**Actividad 2**

**CARRERA:** Ingeniería en Computación

**NOMBRE:** *Efrain Robles Pulido*

**CÓDIGO:** 221350095

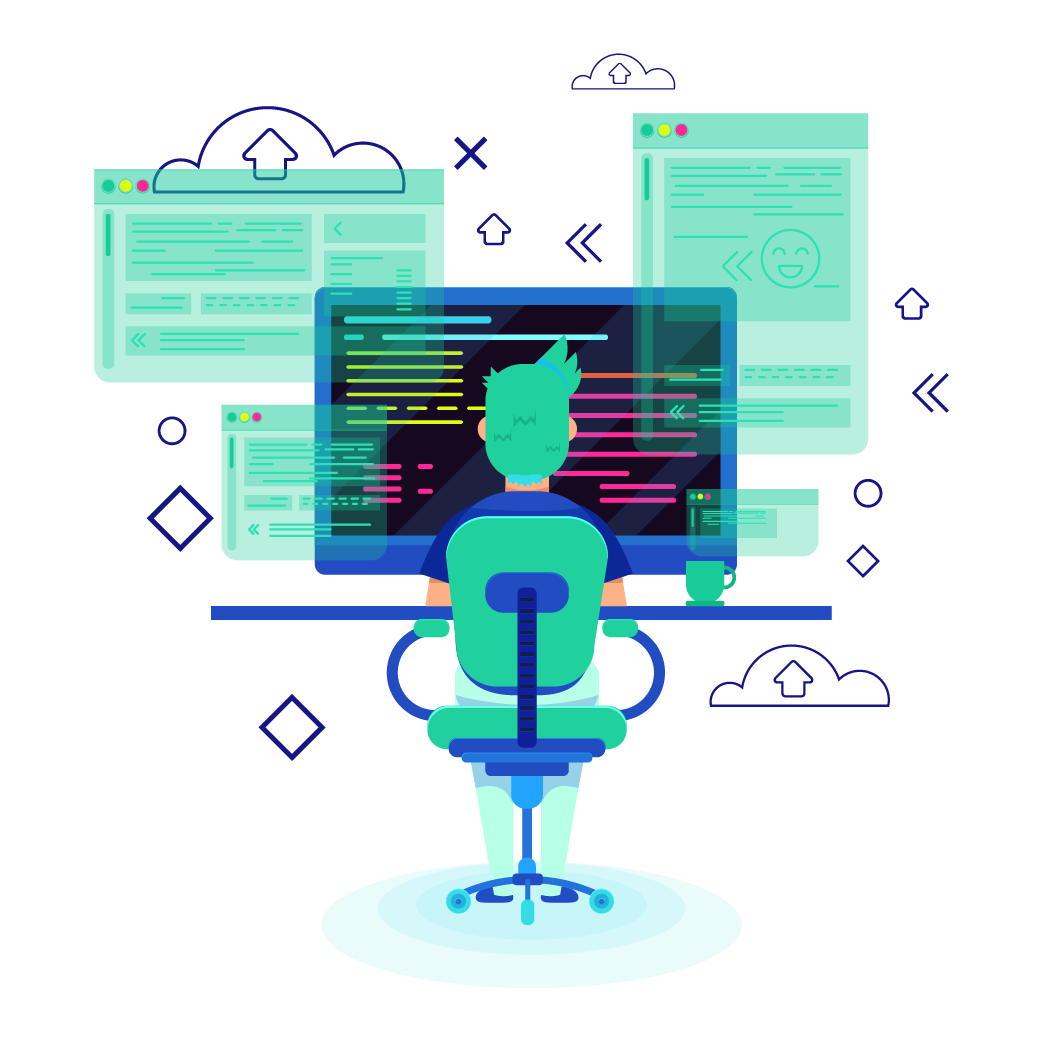
**MATERIA:** Traductores de Lenguajes I

**MAESTRA:** José Juan Meza Espinoza

**SECCIÓN:** D05 **CALENDARIO:** 2023A

**UNIVERSIDAD DE GUADALAJARA**



**e**

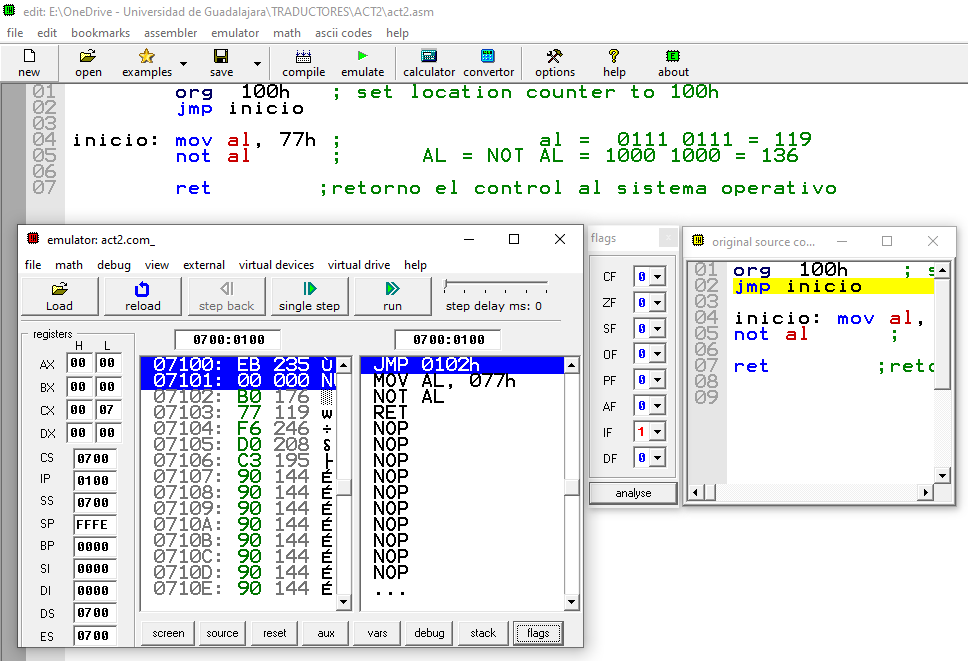
**Desarrollo**

Implementar un programa donde muestres el uso de las instrucciones lógicas (NOT, NEG, AND, OR y XOR)

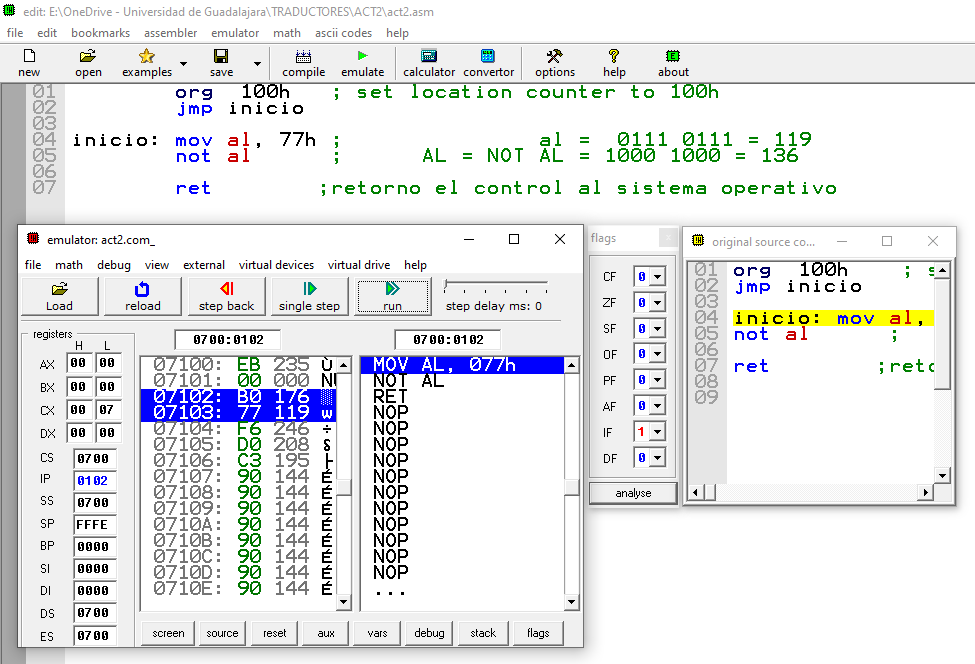
* Tienes que poner el código, comentarlo y pantallas donde muestres (en el emu8086) que si entiendes el funcionamiento de esas instrucciones.

**NOT**

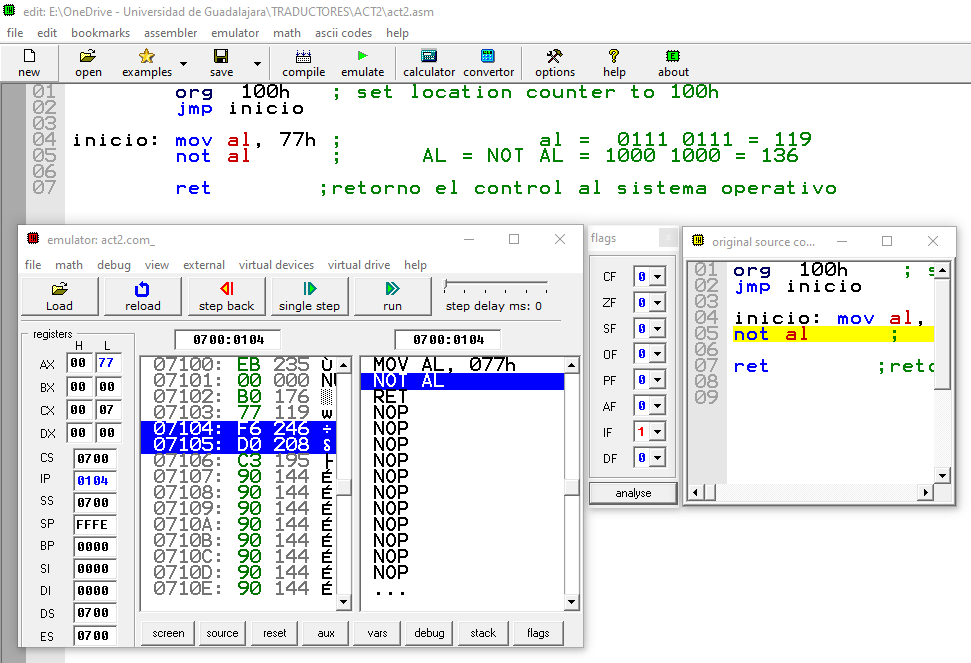
Se realizo la operación NOT en donde primero se introducirá un valor a algún registro de nuestra elección para su respectiva operación, para que a su vez obtener su resultado en el mismo registro usado donde se realizó la operación.



