**1. Assembly Solution:**

;10055 - Hashmat the Brave Warrior

.MODEL SMALL

.STACK 100H

.CODE

MAIN PROC

START:

CALL SCAN

MOV BH, DL ;move first value in bh

CALL SCAN

MOV BL, DL ;move second value in bl

SUB BH, BL ;SUBTRACT THE INPUT VALUES

MOV AX, 0 ;CLEAR AX

MOV AL, BH ;MOVE SUBTRACTION TO AL FOR PRINT PROC

CMP AL, 0

JG P ;IF SUBTRACTION IS NOT GREATER THAN ZERO

NEG AL ;NEGATE SUBTRACTION

P:

CALL PRINT

MOV AH, 2

MOV DL, 10 ;PRINT NEW LINE

INT 21H

MOV DL, 13 ;PRINT CARRIGE RETURN

INT 21H

JMP START ;TAKE INPUT AGAIN

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**2. Assembly Solution:**

;10071 - Back to High School Physics

ORG 100h

MAIN PROC

START:

CALL SCAN ;TAKE V

MOV BH, DL

CALL SCAN ;TAKE T

MOV BL, DL

MOV AX, 0 ;CLEAR AX

MOV AL, BH

IMUL BL ;AX WILL STORE V\*T

MOV BL, 2

IMUL BL ;AX WILL STORE 2VT

CALL PRINT

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**3. Assembly Solution :**

;10079 - Pizza Cutting

;UNLIKE THE UVA PROBLEM

;THIS PROGRAM IS TERMINATE ON INPUTTING ZERO

ORG 100h

MAIN PROC

START:

CALL SCAN ;INPUT NUMBER OF LINES

CMP DL, 0 ;IF INPUT IS ZERO

JE ENDMAIN ;TERMINATE

MOV CX, 0 ;CLEAR CX

MOV CL, DL

MOV AX, 1 ;MOVED 1 TO AX

SUM:

ADD AL, CL

LOOP SUM

CALL PRINT

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START ;TAKE INPUT AGAIN

ENDMAIN:

RET

ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**4. Assembly Solution :**

;10499 - The Land of Justice

ORG 100H

.DATA

nl DB '%', 10, 13, '$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV BX, 25

MOV CX, 4

TESTCASE:

CALL SCAN

CMP AX, 0

JL ENDCASE ;If negative, terminate

CMP AX, 1

JG CALC

MOV AX, 0

JMP PRINTRES

CALC:

IMUL BX

PRINTRES:

CALL PRINT

MOV AH, 9

LEA DX, nl

INT 21h

JMP TESTCASE

ENDCASE:

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**5. Assembly Solution :**

;10970 - Big Chocolate

ORG 100h

MAIN PROC

START:

CALL SCAN ;INPUT M

MOV BH, DL

CALL SCAN ;INPUT N

MOV AX, 0

MOV AL, DL

IMUL BH

DEC AL

CALL PRINT ;OUTPUT RESULT

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START ;TAKE INPUT AGAIN

ENDMAIN:

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**6. Assembly Solution :**

;11172 - Relational Operator

ORG 100h

MAIN PROC

CALL SCAN ;INPUT TESTCASE

MOV CX, 0 ;CLEAR CX

MOV CL, DL

MOV AH, 2

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

TESTCASE:

PUSH CX

CALL SCAN ;TAKE FIRST INPUT

MOV BH, DL

CALL SCAN ;TAKE SECOND INPUT

MOV AH, 2

MOV BL, DL

CMP BH, BL

JG GREATER

JL LESS

EQUAL: ;PRINT EQUAL

MOV DL, '='

INT 21H

JMP NEWLINE

GREATER:

MOV DL, '>'

INT 21H

JMP NEWLINE

LESS:

MOV DL, '<'

INT 21H

NEWLINE:

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

POP CX

LOOP TESTCASE

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**7. Assembly Solution :**

;11799 - Horror Dash

ORG 100H

.DATA

n DW ?

max DW ?

case DW ?

c DW ?

msg DB 'CASE $'

.CODE

MAIN PROC

START:

MOV AX,@DATA

MOV DS,AX

CALL SCAN

XOR CX,CX

MOV CX,AX

MOV CASE,0

CASE1:

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

PUSH CX

MOV MAX,0

CALL SCAN

MOV CX,AX

CASE2:

CALL SCAN

MOV c,AX

CMP AX,MAX

JG EQUAL

LOOP CASE2

JMP NEXT

EQUAL:

MOV AX,c

MOV MAX,AX

LOOP CASE2

NEXT:

POP CX

INC CASE

LEA DX,MSG

MOV AH,9

INT 21h

MOV AX,CASE

CALL PRINT

MOV AH,2

MOV DX,':'

INT 21h

MOV DX,' '

INT 21h

MOV AX,MAX

CALL PRINT

LOOP CASE1

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

JMP START

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUT

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUT:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUT

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**8. Assembly Solution :**

;11805 - Bafana Bafana

ORG 100H

.DATA

case1 DB 'Case $'

case2 DB ': $'

nl DB 10, 13, '$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

CALL SCAN

MOV CX, AX

MOV BX, 1

TESTCASE:

PUSH CX

MOV AH, 9

LEA DX, case1

INT 21h

MOV AX, BX ;Case no

CALL print

PUSH BX

MOV AH, 9

LEA DX, case2

INT 21h

CALL scan

MOV BX, AX ;n

CALL scan

MOV CX, AX ;k

CALL scan ;p

ADD AX, CX ;k+p

CWD

IDIV BX

MOV AX, DX ;(k+p)%n

OR AX, AX

JNZ NOTZERO

ADD AX, BX

NOTZERO:

CALL PRINT

MOV AH, 9

LEA DX, nl

INT 21h

POP BX

INC BX

POP CX

LOOP TESTCASE

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**9. Assembly Solution :**

;10346 - Peter's Smoke

ORG 100H

MAIN PROC

TESTCASE:

CALL SCAN

CMP AX, -1

JE ENDCASE:

MOV BX, AX

CALL SCAN

XCHG BX, AX ;AX = n, BX = k

MOV CX, AX ;CX = total cigarette

CALCULATE:

CWD

IDIV BX

CMP AX, 0

JZ ENDCULATE

ADD CX, AX

ADD AX, DX

JMP CALCULATE

ENDCULATE:

MOV AX, CX

CALL PRINT

MOV AH, 2

MOV DL, 10

INT 21h

MOV DL, 13

INT 21h

JMP TESTCASE

ENDCASE:

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDFILE

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDFILE:

MOV AX, -1

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**10. Assembly Solution :**

;10696 - f91

ORG 100h

MAIN PROC

START:

CALL SCAN ;TAKE INPUT IN DL

CMP DL, 0

JE ENDMAIN ;IF DL = 0, TERMINATE

;COMPARE

CMP DL, 100

JG GREATER

MOV DL, 91

JMP RESULT

GREATER:

SUB DL, 10

RESULT:

MOV AX, 0

MOV AL, DL

CALL PRINT

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START ;TAKE INPUT AGAIN

ENDMAIN:

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**11. Assembly Solution :**

;10783 - Odd Sum

ORG 100h

MAIN PROC

CALL SCAN ;INPUT TESTCASE

MOV CX, 0 ;CLEAR CX

MOV CL, DL

MOV AH, 2

MOV DL, 10 ;PRINT NEW LINE

INT 21H

MOV DL, 13 ;PRINT CARRIEG RETURN

INT 21H

TESTCASE:

PUSH CX

CALL SCAN

MOV BL, DL

CALL SCAN

MOV BH, DL

;INCREAMENT BL IF POSITIVE

MOV CH, BL

AND CH, 1

JNZ JUMP1

INC BL

JUMP1:

;DECREAMENT BH IF POSITIVE

MOV CH, BH

AND CH, 1

JNZ JUMP2

DEC BH

JUMP2:

;SUMMATION

MOV AX, 0 ;CLEAR AX

SUMMATION:

ADD AL, BL

ADD BL, 2

CMP BL, BH

JLE SUMMATION

;PRINT VALUE AT AL

CALL PRINT

MOV AH, 2

MOV DL, 10 ;PRINT NEW LINE

INT 21H

MOV DL, 13 ;PRINT CARRIEG RETURN

INT 21H

POP CX

LOOP TESTCASE

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**12. Assembly Solution :**

;10812 - Beat the Spread!

ORG 100h

.DATA

IMP DB 'impossible', '$'

NL DB 10, 13, '$'

A DB ?

B DB ?

