Search Command: Memory Read Instruction and Efficiency of the Sense Amplifiers

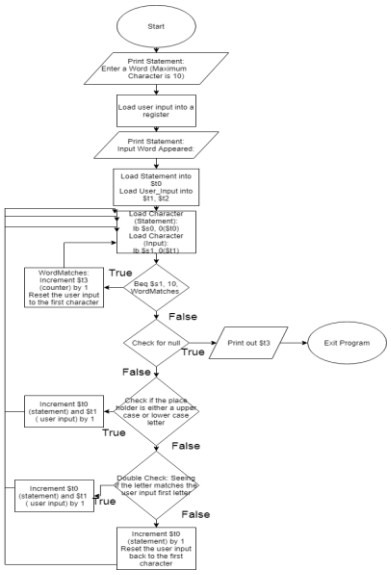
Matthew S. Smith
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*— In this paper, we will look at a code that takes users inputs and displays the frequency of the inputted word. The first part of the paper will discuss an overview of how the code works in MIPS program and a flowchart displaying the process. Then looking at certain cases of testing the code on reliability. After looking at how the code works, you will be able to understand how memory bit, sense amplifiers, and word line works. Once looking over the components of memory read instruction, some calculating was done to find out which sense amplifier would be the most efficient. Comparing all four-sense amplifier, the conclusion was that EASA was the most efficient amplifier to use for the control F code.**

Keywords— Sense amplifier, MTJ (Magnetic Tunnel Transfer), voltage headroom, SM, and bitline.

I. PROJECT DESIGN

The purpose of my code was to function has the control F command. Whenever the user typed in a word, the code should be able to run through the certain statement and tell the user how many times the word appears in the statement. In the beginning of the code, it will output a statement telling the user input a word with maximum of 10 characters. Once it receives the word, the code will store the word into a register. Then taking one character from the input word and the statement and comparing them. If a null was present, the code will increment a register used as a counter for how many word matches. If a null is not present, then the code will check if the letter is either upper case or lower case. To check if the character is upper-case or lower-case, the code took the character that was load from the register that contained the user input. The code would then add 32 to the binary number. Adding 32 will check if the input word is lower case. If the increased user input matched the character from the statement, then it would increment both the input word and statement to keep checking. If the increased input word still did not match. The code then proceeds to decrease the input word by 64. Subtracting 64 from the already incremented input word of 32 back to upper case. Also, this helps the code to see if the input is not a letter. If the characters match, the code will increment again the input word and statement. If the characters do not match, then the input word is reset to the first character. While the statement is still incremented by one. Once the statement reaches it's null, the code will then proceed to print out the register that contained the number of word matches.



FlowChart.png

Click for enlarged view

Fig.1: Flowchart of the assembly program.

```
Enter a Word (Maximum Character is 10):
KnIGhT
Input Word Appeared:6
-- program is finished running --
Enter a Word (Maximum Character is 10):
UcF
Input Word Appeared:3
-- program is finished running --
Enter a Word (Maximum Character is 10):
The
Input Word Appeared:11
-- program is finished running --
```

Fig.2: Sample outputs of assembly program.

After printing out the value, the code will then terminate, and the program will end.

When it came to test the code, I wanted the code to be able to receive any input from the user and provide the correct output. The first case I test was inputting the word Knights. However, the word would be inputted as "KnIGhT." The output of the code was 6, which is the correct number of time the word knights appears. Because the code compares upper and lower-case letters, it shouldn't matter what the case is. The second test was to make sure that the code could work for any word. For the input word, I typed "UcF." The output of the code was 3, which again was the correct number of word that were spelled UCF. The last case was to see if there was a limit on how many times the code would count word matches. The input word for case 3 was "The." The output of the code was 11, which is how many times the word "The" appeared. Because of case 3, the code can count a word a large amount of times.

