**Microprocessor and Computer Architecture Laboratory**

**UE19CS256**

**4th Semester, Academic Year 2020-21**

|  |  |  |
| --- | --- | --- |
| Name: Tushar Y S | SRN: PES1UG19CS545 | Section: I |

Date:08/04/2021

Week#\_\_\_\_9\_\_\_\_\_\_\_Program Number: \_\_\_\_1\_\_\_

Title of the Program:

**5 Stage pipelined simulator**

Consider the following instructions. Execute these instructions using 5 stage pipeline - MIPS architecture simulator.

ADD R0, R1, R2

SUB R3, R0, R4

FP\_LOAD F1, Offset,R1

FP\_ADD F5,F1,F1

Observe the following and note down the results.

Check whether there is data dependency among the instructions?

-If yes, then, how many stall states have been introduced?

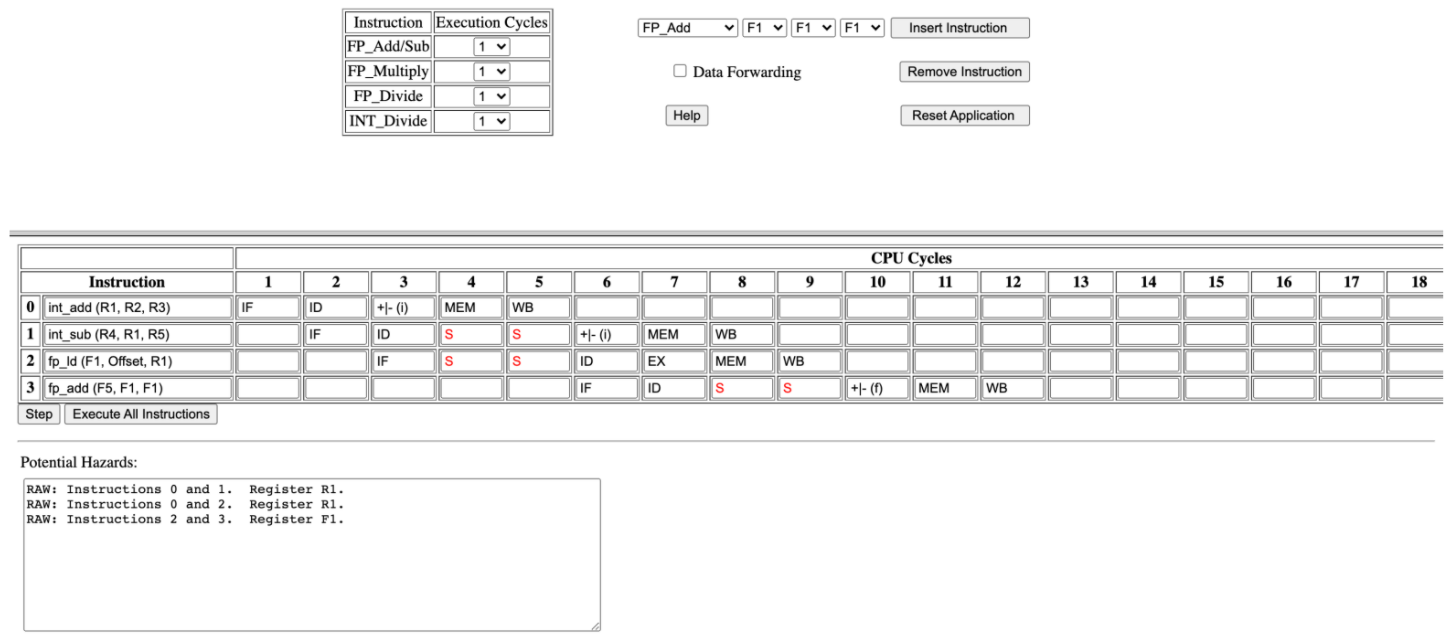
-If data forwarding is applied how many stall states have been reduced?

-Mention the total number of clock cycles used with and without data

forwarding.

