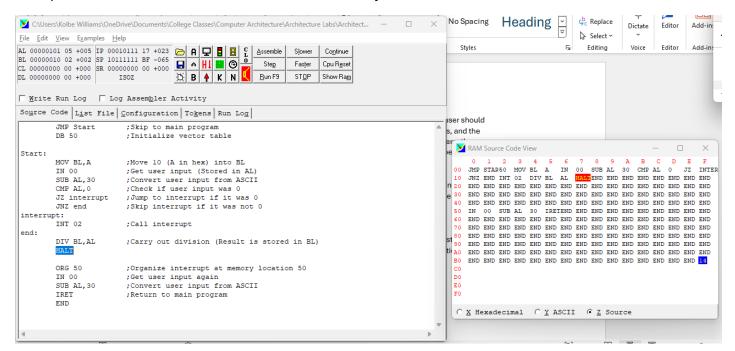
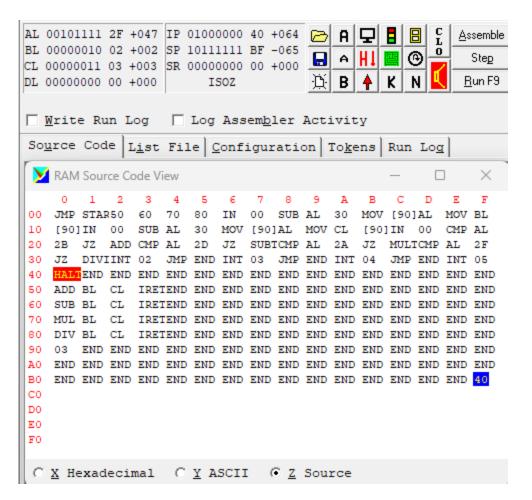
1. Create an interrupt that helps to avoid a "divide by zero" error. The user should be able to enter a number, if the number is zero the interrupt executes, and the program allows the user to reenter the number. If the number is not zero, then the input needs to be converted and divide a value in BL by this number.



I divided the user input (5 in this example) by 10, which is stored in BL

2. Create a calculator program. The user should input two numbers and an option of what operation to carry out. Use software interrupts to carry out the desired operation.

```
JMP Start
                         ;Jump to main program
        DB 50
                         ;Initialize vector table
        DB 60
        DB 70
        DB 80
Start:
        IN 00
                         :Get first number
        SUB AL, 30
                        ;Convert first number from ASCII
        MOV [90],AL
                        ;Move first number to [90]
        MOV BL, [90]
                        ;Move first number into BL
        IN 00
                         ;Get second number
        SUB AL, 30
                        ;Convert second number from ASCII
        MOV [90],AL
                        ;Move second number into [90]
        MOV CL, [90]
                        ;Move second number into CL
        IN 00
                        ;Get operation
                         ;Check if user entered +
        CMP AL, 2B
        JZ add
                        ;Jumpt to add if user entered +
        CMP AL, 2D
                       ;Check if user entered -
        JZ subtract
                        ;Jump to subtract if user entered -
        CMP AL, 2A
                        ;Check if user entered *
        JZ multiply
                        ;Jump to multiply if user entered *
        CMP AL, 2F
                         ;Check if user entered /
        JZ divide
                        ;Jump to divide if user entered /
add:
        INT 02
                        ;Call addition interrupt
        JMP end
subtract:
        INT 03
                        ;Call Subtraction interrupt
        JMP end
multiply:
       INT 04
                     ;Call multiplication
       JMP end
divide:
       INT 05
                     ;Call division interrupt
end:
       HALT
       ORG 50
                     ;Organize addition interrupt
       ADD BL,CL
                    ;Add user input and store the result in BL
       IRET
                     ;Return to main function
       ORG 60
                    Organize subtraction interrupt
       SUB BL,CL
                     ;Subtract second number from the first and store the result in BL
       IRET
                     ;Return to main program
       ORG 70
                     ;Organize multiplication interrupt
       MUL BL,CL
                     ;Multiply user input and store the result in BL
       IRET
                     ;Return to main program
       ORG 80
                    Organize division interrupt
       DIV BL,CL
                    ;Divide second number by the first and store the result in BL
       IRET
                     ;Return to the main program
       END
```



This was the result after entering 6, then 3, and then /

3. Extra Credit (50 points): Improve your calculator program. Use the stack to pass the inputs to the interrupt and the results back to the main function. Additionally, set up your program to exit when the user presses enter.

```
JMP Start
                       ;Jump to main program
        DB 50
                       ;Initialize vector table
        DB 60
        DB 70
        DB 80
Start:
       IN 00
                       ;Get first number
                       ;Check if user input was enter
       CMP AL, OD
       JZ end
                      ;End the program if the user input was enter
       SUB AL, 30
                      ;Convert first number from ASCII
       PUSH AL
                       ; Push the first number onto the stack
       IN 00
                       ;Get second number
                      ;Check if user input was enter
       CMP AL, OD
       JZ end
                      ;End the program if the user input was enter
       SUB AL, 30
                      ;Convert second number from ASCII
                      ; Push the second number onto the stack
       PUSH AL
       IN 00
                      ;Get operation
       CMP AL, OD
                      ;Check if user input was enter
       JZ end
                       ;End the program if the user input was enter
       CMP AL, 2B
                      ;Check if user entered +
       JZ add
                      ;Jumpt to add if user entered +
       CMP AL, 2D
                      ;Check if user entered -
                      ;Jump to subtract if user entered -
        JZ subtract
       CMP AL, 2A
                      ;Check if user entered *
       JZ multiply
                      ;Jump to multiply if user entered *
                       ;Check if user entered /
       CMP AL, 2F
       JZ divide
                       ;Jump to divide if user entered /
add:
       INT 02
                      ;Call addition interrupt
        JMP loop
subtract:
        INT 03
                      ;Call Subtraction interrupt
        JMP loop
multiply:
        INT 04
                      ;Call multiplication
        JMP loop
divide:
       INT 05
                      ;Call division interrupt
loop:
       JMP Start
                      ;Jump to the start until user presses enter
end:
       HALT
```

```
ORG 50
                      :Organize addition interrupt
        POP AL
                      ;Pop return address into AL
        POP CL
                      ;Pop the second number into CL
        POP BL
                      :Pop the first number into BL
       ADD BL.CL
                      ;Add user input and store the result in BL
        PUSH BL
                       ; Push the reult onto the stack
        PUSH AL
                      ; Push the return address back onto the stack
        IRET
                      :Return to main function
       ORG 60
                      ;Organize subtraction interrupt
       POP AL
                      ;Pop return address into AL
        POP CL
                      ;Pop the second number into CL
        POP BL
                       ;Pop the first number into BL
        SUB BL,CL
                       :Subtract second number from the first and store the result in BL
        PUSH BL
                      ; Push the reult onto the stack
        PUSH AT.
                      : Push the return address back onto the stack
        IRET
                      ;Return to main program
        ORG 70
                       :Organize multiplication interrupt
        POP AL
                       ;Pop return address into AL
        POP CL
                        ;Pop the second number into CL
        POP BL
                        ;Pop the first number into BL
        MUL BL, CL
                       ;Multiply user input and store the result in BL
        PUSH BL
                       ; Push the reult onto the stack
        PUSH AL
                        ; Push the return address back onto the stack
        IRET
                        :Return to main program
        ORG 80
                        ;Organize division interrupt
        POP AL
                        ;Pop return address into AL
        POP CL
                        ;Pop the second number into CL
        POP BL
                       ;Pop the first number into BL
                       ;Divide second number by the first and store the result in BL
        DIV BL.CL
        PUSH BL
                        ; Push the reult onto the stack
        PUSH AL
                       :Push the return address back onto the stack
        IRET
                        ;Return to the main program
        END
AL 00001101 OD +013 IP 01001001 49 +073 🗁 A 🖵 🚦 📳
                                                     <u>A</u>ssemble
                                                  C
L
C
BL 00000001 01 +001 SP 10111011 BB -069
                                  □ △ HI ■ ⑤
                                                      Step
CL 00000011 03 +003 SR 00000010 02 +002
                                                     Run F9
                                   <u>₿</u> B 🛉
DL 00000000 00 +000
                      ISOZ
☐ Write Run Log ☐ Log Assembler Activity
Source Code List File Configuration Tokens Run Log
RAM Source Code View
             3
                               8
00 JMP STAR50 60 70 80 IN 00 CMP AL OD JZ END SUB AL 30
10 PUSHAL IN 00 CMP AL OD JZ END SUB AL 30 PUSHAL IN 00
20 CMP AL OD JZ END CMP AL 2B JZ ADD CMP AL 2D JZ SUBTCMP
30 AL 2A JZ MULTCMP AL 2F JZ DIVIINT 02 JMP LOOPINT 03 JMP
40 LOOPINT 04 JMP LOOPINT 05 JMP STARHALTEND END END END END END
50 POP AL POP CL POP BL ADD BL CL PUSHBL PUSHAL IRETEND END
60 POP AL POP CL POP BL SUB BL CL PUSHBL PUSHAL IRETEND END
70 POP AL POP CL POP BL MUL BL CL PUSHBL PUSHAL IRETEND END
80 POP AT, POP CT, POP BT, DTV BT, CT, PUSHBT, PUSHAT, TRETEND END
BO END END END END END END END END END 47 47 01 09 00 06
DO
ΕO
○ X Hexadecimal ○ Y ASCII ⓒ Z Source
```

This was the result after entering each operation (+,-,*,/)with 3 for both numbers each time and then pressing enter to end the program after all 4 operations were carried out and pushed onto the stack.