DIRO
IFT 1215

INTRODUCTION AUX SYSTÈMES INFORMATIQUES

ADDENDUM

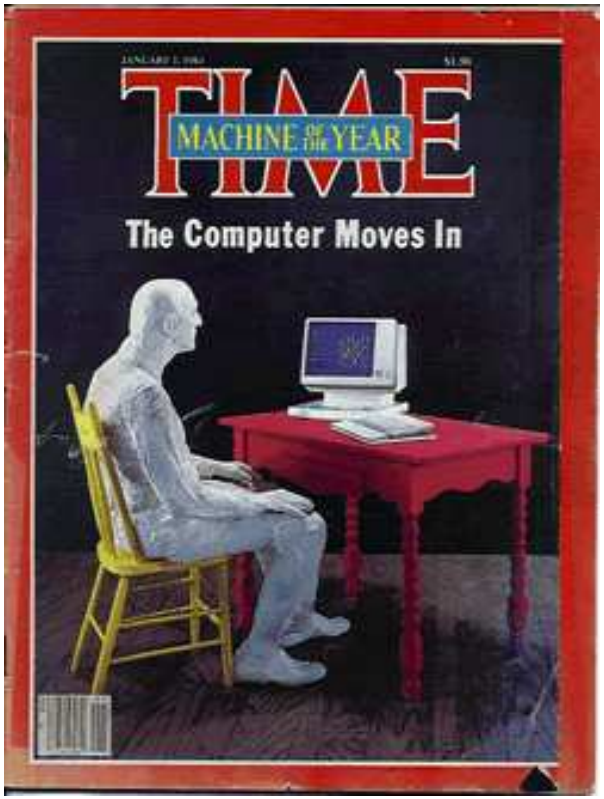
Max Mignotte

Département d'Informatique et de Recherche Opérationnelle
Http: //www.iro.umontreal.ca/∼mignotte/
E-mail: mignotte@iro.umontreal.ca