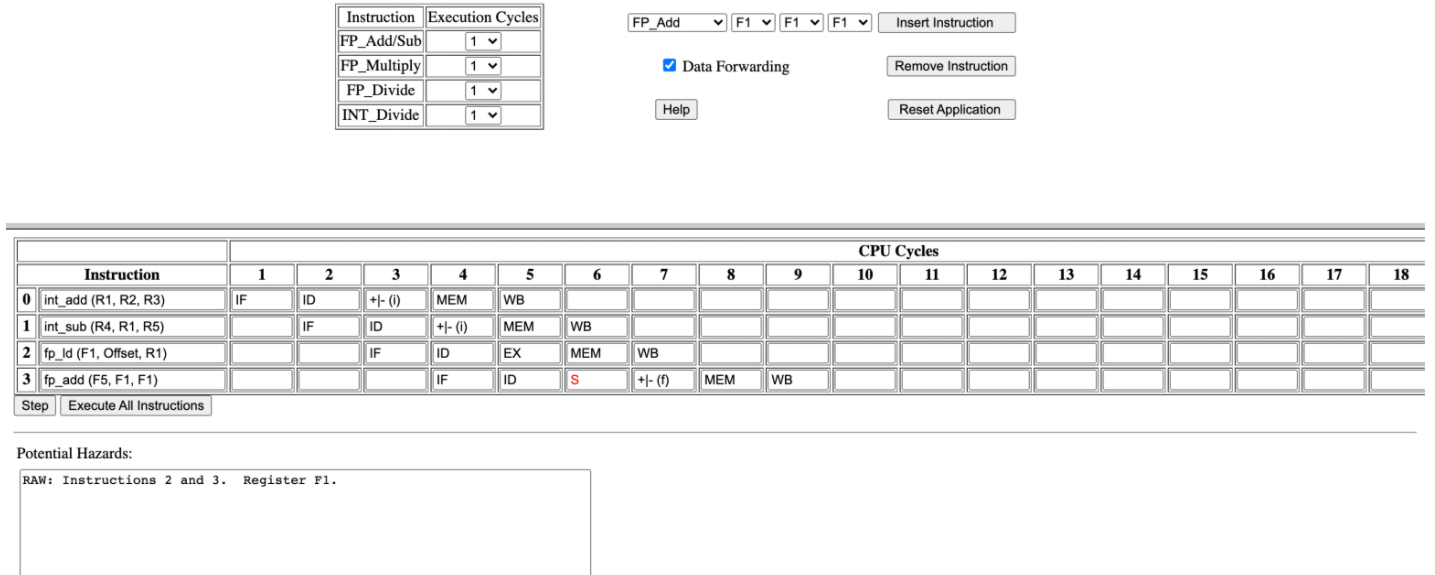
Observations:

Without data forwarding:



* Yes, there is a data dependency among the instructions.
* The number of stalls introduced in the case without data forwarding are 6.
* And the number of clock cycles consumed are 12.

With data forwarding:



* In case of data forwarding, only 1 stall is introduced.
* And the number of clock cycles consumed are 9.

Week#\_\_\_\_9\_\_\_\_\_\_\_Program Number: \_\_\_\_2\_\_\_

Title of the Program:

**Cache simulator**

**Part 1 – Direct mapping:**

1. A computer system uses 16-bit memory addresses. It has a 2K-byte cache

organized in a direct-mapped manner with 64 bytes per cache block.

Assume that the size of each memory word is 1 byte.

(a) Calculate the number of bits in each of the Tag, Block, and Word fields of

the memory address using direct mapped Cache.

(b) When a program is executed, the processor reads data sequentially from the

following word addresses:

128, 144, 2176, 2180, 128, 2176

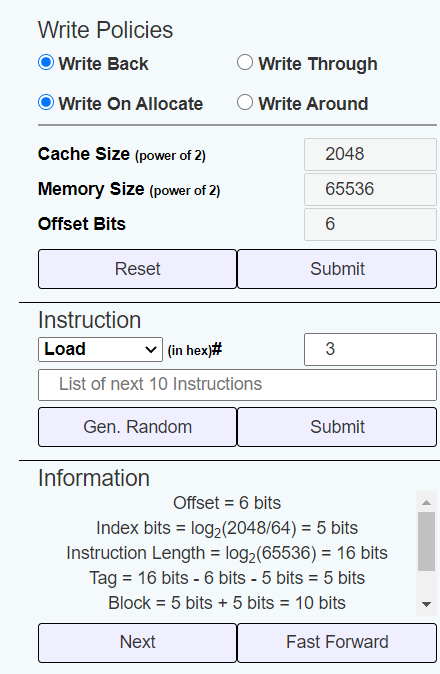
All the above addresses are shown in decimal values (Convert the address to

hexadecimal equivalent of the decimals given above and submit in the

simulator). Assume that the cache is initially empty.

For each of the above addresses, indicate whether the cache access will result in

