Ethan Woyehoski

CSIS 3740

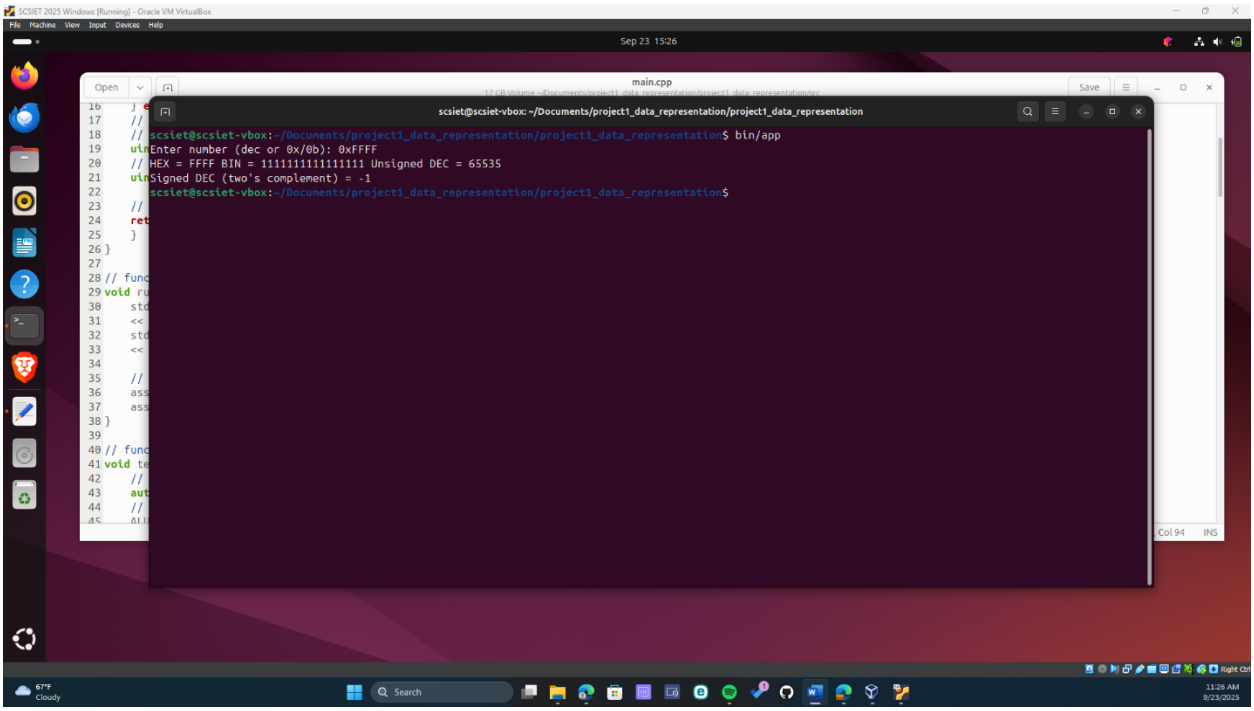
Prof. Jiang

9/24/25

Project One Report

Two's complement is a system that represents signed integers in binary. Positive numbers are still stored in their normal binary form, but negative numbers are represented by inverting the bits of the absolute value and then adding one. This makes binary arithmetic work consistently for both positive and negative numbers. In my program, I convert the negative numbers using a separate function. The function is called in main and passes the unsigned integer that was provided by the user to this helper function. The function first checks the most significant bit. If the MSB is zero, then the number is already positive, and it returns to main. If the MSB is one, the number is negative. In which case, the two's complement function begins inverting (flipping) the absolute value binary bits of the unsigned integer. Then it adds one to the now flipped binary number. Then, it returns the now-signed two's complement integer to the main function, so it can be displayed on the screen to the user. This function ensures that values like 0xFFFF are

displayed correctly as -1 to the user.



My flags in the program help to test several cases that I created. The flags make sure that the ALU does the arithmetic or logic correctly. Each test case checks the numerical result and confirms that the correct flags were set. If the ALU's flags match the expected flags, then the operations are running correctly. The Z flag is for when the program results in a zero after doing the specified operation in the test. For example, 0x0001 minus 0x0001 would equal 0x0000. The N flag is set if the program ends up with a negative number after the desired operation is complete. For example, 0x0000 minus 0x0001 would equal 0xFFFF or negative one. The C flag is set if the program goes over the 16-bit and into the 17-bit. It is also set if the operation requires a borrow to be completed. Both situations result in exceeding the 16-bit unsigned capacity. For example, 0xFFFF plus 0x0001 would equal 0x0000, which would mean a wrap-around occurred. The V flag is for the signed operations overflows. It is set only if the number goes beyond -32768 to 32767. This is the maximum that the program can have, and the flag must be set if it

exceeds this. An example of this would be adding 0x7FFF and 0x0001, which would result in 0x8000 and would cause the number to flip from positive to negative, so the V flag is set. The screenshot below shows how all the flags are set correctly and provides a breakdown of what each test is covering.