Mohamad Mahdi Ziaee

Homework 7

cc-NUMA simulator

Table of Contents

[Description 1](#_Toc467421853)

[Project information 1](#_Toc467421854)

[Sample output 2](#_Toc467421855)

[Source code 15](#_Toc467421856)

[com.ccnuma.main.MainApplication 15](#_Toc467421857)

[com.ccnuma.pojo.CacheEntry 15](#_Toc467421858)

[com.ccnuma.pojo.CPU 16](#_Toc467421859)

[com.ccnuma.pojo.DirectoryEntry 16](#_Toc467421860)

[com.ccnuma.pojo.DirectoryEntryState 17](#_Toc467421861)

[com.ccnuma.pojo.IInstruction 17](#_Toc467421862)

[com.ccnuma.pojo.Instruction 18](#_Toc467421863)

[com.ccnuma.pojo.InstructionType 19](#_Toc467421864)

[com.ccnuma.pojo.LoadInstruction 20](#_Toc467421865)

[com.ccnuma.pojo.Node 20](#_Toc467421866)

[com.ccnuma.pojo.NUMASystem 21](#_Toc467421867)

[com.ccnuma.pojo.StoreInstruction 21](#_Toc467421868)

[com.ccnuma.util.FileUtils 22](#_Toc467421869)

[com.ccnuma.util.SystemSimulatorUtils 22](#_Toc467421870)

# Description

This application represents a cc-NUMA simulator with directory-based cache coherence control. The simulation uses DASH machine protocols.

To execute the application, put the input file (input.txt) into io folder under the root of the project. Then execute the main application in com.ccnuma.main.MainApplication. The application output will be printed in the console.

# Project information

Language: JAVA

Version: 1.8

Github: <https://github.com/momazia/ComputerArchitecture/tree/master/cc-NUMA>

Junit test version: 4

# Sample output

10001100000100010000000001101100 | Access Cost: 100

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000010111 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000100000 01 1 0 0 0

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 00 0 0 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

10001100000100100000000001101100 | Access Cost: 30

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000100000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000010111 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000100000 01 1 0 0 0

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 00 0 0 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

10101100000100010000000001001000 | Access Cost: 100

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000100000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000100000 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000100000 01 1 0 0 0

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 00 0 0 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

10001100000100010000000001101100 | Access Cost: 100

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000100000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000100000 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000100000 01 1 1 0 0

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 00 0 0 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

10001100000100010000000001101100 | Access Cost: 100

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000100000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000100000 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000100000 01 1 1 1 0

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 00 0 0 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

10001100000100010000000011100100 | Access Cost: 100

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000100000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000111110

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 1 1110 00000000000000000000000000111110

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000100000 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000100000 01 1 1 1 0

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 01 0 1 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

10101100000100010000000001101100 | Access Cost: 1

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000100000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0110 00000000000000000000000000100000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000111110

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 1 1110 00000000000000000000000000111110

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000111110

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000100000 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000100000 10 1 1 1 0

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 01 0 1 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

10001100000100010000000001101100 | Access Cost: 135

------------------------- Node (00) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000100000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0110 00000000000000000000000000100000

i Memory Directory

0 00000000000000000000000000000101 00 0 0 0 0

1 00000000000000000000000000000110 00 0 0 0 0

2 00000000000000000000000000000111 00 0 0 0 0

3 00000000000000000000000000001000 00 0 0 0 0

4 00000000000000000000000000001001 00 0 0 0 0

5 00000000000000000000000000001010 00 0 0 0 0

6 00000000000000000000000000001011 00 0 0 0 0

7 00000000000000000000000000001100 00 0 0 0 0

8 00000000000000000000000000001101 00 0 0 0 0

9 00000000000000000000000000001110 00 0 0 0 0

10 00000000000000000000000000001111 00 0 0 0 0

11 00000000000000000000000000010000 00 0 0 0 0

12 00000000000000000000000000010001 00 0 0 0 0

13 00000000000000000000000000010010 00 0 0 0 0

14 00000000000000000000000000010011 00 0 0 0 0

15 00000000000000000000000000010100 00 0 0 0 0

------------------------- Node (01) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000111110

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 1 1110 00000000000000000000000000111110

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000111110

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

16 00000000000000000000000000010101 00 0 0 0 0

17 00000000000000000000000000010110 00 0 0 0 0

18 00000000000000000000000000100000 00 0 0 0 0

19 00000000000000000000000000011000 00 0 0 0 0

20 00000000000000000000000000011001 00 0 0 0 0

21 00000000000000000000000000011010 00 0 0 0 0

22 00000000000000000000000000011011 00 0 0 0 0

23 00000000000000000000000000011100 00 0 0 0 0

24 00000000000000000000000000011101 00 0 0 0 0

25 00000000000000000000000000011110 00 0 0 0 0

26 00000000000000000000000000011111 00 0 0 0 0

27 00000000000000000000000000111110 01 1 1 1 1

28 00000000000000000000000000100001 00 0 0 0 0

29 00000000000000000000000000100010 00 0 0 0 0

30 00000000000000000000000000100011 00 0 0 0 0

31 00000000000000000000000000100100 00 0 0 0 0

------------------------- Node (10) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000100000

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0110 00000000000000000000000000100000

