Arellano Granados Angel Mariano

218123444

Seminario de Traductores de Lenguajes I

I7026 D02

Reporte de Actividades 4

Actividades 4 – Parte 1 – 4

Actividad 4 – Parte 1

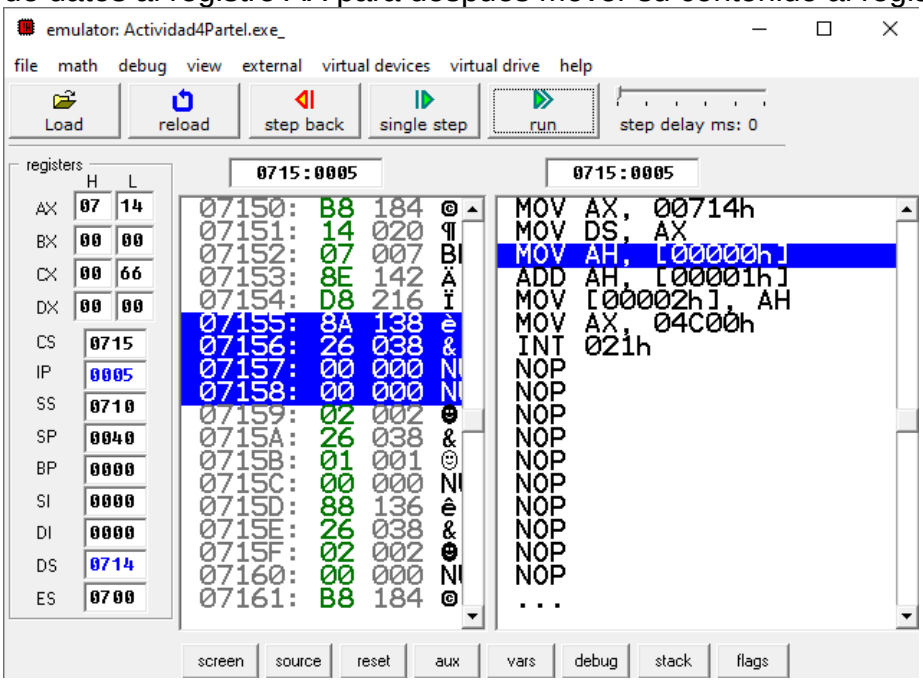
Descripción

Corra el siguiente código fuente en el ensamblador EMU8086 y reporte paso a paso el funcionamiento del programa.

