Agenda du cours

- Cours 1 :
  - Généralités archi/assembleur
  - Manipulation émulateur
  - **Code, UAL, registres, mémoire**
  - Exécution, visualisation registres
- Cours 2 : Hiérarchie des mémoires
  - Différents **types de mémoires**
  - Accès mémoire (code, données, E/S)
  - Manipulation structure de données en assembleur
- Cours 3 : Appel de procédures
  - **Notion de Pile**
  - Appel de procédures
  - Passage de paramètres
  - Sauvegarde de contexte d'exécution
- Cours 4 : **Interruptions**
  - Mécanismes internes
  - Programmation d'Interruptions
  - Application aux E/S
- Cours 5 :
  - Développement programme
  - E/S , IT,..
- Cours 6 :
  - **Examen**

Introduction

- Gestion des entrées/sorties :
  - Transfert d'informations
  - Entre **processeur** ou **mémoire**
  - Et **organe périphérique** local ou distant

- Dans le système d'exploitation :
  - Ensemble de services
  - **« Gestionnaire de Périphériques »**

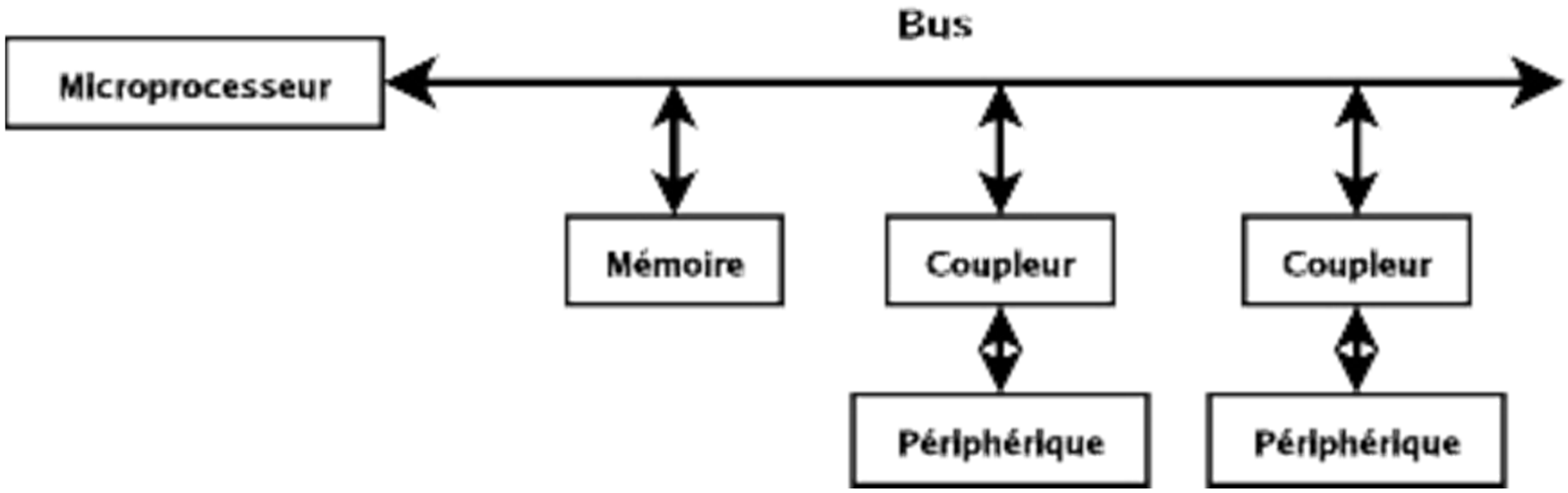
- Chaque équipement E/S est composé de deux parties
  - Un contrôleur
  - Le périphérique physique lui même

Les périphériques

- *Dispositifs servant à l'entrée ou à la sortie des données*
- Attachés à des *contrôleurs* ou *coupleurs*
- Utilisent une *interface* :
  - Bus de données
  - Signaux de commande
  - Incidents & Informations
  - Accès extérieur

