GIF-3002
SYSTÈMES MICRO-PROCESSEURS ET INTERFACES

-Décodeurs d'adresses, bases (8.5)

4 et 5 Novembre 2009

Automne 2009

PLT-2704

U.S. Ganguly, Pr., responsable

M. Klein, MBA. M.Sc., chargé de cours

M. Clément-Bonhomme, B.ing, dépanneur

UNIVERSITÉ LAVAL

Décodeur d'adresses, bases (8.5)

Lorsque le CPU veut parler à un périphérique (ram, port série, un autre CPU etc.) il doit d'abord l'adresser, c'est-à-dire donner mettre l'adresse de ce périphérique sur le bus d'adresse.

C'est alors à tous les périphériques sur le BUS d'adresse de lire l'adresse. Chaque périphérique doit alors comprendre si l'adresse en cours s'adresse à lui ou non. Ce processus est le décodage d'adresse.

Le décodage d'adresse fait partie de la **glue logique** entre le BUS d'adresse et le périphérique. Le décodeur n'est presque jamais inclus avec ou dans le périphérique car sa forme dépend trop de l'architecture générale du montage.

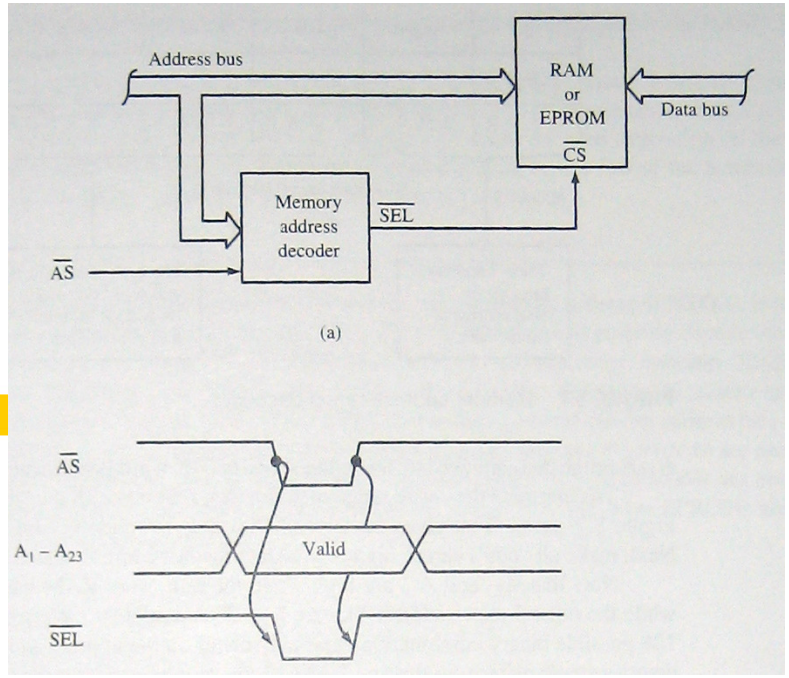
Un bloc de décodage d'adresse gère en général plusieurs périphérique

Décodeur d'adresses, bases (8.5)

La sortie du bloc de décodage est habituellement N broches, N étant le nombre de périphériques de 1 à n.

Si l'adresse sur le BUS s'adresse au périphérique i = 3, alors la glue logique devait activer sa broche #3 de sortie et désactiver les autres.

Les broches de sortie du décodeur sont connectées au ENABLE (ou équivalent: on voit souvent des noms comme E^, EN^, SEL^, CS^ ..) des périphériques sur le BUS. Ainsi le décodeur active le périphérique adressé et désactive (Hi-Z) tous les autres.



UNIVERSITÉ LAVAL

Décodeur d'adresses, bases (8.5)

Exemple 1 (livre p 269):

