Cache Replacement Policies and Signature-based Hit Prediction (SHiP) Cache

Memory Systems HW1

Outline

- Cache replacement policies
- Simulator introduction
- NTHU Online Judge introduction

Replacement Policy

- Each set of an N-way cache has N blocks
- Upon a miss, the cache can consider giving up one cached block to store the requested block
- A selection policy is needed
  - The selection affects performance (not correctness)
- Some cache replacement policies
  1. LRU (least recently used)
  2. NRU (not recently used)
  3. SRRIP (static re-reference interval prediction)
  4. SHiP+SRRIP

1. LRU

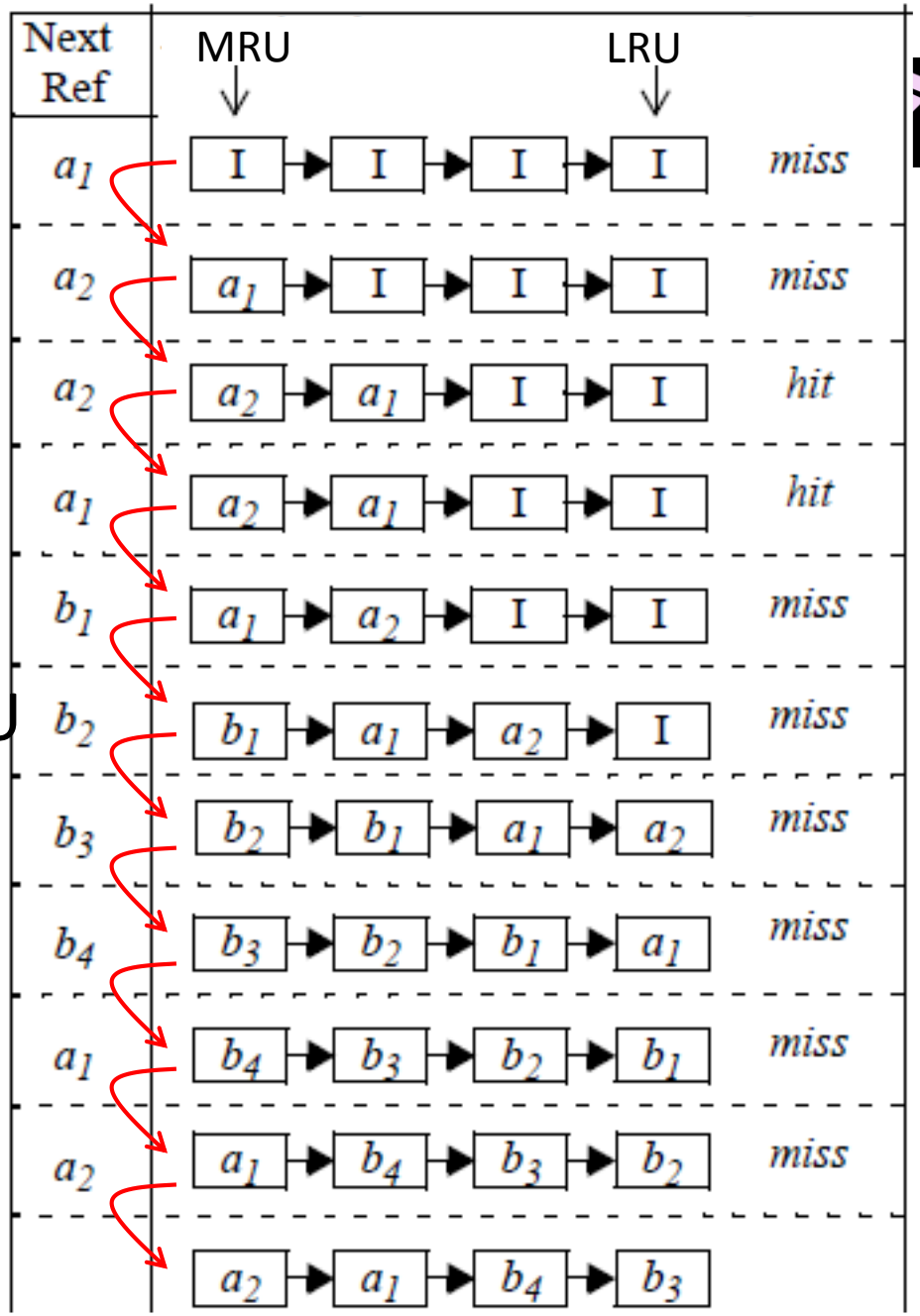
Cache Hit:

(i) move the block to MRU

Cache Miss:

(i) replace the LRU block

(ii) move the block to MRU



2. NRU

Cache Hit:

(i) set the nru-bit of the block to ‘0’

Cache Miss:

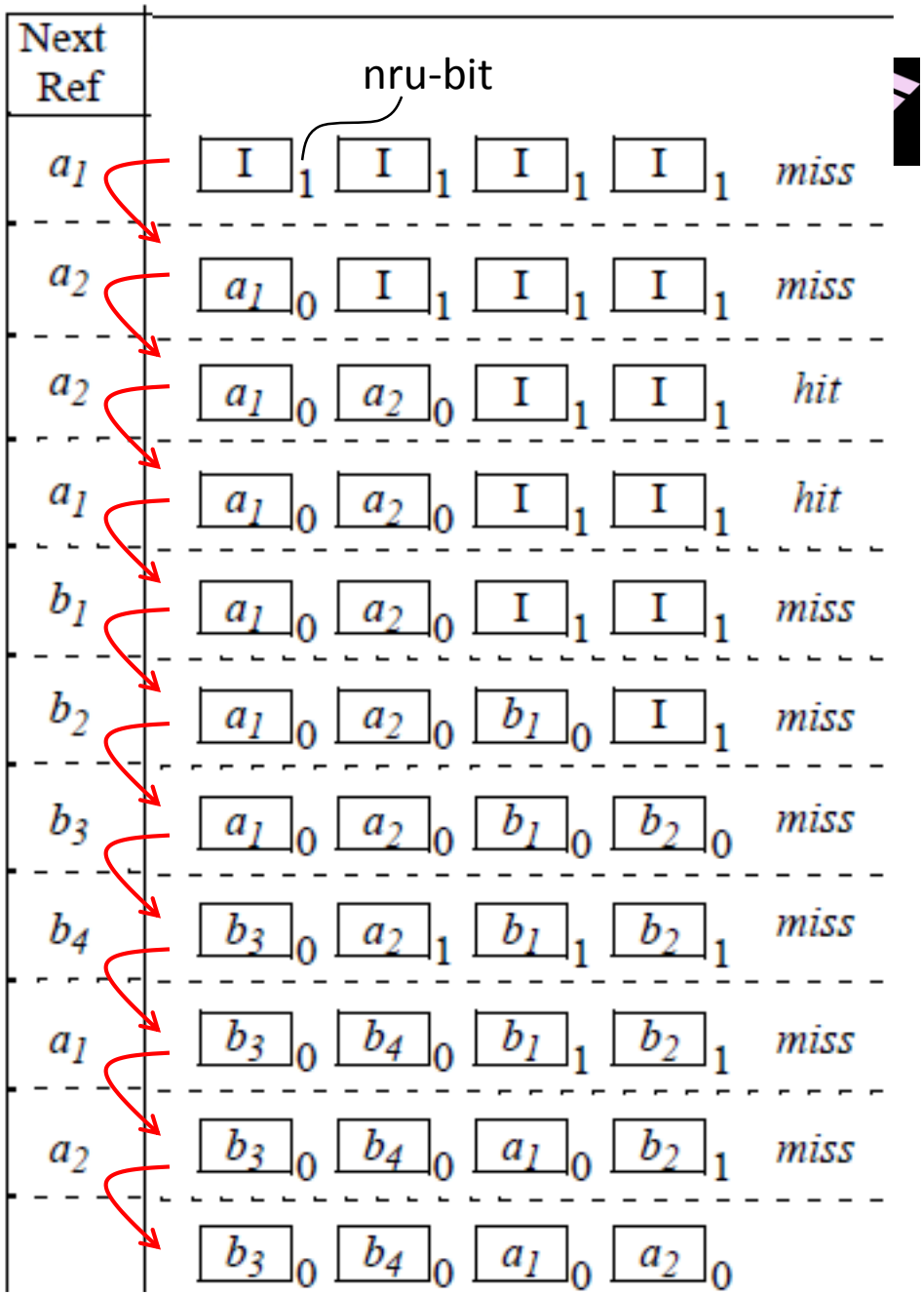
(i) search for the first ‘1’ from the left

(ii) if ‘1’ is found go to step (v)

(iii) set all nru-bits to ‘1’

(iv) goto step (i)