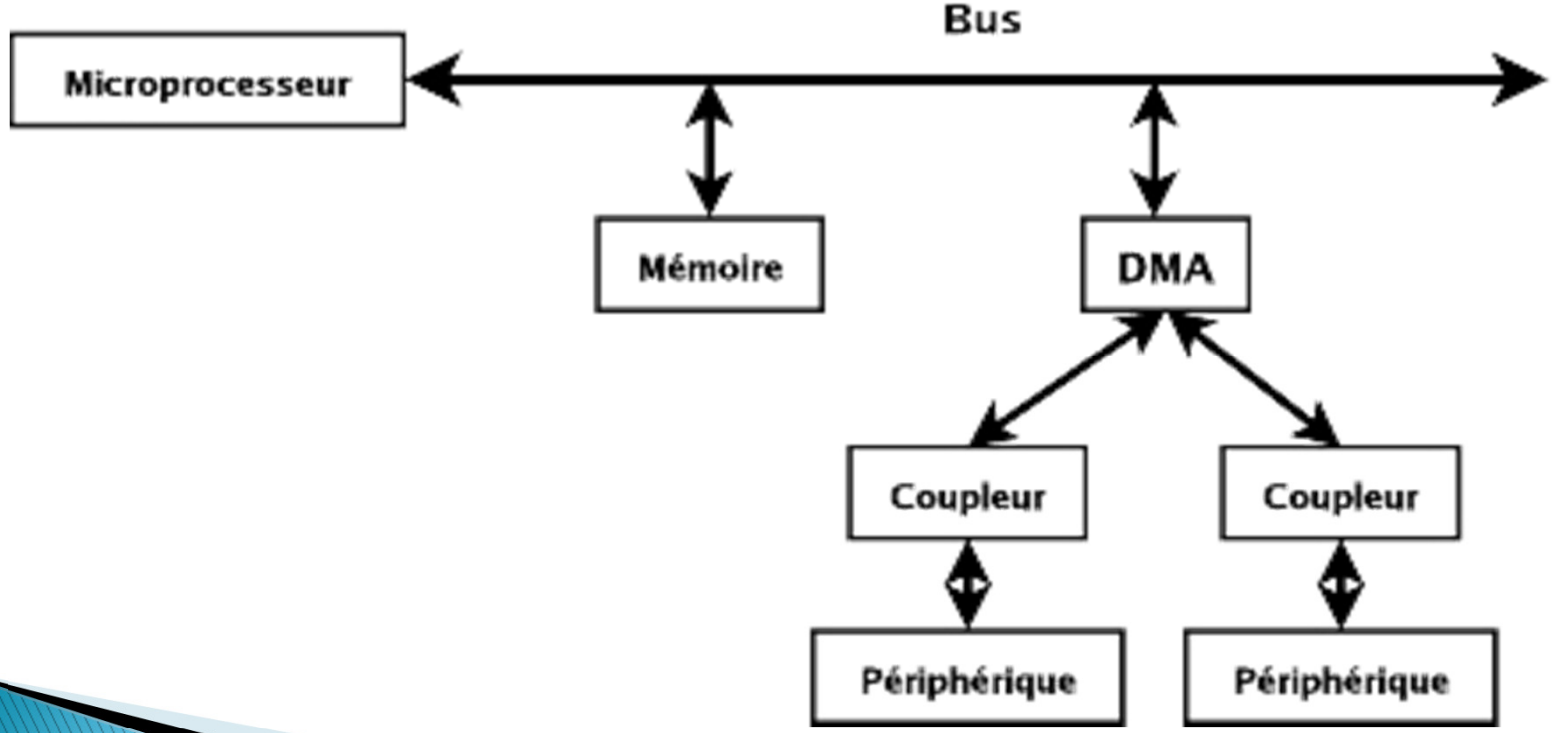
Organisations (1)

- Périphériques branchés via coupleurs
  - Décodage d'adresse
  - Pilotage du périphérique (*contrôleur*)



Organisations (2)

- Périphériques branchés par DMA + coupleur
  - Direct Memory Access
  - Coupleur garde son rôle de contrôleur



Différentes catégories de périphériques

- Périphériques caractères
  - Données = 1 octet
  - Données transmises **en série**.
  - Ex. : clavier, souris, …
- Périphériques de blocs
  - Données = bloc de plusieurs octets
  - Taille fixe (dépend du périphérique)
    128 octets → plusieurs Ko
  - Possibilité de lire/écrire n'importe quel bloc (accès **aléatoire**)
  - Ex. : Disque dur, carte vidéo, carte réseau, …

Coupleurs & Contrôleurs

- Le μP donne les ordres généraux
  - Lire 1ko sur disque à partir de l'@ 3000.
- Le contrôleur pilote le périphérique
  - Avancer la tête de lecture
  - Attendre la rotation du disque
  - Lire pendant 0,1 seconde
  - Envoyer les données sur le bus
- Le contrôleur peut être un autre processeur, ou un microcontrôleur
  - Exemple des cartes vidéos

