- Question de compréhension sur ce tableau

	$M/\overline{IO} = 0$	$M/\overline{IO} = 1$
$\overline{RD} = 0$ (actif)	Entrée IN	lecture
$\overline{WR} = 0$ (actif)	Sortie OUT	écriture

Seq 1)

Exo 1.1

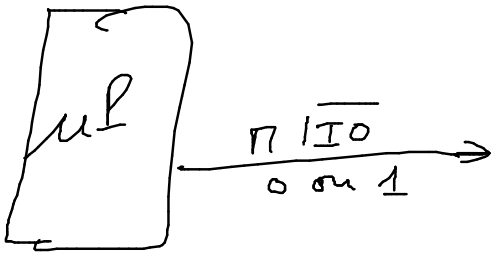
q1) logique de décodage

q2) bus de données

q3) a) lecture : Rom
b) écriture : µP
c) entrée : dispositif d'entrée
d) sortie : µP

q4) a) économie de signaux sur la carte électronique
b) IN et OUT

q5) M/$\overline{IO}$ distingue les accès $\overline{IO}$ des accès mémoires



16) Mem (RAM) : on lit à l'@ n la donnée qu'on a écrite précédemment à cette adresse

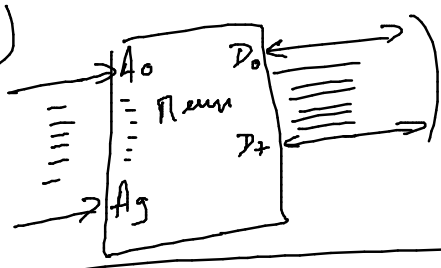
IO : ce qu'on sort à l'adresse n ce n'est pas ce qu'on entre ensuite à partir de cette adresse (pas de mémorisation)

17) si n est le nombre de signaux d'adresses, le nombre d'adresses ≠ qu'on peut coder sur ces n signaux est 2^n EX: 16 signaux d'adresses A_0 -- A_{15} permettent de coder 2^{16} adresses ≠

$2^{16} = 65536$ adresses

000 ——— 0
001
010
011
⋮
111 ——— 1

98)



2^{10} mots de 8 bits
2^{10} octets

Seq 2 Exo 2.1

91) 2 cas : - fetch : lecture d'un code machine
- lecture demandée par l'exécution d'une instruction (lecture programmée)

92) 2 types d'info sur le bus de données
- codes machine
- données manipulées par les instructions (variables)

93) 9600 b/s ≃ 10000 b/s
1 carac 10 bits par caractère
1000 caractères par seconde
20 ms pour 20 caractères ⟺ (1 carac par ms.)

50 Mips = 50 millions d'instructions par seconde

soit 1 million d'instructions en 1/50e de seconde

20 ms

- le temps d'émettre 20 caractères sur un modem, ce processeur a exécuté 1 million d'instructions

overhead : partie de puissance utilisée par l'OS.

Seq 3) Exo 3.1

1) octet = mot de 8 bits
mot = mot de 16 bits = 2 octets
double mot = mot de 32 bits

1 adresse ⟶ 1 octet

2 | 2000 : 4A) 1er mot
| 2001 : 2B)
| 2002 : 3A) 2ème mot
| 2003 : 41

distance de 2 entre 2 mots consécutifs

Q2) little-endian : poids faible est stocké en premier à l'@ n
poids fort en second (@ n+1)

poids fort	poids faible
20	00
+ 20	12
= 40	12

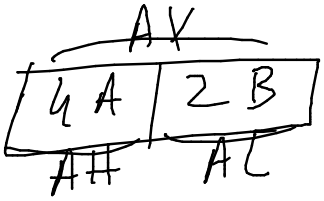
Q3) les registres n'ont pas d'adresses mais un nom AX ou SI

NB : certains registres contiennent une adresse. Ex : SI est un registre d'adresses, son contenu sa valeur peut être envoyé sur le bus d'adresses ou ?

74) registres du 8086 : taille de 16 bits

75) AH et AL ⟷ AX
BH et BL ⟷ BX
CH et CL ⟷ CX
DH et DL ⟷ DX
} 16 bits

DH : 8 bits, DL : 8 bits



76) indicateur = 1 bit
(= flag)

CF = carry flag = retenue
ZF = Zero flag = indicateur de résultat nul
SF = Sign flag = indicateur de résultat négatif