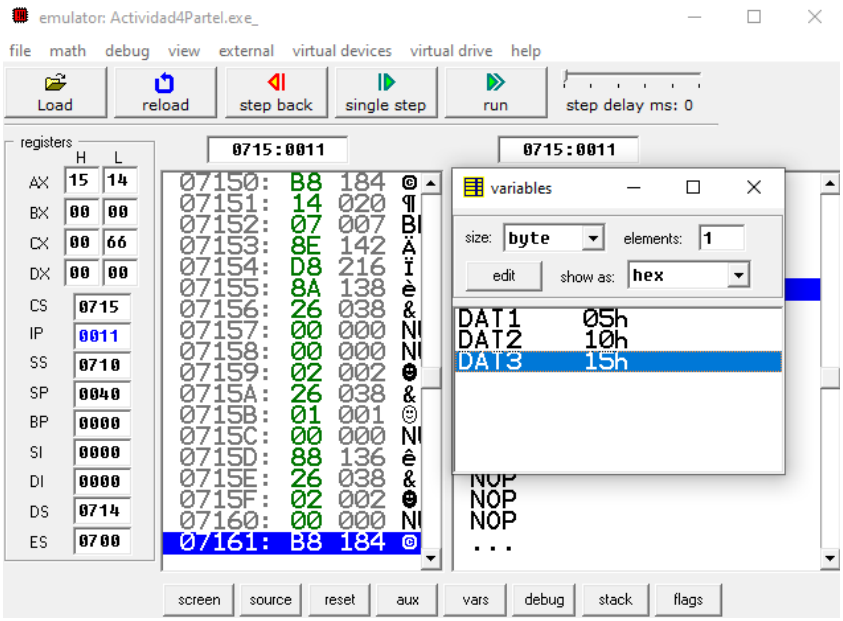
Desarrollo y Resultados

Reporte

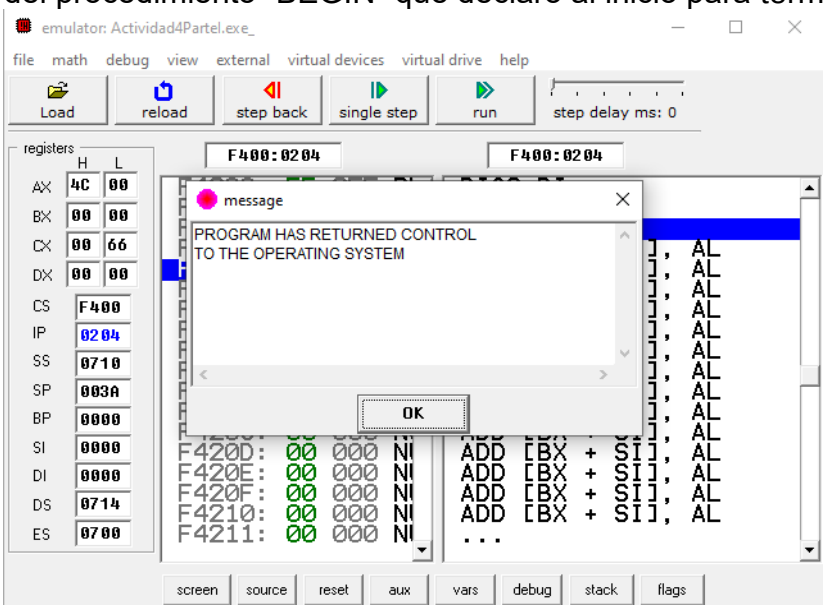
En la línea 01 se usa la directiva PAGE para designar un número máximo de 60 líneas con un máximo de 132 caracteres por línea.
En la línea 02 se emplea la directiva TITLE para darle el título de “PROG1.EXE”.
En la línea 03 se declara que el programa tendrá un modelo pequeño (Model small) esto significa que solo tendrá un segmento de datos y uno de código.
En la Línea 04 se declara que la pila tendrá un máximo de 64 datos.
De las líneas 06 a 09 se encuentra el segmento de datos y se inician 3 variables llamadas DAT1 que contiene 05H, DAT2 que contiene 10H y DAT3 que no contiene un dato aún denotado como “?”.
De las líneas 11 a 23 se encuentra el segmento de código donde primero inicia un procedimiento lejano llamado “BEGIN”, después mueve la dirección del segmento de datos al registro AX para después mover su contenido al registro DS.



ya con acceso al segmento de datos mueve DAT1 que contiene 05H al registro AH para sumarle el DAT2 que contiene 10H, en este punto tendríamos 15H en el registro AH que posteriormente se asignará a la variable DAT3.



Para terminar, moverá 4C00H al registro AX y llamaría a la interrupción 21H que si AH contiene 4CH el programa termina con un código de retorno y marca el final del procedimiento “BEGIN” que declaró al inicio para terminar el programa.



Reflexión

Para comprender el funcionamiento de los programas en ensamblador es importante conocer el funcionamiento de todas las funciones esenciales para un programa simple, así como aprender a buscar el funcionamiento de aquellas que no conozcamos al momento de verlas para poder usarlas en el futuro.