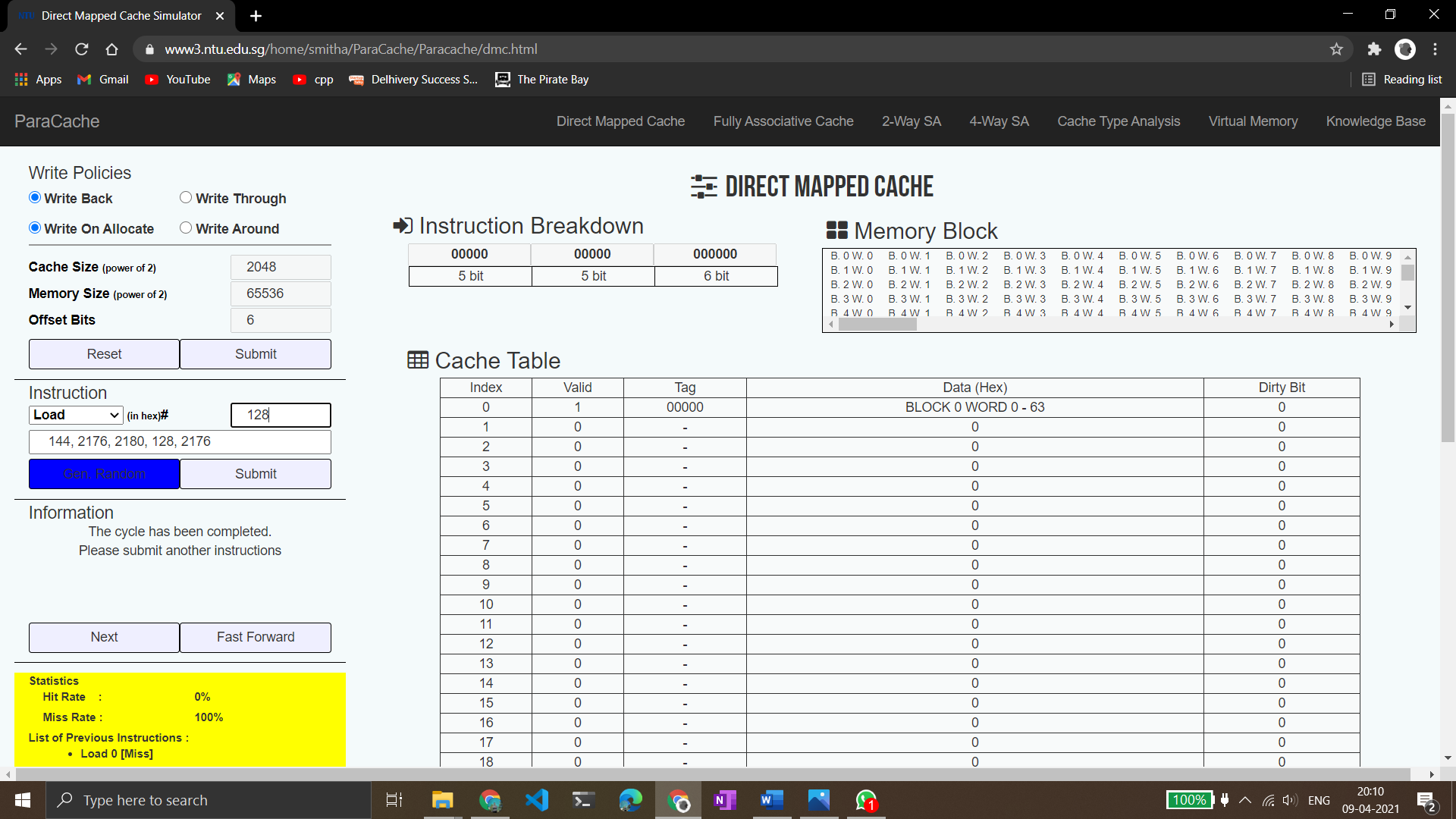
a hit or a miss.