INTRODUCTION

TIMES MAGAZINE: Man of the Year 1982

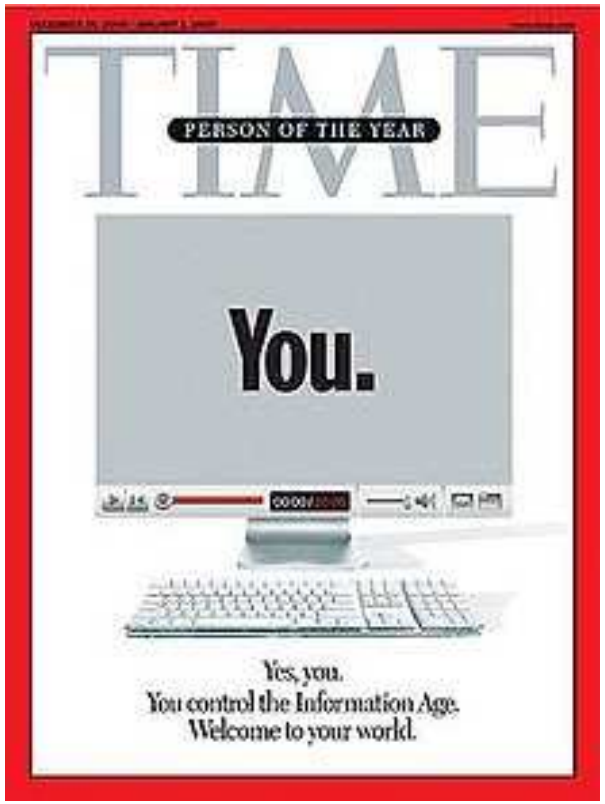
INTEL & MS/DOS & IBM



INTRODUCTION

TIMES MAGAZINE: Man of the Year 2006

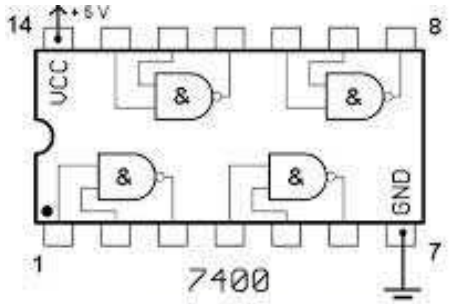
Profession : Informaticien

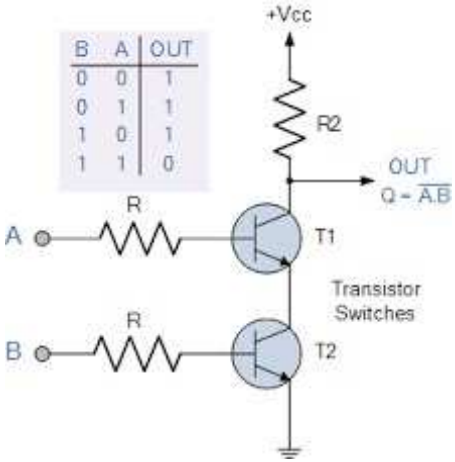


CIRCUITS LOGIQUES

Porte NAND

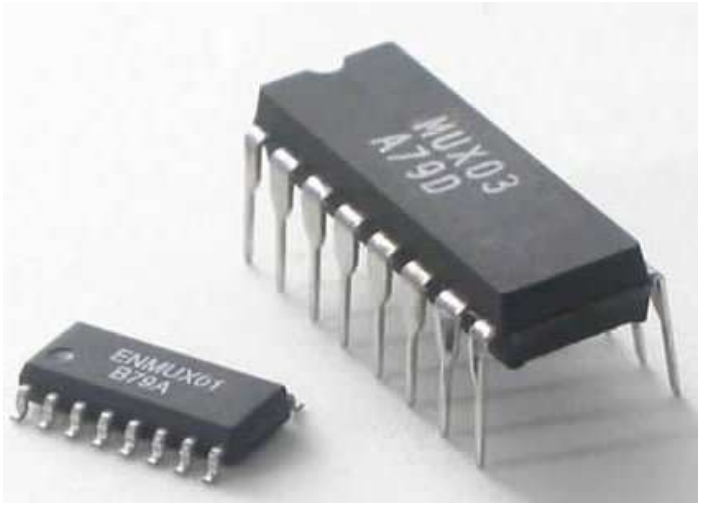
Porte NAND

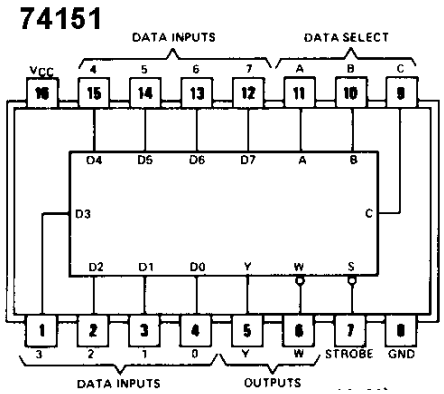




CIRCUITS LOGIQUES
Multiplexeurs [1/2]

Multiplexeurs





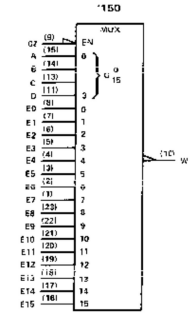
CIRCUITS LOGIQUES
Multiplexeurs [2/2]

Multiplexeurs

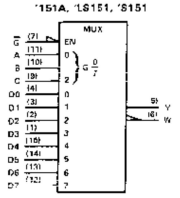
SN74150, SN74151A, SN74LS151, SN74S151
DATA SELECTORS/MULTIPLEXERS

logic symbols†

'150



'151A, 'LS151, 'S151



†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.

'150
FUNCTION TABLE

INPUTS					OUTPUT
SELECT			STROBE		
D	C	B	A	$\overline{G}$	W
X	X	X	X	H	H
L	L	L	L	L	$\overline{E0}$
L	L	L	H	L	$\overline{E1}$
L	L	H	L	L	$\overline{E2}$
L	L	H	H	L	$\overline{E3}$
L	H	L	L	L	$\overline{E4}$
L	H	L	H	L	$\overline{E5}$
L	H	H	L	L	$\overline{E6}$
L	H	H	H	L	$\overline{E7}$
H	L	L	L	L	$\overline{E8}$
H	L	L	H	L	$\overline{E9}$
H	L	H	L	L	$\overline{E10}$
H	L	H	H	L	$\overline{E11}$
H	H	L	L	L	$\overline{E12}$
H	H	L	H	L	$\overline{E13}$
H	H	H	L	L	$\overline{E14}$
H	H	H	H	L	$\overline{E15}$

'151A, 'LS151, 'S151
FUNCTION TABLE

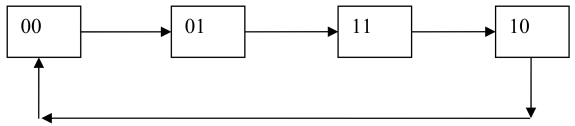
INPUTS				OUTPUTS	
SELECT			STROBE		
C	B	A	$\overline{G}$	Y	W
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

