Sistema web de detección de copias en prácticas de programación

Pedro A. Castillo [1], A. Cañas Vargas [1], Juan J. Castillo-Valdivieso [2], A. Prieto [1]
[1] Dpto. de Arquitectura y Tecnología de los Computadores
ETSI Informática
Universidad de Granada
[2] Dpto. De Tecnología, IES Rey Carlos III, Aguilas (Murcia)
pedro@atc.ugr.es

Resumen

Llegado el momento de corregir las prácticas de programación de una asignatura podemos encontrarnos con que algunos alumnos hayan copiado. Comparar archivos de texto completos es simple y suficiente para detectar copias completas exactas, sin embargo, la detección de copias parciales es una tarea más compleja.

En este trabajo se presenta un sistema web que facilita el uso del servicio MOSS, de forma que el profesor pueda usarlo mediante una interfaz fácil e intuitiva, sin necesidad de saber lenguaje Perl ni teclear órdenes. El sistema propuesto facilita la gestión de los ficheros a comparar, realiza la petición a MOSS para que lleve a cabo la comparación, y por último muestra en una ventana del navegador los resultados de la comparación.

El sistema propuesto se ha utilizado por primera vez en las prácticas de programación en ensamblador de Estructura de los Computadores I durante el curso 2005/2006 habiendo resultado muy eficaz en la detección de copias.

1. Motivación

Las prácticas de programación se prestan especialmente a la copia. Muchos alumnos, al entregar la práctica, toman "prestados" algunas secciones de los programas de otros compañeros, e incluso pueden entregar la práctica de un compañero de años pasados. En general, las copias se intentan ocultar añadiendo o quitando comentarios, cambiando los nombres de variables o identificadores, o cambiando el orden de los módulos [1,2].

Para el profesor puede resultar complicado revisar el código de decenas de alumnos para detectar posibles copias. Es más, si las prácticas se llevan a cabo a partir del código entregado por el profesor, los ficheros de código coincidirán en muchas líneas, sin que haya realmente una copia.

Conviene someter a todos los archivos de código fuente a un procesamiento automático, tanto respecto al resto de prácticas entregadas, como respecto a archivos entregados en años anteriores. Además, hay que realizar una comparación inteligente, que no se limite a comparar y buscar coincidencias en cadenas de caracteres, sino que tenga en cuenta la estructura lógica de los programas.

Existen bastantes aplicaciones que permiten detectar copias. Las basadas en métricas de recuento de atributos mantienen una serie de contadores con algunas características del programa [2,3]. Las basadas en métricas de estructura realizan un análisis del código [4,1,5,6].

En este trabajo se presenta un sistema web que se basa en el servicio MOSS (Measure Of Software Similarity) [4,7] para realizar la detección de copias. Este sistema determina la similitud entre archivos de código en los siguientes lenguajes: C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, HCL2 y ensamblador.

El sistema de detección de copias (SDC) se basa en una interfaz web muy sencilla. Así, un usuario sin conocimientos de programación obtiene la ayuda necesaria para recopilar los archivos de código, enviarlos al sistema, procesarlos e interpretar los resultados (determinar en qué casos hay copia y en cuáles no la hay).

El resto del artículo está estructurado como sigue: en la siguiente sección se detalla el funcionamiento del sistema MOSS. En la sección 3 se describe el sistema web desarrollado. La cuarta sección muestra cómo se ha usado en la corrección de prácticas de ensamblador de 8086