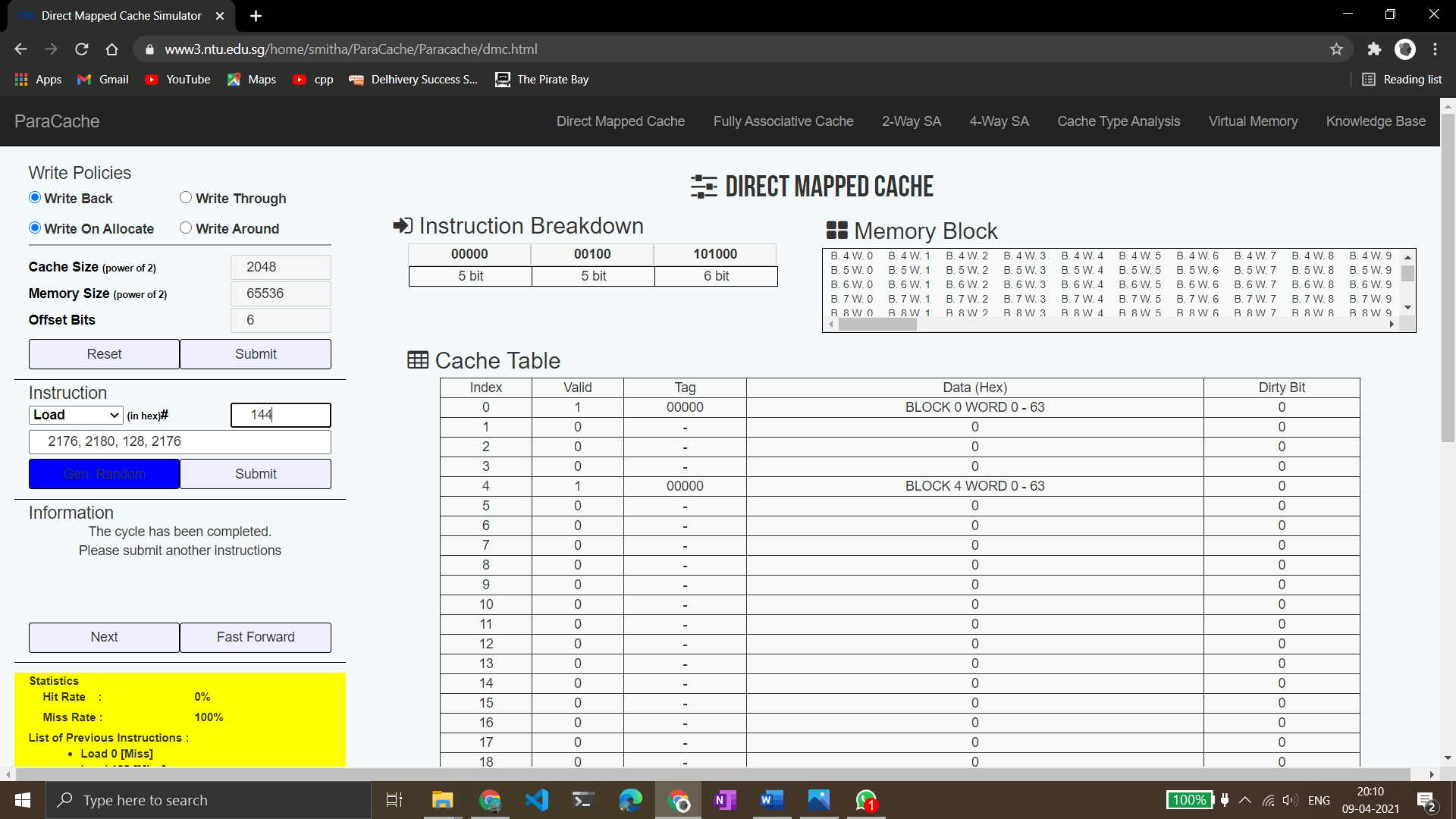


Based on the info given:

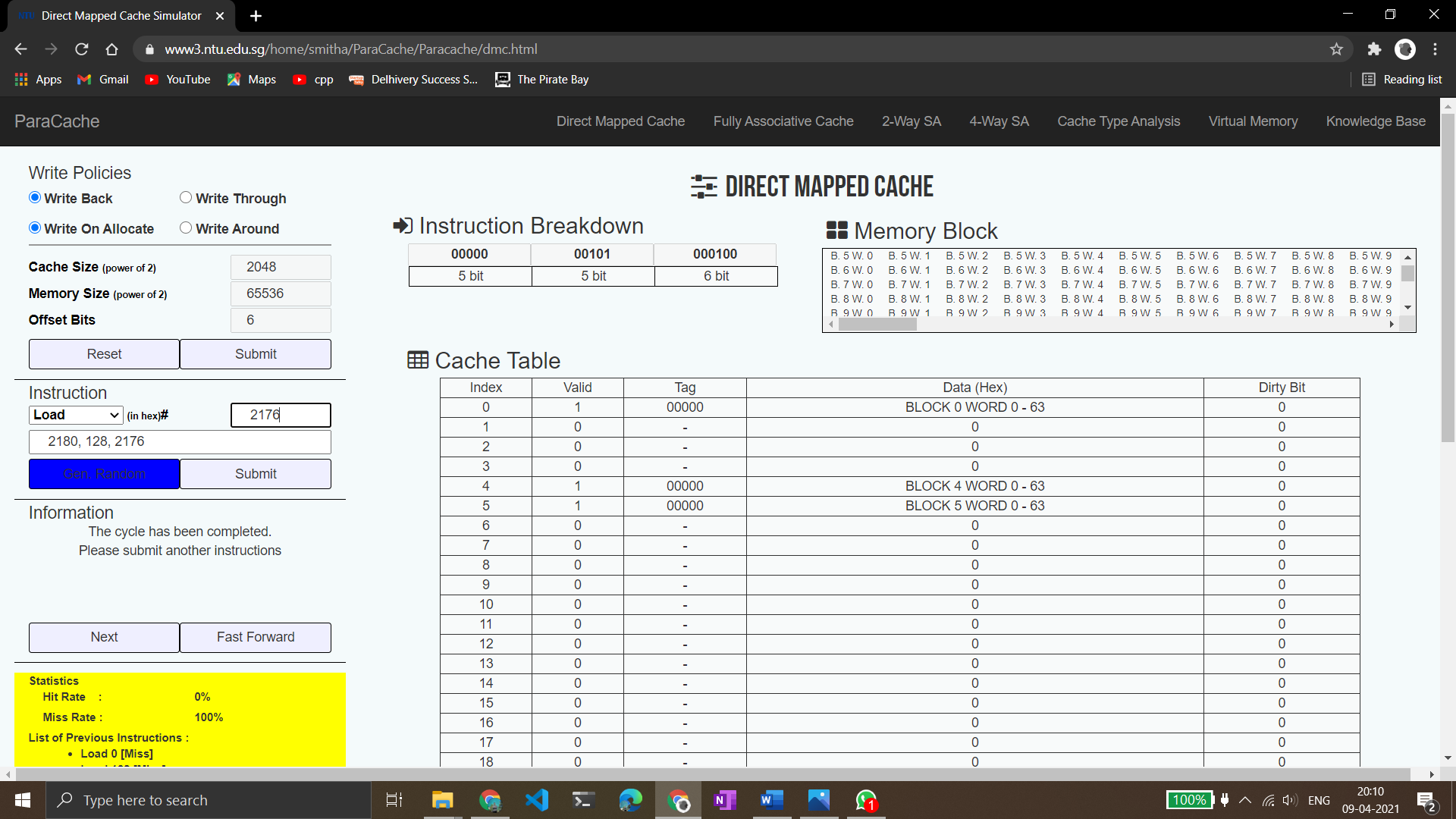
* Index bits: 5
* Tag bits: 5
* Block bits: 10

