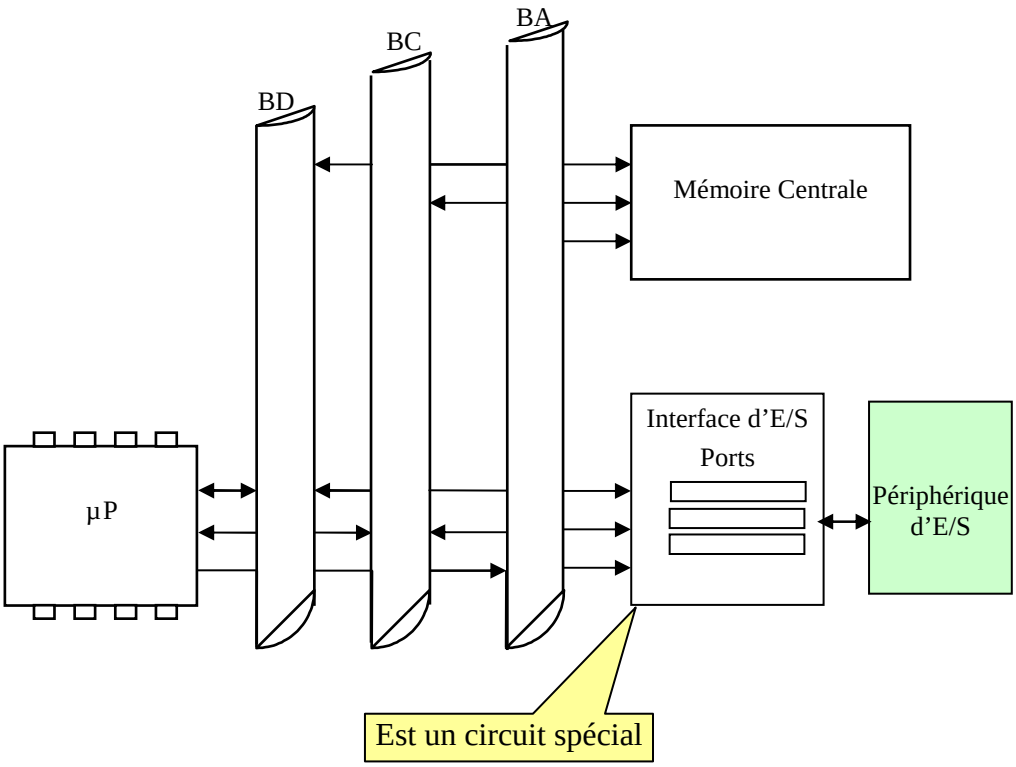
Cours Architecture des ordinateurs (Archi II)
Entrées / Sorties

Resp. Mr Mohamed Feredj – MCA –
Courriel : archiFeredj@gmail.com

1) Objectif

Discuter comment les données sont transférées entre le µP et les périphériques.



Tout transfert de données entre la mémoire et les périphériques d'E/S doit passer par des interfaces d'E/S.

2) C'est quoi une interface d'E/S

1. C'est un circuit spécial placé entre le µP et le périphérique d'E/S.
2. Utilisée pour résoudre les problèmes de distance et d'incompatibilité.
   a - Incompatibilité => Caractéristiques des périphériques ≠ Celles du µP/MC (Vitesse, timing, etc.)
   b - Si Distance de propagation des signaux est grande ➔ retard augmente ➔ synchronisation entre bus non assurée.

3) Les ports du circuit d'interface d'E/S (CI d'E/S) :

Les ports du CI d'E/S sont vus comme des registres (petits espaces mémoires à l'intérieur du CI d' E/S). Tout CI d'E/S comporte 3 types de ports :

3.1) Ports de données :

Pour l'envoi et la réception des données entre le µP et le CI d'E/S ;
C'est port en lecture et écriture

3.2) Ports de contrôle (ou commande):

Pour définir (fixer) le mode de fonctionnement du CI d'E/S;
C'est un port en écriture

3.3) Ports d'état :

Il fournit les informations sur l'état du CI d'E/S.
C'est un port en lecture.

4) Instructions de manipulation des ports du CI d'E/S :

La manipulation des ports revient à envoyer et/ou récupérer des données aux/depuis les ports.

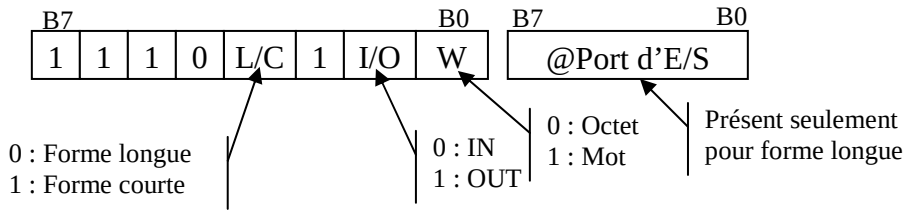
On n'utilise que 2 instructions :

- ***IN*** : Pour récupérer Donnée ;
- ***OUT*** : Pour envoyer Donnée.

