M2-SEM

Concepts avancés d'architecture

Cours 4.2

E/S et Interruptions

Année 2020-2021

Pr. R. BOUDOUR

Quelques exemples d'E/S

Device	Input/Output	Data rate	Type
Keyboard	Input	100 bps	char
Mouse	Input	3800 bps	char
Voice input/output	Input/Output	264 Kbps	block burst
Sound input	Input	3 Mbps	block burst or steady
Scanner	Input	3.2 Mbps	block burst
Laser printer	Output	3.2 Mbps	block burst
Sound output	Output	8 Mbps	block burst or steady
Flash drive	Storage	480-800 Mbps read; 80 Mbps write	block burst
USB	Input or output	1.6-480 Mbps	block burst
Network/Wireless LAN	Input or output	11-100 Mbps	block burst
Network/LAN	Input or output	100-1000 Mbps	block burst
Graphics display	Output	800-8000 Mbps	block burst or steady
Optical disk	Storage	4-400 Mbps	block burst or steady
Magnetic tape	Storage	32-90 Mbps	block burst or steady
Magnetic disk	Storage	240-3000 Mbps	block burst

Gestion des entrées-sorties

- Les interruptions permettent au matériel de communiquer avec le processeur.
- Si on désire que le processeur réagisse rapidement à un évènement extérieur : par exemple, à l'arrivée d'un paquet de données sur une connexion réseau, à la frappe d'un caractère au clavier, à la modification de l'heure (évidemment l'heure change en permanence... nous verrons que le circuit d'horloge, extérieur au processeur, envoie un signal d'interruption à intervalles réguliers de quelques ms).
- Les interruptions sont ainsi surtout utilisées pour la gestion des périphériques de l'ordinateur.

Bornes d'interruptions du 8086

- Les processeurs pour PC possèdent trois bornes pour gérer les interruptions : NMI, INTR, et INTA.
  - NMI est utilisée pour envoyer au processeur une interruption non masquable (NMI signifie " Non Maskable Interrupt "). Le processeur ne peut pas ignorer ce signal, et va exécuter le traitant donné correspondant. Ce signal est normalement utilisé pour détecter des erreurs matérielles (mémoire principale défaillante par exemple).
  - INTR (INTerrupt Request) correspond à une demande d'interruption masquable. Utilisée pour indiquer au processeur l'arrivée d'une interruption.
  - INTA (INTerrupt Acknowledge) : cette borne est mise à 0 lorsque le processeur traite effectivement l'interruption signalée par INTR (c'est-à-dire qu'elle n'est plus masquée).

Indicateur IF

- A un instant donné, les interruptions sont soit masquées soit autorisées, suivant l'état d'un indicateur spécial du registre d'état, IF (Interrupt Flag).
  - Si IF = 1, alors le processeur accepte les demandes d'interruptions masquables, c'est-à-dire qu'il les traite immédiatement ;
  - si IF = 0, alors le processeur ignore ces interruptions.
- L'état de l'indicateur IF peut être modifié à l'aide de deux instructions, CLI (CLear IF pour la mise à 0 de IF), et STI (SeT IF, pour la mise à 1 de IF).

Contrôleur d'interruptions

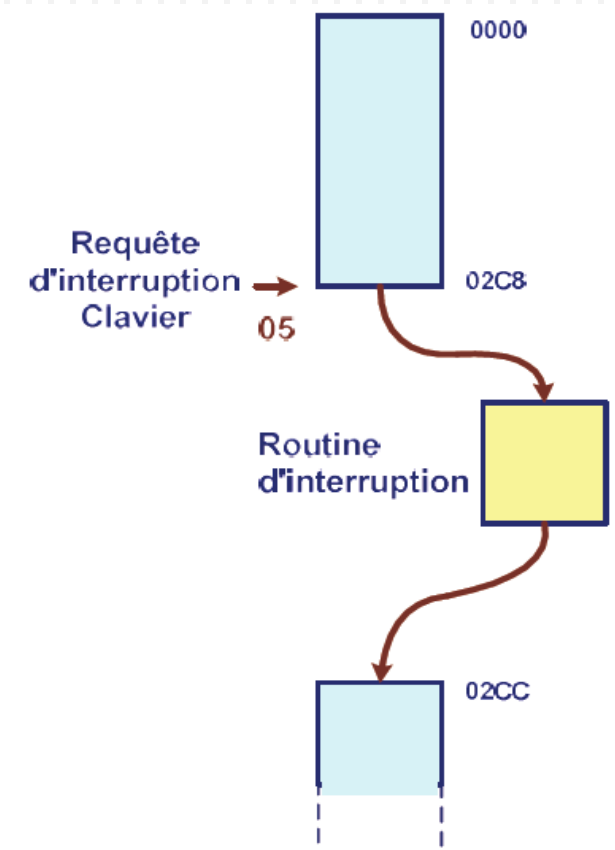
- L'ordinateur est relié a plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR.
- Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques

Contrôleur de périphérique

- Le processeur doit gérer des périphériques :
  - de différents rôles (entrée, sortie, sauvegarde)
  - de différentes vitesses !
  - avec des langages différents
- Sous-traitance de cette gestion à des contrôleurs

Vecteur d'interruption

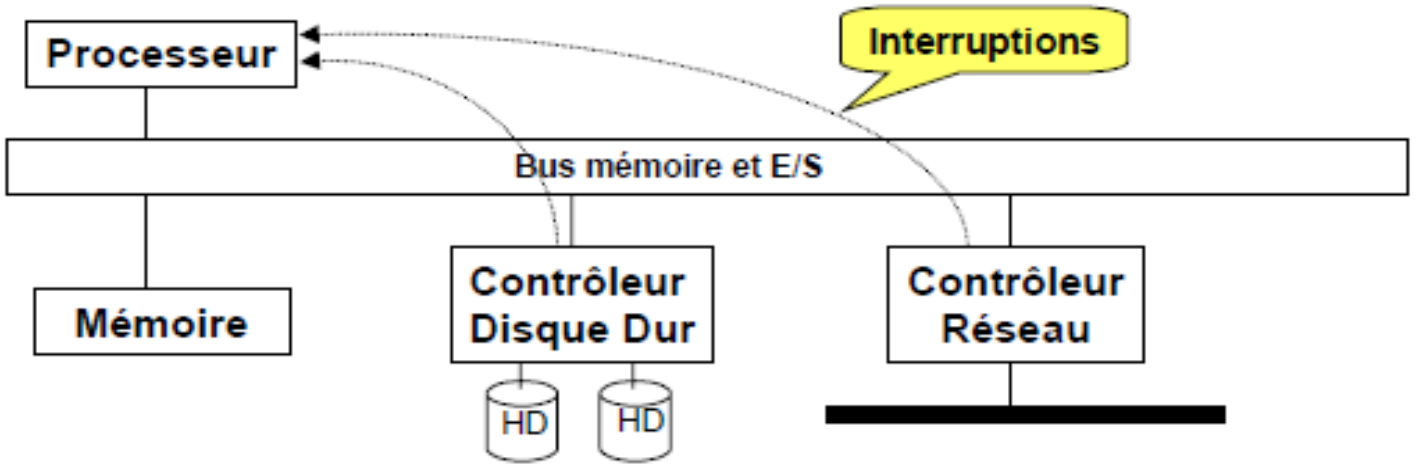
- Moyen technique pour attacher une routine d'interruption (ISR) à une requête d'interruption (IRQ)
  - numéro périphérique
  - adresse physique
  - etc.
- Table des vecteurs
  - Localisation à l'adresse 0:0 pour 8086/8088



