BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH TUẦN 12

Họ và tên: Nguyễn Quý Đức MSSV: 20235682

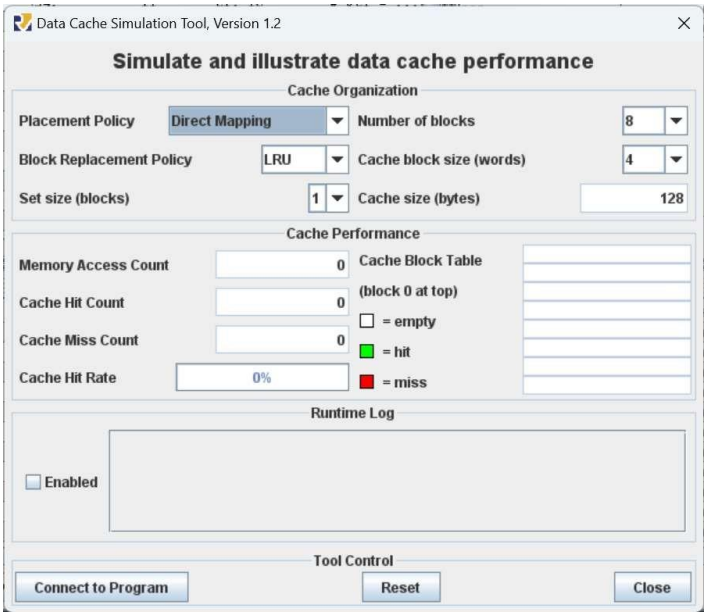
Mã HP: IT3280 Mã lớp: 156788

Assignment 1 – Sử dụng công cụ Data Cache Simulator

1. Mở chương trình **row-major.asm** trong thư mục Lab12. Chương trình này duyệt ma trận kích thước 16x16 theo thứ tự hàng chính (duyệt từng hàng), gán các giá trị từ 0 đến 255. Chương trình thực hiện thuật toán sau:

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

2. Biên dịch chương trình.
3. Từ menu Tools, mở công cụ Data Cache Simulator.

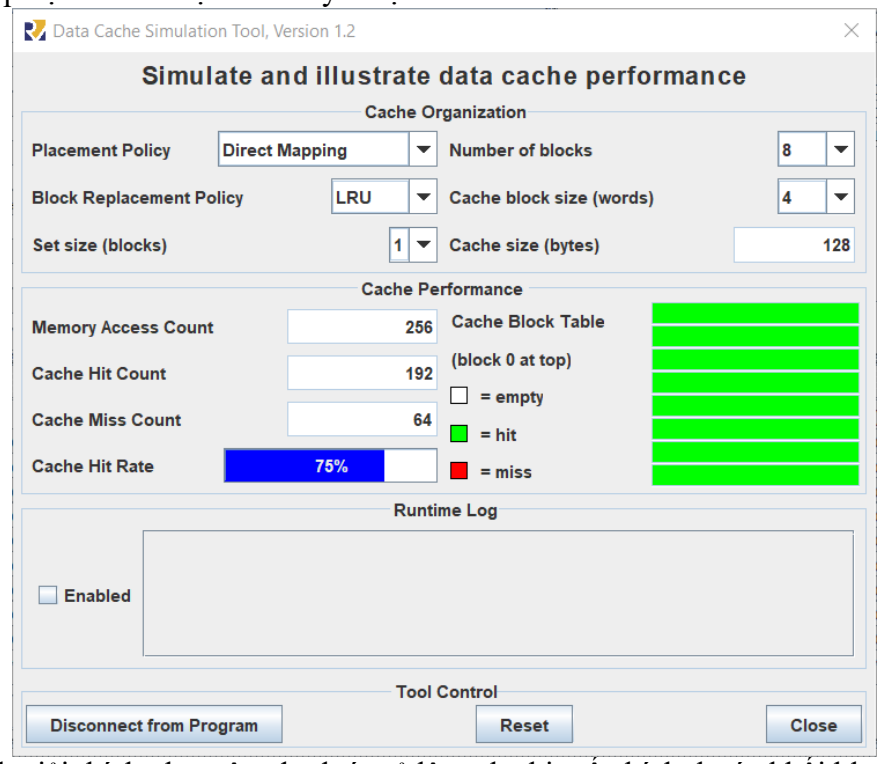


Đây là công cụ giả lập việc sử dụng và hiệu năng của bộ nhớ đệm nhanh khi chương trình được thực hiện. Bao gồm các vùng chức năng chính:

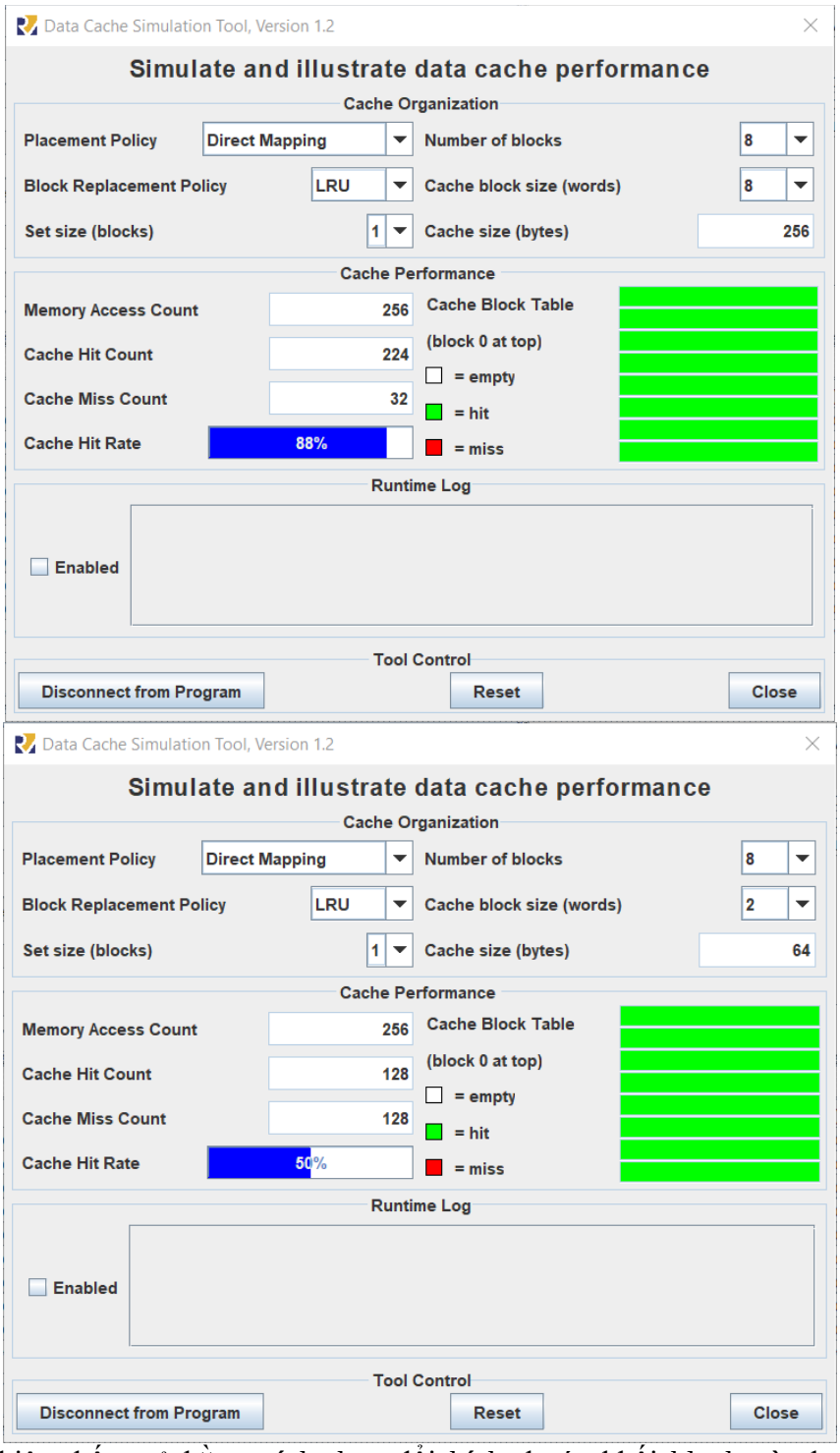
- *Cache Organization*: gồm các thiết lập mô tả cách bộ nhớ đệm được cấu hình.

- *Cache Performance*: với mỗi lần truy nhập bộ nhớ khi chương trình hoạt động, chương trình giả lập sẽ xác định việc truy nhập dữ liệu có thỏa mãn từ bộ nhớ đệm hay không và cập nhật việc hiển thị hiệu năng tương ứng.

4. Nhấn vào nút **Connect to Program**.
5. Quay trở lại RARS, điều chỉnh tốc độ thực hiện là 30 lệnh/s. Thiết lập này làm chậm lại quá trình thực thi chương trình, giúp sinh viên quan sát được hiệu ứng minh họa trong vùng Cache Performance.
6. Nhấn nút chạy chương trình, theo dõi Cache Performance được cập nhật mỗi lần truy nhập bộ nhớ.
7. *Tỷ lệ cache hit sau khi chạy chương trình*? ***75%*** Với mỗi lần cache miss, một khối gồm 4 từ nhớ được ghi vào bộ nhớ đệm. Với cách duyệt theo dòng, các phần tử của ma trận được truy nhập theo cùng thứ tự lưu trữ trong bộ nhớ. Vì vậy sau mỗi lần cache miss là 3 lần cache hit khi 3 phần tử tiếp theo được lưu trữ trong cùng một khối dữ liệu lưu vào bộ đệm. Lượt truy nhập tiếp theo sẽ là cache miss khi chế độ ánh xạ trực tiếp (Direct Mapping) ánh xạ tới khối tiếp theo. Do đó, 3 trong 4 lần truy nhập bộ nhớ sẽ được tìm thấy ở bộ nhớ cache.



8. Với cách giải thích như trên, dự đoán tỷ lệ cache hit nếu kích thước khối block tăng từ 4 thành 8 từ nhớ? $\mathbf{1 - \frac{1}{8} = 87.5\%}$. Hoặc giảm từ 4 thành 2 từ nhớ? $\mathbf{1 - \frac{1}{2} = 50\%}$

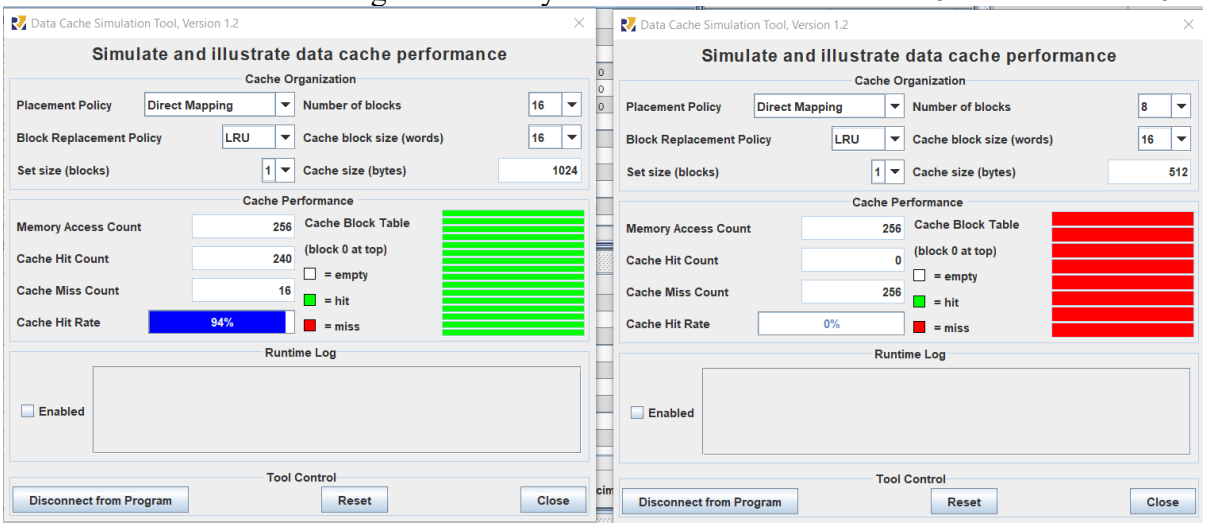


9. Kiểm nghiệm kết quả bằng cách thay đổi kích thước khối block và chạy lại chương trình từ bước 6.
   Chú ý: Khi thay đổi các cài đặt trong Cache Organization, giá trị hiệu năng tính toán trước đó sẽ được khởi tạo lại.
10. Thực hiện lại các bước trên với chương trình **column-major.asm** trong thư mục Lab12. Chương trình này sẽ duyệt các phần tử trong ma trận kích thước 16x16 theo thứ tự cột chính (duyệt từng cột) và lần lượt gán các giá trị từ 0 đến 255. Chương

trình thực hiện thuật toán sau:

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

11. *Hiệu năng truy nhập bộ nhớ cache đối với chương trình này là bao nhiêu?* ***0%***. Vấn đề được xác định ở đây là các từ nhớ không được truy nhập tuần tự như trong chương trình trước. Mỗi lần truy nhập cách nhau 16 từ nhớ. Với những thiết lập hiện tại trong công cụ giả lập, không có 2 lần truy nhập liên tiếp nào cùng chung một khối block, do đó lần truy nhập nào cũng là cache miss.
12. Thay đổi kích thước block thành 16. Khởi tạo lại công cụ giả lập.
13. Mở thêm 1 cửa sổ công cụ Cache Simulator, đặt bên cạnh cửa sổ hiện tại đang mở. Kết nối với chương trình và thay đổi kích thước block thành 16 và số block là 16.

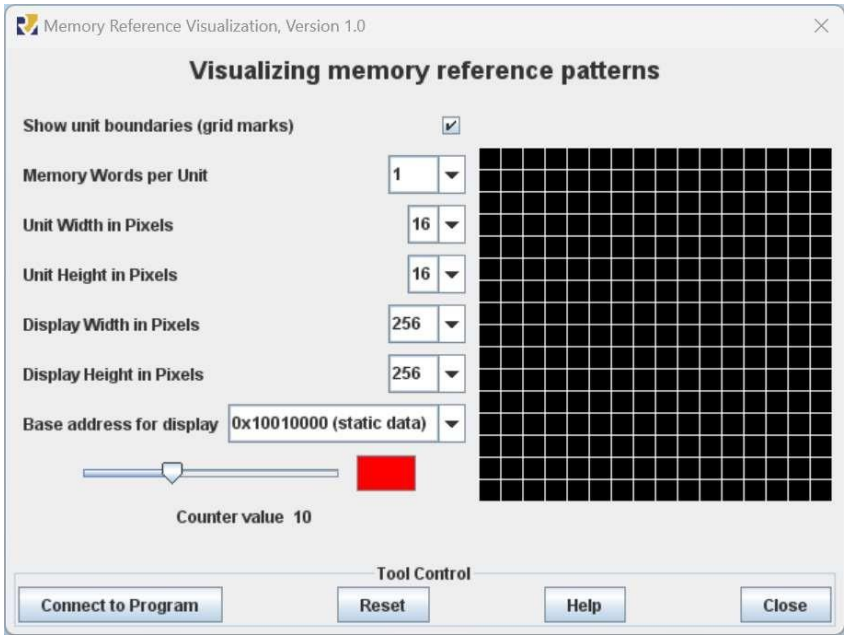


14. Chạy lại chương trình.
    Hiệu năng truy nhập được tính trên cửa sổ cũ là bao nhiêu? ***0%.*** Tăng kích thước block thành 16 từ nhớ không làm tăng hiệu năng bởi vì chỉ có một lần truy nhập với mỗi block, trước khi block được thay thế bởi block khác.

    Hiệu năng truy nhập được tính trên cửa sổ mới là bao nhiêu? ***93.75%*** Trong trường hợp này, toàn bộ ma trận sẽ vừa với bộ nhớ cache và một khi block được đọc, nó sẽ không bị thay thế bởi block khác. Chỉ có lần truy nhập block đầu tiên trả về kết quả là cache miss.

