ARO2

Complément : Instructions

Romuald Mosqueron

Types d'instructions

- **Instructions à 3 opérandes**
  Exemple : ADD r1, r2, r3 => r1 = r2 + r3

- **Instructions à 2 opérandes**
  Exemple : ADD r1, r2 => r1 = r1 + r2

- **Instructions à 1 opérande**
  Exemple : ADD r1 Acc => Acc = Acc + r1

Types d'instructions

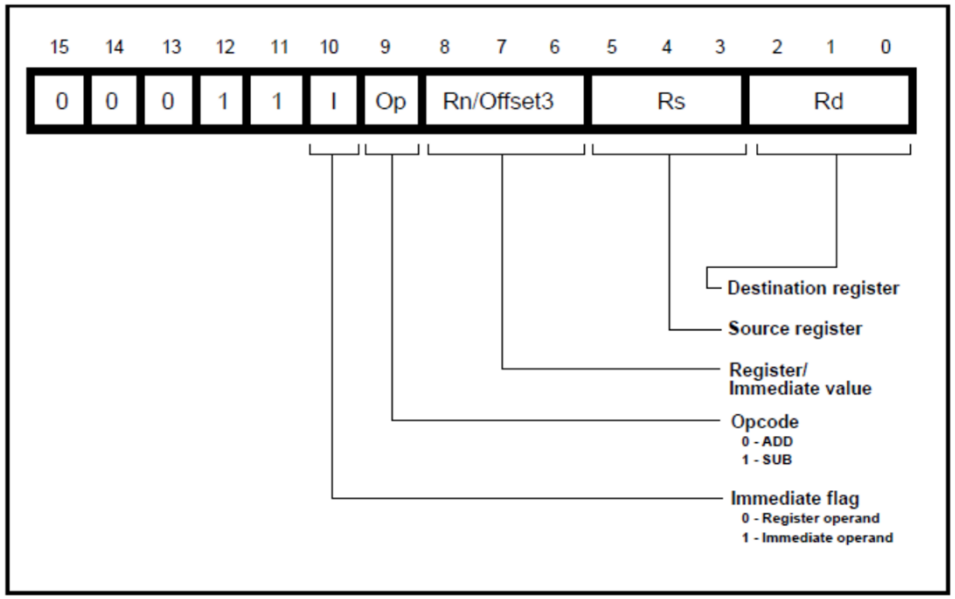
Mnemonic	Instruction	Lo register operand	Hi register operand	Condition codes set	See Section:
ADC	Add with Carry	✓		✓	5.4
ADD	Add	✓	✓	✓①	5.1.3, 5.5, 5.12, 5.13
AND	AND	✓		✓	5.4
ASR	Arithmetic Shift Right	✓		✓	5.1, 5.4
B	Unconditional branch	✓			5.16
B*xx*	Conditional branch	✓			5.17
BIC	Bit Clear	✓		✓	5.4
BL	Branch and Link				5.19
BX	Branch and Exchange	✓	✓		5.5
CMN	Compare Negative	✓		✓	5.4
CMP	Compare	✓	✓	✓	5.3, 5.4, 5.5
EOR	EOR	✓		✓	5.4
LDMIA	Load multiple	✓			5.15
LDR	Load word	✓			5.7, 5.6, 5.9, 5.11
LDRB	Load byte	✓			5.7, 5.9
LDRH	Load halfword	✓			5.8, 5.10
LSL	Logical Shift Left	✓		✓	5.1, 5.4
LDSB	Load sign-extended byte	✓			5.8
LDSH	Load sign-extended halfword	✓			5.8
LSR	Logical Shift Right	✓		✓	5.1, 5.4
MOV	Move register	✓	✓	✓②	5.3, 5.5
MUL	Multiply	✓		✓	5.4
MVN	Move Negative register	✓		✓	5.4