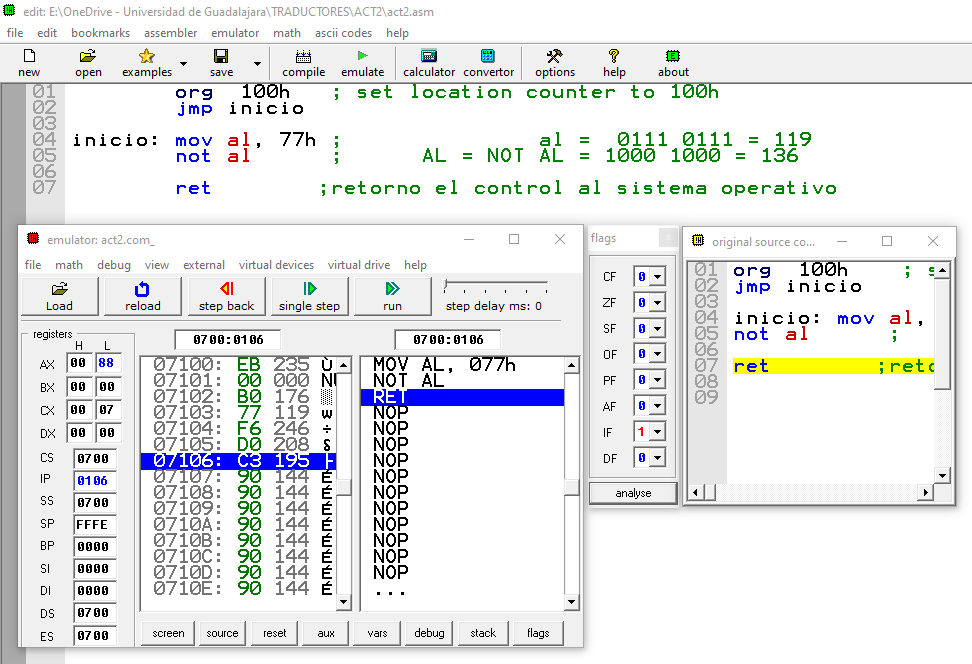
Iniciando la simulación, leerá la primera instrucción en donde saltará a nuestra etiqueta de inicio para leer la instrucción en esa línea de código. Por lo que nuestras banderas están en 0´s porque apenas iniciamos, a excepción de la IF (interrupción) que estará habilitado para nuestro análisis en esta práctica.



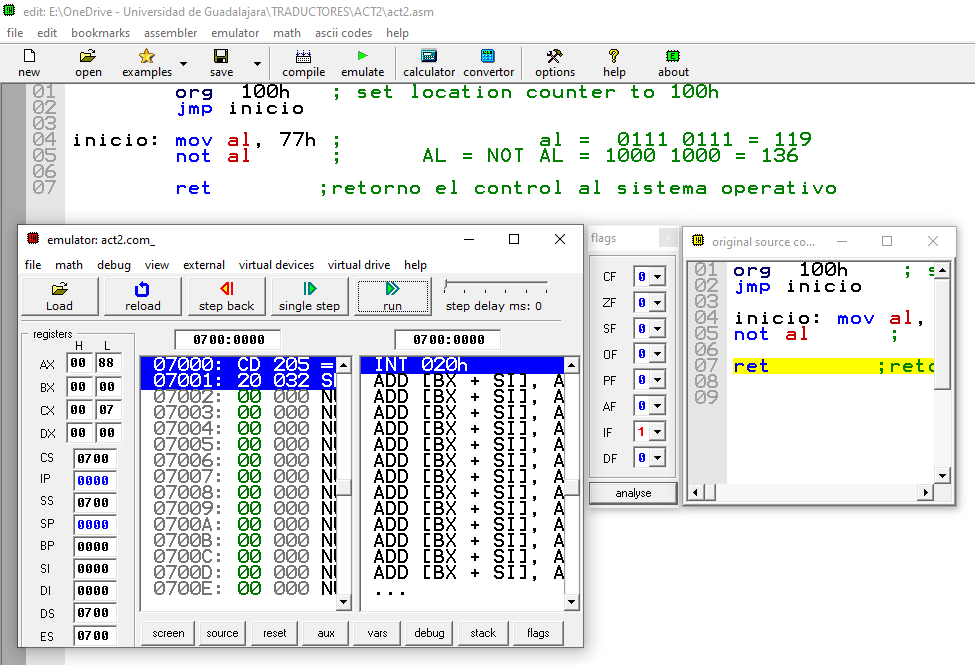
Ahora que se movió en esa línea de código, con MOV se transferirá el valor 77(hexadecimal) al acumulador bajo para almacenar el valor. Las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



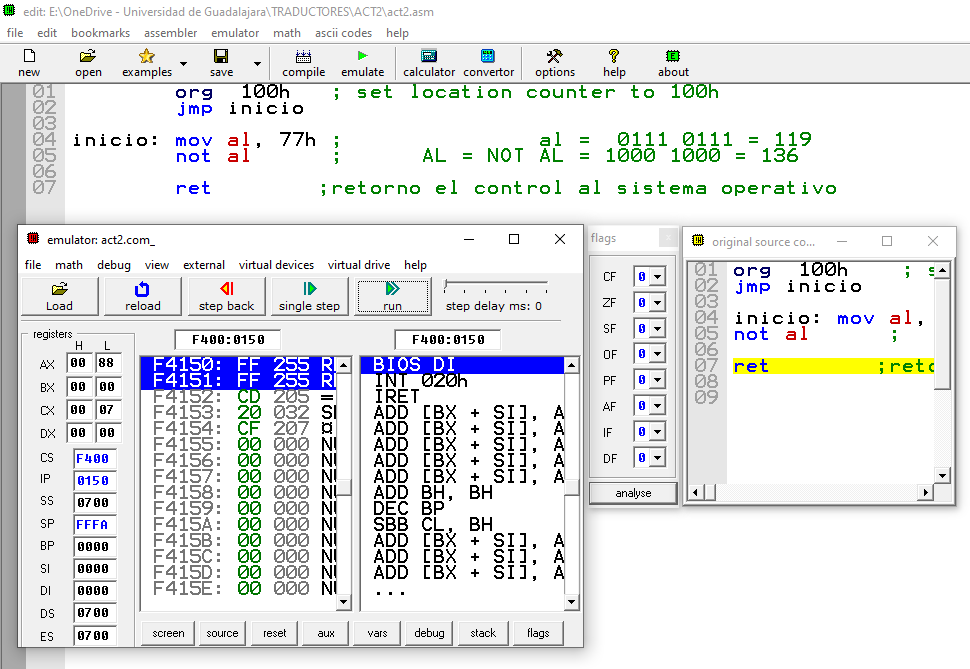
Una vez ejecutado la anterior instrucción, podemos observar que el valor 77(hexadecimal) ya se almacenó en el AL (acumulador en su estado bajo). Para que a continuación se ejecuta la siguiente instrucción.



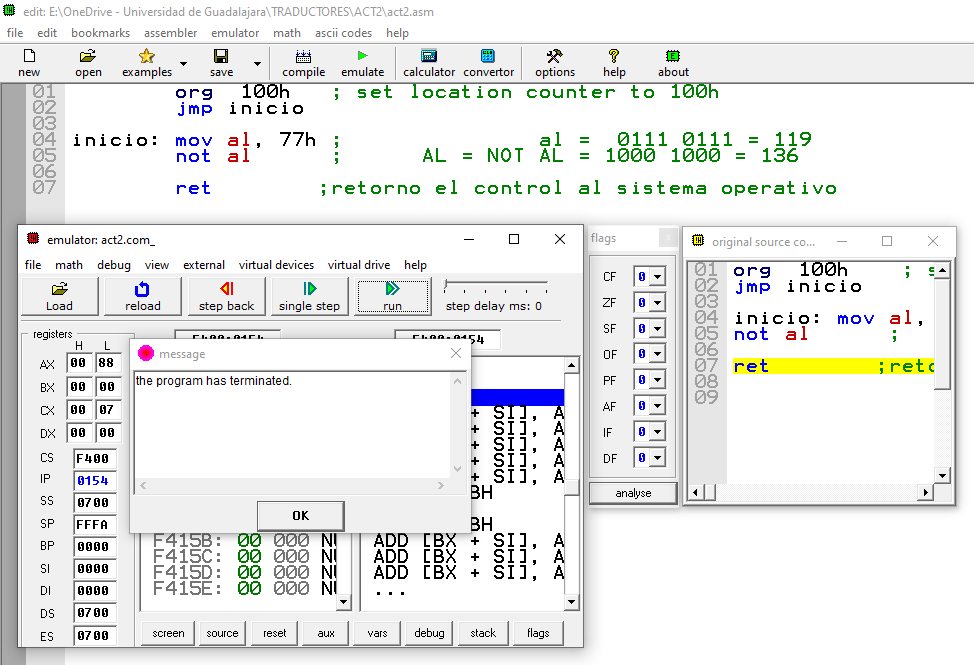
Una vez efectuado la operación NOT al acumulador bajo, se ha invertido los 0 y 1 de su forma binaria para tener como resultado el 88(hexadecimal), y este valor se ha almacenado en el mismo acumulador AL. Además, de que sus banderas no cambiaron debido a que no fue necesario, indicándonos que CF no lleva un acarreo o un arrastre, ZF es un numero diferente a cero, SF es un numero positivo, OF no se ha desbordado el número, PF es número impar, AF no llevamos un acarreo o un arrastre en el bit 3 y 4, IF siempre tendremos interrupciones para el análisis, DF nuestros registros incrementaran solos.



Ahora realizara el retorno para que el sistema operativo tenga el control, una vez leído la instrucción de RET.



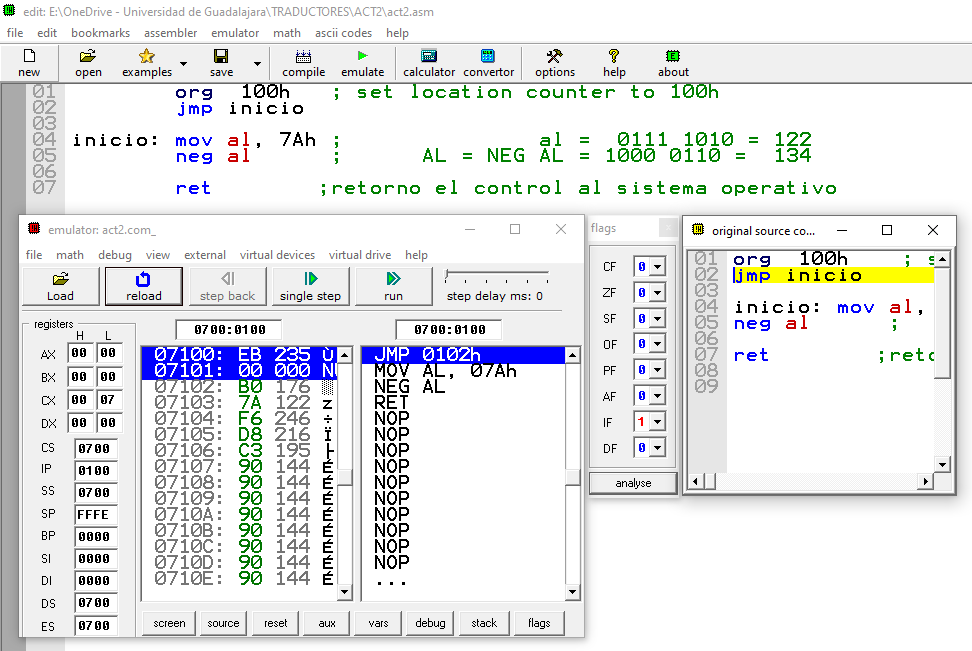
En el que aun está realizando el proceso del retorno del control al sistema operativo.



