MPCA Lab

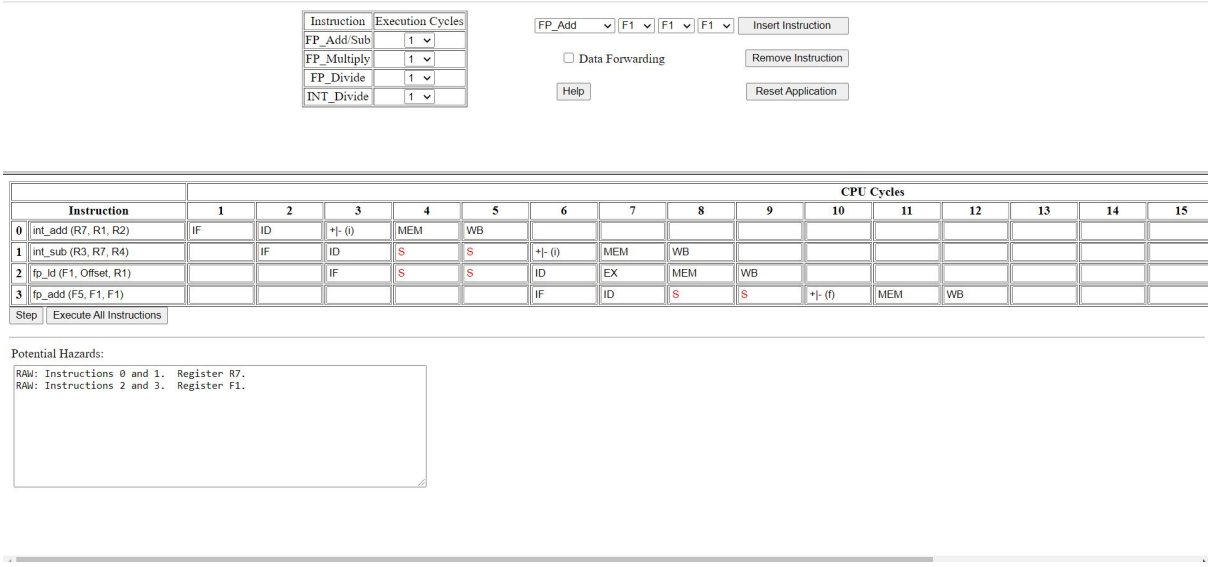
Week 9

Name : Abhishek Aditya BS
SRN : PES1UG19CS019
Section : A

Task 1

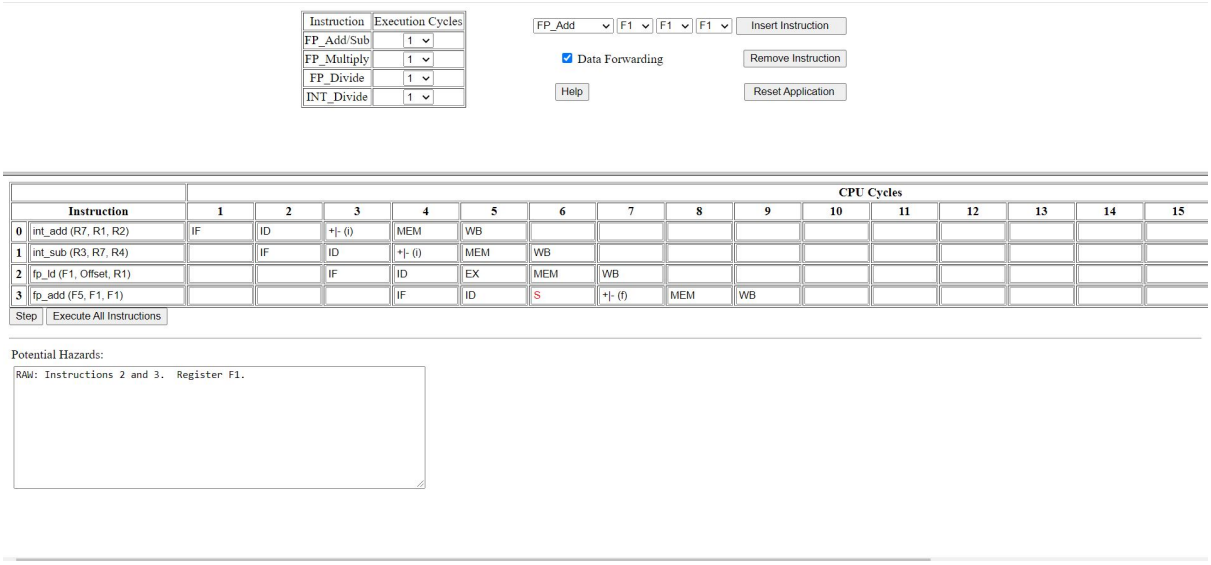
Execute these instructions using 5 stage pipeline - MIPS architecture simulator.

