

**Computer**

**Organization**

**Sarvesh Anand Mankar**

**142203013**

**TY Comp Div-2, T4 Batch**

**Part A: Download and install open-source cache simulator or demonstrator.**

T4: Associativity demonstration and results – Direct Map Caches

**Features of Direct Cache Misses**

1. Associativity: Direct-mapped caches have an associativity of 1, meaning that each cache block in main memory can map to only one specific block in the cache. This simplicity makes them easier to implement and faster in terms of access time compared to higher associativity caches.
2. Mapping Function: The mapping between main memory and cache blocks is determined by a simple mapping function. Typically, it involves dividing the memory address into three fields: the tag, the index, and the offset. The index is used to determine the set in which the block can be placed, and there is only one block per set (direct-mapped).
3. Cache Size: The size of a direct-mapped cache is fixed and is typically smaller than other cache designs. This is because each block in main memory can only be placed in one specific block in the cache, reducing the flexibility of caching.
4. Replacement Policy: Direct-mapped caches have a straightforward replacement policy. Since each set contains only one block, if a new block needs to be loaded into a set that is already occupied, the existing block is replaced.
5. Tag Storage: In addition to the data, each cache block also stores a tag, which is a portion of the memory address. The tag is used to check whether the desired data is present in the cache or not.
6. Speed: Direct-mapped caches are generally faster than set-associative or fully-associative caches for a given size because the mapping is simpler, and there is less contention for access to a specific cache set.
7. Cache Hit and Cache Miss: A cache hit occurs when the data requested is found in the cache. In a direct-mapped cache, this is a straightforward process since there is only one possible location for a given block. A cache miss occurs when the data is not found in the cache, and the required block must be fetched from main memory.

**DineroIV**

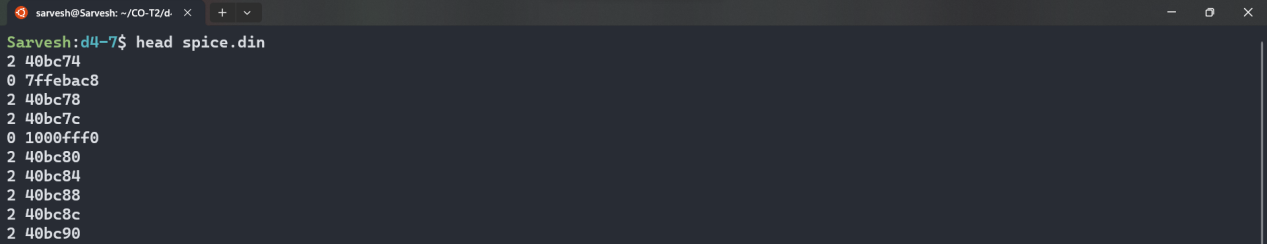
Dinero IV is a cache simulator for memory reference traces. It includes the following major changes over [Dinero III.](http://www.cs.wisc.edu/~larus/warts.html)

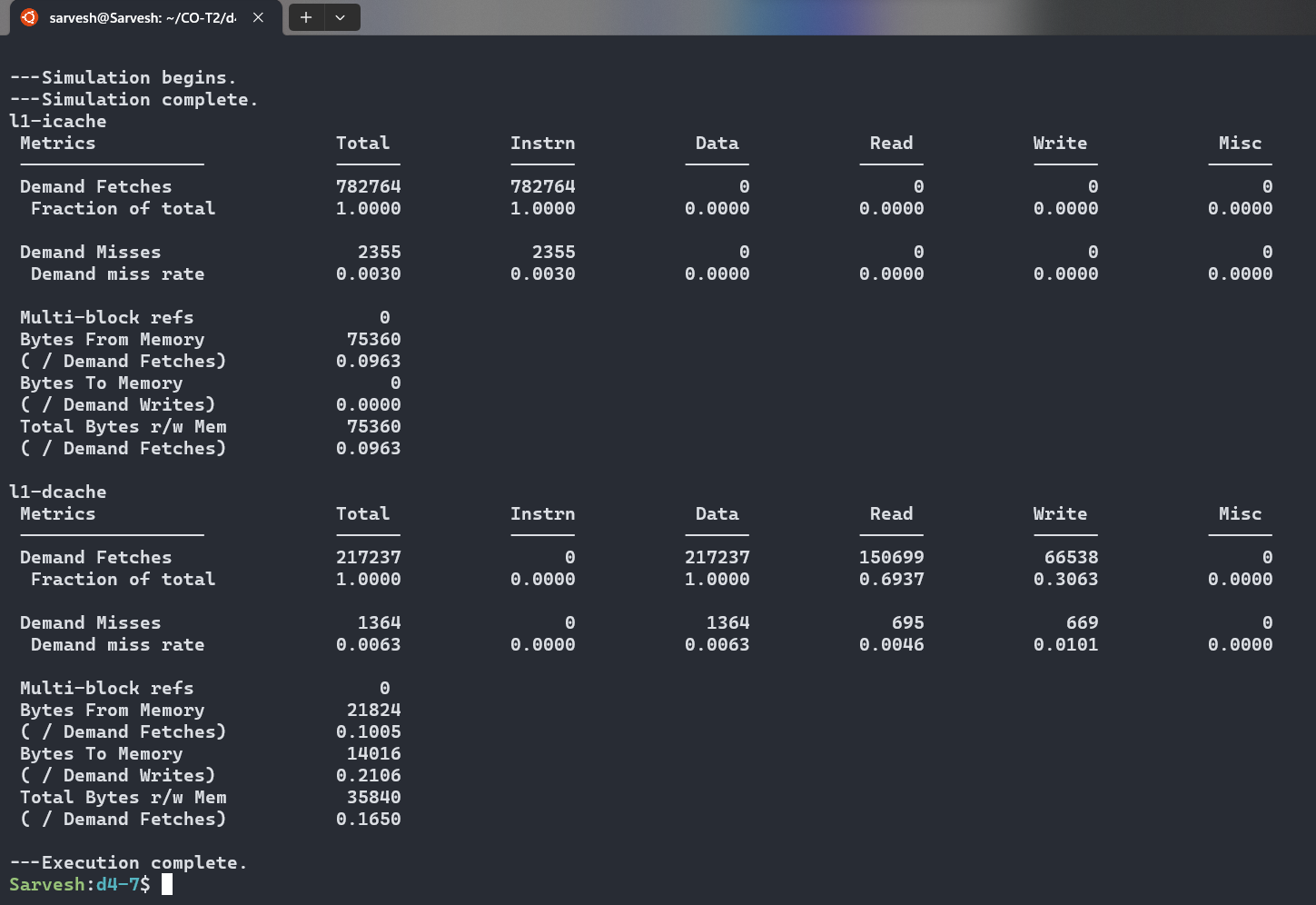
* subroutine-callable interface in addition to trace-reading program
* simulation of multi-level caches
* simulation of dissimilar I and D caches
* better performance, especially for highly associative caches
* classification of compulsory, capacity, and conflict misses
* support for multiple input formats
* cleaned up and modernized code, improved portability

The basic idea is to simulate a memory hierarchy consisting of various caches connected as one or more trees, with reference sources (the processors) at the leaves and a memory at each root. The various parameters of each cache can be set separately (architecture, policy, statistics). During initialization, the configuration to be simulated is built up, one cache at a time, starting with each memory as a special case. After initialization, each reference is fed to the appropriate top-level cache by a single simple function call. Lower levels of the hierarchy are handled automatically.

**Steps to set up DineroIV:**

* gunzip d4-7.tar.gz
* tar -xvf d4-7.tar
* cd d4-7
* ./configure
* make
* wget <http://ace.cs.ohio.edu/~avinashk/classes/ee468/spice.din.z>
* wget <http://ace.cs.ohio.edu/~avinashk/classes/ee468/cc1.din.z>
* gunzip cc1.din.Z
* ./dineroIV -l1-isize 16384 -l1-iassoc 4 -l1-ibsize 32 -l1-irepl l -l1-dsize 32768 -l1-dassoc 2 -l1-dbsize 16 -l1-drepl f -l1-dwalloc a -l1-dwback a -informat d < cc1.din
* gunzip spice.din.Z
* ./dineroIV -l1-isize 16384 -l1-iassoc 4 -l1-ibsize 32 -l1-irepl l -l1-dsize 32768 -l1-dassoc 2 -l1-dbsize 16 -l1-drepl f -l1-dwalloc a -l1-dwback a -informat d < spice.din





Options:

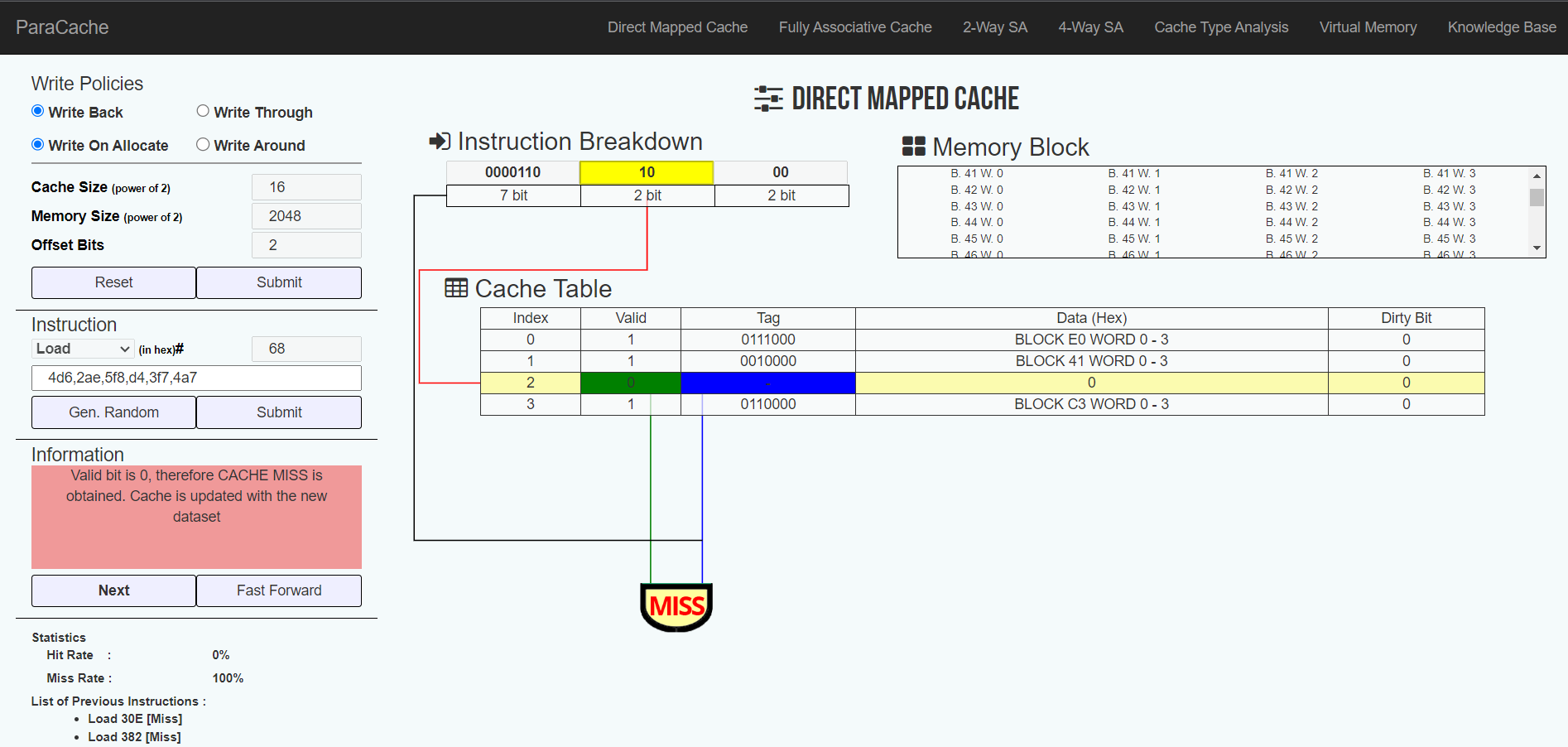
-informat d : Input trace file is in traditional din format.

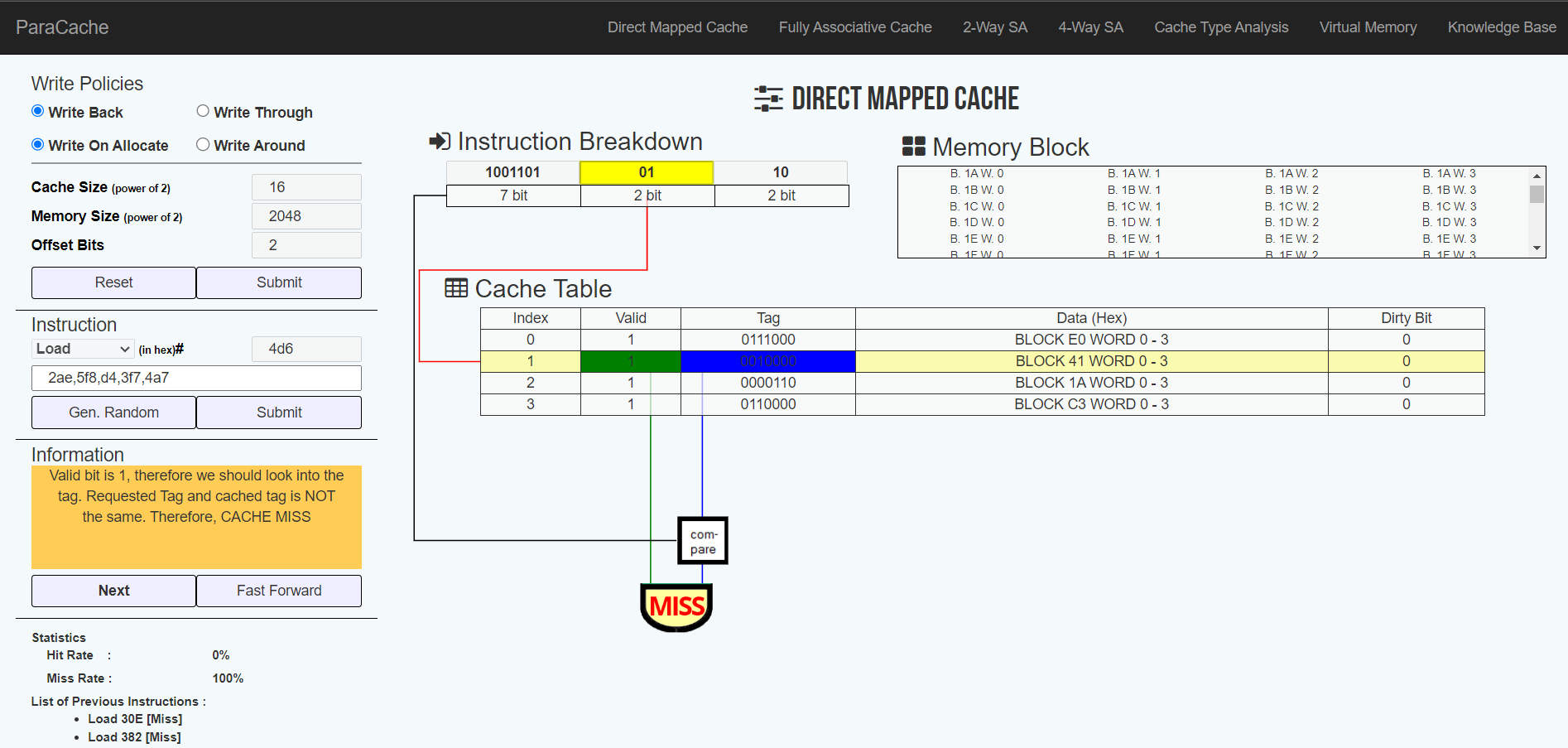
-l1-uassoc 1 : Setting the associativity to 1 as it is a direct mapped cache.

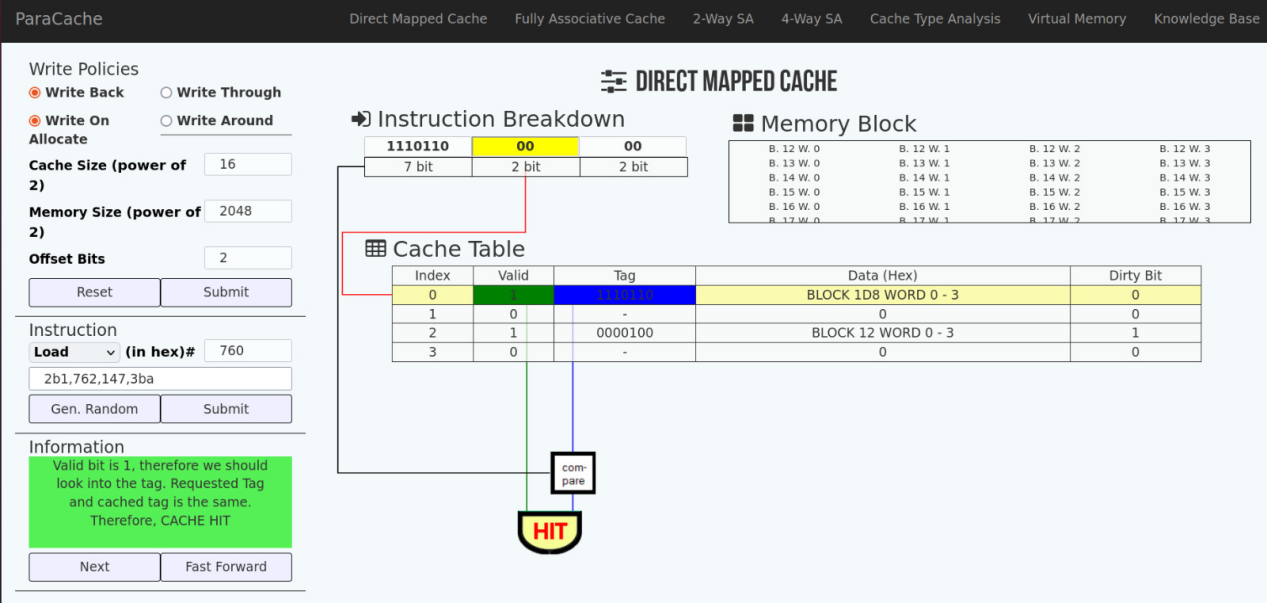
-l1-ubsize 64 : Setting the block size to *64 bytes*.

-l1-usize 32k : Setting the cache size to *32k bytes* or *32768 bytes*.

l1 here means cache level 1 and u means unified cache.







**Conclusion:**

The demand miss rate is relatively low, indicating that a significant portion of memory accesses is satisfied by the L1 Unified Cache.

Observation shows that instruction misses are slightly more frequent than data misses.

The Bytes From Memory and Bytes To Memory metrics provide insight into the

effectiveness of the cache in reducing memory access latencies.

**Part B: Identifying and understanding the**

**dependencies by taking set of instructions with data forwarding.**

**Set 1:**

L.D F6,32(R2)

L.D F2,44(R3)

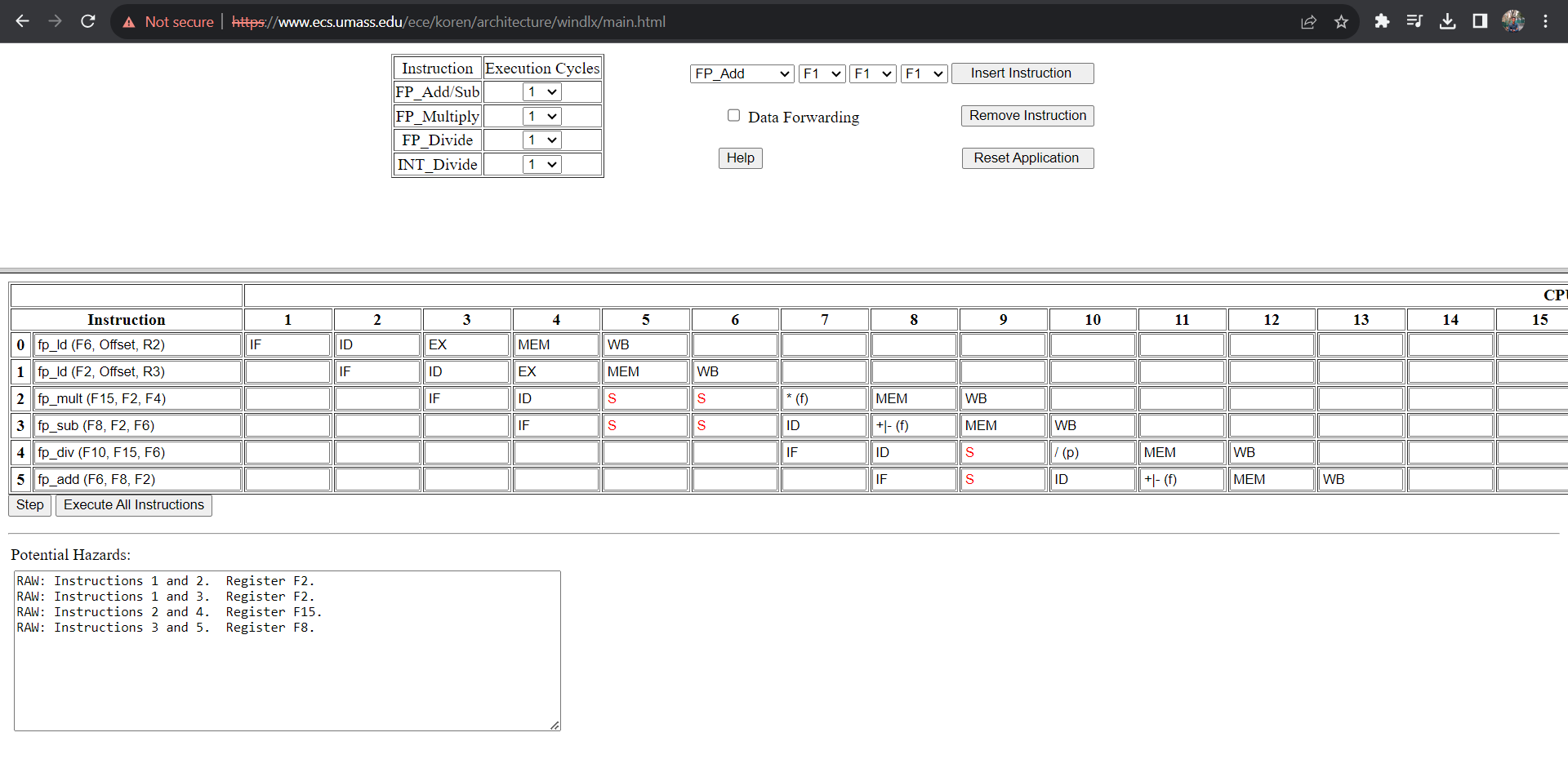
MUL.D F0,F2,F4

SUB.D F8,F2,F6

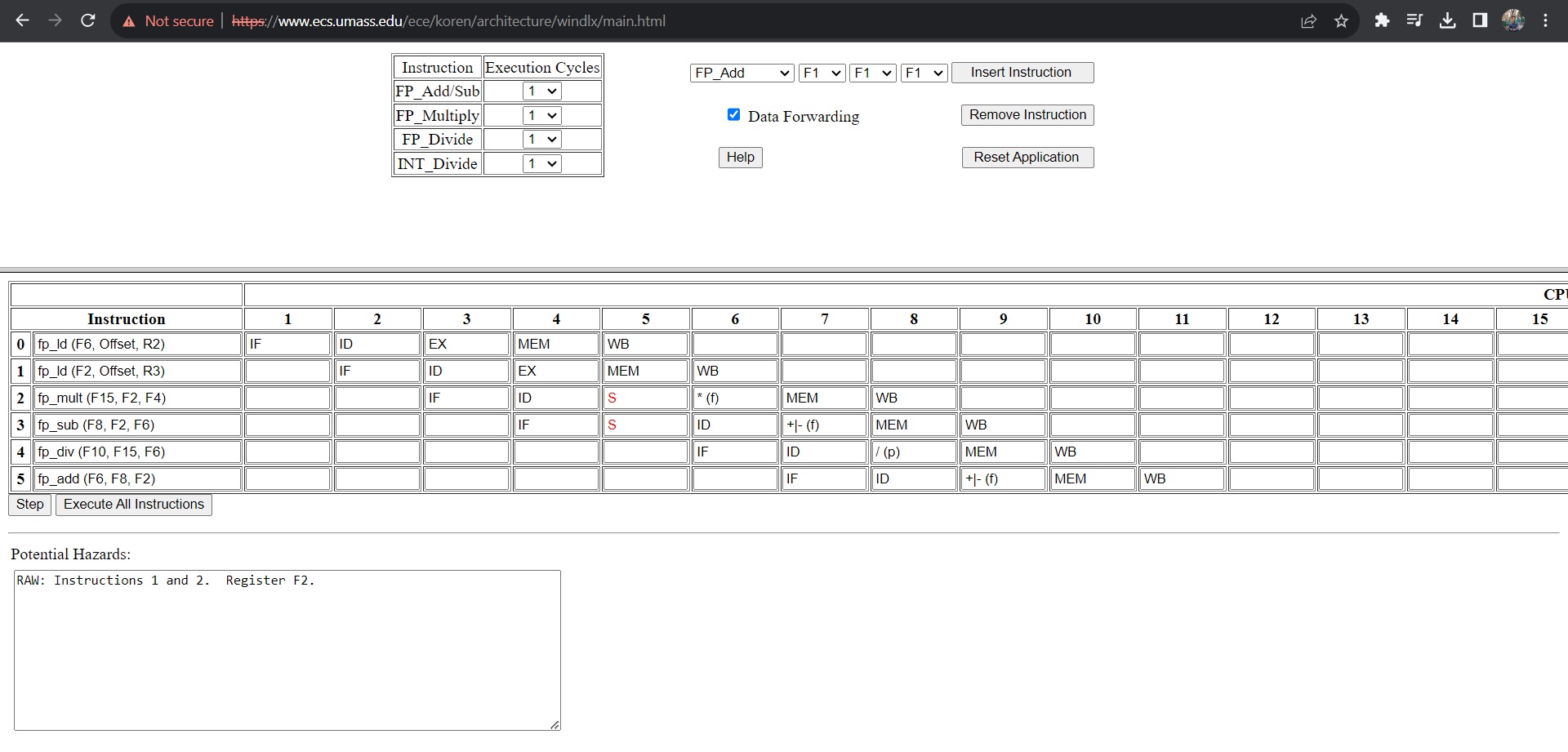
DIV.D F10,F0,F6

ADD.D F6,F8,F2

1. L.D F6,32(R2): No data dependency.
2. L.D F2,44(R3): No data dependency.
3. MUL.D F0,F2,F4: Data dependency on the result of instruction 2 (F2).
4. SUB.D F8,F2,F6: Data dependency on the result of instruction 1 (F6).
5. DIV.D F10,F0,F6: Data dependency on the result of instruction 3 (F0).
6. ADD.D F6,F8,F2: Data dependency on the result of instruction 4 (F8).



*Without Data Forwarding*



*With Data Forwarding*

Set 2:

ADD R1, R2, R3

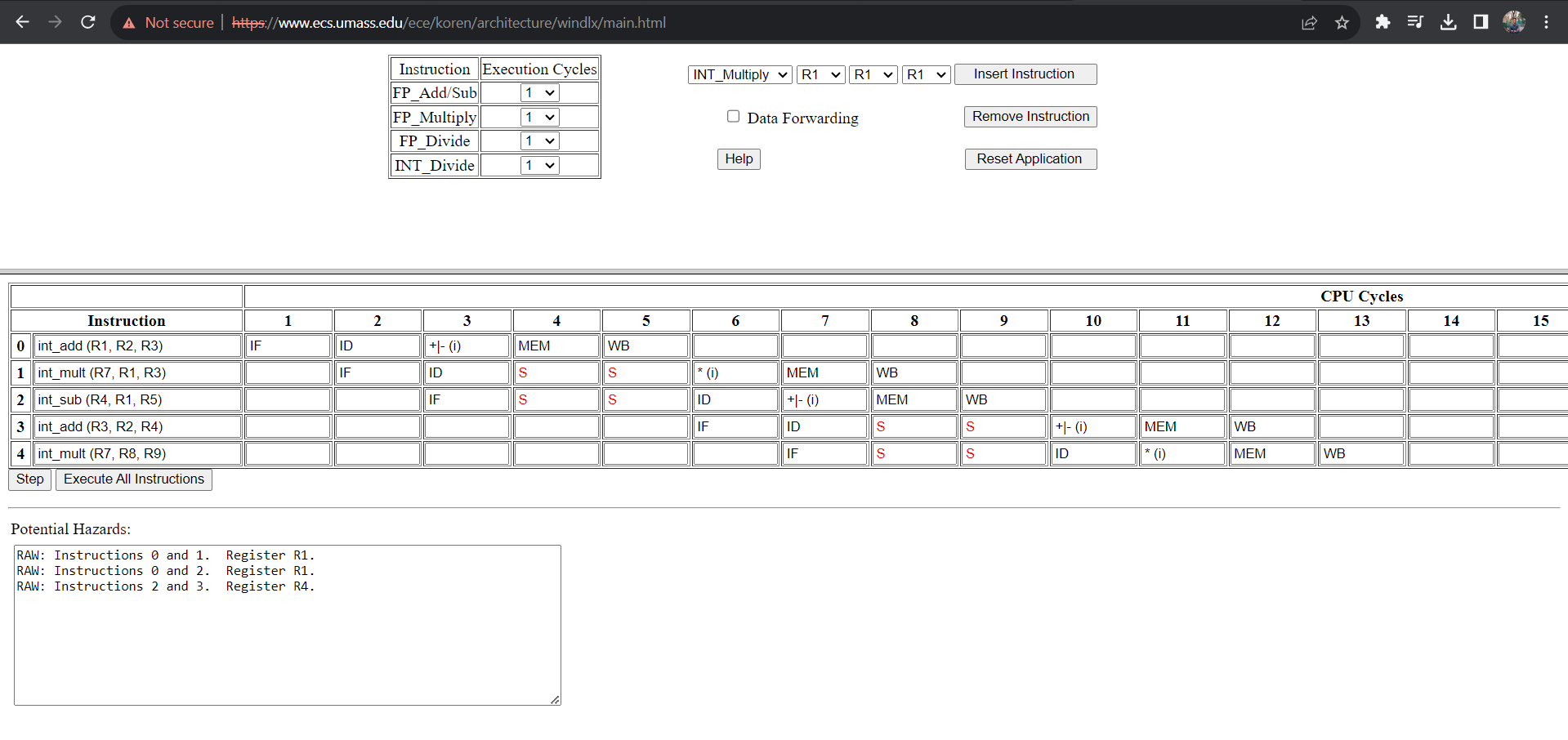
MUL R7, R1, R3

SUB R4, R1, R5

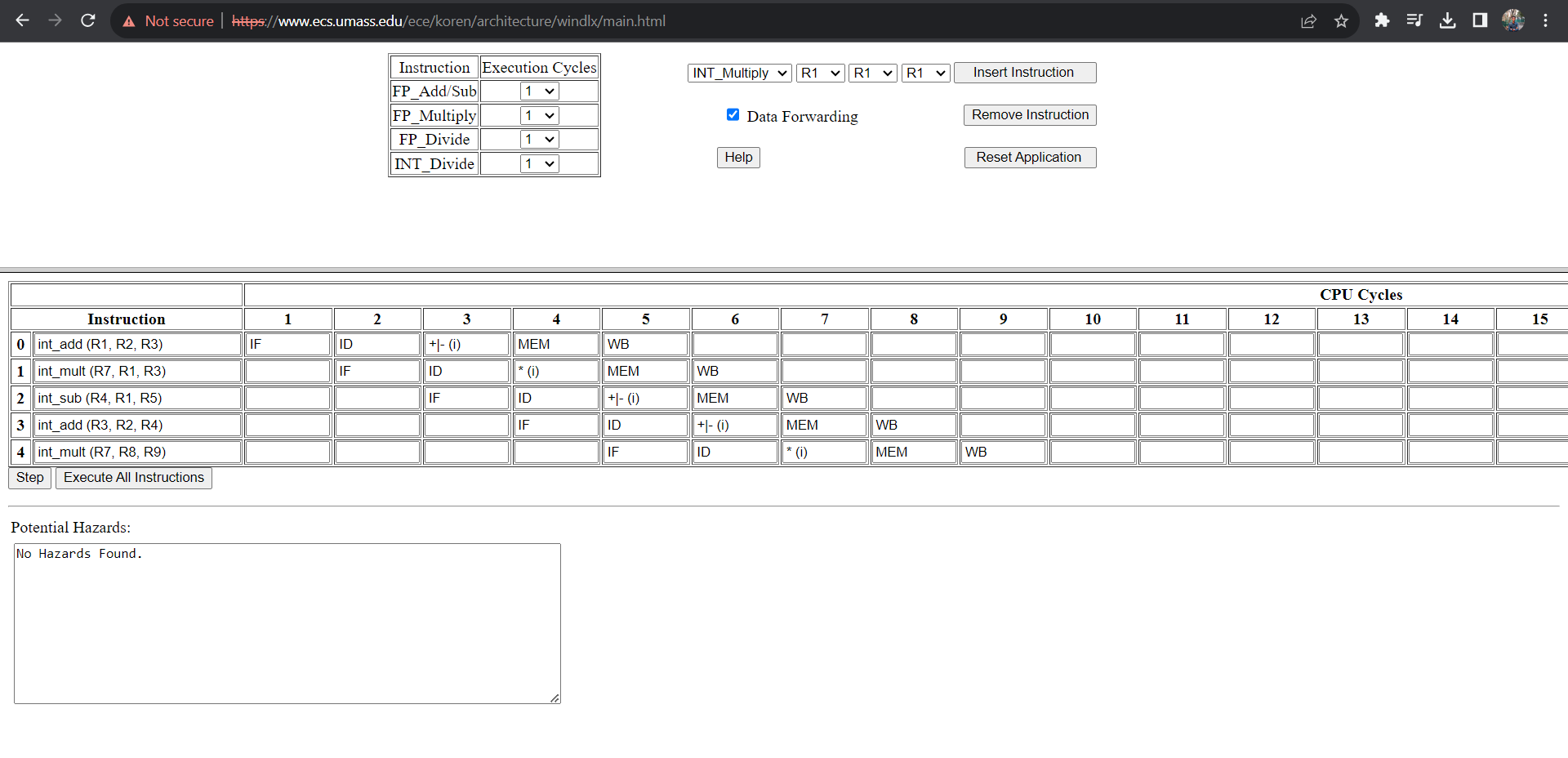
ADD R3, R2, R4

MUL R7, R8, R9

1. ADD R1, R2, R3:No data dependency.
2. MUL R7, R1, R3: Data dependency on the result of instruction 1 (R1).
3. SUB R4, R1, R5: Data dependency on the result of instruction 1 (R1).
4. ADD R3, R2, R4: Data dependency on the result of instruction 3 (R4).
5. MUL R7, R8, R9: No data dependency.



*Without Data Forwarding*



*With Data Forwarding*

**Conclusion:**

To decrease stalls and boost pipeline efficiency overall, data forwarding can be employed to store results in intermediate registers between pipeline segments. This keeps the instruction throughput greater and allows the pipeline to run more smoothly.