Bus et adressage

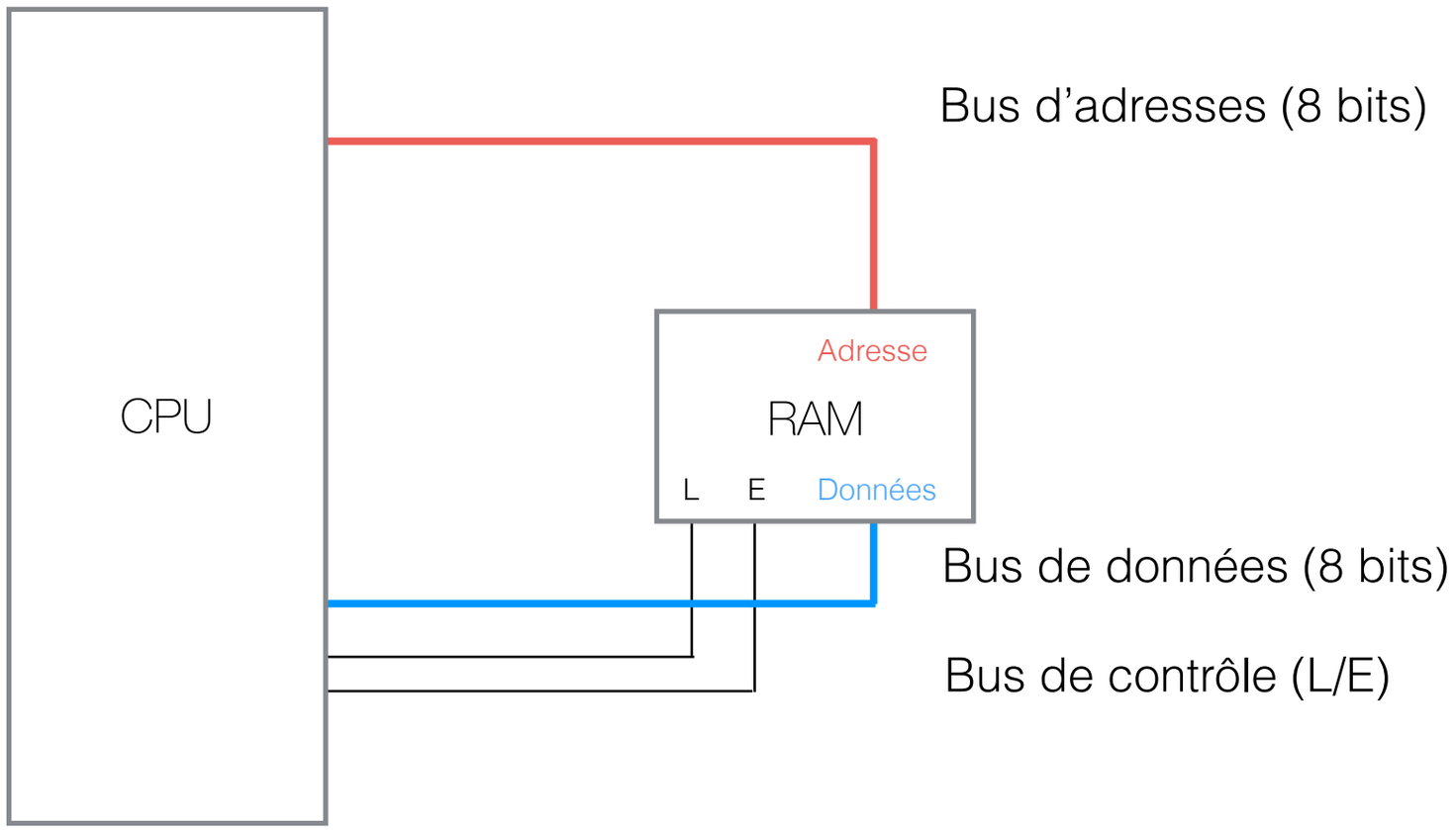
GIF-1001: Ordinateurs: Structure et Applications

