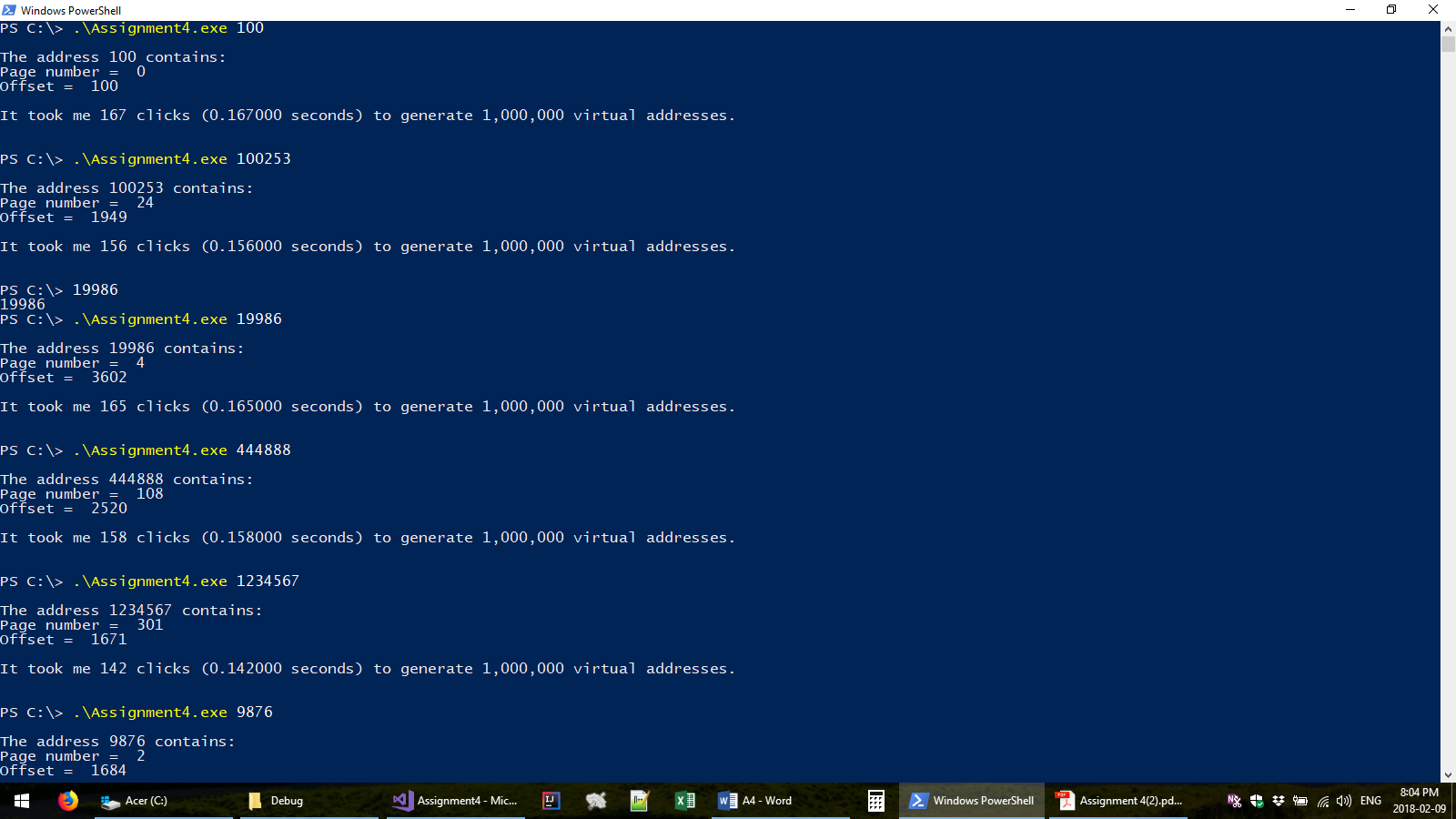
I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work. Jason Choquette 104337378

6 runs of the program:



int main(int argc, char \*\* argv)

{

unsigned int virtual\_address = (unsigned int) atoll(argv[1]);

unsigned int page\_number     = bitwiseDivision(virtual\_address, PAGE\_SIZE);

unsigned int offset     = bitwiseMod(virtual\_address, PAGE\_SIZE);

printf("\nThe address %lu contains:\n", virtual\_address);

printf("Page number =  %lu\n", page\_number);

printf("Offset =  %lu\n\n", offset);

// seed prng

srand((unsigned int)time(NULL));

// begin timer

clock\_t t1 = clock();

int i;

int n = 1000000;

for (i = 0; i < n; i++)

{

virtual\_address = rand(); // rand() range is 0 - 2^32 - 1

bitwiseDivision(virtual\_address, PAGE\_SIZE);

bitwiseMod(virtual\_address, PAGE\_SIZE);

}

// calculate elapsed time

t1 = clock() - t1;

printf("It took me %d clicks (%f seconds) to generate 1,000,000 virtual addresses.\n\n\n", t1,

((float)t1) / CLOCKS\_PER\_SEC);

return 0;

}

The main function calculates the page number and offset of a virtual address. The page number is calculated by dividing the virtual address by the page size.

The offset is calculated taking the modulus of the virtual address by the page number. Both the modulus and division operations are implemented as efficient as possible. The implementations are as follows.

unsigned int bitwiseMod(unsigned int dividend, unsigned int divisor)

{

return dividend & (divisor - 1);

}

The bitwiseMod function simply returns the dividend bit-wise ‘anded’ with the divisor subtracted by 1

unsigned int bitwiseDivision(unsigned int dividend, unsigned int divisor)

{

unsigned int quot;

\_\_asm {

mov  eax, [dividend];// quot = dividend

mov  ecx, [divisor]; // divisor

mov  edx, 32;        // bits\_left

mov  ebx, 0;         // rem

$div\_loop:

add  eax, eax;       // (rem:quot) << 1

adc  ebx, ebx;       //  ...

cmp  ebx, ecx;       // rem >= divisor ?

jb  $quot\_bit\_is\_0;  // if (rem < divisor)

$quot\_bit\_is\_1:          //

sub  ebx, ecx;       // rem = rem - divisor

add  eax, 1;         // quot++

$quot\_bit\_is\_0:

dec  edx;            // bits\_left--

jnz  $div\_loop;      // while (bits\_left)

mov[quot], eax;      // quot

}

return quot;

}

Obtained from: [https://stackoverflow.com/questions/2776211/how-can-i-multiply-and-divide-using-only-bit-shifting-and-adding/19780781#19780781](https://stackoverflow.com/questions/2776211/how-can-i-multiply-and-divide-using-only-bit-shifting-and-adding/19780781%2319780781)