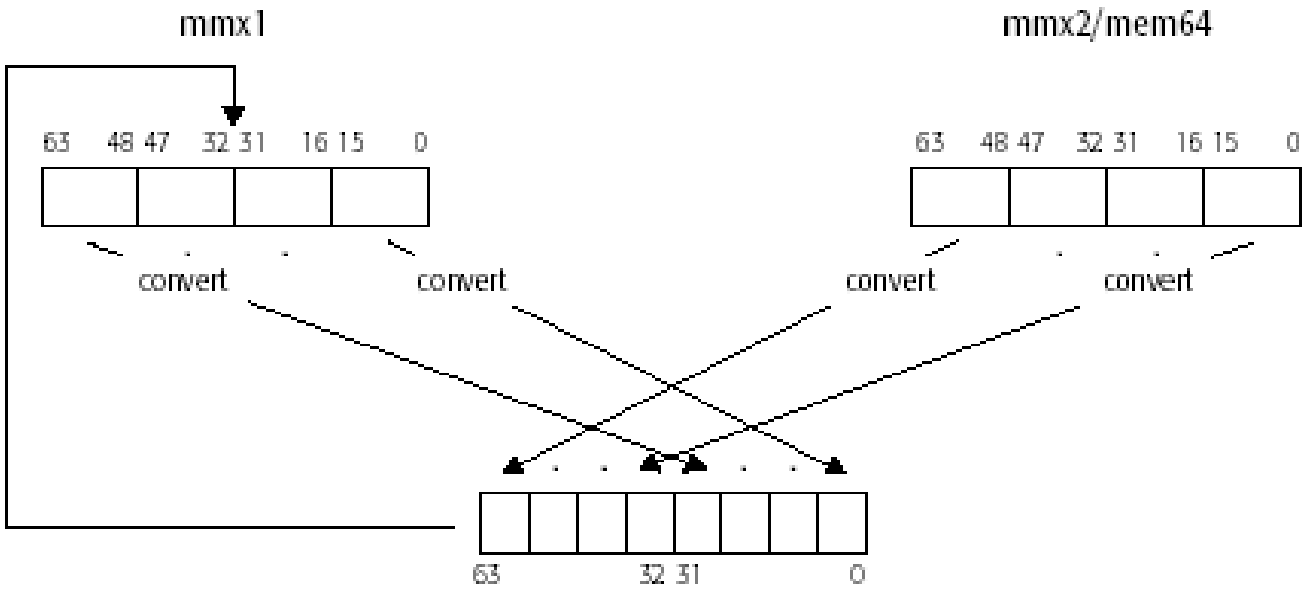
PACK Saturated Signed Word (Dword) to Byte (Word)
Convertir des données signées dans le type de taille immédiatement inférieur avec saturation.



MMX – Conversions de données

PACKUSWB mm, mm/m64
PACKUSDW mm, mm/m64

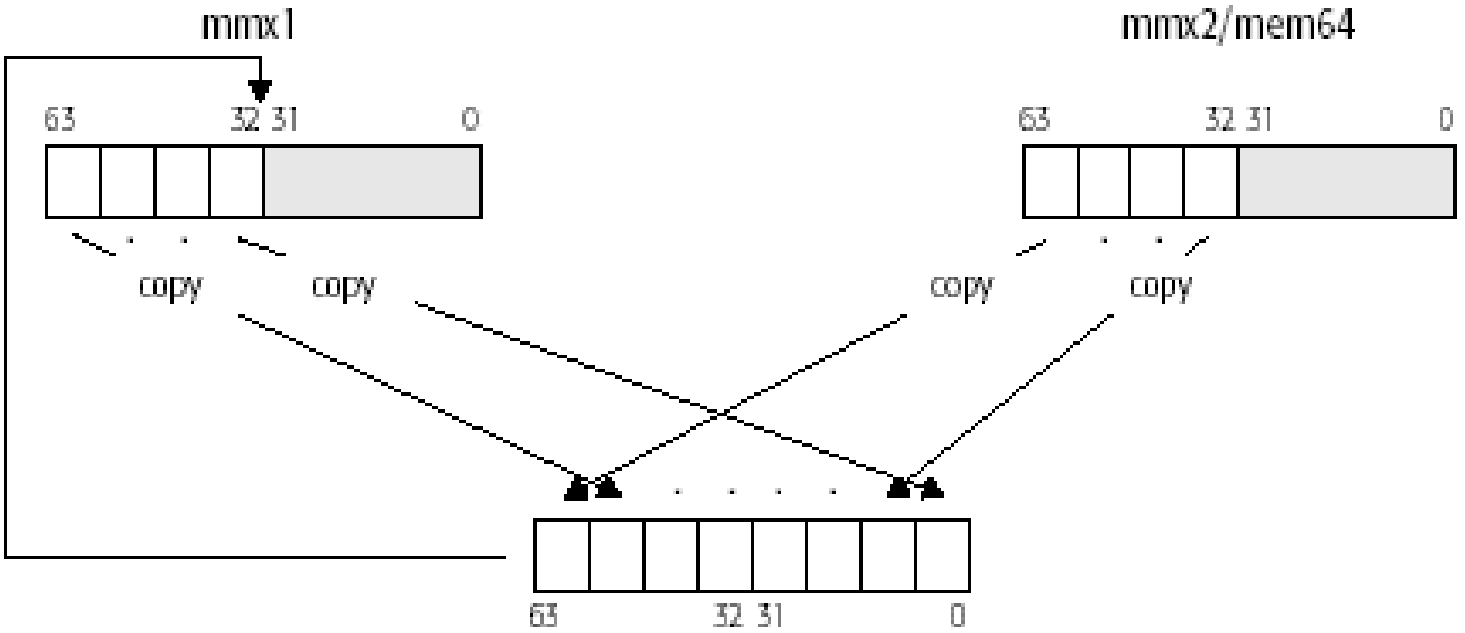
PACK Saturated Unsigned Word (Dword) to Byte (Word) Convertir des données non signées dans le type de taille immédiatement inférieur avec saturation.



