Plagiarism Detection of C Program using Assembly Language

Shashank Chauhan
CSE/ IT Department
Jaypee Institute of Information Technology, Noida, India

Anuja Arora
CSE/ IT Department
Jaypee Institute of Information Technology, Noida, India

Yash Singhal
CSE Department
Graphics Era University, Dehradun, India

ABSTRACT
Source code plagiarism is becoming a common practice among higher education community. People duplicate and modify the source code of other people and show the program as their own program. In this paper, we want to draw researchers' attention towards this problem and projected a novel approach which detects plagiarism in C language code by converting it into assembly language which is done with the help of GCC compiler. Assembly language converted by the compiler is not sensitive to all type of different code transformation, for example-swapping variable names, reformation of language, adding extra comment or blanks. Therefore, assembly language gives rise to reduced amount of variations, if there is a modification in the original code. Previous works in plagiarism compares the whole program but in this paper, we proposed a method which split the C program into assembly language code and divide each function of program into blocks and blocks are transformed into token strings. This method compares each function with other program function and provides a statistical output, according to the token string likeness of that function. If the output is above assigned specific plagiarism similarity threshold value then it counts under the case of plagiarism.

Keywords
Plagiarism, assmebly language, string similarity, Plagiarism detection method, token string

1. INTRODUCTION
The word plagiarism is basically derived from a Latin word: plagiarius, which stands for an abductor, and plagiare, which stands for to steal [1]. In other words plagiarism is called wrongful appropriation or stealing the content, language, ideas, thoughts or expressions such as source code, publication of someone else and representing them as their original work.

In same direction, plagiarized program can be described as a program that has been copied from source program with some modification. Modifications, like changing comments, modifying variable position, replacing equivalent code structure does not require program understanding. In the era of education, plagiarism is a longstanding problem. For example, in programming assignments, students have to write codes which are marked according to the correctness and logic. Unfortunately, source code plagiarism is now a child"s play for everyone due to exposure with outside world with the help of World Wide Web, personal computer, computer network and screen editor programming etc. Involvement of students in plagiarism can be contributed to various reasons-peer pressure, wish to help comrade, time management failure, assignment submission, programming phobia, inadequate access to computer system or software. It has become a common practice among students to reuse the source code and generating a visually different code using routine modification, due to which it becomes tough and impossible to detect plagiarism manually.

Researchers have introduced and validated various code plagiarism approaches such as counting method [3], Metric structural method [4, 5], clustering method [6] and many more [7, 8]. Structural method approach is known as the most suitable approach for detecting plagiarism in source code as this method makes use of tokenization and string matching algorithms to detect the similarities among various codes.

Various structural methods exists to find out source code, few of them are as follows- Plague [6], YAP[9] and JPlag [10]. Plague method generates a structure profile for all source codes and compares them. YAP method creates token for all sources codes and compares all source codes token files. JPlag parse all source codes exist in a directory and transform to token string, further token strings are compared using greedy string tilling algorithm. Faidhi and Robinson [2] had explained six level of code transformation as shown in figure 1.

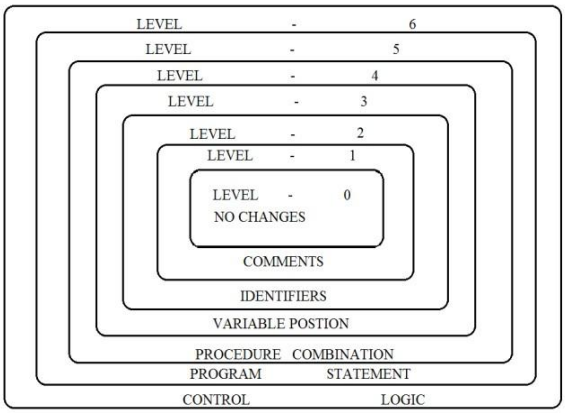

Fig 1: Level of Plagiarism [2]

This procedure is able to detect many cases of plagiarism as presented in the subsequent example shown in figure 2 and figure 3. Level 0 is presented in figure 2 and level 1 is depicted with the help of figure 3. It is observed that code bock "A1" of figure 2 and code block "B1" of figure 3 have same code with few changes. The only change is in the function names, variable names in code block "B1" and few comments have been added in code of figure 3. In code block "A" function is named as "insert" whereas in code block "B" function is named as "put". Code inside both the blocks is same with varying variable names. If users added comments in code statements, then also code does not come under plagiarism by few already available plagiarism detection techniques.