Jean-François Lalonde, Hiver 2019

Aujourd’hui

- Mécanismes de fonctionnement:
  - bus et mémoires
  - adressage

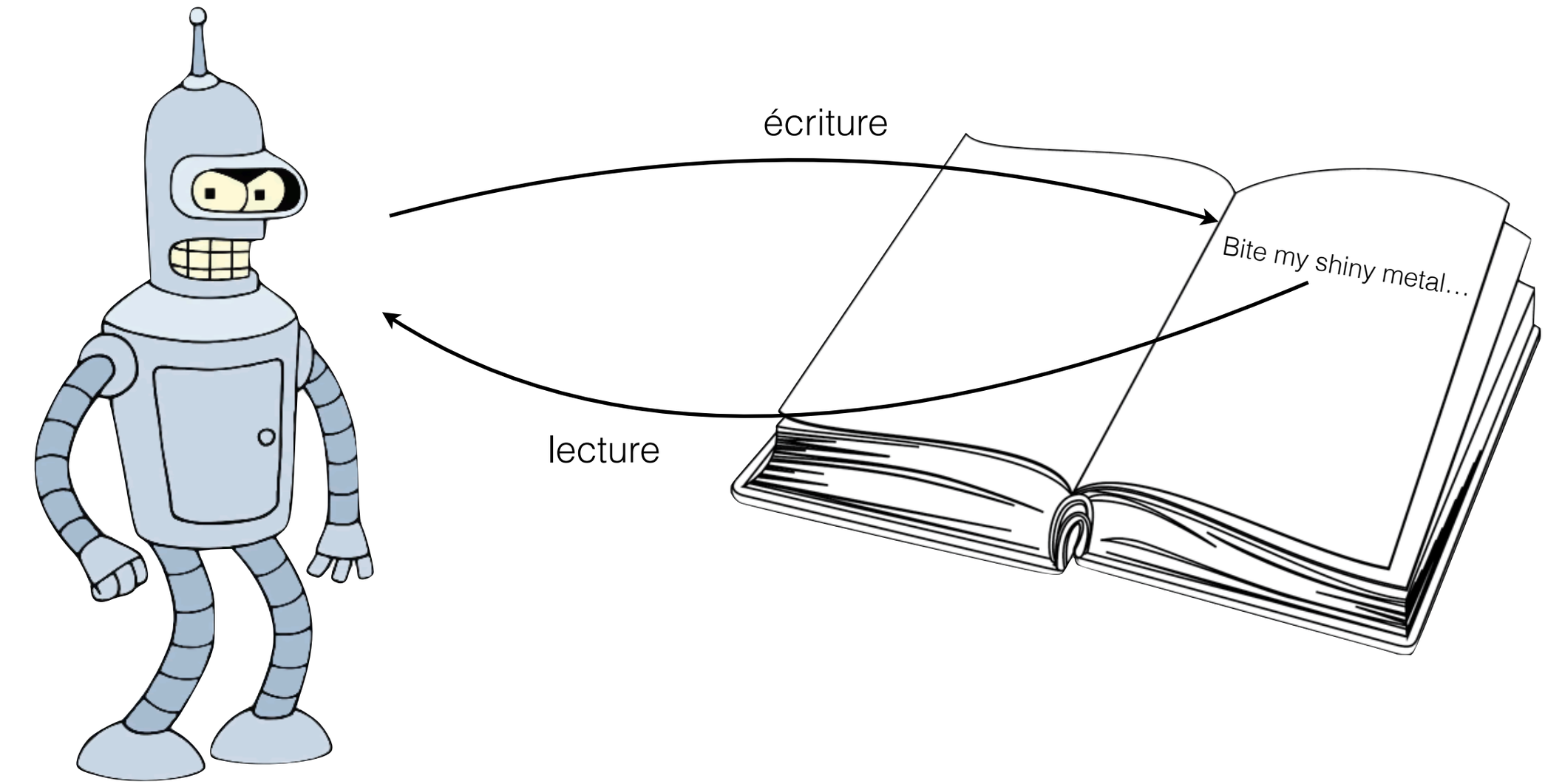
Rappel: CPU, mémoire, bus



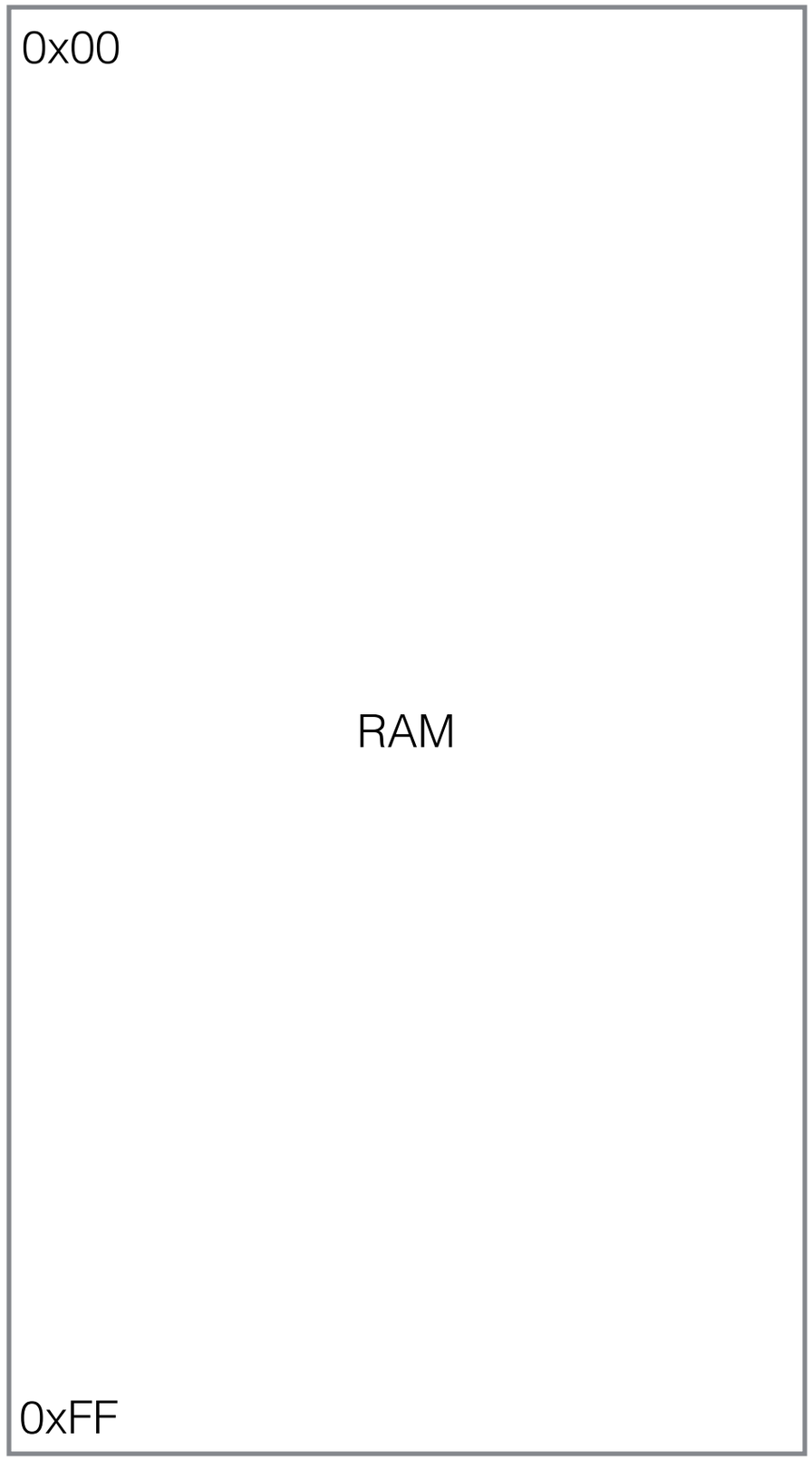
- Questions de rappel:
  - Comment lire des données stockées en mémoire RAM?
  - Comment écrire des données en mémoire RAM?
  - Combien d'adresses la mémoire a-t-elle?
  - Quelle est la taille des mots dans la RAM?

Lecture vs écriture

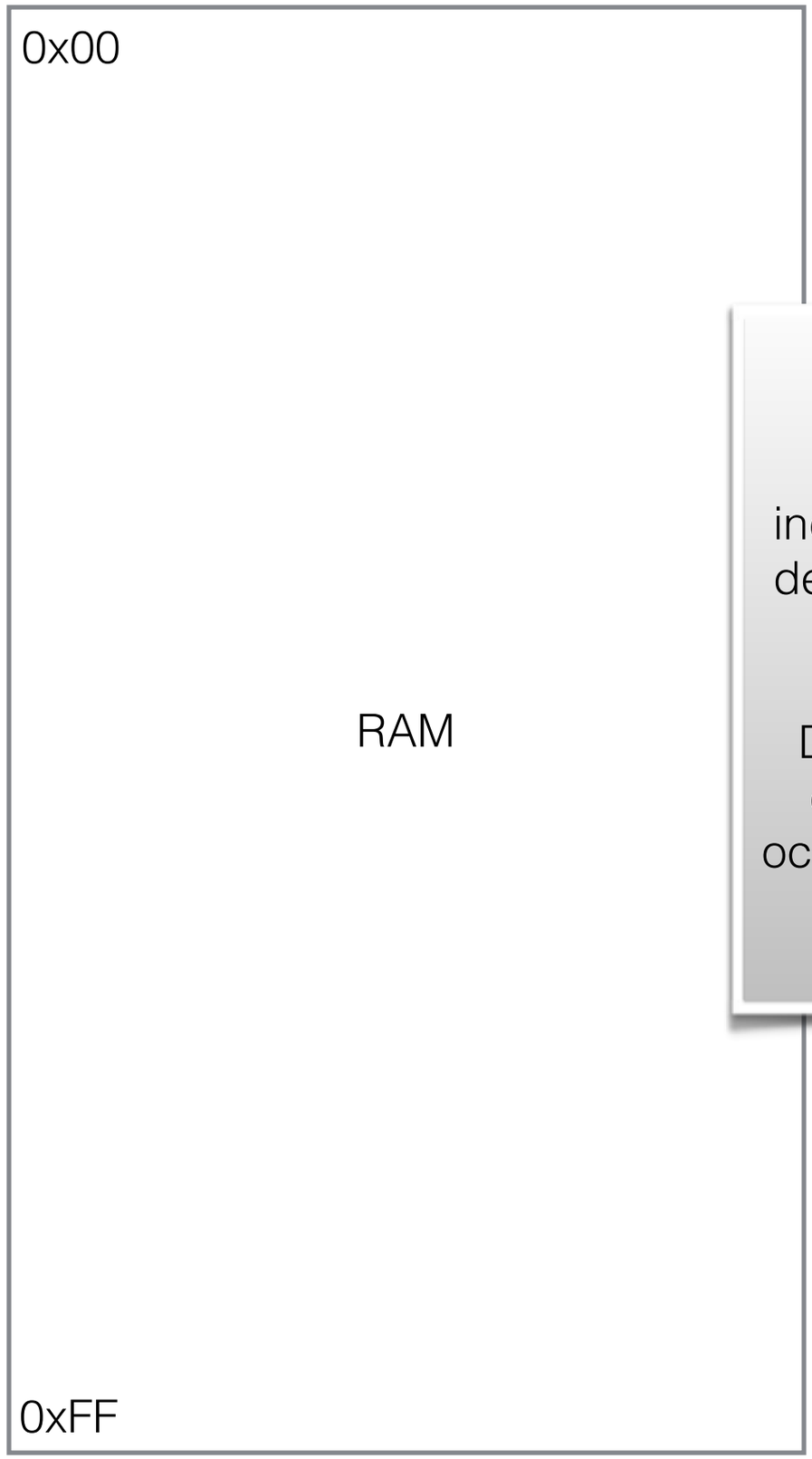
Mettez-vous dans la peau du microprocesseur!