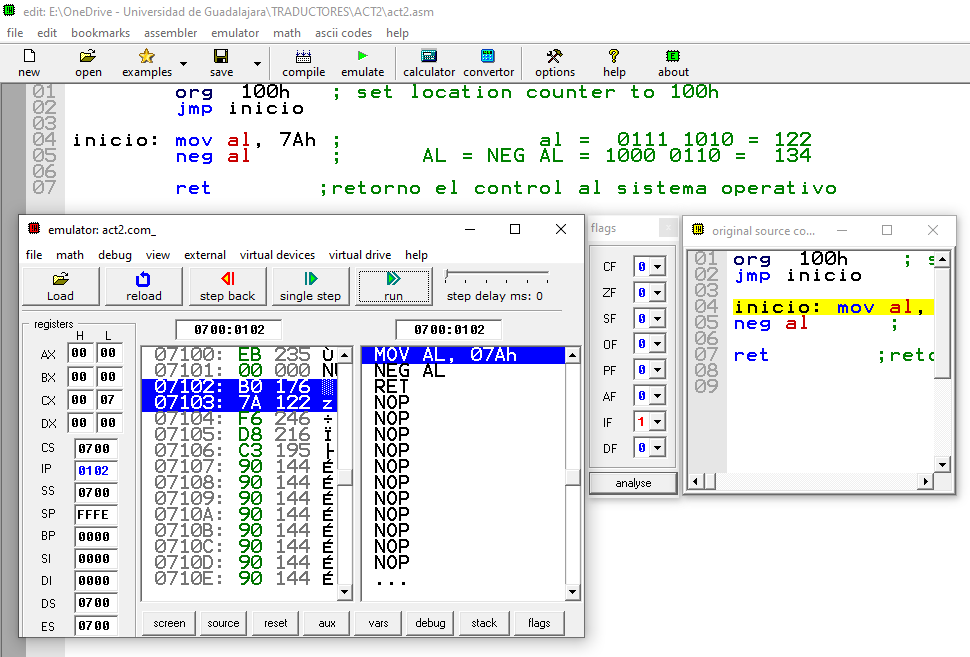
Dando por finalizando el programa de la operación NOT.

**NEG**

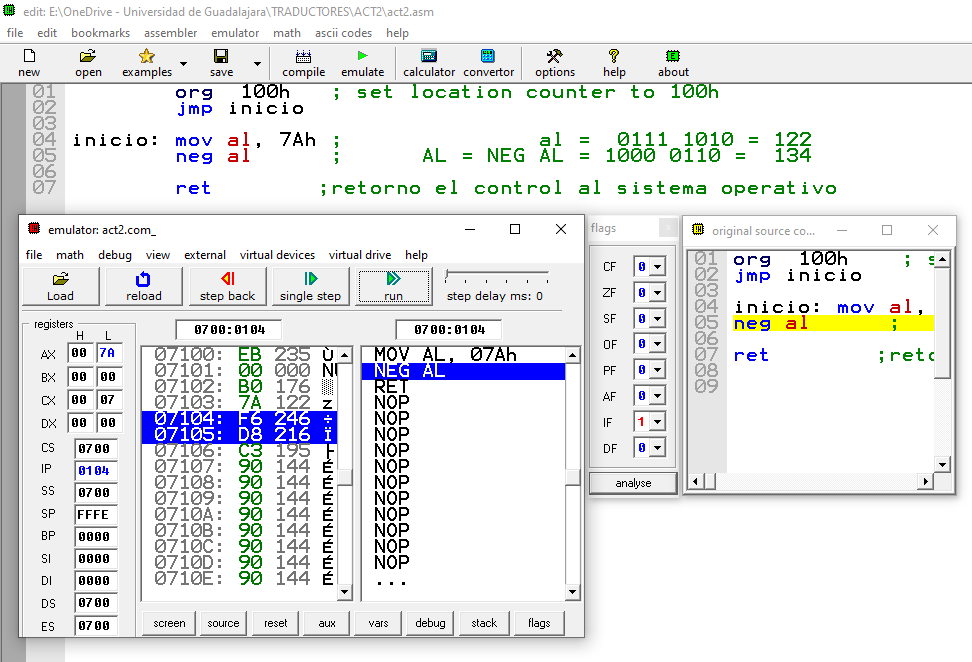
Se realizo la operación NEG en donde primero se introducirá un valor a algún registro de nuestra elección para su respectiva operación, para que a su vez obtener su resultado en el mismo registro usado donde se realizó la operación.



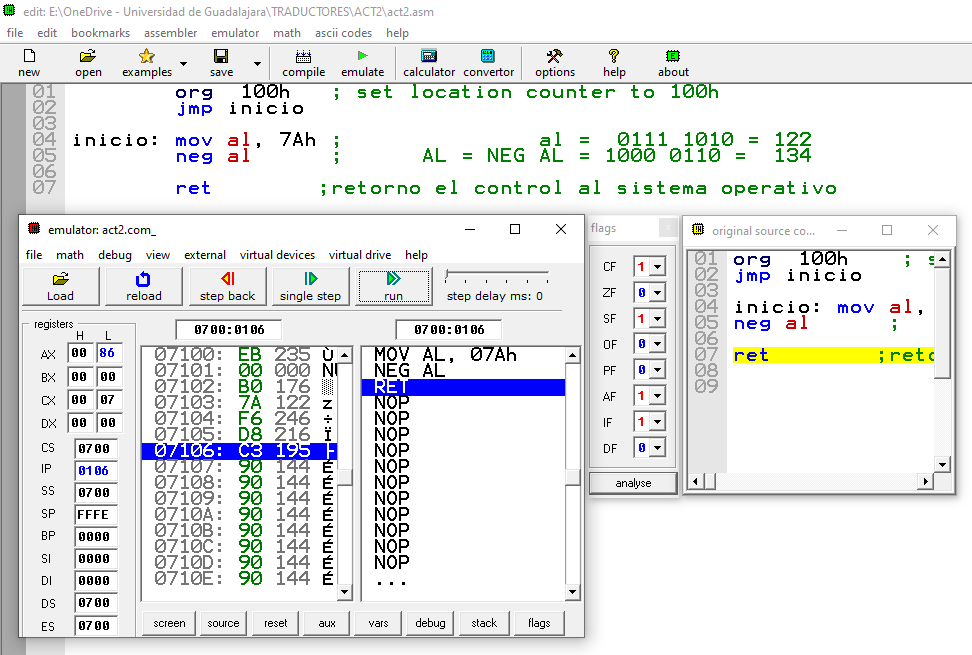
Iniciando la simulación, leerá la primera instrucción en donde saltará a nuestra etiqueta de inicio para leer la instrucción en esa línea de código. Por lo que nuestras banderas están en 0´s porque apenas iniciamos, a excepción de la IF (interrupción) que estará habilitado para nuestro análisis en esta práctica.



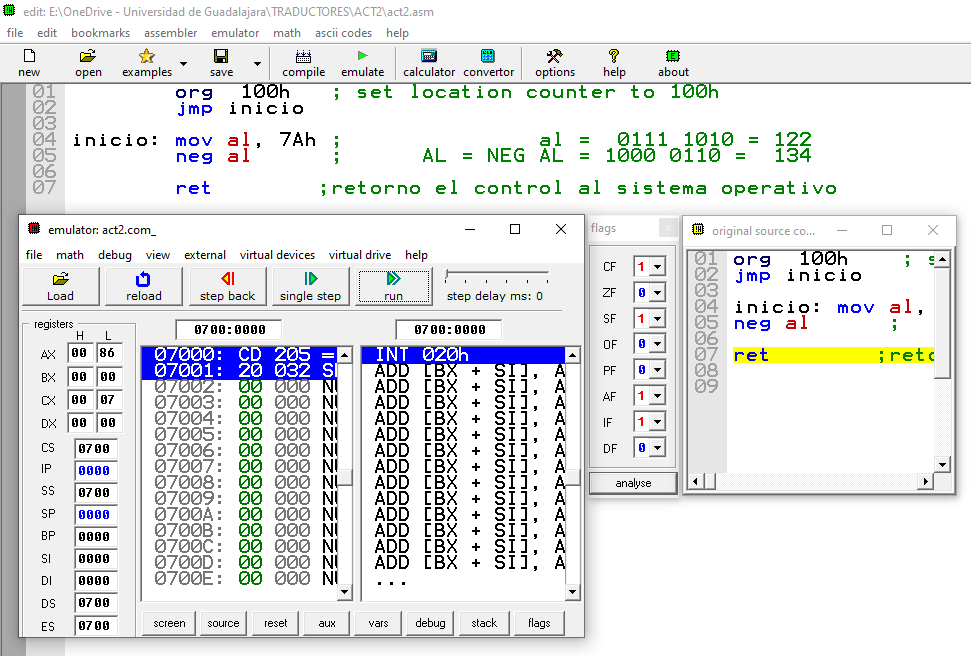
Ahora que se movió en esa línea de código, con MOV se transferirá el valor 7A(hexadecimal) al acumulador bajo para almacenar el valor. Las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



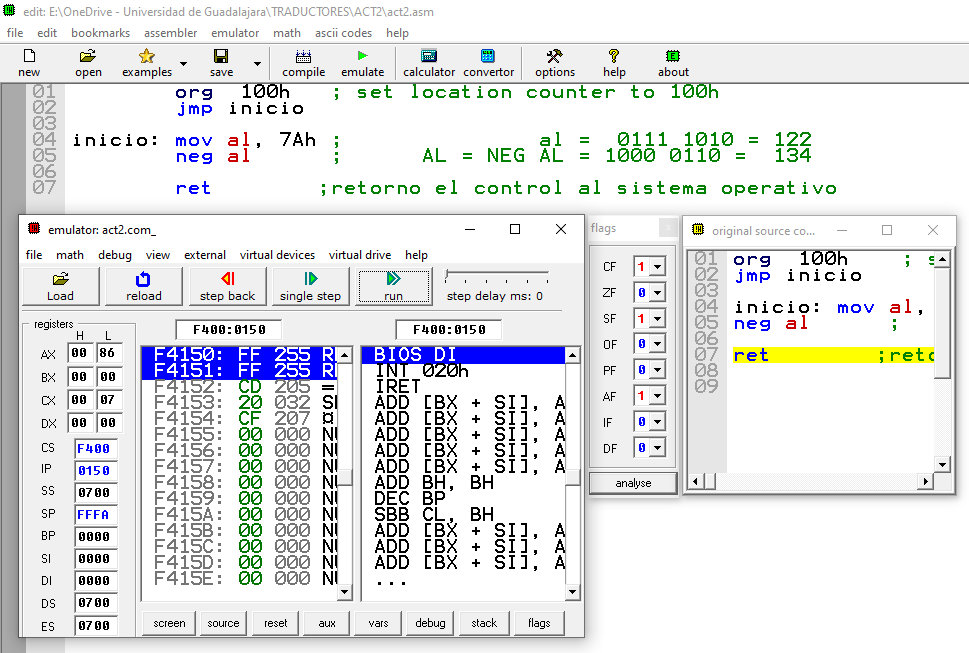
Una vez ejecutado la anterior instrucción, podemos observar que el valor 7A(hexadecimal) ya se almacenó en el AL (acumulador en su estado bajo). Para que a continuación se ejecuta la siguiente instrucción.



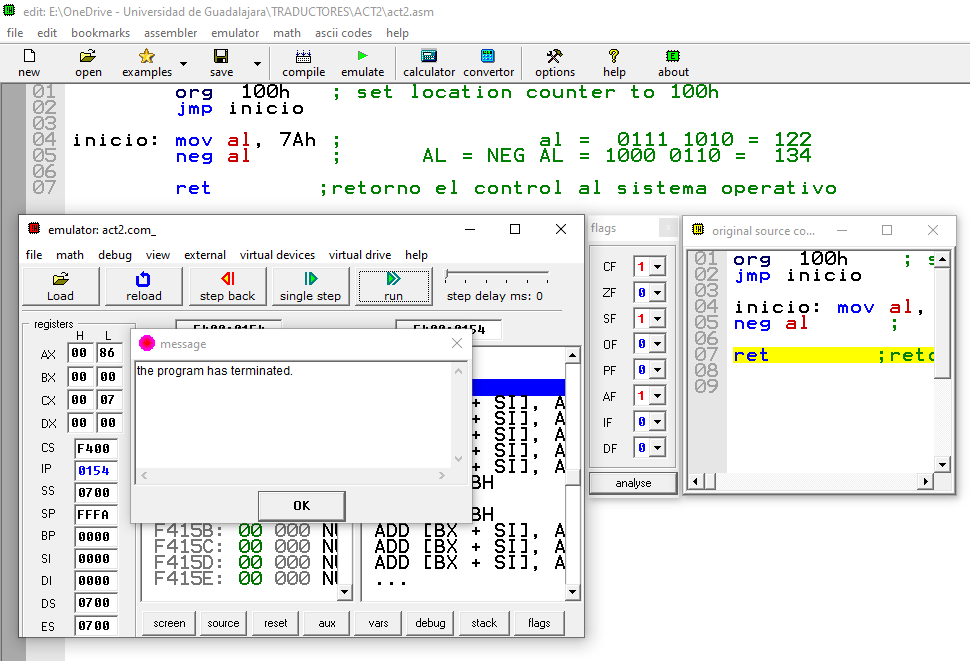
Una vez efectuado la operación NEG al acumulador bajo, se ha invertido los 0 y 1 de su forma binaria a partir del primer 1 encontrado para tener como resultado el 86(hexadecimal). Además, de que sus banderas nos indican que, CF está llevando acarreos o un arrastre, ZF es un numero diferente a cero, SF es un numero negativo (hay un 1 hasta la izq.), OF no se ha desbordado el número, PF es número impar, AF llevamos un acarreo o un arrastre en el bit 3 y 4, IF siempre tendremos interrupciones para el análisis, DF nuestros registros incrementaran solos.