(v) replace the block and set the nru-bit to ‘1’



3. SRRIP

Cache Hit:

(i) set the RRPV of the block to ‘0’

Cache Miss:

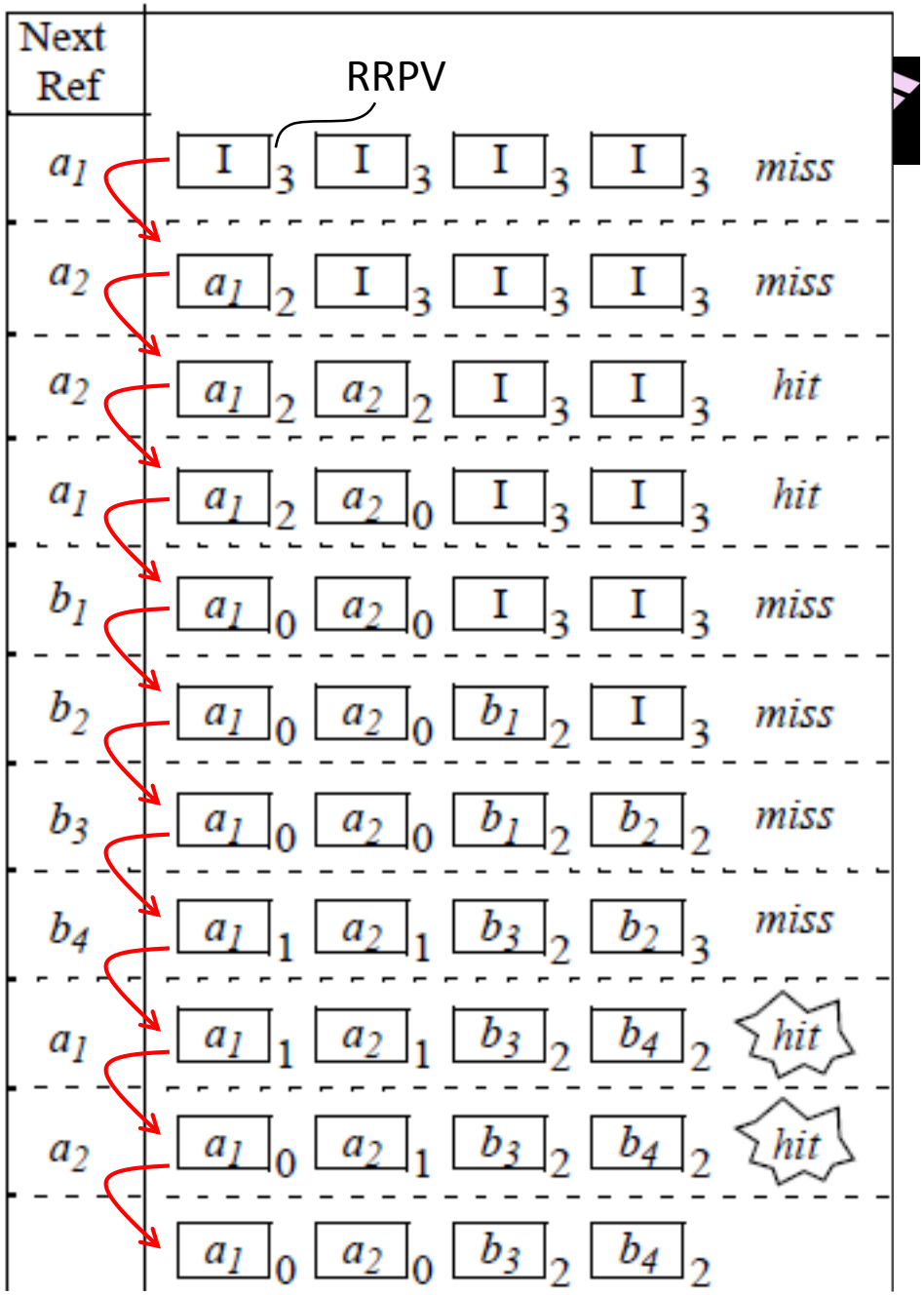
(i) search for the first ‘3’ from the left

(ii) if ‘3’ is found go to step (v)

(iii) increment all RRPVs

(iv) goto step (i)

(v) replace the block and set the RRPV to ‘2’



LRU (least recently used)

Next Ref	RRIP head → RRIP tail	
a_1	I → I → I → I	miss
a_2	a_1 → I → I → I	miss
a_2	a_2 → a_1 → I → I	hit
a_1	a_2 → a_1 → I → I	hit
b_1	a_1 → a_2 → I → I	miss
b_2	b_1 → a_1 → a_2 → I	miss
b_3	b_2 → b_1 → a_1 → a_2	miss
b_4	b_3 → b_2 → b_1 → a_1	miss
a_1	b_4 → b_3 → b_2 → b_1	miss
a_2	a_1 → b_4 → b_3 → b_2	miss
	a_2 → a_1 → b_4 → b_3	

Cache Hit:
(i) move block to MRU
Cache Miss:
(i) replace LRU block
(ii) move block to MRU

NRU (not recently used)

Next Ref					
a_1	I$_1$	I$_1$	I$_1$	I$_1$	miss
a_2	$a_1$$_0$	I$_1$	I$_1$	I$_1$	miss
a_2	$a_1$$_0$	$a_2$$_0$	I$_1$	I$_1$	hit
a_1	$a_1$$_0$	$a_2$$_0$	I$_1$	I$_1$	hit
b_1	$a_1$$_0$	$a_2$$_0$	I$_1$	I$_1$	miss
b_2	$a_1$$_0$	$a_2$$_0$	$b_1$$_0$	I$_1$	miss
b_3	$a_1$$_0$	$a_2$$_0$	$b_1$$_0$	$b_2$$_0$	miss
b_4	$b_3$$_0$	$a_2$$_1$	$b_1$$_1$	$b_2$$_1$	miss
a_1	$b_3$$_0$	$b_4$$_0$	$b_1$$_1$	$b_2$$_1$	miss
a_2	$b_3$$_0$	$b_4$$_0$	$a_1$$_0$	$b_2$$_1$	miss
	$b_3$$_0$	$b_4$$_0$	$a_1$$_0$	$a_2$$_0$	

Cache Hit:
(i) set nru-bit of block to '0'
Cache Miss:
(i) search for first '1' from the left
(ii) if '1' is found go to step (v)
(iii) set all nru-bits to '1'
(iv) goto step (i)
(v) replace block and set nru-bit to '1'

RRIP

Next Ref					
a_1	I$_3$	I$_3$	I$_3$	I$_3$	miss
a_2	$a_1$$_2$	I$_3$	I$_3$	I$_3$	miss
a_2	$a_1$$_2$	$a_2$$_2$	I$_3$	I$_3$	hit
a_1	$a_1$$_2$	$a_2$$_0$	I$_3$	I$_3$	hit
b_1	$a_1$$_0$	$a_2$$_0$	I$_3$	I$_3$	miss
b_2	$a_1$$_0$	$a_2$$_0$	$b_1$$_2$	I$_3$	miss
b_3	$a_1$$_0$	$a_2$$_0$	$b_1$$_2$	$b_2$$_2$	miss
b_4	$a_1$$_1$	$a_2$$_1$	$b_3$$_2$	$b_2$$_3$	miss
a_1	$a_1$$_1$	$a_2$$_1$	$b_3$$_2$	$b_4$$_2$	hit
a_2	$a_1$$_0$	$a_2$$_1$	$b_3$$_2$	$b_4$$_2$	hit
	$a_1$$_0$	$a_2$$_0$	$b_3$$_2$	$b_4$$_2$	

Cache Hit:
(i) set RRPV of block to '0'
Cache Miss:
(i) search for first '3' from the left
(ii) if '3' is found go to step (v)
(iii) increment all RRPVs
(iv) goto step (i)
(v) replace block and set RRPV to '2'