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

32 00000000000000000000000000100101 00 0 0 0 0

33 00000000000000000000000000100110 00 0 0 0 0

34 00000000000000000000000000100111 00 0 0 0 0

35 00000000000000000000000000101000 00 0 0 0 0

36 00000000000000000000000000101001 00 0 0 0 0

37 00000000000000000000000000101010 00 0 0 0 0

38 00000000000000000000000000101011 00 0 0 0 0

39 00000000000000000000000000101100 00 0 0 0 0

40 00000000000000000000000000101101 00 0 0 0 0

41 00000000000000000000000000101110 00 0 0 0 0

42 00000000000000000000000000101111 00 0 0 0 0

43 00000000000000000000000000110000 00 0 0 0 0

44 00000000000000000000000000110001 00 0 0 0 0

45 00000000000000000000000000110010 00 0 0 0 0

46 00000000000000000000000000110011 00 0 0 0 0

47 00000000000000000000000000110100 00 0 0 0 0

------------------------- Node (11) --------------------------

CPU 0:

Register:

S1: 00000000000000000000000000111110

S2: 00000000000000000000000000000000

Cache 0:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 1 0110 00000000000000000000000000111110

CPU 1:

Register:

S1: 00000000000000000000000000000000

S2: 00000000000000000000000000000000

Cache 1:

i v Tag Value

0 0 0000 00000000000000000000000000000000

1 0 0000 00000000000000000000000000000000

2 0 0000 00000000000000000000000000000000

3 0 0000 00000000000000000000000000000000

i Memory Directory

48 00000000000000000000000000110101 00 0 0 0 0

49 00000000000000000000000000110110 00 0 0 0 0

50 00000000000000000000000000110111 00 0 0 0 0

51 00000000000000000000000000111000 00 0 0 0 0

52 00000000000000000000000000111001 00 0 0 0 0

53 00000000000000000000000000111010 00 0 0 0 0

54 00000000000000000000000000111011 00 0 0 0 0

55 00000000000000000000000000111100 00 0 0 0 0

56 00000000000000000000000000111101 00 0 0 0 0

57 00000000000000000000000000111110 01 0 1 0 0

58 00000000000000000000000000111111 00 0 0 0 0

59 00000000000000000000000001000000 00 0 0 0 0

60 00000000000000000000000001000001 00 0 0 0 0

61 00000000000000000000000001000010 00 0 0 0 0

62 00000000000000000000000001000011 00 0 0 0 0

63 00000000000000000000000001000100 00 0 0 0 0

Total Cost: 666 Average Cost: 83.25

# Source code

## com.ccnuma.main.MainApplication

package com.ccnuma.main;

import java.io.IOException;

import java.util.List;

import com.ccnuma.pojo.Instruction;

import com.ccnuma.pojo.NUMASystem;

import com.ccnuma.util.FileUtils;

import com.ccnuma.util.SystemSimulatorUtils;

/\*\*

\* This application runs a cc-NUMa architecture simulator. In order to execute the main application, an input file must be placed under io folder and run the main method below.

\*

\* @author Mahdi Ziaee

\*

\*/

public class MainApplication {

/\*\*

\* Main method to be executed for simulation.

\*

\* @param args

\* @throws IOException

\*/

public static void main(String[] args) throws IOException {

// Reading the instruction file

List<Instruction> instructions = FileUtils.getInstance().readInstructionFile("input.txt");

// Initializing the system

NUMASystem system = SystemSimulatorUtils.getInstance().initializeSimulator();

// Accessing costs initialization

int totalCost = 0;

// Executing each of the instructions

for (Instruction instruction : instructions) {

// Executing the instruction

Integer accessCost = instruction.getIInstruction().execute(system, instruction.getNode(), instruction.getCpu());

totalCost += accessCost;

// Printing the result

SystemSimulatorUtils.getInstance().print(instruction, accessCost);

SystemSimulatorUtils.getInstance().print(system);

}

System.out.println("Total Cost: " + totalCost + " Average Cost: " + (double) totalCost / instructions.size());

}

}

## com.ccnuma.pojo.CacheEntry

package com.ccnuma.pojo;

/\*\*

\* A POJO to represent an entry inside the cache which contains a valid bit, tag field and the actual value.

\*

\* @author Mahdi Ziaee

\*

\*/

public class CacheEntry {

private boolean validBit = false;

private Integer tagField;

private Integer value;

/\*\*

\* The main constructor to set the values.

\*

\* @param validBit

\* @param tagField

\* @param value

\*/

public CacheEntry(boolean validBit, Integer tagField, Integer value) {

this.validBit = validBit;

this.setTagField(tagField);

this.value = value;

}

public CacheEntry() {

}

public boolean isValidBit() {

return validBit;

}

public void setValidBit(boolean validBit) {

this.validBit = validBit;

}

public Integer getValue() {

return value;

}

public void setValue(Integer value) {

this.value = value;

}

public Integer getTagField() {

return tagField;

}

public void setTagField(Integer tagField) {

this.tagField = tagField;

}

}

## com.ccnuma.pojo.CPU

package com.ccnuma.pojo;

import java.util.Map;

/\*\*

\* A POJO to hold the data related to a CPU, like registers and the cache entries.

