Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Systèmes Embarqués 1 & 2

a.07 - Architecture interne

Classes T-2/I-2 // 2017-2018

Daniel Gachet | HEIA-FR/TIC

a.07 | 28.09.2017

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale
Fachhochschule Westschweiz

Contenu

- Introduction
- Architecture interne
- Composants internes
- Modes de fonctionnement
- Registres
- Exécution des instructions
- Encodage des instructions

Introduction

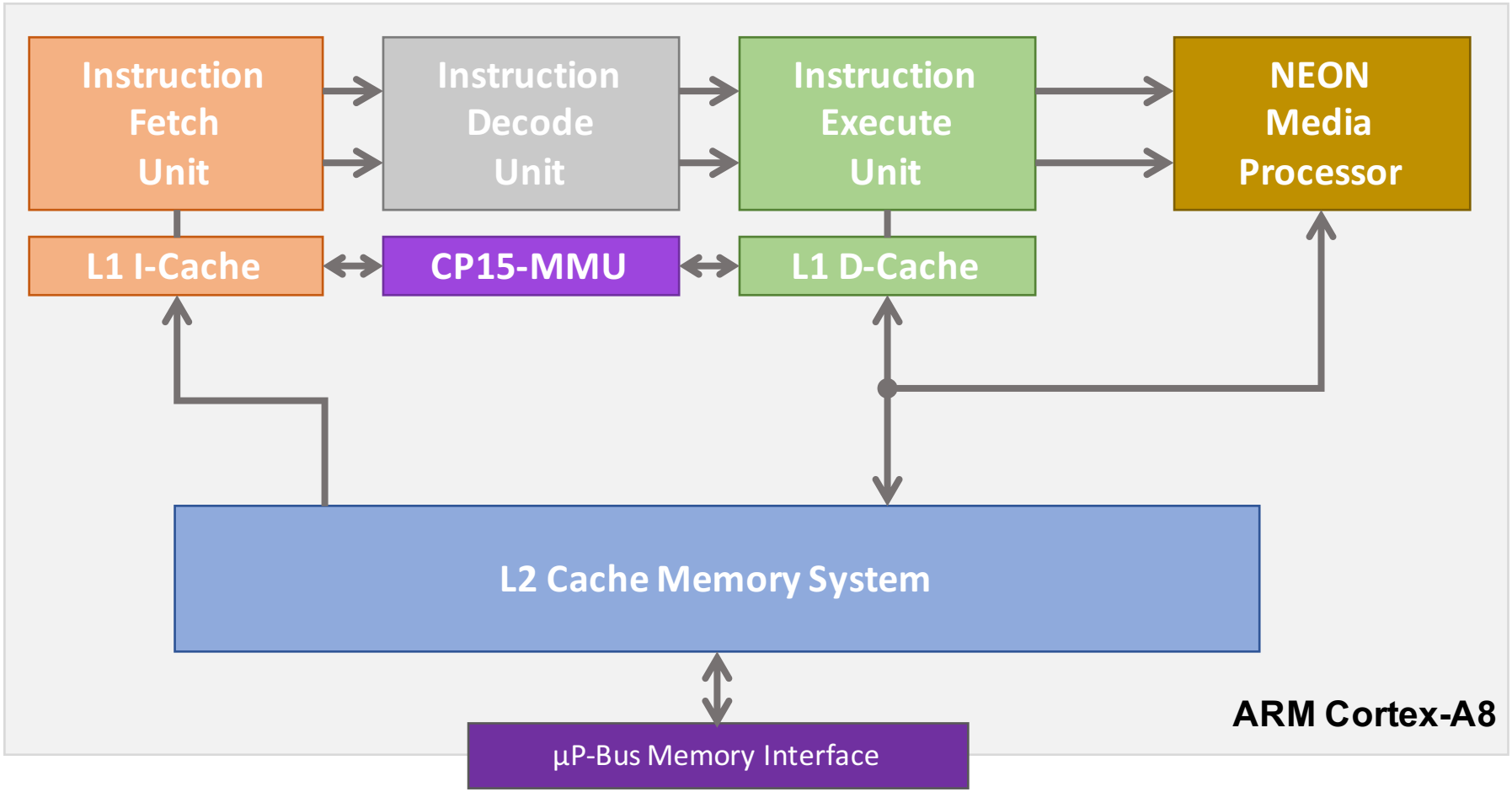
- ARM *(Advanced RISC Machine)* est une famille d'architectures de microprocesseurs pour divers environnements avec un jeu d'instructions de type RISC *(Reduced Instruction Set Computing)*

