

# Báo cáo thực hành KTMT tuần 12

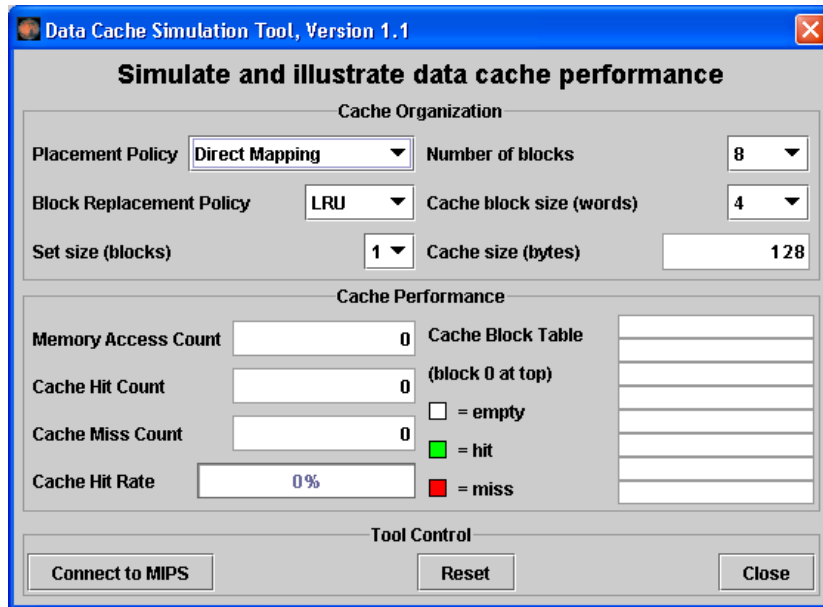
## Phạm Thành Lập-20215076

### Assign 1:

#### 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:  

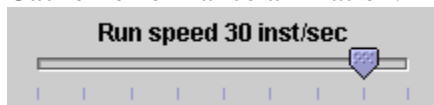
```
for (row = 0; row < 16; row++)  
    for (col = 0; col < 16; col++)  
        data[row][col] = value++;
```
3. Assemble the program.
4. From the **Tools** menu, select **Data Cache Simulator**. A new frame will appear in the middle of the screen.




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? 75%. 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.

- Giải thích: Vì blocksize là 4 (words) nên khi truyền vào cache sẽ là 4 phần tử của mảng theo hàng ngang, do khi gặp phần tử đầu tiên trong mảng thì phần tử đó chưa có trong cache nên sẽ bị đánh miss và lưu 1 block gồm 4 phần tử liên tiếp vào mảng, do duyệt theo hàng ngang lên sau khi truyền 4 phần tử vào thì khi gặp 3 phần tử tiếp theo sẽ được đánh hit do đã tồn tại, cứ tiếp tục như vậy cho đến khi duyệt hết ma trận vì vậy lên tỉ lệ là  $\frac{3}{4} = 75\%$

**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): **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: **192**

Cache Miss Count: **64**

Cache Hit Rate: **75%**

☐ = empty ☒ = hit ☐ = miss

**Runtime Log**

☐ Enabled

**Tool Control**

**Disconnect from MIPS** **Reset** **Close**

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?* **88%**.

- Giải thích: Vì blocksize là 8 (words) nên khi truyền vào cache sẽ là 8 phần tử của mảng theo hàng ngang, do khi gặp phần tử đầu tiên trong mảng thì phần tử đó chưa có trong cache nên sẽ bị đánh miss và lưu 1 block gồm 8 phần tử liên tiếp vào mảng, do duyệt theo hàng ngang lên sau khi truyền 8 phần tử vào thì khi gặp 7 phần tử tiếp theo sẽ được đánh hit do đã tồn tại, cứ tiếp tục như vậy cho đến khi duyệt hết ma trận vì vậy lên tỉ lệ là  $\frac{7}{8} = 87.5\%$  làm tròn thành 88%

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)

8

▼

Set size (blocks)

1

▼

Cache size (bytes)

256

Cache Performance

Memory Access Count

256

Cache Hit Count

224

Cache Miss Count

32

Cache Hit Rate

88%

Cache Block Table

(block 0 at top)

☐ = empty

☒ = hit

☐ = miss

Runtime Log

☐ Enabled

Tool Control

Disconnect from MIPS

Reset

Close

Page

Decreased from 4 words to 2 words? 50%.

- Giải thích: Vì blocksize là 2 (words) nên khi truyền vào cache sẽ là 2 phần tử của mảng theo hàng ngang, do khi gặp phần tử đầu tiên trong mảng thì phần tử đó chưa có trong cache nên sẽ bị đánh miss và lưu 1 block gồm 2 phần tử liên tiếp nhau vào mảng, do duyệt theo hàng ngang lên sau khi truyền 2 phần tử vào thì khi gặp phần tử tiếp theo sẽ được đánh hit do đã tồn tại, cứ tiếp tục như vậy cho đến khi duyệt hết ma trận vì vậy lên tỉ lệ 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: 256 Cache Block Table (block 0 at top)

Cache Hit Count: 128

Cache Miss Count: 128

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.

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.

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.


- Giải thích: Do block size = 4 (words) nên khi gặp phần tử đầu tiên chưa tồn tại trong cache thì cache sẽ lưu vào 1 block gồm 4 phần tử liên tiếp nhau theo hàng ngang của ma trận nhưng vì ta duyệt theo chiều dọc nên 8 block sẽ được lưu hết các giá trị theo 8 hàng ngang tương ứng với các giá trị được duyệt theo chiều dọc sau đó sẽ bị ghi đè lên khi đến hàng thứ 9 chính vì vậy lên khi ta xét đến cột thứ 2 thì giá trị được nạp vào trước đó đã bị đè lên vào không tồn tại nên tỉ lệ ra 0%

## 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	
Cache Hit Count	0	(block 0 at top)	
Cache Miss Count	256	<input type="checkbox"/> = empty	
Cache Hit Rate	0%	<input checked="" type="checkbox"/> = hit <input type="checkbox"/> = miss	

### Runtime Log

☐ Enabled

### Tool Control

Disconnect from MIPS

Reset

Close

13. Change the block size to 16. Note this will reset the tool.
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.
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.
- Giải thích: Do block size = 16 (words) nên khi gặp phần tử đầu tiên chưa tồn tại trong cache thì cache sẽ lưu vào 1 block gồm 16 phần tử liên tiếp nhau theo hàng ngang của ma trận nhưng vì ta duyệt theo chiều dọc nên 8 block sẽ được lưu hết các giá trị theo 8 hàng ngang tương ứng với các giá trị được duyệt theo chiều dọc sau đó sẽ bị ghi đè lên khi đến hàng thứ 9 chính vì vậy lên khi ta xét đến cột thứ 2 thì giá trị được nạp vào trước đó đã bị đè lên vào không tồn tại nên tỉ lệ ra 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



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.

