Systèmes Entrée - Sortie

IFT1225

Dép. d'IRO, Université de Montréal
(professeur E. Cerny)

Plan

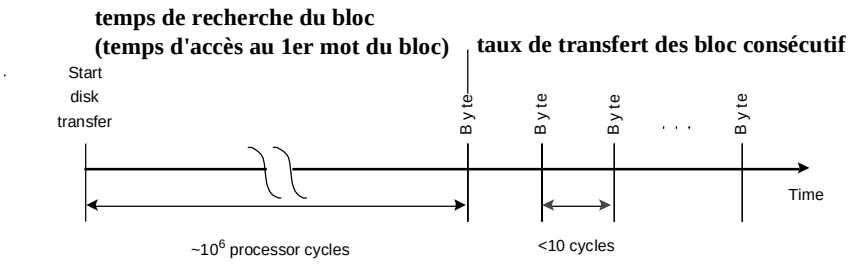
- Bus Entrée-Sortie
- E/S Programmé
- Interruptions
- Accès direct à la mémoire (ADM)
- Processeurs E/S

Caractéristiques d'un transfert E/S

- Sélection de l'appareil E/S et des données dans l'appareil
  - quel périphérique (imprimante, clavier, disque, souris, …)
  - quel bloc de données (disque, bande magnétique, CDROM, ...)
- Quantité de données qui peut être transférée dépend de l'appareil E/S et dans certains cas il faut la spécifier (p.e. avec ADM)
  - 1 octet à la fois (imprimante, clavier)
  - 1 mot à la fois ou un bloc (convertiseur A/N, N/A, Ethernet, …)
  - 1 ou plusieurs blocs (disque, …)
  - Nombre d'octets à lire / écrire (disque, bande, CDROM, …)
- Taux de transfert varie d'un appareil à l'autre (10 octets/sec, …, Mo/sec)
- Synchronisation entre CPUet l'appareil E/S critique, vitesses largement différentes

Caractéristiques temporelles d'un transfert

- Clavier - au plus 1 caractère (octet) par séconde
- Disque - temps pour accéder au bloc (piste, secteur) et ensuite le transfert du bloc d'octets
  - Le temps d'accès varie selon la position du bloc relative à la position de la tête de lecture/écriture au moment de l'émission de la commande



Synchronisation

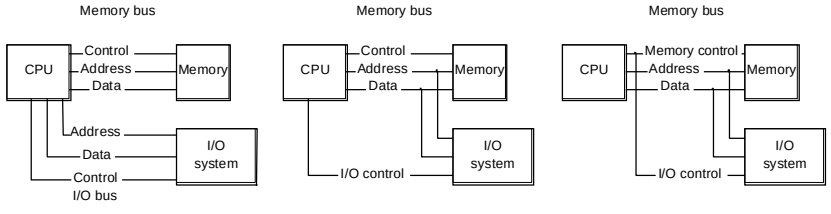
- Pas d'horloge commun entre CPU-mémoire et les périphériques, les opérations de CPU et des périphériques sont **asynchrones**
- Il faut indiquer la disponibilité de données (entrée) ou le fait que le périphérique est prêt à accepter de données
  - Bit de *statut* du périphérique, indique si une nouvelle donnée est disponible (lecture), ou le fait que périphérique peut en accepter une (sortie)
  - Bit de *commande* pour démarrer l'opération d'E/S
  - Bit du *statut d'opération* (erreurs, fin de transfert, …)
- Les données doivent être stable à l'entrée du CPU à l'arrivée de l'horloge
- Le CPU vérifie le statut pour ensuite emettre une commande
- Regrouper les bits de commande et de statut dans des registres qui sont manipulés (+/-) comme un mot de mémoire; aussi des registres de fonction ou d'adresse (sur disque), registre-tampon de données

Bus E/S: indépendant ou partagé avec mémoire

• Peut s'adapter aux particularités d'E/S
• Beaucoup de connexions

• Instructions séparées d'E/S
• Forme de synchronisation peut être différente

•Le plus simple,
•Minimum de lignes
•Le plus lent (pas de parallélisme)



(a) Separate memory and I/O buses (isolated I/O)

