1. $f = 0.80$, Speedup = 1.5, $S = ?$

a) Speed up $= \frac{1}{(1-f) + \frac{f}{S}}$ $= 1.5 = \frac{1}{(1-.80) + (\frac{0.8}{S})}$

$= 1.5 = \frac{1}{(0.2) + (\frac{0.8}{S})}$ $\Rightarrow$ $(1.5)((0.2) + \frac{0.8}{S}) = 1$

$= 0.3 + \frac{1.2}{S} = 1$ $\Rightarrow$ ~~$S = 0.7 + 1.2 = 1.2$~~ $\frac{1.2}{S} = 0.7 \Rightarrow 1.2 = 0.7S$

$\boxed{S = 1.7}$ ✓

b) $\frac{f}{S} = 0$, Speedup $= \frac{1}{(1-f) + \frac{f}{S}} \Rightarrow \frac{1}{(1-f)}$

$= \frac{1}{1-0.8} = \frac{1}{0.2} = \boxed{5}$ ✓

2. Failure system $= \lambda = \frac{1}{t} = t = 8760$

$\Rightarrow 4 \cdot \frac{1}{8760} = \frac{4}{8760} = \frac{1}{2190} =$

MTTF system $= \frac{1}{\text{Failure rate}} = \frac{1}{(\frac{4}{8760})} = \boxed{2190 \text{ hours}}$ ✓

3. a) $CPI = \sum_{i \in \text{inst set}} IC_i \times CPI_i$

$= ((0.2) + (0.4 \cdot 5)(0.2 \cdot 3)(0.2 \cdot 2)) = (0.2) + (2) + (0.6) + (0.4)$

$= \boxed{3.2}$ ✓

b) Speedup $= \frac{\text{Execution Time}_O}{\text{Executive Time}_B} = \frac{IC \cdot CT \cdot CPI_O}{IC \cdot ((CT(0.1) \cdot CT) \cdot CPI_B}$

$= \frac{CT \cdot CPI_O}{CPI_B (CT)(0.1) + CPI_B)(CT)} = \frac{CPI_O}{CPI_B(1.10)}$

$CPI_B = ((0.2) + (0.4 \cdot 2) + (0.2 \cdot 3) + (0.2 \cdot 2) = \boxed{12}$ ✓

$\Rightarrow \frac{3.2}{(2)(1.10)} = \boxed{1.45}$ ✓

5.

```
1. segment .data
2. sayi db 1
3. segment .text
4. global _start
5. _start:
6. mov ecx , sayi
7. inc ecx
8. mov [sayi] , ecx
9. mov eax , 4
10. mov ebx , 1
11. mov ecx , sayi
12. mov edx , 4
13. int 0x80
14. mov eax , 1
15. mov ebx , 0
16. int 0x80
```

6.

A) The following LDIR instruction copies from HL to DE and the number of bytes from BC

```
LD HL, &0000
LD DE, &C000
LD BC, &4000
LDIR
RET
```

B) The following instructions are very similar but move the starting address at 0xA000 to 0xB000 and store it in BC

```
LD HL, &0xa000
LD DE, &0xb000
LD BC, &0xc000
LDIR
RET
```

C) LDIR was introduced to bridge the gap between assembly and high-level languages, because it provided a way to directly load data from the memory into the registers. It also compatible with RISC design principles and makes coding efficient and easy to read.

Sources:

https://landley.net/history/mirror/cpm/z80.html
https://en.wikipedia.org/wiki/Zilog_Z80