EJERCITO ARGENTINO
ESCUELA SUPERIOR TECNICA
"Grl Div D Manuel Nicolás Savio"

MATERIA: Técnicas Digitales II

CURSO: IV – Plan 2007 Electrónica

DOCENTE: Ing. Daniel Steiner.

JEFE TRABAJOS PRACTICOS: Ing. Ariel Dalmas Di Giovanni.

TRABAJO PRACTICO Nro.: 1

TITULO: "Conjunto de instrucciones del microcontrolador"**.**

ALUMNO: Donis, Marina

FECHA DE ENTREGA: 21/05/2021

CALIFICACIÓN:

Introducción

Para que el microcontrolador lleve a cabo una tarea, se le debe indicar exactamente qué debe hacer, y esto se logra mediante la escritura del código que la CPU ejecutará. Para ello se utilizará el conjunto de instrucciones definido por el fabricante, donde se definen las operaciones básicas que puede realizar el procesador. Éstas, organizadas en una secuencia forman lo que conocemos como Software.
Para el desarrollo y prueba del software se utiliza el IDE IAR Embdedded Workbench para 8051.

Documentación utilizada

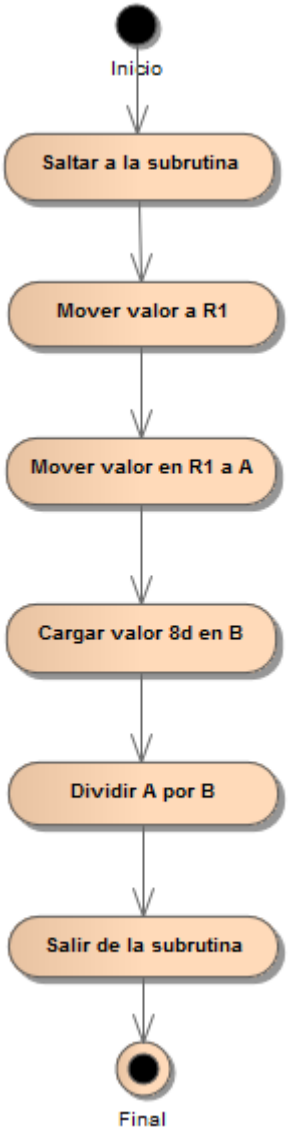
Manual de arquitectura y guía de programación de la familia de microcontroladores 80C51.

Ejercicio 5

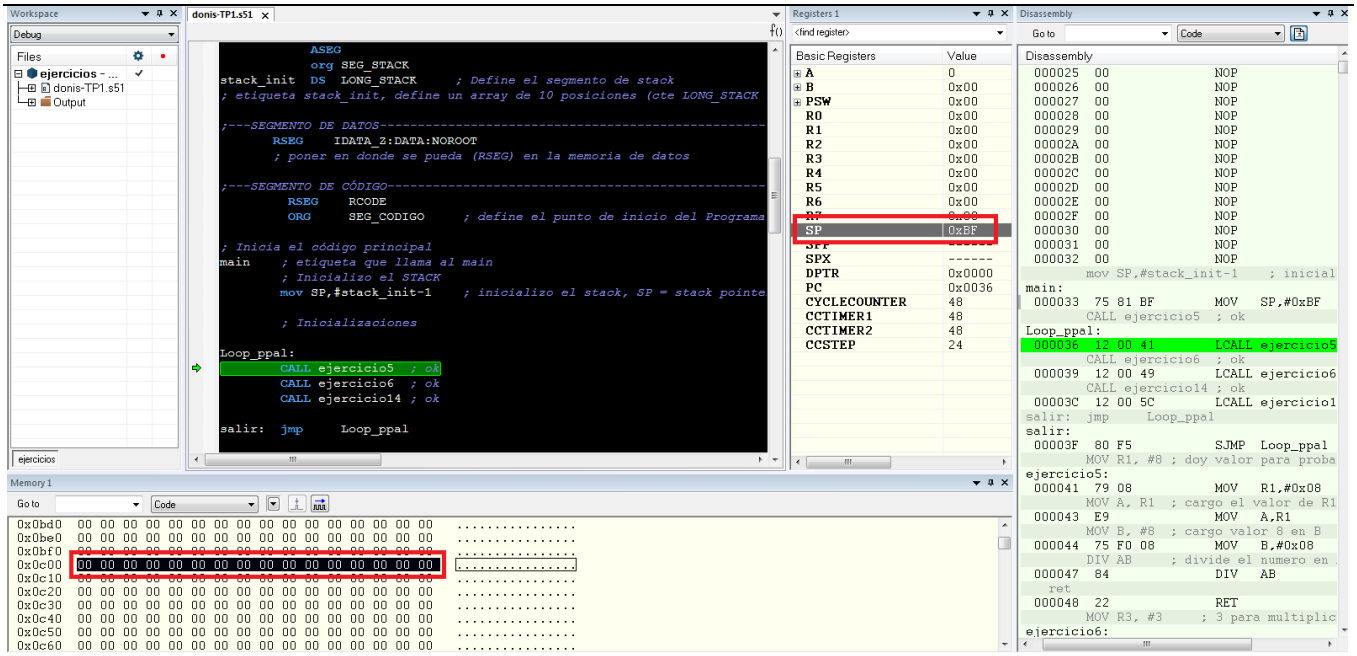
Enunciado

Dividir un número alojado en R1 por 8. Implemente el problema en una subrutina. Analice el comportamiento del microcontrolador cuando se produce el llamado a la subrutina.

