

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.

The screenshot shows an assembly editor window with the following code:

```

AL 00000101 05 +005 IP 00010111 17 +023
BL 00000010 02 +002 SP 10111111 BF -065
CL 00000000 00 +000 SR 00000000 00 +000
DL 00000000 00 +000 ISOZ

JMP Start ;Skip to main program
DB 50 ;Initialize vector table

Start:
MOV BL,A ;Move 10 (A in hex) into BL
IN 00 ;Get user input (Stored in AL)
SUB AL,30 ;Convert user input from ASCII
CMP AL,0 ;Check if user input was 0
JZ interrupt ;Jump to interrupt if it was 0
JNZ end ;Skip interrupt if it was not 0

interrupt:
INT 02 ;Call interrupt

end:
DIV BL,AL ;Carry out division (Result is stored in BL)
HALT

ORG 50 ;Organize interrupt at memory location 50
IN 00 ;Get user input again
SUB AL,30 ;Convert user input from ASCII
IRET ;Return to main program
END

```

The RAM Source Code View window shows the memory layout of the program, with the interrupt routine starting at address 50.

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
        CMP AL,2B        ;Check if user entered +
        JZ add           ;Jump 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

```

| | | | | | | | | |
|---------------------|---------------------|--|---|---|---|---|-------------|----------|
| AL 00101111 2F +047 | IP 01000000 40 +064 | | A | | | | C L O | Assemble |
| BL 00000010 02 +002 | SP 10111111 BF -065 | | A | H | | | | Step |
| CL 00000011 03 +003 | SR 00000000 00 +000 | | B | | K | N | | Run F9 |
| DL 00000000 00 +000 | ISOZ | | | | | | | |

☐ Write Run Log ☐ Log Assembler Activity

Source Code | List File | Configuration | Tokens | Run Log

RAM Source Code View

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|----|--------|--------|-----|------|-----|-----|--------|-----|-----|--------|-----|--------|------|-----|-----|-----|
| 00 | JMP | STAR50 | 60 | 70 | 80 | IN | 00 | SUB | AL | 30 | MOV | [90]AL | MOV | BL | | |
| 10 | [90]IN | 00 | SUB | AL | 30 | MOV | [90]AL | MOV | CL | [90]IN | 00 | CMP | AL | | | |
| 20 | 2B | JZ | ADD | CMP | AL | 2D | JZ | SUB | CMP | AL | 2A | JZ | MULT | CMP | AL | 2F |
| 30 | JZ | DIVI | INT | 02 | JMP | END | INT | 03 | JMP | END | INT | 04 | JMP | END | INT | 05 |
| 40 | HALT | END | END | END | END | END | END | END | END | END | END | END | END | END | END | END |
| 50 | ADD | BL | CL | IRET | END | END | END | END | END | END | END | END | END | END | END | END |
| 60 | SUB | BL | CL | IRET | END | END | END | END | END | END | END | END | END | END | END | END |
| 70 | MUL | BL | CL | IRET | END | END | END | END | END | END | END | END | END | END | END | END |
| 80 | DIV | BL | CL | IRET | END | END | END | END | END | END | END | END | END | END | END | END |
| 90 | 03 | END | END | END | END | END | END | END | END | END | END | END | END | END | END | END |
| A0 | END | END | END | END | END | END | END | END | END | END | END | END | END | END | END | END |
| B0 | END | END | END | END | END | END | END | END | END | END | END | END | END | END | END | 40 |
| C0 | | | | | | | | | | | | | | | | |
| D0 | | | | | | | | | | | | | | | | |
| E0 | | | | | | | | | | | | | | | | |
| F0 | | | | | | | | | | | | | | | | |

☐ X Hexadecimal
☐ Y ASCII
☒ Z Source

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
        CMP AL,0D          ;Check if user input was enter
        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
        CMP AL,0D          ;Check if user input was enter
        JZ end             ;End the program if the user input was enter
        SUB AL,30          ;Convert second number from ASCII
        PUSH AL            ;Push the second number onto the stack
        IN 00              ;Get operation
        CMP AL,0D          ;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             ;Jump 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 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 result 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 result onto the stack
PUSH AL     ;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 result 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
DIV BL,CL   ;Divide second number by the first and store the result in BL
PUSH BL     ;Push the result onto the stack
PUSH AL     ;Push the return address back onto the stack
IRET        ;Return to the main program
END

```

| | | | | | | | | | |
|---------------------|---------------------|--|--|--|--|--|--|--|----------|
| AL 00001101 0D +013 | IP 01001001 49 +073 | | | | | | | | Assemble |
| BL 00000001 01 +001 | SP 10111011 BB -069 | | | | | | | | Step |
| CL 00000011 03 +003 | SR 00000010 02 +002 | | | | | | | | Run F9 |
| DL 00000000 00 +000 | ISOZ | | | | | | | | |

☐ Write Run Log ☐ Log Assembler Activity

Source Code | List File | Configuration | Tokens | Run Log

RAM Source Code View

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|----|---------|--------|-----|---------|-----|-----|------|--------|-----|--------|---------|---------|-----|-----|---------|----|
| 00 | JMP | STAR50 | 60 | 70 | 80 | IN | 00 | CMP | AL | 0D | JZ | END | SUB | AL | 30 | |
| 10 | PUSHAL | IN | 00 | CMP | AL | 0D | JZ | END | SUB | AL | 30 | PUSHAL | IN | 00 | | |
| 20 | CMP | AL | 0D | JZ | END | CMP | AL | 2B | JZ | ADD | CMP | AL | 2D | JZ | SUBTCMP | |
| 30 | AL | 2A | JZ | MULTCMP | AL | 2F | JZ | DIVINT | 02 | JMP | LOOPINT | 03 | JMP | | | |
| 40 | LOOPINT | 04 | JMP | LOOPINT | 05 | JMP | STAR | HALT | END | END | 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 | AL | POP | CL | POP | BL | DIV | BL | CL | PUSHBL | PUSHAL | IRETEND | END | | | |
| 90 | END | END | END | END | END | END | END | END | END | END | END | END | END | END | END | |
| A0 | END | END | END | END | END | END | END | END | END | END | END | END | END | END | END | |
| B0 | END | END | END | END | END | END | END | END | END | END | 47 | 47 | 01 | 09 | 00 | 06 |
| C0 | | | | | | | | | | | | | | | | |
| D0 | | | | | | | | | | | | | | | | |
| E0 | | | | | | | | | | | | | | | | |
| F0 | | | | | | | | | | | | | | | | |

☒ 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.