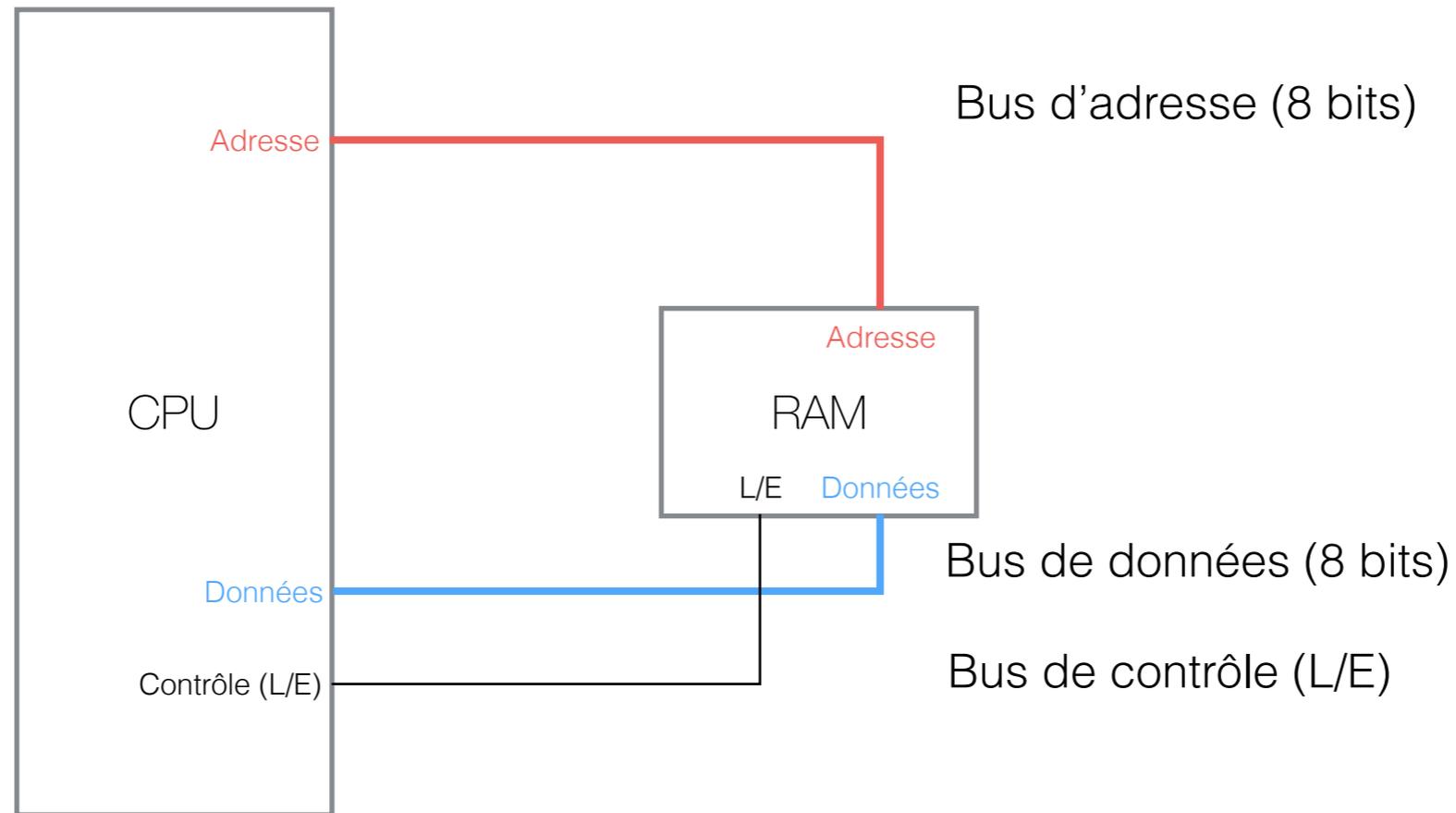


Adressage



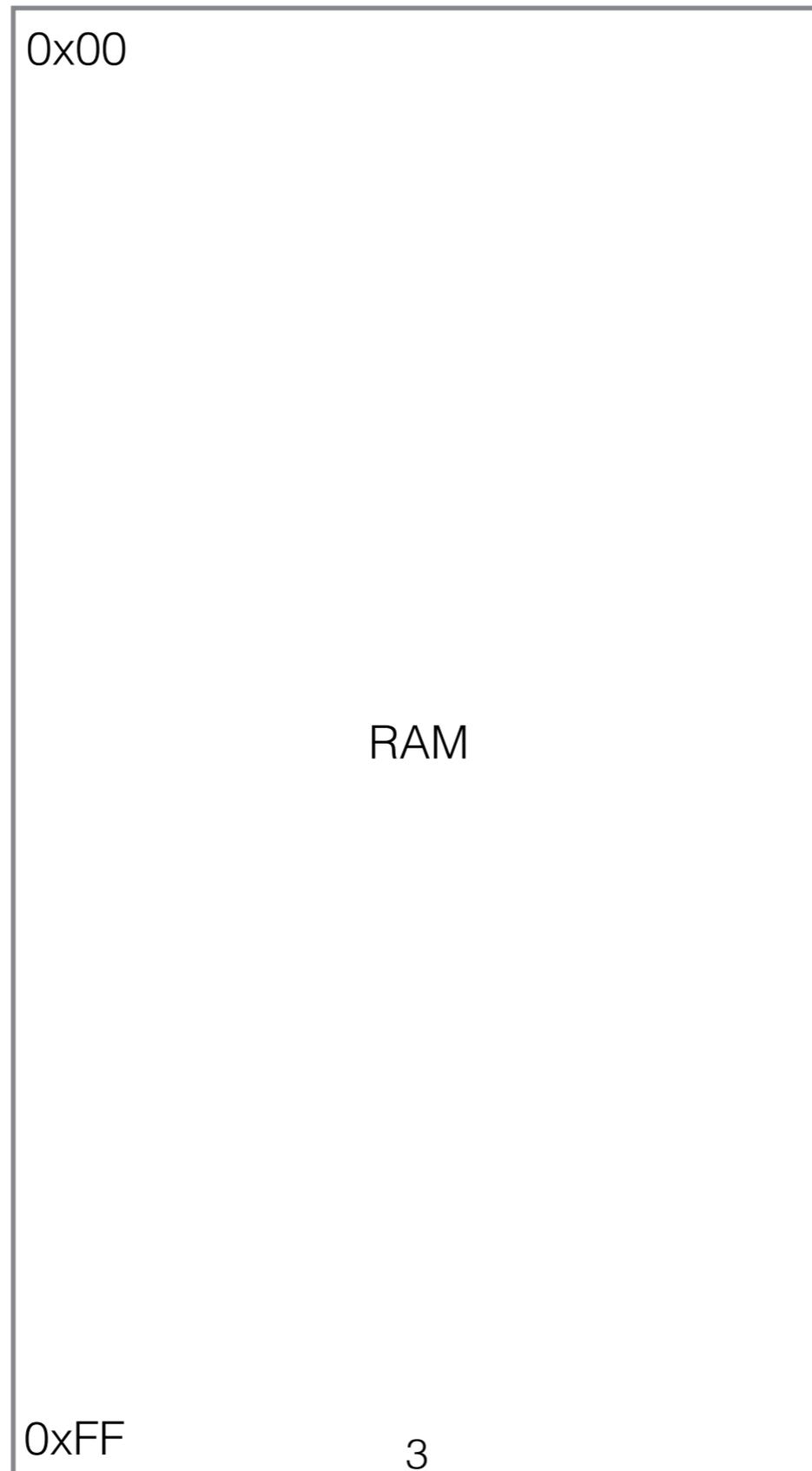
GIF-1001: Ordinateurs: Structure et Applications
Jean-François Lalonde