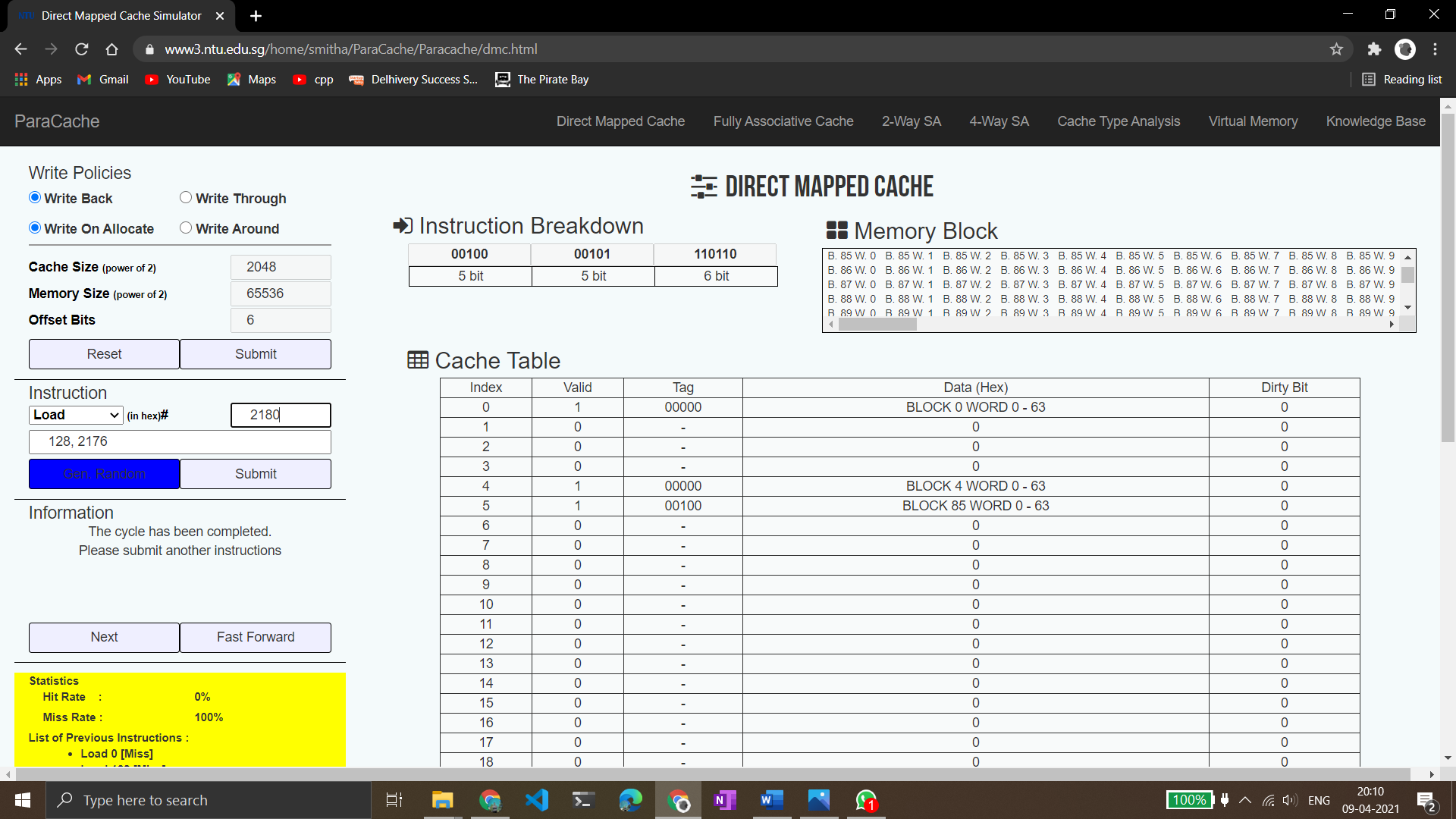
128 - Miss

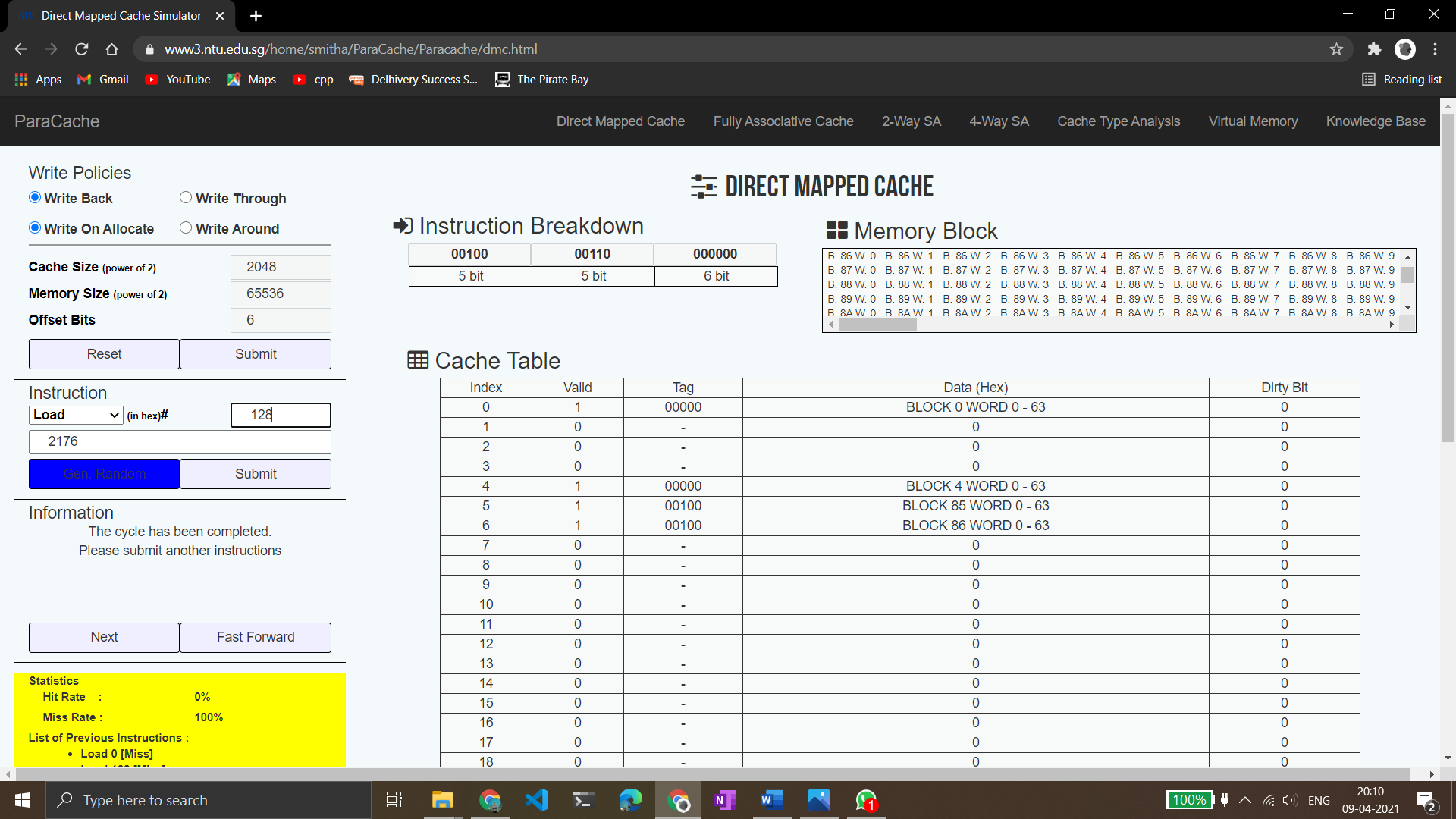


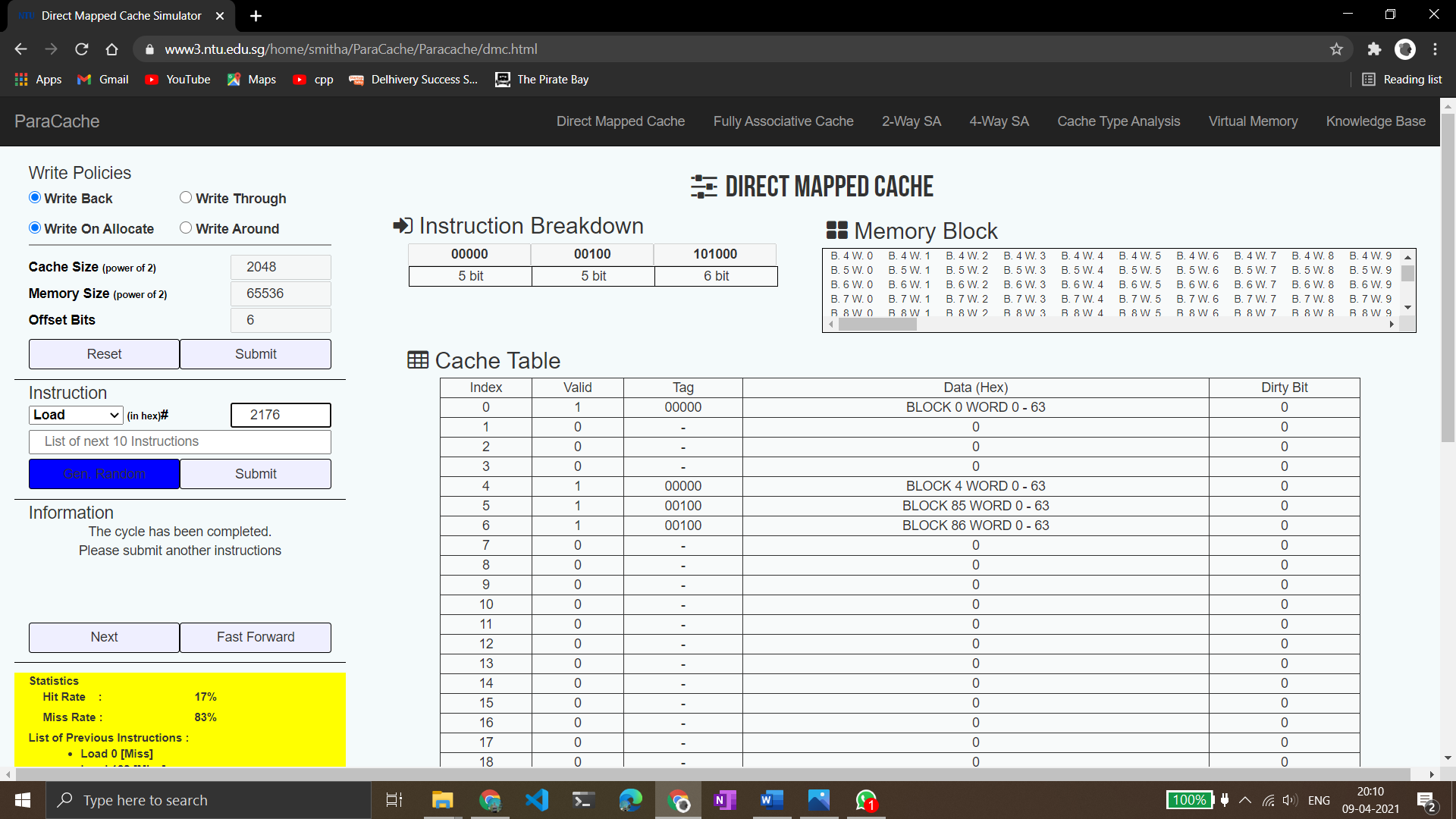
144 - Miss

2176 - Miss

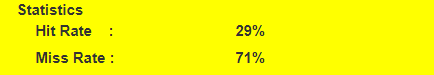


2180 - Miss

128 - Hit

2176 – Hit

