Note on fixed point C programming

```
 //example code that computes the following equation: A=B*C*D+E
 //all values in variables of left hand side are assumed to be in Q15
 //created by PK

#include <math.h>


int main(void) {

        int B=pow(10,4); //Q15 (0.30517578125)
        int C=pow(10,4); //Q15
        int D=pow(10,4); //Q15
        int E=pow(10,4); //Q15
        long A=0;
        int out;

        //A=B*C*D+E
        //B*C   A=(long)B*C; //Q30
        A >>= 15; //Q15 (the lowest 16-bit contain the most significant bits of result B*C)

        //B*C*D
        A = (long)D * (short)A; //lowest 16-bit of A are used to be multiplied by 16-bit in C
                                //->result in Q30

        //B*C*D+E
        A+=((long)E<<15); //A (Q30) = A (Q30) + E<<15 (Q30)


        out = (int) (A>>15); //after shift lower 16 bit are in Q15 and moved to out

}
```

The C-compiler generates the following ASM code:

```
MOV *SP(#01h),T1
MPYM *SP(#00h),T1,AC0
MOV AC0,dbl(*SP(#04h))
```

The variables B,C,D,E where stored on the stack using stack pointer SP.e.g. B is stored at *SP, *SP(#01h) takes value at address SP + 1 (C), , …. Main remark is that MPYM uses two 16-bit registers and stores result in AC0, which is desired! Next the result in AC0 is stored in two consecutive 16-bit registers SP(#04h) and SP(#05h). This can of course be optimized when we think of our own ASM implementation.

The C-compiler generates the following ASM code:

```
MOV *SP(#05h),T1
MPYM *SP(#02h),T1,AC0
MOV AC0,dbl(*SP(#04h))
```

Lower 16-bit part of A stored at SP(#05h) is multiplied by value stored at SP(#02h) which contains D in this cases. The result of the multiplication is stored in AC0…

The C-compiler generates the following ASM code:

```
MOV dbl(*SP(#04h)),AC0
MOV *SP(#03h),AC1
ADD AC1 << #15,AC0
MOV AC0,dbl(*SP(#04h))
```

A (32-bit) is stored in AC0. E is moved to lowest 16-bit of AC1. Then AC1 is shifted 15 bits to the left and then added to AC0. AC0 contains the result and this result is placed in memory.

The C-compiler generates the following ASM code:

```
MOV dbl(*SP(#04h)),AC0
SFTS AC0,#-15,AC0
MOV AC0,*SP(#06h)
```

Note on assembler code: always check your instructions in the Mnenomics instruction set manual (SPRU374G)! E.g.

```
MPYM    *AR1,AC0
```

this multiplication instruction multiplies the value stored at in memory at the address stored in AR1 with the value stored at the bit 16 to 31 (LSB numbered as 0) in AC0.