Ahora realizara el retorno para que el sistema operativo tenga el control, una vez leído la instrucción de RET.



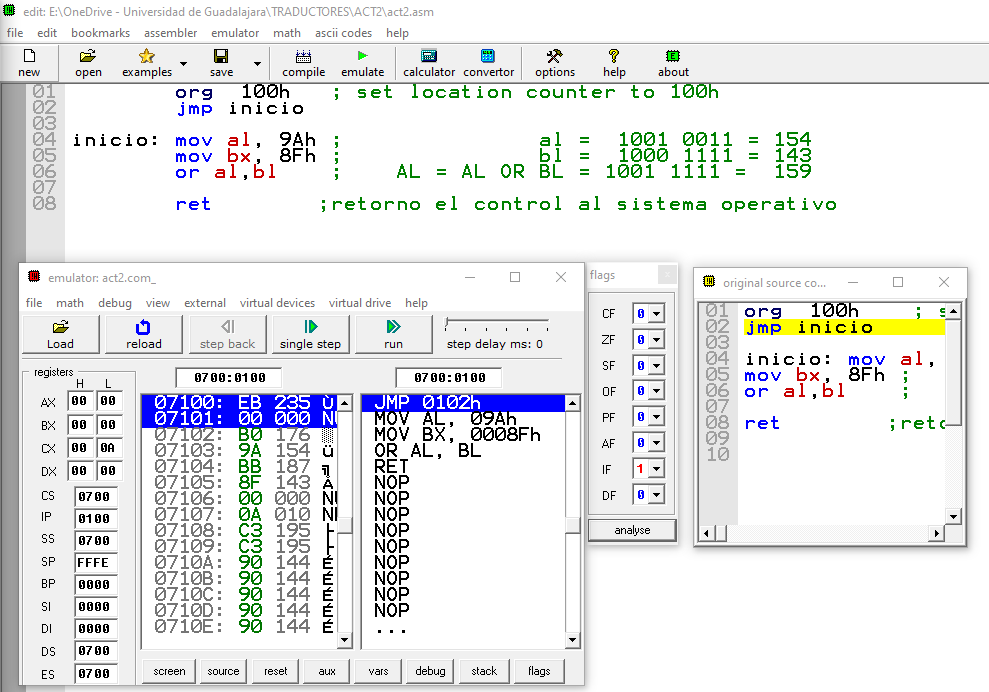
En el que aún está realizando el proceso del retorno del control al sistema operativo.



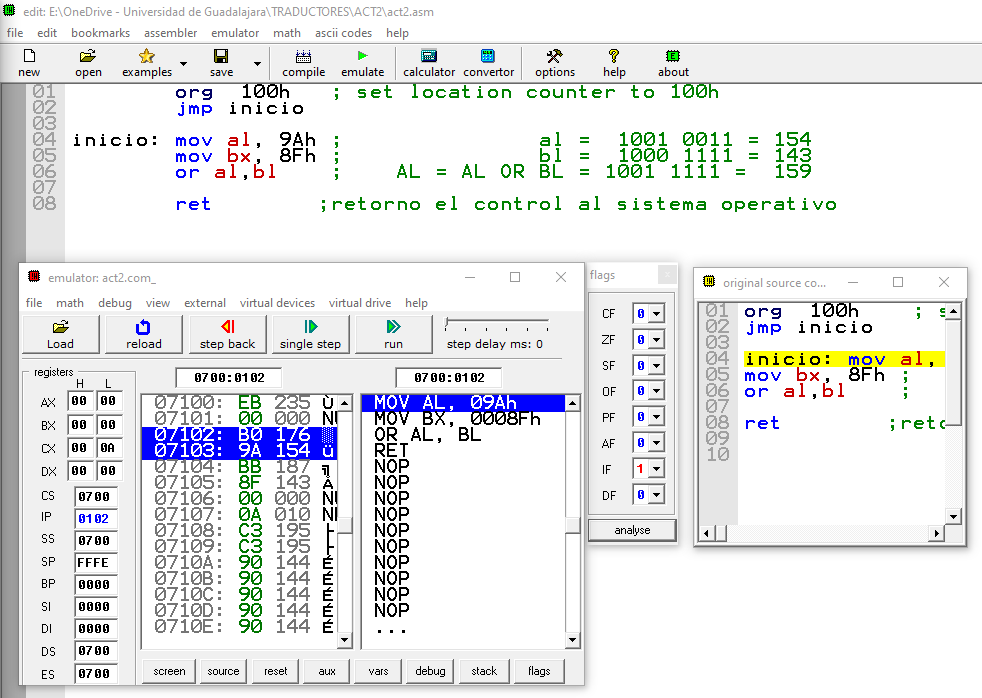
Dando por finalizando el programa de la operación NEG.

**OR**

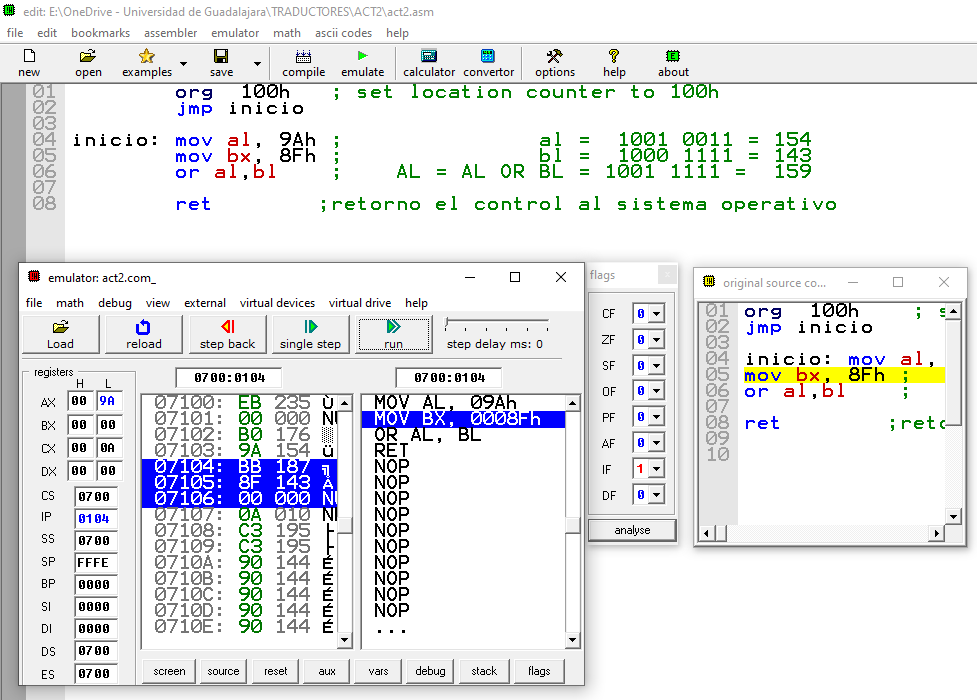
Se realizo la operación OR en donde se introducirá dos valores en diferentes registros de nuestra elección para su respectiva operación, para que a su vez obtener su resultado donde se almacenará en su destino de la operación.



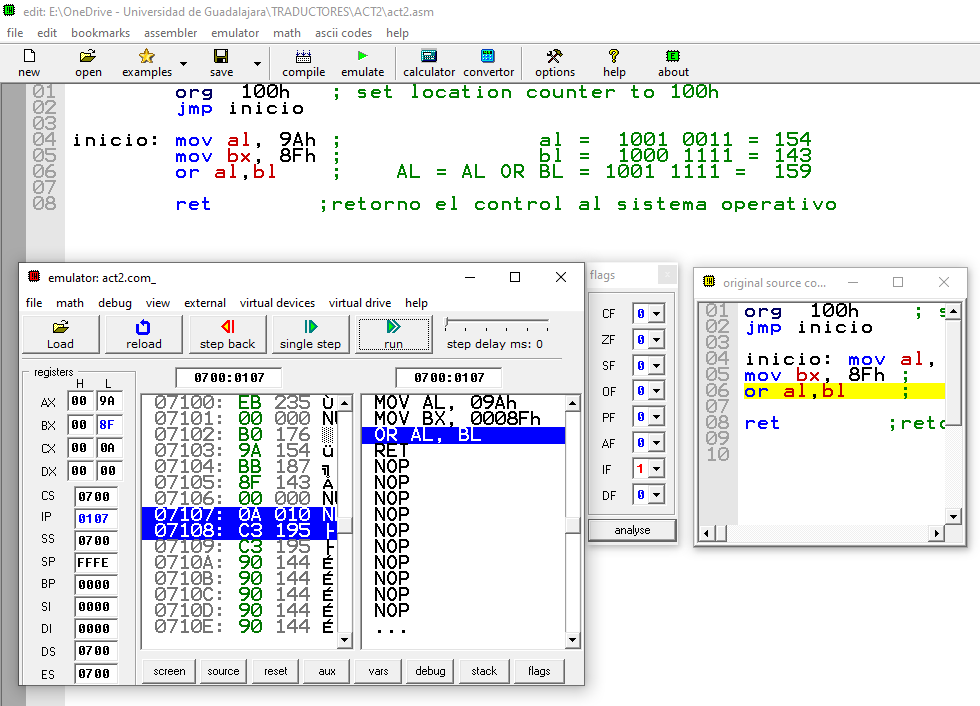
Iniciando la simulación, leerá la primera instrucción en donde saltará a nuestra etiqueta de inicio para leer la instrucción en esa línea de código. Por lo que nuestras banderas están en 0´s porque apenas iniciamos, a excepción de la IF (interrupción) que estará habilitado para nuestro análisis en esta práctica.