Rappel: 3 bus

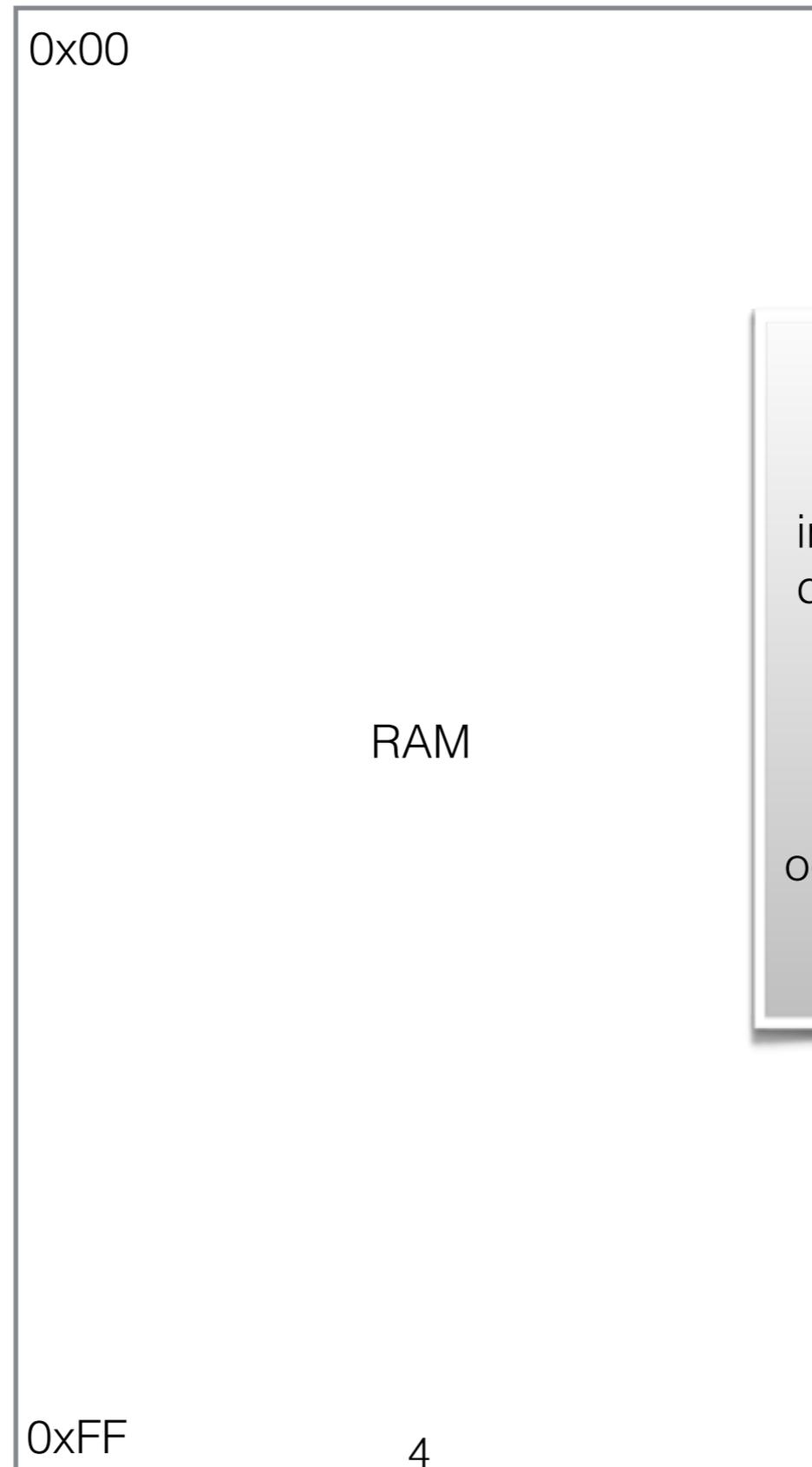


- Le **bus d'adresse** indique l'emplacement de la mémoire (ou des périphériques) visé par la transaction sur le bus.
- Le **bus de données** permet le transfert des données.
- Le **bus de contrôle** contrôle l'utilisation des bus de données et d'adresse.

Carte de la mémoire (*memory map*)



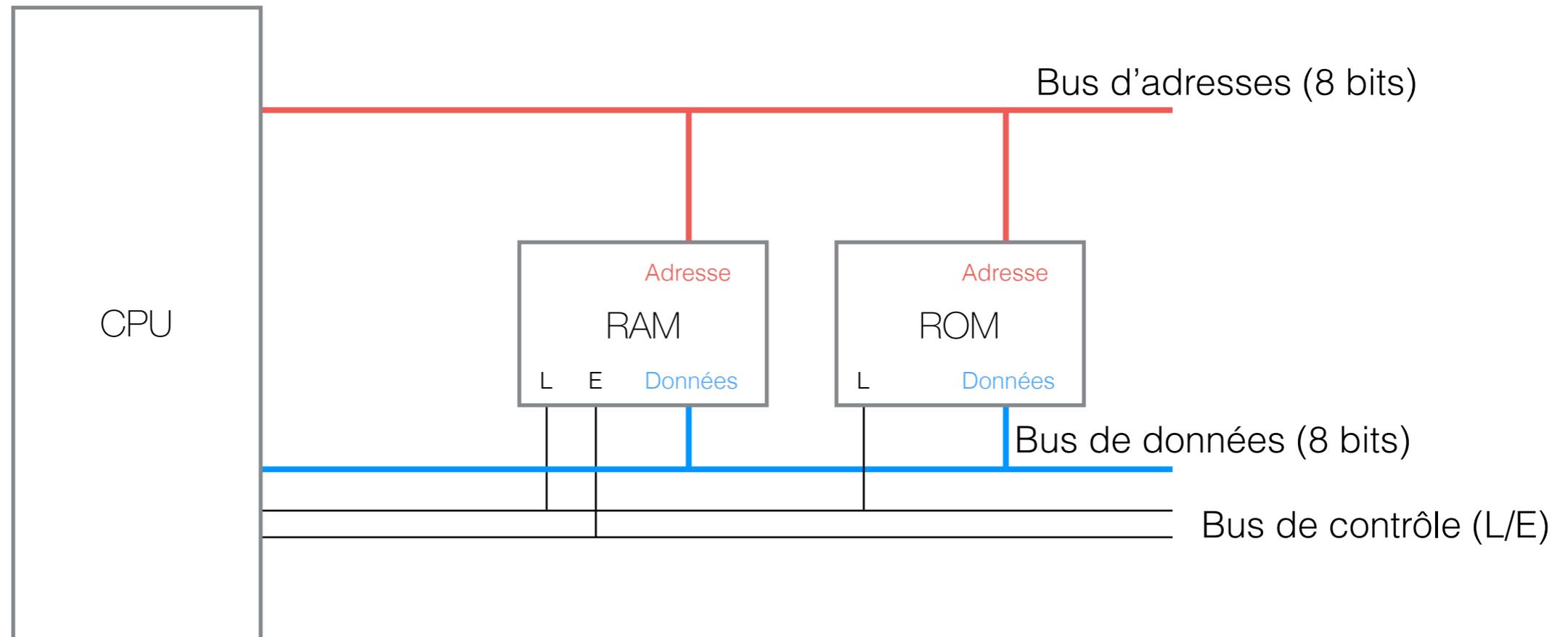
Carte de la mémoire (*memory map*)



