Eric Watson

ECE335 Homework 3

To examine the differences in performance when a series of if-then-else tests are converted into a switch-case configuration, this assignment required us to compare execution times of two routines that perform the same task, but implement them differently. We were required to write a program that generates a source array, populates the array with random integers from 0 to 999, and then counts the number of values that fall in the range of 0-199, 200-399, 400-599, 600-799, and 800-999. In order to count the numbers in each of these ranges, we implemented both a switch statement and a sequence of if-then-else statements to complete the same task. These two methods were used for our comparison and performance analysis.

The table below demonstrates the execution times of the switch statement and if-then-else implementation, counting the numbers for several array sizes.

Number Counting - Execution Time

Array Size	10	100	500	1000	10000	15000
Switch Statement	27 µ sec	29 µ sec	44 µ sec	56 µ sec	290 µ sec	420 µ sec
If-Then-Else	14 µ sec	14 µ sec	22 µ sec	60 µ sec	297 µ sec	462 µ sec

For smaller array sizes we can see that the if-then-else implementation was faster. Upon analyzing my code, I believe this is due to the division operation in my switch expression, which is performed in every iteration of the for loop. Using the suggested approach from the assignment instructions, I divide each array value by 200 to allow me to compare easily with a small integer for each 'case', since the fractional part of the result is discarded. This division is a costly operation to be performed for every array element, thus, for a smaller number of array elements the sequence of comparisons involved with the if-then-else method out performs the switch statement.

As I ran tests with larger arrays sizes I noticed that the switch statement outperformed if-then-else each time. A switch construct is more easily translated into a jump table. The idea behind a jump table is to place a bunch of jump instructions sequentially in memory and add the value to the program counter. This replaces a sequence of comparison instructions (that we see with if-then-else) with an add operation. I have also noticed that the compiler has the ability to optimize the switch statement, where in the case of the if-then-else the code must process each if statement in the order determined by the programmer. For these reasons the switch statement performed better for a larger number of array values.

The screen capture below displays a portion of the .s file assembly code for the program. Here I am showing the switch statement implementation. We can see the setup of each switch case and the corresponding jump table that is established.

```
LBB0_3:                                 ## =>This Inner Loop Header: Depth=1
        movslq  (%rdi), %rax
        imulq   $1374389535, %rax, %rax ## imm = 0x51EB851F
        movq    %rax, %rbx
        shrq    $63, %rbx
        sarq    $38, %rax
        addl    %ebx, %eax
        cmpl    $4, %eax
        ja      LBB0_10
## BB#4:                                ##   in Loop: Header=BB0_3 Depth=1
        movslq  (%r10,%rax,4), %rax
        addq    %r10, %rax
        jmpq    *%rax
LBB0_5:                                 ##   in Loop: Header=BB0_3 Depth=1
        incl    %esi
        jmp     LBB0_10
LBB0_6:                                 ##   in Loop: Header=BB0_3 Depth=1
        incl    %edx
        jmp     LBB0_10
LBB0_7:                                 ##   in Loop: Header=BB0_3 Depth=1
        incl    %ecx
        jmp     LBB0_10
LBB0_8:                                 ##   in Loop: Header=BB0_3 Depth=1
        incl    %r8d
        jmp     LBB0_10
LBB0_9:                                 ##   in Loop: Header=BB0_3 Depth=1
        incl    %r9d
        .align  4, 0x90
LBB0_10:                                ##   in Loop: Header=BB0_3 Depth=1
        addq    $4, %rdi
        decl    %r11d
        jne     LBB0_3
        jmp     LBB0_11
LBB0_1:
        xorl    %r8d, %r8d
        xorl    %ecx, %ecx
        xorl    %edx, %edx
        xorl    %esi, %esi
LBB0_11:                                ## %._crit_edge
        leaq    L_.str(%rip), %rdi
        xorl    %eax, %eax
        popq    %rbx
        popq    %rbp
        jmp     _printf                 ## TAILCALL
        .cfi_endproc
        .align  2, 0x90
L0_0_set_5 = LBB0_5-LJTI0_0
L0_0_set_6 = LBB0_6-LJTI0_0
L0_0_set_7 = LBB0_7-LJTI0_0
L0_0_set_8 = LBB0_8-LJTI0_0
L0_0_set_9 = LBB0_9-LJTI0_0
LJTI0_0:
        .long   L0_0_set_5
        .long   L0_0_set_6
        .long   L0_0_set_7
        .long   L0_0_set_8
        .long   L0_0_set_9
```