Table 5-1: THUMB instruction set opcodes

Mnemonic	Instruction	Lo register operand	Hi register operand	Condition codes set	See Section:
NEG	Negate	✓		✓	5.4
ORR	OR	✓		✓	5.4
POP	Pop registers	✓			5.14
PUSH	Push registers	✓			5.14
ROR	Rotate Right	✓		✓	5.4
SBC	Subtract with Carry	✓		✓	5.4
STMIA	Store Multiple	✓			5.15
STR	Store word	✓			5.7, 5.9, 5.11
STRB	Store byte	✓			5.7
STRH	Store halfword	✓			5.8, 5.10
SWI	Software Interrupt				5.17
SUB	Subtract	✓		✓	5.1.3, 5.3
TST	Test bits	✓		✓	5.4

Types d'instructions

Comment lire les instructions THUMB?



Quel est le code pour SUB R1, R3, #4?

Op	I	THUMB assembler	ARM equivalent	Action
0	0	ADD Rd, Rs, Rn	ADDS Rd, Rs, Rn	Add contents of Rn to contents of Rs. Place result in Rd.
0	1	ADD Rd, Rs, #Offset3	ADDS Rd, Rs, #Offset3	Add 3-bit immediate value to contents of Rs. Place result in Rd.
1	0	SUB Rd, Rs, Rn	SUBS Rd, Rs, Rn	Subtract contents of Rn from contents of Rs. Place result in Rd.
1	1	SUB Rd, Rs, #Offset3	SUBS Rd, Rs, #Offset3	Subtract 3-bit immediate value from contents of Rs. Place result in Rd.

Types d'instructions

- Comment interpréter l'instruction?

La valeur de l'instruction est :

0x4061

⇒ Convertir en binaire

⇒ Chercher dans le document quel est l'instruction.

  ⇒ Opcode?