Périphérique et contrôleur

Chaque périphérique est « piloté » par un contrôleur qui

- contient souvent son propre microprocesseur, ses registres et sa mémoire « tampon »
- s'occupe des commandes détaillées du périphérique :
  - » gestion des incidents
  - » détection d'erreurs
  - » conversion de format ...



E/S (Périphérique et contrôleur)

- Le dialogue Pocesseur/Contrôleur s'effectue grâce aux registres du contrôleur (ses ports) contenant :
  - des données
  - l'état du contrôleur
  - les commandes à effectuer
- Chaque E/S est implantée en mémoire
  - une partie de la mémoire est réservée aux E/S
  - lire/écrire dans cette zone = commander le périphérique

Adressage des périphériques

- Memory-Mapped I/O (MMIO)
  - Les adresses des registres du périphérique apparaissent dans l'espace d'adressage du processeur (cartographie mémoire)
  - Les transferts vers et depuis le périphérique (données, commandes, configuration ou statut) se réalisent de façon identique avec les transferts mémoires (ex. instructions load/store, modes d'adressage impliquant la mémoire, etc.)
- Port-Mapped I/O (PMIO)
  - Deux espaces d'adressage distincts : un pour la mémoire et un pour les périphériques
  - Les transferts vers et depuis les périphériques se réalisent avec des instructions spécifiques (ex. instructions IN et OUT du x86)

MMIO vs PMIO

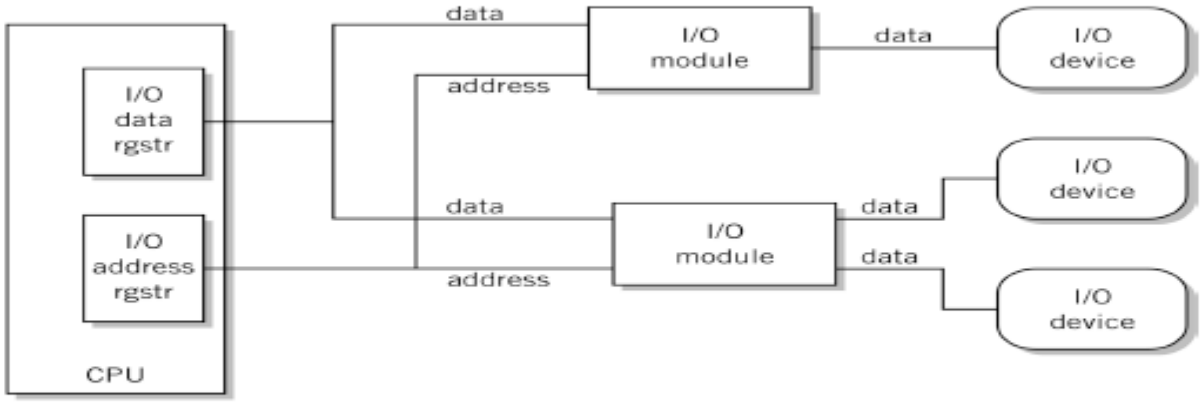
- **Memory-Mapped I/O**
  - Un seul bus à gérer par le processeur
  - Toutes les instructions qui manipulent des opérandes en mémoire peuvent de fait manipuler des opérandes en provenance de périphériques
  - Contrôle d'accès peut être géré par la MMU (mais granularité d'une page)
- **Port-Mapped I/O**
  - Pas de réduction de l'espace d'adressage qui peut être occupé par de la mémoire
  - Contrôle d'accès souvent en tout ou rien (mode superviseur)
  - Instructions spécifiques (ex. architecture x86 : IN et OUT ne peuvent utiliser que le registre (E)AX comme source ou destination du transfert)

Gestion des E/S

- **Trois méthodes de gestion des E/S**
  - La liaison programmée
  - Les E/S pilotées par interruption
  - L'uilisation d'un dispositif permettant l'accès direct à la mémoire

E/S programmée

- Modèle le plus simple dans lequel le contrôleur E/S est connecté à une paire de registres E/S (donnée&adresse) dans le CPU via un bus

