Architecture du processeur

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

January 16, 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 grace aux conditionnement des instructions, permettant de réduire le nombre de jump à effectuer dans les instructions assembleurs.

2 Mémoires et registres

Le processeur que nous utilisons possède 16 registres, dont un program counter. Il possède une unique ROM addressée sur 16 bits, dont les éléments sont des entiers binaires de 32 bits. Notre procésseur n'utilises pas de RAM, bien que les registres soient modélisés sous forme de RAM adressée sous 4 bits. La ROM contient le programme sous forme d'opcodes, générés par du code assembleur ARM.

Le registre rF est le program counter, et pointe sur l'instruction courante.

3 Opcodes

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

0-3	4	5	6	7	8-10	11	12-15	16-19	20-23	24-27	28-31
COND	0	1	Opération sur la RAM								
COND	0	0	I	OPCODE		Set flag	Registre rd	Registre op1	Opérande op2		
COND	0	0	1	OPCODE		Set flag	Registre rd	Registre op1	Shift	Constante	
COND	0	0	0	OPCODE		Set flag	Registre rd	Registre op1	Shift		r2
COND	1	0	1	linking	Offset						
COND	1	1	1	0	Port				Valeur		

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 and op2
1001	TST	set les flags sur op1 and op2
1010	ORR	op1 or op2
1011	BIC	op1 nand op2
1100	NOT	not op2
1101	MOV	op2
1110	EOR	op1 xor op2
1111	TEQ	set les flags sur op1 xor op2

Où c représente le carry

4.2 Instruction spécifique à la montre

L'instruction WAI a été ajoutée. L'opération prends deux paramètres, un port et une valeur sur 12 bits. Le program counter reste bloqué sur l'instruction tant que le port n'envoie pas la valeur indiquée.

4.3 Barrel shifter

Le barrel shifter permet d'effectuer les opérations LSL, ASL, LSR, ASR, ROR et RRX sur l'opérande 2. Il permet aussi de faire un move avec un entier supérieur à 8 bits.

Opcode	Nom assembleur	effet
00	LSL/ASL	effectue un shift gauche
01	ROR	effectue une rotation droite
01	RRX	effectue une rotation droite avec le bit de carry
10	LSR	effectue un shift droit logique
11	ASR	effectue un shift gauche arithmétique