

## SANDIA REPORT

SAND2004-5372 Unlimited Release Printed January 2005

# **Evolutionary Complexity for Protection** of Critical Assets

Michael E. Chandross and Corbett C. Battaile

Prepared by Sandia National Laboratories Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

U.S. Department of Energy Office of Scientific and Technical Information P.O. Box 62 Oak Ridge, TN 37831

Telephone:(865)576-8401Facsimile:(865)576-5728E-Mail:reports@adonis.osti.govOnline ordering:http://www.osti.gov/bridge

Available to the public from U.S. Department of Commerce National Technical Information Service 5285 Port Royal Rd Springfield, VA 22161

Telephone:(800)553-6847Facsimile:(703)605-6900E-Mail:orders@ntis.fedworld.govOnline order:http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online



#### SAND 2004-5372 Unlimited Release Printed January 2005

#### **Evolutionary Complexity for Protection of Critical Assets**

Michael E. Chandross and Corbett C. Battaile Computational Materials and Molecular Sciences

> Sandia National Laboratories P. O. Box 5800 Albuquerque, NM 87185-1411 USA

#### Abstract

This report summarizes the work performed as part of a one-year LDRD project, "Evolutionary Complexity for Protection of Critical Assets." A brief introduction is given to the topics of genetic algorithms and genetic programming, followed by a discussion of relevant results obtained during the project's research, and finally the conclusions drawn from those results. The focus is on using genetic programming to evolve solutions for relatively simple algebraic equations as a prototype application for evolving complexity in computer codes. The results were obtained using the lil-gp genetic program, a C code for evolving solutions to user-defined problems and functions. These results suggest that genetic programs are not well-suited to evolving complexity for critical asset protection because they cannot efficiently evolve solutions to complex problems, and introduce unacceptable performance penalties into solutions for simple ones.

This Page Involuntarily Left Blank

### Contents

Introduction	6
Genetic Algorithms	6
Genetic Programming	7
Obfuscation	
Results	11
Simple Functions	11
Polynomial Functions	12
Conclusions	22
References	22

## List of Figures and Tables

#### Introduction

The natural evolution of organisms has created remarkable systems that slowly change in response to sometimes harsh and unforgiving environments. This process shows that starting from the simplest one-celled bacteria, amazingly complex, well-adapted creatures that function on a high level can evolve in response to external stimuli. It is natural, therefore, to conclude that a similar paradigm might potentially be an extremely useful problem solving technique. Evolutionary methods for solving complex problems were introduced in 1975 by John Holland in his book, *Adaptation in Natural and Artifical Systems*. The field of Evolutionary Computing (EC) is generally divided into two major subfields, genetic algorithms (GA) and genetic programming (GP), which both use the concept of evolutionary methods although for different problems. We describe both GA and GP in detail, below.

#### Genetic Algorithms

With EC methods, in general, one starts with a population of trial solutions to a problem. As an example, we will demonstrate the use of a GA to find the minimum value of the function

$$f(x) = (x - 192)^2,$$
(1)

with an initial population consisting of a collection of random integers. Each organism in the population (*i.e.* each integer) can be assigned a fitness that describes how well it solves the problem. An example population with only four individuals is given in Table 1, below. The organisms in the population here are shown by their genome (*i.e.* their representation in binary notation) in order to simplify later discussions.

In this trivial example, the fitness for each organism n is the value F(n). The population is ranked according to fitness (in this case, lower values being better), and then a certain number of the fittest individuals are replicated in the next generation. In this example, we keep organisms 2 and 3 for transmission to the next generation. This is referred to as reproduction, and is clearly asexual. The more fit individuals in the population also undergo sexual reproduction, which is referred to as crossover. Crossover begins with the selection of a subset of the more fit individuals, which are then randomly paired so that information can be exchanged between the genomes. In our example problem, the genome has been chosen to be the binary representation of the numbers, as shown in Table 1. A random point in the genome is chosen as the crossover point, and the bits to the right of this point are swapped between the parents to give two children, as shown in Fig. 1. After this operation, there are four members of the new population as shown in Table 2.

Paren	nts	Childr	en
10111	010	10111	111
11000	111	11000	010
crossove	r point		

Figure 1. Example of crossover between two genomes.

Organism Number	Genome	Value	Fitness
1	01011100	92	10000
2	10111010	186	36
3	11000111	199	49
4	00011011	27	27225

Table 1. First generation of solutions to  $f(x) = (x - 192)^2$ .

Table 2. Second generation of solutions to  $f(x) = (x - 192)^2$ .

Organism Number	Genome	Value	Fitness
1	10111010	186	36
2	11000111	199	49
3	10111111	191	1
4	11000010	194	4

As is clear from Table 2, the overall fitness of the population has increased greatly. There are now two organisms that are very close to the correct answer of 192. The procedure of reproduction and crossover is continued until either a given number of generations has been reached, or an exact solution is found. There is one final mechanism available for modification of the genome that is distinct from the two forms of reproduction demonstrated above. This operation is the mutation operator, and for this example would consist of randomly flipping a bit in the genome (*i.e.* 0 becomes 1 or vice versa). As in biological evolution, the probability of a beneficial random mutation is small, and thus the rate of mutation in the algorithm must be kept correspondingly low.

#### Genetic Programming

The procedure for GP is essentially the same as that for GA in that a given population is evaluated for fitness, and the more fit individuals are chosen to propagate to the next generation through both reproduction and crossover. The essential difference is that GA seek potential solutions to a given problem (e.g. numbers, blackjack strategy tables, or electronic circuits), whereas GP evolves self-contained computer code whose fitness is determined by its output. There are a number of different methods that can generate, evaluate, and evolve a population of programs, but we will only describe two here. The first method is exemplified by the freely available Avida platform (http://dllab.caltech.edu/avida/). In this code, the genome of the organisms consists of programs in Avida's own stripped-down assembly language which runs on a virtual machine. Each organism contains code that allows it to replicate, and hence to reproduce. The programs compete for CPU time and resources that are allocated based on fitness. We determined that the Avida platform was not appropriate for this project. We instead used the common alternate paradigm developed by Koza [1], in which organisms are represented by snippets of Lisp code. To generate and execute codes we use the package lil-gp [2], which strictly adheres to Koza's methods. Before describing the method of GP with Lisp, however, it is first useful to explain the basics of Lisp itself.

Lisp is a simple language consisting of a small basic instruction set from which more complex instructions can be made. Operations are constructed in the form

(op A B), (2)

where *op* is an operator, also known as a terminal; and *A* and *B* are the arguments. For example, the expression (+35) would evaluate to 3+5, *i.e.* 8. Both *A* and *B* can be expressions of their own, so that the more complex expression (+3(+32)) would also evaluate to 8. The nested operator (+32) is evaluated first, with the result passed up to the enclosing operator.

The structure of Lisp codes makes them ideal to represent as parse trees. The example above can be written as the tree shown in Fig. 2. More complex functions including variables can also be constructed in Lisp. For example, the function  $f(x) = x^3 + x^2 + x$  can be represented by the Lisp expresion (+ (+ (\* X (\* X X)) (\* X X)) X), or more conveniently as the parse tree shown in Fig. 3a. In Fig. 3b we show an alternate method of representing this parse tree that we will use below for longer, more complicated functions that would be excessively large if written as shown in Fig. 3a. It is precisely the method of representing Lisp programs shown in Fig. 3a that makes it ideal for GP. Each individual organism can be represented as a parse tree, and crossover can be achieved by selecting a random node in two organisms, and swapping the subtrees at that node. As an example, if we were to perform crossover between the organisms in Fig.s 2 and 3, with the nodes selected being the ones marked in red, the resulting organisms are those shown in Fig. 4.

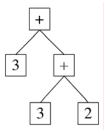


Figure 2. Example of a Lisp parse tree.

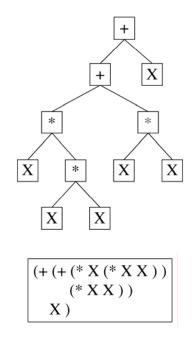


Figure 3. Example representations of Lisp parse trees for  $x^3 + x^2 + x$ .

In GP, as in GA, it has been found that when solutions are generated, they are often exceedingly complex and difficult to understand [3]. As an example, consider the function  $f(x) = x^3$ , given by the simple Lisp expression (\* (\* X X ) X ). A GP used to evolve this function, however, arrives at the equally correct expression (\* (+ (+ (/ X X) (\* X X)) (- (\* X X) (/ X X)) (- (/ (\* X X) (+ X X)) (+ (- X X) (- X X)))). Clearly the evolved expression is more complicated to understand than the original one. It is, of course, possible to develop complicated expressions for  $f(x) = x^3$  by hand, but it is unlikely that human-developed expressions will be as perverse as those derived by a program that evolves solutions. Just as in nature, where organisms evolve into complex and mysterious systems, the results of GP can be obfuscated through indirect and redundant methods resulting from the lack of human intervention. Here we will discuss attempts to exploit this aspect of GP in order to develop intentionally obfuscated code with the goal of protecting of critical intellectual property from reverse-engineering attempts.

#### **Obfuscation**

The goal of code obfuscation is to transform working source into code that is functionally identical, yet much more complex syntactically. Such a transformation is desirable for preventing reverse-engineering of concepts or algorithms that are important intellectual property or crucial for national security. The difficulty with code obfuscation is that, while in some cases it can be easy to identify code that is intentionally obfuscated as compared to code that is not, there is no clear way to quantify the obfuscation because, unlike in cryptography, there has not yet been a theory developed that allows such a measure [4]. Obfuscation differs from cryptography, however, in that once a cryptographic cipher is broken the code is no longer protected. With obfuscated code, the deobfuscation of one section of code, hopefully a time-consuming process, is of little to no use in attempts to deobfuscate other sections. In this sense, obfuscation is a complementary technique to cryptography. In general, actively preventing reverse engineering is a difficult prospect. The International Obfuscated C Code contest is a prime example of the lengths some will go to in order to hamper reverse engineering. As an example, consider the following code, one of the winners in 1998:

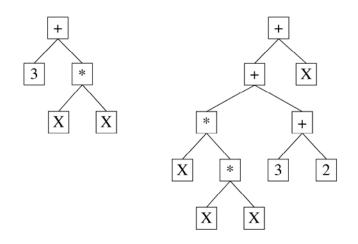


Figure 4. Example Lisp parse trees after crossover.

It is unlikely that anyone can decipher this code, which prints all of The Twelve Days of Christmas, without enormous effort. Similarly, however, writing this code was not a simple process. The emerging view in computer science is that automatic code obfuscation through the use of transformations similar to compiler optimizations is the most appropriate path for the prevention of reverse engineering. On the other hand, some experts feel that this type of obfuscation is impossible because a corresponding deobfuscator can always be devised [5]. This is partially the motivation for evolved complexity – human engineers and programmers, no matter how talented they may be, generally work within constrained mathematical models of idealized systems, and attack problems with particular, well-defined methods. This is why it is possible to create deobfuscators for human-developed obfuscation techniques. The enormous complexity of biological systems, and the correspondingly copious funding currently being allocated to researchers attempting to reverse engineer their functionality, demonstrates that evolution is the ideal technique for developing complex solutions that would never occur to human engineers. As Jostein Gaarder wrote, "If the human brain were simple enough for us to understand, we would still [sic.] be so stupid that we couldn't understand it." [6]

While it is indeed true that no theory of obfuscation has yet been developed, there have been a number of researchers who have studied various techniques used in code obfuscation and attempted to classify them. Such a classification is beyond the scope of this work, and the reader is referred to the excellent review by Campbell [7] for a more complete introduction to obfuscation and the attempts to quantify it.

The GP method will naturally lead to code that is large and difficult to understand even for extremely simple functions. For most practitioners this is an undesirable side effect referred to as "code-bloat," and efforts have been made to try to understand its cause in order to prevention it. This is clearly antithetical to our purposes here, but understanding the cause of code-bloat can also potentially lead to methods for encouraging rather than discouraging it. The major effort in the GP community has been on the relationship of code-bloat and introns (*i.e.* sections of nonfunctioning code such as are found in DNA), although there is no consensus on which is the cause and which is the effect [8,9]. Simple examples of introns that occur in GP include multiplying or dividing large expressions by one [often in the form (/AA)], adding or subtracting zero [often in the form (-AA) or a large, complex expression that is multiplied by this], and combinations of these. Introns are evolutionarily useful for the organism itself since they provide protection from crossover. The more introns that exist in a parse tree, the more likely it is that the subtree selected for crossover is useless to the overall function of that organism, and thus the more likely it is that the code will perform identically before and after crossover.

Introns are one of three types of obfuscation that occur naturally in code produced by GP. These types can be further classified according to the taxonomy of obfuscation due to Collberg *et al.* [4], but this detail is not necessary for our purposes here. The second trivial form of obfuscation arises from the overall allowed tree depth. Often the tree depth is constrained (generally to around 17 [1]) in order to prevent code bloat. Clearly the tree depth and introns are related in the production of code bloat. We have performed some experiments of induced obfuscation through tree depth manipulation which will be described below.