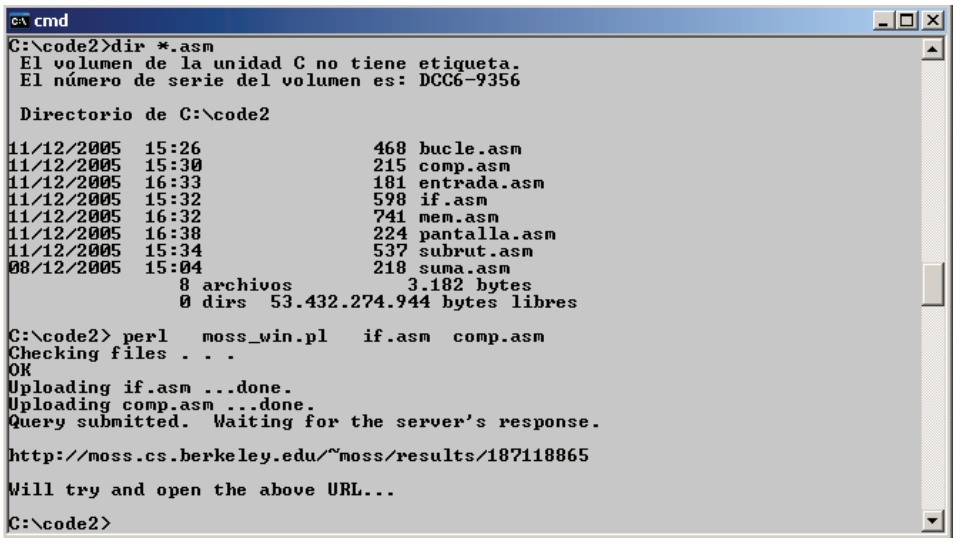

Figura 1. Ejemplo de uso del script MOSS para comparar dos programas de ensamblador. Se debe disponer de un intérprete de Perl correctamente instalado y configurado, y se trabaja mediante línea de comandos

en enero de 2006. Y por último, la sección 5 expone una serie de conclusiones y mejoras al sistema.

2. MOSS: Measure Of Software Similarity

MOSS es un sistema automático para determinar la similitud entre archivos de código fuente escritos en diversos lenguajes de programación. Desde 1994 se viene usando para detectar copias en clase de programación, habiendo resultado muy efectivo.

En lugar de limitarse a buscar trozos de código de un archivo en otros, el sistema realiza un preprocesamiento [7,8,9] para eliminar los signos de puntuación y los caracteres espaciadores. A continuación todo el texto se convierte a minúsculas y sustituye todos los nombres de variables por el identificador "V". De esta forma nos aseguramos de que se utilizará la información semántica del documento para llevar a cabo la comparación y determinar si hay o no copia (en archivos de texto y código, tras eliminar las partes no útiles, la coincidencia de subcadenas puede ser suficiente).

Para utilizar el programa, el nuevo usuario debe obtener un identificador siguiendo las instrucciones que se ofrecen en la página web [4]. Recibirá un script (programa) escrito en Perl que debe modificar especificando su identificador, y a partir de ahí, ya puede utilizarlo.

Sin embargo, a pesar de su potencia, el uso del sistema MOSS resulta complicado para usuarios con pocos conocimientos de programación: hay que instalar un intérprete de lenguaje Perl, y a través de una interfaz de línea de comandos, ir ejecutando dicho script para realiar las comprobaciones.

La Figura 1 muestra una ejecución del script para comprobar si dos archivos de ensamblador (IF.ASM y COMP.ASM) son similares. Vemos que usar el sistema a través de este script de línea de comandos puede resultar engorroso.

MOSS permite comparar no sólo dos archivos sino tantos como necesitemos. Una vez procesados todos los archivos, obtendremos un conjunto de páginas web en las que se nos muestra qué partes de qué archivos son muy similares. El servicio muestra un porcentaje de coincidencia entre archivos, y además permite al usuario ver las secciones de código resaltadas en los textos originales (ver Figuras 6 y 7).