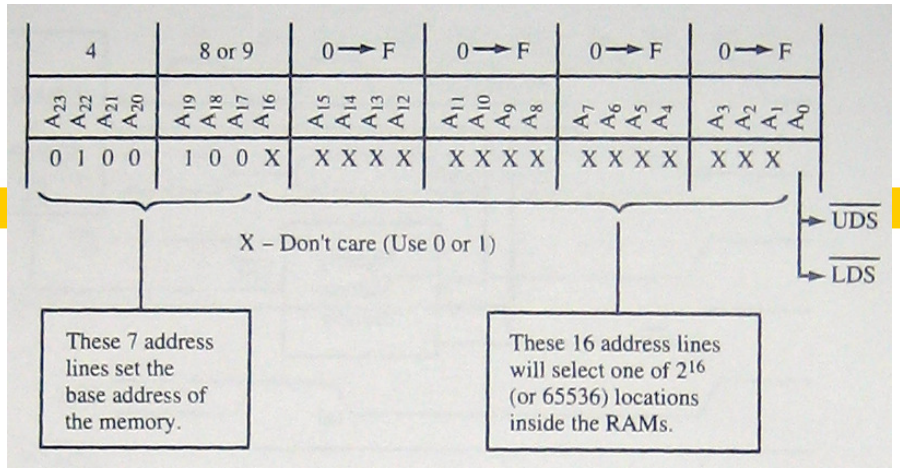
On veut connecter une ram de 64KW (Kilo Word, soient 2 circuits de 64KB, l'un fera les adresses paires, et l'autre les adresses impaires). Cette ram à l'adresse de base $48 000

Comme on doit adresser 64KW (65536 positions de 16 bits), il faut 16 lignes d'adresses (2^16 = 65536) donc de A1 à A17 (A0 n'étant pas disponible et servant à contrôler UDS^/LDS^).

Cela va donc de

$48 0000 = 0100 1000 0000 0000 0000 0000

à $49 FFFF = 0100 1001 1111 1111 1111 1111 (A0)

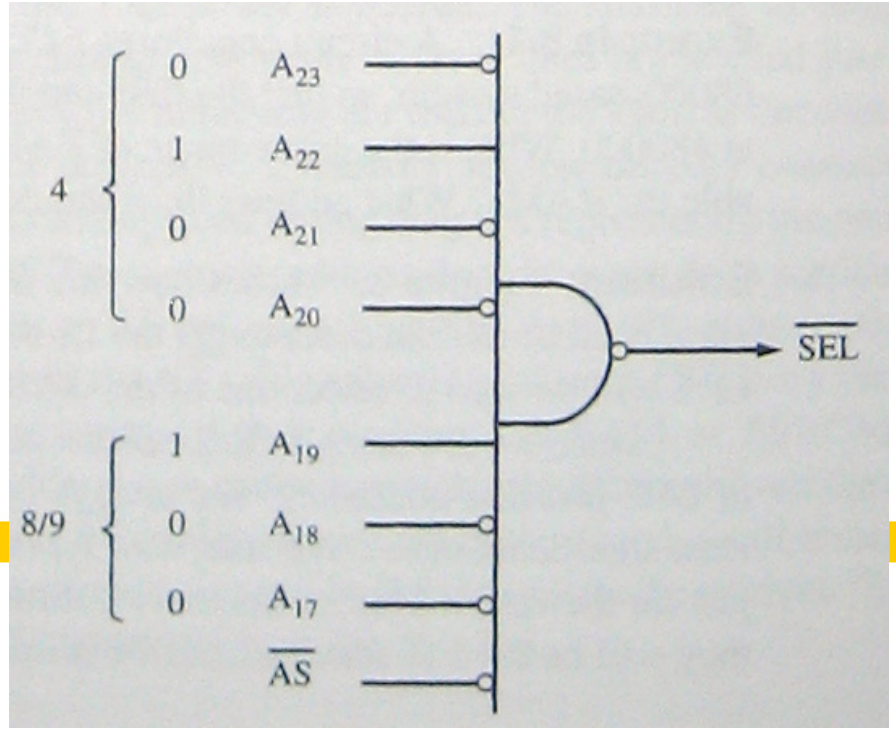


Décodeur d'adresses, bases (8.5)

Exemple 1 (livre p 269):

Le décodeur de = a23….a17a16 …. a1a0
$48 0000 = 0100 1000 0000 0000 0000 0000
à $49 FFFF = 0100 1001 1111 1111 1111 1111 (a0)

Est donc:
(ne pas oublier AS^)