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

TEXAS INSTRUMENTS
POST OFFICE BOX 655012 • DALLAS, TEXAS 75265

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 ($0 \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 0$) :

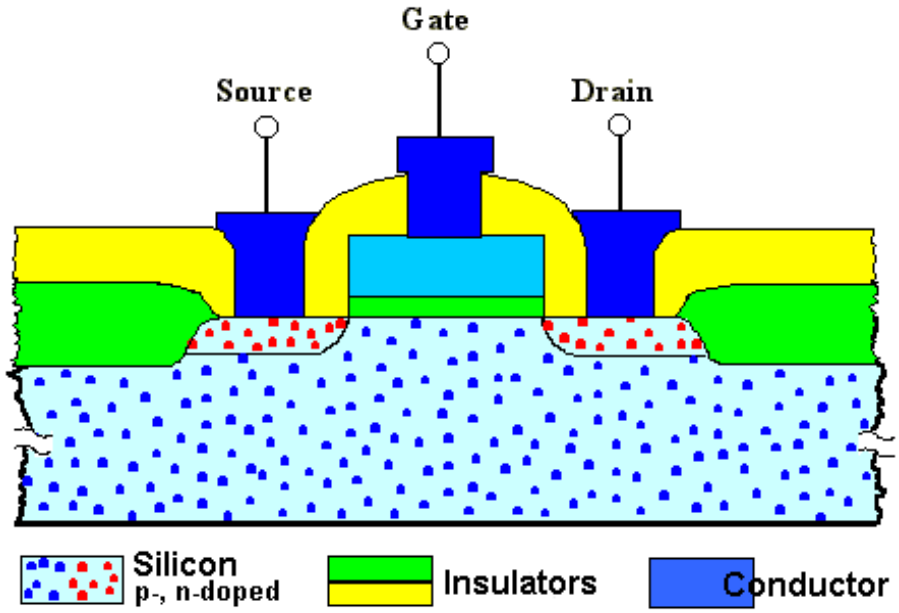


a) Remplir la table de vérité qui donne l'état futur en fonction de l'état présent :

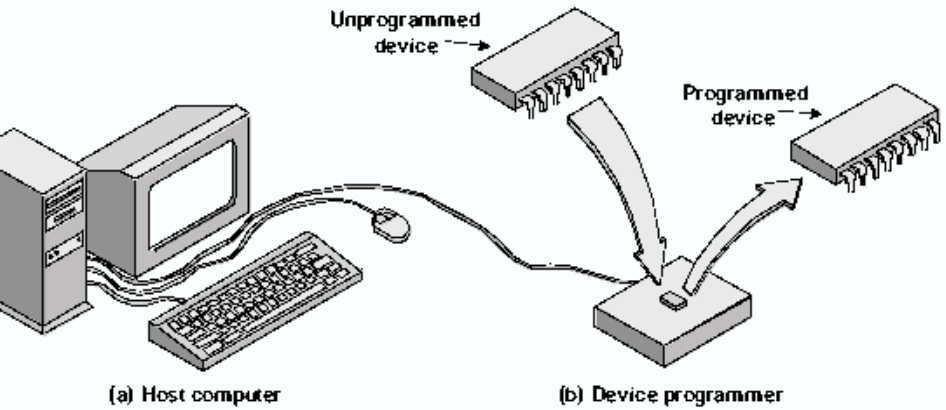
Q_1^t	Q_0^t	Q_1^{t+1}	Q_0^{t+1}
0	0	0	1
0	1	1	1
1	0	0	0
1	1	1	0

b) Donner les expressions booléennes pour Q_1^{t+1} et Q_0^{t+1} en fonction de Q_1^t et Q_0^t (expressions aussi simplifiées que possible en utilisant possiblement la table de Karnaugh):

