# Architecture des Ordinateurs
## Partie II : Microprocesseur

**3. Interruptions et DMA**

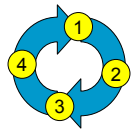
David Simplot
simplot@fil.univ-lille1.fr

## Objectifs

- 1. Mémoire et entrées/sorties
- 2. Instructions machines
- **3. Interruptions et DMA**
- 4. Microprogrammation

- Gestion des interruptions ?

- Quatre phases du µP
  - 1 = Lire
  - 2 = Décoder
  - 3 = Exécuter
    - Args + exécution
  - 4 = Préparer l'instruction suivante
    - ? → interruptions






## Au sommaire...

- **Introduction**
- Interruptions matérielles
- Interruptions logicielles
- Direct Access Memory
- Exemple d'implémentation



## Introduction (1/2)


- INT / DMA …
- Késako ?
  - INT = Interruptions
  - DMA = Direct Memory Access
    - Accès direct en mémoire

- Mécanisme d'interruption
  - Déroutements
    - Lors de l'exécution des instructions
  - Interruptions matérielles
    - Gestion des périphériques
  - Interruptions logicielles
    - Appels système



## Introduction (2/2)

- Déroutements :
  - Débordement numérique
  - Division par zéro
  - Non-reconnaissance d'une instruction
  - Accès à la mémoire illégale
  - …
- Interruptions matérielles
  - Demande de données d'un périphérique
  - Alerte provenant du matériel (batterie faible)
- Interruptions logicielles
  - Appels de fonctions du système d'exploitation
  - Fonctionne sur le même principe que les deux précédentes, mais sont générées par une instruction spécifique : INT

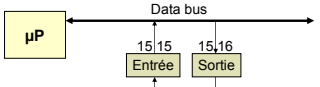


## Au sommaire...

- Introduction
- **Interruptions matérielles**
- Interruptions logicielles
- Direct Access Memory
- Exemple d'implémentation

# Interruptions matérielles (1/7)

- Ex. Programmation de régulation de la température




```
debut:  mov R1, [1515]      ; prise de la température
        cmp R1, 50          ; suffisamment basse ?
        jl stop
        mov [1516], 1       ; on met en route le ventilateur
        jmp debut
stop:   mov [1516], 0       ; on arrête le ventilateur
        jmp debut
```



# Interruptions matérielles (2/7)

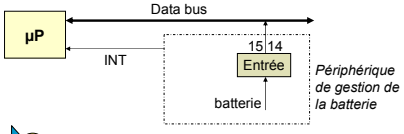
- On introduit un pb supplémentaire :
  - Prise en compte du niveau de la batterie
  - Première méthode → scrutation active (comme pour la t°)

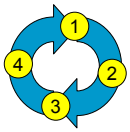
```
debut:  mov R1, [1515]      ; prise de la température
        cmp R1, 50          ; suffisamment basse ?
        jl stop
        mov [1516], 1       ; on met en route le ventilateur
        jmp suite
stop:   mov [1516], 0       ; on arrête le ventilateur
suite:  mov R1, [1414]      ; niveau de la batterie
        cmp R1, 5           ; moins de 5%
        jg debut
        call AURG           ; arrêt d'urgence
        jmp debut
```



# Interruptions matérielles (3/7)

- Deuxième méthode :

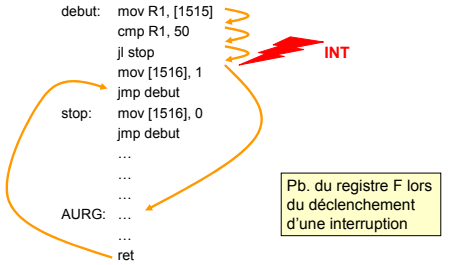



- Si à la phase 4 il y a INT, il ne faut pas exécuter l'instruction suivante, mais un programme particulier
  - Ici le programme particulier, c'est la sous-routine AURG



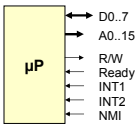
# Interruptions matérielles (4/7)






# Interruptions matérielles (5/7)

- Suivant le µP, il peut y avoir plusieurs lignes d'interruptions
  - INT1, INT2, … (généralement 2 ou 3)
- Il y existe des interruptions non-masquable
  - NMI = Non-Maskable Interruption

