



DIRO
IFT 1215

INTRODUCTION AUX SYSTÈMES INFORMATIQUES

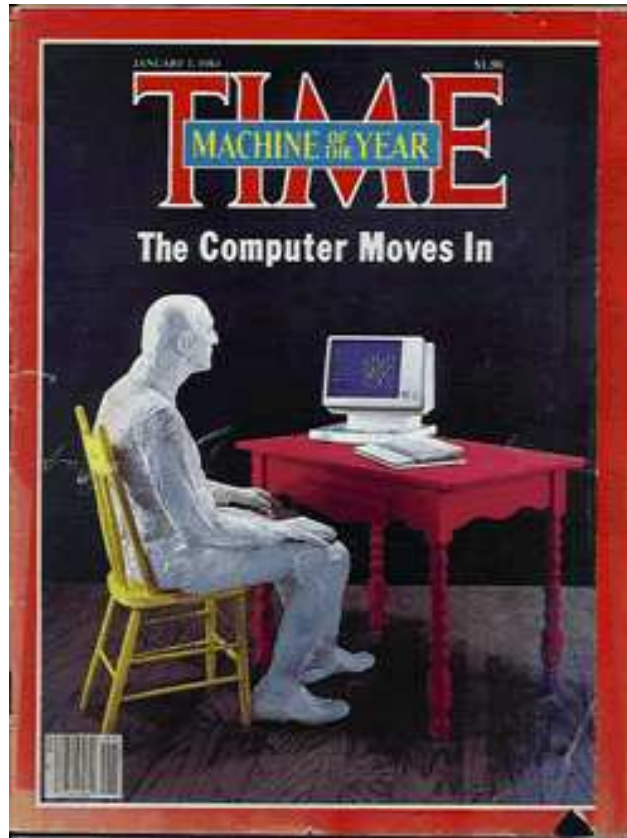
ADDENDUM

Max Mignotte

Département d'Informatique et de Recherche Opérationnelle
Http: [//www.iro.umontreal.ca/~mignotte/](http://www.iro.umontreal.ca/~mignotte/)
E-mail: mignotte@iro.umontreal.ca

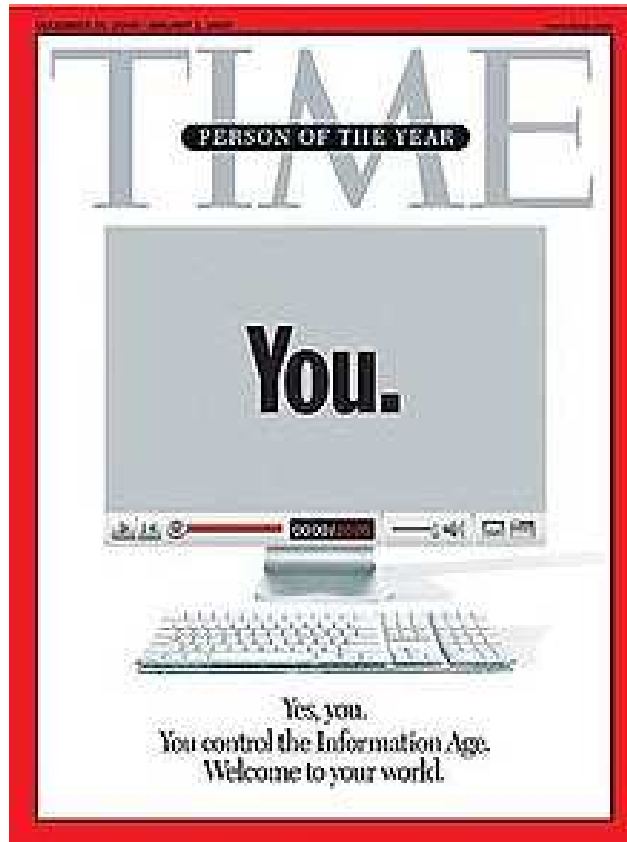
INTRODUCTION
TIMES MAGAZINE: Man of the Year 1982