4. SHiP (Signature-based Hit Predictio

- SHiP can improve over SRRIP

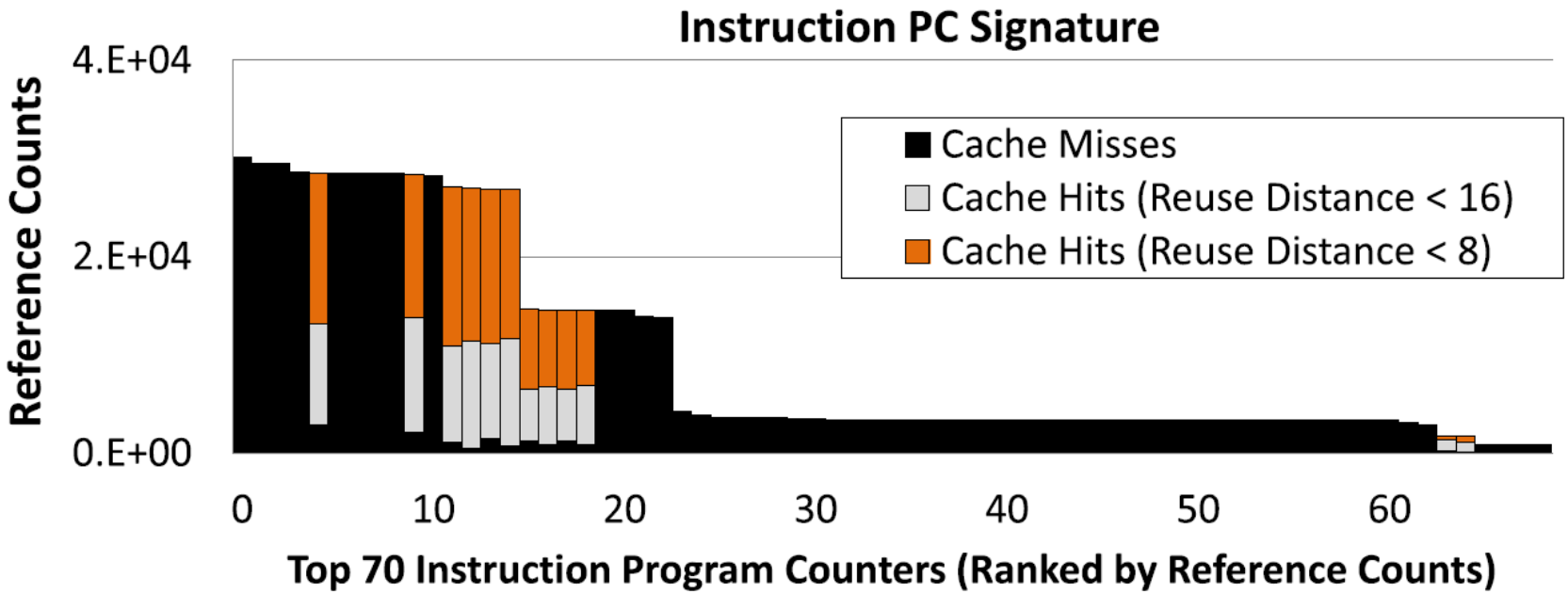
SRRIP	**SRRIP + SHiP**
Cache Hit: (i) set RRPV of block to '0' **Cache Miss**: (i) search for first '3' from left (ii) if '3' found go to step (v) (iii) increment all RRPVs (iv) goto step (i) (v) replace block and set RRPV to '**2**'	**Cache Hit**: (i) set RRPV of block to '0' **Cache Miss**: (i) search for first '3' from left (ii) if '3' found go to step (v) (iii) increment all RRPVs (iv) goto step (i) (v) replace block and set RRPV according to SHiP

SHiP (Signature-based Hit Predict

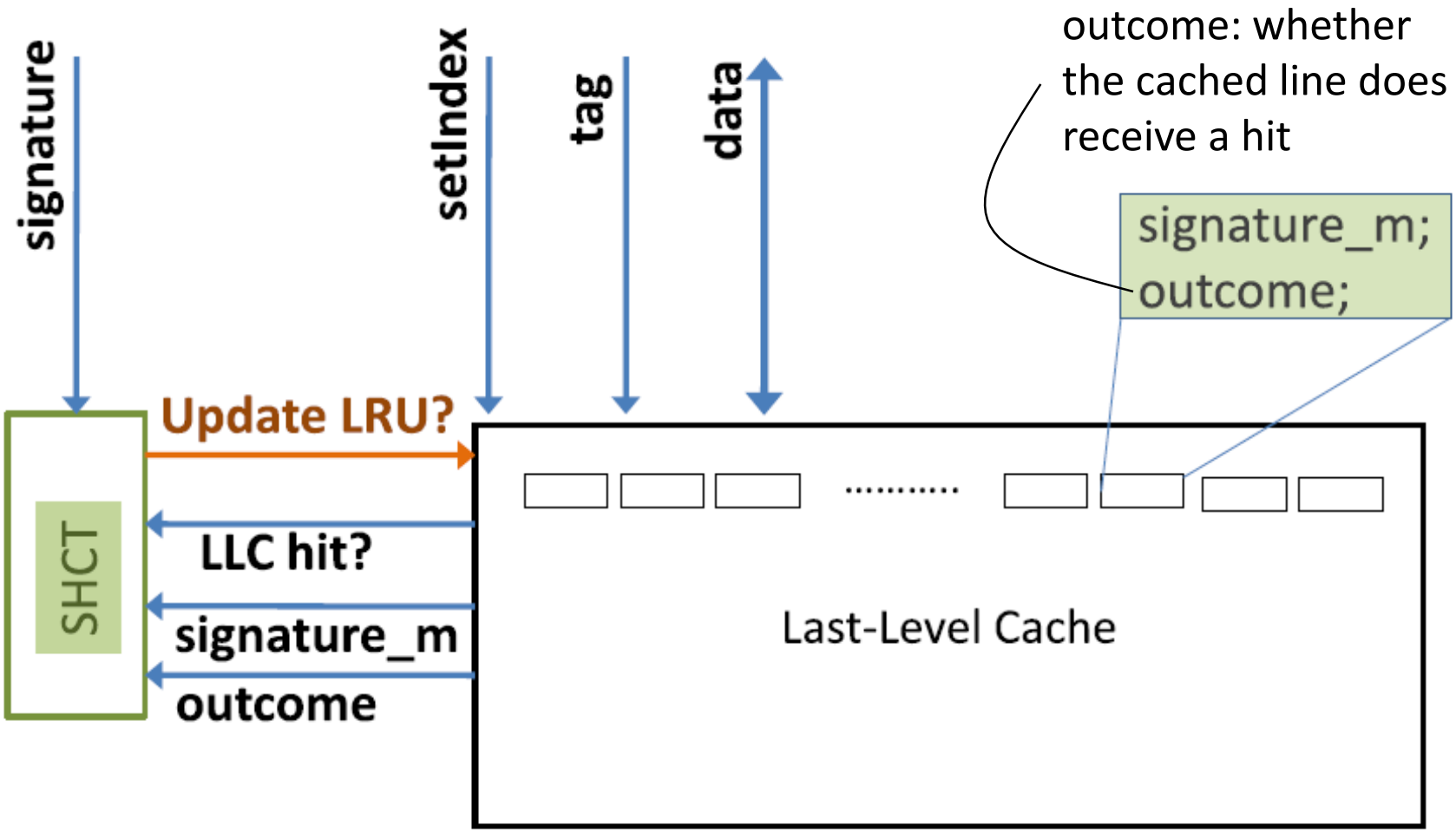
- Some blocks have a very long reuse distance
  - Not very worth being cached
- RRPV of this kind of blocks would be better set to '3' instead of '2'
- This kind of blocks can be identified using signature
  - Memory address
  - Program counter of the load/store instruction
  - etc.

SRRIP + SHiP
Cache Hit: (i) set RRPV of block to ‘0’ **Cache Miss**: (i) search for first ‘3’ from left (ii) if ‘3’ found go to step (v) (iii) increment all RRPVs (iv) goto step (i) (v) replace block and set RRPV according to SHiP

Program Counter Signature



SHiP Architecture



SHiP-PC + SRRIP

```
Cache Hit:
    cache_line.outcome = true;
    Increment SHCT[signature_line];
    RRPV = 0; // same as RRIP

Cache Miss:
    find an old line to evict
    if (evicted_cache_line.outcome == false)
        Decrement SHCT[signature_line];
    insert the new line
    cache_line.outcome = false;
    cache_line.signature_line = hash(PC);
    if (SHCT[hash(PC)] == 0)
        RRPV = 3; // not worth being cached
    else
        RRPV = 2; // same as RRIP
```

Simulator

- SHiP+SRRIP looks good
- We need to quantify the improvement of SHiP+SRRIP over baseline design such as LRU
- Thus, we need simulator to answer the question

Simulator

- Use a Linux workstation (e.g., ws31 of NTHUEE) or use a Linux virtual machine
- Download the simulator kit from the course website
- Unpack the tarball

```
tar -xzvf HW2.tgz
cd CRC
```