ADD R0, R1, R2
SUB R3, R0, R4
FP_LOAD F1, Offset,R1
FP_ADD F5,F1,F1



Q) Check whether there is data dependency among the instructions?
Ans. Yes, there is a Data Dependency (RAW Hazard)

Q) How many stall states have been introduced?
Ans. 6 Stalls without Data Forwarding



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

Ans. With Data Forwarding 5 Stalls have been reduced.

Q) Mention the total number of clock cycles used with and without data forwarding.

Ans. Total Clock Cycles

→ With Data Forwarding - 9

→ Without Data Forwarding - 12

Task 2

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) 64 bytes per cache block so $2^6 = 64$ so 6 bits is the Word

2k-byte cache = 2048 bytes

2048/64 = 32 , $2^5 = 32$ so 5 bits is the Block field

Given 16 bit memory address , Block + Offset = 5+6= 11

16 bits - 11 bits = 5 bits, so 5 bits is the Tag field

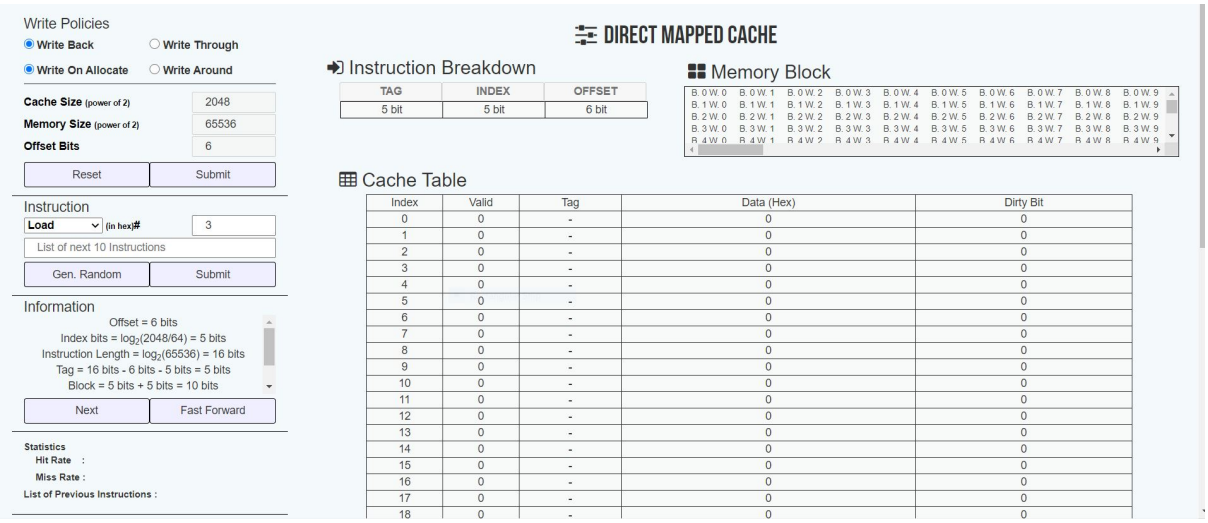
Tag	Block	Word
5 bit	5 bit	6 bit

b) **Direct Mapped Cache**

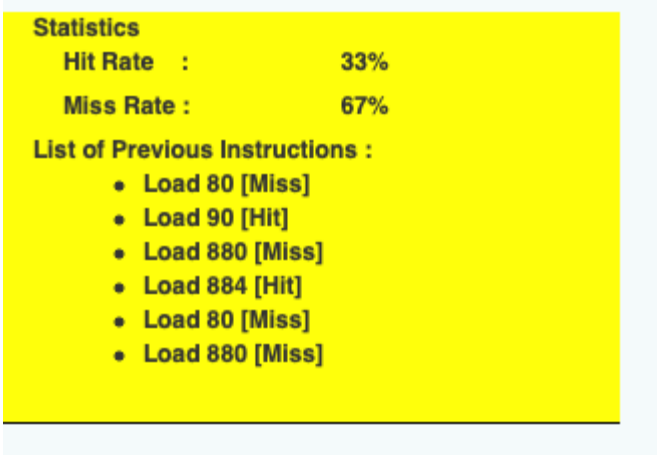
When a program is executed, the processor reads data sequentially from the following word addresses:

Decimal - 128, 144, 2176, 2180, 128, 2176

Hexa - 80, 90, 880, 884, 80, 880

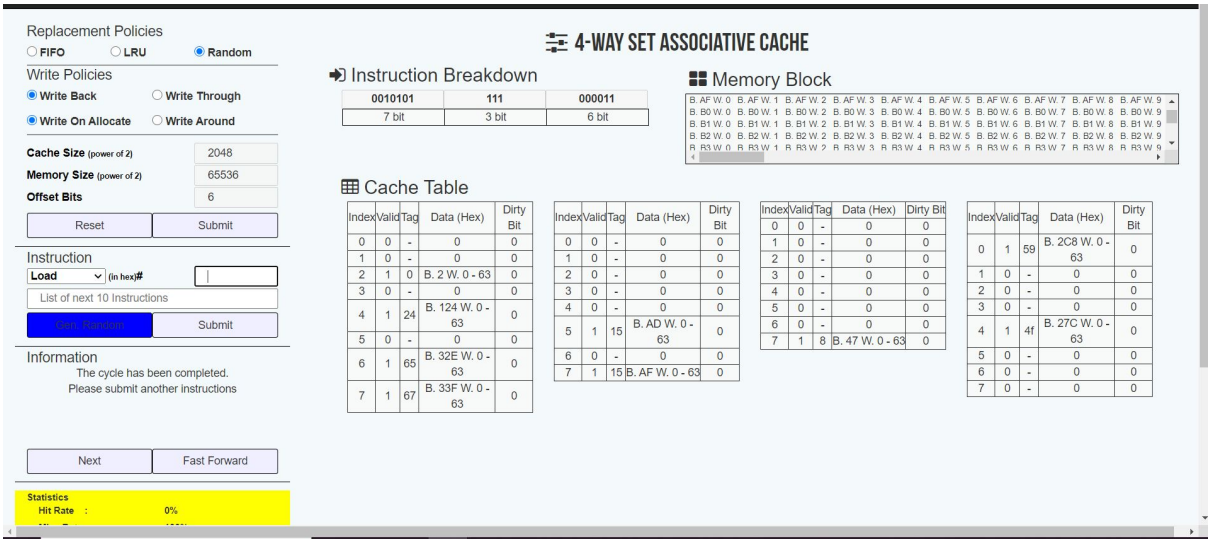


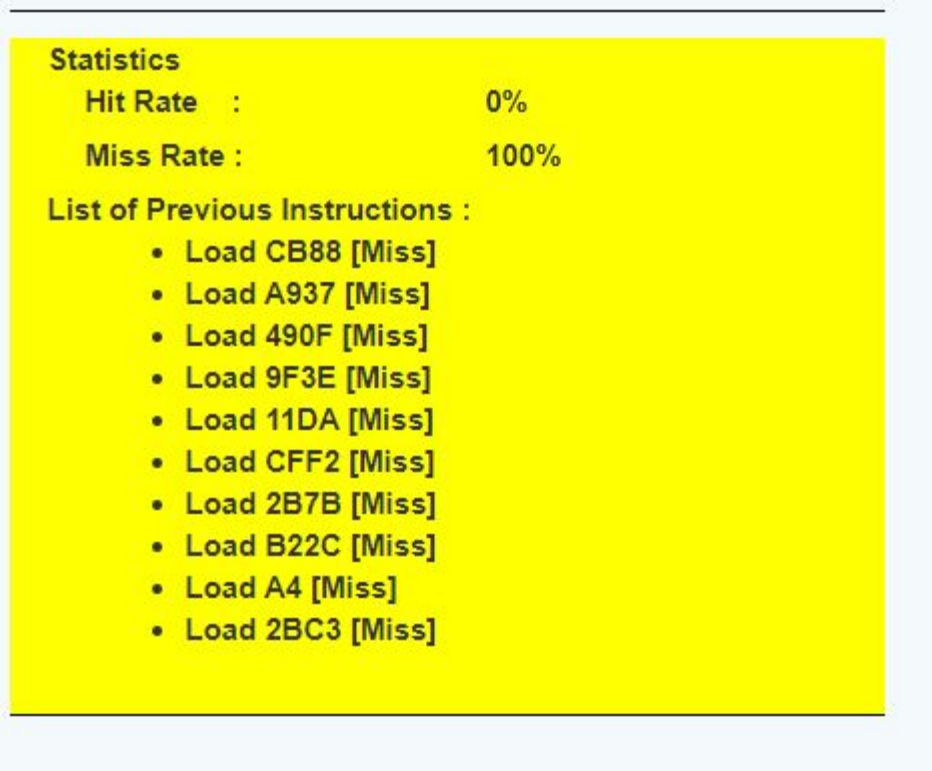
For each of the above addresses, indicate whether the cache access will result in a hit or a miss. Assume cache is empty.