- Giải thích: Do bây giờ số block đã tăng lên 16 vậy lên có thể chứa hết 16 hàng của ma trận vào trong cache do, tuy nhiên do 16 giá trị đầu tiên của các hàng khi được thêm vào không tồn tại nên được đánh miss còn lại được thêm vào theo block với kích thước là 16 nên tỷ lệ là  $240/256 = 94\%$

**Data Cache Simulation Tool, Version 1.2**

### Simulate and illustrate data cache performance

**Cache Organization**

Placement Policy: **Direct Mapping**
Number of blocks: **16**

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

Set size (blocks): **1**
Cache size (bytes): **1024**

**Cache Performance**

Memory Access Count: **256**
Cache Hit Count: **240**
Cache Miss Count: **16**
Cache Hit Rate: **94%**

**Cache Block Table**  
(block 0 at top)  
☐ = empty  
☒ = hit  
☐ = miss

**Runtime Log**
☐ Enabled

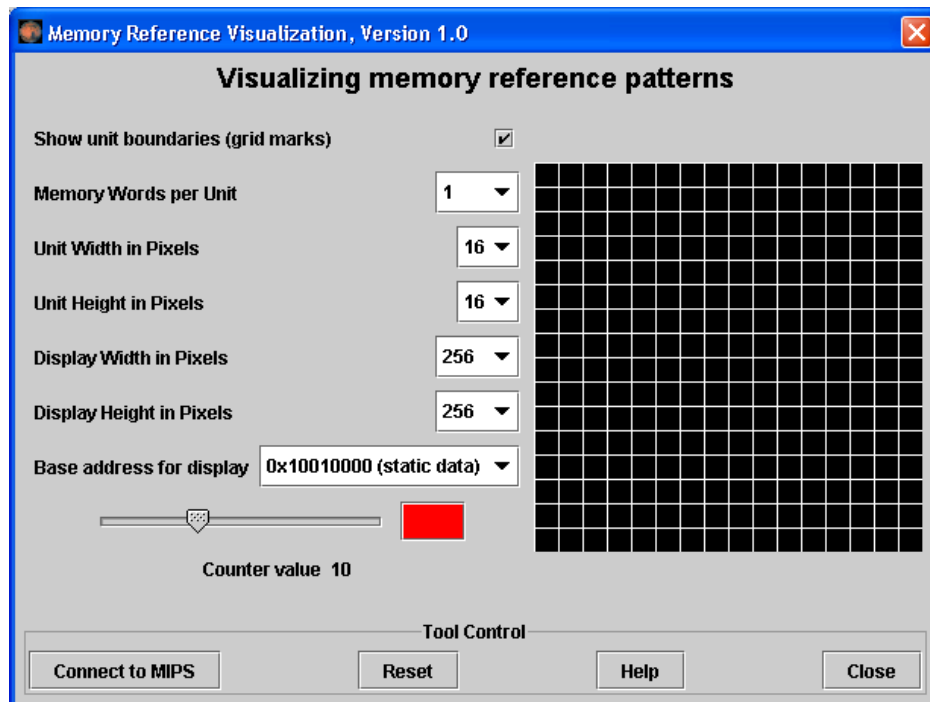
**Tool Control**

Disconnect from MIPS
Reset
Close

## Assign 2:

### The Memory Reference Visualization tool

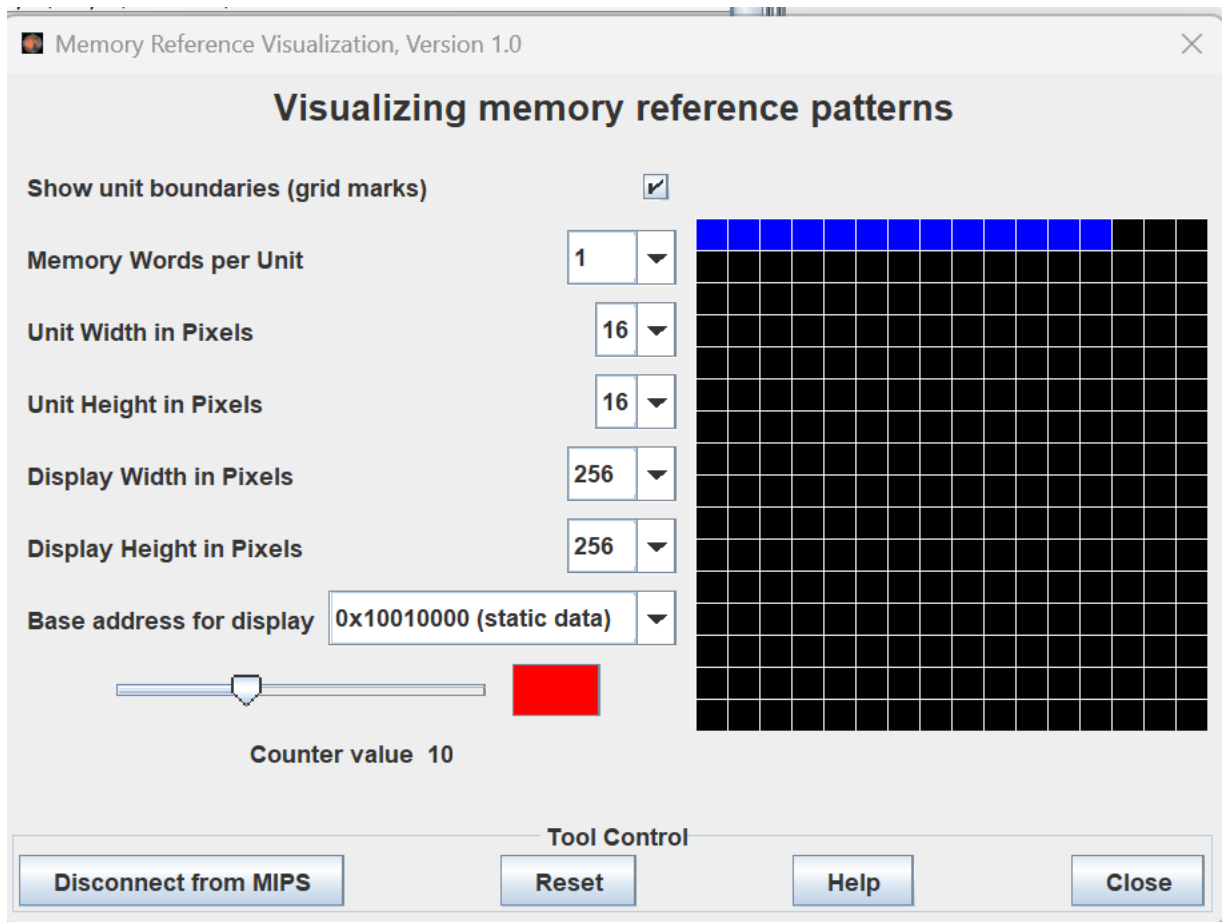
- 1 Open the program `row-major.asm` from the `Examples` folder if it is not already open.
- 2 Assemble the program.
- 3 From the **Tools** menu, select **Memory Reference Visualization**. A new frame will appear in the middle of the screen.



