Hints provided in the project 6 instructions:

```
int number; // The input number
int digit1, digit2, digit3; // The digits of the decimal number
int largestDigit = 0; // The variable for storing the biggest digit
printf("Enter a hexadecimal number: ");
scanf("%x", &number);
digit1 = number / 100;
number = number % 100;
digit2 = number / 10;
digit3 = number % 10;
largestDigit = (digit1 > digit2) ? ((digit1 > digit3) ? digit1 : digit3) :
((digit2 > digit3) ? digit2 : digit3);
printf("The largest digit is: %d\n", largestDigit);
```

```
while (dividend >= divisor) {
        dividend -= divisor;
        quotient++;
}
printf("Quotient: %d\n", quotient);
printf("Remainder: %d\n", dividend);
```

My program approach will be as follows:

1. Read MSB
2. Check if it is 1-9 or A-F
3. Convert to decimal via ascii table dependent on #2
4. multiply by 16 and store in MSB_DEC variable
5. read LSB
6. repeat 2 & 3 then store in n variable

For checking largest digit:

1. add MSB_DEC and LSB_DEC
2. check 1st and 2nd digit
3. if 1st > 2nd, store 1st as largest digit
4. if 1st < 2nd, store 2nd as largest digit
5. otherwise, just store 2nd as largest digit
6. repeat for third digit against the current largest digit and output

My thought process is to use BUN to accomplish loops in checking the largest digit. Its "break case" should depend on if there are two or three digits in the decimal representation of the hex value. Utilizing SPA, SNA, and SZA will also be necessary for checking if the hexadecimal falls between 1-9 or A-F, as well as using it for comparisons of the largest digit. Although I am still generating ideas on how to accomplish this project, I believe many things can be accomplished between the comparisons and branching capabilities of the assembler.