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:

Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

FP_Add F1 F1 F1 Insert Instruction

Data Forwarding Remove Instruction

Help Reset Application

	Instruction	CPU Cycles 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	int_add (R1, R2, R3)	IF	ID	+|- (i)	MEM	WB													
1	int_sub (R4, R1, R5)		IF	ID	S	S	+|- (i)	MEM	WB										
2	fp_ld (F1, Offset, R1)			IF	S	S	ID	EX	MEM	WB									
3	fp_add (F5, F1, F1)						IF	ID	S	S	+|- (f)	MEM	WB						

Step Execute All Instructions

Potential Hazards:

```
RAW: Instructions 0 and 1.  Register R1.
RAW: Instructions 0 and 2.  Register R1.
RAW: Instructions 2 and 3.  Register F1.
```

- 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:

Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

FP_Add F1 F1 F1 Insert Instruction

Data Forwarding Remove Instruction

Help Reset Application

	Instruction	CPU Cycles 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	int_add (R1, R2, R3)	IF	ID	+|- (i)	MEM	WB													
1	int_sub (R4, R1, R5)		IF	ID	+|- (i)	MEM	WB												
2	fp_ld (F1, Offset, R1)			IF	ID	EX	MEM	WB											
3	fp_add (F5, F1, F1)				IF	ID	S	+|- (f)	MEM	WB									

Step Execute All Instructions

Potential Hazards:

```
RAW: Instructions 2 and 3.  Register F1.
```

- 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.

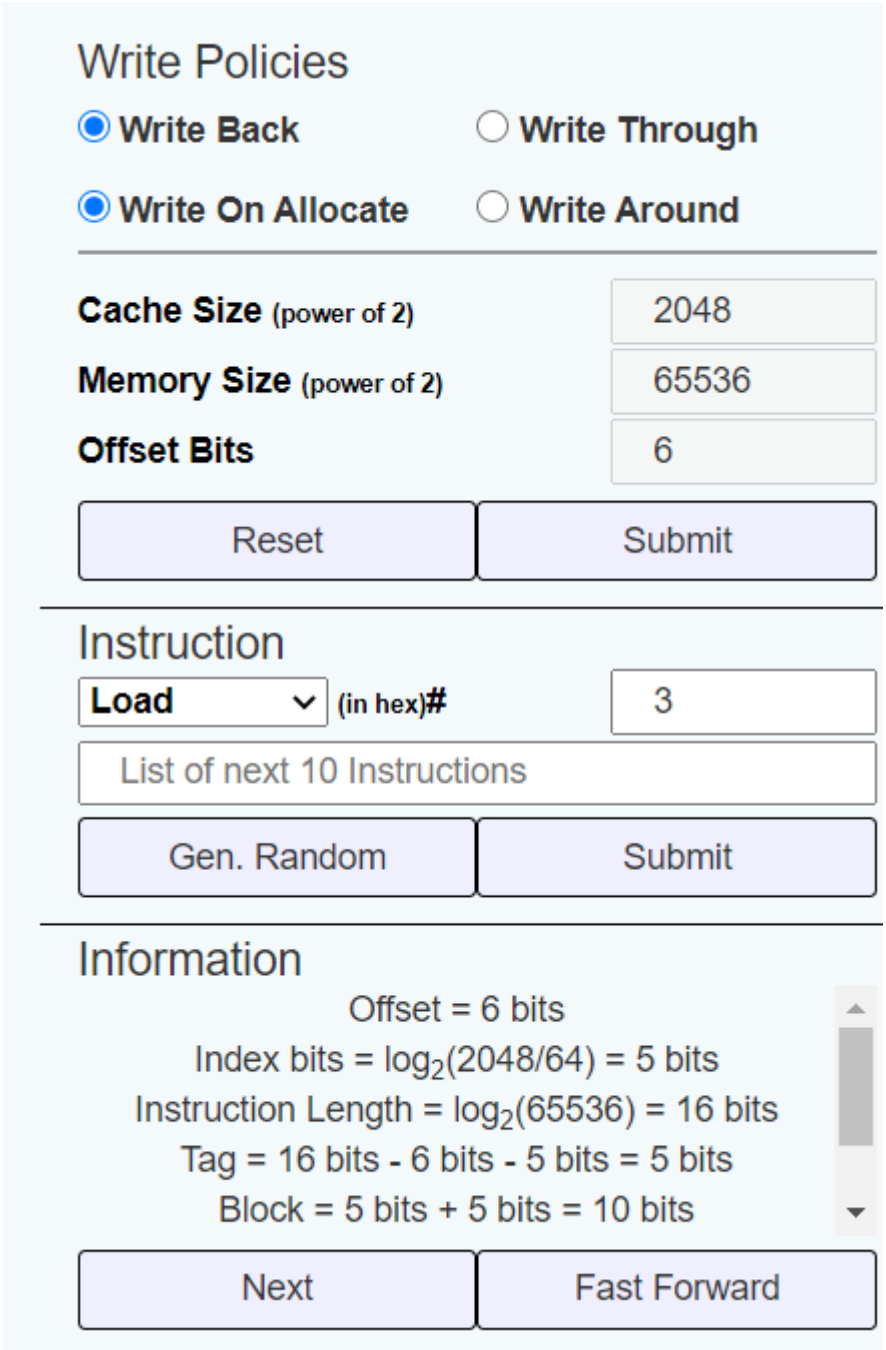
(a) Calculate the number of bits in each of the Tag, Block, and Word fields of the memory address using direct mapped Cache.

(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.

