Abstract

Reengineering of software systems is a topic of importance and in coming time it will be gaining more attention in the world of software systems. Reengineering make the system new to work for another life span. Scope of reengineering is vast and challenging. Reengineering is to reduce the expenses on software systems in the organizations. Reengineering has more scope in the world of software than in the world of hard ware objects. Software systems and software objects do not wear and tear out like hardware objects in the real world. In this paper, we are mainly concerned with the reverse engineering part for one of crucial language called Assembler. Being almost like a machine level language, understanding the program is not a cup of tea for all the programmers. So the problem is being discussed and solutions are tried to be retrieved.

I. INTRODUCTION

“The migration of legacy systems is a difficult endeavour because traditional methods have two principal deficiencies. First, they fail to capture the context of a system, i.e., its domain. Second, the legacy system’s comprehension results are not directly usable for the system evolution. They propose the construction of executable domain models to
alleviate both problems. The construction of an executable domain model entails a process of domain analysis that leads to a domain model, as well as the transition of the former to an executable state. The domain model provides domain expectations that drive legacy system understanding. The executable domain model provides a medium in which the result of the legacy system comprehension can be recorded. In fact, the executable domain model is instantiated using the system requirements derived during program comprehension. The artifact thus created takes the role of the re-engineered program. Their work uses the technique of object-oriented frameworks (OOF) as the executable domain model representation.

![Diagram](attachment:image.png)

**Figure 1: Using Domain Models for Software re-Engineering [7]**

### 1.1 Reverse Engineering for ASSEMBLER

As we have seen in the above sections that it is very important to have domain knowledge prior to start reengineering process. In the legacy systems it is one of the major pain areas where assembler is involved. It is involved in many applications having mainframes as their core technology. Most of the programs are written in COBOL...
though, but still there are many systems which are having assembler as their base language. For such systems time has come to get converted into some new technologies or at least COBOL.

II. UNDERSTANDING THE ISSUE

It may sound easy but it is not that simple as it looks. There are lot of complexities involved in this. Some of the key issues involved in this are:

- It is a machine level language, so involved lots of instructions.
- It is having registers as one of the operands.

There are many instructions which are specific to assembler only and there is no such instruction or equivalent to that in any other programming language. So it becomes tedious task for any engineer to understand the meaning of such instructions. Example of such instruction is load address (LA). These instructions are of utter importance, but have no significance in other programming languages. These types of instructions are needed to be handled carefully, as these are pointing to some memory location and might involve data manipulation. So it is very important to make note of it.

Another major problem is having register as one of the operands. Registers are being used for data manipulation, address resolution, saving a return point or many more. So how to handle these are of great importance. Since does not have a particular data type which is being stored, so direct conversion of such storage is not easy. So handling registers is one of main pain area.

III. PROPOSED SOLUTION

There are mainly two solutions are proposed which are as follows:

3.1 Line by Line Conversion

In this type of conversion, I propose that each line of the assembler code will be converted into the pseudo code. This pseudo code then can be used for the system understanding. In this approach, each instruction will be converted into its respective pseudo code.

But the issue with this approach is that there are many instructions that is specific to assembler and does not have anything to do with the program or its functionality. Or, in this approach we will have a large number of overhead and this will create problem for the normal user to make sense of each line. Example of this approach can be seen in fig 2. To elaborate this, there are many instructions that are used only for the address calculation. It has no deal with the program function. But in this approach these lines will also be converted into pseudo code, which is of no use. Neither the analyst nor the programmer who is going to write new code has to do with this. Only thing he might be interested will be the variable or position in any particular variable where data updation has to be done. The later part can provide an extra edge as well, as it will be easy for one to note the perfect location for the data manipulation. The biggest benefit of this approach will be that the engineer who is going to do forward engineering will have extra information from the previous code, which could be helpful for the new code.
3.2 Group Conversion

In this type of conversion, a group of assembler code will be translated into the pseudo code. This pseudo code will then be used by the analyst to understand the program logic and derive the functionality of the system. This approach will generate the pseudo code will be able to give the overview of the program. What it does? How it does? The result generated from this conversion will be helpful for the layman also that what that program is able to perform.

This approach will be able to generate the precise information from the code and knowledge can be extract easily with this approach. This will be very useful who is from the non technical background. The Domain knowledge which is required for the reverse engineering will be easily extracted from this approach (See fig. 3).

The main problem with this approach may be to provide enough technical knowledge to the engineers who are going to do forward engineering of this. A classic example with respect to assembler could be that, there are many data type conversions are involved which is purely technical part. This approach may generate a document which omit that conversion part and would result only in some functional lines. So one of the most important aspects of data type conversion may be lost, which could be insignificant from functional perspective, but important from technical point of view.

4. TOOLS AND TECHNOLOGIES

There are various tools and technologies which we are going to use. These are:

- Lex
- Bison/Yacc
- C

IV. HIGH LEVEL PROCESS FLOW

The figure 4 shows the overall flow of the project.
V. LIMITATIONS

Like any other solution or approach, these will also have some limitations. Though all cannot be described at first place itself, but then also we will try to figure out few. Some of the limitations could be as follows:

- Pre-processing of the original code may be required. This includes removal of comments from the code and all unnecessary parts should also be removed. In short only executable assembler code should be the input for the software.
- This may required many input files, like one with source code alone, another with data section alone, another could be the calls to programs and procedures.

These are the two main limitations which can be sought at this point of time. We will need to see if some more limitation is there. Also we will have to look for the solutions for these limitations. These should be removable with time.

VI. SUMMARY

In this paper, we have discussed the various solutions possible for the reverse engineering approach for Assembler. Two approaches are discussed and small example has been given out to make better understanding. Also tools and technologies which will be used to develop pseudo code generator tool has been discussed and brief details has been shared. Few things which are expected from the work are improvement in the process of reverse engineering for the Assembler. This should be helpful in reducing the efforts normally put by the engineers. Also the overall cost incurred and time devoted should be minimized. This should be able to give the high level design of an existing system which could be used for the new system design.
VII. CONCLUSION

Anyone looking for reengineering for assembler codes can make use of one of the methods present above. This will be able to help them reduce the time to think of an approach and will give a kick start to proceed ahead. Most of the time wasted to decide on the approach will be saved and help them reduce the time and hard work.

VIII. FUTURE WORK

These given methods are new in the field of reverse engineering for legacy systems having assembler as their base. The future work is to test these methods for suitability to fit on the basis of analysis of current and past data. These methods can be accepted as it is or improved or rejected. Once fit and fine these methods will help in reengineering the legacy software with optimal cost. This work will be beneficial to the both communities, the software managers and the software engineers.

Also the sustainability of the approach and product need to be tested. It has to be analysed by the various people and various types of code needs to be checked. It may be possible that it is not durable for all and may require some alteration. This could be less for some cases and it may be more for some cases. Since it may involve some amount of alteration, so cost also becomes one of the important factors. For few cases, there is possibility that cost incurred for alteration is much more than the entire process. For such cases, it may not be useful and other alternatives need to be looked for.

IX. REFERENCES


AUTHOR’S BIOGRAPHIES

Author Anurag Goyal was born on 9th Feb 1989 and belongs from Bhilai. He has completed his BE (IT) from Shri Shankaracharya College of Engineering and Technology, Bhilai in the year 2010 and currently pursuing MTech (IT) from Shri Shankaracharya Technical Campus, Bhilai. He has worked with Cognizant Technology Solutions, Chennai for around 3.5 years and has
experience in legacy systems. He has interest in legacy systems modernization and do some research activities in that area.

Guide Ms. Madhuri Gupta has guided him preparing this research paper. She is currently pursuing PHd from Chhattisgarh Swami Vivekanand Technical University, Bhilai. She is currently Assistant Professor in IT Dept in Shri Shankaracharya Technical Campus, Bhilai and having an experience of around 7-8 years. She has interest in Cyber Crime and security topics.

TO CITE THIS PAPER