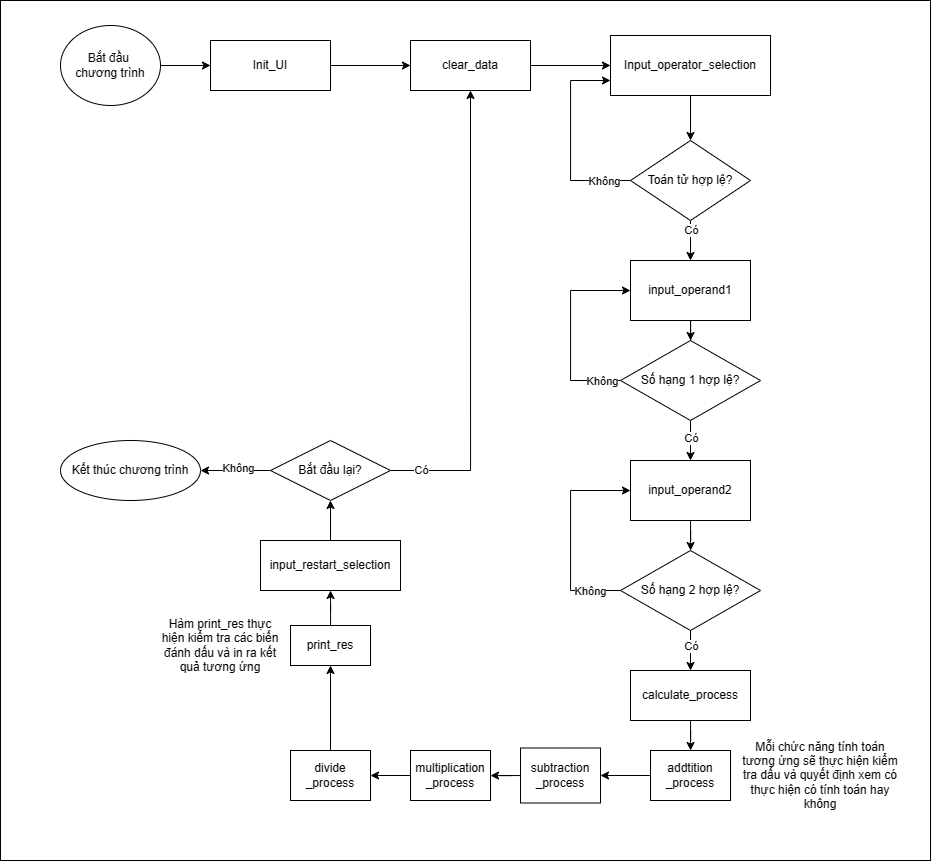
1. **Giới thiệu đề tài.**

* Đề tài: Lập trình Mini Calculator trên Emu 8086.
* Bao gồm: Khởi tạo giao diện chính, thông báo kết quả.

1. **Nội dung chính của đề tài.**

* Nắm rõ cú pháp của chương trình hợp ngữ.
* Sử dụng các biến để lưu các toán tử, toán hạng, sử dụng các hàm tính toán có sẵn của emu8086 để thực hiện tính toán, thực hiện tính toán với kết quả tối đa 16 bit.
* Tìm hiểu và sử dụng các hàm ngắt khác để sử dụng con trỏ, xóa màn hình.
* Sử dụng cách thức ghi đè khoảng trắng lên màn hình để không phải xóa màn hình và in lại nhiều lần.

1. **Flow chart của chương trình**

\* **Quá trình hoạt động của chương trình:**

* 1. Bắt đầu chương trình vào hàm Main.
  2. Tại hàm Main, gọi đến hàm init\_UI để bắt đầu khởi tạo giao diện của chương trình.
  3. Tiếp tục gọi đến hàm clear\_data để reset các biến nhớ của chương trình, tại đây:

Các biến đánh dấu ((is\_divided\_by\_0, is\_res\_over\_flow, is\_restart\_selected, is\_res\_signed) về giá trị mặc định) được đưa về giá trị mặc định bằng 0.

Lần lượt gọi các hàm bắt đầu với **clear\_\*** để thực hiện xóa các nhãn đã hiển thị trên màn hình.

* 1. Tiếp tục gọi đến hàm input\_operator\_selection, tại đây

Chương trình sẽ thực hiện thông báo yêu cầu người dùng nhập phép tính (sử dụng print\_operator\_selection\_noti).

Gọi đến hàm **clear\_input\_label**  để thực hiện xóa nhãn input hiển thị trên màn hình.

Người dùng thực hiện nhập các số từ 1-4 tương ứng với các phép tính hiển thị trên màn hình.

* + - * Nếu input không hợp lệ, gọi đến hàm **print\_wrong\_format\_noti** để in thông báo lỗi, sau đó tiến hành nhập lại.
      * Nếu input hợp lệ, gọi **print\_operator** để thực hiện hiển thị toán tử tương ứng với phép tính đã chọn trên màn hình, tiếp tục thực hiện chương trình.
  1. Tiếp tục gọi đến hàm input\_operand1 để thực hiện nhập toán hạng 1, tại đây

Gọi đến hàm **print\_operand1\_input\_noti** để in ra thông báo yêu cầu người dùng nhập số hạng thứ nhất.

Gọi đến hàm **input\_number** để thực hiện việc nhập số.

Tại hàm **input\_number**, người dùng thực hiện nhập số, kết thúc bởi dấu cách.

* + - * Gọi hàm **clear\_input\_label**, để xóa các giá trị tại nhãn input trên màn hình.
      * Nếu người dùng nhập dấu cách ngay lập tức, chương trình sẽ mặc định người dùng nhập số 0.
      * Nếu người dùng nhập các kí tự khác dấu cách hoặc các chữ số, gọi hàm **print\_wrong\_format\_noti** để in ra thông báo nhập sai định dạng, sau đó thực hiện nhập lại từ đầu.
      * Nếu người dùng nhập số có giá trị vượt quá 65535 (tối đa của 16 bit), gọi hàm **print\_overflow\_num\_noti** để in ra thông báo nhập quá giới hạn cho phép, sau đó thực hiện nhập lại từ đầu.
      * Nếu người nhập số hợp lê, tiếp tục chương trình.

Gọi hàm **print\_operand1** để hiện thị số hạng thứ nhất đã thực hiện trên màn hình.

* 1. Tiếp tục gọi đến hàm input\_operand2 để thực hiện nhập toán hạng 2, luồng hoạt động cũng tương tự input­\_operand1.
  2. Tiếp tục gọi đến hàm calculate\_process để thực hiện tính toán tương ứng với các toán tử và toán hạng đã nhập trước đó.

Gọi lần lượt các hàm addition\_process, multiplication\_process, subtraction\_process, divide\_process.

* + - * Luồng hoạt động chung của các hàm tính toán: thực hiện kiểm tra dấu tương ứng với các hàm trên, nếu thỏa mãn mới thực hiện tính toán, nếu không sẽ nhảy đến cuối hàm và kết thúc.
      * Tại hàm **addition\_process**, thực hiện tính thực hiện phép cộng ứng với các biến **operand1**, **operand2**. Sau khi thực hiện xong sẽ kiểm tra có vượt quá số 16 bit không, nếu có sẽ đánh dấu biến **is\_res\_over\_flow** = 1 và kết thúc hàm. Ngược lại sẽ lưu kết quả vào biến **res.**
      * Tại hàm **subtraction\_process**, thực hiện kiểm tra xem **operand1** và **operand2**, biến nào lớn hơn. Nếu biến **operand2**  > **operand1**, thực hiện dánh dấu biến **is\_res\_signed** = 1, sau đó thực hiện lấy **operand2 – operand1**. Ngược lại lấy **operand1** – **operand2**. Sau đó lưu kết quả trừ cho biến **res**, khi đó biến **res** lưu giá trị tuyệt đối của phép trừ.
      * Tại hàm **multiplication\_process**, thực hiện nhân hai biến **operand1** và **operand2**. Nếu kết quả vượt quá 16 bit sẽ đánh dấu biến **is\_res\_overflow = 1**. Ngược lại sẽ lưu kết quả vào biến **res**.
      * Tại hàm **divide\_process,**  thực hiện kiểm tra xem **operand2** có bằng 0 hay không. Nếu có, thực hiện gán biến **is\_divided\_by\_0** = 1, kết thúc hàm. Ngược lại nếu không sẽ thực hiện phép chia, lưu thương cho biến **res**, lưu dư cho biến **remainder**.
  1. Tiếp tục gọi đến hàm print\_res để thực hiện hiển thị kết quả trên màn hình.

Thực hiện kiểm tra **is\_res\_overflow** có bằng 1 không, nếu có gọi hàm **print\_over\_flow\_num\_noti** để hiển thị thông báo vượt quá giới hạn, kết thúc hàm. Ngược lại sẽ kiểm tra điều kiện tiếp theo.

Thực hiện kiểm tra **is\_res\_signed** có bằng 1 không, nếu có sẽ thực hiện in dấu âm trước, **s**au đó sẽ thực hiện in biến **res** (hiện tại là giá trị tuyệt đối của phép trừ), sau đó kết thúc hàm.

Thực hiện kiểm tra **is\_divide\_by\_0** có bằng 1 không, nếu có gọi hàm **print\_divided\_by\_0\_noti**, sau đó kết thúc hàm.

Nếu các điều kiện trên đều không thỏa mãn, thực hiện hiển thị kết quả sử dụng hàm **print\_number** (với đầu vào là res)và **print\_remainder**. Riêng với hàm **print\_remainder** sẽ kiểm tra biến **operator** có bằng ‘/’ hay không. Nếu có (tức là kết quả phép chia) mới gọi hàm **print\_number** (với đầu vào là biến **remainder**).

* 1. Tiếp tục gọi đến hàm input\_restart\_selection để xem người dùng có muốn thực hiện tính toán tiếp hay không.

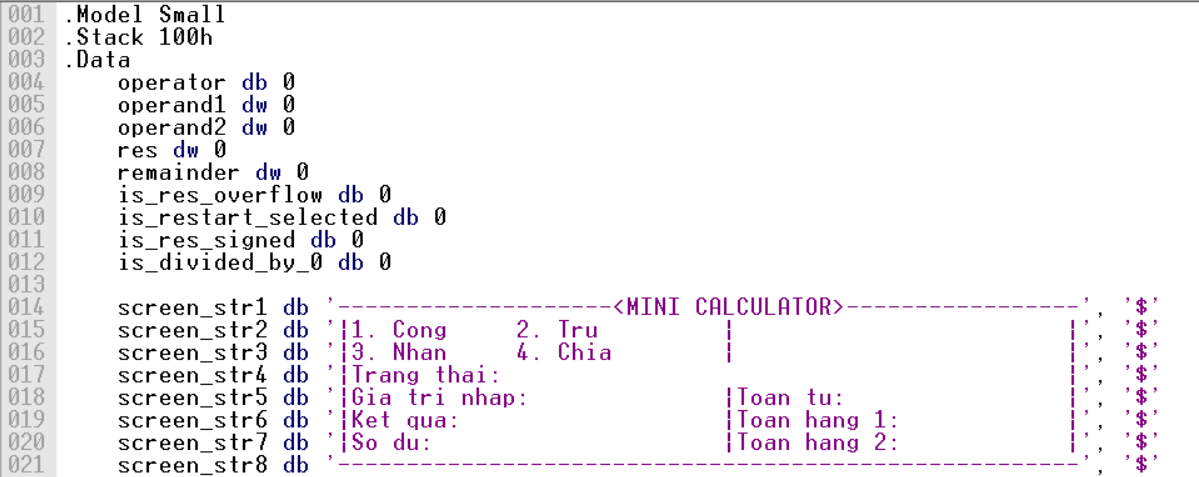
Gọi hàm **print\_restart\_noti** để in ra thông báo hỏi xem người dùng có muốn bắt đầu lại không. Nếu có thì nhập phím 1, nếu không sẽ nhập phím 2.

Nếu người dùng nhập 1, đánh dầu biến **is\_restart\_selected** = 1, kết thúc hàm.

Nếu người dùng nhập 2, đánh dấu biến **is\_restart\_selected** = 0, kết thúc hàm.

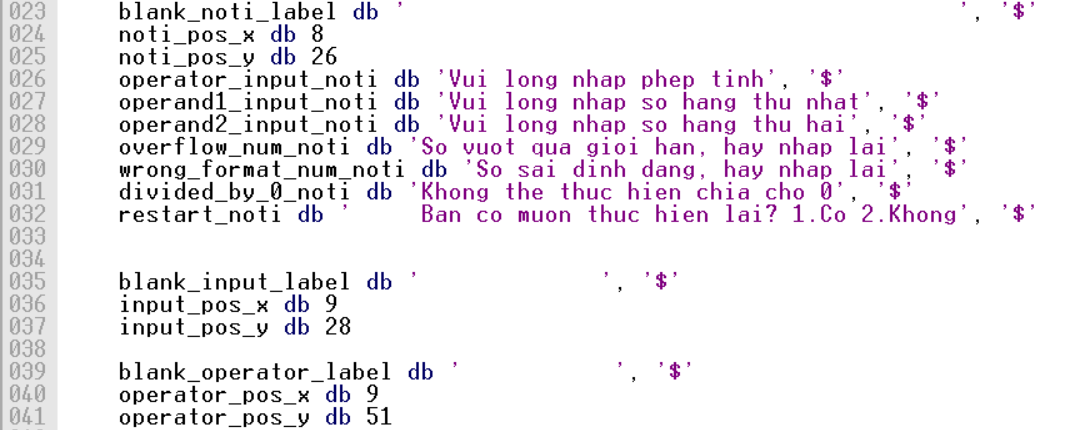
Ngược lại, gọi hàm **print\_wrong\_format\_noti** để thông báo người dùng nhập không hợp lệ, sau đó yêu cầu người dùng nhập lại.

* 1. Tiếp tục thực hiện kiểm tra xem biến is\_restart\_selected có bằng 1 hay không, nếu có sẽ chạy lại chương trình từ hàm clear\_data và tiếp tục thực hiện phép tính mới. Ngược lại sẽ gọi hàm clear\_screen và kết thúc chương trình.

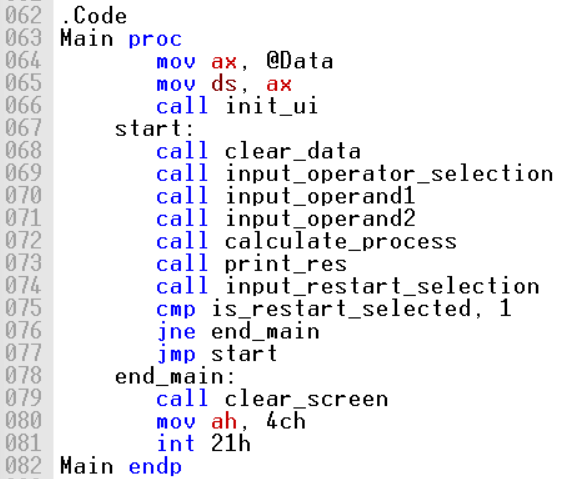
1. **Miêu tả chương trình.**
   1. Data: Khai báo các biến cần sử dụng trong chương trình.

Dòng 4: Khai báo biến operator khởi trị ‘0’, cho biết loại phép tính cần thực hiện ứng với các dấu ‘+’, ‘-‘, ‘\*’, ‘/’.

* + - Dòng 5: Khai báo biến operand1 khởi trị ‘0’, cho biết giá trị toán hạng thứ nhất.
    - Dòng 6: Khai báo biến operand2 khởi trị ‘0’, cho biết giá trị toán hạng thứ hai.
    - Dòng 7: Khai báo biến res khởi trị ‘0’, để lưu kết quả phép tính.
    - Dòng 8: Khai báo biến remainder khởi trị ‘0’, để lưu số dư khi thực hiện phép chia.
    - Dòng 9: Khai báo biến is\_res\_overflow với khởi trị ‘0’ để đánh dấu kiểm tra xem kết quả có bị vươt quá giới hạn hay không, ‘1’ là có, ‘0’ là không.
    - Dòng 10: Khai báo biến is\_restart\_selected với khởi trị ‘0’ để đánh dấu kiểm tra xem người dùng có muốn thực hiện tính toán tiếp không, ‘1’ là có, ‘0’ là không.
    - Dòng 11: Khai báo biến is\_res\_signed với khởi trị ‘0’ để đánh dấu xem kết quả có phải số âm hay không, ‘1’ là có, ‘0’ là không.
    - Dòng 14 – 21: Khai báo các xâu để hiển thị giao diện của chương trình.



* + - Dòng 23: Khai báo biến blank\_noti\_label với giá trị là các khoảng trắng được sử dụng để ghi đè lên khi thực hiện chức năng xóa thông báo.
    - Dòng 24: Khai báo biến noti\_pos\_x với khởi trị là ‘8’ cho biết tọa độ x của nhãn thông báo trên màn hình.
    - Dòng 25: Khai biến noti\_pos\_y với khởi trị là ‘26’ cho biết tọa độ y của nhãn thông báo trên màn hình.
    - Dòng 26: Khai báo xâu kí tự thông báo khi yêu cầu nhập phép tính muốn thực hiện.
    - Dòng 27: Khai báo xâu kí tự thông báo yêu cầu nhập nhập số hạng thứ nhất.
    - Dòng 28: Khai báo xâu kí tự thông báo yêu cầu nhập số hạng thứ hai.
    - Dòng 29: Khai báo xâu kí tự thông báo số vượt quá giới hạn.
    - Dòng 30: Khai báo xâu kí tự số nhập vào sai định dạng cho phép.
    - Dòng 31: Khai báo xâu kí tự thông báo khi thực hiện phép chia cho 0.
    - Dòng 32: Khai bảo xâu kí tự hỏi người sử dụng có muốn tiếp tục thực hiện tính toán hay không
    - Dòng 35: Khai báo xâu kí tự toàn khoảng trống được sử dụng để ghi đè khi xóa nhãn nhập trên màn hình.
    - Dòng 36: Khai báo biến input \_pos\_x với khởi trị ‘9’ cho biết tọa độ x của nhãn nhập trên màn hình.
    - Dòng 37: Khai báo biến input\_pos\_y với khởi trị ‘28’ cho biết tọa độ y của nhãn nhập trên màn hình.
    - Dòng 39: Khai báo xâu kí tự toàn khoảng trống được sử dụng để ghi đè khi xóa nhãn chứa toán tử trên màn hình.
    - Dòng 40: Khai báo biến operator\_pos\_x với khởi trị ‘9’ cho biết tọa độ x của nhãn hiển thị toán tử trên màn hình.
    - Dòng 41: Khai báo biến operator\_pos\_y với khởi trị ‘51’ cho biết tọa độ y của nhãn hiển thị toán tử trên màn hình.
    - Dòng 43: Khai báo xâu toàn khoảng trống để ghi đè khi thực hiện xóa các nhãn hiển thị toán hạng trên màn hình.
    - Dòng 44: Khai báo biến operand1\_pos\_x khởi trị ‘10’ cho biết tọa độ x của nhãn hiển thị toán hạng 1 trên màn hình.
    - Dòng 45: Khai báo biến operand1\_pos\_y khởi trị ‘55’ cho biết tọa độ y của nhãn hiển thị toán hạng 1 trên màn hình.
    - Dòng 47: Khai báo biến operand2\_pos\_x khởi trị ‘11’ cho biết tọa độ x của nhãn hiển thị toán hạng 2 trên màn hình.
    - Dòng 48: Khai báo biến operand2\_pos\_y khởi trị ‘55’ cho biết tọa độ y của nhãn hiển thị toán hạng 2 trên màn hình.
    - Dòng 50: Khai báo xâu toàn khoảng trống để ghi đè khi thực hiện xóa các nhãn hiển thị kết quả trên màn hình.
    - Dòng 51: Khai báo biến res\_pos\_x khởi trị ‘10’ cho biết tọa độ x của nhãn hiển thị kết quả trên màn hình.
    - Dòng 52: Khai báo biến res\_pos\_y khởi trị ‘23’ cho biết tọa độ y của nhãn hiển thị kết quả trên màn hình.
    - Dòng 54: Khai báo xâu toàn khoảng trống để ghi đè khi thực hiện xóa các nhãn hiển thị số dư trên màn hình.
    - Dòng 55: Khai báo biến remainder\_pos\_x khởi trị ‘11’ cho biết tọa độ x của nhãn hiển thị số dư trên màn hình.
    - Dòng 56: Khai báo biến remainder\_pos\_y khởi trị ‘21’ cho biết tọa độ y của nhãn hiển thị số dư trên màn hình.
    - Dòng 58: Khai báo biến xâu toán khoảng trống để ghi đè khi thực hiện xóa các nhãn hiển thị thông báo hỏi có muốn tiếp tục sử dụng chương trình.
    - Dòng 59: Khai báo biến restart\_noti\_pos\_x khởi trị ‘13’ cho biết tọa độ x của nhãn hiển thị thông báo hỏi tiếp tục chương trình không.
    - Dòng 59: Khai báo biến restart\_noti\_pos\_y khởi trị ‘13’ cho biết tọa độ y của nhãn hiển thị thông báo hỏi tiếp tục chương trình không.
  1. Chương trình main: chạy chương trình chính.

Dòng 64: Nạp địa chỉ đoạn dữ liệu cho thanh ghi AX.

Dòng 65:Nạp địa chỉ đoạn dữ liệu bằng thanh ghi AX.

Dòng 66: Gọi hàm **init\_ui** để thực hiện khởi tạo giao diện của chương trình.

Dòng 68: Gọi hàm **clear\_data** để xóa các nhãn hiển thị trên màn hình và nạp giá trị mặc định cho các toán hạng, toán tử và biến đánh dấu.

Dòng 69: Gọi hàm **input\_operator\_selection** để thực hiện nhập phép tính muốn thực hiện.

Dòng 70: Gọi hàm **input\_operand1** để yêu cầu nhập giá trị cho toán hạng thứ nhất.

Dòng 71: Gọi hàm **input\_operand2** để yêu cầu nhập giá trị cho toán hạng thứ hai.

Dòng 72: Gọi hàm **calculate\_process** để thực hiện quá trình tính toán sau khi nhập.

Dòng 73: Gọi hàm **print\_res** để thực hiện in ra kết quả hoặc thông báo kèm theo.

Dòng 74: Gọi hàm **input\_restart\_selection** để hỏi người dùng có muốn tiếp tục thực hiện tính toán hay không, sau đó đánh dấu lại lựa chọn.

Dòng 75: So sánh biến is\_restart\_selectedvới ‘1’.

Dòng 76: Nếu biến is\_restart\_selected khác 1 thì nhảy về nhãn end\_main để kết thúc chương trình.

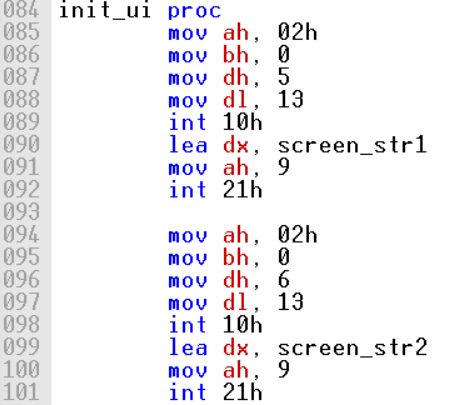
Dòng 77: Biến is\_restart\_selected bằng ‘1’, nhảy về nhãn start để bắt đầu lại chương trình.

Dòng 79: Gọi hàm **clear\_screen** để thực hiện xóa màn hình.

Dòng 80: Nạp giá trị 4CH vào thanh ghi AH.

Dòng 81: Thực hiện hàm 4CH của ngắt INT 21h để kết thúc chương trình.

* 1. Hàm con init\_ui: sử dụng để khởi tạo giao diện chính của chương trình.

**Tham khảo:** hàm 2 của ngắt 10h để di chuyển con trỏ về vị trí xác định, trong đó BH lưu vị trí trang của con trỏ, DH lưu vị trí hàng của con trỏ, DL lưu vị trí cột của con trỏ.

Dòng 85: Nạp giá trị ‘02h’ cho AH.

Dòng 86: Nạp giá trị ‘0’ cho BH.

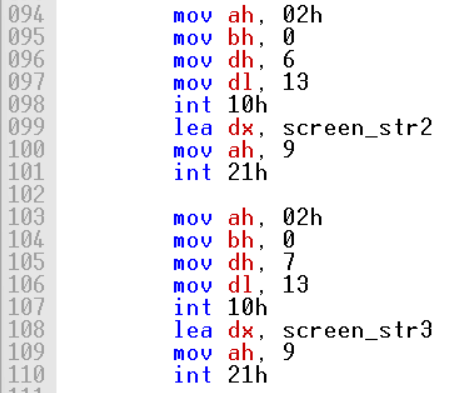
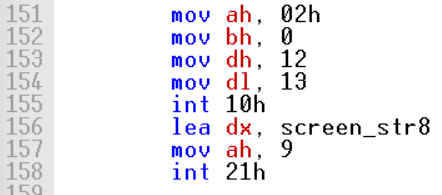
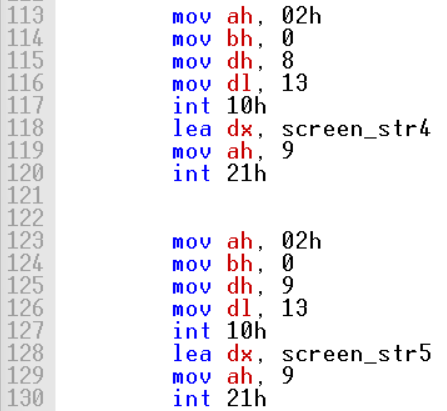
Dòng 87: Nạp giá trị ‘5’ cho DH.

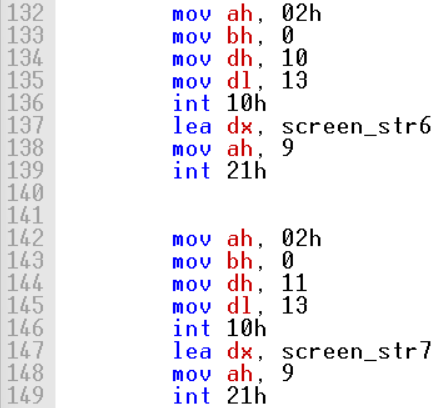
Dòng 88: Nạp giá trị ‘13’ cho DL.

Dòng 89: Thực hiện hàm 2 ngắt 10H để di chuyển vị trí con trỏ.

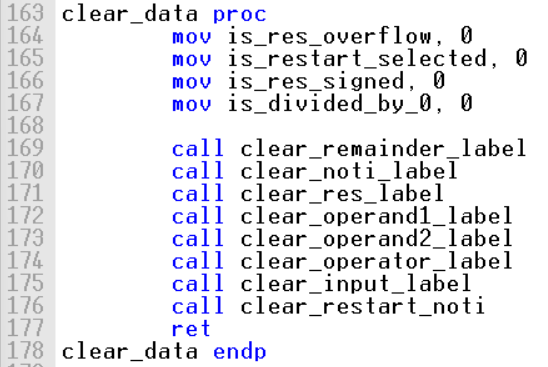
Dòng 90: Nạp giá trị địa chỉ của biến screen\_str1 cho DX.

Dòng 91: Nạp giá trị ‘9’ cho AH.

Dòng 92: Thực hiện hàm 9 ngắt 21h để in ra xâu screen\_str1.

Tương tự với nhóm các dòng 94-101, 103-110, 113-120, 123-130, 132-139, 142-149, 151-158 cũng thực hiện chức năng tương tự lần lượt với xâu screen\_str 2-8

* 1. Hàm con clear\_data: được sử dụng để khởi tạo lại các biến đánh dấu, xóa các nhãn hiển thị trên màn hình.

Dòng 164: Nạp giá trị ‘0’ cho biến is\_res\_overflow.

Dòng 165: Nạp giá trị ‘0’ cho biến is\_restart\_selected.

Dòng 166: Nạp giá trị ‘0’ cho biến is\_restart\_selected.

Dòng 167: Nạp giá trị ‘0’ cho biến is\_divided\_by\_0.

Dòng 169: Gọi hàm **clear\_remainder\_label** để xóa nhãn hiển thị số dư trên màn hình.

Dòng 170: Gọi hàm **clear\_noti\_label** để xóa nhãn hiển thị thông báo trên màn hình.

Dòng 171: Gọi hàm **clear\_res\_label** để xóa nhãn hiển thị kết quả trên màn hình.

Dòng 172: Gọi hàm **clear\_operand1\_label** để xóa nhãn hiển thị số hạng thứ nhất trên màn hình.

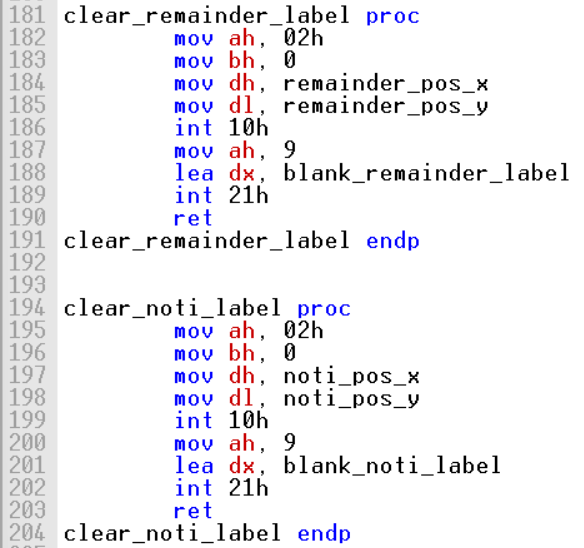
Dòng 173: Gọi hàm **clear\_operand2\_label** để xóa nhãn hiển thị số hạng thứ hai trên màn hình.

Dòng 174: Gọi hàm **clear\_operator\_label** để xóa nhãn hiển thị toán tử trên màn hình.

Dòng 175: Gọi hàm **clear\_input\_label** để xóa nhãn hiển thị số đã nhập trên màn hình.

Dòng 176: Gọi hàm **clear\_restart\_noti** để xóa nhãn hiển thị thông báo có thực hiện tiếp chương trình không.

Dòng 177: Quay trở lại chương trình đã gọi.

* 1. Các hàm clear\_xxx\_label: sử dụng để xóa các nhãn hiển thị trên màn hình.

**Tham khảo:** Sử dụng hàm 2 của ngắt 10h để thực hiện di chuyển vị trí con trỏ.

Dòng 182: Nạp giá trị ‘2’ cho AH.

Dòng 183: Nạp giá trị ‘0’ cho BH.

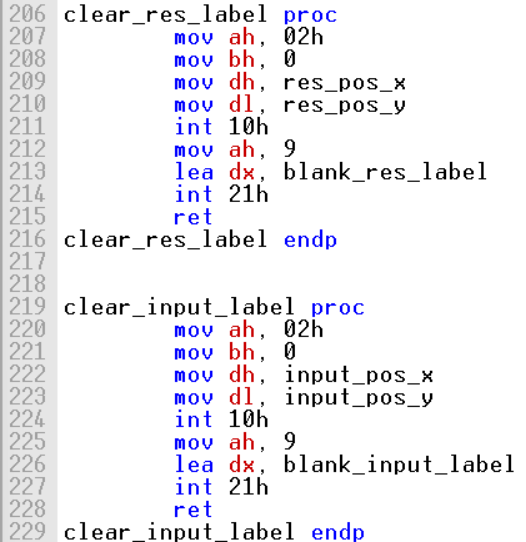
Dòng 184: Nạp giá trị biến remainder\_pos\_x (cho biết tọa độ x của nhãn hiển thị số dư) vào DH.

Dòng 185: Nạp giá trị biến remainder\_pos\_y (cho biết tọa độ ý của nhãn hiển thị số dư) vào DL.

Dòng 186: Thực hiện ngắt 10h, chuyển vị trí con tró đến vị trị remainder\_pos\_x, remainder\_pos\_y.

Dòng 187: Nạp giá trị ‘9’ cho AH.

Dòng 188: Nạp địa chỉ biến blank\_remainder\_label cho DX.

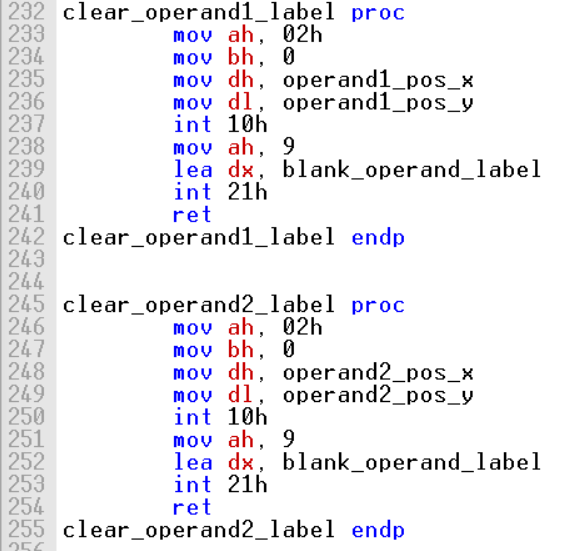
Dòng 189: Thực hiện hàm 9 của ngắt INT 21H để in ra xâu blank\_remainder\_label toàn khoảng trống để ghi đè, xóa đi nhãn hiển thị số dư trước đó.

Dòng 190: Quay lại chương trình đã gọi.

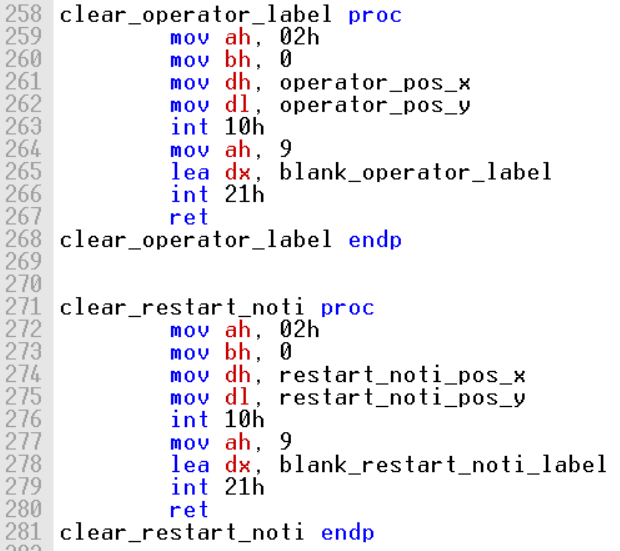
Dòng 195-203: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn thông báo để xóa nhãn hiển thị thông báo trước đó.

Dòng 207-215: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn kết quả để xóa nhãn hiển thị kết quả trước đó.

Dòng 220-228: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn hiển thị số đã nhập để xóa nhãn hiển thị số đã nhập trước đó.

Dòng 233-241: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn hiển thị số hạng thứ nhất để xóa nhãn hiển thị số đã nhập trước đó.

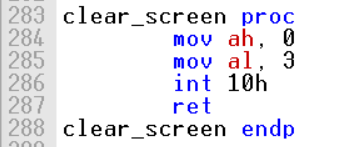
Dòng 233-241: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn hiển thị số hạng thứ nhất để xóa nhãn hiển thị số hạng thứ nhất trước đó.

Dòng 246-254: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn hiển thị số hạng thứ hai để xóa nhãn hiển thị số hạng thứ hai trước đó.

Dòng 259-267: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn hiển thị toán tử để xóa nhãn hiển thị toán tử trước đó.

Dòng 272-280: Thực hiện nhiệm vụ tương tự, in đè khoảng trắng tại vị trí hiển thị nhãn hiển thị thông báo khởi động lại chương trình để xóa nhãn hiển thị trước đó.

* 1. Hàm con clear\_screen để thực hiện xóa toàn màn hình.

**Tham khảo:** Hàm 0 của ngắt 10h để xóa màn hình trong đó AL cho biết chế độ video muốn xóa.

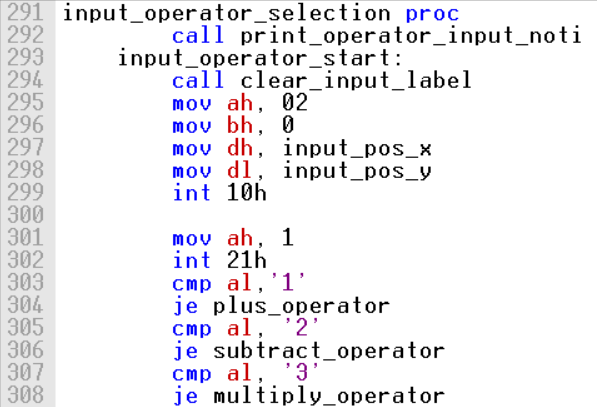
Dòng 284: Nạp ‘0’ cho AH.

Dòng 285: Nạp ‘3’ cho AL. Chế độ video: văn bản, kích thước 80x25, 16 màu, 8 trang.

Dòng 286: Thực hiện hàm 0 của ngắt INT 10H.

Dòng 287: Quay lại chương trình đã gọi.

* 1. Hàm con input\_operator để thực hiện nhập phép tính muốn thực hiện.

Dòng 292: Gọi hàm **print\_operator\_input\_noti** để thực hiện in ra thông báo yêu cầu chọn phép tính muốn thực hiện.

Dòng 294: Gọi hàm **clear\_input\_label** để xóa nhãn hiễn thị các input trước đó.

Dòng 295: Nạp giá trị ‘2’ cho AH.

Dòng 296: Nạp giá trị ‘0’ cho BH.

Dòng 297: Nạp giá trị biến input\_pos\_x cho DH.

Dòng 298: Nạp giá trị biến input\_pos\_y cho DL.

Dòng 299: Thực hiện hàm 2 của ngắt INT 10H để di chuyển vị trí con trỏ về vị trí input\_pos\_x, input\_pos\_y.

Dòng 301: Nạp giá trị ‘1’ vào AH.

Dòng 302: Thực hiện hàm ngắt 1 của INT 21H để nhập kí tự với đầu ra là thanh ghi AL.

Dòng 303: So sánh AL với kí tự ‘1’.

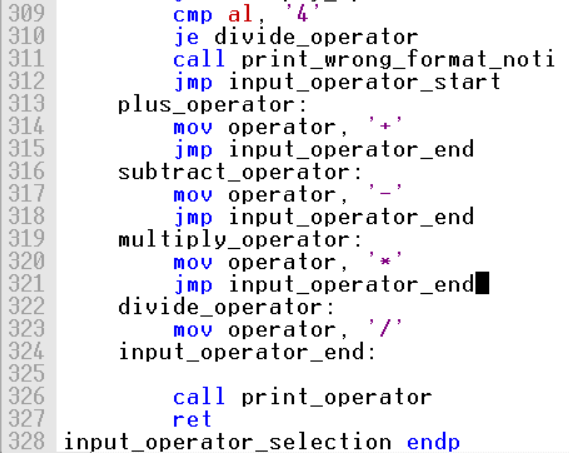
Dòng 304: Nếu AL bằng kí tự ‘1’ thì nhảy đến nhãn plus\_operator.

Dòng 305: So sánh AL với kí tự ‘2’.

Dòng 306: Nếu AL bằng kí tự ‘2’ thì hảy đến nhãn subtract\_operator.

Dòng 307 So sánh AL với kí tự ‘3’.

Dòng 308: Nếu AL bằng kí tự ‘3’ thì nhảy đến nhãn multiply\_operator.

Dòng 309: So sánh AL với kí tự ‘4’.

Dòng 310: Nếu AL bằng kí tự ‘4’ thì nhảy đến nhãn divide\_operator.

Dòng 311: Gọi hàm **print\_wrong\_format\_noti** để thực hiện in ra thông báo nhập sai định dạng.

Dòng 312: Nhảy về nhãn input\_operator\_start để thực hiện nhập lại phép tính muốn thực hiện.

Dòng 314: Nạp giá trị ‘+’ cho biến operator.

Dòng 315: Nhảy đến nhãn input\_operator\_end để kết thúc nhập.

Dòng 317: Nạp giá trị ‘-’ cho biến operator.

Dòng 318: Nhảy đến nhãn input\_operator\_end để kết thúc nhập.

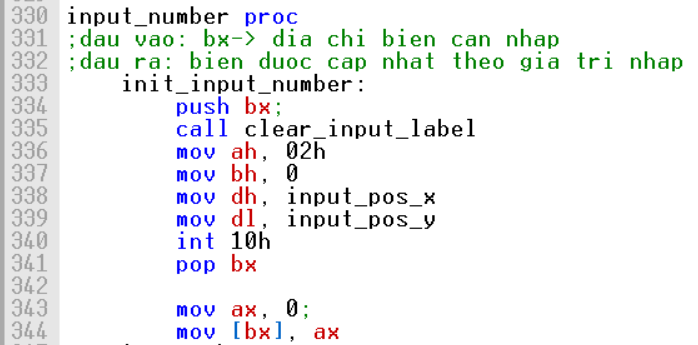
Dòng 320: Nạp giá trị ‘\*’ cho biến operator.

Dòng 321: Nhảy đến nhãn input\_operator\_end để kết thúc nhập.

Dòng 323: Nạp giá trị ‘/’ cho biến operator.

Dòng 326: Gọi hàm **print\_operator** để thực hiện hiển thị toán tử của phép tính đã chọn trên màn hình.

Dòng 327: Quay lại chương trình đã gọi.

* 1. Hàm con input\_number dùng để nhập một số vào một biến.

Dòng 334: lưu tạm biến BX vào stack do các câu lệnh sau có ảnh hưởng đến thanh ghi BX.

Dòng 335: Gọi hàm clear\_input\_label để xóa nhãn input hiển thị trên màn hình.

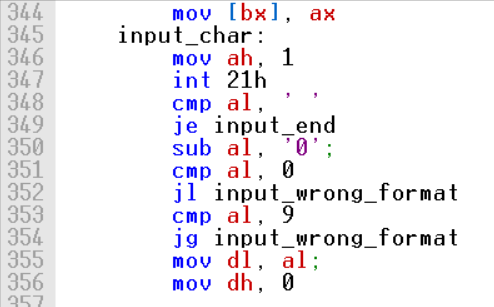
Dòng 336: Nạp giá trị ‘2’ vào thanh ghi AH.

Dồng 337: Nạp giá trị ‘0’ vào thanh ghi BH.

Dòng 338: Nạp giá trị biến input\_pos\_x vào DH.

Dòng 339: Nạp giá trị biến input\_pos\_y vào DL.

Dòng 340: Thực hiện hàm 2 của n gắt INT 10h với đầu vào BH = 0, DH = input\_pos\_x, input\_pos\_y để di chuyển vị trí con trỏ đến nhãn input trên màn hình.

Dòng 341: Lấy lại giá trị vừa BX vừa lưu tạm vào stack ở dòng 334, sau đó gán lại giá trị này cho BX.

Dòng 343: Nạp giá trị ‘0’ vào AX.

Dòng 344: Nạp giá trị AX vào biến nhớ có địa chỉ DS:BX, phải thực hiện nạp ‘0’ thông qua AX do không thể gán trực tiếp giá trị 16 bit vào ô nhớ có địa chỉ DS:BX.

Dòng 346: Nạp giá trị ‘1’ cho AH.

Dòng 347: Thực hiện hàm 1 của ngắt INT 21H để nhập kí tự vào, được đầu ra AL là kí tự vừa nhập.

Dòng 348: So sánh AL với kí tự ‘ ‘ (kí tự đánh dấu kết thúc nhập).

Dòng 349: Thực hiện nhảy đến nhãn input\_end nếu AL = ‘ ‘, ngược lại tiếp tục chương trình.

Dòng 350: Thực hiện tính AL – kí tự ‘0’ và nạp vào AL để chuyển từ mã ascii sang giá trị số.

Dòng 351: So sánh AL với giá trị 0.

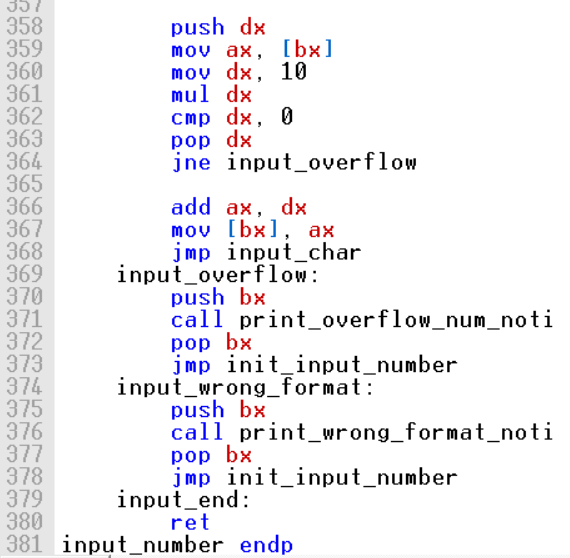
Dòng 352: Thực hiện nhảy đến nhãn input\_wrong\_format nếu AL < 0 (tức giá trị nhập vào không phải kí tự số), ngược lại tiếp tục thực hiện chương trình.

Dòng 353: So sánh AL với giá trị 9.

Dòng 354: Thực hiện nhảy đến nhãn input\_wrong\_format nếu AL > 9 (tức giá trị nhập vào không phải kí tự số), ngược lại tiếp tục thực hiện chương trình.

Dòng 355: Nạp giá trị AL vào DL. (tức lưu chữ số vừa nhập vào DL)

Dòng 356: Nạp giá trị 0 vào DH.

Dòng 358: Lưu tạm giá trị DX (h iện tại có chứa thông tin về giá trị chữ số vừa nhập tại dòng 355) vào trong stack (Do câu lệnh 361 có sử dụng hàm MUL có ảnh hướng đến giá trị DL nên cần lưu tạm vào stack, tránh mất dữ liệu).

Dòng 359: Lưu giá trị tại biến có địa chỉ DS:BX vào AX

Dòng 360: Nạp giá trị ‘10’ vào trong DX

Dòng 361: Thực hiện hàm nhân với toán hạng nguồn là DX, khi đó kết quả AX \* DX sẽ được lưu vào DX:AX (Trong đó DX = 10, AX = giá trị tại biến cần lưu kết quả nhập).

Dòng 362: So sánh DX, 0.

Dòng 363: thực hiện lấy giá trị của đầu của stack chính là giá trị của DX lưu tạm tại dòng 357 (giá trị đó chính là giá trị của chữ số nhập từ bàn phím).

Dòng 364: Do dòng 363 không ảnh hưởng đến cờ so sánh nên giá trị so sánh tại dòng 362 vẫn sử dụng được. Nếu giá trị DX != 0 tức là phép nhân đã phải sử dụng thêm thanh ghi DX để lưu kết quả nên kết quả đã vượt quá 16 bit, khi đó nhảy đến nhãn input\_overflow.

Dòng 366: Trong trường hợp kết quả không vượt quá 16 bit, thực hiện cộng DX và AX, sau đó lưu vào AX.

Dòng 367: Nạp AX vào ô nhớ có địa chỉ DS:BX tức tham số được truyền trước khi gọi hàm.

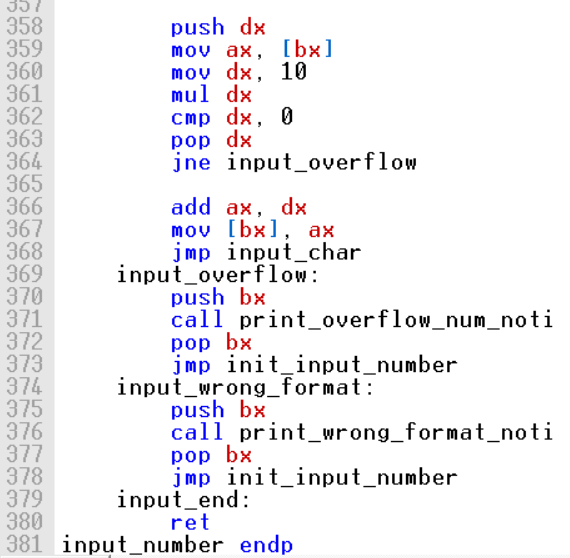
Dòng 368: Nhảy đến nhãn input\_char tiếp tục thực hiện nhập số.

Dòng 370: Đẩy tạm giá trị BX vào stack do lệnh tại dòng 371 có làm thay đổi thanh ghi BX.

Dòng 371: Gọi hàm **print\_overflow\_num\_noti** để thông báo số đã nhập vượt quá giới hạn cho phép.

Dòng 372: Lấy lại giá trị từ đầu stack (giá trị đó là giá trị ta vừa đẩy tạm tại dòng 370), sau đó nạp vào thanh ghi BX.

Dòng 373: Nhảy đến nhãn init\_input\_number để thực hiện nhập lại từ đầu.

Dòng 375: Lưu tạm biến BX vào stack do dòng 376 làm thay đổi giá trị BX.

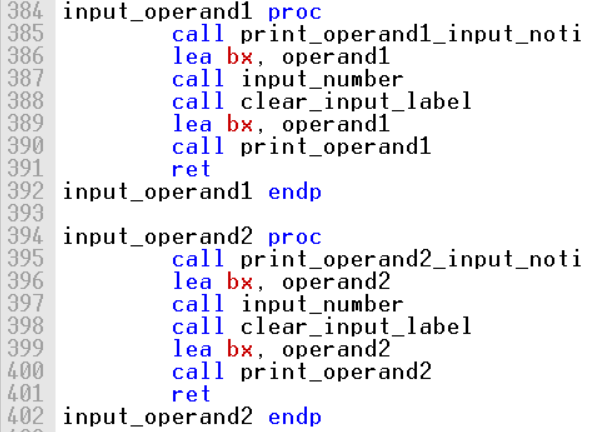
Dòng 376: Gọi hàm **print\_wrong\_format\_noti** để hiển thị thông báo lỗi nhập sai định dạng.

Dòng 377: Lấy lại giá trị vừa lưu vào stack tại dòng 375, sau đó nạp vào thanh ghi BX.

Dòng 378: Nhảy đến nhãn init\_input\_number để thực hiện nhập lại tự đầu.

Dòng 380: Quay lại chương trình đã gọi hàm con.

* 1. Hàm con input\_operand1 và input\_operand2 để thực hiện nhập số hạng thứ nhất và số hạng thứ 2.

Dòng 385: Gọi hàm **print\_operand1\_input\_noti** để thông báo yêu cầu nhập số hạng thứ nhất.

Dòng 386: Nạp địa chỉ của biến operand1 vào BX, BX là đầu vào của hàm con **input\_number**.

Dòng 387: Gọi hàm **input\_number** để thực hiện nhập số hạng thứ nhất.

Dòng 388: Gọi hàm **clear\_input\_label** để xóa nhãn hiển thị số đã nhập trên màn hình.

Dòng 389: Gọi hàm **print\_operand1** để hiển thị số đã nhập tại nhãn hiển thị số hạng thứ nhất.

Dòng 390: Quay trở lại chương trình đã gọi hàm này.

Dòng 394: Gọi hàm **print\_operand2\_input\_noti** để hiển thị thông báo yêu cầu nhập số hạng thứ 2.

Dòng 395: Nạp giá trị địa chỉ của operand2 vào BX (đầu vào của hàm **print\_number)**.

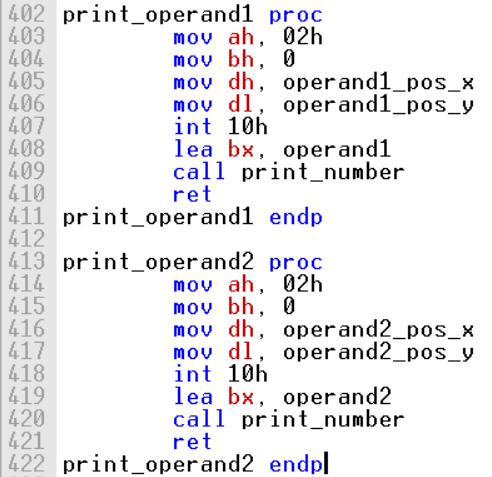
Dòng 396: Gọi hàm **input\_number** để nhập số hạng thứ 2.

Dòng 397: Gọi hàm **clear\_input\_label** để xóa nhãn hiển thị số đã nhập.

Dòng 398: Gọi hàm **print\_operand2** để in ra nhãn số hạng 2 trên màn hình.

Dòng 399: Quay trở lại chương trình đã gọi hàm này.

* 1. Hàm print\_operand1 và print\_operand2 để in nhãn hiển thị số hạng thứ nhất và nhãn hiển thị số hạng thứ 2 ra màn hình.

Dòng 403: Nạp giá trị ‘2’ vào AH.

Dòng 404: Nạp giá trị ‘0’ vào BH.

Dòng 405: Nạp giá trị biến operand1\_pos\_x vào DH.

Dòng 406: Nạp giá trị biến operand1\_pos\_y vào DL.

Dòng 407: Gọi hàm 2 của ngắt INT 10h để di chuyển con trỏ về vị trí operand1\_pos\_x, operand1\_pos\_y.

Dòng 408: nạp giá trị địa chỉ biến operand1 vào BX là tham số cho hàm **print\_number**.

Dòng 409: Gọi hàm **print\_number**.

Dòng 410: Quay lại chương trình đã gọi nó.

Dòng 410: Quay trở lại chương trình đã gọi.

Dòng 414: Nạp giá trị ‘2’ vào AH.

Dòng 415: Nạp giá trị ‘0’ vào BH.

Dòng 416: Nạp giá trị operand2\_pos\_x vào DH.

Dòng 417: Nạp giá trị operand2\_pos\_y vào DL.

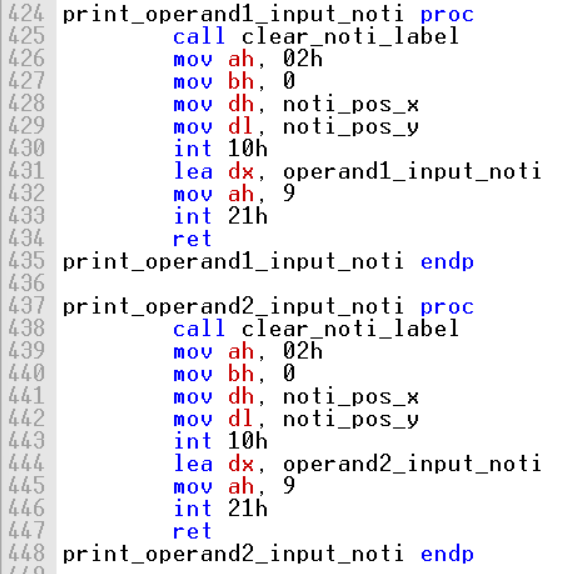
Dòng 418: Thực hiện hàm 2 của ngắt INT 10h để di chuyển con trỏ đến vị trí có tọa độ operand2\_pos\_x và operand2\_pos\_y.

Dòng 419: Nạp địa chỉ biến operand2 vào BX.

Dòng 420: Gọi hàm **print\_number** để in ra biến operand2 tại vị trí con trỏ hiện tại.

Dòng 421: Quay lại chương trình đã gọi nó.

* 1. Các hàm con print\_xxx\_noti được sử dụng để in ra nhãn thông báo trên màn hình.

Dòng 425: Gọi hàm **clear\_noti\_label** để xóa các nhãn hiển thị thông báo trước đó.

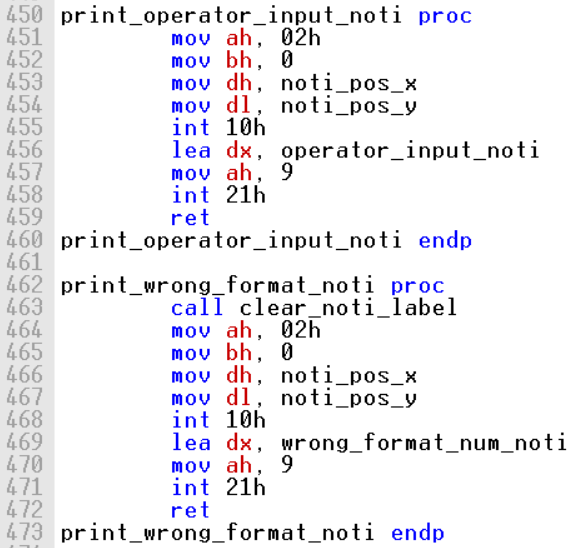
Dòng 426: Nạp giá trị ‘2’ vào AH.

Dòng 427: Nạp giá trị ‘0’ vào BH.

Dòng 428: Nạp giá trị biến noti\_pos\_x vào DH.

Dòng 429: Nạp giá trị biến noti\_pos\_y vào DL.

Dòng 430: Gọi hàm 2 ngắt INT 10H để thực hiện di chuyển con trỏ đến vị trí có tọa độ noti\_pos\_x và noti\_pos\_y.

Dòng 431: Nạp địa chỉ xâu operand1\_input\_noti vào DX.

Dòng 432: Nạp giá trị ‘9’ vào AH.

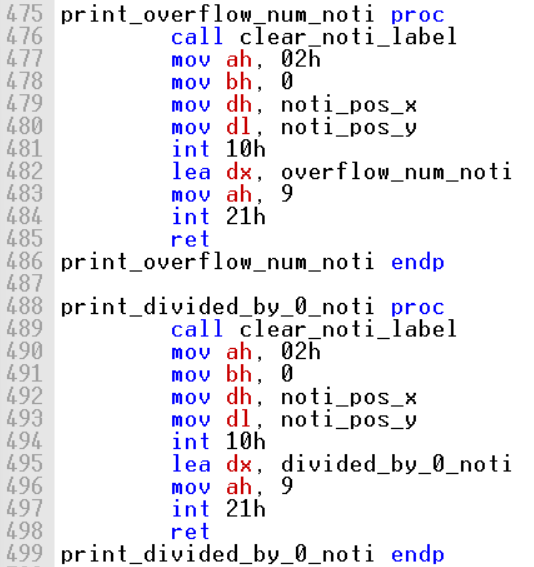
Dòng 433: Gọi hàm 9 ngắt INT 21H để in ra màn hình xâu operand1\_input\_noti yêu cầu người dùng nhập số hạng thứ nhất.

Dòng 434: Quay lại chương trình đã gọi nó.

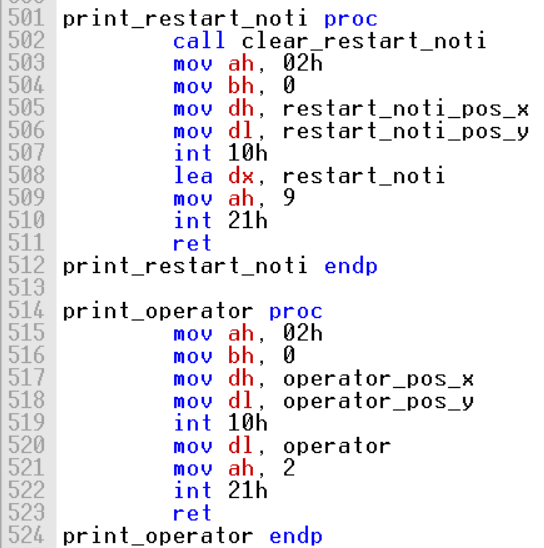
Dòng 438-447: Hoạt động tương tự, thực hiện in ra màn hình xâu operand2\_input\_noti tại vị trí noti\_pos\_x, noti\_pos\_y.

Dòng 451-459: Hoạt động tương tự, thực hiện in ra màn hình xâu operator\_input\_noti tại vị trí noti\_pos\_x, noti\_pos\_y để yêu cầu người dùng nhập toán tử.

Dòng 463-472: Tương tự, in ra màn hình xâu wrong\_format\_num\_noti tại vị trí noti\_pos\_x, noti\_pos\_y để thông báo nhập sai định dạng.

Dòng 476-485: Hoạt động tương tự, thực hiện in ra màn hình xâu over\_flow\_num\_noti tại vị trí noti\_pos\_x, noti\_pos\_y để thông báo số vươt quá giới hạn.

Dòng 489-498: Tương tự, in ra màn hình xâu divided\_by\_0\_noti tại vị trí noti\_pos\_x, noti\_pos\_y để thông báo lỗi sai khi chia cho 0.

Dòng 502-511: Tương tự, in ra màn hình xâu restart\_noti tại vị trí restart\_noti\_pos\_x, restart\_noti\_pos\_y để hỏi người dùng có muốn tiếp tục chương trình không.

Dòng 515: Nạp giá trị ‘2’ vào AH.

Dòng 516: Nạp giá trị ‘0’ vào BH.

Dòng 517: Nạp giá trị biến operator\_pos\_x vào DH.

Dòng 518: Nạp giá trị biến operator\_pos\_y vào DL.

Dòng 519: Gọi hàm 2 của ngắt 10H để chuyển con trỏ đến vị trí operator\_pos\_x, operator\_pos\_y.

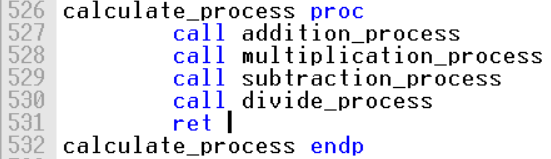
Dòng 520: Nạp giá trị biến operator vào DL.

Dòng 521: Nạp giá trị ‘2’ vào AH.

Dòng 522: Gọi hàm 2 của ngắt INT 21h để in ra màn hình kí tự có mã ascii là DL.

Dòng 523: Quay lại chương trình đã gọi hàm này.

* 1. Hàm con calculate\_process để thực hiện các phép tính cộng, trừ, nhân, chia.
     + Tại mỗi hàm con đối với mỗi phép tính riêng biệt, hàm đó sẽ kiểm tra dấu trước. Nếu thỏa mãn mới thực hiện tính và lưu kết quả hoặc thông báo.

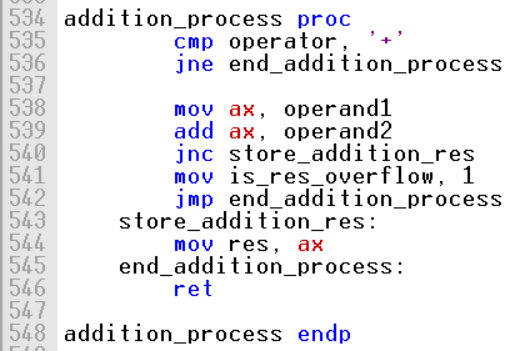
Dòng 527: Gọi hàm **addition\_process** để thực hiện phép tính cộng.

Dòng 528: Gọi hàm **multiplication\_process** để thực hiện phép tính nhân.

Dòng 529: Gọi hàm **subtraction\_process** để thực hiện phép tính trừ.

Dòng 530: Gọi hàm **divide\_process** để thực hiện phép tính chia.

* 1. Hàm con addition\_process để thực hiện phép tính cộng.

Dòng 535: So sánh biến operator với kí tự ‘+’.

Dòng 536: Nếu operator không = kí tự ‘+’, thực hiện nhảy đến nhãn end\_addition\_process và kết thúc hàm con, ngược lại tiếp tục chạy lệnh tiếp theo.

Dòng 538: Nạp giá trị biến operand1 vào AX.

Dòng 539: Thực hiện cộng operand2 với biến AX, phép cộng này ảnh hưởng đến cờ carry. CF = 1 tức là phép cộng có nhớ ra khỏi bit cao nhất. Tức là phép cộng 2 số 16 bit vượt quá 16 bit.

Dòng 540: Nếu cờ carry = 0, thực hiện nhảy đến nhãn store\_addition\_res, ngược lại thực hiện lệnh tiếp theo.

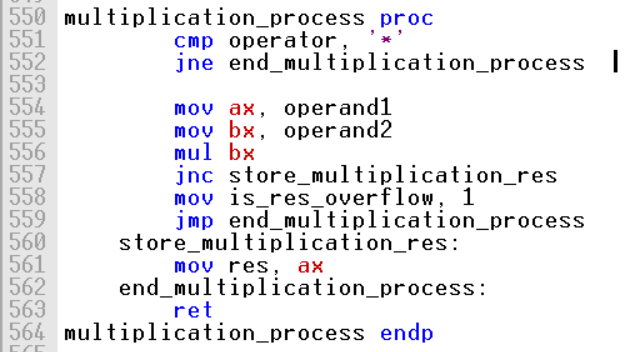
Dòng 541: Nạp giá trị ‘1’ vào biến is\_res\_overflow, đánh dầu kết quả vượt quá giới hạn.

Dòng 542: Nhảy đến nhãn end\_addition\_process.

Dòng 544: Nạp giá trị AX vào biến res.

Dòng 546: Quay lại chương trình chính.

* 1. Hàm con multiplication\_process để thực hiện phép tính nhân.

Dòng 551: So sánh biến operator với kí tự ‘\*’.

Dòng 552: Nếu operator khác ‘\*’ thực hiện nhảy đến nhãn end\_multiplication\_process, kết thúc hàm con, ngược lại thực hiện lệnh tiếp theo.

Dòng 554: Nạp giá trị biến operand1 vào AX.

Dòng 555: Nạp giá trị biến operand2 vào BX.

Dòng 556: Thực hiện phép nhân sử dụng hàm MUL BX. Đây là phép nhân 16 bit. Khi đó chương trình sẽ thực hiện lấy AX \* BX sau đó lưu kết quả vào DXAX. Phép nhân sẽ ảnh hưởng đến cờ carry. Nếu CF = 1 tức là phép nhân có nhớ ra khỏi bit cao nhất hay kết quả phép nhân vượt quá 16 bit.

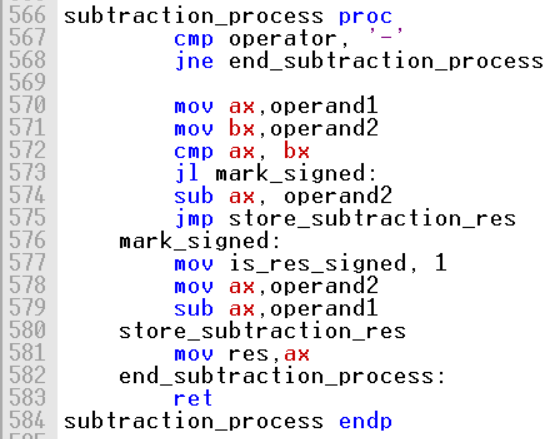
Dòng 557: Nếu CF = 0, thực hiện nhảy đến store\_multiplication\_res để lưu lại kết quả phép nhân. Ngược lại thực hiện lệnh tiếp theo.

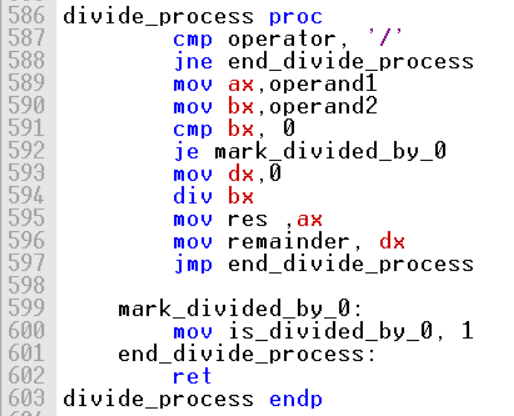
Dòng 558: Nạp giá trị ‘1’ vào biến is\_res\_overflow đánh dấu kết quả vượt quá giới hạn 16 bit.

Dòng 559: Nhảy đến nhãn end\_multiplication\_process để kết thúc hàm con.

Dòng 561: Nạp giá trị AX vào biến res.

Dòng 563: Quay lại chương trình đã gọi nó.

* 1. Hàm con subtraction\_process dùng để thực hiện phép tính trừ.
     + Dòng 567: So sánh biến operator với kí tự ‘-‘
     + Dòng 568: Nếu operator khác ‘-‘ thực hiện nhảy đến end\_subtraction\_process để kết thúc hàm con.
     + Dòng 570: Nạp giá trị operand1 vào AX.
     + Dòng 571: Nạp giá trị operand2 vào BX.
     + Dòng 572: So sánh AX với BX.
     + Dòng 573: Nếu AX < BX nhảy đến nhãn mark\_signed để đánh dấu số âm.
     + Dòng 574: Thực hiện lấy AX – operand2 sau đó nạp kết quả vào AX.
     + Dòng 575: Nhảy đến store\_subtraction\_res để lưu lại kết quả.
     + Dòng 577: Nạp giá trị ‘1’ vào biến is\_res\_signed, đánh dấu kết quả phép trừ là số âm.
     + Dòng 578: Nạp giá trị biến operand2 vào AX.
     + Dòng 579: Thực hiện AX – operand1 sau đó lưu vào thanh ghi AX.
     + Dòng 581: Nạp giá trị AX vào res, res lưu kết quả là trị tuyệt đối của phép trừ.
     + Dòng 583: Quay lại chương trình đã gọi hàm này.
  2. Hàm con divide\_process dùng để thực hiện phép tính chia.

Dòng 587: So sánh operator với kí tự ‘/’.

Dòng 588: Nếu operator khác ‘/’ , nhảy đến hàm end\_divide\_process để kết thúc hàm con. Ngược lại, thực hiện lệnh tiếp theo.

Dòng 589: Nạp giá trị operand1 vào AX.

Dòng 590: Nạp giá trị operand2 vào BX.

Dòng 591: So sánh BX với 0.

Dòng 592: Nếu BX = 0, nhảy đến nhãn mark\_divided\_by\_0.

Dòng 593: Nạp giá trị ‘0’ vào DX. Do khi thực hiện phép chia 16 bit, chương trình sẽ thực hiện lấy DXAX / toán hạng 16 bit => Để chia AX cho BX cần đưa DX về 0 để đảm bảo kết quả thực hiện đúng.

Dòng 594: Thực hiện phép chia DXAX cho BX. Khi đó kết quả phép chia sẽ bao gồm thương được lưu tại AX, số dư lưu tại DX.

Dòng 595: Nạp giá trị AX vào biến res (thương).

Dòng 596: Nạp giá trị DX vào biến remainder (dư).

Dòng 597: Nhảy đến nhãn end\_divide\_process để kết thúc hàm con.

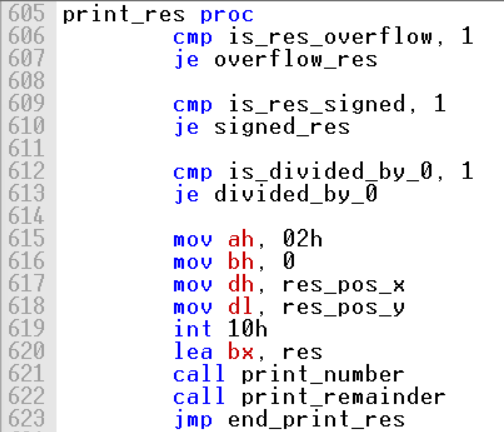
Dòng 600: Nạp giá trị ‘1’ cho biến is\_divided\_by\_0 để đánh dấu đây là phép chia cho 0.

Dòng 602: Quay lại chương trình đã gọi hàm này.

* 1. Chương trình con print\_res được sử dụng để in ra kết quả và các thông báo kèm theo (nếu có).

Dòng 606: So sánh biến is\_res\_overflo w với ‘1’.

Dòng 607: Nếu is\_res\_overflow = 1, thực hiện nhảy đến nhãn overflow\_res, ngược lại thực hiện lệnh tiếp theo.

Dòng 609: So sánh biến is\_res\_signed với ‘1’.

Dòng 610: Nếu is\_res\_signed = 1, thực hiện nhảy đến nhãn signed\_res, ngược lại thực hiện lệnh tiếp theo.

Dòng 612: So sánh is\_divided\_by\_0 với ‘1’.

Dòng 613: Nếu is\_divided\_by\_0 = 1, nhảy đến nhãn divided\_by\_0, ngược lại thực hiện câu lệnh tiếp theo.

Dòng 615: Nạp giá trị ‘2’ cho AH.

Dòng 616: Nạp giá trị ‘0’ cho BH.

Dòng 617: Nạp giá trị biến res\_pos\_x cho DH.

Dòng 618: Nạp giá trị biến res\_pos\_y cho DL.

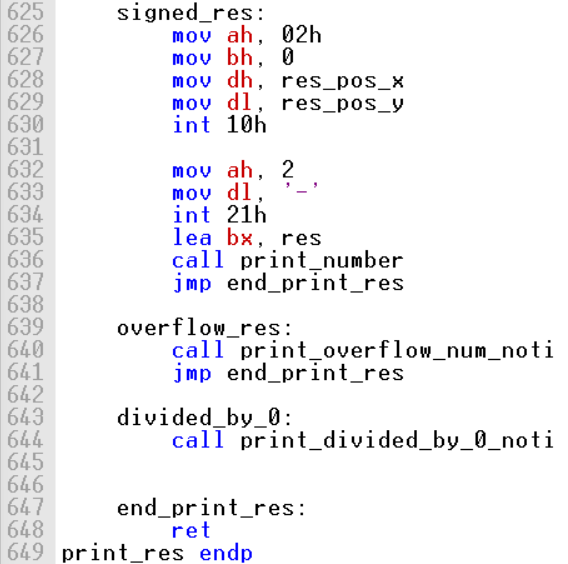
Dòng 619: Gọi hàm 2 của ngắt 10h để thực hiện di chuyển con trỏ về vị trí res\_pos\_x, res\_pos\_y trên màn hình.

Dòng 620: Nạp địa chỉ biến res vào BX. (truyển tham số cho hàm con **print\_number)**.

Dòng 621: Gọi hàm con **print\_number** để in ra res.

Dòng 622: Gọi hàm con **print\_remainder** để in ra số dư (nếu có).

Dòng 623: Thực hiện nhảy đến nhãn end\_print\_res.

Dòng 626: Nạp giá trị ‘2’ vào AH.

Dòng 627: Nạp giá trị ‘0’ vào BH.

Dòng 628: Nạp giá trị biến res\_pos\_x vào DH.

Dòng 629: Nạp giá trị biến res\_pos\_y vào DL.

Dòng 630: Thực hiện hàm 2 của ngắt INT 10h để di chuyển con trỏ về vị trị res\_pos\_x, res\_pos\_y.

Dòng 632: Nạp giá trị ‘2’ vào AH.

Dòng 633: Nạp giá trị kí tự ‘-‘ vào DL.

Dòng 634: Thực hiện hàm 2 của ngắt INT 21h để in ra kí tự DL

Dòng 635: Nạp địa chỉ biến res cho BX là đầu vào của hàm **print\_number**.

Dòng 636: Gọi hàm **print\_number** để in ra biến res.

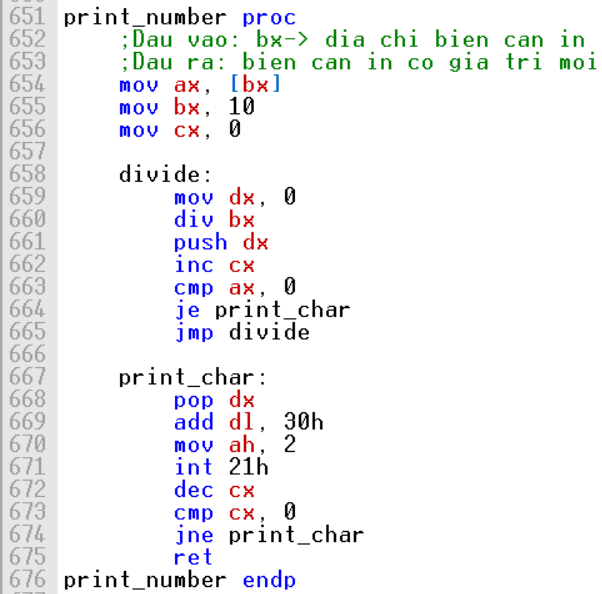
Dòng 637: Nhảy đến nhãn end\_print\_res để kết thúc hàm con.

Dòng 640: Gọi hàm **print\_overflow\_noti** để in ra thông báo kết quả vượt quá giới hạn.

Dòng 641: Nhảy đến nhãn end\_print\_res để kết thúc hàm con.

Dòng 644: Gọi hàm **print\_divided\_by\_0\_noti** để in ra thông báo đây là phép chia cho 0, không hợp lệ.

Dòng 648: Quay lại chương trình đã gọi hàm con này.

* 1.  Hàm con print\_number.

Dòng 654: Nạp giá trị ô nhớ tại địa chỉ DS:BX vào AX.

Dòng 655: Nạp giá trị ‘10’ vào BX.

Dòng 656: Nạp giá trị ‘0’ vào CX, được sử dụng để xác định lượng chữ số cần in.

Dòng 659: Nạp giá trị ‘0’ cho DX đảm bảo thực hiện chia 16 bit với số bị chia là thương của phép tính trước.

Dòng 660: Thực hiện phép chia DXAX cho DX.

Dòng 661: Lưu tạm giá trị DX vào stack, khi đó giá trị của DX chính là giá trị số dư (theo cấu trúc hàm DIV).

Dòng 662: Tăng giá trị CX.

Dòng 663: So sánh AX với 0 (tức so sánh thương của phép chia với 0).

Dòng 664: Nếu AX = 0, nhảy đến nhãn print\_char, ngược lại chạy lệnh tiếp theo.

Dòng 665: Thực hiện nhảy đến nhãn divide.

Dòng 668: Thực hiện lấy các giá trị đã lưu tạm vào stack từ lệnh 661 trước đó, đây chính là giá trị các chữ số của số cần in.

Dòng 669: Thực hiện phép công DL + 30h sau đó lưu vào DL, sử dụng để chuyển từ số sang kí tự ASCII.

Dòng 670: Nạp giá trị ‘2’ vào AH.

Dòng 671: Gọi hàm 2 ngắt INT 21h để thực hiện in kí tự có giá trị DL lên màn hình.

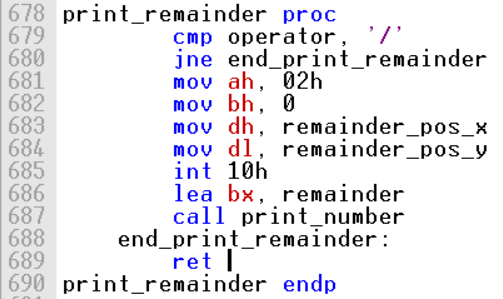
Dòng 672: Giảm giá trị CX đi 1.

Dòng 673: So sánh CX với ‘0’.

Dòng 674: Nếu CX khác ‘0’, nhảy đến nhãn print\_char.

Dòng 675: Quay trở lại chương trình đã gọi hàm này.

* 1. Hàm con print\_remainder để in ra số dư nếu có.

Dòng 679: So sánh operator với kí tự ‘/’

Dòng 680: Nếu operator khác ‘/’ thực hiện nhảy đến nhãn end\_print\_remainder.

Dòng 681: Nạp giá trị ‘2’ vào AH.

Dòng 682: Nạp giá trị ‘0’ vào DH.

Dòng 683: Nạp giá trị biến remainder\_pos\_x vào DH.

Dòng 684: Nạp giá trị biến remainder\_pos\_y vào DL.

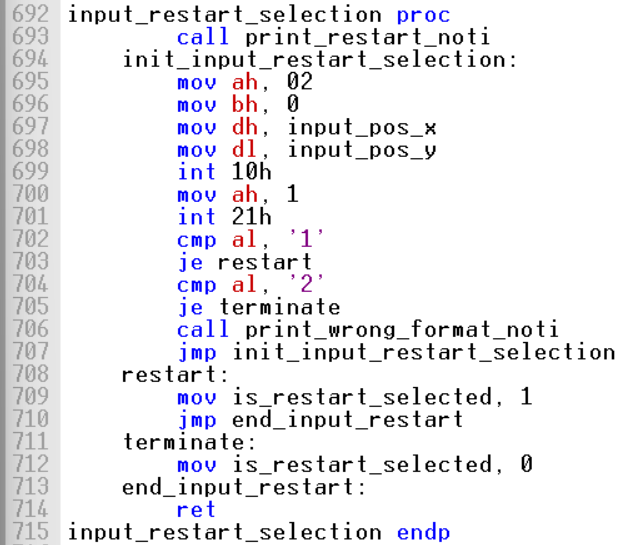
Dòng 685: Thực hiện hàm 2 của ngắt INT 10h để di chuyển con trỏ đến vị trí remainder\_pos\_x, remainder\_pos\_y.

Dòng 686: Nạp giá trị địa chỉ biến remainder vào BX là tham số của hàm **print\_number**.

Dòng 687: Gọi hàm **print\_number** để in ra số dư.

Dòng 689: Quay lại chương trình đã gọi hàm này.

* 1. Hàm con input\_restart\_selection để xác định người dùng có muốn thực hiện tiếp chương trình không.

Dòng 693: Gọi hàm **print\_restart\_noti** để in ra thông báo.

Dòng 695: Nạp giá trị ‘2’ vào AH.

Dòng 696: Nạp giá trị ‘0’ vào BH.

Dòng 697: Nạp giá trị biến input\_pos\_x vào DH.

Dòng 698: Nạp giá trị biến input\_pos\_y vào DL.

Dòng 699: Thực hiện hàm 2 của ngắt INT 10h để di chuyển con trỏ đến vị trí input\_pos\_x, input\_pos\_y.

Dòng 700: Nạp giá trị ‘1’ vào AH.

Dòng 701: Thực hiện hàm 1 của ngắt INT 21h để nhập một kí tự, lưu tại AL.

Dòng 702: So sánh AL với kí tự ‘1’

Dòng 703: Nếu AL bằng kí tự ‘1’, nhảy đến nhãn restart, nếu không. thực hiện lệnh tiếp theo.

Dòng 704: So sánh AL với kí tự ‘2’

Dòng 705: Nếu AL bằng kí tự ‘2’, nhảy đến nhãn terminate, nếu không, thực hiện lệnh tiếp theo.

Dòng 706: Gọi hàm **print\_wrong\_format\_noti** để thông báo nhập sai giá trị cho phép.

Dòng 707: Thực hiện nhảy đến nhãn init\_input\_restart\_selection để thực hiện nhập lại.

Dòng 709: Nạp giá trị ‘1’ cho biến is\_restart\_selected.

Dòng 710: Nhảy đến nhãn end\_input\_restart để chuẩn bị kết thúc hàm con.

Dòng 712: Nạp giá trị ‘0’ cho biến is\_restart\_selected.

Dòng 714: Quay trở lại chương trình đã gọi hàm này.