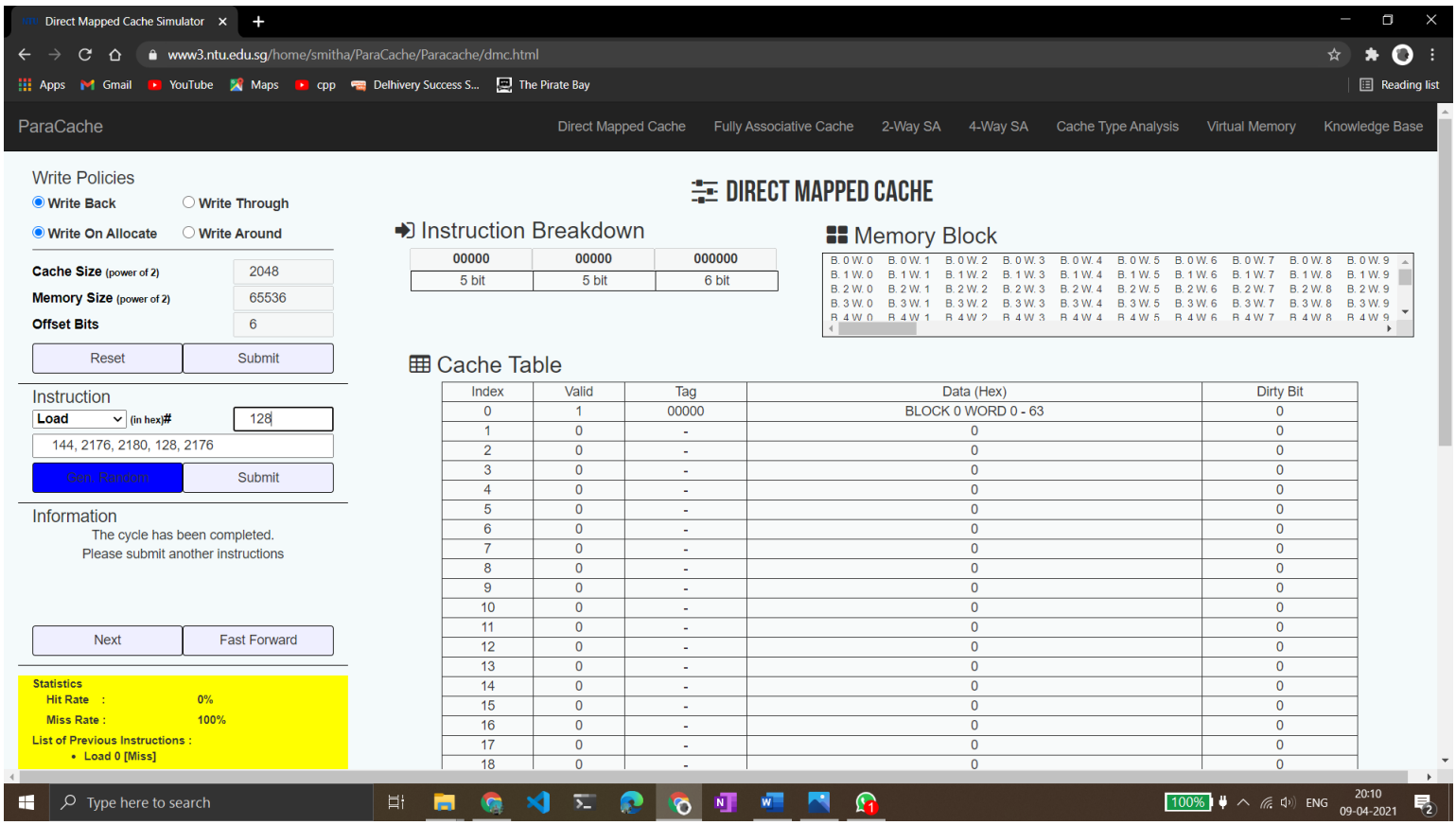
For each of the above addresses, indicate whether the cache access will result in a hit or a miss.