INTEL & MS/DOS & IBM



INTRODUCTION
TIMES MAGAZINE: Man of the Year 2006

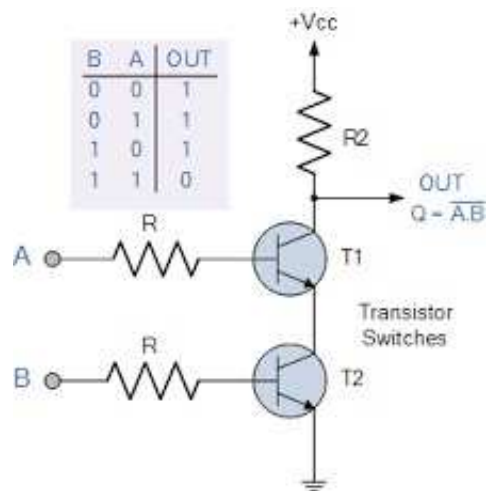
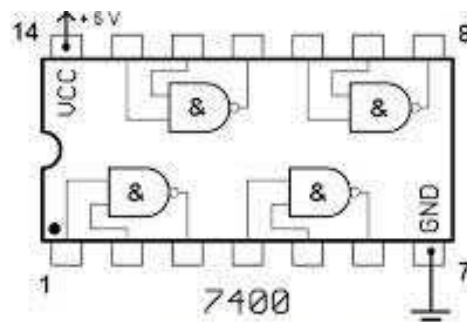
Profession : Informaticien



CIRCUITS LOGIQUES

Porte NAND

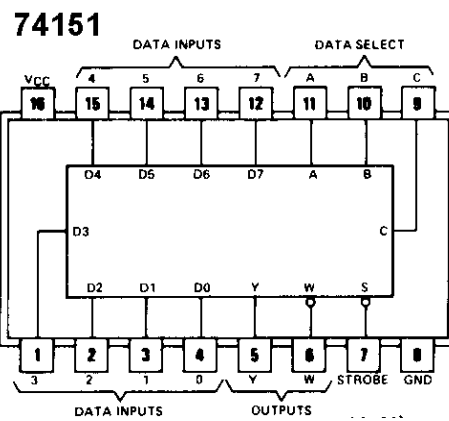
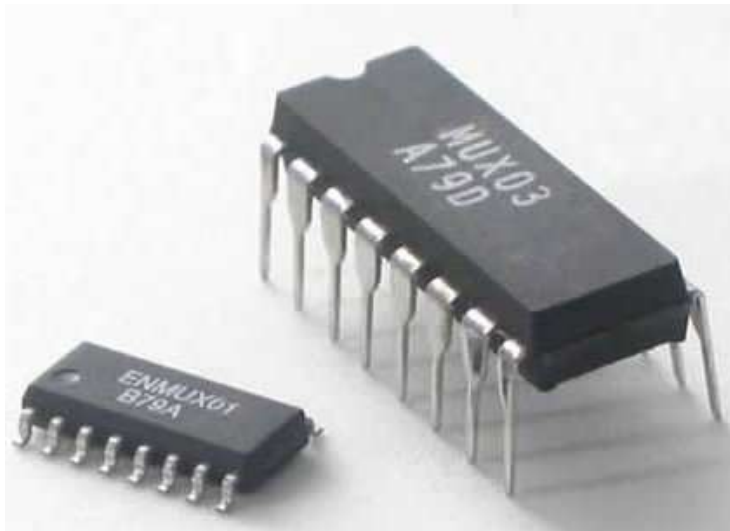
Porte NAND



CIRCUITS LOGIQUES

Multiplexeurs [1/2]

Multiplexeurs



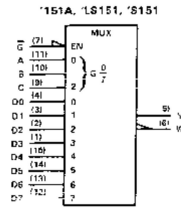
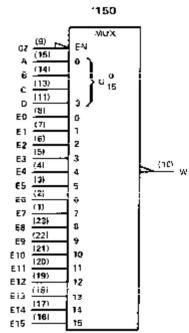
CIRCUITS LOGIQUES

Multiplexeurs [2/2]

Multiplexeurs

DATA SELECTORS/MULTIPLEXERS

logic symbols†



†These symbols are in accordance with ANSI/IEEE Std. 91-1984 and IEC Publication 617-12. Pin numbers shown are D, J, N, and W packages.