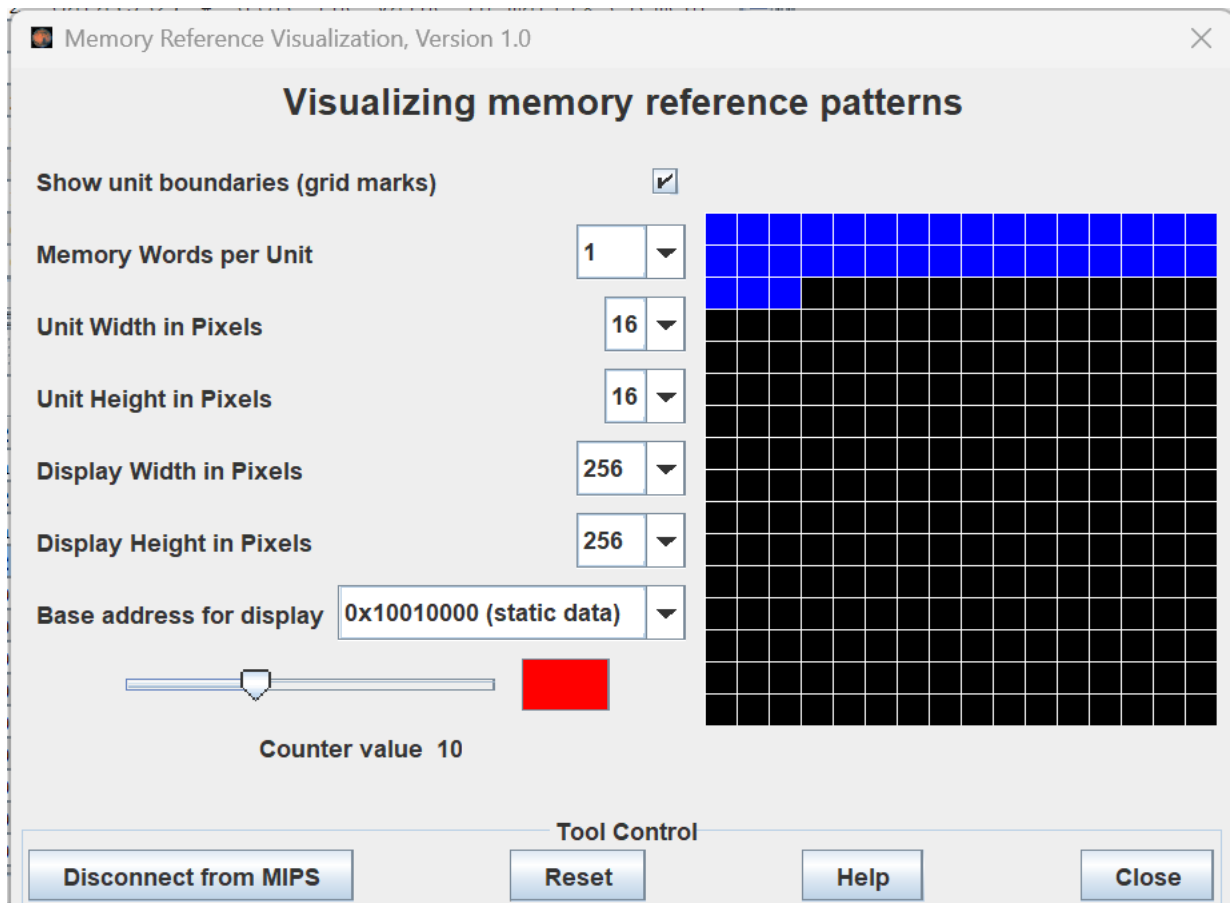
This tool will paint a grid unit each time the corresponding MIPS memory word is referenced. The base address, the first static data segment (`.data` directive) word, corresponds to the upper-left grid unit. Address correspondence continues in row-major order (left to right, then next row down).

