Architecture des systèmes à processeur

Prof. Géraldine Conti

Basé sur les cours des Profs. Sanchez, Starkier, Mosqueron et Dassatti

Cortex-A8 - parallélisme

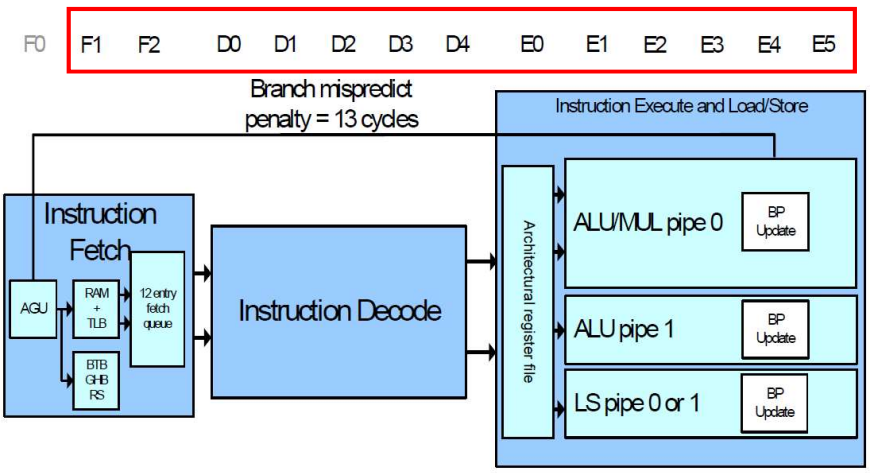
Introduction

- Utilisé dans les **téléphones portables, les consoles de jeux**, …
- Premier processeur à incorporer toutes les nouvelles technologies disponibles dans l'architecture ARMv7
  - **NEON** : pour le processing du signal et média
  - Autres technologies pour l'accélération temps-réel de compilateurs, pour la sécurité,…
- Faible consommation
- Haut degré de performance:
  - **Pipeline revisité, cache L2 intégrée**
- **Thumb-2** pour les instructions (+130 instructions par rapport à Thumb)

Performances

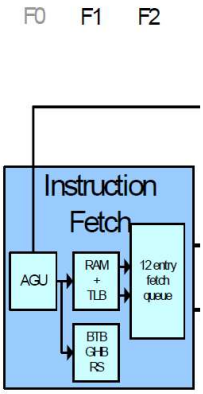
- Fréquence max : 1GHz
- Consommation : 300mW
- Jusqu'à 2000 MIPS/MHz
- IPC (nb instructions par cycles) : 0.9
- 95 % de prédictions de branchement correctes

Pipeline



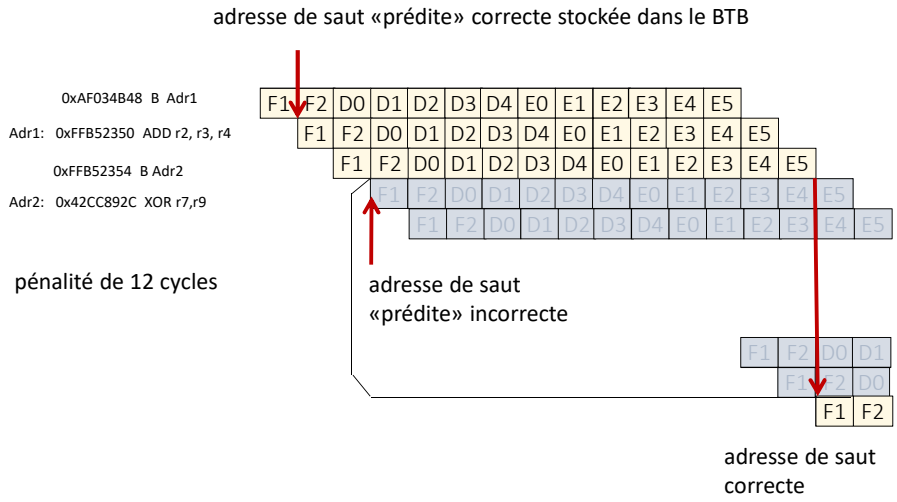
- Architecture « **dual issue** » : Traitement de 2 instructions simultanées ou d'une instruction et d'un load/store
- **In-order**
- Pipeline à **13 niveaux**
- **superscalaire**

