

UE VLSI
cours 3: Présentation Architecture et jeu
d'instructions ARM

Jean-Lou Desbarbieux
UMPC 2017

Sommaire

Architecture

Exécution conditionnelle

Jeu d'instruction

- traitement de données

- branchements

- Accès mémoires simples

- Accès mémoires multiples

- Multiplication

Petits programmes

Architectures

Un foisonnement de déclinaisons suivant les fabricants :

- ▶ **ARMv1** ARM1 : 1985 (prototype, jeu d'instruction 32 bits, pipeline 3 étages),
- ▶ **ARMv2** ARM2, ARM3 : 1987 (Pipeline 3 étages, 24 bits d'adresse, commercialisé par Acorn),
- ▶ **ARMv3** ARM6, ARM7 : 2000 (FPU, jeu d'instruction Thumb 16 bits),
- ▶ **ARMv4** ARM9 : (Pipeline 5 étages),
- ▶ **ARMv5** ,
- ▶ **ARMv6** ARM11 : 2003 (jeu d'instruction Thumb2 16/32 bits),
- ▶ **ARMv6M** Cortex-M[0, 0+, 1] (dédié à l'embarqué),
- ▶ **ARMv7A** Cortex-A[8 ...17] (téléphones, tablettes)
- ▶ **ARMv7M** Cortex-M[3 ...7]
- ▶ **ARMv7R** (temps réel),
- ▶ **ARMv8** Cortex-A[50 ...] (64 bits).

Modes d'exécution

7 modes d'exécution :

- ▶ *USER* (mode normal, actif la majeure partie du temps),
- ▶ *FIQ* (traitement des interruptions prioritaires),
- ▶ *IRQ* (traitement des interruptions ordinaires),
- ▶ *Supervisor* (reset et traitement interruptions logicielles),
- ▶ *Abort* (violation d'accès mémoire),
- ▶ *Undef* (erreur décodage instruction),
- ▶ *System* (mode privilégié avec registres utilisateurs à partir de v4).

Registres

Les registres dépendent du jeu d'instruction mais surtout des modes d'exécution supportés :

User 32

registre 0
registre 1
registre 2
registre 3
registre 4
registre 5
registre 6
registre 7
registre 8
registre 9
registre 10
registre 11
registre 12
registre 13 (SP)
registre 14 (LR)
registre 15 (PC)

CPSR

FIQ 32

registre 0
registre 1
registre 2
registre 3
registre 4
registre 5
registre 6
registre 7
registre fiq8
registre fiq9
registre fiq10
registre fiq11
registre fiq12
registre fiq13
registre fiq14
registre 15 (PC)

CPSR

SPSR fiq

...

Accès aux registres

- ▶ *r0-r14* : Toutes les instructions,
- ▶ *r15 (PC)* : Presque toutes les instructions des interruptions prioritaires,
- ▶ *CPSR* : Instructions spécifiques.

L'instruction MOV permet d'effectuer des transferts à destination de registres :

```
MOV r3 , #4
```

Le PC peut être utilisé comme destination :

```
MOV r15 , r14
```

ou

```
MOV pc , lr
```

Le CPSR peut être modifié par une instruction si suffixée S :

```
MOVS r3 , #4
```

CPSR, les flags

Les drapeaux générés par l'ALU sont mémorisés dans le CPSR :

- N** : Le résultat de l'ALU est négatif (bit 31 égal à 1) ;
- Z** : Le résultat de l'ALU est égal à zéro (32 bits égaux à 0) ;
- C** : Retenue générée l'ALU dans le cas des instructions arithmétiques et le décaleur (*shifter*) dans le cas des instructions logiques.
- V** : Dépassement de capacité dans le cas d'une opération arithmétique signée.

Pour les premières versions de l'architecture, ces 4 *flags* sont associés au PC (réduit à 24 bits) dans le registre 15.

Exécution conditionnelle

Les drapeaux CPSR sont très utilisés dans les architectures ARM :

- ▶ Il est possible pour la plupart des instructions de préciser leur action sur les drapeaux (suffixe S).
- ▶ Les branchements peuvent être conditionnés par la valeur des *flags* ce qui est très commun.
- ▶ Particularité numéro 1 des architectures ARM, l'exécution de presque toutes les instructions peut être conditionnée par la valeur des drapeaux.

