**Part I:**

**Part II:**

The way I implemented my MIPS program to convert a 32-bit signed binary number to its two’s complement was I first prompt the user to enter the 32-bit signed/magnitude binary number as a string. For entering the signed binary number, the user must enter all 32 bits with spaces between for every 4 bits (i.e. 0000 0000 0000 0000 0000 0000 0000 0000). After input, the program passes the string to a function to check if it’s valid signed binary number. A valid signed binary number can have 0’s, 1’s, and spaces. If the string is invalid, then the program will print out their invalid input and print out how it should look like with a small explanation, then re-prompts the user to enter their signed binary number. After the user entered a valid signed binary number, the program passes the string to a function to start converting the signed binary number to its decimal form and then to its two’s complement form. Before the program starts the conversion, it sets up counters for the loop condition and computing the powers of 2 for the respective place value in the signed binary number using logical shift. Once setup is done, the program doesn’t read each character of the string yet, it reads the first character of the string to check if the signed binary number is negative. If the character is negative, then it stores the value -1 in a register to use for later in the program for calculating the its decimal (stores 1 in a register if positive). Next the program starts iterating the string using a loop starting at the second character since it already read the first character. Inside the loop, the program stores the read character into a register, that register is then passed in a function to check if it’s a space character. If the character is not a space, then the program parses the character to integer and multiplies the integer by the result of the logical shift from the counter said earlier. The product is then added to a sum. The program repeats the process for the next characters of the string until it visits the second last character of string to avoid the null character at the end. When the program is done reading the string, it takes the sum and multiplies it by the register that stores -1 or 1 from earlier when the program read the first character of the string. We now have the decimal of the signed binary number. Finally, the program takes the decimal and prints it out using syscall 35 which prints out the integer to binary in two’s complement format.