**Microprocessor and Computer Architecture Laboratory**

**UE19CS256**

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

|  |  |  |
| --- | --- | --- |
| Name: Pranav R. Hegde | SRN: PES1UG19CS343 | Section: F |

Date:09/04/2021

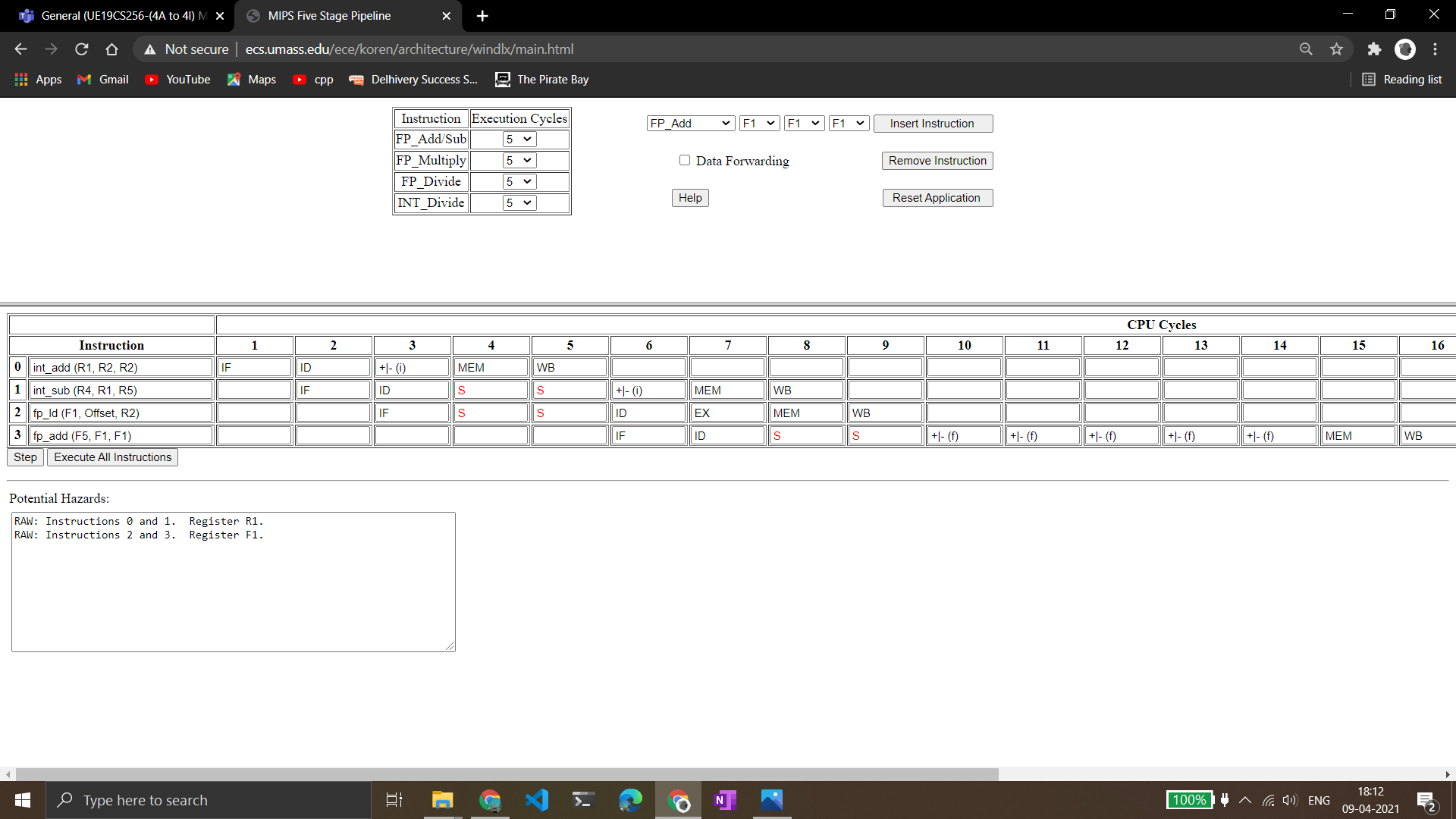
Week#\_\_\_\_9\_\_\_\_\_\_\_Program Number: \_\_\_\_1\_\_\_

Title of the Program:

**5 Stage MIPS pipelined simulator**

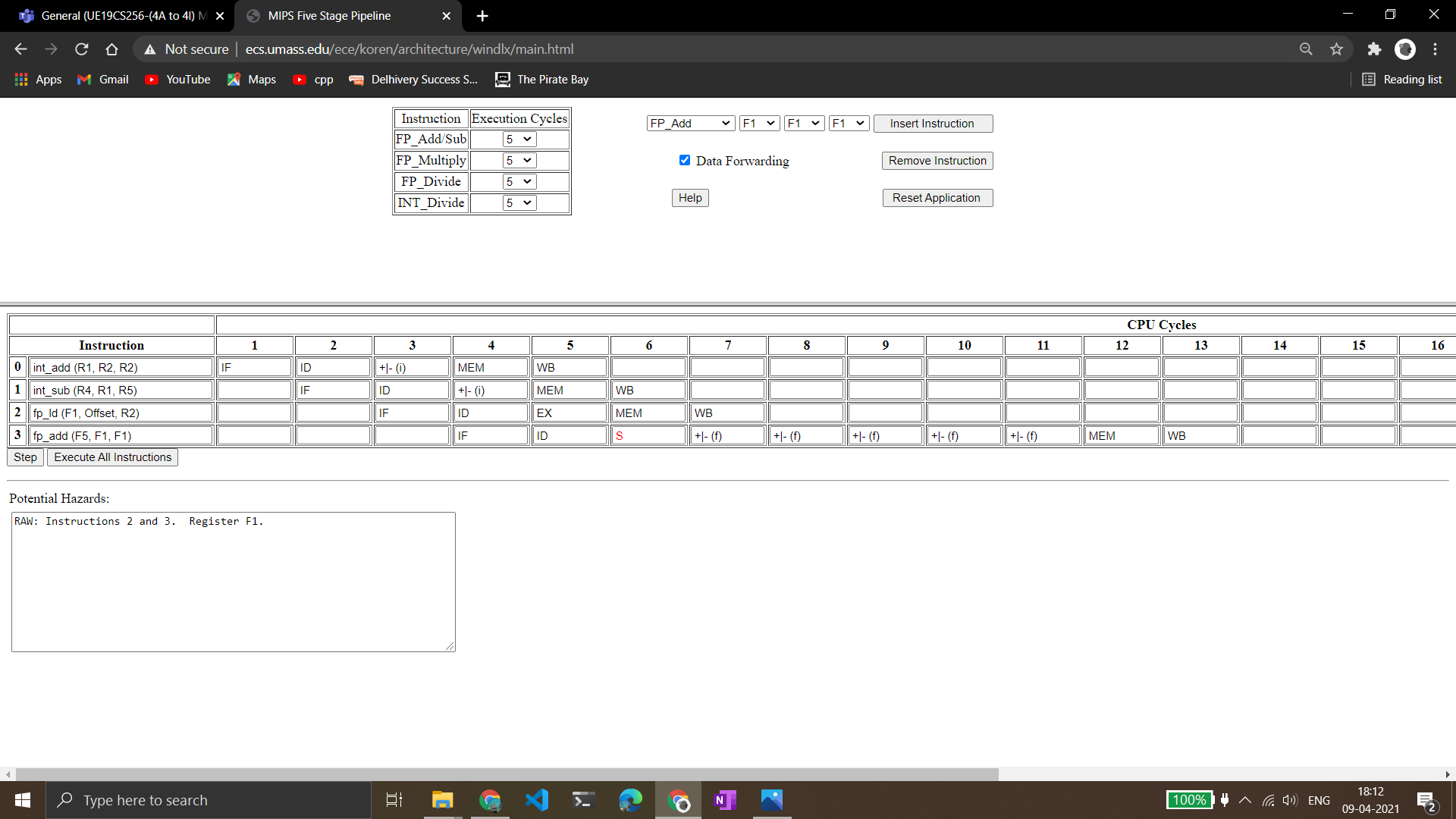
Observations:

Without data forwarding



* Yes, there is a data dependency between the instructions.
* The number of stalls introduced in the case without data forwarding are 6.
* And the number of clock cycles consumed are 16.

With data forwarding:



* In case of data forwarding, only 1 stall is introduced.
* And the number of clock cycles consumed are 13.

**Microprocessor and Computer Architecture Laboratory**

**UE19CS256**

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

|  |  |  |
| --- | --- | --- |
| Name: Pranav R. Hegde | SRN: PES1UG19CS343 | Section: F |

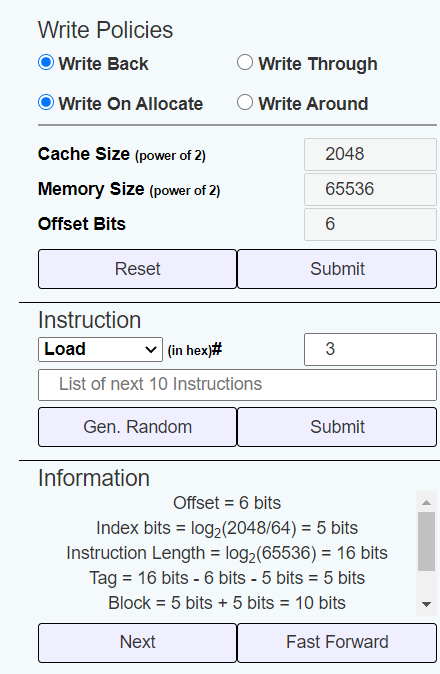
Date:09/04/2021

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

Title of the Program:

**Cache simulator**

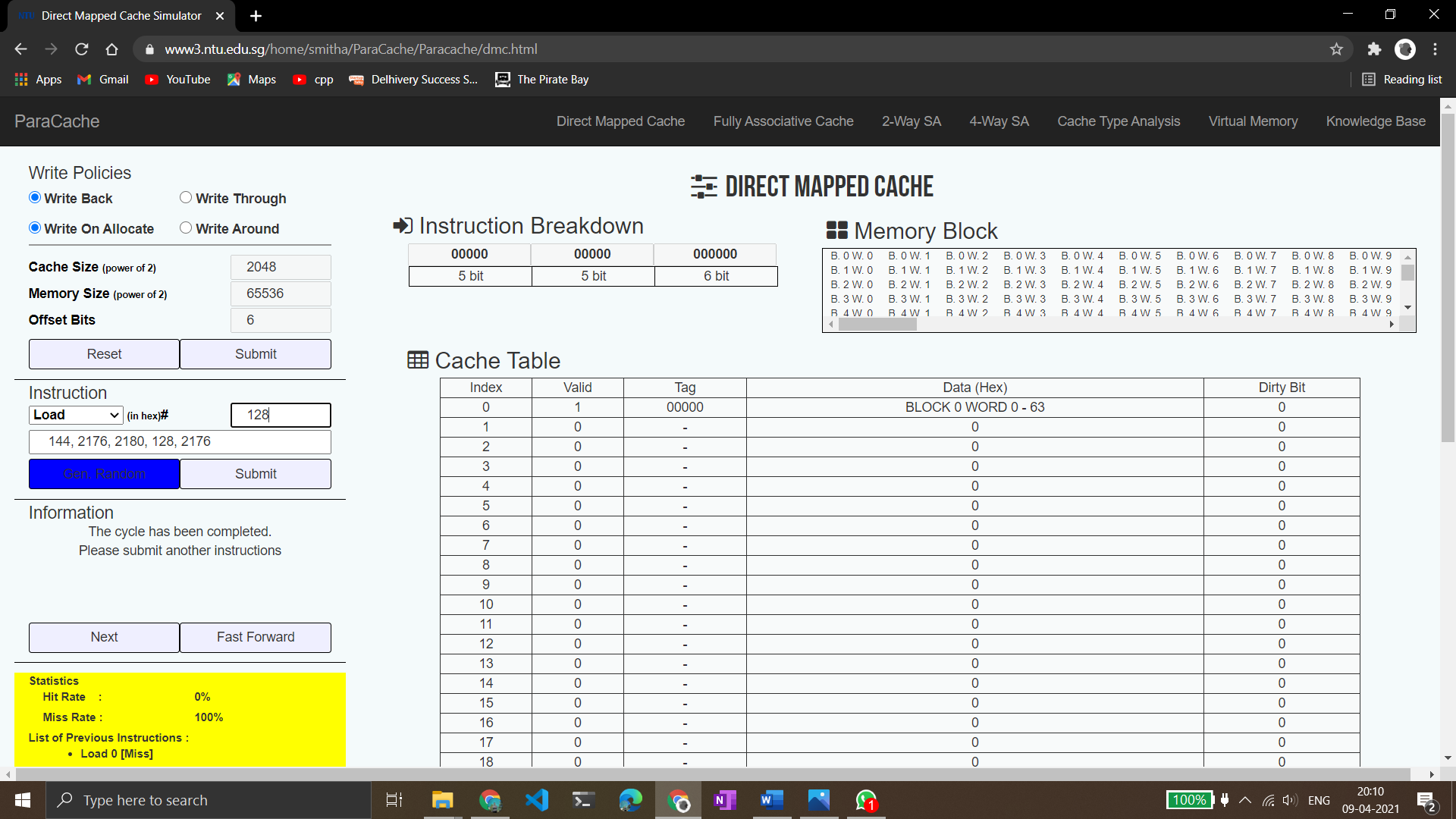
**Part 1 – Direct mapping:**

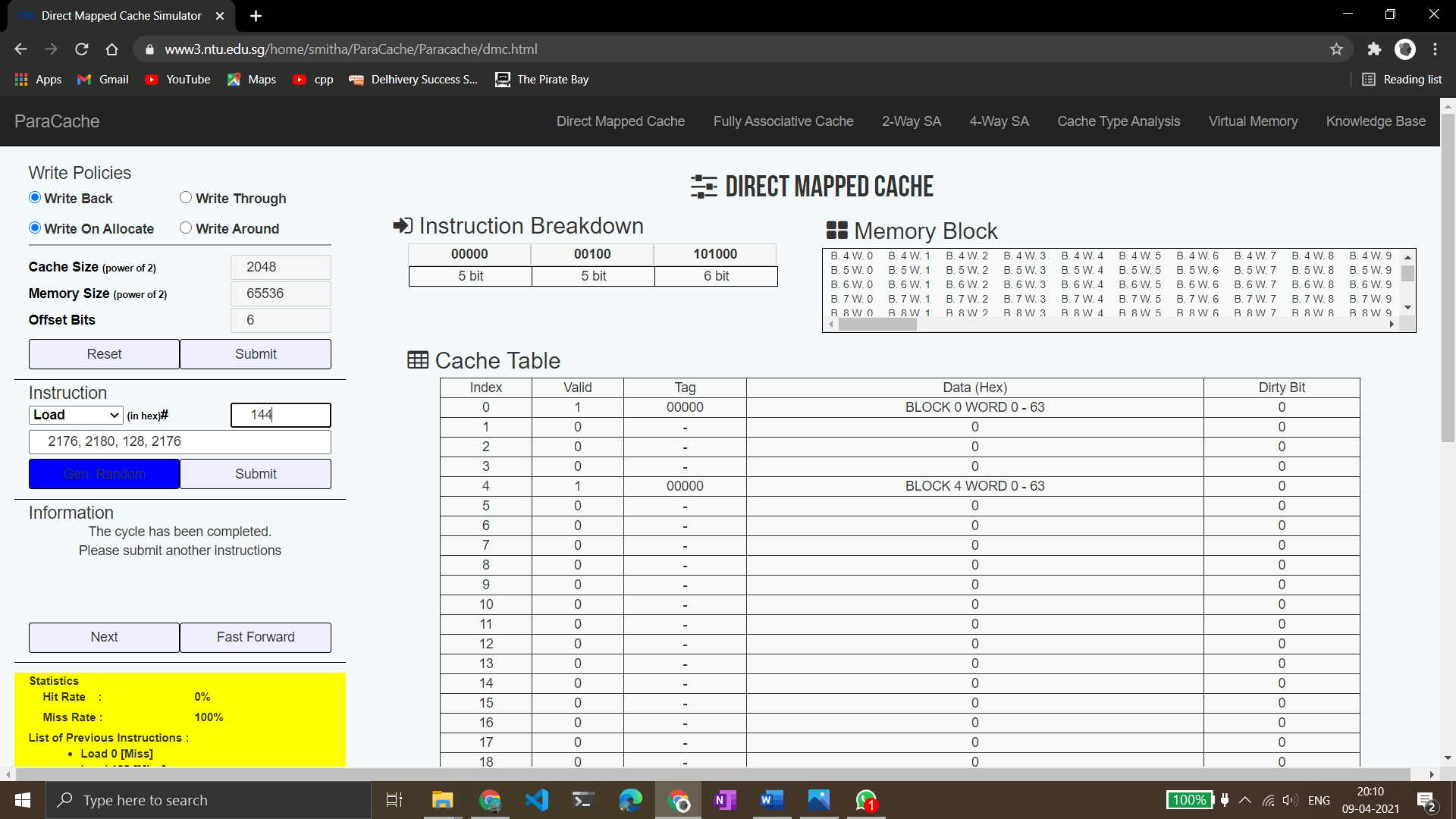