Final Hit and Miss rates:



Total:

Hits: 2

Misses: 4

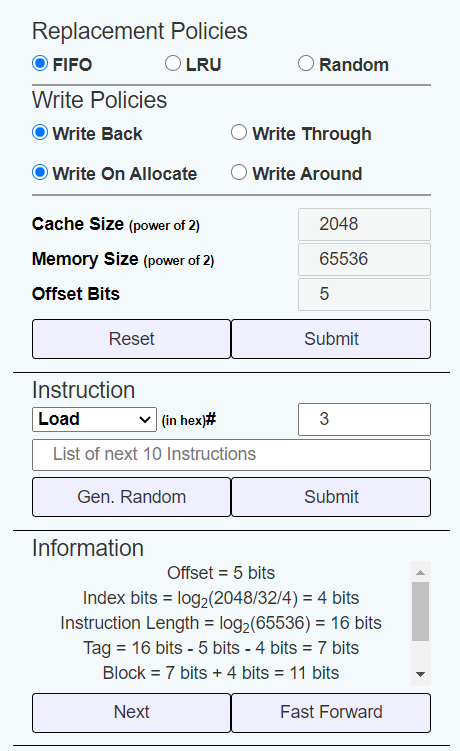
**Part 2: 4 way set associative mapping**

For the above mentioned problem, calculate and execute for 4way set

