题目描述

实现一种整数编码方法，使得待编码的数字越小，编码后所占用的字节数越小。

编码规则如下:

1. 编码时7位一组，每个字节的低7位用于存储待编码数字的补码
2. 字节的最高位表示后续是否还有字节，置1表示后面还有更多的字节，置0表示当前字节为最后一个字节。
3. 采用小端序编码，低位和低字节放在低地址上。
4. 编码结果按16进制数的字符格式输出，小写字母需转换为大写字母

输入描述

输入的为一个字符串表示的非负整数

输出描述

输出一个字符串，表示整数编码的16进制码流

备注

待编码的数字取值范围为[0，1<<64 - 1]

用例

输入	0
输出	00
说明	输出的16进制字符，不足两位的前面补0，如00、01、02。

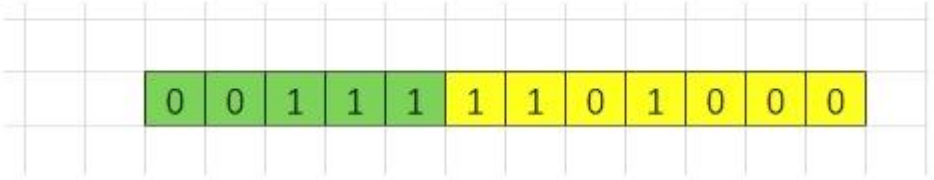
输入	100
输出	64
说明	100的二进制表示为0110 0100，只需要一个字节进行编码; 字节的最高位置0，剩余7位存储数字100的低7位 (110 0100)，所以编码后的输出为64。

输入	1000
输出	E807
说明	1000的二进制表示为0011 1110 1000，至少需要两个字节进行编码; 第一个字节最高位置1，剩余的7位存储数字1000的第一个低7位 (1101000)，所以第一个字节的二进制为1110 1000，即E8; 第二个字节最高位置0，剩余的7位存储数字1000的第二个低7位 (0000111)，所以第一个字节的二进制为0000 0111，即07; 采用小端序编码，所以低字节E8输出在前，高字节07输出在后。

题目解析

我的解题思路如下：

首先将输入的十进制数转为二进制字符串binStr，然后基于binStr“倒序”每7位一段，以用例3画图说明：



如果当前“段”的左边还有后续段，那么当前“段”需要在头部追加“1”形成新字节，如上图黄色段，由于其左边还有，则最终得到的新字节字符串是：“1” + “1101000”，新字节字符串转化为十六进制后为E8

如果当前“段”不足7位，或者左边没有后续段了，那么当前“段”需要在头部追加“0”形成新字节（其实也可以不加），如上图绿色段，由于其左边没有后续段了，则最终得到的新字节字符串是：“0“ + ”00111“，新字节字符串转化为16进制后为7

对于不足两位的16进制，要在前面补足0

Java算法源码

```java
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println(getResult(sc.nextLong()));
  }

  public static String getResult(long num) {
    String bin = Long.toBinaryString(num);

    StringBuilder ans = new StringBuilder();

    int end = bin.length();
    while (end - 7 > 0) {
      ans.append(getHexString("1" + bin.substring(end - 7, end)));
      end -= 7;
    }

    if (end >= 0) {
      ans.append(getHexString(bin.substring(0, end)));
    }

    return ans.toString();
  }

  public static String getHexString(String binStr) {
    String hexStr = Integer.toHexString(Integer.parseInt(binStr, 2));
    if (hexStr.length() == 1) hexStr = "0" + hexStr;
    return hexStr.toUpperCase();
  }
}
```

JS算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

rl.on("line", (line) => {
  console.log(getResult(line));
});

function getResult(numStr) {
  const binStr = BigInt(numStr).toString(2);

  const ans = [];
  let end = binStr.length;
  while (end - 7 > 0) {
    ans.push(getHexString("1" + binStr.substring(end - 7, end)));
    end -= 7;
  }

  if (end >= 0) {
    ans.push(getHexString(binStr.substring(0, end)));
  }

  return ans.join("");
}

function getHexString(binStr) {
  let hexStr = parseInt(binStr, 2).toString(16);
  if (hexStr.length == 1) hexStr = "0" + hexStr;
  return hexStr.toUpperCase();
}
```

Python算法源码

```python
# 输入获取
num = int(input())


def getHexString(binStr):
    # bin函数可以将十进制数转为16进制字符串，但是开头会带0x，因此下面做了字符串截取操作
    hexStr = hex(int(binStr, 2))[2:]
    if len(hexStr) == 1:
        hexStr = "0" + hexStr
    return hexStr.upper()


# 算法入口
def getResult():
    # bin函数可以将十进制数转为二进制字符串，但是开头会带0b，因此下面做了字符串截取操作
    binStr = bin(num)[2:]

    ans = []

    end = len(binStr)
    while end - 7 > 0:
        ans.append(getHexString("1" + binStr[end-7:end]))
        end -= 7

    if end >= 0:
        ans.append(getHexString(binStr[:end]))

    return "".join(ans)


# 算法调用
print(getResult())
```