Assembly Language and Computer Architecture Lab

Week 5 (Homework)

Print decimal output

My program implements the division of hexadecimal: result of multiplication divided by 10 (in the program: **0x77ffffff1** / **0xa**). There are 2 loops:

- Inner loop (**DIVISION**): divide the current number by 10, store the quotient and remainder. Since the current number is stored in 2 separated registers, each pair of digit forms a byte from the left (MSB) to the right (LSB) will be divided by 10, store the quotient to 2 registers (1 for lower part and 1 for upper part incase the quotient is a 64-bit number), the remainder will pair with the next digit of the dividend to form a new byte for division. I use a counter (count from 8 to 0 for each register, because each of them stored 8 hexadecimal digits) to know when the process is finished (it means all digits of 64-bit dividend are used).
- Outer loop (**LOOP_DECIMAL**): Keep implementing division until the dividend is zero. At the beginning, the dividend is the 64-bit number which is the result of multiplication. After that, the previous quotient will be chosen as the new dividend. The remainder of each division will be converted to character follows the **ASCII** table, then stored into a string segment (the pointer to string will run backward from the end of string).

Print hexadecimal output

- For the multiplicands, I used code service **34** to print 2 integers in hexadecimal (because they are both 32-bit number).
- For the result of multiplication as 64-bit number, the upper part is stored in **HI** register, and **LO** for the lower part. These digits in registers will be converted to characters and then added into a string to be printed:
  - Init a string with 16 space character ‘ ‘
  - Loop through each hexadecimal digit of the 64-bit result stored in **HI register** first, then in **LO register** by shifting the register to the left by 1 digit (4 bits) and get its most left digit until both registers are equal zero (it means all digits are taken).
  - For each digit taken from registers, add 48 for representation of number in ASCII table, or add (48 + 39) for representation of alphabet characters (a – f), then store these digit as characters in the memory segment of initialized string.
  - End the string with a **NULL** value.