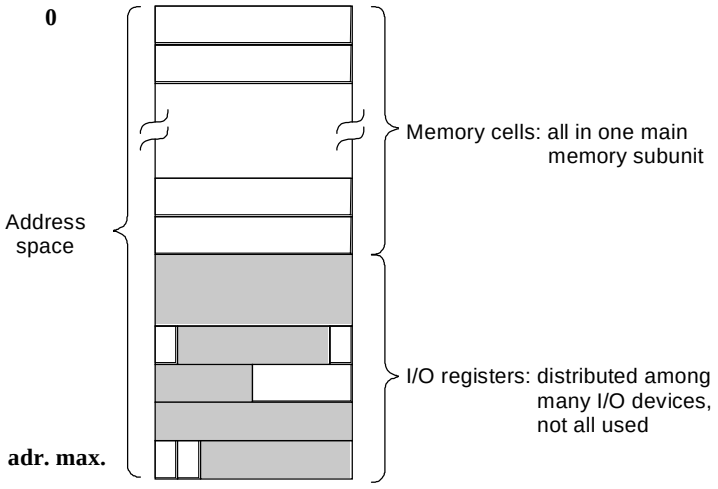
(b) Shared address and data lines

(c) Shared address, data, and control lines (memory-mapped I/O)

E/S dans l'espace d'adresses mémoire

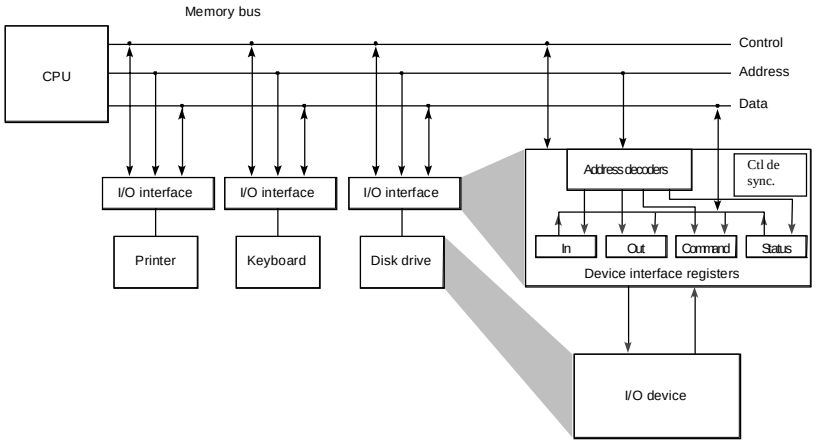
- Bus partagé entre E/S et la mémoire
- Registres de commande et de statut des périphériques apparaissent comme des mots de mémoire
- Un sous-espace d'adresses réservé pour E/S
- Flexible - frontière entre les deux espaces (mémoire vs. E/S) n'est pas rigide - plus de mémoire ou plus d'E/S
- Moins de signaux que 2 bus séparés, mais il faut ajouter quelques lignes supplémentaires sur le bus partagé
  - demandes et acquittements
    - d'interruptions
    - de transferts d'accès direct à la mémoire (ADM)
- Unifie et standardise toute échange de données avec le CPU
- Exemples: MC680xx, PDP-11, i80x86 (2 configurations possibles)

E/S dans l'espace d'adresses mémoire

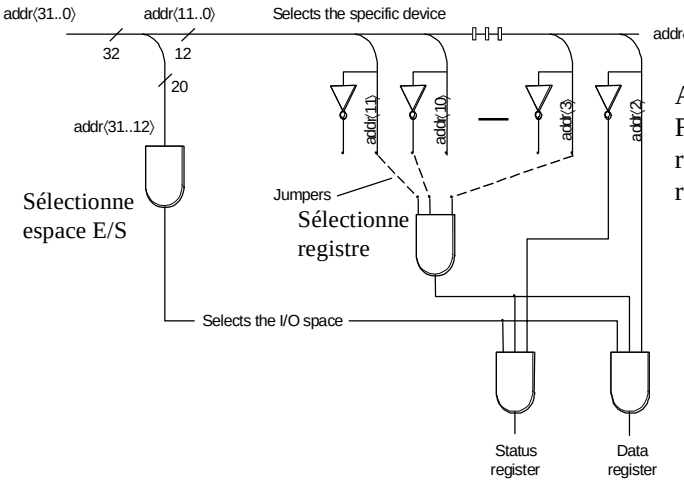


E/S programmé

- **Dans l'espae d'adresses mémoire, mais semblable avec un bus E/S séparé**
- **Périphériques attachés au bus via des registres, unité de contrôle de synchronisation**



