1. Write assembly functions that implement the following C functions:

   a. float sumF32(const float x[], uint32_t count)
      // returns the sum of the elements in an array (x) containing count entries

   b. double prodF64(const double x[], uint32_t count)
      // returns the product of the elements in an array (x) containing count entries

   c. double dotpF64(const double x[], const double y[],uint32_t count)
      // returns the dot product of two arrays (x and y) containing count entries

   d. float maxF32(const float x[], uint32_t count)
      // returns the maximum value in the array (x) containing count entries

2. For the following code, calculate the number of instruction cycles required to execute the following code, using the simplified pipeline timing rules in class, including the time to call this function with BL bro8 and the time to return from the function with BX LR. You can assume that the pipeline is full before the BL bro8 instruction is executed.

`bro8:`	Clocks	Iterations			
`MOV R1, R0`	1	1	1 x 1	=	1
`MOV R0, #0`	1	1	1 x 1	=	1
`MOV R2, #0x00000080`	1	1	1 x 1	=	1
`MOV R3, #0x00000001`	1	1	1 x 1	=	1
`bro8_loop:`					
`TST R1, R2`	1	8	1 x 8	=	1
`ORRNE R0, R3`	1	8	1 x 8	=	1
`MOVS R2, R2, LSR #1`	1	8	1 x 8	=	1
`MOV R3, R3, LSL #1`	1	8	1 x 8	=	1
`BNE bro8_loop`	3 or 1	7+1	(7x3)+(1x1)	= 22	
`BX LR`	2	1	2 x 1	=	2
			Total	=	60

Clock cycles: 60 Clock Cycles

If the clock rate is 2 GHz, what is the execution time in nanoseconds? 30 ns

The only thing that affects the execution path is:
BNE bro8_loop
which depends on the status of R2
R2 is modified by: MOVS R2, R2, LSR #1

So... converting R2 to binary from Hexadecimal looks like this:

0x0	0	0	0	0	0	8	0
0000	0000	0000	0000	0000	0000	1000	0000

It will take 8 LSR before the only bit is shifted out of the register R2.
This means the loop will be taken 8 times because the branch condition will be true 7 times and on the last time the condition will be false

Clock cycles = 60
2 GHz means 2 cycles/nanosecond so

$$\frac{60\ \text{cycles}}{} \times \frac{\text{nanoseconds}}{2\ \text{cycles}} = 30\ \text{nanoseconds}$$

3. Assume SP = 0x20001034 before the following instructions are executed:

Address   Instruction

```
10000000:       BL fn
           fn:

10001000:       MOV R0, #8192
10001004:       MOV R1, #0x10000000
10001008:       MOV R2, #0x7400
1000100C:       PUSH {R0, R1, R2, LR}
           loop:
10001010:       B loop
```

LR = 10000004

After this program enters the endless loop:

What is the value of the SP? 0x20001024

Assuming the processor uses little-endian convention, what is the value of the following memory locations (place X in the blank if there is not enough information):

Address	8-bit Data
0x2000103B	x
0x2000103A	x
0x20001039	x
0x20001038	x
0x20001037	x
0x20001036	x
0x20001035	x
0x20001034	x
0x20001033	0x10
0x20001032	0x00
0x20001031	0x00
0x20001030	0x04
0x2000102F	0x00
0x2000102E	0x00
0x2000102D	0x74
0x2000102C	0x00
0x2000102B	0x10
0x2000102A	0x00
0x20001029	0x00
0x20001028	0x00
0x20001027	0x00
0x20001026	0x00
0x20001025	0x81
0x20001024	0x92

0x20001033 -> 0x20001030 ----> LR
0x2000102F -> 0x2000102C ----> R2
0x2000102B -> 0x20001028 ----> R1
0x20001027 -> 0x20001024 ----> R0

4. Explain the concept of memory virtualization, including the concept of paging and fragmentation. Also explain the role of virtualization in memory protection between running processes ("programs").

Memory virtualization: it is a technique that gives an application program the impression that it has its own contiguous logical memory independent of available physical memory . it used to simulate more memory than it is physically available in a machine.

Paging: Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging.

fragmentation: As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

5. Explain the concept of cache, including the principle of locality. Explain how this can speed up memory accesses.

Cache: a cache is a high-speed data storage layer which stores a subset of data, typically transient in nature, so that future requests for that data are served up faster than is possible by accessing the data’s primary storage location.

Principle of locality:Principle of Locality is the tendency of a processor to access the same set of memory locations repetitively over a short period of time.

Cache memory holds frequently used instructions/data which the processor may require next and it is faster access memory than RAM, since it is on the same chip as the processor. This reduces the need for frequent slower memory retrievals from main memory, which may otherwise keep the CPU waiting. In this way it can speed up memory accesses