Dans chacun des exercices qui suivent, on vous donne une fonction en C, qu'il faut transcrire en assembleur, puis en binaire. Certains des exercices sont corrigés. Vous pouvez vous en inspirer pour faire les autres.

En général il faut d'abord commencer par réécrire la fonction en C en utilisant des instructions plus simples, qui se traduisent chacune en une seule instruction en assembleur.
Par exemple les boucles `for` ou `while` sont remplacées par des `goto...;` et des `if(...) goto...;`
De même `s+=*t++;` devient `int x=*t; t++; s+=x;`
tandis que `*t=*u;` devient `int x=*t; *u=x;`
car la valeur de `*t` qui est rangée dans la ram, doit d'abord être mise dans un registre par un `load` (`x=*t`) avant de pouvoir être utilisée comme opérande d'une addition (`s+=x`) ou recopiée par un `store` (`*u=x`) dans une autre variable située dans la ram.
Pire encore `a++;` devient `const int un=1; a+=un;`
tandis que `a*=45;` devient `const int c45=45; a*=c45;`
car une constante comme 1 ou 45 doit d'abord être rangée dans un registre, par un loadimm16, pour être utilisée comme opérande d'une opération comme une addition ou une multiplication.

Exercice 1 : Copie d'un tableau d'entiers

dans un autre tableau de même taille.

```
void coptab(int n, int t[n], int u[n])
{ for(int i=0;i<n;i++) u[i]=t[i]; }
```

On aura une procédure plus courte en assembleur en partant plutôt de

```
void coptab(int n, int *t, int *u)
{ while(n--) *u++=*t++; }
```

où `n` sert de variable de contrôle de la boucle et indique combien d'itérations restent à faire, et `t` et `u` sont des pointeurs qui avancent dans les tableaux. Cela donne:

```
void coptab(int n,int*t,int*u)  coptab:// r0  r1  r2  r3  r4    coptab:
{ int x;                                // n   t   u   x   un
  const int un=1;               loadimm16 r4,1                  26 4 0 1
  debut:                        debut:
  if((n-=un)==0)                sub r0,r4,r0                    2  0 4 0
   goto fin;                    jc fin  // jc $+5               32 0 0 5
  x=*t;                         load r1,r3                      24 1 3 *
  *u=x;                         store r2,r3                     25 2 3 *
  t+=un;                        add r1,r4,r1                    0  1 4 1
  u+=un;                        add r2,r4,r2                    0  2 4 2
  goto debut;                   jmp debut // jmp $-7            44 255 255 249
  fin:                          fin:
  return;                       ret                             29 * * *
}
```

Exercice 2 : valeur absolue

```
int abs(int a) { return a<0? -a :a; }
int abs(int a)      abs:          // r0  r1      abs:
{ int b=0;          xor r1,r1,r1  // a   b       8  1 1 1
  b=0-a;            sub r1,r0,r1                2  1 0 1
  if(0>a) a=b;      movcg r1,r0                 54 1 0 *
  return a;         ret                         29 * * *
}
```

Exercice 3 : Remplacement des entiers dans un tableau par leur valeur absolue

```
void abstab(int n, int t[n])
{ for(int i=0;i<n;i++) t[i]=|t[i]|; }
void abstab(int n, int *t)
{ while(;n--;t++) *t=|*t|; }
void abstab(int n,int*t)  coptab:// r0  r1  r2  r3  r4      abstab:
{ int x;                        // n   t   y   x   un
  const int un=1;         loadimm16 r4,1                    26 4 0 1
  debut:                  debut:
  if(!n--)                sub r0,r4,r0                      2  0 4 0
   goto fin;              jc fin  // jc $+7                 32 0 0 7
  x=*t;                   load r1,r3                        24 1 3 *
  int y=0;                xor r2,r2,r2                      8  2 2 2
  y-=x;                   sub r2,r3,r2                      2  2 3 2
  if(0>x) x=y;            movcg r2,r3                       54 2 3 *
  *t=x;                   store r1,r3                       25 1 3 *
  t++;                    add r1,r4,r1                      0  1 4 1
  goto debut;             jmp debut // jmp $-9              44 255 255 247
  fin:                    fin:
  return;                 ret                               29 * * *
}
```