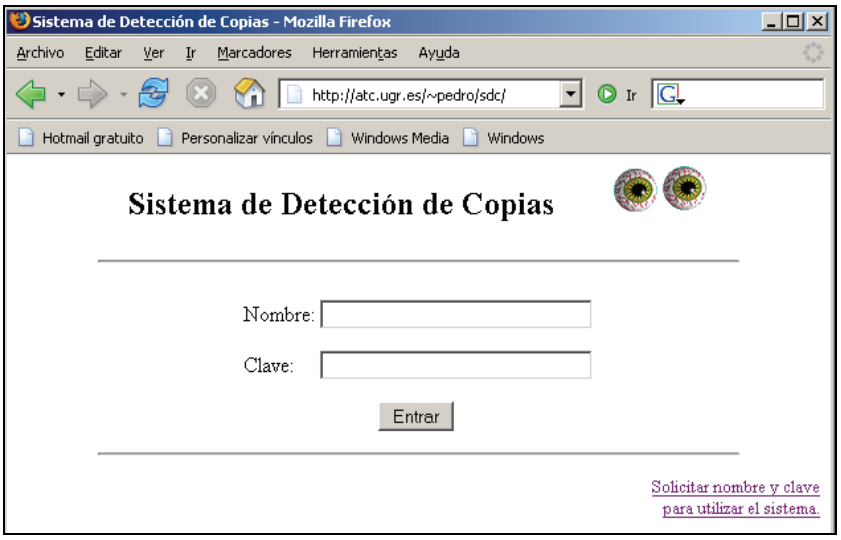

Figura 2. Pantalla de entrada al sistema SDC. Se requiere una identificación previa. El nombre de usuario y clave se puede solicitar a través del enlace que aparece al final de la página de entrada

Por otro lado, hay casos en los que el profesor entrega código a los alumnos, a partir del cual desarrollan sus programas. Conviene indicarle al sistema que esas líneas de código no se tengan en cuenta a la hora de comprobar posibles copias, ya que es muy probable que aparezcan en todos los archivos, y el profesor no lo considerará copia. SDC permite adjuntar un archivo que contenga las líneas de código que MOSS debe ignorar al llevar a cabo la comprobación.

3. SDC: Sistema de Detección de Copias

La interfaz web de SDC está desarrollada en PHP. Son un conjunto de scripts muy ligeros que se ejecutan de forma muy rápida. El sistema se ejecuta en el servidor web del Departamento de Arquitectura y Tecnología de los Computadores. Se puede acceder a través de la URL `http://atc.ugr.es/~pedro/sdc/`

Cada profesor dispone de una cuenta en el sistema pudiendo almacenar tantos archivos como quiera comparar posteriormente. Para evitar copias entre alumnos de diferentes grupos, se pueden crear cuentas comunes a todos los profesores de una asignatura, para que pongan en común todas las prácticas para cotejarlas.

Las Figuras 2 a 7 muestran un ejemplo real de utilización del sistema (identificación, añadir archivos o eliminarlos, comprobar si existen copias, e interpretar los resultados del sistema).

Una vez ha subido al sistema todas las prácticas, puede ejecutar la comprobación de copias, en la cual, SDC envía todos los datos al servicio MOSS para que realice la comprobación, y éste devuelve un conjunto de páginas web en las que veremos qué archivos tienen líneas en común.

Para realizar varias comprobaciones diferentes, puede eliminar todos los archivos que tiene en un momento dado, para subir otros nuevos (los de otra clase de prácticas, por ejemplo).

Cuando el conjunto de archivos a comprobar es muy grande, resulta muy engorroso ir añadiéndolos uno a uno. Para estos casos, SDC permite el envío de ficheros comprimidos en formato ZIP. De esta forma, en nuestro ordenador local incluimos todas las prácticas en un fichero comprimido, y sólo tenemos que subir ese fichero al sistema, que detecta si es un fichero ZIP, y en ese caso lo descomprime.

En el caso de que el profesor haya entregado código común a todos los alumnos para facilitar la realización de las prácticas, se puede adjuntar un fichero que contenga todo el código y posteriormente indicar al sistema que queremos tener en cuenta ese código para que no sea detectado como copia.

