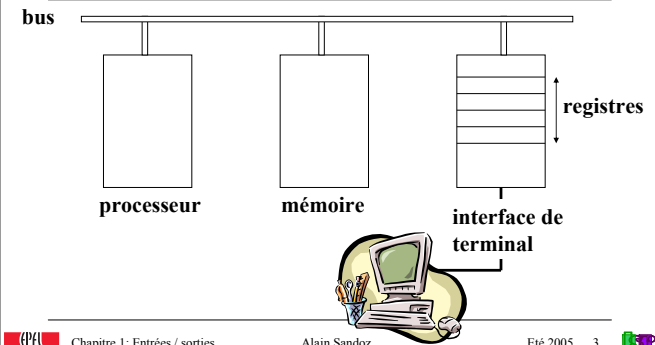


# Entrées / sorties et interruptions

## Programmation des entrées sorties

1. Entrées-sorties en mode boucle d'attente
2. Entrées-sorties en mode interruption
3. Masquage des interruptions
4. Entrées-sorties en mode DMA (Direct Memory Access)

## Architecture du système



## Comment le processeur peut-il lire ou écrire les registres de l'interface?

- soit en utilisant un espace d'adresse spécial pour les entrées-sorties (bus spécial pour accéder aux interfaces);
- soit en utilisant pour les entrées-sorties le même espace d'adresse que pour l'accès à la mémoire (*memory mapped I/O*).

**Solution 1 (Intel): le processeur doit disposer d'instructions spéciales pour accéder aux registres de l'interface.**

**Solution 2 (Motorola): les instructions d'accès à la mémoire sont utilisées pour accéder aux registres d'interface et les adresses correspondant à la mémoire sont différentes des adresses correspondant aux registres d'interface.**

## Ecriture en mode boucle d'attente

L'écran dispose de deux registres:

- dans le *registre d'état* (par exemple à l'adresse 177564), le bit 7 indique si l'interface est prête ( $I = \text{oui}$ ;  $0 = \text{non}$ );
- le *registre de donnée* (par exemple à l'adresse 177566).

7

Pour écrire:

- attendre que le bit 7 du registre d'état soit à 1;
- écrire le caractère dans le registre de donnée (l'écriture du registre de donnée remet le bit 7 du registre d'état à 0).

## Ecriture en mode interruption

- dans le registre d'état de l'écran (adresse 177564):  
le bit 7 indique si l'interface est prête ( $I = \text{oui}$ ;  $0 = \text{non}$ );  
le bit 6 permet de contrôler le mode interruption ( $I = \text{interr}$ ;  $0 = \text{pas interr}$ ).
- registre de donnée de l'écran à l'adresse 177566.

7 6

Pour écrire:

- enclencher le mode interruption (écrire 1 dans bit 6 du registre d'état);
- *lors de l'interruption*: écrire le caractère dans le registre de donnée.

## Lecture en mode boucle d'attente

- dans le registre d'état du clavier (par exemple à l'adresse 177560), le bit 7 indique si l'interface est prête ( $I = \text{oui}$ ;  $0 = \text{non}$ ).
- registre de donnée du clavier (à l'adresse 177562):

7

Pour lire:

- attendre que le bit 7 du registre d'état soit à 1;
- lire le caractère dans le registre de donnée (la lecture du registre de donnée remet le bit 7 du registre d'état à 0).

Remarque: un caractère *c* lu apparaît à l'écran parce que *c* est écrit après avoir été lu !

## Lecture en mode interruption

- dans le registre d'état du clavier (adresse 177560):  
le bit 7 indique si l'interface est prête ( $I = \text{oui}$ ;  $0 = \text{non}$ ),  
le bit 6 permet de contrôler le mode interruption ( $I = \text{interr}$ ;  $0 = \text{pas interr}$ ).
- registre de donnée du clavier (adresse 177562)

7 6

Pour lire:

- enclencher le mode interruption (écrire 1 dans bit 6 du registre d'état);
- *lors de l'interruption*: lire le caractère dans le registre de donnée.

- 
- quand l'interruption survient-elle?
  - quel est le code exécuté au moment de l'interruption?
  - qu'est-ce qui assure qu'une deuxième interruption ne survient pas durant le traitement de la première interruption?

---

Quel est le code exécuté au moment de l'interruption?

- à chaque *source d'interruption* est associé un emplacement en mémoire appelé *vecteur d'interruption* (un vecteur d'interruption pour les interruptions liées au clavier, un vecteur d'interruption pour les interruptions liées à l'écran, etc.);
- le vecteur d'interruption *indique l'adresse* du code à exécuter lors de l'interruption;
- ce code se termine par une instruction qui retourne le contrôle à l'adresse où l'exécution a été interrompue (cette adresse est généralement mémorisée sur la pile).

---

Quand l'interruption survient-elle?

(par exemple)

- lorsque le bit 7 est à 1, et que le bit 6 passe de 0 à 1 (l'interface est prête au moment de l'enclenchement du mode interruption);
- lorsque le bit 6 est à 1, et que le bit 7 passe de 0 à 1 (lorsque l'interface devient prête).

---

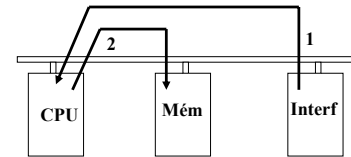
Qu'est-ce qui assure qu'une deuxième interruption ne survient pas durant le traitement de la première interruption?

- grâce au mécanisme de *masquage des interruptions*;
  - une priorité fixe est associée à chaque source d'interruption (par exemple 4 pour une interruption du clavier ou de l'écran);
  - une priorité variable est associée au processeur (par exemple entre 0, priorité minimale, et 7, priorité maximale). Cette priorité est définie par logiciel, par une écriture dans le *registre d'état du processeur* (processor status word).
- une interruption survient seulement si  
 $la\ priorité\ processeur < la\ priorité\ de\ l'interruption$
- si  $la\ priorité\ processeur \geq la\ priorité\ de\ l'interruption$ , alors l'interruption est *masquée*. Elle surviendra dès que la priorité du processeur devient inférieure à la priorité de l'interruption.

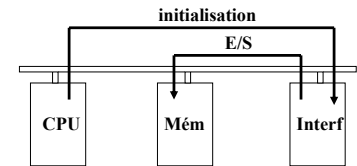
### Principe d'application:

- le processeur s'exécute normalement à la priorité 0;
- le code qui traite une interruption de priorité  $x > 0$  s'exécute avec le processeur à la priorité  $x$  ;
- cela garantit qu'aucune interruption de la même source ne peut interrompre le processeur durant le traitement d'une interruption.

### E/S standard:



### E/S DMA



## E/S en mode DMA (Direct Memory Access)

### E/S normale (non DMA):

le processeur exécute une (suite d') instruction(s) pour lire chaque octet lu ou pour écrire chaque octet.

### E/S DMA:

le processeur initialise l'E/S et il est informé (par interruption) de la fin de l'E/S. L'interface est capable d'écrire directement en mémoire.

Exemple: lecture depuis le disque. L'initialisation consiste à:

- indiquer l'adresse disque du 1<sup>er</sup> octet à lire;
- indiquer l'adresse mémoire pour le 1<sup>er</sup> octet lu;
- indiquer le nombre d'octets à lire;
- lancer le transfert.