# Final Project Guidance

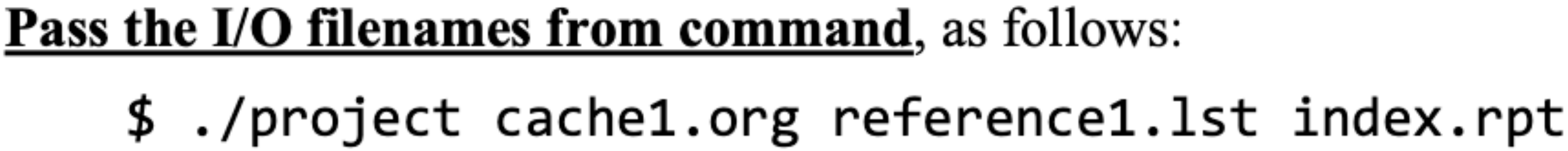
2022/12/19

林承賢

# Outline

1. Pass data from command line
2. Read / Output file
3. Data structure of Cache
4. Simulation
5. Part 2

# Pass the file name



**Pass the I/O filenames from command**, as follows:

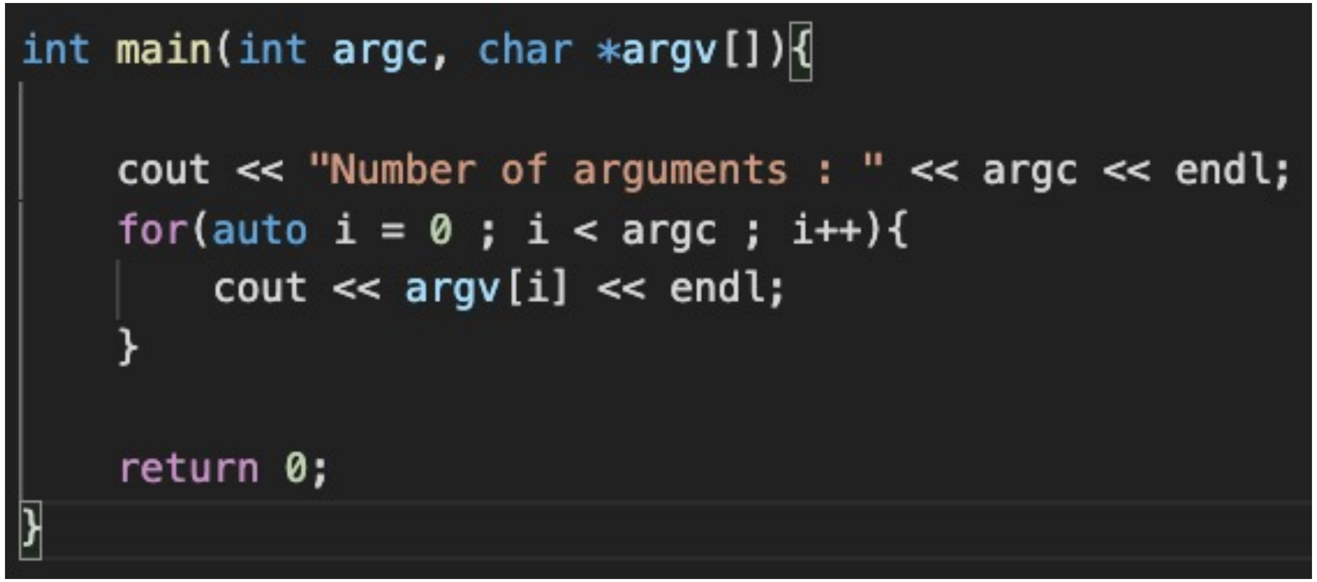
```
    $ ./project cache1.org reference1.lst index.rpt
```

You can use “argc” and ”argv” to pass the information of command line into your program.

# Example : argc , argv

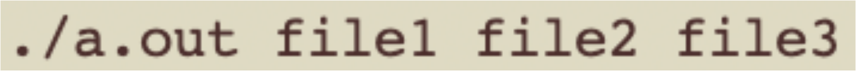
argc : The number of arguments in the command
argv : The array which store all of the arguments

Sample code



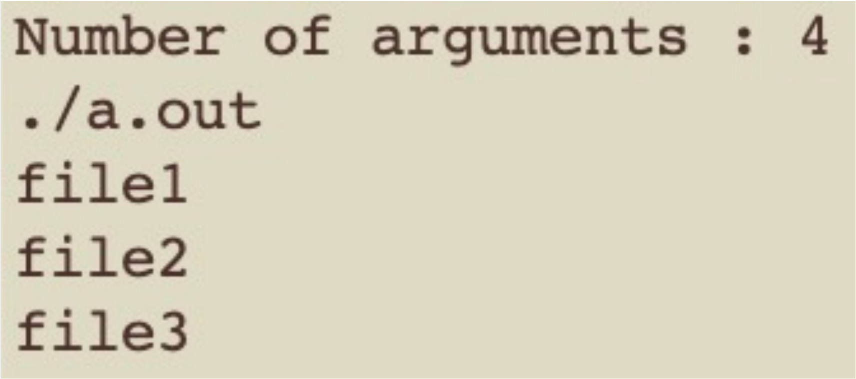
```
int main(int argc, char *argv[]){

    cout << "Number of arguments : " << argc << endl;
    for(auto i = 0 ; i < argc ; i++){
        cout << argv[i] << endl;
    }

    return 0;
}
```