MMX – Conversions de données

PUNPCKHBW mm, mm/m64
PUNPCKHWD mm, mm/m64
PUNPCKHDQ mm, mm/m64

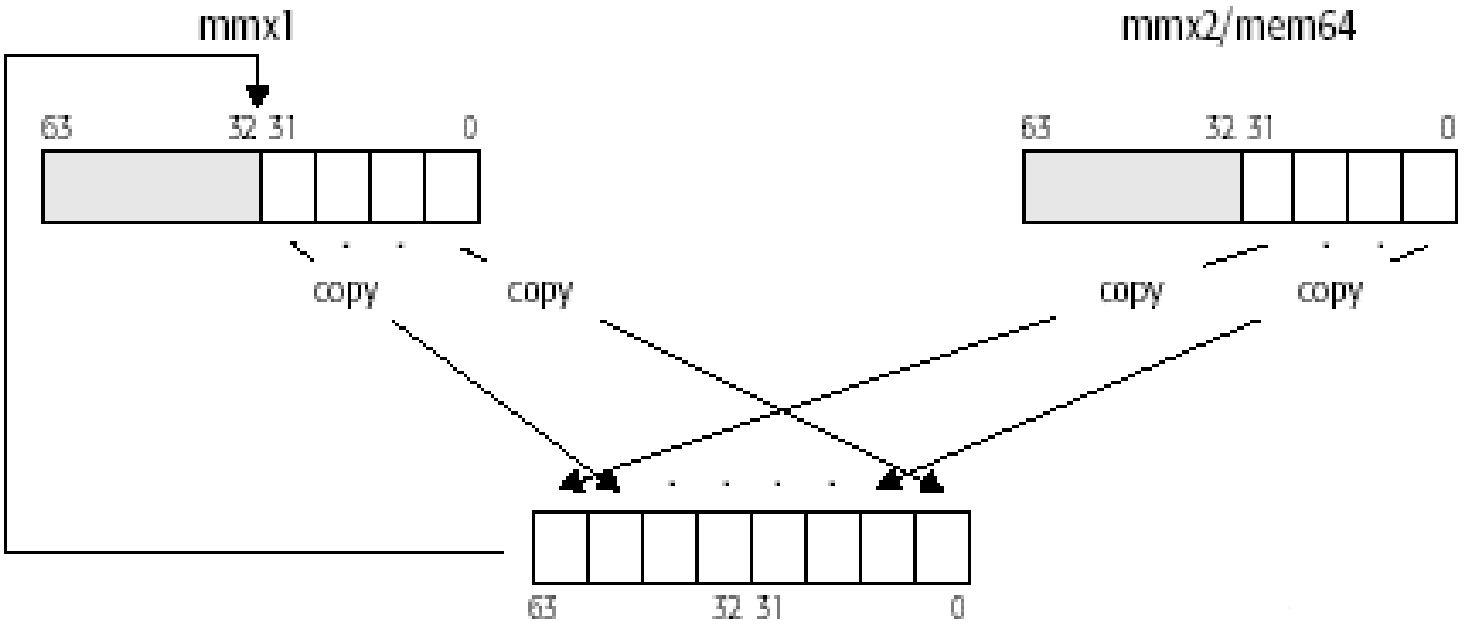
UNPaCK High Packed Data – Copie en les intercalant les données hautes d'un type de données MMX.



MMX – Conversions de données

PUNPCKLBW mm, mm/m64
PUNPCKLWD mm, mm/m64
PUNPCKLDQ mm, mm/m64

UNPaCK Low Packed Data – Copie en les intercalant les données basses d'un type de données MMX.



MMX – Opérations logiques

- Ces instructions permettent de réaliser les opérations logiques suivantes :
  - ET (AND)
  - OU (OR)
  - OU exclusif (XOR)
  - ET NON (AND NOT)

MMX – Opérations logiques

PAND mm, mm/m64
PANDN mm, mm/m64
POR mm, mm/m64
PXOR mm, mm/m64

Effectue une opération logique.

En particulier :

PANDN MM0, MM1 ➔ MM0 ← MM0 ET NON MM1.

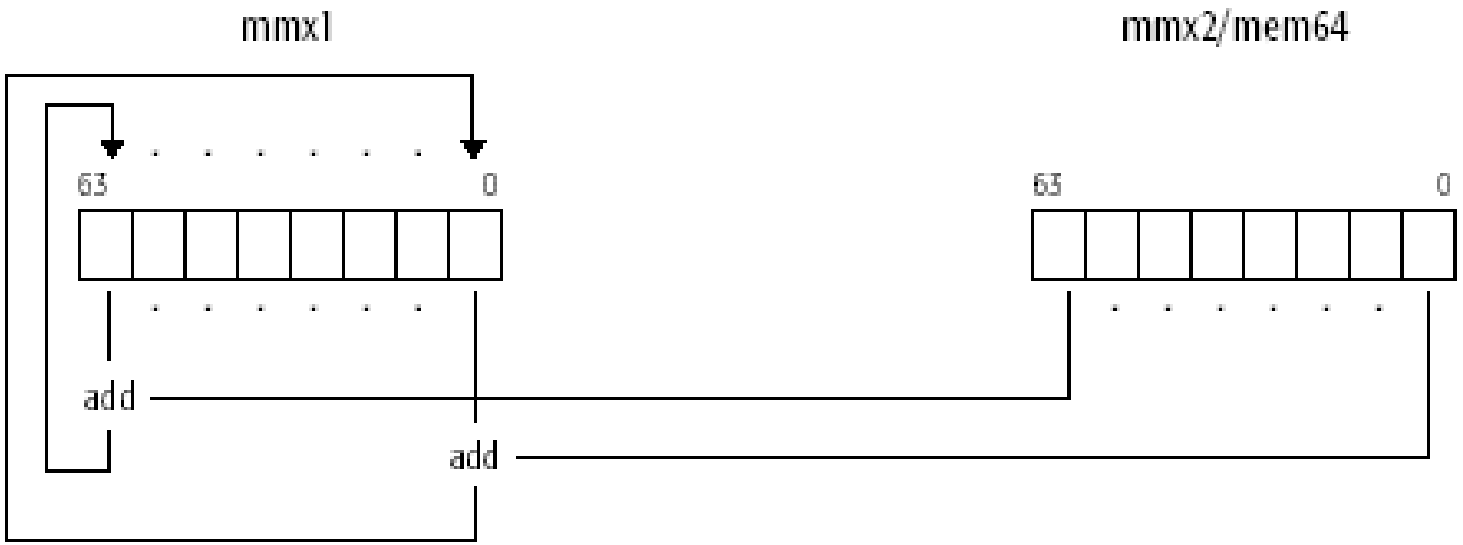
MMX – Opérations arithmétiques

- Ces instructions permettent de faire des calculs sur des types de données packées :
  - Additions
  - Soustractions
  - Multiplications
  - Multiplication-accumulations.
- Elles fonctionnent soit en mode saturé, soit en mode wrap-around, sur des entiers signés ou non signés.

MMX – Opérations arithmétiques

PADDB mm,mm/m64
PADDW mm,mm/m64
PADDD mm,mm/m64
PADDQ mm, mm/m64

Ajoute deux à deux les données d'un type MMX signé.

