```
.text
jal zero, main

#================================
# NO TOCAR HASTA AQUI
#================================

# Limpia los bits impares de a y devuelve accum + limpiar_impares(a) - b
# int32_t restar_limpiando_impares(int32_t accum, int32_t a, int32_t b)

call_restar_limpiando_impares: # a0:accum, a1:a, a2:b
li t0, 0x55555555
and t1, a1, t0
sub t2, t1, a2
add a0, a0, t2
ret


# Limpia los bits pares de a y devuelve accum + limpiar_pares(a) - b
# int32_t restar_limpiando_pares(int32_t accum, int32_t a, int32_t b)

call_restar_limpiando_pares: # a0:accum, a1:a, a2:b
li t0, 0xAAAAAAAA
and t1, a1, t0
sub t2, t1, a2
add a0, a0, t2
ret


# Devuelve 1 si index es par, 0 en caso contario
# int32_t posicion_par(int32_t a, int32_t index, int32_t length)

call_posicion_par: # a0 a, a1 index, a2 length
andi t1, a1, 0x00000001
beqz t1, return_par
li a0, 0x00000000
ret
return_par:
li a0, 0x00000001
ret


# Devuelve 1 si a es negativo, 0 en caso contario
# int32_t numero_negativo(int32_t a, int32_t index, int32_t length)

call_numero_negativo: # a0 a, a1 index, a2 length
bltz a0, return
li a0, 0x0
ret
return:
li a0, 0x1
ret
```

```
#================================
# NO TOCAR DESDE AQUI
#================================

# a0:fn, a1:array, a2: length, a3:expected, a3: fn_name
test_ej:
# prologue
addi sp, sp, -40
sw s9, 36(sp)
sw s8, 32(sp)
sw s7, 28(sp)
sw s6, 24(sp)
sw s5, 20(sp)
sw s4, 16(sp)
sw s3, 12(sp)
sw s2, 8(sp)
sw s1, 4(sp)
sw s0, 4(sp)
add s0, a0, zero #save fn
add s1, a1, zero #save array
add s2, a2, zero #save length
add s3, a3, zero #save expected
add s4, a4, zero #save fn_name
add s6, ra, zero #save return address
add a0, s1, zero
add a1, s2, zero
jalr ra, s0(0)              #call op
lw s7, 0(s3)                #get expected
add s8, a0, zero        #save result
beq s8, s7, no_mismatch_ej
la a0, msg_error_in    #print error message
li a7, ecall_print_string
ecall
add a0, s4, zero
li a7, ecall_print_string
ecall
la a0, msg_error_element
li a7, ecall_print_string
ecall
add a0, s9, zero
li a7, ecall_print_hex
ecall
la a0, msg_error_expecting
li a7, ecall_print_string
ecall
add a0, s7, zero
li a7, ecall_print_hex
ecall
la a0, msg_error_got
li a7, ecall_print_string
ecall
add a0, s8, zero
li a7, ecall_print_hex
ecall
li a0, ascii_new_line
```

```
li a7, ecall_print_char
ecall
no_mismatch_ej:
add ra, s6, zero        #restore ra
lw s9, 36(sp)
lw s8, 32(sp)
lw s7, 28(sp)
lw s6, 24(sp)
lw s5, 20(sp)
lw s4, 16(sp)
lw s3, 12(sp)
lw s2, 8(sp)
lw s1, 4(sp)
lw s0, 0(sp)
addi sp, sp, 40
ret


# a0:fn, a1:array, a2: length, a3:expected, a3: fn_name
test_fn_index:
# prologue
addi sp, sp, -40
sw s9, 36(sp)
sw s8, 32(sp)
sw s7, 28(sp)
sw s6, 24(sp)
sw s5, 20(sp)
sw s4, 16(sp)
sw s3, 12(sp)
sw s2, 8(sp)
sw s1, 4(sp)
sw s0, 4(sp)
add s0, a0, zero        #save fn
add s1, a1, zero        #save array
add s2, a2, zero        #save length
add s3, a3, zero        #save expected
add s4, a4, zero        #save fn_name
add s6, ra, zero        #save return address
add a0, s1, zero
add a1, s2, zero
jalr ra, s0(0)          #call op
lw s7, 0(s3)            #get expected
add s8, a0, zero         #save result
beq s8, s7, no_mismatch_index
la a0, msg_error_in    #print error message
li a7, ecall_print_string
ecall
add a0, s4, zero
li a7, ecall_print_string
ecall
la a0, msg_error_element
li a7, ecall_print_string
ecall
add a0, s9, zero
li a7, ecall_print_hex
```

```
ecall
la a0, msg_error_expecting
li a7, ecall_print_string
ecall
add a0, s7, zero
li a7, ecall_print_hex
ecall
la a0, msg_error_got
li a7, ecall_print_string
ecall
add a0, s8, zero
li a7, ecall_print_hex
ecall
li a0, ascii_new_line
li a7, ecall_print_char
ecall
no_mismatch_index:
add ra, s6, zero          #restore ra
lw s9, 36(sp)
lw s8, 32(sp)
lw s7, 28(sp)
lw s6, 24(sp)
lw s5, 20(sp)
lw s4, 16(sp)
lw s3, 12(sp)
lw s2, 8(sp)
lw s1, 4(sp)
lw s0, 0(sp)
addi sp, sp, 40
ret


# a0:fn, a1:input, a2:expected, a3:length, a4: fn_name
test_fn:
# prologue
addi sp, sp, -40
sw s9, 36(sp)
sw s8, 32(sp)
sw s7, 28(sp)
sw s6, 24(sp)
sw s5, 20(sp)
sw s4, 16(sp)
sw s3, 12(sp)
sw s2, 8(sp)
sw s1, 4(sp)
sw s0, 4(sp)
add s0, a0, zero          #save fn