Input



```
./a.out file1 file2 file3
```

Output



```
Number of arguments : 4
./a.out
file1
file2
file3
```

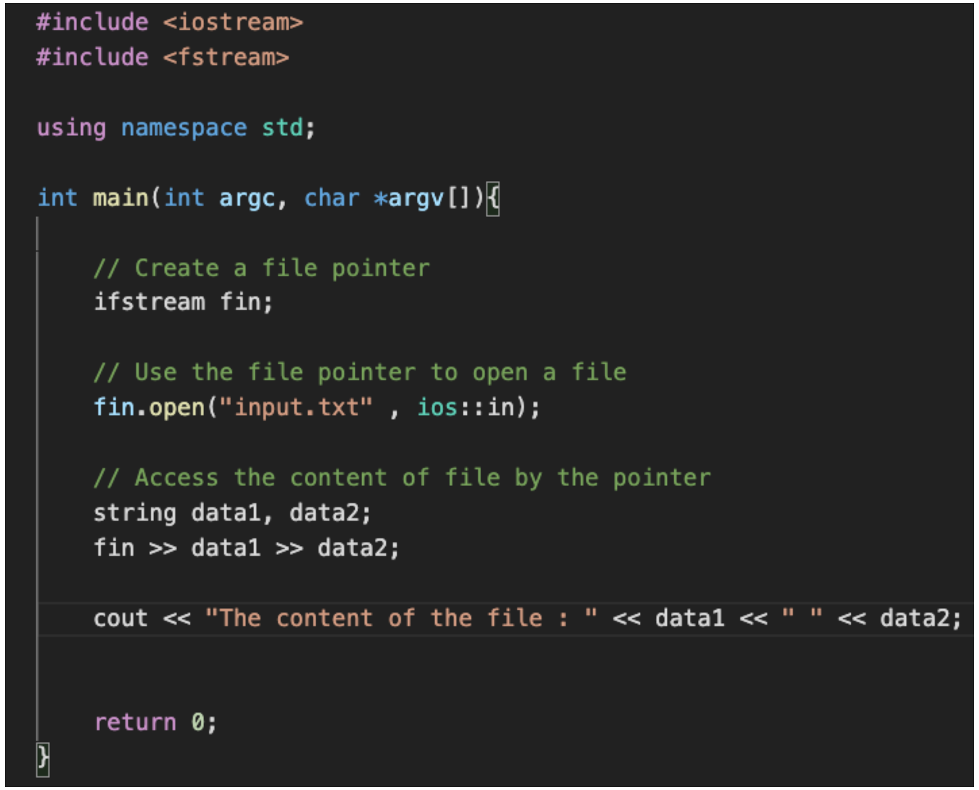
# Outline

1. Pass data from command line
2. Read / Output file
3. Data structure of Cache
4. Simulation
5. Part 2

# I/O in C++

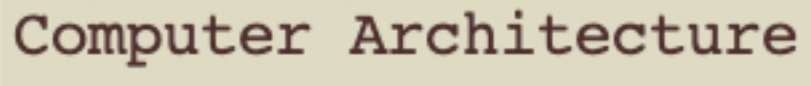
You can use <fstream> library to read or write a file.

Read example :



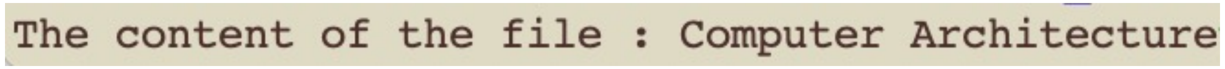
```
#include <iostream>
#include <fstream>

using namespace std;

int main(int argc, char *argv[]){

    // Create a file pointer
    ifstream fin;

    // Use the file pointer to open a file
    fin.open("input.txt" , ios::in);

    // Access the content of file by the pointer
    string data1, data2;
    fin >> data1 >> data2;

    cout << "The content of the file : " << data1 << " " << data2;


    return 0;
}
```

