

ערעור רטוב 2 – חלק 1

מגישים: רועי מרום ותואם אלהרר

על מנת לעבור את *runtime test* החלפנו את הלולאה *count_1_bit_loop* בפונקציה *hemming_weight*. אשר בודקת כמה ביטים דולקים יש ברגיסטר %r9, בפקודה *popcnt*. (שמנו את החלק אשר החלפנו בהערה בקוד).

הקוד לפני השינוי:

```
102 # parameters: rdi = unsigned long* codeword ; rsi = unsigned long len #
103 hamming_weight:
104     # callee convention -PROLOGUE #
105     pushq %rbp
106     movq %rsp, %rbp
107
108     xor %r8, %r8 # r8 is the loop index
109     xor %rax, %rax # 1-bit counter
110     memory_to_register_loop:
111         cmp %rsi, %r8
112         je finish
113         movq (%rdi, %r8, 8), %r9 # r9 is 8 byte section of codeword
114         count_1_bits_loop:
115             cmp $0, %r9
116             je outer_counter
117             shr $1, %r9 # the bit is now in the carry flag
118             jnc skip_add
119             inc %rax
120             skip_add: jmp count_1_bits_loop
121         outer_counter:
122             inc %r8
123             jmp memory_to_register_loop
124
125     finish:
126     # EPILOGUE #
127     movq %rbp, %rsp
128     popq %rbp
129     ret
```

הקוד לאחר השינוי:

```
102 # parameters: rdi = unsigned long* codeword ; rsi = unsigned long len #
103 hamming_weight:
104     # callee convention -PROLOGUE #
105     pushq %rbp
106     movq %rsp, %rbp
107
108     xor %r8, %r8 # r8 is the loop index
109     xor %rax, %rax # 1-bit counter
110     memory_to_register_loop:
111         cmp %rsi, %r8
112         je finish
113         movq (%rdi, %r8, 8), %r9 # r9 is 8 byte section of codeword
114         popcnt %r9, %rdx # THE CHANGE - using POPCNT to count the number of 1 bits.
115         add %rdx, %rax
116         # count_1_bits_loop:
117         #     cmp $0, %r9
118         #     je outer_counter
119         #     shr $1, %r9 # the bit is now in the carry flag
120         #     jnc skip_add
121         #     inc %rax
122         #     skip_add: jmp count_1_bits_loop
123         outer_counter:
124         inc %r8
125         jmp memory_to_register_loop
126
127     finish:
128     # EPILOGUE #
129     movq %rbp, %rsp
130     popq %rbp
131     ret
```