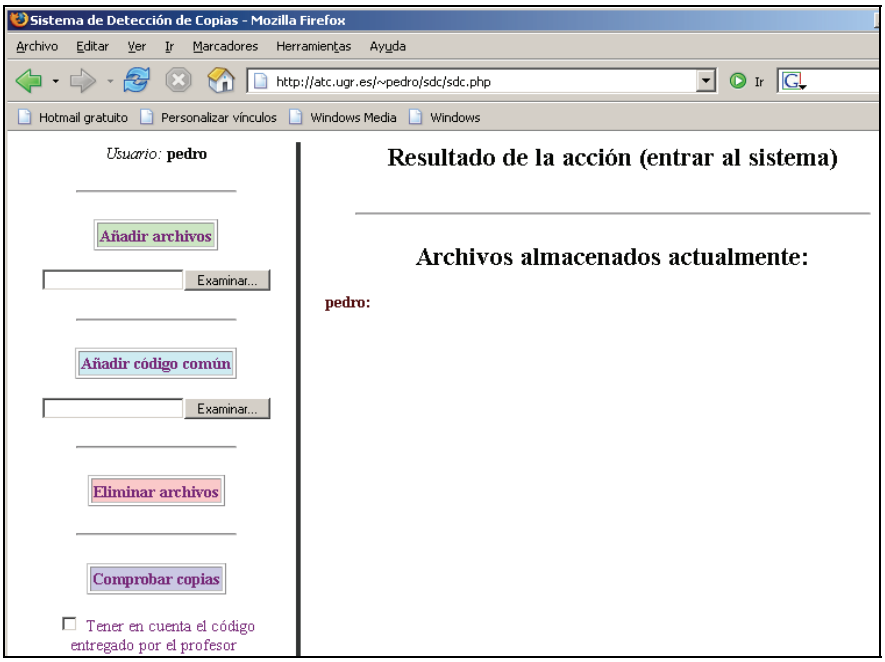

Figura 3. Pantalla de opciones del sistema SDC. Podemos añadir nuevos archivos de código, eliminar todos los almacenados en el sistema actualmente, o llevar a cabo la comprobación de copia entre dichos archivos

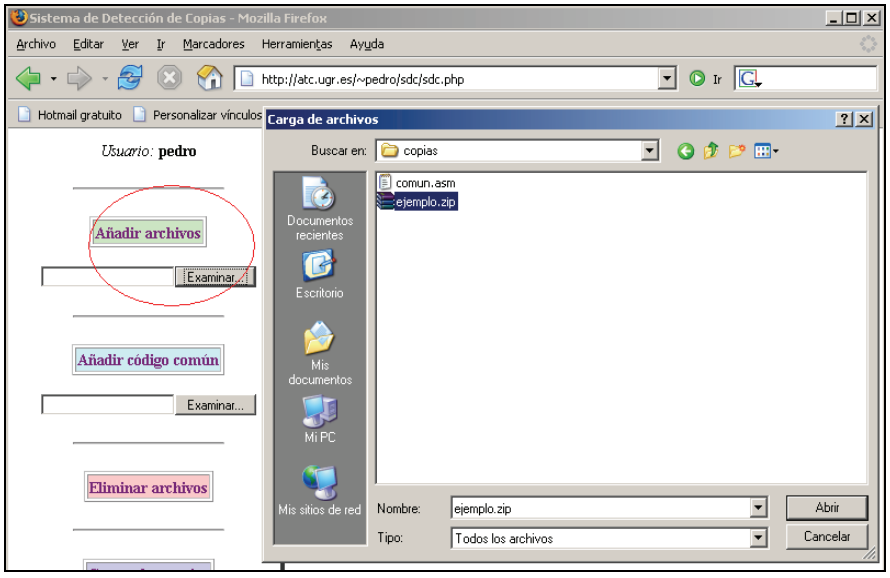

Figura 4. Mediante la opción de "Añadir archivos" podemos añadir nuevas prácticas al conjunto que guarda el sistema para cada usuario. El total de archivos aparece en la ventana, a la derecha