input.txt



```
Computer Architecture
```

Output of the program

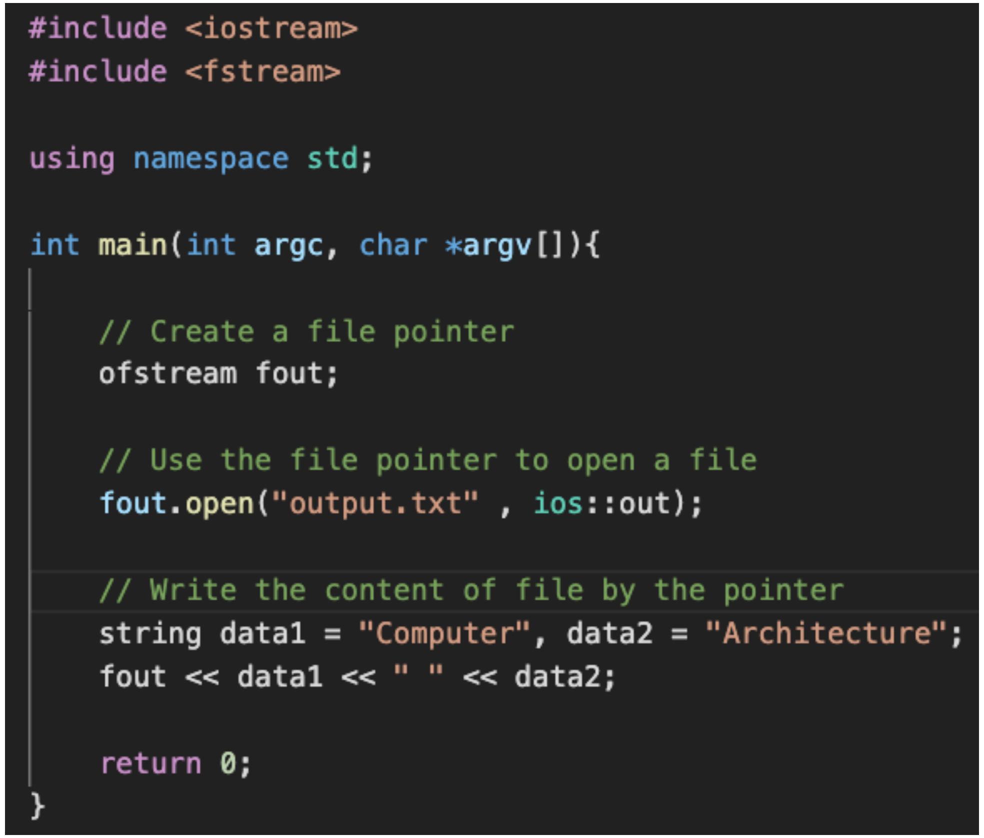


```
The content of the file : Computer Architecture
```

# I/O in C++

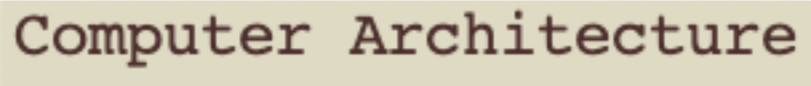
You can use <fstream> library to read or write a file.

Write example :



```
#include <iostream>
#include <fstream>

using namespace std;

int main(int argc, char *argv[]){

    // Create a file pointer
    ofstream fout;

    // Use the file pointer to open a file
    fout.open("output.txt" , ios::out);

    // Write the content of file by the pointer
    string data1 = "Computer", data2 = "Architecture";
    fout << data1 << " " << data2;

    return 0;
}
```