La **carte de la mémoire** (*memory map* en anglais) indique l'adresse de début et de fin de toutes les composantes placées sur les bus.

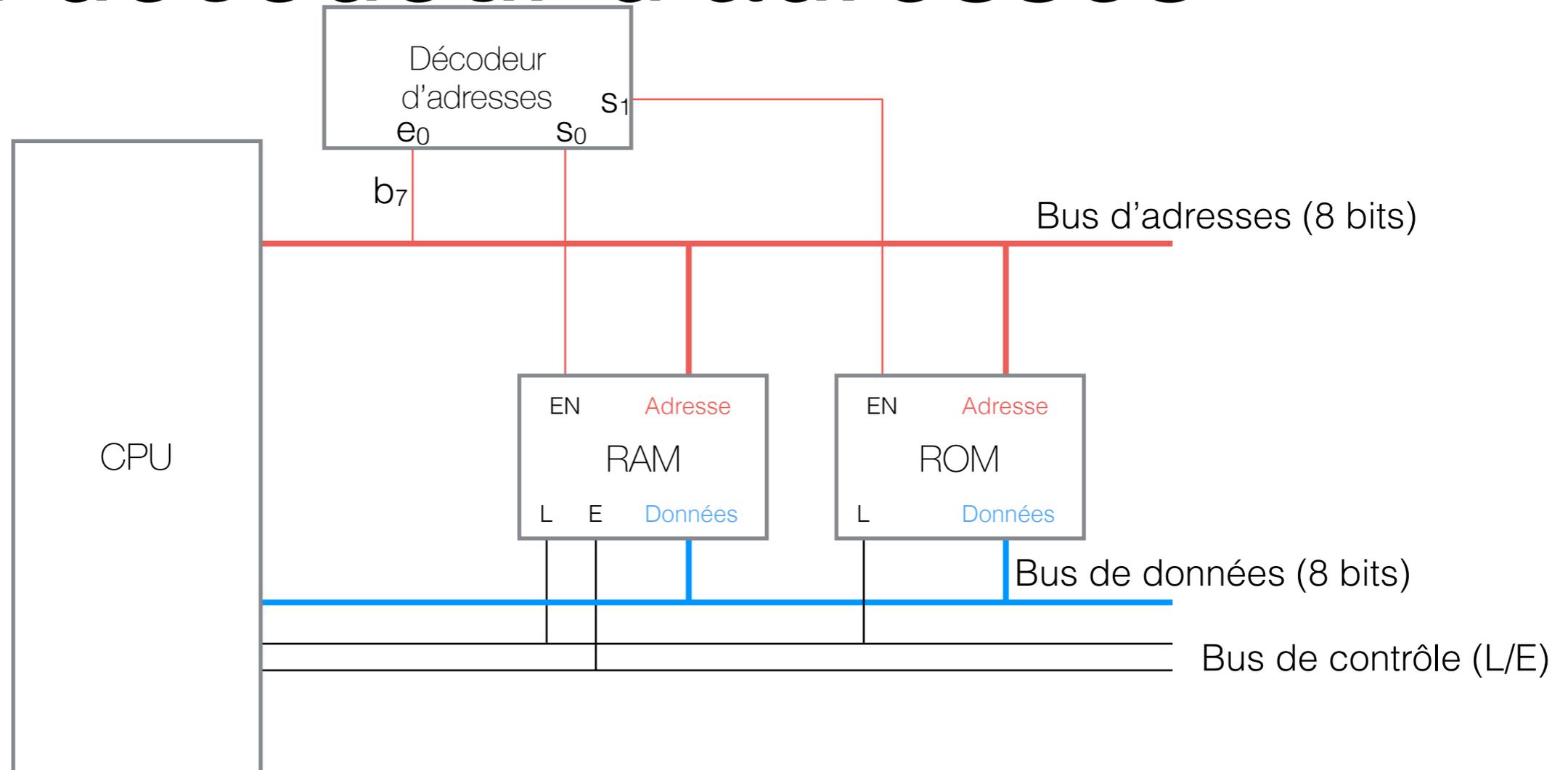
Dans cet exemple, il n'y a qu'une composante (la RAM), alors elle occupe toute la plage d'adresses du micro-processeur.

Bus: adressage



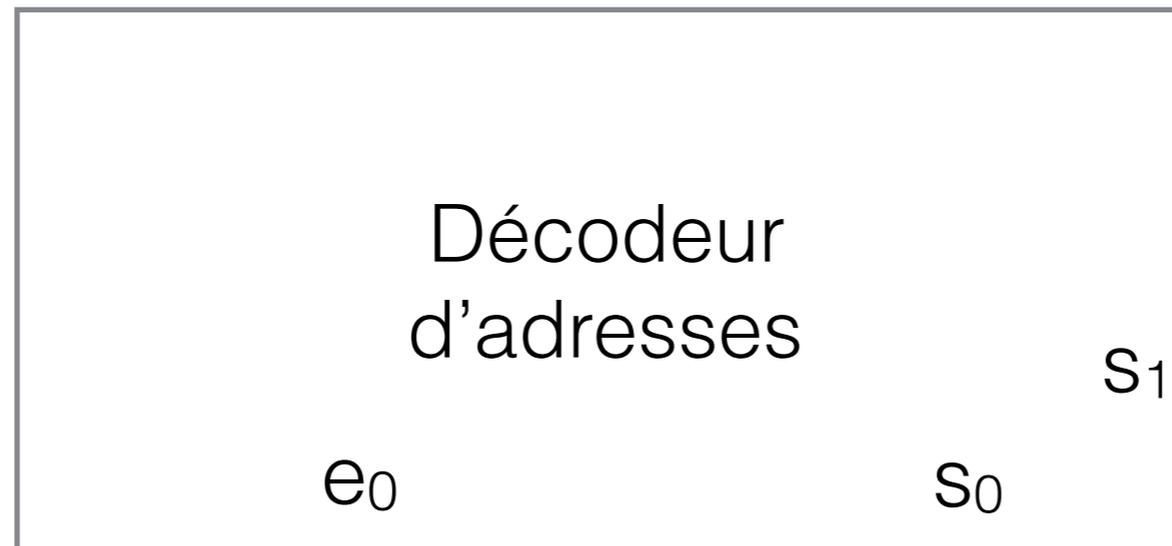
- Questions:
 - Comment faire pour sélectionner la bonne mémoire?