\*

\* @author Mahdi Ziaee

\*

\*/

public class CPU {

private Map<Integer, Integer> registers;

private Map<Integer, CacheEntry> cacheEntries;

public Map<Integer, CacheEntry> getCacheEntries() {

return cacheEntries;

}

public void setCacheEntries(Map<Integer, CacheEntry> cacheEntries) {

this.cacheEntries = cacheEntries;

}

public Map<Integer, Integer> getRegisters() {

return registers;

}

public void setRegisters(Map<Integer, Integer> registers) {

this.registers = registers;

}

}

## com.ccnuma.pojo.DirectoryEntry

package com.ccnuma.pojo;

import java.util.Map;

/\*\*

\* A POJO to represent an entry inside the directory. Each entry contains a state and a set of values for each of the nodes in the system.

\*

\* @author Mahdi Ziaee

\*

\*/

public class DirectoryEntry {

private Map<Integer, Integer> values;

private DirectoryEntryState state;

public Map<Integer, Integer> getValues() {

return values;

}

public void setValues(Map<Integer, Integer> values) {

this.values = values;

}

public DirectoryEntryState getState() {

return state;

}

public void setState(DirectoryEntryState state) {

this.state = state;

}

}

## com.ccnuma.pojo.DirectoryEntryState

package com.ccnuma.pojo;

/\*\*

\* An enumeration for different states of directory.

\*

\* @author Mahdi Ziaee

\*

\*/

public enum DirectoryEntryState {

UN\_CACHED(0), SHARED(1), DIRTY(2);

private int value;

DirectoryEntryState(int value) {

this.value = value;

}

public int getValue() {

return this.value;

}

}

## com.ccnuma.pojo.IInstruction

package com.ccnuma.pojo;

/\*\*

\* A POJO to represent the instructions read from the input file. It follows I-Instruction structure.

\*

\* @author Mahdi Ziaee

\*

\*/

public abstract class IInstruction {

private Integer rs;

private Integer rt;

private Integer offset;

/\*\*

\* The main constructors which sets the values into the properties of this class.

\*

\* @param instruction

\*/

public IInstruction(String instruction) {

this.rs = Integer.parseInt(instruction.substring(6, 11), 2);

this.rt = Integer.parseInt(instruction.substring(11, 16), 2);

this.offset = Integer.parseInt(instruction.substring(16), 2);

}

/\*\*

\* The main method to be implemented by different types of instructions which represents the instruction's execution in the system.

\*

\* @param system

\* @param nodeNumber

\* @param cpuNumber

\* @return

\*/

public abstract Integer execute(NUMASystem system, Integer nodeNumber, Integer cpuNumber);

public Integer getRs() {

return rs;

}

public void setRs(Integer rs) {

this.rs = rs;

}

public Integer getRt() {

return rt;

}

public void setRt(Integer rt) {

this.rt = rt;

}

public Integer getOffset() {

return offset;

}

public void setOffset(Integer offset) {

this.offset = offset;

}

}

## com.ccnuma.pojo.Instruction

package com.ccnuma.pojo;

/\*\*

\* A POJO to represent the raw / actual input of the simulator.

\*

\* @author Mahdi Ziaee

\*

\*/

public class Instruction {

private Integer node;

private Integer cpu;

private IInstruction iInstruction;

private String raw;

public Instruction() {

}

/\*\*

\* Sets the properties of this class and uses the factory to instantiate proper instruction type.

\*

\* @param node

\* @param cpu

\* @param instruction

\*/

public Instruction(Integer node, Integer cpu, String instruction) {

super();

this.setNode(node);

this.setCpu(cpu);

this.iInstruction = instructionFactory(instruction);

this.raw = instruction;

}

/\*\*

\* A factory to instantiate a proper I-Instruction type based on the first 6 bits.

\*

\* @param instruction

\* @return

\*/

private IInstruction instructionFactory(String instruction) {

switch (InstructionType.findValueOf(instruction)) {

case LOAD:

return new LoadInstruction(instruction);

case STORE:

return new StoreInstruction(instruction);

}

return null;

}

public IInstruction getIInstruction() {

return iInstruction;

}

public void setIInstruction(IInstruction instruction) {

this.iInstruction = instruction;

}

public Integer getNode() {

return node;

}

public void setNode(Integer node) {

this.node = node;

}

public Integer getCpu() {

return cpu;

}

public void setCpu(Integer cpu) {

this.cpu = cpu;

}

public String getRaw() {

return this.raw;

}

}

## com.ccnuma.pojo.InstructionType

package com.ccnuma.pojo;

/\*\*

\* An enumeration to represent different instruction types.

\*

\* @author Mahdi Ziaee

\*

\*/

public enum InstructionType {

LOAD("100011"), STORE("101011");

private String value;

private InstructionType(String value) {

this.value = value;

}

public String getValue() {

return this.value;

}

/\*\*

\* Finds the type of the instruction passed based on the first 6 bits. If it does not match, it returns null.

\*

\* @param instruction

\* @return

\*/

public static InstructionType findValueOf(String instruction) {

if (instruction == null) {

return null;

}

for (InstructionType type : InstructionType.values()) {

if (instruction.substring(0, 6).equals(type.getValue())) {

return type;

}

}

return null;

}

}

## com.ccnuma.pojo.LoadInstruction

package com.ccnuma.pojo;

import com.ccnuma.util.SystemSimulatorUtils;

/\*\*

\* The class represent a load instruction. A load instruction reads the data.