output.txt



```
Computer Architecture
```

# Outline

1. Pass data from command line
2. Read / Output file
3. Data structure of Cache
4. Simulation
5. Part 2

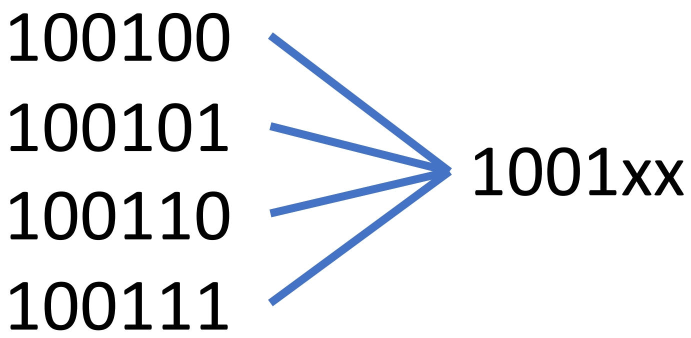
# Data structure

You can use

1. Struct
2. Class
3. Array
4. Vector (STL)
5. Map / Unorder map (STL)
6. Queue (STL)

# Data structure

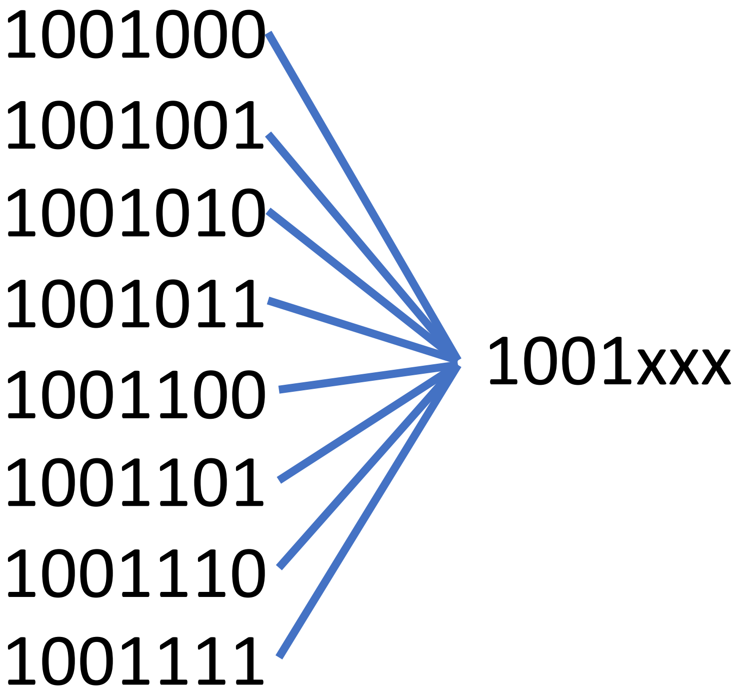
Block size : 4 bytes



100100
100101
100110
100111

1001xx

Block size : 8 bytes



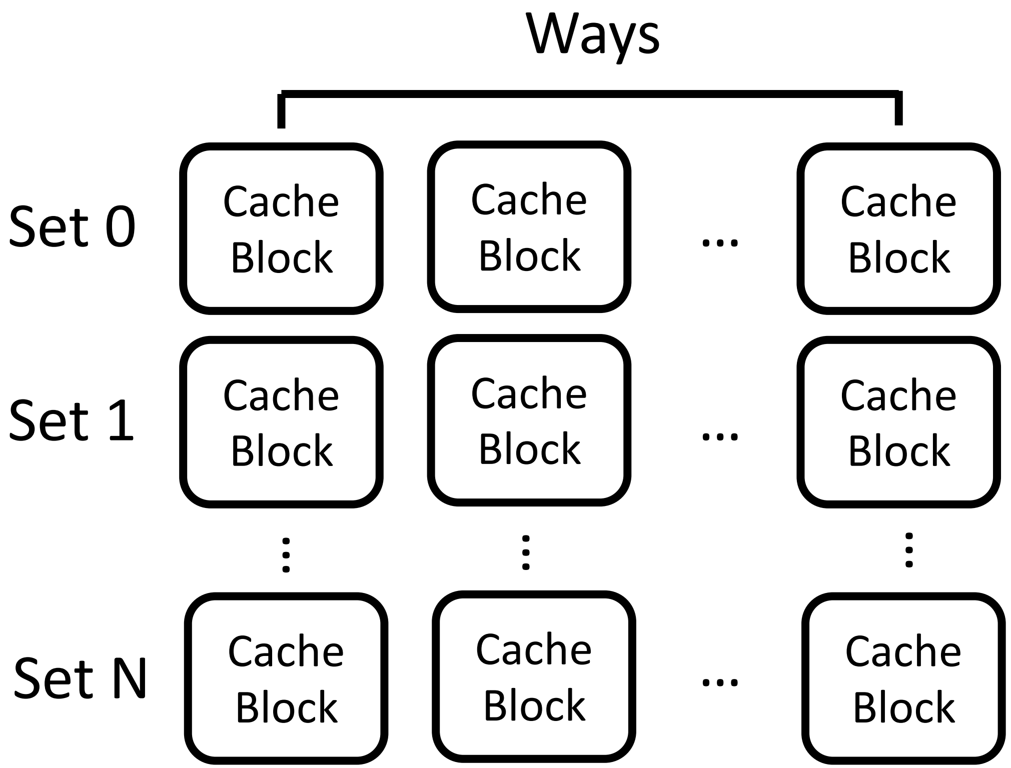
1001000
1001001
1001010
1001011
1001100
1001101
1001110
1001111

1001xxx

# Data structure

Associativity : Number of ways in a set

Cache_sets : Number of sets in a cache



Ways

|  |  |  |  |  |
|---|---|---|---|---|
| Set 0 | Cache Block | Cache Block | … | Cache Block |
| Set 1 | Cache Block | Cache Block | … | Cache Block |
|  | ⋮ | ⋮ |  | ⋮ |
| Set N | Cache Block | Cache Block | … | Cache Block |

# Outline

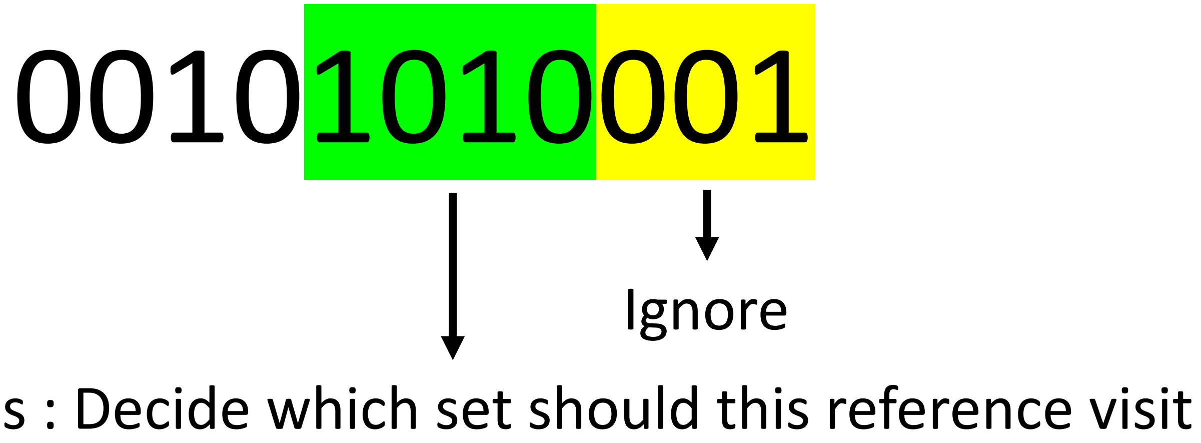
1. Pass data from command line
2. Read / Output file
3. Data structure of Cache
4. Simulation
5. Part 2

