ISSN 2319 - 5975

Volume 8, No.3, April - May 2019

International Journal of Networks and Systems

Available Online at http://www.warse.org/IJNS/static/pdf/file/ijns12832019.pdf

https://doi.org/10.30534/ijns/2019/12832019

Implementation of Neural Network in Assembler

Benjamin A Jacob[1],Arjun Raj[2], Akshaymon K V[3] , Nimmymol Manuel[4]

[1]Mangalam College of Engineering, Kottayam, India, thisisbajacob@gmail.com

[2]Mangalam College of Engineering, Kottayam, India, arjunraj.pala@gmail.com

[3]Mangalam College of Engineering, Kottayam, India, kvakshaymon@gmail.com

[4]Mangalam College of Engineering, Kottayam, India, nimmymol.manuel@mangalam.in

ABSTRACT

Assemblers are used to convert assembly language programs into corresponding machine code and therefore machine dependent system software. It is the great diversity in existing microprocessor architectures that prohibits the universal use of any simple software products. This paper introduces Neural Network in assemblers so that the assembly language of a computer can be extended which helps to process a language that is completely different from native assembly language.

Key words: Assembler, Back propagation algorithm, Database, Neural network.

1. INTRODUCTION

Assembler is machine dependant system software. The machine dependency of assemblers can be considered as a drawback because of the use of different instruction sets and other specific machine dependent features. As each machine contains different instruction set, it is almost impossible to run an assembly language instruction of one system in another [3]. To solve this, several cross platform assemblers such as PASM, NASM are used. Cross platform assembler is an assembler that generates code for a platform other than the one on which it runs or which is running on a host platform and generates a machine code for another destination platform [4], [5]. But these too have their own limitations as they are limited by the non identical instruction sets in various processors. This causes the manufacturer's to make assembly language which supports each different machine. This paper aims to provide a new pathway in the field of assembly language. Instead of creating different assemblers, a single universal assembly system can be implemented with the help of Neural Network being introduced to it.

2. BASIC CONCEPTS

The Basic concepts used are a multi layered feed forward network form of Neural Network, and the back propagation algorithm which uses supervised learning technique for training the neural network. These concepts are implemented in assemblers in order to make it universal.

2.1 Neural Network

Neural Network is an information processing system, which resembles the behavior of the brain to solve real time application problems. The Neural Network itself is not actually an algorithm, but a framework for many other machine learning algorithms to work together and process complex data inputs [5].

2.2 Multilayer Feed Forward Networks

Network architecture is the arrangement of, a set of neurons to form layers and the connection pattern within and between these layers [1]. A multi layered feed forward network is used here because in back propagation learning algorithm which is used to train the Neural Network here, is normally applied to such networks (Multi layered feed forward network).

A multi layered feed forward network contains an input and output layer along with a set of hidden layers. Each connection between the layers will be associated with a weight value which is updated using the learning algorithm used. Figure 1 represents the architecture of Multi layered feed forward network.

2.3 Supervised Learning

Here the connection weights of the Neural Networks are adjusted and updated using supervised learning rule. Learning is the process by which a Neural Network adapts itself to a stimulus by making required parameter or weight adjustments in order to produce the desired output [1]. In supervised learning an input vector is provided to the Neural Network which in turn produces the corresponding output vector which is the actual output. This output vector is compared with the desired output

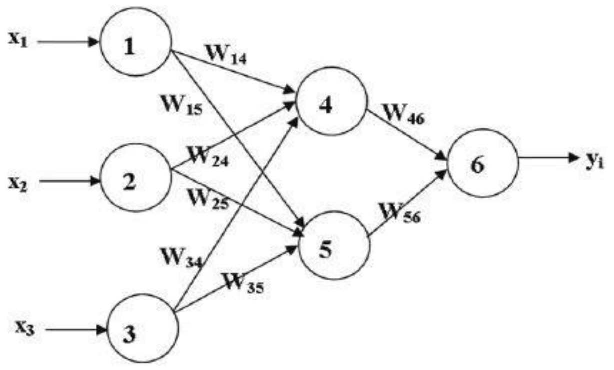

Figure 1: Architecture of Multi - layered feed forward network [8].

and if there is any difference between the two output vectors an error signal will be generated using which the weight adjustments are done. Generally, this learning process is to tune the parameters in the model and to produce output that best fits the output in the learning set for a given input [4].

2.3 Back propagation Network

Back propagation is a method used in Neural Networks to calculate the gradient that is needed for the calculation of weights to be used in the network [2]. There are two phases in the back propagation learning algorithm, a feed forward phase and a back propagation of error phase. In the first phase the input signal received by each neuron is transmitted to the processing unit in the next layer after calculating the net input in each unit. In the second phase each unit in the output layer receives a target pattern which is compared with actual output to find out the error, if any. If an error is being detected it is fed backward to the neurons in the preceding hidden layer in order to update the weights. After that, the contribution of hidden layer for the error occurred in the output layer is calculated to update the weights of neurons in the previous layer. This process is repeated for all the preceding layers .Figure 2 shows the architecture of a back propagation network.