- Set the variable 'PIN_DIR' in pinkit/CONFIG/makefile.gnu.config to the absolute path for pinkit/pin-2.7-31933-gcc.3.4.6-ia32_intel64-linux/ in the kit. When you edit the makefile.gnu.config file, modify the fourth line to:

```
PIN_DIR ?= $FULL_PATH_TO_CRC_KIT/CRC/pinkit/pin-2.7-31933-gcc.3.4.6-ia32_intel64-linux
```

ref: http://www.jilp.org/jwac-1/replacement_framework_description.html

Simulator

- To compile the simulator, make sure you are in the CRC/ directory and that your gcc compiler path is set up correctly, then type (depending on whether your machine is 32 or 64 bits):
  make CMPsim32
  or
  make CMPsim64
  - type "uname -a" in Linux to determine whether your machine is 32-bit (IA-32, i686) or 64-bit (x86_64)
- Traces are available in the CRC/traces director

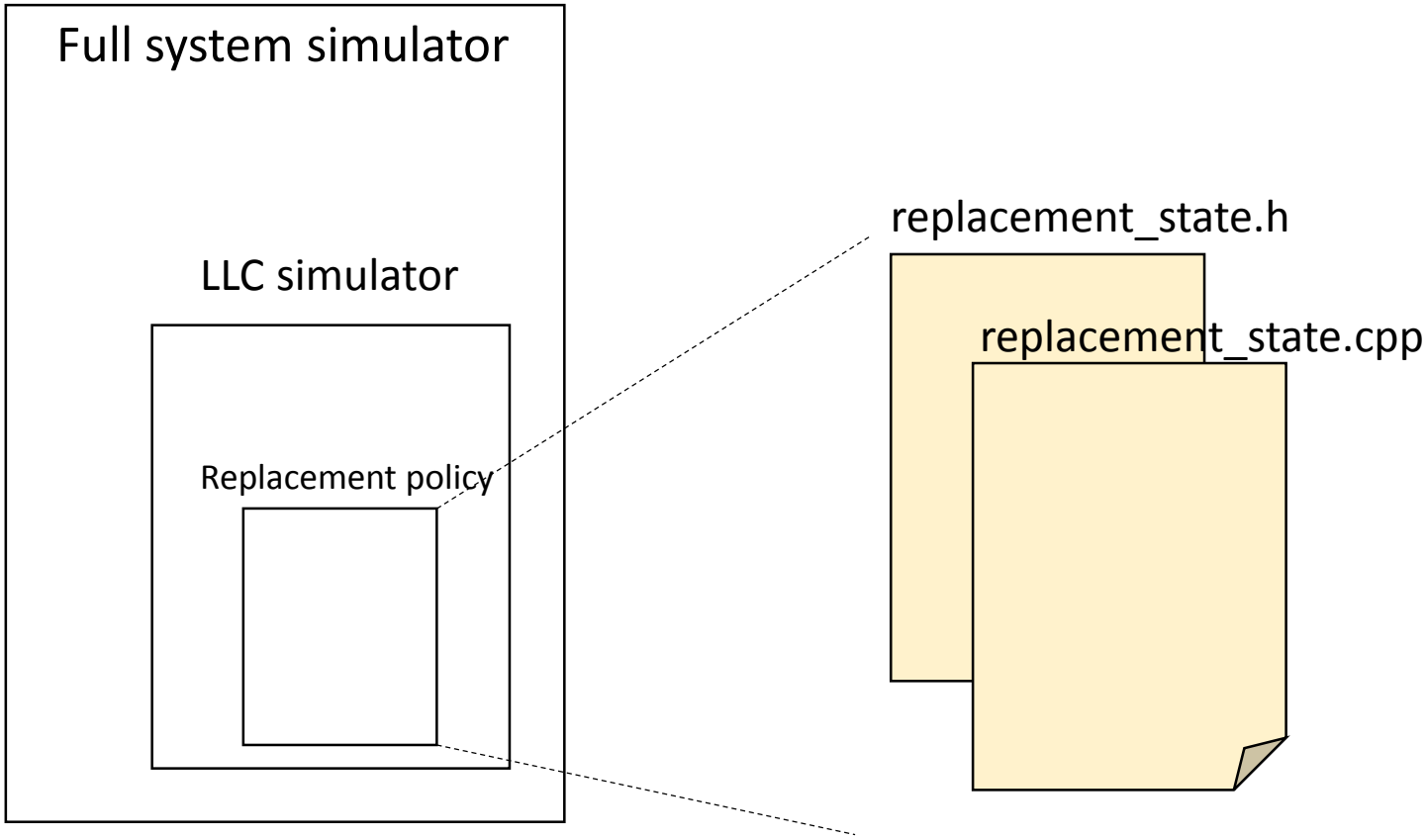
Run Simulation

- Edit the full path information of trace1 to trace4 in the CRC/traces/ directory
  - FULL_PATH/CRC/traces/art.out.trace.gz

- Edit the 8th line of runall.sh in the CRC/runs/ directory
  - Change "CMPsim.usetrace.64" to "CMPsim.usetrace.32" if your machine is 32-bit

Modify Simulator

- You are allowed to modified only two files
  - CRC/src/LLCsim/replacement_state.h
  - CRC/src/LLCsim/replacement_state.cpp

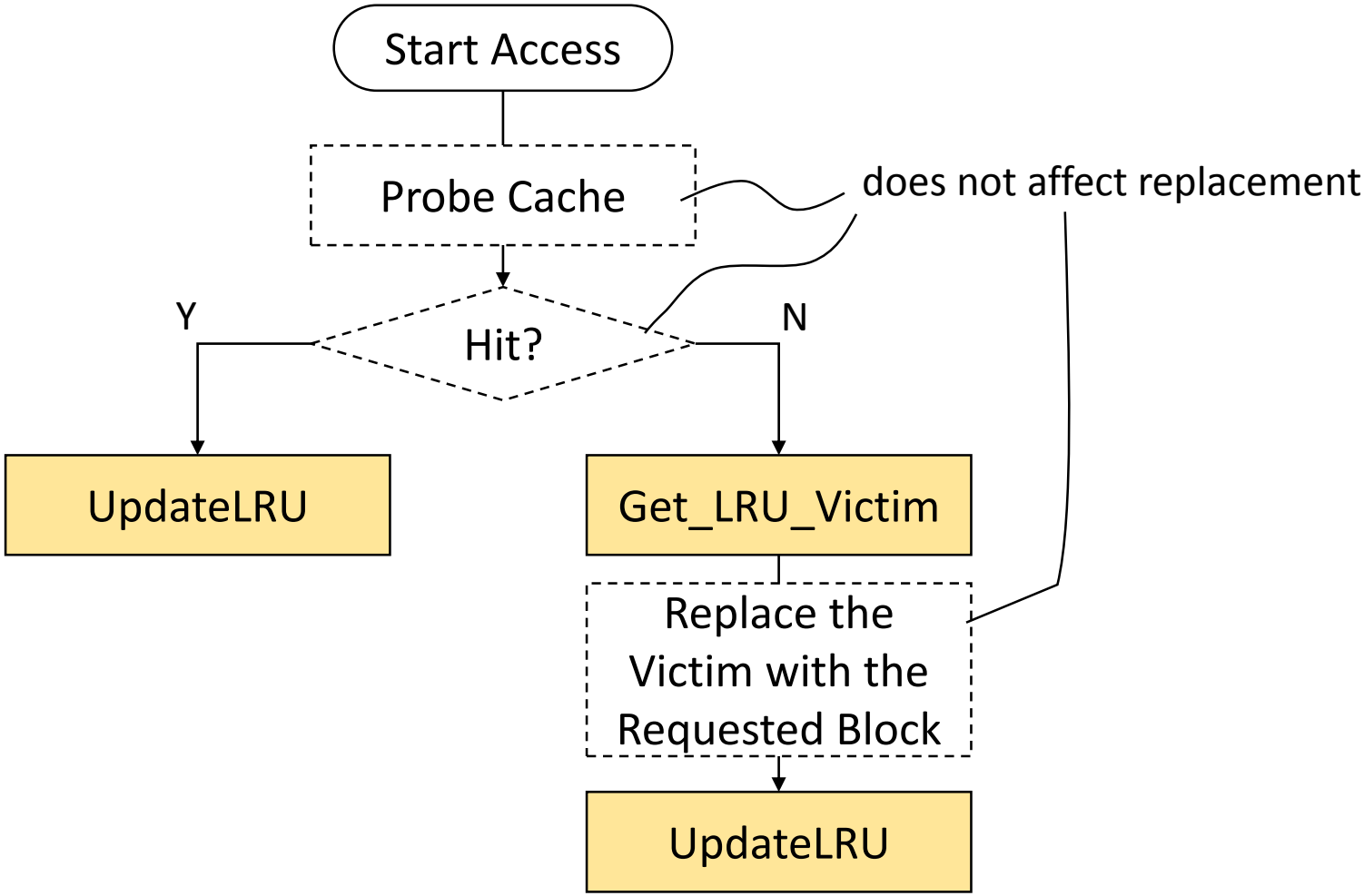


replacement_state.h

```
class CACHE_REPLACEMENT_STATE
{
  private:
    UINT32 numsets;
    UINT32 assoc;
    UINT32 replPolicy;
    LINE_REPLACEMENT_STATE   **repl;
    COUNTER mytimer;  // tracks # of references to the cache
    // Add extra state for cache here

  public:

    CACHE_REPLACEMENT_STATE( UINT32 _sets, UINT32 _assoc, UINT32 _pol );
    INT32  GetVictimInSet( … );
    void   UpdateReplacementState( UINT32 setIndex, INT32 updateWayID );
    void   SetReplacementPolicy( UINT32 _pol ) { replPolicy = _pol; }
    void   IncrementTimer() { mytimer++; }
    void   UpdateReplacementState( … );
    ostream&   PrintStats( ostream &out);

  private:

    void   InitReplacementState();
    INT32  Get_Random_Victim( UINT32 setIndex );
    INT32  Get_LRU_Victim( UINT32 setIndex );
    void   UpdateLRU( UINT32 setIndex, INT32 updateWayID );
};
```

```
typedef struct
{
    UINT32  LRUstackposition;
    // Add extra "per-line" state here

} LINE_REPLACEMENT_STATE;
```