Numbers of algorithm are available in order to detect plagiarism with the help of similarity structural level and even we can use these methods to produce good results. But these methods become useless, if a short program has unnecessary repetition of statements and these processes consumes a lot of time to compare source code and to detect plagiarism.

```
#include<stdio.h>
struct node{
   int value;
   struct node *next;
};
typedef struct node nd;
nd n1,*ptr;
void insert(nd **,int);
void printflist(nd *);
int main()
{struct node *head=NULL;
   insert(&head,7);
   insert(&head,5);
   insert(&head,7);
   insert(&head,5);
   insert(&head,1);
   insert(&head,1);
   printlist(head);
}
void insert(nd **hd,int val)
{nd *newnode=(nd*)malloc(sizeof(nd));
   newnode->value=val;
   newnode->next=NULL;
   if(*hd==NULL)
      *hd=newnode;
   else{
      nd *tmp;
      tmp=*hd;
      while(tmp->next!=NULL)
      {
         tmp=tmp->next;
      }
      tmp->next=newnode;
   }
}
void printlist(nd *head)
{nd *tmp;
      tmp=head;
      while(tmp->next!=NULL)
      {
          printf("%d\n",tmp->value);
         tmp=tmp->next;
      }
      printf("%d",tmp->value);
}
```

A1

Fig 2: Level 0 Example

So, here is a need of efficient approach which can detect plagiarism in code with less time complexity and that is prime motivation behind the projected work. We proposed a novel method which detects plagiarism with the help of assembly language. In the proposed approach targeted source code converts into assembly language which is transformed and generated by the GNU compiler collection (GCC) compiler. This method divides the assembly language code into blocks and blocks are transformed into token strings. This method compares the token string likeness and provides a statistical output.

```
#include<stdio.h>
#include<malloc.h>
int main()  //execution start
{  cy *sir='\0';
          int m,o,p,int s=0;
          float k;
          char b[27];char a[10];
          view:
          printf("Enter the value: ");
          scanf("%d",&m);
          put(&sir,m);
          printf("insert more?(yes/no): \n");
          scanf("%s",a);
          if(ch[0]=='yes')
          {goto view;
          }
          print();/*inserstion(static):- put(&sir,12); put(&sir,15);put(&sir,30);*/
          printf("list: ");
          seesaw(sir);}
struct copy  // structure initialized
{int ans;
          struct copy *rt;};
typedef struct copy cy;
void   seesaw(cy *head)
{while(head!='\0')
            {printf("%d ",head->ans);
                    head=head->rt;}
}
int put(cy **kn,int all)
{cy *kaumudi;
          kaumudi=(cy*)malloc(sizeof(cy));
          kaumudi->ans=all;
          kaumudi->rt='\0';
          if(*kn=='\0')
          {*kn=kaumudi;
          }
          else{cy *kaju;
                    kaju=*kn;//hello
                    while(kaju->rt!='\0')
                    {kaju=kaju->rt;//hi
                    }kaju->rt=kaumudi;
          }return 0;
}
void print ()
{printf("all done");
}
```

B1

Fig 3: Level 1 Example

2. RELATED WORK

A lot of research work has been made on plagiarism detection techniques and many tools have been invented. Initially the focus was on counting based approaches. In these approaches, the operators, operands, identifiers and keywords are extracted by applying certain algorithms. Halstead‟s metric was a counting based approach. It calculated the number of unique operators and operands in the code and calculated their proportion from total operators. Initially the results were promising but this method is now obsolete.

Later systems such as those of Donaldson, Lancaster, and Sposato [8], Grier [11], Berghel and Sallah [7] and Faidhi and Robinson [2] introduced a much larger number of metrics and notions of similarity for the resulting feature vector in order to improve performance.

The other most popular and acceptable approach is structure based comparison rather than just of summary indicators. In this approach some common traits of the code were analyzed and code is converted into tokens. These tokens were used for comparison. This type of token formation reduces the dependency on a particular language. Some tools based on this approach are discussed below:

MOSS is based on this approach. Moss is defined as measure of software similarity [14]. It is a plagiarism detection method produced by Alex Aiken and UC Berkley. A technique called winnowing is used to locate matching sequence. File is divided into k-grams which are connecting substring of length k. It supports a wide range of languages such as C, C++,java

etc. Another tool is JPLAG[13] which can be used to check plagiarism of source code written in C, C++ and Java. Firstly, source code in the directory is parsed and then transformed into token strings. After this these tokens are compared by running karp-rabin greedy string tiling algorithm. The comparison results were shown in HTML file which can be visited by using any browser.

Dick Grune [12] developed a tool known as SIM which was based on structure comparison. Firstly it tokenized the source program and then created a forward reference table that could be used for detecting the best matches between new files and the text after completing comparison in both the files. Plaggie [15] is a stand-alone source code plagiarism detection tool developed by Aleksi Ahtiainen and Mikko Rahikainen. This tool detects plagiarism in Java programming exercises. It specialized for typical outcomes of programming exercises: small files based on a template. It firstly parses the programs into tokens and these tokens are compared by greedy string tiling algorithm. It is used only for java language.

A algorithm based on assembly language was proposed by Shuqian Shan, Fengjuan Guo, and Jiaxun Ren[16]. They converted the source code into assembly language and then used the string comparison algorithm to detect the similarity. The algorithm was an improved version of karp-rabin algorithm. This type of conversion into assembly language optimizes the code and hence increases the efficiency of detection system. This algorithm compares the text string with the target string and produces percentage of similarity between two sections in the result.

To provide the similarity visualization between C program source codes, Akhil Gupta and Dr. Sukhvir Singh [17] used the lexical analysis technique. The plagiarism categories which can be detected as modification of the order of statement, comment modification, data type modification, copying the whole program or replacing control structure with similar control structure.

The authors Haritha N., M.bhavani and K.Thammi [19] developed a system that detects plagiarism in C language. This system checks both the folders and files containing source code. The system is divided into three phases: firstly tokens are formed, secondly finger prints are created using N-grams technique and lastly comparison is made by using the Jaccard‟s similarity coefficient. This is used to check all the programs exist in the folder. This system gives a pictorial representation of the result. It compares a single program with the set of programs.

A clustering based approach, P-detect is used for detecting plagiarism in source datasets, developed by Lefteris Moussiades and Athena Vakali [18].This P-detect firstly take the set of programs as input and represents the programs as a set of keywords. A similarity measure evaluation by Jaccard‟s similarity coefficient is performed for each pair of programs. We get a pair-wise similarity measure for each pair of program and store in the form of text file. This file is then passed through any clustering algorithm along with a minimum cut-off value of similarity. The pair which shows higher degree of similarity than minimum cut-off value comes as result in form of weighted non directed graph. Plagiarized pairs are analyzed by this graph in which vertices represent program and edges represent similarities between programs.

3. ASSEMBLY LANGUAGE

Assembly language is a low level language which can be converted by the GCC compiler. GCC compiler is a collection of GNU compiler and it is insensitive to all types of different code transformation. For example-swapping variable names, reformation of language, adding extra comment or blanks. In comparison to C programming language, it has less complex structure and many intermediate language usually get mapped by one command of high level language.

Therefore it becomes an easier task for the method to sort out similarity among the token string of the various C program. That‟s why in this paper, we have projected a method which does not detect plagiarism on the original code, but on the assembly language code that is transformed and generated by the GCC compiler. Table1 shows some samples of common assembly language instruction are shown, which are used in assembly language to perform specific task.

Table 1.Few Sample Assembly Language Instructions

Assembly Language instructions	Description
ADD	Add two values, returning a new value
SUB	Subtract two values, returning a new value
CALL	In order to call a function
JUMP	Jump from one statement to another statement
MOVE	To move value to registers

Maximum students/ persons those are involved in plagiarism, modify the layout of the sentence structure to generate the outcome spot on. Those who have understanding of program practice semantic mean and make changes in the copied code such as: alter the expression with the similar structure, add large quantity of statement in short programming and show results in very minor resemblance etc.

Zhao, C. and Yan, H. introduced a method in which program get converted into assembly language by the help of disassemble and compiler optimization [11]. The similarity results are produced in the form of a threshold, which are further clustered to represent the result. Semantic mean can be detected by this approach. The idea of converting source code into assembly language and comparing the assembly language code, not using method of clustering, has been drawn by this paper with few enhancements in the approach.

The purpose of transforming source code into assemble language and comparing the assembly language is that after assembling, assembly language will filter out all type of blanks and comments primarily. Further, the transformed assembly language transform into token string which is basically the enhanced method. Further, we use our algorithm to find the string likeness to get better similarity between the obtained source codes.

4. PROJECTED PLAGIARISM DETECTION SYSTEM

Seriousness of plagiarism has urged to find better ways to detect and avoid it. On discussing plagiarism detection technique, a lot of work can be found on this topic by many researchers and many theories have been proposed to detect plagiarism. To improve the shortcomings of previous techniques, a new technique to detect plagiarism is proposed. This proposed technique resolves the problems which were raised in earlier techniques.

In plagiarism, instead of copying the whole file, small section of codes from different code files are copied. This type of activity is not easily and perfectly detected from available approaches. The proposed approach compares the source code file with the n files and check which part is copied from which file. The results obtained are quite good as presented in result section. The work flow of the technique is depicted in figure 4 and partitioned as mentioned below:

4.1 Conversion into assembly language

Using GCC compiler, system converts all the available code files into assembly language code files and out of all mark one file as a source file in which act of plagiarism is to be checked. While conversion into assembly language, system removes all the comments and extra stuff from the code and just source code portion is remained in the assembly language codes. These removals are really helpful as it increases the accuracy of the technique. Even, as number of statements has been reduced so space and time complexity as well reduced as compared to not reduced statement code.

4.2 Conversion into blocks

In this module all the assembly language code files are divided into smaller blocks. This type of block division helps to detect plagiarism even if only a smaller part of code is copied. Source file is taken as "a" and its block divisions are taken as"a1, a2, a3" and so on. The other files are taken in similar manner and a name is given to those blocks.

4.3 Tokens formation

After the block division tokens are generated from the code present in each block of files. A dictionary containing all those tokens is created. This dictionary helps to calculate the frequency of all tokens. The tokens reside inside the blocks and every block has its own tokens.