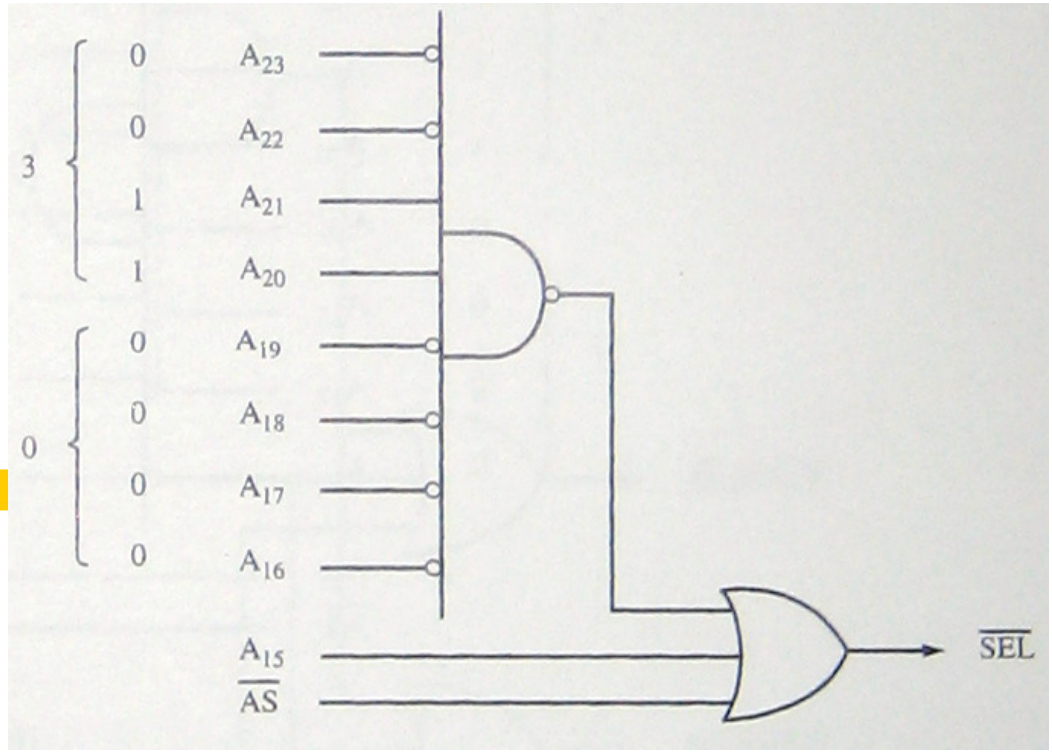
Décodeur d'adresses, bases (8.5)

Exemple 2 : 16KW de RAM, base = $30 0000 -> 14 bits de A14 à A1 car 2^14 = 16 834

Le décodeur de = a23…. a15a14 …. a1a0
$30 0000 = 0011 0000 0000 0000 0000 0000
à $30 7FFF = 0011 0000 0111 1111 1111 1111 (a0)

Est donc:
(ne pas oublier AS^)

Pourquoi a15 et AS^ sont à part?



Décodeur d'adresses, bases (8.5)

Exemple 3 : 16KW de ROM, base = $60 0000 et 16KW de RAM, base $70 000

Décodeur ROM = a23…. a15a14 …. a1a0
$60 0000 = 011**0** 0000 0000 0000 0000 0000
à $60 7FFF = 011**0** 0000 0111 1111 1111 1111 (a0)

+

Décodeur RAM = a23…. a15a14 …. a1a0
$70 0000 = 011**1** 0000 0000 0000 0000 0000
à $70 7FFF = 011**1** 0000 0111 1111 1111 1111 (a0)

Décodeur ROM = a23…. a15a14 …. a1a0

$60 0000 = 0110 0000 0000 0000 0000 0000

à $60 7FFF = 0110 0000 0111 1111 1111 1111 (a0)