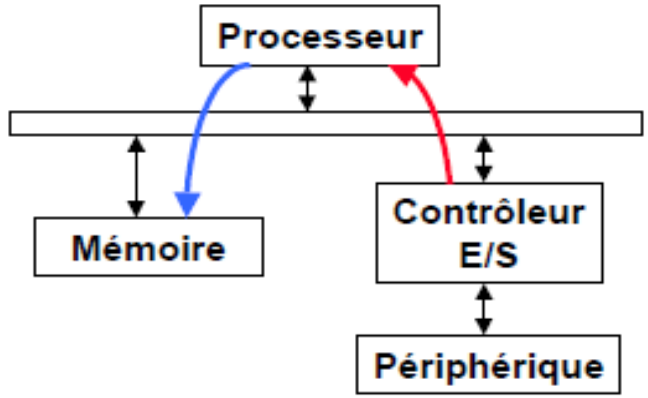


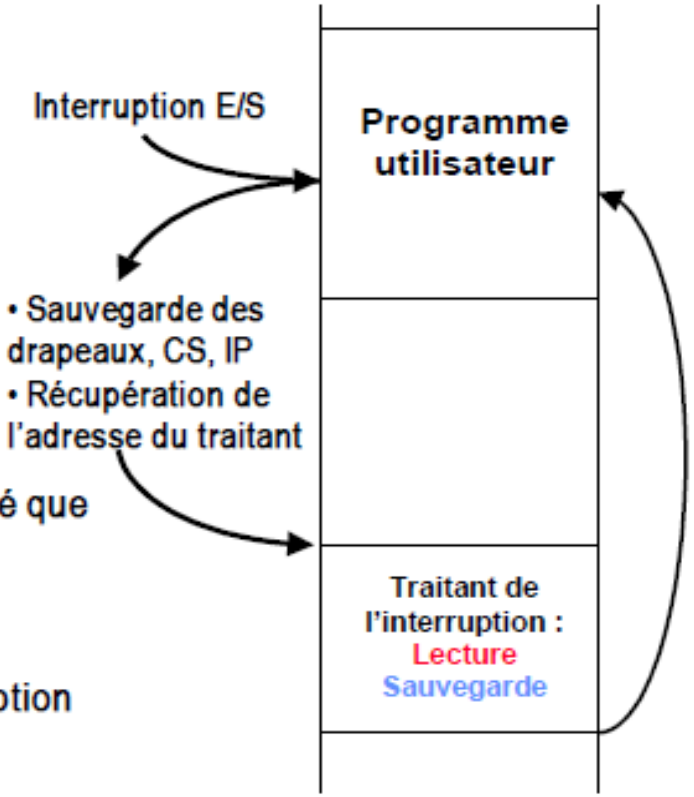
- Processeur central est totalement utilisé pour contrôler et piloter les échanges avec le périphérique
  - Transfert d'un mot à la fois; le CPU reste bloqué durant toute la durée de l'échange

E/S par interruption

- Comment l'ordinateur identifie le périphérique demandant une interruption?
- Scrutation
  - Identification du composant par scrutation, en interrogeant, tous les périphériques
- Interruption "vectored"
  - Le périphérique après avoir déposé un signal d'interruption, place sur le bus de communication l'identification de l'interruption
  - Matériel supplémentaire

Identification par interruption





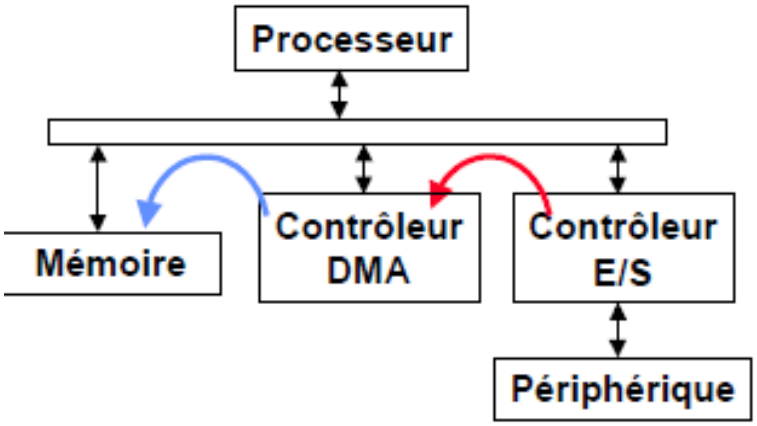
Avantage = le programme utilisateur n'est arrêté que pendant le temps du transfert

Inconvénient = besoin de plus de matériel :

le contrôleur doit savoir générer une interruption

le processeur doit savoir la détecter, ...

E/S par transfert direct mémoire



- Le processeur envoie au contrôleur DMA :
  - l'adresse de début
  - la longueur des données
  - le sens du transfert

  puis il déclenche le transfert.

- Le contrôleur DMA prend en charge :
  - les commandes pour le contrôleur de périphériques
  - les commandes et adresses pour la mémoire

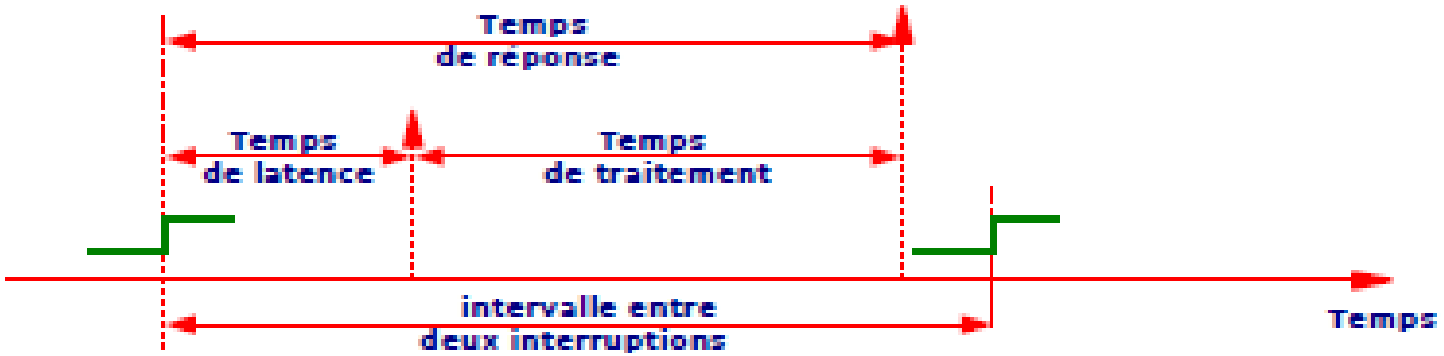
Avantages :

- DMA externe / processeur
- transfert de données sans passer par le processeur !
- DMA prioritaire sur le Bus

Installation des handlers

- A l'initialisation du système (exceptions) ou des périphériques (interruptions), à l'aide de la table d'interruptions, le vecteur d'interruption va pointer vers l'adresse de l'Interrupt Service Routine (ISR) ou de l'Exception Service Routine (ESR) correspondante.
- la plupart des OS Temps Réel proposent
  - des handlers par défaut
  - ...