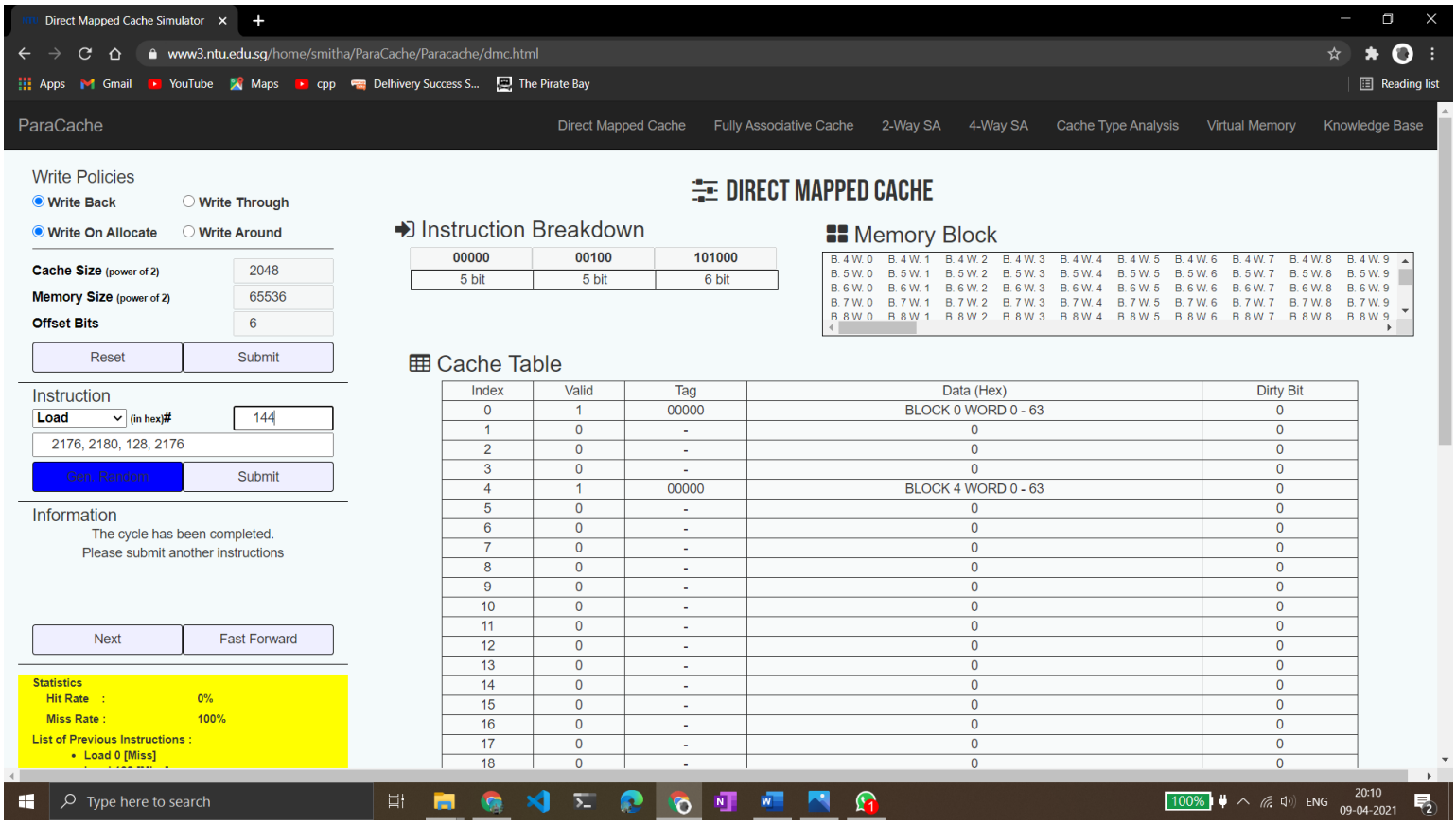
Based on the info given:

- Index bits: 5
- Tag bits: 5
- Block bits: 10

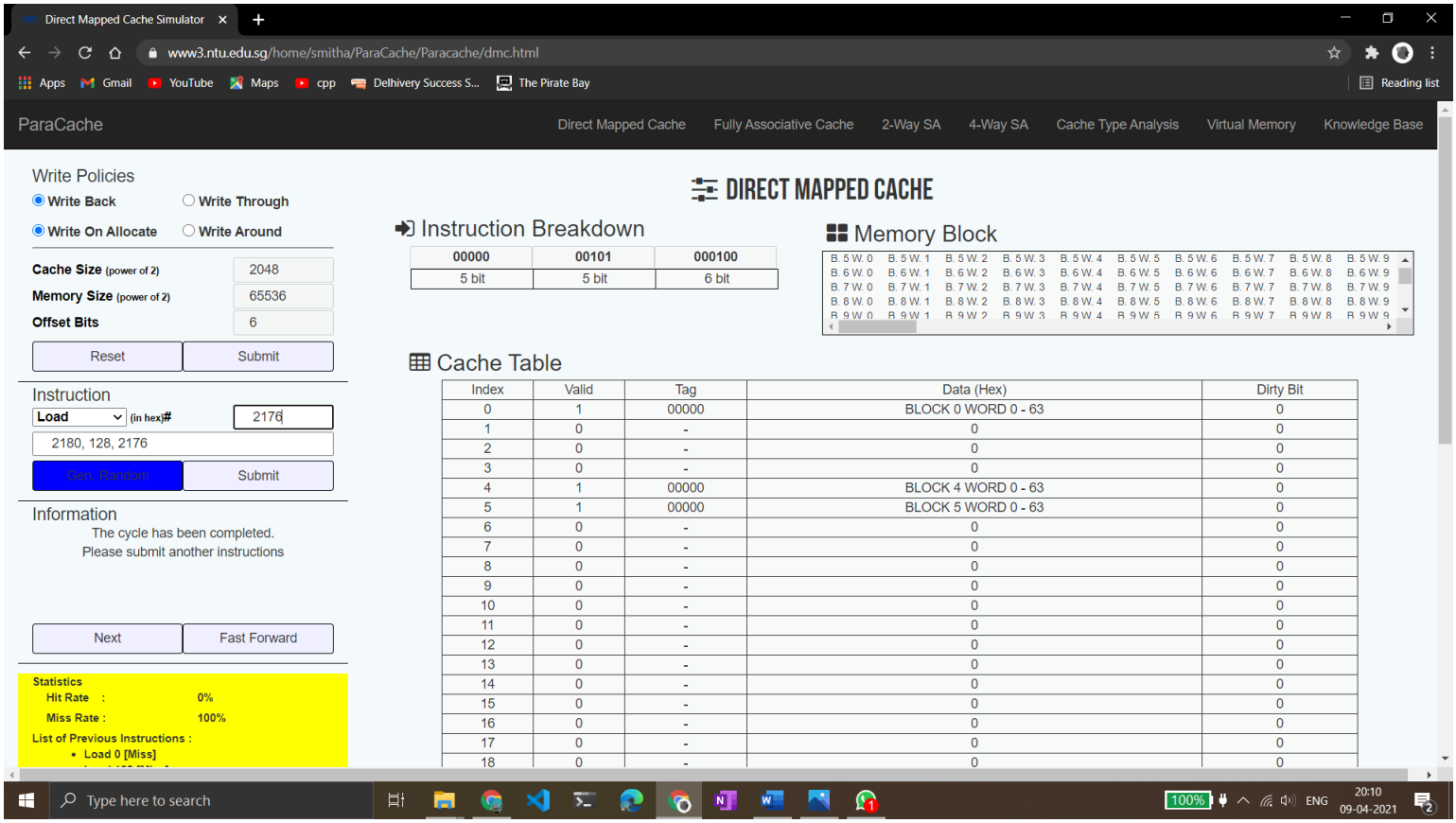
128 - Miss



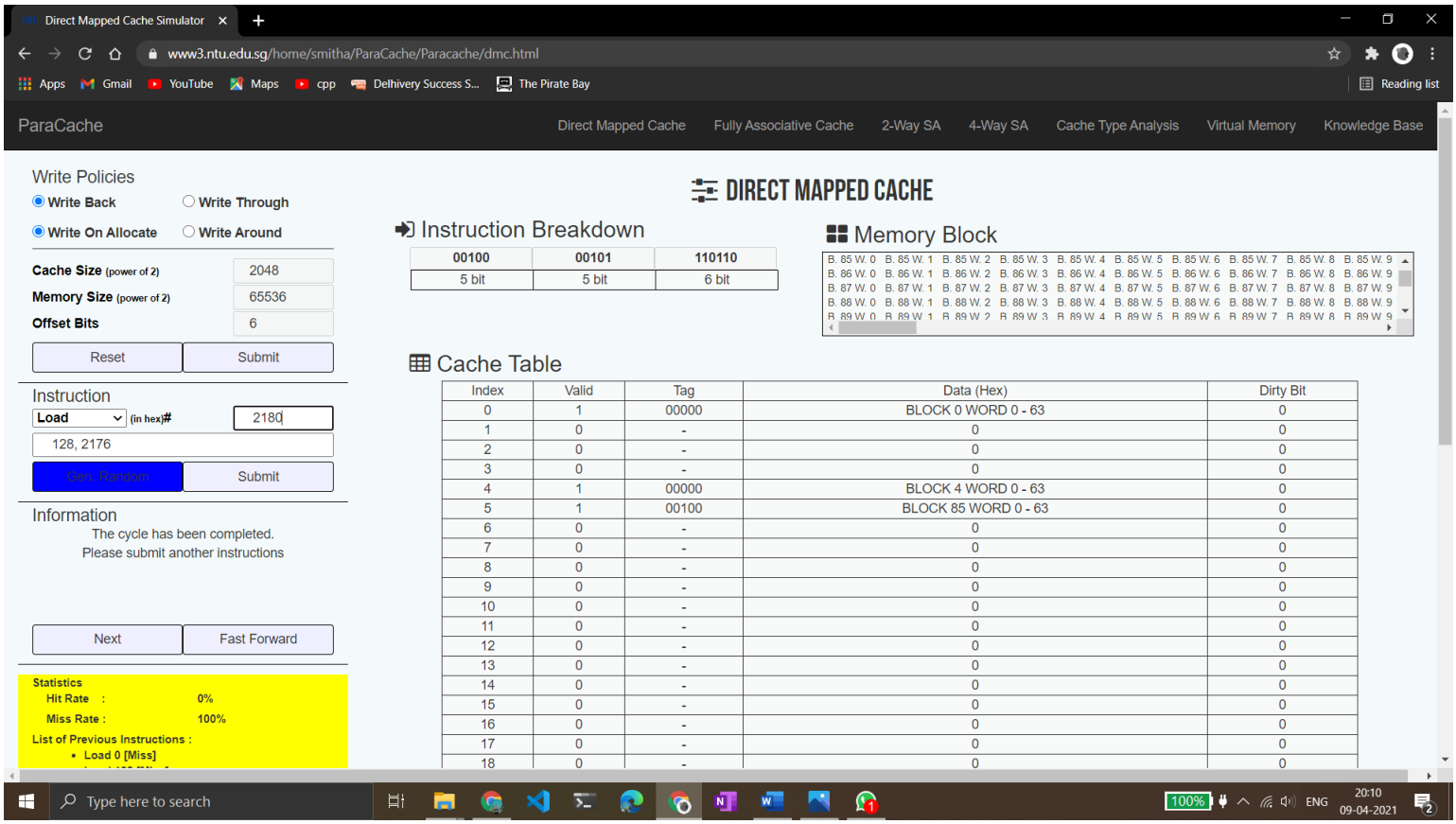
144 - Miss



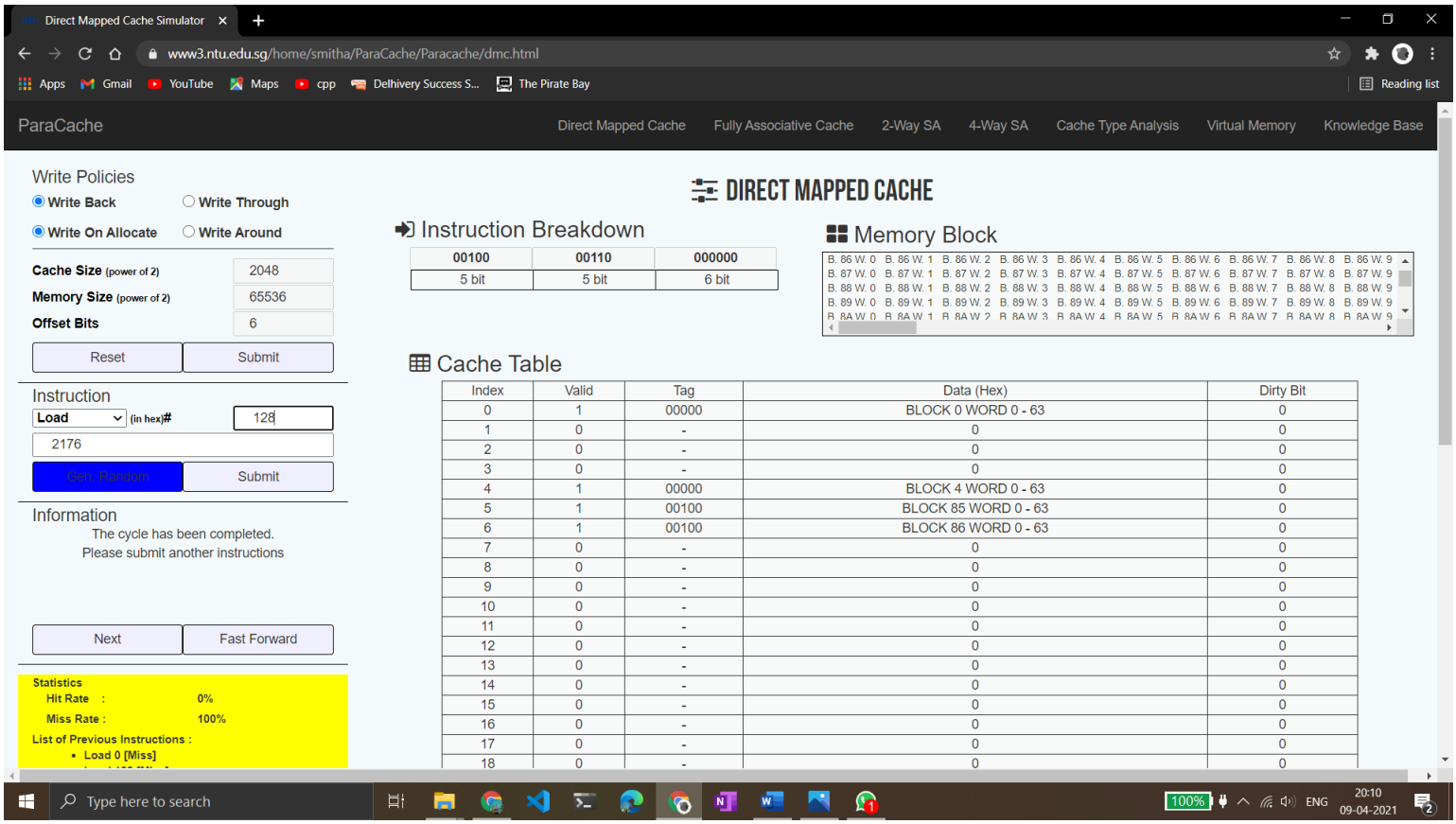
2176 - Miss


Direct Mapped Cache Simulator
www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/dmc.html
ParaCache
Direct Mapped Cache
Fully Associative Cache
2-Way SA
4-Way SA
Cache Type Analysis
Virtual Memory
Knowledge Base
Write Policies
Write Back
Write Through
Write On Allocate
Write Around
Cache Size (power of 2) 2048
Memory Size (power of 2) 65536
Offset Bits 6
Reset
Submit
Instruction
Load (in hex)# 2176
2180, 128, 2176
Gen. Random
Submit
Information
The cycle has been completed.
Please submit another instructions
Next
Fast Forward
Statistics
Hit Rate : 0%
Miss Rate : 100%
List of Previous Instructions :
Load 0 [Miss]
DIRECT MAPPED CACHE
Instruction Breakdown
00000 | 00101 | 000100
5 bit | 5 bit | 6 bit
Memory Block
Cache Table
Index | Valid | Tag | Data (Hex) | Dirty Bit
0 | 1 | 00000 | BLOCK 0 WORD 0 - 63 | 0
1 | 0 | - | 0 | 0
2 | 0 | - | 0 | 0
3 | 0 | - | 0 | 0
4 | 1 | 00000 | BLOCK 4 WORD 0 - 63 | 0
5 | 1 | 00000 | BLOCK 5 WORD 0 - 63 | 0
6 | 0 | - | 0 | 0
7 | 0 | - | 0 | 0
8 | 0 | - | 0 | 0
9 | 0 | - | 0 | 0
10 | 0 | - | 0 | 0
11 | 0 | - | 0 | 0
12 | 0 | - | 0 | 0
13 | 0 | - | 0 | 0
14 | 0 | - | 0 | 0
15 | 0 | - | 0 | 0
16 | 0 | - | 0 | 0
17 | 0 | - | 0 | 0
18 | 0 | - | 0 | 0