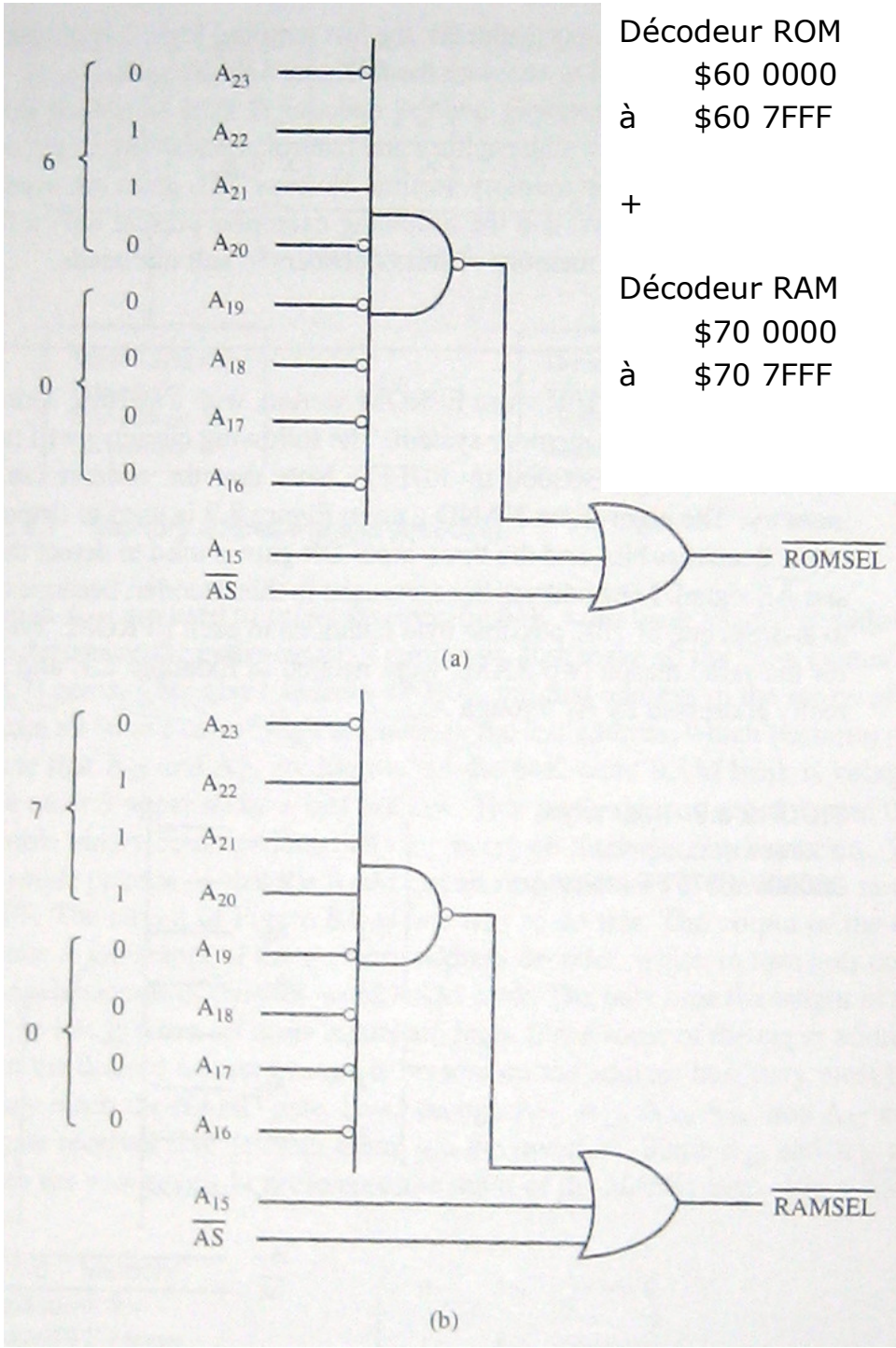
+

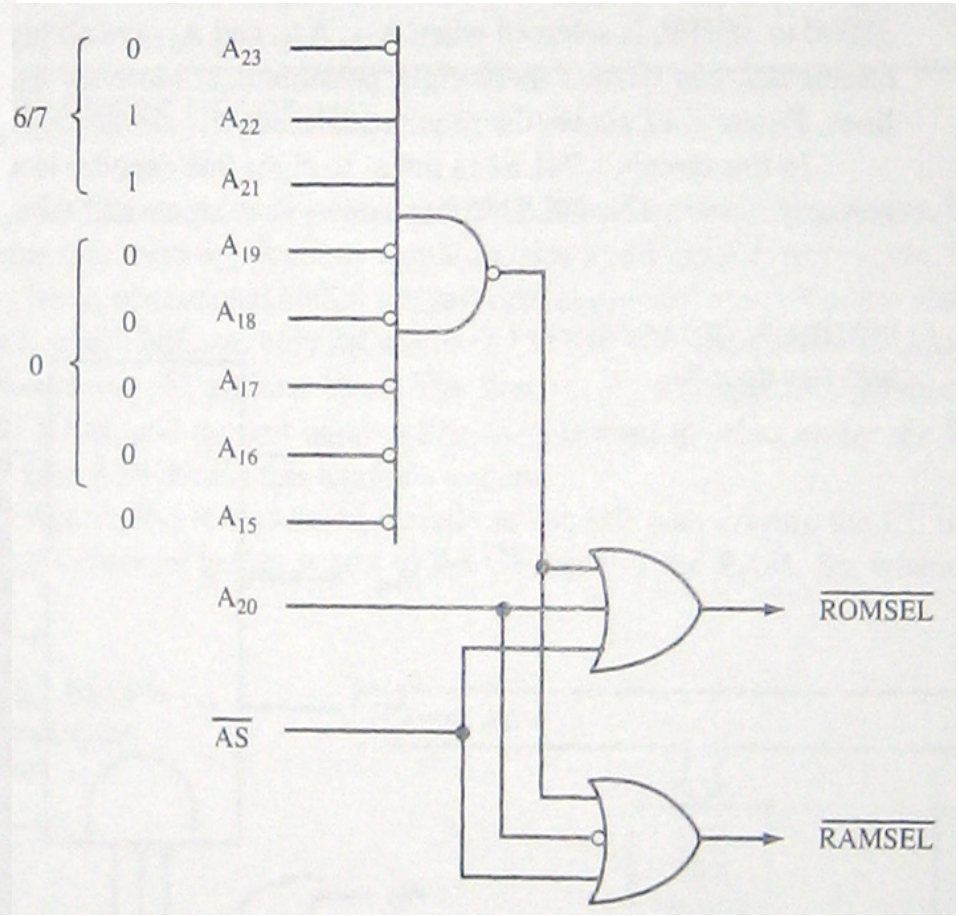
Décodeur RAM = a23…. a15a14 …. a1a0

$70 0000 = 0111 0000 0000 0000 0000 0000

à $70 7FFF = 0111 0000 0111 1111 1111 1111 (a0)

Ce qui ce simplifie par le schéma ci-dessous:





Décodage partiel (8.6)