Carte de la mémoire (*memory map*)



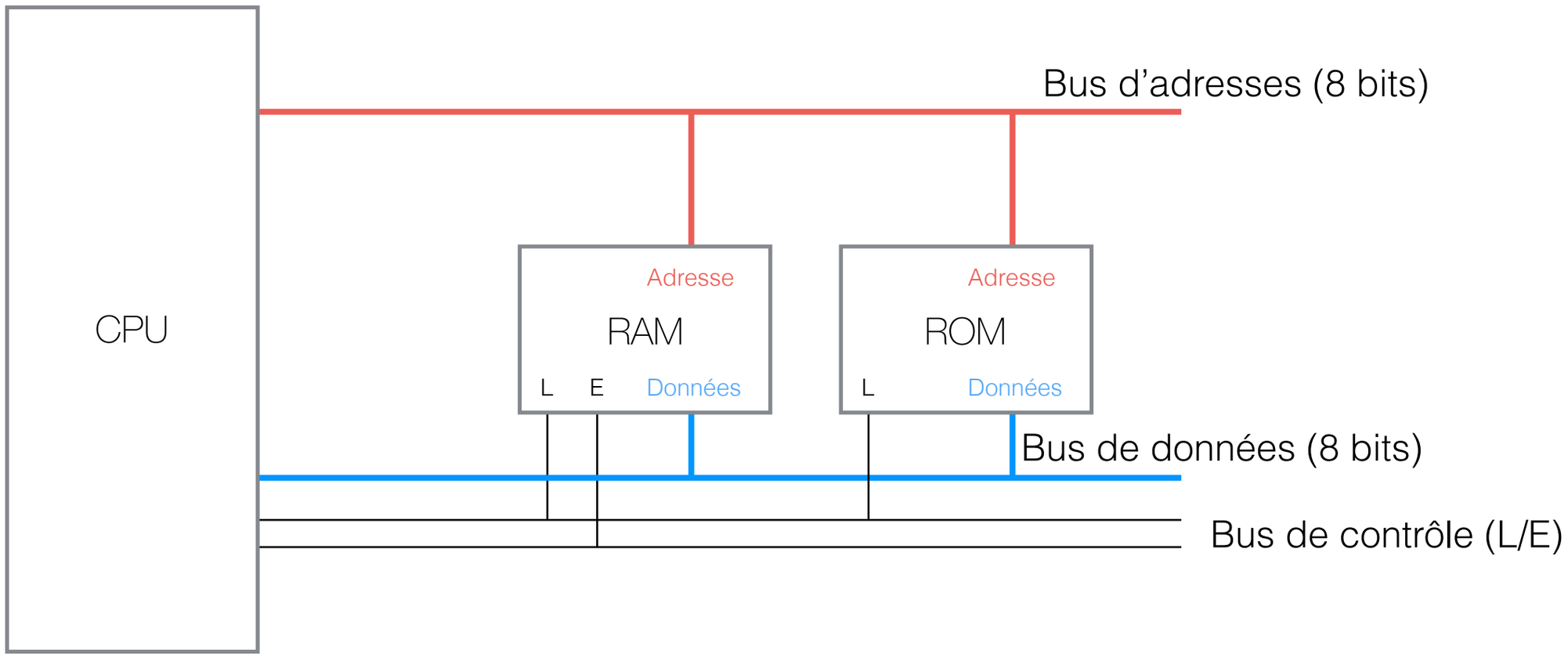
Carte de la mémoire (*memory map*)



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.

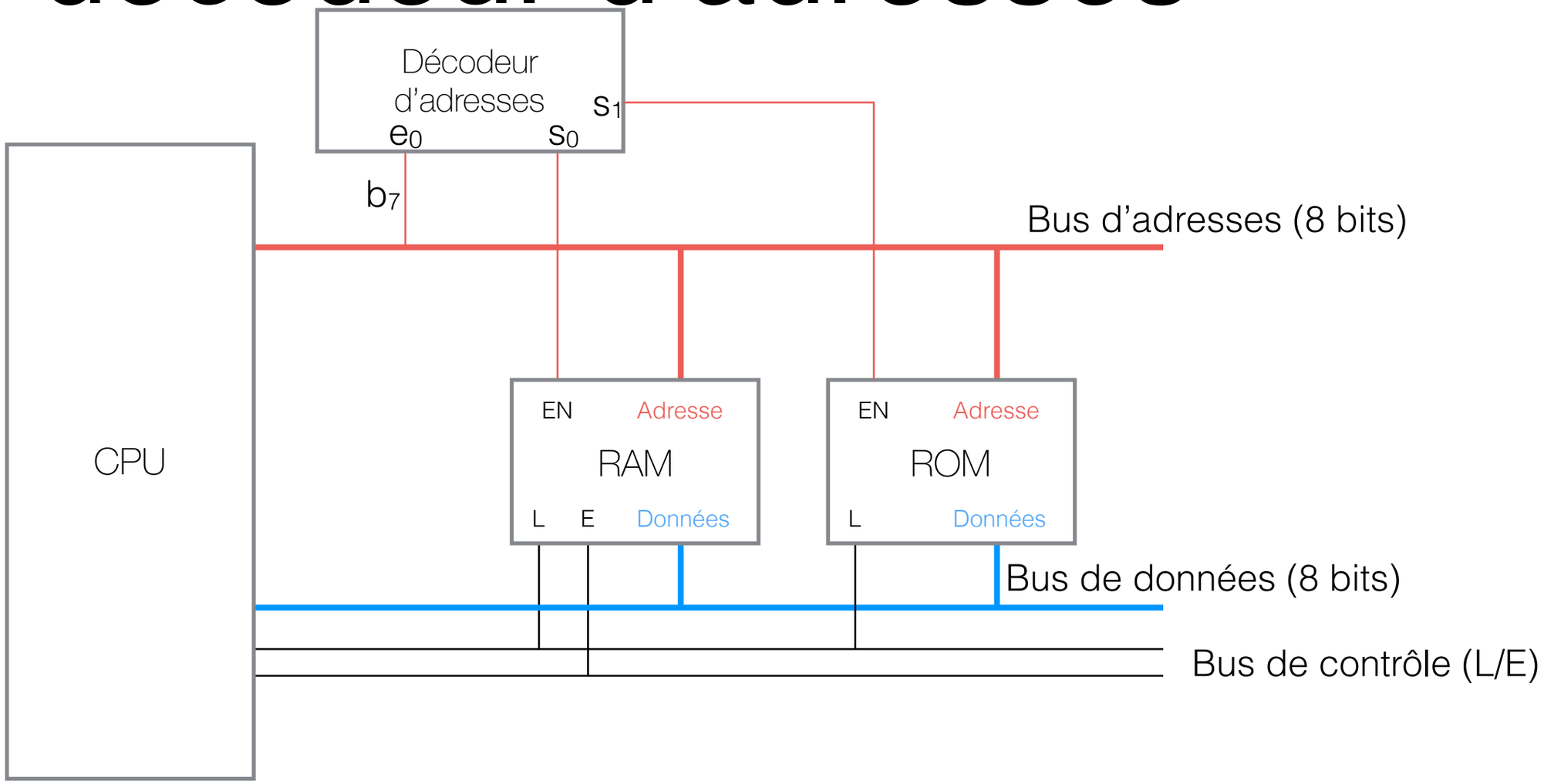
Dans cet exemple, il n'y a qu'une composante (la RAM), alors elle occupe toute la plage d'adresses du micro-processeur.

Bus: adressage



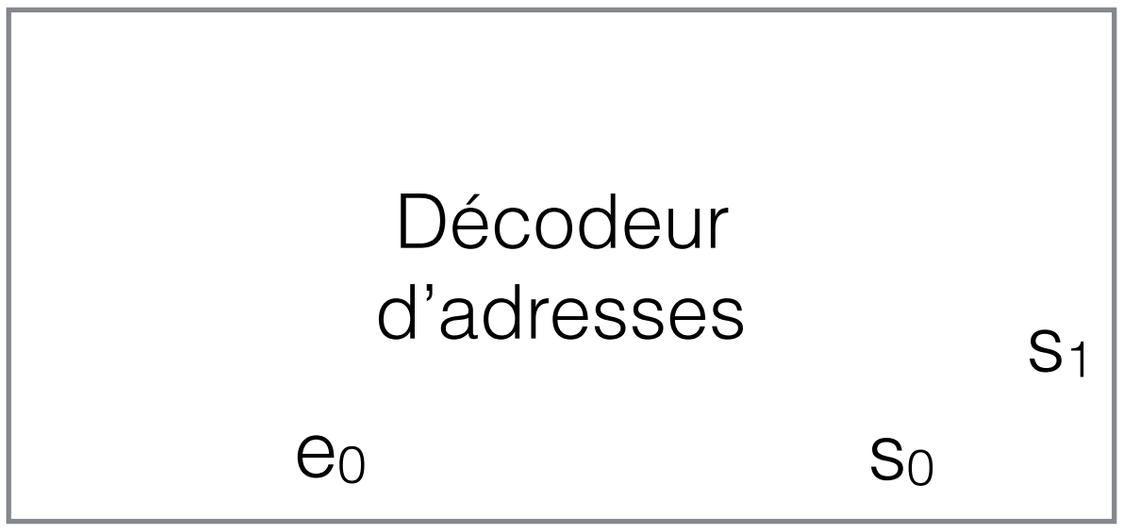
- Questions:
  - Comment faire pour sélectionner la bonne mémoire?

