1 This subroutine performs addition on two 16bit integers. The subroutine first sets the x,y, and z registers to point to specific locations in memory, those are x:0100, y:0102, z:0104. Because this is an 8bit architecture and we are adding 16 bit numbers the addition must be carried out in two steps. First the lower 8 bits of X and Y are loaded into A and B respectively. Using a post increment X and Y are incremented during the load to point to the higher 8 bits for a later step. A and B are added together and stored into Z, which is also incremented on load. Because we are trying to add 16 bit numbers on an 8 but architecture there may be a carry which add will set if necessary. Again A and B are loaded with X and Y, this time loading the higher 8 bits because of the previous increment to X and Y. Addition is performed again, this time using ADC which will add the carry to the result if it is set. The result is stored in Z, which is incremented again just in case there is a further carry. Last there is a conditional branch that branches if the carry flag has been cleared. If there was no carry then the subroutine returns as is. If there was a carry then it places the value of XH into Z. Z currently points to the high value of the result and XH is a one, effectively adding the carry to the result.

2 Any two 16 bit numbers that when added result in a number larger than a 16 bit number can represent will result in the extra lines being executed. “ffff” and “ffff” are examples.

3 The purpose of the line st Z, XH is to append a 1 onto the result of this operation(addition) in case there is overflow.