

# Architecture du processeur

Pluvinage Lucas, Fehr Mathieu, Dang-nhu Hector, Voss Malachi

January 12, 2017

## 1 Modèle de processeur

Nous avons choisi de beaucoup nous inspirer des processeurs ARM, qui ont l'avantage d'avoir un jeu d'instruction à la fois simple et complet, tout en possédant une certaine efficacité grâce aux conditionnement des instructions, permettant de réduire le nombre de jump à effectuer dans les instructions assembleurs.

## 2 Mémoires et registres

Le processeur possède une unique RAM et une unique ROM. Au démarrage de la simulation, la RAM est entièrement vide. La ROM contient le programme sous forme d'opcodes, générés par du code assembleur.

Le processeur contient 16 registres adressés sur 4 bits. Le registre %r00 pointe sur l'instruction actuelle à lire dans la ROM.

## 3 Opcodes

Une instruction assembleur est écrite sous 32 bits, et est structurée ainsi :

0-3	4	5	6	7-10	11	12-15	16-19	20-23	24-27	28-31
COND		1	Opération sur la RAM							
COND		0	1	OPCODE	Set flag	Registre rd	Registre op1	Opérande op2		
COND		0	1	OPCODE	Set flag	Registre rd	Registre op1	Shift	Constante	
COND		0	0	OPCODE	Set flag	Registre rd	Registre op1	Shift		r2

### 3.1 Flags

A chaque opération, si le booléen set flag est activé, alors les flags sont recalculés :

- N : La dernière instruction renvoie un entier négatif
- Z : La dernière instruction renvoie zéro
- C : La dernière instruction fait un dépassement dans la représentation non signée
- V : La dernière instruction fait un dépassement dans la représentation signée

### 3.2 Conditionnelles

Voici les codes des conditions d'executions des opérations:

Code	Nom	Condition sur les flags	Signification (pour CMP ou SUB)
0000	EQ	Z	Égalité
0001	NEQ	$\bar{Z}$	Non égalité
0010	CS/HS	C	Carry set
0011	CC/LO	$\bar{C}$	Carry clear
0100	MI	N	Négatif
0101	PL	$\bar{N}$	Positif ou nul
0110	VS	V	Overflow signé
0111	VC	$\bar{V}$	Pas d'overflow signé
1000	HI	C and $\bar{Z}$	Strictement plus grand non signé
1001	LS	$\bar{C}$ or Z	Plus petit non signé
1010	GE	N == V	Plus grand signé
1011	LT	N != V	Strictement plus petit signé
1100	GT	$\bar{Z}$ and (N == V)	Strictement plus grand signé
1101	LE	Z or (N != V)	Plus petit signé
1111	AL		Toujours exécuté

## 4 Instructions assembleur

La syntaxe pour effectuer les opérations en assembleur suit la syntaxe spécifiée sur les processeurs ARM.

### 4.1 Opérations arithmétiques et booléennes

Voici les opérations assembleur arithmétiques et booléennes :

Opcode	Nom assembleur	effet
0000	ADD	$R[rd] = op1 + op2$
0001	ADC	$R[rd] = op1 + op2 + c$
0010	RSB	$R[rd] = op2 - op1$
0011	RSC	$R[rd] = op2 - op1 + c - 1$
0100	CMP	set les flags sur $op1 - op2$
0101	CMN	set les flags sur $op1 + op2$
0110	SUB	$R[rd] = op1 - op2$
0111	SBC	$R[rd] = op1 - op2 + c - 1$
1000	AND	$op1 \text{ and } op2$
1001	TST	set les flags sur $op1 \text{ and } op2$
1010	ORR	$op1 \text{ or } op2$
1011	BIC	$op1 \text{ nand } op2$
1100	NOT	not $op2$
1101	MOV	$op2$
1110	EOR	$op1 \text{ xor } op2$
1111	TEQ	set les flags sur $op1 \text{ xor } op2$

Où c représente le carry

### 4.2 Accès à la mémoire

Deux opérations sont possibles lors de l'accès à la mémoire :

Opcode	Nom assembleur	effet
0	LDR	$R[Rd] = \text{mémoire à l'adresse } op2$
1	STR	$\text{mémoire à l'adresse } op2 = R[Rd]$