RC2 - Programmation NEON-v8

Ce cours explique comment utiliser les instructions SIMD NEON de l'ARMv8 pour accélérer les algorithmes multimédia

Objectifs

- Ce cours a été conçu pour les programmeurs souhaitant exécuter des algorithmes multimédia sur les unités d'exécution NEON Single Instruction Multiple Data des processeurs ARMv8.
- L'évolution de l'architecture NEON entre l'ARMv7 et l'ARMv8 est détaillée.
- Chaque famille d'instructions est détaillée, d'abord au niveau assembleur, puis au niveau C à l'aide des macros définies dans le fichier arm_neon.h.
- Plusieurs utilisations subtiles des instructions de traitement sont présentées.
- Les instructions de chargement / rangement de vecteurs et d'éléments de vecteurs sont étudiées, et des règles d'organisation des données en mémoire sont fournies pour minimiser le nombre d'accès mémoire.
- Le fonctionnement sous-jacent du cache ainsi que les mécanismes de préchargement (instruction et prefetch matériel) sont détaillés pour expliquer comment un traitement peut être pipeliné .
- Le cours montre comment des algorithmes typiques du traitement du signal comme le FIR et la FFT peuvent être vectorisés puis optimisés pour s'exécuter sur l'unité NEON.
- Les opérations cryptographiques sont également détaillées, avec l'explication des algorithmes pris en charge.

Prérequis

- Connaissance des jeux d'instructions ARMv7.

Environnement du cours

- Cours théorique
  - Support imprimé et PDF (en anglais).
  - Assistance du formateur tout au long de la formation.
- Chaque session débute par un point avec les stagiaires.

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Plan du cours

Premier jour

Introduction à NEON

- Clarification des ressources partagées entre NEON et l'unité flottante scalaire
- Explication des différences entre AArch32 et AArch64
- Bancs de registres NEON
  - Registres S, D et Q (AArch32)
  - Registres B, H, S, D et V (AArch64)
- Types de données
- Vecteur et scalaire

- Registres système associés
- Questions d'alignement
- Activation de NEON
- Différences entre NEONv7 et NEONv8

Syntaxe des instructions NEON

- Instructions produisant des résultats plus larges / plus étroits
- Modificateurs d'instructions
- Choix de la forme
- Choix du type d'opérande / de résultat
- Souplesse de la syntaxe
- Déclaration de vecteurs initialisés en langage C
- Utilisation d'unions avec des vecteurs et des tableaux de vecteurs pour simplifier le débogage
- Conversion de type des vecteurs

Instructions de transfert de données

- Move
- Swap
- Table lookup
- Transposition de vecteur
- Zip / unzip de vecteurs
- Transfert de données entre NEON et l'unité entière
  - Travaux pratiques : clarification des instructions narrow et long, construction d'un vecteur à partir d'octets choisis dans une paire de vecteurs

Instructions arithmétiques

- Instructions arithmétiques
  - Add, arithmétique modulo et saturée
  - Division / doublement du résultat
  - Arrondi
  - Subtract
  - Multiply
  - Multiply accumulate / Multiply subtract
  - Valeur absolue
  - Min / Max
- Instructions de conversion
  - Conversion de nombres flottants en nombres à virgule fixe
  - Conversion de nombres à virgule fixe en nombres flottants
- Instructions arithmétiques avancées
  - Estimation de l'inverse, estimation de l'inverse de la racine carrée, algorithme de Newton-Raphson
  - Instructions pairwise

Deuxième jour

Instructions logiques et sur champs de bits

- Comparaison d'éléments
- Instructions logiques
  - ET logique, Bit Clear, OU, OU exclusif
  - Opérations avec valeurs immédiates
- Instructions sur champs de bits
  - Comptage des zéros, des uns et des bits de signe de tête
  - Instructions d'insertion bit à bit
  - Instructions d'insertion bit à bit conditionnelles, éviter les branchements

- Décalages avec arrondi et saturation possibles
- Inversion de champ de bits

Extension cryptographique NEON

- L'extension Cryptography
- Algorithmes pris en charge
  - AES
  - SHA1
  - SHA256

Techniques d'optimisation

- Vectorisation automatique
- Réglage des boucles pour un résultat optimal
  - Éviter les rebouclages
  - Éviter les conditions dépendantes de la boucle
  - Éviter les terminaisons anticipées
  - Remplissage des boucles
- Pointeurs et tableaux
  - adressage indirect
  - aliasing de pointeurs et restrict
- Appels de fonction et inlining
  - promesses
- Éviter les dépendances de données

Exemples de code NEON

- Filtre FIR
  - Conversion de l'algorithme scalaire en algorithme vectoriel
  - Recherche des instructions NEON codant l'algorithme vectoriel
  - Optimisation du code
  - Utilisation du moniteur de performance pour régler l'algorithme
- FFT (DFT)
  - Conversion de l'algorithme scalaire en algorithme vectoriel, comprendre comment les propriétés du cercle permettent de traiter 4 angles simultanément
  - Recherche des instructions NEON codant l'algorithme vectoriel
  - Optimisation du code
  - Utilisation du moniteur de performance pour régler l'algorithme