Problem #3: Write a short program that will accept a series of numbers (up to 20 numbers) and will calculate the average of the numbers.

The program will start from DOS, and when finished, will return to DOS. The program will be written in assembler. The program will first ask the user to enter the numbers (using a DECIN procedure which you can assume is available to link to, but make sure the program knows about his label!). You will hit return, with no input, to signify the end of the list. The program will then calculate the integer part of the average and report the result to the screen (Printing the integer average using the DE Cout procedure, which you can assume is available to link to). The program should work for all numbers of samples \( N \), including \( N = 0 \) and above (1, 2, 3, 4, 5, etc.). You should test for the \( N = 0 \) case and print an appropriate remark if this is the case.

Be sure to define stack and data segments. The final program should be in a 'ready to compile' form of a .ASM program. You do not have to write out the actual DECIN and DE Cout procedures, just assume they are available to link to and call them in the proper manner.

See attached sheet.
DSEG SEGMENT
MSG1 DB 'PLEASE ENTER NUMBER$: ','
MSG2 DB 'NUMBER OF NUMBERS ENTERED: $'
MSG3 DB 'SUM OF ALL NUMBERS: $'
MSG4 DB 'AVERAGE OF ALL NUMBERS: $'
MSG5 DB 'NO NUMBERS ENTERED: NO AVERAGE', 0DH, 0AH, '$'
CRLF DB 0DH, 0AH, '$'
SUM DW 0
NLIST DW 0
DSEG ENDS

SSEG SEGMENT STACK
DW 100H DUP(?)
SSEG ENDS

CSEG SEGMENT
ASSUME CS:CSEG, SS:SSEG, DS:DSEG
EXTERN DECIN:FAR
EXTERN DECOUT:FAR
MAIN PROC FAR
START: PUSH DS
MOV AX, 0
PUSH AX

MOV AX, DSEG
MOV DS, AX

MOV AH, 09
MOV DX, OFFSET MSG1
INT 21H
MOV AH, 09
MOV DX, OFFSET CRLF
INT 21H

GETN: CALL DECIN
CMP AX, 0
JL FINISH
INC NLIST
ADD SUM, AX
CMP NLIST, 20H
JE FINISH
JMP GETN

FINISH: CMP NLIST, 0
JE ZERO

MOV AH, 09
MOV DX, OFFSET MSG2
INT 21H
MOV AX, NLIST

CALL DEcout
MOV AH, 09
MOV DX, OFFSET CRLF
INT 21H

MOV AH, 09
MOV DX, OFFSET MSG3
INT 21H
MOV AX, SUM

CALL DEcout
MOV AH, 09
MOV DX, OFFSET CRLF
INT 21H

MOV AH, 09
MOV DX, OFFSET MSG4
INT 21H

MOV DX, 0
MOV AX, SUM
MOV BX, NLIST
DIV BX

CALL DEcout

MOV AH, 09
MOV DX, OFFSET CRLF
INT 21H

JMP DONE

ZERO: MOV AH, 09
MOV DX, OFFSET MSG5
INT 21H

DONE: RET
MAIN ENDP
CSEG ENDS

END START
Problem #1: At the beginning of a IBM PC assembly language Code segment program, the following instructions are executed:

MOV AX, 0562H
MOV BX, 0504H
MOV CX, 99CDH
MOV DX, 0881H
PUSH AX
PUSH CL
PUSH DH
PUSH BX
POP AL
POP BX
POP AH
POP DX
POP CX

Write down the contents of the AX, BX, CX, and DX registers after these instructions have been executed.

After pushers
Stack has

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>04</td>
<td>05</td>
</tr>
<tr>
<td>09</td>
<td>0D</td>
</tr>
<tr>
<td>62</td>
<td>05</td>
</tr>
</tbody>
</table>

So pop AL gives

AL ≤ 04

pop BX gives

BX ≤ 0805

pop AH gives

AH ≤ CD

pop DX gives

DX ≤ 0562

pop CX indeterminate →

depends on unknown stuff on the stack.
Problem #2: Suppose we wanted to redirect the interrupt vector for software interrupt 4BH to point to an ISR located in segment 4073H, with the code beginning on line 194DH of this segment.

1. What are the full addresses (both segment and offset) for the first byte of the location of the interrupt vector for software interrupt 4BH.