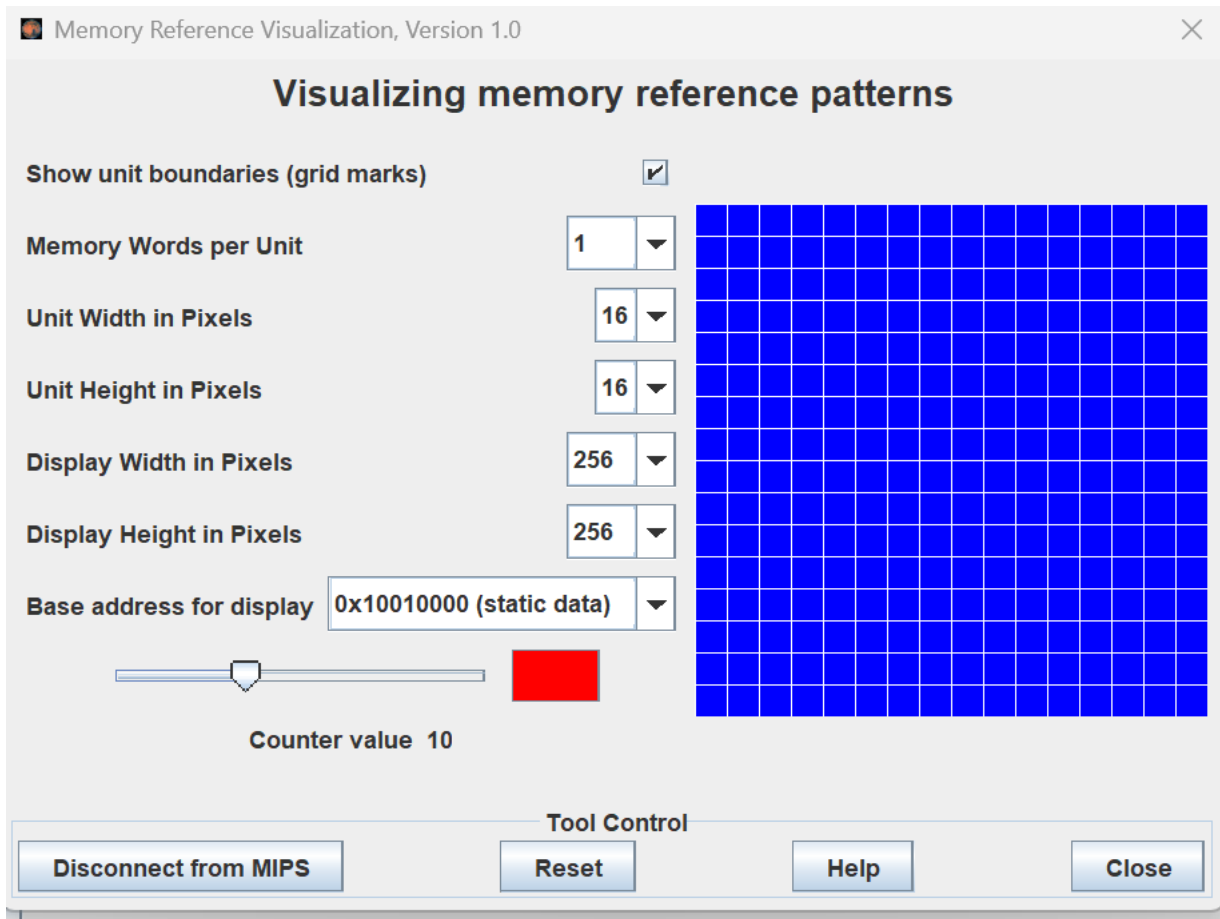
The color depends on the number of times the word has been referenced. Black is 0, blue is 1, green is 2, yellow is 3 and 4, orange is 5 through 9, red is 10 or higher. View the scale using the tool's slider control. You can change the color (but not the reference count) by clicking on the color patch.

- 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.*

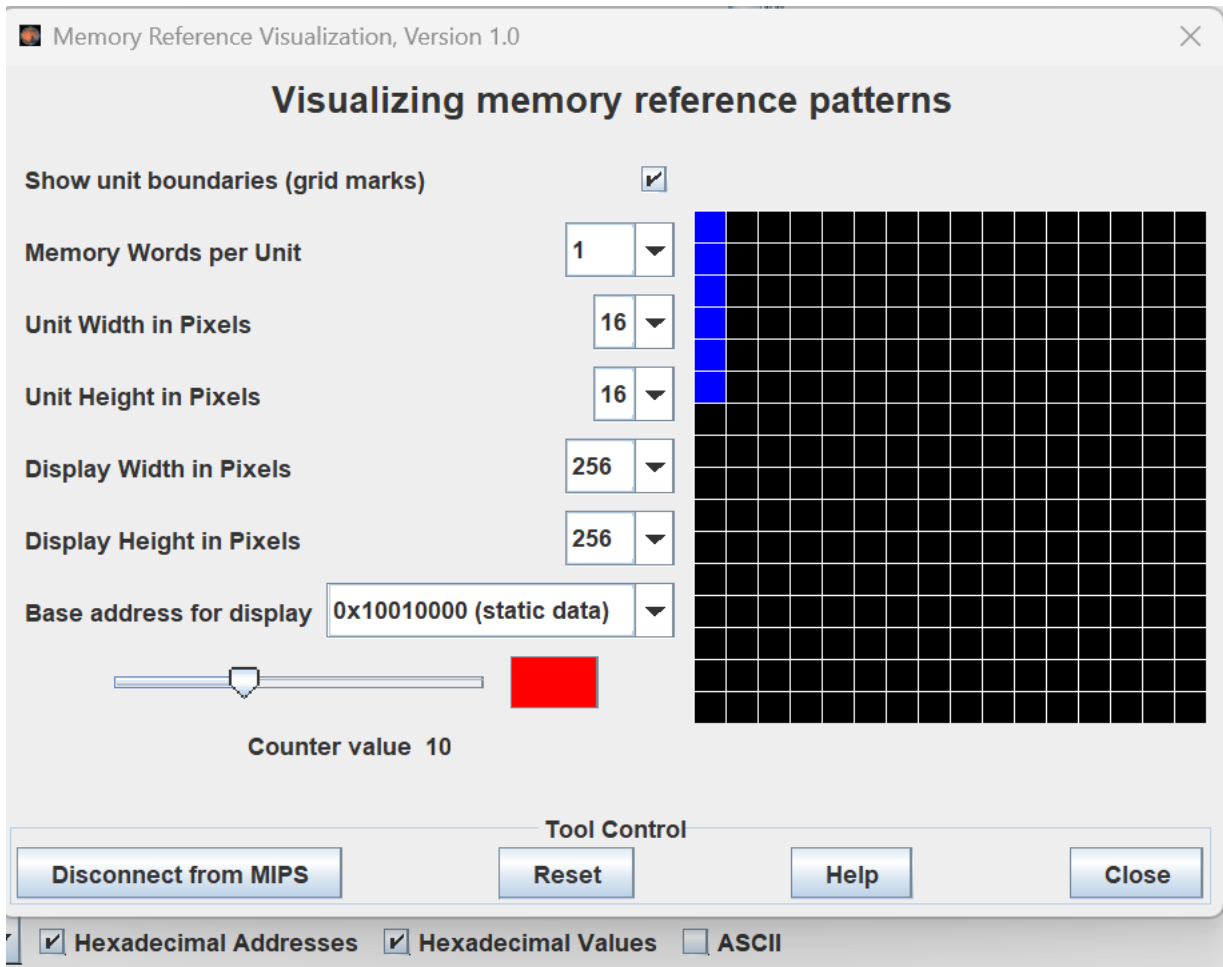


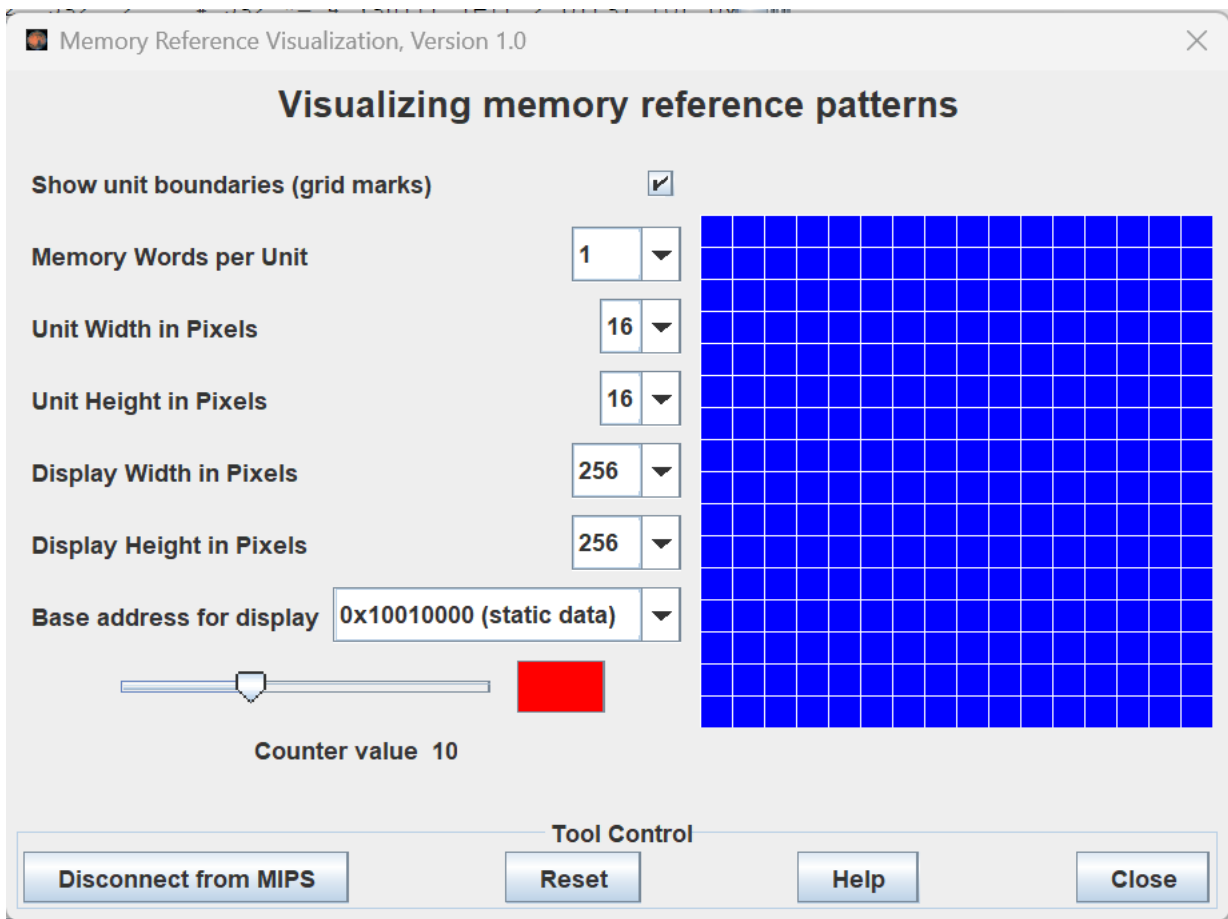
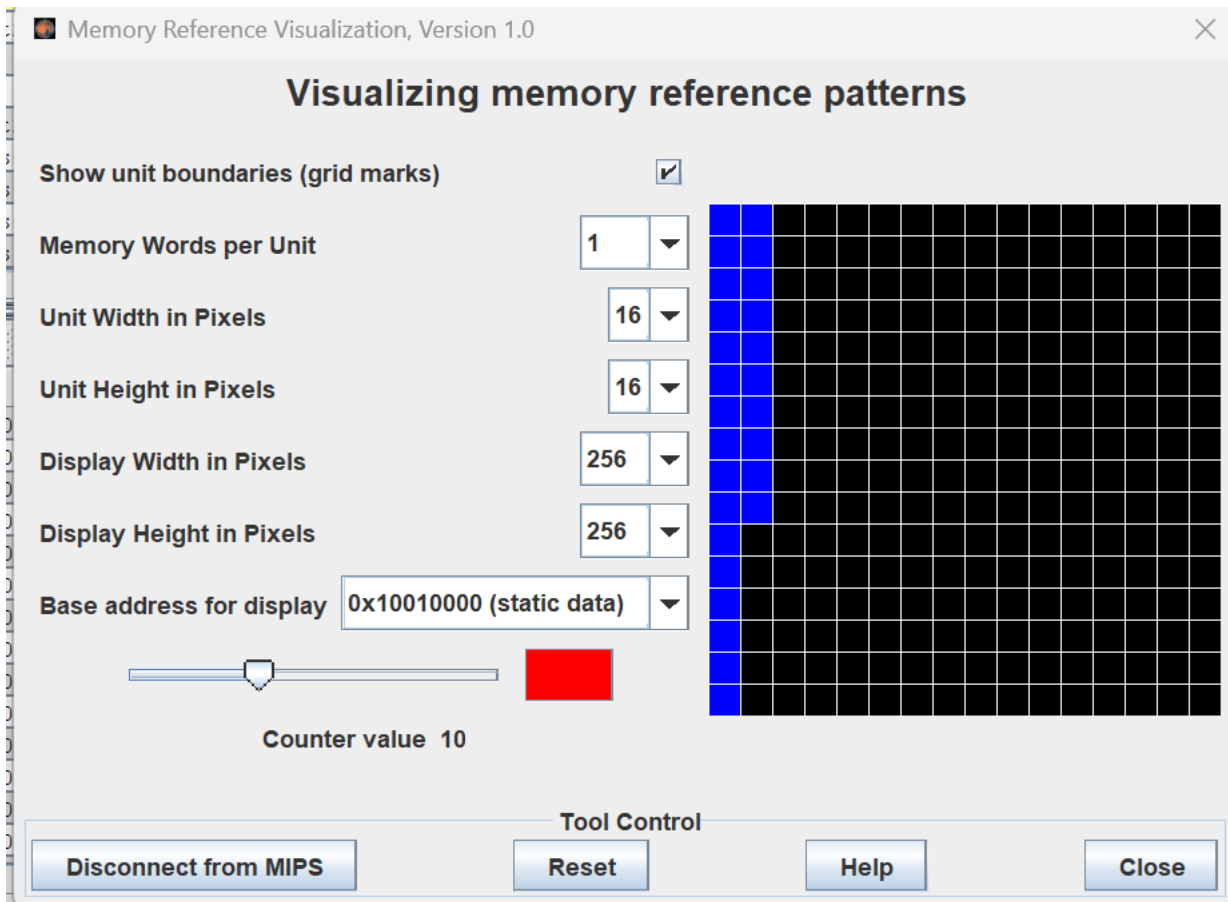
7



