1a) No, number of instructions executed per second is not a reliable way of comparing performance. This is for the following reasons:

- There might be some instructions which a processor would take more time or cycles to execute, compared to another one. A program largely comprising of these instructions would be slower on a processor which may have a higher number of instructions executed per second compared to another one with a lower number of instructions executed per second.
- The instruction set of a program may convert a program into a large number of instructions. With a high enough number of instructions, the processor could take more time to finish running a program compared to another one with a lower number of instructions executed per second.
- The clock period (cycle time), and thus the clock rate is not clear with the difference in number of instructions executed per second. A processor may have a lower cycle time (or higher clock rate), but due to a higher CPI (cycles per instruction), have a lower number of instructions executed per second.

There are benchmark tests (such as SPEC or EEMBC) which help in distinguishing between processor performances. These should be used, while keeping in mind the pros and cons of each test.

b) Let old execution time be T_{old}

~~Let~~ $T_{new} = (1-f)\,T_{old} + \dfrac{f\,T_{old}}{n}$

$= \dfrac{(1-f)\,n\,T_{old} + f\,T_{old}}{n}$

~~$= (1-f)T_{old} + T$~~

$= \dfrac{(1-f)n + f}{n}\; T_{old}$

Speedup $= \dfrac{T_{old}}{T_{new}}$

$= T_{old} \times \dfrac{n}{((1-f)n + f)\,T_{old}}$

$= \dfrac{n}{n - fn + f}$

c) Given :- c = speedup

$c = \dfrac{n}{n - fn + f}$

$\Rightarrow$ ~~$n(1-$~~ $n - fn + f = \dfrac{n}{c}$

$\Rightarrow f - fn = \dfrac{n}{c} - n$

$\Rightarrow f(1-n) = \dfrac{n - nc}{c}$

$\Rightarrow f = \dfrac{n - nc}{c - nc}$

d) Let number of months $= x$

Given :- $x m T_{old} = T_{new}$

$x m T_{old} = \frac{(1-f)n + f}{n} T_{old}$

$x = \frac{(1-f)n + f}{mn}$ months

e) Let old time $= T_{old}$

New time $= T_{new} = (1-f-g)T_{old} + \frac{f T_{old}}{n} + \frac{g T_{old}}{k}$

$= \frac{(1-f-g)nk + fk + ng}{nk} T_{old}$

Speedup $= \frac{T_{\cancel{new}\,old}}{T_{new}}$

$= \frac{T_{old} \times nk}{[(1-f-g)nk + fk + ng] T_{old}}$

$= \frac{nk}{(1-f-g)nk + fk + ng}$

2a) struct d1 →

- short i →

  offset = 0 bytes

  size = 2 bytes

- long j →

  Offset = 6 bytes

  size = 8 bytes

- int *k →

  Offset = 0 bytes

  size = 8 bytes

- short *l →

  Offset = 0 bytes

  size = 8 bytes

Total size of d1 = 32 bytes
Overall alignment = 8

struct d2 →
- char a[3] →
  offset = 0 bytes
  size = 3 bytes
- char *b[4] →
  offset = 5 bytes
  size = ~~8 bytes~~ 32 bytes

Overall size = ~~16~~ 40 bytes
Overall alignment = 8

struct d3 →
~~• ch~~
- struct d1 c →
  offset = 0 bytes
  size = 32 bytes
- struct d2 e[2] →
  offset = 0 bytes
  size = 80 bytes

Overall size = 112 bytes
Overall alignment = 8

i) ~~leaq~~ MOVq 16(%rdi), %rax
ii) leaq 32(%rdi, %rsi, 40), %rax

2b)
```
long sum(long *init, int seed) {
    int result = randomise(seed, init);
    int count = 0;
                 (result != 0 || count > 0)                    -count
    while (~~result~~ != 0 && init != (init + result * 8N))
    {
        result = count == 0 ? *init : result + *init;
        init += 8;
        ~~so~~ count += 8;
    }
    return count == 0 ? 0 : result;
}
```

c) $C/B = 8 = 2^3$ blocks

$N = 2$

Index bits $= \log_2(2^3/2)$

$= 2$ bits

$B = 2^2$ beytes

$O = \log_2(2^2) = 2$ bits

Tag bits $= 12 - 2 - 2 = 8$ bits

0x42B :-

Binary $=$ 0~~1~~00 0010 $\underline{10}\,\underline{11}_2$

$O = 11_2 = 0x3$

$I = 10_2 = 0x2$

$T = 0x42$

HIT! Byte returned :- AB

0x474 :-

Binary $=$ 0100 0111 0100_2

$O = 0x0$

$I = 0x1$

$T = 0x47$

Valid bit 0, MISS!