# MIPS Assembly Program ALU Instructions Employing Multiple Look-Up Table (LUT) Designs

**Jadyn Lalich**
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*— Look-up tables (LUTs) can be used to implement ALU instructions and depending on their design the total energy consumption of a program will vary. Their different designs and LUTs in general are discussed at length in relation to the total energy of a program designed to search for a key word in a string. This program is implemented using MIPS Assembly on MARS where a user inputs their search word and the output of the program is the total number of occurrences of that word. The instruction statistics were collected and then used in calculating the total energy consumption to determine which LUT design consumed the least energy. After reviewing the different aspects of LUT designs such as magnetic tunnel junction (MTJ), CMOS, spintronic, and SRAM-based FPGA devices it was calculated that design [1] had the lowest total energy consumption of 264.076 pJ.**

***Keywords—SRAM, Non-Volatile Memory, FPGA, Lookup Table, MOSFET, NMOS, PMOS, and Magnetic Tunnel Junction.***

## I. Project Design

The entirety of this project uses basic loops and syscalls as the building blocks of its functionality. The objective of the code is to search for a word (less than 10 characters) entered by the user. The program then searches for that word in the hardcoded string and outputs the number of occurrences. The caveat being that regardless of if the word is a mix of upper and lower case letters it must still find the word. To achieve this functionality the word and string must be compared Byte by Byte. The base addresses of the string and word are loaded into two separate registers so they can be manipulated to increment or decrement which character is being compared in either. The outer most loop is where different criteria is checked and then branches to sub loops to execute what should be updated. The outer loop loads the Bytes based on the manipulated base addresses and then checks to see if the string is at its end, if the search word is at its end, or if the character in the word matches the string.

To check for the end of the string the Byte of the manipulated base address is loaded and compared to the NULL character, and if they match the program branches to exit. Next seeing if the search word is at the end of its characters the outer loop has a branch statement checking for the number 10 which is ASCII for line feed/new line. If the search word is at its end it branches to the Match loop where the base address of the word is reset to the start and the found word counter is incremented by one. If neither of those branches have been taken, the program then

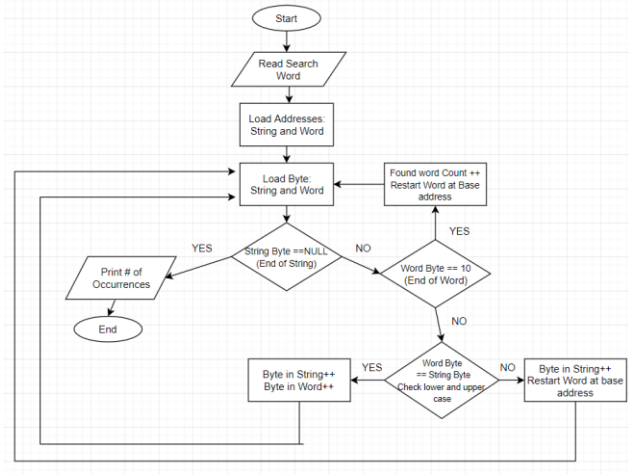


Fig.1: Flowchart of the assembly program.

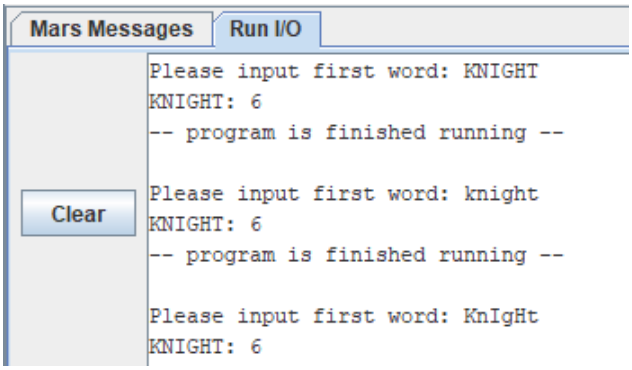


Fig.2: Sample outputs of the assembly program.

checks the character in the word vs the character in the string. If there is no match then the search character is incremented by 32 checked and decremented by 64 if either of those were a match the program branches to the match character loop where both the string and word base addresses are incremented by one. The reasoning behind the 32 and -64 are to run through both the upper and lowercase of the current character while checking for a match. Lastly if there were no matches the search word is reset to the start character and the string base address in incremented by one. It should be noted that each of these sub loops jump back to the outer loop until the end of the string is met.

Upon exiting the loops, the number of occurrences of the search word is printed as the output and the program completes. There are numerous search words to check the functionality of this code but three have been chosen which support that the program is working properly. The inputs tested are KNIGHT, knight, and KnIgHt. The first input test to see if the program can handle only an uppercase search word, the second input demonstrates only lowercase, and finally it is tested to see if we will get the same output when there is a mix of upper and lowercase letters. Based on the hardcoded string the word "knight" no matter its format occurs six times so each of those test cases should produce the number six. As shown in Figure 2 it can be concluded the program is working as desired.

## II. Look-Up Table Circuit

The ALU instruction designs for the program implemented in this paper take into account the use of Look-up Tables (LUT) over the use of two input Boolean gates. The main notable differences are efficiency, power, and size. A Spin-MTJ-based LUT is energy efficient and can implement complex logic fast [2]. They have programmable implementation of logic functions making reconfiguration manageable. Whereas a two input logic gate will have more gate delays and require more energy depending on the complexity. Something that may need a rather large amount of two input Boolean gates can be implemented in only a few steps with a LUT. With non-volatile memory different logic can be implemented and stay on the device rather than using multiple two input Boolean gates. Once a circuit is created with gates there is no reprograming its function. A LUT can compute all these Boolean gates using a truth table although as the complexity of the logic increases so does the combinational delay [5] proving that these tables are not perfect but they still are very efficient with respect the two input gates.

A notable issue with LUTs is its power dissipation. In the small technology scaling world today there is leakage effecting the power dissipation of these designs. In CMOS technology the source and the drain of the MOSFET are remarkably close together in this scale that even without the presence of a tunnel there is some current leakage dissipating the power. That is part of the reason MJT designed devices are preferred over CMOS [7]. SRAM-based FPGA design struggles to correct this issue and hybrid methods are being proposed [4] to improve the efficiency of power. An exciting aspect of MJT devices though is that they only need a small voltage that is large enough to bias the device which makes the power needed minimal [3]. Another spin on the MJT emerging technology is spintronic devices since they also offer a solution to the power, volatility, and a few other critical issues in their SRAM predecessor as well. [6].

Area of a LUT has both its pros and cons with respect to functionality and depends on design. Since MOSFETs are the main component of most LUTs, many of their quarks effect the functionality. Real world circuitry is not perfect and the voltage levels produced by an NMOS alone run the risk of having a weak 1 or 0 which are essential for logic gate computations. A way to try and fix this error is to compromise area and add a PMOS to counter balance and boost the voltage [1]. In general there are many areas of research for LUT designs as there functionality poses new possibilities in high speed computing.

## III. Results and Discussion

The total energy consumption instruction statistics of the program were obtained using the MARS4.5 Instruction Statistics tool. The calculations made below use the assumption that all the designs have the same energy consumption per instruction for everything except the ALU instructions. The ALU instruction values vary based on the different look up table designs and the values are provided in Table I. The other energy consumption per instruction values are Branch = 3fJ, Jump = 2fJ, Memory = 200fJ, and Other = 5fJ. The program design has a dynamic instruction count broken down into these sections: $ALU_{inst.}$= 3688, $Jump_{inst.}$= 632, $Branch_{inst.}$= 3071, $Memory_{inst.}$= 1266, and $Other_{inst.}$= 6. To calculate the total energy consumption the following formula was used:

$$Energy\_Consumption=(ALU*ALU_{inst.})+(Jump*Jump_{inst.})+(Branch*Branch_{inst.})+(Memory*Memory_{inst.})+(Other*\ Other_{inst.})$$

It should be noted that the total energy consumption calculations for each ALU LUT design are shown in Table II.

Table I: Energy consumption for a single ALU Instruction in the designs provided in [1-4].

| Design | Energy Consumption For Each ALU Instruction |
|---|---|
| [1] | 0.1 fJ |
| [2] | 1.2 fJ |
| [3] | 0.6 fJ |
| [4] | 0.25 fJ |

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

| Design | Total Energy Consumption |
|---|---|
| [1] | 264.076 pJ |
| [2] | 268.133 pJ |
| [3] | 265.920 pJ |
| [4] | 264.629 pJ |

## IV. Conclusion

The prospect of new technologies and even small improvements on old technologies leave their impact and inspire people to think of things in different ways. Through learning about ALU instruction implementation using LUTs many different technical designs and ideas were brought into light. Such as magnetic tunnel junction technologies, Spintronics, SRAM-based FPGA devices, CMOS, non-volatile memory, and leakage current. All of which are being discussed and researched around the globe. As for some of the technologies and designs tested for their total energy consumptions design [1] and [4] were the most similar with design [1] calculated to consume the least amount of energy for the program. Observing the different energy consumptions per design in Table II also provides insight into just how different one design is relative to another.

## References


[1] A. Alzahrani and R. F. DeMara, "Process variation immunity of alternative 16nm HK/MG-based FPGA logic blocks," *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Fort Collins, CO, 2015, pp. 1-4.

[2] W. Zhao, E. Belhaire, C. Chappert, F. Jacquet, P. Mazoyer, "New non-volatile logic based on spin-MTJ," *physica status solidi (a)*, vol. 205, no. 6, pp. 1373-7, 2008.

[3] D. Suzuki, M. Natsui and T. Hanyu, "Area-efficient LUT circuit design based on asymmetry of MTJ's current switching for a nonvolatile FPGA," 2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), Boise, ID, 2012, pp. 334-337.

[4] Y. Zhou, S. Thekkel and S. Bhunia, "Low power FPGA design using hybrid CMOS-NEMS approach," *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, Portland, OR, 2007, pp. 14-19.

[5] R. Alhalabi, G. Di Pendina, I. Prejbeanu, and E. Nowak, "High Speed and High-Area Efficiency Non-Volatile Look-Up Table Design Based on Magnetic Tunnel Junction." *2017 17Th Non-Volatile Memory Technology Symposium (NVMTS), Non-Volatile Memory Technology Symposium (NVMTS), 2017 17Th*, 2017, p.1.

[6] Ramtin Zand, Arman Roohi, Ronald F. DeMara, "Energy-Efficient and Process-Variation-Resilient Write Circuit Schemes for Spin Hall Effect MRAM Device", *Very Large Scale Integration (VLSI) Systems IEEE Transactions on*, vol. 25, pp. 2394-2401, 2017, ISSN 1063-8210.

[7] D. Suzuki, M. Natsui, T. Endoh, H. Ohno, and T. Hanyu, "Design of a Compact Nonvolatile Four-Input Logic Element Using a Magnetic Tunnel Junction and Metal-Oxide-Semiconductor Hybrid Structure." *Japanese Journal Of Applied Physics* 51, no. 4