associativity and fully associative mapping technique. For each technique

randomly generate ten addresses and indicate whether the cache access

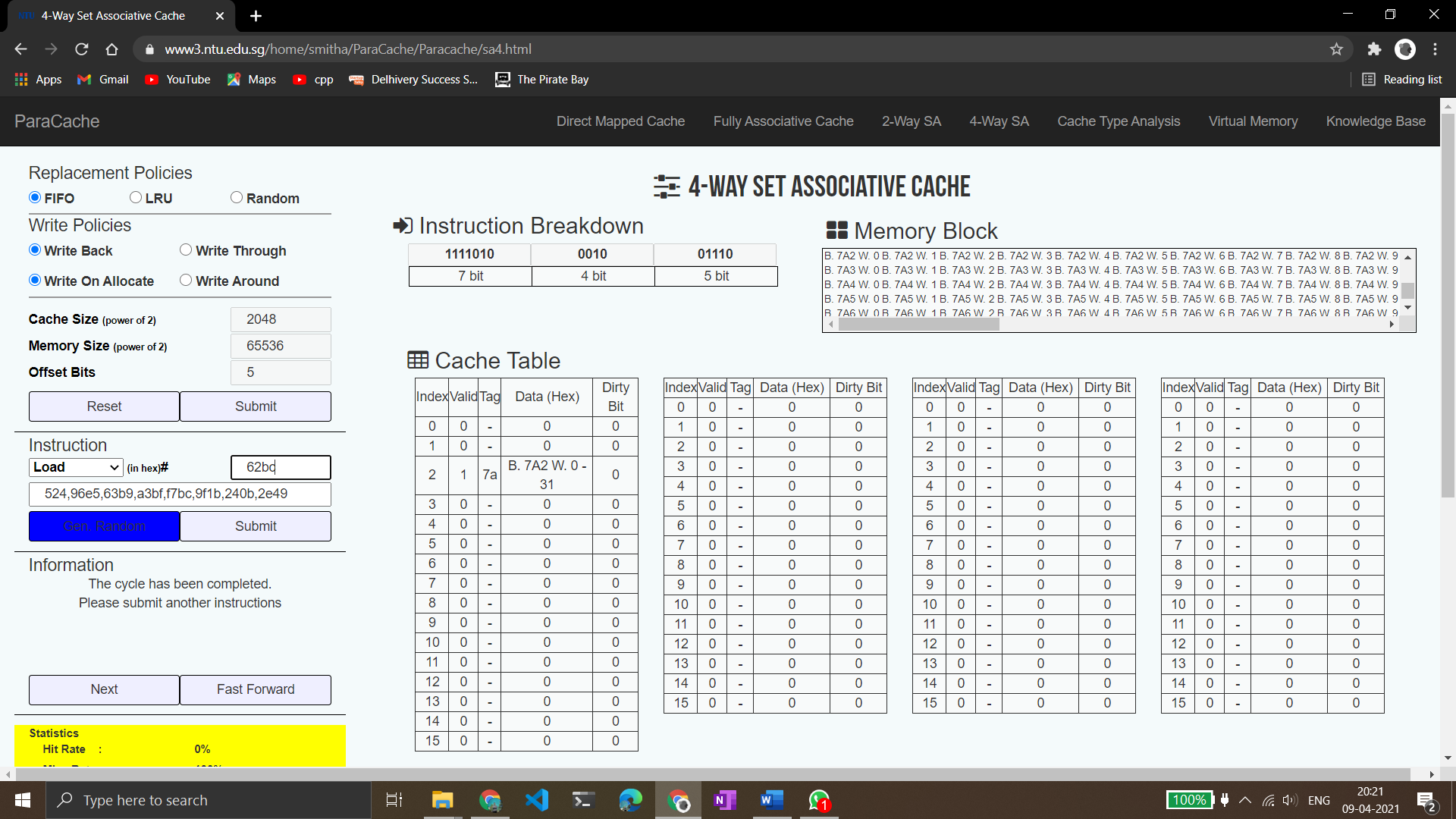
will result in a hit or a miss. Assume block replacement policy as random.