\*

\* @author Mahdi Ziaee

\*

\*/

public class LoadInstruction extends IInstruction {

public LoadInstruction(String instruction) {

super(instruction);

}

@Override

public Integer execute(NUMASystem system, Integer nodeNumber, Integer cpuNumber) {

return SystemSimulatorUtils.getInstance().read(system, nodeNumber, cpuNumber, this);

}

}

## com.ccnuma.pojo.Node

package com.ccnuma.pojo;

import java.util.Map;

/\*\*

\* A POJO to represent each node in the system. It contains a set of CPUs, memory blocks and directory entries.

\*

\* @author Mahdi Ziaee

\*

\*/

public class Node {

private Map<Integer, CPU> cpus;

private Map<Integer, Integer> memoryBlocks;

private Map<Integer, DirectoryEntry> directoryEntries;

public Map<Integer, DirectoryEntry> getDirectoryEntries() {

return directoryEntries;

}

public void setDirectoryEntries(Map<Integer, DirectoryEntry> directoryEntries) {

this.directoryEntries = directoryEntries;

}

public Map<Integer, Integer> getMemoryBlocks() {

return memoryBlocks;

}

public void setMemoryBlocks(Map<Integer, Integer> memoryBlocks) {

this.memoryBlocks = memoryBlocks;

}

public Map<Integer, CPU> getCpus() {

return cpus;

}

public void setCpus(Map<Integer, CPU> cpus) {

this.cpus = cpus;

}

}

## com.ccnuma.pojo.NUMASystem

package com.ccnuma.pojo;

import java.util.Map;

/\*\*

\* The main POJO to hold different set of nodes in the simulator.

\*

\* @author Mahdi Ziaee

\*

\*/

public class NUMASystem {

private Map<Integer, Node> nodes;

public Map<Integer, Node> getNodes() {

return nodes;

}

public void setNodes(Map<Integer, Node> nodes) {

this.nodes = nodes;

}

}

## com.ccnuma.pojo.StoreInstruction

package com.ccnuma.pojo;

import com.ccnuma.util.SystemSimulatorUtils;

/\*\*

\* The class represent a store instruction. A store instruction writes data.

\*

\* @author Mahdi Ziaee

\*

\*/

public class StoreInstruction extends IInstruction {

public StoreInstruction(String instruction) {

super(instruction);

}

@Override

public Integer execute(NUMASystem system, Integer nodeNumber, Integer cpuNumber) {

return SystemSimulatorUtils.getInstance().write(system, nodeNumber, cpuNumber, this);

}

}

## com.ccnuma.util.FileUtils

package com.ccnuma.util;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.List;

import com.ccnuma.pojo.Instruction;

/\*\*

\* A utility class which takes care of all the file related logics.

\*

\* @author Mahdi Ziaee

\*

\*/

public class FileUtils {

private static final String IO\_PATH = "../cc-NUMA/io/";

/\*\*

\* static Singleton instance

\*/

private static FileUtils instance;

/\*\*

\* Private constructor for singleton

\*/

private FileUtils() {

}

/\*\*

\* Static getter method for retrieving the singleton instance

\*/

public static FileUtils getInstance() {

if (instance == null) {

instance = new FileUtils();

}

return instance;

}

/\*\*

\* Reads the content of the fileName given and constructs a list of instructions representing the binary instructions in the file.

\*

\* @param fileName

\* @return

\* @throws IOException

\*/

public List<Instruction> readInstructionFile(String fileName) throws IOException {

List<String> lines = Files.readAllLines(Paths.get(FileUtils.IO\_PATH + fileName));

List<Instruction> instructions = new ArrayList<>();

for (String line : lines) {

instructions.add(new Instruction(Integer.parseInt(line.substring(0, 2), 2), Integer.parseInt(line.substring(2, 3), 2), line.substring(5)));

}

return instructions;

}

}

## com.ccnuma.util.SystemSimulatorUtils

package com.ccnuma.util;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import com.ccnuma.pojo.CPU;

import com.ccnuma.pojo.CacheEntry;

import com.ccnuma.pojo.DirectoryEntry;

import com.ccnuma.pojo.DirectoryEntryState;

import com.ccnuma.pojo.IInstruction;

import com.ccnuma.pojo.Instruction;

import com.ccnuma.pojo.NUMASystem;

import com.ccnuma.pojo.Node;

/\*\*

\* The main simulator utility class.

\*

\* @author Mahdi Ziaee

\*

\*/

public class SystemSimulatorUtils {

private static final int RT\_BASE\_NUMBER = 16;

private static final int WORD\_SIZE = 32;

private static final int NUMBER\_OF\_WORDS\_IN\_MEMORY = 16;

private static final int NUMBER\_OF\_NODES = 4;

private static final int NUMBER\_OF\_REGISTERS = 2;

private static final int NUMBER\_OF\_CPUS = 2;

private static final int NUMBER\_OF\_CACHE\_ROWS = 4;

/\*\*

\* static Singleton instance

\*/

private static SystemSimulatorUtils instance;

/\*\*

\* Private constructor for singleton

\*/

private SystemSimulatorUtils() {

}

/\*\*

\* Static getter method for retrieving the singleton instance

\*/

public static SystemSimulatorUtils getInstance() {

if (instance == null) {

instance = new SystemSimulatorUtils();

}

return instance;

}

/\*\*

\* Initializes the simulator. The main method to be called before using the system.

