Arthur Wei

Sanjana Sasmal

Shreya Kulkarni

The program’s first line is mov, which has the operands of opWord which contains 100 and registry ax. It sets the register ax to 100. Then it does mul which is multiplying register ax with ax with is itself so it would be 100 \* 100. The assembler will store this as dx:ax which effectively is a 32 bit register because a 16 and a 16 bit register is added together. Ax will store as much as possible but if the number is bigger than what a 8-bit register can hold it will spill over to dx which, in this case, is not needed. To confirm this is correct we need to multiply dx by 65536 which is 0 then add the value of ax which is 10000 to it. Which gives us the correct answer.

The next lines deal with a long which are 32 bit numbers. We will be dealing with 32 bit registers which are basically the same. We first set the register eax to 1000. Then, it multiplies the register eax with itself (again) , so the new value of eax 1000\*1000 = 1000000. This number also does not overflow the registry it stays only in eax. To prove this, if you were to print edx and multiply it by 65536, you would get 0. Then you would add 1000000 to it which would get you the answer.

The next part is division. The next block sets the register ax to 100 and the register dx to 100, so the value in ax is (100\*65536 ) + 100 so the final value in register ax is 6553700. It divides the value in register ax by 4000. 6553700/4000 = 1638 remainder 1700The result is that the quotient is stored in ax as 1638 and the remainder is stored in dx as 1700. So it would 1700 : 1638

The next block sets the register eax to 1000 and the register edx to 1000, so the value in eax is (1000\* 2^32) + 1000 so the final value in register eax is 4294968296 and divides the value in eax by 5000. 4294968296/5000 = 858993459R2000. The quotient, 858993459, is stored in eax is and the remainder, 2000, is stored in edx.