# Extract the reference

Block size : 8 bytes

Number of set : 16

Indexing policy : LSB

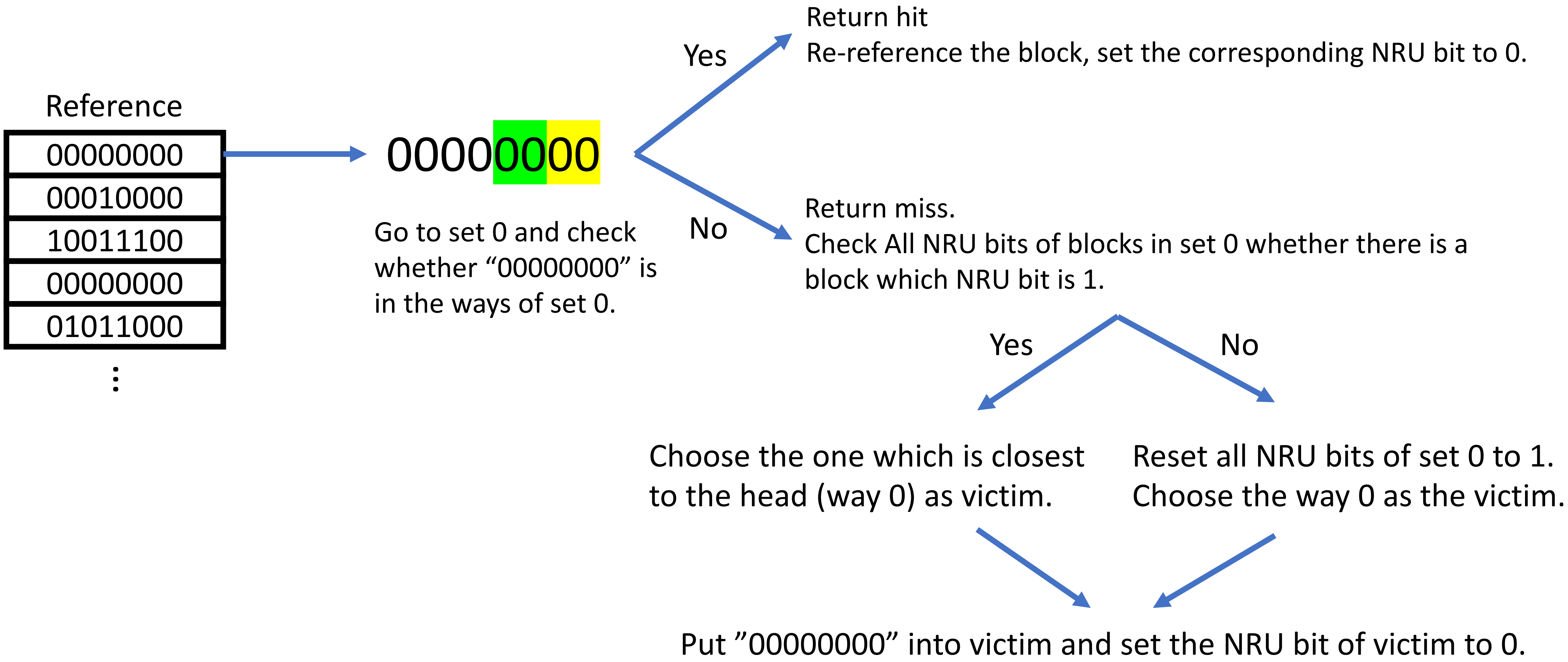


00101010001

Ignore

Indexing bits : Decide which set should this reference visit

# Read references



| Reference |
|---|
| 00000000 |
| 00010000 |
| 10011100 |
| 00000000 |
| 01011000 |
| ⋮ |

00000000

Go to set 0 and check whether “00000000” is in the ways of set 0.

Yes

Return hit

Re-reference the block, set the corresponding NRU bit to 0.

No

Return miss.

Check All NRU bits of blocks in set 0 whether there is a block which NRU bit is 1.

Yes

Choose the one which is closest to the head (way 0) as victim.

No

Reset all NRU bits of set 0 to 1.

Choose the way 0 as the victim.

Put ”00000000” into victim and set the NRU bit of victim to 0.