\*

\* @return

\*/

public NUMASystem initializeSimulator() {

NUMASystem system = new NUMASystem();

system.setNodes(initializeNodes());

return system;

}

/\*\*

\* Initializes each nodes in the systems.

\*

\* @return

\*/

public Map<Integer, Node> initializeNodes() {

Map<Integer, Node> nodes = new HashMap<>();

for (int i = 0; i < NUMBER\_OF\_NODES; i++) {

nodes.put(i, initializeNode(i));

}

return nodes;

}

/\*\*

\* Initializes individual nodes based on the node number passed.

\*

\* @param nodeNumber

\* @return

\*/

private Node initializeNode(int nodeNumber) {

Node node = new Node();

node.setCpus(initializeCPUs());

node.setDirectoryEntries(initializeDirectoryEntries(nodeNumber));

node.setMemoryBlocks(initializeMemoryBlocks(nodeNumber));

return node;

}

/\*\*

\* Initializes the memory blocks needed in the node number given.

\*

\* @param nodeNumber

\* @return

\*/

public Map<Integer, Integer> initializeMemoryBlocks(int nodeNumber) {

Map<Integer, Integer> result = new HashMap<>();

int startIndex = nodeNumber \* NUMBER\_OF\_WORDS\_IN\_MEMORY;

int endIndex = startIndex + NUMBER\_OF\_WORDS\_IN\_MEMORY;

for (int i = startIndex; i < endIndex; i++) {

result.put(i, i + 5);

}

return result;

}

/\*\*

\* Initializes the directory entries for the given node number.

\*

\* @param nodeNumber

\* @return

\*/

private Map<Integer, DirectoryEntry> initializeDirectoryEntries(int nodeNumber) {

Map<Integer, DirectoryEntry> entries = new HashMap<>();

int startIndex = nodeNumber \* NUMBER\_OF\_WORDS\_IN\_MEMORY;

int endIndex = startIndex + NUMBER\_OF\_WORDS\_IN\_MEMORY;

for (int i = startIndex; i < endIndex; i++) {

entries.put(i, initializeDirectoryEntry());

}

return entries;

}

/\*\*

\* Initializes a directory entry and sets the default values.

\*

\* @return

\*/

private DirectoryEntry initializeDirectoryEntry() {

DirectoryEntry entry = new DirectoryEntry();

Map<Integer, Integer> values = new HashMap<>();

for (int i = 0; i < NUMBER\_OF\_NODES; i++) {

values.put(i, 0);

}

entry.setValues(values);

entry.setState(DirectoryEntryState.UN\_CACHED);

return entry;

}

/\*\*

\* Initializes CPUs in the system.

\*

\* @return

\*/

public Map<Integer, CPU> initializeCPUs() {

Map<Integer, CPU> cpus = new HashMap<>();

for (int i = 0; i < NUMBER\_OF\_CPUS; i++) {

CPU cpu = new CPU();

cpu.setRegisters(initializeRegisters());

cpu.setCacheEntries(initializeCacheEntries());

cpus.put(i, cpu);

}

return cpus;

}

/\*\*

\* Initializes all the cache entries in the system.

\*

\* @return

\*/

private Map<Integer, CacheEntry> initializeCacheEntries() {

Map<Integer, CacheEntry> entries = new HashMap<>();

for (int i = 0; i < NUMBER\_OF\_CACHE\_ROWS; i++) {

entries.put(i, new CacheEntry());

}

return entries;

}

/\*\*

\* Initializes the registers with default value.

\*

\* @return

\*/

private Map<Integer, Integer> initializeRegisters() {

Map<Integer, Integer> registers = new HashMap<>();

for (int i = 1; i <= NUMBER\_OF\_REGISTERS; i++) {

registers.put(i, 0);

}

return registers;

}

/\*\*

\* Prints the system file given into the console. It shows all the nodes information like cache data, memory content etc.

\*

\* @param system

\*/

public void print(NUMASystem system) {

Map<Integer, Node> nodes = system.getNodes();

// Looping through the nodes

for (int i = 0; i < NUMBER\_OF\_NODES; i++) {

String nodeNumber = format(i, 2);

System.out.println("\t------------------------- Node (" + nodeNumber + ") --------------------------");

// Going through the CPUs

for (Integer cpuNumber : nodes.get(i).getCpus().keySet()) {

CPU cpu = nodes.get(i).getCpus().get(cpuNumber);

System.out.println("CPU " + cpuNumber + ":");

// Printing the registers

System.out.println("\tRegister:");

for (Integer registerNumber : cpu.getRegisters().keySet()) {

System.out.println("\t\tS" + registerNumber + ": " + format(cpu.getRegisters().get(registerNumber)));

}

// Printing the cache content

System.out.println("\tCache " + cpuNumber + ":");

System.out.println("\t\ti v Tag Value");

for (int j = 0; j < NUMBER\_OF\_CACHE\_ROWS; j++) {

CacheEntry cacheEntry = cpu.getCacheEntries().get(j);

System.out.println("\t\t" + j + " " + printBoolean(cacheEntry.isValidBit()) + " " + format(cacheEntry.getTagField(), 4) + " " + format(cacheEntry.getValue()));

}

System.out.println();

}

// Printing memory and directory contents

int startIndex = i \* NUMBER\_OF\_WORDS\_IN\_MEMORY;

int endIndex = startIndex + NUMBER\_OF\_WORDS\_IN\_MEMORY;

System.out.println("i\tMemory\t\t\t\t\tDirectory");

Map<Integer, Integer> memoryBlocks = nodes.get(i).getMemoryBlocks();

Map<Integer, DirectoryEntry> directoryEntries = nodes.get(i).getDirectoryEntries();

for (int j = startIndex; j < endIndex; j++) {

System.out.println(j + "\t" + format(memoryBlocks.get(j)) + "\t" + printDirectoryValues(directoryEntries.get(j)));

}

System.out.println();

}

}

/\*\*

\* Formats the given value with padding size of 32.