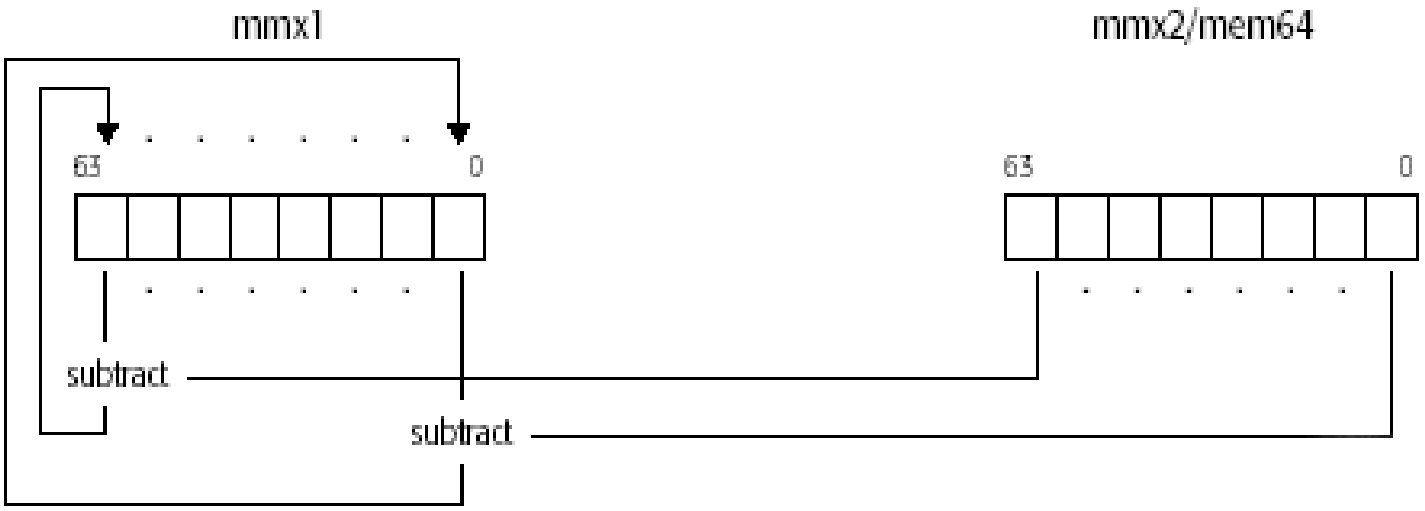


Les données peuvent être saturées : PADDSB, PADDSW
Les données peuvent être non signées saturées : PADDUSB, PADDUSW.

MMX – Opérations arithmétiques

PSUBB mm,mm/m64
PSUBW mm,mm/m64
PSUBD mm,mm/m64

Soustrait deux à deux les données d'un type MMX signé avec saturation.



Les données peuvent être saturées : PSUBSB, PSUBSW.
Les données peuvent être non signées saturées : PSUBUSB, PSUBUSW.

MMX – Opérations arithmétiques

- Exemple : on veut calculer la différence absolue x de deux vecteurs de mots non signés u et v :

$$x = | u - v |.$$

- L'algorithme est simple : pour chaque composante u_i et v_i de u et v, on a :

if ($u_i > v_i$)
$x_i = (u_i - v_i)$;
else
$x_i = (v_i - u_i)$;

- Il n'y a pas d'instructions conditionnelles en *MMX*. On va utiliser le calcul avec saturation pour contourner le problème.

MMX – Opérations arithmétiques

- Si on calcule les deux différences ($u_i - v_i$) et ($v_i - u_i$) **avec saturation**, l'une sera nécessairement nulle ➔ celle qui aurait été négative.
- On calcule un OU entre ces deux résultats : seule la différence positive reste.
- On suppose que MM0 et MM1 contiennent les deux vecteurs.

MMX – Opérations arithmétiques

État initial

0001h	0002h	0400h	0001h	MM0
0002h	0001h	0550h	0000h	MM1

```
MOVQ    MM2, MM0        ; MM2 ← MM0
PSUBUSW MM0, MM1        ; MM0 ← MM0 – MM1
PSUBUSW MM1, MM2        ; MM1 ← MM1 – MM2
POR MM0, MM1            ; MM0 ← MM0 OU MM1
```

MMX – Opérations arithmétiques

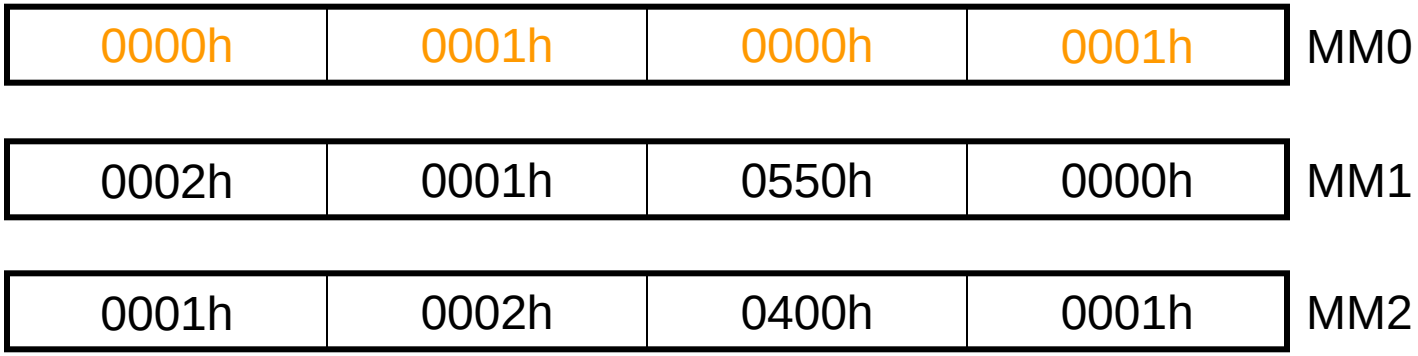
MOVQ MM2, MM0

0001h	0002h	0400h	0001h	MM0
0002h	0001h	0550h	0000h	MM1
0001h	0002h	0400h	0001h	MM2

```
MOVQ    MM2, MM0        ; MM2 ← MM0
PSUBUSW MM0, MM1        ; MM0 ← MM0 – MM1
PSUBUSW MM1, MM2        ; MM1 ← MM1 – MM2
POR MM0, MM1            ; MM0 ← MM0 OU MM1
```

MMX – Opérations arithmétiques

PSUBUSW MM0, MM1




```
MOVQ    MM2, MM0        ; MM2 ← MM0
PSUBUSW MM0, MM1        ; MM0 ← MM0 – MM1
PSUBUSW MM1, MM2        ; MM1 ← MM1 – MM2
POR MM0, MM1            ; MM0 ← MM0 OU MM1
```

MMX – Opérations arithmétiques

PSUBUSW MM1, MM2


```
MOVQ    MM2, MM0        ; MM2 ← MM0
PSUBUSW MM0, MM1        ; MM0 ← MM0 – MM1
PSUBUSW MM1, MM2        ; MM1 ← MM1 – MM2
POR MM0, MM1            ; MM0 ← MM0 OU MM1
```

MMX – Opérations arithmétiques

POR MM0, MM1

0001h	0001h	0150h	0001h	MM0
0001h	0000h	0150h	0000h	MM1
0001h	0002h	0400h	0001h	MM2

```
MOVQ    MM2, MM0        ; MM2 ← MM0
PSUBUSW MM0, MM1        ; MM0 ← MM0 – MM1
PSUBUSW MM1, MM2        ; MM1 ← MM1 – MM2
POR MM0, MM1            ; MM0 ← MM0 OU MM1
```

