Solver: Jason Lim Kian Voon

1)

a)

	R0	SR
i)	0x800	0x004
ii)	0x000	0x002
iii)	0x111	0x000
iv)	0xC8B	0x005
v)	0x429	0x009

b)

i)

	R1	R2	[0x101]	SR
MOV R1, #16	0x010			
MOV R2, #16		0x010		
CMP R1, [0x100]				0x004
JGT Done	(JGT = Jump if SR Z=0 and V=N) N =/= V, Don't jump			
ADD R1, #8	0x018			
SUB R2, #1		0x00F		
MOV [0x101], R1			0x018	
CMP R1, [0x100]				0x004
JGT Done	N =/= V, Don't jump			
ADD R1, #8	0x020			
SUB R2, #1		0x00E		
MOV [0x101], R1			0x020	
CMP R1, [0x100]				0x004
JGT Done	N =/= V, Don't jump			
ADD R1, #8	0x028			
SUB R2, #1		0x00D		
MOV [0x101], R1			0x028	
CMP R1, [0x100]				0x003
JGT Done	Z = 1, Don't jump			
ADD R1, #8	0x030			
SUB R2, #1		0x00C		
MOV [0x101], R1			0x030	
CMP R1, [0x100]				0x001
JGT Done	Z = 0, N = V = 0, Jump to Done			
PSH R2	**0x030**	**0x00C**	**0x030**	**0x001**

ii) There is 2 memory access for each loop (CMP R1, [0x100] and MOV [0x101], R1)
Figure Q1b is looped for 4 times. 1 more memory access for the CMP R1, [0x100] before jumping to Done. Total memory access cycles = $2 \times 4 + 1 = 9$

iii)
```
Start   MOVS R0, #8
        MOV R1, #16
        MOV R2, R1
        MOV R3, 0x100
Loop    CMP R1, [R3+0]
        JGT Done
        ADD R1, R0
        DEC R2
        MOV [R3+1], R1
        JMP Loop
Done    PSH R2
```

2)

a)

I1	PSH [0x100]
I2	MOV [0x101], R0
I3	INC SP
I4	PSHM 14
I5	MOV R1, [SP+5]
I6	DEC R2
I7	JNE Loop
I8	POPM 14
I9	RET

b) It is counting the number of times the binary format of N1 changes from 1 to 0 or 0 to 1.
Includes the first check of the rightmost character, with the initial comparison against 0.
0x031 = 0000 0011 0001, therefore R0 = 4

c)
```
if (Var3 != 1) {
        Var0 += 1;
        Var3 = 1;
}
```

d) If PC = 0x100. What is in memory address 0x100 will be treated as an instruction.
First instruction will be 0x031: Represents MOV R3, R1
Next instruction will be 0xBFE: Represents JMP -2 (At this point PC will be 0x102)
After -2, PC will return to 0x100 which is MOV R3, R1 again and be trapped in an infinite loop.

3)

a)

i) Android and Smart Home Application have already taken up 10GB out of 16GB.
More storage may be needed for the user for additional applications.

ii) NAND Flash as it is able to accommodate the slim 7mm profile. HDD is not suitable.

iii) Because NOR flash can support execute-in-place while NAND flash does not.

iv) To allow 1Mbyte to be reserved as cache for important/frequently accessed data.

b)

i) It would receive some non-ASCII characters as well with the mismatch in baud rate of the UART links. But sometimes (though rare) it may receive ASCII characters too.

ii) A send to B at 28800 bps while B to C is only at 2400 bps, 28800/2400 = 12 times slower.
To send a 1xxx xxxx xxxx (x can be any bit), since B to C is slower, in the time taken for B to receive 12 bits, C only receives a single bit, which is 1 in this case.
C send to D at 28800 bps again but instead of receiving what A sent to B, D is receiving from C ‘1’ and it’s faster by 12 times, so D receives 12 times of the ‘1’ instead.
D then finally sends back to A at 6 times slower (28800 / 4800 = 6).

c) Since cache uses Virtual Address for address mapping as stated in the question:
For direct mapped cache,
16 cache blocks -> 4 bits for block index
Block size of 16 bytes -> 4 bits for offset
Virtual Memory Size of 1Mbyte -> 20 bits for address

0x00119 =	0000	0000	0001	0001	1001
		tag		block	offset

Cache Block = 2nd Block (Block 0001) and Tag Value = 0x001

d)

i) I3 and I4, Data Conflict as I3 has yet to update the value of R0 before I4 uses it
I4 and I5, Data Conflict as I4 has yet to update the value of R2 yet before I5 uses it
I6 Branch Delay as I7 and I8 will run before looping if there is a loop
I8 and I9, Data Conflict as I8 has yet to update the value of R0 before I9 uses it

ii) Insert NOP instructions between dependent instructions.
Branch delay between I6 and I7, can also use delayed branching/dynamic branch prediction.

iii) 3 cycles are saved

4)

a) B
b) E
c) B
d) D
e) B
f) C
g) C
h) A
i) C
j) E

--End of Answers--