Evaluating Energy Consumption in Memory Hardware with Respect to Memory Write Operations

Andre Samaroo
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL 32816-2362

***Abstract*—This report, "Evaluating Energy Consumption in Memory Hardware with Respect to Memory Write Operations," looks at the energy consumption of different memory systems using phase-change memory and optimized spin torque transfer devices. In order to evaluate the best model for reducing energy leakage, this report looks at the energy use to write data into memory for PCM, optimized STT, and STT with a BGIM cell. The program used to test the memory energy consumption searches for a user defined key word in a user defined phrase. After finding the amount of energy used in the memory write operations of each model and the number of dynamic instructions count for the experimental program, it is possible to calculate the total energy used for each of the hardware models. This report finds that the BGIM cell offers the best energy reduction of all the models consuming only 171898.88 fJ, or 171.89pJ in total. The BGIM cell also offers faster memory access and use bidirectionality to reduce the hardware used in previous models.**

Keywords—Non-Volitile Memory, Memory Write, Magnetic Tunneling Junction, Phase-Change Memory, Energy Leakage, Power Gating

I. Introduction

The program for this project was to take two inputs from the user: a phrase and a key word. The program searches for the occurrences of the key word in the given phrase and reports the number of occurrences and the indices at which they occurr to the console in formatted output. The program should match the keyword with any occurrence regardless of case or if the key word occurs within a larger word. The code for the program consists of a block of code to scan input from the user, a block to serve as the main body of the code, three blocks that execute instructions based on the conditions tested in the main block, and the final block of code should print out the formatted output.

The first inputs were the search phrase, "knight," and the search word, "knight." These represented a simple case that would test the basic functionality of the program. The corresponding output was 1 occurrence at index 1 as shown in Figure 2. The second test inputs were the phrase and word given in the project description in order to test the basic test case given to the students. The output for this case was 6 occurrences as indices 27, 29, 42, 75, 96, and 100. The third test case used the test phrase, "Knight knight KnIgHt kNiGhT," and the test word, "knight" in order to test the case insensitivity of the program to ensure it would run without regard for the case of the input strings. The output should be 4 occurrences at indices 1, 2, 3, and 4.

A. Project Design

The project code was made of three parts: scanning input, the main body of the code, and printing the output.

The first part uses syscall to print the prompt to ask the user to "Please type in your input statement:". The syscall in this program can take a phrase up to 800 characters in allocated memory. It then prints the prompt to ask "Please type in a word (up to 10 characters) that you are looking for (e.g. Knight or KnIGhT, knight, …):" which is scanned into allocated memory space for the word. This is followed by the initialization of the address registers at the address of the phrase in s1, word in s0, and the index array in s2. Also, the occurrence counter, index counter, and test variable are initialized.

The second part is the main body of the code. This block tests the various conditions of the code to perform the appropriate actions. The t0 and t1 registers are loaded with s0 and s1 respectively. Each register should contain one byte which is one letter from the key word and one letter from the search phrase.

The program first checks for the enter character. If t0 equals 10, the program increases the occurrence counter, t3, stores the index number in the index array, increment the address of the array by 4 for the next word, and loads the address of the word into the address register again to reset the register and search for the next occurrence of the key word. then it jumps back to the main block. If t1 equals 10, then the program should jump to the printing block.

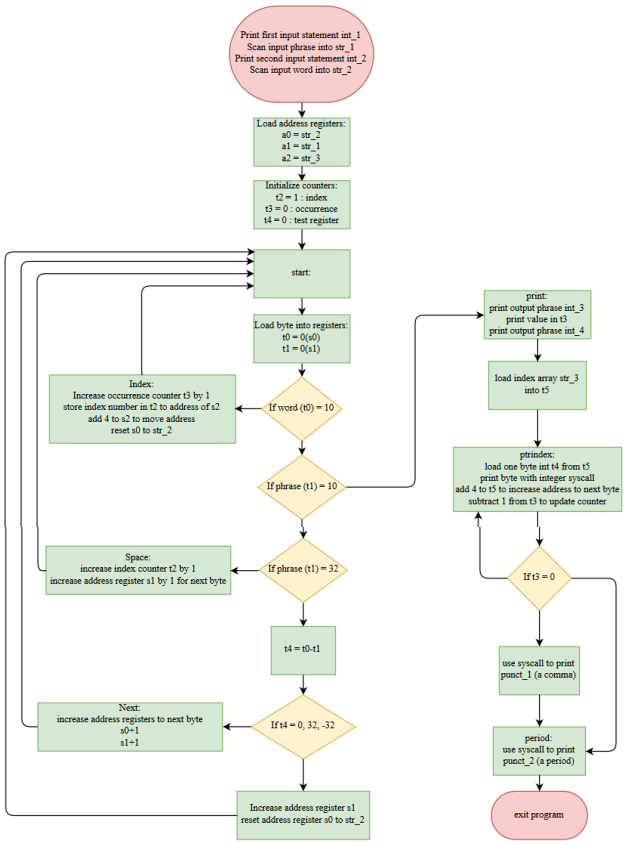
This program also looks for spaces in order to count the indices. When t1 equals 32, there is a space in the phrase, so the program jumps to the index counter, increases the index number in t2, increases the address register a1, and jumps back to the main block.