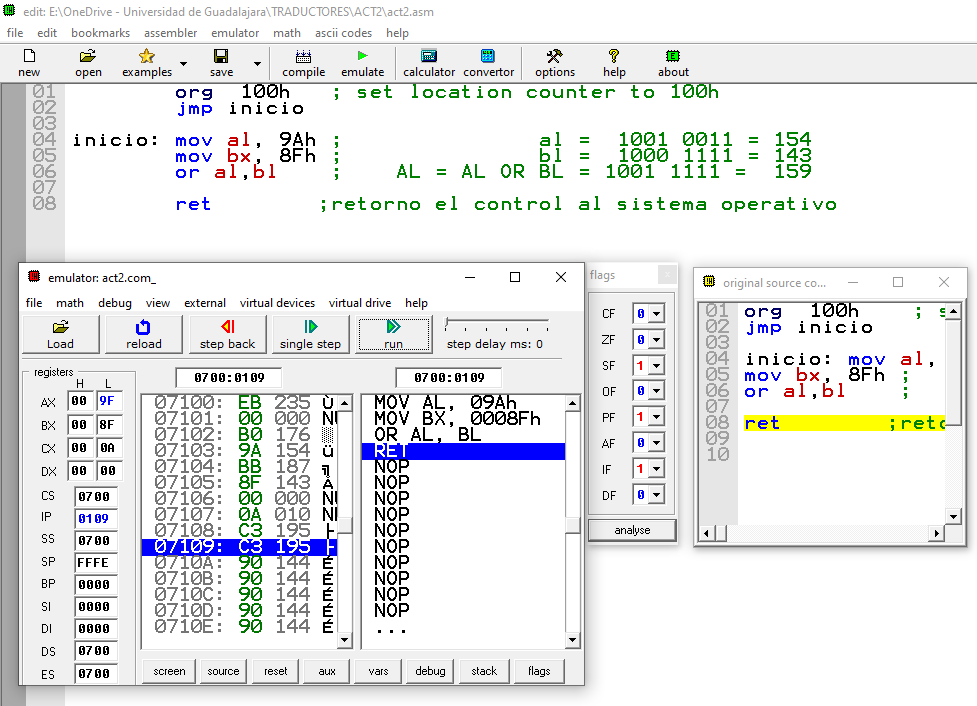
Ahora que se movió en esa línea de código, MOV transferirá el valor 9A(hexadecimal) al acumulador bajo para almacenar el valor. Las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



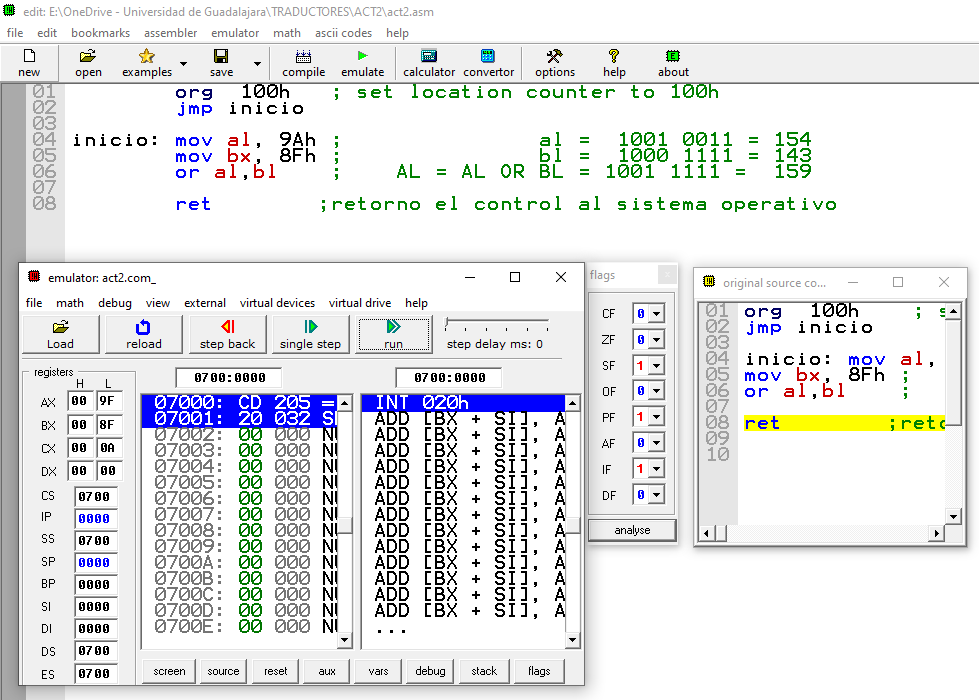
Una vez ejecutado la anterior instrucción, podemos observar que el valor 9A(hexadecimal) ya se almacenó en el AL (acumulador en su estado bajo). Ahora el siguiente MOV transferirá el valor 8F(hexadecimal) al índice base bajo para almacenar su valor.



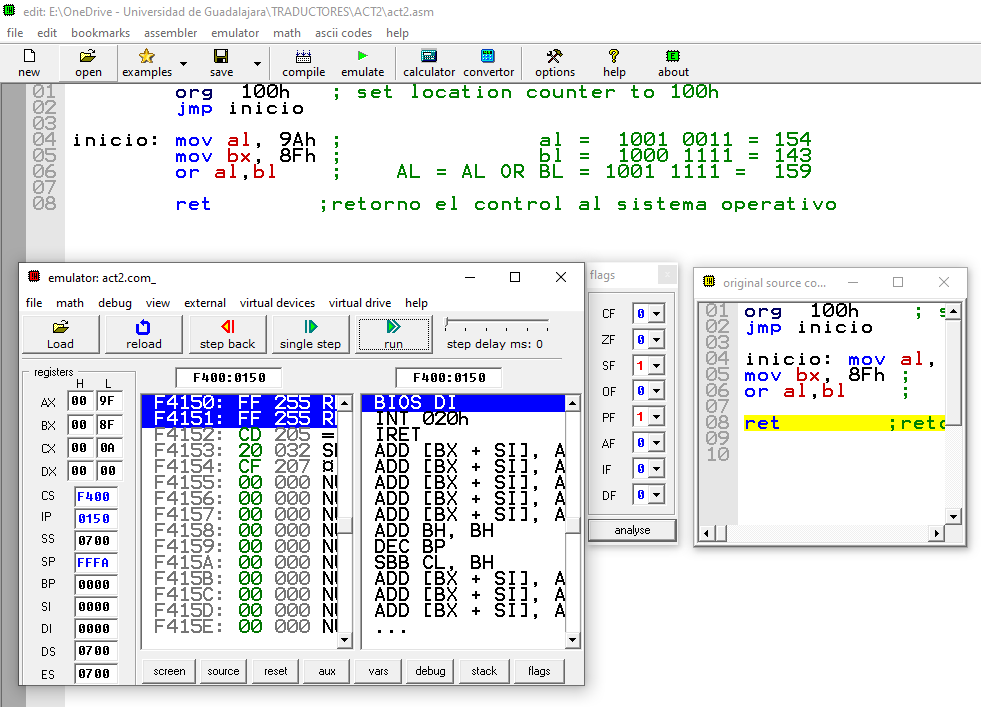
Se puede observar que se almaceno el valor 8F(hexadecimal) en el BL (índice base bajo). Y las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



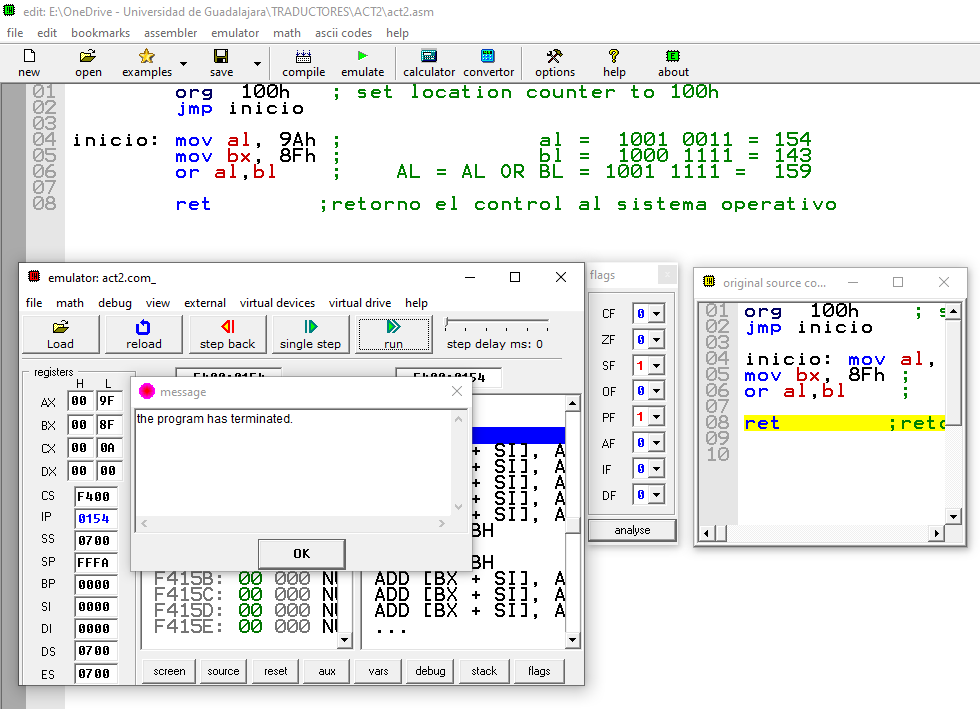
Una vez que nuestros valores se almacenaron se realizara la operación OR para tener como resultado el 9F(hexadecimal), y este valor se ha almacenado en el mismo acumulador base, que era nuestro destino en la operación. Además, de que sus banderas cambiaron, indicando que CF no lleva un acarreo o un arrastre, ZF es un numero diferente a cero, SF es un numero negativo (hay un 1 hasta la izq.), OF no se ha desbordado el número, PF es número par, AF no llevamos un acarreo o un arrastre en el bit 3 y 4, IF siempre tendremos interrupciones para el análisis, DF nuestros registros incrementaran solos.



Ahora realizara el retorno para que el sistema operativo tenga el control, una vez leído la instrucción de RET.



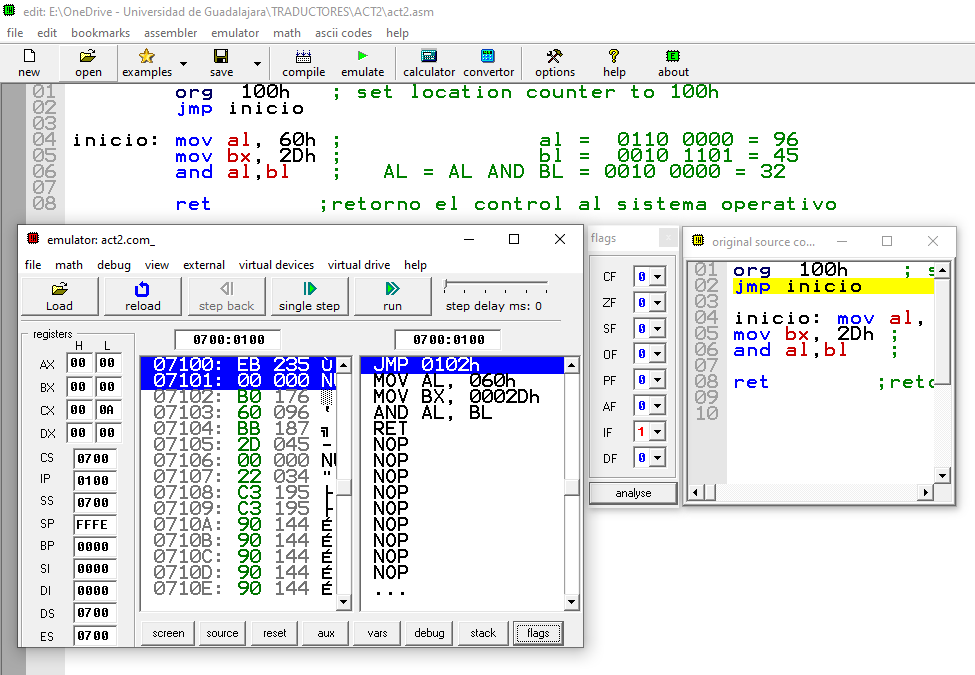
En el que aún está realizando el proceso del retorno del control al sistema operativo.