Bus: décodeur d'adresses



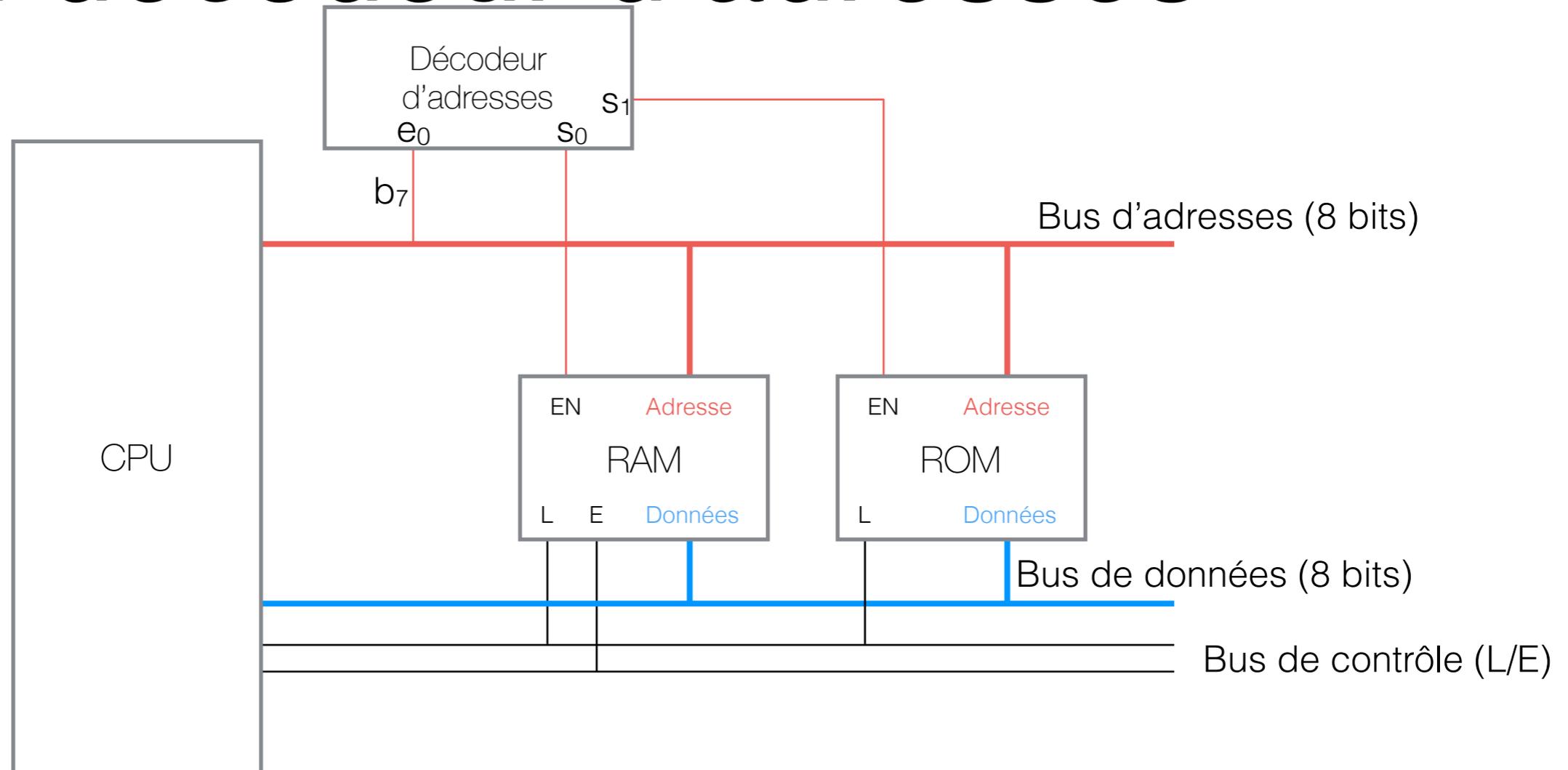
- Solution: « emprunter » un bit (b_7) et un décodeur d'adresses

Bus: décodeur d'adresses



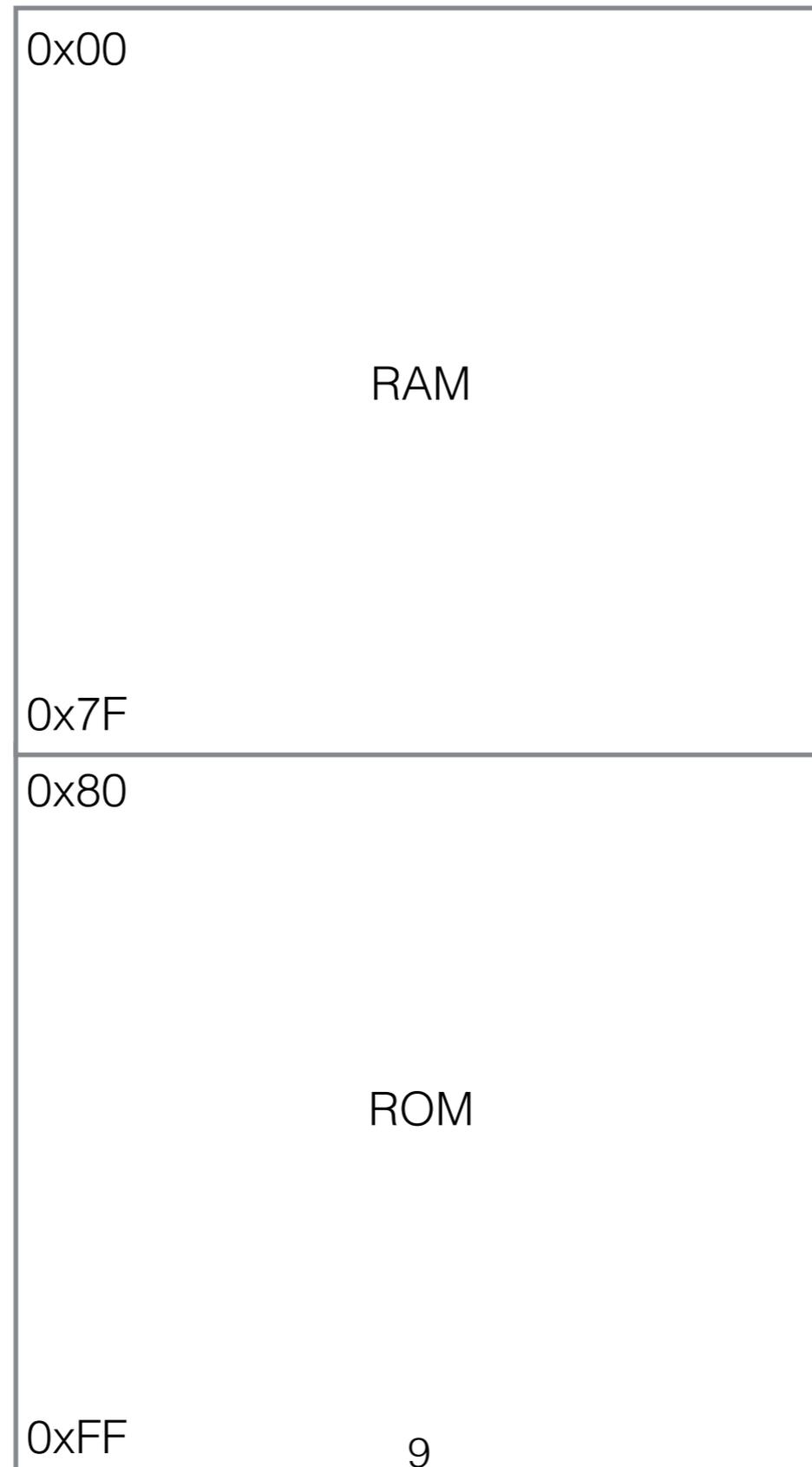
- Le décodeur d'adresses est un circuit logique qui sélectionne une sortie en fonction des entrées.
- Par exemple, le décodeur ci-haut possède une entrée « e_0 » et deux sorties « s_0 » et « s_1 ». La valeur des sorties est calculée comme suit:
 - Si $e_0 = 0$ alors $s_0 = 1$, $s_1 = 0$
 - Si $e_0 = 1$ alors $s_0 = 0$, $s_1 = 1$
- Ce genre de circuit est aussi connu sous le nom de démultiplexeur (demux)