Módulos de software

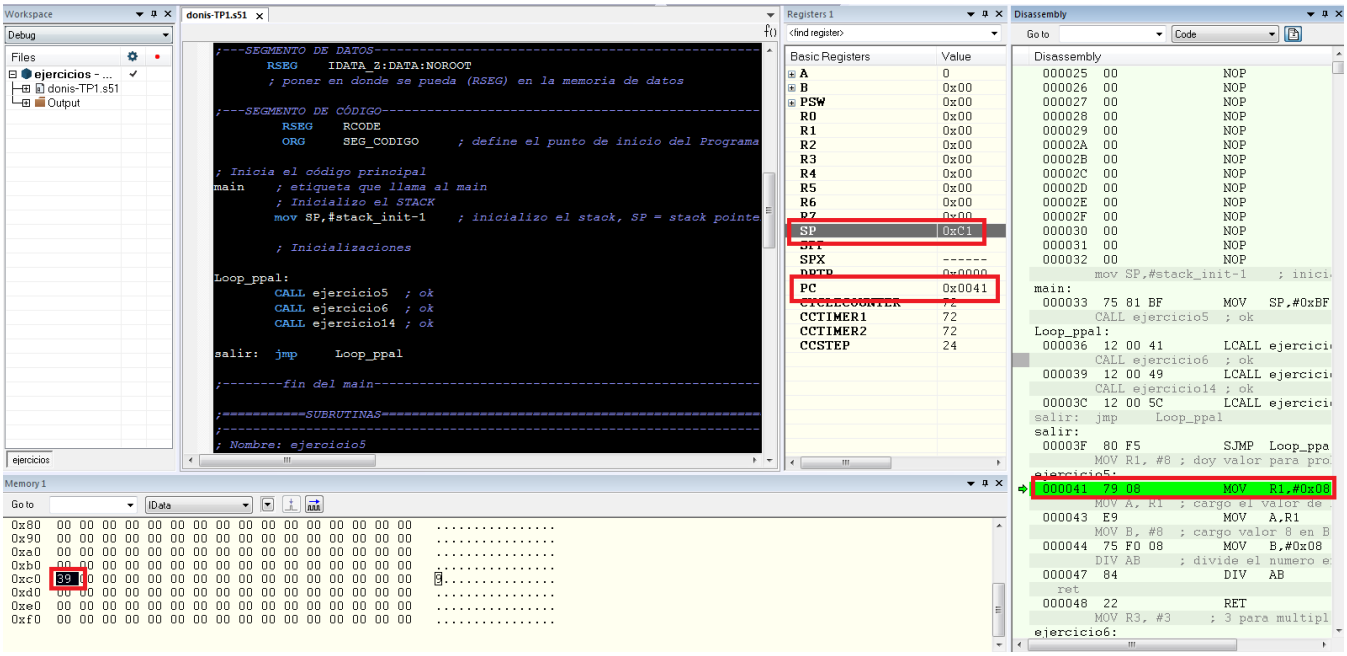


Al principio del código se define que el segmento de stack de inicializará en la posición C0h (SEG_STACK) y que tiene una longitud de 10 bytes (LONG_STACK). Al iniciar el main, la primera instrucción inicializa el puntero con la dirección anterior a la primera del segmento (stack_init-1 = BFh). Esto es porque las direcciones tienen una longitud de 2 bytes.



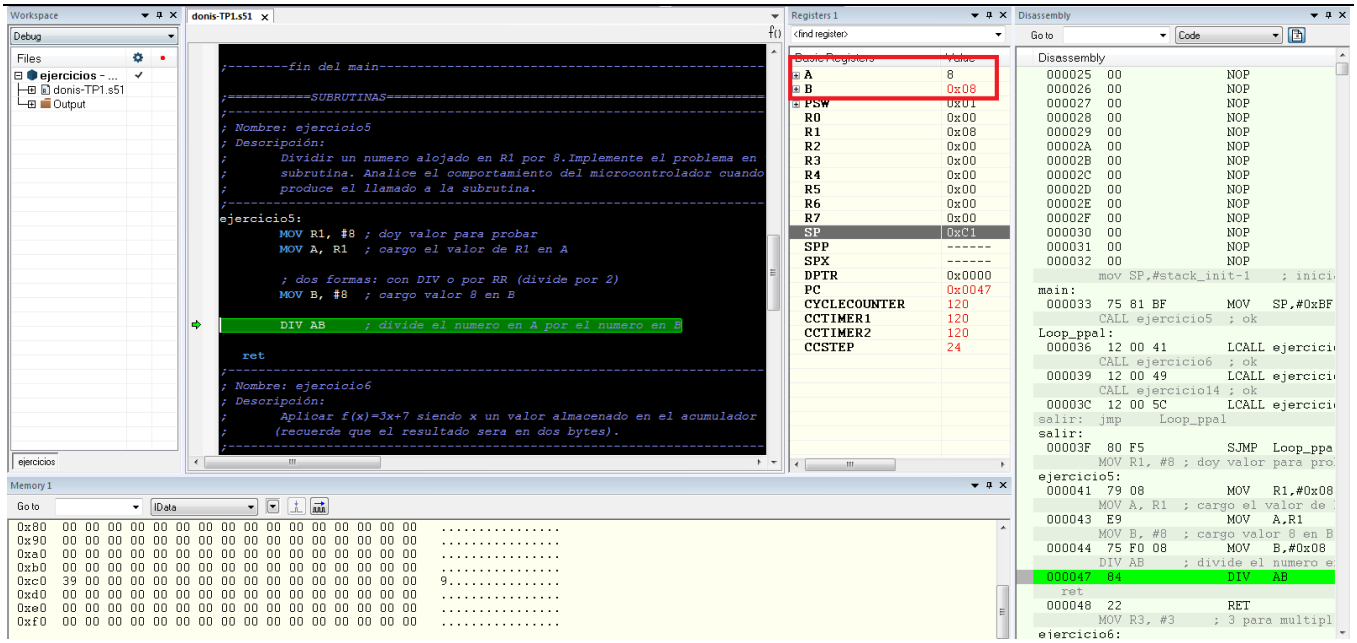
Cuando se realiza una CALL a la subrutina ejercicio5, en la posición apuntada por el SP se almacena la dirección de retorno. Esto es porque luego de correr la subrutina ejercicio5 se debe volver a la dirección 39h, que es la que contiene la siguiente instrucción.

El PC, que está apuntando a la dirección con el CALL (36h), saltará a la dirección en donde comienza la subrutina ejercicio5.

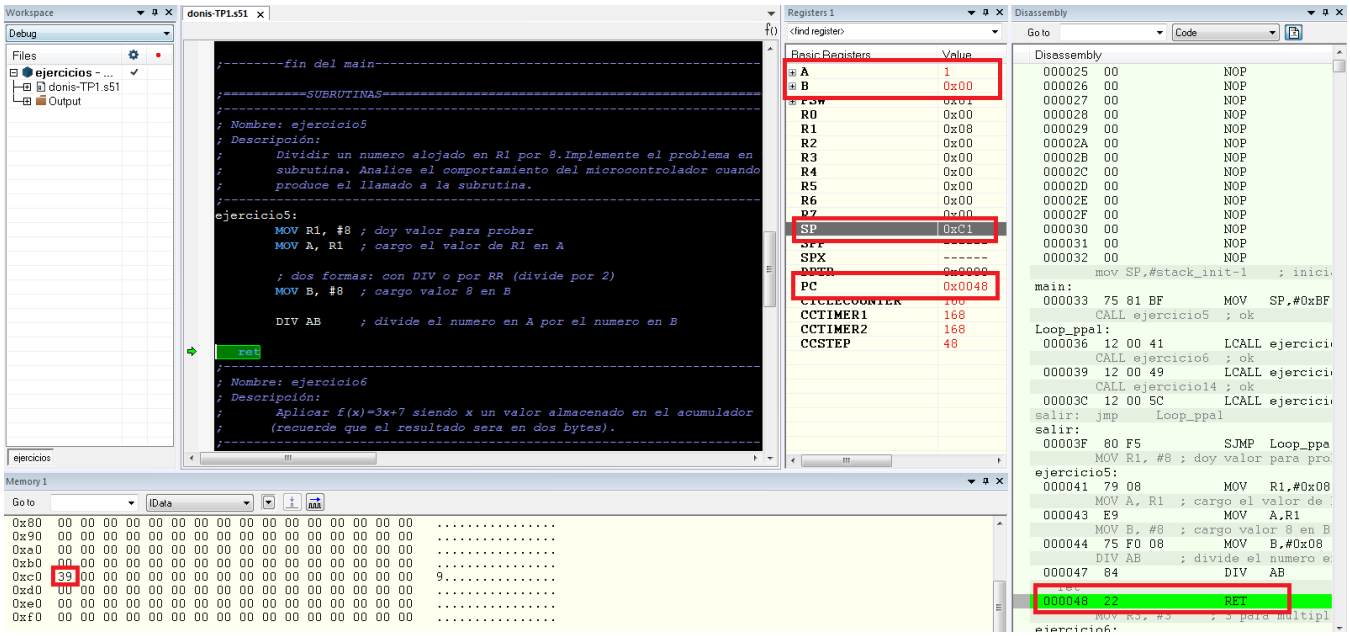


El CALL hace que el PC salte al inicio de la subrutina ejercicio5 en la posición 41h.

En la posición C0h se almacenó el valor 39h, que es parte de la dirección de retorno para el PC cuando se termine de ejecutar la subrutina. Como las direcciones tienen un tamaño de 2 bytes, la parte baja de la dirección (39h) queda almacenada en C0h, y la parte alta (00h) en C1h, por lo que el SP queda apuntando a la dirección C1h.

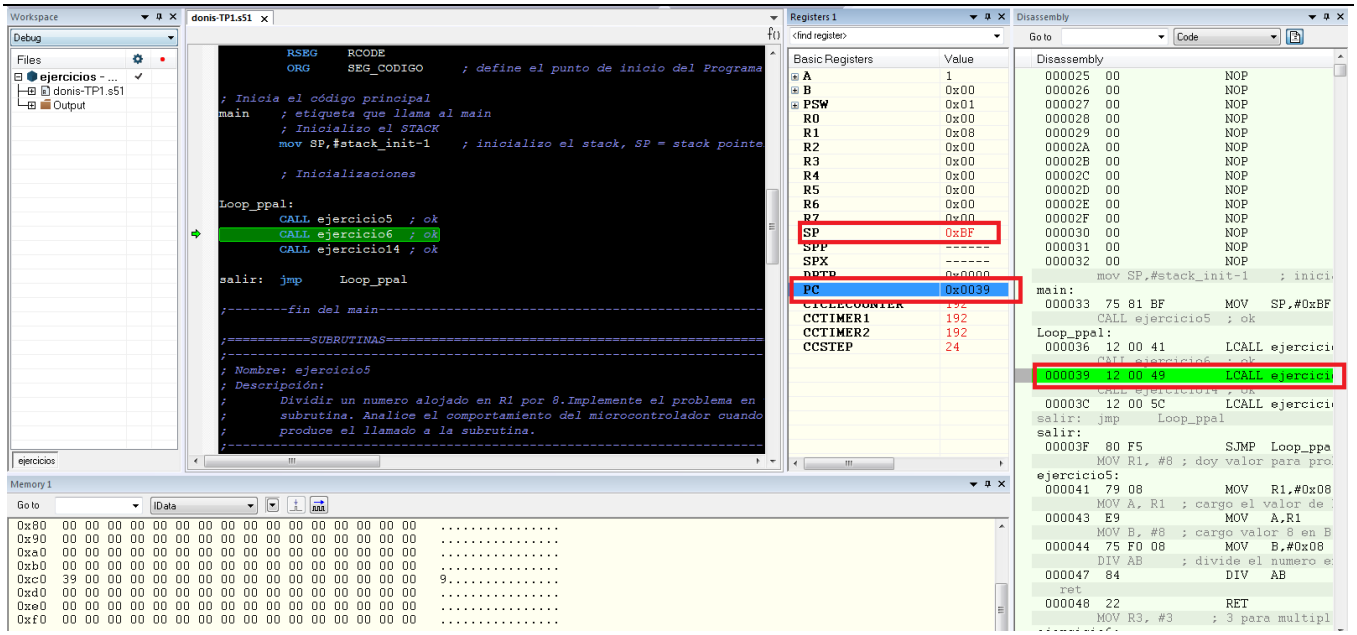


Luego de esto se corren las siguientes 3 instrucciones, que le dan valor al dividendo A = R1 y el divisor B = 8. En este caso, se almacena el valor de 8d en R1 para pruebas.



Luego de realizar la operación de división se puede ver que se actualizaron los valores de A y B con el resultado. A tiene el resultado entero y B el módulo. Al llegar a la última instrucción de la subrutina, RET, con el PC apuntando a 48h, se sale de la subrutina para volver al punto de retorno.

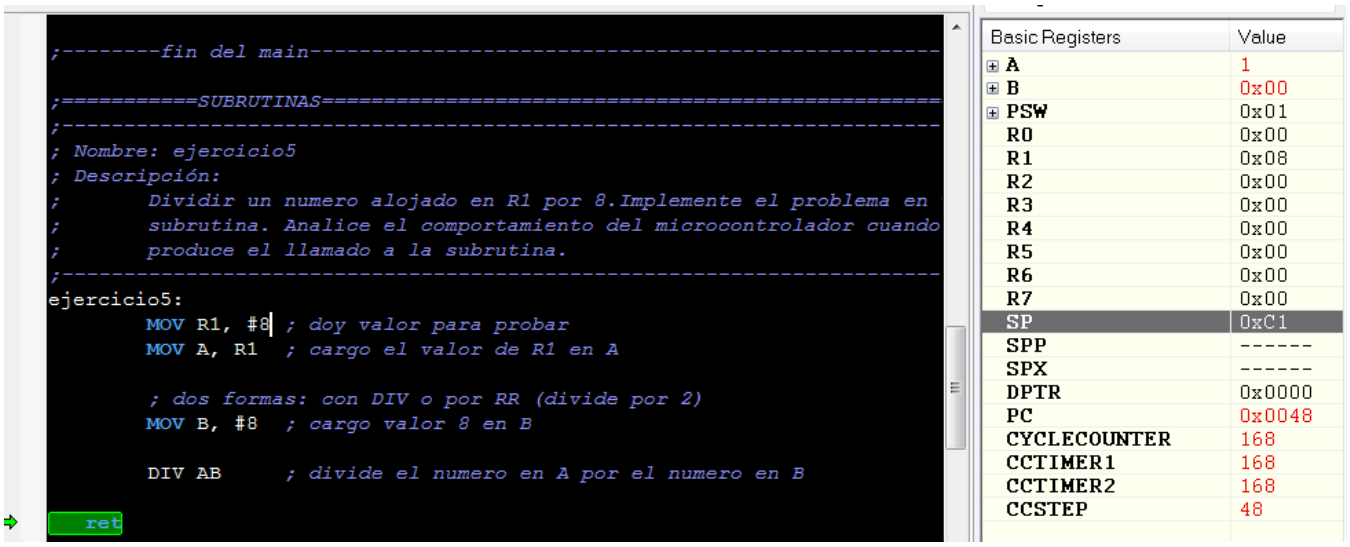
Para esto, se lee la dirección que fue almacenada en el stack al que apunta el SP, 39h, y se carga en el PC. Es importante notar que luego de esto el SP nuevamente decrementa a BFh como en el inicio del programa, es decir que volvió a su valor inicial.



Luego de esto sigue normalmente la ejecución del main, retomando desde la dirección de retorno 39h.

