Informe de la práctica 7

Introducción

En esta 7ª práctica de Introducción a los Ordenadores encontraremos los objetivos, los ejercicios planteados tanto guiados como autónomos y las conclusiones:

- Programar en el ensamblador del i8085 varias aplicaciones, demostrando los conocimientos adquiridos en teoría.

PARTE GUIADA

1. Introducción de datos por consola y muestra de los datos por pantalla

Analiza, con ayuda del profesor el siguiente código. Cárgalo en el simulador y ejecutalo paso a paso. Fíjate cuál es la primera dirección de memoria de texto del i8085. ¿Cuál es la función del par de registros BC en este código?

El par de registros BC sirven para guardar la dirección de memoria E000h, que es la dirección que apunta a la memoria de texto del microprocesador.

Esta dirección sirve para que, todo lo que se guarde a partir de dicha dirección, aparezca por la pantalla de texto en formato ASCII.

Código 1

```
; Ejemplo de programa

; Simulador de consola

; Asociado a la interrupción TRAP


.org 100h

pila:                          ; Posición Pila
```

```
.org 200h                ; Programa Principal
        lxi H, pila      ; Puntero de pila apuntando a 100h
        sphl
        mvi B, E0h       ; Par BC apuntando
        mvi C, 00h       ; a la memoria de texto
bucle:
        jmp bucle        ; Loop infinito

.org 0024h               ; Dirección de interrupción TRAP
        call string_in   ; LImada a subrutina de introducción
        ret              ; de datos por consola

.org 300h                ; Rutina captura y muestra
string_in:
        in 00h           ; Puerto de entrada
        cpi 00h          ; Si no hay carácter introducido, sale
        jz no_tecla      ; Si hay, escríbelo por pantalla

tecla:
        stax B
        inx B

no_tecla:
        ret
```

2. Detección de carácteres no deseados

Modifica el código anterior tal que sólo se puedan introducir por teclado valores numáricos del 0 al 9 y los operadores +, - , & , |. Ejecútalo para ver como funciona.

Código 2

```
.define                          ; Número de carácteres permitidos
        allowed_count 15
.data 00h                                   ; Carácteres Permitidos:
        allowed: db 30h,31h,32h,33h,34h,    ; Números de 0 a 9
                    35h,36h,37h,38h,39h,    ; +, -, &, |
                    2Bh,2Dh,26h,7Ch,3Dh
.org 100h
        pila:                   ; Posición Pila

.org 200h                       ; Programa Principal
        lxi H, pila             ; Puntero de pila apuntando a 100h
        sphl
        mvi B, E0h              ; Par BC apuntando
        mvi C, 00h              ; a la memoria de texto
bucle:
        jmp bucle               ; Loop infinito

.org 0024h                      ; Dirección de interrupción TRAP
        call string_in          ; Llamada a subrutina de introducción
        ret                     ; de datos por consola

.org 300h                       ; Rutina captura y muestra
string_in:
```

```
        in 00h                  ; Puerto de entrada
        cpi 00h                 ; Si no había carácter introduido, sale
        jz no_tecla             ; Si habia, comprueba que está
                                ; permitido
tecla:
        call check_allowed      ; Carácter Permitido? Si no 00h
        cpi 00h                 ; Escribidlo
        jz no_tecla
        stax B
        inx B
no_tecla:
        ret

check_allowed:                  ; Subrutina control carácteres
        push D
        push H
        mvi E, allowed_count
        lxi H, allowed

allowed_loop:                   ; Comprueba si el carácter está en
        mov D,M                 ; la lista de carácteres permitidos
        cmp D
        jz is_allowed
        inx H
        dcr E
        jnz allowed_loop
        jmp not_allowed
```

```
is_allowed:                    ; Está Permitido: No modificar
        mov A,D
        jmp end_allowed

not_allowed:                   ; No Permitido: Poner a 00h
        mvi A,00h

end_allowed:
        pop H
        pop D
        ret
```

* Nota: la definición de los carácteres permitidos tiene que realizarse en una única línia de código para que al ensamblar el código no dé error. Aquí se ha escrito en más de una línia por questión de espacio.

PARTE NO GUIDA

1. Suma de dos valores introducidos por consola

Diseñad una subrutina que a partir de dos números (base 10) introducidos por el teclado del simulador i8085 haga la suma y presente el resultado en la pantalla de texto del i8085. Haced servir direccionamiento directo e indirecto e indicad en el código donde tenemos estos direccionamientos.

A modo de leyenda para todos los códigos del informe :

- El direccionamiento directo se verá en el código subrayado en azul claro (por ejemplo, **in 00h**).
- El direccionamiento indirecto se verá subrayado en amarillo (por ejemplo, **call string_in**).
- Los comentarios en **verde** significan que el código no se ha modificado desde el programa anterior, es decir, hacen lo mismo siempre y se pueden obviar .
- Los comentarios en **morado** significan que son partes nuevas de código o con modificaciones respecto al programa anterior.

✓ **Tarea 1.** Subir el código:

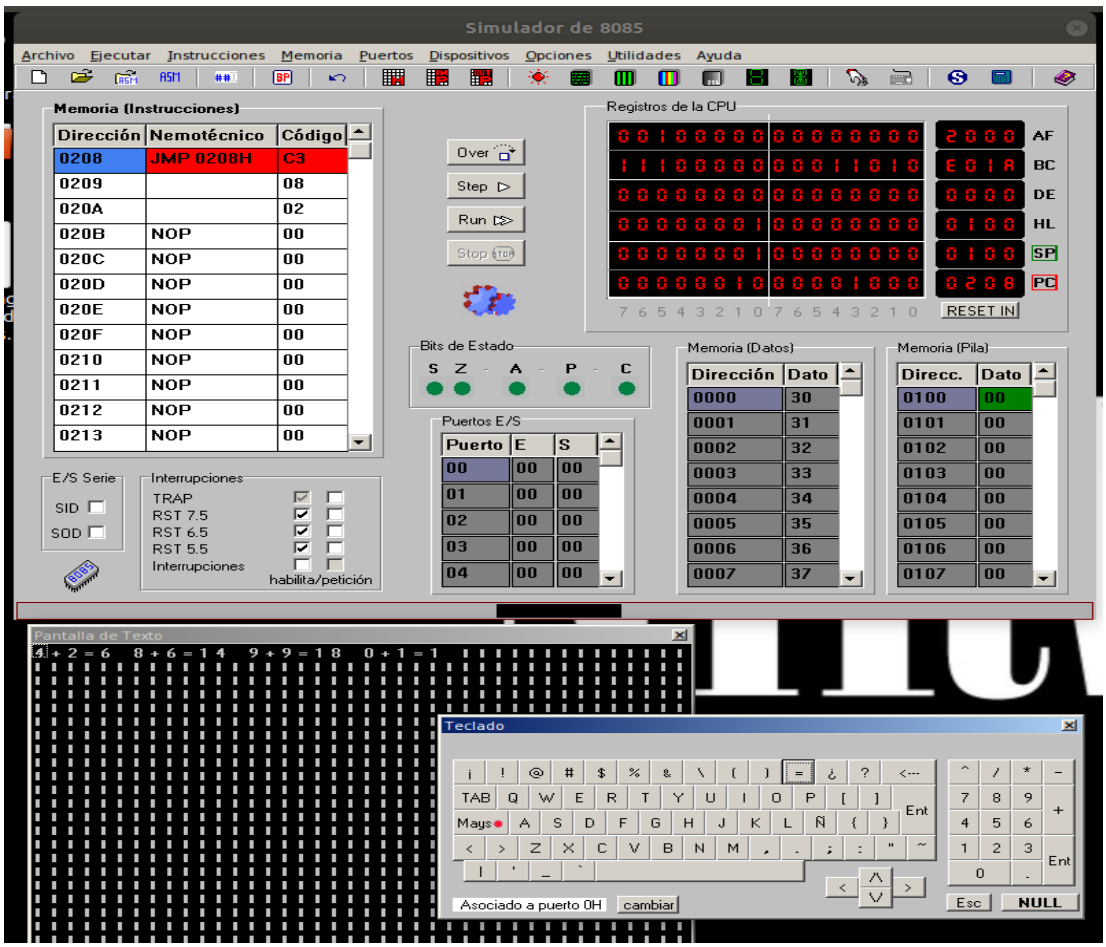
```
.define                 ; Número de carácteres permitidos
        allowed_count 15
.data 00h               ; Carácteres Permitidos: 0 a 9,+, -, &, |, =
        allowed: db
30h,31h,32h,33h,34h,35h,36h,37h,38h,39h,2Bh,2Dh,26h,7Ch,3Dh


.org 100h
        pila:                   ; Posición Pila


.org 200h                       ; Programa Principal
        lxi H, pila             ; Puntero de pila apuntando a 100h
```

```
        sphl
        mvi B, E0h          ; Par BC apuntando
        mvi C, 00h          ; a la memoria de texto
bucle:
        jmp bucle           ; Loop infinito

.org 0024h                  ; Dirección de interrupción TRAP
        call string_in      ; Llamada a subrutina de introducción
        ret                 ; de datos por consola

.org 300h                   ; Rutina que captura y muestra
string_in:
        in 00h              ; Puerto de entrada
        cpi 00h             ; Si no había carácter introduido, sale
        jz no_tecla         ; Si había, comprueba que está
                            ; permitido
tecla:
        call check_allowed  ; Carácter Permitido? Sino 00h
        cpi 00h             ; Compara si está permitido
        jz no_tecla         ; Si no lo está, es 00h y saltamos
        stax B              ; Si está permitido lo imprimimos
        inx B               ; por pantalla y avanzamos un espacio

no_tecla:
        ret                 ; Fin de la subrutina
```

```
check_allowed:          ; Subrutina para el control carácteres
        push D
        push H
        mvi E, allowed_count    ; Contador de carácteres permitidos
        lxi H, allowed          ; Primer carácter permitido

allowed_loop:           ; Comprueba si el carácter está en
        mov D,M         ; la lista de carácteres permitidos
        cmp D           ; Si A y D contienen el mismo carácter,
        jz is_allowed   ; saltamos a is_allowed
        inx H           ; Sino seguimos
        dcr E           ; Decrementamos el contador
        jnz allowed_loop        ; Hasta que E<=0 o carácter permitido
        jmp not_allowed         ; Si E <= 0, finalizamos el loop

is_allowed:             ; Carácter permitido
        mov A,D         ; Lo movemos a acumulador

        CPI 3DH         ; Si es el símbolo «=»,
        JZ suma         ; saltamos a la etiqueta «suma»

        jmp end_allowed ; Sino finalizamos la subrutina y
                        ; pedimos el siguiente carácter

not_allowed:            ; Carácter no Permitido
        mvi A,00h       ; Pone A a 00h, no escribe nada
```

```
end_allowed:            ; Fin de subrutina check_allowed
        pop H
        pop D
        ret

suma:                   ; Realizamos la operación suma
        STAX B          ; Imprimimos el símbolo "="
        DCX B           ; Decrementamos 1 vez para obtener el
                        ; último número de la suma

        LDAX B          ; Lo cargamos a acumulador
        MOV E,A         ; y luego al registro E

        DCX B           ; Decrementamos 2 veces el par BC
        DCX B           ; para obtener el primer número
        LDAX B          ; y lo cargamos a acumulador

        ADD E           ; Sumamos  el contenido de A y E y
                        ; lo guardamos en A
        INX B           ; Incrementamos 4 veces el par BC
        INX B           ; para situarnos en la posición de la
        INX B           ; pantalla justo después del "="
        INX B
        SUI 6AH         ; Le restamos 60H para que dé un
                        ; número comprendido entre 0 y 9
                        ; (la suma mínima 0+0 en hexadecimal
                        ; es 30H + 30H = 60H)
                        ; A la vez 0AH, ya que el carry salta
                        ; a partir del número 10D = 0AH
```

```
        MOV H,A             ; Movemos el resultado de A a H

        JP carry            ; Si el resultado después de la resta es
                            ; positivo, hay carry, si no seguimos

        ADI 3AH             ; Sumamos 3AH = 30H + 0AH para
                            ; obtener carácteres de 30H a 39H y no
                            ; hexadecimales negativos

imprimir:                   ; Etiqueta para imprimir por pantalla
        STAX B              ; después de la suma y/o carry
        INX B
        MVI A, 20H          ; Añadimos un espacio de
                            ; separación entre las sumas
        JMP end_allowed     ; Finalizamos la subrutina check_allowed

carry:                      ; Etiqueta para calcular el carry
        MVI A, 31H          ; Como estamos trabajando con 1 dígito,
                            ; la suma máxima es 9+9=18 y
                            ; podemos imprimir directamente un 1
        STAX B              ; Lo mostramos por pantalla
        INX B               ; e incrementamos una posición

        MOV A,H             ; Recogemos el resultado de la resta
                            ; de 6AH, que tendrá la forma de
                            ; 00H hasta 09H
        ADI 30H             ; Para que sea un carácter del 0 al 9,
                            ; le sumamos 30H para tener 30H a 39H
        JMP imprimir        ; Salto incondicional a imprimir
```



"Demostración gráfica de la suma en i8085"

- ✓ ¿Cómo gestionais el problema del signo?

En mi código, no contemplo la suma de números negativos, es decir, mis carácteres numéricos permitidos son el 0 al 9. Esta suma de dígitos enteros negativos se realizará con la resta.

En cuanto al operando «+», en este caso, como solo realizo la suma, solo comparo si ya se ha introducido el símbolo «=» para realizarla. Una vez estemos en el código ensamblador final, sí compararé que el operando introducido sea el «+».

- ✓ ¿Cómo gestionais el problema del overflow?

Al realizar la suma de números del 0 al 9 nos podemos encontrarnos con 2 casos:

a) Sumas cuyo resultado sea de un único dígito.

b) Sumas cuyo resultado tenga 2 dígitos.

En el primer caso, el máximo resultado será 9 y no se producirá ningún overflow para mostrar por pantalla de texto. Es decir, no necesitaremos entrar en la etiqueta de «carry».

En el segundo caso, tenemos sumas del estilo 4+6=10 hasta 9+9=18 (siendo este el máximo resultado). Por lo tanto, se produce un overflow a la hora de mostrar por pantalla el resultado.

Como estamos haciendo sumas de 1 dígito, podemos seguir la estrategia de imprimir primero un «1» y luego el 2º dígito. Para ello, debemos seguir el siguiente procedimiento que, para que quede más claro, lo haremos a través de un ejemplo (5+5):

1. Sumamos los 2 dígitos en hexadecimal: 35H +35H = 6AH
2. Este número en hexadecimal no nos interesa. Como este tipo de sumas, cuyo resultado es 10, son las sumas mínimas para que se produzca un overflow, restaremos 6AH.

   Esto se debe a que restando 60H tendremos números hexadecimales del estilo 00H a 09H. A la vez, si restamos 0AH == 10D sabremos si hay overflow o no.
3. Después de hacer la resta, obtendremos un número positivo o negativo: si es positivo o 00H se produce el overflow y saltaremos al apartado de carry (en el ejemplo obtendríamos 00H y saltaremos).

   En el caso de obtener un número negativo, pasamos a la siguiente línia y sumamos 3AH = 30H + 0AH. Esto se hace para obtener números hexadecimales del 30H al 39H y revertir la comprobación del overflow.
4. En la etiqueta «carry», primero mostraremos por pantalla el «1» inicial y luego recogeremos el resultado de la resta de 6AH (en este ejemplo el 00H) y le sumaremos 30H para que sea un carácter númerico del 0 al 9 (tendremos 30H == 0D).
5. Por último, saltamos incondicionalmente a la etiqueta «imprimir». Mostramos por pantalla el 2º dígito junto a un espacio entre operaciones: en la pantalla de texto se verá 5+5=10_ siendo «_» la barra espaciadora (20H).

2. Resta de dos valores introducidos por consola

Diseñad una subrutina que a partir de dos números (base 10) introducidos por teclado del simulador i8085 haga la resta y presente el resultado en la pantalla de texto del i8085. Haced servir direccionamiento directo y indirecto y indicad en el código donde tenemos estos direccionamientos.

- ✓ **Tarea 2.** Subid el código de la resta:

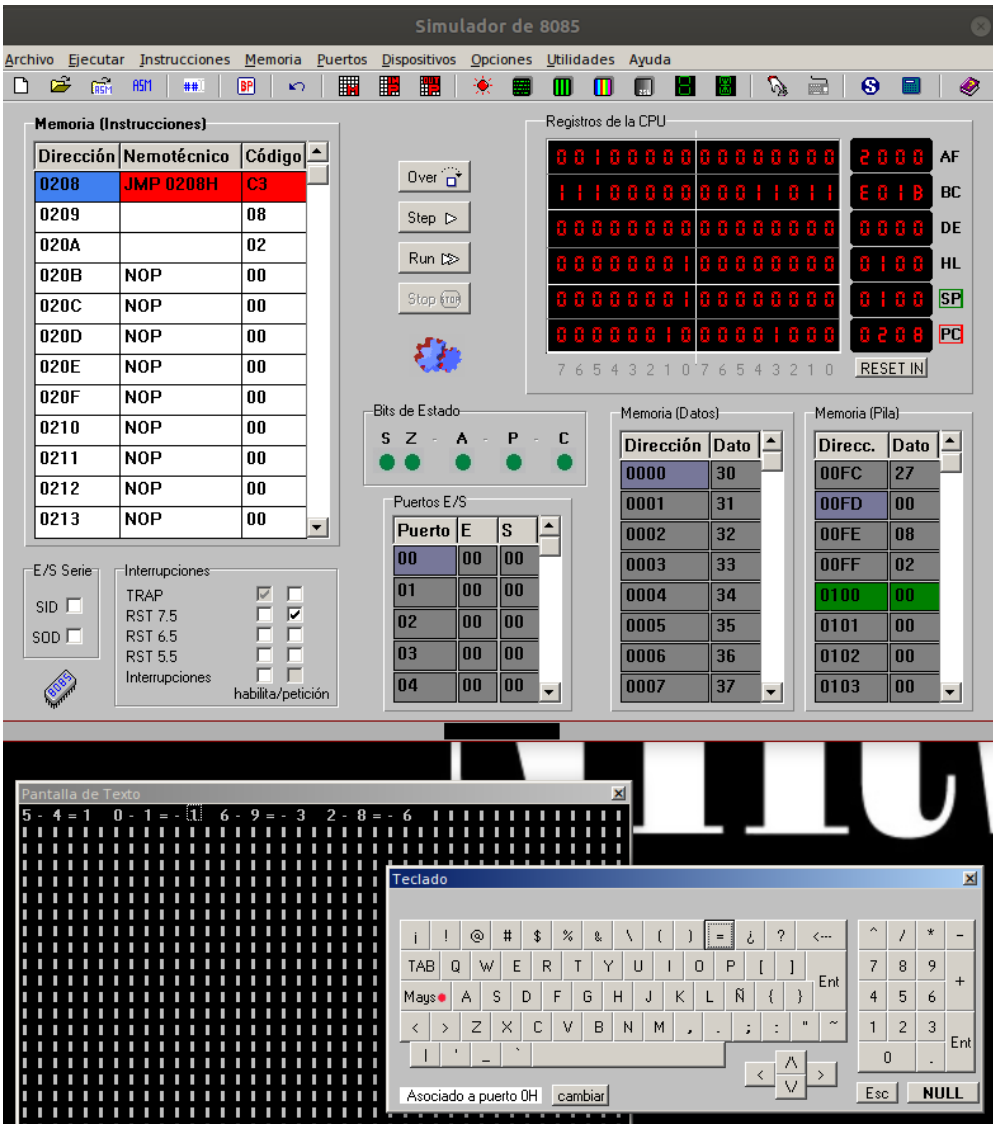
```
.define                 ; Número de carácteres permitidos
        allowed_count 15
.data 00h               ; Carácteres Permitidos: 0 a 9,+, -, &, |, =
        allowed: db
30h,31h,32h,33h,34h,35h,36h,37h,38h,39h,2Bh,2Dh,26h,7Ch,3Dh


.org 100h
        pila:                   ; Posición Pila

.org 200h                       ; Programa Principal
        lxi H, pila             ; Puntero de pila apuntando a 100h
        sphl
        mvi B, E0h              ; Par BC apuntando
        mvi C, 00h              ; a la memoria de texto

bucle:
        jmp bucle               ; Loop infinito

.org 0024h                      ; Dirección de interrupción TRAP
        call string_in          ; Llamada a subrutina de introducción
        ret                     ; de datos por consola
```

```
.org 300h                   ; Rutina que captura y muestra
string_in:
        in 00h              ; Puerto de entrada
        cpi 00h             ; Si no había carácter introduido, sale
        jz no_tecla         ; Si había, comprueba que está
                            ; permitido
tecla:
        call check_allowed  ; Carácter Permitido? Sino 00h
        cpi 00h             ; Compara si está permitido
        jz no_tecla         ; Si no lo está, es 00h y saltamos
        stax B              ; Si está permitido lo imprimimos
        inx B               ; por pantalla y avanzamos un espacio

no_tecla:
        ret                 ; Fin de la subrutina

check_allowed:              ; Subrutina para el control carácteres
        push D
        push H
        mvi E, allowed_count ; Contador de carácteres permitidos
        lxi H, allowed      ; Primer carácter permitido

allowed_loop:               ; Comprueba si el carácter está en
        mov D,M             ; la lista de carácteres permitidos
        cmp D               ; Si A y D contienen el mismo carácter,
        jz is_allowed       ; saltamos a is_allowed
        inx H               ; Sino seguimos
```

```
        dcr E               ; Decrementamos el contador
        jnz allowed_loop    ; Hasta que E<=0 o carácter permitido
        jmp not_allowed     ; Si E <= 0, finalizamos el loop

is_allowed:                 ; Carácter permitido
        mov A,D             ; Lo movemos a acumulador
        CPI 3DH             ; Si es el símbolo «=»,
        JZ resta            ; saltamos a la etiqueta «resta»
        jmp end_allowed     ; Sino finalizamos la subrutina y
                            ; pedimos el siguiente carácter
not_allowed:                ; Carácter no Permitido
        mvi A,00h           ; Pone A a 00h, no escribe nada

end_allowed:                ; Fin de subrutina check_allowed
        pop H
        pop D
        ret

resta:                      ; Realizamos la operación resta
        STAX B              ; Imprimimos el símbolo “=”
        DCX B               ; Decrementamos 1 vez para obtener el
                            ; ultimo número de la resta

        LDAX B              ; Lo cargamos a acumulador
        MOV E,A             ; y luego al registro E

        DCX B               ; Decrementamos 2 veces el par BC
        DCX B               ; para obtener el primer número
```

```
        LDAX B          ; y lo cargamos a acumulador
        MOV H, A        ; Guardamos el primer número en H
        SUB E           ; Restamos  el contenido de A y E y
                        ; lo guardamos en A
        INX B           ; Incrementamos 4 veces el par BC
        INX B           ; para situarnos en la posición de la
        INX B           ; pantalla justo después del =
        INX B
        JM signo        ; Si el resultado después de la resta es
                        ; negativo, hay overflow; si no seguimos
        ADI 30H         ; Sumamos 30H para obtener carácteres
                        ; de 30H a 39H y no hexadecimales
                        ; del tipo 00h a 09H

imprimir:               ; Etiqueta para imprimir por pantalla
        STAX B          ; después de la resta y/o signo
        INX B
        MVI A, 20H      ; Añadimos un espacio de
                        ; separación entre las sumas
        JMP end_allowed ; Finalizamos la subrutina check_allowed

signo:                  ; Etiqueta para el signo de la resta
        MVI A, 2DH      ; Ya sabemos que el resultado es
                        ; negativo, imprimimos el signo "-"

        STAX B          ; Lo mostramos por pantalla
        INX B           ; e incrementamos una posición
        MOV A,E         ; Recogemos el segundo número
                        ; de la resta y lo ponemos en A
```

```
SUB H           ; La estrategia es la siguiente:
                ; la resta en valor absoluto de 2 digítos
                ; tendrá el mismo resultado
                ; independientemente del orden de los
                ; factores (ex.: |4 - 5| = 1 = |5 - 4| )
                ; Por ello para obtener el resultado
                ; restamos el 2º número menos el 1º
                ; y así tenemos un resultado positivo.
ADI 30H         ; Para que sea un carácter del 0 al 9
                ; le sumamos 30H y tenemos 30H a 39H
JMP imprimir    ; Salto incondicional a imprimir
```



"Demostración gráfica de la resta en i8085"

- ✓ ¿Cómo gestionais el problema del signo? ¿Y el problema del carry?

En la resta de números del 0 al 9, también nos podemos encontrar con 2 casos:

a) Restas cuyo resultado sea un dígito positivo (i.e. 5 - 3 = 2).

b) Restas cuyo resultado sea un dígito negativo (i.e. 3 - 5 = -2).

En el primer, caso con solo hacer un SUB de acumulador y otro registro, en mi caso E, obtendremos el resultado correcto. Aún así, habrá que sumarle 30H, ya que al restar 35H – 33H = 02H y no es un carácter de los que queremos mostrar en la pantalla de texto, que son del 30H==0D al 39H==9D).

En el segundo caso, se produce un carry ya que obtenemos un resultado negativo. Para ello, debemos primero comprobar que esto es lo que está ocurriendo y luego conseguir el resultado que queremos.

Es muy fácil de comprobar comparando los flags de signo: si el contenido de acumulador nos da un flag de signo = 1, hay un carry porque el resultado es negativo y debemos saltar a la etiqueta «signo». Si el flag de signo es 0, pasamos a la siguiente línia y sumaos 30H para poder mostrar por pantalla correctamente.

La etiqueta «signo» sirve para:

1. Imprimir el signo «-», de ahí su nombre.
2. Recoger los números implicados en la resta y cambiarles el orden. Es decir, restaremos el 2º dígito menos el 1º, ya que esto nos dará un valor positivo.

   En el código lo he explicado así: |4 - 5| = 1 = |5 – 4|, es decir, da igual el orden de los factores si la resta está en valor absoluto. Por lo tanto, podemos imprimir 1º el signo y luego, el número resultante.
3. Sumar 30H para obtener números hexadecimales de 30H a 39H y no de 00H a 09H, que son el resultado de hacer por ejemplo 35H – 34H = 01H.

3. Ensamblando el código

A partir de los códigos generados en los apartados 1 y 2, haced un programa capaz de hacer sumas, restas, AND's y OR's.

