```
   .p2align 4,,15
   .globl  p1
   .type   p1, @function
p1:
.LFB0:
   .cfi_startproc
   jmp     .L8
   .p2align 4,,10
   .p2align 3
.L11:
   leaq    1(%rax,%rax,2), %rax
.L4:
   movq    %rax, (%rdi,%rsi,8)
.L8:
   testq   %rsi, %rsi
   je      .L10
   subq    $1, %rsi
   movq    (%rdi,%rsi,8), %rax
   testb   $1, %al
   jne     .L11
   movq    %rax, %rdx
   shrq    $63, %rdx
   addq    %rdx, %rax
   sarq    %rax
   jmp     .L4
   .p2align 4,,10
   .p2align 3
.L10:
   rep
   ret
   .cfi_endproc
.LFE0:
   .size   p1, .-p1
```

```
   .p2align 4,,15
   .globl  p2
   .type   p2, @function
p2:
.LFB1:
   .cfi_startproc
   jmp     .L17
   .p2align 4,,10
   .p2align 3
.L15:
   subq    $1, %rdx
   movq    (%rdi,%rdx,8), %rcx
   movq    (%rsi,%rdx,8), %rax
   leaq    (%rcx,%rax), %r8
   subq    %rax, %rcx
   movq    %r8, (%rdi,%rdx,8)
   movq    %rcx, (%rsi,%rdx,8)
.L17:
   testq   %rdx, %rdx
   jne     .L15
   rep
   ret
   .cfi_endproc
.LFE1:
   .size   p2, .-p2
```

```
   .p2align 4,,15
   .globl  f
   .type   f, @function
f:
.LFB2:
   .cfi_startproc
   xorl    %eax, %eax
   testq   %rsi, %rsi
   je      .L19
   subq    $1, %rsi
   .p2align 4,,10
   .p2align 3
.L20:
 movq (%rdi,%rsi,8),%rdx
   subq    $1, %rsi
   leaq    -1(%rdx),%rcx
   imulq   %rcx, %rdx
   addq    %rdx, %rax
   cmpq    $-1, %rsi
   jne     L20
.L19:
   rep
   ret
   .cfi_endproc
.LFE2:
   .size   f, .-f
```

Retrouvez le source C des deux procédures `p1` et `p2` et de la fonction `f`.

Refaites cette question pour `p1` sans les 3 lignes `movq %rax,%rdx  shrq $63,%rdx  addq %rdx,%rax`.

Dans la boucle de `p2`, remplacez la décrémentation de `rdx`, le `testq` et le `jne` par une décrémentation et un `jnc`.

Ecrivez l'équivalent en assembleur de la fonction C ayant pour prototype `long st(long t[],long n);` qui calcule $\sum_{i=0}^{n-1} i \times t[i]$.

Rappels : `imul`, `idiv` et `sar` (Shift Arithmetic Right) sont des opérations signées, tandis que `mul`, `div` et `shr` (SHift Right) sont des opérations non signées.

```
sar : -43>>3 = -6
shr : 42u>>3 = 5u
xor : 45^45 = 0
and : 11&13 = 9
```

```
void p1(long t[],long n)
{ long x; while(n) x=t[--n], t[n]=x&1?3*x+1:x/2; }
void p2(long t[],long u[],long n)
{ long x,y; while(n) x=t[--n], y=u[n], t[n]=x+y, u[n]=x-y; }
long f(long t[],long n)
{ long s=0; while(n--) s+=t[n]*(t[n]-1); return s; }
void p1(long t[],long n)
{ long x; while(n) x=t[--n], t[n]=x&1?3*x+1:x>>1; }
```

```
   .p2align 4,,15
   .globl  p2
   .type   p2, @function
p2:
.LFB1:
   .cfi_startproc
   subq    $1, %rdx             if(n--)
   jc      .L17
   .p2align 4,,10
   .p2align 3
.L15:                           do
   movq    (%rdi,%rdx,8), %rcx   x=t[n],
   movq    (%rsi,%rdx,8), %rax   y=u[n],
   leaq    (%rcx,%rax), %r8      z=x+y,
   subq    %rax, %rcx            x-=y,
   movq    %r8, (%rdi,%rdx,8)    t[n]=z,
   movq    %rcx, (%rsi,%rdx,8)   u[n]=x;
   subq    $1, %rdx             while(n--);
   jnc     .L15
.L17:
   rep
   ret
   .cfi_endproc
.LFE1:
   .size   p2, .-p2
```

```
   .p2align 4,,15
   .globl  st
   .type   st, @function
st:
.LFB4:
   .cfi_startproc
   xorl    %eax, %eax          s=0;
   testq   %rsi, %rsi          if(!n) return;
   je      .L32
   subq    $1, %rsi            n--;
   .p2align 4,,10
   .p2align 3
.L33:                          do
   movq    %rsi, %rdx            x=n,
   imulq   (%rdi,%rsi,8), %rdx   x*=t[n],
   subq    $1, %rsi              n--,
   addq    %rdx, %rax            s+=x;
   cmpq    $-1, %rsi           while(n!=-1);
   jne     .L33
.L32:
   rep
   ret                         return s;
   .cfi_endproc
.LFE4:
   .size   st, .-st
```