LRU Example


Start Access
Probe Cache
does not affect replacement
Hit?
Y
N
UpdateLRU
Get_LRU_Victim
Replace the Victim with the Requested Block
UpdateLRU

replacement_state.cpp

```
CACHE_REPLACEMENT_STATE::CACHE_REPLACEMENT_STATE( UINT32 _sets,
UINT32 _assoc, UINT32 _pol )
{

    numsets    = _sets;
    assoc      = _assoc;
    replPolicy = _pol;

    mytimer    = 0;

    InitReplacementState();
}
```

replacement_state.cpp

```
void CACHE_REPLACEMENT_STATE::InitReplacementState()
{
    // Create the state for sets, then create the state for the ways
    repl  = new LINE_REPLACEMENT_STATE* [ numsets ];

    // ensure that we were able to create replacement state
    assert(repl);

    // Create the state for the sets
    for(UINT32 setIndex=0; setIndex<numsets; setIndex++)
    {
        repl[ setIndex ]  = new LINE_REPLACEMENT_STATE[ assoc ];

        for(UINT32 way=0; way<assoc; way++)
        {
            // initialize stack position (for true LRU)
            repl[ setIndex ][ way ].LRUstackposition = way;
        }
    }

    // Contestants:  ADD INITIALIZATION FOR YOUR HARDWARE HERE

}
```

replacement_state.cpp

```
INT32 CACHE_REPLACEMENT_STATE::GetVictimInSet( UINT32 tid, UINT32 setIndex, const
LINE_STATE *vicSet, UINT32 assoc, Addr_t PC, Addr_t paddr, UINT32 accessType )
{
    // If no invalid lines, then replace based on replacement policy
    if( replPolicy == CRC_REPL_LRU )
    {
        return Get_LRU_Victim( setIndex );
    }
    else if( replPolicy == CRC_REPL_RANDOM )
    {
        return Get_Random_Victim( setIndex );
    }
    else if( replPolicy == CRC_REPL_CONTESTANT )
    {
        // ADD YOUR VICTIM SELECTION FUNCTION HERE
    }

    // We should never get here
    assert(0);

    return -1; // Returning -1 bypasses the LLC
}
```

replacement_state.cpp

```
INT32 CACHE_REPLACEMENT_STATE::Get_LRU_Victim( UINT32 setIndex )
{
    // Get pointer to replacement state of current set
    LINE_REPLACEMENT_STATE *replSet = repl[ setIndex ];

    INT32   lruWay   = 0;

    // Search for victim whose stack position is assoc-1
    for(UINT32 way=0; way<assoc; way++)
    {
        if( replSet[way].LRUstackposition == (assoc-1) )
        {
            lruWay = way;
            break;
        }
    }

    // return lru way
    return lruWay;
}
```

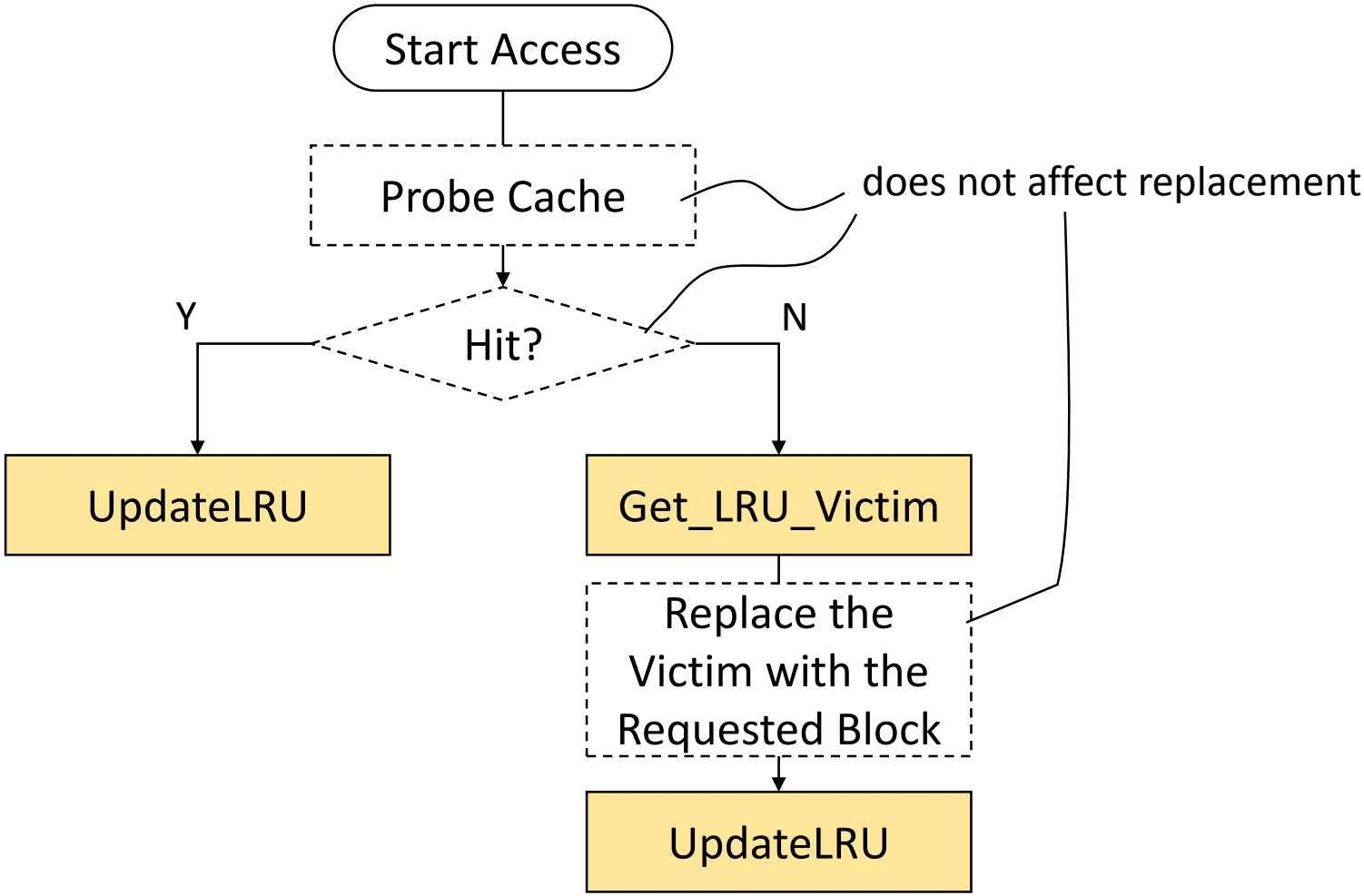
replacement_state.cpp

```
void CACHE_REPLACEMENT_STATE::UpdateReplacementState(
    UINT32 setIndex, INT32 updateWayID, const LINE_STATE *currLine,
    UINT32 tid, Addr_t PC, UINT32 accessType, bool cacheHit )
{

    if( replPolicy == CRC_REPL_LRU )
    {
        UpdateLRU( setIndex, updateWayID );
    }
    else if( replPolicy == CRC_REPL_RANDOM )
    {
        // Random replacement requires no replacement state update
    }
    else if( replPolicy == CRC_REPL_CONTESTANT )
    {
        // ADD YOUR UPDATE REPLACEMENT STATE FUNCTION HERE
        // Feel free to use any of the input parameters to make
        // updates to your replacement policy
    }


}
```

