Deep Belief Networks with synchronal calculations of Energy Consumptions, and MIPS Assembly Programs amongst robust designs

Paul Amoruso
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*— Many important topics regarding the assembly code and Deep Belief Networks will be the focus. The assembly program written will find all the instances of a certain word that appears in an input statement and at what locations. By providing significant workings of strings, addresses, data, and memory contents this program successfully filters out the specific word in the input statement. With the ability to loop through the registers that store the individual letters of each word in order to find the specific word in the input statement. With the program also checking for capital and lowercase letters that keeps track of indexes based on words rather than spaces. This paper highlights novice ideas on Deep Belief Networks (DBNs) and what they are comprised of. By looking at the reference papers, a picture can be painted on what Deep Belief Networks are, and why they contain Restricted Boltzmann machine (RBM).**

Keywords—Deep Belief Network (DBN), Neural Network, Restricted Boltzmann machine (RBM), deep Convolutional Neural Networks, FPGA, Generative Adversarial Network algorithms, generative models, VLSI.

I. Introduction

A. Project Design

In this MIPS code, the objective was to increase understanding of strings, addresses, data, and memory contents. With a goal of string manipulation in order to find the number of times a specific word appears in an input statement and at what indexes the specific word appears. While it is not needed, this code requires a user input of an input statement, rather than having a hard-coded input statement to parse through. The program starts off by printing a string that requests the user to input a statement, then the input statement is stored in a space of 1000 called "string." Next, the program asks for a specific word to search for in the input statement and is stored in space of 11 called "words." Once the statement and specified word are collected the program moves onto a label called "StoreCharOfWord" which stores all the letters from the word to individual ten registers from $t6 to $s5. By storing each of the possible letters in individual registers, there is an ability to manipulate the letters for checking upper or lowercase letters without changing the original word given and use of extra loops to parse through an array. After storing all the letters of the word, the focus jumps to the code that loops the input statement one character at a time using $t2 and comparing the current letter in $t2 with the registers $t6 through $s5. In overview, the program traverses the input statement till it finds the first letter in the word (in either upper case or lowercase form), then it moves to the next letter in the input statement and the word to see if they match as well. If so, the process continues until it hits the end of the word, which prompts the program to jump to the label "Count", which stores the current index and increments the number of times the word has been found at the moment. This process of looking for the word repeats till the end of the input statement, where it is then at the printing stage. There are three printing labels that print the first part of the output all the way to the end of the word searched for, a second label that contains the code to print the end of the sentence " " was found in the input statement: " and the string that leads up to the indexes the word was found at. Then the second to last label, loops through the space "indexesvals" allocated for all the indexes the specific word appears in the input statement. As it reaches the end of printing all the indexes, the program jumps to label "end" where it prints a period at the end of the indexes and load the value 10 to $v0 to end the program. A, concise and clear high-quality flowchart of the program can be seen in Fig. 1.

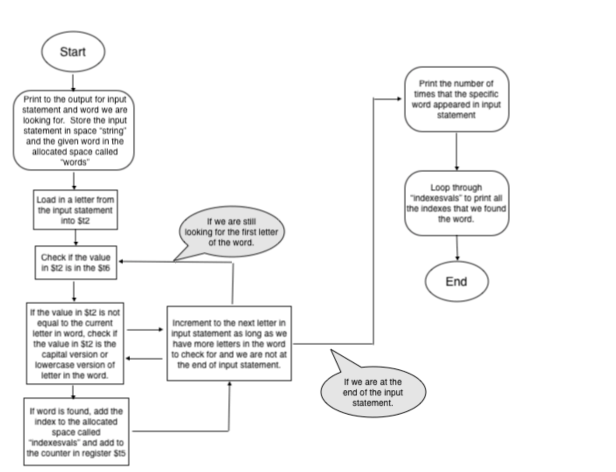
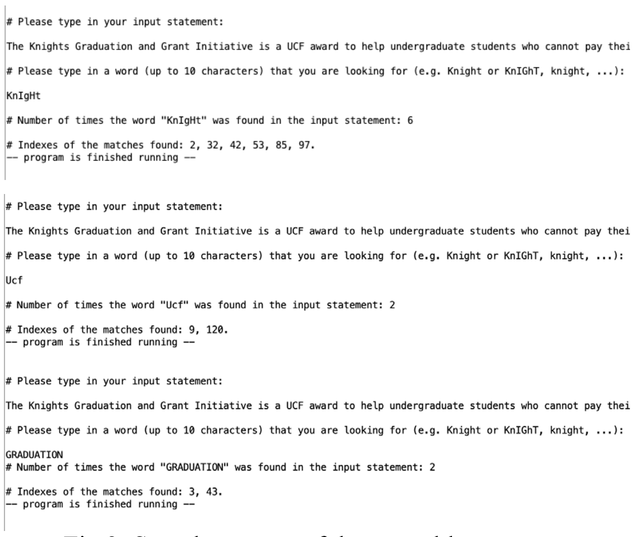