\*

\* @param value

\* @return

\*/

private String format(Integer value) {

return format(value, WORD\_SIZE);

}

/\*\*

\* Formats the given value using the padding size given. If the value is null, it will be set to zero.

\*

\* @param value

\* @param padding

\* @return

\*/

private String format(Integer value, int padding) {

if (value == null) {

value = 0;

}

return String.format("%" + padding + "s", Integer.toBinaryString(value)).replace(' ', '0');

}

/\*\*

\* Prints the content of the directory values for all the nodes in the system.

\*

\* @param directoryEntry

\* @return

\*/

private String printDirectoryValues(DirectoryEntry directoryEntry) {

String result = format(directoryEntry.getState().getValue(), 2) + " ";

for (int i = 0; i < NUMBER\_OF\_NODES; i++) {

result += directoryEntry.getValues().get(i) + " ";

}

return result;

}

/\*\*

\* Converts the boolean value to 1 = true, 0 = false format.

\*

\* @param bool

\* @return

\*/

private int printBoolean(boolean bool) {

return bool ? 1 : 0;

}

/\*\*

\* Goes through the 2 write steps needed for the given instruction and returns accessing cost for that instruction.

\*

\* @param system

\* @param nodeNumber

\* @param cpuNumber

\* @param instruction

\* @return

\*/

public Integer write(NUMASystem system, int nodeNumber, int cpuNumber, IInstruction instruction) {

// 1. Search local cache

Integer localCacheValue = searchLocalCache(system.getNodes().get(nodeNumber).getCpus().get(cpuNumber), instruction);

int memoryAddress = getMemoryAddress(instruction);

Integer registerValue = getDataFromRegister(system, nodeNumber, cpuNumber, instruction);

if (localCacheValue != null) {

// Get exclusive access prior to writing the value

if (isDirectoryStateEqual(system, memoryAddress, DirectoryEntryState.SHARED)) {

// Invalidate other copies

invalidateCaches(system, instruction, memoryAddress);

// Update the cache value

storeDataIntoCache(system, nodeNumber, cpuNumber, registerValue, instruction);

// Change Directory state to Dirty

updateDirectory(system, nodeNumber, memoryAddress, DirectoryEntryState.DIRTY);

return 1;

}

}

// 2. Update home node memory

updateMemoryAndDirectory(system, memoryAddress, registerValue, instruction);

return 100;

}

/\*\*

\* Invalidates the other cache values by finding all the shared nodes from the directory and setting their valid bit to 0.

\*

\* @param system

\* @param instruction

\* @param memoryAddress

\*/

private void invalidateCaches(NUMASystem system, IInstruction instruction, int memoryAddress) {

List<Integer> sharedNodeNumbers = findSharedNodesFromDirectory(system, memoryAddress);

invalidateCaches(system, sharedNodeNumbers, instruction);

}

/\*\*

\* Updates the content of the memory and directory. This represents the second step in the write process.

\*

\* @param system

\* @param memoryAddress

\* @param value

\* @param instruction

\*/

private void updateMemoryAndDirectory(NUMASystem system, int memoryAddress, Integer value, IInstruction instruction) {

for (Node node : system.getNodes().values()) {

if (node.getMemoryBlocks().containsKey(memoryAddress)) {

// Updating the value of the memory

node.getMemoryBlocks().put(memoryAddress, value);

switch (node.getDirectoryEntries().get(memoryAddress).getState()) {

case UN\_CACHED: // Do nothing

break;

case SHARED:

// Invalidating other cache copies

invalidateCaches(system, instruction, memoryAddress);

// Won't change the state

break;

case DIRTY:

// Invalidating other cache copies

invalidateCaches(system, instruction, memoryAddress);

// Changing the state to SHARED

node.getDirectoryEntries().get(memoryAddress).setState(DirectoryEntryState.DIRTY);

break;

}

return;

}

}

}

/\*\*

\* Gets the data from register of the cpu for the node number given using the instruction passed.

\*

\* @param system

\* @param nodeNumber

\* @param cpuNumber

\* @param instruction

\* @return

\*/

private Integer getDataFromRegister(NUMASystem system, int nodeNumber, int cpuNumber, IInstruction instruction) {

Integer registerNumber = getRegisterNumber(instruction);

return system.getNodes().get(nodeNumber).getCpus().get(cpuNumber).getRegisters().get(registerNumber);

}

/\*\*

\* Invalidates the shared node numbers passed by setting their valid bit to false.