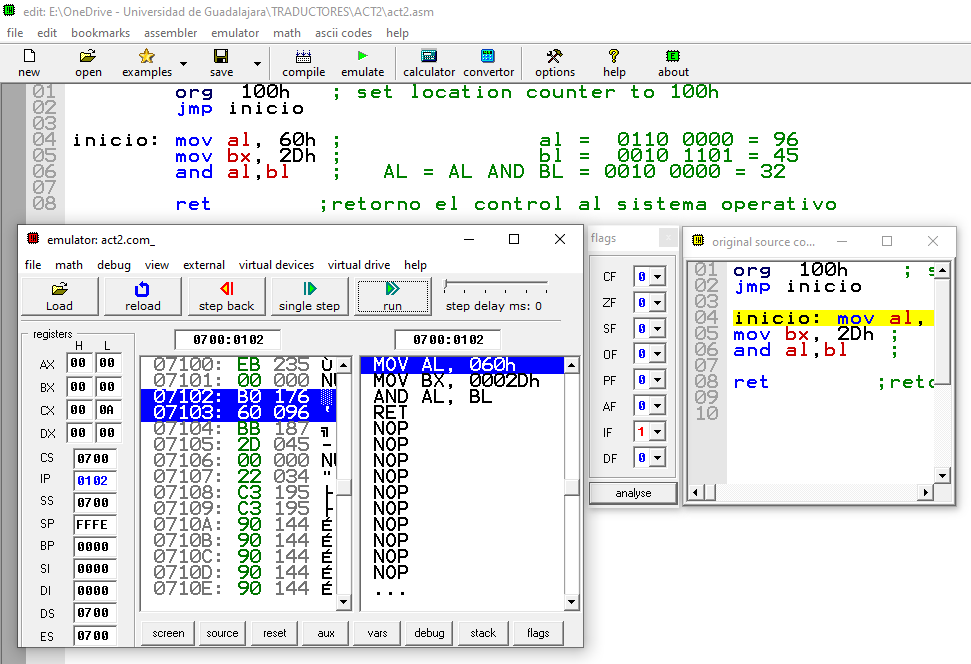
Dando por finalizando el programa de la operación OR.

**AND**

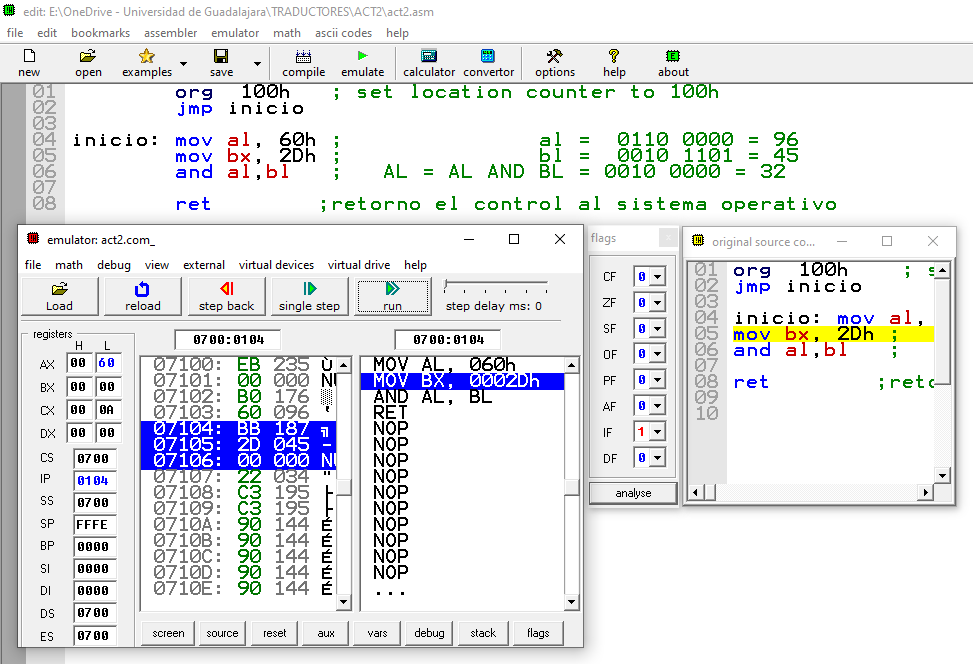
Se realizo la operación AND en donde se introducirá dos valores en diferentes registros de nuestra elección para su respectiva operación, para que a su vez obtener su resultado donde se almacenará en su destino de la operación.



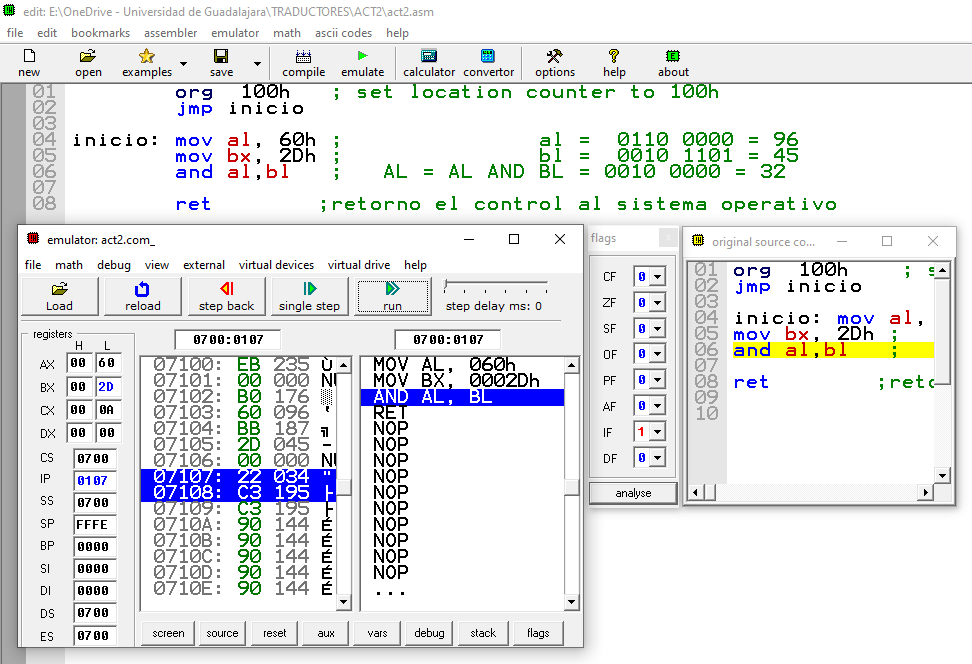
Iniciando la simulación, leerá la primera instrucción en donde saltará a nuestra etiqueta de inicio para leer la instrucción en esa línea de código. Por lo que nuestras banderas están en 0´s porque apenas iniciamos, a excepción de la IF (interrupción) que estará habilitado para nuestro análisis en esta práctica.



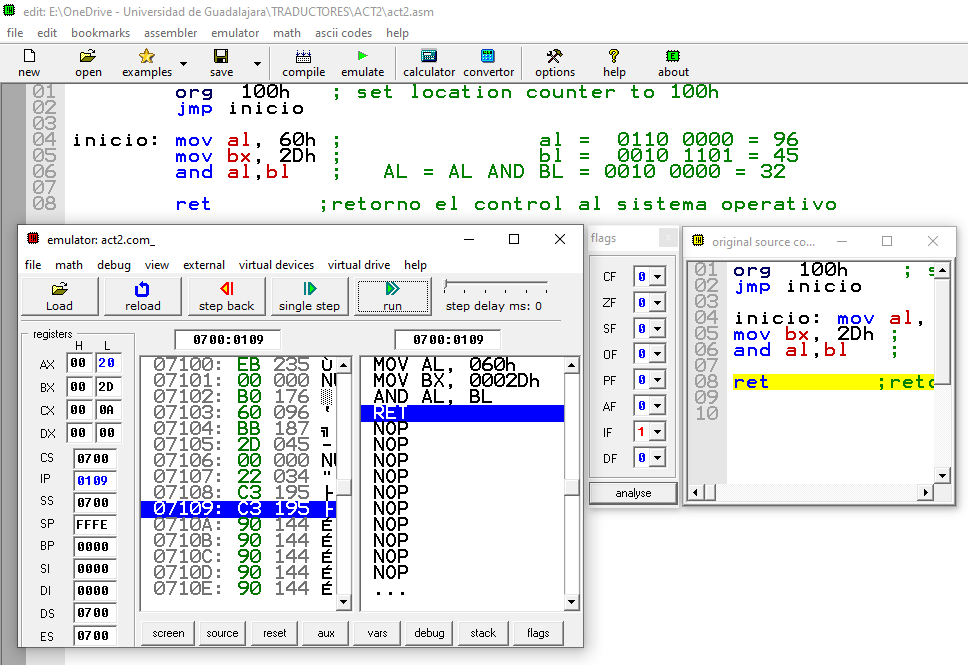
Ahora que se movió en esa línea de código, MOV transferirá el valor 60(hexadecimal) al acumulador bajo para almacenar el valor. Las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



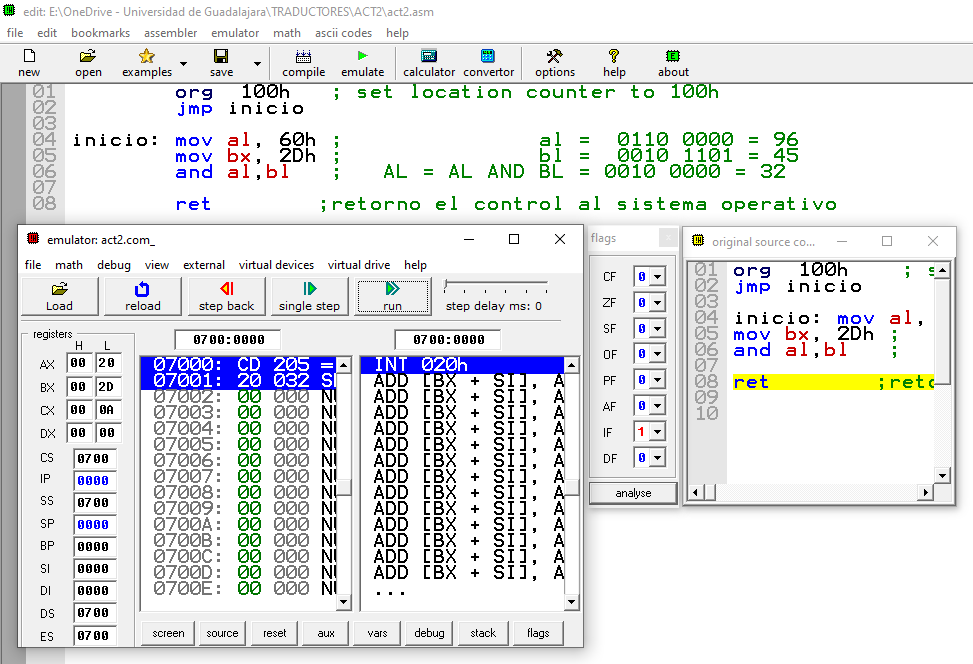
Una vez ejecutado la anterior instrucción, podemos observar que el valor 60(hexadecimal) ya se almacenó en el AL (acumulador en su estado bajo). Ahora el siguiente MOV transferirá el valor 2D(hexadecimal) al índice base bajo para almacenar su valor.