Observations based on given info:

* Index bits: 4
* Tag bits: 7
* Block bits: 11

Generated a random set of numbers:

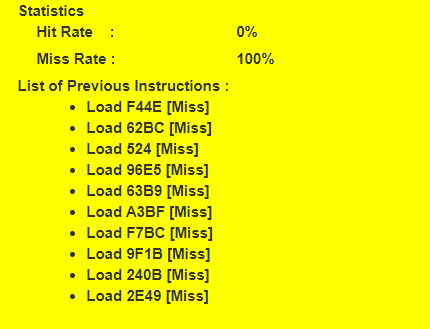


All the numbers lead to misses.

Final statistics:

Misses = 10

Hits = 0



Week#\_\_\_\_9\_\_\_\_\_\_\_Program Number: \_\_\_\_3\_\_\_

Title of the Program:

**Given a ‘c’ code convert it in its equivalent Arm Code and execute in ARM simulator.**

1) x = (a + b) – c:

ARM assembly code:

.text

mov r0, #0 ;considered x

mov r1, #2 ;considered a

mov r2, #3 ;considered b

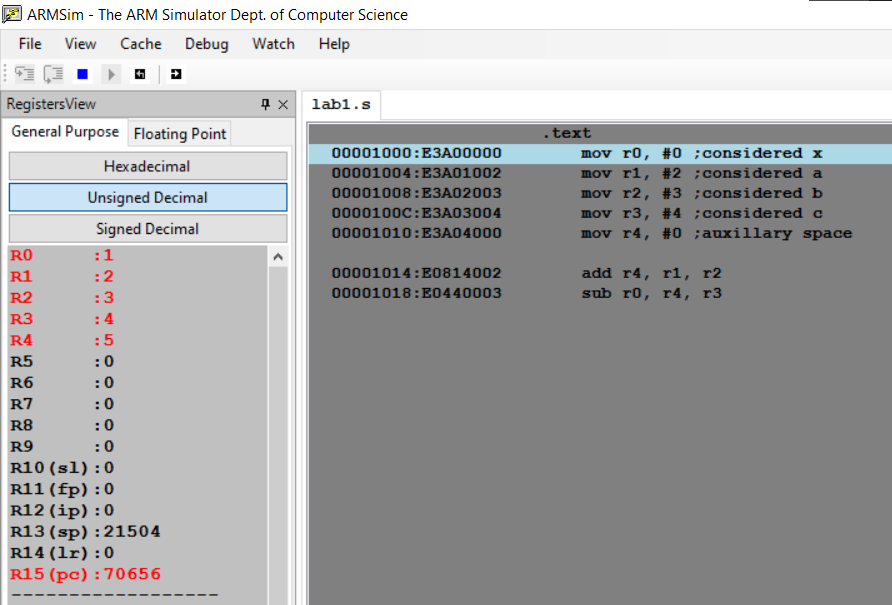
mov r3, #4 ;considered c

mov r4, #0 ;auxillary space

add r4,r1,r2

sub r0,r4,r3

Output screenshot:



2)z = (a << 2) | (b & 15):

ARM assembly code:

.text

mov r0, #0 ;considered z

mov r1, #2 ;considered a

mov r2, #3 ;considered b

mov r3, #4 ;auxillary space 1

mov r4, #0 ;auxillary space 2

and r3, r2, #15

mov r4, r1, LSL #2

orr r0, r3, r4

Output screenshot:

