CS-1216 – Monsoon 2023 – Assignment 3

Rutam Kathale, rutam.kathale@gmail.com

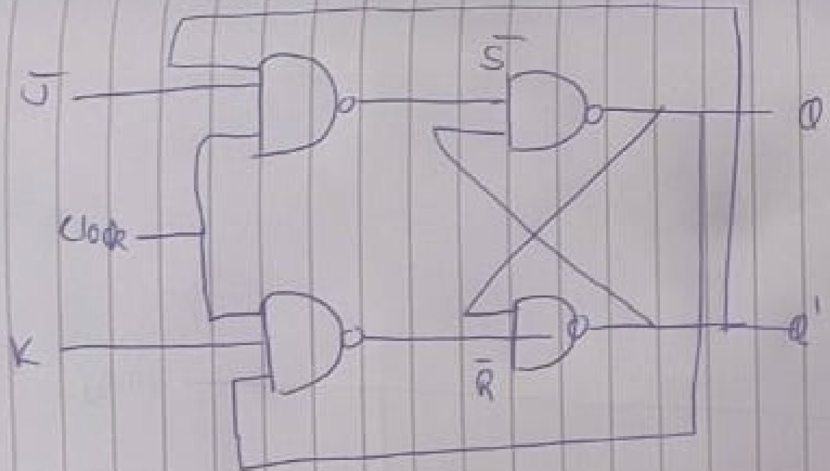
November 5, 2023

Problem 1. What is the difference between a J-K latch and a J-K flip-flop? What are the two ways to implement a J-K flip-flop?

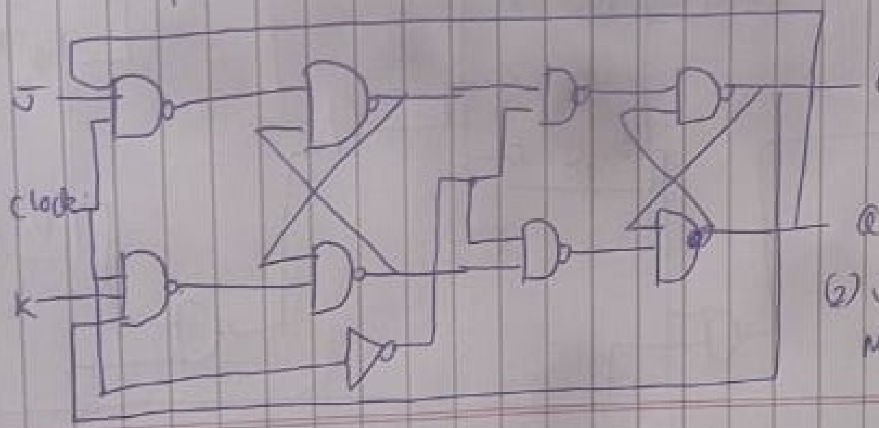
A J-K latch works in real-time based on inputs. It is 'level-triggered' and so changes the output based on input (it can't hold a 'state' per se.). They are also not based on clock signal but mainly based on the inputs.
A J-K flip flop, in contrast changes outputs mainly based on clock signal. It is 'edge-triggered' and so, it changes based on signals going fro high->low or low->high meaning it's in synchronicity with the clock's edges. J-K flip-flops are itself based on latch design, except having an arrangement to accomodate for changes in clock signal.
The two ways of implementing J-K flip-flop is using an S-R flip-flop and using Master-slave configuration.
Note: Since these implementations are very standard, it would be similar to the implementations available online (implementations themselves online are similar across websites). I used this website mainly for an understanding of J-K flip-flop using S-R flip-flop and the class slides for J-K Flip-flop using Master-slave configuration.

Using S-R flip-flop, we just add the clock input (since it is synchronous) in addition to the usual implementation of an S-R latch. The S-R (Set-Reset) flip-flop has two inputs, S (Set) and R (Reset), and two outputs, Q and $\overline{Q}$ (also as a side note, I am using Q' and $\overline{Q}$ interchangibly but denoting the same thing). It consists of two NAND gates arranged such that the output of each NAND gate is connected to one of the inputs of the other. When a J-K flip-flop is implemented using an S-R flip-flop, race conditions can occur causing the flip-flop to set and reset states unpredictably causing an instability.
To mitigate this issue, another implementation of J-K flip flop involves a 'Master-slave' configuration. This implementation basically uses 2 S-R flip flops, connected together. One flip-flop activates on rising edge of clock pulse (Master S-R Flip-flop) while other activates on falling edge (Slave S-R Flip-flop). On the rising clock edge, master flip-flop activates and processes signals J and K but this doesn't immediately change the output since the slave flip-flop works on sort of inverted clock signals and is inactive in the above case. Only on the event of falling-edge, the data processed by the master flip-flop is forwarded to slave flip-flop and then the overall output changes. This implementation doesn't suffer with race conditions and is stable due to the use of inverted clock signal and employs two separate stages for processing the inputs and outputs. Due to this mechanism, output can only change after each clock cycle.