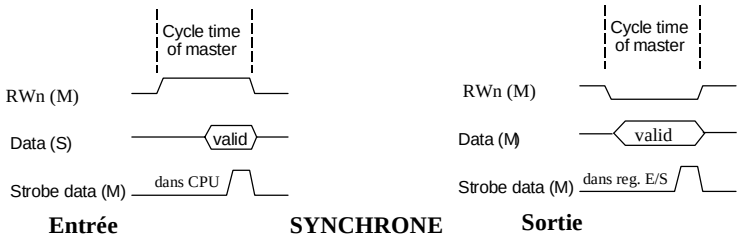
Exemple: Décodage de reg's E/S de SRC



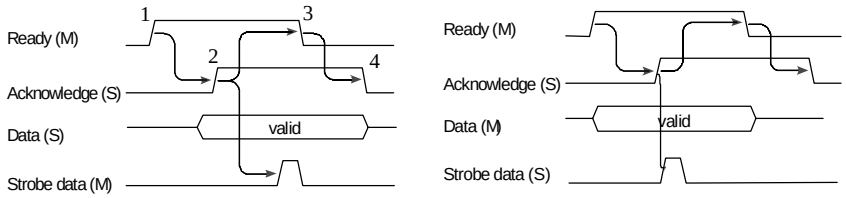
Adresses au dessus $FFFFF000_{16}$ réservées pour reg's E/S

- 1024 registres de 32 bits
- $FFFF0000_{16}$ à $FFFFFFFF_{16}$ adressées par déplacement négatif (relatif à 0)

Protocoles de bus

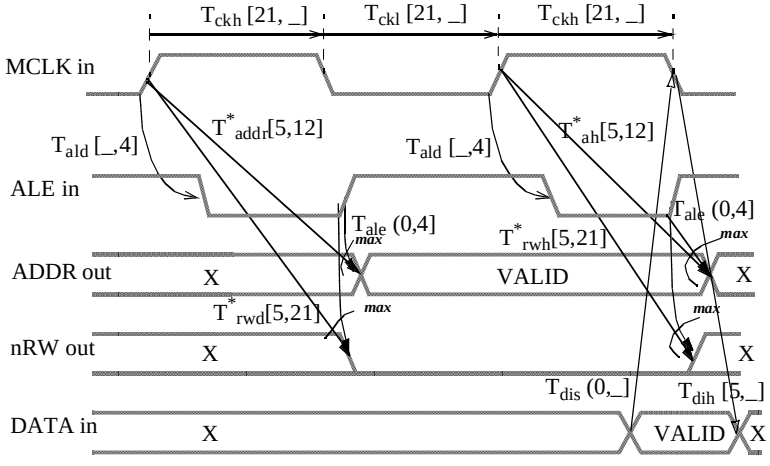


Entrée **SYNCHRONE** **Sortie**



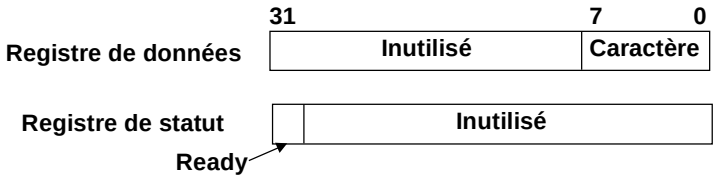
Entrée **ASYNCHRONE (à 4 cycles)** **Sortie**

