

Laboratorio Sesión 04: Ensamblador: subrutinas y estructuras

Objetivo

El objetivo de esta sesión es introducir la programación en ensamblador de subrutinas y estructuras en la arquitectura x86 bajo Linux. En concreto, se trabajarán aspectos como la programación de llamadas a subrutinas, la codificación de subrutinas completas y el acceso a estructuras de datos. Además se repasan conceptos como la programación de estructuras de control y el acceso a vectores y/o matrices.

Conocimientos Previos

Para realizar esta práctica deberíais repasar las traducciones directas de C a ensamblador del x86 de la codificación de subrutinas. Además deberíais repasar la forma en que se almacenan y acceden las estructuras.

Los “Makefile”

Entre los ficheros de esta práctica encontraréis un elemento nuevo: un fichero llamado `Makefile`. Este fichero auxiliar sirve para compilar códigos cuando hay muchos ficheros de entrada distintos sin tener que teclear todos los nombres cada vez. Para utilizarlo basta con que tecleéis `make` y él mismo generará el resultado. Si abríis el fichero con un editor de textos veréis que su sintaxis es muy sencilla. En esta práctica podéis ejecutar el `make` con tres parámetros diferentes (uno por apartado): `test0`, `test1` y `test2`.

Estudio Previo

1. Explicad qué significan y para qué sirven los parámetros “argc” “argv” del `main`.
2. Explicad para qué sirve la función `atoi`
3. Dibujad el struct `S1`:

```
typedef struct {  
    char c;  
    int k;  
    int *m;  
} S1;
```

4. Dibujad el bloque de activación de las rutinas `Buscar` y `BuscarElemento`.
5. Dibujad el bloque de activación de la rutina `main`.
6. Dibujad el bloque de activación y traducid el siguiente código a ensamblador del x86:

```
int SencillaSub(S1 a, char b) {  
    int i;  
  
    if (a.c==b)  
        i=0;  
    else  
        i=*(a.m);  
    return i+a.k;  
}
```

7. Dibujad el bloque de activación y traducid a ensamblador del x86 la siguiente subrutina:

```
int LlamaSencilla(S1 x, char y) {  
    *(x.m)=SencillaSub(x,x.c);  
    return x.k;  
}
```

Trabajo a realizar durante la Práctica

1. Compilad (`make test0`) y probad la aplicación original escrita en C. Anotad algunos resultados originales para poder comprobar que vuestros códigos funcionan bien.
2. Traducid a ensamblador del x86 la rutina `BuscarElemento`. Utilizad como base el fichero `BuscaElemento.s` y comprobad el resultado con ayuda de los ficheros `Main.c` y `Buscar.c` (podéis usar `make test1` para compilar). Entregad en el Racó de la asignatura el fichero `BuscaElemento.s`.
3. Traducid a ensamblador del x86 la rutina `Buscar`. Utilizad como base el fichero `Busca.s`. Probadla con el fichero `Main.c` y con la rutina en ensamblador del apartado anterior (`make test2`). Entregad en el Racó de la asignatura el fichero `Busca.s`.
4. Recordad entregar en el Racó de la asignatura los ficheros `BuscaElemento.s` y `Busca.s`. Debéis entregar sólo los dos ficheros fuentes, sin comprimir ni cambiarles el nombre, y sólo una versión por pareja de laboratorio (es indistinto que miembro de la pareja entregue).

Grupo: _____

Hoja de respuesta al Estudio Previo

1. Explicad qué significan y para qué sirven los parámetros “argc” y “argv” del main.

2. Explicad para qué sirve la función atoi

3. Dibujad el struct S1:

[illegible]

4. Dibujad el bloque de activación de las rutinas Buscar y BuscarElemento.

	0	1	2
0			
1			
2			

5. Dibujad el bloque de activación de la rutina `main`.

--	--

6. El bloque de activación y el código de `SencillaSub` en ensamblador del x86 es:

--	--

7. El bloque de activación y el código de LlamaSencilla en ensamblador del x86 es:

--	--