1) Donnez les résultats numériques des évaluations suivantes:

```
unsigned foo(int v){
     unsigned res;
     if (v==0)
          res=32;
     else{
               v = (v^(v-1))>>1;
               for (res = 0; v!=0 ; res++)
                    v = (v >> 1);
     }
     return res;
}
int main(){
     unsigned n1=0xFDB97531, n2=0xECA86420;
     int n3=124,n4=-124;
     float n5=124.25;
     printf("%x\n",n3); //val1
     printf("%x\n",n4);//val2
     printf("%x\n",n1 & n2); //val3
     printf("%x\n",n1 | n2);//val4
     printf("%x\n",n1 ^ n2);//val5
     printf("%x\n",n2 >>4);//val6
     printf("%x\n",(int)n2>>4);//val7
     printf("%x\n",(*(unsigned*)&(n3)));//val8
     printf("%u\n",foo( 124));//val9
     printf("%u\n",foo(-124));//val10
     printf("%u\n",foo(   2));//val11
     printf("%u\n",foo(   4));//val12
     printf("%u\n",foo(   0));//val13
     printf("%u\n",foo(  -1));//val14
}
```

val1=**0x** 0000007C , val2=**0x** FFFFFF84

val3=**0x** ECA86420 , val4=**0x** FDB97531

val5=**0x** 11111111 , val6=**0x** 0ECA8642

val7=**0x** FECA8642 , val8=**0x** 42F88000

val9= 2 , val10= 2

val11= 1 , val12= 2

val13= 32 , val14= 0

Complétez les expressions du code de la fonction foo précéden

```
unsigned foo(int v){
    unsigned res;
    if (v==0)
        res=32;
    else{
            v = (v^(v-1))>>1;
            for (res = 0; v!=0 ; res++)
                v = (v >> 1);
    }
    return res;
}
```

Vos réponse ne contiendront pas d'espace, les noms des reg

```
.data
   nbr : .word -124
.text
   lw --exp1--,--exp2--
   jal foo
   move --exp3--,--exp4--
   li --exp5--,1
   syscall
   li $v0,--exp6--
   syscall
```

```
foo:
# $a0 param - $v0 résultat
   sw $t0,0($sp)
   sw $t1,-4($sp)
   addi $sp,$sp,--exp7--
   bne $a0,--exp8--,fooSuite
   li --exp9--,--exp10--
   b  --exp11--
fooSuite:
   addi $t0,--exp12--,-1
   xor $t0,$a0,--exp13--
   srl $t1,--exp14--,--exp15--
   li $v0,--exp16--
fooWhile:
   beq $t1,$0,fooFin
   srl $t1,--exp17--,--exp18--
   addi $v0,--exp19--,--exp20--
   b fooWhile
fooFin:
   addi $sp,$sp,--exp21--
   sw $t0,0($sp)
   sw $t1,-4($sp)
   jr --exp22--
```

--exp1-- $a0 , --exp2-- nbr

--exp3-- $a0 , --exp4-- $v0

--exp5-- $v0 , --exp6-- 10

--exp7-- -8 , --exp8-- $0

--exp9-- $v0 , --exp10-- 32

--exp11-- fooFin , --exp12-- $a0

--exp13-- $t0 , --exp14-- $t0

--exp15-- 1 , --exp16-- 0

--exp17-- $t1 , --exp18-- 1

--exp19-- $v0 , --exp20-- 1

--exp21-- 8 , --exp22-- $ra

Soit le programme suivant utilisé pour évaluer les performances du cache

```
# initialement $a0=0x10010000
# initialement $a1=16
testCache:
   lw $t0,0($a0) # instruction 1
   add $t1,$t0,-1 # instruction 2
   sw $t1,256($a0) # instruction 3
   add $a0,$a0,4 # instruction 4
   addi $a1,$a1,-1 # instruction 5
   bne $a1,$0,testCache # instruction 6
```

1) Soit un cache contenant 256 octets organisé en bloc de 32 octets avec une fonction de correspondance directe. La taille des adresses est de 32 bits.

- Quelle est la taille (décimale) en bit du déplacement 5 ; de l'index 3 et de l'étiquette 24

A la première itération pour l'accès en lecture :

- Donnez la valeur (hexadécimale) de l'étiquette **0x** 100100

- Donnez la valeur (décimale) de l'index 0

A la première itération pour l'accès en écriture:

- Donnez la valeur (hexadécimale) de l'étiquette **0x** 100101

- Donnez la valeur (décimale) de l'index 0

En déduire le taux de succès global (décimal) pour ce cache 0 /32

2) Soit un cache contenant 1024 octets organisé en bloc de 64 octets avec une fonction de correspondance directe. La taille des adresses est de 32 bits.