Exemple de bus synchrone: ARM7 - cycle de lecture de SRAM

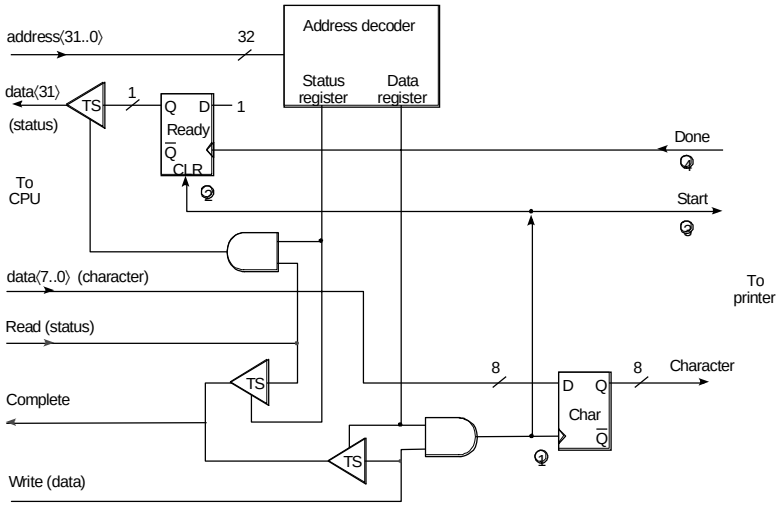


SRC: Sortie programmée de caractères

- Le périphérique exige:
  - 8 bits de données = un caractère ASCII
  - Signal Start pour valider les données et démarrer l'opération
  - Les données doivent être stables jusqu'à l'arrivée de Done
- Alignement entre le bus et le périphérique
  - caractère = 8 bits inférieurs du mot de 32 bits
  - Signal Start dérivé du signal de chargement du registre de données de l'interface
  - Bit de statut Ready pour indiquer que le périphérique est prêt
  - Bit Ready positionné à 0 avec Start et remis à 1 avec Done



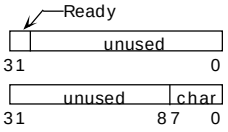
SRC: Interface de sortie de caractères



Fragment du programme pour la sortie de car.

Registre de statut COSTAT = FFFFF110H

Registre de données COUT = FFFFF114H




```
      lar    r3, Wait      ;Adresse de branchement pour"Wait"
      ldr    r2, Char      ;Charger le car. pour la sortie
Wait: ld     r1, COSTAT    ;Charger le contenu du reg. statut
      brpl   r3, r1        ;Y a-t-il un car.?
      st     r2, COUT      ;Oui, sortir le car. et démarrer
```

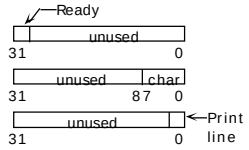
- Un SRC de 10 MIPS exécuterait 10,000 instructions en attendant l'imprimante de 1,000 caractères/sec …
- Pour écrire une ligne de texte sur l'imprimante, il faut combiner lasortie d'un car dans une boucle

Écrire une ligne de texte

Reg. destatut LSTAT = FFFFF130H

Reg. de données LOUT = FFFFF134H

Reg. de commande LCMD = FFFFF138H




```
      lar    r1, Buff      ;Pointeur vers une ligne de texte
      la     r2, 80        ;Compteur de caractères
      lar    r3, Wait      ;l'adresse de branchement
Wait: ld     r0, LSTAT     ;Lire le statut
      brpl   r3, r0        ;Imprimante prête?
      ld     r0, 0(r1)     ;Prochain caractère du tampon
      st     r0, LOUT      ;et l'envoyer
      addi   r1, r1, 4     ;Avancer pointeur au car. suivant
      addi   r2, r2, -1    ;et incrémenter compteur
      brnz   r3, r2        ;Si non dernier car., alors répéter
      la     r0, 1         ;Charger la commande d'"Imprimer"
      st     r0, LCMD      ;et l'envoyer à l'imprimante
```

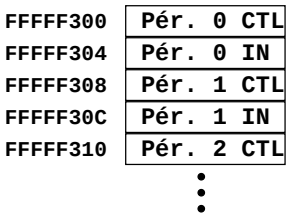
•Souvent une ligne de texte se termine avant 80 (ou 132) caractères - arrêter lorsque l'on trouve le car. "LF" ou "CR" dans le tampon. Où insérer cette vérification additionnelle?

•Comprimer 4 caractères dans chaqu mot ("packed character array")…?

Gestion de plusieurs périphériques (entrée) en parallèle

- 32 périphériques lents, p.e.
  - Clavier à 10 car/sec
  - Taux max. de 1 car. chaque 3 ms
- Chaque périphérique a un registre de commande et de statut
  - Seulement le bit 31 est utilisé - periphérique est prêt (sortie) ou un car. est disponible (entrée)
  - Le pilote d'E/S répétitivement examine (*polling*) les bits Ready et réagit en conséquence
- Chaque périphérique est représenté par un reg. de données
  - Bits 7..0 du DR d'un mot de 32 bits retient le car.
- Le logiciel obtient pointeurs aux tampons et retourne drapeaux *Done* (mis à 1 lorsque le car."CR"est reçu du périphérique)
  - le programme ne fait rien tant que le programme appelant ne remet Done à 0

Pilote pour 32 entrées...



