Q1 (a).

1. Mutual exclusion: only one vehicle can occupy an intersection at a time. For example, if two vehicles approach an intersection, and both want to pass through the road, only one vehicle can pass through since the intersection only allows one vehicle to pass at a time.
2. Hold and wait: the vehicle occupies an intersection and needs to wait for additional resources. For example, the vehicle in an intersection has to wait for another intersection to clear before it can proceed.
3. No preemption: an intersection can be cleared only voluntarily by the vehicle occupying it. For example, once a vehicle occupies an intersection, it cannot be forced to move aside for another vehicle to occupy.
4. Circular wait: there exists a set of waiting vehicles that are waiting for an intersection to clear that is occupied by the next vehicle, so then they can proceed. For example, vehicle A is waiting for an intersection occupied by vehicle B, vehicle B is waiting for vehicle C, and vehicle D is waiting for vehicle A.

Q1 (b).

Implement a Strict One-Way Traffic Rule: where vehicles can only move in one direction at a time within the intersection. For example, all the vehicles in an intersection have to move left/right. As a result, this can eliminate the circular wait condition by ensuring that vehicles do not need to wait for each other in a circular chain because they are in the same direction.

Q2 (a).

Need

Thread	A	B	C	D
P0	0	0	1	0
P1	0	8	5	0
P2	1	0	1	1
P3	0	1	3	0
P4	0	3	5	3

Available

Thread	A	B	C	D
	3	5	3	1
P3	0	1	3	0
	3	4	0	1
+	0	6	5	2
	3	10	5	3
P1	0	8	5	0
	3	2	0	3
+	2	8	5	0
	5	10	5	3
P4	0	3	5	3
	5	7	0	0
+	0	3	6	6
	5	10	6	6
P2	1	0	1	1
	4	10	5	5
+	2	2	3	6
	6	12	8	11
P0	0	0	1	0
	6	12	7	11
+	4	0	1	3
	10	12	8	14

A has 10 resources.

B has 12 resources.

C has 8 resources.

D has 14 resources.

Q2 (b).

Yes, the system is in a safe state with a safe sequence of $\{P_3 \rightarrow P_1 \rightarrow P_4 \rightarrow P_2 \rightarrow P_0\}$

Q2 (c).

Available

A	B	C	D
3	5	3	1

Request from P_1 (0, 6, 4, 0)

1. Request $\leqslant$ P_1 Need (0, 8, 5, 0)
2. Request $\geqslant$ Available (3, 5, 3, 1)
3. Then P_1 has to wait until the Available $\geqslant$ Request

Q3 (a).

If the page number is the TLB, then it takes 260 nanoseconds.

Otherwise, it takes 260 × 2 = 520 nanoseconds. Besides, we can let $T_{\text{page\_table}}$ be the time to access the page table, therefore it takes total $T_{\text{page\_table}}$ + 260 nanoseconds.

Q3 (b).

$$\begin{aligned}\text{EAT} &= 0.8 \times 260 + 0.2 \times 260 \times 2 \\ &= 312 \text{ nanoseconds}\end{aligned}$$

Q4 (a).

Virtual address: $2^9 = 512$ for VPN | offset 2^5 (page size 32)

Physical address: $2^7 = 128$ → 2^2 for PFN | offset 2^5 (page size 32)

Format of PTE: valid | physical page number

Translate workflow: look at VPN → virtual page number → look up the page table → see if it is valid or invalid → look for the physical page number → add it onto the offset

Therefore, the effective range of page table size is 16384 bits and in this design, the process can use 16384 bits.

Q4 (b).

The page table design is effective since it allows the process to address a 16KB virtual address space with a page size of 32 bytes

To enhance the page table, adopting a multi-level page table can effectively minimise its overall size. For instance, incorporating a hierarchical structure significantly improves the efficiency of the page table. Specifically, introducing a page directory entry (PDE) serves as an index for accessing the second level, where the page-table entries (PTE) are located. Additionally, enhancing the structure involves including valid bits for both PDE and PTE, positioned as the leftmost bit. Also, the VPN should split into three parts, PDE, PTE and offset.

Q5 (a).

page frames	1	3	2	4	4	2	5	1	6	7	8	7	8	9	7	8	9	4	5	4	5	3
	1	1	1	1			1		6	7	7			7					5			5
		3	3	3			3		3	3	8			8					8			3
			2	2			5		5	5	5			9					9			9
				4			4		4	4	4			4					4			4

The total number of page faults is 11.

Q5 (b).

page frames	1	3	2	4	4	2	5	1	6	7	8	7	8	9	7	8	9	4	5	4	5	3
	1	1	1	1			5	5	5	5	8			8				8	8			3
		3	3	3			3	1	1	1	1			9				9	9			9
			2	2			2	2	2	7	7			7				7	5			5
				4			4	4	6	6	6			6				4	4			4

The total number of page faults is 13.