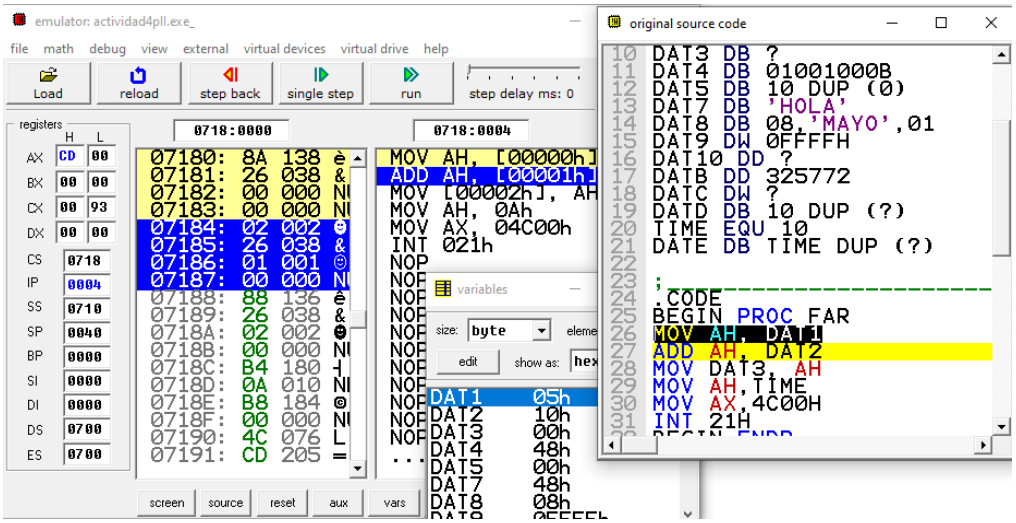
Actividad 4 – Parte 2

Descripción

Identifique y reporte los problemas que surgen al no cargar el segmento de datos en el segmento de código, corrija y reporte los cambios.

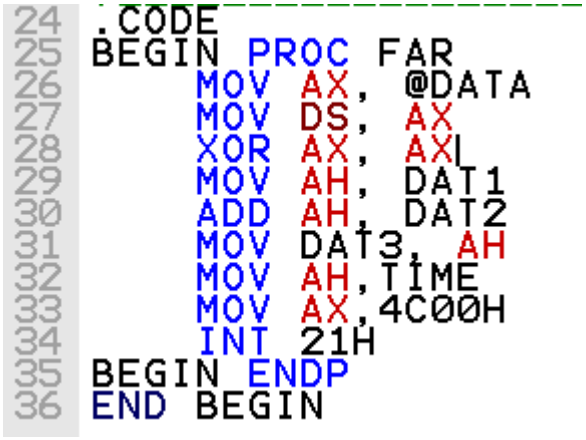
Desarrollo y Resultados

Reporte de error:
Al no cargar el segmento de datos al inicio del segmento de código cuando se pide acceder al valor de una variable este obtiene basura de las primeras direcciones de memoria del disco.

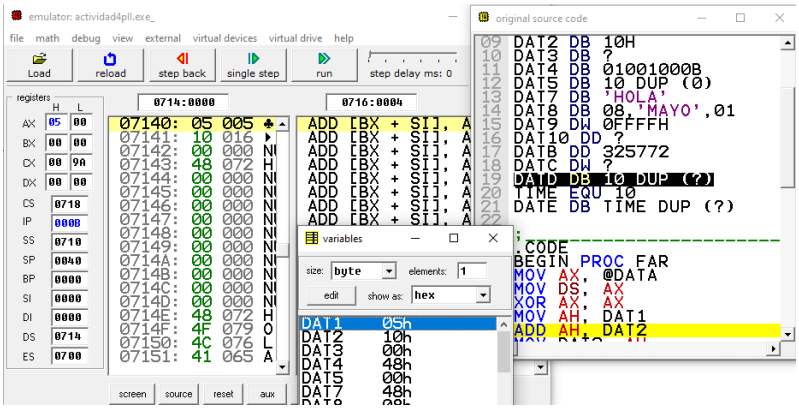


Cuando DAT1 tiene el valor de 05H este carga CDH que pertenece a la dirección [00000H].