Bus: décodeur d'adresses

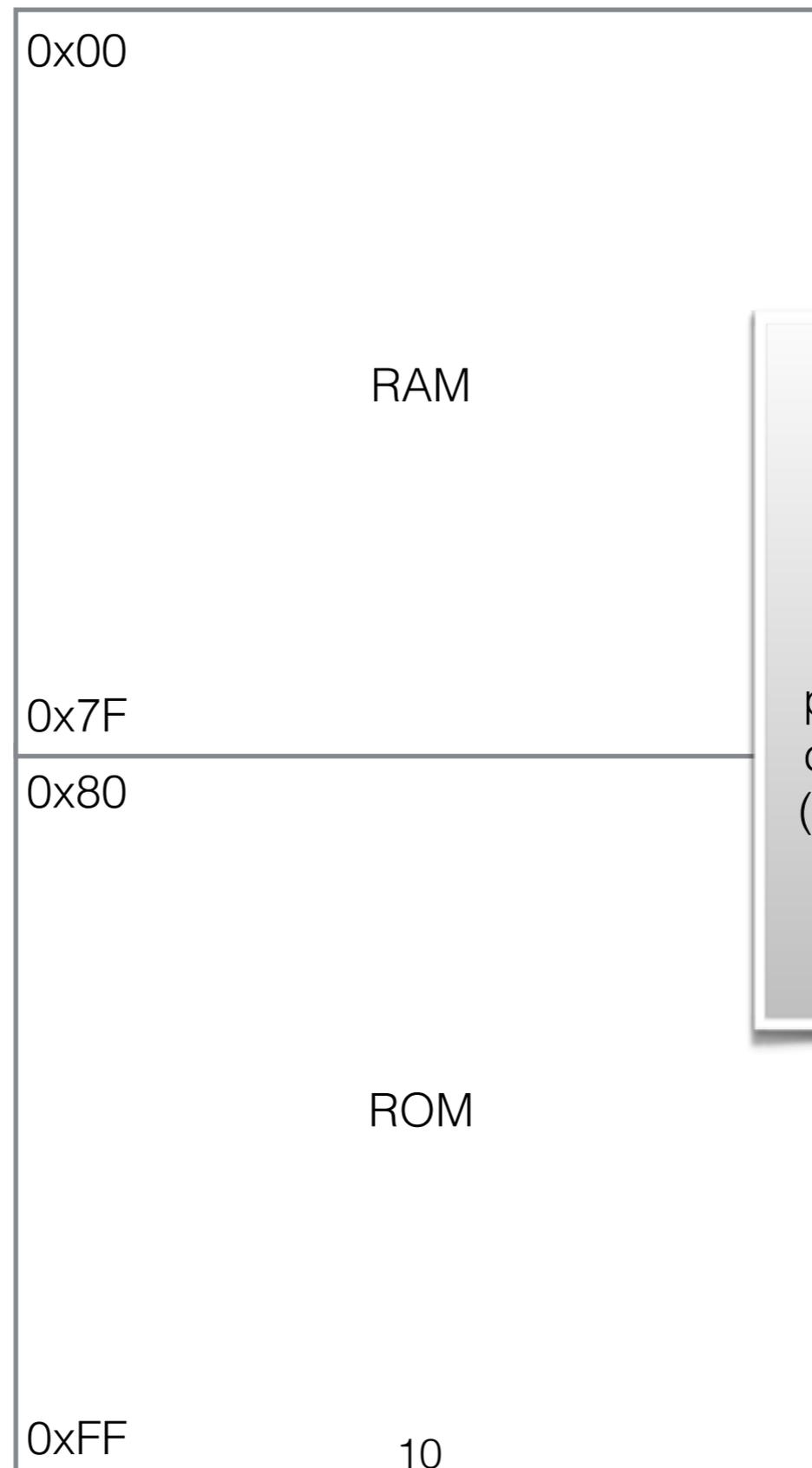


- Questions:
 - Quelle est la taille maximale de RAM et ROM (en octets)?
 - Aux yeux du CPU, quelle est l'adresse du premier emplacement mémoire en RAM? en ROM?
 - Quelle est la carte de la mémoire (*memory map*) de ce système?

Carte de la mémoire (*memory map*)