FETCH

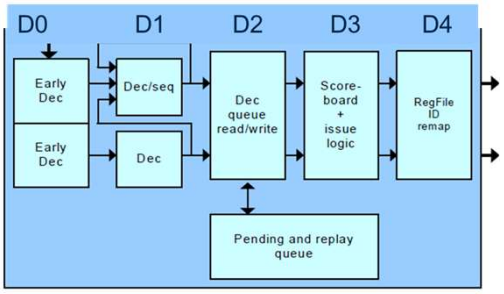


- **F0:** Génération des adresses d'instruction dans **l'Address Generation Unit (AGU).** L'adresse est soit l'adresse suivante (incrémentation du PC), soit l'adresse de saut prédite (venant du BTB)
- **F1:** Fetch de 4 instructions par cycle dans la mémoire cache d'instruction. **Mécanisme de prédiction de branchement** (*basé sur l'historique*)
  - **Branch Target Buffet (BTB)** - 512 entrées
    - Indique si oui/non l'adresse courante va retourner une instruction Branch et l'adresse cible
      - Si oui : l'adresse de saut est prédite et accès au GHB
  - **Global History Buffer (GHB)** - 4096 entrées
    - contient l'historique des branchements
    - la mise à jour se fait en E4
  - Le **return stack (RS)** contient les adresses de retour dans une pile
- **F2**: queue de 12 instructions en attente de décodage

Pénalité de branchement

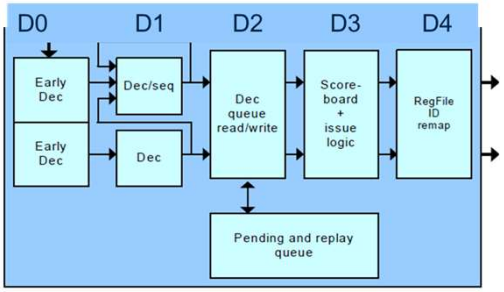


DECODE (1)



- **D0-D1:**
  - décodage « en avance » des instructions 32 bits ou 2 x 16 bits :
  - Extraction de l'opcode, identification des opérandes
  - Séparation des instructions d'opérations multiples en plusieurs instructions simples
  - Chargement des instructions dans la queue (D2)

DECODE (2)



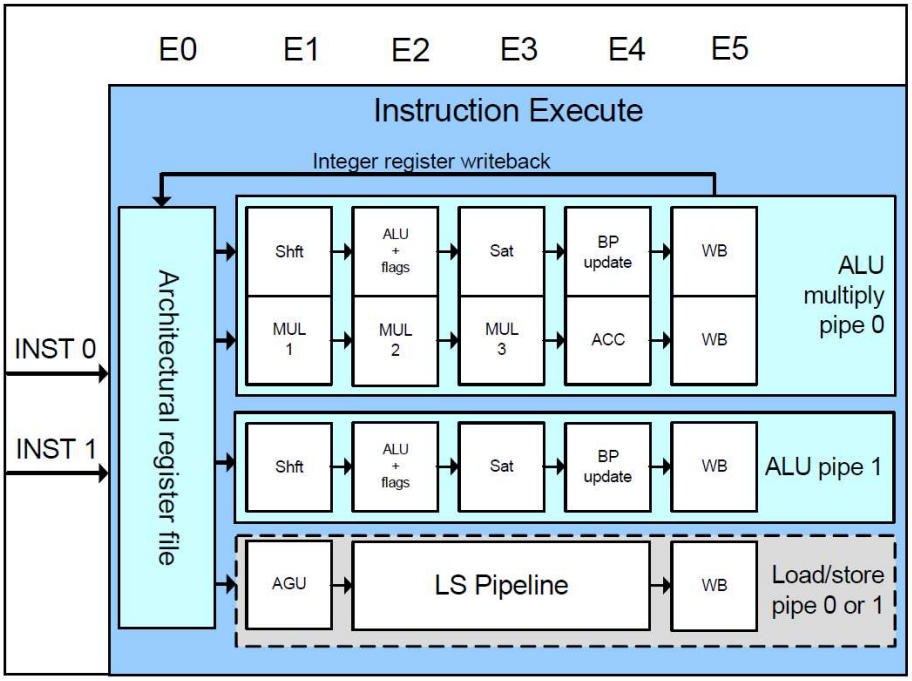
- **D2:** permet de reprendre une instruction non traitée à cause d'un aléa => pending queue
- **D3 :** Détection des aléas par scoreboarding, détermination des instructions à exécuter (0, 1 ou 2)
- **D4 :** accès à la banque de registres. Utilisation de registres temporaires