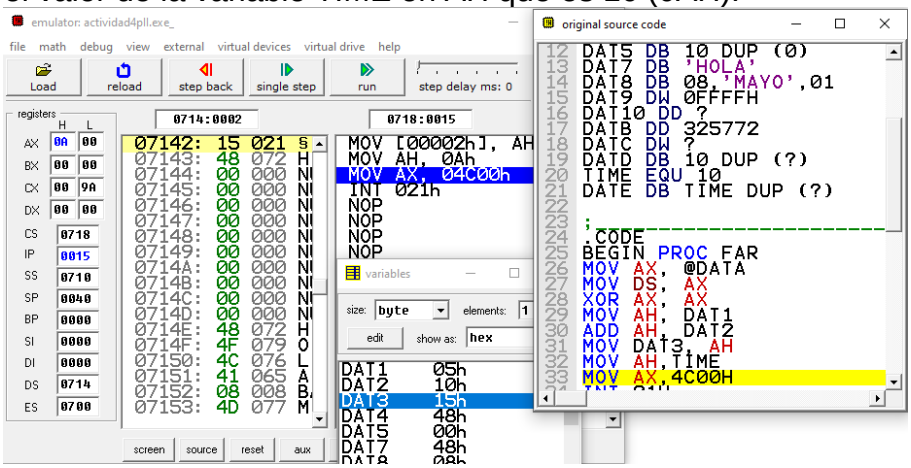
Reporte de solución:
Para solucionar el error agregue las líneas 26-28 donde cargamos la dirección del segmento de datos al registro AX y después lo movemos a DS,
la línea 28 solo limpia el registro AX.



Con estos cambios ya se corrige el problema inicial y ya logra recibir el valor de DAT1 que es 05H.



Así como el resto de procesos del programa como sumar DAT1 y DAT2 (05H+10H) y guardarlo en DAT3 que termina con el valor de 15H, también carga el valor de la variable TIME en AH que es 10 (0AH).



Por último, pone 4200H en AX para llamar a la interrupción 21H y concluir el programa.

Reflexión

Con este ejemplo vemos la importancia que tiene cargar el segmento de datos, un simple detalle de un par de líneas que puede hacer que todo el programa falle por completo, por ello es importante conocer que pasa cuan lo olvidamos para no cometer ese error en futuros programas.

Actividad 4 – Parte 3

Descripción

Investigar el funcionamiento de la instrucción JMP, analice y compruebe el funcionamiento en el siguiente código fuente. Identifique en qué casos es útil una iteración infinita.

Desarrollo y Resultados

Unconditional Jump Instructions

These instructions are used to jump on a particular location unconditionally, i.e. there is no need to satisfy any condition for the jump to take place. There are three types of procedures used for unconditional jump. They are:

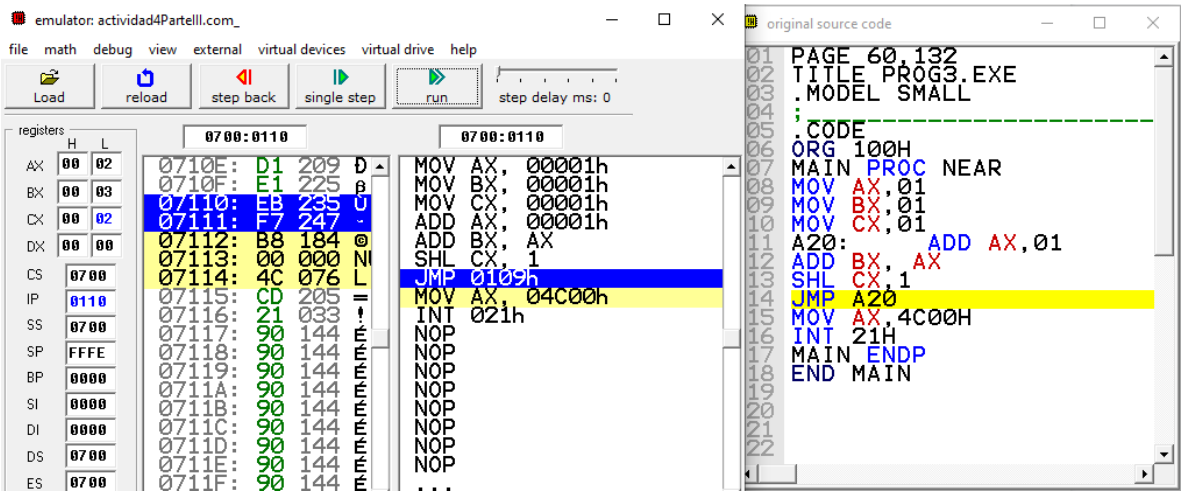
- NEAR – This procedure targets within the same code segment. (Intra-segment)
- SHORT - This procedure also targets within the same code segment, but the offset is 1 byte long. (Intra-segment)
- FAR - In this procedure, the target is outside the segment and the size of the pointer is double word. (Inter-segment)