Start
Print to the output for input statement and word we are looking for. Store the input statement in space "string" and the given word in the allocated space called "words"
Load in a letter from the input statement into $t2
Check if the value in $t2 is in the $t6
If the value in $t2 is not equal to the current letter in word, check if the value in $t2 is the capital version or lowercase version of letter in the word.
If word is found, add the index to the allocated space called "indexesvals" and add to the counter in register $t5
If we are still looking for the first letter of the word.
Increment to the next letter in input statement as long as we have more letters in the word to check for and we are not at the end of input statement.
If we are at the end of the input statement.
Print the number of times that the specific word appeared in input statement
Loop through "indexesvals" to print all the indexes that we found the word.
End

Fig.1: Flowchart of the assembly program.

B. Test Cases

In the testing process of this code, the program was stress tested with multiple words with various lengths ranging from one letter to ten letters. It was also tested with searching for words with capital and lowercase letters to ensure that the program can find all possible words no matter the length or case. Some of the cases that were tested with the program was with the input statement *"The Knights Graduation and Grant Initiative is a UCF award to help undergraduate students who cannot pay their tuition and their difficulty would not allow them to finish their degree. The Knights Success Grant is the most well-known program inside the Knights Graduation and Grant Initiative. In order to be awarded the Knights Success Grant, you need to be referred but it does not mean that all students who are referred will be awarded the grant. The students who want to apply for the Knights Success Grant need to submit a required application and complete the Knights Success Grant web course. For more information, you can stop by their office in the Registrar's Office on the main campus of UCF."* And with the search words: "KnIgHt", "Ucf", and "GRADUATION". By using that specific input statement, since that is the input statement given in the assignment. Given the word "KnIgHt" it can be checked that it appears 6 times with indexes at 2, 32, 42, 53, 85, 97, and the word "Ucf" appears two times at indexes 9, and 120. Using the all Caps ten letter word "GRADUATION", it has been proved that the code can find it located twice in indexes 3, and 43. A, concise and clear high-quality output of the program can be seen in Fig. 2.



```
# Please type in your input statement:

The Knights Graduation and Grant Initiative is a UCF award to help undergraduate students who cannot pay thei

# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, ...):

KnIgHt

# Number of times the word "KnIgHt" was found in the input statement: 6

# Indexes of the matches found: 2, 32, 42, 53, 85, 97.
-- program is finished running --


# Please type in your input statement:

The Knights Graduation and Grant Initiative is a UCF award to help undergraduate students who cannot pay thei

# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, ...):

Ucf

# Number of times the word "Ucf" was found in the input statement: 2

# Indexes of the matches found: 9, 120.
-- program is finished running --


# Please type in your input statement:

The Knights Graduation and Grant Initiative is a UCF award to help undergraduate students who cannot pay thei

# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, ...):

GRADUATION
# Number of times the word "GRADUATION" was found in the input statement: 2

# Indexes of the matches found: 3, 43.
-- program is finished running --
```

Fig.2: Sample outputs of the assembly program.

II. DBN Circuit

At first glance the study of Deep Belief Networks (DBN), for anyone familiar with the topic of neural networks, it is evident that it is a graphical model aka a deep neural network composed of many layers that are "hidden" with connection from one layer to the next without any connections in the same layer. According to the reference [1], they utilized a 784 x 200 x 10 DBN circuit in SPICE for a pattern recognition application using the MNIST dataset. The idea of DBN's are comprised of different parts, such as in this case where reference [1] implements Restricted Boltzmann machine (RBM) in order to have DBN with impressive learning abilities.

Similar to reference [1], reference [3] introduces Deep Belief Networks as a network made of RBMs. Due to the fact that DBNs are made of RBMs, reference [3] explains many aspects of the machine from the size of the nodes such as 2000 X 500 nodes, as well as explaining how the RBMs can take weeks to train on a regular desktop computer. In response to relatively slow training networks, there has been a push for a hardware RBM framework of a semiconductor device with programmable logic (a Field Programmable Gate Arrays). Reference [3] also points out how many of the common approaches for parallelism in neural networks to get high-performance, systems are faced with limited resolution and small networks. However, FPGAs bring the advantage of arrangement of processing elements to customize the abilities and performances of the networks.