NOM	MNEMONIQUE	DESCRIPTION
IN (Lecture des ports) :		
Forme longue: Octet	IN AL, Port	(AL) ⇐ (Port)
Forme longue: Mot	IN AX, Port	(AX) ⇐ (Port+1:Port)
Forme courte: Octet	IN AL, DX	(AL) ⇐ ((DX))
Forme courte: Mot	IN AX, DX	(AX) ⇐ ((DX))
OUT (Ecriture dans ports) :		
Forme longue: Octet	OUT Port, AL	(Port) ⇐ (AL)
Forme longue: Mot	OUT Port, AX	(Port+1:Port) ⇐ (AX)
Forme courte: Octet	OUT DX, AL	((DX)) ⇐ (AL)
Forme courte: Mot	OUT DX, AX	((DX)) ⇐ (AX)

Remarque : *Attention, on n'utilise que les registres AL ou AX et DX pour manipuler les ports d'E/S. En effet, par exemple,* ***IN AL, BX non autorisée***

Code machine



Temps d'exécution des instructions d'E/S :

Forme Longue : 10 CM (Cycle Machine)
Forme Courte : 08 CM

5) La réalisation des opérations d'E/S :

Dans cette partie, on montre qu'il y'a plusieurs techniques permettant la réalisation d'E/S :

5.1) Les E/S directes programmées / avec attente de disponibilité (E/S directe synchrone)

Cette technique exige que le µP suive le déroulement du transfert des données du début à la fin : Donc, après avoir lancé une opération d'E/S, le µP consulte continuellement le port d'état du CI d'E/S jusqu'à ce que ce dernier signal la fin de l'opération.

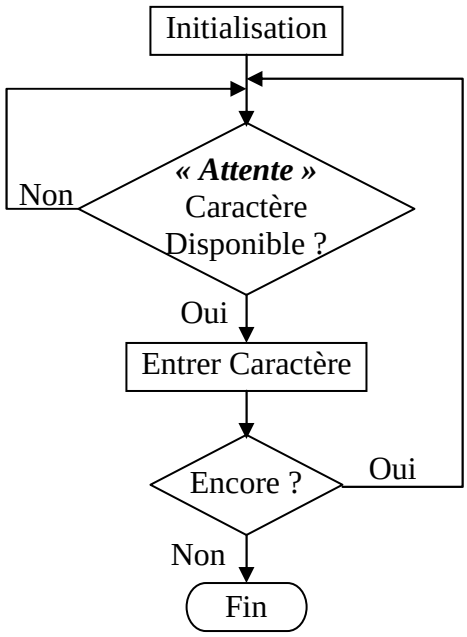
Exemple : Entrée programmée d'une suite de caractères à partir d'un périphérique :

Prog assembleur

```
; On suppose que le 4ème bit du port d'état
; c'est lui qui indique la présence (1) ou non
; (0) d'un caractère sur le port de données

        MOV CX, 0AH
Attente : IN AL, Port_Etat
        TEST   AL,  08H
        JZ Attente
        IN AL, Port_Donnee
        …….
        DEC CX
        JNZ Attente
```

Organigramme



Inconvénient : ➔ La consultation continue du bit d'état mène à *Attente active*

5.2) Les d'E/S contrôlées par interruption

5.2.1) Les d'E/S directes

- Cette technique permet au µP d'éviter l'attente inactive.
- Donc, le µP n'a qu'à envoyer les données au CI d'E/S et par la suite peut faire d'autres tâches en attendant la fin du transfert ➔ Transfert de données se fait en parallèle à l'exécution d'autres tâches.

- Une fois le transfert des données est terminé, le µP est informé par un signal d'interruption.

Donc, les étapes du mode d'E/S par interruption sont :

1. Le µP ordonne le CI d'E/S de lancer le transfert de données et puis va faire d'autres tâches en parallèle du transfert de données. Le transfert peut être pour l'entrée ou la sortie des données ;

2. Le CI d'E/S lance le transfert physique des données ;

3. Une fois que le transfert physique des données est terminé, le CI d'E/S envoie une demande d'interruption au µP ;

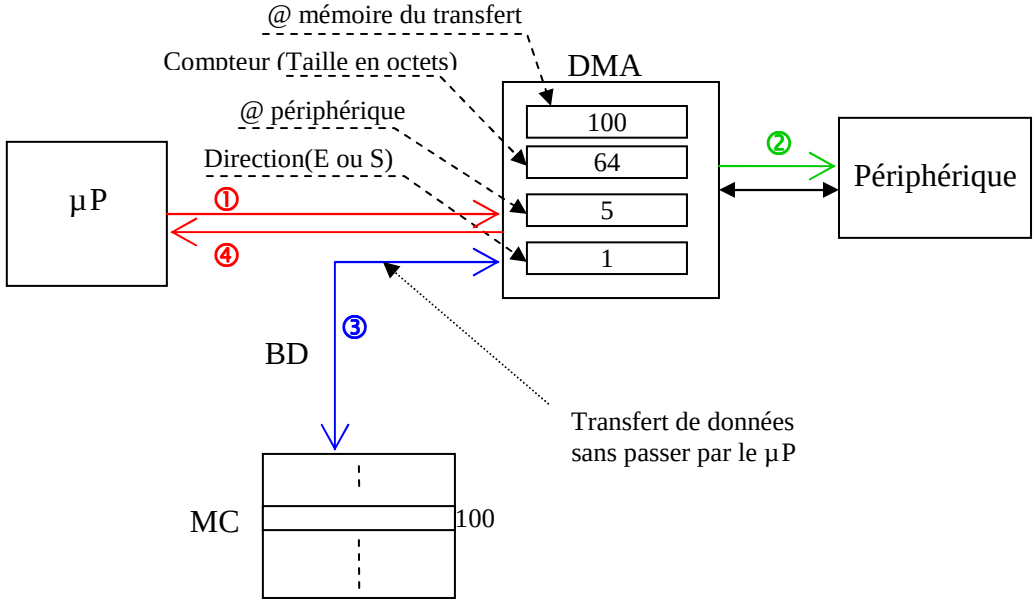
4. Le µP traite la demande d'interruption suivant les étapes qu'on a vu dans le chapitre Interruptions.

Inconvénient : ➔ Après chaque transfert physique de données depuis/vers le Port de données, le µP est interrompu ➔ Traitement d'1 It par octet/mot échangé.

5.2.2) E/S indirecte asynchrone (Cas du DMA)

La DMA permet d'éviter l'inconvénient précédent. De plus, les données à transférer ne passent plus par le µP.

Les étapes de ce mode sont :

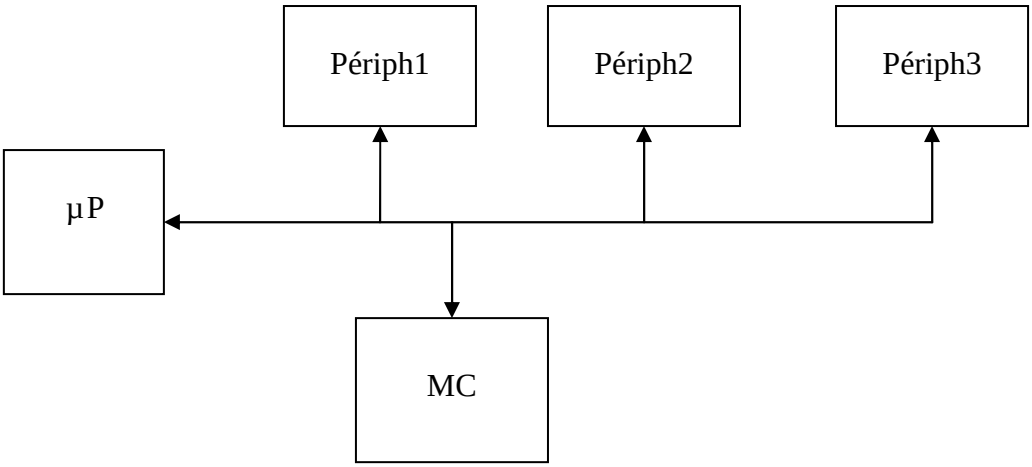


① Le µP positionne les ports (registres) de la DMA en lui indiquant l'@ mémoire dans/depuis laquelle les données seront sauvegardées/envoyées, la taille des données à transférer, l'@ du périphérique d'E/S et la direction (entrée ou sortie) ;

② + ③ La DMA lance le transfert physique des données sur le périphérique comme suit :
- 2.1 Lancer le transfert du mot
- 2.2 Décrémenter @ mémoire
- 2.3 Décrémenter Compteur
- 2.4 Si Compteur > 0 alors aller à 2.1

④ Envoyer demande d'interruption au µP pour lui signaler la fin de l'opération d'E/S.