The third form of obfuscation from GP is algorithmic obfuscation. Algorithmic obfuscation is essentially using a complicated algorithm where a simpler one would do - in some sense it is the opposite of simplification of an equation. This form is arguably the most important for true code obfuscation, as introns can be easy to spot and ignore. This can greatly speed up understanding of a parse tree, particularly when large sub-trees (as occur with increased tree depth) can be ignored.

#### Results

#### Simple Functions

We begin by presenting results of GP runs to produce simple functions. The goal here is to study the method itself and to understand the types of obfuscation produced. To this end, we will show examples of the three types of obfuscation described above as produced by actual GP runs on a simple function. For this section we choose the trivial function f(x) = 2x.

We begin with an example of an intron. In Fig. 5 we show a successful run in lil-gp to generate f(x) = 2x. The code shown evaluates to X + [(X - X) + X], or 2X. The appearance of the intron (-X X) does little to obfuscate the code here, and it is clear that it can be ignored upon only cursory examination of the parse tree. It is not surprising that this example is trivial, however, since it was generated as one of the original random trees in the population, and just happened to be correct.

The code shown in Fig. 6 in the alternative format is an excellent example of a more complicated intron that cannot easily be distinguished from important code. This code appeared in generation 10 of the run, and contains 47 nodes with a tree depth of 7. The code is equivalent to the expression

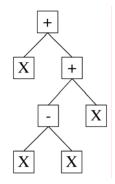


Figure 5. An example of an intron in a GP-evolved solution for f(x) = 2x.

$$(X+X) - \frac{(X-X)^{2} - (X^{2} - X^{2})}{\frac{X-X}{X^{2}} + \frac{X}{X} \left( 4X - \frac{X-X}{\frac{X}{X}} \right)},$$
(3)

where it can easily be understood that the third term is identically zero. It is, however, also more difficult to parse this expression than the trivial one shown in Fig. 5. This code shows how introns and increased tree depth lead to code bloat.

To give an example of algorithmic obfuscation we move to the slightly more complicated function  $f(x) = x^3$ . For this example we show in Fig. 7a and 7b two different results that evolved to find this solution. In Fig. 7a, the code shown evaluates to  $[X^2 * X - (X - X)(X - X)] - [(X + X) - (X + X)] [X^2 * X^2]$ , which simplifies to  $X^3 + 0$ . Although this code looks complicated, it is essentially obfuscation by introns, and is not particularly interesting. The purpose of showing this code, however, is that while it looks nearly identical to the code shown in Fig. 7b, there are distinct differences.

The code shown in Fig. 7b is an example of algorithmic obfuscation. Evaluation of the tree leads to the partially simplified expression  $(X/X + X^2 + X^2 - X/X) [X^2/2X - (X - X) + (X - X)]$ . Clearly there are introns in this expression, but after their removal further simplification steps give  $(X^2 + X^2)(X^2/2X)$ , then  $2X^2 * (X/2)$ , and finally  $X^3$ . The fundamental difference between the expressions in Fig. 7a and 7b is that in Fig. 7a after removal of the introns, one is left with simply  $X^3$ ; whereas in Fig. 7b, removal of the introns leaves one with an expression that must be evaluated to give  $X^3$ . This, then, is what is meant by algorithmic obfuscation.

#### **Polynomial Functions**

Let us now turn our attention to the practical example of the obfuscation of the polynomial,  $f(x) = x^3 + x^2 + x$ . While it is clear from the preceding discussion that introns and algorithmic obfuscation tend to arise naturally during the course of GP optimization, we will further encourage obfuscation by incorporating not only the accuracy of the output but also the size of

Figure 6. An expression for f(x) = 2x that contains numerous introns. Everything after the first line equates to 0.

the evolved tree in the fitness criterion. Specifically, consider a GP in which the available terminals are +, -, \*, and / and the arguments are functions of X. (This is the same sort of GP presented above.) In order to evolve a program that computes  $f(x) = x^3 + x^2 + x$ , it is natural to choose a fitness function like

$$F = \sum_{trials} \left| f(x) - g(x) \right| \tag{4}$$

where the summation is performed over some predetermined set of fitness cases or "trials" (*i.e.* values of *x*), and g(x) is the evolved GP that is attempting to evaluate to f(x). Lower values of *F* represent a higher fitness. While the optimum fitness value, F = 0, might never be reached (in a tractable time) for complex target functions, something as simple as  $f(x) = x^3 + x^2 + x$  is generally achievable in relatively few generations of the GP.

For example, we ran lil-gp with input values of 5120 trees (*i.e.* organisms), initial tree depth between 2 and 8, maximum depth of 32, a 9:1 ratio of crossover to reproduction rates, and 200 randomly-selected trial values for *x* between -1 and 1. This required only one generation to evolve the Lisp tree (+(/(\*XX)(\*(/XX)X))(\*(+X(\*XX))(+X(-XX))))).

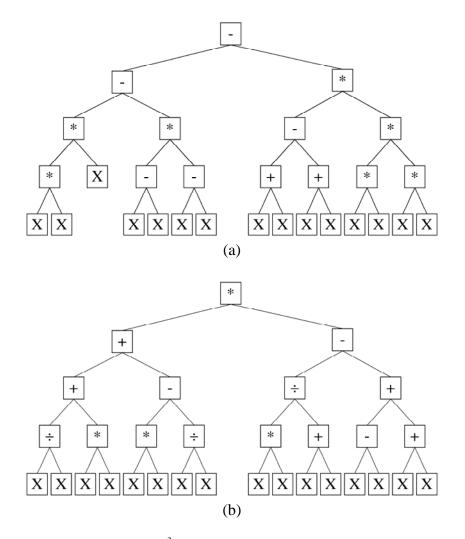


Figure 7. Evolved versions of  $f(x) = x^3$  that show a) no evidence and b) clear evidence of algorithmic obfuscation.

Substituting (-X X) = 0, (/X X) = 1, and  $(*X X) = X^2$  yields  $(+(/X^2 (*1 X)) (*(+X X^2) (+X 0)))$ , which clearly reduces to  $(+X (*(+X X^2) X)) = X + X^2 + X^3$ . Thus, the evolved program contains introns, but is equivalent to the target function  $f(x) = x^3 + x^2 + x$ . For comparison, the fitness function in Eq. 4 was used to evolve solutions to both  $f(x) = x^4 + x^3 + x^2 + x$  and  $f(x) = x^5 + x^4 + x^3 + x^2 + x$ , with 10240 trees, initial tree depth between 2 and 10, and maximum depth of 64. The first accurate solution to the fourth-order polynomial lived in the 33<sup>rd</sup> generation and had the form (\*X (+(/X X) (\*(+(\*(+(\*X X) X) X) X) (/X X)))), which contains two instances of the intron (/X X). (The construct, X+1, is common in these examples for obvious reasons, and the only viable mechanism for generating a 1 is via (/X X).) Surprisingly, for the fifth-order polynomial, the eighth generation contained the individual (-(\* (+(\*(X X) X) X) X) (+X(\*X X))))(-X(+X X))), which is a match to the target function and contains no introns.

In order to see how quickly the GP can arrive at a completely unobfuscated solution, it is useful to modify the fitness function to discourage bloat, such that

$$F = \sum_{\text{trials}} N \left[ \left| f(x) - g(x) \right| + 1 \right], \tag{5}$$

where *N* is the number of nodes in the tree. While this is an overly simplified representation of obfuscation as applied to the fitness function, it is satisfactory for our present purpose. In this case, using lil-gp with the same input parameters as above, the GP arrived after only two generations at the solution (+(\*(+(\*X X) X) X) X)), which is clearly  $X^3 + X^2 + X$ . For comparison, the fitness function in Eq. 5 was used as before to evolve solutions to both  $f(x) = x^4 + x^3 + x^2 + x$  and  $f(x) = x^5 + x^4 + x^3 + x^2 + x$  over 256 generations, with 10240 trees, initial tree depth between 2 and 10, and maximum depth of 64. The best solution to the fourth-order polynomial lived in the sixth generation and had the form (\*(+(/X X) X) (+X(\*(\*X X) X)))), which again contains the intron (/X X). The GP could not evolve a match to the fifth-order polynomial within 256 generations, and ended up with the rather poor solution of X itself with a fitness of F = 97.6. (The strong bias in Eq. 5 against large trees is partly to blame for this.)

The preceding examples demonstrate that the feasibility of evolving polynomial functions decreases rapidly with increasing complexity of the target function. However, for the purposes of the present study, it is useful to consider tractable functions and to examine how the GP evolves solutions for them when obfuscation is rewarded rather than penalized. To do this, we use the fitness function

$$F = \sum_{\text{trials}} \left[ \left| f(x) - g(x) \right| + 1 \right] / N.$$
(6)

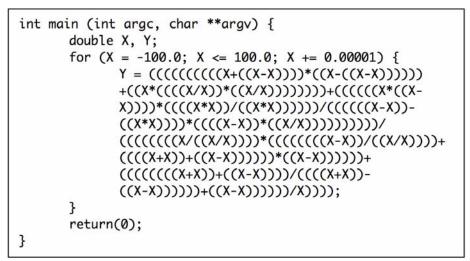
and the same input parameters as above, to evolve  $f(x) = x^3 + x^2 + x$ . The fittest individual lived in generation 10 and is shown in Fig. 8. The tree in Fig. 8 simplifies to exactly  $X^3 + X^2 + X$ , but clearly has numerous introns and algorithmic obfuscations. As discussed above, while this tree can be simplified to reveal its functionality in a relatively short time, a more complex code containing many similarly obfuscated functions would be much harder to decipher.

However, it is apparent that the evaluation of the tree in Fig. 8 might require substantially more computer time than the evaluation of  $X^3 + X^2 + X$  itself. To quantify this, we converted the tree into the C code shown in Fig. 9a, and compared its performance with the code in Fig. 9b. The codes were compiled using the GNU Compiler Suite's gcc 3.3.3 on an 800Mhz Intel Pentium III Xeon processor running Red Hat Fedora Core 2 (kernel 2.6.5 and glibc 2.3.3). The assembler

instructions for two main() functions from Fig. 9 are shown in Fig. 10. Each benchmark represents the average over ten executions of the code. Without any compiler optimizations, the target code (Fig. 9b) evaluated in 0.4 sec and the evolved code (Fig. 9a) in 18.90 sec. When compiled with compiler optimization at level 3 (*via* the –O3 flag), producing the assembler instructions in Fig. 11, each code completed in 0.15 sec. This is because, as is evident from Fig. 11, the compiler's optimizations have simplified the assembler instructions of the obfuscated code (Fig. 9a) so that they are identical to those of the target code (Fig. 9b). In an attempt to reduce the optimizer's ability to rearrange the code, we converted all the arithmetic operations into function calls, as reflected in Fig. 12. While this is in general a bad idea from the perspective of the code's performance, it is a useful exercise for the present purpose. Without compiler optimizations, the compiled codes in Fig. 12b and 12a executed in 4.41 and 54.28 sec, respectively, compared to 2.48 and 35.85 sec with level 3 optimization. In this case, the compiler's optimizer was unable to substantially simplify the code in Fig. 12a, as evidenced by the assembler instructions in Fig. 13.

$$\begin{array}{c} (+X \\ (*X \\ (+(*(/XX)) \\ (+X \\ (*(-XX)) \\ (-XX)))) \\ (+(-(/(*(*(/XX)) \\ (-XX)) \\ (-(*XX)) \\ (-(*XX)) \\ (*(*XX)) \\ (*(+(+XX) \\ (/XX))) \\ (*(*XX)) \\ (*(*XX) \\ (/XX)) \\ (+(+(/(*XX) \\ (/XX)) \\ (+(+(XX) \\ (-XX))) \\ (+(+(XX) \\ (-XX))) \\ (+(+XX) \\ (-XX))) \\ (-(+XX) \\ (+(XX)) \\ (+(XX)) \\ (+(XX)) \\ (-(+XX) \\ (+(XX))) \\ (+(-(XX) \\ (+(XX))) \\ (+(-(XX) \\ (+(XX))))) \\ (-(*XX) \\ (+(XX))))) \\ \end{array}$$

Figure 8. A solution to  $f(x) = x^3 + x^2 + x$ , evolved with a fitness function rewarding large trees.



(a)

int main (int argc, char \*\*argv) {
 double X, Y;
 for (X = -100.0; X <= 100.0; X += 0.00001) {
 Y = (X\*X\*X) + (X\*X) + X;
 }
 return(0);
}</pre>

(b)

Figure 9. C code representations of a) the tree in Fig. 8, and b) the function  $X^3 + X^2 + X$ .

	- 08048330	~ <m< th=""><th>ain&gt;</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></m<>	ain>												
8048330	55		unr.		ish	\$ehn	80483cf.	de ca				fmiln	\$st \$st(1)		
0040330	99 05			-		Som Sohn	90493d1.	do do				fmiln	Set Set(1)		
0040330	02 ~ 10	D		in con	h h	could some	0040301.	do fo				fdiam	Gat Gat(1)		
0040331				Su		SUNTO, SEED	0040303:	de 19				fadde	SL, SSL(1)		
0040342	b0 00 00	0 00	00	di	ICI .	Source Boom	0040305:	de CI				ET AL	ter ter ter	(Rohm)	
8048343	20 -1	00 0	00	nc	JV.	SUXU, teax	804830/:	00 45	10			Edinal	OXIIIIII8	(teop)	
804834a	29 64		00	SU	D	teax, tesp	80483da:	CC /5	18			TOIVI	UXIIIIII8	(tepp)	
804834C	DB 00 00	0 00	00	mo	v	SUXU, teax	8048300:	00 45	18			TIGI	UXIIIII18	(tepp)	
8048351	ba 00 00	0 59	CU	mo	v	\$0xc0590000, *eax	80483e0:	de II	-			taivp	*st, *st(1)		
8048356	89 45 1	3		mo	v	<pre>*eax, 0xfffffff8(*ebp)</pre>	80483e2:	dd 45	18			tial	Oxfffffff8	(tepp)	
8048359	89 55 fc	C		mo	v	<pre>%edx,0xfffffffc(%ebp)</pre>	80483e5:	dc 65	18			fsubl	0xfffffff8	(tebp)	
804835c	dd 45 f	В	1000	fl	ldl	0xfffffff8(%ebp)	80483e8:	dd 45	f8			fldl	0xfffffff8	(%ebp)	
804835f	dd 05 28	8 85	04	08 fl	ldl	0x8048528	80483eb:	dc 75	f8			fdivl	0xfffffff8	(%ebp)	
8048365	da e9			fu	comp	2	80483ee:	de f9				fdivrp	%st,%st(1)		
8048367	df e0			fn	nstsw	tax	80483f0:	dd 45	f8			fldl	0xfffffff8	(%ebp)	
8048369	f6 c4 05	5		te	est	\$0x5, %ah	80483f3:	dc 45	f8			faddl	0xfffffff8	(tebp)	
804836c	74 05			je	2	8048373 <main+0x37></main+0x37>	80483f6:	dd 45	f8			fldl	0xffffff8	(%ebp)	
804836e	e9 da 00	0 00	00	jn	np (	804844d <main+0x111></main+0x111>	80483f9:	dc 65	f8			fsubl	0xfffffff8	(%ebp)	
8048373	dd 45 f8	В		- f1	lai	0xfffffff8(%ebp)	80483fc:	de c1				faddp	%st,%st(1)		
8048376	dc 65 ff	3		fs	subl	0xffffffff8(%ebp)	80483fe:	de cl				faddp	%st, %st(1)		
8048379	dc 45 ff	3		fa	addl	0xfffffff8(%ebp)	8048400:	dd 45	f8			fldl	0xfffffff8	(%ebp)	
804837c	dd 45 f8	В		fl	ldl	0xfffffff8(%ebp)	8048403:	dc 65	f8			fsubl	0xfffffff8	(tebp)	
804837f	cc 65 ff	3		fs	subl	0xffffffff8(%ebp)	8048406:	de c9				fmulp	%st,%st(1)	1.000 A.	
8048382	dd 45 f8	В		fl	ldl	0xfffffff8(%ebp)	8048408:	de c9				fmulp	%st, %st(1)		
8048385	deel			fs	subp	%st,%st(1)	804840a:	dd 45	f8			fldl	0xfffffff8	(%ebp)	
8048387	de c9			fm	allo	%st,%st(1)	804840d:	dc 45	f8			faddl	0xfffffff8	(%ebp)	
8048389	dd 45 f8	В		fl	ldl	0xfffffff8(%ebp)	8048410:	dd 45	f8			fldl	0xfffffff8	(tebp)	
804838c	dc 75 ff	3		fd	livl	0xfffffff8(%ebb)	8048413:	dc 65	f8			fsubl	0xfffffff8	(%ebp)	
804838f	dd 45 f8	в		fl	ldl	0xfffffff8(%ebo)	8048416:	de c1				faddp	%st,%st(1)		
8048392	dc 75 ff	3		fd	livl	0xffffffff8(%ebp)	8048418:	dd 45	f8			fldl	0xfffffff8	(tebp)	
8048395	de c9			fm	nulp	%st,%st(1)	804841b:	dc 45	f8			faddl	0xfffffff8	(tebp)	
8048397	dc 4d ff	3		fm	null	0xffffffff(%ebp)	804841e:	dd 45	f8			fldl	0xfffffff8	(%ebp)	
804839a	de c1			fa	addp	%st,%st(1)	8048421:	dc 65	f8			fsubl	0xfffffff8	(tebp)	
804839c	dd 45 ft	В		fl	ldl	0xffffffff8(%ebp)	8048424:	de e9				fsubrp	%st,%st(1)		
804839f	dc 65 ft	3		fs	subl	0xffffffff8(%ebb)	8048426:	de f9				fdivro	%st, %st(1)		
80483a2	dc 4d ff	3		fm	null	0xfffffff8(%ebp)	8048428:	dd 45	f8			fldl	0xfffffff8	(%ebp)	
80483a5	dd 45 f8	В		fl	ldl	0xfffffff8(%ebb)	804842b:	dc 65	f8			fsubl	0xfffffff8	(tebp)	
80483a8	dc 4d ff	3		fm	null	0xffffffff8(%ebp)	804842e:	de cl				faddp	%st,%st(1)		
80483ab	dd 45 f8	В		fl	ldl	0xfffffff8(%ebp)	8048430:	de cl				faddo	%st, %st(1)		
80483ae	dc 4d f	3		fm	null	Oxffffffff8(%ebp)	8048432:	dc 75	f8			fdivl	0xfffffff8	(%ebp)	
80483b1	de f9			fd	livro	%st,%st(1)	8048435:	de f9				fdivro	%st,%st(1)		
80483b3	de c9			fm	alu	%st, %st(1)	8048437:	dd 5d	f0			fstpl	0xfffffff0	(%ebp)	
80483b5	dd 45 f8	В		fl	ldl	0xfffffffff(%ebp)	804843a:	dd 45	£8			fldl	0xfffffff8	(tebp)	
80483b8	dc 65 ff	3		fs	subl	0xfffffff8(%ebp)	804843d:	dd 05	30 8	85 04	08	fldl	0x8048530		
80483bb	dd 45 f8	В		fl	ldl	0xfffffff8(%ebp)	8048443:	de c1				faddp	%st,%st(1)		
80483be	dc 4d ff	3		fm	null	0xfffffff8(%ebp)	8048445:	dd 5d	f8			fstpl	0xfffffff8	(%ebp)	
80483c1	de e9			fs	subrp	%st,%st(1)	8048448:	e9 0f	ff f	f ff		ami	804835c <	main+0x20>	
80483c3	dd 45 ff	3		fl	ldl	0xfffffff8(%ebp)	804844d:	b8 00	00 0	00 00		mov	\$0x0, teax		
80483c6	dc 65 ff	3		fs	subl	0xfffffff8(%ebp)	8048452:	C9				leave	10.000		
80483c9	dd 45 ff	3		fl	ldl	<pre>%ebp %esp,%ebp \$0xffffff0,%esp \$0xofffff0,%esp \$0xof90000,%edx %eax,%esp \$0xo590000,%edx %eax,0xffffff8(%ebp) 0xffffff8(%ebp) 0xfffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xffffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xfffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xffff8(%ebp) 0xff8(%ebp) 0xff8(%ebp) 0xff8(%ebp) 0xff8(%ebp) 0xff8(%ebp) 0xff8(%ebp) 0xff8(%ebp) 0xff8(%ebp</pre>	8048453:	C3				ret			
80483cc	dc 75 ft	3		fd	livl	0xfffffff8(%ebp)									

(a)

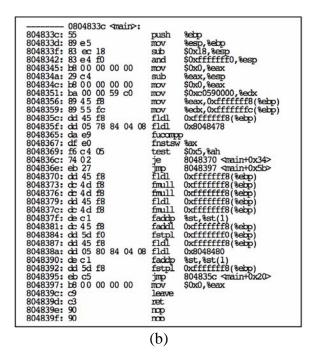
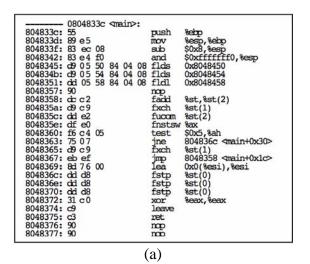


Figure 10. Assembler instructions for the C codes in Fig. 9, compiled without optimization.



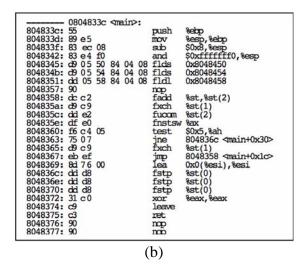


Figure 11. Assembler instructions for the C codes in Fig. 9, compiled with level 3 optimization. Note that the two sets of assembler instructions are identical.

```
double a(double X, double Y);
double s(double X, double Y);
double m(double X, double Y);
double d(double X, double Y);
int main (int argc, char **argv) {
      double X, Y;
      for (X = -100.0; X \le 100.0; X += 0.00001) {
             Y =
             (d((a((m((a(X,(s(X,X)))),(s(X,(s(X,X)))))),(
             m(X,(m((d(X,X)),(d(X,X))))))),(d((m((m(X,
             (s(X,X)))),(d((m(X,X)),(m(X,X)))))),(m((s((
             s(X,X)),(m(X,X)))),(m((s(X,X)),(d(X,X)))))))))))
             ,(d((a((m((d(X,(d(X,X)))),(m((a((d((s(X,X)),(
             d(X,X))),(a((a(X,X)),(s(X,X))))),(s(X,X)))))
             ,(a((d((a((a(X,X)),(s(X,X)))),(s((a(X,X)),(
             s(X,X))))),(s(X,X))))),X))));
      }
        return(0);
}
double a(double X, double Y) { return(X+Y); }
double s(double X, double Y) { return(X-Y); }
double m(double X, double Y) { return(X*Y); }
double d(double X, double Y) { return(X/Y); }
```

(a)

```
double a(double X, double Y);
double s(double X, double Y);
double m(double X, double Y);
double d(double X, double Y);
int main (int argc, char **argv) {
      double X, Y;
      for (X = -100.0; X <= 100.0; X += 0.00001) {
           Y = a(a(m(m(X,X),X),m(X,X)),X);
      }
      return(0);
}
double a(double X, double Y) { return(X+Y); }
double s(double X, double Y) { return(X-Y); }
double m(double X, double Y) { return(X*Y); }
double d(double X, double Y) { return(X/Y); }
```

(b)

Figure 12. C code representations of a) the tree in Fig. 8, and b) the function  $X^3 + X^2 + X$ , using function calls in place of arithmetic operators.

