Q1

- virtual memory space: $2^{32}\ bytes$
- physical memory space: $2^{18}\ bytes$
- page size: 4096 bytes = $2^{12}\ bytes$
- virtual address generated by process: 0x 1112 3456

General paging algorithm:
In a paging environment, logical memory is divided into fixed size 'pages' and physical memory is divided into fixed sized 'frames'. Frame size = page size = $2^{12}\ bytes$.

1. Given a logical address generated by the CPU, hardware determines the logical page of this logical address and at what offset.
2. The hardware uses the page number as an index into the page table. The page table contains the base address of the frame in physical memory
3. Add the offset and base address of frame to derive the complete physical address

Software operations

- **Generate logical address:** The CPU generates the logical (virtual) address from a user process: $0x\ 1112\ 3456$

Hardware operations

- **Determine page number and page offset of logical address**: (logical address = page number + page offset)
  - ➔ Given the logical address space size as $2^{32}\ bytes$ and page size as $2^{12}\ bytes$
    - ▪ Page number = the high order 32 -12 = 20 bits of the logical address
    - ▪ Page offset = the 12 low-order bits of the logical address

$$logical\ address = 0x\ 1112\ 3456$$
$$page\ number = 0x\ 11123$$
$$page\ offset = 0x\ 456$$

- **Derive base address of frame**
  - ➔ Use the page number as an index into a page table. The page table is implemented in hardware using dedicated registers or a translation Look-aside buffer and contains the base address of the target frame in physical memory
  - ➔ Suppose the page number $0x\ 11123$ returns us frame X from the page table

- **Derive complete physical address**
  - ➔ Add the base address of the target frame with the page offset to obtain the complete physical address
  - ➔ $Physical\ address = (X * page\ size) + offset$
    $= (X * 4096\ bytes) + 0x456$

Q2

- Page fault service time = 8ms if empty page is available or replaced page is not modified
  OR = 20ms if the replaced page was modified
- Memory access time = 100ns = 0.0001ms
- Replaced page is modified 70% of the time
- We want effective memory access time of no more than 200 ns (0.0002ms)

Let P = probability of a page fault
Let M = normal memory access time (no page fault): $100ns = 0.0001ms$

If 70% of the time the page fault service time is 20ms and 8ms otherwise, then
The effective page fault service time is

$$E_{pf} = 20ms * 70\% + 8ms * 30\% = 16.4ms$$

The effective memory access time is

$$\begin{aligned} E_{ma} &= P * (page\ fault\ service\ time) + (1 - P) * M, \quad page\ fault\ service\ time = E_{pf} \\ &= (P * 16.4ms) + (1 - P) * 0.0001ms \\ &= 16.3999ms * P + 0.0001ms \end{aligned}$$

Solve for the maximum acceptable page-fault rate such that the effective access time of no more than 0.0002ms

$$0.0002ms > 16.3999ms * P + 0.0001ms$$
$$P < 0.000006$$

Therefore, we can allow fewer than one memory access out of 166,667 to page fault