74150

FUNCTION TABLE

INPUTS				STROBE EN	OUTPUT	
D	C	B	A		Y	W
X	X	X	X	H	H	H
L	L	L	L	L	E0	E0
L	L	L	H	L	E1	E1
L	L	H	L	L	E2	E2
L	L	H	H	L	E3	E3
L	H	L	L	L	E4	E4
L	H	L	H	L	E5	E5
L	H	H	L	L	E6	E6
L	H	H	H	L	E7	E7
H	L	L	L	L	E8	E8
H	L	L	H	L	E9	E9
H	L	H	L	L	E10	E10
H	L	H	H	L	E11	E11
H	H	L	L	L	E12	E12
H	H	L	H	L	E13	E13
H	H	H	L	L	E14	E14
H	H	H	H	L	E15	E15

74151A, 74151, 74151

FUNCTION TABLE

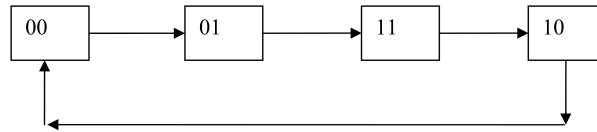
INPUTS				STROBE EN	OUTPUTS	
D	C	B	A		Y	W
X	X	X	X	H	L	H
L	L	L	L	L	D0	D0
L	L	L	H	L	D1	D1
L	L	H	L	L	D2	D2
L	L	H	H	L	D3	D3
L	H	L	L	L	D4	D4
L	H	L	H	L	D5	D5
L	H	H	L	L	D6	D6
L	H	H	H	L	D7	D7

H = high level, L = low level, X = irrelevant
 E0, E1, ... E15 = the complement of the level of the respective E input
 D0, D1, ... D7 = the level of the D respective input

CIRCUITS LOGIQUES

Compteur Code Gray

On voudrait construire ce qu'on appelle un compteur appelé compteur Code Gray sur 2 bits. Ce compteur parcourt le cycle suivant (0 → 1 → 3 → 2 → 0) :



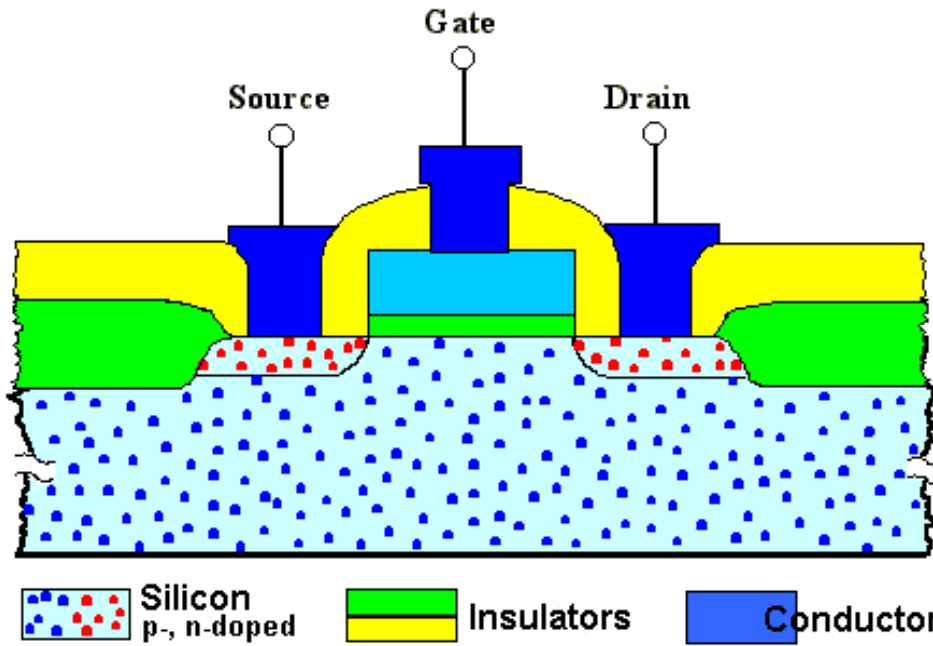
a) Remplir la table de vérité qui donne l'état futur en fonction de l'état présent :