\*

\* @param system

\* @param sharedNodeNumbers

\* @param instruction

\*/

private void invalidateCaches(NUMASystem system, List<Integer> sharedNodeNumbers, IInstruction instruction) {

int cacheIndex = getCacheIndex(instruction);

Integer tagField = getTagField(instruction);

for (Integer sharedNodeNumber : sharedNodeNumbers) {

for (CPU cpu : system.getNodes().get(sharedNodeNumber).getCpus().values()) {

// Finding the other matching copies

if (cpu.getCacheEntries().containsKey(cacheIndex) && tagField.equals(cpu.getCacheEntries().get(cacheIndex).getTagField())) {

cpu.getCacheEntries().get(cacheIndex).setValidBit(false);

}

}

}

}

/\*\*

\* Checks to see if the memory address passed has the same state as the given state.

\*

\* @param system

\* @param memoryAddress

\* @param givenState

\* @return

\*/

private boolean isDirectoryStateEqual(NUMASystem system, int memoryAddress, DirectoryEntryState givenState) {

for (Node node : system.getNodes().values()) {

// Finding the home directory

if (node.getDirectoryEntries().containsKey(memoryAddress)) {

DirectoryEntryState state = node.getDirectoryEntries().get(memoryAddress).getState();

return givenState.equals(state);

}

}

return false;

}

/\*\*

\* The main read function takes care of the 4 main steps in reading an instruction.

\*

\* @param system

\* @param nodeNumber

\* @param cpuNumber

\* @param instruction

\* @return

\*/

public Integer read(NUMASystem system, int nodeNumber, int cpuNumber, IInstruction instruction) {

// 1. Searching local cache

Integer localCacheValue = searchLocalCache(system.getNodes().get(nodeNumber).getCpus().get(cpuNumber), instruction);

if (localCacheValue != null) {

// Loading the data into register

loadDataIntoRegister(system, nodeNumber, cpuNumber, instruction, localCacheValue);

return 1;

}

// 2. Searching other caches in the local node

int otherCpuNumber = (~cpuNumber) & 1;

localCacheValue = searchLocalCache(system.getNodes().get(nodeNumber).getCpus().get(otherCpuNumber), instruction);

if (localCacheValue != null) {

// Loading the data coming from other cache into into both local

// cache and register

loadDataIntoCacheAndRegister(system, nodeNumber, cpuNumber, instruction, localCacheValue);

return 30;

}

// 3. Searching the home node's memory/directory

int memoryAddress = getMemoryAddress(instruction);

Integer memoryValue = containsMostRecentCleanData(system, memoryAddress);

if (memoryValue != null) {

// Meaning the data is clean and it is the most recent version, So we load it into local cache and register

loadDataIntoCacheAndRegister(system, nodeNumber, cpuNumber, instruction, memoryValue);

// Updating the directory state and node flag

updateDirectory(system, nodeNumber, memoryAddress, DirectoryEntryState.SHARED);

return 100;

}

// 4. The data is dirty, looking up for the latest data in the dirty node

List<Integer> dirtyNodeNumbers = findSharedNodesFromDirectory(system, memoryAddress);

Integer latestValue = findLatestValue(system, instruction, dirtyNodeNumbers);

if (storeDataIntoMemory(system, memoryAddress, latestValue)) {

// Storage was successful, updating directory.

updateDirectory(system, nodeNumber, memoryAddress, DirectoryEntryState.SHARED);

// Reading the data from memory into cache and register

loadDataIntoCacheAndRegister(system, nodeNumber, cpuNumber, instruction, latestValue);

return 135;

}

return null;

}

/\*\*

\* Stores the value given into the memory address passed.

\*

\* @param system

\* @param memoryAddress

\* @param value

\* @return

\*/

private boolean storeDataIntoMemory(NUMASystem system, int memoryAddress, Integer value) {

for (Node node : system.getNodes().values()) {

// Finding the home directory

if (node.getMemoryBlocks().containsKey(memoryAddress)) {

node.getMemoryBlocks().put(memoryAddress, value);

return true;

}

}

return false;

}

/\*\*

\* Finds the latest valid value among all the dirty nodes by checking the tag field and index.

\*

\* @param system

\* @param instruction

\* @param dirtyNodeNumbers

\* @return

\*/

private Integer findLatestValue(NUMASystem system, IInstruction instruction, List<Integer> dirtyNodeNumbers) {

Integer localCacheValue;

for (Integer dirtyNodeNumber : dirtyNodeNumbers) {

// Looking at all cache values

for (int i = 0; i < NUMBER\_OF\_CPUS; i++) {

localCacheValue = searchLocalCache(system.getNodes().get(dirtyNodeNumber).getCpus().get(i), instruction);

if (localCacheValue != null) {

// Means we found the latest value

return localCacheValue;

}

}

}

return null;

}

/\*\*

\* Finds the list of all the shared nodes using the memory address in the directory.

\*

\* @param system

\* @param memoryAddress

\* @return

\*/

private List<Integer> findSharedNodesFromDirectory(NUMASystem system, int memoryAddress) {

List<Integer> result = new ArrayList<>();

for (Node node : system.getNodes().values()) {

// Finding the home directory

if (node.getDirectoryEntries().containsKey(memoryAddress)) {

Map<Integer, Integer> nodeFlags = node.getDirectoryEntries().get(memoryAddress).getValues();

for (Integer nodeNumber : nodeFlags.keySet()) {

if (nodeFlags.get(nodeNumber) == 1) {

result.add(nodeNumber);

}

}

}

}

return result;

}

/\*\*

\* Sets the state and value of the new node passed for the given memory address in the directory.

