```
   .p2align 4,,15
.globl p1
   .type   p1, @function
p1:
.LFB0:
   .cfi_startproc
   subq   $1, %rsi
   testq  %rsi, %rsi
   jle   .L4
   leaq   (%rdi,%rsi,8), %rdx
   xorl   %eax, %eax
   .p2align 4,,10
   .p2align 3
.L3:
   movq   (%rdx), %r8
   movq   (%rdi,%rax,8), %rcx
   subq   $1, %rsi
   movq   %r8, (%rdi,%rax,8)
   addq   $1, %rax
   movq   %rcx, (%rdx)
   subq   $8, %rdx
   cmpq   %rax, %rsi
   jg     .L3
.L4:
   rep
   ret
   .cfi_endproc
.LFE0:
   .size   p1, .-p1
```

```
   .p2align 4,,15
.globl p2
   .type   p2, @function
p2:
.LFB1:
   .cfi_startproc
   testq  %rdx, %rdx
   je     .L10
   subq   $1, %rdx
   leaq   0(,%rdx,8), %rax
   addq   %rax, %rdi
   addq   %rax, %rsi
   jmp   .L9
   .p2align 4,,10
   .p2align 3
.L12:
   subq   $1, %rdx
.L9:
   movq   (%rdi), %rax
   movq   (%rsi), %rcx
   movq   %rcx, (%rdi)
   movq   %rax, (%rsi)
   subq   $8, %rdi
   subq   $8, %rsi
   testq  %rdx, %rdx
   jne    .L12
.L10:
   rep
   ret
   .cfi_endproc
.LFE1:
   .size   p2, .-p2
```

```
   .p2align 4,,15
.globl f
   .type   f, @function
f:
.LFB2:
   .cfi_startproc
   leaq   -1(%rsi), %rdx
   testq  %rdx, %rdx
   movq   (%rdi,%rdx,8), %rax
   je     .L14
   leaq -16(%rdi,%rsi,8), %rcx
   .p2align 4,,10
   .p2align 3
.L15:
   movq   (%rcx), %rsi
   cmpq   %rsi, %rax
   cmovl  %rsi, %rax
   subq   $8, %rcx
   subq   $1, %rdx
   jne   .L15
.L14:
   rep
   ret
   .cfi_endproc
.LFE2:
   .size   f, .-f
```

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

`cmovl` est un `mov` conditionnel, qui s'exécute selon les flags, comme un branchement conditionnel `jl`.

Récrivez `p2`, en remplaçant les deux décrémentations de `rdx`, le `testq` et le `jne` par une seule décrémentation et un `jnc`.

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

```
void p1(long t[],long n)
{ long x,i=0; for(;i<--n;i++) x=t[i], t[i]=t[n], t[n]=x; }
void p2(long t[],long u[],long n)
{ long x; while(n) x=t[--n], t[n]=u[n], u[n]=x; }
long f(long t[],long n)
{ long m=t[--n]; while(n) if(t[--n]>m) m=t[n]; return m; }
```

```
   .p2align 4,,15
.globl p2
   .type   p2, @function
p2:
.LFB1:
   .cfi_startproc
   testq  %rdx, %rdx
   je     .L10
   leaq   0(,%rdx,8), %rax
   addq   %rax, %rdi
   addq   %rax, %rsi
   .p2align 4,,10
   .p2align 3
.L12:
   subq   $8, %rdi
   subq   $8, %rsi
   movq   (%rdi), %rax
   movq   (%rsi), %rcx
   movq   %rcx, (%rdi)
   movq   %rax, (%rsi)
   subq   $1, %rdx
   jne    .L12
.L10:
   rep
   ret
   .cfi_endproc
.LFE1:
   .size   p2, .-p2
```

```
   .p2align 4,,15
.globl somabs
   .type   somabs, @function
somabs:
.LFB2:
   .cfi_startproc
   xorl   %eax, %eax
   subq   $1, %rsi
   jl     .L14
   .p2align 4,,10
   .p2align 3
.L15:
   movq   (%rdi,%rsi,8), %ebx
   testq  %ebx, %ebx
   jge    .L16
   negq   %ebx
.L16:
   addq   %ebx, %eax
   subq   $1, %rsi
   jnc    .L15
.L14:
   rep
   ret
   .cfi_endproc
.LFE2:
   .size   somabs, .-somabs
```