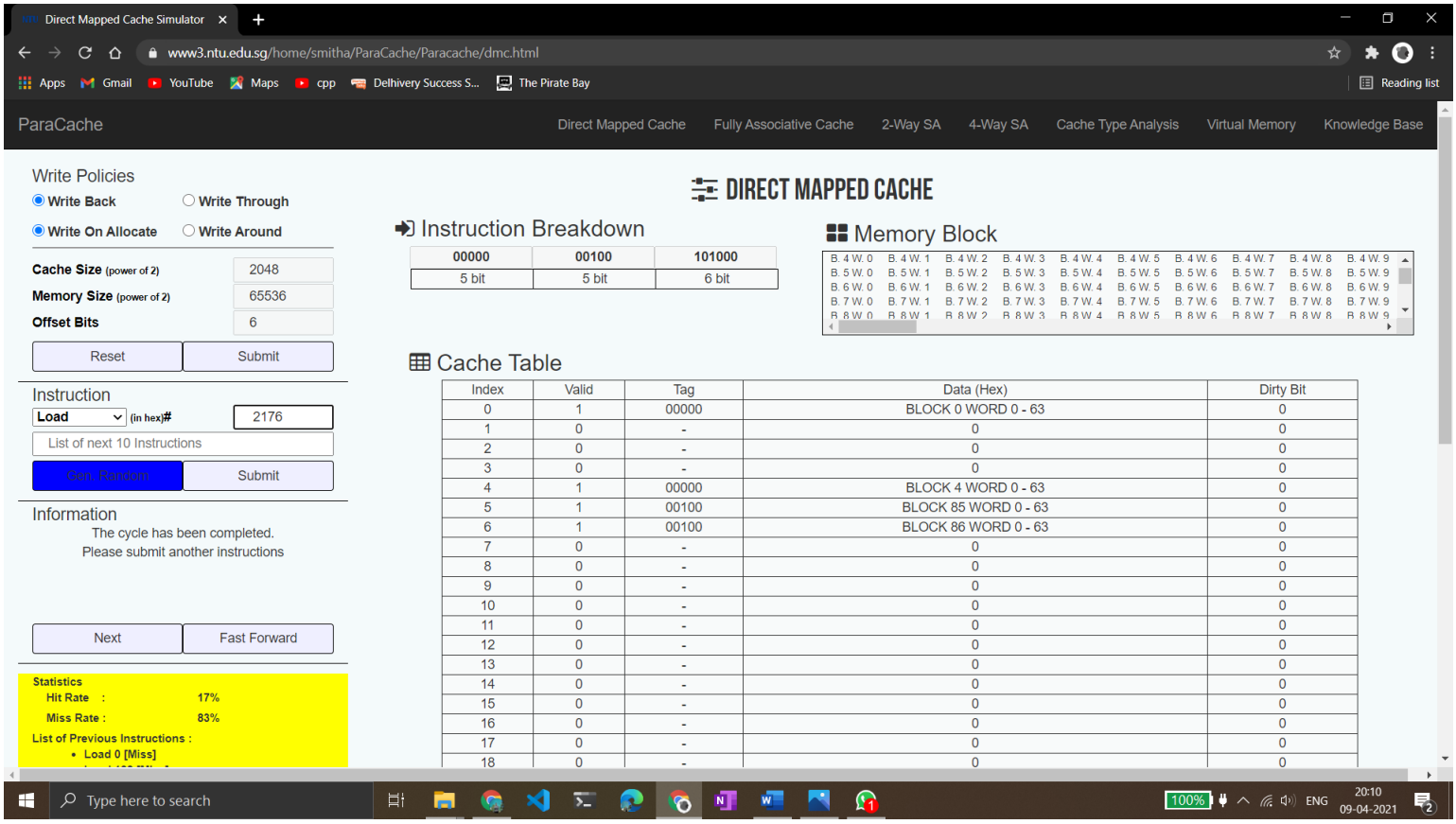
2180 - Miss


Direct Mapped Cache Simulator
www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/dmc.html
ParaCache
Direct Mapped Cache
Fully Associative Cache
2-Way SA
4-Way SA
Cache Type Analysis
Virtual Memory
Knowledge Base
Write Policies
Write Back
Write Through
Write On Allocate
Write Around
Cache Size (power of 2) 2048
Memory Size (power of 2) 65536
Offset Bits 6
Reset
Submit
Instruction
Load (in hex)# 2180
128, 2176
Gen. Random
Submit
Information
The cycle has been completed.
Please submit another instructions
Next
Fast Forward
Statistics
Hit Rate : 0%
Miss Rate : 100%
List of Previous Instructions :
Load 0 [Miss]
DIRECT MAPPED CACHE
Instruction Breakdown
00100 | 00101 | 110110
5 bit | 5 bit | 6 bit
Memory Block
Cache Table
Index | Valid | Tag | Data (Hex) | Dirty Bit
0 | 1 | 00000 | BLOCK 0 WORD 0 - 63 | 0
1 | 0 | - | 0 | 0
2 | 0 | - | 0 | 0
3 | 0 | - | 0 | 0
4 | 1 | 00000 | BLOCK 4 WORD 0 - 63 | 0
5 | 1 | 00100 | BLOCK 85 WORD 0 - 63 | 0
6 | 0 | - | 0 | 0
7 | 0 | - | 0 | 0
8 | 0 | - | 0 | 0
9 | 0 | - | 0 | 0
10 | 0 | - | 0 | 0
11 | 0 | - | 0 | 0
12 | 0 | - | 0 | 0
13 | 0 | - | 0 | 0
14 | 0 | - | 0 | 0
15 | 0 | - | 0 | 0
16 | 0 | - | 0 | 0
17 | 0 | - | 0 | 0
18 | 0 | - | 0 | 0

128 - Hit



2176 – Hit



Final Hit and Miss rates:

Statistics	
Hit Rate :	29%
Miss Rate :	71%

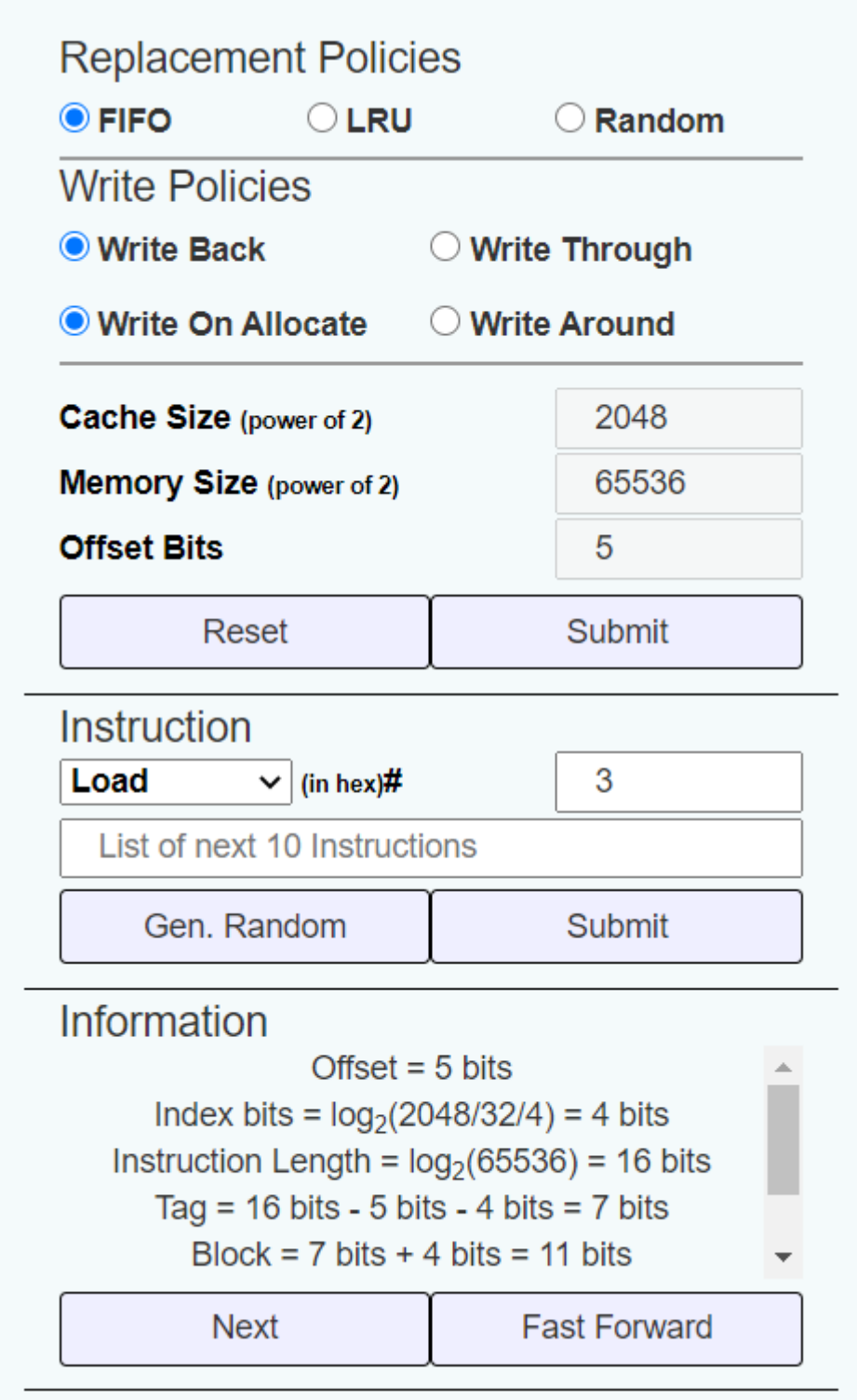
Total:

Hits: 2

Misses: 4

Part 2: 4 way set associative mapping

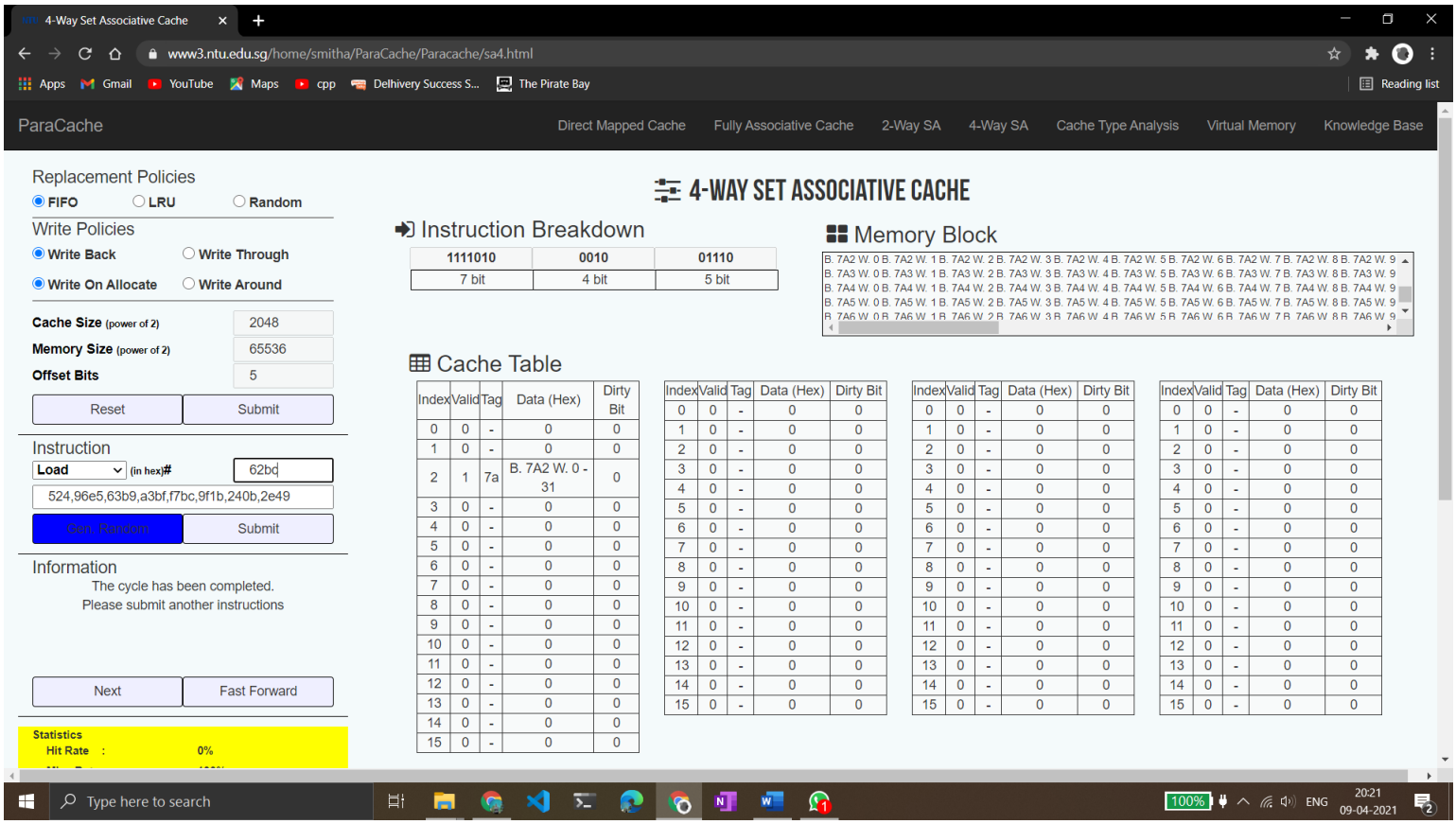
For the above mentioned problem, calculate and execute for 4way set associativity and fully associative mapping technique. For each technique randomly generate ten addresses and indicate whether the cache access will result in a hit or a miss. Assume block replacement policy as random.



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

Statistics

Hit Rate : 0%

Miss Rate : 100%

List of Previous Instructions :

- Load F44E [Miss]
- Load 62BC [Miss]
- Load 524 [Miss]
- Load 96E5 [Miss]
- Load 63B9 [Miss]
- Load A3BF [Miss]
- Load F7BC [Miss]
- Load 9F1B [Miss]
- Load 240B [Miss]
- Load 2E49 [Miss]

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.

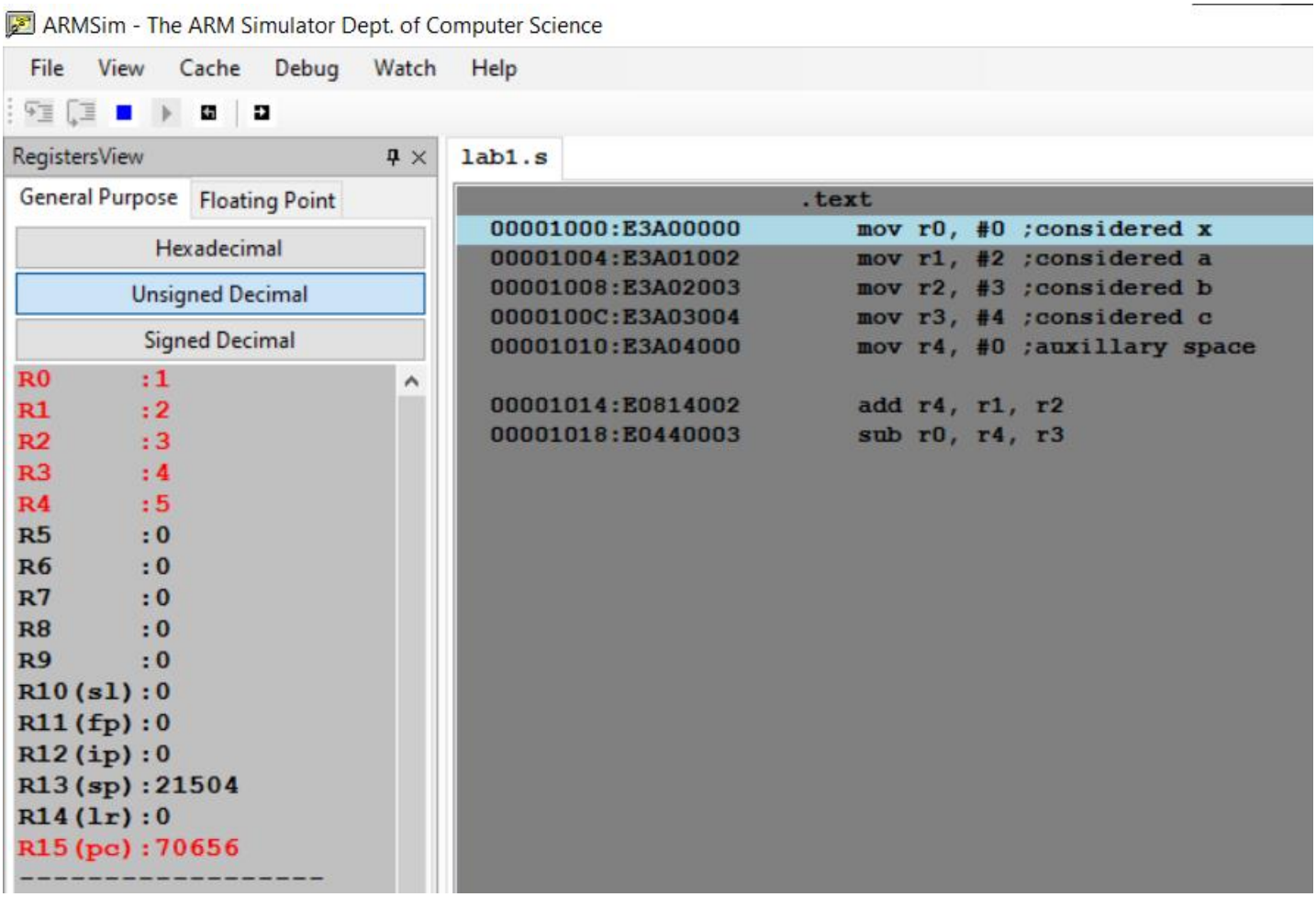
1) $x = (a + b) - c$:

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) $z = (a << 2) \mid (b \& 15)$:

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: