A Case-Insensitive Search Algorithm Implemented Using MIPS Assembly

Christopher Clifford
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*— This paper outlines the construction of a CMOS Case-Insensitive search algorithm implemented with MIPS assembly code to index occurrences of a keyword in a user defined phrase. The objective of this paper is to evaluate how technologies other than conventional CMOS design could potentially lower power consumption in this design. The fundamental designs that will be evaluated are 1, 2, 3. To test the performance of these designs, a MIPS assembly program was created that contains a search algorithm which looks for a keyword within a user provided phrase. The inputs to the program are a user keyword – of maximum 10 ASCII characters – and a user phrase to search for this keyword. The program outputs the absolute location of these occurrences within the user phrase. The case insensitive search algorithm was studied for dynamic instruction count and most prevalent instruction type to estimate energy consumption in all the proposed technologies. We found that for this search algorithm, the most efficient technology to use was the MSA-PCSA Spin-Hall Effect Magnetic RAM with an estimated energy consumption of 25867.15 fj.**

***Keywords*—**

Non-volatile Memory

Search algorithm

Leakage energy

Power/Energy Consumption

Circuit Complexity (number of transistors and channel of transistors)

Scalability

General Purpose Registers (GPR)

Complimentary Metal Oxide Semiconductor (CMOS)

Spin-transfer torque magnetic random access memory (STT-MRAM)

I. Introduction

This section should have at least 2 paragraphs. One for describing your coding method and one two explain the test inputs, as well as your code outputs.

A. Project Design

This MIPS algorithm was implemented by first determining the size of the keyword that would be analyzed. After the number of characters in the string is determined, the program

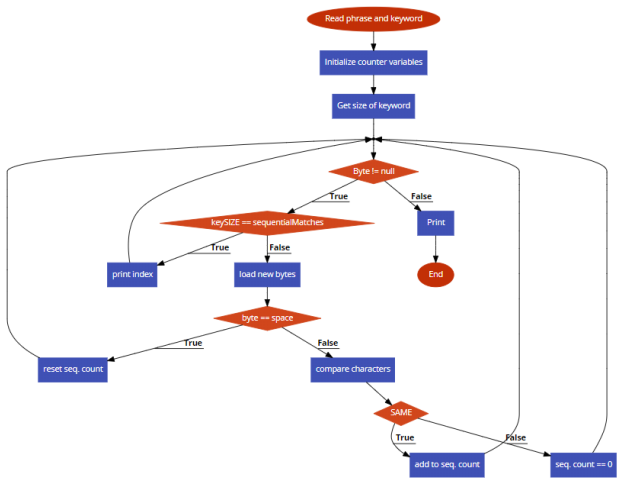

Fig.1: Flowchart of the assembly program.

Test Case 1

```
All of the knights of the roundtable were knights.


# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, &): knight
# Indexes of the matches found: # Please type in your input statement:

4, 9, Number of times the word "" was found in the input statement: knight
" was found in the input statement: 2
-- program is finished running --
```

Test Case 2

```
"Go UCF Kni ghts, charge on knights!"


# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, &): knight
# Indexes of the matches found: # Please type in your input statement:

7, Number of times the word "" was found in the input statement: knight
" was found in the input statement: 1
-- program is finished running --
```

Test Case 3

```
"The color of the KnIgHtS are black and gold. Go knights!"


# Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, &): knight
# Indexes of the matches found: # Please type in your input statement:

5, 11, Number of times the word "" was found in the input statement: knight
" was found in the input statement: 2
-- program is finished running --
```

Fig.2: Sample outputs of assembly program.

iterates each character and increments a counter variable when both the bytes being compared match. If the characters do not match, the sequential counter is reset to zero. When multiple comparisons match in a row, the sequential counter will continue to increase. Once the sequential counter reaches the size of the keyword, the occurrences is printed, and the sequential counter is reset to zero.

B. *Test Cases*

To validate the correctness of this program, three test input cases were devised. The goal of each of these test inputs was to test this program against 1) input where the keyword was contained within a larger word 2) input where a keyword was spliced by a ' ' character 3) input where the keyword contained both upper and lower case characters.

To test these scenarios the following 3 input sentences were chosen:

Key word: "Knight"

1) "All of the knights of the roundtable were knights."

2) "Go UCF Kni ghts, charge on knights!"

3) "The color of the KnIgHtS are black and gold. Go knights!"

The output of this program follows a specific format. First the program asks the user to input the string to be analyzed. Then the program asks the user to type a keyword (up to 10 characters) that they are looking for in the phrase. The output of the program is the number of occurrences the input was found in the phrase as well as their corresponding indexes within the phrase.

 # Please type in your input statement:

UCF, its athletic program, and the university's alumni and sports fans are sometimes jointly referred to as the UCF Nation, and are represented by the mascot Knightro. The Knight was chosen as the university mascot in 1970 by student election. The Knights of Pegasus was a submission put forth by students, staff, and faculty, who wished to replace UCF's original mascot, the Citronaut, which was a mix between an orange and an astronaut. The Knights were also chosen over Vincent the Vulture, which was a popular unofficial mascot among students at the time. In 1994, Knightro debuted as the Knights official athletic mascot.

 # 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: 27, 29, 42, 75, 96, 100.

Figure 3 Sample output of search algorithm

II. Memory Bit-Cells

Memory cells are a fundamental piece of computer hardware that store either a binary 1 or 0. A massive number of these cells are combined to store larger quantities of information necessary for computation. New alternatives to conventional non-volatile binary memory storage has become available in recent years. These technologies include Spin-transfer torque magnetic random-access memory (STT-MRAM), Spin-Hall Effect Magnetic RAM (SHE-MRAM), Phase Change Memory (PCM), and Resistive RAM (RRAM) [1-2]. One of the challenges of these new memory technologies is reliability problems caused by process variations [1, 5]. More accurate sense amplifiers are needed to produce more confident memory systems [5].

In conventional CMOS memory with binary memory, either a 1 (high voltage) or 0 (low voltage) is stored in a capacitor. When accessed by the word line transistor, the data from the capacitors is transferred to the bit line. These values reside within the cells through capacitors and must be charged in order to combat parasitic capacitance that will drain the charge and could potentially corrupt the data.

Flash memory is an EEPROM memory that can store data without power, i.e. nonvolatile. The name is coined because tunneling electrons can change the charge on the gates rapidly. Flash memory can be either NOR or NAND logic. Commonly used devices such as flash drives and cell phone memory. In NOR flash each cell is directly connected to the bit line. The memory is called NOR because when the cell word line is brought to high voltage, the bit line is pulled low by a transistor.

While there are many memory devices in research and available on the market, they all serve similar functions have commonalities. It is important however to distinguish the benefits of Flash memory compared to hard drives. Flash memory has a much higher shock tolerance and is more durable than hard drives however hard drives have a higher overwrite capability.

III. Results and Discussion

Here we compare energy consumption between the 4 studied designs. Determined by the MIPS instruction counter tool, our test string had a dynamic instruction count of 15918 instructions. Using conventional energy consumption values from typical CMOS devices, we can calculate – using the below values – the estimated energy consumption of our memory using the different designs.

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
MSA-PCSA [1]	0.35 fJ
MSA-SPCSA [1]	1.15 fJ
Dynamic Latched Comparator[2]	37.6 fJ
STT-MRAM[3]	0.51 fJ

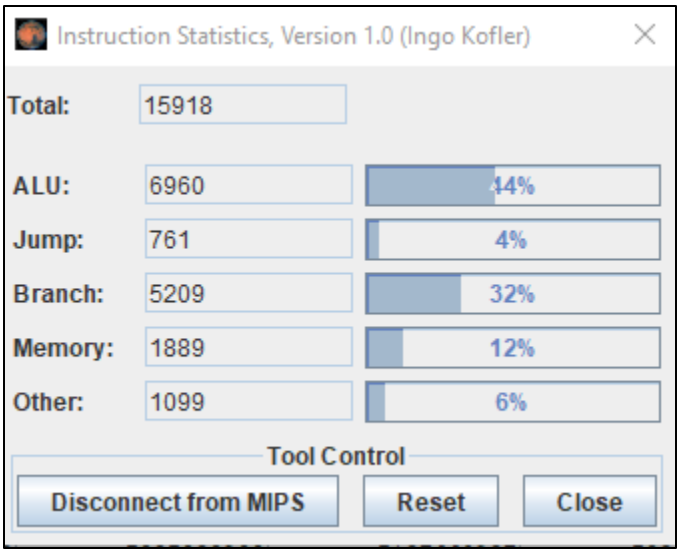

Figure 4 Instruction Statistics

Figure # Instruction count breakdown of sample input

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

Since we are only concerned with memory, the cumulative energy consumption of the ALU, Branch, Jump, and Other have been summed into one quantity that equals 25156 fj. The energy consumptions of the above designs for memory only are: MSA-PCSA 711.15 fj, MSA-SPCSA 2222.35 fj, Dynamic Latched Comparator 71076.4 fj and STT-MRAM 1013.39 fj.

Energy consumption data provided in references [1-3] was used to calculate the total energy consumption of the program. The below table lists the required energy consumption to perform each memory write operation using the different circuits proposed in [1-3].

Table 1 Energy Consumption for read operations
Table 2 Total Energy Consumption

IV. Conclusion

Through this project, it is clear there are many methods used to store data and new technologies have made vast improvements in the power and efficiency of our computers. Technologies such as flash memory and spintronics, while not entirely reliable due to process variations, show great potential with further research to make our computers more efficient. This project also stressed the importance of coding algorithms. Through techniques such as rolling loops it is possible to

significantly reduce execution time and minimize the total energy consumption of a design. The best design analyzed herein was the MSA-PCSA Spin-Hall Effect Magnetic RAM consuming 25867.15 fj of energy.

References

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

Design	Total Energy Consumption
MSA-PCSA [1]	25867.15 fj
MSA-SPCSA [1]	27378.35 fj
Dynamic Latched Comparator[2]	96232.4 fj
STT-MRAM[3]	26169.39

[1] S. Salehi, N. Khoshavi, R. Zand, and R. F. DeMara, "Self-Organized Sub-bank SHE-MRAM-based LLC: an Energy-Efficient and Variation-Immune Read and Write Architecture," Integration, The VLSI Journal, accepted and in-press.

[2] HeungJun Jeon and Yong-Bin Kim. 2010. A low-offset high-speed double-tail dual-rail dynamic latched comparator. In Proceedings of the 20th symposium on Great lakes symposium on VLSI (GLSVLSI '10). ACM, New York, NY, USA, 45-48.

[3] R. Bishnoi, F. Oboril, M. Ebrahimi and M. B. Tahoori, "Self-Timed Read and Write Operations in STT-MRAM," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 5, pp. 1783-1793, May 2016.

[4] C. Heegard, "On the Capacity of Computer Memory with Defects sym m e tric," IEEE Trans. Inf. Theory, vol. I, no. c, 1983.

[5] C.-Y. Lu, K.-Y. Hsieh, and R. Liu, "Future challenges of flash memory technologies," 2008.

[6] Pavan, Paolo, Roberto Bez, Piero Olivo, and Enrico Zanoni. "Flash memory cells-an overview." Proceedings of the IEEE 85, no. 8 (1997): 1248-1271.