Fetch

ARO

3 - Fetch

Definition

La documentation suivante traite du fonctionnement du registre de compteur de programme (PC) dans les architectures informatiques. Elle explique comment le PC stocke l'adresse de l'instruction en cours d'exécution, comment il est incrémenté et géré lors de sauts inconditionnels ou conditionnels. De plus, elle aborde la gestion des sauts, des interruptions et des vecteurs d'interruption dans les processeurs.

Table des matières

1. Opération FETCH

L’opération de fetch (récupération) dans un processeur consiste à récupérer l’instruction suivante depuis la mémoire, en utilisant l’adresse stockée dans le compteur de programme (PC). Cette instruction est ensuite placée dans le registre d’instruction (IR) pour être décodée et exécutée ultérieurement.

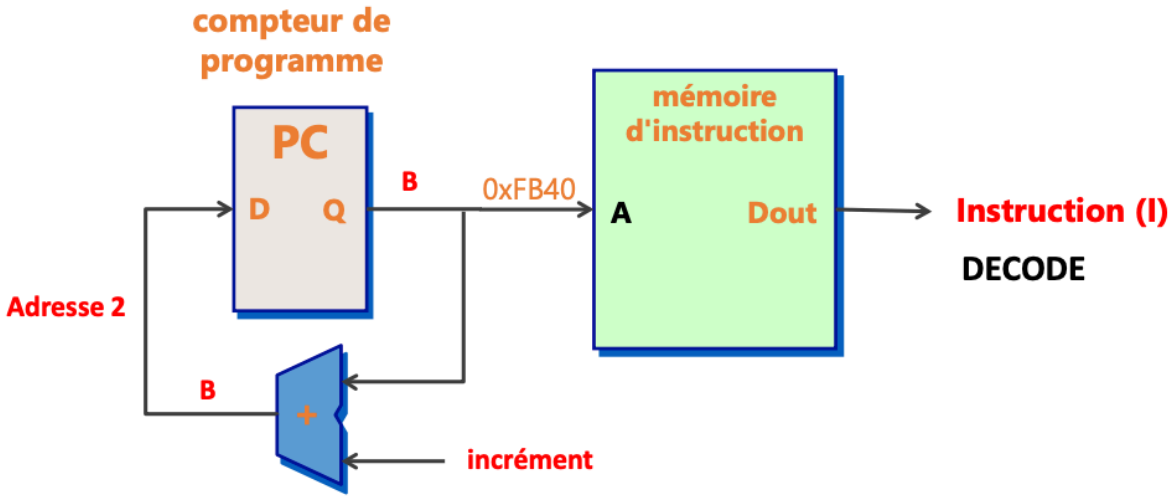
2. Program Counter (PC)

Le programme counter (PC) est un registre qui contient l'adresse de l'instruction courante à éxécuter. Il est incrémenté à chaque instruction pour passer à l'instruction suivante **sauf si un saut est effectué**.

2.1. Incrémenter le PC

La valeur de l'incrément correspond au nombre de bytes de l'instruction, indépendamment du bus d'adresses!