2. After the interrupt vector has been changed to point to the new interrupt service routine, the contents of the 4 consecutive bytes of memory beginning at the address you have specified above, are changed. Write down the contents of these 4 memory locations. To make it absolutely clear to the grader, specify both the segment and offset of each consecutive byte, and the contents of that byte. No credit will be given unless all three items are listed for each byte.

\[ 4B = \begin{array}{c} 0100 \\ 1011 \end{array} \]

\[ 4B + \text{4B} = \begin{array}{c} 1000 \\ 1011 \\ 00 \end{array} \]

\[ 0 \text{Z} \text{Z} \text{C} \text{H} \]

So interrupt vector is at 0000:022C

\[
\begin{array}{c}
\text{offset} \\
\text{segment} \\
0000:022C \\
0000:022D \\
0000:022E \\
0000:022F \\
\end{array}
\]

\[
\begin{array}{c}
4D \\
19 \\
93 \\
40 \\
\end{array}
\]
Write a short program that determines if an inputted number is a prime number. The program will start from DOS, and when finished, will return to DOS. The program will be written in assembler.

The program will first ask the user to enter the number (using a DECIN procedure which you can assume is available to link to, but make sure the program knows about his label!). The program will then determine if the number is prime and report the result to the screen (Printing a message 'NUMBER IS PRIME' or 'NUMBER IS NOT PRIME').

The program should work for N=1 and above (1, 2, 3, 5, 7, etc.).

The easiest way to determine if a number is prime is to see if it is divisible by any number smaller than it, with a remainder of zero from the division. Note that the assembler division command returns the remainder, so the procedure should be straightforward.

Be sure to define stack and data segments. The final program should be in a 'ready to compile' form.

```
DSEG SEGMENT
   MSG1 DB 'NUMBER IS PRIME $'
   MSG2 DB 'NUMBER IS NOT PRIME $'
DSEG ENDS
SSEG SEGMENT STACK
   DWORD Loop Dup (?)
SSEG ENDS
CSEG SEGMENT
   ASSUME DS:DSEG, SS:SSEG, CS:CSEG
EXTRN DECIN % FAR
Main Proc FAR
First %
   PUSH DS
   MOV AX, 0
   PUSH AX
   MOV AX,DSEG
   MOV DS, AX
   CALL DECIN
```
CMP AX, 0H
JL DONE
CMP AX, 03H
JLE PRIME
MOV CX, AX
MOV BX, 2

NEXT:
MOV DX, 0
MOV AX, CX
DIV BX
CMP DX, 0
JE NOTPRIME
INC BX
CMP BX, CX
JL NEXT

PRIME:
MOV AL, 09H
MOV DX, OFFSET MSB1
INT 21H
JMP DONE

NOTPRIME:
MOV AL, 09H
MOV DX, OFFSET MSB2
INT 21H

DONE:
RET
MAIN ENDP
CSEG ENDS
END FIRST
The Fibonacci sequence is generated by the following rules:

- The first two elements are 1 and 1.
- The next element is calculated as the sum of the previous two elements.

The first seven elements of the sequence are therefore:
1, 1, 2, 3, 5, 8, 13

You are to write the code segment for a program that will generate the Nth Fibonacci number. Assume that the number N has been stored in AX. The answer should be stored in DX at the end of the program. You may write the solution in ‘pseudo-DEBUG’. Pseudo-DEBUG is identical to DEBUG, except that you have the added capability of using label for lines (e.g. JMP NEXT). Note that the LOOP instruction will also work as it is part of the 8088 instruction set. You code should work for $1 \leq N \leq 22$

There are many solutions here. Here is an easy one:

```assembly
    MOV BX, 1
    MOV DX, 1
    SUB AX, 02 ; take care of first two elements
    JLE DONE
    MOV CX, AX

    LOOP1: Push DX
             add DX, BX
             POP BX
             LOOP LOOP1

    DONE:  NOP
```
(A) Which of the following addresses point to the same absolute memory location?

1. CCDE:0505 and CAED:2415
   \[ \frac{CCDE}{0505} = \frac{CAED}{2415} \]
   Same (\#1)

2. 1F34:8176 and 2703:0387
   \[ \frac{1F34}{8176} \neq \frac{2703}{0387} \]
   Different (\#2)

3. 99FF:FED0 and A833:1B90
   \[ \frac{99FF}{FED0} \neq \frac{A833}{1B90} \]
   Cannot be the same!!
   Same (\#3)

(B) Each of the following instructions has an error which will not allow the compiler to build the object file. Identify the error and fix it so that the statement will execute the desired function. Note that you may need additional lines to accomplish the fix.

