

# Gestion de la mémoire

## ARO

2 - Instructions

## Definition

TBD

### Table des matières

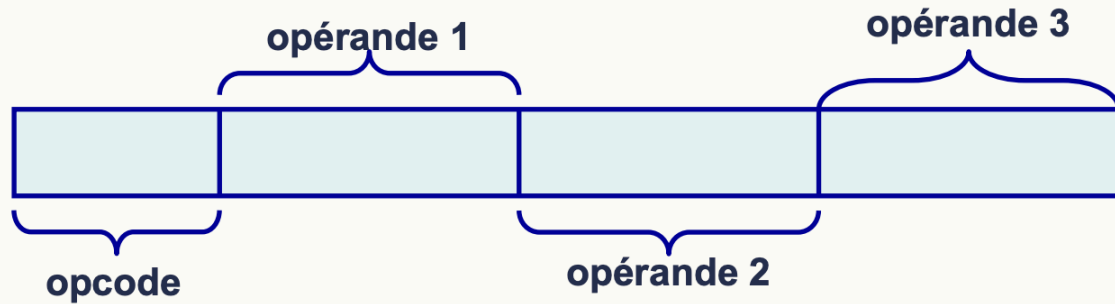
- 1. Types d'instruction ..... 2
- 2. Format d'une instruction ..... 3
  - 2.1. Opcode ..... 3
  - 2.2. Opérande(s) ..... 3
  - 2.3. Exemple(s) ..... 3
    - 2.3.1. 1 opérande ..... 3
    - 2.3.2. 2 opérandes ..... 3
    - 2.3.3. 3 opérandes ..... 3
- 3. Execution d'une instruction ..... 4
- 4. Modes d'adressage ..... 5
  - 4.1. Adressage immédiat ..... 5
  - 4.2. Adressage direct ..... 5
    - 4.2.1. Absolu ..... 5
    - 4.2.2. Par registre ..... 5
  - 4.3. Adressage indirect ..... 5
- 5. Comment interpréter une instruction ..... 7

## **1. Types d'instruction**

Il existe 3 types d'instructions que nous allons voir.

1. instructions de calcul/traitement → (arithmétique et logique)
2. instructions de transfert de donnée
  - interne → interne
  - interne → externe
3. instructions de branchement (saut)

## 2. Format d'une instruction



### 2.1. Opcode

Le champ «opcode» est l'identificateur de l'instruction et permet donc de définir ce que doit faire celle-ci.

### 2.2. Opérande(s)

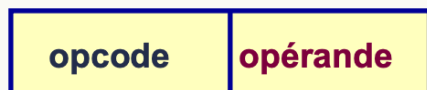
- Les champs opérandes spécifient la destination et les deux opérandes pour le traitement
  - Le nombre d'opérandes peut varier de 1 à 3.
  - Ci-dessous exemple avec le cas général

### 2.3. Exemple(s)

#### 2.3.1. 1 opérande

INC r1

Incrément de r1 de 1.



#### 2.3.2. 2 opérandes

ADD r1, r2

Addition de r1 et r2 puis stockage dans r1.



#### 2.3.3. 3 opérandes

ADD r1, r2, r3

Addition de r2 et r3 dans r1.



### 3. Execution d'une instruction

Un processeur effectue sans arrêt une boucle composée de trois phases:

1. **FETCH** → recherche de l'instruction : dans un premier temps le processeur va chercher l'instruction à exécuter en récupérant l'instruction pointée par le PC (**Program Counter**). Cette instruction sera stockée dans un autre registre du processeur, le IR (**Instruction Register**).
2. **DECODE** → décodage de l'instruction : dans un deuxième temps, chaque instruction est identifiée, grâce à un code (**opcode**). En fonction de ce code, le processeur choisit la tâche à exécuter, c'est-à-dire la séquence de micro-instructions à exécuter.
3. **EXECUTE** → exécution de l'instruction : finalement on exécute l'instruction puis revenons à la première.

Deux étapes supplémentaires seront vues dans la suite du cours qui permettront de compléter ce que fait un processeur, soit **MEMORY** et **WRITE BACK**.

## 4. Modes d'adressage

Le mode d'adressage correspond à la méthode d'accès aux données. En fonction de l'instruction à exécuter, la donnée à utiliser peut être différente. Nous pouvons imaginer quelques cas :

- Valeur immédiate: opérande = donnée
- Valeur dans un registre: opérande = no registre
- Valeur en mémoire: opérande = adresse
  - nombreuses variantes pour ce dernier cas

### 4.1. Adressage immédiat

Pour ce qui est de l'adressage immédiat nous aurons une valeur constante dans le champ d'instruction.

```
mov rd, #valeur
add rd, #cte
```

### 4.2. Adressage direct

Lors de l'adressage direct nous adressons directement l'adresse de l'opérande. De plus, il existe 2 sous-catégories d'adressage direct :