Bus: décodeur d'adresses



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

Bus: décodeur d'adresses

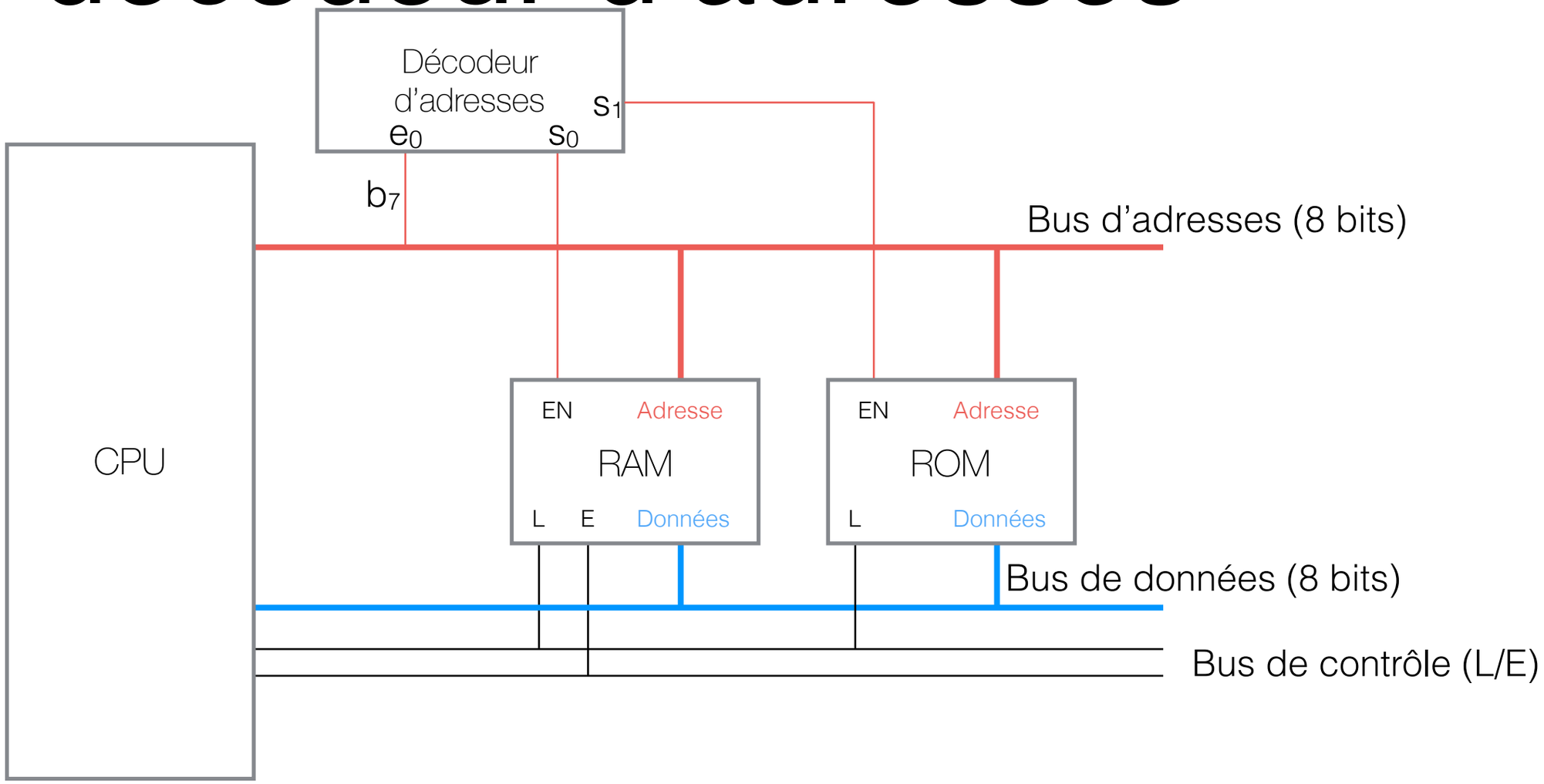


- Le décodeur d'adresses est un circuit logique qui sélectionne une sortie en fonction des entrées.

- 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 $s_0 = 0$, $s_1 = 1$

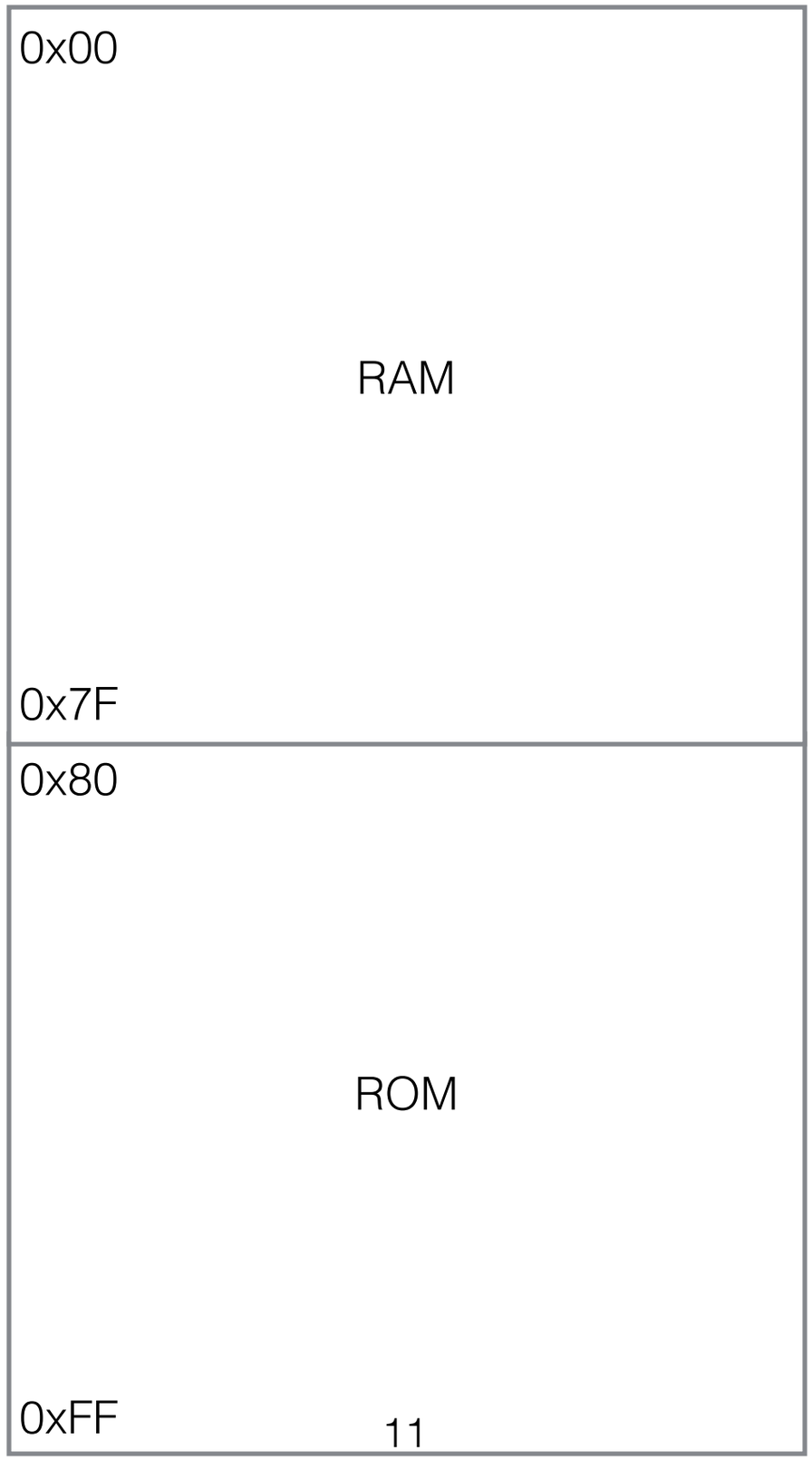
- Ce genre de circuit est aussi connu sous le nom de démultiplexeur (demux)