add s1, a1, zero         #save input
add s2, a2, zero         #save expected
add s3, a3, zero         #save length
srai s3, s3, 2   #length /2
add s4, a4, zero         #save fn_name
slli s5, a3, 2           #length * 4
add s5, s5, s1           #end of input
addi s5, s5, -4          #adjust the last element
```

```
add s6, ra, zero        #save return address
add s9, zero, zero      #current index
loop_check_expected:
add a0, zero, zero      #set accum to zero
lw a1, 0(s1)            #get a
lw a2, 0(s5)            #get b
jalr ra, s0(0)          #call op
lw s7, 0(s2)            #get expected
add s8, a0, zero        #save result
beq s8, s7, no_mismatch
la a0, msg_error_in     #print error message
li a7, ecall_print_string
ecall
add a0, s4, zero
li a7, ecall_print_string
ecall
la a0, msg_error_element
li a7, ecall_print_string
ecall
add a0, s9, zero
li a7, ecall_print_hex
ecall
la a0, msg_error_with
li a7, ecall_print_string
ecall
lw t0, 0(s1)      #get a
add a0, t0, zero
li a7, ecall_print_hex
ecall
la a0, msg_error_and
li a7, ecall_print_string
ecall
lw t1, 0(s5)      #get b
add a0, t1, zero
li a7, ecall_print_hex
ecall
la a0, msg_error_expecting
li a7, ecall_print_string
ecall
add a0, s7, zero
li a7, ecall_print_hex
ecall
la a0, msg_error_got
li a7, ecall_print_string
ecall
add a0, s8, zero
li a7, ecall_print_hex
ecall
li a0, ascii_new_line
li a7, ecall_print_char
ecall
no_mismatch:
addi s1, s1, 4  #move i
addi s5, s5, -4         #move j
addi s2, s2, 4  #move expected
```

```
addi s3, s3, -1  #decrement length
addi s9, s9, 1  #increment index
bnez s3, loop_check_expected
add ra, s6, zero        #restore ra
lw s9, 36(sp)
lw s8, 32(sp)
lw s7, 28(sp)
lw s6, 24(sp)
lw s5, 20(sp)
lw s4, 16(sp)
lw s3, 12(sp)
lw s2, 8(sp)
lw s1, 4(sp)
lw s0, 0(sp)
addi sp, sp, 40
ret

main:

# Corriendo test para restar_limpiando_impares
la a0, call_restar_limpiando_impares
la a1, input_array
la a2, array_expected_restar_limpiando_impares
la a3, array_length
la a4, msg_name_restar_limpiando_impares
jal ra, test_fn
# Corriendo test para restar_limpiando_pares
la a0, call_restar_limpiando_pares
la a1, input_array
la a2, array_expected_restar_limpiando_pares
la a3, array_length
la a4, msg_name_restar_limpiando_pares
jal ra, test_fn
# Corriendo test para posicion_par
la a0, call_posicion_par
la a1, input_array
la a2, array_expected_posicion_par
la a3, array_length
la a4, msg_name_posicion_par
jal ra, test_fn
# Corriendo test para numero_negativo
la a0, call_numero_negativo
la a1, input_array
la a2, array_expected_numero_negativo
la a3, array_length
la a4, msg_name_numero_negativo
jal ra, test_fn


add a0, zero, zero
li a7, ecall_exit
ecall

.data
.equ array_length, 18
```

```
target_array: .word 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
input_array: .word 0x00000000 0x11111111 0x00008000 0x17777777 0x00000003 0x00000005 0x00000007 0x80000000 0x0000000d 0xffff0000 0x00000001 0x12222222 0x13333333 0x00000002 0x1bbbbbbb 0x8ddddddd 0x0000000b 0xffffffff
input_array_small: .word 0x00000000 0x0000011 0x0000080 0x00000027 0x00000003 0x00000005 0x00000007 0x00000020 0x0000000d 0x0000002f 0x00000001 0x00000022 0x00000033 0x00000002 0x000000bb 0x0000002d 0x0000000b 0x00000045
msg_name_restar_limpiando_impares: .string "restar_limpiando_impares"
array_expected_restar_limpiando_impares: .word 0x00000001 0x11111106 0x72222223 0xf999999a
msg_name_restar_limpiando_pares: .string "restar_limpiando_pares"
array_expected_restar_limpiando_pares: .word 0x00000001 0xfffffff5 0x7222a223 0xe6666667
msg_name_posicion_par: .string "posicion_par"
array_expected_posicion_par: .word 0x00000001 0x00000000 0x00000001 0x00000000
msg_name_numero_negativo: .string "numero_negativo"
array_expected_numero_negativo: .word 0x00000000 0x00000000 0x00000000 0x00000000


msg_error_in: .string "Error en "
msg_error_element: .string " en el elemento "
msg_error_with: .string " con "
msg_error_and: .string " y "
msg_error_expecting:.string " se esperaba "
msg_error_got: .string " se recibio "

.equ ecall_print_string, 4
.equ ecall_print_char, 11
.equ ecall_print_hex, 34
.equ ecall_exit, 93
.equ ascii_new_line, 10
.equ ascii_space, 32
```