1. XCHG [BX], [BX+2]; SWAP MEMORY BYTE BX with BX+2

2. MOV DS, 459H ; INITIALIZE DS
   No direct addressing of DS
   MOV DS, 459H

3. MOV CX, 09 ; LOOP SUBTRACTION
   LOOP%1: MOV AX, [BX+2]
   SUB AX, CX
   INC BX
   INC [BX]
   LOOPZ LOOP%1
   Cannot use LOOP%1 label
   Use loops instead
   Use loops instead

4. INT AH ; ACCESS INTERRUPT NUMBER TEN
   AH is a register. \#AH in interrupt 10
   INT \#AH

5. XCHG BX, [CX] ; SWAP REGISTER WITH MEMORY LOCATION
   Cannot use CX for indirect addressing

   Only BX can be used for that
   One possible solution
   \[ \text{Push DX} \]
   MOV DX, BX
   MOV BX, CX
   XCHG DX, [BX]
   MOV BX, DX
   Pop DX
Physics 563 Midterm Exam #1

Problem #1: You are given the following IBM PC assembler program:

PUSH BX
SUB DX,AX
JS TEST ←
POP BX
MOV [CX]2, AH
MUL CX
TEST: POP AX
DIV BX

At the point marked with the arrow, an external interrupt occurs on IRQ7. Given the following dump of segment 00 of memory, what are the values of CS and IP of the FIRST INSTRUCTION of the Interrupt Service Routine?

ADDRESS = XXXX + Y

<table>
<thead>
<tr>
<th>XXXX</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>90</td>
<td>A1</td>
<td>DD</td>
<td>31</td>
<td>09</td>
<td>00</td>
<td>32</td>
<td>33</td>
<td>86</td>
<td>32</td>
<td>15</td>
<td>67</td>
<td>35</td>
<td>D1</td>
<td>0E</td>
<td>A4</td>
</tr>
<tr>
<td>0010</td>
<td>CE</td>
<td>CB</td>
<td>4D</td>
<td>2F</td>
<td>FF</td>
<td>09</td>
<td>4E</td>
<td>9A</td>
<td>11</td>
<td>40</td>
<td>00</td>
<td>82</td>
<td>21</td>
<td>BB</td>
<td>BE</td>
<td>27</td>
</tr>
<tr>
<td>0020</td>
<td>97</td>
<td>8E</td>
<td>5F</td>
<td>2B</td>
<td>0F</td>
<td>79</td>
<td>DE</td>
<td>6A</td>
<td>71</td>
<td>48</td>
<td>C0</td>
<td>F2</td>
<td>11</td>
<td>0B</td>
<td>BO</td>
<td>28</td>
</tr>
<tr>
<td>0030</td>
<td>61</td>
<td>DB</td>
<td>40</td>
<td>F4</td>
<td>F0</td>
<td>0D</td>
<td>4D</td>
<td>9F</td>
<td>1E</td>
<td>10</td>
<td>09</td>
<td>85</td>
<td>2A</td>
<td>43</td>
<td>FE</td>
<td>A7</td>
</tr>
<tr>
<td>0040</td>
<td>84</td>
<td>7F</td>
<td>53</td>
<td>FE</td>
<td>EF</td>
<td>05</td>
<td>E4</td>
<td>DA</td>
<td>47</td>
<td>4F</td>
<td>4E</td>
<td>72</td>
<td>61</td>
<td>BD</td>
<td>80</td>
<td>2C</td>
</tr>
<tr>
<td>0050</td>
<td>E6</td>
<td>A2</td>
<td>50</td>
<td>26</td>
<td>6A</td>
<td>63</td>
<td>78</td>
<td>00</td>
<td>8E</td>
<td>DD</td>
<td>62</td>
<td>10</td>
<td>04</td>
<td>08</td>
<td>C5</td>
<td>B1</td>
</tr>
</tbody>
</table>

IRQ7 has vector: 20H + 7 * 4 = 28D

432AH = IP

A7FEH = CS.

+4 pts for calculating address
+6 pts for getting CS IP
Problem #1: a) Just before the execution of the instruction indicated by the arrow, the registers of the PC have the following values.