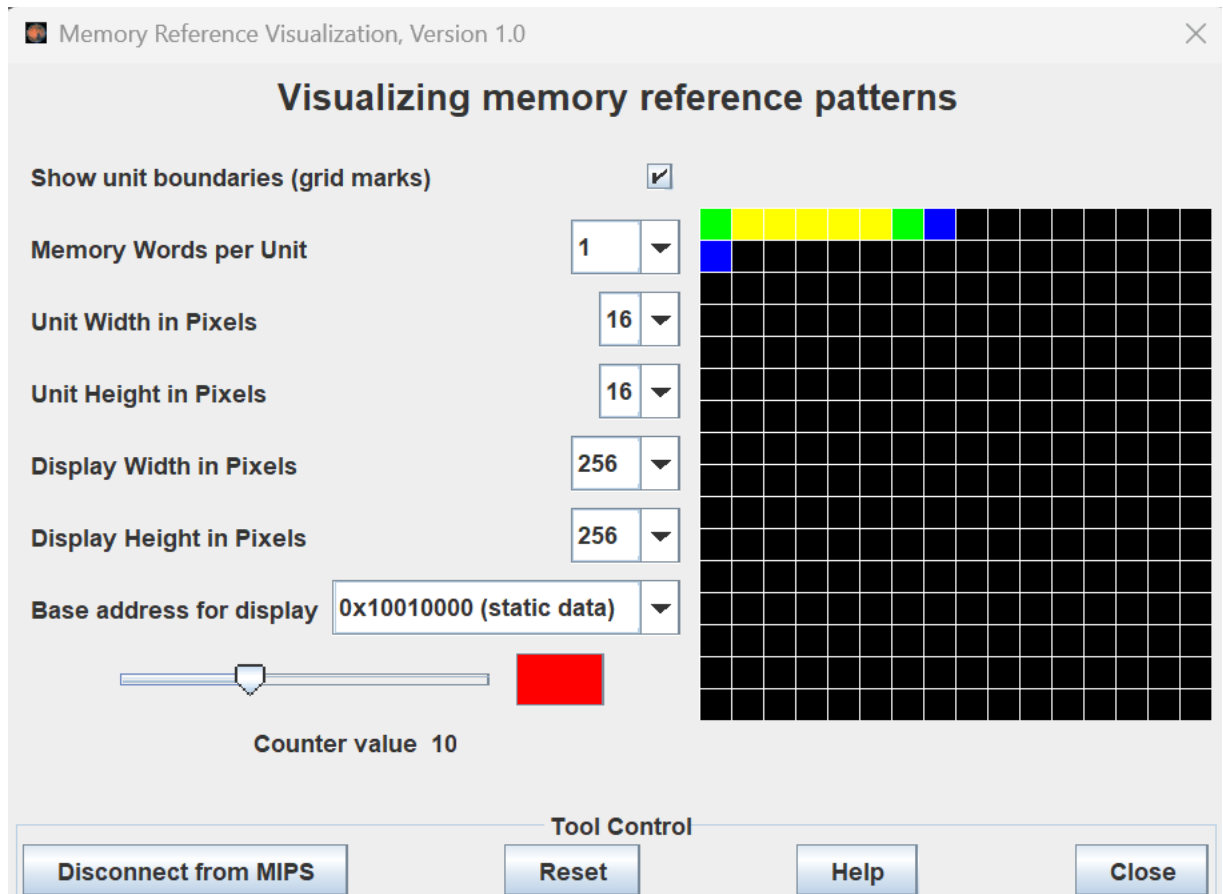


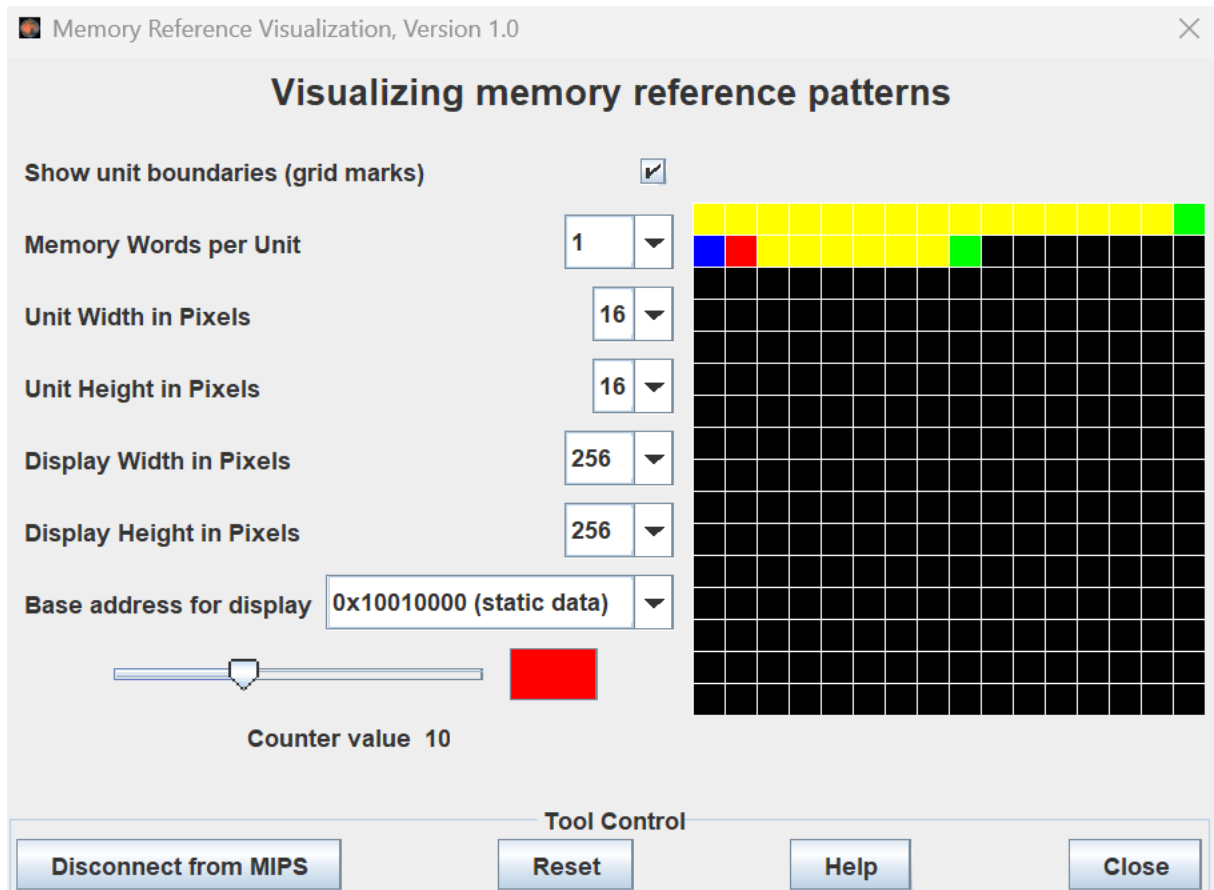
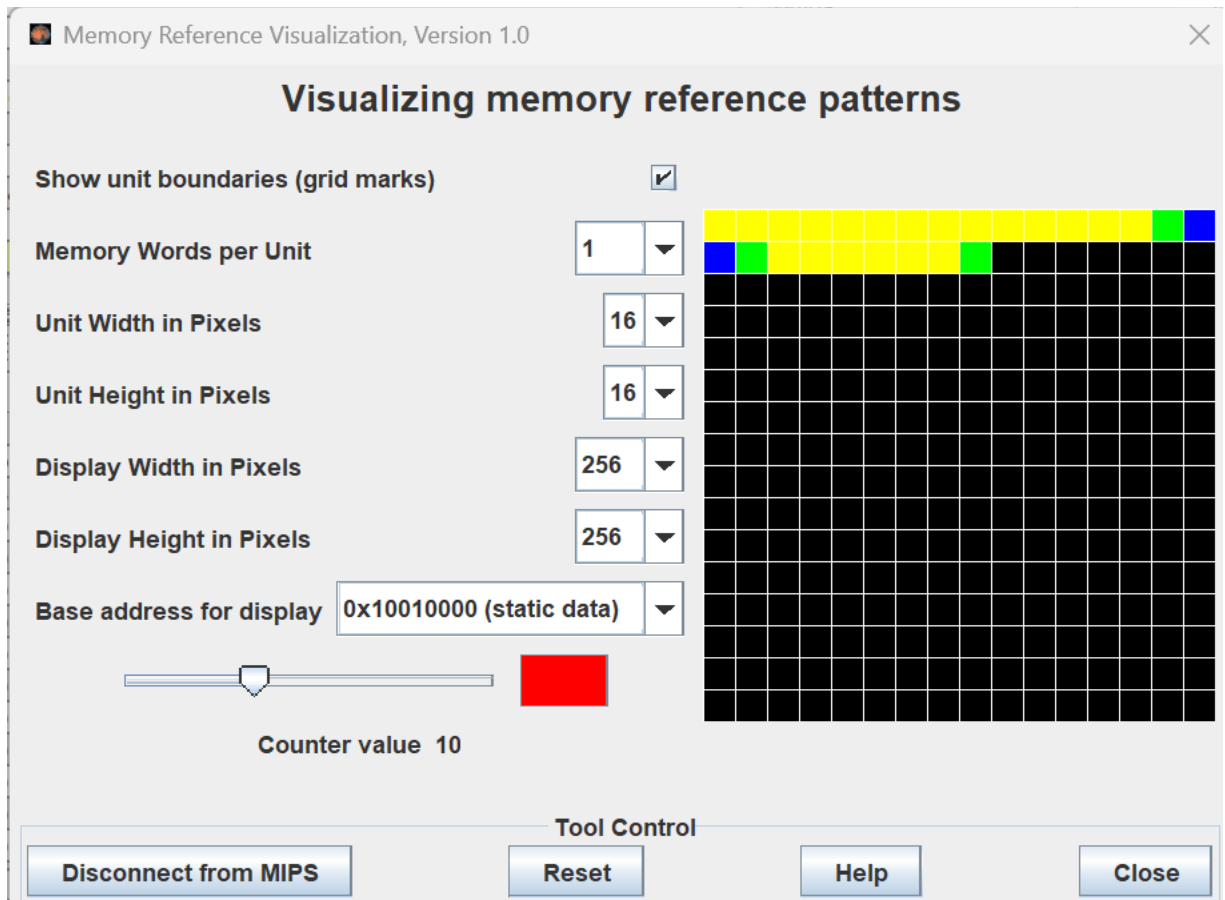
- 8 Hopefully you observed that the animation sequence corresponded to the expected memory access sequence of the row-major.asm program. *If you have trouble seeing the blue*, reset the tool, move the slider to position 1, change the color to something brighter, and re-run.
- 9 Repeat steps 2 through 7, for **column-major.asm**. You should observe that the animation sequence corresponded to the expected memory access sequence of this program.





