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:
```

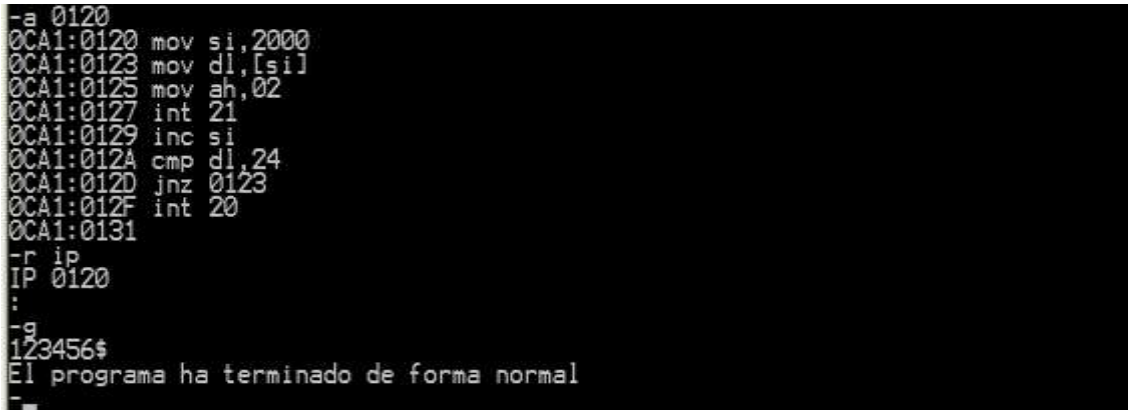
```
MOV R7, #3   /* llamada al sistema read */
MOV R0, #0         /* stdin teclado */
MOV R2, #7         /* cantidad de char a leer */
LDR R1, =cadena  /* buffer de entrada */
SWI 0

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 $.

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 $.



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.