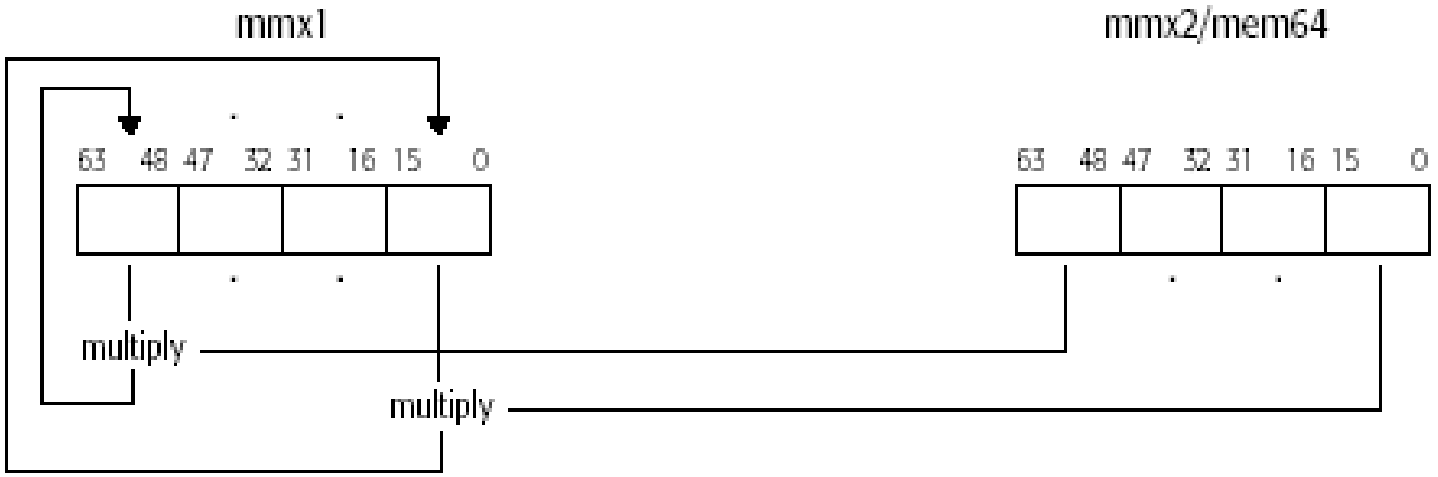
MMX – Opérations arithmétiques

PMULHW mm, mm/m64
PMULLW mm, mm/m64

Multiplie deux à deux des mots de 16 bits packés

PMULH stocke les 16 bits de poids fort des résultats

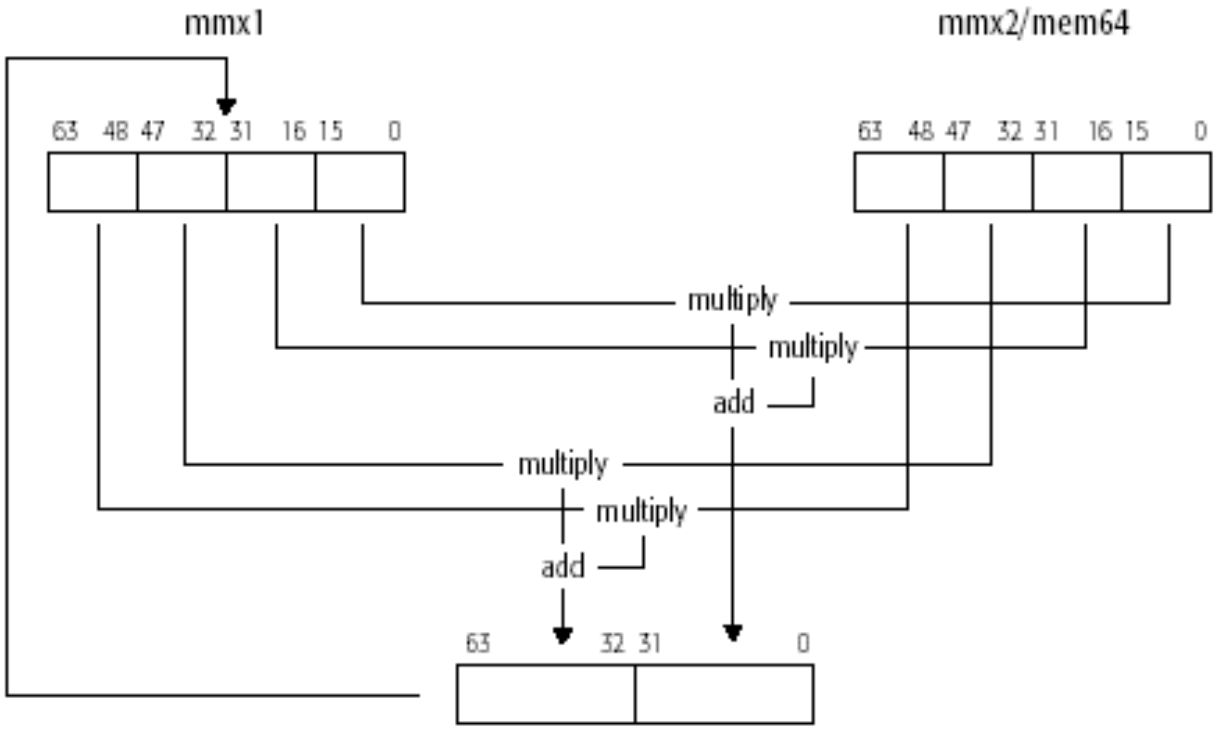
PMULL stocke les 16 bits de poids faible des résultats



MMX – Opérations arithmétiques

PMADDWD mm, mm/m64

Multiplie deux à deux des données 16 bits, puis additionne les produits 32 bits deux à deux.



MMX – Opérations arithmétiques

- Exemple : multiplication complexe
  - On suppose qu'on veut calculer le produit d'un nombre complexe $a + i\ b$ avec une constante complexe $c_{Re} + i\ c_{Im}$. Les coefficients sont des entiers 16 bits.
  - La partie réelle du résultat est :

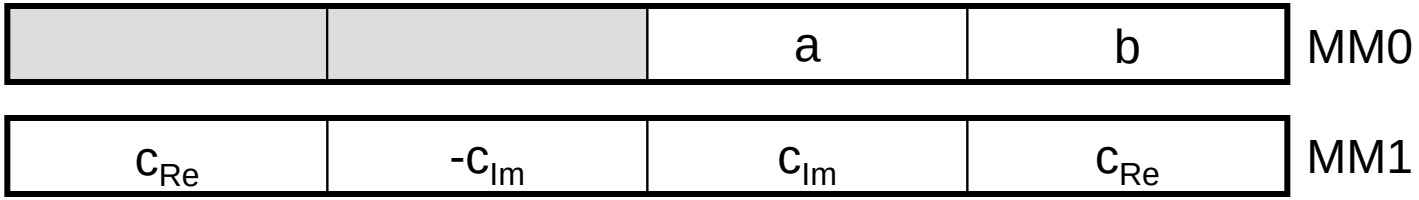
    $a \cdot c_{Re} - b \cdot c_{Im}$
  - La partie imaginaire du résultat est :