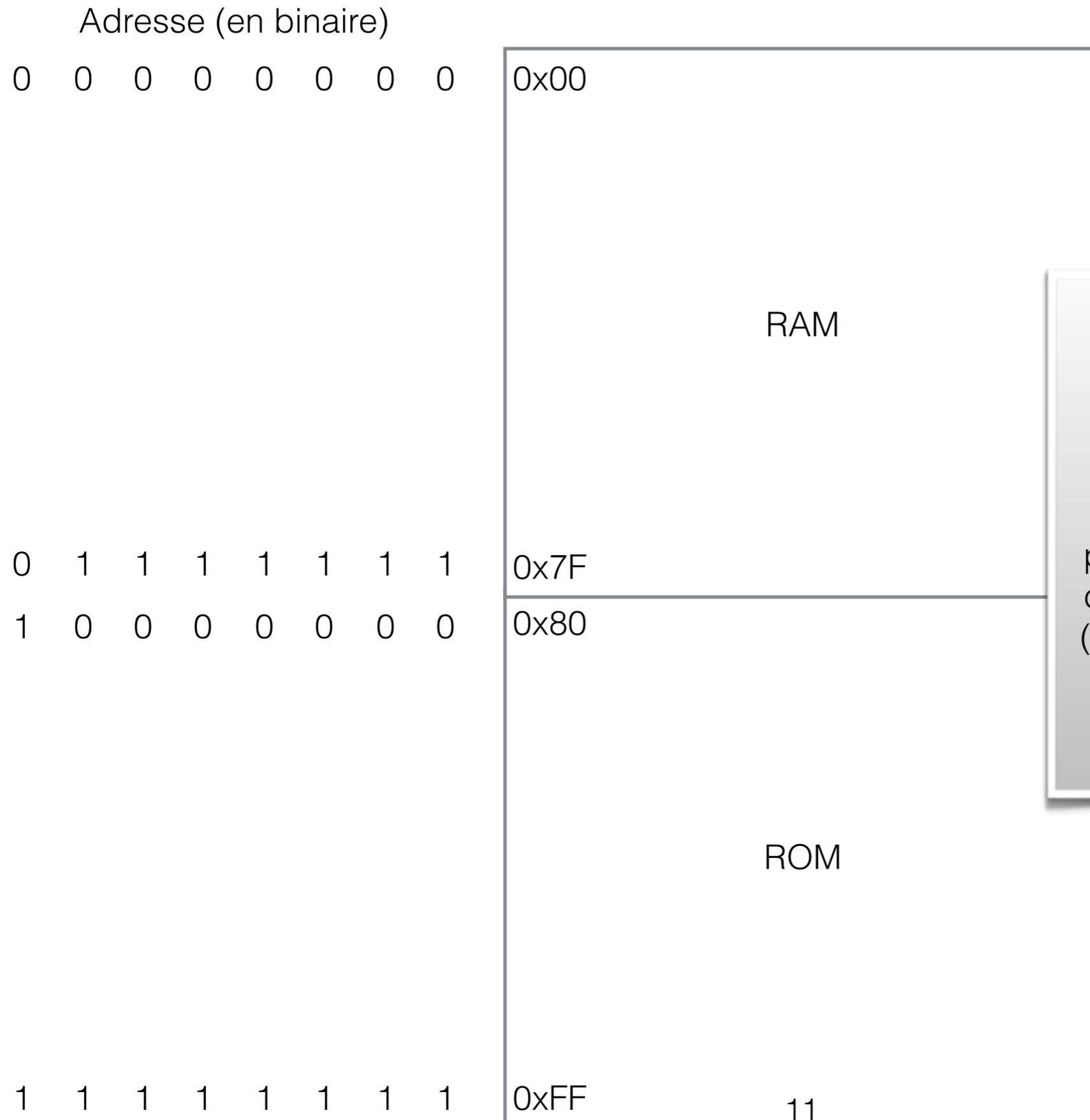
Carte de la mémoire (*memory map*)



Nous avons maintenant deux composants sur les bus (RAM et ROM).

La plage d'adresses du microprocesseur (0x00 à 0xFF) est donc divisée en deux: la première moitié (de 0x00 à 0x7F) pour la RAM, et la seconde (de 0x80 à 0xFF) pour la ROM.

Carte de la mémoire (*memory map*)



Nous avons maintenant deux composants sur les bus (RAM et ROM).

La plage d'adresses du microprocesseur (0x00 à 0xFF) est donc divisée en deux: la première moitié (de 0x00 à 0x7F) pour la RAM, et la seconde (de 0x80 à 0xFF) pour la ROM.

Carte de la mémoire (*memory map*)

Adresse (en binaire)



Le bit b₇ est utilisé par le décodeur d'adresse pour déterminer quelle composante sera activée!

RAM

ROM

Nous avons maintenant deux composantes sur les bus (RAM et ROM).

La plage d'adresses du microprocesseur (0x00 à 0xFF) est donc divisée en deux: la première moitié (de 0x00 à 0x7F) pour la RAM, et la seconde (de 0x80 à 0xFF) pour la ROM.

Activation (*enable*)

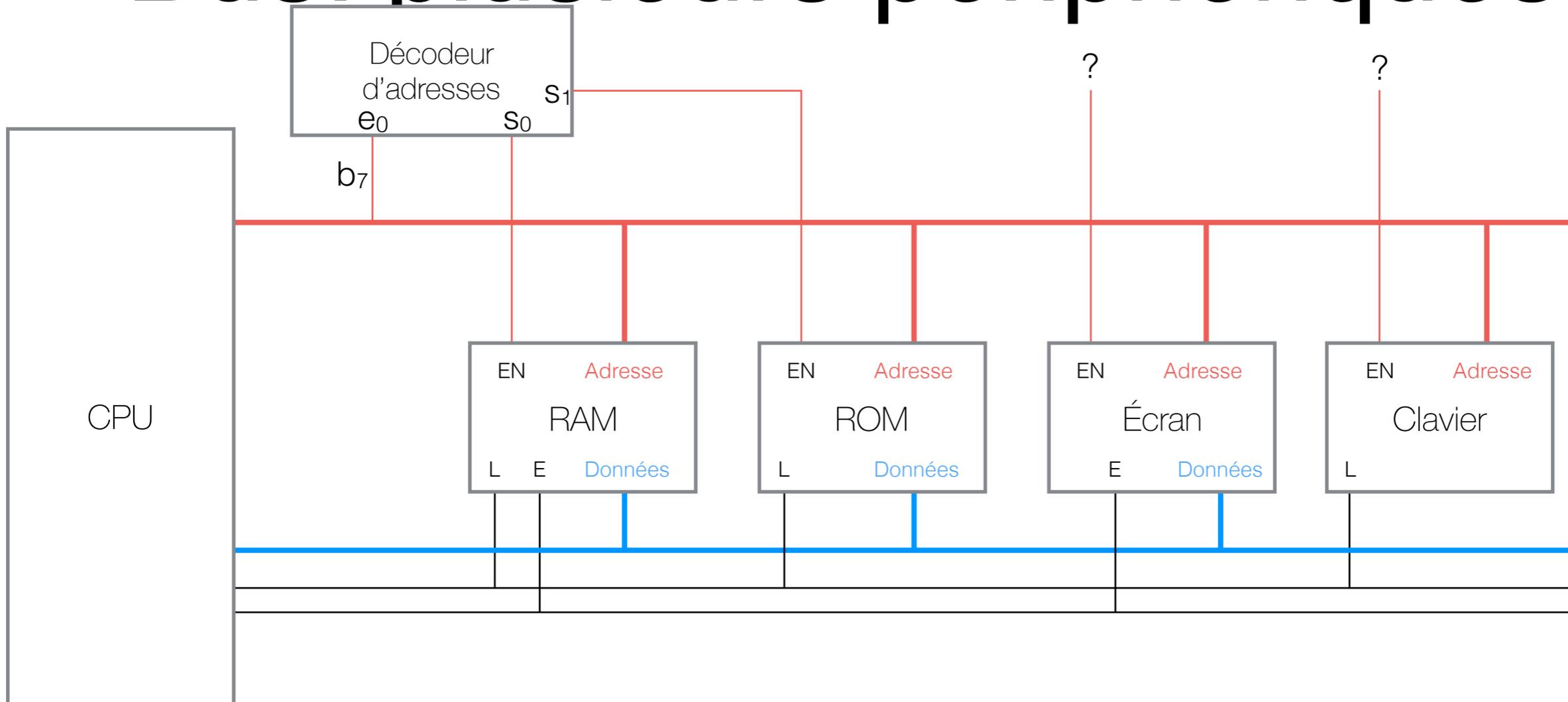
- Un bus relie le CPU à plusieurs composantes
 - Plusieurs composantes sont donc branchées sur le même circuit.
- Question: comment faire pour qu'une seule composante puisse accéder au bus à la fois?
- Réponse:
 - chaque bloc mémoire possède un signal d'activation (*enable*) qui indique si elle est sélectionnée pour lecture ou écriture sur le bus de données
 - sinon, la composante ne communique pas avec le bus de données (elle est en « haute impédance »)

Pour votre information seulement... pas besoin de bien comprendre le concept de « haute impédance » dans le cours.

