ARQUITECTURA DE COMPUTADORAS
TRABAJO PRÁCTICO N° 3
LENGUAJE ENSAMLADOR

1. Mediante el software de simulación http://schweigi.github.io/assembler-simulator/
   a) Programa que que sume dos datos
   b) Implementar un programa que realice la suma y la resta con dos datos almacenado en memoria.
   c) Escribir un programa que compare dos números. Si son iguales el programa debe finalizar y si son distintos los debe sumar.
   d) Un programa que lea un dato identifique si es par o impar.
   e) Programa que indique el funcionamiento de el stack. (pila)

```
; a) ejemplo simple
; suma


        MOV A, 0x04
        MOV B, 0x0A
        ADD A,B
        MOV [16],B
        MOV [17],A
        HLT             ; fin de ejecución

; b) ejemplo simple
; suma


        MOV [32], 0x04
        MOV [33], 0x0A
        MOV B,[32]
        MOV A,[33]
        ADD A,B
        HLT             ; fin de ejecución

; c) ejemplo simple
; comparación


        MOV A, 0x0A
        MOV B, 0x0A
        CMP A,B
        JE      FIN
        ADD A,B
        MOV [32],A
        HLT             ; fin de ejecucuión
FIN:
        HLT             ; fin de ejecución

; d) ejemplo simple
; número par - bucle infinito - impar finaliza

INICIO:
        MOV [32], 0x04
        MOV [33], 0x0B
        MOV B,[32]
        MOV A,[33]
        AND A,0x01
        JZ INICIO
        HLT             ; fin de ejecución
```

```
; e) ejemplo simple
; puntero de pila


        MOV A,0x06
LOOP:   DEC A
        PUSH A
        CMP A,0x00
        JNZ LOOP
        HLT


; e) ejemplo simple
; carga y descarga del stack

        MOV A,0x06
LOOP:   DEC A
        PUSH A
        CMP A,0x00
        JNZ LOOP
LOOP1:  INC A
        POP A
        CMP A,0x06
        JNZ LOOP1
        HLT


; f) Ejemplo Salida de caracteres


                MOV D, 0xE8     ; Point to output
                MOV A, 0x40
SALIDA:         MOV [D],A
                INC D
                INC A
                JMP SALIDA
                HLT
```

2. Mediante el software de simulación http://schweigi.github.io/assembler-simulator/

a) Cargar números en las direcciones 60,61,62 y 63.
Restarle una constante (por ejemplo, el hexadecimal 7)
Transferir el resultado a las direcciones 70,71, 72 y 73

b) Cargar N números (POR EJEMPLO 16) a partir de la dirección 60.
Terminar el ingreso de números, si ingresa un dato igual a cero.

c) Cargar N números a partir de la dirección 60.
Restarle una constante (por ejemplo, el hexadecimal 5)
Terminar el ingreso de números, si el resultado de la resta es CERO.

d) Cargar la línea de memoria RAM desde la memoria 40 a la 4F con 16 DATOS y transferirlos a partir de la dirección de memoria 60

e) Ejemplo de Hello World en español. Cambiar la salida por: Hola Mundo. ¿Qué tal?

Explicar y/o comentar el programa en español

```
; ejemplo simple
; escribe Hola Mundo en la salida

        JMP start
hello:  DB "Hello World¡" ; Variable
        DB 0                ; String terminator

start:
        MOV C, hello    ; Point to var
        MOV D, 232      ; Point to output
        CALL print
        HLT               ; Stop execution

print:                  ; print(C:*from, D:*to)
        PUSH A
        PUSH B
        MOV B, 0
.loop:
        MOV A, [C]      ; Get char from var
        MOV [D], A      ; Write to output
        INC C
        INC D
        CMP B, [C]      ; Check if end
        JNZ .loop       ; jump if not

        POP B           ; ¿?
        POP A           ; ¿?
        RET
```

1)

```
        MOV [60], 0x0A
        MOV [61], 0x09

        MOV A, [60]
        SUB A, 0x07
        MOV [70], A
        MOV A, [61]
        SUB A, 0x07
        MOV [71], A
        HLT
```

2)

```
        MOV C,0x10
        MOV D,0x60
LOOP:   CMP B,C
        JNZ LOOP1
        HLT
LOOP1:  MOV [D],C
        INC D
        DEC C
        JMP LOOP
        HLT
```

3)

```
        MOV C,0x10
        MOV D,0x60
LOOP:   CMP B,C
        JNZ LOOP1
        HLT
LOOP1:  MOV [D],C
        INC D
        DEC C
        MOV A,C
        SUB A,0x05
        JZ FIN
        JMP LOOP
FIN:    HLT
```

4)

```
        JMP INICIO
NUM:    DB "ABCDEFGHIJKL00¡" ; Variable
        DB 0                    ;  terminator

INICIO:
        MOV C, NUM     ; puntero a la variable
        MOV D, 0x40    ; puntero a dirección

.loop:
        MOV A, [C]      ;
        MOV [D], A      ;
        INC C
        INC D
        CMP B, [C]      ;
        JNZ .loop       ;
        MOV C,0x40
        MOV D,0x60
TRAN:   MOV A, [C]
        MOV [D],A
        INC C
        INC D
        CMP A,B
        JNZ TRAN:
```

HLT