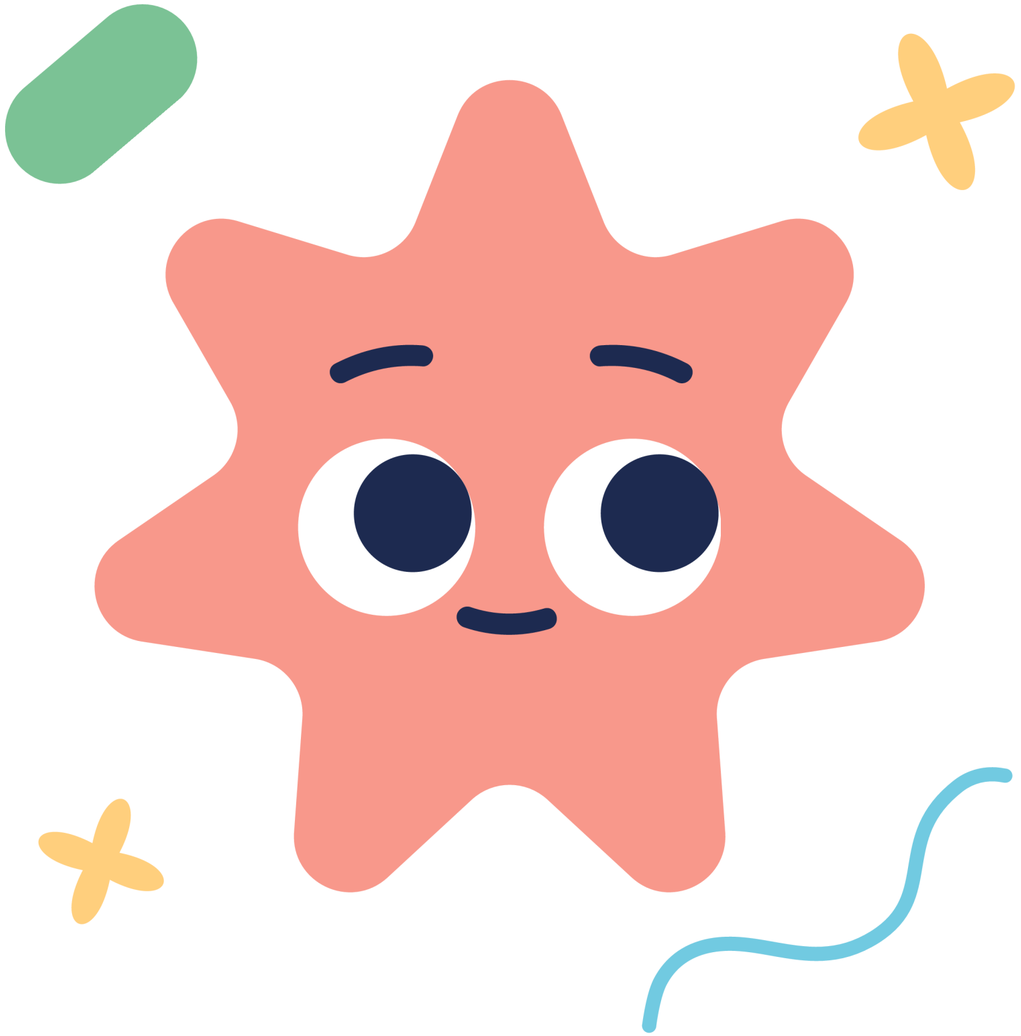
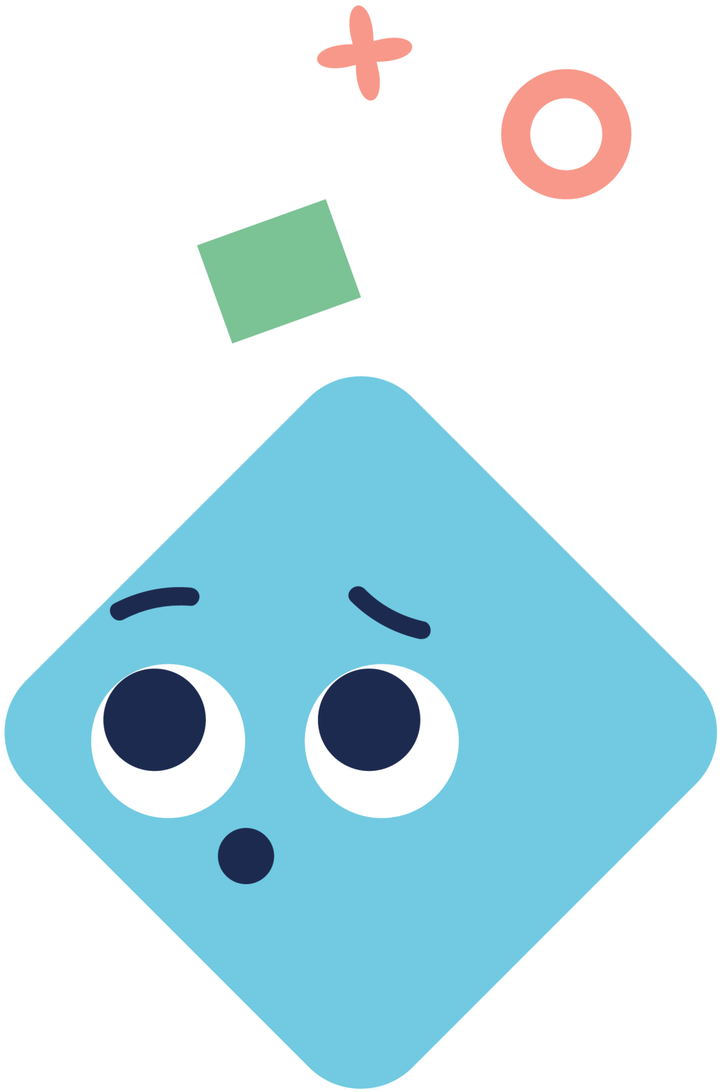
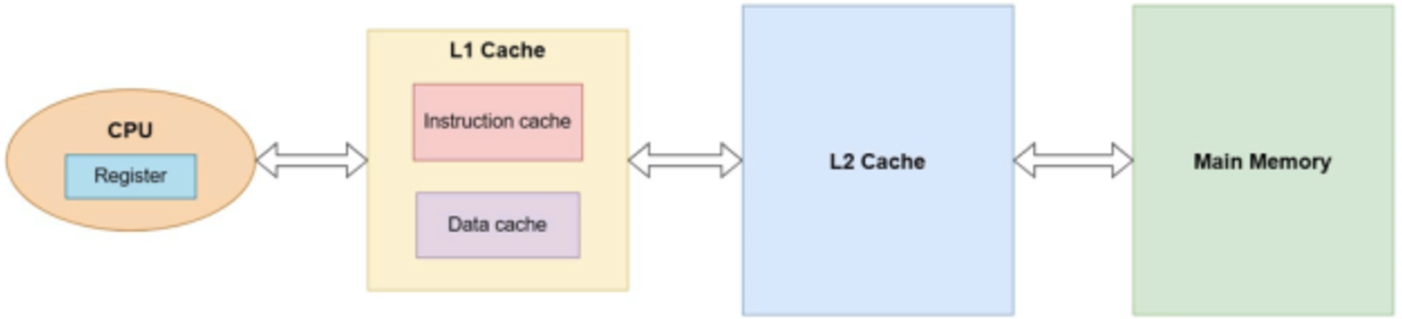
Group 6

Cache Design

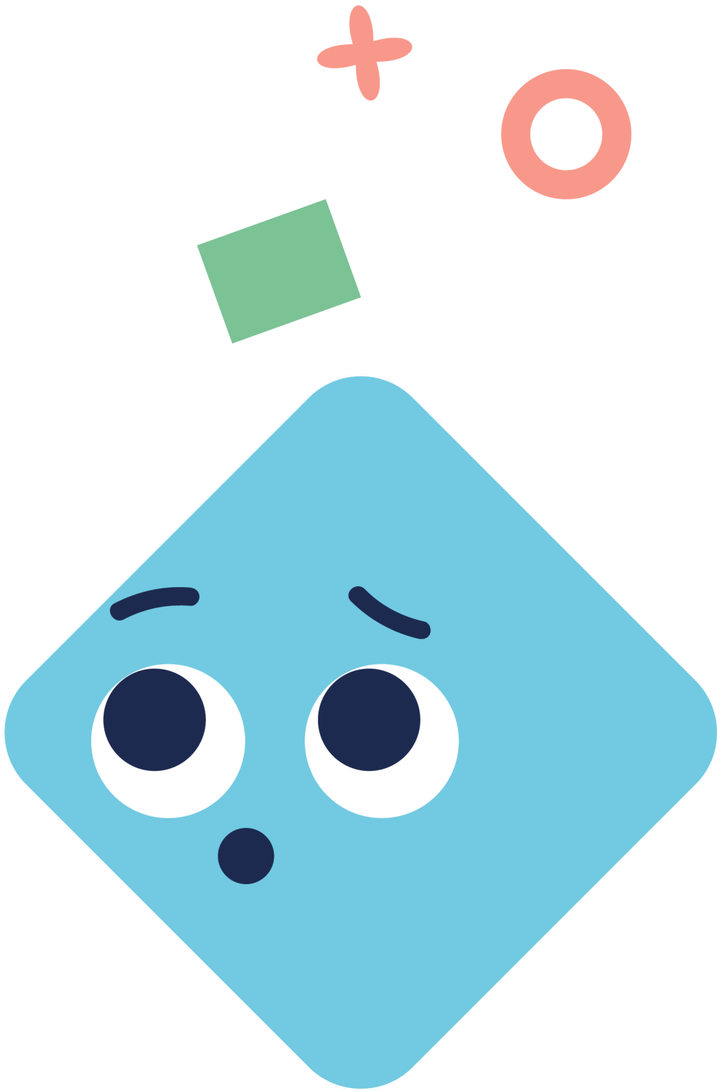




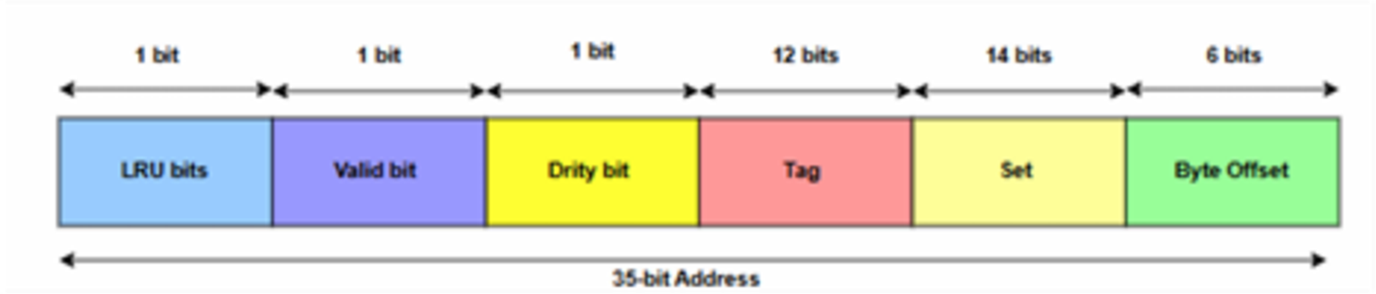
System design



Block diagram



System design

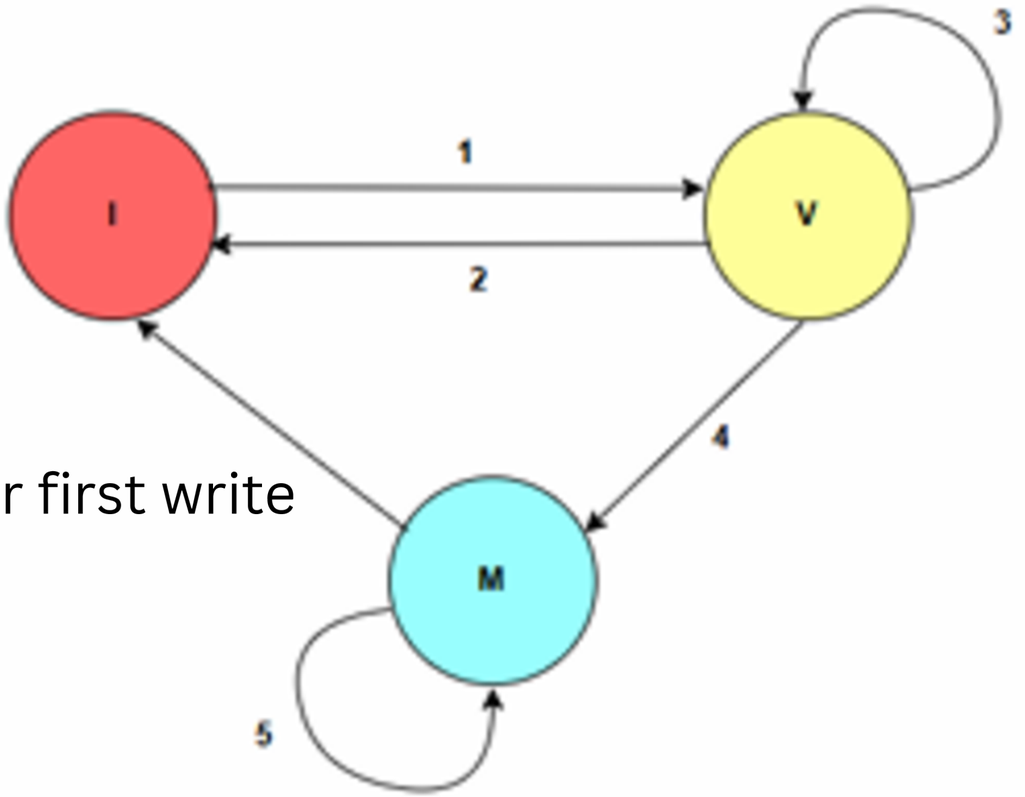


Instruction cache organization

Data cache organization

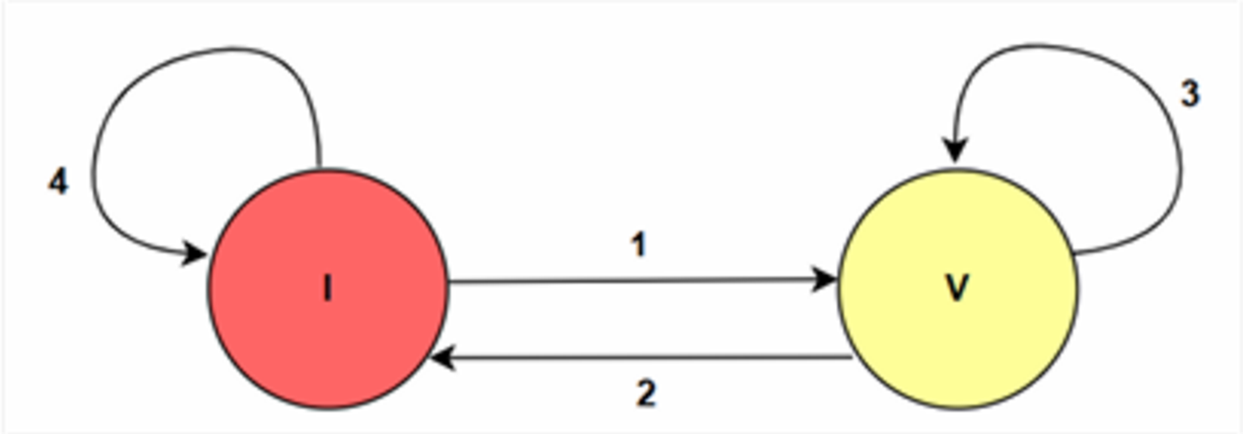
Cache Protocol

Data cache protocol



1. n =0or n=1,Data Read or Data Write but only for first write
2. n =3or n=8,Evict command from L2 or Reset
3. n =0, Data Read
4. n =1, Data Write
5. n =0 or n=1, Data Read or Write
6. n =3or n=8,Evict command from L2 or Rese

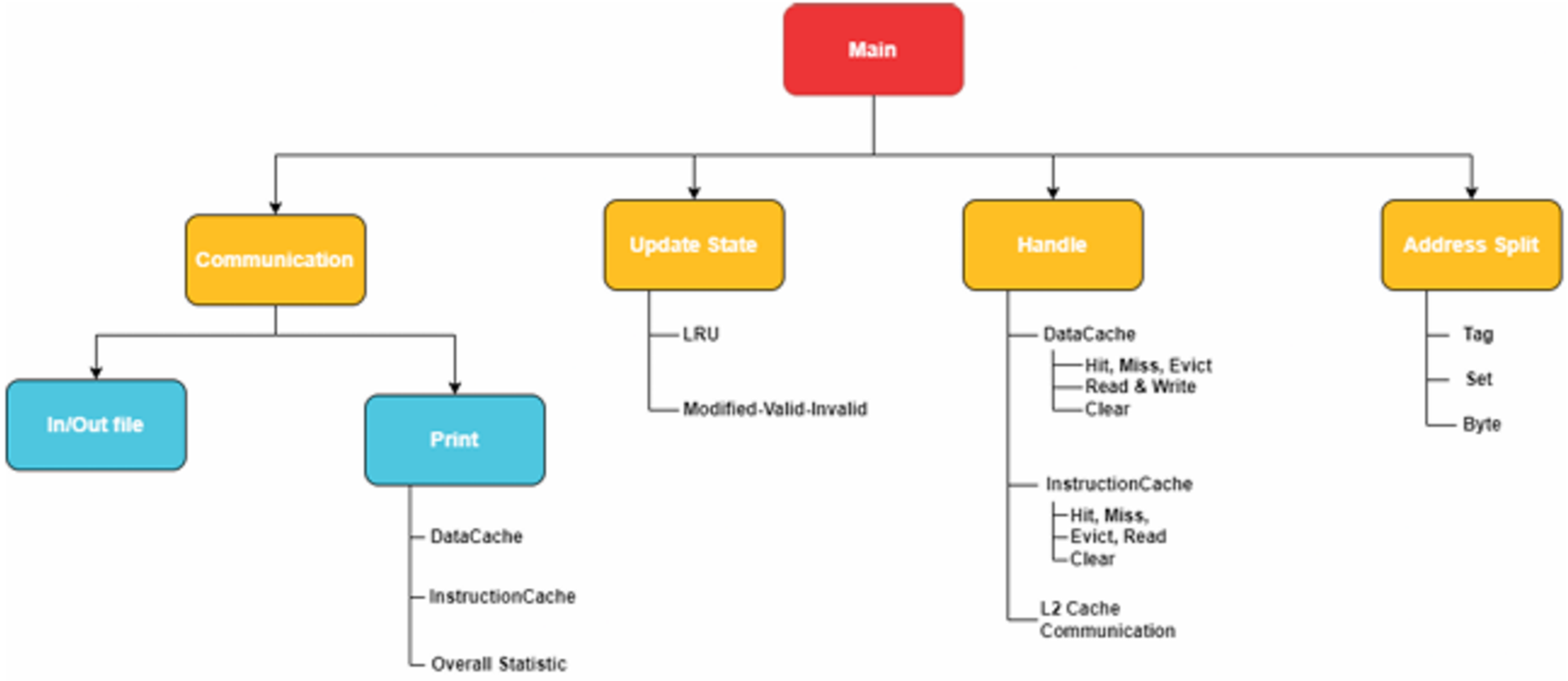
Cache Protocol



1. n = 2, Read Instruction fetch
2. n = 8, Reset
3. n = 2, Read Instruction fetch
4. n = 8, Rese

Instruction cache protocol

Function Decomposition



Pseudo_code

```
void L2COM(int set_index, int line_index, int type) {
    // Handles communication with L2 cache
    Open ComFile in append mode
    If ComFile is unavailable, print error and return
    If valid mode and type:
        Write appropriate L2 communication message based on type
    Close ComFile
}

void UpdateState_DataCache(int set, int way, int event) {
    // Updates the state of a cache line in the data cache
    Switch current state:
        Case 'M':
            Handle transitions for RESET or L2_EVICT change to Invalid state
            else  Read or Write remain the same state of Modified line
        Case 'V':
            Handle transitions for RESET, WRITE, or L2_EVICT
            Reset --> Invalid state
            L1_WRITE_DATA --> Modified
            L2_EVICT --> Invalid
            else Valid
        Default:
            Handle transitions for READ or WRITE
            L1_READ_DATA or L1_WRITE_DATA --> Valid
            else evict from L2 or RESET --> Invalid
}

void UpdateState_InsCache(int set, int way, int event) {
    // Updates the state of a cache line in the instruction cache
    Switch current state:
        Case 'V':
            Handle transitions for RESET or L2_EVICT change state to Invalid
            L1_READ_INST --> Valid
        Default:
            Handle transitions for RESET or FETCH
            L1_READ_INST --> Valid
            RESET or L2_EVICT --> Invalid
}
void DataUpdateLRU(int set, int way) {
    // Updates LRU for data cache
    Adjust LRU values for all lines in the set
    Mark the current line as Most Recently Used (MRU)
}

void DataEvict(int set_index, int evict_tag) {
    // Evicts a line from the data cache
    For each line in the set:
        If line matches evict_tag:
            Handle eviction based on its current state
            Break
}

void DataHit(int set_index, int line_index, int event) {
    // Handles data cache hit
    Print type of hit (read or write)
    Update cache state, LRU, and hit statistics
}

void DataMiss(int set_index, int line_index, int new_tag, int event) {
    // Handles data cache miss
    Print type of miss
    Evict LRU line if required
    Fetch new data and update cache state, address, and LRU
    Update miss statistics
}
```

Pseudo_code

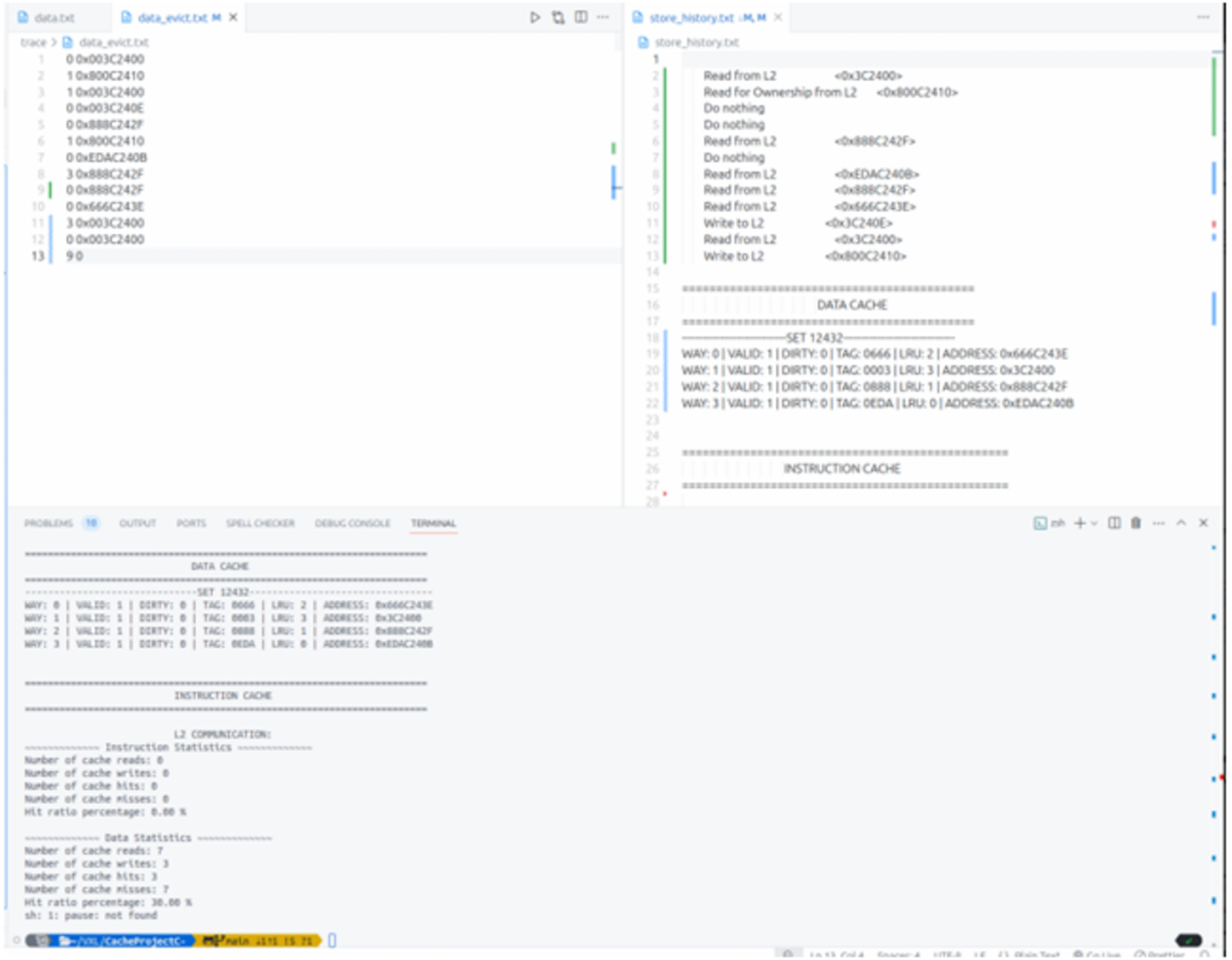
```
void DataReadWrite(int set_index, int req_tag, int event) {
    // Handles read/write operation for data cache
    Search for tag in the set:
        If found, call DataHit
        If not, call DataMiss with appropriate line
}

void DataClear() {
    // Clears all data cache entries
    For each set and line:
        Reset state and LRU values to default
}

void PrintDataCache() {
    // Prints all valid data cache entries
    For each set and line:
        If line is valid, print details
}

void InsEvict(int set_index, int evict_tag) {
    // Evicts a line from the instruction cache
    For each line in the set:
        If line matches evict_tag:
            Handle eviction and update state
}

void InsHit(int set_index, int line_index, int event) {
    // Handles instruction cache hit
    Print "Instruction cache: Read Hit"
    Update cache state and LRU
    Increment hit statistics
}

void InsMiss(int set_index, int line_index, int new_tag, int event) {
    // Handles instruction cache miss
    Print "Instruction cache: Read Miss"
    Update cache state, address, and LRU
    Increment miss statistics
}
```

```
void InsRead(int set_index, int req_tag, int event) {
    // Handles read operation for instruction cache
    Search for tag in the set:
        If found, call InsHit
        If not, call InsMiss with appropriate line
}

void InsClear() {
    // Clears all instruction cache entries
    For each set and line:
        Reset state and LRU values to default
}

void PrintInsCache() {
    // Prints all valid instruction cache entries
    For each set and line:
        If line is valid, print details
}
```

Test for hit rate

Test case		Read data L1			
Action		Hit	Miss	Hit ratio	Result
0	10019d94		x		PASS
0	10019d88	x			PASS
0	20019d88		x		PASS
0	10019d94	x			PASS
0	10019d94	x			PASS
0	40019d94		x		PASS
0	10019d94	x			PASS
0	30019d94		x		PASS
0	70019d94		x		PASS
9	0	4	5	44.44%	PASS
0	40019d94	x			PASS
0	40019d94	x			PASS
3	10019d94				PASS
0	40019d94	x			PASS
0	10019d94		x		PASS
0	10019d94	x			PASS
9	0	8	6	57.14 %	PASS

Evict line


data.txt
data_evict.txt M
trace > data_evict.txt
0 0x003C2400
1 0x800C2410
1 0x003C2400
0 0x003C240E
0 0x888C242F
1 0x800C2410
0 0xEDAC240B
3 0x888C242F
0 0x888C242F
0 0x666C243E
3 0x003C2400
0 0x003C2400
9 0
store_history.txt M, M
store_history.txt
Read from L2 <0x3C2400>
Read for Ownership from L2 <0x800C2410>
Do nothing
Do nothing
Read from L2 <0x888C242F>
Do nothing
Read from L2 <0xEDAC240B>
Read from L2 <0x888C242F>
Read from L2 <0x666C243E>
Write to L2 <0x3C240E>
Read from L2 <0x3C2400>
Write to L2 <0x800C2410>
DATA CACHE
SET 12432
WAY: 0 | VALID: 1 | DIRTY: 0 | TAG: 0666 | LRU: 2 | ADDRESS: 0x666C243E
WAY: 1 | VALID: 1 | DIRTY: 0 | TAG: 0003 | LRU: 3 | ADDRESS: 0x3C2400
WAY: 2 | VALID: 1 | DIRTY: 0 | TAG: 0888 | LRU: 1 | ADDRESS: 0x888C242F
WAY: 3 | VALID: 1 | DIRTY: 0 | TAG: 0EDA | LRU: 0 | ADDRESS: 0xEDAC240B
INSTRUCTION CACHE
PROBLEMS 10
OUTPUT
PORTS
SPELL CHECKER
DEBUG CONSOLE
TERMINAL
zsh
DATA CACHE
SET 12432
WAY: 0 | VALID: 1 | DIRTY: 0 | TAG: 0666 | LRU: 2 | ADDRESS: 0x666C243E
WAY: 1 | VALID: 1 | DIRTY: 0 | TAG: 0003 | LRU: 3 | ADDRESS: 0x3C2400
WAY: 2 | VALID: 1 | DIRTY: 0 | TAG: 0888 | LRU: 1 | ADDRESS: 0x888C242F
WAY: 3 | VALID: 1 | DIRTY: 0 | TAG: 0EDA | LRU: 0 | ADDRESS: 0xEDAC240B
INSTRUCTION CACHE
L2 COMMUNICATION:
Instruction Statistics
Number of cache reads: 0
Number of cache writes: 0
Number of cache hits: 0
Number of cache misses: 0
Hit ratio percentage: 0.00 %
Data Statistics
Number of cache reads: 7
Number of cache writes: 3
Number of cache hits: 3
Number of cache misses: 7
Hit ratio percentage: 30.00 %
sh: 1: pause: not found
/CacheProjectC-
main ↓11 !15 ?5

Write behavior

store_history.txt

```
Read from L2              <0x3C2400>
Read for Ownership from L2    <0x800C2410>
Read for Ownership from L2    <0x903C2400>
Do nothing
Read from L2              <0x888C242F>
Do nothing
Read from L2              <0xEDAC240B>
Read from L2              <0xAAAC240B>
Read for Ownership from L2    <0x3C2410>
Read for Ownership from L2    <0x903C2410>
Write to L2             <0x800C2410>
Read for Ownership from L2    <0x800C2410>
Do nothing
Read for Ownership from L2    <0x600C2410>
Read for Ownership from L2    <0x703C2400>
Read from L2              <0x888C242F>
Read from L2              <0x666C243E>
Write to L2             <0x903C2400>
Read from L2              <0x3C2400>

=========================================
                DATA CACHE
=========================================
----------------------SET 12432----------------------
WAY: 0 | VALID: 1 | DIRTY: 0 | TAG: 0003 | LRU: 3 | ADDRESS: 0x3C2400
WAY: 1 | VALID: 1 | DIRTY: 0 | TAG: 0666 | LRU: 2 | ADDRESS: 0x666C243E
```

trace > data_write_through.txt

```
0 0x003C2400
1 0x800C2410
1 0x903C2400
0 0x003C240E
0 0x888C242F
1 0x800C2410
0 0xEDAC240B
0 0xAAAC240B
1 0x003C2410
1 0x903C2410
1 0x800C2410
1 0x903C2400
1 0x600C2410
1 0x703C2400
0 0x888C242F
0 0x666C243E
0 0x003C2400
9 0
```

PROBLEMS OUTPUT PORTS SPELL CHECKER DEBUG CONSOLE TERMINAL

```
Data cache: Write Miss but Write Through because first write (unoccupied)

0 0x003C240E
Data cache: Read Hit

0 0x888C242F
Data cache: Read Miss (unoccupied)

1 0x888C2410
Data cache: Write Hit

0 0xEDAC240B
Data cache: Read Miss (conflict)
Evict way 2

0 0xAAAC240B
Data cache: Read Miss (conflict)
Evict way 0

1 0x003C2410
Data cache: Write Miss but Write Through because first write (conflict)
Evict way 3

1 0x903C2410
Data cache: Write Miss but Write Through because first write (conflict)
Evict dirty data of way 1

1 0x800C2410
Data cache: Write Miss but Write Through because first write (conflict)
Evict way 2

1 0x903C2400
Data cache: Write Hit

1 0x600C2410
Data cache: Write Miss but Write Through because first write (conflict)
Evict way 0

1 0x703C2400
Data cache: Write Miss but Write Through because first write (conflict)
Evict way 3

0 0x888C242F
Data cache: Read Miss (conflict)
```

Thank You For Playing!

See You Next Time!