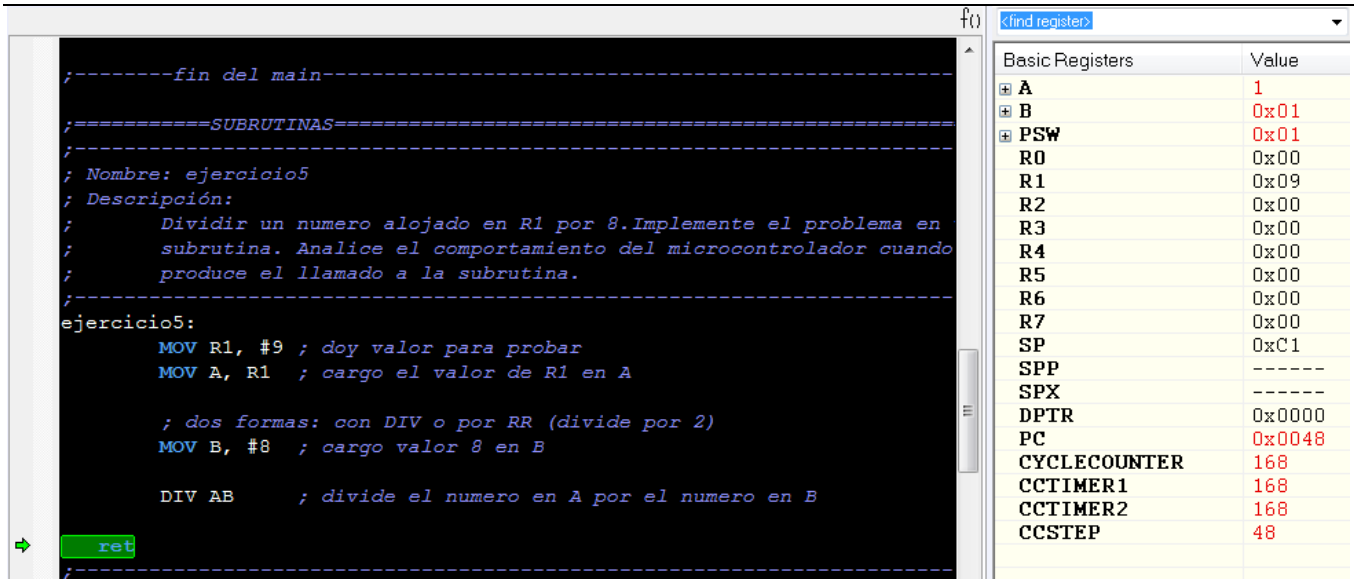
El código fuente se encuentra dentro del archivo donis-TP1.s51.

Pruebas y ensayos



Primero se tomó como valor de prueba para la división el número 8d y fue almacenado en el registro 1.

Al ser dividido por 8d, da como resultado la parte entera 1 y el módulo 0.



```
;--------fin del main-------------------------------------------------------

;==========SUBRUTINAS=================================================
;-----------------------------------------------------------------------
; Nombre: ejercicio5
; Descripción:
;       Dividir un numero alojado en R1 por 8.Implemente el problema en
;       subrutina. Analice el comportamiento del microcontrolador cuando
;       produce el llamado a la subrutina.
;-----------------------------------------------------------------------
ejercicio5:
        MOV R1, #9 ; doy valor para probar
        MOV A, R1  ; cargo el valor de R1 en A

        ; dos formas: con DIV o por RR (divide por 2)
        MOV B, #8  ; cargo valor 8 en B

        DIV AB     ; divide el numero en A por el numero en B

    ret
;-----------------------------------------------------------------------
```

Basic Registers	Value
A	1
B	0x01
PSW	0x01
R0	0x00
R1	0x09
R2	0x00
R3	0x00
R4	0x00
R5	0x00
R6	0x00
R7	0x00
SP	0xC1
SPP	------
SPX	------
DPTR	0x0000
PC	0x0048
CYCLECOUNTER	168
CCTIMER1	168
CCTIMER2	168
CCSTEP	48

Como segunda prueba se tomó como valor el número 9d, y se tuvo como resultado la parte entera 1 y el módulo 1.

Con esto se pudo verificar el correcto funcionamiento de la subrutina.

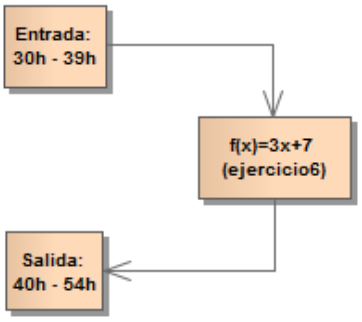
Ejercicio 14

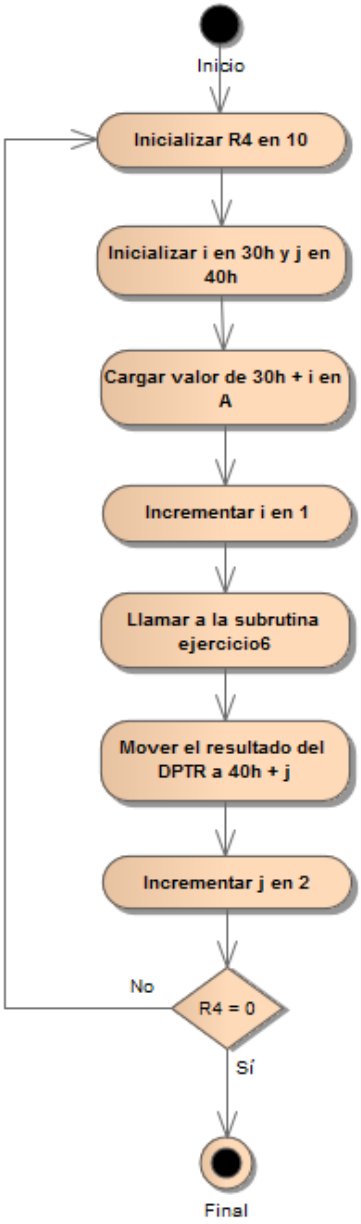
Enunciado

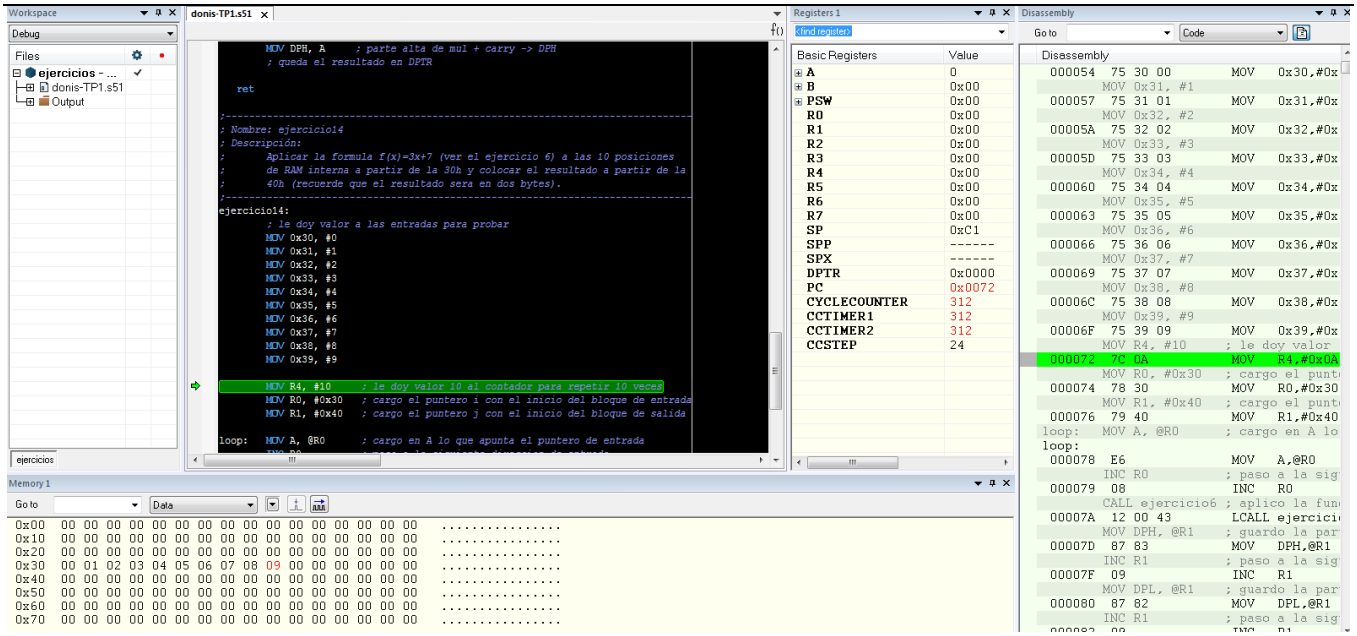
Aplicar la formula f(x)=3x+7 (ver el ejercicio 6) a las 10 posiciones de RAM interna a partir de la 30h y colocar el resultado a partir de la 40h (recuerde que el resultado será en dos bytes).

