ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

KIẾN TRÚC MÁY TÍNH - THỰC HÀNH (CO2008)

Bài tập lớn - Học kì 241

CHƯƠNG TRÌNH IN PHỔ SAO

Giảng viên: Huỳnh Phúc Nghị
Lớp: L09
Nhóm: 8
Danh sách sinh viên: Nguyễn Công Anh Luân - 2111703
Nguyễn Quốc Đạt - 2210694
Võ Thị Đông Nghi - 2312254

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 11 - 2024

Contents

1 Tổng quan đề tài

1.1 Mục tiêu đề tài

Đề tài yêu cầu viết một chương trình trên Mars MIPS 4.5 để tạo ra hình ảnh "phổ sao" dựa trên chuỗi số ASCII được cung cấp trong file `STRING.TXT`. Cụ thể, chương trình sẽ đọc các ký tự số từ file, mỗi chữ số biểu thị số lượng ký tự sao (*) sẽ được in ra theo chiều dọc trong một cột tương ứng. Kết quả của chương trình là hình ảnh ASCII của phổ sao được lưu trong file `PHO_SAO.TXT`.

Đề tài "Chương trình in phổ sao" tập trung vào việc phát triển một chương trình MIPS để tạo ra hình ảnh phổ sao bằng ký tự từ một chuỗi số đầu vào. Mục tiêu của đề tài là cung cấp một công cụ để xử lý chuỗi và trực quan hóa dữ liệu dưới dạng đồ họa ASCII, củng cố kiến thức về xử lý chuỗi, thao tác với file và quản lý dữ liệu đầu ra trong môi trường MIPS. Việc tạo hình bằng ký tự ASCII có ý nghĩa trong việc hiển thị dữ liệu một cách trực quan trên các thiết bị hạn chế về tài nguyên, đồng thời giúp ta làm quen với các thao tác lập trình cơ bản trong kiến trúc máy tính.

1.2 Lợi ích

- **Học tập về kiến trúc máy tính và ngôn ngữ Assembly**: giúp mở rộng và củng cố kiến thức về các khái niệm lập trình cơ bản trong kiến trúc máy tính như các loại thanh ghi, xử lý dữ liệu nhị phân, và quản lý bộ nhớ.
- **Kỹ năng xử lý chuỗi và thao tác file**: Bài toán yêu cầu đọc dữ liệu từ file và xuất kết quả ra file khác, từ đó nắm vững các thao tác vào/ra (I/O) trong MIPS và các lệnh thao tác chuỗi ASCII.
- **Trực quan hóa dữ liệu bằng đồ họa ASCII**: Bài toán này rèn luyện khả năng trực quan hóa dữ liệu thông qua các ký tự ASCII đơn giản, giúp sinh viên học cách biểu diễn thông tin một cách dễ hiểu trong môi trường hạn chế về đồ họa.
- **Tăng khả năng tư duy logic và thuật toán**: Quá trình viết và tối ưu mã trong MIPS đòi hỏi sự tư duy logic chặt chẽ.

1.3 Khó khăn

- **Giới hạn về cú pháp và câu lệnh**: Ngôn ngữ MIPS có cú pháp và số lượng câu lệnh hạn chế so với các ngôn ngữ lập trình bậc cao, có gặp chút khó khăn trong việc thực hiện các phép tính phức tạp hoặc xử lý chuỗi dài với các lệnh MIPS hạn chế.
- **Khả năng xử lý lỗi nhập liệu**: Bài toán yêu cầu dữ liệu đầu vào phải ở dạng số ASCII. Việc đảm bảo chương trình hoạt động chính xác khi gặp phải các ký tự không mong muốn hoặc dữ liệu đầu vào không đúng chuẩn là một thách thức lớn.
- **Quản lý bộ nhớ thủ công**: MIPS không có quản lý bộ nhớ tự động như các ngôn ngữ bậc cao. Việc thao tác thủ công trên các thanh ghi và bộ nhớ yêu cầu phải hiểu rõ cấu trúc hệ thống và quản lý bộ nhớ chính xác để tránh lỗi.

1.4 So sánh với các bài toán tương tự

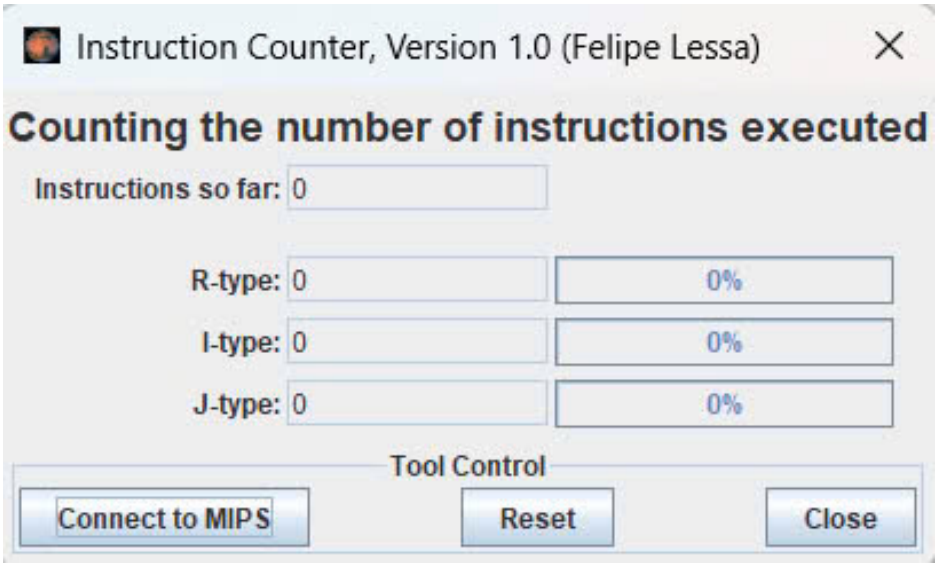
- **So với biểu đồ cột (bar chart) trong ngôn ngữ bậc cao**: Tạo phổ sao bằng MIPS tương tự như việc in biểu đồ cột trong Python hoặc C.
- **So với các bài toán xử lý chuỗi đơn giản**: Các bài toán xử lý chuỗi ASCII thông thường không yêu cầu nhiều về trực quan hóa. Với "phổ sao," việc tạo hình ảnh trực quan đòi hỏi sinh viên không chỉ xử lý chuỗi mà còn tính toán vị trí của các ký tự trên màn hình, mang lại một mức độ phức tạp cao hơn.

2 Giải pháp hiện thực

- **Bước 1:** Đọc chuỗi đầu vào từ tập tin STRING.TXT để xác định số lượng input thông qua hàm READ_FILE.
- **Bước 2:** Với từng input, xác định độ dài của chuỗi kí số nhập vào bằng hàm GET_LENGTH để thực hiện in ra phổ sao tương ứng với hàm CREATE_OUTPUT, đồng thời cũng kiểm tra các kí tự khác với kí tự số bằng hàm Character_is_invalid để dừng chương trình đang thực thi cũng như in ra kí tự không hợp lệ đó.
- **Bước 3:** Ghi lần lượt từng chuỗi input, phổ sao tương ứng của input đó vào tập tin PHO_SAO.TXT và các output sẽ được phân tách với nhau bằng một dòng chứa các dấu '-'. Đồng thời, sau khi mỗi input được ghi thành công sẽ hiện ra một dòng tin "Done testcase #-n" để thông báo cho người dùng biết testcase thứ n đã được thực hiện thành công.

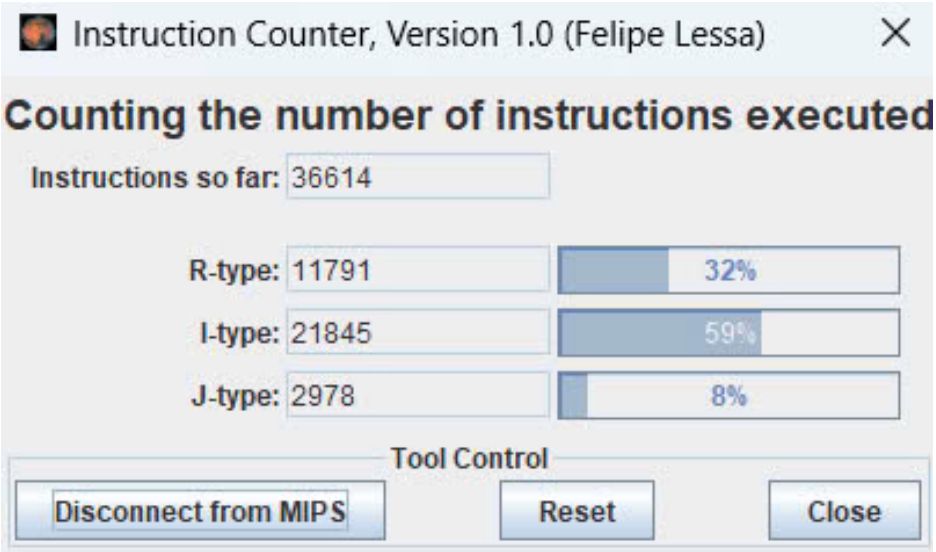
3 Thống kê số lệnh

Để có thể thống kê tổng số lệnh đã được thực thi trong chương trình cũng như số lệnh của từng nhóm lệnh, ta sử dụng chức năng Instruction Counter trong mục Tools.



Hình 1: Chức năng Instruction Counter trong MARS MIPS

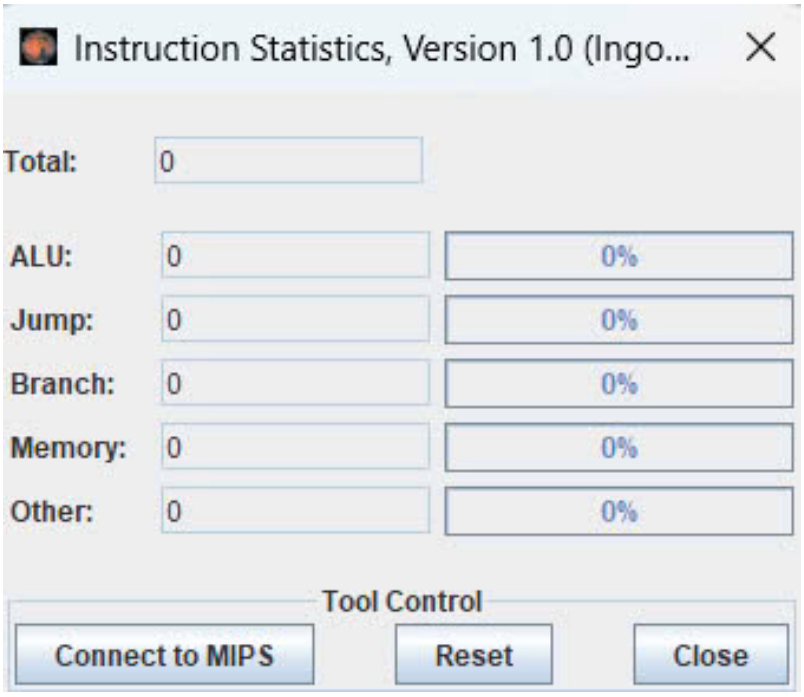
Sau khi thực hiện Connect to MIPS và chạy chương trình, ta có được tổng số lệnh cũng như số lượng từng nhóm lệnh như sau:



Hình 2: Thống kê số lượng lệnh của chương trình

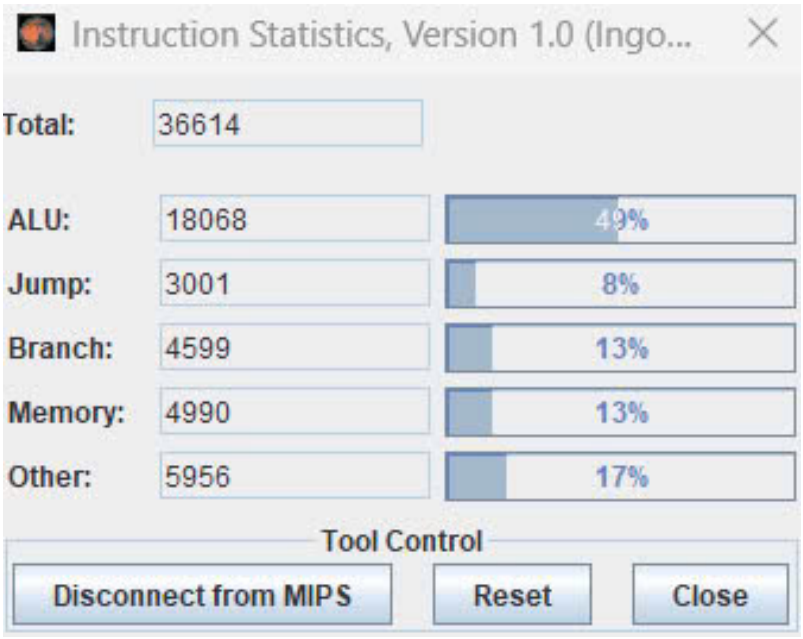
Có thể thấy được rằng, tổng số lệnh được thực thi trong chương trình là 36614 lệnh, trong đó có 11791 lệnh thuộc R-type, 21845 lệnh thuộc I-type và 2978 lệnh thuộc J-type.

Bên cạnh đó, ta cũng có thể sử dụng chức năng Instruction Statistics trong cùng mục Tools để thống kê tổng số lệnh cũng như phân loại chúng theo từng nhóm chức năng



Hình 3: Chức năng Instruction Statistics trong MARS MIPS

Tương tự với chức năng Instruction Counter, sau khi thực hiện Connect to MIPS và chạy chương trình, ta thu được kết quả như sau:



Hình 4: Chức năng Instruction Statistics trong MARS MIPS

4 Tính thời gian chạy của chương trình

Giả sử các lệnh trong chương trình có CPI trung bình bằng 1, kết hợp với tổng số lệnh được thực thi trong chương trình là 36614, ta có thể tính được thời gian thực thi của chương trình như sau:

$$\text{Tổng số chu kỳ} = \text{Tổng số lệnh} \times \text{CPI trung bình} = 36614 \times 1 = 36614 \text{ (Chu kỳ)}$$

$$\text{Thời gian thực thi} = \frac{\text{Tổng số chu kỳ}}{\text{Clock rate}} = \frac{36614}{1\,\text{GHz}} = 36.614\,(\mu\text{s})$$

Lưu ý: Số lượng lệnh cũng như thời gian chạy chương trình trên áp dụng cho 10 testcase được sử dụng trong tập tin STRING.TXT. Các trường hợp ngoại lệ bên dưới sẽ được thực thi riêng và không bao gồm trong tập tin STRING.TXT

5 Hướng dẫn chạy chương trình

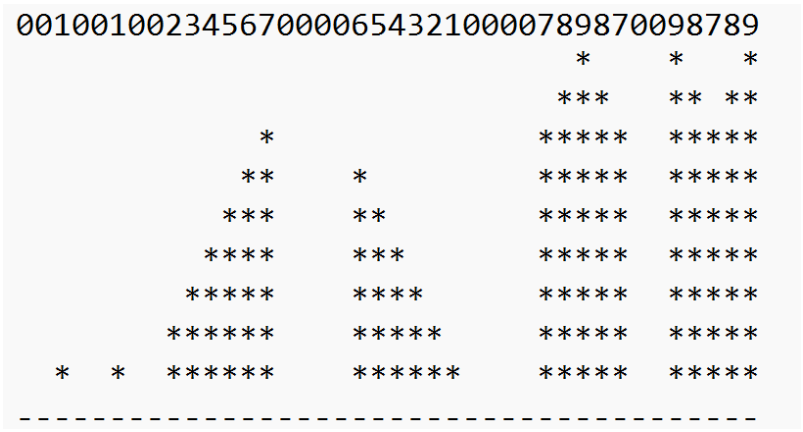
Để có thể thực thi chương trình nhằm đọc được file đầu vào cũng như ghi kết quả vào file đầu ra, ta thực hiện theo các bước sau

- Tạo 2 file trống STRING.TXT và PHO_SAO.TXT và đặt chúng vào chung thư mục nơi chứa phần mềm mô phỏng MARS MIPS.
- Nhập dữ liệu đầu vào vào file STRING.TXT.
- Mở MARS MIPS và thực thi chương trình.

6 Kết quả kiểm thử

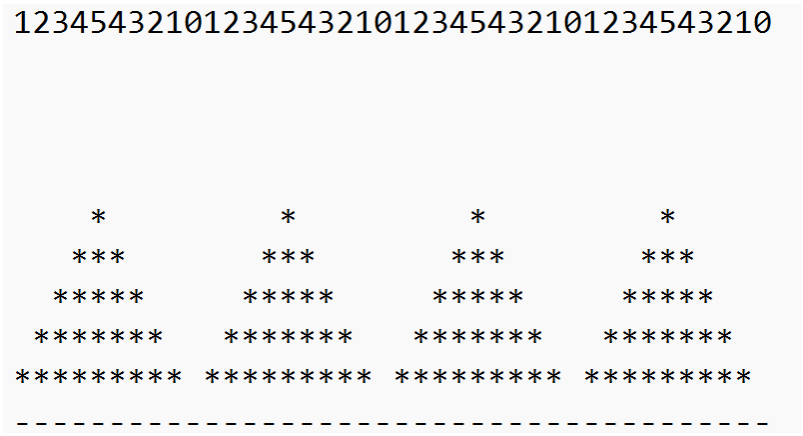
Với dữ liệu đầu vào chuẩn trong tập tin STRING.TXT gồm 10 testcase, ta có kết quả lần lượt là:

- **Testcase 01:** 00100100234567000065432100007898700987 89:



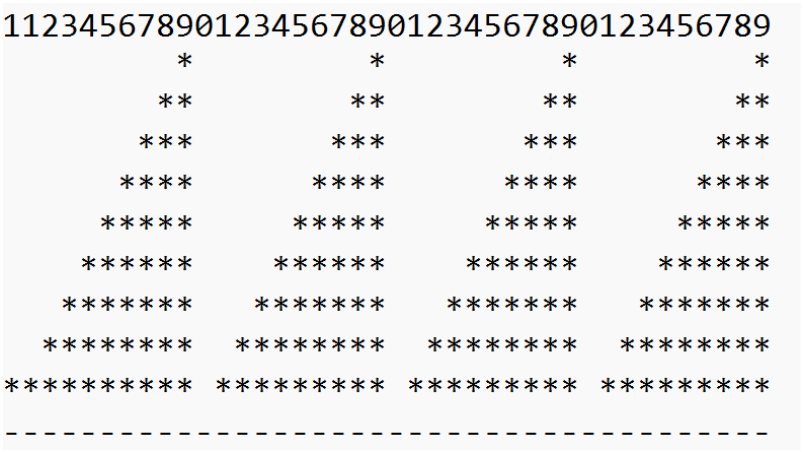
Hình 5: Kết quả kiểm thử testcase 01

- **Testcase 02:** 12345432101234543210123454321012345432 10:



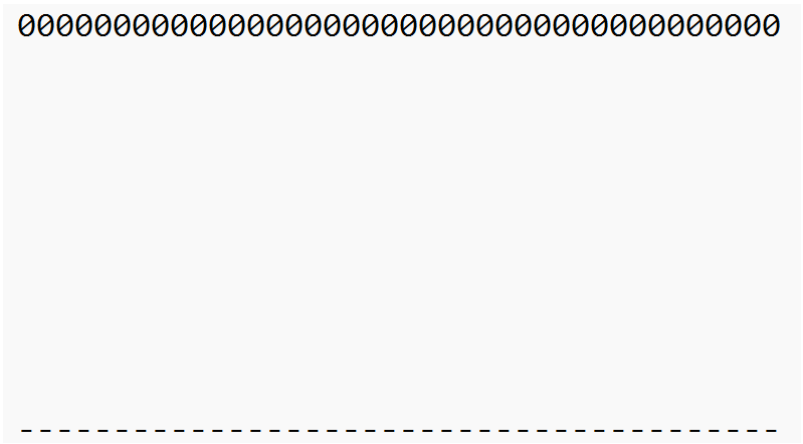
Hình 6: Kết quả kiểm thử testcase 02

- **Testcase 03:** 12345432101234543210123454321012345432 10:



Hình 7: Kết quả kiểm thử testcase 03

- **Testcase 04:** 0000000000000000000000000000000000000000:



Hình 8: Kết quả kiểm thử testcase 04

- **Testcase 05:** 9999999999999999999999999999999999999999:

```
9999999999999999999999999999999999999999
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************
****************************************

----------------------------------------
```

Hình 9: Kết quả kiểm thử testcase 05

- **Testcase 06:** 1111111111111111111111111111111111111111:

```
1111111111111111111111111111111111111111








****************************************
----------------------------------------
```

Hình 10: Kết quả kiểm thử testcase 06

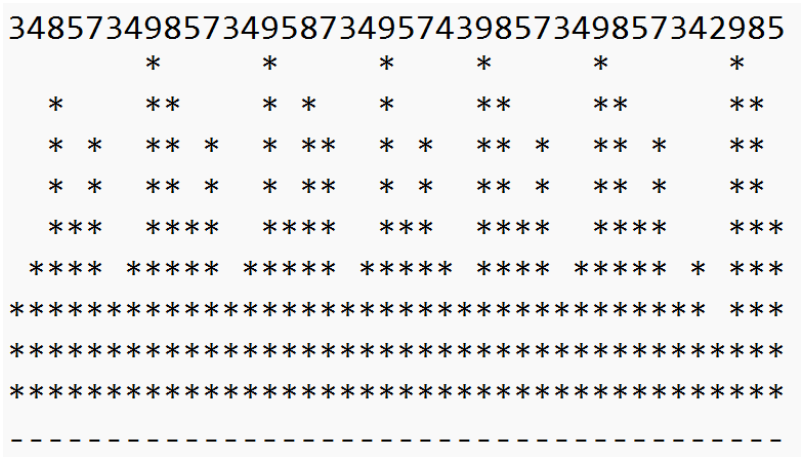
- **Testcase 07:** 4572435094238753940578324234098573429058:

```
4572435094238753940578324234098573429058
        *       *            *      *
        *   *   *    *       **     *  *
  *     *   **  *   **       ** *   *  *
  *     *   **  *   **       ** *   *  *
 **   * *   *** *  ***       ****   * **
*** * * **  *** ** ***  *  * **** * * **
*** *** ** ******* **** * ** ****** * **
******* ********** ********* ******** **
******* ********** ********* ******** **

----------------------------------------
```

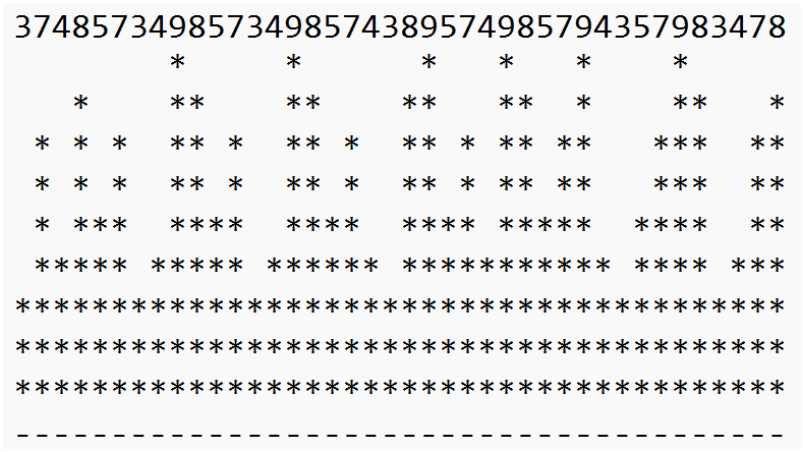
Hình 11: Kết quả kiểm thử testcase 07

- **Testcase 08:** 34857349857349587349574398573498573429 85:



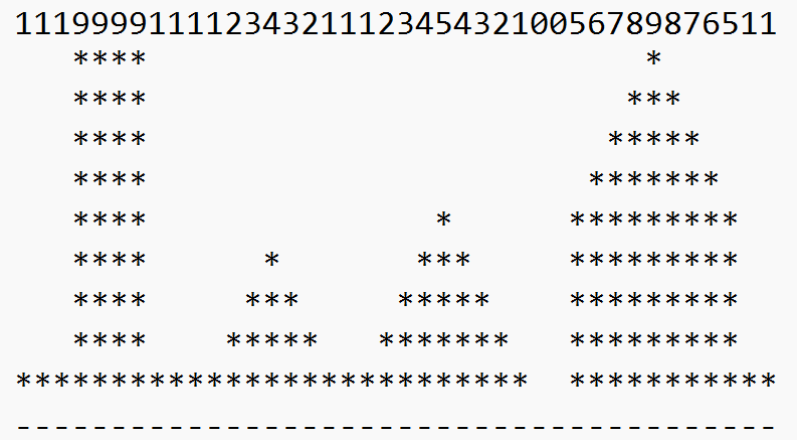
Hình 12: Kết quả kiểm thử testcase 08

- **Testcase 09:** 37485734985734985743895749857943579834 78:



Hình 13: Kết quả kiểm thử testcase 09

- **Testcase 10:** 11199991111234321112345432100567898765 11:



Hình 14: Kết quả kiểm thử testcase 10

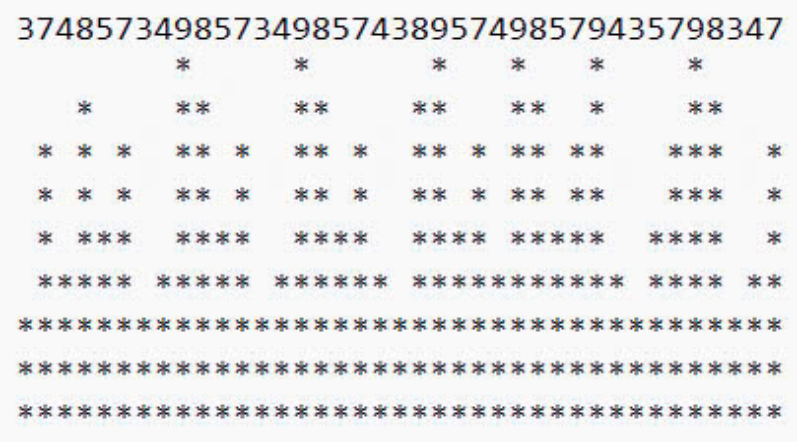
7 Một số trường hợp khác

Ngoài việc kiểm thử với dữ liệu đầu vào chuẩn, ta có thể kiểm tra với một số dữ liệu đầu vào khác để đảm bảo chương trình thực thi chính xác hơn:

- **Trường hợp thiếu kí số:**
  - Ở trường hợp này, độ dài chuỗi nhỏ hơn 40, nhóm đã cải thiện đoạn Code để chạy được trường hợp này: Đầu vào có bao nhiêu kí tự thì đầu ra có bấy nhiêu số lượng cột được in ra.
  - Với input: 3984573948753948573, ta có kết quả như sau:

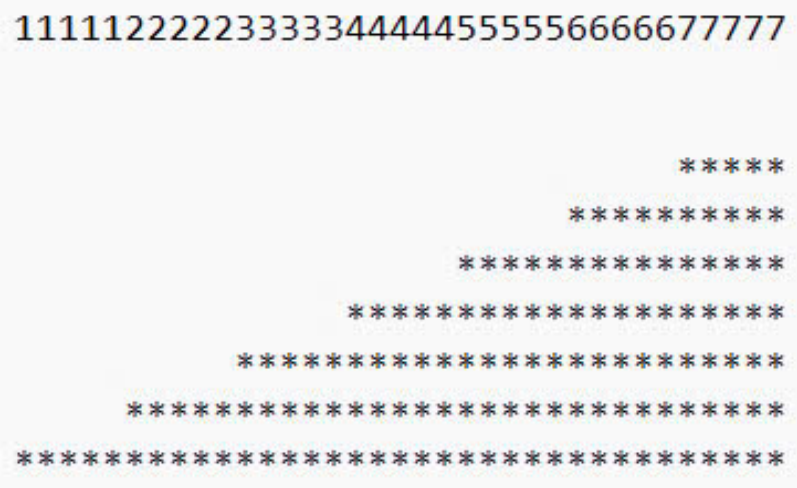
Hình 15: Trường hợp thiếu kí số_test01

- Với input gồm 39 kí tự 374857349857349857438957498579435798347, ta có kết quả như sau:



Hình 16: Trường hợp thiếu kí số_test02

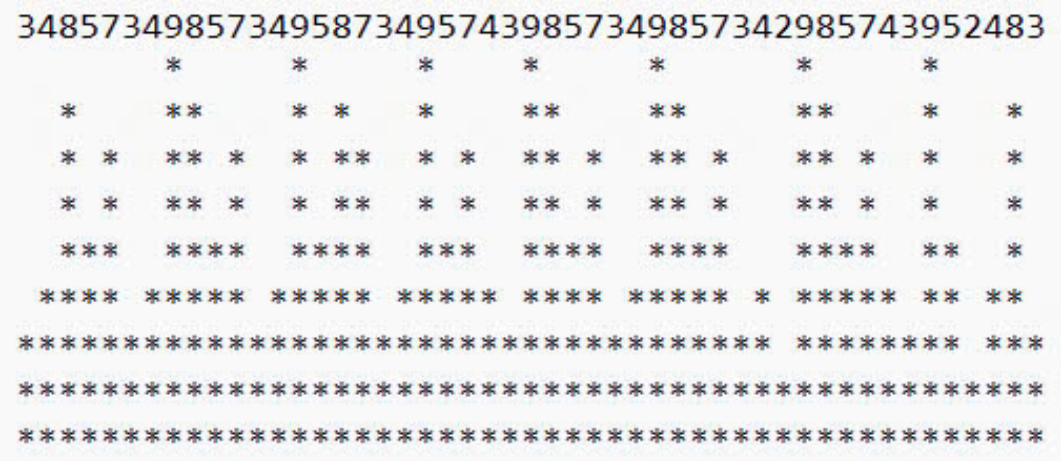
- Với input gồm 35 kí tự 11111222223333344444555556666677777, ta có kết quả như sau:



Hình 17: Trường hợp thiếu kí số_test03

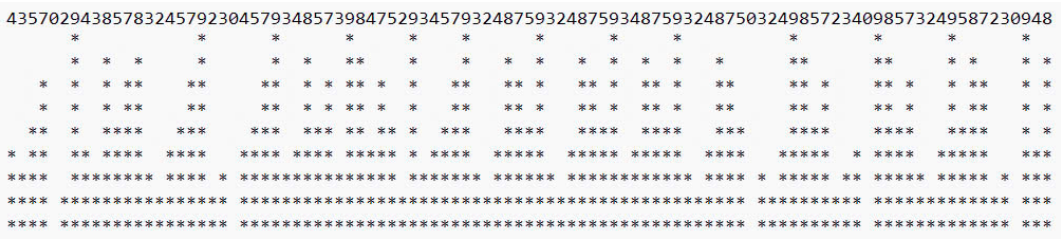
- **Trường hợp thừa kí số:**

- Với input gồm 49 kí tự 34857349857349587349574398573498573429857439524 83, ta có kết quả như sau:



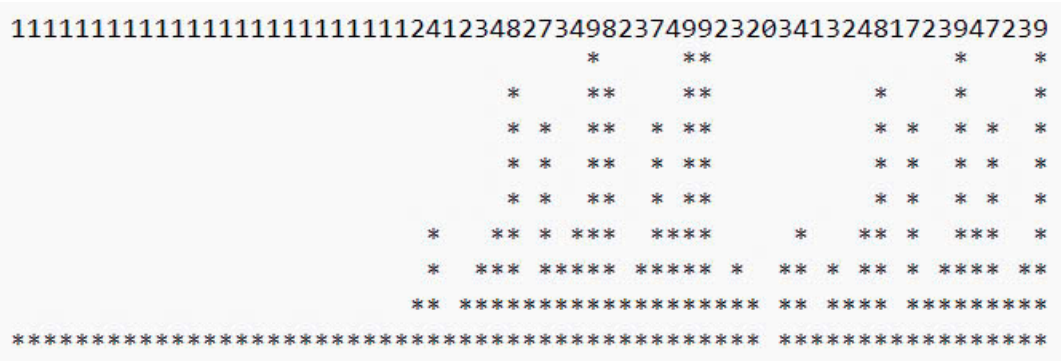
Hình 18: Trường hợp thừa kí số_test01

- Với input như trong hình, ta có:



Hình 19: Trường hợp thừa kí số_test02

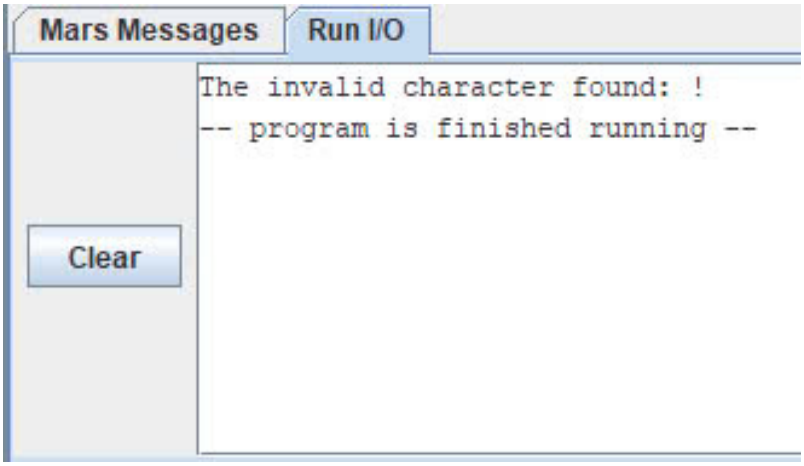
- Với input: `11111111111111111111111112412348273498237499232034132481723947239`, ta có kết quả như sau:



Hình 20: Trường hợp thừa kí số_test03

- **Trường hợp input chứa kí tự khác:**

  Với input: `12345!!!@@@67890`, chương trình sẽ in ra kí tự đầu tiên không hợp lệ, và dừng không chạy nữa, đồng thời không in ra bất kì phổ sao trước đó nếu có.



Hình 21: Kết quả khi phát hiện kí tự không hợp lệ

8 Kết luận

- Bài tập tạo hình "phổ sao" bằng chuỗi ASCII giúp ta thực hành khả năng xử lý chuỗi, thao tác file và quản lý dữ liệu đầu ra dưới dạng đồ họa ký tự trong môi trường lập trình MIPS.
- Qua bài này, ta đã học được cách đọc và ghi file, đồng thời ứng dụng phương pháp xử lý chuỗi đơn giản để tạo ra hình ảnh ASCII. Ngoài ra, bài tập cũng giúp ta hiểu thêm về thống kê lệnh và tối ưu hóa chương trình thông qua các công cụ hỗ trợ như Instruction Counter và Instruction Statistics trong MARS MIPS.
- Bằng cách thử nghiệm với nhiều đầu vào khác nhau, ta cũng học được cách xử lý các trường hợp đặc biệt như thiếu hoặc thừa ký tự, cũng như quản lý các ký tự không hợp lệ. Đây là bài học hữu ích trong việc lập trình ứng dụng và xử lý dữ liệu trực quan trên các thiết bị hạn chế về tài nguyên.