- 32 paires des reg's CICNTL et CIN

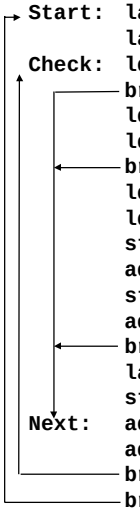
r0 - Reg. de travail; r1 - entrée du car.

r2 - Indice du périphérique; r3 - aucun actif

- 32 Paires pointeur au tampon, drapeau Done dans 64 mots consécutifs en mémoire

```
CICTL   .equ    FFFFF300H       ;1er reg.CICNTL
CIN     .equ    FFFFF304H       ;1er reg. CIN
CR      .equ    13              ;ASCII "carriage return"
Bufp:   .dcw    1               ;Adresse du 1er tampon
Done:   .dcw    63              ;1er Done et les autres paires
Driver: lar     r4, Next        ;Adr. de branch.- périph. suivant
        lar     r5, Check       ; Vérifie si pér. actif
        lar     r6, Start       ; prochain passage de vérification
```

Pilote pour 32 entrées… (suite)



```
Start:  la      r2, 0           ;Index du 1er pér.
        la      r3, 1           ;drapeau "tout inactif"
Check:  ld      r0,Done(r2)     ;est-ce pér. actif?
        brmi    r4, r0          ;si non,alors avance au pér.suivant
        ld      r3, 0           ;remet"tout inactif" à 0
        ld      r0,CICTL(r2)    ;Lire CICNTL
        brpl    r4, r0          ;avance au suivant si pas prêt
        ld      r0,CIN(r2)      ;Obtient l car. et le pointeur
        ld      r1,Bufp(r2)     ;au tampon correspondant
        st      r0, 0(r1)       ;Range le car.
        addi    r1,r1,4         ;Avance le ptr. de car.
        st      r1,Bufp(r2)     ;et le retourne à la mémoire
        addi    r0,r0,-CR       ;Vérifie si CR
        brnz    r4, r0          ;si non,avance au périph. suiv.
        la      r0, -1          ;Mettre Done à -1 après
        st      r0,Done(r2)     ;avoir détecté CR
Next:   addi    r2,r2,8         ;Avancer le ptr. de périphérique
        addi    r0,r2,-256      ;et si pas le dernier
        brnz    r5, r0          ;alors vérifie le suivant
        brzr    r6, r3          ;Si un pér. actif, pass suivant
```

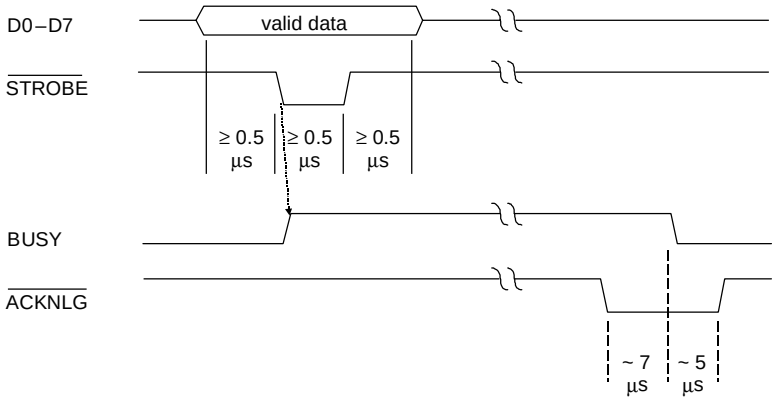

Pilote pour 32 entrées… analyse

- Si tout périphérique a un car. prêt
  - Alors 32 octets sont entrés en 547 instructions
  - Donc 585 KB/s avec un CPU de 10 MIPS
- Latence: Si le CPU juste manque le Ready, alors 538 instructions sont exécutées avant de vérifier de nouveau ce bit !
  - Délai de 53.8 µsec implique max. 18.6 Kcars/s par périphérique afin d'éviter une perte de données
- Les claviers (humaines?) sont suffisamment lents...

Interface parallèle d'imprimantes de Centronics

Interface Signal	Direction	Déscription
STROBE	**Out**	**Impulsion Data out (*strobe*)**
D0	**Out**	**Bit de données moins significatif**
D1	**Out**	**Données ...**
...	...	...
D7	**Out**	**Bit le plus significatif**
ACKNLG	**In**	**Impulsion avec le dernier car.**
BUSY	**In**	**Occupé**
PE	**In**	**Aucun papier si 1**
SLCT	**In**	**Constante 1**
AUTOFEEDXT	**Out**	**Avance auto à la ligne suivante**
INIT	**Out**	**Initialisation de l'imprimante**
ERROR	**In**	**Erreur si 0**
SLCTIN	**Out**	**Desélectionner le protocole**

