

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

Remove Instruction

Reset Application

☐ Data Forwarding

Help

		CPU Cycles																	
Instruction		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_id (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

Remove Instruction

Reset Application

☒ Data Forwarding

Help

		CPU Cycles																	
Instruction		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_id (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.

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)#

3

List of next 10 Instructions

Gen. Random

Submit

Information

Offset = 6 bits

Index bits = $\log_2(2048/64) = 5$ bits

Instruction Length = $\log_2(65536) = 16$ bits

Tag = 16 bits - 6 bits - 5 bits = 5 bits

Block = 5 bits + 5 bits = 10 bits

Next

Fast Forward

Based on the info given:

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

128 - Miss

Direct Mapped Cache Simulator

www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/dmc.html

Apps Gmail YouTube Maps cpp Delivery Success S... The Pirate Bay

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)# 128

144, 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	00000	000000
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 0 W 8	B 0 W 9
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 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 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
R 4 W 0	R 4 W 1	R 4 W 2	R 4 W 3	R 4 W 4	R 4 W 5	R 4 W 6	R 4 W 7	R 4 W 8	R 4 W 9

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	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

Type here to search

100% 20:10 09-04-2021

144 - Miss

Direct Mapped Cache Simulator

www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/dmc.html

Apps Gmail YouTube Maps cpp Delivery Success S... The Pirate Bay

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)# 144

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	00100	101000
5 bit	5 bit	6 bit

Memory Block

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

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	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

Type here to search

100% 20:10 09-04-2021

2176 - Miss

Direct Mapped Cache Simulator

www3.ntu.edu.sg/home/smitha/ParaCache/Paracache/dmc.html

Apps

Gmail

YouTube

Maps

cpp

Delivery Success S...

The Pirate Bay

Reading list

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

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

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

Apps

Gmail

YouTube

Maps

cpp

Delivery Success S...

The Pirate Bay

Reading list

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

B. 85 W. 0	B. 85 W. 1	B. 85 W. 2	B. 85 W. 3	B. 85 W. 4	B. 85 W. 5	B. 85 W. 6	B. 85 W. 7	B. 85 W. 8	B. 85 W. 9
B. 86 W. 0	B. 86 W. 1	B. 86 W. 2	B. 86 W. 3	B. 86 W. 4	B. 86 W. 5	B. 86 W. 6	B. 86 W. 7	B. 86 W. 8	B. 86 W. 9
B. 87 W. 0	B. 87 W. 1	B. 87 W. 2	B. 87 W. 3	B. 87 W. 4	B. 87 W. 5	B. 87 W. 6	B. 87 W. 7	B. 87 W. 8	B. 87 W. 9
B. 88 W. 0	B. 88 W. 1	B. 88 W. 2	B. 88 W. 3	B. 88 W. 4	B. 88 W. 5	B. 88 W. 6	B. 88 W. 7	B. 88 W. 8	B. 88 W. 9
R. 89 W. 0	R. 89 W. 1	R. 89 W. 2	R. 89 W. 3	R. 89 W. 4	R. 89 W. 5	R. 89 W. 6	R. 89 W. 7	R. 89 W. 8	R. 89 W. 9

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

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)#

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	00110	000000
5 bit	5 bit	6 bit

Memory Block

B. 86 W. 0	B. 86 W. 1	B. 86 W. 2	B. 86 W. 3	B. 86 W. 4	B. 86 W. 5	B. 86 W. 6	B. 86 W. 7	B. 86 W. 8	B. 86 W. 9
B. 87 W. 0	B. 87 W. 1	B. 87 W. 2	B. 87 W. 3	B. 87 W. 4	B. 87 W. 5	B. 87 W. 6	B. 87 W. 7	B. 87 W. 8	B. 87 W. 9
B. 88 W. 0	B. 88 W. 1	B. 88 W. 2	B. 88 W. 3	B. 88 W. 4	B. 88 W. 5	B. 88 W. 6	B. 88 W. 7	B. 88 W. 8	B. 88 W. 9
B. 89 W. 0	B. 89 W. 1	B. 89 W. 2	B. 89 W. 3	B. 89 W. 4	B. 89 W. 5	B. 89 W. 6	B. 89 W. 7	B. 89 W. 8	B. 89 W. 9
R. 8A W. 0	R. 8A W. 1	R. 8A W. 2	R. 8A W. 3	R. 8A W. 4	R. 8A W. 5	R. 8A W. 6	R. 8A W. 7	R. 8A W. 8	R. 8A W. 9

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	1	00100	BLOCK 86 WORD 0 - 63	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

2176 – Hit

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

List of next 10 Instructions

Gen. Random

Submit

Information

The cycle has been completed.
Please submit another instructions

Next

Fast Forward

Statistics

Hit Rate : 17%
Miss Rate : 83%

List of Previous Instructions :

Load 0 [Miss]

DIRECT MAPPED CACHE

Instruction Breakdown

00000	00100	101000
5 bit	5 bit	6 bit

Memory Block

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

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	1	00100	BLOCK 86 WORD 0 - 63	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

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.

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

5

Reset

Submit

Instruction

Load ▾ (in hex)#

3

List of next 10 Instructions

Gen. Random

Submit

Information

Offset = 5 bits

Index bits = $\log_2(2048/32/4) = 4$ bits

Instruction Length = $\log_2(65536) = 16$ bits

Tag = 16 bits - 5 bits - 4 bits = 7 bits

Block = 7 bits + 4 bits = 11 bits

Next

Fast Forward

Observations based on given info:

- Index bits: 4
- Tag bits: 7
- Block bits: 11

Generated a random set of numbers:

4-WAY SET ASSOCIATIVE CACHE

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: 5

Instruction
 Load (in hex)# 62bd
 524,96e5,63b9,a3bf,f7bc,9f1b,240b,2e49

Instruction Breakdown

1111010	0010	01110
7 bit	4 bit	5 bit

Memory Block

B	7A2 W	0 B	7A2 W	1 B	7A2 W	2 B	7A2 W	3 B	7A2 W	4 B	7A2 W	5 B	7A2 W	6 B	7A2 W	7 B	7A2 W	8 B	7A2 W	9 B
7A3 W	0 B	7A3 W	1 B	7A3 W	2 B	7A3 W	3 B	7A3 W	4 B	7A3 W	5 B	7A3 W	6 B	7A3 W	7 B	7A3 W	8 B	7A3 W	9 B	
7A4 W	0 B	7A4 W	1 B	7A4 W	2 B	7A4 W	3 B	7A4 W	4 B	7A4 W	5 B	7A4 W	6 B	7A4 W	7 B	7A4 W	8 B	7A4 W	9 B	
7A5 W	0 B	7A5 W	1 B	7A5 W	2 B	7A5 W	3 B	7A5 W	4 B	7A5 W	5 B	7A5 W	6 B	7A5 W	7 B	7A5 W	8 B	7A5 W	9 B	
7A6 W	0 R	7A6 W	1 R	7A6 W	2 R	7A6 W	3 R	7A6 W	4 R	7A6 W	5 R	7A6 W	6 R	7A6 W	7 R	7A6 W	8 R	7A6 W	9 R	

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

Information
 The cycle has been completed.
 Please submit another instructions

Statistics
 Hit Rate : 0%

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

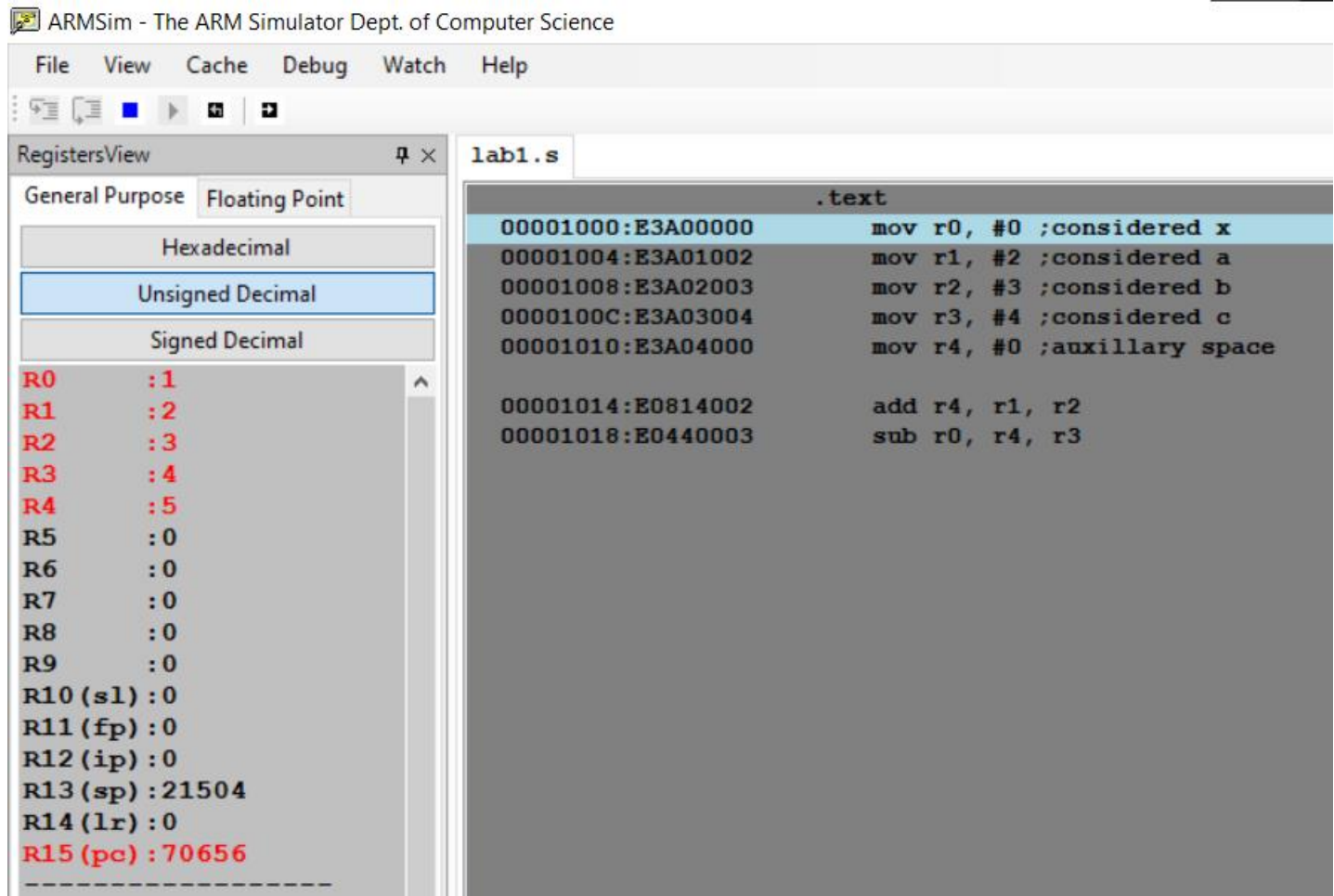
mov r3, #4 ;considered c

mov r4, #0 ;auxillary space

add r4,r1,r2

sub r0,r4,r3

Output screenshot:



2) $z = (a \ll 2) | (b \& 15)$:

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:

ARMSim - The ARM Simulator Dept. of Computer Science

The screenshot displays the ARMSim ARM simulator interface. The top menu bar includes File, View, Cache, Debug, Watch, and Help. Below the menu is a toolbar with icons for file operations and execution. The main window is divided into two panes. The left pane, titled 'RegistersView', shows a list of registers (R0 to R15) with their current values. The right pane, titled 'lab1.s', shows the assembly code for a program.

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

Register	Value
R0	:11
R1	:2
R2	:3
R3	:3
R4	:8
R5	:0
R6	:0
R7	:0
R8	:0
R9	:0
R10 (s1)	:0
R11 (fp)	:0
R12 (ip)	:0
R13 (sp)	:21504
R14 (lr)	:0
R15 (pc)	:70656

lab1.s

.text

```
00001000:E3A00000    mov r0, #0 ;considered z
00001004:E3A01002    mov r1, #2 ;considered a
00001008:E3A02003    mov r2, #3 ;considered b

0000100C:E3A03004    mov r3, #4 ;auxillary space 1
00001010:E3A04000    mov r4, #0 ;auxillary space 2

00001014:E202300F    and r3, r2, #15
00001018:E1A04101    mov r4, r1, LSL #2
0000101C:E1830004    orr r0, r3, r4
```