1-7) AX, BX, CX, DX = registres adressables

7-8) 20 signaux d'adresses = A_0 à A_{19}

$\Rightarrow 2^{20}$ adresses $\neq$ donc 2^{20} octets $\neq$

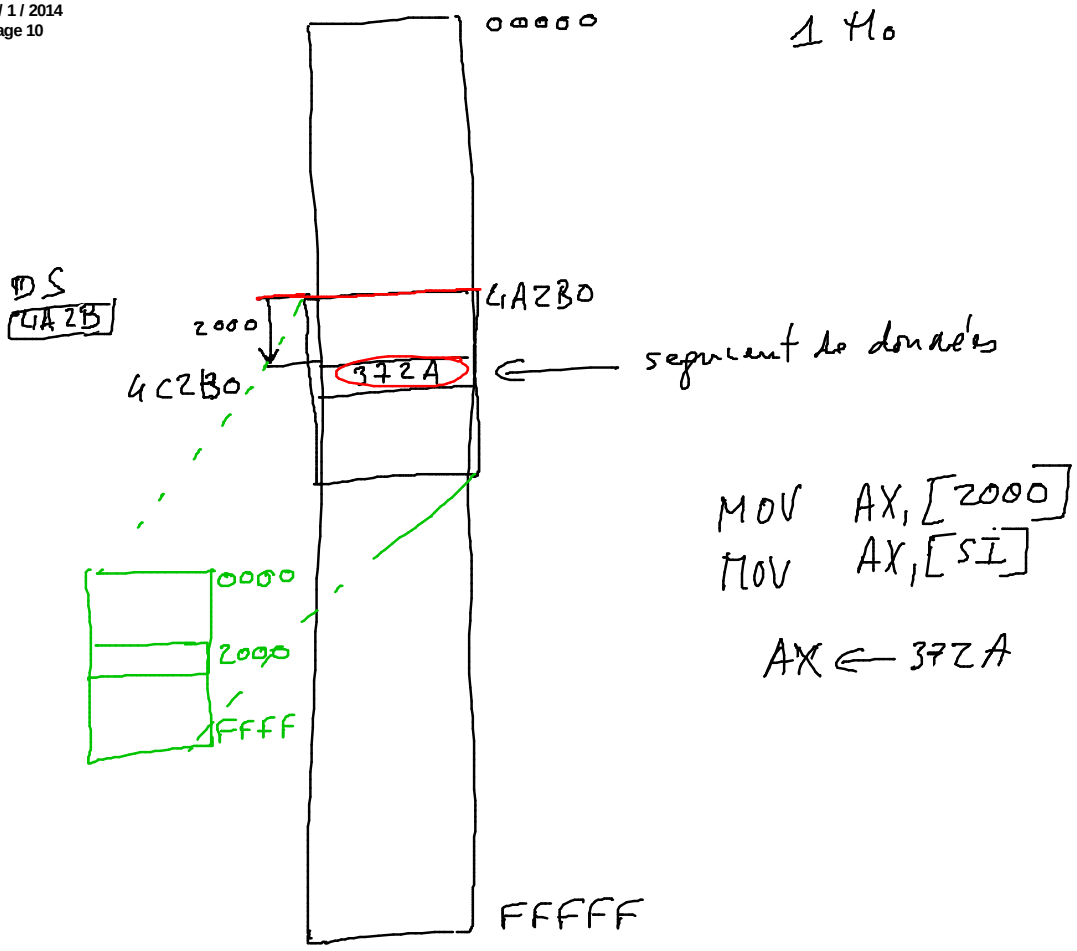
$2^{20} = \underline{1\ Mo}$

9-9) une adresse offset est relative à l'adresse de début d'un segment

les registres de segment du 8086 sont CS, DS, SS, ES

Code Segment, Data Segment, Stack Segment

- les registres de segment servent à positionner les segments dans la mémoire totale
- un segment est un espace adressable de 2^{16} adresses (donc 2^{16} octets)



MOV AX, [2000]
MOV AX, [SI]

AX ⟵ 372A

q10) @ Y = @ envoyée sur le bus d'adresses
↳ taille de 20 bits
@ offset = taille de 16 bits
@ segment = ce qu'on met dans un registre
de segment = taille de 16 bits
@ segmentée = @ spécifiée sous forme du
couple @ segment : @ offset
↳ taille de 32 bits

q11) 4A2B : 027C

$$\begin{array}{r} 4A2B0 \\ +\ 027C \\ \hline = 4A52C \end{array}$$

q12) mot = 16 bits = 2 octets
double mot = 32 bits = 4 octets
paragraphe = 128 bits = 16 octets
↳ 16 décimal
10 hexa

4A2B0
↳ se termine par un 0
multiple de 10 hexa
↳ appelle aussi frontière de paragraphe

q12) max : 64 ko

q13) multiple de 16 décimal
ou 10 hexa

q14) CS : 0100 ⟺ 427C : 0100
Vrai si CS contient 427C

11)

BX, SI ou DI spécifient l'adresse offset
↳ associés à DS
MOV AX, [SI] → accès au segment de données

BP, SP
↳ associés à SS
MOV AX, [BP] → accès au segment de pile

PUSH AX
on empile à une adresse offset fournie par SP et associée à SS

IP = Instruction Pointer
↳ associé au segment de code

9 16)