0CC3:0100 MOV AX, 0005H
0CC3:0104 MOV BX, 0004H
0CC3:0107 MOV CX, 0002H
0CC3:010A MOV DX, 0000H
0CC3:010D FFF: MOV DS:[BX], CH ←
0CC3:010F MOV DS:[BX][-2], AH
0CC3:0112 DEC BX
0CC3:0113 MOV DS:[BX], CX

AX=0005 BX=0004 CX=0002 DX=0000 SP=FFEE SI=0000 DI=0000
DS=0CC3 ES=0CC3 SS=0CC3 IP=0100 FLAG = 7202

At this point in time, before the indicated instruction is executed, hardware interrupt IRQC occurs. If segment 0000 of memory has the following contents, calculate the segment and offset of the first line of the interrupt service routine that will get executed when IRQC occurs.

\[ 4 \times C + 20H \]

\[ \Rightarrow C = 1100 \quad 4 \times C = 110000 = 30H \]

\[ 4 \times C + 30H = 50H \]

\[ \text{IP} = \text{A2EF} \]

\[ \text{CS} = \text{2650} \]

<table>
<thead>
<tr>
<th>XXXX</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>90</td>
<td>A1</td>
<td>DD</td>
<td>31</td>
<td>09</td>
<td>00</td>
<td>32</td>
<td>33</td>
<td>86</td>
<td>32</td>
<td>15</td>
<td>67</td>
<td>35</td>
<td>D1</td>
<td>0E</td>
<td>A4</td>
</tr>
<tr>
<td>0010</td>
<td>CE</td>
<td>CB</td>
<td>4D</td>
<td>2F</td>
<td>FF</td>
<td>09</td>
<td>4E</td>
<td>9A</td>
<td>11</td>
<td>40</td>
<td>00</td>
<td>82</td>
<td>21</td>
<td>BB</td>
<td>BE</td>
<td>27</td>
</tr>
<tr>
<td>0020</td>
<td>97</td>
<td>8E</td>
<td>5F</td>
<td>2B</td>
<td>0F</td>
<td>79</td>
<td>DE</td>
<td>6A</td>
<td>71</td>
<td>48</td>
<td>C0</td>
<td>F2</td>
<td>11</td>
<td>0B</td>
<td>B0</td>
<td>28</td>
</tr>
<tr>
<td>0030</td>
<td>61</td>
<td>DB</td>
<td>40</td>
<td>F4</td>
<td>F0</td>
<td>0D</td>
<td>4D</td>
<td>9F</td>
<td>1E</td>
<td>10</td>
<td>09</td>
<td>85</td>
<td>2A</td>
<td>43</td>
<td>FE</td>
<td>A7</td>
</tr>
<tr>
<td>0040</td>
<td>84</td>
<td>7E</td>
<td>53</td>
<td>FE</td>
<td>EF</td>
<td>05</td>
<td>E4</td>
<td>DA</td>
<td>47</td>
<td>4F</td>
<td>4E</td>
<td>72</td>
<td>61</td>
<td>BD</td>
<td>80</td>
<td>2C</td>
</tr>
<tr>
<td>0050</td>
<td>FE</td>
<td>A2</td>
<td>50</td>
<td>26</td>
<td>6A</td>
<td>63</td>
<td>78</td>
<td>00</td>
<td>8E</td>
<td>DD</td>
<td>62</td>
<td>10</td>
<td>04</td>
<td>08</td>
<td>C5</td>
<td>B1</td>
</tr>
</tbody>
</table>
b) The stack had the following values before the interrupt occurred. What will the value of SP after the interrupt occurs, when the computer begins to execute the first line of the ISR? List all of the changes in the stack values (list the changed memory location + new value) at this point in time.

SEGMENT 0CC3
ADDRESS = XXXX + Y

1. push IF
2. push CS
3. push IP

\[ \text{push does} \]
\[ \begin{align*}
\text{①} & \text{ inc SP} \\
\text{②} & \text{ write high-order byte} \\
\text{③} & \text{ inc SP} \\
\text{④} & \text{ write low-order byte}
\end{align*} \]

\[ \text{FFE8} \quad 9 \quad A \quad B \quad C \quad D \]
\[ \text{0D} \quad 01 \quad C3 \quad 0C \quad 02 \quad 72 \]

\[ \text{OA} \]
Also acceptable (ambiguous)