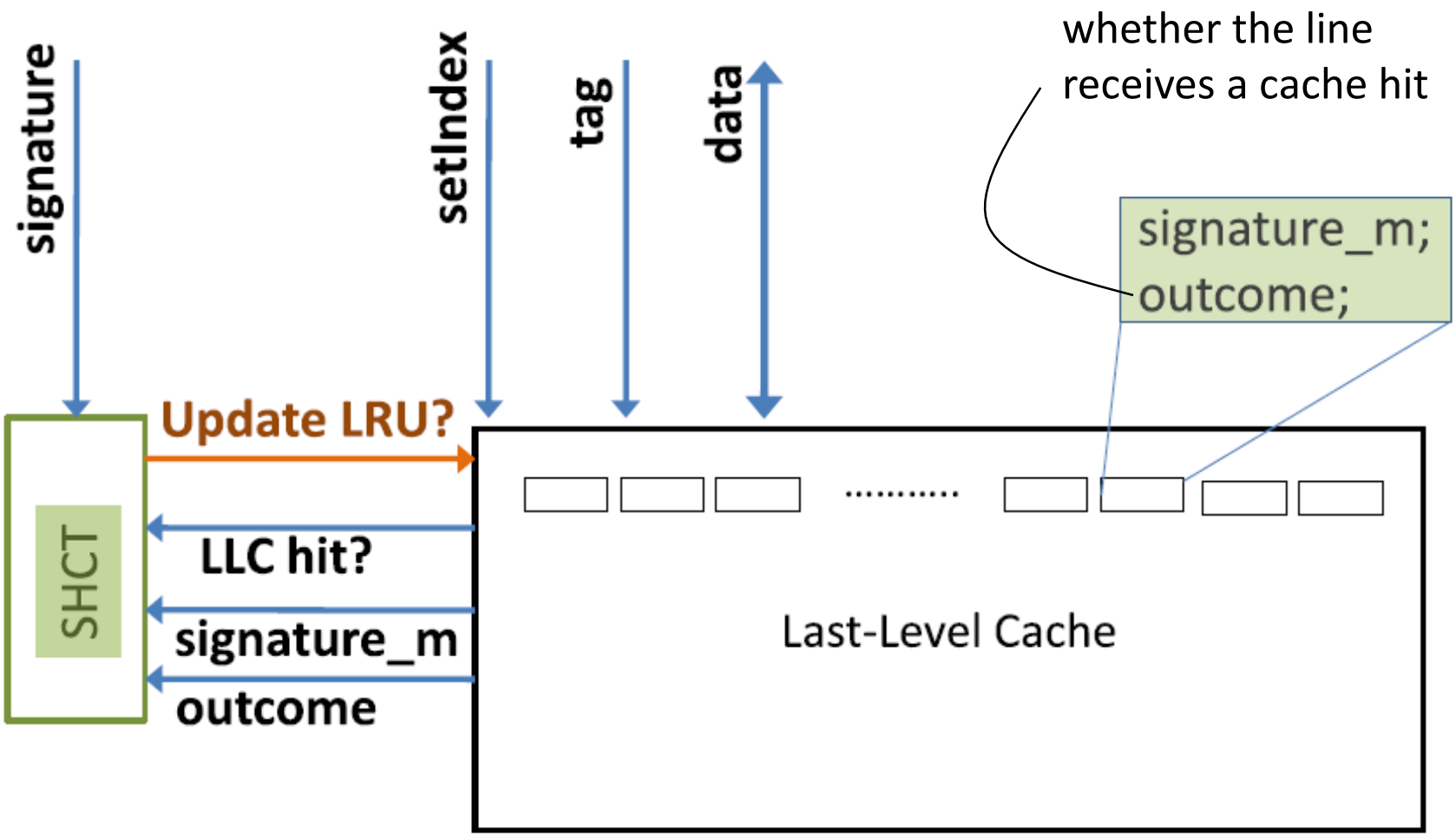
replacement_state.cpp

```
void CACHE_REPLACEMENT_STATE::UpdateLRU( UINT32 setIndex, INT32 updateWayID )
{
    // Determine current LRU stack position
    UINT32 currLRUstackposition = repl[ setIndex ][ updateWayID ].LRUstackposition;

    // Update the stack position of all lines before the current line
    // Update implies incremeting their stack positions by one
    for(UINT32 way=0; way<assoc; way++)
    {
        if( repl[setIndex][way].LRUstackposition < currLRUstackposition )
        {
            repl[setIndex][way].LRUstackposition++;
        }
    }

    // Set the LRU stack position of new line to be zero
    repl[ setIndex ][ updateWayID ].LRUstackposition = 0;
}
```

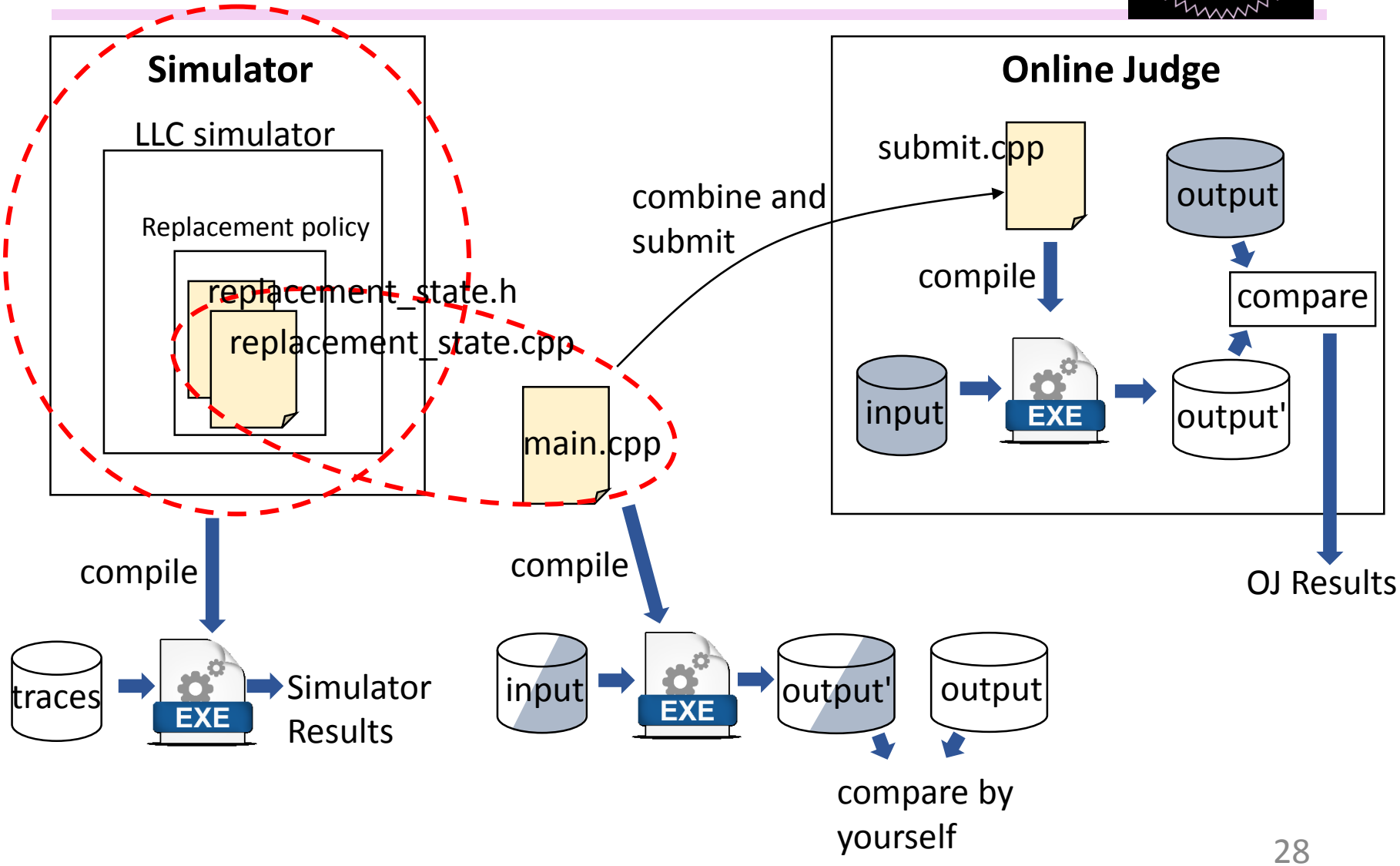
LRU Example


Start Access
Probe Cache
does not affect replacement
Hit?
Y
N
UpdateLRU
Get_LRU_Victim
Replace the Victim with the Requested Block
UpdateLRU

