CS322 ALGORITHM LABORATORY
MINI PROJECT
EXTENDING FUNCTIONALITY OF MARS MIPS ASSEMBLER

SUBMITTED BY: KHUSHI PRASAD   ROLL NUMBER: 2001CS38

CACHE:

They refer to memory storage managed to take advantage of locality of access. They bridge the speed gap between processor and main memory. A cache provides the processor requested data from the memory. If it is not found in the cache, the data item is then stored in the cache from the main memory. However, cache has limited storage.

So, in case of full storage, how to decide which block to replace? There are certain Block Replacement Policies for the same.

BLOCK REPLACEMENT POLICIES:

Mars already provides us with two types of replacement policies - LRU and Random. We have extended this functionality and included 3 more block replacement policies - LFU, FIFO AND LRU-2.

1. **Least Recently Used**:
   This algorithm replaces the block that has not been used for the longest period of time. It is based on the observation that blocks that have not been used for a long time will probably remain unused for the longest time and thus, be replaced.

2. **Random**:
   This algorithm replaces blocks randomly.

Following Cache Simulation functionalities have been added:

1. **Least Frequently Used**:

This algorithm replaces the block that has been accessed the least number of times.

IMPLEMENTATION:
Defined 2 new variables:

- accesscount -> one to keep track of the number of times the respective block has been accessed.
- leastAccesscount -> one to keep track of the global minima.

We initialise the variable <accesscount> whenever the respective memory block is accessed.

- When a set is full or it is not in the cache, it is initialised to 1.
- When it is found in the cache, the value of the variable is increased by one.

In case of full set,

- We iterate over all the blocks updating <leastAccesscount> and storing the respective block.
- At the end of the loop, the block that has been accessed the least number of times is replaced.

---

2. **First IN First OUT**:
   In this algorithm, the oldest block, which has spent the longest time in memory is chosen and replaced

IMPLEMENTATION:
Define 2 new variables:

- creationtime -> to store the time when the blocks were first used.
- leastRecentCreationtime -> to store the least creation time.

We initialise the variable <creationtime> whenever the respective memory block is accessed.

- When the set is full or the data item is found in cache, the respective variable is initialised to memoryAccesscount.

In case of full set,

- We iterate over all the blocks updating <leastRecentCreationtime> and storing the respective block.
- At the end of the loop, the block that has been accessed the earliest is replaced.

---

3. **LRU-2:**
   In this, instead of the most recent access time, we compare the second most recent access time of the block. Generally speaking, LRU-k compares the kth recent access time for comparison and deciding which block has to be replaced in case of a full set.

IMPLEMENTATION:
Define 2 new variables:
- secondmostRecentAccesstime -> to store the time when the blocks were used second most recent time.
- leastRecentsecondAccessTime -> to store the least second most access time of all the blocks.

We initialise the variable <secondmostRecentAccesstime> whenever the respective memory block is accessed.
- In case the set is full or the data item is not in the cache, we initialise the respective variable with a large arbitrary number.
- In case the data item is already in the cache and need not be fetched from the memory, we assign the value of the variable <mostRecentAccesstime> to <secondmostRecentAccesstime> and update the value of <mostRecentAccesstime>.

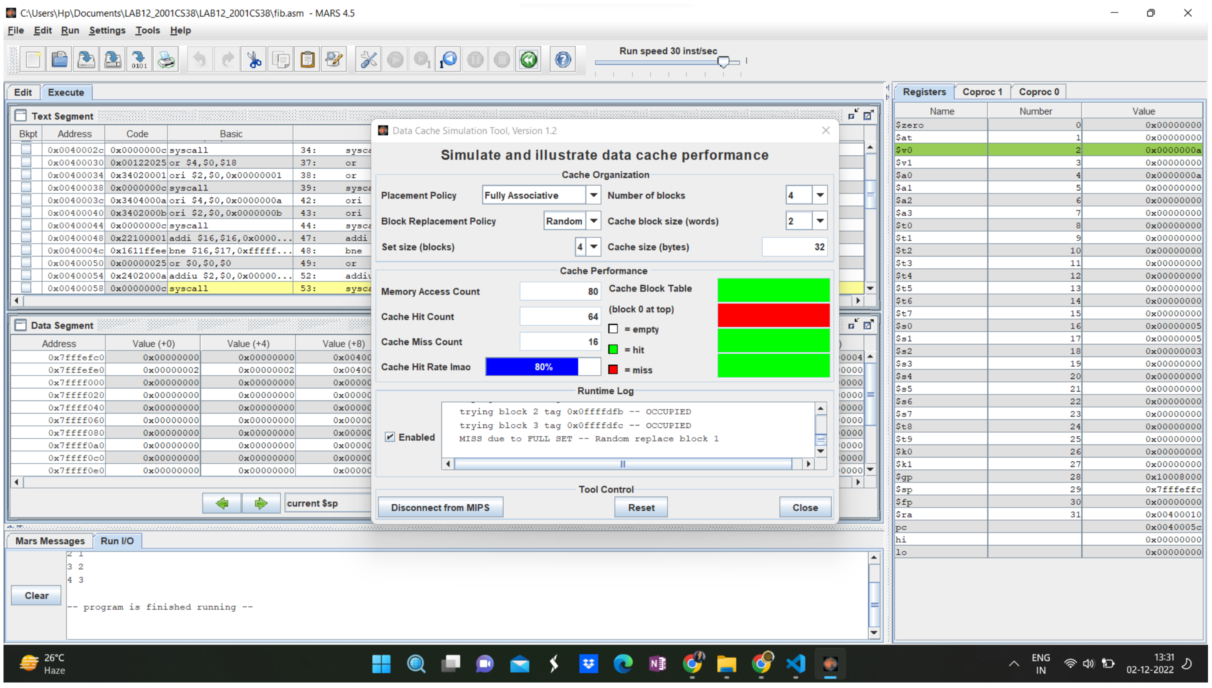
In case of full set,
- We iterate over all the blocks updating <leastRecentsecondAccessTime> and storing the respective block.
- At the end of the loop, the block with the least second most recent access time is replaced.

PERFORMANCE COMPARISON:

For this, we will consider Fully Associative Mapping as in a fully associative mapping all blocks are likely to be replaced.
We will consider 4 blocks in the cache, where each block can store 2 words each.
For performance comparison, we will consider two standard programs - Fibonacci and Factorial of a Number (Recursive Method).

FIBONACCI RECURSIVE CODE:

1. RANDOM:



C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fib.asm - MARS 4.5

Text Segment

Bkpt	Address	Code	Basic	
	0x0040002c	0x0000000c	syscall	34: sysca
	0x00400030	0x00122025	or $4,$0,$18	37: or
	0x00400034	0x34020001	ori $2,$0,0x00000001	38: or
	0x00400038	0x0000000c	syscall	39: sysca
	0x0040003c	0x3404000a	ori $4,$0,0x0000000a	42: ori
	0x00400040	0x3402000b	ori $2,$0,0x0000000b	43: ori
	0x00400044	0x0000000c	syscall	44: sysca
	0x00400048	0x22100001	addi $16,$16,0x0000...	47: addi
	0x0040004c	0x1611ffee	bne $16,$17,0xfffff...	48: bne
	0x00400050	0x00000025	or $0,$0,$0	49: or
	0x00400054	0x2402000a	addiu $2,$0,0x00000...	52: addiu
	0x00400058	0x0000000c	syscall	53: sysca