Q_1'	Q_0'	Q_1^{t+1}	Q_0^{t+1}
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

b) Donner les expressions booléennes pour Q_1^{t+1} et Q_0^{t+1} en fonction de Q_1' et Q_0' (expressions aussi simplifiées que possible en utilisant possiblement la table de Karnaugh):

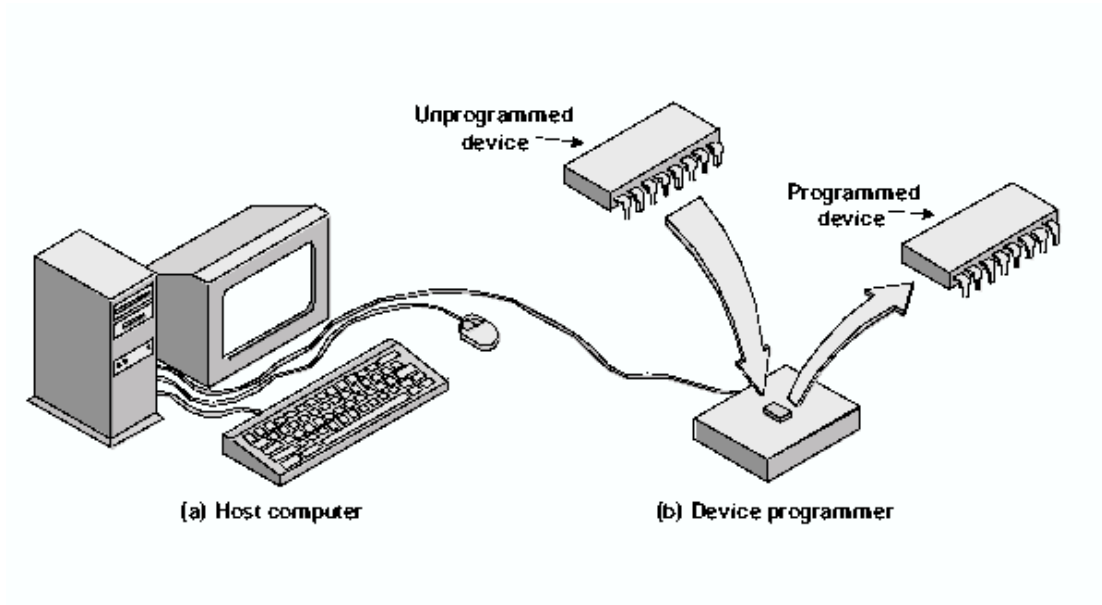
CIRCUITS LOGIQUES

Transistor



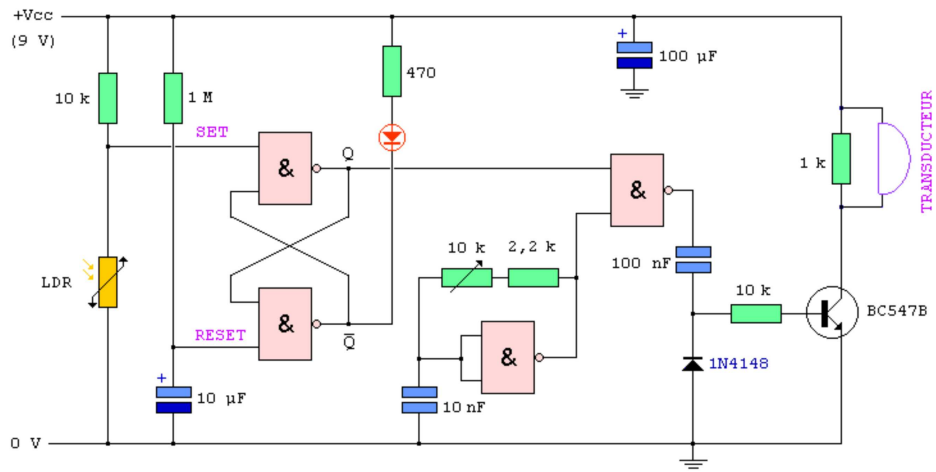
CIRCUITS LOGIQUES

Programmation d'une PLA



CIRCUITS LOGIQUES

Détecteur de Lumière



LANGAGE MACHINE

Comment "Ils" font sauter les protections [1/2]

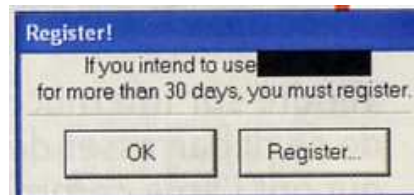
Comment un pirate "cracke" t-il un shareware ?

I. La caisse à outils du pirate

Un simple PC sous Windows + un debugger
(Le debugger permet de suivre, instruction après instruction, le fonctionnement du logiciel mais aussi le mettre en PAUSE)

II. Premières découvertes

Au lancement du shareware



III. Une valeur fantaisiste



A ce moment, le pirate lance le debugger et lui demande de suspendre le fonctionnement du logiciel dès que celui-ci utilise la commande

`User32.GetDlgItemTextA`

