# 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

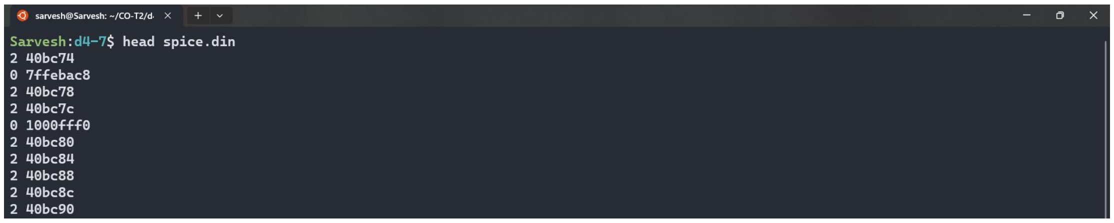
Dinero IV is a cache simulator for memory reference traces. It includes the following major changes over Dinero III.

- 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

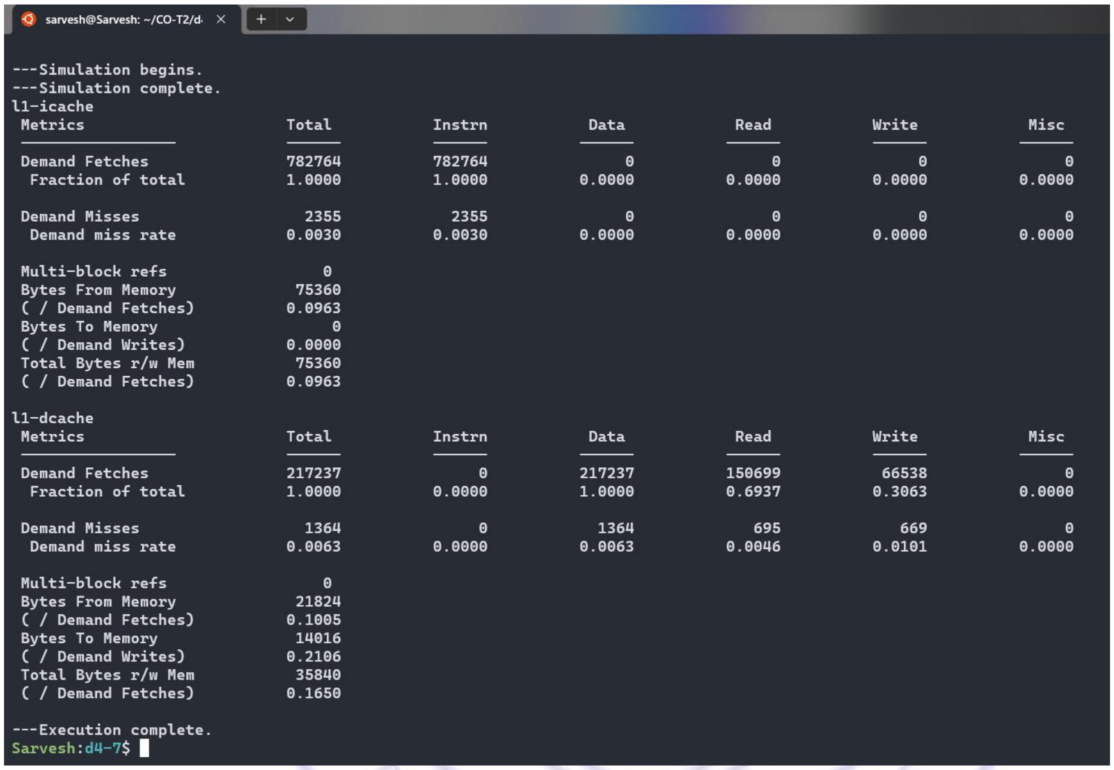
The basic idea is to simulate a memory hierarchy consisting of various caches connected as one or more trees, with reference sources (the processors) at the leaves and a memory at each root. The various parameters of each cache can be set separately (architecture, policy, statistics). During initialization, 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



```
Sarvesh:d4-7$ head spice.din
2 40bc74
0 7ffebac8
2 40bc78
2 40bc7c
0 1000fff0
2 40bc80
2 40bc84
2 40bc88
2 40bc8c
2 40bc90
```



```
---Simulation begins.
---Simulation complete.
l1-icache
 Metrics                      Total           Instrn           Data            Read           Write            Misc
 -----------------            ------          ------          ------          ------          ------          ------
 Demand Fetches               782764          782764               0               0               0               0
  Fraction of total           1.0000          1.0000          0.0000          0.0000          0.0000          0.0000

 Demand Misses                  2355            2355               0               0               0               0
  Demand miss rate            0.0030          0.0030          0.0000          0.0000          0.0000          0.0000

 Multi-block refs                  0
 Bytes From Memory             75360
 ( / Demand Fetches)          0.0963
 Bytes To Memory                   0
 ( / Demand Writes)           0.0000
 Total Bytes r/w Mem           75360
 ( / Demand Fetches)          0.0963

l1-dcache
 Metrics                      Total           Instrn           Data            Read           Write            Misc
 -----------------            ------          ------          ------          ------          ------          ------
 Demand Fetches               217237               0          217237          150699           66538               0
  Fraction of total           1.0000          0.0000          1.0000          0.6937          0.3063          0.0000

 Demand Misses                  1364               0            1364             695             669               0
  Demand miss rate            0.0063          0.0000          0.0063          0.0046          0.0101          0.0000

 Multi-block refs                  0
 Bytes From Memory             21824
 ( / Demand Fetches)          0.1005
 Bytes To Memory               14016
 ( / Demand Writes)           0.2106
 Total Bytes r/w Mem           35840
 ( / Demand Fetches)          0.1650

---Execution complete.
Sarvesh:d4-7$
```

## Options:

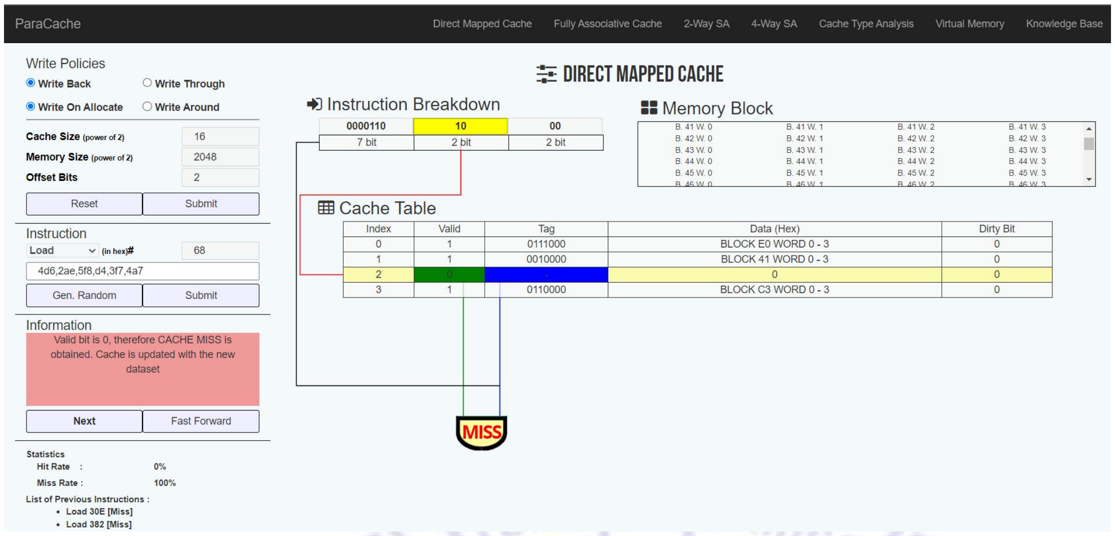
-informat d : Input trace file is in traditional din format.

-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.



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) | 16 |
| Memory Size (power of 2) | 2048 |
| Offset Bits | 2 |

Reset | Submit

Instruction
Load (in hex)# 68
4d6,2ae,5f8,d4,3f7,4a7
Gen. Random | Submit

Information
Valid bit is 0, therefore CACHE MISS is obtained. Cache is updated with the new dataset

Next | Fast Forward

Statistics
Hit Rate : 0%
Miss Rate : 100%
List of Previous Instructions :
- Load 30E [Miss]
- Load 382 [Miss]

DIRECT MAPPED CACHE

Instruction Breakdown

| 0000110 | 10 | 00 |
|---|---|---|
| 7 bit | 2 bit | 2 bit |

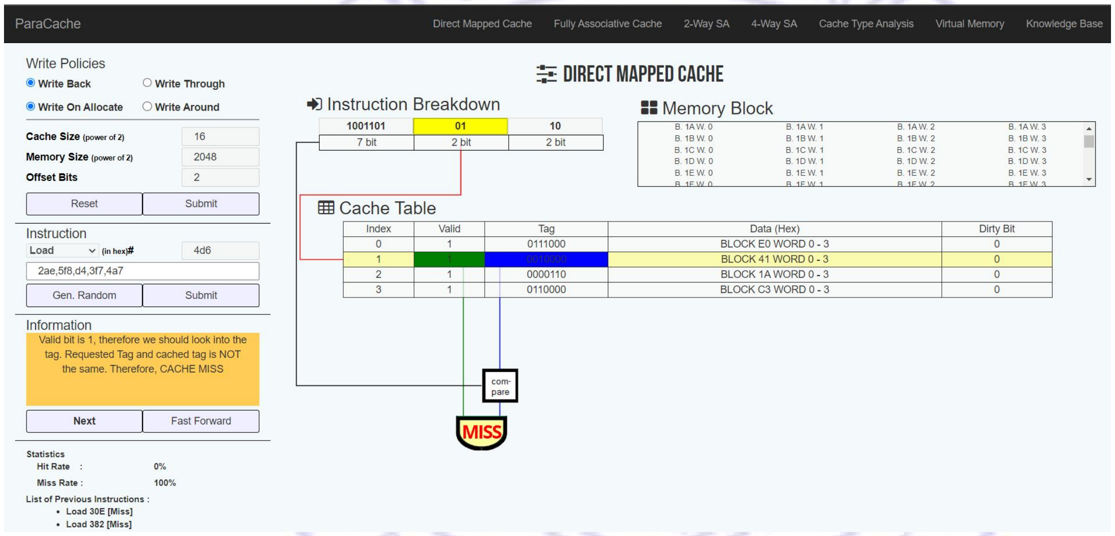
Memory Block

| | | | |
|---|---|---|---|
| B. 41 W. 0 | B. 41 W. 1 | B. 41 W. 2 | B. 41 W. 3 |
| B. 42 W. 0 | B. 42 W. 1 | B. 42 W. 2 | B. 42 W. 3 |
| B. 43 W. 0 | B. 43 W. 1 | B. 43 W. 2 | B. 43 W. 3 |
| B. 44 W. 0 | B. 44 W. 1 | B. 44 W. 2 | B. 44 W. 3 |
| B. 45 W. 0 | B. 45 W. 1 | B. 45 W. 2 | B. 45 W. 3 |

Cache Table

| Index | Valid | Tag | Data (Hex) | Dirty Bit |
|---|---|---|---|---|
| 0 | 1 | 0111000 | BLOCK E0 WORD 0 - 3 | 0 |
| 1 | 1 | 0010000 | BLOCK 41 WORD 0 - 3 | 0 |
| 2 | 0 | - | 0 | 0 |
| 3 | 1 | 0110000 | BLOCK C3 WORD 0 - 3 | 0 |

MISS



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) | 16 |
| Memory Size (power of 2) | 2048 |
| Offset Bits | 2 |

Reset | Submit

Instruction
Load (in hex)# 4d6
2ae,5f8,d4,3f7,4a7
Gen. Random | Submit

Information
Valid bit is 1, therefore we should look into the tag. Requested Tag and cached tag is NOT the same. Therefore, CACHE MISS

Next | Fast Forward

Statistics
Hit Rate : 0%
Miss Rate : 100%
List of Previous Instructions :
- Load 30E [Miss]
- Load 382 [Miss]

DIRECT MAPPED CACHE

Instruction Breakdown

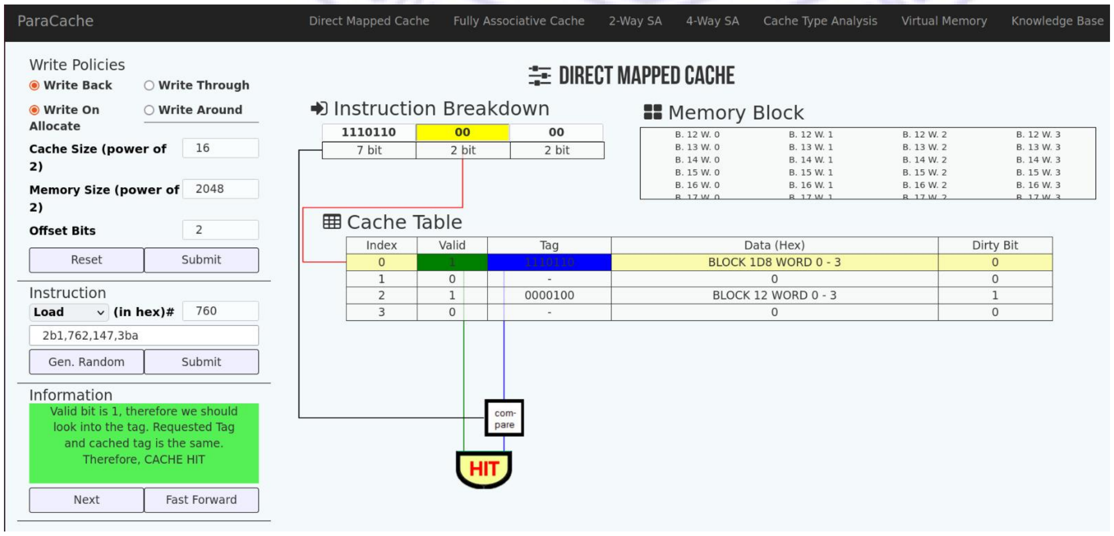
| 1001101 | 01 | 10 |
|---|---|---|
| 7 bit | 2 bit | 2 bit |

Memory Block

| | | | |
|---|---|---|---|
| B. 1A W. 0 | B. 1A W. 1 | B. 1A W. 2 | B. 1A W. 3 |
| B. 1B W. 0 | B. 1B W. 1 | B. 1B W. 2 | B. 1B W. 3 |
| B. 1C W. 0 | B. 1C W. 1 | B. 1C W. 2 | B. 1C W. 3 |
| B. 1D W. 0 | B. 1D W. 1 | B. 1D W. 2 | B. 1D W. 3 |
| B. 1E W. 0 | B. 1E W. 1 | B. 1E W. 2 | B. 1E W. 3 |

Cache Table

| Index | Valid | Tag | Data (Hex) | Dirty Bit |
|---|---|---|---|---|
| 0 | 1 | 0111000 | BLOCK E0 WORD 0 - 3 | 0 |
| 1 | 1 | 0010000 | BLOCK 41 WORD 0 - 3 | 0 |
| 2 | 1 | 0000110 | BLOCK 1A WORD 0 - 3 | 0 |
| 3 | 1 | 0110000 | BLOCK C3 WORD 0 - 3 | 0 |



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) | 16 |
| Memory Size (power of 2) | 2048 |
| Offset Bits | 2 |

Reset | Submit

Instruction
Load (in hex)# 760
2b1,762,147,3ba
Gen. Random | Submit

Information
Valid bit is 1, therefore we should look into the tag. Requested Tag and cached tag is the same. Therefore, CACHE HIT

Next | Fast Forward

DIRECT MAPPED CACHE

Instruction Breakdown

| 1110110 | 00 | 00 |
|---|---|---|
| 7 bit | 2 bit | 2 bit |

Memory Block

| | | | |
|---|---|---|---|
| B. 12 W. 0 | B. 12 W. 1 | B. 12 W. 2 | B. 12 W. 3 |
| B. 13 W. 0 | B. 13 W. 1 | B. 13 W. 2 | B. 13 W. 3 |
| B. 14 W. 0 | B. 14 W. 1 | B. 14 W. 2 | B. 14 W. 3 |
| B. 15 W. 0 | B. 15 W. 1 | B. 15 W. 2 | B. 15 W. 3 |
| B. 16 W. 0 | B. 16 W. 1 | B. 16 W. 2 | B. 16 W. 3 |

Cache Table

| Index | Valid | Tag | Data (Hex) | Dirty Bit |
|---|---|---|---|---|
| 0 | 1 | 1110110 | BLOCK 1D8 WORD 0 - 3 | 0 |
| 1 | 0 | - | 0 | 0 |
| 2 | 1 | 0000100 | BLOCK 12 WORD 0 - 3 | 1 |
| 3 | 0 | - | 0 | 0 |

## 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)

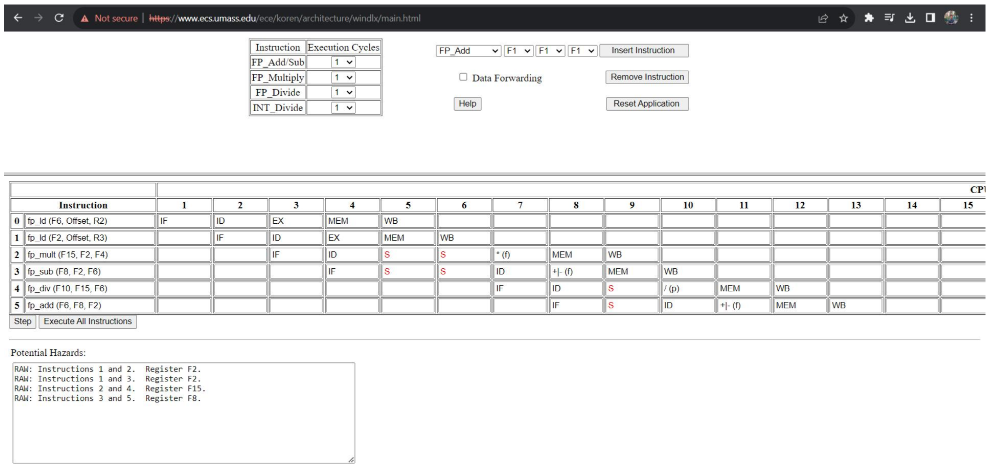
MUL.D F0,F2,F4

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).



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

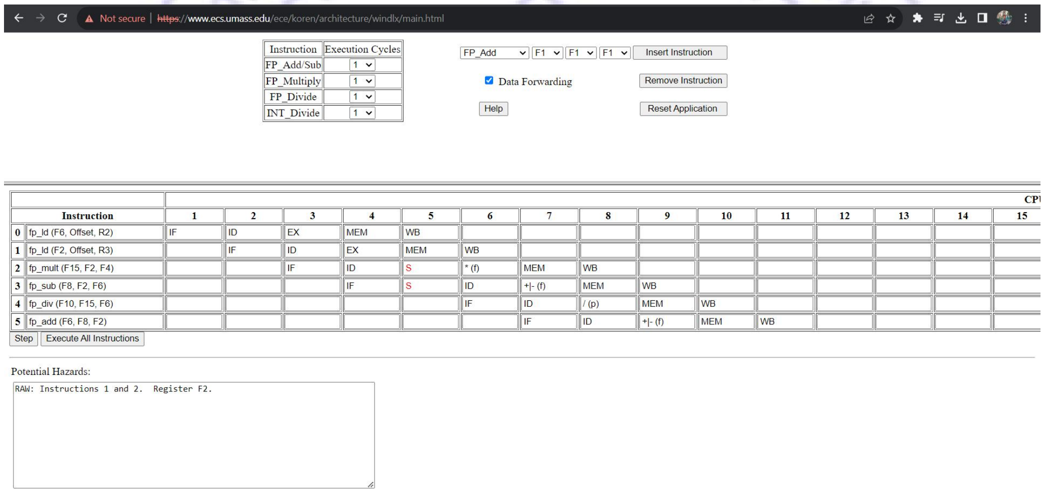
☐ Data Forwarding

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | CP |
| | Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | fp_ld (F6, Offset, R2) | IF | ID | EX | MEM | WB | | | | | | | | | | |
| 1 | fp_ld (F2, Offset, R3) | | IF | ID | EX | MEM | WB | | | | | | | | | |
| 2 | fp_mult (F15, F2, F4) | | | IF | ID | S | S | * (f) | MEM | WB | | | | | | |
| 3 | fp_sub (F8, F2, F6) | | | | IF | S | S | ID | +\|- (f) | MEM | WB | | | | | |
| 4 | fp_div (F10, F15, F6) | | | | | | | IF | ID | S | / (p) | MEM | WB | | | |
| 5 | fp_add (F6, F8, F2) | | | | | | | | IF | S | ID | +\|- (f) | MEM | WB | | |

Potential Hazards:

```
RAW: Instructions 1 and 2.  Register F2.
RAW: Instructions 1 and 3.  Register F2.
RAW: Instructions 2 and 4.  Register F15.
RAW: Instructions 3 and 5.  Register F8.
```

*Without Data Forwarding*



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

☑ Data Forwarding

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | CP |
| | Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | fp_ld (F6, Offset, R2) | IF | ID | EX | MEM | WB | | | | | | | | | | |
| 1 | fp_ld (F2, Offset, R3) | | IF | ID | EX | MEM | WB | | | | | | | | | |
| 2 | fp_mult (F15, F2, F4) | | | IF | ID | S | * (f) | MEM | WB | | | | | | | |
| 3 | fp_sub (F8, F2, F6) | | | | IF | S | ID | +\|- (f) | MEM | WB | | | | | | |
| 4 | fp_div (F10, F15, F6) | | | | | | IF | ID | / (p) | MEM | WB | | | | | |
| 5 | fp_add (F6, F8, F2) | | | | | | | IF | ID | +\|- (f) | MEM | WB | | | | |

Potential Hazards:

```
RAW: Instructions 1 and 2.  Register F2.
```

*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

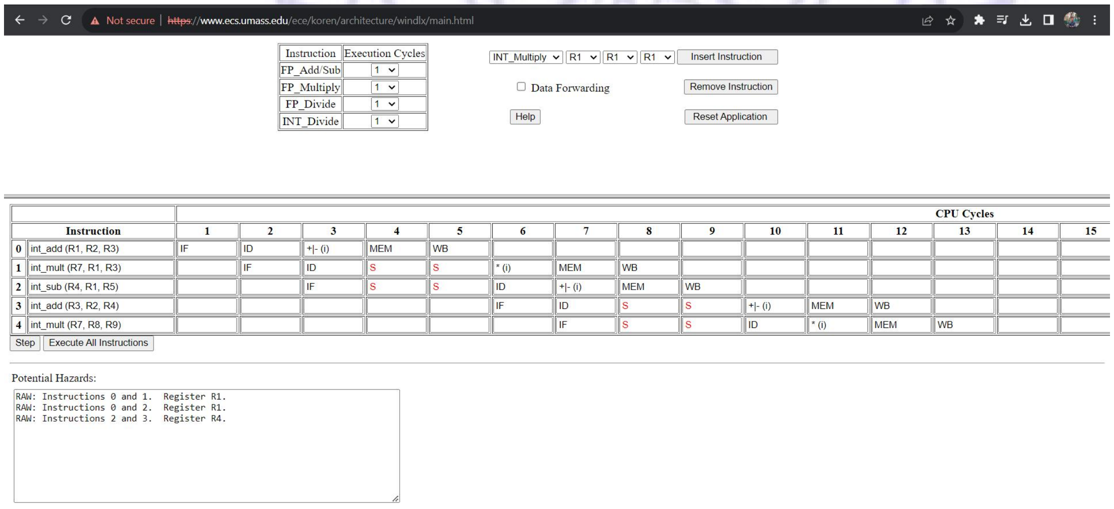
1. ADD R1, R2, R3:No data dependency.

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.



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

INT_Multiply R1 R1 R1

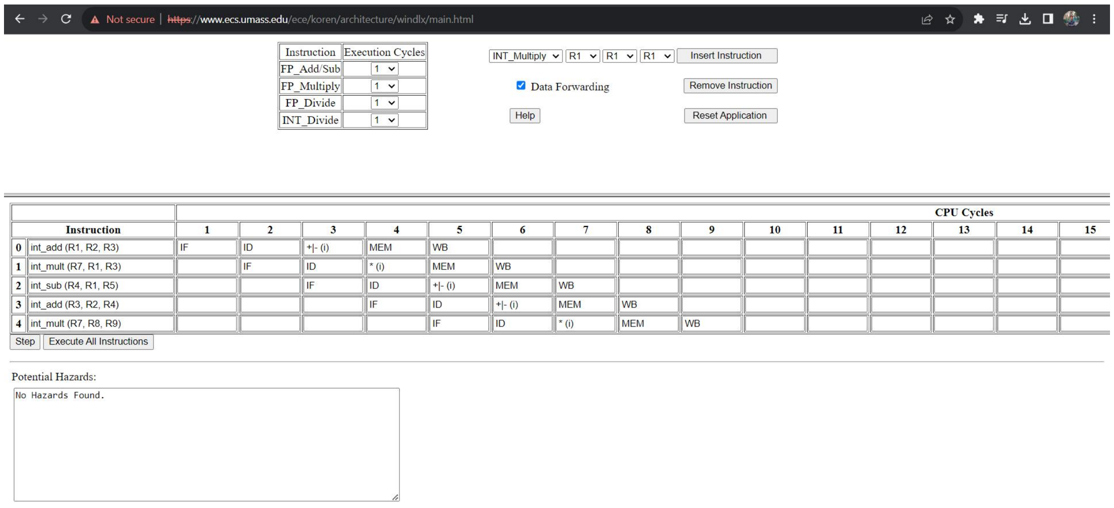
☐ Data Forwarding

| | Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | int_add (R1, R2, R3) | IF | ID | +\|- (i) | MEM | WB | | | | | | | | | | |
| 1 | int_mult (R7, R1, R3) | | IF | ID | S | S | * (i) | MEM | WB | | | | | | | |
| 2 | int_sub (R4, R1, R5) | | | IF | S | S | ID | +\|- (i) | MEM | WB | | | | | | |
| 3 | int_add (R3, R2, R4) | | | | | | IF | ID | S | S | +\|- (i) | MEM | WB | | | |
| 4 | int_mult (R7, R8, R9) | | | | | | | IF | S | S | ID | * (i) | MEM | WB | | |

Potential Hazards:

```
RAW: Instructions 0 and 1.  Register R1.
RAW: Instructions 0 and 2.  Register R1.
RAW: Instructions 2 and 3.  Register R4.
```

*Without Data Forwarding*



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

Data Forwarding ✓

| | | CPU Cycles | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | int_add (R1, R2, R3) | IF | ID | +\|- (i) | MEM | WB | | | | | | | | | | |
| 1 | int_mult (R7, R1, R3) | | IF | ID | * (i) | MEM | WB | | | | | | | | | |
| 2 | int_sub (R4, R1, R5) | | | IF | ID | +\|- (i) | MEM | WB | | | | | | | | |
| 3 | int_add (R3, R2, R4) | | | | IF | ID | +\|- (i) | MEM | WB | | | | | | | |
| 4 | int_mult (R7, R8, R9) | | | | | IF | ID | * (i) | MEM | WB | | | | | | |

Potential Hazards:

```
No Hazards Found.
```

*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.