

PrácticaFinalCode2

▼ Class	Fundamentos de Computadores
▼ Type	Programación
🔗 Materials	https://github.com/Jloen1999/PracticaFinalCode2/blob/main/README.md
☑ Reviewed	<input type="checkbox"/>
👤 Property	
📅 Date	@April 24, 2022
☰ Property 1	Jose Luis Obiang Ela Nanguan


PRIMERA PRÁCTICA FINAL DE CODE2

1.ENUNCIADO

Se trata de realizar, en CODE 2, un proceso que realice un pequeño trabajo de "criptografía".

Lo primero que hará nuestro programa es visualizar por el Puerto de Salida OP02 el valor "H'AAAA" para indicarnos que va a comenzar el proceso de entrada de valores:

A continuación, nuestro programa va a ir visualizando por el Puerto de Salida OP01 las posiciones de memoria (H'A000, H'A001, H'A002, ..., H'A009) donde vamos a ir introduciendo las "traducciones alfabéticas" correspondientes a la clave numérica introducida. Así, visualizaremos por el Puerto de Salida OP01 la posición H'A000 para que se introduzca, por el Puerto de Entrada IP01, el valor A; se visualizará, por el Puerto de Salida OP01, la posición H'A001 para que se introduzca, por el Puerto de Entrada IP01, el valor B, ... La correspondencia se denota a continuación:

<u>Aa</u> POSICIONES	 CONTENIDOS
<u>H'A000</u>	A
<u>H'A001</u>	B
<u>H'A002</u>	C
<u>H'A003</u>	D
<u>H'A004</u>	E
<u>H'A005</u>	F
<u>H'A006</u>	A
<u>H'A007</u>	B
<u>H'A008</u>	C
<u>H'A009</u>	D

2.CÓDIGO DEL PROGRAMA

```

;COMIENZO DEL PROGRAMA

ORG H'0000 ;Posición inicial del programa

;DEFINICIÓN DE VARIABLES/CONSTANTES
VAL_AA EQU H'AA
VAL_A0 EQU H'A0
VAL_BB EQU H'BB
VAL_0A EQU H'0A
VAL_UNO EQU H'01
VAL_CERO EQU H'00
VAL_NUEVE EQU H'09
PUERTO1 EQU H'01
PUERTO2 EQU H'02

;ASIGNACIÓN DE VARIABLES A UTILIZAR

;EN EL PROGRAMA A REGISTROS
REGISTRO_OP2 EQU r5 ;Registro que visualiza el valor del puerto OP02
REGISTRO_SEIS EQU r6
REGCONTENT_B EQU rB ;Registro que almacenará el el valor H'BBBB
REGCONTENT_A EQU rA ;Registro que almacenará el el valor H'AAAA
CERO EQU r0
INDICE EQU r1
MAX EQU r2
REGISTRO_KEY EQU R3 ;Registro que representa la clave de entrada
VAL_KEY EQU r4 ;Registro que representa el valor correspondiente a la clave de entrada