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

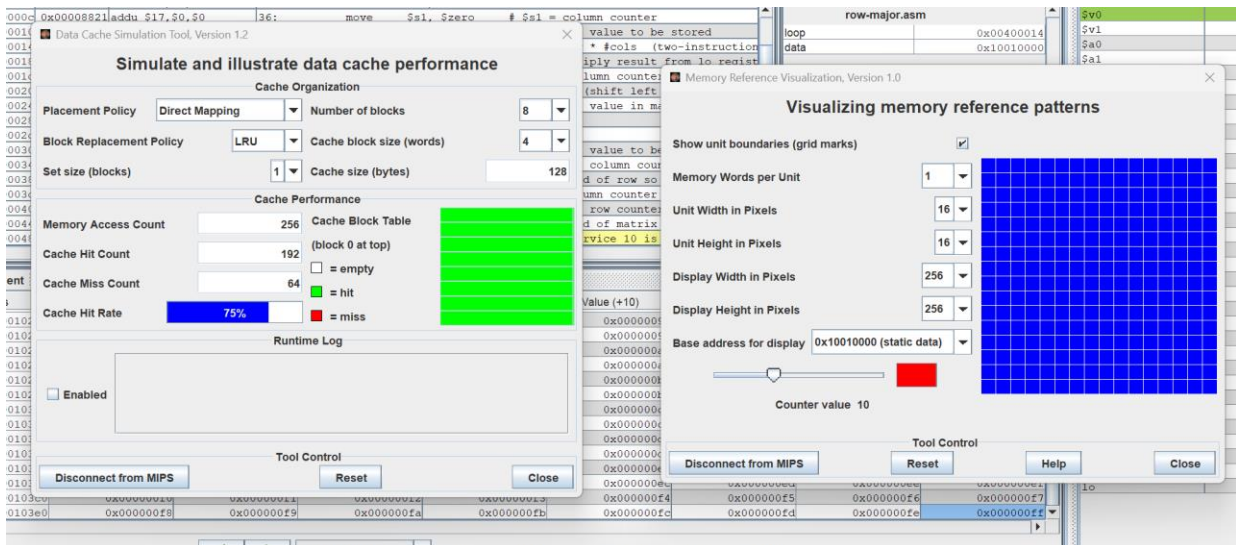
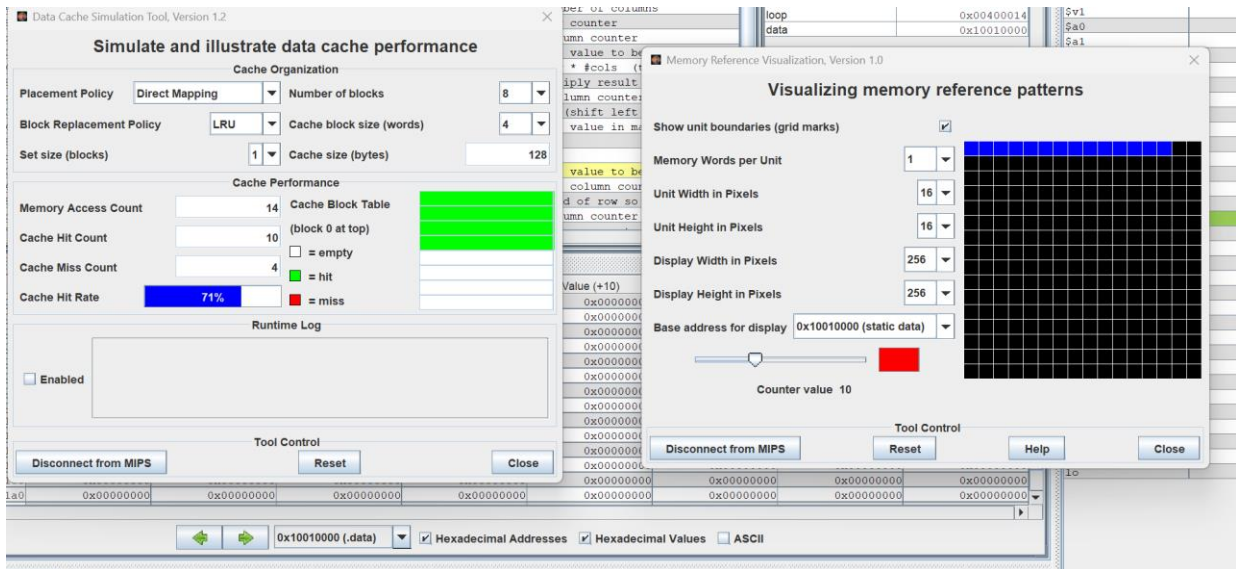






(Optional) Create a new instance of the Data Cache Simulator. Move the two frames around so you can see both. Connect the cache simulator to MIPS and reset the Memory Reference Visualization. Re-run the program. This exercise illustrates that two different tools can be used simultaneously.

**Ví dụ: Ta chạy file row-major.asm rồi mở đồng thời 2 cửa sổ Data Cache Simulator và Memory Reference Visualization lên để quan sát.**



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

and memory reference patterns in general.