Organisation Interne d'un ordinateur

Denis Bouhineau Fabienne Carrier Stéphane Devismes

Université Grenoble Alpes

13 avril 2020

Plan

1. Introduction
2. Contrôleur d'entrées-sorties
3. Décodage d'adresses
4. Mémoire cache
5. Mémoire virtuelle

Plan

1 Introduction

2 Contrôleur d'entrées-sorties

3 Décodage d'adresses

4 Mémoire cache

5 Mémoire virtuelle

Organisation de l'ordinateur

Il existe différentes sortes d'ordinateurs :

Organisation de l'ordinateur

Il existe différentes sortes d'ordinateurs :

- Les *PC*
- Les ordinateurs portables

Organisation de l'ordinateur

Il existe différentes sortes d'ordinateurs :

- Les *PC*
- Les ordinateurs portables
- Les serveurs
- Les super-calculateurs

Organisation de l'ordinateur

Il existe différentes sortes d'ordinateurs :

- Les *PC*
- Les ordinateurs portables
- Les serveurs
- Les super-calculateurs
- Les téléphones portables
- Dans l'électro-ménager, ...

Organisation de l'ordinateur

Il existe différentes sortes d'ordinateurs :

- Les *PC*
- Les ordinateurs portables
- Les serveurs
- Les super-calculateurs
- Les téléphones portables
- Dans l'électro-ménager, ...

On parle souvent de systèmes embarqués **pour désigner un ordinateur qui ne ressemble pas à un ordinateur !** C'est-à-dire, pas de clavier, pas de souris, pas de disque, pas d'écran, mais un processeur avec un programme

Organisation de l'ordinateur

Il existe différentes sortes d'ordinateurs :

- Les *PC*
- Les ordinateurs portables
- Les serveurs
- Les super-calculateurs
- Les téléphones portables
- Dans l'électro-ménager, ...

On parle souvent de systèmes embarqués **pour désigner un ordinateur qui ne ressemble pas à un ordinateur !** C'est-à-dire, pas de clavier, pas de souris, pas de disque, pas d'écran, mais un processeur avec un programme

Exemple : Le contrôleur de distribution d'essence d'une station service.

Critères de classification

Critères de classification

- Peut-on ajouter des programmes et les lancer ? (cartouche de jeu, par exemple ...)

Critères de classification

- Peut-on ajouter des programmes et les lancer ? (cartouche de jeu, par exemple ...)
- Peut-on ajouter des programmes en langage machine écrits, compilés et assemblés soi-même ?

Critères de classification

- Peut-on ajouter des programmes et les lancer ? (cartouche de jeu, par exemple ...)
- Peut-on ajouter des programmes en langage machine écrits, compilés et assemblés soi-même ?
- Y-a-t-il des mémoires secondaires ? (disque dur, clé USB, CD-Rom, . . .)

Points communs

Il y a des points communs :

Points communs

Il y a des points communs :

- Processeur

Points communs

Il y a des points communs :

- Processeur
- Mémoire

Points communs

Il y a des points communs :

- Processeur
- Mémoire
- Bus adresses, données

Points communs

Il y a des points communs :

- Processeur
- Mémoire
- Bus adresses, données
- Signaux de contrôle ($Read/\overline{Write}$, autres),

Points communs

Il y a des points communs :

- Processeur
- Mémoire
- Bus adresses, données
- Signaux de contrôle ($Read/\overline{Write}$, autres),
- ...
- Contrôleurs d'entrées-sorties
- Décodeur

Plan

1 Introduction

2 Contrôleur d'entrées-sorties

3 Décodage d'adresses

4 Mémoire cache

5 Mémoire virtuelle

Vision externe (exemple)

- Un simple circuit avec 4 registres qui fait l'interface entre le périphérique et la machine (niveau OS)
- Le circuit est vu par le processeur (espace d'adressage) comme 4 mots :
  - `Mcommande` (un mot de commande)
  - `Métat` (un mot d'état)
  - `Mdonnéessort` (données sortantes)
  - `Mdonnéesentr` (données entrantes)
- Géré par un logiciel : le pilote

Lectures/écriture par le processeur (via le pilote)

Ecriture Lors d'une écriture (instruction `STORE`) dans un des trois registres `Mcommande, Métat, Mdonnéessort`, le processeur envoie la donnée à écrire dans le bus de donnée puis la donnée est transférée du bus de donnée au registre du circuit contrôleur d'Entrées-Sorties.

Lectures/écriture par le processeur (via le pilote)

Ecriture Lors d'une écriture (instruction `STORE`) dans un des trois registres `Mcommande`, `Métat`, `Mdonnéessort`, le processeur envoie la donnée à écrire dans le bus de donnée puis la donnée est transférée du bus de donnée au registre du circuit contrôleur d'Entrées-Sorties.

Lecture Lors d'une lecture (instruction `LOAD`) dans le registre `Mdonnéesentr` ou dans le registre `Métat`, le processeur récupère le contenu du registre `Mdonnéesentr` (resp. `Métat`) via le bus de donnée.

Lectures/écriture par le processeur (via le pilote)

Ecriture Lors d'une écriture (instruction `STORE`) dans un des trois registres `Mcommande, Métat, Mdonnéessort`, le processeur envoie la donnée à écrire dans le bus de donnée puis la donnée est transférée du bus de donnée au registre du circuit contrôleur d'Entrées-Sorties.

Lecture Lors d'une lecture (instruction `LOAD`) dans le registre `Mdonnéesentr` ou dans le registre `Métat`, le processeur récupère le contenu du registre `Mdonnéesentr` (resp. `Métat`) via le bus de donnée.

Les 4 mots (`Mcommande, Métat, Mdonnéessort, Mdonnéesentr`) seront supposés aux adresses CNTRL, CNTRL+1, +2, +3.

Attention : `CNTRL` est une adresse constante, pas déterminée par l'assembleur. Elle a un sens physique qu'on va voir maintenant. ..

Notations

Commande	**Signification**
`LOAD` reg, `Métat`	METTRE l'adresse `CNTRL` dans un registre `regad` puis `LOAD` reg, `[regad+1]`
`STORE` reg, `Mdonnéessort`	METTRE l'adresse `CNTRL` dans un registre `regad` puis `STORE` reg, `[regad+2]`

Utilisation du contrôleur d'entrées-sorties : exemple

Dans la documentation technique d'un contrôleur, certaines valeurs sont prédéfinies.

Utilisation du contrôleur d'entrées-sorties : exemple

Dans la documentation technique d'un contrôleur, certaines valeurs sont prédéfinies.

Considérons ici l'exemple (simpliste) d'un contrôleur graphique avec les valeurs prédéfinies suivantes :

- *libre* : cette valeur **dans le mot d'état** signifie que le contrôleur est prêt à accepter une commande. Le contrôleur tient à jour son état de disponibilité pour que le processeur puisse savoir si il est occupé ou non.
- *fond_écran* : cette valeur, **envoyée vers le mot de commande**, affiche un fond d'écran de la couleur contenue dans le registre de données sortantes.
- *rouge* : code de la couleur rouge, **peut être contenu dans le registre de données sortantes**.

Exercice

Ecrire un programme qui change la couleur de fond d'écran en rouge via le contrôleur graphique.

Exercice

Ecrire un programme qui change la couleur de fond d'écran en rouge via le contrôleur graphique.

recom :
```
LOAD reg, Métat
CMP reg, libre
BNE recom
STORE rouge, Mdonnéessort
STORE fond_écran, Mcommande
```

Exercice

Ecrire un programme qui change la couleur de fond d'écran en rouge via le contrôleur graphique.