```

```

REGISTRO_D EQU rD ;Registro de Dirección de Memoria
REGCONTENT EQU r7 ;Registro que contiene el valor de cada posición de memoria
RESTO EQU r8      ;Registro que contiene el valor del resto de las sustracciones
CONTEO EQU r9

                                ;INICIALIZACIÓN DE VARIABLES/CONSTANTES
LLI REGISTRO_OP2, H'AA ;;Inicializar la variable REGISTRO_OP2(r5). Cargamos la parte baja
H'AA
LHI REGISTRO_OP2, H'AA ;cargamos la parte alta H'AA
LLI REGCONTENT_B, VAL_BB ;;Inicializar la variable REGCONTENT_B(rB). Cargamos la parte ba
ja H'BB
LHI REGCONTENT_B, VAL_BB ;cargamos la parte alta H'BB
LLI INDICE, VAL_UNO ;;Inicializar la variable INDICE a VAL_UNO(H'01)
LLI REGCONTENT_A, VAL_CERO ;inicialización del índice REGCONTENT_A(rA). Cargamos la parte
baja H'00
LHI REGCONTENT_A, VAL_A0 ;cargamos la parte alta H'A0
LLI REGISTRO_D, VAL_CERO
LLI MAX, VAL_0A ;Tamaño de tabla: MAX=H'A00A. Cargamos la parte baja H'0A
LHI MAX, VAL_A0 ;cargamos la parte alta H'A0

                                ;CÓDIGO DEL PROGRAMA

OUT OP02, REGISTRO_OP2 ;Instrucción de salida del contenido del registro REGISTRO_OP2(rC)
a través del puerto OP02, es decir, visualizamos por el puerto de salida OP02 el valor
H'AAAA

                                ;**** PROCEDIMIENTOS ****
Main: ;Programa Principal
      LLI REGISTRO_D, lo(DirMemoria)
      LHI REGISTRO_D, hi(DirMemoria)
      BR ;Salto a la función DirMemoria

IncrementComp: ;Función que incrementa el valor de REGCONTENT_A y comprueba que no supere
el tamaño máximo
ADDS REGCONTENT_A, REGCONTENT_A, INDICE ;Incrementamos el valor del Registro REGCONTENT_
A
SUBS RESTO, MAX, REGCONTENT_A ;Controlamos que no superamos el límite
#BZ Continuar ;Si BZ=1 saltamos a la función Continuar
#BR DirMemoria ;SiNo saltamos a la función DirMemoria

DirMemoria:
OUT PUERTO1, REGCONTENT_A ;Instrucción de salida del contenido de REGCONTENT_A(rA) a trav
és del puerto OP01, es decir, ;muestra el valor actual del REGCONTENT_A por el
puerto OP01,
ADDS REGISTRO_D, REGCONTENT_A, CERO ;Pasamos la dirección de REGCONTENT_A actual en memor
ia
IN REGCONTENT, PUERTO1 ;cargamos el valor que introduzamos por el puerto IP01 a un nuevo
registro.
ST [VAL_CERO], REGCONTENT ;almacenamos el valor de REGCONTENT(r7) en memoria

```

```

#BR IncrementComp    ;Volvemos a la función IncrementComp

Continuar:
ADDS REGISTRO_D, REGCONTENT_A, CERO    ;Pasamos el índice al registro de dirección
OUT PUERTO1, REGCONTENT_B    ;Visualizamos por el puerto OP01 el valor del registro REGCONTE
NT_B

;Inicializamos nuevamente los registros a utilizar en esta segunda parte del programa
LLI rA, H'00
LLI rB, H'00
LLI rC, H'00
LLI rD, H'00
LLI r0, H'00
LLI r1, H'00
LLI r2, H'00

;-----
-----

;una vez que tengo los valores en memoria visualizo H'BBBB por OP01 para indicar que ya pu
edo
;introducir la clave a traducir

IN r0, H'01
;Metemos la clave en 3 registros diferentes
LLI r3, H'00
ADDS r1, r0, r3    ;--->1234
ADDS r2, r0, r3    ;--->1234
ADDS r3, r0, r3    ;--->1234

;Ahora hacemos los desplazamientos necesarios para quedarnos con el siguiente resultado
;---> 0001
;---> 0002
;--> 0003
;--> 0004

;InicializarVariables1:
    LLI r4, H'01
    LLI r5, H'00    ;Almacena el total de veces que se repite la ejecución de la función
    LLI r6, H'00    ;Almacena El total de desplazamientos
    LLI r7, H'0C    ;Almacena el resultado de restar el total de veces de ejecución de
la función y el total de desplazamientos.

MoveFirstNumber:    ;desplazamos 12 veces a la derecha para quedar 0001
    SHR r0
    ADDS r5, r5, r4
    LLI rD, LO (MoveFirstNumber)
    LHI rD, HI (MoveFirstNumber)
    SUBS r6, r5, r7
    BS    ;Si el total de veces que se ejecuta la función es menor al total de
;desplazamientos se repite el bucle

;InicializarVariables2:
    LLI r4, H'01

```

```

        LLI r5, H'00 ;Almacena el total de veces que se repite la ejecución de la función
        LLI r6, H'00 ;Almacena El total de desplazamientos
        LLI r7, H'04 ;Almacena el resultado de restar el total de veces de ejecución de
la función y el total de desplazamientos.

MoveSecondNumber1: ;Desplazamos 4 veces a la izq para quedar 2340
        SHL r1
        ADDS r5, r5, r4
        LLI rD, LO (MoveSecondNumber1)
        LHI rD, HI (MoveSecondNumber1)
        SUBS r6, r5, r7
        BS ;Si el total de veces que se ejecuta la función es menor al total de
;desplazamientos se repite el bucle
;InicializarVariables3:
        LLI r4, H'01
        LLI r5, H'00 ;Almacena el total de veces que se repite la ejecución de la función
        LLI r6, H'00 ;Almacena El total de desplazamientos
        LLI r7, H'0C ;Almacena el resultado de restar el total de veces de ejecución de
la función y el total de desplazamientos.

MoveSecondNumber2: ;Desplazamos 12 veces a la derecha para quedar 0002
        SHR r1
        ADDS r5, r5, r4
        LLI rD, LO (MoveSecondNumber2)
        LHI rD, HI (MoveSecondNumber2)
        SUBS r6, r5, r7
        BS ;Si el total de veces que se ejecuta la función es menor al total de
;desplazamientos se repite el bucle
;InicializarVariables4:
        LLI r4, H'01
        LLI r5, H'00 ;Almacena el total de veces que se repite la ejecución de la función
        LLI r6, H'00 ;Almacena El total de desplazamientos
        LLI r7, H'08 ;Almacena el resultado de restar el total de veces de ejecución de
la función y el total de desplazamientos.

MoveThirdNumber1: ;Desplazamos 8 veces a la izq para quedar 3400
        SHL r2
        ADDS r5, r5, r4
        LLI rD, LO (MoveThirdNumber1)
        LHI rD, HI (MoveThirdNumber1)
        SUBS r6, r5, r7
        BS ;Si el total de veces que se ejecuta la función es menor al total de
;desplazamientos se repite el bucle

;InicializarVariables5:
        LLI r4, H'01
        LLI r5, H'00 ;Almacena el total de veces que se repite la ejecución de la función
        LLI r6, H'00 ;Almacena El total de desplazamientos
        LLI r7, H'0C ;Almacena el resultado de restar el total de veces de ejecución de
la función y el total de desplazamientos.

MoveThirdNumber2: ;Desplazamos 12 veces a la derecha para quedar 0003
        SHR r2
        ADDS r5, r5, r4

```

```

        LLI rD, LO (MoveThirdNumber2)
        LHI rD, HI (MoveThirdNumber2)
        SUBS r6, r5, r7
        BS ;Si el total de veces que se ejecuta la función es menor al total de
;desplazamientos se repite el bucle

;InicializarVariables6:
        LLI r4, H'01
        LLI r5, H'00 ;Almacena el total de veces que se repite la ejecución de la función
        LLI r6, H'00 ;Almacena El total de desplazamientos
        LLI r7, H'0C ;Almacena el resultado de restar el total de veces de ejecución de
la función y el total de desplazamientos.

MoveFourNumber1: ;Desplazamos 12 veces a la izq para quedar 4000
        SHL r3
        ADDS r5, r5, r4
        LLI rD, LO (MoveFourNumber1)
        LHI rD, HI (MoveFourNumber1)
        SUBS r6, r5, r7
        BS ;Si el total de veces que se ejecuta la función es menor al total de
;desplazamientos se repite el bucle

;InicializarVariables7:
        LLI r4, H'01
        LLI r5, H'00 ;Almacena el total de veces que se repite la ejecución de la función
        LLI r6, H'00 ;Almacena El total de desplazamientos
        LLI r7, H'0C ;Almacena el resultado de restar el total de veces de ejecución de
la función y el total de desplazamientos.

MoveFourNumber2: ;Desplazamos 12 veces a la derecha para quedar 0004
        SHR r3
        ADDS r5, r5, r4
        LLI rD, LO (MoveFourNumber2)
        LHI rD, HI (MoveFourNumber2)
        SUBS r6, r5, r7
        BS ;Si el total de veces que se ejecuta la función es menor al total de
;desplazamientos se repite el bucle

;Procedemos a la traducción de la clave
;Inicializar_RD1:
        LLI rD, H'00
        LHI rD, H'A0
        ADDS rD, rD, r0
        LD r4, [rD+H'00]

;Inicializar_RD2:
        LLI rD, H'00
        LHI rD, H'A0
        ADDS rD, rD, r1
        LD r5, [rD+H'00]

;Inicializar_RD3:
        LLI rD, H'00

```

```

        LHI rD, H'A0
        ADDS rD, rD, r2
        LD r6, [rD+H'00]

;Inicializar_RD4:
        LLI rD, H'00
        LHI rD, H'A0
        ADDS rD, rD, r3
        LD r7, [rD+H'00]

;Metemos en un registro la traducción de la clave final
;la clave 1234 nos debe devolver BCDE
;VAL_KEY: ABCD
        LLI r8, H'00
        LHI r8, H'00

; Añadimos y desplazamos de maner sucesiva la traducción final de la clave

        ADDS r8, r8, r4
        LLI rD, L0 (Desplazar)
        LHI rD, HI (Desplazar)
        CALLR
        ADDS r8, r8, r5
        LLI rD, L0 (Desplazar)
        LHI rD, HI (Desplazar)
        CALLR
        ADDS r8, r8, r6
        LLI rD, L0 (Desplazar)
        LHI rD, HI (Desplazar)
        CALLR
        ADDS r8, r8, r7

;-----
;-----
;-----

;Finalmente guardamos el valor resultante en la direccion de memoria H'A00A

        LLI rD, H'0A
        LHI rD, H'A0

        ST [rD+H'00], r8

;-----
;-----
;-----

        HALT

Desplazar:
        LLI r0, H'01

```

```

;uso el registro r5 para almacenar el conteo
LLI r1, H'00
;uso el registro r6 para almacenar el resultado de SUBS
LLI r2, H'00
;uso el registro r7 para almacenar el numero de veces que hago la instruccion
;en este caso, 12
LLI r3, H'04
bucle:
    SHL r8
    ADDS r1, r1, r0
    LLI rD, L0 (bucle)
    LHI rD, HI (bucle)
    SUBS r2, r1, r3
    BS
RET
END

```

3. Memoria

__Para realizar esta practica, he dividido la tarea a realizar 6 partes:

1. Almacenamiento de valores en memoria introducidos por el puerto IP01 por cada posición visualizada en OP01
2. Lectura de la cifra, guardamos los valores introducidos por el puerto IP01 en su correspondiente posición en memoria
3. Desplazamiento de cada una de las cifras de la clave intorducida por el puerto IP01
4. Accedemos primero en las posiciones en memoria pero para ello primero descomponemos las cifras.

Aqui es donde viene lo “complicado” puesto que para posteriormente acceder a las poisciones de memoria tengo que descomponer el numero en cifras, de modo que si el numero es “1234” las cifras seran 1, 2, 3 y 4.

Práctica CODE 2. Curso 2021/ 2022 3

Para esto copio el numero original almacenado en un registro en otros 3 registros y mediante operaciones de desplazamiento en cada uno de ellos, voy haciendo que quede la cifra en la posicion del bit menos significativo y el resto ceros. Esto es: 0001, 0002, 0003 y 0004.

5. Traducción de las cifras

Para traducir las cifras a su clave, simplemente accedo a la posición de memoria determinada por $rD +$ la cifra (que está almacenada en un registro) y guardo el contenido de dicha posición en registros. Si el número era 1234, el contenido de los registros será 000B, 000C, 000D y 000E.

6. Composición de la clave

Una vez tengo las cifras en clave tengo que meter la cifra más significativa en un registro, desplazar a la izquierda 4 veces y sumar la siguiente cifra más significativa. Repeto este proceso 3 veces y por último añado la última cifra (la menos significativa) a dicho registro.

Para esta instrucción repetitiva he creado una subrutina a la cual llamo cada vez que hay que hacer el desplazamiento a la izquierda.

7. Almacenamiento de la clave en memoria

Una vez tengo la clave en el registro, la almaceno en la posición de memoria $H'A00A$. Esto lo hago a través de la instrucción $ST[rD+H'00], rX$ dando valor $H'A0$ a la parte alta de rD y valor $H'0A$ a la parte baja.____

4. Bibliografía

He recurrido a la documentación del campus y a videos de Youtube para solventar las dudas que me surgían a medida que avanzaba en el proyecto.

5. Observación

Me ha resultado bastante entretenido pero al mismo tiempo un lío de cabeza ya que el cerebro humano no está programado para pensar en lenguaje máquina.

He aprendido mucho sobre el funcionamiento interno de las máquinas, los programas operando en a nivel bajo.