Syntax: JMP procedure_namememory_location
Example: JMP short target

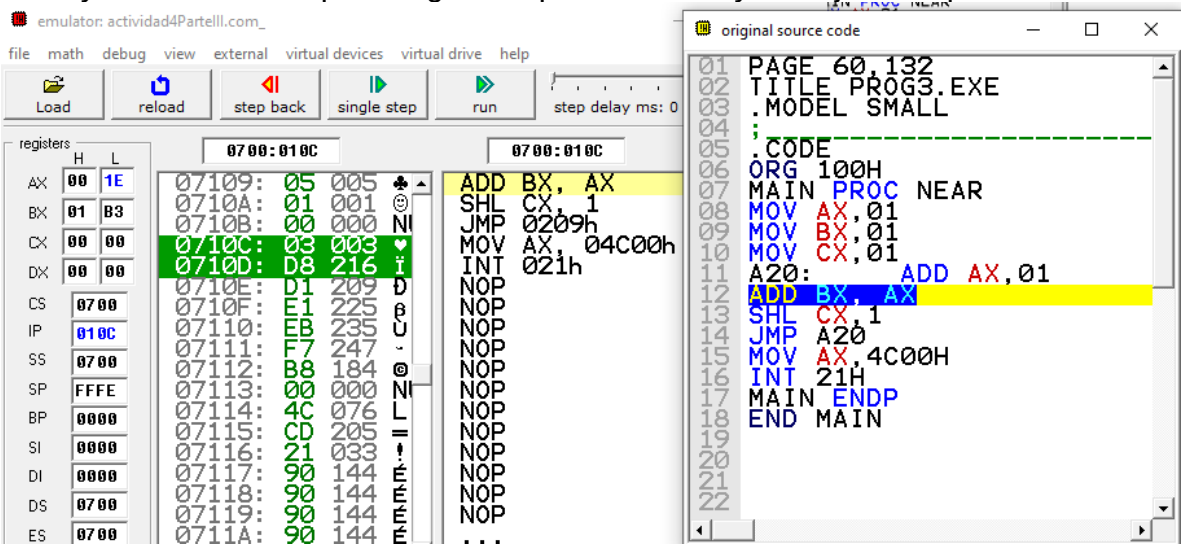
Includehelp. (2022). Jump instructions in 8086 microprocessor. Includehelp.com - Learn Latest Computer Programming Languages with solved programs, tutorials, aptitude solutions. https://www.includehelp.com/embedded-system/jump-instructions-in-8086-microprocessor.aspx

Análisis del código

En el programa iniciar el procedimiento MAIN se colocan 01 en los registros AX, BX y CX, después inicia una etiqueta llamada A20 que suma uno al valor del registro AX, suma los valores de AX y BX para dejar el resultado en BX y hace un desplazamiento lógico a la izquierda de un paso a CX, para hacer un salto incondicional a A20 creando un ciclo infinito que impide que el programa termine.



La acción que se lleva a cabo en el registro CX acaba a las pocas iteraciones porque no hay suficientes bits para seguir desplazando el 1 y se dejan de apreciar cambios.



Aplicaciones de las iteraciones infinitas

- Sistemas de notificación
- Tareas de baja prioridad en algoritmos
- Ataques a servidores
- Hacer que un programa sea más sencillo, pero más costoso

Reflexión

Los ciclos son de las estructuras mas usadas en todos los lenguajes de programación, aprender su funcionamiento único en ensamblador es esencial para el curso, así como notar la importancia de las etiquetas y su correcto uso y no car en ciclos infinitos (a menos que ese sea el propósito del programa) que también tiene usos muy interesantes en la computación actual.

Actividad 4 – Parte 4

Descripción

Realice una investigación sobre el uso de la instrucción LOOP, ejecute el siguiente programa y reporte lo que sucede.

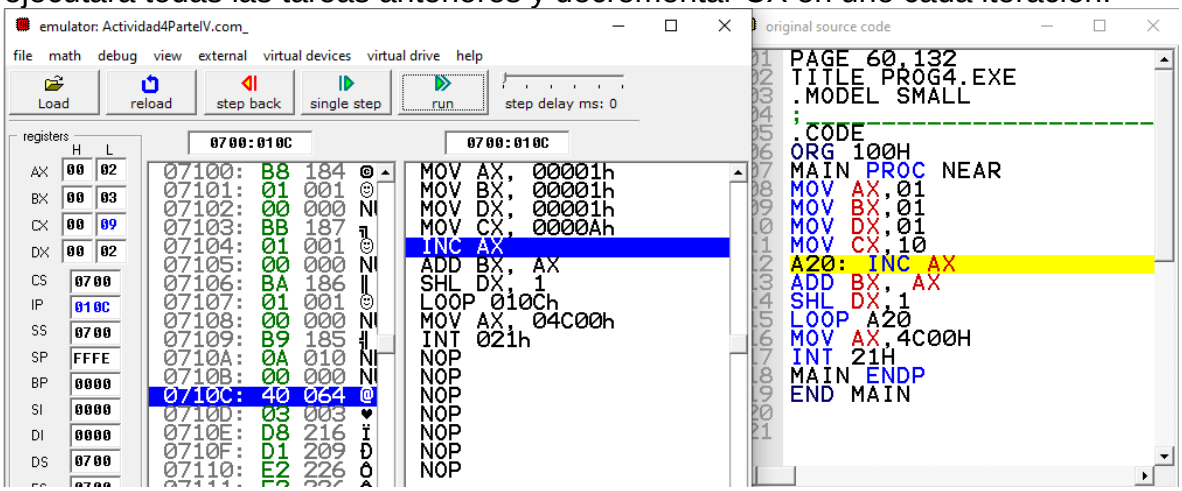
Desarrollo y Resultados

The LOOP instruction executes the group of instructions a number of times and it uses relative addressing mode. The number of iterations will depend on the condition to be satisfied. The CX register will perform the LOOP operation on the instructions to be iterated.
For every execution of LOOP instruction, the CX is automatically decremented by one without affecting flags and performs a short jump to the target address. This loop will continue until the CX becomes zero. When CX = 0, the execution of the loop will stop and the instructions after the LOOP will start execution.

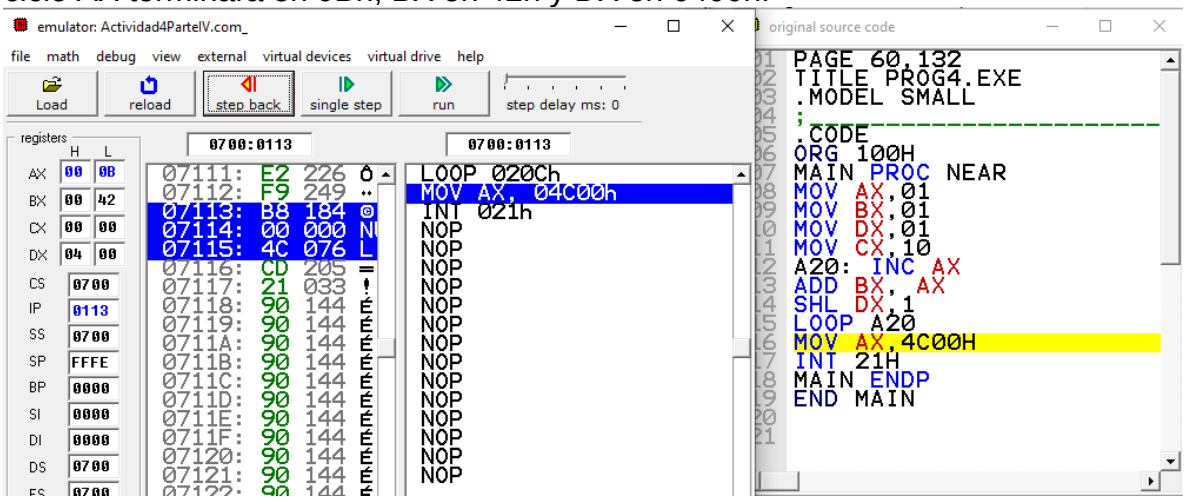
Mnemonic	Meaning	Format	Operation
LOOP	Loop	LOOP short-label	Decrements CX by one, if CX ≠ 0, jump to “short-label”.

Electronics Mind. (2022, January 2). Loop instructions of 8086 microprocessor - Types & examples. https://www.electronicsmind.com/loop-instructions-of-8086-microprocessor/

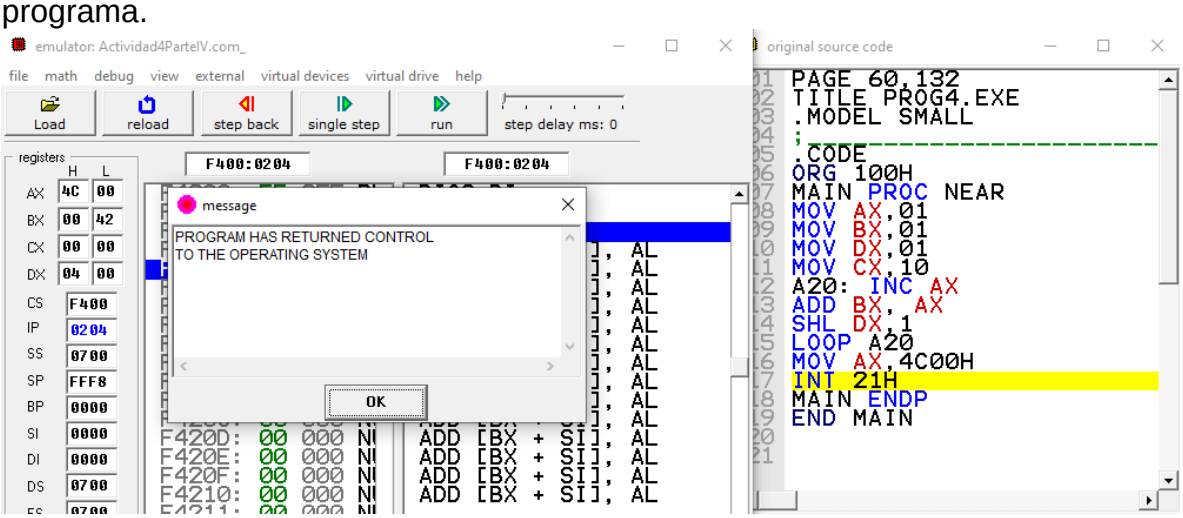
Reporte del programa:
El programa MAIN coloca un número 01h en los registros AX, BX y DX y el número 10d en el registro CX que controlará el número de iteraciones del ciclo, después inicia una etiqueta llamada A20 que incrementa AX en uno, suma BX y AX y desplaza un bit a la izquierda DX, ahora se la etiqueta A20 entra en un ciclo que ejecutará todas las tareas anteriores y decrementar CX en uno cada iteración.



Haciendo un total de 10 iteraciones más la que se hizo al inicio antes de entrar al ciclo AX terminará en 0Bh, BX en 42h y DX en 0400h.



Posteriormente se llama a la interrupción 21H con 4C00H en AX para terminar el programa.



Reflexión

LOOP es también un instrucción para hace ciclos de una manera mas segura que usar JMP ya que es menos probable hacer iteraciones infinitas, algo parecido al for en C++, pero ambas tienen una gran variedad de usos únicos que fue interesante notar.