0044030         • 004404030         • 00440300         • 00440300         • 00440300         • 00440300         • 00440000         • 00440000         • 004400000         • 004400000         • 004400000         • 004400000         • 004400000         • 004400000         • 004400000         • 004400000         • 0044000000         • 0044000000         • 0044000000		129075				
$ \begin{array}{c} 0 = 0 \\ 0 = 0 $	0804833c <mai< td=""><td>⊳:</td><td>0</td><td>004040</td><td>Catan 1</td><td>(0)</td></mai<>	⊳:	0	004040	Catan 1	(0)
$ \begin{array}{c} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 $	804833C: 55	push	tepp	804848a: dd 1C 24	ISTPL	(*esp)
Single	8048331: 56	nush	sesp, sepp	8048492: dd 5c 24 08	fstpl	040510 412
B045311         B0         B045317         B045377         B04	8048340: 53	push	%ebx	8048496: 56	push	Ses1
Bottati di D, 5 di B 6 00 05 (Lin bottated of provide	8048341: 83 ec 10	sub	\$0x10, %esp	8048497: 53	push	%ebax
$ \begin{array}{c} 0 = 0 \\ 0 = 0 $	8048344: d9 05 e8 86 0	4 08 flds	0x80486e8	8048498: 56	push	*esi
$ \begin{array}{c} 8044341: 60 = 4 10 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 6 \\ 8044351: 60 = 75 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 60 = 80 \\ 804445: 50 \\ 804445: 60 = 80 \\ 804445: 50 \\ 80445: 50 \\ 8$	804834a: dd 5d f0	fstpl	0xfffffff0(%ebp)	8048499: 53	push	%ebx
BededSis: Bo 5: 0 for more parallel best best best best best best best best	804834d: 83 e4 f0	and	\$0xfffffff0,%esp	804849a: e8 59 01 00 00	call	80485f8 <n></n>
Bolasses Bb 75 PA more post fitting (Meep), Mees Bolasses So pach Mees Bolasses S5 pach Mees Bolasses S5 pach Mees Bolasses S5 pach Mees Bolasses S5 pach Mees Bolasses S5 pach Mees Bolas	8048350: 8b 5d f0	mov	0xfffffff((%ebp),%ebx	804849f: dd 5c 24 08	fstpl	0x8(%esp)
Bolates B 1 b month weight weight between the set of th	8048353: 8b 75 f4	mov	Oxffffffff4(%ebp),%esi	80484a3: 56	push	esi
Biological Construction         Parallel Network         Parallel Network         Parallel Network         Parallel Network           Biological Construction         Parallel Network         Biological Construction         Biological Construction         Biological Construction           Biological Construction         Paralle Network         Biological Construction         Biological Construction         Biological Construction         Biological Construction           Biological Construction         Biological Construction         Biological Construction         Biological Construction         Biological Construction         Biological Construction           Biological Construction         Biological Construction         Biological Construction         Biological Construction         Biological Construction         Biological Construction           Biological Construction         Bio	8048356: 89 16	mov	tesi, tesi	8048484: 53	push	*eox
Diddless         Bit of the set of	8048338: 30	push	Ses1	80484a5: 50 90494a6: 52	push	tesi
064325:         05         00 <t< td=""><td>8048359: 55</td><td>push</td><td>SOVA Soco</td><td>8048420: 55</td><td>call</td><td>8048500 (8)</td></t<>	8048359: 55	push	SOVA Soco	8048420: 55	call	8048500 (8)
000000000000000000000000000000000000	8048354. 56	nuch	soci	80484ac: 83 c4 10	add	S0v10 seen
Biologics         Edit Science         Description         Biologics	804835e: 53	nush	kebx	80484af: dd 1c 24	fstpl	(Sesp)
Bidd306: 53 - G         Bidd307: 54 - G         Bidd307: 5	804835f: 56	push	tesi	80484b2: e8 35 01 00 00	call	80485ec <s></s>
064353: dt 6         65 02 00 00         Call, 00455cc vac         06435cc vac         fatt, 10455cc vac           06435cc vac         Dagi (Memp)         06435cc vac         06435cc vac         Dagi (Memp)           06435cc vac         Dagi (Memp)         06435cc vac         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         06435cc vac         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         06435cc vac         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)         Dagi (Memp)           06435cc vac         Dagi (Memp) </td <td>8048360: 53</td> <td>push</td> <td>%ebx</td> <td>80484b7: 83 c4 10</td> <td>add</td> <td>\$0x10, tesp</td>	8048360: 53	push	%ebx	80484b7: 83 c4 10	add	\$0x10, tesp
Boldstöck dd 5c 24 68 print bask besk besk besk besk besk besk besk be	8048361: e8 8 6 02 00 0	call	80485ec <s></s>	80484ba: dd 1c 24	fstpl	(tesp)
Boldstor, 55 path west. Boldstor, 56 path west. Boldstor, 57 path west. Boldst	8048366: dd 5c 24 08	fstpl	0x8(%esp)	80484bd: e8 36 01 00 00	call	80485f8 <n></n>
Boldstock         Disk         path         Wear         Boldstock         So         path         Wear           Boldstock         53         path         Wear         Boldstock         55         path         Wear           Boldstock         53         path         Wear         Boldstock         55         path         Wear           Boldstock         55         path         Wear         Boldstock         55         path         Wear           Boldstock         55         path         Wear         Boldstock         55         path         Wear           Boldstock         56         0         0         0         Dath         Wear         Boldstock         56         path         Wear           Boldstock         56         0         Dath         Wear         Boldstock         56         path         Wear         Path	804836a: 56	push	tesi	80484c2: dd 5c 24 08	fstpl	0x8(%esp)
Biological DS         Partial Mark         Biological DS         Partial Mark         Biological DS           Biological DS         CA         DO DO DO         Fatpl Dod(Marc) DS         DO DO DO         Call Biological DS           Biological DS         DO DO DO         Fatpl Dod(Marc) DS         DO DO DO         DO DO DO         DO DO DO           Biological DS         DO DO DO         Call DS         Biological DS         DO DO DO         Call DS         Biological DS           Biological DS         DO DO DO         Call DS         DO DO DO         Call DS         Biological DS         DO DO DO         Call DS         DO DO DO         DO DO DO DO         DO DO DO         DO DO DO         DO DO DO         DO DO DO         DO DO DO         DO DO DO DO         DO DO DO DO DO         DO DO DO DO DO DO         DO DO DO DO DO DO DO DO DO DO DO DO DO D	8048360: 53	push	*ebx	8048466: 56	pusn	tesi
Boddade:         add p = 02 00 00         Lol 1         Boddade:         Boddade:         add p = 02         Boddade:	804830C: 30 9049363, E2	push	Ses1	80484C7: 33	push	Beerik Beeri
B043375: dl 15 0 26 08         Call B045656 cp           B043377: dl 15 0 26 08         path Meax         B0448475: dl 5 24 08         path Meax           B043377: dl 15 0 26 00         path Meax         B0448475: dl 5 24 08         path Meax           B043377: dl 15 0 26 00         path Meax         B0448475: dl 5 0         path Meax           B043376: dl 10 0 00         call B04566 cp         B044467: dl 10 24         path Meax           B043376: dl 10 0 00         call B04566 cq         B044467: dl 10 24         path Meax           B043376: dl 10 0 00         call B04566 cq         B044467: dl 10 24         path Meax           B043376: dl 10 0 00         call B04566 cq         B044467: dl 10 24         path Meax           B043386: dl 10 0 00         call B04566 cq         B044467: dl 10 24         path Meax           B043386: dl 10 0 00         call B04566 cq         B044467: dl 10 24         B044467: dl 10 24           B043386: dl 10 0 00         call B04566 cq         B0444667: dl 10 24         B044467: dl 10 24           B043386: dl 10 0 00         call B04566 cq         B044467: dl 10 24         B0444665           B043386: dl 10 0 00         call B04566 cq         B0444665: dl 10 24         B0444655           B043386: dl 10 20         path Meax         B0444665: dl 10 24         B0446665: dl 10 24	9049360: 09 79 02 00 0	push	9049500 (P)	90494c0: 50 90494c9: 53	push	BODY STATE
B043377:         Sic 2 & 0 & Dir         print         Beesl         Dir         B043377:         Sic 2 & 0 & Dir         Farty Dorigi (Reep)           B043377:         Sic 2 & 0 & Dir         print         Beesl         B0448435:         Sic 2 & 0 & Dir         Print         Beesl         B044843:         Sic 2 & 0 & Dir         Print         Beesl         B0448435:         Sic 2 & 0 & Dir         Print         Beesl         B044843:         Sic 2 & 0 & Dir         Print         Beesl         Bo448445:         Sic 2 & 0 & Dir         Dir         Bo44845:         Bo44845:         Sic 2 & 0 & Dir         Dir         Bo44845:         Bi44845:         Bi44845:         Bi44845:         Bi4485:         Bi4485: <td< td=""><td>8048373: dd 5c 24 08</td><td>fstnl</td><td>0x8(%esp)</td><td>80484ca: e8 29 01 00 00</td><td>call</td><td>80485f8 m&gt;</td></td<>	8048373: dd 5c 24 08	fstnl	0x8(%esp)	80484ca: e8 29 01 00 00	call	80485f8 m>
Biological State         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Biological State         Dirath         Vesic           Biological State         Signed Control         Dirath         Vesic         Dion         Dio	8048377: 56	nush	Resi	80484cf: di 5c 24 08	fstpl	0x8(%esp)
Bidd375: Bidd375; Bidd375; Bidd375; Bidd375; Bidd375; Bidd375; Bidd375; Bidd375	8048378: 53	push	*ebx	80484d3: 56	push	resi
B04337b:         B03         Dush         Metx         B043465:         So         push         Metal           B04337b:         B4 10:0         0:01:0:0:0         0:01:0:0:0:0         0:01:0:0:0:0         0:01:0:0:0:0:0         0:01:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:	8048379: 56	push	*esi	80484d4: 53	push	%ebx
B043307: 66 50 02 00 00         Call         B044350: 63         Display         Path         Webx           B043308: 63 61 20         00         Call         B044367: 63         Display         B044467: 63         Display         B044507: 63         Display         Display         B044507: 63         Display	804837a: 53	push	%ebx	80484d5: 56	push	*esi
B048380: IS C4 ID         achd         SDx10, Mesp         B048427: 66 IC 01 00 00         cclub SDx20, Mesp           B048381: di 10 24 00 00         citti di Mesp)         B048421: 66 IC 01 00 00         cdl         SDx10, Mesp           B048381: di 10 24 00 00         citti di Mesp)         B048421: 66 IC 01 00 00         cdl         SDx10, Mesp           B048381: di 10 24 00 00         citti di Mesp)         B048421: 66 IS C4 08         fstpl         Mess           B048381: di 10 24 00 00         pash         Mesi.         B0484821: 61 IS C4 08         pash         Mesi.           B048381: di 50 24 08         pash         Mesi.         B0484821: 65 IS C4 08         pash         Mesi.           B048381: di 50 24 08         pash         Mesi.         B0484826: 63 ID C0 00 00         call.         B048580: 62 ID C4 ID C	804837b: e8 60 02 00 0	call	80485e0 <a></a>	80484d6: 53	push	%ebx
But stati         Cli C A         Int D         Int D         Description         Bit Add C         D         Cat D         Count         Description           B0443561         65 C A 008         Full         B044351         56 C A 008         Full         B044451         57 C A 008         Full         B044451         90         Pash         Meak         B0445516         10 C A 00         Call         B0445516	8048380: 83 c4 10	add	\$0x10, %esp	80484d7: e8 1c 01 00 00	call	80485f8 AP
Bornson: G. 1 M. U. 00 UN         Call. Develope: Color         Bornson: G. 1 M. M. 00 UN         Internal Method         Internal Method           B043351: 56         Bush         Mail Method         B0448401: 56         G. 10 A 00 ON         Gatal         B046850: 46           B043351: 56         Bush         Mail Method         B0448401: 53         Push         Method           B043351: 56         B0484401: 53         Push         Method         B0448401: 50         Push         Method           B0433521: 55         B0484401: 53         Push         Method         B0448401: 50         Push         Method           B0443551: 55         Dath         Method         B0448401: 51         B044841: 59         Push         Method           B0443551: 55         Dath         Method         B0448561: 51         Dath         Method         B0448561: 51         Push         Method           B0443551: 55         G 10 00         Call         B044550: 42         B044550: 45         B044650: 45         B044650: 45         B044650:	8048383: dd 1c 24	fstpl	(tesp)	80484dc: 83 c4 10	add	SUX10, sesp
Dermanner, So A. S. No.         Antype         Dermanner, S. S. A. S. No.         Antype         Dermanner, S. S. A. S. No.         Dermanner, S.	8048386: e8 61 02 00 0	fatal	048580 5	8048401: 00 1C 24	ISTPL	( accop)
Bit Entropy         Teach         Control         Contro         Control         Control	804838f. 56	Istpl	tor (tesp)	8048467: 01 50 24 09	fetnl	0v8(seen)
Bit Start         Find Start	8048390: 53	push	sebx	80484eb: 56	nuch	soci (cesp)
Biological Sign in the set is a set in the	8048391: 56	nush	sesi	80484ec: 53	push	%ebx
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	8048392: 53	push	%ebx	80484ed: 56	push	Sesi.
Bodd Sign: ch         Could (temp)         Bodd Med: 1: eff 20 00 00 00         Could Sign: Ch         Bodd Med: 1: eff 20 00 00         Could Sign: Ch           Bodd Sign: Sh         push         Weix         Bodd Mif: Sign: Ch	8048393: e8 54 02 00 0	) call	80485ec <s></s>	80484ee: 53	push	%ebx
B048390: 53         push         Wesi         B048476: 59         pop         Weex           B048390: 53         push         Weix         B048476: 53         pox         Pox           B048390: 63         B0 C0 00         call         B048500' cor         B048476: 53         push         Weix           B048380: 63         B0 C4 10         add         SOL0, Weep         B048476: 63         push         Weix           B048380: 63 0 C2 00         add         SOL0, Weep         B048460: 63 0 C2 00         call         B048500' cor         add         SOL0, Weep           B048380: 61 1c 24         fstpl         (Weep)         B048501: 63 0 cl 00         call         SOL0, Weep           B048380: 62 40         add         SOL0, Weep         B048511: 61 cl 24         fstpl         (Weep)           B048380: 62 40         fstpl         (Weep)         B048511: 62 0 cl 10         add         SOL0, Weep           B048380: 62 40         fstpl         (Weep)         B048511: 55         push         Weix           B048380: 62 40         fstpl         (Weep)         B048511: 55         push         Weex           B048380: 63         fstpl         (Weep)         B048511: 55         push         Weex	8048398: dd 5c 24 08	fstpl	0x8(%esp)	80484ef: e8 f8 00 00 00	call	80485ec <s></s>
B048394: 53         push         Weak         B048455: 53         push         Weak           B048394: 53         push         Weak         B048364: 53         push         Weak           B048394: 53         push         Weak         B048364: 53         push         Weak           B048394: 53         push         Weak         B048454: 53         push         Weak           B048394: 63         push         Weak         B048500: 53         push         Weak           B048350: 63         push         Keak         B048500: 53         push         Gath         B048500: 63         push         Gath         SOL10, keep           B048350: 64         1c<24	804839c: 56	push	tesi	80484f4: 59	pop	Secx.
00000000000000000000000000000000000	804839d: 53	push	%ebx	80484f5: 58	pop	*eax
0018232: 120       120       100       000       plus:       matrix         0018232: 120       120       100       add       SOLID, Wessp       0018142: 123       plus:       matrix         0018230: 03 04 100       add       SOLID, Wessp       0018250: 03 c4 10       add       SOLID, Wessp       0048500: 03 c4 10       add       SOLID, Wessp         0018230: 03 c4 10       add       SOLID, Wessp       0048500: 03 c4 10       add       SOLID, Wessp       0048500: 03 c4 10       add       SOLID, Wessp         0018310: 03 c4 10       add       SOLID, Wessp       0048500: 03 c4 10       add       SOLID, Wessp       0048510: 03 c4 10       add       SOLID, Wessp         0018310: 03 c4 10       add       SOLID, Wessp       0048510: 03 c4 10       add       SOLID, Wessp       0048510: 03 c4 10       add       SOLID, Wessp         0018310: 03 c4 10       add       SOLID, Wessp       0048510: 03 c4 10       add       SOLID, Wessp       0048510: 03 c4 10       add       SOLID, Wessp         0018320: 05 c4 10       add       SOLID, Wessp       0048510: 05 c4 10       add       SOLID, Wessp         0018320: 05 c4 10       add       SOLID, Wessp       0048510: 05 c4 10       add       SOLID, Wessp         0018300: 05	8048396: 56	push	*esi	8048416: CC 1C 24	ISTPL	(*esp)
