**BÁO CÁO THỰC HÀNH**

**IT3280 – 156788 – THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

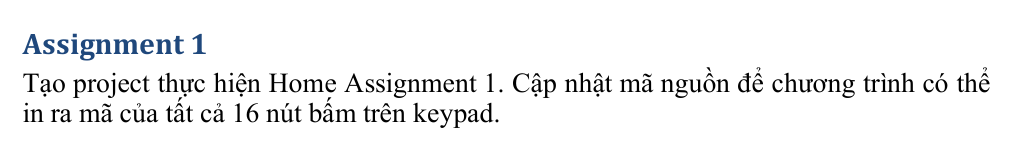
**NỘI DUNG**

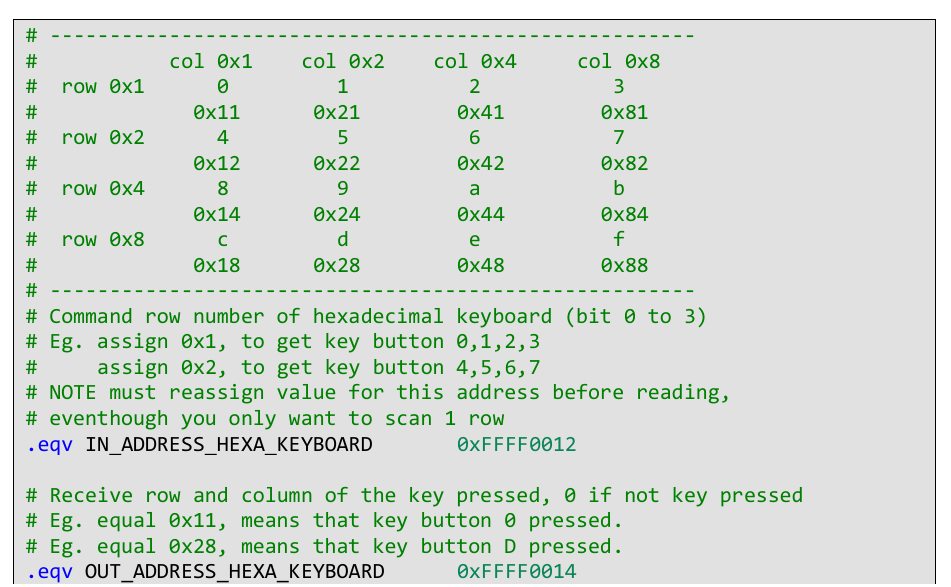
**Bài 11. Lập trình xử lý ngắt**

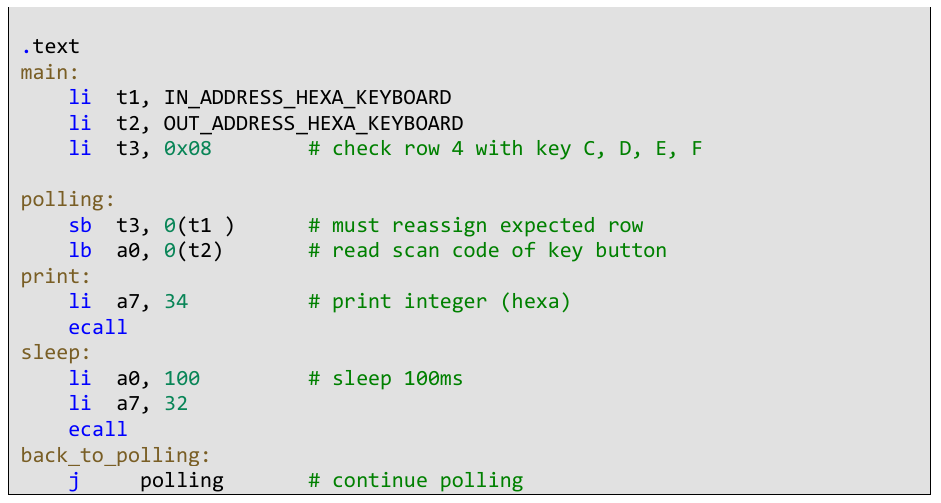
|  |  |
| --- | --- |
| **Họ và tên** | Nguyễn Minh Quân |
| **Mã số sinh viên** | 20235816 |

**Assignment 1**

**1. Yêu cầu**



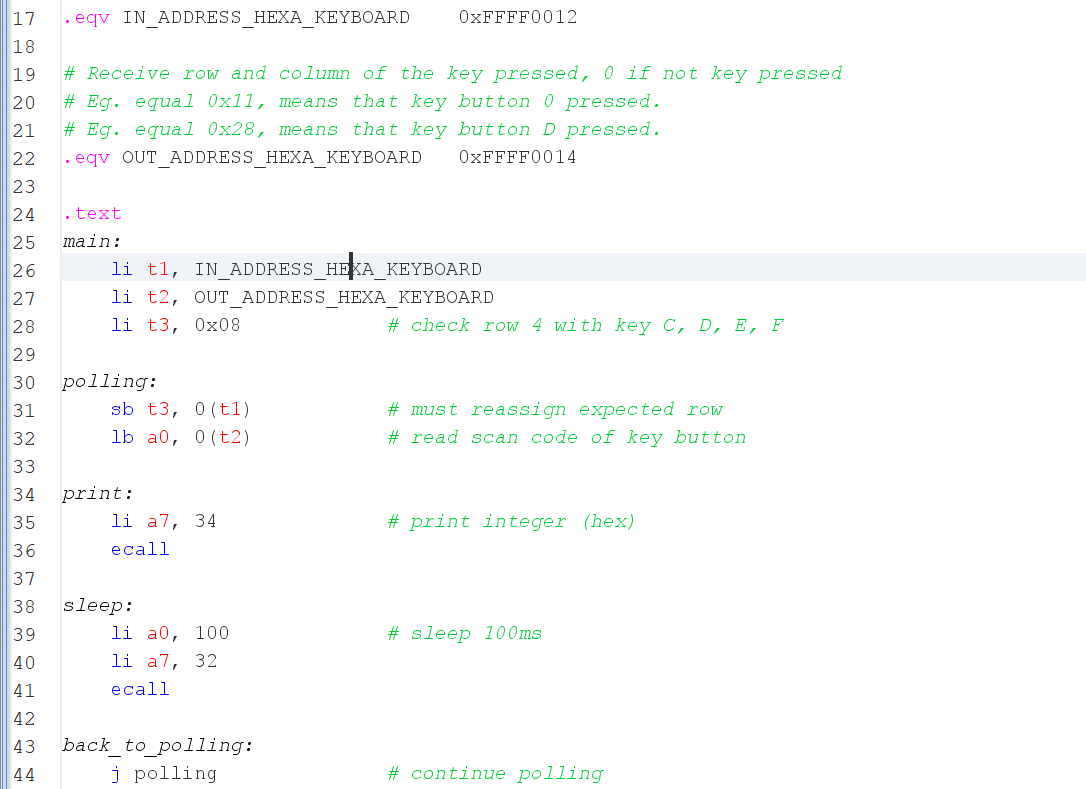




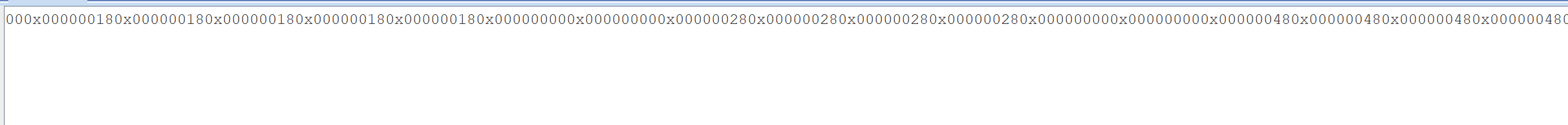
**2. Thực hiện**

a. Chạy chương trình ban đầu

\* Chương trình

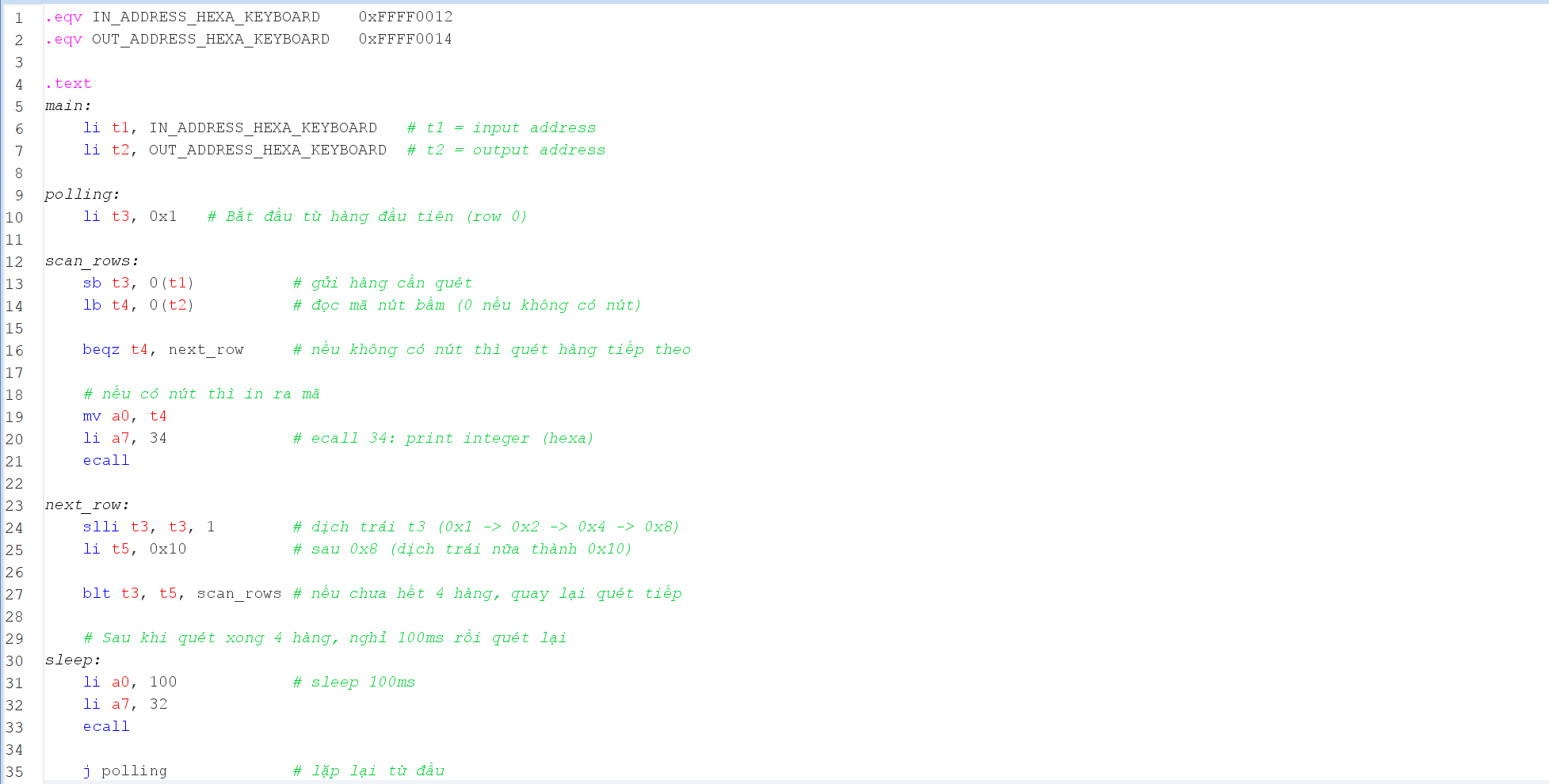


\* Kết quả

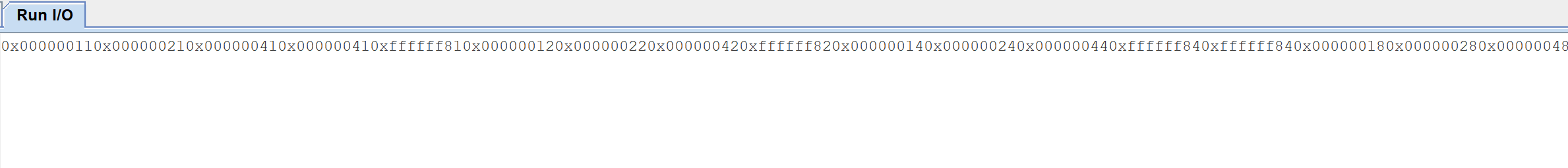


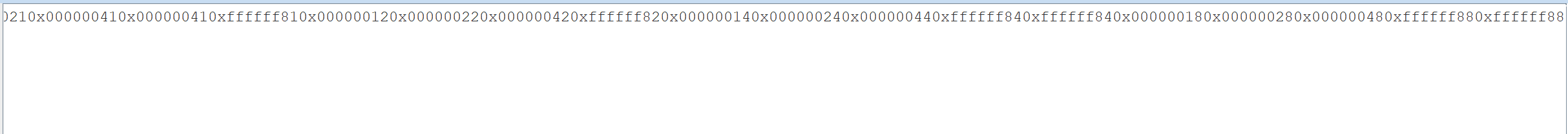
b. In ra mã 16 nút bấm trên keypad

\* Chương trình:



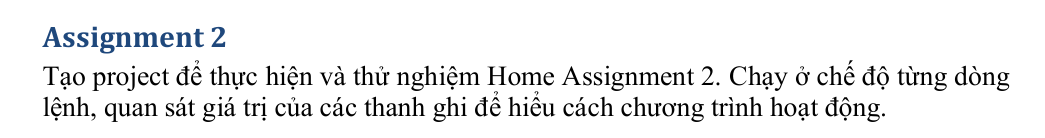
\* Kết quả





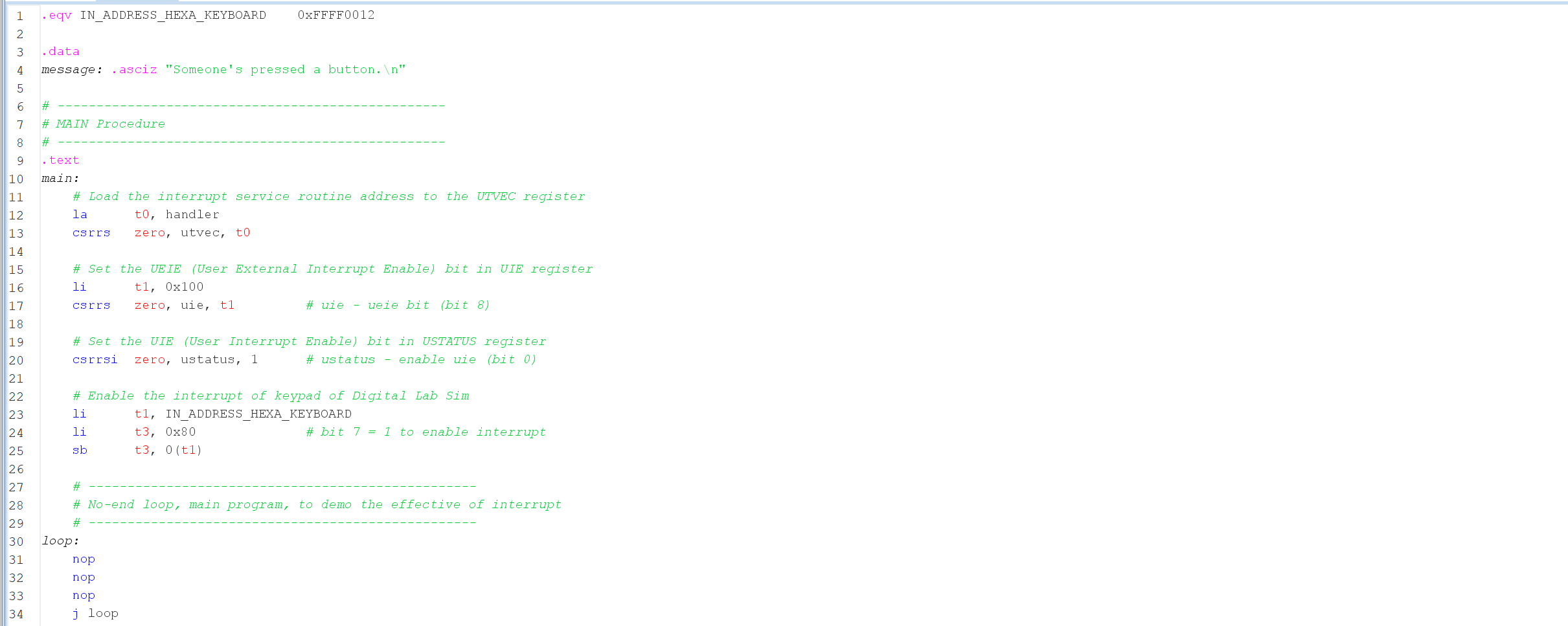
**Assignment 2**

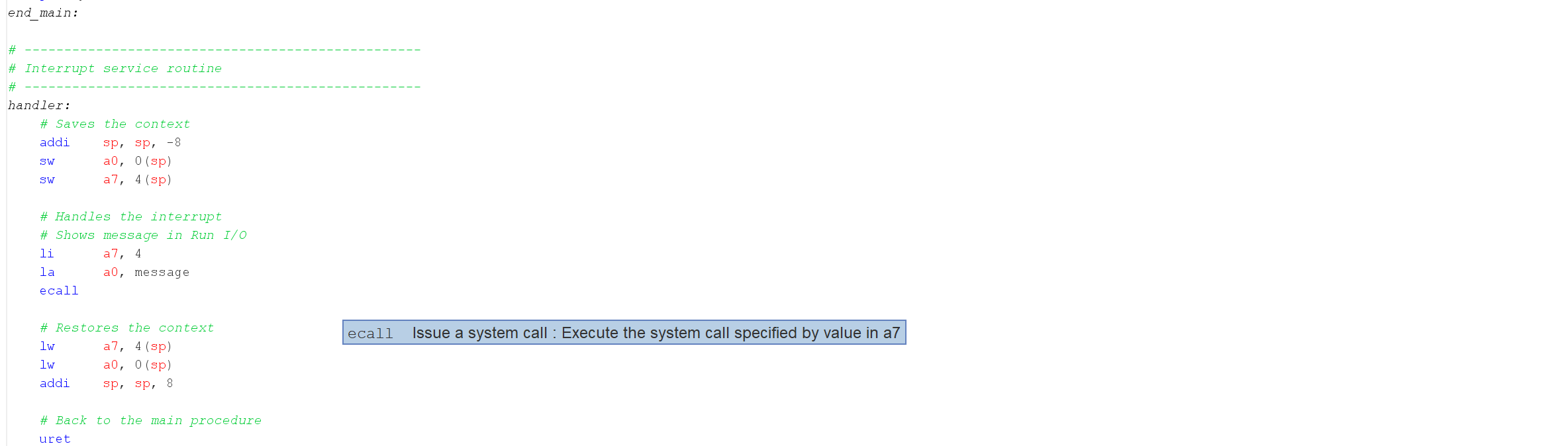
**1. Yêu cầu**



**2. Thực hiện**

a. Chương trình





b. Các giá trị thanh ghi và cách hoạt động

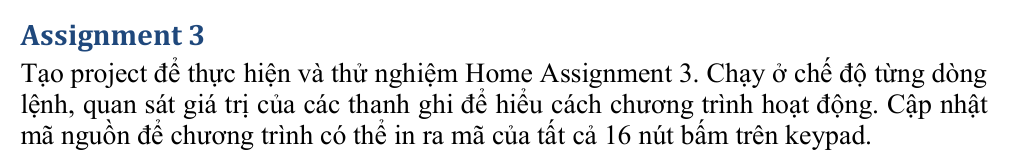
- Trong main: thiết lập môi trường interrupt ➔ chỉ t0, t1, t3 thay đổi.

- Trong handler: lưu và phục hồi context ➔ sp, a0, a7 thay đổi tạm thời để xử lý ngắt.

|  |  |  |
| --- | --- | --- |
| **Thanh ghi** | **Vị trí thay đổi** | **Ý nghĩa / Công dụng** |
| t0 | main | Nạp địa chỉ handler vào t0 để gán cho utvec |
| t1 | main | Nạp các giá trị cấu hình enable interrupt (0x100) và địa chỉ bàn phím |
| t3 | main | Nạp giá trị 0x80 để enable interrupt bàn phím |
| sp | handler | Lưu và khôi phục ngữ cảnh (dịch con trỏ stack) |
| a0 | handler | Lưu giá trị cũ vào stack + nạp địa chỉ message để in ra |
| a7 | handler | Lưu giá trị cũ vào stack + nạp số dịch vụ hệ thống ecall (dịch vụ in chuỗi) |

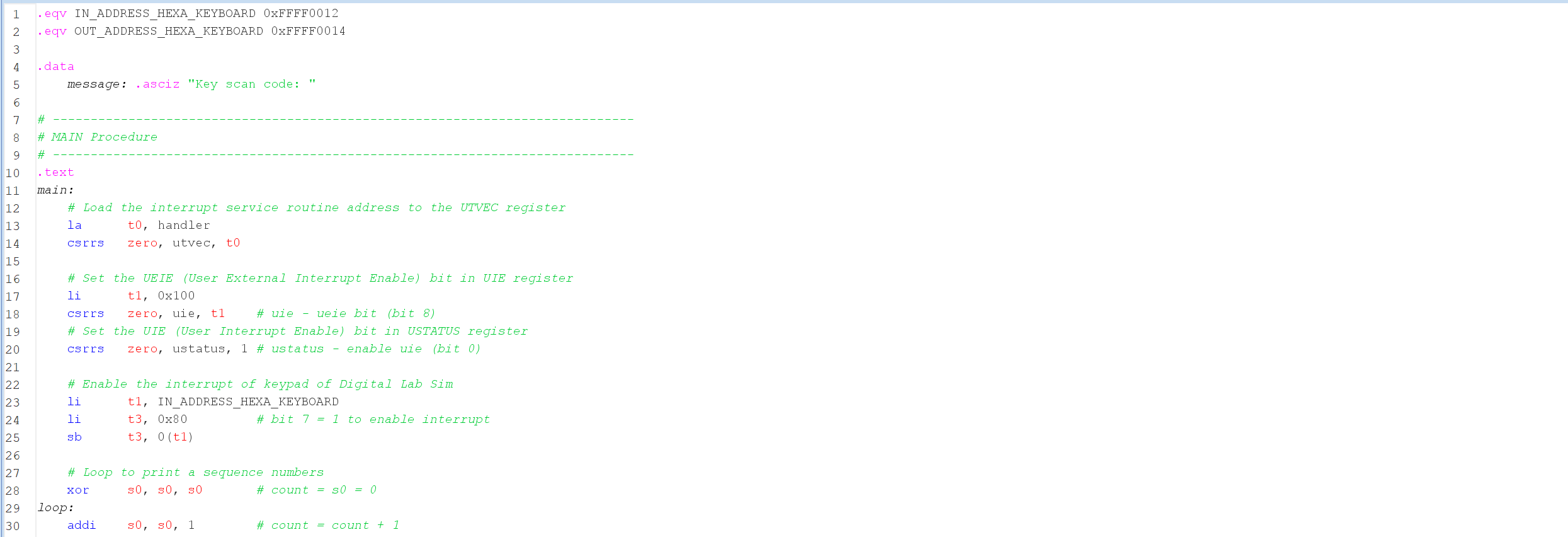
**Assignment 3**

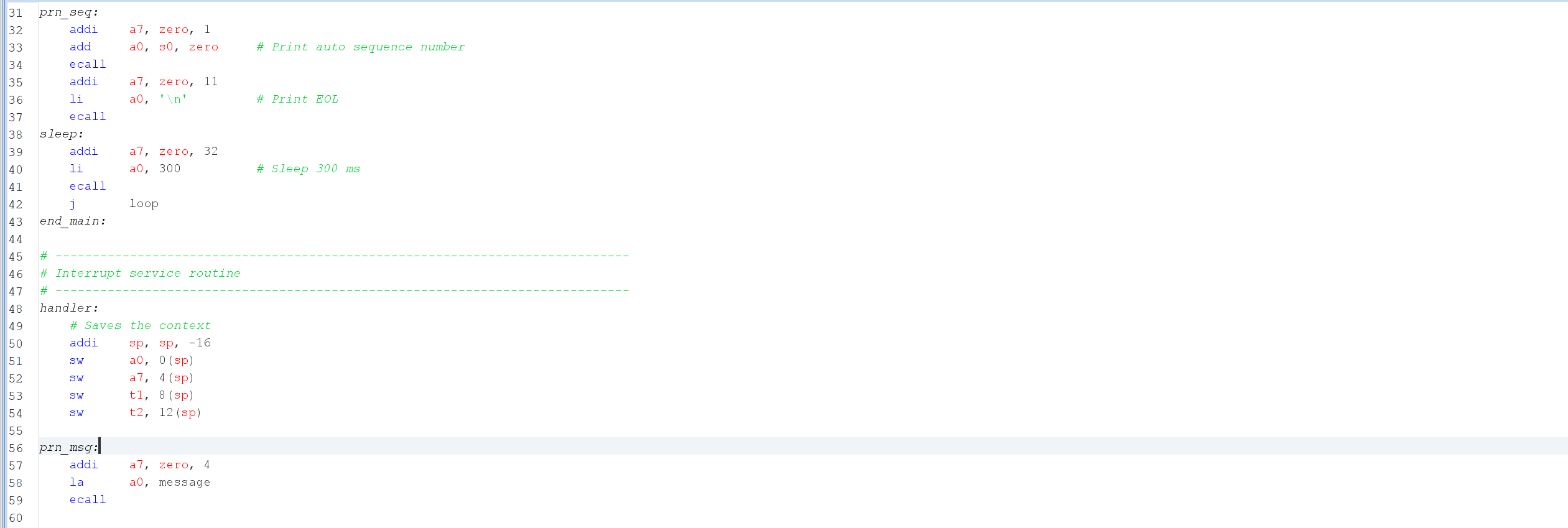
**1. Yêu cầu**



**2. Thực hiện**

a. Chương trình







b. Sự thay đổi các thanh ghi

- t0: lưu địa chỉ handler

- t1: địa chỉ thiết bị ngoại vi (IN/OUT bàn phím)

- t2: đọc giá trị dòng bàn phím

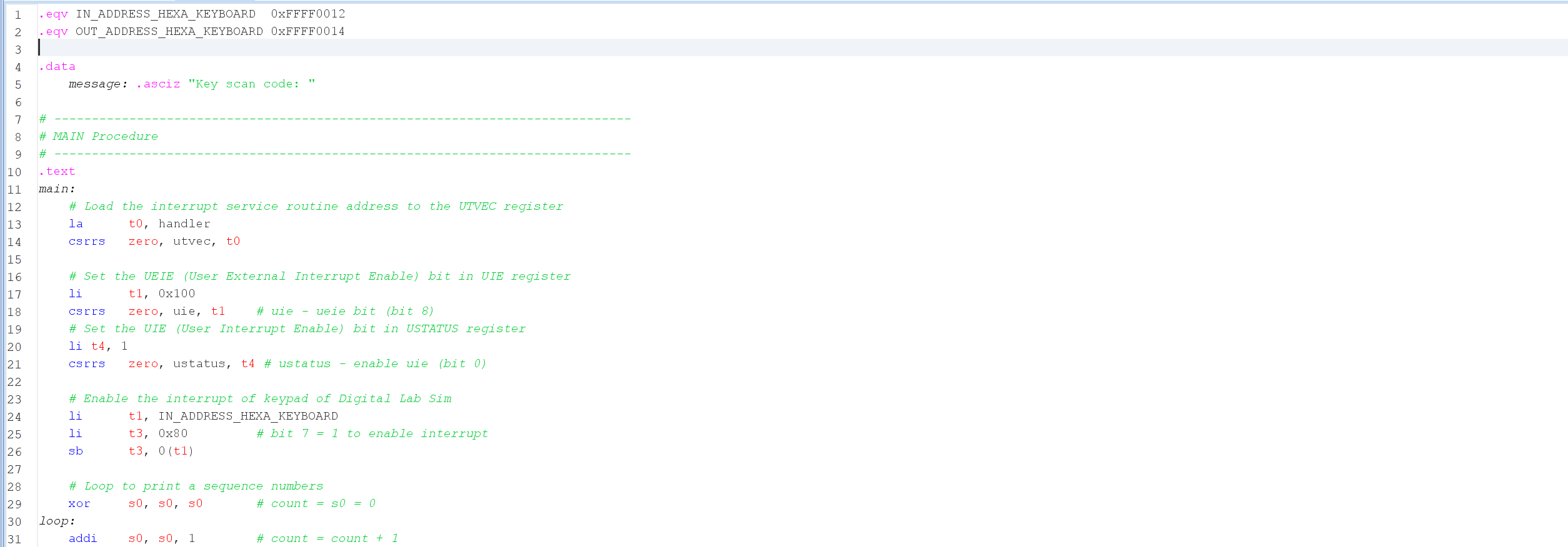
- a0: dữ liệu cần in (số, chuỗi, mã phím)

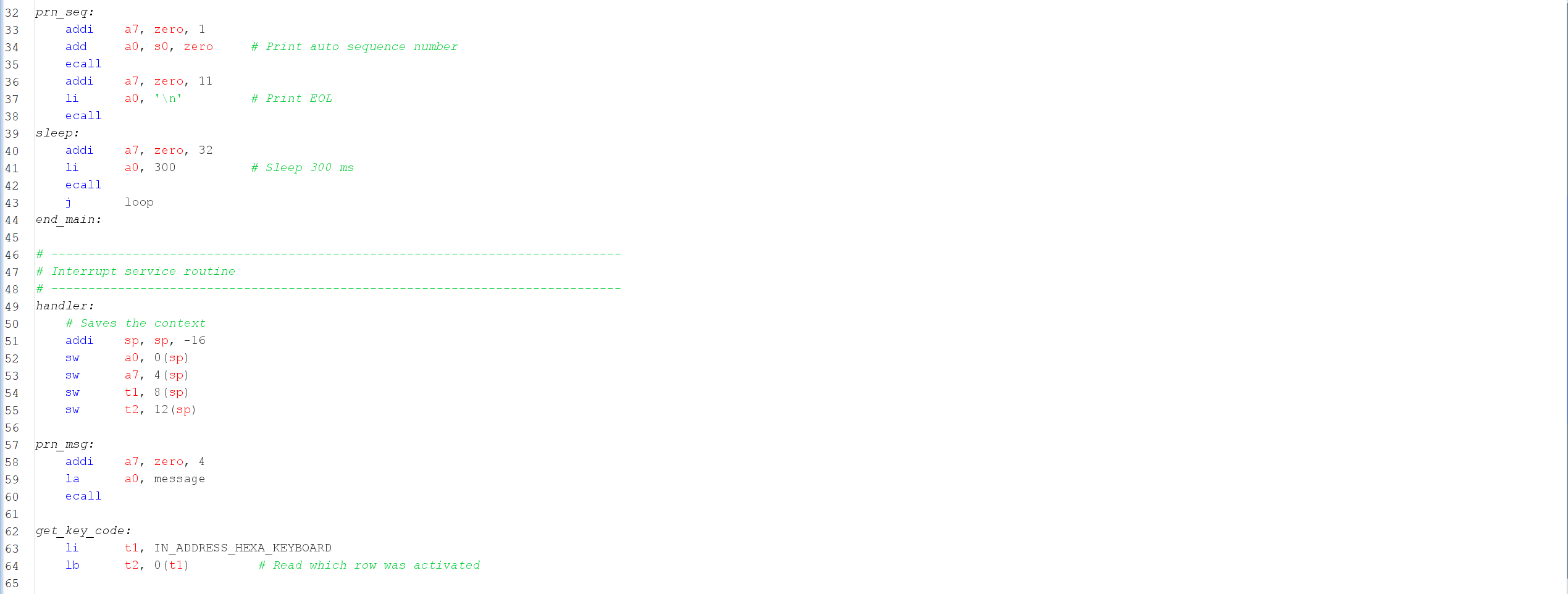
- a7: syscall code (ecall)

- s0: biến đếm số vòng lặp

- sp: thay đổi khi lưu/phục hồi context

c. In ra mã 16 nút trên keypad

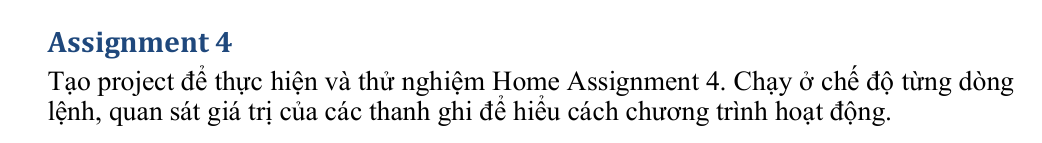






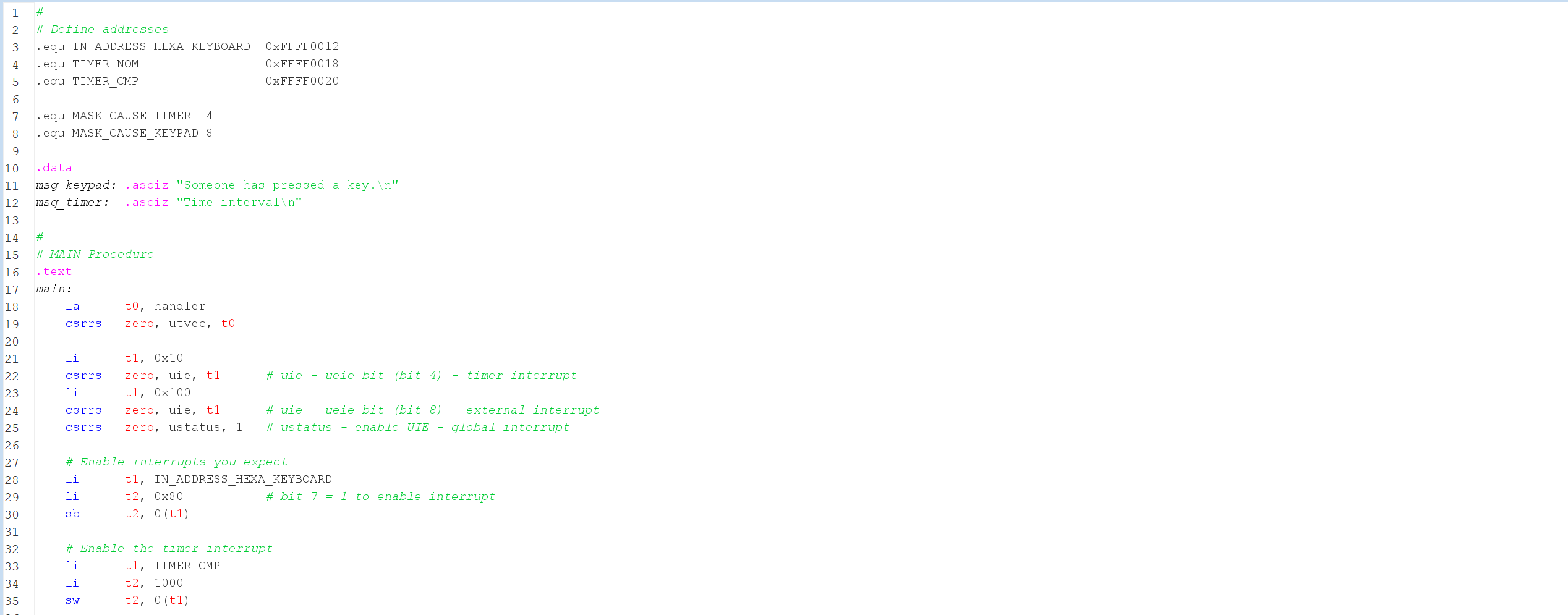
**Assignment 4**

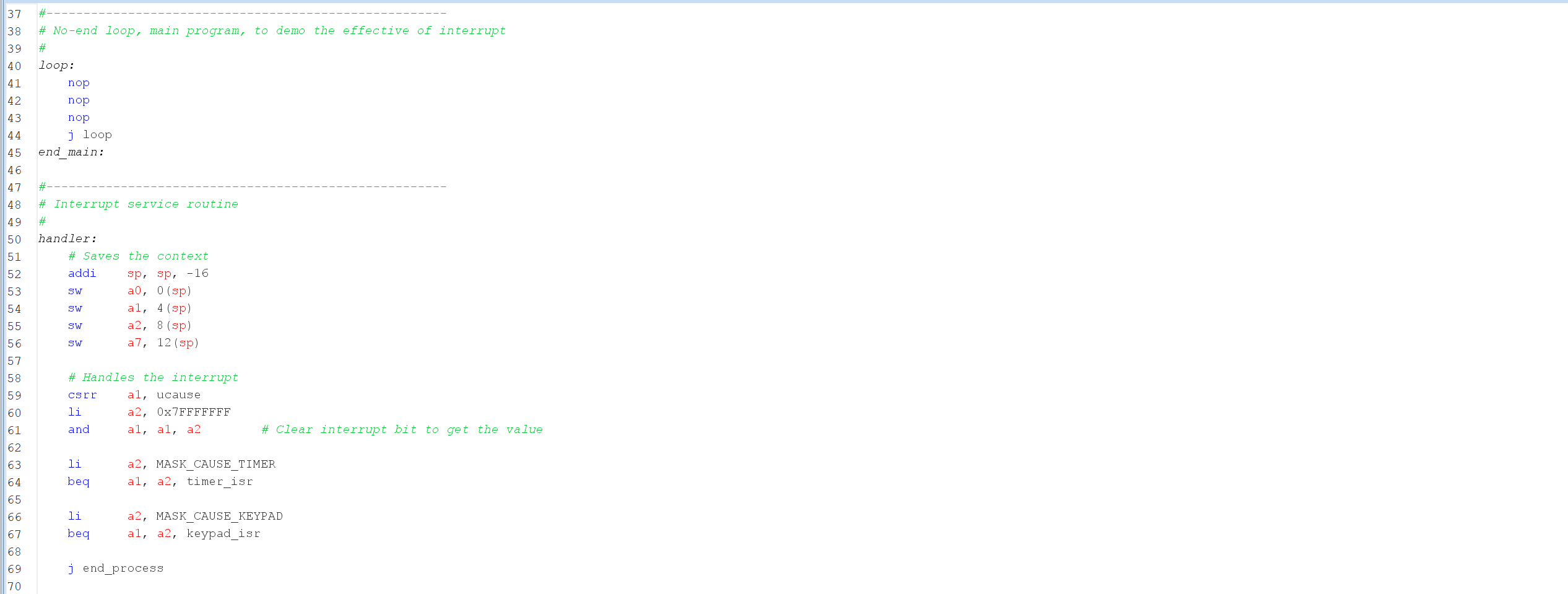
**1. Yêu cầu**



**2. Thực hiện**

a. Chương trình







b. Các giá trị thanh ghi và cách hoạt động

\* Thay đổi thanh ghi

|  |  |  |
| --- | --- | --- |
| **Thanh ghi** | **Vai trò trong bài** | **Khi thay đổi** |
| utvec | Địa chỉ hàm ngắt | Được thiết lập lúc khởi động |
| uie | Cho phép ngắt timer + keypad | Được bật trong main |
| ustatus | Bật toàn bộ user interrupt | Được bật trong main |
| ucause | Xác định loại ngắt xảy ra | Đọc và phân tích trong handler |
| sp | Lưu phục hồi context | Giảm đi/lấy lại 16 bytes |
| a0 a1 a2 a7 | Dùng tạm cho ecall và tính toán | Bị save/restore trong handler |

\* Cách hoạt động

 **Khởi động chương trình**:

* Cài đặt địa chỉ hàm ngắt (handler) vào thanh ghi utvec.
* Bật cho phép ngắt timer + bàn phím (uie, ustatus).
* Bật ngắt riêng cho bàn phím hexa và lập lịch timer lần đầu (CMP = 1000).

 **Vào vòng lặp loop:**:

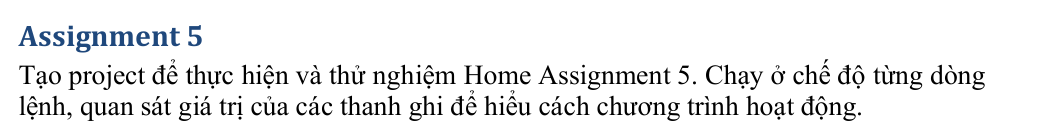
* Chương trình chính chỉ nop (không làm gì), chờ có ngắt xảy ra.

 **Khi có ngắt** (timer hoặc phím):

* CPU tự động nhảy vào handler.
* handler sẽ:
  + Lưu trạng thái thanh ghi (a0, a1, a2, a7).
  + Đọc nguyên nhân ngắt từ ucause.
  + Nếu:
    - Ngắt timer ➔ in "Time interval", cộng thêm 1000 đơn vị vào timer.
    - Ngắt bàn phím ➔ in "Someone has pressed a key!".
  + Phục hồi trạng thái thanh ghi.
  + Trở lại chương trình chính bằng uret.

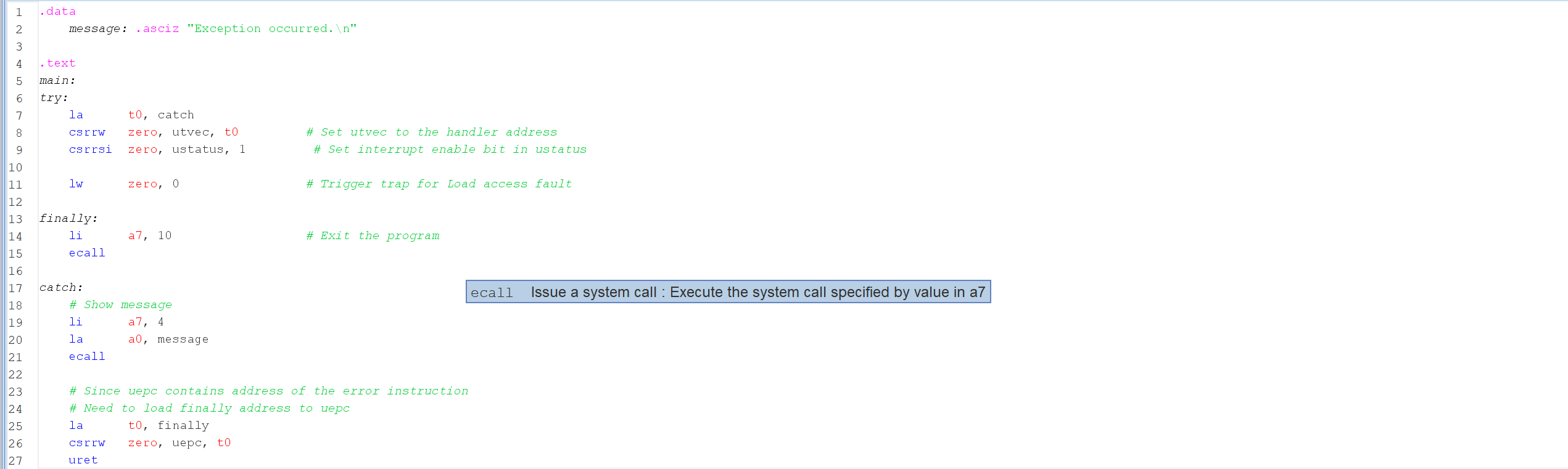
**Assignment 5**

**1. Yêu cầu**



**2. Thực hiện**

a. Chương trình



b. Các giá trị thanh ghi và cách hoạt động

\* Thay đổi của các thanh ghi:

- t0: Dùng để lưu trữ địa chỉ của các nhãn (như catch, finally). Nó sẽ được thay đổi khi thiết lập ngoại lệ và khi thay đổi uepc.

- a0: Dùng để lưu trữ địa chỉ chuỗi thông báo ngoại lệ (message) khi gọi ecall để in thông báo.

- a7: Dùng để chỉ thị hệ thống gọi (syscall). Trong trường hợp này, nó sẽ được đặt thành 4 để gọi lệnh in chuỗi và 10 để gọi lệnh thoát.

- utvec: Lưu trữ địa chỉ của hàm xử lý ngoại lệ. Sau khi được thiết lập, nếu có ngoại lệ xảy ra, chương trình sẽ chuyển đến địa chỉ này.

- ustatus: Lưu trữ các cờ trạng thái của hệ thống, trong đó có cờ cho phép ngắt và ngoại lệ

- uepc: Lưu trữ địa chỉ của lệnh sẽ được thực thi sau khi xử lý ngoại lệ.

\* Cách hoạt động:

**Thiết lập bắt ngoại lệ:**

* la t0, catch: Lệnh này tải địa chỉ nhãn catch vào thanh ghi t0.
* csrrw zero, utvec, t0: Lệnh này ghi giá trị trong t0 (địa chỉ của nhãn catch) vào thanh ghi utvec, thiết lập địa chỉ của bộ xử lý ngoại lệ. utvec lưu trữ địa chỉ của hàm xử lý ngoại lệ.
* csrrsi zero, ustatus, 1: Lệnh này bật bit "interrupt enable" trong thanh ghi ustatus. Điều này cho phép các ngoại lệ có thể xảy ra (nghĩa là xử lý ngắt hoặc lỗi sẽ được thực thi khi xảy ra).

**Gây ra ngoại lệ (Load access fault):**

* lw zero, 0: Lệnh này cố gắng tải dữ liệu từ địa chỉ 0 vào thanh ghi zero. Tuy nhiên, địa chỉ này không hợp lệ, và sẽ gây ra một ngoại lệ (Load access fault). Ngoại lệ này sẽ khiến chương trình nhảy đến địa chỉ đã được thiết lập trong utvec, đó là nhãn catch.

**Xử lý ngoại lệ (catch):**

* li a7, 4: Lệnh này đặt giá trị 4 vào thanh ghi a7, chỉ thị cho hệ thống gọi hệ thống (ecall) để thực hiện việc in một chuỗi (tương ứng với lệnh "write" trong syscall).
* la a0, message: Tải địa chỉ của chuỗi thông báo ngoại lệ vào thanh ghi a0, để chuỗi này có thể được in ra.
* ecall: Lệnh này gọi hệ thống, thực hiện việc in thông báo ngoại lệ.

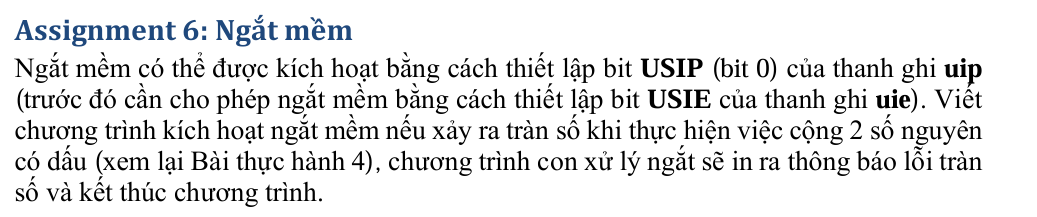
**Cuối cùng (finally):**

* la t0, finally: Tải địa chỉ của nhãn finally vào thanh ghi t0.
* csrrw zero, uepc, t0: Lệnh này thay đổi giá trị của thanh ghi uepc (User Exception Program Counter) thành địa chỉ của nhãn finally. uepc lưu trữ địa chỉ của lệnh sẽ được thực thi sau khi xử lý ngoại lệ. Địa chỉ này chính là nơi tiếp tục chương trình sau khi ngoại lệ được xử lý.
* uret: Lệnh này thực hiện việc quay lại chương trình sau khi xử lý ngoại lệ. Nó sẽ đưa chương trình quay lại tiếp tục thực thi từ địa chỉ được lưu trong uepc.

**Thoát chương trình**

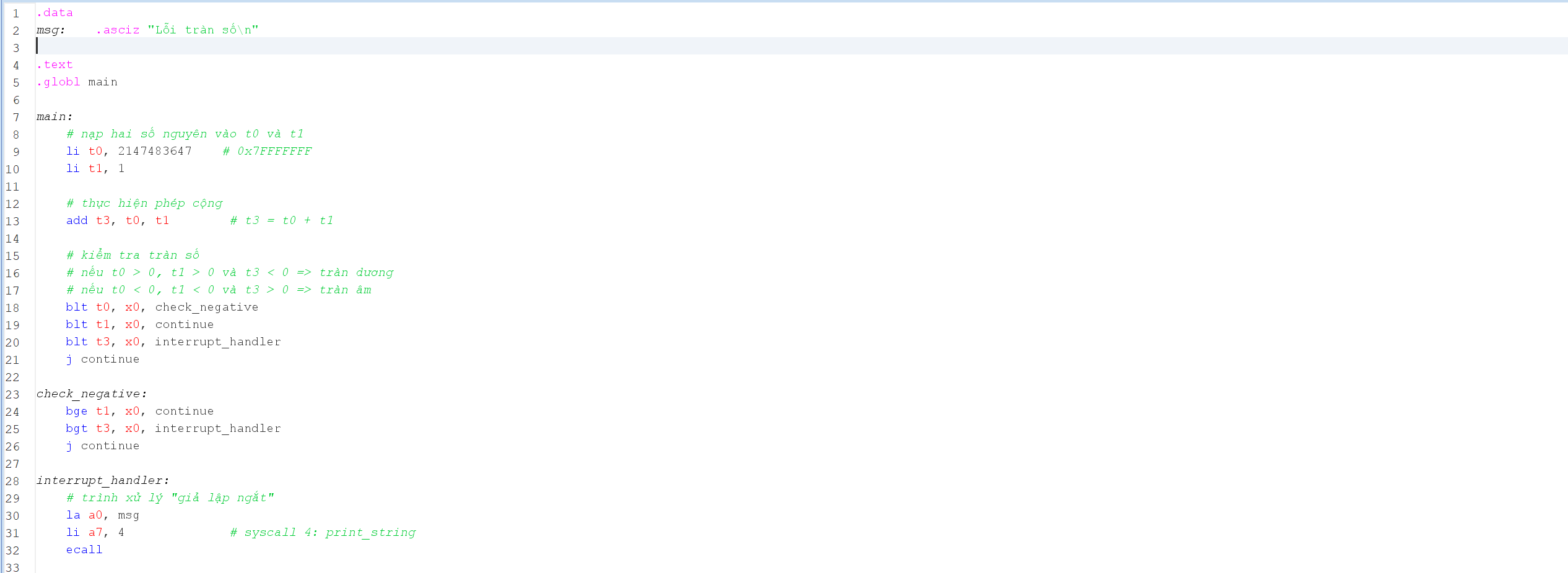
**Assignment 6**

**1. Yêu cầu**



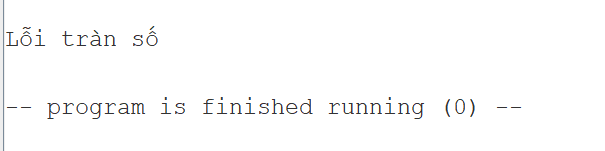
**2. Thực hiện**

a. Chương trình (đầu vào có tràn số)



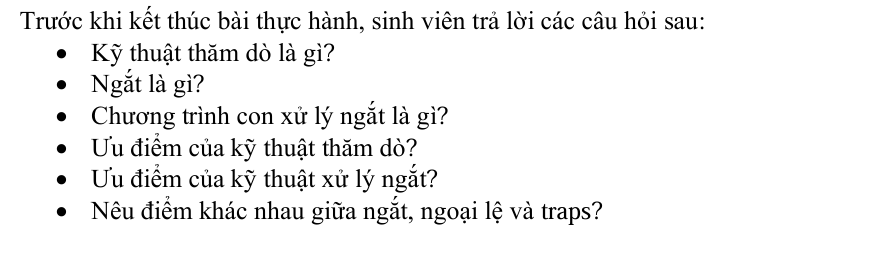


b. Kết quả



**Kết luận**

**1. Yêu cầu**



**2. Thực hiện**

- Kỹ thuật thăm dò là gì?

➔ Là cách chương trình chủ động kiểm tra liên tục trạng thái thiết bị ngoại vi.

- Ngắt là gì?

➔ Là tín hiệu yêu cầu CPU tạm dừng công việc hiện tại để xử lý một sự kiện khẩn cấp.

- Chương trình con xử lý ngắt là gì?

➔ Là đoạn mã được gọi tự động để xử lý sự kiện ngắt.

- Ưu điểm của kỹ thuật thăm dò?

➔ Dễ lập trình, không cần phần cứng hỗ trợ ngắt.

- Ưu điểm của kỹ thuật xử lý ngắt?

➔ Hiệu quả, tiết kiệm tài nguyên CPU vì CPU chỉ xử lý khi có sự kiện.

- Điểm khác nhau giữa ngắt, ngoại lệ và traps?

➔ Ngắt: do thiết bị bên ngoài yêu cầu.

Ngoại lệ: do lỗi hoặc điều kiện đặc biệt bên trong CPU.

Trap: ngoại lệ có chủ ý, thường do chương trình yêu cầu dịch vụ hệ điều hành.