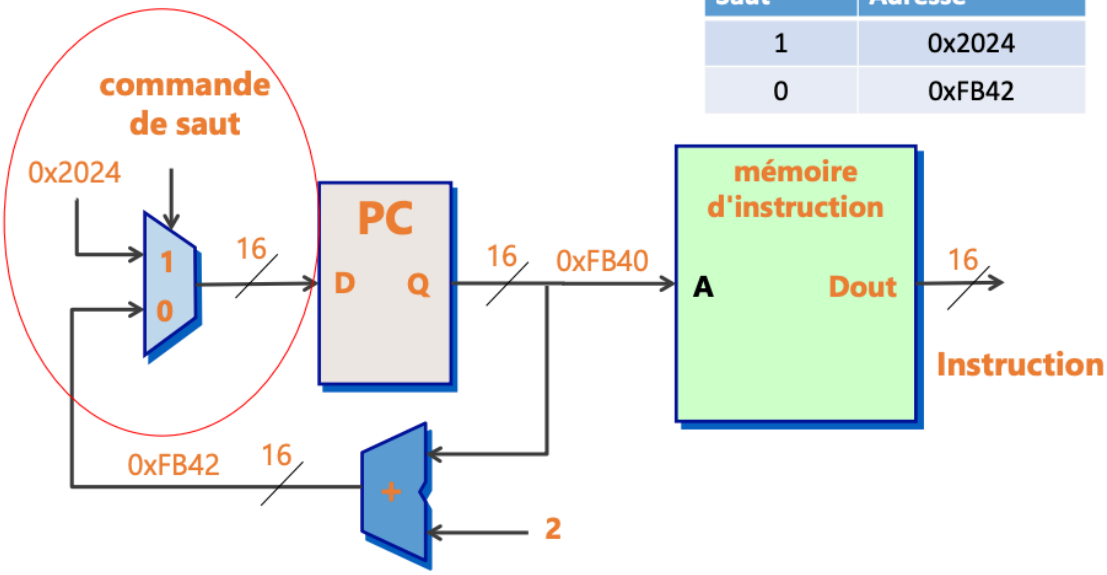
Code d'instruction	Incrément
8 bits = 1 byte	1
16 bits = 2 bytes	2
32 bits = 4 bytes	4



2.2. Adresse suivante avec un saut

Dans le cas ou un saut doit être réalisé l'incrément de l'adresse est quand même réalisé sauf que celle-ci n'est pas insérée dans le PC à la place on viendra y insérer l'adresse de la prochaine instruction suite au saut.

Saut	Adresse
1	0x2024
0	0xFB42



12

3. Gestion des sauts

Les sauts sont des instructions qui permettent de modifier le PC pour exécuter une instruction à une adresse différente de celle suivante. Il existe plusieurs types de sauts:

3.1. Déclanché par une instruction

- Saut inconditionnel
  - Exemple : Retour au début d'un programme
- Saut conditionnel
  - Exemple : Exécution d'une boucle
- Appel de fonction

3.2. Déclanché matériellement

- Interruption
- Exception

3.3. Saut inconditionnel

L'opcode pour un saut inconditionnel prends 5 bits et le reste est alloué pour donner l'adresse de la prochaine instruction à exécuter.

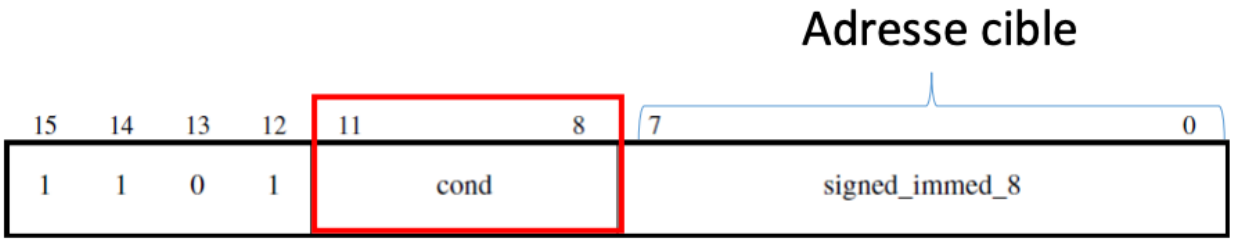
3.3.1. Calcul de l'adresse de saut

Pour calculer l'adresse de saut il suffit d'utiliser la formule suivante :

$$\text{Adr} = \text{PC} + \text{extension\_16bits}(\text{offset}_{11} * 2) + 4$$

Code d'instruction	Incrément
Adr	Adresse finale du saut
PC	Adresse de l'instruction courante
Extension 16 bits	Extension de l'adresse de saut en y ajoutant la valeur du bit de signe
Offset	Correspond à l'instruction moins les 5 bits de l'opcode
4	Valeur en fixe à ajouter à l'adresse de saut