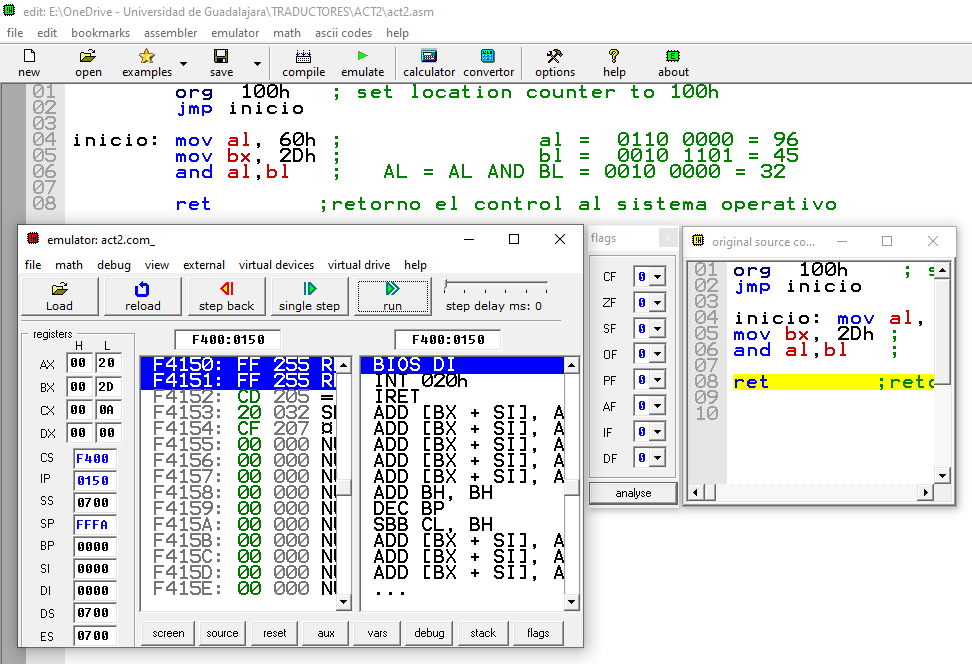
Se puede observar que se almaceno el valor 2D(hexadecimal) en el BL (índice base bajo). Y las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



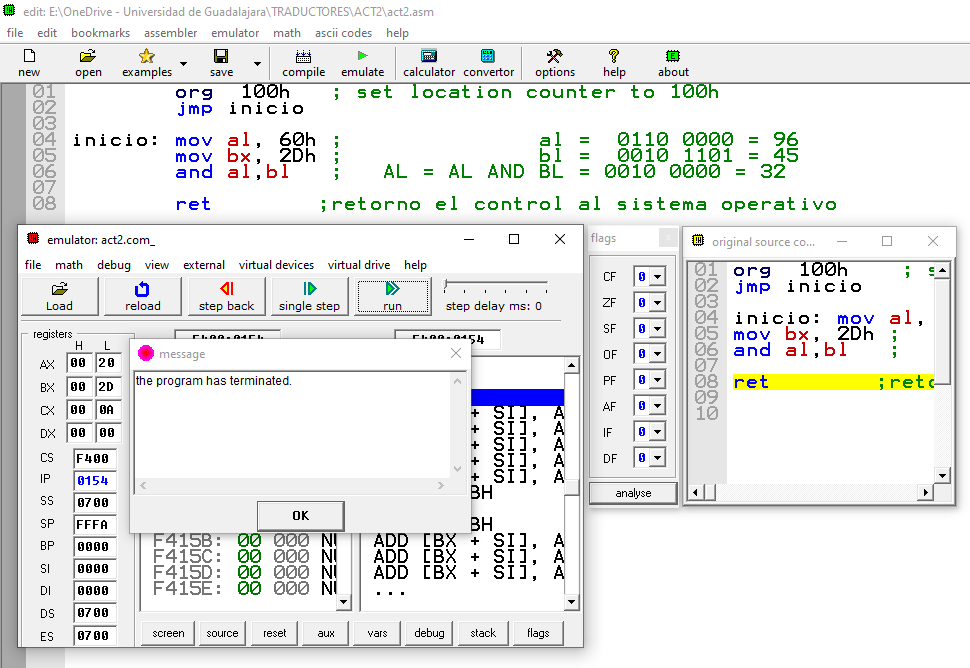
Una vez que nuestros valores se almacenaron se realizara la operación AND para tener como resultado el 20(hexadecimal), y este valor se ha almacenado en el mismo acumulador base, que era nuestro destino en la operación. Además, de que sus banderas no cambiaron debido a que no fue necesario, indicándonos que CF no lleva un acarreo o un arrastre, ZF es un numero diferente a cero, SF es un numero positivo, OF no se ha desbordado el número, PF es número impar, AF no llevamos un acarreo o un arrastre en el bit 3 y 4, IF siempre tendremos interrupciones para el análisis, DF nuestros registros incrementaran solos.



Ahora realizara el retorno para que el sistema operativo tenga el control, una vez leído la instrucción de RET.



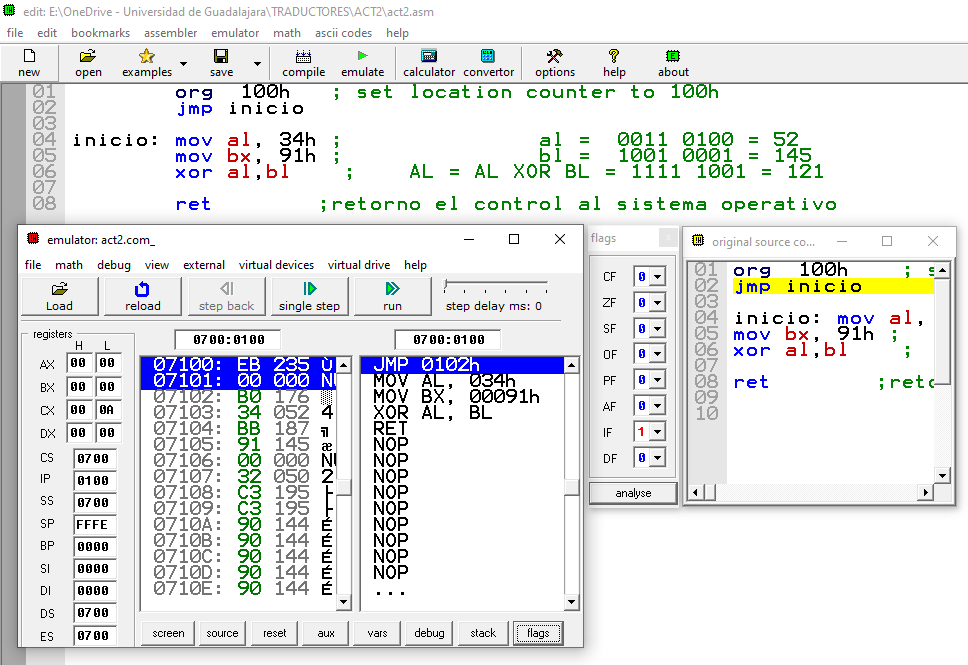
En el que aún está realizando el proceso del retorno del control al sistema operativo.



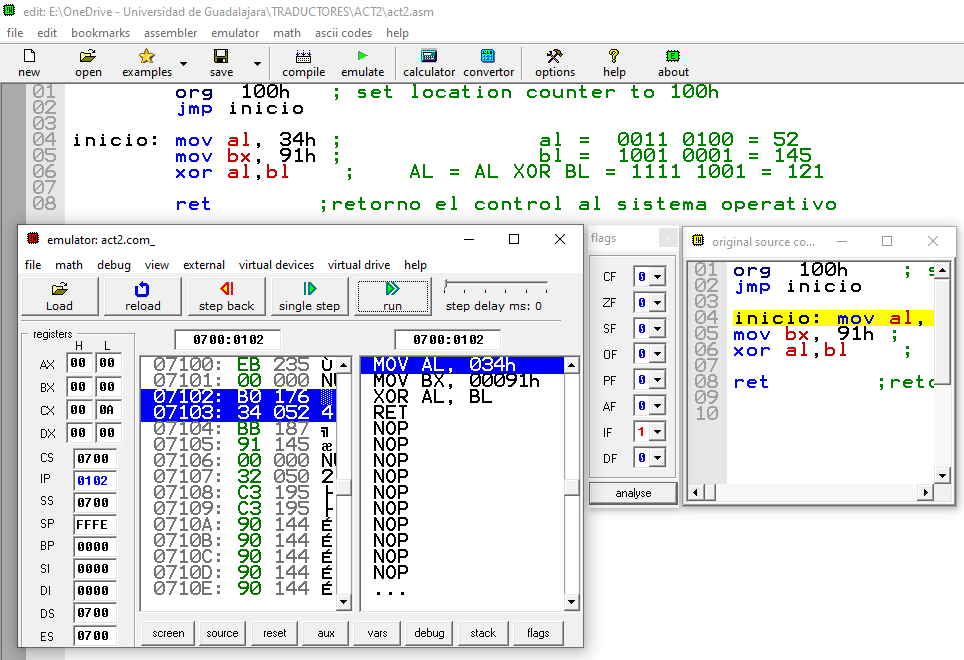
Dando por finalizando el programa de la operación AND.

**XOR**

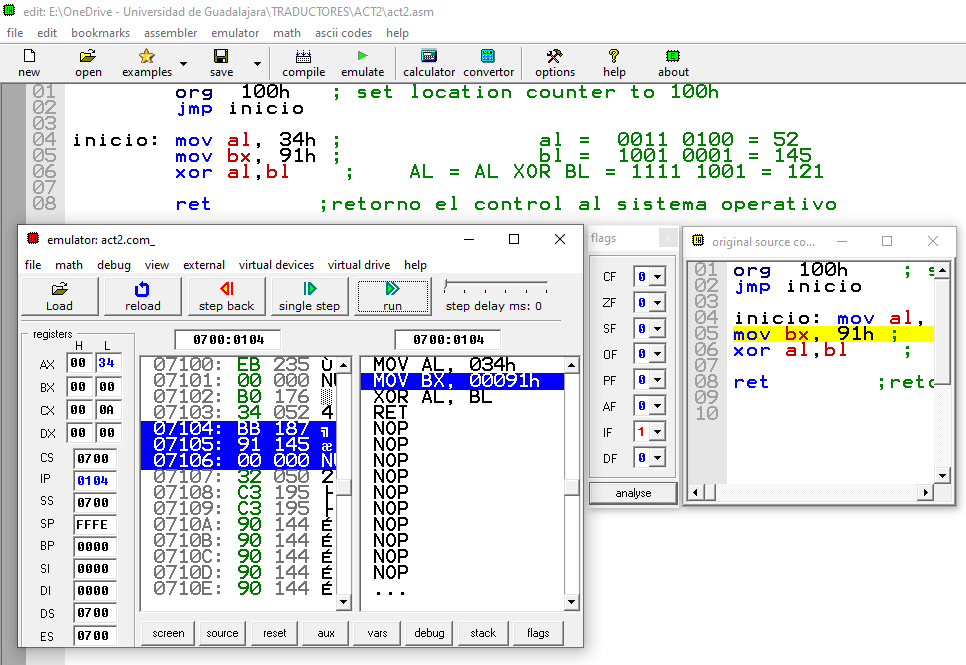
Se realizo la operación XOR en donde se introducirá dos valores en diferentes registros de nuestra elección para su respectiva operación, para que a su vez obtener su resultado donde se almacenará en su destino de la operación.