LANGAGE MACHINE

Comment "Ils" font sauter les protections [2/2]

IV. Examen de mémoire

Le debugueur a détecté la commande:

User32.GetDlgItemTextA et affiche les instructions en cours d'exécution, pas à pas que l'on peut suivre ...



```
0008:8067A5AE C3
0008:8067A5AF 90
0008:8067A5B0 33C0
0008:8067A5B2 A024F0DFF
0008:8067A5B7 C60524F0DFF
0008:8067A5BE C3
0008:8067A5BF 90
(DISPATCH)-KTEB(8053F230)-
Screen found at BB0660F0
Screen width: 80
Screen height: 60
NTICE: Load32 START=750A06
NTICE: Exit32 PID=494 MOI
```

Le nom d'utilisateur est lu, puis stocké dans une zone mémoire. Ensuite c'est le numéro de série qui est enregistré.

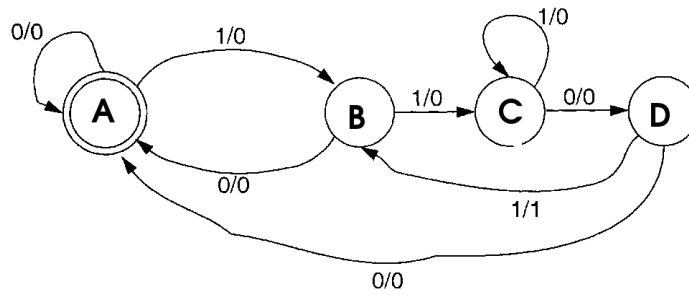
Le pirate demande au debugueur de suspendre le fonctionnement du logiciel lors de l'utilisation de la commande

User32.IstrcmpA

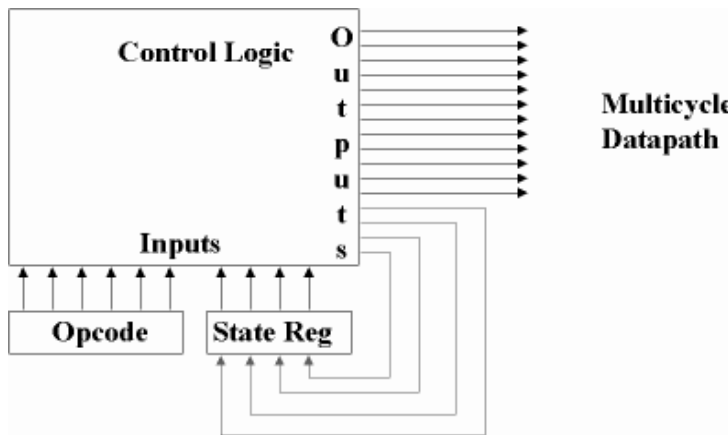
...puis interrompt la pause pour voir la suite des opérations... **BINGO** une des 2 adresses de cette commande correspond à la zone mémoire du code enregistré et l'autre correspond au code du shareware !