Both the implementations are attached on next page.



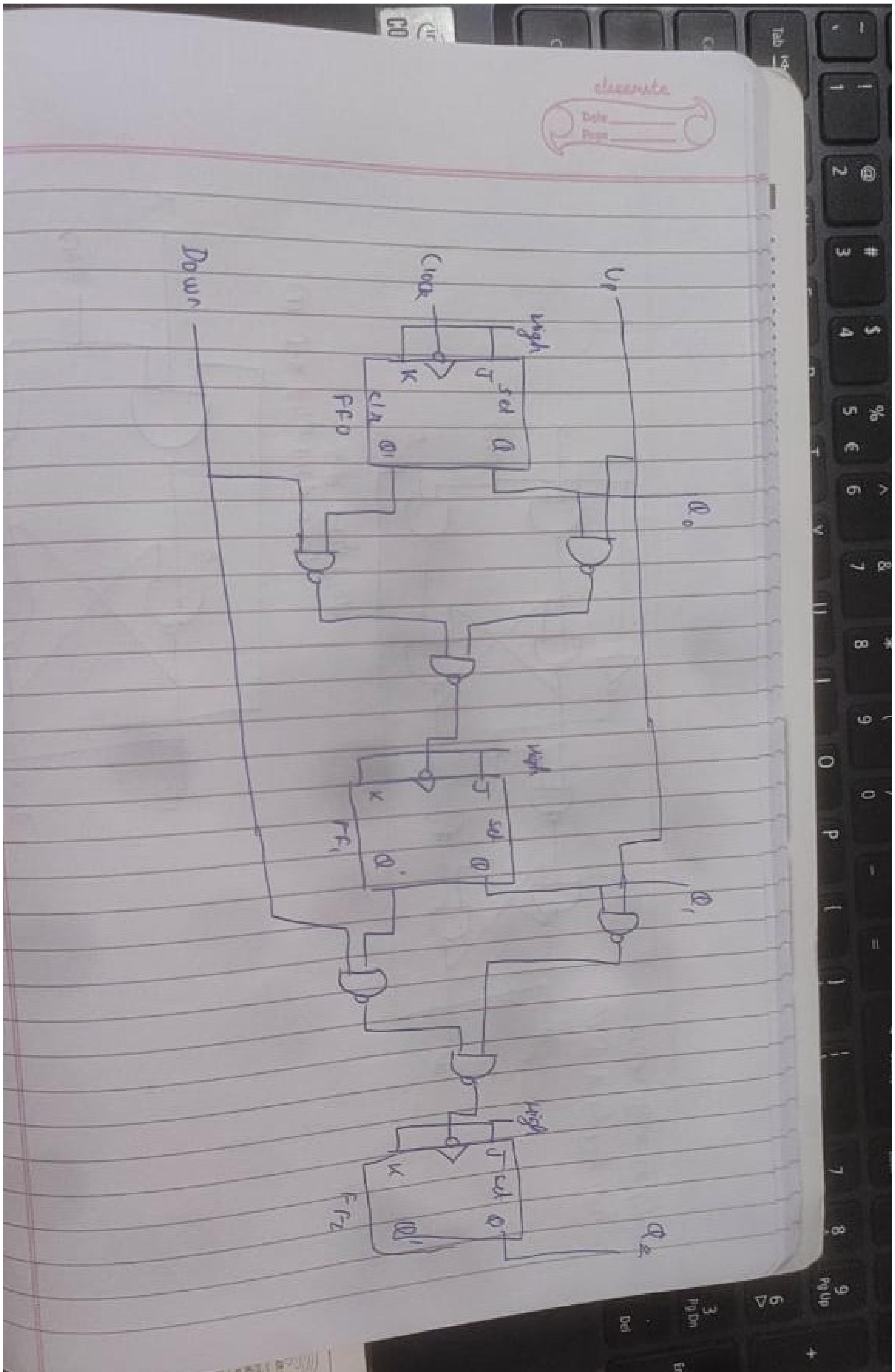
(1) J-K Flip Flop using S-R Flip-Flop



(2) J-K Flip Flop using Master Slave Configuration

Problem 2. Can you use a non-toggling mode flip-flop in asynchronous counter design? If yes, explain with a circuit diagram. If no, justify.