    $a \cdot c_{Im} + b \cdot c_{Re}$
  - On suppose que la constante complexe est donnée sous la forme suivante, sur 64 bits :

c_{Re}	$-c_{Im}$	c_{Im}	c_{Re}

MMX – Opérations arithmétiques

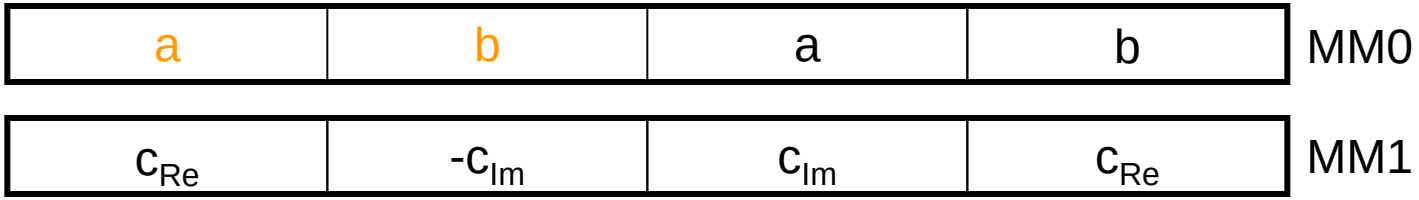
État initial




```
PUNPCKLDQ MM0, MM0  ; Convertir sous la forme [Re Im Re Im]
PMADDWD MM0, MM1    ; Calculer le produit complexe
```

MMX – Opérations arithmétiques

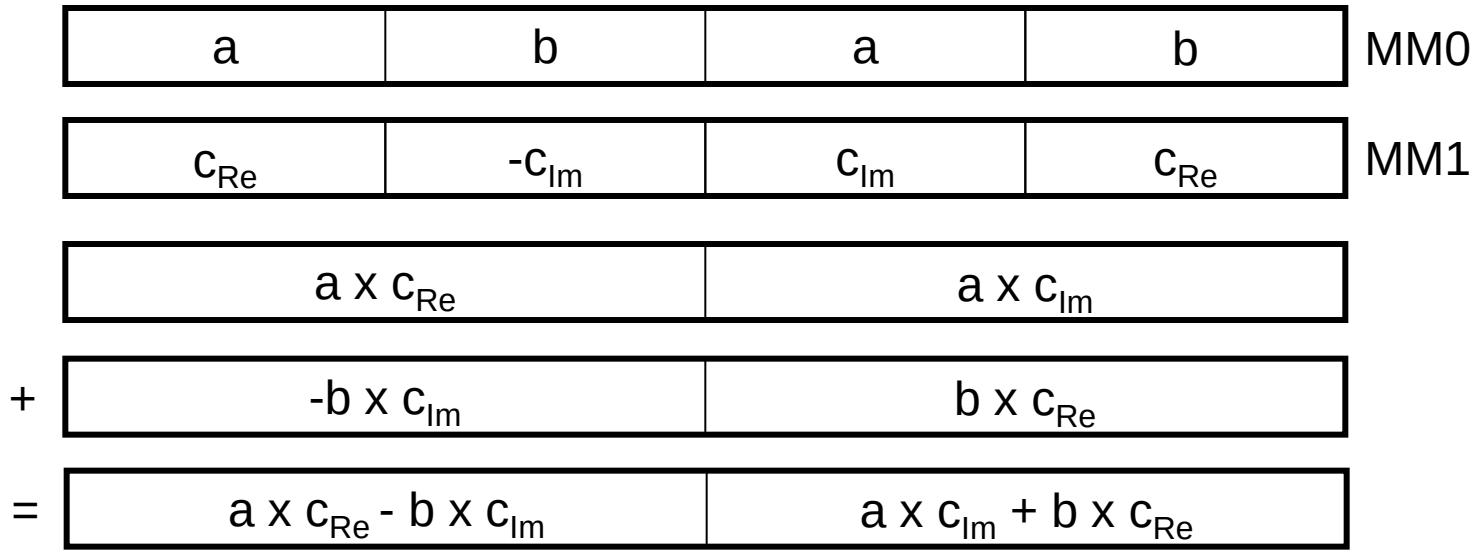
PUNPCKLDQ MM0, MM0




```
PUNPCKLDQ MM0, MM0  ; Convertir sous la forme [Re Im Re Im]
PMADDWD MM0, MM1    ; Calculer le produit complexe
```

MMX – Opérations arithmétiques

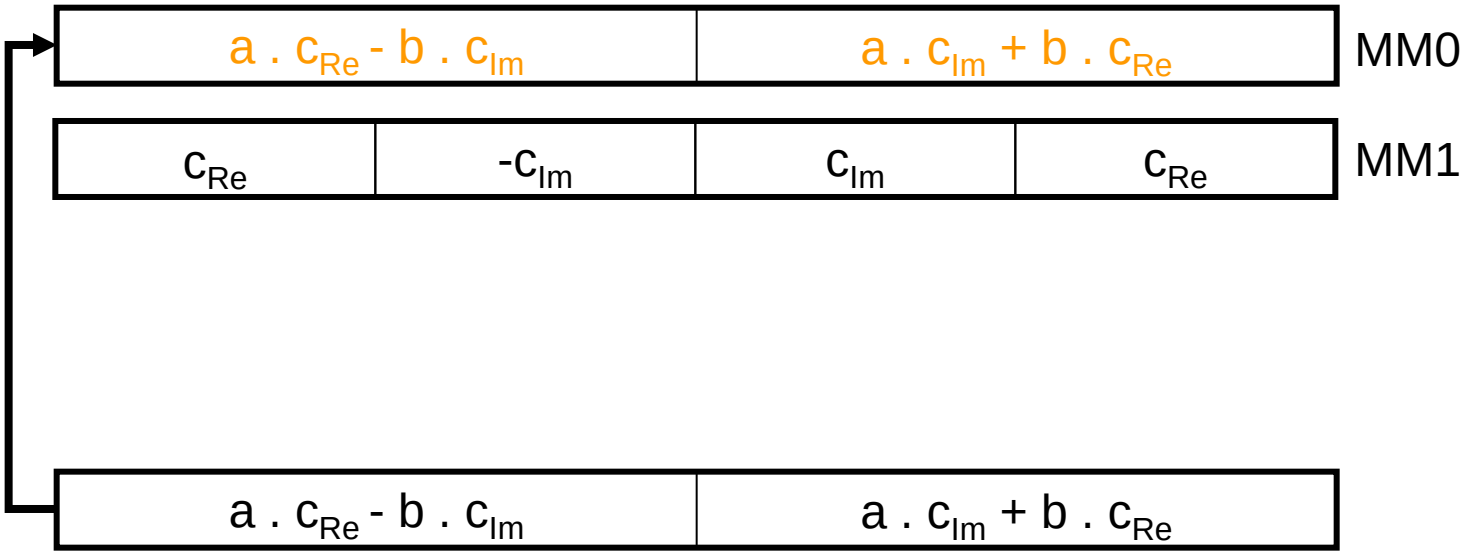
PMADDWD MM0, MM1- Etape 1




```
PUNPCKLDQ MM0, MM0  ; Convertir sous la forme [Re Im Re Im]
PMADDWD MM0, MM1    ; Calculer le produit complexe
```

