

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

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

FP_Add ▾ | F1 ▾ | F1 ▾ | F1 ▾ |

☐ Data Forwarding

Instruction	CPU Cycles														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 int_add (R7, R1, R2)	IF	ID	+- (i)	MEM	WB										
1 int_sub (R3, R7, R4)		IF	ID	S	S	+- (i)	MEM	WB							
2 fp_ld (F1, Offset, R1)			IF	S	S	ID	EX	MEM	WB						
3 fp_add (F5, F1, F1)						IF	ID	S	S	+- (f)	MEM	WB			

Step |

Potential Hazards:

RAW: Instructions 0 and 1. Register R7.
RAW: Instructions 2 and 3. Register 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

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
Remove Instruction

Help
Reset Application

Instruction	CPU Cycles														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 int_add (R7, R1, R2)	IF														
1 int_sub (R3, R7, R4)		IF													
2 fp_ld (F1, Offset, R1)			IF												
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.

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

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 Breakdown

TAG	INDEX	OFFSET
5 bit	5 bit	6 bit

Memory Block

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

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

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]

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

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]

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

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

Reset Submit

Instruction
Load (in hex#)
List of next 10 instructions
Load CB88 Submit

Information
The cycle has been completed.
Please submit another instructions

Next Fast Forward

Statistics
Hit Rate : 0%

4-WAY SET ASSOCIATIVE CACHE

Instruction Breakdown
0010101 111 000011
7 bit 3 bit 6 bit

Memory Block
B AF W 0 B AF W 1 B AF W 2 B AF W 3 B AF W 4 B AF W 5 B AF W 6 B AF W 7 B AF W 8 B AF W 9
B B0 W 0 B B0 W 1 B B0 W 2 B B0 W 3 B B0 W 4 B B0 W 5 B B0 W 6 B B0 W 7 B B0 W 8 B B0 W 9
B B1 W 0 B B1 W 1 B B1 W 2 B B1 W 3 B B1 W 4 B B1 W 5 B B1 W 6 B B1 W 7 B B1 W 8 B B1 W 9
B B2 W 0 B B2 W 1 B B2 W 2 B B2 W 3 B B2 W 4 B B2 W 5 B B2 W 6 B B2 W 7 B B2 W 8 B B2 W 9
R R3 W 0 R R3 W 1 R R3 W 2 R R3 W 3 R R3 W 4 R R3 W 5 R R3 W 6 R R3 W 7 R R3 W 8 R R3 W 9

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	0	-	0	0
2	1	0	B. 2 W. 0 - 63	0
3	0	-	0	0
4	1	24	B. 124 W. 0 - 63	0
5	0	-	0	0
6	1	65	B. 32E W. 0 - 63	0
7	1	67	B. 33F 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	1	15	B. AD W. 0 - 63	0
6	0	-	0	0
7	1	15	B. AF 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	1	8	B. 47 W. 0 - 63	0

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	1	59	B. 2C8 W. 0 - 63	0
1	0	-	0	0
2	0	-	0	0
3	0	-	0	0
4	1	4f	B. 27C W. 0 - 63	0
5	0	-	0	0
6	0	-	0	0
7	0	-	0	0

Statistics

Hit Rate : 0%

Miss Rate : 100%

List of Previous Instructions :

- Load CB88 [Miss]
- Load A937 [Miss]
- Load 490F [Miss]
- Load 9F3E [Miss]
- Load 11DA [Miss]
- Load CFF2 [Miss]
- Load 2B7B [Miss]
- Load B22C [Miss]
- Load A4 [Miss]
- Load 2BC3 [Miss]

Fully 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

6

Reset

Submit

Instruction

Load

(in hex)#

5ac7

List of next 10 Instructions

Load Previous Instructions

Submit

Information

The cycle has been completed.
Please submit another instructions

Next

Fast Forward

Statistics

Hit Rate

:

0%

FULLY ASSOCIATIVE CACHE

Instruction Breakdown

1010111111

100011

10 bit

6 bit

Memory Block

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

B 2C0 W 0 B 2C0 W 1 B 2C0 W 2 B 2C0 W 3 B 2C0 W 4 B 2C0 W 5 B 2C0 W 6 B 2C0 W 7 B 2C0 W 8 B 2C0 W 9

B 2C1 W 0 B 2C1 W 1 B 2C1 W 2 B 2C1 W 3 B 2C1 W 4 B 2C1 W 5 B 2C1 W 6 B 2C1 W 7 B 2C1 W 8 B 2C1 W 9

B 2C2 W 0 B 2C2 W 1 B 2C2 W 2 B 2C2 W 3 B 2C2 W 4 B 2C2 W 5 B 2C2 W 6 B 2C2 W 7 B 2C2 W 8 B 2C2 W 9

B 2C3 W 0 B 2C3 W 1 B 2C3 W 2 B 2C3 W 3 B 2C3 W 4 B 2C3 W 5 B 2C3 W 6 B 2C3 W 7 B 2C3 W 8 B 2C3 W 9

Cache Table

Index	Valid	Tag	Data (Hex)	Dirty Bit
0	0	-	0	0
1	1	0001100101	BLOCK 65 WORD 0 - 63	0
2	1	0001000101	BLOCK 45 WORD 0 - 63	0
3	0	-	0	0
4	0	-	0	0
5	0	-	0	0
6	0	-	0	0
7	1	1010111111	BLOCK 2BF WORD 0 - 63	0
8	0	-	0	0
9	0	-	0	0
10	0	-	0	0
11	1	1011010110	BLOCK 2D6 WORD 0 - 63	0
12	0	-	0	0
13	0	-	0	0
14	0	-	0	0
15	1	0011001001	BLOCK C9 WORD 0 - 63	0
16	1	0000111010	BLOCK 3A WORD 0 - 63	0
17	1	0100101101	BLOCK 12D WORD 0 - 63	0
18	0	-	0	0

Statistics

Hit Rate

:

0%

Miss Rate

:

100%

List of Previous Instructions :

• Load F640 [Miss]

• Load EA3 [Miss]

• Load B5B6 [Miss]

• Load 114D [Miss]

• Load 363D [Miss]

• Load 4B79 [Miss]

• Load 1963 [Miss]

• Load 326C [Miss]

• Load DA75 [Miss]

• Load AFE3 [Miss]

Next Index:

-

Last Index:

-

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
```

The screenshot displays the ARM simulator interface. On the left, the **RegistersView** shows the state of the registers: R0 is 5, R1 is 10, R2 is 2, R3 is 13, and the Program Counter (PC) is 4116. The **PluginUIView** on the right shows the assembly code being executed, with the instruction **SWI 0x011** highlighted. The **OutputView** at the bottom shows the execution log, including the loading of the assembly file and the execution ending.

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

```
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
```

The screenshot displays the ARM simulator interface for the second program. The **RegistersView** shows R0=10, R1=20, R2=15, R3=20, and the PC is 1010. The **PluginUIView** shows the assembly code, with **SWI 0x011** highlighted. The **OutputView** shows the execution log, including the loading of the assembly file and the execution ending.