Interface parallèle d'imprimantes de Centronics

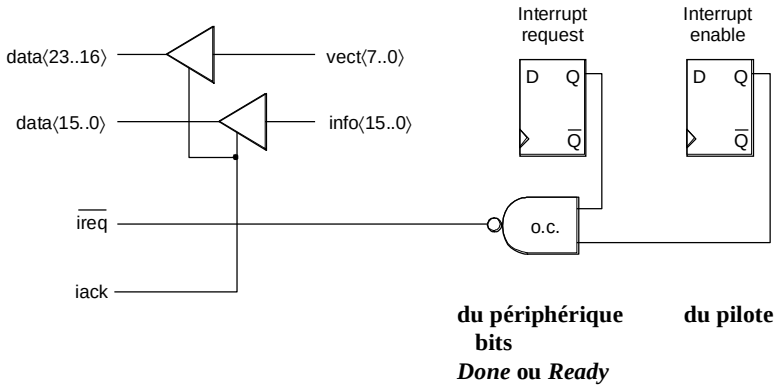


- Temps minimaux pour signaux de sortie
- Temps nominaux pour signaux d'entrée

Interruptions E/S

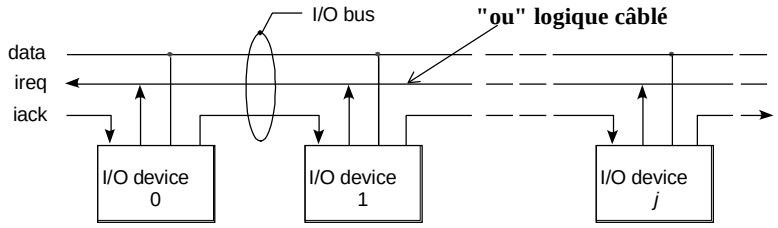
- Plutôt que le CPU attendre le périphérique, c'est son interface qui demande une interruption (forme d'exception) lorsqu'il est prêt
- SRC: l'interface doit fournir l'adresse du vecteur d'interruption et lesbits d'information
- LeCPU indique par un signal d'acquittement quand l'interface doit envoyer cette information
- Demande d'interruption et l'acquittement forment un protocole de communication - "handshake"
- On peut inhiber les interruption de chaque interfac périphérique individuellement - bits d'autorisation d'interruptions dans le registre de commande et/ou de statut (CNTL ou CMD)

Circuit simplifié de demande d'interruption



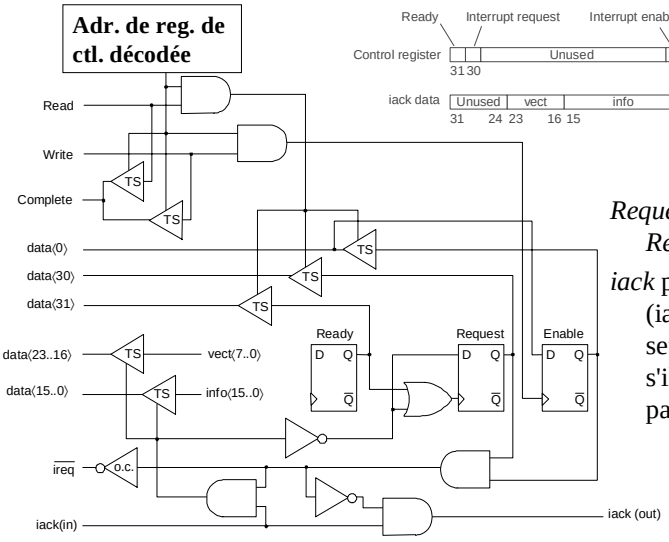
Chaînage d'interruptions

- **Plusieurs interfaces et une seule ligne d'interruption… comment organiser les priorités et l'acheminement d'acquittements?**
- **Solution simple - chaînage des acquittements (*daisy-chaining*)**
- **Interface plus éloigné du processeur a un plus base priorité**



bloque le passage de *iack* au suivant s'il demande *ireq*

Logique d'interface avec interruptions



Request mis à 1 par *Ready*, à 0 par *iack*

iack passé ausuivant (iack (out)) seulement s'il ne demande pas interruption

Fonctions de la procédure de service d'interruptions

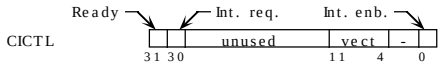
(1) Sauver l'état du programme interropmpu (essentiel!), possiblement inhiber d'autres interruptions