MMX – Opérations arithmétiques

PMADDWD MM0, MM1- Etape 2




```
PUNPCKLDQ MM0, MM0  ; Convertir sous la forme [Re Im Re Im]
PMADDWD MM0, MM1    ; Calculer le produit complexe
```

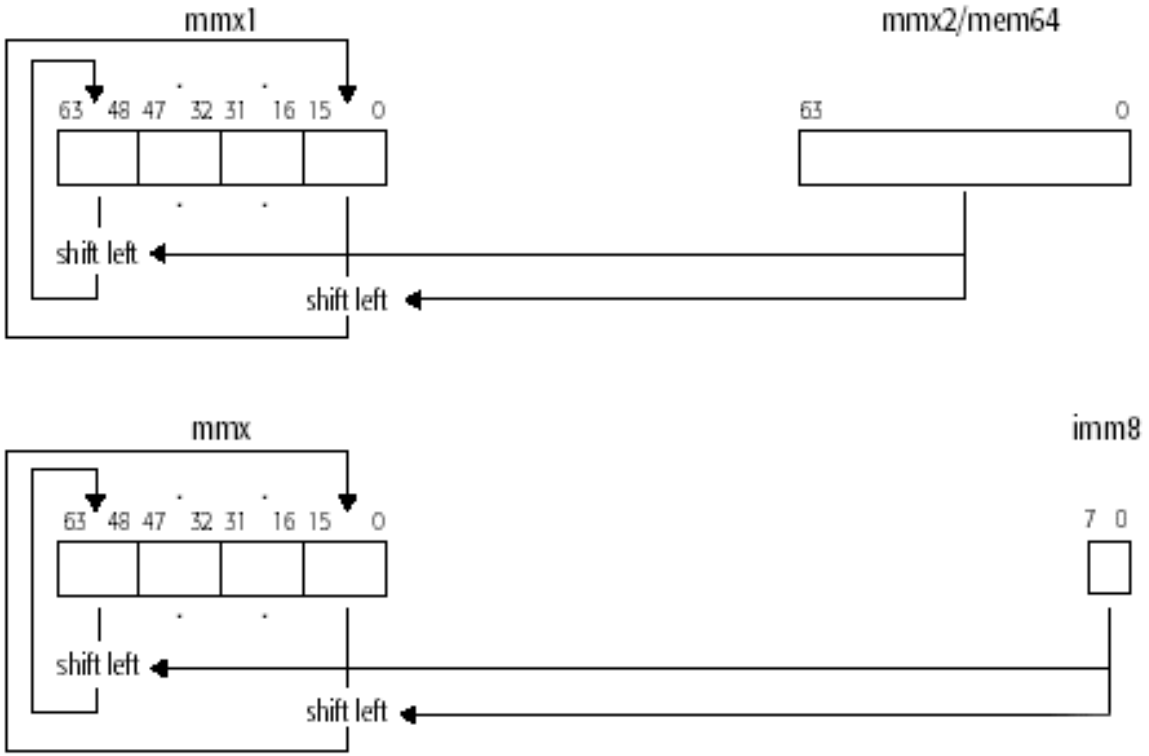
MMX – Décalages

- Ces instructions permettent de décaler indépendamment les données d'un ensemble packé, soit d'un nombre de bits contenu dans un registre MMX 64 bits, soit d'un nombre de bits précisé en valeur immédiate. Les décalages peuvent être :
  - A gauche,
  - A droite sans extension de signe (on fait rentrer des 0)
  - A droite avec extension de signe (décalage arithmétique).

MMX – Décalages

PSLLW mm, mm/m64/imm8
PSLLD mm, mm/m64/imm8
PSLLQ mm, mm/m64/imm8

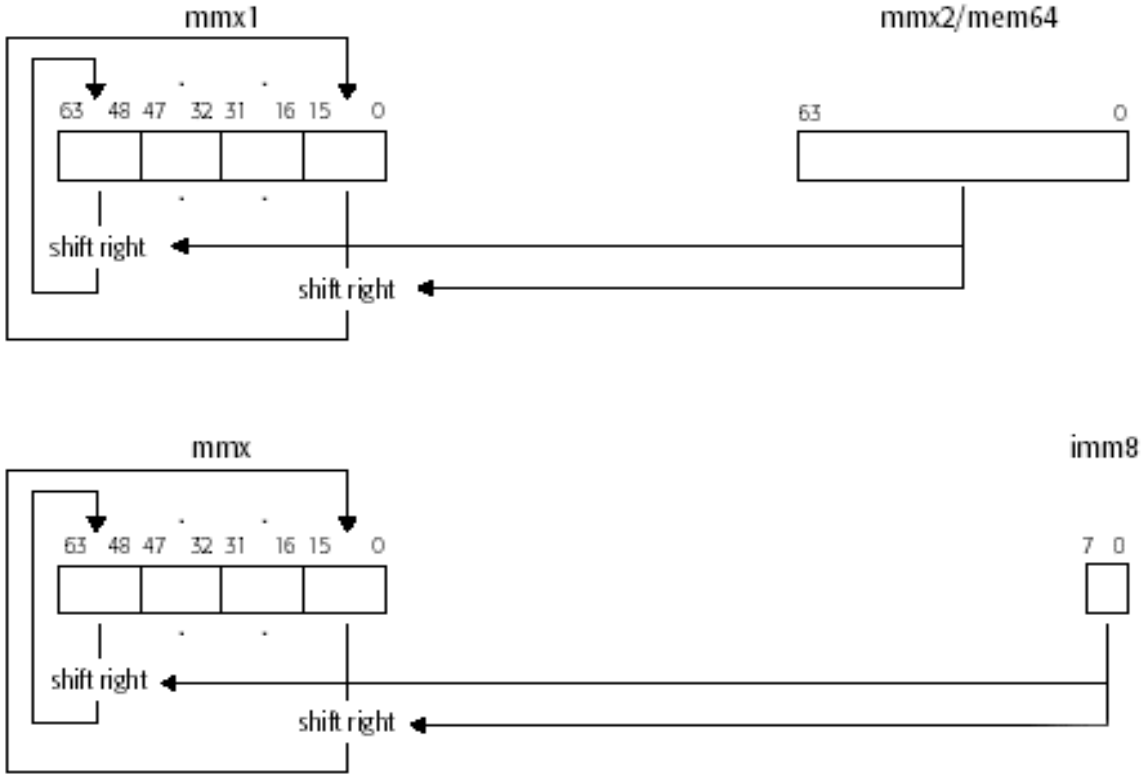
Décaler à gauche chaque élément de données.