# Outline

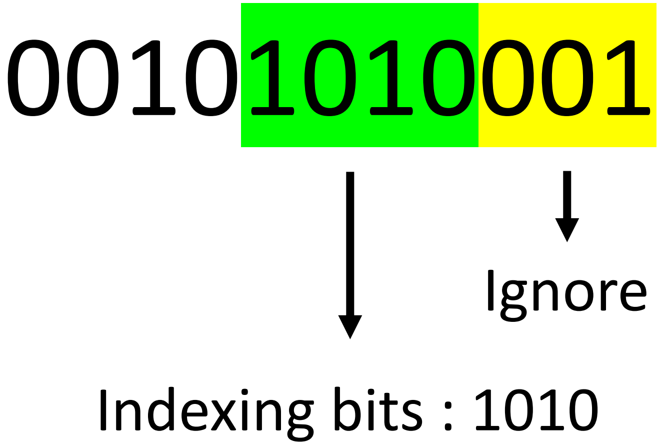
1. Pass data from command line
2. Read / Output file
3. Data structure of Cache
4. Simulation
5. Part 2

# Extract the reference

Block size : 8 bytes

Number of set : 16

Indexing policy : LSB



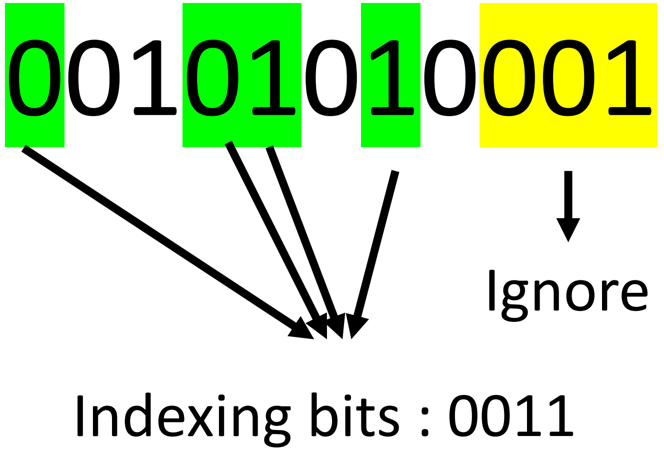
0010101001

Ignore

Indexing bits : 1010

Block size : 8 bytes

Number of set : 16



0010101001

Ignore

Indexing bits : 0011

# How to find proper indexing bits

1. Exhausted search? ✗
2. Read the reference paper from p10 to p13? ✓
3. Attend the 助教課 ✓

# The method of reference paper

Correlation of two bits : $C_{i,j} = \min(E_{i,j}, D_{i,j}) / \max(E_{i,j}, D_{i,j})$

$E_{i,j}$ : The number of references having the same values at address bits $a_i$ and $a_j$ .

$D_{i,j}$ : The number of references having different values at address bits $a_i$ and $a_j$ .

| $a_7$ | $a_6$ | $a_5$ | $a_4$ | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |

Offset bits

Example : $C_{2,3}$

$E_{2,3} = 4$

$D_{2,3} = 1$

$C_{2,3} = 1/4$

Example : $C_{5,7}$

$E_{5,7} = 4$

$D_{5,7} = 1$

$C_{5,7} = 1/4$

Correlation array

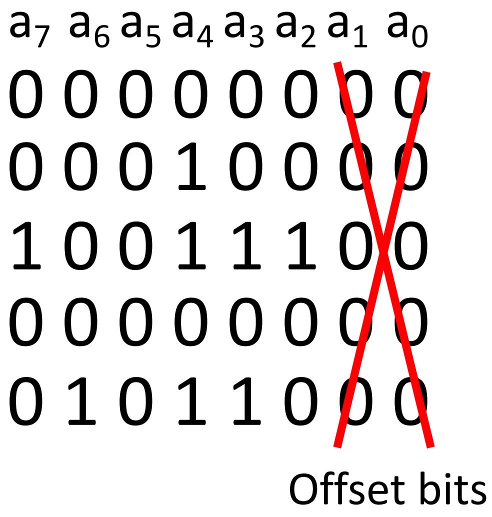
| | | | | | |
|---|---|---|---|---|---|
| $C_{2,2}$ | $C_{2,3}$ | $C_{2,4}$ | $C_{2,5}$ | $C_{2,6}$ | $C_{2,7}$ |
| $C_{3,2}$ | $C_{3,3}$ | $C_{3,4}$ | $C_{3,5}$ | $C_{3,6}$ | $C_{3,7}$ |
| $C_{4,2}$ | $C_{4,3}$ | $C_{4,4}$ | $C_{4,5}$ | $C_{4,6}$ | $C_{4,7}$ |
| $C_{5,2}$ | $C_{5,3}$ | $C_{5,4}$ | $C_{5,5}$ | $C_{5,6}$ | $C_{5,7}$ |
| $C_{6,2}$ | $C_{6,3}$ | $C_{6,4}$ | $C_{6,5}$ | $C_{6,6}$ | $C_{6,7}$ |
| $C_{7,2}$ | $C_{7,3}$ | $C_{7,4}$ | $C_{7,5}$ | $C_{7,6}$ | $C_{7,7}$ |

# The method of reference paper

Quality measurement : $Q_i = \min(Z_i , O_i) / \max(Z_i , O_i)$

$Z_i$ : The number of references having the value ”0” at address bit $a_i$ .