3.4. Saut conditionnel



Pour calculer l'adresse de saut il suffit d'utiliser la formule suivante **attention elle est légèrement différente de celle pour le saut inconditionnel** :

$$\text{Adr} = \text{PC} + \text{extension\_16bits}(\text{offset}_{8} * 2) + 4$$

3.4.1. Gestion de la condition

	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	-		-

- **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**

4. Interruption

Les interruptions sont des signaux envoyés par des périphériques pour signaler un événement. Lorsqu'une interruption est reçue, le processeur suspend l'exécution du programme en cours et exécute une routine de traitement de l'interruption. Une fois la routine terminée, le processeur reprend l'exécution du programme interrompu. Il existe plusieurs type d'interruption:

- **Interruption par périphériques**
  - timer, clavier, entrées /sorties (port série, parallèle, USB, disques, ...)
- **Erreurs systèmes**
  - division par zéro, dépassement de capacité, protection mémoire, ...
- **Interruption processeur (IPI)**
  - interruption par un autre processeur (système multi-processeurs)

4.1. Gestion des interruptions

Il existe 5 modes de gestion des interruptions:

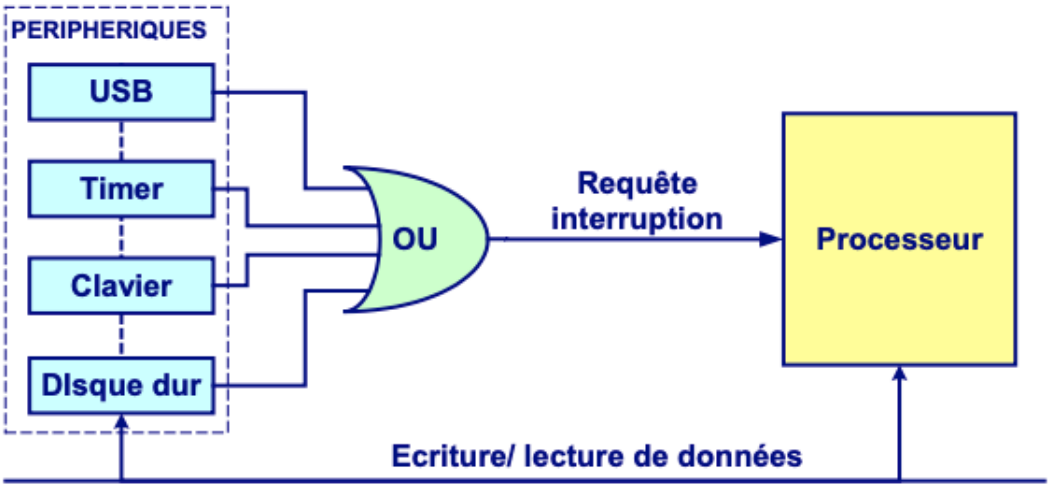
- Sans interruption (scrutation)
- Interruptions multi-sources sans identification de la source
- Interruptions multi-sources avec identification de la source
- Interruptions chaînées
- Interruptions imbriquées

4.1.1. Scrutation

Le processeur scrute périodiquement l'état des périphériques pour savoir s'ils ont besoin de son intervention. Cette méthode est simple mais consomme beaucoup de ressources.