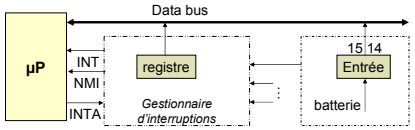



- Les autres interruptions sont donc masquables
  - Instructions DI (disable int.) et EI (enable int.)
  - Utile pour les sections critiques et initialisations



# Interruptions matérielles (6/7)

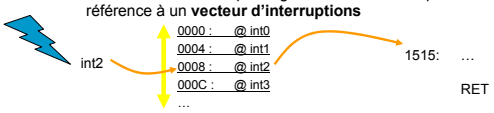
- Comment gérer tous les périphériques avec seulement 2 ou 3 signaux d'interruptions ?




- Le registre contient les informations sur l'interruption à traiter
  - La première instruction du programme de traitement des interruptions va donc être de lire ce registre sur le bus de données.
  - Pour éviter les conflits sur le bus de données, cette information n'est mise sur le bus que sur envoi du signal INTA

# Interruptions matérielles (7/7)

- L'information donnée par le gestionnaire d'interruption fait référence à un **vecteur d'interruptions**




- Au choix :
  - Implémentation soft :
    - Un seul branchement d'INT lors de la phase 4 qui lui va consulter le vecteur d'interruption
  - Implémentation hard :
    - Lors de la phase 4, on consulte le registre d'interruption et on fait le bon branchement…



# Au sommaire...

- Introduction
- Interruptions matérielles
- **Interruptions logicielles**
- Direct Access Memory
- Exemple d'implémentation



# Interruptions logicielles

- Identique à une interruption matérielle, mais elle est déclenchée par une instruction :
  - INT X
  - où X est le numéro de l'interruption que l'on veut déclencher…
  - On se réfère toujours au vecteur d'interruption (qui peut être changé dynamiquement).
- Ex. avec l'architecture wintel :
  - INT 10h : appels BIOS
    - Basic Input/Output System
  - INT 21h : appels fonctions « système » (ici DOS)



# Au sommaire...

- Introduction
- Interruptions matérielles
- Interruptions logicielles
- **Direct Access Memory**
- Exemple d'implémentation

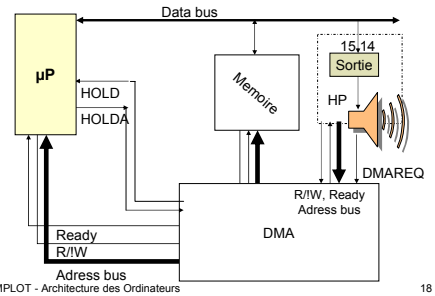


# Direct memory access (1/2)

- Avec les interruptions, on a transformé les attentes actives (*e.g.* par scrutation) en attente passive :
  - Le processeur peut faire une autre tache en attendant le périphérique
- Mais pour le transfert d'informations, c'est le µP qui envoie sert de passerelle entre le périphérique et la mémoire…
- Solution : DMA
  - Accès Direct à la Mémoire
  - Direct Memory Access



# Direct memory access (2/2)



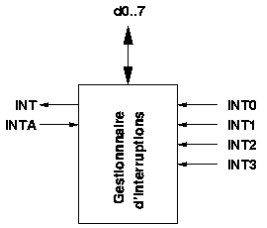
# Au sommaire...





# Exemple d'implémentation (1/7)

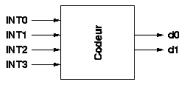
- Gestionnaire d'interruptions
  - 4 périphériques






# Exemple d'implémentation (2/7)

- Signal INT
  - INT = INT0 + INT1 + INT2 + INT3
- Sur réception d'un signal INTA
  - Positionner d0..7 le numéro de l'interruption levée
  - Utilisation d'un codeur