(2) Faire les opérations comme avec E/S programmée (sans l'attente)

(3) Redémarrer ou arrêter le périphérique asservi

(4) Restaurer l'état du programme et autoriser les interruptions, retourner au programme interrompu

Procédure d'E/S des caractères par interruption




```
;Getln est appelé avec l'adr. de retour dans R31 and un pointeur
;au tampon de carsdans R1. Ilva lire des cars jusqu'au "CR"
;avec l'aide d'interruptions, Done mis à -1 lorsque terminé
CR     .equ   13           ;code ASCII pour "CR"
CIvec  .equ   01F0H        ;adresse du vecteur d'interruption
Bufp:  .dw    1            ;Pointeur au prochain car. dans tampon
Save:  .dw    2            ;sauver regs ici
Done:  .dw    1            ;Mis à -1 si entrée complétée
Getln: st     r1, Bufp     ;Sauver le pointeur au tampon de cars
       edi                 ;Inhiber ints lorsque l'onchange masque
       la     r2, 1F1H     ;Charge adr. du vecteur et bit enable
       st     r2, CICTL    ;et dépose dans reg. de ctl.
       la     r3, 0        ;mise-à-0 du drapeau Done
       st     r3, Done     ;
       een                 ;autoriser interruptions
       br     r31          ;retourner au programme appelant
```

Routine de service d'interruption

```
        .org    CIvec           ;Routine à l'adresse du vecteur
        str     r0, Save        ;Sauver registres utiliés par routine
        str     r1, Save+4
        ldr     r1, Bufp        ;Pointeur à position dans tampon
        ld      r0, CIN         ;Obtient car et démarre périphérique
        st      r0, 0(r1)       ;Ranger car dans tampon
        addi    r1, r1, 4       ;Avancer pointeur
        str     r1, Bufp        ;et le ranger pour le prochain car
        lar     r1, Exit        ;Adr. de branchement
        addi    r0,r0, -CR      ;Est-ce "CR"? addi avec minus CR.
        brnz    r1, r0          ;Exit si non CR
        la      r0, 0           ;Inhiber périphérique
        st      r0, CICTL       ; en inhibant interruptions
        la      r0, -1          ;Indique -1 dans le drapeau Done
        str     r0, Done
Exit:   ldr     r0, Save        ;Restaurer registres
        ldr     r1, Save+4
        rfi                     ;Retour au programme interrompu
```


Temps de réponse d'interruptions

- Réponse à la prochaine interruption retardée jusqu'au rfi
- Routine inhibe interruption par au plus 17 instructions
- Avec l''horloge deCPU de 20MHz, il prend 10 cycles pour acquitter l'interruption, et le taux d'exécution moyen est 8 CPI

  La 2e interruption peut être retardée par

  $$(10 + 17 \times 8) / 20 = 7.3\ \mu\text{sec}$$

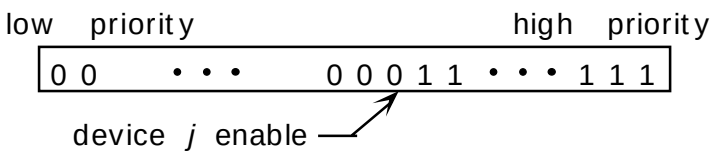
- Certaines périphériques rapides ne peuvent pas attendre la terminaison d'uneroutine de service d'un autre appareil
- Système de priorités, doit pouvoir interrompre un routine de service par une autre interruption - **interruptions imbriquées**
- Inhiber intrruptions seulement lors de manipulation de l'état du processeur ou du périphérique pour éviter des situations inconsistantes

