The task that I'm faced with is to convert a two digit octal number to a decimal and check if it's a multiple of 3 or 5. If it is, print out the decimal number, if not print 0. The input to the assembler is a two digit octal number that is being entered as ASCII characters. Additionally, the assembler reads the char input as a decimal. It's important to note that each character is read individually, which means each digit gets converted one at a time. The steps to solve this problem are as follows:

1) Store in the input values
2) Convert the input two digit octal digits to decimal, then combine the values
3) Check for multiplicity of 3 or 5
4) Output the values

1) The first step is to store the digits that are being inputted. To do this in assembly I will use INP to read the characters and STA to store the accumulator contents in the memory. This will be done twice since each character is being read individually.

2) The next step will be to convert an octal to a decimal number. My C++ code below follows the concept I want to use for this project. In assembly, I will need to subtract the first digit and second digit by 48 to get the decimal equivalent of the octal since '0' is decimal 48. When subtracting there are no instructions for assembly so we need to get the second complement and use ADD. To do this we take the first complement and increment by 1. Then to go from octal to decimal we use a math formula where the first number is multiplied by 8^1 or 8 and the second digit by 8^0 which is 1. In assembly, I will make this happen by using LDA MSD to load the most significant digit and CMA to take the complement of the value in the accumulator, and INC to increment to find the second complement. This performs a subtraction and then it will get stored with STA. To add the two values the instruction that will be used is ADD, which all together will be acting as a subtration.

```
char MSD, LSD; //most significant digit and least significant digit
int value;
Cout << "Enter the first character";
cin >> MSD;
cout << "Enter the second character";
cin >> LSD;
MSD = MSD - '0'; //minus zero because it's a char, in assembly - 48 will need to be used because 0 char represents 48 as decimal.
LSD = LSD - '0';
MSD = MSD *8;      //only need to multiply the first digit by 8 because the conversion from an octal to decimal takes the first value *8^1 and second *8^0 which equals to 1 so it can be ignored
value = MSD + LSD;
```

3) Now to check to see if a value is a multiple of 3 or 5 I created a C++ code that is below and will follow the same steps in assembly. The issue that I face with assembly is that I can't use division since that is not available in the instructions, so I have to do repeated subtraction. Additionally, the accumulator is going to get checked each time and if it's greater than 0 then it needs to keep subtracting, if it's equal to zero it is a multiple and if it's a negative it's not a multiple. To implement this in assembly language a loop is required because of repeated subtraction. Now this language doesn't provide loops so I need to use conditions that make it jump to the top to repeat the body. There needs to be a condition that makes you jump to the top of the loop to repeat and one that exits the loop. There will be a "LOOP_START" to indicate the beginning of the loop and the initial value will be the value from part 2 which is my decimal. My end value and stopping condition will be if the value is or isn't a multiple of 3 or 5. I will use CMA and INC to get the second complement just like part 2 to do repeated subtraction, and will use SPA which skips to the next instruction if the accumulator is positive, SNA which skips to the next instruction if the accumulator is negative, and SZA which skips if the accumulator is zero. All of these will help determine if the loop should jump to the beginning or exit the loop. If the remainder after subtracting is positive, then it needs to repeat the loop, if it's negative it needs to exit, and if it's zero exit because it's a multiple. There will be a "LOOP_EXIT" with instructions on how the program behaves after which is the printing portion.

```
int remainderThree = value; //the decimal value is being initialized to test if it's a multiple of 3
while (remainderThree >= 3){
        remainderThree -= 3;
}
int remainderFive = value; //the decimal value is being initialized to test if it's a multiple of 5
while (remainderFive >= 5){
        remainderFive -= 5;
}
```

4) Lastly, I will create subroutines to output the decimal if the the value is a multiple of 3 or 5 with OUT and another that resets the value to 0 if it's not a multiple and displays it with OUT. I will also use END to finish my program.

```
if (remainderThree == 0 || remainderFive == 0){
        count << value << endl;
}
else {
        Count << "0" << endl;
}
```