8048326: 80 c4 10       60 c1 (keep)       8048475: c2 d1 (keep)       8048475: c2 d1 (keep)       60 c1 (keep)         8048383: d1 (c 24)       fstpl (keep)       8048500: 83 c4 10       add (kot1) keep       8048500: 83 c4 10       add (kot1) keep         8048383: d1 (c 24)       fstpl (keep)       8048500: 83 c4 10       add (kot1) keep       8048500: 83 c4 10       add (kot1) keep         8048380: d2 (c 24)       add (kot1) keep       8048500: 83 c4 10       add (kot1) keep       8048500: 83 c4 10       add (kot1) keep         8048380: d3 (c 24)       fstpl (keep)       8048500: 83 c4 10       add (kot1) keep       8048510: d6 c2 40 8       fstpl (keep)         8048380: d5 (c 24)       fstpl (keep)       8048510: d5 c3 c4 10       add (keep)       8048510: d5 c3 c4 10       add (keep)         8048380: d5 (c 24)       fstpl (keep)       8048510: d5 c3 c4 08       fstpl (keep)       8048510: d5 c3 c4 08       fstpl (keep)         8048301: d5 (c 24)       push keex       8048510: d5 c1 00 00 call 8048604 c4-       8048510: d5 c1 00 00 call 8048604 c4-         8048301: d5 (c 24)       fstpl (keep)       8048510: d6 c1 00 00 call 8048604 c4-       8048521: d4 5 c2 40 08       fstpl (keep)         8048301: d5 (c 24)       fstpl (keep)       8048521: d4 5 c2 40 08       fstpl (keep)       8048521: d4 5 c2 40 08       fstpl (keep)	8048391: 55 8048390: c8 3b 02 00 0	push call	80485-00 <>>	8048419: 30 80484fa: 53	push	1000
8048323: dd 1c 24       fright       10 000       call       804830: dd 1c 24       fright       (Wesp)         8048330: dd 1c 24       fright       604850: dd 1c 24       fright       (Wesp)         8048330: dd 1c 24       fright       604850: dd 1c 24       fright       (Wesp)         8048330: dd 1c 24       fright       604850: dd 1c 24       fright       (Wesp)         8048330: dd 1c 24       fright       604850: dd 1c 24       fright       (Wesp)         8048330: dd 1c 24       fright       604850: dd 1c 24       fright       (Wesp)         8048330: dd 1c 24       fright       804850: dd 1c 24       fright       804850: dd 1c 24       fright         8048330: dd 5c 24 08       fright       804850: dd 5c 24 08       fright       804850: dd 5c 24 08       fright         8048320: 55       pusht       West       8048521: dd 5c 24 08       fright	80483a5+ 83 c4 10	add	\$0x10 %	80484fb: e8 f8 00 00 00	call	80485f8 dtb
804830:       e8 30 C2 00 00       call       604830:       e8 30 C2 00 00       call       604830:       field       field <t< td=""><td>80483a8: dd 1 c 24</td><td>fstpl</td><td>(Sesp)</td><td>8048500: 83 c4 10</td><td>add</td><td>\$0x10.9esp</td></t<>	80483a8: dd 1 c 24	fstpl	(Sesp)	8048500: 83 c4 10	add	\$0x10.9esp
8048300: 83 c4 10       add       50x10, %eep       8048500: 80 c4 10       add       50x10, %eep         8048303: 81 c 24       fstpl.       (%eep)       8048500: 83 c4 10       add       50x10, %eep         8048303: 81 c 24       add       50x10, %eep       8048500: 83 c4 10       add       50x10, %eep         8048300: 82 c4 10       add       50x10, %eep       8048510: 63 c4 50       push       %eesi         8048300: 82 c4 10       add       50x10, %eep       8048511: 64 c2 0       push       %eesi         8048300: 85 c4 10 c 24       08 fstpl.       0x84851: 55       push       %eesi       8048512: 65       push       %eesi         8048300: 55       push       %esi       8048521: 65 c2 00       push       %eesi       8048521: 65 c1 00       push       %eesi         8048302: 65       push       %eesi       8048521: 65 c2 00       push       %eesi       8048522: 65       push       %eesi         8048302: 65 c3       push       %eesi       8048522: 65       push       %eesi       8048522: 65       push       %eesi         8048302: 65 c3       push       %eesi       8048522: 65       push       %eesi       8048526: 75       push       %eesi       8046526: 75       push	80483ab: e8 30 02 00 0	) call	80485e0 <a></a>	8048503; dd 1c 24	fstpl	(Sesp)
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	80483b0: 83 c4 10	add	\$0x10,%esp	8048506: e8 ed 00 00 00	call	80485f8 ATP
80483b6: e6 49 02 00 00       call       80483b6: e6 49 02 00 00       call       80483b6: e6 49 02 00 00       call       80483b6: e6 49 00 00       call       80485b6: e6 49 00 00       call       80485b6: e6 e6 00 00       fstpl       0x85b6: e6	80483b3: dd 1c 24	fstpl	(%esp)	804850b: 83 c4 10	add	\$0x10,%esp
80483bb: 63 c4 10       add       \$0x10, kesp       80483b: 61 c 24       fstpl       0x467be: d5 c2 4 08       fstpl       0x467be: d5 c2 4 08         80483b: c41 c 24       fstpl       0x467be: d5 c2 4 08       fstpl       0x467be: d5 c2 4 08       push       kesi         80483b: c41 c 24       fstpl       0x467be: d5 c2 4 08       fstpl       0x467be: d5 c2 4 08       push       kesi         80483b: c55       push       kesi       80485b: c55       push       kesi       80485b: c56       push       kesi         80483b: c55       push       kesi       80485ci: 55       push       kesi       80485ci: 55       push       kesi         80483ci: c56       push       kesi       80485ci: 55       push       kesi       80485ci: 55       push       kesi         80483ci: c56       push       kesi       80485ci: 55       push       kesi       80485ci: 55       push       kesi         80483ci: c56       push       kesi       80485ci: 58       push       kesi       80485ci: 58       push       kesi         80483ci: c56       push       kesi       80485ci: 58       push       kesi       80485ci: 58       push       kesi         80483ci: c56       push	80483b6: e8 4 9 02 00 0	) call	8048604 <	804850e: dd 1c 24	fstpl	(sesp)
804830:e: df lc 24       fstpl (Wesp)       8048511: 53       push Wesi         804830:: df lc 24 08       fstpl 0x6(Wesp)       8048512: 55       push Wesi         804830:: 53       push Wesi       8048512: 55       push Wesi         804830:: 53       push Wesi       8048512: 55       push Wesi         804830:: 53       push Wesi       8048512: 56       push Wesi         804830:: 53       push Wesi       8048527: 65       cd 15 c 24 08       fstpl 0x6(Wesp)         804830:: 53       push Wesi       8048529: 56       push Wesi       8048529: 53       push Wesi         804830: 53       push Wesi       8048529: 56       push Wesi       8048529: 53       push Wesi         804830: 53       push Wesi       8048520: 68 dx 00 00       call 8046604 cd-         804830: 54       50 def350: 53       push Wesi       8048530: 58       push Wesi         804830: 55       push Wesi       8048530: 58       pop Weax       8048551: 58       pop Weax         804830: 53       push Wesi       8048530: 58       pop Weax       8048551: 58       pop Weax         8048366: 56       push Wesi       8048530: 58       pop Weax       8048551: 58       pop Weax         8048366: 56       push Weai       804850: 5	80483bb: 83 c 4 10	add	\$0x10, tesp	8048511: e8 ee 00 00 00	call	8048604 <⇔
8048361: 68 14 02 00 00Call8048362 CaP8048316: 56pusheesi8048362: 63pushkesi8048316: 53pushkesi8048362: 63pushkesi8048316: 53pushkesi8048362: 63pushkesi8048316: 53pushkesi8048362: 63pushkesi8048316: 53pushkesi8048362: 63pushkesi8048516: 62 10 00 00call804485616: 62 10 00 008048362: 63pushkesi8048516: 62 10 00pushkesi8048362: 63pushkesi8048526: 63pushkesi8048362: 63pushkesi8048526: 63pushkesi8048362: 63pushkesi8048526: 64 40 00 00call8044504: 408048362: 63pushkesi8048526: 64 40 00 00 00call8044504: 408048362: 63pushkesi8048536: 64 bd 00 00 00call8044564: 408048362: 63pushkesi8048536: 66 bd 00 00 00call8044566: 668048362: 53pushkesi804856: 58popkesi8048362: 53pushkesi804856: 58push804856: 588048362: 53pushkesi804856: 58push8048362: 56pushkesi804856: 58push8048362: 56pushkesi804856: 58push8048362: 56pushkesi804856: 56push8048362: 56pushkesi<	80483be: dd 1c 24	fstpl	(tesp)	8048516: dd 5c 24 08	fstpl	0x8(*esp)
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	80483C1: e8 1a 02 00 0	fatal	8048560 <a></a>	804851a: 56 904951b: 52	push	tesi
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	90493co: CL 5C 24 08	TSCDI	(val esp)	904951c+ 56	push	Bogi
804830c:       56       push       testi       80483ci:       80485ci:       61       00       00       5011       8048604       42b         804830c:       641       50       24       08       fstpl       000       5011       8048504       50       24       08       fstpl       000       5011       8048504       42b       50       push       Wesi       9048529:       50       push       Wesi       9048520:       80       40       00	80483ch: 53	rush	keby	804851d: 53	nush	Sohr
804430:c:53push%ebx804850:c:52.408fstpl0x6 (%esp)804430:c:61 9 02 00 00call804850:c:8048528:53push%esi804433:d:cd 5 c:24 08fstpl0x6 (%esp)8048528:53push%esi804433:d:cd 5 c:push%esi8048528:53push%esi8048326:53push%esi8048528:53push%esi8048326:cd 0 c:02 00 00call8048526:cd 10add\$0040504:8048326:cd 5 c:24 08fstpl0x6 (%esp)8048531:dd 1 c:24\$00.008048326:cd 5 c:24 08fstpl0x6 (%esp)8048531:dd 2518:sd 200.008048326:53push%esi8048531:53pop%eax8048326:53push%esi8048531:dd 251:dd 251:dd 251:8048326:53push%esi8048531:dd 251:dd 251:dd 251:8048326:53push%esi8048531:dd 251:dd 251:dd 251:8048326:53push%esi8048541:53mush%esi8048326:54:0000call8048561:dd 251:dd 251:8048326:56push%esi8048541:55push%esi8048326:56push%esi80485561:dd 251:dd 251	80483cc: 56	push	8esi	804851c: c8 c1 00 00 00	call	8048604 <>
804830ce: eff 19 02 00 00       call       804850c: eff 19 02 00 00       call       8048527: 56       push       wesi         8048331: 56       push       wesi       8048528: 53       push       wesi         8048332: 56       push       wesi       8048528: 53       push       wesi         8048336: 53       push       wesi       8048520: 64 40 00 00       call       8048520: 56         8048336: 53       push       wesi       8048530: 83 c4 10       add       Stock       stock         8048336: 56       push       wesi       8048530: 83 c4 10       add       Stock       stock         8048326: 53       push       wesi       8048530: 85       pop       weax         8048326: 56       push       weai       8048530: 85       pop       weax         8048326: 56       push       weai       8048530: 85       push       weai         8048326: 53       push       weai       8048540: 56       push       weai         8048326: 64 81 01       odd       Sock 10, weap       8048541: 65       push       weai         8048326: 64 82 01       odd       Sock 10, weap       8048542: 64       push       weai         8048326: 65       p	80483cd: 53	push	%ebx	8048523: dd 5c 24 08	fstpl	0x8(%esp)
8048323: dd 5c 24 08       fstpl       0x81 %esp)       8048528: 53       push       webx         8048326: 53       push       webx       8048529: 56       push       webx         8048326: 53       push       webx       8048520: 56       push       webx         8048326: 53       push       webx       8048530: 82 (40 00 00 00       call       8048604          8048326: 54       push       webx       8048530: 86 (40 00 00 00       call       8048561        fstpl       (%esp)         8048326: 55       push       webx       8048530: 80 (10 00 00       call       8048561        fstpl       (%esp)         8048326: 56       push       webx       8048530: 40 (10 c2 4       fstpl       (%esp)         8048326: 66 f3 01 00 00       call       8048560        8048541: 55       push       %est         8048326: 66 f3 01 00 00       call       8048560        8048541: 55       push       %est         8048326: 56       push       %est       8048541: 55       push       %est         8048326: 66 f3 01 00 00       call       8048560        8048541: 55       push       %est         8048326: 56       push       %est       8048561: 65       push	80483ce: e8 19 02 00 0	) call	80485ec <s></s>	8048527: 56	push	tesi.
80483d7: 56       push       wesi       804852a: 53       push       wesi         80483d5: 53       push       wesi       804852a: 53       push       wesi         80483d5: 54       push       wesi       804852a: 53       push       wesi         80483d5: 54       push       wesi       8048536: 66 d4 00 00       cald       8044556: 98 d445536: 86 d4 10         80483d5: 54       push       wesi       8048536: 66 d1 10       add       \$0x10, wesp         80483e4: 56       push       wesi       8048536: 56 push       wesi       8048536: 56 push       wesi         80483e6: 56       push       wesi       8048536: 56 push       wesi       8048536: 56 push       wesi         80483e6: 56       push       wesi       8048536: 56 push       wesi       8048536: 56 push       wesi         80483e6: 54       push       wesi       8048546: 56 push       wesp       8048546: 56 push       wesi         8048365: 54       push       wesi       8048546: 56 push       wesi       8048546: 53 push       wesi         8048366: 56       push       wesi       8048546: 56 push       wesi       8048546: 53 push       wesi         8048367: 56       push       wesi	80483d3: dd 5c 24 08	fstpl	0x8(%esp)	8048528: 53	push	*ebx
80483d3: 53pushbetx80482d1: 33pushbetx80483d2: 53pushbetx80482d1: 33pushbetx80483d2: 53pushbetx804852b: ef d4 00 00 00add\$0x10, beesp80483d2: 64 0c 02 00 00call804853c: 53pushbetx804853b: 54pop80483d2: 55pushbetx804853b: 58popbetx80483ef: 53pushbetx804853b: 58popbetx80483ef: 53pushbetx804853b: 53pushbetx80483ef: 53pushbetx804854c: 53pushbetx80483ef: 53pushbetx804854c: 55pushbetx80483ef: 54pushbetx804854c: 55pushbetx80483ef: 53pushbetx804854c: 53pushbetx80483ef: 68 c4 10add\$0x10, beep804854c: 53pushbetx80483f6: d4 5c 24 08fstpl0x8 (beep)804854c: 55pushbetx80483f6: 56pushbesi804854c: 53pushbesi80483f6: 56pushbesi804854c: 53pushbesi80483f6: 56pushbesi804854c: 55pushbesi80483f6: 56pushbesi804856c: 45pushbesi80483f6: 56pushbesi804856c: 56pushbesi80483f6: 56pushbesi8048556c: 53pushbesi8048400: 66 ff 01 00 00c	80483d7: 56	push	%esi	8048529: 56	push	tesi
0040320: 36       push       webx       0040520: 66 d4 00 00 00       call       0040520: 65         8048326: 66 00 00       call       804856: 58       8048530: 66 d4 00 00 00       call       804856: 68         8048326: 56       push       webx       8048530: 58       pop       weax         8048326: 53       push       webx       8048530: 58       pop       weax         8048326: 53       push       webx       8048530: 64 l1 c 24       fstpl       (%esp)         8048326: 53       push       webx       8048530: 64 l1 c 24       fstpl       (%esp)         8048326: 64 10       ocall       804850: 52       push       webx       8048540: 64 l1 c 24       fstpl       (%esp)         8048366: 68 13 01 00 00       call       804850: 52       push       webx       8048541: 53       push       webx         8048367: 64 l2 24       80 c4 10       add       \$0x10, %esp       8048541: 55       push       %est         8048367: 55       push       %est       8048541: 55       push       %est         8048367: 56       push       %est       8048541: 55       push       %est         8048367: 56       push       %est       8048541: 56       push	8048308: 53	push	*ebx	804852a: 53	push	HEDK COA ST
00103241: 35       20       00100       00100       00100       00100       00100       00100       00100       00100       00100       001000       001000       001000       001000       001000       0010000       0010000       0010000       0010000       0010000       0010000       0010000       0010000       0010000       00100000       00100000       00100000       00100000       00100000       00100000000       001000000000000       00100	8048309: 50 90493da+ 53	push	tesi toby	8048520: 68 04 00 00 00	add	8048604 CP
80483e0: dl 5c 24 08       fstpl       0x8(tesp)       8048551: de bd 00 00 00       call       80485f8 qm>         80483e4: 56       push       tesi       8048530: 58       pop       teax         80483e5: 53       push       tesi       8048530: 58       pop       teax         80483e5: 53       push       tesi       8048530: 58       pop       teax         80483e6: 56       push       tesi       8048530: cd 1c 24       fstpl       (%esp)         80483e6: 86 f3 01 00 00       call       8048560 <a>       8048541: 53       push       teax         80483f3: e6 e8 01 00 00       call       8048560 <a>       8048542: e8 b1 00 00 00       call       8048561: 53       push       teax         80483f5: cd 5c 24 08       push       tesi       8048542: 53       push       tesi         80483f5: 53       push       tesi       8048542: 53       push       tesi         80483f6: 54       push&lt;</a></a>	80483db: e8 0 c 02 00 0	) call	80485ec <s></s>	8048533: dd 1c 24	fstpl	(tesp)
B0483e4: 56pugh %esiB0483e5: 53pop%eax80483e5: 53push %esi804853c: 5apop%eax80483e5: 53push %esi804853c: 5apop%eax80483e6: 56push %esi804853c: 5apush %esi80483e6: 86 f 3 00 00 call80485e0 ca>8048541: 53push %esi80483e1: 81 cd 10 add\$0x10, %esp8048541: 53push %esi80483f3: e6 e8 01 00 00call80485e0 ca>8048541: 56push %esi80483f3: c3 c2 4 08fstpl0x6(%esp)8048541: 56push %esi80483f1: 53push %esi8048541: 56push %esi80483f1: 53push %esi8048551: 58pop80483f1: 53push %esi8048551: 69pop8048401: 66 f 01 00 00call8048561: 56push %esi8048402: 53push %esi8048551: 64 8c 00 00 00call 804856c ca>8048401: 56push %ebx8048551: 64 8c 00 00 00call 804856c ca>8048401: 56push %ebx8048551: 63 push %ebx8048401: 56push %ebx8048551: 63 push %ebx8048402: 53push %ebx8048551: 64 8c 00 00 008048402: 56push %ebx8048561: 538048401: 56<	80483e0: dd 5c 24 08	fstpl	0x8(%esp)	8048536: e8 bd 00 00 00	call	80485f8 <₩