Assignment 2 – Sử dụng công cụ Memory Reference Visualization

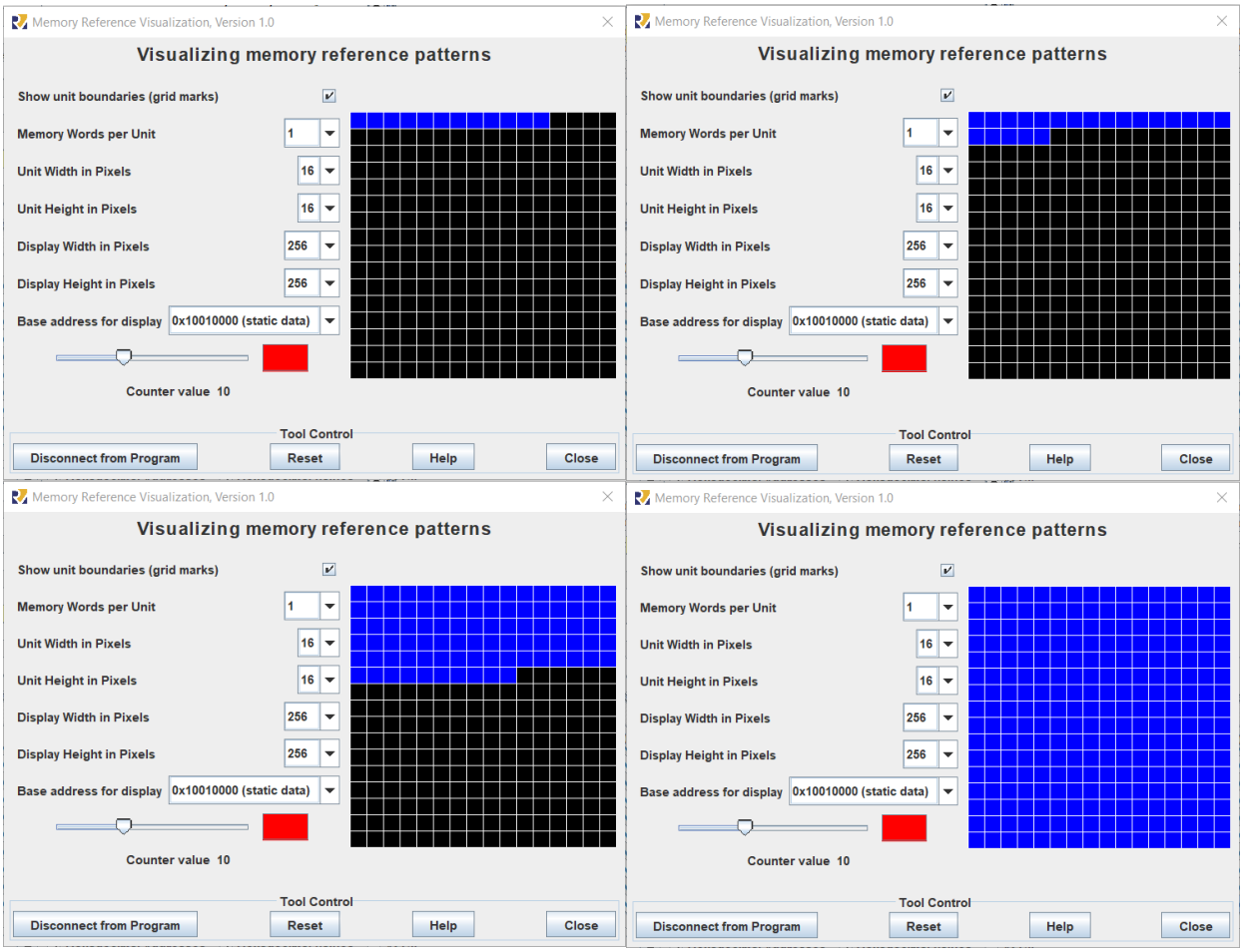
1. Mở chương trình **row-major.asm** trong thư mục **Lab12**.
2. Dịch chương trình.
3. Từ menu **Tools**, mở công cụ **Memory Reference Visualization**.