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x7fffefc0	0x00000000	0x00000000	0x00400
0x7fffefe0	0x00000002	0x00000002	0x00400
0x7ffff000	0x00000000	0x00000000	0x00000
0x7ffff020	0x00000000	0x00000000	0x00000
0x7ffff040	0x00000000	0x00000000	0x00000
0x7ffff060	0x00000000	0x00000000	0x00000
0x7ffff080	0x00000000	0x00000000	0x00000
0x7ffff0a0	0x00000000	0x00000000	0x00000
0x7ffff0c0	0x00000000	0x00000000	0x00000
0x7ffff0e0	0x00000000	0x00000000	0x00000

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	Random	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	80
Cache Hit Count	64
Cache Miss Count	16
Cache Hit Rate Imao	80%

Cache Block Table (block 0 at top): 🟩 🟥 🟩 🟩

☐ = empty 🟩 = hit 🟥 = miss

Runtime Log [✓ Enabled]

```
trying block 2 tag 0x0ffffdfb -- OCCUPIED
trying block 3 tag 0x0ffffdfc -- OCCUPIED
MISS due to FULL SET -- Random replace block 1
```

Run I/O

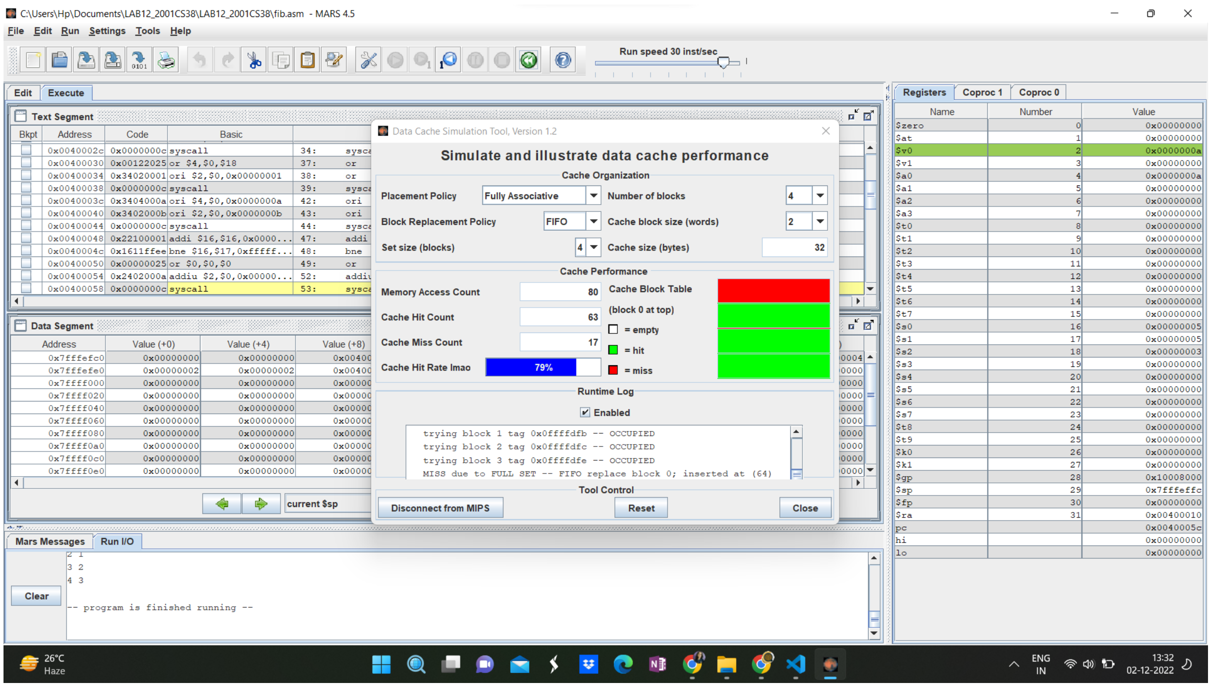
```
2 1
3 2
4 3

-- program is finished running --
```

Registers

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000000a
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000005
$s1	17	0x00000005
$s2	18	0x00000003
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x0040005c
hi		0x00000000
lo		0x00000000

2. FIFO:


C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fib.asm - MARS 4.5

Run speed 30 inst/sec

Text Segment

Bkpt	Address	Code	Basic		
	0x0040002c	0x0000000c	syscall	34:	sysca
	0x00400030	0x00122025	or $4,$0,$18	37:	or
	0x00400034	0x34020001	ori $2,$0,0x00000001	38:	or
	0x00400038	0x0000000c	syscall	39:	sysca
	0x0040003c	0x3404000a	ori $4,$0,0x0000000a	42:	ori
	0x00400040	0x3402000b	ori $2,$0,0x0000000b	43:	ori
	0x00400044	0x0000000c	syscall	44:	sysca
	0x00400048	0x22100001	addi $16,$16,0x0000...	47:	addi
	0x0040004c	0x1611ffee	bne $16,$17,0xfffff...	48:	bne
	0x00400050	0x00000025	or $0,$0,$0	49:	or
	0x00400054	0x2402000a	addiu $2,$0,0x00000...	52:	addiu
	0x00400058	0x0000000c	syscall	53:	sysca

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x7fffefc0	0x00000000	0x00000000	0x00400
0x7fffefe0	0x00000002	0x00000002	0x00400
0x7ffff000	0x00000000	0x00000000	0x00000
0x7ffff020	0x00000000	0x00000000	0x00000
0x7ffff040	0x00000000	0x00000000	0x00000
0x7ffff060	0x00000000	0x00000000	0x00000
0x7ffff080	0x00000000	0x00000000	0x00000
0x7ffff0a0	0x00000000	0x00000000	0x00000
0x7ffff0c0	0x00000000	0x00000000	0x00000
0x7ffff0e0	0x00000000	0x00000000	0x00000

current $sp

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	FIFO	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	80
Cache Hit Count	63
Cache Miss Count	17
Cache Hit Rate Imao	79%

Cache Block Table (block 0 at top): 🟥 🟩 🟩 🟩

☐ = empty 🟩 = hit 🟥 = miss

Runtime Log

☑ Enabled

```
  trying block 1 tag 0x0ffffdfb -- OCCUPIED
  trying block 2 tag 0x0ffffdfc -- OCCUPIED
  trying block 3 tag 0x0ffffdfe -- OCCUPIED
  MISS due to FULL SET -- FIFO replace block 0; inserted at (64)
```

Tool Control

Registers

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000000a
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000005
$s1	17	0x00000005
$s2	18	0x00000003
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x0040005c
hi		0x00000000
lo		0x00000000

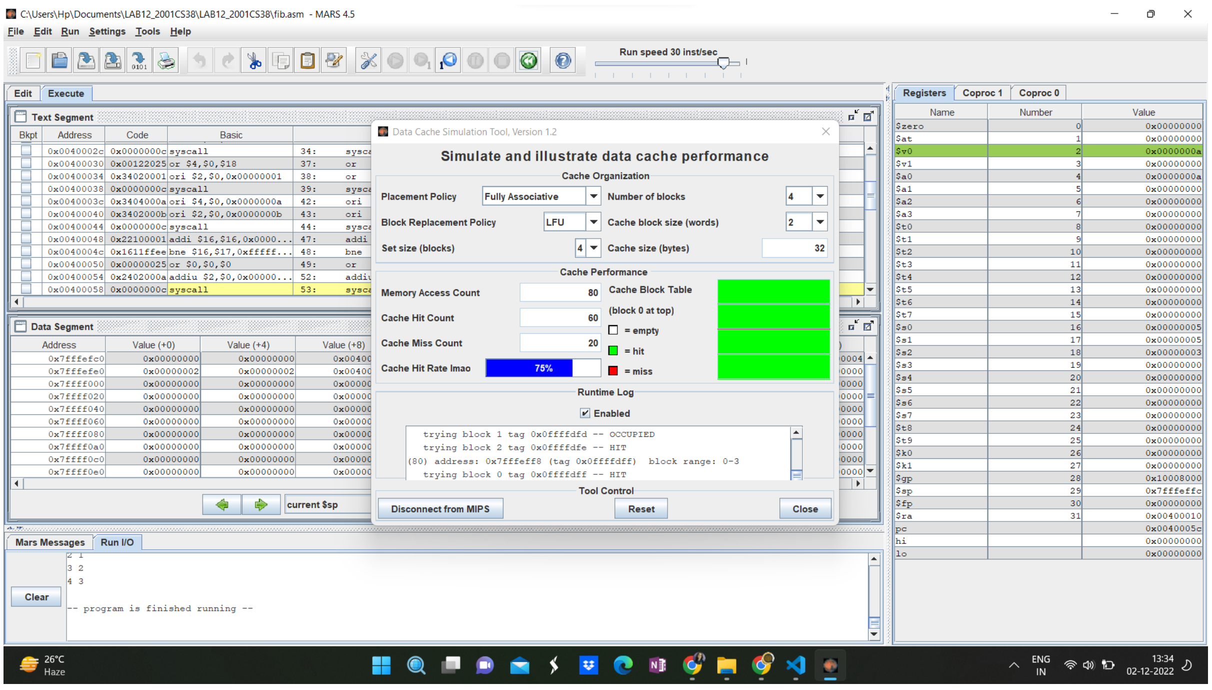
Run I/O

```
2 1
3 2
4 3

-- program is finished running --
```

26°C Haze — ENG IN 13:32 02-12-2022

3. LFU:


C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fib.asm - MARS 4.5

Run speed 30 inst/sec

Text Segment

Bkpt	Address	Code	Basic		
	0x0040002c	0x0000000c	syscall	34:	sysca
	0x00400030	0x00122025	or $4,$0,$18	37:	or
	0x00400034	0x34020001	ori $2,$0,0x00000001	38:	or
	0x00400038	0x0000000c	syscall	39:	sysca
	0x0040003c	0x3404000a	ori $4,$0,0x0000000a	42:	ori
	0x00400040	0x3402000b	ori $2,$0,0x0000000b	43:	ori
	0x00400044	0x0000000c	syscall	44:	sysca
	0x00400048	0x22100001	addi $16,$16,0x0000...	47:	addi
	0x0040004c	0x1611ffee	bne $16,$17,0xfffff...	48:	bne
	0x00400050	0x00000025	or $0,$0,$0	49:	or
	0x00400054	0x2402000a	addiu $2,$0,0x00000...	52:	addiu
	0x00400058	0x0000000c	syscall	53:	sysca

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x7fffefc0	0x00000000	0x00000000	0x00400
0x7fffefe0	0x00000002	0x00000002	0x00400
0x7ffff000	0x00000000	0x00000000	0x00000
0x7ffff020	0x00000000	0x00000000	0x00000
0x7ffff040	0x00000000	0x00000000	0x00000
0x7ffff060	0x00000000	0x00000000	0x00000
0x7ffff080	0x00000000	0x00000000	0x00000
0x7ffff0a0	0x00000000	0x00000000	0x00000
0x7ffff0c0	0x00000000	0x00000000	0x00000
0x7ffff0e0	0x00000000	0x00000000	0x00000

current $sp

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	LFU	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	80
Cache Hit Count	60
Cache Miss Count	20
Cache Hit Rate Imao	75%

Cache Block Table (block 0 at top): 🟩 🟩 🟩 🟩

☐ = empty 🟩 = hit 🟥 = miss

Runtime Log

☑ Enabled

```
  trying block 1 tag 0x0ffffdfd -- OCCUPIED
  trying block 2 tag 0x0ffffdfe -- HIT
(80) address: 0x7fffeff8 (tag 0x0ffffdff)  block range: 0-3
  trying block 0 tag 0x0ffffdff -- HIT
```

Tool Control

Registers

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x0000000a
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000005
$s1	17	0x00000005
$s2	18	0x00000003
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400010
pc		0x0040005c
hi		0x00000000
lo		0x00000000

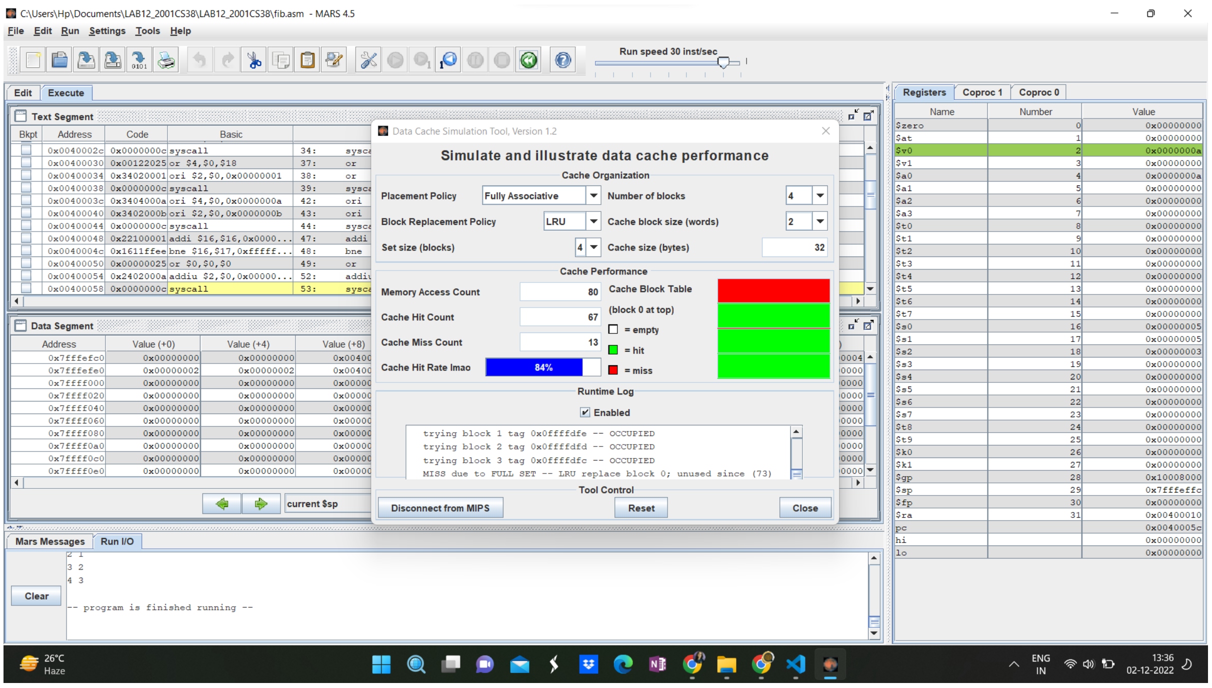
Run I/O

```
2 1
3 2
4 3

-- program is finished running --
```

26°C Haze — ENG IN 13:34 02-12-2022

4. LRU:


C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fib.asm - MARS 4.5

Run speed 30 inst/sec

Text Segment

```
Bkpt  Address     Code        Basic
      0x0040002c  0x0000000c  syscall                    34:   sysca
      0x00400030  0x00122025  or $4,$0,$18               37:   or
      0x00400034  0x34020001  ori $2,$0,0x00000001       38:   or
      0x00400038  0x0000000c  syscall                    39:   sysca
      0x0040003c  0x3404000a  ori $4,$0,0x0000000a       42:   ori
      0x00400040  0x3402000b  ori $2,$0,0x0000000b       43:   ori
      0x00400044  0x0000000c  syscall                    44:   sysca
      0x00400048  0x22100001  addi $16,$16,0x0000...     47:   addi
      0x0040004c  0x1611ffee  bne $16,$17,0xfffff...     48:   bne
      0x00400050  0x00000025  or $0,$0,$0                49:   or
      0x00400054  0x2402000a  addiu $2,$0,0x00000...     52:   addiu
      0x00400058  0x0000000c  syscall                    53:   sysca
```

Data Segment

```
Address     Value (+0)  Value (+4)  Value (+8)
0x7fffefc0  0x00000000  0x00000000  0x00400
0x7fffefe0  0x00000002  0x00000002  0x00400
0x7ffff000  0x00000000  0x00000000  0x0000
0x7ffff020  0x00000000  0x00000000  0x0000
0x7ffff040  0x00000000  0x00000000  0x0000
0x7ffff060  0x00000000  0x00000000  0x0000
0x7ffff080  0x00000000  0x00000000  0x0000
0x7ffff0a0  0x00000000  0x00000000  0x0000
0x7ffff0c0  0x00000000  0x00000000  0x0000
0x7ffff0e0  0x00000000  0x00000000  0x0000
```

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	LRU	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	80
Cache Hit Count	67
Cache Miss Count	13
Cache Hit Rate Imao	84%

Cache Block Table (block 0 at top): 🟥 🟩 🟩 🟩

□ = empty 🟩 = hit 🟥 = miss

Runtime Log

✓ Enabled

```
  trying block 1 tag 0x0ffffdfe -- OCCUPIED
  trying block 2 tag 0x0ffffdfd -- OCCUPIED
  trying block 3 tag 0x0ffffdfc -- OCCUPIED
  MISS due to FULL SET -- LRU replace block 0; unused since (73)
```

Registers

```
Name   Number  Value
$zero  0       0x00000000
$at    1       0x00000000
$v0    2       0x0000000a
$v1    3       0x00000000
$a0    4       0x0000000a
$a1    5       0x00000000
$a2    6       0x00000000
$a3    7       0x00000000
$t0    8       0x00000000
$t1    9       0x00000000
$t2    10      0x00000000
$t3    11      0x00000000
$t4    12      0x00000000
$t5    13      0x00000000
$t6    14      0x00000000
$t7    15      0x00000000
$s0    16      0x00000005
$s1    17      0x00000005
$s2    18      0x00000003
$s3    19      0x00000000
$s4    20      0x00000000
$s5    21      0x00000000
$s6    22      0x00000000
$s7    23      0x00000000
$t8    24      0x00000000
$t9    25      0x00000000
$k0    26      0x00000000
$k1    27      0x00000000
$gp    28      0x10008000
$sp    29      0x7fffeffc
$fp    30      0x00000000
$ra    31      0x00400010
pc             0x0040005c
hi             0x00000000
lo             0x00000000
```

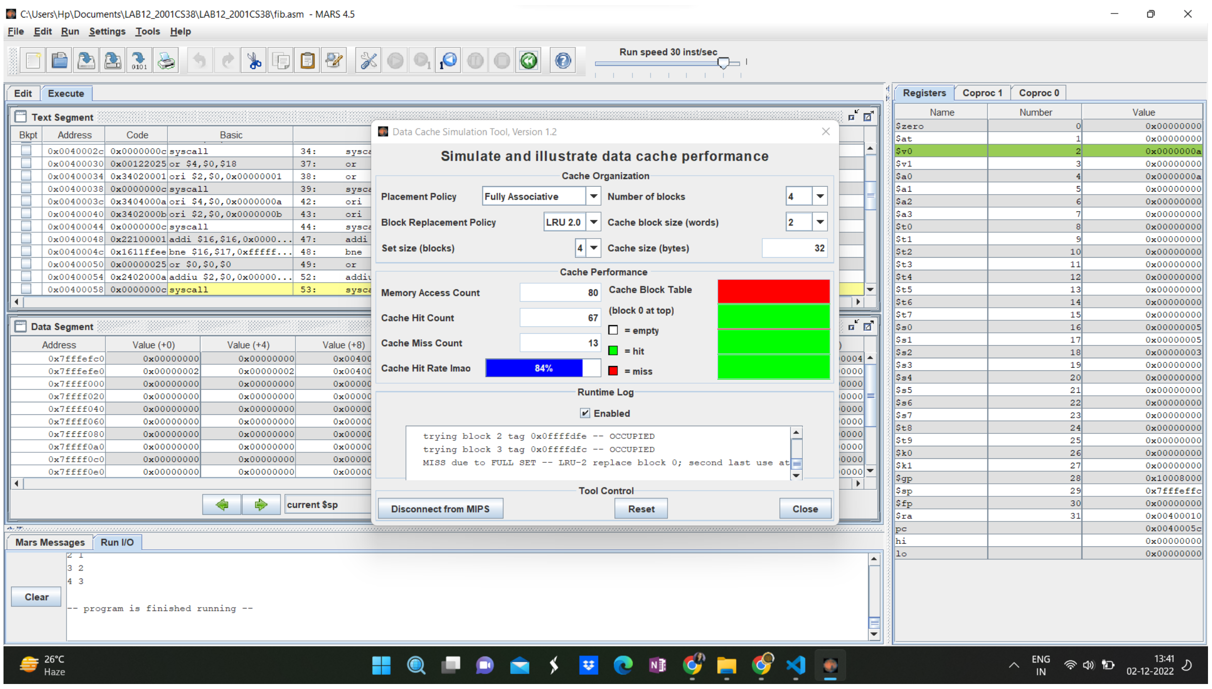
Mars Messages / Run I/O

```
2 1
3 2
4 3

-- program is finished running --
```

26°C Haze ENG IN 13:36 02-12-2022

5. LRU 2


C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fib.asm - MARS 4.5

Run speed 30 inst/sec

Text Segment

```
Bkpt  Address     Code        Basic
      0x0040002c  0x0000000c  syscall                    34:   sysca
      0x00400030  0x00122025  or $4,$0,$18               37:   or
      0x00400034  0x34020001  ori $2,$0,0x00000001       38:   or
      0x00400038  0x0000000c  syscall                    39:   sysca
      0x0040003c  0x3404000a  ori $4,$0,0x0000000a       42:   ori
      0x00400040  0x3402000b  ori $2,$0,0x0000000b       43:   ori
      0x00400044  0x0000000c  syscall                    44:   sysca
      0x00400048  0x22100001  addi $16,$16,0x0000...     47:   addi
      0x0040004c  0x1611ffee  bne $16,$17,0xfffff...     48:   bne
      0x00400050  0x00000025  or $0,$0,$0                49:   or
      0x00400054  0x2402000a  addiu $2,$0,0x00000...     52:   addiu
      0x00400058  0x0000000c  syscall                    53:   sysca
```

Data Segment

```
Address     Value (+0)  Value (+4)  Value (+8)
0x7fffefc0  0x00000000  0x00000000  0x00400
0x7fffefe0  0x00000002  0x00000002  0x00400
0x7ffff000  0x00000000  0x00000000  0x0000
0x7ffff020  0x00000000  0x00000000  0x0000
0x7ffff040  0x00000000  0x00000000  0x0000
0x7ffff060  0x00000000  0x00000000  0x0000
0x7ffff080  0x00000000  0x00000000  0x0000
0x7ffff0a0  0x00000000  0x00000000  0x0000
0x7ffff0c0  0x00000000  0x00000000  0x0000
0x7ffff0e0  0x00000000  0x00000000  0x0000
```

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	LRU 2.0	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	80
Cache Hit Count	67
Cache Miss Count	13
Cache Hit Rate Imao	84%

Cache Block Table (block 0 at top): 🟥 🟩 🟩 🟩

□ = empty 🟩 = hit 🟥 = miss

Runtime Log

✓ Enabled

```
  trying block 2 tag 0x0ffffdfe -- OCCUPIED
  trying block 3 tag 0x0ffffdfc -- OCCUPIED
  MISS due to FULL SET -- LRU-2 replace block 0; second last use at
```

Registers

```
Name   Number  Value
$zero  0       0x00000000
$at    1       0x00000000
$v0    2       0x0000000a
$v1    3       0x00000000
$a0    4       0x0000000a
$a1    5       0x00000000
$a2    6       0x00000000
$a3    7       0x00000000
$t0    8       0x00000000
$t1    9       0x00000000
$t2    10      0x00000000
$t3    11      0x00000000
$t4    12      0x00000000
$t5    13      0x00000000
$t6    14      0x00000000
$t7    15      0x00000000
$s0    16      0x00000005
$s1    17      0x00000005
$s2    18      0x00000003
$s3    19      0x00000000
$s4    20      0x00000000
$s5    21      0x00000000
$s6    22      0x00000000
$s7    23      0x00000000
$t8    24      0x00000000
$t9    25      0x00000000
$k0    26      0x00000000
$k1    27      0x00000000
$gp    28      0x10008000
$sp    29      0x7fffeffc
$fp    30      0x00000000
$ra    31      0x00400010
pc             0x0040005c
hi             0x00000000
lo             0x00000000
```

Mars Messages / Run I/O

```
2 1
3 2
4 3

-- program is finished running --
```

26°C Haze ENG IN 13:41 02-12-2022

ANALYSIS:

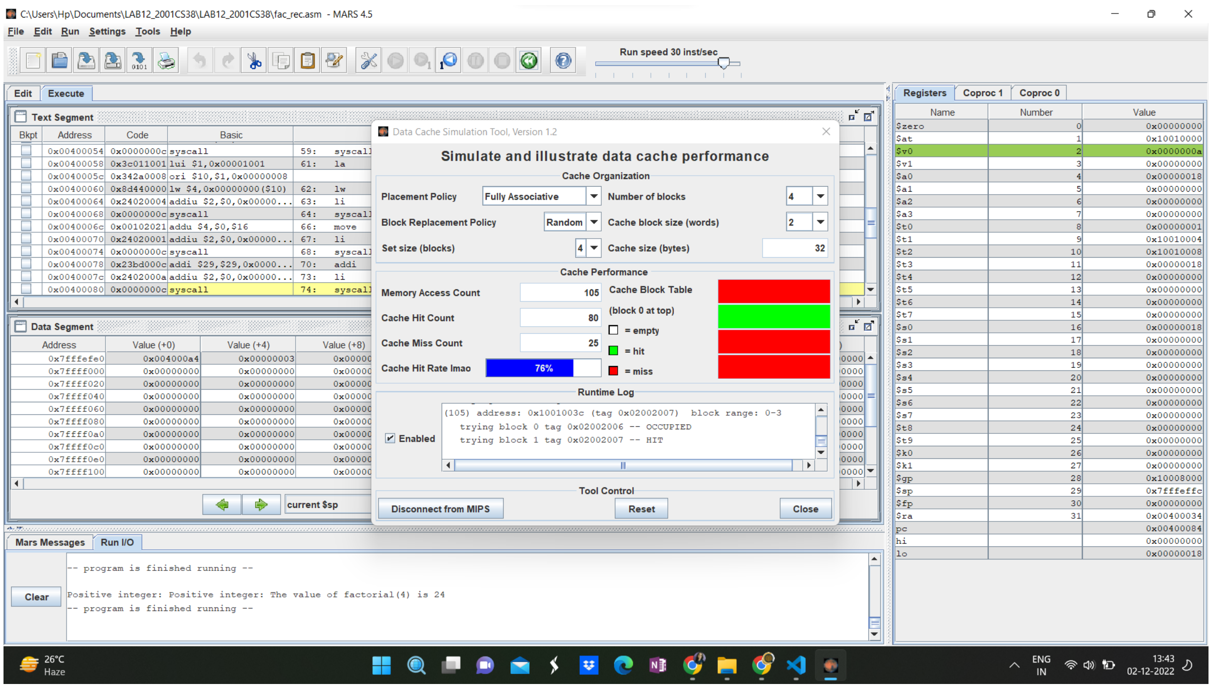
REPLACEMENT POLICY	HIT RATE
RANDOM	80%
FIFO	75%

LFU	75%
LRU	84%
LRU-2	84%

FIFO and LFU replacement policies have the worst performance. They have the least cache hit rate. And for FIFO as it is observed, the cache hit rate decreases when the number of blocks is increased. It can be attributed to the fact that FIFO sometimes replaces an active block and brings it back after two or three memory accesses. This takes many times, because it writes in cache and brings it back to main memory in two steps.
LRU is the better algorithm to implement in these conditions. The cache hit rate for LRU is even greater than Random.

FACTORIAL RECURSIVE CODE:

1. RANDOM:



C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fac_rec.asm - MARS 4.5

Text Segment

Bkpt	Address	Code	Basic	
	0x00400054	0x0000000c	syscall	59: syscall
	0x00400058	0x3c011001	lui $1,0x00001001	61: la
	0x0040005c	0x342a0008	ori $10,$1,0x00000008	
	0x00400060	0x8d440000	lw $4,0x00000000($10)	62: lw
	0x00400064	0x24020004	addiu $2,$0,0x00000...	63: li
	0x00400068	0x0000000c	syscall	64: syscall
	0x0040006c	0x00102021	addu $4,$0,$16	66: move
	0x00400070	0x24020001	addiu $2,$0,0x00000...	67: li
	0x00400074	0x0000000c	syscall	68: syscall
	0x00400078	0x23bd000c	addi $29,$29,0x0000...	70: addi
	0x0040007c	0x2402000a	addiu $2,$0,0x00000...	73: li
	0x00400080	0x0000000c	syscall	74: syscall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x7fffefe0	0x004000a4	0x00000003	0x00000
0x7ffff000	0x00000000	0x00000000	0x00000
0x7ffff020	0x00000000	0x00000000	0x00000
0x7ffff040	0x00000000	0x00000000	0x00000
0x7ffff060	0x00000000	0x00000000	0x00000
0x7ffff080	0x00000000	0x00000000	0x00000
0x7ffff0a0	0x00000000	0x00000000	0x00000
0x7ffff0c0	0x00000000	0x00000000	0x00000
0x7ffff0e0	0x00000000	0x00000000	0x00000
0x7ffff100	0x00000000	0x00000000	0x00000

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	Random	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	105
Cache Hit Count	80
Cache Miss Count	25
Cache Hit Rate Imao	76%

Cache Block Table (block 0 at top): 🟥 🟩 🟥 🟥

□ = empty, 🟩 = hit, 🟥 = miss

Runtime Log [✓ Enabled]

```
(105) address: 0x1001003c (tag 0x02002007)  block range: 0-3
   trying block 0 tag 0x02002006 -- OCCUPIED
   trying block 1 tag 0x02002007 -- HIT
```

Registers

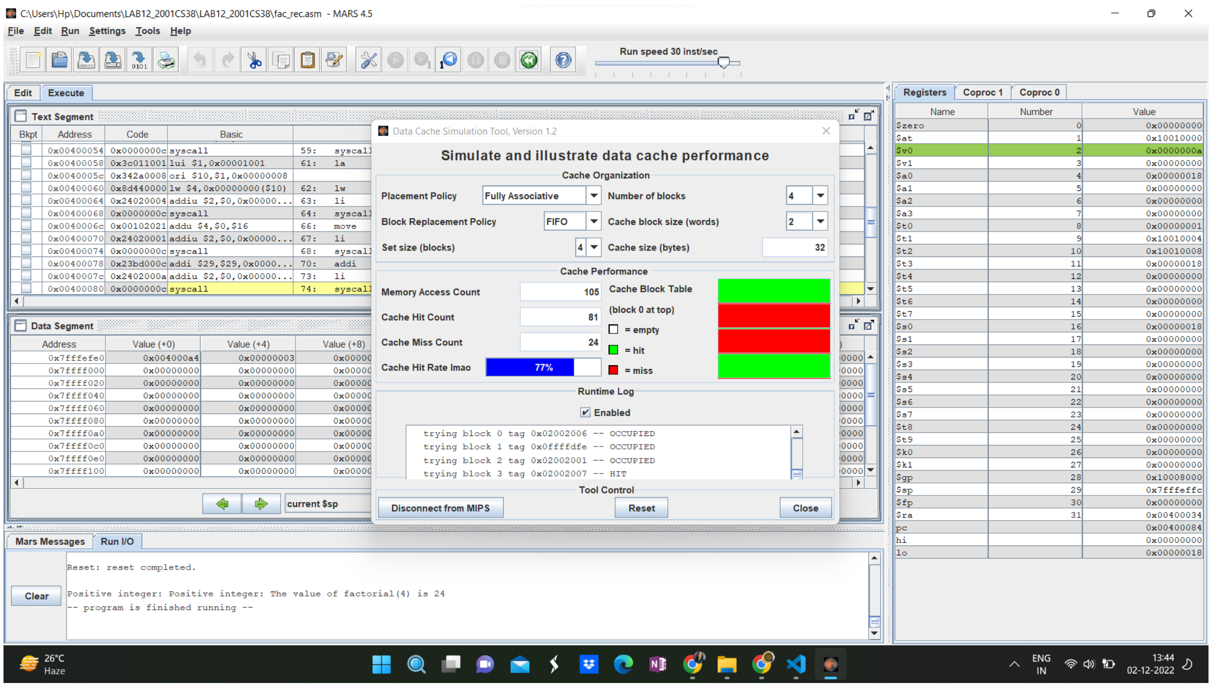
Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000018
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x10010004
$t2	10	0x10010008
$t3	11	0x00000018
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000018
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400034
pc		0x00400084
hi		0x00000000
lo		0x00000018

Run I/O

```
-- program is finished running --

Positive integer: Positive integer: The value of factorial(4) is 24
-- program is finished running --
```

2. FIFO:


C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fac_rec.asm - MARS 4.5

Text Segment

Bkpt	Address	Code	Basic	
	0x00400054	0x0000000c	syscall	59: syscall
	0x00400058	0x3c011001	lui $1,0x00001001	61: la
	0x0040005c	0x342a0008	ori $10,$1,0x00000008	
	0x00400060	0x8d440000	lw $4,0x00000000($10)	62: lw
	0x00400064	0x24020004	addiu $2,$0,0x00000...	63: li
	0x00400068	0x0000000c	syscall	64: syscall
	0x0040006c	0x00102021	addu $4,$0,$16	66: move
	0x00400070	0x24020001	addiu $2,$0,0x00000...	67: li
	0x00400074	0x0000000c	syscall	68: syscall
	0x00400078	0x23bd000c	addi $29,$29,0x0000...	70: addi
	0x0040007c	0x2402000a	addiu $2,$0,0x00000...	73: li
	0x00400080	0x0000000c	syscall	74: syscall

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x7fffefe0	0x004000a4	0x00000003	0x00000
0x7ffff000	0x00000000	0x00000000	0x00000
0x7ffff020	0x00000000	0x00000000	0x00000
0x7ffff040	0x00000000	0x00000000	0x00000
0x7ffff060	0x00000000	0x00000000	0x00000
0x7ffff080	0x00000000	0x00000000	0x00000
0x7ffff0a0	0x00000000	0x00000000	0x00000
0x7ffff0c0	0x00000000	0x00000000	0x00000
0x7ffff0e0	0x00000000	0x00000000	0x00000
0x7ffff100	0x00000000	0x00000000	0x00000

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	FIFO	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	105
Cache Hit Count	81
Cache Miss Count	24
Cache Hit Rate lmao	77%

Cache Block Table (block 0 at top): 🟩 🟥 🟥 🟩

□ = empty 🟩 = hit 🟥 = miss

Runtime Log

✓ Enabled

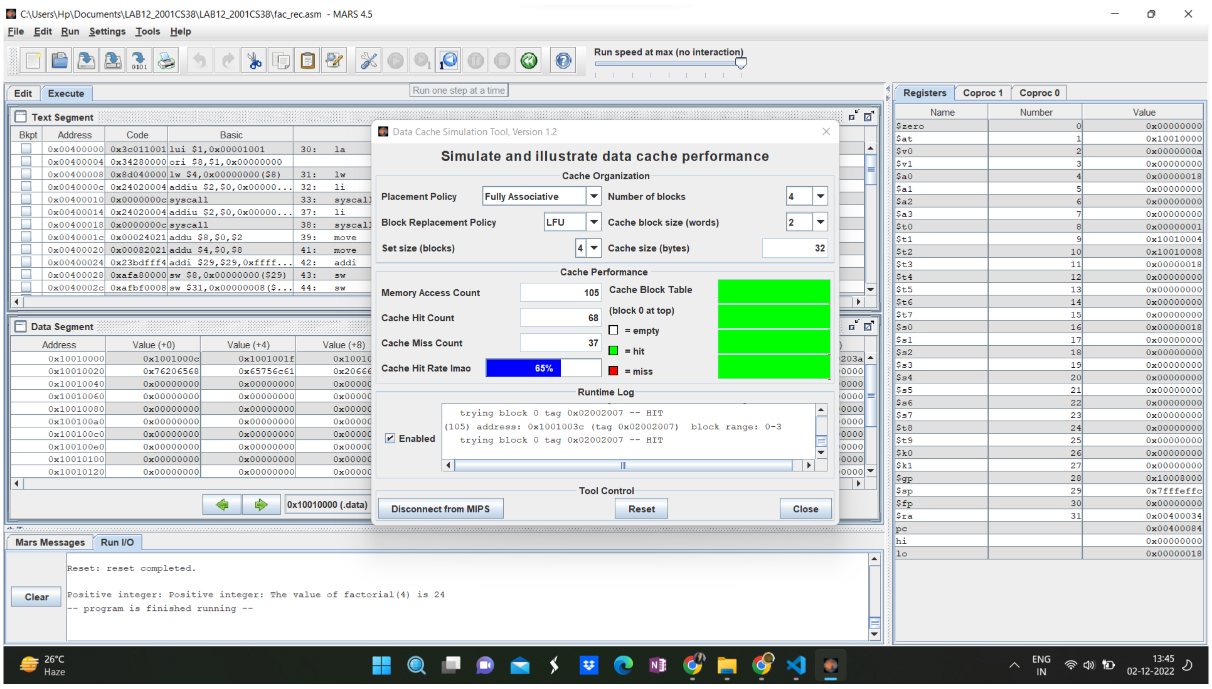
```
   trying block 0 tag 0x02002006 -- OCCUPIED
   trying block 1 tag 0x0ffffdfe -- OCCUPIED
   trying block 2 tag 0x02002001 -- OCCUPIED
   trying block 3 tag 0x02002007 -- HIT
```

```
Reset: reset completed.

Positive integer: Positive integer: The value of factorial(4) is 24
-- program is finished running --
```

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000018
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x10010004
$t2	10	0x10010008
$t3	11	0x00000018
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000018
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400034
pc		0x00400084
hi		0x00000000
lo		0x00000018

3. LFU:


C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fac_rec.asm - MARS 4.5

File Edit Run Settings Tools Help
Run speed at max (no interaction)
Run one step at a time
Edit
Execute

Text Segment

Bkpt	Address	Code	Basic	
	0x00400000	0x3c011001	lui $1,0x00001001	30: la
	0x00400004	0x34280000	ori $8,$1,0x00000000	
	0x00400008	0x8d040000	lw $4,0x00000000($8)	31: lw
	0x0040000c	0x24020004	addiu $2,$0,0x00000...	32: li
	0x00400010	0x0000000c	syscall	33: syscall
	0x00400014	0x24020004	addiu $2,$0,0x00000...	37: li
	0x00400018	0x0000000c	syscall	38: syscall
	0x0040001c	0x00024021	addu $8,$0,$2	39: move
	0x00400020	0x00082021	addu $4,$0,$8	41: move
	0x00400024	0x23bdfff4	addi $29,$29,0xffff...	42: addi
	0x00400028	0xafa80000	sw $8,0x00000000($29)	43: sw
	0x0040002c	0xafbf0008	sw $31,0x00000008($...	44: sw

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)
0x10010000	0x1001000c	0x1001001f	0x10010
0x10010020	0x76206568	0x65756c61	0x20666
0x10010040	0x00000000	0x00000000	0x00000
0x10010060	0x00000000	0x00000000	0x00000
0x10010080	0x00000000	0x00000000	0x00000
0x100100a0	0x00000000	0x00000000	0x00000
0x100100c0	0x00000000	0x00000000	0x00000
0x100100e0	0x00000000	0x00000000	0x00000
0x10010100	0x00000000	0x00000000	0x00000
0x10010120	0x00000000	0x00000000	0x00000

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	LFU	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	105
Cache Hit Count	68
Cache Miss Count	37
Cache Hit Rate lmao	65%

Cache Block Table (block 0 at top): 🟩 🟩 🟩 🟩

□ = empty 🟩 = hit 🟥 = miss

Runtime Log

✓ Enabled

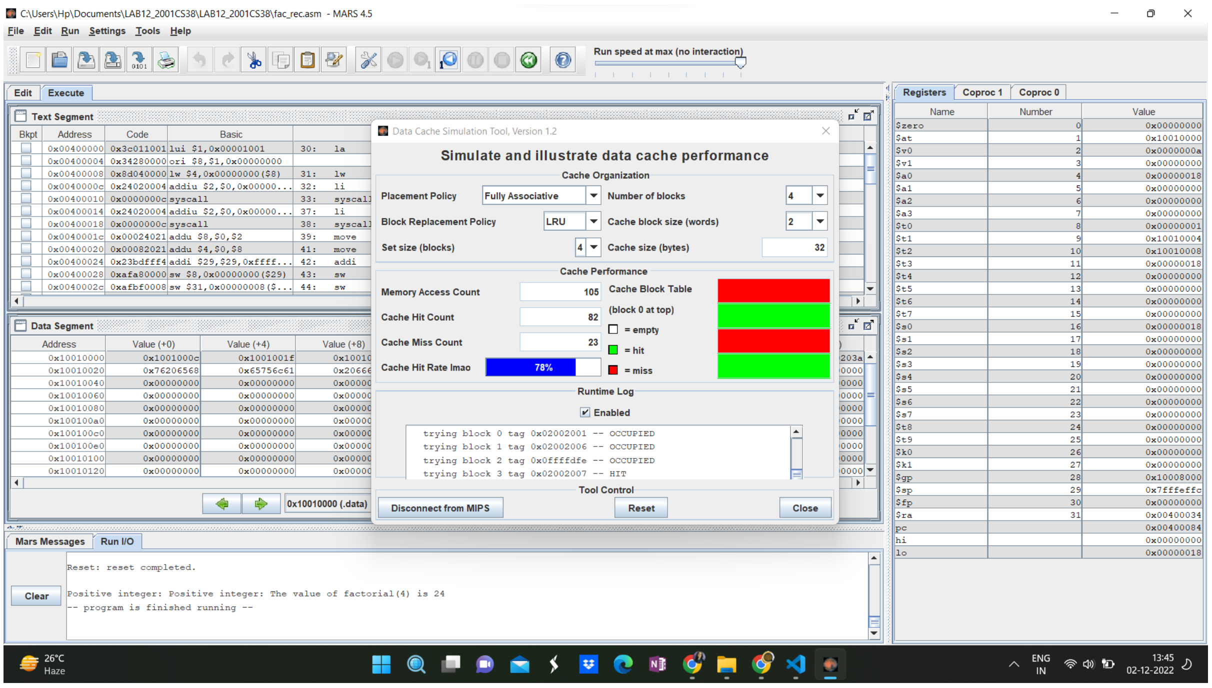
```
   trying block 0 tag 0x02002007 -- HIT
(105) address: 0x1001003c (tag 0x02002007)  block range: 0-3
   trying block 0 tag 0x02002007 -- HIT
```

```
Reset: reset completed.

Positive integer: Positive integer: The value of factorial(4) is 24
-- program is finished running --
```

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000018
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x10010004
$t2	10	0x10010008
$t3	11	0x00000018
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000018
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400034
pc		0x00400084
hi		0x00000000
lo		0x00000018

4. LRU:



C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fac_rec.asm - MARS 4.5

Run speed at max (no interaction)

Text Segment

```
Bkpt Address    Code       Basic
     0x00400000 0x3c011001 lui $1,0x00001001              30:  la
     0x00400004 0x34280000 ori $8,$1,0x00000000
     0x00400008 0x8d040000 lw $4,0x00000000($8)           31:  lw
     0x0040000c 0x24020004 addiu $2,$0,0x00000...         32:  li
     0x00400010 0x0000000c syscall                        33:  syscall
     0x00400014 0x24020004 addiu $2,$0,0x00000...         37:  li
     0x00400018 0x0000000c syscall                        38:  syscall
     0x0040001c 0x00024021 addu $8,$0,$2                  39:  move
     0x00400020 0x00082021 addu $4,$0,$8                  41:  move
     0x00400024 0x23bdfff4 addi $29,$29,0xffff...         42:  addi
     0x00400028 0xafa80000 sw $8,0x00000000($29)          43:  sw
     0x0040002c 0xafbf0008 sw $31,0x00000008($...         44:  sw
```

Data Segment

```
Address     Value (+0)  Value (+4)  Value (+8)
0x10010000  0x1001000c  0x1001001f  0x10010
0x10010020  0x76206568  0x65756c61  0x20666
0x10010040  0x00000000  0x00000000  0x00000
0x10010060  0x00000000  0x00000000  0x00000
0x10010080  0x00000000  0x00000000  0x00000
0x100100a0  0x00000000  0x00000000  0x00000
0x100100c0  0x00000000  0x00000000  0x00000
0x100100e0  0x00000000  0x00000000  0x00000
0x10010100  0x00000000  0x00000000  0x00000
0x10010120  0x00000000  0x00000000  0x00000
```

0x10010000 (.data)

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	LRU	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	105
Cache Hit Count	82
Cache Miss Count	23
Cache Hit Rate Imao	78%

Cache Block Table (block 0 at top): 🟥 🟩 🟥 🟩

☐ = empty 🟩 = hit 🟥 = miss

Runtime Log

☑ Enabled

```
  trying block 0 tag 0x02002001 -- OCCUPIED
  trying block 1 tag 0x02002006 -- OCCUPIED
  trying block 2 tag 0x0ffffdfe -- OCCUPIED
  trying block 3 tag 0x02002007 -- HIT
```

Tool Control

Registers

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000018
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x10010004
$t2	10	0x10010008
$t3	11	0x00000018
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000018
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400034
pc		0x00400084
hi		0x00000000
lo		0x00000018

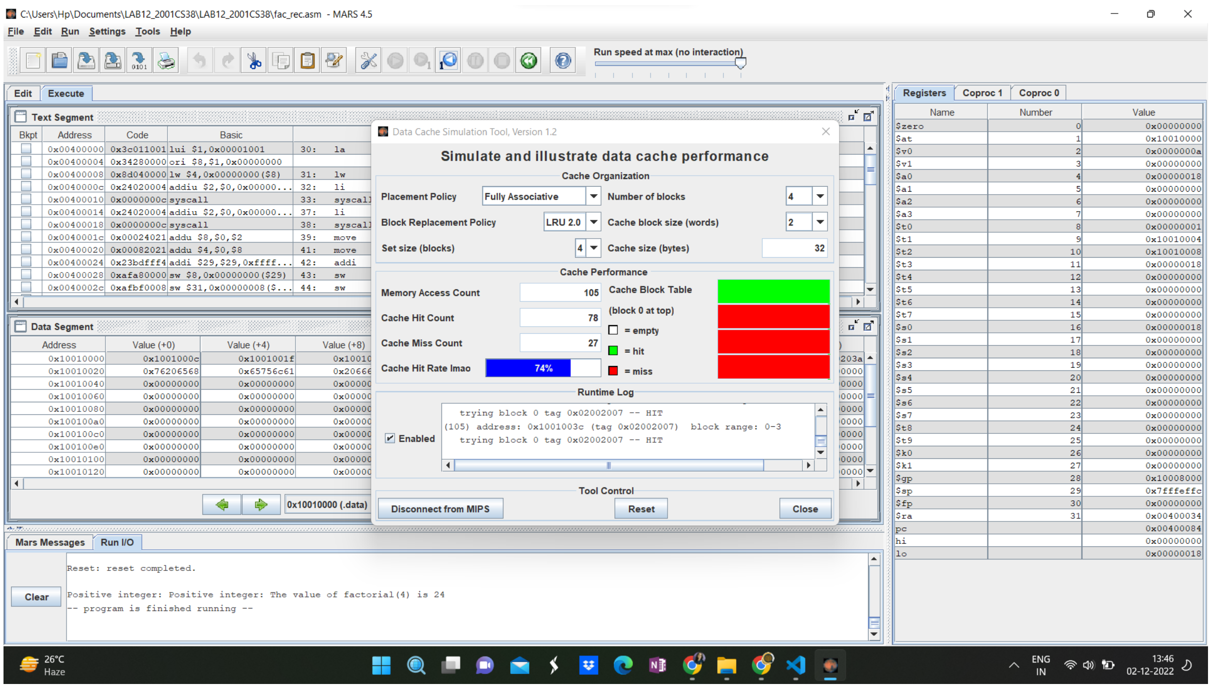
Mars Messages / Run I/O

```
Reset: reset completed.

Positive integer: Positive integer: The value of factorial(4) is 24
-- program is finished running --
```

26°C Haze — ENG IN 13:45 02-12-2022

5. LRU 2:



C:\Users\Hp\Documents\LAB12_2001CS38\LAB12_2001CS38\fac_rec.asm - MARS 4.5

Run speed at max (no interaction)

Text Segment

```
Bkpt Address    Code       Basic
     0x00400000 0x3c011001 lui $1,0x00001001              30:  la
     0x00400004 0x34280000 ori $8,$1,0x00000000
     0x00400008 0x8d040000 lw $4,0x00000000($8)           31:  lw
     0x0040000c 0x24020004 addiu $2,$0,0x00000...         32:  li
     0x00400010 0x0000000c syscall                        33:  syscall
     0x00400014 0x24020004 addiu $2,$0,0x00000...         37:  li
     0x00400018 0x0000000c syscall                        38:  syscall
     0x0040001c 0x00024021 addu $8,$0,$2                  39:  move
     0x00400020 0x00082021 addu $4,$0,$8                  41:  move
     0x00400024 0x23bdfff4 addi $29,$29,0xffff...         42:  addi
     0x00400028 0xafa80000 sw $8,0x00000000($29)          43:  sw
     0x0040002c 0xafbf0008 sw $31,0x00000008($...         44:  sw
```

Data Segment

```
Address     Value (+0)  Value (+4)  Value (+8)
0x10010000  0x1001000c  0x1001001f  0x10010
0x10010020  0x76206568  0x65756c61  0x20666
0x10010040  0x00000000  0x00000000  0x00000
0x10010060  0x00000000  0x00000000  0x00000
0x10010080  0x00000000  0x00000000  0x00000
0x100100a0  0x00000000  0x00000000  0x00000
0x100100c0  0x00000000  0x00000000  0x00000
0x100100e0  0x00000000  0x00000000  0x00000
0x10010100  0x00000000  0x00000000  0x00000
0x10010120  0x00000000  0x00000000  0x00000
```

0x10010000 (.data)

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	Fully Associative	Number of blocks	4
Block Replacement Policy	LRU 2.0	Cache block size (words)	2
Set size (blocks)	4	Cache size (bytes)	32

Cache Performance

Memory Access Count	105
Cache Hit Count	78
Cache Miss Count	27
Cache Hit Rate Imao	74%

Cache Block Table (block 0 at top): 🟩 🟥 🟥 🟥

☐ = empty 🟩 = hit 🟥 = miss

Runtime Log

☑ Enabled

```
  trying block 0 tag 0x02002007 -- HIT
(105) address: 0x1001003c (tag 0x02002007)  block range: 0-3
  trying block 0 tag 0x02002007 -- HIT
```

Tool Control

Registers

Name	Number	Value
$zero	0	0x00000000
$at	1	0x10010000
$v0	2	0x0000000a
$v1	3	0x00000000
$a0	4	0x00000018
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000001
$t1	9	0x10010004
$t2	10	0x10010008
$t3	11	0x00000018
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000018
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00400034
pc		0x00400084
hi		0x00000000
lo		0x00000018

Mars Messages / Run I/O

```
Reset: reset completed.

Positive integer: Positive integer: The value of factorial(4) is 24
-- program is finished running --
```

26°C Haze — ENG IN 13:46 02-12-2022

ANALYSIS:

REPLACEMENT POLICY	HIT RATE
RANDOM	76%

FIFO	77%
LFU	65%
LRU	74%
LRU-2	74%

As observed, LFU has the least Cache Hit rate out of all Replacement Policies. In the above program, the smaller numbers will be used more times as compared to the larger numbers. Thus, in case of a full set, the LFU replacement policy will replace the block containing the larger numbers (of the numbers stored in the cache). But these numbers will also be required further in computation. This will cause an increase in Miss Rate.
In this case, FIFO performs the best and has a Cache HIt Rate greater than the Random Cache Hit Rate. FIFO stores the number nearest to the argument in cache and thus reduces miss rate.

CONCLUSION:

Thus, from our observation of the above two recursive codes, we can summarise the replacement policies as under:

REPLACEMENT POLICY	**COMMENT**
RANDOM	
FIFO	Might replace important blocks
LFU	Not Efficient
LRU	Excellent, but difficult to implement, as it requires substantial hardware assistance
LRU-2	Efficiency increases as the size of cache increases

Video Link:
https://drive.google.com/drive/folders/1VM16_2UeBGkK9XlCCpzFHpE3b209zsu1?usp=sharing