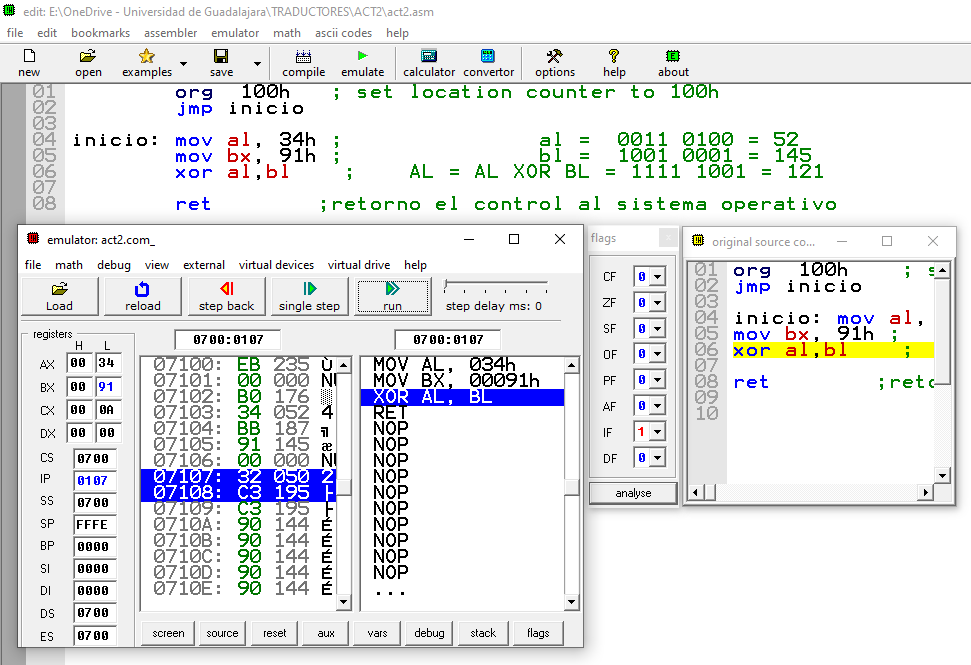
Iniciando la simulación, leerá la primera instrucción en donde saltará a nuestra etiqueta de inicio para leer la instrucción en esa línea de código. Por lo que nuestras banderas están en 0´s porque apenas iniciamos, a excepción de la IF (interrupción) que estará habilitado para nuestro análisis en esta práctica.



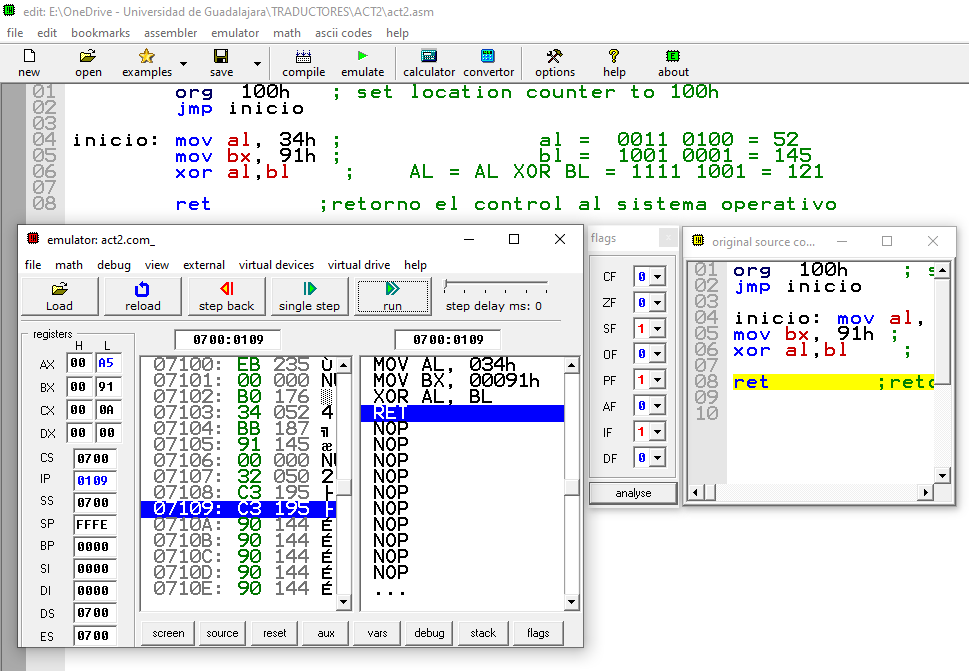
Ahora que se movió en esa línea de código, MOV transferirá el valor 34(hexadecimal) al acumulador bajo para almacenar el valor. Las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



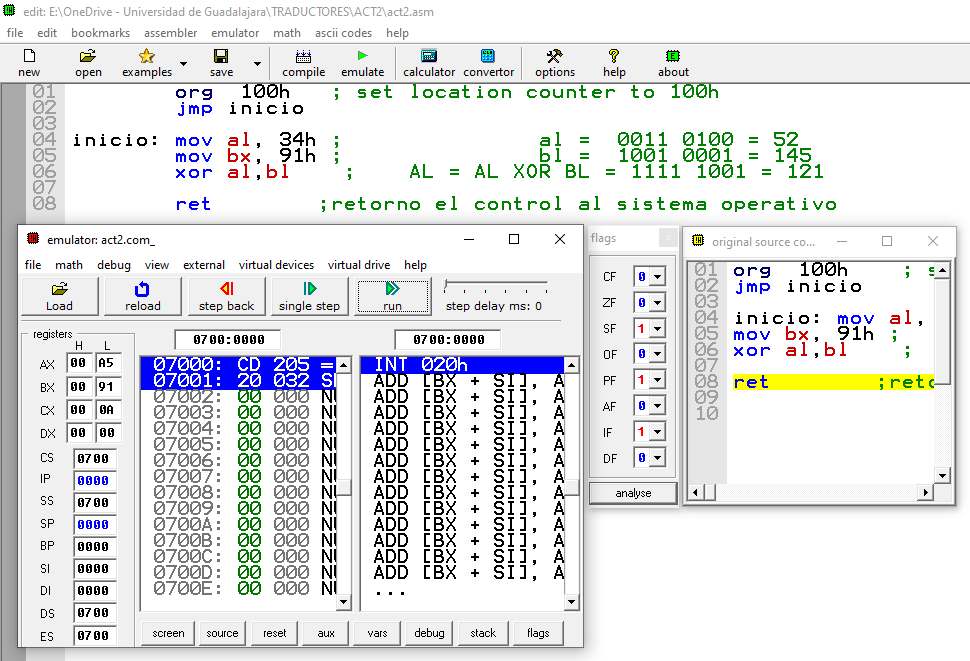
Una vez ejecutado la anterior instrucción, podemos observar que el valor 34(hexadecimal) ya se almacenó en el AL (acumulador en su estado bajo). Ahora el siguiente MOV transferirá el valor 91(hexadecimal) al índice base bajo para almacenar su valor.



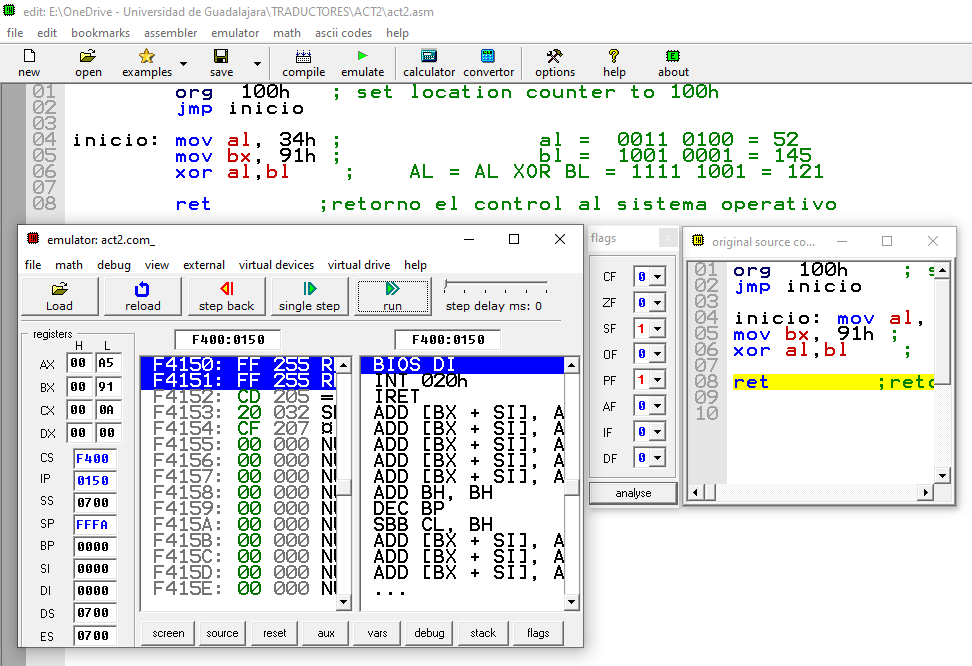
Se puede observar que se almaceno el valor 91(hexadecimal) en el BL (índice base bajo). Y las banderas aun no cambian debido a que aún no se ejecuta una operación correspondiente.



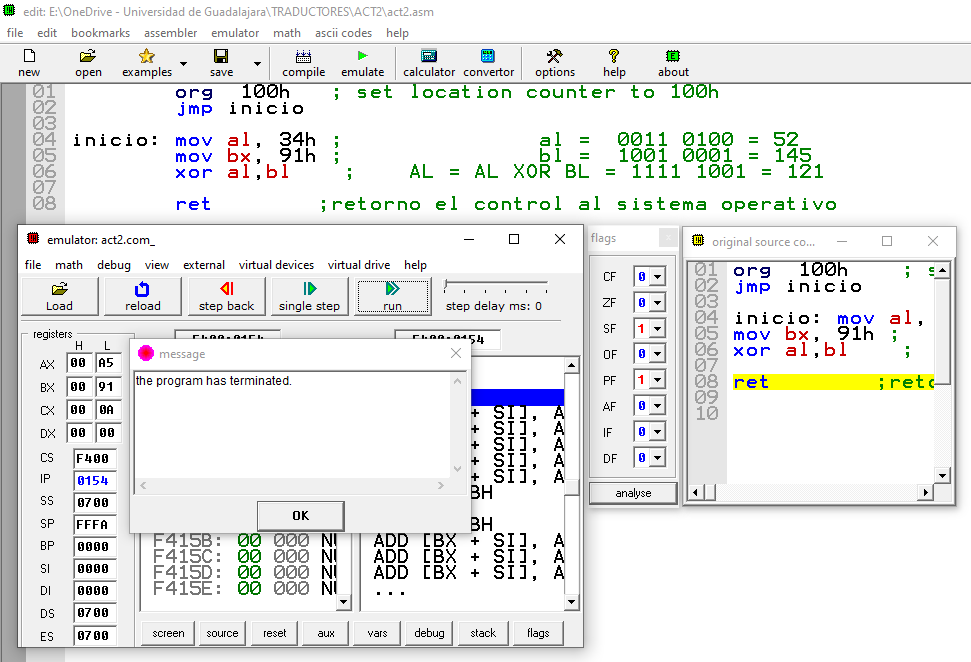
Una vez que nuestros valores se almacenaron se realizara la operación XOR para tener como resultado el A5(hexadecimal), y este valor se ha almacenado en el mismo acumulador base, que era nuestro destino en la operación. Además, de que sus banderas cambiaron, indicando que CF no lleva un acarreo o un arrastre, ZF es un numero diferente a cero, SF es un numero negativo (hay un 1 hasta la izq.), OF no se ha desbordado el número, PF es número par, AF no llevamos un acarreo o un arrastre en el bit 3 y 4, IF siempre tendremos interrupciones para el análisis, DF nuestros registros incrementaran solos.



Ahora realizara el retorno para que el sistema operativo tenga el control, una vez leído la instrucción de RET.



En el que aún está realizando el proceso del retorno del control al sistema operativo.



Dando por finalizando el programa de la operación XOR.

**Conclusión**

La realización de esta actividad me ha ayudado a practicar para utilizar el lenguaje ensamblador en el emu 8086, ya que pude utilizar las operaciones lógicas básicas ya vistas en la carrera, así como aplicar lo que investigué de la anterior actividad. Además, pude identificar y apreciar los cambios de acuerdo con las operaciones realizadas en las banderas del microprocesador, que nos permite conocer el estado que se encuentra como si es par, si tiene algún acarreo, el signo o si es cero los resultados que obtengamos de las operaciones. Finalmente, aprendí todo lo que conlleva realizar las operaciones en el lenguaje ensamblador, lo que es muy parecido al lenguaje máquina, permitiéndonos administrar los recursos.

**Bibliografía:**

Brey, B. B. (2006). Microprocesadores Intel - 7 Edición (7a). Pearson Publications Company.