#### 4.2.1. Absolu

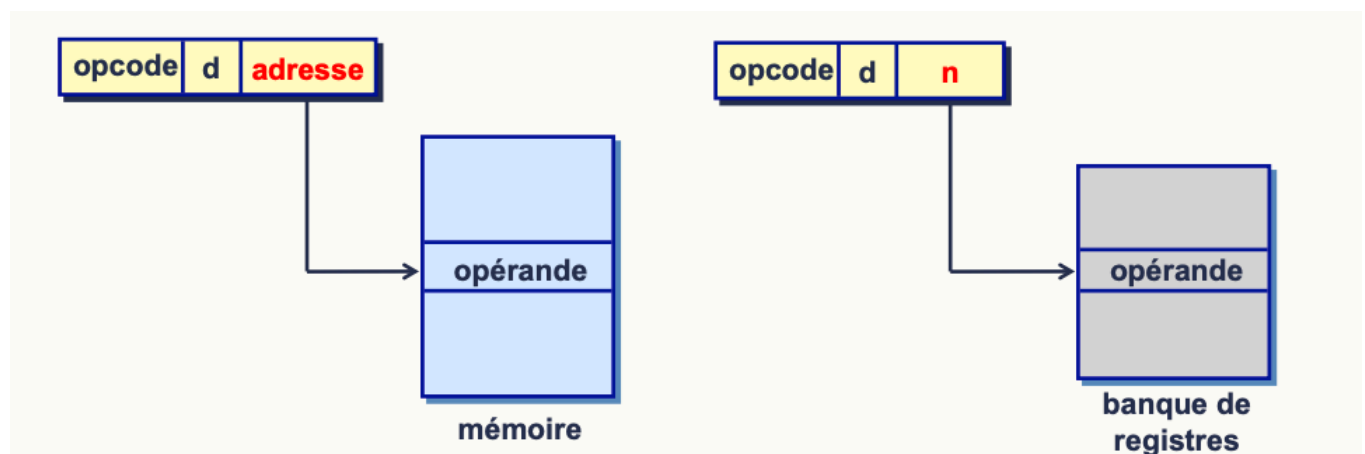
Dans ce cas, l'adresse est directement ajoutée dans le champ d'instruction.

```
mov r1, 0x1234
add r1, 0x1234
```

#### 4.2.2. Par registre

Dans ce cas, l'adresse est stockée dans un registre donc on met le numéro du registre.

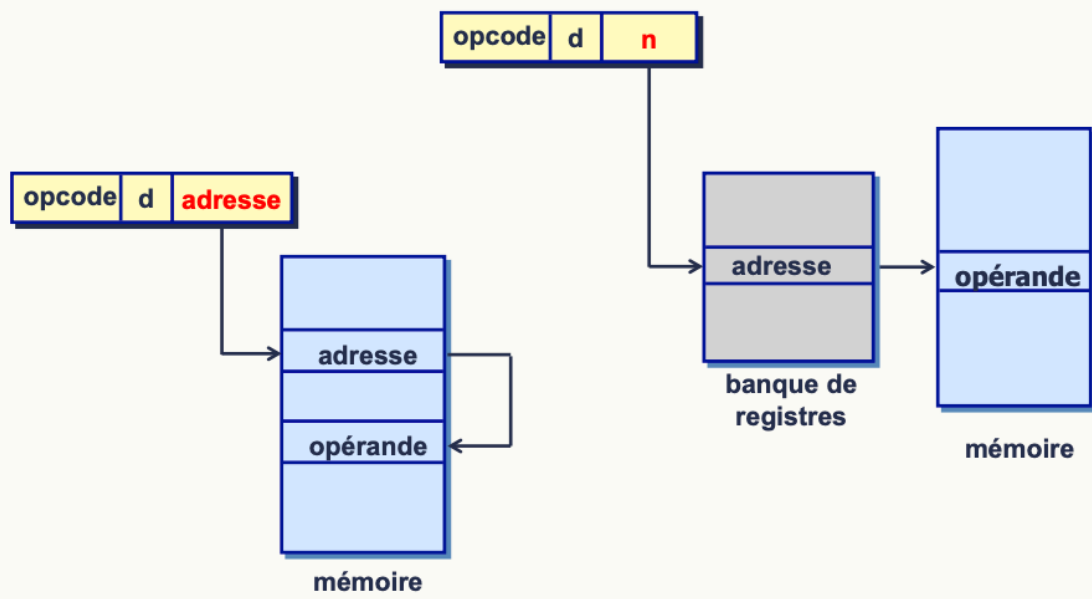
```
mov r1, r2
add r1, r2
```



### 4.3. Adressage indirect

Lors de l'adressage indirect, l'adresse de l'opérande est stockée dans un registre.

```
mov r1, [0x1234]
add r1, [2]
```



### 5. Comment interpréter une instruction

Pour interpréter une instruction, il faut suivre les étapes suivantes: Dans notre situation notre instruction vaut 0x1F19 et nous travaillons sur un processeur 16 bits.

- 1. La convertir en binaire: 0001 1111 0001 1001
- 2. Chercher dans la table des instructions l’opcode correspondant à 00011
- 3. Interpreter les opérandes en fonction de l’opcode trouvé.

1514131211109876543210

000111OpRn/Offset3RsRd

Destination register

Source register

Register/ Immediate value

Opcode  
0 - ADD  
1 - SUB

Immediate flag  
0 - Register operand  
1 - Immediate operand

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.

SUB R1, R3, #4