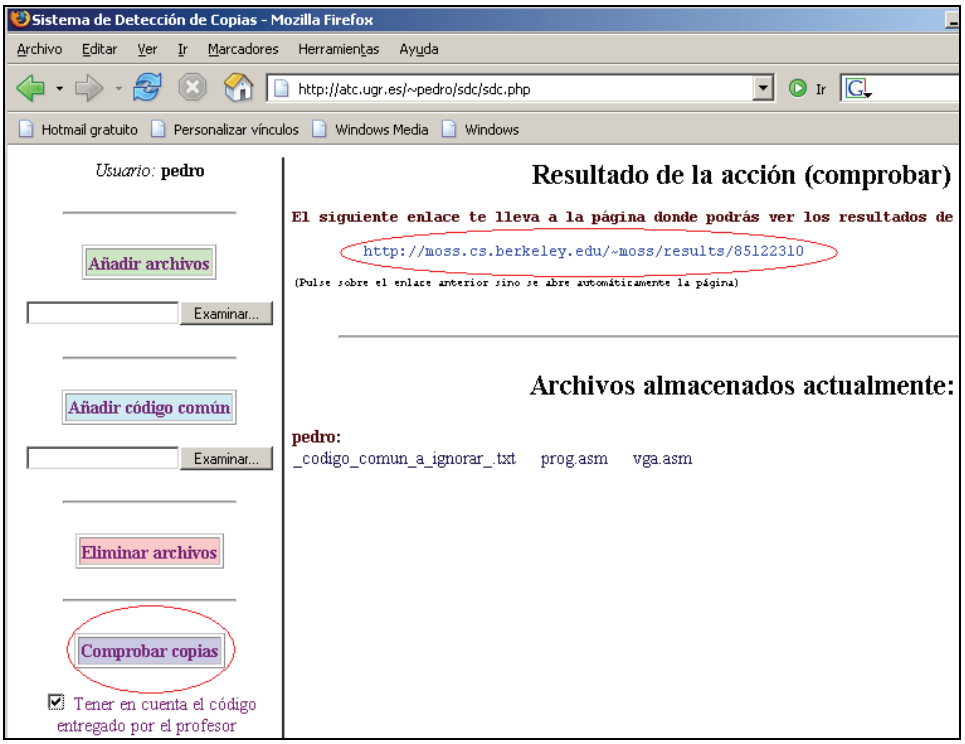

Figura 5. Mediante la opción de "Comprobar copias" enviamos al servicio MOSS los archivos almacenados hasta el momento, y recibimos un enlace a través del cual veremos los resultados

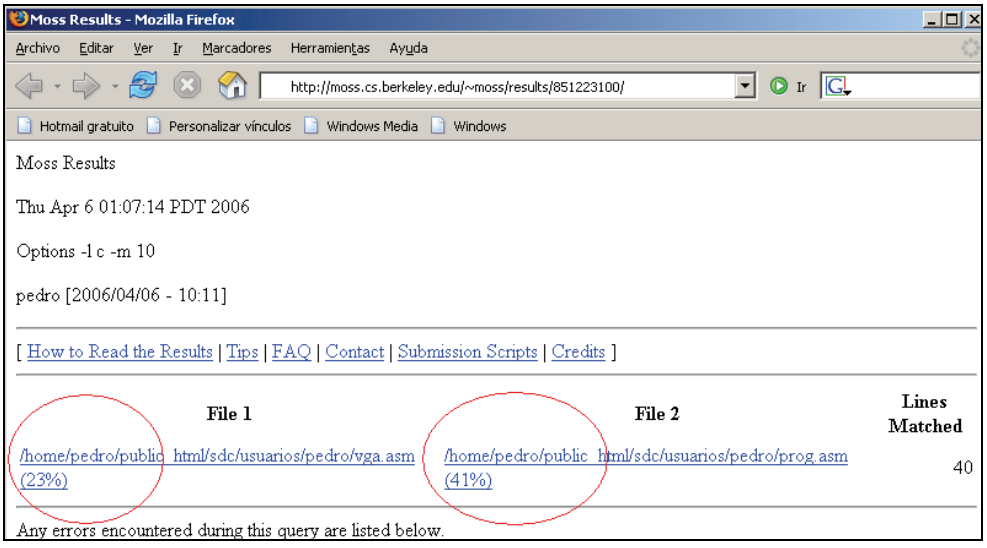

Figura 6. La pantalla principal de MOSS muestra el identificador de la petición (usuario, fecha y hora de petición), junto con la lista de comparaciones y porcentajes de coincidencia entre los diferentes archivos