Based on the info given:

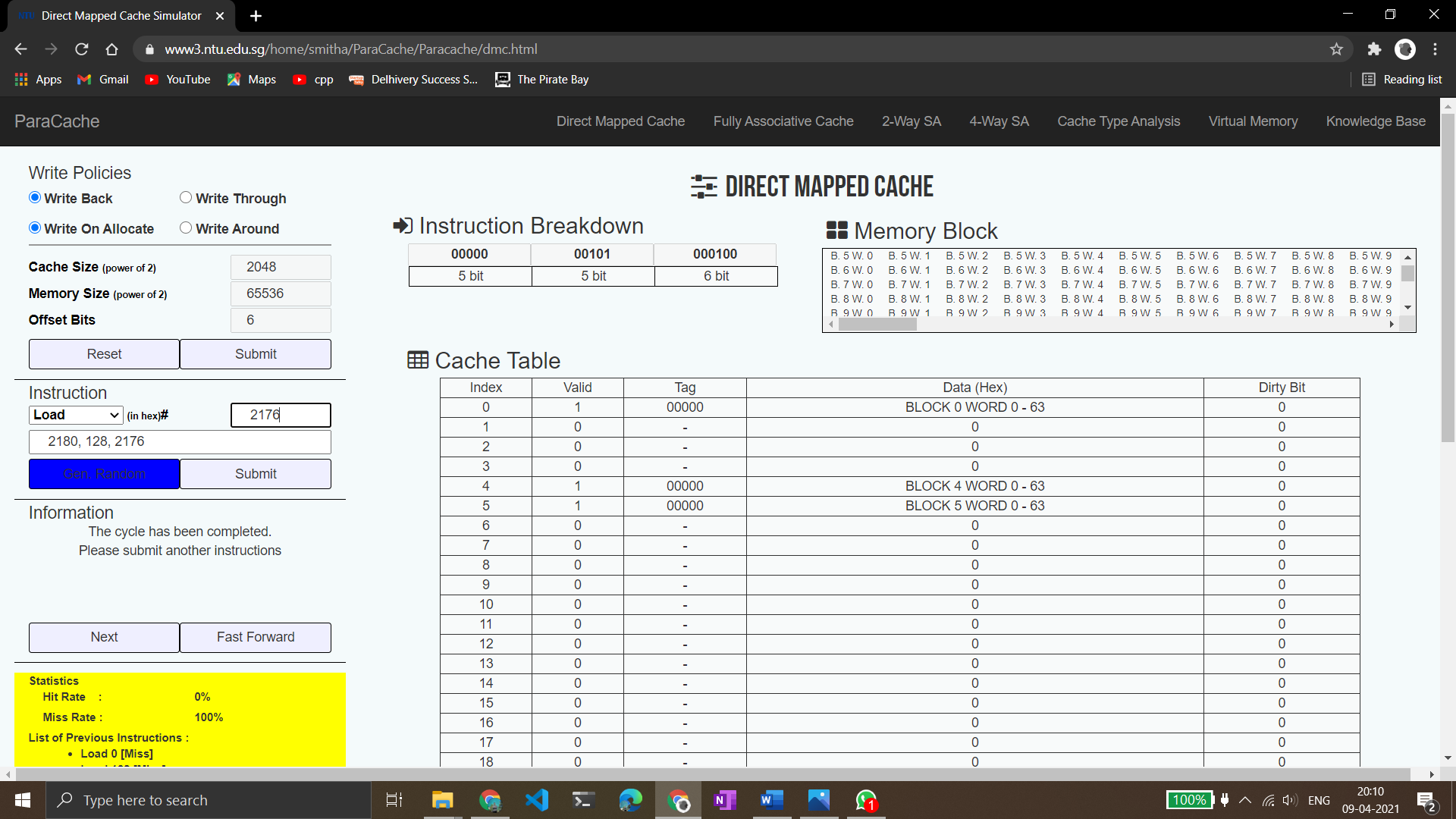
* Index bits: 5
* Tag bits: 5
* Block bits: 10

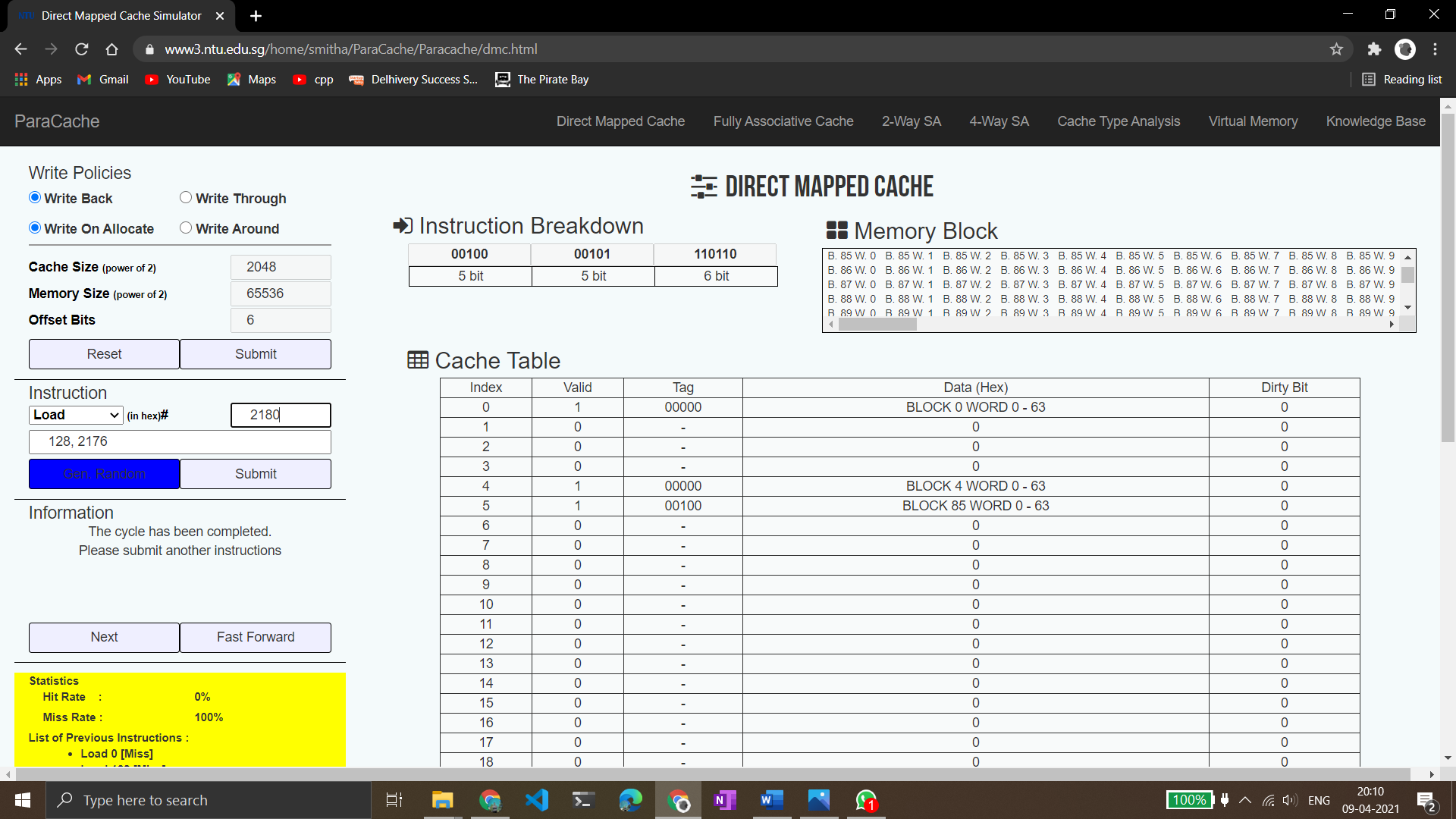
128 - Miss

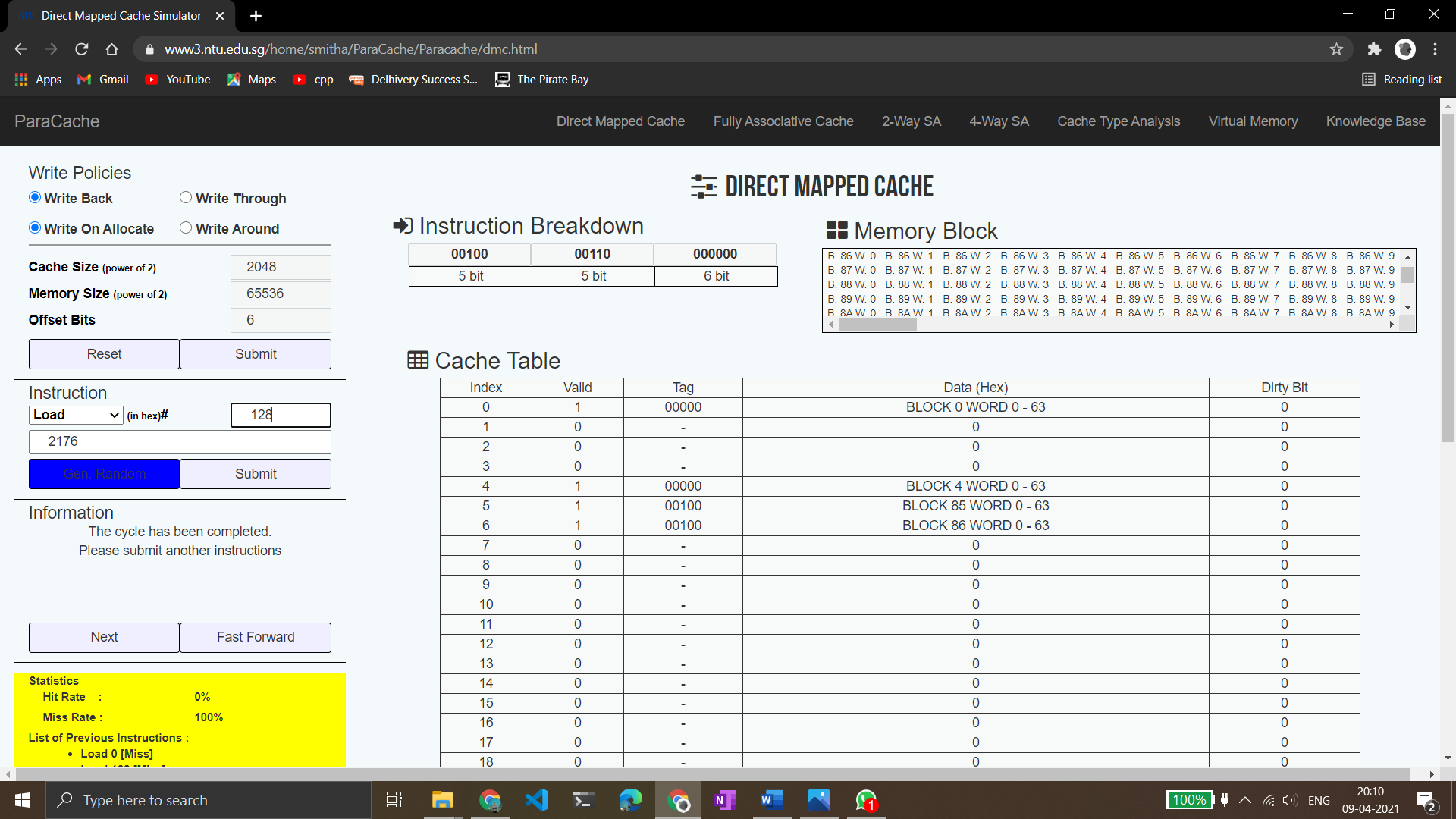


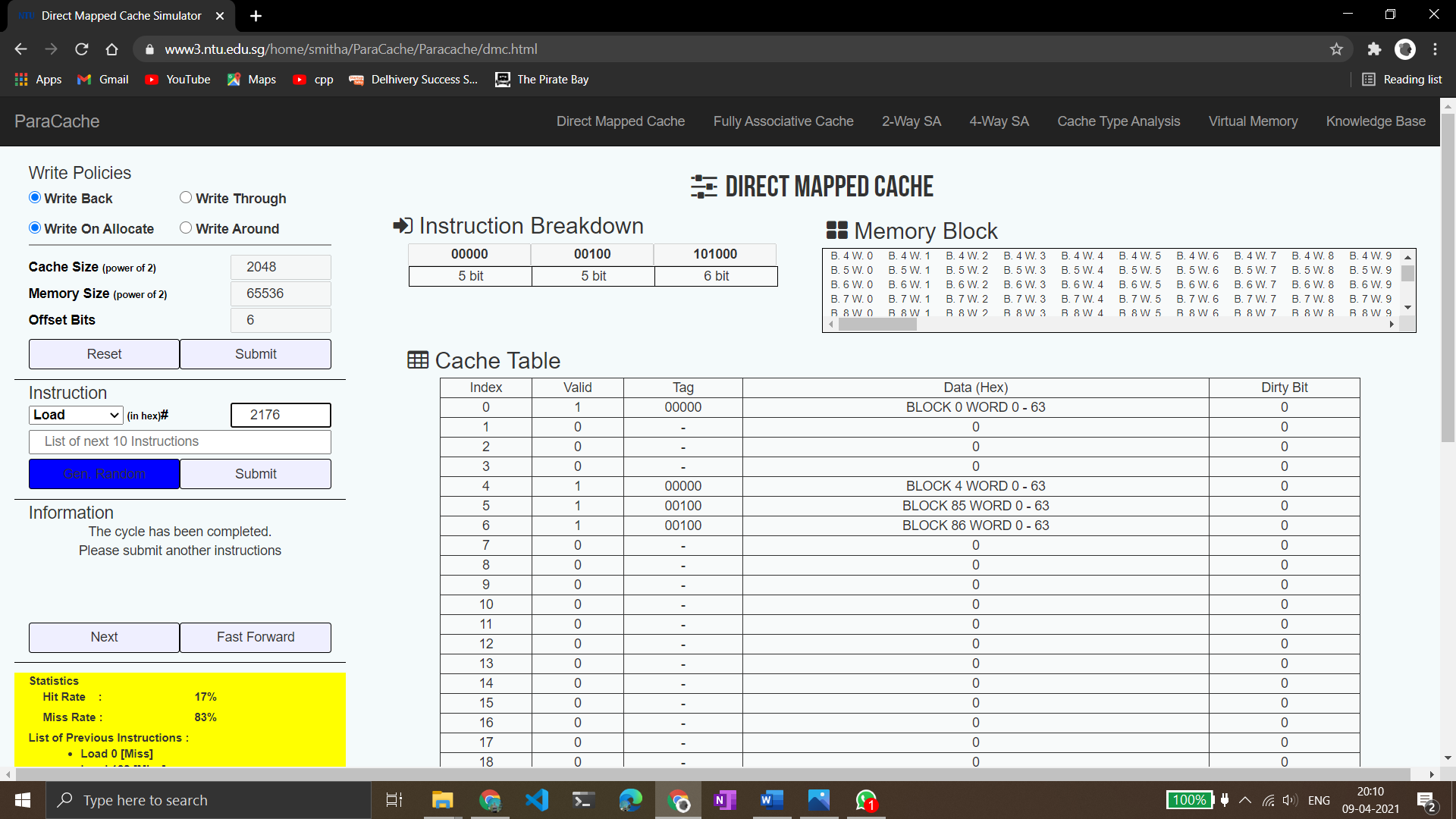
144 - Miss

2176 - Miss

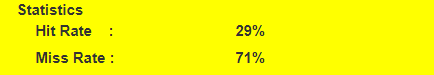


2180 - Miss

128 - Hit

2176 – Hit

Final Hit and Miss rates:

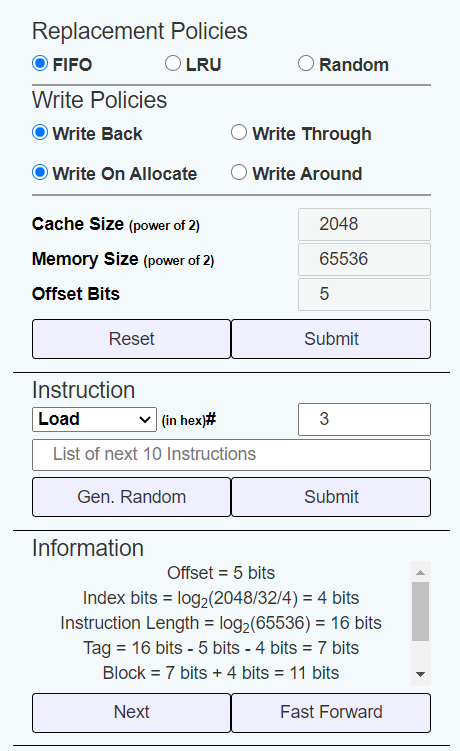


Total:

Hits: 2

Misses: 4

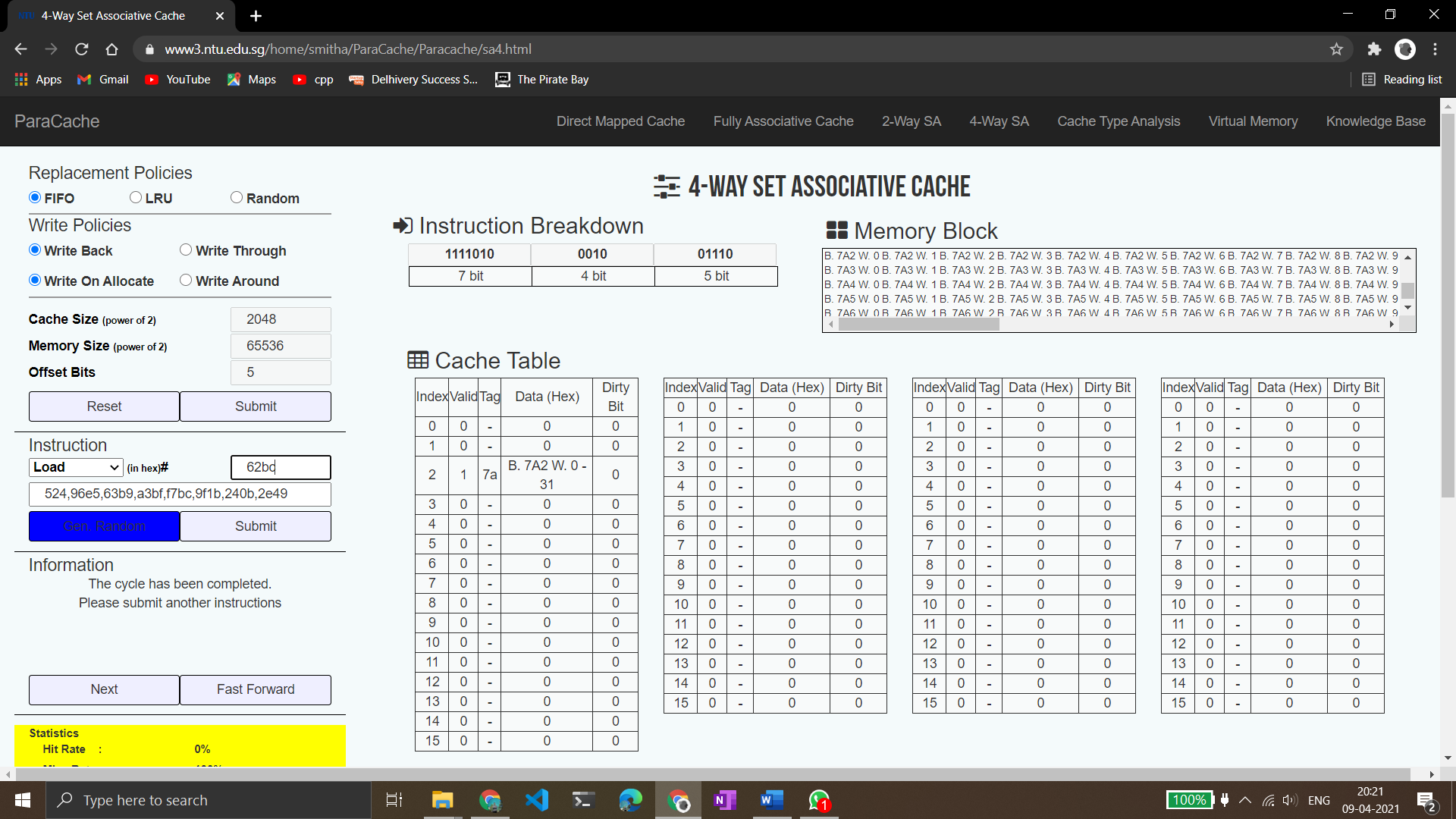
**Part 2: 4 way set associative mapping**



Observations based on given info:

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

Generated a random set of numbers:

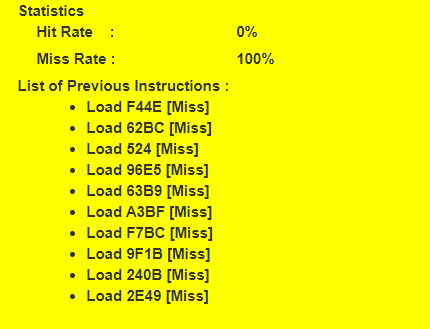


All the numbers lead to misses.

Final statistics:

Misses = 10

Hits = 0



**Microprocessor and Computer Architecture Laboratory**

**UE19CS256**

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

|  |  |  |
| --- | --- | --- |
| Name: Pranav R. Hegde | SRN: PES1UG19CS343 | Section: F |

Date:09/04/2021

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

Title of the Program:

**Write the equivalent ARM assembly instructions for the given C code**

**Part A -** x = (a + b) – c:

The code:

.text

    mov r0, #0 *;considered x*

    mov r1, #2 *;considered a*

    mov r2, #3 *;considered b*

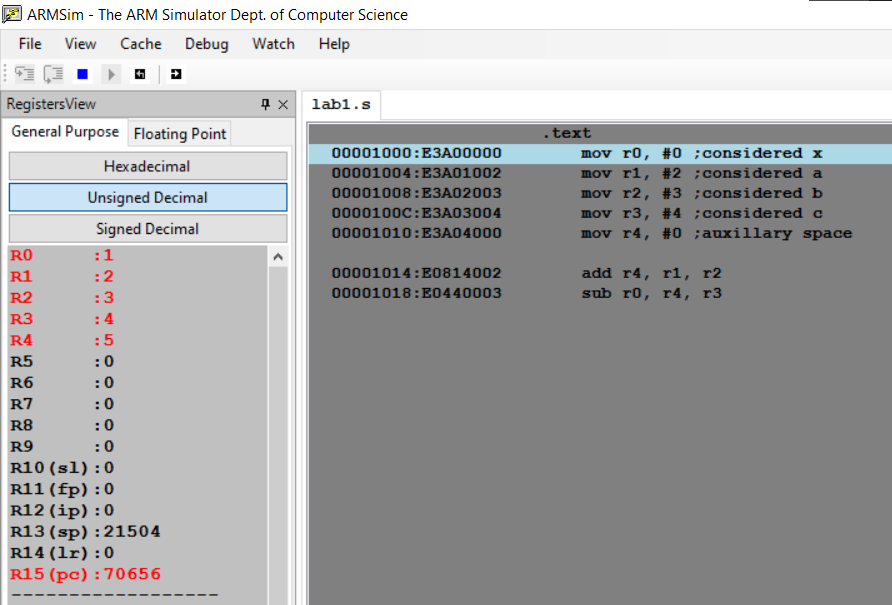
    mov r3, #4 *;considered c*

    mov r4, #0 *;auxillary space*

    add r4, r1, r2

    sub r0, r4, r3

The arm sim executed screenshot:



**Part B -** z = (a << 2) | (b & 15):

The 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

The arm sim executed screenshot:

