

## Tarea 1 | Entorno de desarrollo y subrutinas

### Objetivos

- Instalar y familiarizarse con las primeras herramientas de trabajo, simulador y *debugger*, que serán luego utilizadas para los laboratorios.
- Escribir una subrutina sencilla en lenguaje *assembler*.

### Requisitos previos

- Instalar las herramientas (z80-tools) disponibles en la página del curso.
- Seguir la “*Guía para las herramientas del laboratorio*” en forma completa. En el capítulo 2 usar la alternativa “2.1. Utilizando un simulador del procesador z80...”.

### Descripción General

Se desea implementar una subrutina *calc* que realice operaciones con el contenido de la memoria ROM y con las constantes  $\alpha$  y  $\beta$  de 8 bits, determinadas a partir del número de la cédula de cada estudiante como se describe a continuación. Si los dígitos de la cédula de identidad son  $d_6$  .  $d_5d_4d_3$  .  $d_2d_1d_0$  (NO se considera el dígito verificador), entonces,  $\alpha$  es el número obtenido haciendo el OR entre la constante 0x80 y el número 0xd1d0 (la notación 0x indica números hexadecimales (base 16)) y  $\beta$  es 0xd4d3 .

Por ejemplo: para la cédula de identidad 1.234.567 los valores de las constantes son:

- $\alpha = 0xd1d0 \text{ OR } 0x80 \Rightarrow \alpha = 0x67 \text{ OR } 0x80 \Rightarrow \alpha = 0xE7$  (1110 0111b)
- $\beta = 0xd4d3 = 0x34$  (0011 0100b)

La subrutina *calc* a realizar depende del último dígito ( $d_0$ ) de la cédula de cada estudiante (anterior al dígito verificador), según se describe en la siguiente tabla:

Dígito $d_0$ de la CI	0 o 1	2, 3 o 4	5, 6 o 7	8 o 9
Número de versión	1	2	3	4

En el caso ejemplo el último dígito es 7, por lo que la versión que corresponde es la 3.

El estudiante deberá:

- Comprender cómo es invocada la subrutina por el código suministrado en el archivo *plantilla.s*.
- Diseñar la solución al problema planteado. Se sugiere escribir un diagrama de flujo o pseudocódigo<sup>1</sup> de la solución antes de comenzar a escribir las instrucciones en *assembler*.
- Renombrar el archivo de la plantilla como "numero\_de\_cedula.s" (p. ej.: 1234567.s) y escribir el código *assembler* de la subrutina en el lugar indicado en la plantilla.

<sup>1</sup> Descripción informal de un algoritmo utilizando estructuras de control similares a las de un lenguaje de alto nivel, como “if”, “while” o “for”.

- Obtener una compilación libre de errores.
- Probar el funcionamiento correcto del programa en el simulador, comparando los resultados obtenidos con los esperados. Para eso deberán utilizarse los comandos adecuados del debugger para correr paso a paso, detener la ejecución en un "breakpoint" y consultar el contenido de registros y lugares de memoria.

La entrega consistirá solamente en el archivo con el programa fuente. Se verificará que el programa compile sin errores y que el resultado del cálculo sea el correcto.

En lo que sigue se utiliza la notación  $M[\text{dir}]$  para referirse al byte almacenado en la dirección  $\text{dir}$  de la memoria del sistema. Observar además que el valor máximo de  $\text{dir}$ , que es  $\beta + \alpha - 1$ , **puede no ser representable en 8 bits** por lo que es necesario utilizar registros de 16 bits para almacenarlo.

## Versión 1

---

La subrutina *calc* deberá ejecutar la operación **ADD** de las constantes  $\alpha$  y  $\beta$ , y guardar el resultado en el lugar 0xB400 de memoria. Luego deberá contar cuántos enteros **pares** (bit menos significativo es 0) hay en la region de memoria ( $M[\beta]$ ,  $M[\beta + 1]$ , ...,  $M[\beta + \alpha - 1]$ ), y guardar el resultado en la dirección 0xB401 de memoria. Finalmente, deberá retornar correctamente al programa principal.

## Versión 2

---

La subrutina *calc* deberá ejecutar la operación **ADD** de las constantes  $\alpha$  y  $\beta$ , y guardar el resultado en el lugar 0xB400 de memoria. Luego deberá contar cuántos enteros que sean **mayores a 127** hay en la region de memoria ( $M[\beta]$ ,  $M[\beta + 1]$ , ...,  $M[\beta + \alpha - 1]$ ), y guardar el resultado en la dirección 0xB401 de memoria. Finalmente, deberá retornar correctamente al programa principal.

## Versión 3

---

La subrutina *calc* deberá ejecutar la operación **SUB** de las constantes  $\alpha$  y  $\beta$ , y guardar el resultado en el lugar 0xB400 de memoria. Luego deberá contar cuántos enteros **impares** (bit menos significativo es 1) hay en la region de memoria ( $M[\beta]$ ,  $M[\beta + 1]$ , ...,  $M[\beta + \alpha - 1]$ ), y guardar el resultado en la dirección 0xB401 de memoria. Finalmente, deberá retornar correctamente al programa principal.

## Versión 4

---

La subrutina *calc* deberá ejecutar la operación **SUB** de las constantes  $\alpha$  y  $\beta$ , y guardar el resultado en el lugar 0xB400 de memoria. Luego deberá contar cuántos enteros que sean **mayores a 127** hay en la region de memoria ( $M[\beta]$ ,  $M[\beta + 1]$ , ...,  $M[\beta + \alpha - 1]$ ), y guardar el resultado en la dirección 0xB401 de memoria. Finalmente, deberá retornar correctamente al programa principal.

## Entrega

La entrega consistirá en el archivo de texto con el programa escrito por el estudiante. El archivo a entregar debe tener como nombre el número de C.I. (sin puntos ni guiones y SIN

el dígito verificador) y con extensión “.s”. Por ejemplo si el nro. de cédula es 1.234.567-8 el archivo debe llamarse 1234567.s. El archivo debe subirse a través de la tarea que será creada en la página del curso en la plataforma EVA de facultad. La entrega **vence el día domingo 24 de marzo a las 23:59hs.**