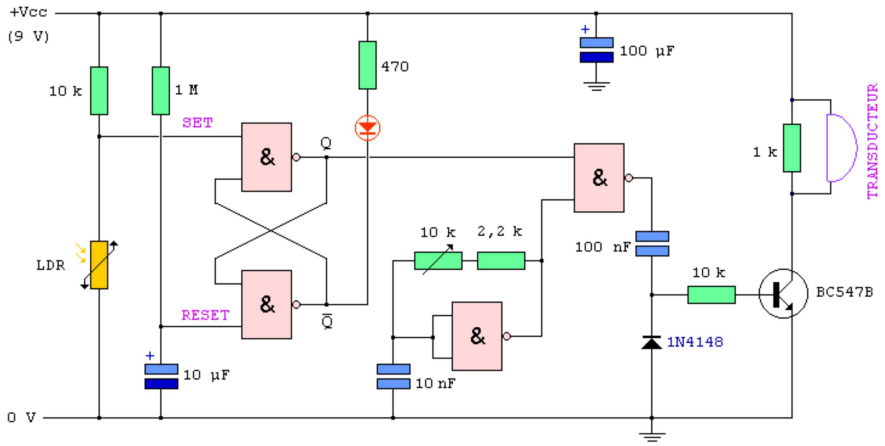
CIRCUITS LOGIQUES
Transistor



CIRCUITS LOGIQUES
Programmation d'une PLA



CIRCUITS LOGIQUES
Détecteur de Lumière



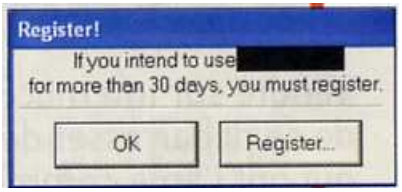
Comment un pirate "cracke" t-il un shareware ?

I. La caisse à outils du pirate

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

II. Premières découvertes

Au lancement du shareware



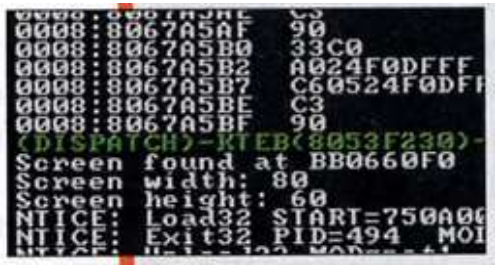
III. Une valeur fantaisiste

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

User32.GetDlgItermTextA

IV. Examen de mémoire

Le debuggeur a détecté la commande:
User32.GetDlgItermTextA et affiche les instructions en cours d'exécution, pas à pas que l'on peut suivre ...



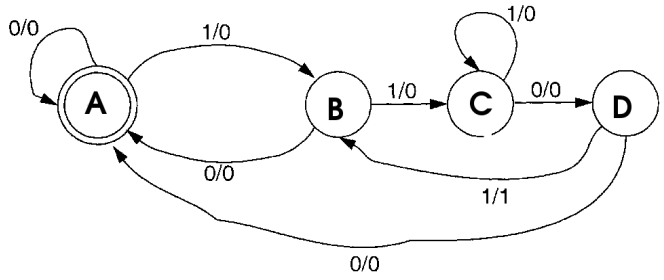
Le nom d'utilisateur est lu, puis stocké dans une zone mémoire. Ensuite c'est le numéro de série qui est enregistré.

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

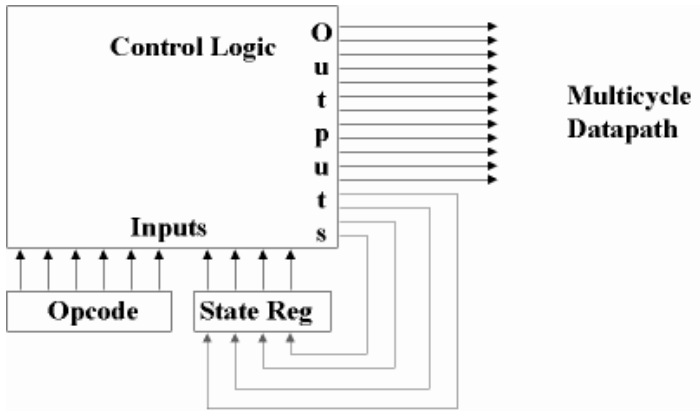
User32.IstrcmpA

...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 !