\*

\* @param system

\* @param newNodeNumber

\* @param memoryAddress

\* @param state

\*/

private void updateDirectory(NUMASystem system, int newNodeNumber, int memoryAddress, DirectoryEntryState state) {

for (Node node : system.getNodes().values()) {

// Finding the home directory

if (node.getDirectoryEntries().containsKey(memoryAddress)) {

node.getDirectoryEntries().get(memoryAddress).setState(state);

node.getDirectoryEntries().get(memoryAddress).getValues().put(newNodeNumber, 1);

}

}

}

/\*\*

\* Stores the value given into cache based on the instruction passed.

\*

\* @param system

\* @param nodeNumber

\* @param cpuNumber

\* @param value

\* @param instruction

\*/

private void storeDataIntoCache(NUMASystem system, int nodeNumber, int cpuNumber, Integer value, IInstruction instruction) {

Integer cacheIndex = getCacheIndex(instruction);

Integer tagField = getTagField(instruction);

system.getNodes().get(nodeNumber).getCpus().get(cpuNumber).getCacheEntries().put(cacheIndex, new CacheEntry(true, tagField, value));

}

/\*\*

\* Store the value into cache and register.

\*

\* @param system

\* @param nodeNumber

\* @param cpuNumber

\* @param instruction

\* @param value

\*/

private void loadDataIntoCacheAndRegister(NUMASystem system, int nodeNumber, int cpuNumber, IInstruction instruction, Integer value) {

Integer registerNumber = getRegisterNumber(instruction);

Integer cacheIndex = getCacheIndex(instruction);

Integer tagField = getTagField(instruction);

system.getNodes().get(nodeNumber).getCpus().get(cpuNumber).getRegisters().put(registerNumber, value);

system.getNodes().get(nodeNumber).getCpus().get(cpuNumber).getCacheEntries().put(cacheIndex, new CacheEntry(true, tagField, value));

}

/\*\*

\* Checks to see if the memory address given has state shared or un cached, if yes, it returns the value in the memory, else (dirty) returns null.

\*

\* @param system

\* @param memoryAddress

\* @return

\*/

private Integer containsMostRecentCleanData(NUMASystem system, int memoryAddress) {

for (Node node : system.getNodes().values()) {

if (node.getDirectoryEntries().containsKey(memoryAddress)) {

DirectoryEntryState state = node.getDirectoryEntries().get(memoryAddress).getState();

if (DirectoryEntryState.SHARED.equals(state) || DirectoryEntryState.UN\_CACHED.equals(state)) {

// Returning the memory value

return node.getMemoryBlocks().get(memoryAddress);

} else if (DirectoryEntryState.DIRTY.equals(state)) {

return null;

}

}

}

return null;

}

/\*\*

\* Sets the value passed into register.

\*

\* @param system

\* @param nodeNumber

\* @param cpuNumber

\* @param instruction

\* @param value

\*/

private void loadDataIntoRegister(NUMASystem system, int nodeNumber, int cpuNumber, IInstruction instruction, Integer value) {

Integer registerNumber = getRegisterNumber(instruction);

system.getNodes().get(nodeNumber).getCpus().get(cpuNumber).getRegisters().put(registerNumber, value);

}

/\*\*

\* Gets the RT register number from the load instruction

\*

\* @param instruction

\* @return

\*/

private Integer getRegisterNumber(IInstruction instruction) {

return instruction.getRt() - RT\_BASE\_NUMBER;

}

/\*\*

\* Searches the local cache to see if the valid bit is 1 and the tag field matches the instruction. If yes, it returns the value of the cache, otherwise null.

\*

\* @param cpu

\* @param instruction

\* @return

\*/

public Integer searchLocalCache(CPU cpu, IInstruction instruction) {

int cacheIndex = getCacheIndex(instruction);

// Checking the valid bit

CacheEntry cacheEntry = cpu.getCacheEntries().get(cacheIndex);

if (cacheEntry.isValidBit() && cacheEntry.getTagField() != null) {

int tag = getTagField(instruction);

if (cacheEntry.getTagField().equals(tag)) {

return cacheEntry.getValue();

}

}

return null;

}

/\*\*

\* Finds the memory address by right shifting 2 bits.

\*

\* @param instruction

\* @return

\*/

private int getMemoryAddress(IInstruction instruction) {

return instruction.getOffset() >> 2;

}

/\*\*

\* Calculates the tag field based on the instructions offset: 4 right shifts

\*

\* @param instruction

\* @return

\*/

private Integer getTagField(IInstruction instruction) {

return instruction.getOffset() >> 4;

}

/\*\*

\* Calculates the cache index based on the instruction given by getting

\* the most right two bits.

\*

\* @param instruction

\* @return

\*/

private int getCacheIndex(IInstruction instruction) {

return getMemoryAddress(instruction) & 3;

}

/\*\*

\* Prints the raw instruction content coming from the file in binary format and the access cost.

\*

\* @param instruction

\* @param accessCost

\*/

public void print(Instruction instruction, Integer accessCost) {

System.out.println(instruction.getRaw() + " | Access Cost: " + accessCost);

}

}