Banque de registres

System and User	FIQ	Supervisor	Abort	IRQ	Undefined	Secure monitor
r0	r0	r0	r0	r0	r0	r0
r1	r1	r1	r1	r1	r1	r1
r2	r2	r2	r2	r2	r2	r2
r3	r3	r3	r3	r3	r3	r3
r4	r4	r4	r4	r4	r4	r4
r5	r5	r5	r5	r5	r5	r5
r6	r6	r6	r6	r6	r6	r6
r7	r7	r7	r7	r7	r7	r7
r8	r8_fiq	r8	r8	r8	r8	r8
r9	r9_fiq	r9	r9	r9	r9	r9
r10	r10_fiq	r10	r10	r10	r10	r10
r11	r11_fiq	r11	r11	r11	r11	r11
r12	r12_fiq	r12	r12	r12	r12	r12
r13	r13_fiq	r13_svc	r13_abt	r13_irq	r13_und	r13_mon
r14	r14_fiq	r14_svc	r14_abt	r14_irq	r14_und	r14_mon
r15	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)	r15 (PC)

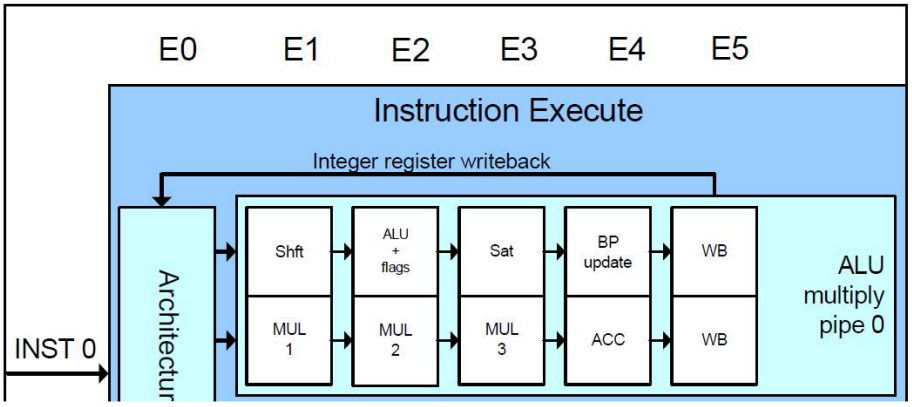
EXECUTE (1)

Entiers



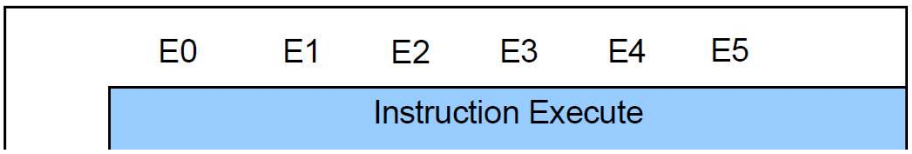
EXECUTE (2)

Entiers

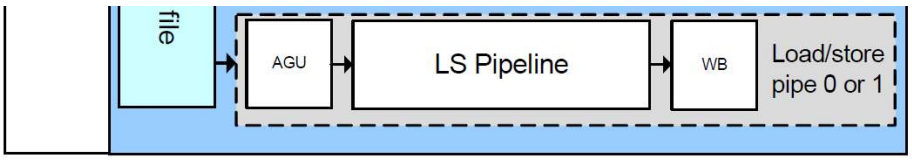


- E0 : Lecture de la banque de registre (6 registres lus simultanément)
- E1 : Shifter / multiplication (1)
- E2 : ALU / multiplication (2)
- E3 : Saturation / multiplication (3)
- E4 : Mise à jour de la prédiction de branchement /accumulation
- E5 : écriture banque de registre (Write back)

EXECUTE (3)



- E0: Lecture de la banque de registre : adresse mémoire et source si Store
- E1: Calcul de l'adresse mémoire
- E2 : Lecture dans la cache
- E3 Load : Forwarding (envoi direct à l'ALU)
- E3 Store: Ecriture dans la cache
- E4 : Mise à jour cache niveau 2
- E5 (Load): écriture banque de registre (Write back)