Bus: adressage

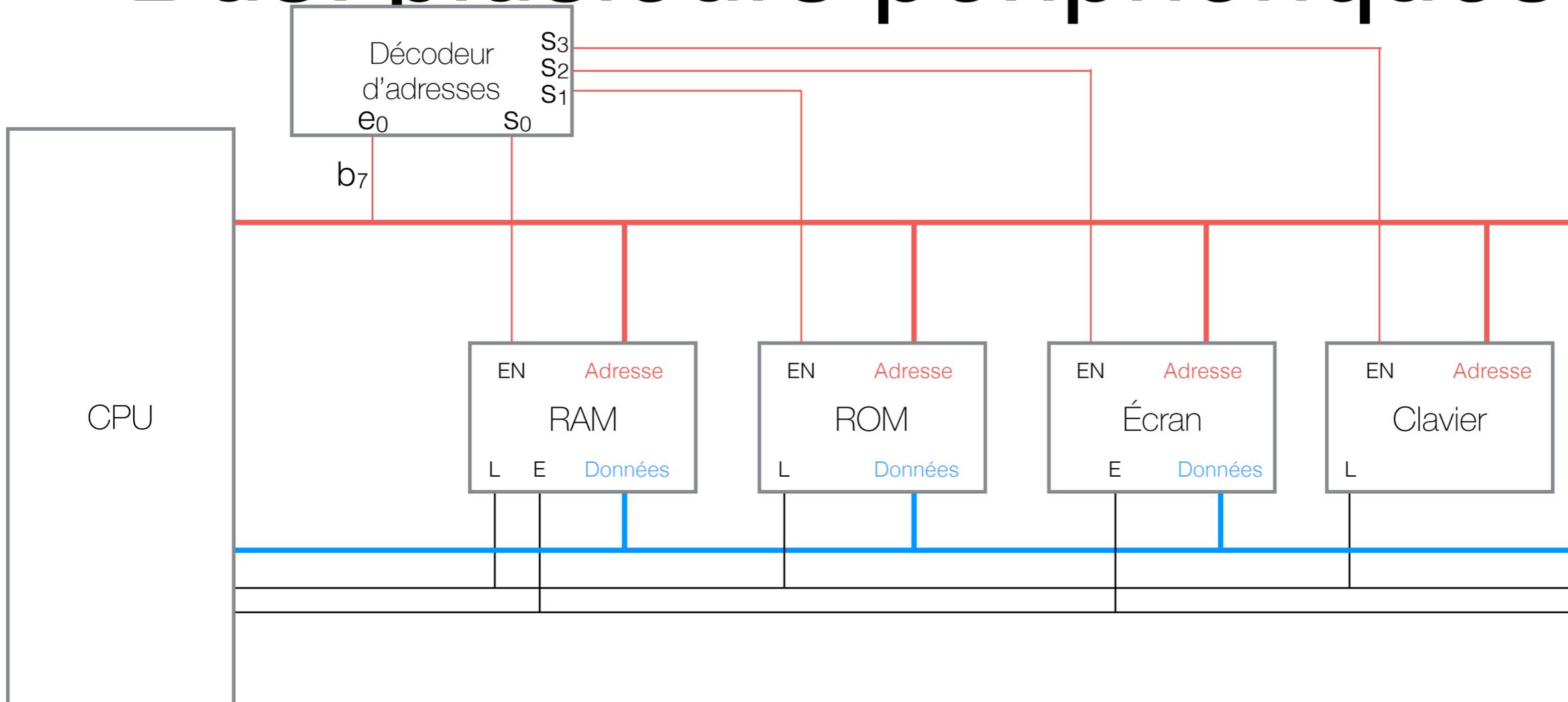
- Un bus relie le CPU à plusieurs composantes
- Question: comment déterminer quelle composante devrait être activée?
- Réponse:
 - c'est le décodeur d'adresse qui détermine quelle composante est activée (avec le *enable*) selon l'adresse spécifiée sur le bus d'adresse
 - les autres composantes ne communiquent pas avec le bus de données (sont en « haute impédance »)

Bus: plusieurs périphériques



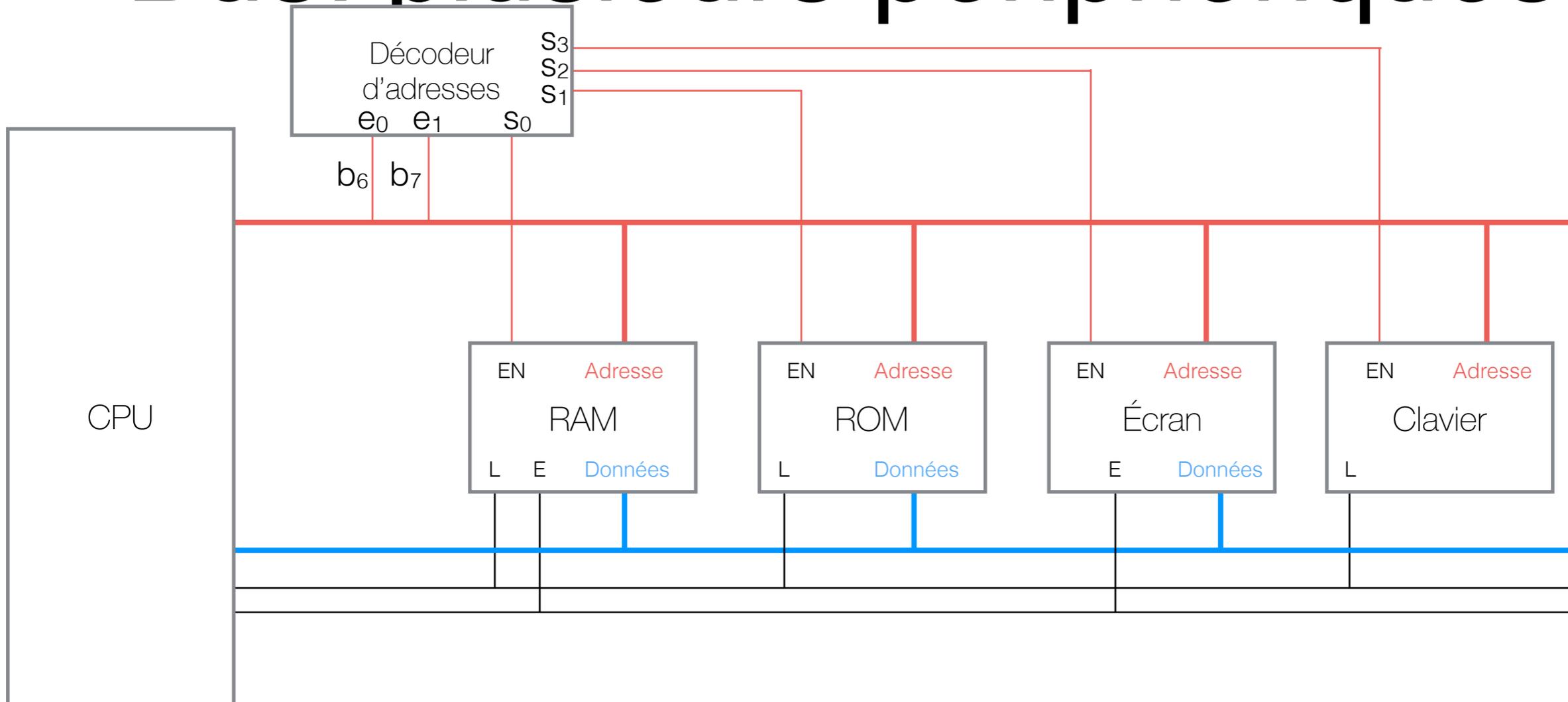
- Question:
 - Comment faire pour supporter plus que deux mémoires?