Quelques considérations sur le temps



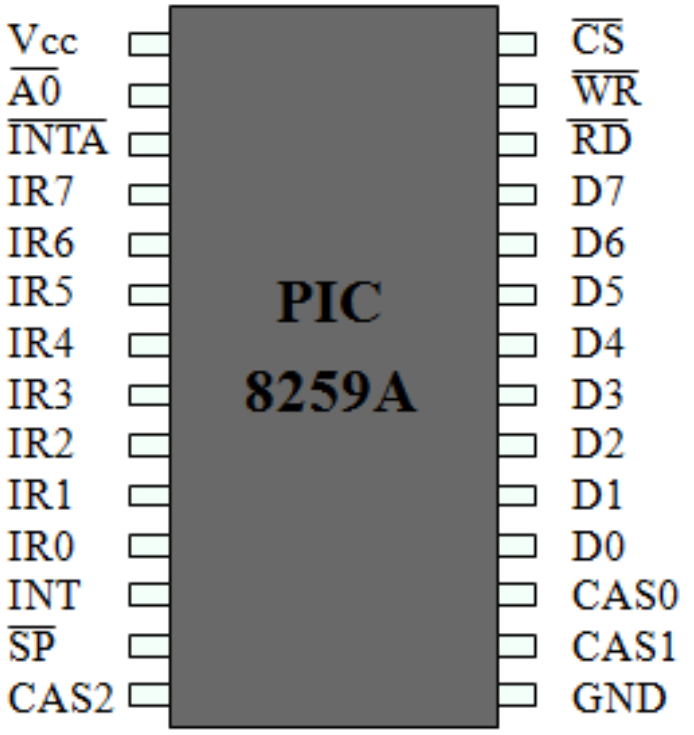
- Le temps de latence dépend :
  - du processeur
  - des interruptions de plus haute priorité en cours
  - du masquage
- Le temps de traitement ne dépend que du programmeur

Interruptions matérielles et PICs

INT (Hex)	IRQ	Function
77	IRQ15	Reserved
76	IRQ14	Hard Disk Drive
75	IRQ13	Maths Co-Processor
74	IRQ12	PS/2 Mouse
73	IRQ11	Reserved
72	IRQ10	Reserved
71	IRQ9	Redirected IRQ2
70	IRQ8	Real Time Clock
0F	IRQ7	Parallel Port
0E	IRQ6	Floppy Disk Controller
0D	IRQ5	Reserved/Sound Card
0C	IRQ4	Serial Port
0B	IRQ3	Serial Port
0A	IRQ2	PIC2
09	IRQ1	Keyboard
08	IRQ0	System Timer

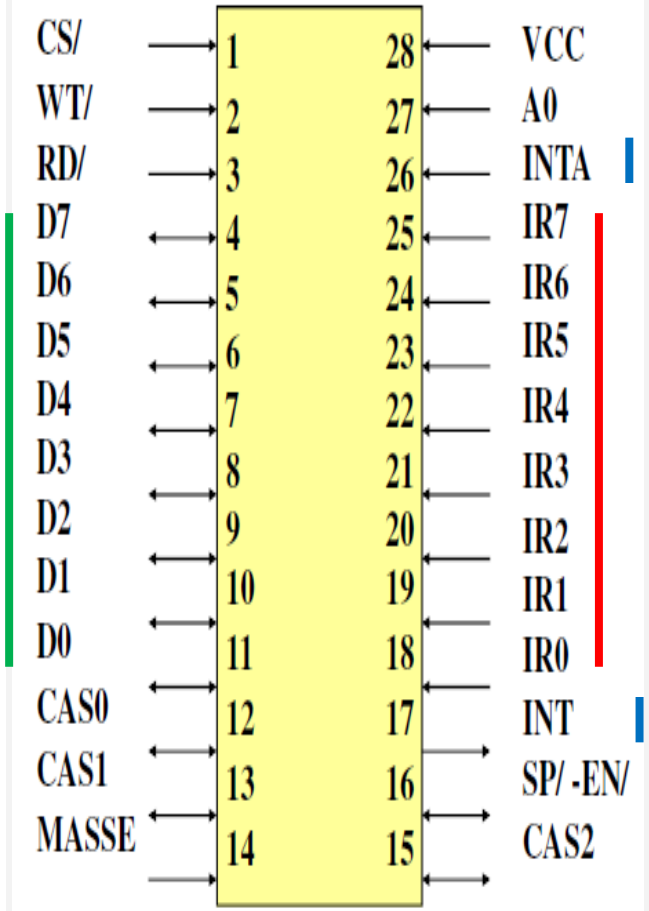
Le contrôleur d'Interruption 8259A

- L'ordinateur est relié a plusieurs périphériques, mais nous venons de voir qu'il n'y avait qu'un seul signal de demande d'interruption, à savoir INTR.
- Le contrôleur d'interruptions est un circuit spécial, extérieur au processeur, dont le rôle est de distribuer et de mettre en attente les demandes d'interruptions provenant des différents périphériques.



Contrôleur d'interruption 8259A

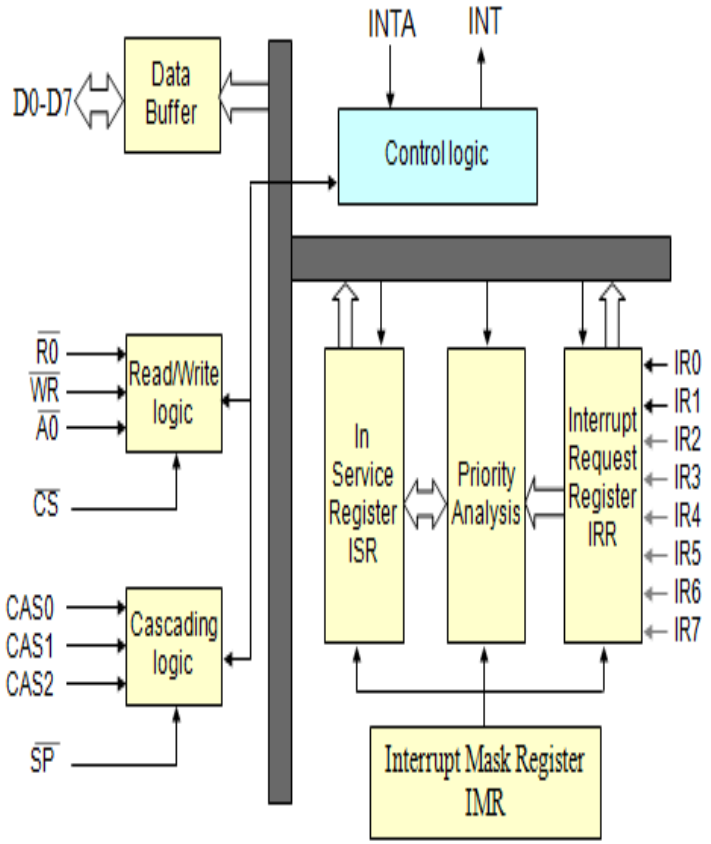
- 8 niveaux d'IT (IR0 -IR7) gèrent les périphériques
- INT et INTA : demande et accusé de réception de l'interruption
- D0 à D7 : Bus de donnée
- Chip Select: Sélection du PIC afin qu'il puisse être accédé
- A0, WT et RD : Connexion au bus d'adresse pour lecture et écriture sur le PIC
- CAS0-1-2 : Multiplication des IT.
- SP-EN: PIC en mode maître ou esclave



Contrôleur d'interruption 8259A

Découpage par fonctions :

- logique de contrôle => connexion au micro-processeur
- Buffer de données => connexion au bus de données
- logique de lecture /écriture
- Cascade - Comparateur : Gestion de PIC en cascade
- registre des IT en service **ISR**
- registre de demande d'IT **IRR**
- résolveur de priorité lorsque plusieurs PIC sont en cascade
- Registre du masque d'IT **IMR,** mémorise les IT interdites



Contrôleur d'interruption 8259A

Traitement d'une interruption:

- Demande du périphérique IRQ0-7
- Réception par le PIC + positionnement IRR
- Evaluation de la demande (Priorité)
- PIC informe le µC => INT
- Le µC prend connaissance du flag IF + contexte => INTA
- Réception de INTA par le PIC
- Positionnement de ISR et IRR
- PIC => bus de donnée le type de l'IT
- Le µC déduit son traitement => Table des vecteurs à l'indice 4*N°IT
- Branchement du sous programme
- Reprise de la tâche interrompue

Le contrôleur d'Interruption 8259A

- Le contrôleur est relié aux interfaces gérant les périphériques par les bornes IRQ (InteRrupt reQuest).
- Il gère les demandes d'interruption envoyées par les périphériques, de façon à les envoyer une par une au processeur via INTR.
- Il est possible de programmer le contrôleur pour affecter les priorités différentes à chaque périphérique, mais nous n'aborderons pas ce point dans ce cours.
- Avant d'envoyer l'interruption suivante, le contrôleur attend d'avoir reçu le signal INTA, indiquant que le processeur a bien traité l'interruption en cours.

Le contrôleur d'Interruption 8259A

- Le 8259A peut accepter les interruptions de 8 sources externes, et on peut gérer jusqu'à 64 sources différentes en cascadant plusieurs 8259A. Il gère la priorité entre les interruptions simultanées, interrompt le processeur et lui passe un code pour identifier la source d'interruption.
- Une source d'interruption est connectée à chacune des 8 entrées IR0 à IR7. Selon sa priorité, et s'il n'y a pas d'autre interruption en cours, le PIC décide s'il peut transmettre l'interruption au CPU.
- Si oui, il affirme la ligne INT, qui est connectée à l'entrée INTR du CPU. Si le CPU est prêt à accepter l'interruption, il répond au PIC via la ligne INTA.

Le contrôleur d'Interruption 8259A

- Le PIC répond à son tour en envoyant le numéro d'interruption sur les lignes bus de données D0 à D7. Ce numéro est un index dans la table des vecteurs d'interruption.
- Le CPU est maintenant prêt à appeler le sous-programme de traitement d'interruption approprié.
- Quand le sous-programme de traitement d'interruption a terminé son exécution, il en avertit le PIC pour qu'il puisse permettre à d'autres interruptions d'atteindre le CPU :

```
mov     al, 0x20
out     0x20, al
```

Le contrôleur d'Interruption 8259A

- Le PIC comporte deux sortes de commandes :
  - les commandes d'initialisation (ICW)
  - et les commandes opérationnelles (OCW).
- Ces commandes sont appliquées à deux registres internes, situés respectivement aux adresses 0x20 et 0x21 dans l'espace mémoire d'entrée/sortie.

Processus d'Interruption

INT NN *est une instruction à 2 octets (**Opcode + Nombre**). Il existe 256 INTs (Software + Hardware)*

Interruptions Hardware →

- ***INTA,*** → *Output de CPU*
- ***INTR,*** → *Input vers CPU et Masquée par **CLI** et **STI***
- ***NMI*** → *Input vers CPU et non Masquée*

1. *Finir l'instruction en cours.*
2. *PUSH **FR, CS, IP** dans la pile*
3. *Saute à la **Table des vecteurs d'Interruptions***
4. *Chercher CS:IP pour exécuter l'ISR associé*
5. *A la fin d'ISR → l'instruction **IRET** → CPU à POPer de la pile **FR CS, IP**.*
6. *CPU continue son programme normalement.*

Processus d'Interruption

1. *FR → Pile (SP → SP –2).*
2. *IF et TF → 0 ➔ Le système ignore les INTR et désactive les 'Single-Stepping' pendant l'exécution de ISR*
3. *CS → Pile (SP → SP –2), IP → Pile (SP → SP –2).*
4. *Le type d'INT (NN) x 4 pour obtenir l'adresse (CS:IP) de l'ISR de la* ***Table des vecteurs d'Interruptions.***
5. *CPU commence à exécuter le programme ISR (à partir de CS:IP)*
6. *La dernière instruction d'ISR est IRET pour: Pile → CS, IP, FR.*

Déroulement détaillé d'une interruption

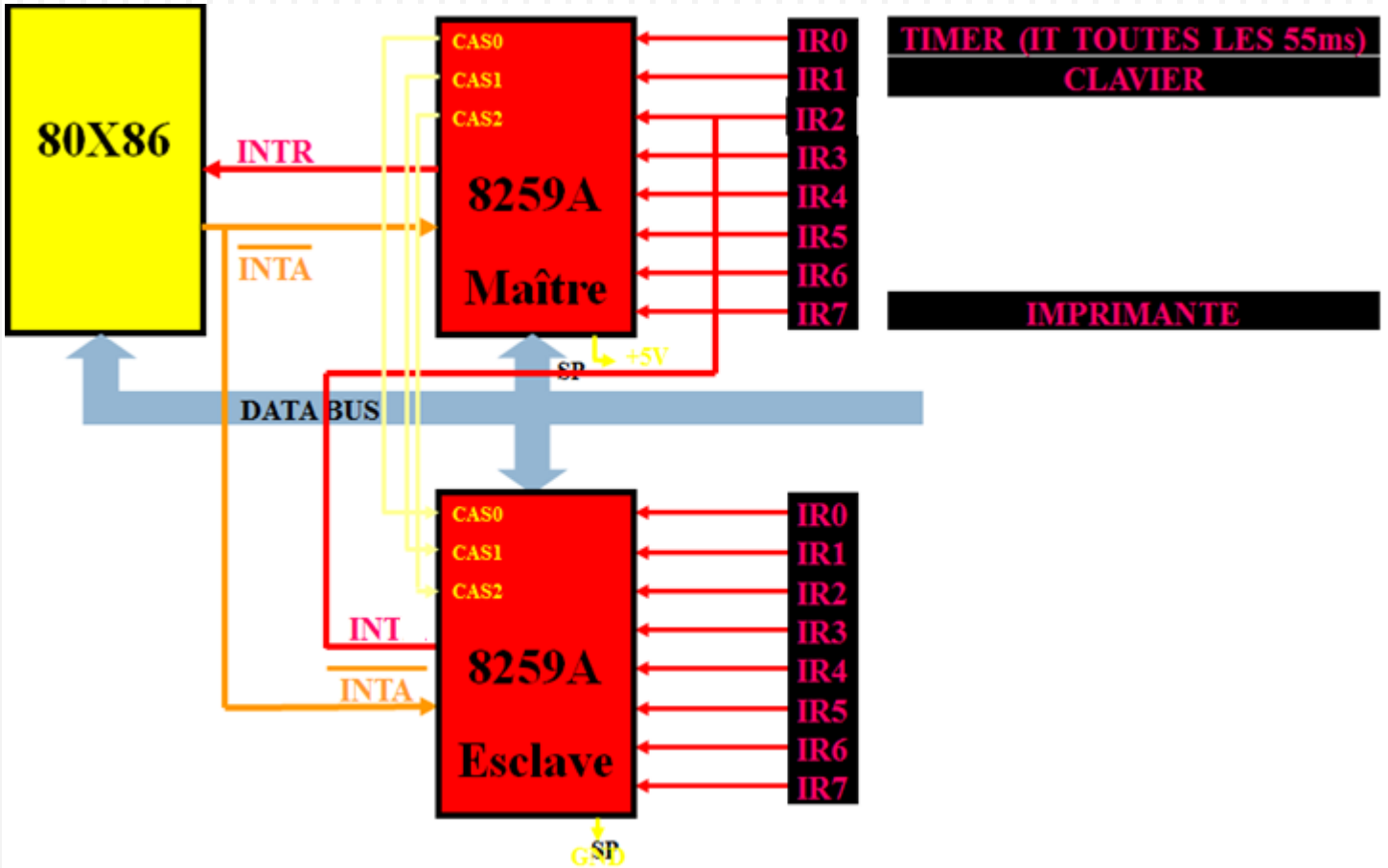
Reprenons les différents évènements liés à la réception d'une interruption masquable :

1. Un signal INT est émis par un périphérique (ou plutôt par l'interface gérant celui-ci).
2. Un contrôleur d'interruptions reçoit ce signal sur une de ses bornes IRQ_i. Dès que cela est possible (suivant les autres interruptions en attente de traitement), le contrôleur envoie un signal sur sa borne INT.
3. Le processeur prend en compte le signal sur sa borne INTR après avoir achevé l'exécution de l'instruction en cours (ce qui peut prendre quelques cycles d'horloge). Si l'indicateur IF=0, le signal est ignoré, sinon, la demande d'interruption est acceptée.
4. Si la demande est acceptée, le processeur met sa sortie INTA au niveau 0 pendant 2 cycles d'horloge, pour indiquer au contrôleur qu'il prend en compte sa demande.
5. En réponse, le contrôleur d'interruption place le numéro de l'interruption associé à la borne IRQ_i sur le bus de données.

Déroulement détaillé d'une interruption

6. Le processeur lit le numéro de l'interruption sur le bus de données et l'utilise pour trouver le vecteur d'interruption. Ensuite, le processeur :
    1. sauvegarde les indicateurs du registre d'état sur la pile;
    2. met l'indicateur IF à 0 (masque les interruptions suivantes);
    3. sauvegarde CS et IP sur la pile;
    4. cherche dans la table des vecteurs d'interruptions l'adresse du traitant d'interruption, qu'il charge dans CS:IP.
7. La procédure traitant l'interruption se déroule. Pendant ce temps, les interruptions sont masquées (IF=0). Si le traitement est long, on peut dans certains cas ré-autoriser les interruptions avec l'instruction STI.
8. La procédure se termine par l'instruction IRET, qui restaure CS, IP et les indicateurs à partir de la pile, ce qui permet de reprendre le programme qui avait été interrompu