LANGAGE MACHINE

Séquenceur Microprogrammé [1/3]

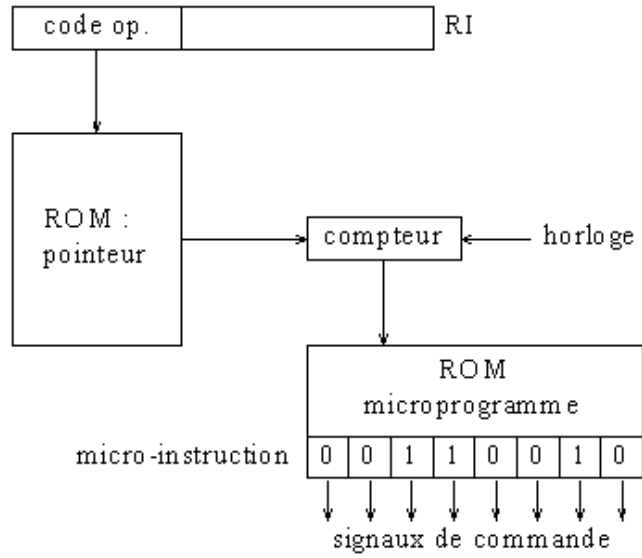


$q_1(t)$	$q_0(t)$	E	$q_1(t+1)$	$q_0(t+1)$	S
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1



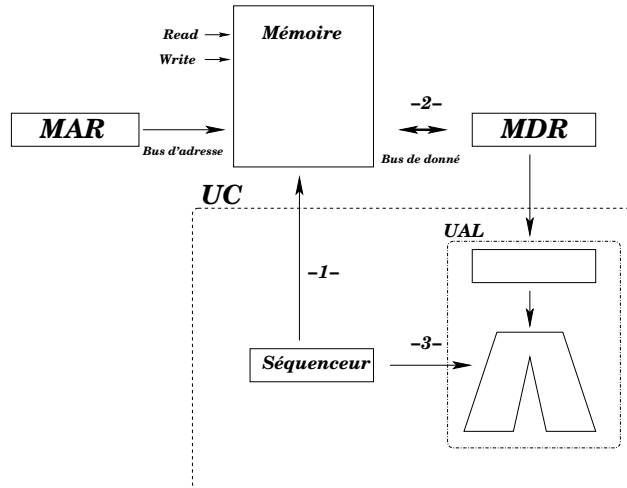
LANGAGE MACHINE

Séquenceur Microprogrammé [2/3]



LANGAGE MACHINE

Séquenceur Microprogrammé [3/3]