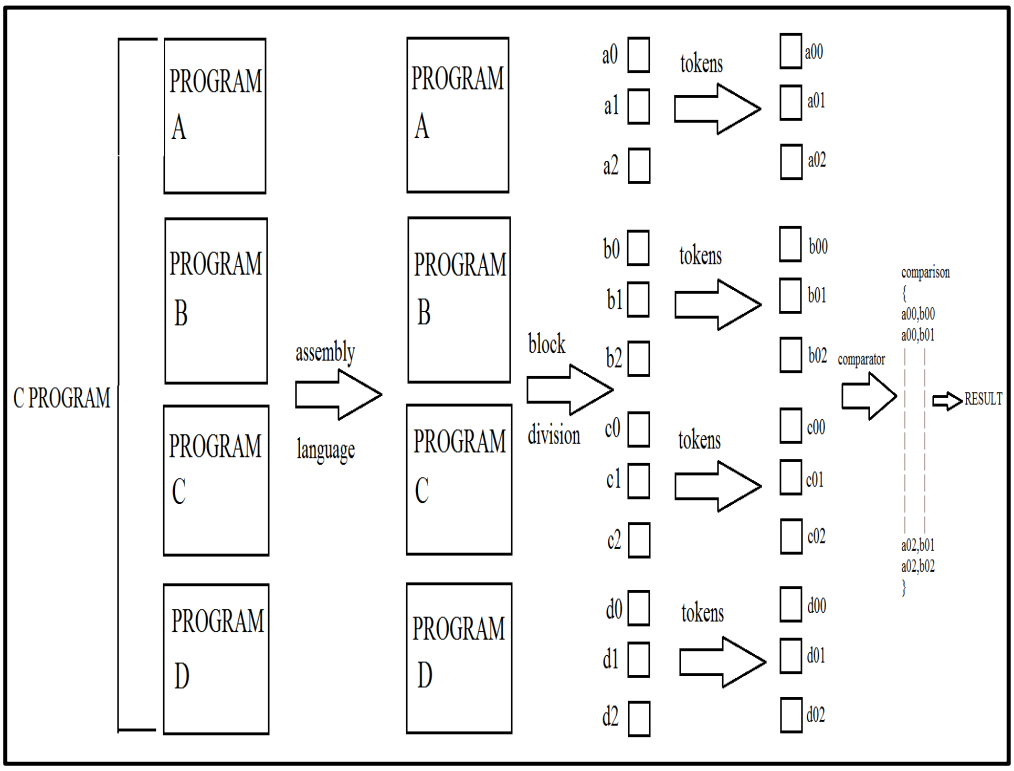

Fig 4: Work flow of projected Plagiarism Detection system

4.4 Comparison of Tokens

Blocks of source code are created one by one and are compared with the blocks of other code files. For example- take block a0 of source code file and select one token from this block. One block(b1) from other file(b) is selected and compared with source code block token(a1) with the tokens of this block. If a match is found for a token of source block(a0) with the key of block(b1) then the value of that key is compared key of source block token. Only the minimum value is considered and this value is added into list. Different lists are obtained for comparison between different blocks. Percentage match is calculated by using list values of each block.

This procedure is repeated for every block of source code with the all blocks of other files. This percentage is used as matching percentage between blocks. A threshold matching percentage for two blocks is also defined. If the percentage is greater than this threshold percentage, then blocks are declared to be copied. The output is taken in excel file and the statistical output is shown in the form of a graph.

5. RESULTS

In our analysis, 100 random programs have been compared with the source program. After comparing the blocks of these programs, the percentage of comparison between blocks is shown below. As we know that every program cannot be unique so we set a Threshold value (minimum percentage level) of comparison .The minimum percentage level is 80% because it shows us good results. The output produce after comparison is satisfactory.

Table 2 shows the plagiarism comparison results of the nive file blocks. All 9 blocks have been compared with all other blocks and the block average results have been displayed in the table 2.

	Text 10	Text 11	Text 12	Text 13	Text 14	Text 15	Text 16	Text 17	Text 18
Text 10		64.66	59.43	59.89	40.91	47.67	40.94	69.24	23.05
Text 11	64.66		64.53	61.48	50.89	60.51	50.10	75.22	26.43
Text 12	59.43	64.53		90.13	45.43	42.47	41.97	66.16	29.61
Text 13	59.89	61.48	90.13		44.92	40.66	39.36	65.25	27.99
Text 14	40.91	50.89	45.43	44.92		57.38	45.15	44.94	29.52
Text 15	47.67	60.51	42.47	40.66	57.38		64.59	45.3	29.62
Text 16	40.64	50.10	41.97	39.96	45.15	64.59		46.19	28.16
Text 17	69.24	75.22	66.16	65.25	44.94	45.3	46.19		26.06
Text 18	23.43	26.61	29.61	27.99	29.52	29.62	28.16	26.06	

Table 2. Plagiarism Comparison Result of Few sample Programs

Figure 5 shows plagiarism result of code exist in figure 2 and figure 3 which shows plagiarism detection result in form of bar chart of code presents in figures and assembly language based detection shows approximately 94 percentage C code matching exists in text file 10 and text file 19.

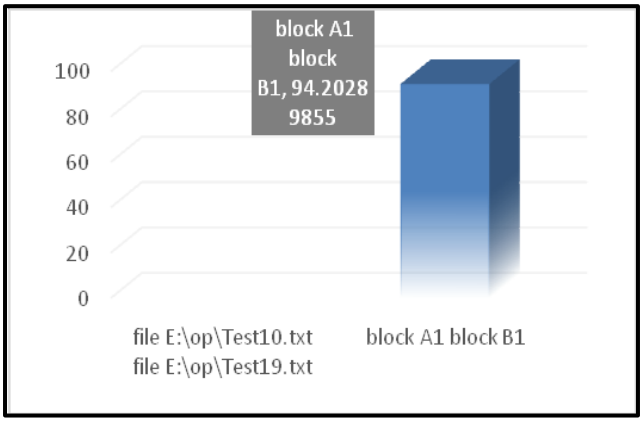

Fig 5: Plagiarism Results

6. CONCLUSION AND FUTURE WORK

The prime objective of this research work was to present an effective approach to detect plagiarism of C code. Therefore, this paper puts forwards a technique for identifying similarity of the source code written in C language. To perform this, we applied a technique in which we match code similarity corresponding to transformed low level assembly language code, further blocks have been formed of token string for converted assembly code of targeted C programs. This paper refers to algorithms, function and phases that helps to transform and equate the source code. According to results, various benefits are there while using low level assembly language. Those benefits are- techniques are insensible to common code transformation and need no other source code processor. Proposed method can be modified by improving preprocessing phases and similarity algorithm, which should improve the results on above talk about transformation, and deliver better performance.

