Bài tập lớn KTMT

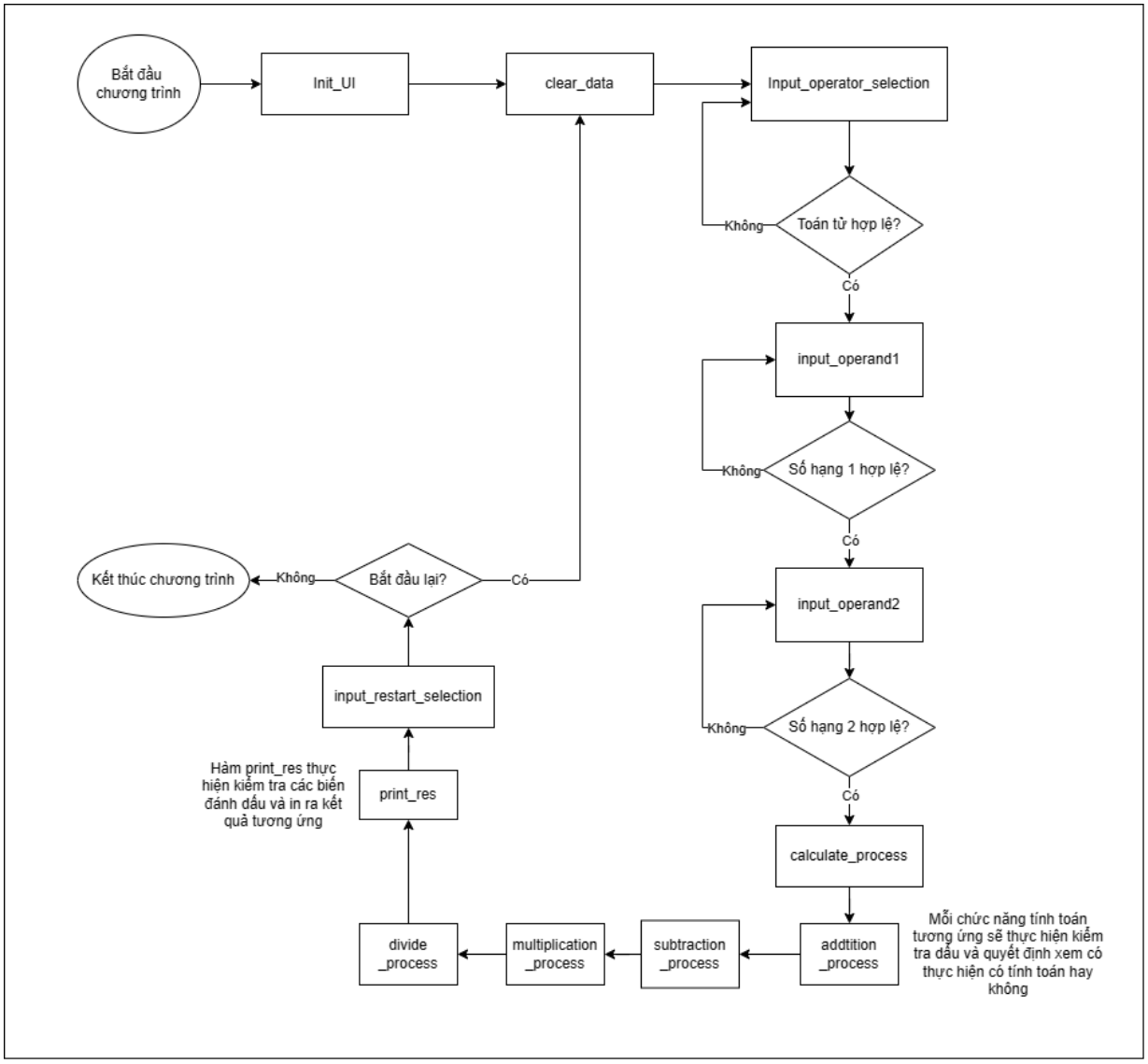
I. Giới thiệu đề tài.

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

II. 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.

III. 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.
4. **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.

5. **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.

o 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.
o 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.
o 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.

6. **Tiếp tục gọi đến hàm input_operand2 để thực hiện nhập toán hạng 2, luồng hoạt động cũng tương tự input_operand1.**
7. **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.
     o 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.
     o 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.**
     o 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.
     o 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**.
     o 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**.

8. **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**).

9. **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 dấu 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.
10. **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.**

IV. Miêu tả chương trình.

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

```
001 .Model Small
002 .Stack 100h
003 .Data
004     operator db 0
005     operand1 dw 0
006     operand2 dw 0
007     res dw 0
008     remainder dw 0
009     is_res_overflow db 0
010     is_restart_selected db 0
011     is_res_signed db 0
012     is_divided_by_0 db 0
013
014     screen_str1 db '--------------------<MINI CALCULATOR>-----------------', '$'
015     screen_str2 db '|1. Cong     2. Tru         |                        |', '$'
016     screen_str3 db '|3. Nhan     4. Chia        |                        |', '$'
017     screen_str4 db '|Trang thai:                                          |', '$'
018     screen_str5 db '|Gia tri nhap:              |Toan tu:                |', '$'
019     screen_str6 db '|Ket qua:                   |Toan hang 1:            |', '$'
020     screen_str7 db '|So du:                     |Toan hang 2:            |', '$'
021     screen_str8 db '-----------------------------------------------------', '$'
```

- 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 vượt 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.

```
023     blank_noti_label db '                                                ', '$'
024     noti_pos_x db 8
025     noti_pos_y db 26
026     operator_input_noti db 'Vui long nhap phep tinh', '$'
027     operand1_input_noti db 'Vui long nhap so hang thu nhat', '$'
028     operand2_input_noti db 'Vui long nhap so hang thu hai', '$'
029     overflow_num_noti db 'So vuot qua gioi han, hay nhap lai', '$'
030     wrong_format_num_noti db 'So sai dinh dang, hay nhap lai', '$'
031     divided_by_0_noti db 'Khong the thuc hien chia cho 0', '$'
032     restart_noti db '     Ban co muon thuc hien lai? 1.Co 2.Khong', '$'
033
034
035     blank_input_label db '              ', '$'
036     input_pos_x db 9
037     input_pos_y db 28
038
039     blank_operator_label db '            ', '$'
040     operator_pos_x db 9
041     operator_pos_y db 51
```

- 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.

```
043     blank_operand_label db '            ', '$'
044     operand1_pos_x db 10
045     operand1_pos_y db 55
046
047     operand2_pos_x db 11
048     operand2_pos_y db 55
049
050     blank_res_label db '              ', '$'
051     res_pos_x db 10
052     res_pos_y db 23
053
054     blank_remainder_label db '                    ', '$'
055     remainder_pos_x db 11
056     remainder_pos_y db 21
057
058     blank_restart_noti_label db '                                                  ', '$'
059     restart_noti_pos_x db 13
060     restart_noti_pos_y db 13
```

- 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.

2. **Chương trình main: chạy chương trình chính.**

```
062 .Code
063 Main proc
064         mov ax, @Data
065         mov ds, ax
066         call init_ui
067     start:
068         call clear_data
069         call input_operator_selection
070         call input_operand1
071         call input_operand2
072         call calculate_process
073         call print_res
074         call input_restart_selection
075         cmp is_restart_selected, 1
076         jne end_main
077         jmp start
078     end_main:
079         call clear_screen
080         mov ah, 4ch
081         int 21h
082 Main endp
```

- 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_selected với ‘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.

3. **Hàm con init_ui: sử dụng để khởi tạo giao diện chính của chương trình.**

```
084 init_ui proc
085         mov ah, 02h
086         mov bh, 0
087         mov dh, 5
088         mov dl, 13
089         int 10h
090         lea dx, screen_str1
091         mov ah, 9
092         int 21h
093
094         mov ah, 02h
095         mov bh, 0
096         mov dh, 6
097         mov dl, 13
098         int 10h
099         lea dx, screen_str2
100         mov ah, 9
101         int 21h
```

- **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.

```
094         mov ah, 02h
095         mov bh, 0
096         mov dh, 6
097         mov dl, 13
098         int 10h
099         lea dx, screen_str2
100         mov ah, 9
101         int 21h
102
103         mov ah, 02h
104         mov bh, 0
105         mov dh, 7
106         mov dl, 13
107         int 10h
108         lea dx, screen_str3
109         mov ah, 9
110         int 21h
```

```
113         mov ah, 02h
114         mov bh, 0
115         mov dh, 8
116         mov dl, 13
117         int 10h
118         lea dx, screen_str4
119         mov ah, 9
120         int 21h
121
122
123         mov ah, 02h
124         mov bh, 0
125         mov dh, 9
126         mov dl, 13
127         int 10h
128         lea dx, screen_str5
129         mov ah, 9
130         int 21h
```

```
132         mov ah, 02h
133         mov bh, 0
134         mov dh, 10
135         mov dl, 13
136         int 10h
137         lea dx, screen_str6
138         mov ah, 9
139         int 21h
140
141
142         mov ah, 02h
143         mov bh, 0
144         mov dh, 11
145         mov dl, 13
146         int 10h
147         lea dx, screen_str7
148         mov ah, 9
149         int 21h
```

```
151         mov ah, 02h
152         mov bh, 0
153         mov dh, 12
154         mov dl, 13
155         int 10h
156         lea dx, screen_str8
157         mov ah, 9
158         int 21h
```

- 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

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

```
163 clear_data proc
164         mov is_res_overflow, 0
165         mov is_restart_selected, 0
166         mov is_res_signed, 0
167         mov is_divided_by_0, 0
168
169         call clear_remainder_label
170         call clear_noti_label
171         call clear_res_label
172         call clear_operand1_label
173         call clear_operand2_label
174         call clear_operator_label
175         call clear_input_label
176         call clear_restart_noti
177         ret
178 clear_data endp
```

- 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.

5. **Các hàm clear_xxx_label: sử dụng để xóa các nhãn hiển thị trên màn hình.**

```
181 clear_remainder_label proc
182         mov ah, 02h
183         mov bh, 0
184         mov dh, remainder_pos_x
185         mov dl, remainder_pos_y
186         int 10h
187         mov ah, 9
188         lea dx, blank_remainder_label
189         int 21h
190         ret
191 clear_remainder_label endp
192
193
194 clear_noti_label proc
195         mov ah, 02h
196         mov bh, 0
197         mov dh, noti_pos_x
198         mov dl, noti_pos_y
199         int 10h
200         mov ah, 9
201         lea dx, blank_noti_label
202         int 21h
203         ret
204 clear_noti_label endp
```

- **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.

```
206 clear_res_label proc
207         mov ah, 02h
208         mov bh, 0
209         mov dh, res_pos_x
210         mov dl, res_pos_y
211         int 10h
212         mov ah, 9
213         lea dx, blank_res_label
214         int 21h
215         ret
216 clear_res_label endp
217
218
219 clear_input_label proc
220         mov ah, 02h
221         mov bh, 0
222         mov dh, input_pos_x
223         mov dl, input_pos_y
224         int 10h
225         mov ah, 9
226         lea dx, blank_input_label
227         int 21h
228         ret
229 clear_input_label endp
```

- 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.

```
232 clear_operand1_label proc
233         mov ah, 02h
234         mov bh, 0
235         mov dh, operand1_pos_x
236         mov dl, operand1_pos_y
237         int 10h
238         mov ah, 9
239         lea dx, blank_operand_label
240         int 21h
241         ret
242 clear_operand1_label endp
243
244
245 clear_operand2_label proc
246         mov ah, 02h
247         mov bh, 0
248         mov dh, operand2_pos_x
249         mov dl, operand2_pos_y
250         int 10h
251         mov ah, 9
252         lea dx, blank_operand_label
253         int 21h
254         ret
255 clear_operand2_label endp
```

- 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.

```
258 clear_operator_label proc
259         mov ah, 02h
260         mov bh, 0
261         mov dh, operator_pos_x
262         mov dl, operator_pos_y
263         int 10h
264         mov ah, 9
265         lea dx, blank_operator_label
266         int 21h
267         ret
268 clear_operator_label endp
269
270
271 clear_restart_noti proc
272         mov ah, 02h
273         mov bh, 0
274         mov dh, restart_noti_pos_x
275         mov dl, restart_noti_pos_y
276         int 10h
277         mov ah, 9
278         lea dx, blank_restart_noti_label
279         int 21h
280         ret
281 clear_restart_noti endp
```

- 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.

6. **Hàm con clear_screen để thực hiện xóa toàn màn hình.**

```
283 clear_screen proc
284         mov ah, 0
285         mov al, 3
286         int 10h
287         ret
288 clear_screen endp
```

- **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.

7. **Hàm con input_operator để thực hiện nhập phép tính muốn thực hiện.**

```
291 input_operator_selection proc
292         call print_operator_input_noti
293     input_operator_start:
294         call clear_input_label
295         mov ah, 02
296         mov bh, 0
297         mov dh, input_pos_x
298         mov dl, input_pos_y
299         int 10h
300
301         mov ah, 1
302         int 21h
303         cmp al,'1'
304         je plus_operator
305         cmp al, '2'
306         je subtract_operator
307         cmp al, '3'
308         je multiply_operator
```

- 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 hiển 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.

```
309         cmp al, '4'
310         je divide_operator
311         call print_wrong_format_noti
312         jmp input_operator_start
313     plus_operator:
314         mov operator, '+'
315         jmp input_operator_end
316     subtract_operator:
317         mov operator, '-'
318         jmp input_operator_end
319     multiply_operator:
320         mov operator, '*'
321         jmp input_operator_end
322     divide_operator:
323         mov operator, '/'
324     input_operator_end:
325
326         call print_operator
327         ret
328 input_operator_selection endp
```

- 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.

8. Hàm con input_number dùng để nhập một số vào một biến.

```
330 input_number proc
331 ;dau vao: bx-> dia chi bien can nhap
332 ;dau ra: bien duoc cap nhat theo gia tri nhap
333     init_input_number:
334         push bx;
335         call clear_input_label
336         mov ah, 02h
337         mov bh, 0
338         mov dh, input_pos_x
339         mov dl, input_pos_y
340         int 10h
341         pop bx
342
343         mov ax, 0;
344         mov [bx], ax
```

- 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.
- Dòng 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.

```
344         mov [bx], ax
345     input_char:
346         mov ah, 1
347         int 21h
348         cmp al, ' '
349         je input_end
350         sub al, '0';
351         cmp al, 0
352         jl input_wrong_format
353         cmp al, 9
354         jg input_wrong_format
355         mov dl, al;
356         mov dh, 0
```

- 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.

```
358         push dx
359         mov ax, [bx]
360         mov dx, 10
361         mul dx
362         cmp dx, 0
363         pop dx
364         jne input_overflow
365
366         add ax, dx
367         mov [bx], ax
368         jmp input_char
369     input_overflow:
370         push bx
371         call print_overflow_num_noti
372         pop bx
373         jmp init_input_number
374     input_wrong_format:
375         push bx
376         call print_wrong_format_noti
377         pop bx
378         jmp init_input_number
379     input_end:
380         ret
381 input_number endp
```

- 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.

```
358         push dx
359         mov ax, [bx]
360         mov dx, 10
361         mul dx
362         cmp dx, 0
363         pop dx
364         jne input_overflow
365
366         add ax, dx
367         mov [bx], ax
368         jmp input_char
369     input_overflow:
370         push bx
371         call print_overflow_num_noti
372         pop bx
373         jmp init_input_number
374     input_wrong_format:
375         push bx
376         call print_wrong_format_noti
377         pop bx
378         jmp init_input_number
379     input_end:
380         ret
381 input_number endp
```

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

9. 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.

```
384 input_operand1 proc
385         call print_operand1_input_noti
386         lea bx, operand1
387         call input_number
388         call clear_input_label
389         lea bx, operand1
390         call print_operand1
391         ret
392 input_operand1 endp
393
394 input_operand2 proc
395         call print_operand2_input_noti
396         lea bx, operand2
397         call input_number
398         call clear_input_label
399         lea bx, operand2
400         call print_operand2
401         ret
402 input_operand2 endp
```

- 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.

10. 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.

```
402 print_operand1 proc
403         mov ah, 02h
404         mov bh, 0
405         mov dh, operand1_pos_x
406         mov dl, operand1_pos_y
407         int 10h
408         lea bx, operand1
409         call print_number
410         ret
411 print_operand1 endp
412
413 print_operand2 proc
414         mov ah, 02h
415         mov bh, 0
416         mov dh, operand2_pos_x
417         mov dl, operand2_pos_y
418         int 10h
419         lea bx, operand2
420         call print_number
421         ret
422 print_operand2 endp
```

- 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.

11. Các hàm con print_xxx_noti được sử dụng để in ra nhãn thông báo trên màn hình.

```
424 print_operand1_input_noti proc
425         call clear_noti_label
426         mov ah, 02h
427         mov bh, 0
428         mov dh, noti_pos_x
429         mov dl, noti_pos_y
430         int 10h
431         lea dx, operand1_input_noti
432         mov ah, 9
433         int 21h
434         ret
435 print_operand1_input_noti endp
436
437 print_operand2_input_noti proc
438         call clear_noti_label
439         mov ah, 02h
440         mov bh, 0
441         mov dh, noti_pos_x
442         mov dl, noti_pos_y
443         int 10h
444         lea dx, operand2_input_noti
445         mov ah, 9
446         int 21h
447         ret
448 print_operand2_input_noti endp
```

- 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.

```
450 print_operator_input_noti proc
451         mov ah, 02h
452         mov bh, 0
453         mov dh, noti_pos_x
454         mov dl, noti_pos_y
455         int 10h
456         lea dx, operator_input_noti
457         mov ah, 9
458         int 21h
459         ret
460 print_operator_input_noti endp
461
462 print_wrong_format_noti proc
463         call clear_noti_label
464         mov ah, 02h
465         mov bh, 0
466         mov dh, noti_pos_x
467         mov dl, noti_pos_y
468         int 10h
469         lea dx, wrong_format_num_noti
470         mov ah, 9
471         int 21h
472         ret
473 print_wrong_format_noti endp
```

- 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.

```
475 print_overflow_num_noti proc
476         call clear_noti_label
477         mov ah, 02h
478         mov bh, 0
479         mov dh, noti_pos_x
480         mov dl, noti_pos_y
481         int 10h
482         lea dx, overflow_num_noti
483         mov ah, 9
484         int 21h
485         ret
486 print_overflow_num_noti endp
487
488 print_divided_by_0_noti proc
489         call clear_noti_label
490         mov ah, 02h
491         mov bh, 0
492         mov dh, noti_pos_x
493         mov dl, noti_pos_y
494         int 10h
495         lea dx, divided_by_0_noti
496         mov ah, 9
497         int 21h
498         ret
499 print_divided_by_0_noti endp
```

```
501 print_restart_noti proc
502         call clear_restart_noti
503         mov ah, 02h
504         mov bh, 0
505         mov dh, restart_noti_pos_x
506         mov dl, restart_noti_pos_y
507         int 10h
508         lea dx, restart_noti
509         mov ah, 9
510         int 21h
511         ret
512 print_restart_noti endp
513
514 print_operator proc
515         mov ah, 02h
516         mov bh, 0
517         mov dh, operator_pos_x
518         mov dl, operator_pos_y
519         int 10h
520         mov dl, operator
521         mov ah, 2
522         int 21h
523         ret
524 print_operator endp
```

- 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 vượt 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.

12. **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.

```
526 calculate_process proc
527         call addition_process
528         call multiplication_process
529         call subtraction_process
530         call divide_process
531         ret
532 calculate_process endp
```

    - 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.
13. **Hàm con addition_process để thực hiện phép tính cộng.**

```
534 addition_process proc
535         cmp operator, '+'
536         jne end_addition_process
537
538         mov ax, operand1
539         add ax, operand2
540         jnc store_addition_res
541         mov is_res_overflow, 1
542         jmp end_addition_process
543     store_addition_res:
544         mov res, ax
545     end_addition_process:
546         ret
547
548 addition_process endp
```

    - 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.

14. Hàm con multiplication_process để thực hiện phép tính nhân.

```
550 multiplication_process proc
551         cmp operator, '*'
552         jne end_multiplication_process
553
554         mov ax, operand1
555         mov bx, operand2
556         mul bx
557         jnc store_multiplication_res
558         mov is_res_overflow, 1
559         jmp end_multiplication_process
560     store_multiplication_res:
561         mov res, ax
562     end_multiplication_process:
563         ret
564 multiplication_process endp
```

- 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.

15. Hàm con subtraction_process dùng để thực hiện phép tính trừ.

```
566 subtraction_process proc
567         cmp operator, '-'
568         jne end_subtraction_process
569
570         mov ax,operand1
571         mov bx,operand2
572         cmp ax, bx
573         jl mark_signed:
574         sub ax, operand2
575         jmp store_subtraction_res
576     mark_signed:
577         mov is_res_signed, 1
578         mov ax,operand2
579         sub ax,operand1
580     store_subtraction_res
581         mov res,ax
582     end_subtraction_process:
583         ret
584 subtraction_process endp
```

- 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.

16. Hàm con divide_process dùng để thực hiện phép tính chia.

```
586 divide_process proc
587         cmp operator, '/'
588         jne end_divide_process
589         mov ax,operand1
590         mov bx,operand2
591         cmp bx, 0
592         je mark_divided_by_0
593         mov dx,0
594         div bx
595         mov res ,ax
596         mov remainder, dx
597         jmp end_divide_process
598
599     mark_divided_by_0:
600         mov is_divided_by_0, 1
601     end_divide_process:
602         ret
603 divide_process endp
```

- 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.

17. 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.

```
605 print_res proc
606         cmp is_res_overflow, 1
607         je overflow_res
608
609         cmp is_res_signed, 1
610         je signed_res
611
612         cmp is_divided_by_0, 1
613         je divided_by_0
614
615         mov ah, 02h
616         mov bh, 0
617         mov dh, res_pos_x
618         mov dl, res_pos_y
619         int 10h
620         lea bx, res
621         call print_number
622         call print_remainder
623         jmp end_print_res
```

- 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. (truyền 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.

```
625     signed_res:
626         mov ah, 02h
627         mov bh, 0
628         mov dh, res_pos_x
629         mov dl, res_pos_y
630         int 10h
631
632         mov ah, 2
633         mov dl, '-'
634         int 21h
635         lea bx, res
636         call print_number
637         jmp end_print_res
638
639     overflow_res:
640         call print_overflow_num_noti
641         jmp end_print_res
642
643     divided_by_0:
644         call print_divided_by_0_noti
645
646
647     end_print_res:
648         ret
649 print_res endp
```

- 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.

18. Hàm con print_number.

```
651 print_number proc
652     ;Dau vao: bx-> dia chi bien can in
653     ;Dau ra: bien can in co gia tri moi
654     mov ax, [bx]
655     mov bx, 10
656     mov cx, 0
657
658     divide:
659         mov dx, 0
660         div bx
661         push dx
662         inc cx
663         cmp ax, 0
664         je print_char
665         jmp divide
666
667     print_char:
668         pop dx
669         add dl, 30h
670         mov ah, 2
671         int 21h
672         dec cx
673         cmp cx, 0
674         jne print_char
675         ret
676 print_number endp
```

- 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.

19. Hàm con print_remainder để in ra số dư nếu có.

```
678 print_remainder proc
679         cmp operator, '/'
680         jne end_print_remainder
681         mov ah, 02h
682         mov bh, 0
683         mov dh, remainder_pos_x
684         mov dl, remainder_pos_y
685         int 10h
686         lea bx, remainder
687         call print_number
688     end_print_remainder:
689         ret
690 print_remainder endp
```

- 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.

20. Hàm con input_restart_selection để xác định người dùng có muốn thực hiện tiếp chương trình không.

```
692 input_restart_selection proc
693         call print_restart_noti
694     init_input_restart_selection:
695         mov ah, 02
696         mov bh, 0
697         mov dh, input_pos_x
698         mov dl, input_pos_y
699         int 10h
700         mov ah, 1
701         int 21h
702         cmp al, '1'
703         je restart
704         cmp al, '2'
705         je terminate
706         call print_wrong_format_noti
707         jmp init_input_restart_selection
708     restart:
709         mov is_restart_selected, 1
710         jmp end_input_restart
711     terminate:
712         mov is_restart_selected, 0
713     end_input_restart:
714         ret
715 input_restart_selection endp
```

- 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.