

Laboratory Exercise 12

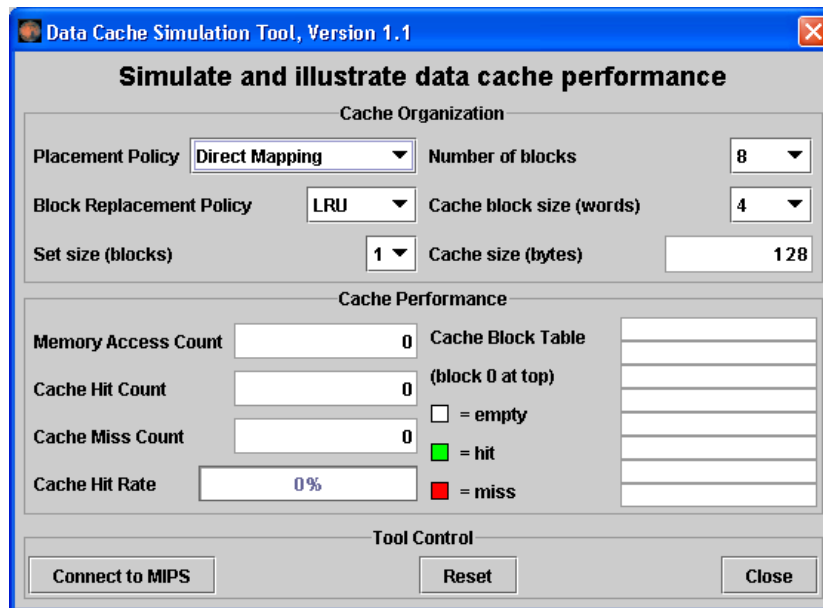
Cache Memory

PHẠM VÂN ANH 20214988

Running the Data Cache Simulator tool



1. Close any MIPS programs you are currently using.
2. Open the program **row-major.asm** from the Examples folder. This program will traverse a 16 by 16 element integer matrix in row-major order, assigning elements the values 0 through 255 in order. It performs the following algorithm:

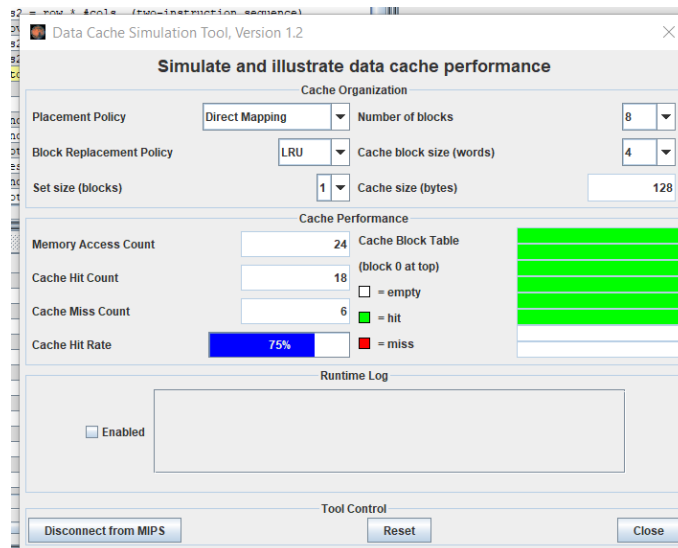
```
for (row = 0; row < 16; row++)  
    for (col = 0; col < 16; col++)  
        data[row][col] = value++;
```
3. Assemble the program.
4. From the **Tools** menu, select **Data Cache Simulator**. A new frame will appear in the middle of the screen.



This is a MARS Tool that will simulate the use and performance of cache memory when the underlying MIPS program executes. Notice its three major sections:

- *Cache Organization:* You can use the combo boxes to specify how the cache will be configured for this run. Feel free to explore the different settings, but the default is fine for now.

- *Cache Performance:* With each memory access during program execution, the simulator will determine whether or not that access can be satisfied from cache and update the performance display accordingly.
 - *Tool Control:* These buttons perform generic control functions as described by their labels.
5. Click the tool's **Connect to MIPS** button. This causes the tool to register as an observer of MIPS memory and thus respond during program execution.
 6. Back in MARS, adjust the **Run Speed slider** to 30 instructions per second. It is located at the right side of the toolbar. This slows execution so you can watch the Cache Performance animation.
- 
7. In MARS, run the program using the **Run** toolbar button , the menu item or keyboard shortcut. Watch the Cache Performance animate as it is updated with every access to MIPS memory.
 8. *What was the final cache hit rate?* _____. With each miss, a block of 4 words are written into the cache. In a row-major traversal, matrix elements are accessed in the same order they are stored in memory. Thus each cache miss is followed by 3 hits as the next 3 elements are found in the same cache block. This is followed by another miss when Direct Mapping maps to the next cache block, and the patterns repeats itself. So 3 of every 4 memory accesses will be resolved in cache.
 - 75%. Vì mỗi Cache có 4 phần tử, khi truy cập phần tử đầu tiên của khối, ta sẽ kiểm tra trong bộ nhớ Cache, lúc này trong Cache chưa có dữ liệu, nên sẽ đọc dữ liệu trong Bộ nhớ chính, sau đó được chuyển vào bộ nhớ Cache. Nên phần tử đầu tiên của Cache sẽ trống, 3 phần tử tiếp theo lần lượt được đọc trong Cache → Hiệu suất là $3/4 = 75\%$



9. Given that explanation, *what do you predict the hit rate will be if the block size is increased from 4 words to 8 words?* _____. *Decreased from 4 words to 2 words?* _____.

- From 4 words to 8 words: 87,5%. Vì mỗi Cache bây giờ có 8 phần tử, khi truy cập phần tử đầu tiên của khối, ta sẽ kiểm tra trong bộ nhớ Cache, lúc này trong Cache chưa có dữ liệu, nên sẽ đọc dữ liệu trong Bộ nhớ chính, sau đó được chuyển vào bộ nhớ Cache. Nên phần tử đầu tiên của Cache sẽ trống, 7 phần tử tiếp theo lần lượt được đọc trong Cache → Hiệu suất là $7/8 = 87,5\%$

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping (dropdown) Number of blocks: 8 (dropdown)

Block Replacement Policy: LRU (dropdown) Cache block size (words): 8 (dropdown)

Set size (blocks): 1 (dropdown) Cache size (bytes): 256 (text box)

Cache Performance

Memory Access Count: 16 (text box) Cache Block Table (block 0 at top)

Cache Hit Count: 14 (text box) ☐ = empty

Cache Miss Count: 2 (text box) ☒ = hit

Cache Hit Rate: 88% (progress bar) ☐ = miss

Runtime Log

☐ Enabled

Tool Control

Disconnect from MIPS (button) Reset (button) Close (button)

- From 4 words to 2 words: 50%. Vì mỗi Cache bây giờ có 2 phần tử, khi truy cập phần tử đầu tiên của khối, ta sẽ kiểm tra trong bộ nhớ Cache, lúc này trong Cache chưa có dữ liệu, nên sẽ đọc dữ liệu trong Bộ nhớ chính, sau đó được chuyển vào bộ nhớ Cache. Nên phần tử đầu tiên của Cache sẽ trống, phần tử tiếp theo lần lượt được đọc trong Cache → Hiệu suất là $1/2 = 50\%$

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping Number of blocks: 8

Block Replacement Policy: LRU Cache block size (words): 2

Set size (blocks): 1 Cache size (bytes): 64

Cache Performance

Memory Access Count: 8 Cache Block Table (block 0 at top):

Cache Hit Count: 4

Cache Miss Count: 4

Cache Hit Rate: 50%

☐ = empty ☒ = hit ☐ = miss

Runtime Log

☐ Enabled

Tool Control

Disconnect from MIPS Reset Close

10. Verify your predictions by modifying the block size and re-running the program from step 7.

NOTE: when you modify the Cache Organization, the performance values are automatically reset (you can always use the tool's **Reset** button).

NOTE: You have to **reset**  the MIPS program before you can re-run it.

NOTE: Feel free to adjust the **Run Speed slider** to maximum speed anytime you want.

- Hiệu suất là 93,75%

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping Number of blocks: 8

Block Replacement Policy: LRU Cache block size (words): 16

Set size (blocks): 1 Cache size (bytes): 512

Cache Performance

Memory Access Count: 16 Cache Block Table (block 0 at top):

Cache Hit Count: 15

Cache Miss Count: 1

Cache Hit Rate: 94%

☐ = empty ☒ = hit ☐ = miss

Runtime Log

☐ Enabled

Tool Control

Disconnect from MIPS Reset Close

11. Repeat steps 2 through 10 for program **column-major.asm** from the Examples folder. This program will traverse a 16 by 16 element integer matrix in column-major order, assigning elements the values 0 through 255 in order. It performs the following algorithm:

```
for (col = 0; col < 16; col++)
    for (row = 0; row < 16; row++)
        data[row][col] = value++;
```

NOTE: You can leave the Cache Simulator in place, move it out of the way, or close it. It will not interfere with the actions needed to open, assemble, or run this new program and will remain connected to MIPS memory. If you do not close the tool, then skip steps 4 and 5.

```
30      .data
31 data:  .word    0 : 256      # storage for 16x16 matrix of words
32      .text
33      li        $t0, 16      # $t0 = number of rows
34      li        $t1, 16      # $t1 = number of columns
35      move      $s0, $zero    # $s0 = row counter
36      move      $s1, $zero    # $s1 = column counter
37      move      $t2, $zero    # $t2 = the value to be stored
38 # Each loop iteration will store incremented $t1 value into next element of matrix.
39 # Offset is calculated at each iteration. offset = 4 * (row*cols+col)
40 # Note: no attempt is made to optimize runtime performance!
41 loop:  mult     $s0, $t0      # $s2 = row * #cols (two-instruction sequence)
42        mflo     $s2          # move multiply result from lo register to $s2
43        add     $s2, $s2, $s1  # $s2 += column counter
44        sll     $s2, $s2, 2    # $s2 *= 4 (shift left 2 bits) for byte offset
45        sw      $t2, data($s2) # store the value in matrix element
46        addi    $t2, $t2, 1    # increment value to be stored
47 # Loop control: If we increment past last column, reset column counter and increment row counter
48 #               If we increment past last row, we're finished.
49        addi    $s0, $s0, 1    # increment column counter
50        bne     $s0, $t0, loop # not at end of row so loop back
51        move    $s0, $zero     # reset column counter
52        addi    $s1, $s1, 1    # increment row counter
53        bne     $s1, $t1, loop # not at end of matrix so loop back
54 # We're finished traversing the matrix.
55        li      $v0, 10        # system service 10 is exit
56        syscall                # we are outta here.
57
```

Code sau khi sửa đọc theo cột

12. What was the cache performance for this program? ____0%____. The problem is the memory locations are now accessed not sequentially as before, but each access is 16 words beyond the previous one (circularly). With the settings we've used, no two consecutive memory accesses occur in the same block so every access is a miss.
- Khi đọc dữ liệu của mảng theo cột, 2 phần tử liên tiếp trong cột sẽ cách nhau 16 words. Khi đọc phần tử đầu tiên cache chưa có dữ liệu, nên sẽ lấy dữ liệu tại bộ nhớ chính (main) và lưu thành khối vào cache, phần tử tiếp theo do cách nhau 16 words nên trong bộ nhớ cache khi này sẽ chưa có và tiếp tục vào bộ nhớ chính lấy dữ liệu. Do kích thước khối không đủ lớn nên dữ liệu sau sẽ được ghi đè lên
→ Hiệu suất hit là 0%.

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping Number of blocks: 8

Block Replacement Policy: LRU Cache block size (words): 4

Set size (blocks): 1 Cache size (bytes): 128

Cache Performance

Memory Access Count: 256 Cache Block Table (block 0 at top)

Cache Hit Count: 0

Cache Miss Count: 256

Cache Hit Rate: 0%

☐ = empty ☐ = hit ☐ = miss

Runtime Log

☐ Enabled

Tool Control

Disconnect from MIPS Reset Close

13. Change the block size to 16. Note this will reset the tool.

- Với kích thước khối là 16 - lưu cả hàng của ma trận vào khối.
 - Với lần truy cập đầu tiên, không có dữ liệu trong cache, nên truy cập ra bộ nhớ chính (main) để lấy và lưu dữ liệu theo khối vào cache.
 - Phần tử tiếp theo sw là cách 16 words (hàng mới) → chưa được lưu trong cache, tiếp tục truy cập vào bộ nhớ chính. Sau khi ghi đầy 8 khối, cache sẽ bắt đầu ghi đè, nên việc hiệu suất hit thành công là 0%.

Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping Number of blocks: 8

Block Replacement Policy: LRU Cache block size (words): 16

Set size (blocks): 1 Cache size (bytes): 512

Cache Performance

Memory Access Count: 256 Cache Block Table (block 0 at top)

Cache Hit Count: 0

Cache Miss Count: 256

Cache Hit Rate: 0%

☐ = empty ☐ = hit ☐ = miss

Runtime Log

☐ Enabled

Tool Control

Disconnect from MIPS Reset Close

14. Create a second instance of the Cache Simulator by once again selecting **Data Cache Simulator** from the **Tools** menu. Adjust the two frames so you can view both at the same time. Connect the new tool instance to MIPS, change its block size to 16 and change its number of blocks to 16.

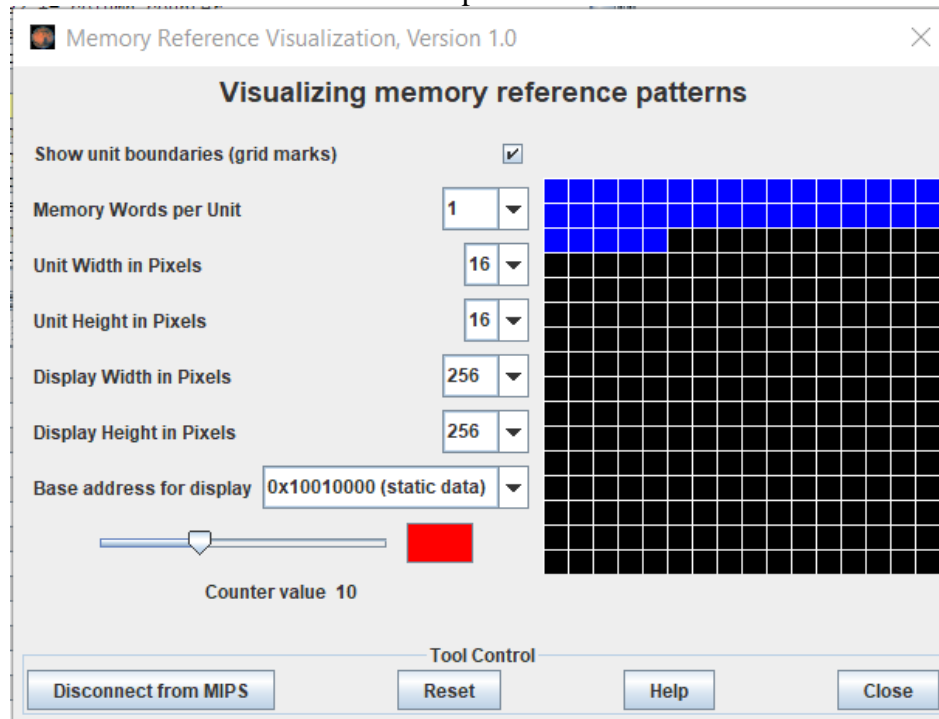
- Hiệu suất là $15/16 = 93.75\%$
- Do sau khi cột đầu tiên được thực hiện (hiệu suất là 0%), cache sẽ lưu toàn bộ dữ liệu của ma trận (tương đương ma trận trong main). Vì vậy các lần truy cập sau đều đã có sẵn trong cache nên tỷ lệ hit lúc này là 100%. Tổng kết lại ta thu được hiệu suất : $(256-16)/256 = 15/16$.

15. Re-run the program. *What is the cache performance of the original tool instance?* ____0%_____. Block size 16 didn't help because there was still only one access to each block, the initial miss, before that block was replaced with a new one. *What is the cache performance of the second tool instance?* ____94%_____. At this point, the entire matrix will fit into cache and so once a block is read in it is never replaced. Only the first access to a block results in a miss.

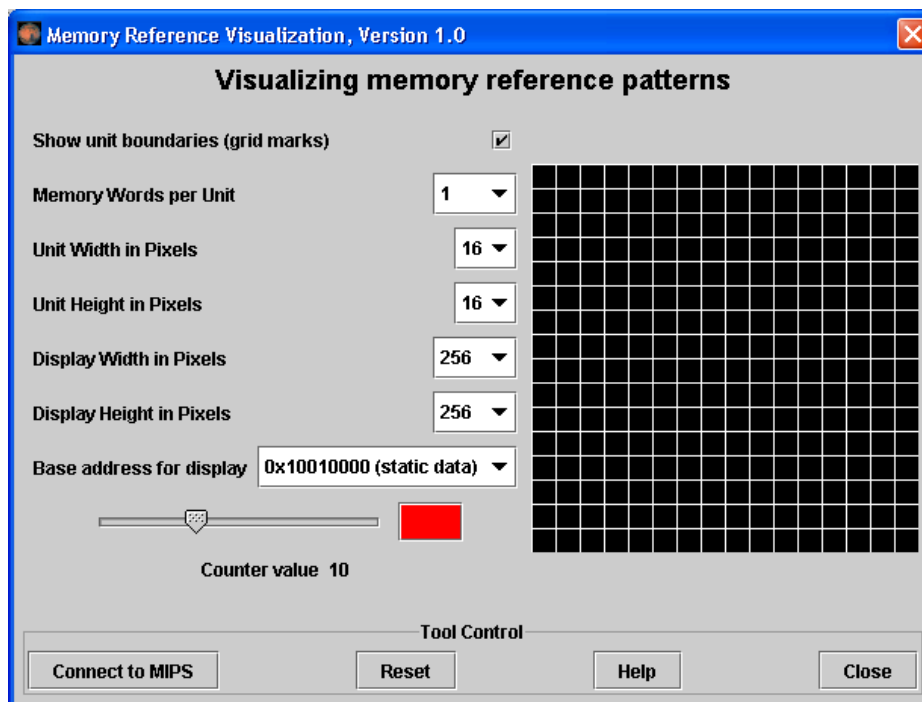
- Chương trình thứ 1, kết quả thu được là 0%, vì lần truy cập đầu tiên chưa có dữ liệu nên vào Cache. Tiếp tục như vậy, nhưng do số lượng block không đủ lớn nên cache sẽ ghi đè, làm mất dữ liệu trước.
- Trong khi chương trình 2 hiệu suất là 94%, vì chỉ sau lần duyệt cột đầu tiên toàn bộ dữ liệu từ bộ nhớ đã có trong cache. Nên các cột tiếp theo truy cập hoàn toàn tại Cache.

The Memory Reference Visualization tool

1. Open the program **row-major.asm** from the **Examples** folder if it is not already open.



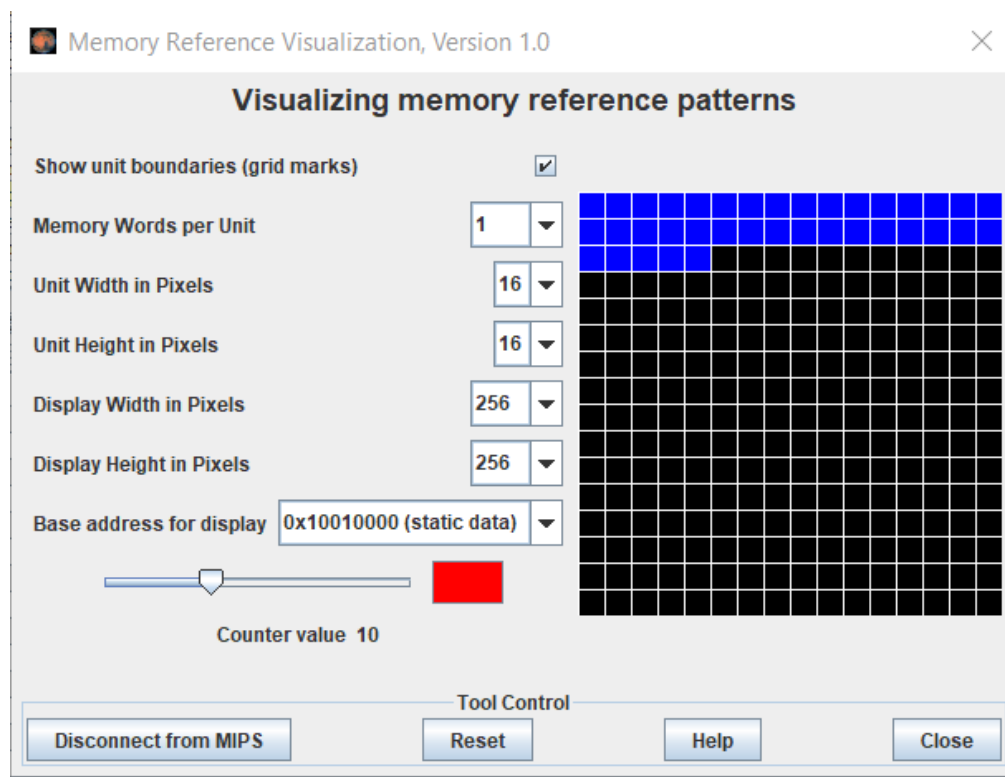
2. Assemble the program.
3. From the **Tools** menu, select **Memory Reference Visualization**. A new frame will appear in the middle of the screen.

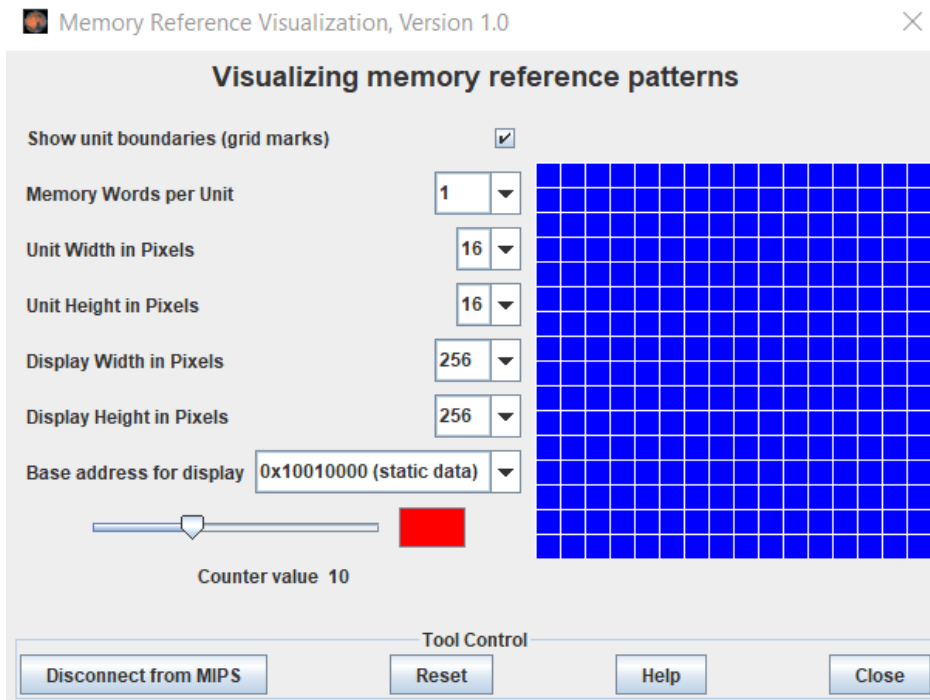


This tool will paint a grid unit each time the corresponding MIPS memory word is referenced. The base address, the first static data segment (.data directive) word, corresponds to the upper-left grid unit. Address correspondence continues in row-major order (left to right, then next row down).

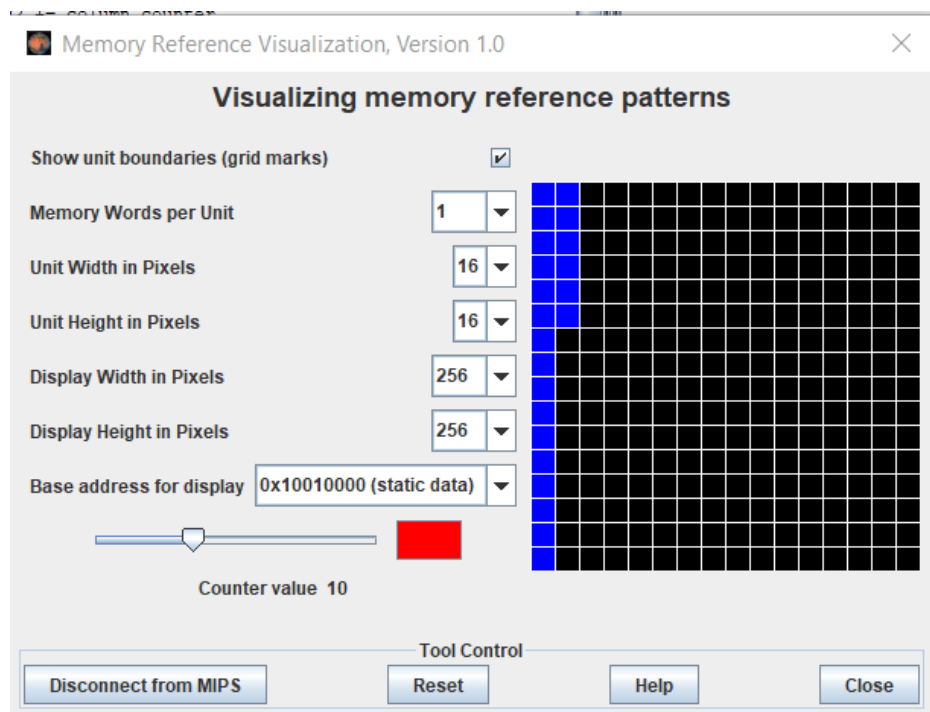
The color depends on the number of times the word has been referenced. Black is 0, blue is 1, green is 2, yellow is 3 and 4, orange is 5 through 9, red is 10 or higher. View the scale using the tool's slider control. You can change the color (but not the reference count) by clicking on the color patch.

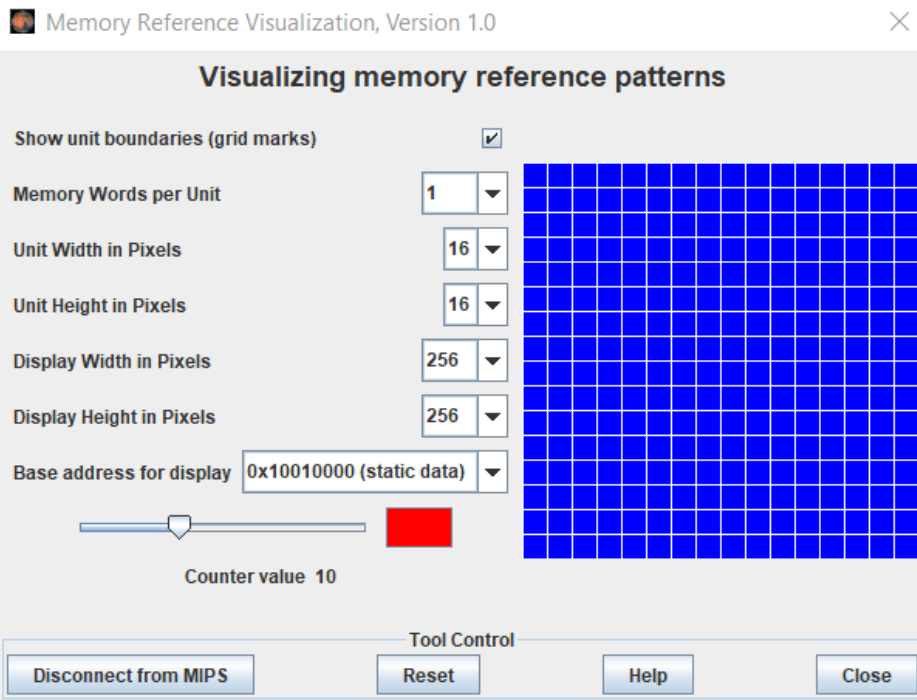
4. Click the tool's **Connect to MIPS** button. This causes the tool to register as an observer of MIPS memory and thus respond during program execution.
5. Back in MARS, adjust the **Run Speed slider** to 30 instructions per second.
6. Run the program. Watch the tool animate as it is updated with every access to MIPS memory. *Feel free to stop the program at any time.*
7. Hopefully you observed that the animation sequence corresponded to the expected memory access sequence of the row-major.asm program. *If you have trouble seeing the blue*, reset the tool, move the slider to position 1, change the color to something brighter, and re-run.





8. Repeat steps 2 through 7, for **column-major.asm**. You should observe that the animation sequence corresponded to the expected memory access sequence of this program.

