

**Ejemplo de un programa en lenguaje ensamblador sobre un procesador ARM en Raspberry Pi 3 en un entorno operativo Linux Raspbian, que muestra una cadena de caracteres en pantalla, que fueron almacenados en el área de data del proceso.**

```
.data
    cadena : .ascii " Hola Mundo Cruel 2020 2021 2022 2023 ..... ! \n"

.text
.global main

main : push { r7, lr }      /* apilamos reg. en el stack */
      mov r0, #1          /* 1 codigo de la salida estándar archivo pantalla*/
      ldr r1, = cadena    /* direccion de memoria de la cadena a enviar */
      mov r2, #49         /* longitud */
      mov r7, #4          /* 4 es el codigo de la llamada al sistema write*/
      swi #0              /* se realiza la llamada al sistema write*/
      mov r0, #0          /* devolvemos ok */
      pop { r7, lr }      /* desapilamos reg. del stack */
      bx lr              /* salimos de main */
```

/\*

La instrucción que ejecuta la llamada al sistema es swi #0, siempre tendrá cero como valor inmediato. El código numérico de la llamada y el número de parámetros podemos buscarlo en cualquier manual de Linux, buscando "Linux system call table". En nuestro caso la llamada write se corresponde con el código 4 y acepta tres parámetros: manejador del archivo, dirección de los datos a escribir (nuestra cadena) y longitud de los datos. En nuestro ejemplo, el manejador del archivo es el 1, que está conectado con la salida estándar o lo que es lo mismo, con la pantalla.

\*/

/\*

Las interrupciones software son subrutinas que se incluyen en el sistema operativo y que son llamadas desde el programa ejecutando la instrucción swi #n. Se denominan así porque el mecanismo de funcionamiento es el mismo que en las de hardware.

\*/

**Ejemplo de un programa en lenguaje ensamblador sobre un procesador ARM en Raspberry Pi 3, en un entorno operativo Linux Raspbian, que permite ingresar una cadena de caracteres por teclado y luego muestra la cadena de caracteres en pantalla.**

```
.data
    cadena: .ascii ".....\n"
```

```
.text
```

```
.global main
```

```
main:
```

```

MOV R7, #3 /* llamada al sistema read */
MOV R0, #0 /* stdin teclado */
MOV R2, #7 /* cantidad de char a leer */
LDR R1, =cadena /* buffer de entrada */
SWI 0

```

```

MOV R7, #4 @ llamada al sistema write
MOV R0, #1 @ stdout monitor
MOV R2, #19 @ cantidad de char a escribir
LDR R1, =cadena @ buffer de salida
SWI 0

```

```

MOV R7, #1 /* llamada al sistema exit */
SWI 0

```

**Ejemplo de un programa en lenguaje ensamblador sobre un procesador INTEL i3 usando el debug nativo en un entorno operativo Windows XP, que permite ingresar una cadena de caracteres por teclado hasta ingresar el signo \$.**

```

-a 0100
0CA1:0100 mov si,2000
0CA1:0103 mov ah,01
0CA1:0105 int 21
0CA1:0107 mov [si],al
0CA1:0109 inc si
0CA1:010A cmp al,24
0CA1:010C jnz 0103
0CA1:010E int 20
0CA1:0110
-a
123456$
El programa ha terminado de forma normal
-d 2000
0CA1:2000 31 32 33 34 35 36 24 72-61 20 6F 20 65 73 74 61 123456$ra o esta
0CA1:2010 62 6C 65 63 65 20 6C 61-20 68 6F 72 61 20 64 65 blece la hora de
0CA1:2020 6C 20 73 69 73 74 65 6D-61 2E 00 0A 0D 0A 54 49 l sistema....Il
0CA1:2030 4D 45 20 5B 68 6F 72 61-50 00 0A 0D 0A 86 45 73 ME [hora]....Es
0CA1:2040 63 72 69 62 61 20 54 49-4D 45 20 73 69 6E 20 6E criba TIME sin n
0CA1:2050 69 6E 67 A3 6E 20 70 61-72 A0 6D 65 74 72 6F 20 ing.n par,metro
0CA1:2060 70 61 72 61 20 76 65 72-20 6C 61 20 68 6F 72 61 para ver la hora
0CA1:2070 20 61 63 74 75 61 6C 20-79 20 70 6F 64 65 72 20 actual y poder

```

**Ejemplo de un programa en lenguaje ensamblador sobre un procesador INTEL i3 usando el debug nativo en un entorno operativo Windows XP, que permite mostrar una cadena de caracteres por pantalla desde el desplazamiento de memoria 2000 hasta encontrar el código ascii del signo \$.**

```

-a 0120
0CA1:0120 mov si,2000
0CA1:0123 mov dl,[si]
0CA1:0125 mov ah,02
0CA1:0127 int 21
0CA1:0129 inc si
0CA1:012A cmp dl,24
0CA1:012D jnz 0123
0CA1:012F int 20
0CA1:0131
-r ip
IP 0120
.:
-g
123456$
El programa ha terminado de forma normal

```

**Ejemplo de un programa en lenguaje ensamblador sobre un procesador INTEL i3 usando el debug nativo en un entorno operativo Windows XP, que permite mostrar una cadena de caracteres por pantalla desde el desplazamiento de memoria 2000 usando la función 09 de la INT 21.**

```
-a 0140
0CA1:0140 mov ah,09
0CA1:0142 mov dx,2000
0CA1:0145 int 21
0CA1:0147 int 20
0CA1:0149
-
-e 2000
0CA1:2000 0A.31 38.32 4D.33 75.34 65.24
-
-r ip
IP 0100
:0140
-
-g
1234
El programa ha terminado de forma normal
-
```