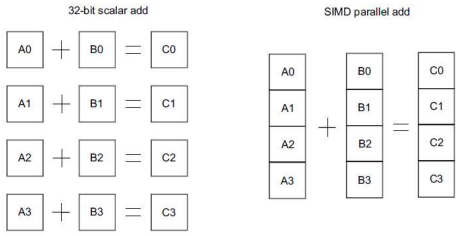
Cache L1 et L2

- Cache niveau 1 : 16 ou 32 KB instructions + 16 ou 32 Ko données2
- Cache niveau 2 : 0 - 1 Mo3
- opt. ECC
- MMU + TrustZone

Cortex-A8 – coprocesseur NEON

Introduction

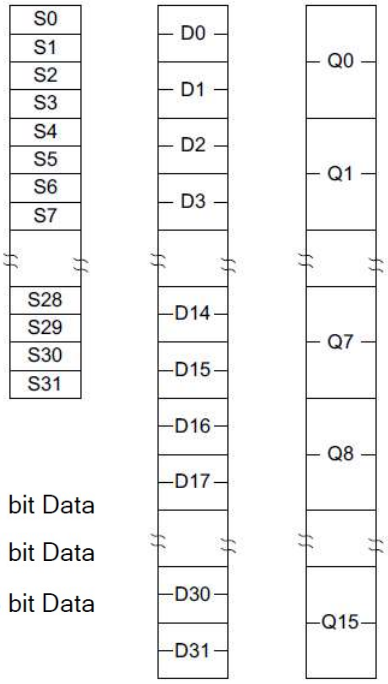
- Unité de calcul SMID (**traitement vectoriel** 64 bits) qui peut être intégrée à l'ARM Cortex
- Instructions spéciales **vectorisées** pour
  - Traitement par blocs (**pixels** ou autre)
  - Traitement a**udio/vidéo/image** et codecs
- Données: **entiers** (signés ou non) ou **flottants** 8, 16, 32 ou 64 bits
- Orienté **traitement du signal**
- **Librairies** utilisant du code NEON
  - OpenMAX: API pour le traitement audio, vidéo et images
  - Eigen2 : librairie mathématique algèbre linéaire et calcul matriciel
  - Pixman : librairie graphique 2D
  - GPL H.264 encodeur vidéo (MPEG-4).
  - Math-neon: librairie mathématique C optimisé NEON

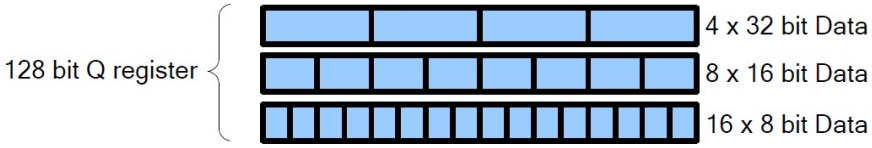


Registres

128 bytes de registres vus comme :

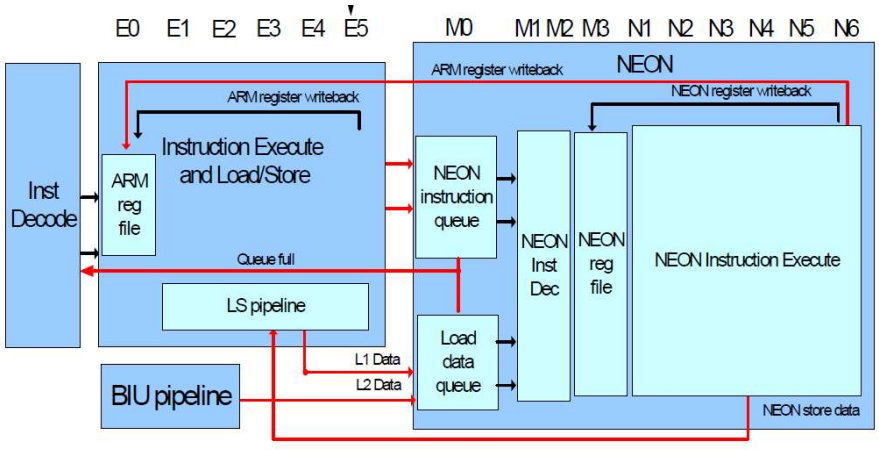
- S: 32 registres 32 bits
- D: 32 registres 64 bits ou combinaison
- Q: 16 registres 128 bits ou combinaison





Pipeline (1)

- Pipeline NEON à la suite du pipeline «Integer»
- **Pas de fetch**, reçoit 1 ou 2 instructions de E5



Pipeline (2)

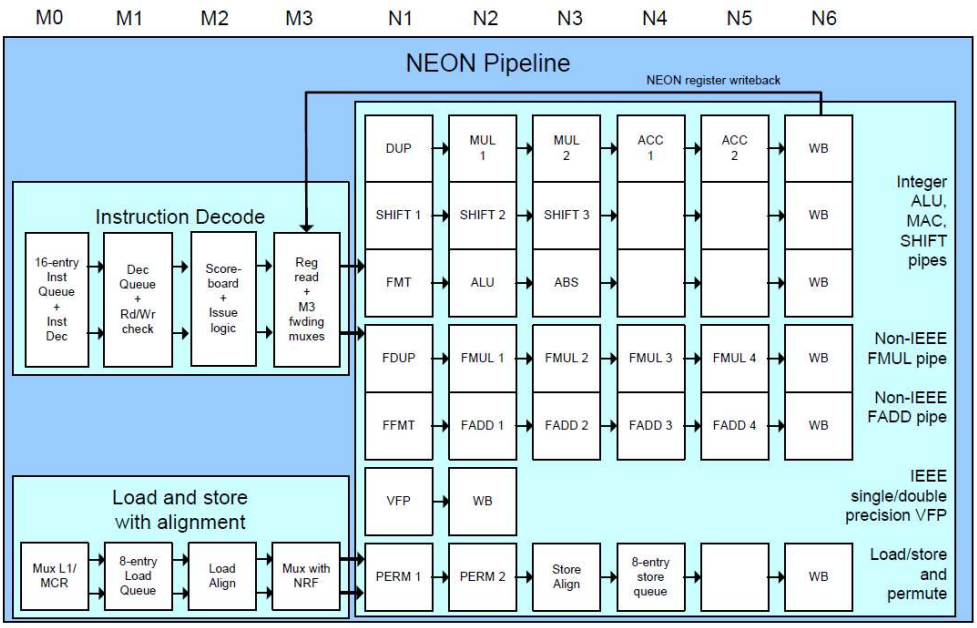
DECODE :

M0: Queue d'instructions NEON

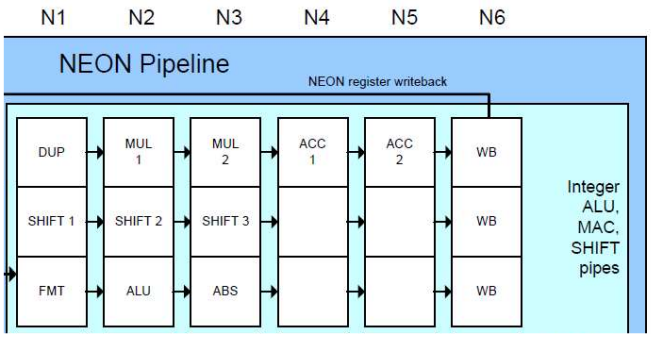
M1: Décodage

M2: Scoreboard (aléas)

M3: Lecture registres



Traitement des *entiers* vectorisés



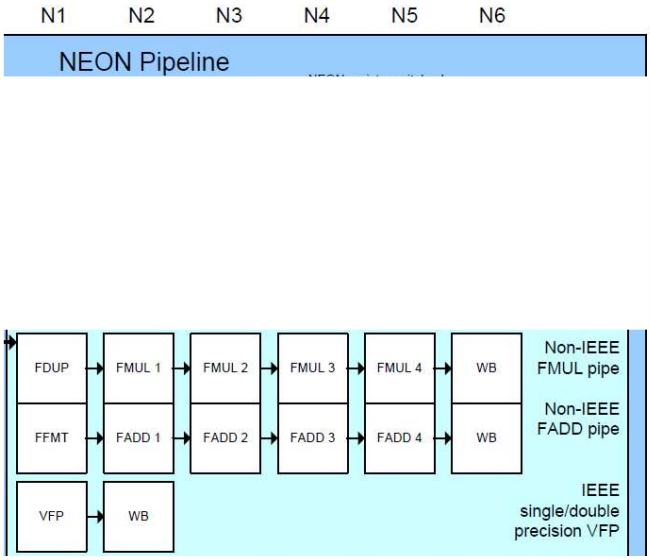
MAC : multiplication (2x32x16 bits),
Shift (2x64 bits) + accumulation
ALU(2x64 bits): compare, test , max

Traitement des *flottants* (NFP)

Multiplication (2x32 bits, simple-précision)

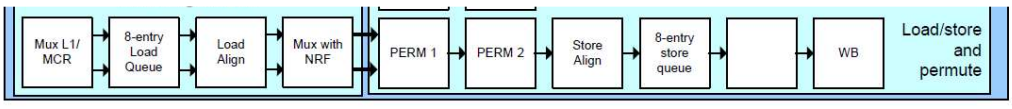
Addition(2x32 bits, simple-précision)

VFP, implémentation non-pipeliné de la norme IEEE

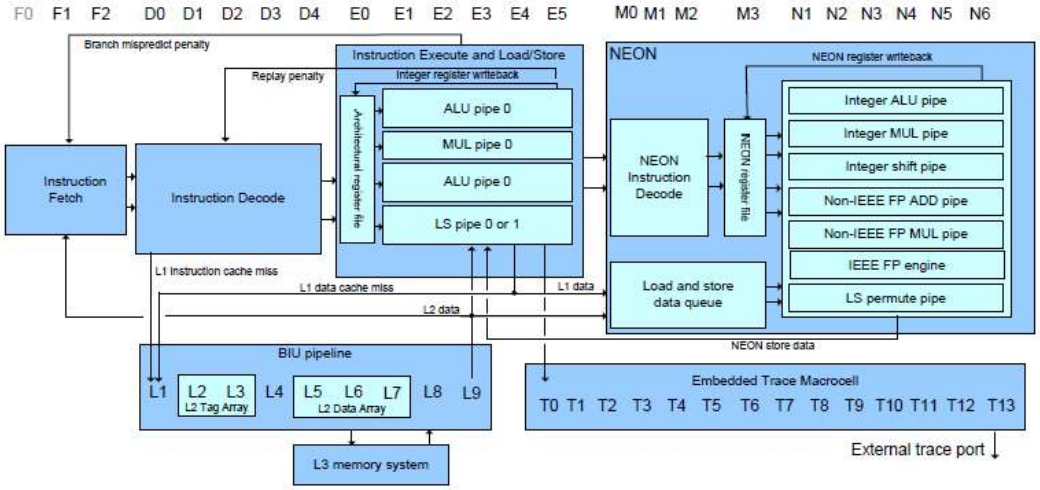


Load/Store

Load /store pipeline, bus 128 bits, données provenant des caches L1 ou L2



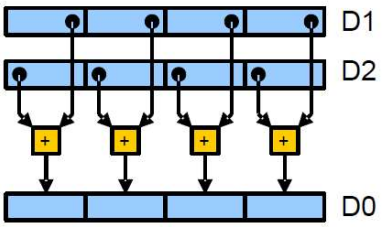
Pipeline complet (NEON, Trace)



Instructions vectorisées : VADD

- Addition de 4 entiers 16 bits (4 additions)
- Source : 4 entiers stockés dans les vecteurs 64 bits D1 et D2
- Résultat: 4 entiers 16 bits dans le vecteur 64 bits D0

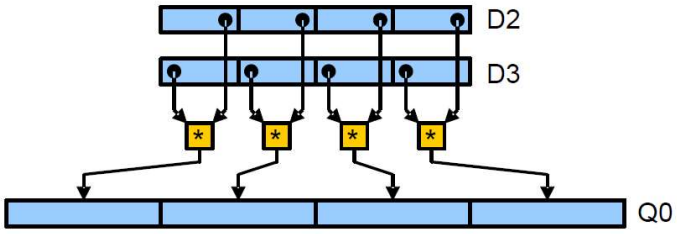
```
VADD.I16 D0, D1, D2
```



Instructions vectorisées : VMUL

- Multiplication de 4 entiers 16 bits (4 multiplications)
- Source : 4 entiers stockés dans les vecteurs 64 bits D2 et D3
- Résultat: 4 entiers 32 bits dans le vecteur 128 bits Q0

```
VMUL.I32.S16 Q0, D2, D3;
```
This instruction will cause



Instructions vectorisées : VSHR

- Shift de 4 entiers 32 bits (4 shifts)
- Source : 4 entiers stockés dans le vecteur 128 bits Q1
- Résultat: 4 entiers 16 bits dans le vecteur 64 bits D0

```
VSHR.I16.I32 D0, Q1,#5
```
This instruction will ca