7. REFERENCES

[1] http://www.historians.org/about-aha-andmembership/ governance/policies-and-documents/statement onplagiarism.

[2] J.A.W Faidhi and S. K. Robinson, ”An empirical approach fordetecting program similarity and plagiarism within a university programming environment,” Comput. Educ.. vol. 11. pp. 11-19, 1987.

[3] Ottenstein, K.J.: An Algorithmic Approach to the Detection and Prevention of Plagiarism.CSD-TR200 103(2), 32–39 (1976).

[4] Schleimer, S., Wilkerson, D., Aiken, A.:Winnowing:LocalAlgorithmsforDocumentFingerprinting. In: ACM SIGMOD 2003, pp. 204–212. ACM Press, SanDiego (2003)

[5] Wise, M.J.: YAP3: improved detection of similarities in computer program and other texts.In: Proceedings of the Twenty-Seventh SIGCSE Technical Symposium on ComputerScience Education, vol. 28(1), pp. 130–134. Association for Computing Machinery, NewYork (1996)

[6] G. Whale, “Plague : plagiarism detection using program structure,” Dept. of Computer Science Technical Report 8805, University of NSW,Kensington, Australia, 2008

[7] H. L. Berghel and D. L. Sallach. Measurements of program similarity in identical task environments. ACM SIGPLAN Notices, 19(8):65–76, August 1984.

[8] John L. Donaldson, Ann-Marie Lancaster, and Paul H. Sposato. A plagiarism detection system. ACM SIGSCE Bulletin (Proc. of 12th SIGSCE Technical Symp.), 13(1):21–25, February 1981

[9] M. J. Wise, “Detection of Similarities in Student Programs: YAP'ing may be Preferable to Plague'ing,” ACM SIGSCE Bulletin (proc. Of 23rd SIGCSE Technical Symp.), 2002.

[10] P. Lutz, M. Guido, and M. Phlippsen, “JPlag: Finding plagiarisms among a set of programs,”Fakultätfür Informatik Technical Report 2000-1, Universität Kalrsruhe, Karlsruhe, Germany, 2000. International Journal of Computer Theory and Engineering Vol. 4, No. 2, April 2012

[11] Sam Grier. A tool that detects plagiarism in Pascal programs. ACM SIGSCE Bulletin (Proc. of 12th SIGSCE Technical Symp.), 13(1):15– 20, February 1981.

[12] DickGrune website regarding to similarity measure URL:http://www.dickgrune.com/Programs/similarity_tester/

[13] Jplag tool site URL: http://jplag.ipd.kit.edu

[14] Divya Luke, Divya P.S, Sony L Johnson, Sreeprabha S, Elizabeth.B.Varghese, 2014, “Software Plagiarism Detection Techniques: A Comparative Study”, International Journal of Computer Science and Information Technologies, Vol. 5 (4), ISSN: 0975-9646

[15] Enrique Flores, Alberto Barr´on-Cede˜no, Paolo Rosso, Lidia Moreno , Jun 2012, “DeSoCoRe: Detecting Source Code Re-Use across Programming Languages” , NAACL-HLT 2012

[16] Shan S.,Guo F.,Ren J.:similarity detection method based on assmebly language and string matching

[17] Gupta A., Singh S.: lexical analysis for the measurement of conceptual duplicity between c programs , in proceedings of vol. 1 issue, AUGUST 2013.

[18] Moussiades L., and Vakali A.,:a clustering approach for detecting plagiarism in source code datasets,in proceeding of the computer journal(november 2005) 48(b):6551-661:10.1092/comjnl/bxh119 first published online/:june 24,2005.

[19] Haritha, N., Bhavani, M., & Thammi Reddy, K. (2011). C Code Plagiarism Detection System. International Journal of Science and Advanced Technology, 1(5), 198-203.