MMX – Décalages

PSRLW mm, mm/m64/imm8
PSRLD mm, mm/m64/imm8
PSRLQ mm, mm/m64/imm8

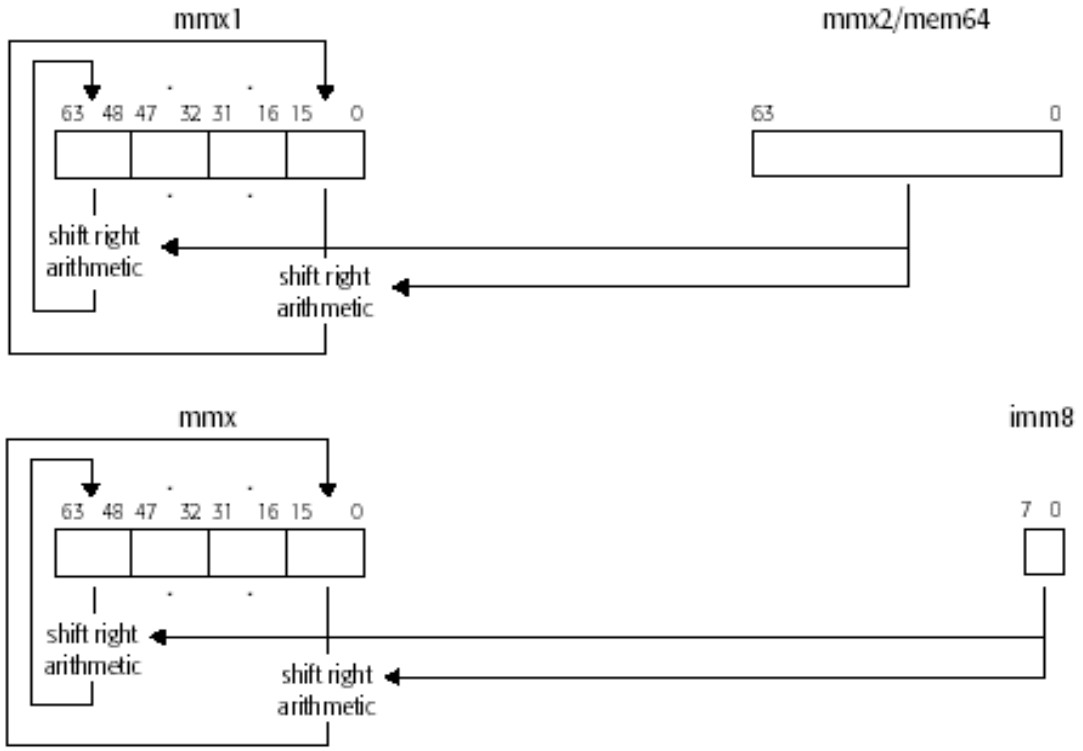
Décaler à droite chaque élément de données.



MMX – Décalages

PSRAW mm, mm/m64/imm8
PSRAD mm, mm/m64/imm8

Décaler à droite chaque élément de données, étendre par le bit de signe (décalage arithmétique).



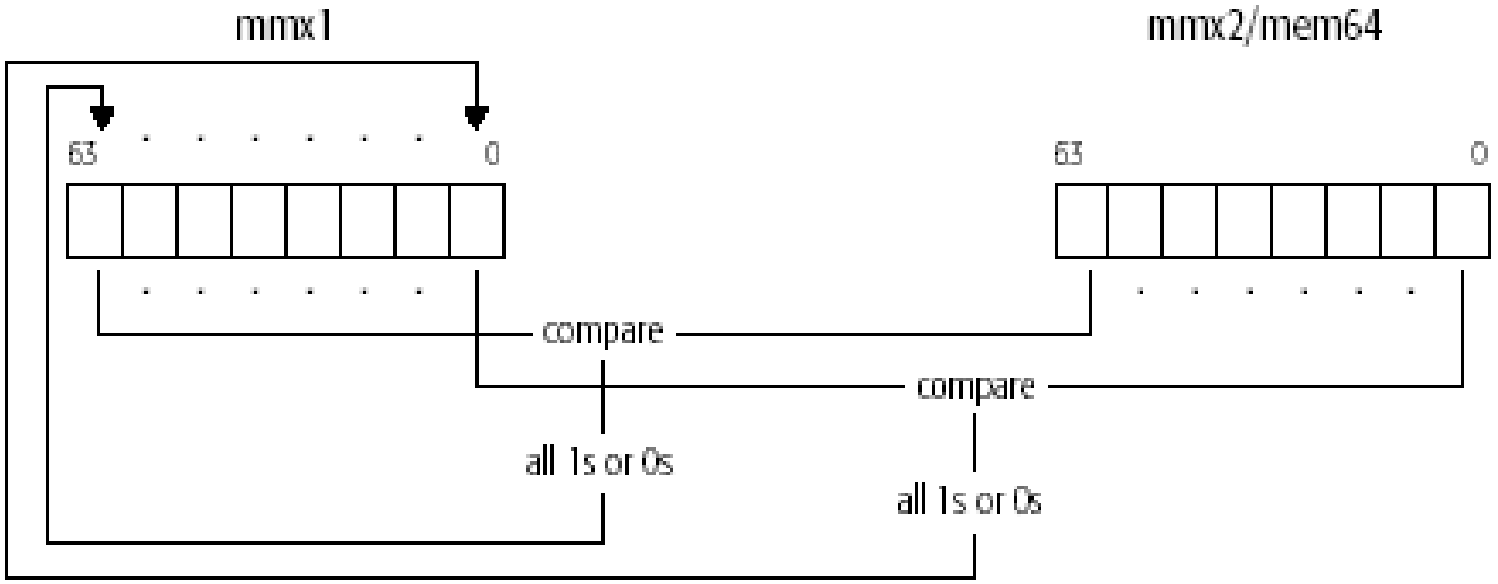
MMX – Comparaisons

- Ces instructions permettent de comparer deux à deux les données d'un ensemble packé.
  - Si la comparaison est vraie, alors tous les bits du résultat sont mis à 1.
  - Si elle est fausse, alors tous les bits du résultat sont mis à 0.
- On peut alors utiliser des opérations logiques pour sélectionner des données dans un registre MMX, afin de réaliser des déplacements de données conditionnels.