The screen-capture below displays a portion of the .s file assembly code, displaying a piece of the if-then-else implementation. We can see the sequence of comparison instructions performed that correspond with the if statements written in my c program, in the same order as they are written.

```
LBB1_1:                                 ## =>This Inner Loop Header: Depth=1
        movl    (%r15), %eax
        cmpl    $199, %eax
        jg      LBB1_3
## BB#2:                                ##   in Loop: Header=BB1_1 Depth=1
        incl    %ebx
        jmp     LBB1_12
        .align  4, 0x90
LBB1_3:                                 ##   in Loop: Header=BB1_1 Depth=1
        cmpl    $399, %eax              ## imm = 0x18F
        jg      LBB1_5
## BB#4:                                ##   in Loop: Header=BB1_1 Depth=1
        incl    %r13d
        jmp     LBB1_12
        .align  4, 0x90
LBB1_5:                                 ##   in Loop: Header=BB1_1 Depth=1
        leal    -400(%rax), %edx
        cmpl    $199, %edx
        ja      LBB1_7
## BB#6:                                ##   in Loop: Header=BB1_1 Depth=1
        incl    %ecx
        jmp     LBB1_12
LBB1_7:                                 ##   in Loop: Header=BB1_1 Depth=1
        leal    -600(%rax), %edx
        cmpl    $199, %edx
        ja      LBB1_9
## BB#8:                                ##   in Loop: Header=BB1_1 Depth=1
        incl    %r12d
        jmp     LBB1_12
LBB1_9:                                 ##   in Loop: Header=BB1_1 Depth=1
        addl    $-800, %eax             ## imm = 0xFFFFFFFFFFFFFCE0
        cmpl    $199, %eax
        ja      LBB1_11
## BB#10:                               ##   in Loop: Header=BB1_1 Depth=1
        incl    %r9d
        jmp     LBB1_12
```

The terminal output below displays the functionality of my program, which prints the number counts and execution times for both methods upon completing the counting process.

ARRAY SIZE: 10

```
Erics-MacBook-Pro-2:HW03 Watson$ ./a.out
Input Array Size: 10
Switch Statement Counts:
0 - 199:     6
200 - 399:   0
400 - 599:   4
600 - 799:   0
800 - 999:   0
Switch Statement - Count Execution Time: 0.000027 seconds

If-Then-Else Statement Counts:
0 - 199:     6
200 - 399:   0
400 - 599:   4
600 - 799:   0
800 - 999:   0
If-Then-Else Statement - Count Execution Time: 0.000014 seconds
```

ARRAY SIZE: 100

```
Erics-MacBook-Pro-2:HW03 Watson$ ./a.out
Input Array Size: 100
Switch Statement Counts:
0 - 199:     16
200 - 399:   25
400 - 599:   18
600 - 799:   14
800 - 999:   27
Switch Statement - Count Execution Time: 0.000029 seconds

If-Then-Else Statement Counts:
0 - 199:     16
200 - 399:   25
400 - 599:   18
600 - 799:   14
800 - 999:   27
If-Then-Else Statement - Count Execution Time: 0.000014 seconds
```

ARRAY SIZE: 500

```
Erics-MacBook-Pro-2:HW03 Watson$ ./a.out
Input Array Size: 500
Switch Statement Counts:
0 - 199:     107
200 - 399:   104
400 - 599:   86
600 - 799:   96
800 - 999:   107
Switch Statement - Count Execution Time: 0.000044 seconds

If-Then-Else Statement Counts:
0 - 199:     107
200 - 399:   104
400 - 599:   86
600 - 799:   96
800 - 999:   107
If-Then-Else Statement - Count Execution Time: 0.000022 seconds
```

ARRAY SIZE: 1000

```
Erics-MacBook-Pro-2:HW03 Watson$ ./a.out
Input Array Size: 1000
Switch Statement Counts:
0 - 199:     190
200 - 399:   209
400 - 599:   185
600 - 799:   206
800 - 999:   210
Switch Statement - Count Execution Time: 0.000056 seconds

If-Then-Else Statement Counts:
0 - 199:     190
200 - 399:   209
400 - 599:   185
600 - 799:   206
800 - 999:   210
If-Then-Else Statement - Count Execution Time: 0.000060 seconds
```

ARRAY SIZE: 10,000

```
Erics-MacBook-Pro-2:HW03 Watson$ ./a.out
Input Array Size: 10000
Switch Statement Counts:
0 - 199:     2007
200 - 399:   2030
400 - 599:   1948
600 - 799:   1993
800 - 999:   2022
Switch Statement - Count Execution Time: 0.000293 seconds

If-Then-Else Statement Counts:
0 - 199:     2007
200 - 399:   2030
400 - 599:   1948
600 - 799:   1993
800 - 999:   2022
If-Then-Else Statement - Count Execution Time: 0.000297 seconds
```

ARRAY SIZE: 15,000

```
Erics-MacBook-Pro-2:HW03 Watson$ ./a.out
Input Array Size: 15000
Switch Statement Counts:
0 - 199:     2927
200 - 399:   3033
400 - 599:   3061
600 - 799:   2963
800 - 999:   3016
Switch Statement - Count Execution Time: 0.000420 seconds

If-Then-Else Statement Counts:
0 - 199:     2927
200 - 399:   3033
400 - 599:   3061
600 - 799:   2963
800 - 999:   3016
If-Then-Else Statement - Count Execution Time: 0.000462 seconds
```

Eric Watson

ECE335 Homework 3 Source Code (HW03.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

void switch_statement (int A[], int size){
  int i;
  int count0 = 0;                       // tally for 0 - 199
  int count1 = 0;                       // 200- 399
  int count2 = 0;                       // 400-599
  int count3 = 0;                       // 600-799
  int count4 = 0;                       // 800-999

  for (i=0; i< size; i++){

    switch(A[i]/200){                   // number divided by 200

      case 0:                   // if number divided by 200 returns 0,
        count0++;               //  then it must be in the range 0-199
        break;

      case 1:
        count1++;
        break;

      case 2:
        count2++;
        break;

      case 3:
        count3++;
        break;

      case 4:
        count4++;
        break;
    }
  }

// Print out the tallys
  printf("Switch Statement Counts:\n"
         "0 - 199:     %i \n"
         "200 - 399:   %i \n"
         "400 - 599:   %i \n"
         "600 - 799:   %i \n"
         "800 - 999:   %i \n", count0,count1,count2,count3,count4);
}

void iffy(int B[], int size){
  int i = 0;
  int count0 = 0;                       // tally for 0 - 199
  int count1 = 0;                       // 200- 399
  int count2 = 0;                       // 400-599
  int count3 = 0;                       // 600-799
  int count4 = 0;                       // 800-999

  for (i=0; i < size; i++){
    if (B[i] < 200){
      count0++;
    }
    else if(B[i] >= 200 && B[i] < 400){
      count1++;
    }
    else if(B[i] >= 400 && B[i] < 600){
      count2++;
    }
```

```
    else if(B[i] >= 600 && B[i] < 800){
      count3++;
    }
    else if(B[i] >= 800 && B[i] < 1000){
      count4++;
    }
    else{
      printf("Value out of range, check random number generator.\n");
    }
  }
  // Print out the tallys
    printf("If-Then-Else Statement Counts:\n"
           "0 - 199:     %i \n"
           "200 - 399:   %i \n"
           "400 - 599:   %i \n"
           "600 - 799:   %i \n"
           "800 - 999:   %i \n", count0,count1,count2,count3,count4);
}
int main(){

        clock_t begin_switch, end_switch, begin_iffy, end_iffy;
        double switch_time;
        double iffy_time;
        int size,i,j,k,l,m,x,w,z;

        time_t seconds;
        time(&seconds);
        srand((unsigned int) seconds);

        printf("Input Array Size: ");
        scanf("%d", &size);
        int A[size];                        // initialize array of input size
        int B[size];                        // initialize array for second couting method

        for (i=0; i< size; i++){            //populate with random integers
                A[i] = rand() % 1000;       // use modulus to limit range from 0 to 999
  }

/*DEBUGGING - Make sure the integers are randomly generated
        printf("The randomly generated values are:\n");
        for (j =0; j < size; j++){
                printf("%d\n", A[j]);
        }
*/

// Copy the Random valued integer array for the second counting method
        for (z=0; z < size; z++){
                B[z] = A[z];
        }

        begin_switch = clock();                 // start clock
        switch_statement(A,size);               // run switch statement counting method
        end_switch = clock();                   // end clock

        switch_time = (double)(end_switch-begin_switch)/ CLOCKS_PER_SEC;
//calculate time (sec)

        printf("Switch Statement - Count Execution Time: %f seconds\n\n", switch_time);

        begin_iffy = clock();                   //start clock
        iffy(B, size);                          // run if-then-else counting method
        end_iffy = clock();                     // end clock
        iffy_time = (double)(end_iffy-begin_iffy)/ CLOCKS_PER_SEC;

        printf("If-Then-Else Statement-Count Execution Time: %f seconds\n\n", iffy_time);

        return(0);
}
```