```
struct ogg_stream {
   0 uint8_t *buf;                  → PADDING de 24 bits
  32 uint32_t duration;
  64 uint32_t serial;               → PADDING de 32 bits
 128 uint64_t granule;
 192 uint64_t start_granule;
 256 int32_t header;
 288 int32_t nsegs;
 320 uint8_t segments[27];          → PADDING de 8 bits.
 544 int32_t incomplete;
 576 int32_t page_end;
 608 int32_t start_trimming;
 640 int32_t end_trimming;
 672 uint8_t *new_metadata;         → PADDING de 24 bits.
 704 int32_t new_metadata_size;
 736 void *private;
 744 const struct ogg_codec *codec;
};   PADDING de 16 bits
```

→ size(struct ogg_stream) = 96 bytes !

```
struct ogg_state {
   0 uint64_t pos;
  64 int32_t curidx;
  96 struct ogg_state *next;        → PADDING de 24 bits
 128 int32_t nstreams;
 160 struct ogg_stream streams[1];
}   PADDING de 16 bits.
```

→ size(struct ogg_state) = 24 bytes

```
struct ogg {
   0 struct ogg_stream *streams;    → PADDING de 24 bits
  32 int32_t nstreams;
  64 int32_t headers;
  96 int32_t curidx;
 128 int64_t page_pos;
 192 struct ogg_state *state;
     PADDING de 56 bits
}
```

→ size(struct ogg) = 32 bytes.

```
int ogg_find_stream(struct ogg *ogg, int32_t serial)
{
    for (int i = 0; i < ogg->nstreams; i++){
        if (ogg->streams[i].serial == serial)
            return i;
    }
    return -1;
}
```

```
ogg_find_stream:
        movl    $-1, %eax
        movl    $0, %edx        //i=0
        jmp     .cmp

.cmp:
        movl    32(%rdi), %ecx
        cmpl    %ecx, %edx
        jl      .loop           //salta si i < ogg->nstreams
        ret                     <- return -1

.loop:
        leaq    (%rdi, %edx, 768), %r8      //ogg->streams[i]
        movq    64(%r8), %r9
        movsql  (%r9), %r9d
        movsql  (%rsi), %esi        } 64 a 32 bits

        cmpl    %r9d, %esi
        je      .equals
        addl    $1, %edx        //i++
        jmp     .cmp

.equals:
        movl    %edx, %eax
        ret
```

```
struct ogg_state *ogg_find_state(struct ogg *ogg)
{
    struct ogg_state *state = ogg->state;
    while(state != NULL && state->curidx != ogg->curidx){
        state = state->next;
    }
    return (state!=NULL && state->curidx == ogg->curidx)? state : NULL;
}
```

rdi

```
ogg_find_state:
        movq    192(%rdi), %rsi         //rsi -> *state.
        movq    $0, %rax
        jmp     .cmp

.cmp:
        cmpq    $0, %rsi        } state != NULL
        je      .is_null
        leaq    96(%rdi), %rdx  }
        cmpq    64(%rsi), %rdx  } state->curidx != ogg->curidx
        je      .is_equal       64(%rsi)        96(%rdi)
        leaq    96(%rsi), %rcx  }
        movq    %rcx, %rsi      } state = state->next.
        jmp     .cmp

.is_null:
        ret     //%rax seteado en NULL.

.is_equal:
        movq    %rsi, %rax      //devuelvo *state.
        ret
```

CARRERA: Ing. en Informática Plan (1986)

UNIVERSIDAD DE BUENOS AIRES
Facultad de INGENIERIA

CODIGO DE REGISTRO DEL ALUMNO

18 41741488
AÑO

Dirección de Títulos y Planes Registro de Alumnos

Lucas Bilé
FIRMA DEL ALUMNO

FIRMA DEL EMPLEADO

NOTA: Esta libreta constituye un documento interno de la Universidad que acredita al alumno como tal ante sus distintos organismos.

103252

UNIVERSIDAD DE BUENOS AIRES

Apellido BILO

Nombre Lucas

Nacido el 06 de MAYO de 1999

en Prov. de Bs. As.

Título secundario Bachiller

Doc. Nac. Id. 41741488

Cédula de Identidad

Pasaporte Nº

Domicilio S. de Bustamante 1173 7° A

Localidad CABA

Tel.:

Nota: Anotaciones válidas únicamente para uso interno de la Universidad