Implement SHiP



NTHU Online Judge



Input

number of sets, **S** — 1

number of ways per set, **W** — 4

replacement policy — 2

number of SHCT entries — 10

number of following operations — 4

0	100061	0	1	3
1	100022	0	1	3
2	100090	0	0	
3	100060	0	0	

serial number

program counter

set (0~S-1)

hit?

hit way (0~W-1)

Output

number of sets
number of ways per set
replacement policy
number of SHCT entries
number of following operations
serial number, program counter, hit/miss, hit way

```
1
4
2
10
4
0 100061 0 Hit 3
RRPV: 3 3 3 (0)
Signature: 0 0 0 0
outcome: 0 0 0 (1)
SHCT: (2) 1 1 1 1 1 1 1 1 1 ↵
↵
1 100022 0 Hit 3
RRPV: 3 3 3 (0)
Signature: 0 0 0 0
outcome: 0 0 0 (1)
SHCT: (3) 1 1 1 1 1 1 1 1 1

2 100090 0 Replace 0
RRPV: (2) 3 3 0
Signature: (0) 0 0 0
outcome: (0) 0 0 1
SHCT: (2) 1 1 1 1 1 1 1 1 1 ↵
```

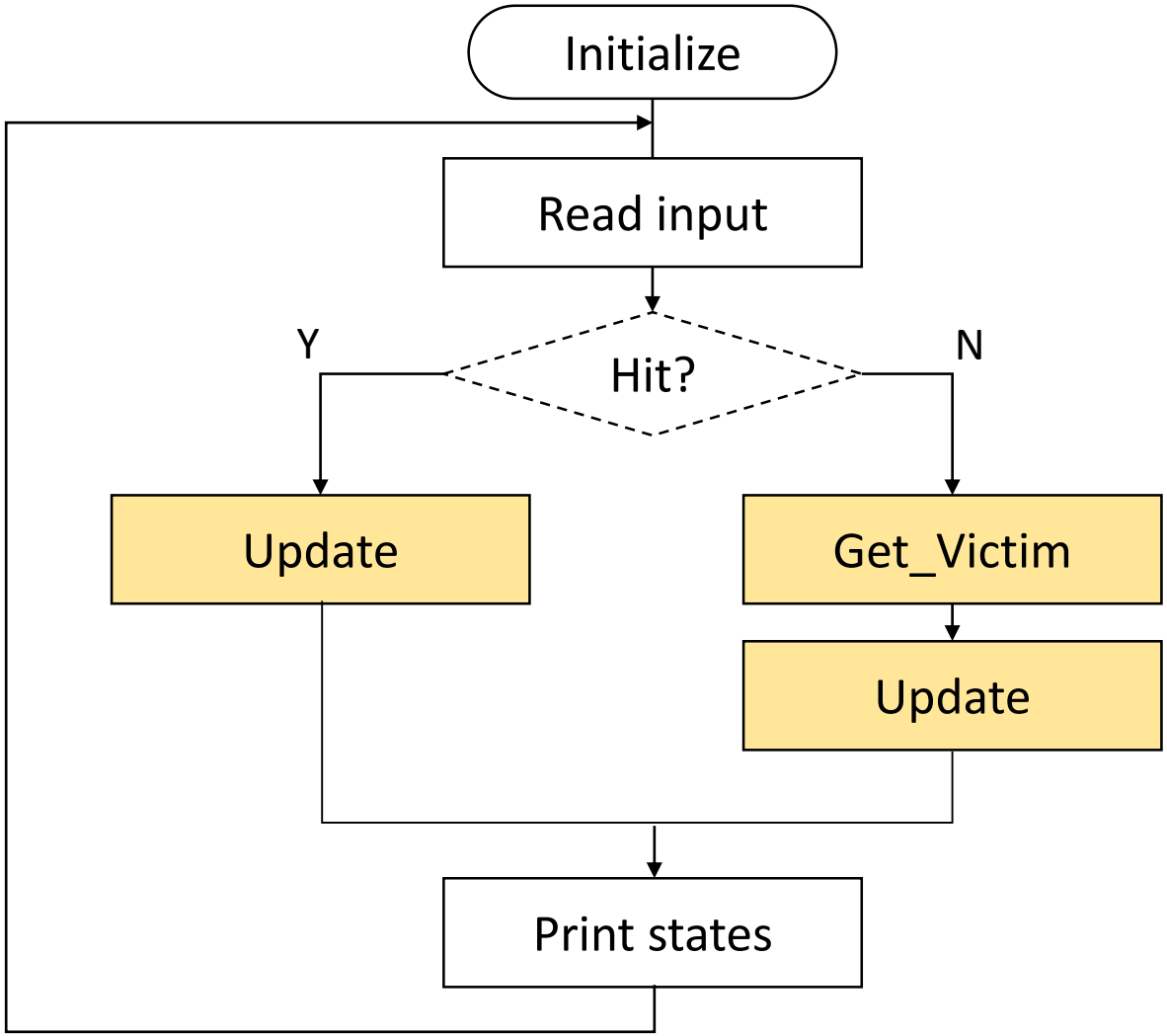
detailed status of the set

detailed status of the SHCT table

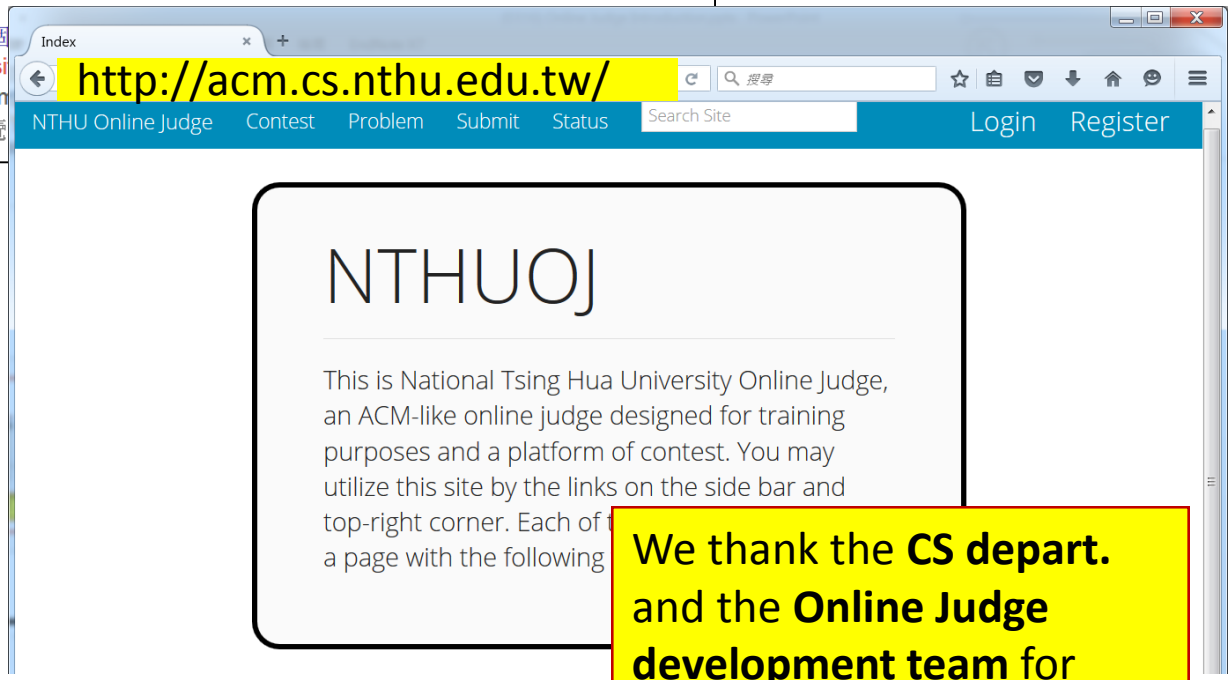
parenthesis indicates touched value (not necessary changed)

Replaced way is shown for a miss

main.cpp


Initialize
Read input
Hit?
Y
N
Update
Get_Victim
Update
Print states

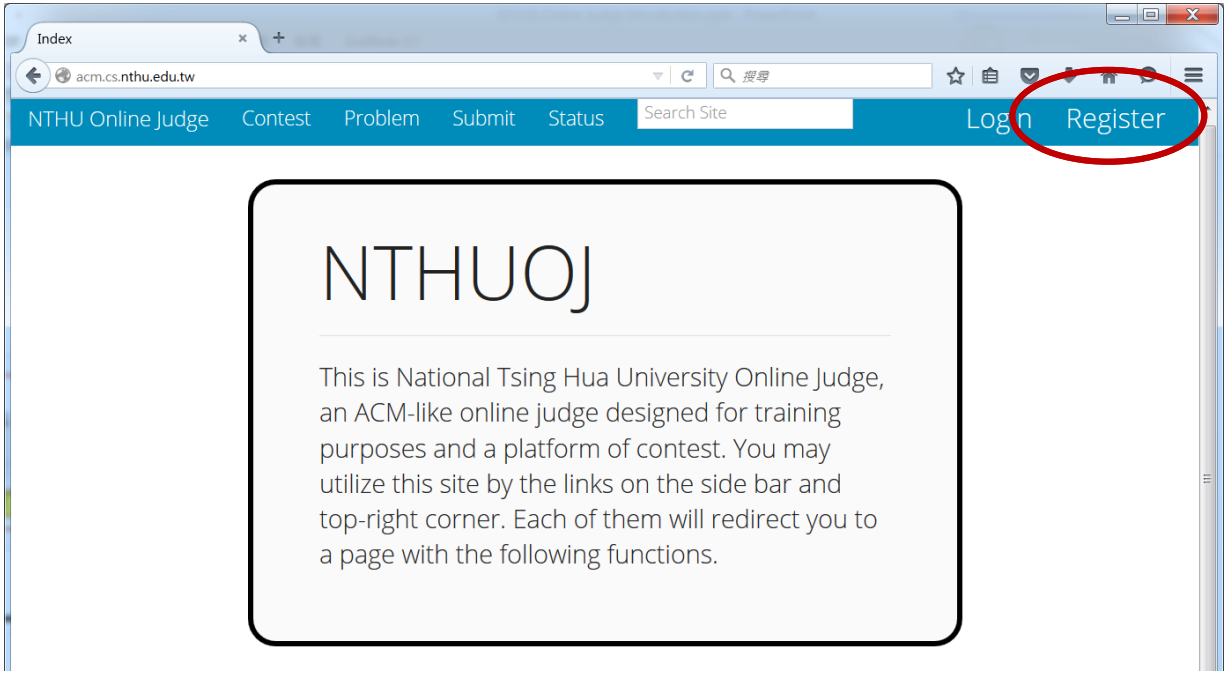
NTHU Online Judge



We thank the **CS depart.** and the **Online Judge development team** for supporting this service.

Login

- Register your ID:
  MS+studentID
  e.g., MS10203040506



Homework Page

http://acm.cs.nthu.edu.tw/problem/10967/

NTHU Online Judge

10967 - SHiP Cache

Status | Limits Submit

Description

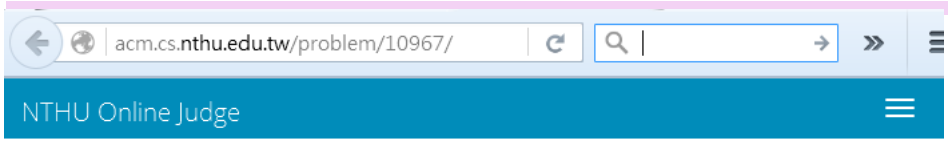
Simulating the SHiP (Signature-bsed Hit Prediction) policy.

Input

The first five lines are as follows:

1. The number of sets of the cache.
2. The number of ways of the cache.
3. "2", which indicates the SHiP policy.
4. The number of saturating counters used by the SHiP policy.
5. The number of operations to simulate.

Submit Your Code



10967 - SHiP Cache

Status | Limits | Submit

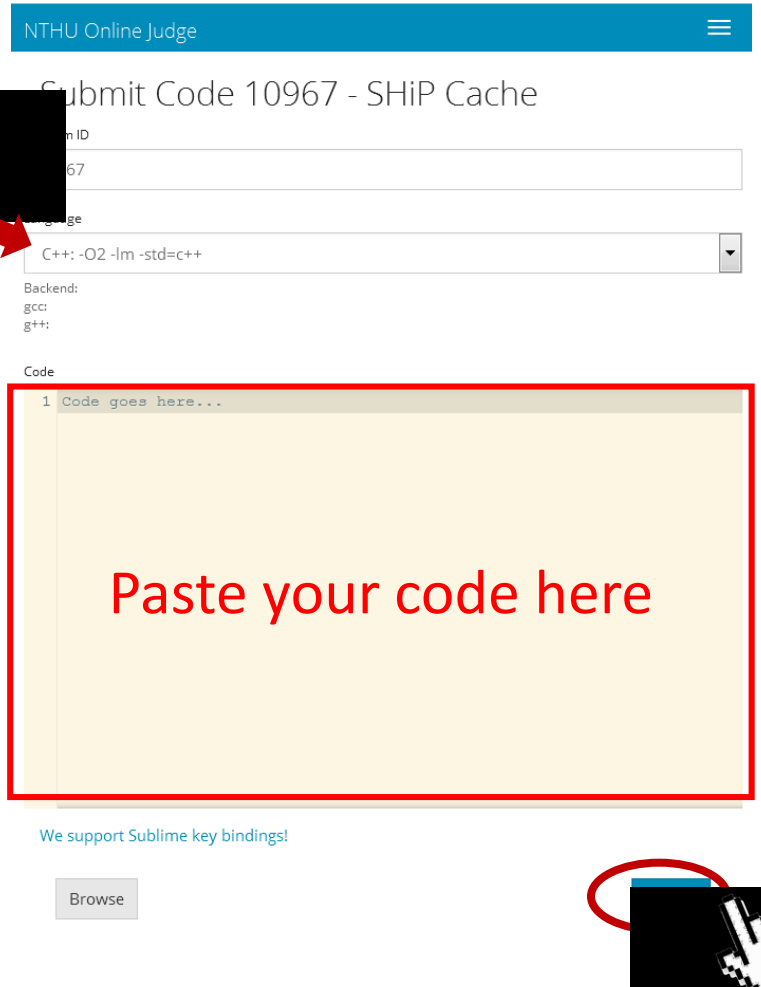
Description

Simulating the SHiP (Signature-bsed Hit Prediction) policy.

Input

The first five lines are as follows:

1. The number of sets of the cache.
2. The number of ways of the cache.
3. "2", which indicates the SHiP policy.
4. The number of saturating counters used by the SHiP policy.
5. The number of operations to simulate.



Check Your Results

acm.cs.nthu.edu.tw/problem/10967/

NTHU Online Judge

10967 - SHiP Cache

Status | Limits Submit

Description

Simulating the SHiP (Signature-bsed Hit Prediction) policy.

Input

The first five lines are as follows:

1. The number of sets of the cache.
2. The number of ways of the cache.
3. "2", which indicates the SHiP policy.
4. The number of saturating counters used by the SHiP policy.
5. The number of operations to simulate.

SID	Submit Time	Username	Problem	Status	Source
1347751	March 16, 2016, 4:01 p.m.	judgeDS2	10967 - SHiP Cache	All Accepted (10/10)	C++
1347738	March 16, 2016, 2:20 p.	judgeDS2	10967 - SHiP Cache	Not	C++
1347736	M 20 p.				
1347732	M 20 p.			pted	C++
1347731	M 20 p.			pted 0)	C++
1347721	M 20 11			pted 0)	C++
1347720	March 16, 2016, 11:44 a.m.	judgeDS2	10967 - SHiP Cache	Not Accepted (6/10)	C++
1347649	March 15, 2016, 6:17 p.m.	judgeDS2	10967 - SHiP Cache	All Accepted (10/10)	C++
1347648	March 15, 2016, 6:03 p.m.	judgeDS2	10967 - SHiP Cache	Compile Error	C++

1.
2. Wrong Answer CPU: 1
3. Wrong Answer CPU: 1
4. Wrong Answer CPU: 15 ms
5. Wrong Answer CPU: 13 ms
6. Wrong Answer CPU: 19 ms
7. Wrong Answer CPU: 27 ms
8. Wrong Answer CPU: 35 ms
9. Accepted CPU: 68 ms
10. Accepted CPU: 134 ms

Accepted CPU: 246 ms

Hints

- Recommend you to develop, test, and debug programs in the Linux environment

- Online Judge can display error types (wrong answer, compiler error, etc.) and compiler errors
- But Online Judge does not show the exact error situation (e.g., the inputs, expected outputs, and your outputs)

Hints

- Use tools to help comparing your answer with the expected answer
- Non-standard libraries and some functions such as file IO are restricted in NTHU OJ. Using them leads to a "restricted function" error
- Extra space, newline characters, etc. lead to "representation error"
  - Online judge performs character to character matching to determine whether the output is correct

Grading Policy

- Grading
  - 100% Online judge results (including submit.cpp)
  - Please make sure that I can perform simulation using the original simulator code and your code
  - -10% per week till 70%

Other Details

- Other details will be announced at eeclass
- Please post your questions in the eeclass 討論區
- Please do not share or post your code or copy other students' code
  - Zero credit for all equivalent code