MMX – Comparaisons

PCMPEQB mm, mm/m64
PCMPEQW mm, mm/m64
PCMPEQD mm, mm/m64

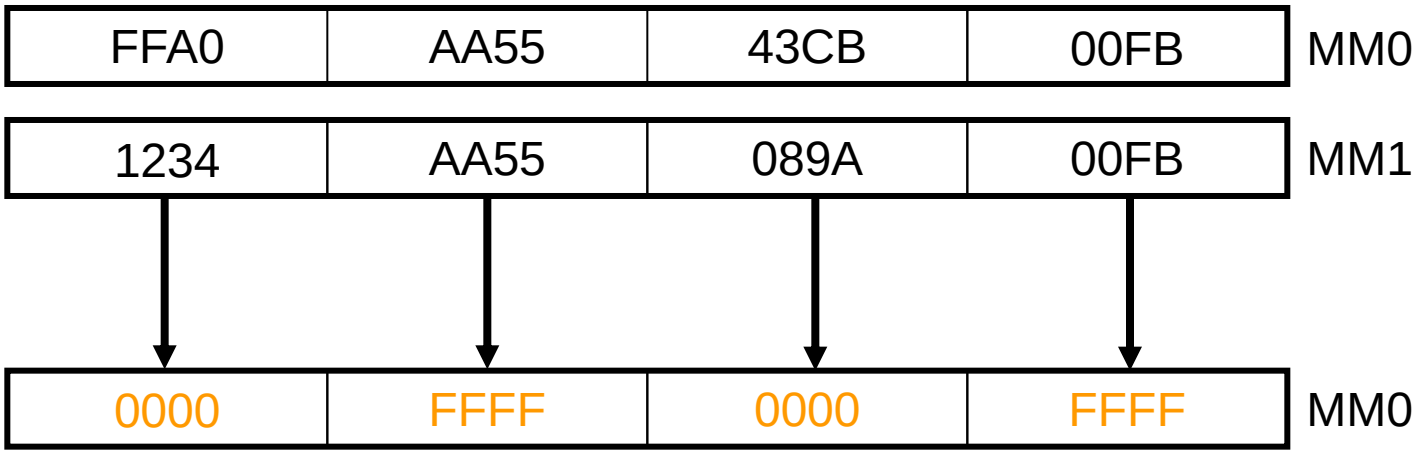
Comparer chaque élément de la source et de la destination. S'ils sont égaux, l'élément correspondant de la destination est mis à un, sinon à 0.



Les instructions **PCMPGT(B,W,D)** comparent deux à deux des données signées, et testent si la source est supérieure à la destination.

MMX – Comparaisons

PCMPEQW MM0, MM1



MMX – Contrôle d'état

- On a vu que le coprocesseur arithmétique et l'unité MMX étaient mutuellement exclusives, puisque l'état MMX et coprocesseur est stocké dans les mêmes registres.
- Il faut donc une instruction qui indique au processeur que l'utilisation de l'unité MMX est terminée, et qu'on souhaite à nouveau utiliser le coprocesseur : EMMS.
- Il suffit donc de terminer un algorithme MMX par l'instruction EMMS.

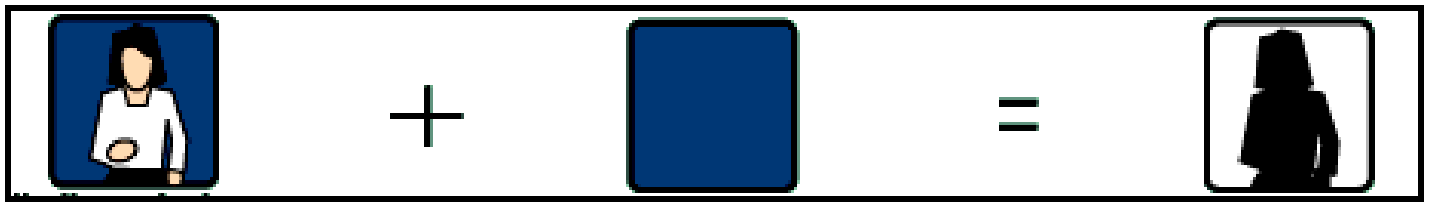
Un exemple : le chroma-key

MMX – Chroma Key

- Le principe du *Chroma Key* (incrustation vidéo) est utilisé pour permettre d'incruster sur un fond variable un personnage.
- Pour cela, on filme le personnage devant un fond de couleur unie (en général bleu ou vert), puis on mélange cette image avec l'image de fond choisie.

MMX – Chroma Key

- Première étape : détourer le personnage



- Il faut créer un masque dont les pixels valent 0 si c'est un pixel du personnage, 1 sinon.
  ➔ PCMPEQB

MMX – Chroma Key

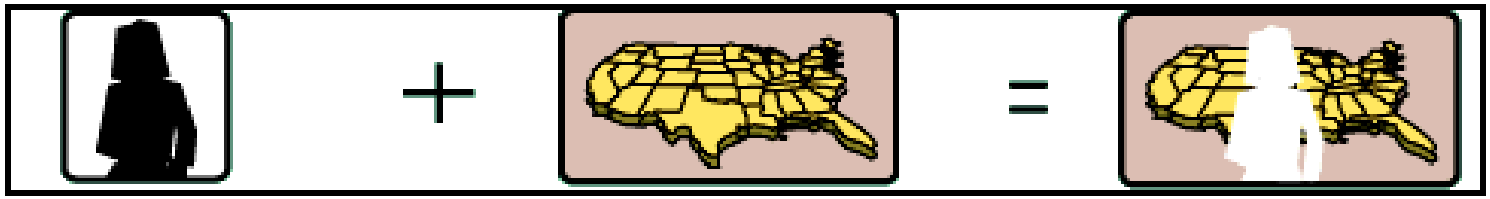
- Seconde étape : éliminer le fond bleu

On prend le masque, on l'inverse et on fait un ET avec l'image de départ pour donner la première image intermédiaire

➔ PANDNB

MMX – Chroma Key

- Troisième étape : construction du fond final

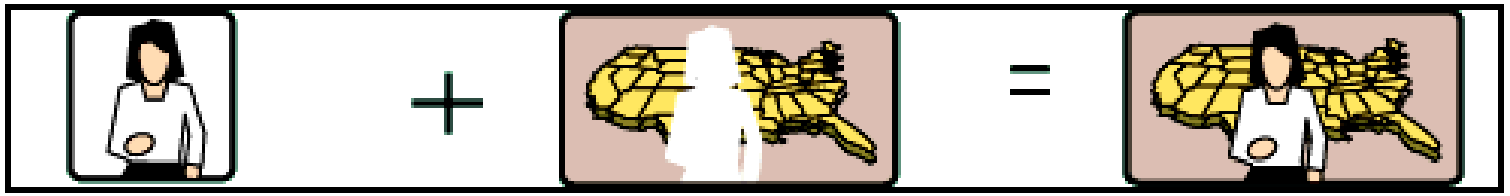


On élimine le masque du fond en utilisant le masque pour obtenir la seconde image intermédiaire

➔ PAND

MMX – Chroma Key

- Quatrième étape : combiner les deux images intermédiaires



Il suffit de faire un OU entre les deux images pour obtenir l'image finale

➔ POR

MMX - Conclusion

- MMX apporte en général une accélération non négligeable des algorithmes.
- Cependant, il souffre de limitations :
  - Impossibilité de réarranger finement les données à l'intérieur d'un ensemble packé
  - Et surtout, MMX ne traite que des entiers, éventuellement réels à virgule fixe, ce qui limite les possibilités en terme de calculs

  ➔ Introduction des extensions SSE (Intel) et 3DNow! (AMD) pour supprimer ces inconvénients.