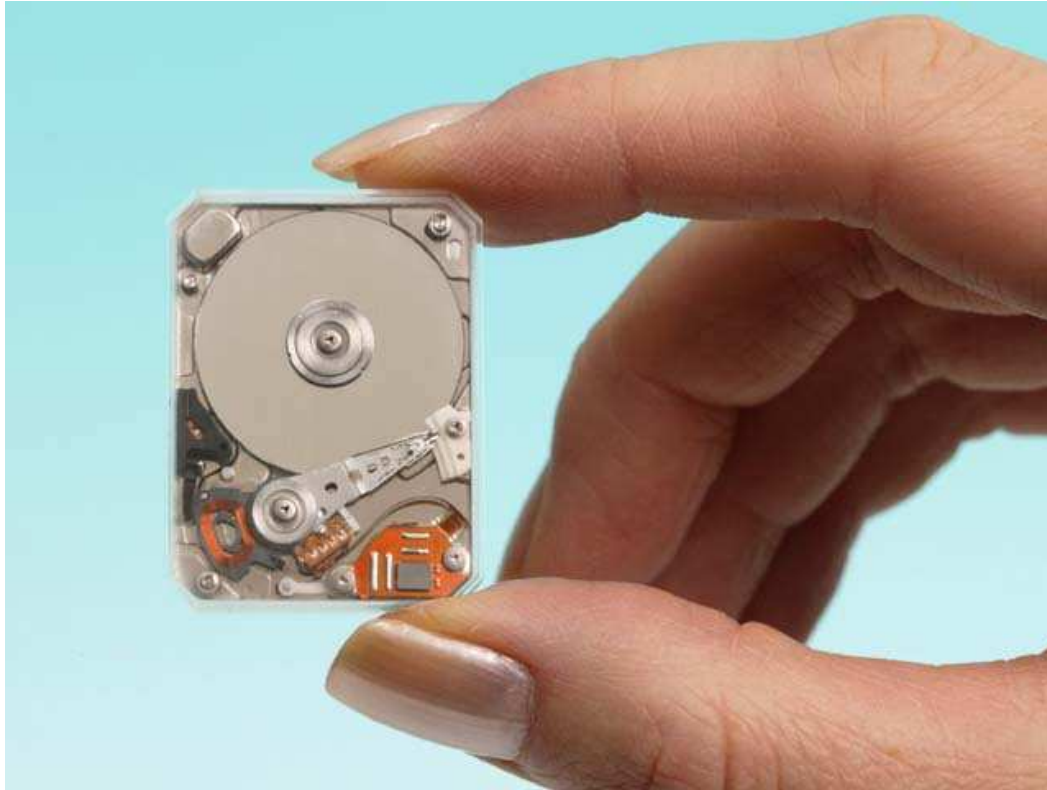
Read	Write	Read	Write	Read	Write	Read	Write
Mémoire		MDR		MAR		ACC	

LANGAGE MACHINE

Exercice

	IN	
	STO	AD1
	IN	
	STO	AD2
	LDA	AD1
LOOP	STA	DTINST
DTINST	LDA	TEM
	OUT	
	LDA	AD1
	ADD	ONE
	STO	ADC
	SUB	AD2
	BZ	STOP
	LDA	ADC
	BR	LOOP
STOP	HLT	
ADC	DAT	0
AD1	DAT	0
AD2	DAT	0
ONE	DAT	1
TEM	DAT	0

ENTRÉE/SORTIE
LE PLUS PETIT DISQUE DUR !



Toshiba

10 grammes et 0.85" capacité 8 Go
3600 tours/min
Destiné au portable PDA

SHELL SCRIPTS

Exemples [1/2]

```
#!/bin/bash
# avg.sh script

avg=0
for val in $*
do
let avg=avg+$val
done
let avg=avg/$#
echo avg=$avg

./avg.sh 100 70 80
> avg=83
```

```
for i in *
do
j='echo $i | tr A-Z a-z'
mv $i $j
done
```

**Changera les noms de fichier
de Majuscule en minuscule**

```
ls -la -R | grep 'Mar 19'
find -mtime 4
find -ctime 4
```

**Affichera en listing ou trouvera des fichiers:
modifiés le 19 mars ou modifiés ou créés
il y a moins de 4 jours**

SHELL SCRIPTS

Exemples [2/2]

```
#!/bin/bash
# find2Str.sh script
for file in *
do
if grep $1 $file && grep $2 $file
then
echo $file trouvé
fi
done
```

Recherchera tous les fichier
ayant les deux chaines de caracteres passées
en arguments au programme

```
cat file | while read line
do
moyenne='echo $line | cut -d: -f6'
echo $moyenne > tmp.dat
done
```

```
cpt=1
Nblines='cat file.dat | wc -l'
while [ $cpt -le $Nblines ]
do
line='head -n $cpt file.dat | tail -n 1'
done
```

```
file.dat
NOM      :CODE      : .Tp: .Intra: .Final: .Moy
Actarus  :ACTA28658 : 45: 75: 100: 95
Alcor    :ALCO18083 : 83: 64: 95: 87
Venusia  :VENU13097 : 83: 55: 95: 85
```