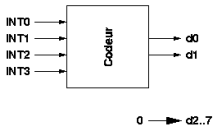



| INT3..0 | d1..0 |
|---|---|
| 0 0 0 0 | ? ? |
| 0 0 0 1 | 0 0 |
| 0 0 1 0 | 0 1 |
| 0 0 1 1 | ? ? |
| 0 1 0 0 | 1 0 |
| 0 1 0 1 | ? ? |
| 0 1 1 0 | ? ? |
| 0 1 1 1 | ? ? |
| 1 0 0 0 | 1 1 |
| 1 0 0 1 | ? ? |
| 1 0 1 0 | ? ? |
| 1 0 1 1 | ? ? |
| 1 1 0 0 | ? ? |
| 1 1 0 1 | ? ? |
| 1 1 1 0 | ? ? |
| 1 1 1 1 | ? ? |



# Exemple d'implémentation (3/7)

- On instaure une priorité




| INT3..0 | d1..0 |
|---|---|
| 0 0 0 0 | 0 0 |
| 0 0 0 1 | 0 0 |
| 0 0 1 0 | 0 1 |
| 0 0 1 1 | 0 0 |
| 0 1 0 0 | 1 0 |
| 0 1 0 1 | 0 0 |
| 0 1 1 0 | 0 1 |
| 0 1 1 1 | 0 0 |
| 1 0 0 0 | 1 1 |
| 1 0 0 1 | 0 0 |
| 1 0 1 0 | 0 1 |
| 1 0 1 1 | 0 0 |
| 1 1 0 0 | 1 0 |
| 1 1 0 1 | 0 0 |
| 1 1 1 0 | 0 1 |
| 1 1 1 1 | 0 0 |



# Exemple d'implémentation (4/7)

- Comment calculer d0 et d1
  - Utilisation d'un programme ad'hoc
    - Voir sur le site du cours
    - Fonction à 4 variables logiques
    - Pour d0 :
      - 5 points vrais (le reste à faux) : 2, 6, 8, 10 et 14
      - d0 = (INT3.$\overline{INT2}$ + INT1).$\overline{INT0}$
    - Pour d1 :
      - 3 points vrais (le reste à faux) : 4, 8 et 12
      - d1 = (INT3 + INT2).INT1.INT0



```
simplot@brunehaut-~/ens/a.../old/C/opti] ./qmc
Nombre de variables de votre fonction logique :
4
Vos variables seront étiquetées de a3 à a0.

Nombre de lignes de la table de vérité : 16

Vous désirez faire la saisie par :
 1. points vrais - le reste à faux,
 2. points faux - le reste à vrai,
 3. points vrais et points faux - le reste indifférent,
 4. points vrais et points indifférents - le reste à faux,
 5. points faux et points indifférents - le reste à vrai.
Votre choix : 1

Nombre de points vrais : 3
Nombre de points faux : 13
Nombre de points indefinis : 0
Entrez en décimal (a0=pds faible) les points vrais
4
8
12
Mode d'optimisation :
 ...
```

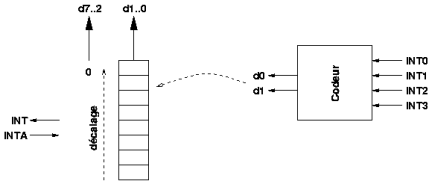
```
Etape 1 : trouver les implicants premiers
...

Liste des implicants premiers :
*100
1*00

Etape 2 : recouvrement optimal

liste des termes à valider
{ 0}  4 0100
{ 1}  8 1000
{ 2}  12 1100

                    4   8  12
[  0] *100 ----X---|---X 3
[  1] 1*00 ----|---X---X 3

suppression des lignes inutiles... aucune
règle 1 : IP obligatoires... [0] [1]
terminé !

f = a2!a1!a0 + a3!a1!a0

simplot@brunehaut-~/ens/a.../old/C/opti]
```

# Exemple d'implémentation (5/7)

- Problème de cette implémentation :
  - Le périphérique peut monopoliser le CPU…
  - Il faudrait mémoriser les INT au fur et à mesure qu'elles arrivent
    - ⇒ utilisation d'un buffer
- Buffer = FIFO (first in first out)
  - (rappel, la pile = LIFO)
  - Utilisation de registres chaînés les uns avec les autres



# Exemple d'implémentation (6/7)




- Pbs =
  - Où stocker dans mes huit registres ?
  - Comment informer mon périphérique qu'il a été enregistré ?
  - Comment faire la différence entre une case vide et l'interruption 0 ?



# Exemple d'implémentation (7/7)