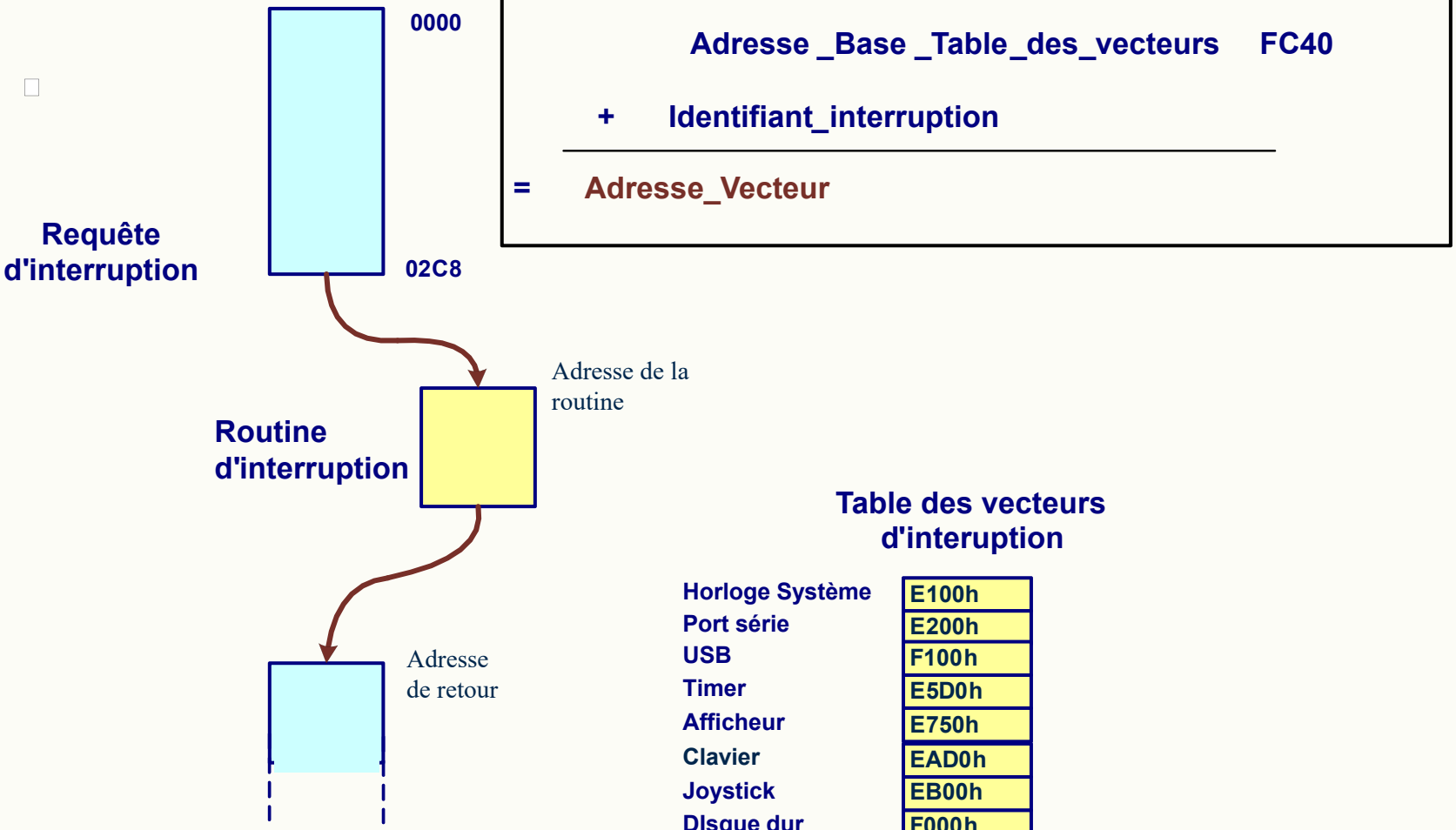
⇒ Interpréter l'instruction

Récapitulatif des divers modes d'adressage

- Adressage immédiat # valeur
  Opérande => directement la donnée
- Adressage par registre Rn
  Opérande => n° du registre où se trouve la donnée
- Adressage absolu (direct)
  Opérande => adresse mémoire de la donnée
- Adressage indirect [..]
  Opérande => adresse où se trouve l 'adresse de la donnée
- Adressage indirect par registre [Rn]
  Opérande => n° du registre qui contient l 'adresse où se trouve la donnée
- Adressage indirect par registre avec offset [Rn, #+/- offset]
  Opérande => n° du registre qui contient l 'adresse où se trouve la donnée
- Adressage par pointeur [Stack]
  Le pointeur (registre spécial) contient l'adresse de la donnée dans la pile
- Adressage par pointeur plus index [Stack + Rn]
  Index => offset contenu dans le registre Rn
- Adressage par pointeur plus index et offset [Stack + Rn + #offset]
  Index => offset contenu dans le registre Rn

Vecteurs d'interruption

- La position du vecteur d'interruption dans la table correspond à l'identifiant de la source



Vecteurs d'interruption

Analyse de l'exemple :

- La table des vecteurs d'interruption (zone dans la mémoire) contient les adresses des routines d'interruption de chacun des périphériques, classées selon les numéros affectés aux sources d'interruption.
- Ainsi le clavier est la source d'interruption 5.
- L'adresse de sa routine d'interruption est placée à la 6ème position de la table (la 1ère est la source 0).
- Lorsque l'interruption du clavier arrive, un mécanisme hardware ajoute le numéro de cette interruption (5) à l'adresse de base de la table des vecteurs (FC40, celle de la source 0) .
- Comme les adresses sont sur 32 bits dans cette exemple, on multiplie par 4 le numéro de l'interruption avant de l'additionner à l'adresse de base (FC40 + 4 x 5 = FC54)
- La table contient l'adresse de la routine d'interruption clavier (EAD0) à l'adresse FC54.

