Batch Number Conversion using MIPS Assembly Language

# Problem Statement

The task is to write a MIPS Assembly language program that reads a file and performs conversions present in each line of the text. Each line of text consists of Input Type (one character, b: binary, d: decimal, or h: hexadecimal), Input Length (1 to 32), Output Type (one character, b: binary, d: decimal, or h: hexadecimal), Colon (:) and space and value input (signed/unsigned decimal, binary or hexadecimal with optional whitespace). The program should produce output of the format Input Type (one character, b: binary, d: decimal, or h: hexadecimal), Colon (;) and space and converted output value. Include test file with all combinations of conversions including implicitly and explicitly signed values. Binary and hexadecimal output should always include thirty-two bits. Decimal output must not include leading zeroes.

# Approach

To write this assignment program in MIPS assembly language, a simulator IDE called MARS (MIPS Assembler and Runtime Simulator) is used. The decision was to calculate and perform conversions based on each line of text from an input file given by the user. The filename length and total bytes in the file should be less than 128 bytes. Buffer size for output variable should be less than 41 bytes. The file read buffer should be one byte.

# Solution

For solving the given problem, we open and read the file in MIPS line by line iterating through each byte character and based on the input type, input length, output type and input value present on each line, we perform the necessary mathematical calculations on the ASCII input value to produce the required conversion in decimal, binary or hexadecimal format.

The program consists of a main loop to iterate the input file from top to bottom. This main loop is mainly divided into two sections which are the input section and the output section. The input section reads each ASCII character present in each line and determines the input type, input length and output type. The input type helps in determining the base for converting the input value into decimal value which will later get converted into output value based on output type. While reading the input value whitespace is ignored and optional sign is also handled.

The output section performs the main conversion of various combinations such as from binary to decimal, decimal to binary, binary to hexadecimal, and so on using certain algorithm. The conversion of a decimal to binary number is done using the two's complement method wherein the decimal number is divided by 2 repeatedly and adding the remainder to the output string.

Similarly, the process for all other conversions is followed including implicitly and explicitly signed values. Thus, the output for different conversions based on the input file is printed on the console as per the requirement given in the problem statement.

# Assembler Output

Figure 1 shows the successful operation of converting high level language to assembly language. There was no error while assembling it. The bottom message “Assemble: operation completed successfully” confirms it.

Graphical user interface, text, application, email

Description automatically generated

Figure 1 Assembler Output - Shows that the code has been assembled successfully without any errors.

# Output of the Program

Figure 2 shows the program output. It shows the output corresponding to all possible combinations present in the given input file (file.txt). We can also see the program has finished running.

Table

Description automatically generated with medium confidence

Text

Description automatically generated

Figure 2 Program Output – shows the program output with different possible conversations.