- ✓ **Tarea 3.** Subid el código final:

Antes de mirar el código final, en el Anexo están los códigos de AND y OR que he ultilizado para implementar este programa.

CÓDIGO FINAL DE UNA CIFRA

```
.define                 ; Número de carácteres permitidos
        allowed_count 15
.data 00h               ; Carácteres Permitidos: 0 a 9,+, -, &, |, =
        allowed: db
30h,31h,32h,33h,34h,35h,36h,37h,38h,39h,2Bh,2Dh,26h,7Ch,3Dh


.org 100h
        pila:                   ; Posición Pila

.org 200h                       ; Programa Principal
        lxi H, pila             ; Puntero de pila apuntando a 100h
        sphl
        mvi B, E0h              ; Par BC apuntando
        mvi C, 00h              ; a la memoria de texto
bucle:
        jmp bucle               ; Loop infinito

.org 0024h                      ; Dirección de interrupción TRAP
        call string_in          ; Llamada a subrutina de introducción
        ret                     ; de datos por consola
```

```
.org 300h                   ; Rutina que captura y muestra
string_in:
        in 00h              ; Puerto de entrada
        cpi 00h             ; Si no había carácter introducido, sale
        jz no_tecla         ; Si había, comprueba que está
                            ; permitido
tecla:
        call check_allowed  ; Carácter Permitido? Sino 00h
        cpi 00h             ; Compara si está permitido
        jz no_tecla         ; Si no lo está, es 00h y saltamos
        stax B              ; Si está permitido lo imprimimos
        inx B               ; por pantalla y avanzamos un espacio

no_tecla:
        ret                 ; Fin de la subrutina

check_allowed:              ; Subrutina para el control carácteres
        push D
        push H
        mvi E, allowed_count ; Contador de carácteres permitidos
        lxi H, allowed      ; Primer carácter permitido

allowed_loop:               ; Comprueba si el carácter está en
        mov D,M             ; la lista de carácteres permitidos
        cmp D               ; Si A y D contienen el mismo carácter,
        jz is_allowed       ; saltamos a is_allowed
        inx H               ; Sino seguimos
```

```
        dcr E               ; Decrementamos el contador
        jnz allowed_loop    ; Hasta que E<=0 o carácter permitido
        jmp not_allowed     ; Si E <= 0, finalizamos el loop

is_allowed:                 ; Carácter permitido
        mov A,D             ; Lo movemos a acumulador
        CPI 3DH             ; Si es el símbolo «=»,
        JZ quien_es_quien   ; saltamos a «quien_es_quien», para
                            ; averiguar que operación es
        jmp end_allowed     ; Sino finalizamos la subrutina y
                            ; pedimos el siguiente carácter

not_allowed:                ; Carácter no permitido
        mvi A,00h           ; Pone A a 00h, no escribe nada

end_allowed:                ; Fin de subrutina check_allowed
        pop H
        pop D
        ret
quien_es_quien:
        STAX B              ; Imprimimos el símbolo “=”
        DCX B               ; Decrementamos 1 vez para obtener el
                            ; último número de la operación
        LDAX B              ; Lo cargamos a acumulador
        MOV E,A             ; y luego lo movemos al registro E

        DCX B               ; Decrementamos 1 vez más el par BC
                            ; para obtener el operando
```

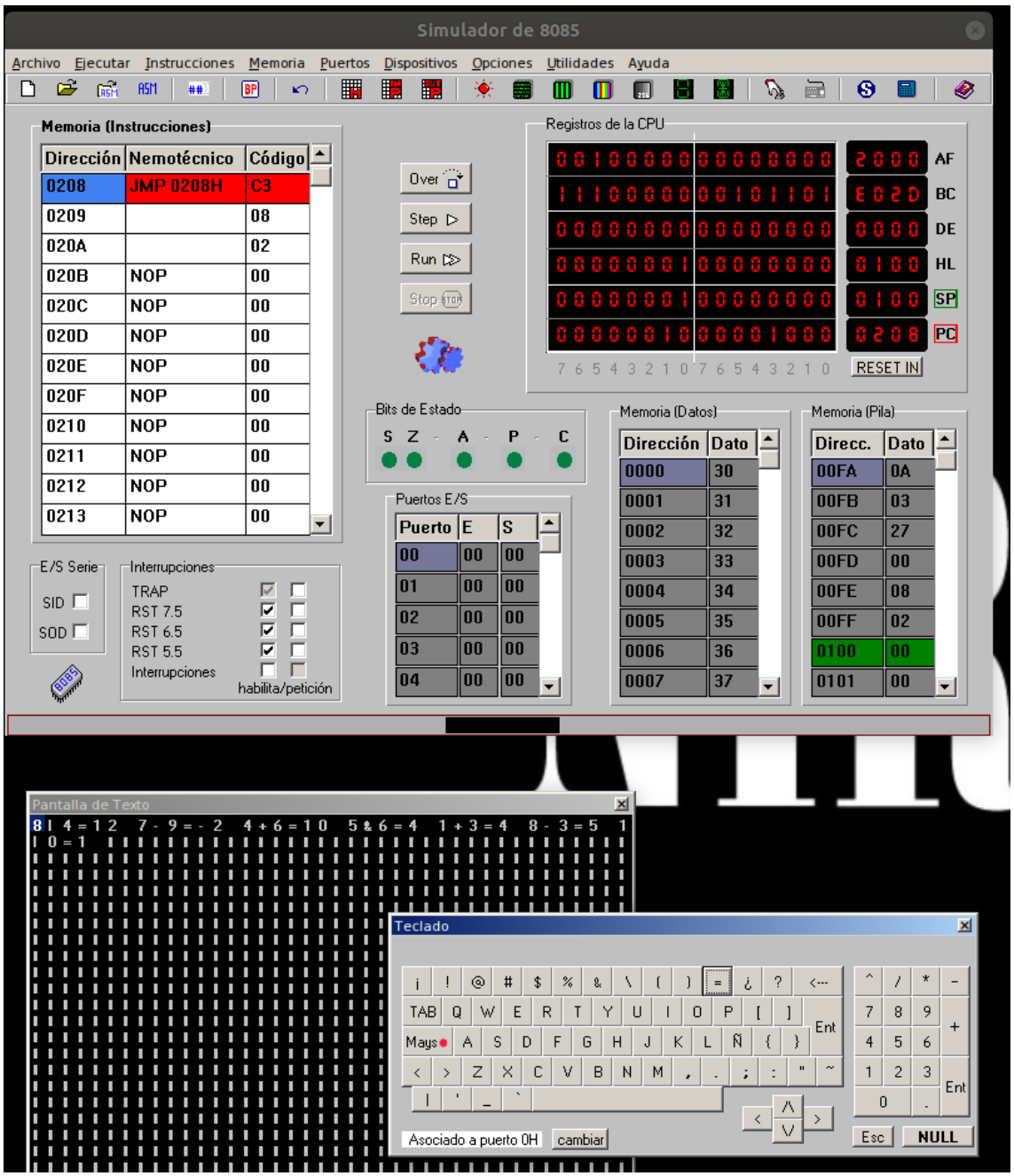
```
        LDAX B              ; y cargarlo en acumulador

        CPI 2BH             ; Si el operando es "+",
        JZ suma             ; saltamos a la etiqueta "suma"
                            ; Sino seguimos comparando
        CPI 2DH             ; Si el operando es "-",
        JZ resta            ; saltamos a la etiqueta "resta"
                            ; Sino seguimos comparando
        CPI 26H             ; Si el operando es "&",
        JZ and              ; saltamos a la etiqueta "and"
                            ; Sino seguimos comparando
        CPI 7CH             ; Si el operando es "|",
        JZ or               ; saltamos a la etiqueta "or"
        JMP not_allowed     ; Si no es ningún operando, finalizamos

suma:                       ; Realizamos la operación suma
        DCX B               ; Obtenemos el primer número
        LDAX B              ; y lo cargamos a acumulador
        ADD E               ; Sumamos el contenido de A y E y
                            ; lo guardamos en A
        INX B               ; Incrementamos 4 veces el par BC
        INX B               ; para situarnos en la posición de la
        INX B               ; pantalla justo después del "="
        INX B

        SUI 6AH             ; Le restamos 60H para que dé un
                            ; número comprendido entre 0 y 9
                            ; (la suma mínima 0+0 en hexadecimal
                            ; es 30H + 30H = 60H)
```

```
                        ; A la vez 0AH, ya que el carry salta
                        ; a partir del número 10D = 0AH
        MOV H,A         ; Movemos el resultado de A a H
        JP carry        ; Si el resultado después de la resta es
                        ; positivo, hay carry; sino seguimos
        ADI 3AH         ; Sumamos 3AH = 30H + 0AH para
                        ; obtener carácteres de 30H a 39H y no
                        ; hexadecimales negativos
        JMP imprimir    ; Salto incondicional a imprimir

carry:                  ; Etiqueta para el carry de la suma
        MVI A, 31H      ; Como estamos trabajando con 1 dígito,
                        ; la suma máxima es 9+9=18 y
                        ; podemos imprimir directamente un 1
        STAX B          ; Lo mostramos por pantalla
        INX B           ; e incrementamos una posición
        MOV A,H         ; Recogemos el resultado de la resta
                        ; de 6AH, que tendrá la forma de
                        ; 00H hasta 09H
        ADI 30H         ; Para que sea un carácter del 0 al 9,
                        ; le sumamos 30H y tenemos 30H a 39H
        JMP imprimir    ; Salto incondicional a imprimir

resta:                  ; Realizamos la operación resta
        DCX B           ; Obtenemos el primer número
        LDAX B          ; y lo cargamos a acumulador
        MOV H, A        ; Guardamos el primer número en H
        SUB E           ; Restamos el contenido de A y E y
                        ; lo guardamos en A
```

```
        INX B           ; Incrementamos 4 veces el par BC
        INX B           ; para situarnos en la posición de la
        INX B           ; pantalla justo después del "="
        INX B
        JM signo        ; Si el resultado después de la resta es
                        ; negativo, hay overflow; si no seguimos

        ADI 30H         ; Sumamos 30H para obtener carácteres
                        ; de 30H a 39H y no hexadecimales
                        ; del tipo 00h a 09H
        JMP imprimir    ; Salto incondicional a imprimir

signo:                  ; Etiqueta para el signo de la resta
        MVI A, 2DH      ; Ya sabemos que el resultado es
                        ; negativo, imprimimos el signo "-"

        STAX B          ; Lo mostramos por pantalla
        INX B           ; e incrementamos una posición
        MOV A,E         ; Recogemos el segundo número
                        ; de la resta y lo ponemos en A

        SUB H           ; La estrategia es la siguiente:
                        ; la resta en valor absoluto de 2 digítos
                        ; tendrá el mismo resultado
                        ; independientemente del orden de los
                        ; factores (ex.: |4 - 5| = 1 = |5 - 4| )
                        ; Por ello, para obtener el resultado,
                        ; restamos el 2º número menos el 1º
                        ; y así tenemos un resultado positivo
```

```
        ADI 30H         ; Para que sea un carácter del 0 al 9
                        ; le sumamos 30H y tenemos 30H a 39H
        JMP imprimir    ; Salto incondicional a imprimir


and:                    ; Realizamos la operación AND
        DCX B           ; Obtenemos el primer número
        LDAX B          ; y lo cargamos a acumulador
        ANA E           ; Realizamos A & E == A AND E y
                        ; lo guardamos en A
        INX B           ; Incrementamos 4 veces el par BC
        INX B           ; para situarnos en la posición de la
        INX B           ; pantalla justo después del "="
        INX B
        JMP imprimir    ; Salto incondicional a imprimir

or:                     ; Realizamos la operación OR
        DCX B           ; Obtenemos el primer número
        LDAX B          ; y lo cargamos a acumulador

        ORA E           ; Realizamos A | E == A OR E y
                        ; lo guardamos en A
        INX B           ; Incrementamos 4 veces el par BC
        INX B           ; para situarnos en la posición de la
        INX B           ; pantalla justo después del "="
        INX B

        SUI 3AH         ; Le restamos 30H para que dé un
                        ; número comprendido entre 0 y 9
```

```
                        ; A la vez 0AH, ya que el carry salta
                        ; a partir del número 10D = 0AH

        MOV H,A         ; Movemos el resultado de A a H
        JP carry_or     ; Si el resultado después de la resta es
                        ; positivo, hay carry; si no seguimos
        ADI 3AH         ; Sumamos 3AH = 30H + 0AH para
                        ; obtener carácteres de 30H a 39H y no
                        ; hexadecimales negativos
        JMP imprimir    ; Salto incondicional a imprimir

carry_or:               ; Etiqueta para el carry de OR
        MVI A, 31H      ; Como estamos trabajando con 1 dígito,
                        ; el resultado máximo es 15= 1111b y
                        ; podemos imprimir directamente un 1
        STAX B          ; Lo mostramos por pantalla
        INX B           ; e incrementamos una posición
        MOV A,H         ; Recogemos el resultado de la resta
                        ; de 3AH, que tendrá la forma de
                        ; 00H hasta 09H
        ADI 30H         ; Para que sea un carácter del 0 al 9,
                        ; le sumamos 30H para tener 30H a 39H

imprimir:               ; Etiqueta para imprimir por pantalla
        STAX B
        INX B
        MVI A, 20H      ; Añadimos un espacio de
                        ; separación entre las operaciones
        JMP end_allowed ; Finalizamos la subrutina check_allowed
```



"Demostración gráfica de todas las operaciones en i8085"

✓ **Pregunta 1:** ¿Qué diferencia hay entre la suma y la OR?

i) Son iguales.

ii) La OR es una operación lógica y la suma es una operación aritmética.

iii) la OR es una operació aritmética y la suma es una operación lógica.

iv) Ninguna de las anteriores es correcta.

✓ **Pregunta 2**: La instrucción STA 1234h ...

i) es una operación que carga el contenido de la posición de memoria 1234h en el acumulador.

ii) usa direccionamiento directo.

iii) usa direccionamiento inmediato.

iv) Todas son ciertas.

Conclusiones

En esta séptima y última práctica hemos consolidado y afianzado todos los conocimientos de la asignatura con i8085: instrucciones, operaciones aritmético-lógicas, subrutinas, displays, teclado, etc.

Para ellos, hemos necesitado:

- Realizar las 3 partes de este informe, con sus respectivas tareas, programas y demostraciones de funcionamiento a través de capturas de pantalla de mi escritorio.
- Ver el directo, donde se explicaba la práctica. Además he consulado los PDFs del i8085, para ver las instrucciones, sus direccionamientos, las subrutinas y los displays.
- Muchos cálculos a papel y bolígrafo para entender como funcionaría el código al que queríamos llegar y como se realizan internamiente las operaciones suma, resta, and y or.
- Prueba y error de los programas en el simulador, corrigiendo los fallos a base de «debuggear» el código y de estudiar como arreglarlos de forma óptima

En resumen, doy por cumplido el objetivo propuesto al incio del informe. Sin embargo, se ha notado el aumento de complejidad de la práctica respecto a las anteriores.

Anexo

En este apartado extra expondré los códigos básicos tanto para el AND como para el OR. Y el código de 3 cifras En ellos me he basado para hacer el código final de 1 cifra y para optimizar este último, dando como resultado el de 3 cifras.

1. AND

```
.define                         ; Número de carácteres permitidos
        allowed_count 15
.data 00h                       ; Carácteres Permitidos: 0 a 9,+, -, &, |, =
        allowed: db
30h,31h,32h,33h,34h,35h,36h,37h,38h,39h,2Bh,2Dh,26h,7Ch,3Dh


.org 100h
        pila:                   ; Posición Pila
.org 200h                       ; Programa Principal
        lxi H, pila             ; Puntero de pila apuntando a 100h
        sphl
        mvi B, E0h              ; Par BC apuntando
        mvi C, 00h              ; a la memoria de texto
bucle:
        jmp bucle               ; Loop infinito


.org 0024h                      ; Dirección de interrupción TRAP
        call string_in          ; Llamada a subrutina de introducción
        ret                     ; de datos por consola
```

```
.org 300h                   ; Rutina que captura y muestra
string_in:
        in 00h              ; Puerto de entrada
        cpi 00h             ; Si no había carácter introduido, sale
        jz no_tecla         ; Si había, comprueba que está
                            ; permitido
tecla:
        call check_allowed  ; Carácter Permitido? Sino 00h
        cpi 00h             ; Compara si está permitido
        jz no_tecla         ; Si no lo está, es 00h y saltamos
        stax B              ; Si está permitido lo imprimimos
        inx B               ; por pantalla y avanzamos un espacio
no_tecla:
        ret                 ; fin de la subrutina

check_allowed:              ; Subrutina para el control carácteres
        push D
        push H
        mvi E, allowed_count ; Contador de carácteres permitidos
        lxi H, allowed      ; Primer carácter permitido
allowed_loop:               ; Comprueba si el carácter está en
        mov D,M             ; la lista de carácteres permitidos
        cmp D               ; Si A y D contienen el mismo carácter,
        jz is_allowed       ; saltamos a is_allowed
        inx H               ; Sino seguimos
        dcr E               ; Decrementamos el contador
        jnz allowed_loop    ; Hasta que E<=0 o carácter permitido
```

```
        jmp not_allowed         ; Si E <= 0, finalizamos el loop
is_allowed:                     ; Carácter permitido
        mov A,D                 ; Lo movemos a acumulador
        CPI 3DH                 ; Si es el =,
        JZ and                  ; saltamos a «and»
        jmp end_allowed         ; Sino finalizamos la subrutina y
                                ; pedimos el siguiente carácter
not_allowed:                    ; Carácter no Permitido
        mvi A,00h               ; Pone A a 00h, no escribe nada
end_allowed:                    ; Fin de subrutina check_allowed
        pop H
        pop D
        ret

and:                            ; Realizamos el and
        STAX B                  ; Imprimimos el =
        DCX B                   ; Decrementamos 1 vez para obtener el
                                ; ultimo número de la suma
        LDAX B                  ; Lo cargamos a acumulador
        MOV E,A                 ; y luego al registro E
        DCX B                   ; Decrementamos 2 veces el par BC
        DCX B                   ; para obtener el primer número
        LDAX B                  ; y lo cargamos a acumulador
        ANA E                   ; Realizamos A & E == A AND E y
                                ; lo guardamos en A
        INX B                   ; Incrementamos 4 veces el par BC
        INX B                   ; para situarnos en la posición de la
        INX B                   ; pantalla justo después del =
```

```
        INX B
imprimir:                   ; Etiqueta para imprimir por pantalla
        STAX B              ; después de la operación lógica AND
        INX B
        MVI A, 20H          ; Añadimos un espacio de
                            ; separación entre las sumas
        JMP end_allowed     ; Finalizamos la subrutina check_allowed
```

2. OR

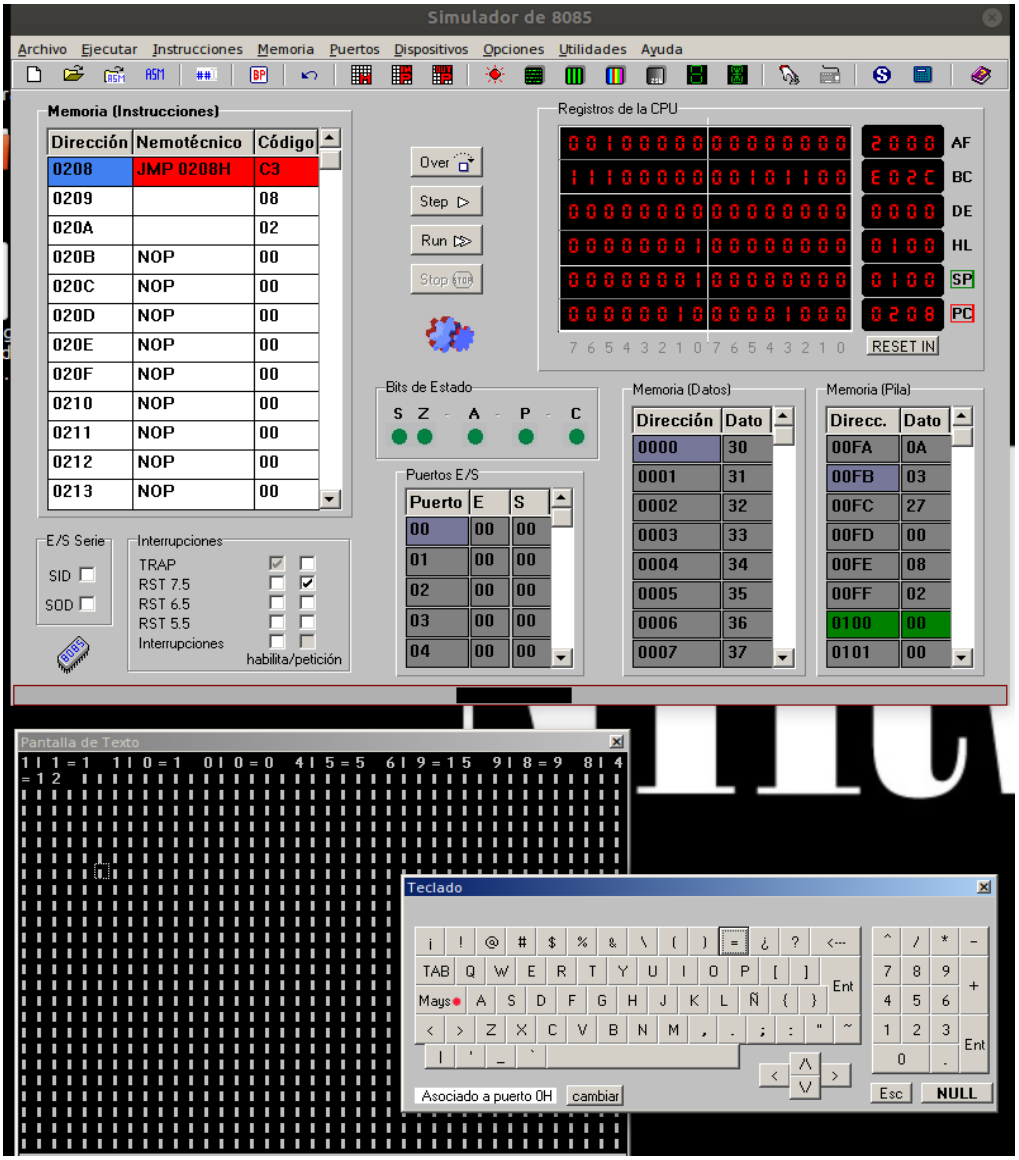
```
.define                     ; Número de carácteres permitidos
        allowed_count 15
.data 00h                   ; Carácteres Permitidos: 0 a 9,+, -, &, |, =
        allowed: db
30h,31h,32h,33h,34h,35h,36h,37h,38h,39h,2Bh,2Dh,26h,7Ch,3Dh
.org 100h
        pila:               ; Posición Pila

.org 200h                   ; Programa Principal
        lxi H, pila         ; Puntero de pila apuntando a 100h
        sphl
        mvi B, E0h          ; Par BC apuntando
        mvi C, 00h          ; a la memoria de texto
bucle:
        jmp bucle           ; Loop infinito

.org 0024h                  ; Dirección de interrupción TRAP
        call string_in      ; Llamada a subrutina de introducción
        ret                 ; de datos por consola
```

```
.org 300h                ; Rutina que captura y muestra
string_in:
        in 00h           ; Puerto de entrada
        cpi 00h          ; Si no había carácter introduido, sale
        jz no_tecla      ; Si había, comprueba que está
                         ; permitido
tecla:
        call check_allowed   ; Carácter Permitido? Sino 00h
        cpi 00h          ; Compara si está permitido
        jz no_tecla      ; Si no lo está, es 00h y saltamos
        stax B           ; Si está permitido lo imprimimos
        inx B            ; por pantalla y avanzamos un espacio
no_tecla:
        ret              ; fin de la subrutina

check_allowed:           ; Subrutina para el control carácteres
        push D
        push H
        mvi E, allowed_count ; Contador de carácteres permitidos
        lxi H, allowed   ; Primer carácter permitido

allowed_loop:            ; Comprueba si el carácter está en
        mov D,M          ; la lista de carácteres permitidos
        cmp D            ; Si A y D contienen el mismo carácter,
        jz is_allowed    ; saltamos a is_allowed
        inx H            ; Sino seguimos
```

```
        dcr E               ; Decrementamos el contador
        jnz allowed_loop    ; Hasta que E<=0 o carácter permitido
        jmp not_allowed     ; Si E <= 0, finalizamos el loop

is_allowed:                 ; Carácter permitido
        mov A,D             ; Lo movemos a acumulador
        CPI 3DH             ; Si es el =,
        JZ or               ; saltamos a or
        jmp end_allowed     ; Sino finalizamos la subrutina y
                            ; pedimos el siguiente carácter
not_allowed:                ; Carácter no Permitido
        mvi A,00h           ; Pone A a 00h, no escribe nada
end_allowed:                ; Fin de subrutina check_allowed
        pop H
        pop D
        ret

or:                         ; Realizamos el OR
        STAX B              ; Imprimimos el =
        DCX B               ; Decrementamos 1 vez para obtener el
                            ; ultimo número de la OR
        LDAX B              ; Lo cargamos a acumulador
        MOV E,A             ; y luego al registro E

        DCX B               ; Decrementamos 2 veces el par BC
        DCX B               ; para obtener el primer número
        LDAX B              ; y lo cargamos a acumulador
```

```
        ORA E           ; Realizamos A | E == A OR E y
                        ; lo guardamos en A
        INX B           ; Incrementamos 4 veces el par BC
        INX B           ; para situarnos en la posición de la
        INX B           ; pantalla justo después del =
        INX B
        SUI 3AH         ; Le restamos 30H para que dé un
                        ; número comprendido entre 0 y 9
                        ; A la vez 0AH, ya que el carry salta
                        ; a partir del número 10D = 0AH

        MOV H,A         ; Movemos el resultado de A a H
        JP carry_or     ; Si el resultado después de la resta es
                        ; positivo, hay carry, si no seguimos
        ADI 3AH         ; Sumamos 3AH = 30H + 0AH para
                        ; obtener carácteres de 30H a 39H y no
                        ; hexadecimales negativos

imprimir:               ; Etiqueta para imprimir por pantalla
        STAX B          ; después de la OR
        INX B
        MVI A, 20H      ; Añadimos un espacio de
                        ; separación entre las sumas
        JMP end_allowed ; Finalizamos la subrutina check_allowed

carry_or:               ; Etiqueta para calcular el carry de OR
        MVI A, 31H      ; Como estamos trabajando con 1 dígito,
                        ; el resultado máximo es 15= 1111b y
                        ; podemos imprimir directamente un 1
```

```
STAX B          ; Lo mostramos por pantalla
INX B           ; e incrementamos una posición
MOV A,H         ; Recogemos el resultado de la resta
                ; de 3AH, que tendrá la forma de
                ; 00H hasta 09H
ADI 30H         ; Para que sea un carácter del 0 al 9
                ; le sumamos 30H para tener 30H a 39H
JMP imprimir    ; Salto incondicional a imprimir
```



"Demostración gráfica de la OR en i8085"

CÓDIGO FINAL DE UNA CIFRA

Este código presenta diversos cambios respecto al de 1 cifra. Esto es debido a que, al tener de 1 a 3 cifras los números, no he podido usar muchas de las tácticas de optimización del código final de 1 cifra.

Aún así, se mantine la esencia del programa expuesto anteriomente:

- Introducimos dos números, sean de 1, 2 o 3 cada uno, y separados por un operando (+,-,&,|).
- Hasta que no llegamos al símbolo «=», no buscaremos que tipo de operación estamos a realizar ni el largo de las cifras.
- Una vez introducido, comprobaremos si son de 1, 2 o 3 cifras y los iremos compactando usando el método del 123 = 1*100 + 2*10 + 3*1.
- A partir de ahí, realizaremos la operación y la mostraremos por pantalla, seguida de un espacio en blanco para separar diferentes operaciones.

Cabe resaltar que tanto sumas como restas, cuyos resultados o dígitos a operar superen a 255 , tendrán un resultado por pantalla erróneo.

```
.define                 ; Número de carácteres permitidos
        allowed_count 15
        contador_x10 9
        contador_x100 99
.data 00h               ; Carácteres Permitidos: 0 a 9,+, -, &, |, =
        allowed: db
30h,31h,32h,33h,34h,35h,36h,37h,38h,39h,2Bh,2Dh,26h,7Ch,3Dh


.org 100h
        pila:                   ; Posición Pila

.org 200h                       ; Programa Principal
        lxi H, pila             ; Puntero de pila apuntando a 100h
        sphl
```

```
        mvi B, E0h          ; Par BC apuntando
        mvi C, 00h          ; a la memoria de texto
        call imprimir_espacio ; Imprimirmos un espacio inicial
bucle:
        jmp bucle           ; Loop infinito
.org 0024h                  ; Dirección de interrupción TRAP
        call string_in      ; Llamada a subrutina de introducción
        ret                 ; de datos por consola
.org 300h                   ; Rutina que captura y muestra
string_in:
        in 00h              ; Puerto de entrada
        cpi 00h             ; Si no había carácter introducido, sale
        jz no_tecla         ; Si había, comprueba que está
                            ; permitido
tecla:
        call check_allowed  ; Carácter Permitido? Sino 00h
        cpi 00h             ; Compara si está permitido
        jz no_tecla         ; Si no lo está, es 00h y saltamos
        stax B              ; Si está permitido lo imprimimos
        inx B               ; por pantalla y avanzamos un espacio
        CPI 3DH             ; Si es el símbolo «=»,
        JZ buscar_operacion ; saltamos a «buscar_operacion»,
                            ; para averiguar que operación es
        ret
no_tecla:
        ret                 ; Fin de la subrutina
```

```
check_allowed:          ; Subrutina para el control carácteres
        push D
        push H
        mvi E, allowed_count    ; Contador de carácteres permitidos
        lxi H, allowed          ; Primer carácter permitido

allowed_loop:           ; Comprueba si el carácter está en
        mov D,M         ; la lista de carácteres permitidos
        cmp D           ; Si A y D contienen el mismo carácter,
        jz is_allowed   ; saltamos a is_allowed
        inx H           ; Sino seguimos
        dcr E           ; Decrementamos el contador
        jnz allowed_loop        ; Hasta que E<=0 o carácter permitido
        jmp not_allowed         ; Si E <= 0, finalizamos el loop

is_allowed:             ; Carácter permitido
        mov A,D         ; Lo movemos a acumulador
        jmp end_allowed ; Finalizamos la subrutina y
                        ; pedimos el siguiente carácter

not_allowed:            ; Carácter no permitido
        mvi A,00h       ; Pone A a 00h, no escribe nada

end_allowed:            ; Fin de subrutina check_allowed
        pop H
        pop D
        ret
```

```
.org 500H
buscar_operacion:          ; Subrutina para operar los dígitos
        MOV H, B           ; Primero movemos los punteros a
        MOV L, C           ; a la memoria de texto a HL
        DCX H              ; y los decrementamos 2 veces
        DCX H              ; hasta el 2º número a operar
        MOV A, M           ; Lo pasamos a acumulador

        SUI 30H            ; Le restamos 30H al contenido de
        MOV E, A           ; A y lo guardamos en E
        DCX H              ; Volvemos a decrementar HL

        CALL check_if_symbol   ; Comprobamos si es el operando
        CPI 00H            ; Si lo es, es un número de 1 cifra
        JZ operacion_encontrada    ; Sino, seguimos buscando

        CALL multiplicar_x10   ; Multiplicamos las decenas por 10
        ADD E              ; Las sumamos a las unidades,
        MOV E, A           ; guardándolas otra vez en E
        DCX H              ; Decrementamos una posición

        CALL check_if_symbol   ; Si encontramos el operando,
        CPI 00H            ; es un número de 2 cifras
        JZ operacion_encontrada    ; Sino, seguimos porque es de 3
                           ; cifras
        CALL multiplicar_x100  ; Multiplicamos las centenas x100
        ADD E              ; Las sumamos con las decenas y
        MOV E, A           ; las unidades y todo queda en E
        DCX H              ; Conseguimos el operando
```

```
operacion_encontrada:       ; Una vez tengamos el operando
        DCX H               ; Realizaremos los mismos pasos
        MOV A, M            ; para el otro número a operar

        SUI 30H             ; Restamos 30H al contenido de A
        MOV D, A            ; y los guardamos en D

        DCX H               ; Decrementamos una posición
        MOV A, M            ; para estudiar el primer número
                            ; introducido hasta que
        CPI 20H             ; encontremos un espacio blanco
        JZ incrementar_x2   ; Si es, tenemos un número de 1
                            ; cifra, sino seguimos buscando
        MOV  A, M
        CALL multiplicar_x10 ; Multiplicamos las decenas por 10
        ADD D               ; Las sumamos a las unidades y el
        MOV D, A            ; resultado lo guardamos en D

        DCX H               ; Decrementamos una posición y
        MOV A, M            ; buscamos otra vez el espacio en
                            ; blanco
        CPI 20H             ; Si lo es, tenemos un número de
        JZ incrementar      ; 2 cifras, sino será de 3 cifras

        MOV A, M
        CALL multiplicar_x100 ; Multiplicamos las centenas x100
        ADD D               ; Las sumamos a las unidades y
        MOV D, A            ; las decenas y todo queda en D
```

```
incrementar:                    ; Regresamos al operando
        INX H                   ; Tengamos 2 o 3 cifras

incrementar_x2:                 ; o una única cifra
        INX H                   ; Después de reorganizar las cifras
        INX H                   ; de los números, miraremos que
                                ; operación hay que realizar
        MOV A, M

        CPI 2BH                 ; Si el símbolo es “+”, saltamos
        JZ suma                 ; Sino, seguimos buscando

        CPI 2DH                 ; Si el símbolo es “-”, saltamos
        JZ resta                ; Sino, seguimos buscando

        CPI 26H                 ; Si el símbolo es “&”, saltamos
        JZ and                  ; Sino, seguimos buscando

        CPI 7CH                 ; Si el símbolo es “|”, saltamos
        JZ or                   ; Sino, imprimimos un espacio y
                                ; se descarta la operación
        JMP imprimir_espacio
        ret

.org 600H                       ; Subrutina para comprobar que
check_if_symbol:                ; el símbolo es correcto y limpiar
        MOV A, M                ; después el acumulador
        CPI 2BH                 ; Si es la suma (+)
        JZ allowed_symbol
```

```
        CPI 2DH                 ; Si es la resta (-)
        JZ allowed_symbol

        CPI 26H                 ; Si es el AND (&)
        JZ allowed_symbol

        CPI 7CH                 ; Si es el OR (|)
        JZ allowed_symbol
        ret

allowed_symbol:
        MVI A, 00h              ; Limpiamos el acumulador
        ret


.org 700h
suma:                           ; Subrutina para sumar
        MOV A, D                ; Pasamos el 1º número a A
        ADD E                   ; y lo sumamos con el 2º
        CALL num_cifras         ; Reestructuraremos el resultado
        CALL imprimir_espacio   ; para que se vea por pantalla,
        MVI A, 00H              ; con un espacio y limpiamos A
        ret

.org 800H
resta:                          ; Subrutina para restar
        MOV A, D                ; Pasamos el 1º número a A
        SUB E                   ; y le restamos  el 2º
        JP es_positivo          ; Si el resultado es negativo
```