Módulos de software

Para la resolución de este ejercicio se tomó como base la solución del ejercicio 6, reutilizando la subrutina ejercicio6, que aplica la función y devuelve el resultado en el DPTR. El resto del código desarrollado para este ejercicio se planteó con el objetivo de aplicar la fórmula desde la posición 30h. Como se debe realizar la operación a 10 posiciones, el bloque de direcciones a utilizar como entrada es 30h – 39h.
Luego de realizar la operación, el resultado se debe almacenar a partir de la posición 40h.

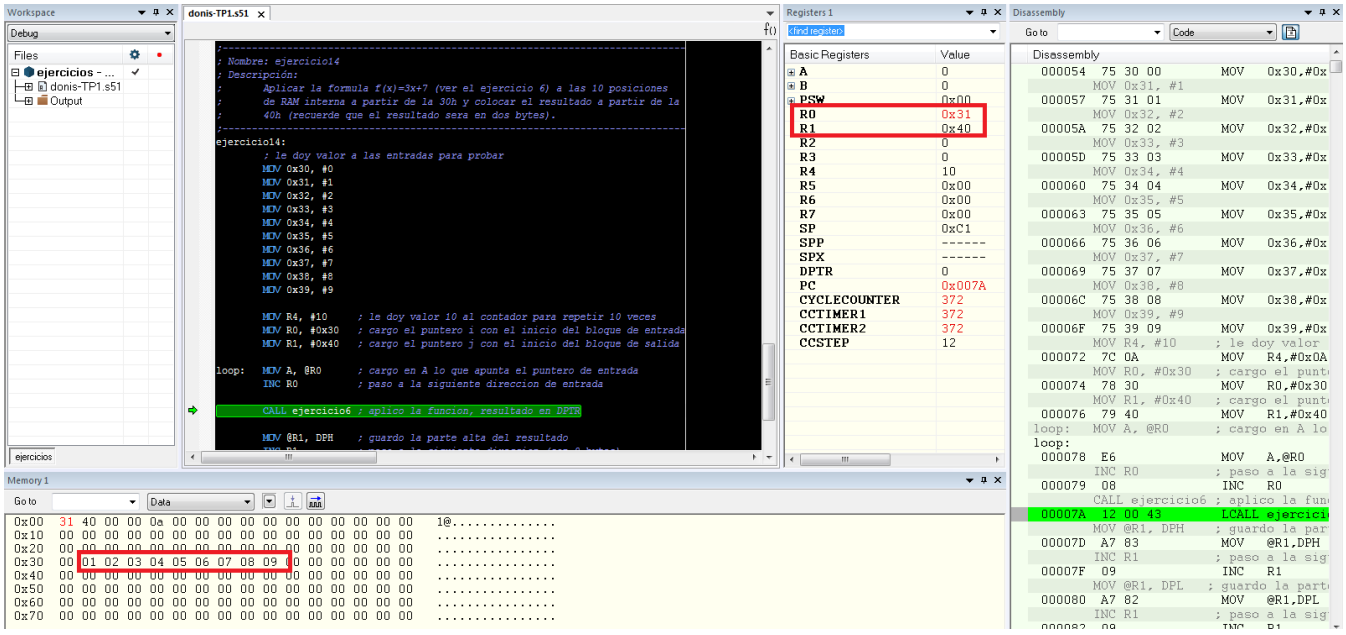




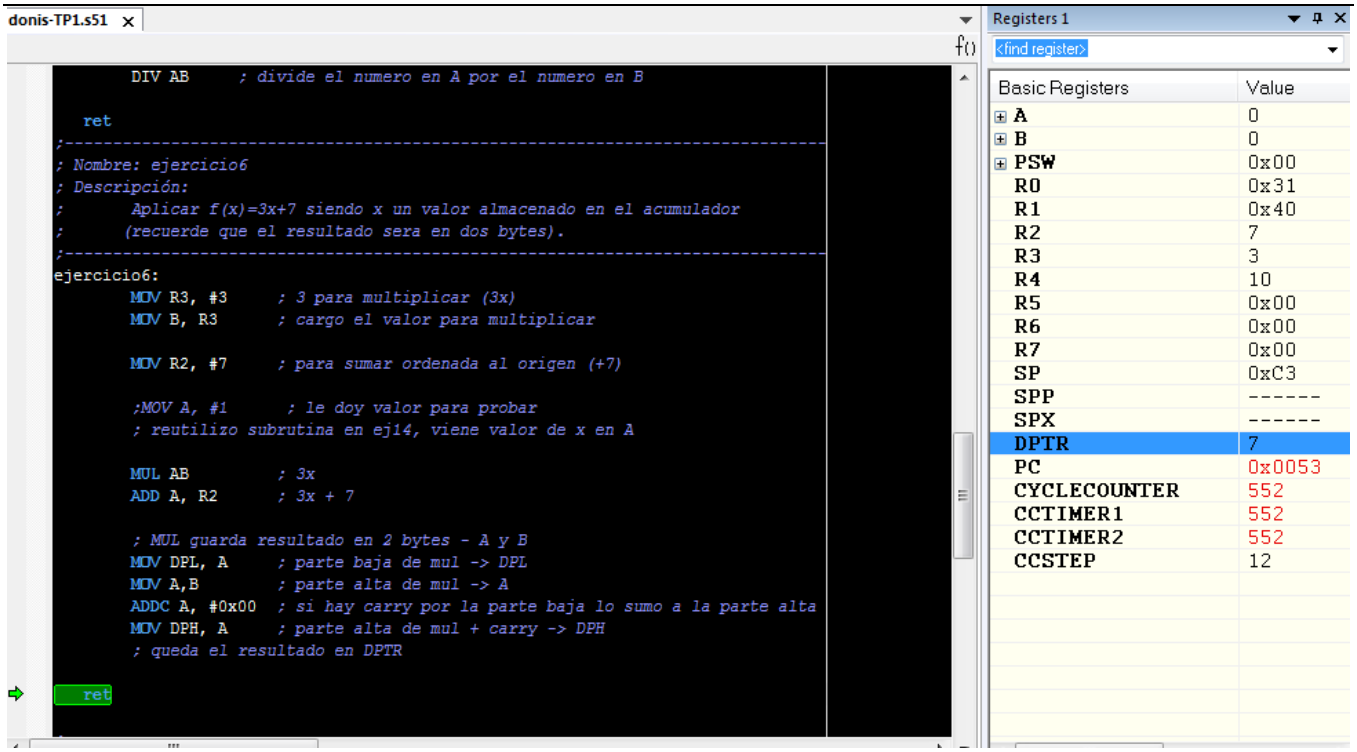


Se inicializa el registro R4 en 10 como forma de contador, para utilizarlo en conjunto con la instrucción DJNZ. Esta decrementa en 1 el valor en R4 y vuelve a la etiqueta loop si R4 es distinto de 0, logrando repetir el loop 10 veces, una para cada posición de entrada (30h – 39h).

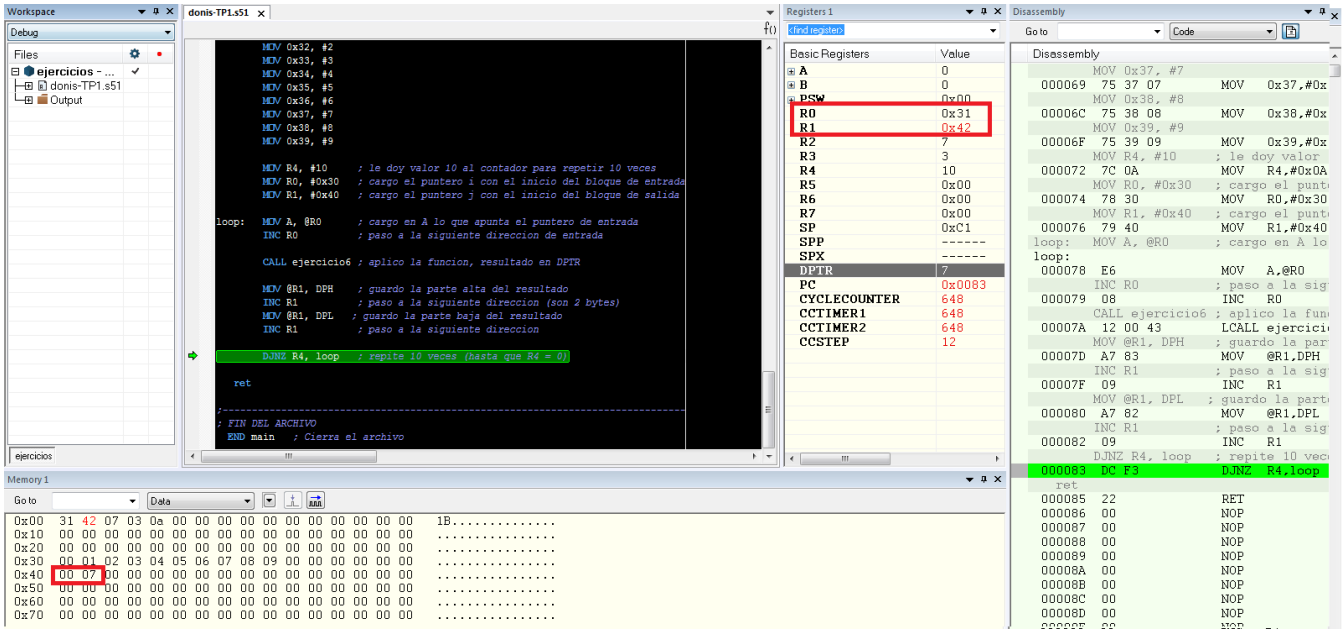
Se inicializan R0 como puntero al inicio del bloque de direcciones de entrada y R1 como puntero al inicio de bloque de direcciones de salida, y se incrementan a medida que se repite la aplicación de la función.



Antes de llamar a la subrutina ejercicio6 que aplica la función, se puede ver que los valores de prueba fueron cargados en memoria en el bloque de direcciones de entrada (C0h - C9h) y que el puntero a las direcciones de entrada ya fue incrementado en 1.

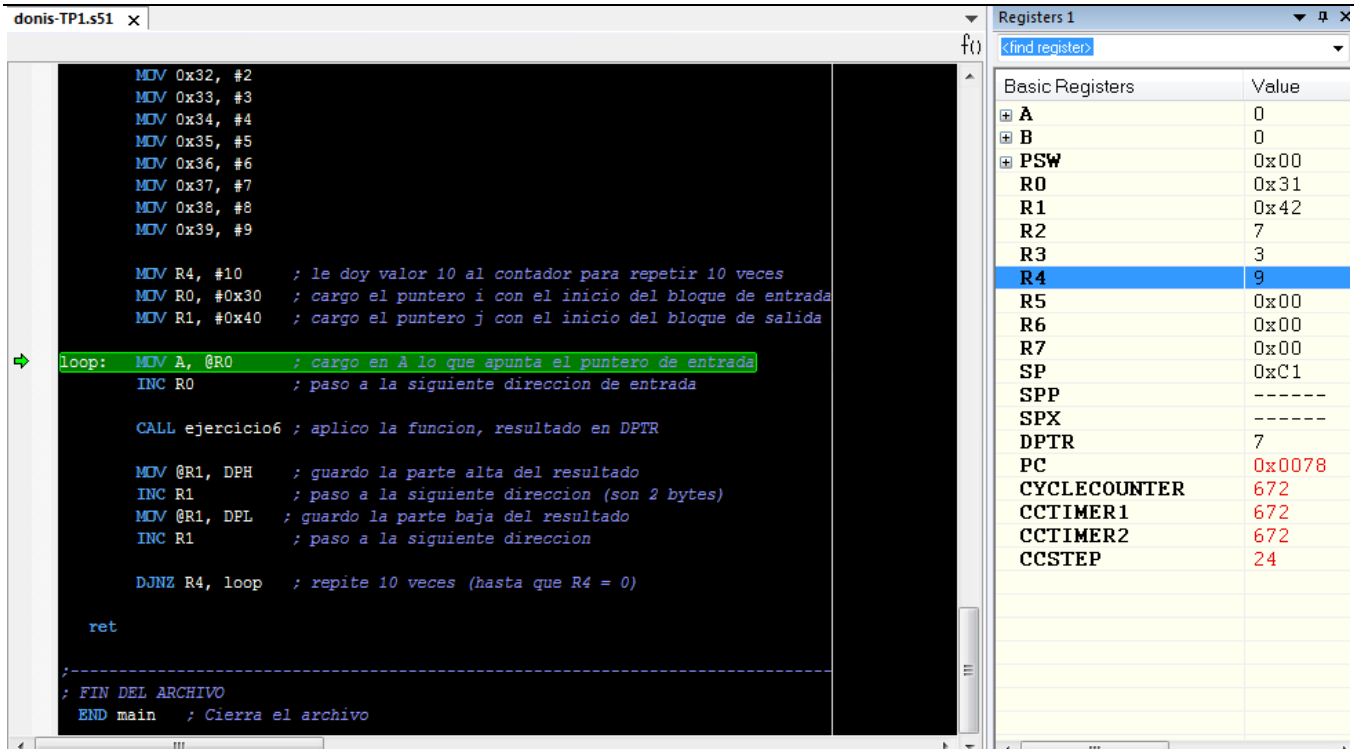


Se llama a la subrutina ejercicio6 que aplica la función f(a)=3a+7 y guarda el resultado en el DPTR.

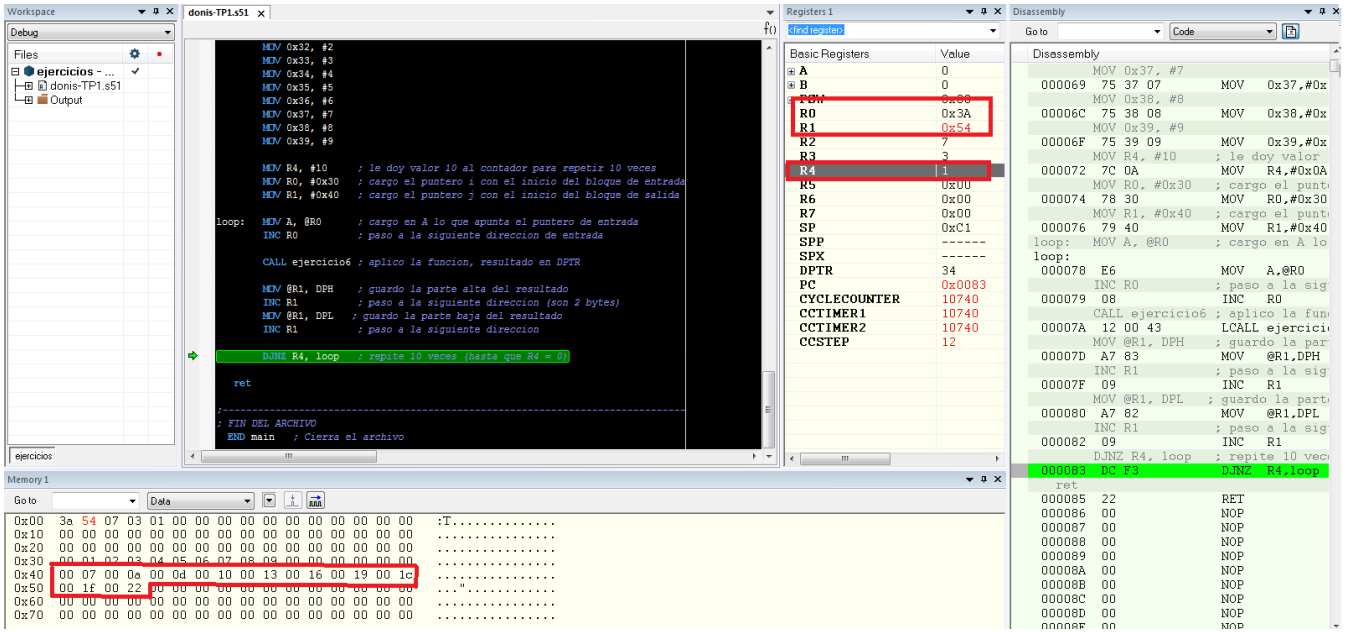


Al retornar de la subrutina el resultado está almacenado en el DPTR. Se guarda la parte alta en la dirección a la que apunta R1, luego se incrementa R1 en 1 y se guarda la parte baja en esa dirección. Esto es debido a que el resultado es en dos bytes.

Finalmente se incrementa R1 en 1 una vez más para almacenar el resultado de la siguiente iteración y DJNZ analiza si se debe volver a saltar a la etiqueta loop según el valor en R4.



En el caso de que R4 > 0, se salta a la etiqueta loop y se repite el proceso con las siguientes direcciones de entrada y salida.



Luego de 10 iteraciones, R4 quedará en 1. Al correr la instrucción DJNZ, se decrementa R4 en 1, quedando en 0, y se pasa a la siguiente instrucción en vez de volver a saltar al loop. Se puede observar que todos los resultados fueron almacenados en las direcciones 40h – 54h (20 posiciones, ya que son 10 resultados de 2 bytes).

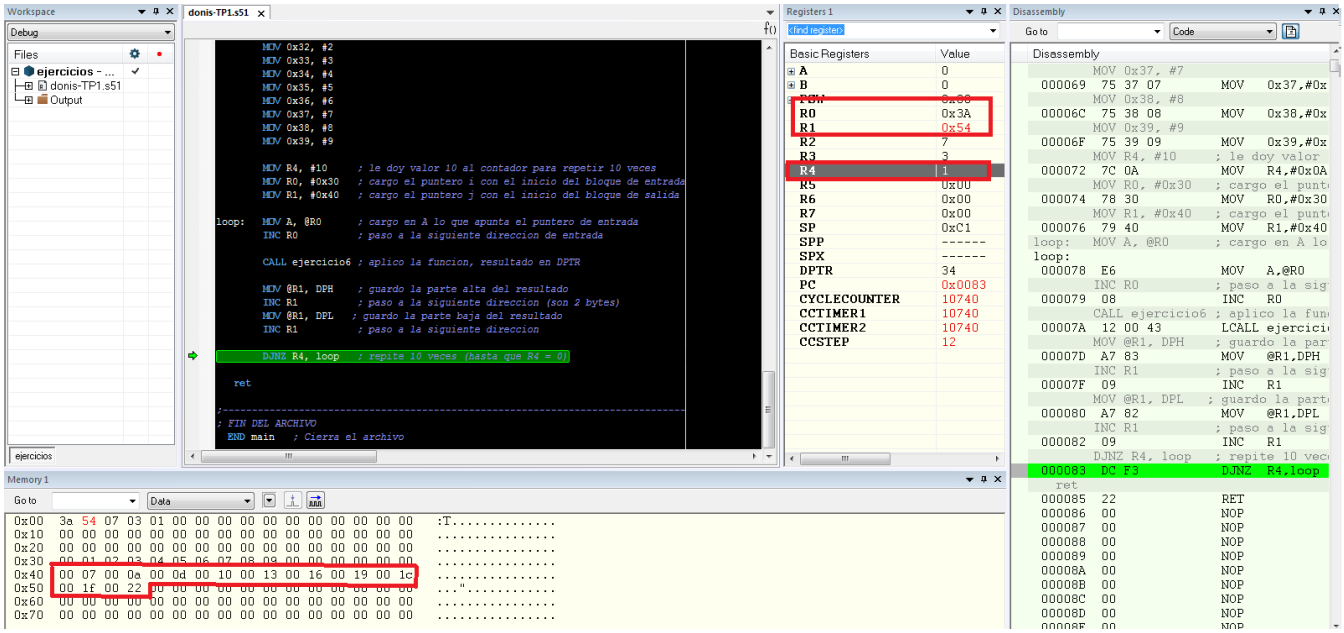
El código fuente de los ejercicios 6 y 14 se encuentra dentro del archivo donis-TP1.s51.

Pruebas y ensayos

Ya se había probado anteriormente a resolver este ejercicio el funcionamiento de la subrutina ejercicio6 dándole valor a A y verificando que efectivamente la función guardaba el valor correcto en el DPTR.

Para probar el código de este ejercicio, se inicializó el bloque de posiciones de entrada con números del 0d al 9d para probar el funcionamiento de la subrutina ejercicio14.

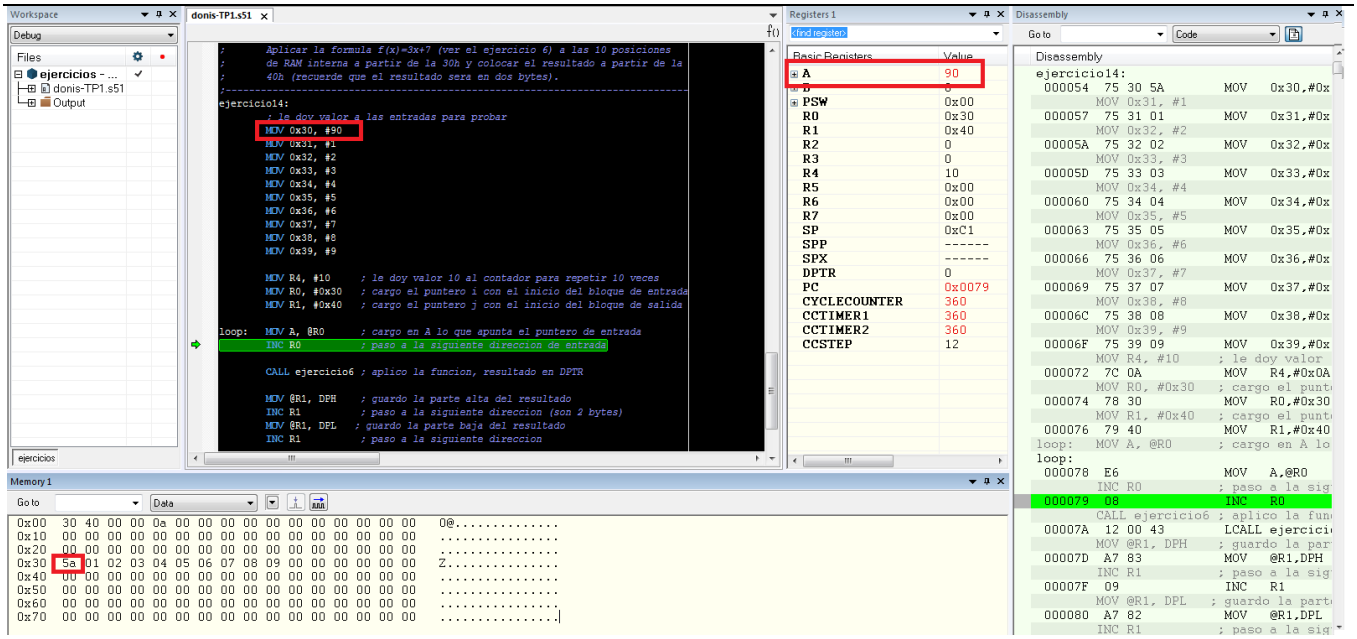
Luego de que el loop se ejecutara 10 veces, se verificó que los valores almacenados eran correctos:



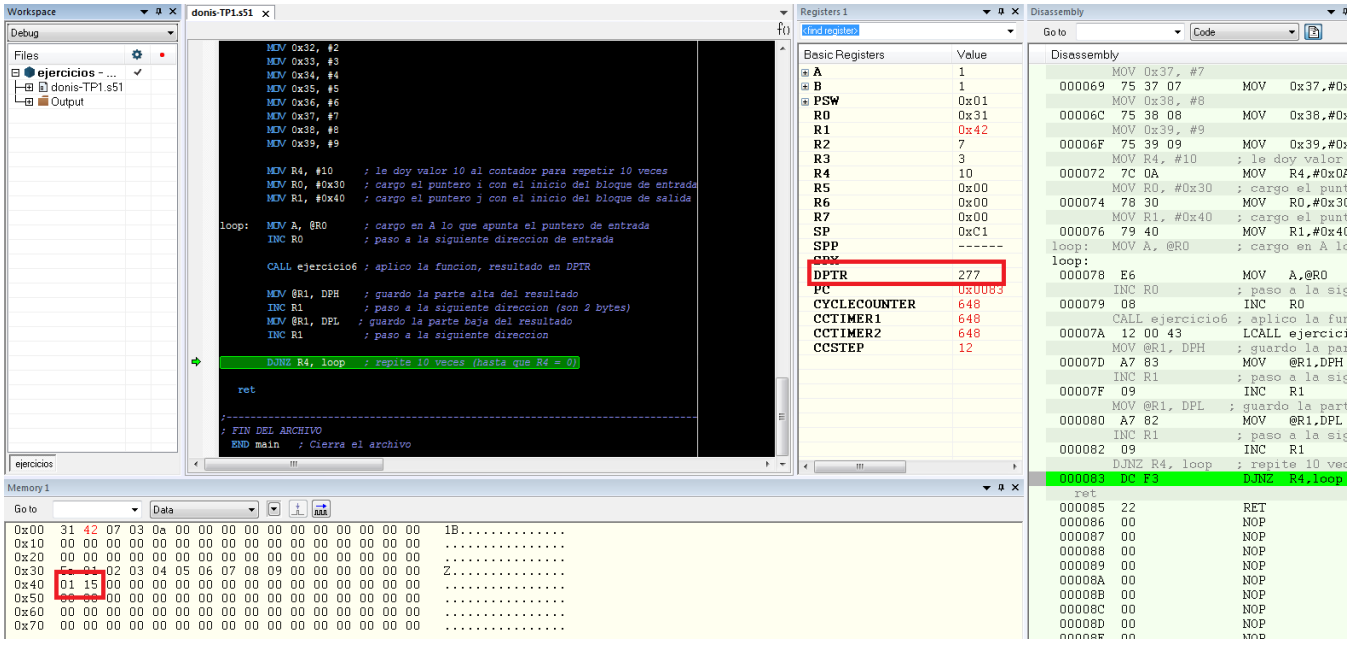
Por ejemplo:

- En la segunda iteración, x=1. Por ende, f(1)=1*3+7=10d. Se puede ver que en las posiciones 42h – 43h se almacenó este valor, Ah.
- En la última iteración, x=9. Por ende, f(9)=9*3+7=34d. Se puede ver que en las posiciones 53h – 54h se almacenó este valor, 22h.

Finalmente se realizó una prueba con un resultado mayor a FFh para verificar que la parte alta del resultado se guarda correctamente. Para esto, se le dio a 30h el valor 90d.



Luego de realizar la primera iteración se verifica que se almacena correctamente el valor tomado del DPTR. En este caso x=90 y f(90)=90*3+7=277d, y se ve almacenado este valor en las posiciones 40h – 41h, 115h.



Conclusiones

La resolución de la guía de ejercicios me ayudó a conocer mejor el set de instrucciones ofrecido por el fabricante y poner en práctica lo visto en clases teóricas. Personalmente ya conocía varias instrucciones de assembler, pero nunca las había puesto en práctica ni había visto el set de instrucciones de ningún microcontrolador en particular ya que habíamos realizado la codificación en papel, a mano.

Tener el template de código me resultó muy útil a la hora de resolver los ejercicios y me permitió enfocarme más en resolverlos. La clase en la que se nos explicó todo lo que contenía el template me ayudó a entender qué es lo que se inicializaba al correr el programa.

Un IDE siempre es una buena herramienta para desarrollar y probar código, y me fue fácil acostumbrarme a utilizar IAR Workbench ya que es bastante intuitivo y también varias de sus funcionalidades fueron mostradas en clase.