80483e5: 53push%ebx804853c: 5apush%ebx80483e6: 56push%ebx804853c: 5apush%ebx80483e6: 56push%ebx8048540: 56push%ebx80483e8: eff f 3 01 00 00call8048540: 56push%ebx80483e8: eff f 3 01 00 00call8048560: 4b8048541: 53push%ebx80483f0: da lc 24fstpl(%esp)8048541: 53push%ebx80483f3: ed e 8 01 00 00call8048540: 53push%esix80483f3: ed e 5224 08fstpl0x8(%esp)8048541: 53push80483f2: 55push%ebx8048542: 53push%ebx80483f2: 56push%ebx8048554: 59push%ebx80483f2: 56push%ebx8048555: 58push%ebx8048400: eff f 01 00 00call8048555: 58push%esi8048401: db 52 24 08fstpl0x8(%esp)8048555: 58push8048402: db 52 24 08fstpl0x8(%esp)8048555: 58push8048402: db 52 24 08fstpl0x8(%esp)8048556: db 1c 24fstpl8048402: db 52 24	80483e4: 56	push	%esi	804853b: 58	pop	%eax
8048366:56         push         %esi         8048360:56         push         %esi           8048360:46:50         push         %ebx         8048540:56         push         %ebx           8048360:48:66:75:30         push         %ebx         8048541:53         push         %ebx           8048360:48:66:40         add         \$0x10,%esp         8048541:63         push         %ebx           8048361:31:66:61:01:024         fstpl         (%esp)         8048541:53         push         %esi           8048361:61:61:024         fstpl         0x8(%esp)         8048541:53         push         %esi           8048361:63:02         push         %esi         8048541:53         push         %esi           8048376:56         push         %esi         8048541:53         push         %eax           8048376:56         push         %ebx         8048541:53         push         %ebx           8048376:53         push         %ebx         8048541:54         gp         pop         %eax           8048371:53         push         %ebx         8048541:53         push         %ebx           8048401:64:57         push         %ebx         8048551:58         pop         %eexi	80483e5: 53	push	%ebx	804853c: 5a	pop	%edx
8048367: 53       push       %ebx       8048540: 56       push       %esi         80483ee: 86 f 3 00 00       call       80485ee       8048510: 56       push       %ebx         80483ee: 83 c 4 10       add       \$0x10, %esp       8048541: 63 c       push       %ebx         80483ee: 83 c 4 10       add       \$0x10, %esp       8048541: 56       push       %esi         80483f8: dd 5c 24 08       fstpl       0x8(%esp)       804854c: 53       push       %ebx         80483f8: dd 5c 24 08       fstpl       0x8(%esp)       804854c: 56       push       %esi         80483f6: 55       push       %esi       804854c: 56       push       %ebx         80483f6: 56       push       %ebx       804854c: 53       push       %ebx         80483f6: 56       push       %ebx       804854c: 53       push       %ebx         80483f6: 53       push       %ebx       804855c: 58       push       %ebx         8048400: 6d ff 01 00 00       call       804855c: 58       push       %esi       804855c: 58       push       %esi         804840c: 53       push       %esi       804855c: 68       push       %esi       804855c: 58       push       %esi	80483e6: 56	push	tesi	804853d: dd 1c 24	fstpl	(%esp)
ucus callB0485e0 <a>B0485e0 <a>B0485e0 <a>B0485e0 <a>B0485e0 <a>B0485c0 <a>B04</a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a></a>	80483e7: 53	push	tebx	8048540: 56	push	9651
Outlow esspOutlow esspOutlow esspOutlow esspOutlow esspOutlow essp8048376: 3d 1c 24fstpl(%esp)8048547: 5d 1c 24 08fstpl(%esp)8048376: 5d 1c 2408fstpl(%esp)8048547: 5dpush%esi8048376: 5dpush%esi8048547: 5dpush%esi8048376: 5dpush%esi8048547: 6d 1c 24%esi8048376: 5dpush%esi8048547: 6d 98 00 00 00Call80485628048376: 5dpush%esi8048551: 58pop%eax8048376: 5dpush%esi8048551: 58pop%eax8048405: cd 15c 24 08fstpl0x8(%esp)8048555: 58pop%eax8048405: dd 5c 24 08fstpl0x8(%esp)8048555: 58push%eax8048401: 53push%esi8048555: 56push%eax8048401: 53push%ebx8048551: d8 8c 00 00 00call80485628048401: 53push%ebx8048551: d8 8c 00 00 00call80485628048401: 53push%ebx8048551: d5 24 08fstpl0x8(%esp)8048401: 5410 add000 call8048561: 53push%eai8048402: 53push%ebx8048561: 53push%eai8048402: 5410 add80410; 468048561: 53push%eai8048411: 83 c4 10add 40485608048561: 53push%	80483e8: e8 13 01 00 0	call	80485e0 <a></a>	8048541: 53	push	90495£9
USUL (VEED)OUNCE OF CALL BOARD8048376: Cell and Cell	9048360; d3 C4 10	add	(hogo)	9049542: e8 DI 00 00 00	fetal	OV8020
Output: 50Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2">Colspan="2"Cols	80483f3+ 68 68 01 00 0	) Call	8048500 <a></a>	804854b: 56	Tach	seei
80483fc:       56       push       %esi       804854c:       55       push       %esi         80483fc:       56       push       %esi       804854c:       56       push       %esi         80483fc:       56       push       %esi       804854c:       58       pop       %eax         80483fc:       56       push       %esi       804855c:       59       pop       %eax         8048405:       def foi       00       00       call       8048604       cold       804855c:       def       push       %eax         8048400:       def foi       00       00       call       8048604       804855c:       ddl       fc 24       fstpl       %eax         8048400:       53       push       %easi       804855c:       ddl       fc 24       fstpl       %easi         8048402:       53       push       %easi       804855c:       ddl       fc 24       fstpl       %easi         8048402:       33       push       %ebx       804855c:       fc 24       08       fstpl       %easi         8048402:       33       push       %ebx       804855c:       fstpl       %easi       804856c:	80483f8; di 5c 24 08	fstpl	0x8(%esp)	804854c: 53	push	%ebx
80483fd: 53       push       % bex       80485fe: 53       push       % bex         80483fe: 56       push       % bex       80485fe: 53       push       % bex         80483fe: 56       push       % bex       80485fe: 69       98 00 00 00       call       80485fe: 69         80483fe: 56       push       % bex       804855fe: 58       pop       % eax         8048400: e6 ff 01 00 00       call       804804        804855fe: 56       push       % eax         8048409: 56       push       % esi       804855fe: 61 lc 24       fstpl       % esp)         8048401: 53       push       % esi       804855a: 53       push       % esi         8048402: 53       push       % esi       804855a: 53       push       % esi         8048402: 53       push       % esi       804855a: 53       push       % esi         8048402: 53       push       % esi       8048561: d1 lc 24       fstpl       % esp)         8048412: 83 c4 10       add       % ox10, % esp       8048561: 53       push       % eax         8048411: 86 c7 01 00 00       call       % ox10, % esp       8048561: 53       push       % eax         8048412: 86 c7 10       add	80483fc: 56	push	tesi	804854d: 56	push	tesi.
8048376: 56       push       %esi       8048547: e6 98 00 00 00       call       804856: <\$>         8048376: 53       push       %ebx       8048555: 58       pup       %eax         8048400: e6 ff 01 00 00       fstpl       0x80 (%esp)       8048555: 58       pup       %eax         8048400: e6 ff 01 00 00       fstpl       0x80 (%esp)       8048555: 58       push       %eax         8048401: 53       push       %esi       8048556: cdi 1c 24       fstpl       (%esp)         8048402: 55       push       %esi       8048555: cdi 1c 24       fstpl       (%esp)         8048402: 53       push       %esi       8048551: cdi 8c 00 00 00       call       8048562       sexi         8048402: 63       push       %esi       8048551: cdi 8c 00 00 00       call       8048562       sexi         8048401: 60 cd 01 00 00       call       8048562: cs>       8048561: cdi 5c 24 08       fstpl       0x80 (%esp)         8048411: 83 c4 10       add       \$0x10, %esp       8048561: 53       push       %eax         8048421: 86 c7 10 00 00       call       \$0x10, %esp       8048561: 53       push       %eax         8048421: 86 c4 10       add       \$0x10, %esp       8048561: 53       push <td>80483fd: 53</td> <td>push</td> <td>%ebx</td> <td>804854e: 53</td> <td>push</td> <td>%ebx</td>	80483fd: 53	push	%ebx	804854e: 53	push	%ebx
80483311: 53pushwebx8048301: e6 ff 01 00 00call 8048604colspan="2">colspan="2"8048400: e6 fist colspan="2">colspan="2"8048551: 58colspan="2"colspan="2"8048400: e6 da 01 00 00call8048551: 53push%easi8048412: 83 c4 10add\$0x10,%esp8048561: 56push%easi8048412: 83 c4 10add\$0x10,%esp8048561: 687f 00 00 00call8048561: 688048421: d1 c 24fstpl%esp8048561: 64c24fstpl%eax8048421: d1 c 24fstpl%esp8048561: 58pop%eax8048421: d1 c 24fstpl%esp8048561: 64c24fstpl%	80483fe: 56	push	tesi	804854f: e8 98 00 00 00	call	80485ec <s></s>
0049400: eb ii u 00 00 call       8048094 < Co 20	8048311: 53	push	tebx	8048554: 59	pop	*ecx
OutputIstopUnit (wesp)OutputOutputIstop(wesp)8048409:56pushwesi8048559:56pushwesi8048401:53pushwesi8048551:53pushwesi8048401:55pushwesi8048551:6880 00 00call8048562:538048402:53pushwesi8048551:63pushwesi8048402:63pushwesi8048550:d15c 24 08fstpl0x8(wesp)8048402:83 c4 10add\$0x10, wesp8048561:53pushweax8048415:d1 c 24fstpl(wesp)8048561:53pushweax8048412:83 c4 10add\$0x10, wesp8048561:63pushweax8048412:83 c4 10add\$0x10, wesp8048561:53pushweax8048421:83 c4 10add\$0x10, wesp8048561:53pushweax8048423:e6 b8 01 00 00call8048561:53popweax8048423: 83 c4 10add\$0x10, wesp8048561:53popweax8048423: 83 c4 10add\$0x10, wesp8048561:53popweax8048423: 83 c4 10add\$0x10, wesp8048561:53popweax8048428: 83 c4 10add\$0x10, wesp8048561:63pushweax8048429: e6 c5 01 00 00call8048573:53pushweax8048433: d1 5c 24 08fstpl0x8(wesp)8048573:53pushweax8048433: d1 5c 24 08fstpl0x8(wesp)804	8048400: e8 11 01 00 0	fatral	0048604 <0>	8048555: 58 9049556: dd 1a 34	pop	(hogen)
With the second seco	8048409 56	Iscol	see)	8048559 56	nich	(ecsp)
Both West         Both West <t< td=""><td>804840a: 53</td><td>push</td><td>%ebx</td><td>804855a: 53</td><td>push</td><td>herry .</td></t<>	804840a: 53	push	%ebx	804855a: 53	push	herry .
804840c:         53         push         webx         804850:         cdl 5c 24 08         fstpl         0x8(wesp)           8048402:         e8 da 01 00 00         call         80485ce         8048561:         53         push         wesi           8048402:         e8 da 01 00 00         call         80485ce         8048561:         53         push         wesi           8048415:         d3 1c 24         fstpl         (%esp)         8048561:         53         push         wesi           8048411:         e8 c 7 00 00 00         call         804850:         8048561:         53         push         wesi           8048411:         e8 c 7 100         00 00         call         804850:         e8 7 f 00 00 00         call         8048561:         s6         push         webx           8048420:         cd 1c 24         fstpl         (%esp)         8048561:         s0         pop         weax           8048423:         e8 b8 01 00 00         call         8048561:         s0         pop         weax           8048428:         sd 1 c 24         fstpl         (%esp)         8048561:         s0         pop         weax           8048428:         sd 2 100         add	804840b: 56	push	tesi	804855b; e8 8c 00 00 00	call	80485ec <>>
804840d: e6 da 01 00 00       call 80485cc <s>       8048561: 56       push %esi         8048412: 83 c4 10       add \$0x10,%esp       8048565: 53       push %exx         8048413: 86 c7 01 00 00       call 804864       8048565: 53       push %exx         8048418: 86 c7 01 00 00       call 804864       8048565: 53       push %exx         8048418: 86 c7 01 00 00       call 804864       8048567: 53       push %exx         8048418: 86 c7 01 00 00       call 804864       8048561: 58       push %exx         8048420: dt 1c 24       fstpl (%esp)       8048566: 58       pop %eax         8048423: e6 b8 01 00 00       call 8048560 <a>       8048566: 5a       pop %eax         8048428: 83 c4 10       add \$0x10,%esp       8048566: 5a       pop %eax         8048423: e6 b8 01 00 00       call 8048561 <ca>       8048561: d1 c 24       fstpl (%esp)         8048428: 83 c4 10       add \$0x10,%esp       8048561: d1 c 24       fstpl (%esp)         804842b: dt 1c 24       fstpl (%esp)       8048571: 55       push %eax         804842b: dt 1c 24       fstpl (%esp)       8048573: 53       push %eax         8048421: dt 1c 24       fstpl (%esp)       8048574: e8 67 00 00 00       call 80485e0 <a>         8048433: dt 5c 24 08       fstpl 0x8(%esp)       8048574: e8</a></ca></a></s>	804840c: 53	push	%ebx	8048560: dd 5c 24 08	fstpl	0x8(%esp)
8048412: 83 c4 10       add       \$0x10, %esp       8048555: 53       push       %ebx         8048415: dd 1c 24       fstpl (%esp)       8048565: 53       push       %ebx         8048418: e6 e7 01 00 00       call       804864 <	804840d: e8 da 01 00 0	call	80485ec <s></s>	8048564: 56	push	tesi i
8048415: ddl 1c 24       fstpl (%esp)       804856: 56       push %esi         8048418: e8 e7 00 00 ocall       8048504 e4>       804856: 53       push %esi         8048418: e8 e7 00 00 ocall       add \$0x10, %esp       804856: 53       push %esi         8048412: e8 b8 01 00 00       call \$0x10, %esp       804856: 58       pop %eax         8048420: ddl 1c 24       fstpl (%esp)       804856: 5a       pop %eax         8048428: e8 c4 10       add \$0x10, %esp       804856: 5a       pop %eax         8048428: e8 c4 10       add \$0x10, %esp       804856: 5a       pop %eax         8048428: e8 c4 10       add \$0x10, %esp       804856: 5a       pop %eax         8048428: e8 c5 01 00 00       call \$0x10, %esp       804856: 5a       pop %eax         8048428: e8 c5 01 00 00       call \$0x10, %esp       804857: 56       push %esi         8048429: e8 c5 01 00 00       call \$04856 am>       804857: 53       push %esi         8048429: e8 c5 01 00 00       call \$04856 am>       804857: 53       push %esi         8048433: dd 5c 24 08       fstpl (%esp)       804857: 53       push %esi         8048433: dd 5c 24 08       fstpl (%esp)       804857: 66 70 00 00 00       call 80485e0 <>>	8048412: 83 c4 10	add	\$0x10, tesp	8048565: 53	push	sepx .
8048418: es e 7 01 00 00       call 804804 <c>       804857: 53       push %ebx         804841d: 83 c4 10       add \$0x10,%esp       8048561: 53       pop %eax         8048420: d1 1c 24       fstpl (%esp)       8048566: 58       pop %eax         8048423: e6 b8 01 00 00       call 8048560 <c>       8048566: 58       pop %eax         8048423: e6 b8 01 00 00       call 8048560 <c>       8048566: 58       pop %eax         8048423: e6 b8 01 00 00       call 8048560 <c>       8048566: 58       pop %eax         8048423: e6 b8 01 00 00       call 8048569       8048566: 58       pop %eax         8048425: d1 c 24       fstpl (%esp)       8048572: 56       push %eax         8048422: e6 c 5 01 00 00       call 80485f8 <m> 8048573: 53       push %eax         8048433: d1 5 c 24 08       fstpl (%esp)       8048574: e8 67 00 00 00       call 80485e0 <c></c></m></c></c></c></c>	8048415: dd 1c 24	fstpl	(tesp)	8048566: 56	push	tesi
Output         Constraint         Constraint<	8048418: e8 e7 01 00 0	call	8048604 <0>	8048567: 53	push	Sector Contraction
0000201 Cu 1 Co 24Istpl (vesp) $00000$ call 80485c0 $< a >$ $00000$ call 80485c0 $< a >$ $00000$ call 80485c0 $< a >$ 8048428: 83 c4 10add $< 0x10, 4esp$ $8048561$ cd 1c 24fstpl (4esp)804842b: dd 1c 24fstpl (4esp) $8048572: 56$ push 4esi804842e: e8 c5 01 00 00call 80485f8 $< m >$ $8048573: 53$ push 4esi8048433: dd 5c 24 08fstpl (0x8) $8048574: e8 67 00 00 00$ call 80485e0 $< a >$	8048410: 83 C4 10	add	SUXID, sesp	8048568: e8 /1 00 00 00	call	8 0465ec <s></s>
804842b: dd 1c 24     fstpl (%esp)     80485fi: dd 1c 24     fstpl (%esp)       804842b: dd 1c 24     fstpl (%esp)     80485fi: dd 1c 24     fstpl (%esp)       804842b: dd 1c 24     fstpl (%esp)     80485fi: dd 1c 24     fstpl (%esp)       8048431: dd 5c 24 08     fstpl (%esp)     8048571: 53     push %esi       8048433: dd 5c 24 08     fstpl (%esp)     8048574: e8 67 00 00 00     call 80485e0 <a></a>	8048423; c2 b2 01 00 0	l Call	8048500 <2	8048560: 53	pop	body
804842b:         dd l c 24         fstpl         (%esp)         8048572:         56         push         %esi           804842e:         e8 c 5 01 00 00         call         8048578:         53         push         %esi           8048433:         da 5 c 24 08         fstpl         0x8(%esp)         8048574:         e8 67 00 00 00         call         80485e0 <a></a>	8048428: 83 04 10	add	\$0x10.%esp	804856f; dd 1c 24	fstnl	(%esp)
804842e: e8 c5 01 00 00 call 80485f8 <pre>d0485f8 <pre>d0485f8 <pre>d048573: 53</pre> push %ebx 8048433: dd 5c 24 08 fstpl 0x8(%esp) 8048574: e8 67 00 00 00 call 80485e0 <pre></pre></pre></pre>	804842b: dd 1c 24	fstpl	(%esp)	8048572: 56	push	sesi
8048433: dd 5 c 24 08 fstpl 0x8(%esp) 8048574: e8 67 00 00 00 call 80485e0 <a></a>	804842e: e8 c5 01 00 0	) call	80485f8 <m></m>	8048573: 53	push	%ebx
	8048433: dd 5c 24 08	fstpl	0x8(%esp)	8048574: e8 67 00 00 00	call	80485e0 <a></a>
(a)	-			<i>(</i> )		