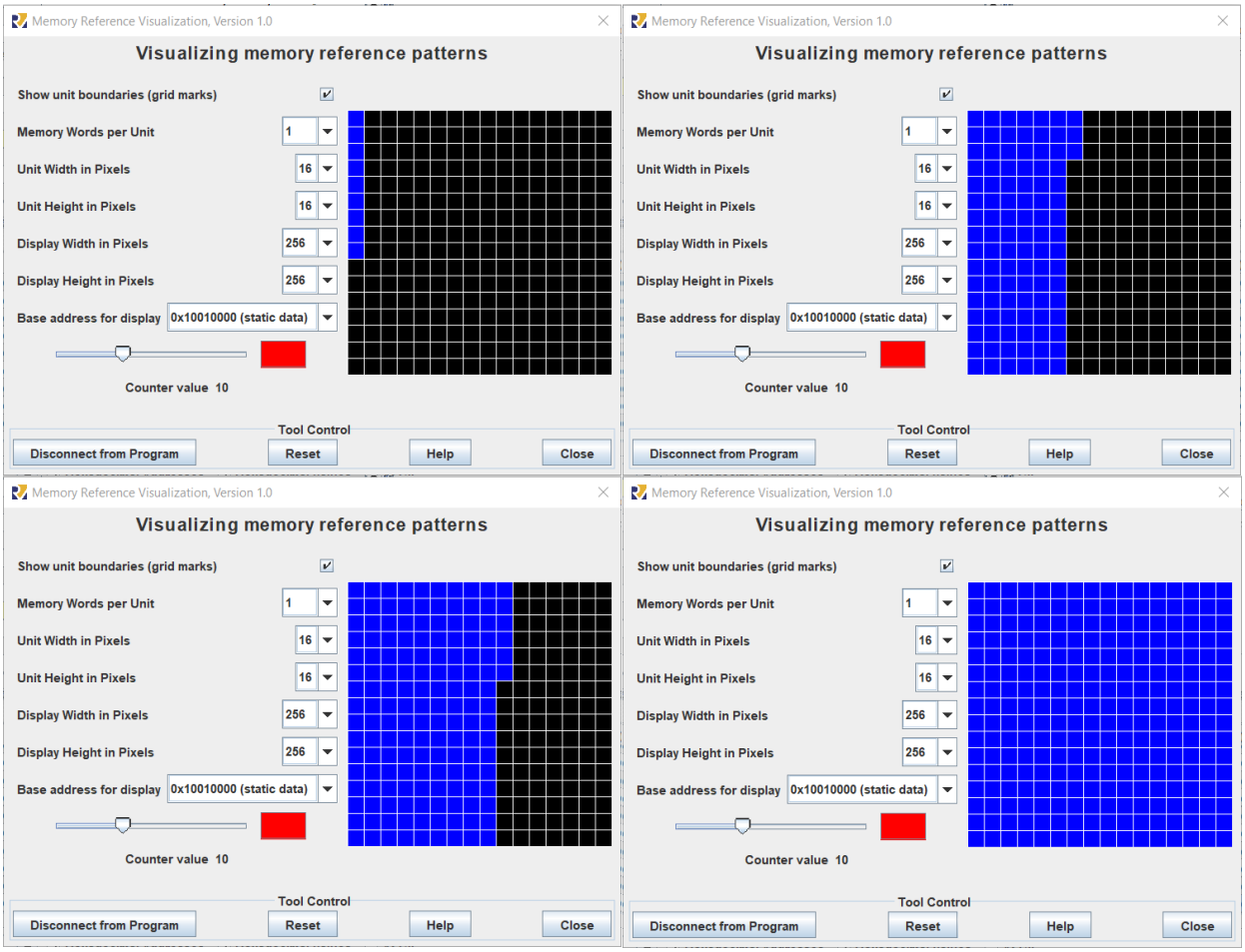
Công cụ này sẽ tô màu vào mỗi ô trong lưới ở bên phải mỗi khi một từ nhớ được tham chiếu. Địa chỉ cơ sở (ứng với chỉ thị biên dịch .data) tương ứng với ô ở góc trên bên trái. Địa chỉ được gán theo thứ tự hàng (từ trái qua phải, từ trên xuống dưới).
Màu các ô phụ thuộc vào số lần từ nhớ tương ứng được tham chiếu. Màu đen là 0, màu xanh lam là 1, màu xanh lá là 2, màu vàng là 4, màu cam từ 5 đến 9, màu đỏ là 10 và cao hơn.

4. Nhấn vào nút **Connect to Program**.
5. Điều chỉnh tốc độ thực hiện lệnh trong RARS là 30 lệnh/s.
6. Chạy chương trình. Quan sát công cụ minh họa việc truy nhập bộ nhớ.
7. Lặp lại các bước 1 đến 6 với chương trình **column-major.asm**.
8. Lặp lại các bước 1 đến 6 với chương trình **fibonacci.asm**.

row-major.asm:



column-major.asm



fibonacci.asm