



EJERCITO ARGENTINO
ESCUELA SUPERIOR TECNICA
“Gr1 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 Embedded Workbench para 8051.

Documentación utilizada

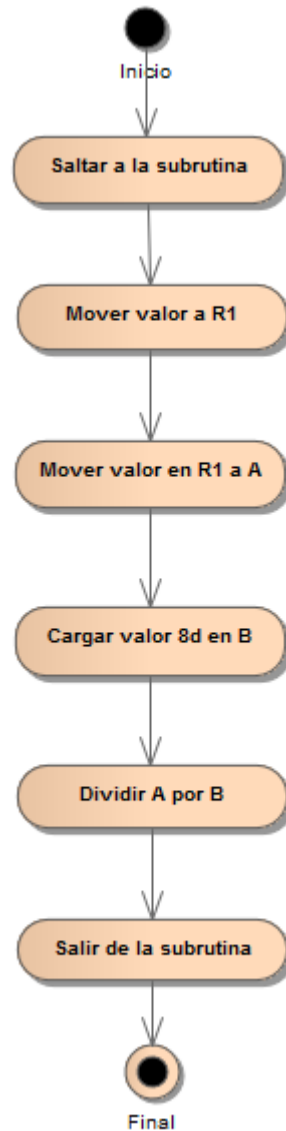
Manual de arquitectura y guía de programación de la familia de microcontroladores 80C51.

Ejercicio 5

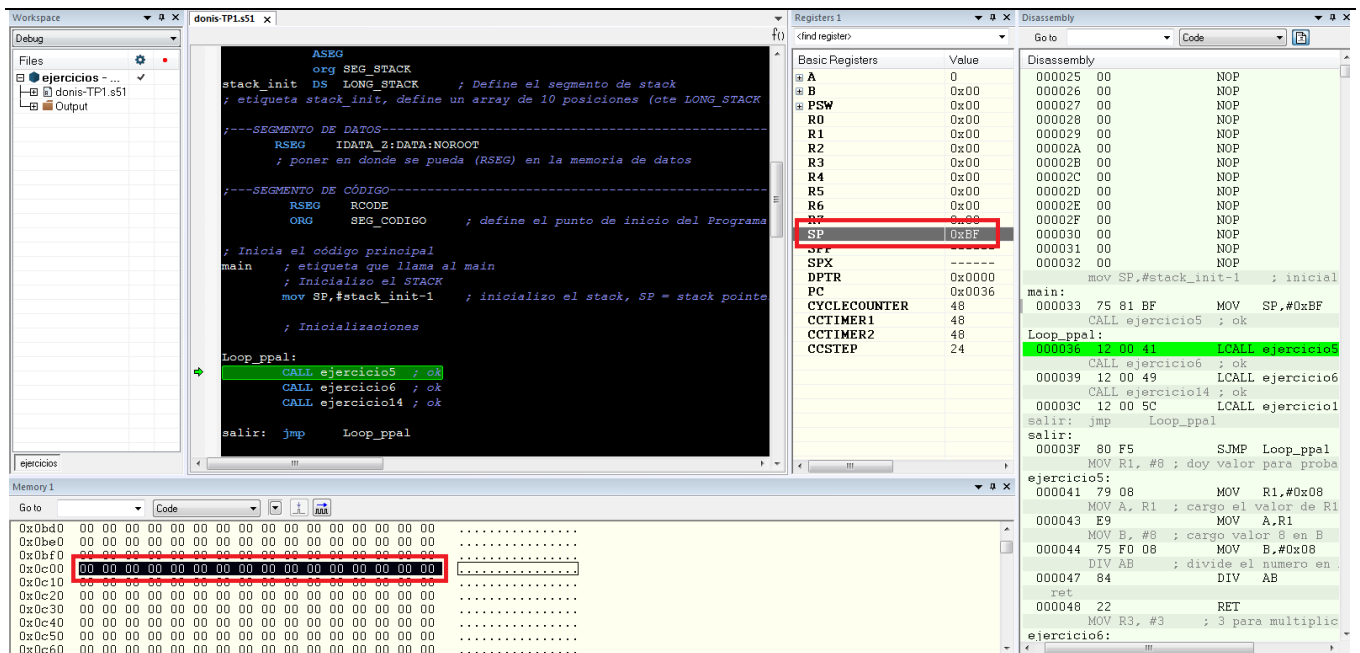
Enunciado

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

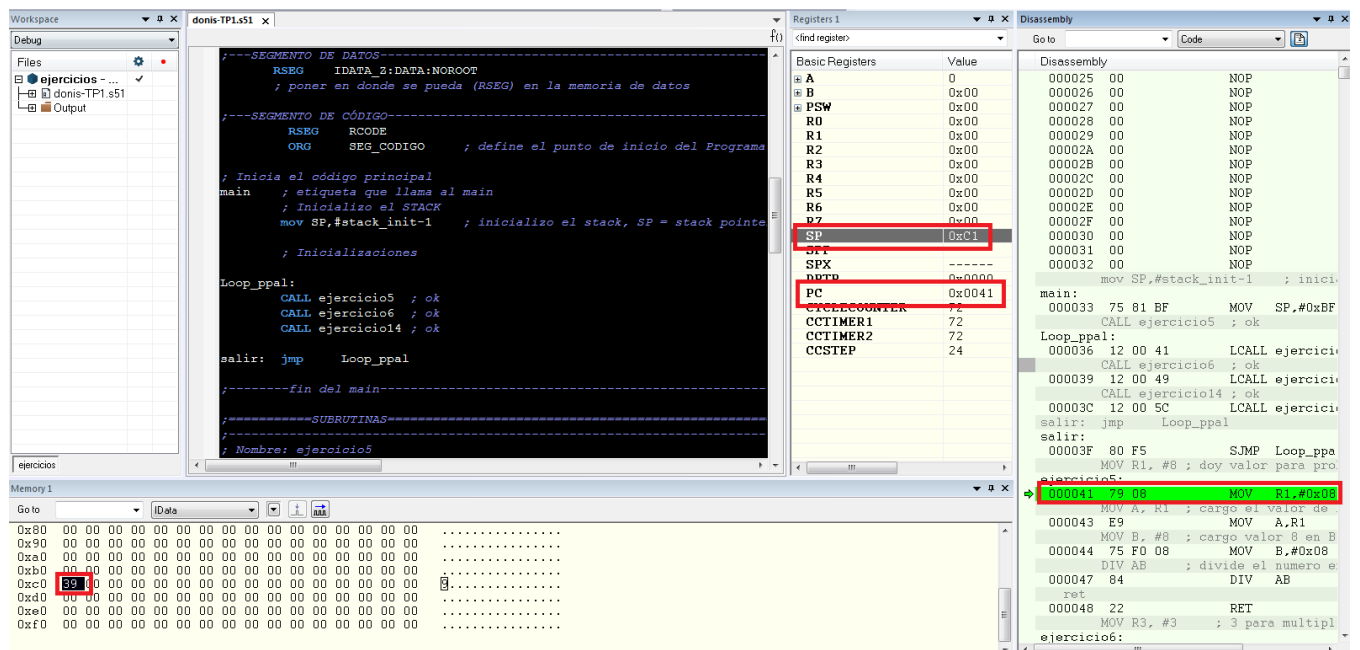


Al principio del código se define que el segmento de stack se inicializará en la posición C0h (SEG_STACK) y que tiene una longitud de 10 bytes (LONG_STACK). Al iniciar el main, la primera instrucción inicializa el puntero con la dirección anterior a la primera del segmento (stack_init-1 = BFh). Esto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



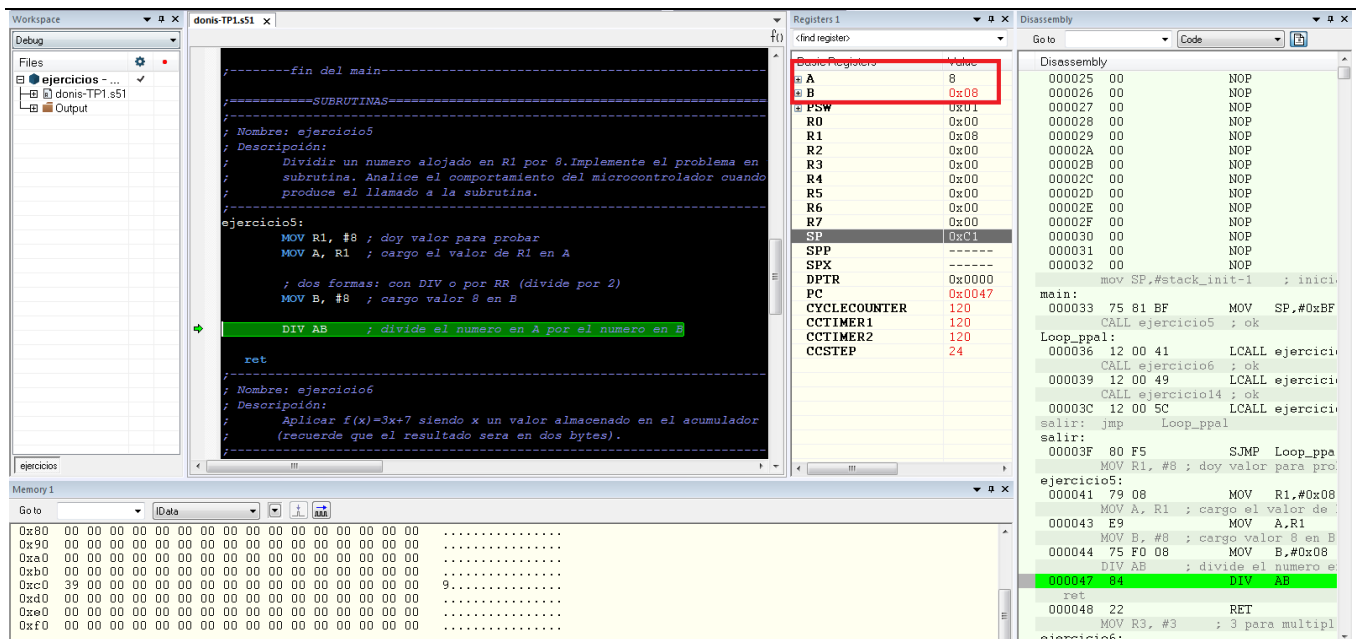
Cuando se realiza una CALL a la subrutina ejercicio5, en la posición apuntada por el SP se almacena la dirección de retorno. Esto es porque luego de correr la subrutina ejercicio5 se debe volver a la dirección 39h, que es la que contiene la siguiente instrucción.

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

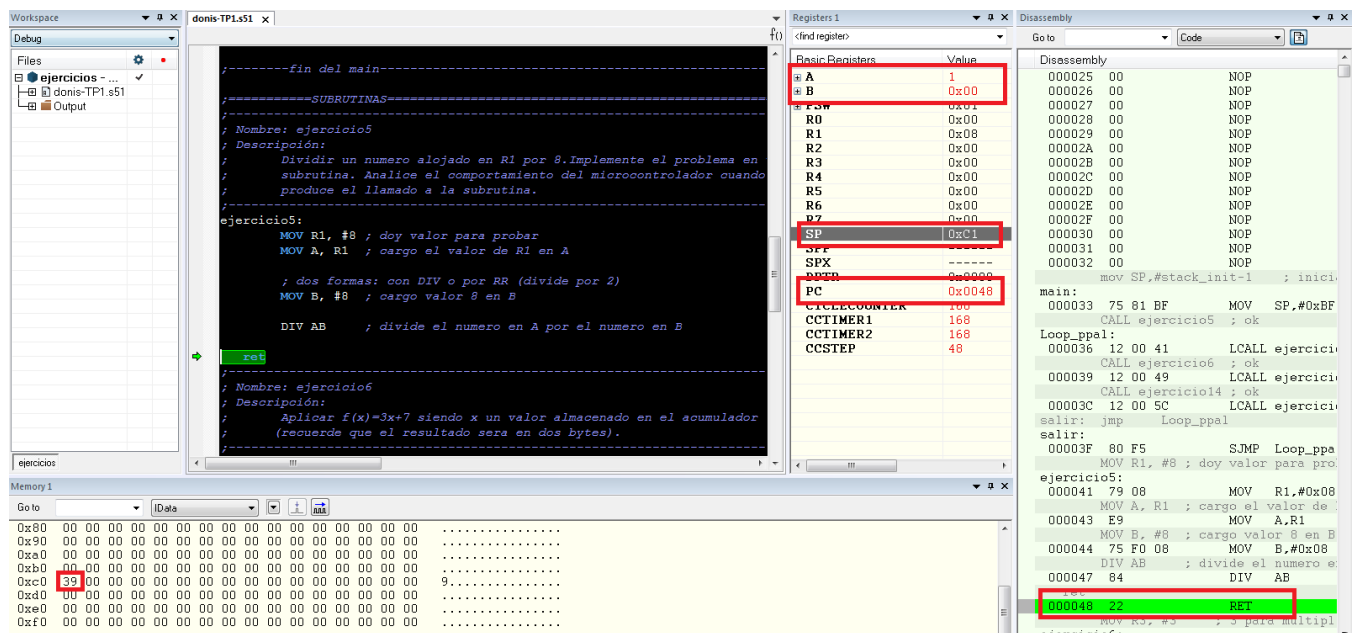


El CALL hace que el PC salte al inicio de la subrutina ejercicio5 en la posición 41h.

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.

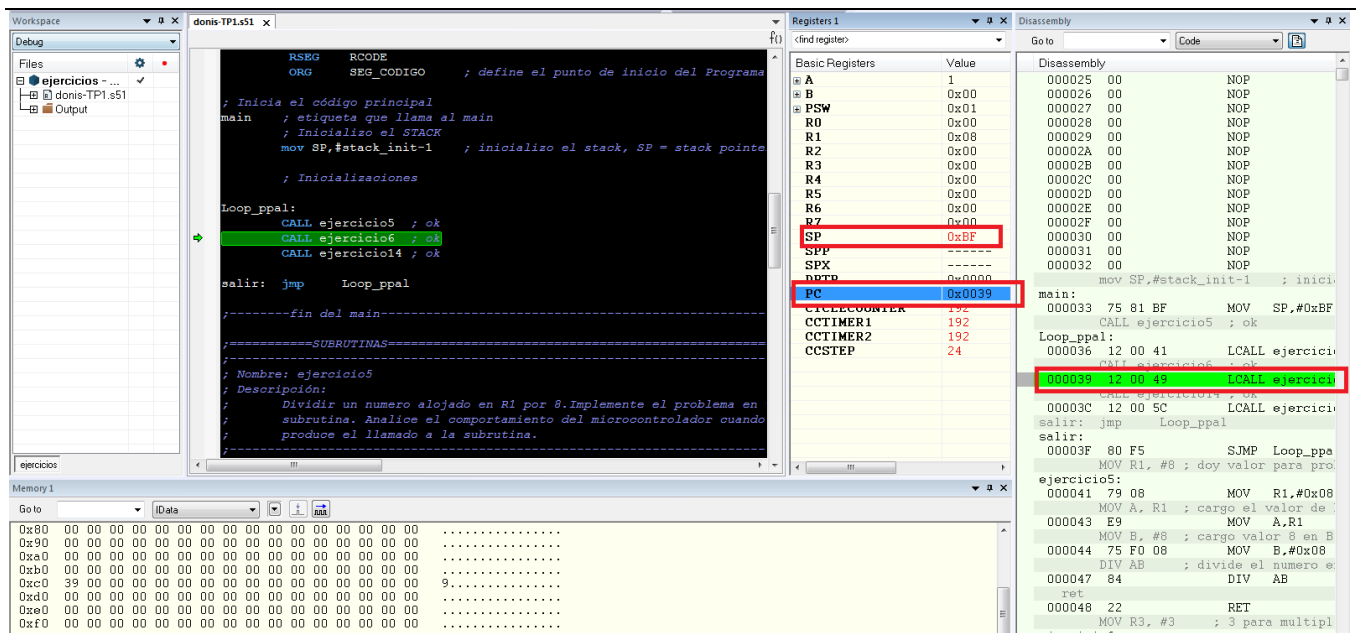


Luego de esto se corren las siguientes 3 instrucciones, que le dan valor al dividendo $A = R1$ y el divisor $B = 8$. En este caso, se almacena el valor de 8d en R1 para pruebas.



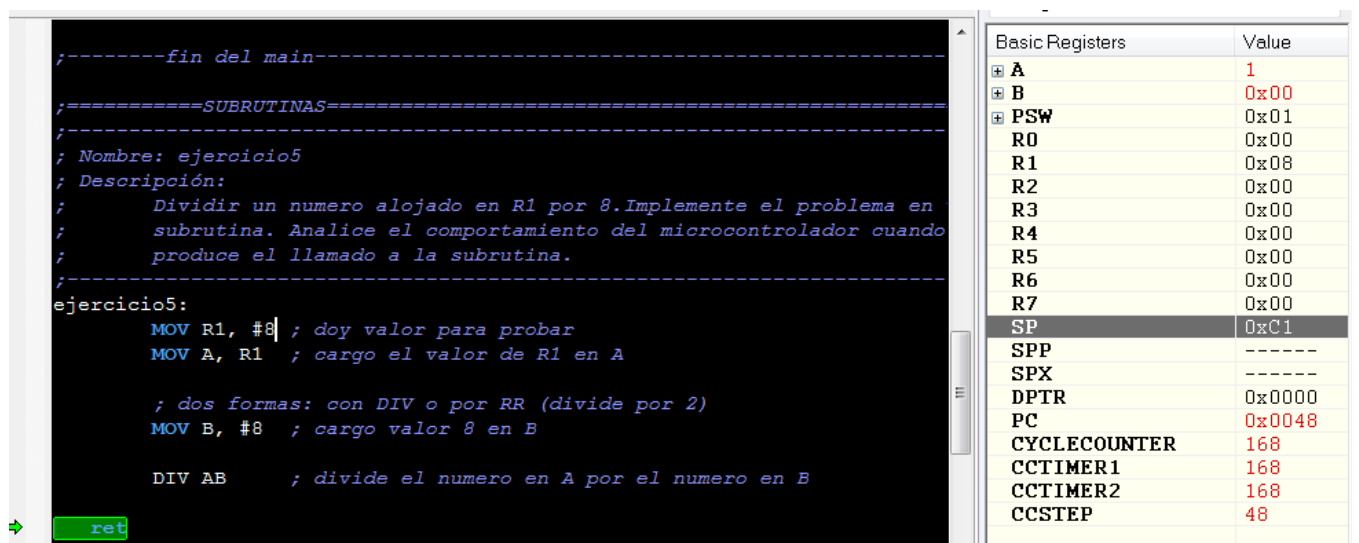
Luego de realizar la operación de división se puede ver que se actualizaron los valores de A y B con el resultado. A tiene el resultado entero y B el módulo. Al llegar a la última instrucción de la subrutina, RET, con el PC apuntando a 48h, se sale de la subrutina para volver al punto de retorno.

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 decreuenta 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-----
;=====SUBROUTINAS=====
;
; 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.

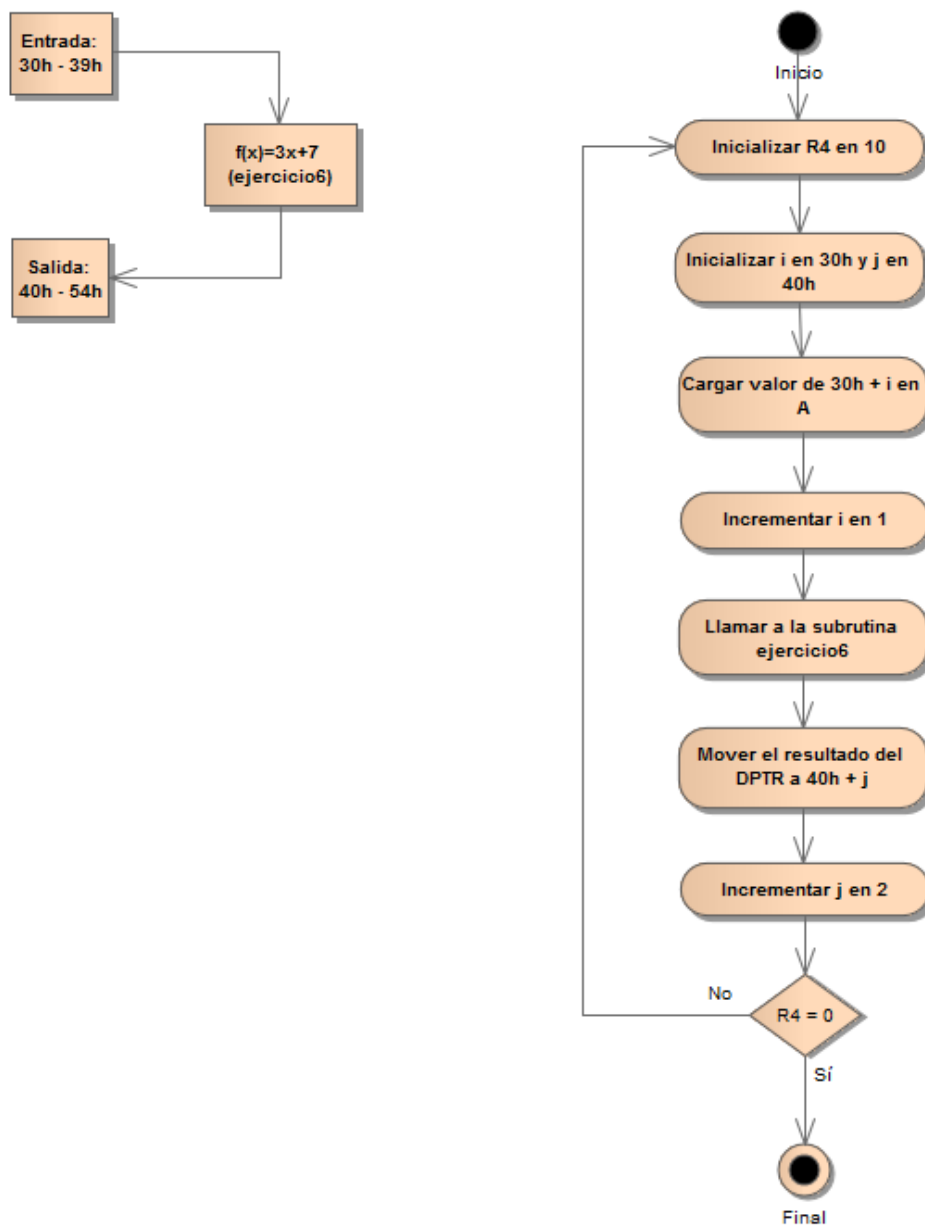
Ejercicio 14

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.



The screenshot shows the donis-TP1.s51 IDE with the following assembly code in the main window:

```

MOV DPH, A ; parte alta de mul + carry -> DPH
; queda el resultado en DPTR

ret

; Nombre: ejercicio14
; Descripción:
; Aplicar la formula f(x)=3x+7 (ver el ejercicio 6) a las 10 posiciones
; de RAM internas a partir de la 30h y colocar el resultado a partir de la
; 40h (recuerde que el resultado sera en dos bytes).

ejercicio14:
; le doy valor a las entradas para probar
MOV R4, #10
MOV R0, #0x30
MOV R1, #0x40
loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada
; le doy valor a las entradas para probar
MOV R4, #10
MOV R0, #0x30
MOV R1, #0x40
loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada

```

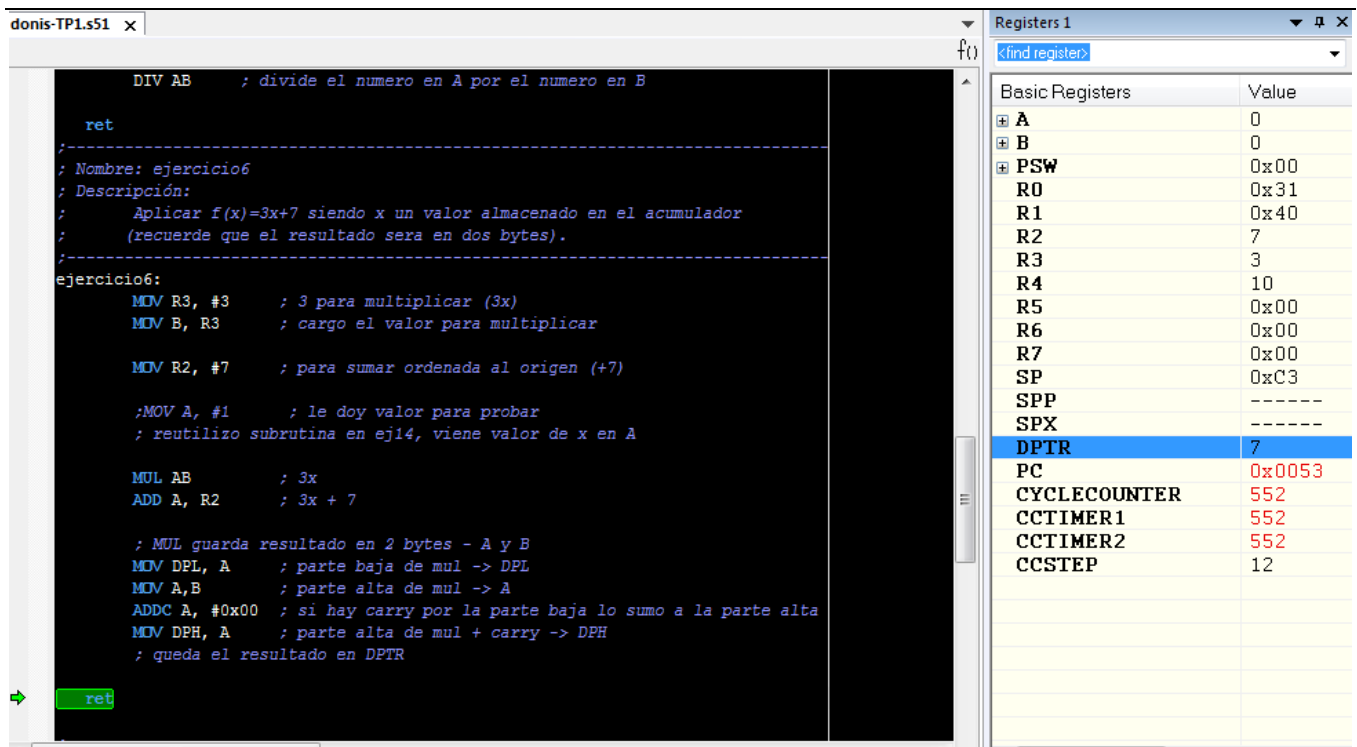
The registers window shows the following values:

Register	Value
A	0
B	0x00
PSW	0
R0	0x00
R1	0x00
R2	0x00
R3	0x00
R4	10
R5	0x00
R6	0x00
R7	0x00
SP	0xC1
SPP	-----
SPX	-----
DPTR	0x0000
PC	0x0072
CYCLECOUNTER	312
CCTIMER1	312
CCTIMER2	312
CCSTEP	24

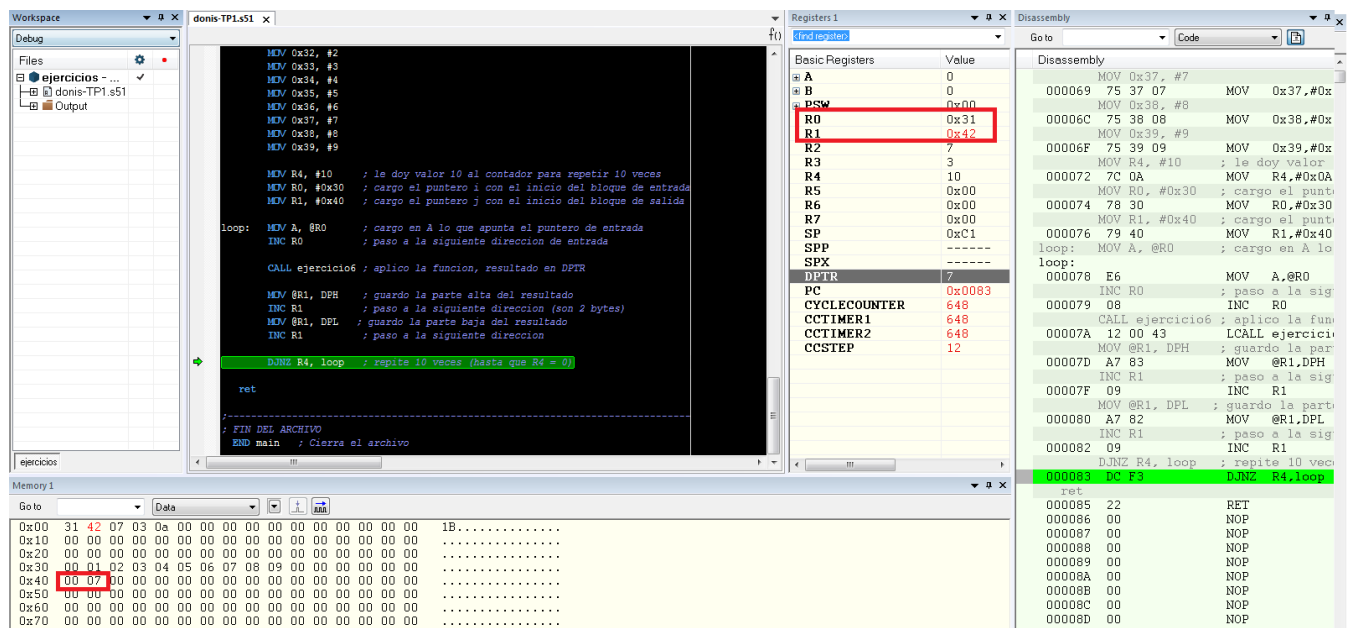
The disassembly window shows the following code:

```

000054 75 30 00 MOV 0x30,0x0
000057 75 31 01 MOV 0x31,0x0
00005A 75 32 02 MOV 0x32,0x0
00005D 75 33 03 MOV 0x33,0x0
000060 75 34 04 MOV 0x34,0x0
000063 75 35 05 MOV 0x35,0x0
000066 75 36 06 MOV 0x36,0x0
000069 75 37 07 MOV 0x37,0x0
00006C 75 38 08 MOV 0x38,0x0
00006F 75 39 09 MOV 0x39,0x0
000072 7C 0A MOV R4,0x0A
000074 78 30 MOV R0,0x30
000077 79 40 MOV R1,0x40
00007A 75 30 00 MOV 0x30,0x0
00007D 75 31 01 MOV 0x31,0x0
000080 75 32 02 MOV 0x32,0x0
000083 75 33 03 MOV 0x33,0x0
000086 75 34 04 MOV 0x34,0x0
000089 75 35 05 MOV 0x35,0x0
00008C 75 36 06 MOV 0x36,0x0
00008F 75 37 07 MOV 0x37,0x0
000092 75 38 08 MOV 0x38,0x0
000095 75 39 09 MOV 0x39,0x0
000098 7C 0A MOV R4,0x0A
00009B 78 30 MOV R0,0x30
00009E 79 40 MOV R1,0x40
0000A1 75 30 00 MOV 0x30,0x0
0000A4 75 31 01 MOV 0x31,0x0
0000A7 75 32 02 MOV 0x32,0x0
0000AA 75 33 03 MOV 0x33,0x0
0000AD 75 34 04 MOV 0x34,0x0
0000B0 75 35 05 MOV 0x35,0x0
0000B3 75 36 06 MOV 0x36,0x0
0000B6 75 37 07 MOV 0x37,0x0
0000B9 75 38 08 MOV 0x38,0x0
0000BC 75 39 09 MOV 0x39,0x0
0000BF 7C 0A MOV R4,0x0A
0000C2 78 30 MOV R0,0x30
0000C5 79 40 MOV R1,0x40
0000C8 75 30 00 MOV 0x30,0x0
0000CB 75 31 01 MOV 0x31,0x0
0000CE 75 32 02 MOV 0x32,0x0
0000D1 75 33 03 MOV 0x33,0x0
0000D4 75 34 04 MOV 0x34,0x0
0000D7 75 35 05 MOV 0x35,0x0
0000DA 75 36 06 MOV 0x36,0x0
0000DD 75 37 07 MOV 0x37,0x0
0000E0 75 38 08 MOV 0x38,0x0
0000E3 75 39 09 MOV 0x39,0x0
0000E6 7C 0A MOV R4,0x0A
0000E9 78 30 MOV R0,0x30
0000EC 79 40 MOV R1,0x40
0000EF 75 30 00 MOV 0x30,0x0
0000F2 75 31 01 MOV 0x31,0x0
0000F5 75 32 02 MOV 0x32,0x0
0000F8 75 33 03 MOV 0x33,0x0
0000FB 75 34 04 MOV 0x34,0x0
0000FE 75 35 05 MOV 0x35,0x0
000101 75 36 06 MOV 0x36,0x0
000104 75 37 07 MOV 0x37,0x0
000107 75 38 08 MOV 0x38,0x0
00010A 75 39 09 MOV 0x39,0x0
00010D 7C 0A MOV R4,0x0A
000110 78 30 MOV R0,0x30
000113 79 40 MOV R1,0x40
000116 75 30 00 MOV 0x30,0x0
000119 75 31 01 MOV 0x31,0x0
00011C 75 32 02 MOV 0x32,0x0
00011F 75 33 03 MOV 0x33,0x0
000122 75 34 04 MOV 0x34,0x0
000125 75 35 05 MOV 0x35,0x0
000128 75 36 06 MOV 0x36,0x0
00012B 75 37 07 MOV 0x37,0x0
00012E 75 38 08 MOV 0x38,0x0
000131 75 39 09 MOV 0x39,0x0
000134 7C 0A MOV R4,0x0A
000137 78 30 MOV R0,0x30
00013A 79 40 MOV R1,0x40
00013D 75 30 00 MOV 0x30,0x0
000140 75 31 01 MOV 0x31,0x0
000143 75 32 02 MOV 0x32,0x0
000146 75 33 03 MOV 0x33,0x0
000149 75 34 04 MOV 0x34,0x0
00014C 75 35 05 MOV 0x35,0x0
00014F 75 36 06 MOV 0x36,0x0
000152 75 37 07 MOV 0x37,0x0
000155 75 38 08 MOV 0x38,0x0
000158 75 39 09 MOV 0x39,0x0
00015B 7C 0A MOV R4,0x0A
00015E 78 30 MOV R0,0x30
000161 79 40 MOV R1,0x40
000164 75 30 00 MOV 0x30,0x0
000167 75 31 01 MOV 0x31,0x0
00016A 75 32 02 MOV 0x32,0x0
00016D 75 33 03 MOV 0x33,0x0
000170 75 34 04 MOV 0x34,0x0
000173 75 35 05 MOV 0x35,0x0
000176 75 36 06 MOV 0x36,0x0
000179 75 37 07 MOV 0x37,0x0
00017C 75 38 08 MOV 0x38,0x0
00017F 75 39 09 MOV 0x39,0x0
000182 7C 0A MOV R4,0x0A
000185 78 30 MOV R0,0x30
000188 79 40 MOV R1,0x40
00018B 75 30 00 MOV 0x30,0x0
00018E 75 31 01 MOV 0x31,0x0
000191 75 32 02 MOV 0x32,0x0
000194 75 33 03 MOV 0x33,0x0
000197 75 34 04 MOV 0x34,0x0
00019A 75 35 05 MOV 0x35,0x0
00019D 75 36 06 MOV 0x36,0x0
0001A0 75 37 07 MOV 0x37,0x0
0001A3 75 38 08 MOV 0x38,0x0
0001A6 75 39 09 MOV 0x39,0x0
0001A9 7C 0A MOV R4,0x0A
0001AC 78 30 MOV R0,0x30
0001AF 79 40 MOV R1,0x40
0001B2 75 30 00 MOV 0x30,0x0
0001B5 75 31 01 MOV 0x31,0x0
0001B8 75 32 02 MOV 0x32,0x0
0001BB 75 33 03 MOV 0x33,0x0
0001BE 75 34 04 MOV 0x34,0x0
0001C1 75 35 05 MOV 0x35,0x0
0001C4 75 36 06 MOV 0x36,0x0
0001C7 75 37 07 MOV 0x37,0x0
0001CA 75 38 08 MOV 0x38,0x0
0001CD 75 39 09 MOV 0x39,0x0
0001C8 7C 0A MOV R4,0x0A
0001CB 78 30 MOV R0,0x30
0001CE 79 40 MOV R1,0x40
0001D1 75 30 00 MOV 0x30,0x0
0001D4 75 31 01 MOV 0x31,0x0
0001D7 75 32 02 MOV 0x32,0x0
0001DA 75 33 03 MOV 0x33,0x0
0001DD 75 34 04 MOV 0x34,0x0
0001E0 75 35 05 MOV 0x35,0x0
0001E3 75 36 06 MOV 0x36,0x0
0001E6 75 37 07 MOV 0x37,0x0
0001E9 75 38 08 MOV 0x38,0x0
0001EC 75 39 09 MOV 0x39,0x0
0001EF 7C 0A MOV R4,0x0A
0001F2 78 30 MOV R0,0x30
0001F5 79 40 MOV R1,0x40
0001F8 75 30 00 MOV 0x30,0x0
0001FB 75 31 01 MOV 0x31,0x0
0001FE 75 32 02 MOV 0x32,0x0
000201 75 33 03 MOV 0x33,0x0
000204 75 34 04 MOV 0x34,0x0
000207 75 35 05 MOV 0x35,0x0
00020A 75 36 06 MOV 0x36,0x0
00020D 75 37 07 MOV 0x37,0x0
000210 75 38 08 MOV 0x38,0x0
000213 75 39 09 MOV 0x39,0x0
000216 7C 0A MOV R4,0x0A
000219 78 30 MOV R0,0x30
00021C 79 40 MOV R1,0x40
00021F 75 30 00 MOV 0x30,0x0
000222 75 31 01 MOV 0x31,0x0
000225 75 32 02 MOV 0x32,0x0
000228 75 33 03 MOV 0x33,0x0
00022B 75 34 04 MOV 0x34,0x0
00022E 75 35 05 MOV 0x35,0x0
000231 75 36 06 MOV 0x36,0x0
000234 75 37 07 MOV 0x37,0x0
000237 75 38 08 MOV 0x38,0x0
00023A 75 39 09 MOV 0x39,0x0
00023D 7C 0A MOV R4,0x0A
000240 78 30 MOV R0,0x30
000243 79 40 MOV R1,0x40
000246 75 30 00 MOV 0x30,0x0
000249 75 31 01 MOV 0x31,0x0
00024C 75 32 02 MOV 0x32,0x0
00024F 75 33 03 MOV 0x33,0x0
000252 75 34 04 MOV 0x34,0x0
000255 75 35 05 MOV 0x35,0x0
000258 75 36 06 MOV 0x36,0x0
00025B 75 37 07 MOV 0x37,0x0
00025E 75 38 08 MOV 0x38,0x0
000261 75 39 09 MOV 0x39,0x0
000264 7C 0A MOV R4,0x0A
000267 78 30 MOV R0,0x30
00026A 79 40 MOV R1,0x40
00026D 75 30 00 MOV 0x30,0x0
000270 75 31 01 MOV 0x31,0x0
000273 75 32 02 MOV 0x32,0x0
000276 75 33 03 MOV 0x33,0x0
000279 75 34 04 MOV 0x34,0x0
00027C 75 35 05 MOV 0x35,0x0
00027F 75 36 06 MOV 0x36,0x0
000282 75 37 07 MOV 0x37,0x0
000285 75 38 08 MOV 0x38,0x0
000288 75 39 09 MOV 0x39,0x0
00028B 7C 0A MOV R4,0x0A
00028E 78 30 MOV R0,0x30
000291 79 40 MOV R1,0x40
000294 75 30 00 MOV 0x30,0x0
000297 75 31 01 MOV 0x31,0x0
00029A 75 32 02 MOV 0x32,0x0
00029D 75 33 03 MOV 0x33,0x0
0002A0 75 34 04 MOV 0x34,0x0
0002A3 75 35 05 MOV 0x35,0x0
0002A6 75 36 06 MOV 0x36,0x0
0002A9 75 37 07 MOV 0x37,0x0
0002AC 75 38 08 MOV 0x38,0x0
0002AF 75 39 09 MOV 0x39,0x0
0002B2 7C 0A MOV R4,0x0A
0002B5 78 30 MOV R0,0x30
0002B8 79 40 MOV R1,0x40
0002BB 75 30 00 MOV 0x30,0x0
0002BE 75 31 01 MOV 0x31,0x0
0002C1 75 32 02 MOV 0x32,0x0
0002C4 75 33 03 MOV 0x33,0x0
0002C7 75 34 04 MOV 0x34,0x0
0002CA 75 35 05 MOV 0x35,0x0
0002CD 75 36 06 MOV 0x36,0x0
0002CF 75 37 07 MOV 0x37,0x0
0002D2 75 38 08 MOV 0x38,0x0
0002D5 75 39 09 MOV 0x39,0x0
0002D8 7C 0A MOV R4,0x0A
0002DB 78 30 MOV R0,0x30
0002DE 79 40 MOV R1,0x40
0002E1 75 30 00 MOV 0x30,0x0
0002E4 75 31 01 MOV 0x31,0x0
0002E7 75 32 02 MOV 0x32,0x0
0002EA 75 33 03 MOV 0x33,0x0
0002ED 75 34 04 MOV 0x34,0x0
0002F0 75 35 05 MOV 0x35,0x0
0002F3 75 36 06 MOV 0x36,0x0
0002F6 75 37 07 MOV 0x37,0x0
0002F9 75 38 08 MOV 0x38,0x0
0002FC 75 39 09 MOV 0x39,0x0
0002FF 7C 0A MOV R4,0x0A
000302 78 30 MOV R0,0x30
000305 79 40 MOV R1,0x40
000308 75 30 00 MOV 0x30,0x0
00030B 75 31 01 MOV 0x31,0x0
00030E 75 32 02 MOV 0x32,0x0
000311 75 33 03 MOV 0x33,0x0
000314 75 34 04 MOV 0x34,0x0
000317 75 35 05 MOV 0x35,0x0
00031A 75 36 06 MOV 0x36,0x0
00031D 75 37 07 MOV 0x37,0x0
000320 75 38 08 MOV 0x38,0x0
000323 75 39 09 MOV 0x39,0x0
000326 7C 0A MOV R4,0x0A
000329 78 30 MOV R0,0x30
00032C 79 40 MOV R1,0x40
00032F 75 30 00 MOV 0x30,0x0
000332 75 31 01 MOV 0x31,0x0
000335 75 32 02 MOV 0x32,0x0
000338 75 33 03 MOV 0x33,0x0
00033B 75 34 04 MOV 0x34,0x0
00033E 75 35 05 MOV 0x35,0x0
000341 75 36 06 MOV 0x36,0x0
000344 75 37 07 MOV 0x37,0x0
000347 75 38 08 MOV 0x38,0x0
00034A 75 39 09 MOV 0x39,0x0
00034D 7C 0A MOV R4,0x0A
000350 78 30 MOV R0,0x30
000353 79 40 MOV R1,0x40
000356 75 30 00 MOV 0x30,0x0
000359 75 31 01 MOV 0x31,0x0
00035C 75 32 02 MOV 0x32,0x0
00035F 75 33 03 MOV 0x33,0x0
000362 75 34 04 MOV 0x34,0x0
000365 75 35 05 MOV 0x35,0x0
000368 75 36 06 MOV 0x36,0x0
00036B 75 37 07 MOV 0x37,0x0
00036E 75 38 08 MOV 0x38,0x0
000371 75 39 09 MOV 0x39,0x0
000374 7C 0A MOV R4,0x0A
000377 78 30 MOV R0,0x30
00037A 79 40 MOV R1,0x40
00037D 75 30 00 MOV 0x30,0x0
000380 75 31 01 MOV 0x31,0x0
000383 75 32 02 MOV 0x32,0x0
000386 75 33 03 MOV 0x33,0x0
000389 75 34 04 MOV 0x34,0x0
00038C 75 35 05 MOV 0x35,0x0
00038F 75 36 06 MOV 0x36,0x0
000392 75 37 07 MOV 0x37,0x0
000395 75 38 08 MOV 0x38,0x0
000398 75 39 09 MOV 0x39,0x0
00039B 7C 0A MOV R4,0x0A
00039E 78 30 MOV R0,0x30
0003A1 79 40 MOV R1,0x40
0003A4 75 30 00 MOV 0x30,0x0
0003A7 75 31 01 MOV 0x31,0x0
0003AA 75 32 02 MOV 0x32,0x0
0003AD 75 33 03 MOV 0x33,0x0
0003B0 75 34 04 MOV 0x34,0x0
0003B3 75 35 05 MOV 0x35,0x0
0003B6 75 36 06 MOV 0x36,0x0
0003B9 75 37 07 MOV 0x37,0x0
0003BC 75 38 08 MOV 0x38,0x0
0003BF 75 39 09 MOV 0x39,0x0
0003C2 7C 0A MOV R4,0x0A
0003C5 78 30 MOV R0,0x30
0003C8 79 40 MOV R1,0x40
0003CB 75 30 00 MOV 0x30,0x0
0003CE 75 31 01 MOV 0x31,0x0
0003D1 75 32 02 MOV 0x32,0x0
0003D4 75 33 03 MOV 0x33,0x0
0003D7 75 34 04 MOV 0x34,0x0
0003DA 75 35 05 MOV 0x35,0x0
0003DD 75 36 06 MOV 0x36,0x0
0003E0 75 37 07 MOV 0x37,0x0
0003E3 75 38 08 MOV 0x38,0x0
0003E6 75 39 09 MOV 0x39,0x0
0003E9 7C 0A MOV R4,0x0A
0003EC 78 30 MOV R0,0x30
0003EF 79 40 MOV R1,0x40
0003F2 75 30 00 MOV 0x30,0x0
0003F5 75 31 01 MOV 0x31,0x0
0003F8 75 32 02 MOV 0x32,0x0
0003FB 75 33 03 MOV 0x33,0x0
0003FE 75 34 04 MOV 0x34,0x0
000401 75 35 05 MOV 0x35,0x0
000404 75 36 06 MOV 0x36,0x0
000407 75 37 07 MOV 0x37,0x0
00040A 75 38 08 MOV 0x38,0x0
00040D 75 39 09 MOV 0x39,0x0
000410 7C 0A MOV R4,0x0A
000413 78 30 MOV R0,0x30
000416 79 40 MOV R1,0x40
000419 75 30 00 MOV 0x30,0x0
00041C 75 31 01 MOV 0x31,0x0
00041F 75 32 02 MOV 0x32,0x0
000422 75 33 03 MOV 0x33,0x0
000425 75 34 04 MOV 0x34,0x0
000428 75 35 05 MOV 0x35,0x0
00042B 75 36 06 MOV 0x36,0x0
00042E 75 37 07 MOV 0x37,0x0
000431 75 38 08 MOV 0x38,0x0
000434 75 39 09 MOV 0x39,0x0
000437 7C 0A MOV R4,0x0A
00043A 78 30 MOV R0,0x30
00043D 79 40 MOV R1,0x40
000440 75 30 00 MOV 0x30,0x0
000443 75 31 01 MOV 0x31,0x0
000446 75 32 02 MOV 0x32,0x0
000449 75 33 03 MOV 0x33,0x0
00044C 75 34 04 MOV 0x34,0x0
00044F 75 35 05 MOV 0x35,0x0
000452 75 36 06 MOV 0x36,0x0
000455 75 37 07 MOV 0x37,0x0
000458 75 38 08 MOV 0x38,0x0
00045B 75 39 09 MOV 0x39,0x0
00045E 7C 0A MOV R4,0x0A
000461 78 30 MOV R0,0x30
000464 79 40 MOV R1,0x40
000467 75 30 00 MOV 0x30,0x0
00046A 75 31 01 MOV 0x31,0x0
00046D 75 32 02 MOV 0x32,0x0
000470 75 33 03 MOV 0x33,0x0
000473 75 34 04 MOV 0x34,0x0
000476 75 35 05 MOV 0x35,0x0
000479 75 36 06 MOV 0x36,0x0
00047C 75 37 07 MOV 0x37,0x0
00047F 75 38 08 MOV 0x38,0x0
000482 75 39 09 MOV 0x39,0x0
000485 7C 0A MOV R4,0x0A
000488 78 30 MOV R0,0x30
00048B 79 40 MOV R1,0x40
00048E 75 30 00 MOV 0x30,0x0
000491 75 31 01 MOV 0x31,0x0
000494 75 32 02 MOV 0x32,0x0
000497 75 33 03 MOV 0x33,0x0
00049A 75 34 04 MOV 0x34,0x0
00049D 75 35 05 MOV 0x35,0x0
0004A0 75 36 06 MOV 0x36,0x0
0004A3 75 37 07 MOV 0x37,0x0
0004A6 75 38 08 MOV 0x38,0x0
0004A9 75 39 09 MOV 0x39,0x0
0004AC 7C 0A MOV R4,0x0A
0004AF 78 30 MOV R0,0x30
0004B2 79 40 MOV R1,0x40
0004B5 75 30 00 MOV 0x30,0x0
0004B8 75 31 01 MOV 0x31,0x0
0004BB 75 32 02 MOV 0x32,0x0
0004BE 75 33 03 MOV 0x33,0x0
0004C1 75 34 04 MOV 0x34,0x0
0004C4 75 35 05 MOV 0x35,0x0
0004C7 75 36 06 MOV 0x36,0x0
0004CA 75 37 07 MOV 0x37,0x0
0004CD 75 38 08 MOV 0x38,0x0
0004CF 75 39 09 MOV 0x39,0x0
0004D2 7C 0A MOV R4,0x0A
0004D5 78 30 MOV R0,0x30
0004D8 79 40 MOV R1,0x40
0004DB 75 30 00 MOV 0x30,0x0
0004DE 75 31 01 MOV 0x31,0x0
0004E1 75 32 02 MOV 0x32,0x0
0004E4 75 33 03 MOV 0x33,0x0
0004E7 75 34 04 MOV 0x34,0x0
0004EA 75 35 05 MOV 0x35,0x0
0004ED 75 36 06 MOV 0x36,0x0
0004F0 75 37 07 MOV 0x37,0x0
0004F3 75 38 08 MOV 0x38,0x0
0004F6 75 39 09 MOV 0x39,0x0
0004F9 7C 0A MOV R4,0x0A
0004FC 78 30 MOV R0,0x30
0004FF 79 40 MOV R1,0x40
000502 75 30 00 MOV 0x30,0x0
000505 75 31 01 MOV 0x31,0x0
000508 75 32 02 MOV 0x32,0x0
00050B 75 33 03 MOV 0x33,0x0
00050E 75 34 04 MOV 0x34,0x0
000511 75 35 05 MOV 0x35,0x0
000514 75 36 06 MOV 0x36,0x0
000517 75 37 07 MOV 0x37,0x0
00051A 75 38 08 MOV 0x38,0x0
00051D 75 39 09 MOV 0x39,0x0
000520 7C 0A MOV R4,0x0A
000523 78 30 MOV R0,0x30
000526 79 40 MOV R1,0x40
000529 75 30 00 MOV 0x30,0x0
00052C 75 31 01 MOV 0x31,0x0
00052F 75 32 02 MOV 0x32,0x0
000532 75 33 03 MOV 0x33,0x0
000535 75 34 04 MOV 0x34,0x0
000538 75 35 05 MOV 0x35,0x0
00053B 75 36 06 MOV 0x36,0x0
00053E 75 37 07 MOV 0x37,0x0
000541 75 38 08 MOV 0x38,0x0
000544 75 39 09 MOV 0x39,0x0
000547 7C 0A MOV R4,0x0A
00054A 78 30 MOV R0,0x30
00054D 79 40 MOV R1,0x40
000550 75 30 00 MOV 0x30,0x0
000553 75 31 01 MOV 0x31,0x0
000556 75 32 02 MOV 0x32,0x0
000559 75 33 03 MOV 0x33,0x0
00055C 75 34 04 MOV 0x34,0x0
00055F 75 35 05 MOV 0x35,0x0
000562 75 36 06 MOV 0x36,0x0
000565 75 37 07 MOV 0x37,0x0
000568 75 38 08 MOV 0x38,0x0
00056B 75 39 09 MOV 0x39,0x0
00056E 7C 0A MOV R4,0x0A
000571 78 30 MOV R0,0x30
000574 79 40 MOV R1,0x40
000577 75 30 00 MOV 0x30,0x0
00057A 75 31 01 MOV 0x31,0x0
00057D 75 32 02 MOV 0x32,0x0
000580 75 33 03 MOV 0x33,0x0
000583 75 34 04 MOV 0x34,0x0
000586 75 35 05 MOV 0x35,0x0
000589 75 36 06 MOV 0x36,0x0
00058C 75 37 07 MOV 0x37,0x0
00058F 75 38 08 MOV 0x38,0x0
000592 75 39 09 MOV 0x39,0x0
000595 7C 0A MOV R4,0x0A
000598 78 30 MOV R0,0x30
00059B 79 40 MOV R1,0x40
00059E 75 30 00 MOV 0x30,0x0
0005A1 75 31 01 MOV 0x31,0x0
0005A4 75 32 02 MOV 0x32,0x0
0005A7 75 33 03 MOV 0x33,0x0
0005AA 75 34 04 MOV 0x34,0x0
0005AD 75 35 05 MOV 0x35,0x0
0005B0 75 36 06 MOV 0x36,0x0
0005B3 75 37 07 MOV 0x37,0x0
0005B6 75 38 08 MOV 0x38,0x0
0005B9 75 39 09 MOV 0x39,0x0
0005BC 7C 0A MOV R
```



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.

```

MOV 0x32, #2
MOV 0x33, #3
MOV 0x34, #4
MOV 0x35, #5
MOV 0x36, #6
MOV 0x37, #7
MOV 0x38, #8
MOV 0x39, #9

MOV R4, #10 ; le doy valor 10 al contador para repetir 10 veces
MOV R0, #0x30 ; cargo el puntero i con el inicio del bloque de entrada
MOV R1, #0x40 ; cargo el puntero j con el inicio del bloque de salida

loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada
INC R0 ; paso a la siguiente direccion de entrada

CALL ejercicio6 ; aplico la funcion, resultado en DPTR

MOV @R1, DPH ; guardo la parte alta del resultado
INC R1 ; paso a la siguiente direccion (son 2 bytes)
MOV @R1, DPL ; guardo la parte baja del resultado
INC R1 ; paso a la siguiente direccion

DJNZ R4, loop ; repite 10 veces (hasta que R4 = 0)

ret

; FIN DEL ARCHIVO
END main ; Cierra el archivo

```

Basic Registers	Value
A	0
B	0
PSW	0x00
R0	0x31
R1	0x42
R2	7
R3	3
R4	9
R5	0x00
R6	0x00
R7	0x00
SP	0xC1
SPP	-----
SPX	-----
DPTR	7
PC	0x0078
CYCLECOUNTER	672
CCTIMER1	672
CCTIMER2	672
CCSTEP	24

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.

```

MOV 0x32, #2
MOV 0x33, #3
MOV 0x34, #4
MOV 0x35, #5
MOV 0x36, #6
MOV 0x37, #7
MOV 0x38, #8
MOV 0x39, #9

MOV R4, #10 ; le doy valor 10 al contador para repetir 10 veces
MOV R0, #0x30 ; cargo el puntero i con el inicio del bloque de entrada
MOV R1, #0x40 ; cargo el puntero j con el inicio del bloque de salida

loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada
INC R0 ; paso a la siguiente direccion de entrada

CALL ejercicio6 ; aplico la funcion, resultado en DPTR

MOV @R1, DPH ; guardo la parte alta del resultado
INC R1 ; paso a la siguiente direccion (son 2 bytes)
MOV @R1, DPL ; guardo la parte baja del resultado
INC R1 ; paso a la siguiente direccion

DJNZ R4, loop ; repite 10 veces (hasta que R4 = 0)

ret

; FIN DEL ARCHIVO
END main ; Cierra el archivo

```

Basic Registers	Value
A	0
B	0
PSW	0x00
R0	0x3A
R1	0x54
R2	7
R3	3
R4	1
R5	0x00
R6	0x00
R7	0x00
SP	0xC1
SPP	-----
SPX	-----
DPTR	34
PC	0x0083
CYCLECOUNTER	10740
CCTIMER1	10740
CCTIMER2	10740
CCSTEP	12

Disassembly	Code
000069	MOV 0x37, #7
00006A	MOV 0x37, #0x
00006B	MOV 0x38, #8
00006C	MOV 0x38, #0x
00006D	MOV 0x39, #9
00006E	MOV 0x39, #0x
00006F	MOV R4, #10 ; le doy valor
000070	MOV R4, #0x0A
000071	MOV R0, #0x30 ; cargo el pun
000072	MOV R0, #0x30
000073	MOV R1, #0x40 ; cargo el punt
000074	MOV R1, #0x40
000075	loop: MOV A, @R0 ; cargo en A lo
000076	loop:
000077	MOV A, @R0
000078	INC R0 ; paso a la sig
000079	INC R0
00007A	CALL ejercicio6 ; aplico la fun
00007B	LCALL ejercici
00007C	MOV @R1, DPH ; guardo la par
00007D	MOV @R1, DPH
00007E	INC R1 ; paso a la sig
00007F	INC R1
000080	MOV @R1, DPL ; guardo la part
000081	MOV @R1, DPL
000082	INC R1 ; paso a la sig
000083	INC R1
000084	DJNZ R4, loop ; repite 10 vec
000085	DJNZ R4, loop
000086	ret
000087	RET
000088	NOP
000089	NOP
00008A	NOP
00008B	NOP
00008C	NOP
00008D	NOP
00008E	NOP
00008F	NOP

Memory 1	Data
0x00	3a 54 07 03 01 00 00 00 00 00 00 00 00 00 00 00
0x10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x30	00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
0x40	00 07 00 0a 00 0d 00 10 00 13 00 16 00 19 00 1c
0x50	00 1f 00 22 00 00 00 00 00 00 00 00 00 00 00 00
0x60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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:

The screenshot shows a debugger interface with three main panes:

- Assembly:** Displays assembly code for a loop. Key instructions include:
 - `MOV R4, #10 ; le doy valor 10 al contador para repetir 10 veces`
 - `MOV R0, #0x30 ; cargo el puntero i con el inicio del bloque de entrada`
 - `MOV R1, #0x40 ; cargo el puntero j con el inicio del bloque de salida`
 - `loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada`
 - `INC R0 ; paso a la siguiente direccion de entrada`
 - `CALL ejercicio6 ; aplico la funcion, resultado en DPTR`
 - `MOV @R1, DPTR ; guardo la parte alta del resultado`
 - `INC R1 ; paso a la siguiente direccion (son 2 bytes)`
 - `MOV @R1, DPL ; guardo la parte baja del resultado`
 - `INC R1 ; paso a la siguiente direccion`
 - `DJNZ R4, loop ; repite 10 veces (hasta que R4 = 0)`
 - `ret`
- Registers:** Shows the state of registers. R4 is highlighted with a value of 1. Other registers like R0 (0x3A), R1 (0x54), and DPTR (34) are also visible.
- Memory:** Shows a memory dump. The first few bytes are highlighted: 3a 54 07 03 01 00 00 00 00 00 00 00 00 00 00 00. These correspond to the values 42h, 43h, 44h, 45h, etc., in hexadecimal.

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.

Workspace: donis-TP1.s51

```

; 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 sera en dos bytes).

ejercicio14:
; le doy valor a las entradas para probar
MOV R0, #30
MOV R1, #40
MOV R2, #10
MOV R3, #0
MOV R4, #0
MOV R5, #0
MOV R6, #0
MOV R7, #0
MOV R8, #0
MOV R9, #0

MOV R4, #10 ; le doy valor 10 al contador para repetir 10 veces
MOV R0, #0x30 ; cargo el puntero i con el inicio del bloque de entrada
MOV R1, #0x40 ; cargo el puntero j con el inicio del bloque de salida

loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada
      INC R0 ; paso a la siguiente direccion de entrada

      CALL ejercicio6 ; aplico la funcion, resultado en DPTR

      MOV @R1, DPH ; guardo la parte alta del resultado
      INC R1 ; paso a la siguiente direccion (son 2 bytes)
      MOV @R1, DPL ; guardo la parte baja del resultado
      INC R1 ; paso a la siguiente direccion

      DJNZ R4, loop

      ret

; FIN DEL ARCHIVO
END main ; Cierre el archivo
  
```

Registers:

Register	Value
A	90
B	0
PSW	0x00
R0	0x30
R1	0x40
R2	10
R3	0
R4	10
R5	0x00
R6	0x00
R7	0x00
SP	0xC1
SPP	-----
SPX	-----
DPTR	0
PC	0x0079
CYCLECOUNTER	360
CCTIMER1	360
CCTIMER2	360
CCSTEP	12

Disassembly:

```

ejercicio14:
000054 75 30 5A      MOV    0x30,@R0
000057 75 31 01      MOV    0x31,@R0
00005A 75 32 02      MOV    0x32,@R0
00005D 75 33 03      MOV    0x33,@R0
000060 75 34 04      MOV    0x34,@R0
000063 75 35 05      MOV    0x35,@R0
000066 75 36 06      MOV    0x36,@R0
000069 75 37 07      MOV    0x37,@R0
00006C 75 38 08      MOV    0x38,@R0
00006F 75 39 09      MOV    0x39,@R0
000072 7C 0A      MOV    R4,#10
000075 75 30 0A      MOV    R0,#0x30
000078 78 30      MOV    R0,#0x30
00007B 79 40      MOV    R1,#0x40
00007E 79 40      MOV    R1,#0x40
loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada
      INC R0 ; paso a la siguiente direccion de entrada
      CALL ejercicio6 ; aplico la funcion, resultado en DPTR
      MOV @R1, DPH ; guardo la parte alta del resultado
      INC R1 ; paso a la siguiente direccion (son 2 bytes)
      MOV @R1, DPL ; guardo la parte baja del resultado
      INC R1 ; paso a la siguiente direccion
      DJNZ R4, loop
      ret
  
```

Memory:

Address	Data
0x00	30 40 00 00 0a 00 00 00 00 00 00 00 00 00 00 00
0x10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x30	01 02 03 04 05 06 07 08 09 00 00 00 00 00 00 00
0x40	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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.

Workspace: donis-TP1.s51

```

MOV R0, #30
MOV R1, #40
MOV R2, #10
MOV R3, #0
MOV R4, #0
MOV R5, #0
MOV R6, #0
MOV R7, #0
MOV R8, #0
MOV R9, #0

MOV R4, #10 ; le doy valor 10 al contador para repetir 10 veces
MOV R0, #0x30 ; cargo el puntero i con el inicio del bloque de entrada
MOV R1, #0x40 ; cargo el puntero j con el inicio del bloque de salida

loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada
      INC R0 ; paso a la siguiente direccion de entrada

      CALL ejercicio6 ; aplico la funcion, resultado en DPTR

      MOV @R1, DPH ; guardo la parte alta del resultado
      INC R1 ; paso a la siguiente direccion (son 2 bytes)
      MOV @R1, DPL ; guardo la parte baja del resultado
      INC R1 ; paso a la siguiente direccion

      DJNZ R4, loop

      ret

; FIN DEL ARCHIVO
END main ; Cierre el archivo
  
```

Registers:

Register	Value
A	1
B	1
PSW	0x01
R0	0x31
R1	0x42
R2	7
R3	3
R4	10
R5	0x00
R6	0x00
R7	0x00
SP	0xC1
SPP	-----
SPX	-----
DPTR	277
PC	0x0083
CYCLECOUNTER	648
CCTIMER1	648
CCTIMER2	648
CCSTEP	12

Disassembly:

```

ejercicio14:
000069 75 37 07      MOV    0x37,@R0
00006C 75 38 08      MOV    0x38,@R0
00006F 75 39 09      MOV    0x39,@R0
000072 7C 0A      MOV    R4,#10
000075 75 30 0A      MOV    R0,#0x30
000078 78 30      MOV    R0,#0x30
00007B 79 40      MOV    R1,#0x40
00007E 79 40      MOV    R1,#0x40
loop: MOV A, @R0 ; cargo en A lo que apunta el puntero de entrada
      INC R0 ; paso a la siguiente direccion de entrada
      CALL ejercicio6 ; aplico la funcion, resultado en DPTR
      MOV @R1, DPH ; guardo la parte alta del resultado
      INC R1 ; paso a la siguiente direccion (son 2 bytes)
      MOV @R1, DPL ; guardo la parte baja del resultado
      INC R1 ; paso a la siguiente direccion
      DJNZ R4, loop
      ret
  
```

Memory:

Address	Data
0x00	31 42 07 03 0a 00 00 00 00 00 00 00 00 00 00 00
0x10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x30	01 02 03 04 05 06 07 08 09 00 00 00 00 00 00 00
0x40	01 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

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.