<table>
<thead>
<tr>
<th>XXXX</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFA0</td>
<td>90</td>
<td>A1</td>
<td>DD</td>
<td>31</td>
<td>09</td>
<td>00</td>
<td>32</td>
<td>33</td>
<td>86</td>
<td>32</td>
<td>15</td>
<td>67</td>
<td>35</td>
<td>D1</td>
<td>0E</td>
<td>A4</td>
</tr>
<tr>
<td>FFB0</td>
<td>CE</td>
<td>CB</td>
<td>4D</td>
<td>2F</td>
<td>FF</td>
<td>09</td>
<td>4E</td>
<td>9A</td>
<td>11</td>
<td>40</td>
<td>00</td>
<td>82</td>
<td>21</td>
<td>B6</td>
<td>BE</td>
<td>27</td>
</tr>
<tr>
<td>FFC0</td>
<td>97</td>
<td>8E</td>
<td>5F</td>
<td>2B</td>
<td>0F</td>
<td>79</td>
<td>DE</td>
<td>6A</td>
<td>01</td>
<td>48</td>
<td>C0</td>
<td>F2</td>
<td>31</td>
<td>3B</td>
<td>B0</td>
<td>28</td>
</tr>
<tr>
<td>FFDO</td>
<td>61</td>
<td>DB</td>
<td>40</td>
<td>F4</td>
<td>F0</td>
<td>0D</td>
<td>4D</td>
<td>9F</td>
<td>1F</td>
<td>10</td>
<td>09</td>
<td>85</td>
<td>2A</td>
<td>43</td>
<td>FE</td>
<td>A7</td>
</tr>
<tr>
<td>FFE0</td>
<td>84</td>
<td>7F</td>
<td>53</td>
<td>FE</td>
<td>7F</td>
<td>05</td>
<td>74</td>
<td>FA</td>
<td>47</td>
<td>4F</td>
<td>4E</td>
<td>72</td>
<td>61</td>
<td>7D</td>
<td>80</td>
<td>2C</td>
</tr>
<tr>
<td>FFF0</td>
<td>76</td>
<td>A2</td>
<td>50</td>
<td>26</td>
<td>6A</td>
<td>E3</td>
<td>78</td>
<td>00</td>
<td>8E</td>
<td>DD</td>
<td>62</td>
<td>10</td>
<td>14</td>
<td>09</td>
<td>C5</td>
<td>B1</td>
</tr>
</tbody>
</table>
Consider the following section of assembler code. If DS = 4DFCH, what will be stored in the first 10D bytes of segment 4DFC after the code section has finished execution?

First time through loop:

\[ A_0 = 0021 \]
\[ B_0 = 0 \]
\[ C_0 = 04H \]

Now, \( B_0 \) gets pushed onto stack.

\[ B_0 = 04 \]
\[ 0021 \times 4 \]

\[ MUL B_0 \] makes \( A_0 \leftarrow A_0 \times B_0 = 0084 \]

\[ POP B_0 \rightarrow B_0 = 0 \]

\[ MOV [BX, [SI]], A_0 \rightarrow 21 \leftarrow \text{stores 21 at 0000} \] 3 Big and aw!

Now, \( B_0 = 1 \), \( SI = 1 \) and loop \( C_0 = 9 \)

Next \( SI \) will be \( 12 \) \( B_0 \) \( 0004 \) to location \( B_0 + SI = 02 \)

\[ 0000 \quad 0000 \quad 0100 \quad 0100 \]

\[ 
\text{Next Loop} 8 \\
\text{Next Loop} 7 \\
\text{Next Loop} 6 \\
\text{Next Loop} 5 \\
\text{Next Loop} 4 \\
\text{Next Loop} 3 \\
\text{Next Loop} 2 \\
\text{Next Loop} 1 \\
\text{Next Loop} 0 \\
\]

Final Result 6

| 4DFC 0000 | 084H |
| 0001 | 00H |
| 0002 | 10H |
| 0003 | 02H |
| 0004 | 40H |

| 4DFC 0005 | 08H |
| 0006 | 0FH |
| 0007 | 21H |
| 0008 | 0FH |
| 0009 | 84H |
Problem #3: The following main routine calls a procedure

2FDA:0100 call MYPROC ← push $E$ Return segment, then return address $E$ push 2FDA
2FDA:0103 ADD BX,AX

Inside the function MYPROC, we call another procedure DUMB-PROC
3430:0123 call DUMB-PROC → push 3430
3430:0126 SUB BX,CX push $126$

The procedure DUMB-PROC consists of only the following lines:
4412:012A MOV BX,0A0EH $Bx=0A0EH$
4412:012D PUSH BX push $A0EH$
4412:012E NOP ←
4412:012F RET

The computer process is stopped at the point of the arrow.

If we execute the following commands from the place where the process stopped

POP AX → $A0EH$
POP BX → $126$
POP BX → 3430
POP CX → $103$
POP DX → 2FDA

What are the values of AX, BX, CX, and DX?

