MPCA Lab Week 9: Working on Pipeline and Cache simulator

Task1: 5 stage pipeline Simulator.

Link: [MIPS Five Stage Pipeline](#) (umass.edu)

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

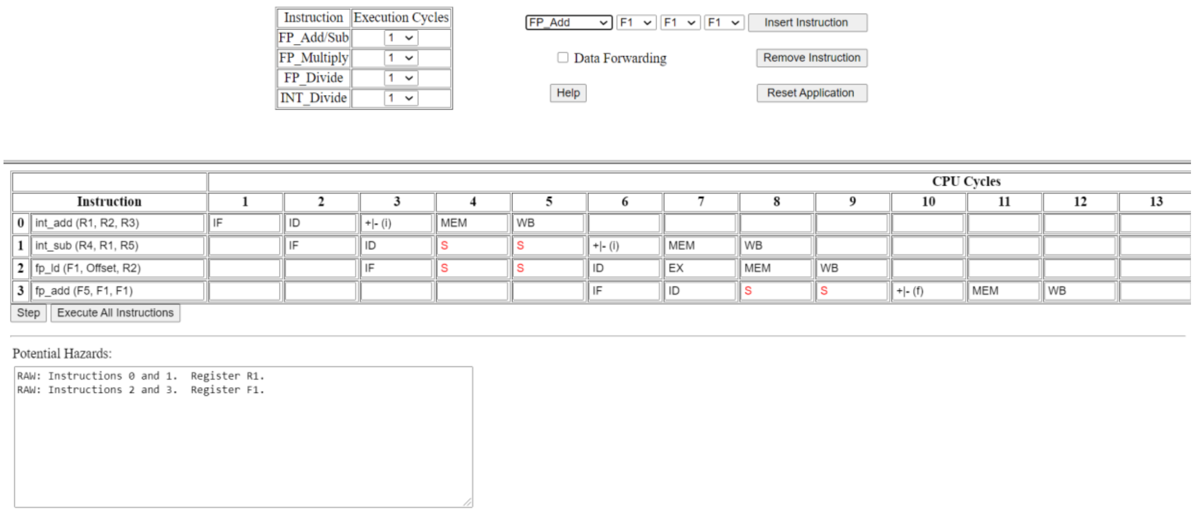
ADD R0, R1, R2

SUB R3, R0, R4

FP_LOAD F1, Offset,R1

FP_ADD F5,F1,F1

Screenshot of 5stage pipeline simulator when above instructions are executed without data forwarding:



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

		CPU Cycles												
	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13
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, R2)			IF	S	S	ID	EX	MEM	WB				
3	fp_add (F5, F1, F1)						IF	ID	S	S	+|- (f)	MEM	WB	

Potential Hazards:

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

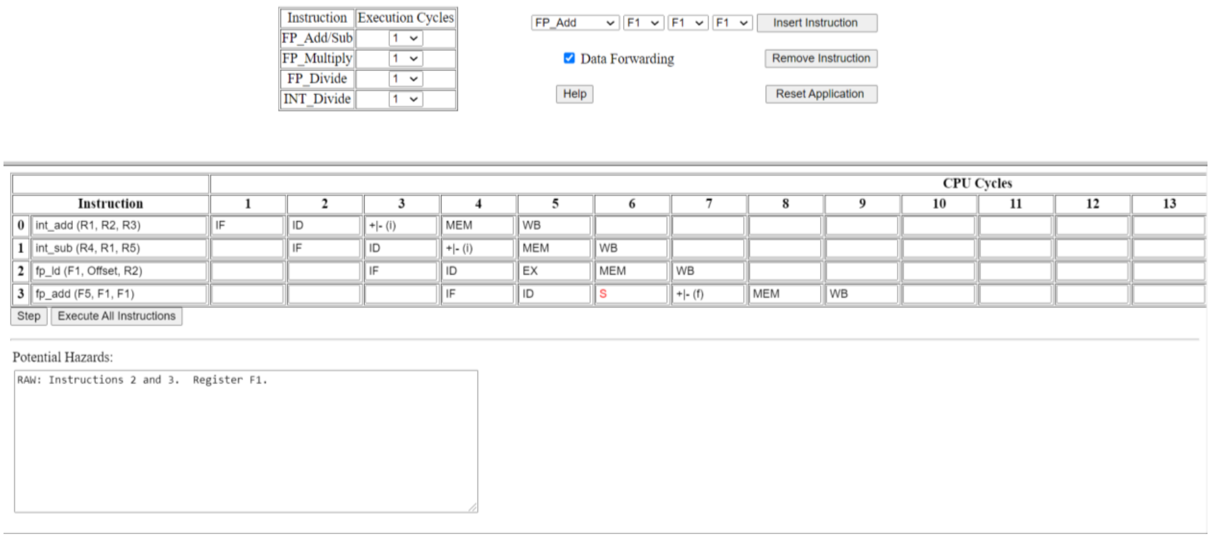
Observe the following and note down the results. Check whether there is data dependency among the instructions?

Yes, RAW hazard in instructions 0 and 1 for R1 as there is no data forwarding the value of R1 can be obtained only after WB stage. Another RAW hazard in instructions 2 and 3 for F1 and this cannot be eliminated even with data forwarding (but no. of stalls will reduce).

If yes, then, how many stall states have been introduced?

2 stalls in instruction 1 and 2 and 2 more stalls in instruction 3. So, totally **4** stalls.

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



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

		CPU Cycles												
	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13
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, R2)			IF	ID	EX	MEM	WB						
3	fp_add (F5, F1, F1)				IF	ID	S	+|- (f)	MEM	WB				

Potential Hazards:

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

The above screenshot is for the same with data forwarding. As we can see the number of stalls is reduced from 4 to 2.

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

With Data Forwarding **9** Cycles

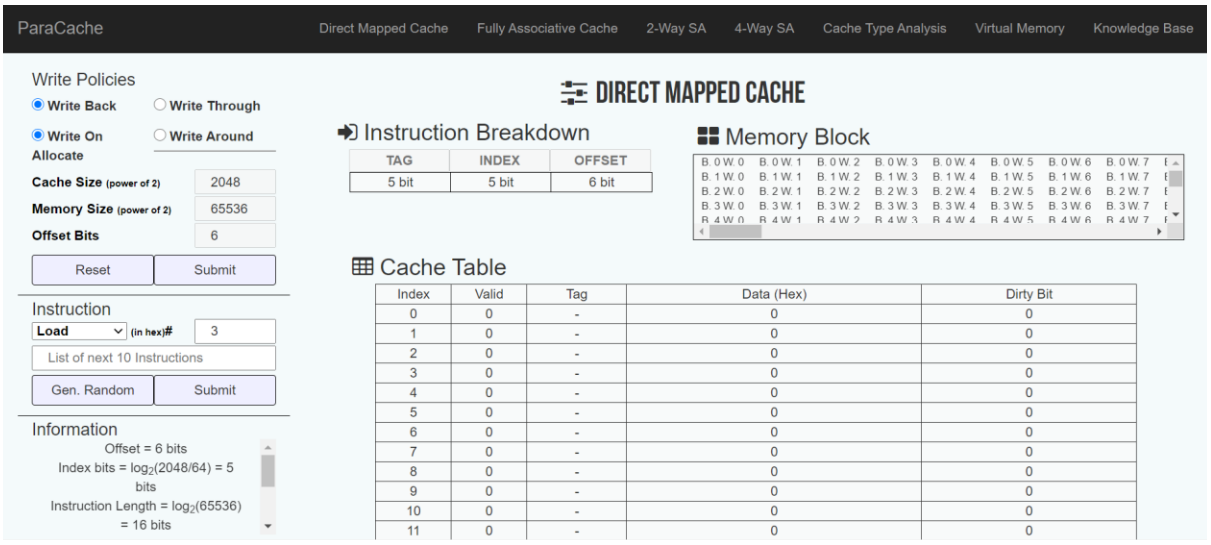
Without Data Forwarding **12** cycles.

Task2: Cache Simulator:

Link: [ParaCache Main Menu (ntu.edu.sg)]

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.

Screenshot of Paracache Simulator:



DIRECT MAPPED CACHE

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

Instruction

Load (in hex)# 3

List of next 10 Instructions

Information

Offset = 6 bits
Index bits = $\log_2(2048/64) = 5$ bits
Instruction Length = $\log_2(65536)$ = 16 bits

Instruction Breakdown

TAG	INDEX	OFFSET
5 bit	5 bit	6 bit

Memory Block

B. 0 W. 0	B. 0 W. 1	B. 0 W. 2	B. 0 W. 3	B. 0 W. 4	B. 0 W. 5	B. 0 W. 6	B. 0 W. 7
B. 1 W. 0	B. 1 W. 1	B. 1 W. 2	B. 1 W. 3	B. 1 W. 4	B. 1 W. 5	B. 1 W. 6	B. 1 W. 7
B. 2 W. 0	B. 2 W. 1	B. 2 W. 2	B. 2 W. 3	B. 2 W. 4	B. 2 W. 5	B. 2 W. 6	B. 2 W. 7
B. 3 W. 0	B. 3 W. 1	B. 3 W. 2	B. 3 W. 3	B. 3 W. 4	B. 3 W. 5	B. 3 W. 6	B. 3 W. 7
B. 4 W. 0	B. 4 W. 1	B. 4 W. 2	B. 4 W. 3	B. 4 W. 4	B. 4 W. 5	B. 4 W. 6	B. 4 W. 7

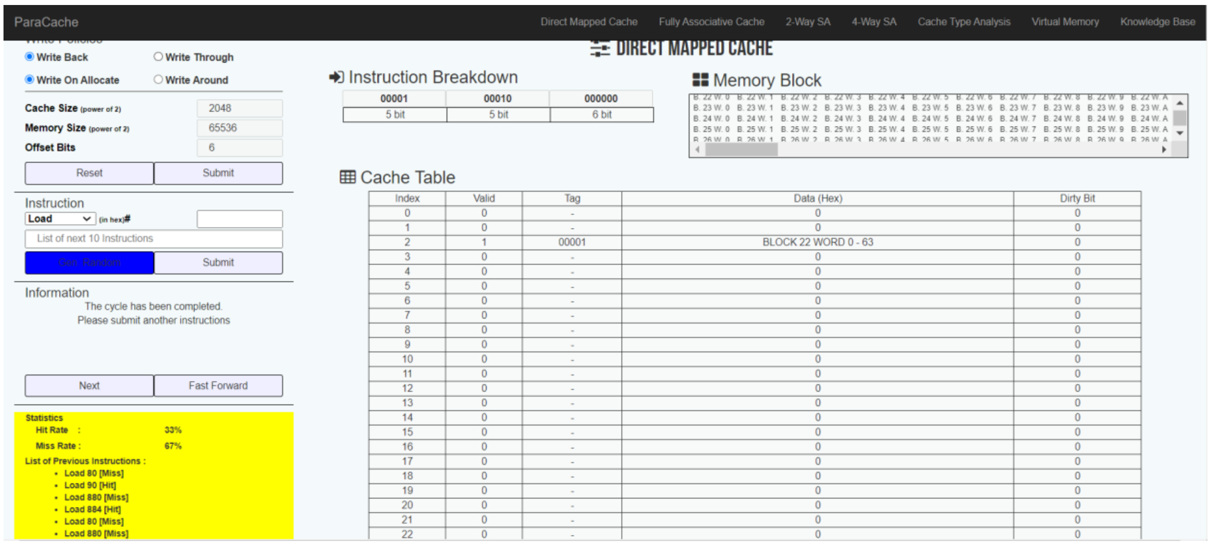
Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	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

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

Offset/Word = 64 = 2^6 = 6 bits
Index bits/Block = $\log_2(2048/64)$ = 5 bits
Instruction Length = $\log_2(65536)$ = 16 bits
Tag = 16 bits - 6 bits - 5 bits = 5 bits

(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.



[✓ Write Back] Write Through

[✓ Write On Allocate] Write Around

Cache Size (power of 2)	2048
Memory Size (power of 2)	65536
Offset Bits	6

Instruction

Load (in hex)#

List of next 10 Instructions

Information

The cycle has been completed.
Please submit another instructions

Statistics

Hit Rate : 33%

Miss Rate : 67%

List of Previous Instructions :

- Load 80 [Miss]
- Load 90 [Hit]
- Load 880 [Miss]
- Load 884 [Hit]
- Load 80 [Miss]
- Load 880 [Miss]

Instruction Breakdown

00001	00010	000000
5 bit	5 bit	6 bit

Memory Block

B. 22 W. 0	B. 22 W. 1	B. 22 W. 2	B. 22 W. 3	B. 22 W. 4	B. 22 W. 5	B. 22 W. 6	B. 22 W. 7	B. 22 W. 8	B. 22 W. 9	B. 22 W. A
B. 23 W. 0	B. 23 W. 1	B. 23 W. 2	B. 23 W. 3	B. 23 W. 4	B. 23 W. 5	B. 23 W. 6	B. 23 W. 7	B. 23 W. 8	B. 23 W. 9	B. 23 W. A
B. 24 W. 0	B. 24 W. 1	B. 24 W. 2	B. 24 W. 3	B. 24 W. 4	B. 24 W. 5	B. 24 W. 6	B. 24 W. 7	B. 24 W. 8	B. 24 W. 9	B. 24 W. A
B. 25 W. 0	B. 25 W. 1	B. 25 W. 2	B. 25 W. 3	B. 25 W. 4	B. 25 W. 5	B. 25 W. 6	B. 25 W. 7	B. 25 W. 8	B. 25 W. 9	B. 25 W. A
B. 26 W. 0	B. 26 W. 1	B. 26 W. 2	B. 26 W. 3	B. 26 W. 4	B. 26 W. 5	B. 26 W. 6	B. 26 W. 7	B. 26 W. 8	B. 26 W. 9	B. 26 W. A

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	1	00001	BLOCK 22 WORD 0 - 63	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	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
19	0	-	0	0
20	0	-	0	0
21	0	-	0	0
22	0	-	0	0

128=0x80

144=0x90

2176=0x880

2180=0x884

128=0x80

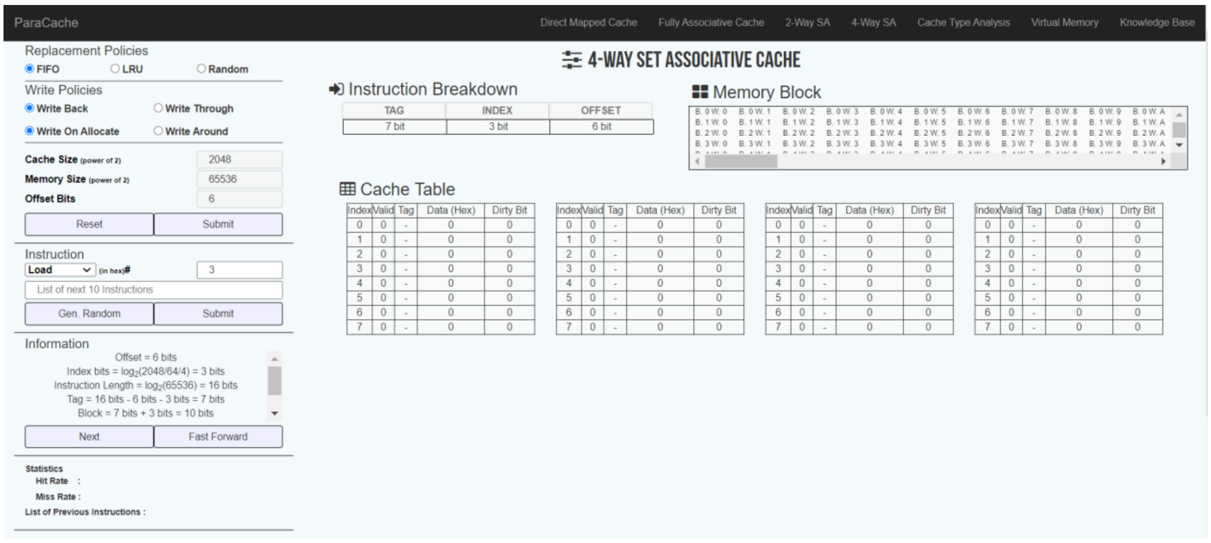
2176=0x880

Decimal	Hexadecimal Equivalent	Hit/Miss
128	80	Miss
144	90	Hit
2176	880	Miss
2180	884	Hit
128	80	Miss
2176	880	Miss

2. **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.**

4-way Set Associative:

(a) Calculate the number of bits in each of the Tag, Set, and Word fields of the memory address.


ParaCache — Direct Mapped Cache · Fully Associative Cache · 2-Way SA · 4-Way SA · Cache Type Analysis · Virtual Memory · Knowledge Base

Replacement Policies: [✓ FIFO] LRU Random

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

Instruction: Load (in hex)# 3

List of next 10 Instructions

Information

Offset = 6 bits
Index bits = $\log_2(2048/64/4)$ = 3 bits
Instruction Length = $\log_2(65536)$ = 16 bits
Tag = 16 bits - 6 bits - 3 bits = 7 bits
Block = 7 bits + 3 bits = 10 bits

Statistics
Hit Rate :
Miss Rate :
List of Previous Instructions :

4-WAY SET ASSOCIATIVE CACHE

Instruction Breakdown

TAG	INDEX	OFFSET
7 bit	3 bit	6 bit

Memory Block

B. 0 W. 0	B. 0 W. 1	B. 0 W. 2	B. 0 W. 3	B. 0 W. 4	B. 0 W. 5	B. 0 W. 6	B. 0 W. 7	B. 0 W. 8	B. 0 W. 9	B. 0 W. A
B. 1 W. 0	B. 1 W. 1	B. 1 W. 2	B. 1 W. 3	B. 1 W. 4	B. 1 W. 5	B. 1 W. 6	B. 1 W. 7	B. 1 W. 8	B. 1 W. 9	B. 1 W. A
B. 2 W. 0	B. 2 W. 1	B. 2 W. 2	B. 2 W. 3	B. 2 W. 4	B. 2 W. 5	B. 2 W. 6	B. 2 W. 7	B. 2 W. 8	B. 2 W. 9	B. 2 W. A
B. 3 W. 0	B. 3 W. 1	B. 3 W. 2	B. 3 W. 3	B. 3 W. 4	B. 3 W. 5	B. 3 W. 6	B. 3 W. 7	B. 3 W. 8	B. 3 W. 9	B. 3 W. A

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

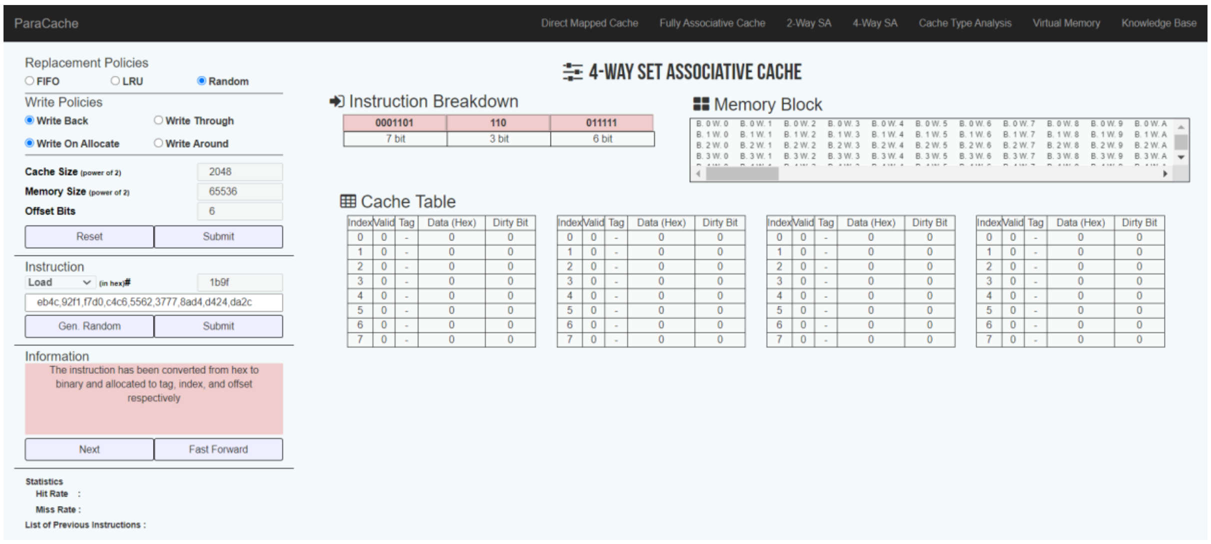
Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Offset/Word = $\log_2(64) = \log_2(2^6) = 6$ bits
Index bits/Set = $\log_2(2048/64/4) = 3$ bits
Instruction Length = $\log_2(65536) = 16$ bits
Tag = 16 bits - 6 bits - 3 bits = 7 bits

Randomly generated instructions:

1b9f,eb4c,92f1,f7d0,c4c6,5562,3777,8ad4,d424,da2c


ParaCache — Direct Mapped Cache · Fully Associative Cache · 2-Way SA · 4-Way SA · Cache Type Analysis · Virtual Memory · Knowledge Base

Replacement Policies: FIFO LRU [✓ Random]

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

Instruction: Load (in hex)# 1b9f

eb4c,92f1,f7d0,c4c6,5562,3777,8ad4,d424,da2c

Information

The instruction has been converted from hex to binary and allocated to tag, index, and offset respectively

Statistics
Hit Rate :
Miss Rate :
List of Previous Instructions :

4-WAY SET ASSOCIATIVE CACHE

Instruction Breakdown

0001101	110	011111
7 bit	3 bit	6 bit

Memory Block

B. 0 W. 0	B. 0 W. 1	B. 0 W. 2	B. 0 W. 3	B. 0 W. 4	B. 0 W. 5	B. 0 W. 6	B. 0 W. 7	B. 0 W. 8	B. 0 W. 9	B. 0 W. A
B. 1 W. 0	B. 1 W. 1	B. 1 W. 2	B. 1 W. 3	B. 1 W. 4	B. 1 W. 5	B. 1 W. 6	B. 1 W. 7	B. 1 W. 8	B. 1 W. 9	B. 1 W. A
B. 2 W. 0	B. 2 W. 1	B. 2 W. 2	B. 2 W. 3	B. 2 W. 4	B. 2 W. 5	B. 2 W. 6	B. 2 W. 7	B. 2 W. 8	B. 2 W. 9	B. 2 W. A
B. 3 W. 0	B. 3 W. 1	B. 3 W. 2	B. 3 W. 3	B. 3 W. 4	B. 3 W. 5	B. 3 W. 6	B. 3 W. 7	B. 3 W. 8	B. 3 W. 9	B. 3 W. A

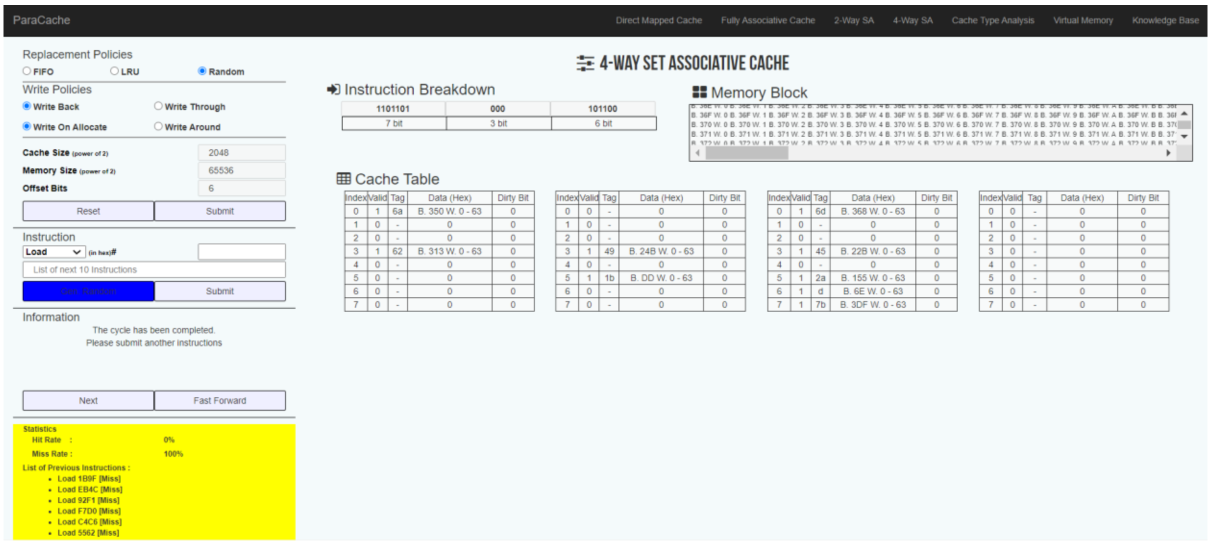
Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0



Replacement Policies
FIFO   LRU   [✓ Random]

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

Information
The cycle has been completed.
Please submit another instructions

Statistics
Hit Rate : 0%
Miss Rate : 100%
List of Previous Instructions :
- Load 1B9F [Miss]
- Load EB4C [Miss]
- Load 92F1 [Miss]
- Load F7D0 [Miss]
- Load C4C6 [Miss]
- Load 5562 [Miss]

4-WAY SET ASSOCIATIVE CACHE

Instruction Breakdown

1101101	000	101100
7 bit	3 bit	6 bit

Memory Block

```
B. 36F W. 0 B. 36F W. 1 B. 36F W. 2 B. 36F W. 3 B. 36F W. 4 B. 36F W. 5 B. 36F W. 6 B. 36F W. 7 B. 36F W. 8 B. 36F W. 9 B. 36F W. A B. 36F W. B B. 36
B. 370 W. 0 B. 370 W. 1 B. 370 W. 2 B. 370 W. 3 B. 370 W. 4 B. 370 W. 5 B. 370 W. 6 B. 370 W. 7 B. 370 W. 8 B. 370 W. 9 B. 370 W. A B. 370 W. B B. 37
B. 371 W. 0 B. 371 W. 1 B. 371 W. 2 B. 371 W. 3 B. 371 W. 4 B. 371 W. 5 B. 371 W. 6 B. 371 W. 7 B. 371 W. 8 B. 371 W. 9 B. 371 W. A B. 371 W. B B. 37
```

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	1	6a	B. 350 W. 0 - 63	0
1	0	-	0	0
2	0	-	0	0
3	1	62	B. 313 W. 0 - 63	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

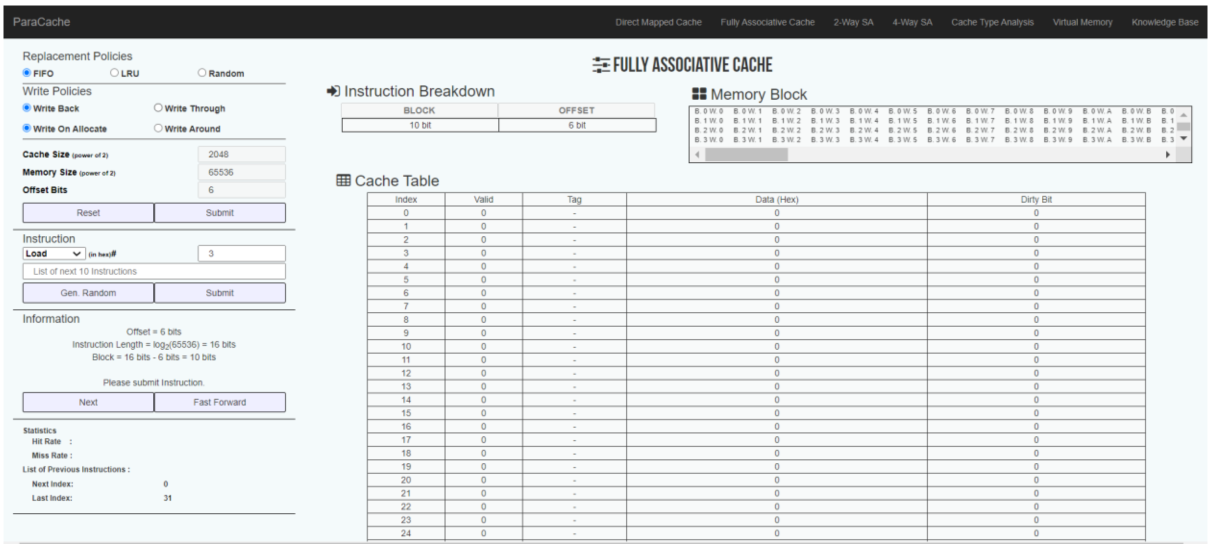
Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	1	49	B. 24B W. 0 - 63	0
4	0	-	0	0
5	1	1b	B. DD W. 0 - 63	0
6	0	-	0	0
7	0	-	0	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	1	6d	B. 368 W. 0 - 63	0
1	0	-	0	0
2	0	-	0	0
3	1	45	B. 22B W. 0 - 63	0
4	0	-	0	0
5	1	2a	B. 155 W. 0 - 63	0
6	1	d	B. 6E W. 0 - 63	0
7	1	7b	B. 3DF W. 0 - 63	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Instruction	**Hit/Miss**
1b9f	Miss
Eb4c	Miss
92f1	Miss
F7d0	Miss
C4c6	Miss
5562	Miss
3777	Miss
8ad4	Miss
D424	Miss
Da2c	Miss

Fully Associative:



Replacement Policies
[✓ FIFO]   LRU   Random

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

3

Information
Offset = 6 bits
Instruction Length = $\log_2(65536)$ = 16 bits
Block = 16 bits - 6 bits = 10 bits

Please submit Instruction.

Statistics
Hit Rate :
Miss Rate :
List of Previous Instructions :
Next Index: 0
Last Index: 31

FULLY ASSOCIATIVE CACHE

Instruction Breakdown

BLOCK	OFFSET
10 bit	6 bit

Memory Block

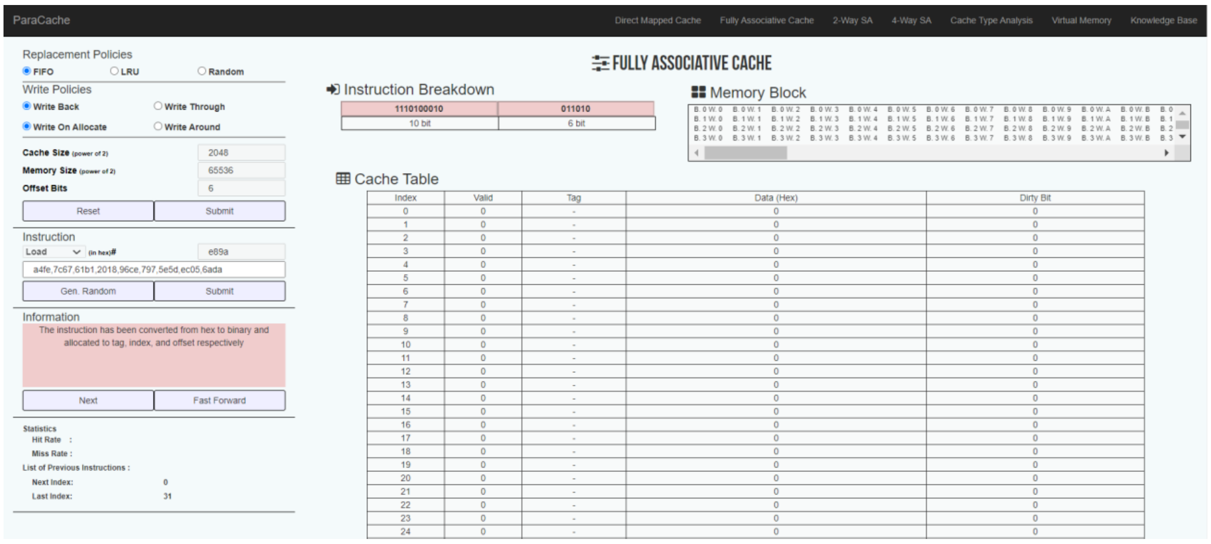
```
B. 0 W. 0  B. 0 W. 1  B. 0 W. 2  B. 0 W. 3  B. 0 W. 4  B. 0 W. 5  B. 0 W. 6  B. 0 W. 7  B. 0 W. 8  B. 0 W. 9  B. 0 W. A  B. 0 W. B  B. 0
B. 1 W. 0  B. 1 W. 1  B. 1 W. 2  B. 1 W. 3  B. 1 W. 4  B. 1 W. 5  B. 1 W. 6  B. 1 W. 7  B. 1 W. 8  B. 1 W. 9  B. 1 W. A  B. 1 W. B  B. 1
B. 2 W. 0  B. 2 W. 1  B. 2 W. 2  B. 2 W. 3  B. 2 W. 4  B. 2 W. 5  B. 2 W. 6  B. 2 W. 7  B. 2 W. 8  B. 2 W. 9  B. 2 W. A  B. 2 W. B  B. 2
B. 3 W. 0  B. 3 W. 1  B. 3 W. 2  B. 3 W. 3  B. 3 W. 4  B. 3 W. 5  B. 3 W. 6  B. 3 W. 7  B. 3 W. 8  B. 3 W. 9  B. 3 W. A  B. 3 W. B  B. 3
```

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	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
19	0	-	0	0
20	0	-	0	0
21	0	-	0	0
22	0	-	0	0
23	0	-	0	0
24	0	-	0	0

Offset/Word = 6 bits
Instruction Length = $\log_2(65536)$ = 16 bits
Block = 16 bits - 6 bits = 10 bits

Randomly generated instructions: e89a, a4fe,7c67,61b1,2018,96ce,797,5e5d,ec05,6ada



Replacement Policies
[✓ FIFO] LRU Random

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

Instruction
Load (in hex)# e89a
a4fe,7c67,61b1,2018,96ce,797,5e5d,ec05,6ada

Information
The instruction has been converted from hex to binary and allocated to tag, index, and offset respectively

Statistics
Hit Rate :
Miss Rate :
List of Previous Instructions :
Next Index: 0
Last Index: 31

FULLY ASSOCIATIVE CACHE

Instruction Breakdown

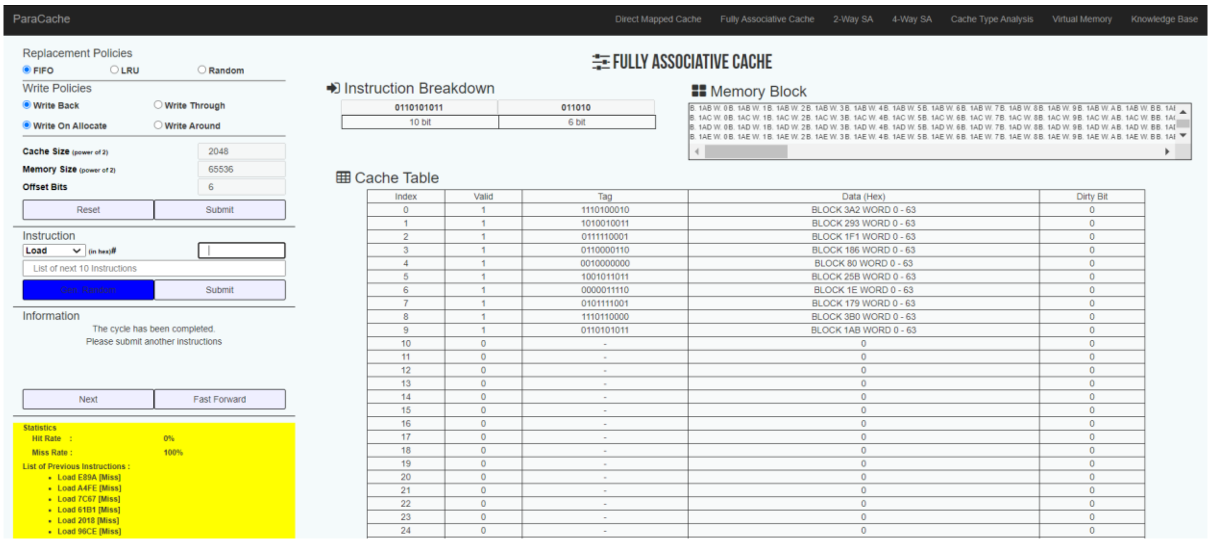
1110100010	011010
10 bit	6 bit

Memory Block

```
B.0W.0 B.0W.1 B.0W.2 B.0W.3 B.0W.4 B.0W.5 B.0W.6 B.0W.7 B.0W.8 B.0W.9 B.0W.A B.0W.B B.0
B.1W.0 B.1W.1 B.1W.2 B.1W.3 B.1W.4 B.1W.5 B.1W.6 B.1W.7 B.1W.8 B.1W.9 B.1W.A B.1W.B B.1
B.2W.0 B.2W.1 B.2W.2 B.2W.3 B.2W.4 B.2W.5 B.2W.6 B.2W.7 B.2W.8 B.2W.9 B.2W.A B.2W.B B.2
B.3W.0 B.3W.1 B.3W.2 B.3W.3 B.3W.4 B.3W.5 B.3W.6 B.3W.7 B.3W.8 B.3W.9 B.3W.A B.3W.B B.3
```

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	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
19	0	-	0	0
20	0	-	0	0
21	0	-	0	0
22	0	-	0	0
23	0	-	0	0
24	0	-	0	0



Replacement Policies
[✓ FIFO] LRU Random

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

Instruction
Load (in hex)#
List of next 10 Instructions

Information
The cycle has been completed.
Please submit another instructions

Statistics
Hit Rate : 0%
Miss Rate : 100%
List of Previous Instructions :
- Load E89A [Miss]
- Load A4FE [Miss]
- Load 7C67 [Miss]
- Load 61B1 [Miss]
- Load 2018 [Miss]
- Load 96CE [Miss]

FULLY ASSOCIATIVE CACHE

Instruction Breakdown

0110101011	011010
10 bit	6 bit

Memory Block

```
B. 1AB W. 0B. 1AB W. 1B. 1AB W. 2B. 1AB W. 3B. 1AB W. 4B. 1AB W. 5B. 1AB W. 6B. 1AB W. 7B. 1AB W. 8B. 1AB W. 9B. 1AB W. AB. 1AB W. BB. 1AI
B. 1AC W. 0B. 1AC W. 1B. 1AC W. 2B. 1AC W. 3B. 1AC W. 4B. 1AC W. 5B. 1AC W. 6B. 1AC W. 7B. 1AC W. 8B. 1AC W. 9B. 1AC W. AB. 1AC W. BB. 1A(
B. 1AD W. 0B. 1AD W. 1B. 1AD W. 2B. 1AD W. 3B. 1AD W. 4B. 1AD W. 5B. 1AD W. 6B. 1AD W. 7B. 1AD W. 8B. 1AD W. 9B. 1AD W. AB. 1AD W. BB. 1AI
B. 1AE W. 0B. 1AE W. 1B. 1AE W. 2B. 1AE W. 3B. 1AE W. 4B. 1AE W. 5B. 1AE W. 6B. 1AE W. 7B. 1AE W. 8B. 1AE W. 9B. 1AE W. AB. 1AE W. BB. 1AI
```

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	1	1110100010	BLOCK 3A2 WORD 0 - 63	0
1	1	1010010011	BLOCK 293 WORD 0 - 63	0
2	1	0111110001	BLOCK 1F1 WORD 0 - 63	0
3	1	0110000110	BLOCK 186 WORD 0 - 63	0
4	1	0010000000	BLOCK 80 WORD 0 - 63	0
5	1	1001011011	BLOCK 25B WORD 0 - 63	0
6	1	0000011110	BLOCK 1E WORD 0 - 63	0
7	1	0101111001	BLOCK 179 WORD 0 - 63	0
8	1	1110110000	BLOCK 3B0 WORD 0 - 63	0
9	1	0110101011	BLOCK 1AB WORD 0 - 63	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
19	0	-	0	0
20	0	-	0	0
21	0	-	0	0
22	0	-	0	0
23	0	-	0	0
24	0	-	0	0

Instruction	Hit/Miss
E89a	Miss
A4fe	Miss
7c67	Miss
61b1	Miss
2018	Miss
96ce	Miss
797	Miss
5e5d	Miss
Ec05	Miss
6ada	Miss

e89a, a4fe,7c67,61b1,2018,96ce,797,5e5d,ec05,6ada

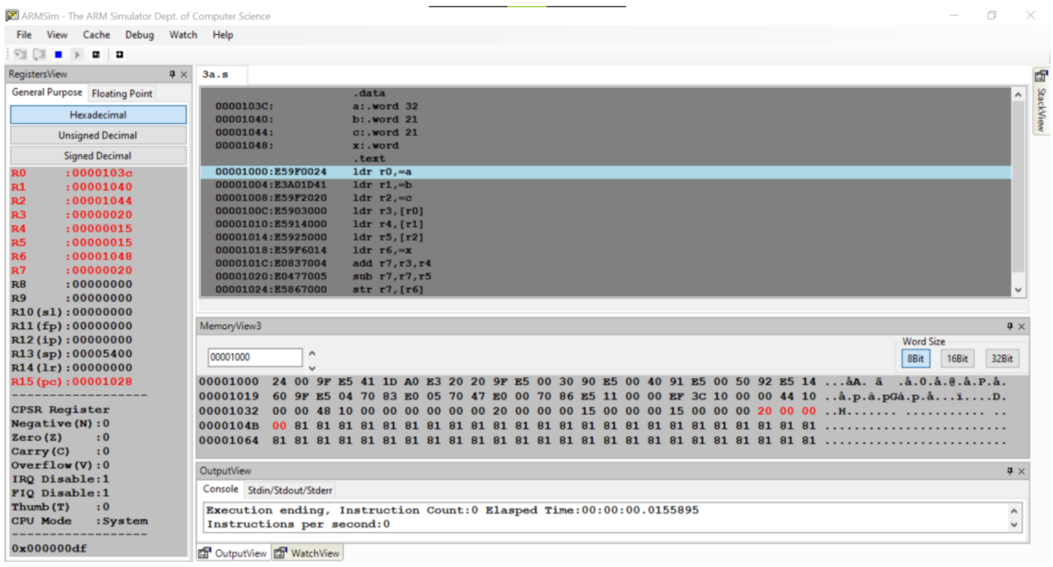
Task3: Use Arm Simulator Given a ‘c’ code convert it in its equivalent Arm Code and execute in ARM simulator

1) x = (a + b) – c

Code:

```
.data

a:.word 32

b:.word 21

c:.word 21

x:.word

.text

ldr r0,=a

ldr r1,=b

ldr r2,=c

ldr r3,[r0]

ldr r4,[r1]

ldr r5,[r2]

ldr r6,=x

add r7,r3,r4

sub r7,r7,r5

str r7,[r6]

swi 0x11
```

Screenshot:



```
R0       :0000103c
R1       :00001040
R2       :00001044
R3       :00000020
R4       :00000015
R5       :00000015
R6       :00001048
R7       :00000020
R8       :00000000
R9       :00000000
R10(sl)  :00000000
R11(fp)  :00000000
R12(ip)  :00000000
R13(sp)  :00005400
R14(lr)  :00000000
R15(pc)  :00001028
------------------
CPSR Register
Negative(N):0
Zero(Z)    :0
Carry(C)   :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :System
------------------
0x000000df
```

```
                       .data
0000103C:              a:.word 32
00001040:              b:.word 21
00001044:              c:.word 21
00001048:              x:.word
                       .text
00001000:E59F0024      ldr r0,=a
00001004:E3A01D41      ldr r1,=b
00001008:E59F2020      ldr r2,=c
0000100C:E5903000      ldr r3,[r0]
00001010:E5914000      ldr r4,[r1]
00001014:E5925000      ldr r5,[r2]
00001018:E59F6014      ldr r6,=x
0000101C:E0837004      add r7,r3,r4
00001020:E0477005      sub r7,r7,r5
00001024:E5867000      str r7,[r6]
```

```
00001000  24 00 9F E5 41 1D A0 E3 20 20 9F E5 00 30 90 E5 00 40 91 E5 00 50 92 E5 14 ...åA. ã  .å.0.å.@.å.P.å.
00001019  60 9F E5 04 70 83 E0 05 70 47 E0 00 70 86 E5 11 00 00 EF 3C 10 00 00 44 10 ..å.p.à.pGà.p.å...ï....D.
00001032  00 00 48 10 00 00 00 00 00 00 20 00 00 00 15 00 00 00 15 00 00 00 20 00 00 ..H....... .......... ..
0000104B  00 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 .........................
00001064  81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 .........................
```

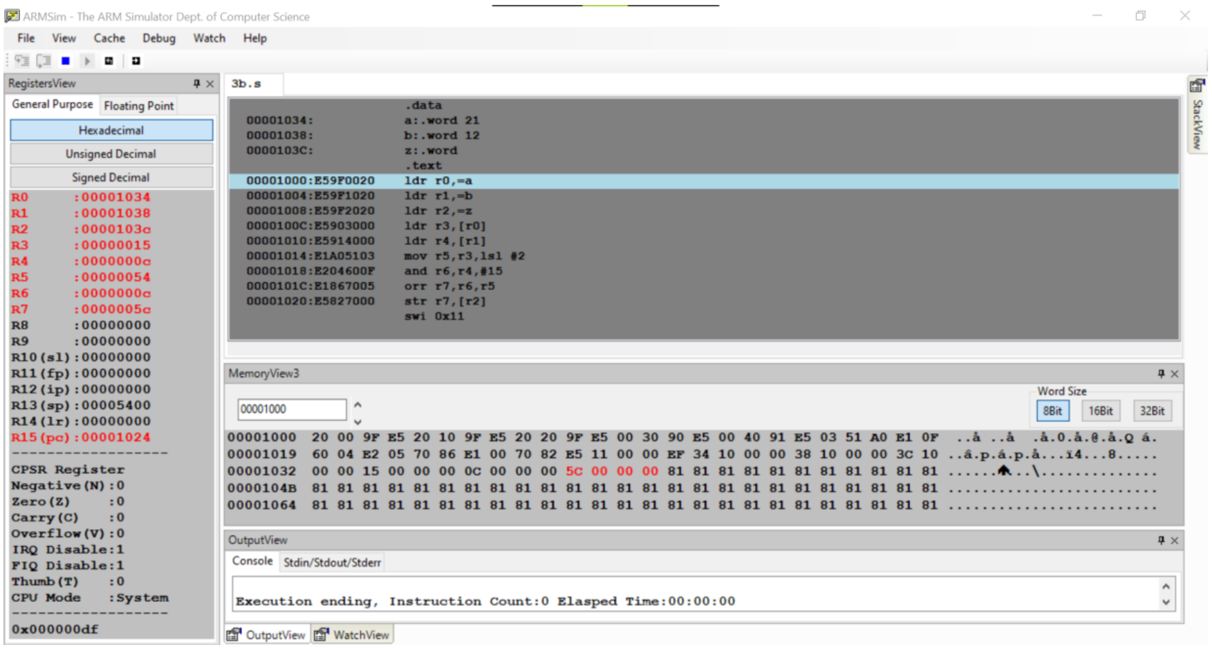
```
Execution ending, Instruction Count:0 Elasped Time:00:00:00.0155895
Instructions per second:0
```

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

Code:

```
.data
a:.word 21
b:.word 12
z:.word
.text
ldr r0,=a
ldr r1,=b
ldr r2,=z
ldr r3,[r0]
ldr r4,[r1]
mov r5,r3,lsl #2
and r6,r4,#15
orr r7,r6,r5
str r7,[r2]
swi 0x11
```

Screenshot:



```
R0       :00001034
R1       :00001038
R2       :0000103c
R3       :00000015
R4       :0000000c
R5       :00000054
R6       :0000000c
R7       :0000005c
R8       :00000000
R9       :00000000
R10(sl):00000000
R11(fp):00000000
R12(ip):00000000
R13(sp):00005400
R14(lr):00000000
R15(pc):00001024
-------------------
CPSR Register
Negative(N):0
Zero(Z)     :0
Carry(C)    :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :System
-------------------
0x000000df
```

```
                        .data
00001034:               a:.word 21
00001038:               b:.word 12
0000103C:               z:.word
                        .text
00001000:E59F0020       ldr r0,=a
00001004:E59F1020       ldr r1,=b
00001008:E59F2020       ldr r2,=z
0000100C:E5903000       ldr r3,[r0]
00001010:E5914000       ldr r4,[r1]
00001014:E1A05103       mov r5,r3,lsl #2
00001018:E204600F       and r6,r4,#15
0000101C:E1867005       orr r7,r6,r5
00001020:E5827000       str r7,[r2]
                        swi 0x11
```

```
00001000
00001000  20 00 9F E5 20 10 9F E5 20 20 9F E5 00 30 90 E5 00 40 91 E5 03 51 A0 E1 0F  ..å ..å  .å.0.å.@.å.Q á.
00001019  60 04 E2 05 70 86 E1 00 70 82 E5 11 00 00 EF 34 10 00 00 38 10 00 00 3C 10  ..â.p.á.p.å...ï4...8.....
00001032  00 00 15 00 00 00 0C 00 00 00 5C 00 00 00 81 81 81 81 81 81 81 81 81 81 81  ......  ..\.............
0000104B  81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81  .........................
00001064  81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81 81  .........................
```

```
Execution ending, Instruction Count:0 Elasped Time:00:00:00
```