$O_i$ : The number of references having the value ”1” at address bit $a_i$ .



$$\begin{array}{cccccccc} a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$

Offset bits

Example : $Q_2$

$Z_2 = 4$

$O_2 = 1$

$Q_2 = 1/4$

Example : $Q_5$

$Z_5 = 5$

$O_5 = 0$

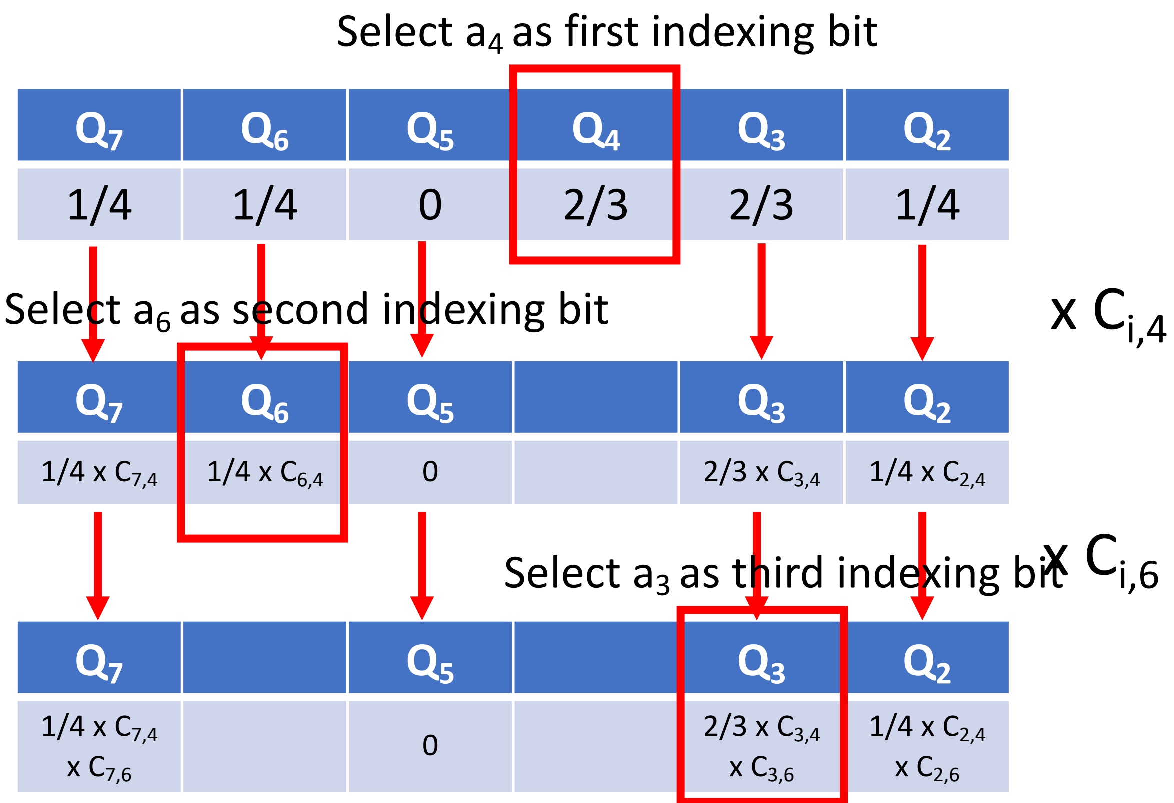
$Q_5 = 0$

| $Q_7$ | $Q_6$ | $Q_5$ | $Q_4$ | $Q_3$ | $Q_2$ |
|---|---|---|---|---|---|
| 1/4 | 1/4 | 0 | 2/3 | 2/3 | 1/4 |

# The method of reference paper

Number of indexing bits : 3

Select the current best quality as indexing bit.



Select $a_4$ as first indexing bit

| $Q_7$ | $Q_6$ | $Q_5$ | $Q_4$ | $Q_3$ | $Q_2$ |
|---|---|---|---|---|---|
| 1/4 | 1/4 | 0 | 2/3 | 2/3 | 1/4 |

Select $a_6$ as second indexing bit

x $C_{i,4}$

| $Q_7$ | $Q_6$ | $Q_5$ | | $Q_3$ | $Q_2$ |
|---|---|---|---|---|---|
| 1/4 x $C_{7,4}$ | 1/4 x $C_{6,4}$ | 0 | | 2/3 x $C_{3,4}$ | 1/4 x $C_{2,4}$ |

