**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**\*\*\*\*\*\*\*\***

A picture containing icon

Description automatically generated

**BÁO CÁO**

**Bài tập thực hành tuần 12**

**Học phần: Thực hành kiến trúc máy tính**

**Giảng viên hướng dẫn:** Lê Bá Vui

**Sinh viên thực hiện:** Nguyễn Bình An - 20225591

**Hà Nội, tháng 5 năm 2024**

Assignment 1

Thực hiện từ bước 1-7 :

A screenshot of a computer

Description automatically generated

Result:

A screenshot of a computer

Description automatically generated

*Tỉ lệ cache hit rate cuối cùng:* \_\_75%\_\_Với mỗi lần tìm không thấy (cache miss), một khối 4-words sẽ được ghi vào cache. Trong quá trình duyệt theo hàng (row-major traversal), các phần tử ma trận được truy cập theo thứ tự chúng được lưu trữ trong bộ nhớ. Do đó, mỗi lần bỏ lỡ bộ đệm sẽ được theo sau bởi 3 lần truy cập (cache hit) khi 3 phần tử tiếp theo được tìm thấy trong cùng một khối cache. Tiếp theo là một lần cache miss khác khi ánh xạ trực tiếp (Direct Mapping) đến khối cache tiếp theo, và lặp lại chinh nó. Vì vậy, 3 trong số 4 lần truy cập bộ nhớ sẽ được giải quyết trong bộ đệm.9.

*Dự đoán, khi tăng blocksize từ 4 lên 8 thì tỉ lệ Cache hit rate là* : \_\_\_87%\_\_\_:

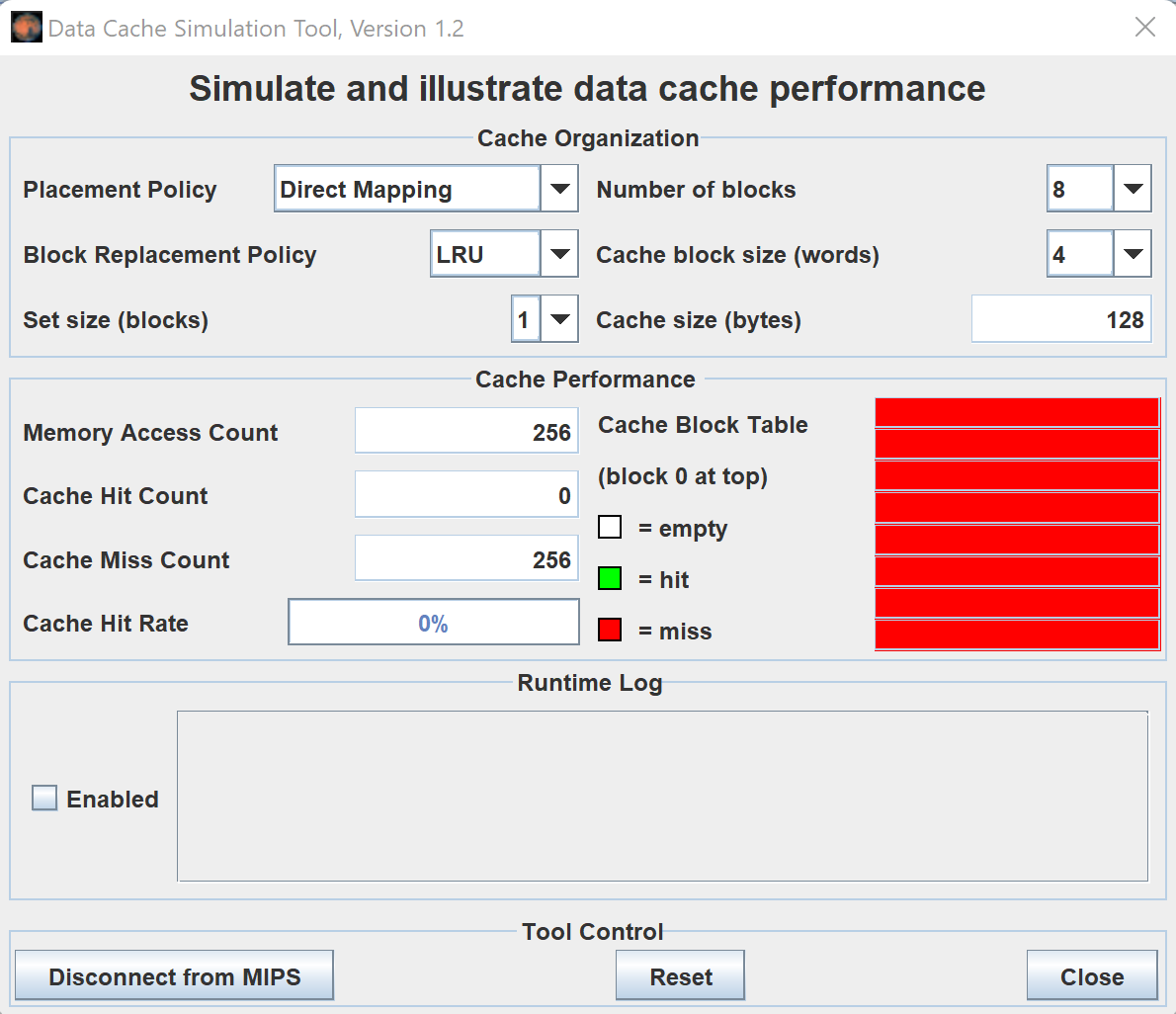
A screenshot of a computer

Description automatically generatedMỗi khi có cache miss, một khối gồm 8 số nguyên được ghi vào cache. Trong quá trình duyệt theo thứ tự hàng, các phần tử ma trận được truy cập theo cùng thứ tự mà chúng được lưu trữ trong bộ nhớ. Do đó, sau mỗi cache miss sẽ có 7 lần cache hit khi 7 phần tử tiếp theo cũng nằm trong cùng một khối cache. Sau đó, lại xảy ra một cache miss khi ánh xạ theo phương pháp Direct Mapping chuyển đến khối cache kế tiếp, và mô hình này lặp đi lặp lại. Vì vậy, 7 trên 8 lần truy cập bộ nhớ sẽ được giải quyết trong cache.

*Khi giảm blocksize từ 4 xuống 2 thì thỉ lệ Cache hit rate*: \_\_\_50%\_\_\_.

A screenshot of a computer

Description automatically generatedMỗi khi có cache miss, một khối gồm 1 số nguyên được ghi vào cache. Trong quá trình duyệt theo thứ tự hàng, các phần tử ma trận được truy cập theo cùng thứ tự mà chúng được lưu trữ trong bộ nhớ. Do đó, sau mỗi cache miss sẽ có 1 lần cache hit khi 1 phần tử tiếp theo cũng nằm trong cùng một khối cache. Sau đó, lại xảy ra một cache miss khi ánh xạ theo phương pháp Direct Mapping chuyển đến khối cache kế tiếp, và mô hình này lặp đi lặp lại. Vì vậy, 1 trên 2 lần truy cập bộ nhớ sẽ được giải quyết trong cache.

*A screenshot of a computer

Description automatically generatedKhi duyệt mảng theo cột, Cache Hit Rate bằng* 0% đối với Cache Block Size bằng 4 words và bằng 16 words

*Khi duyệt mảng theo cột và sử dụng kích thước khối Cache là 4 words hoặc 16 words*, Cache Hit Rate bằng 0% vì các vị trí bộ nhớ được truy cập không liên tiếp và không có hai truy cập bộ nhớ liên tiếp nào nằm trong cùng một khối cache.

* không tận dụng được cache và hiệu suất cache là không tốt.

Khi thay đổi số lượng của Block lên 16

A screenshot of a computer

Description automatically generatedLần duyệt đầu Cache Hit Rate bằng 0%

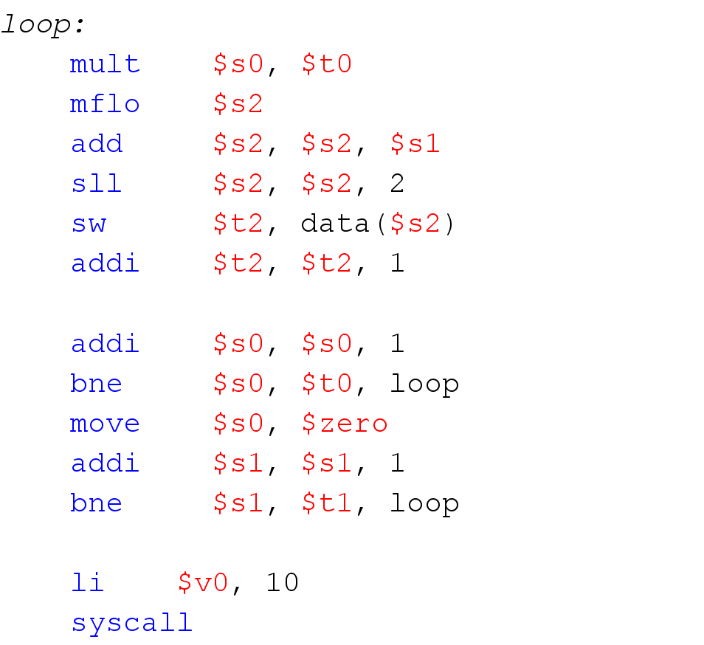
A screenshot of a computer

Description automatically generatedSau lần duyệt đầu Cache Hit Rate bằng 94%

Khi sử dụng 16 khối cache với kích thước là 16 words, trong 16 lần duyệt đầu tiên, mỗi khi duyệt một phần tử, một khối cache mới sẽ được tạo ra cho phần tử đó. Do đó, trong 16 lần duyệt đầu tiên, sẽ có 16 lần cache miss. Tuy nhiên, từ lần duyệt thứ 17 trở đi, khi duyệt một phần tử, chúng ta sẽ quay lại một khối cache cũ mà đã được sử dụng trước đó và không có khối nào bị ghi đè. Do đó, trong các lần duyệt tiếp theo, sẽ xảy ra cache hit cho mỗi phần tử, vì chúng đã được lưu trữ trong cache. Tổng cộng, sẽ có 16 lần cache miss và 240 lần cache hit, tỷ lệ hit là khoảng 94%.

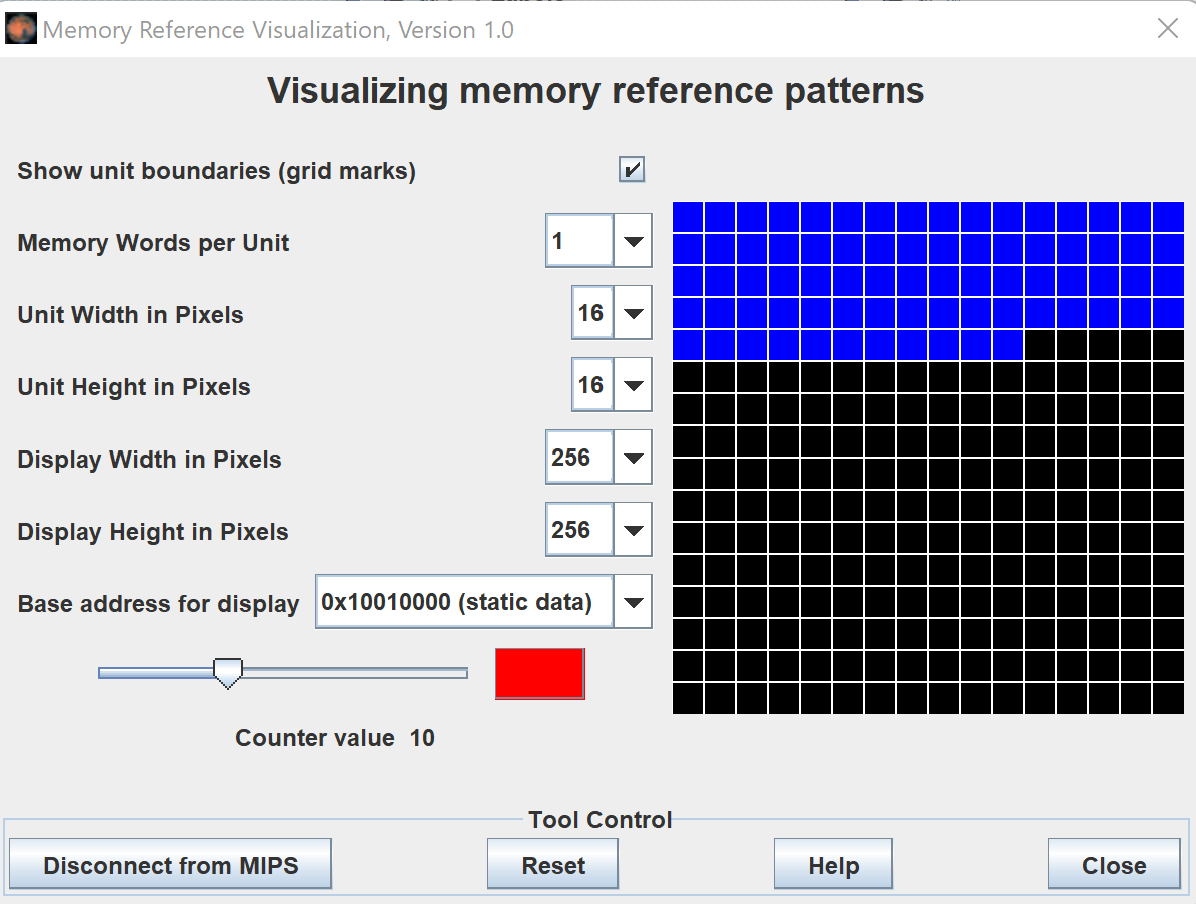
Với 16 khối cache có kích thước 16 words, sau lần duyệt đầu tiên, cache có thể lưu trữ mảng 16x16, dẫn đến tỷ lệ hit tương đối cao.

Sửa code để duyệt theo cột:



The Memory Reference Visualization tool

Chạy file **r****ow-major.asm**

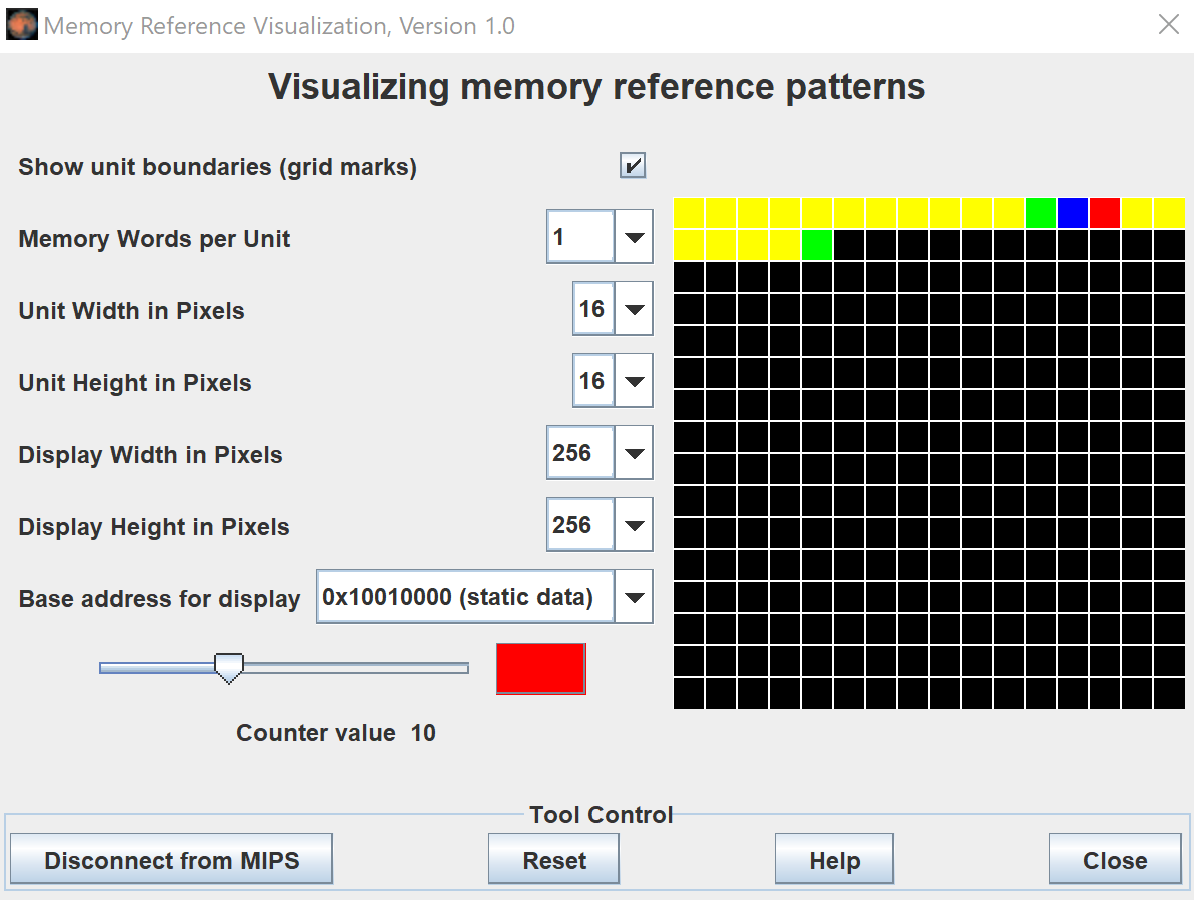
****

Chạy file **column-major.asm**

**A screenshot of a computer

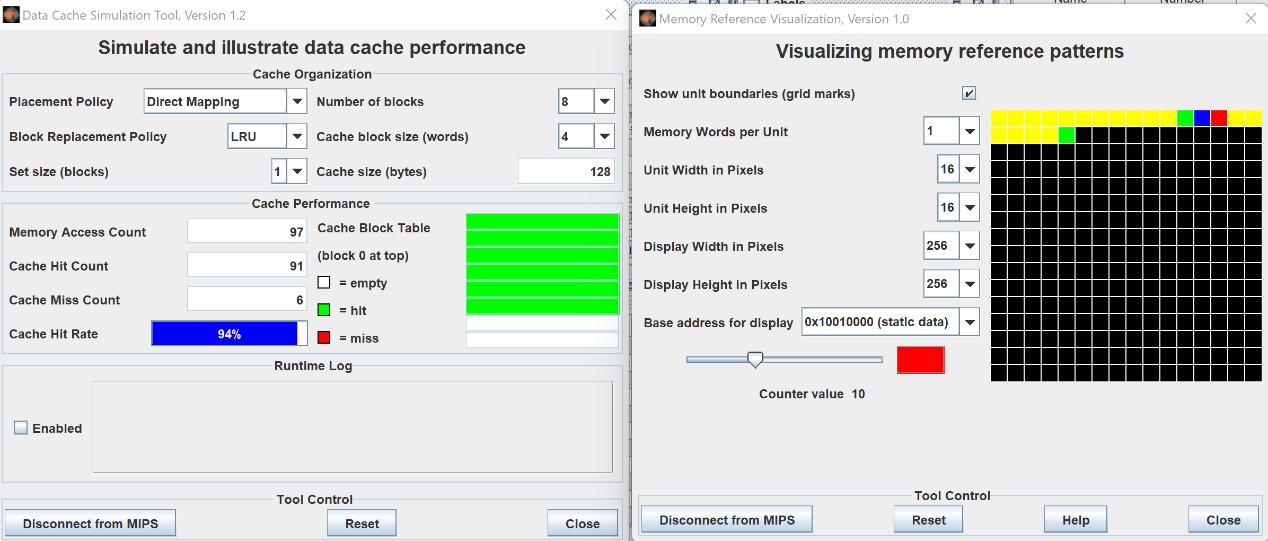
Description automatically generated**

Chạy file **fibonacci.asm**



Tính 12 số fibonacci đầu

Truy cập lần đầu: xanh dương, lần 2: xanh lá, lần 3: vàng, lần 4: đỏ

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.