Application :
Soit le système suivant :



Et supposons qu'on a trois procédure ProcInput1, ProcInput2 et ProcInput3 dont chacune est dédiée à un périphérique et qui effectue un traitement après la présence d'une donnée sur le port donnée.

De plus, une opération d'entrée est réalisée suivant la méthode E/S programmées (E/S directes synchrones). Donc,
1. le µP consulte le $3^{ème}$ bit (B3) du port d'état du périphérique i ;
2. Si B3 = 1 alors il appelle ProcInputi. Ensuite, dans les 2 cas (B3=1 ou 0), le µP passe au périphérique suivant (à tour de rôle).
3. Après avoir effectué les opérations d'entrée sur les 3 périphériques, le µP consulte une variable Flag pour savoir s'il doit répéter (Flag = 0) ou terminer (Flag = 1) les opérations d'entrée.

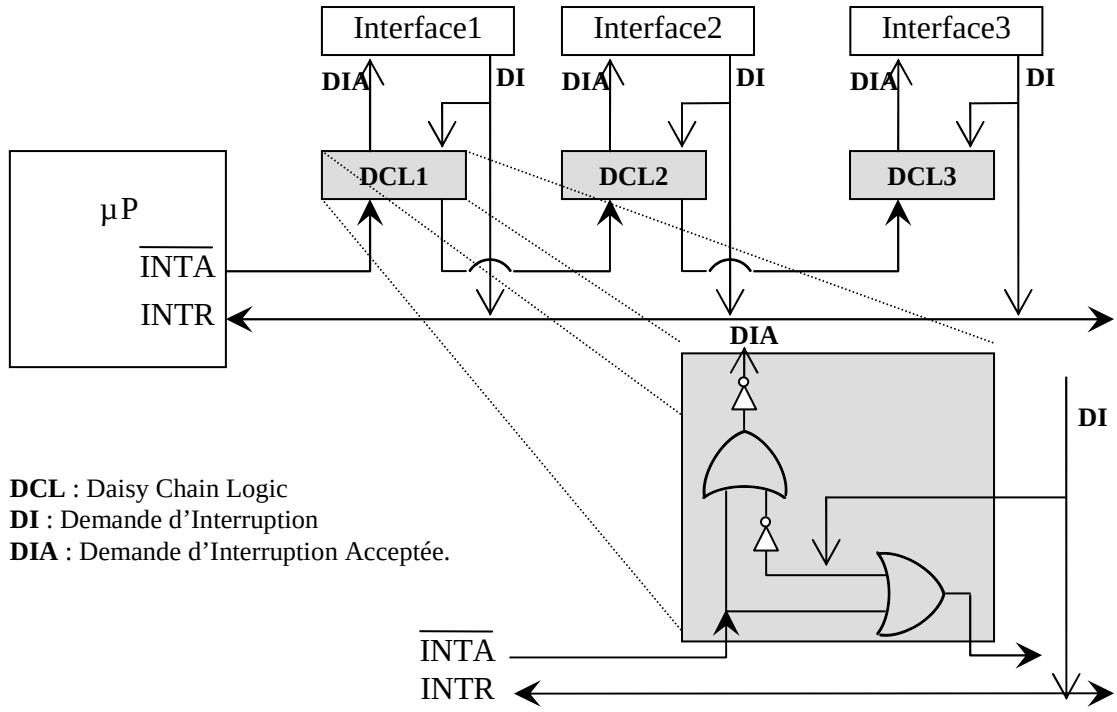
Ecrire le programme assembleur permettant la réalisation des opérations d'entrée.

5.2.3) Comment identifier la source d'une demande d'interruption ?
Il existe différentes méthodes de détection de la source d'une interruption :

a) Par Polling (par sondage) : Solution logicielle
Cette technique utilise un programme défini au préalable pour de déterminer les priorités des interruptions.

- Est adéquate pour un petit nombre de sources d'interruption, sinon le temps d'identification des sources devient très important.

b) Par Daisy Chain (Interruption chaînées ou priorités chaînées) : Solution matérielle
Cette technique utilise un composant matériel (contenant des portes logiques) pour identifier la source d'interruption.



DCL : Daisy Chain Logic
DI : Demande d'Interruption
DIA : Demande d'Interruption Acceptée.

Avantages :

- Pas de programmation pour déterminer la priorité d'une interruption ;
- Réalisation très simple : Nécessite des bloc logiques très simples à concevoir ;
- Ajout et suppression d'autres sources d'interruption est très simple.

Inconvénients :

- On ne peut pas modifier l'ordre de priorité des interruptions par programmation.

b) Par Contrôleur Programmable d'Interruption
Il sera traité dans le chapitre suivant.