Bus: décodeur d'adresses

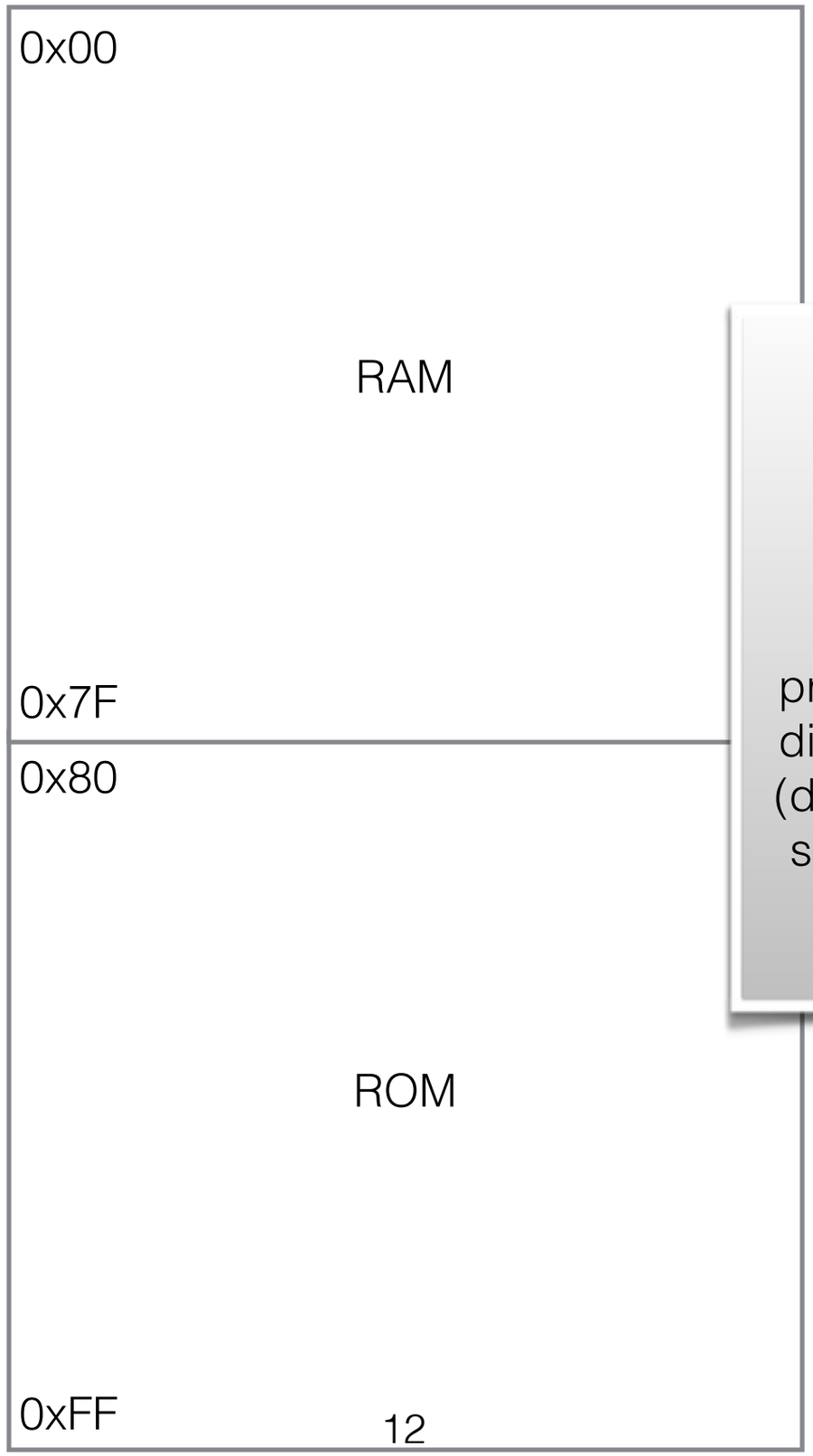


- Questions:
  - Quelle est la taille maximale de RAM et ROM (en octets)?
  - Aux yeux du CPU, quelle est l'adresse du premier emplacement mémoire en RAM? en ROM?
  - Quelle est la carte de la mémoire (*memory map*) de ce système?

Carte de la mémoire (*memory map*)



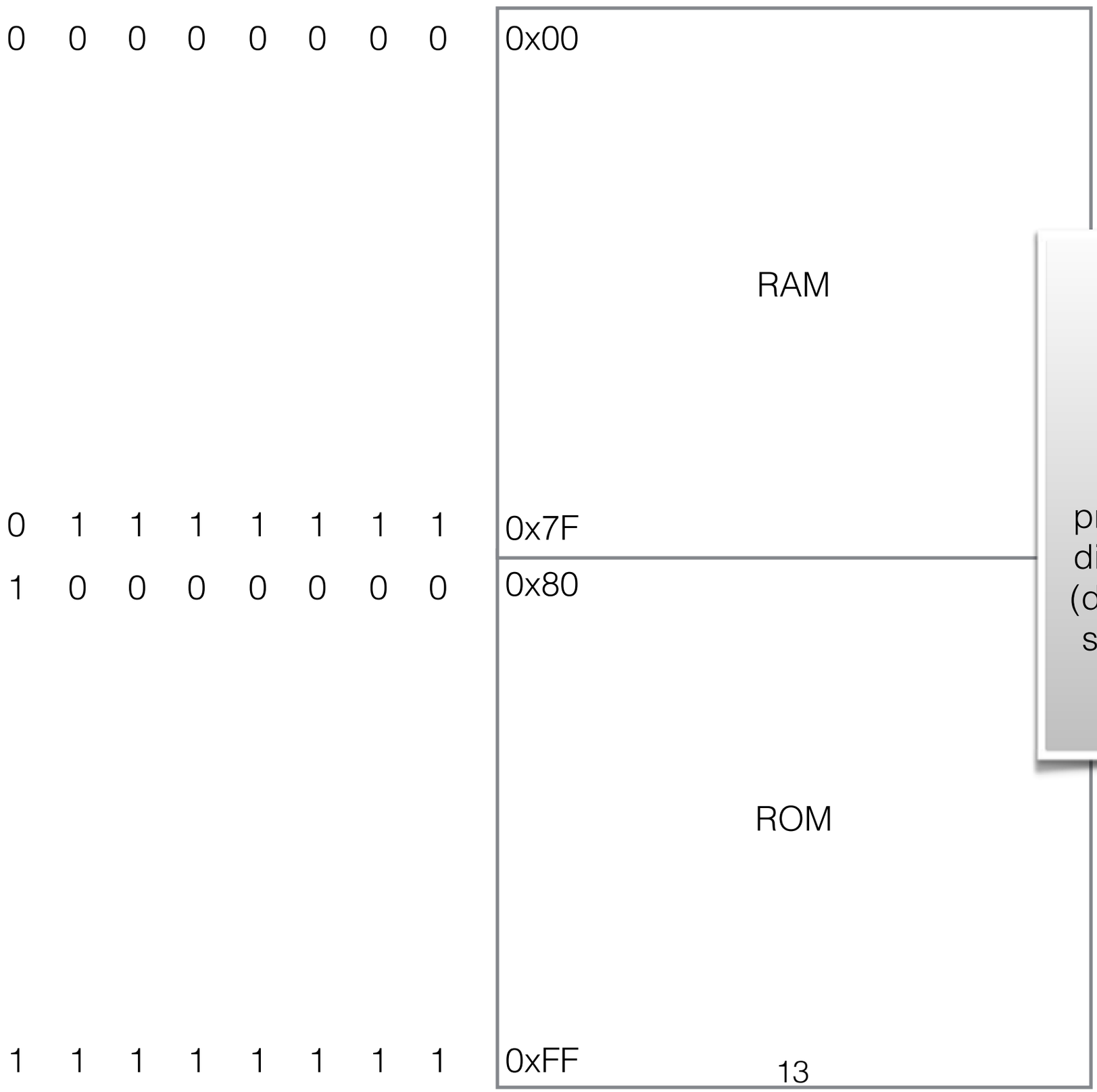
Carte de la mémoire (*memory map*)



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

La plage d'adresses du micro-processeur (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.

