TD archi 5/11/20

Q1 : soit EW0 un mot de 16 bits contenant la valeur de 16 capteurs tout ou rien : le volet de la cuisine est ouvert (bit0), la température ext est inférieure à 0 (bit1), la porte du garage est ouverte (bit2),...

en utilisant un masque (lequel ?) comment savoir si la porte de garage est ouverte ?

```
m=4 ou m=0b100 ou m=0x4 ou m=1<<2
# par défaut en décimal, 0b pour binaire, 0x pour hexa, 0o pour octal
```

Quelle que soit l'écriture, il y a exactement la même chose dans la mémoire m de l'ordinateur (et c'est en binaire.
Pour afficher : si c'est un programme utiliser print ! print(m) en décimal, print(bin(m)) en binaire, print(hex(m)) en hexa. Si on est en mode interactif, print n'est pas obligatoire

```
print(m,bin(m),hex(m),oct(m)) #donne 4 0b100 0x4 0o4
```

Q2 : essayer en python (quelqu'un nous montre sur son ordi ?)

```
m=1<<2
ew0=41
bin(ew0) #affiche en binaire : 0b101001
ew0 & m  #affiche 0 car 0b101001 & 0b000100 donne 0 (bit2=0)

ew0=45
bin(ew0) #affiche en binaire : 0b101101
ew0 & m  #affiche 4 car 0b101101 & 0b000100 donne 4 (000100 en b)
```

conclusion : le masquage donne 0 si le bit est 0, autre chose (un 1 entouré de zéros) si le bit est 1

Q3 : comment savoir si le bit N est vrai ou faux (0<=N<=15)

```
N=5
if ew0 & (1<<N) !=0 :
  print("ouvert")
else :
  print("non ouvert")
```

Q4 : soit MW0 un mot de 16 bits contenant par exemple la valeur hexa B9A5. Quelles opérations (masquages et décalages) faire pour que les bits 4 à 7 soient échangés avec les bits 8 à 11 (pour votre exemple, BA95) ?

```
mw0=0xB9A5
v1=mw0 & 0xF00F #garde le 1er et 4è quartet et masque les 2 du milieu
v2=(mw0 & 0x00F0)<<4 #le 2è quartet décallé à gauche
v3=(mw0 & 0x0F00)>>4 #le 3è quartet décallé à droite
v=v1|v2|v3
hex(v)    #donne '0xba95'
```

Q5 : essayer en python

j'essayais d'afficher les masques et valeurs en binaire, mais bin n'affichant pas les 0 devant, rien n'était aligné donc difficile à comprendre. En tout cas, ça y est j'ai trouvé : format(xxx,"016b") affiche en **b**inaire sur 16 bits avec des 0 devant

```
format(mw0,"016b")     #donne '1011100110100101'
format(0xF00F,"016b")  #donne '1111000000001111'
format(v1,"016b")      #donne '1011000000000101'
```

et donc là ça vous aligne bien, et donc on voit bien la valeur initiale, le masque, le résultat avec les 2 quartets extrêmes gardés et les deux du milieu mis à 0

Q6 : un jeu de lumière, composé de 8 lampes, est relié au mot de 8 bits nommé AB2. Ecrire un programme python qui réalise un chenillard : allume la première, puis la seconde, la 3è... (une seule allumée à la fois) et recommence sans arrêt (pour tester on se limitera à 10 tours)

```
ab2=1
for x in range(8):
  format(ab2,"08b")
  ab2=ab2<<1
# -> donne :
# '00000001'
# '00000010'
# '00000100'
# '00001000'
# '00010000'
# '00100000'
# '01000000'
# '10000000'
for x in range(8):
  ab2=1<<x  #1 décallé x fois donc avec x zéros derrière
  format(ab2,"08b")
# -> donne exactement la même chose
```

Q7 : soit AW0 un mot de 16 bits permettant d'allumer (1) ou éteindre (0) 16 actionneurs tout ou rien : monter le volet de la cuisine (bit 0), allumer le chauffage (bit1), allumer le frigo (bit2)...

- comment allumer le frigo sans rien changer aux autres actionneurs ?
- comment l'éteindre ?
- comment l'inverser (s'il est éteint, l'allumer; et réciproquement) ?

Rq : a) masque avec juste le Nième bit à 1 : 1<<N.

```
N=5
m1=1<<N
format(m1,"016b") #donne '0000000000100000'
```

b) masque avec tous les bits à 1 sauf le Nième (~ = complément à 1, - = complément à 2) :

```
m2=~(1<<N)
```

Aïe ! l'afficher (par bin ou format) ne marche pas car négatif ! Par contre, prendre ce masque et l'appliquer à une suite de 1 montrera bien que seul le Nième bit est mis à 0 et pas les autres

```
bin(0xFFFFFFFF & m2) #donne '0b11111111111111111111111111011111'
```

c) masque avec les N derniers bits à 1 et ceux devant à 0 : (1<<N)-1

```
m3=(1<<N)-1
format(m3,"016b") # donne '0000000000011111
```

donc allumer (forcer un 1) c'est faire un OU (|) avec un masque avec un 1 à cet endroit, éteindre (forcer un 0) c'est faire un ET (&) avec que des 1 (tout garder) et un 0 à cet endroit, et inverser c'est appliquer un ou exclusif (^) avec un 1 à cet endroit :

```
aw0=0b1110001110001010    #valeur au pif pour tester
m=1<<2  #le fQ7 :rigo est au bit 2
format(aw0 | m,"016b")      #donne '1110001110001110'
format(aw0 & ~m,"016b")     #donne '1110001110001010'
format(aw0 ^ m,"016b")      #donne '1110001110001110'
```