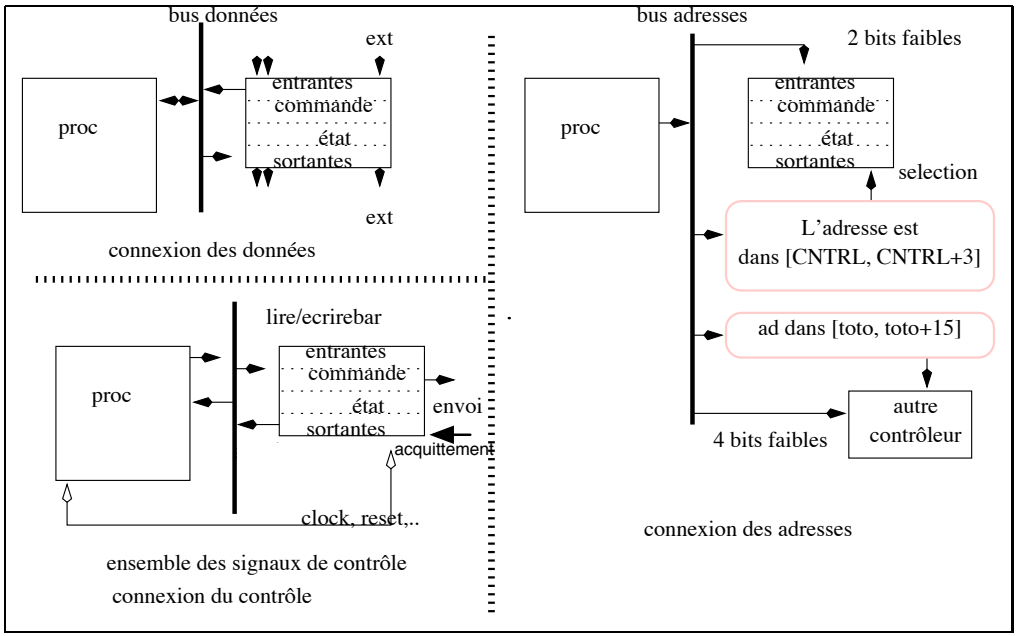
recom : `LOAD` reg, `Métat`
`CMP` reg, *libre*
`BNE` recom
`STORE` *rouge*, `Mdonnéessort`
`STORE` *fond_écran*, `Mcommande`

Remarque : On aurait de même des actions d'envoi :

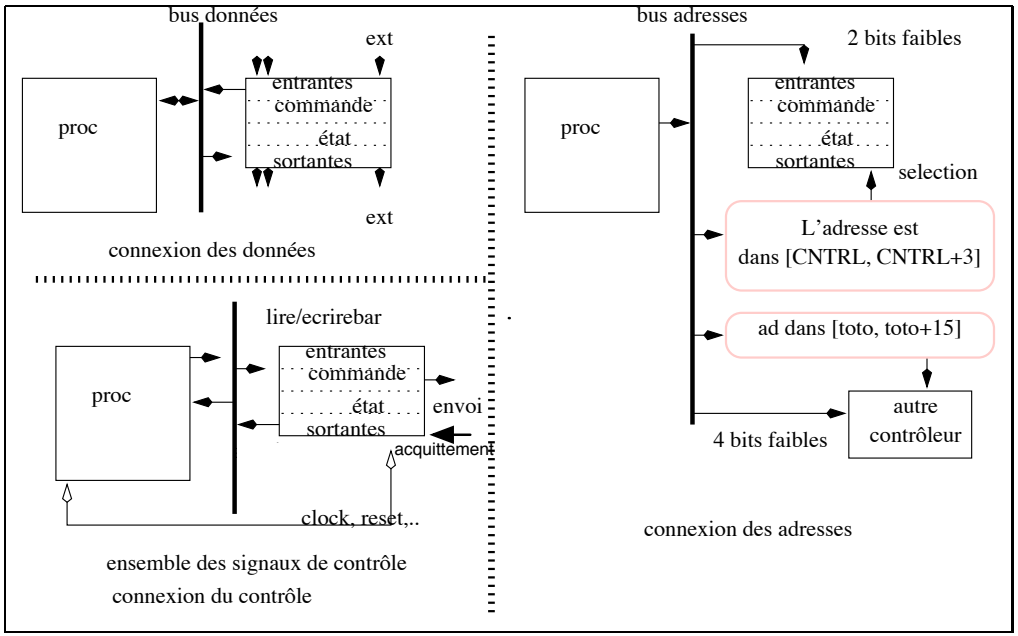
- d'un caractère à l'écran,
- de positionnement de la tête de lecture d'un disque dur,
- ...

Les contrôleurs réels sont parfois très simples, parfois très complexes (simple affichage comme ci-dessus, contrôleur réseau...) parfois pour une seule commande, plusieurs données...

Etude du matériel d'entrées-sorties : *les entrées*

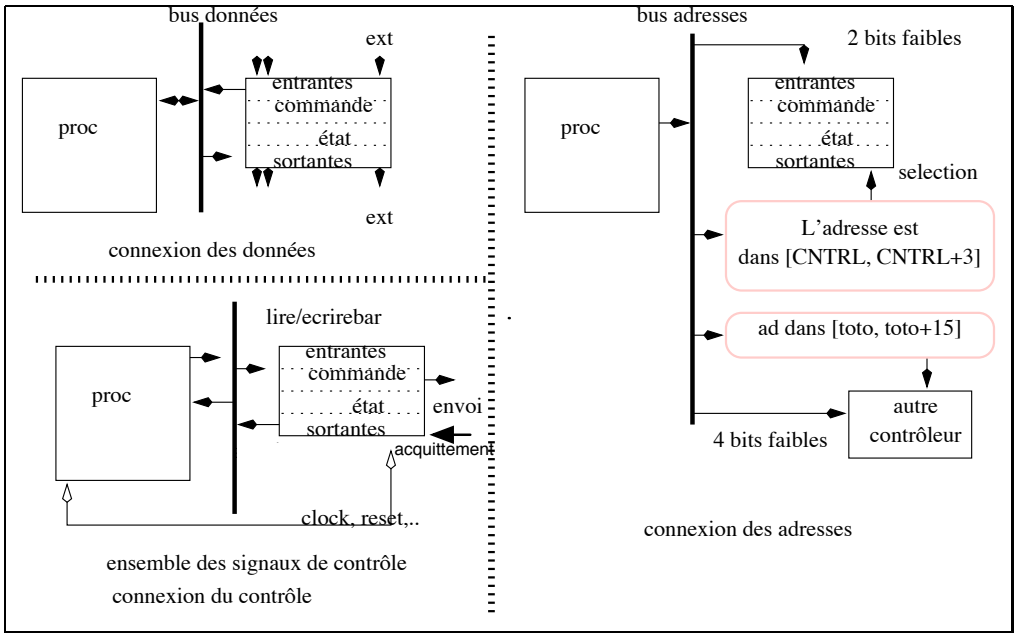


Etude du matériel d'entrées-sorties : *les entrées*



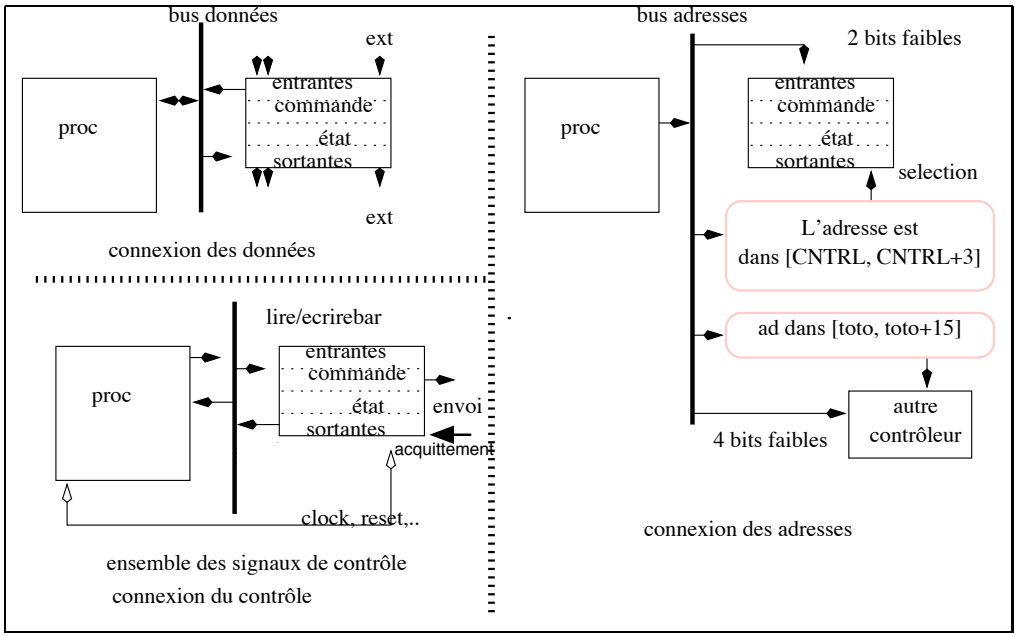
- bus données (lié au processeur)

