ISM MPSI/PCSI, Informatique
Année 2023/2024

Corrigé du TP Informatique 14

Exercice 1

1. On saisit :

```
def binaire(n):
    """Conversion dec->bin"""
    res=[]
    a=n
    while a>0:
        res.append(a%2)
        a//=2
    return res
```

On teste :

```
>>> binaire(123)
[1, 1, 0, 1, 1, 1, 1]
>>> bin(123)
'0b1111011'
```

2. On saisit :

```
def decimale(L):
    """Conversion bin->dec selon Horner"""
    n=len(L)
    res=0
    for k in range(n-1,-1,-1):
        res=res*2+L[k]
    return res
```

On teste :

```
>>> decimale(binaire(123))
123
```

3. On saisit :

```
tn=[123,2020,2**20]
for n in tn:
    print(len(binaire(n)),int(np.log2(n))+1)
    print(binaire(n))
    print(binaire(2*n))
    print(binaire(n//2))
```

On observe :

```
7 7
[1, 1, 0, 1, 1, 1, 1]
[0, 1, 1, 0, 1, 1, 1, 1]
[1, 0, 1, 1, 1, 1]
...
```

Exercice 2

1. On saisit :

```
def dec_bin8(n):
    """Conversion dec->bin sur 8 bits"""
    res,a=[],n
    for k in range(8):
        res.append(a%2)
        a//=2
    return res
```

2. On saisit :

```
def bin8_dec(L):
    """Conversion bin sur 8 bits->dec selon schéma de Horner"""
    res=0
    for k in range(7,-1,-1):
        res=res*2+L[k]
    return res
```

3. On saisit :

```
def add8(L1,L2):
    """Addition de deux entiers codés en binaire 8 bits"""
    res=[]
    carry=False                 # carry : booléen de retenue
    for k in range(8):
        if L1[k]!=L2[k]:        # cas 0/1 ou 1/0
            if carry:
                res.append(0)
            else:
                res.append(1)
        else:                   # cas 0/0 ou 1/1
            if carry:
                res.append(1)
            else:
                res.append(0)
            if L1[k]==0:        # cas 0/0 : perte de retenue
                carry=False
            else:               # cas 1/1 : retenue systématique
                carry=True
    return res
```

4. On teste :

```
print("Plage : [ 0,",bin8_dec([1]*8),"]")
for c in [(12,29),(100,70),(200,55),(200,56),(200,200)]:
    a,b=c
    ta,tb=dec_bin8(a),dec_bin8(b)
    print(a+b,bin8_dec(add8(ta,tb)))
```

Exercice 3

1. On saisit :

```
def dec_sign8(n):
    """Conversion dec->bin signé sur 8 bits"""
    res=[]
    a=n
    for k in range(7):
        res.append(a%2)
        a//=2
    res.append(int(a<0))
    return res
```

2. On saisit :

```
def sign8_dec(L):
    """Conversion bin signé sur 8 bits->dec"""
    res=-L[-1]
    for k in range(6,-1,-1):
        res=res*2+L[k]
    return res
```

3. On teste :

```
print("Plage : [",sign8_dec([0]*7+[1]),",",sign8_dec([1]*7+[0]),"]")
for c in [(1,-1),(15,20),(-100,27),(100,27),(100,-27),(-100,-28)]:
    a,b=c
    ta,tb=dec_sign8(a),dec_sign8(b)
    print(a+b,sign8_dec(add8(ta,tb)))
```

On observe :

```
Plage : [ -128 , 127 ]
0 0
35 35
...
```

Ainsi, la fonction `add8` prévue initialement pour l'addition de deux entiers non signés codés sur 8 bits fonctionne parfaitement sur des entiers signés codés sur 8 bits. La compatibilité de l'algorithme d'addition est l'argument décisif pour le choix du codage en complément à 2.