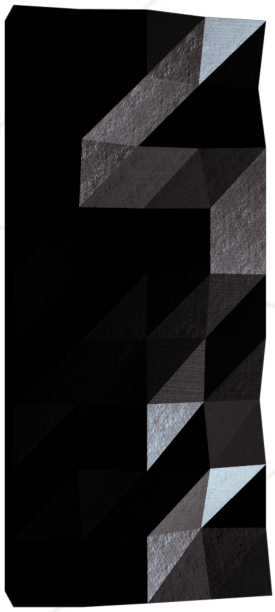

Libasm

Assembly yourself!

Resumen: El objetivo de este proyecto es familiarizarse con el lenguaje ensamblador.

Índice general

Capítulo I

Introducción

El lenguaje ensamblador, a menudo llamado asm, es un lenguaje de bajo nivel para ordenadores - u otros dispositivos programables - en el cual existe una relación muy fuerte entre el lenguaje y la arquitectura del dispositivo.
Cada ordenador dispone de un asm específico. Por el contrario, los lenguajes de alto nivel suelen ser multiplataforma pero necesitan ser compilados o interpretados. Al lenguaje ensamblador también se le puede llamar lenguaje máquina simbólico.

Capítulo II

Instrucciones generales

- Sus funciones no pueden pararse de forma inesperada (segmentation fault, bus error, double free, etc.) salvo en el caso de comportamientos indefinidos. Si esto ocurre, se considerará que su proyecto no es válido y tendrá un `0` en la evaluación.
- Su `Makefile` debe incluir al menos las reglas `$(NAME)`, `all`, `clean`, `fclean` y `re`. Su Makefile solo debe compilar/enlazar los archivos necesarios.
- Para entregar ejercicios extras en su proyecto tendrá que incluir una regla `bonus` en su Makefile que añadirá los distintos headers, bibliotecas o funciones que estén prohibidas en la parte principal del proyecto. Los ejercicios extras deberán estar en un archivo `_bonus.{c/h}`. Las evaluaciones de la parte obligatoria y de la parte extra se hacen por separado.
- Le recomendamos que cree programas de prueba para sus proyectos, aunque ese trabajo **no será ni entregado ni evaluado**. Esto le dará la oportunidad de probar fácilmente su trabajo al igual que el de sus compañeros.
- Entregue su trabajo en el repositorio git que se le ha asignado. Solo se evaluará el trabajo entregado a través del git. Si Deepthought le tiene que evaluar, lo hará después de las evaluaciones de sus compañeros. Si surge un error durante la evaluación de Deepthought, esta última se parará.
- Tiene que escribir en asm 64 bits. Cuidado con las “calling convention”, es decir el protocolo de llamadas entre funciones.
- Tiene que entregar archivos ’.s’ y no asm inline
- Debe compilar su código con nasm
- Debe utilizar la sintaxis Intel y no la de AT&T.

Capítulo III

Parte obligatoria

- Su biblioteca se debe llamar libasm.a
- Tiene que entregar un main que probará sus funciones y compilará con su biblioteca para mostrar que funciona.
- Tiene que escribir las funciones siguientes:
  - ft_strlen (man 3 strlen)
  - ft_strcpy (man 3 strcpy)
  - ft_strcmp (man 3 strcmp)
  - ft_write (man 2 write)
  - ft_read (man 2 read)
  - ft_strdup (man 3 strdup, se autoriza malloc)
- Debe verificar los errores durante las syscalls et reenviarlas correctamente.
- Vuestro código debe permitir la lectura de la variable errno (de <errno.h>) desde un fichero .c
- Para esto, está autorizado `extern ___error`

Capítulo IV

Parte extra

Puede volver a escribir las funciones siguientes en asm: La lista enlazada utilizará la estructura siguiente:

```
typedef struct                 s_list
{
  void                        *data;
  struct s_list       *next;
}                              t_list;
```

- ft_atoi_base (Como la de la piscina)
- ft_list_push_front (Como la de la piscina)
- ft_list_size (Como la de la piscina)
- ft_list_sort (Como la de la piscina)
- ft_list_remove_if (Como la de la piscina)