Bus: plusieurs périphériques



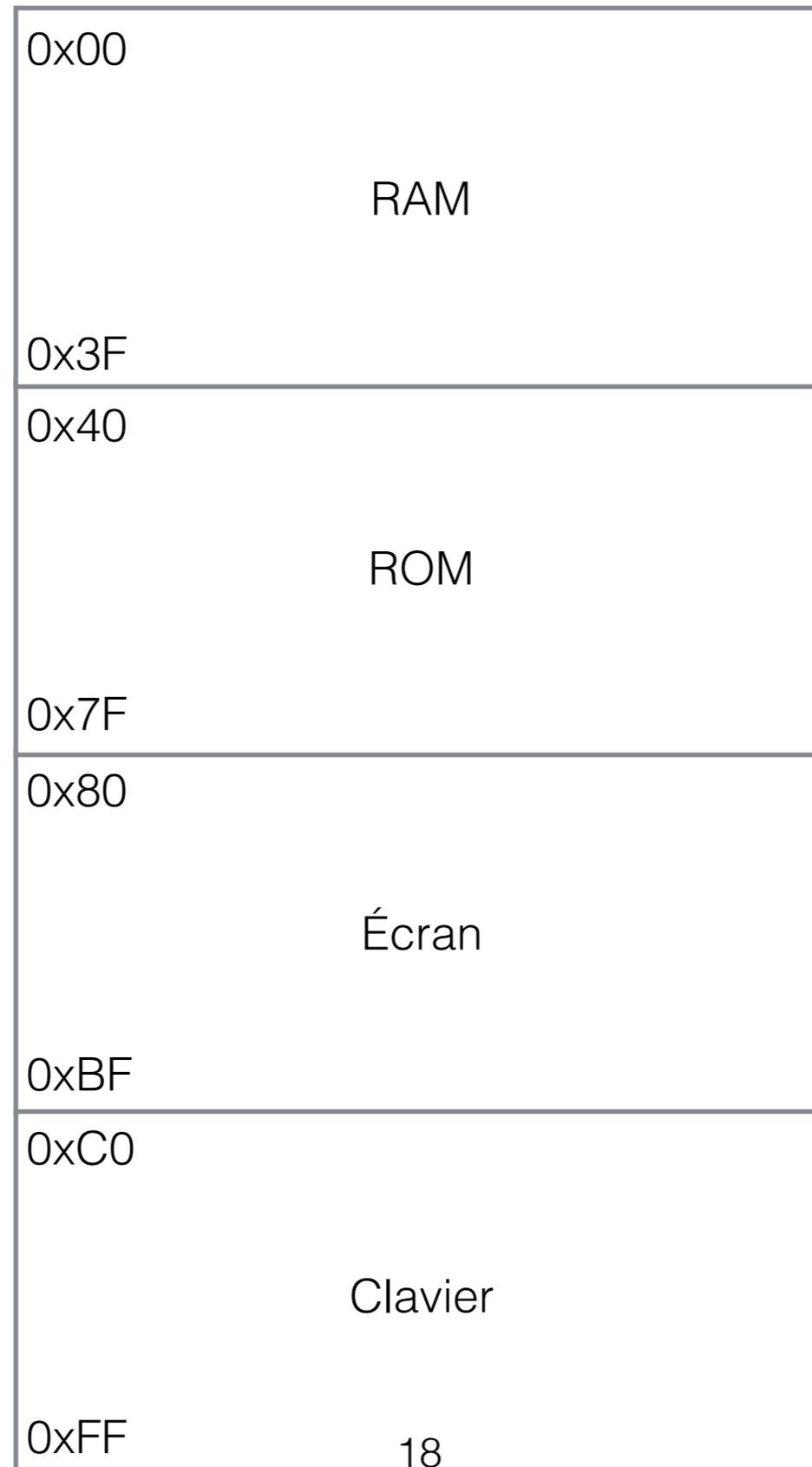
- Question:
 - une entrée, quatre sorties?

Bus: plusieurs périphériques



- Question:
 - quelle est la carte de la mémoire (*memory map*) de ce système?

Carte de la mémoire (*memory map*)



Carte de la mémoire (*memory map*)

Adresse (en binaire)

0	0	0	0	0	0	0	0	0x00	RAM
0	0	1	1	1	1	1	1	0x3F	
0	1	0	0	0	0	0	0	0x40	ROM
0	1	1	1	1	1	1	1	0x7F	
1	0	0	0	0	0	0	0	0x80	Écran
1	0	1	1	1	1	1	1	0xBF	
1	1	0	0	0	0	0	0	0xC0	Clavier
1	1	1	1	1	1	1	1	0xFF	

Cette fois, les bits b_7 et b_6 sont utilisés par le décodeur d'adresse pour déterminer quelle composante sera activée!

Entrées-sorties (périphériques)

- Les I/Os (Input-Output, entrées-sorties) servent d'interface avec l'utilisateur, les périphériques et d'autres ordinateurs.
- Des lignes de contrôle existent pour gérer certains I/Os spécifiques.
- Les I/Os ne sont pas que passifs
 - Ils peuvent générer des interruptions pour signaler des événements au microprocesseur (exemple: touche de clavier enfoncée)

Entrées-sorties: adressage

- Deux façons principales pour déterminer les adresses des I/Os
- Memory-Mapped I/O (MMIO)
 - les I/Os sont gérés exactement comme la mémoire: pour accéder à un I/O, on lit ou écrit une adresse du système
 - nous explorerons cette organisation dans le TP1
- Port-Mapped I/O (PMIO)
 - les I/Os ont leurs propres adresses, séparées des adresses systèmes
 - on emploie des instructions spécifiques aux I/Os pour y accéder
 - ex: x86