Etude du matériel d'entrées-sorties : *les entrées*



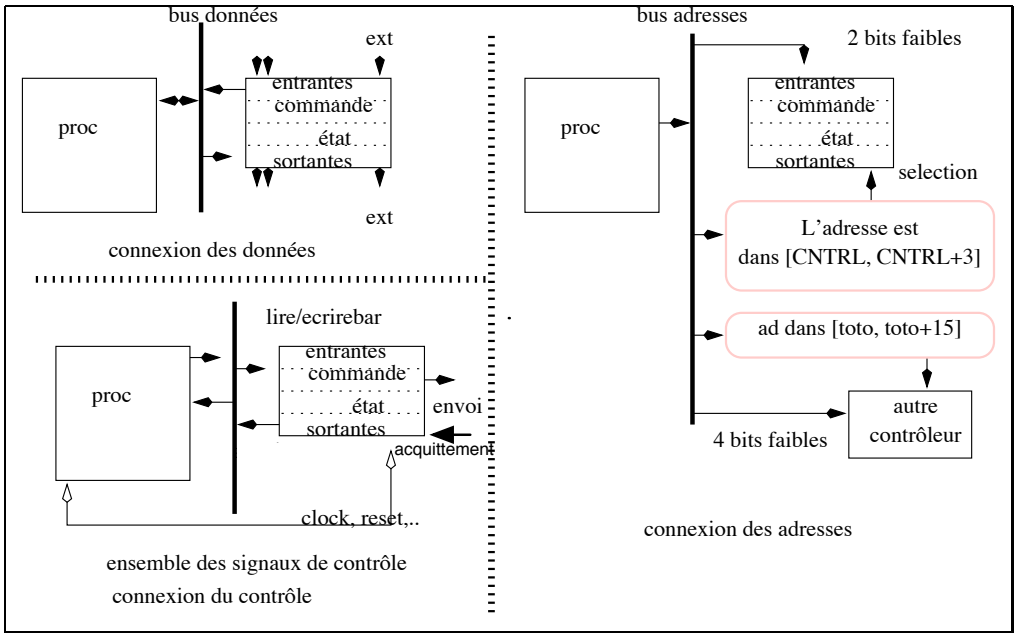
- bus données (lié au processeur)
- deux bits de bus adresses (pour sélectionner l'un des 4 mots `CNTRL` +0, +1, +2 ou +3)

Etude du matériel d'entrées-sorties : *les entrées*



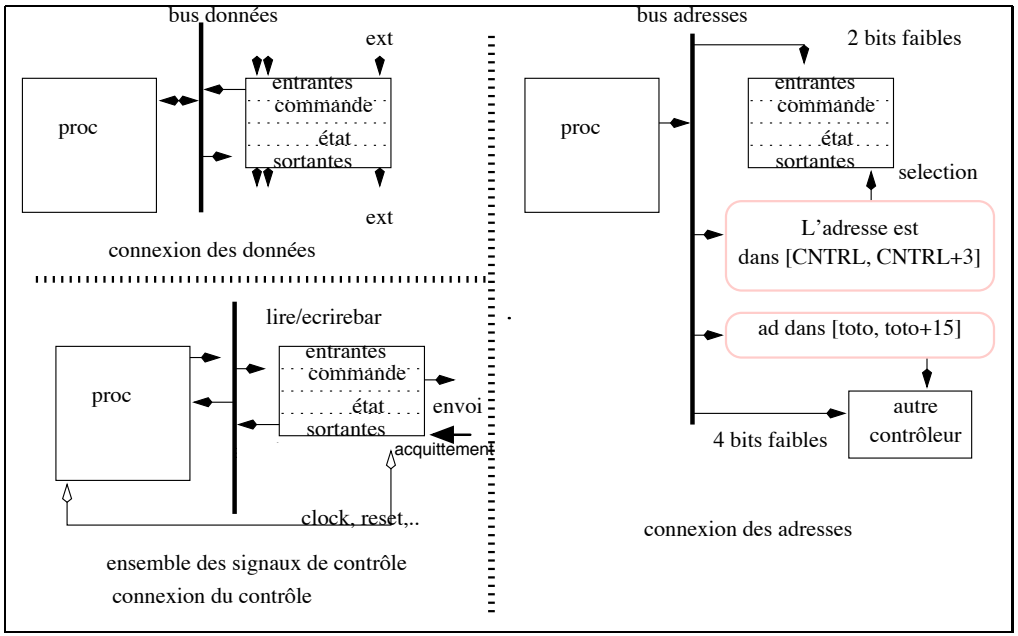
- bus données (lié au processeur)
- deux bits de bus adresses (pour sélectionner l'un des 4 mots `CNTRL` +0, +1, +2 ou +3)
- un signal de sélection provenant du décodeur d'adresses

Etude du matériel d'entrées-sorties : *les entrées*



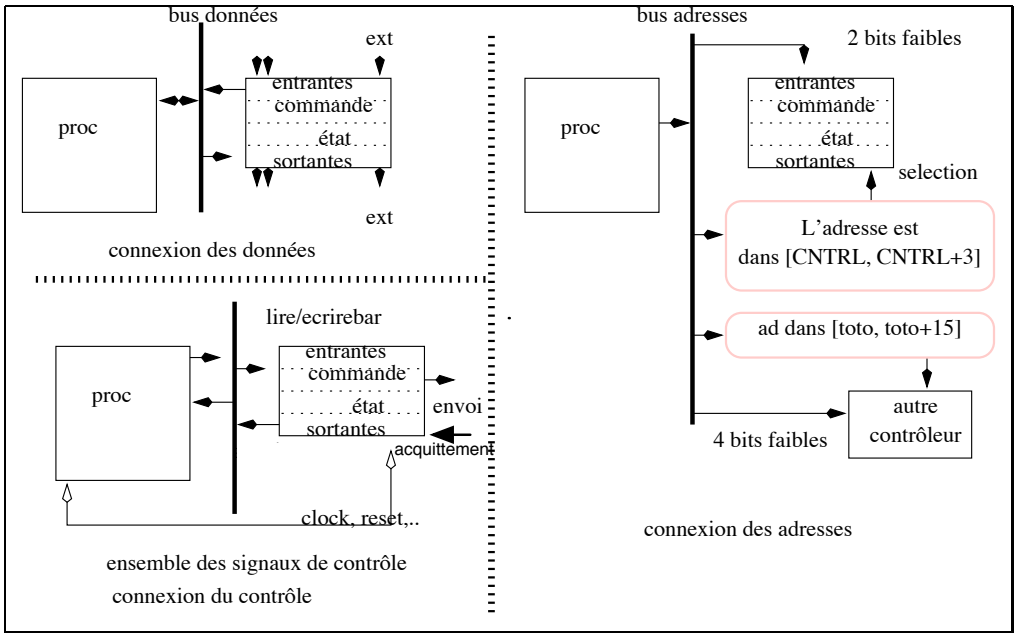
- bus données (lié au processeur)
- deux bits de bus adresses (pour sélectionner l'un des 4 mots `CNTRL` +0, +1, +2 ou +3)
- un signal de sélection provenant du décodeur d'adresses
- le signal *Read*/$\overline{\textit{Write}}$ du processeur