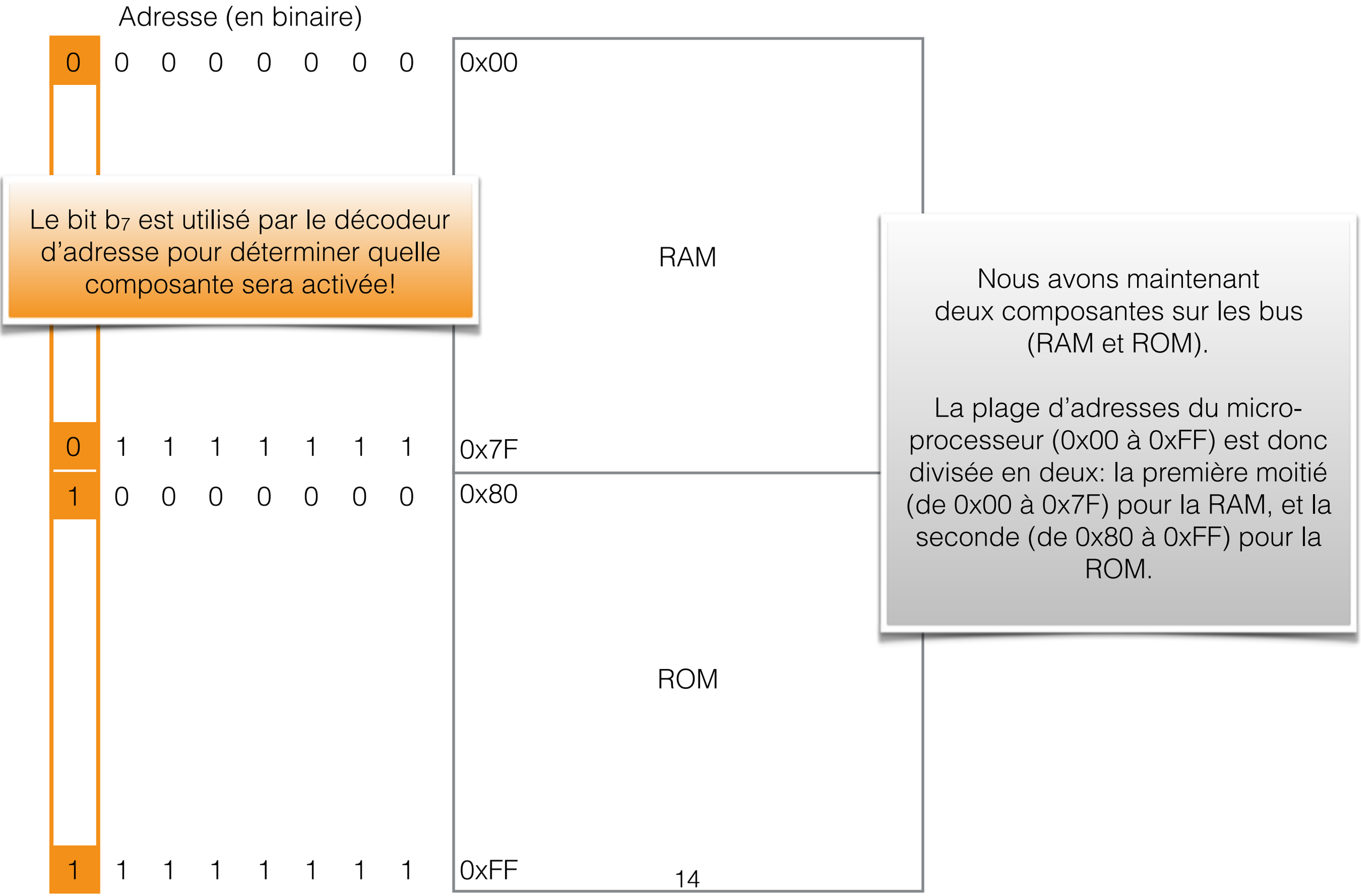
Carte de la mémoire (*memory map*)



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

La plage d'adresses du micro-processeur (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*)



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?

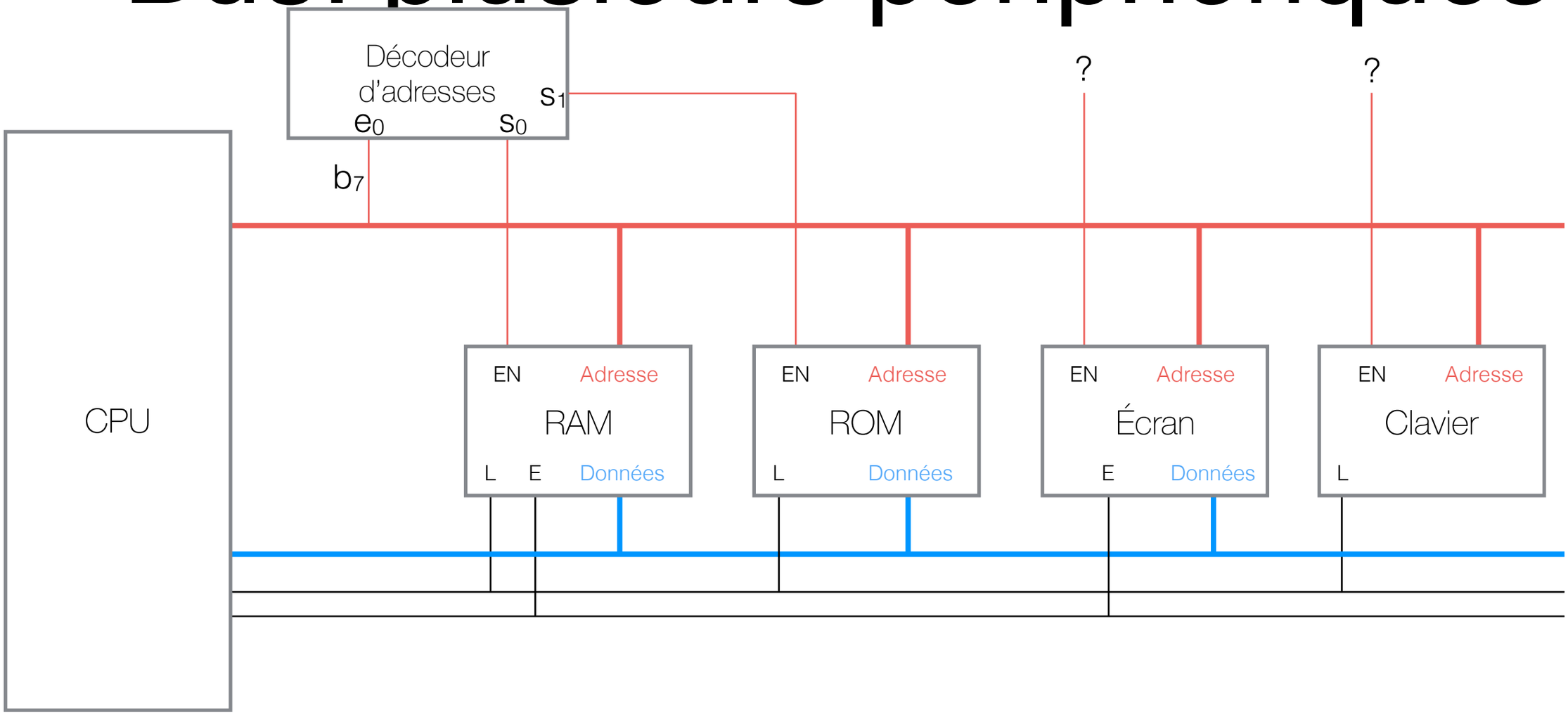
- Réponse:
  - chaque bloc mémoire possède un signal d'activation (*enable*) qui indique si elle est sélectionnée pour lecture ou écriture sur le bus de données
  - sinon, la composante ne communique pas avec le bus de données (elle est en « haute impédance »)

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

Bus: adressage

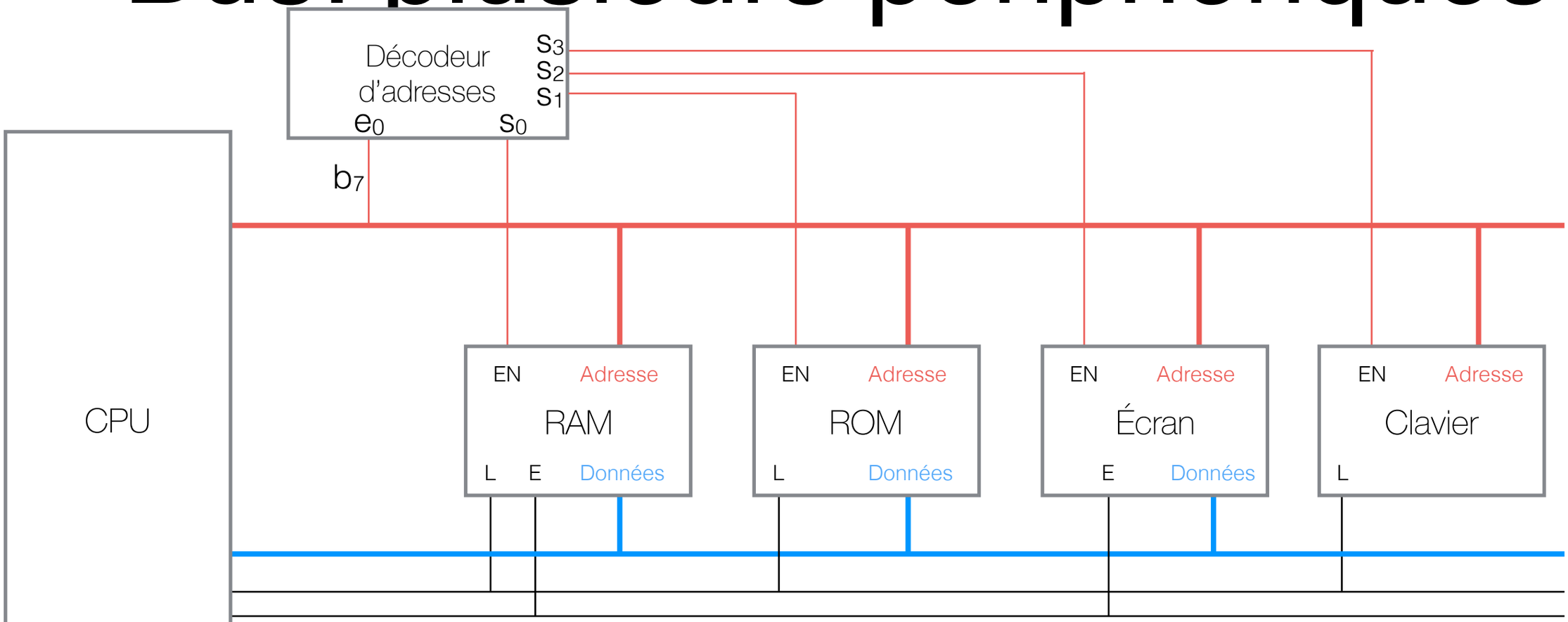
- Un bus relie le CPU à plusieurs composantes
- Question: comment déterminer quelle composante devrait être activée?
- Réponse:
  - c'est le décodeur d’adresse qui détermine quelle composante est activée (avec le *enable*) selon l’adresse spécifiée sur le bus d’adresse
  - les autres composantes ne communiquent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