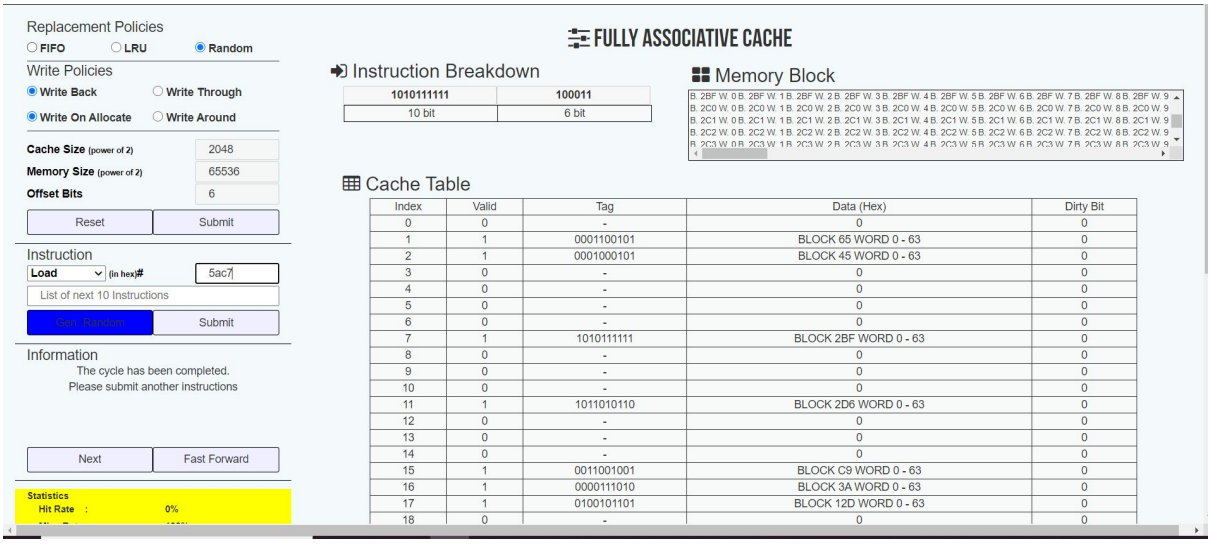
2. For the above mentioned problem, calculate and execute for 4-way set associativity and fully associative mapping technique. Each technique randomly generates ten addresses and indicates whether the cache access will result in a hit or a miss. Assume block replacement policy as random.