$AX = A0EH$
$BX = 3430$
$CX = 103$
$DX = 2FDA$
Problem #3: Write a short program that will accept a series of numbers (up to 20 numbers) and will calculate the average of the numbers.

The program will start from DOS, and when finished, will return to DOS. The program will be written in assembler. The program will first ask the user to enter the numbers (using a DECIN procedure which you can assume is available to link to, but make sure the program knows about his label!). You will hit return, with no input, to signify the end of the list. The program will then calculate the integer part of the average and report the result to the screen (Printing the integer average using the DECOUT procedure, which you can assume is available to link to). The program should work for all numbers of samples $N$, including $N=0$ and above (1, 2, 3, 4, 5, etc.). You should test for the $N=0$ case and print an appropriate remark if this is the case.

Be sure to define stack and data segments. The final program should be in a 'ready to compile' form of a .ASM program. You do not have to write out the actual DECIN and DECOUT procedures, just assume they are available to link to and call them in the proper manner.

See attached sheet.
DSEG SEGMENT
MSG1 DB 'PLEASE ENTER NUMBER$: '
MSG2 DB 'NUMBER OF NUMBERS ENTERED: '$
MSG3 DB 'SUM OF ALL NUMBERS: '$
MSG4 DB 'AVERAGE OF ALL NUMBERS: '$
MSG5 DB 'NO NUMBERS ENTERED: NO AVERAGE', 0DH, 0AH, '$'
CRLF DB 0DH, 0AH, '$'
SUM DW 0
NLIST DW 0
DSEG ENDS

SSEG SEGMENT STACK
DW 100H DUP(?)
SSEG ENDS

CSEG SEGMENT
ASSUME CS:CSEG, SS:SSEG, DS:DSEG
EXTRN DECIN:FAR
EXTRN DECOUT:FAR
MAIN PROC FAR
START: PUSH DS
MOV AX, 0
PUSH AX

MOV AX, DSEG
MOV DS, AX

MOV AH, 09
MOV DX, OFFSET MSG1
INT 21H
MOV AH, 09
MOV DX, OFFSET CRLF
INT 21H

GETN: CALL DECIN
CMP AX, 0
JL FINISH
INC NLIST
ADD SUM, AX
CMP NLIST, 20H
JE FINISH
JMP GETN

FINISH: CMP NLIST, 0
JE ZERO

MOV AH, 09
MOV DX, OFFSET MSG2
INT 21H
MOV AX, NLIST
 CALL DECOUT
 MOV AH, 09
 MOV DX, OFFSET CRLF
 INT 21H

MOV AH, 09
 MOV DX, OFFSET MSG3
 INT 21H
 MOV AX, SUM

CALL DECOUT
 MOV AH, 09
 MOV DX, OFFSET CRLF
 INT 21H

MOV AH, 09
 MOV DX, OFFSET MSG4
 INT 21H

MOV DX, 0
 MOV AX, SUM
 MOV BX, NLIST
 DIV BX

CALL DECOUT

MOV AH, 09
 MOV DX, OFFSET CRLF
 INT 21H

JMP DONE

ZERO: MOV AH, 09
 MOV DX, OFFSET MSG5
 INT 21H

DONE: RET
 MAIN ENDP
 CSEG ENDS

END START
Problem #2:

1. What are the full addresses (both segment and offset) for the first byte of the location of the interrupt vector for hardware interrupt IRQ4?

2. Write a interrupt service routine that will increment a memory location called 'STATUS' that has been defined as a word in the data segment each time the hardware interrupt is called.

3. To enable this interrupt, you must unmask the hardware interrupt. Write down the assembly language code necessary to do this.

(1) \texttt{SEGMENT 0000} \\
\texttt{IRQ4 OFFSET} - 4 \times 4 + 20H = 30H \\
\texttt{0000:0030H} \\

(2) \texttt{MY_ISR: STI} \\
\texttt{PUSH DS} \\
\texttt{PUSH AX} \\
\texttt{MOV AX, 'SEG\_STATUS'} \\
\texttt{MOV DS, AX} \\
\texttt{INC STATUS} \\
\texttt{MOV AL, 20H} \\
\texttt{OUT 20H, AL} \\
\texttt{POP AX} \\
\texttt{POP DS} \\
\texttt{IRET} \\

(3) \texttt{IN AL, 21H} \\
\texttt{AND AL, 11101111B} \\
\texttt{OUT 21H, AL}