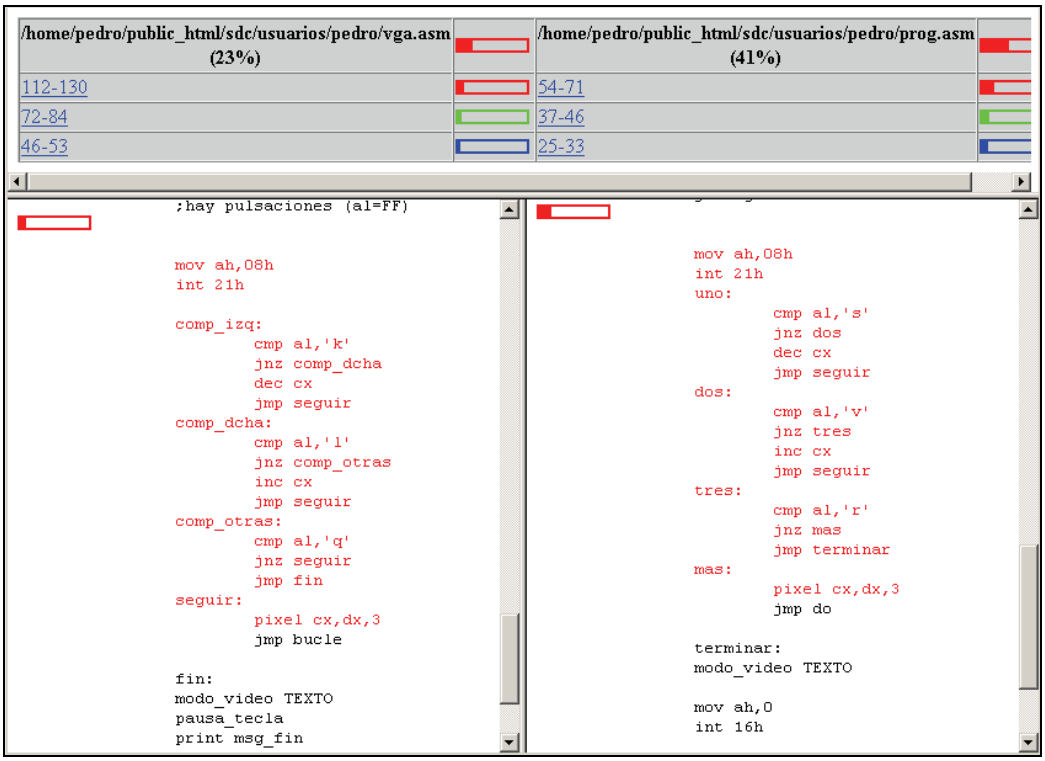

Figura 7. Los enlaces de la primera pantalla nos llevan a otra donde veremos comparados línea a línea cada par de archivos, y resaltadas en colores las zonas en que coinciden. A partir de este análisis automático, el profesor debe decidir si realmente existe copia o no entre los archivos indicados por el sistema

4. Ejemplo de aplicación de SDC

El sistema propuesto se ha utilizado por primera vez en las prácticas de Estructura de los Computadores I (asignatura de Ingeniería Informática de la Universidad de Granada) durante el curso 2005/2006.

En la descripción de cómo usar el sistema de la sección anterior, se han utilizado dos prácticas realizadas durante este curso. En este caso, los alumnos disponían de unas subrutinas ya probadas que podían utilizar si lo necesitaban. Al realizar la comprobación global de todas las prácticas, el sistema alertó de la similitud entre dos archivos (PROG.ASM y VGA.ASM).

La Figura 8 muestra una porción de código que SDC estimó que coincidía entre ambas prácticas.

Las Figuras 3 y 4 muestran el modo de subir archivos al sistema para realizar posteriormente la comparación.

La Figura 5 muestra cómo se realiza la llamada a MOSS y la Figura 6 muestra los resultados de la comparación entre ambos ficheros. Vemos que PROG.ASM tiene un 60% de código en común con VGA.ASM

Al seguir ese enlace, la página en la que se comparan ambos archivos (Figura 7) nos mostró qué partes coincidían. Ya que había material común entregado por el profesor, el sistema ignoró esos módulos que aparecían en todos los archivos. Esa página muestra los códigos originales, resaltando en colores las partes comunes.