- Quelle est la taille (décimale) en bit du déplacement 6 , de l'index 4 et de l'étiquette 22

A la première itération pour l'accès en lecture :

- Donnez la valeur (hexadécimale) de l'étiquette **0x** 40040

- Donnez la valeur (décimale) de l'index 0

A la première itération pour l'accès en écriture:

- Donnez la valeur (hexadécimale) de l'étiquette **0x** 40040

- Donnez la valeur (décimale) de l'index 4

En déduire le taux de succès global (décimal) pour ce cache 30 /32

Soit le programme suivant

```
# initialement $a0=0x10010000
# initialement $a1=16
testCache:
   lw $t0,0($a0) # instruction 1
   add $t1,$t0,-1 # instruction 2
   sw $t1,256($a0) # instruction 3
   add $a0,$a0,4 # instruction 4
   addi $a1,$a1,-1 # instruction 5
   bne $a1,$0,testCache # instruction 6
```

Le programme ci-dessus est représenté par le nombre 123456. Les différents représente le numéros des instruction. (1 représente l'instruction lw $t0... 2 représente l'instruction add $t1...)

Soit un pipeline avec envoi et branchement au plus tôt. A la première itération, la première instruction (n°1) n'a pas de dépendance et commence au cycle numéro 1.

1) Lors de la deuxième itération de la fonction , a quelle cycle (valeur décimale) commence la première instruction (n°1) 10

2) Donner le nombre entier (décimal) représentant l'ordre des instructions du programme réorganisé pour ne pas perdre de cycle en utilisant l'envoi et le branchement retardé d'un cycle

152364

```
# initialement $a0=0x10010000
# initialement $a1=16
testCache:
    lw $t0,0($a0) # instruction 1
    add $t1,$t0,-1 # instruction 2
    sw $t1,256($a0) # instruction 3
    add $a0,$a0,4 # instruction 4
    addi $a1,$a1,-1 # instruction 5
    bne $a1,$0,testCache # instruction 6
```

1) Donnez les valeurs hexadécimales 32 bits représentant l'instruction 2 : **0x** 2109FFFF , l'instruction 4 : **0x** 20840004 et l'instruction 6 **0x** 14A0FFFA

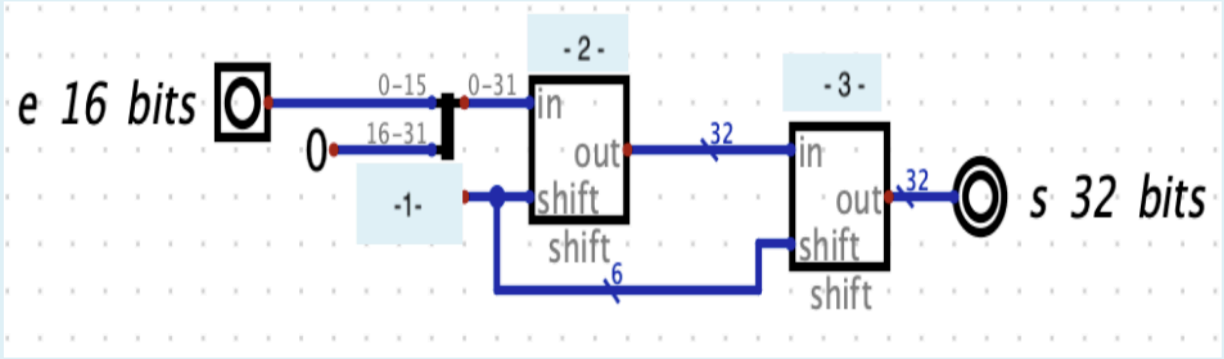
2) Le registre $t0 contient la valeur **0x7FFFFFFF.** Nous lançons le programme suivant

```
addi $t1,$t0,1
addiu $t2,$t0,1
```

Quelle instruction provoque l'arrêt de l'exécution du programme addi .

Donnez la valeur (décimale) de la cause de l'exception 12

Nous souhaitons créer un circuit qui effectue l'extension signée d'une donnée 16 bits en un donnée 32 bits en utilisant le circuit ci-dessous.



Quelle valeur (décimale) devons placer sur l'entrée **-1-** 16

De quel type doit être le circuit de décalage **-2-** logique gauche

De quel type doit être le circuit de décalage **-3-** arithmétique droite

Paragraphe

1) foo : nombre de bit à 0 à droite d'un nombre
2) Cache : Dans le premier cas, il y a concurrence des données lors de l'accès au bloc. Pas dans le deuxième cas.
3) 1
   5 nous devons séparer le lw de son utilisation
   2
   3
   6
   4 position mémoire suivante décalé après le branchement
4) Retour 6 instructions après la suivante.
6) Décalage de 16 a gauche pour mettre le bit de signe dans les poids fort
    Décalage à gauche (logique)
   Décalage arithmétique (conserver le signe)