Exercice 4 : Combinaison linéaire de deux vecteurs d'entiers

```
void cltab(int n, int t[n], int a, int u[n], int b, int v[n])
{ for(int i=0;i<n;i++) v[i]=t[i]*a+u[i]*b|; }
void cltab(int n, int*t, int a, int*u, int b, int*v)
{ while(!n--) *v++=*t++*a+*u++*b; }
void cltab(int n, int*t, int a, int*u, int b, int*v)// x=*t,*t*a  y=*u,*u+b, un=1
           // r0     r1     r2     r3     r4     r5    r6         r7         r8
{                                cltab:                          cltab:
  const int un=1;                loadimm16 r8,1                  26 8 0 1
  debut:                         debut:
  if(!n--)                       sub r0,r8,r0                    2  0 8 0
   goto fin;                     jc fin  // jc $+10             32 0 0 10
  int x=*t;                      load r1,r6                      24 1 6 *
  x*=a;                          mul r6,r2,r6                    20 6 2 6
  int y=*t;                      load r3,r7                      24 3 7 *
  y*=b;                          mul r7,r4,r7                    20 7 4 7
  x+=y;                          add r6,r7,r6                    0  6 7 6
  *v=x;                          store r5,r6                     25 5 6 *
  t++;                           add r1,r8,r1                    0  1 8 1
  u++;                           add r3,r8,r3                    0  3 8 3
  v++;                           add r5,r8,r5                    0  5 8 5
  goto debut;                    jmp debut // jmp $-12           44 255 255 244
  fin:                           fin:
  return;                        ret                             29 * * *
}
```

Exercice 5 : Somme des éléments d'un tableau d'entiers

```
int somtab(int n, int t[n])              int somtab(int n, int*t)
{ int s=0;                               { int s=0;
  for(int i=0;i<n;i++) s+=t[i]; }          while(n--) s+=*t++; }
  return s;                                return s;
}                                        }
void somtab(int n,int*t)       somtab:// r0  r1  r2  r3  r4    somtab:
{ int x;                               // n   t   s   x   un
  const int un=1;              loadimm16 r4,1                  26 4 0 1
  int s=0;                     xor r2,r2,r2                    8  2 2 2
  debut:                       debut:
  if(!n--)                     sub r0,r4,r0                    2  0 4 0
   goto fin;                   jc fin  // jc $+4               32 0 0 4
  x=*t;                        load r1,r3                      24 1 3 *
  s+=x;                        add r2,r3,r2                    0  2 3 2
  t+=un;                       add r1,r4,r1                    0  1 4 1
  goto debut;                  jmp debut // jmp $-6            44 255 255 250
  fin:                         fin:
  return;                      ret                             29 * * *
}
```

Exercice 6 : Somme des valeurs absolues des éléments d'un tableau d'entiers

```
int somabstab(int n, int t[n])
{ int s=0;
  for(int i=0;i<n;i++) s+=|t[i]|; }
  return s;
}
int somabstab(int n, int*t)
{ int s=0;
  while(n--) s+=|*t++|; }
  return s;
}
```

```
void somabstab(int n,int*t)
{ int x,y;
  const int un=1;
  int s=0;
  debut:
  if(!n--)
   goto fin;
  x=*t;
  int y=0;
  y-=x;
  if(0>x) x=y;
  s+=x;
  t++;
  goto debut;
  fin:
  return;
}
```

```
somabstab:// r0  r1  r2  r3  r4  r5
          // n   t   s   x   un  y
loadimm16 r4,1
xor r2,r2,r2
debut:
sub r0,r4,r0
jc fin  // jc $+7
load r1,r3
xor r5,r5,r5
sub r5,r3,r5
movcg r5,r3
add r2,r3,r2
add r1,r4,r1
jmp debut // jmp $-9
fin:
ret
```

```
somabstab:
26 4 0 1
8  2 2 2
2  0 4 0
32 0 0 7
24 1 3 *
8  5 5 5
2  5 3 5
54 5 3 *
0  2 3 2
0  1 4 1
44 255 255 247
29 * * *
```