Bus: plusieurs périphériques



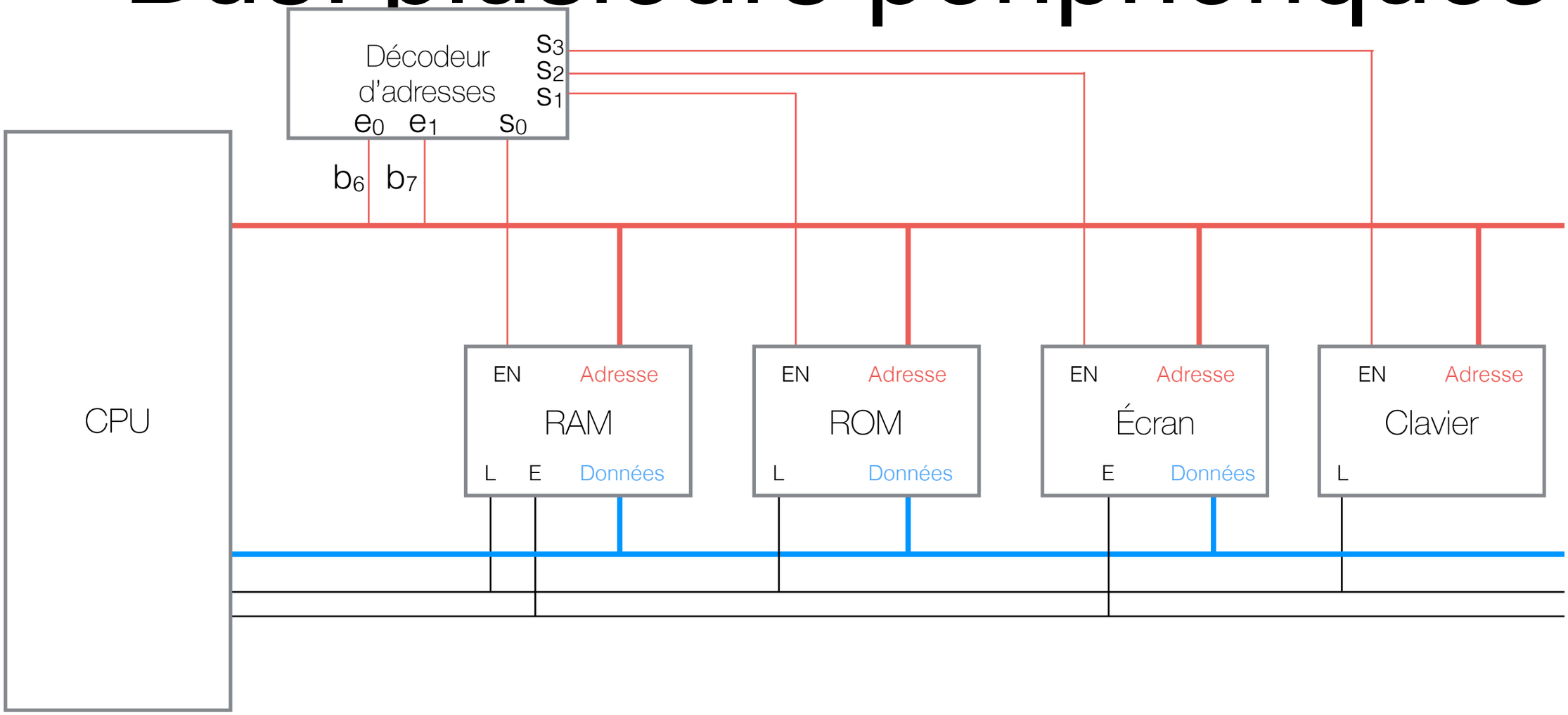
- Question:
  - Comment faire pour supporter plus que deux mémoires?

Bus: plusieurs périphériques



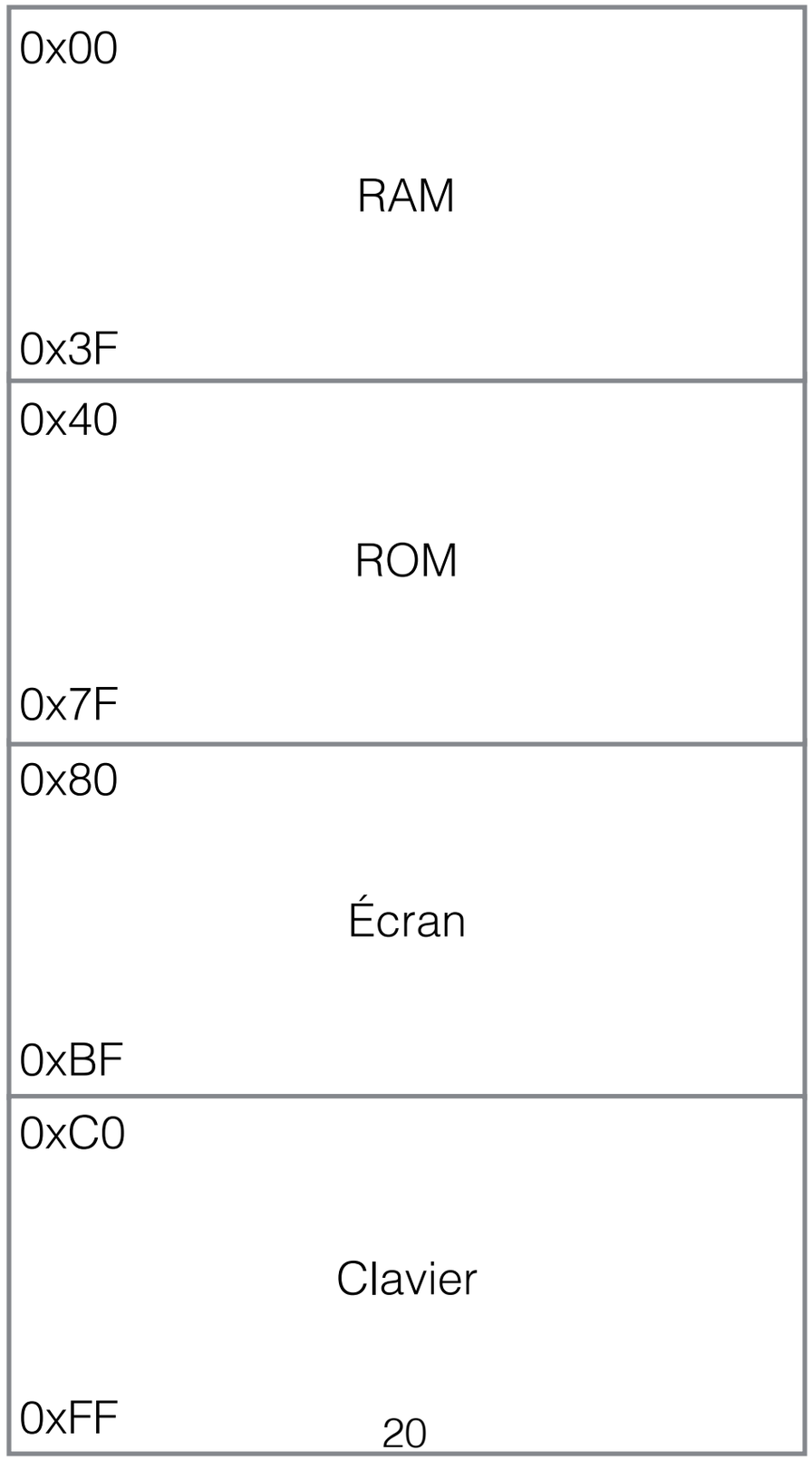
- Question:
  - une entrée, quatre sorties?