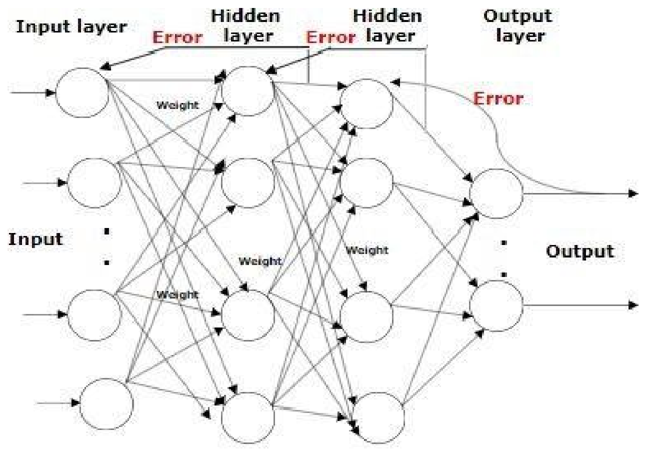

Figure 2: Architecture of back propagation network.

3. IMPLEMENTATION OF NEURAL NETWORK

Assemblers are machine dependent. They are related to the architecture of the machine on which they run. This is because of the difference in the addressing modes as well as the instruction sets that vary with the processors. Therefore different processors require different assembly language programs [9], [10]. The basic function of assembler is shown in Figure 3. By introducing the concept of Neural Networks, assemblers can be modified to accept any assembly language program irrespective of the processor. The Neural Network is used to group the given instruction into a particular set and to find out the addressing modes and instruction type used in each instruction. And with this information the conversion process is being done.

The first step is to create a database which contains all possible instructions sets, of all the available processors. In the database each table contains the instruction sets of a particular processor Each column contains the different instructions available in a particular processor and each row contains the different addressing modes used in that processor. Table 1 represents how the instructions (only some instructions of 8086 processor) are being stored in the database. Once the entire database is created it can be used any number of times for conversion.

Once the database is created the initial phase is over. The next phase is to use two neural networks to identify the type of each instruction (eg. whether it is an addition instruction or multiplication instruction and so on.) and the addressing modes used in the given assembly language. For that, at first the neural network is to be trained using back propagation algorithm. The aim of using back propagation algorithm is to train the multi layer feed forward neural network to achieve a balance between the nets ability to respond (memorization) and its ability to give reasonable responses to the input that is similar but not identical to the one that was used in training [1].

The first neural network identifies the group to which the given instruction belongs. The Neural Network is trained using the instructions from different processors using the technique of supervised learning. The advantage of back propagation algorithm is that it need not to be trained with all the instructions of all the processors, rather a small group of instructions from each processors are chosen and the network is trained using that. On receiving the assembly language instructions the first neural network identifies the type of instruction (i.e. to which group of instruction do the given instruction belong to) and the second neural network uses the output from the previous neural network to identify the addressing mode of the instruction. Both these data are found and stored in two separate variables.

The information obtained from the network is used to find the corresponding instruction of the specified processor. A platform independent programming language is used for this purpose. The database is used in this phase for the purpose of finding out the equivalent instruction for conversion. The algorithm given below describes the step to be performed during the conversion process

Connection and conversion Algorithm

Step 1: Initialize the variable 'table_name' with the name of the Processor used in the Host machine.

Step 2: Connect to database

Step 3: Send SQL query

Step 4: The corresponding instruction is fetched from the table 'tablename' with the help of any searching algorithm (Here, the output from neural networks is used for searching)

Step 5: Assign the data values to the retrieved instruction

Step 6: Close the connection.

At first the database is being searched to find out the table which corresponds to the processor used in the host machine. Once the table is found out, the column corresponding to the instruction type is selected and then the row corresponding to the required addressing mode is being selected. After that the corresponding instruction is fetched from the table and is used for the conversion process. Thus the given instruction in any assembly language is converted to the assembly language corresponding to the processor used in the host machine. Now the assembler can convert the assembly language to its machine equivalent format.

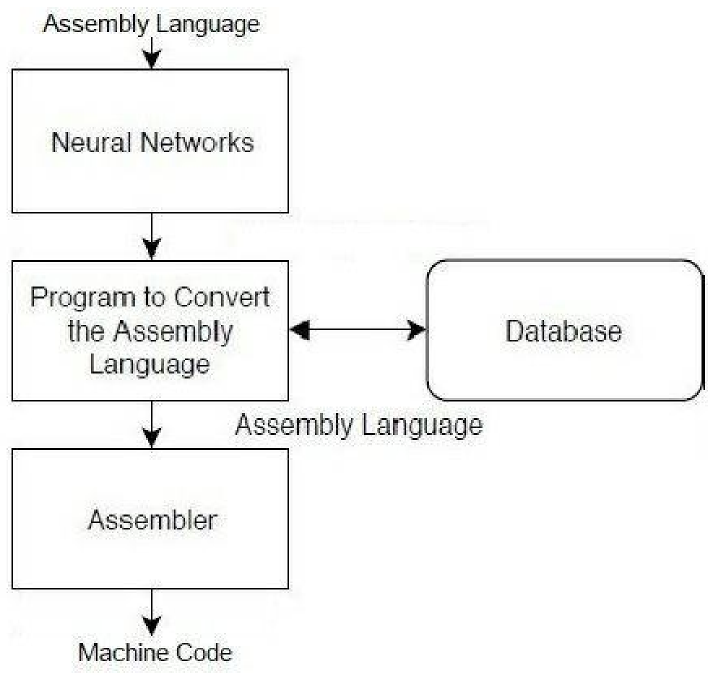

Figure 3: Flow chart of Conversion of assembly language.

Table 1: Instruction set in 8086 microprocessor

Addressing Modes	Move	Addition	Subtraction	Multiplication	Division
Register	MOV AX,BX	ADD AX,BX	SBB AX,BX	MUL BH	DIV BX
Immediate	MOV AX,5000H	ADD AX,5000H	SBB AX,5000H	-	-
Indexed	MOV AX,[SI]	ADD AX,[SI]	SBB AX,[SI]	MUL AX,[SI]	DIV AX,[SI]
Register relative	MOV AX,50H[BX]	ADD AX,50H[BX]	SBB AX,50H[BX]	MUL AX,50H[BX]	DIV AX,50H[BX]

Table 2: Comparison

Cross Assembler(Macro based)	**Neural Assembler**
Mostly Hardware Specific (It depend upon the type of processor)	It does not depend upon the type of host processor
The chance of getting an error is high	The chance of getting an error is low
Group macros are used.	Neural networking is used.
Less User friendly	User friendly
Difficulty in maintenance and update	Easy to maintain and update
Difficult to implement	Easy to implement

5. RESULT

Implementing neural network in assemblers resolves the basic drawbacks of macro based cross assemblers which is shown in Table 2 [11].

6. CONCLUSION

Assembler being machine dependent prevents the generalization of assembly languages among processors. Introducing the concept of neural network helps in modifying the machine dependent features of the assembler. This modification allows different assembly language programs to run on different machines.

The most important drawback of a cross assembler is that it can’t be used for all the processors. For example, cross assembler designer for 8080 processor can’t be implemented or used in machines using 6500 processors. The basic reason behind that is, 6500 instruction set contains explicit addressing modes where as 8080 does not. This can be overcome by implementing neural network in assemblers, as the addressing mode of each instruction is found out with the help of neural network and then it is converted accordingly [6], [7], and [11].

A further step would be the use of efficient search optimization techniques so that the time consumed whereas searching the database can be reduced. And the use of back propagation algorithm can be expensive, as the network converges slowly. And by using other efficient training algorithms will help to avoid that.

ACKNOWLEDGEMENT

The authors wish to thank Principal Dr J M Mathana , Dr Vinod P Vijayan H.O.D, Computer Science Department, Mrs .Neena Joseph for the proper guidance, valuable support and helpful comments during the proofreading .

REFERENCES

1. S. N. Sivanandam and S. N. Deepa. **Principles of Soft Computing**, 2nd ed, pp. 17-70, 2016.
2. Goodfellow, Ian; Bengio, Yoshua;Courville,Aaaron. **Deep Learning,** MIT Press, 2016, pp 196. ISBN 9780262035613.
3. Agrawala, A. K., & Rauscher, T. G. **Foundations of microprogramming: architecture, software, and applications**, *Academic press,* ch3, 2014.
4. Claus Svarer. **Neural Networks for signal processing**, *CONNECT, Electronics Institute Technical University of Denmark DK-2800 Lyngby Denmark, Ph.D. Thesis,* ch2, pp 17-24
5. Build with AI | DeepAI. *DeepAI*. Retrieved 2018-10-06.
6. https://en.wiktionary.org/wiki/cross_assembler.
7. https://www.c64-wiki.com/wiki/Cross_Assembler.
8. https://www.google.com/url?sa=i&source=images&cd= &cad=rja&uact
9. J. U. Duncombe. **Infrared navigation—Part I: An assessment of feasibility**, *IEEE Trans. Electron*
10. Devices,vol. ED-11, pp. 34-39 ,Jan 1959
11. Karel R. Tavernier and Paul H. Notredame, **Macro-Based Cross Assemblers** *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-6, NO. 4, JULY 1980.* https://doi.org/10.1109/TSE.1980.234489