- La série des processeurs ARM Cortex version 7 se décline en 3 familles principales
  - La famille ARM Cortex-**A** (v7-A)
    - Processeurs d'applications pour des systèmes d'exploitation complets
  - La famille ARM Cortex-**R** (v7-R)
    - Processeurs embarqués pour des applications de contrôle temps réel
  - La famille ARM Cortex-**M** (v7-M)
    - Processeurs pour des applications de type SoC *(System on Chip)*

Architecture des processeurs ARM

- Le µP ARM implémente les caractéristiques propres aux processeurs RISC
  - instructions de longueur unique et fixe (32 bits) afin de simplifier leur décodage
  - registres de taille et de format uniforme (32 bits)
  - mécanismes de transfert *(load and store)* avec des opérations s'effectuant uniquement sur les registres et non avec la mémoire
  - modes d'adressage simples où toutes les adresses de transfert sont déterminées uniquement à partir du contenu des registres
  - modes d'adressage auto-incrémentés et auto-décrémentés afin d'optimiser l'exécution de boucles
  - instructions assembleur s'exécutant conditionnellement pour optimiser la vitesse

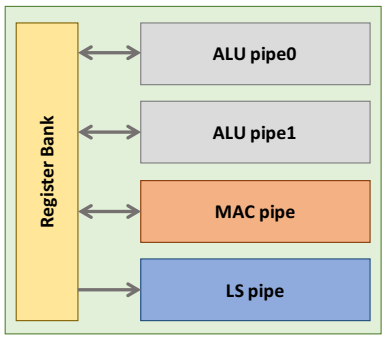
Architecture interne du µP ARM Cortex-A8

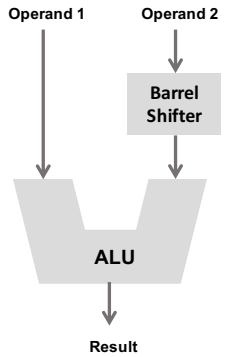


- *Instruction Fetch Unit*
  Cette unité est responsable pour le chargement des nouvelles instructions à exécuter. Elle est connectée à la mémoire principale externe du µP par l'intermédiaire de la mémoire cache d'instructions de premier niveau (L1 I-Cache).
- *Instruction Decode Unit*
  Cette unité est chargée d'analyser les instructions pour déterminer les opérandes et le type d'instruction à exécuter. Le µP est capable de traiter deux instructions en parallèle.

Composantes internes (II)

- *Instruction Execute Unit*
  Cette unité composée de quatre pipelines
  - deux pipelines symétriques pour les opérations arithmétiques et logiques (ALU)
  - un pipeline pour les opérations de multiplications (MAC)
  - un pipeline pour les opérations de transfert (LS)





  - L'ALU est équipée d'un barrel shifter pour exécuter les instructions avec des opérations de rotation ou de décalage en 1 cycle d'horloge
  - Toutes les opérations sont effectuées sur des mots de 32 bits
- L'unité est connectée à la mémoire principale par l'intermédiaire de la mémoire cache de donnée de premier niveau (L1 D-Cache)

Composantes internes (III)

- *NEON Media Processeur*
  Ce processeur permet d'effectuer des opérations avancées SIMD et d'exécuter des opérations sur les nombres réels
- *CP15-MMU*
  Le CP15 et la MMU *(Memory Management Unit)* sont chargés du contrôle des accès avec la mémoire principale externe et les périphériques ainsi que de la translation des adresses virtuelles en adresses physiques
- *L2 Cache Memory System*
  La mémoire cache de niveau 2 implémente la mémoire tampon entre la mémoire principale et les unités internes du µP
- *µP-Bus Memory Interface*
  Cette interface est l'interface de communication entre le µP et les périphériques externes (mémoire et/ou contrôleurs)

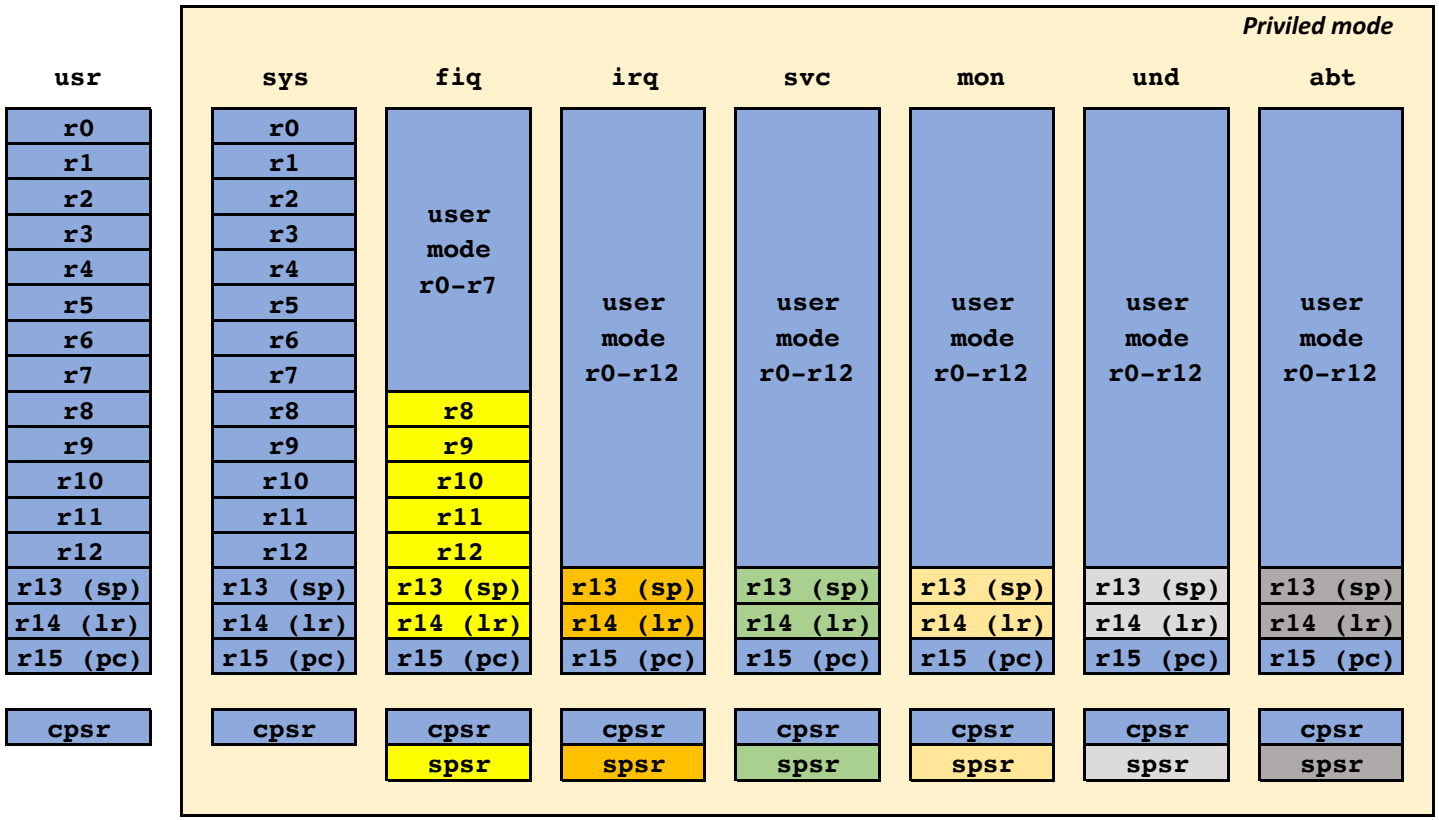
Modes de fonctionnement

- Le processeur implémente de 8 modes de fonctionnements

Mode	Abrv.	Code	Description
User	**usr**	**0b10000**	**Mode d'exécution normal de programme par les utilisateurs (OS)**
FIQ	fiq	0b10001	Mode actif lors du traitement d'interruptions demandant un traitement rapide
IRQ	**irq**	**0b10010**	**Mode actif lors du traitement d'interruptions normales**
Supervisor	**svc**	**0b10011**	**Mode utilisé lors de l'exécution de code propre au système d'exploitation (OS)**
Monitor	mon	0b10110	Mode sécurisé, disponible seulement avec l'extension
Abort	abt	0b10111	Mode actif lors d'accès invalides avec la mémoire
Undefined	und	0b11011	Mode actif lors de l'exécution d'instructions non définies / supportées par le processeur
System	sys	0b11111	Mode privilégié permettant d'accéder les registres du mode utilisateur

Registres internes

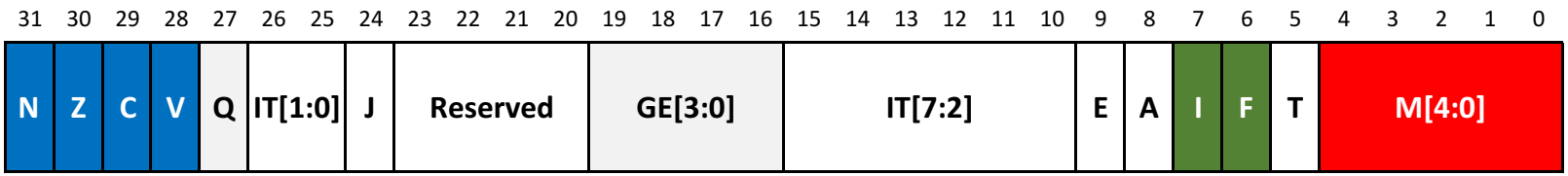
- Le processeur dispose de 40 registres internes à 32 bits



- En mode privilégié, le µP a accès à l'ensemble de ses ressources internes (registres, coprocesseurs,...)
- En mode non privilégié, le µP dispose d'un accès limité

Program Status Register

- Le registre de statut est accessible depuis tous les modes de fonctionnement du processeur. Il contient les fanions de conditions (flags), les bits de contrôle des interruptions, les bits de contrôle du mode de fonctionnement du processeur ainsi que d'autres bits relatifs au fonctionnement du µP



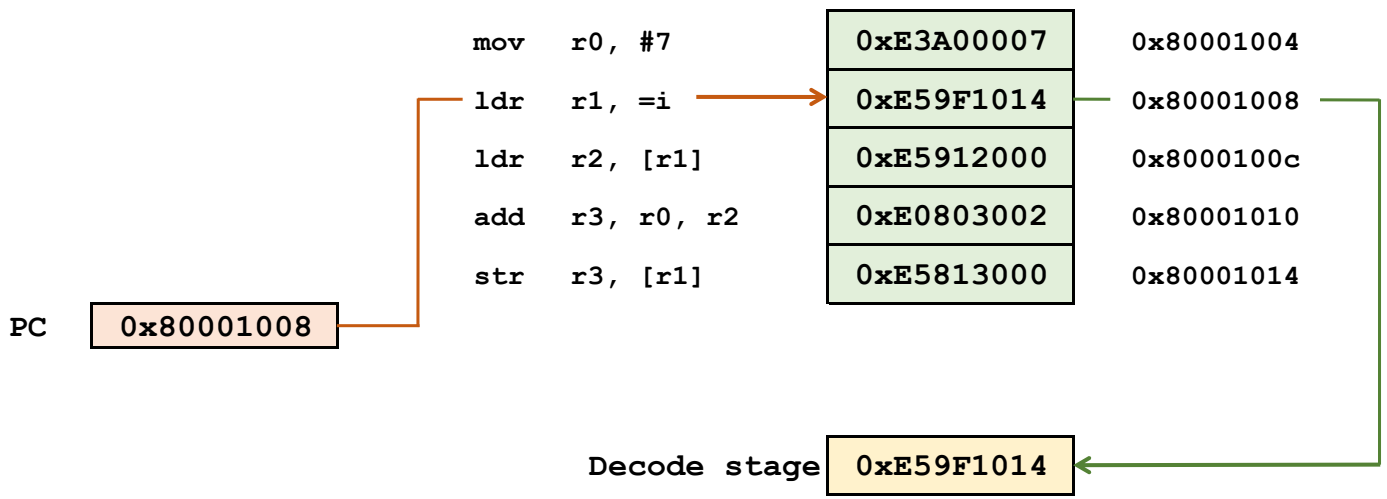
- En mode privilégié, ce registre est nommé CPSR *(Current Program Status Register)*
- Le registre SPSR *(Saved Program Status Register)* sert à la sauvegarde du CPSR
- En mode non privilégié, ce registre est nommé APSR *(Application Program Status Register)* où seuls les fanions sont accessibles (bits[31:27]+[19:16])

Program Status Register (II)

- « Condition code flags »
  The N, Z, C, and V (Negative, Zero, Carry and oVerflow) bits are collectively known as the condition code flags, often referred to as flags. The condition code flags in the CPSR/APSR can be tested by most instructions to determine whether the instruction is to be executed.
- « Interrupt disable bits »
  I, and F are the interrupt disable bits
  - I bit disables IRQ *(Interrupt Request)* interrupts when set
  - F bit disables FIQ *(Fast Interrupt Request)* interrupts when set
- « Mode bits »
  M[4:0] are the mode bits. These determine the mode in which the processor operates (usr, sys, fiq, irq, svc, mon, abt, und).

Program Counter

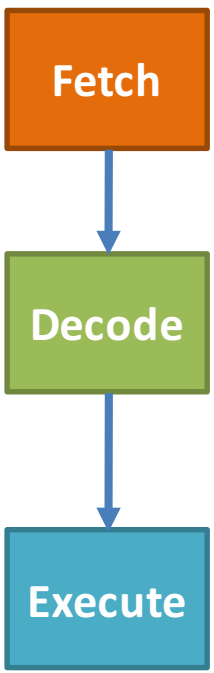
- Chaque instruction assembleur est codée sur 32 bits et est contenue dans une partie consécutive de la mémoire
- Le compteur ordinal *(PC - Program Counter)* contient la position courante du programme en cours d'exécution
- Le contenu des 32 bits pointés par le PC est chargé dans le registre d'instruction *(Instruction Fetch Unit)*, puis décodé *(Instruction Decode Unit)* et finalement exécuté *(Instruction Execute Unit)*
- Le PC est ensuite incrémenté de 4 afin de pouvoir exécuter l'instruction suivante



Exécution - instructions

- L'exécution des instructions assembleur est décomposée en plusieurs étapes, p. ex. en 3 (ARM7), 5 (ARM9) ou 13 (ARM Cortex-A8)

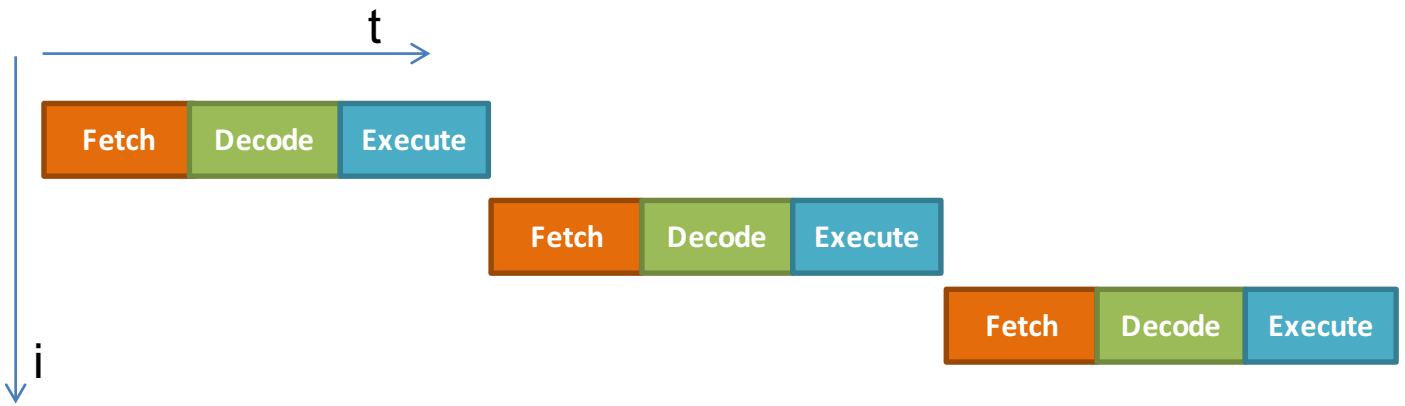
ARM 7 **add r3, r2, r0** ; 0xE0803002



Recherche de l'instruction dans la mémoire	*Lecture du code de l'instruction (0xE0803002)*
Décodage des registres utilisés dans l'instruction	*Identification des registres R3, R2 et R0*
Lecture des valeurs depuis la banque des registres; opération de shift et ALU; Ecriture des valeurs dans la banque des registres	*Lecture des valeurs contenues dans les registres R0 et R2; Addition des R0 et R2; Sauvegarde du résultat dans le registre R3*

Exécution – performances

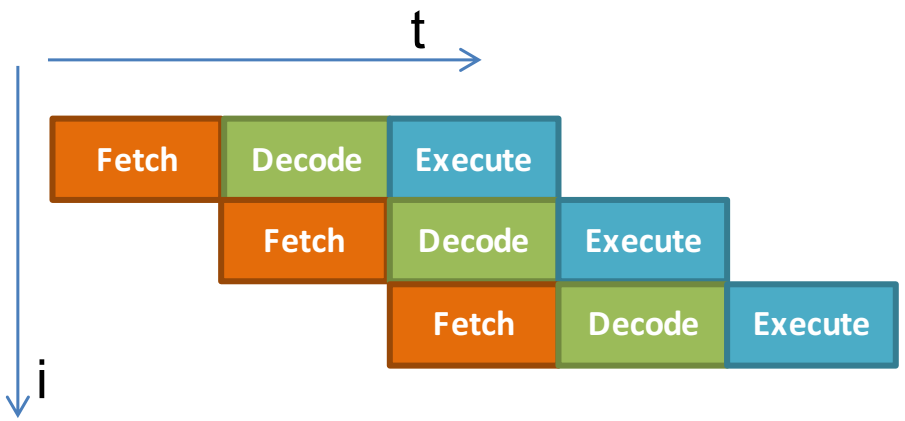
- Les performances d'un microprocesseur sont fortement dépendantes du temps qu'il utilise pour exécuter les instructions assembleur d'un programme
- Ces instructions sont exécutées séquentiellement, en traitant à chaque cycle d'horloge une partie de l'instruction