II. Memory Bit-Cells

When looking at the hardware of the processor and the motherboard. The "1's" and "0's" are the measurement of the voltage on that line. The lines that transmit the binary representation are called "bitline." Once the SM (Sense Margin) detects the record voltage, it then begins to compare with the reference voltage. After the SM records the voltage, the SA (Sense Amplifier) takes the low signal and amplifies it [1], [4]. With the signal be amplified, the signal can then be processed by the logic controls like XOR, S and D latches, and Differential amplifier control circuits [2]. The SA is crucial for the reading of data from the memory. Looking at the SA, there are three components that make up it. First looking at the Transition Gates (TG), the TG can allow and block signal levels [6]. The next component is the Magnetic Tunnel Junction (MTJ), which is made up of two magnetic layers that are separated by a dielectric barrier [4]. The MTJ increase the SM and the voltage headroom if the resistance of the MTJ is reduced. With the decrease in the resistance, the error of the system is also reduced. Which in turn means that the error rate is proportional to the resistance of the MTJ [1], [2]. The last component is NMOS transistors. The NMOS is used to control sensing current flowing through the MUX and block BL voltage [3].

III. Results and Discussion

After testing the final code, using the provided energy listed below to see how much energy is used. Here are the known energies of each step in the code:

1) ALU = 1fJ
2) Branch = 3fJ
3) Jump = 2 fJ
4) Memory = Read Energy (Table 1 values) + Write Energy (50 fJ)
5) Other = 5fJ

For the read memory, four different SA will be used to see which amplifier will be the most efficient. Here is the energy of each amplifier with the read memory instruction:

Table 1

Table I: Energy consumption for a single bit-cell read operation in the designs provided in [1-3].

Design	Energy Consumption For Each Bit-cell's Read Operation
EASA [1]	0.23 fJ
VISA [1]	1.86 fJ
PWSA [2]	36.0 fJ
BVSC [3]	195.5 fJ

EASA, VISA, PWSA, and BVSC are four different types of sense amplifiers. Each amplifier contributes to different aspect of the hardware. For example, EASA is an amplifier that is energy aware and VISA is variation immune.

Using MARS software tool Statistics, the total energy for every instruction type. Here is the total energy of the code with each amplifier being tested:

Table 2

Table II: Total Energy consumption for the assembly program using designs provided in [1-3].

Design	Total Energy Consumption
EASA [1]	1.35157E-11
VISA [1]	1.56163E-11
PWSA [2]	5.96113E-11
BVSC [3]	2.65154E-10

IV. Conclusion

To conclude, looking at the total energy from each of the four amplifiers. The most efficient SA was the EASA. However, the EASA is always going to be the most efficient when it comes to the total energy. When looking at the provided energies for each SM, the EASA has the minimums amount of energy. Again, each SM is designed to approach different aspects of the performance of the hardware.

Project 3 provides an in-depth analysis of the circuitry components and challenges of designing memory read circuits. Also, project 3 shows the structure of how the control F command works in assembly language.

References

[1] S. Salehi and R. F. DeMara, "Process variation immune and energy aware sense amplifiers for resistive non-volatile memories," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, 2017, pp. 1-4.

[2] H. Lee *et al*., "Design of a Fast and Low-Power Sense Amplifier and Writing Circuit for High-Speed MRAM," in *IEEE Transactions on Magnetics*, vol. 51, no. 5, pp. 1-7, May 2015.

[3] F. Ren, H. Park, R. Dorrance, Y. Toriyama, C. K. K. Yang and D. Marković, "A body-voltage-sensing-based short pulse reading circuit for spin-torque transfer RAMs (STT-RAMs)," Thirteenth International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, 2012, pp. 275-282.

[4] J.M. Slaughter, E.Y. Chen, R. Whig, B.N. Engel, J. Janesky, and S. Tehrani, "Magnetic Tunnel Junction Materials for Electronic Apllications," Materials for Magnetic Memory: Overview, June 2000, JOM-e

[5] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, Doug Burger, "Architecture Support for Disciplined Approximate Programming," Dual-Voltage SRAM Array, ASPLOS XVII Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems, pages 301-312

[6] "What is a Transmission Gate (Analog Switch)," Maxim Integrated, June 10, 2008,