Select $a_3$ as third indexing bit

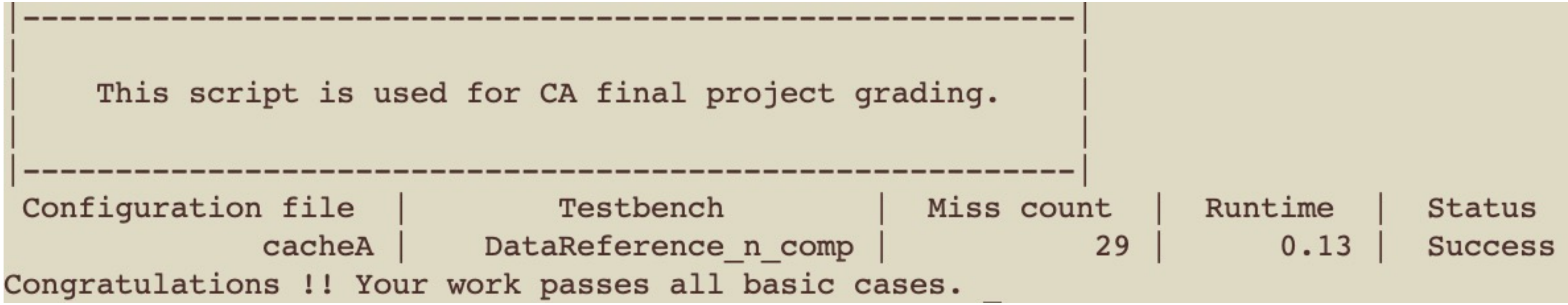
x $C_{i,6}$

| $Q_7$ | | $Q_5$ | | $Q_3$ | $Q_2$ |
|---|---|---|---|---|---|
| 1/4 x $C_{7,4}$<br>x $C_{7,6}$ | | 0 | | 2/3 x $C_{3,4}$<br>x $C_{3,6}$ | 1/4 x $C_{2,4}$<br>x $C_{2,6}$ |

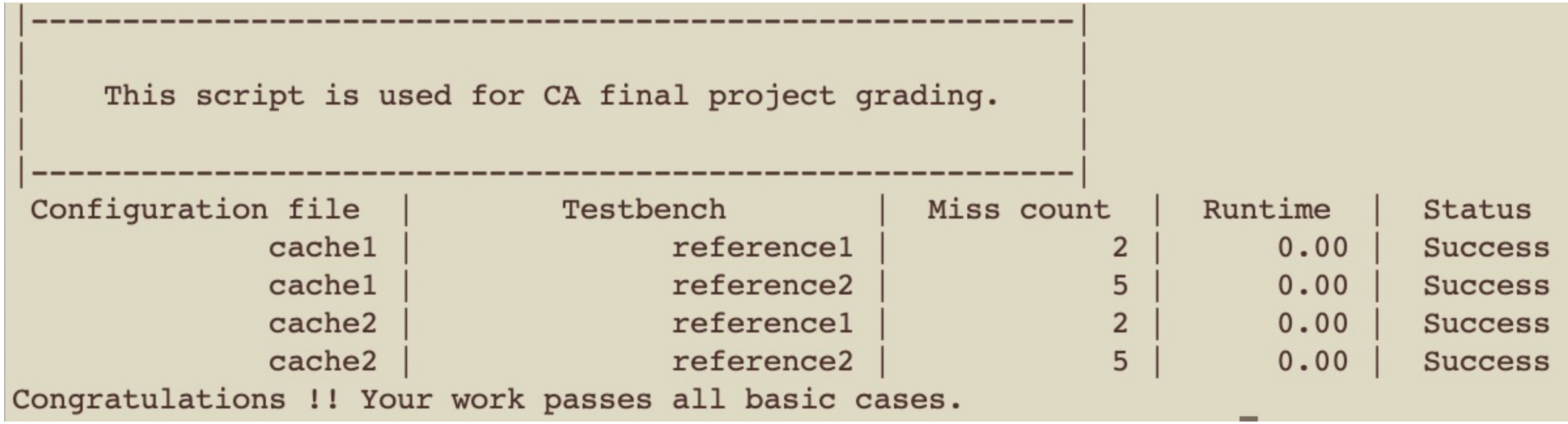
Correlation array

| | | | | | |
|---|---|---|---|---|---|
| $C_{2,2}$ | $C_{2,3}$ | $C_{2,4}$ | $C_{2,5}$ | $C_{2,6}$ | $C_{2,7}$ |
| $C_{3,2}$ | $C_{3,3}$ | $C_{3,4}$ | $C_{3,5}$ | $C_{3,6}$ | $C_{3,7}$ |
| $C_{4,2}$ | $C_{4,3}$ | $C_{4,4}$ | $C_{4,5}$ | $C_{4,6}$ | $C_{4,7}$ |
| $C_{5,2}$ | $C_{5,3}$ | $C_{5,4}$ | $C_{5,5}$ | $C_{5,6}$ | $C_{5,7}$ |
| $C_{6,2}$ | $C_{6,3}$ | $C_{6,4}$ | $C_{6,5}$ | $C_{6,6}$ | $C_{6,7}$ |
| $C_{7,2}$ | $C_{7,3}$ | $C_{7,4}$ | $C_{7,5}$ | $C_{7,6}$ | $C_{7,7}$ |

# TA’s result of released testcases



```
|----------------------------------------------------------|
|                                                          |
|    This script is used for CA final project grading.     |
|                                                          |
|----------------------------------------------------------|
 Configuration file |         Testbench         | Miss count |  Runtime |  Status
             cacheA |   DataReference_n_comp    |         29 |     0.13 |  Success
Congratulations !! Your work passes all basic cases. _
```



```
|----------------------------------------------------------|
|                                                          |
|    This script is used for CA final project grading.     |
|                                                          |
|----------------------------------------------------------|
 Configuration file |          Testbench         | Miss count |  Runtime |  Status
             cache1 |                 reference1 |          2 |     0.00 |  Success
             cache1 |                 reference2 |          5 |     0.00 |  Success
             cache2 |                 reference1 |          2 |     0.00 |  Success
             cache2 |                 reference2 |          5 |     0.00 |  Success
Congratulations !! Your work passes all basic cases.
```