- Ce mode d'opération est simple, mais peu efficace

Exécution – pipeline

- Pour augmenter ces performances et réduire le temps d'exécution, il serait judicieux d'exécuter plusieurs instructions en parallèle
- Cette technique est connue sous le nom de pipelining



- Pour cela il suffit de dupliquer certaines ressources du microprocesseur (p.ex. l'ALU avec une logique d'incrémentation des adresses)

Encodage du jeu d'instructions

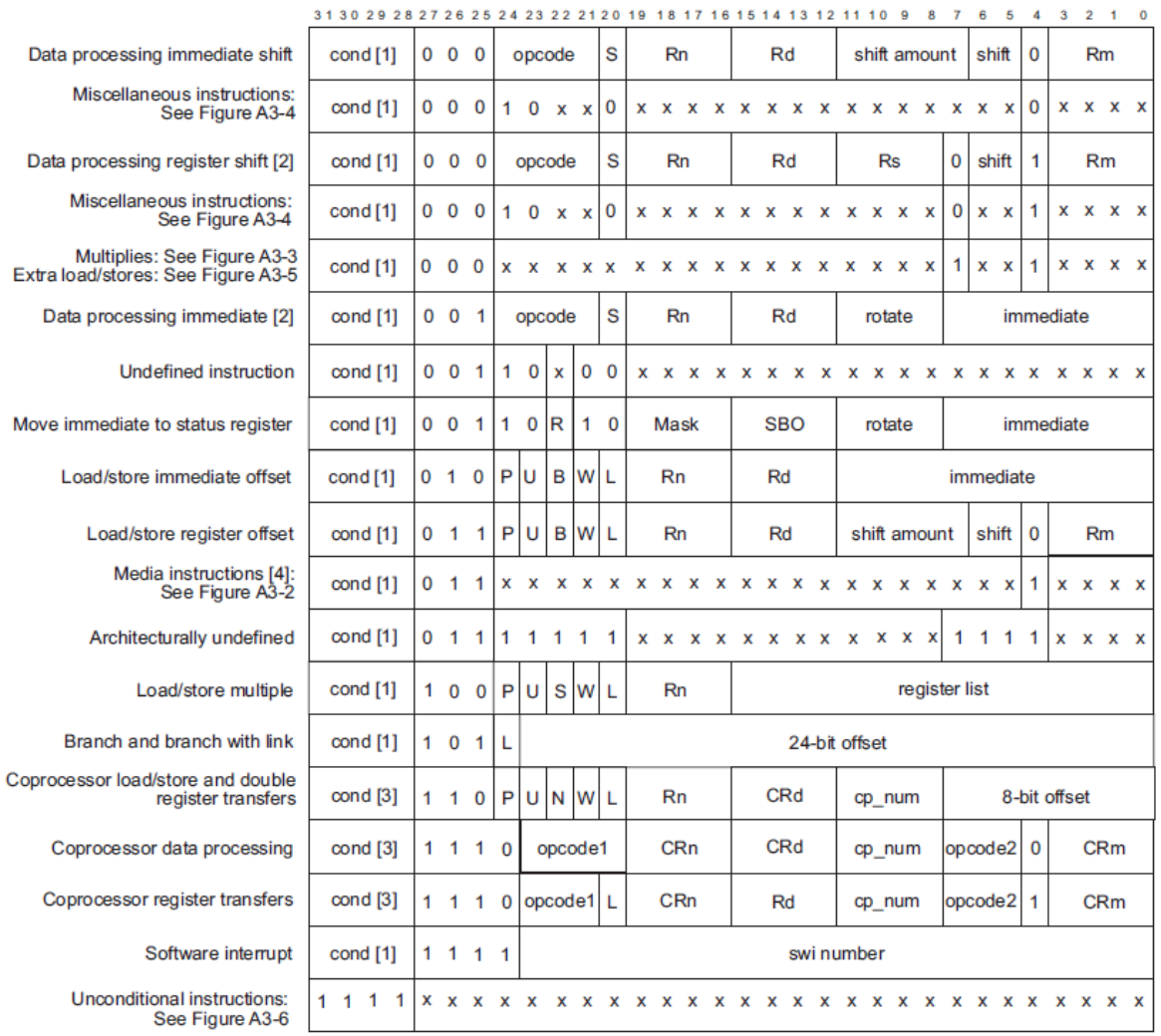
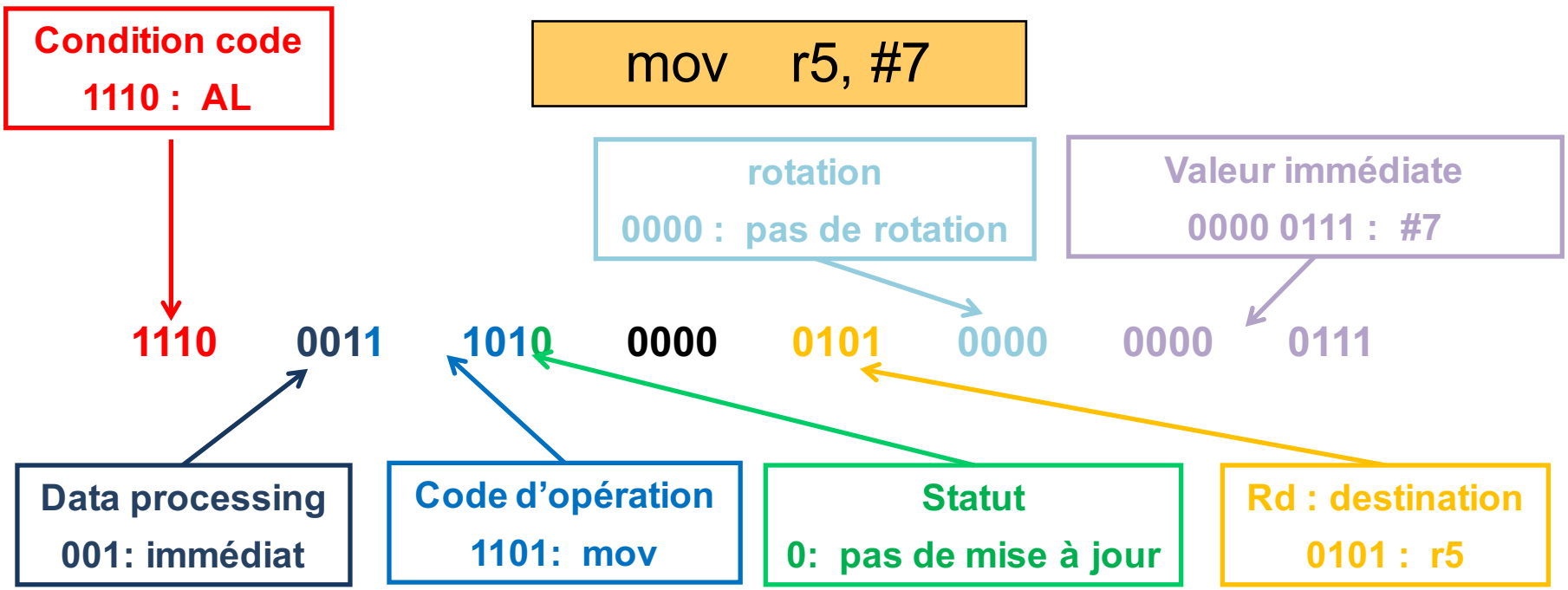


Figure A3-1 ARM instruction set summary

[0] www.arm.com

Exemple d'instruction simple



code machine 0xE3A05007

action sauvegarde la valeur 7 dans le registre R5