In Reference [4], the authors go into further detail compared to the other references on Deep Belief Networks and their composition to better explain how they consume more power and high resource utilizations. As a common description delivered by most research papers, it is clear that DBNs are made of RBMs consisting of many layers of nodes. And by using RBMs, pretrained machines are obtained using Gradient based Contrastive Divergence (GCD) algorithms, gradient descent, and backpropagation for fine-tuned results. With the main computation kernel consisting of nearly hundreds of vector-matrix multiplications. Each of these multiplications can get costly when it comes to employing in hardware, as the networks experiences high silicon area and power consumption with VLSI parallel implementations

With continuous information on machine learning and improvements, it is clear that the topic is synonymous with topics of DBNs, deep Convolutional Neural Networks, RBMs, and Generative Adversarial Network algorithms. Many of these are known by their famous applications in computer vision, such as deep Convolutional Neural Networks which replicate a very close representation human perception. Boosted Deep Belief Network in reference [5] are known by their three training stages iteratively within a unified framework of loops. And convolutional deep belief networks, known for their unsupervised learning hierarchical generative models, are so closely similar to DBNs.

III. Results and Discussion

In this section, the following tables contain the energy consumption for the branch instruction that refer to the individual references, and table 2 contains the total energy consumption with reference to the following energy per instruction values:

1) *ALU = 2 pJ*
2) *Branch = Refer to Table I*
3) *Jump = 4 pJ*

4) *Memory = 100 pJ*
5) *Other = 5 pJ*

By implementing the DBN circuits from reference [1] to [4] into the code written to find a word in an input statement, it is seen how these circuits affect the total energy consumption of the code. The below table lists the required energy consumption to perform each branch prediction based on the different technologies proposed in [1-4].

Table I: Energy consumption for a branch instruction in the designs provided in [1-4].

Design	Energy Consumption For Each Branch Instruction
[1]	0.2 pJ
[2]	$0.03e^{+5}$ pJ
[3]	$6e^{+5}$ pJ
[4]	$5e^{+5}$ pJ

The following table was calculated by the MARS4.5 tool named "Instruction Statistics" to calculate the number of times a certain instruction was performed. Once the number of times the instruction was performed, multiply the energy for the specific instruction by the number of times it appeared and sum it up to get the total energy consumption. Using the input statement from the test case, and the search word "KnIgHt", the following is the total energy consumption seen in table II:

Table II: Total Energy consumption for the assembly program using designs provided in [1-4].

Design	Total Energy Consumption
[1]	110237pJ
[2]	1.51512E7pJ
[3]	3.00851E9pJ
[4]	2.50711E9pJ

IV. Conclusion

This paper has brought many important topics to light and at its best to describe the MARS program and DBNs. In summary, the assembly program written to find all the instances a certain word appears in an input statement and at what locations, provides significant workings of strings, addresses, data, and memory contents. By allocating space for the input statement and the specific searched word, it gives the ability to loop through the registers that store the individual letters of each word in the input statement, and the search word to check for capital and lowercase letters in order to find the instances of the specific word. By having a way to check for a space and loop till the next word in the input statement, and increment of the index all played an important role in having a successful program that keeps track of the indexes based on the words rather than spaces and incrementing the register value or decrementing the register value by 32 to check for capital or lower case letters. While it might take a novice familiarity of neural networks to understand DBNs, it can be seen as no more than a deep neural network, comprised of many layers each connected to the next however none of the nodes connected to each other in the same layer. It can be noticed that many of the papers that refer to Deep Belief Networks contain Restricted Boltzmann machine (RBM), which allow for their reported impressive learning abilities. DBNs are also formally known as generative graphical models. Many of these papers on DBNs circuits focus on pointing out common flaws and improvements that can lead to better more equipped machines that deal with the energy needs of DBNs. It is also shown that the total energy consumption is proportional to the energy consumption for the branch instruction. For instance, it can be seen that the total energy consumption using the energy consumption for the branch instruction from reference [1] is 110237pJ compared to any of the larger total values calculated with their larger energy consumption for the branch instruction.

V. Acknowledgment

I would like to thank all the readers for their time spent reading this paper. I also would like to thank Dr. DeMara, Mousam, and Meghana for the opportunity to do this report topic and in aiding in the accomplishment of this paper.

References

[1] H. Pourmeidani, S. Sheikhfaal, R. Zand, and R. F. DeMara, "Probabilistic Interpolation Recoder for Energy-Error-Product Efficient DBNs with p-bit Devices," IEEE Transactions on Emerging Topics in Computing, 2020.

[2] A. Roohi, S. Sheikhfaal, S. Angizi, D. Fan, and R. F. DeMara, "ApGAN: Approximate GAN for Robust Low Energy Learning from Imprecise Components," IEEE Transaction on Computer, vol. 69, no. 3, 2020.

[3] D. Le Ly and P. Chow, "High-performance reconfigurable hardware architecture for restricted boltzmann machines," IEEE Transactions on Neural Networks, vol. 21, no. 11, pp. 1780–1792, 2010.

[4] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "Vlsi implementation of deep neural network using integral stochastic computing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 10, pp. 2688–2699, 2017.

[5] P. Liu, S. Han, Z. Meng, and Y. Tong, "Facial Expression Recognition via a Boosted Deep Belief Network," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[6] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," Proceedings of the 26th Annual International Conference on Machine Learning - ICML 09, 2009.