Décodage partiel: On ne tient pas compte des MSB. Seuls les LSB « utiles » sont pris en compte:

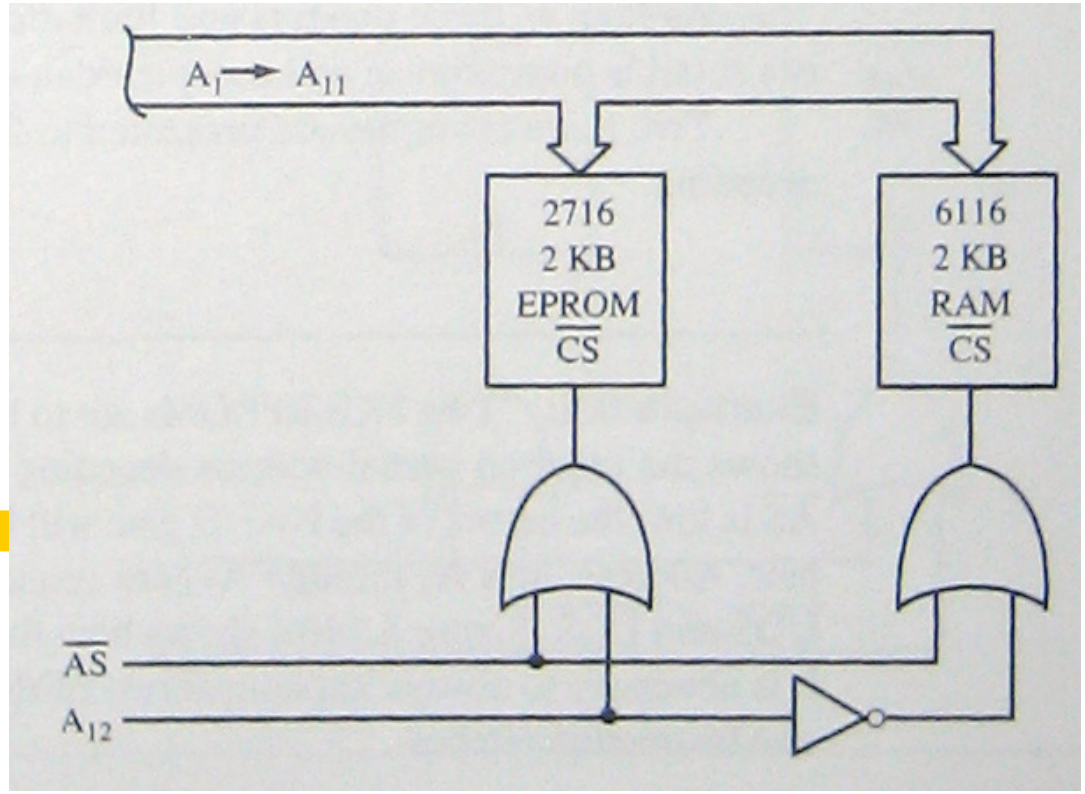
Ex:

EPROM 2KW base1 = \$00 0000 = (~~0000-0000-000~~)**0 0000 0000 0000**

RAM 2KW, base2 = \$00 1000 = (~~0000-0000-000~~)**1 0000 0000 0000**

Avantage: décodeur plus simple.