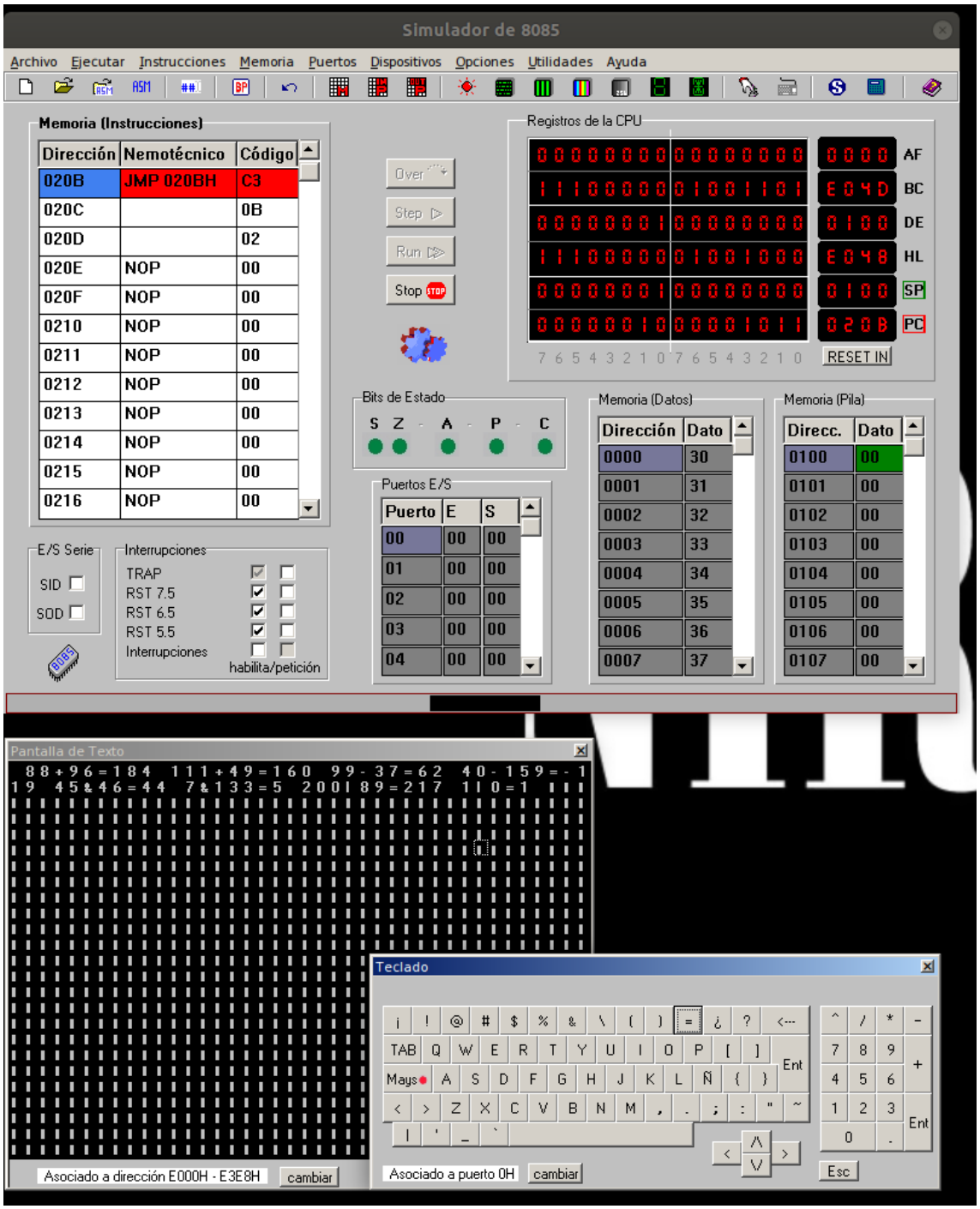
```
        CALL signo              ; miramos el signo
es_positivo:                    ; Sino continuamos
        CALL num_cifras         ; Reestructuraremos el resultado
        CALL imprimir_espacio   ; para que se vea por pantalla,
        MVI A, 00H              ; con un espacio y limpiamos A
        ret

.org 900H
signo:                          ; Subrutina para el signo de resta
        MOV A, E                ; Invertimos los números y
        SUB D                   ; restamos el 2º menos el 1º
        PUSH PSW
        MVI A, 2DH              ; Imprimimos el signo "-"
        STAX B
        INX B
        POP PSW
        ret


.org A00H
and:                            ; Subrutina para el AND
        MOV A, D                ; Pasamos el 1º número a A
        ANA E                   ; y hacemos A AND E
        CALL num_cifras         ; Reestructuraremos el resultado
        CALL imprimir_espacio   ; para que se vea por pantalla,
        MVI A, 00H              ; con un espacio y limpiamos A
        ret
```

```
.org B00H
or:                         ; Subrutina para el OR
        MOV A, D            ; Pasamos el 1º número a A
        ORA E               ; y hacemos A OR E
        CALL num_cifras     ; Reestructuraremos el resultado
        CALL imprimir_espacio ; para que se vea por pantalla,
        MVI A, 00H          ; con un espacio y limpiamos A
        ret


.org C00H
multiplicar_x10:            ; Subrutina para multiplicar por 10
        PUSH D              ; las decenas
        SUI 30H             ; Les restamos 30H para
        MVI D, contador_x10 ; establecer el contador a 9

bucle_multiplicar_x10:      ; Mientras no llegue D <= 0
        ADD M               ; sumamos las decenas 10 veces
        SUI 30H             ; en formato 00H a 09H
        DCR D
        JNZ bucle_multiplicar_x10

        POP D
        ret

multiplicar_x100:           ; Subrutina para multiplicar por
        PUSH D              ; 100 las centenas
        SUI 30H             ; Les restamos 30H para
        MVI D, contador_x100 ; establecer el contador a 99
```

```
bucle_multiplicar_x100:          ; Mientras no legue a D <= 0
        ADD M                    ; sumamos las decenas 100 veces
        SUI 30H                  ; en fomato 00H a 09H
        DCR D
        JNZ bucle_multiplicar_x100

        POP D
        ret

.org D00H
num_cifras:                      ; Subrutina para reestructurar las
        PUSH D                   ; cifras de los números y poder
        PUSH H                   ; mostrarlas bien por pantalla

        MOV H, A

        SUI 64H                  ; Si al restar a acumulador 64H
        JP tres_cifras           ; el resultado es positivo, tenemos
        ADI 64H                  ; un número de 3 cifras
                                 ; Sino lo deshacemos y miramos
        SUI 0AH                  ; si es de 2 cifras
        JP dos_cifras            ; Si el resultado es positivo, es de
        ADI 0AH                  ; 2 cifras, sino lo deshacemos

        JMP una_cifra            ; El número es de una única cifra

tres_cifras:                     ; Establecemos un contador para
        MVI D, 00H               ; obtener las centenas finales
```

```
compactar_centenas:     ; Cada resta de 64H equivale a
        INR D           ; una unidad de las centenas
        SUI 64H         ; Por ello seguimos hasta que el
        JP compactar_centenas   ; flag de cero se active

        ADI 64H         ; Como nos hemos pasado,
        MOV H, A        ; lo deshacemos para tener >= 0
        MOV A, D
                        ; Como el resultado tiene la forma
        ADI 30H         ; 00h a 09H, sumamos 30H
        STAX B          ; Lo imprimimos por pantalla
        INX B           ; y pasamos a las decenas

        MOV A, H

        SUI 0AH         ; Comprobamos que no sea 0
        MVI D, 00H      ; Limpiamos el acumulador
        JM cifra_cero   ; Si es el cero salta, sino seguimos

dos_cifras:             ; Establecemos un contador para
        MVI D, 00H      ; obtener las decenas finales

compactar_decenas:      ; Cada resta de 0AH equivale a
        INR D           ; una unidad de las decenas
        SUI 0AH         ; Por ello seguimos hasta que el
        JP compactar_decenas    ; flag de cero se active
```

```
cifra_cero:                    ; Si las decenas eran 0,
        ADI 0AH                ; deshacemos la resta de 0AH
        MOV H, A
        MOV A, D
                               ; Como el resultado tiene la forma
        ADI 30H                ; 00h a 09H, sumamos 30H
        STAX B                 ; y lo imprimimos por pantalla
        INX B                  ; pasamos a las unidades

        MOV A, H

una_cifra:                     ; Las unidades también son de
        ADI 30H                ; 00H a 09H, les sumamos 30H
        STAX B                 ; Imprimimos el último dígito
        INX B                  ; e incrementamos una posición

        POP H
        POP D
        RET

.org E00H
imprimir_espacio:              ; Subrutina para imprimir un
        PUSH PSW               ; espacio en blanco y separar
        MVI A, 20H             ; las operaciones entre si
        STAX B
        INX B
        POP PSW
        ret
```



Demostración gráfica del código final de tres cifras