Bus: plusieurs périphériques

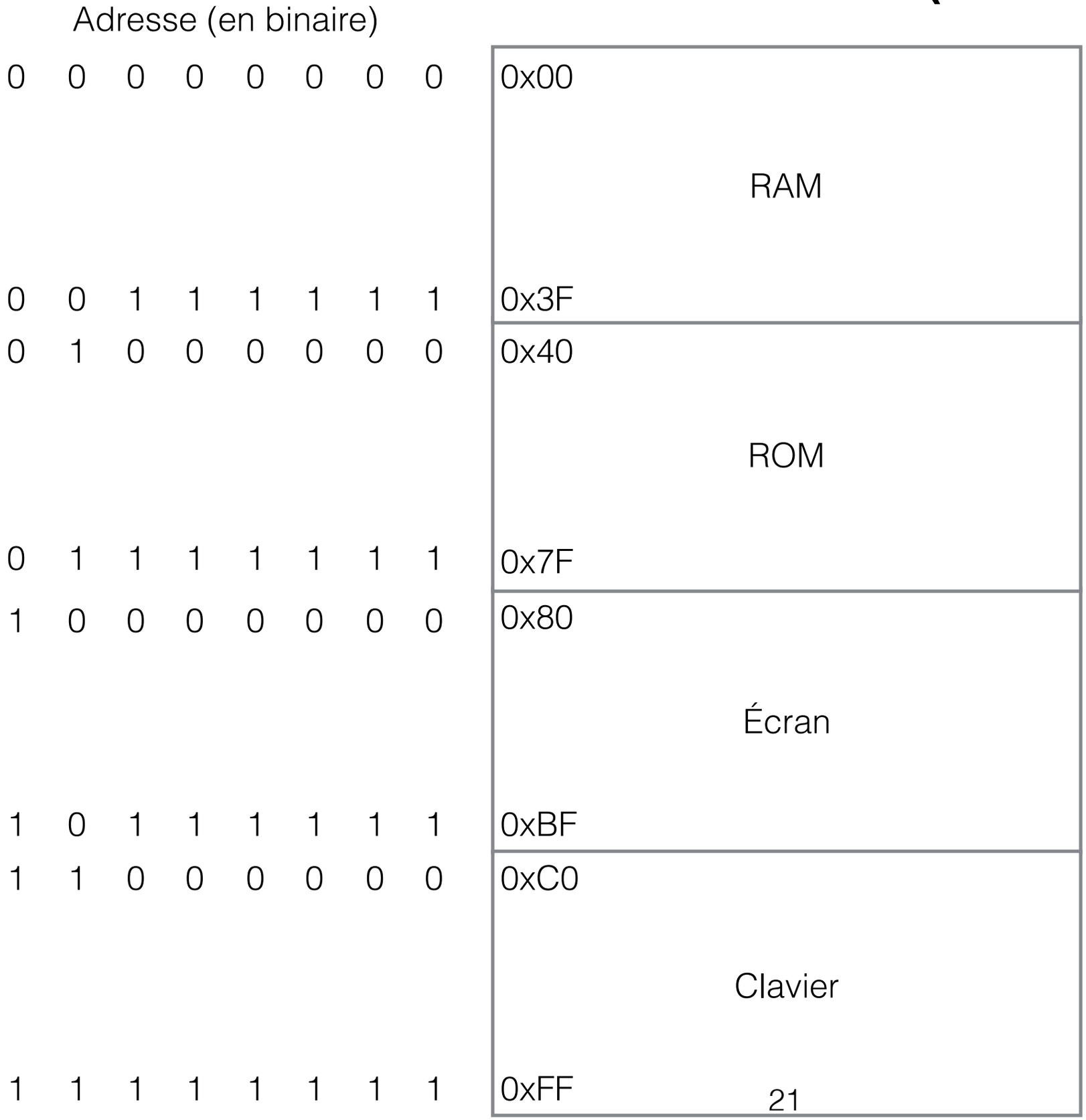


- Question:
  - quelle est la carte de la mémoire (*memory map*) de ce système?

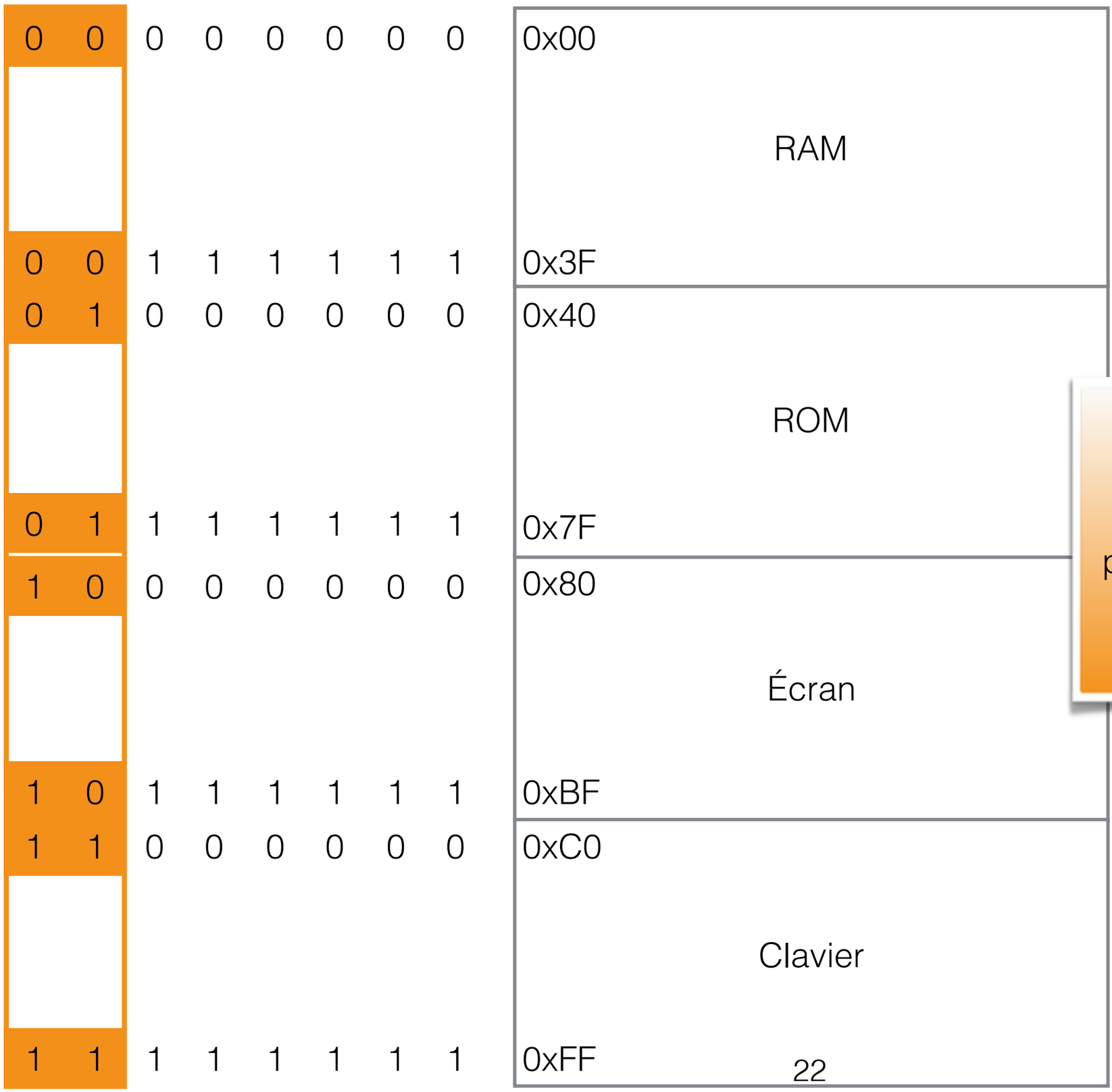
Carte de la mémoire (*memory map*)



Carte de la mémoire (*memory map*)



Carte de la mémoire (*memory map*)



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'usager, 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 évènements 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

Démonstration: TP1