Etude du matériel d'entrées-sorties : *les entrées*



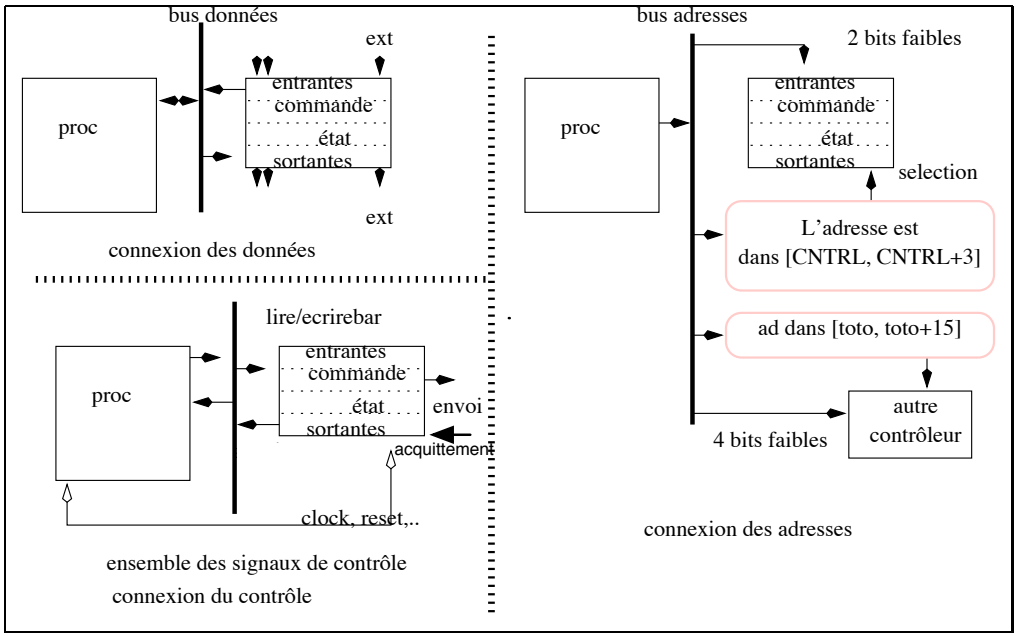
- bus données (lié au processeur)
- deux bits de bus adresses (pour sélectionner l'un des 4 mots `CNTRL` +0, +1, +2 ou +3)
- un signal de sélection provenant du décodeur d'adresses
- le signal *Read*/$\overline{\textit{Write}}$ du processeur
- un paquet de données (8 fils) venant du monde extérieur. Disons pour simplifier 8 interrupteurs

Etude du matériel d'entrées-sorties : *les entrées*



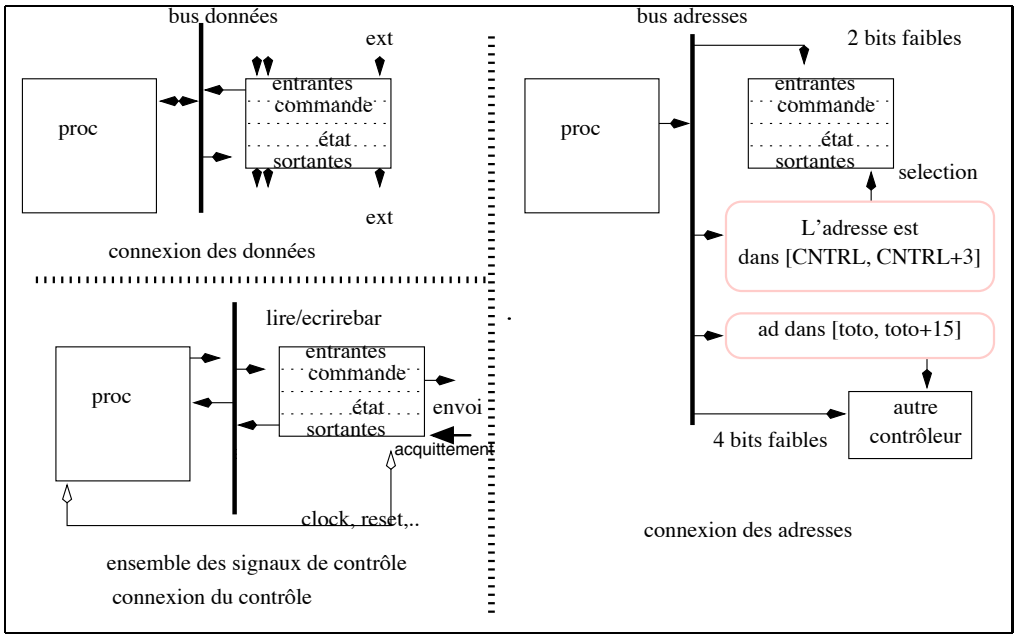
- bus données (lié au processeur)
- deux bits de bus adresses (pour sélectionner l'un des 4 mots `CNTRL` +0, +1, +2 ou +3)
- un signal de sélection provenant du décodeur d'adresses
- le signal *Read*/$\overline{\textit{Write}}$ du processeur
- un paquet de données (8 fils) venant du monde extérieur. Disons pour simplifier 8 interrupteurs
- le signal d'horloge (par exemple le même que le processeur). On peut raisonner comme si, à chaque front de l'horloge la valeur venant des interrupteurs était échantillonnée dans le registre `Mdonnéesentr`.

Etude du matériel d'entrées-sorties : *les entrées*

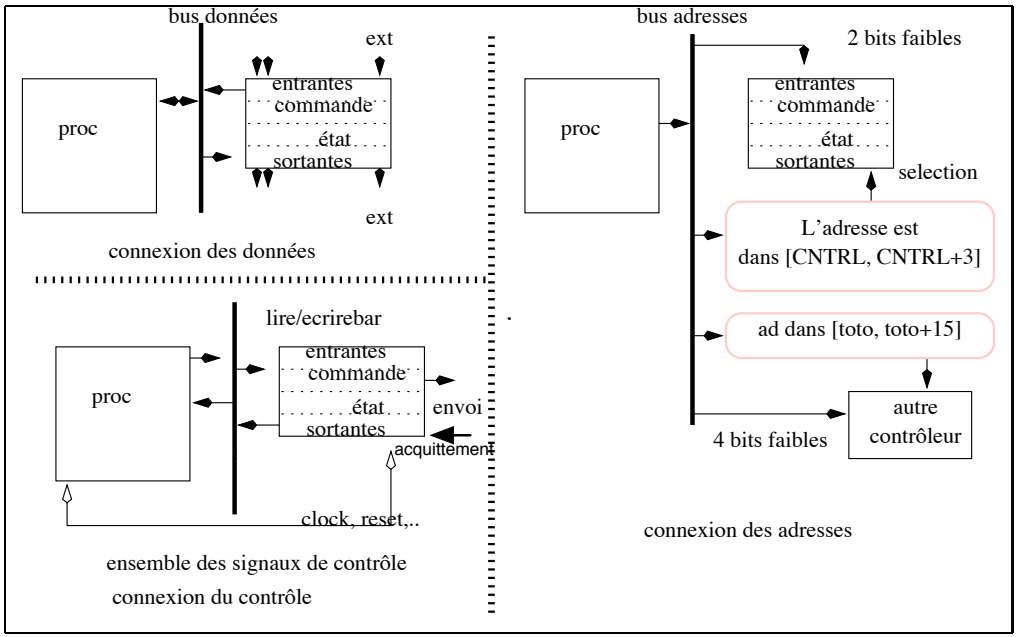


- bus données (lié au processeur)
- deux bits de bus adresses (pour sélectionner l'un des 4 mots `CNTRL` +0, +1, +2 ou +3)
- un signal de sélection provenant du décodeur d'adresses
- le signal *Read*/$\overline{\textit{Write}}$ du processeur
- un paquet de données (8 fils) venant du monde extérieur. Disons pour simplifier 8 interrupteurs
- le signal d'horloge (par exemple le même que le processeur). On peut raisonner comme si, à chaque front de l'horloge la valeur venant des interrupteurs était échantillonnée dans le registre `Mdonnéesentr`.
- une entrée ACQUITTEMENT si c'est un contrôleur de sortie.