Espace d'adressage

- Chaque contrôleur a une **adresse**.
- Pour le processeur, 2 possibilités :
  - Adresses partagées avec la mémoire
    Ex. sur le 68000, @=0→1FFFh
  - Adresses et/ou signaux de contrôle séparés
    - Lecture ou écriture en mémoire
    - Lecture ou écriture sur périphérique
    - Instructions séparées (*In*/*Out* Vs *Mov* sur Intel x86)
- En général :
  - Beaucoup de mémoire
  - Peu de périphériques

Pilotes de périphériques

- Programme qui pilote un périphérique (*Driver* ou *Handler*)
- Gère directement l'interface du coupleur
- Traite les interruptions du coupleur
- Détecte et traite les erreurs
- Offre des primitives au programmeur
  - Commande du périphérique
  - Informations sur le périphérique

E/S Synchrones

- Pas de parallélisme entre commande et transfert des données
  - Le μP donne l'ordre
  - Le μP transfère les données
  - Le périphérique travaille
  - Le μP reçoit les données
- Périphériques lents (Vs Processeur)
  - ➔ perte de puissance/vitesse
  - Utilisées dans les cas les plus simple (rien d'autre à faire en attendant)

E/S asynchrones

- Gestion du contrôleur par **interruption**
  - Ex. : Clavier → 1 interruption par touche
- Ne mobilise pas le processeur
  - Lance la commande
  - Transfère les données
  - Fait autre chose (le périphérique travaille)
  - Est interrompu (le périphérique a fini)
  - Reprend son travail en cours…

Les interruptions matérielles

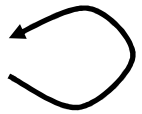
- Signal envoyé par un périphérique au processeur
  - Ex. : touche clavier sur IBM-PC
    port 60_h → interruption n°9
- Processeur reçoit une interruption :
  - Arrête son travail en cours (mémorise contexte)
  - Exécute une routine d'interruption :
    programme spécialisé situé à une @ précise
  - Restaure le contexte sauvegardé
  - Reprend son travail
- Plusieurs périphériques, 1 processeur
  - → *Programmable Interrupt Controler* (PIC)
  - gère les priorités, transmet au processeur

Transfert de données

- Ex. : Périphérique → Mémoire centrale
  - μP initie l'accès au périphérique
  - Périph. récupère les données (vérifie CRC, etc. …)
  - Interrompt le processeur
  - Envoie les données au processeur
  - μP envoie les données à la mémoire
  - Relâche le processeur
- → pas efficace

Accès direct à la mémoire (DMA)

- Optimisation de l'algorithme :
  - μP initie l'accès au périphérique
    Précise @ mémoire où stocker les données
  - Périph. récupère les données
    (vérifie CRC, etc. …)
  - Envoie les données *à la mémoire*
  - Interrompt le processeur
    « *Transfert terminé* »
- Efficace
  - Surtout pour les périphériques de type bloc



Deux types de Transfert DMA

- Par giclée (*Burst*)
  - Récupère les données
  - Réquisitionne le bus mémoire
  - Effectue plusieurs transferts
  - Rend le bus mémoire
  - Méthode adaptée aux périphériques « bloc »
- Par vol de cycle
  - Récupère les données (Processeur ⇔ mémoire)
  - Transfert les données (Processeur attend)
  - Méthode adaptée aux périphériques rapides à transferts rapide (sériels ?)

Les canaux / unités d'échange

- Généralisation du DMA
- Utilise un processeur spécialisé
  - Esclave du processeur central
  - Jeu d'instructions spécialisé :
    - Activer coupleurs
    - Transférer données
  - Prévient le processeur central quand transfert fini. (Interruption)

Les canaux / unités d'échange

- Les périphériques sont lents
- Les processeurs sont rapides
- ➔ Processeurs attendent Périphériques
- ➔ Ajout de tampons (*Buffers, FIFO*)
  - Zone mémoire entre périphérique et μP
  - Périphérique stocke dans tampon (lent)
  - Processeur lit dans tampon (grande vitesse)
- **Découple** le processeur du périphérique

Double-Buffering

- Lecture/Ecriture simultanée dans tampon
- ➔ Conflits → Perte d'efficacité.
- Utilisation d'un double-tampon :
  - Processeur stocke dans un tampon (1)
  - Périphérique lit un autre tampon (2)
  - Le double-tampon recopie le (1) dans le (2)