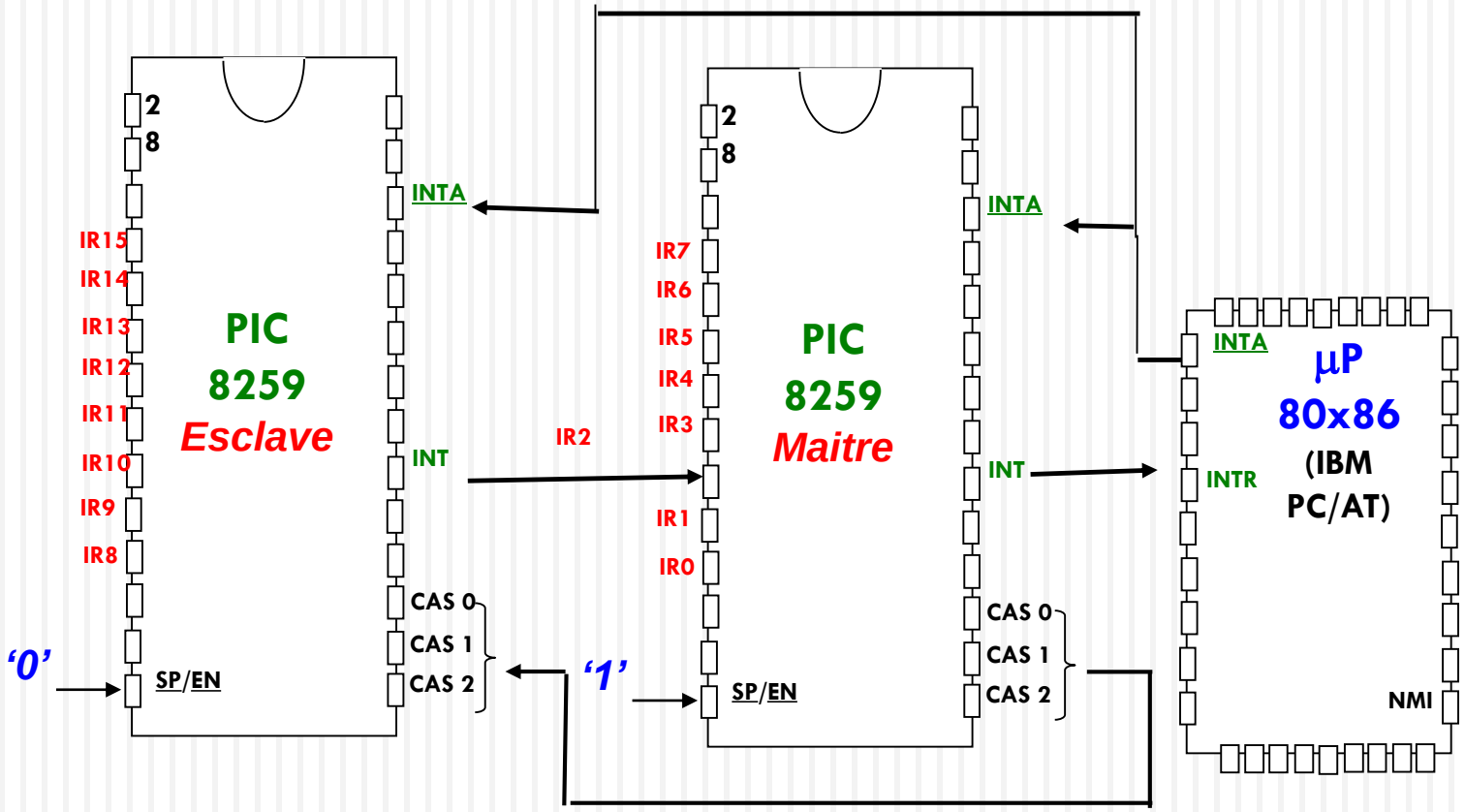
Deux contrôleurs d'interruptions



Deux contrôleurs d'interruptions

- L'insertion d'un deuxième PIC sur la ligne IRQ2 du premier PIC a plusieurs conséquences découlant surtout d'un désir de conserver la compatibilité avec les designs n'ayant qu'un seul PIC :
  - Les périphériques connectés sur IRQ2 ont été connectés sur IRQ9 (re-direction). Comme le vecteur d'interruption de IRQ9 n'est pas le même que celui de IRQ2, il faut que IRQ9 appelle le vecteur d'interruption de IRQ2 pour les designs avec un seul PIC.
  - Désactiver l'IRQ2 à l'intérieur du PIC principal désactive les IRQ8 à IRQ15.
  - Les interruptions hardware 8 à 15 sont plus prioritaires que les interruptions hardware 3 à 7.
  - Les routines de service d'interruption hardware IRQ8 à IRQ15 doivent gérer 2 PICs (c'est-à-dire envoyer 2 End Of Interrupt).

PICs et Priorités



l'ordre de priorité est donc le suivant :

0 > 1 > 8 > 9 > 10 > 11 > 12 > 13 > 14 >15> 3 > 4 > 5 > 6 > 7

Séquence d'événement pour 2 PICs

- La séquence d'évènements suivante se produit lorsqu'un périphérique produit une interruption :
  - Le PIC reçoit et traite l'interruption
    - Un registre interne du PIC permet au programmeur du 8086 de masquer (désactiver) certaines interruptions
    - Le PIC met l'interruption dans un buffer
    - Le PIC regarde les priorités des interruptions et détermine si l'interruption matérielle courante est la plus prioritaire
  - La ligne INTR est activée par le PIC pour dire au 8086 qu'il y a interruption
  - Une impulsion de 0V provenant du 8086 apparaît sur la ligne INTA pour signaler au PIC que l'interruption est reçue (Acknowledged). La ligne INTR est désactivée après l'impulsion.

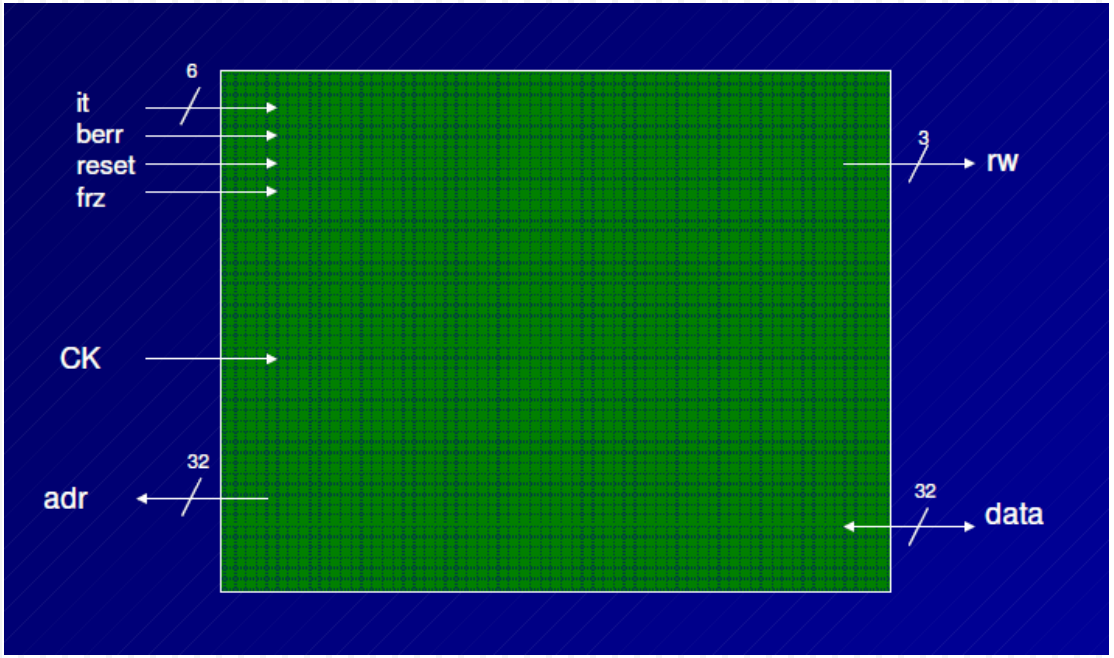
Séquence d'événement pour 2 PICs

- Une deuxième impulsion de 0V provenant du 8086 demande au PIC de mettre le numéro de l'interruption sur le bus de donnée.
- Le PIC met le numéro de l'INT sur le bus de donnée : il ne met pas le numéro de l'IRQ.
- Le 8086 exécute la routine de service de l'interruption
- Le 8086 envoie un EOI (End Of Interrupt) pour dire au PIC que l'interruption est terminée.