Etude du matériel d'entrées-sorties : *les sorties*

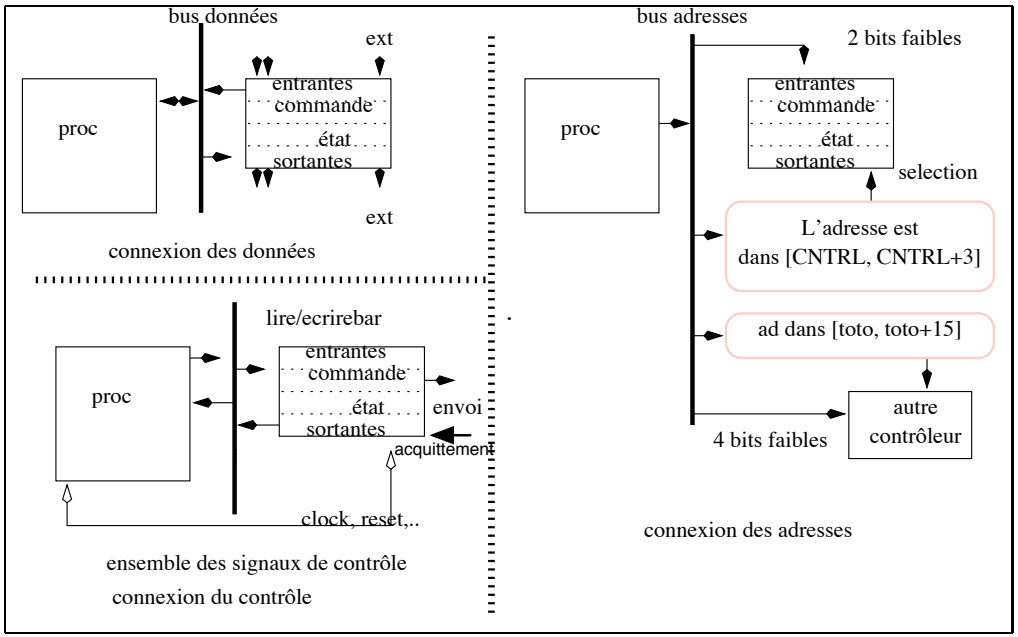


Etude du matériel d'entrées-sorties : *les sorties*



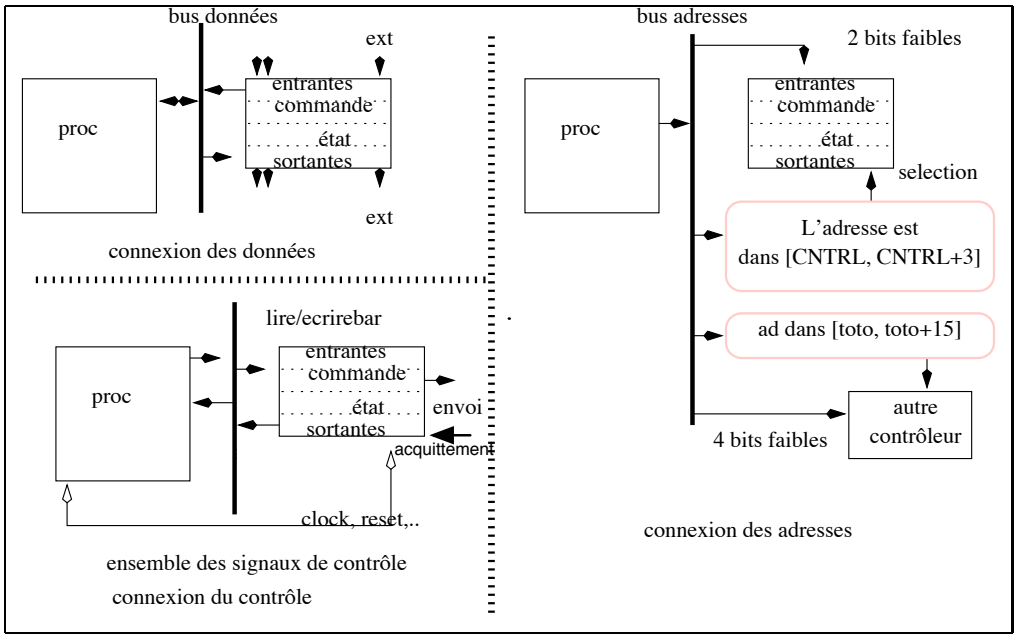
- Il délivre sur le bus données du processeur le contenu du registre `Mdonnéesentr` si il y a sélection, lecture et adressage de `Mdonnéesentr`, c'est-à-dire si le processeur exécute une instruction `LOAD` à l'adresse `CNTRL` +3

Etude du matériel d'entrées-sorties : *les sorties*



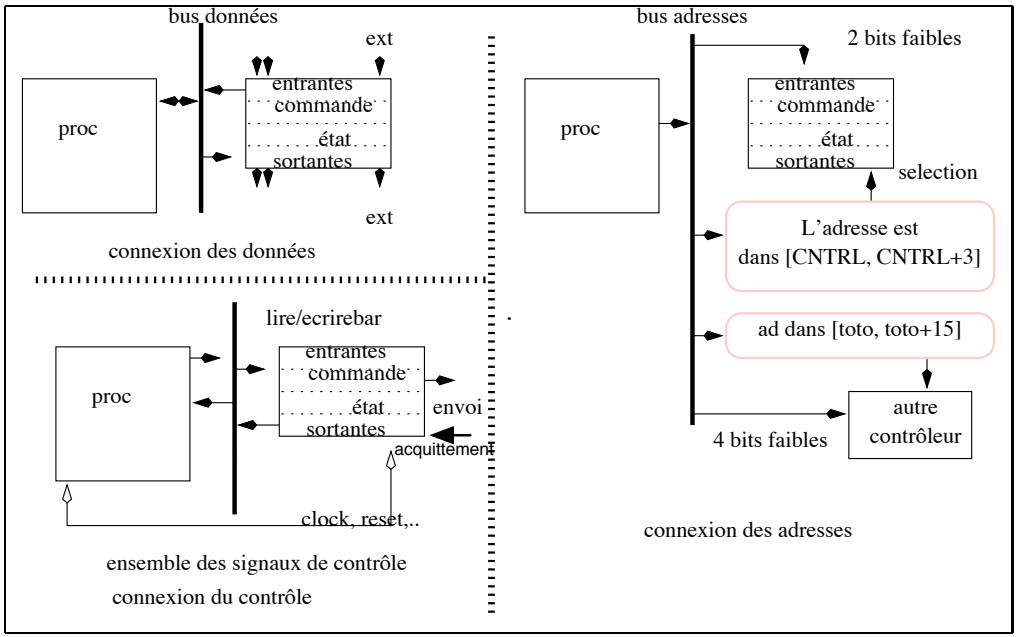
- Il délivre sur le bus données du processeur le contenu du registre `Mdonnéesentr` si il y a sélection, lecture et adressage de `Mdonnéesentr`, c'est-à-dire si le processeur exécute une instruction `LOAD` à l'adresse `CNTRL` $+3$
- Il délivre sur le bus données du processeur le contenu du registre `Métat` si il y a sélection, lecture et adressage de `Métat`, c'est-à-dire si le processeur exécute une instruction `LOAD` à l'adresse `CNTRL` $+1$.

Etude du matériel d'entrées-sorties : *les sorties*



- Il délivre sur le bus données du processeur le contenu du registre `Mdonnéesentr` si il y a sélection, lecture et adressage de `Mdonnéesentr`, c'est-à-dire si le processeur exécute une instruction `LOAD` à l'adresse `CNTRL` $+3$
- Il délivre sur le bus données du processeur le contenu du registre `Métat` si il y a sélection, lecture et adressage de `Métat`, c'est-à-dire si le processeur exécute une instruction `LOAD` à l'adresse `CNTRL` $+1$.
- On peut raisonner comme si le contenu du registre `Mdonnéessort` était affiché en permanence sur 8 pattes de sorties vers l'extérieur (8 diodes, par exemple).

Etude du matériel d'entrées-sorties : *les sorties*



- Il délivre sur le bus données du processeur le contenu du registre `Mdonnéesentr` si il y a sélection, lecture et adressage de `Mdonnéesentr`, c'est-à-dire si le processeur exécute une instruction `LOAD` à l'adresse `CNTRL` +3
- Il délivre sur le bus données du processeur le contenu du registre `Métat` si il y a sélection, lecture et adressage de `Métat`, c'est-à-dire si le processeur exécute une instruction `LOAD` à l'adresse `CNTRL` +1.
- On peut raisonner comme si le contenu du registre `Mdonnéessort` était affiché en permanence sur 8 pattes de sorties vers l'extérieur (8 diodes, par exemple).
- Une sortie ENVOI si c'est un contrôleur de sortie.

Plan

1 Introduction

2 Contrôleur d'entrées-sorties

3 Décodage d'adresses

4 Mémoire cache

5 Mémoire virtuelle

Exemple

00	selrom
01	selrom
02	selrom
...	
3E	selrom
3F	selrom
40	/////
...	
57	/////
58	selcntr
59	selcntr
5A	selcntr
5B	selcntr
5C	/////
...	
5F	/////
60	/////
...	
7E	/////
7F	/////
80	selram
81	selram
...	
FE	selram
FF	selram

Exemple

00	selrom
01	selrom
02	selrom
...	
3E	selrom
3F	selrom
40	/////
...	
57	/////
58	selcntr
59	selcntr
5A	selcntr
5B	selcntr
5C	/////
...	
5F	/////
60	/////
...	
7E	/////
7F	/////
80	selram
81	selram
...	
FE	selram
FF	selram

- On raisonne avec des adresses sur 8 bits.