L'objectif de cette utilisation de prédicat est de réduire le nombre de branchements.

Exécution conditionnelle exemple

Il est possible de conditionner un instruction pour qu'elle ne soit exécutée que si le *flag Z* est égal à 1, typiquement si l'instruction précédente a produit un résultat égal à 0 :

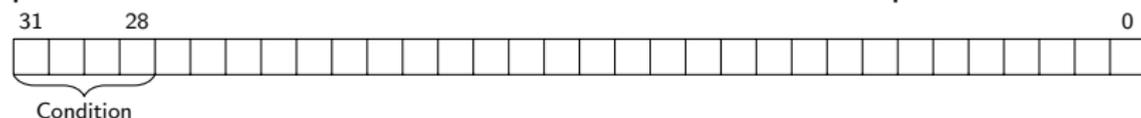
```
MOV r3,#4
```

Devient dans ce cas :

```
MOVEQ r3,#4
```

prédicats

Toutes les instructions sont codées sur 32 bits dont les 4 bits de poids fort définissent les 16 conditions d'exécution possibles :



0000 **EQ** - $Z = 1$

0001 **NE** - $Z = 0$

0010 **HS/CS** - $C = 1$

0011 **LO/CC** - $C = 0$

0100 **MI** - $N = 1$

0101 **PL** - $N = 0$

0110 **VS** - $V = 1$

0111 **VC** - $V = 0$

1000 **HI** - $C = 1$ et $Z = 0$

1001 **LS** - $C = 0$ ou $Z = 1$

1010 **GE** - supérieur ou égal

1011 **LT** - strictement inférieur

1100 **GT** - strictement supérieur

1101 **LE** - inférieur ou égal

1110 **AL** - toujours

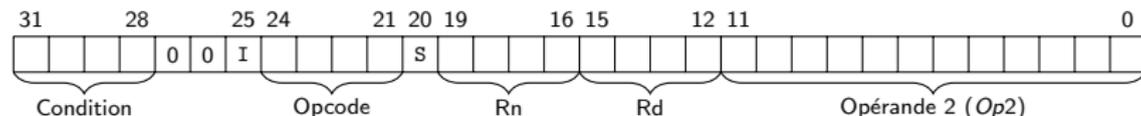
1111 **NV** - réservé.

types d'instructions

Les instructions assembleur ARM sont organisées en catégories :

- ▶ Traitement de données (les plus fréquentes),
- ▶ multiplications,
- ▶ *swap*,
- ▶ branchements,
- ▶ transfert mémoire,
- ▶ coprocesseur,
- ▶ interruptions logicielles.

traitement de données (codage)



Condition : L'instruction n'est exécutée que si la condition sur les *flags* est satisfaite ;

I : Indique si l'opérande 2 est du type immédiat ;

Opcode : Définit l'instruction (voir tableau) ;

S : Indique si les *flags* doivent être mis à jours ;

Rn : Registre correspondant généralement au premier opérande ;

Rd : Registre correspondant généralement à la destination ;

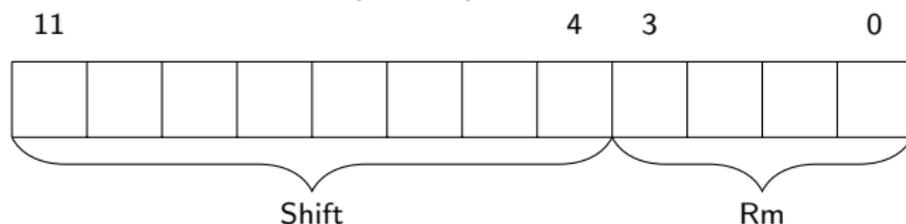
Opérande 2 : Second opérande (multitude de variantes).

Opcode

- 0000 - **AND** : $Rd \leftarrow Rn \text{ AND } Op2$
- 0001 - **EOR** : $Rd \leftarrow Rn \text{ XOR } Op2$
- 0010 - **SUB** : $Rd \leftarrow Rn - Op2$
- 0011 - **RSB** : $Rd \leftarrow Op2 - Rn$
- 0100 - **ADD** : $Rd \leftarrow Rn + Op2$
- 0101 - **ADC** : $Rd \leftarrow Rn + Op2 + C$
- 0110 - **SBC** : $Rd \leftarrow Rn - Op2 + C - 1$
- 0111 - **RSC** : $Rd \leftarrow Op2 - Rn + C - 1$
- 1000 - **TST** : Positionne les *flags* pour $Rn \text{ AND } Op2$
- 1001 - **TEQ** : Positionne les *flags* pour $Rn \text{ XOR } Op2$
- 1010 - **CMP** : Positionne les *flags* pour $Rn - Op2$
- 1011 - **CMN** : Positionne les *flags* pour $Rn + Op2$
- 1100 - **ORR** : $Rd \leftarrow Rn \text{ OR } Op2$
- 1101 - **MOV** : $Rd \leftarrow Op2$
- 1110 - **BIC** : $Rd \leftarrow Rn \text{ AND NOT } Op2$
- 1111 - **MVN** : $Rd \leftarrow \text{NOT } Op2$

Opérande 2 registre

Cas où le champ I (bit 25) est égal à 0 :

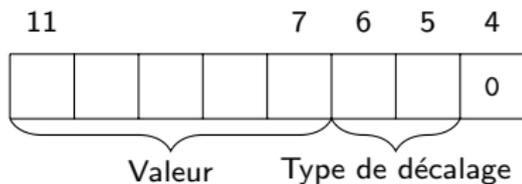


Rm : Registre utilisé comme second opérande ;

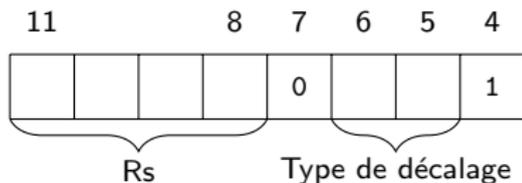
Shift : Décalage appliqué au registre Rm.

Décalage *shift*

Cas où le bit 4 est égal à 0 :



Cas où le bit 4 est égal à 1 :



Il existe 4 types de décalage :

- 00 - Décalage à gauche logique,
- 01 - Décalage à droite logique,
- 10 - Décalage à droite arithmétique,
- 11 - Rotation à droite.

La valeur maximale pour un décalage est 31 pour un registre de 32 bits, valeur codée sous la forme d'un immédiat codé sur 5 bits ou valeur des 5 bits de poids faible du registre Rs.

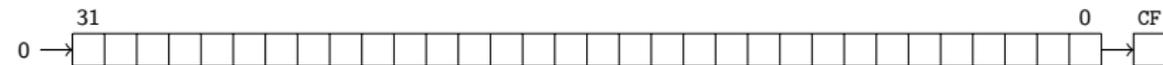
Décalage *shift* suite

Une instruction peut spécifier 5 modes de décalage de son opérande 2 :

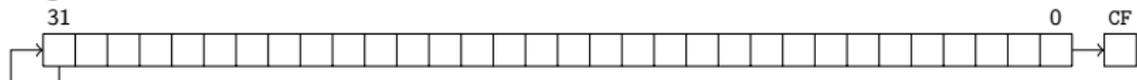
LSL



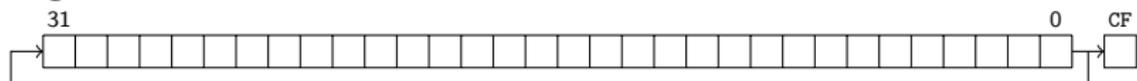
LSR



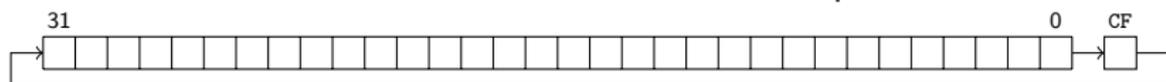
ASR



ROR



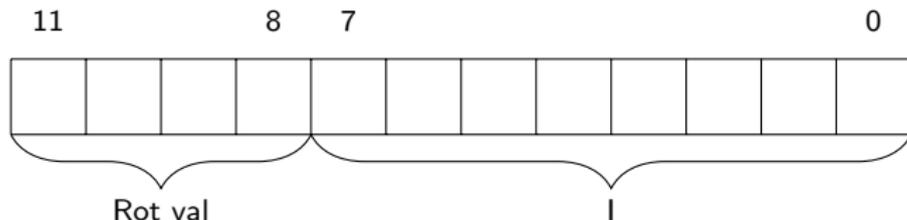
RRX limité à 1 bit, codé comme un ROR avec pour valeur 0.



Si absence de décalage LSL avec pour valeur 0.

Opérande 2 immédiat

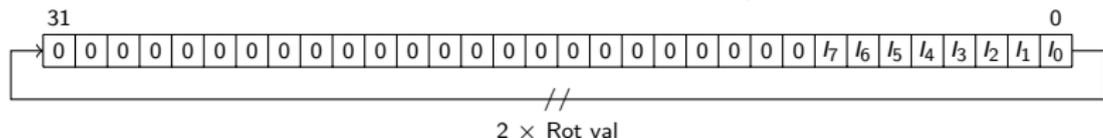
Cas où le champ I (bit 25) est égal à 1 :



I : Valeur utilisée comme second opérande,

Rot val : Décalage appliqué à cette valeur.

Extension sur 32 bits de l'immédiat 8 bits par rotation.



Exemples

Syntaxe générale : $\langle opcode \rangle \{cond\}\{S\} Rd, Rn, \langle Op2 \rangle$

MOV r3, #512

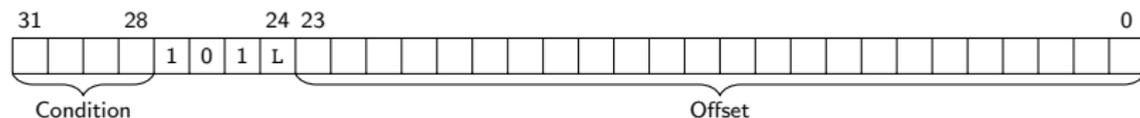
ADDS r6, r5, r4

SUBS r8, r7, r6

ANDEQ r12, r13, r5, **LSL** #7

Donnez le codage correspondant en mémoire.

Branchements (codage)



Condition : L'instruction n'est exécutée que si la condition sur les *flags* est satisfaite ;

L : Si $Link = 1$ $R14 \leq PC + 4$;

Offset : $PC \leq PC + 8 + (Offset \times 4)$.

Syntaxe générale : $B\{L\}\{cond\} < label >$

Exemples

L1:

```
SUBS r3 , r3 , #1
```

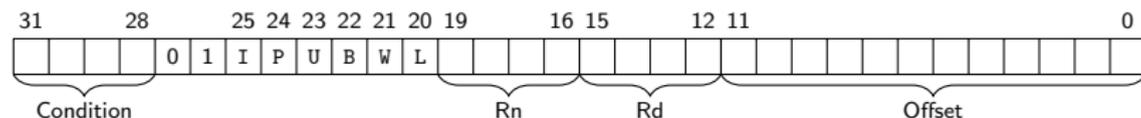
```
BEQ L1
```

```
BL L3
```

L3:

Donnez le codage correspondant en mémoire.

Accès mémoires simples (codage)



Condition : L'instruction n'est exécutée que si la condition sur les *flags* est satisfaite ;

I : L'*Offset* correspond à un immédiat si égal 0 ;

P : Pré/Post indexation (Pré si 1) ;

U : *Up/Down* ajout de l'*Offset* si égal 1 ;

B : *Byte/Word* octet si égal 1 ;

W : *Write-back* modification adresse de base si égal 1 ;

L : *Load/Store* lecture mémoire si égal 1 ;

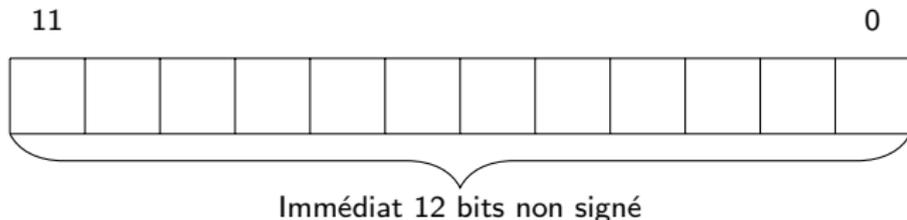
Rn : Registre de base (adresse) ;