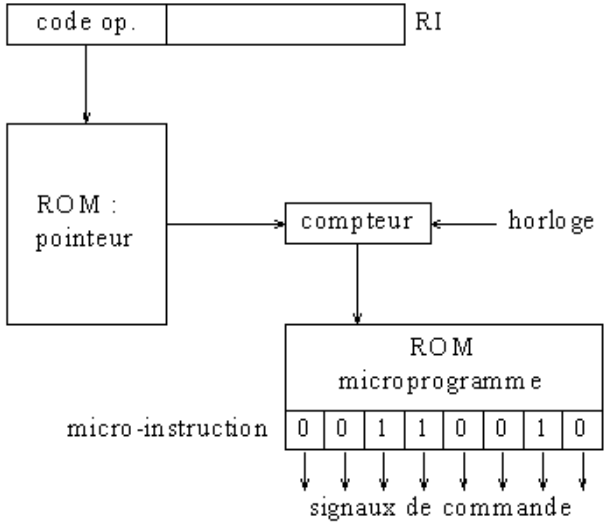
LANGAGE MACHINE
Séquenceur Microprogrammé [1/3]



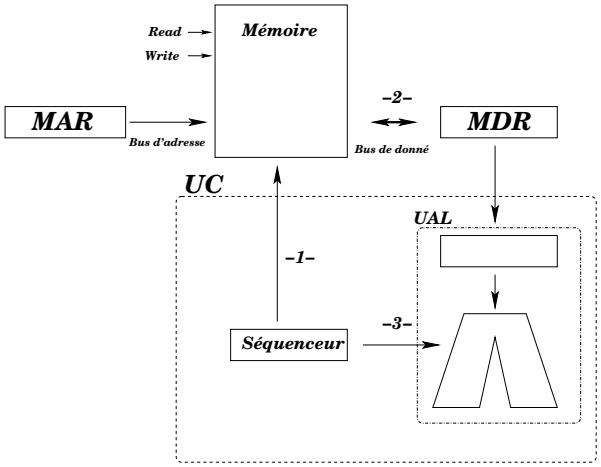
$q_1(t)$	$q_0(t)$	E	$q_1(t+1)$	$q_0(t+1)$	S
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1



LANGAGE MACHINE
Séquenceur Microprogrammé [2/3]



LANGAGE MACHINE
Séquenceur Microprogrammé [3/3]



Read	Write	Read	Write	Read	Write	Read	Write
Memoire		MDR		MAR		ACC	

LANGAGE MACHINE
Exercice

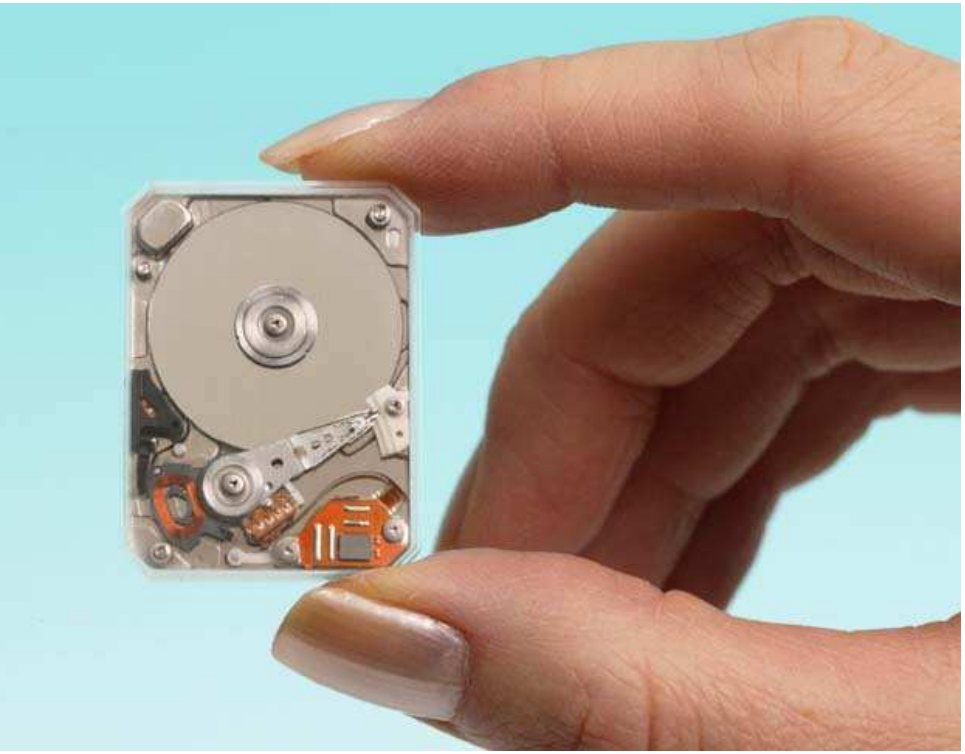
```
          IN
          STO   AD1
          IN
          STO   AD2
          LDA   AD1
LOOP      STA   DTINST

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