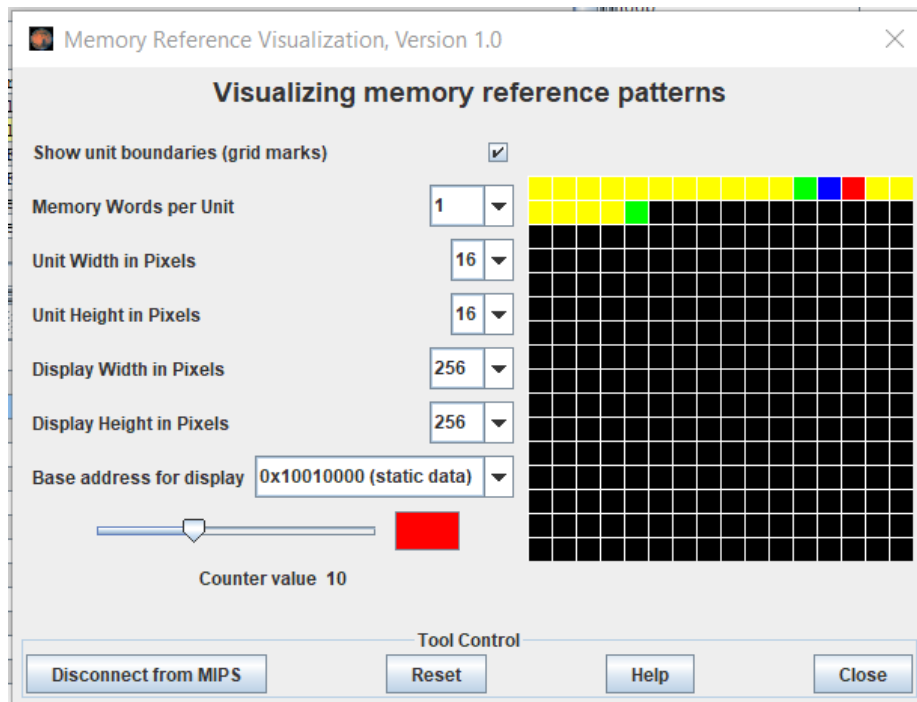




9. Repeat again for **fibonacci.asm** to observe the animated pattern of memory references. Adjust the run speed and re-run if necessary.

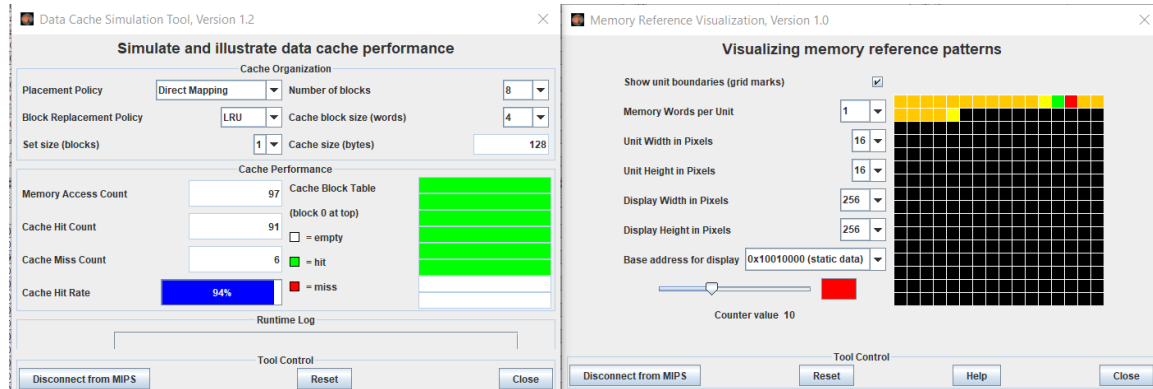
- Số lần truy cập – số màu tương ứng:

1	2	3, 4	5->9	10+



10. (Optional) Create a new instance of the Data Cache Simulator. Move the two frames around so you can see both. Connect the cache simulator to MIPS and reset the Memory Reference Visualization. Re-run the program. This exercise illustrates that two different tools can be used simultaneously.

The Memory Reference Visualization tool could be useful in an operating systems course to illustrate spatial and temporal locality and memory reference patterns in general.



- Data Cache Simulator: thể hiện sự tương tác và hiệu suất của bộ nhớ Cache dữ liệu
- Memory Reference Visualization biểu thị làm nổi bật các vị trí bộ nhớ được truy cập và tần suất truy cập