Figure 13. Assembler instructions for the C codes in Fig. 12 with level 3 optimization.

$\begin{array}{c} 8048437: 56\\ 8048438: 53\\ 8048439: 56\\ 8048439: 56\\ 8048439: 56\\ 8048439: 56\\ 8048440: 58\\ 8048440: 58\\ 8048442: dd 1c 24\\ 8048442: dd 1c 24\\ 8048442: dd 1c 24\\ 8048446: 53\\ 8048446: 53\\ 8048446: 53\\ 8048447: dd 1c 24\\ 8048452: e8 a1 01 00 00\\ 8048442: dd 1c 24\\ 8048452: e8 a1 01 00 00\\ 8048462: 83 c4 10\\ 8048452: dd 1c 24\\ 8048453: dd 1c 24\\ 8048453: dd 1c 24\\ 8048453: dd 1c 24\\ 8048463: dd 7c 01 00 00\\ 8048462: 83 c4 10\\ 8048463: dd 5c 24 08\\ 8048471: 55\\ 8048472: 53\\ 8048474: 53\\ 8048472: 53\\ 804848472: 53\\ 80484848482\\ 8048484821: 53\\ 8048484821\\ 8048484821\\ 8$	push %esi push %esi push %esi push %esi call 8048604 <d> pop %eax pop %eax fstpl (%esp) push %esi call 8048604 <d> fstpl %esp) call 8048560 <d> fstpl %esp fstpl %esp push %esi push %esi push %esi push %esi push %esi push %esi push %esi push %esi push %esi push %esi</d></d></d>	8048579: 83 c4 10 8048577: c3 c4 10 8048571: c6 74 00 00 00 8048584: 83 c4 10 8048587: c6 74 00 00 00 8048587: c6 74 00 00 00 8048587: c6 1c 24 8048592: c6 1c 24 8048592: c6 1c 24 8048593: c6 1c 24 8048593: c6 1c 24 8048593: c6 1c 24 8048533: c3 c4 10 8048534: c3 c5 00 00 00 8048535: c4 c5 10 8048535: c4 c5 10 8048536: c9 05 cc 86 04 08 8048550: c6 05 50 86 04 08 8048550: c6 05 50 8048550: c6 05 8048550: c6 05 8048550: c6 05 8048550: c4 10 8048550: c6 05 8048550: c6 15 8048550: c6 15 8048550: c6 15 8048555: c6 15 80485555: c6 15 80485555555555555555555555555555555555	add \$0x10,%esp fstpl (%esp) call 80485f8 4TP add \$0x10,%esp fstpl (%esp) call 80485f8 4TP add \$0x10,%esp fstpl (%esp) call 8048560 <2> add \$0x10,%esp fstpl (%esp) call 8048604 <2P fstp %st(0) mov %esx,0xfffffff0(%ebp) fstp %st(0) fld 0xffffffff0(%ebp) fldl 0xfffffff0(%ebp) fldl 0xfffffff0(%ebp) fldl 0xfffffff0(%ebp) fldl 0xfffffff0(%ebp) fldl 0xffffffff0(%ebp) fldl 0xffffffff0(%ebp) fldl 0xffffffff0(%ebp) fldl 0xffffffff0(%ebp) fldl 0xffffffff0(%ebp) fldl 0xffffffff0(%ebp) fldl 0xffffffff0(%ebp) fldl 0xffffffffffffffffffffffffffffffffffff
8048482: e8 65 01 00 00 8048487: 83 c4 10	call 80485ec <s> add \$0x10,%esp</s>	80485dd: c3 80485de: 89 f6	ret mov %esi,%esi
	push %ebp mov %esp,%ebp fldl 0x8(%ebp) faddl 0x8(%ebp) leave ret nop	080485f8 ≪>: 80485f8: 55 80485f9: 89 e5 80485fb: dd 45 10 80485fe: dc 6d 08 8048601: c9 8048602: c3 8048603: 90 9090604 cb:	push %ebp mov %esp,%ebp fildl 0x10(%ebp) fsubrl 0x8(%ebp) leave ret nop
80485ec: 55 80485ec: 89 e5 80485ef: dd 45 10 80485f2: dd 45 10 80485f2: dc 40 8 80485f5: c9 80485f5: c3 80485f7: 90	push %ebp mov %esp,%ebp fldl 0x10(%ebp) fmull 0x8(%ebp) leave ret nop	8048604: 55 8048607: dd 45 10 8048607: dd 45 10 804860a: dc 7d 08 8048604: c3 8048604: c3 8048604: c3	push %ebp nov %esp,%ebp fldl 0x10(%ebp) fdivrl 0x8(%ebp) leave ret nop

(a) (cont'd)

80483e7: 90	nop	80483ff: 90 (b)	ncp
000483dC <tn>: 00483dC: 55 80483dd: 89 e5 80483df: dd 45 10 80483e2: dc 4d 08 80483e5: c9 80483e6: c3</tn>	push %ebp mov %esp,%ebp fidl 0x10(%ebp) fmull 0x8(%ebp) leave ret	0048314 <0>: 8048314: 55 8048315: 89 e5 8048317: dd 45 10 8048316: cd 7d 08 8048316: c3 8048316: c3	push %ebp mov %esp,%ebp fldl 0x10(%ebp) fdivrl 0x8(%ebp) leave ret
080483d0 <a>: 00483d0: 55 00483d1: 89 e 5 00483d5: dd 45 10 00483d6: dc 45 08 00483d9: c9 00483d9: c3 00483d9: c3</a>	push %ebp mov %esp,%ebp fidl 0x10(%ebp) faddl 0x8(%ebp) leave ret nop	080483e8 ≪>: 80483e8: 55 80483e9: 89 e5 80483eb: dd 45 10 80483f1: c9 80483f1: c9 80483f2: c3 80483f4 ≪>: 80483f4: 55 80483f4: 55 80483f4: 55 80483f4: dd 45 10 80483fa: dc 7d 08 80483f4: c3 80483fe: c3 80483ff: 90	push %ebp mov %esp,%ebp fildl 0x10(%ebp) fsubrl 0x8(%ebp) leave ret nop
$\begin{array}{l} 04834b: dd 5d f8\\ 0048360: 89 f6\\ 0048350: ff 75 fc\\ 0048351: ff 75 fc\\ 0048362: ff 75 f8\\ 0048362: ff 75 f8\\ 0048362: ff 75 fc\\ 0048362: ff 75 fc\\ 0048362: ff 75 fc\\ 0048362: ff 75 fc\\ 0048374: 83 ec 08\\ 0048374: 83 ec 08\\ 0048374: ff 75 fc\\ 0048383: e8 54 00 00 00\\ \end{array}$	fstpl 0xffffff8(%ebp) pushl 0xffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) pushl 0xfffffff6(%ebp) call 80483dc	804838b: dd 1 24 804838b: dd 1 24 8048396: dd 1 24 8048393: 83 c4 10 8048393: 83 c4 10 804839: e8 32 00 00 00 804839: e8 32 00 00 00 804839: e8 32 c4 10 80483a4: e6 27 00 00 00 80483a4: e6 27 00 00 00 80483a4: e6 27 00 00 00 80483b4: dd 5 f8 80483b4: dd 5 f8 80483b4: dd 5 c6 84 04 00 80483b4: dd 5 c7 8 80483c7: 74 87 80483c9: 31 c0 80483c1: c9 80483c2: c3 80483c2: c3 80483c2: c3 80483c2: c3 80483c2: c3	call 8048340 <>> call 8048340 <>> call 8048340 <>> fstpl (%esp) fstp %st(0) field 0xfffffff8(%ebp) fidd 0xfffffff8(%ebp) fidd 0x80484e0 fucomp %st(1) fnstsw %ax add \$0x10,%esp test \$0x5,%ah fstpl 0xfffffff8(%ebp) je 8048350 <main+0x14> xor %eax,%eax leave ret lea 0x0(%esi),%esi</main+0x14>
804833c: 55 804833d: 89 e5 804833f: 83 ec 08 8048342: d9 05 d8 84 04 08 8048348: 83 e4 f0	push %ebp mov %esp,%ebp sub \$0x8,%esp fids 0x80484d8 and \$0xffffff0,%esp	8048388: 83 c4 10 804838b: dd lc 24 804838e: e8 49 00 00 00 8048393: 83 c4 10 8048396: dd lc 24	add \$0x10, %esp fstpl (%esp) call 80483dc <n> add \$0x10, %esp fstpl (%esp)</n>

Figure 13 (cont'd). Assembler instructions for the C codes in Fig. 12 with level 3 optimization.

#### Conclusions

Our results point to a few fundamental shortcoming inherent in GP, when applied to obfuscation for asset protection.

First, the very nature of evolution by reproduction and mutation makes it unlikely that a random population will converge to any complex solutions in a tractable time frame. In addition, the primary mechanism for obfuscation is complexity through introns, and the presence of introns will by definition increase the number of operations required to evaluate a function, for example. Therefore, using the tools and approaches detailed herein, it is practical to apply GP only to relatively simple functions and algorithms, and the natural process of obfuscation by introns leads to dramatic penalties in performance when compared to an unobfuscated solution, as evidenced by the benchmarks described above.

Second, introns are potentially very easy to identify, especially in algorithms that use basic operators like arithmetic. This means not only that a human might quickly simplify smaller algorithms obfuscated using a GP, but also that a computer can easily simplify the algorithm to its target form through compiler optimization, for example. While steps can be taken to mitigate this shortcoming, *e.g.* by replacing basic operations with function calls, this procedure is potentially impractical in both its logistical implications and its impact on the performance of the obfuscated algorithm.

We therefore conclude that GP is not an appropriate mechanism for the obfuscation of code because complex functions can not be reproduced exactly, and the obfuscation of multiple smaller functions will lead to unacceptable penalties in performance.

#### References

- [1] John R. Koza, Genetic Programming (MIT Press, 1992).
- [2] http://garage.cps.msu.edu/software/software-index.html#lilgp
- [3] John R. Koza, Martin A. Keane, and Matthew J. Streeter, *Scientific American*, Februrary 2003.
- [4]Christian Collberg, Clark Thomborson, and Douglas Low, Technical Report 148, Dept. of Computer Science, University of Auckland (1997).
- [5] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang, Electronic Colloquium on Computational Complexity, Report No. 57 (2001).
- [6] Jostein Gaarder, Sophie's World (Phoenix House, 1995) p. 256.
- [7] Philip L. Campbell, SAN2004-2198 (2004).
- [8] S. Luke, Proceedings of GECCO 2000, 228-235 (2000).
- [9] Peter W.H. Smith, in <u>Soft Computing and its Techniques</u>, edited by R. John and R. Birkenhead. pp. 166-171. Physica-Verlag (1999).

## Distribution

1	MS 0123	Donna L. Chavez (LDRD Office), 01011
1	MS 0455	Hamilton E. Link, 05632
1	MS 0785	Cheryl L. Beaver, 05614
1	MS 0785	Philip L. Campbell, 05616
2	MS 0899	Technical Library, 09616
1	MS 1205	James R. Gosler, 05004
2	MS 1411	Corbett C. Battaile, 01834
10	MS 1411	Michael E. Chandross, 01834
1	MS 1411	H. Eliot Fang, 01834
	160.0010	

1 MS 9018 Central Technical Files, 08945-1