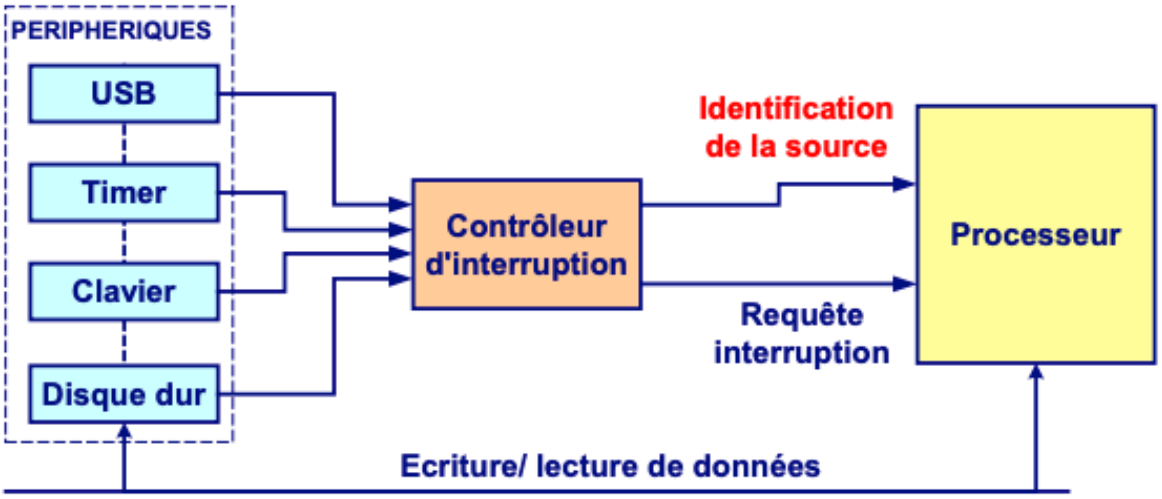
4.1.2. Interruptions multi-sources sans identification de la source

Les interruptions sont gérées par un seul vecteur d'interruption. Le processeur doit interroger chaque périphérique pour savoir lequel a généré l'interruption.



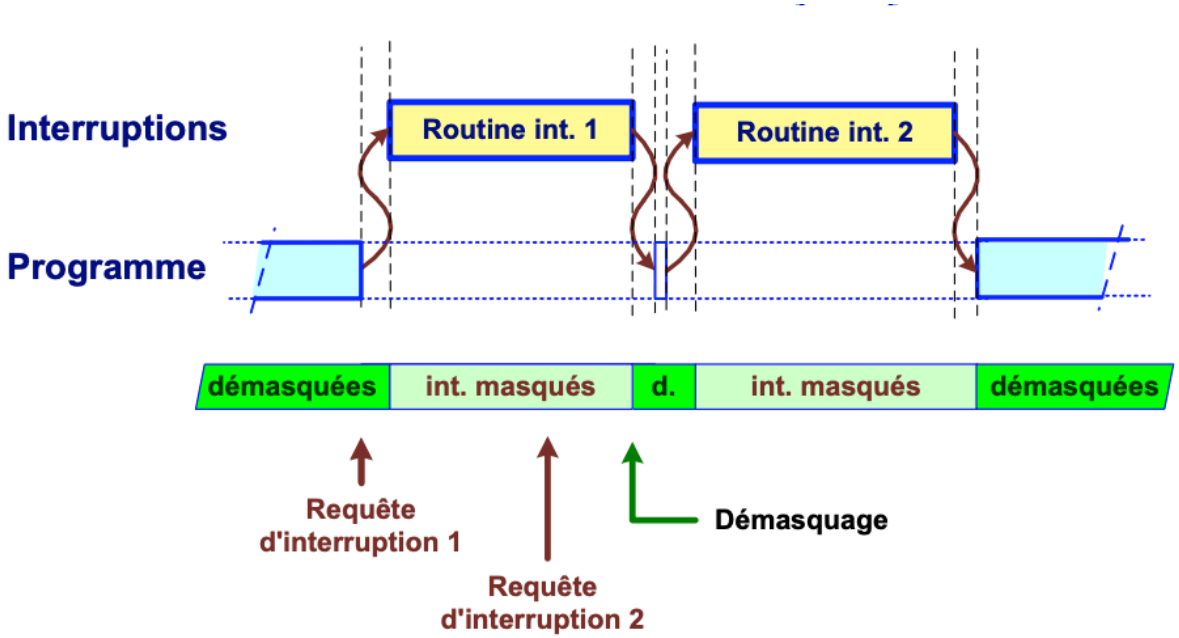
4.1.3. Interruptions multi-sources avec identification de la source

Chaque périphérique possède un vecteur d'interruption. Le processeur interroge chaque périphérique pour savoir lequel a généré l'interruption.