Else, the program should subtract the values in t0 and t1, then store the value in t4 to test for equality. If the values are equal, then the t4 register is zero. If t0 is a capital letter, then t4 is -32. If t1 is a capital letter, then t4 is 32. In any of these cases, the program should jump to the "next" label to compare the rest of

the characters. The address registers a0 and a1 are increased by one to move to the next bit and then return to the main block.

The last block is the printing block. Here, the output message, “Number of times the word was found in the input statement:” is printed, then the number of occurrences in the t3 register is printed. Then, the index message, “Indexes of the matches found:” is printed. In order to print all elements in the index array, a loop is used to print a byte from the array, print a comma, and then increment the address of the index array to print the next byte. When the byte equals zero, the program branches from the loop and prints a period. Once the output is printed, the program closes.

Fig.1: Flowchart of the assembly program.



B. Test Cases

The three test cases used on this program include simple inputs, sample inputs, and inputs with different capitalizations (, a mix of capital and lowercase letters). The first test case used the test phrase, “knight,” and the key word, “knight.” This case tested the program to see if it would run on its most simple level of recognizing the same word. The output of the program, as shown in Figure 2a, is 1 occurrence at index 1. The second case tested the sample output that was given with the project assignment. The test phrase was, “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.” and the key word was, “KnIgHt.” This case tested an input phrase that was longer, including full sentences, and the case insensitivity of the program for the key word. The program output, as shown in Figure 2b, counted 6 occurrences as indices, 27, 29, 42, 75, 96, and 100. The third test case used the test phrase, “Knight knight KnIgHt kNiGhT,” and the test word, “knight.” This case tested the case insensitivity of the test phrase and a simple key word. The output of the program, as shown in Figure 2c, counted 4 occurrences at indices 1, 2, 3, and 4.

Fig.2: Sample outputs of the assembly program.

a) Case 1

```
# Please type in your input statement:

knight

# Please type in a word (up to 10 characters) that you are looking

knight

# Number of times the word was found in the input statement: 1
# Indexes of the matches found: 1.
-- program is finished running --
```

b) Case 2

```
# Please type in your input statement:

UCF, its athletic program, and the university's alumni and sports

# Please type in a word (up to 10 characters) that you are lookin

KnIgHt

# Number of times the word was found in the input statement: 6
# Indexes of the matches found: 27, 30, 43, 79, 101, 105.
-- program is finished running --
```

c) Case 3

```
# Please type in your input statement:

Knight knight KnIgHt kNiGhT

# Please type in a word (up to 10 characters) that you are lookin

knight

# Number of times the word was found in the input statement: 4
# Indexes of the matches found: 1, 2, 3, 4.
-- program is finished running --
```

II. Memory Bit-Cells

The hardware for memory access has three states: standby, read, and write [6]. The bit and word lines help determine which state the hardware should execute. In the case of write, the word

line enables the transistors and allows the data in the bit line to be written into memory [6]. In the case of read, the data from the bit line goes to output instead of the memory cell. Memory access is one of the most energy intensive operations a device can perform, especially with the continual efforts of tech companies to increase memory capacity. From the different models that were developed to reduce energy leakage, phase-change memory (PCM) and spin-torque transfer technology (STT) present the best options for development [3]. PCM uses a phase-change material which can take on a crystalline or amorphous state depending on the temperature. These states correspond to logical "1" or "0" [3]. The bit and word lines of PCM function the same as with previous technology. The draw-back of PCM is that the temperature sensitive components make it less durable than STT [1].

