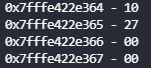
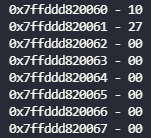
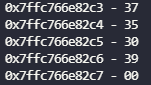
**Laboratório 2**

1. Função dump
   1. **Crie um arquivo C contendo as funções dump e main, compile-o e execute o programa. Explique o resultado desse programa (qual o tamanho, em bytes, de um inteiro? como esses bytes estão ordenados na memória?). Experimente, se necessário, outros valores para i.**  
        
      O resultado foi:  
        
      Que é o valor em hexadecimal para 10000 e em 4 bytes pois é um *int*
   2. **Aplique a mesma técnica para ver como um long é armazenado.**O resultado foi:  
      Agora foram alocados 8 bytes para armazenar a variável i
   3. **Aplique a mesma técnica para ver como um short é armazenado.**Agora somente 2 bytes foram alocados
   4. **Aplique a mesma técnica para ver como um char é armazenado. Lembre-se que uma variável do tipo char armazena um valor inteiro. Experimente atribuir a essa variável tanto um caractere como o valor do código ASCII correspondente (por exemplo, 'a' e 97).**61 é 91 em hexadecimal
   5. **A mesma técnica pode ser usada para vermos como uma string é armazenada:**p[] = “7509”  
        
      **Modifique o exemplo acima para descobrir os códigos ASCII dos caracteres 'A', ' ', '\n', e '$'.**  
      **'**A**'** =   
      **' ' =**  **'**\n**' =**  **'**$**' =** 
2. Compile e execute o seguinte programa:
   1. **Procure entender como funciona a função string2num. O que está acontecendo? O que será o valor de (\*s - '0') a cada iteração? E o valor de a?**  
      Ele está traduzindo o número em formato de string para um int. O valor de (\*s - '0') é a chave para isso pois o resultado desse cálculo retorna o número em int de cada caractere na string. O ‘a’ é nada mais do que o número int montado
   2. **Modifique a função string2num do exercício anterior para converter strings contendo numerais escritos em qualquer base entre 2 e 10. A base em que o numeral está escrito deverá ser dada como um segundo parâmetro (do tipo int) para a função.**