Tras una comprobación por parte del profesor, se llegó a la conclusión de que existía copia (tal y como luego reconocieron los alumnos).

```
. . .
   pixel 1,1,3
   pixel 319,1,3
   mov cx,100
   mov dx,100
   bucle:
       mov ah,0bh
       int 21h
       pixel cx,dx,0
       cmp al,00
       jz seguir
       mov ah,08h
       int 21h
       comp_izq:
           cmp al,'k'
           jnz comp_dcha
           dec cx
           jmp seguir
       comp_dcha:
           cmp al,'l'
           jnz comp_otras
           inc cx
           jmp seguir
       comp_otras:
           cmp al,'q'
           jnz seguir
           jmp fin
       seguir:
           pixel cx,dx,3
           jmp bucle
       fin:
       modo_video TEXTO
. . .
```

```
. . .
    modo_video GRAFICO
    mov cx,6
    mov dx,18
    do:
        mov ah,0bh
        int 21h
        pixel cx,dx,0
        cmp al,00
        jz seguir
        mov ah,08h
        int 21h
        uno:
            cmp al,'s'
            jnz dos
            dec cx
            jmp seguir
        dos:
            cmp al,'v'
            jnz tres
            inc cx
            jmp seguir
        tres:
            cmp al,'r'
            jnz mas
            jmp terminar
        mas:
            pixel cx,dx,3
            jmp do
        terminar:
        modo_video TEXTO
. . .
```

Figura 8. Porción de los archivos PROG.ASM y VGA.ASM (prácticas de Estructura de Computadores I del curso 2005/2006) en los que el sistema descrito encontró numerosas líneas de código copiadas

5. Conclusiones

En este trabajo se ha presentado un sistema de detección automática de copias (SDC) basado en un servicio robusto y muy utilizado (MOSS).

SDC se ha utilizado por primera vez durante el curso 2005/2006 en las prácticas de ensamblador de 8086 de la asignatura de Estructura de los Computadores I.

El sistema resulta muy sencillo de utilizar, permitiendo comprobar tantos archivos como sea necesario mediante una interfaz web muy intuitiva. Los resultados se muestran en una nueva ventana de navegador, mostrando el porcentaje de coincidencia entre archivos, y resaltado en colores las partes comunes.

Además, SDC es robusto para la tarea, ya que al estar basado en MOSS, aún cuando se hicieran modificaciones a los ficheros de las prácticas, las copias seguirían detectándose.

Esperamos que la potencia del sistema de análisis junto con su facilidad de uso sirva de utilidad a los profesores de prácticas a la hora de detectar y evitar posibles copias. El sistema es totalmente aplicable en otros centros y asignaturas, así pues, la implementación realizada

en PHP se ha dejado disponible para aquellos profesores que lo soliciten a los autores.

A pesar de los buenos resultados del servicio MOSS [7], convendría usar más de una herramienta para asegurarnos de que el sistema detecta todas las copias, y además, no nos avisa de falsos positivos. Así pues, como mejora de SDC, se intentará incluir también el sistema JPLAG [6] y procesar los resultados que ofrezcan ambos para hacer una estimación conjunta y más fiable.

Referencias

[1] Pedro J. Clemente, Alberto Gómez, Julia González. La copia de prácticas de programación: el problema y su detección. Actas de las X Jornadas de Enseñanza Universitaria de Informática. JENUI2004. páginas 203-210. Alicante. Julio de 2004.

[2] Whale, Identification of Program Similarity in Large Populations". The Computer Journal 33(2), pp. 140-146 . 1990

[3] Faidhi, Robinson, An Empirical Approach for Detecting Program Similarity within a University Programing Environment, Computers and Ed. 11(1),pp.11-19 . 1987

[4] MOSS. Universidad de Berkeley. URL: www.cs.berkeley.edu/~aiken/moss.html

[5] M. J. Wise, Improved Detection of Similarities in Computer Program and other Texts, Twenty-Seventh SIGCSE Philadelphia, U.S.A., pp. 130-134 . Febrero 15-17, 1996

[6] JPLAG. www.jplag.de

[7] S. Schleimer, D.S. Wilkerson and A. Aiken. Winnowing: Local Algorithms for Document Fingerprinting. SIGMOD2003, June 9-12. ACM 1-58113-634-X/03/06. 2003. URL: theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf

[8] B.S. Baker. On finding duplication and near-duplication in large software systems. In L. Wills, P. Newcomb, and E. Chikofsky, editors. Second Working Conference on Reverse Engineering, pp.86-95, IEEE Computer Society Press. 1995

[9] B.S. Baker and U. Manber. Deducing similarities in java sources from bytecodes. In Proc. Of Usenix Annual Technical Conf., pp.179-190. 1998