Service pour interruptions imbriquées

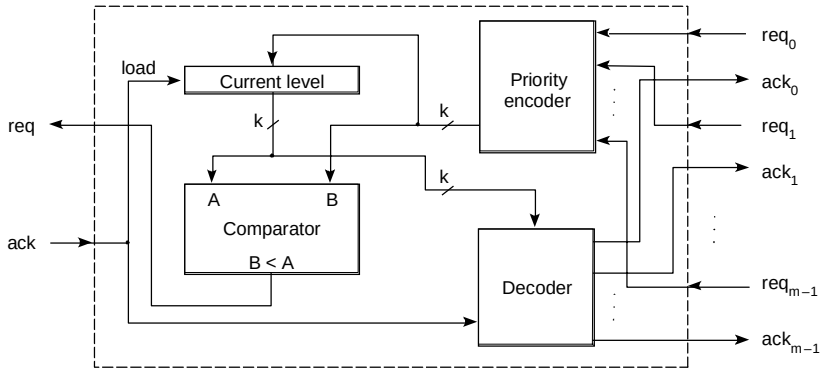
(1) Sauver l'état changé par interruption (IPC et II);
(2) Inhiber les iterruptions moins prioritaires;
(3) Autoriser le système d'interruptions global;
(4) Asservir le périphérique;
(5) Inhiber le système d'interruptions;
(6) Autoriser interruptions moins prioritaires;
(7) Restaurer IPC et II;
(8) Retourner au programme interrompu et autoriser le système d'interruptions.

Masque d'interruption

- Bits d'autorisation de périphériqus ordonnés par priorité peuvent former un système de priorité d'interruptions
- Formationd'un **masque d'interruption** avec ses bits
- Lors d'exécution de service du périphérique j la valeur du masque est la suivante



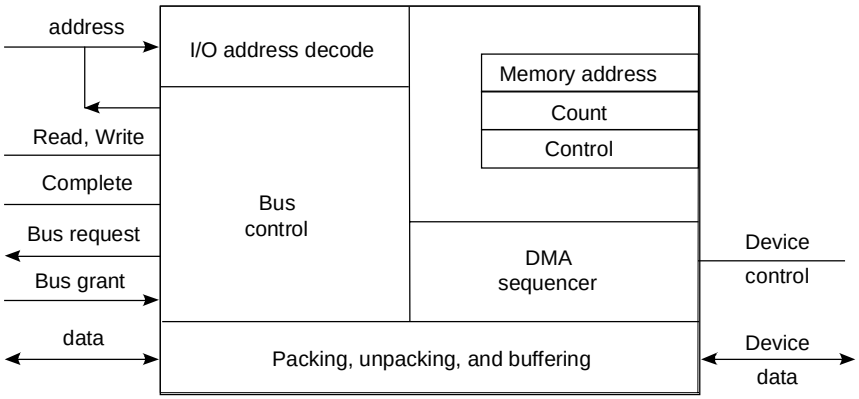
Système d'interruption avec m = 2^k niveaux



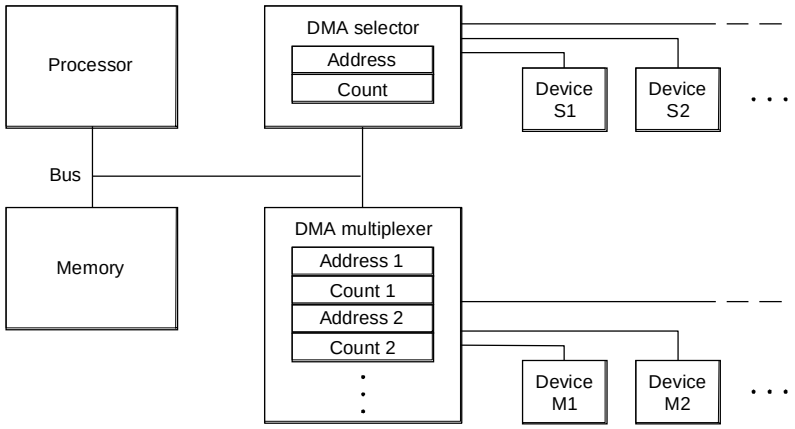
Accès direct à la mémoire

- Nécessaire pour des périphériques rapides (disques, video) - permet de transférer des données entre la mémoire et le périphérique sans l'intervention du processeur
- Requiert une interface ADM (*DMA*)
- Il doit être programmé et le transfert démarré par programme
- Gagne contrôle du bus mémoire pour effectuer un ou plusieurs cycles d'accès
- Réalise les opérations du transfert de données
- Interruption est générée seulement à la fin du transfert ou en cas d'erreur

Structure d'interface avec ADM



ADM: types sélecteur et multiplexeur



Processeurs E/S

- Pour décharger un superordinateur des tâches E/S
- Combiner tout contrôle des périphériques dans un appareil programmé - un processeur E/S
- Communication avec le CPU via une mémoire partagée et des registres de commande et de statut
- Souvent un petit ordinateur général, p.e., VAX-11 avec un Cray-2
- Sur un réseaux Internet … postes de travail serviteurs de calcul ou d'E/S - disques, imprimantes, etc.