Table des interruptions

INT (Hex)	IRQ	Common Uses
00 - 01	Exception Handlers	-
2	Non-Maskable IRQ	Non-Maskable IRQ (Parity Errors)
03 - 07	Exception Handlers	-
8	Hardware IRQ0	System Timer
9	Hardware IRQ1	Keyboard
0A	Hardware IRQ2	Redirected
0B	Hardware IRQ3	Serial Comms. COM2/COM4
0C	Hardware IRQ4	Serial Comms. COM1/COM3
0D	Hardware IRQ5	Reserved/Sound Card
0E	Hardware IRQ6	Floppy Disk Controller
0F	Hardware IRQ7	Parallel Comms.
10 - 6F	Software Interrupts	-
70	Hardware IRQ8	Real Time Clock
71	Hardware IRQ9	Redirected IRQ2
72	Hardware IRQ10	Reserved
73	Hardware IRQ11	Reserved
74	Hardware IRQ12	PS/2 Mouse
75	Hardware IRQ13	Math's Co-Processor
76	Hardware IRQ14	Hard Disk Drive
77	Hardware IRQ15	Reserved
78 - FF	Software Interrupts	-

MIPS R3000



MIPS offre 6 lignes d'interruptions

Déroutement d'interruption

- Lors du démarrage du PC, la table des vecteurs d'interruption est chargée avec des valeurs par défaut.
- Pour détourner une interruption, il suffit de changer la table des vecteurs d'interruptions. Changer le CS et l'IP à l'adresse 4*(xx de l'INT à détourner) permet de changer le code exécuté lorsque l'interruption se produira.
- En temps normal, la nouvelle ISR appellera l'ancienne ISR sous certaines conditions. Pour cette raison, sauvegarder le CS et l'IP de l'instruction que l'on remplace est une pratique recommandée (habituellement la sauvegarde se fait dans des variables déclarées à cet effet).

Déroutement d'interruption

- **Deux fonctions : 25h et 35h**
  - **Fonction 25h int 21h DOS**

    Cette fonction permet de modifier le contenu d'un vecteur d'une interruption en y référençant une nouvelle routine
    - **Paramètres d'entrée :**

      AH = 25h

      AL = Numéro de l'interruption de 0 à 255

      DX = Offset de la nouvelle routine

      DS = Segment de la nouvelle routine
    - **Effet de sortie : Nouveau vecteur d'interruption**

  - *Rq : Cette fonction ne modifie pas les registres du processeur*

Déroutement d'interruption

- **Fonction 35h int 21h DOS**

  Cette fonction permet de lire le contenu de l'un des 256 vecteurs d'interruption. Elle devrait précéder tout appel à la fonction 25H.

  - **Paramètres d'entrée :**

    AH = 35h

    AL = Numéro de l'interruption de 0 à 255

    BX = Offset de l'ancienne routine

    ES = Segment de l'ancienne routine

  - *Rq : Mis à part ES et BX , aucun autre registre n'est altéré.*

Exemple : Horloge

- L'horloge d'un PC peut être considéré comme un périphérique d'un type particulier. Il s'agit d'un circuit électronique cadencé par un oscillateur à quartz (comme une montre ordinaire), qui est utilisé pour gérer l'heure et la date, que de nombreux programmes utilisent.

Exemple : Horloge

- L'horloge envoie une interruption matérielle au processeur toutes les 0,055 secondes (soit 18,2 fois par seconde). Le vecteur correspondant (pour le traitant d'interruption) est le numéro 08H.
- Pour gérer l'heure, l'OS installe un traitant pour l'interruption 08H.
- Ce traitant incrémente simplement un compteur, qui est un nombre entier codé sur 32 bits et toujours rangé à l'adresse 0040 : 006C en mémoire principale.
- Ainsi, si un programme désire connaître l'heure, il lui suffit de lire cet emplacement mémoire, qui change "automatiquement " 18,2 fois par seconde. En langage C, on pourra utiliser la fonction time() qui effectue un appel système pour connaître l'heure courante.

Exemple : Horloge

- En général, les périphériques qui reçoivent des données de l'extérieur mettent en œuvre un mécanisme d'interruption : clavier, liaisons séries (modem, souris, ...) et parallèles (imprimantes), interfaces réseau, contrôleurs de disques durs et CD-ROMS, etc.

Disque dur et scrutation

- Soit un programme lisant des données sur un disque dur, les traitant et les affichant sur l'écran.
- Voici l'algorithme général sans utiliser d'interruption :
  - Répéter :
    - envoyer au contrôleur de disque une demande de lecture d'un bloc de données ;
    - attendre tant que le disque ne répond pas (scrutation) ;
    - traiter les données ;
    - afficher les résultats.
- Cette méthode simple est appelée entrée/sortie par scrutation.

Disque dur et scrutation

- L'étape 2 est une boucle de scrutation, de la forme :
  - Répéter :
    - regarder si le transfert du disque est terminé. Tant qu'il n'est pas terminé.
- La scrutation est simple mais inefficace : l'ordinateur passe la majorité de son temps à attendre que les données soient transférées depuis le disque dur. Pendant ce temps, il répète la boucle de scrutation.

Disque dur et interruption

- Ce temps pourrait être mis à profit pour réaliser une autre tâche. Très grossièrement, les entrées/sorties par interruption fonctionnent sur le modèle suivant :
  - Installer un traitant d'interruption disque qui traite les données reçues et les affiche ;
  - envoyer au contrôleur de disque une demande de lecture des données ;
  - faire autre chose (un autre calcul ou affichage par exemple).

Disque dur et interruption

- Dans ce cas,
  - dès que des données arrivent,
  - le contrôleur de disque envoie une interruption (via le contrôleur d'interrruptions) au processeur,
  - qui arrête temporairement le traitement en cours pour s'occuper des données qui arrivent.
  - Lorsque les données sont traitées, le traitement suspendu reprend (IRET).
  - Pendant l'opération (lente) de lecture du disque dur, le processeur peut faire autre chose.

Questions

Q1. Cascader 3 contrôleurs d'interruption dont 1 maître et 2 esclaves en mettant sur le schéma les connexions nécessaires : broches INTR, INT, Cas0, cas1, cas2, données, etc.

Q2. Présenter le système d'interruptions d'un processeur ARM

Q3. Idem pour un microcontrôleur


Thank's