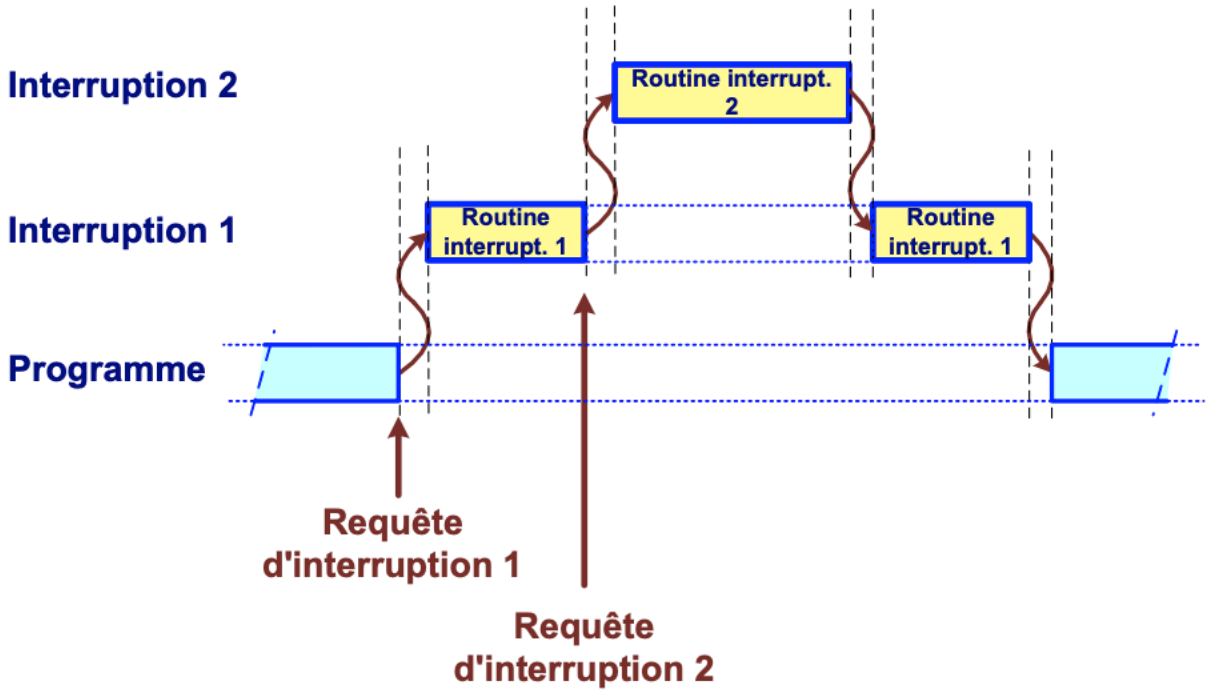
4.1.4. Interruptions chaînées

Les adresses des interruptions sont stockées dans une pile de stockage (stack) puis déplié au fur et à mesure de l'exécution des routines d'interruption.



4.1.5. Interruptions imbriquées

Les interruptions sont gérées par un contrôleur d'interruption qui attribue une priorité à chaque interruption. Lorsqu'une interruption est reçue, le contrôleur d'interruption détermine si elle doit être traitée ou non.



5. Vecteur d'interruption

Moyen technique pour attacher une routine d'interruption (ISR) à une requête d'interruption (IRQ). Le vecteur d'interruption est une table qui contient l'adresse de la routine d'interruption associée à chaque requête d'interruption. Lorsqu'une interruption est reçue, le processeur consulte le vecteur d'interruption pour connaître l'adresse de la routine d'interruption à exécuter.

5.1. Calculer l'adresse du vecteur d'interruption

$$\text{Adresse vecteur} = \text{Adresse base} + (\text{No de l'interruption} * \text{Nbre bytes du bus d'instruction})$$