STT is based on the magnetic orientation of the materials used in its circuits [1]. STT devices use magnetic tunneling junctions (MTJ). An MTJ consists of a layer of fixed magnetic material on top of an insulating layer on top of a free magnetic layer which rest on a conductor [2]. Depending on the flow of the current, the magnetic orientation will be parallel or anti-parallel [2]. This way, the data is stored in the magnetic resistance instead of the voltage of the capacitor [2]. This also means that the data storage is non-volatile, or that it won't degrade over time [1]. In addition to the bit and word line, STT devices also have a sensor line to detect the magnetic orientation [1]. While STT offers a faster and more durable than PCM, it requires more current to change the magnetic orientation, so methods for optimizing STT have been developed [2].

One solution is to use power gating (PG) to reduce energy leakage during idle operations, or stand by mode [6]. In 2-macro architecture, memory is transferred between RAM and Flash [6]. Between memory transfers, there is idle time where the energy is still flowing and, therefore, leaking [6]. PG shuts the power off when the device in in stand by and turns it back on again when needed [6]. Another optimization design for STT devices includes the Bit-Grained Instant-on Memory (BGIM) cell. BGIM gets rid of the sensor line and instead uses bidirectionality as a way of determining the state of the memory device [4]. It also gets rid of the 2-macro design which gets rid of the data transfers between structures [4]. This solves the energy leakage issue from the previous model and no additional energy is needed for the sensor line.

III. Results and Discussion

In this section, the energy consumption of the program is calculated using the below energy consumption per instruction values:

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

Using inputs from test case 2, the total energy of the program was calculated. The total ALU instructions was 5274. The total number of branch instructions was 642. The total number of jump instructions was 3431. The total number of memory instructions was 1288. The total number of other instructions was 27. So, using these numbers and the energy values from Table I, the values in Table II were calculated.

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

Design	Energy Consumption For Each ALU Instruction
[1]	121.51 fJ
SHE-iMTJ [2]	189.7 fJ
SHE-pMTJ [2]	252.4 fJ
[3]	836.2 fJ

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

Design	Total Energy Consumption
[1]	171989.88 fJ
SHE-iMTJ [2]	259818.6 fJ
SHE-pMTJ [2]	340576.2 fJ
[3]	1092510.6 fJ

IV. Conclusion

With the increases in memory capacity, it is important now more than ever to reduce the amount of energy leakage from memory write operations. Two models of interest are PCM and STT. PCM offers an energy saving alternative but at the cost of durability and speed. STT is a model based on magnetic orientation. It requires a high current to change the magnetic orientation, but it offers non-volatile memory, high speed memory access, and reduces the energy consumption of memory devices. So, to further optimize the STT device, power gating and BGIM cells can be used to further reduce the leakage. Topics researched and explained in this report include the circuitry involved in memory systems, magnetic methods for post-CMOS devices, using memory write as a metric for energy consumption comparisons, the use of search strings in assembly code, and memory allocation in the .data field of assembly code. BGIM cell offer the most savings as evidenced in the results by the lowest total energy of the program from Table II and the lowest energy per instruction in Table I.

References

[1] E. Kultursay et al., "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," 2013 IEEE International Symposium on Performance Analysis of Systems and Software, pp. 256-267, Apr 2013.

[2] P. Chiu et al., "Low Store Energy, Low VDDmin, 8T2R Nonvolatile Latch and SRAM With Vertical-Stacked Resistive Memory (Memristor) Devices for Low Power Mobile Applications," in IEEE Journal of Solid-State Circuits, vol. 47, no. 6, pp. 1483-1496, June 2012.

[3] P. Zhou, B. Zhao, J. Yang, Y. Zhang, “A durable and energy efficient main memory using phase change memory technology,” in ACM SIGARCH Computer Architecture News, vol. 37, Issue 3, pp. 14-23, June 2009.

[4] S. Salehi and R. F. DeMara, "BGIM: Bit-Grained Instant-on Memory Cell for Sleep Power Critical Mobile Applications," 2018 IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA, 2018, pp. 342-345.

[5] S. Singh, and V. Lakhmani, “Read and Write Stability of 6T SRAM,” in International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Volume 3, Issue 5, May 2014.

[6] W. Kang, W. Lv, Y. Zhang and W. Zhao, "Low Store Power High-Speed High-Density Nonvolatile SRAM Design with Spin Hall Effect-Driven Magnetic Tunnel Junctions," in IEEE Transactions on Nanotechnology, vol. 16, no. 1, pp. 148-154, Jan. 2017.