Rd : Registre source (écriture) ou destination ;

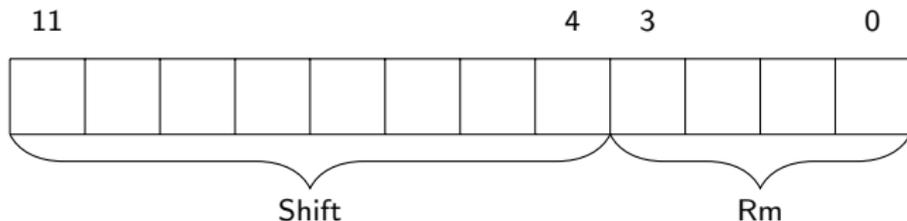
Offset : Immédiat ou registre combiné au registre de base pour constituer l'adresse.

Offset

Cas où le champ I (bit 25) est égal à 0 :



Cas où le champ I (bit 25) est égal à 1 :



Même codage que pour les instructions de traitement de donnée.

Exemples

Syntaxe générale : $\langle LDR|STR \rangle \{cond\}\{B\} Rd, \langle Address \rangle$

$\langle address \rangle$ peut correspondre à :

$\langle label \rangle$

$[Rn]$

$[Rn, \langle \#expression \rangle]$

$[Rn, \{+/-\}Rm, \langle shift \rangle]\{!\}$

$[Rn], \langle \#expression \rangle$

$[Rn], \{+/-\}Rm, \langle shift \rangle$

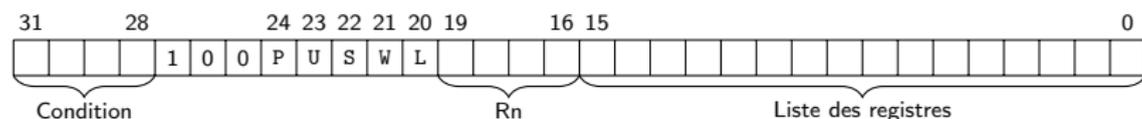
STR R1 , [R2 , R4] !

STR R1 , [R2] , R4

LDREQB R1 , [R2 , R3 , **LSL** #2]

Donnez le codage correspondant en mémoire.

Accès mémoires multiples (codage)



Condition : L'instruction n'est exécutée que si la condition sur les *flags* est satisfaite ;

P : Pré/Post indexation (Pré si 1) ;

U : *Up/Down* ajout de l'*Offset* si égal 1 ;

S : Voir spécification détaillée ;

W : *Write-back* modification adresse de base si égal 1 ;

L : *Load/Store* lecture mémoire si égal 1 ;

Rn : Registre de base (adresse) ;

Liste : Liste des registres source / destination.

Exemples

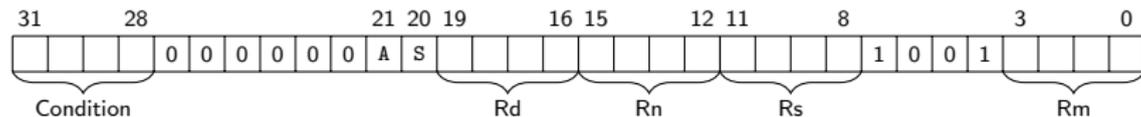
Syntaxe générale :

$\langle LDM|STM \rangle \{cond\} \langle FD|ED|FA|EA|IA|IB|DA|DB \rangle Rn\{!\}, \langle Rlist \rangle$

LDMFD **SP!** , { R0 , R1 , R2 }

STMIA R0 , { R0–R15 }

Multiplication (codage)



Condition : L'instruction n'est exécutée que si la condition sur les *flags* est satisfaite ;

A : $Rd \leq Rm \times Rs + Rn$ si égal 1 sinon
 $Rd \leq Rm \times Rs$.

S : *flags* modifiés si égal 1.

Exemples

Syntaxe générale : *MUL*{*cond*}{*S*}*Rd, Rm, Rs* ou
MLA{*cond*}{*S*}*Rd, Rm, Rs, Rn*

MUL R1 , R2 , R3

MLAEQS R1 , R2 , R3 , R4

PGCD

```
while (R1 != R2)
{
  if (R1 > R2) R1 -= R2;
  else R2 -= R1;
}
```