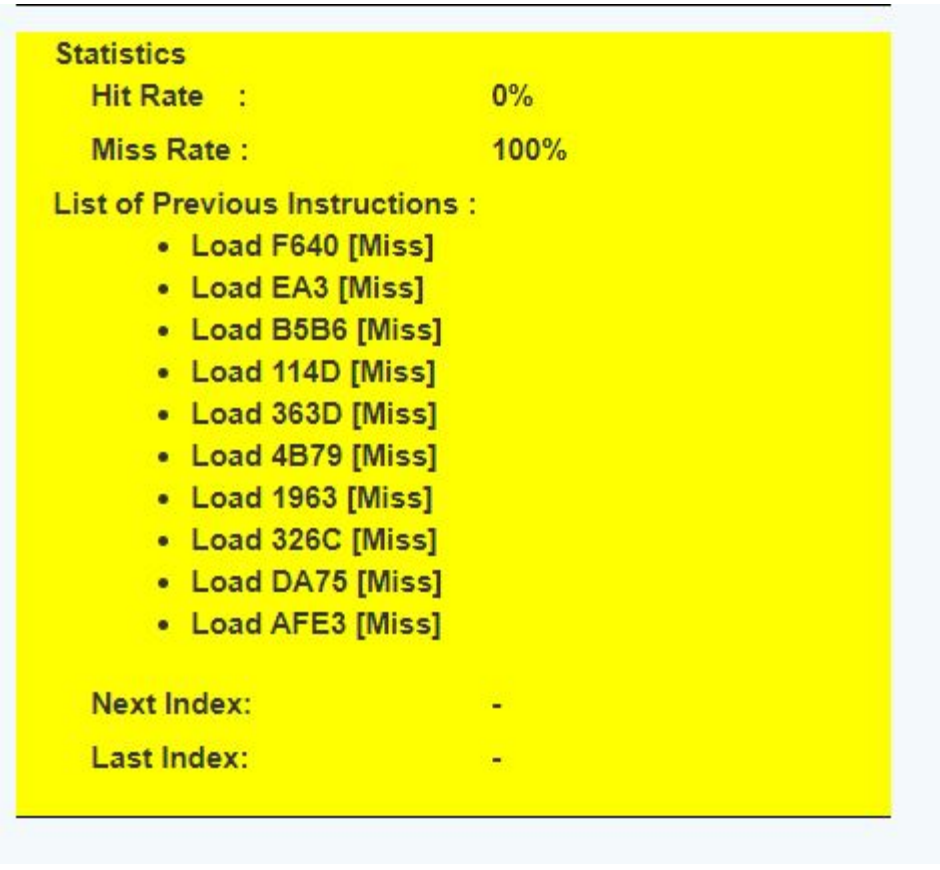
4 Way Set Associative





Fully Associative Cache



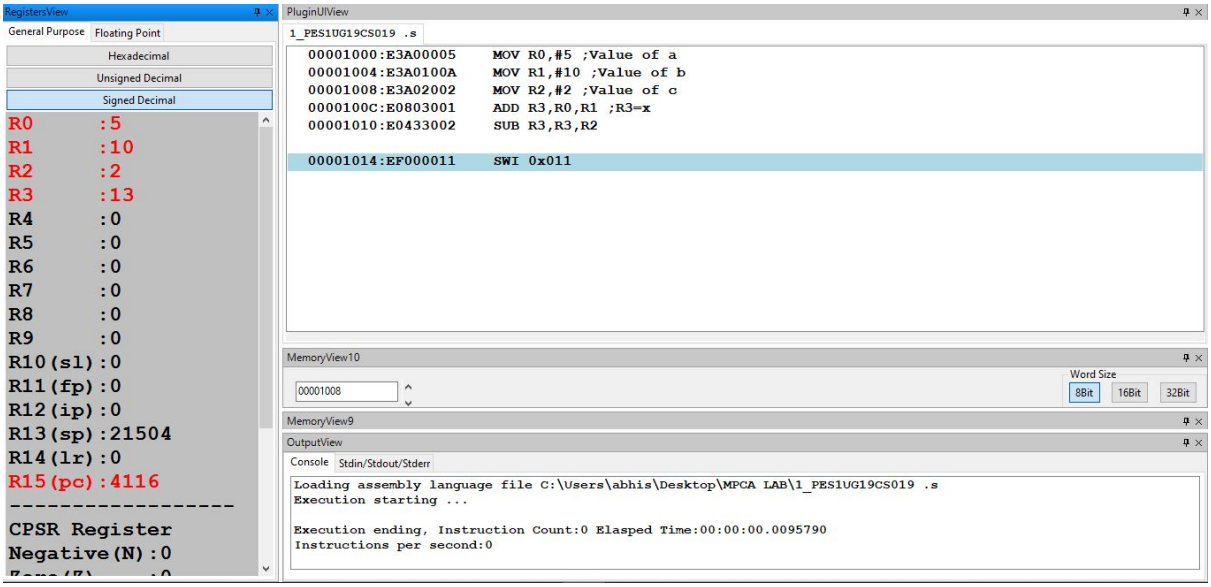


Task 3

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

1) x = (a + b) - c

```
MOV R0,#5 ;Value of a
MOV R1,#10 ;Value of b
MOV R2,#2 ;Value of c
ADD R3,R0,R1 ;R3=x
SUB R3,R3,R2
SWI 0x011
```



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

```
MOV R0,#10 ;Value of a
MOV R1,#20 ;Value of b
AND R2,R1,#15 ;(b & 15)
ORR R3,R2,R0,LSL #2 ;R3 = z
SWI 0x011
```