Exemple

00	selrom
01	selrom
02	selrom
...	
3E	selrom
3F	selrom
40	/////
...	
57	/////
58	selcntr
59	selcntr
5A	selcntr
5B	selcntr
5C	/////
...	
5F	/////
60	/////
...	
7E	/////
7F	/////
80	selram
81	selram
...	
FE	selram
FF	selram

- On raisonne avec des adresses sur 8 bits.
- L'adresse A est codé sur 8 bits $A7,\ldots,A0$.

Exemple

00	selrom
01	selrom
02	selrom
...	
3E	selrom
3F	selrom
40	/////
...	
57	/////
58	selcntr
59	selcntr
5A	selcntr
5B	selcntr
5C	/////
...	
5F	/////
60	/////
...	
7E	/////
7F	/////
80	selram
81	selram
...	
FE	selram
FF	selram

- On raisonne avec des adresses sur 8 bits.
- L'adresse A est codé sur 8 bits $A7,\ldots,A0$.
- On représente les valeurs portées par ces 8 fils par deux chiffres hexadécimaux.

Exemple

00	selrom
01	selrom
02	selrom
...	
3E	selrom
3F	selrom
40	/////
...	
57	/////
58	selcntr
59	selcntr
5A	selcntr
5B	selcntr
5C	/////
...	
5F	/////
60	/////
...	
7E	/////
7F	/////
80	selram
81	selram
...	
FE	selram
FF	selram

- On raisonne avec des adresses sur 8 bits.
- L'adresse A est codé sur 8 bits $A7,\ldots,A0$.
- On représente les valeurs portées par ces 8 fils par deux chiffres hexadécimaux.
- On note val_x une valeur représentée en hexa.

Exemple

00	selrom
01	selrom
02	selrom
...	
3E	selrom
3F	selrom
40	/////
...	
57	/////
58	selcntr
59	selcntr
5A	selcntr
5B	selcntr
5C	/////
...	
5F	/////
60	/////
...	
7E	/////
7F	/////
80	selram
81	selram
...	
FE	selram
FF	selram

- On raisonne avec des adresses sur 8 bits.
- L'adresse A est codé sur 8 bits $A7,\ldots,A0$.
- On représente les valeurs portées par ces 8 fils par deux chiffres hexadécimaux.
- On note val_x une valeur représentée en hexa.
- On dira que l'entrée $A \ll$ vaut $\gg 67_x$ pour dire que les 8 fils sont dans les états 0110 0111.

Sélecteur pour la ROM

Un mot (lu ou écrit) est pris dans la mémoire morte (ROM) si et seulement si :

- l'entrée A est telle que $0_x \leq A \leq 3F_x$
- c'est-à-dire $A7$ $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ sont compris entre 0 0 0 0 0 0 0 0 et 0 0 1 1 1 1 1 1

Sélecteur pour la ROM

Un mot (lu ou écrit) est pris dans la mémoire morte (ROM) si et seulement si :

- l'entrée A est telle que $0_x \leq A \leq 3F_x$
- c'est-à-dire $A7$ $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ sont compris entre 0 0 0 0 0 0 0 0 et 0 0 1 1 1 1 1 1

Ceci couvre les 64 monômes canoniques pour lesquels $A7 = 0$ et $A6 = 0$.

Sélecteur pour la ROM

Un mot (lu ou écrit) est pris dans la mémoire morte (ROM) si et seulement si :

- l'entrée A est telle que $0_x \leq A \leq 3F_x$
- c'est-à-dire $A7$ $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ sont compris entre 0 0 0 0 0 0 0 0 et 0 0 1 1 1 1 1 1

Ceci couvre les 64 monômes canoniques pour lesquels $A7 = 0$ et $A6 = 0$.

Le boîtier physique de ROM, qui contient 64 mots reçoit deux types d'information :

- les 6 fils A5 A4 A3 A2 A1 A0 qui permettent d'adresser un des 64 mots
- Un fil de sélection `selrom` qui vaut 1 si et seulement si $A7 = 0$ et $A6 = 0$. Donc `selrom` $= \overline{A7}.\overline{A6}$.

Sélecteur pour le contrôleur d'entrées-sorties

Sélecteur pour le contrôleur d'entrées-sorties

Le mot (lu ou écrit) est pris dans le contrôleur d'entrées-sorties (vu précédemment) si et seulement si :

- l'entrée A est telle que $58_x \leq A \leq 5B_x$ (On a choisi que $CNTRL = 58_x$)
- c'est-à-dire $A7$ $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ sont compris entre 0 1 0 1 1 0 0 0 et 0 1 0 1 1 0 1 1

Ceci couvre les 4 monômes canoniques pour lesquels $A7 = 0$, $A6 = 1$, $A5 = 0$, $A4 = 1$, $A3 = 1$, $A2 = 0$.

Sélecteur pour le contrôleur d'entrées-sorties

Le mot (lu ou écrit) est pris dans le contrôleur d'entrées-sorties (vu précédemment) si et seulement si :

- l'entrée A est telle que $58_x \leq A \leq 5B_x$ (On a choisi que $CNTRL = 58_x$)
- c'est-à-dire $A7$ $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ sont compris entre 0 1 0 1 1 0 0 0 et 0 1 0 1 1 0 1 1

Ceci couvre les 4 monômes canoniques pour lesquels $A7 = 0$, $A6 = 1$, $A5 = 0$, $A4 = 1$, $A3 = 1$, $A2 = 0$.

Le boîtier physique de contrôleur, qui contient 4 mots, reçoit deux types d'information :

- les 2 fils $A1$ $A0$ qui permettent d'adresser un des 4 mots
- Un fil de sélection `selcntr` qui vaut 1 si et seulement si $A7 = 0$, $A6 = 1$, $A5 = 0$, $A4 = 1$, $A3 = 1$, $A2 = 0$. Donc, `selcntr` $= \overline{A7}.A6.\overline{A5}.A4.A3.\overline{A2}$.

Sélecteur pour la RAM

Sélecteur pour la RAM

Le mot (lu ou écrit) est pris dans la mémoire vive (RAM) si et seulement si :

- l'entrée A est telle que $80_x \leq A \leq FF_x$
- c'est-à-dire $A7$ $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ sont compris entre 1 0 0 0 0 0 0 0 et 1 1 1 1 1 1 1 1

Ceci couvre les 128 monômes canoniques pour lesquels $A7 = 1$.

Sélecteur pour la RAM

Le mot (lu ou écrit) est pris dans la mémoire vive (RAM) si et seulement si :

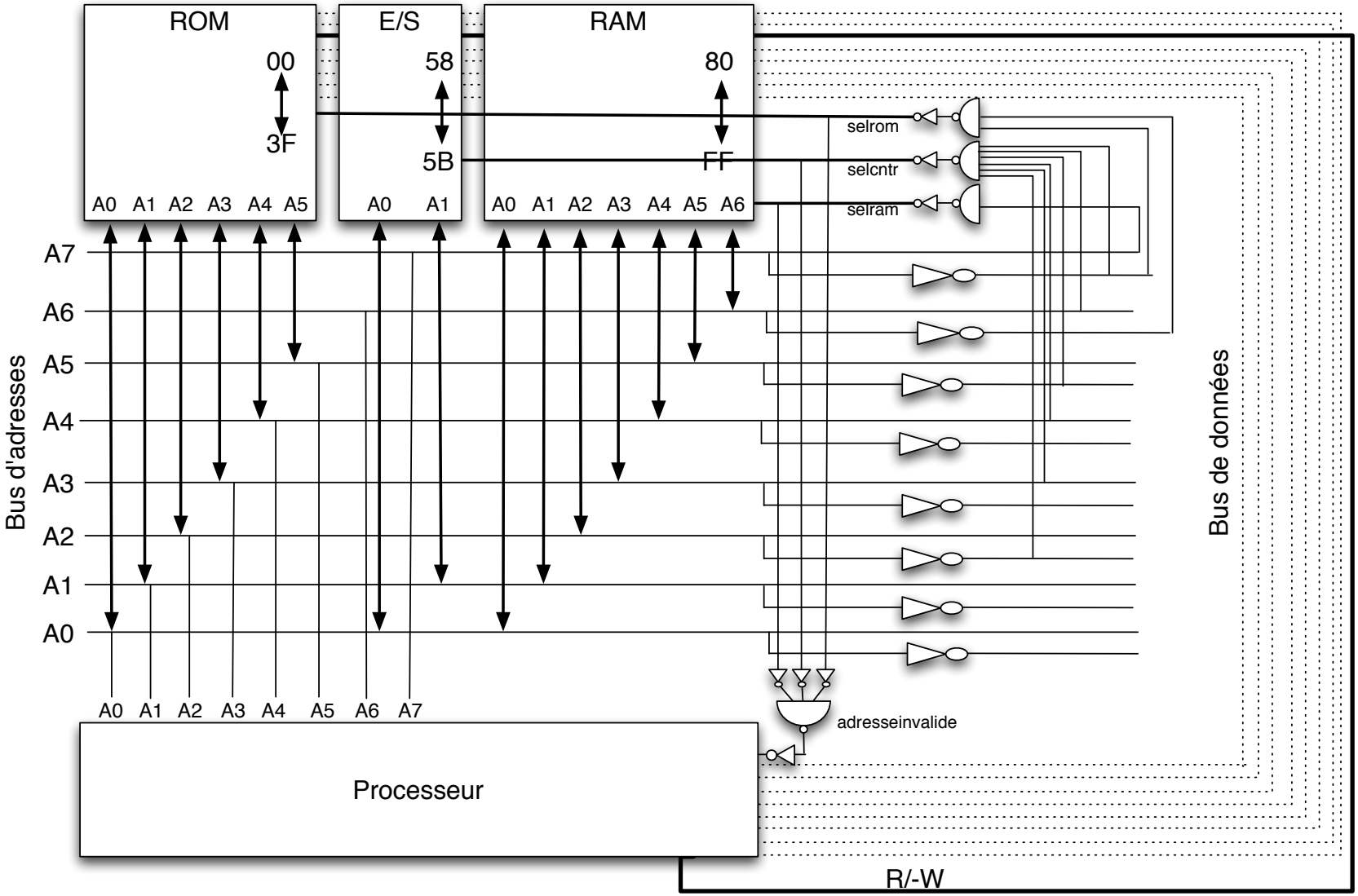
- l'entrée A est telle que $80_x \leq A \leq FF_x$
- c'est-à-dire $A7$ $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ sont compris entre 1 0 0 0 0 0 0 0 et 1 1 1 1 1 1 1 1

Ceci couvre les 128 monômes canoniques pour lesquels $A7 = 1$.

Le boîtier physique de RAM, qui contient 128 mots, reçoit deux types d'information :

- les 7 fils $A6$ $A5$ $A4$ $A3$ $A2$ $A1$ $A0$ qui permettent d'adresser un des 128 mots
- Un fil de sélection `selram` qui vaut 1 si et seulement si $A7 = 1$. Donc, `selram` $= A7$.

Connexions processeur/contrôleur/mémoires/décodeur



Plan

1 Introduction

2 Contrôleur d'entrées-sorties

3 Décodage d'adresses

4 Mémoire cache

5 Mémoire virtuelle

Pour aller plus vite ...

La mémoire est lente, chère et prend de la place !

Mais, principes de localité :

Pour aller plus vite ...

La mémoire est lente, chère et prend de la place !

Mais, principes de localité :

- Principe de localité spatiale :

Pour aller plus vite ...

La mémoire est lente, chère et prend de la place !

Mais, principes de localité :

- Principe de localité spatiale :
  - l'accès à une donnée X va probablement être suivi d'accès à d'autres données Y, Z proches de X.

Pour aller plus vite ...

La mémoire est lente, chère et prend de la place !

Mais, principes de localité :

- Principe de localité spatiale :
  - l'accès à une donnée X va probablement être suivi d'accès à d'autres données Y, Z proches de X.
  - **exemple :** instructions, éléments de tableau.

Pour aller plus vite ...

La mémoire est lente, chère et prend de la place !

Mais, principes de localité :

- Principe de localité spatiale :
  - l'accès à une donnée X va probablement être suivi d'accès à d'autres données Y, Z proches de X.
  - **exemple :** instructions, éléments de tableau.
- Principe de localité temporelle :

Pour aller plus vite ...

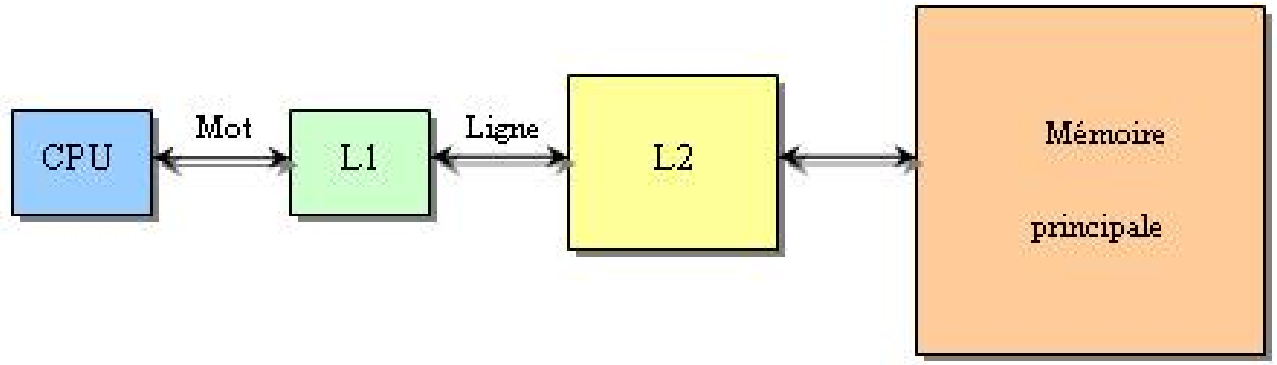
La mémoire est lente, chère et prend de la place !

Mais, principes de localité :

- Principe de localité spatiale :
  - l'accès à une donnée X va probablement être suivi d'accès à d'autres données Y, Z proches de X.
  - **exemple :** instructions, éléments de tableau.
- Principe de localité temporelle :
  - l'accès à une donnée X à un instant donné va probablement être suivi d'autres accès à cette même donnée.

Organisation mémoire

Structuration en caches de plusieurs niveaux



source wikipedia.

Pour le processeur, il n'y a qu'une mémoire : la Mémoire.

Vie du cache

Le cache répond aux demandes du processeur, **en lecture** :

Vie du cache

Le cache répond aux demandes du processeur, **en lecture** :

- si la donnée est disponible : ok

Vie du cache

Le cache répond aux demandes du processeur, **en lecture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible

Vie du cache

Le cache répond aux demandes du processeur, **en lecture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible
  - il faut aller la chercher en mémoire

Vie du cache

Le cache répond aux demandes du processeur, **en lecture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible
  - il faut aller la chercher en mémoire
  - faire de la place dans le cache (quelle autre donnée enlever ?)

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

- si la donnée est disponible : ok

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible
  - il faut aller la chercher en mémoire

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible
  - il faut aller la chercher en mémoire
  - faire de la place dans le cache (quelle autre donnée enlever ?)

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible
  - il faut aller la chercher en mémoire
  - faire de la place dans le cache (quelle autre donnée enlever ?)

puis faire les modifications

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible
  - il faut aller la chercher en mémoire
  - faire de la place dans le cache (quelle autre donnée enlever ?)

puis faire les modifications

- Écriture immédiate en mémoire ?

Vie du cache

Le cache répond aux demandes du processeur, **en écriture** :

- si la donnée est disponible : ok
- si la donnée n'est pas disponible
  - il faut aller la chercher en mémoire
  - faire de la place dans le cache (quelle autre donnée enlever ?)

puis faire les modifications

- Écriture immédiate en mémoire ?
- Écriture différée ?

