

## Pile et appels de fonction

```
# int g(int i) {  
#     return i&1;  
# }  
# int f(int *t, int n) {  
#     int i;  
#     for (i=0; i<n; i++)  
#         t[i] = g(t[i]);  
# }  
# int t[] = {3,5,2,6};  
# int main(void) {  
#     return f(t, 4);  
# }
```

```
main:    irmovl Stack,%esp  
         irmovl 4,%eax      # empiler le deuxième paramètre 4  
         pushl  %eax  
         irmovl t,%eax     # empiler le premier paramètre t  
         pushl  %eax  
         call   f  
         halt  
  
g:       mrmovl 4(%esp),%eax  
         iandl  1,%eax  
         ret
```

```

# rappel:
# int f(int *t, int n) {
#     int i;
#     for (i=0; i<n; i++)
#         t[i] = g(t[i]);
# }

f:     pushl  %ebp
       rrmovl %esp,%ebp

       irmovl 0,%eax
       pushl  %eax           # i: -4(%ebp)
       mrmovl 8(%ebp),%eax   # p = t: -8(%ebp)
       pushl  %eax

loop:  mrmovl -4(%ebp),%eax  # i
       mrmovl 12(%ebp),%ecx # n
       subl  %eax,%ecx      # n-i <= 0 ?
       jle   fin

       mrmovl -8(%ebp),%ecx # p
       mrmovl (%ecx),%ecx   # *p
       pushl  %ecx         # paramètre pour g
       call  g
       popl   %ecx         # nettoyer le paramètre
       mrmovl -8(%ebp),%ecx # p
       rmmovl %eax,(%ecx)  # *p = résultat de g

       mrmovl -4(%ebp),%ecx # i
       iaddl  1,%ecx
       rmmovl %ecx,-4(%ebp) # i++

       mrmovl -8(%ebp),%ecx # p
       iaddl  4,%ecx
       rmmovl %ecx,-8(%ebp) # p++
       jmp   loop

fin:   rrmovl %ebp,%esp
       popl   %ebp
       ret

       .pos 0x100
t:     .long 3
       .long 5
       .long 2
       .long 6

       .pos 0x200
Stack:

```