We can use a non-toggling mode flip-flop in asynchronous counter design. In an asynchronous counter, clock pulse can be given for first flip-flop and we can simulate next flip-flops to have the input as output of this flip-flop.
Output of each flip-flop is used as the clock input for the next flip-flop in the sequence. First flip-flop's clock signal is external, while the clock signal of the next flip-flops is dependent on the previous one. Ripple Effect is created if the output of the first flip-flop activates the next flip-flops (since its change is a change in clock signal for subsequent flip-flops. By using this configuration, the ouput of the counter be the output of the flip-flops from LSB to MSB. The circuit diagram for the same is attached in next page (This diagram is taken from the slides - L13, Page 12).



Up
Clock
Down
High
J
K
sd
clr
Q
Q'
FF0
Q_0
FF1
Q_1
FF2
Q_2

Problem 3. Design a synchronous counter that generates the sequence: 0, 2, 4, 5, 7, 0, 2, 4, ... Show all the steps of your design.

So, our sequence is 0 -> 2 -> 4 -> 5 -> 7 ->0.
We would need 3 bits to represent all numbers in the sequence (mod 8).
So our state Diagram would look like:
000 (S1) -> 010 (S2) -> 100 (S3) -> 101 (S4) -> 111 (S5) -> 000 (S1).
We will use T Flip-flops to design this counter. Its excitation table is:

Q_n 1	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Based on that, our truth table will be:

Q_c	Q_b	Q_a	Q_c*	Q_b*	Q_a*	T_c	T_b	T_a
0	0	0	0	1	0	0	1	0
0	0	1	x	x	x	x	x	x
0	1	0	1	0	0	1	1	0
0	1	1	x	x	x	x	x	x
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	0	x	0	x	x	x	x
1	1	1	0	x	0	1	1	1

Then we draw k-maps for T_c, T_b, and T_a

T_c:

-	Q_c, Q_b	00	01	11	10
Q_a	-	-	-	-	-
0	-	-	1	x	-
1	-	-	x	1	-

T_b:

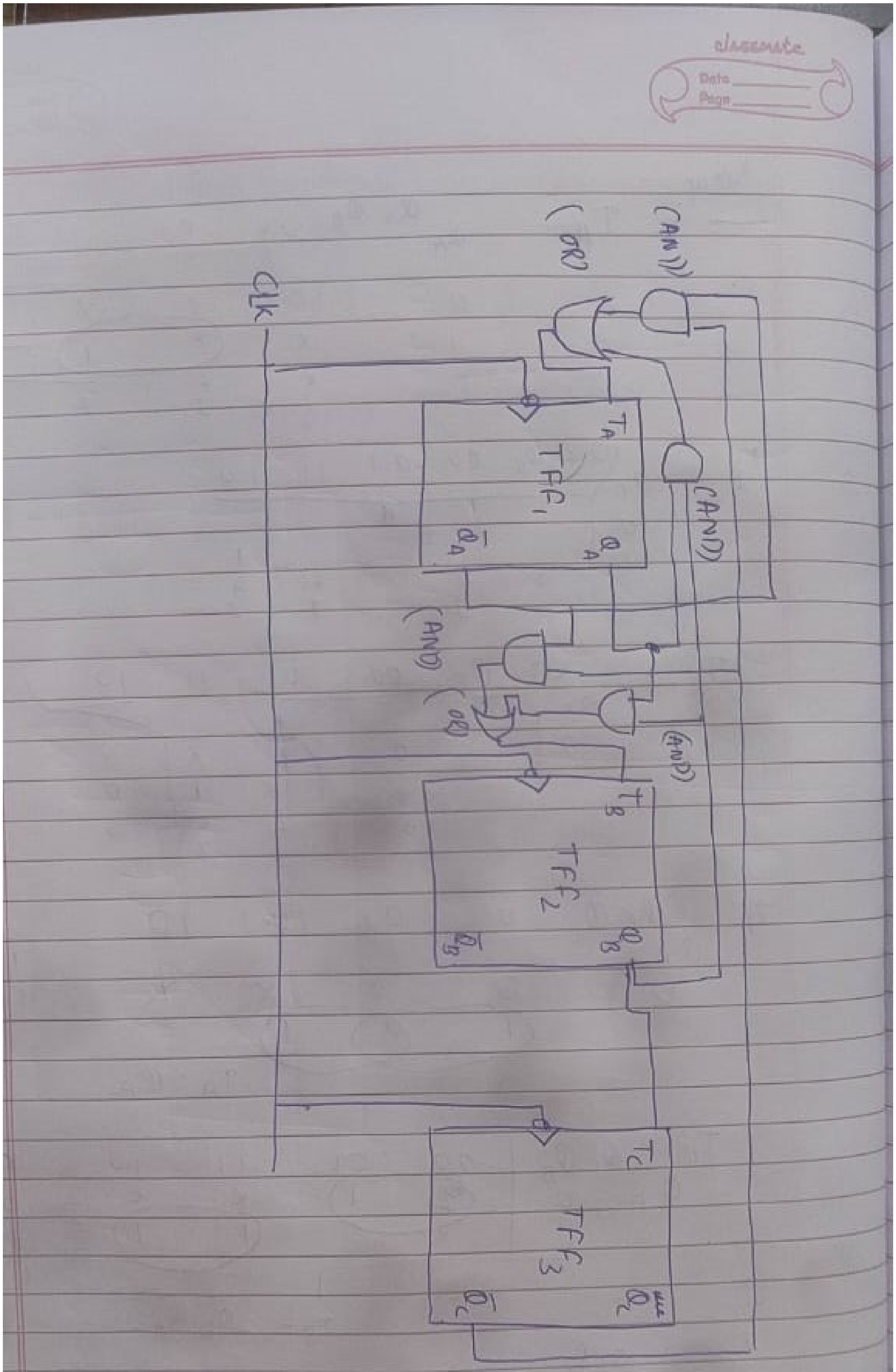
-	Q_c, Q_b	00	01	11	10
Q_a	-	-	-	-	-
0	-	1	1	-	-
1	-	-	-	1	1