Cas particuliers : Données *vs* Programme

Cas particuliers : Données *vs* Programme

- **Données :**

Cas particuliers : Données *vs* Programme

- **Données :**
  - modifiable

Cas particuliers : Données *vs* Programme

- **Données :**
  - modifiable
  - principe de localité respecté pour toutes les variables dans la pile (même lors des appels de fonction)

Cas particuliers : Données *vs* Programme

- **Données :**
  - modifiable
  - principe de localité respecté pour toutes les variables dans la pile (même lors des appels de fonction)
- **Programme :**

Cas particuliers : Données *vs* Programme

- **Données :**
  - modifiable
  - principe de localité respecté pour toutes les variables dans la pile (même lors des appels de fonction)
- **Programme :**
  - non modifiable (en général)

Cas particuliers : Données *vs* Programme

- **Données :**
  - modifiable
  - principe de localité respecté pour toutes les variables dans la pile (même lors des appels de fonction)
- **Programme :**
  - non modifiable (en général)
  - principe de localité rompu lors des appels de fonction

Cas particuliers : Données *vs* Programme

- **Données :**
  - modifiable
  - principe de localité respecté pour toutes les variables dans la pile (même lors des appels de fonction)
- **Programme :**
  - non modifiable (en général)
  - principe de localité rompu lors des appels de fonction

Conclusion : séparation des caches Données - Programme

Plan

1 Introduction

2 Contrôleur d'entrées-sorties

3 Décodage d'adresses

4 Mémoire cache

5 Mémoire virtuelle

Pour aller plus loin ...

Pour le processeur, la mémoire n'est pas idéale ...

Pour aller plus loin ...

Pour le processeur, la mémoire n'est pas idéale ...

- La mémoire physique réelle (RAM) est trop petite

Pour aller plus loin ...

Pour le processeur, la mémoire n'est pas idéale ...

- La mémoire physique réelle (RAM) est trop petite
  - alors que la mémoire secondaire (HD) est immense

Pour aller plus loin ...

Pour le processeur, la mémoire n'est pas idéale ...

- La mémoire physique réelle (RAM) est trop petite
  - alors que la mémoire secondaire (HD) est immense
- Chaque processus voudrait avoir la mémoire à lui tout seul, identique d'une fois sur l'autre

Pour aller plus loin ...

Pour le processeur, la mémoire n'est pas idéale ...

- La mémoire physique réelle (RAM) est trop petite
  - alors que la mémoire secondaire (HD) est immense
- Chaque processus voudrait avoir la mémoire à lui tout seul, identique d'une fois sur l'autre
- Il faudrait pouvoir partager certaines zones de la mémoire

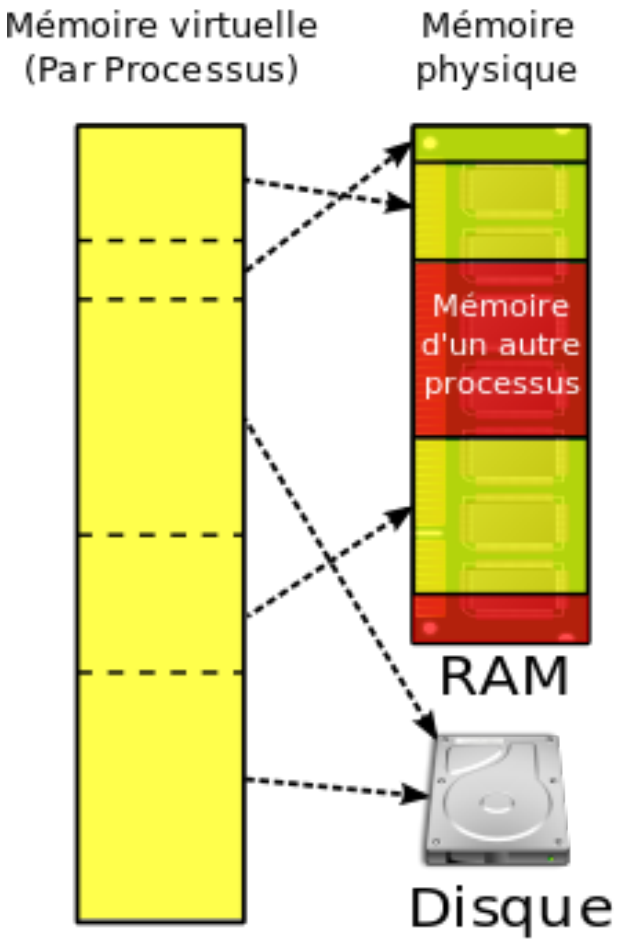
Pour aller plus loin ...

Pour le processeur, la mémoire n'est pas idéale ...

- La mémoire physique réelle (RAM) est trop petite
  - alors que la mémoire secondaire (HD) est immense
- Chaque processus voudrait avoir la mémoire à lui tout seul, identique d'une fois sur l'autre
- Il faudrait pouvoir partager certaines zones de la mémoire
- Il faudrait pouvoir protéger certaines zones de la mémoire

Extension de la mémoire physique

La mémoire vue du processeur
et en vrai ... (source wikipedia)



Partage de la mémoire et protection

Mécanisme de pagination

Partage de la mémoire et protection

Mécanisme de pagination

- La mémoire réelle est organisée en pages

Partage de la mémoire et protection

Mécanisme de pagination

- La mémoire réelle est organisée en pages
- Une adresse mémoire virtuelle (idéale) est associé à un couple **[numéro de page, adresse dans la page]**

Partage de la mémoire et protection

Mécanisme de pagination

- La mémoire réelle est organisée en pages
- Une adresse mémoire virtuelle (idéale) est associé à un couple **[numéro de page, adresse dans la page]**
- Les pages peuvent être protégées

Partage de la mémoire et protection

Mécanisme de pagination

- La mémoire réelle est organisée en pages
- Une adresse mémoire virtuelle (idéale) est associé à un couple **[numéro de page, adresse dans la page]**
- Les pages peuvent être protégées
- Les pages peuvent être partagées

Illustration

Partage du code entre trois processus exécutant le même programme, chacun sur des données particulières.

Illustration

Partage du code entre trois processus exécutant le même programme, chacun sur des données particulières.

- Le programme est constitué de 3 pages de code

Illustration

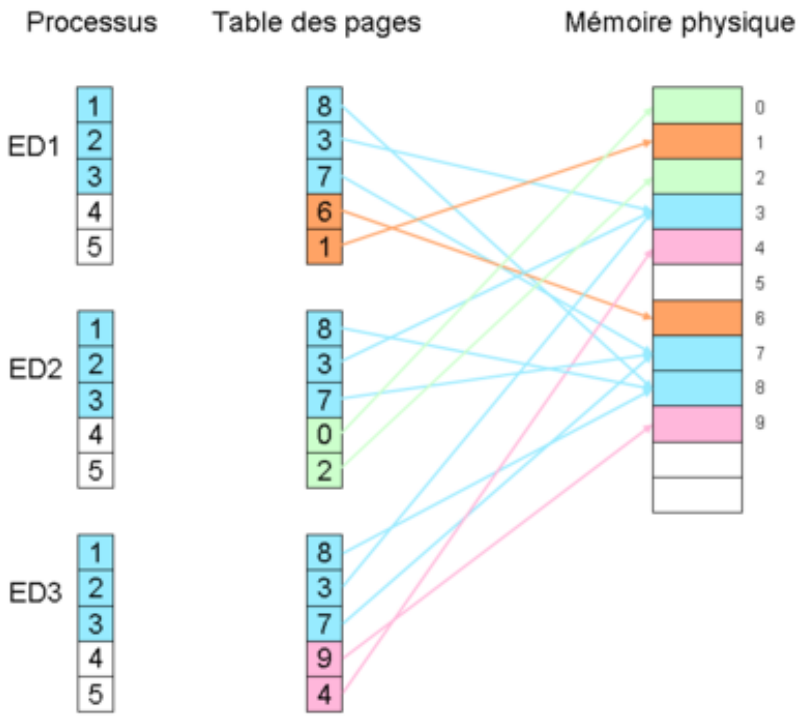
Partage du code entre trois processus exécutant le même programme, chacun sur des données particulières.

- Le programme est constitué de 3 pages de code
- Le programme comporte 2 pages de données

Illustration

Partage du code entre trois processus exécutant le même programme, chacun sur des données particulières.

- Le programme est constitué de 3 pages de code
- Le programme comporte 2 pages de données



source wikipedia.

Remarque finale

Une partie de ce cours était à la frontière entre Architecture et Système.

Remarque finale

Une partie de ce cours était à la frontière entre Architecture et Système.

- **Architecture :** parce qu'il y a des circuits impliqués

Remarque finale

Une partie de ce cours était à la frontière entre Architecture et Système.

- **Architecture :** parce qu'il y a des circuits impliqués
- **Système :** pour certains algorithmes (allocation) et concepts (processus)