Inconvénient: si on veut ajouter d'autres périphérique par la suite il faudra refaire une part Plus importante du décodeur



Décodage partiel (8.6)

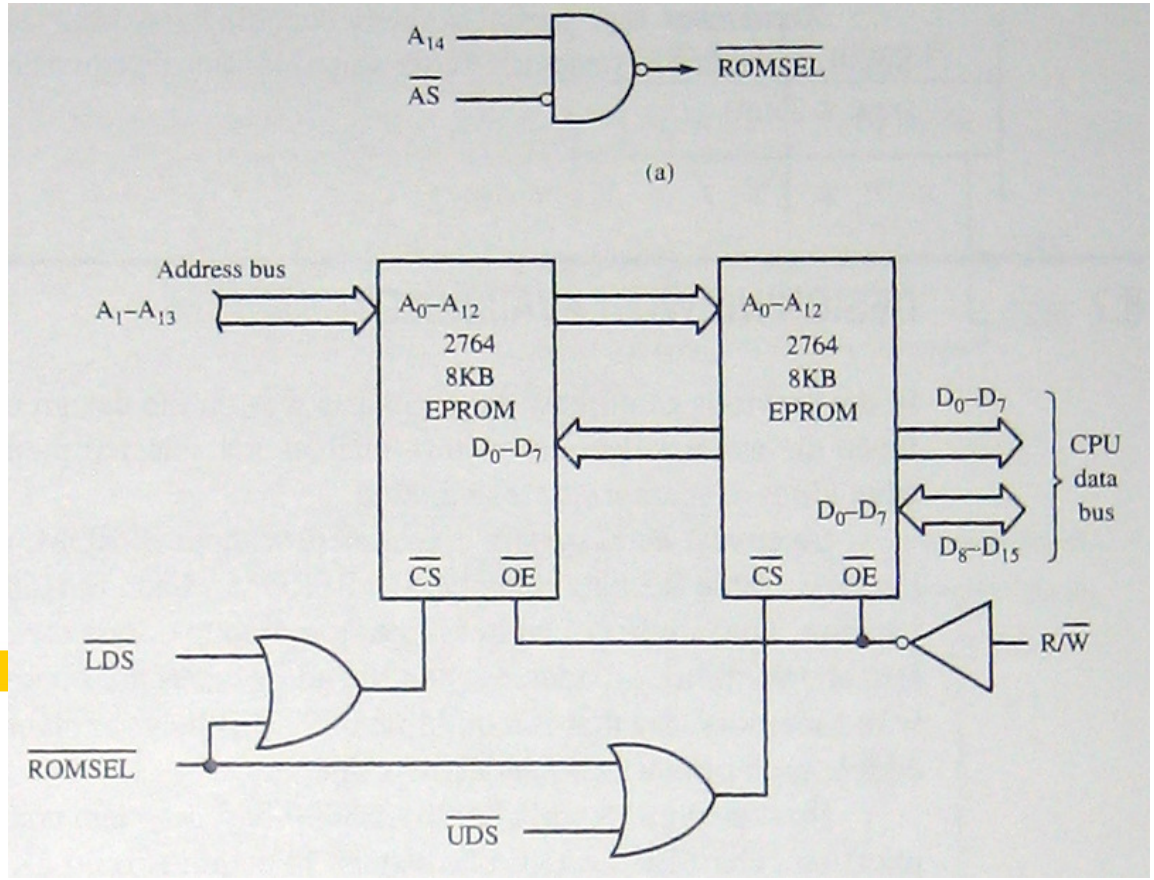
Ex: EPROM 8KW base = $ 00 4000 = ~~0000 0000 0~~**100 0000 0000 000**~~0~~

On indique cette fois-ci comment on accède à l'une ou l'autre des 2 EPROM de 8 KB chacune pour obtenir les 8 KW, grâce à **LDS^, UDS^**:

CS^= Chip Enable
OE^= Output Enable

Note1: OE^ signifie en réalité bus de données. Quand CS^= 1, OE^ est sans effet.

Note2: Le ROMSEL^ du haut se connecte en bas du circuit.



UNIVERSITÉ LAVAL