Saut conditionnel

CPSR =

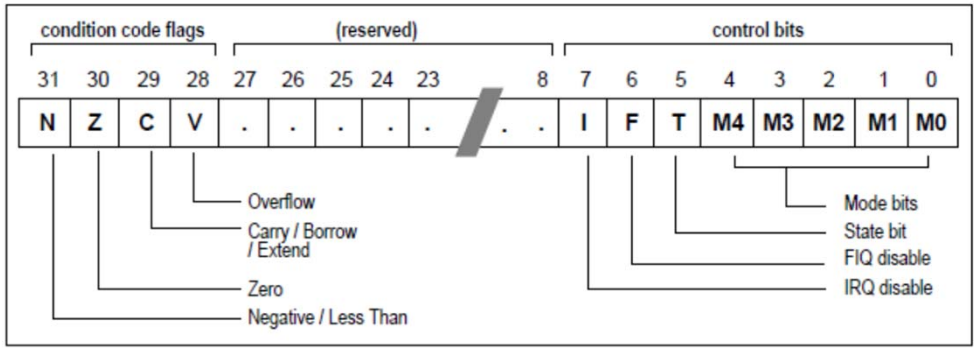


Figure 3-6: Program status register format

Instruction Branch

15	14	13	12	11 … 8	7 … 0
1	1	0	1	cond	signed_immed_8

Si « True » alors Saut à l'adresse
Si « False » alors passage à l'instruction suivante

Saut conditionnel

CPSR =

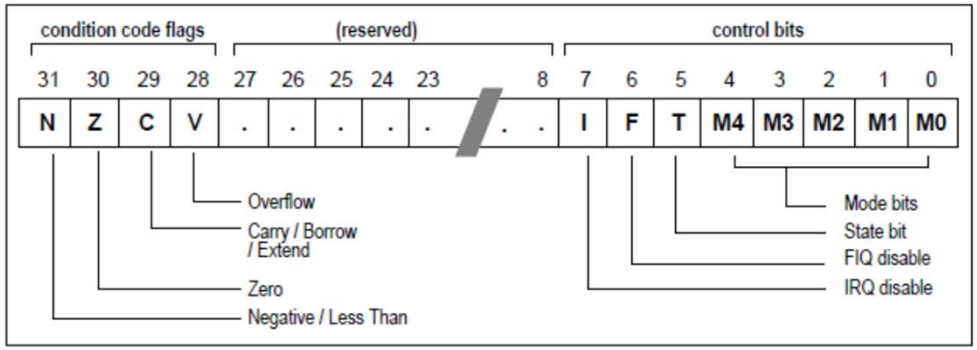


Figure 3-6: Program status register format

- **Bits de condition (N Z C V)**
  - **N = Résultat d'une opération de l'ALU négatif**
  - **Z = Résultat d'une opération de l'ALU nul**
  - **C = Carry(report) provenant de l'ALU**
  - **V = Overflow (dépassement) d'une opération de l'ALU**

Saut conditionnel

	Mnemonic extension	Meaning	Condition flag state
0000	EQ	Equal	Z set
0001	NE	Not equal	Z clear
0010	CS/HS	Carry set/unsigned higher or same	C set
0011	CC/LO	Carry clear/unsigned lower	C clear
0100	MI	Minus/negative	N set
0101	PL	Plus/positive or zero	N clear
0110	VS	Overflow	V set
0111	VC	No overflow	V clear
1000	HI	Unsigned higher	C set and Z clear
1001	LS	Unsigned lower or same	C clear or Z set
1010	GE	Signed greater than or equal	N set and V set, or N clear and V clear (N == V)
1011	LT	Signed less than	N set and V clear, or N clear and V set (N != V)
1100	GT	Signed greater than	Z clear, and either N set and V set, or N clear and V clear (Z == 0,N == V)
1101	LE	Signed less than or equal	Z set, or N set and V clear, or N clear and V set (Z == 1 or N != V)
1110	AL	Always (unconditional)	-
1111	-		-