T_a:

-	Q_c, Q_b	00	01	11	10
Q_a	-	-	-	-	-
0	-	-	-	x	1
1	-	-	x	1	0

So we select the quad from T_c, select group of horizontal ones in T_b, and select x and 1 horizontally on T_a.
So, we get the equations:
$T_c = Q_b$,

$T_b = \overline{Q_c Q_a} + Q_c Q_a,$
$T_a = \overline{Q_a} Q_c + Q_a Q_b$
Based on this, we can create the Logic Diagram (refer to next page).


(AND)
(OR)
Clk
T_A
TFF_1
Q_A
$\bar{Q}_A$
(AND)
(AND)
(OR)
(AND)
T_B
TFF_2
Q_B
$\bar{Q}_B$
T_C
TFF_3
Q_C
$\bar{Q}_C$

Problem 4. A block-set associative cache memory consists of 128 blocks divided into four-block sets. The main memory consists of 16,384 blocks and each block contains 256 eight-bit words. How many bits are required for addressing the main memory? How many bits are needed to represent the TAG, SET and WORD fields?

So, each block of main memory has 256 eight-bit words and main memory has 16,384 blocks in total. We know that SET bits are log (to base 2) of the number of blocks divided by the number of sets (from notes). We have 128 blocks in cache and 4 sets.
So total sets = 128/4 = 32
So, number of SET bits are
$\log_2(32) = 5$
We can obtain the WORD bits by just taking log of number of words in a block (in base 2).
Since we are given that each block has 256 eight-bit words, number of WORD bits:
$\log_2(256) = 8$ bits.
The remaining bits would constitute the TAG bits but for that, we first need to know how many bits do we need to address the main memory.
We have 16384 blocks of main memory and each block has 256 words. So, total words in main memory are: 16384 x 256 = 4194304.
So, the number of bits to address the main memory are log of total words in base 2 i.e.
$\log_2(4194304) = 22$
. So, we need 22 bits for main memory. Then, TAG bits become
22 - 8 - 5 = 9.
So, we need 9 bits for TAG field.

Problem 5. Consider a system having two level caches with hit ratios 0.8 in level 1 and 0.9 in level 2. Access times of level 1 cache, level 2 cache and main memory are 1 ns, 10 ns and 500 ns respectively. What is the average access time of the system?

We can calculate the AAT by calculating L1 hit time, L1 miss and L2 hit time, and L1 miss L2 miss time and add them together Miss ratio of L1 is 1 - Hit ratio that is 0.2 and similarly 0.1 for L2.
So, L1 Hit time = 0.8 x 1ns = 0.8ns
L1 Miss and L2 Hit Time = 0.2 x 0.9 x 11 ns= 1.98 ns
L1 miss and L2 miss: 0.2 x 0.1 s (1 ns + 10 ns + 500 ns) = 0.02 x 511 ns = 10.22 ns.
Adding all of them to get AAT = 0.8 + 1.98 + 10.22 = 13ns.
So the average access time is 13 ns.
NOTE: It can be slightly approximately different answer due to the way I am rounding off the answers in intermediate calculations.