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

CALL SCAN ;INPUT TESTCASE

MOV CX, 0 ;CLEAR CX

MOV CL, DL

LEA DX, NL

MOV AH, 9

INT 21H

TESTCASE:

PUSH CX

CALL SCAN ;INPUT SUM

MOV [A], DL

CALL SCAN ;INPUT DIFF

MOV [B], DL

CMP [A], DL

;IF SUM IS LESS THAN DIFF, IMPOSSIBLE

JL IMPOSSIBLE

MOV CX, 0 ;CLEAR CX

ADD CL, [A]

ADD CL, [B]

AND CL, 1

;IF SUM OF SUM AND DIFF IS ODD, IMPOSSIBLE

JNZ IMPOSSIBLE

CWD

MOV AX, 0

ADD AL, [A]

ADD AL, [B]

MOV CX, 2

IDIV CX ;DIVIDE AL BY 2

CALL PRINT ;PRINT GREATER NUMBER STORED IN AL

MOV DL, ' '

INT 21H ;PRINT SPACE

CWD

MOV AX, 0

MOV AL, [A]

SUB AL, [B]

MOV CX, 2

IDIV CX ;DIVIDE AL BY 2

CALL PRINT ;PRINT SMALLER NUMBER STORED IN AL

JMP NEWLINE

IMPOSSIBLE:

LEA DX, IMP

MOV AH, 9

INT 21H

NEWLINE:

LEA DX, NL

MOV AH, 9

INT 21H

POP CX

LOOP TESTCASE

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

**13. Assembly Solution :**

;10929 - You can say 11

ORG 100h

.DATA

input DB 1002 dup(?)

yes DB ' is a multiple of 11', 10, 13, '$'

no DB ' is not a multiple of 11', 10, 13, '$'

.CODE

MOV AX, @DATA

MOV DX, AX

TESTCASE:

MOV SI, 0

MOV CX, 0

MOV AH, 1

INT 21h

CMP AL, '0'

JE ENDCASE

INPUTLOOP:

MOV SI[input], AL

INC SI

INT 21h

CMP AL, 13

JZ ENDINPUT

JMP INPUTLOOP

ENDINPUT:

MOV SI[input], 36

MOV CX, SI

DEC CX

DEC CX

MOV BX, 10 ;for multiplication

XOR AX, AX

MOV AL, 0[input]

SUB AL, 30h

IMUL BX

ADD AL, 1[input]

SUB AL, 30h

MOV SI, 2

CMP CX, 1

JL COMP

DIVISION:

CWD

MOV BX, 11

IDIV BX

MOV AX, DX

MOV BX, 10

IMUL BX

ADD AL, SI[input]

SUB AL, 30h

INC SI

LOOP DIVISION:

COMP:

PUSH AX

MOV AH, 9

LEA DX, input

INT 21h

POP AX

MOV BX, 11

CWD

IDIV BX

CMP DX, 0

JZ PRINTYES

MOV AH, 9

LEA DX, no

INT 21h

JMP TESTCASE

PRINTYES:

MOV AH, 9

LEA DX, yes

INT 21h

JMP TESTCASE

ENDCASE:   
RET

**14. Assembly Solution :**

;11219 - How old are you?

ORG 100H

.DATA

inv DB 'Invalid birth date', 10, 13, '$'

chk DB 'Check birth date', 10, 13, '$'

nl DB 10, 13, '$'

d1 DW ?

d2 DW ?

m1 DW ?

m2 DW ?

y1 DW ?

y2 DW ?

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

CALL SCAN

MOV CX, AX

TESTCASE:

PUSH CX

CALL SCAN ;Reads the new line

CALL SCAN

MOV [d1], AX

CALL SCAN

MOV [m1], AX

CALL SCAN

MOV [y1], AX

CALL SCAN

MOV [d2], AX

CALL SCAN

MOV [m2], AX

CALL SCAN

MOV [y2], AX

;Calculate the result

MOV AX, [y1]

SUB AX, [y2]

MOV BX, [m1]

CMP BX, [m2]

JL MLESS

JG PRINTRES

MOV BX, [d1]

CMP BX, [d2]

JGE PRINTRES

MLESS:

DEC AX

PRINTRES:

CMP AX, 0

JL PRINTINV

CMP AX, 130

JG PRINTCHK

CALL PRINT

MOV AH, 9

LEA DX, nl

JMP ENDMAIN

PRINTCHK:

MOV AH, 9

LEA DX, chk

JMP ENDMAIN

PRINTINV:

MOV AH, 9

LEA DX, inv

ENDMAIN:

INT 21h

POP CX

LOOP TESTCASE

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

CMP AL, '/'

JE ENDINPUT

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**15. Assembly Solution :**

;11479 - Is this the easiest problem

ORG 100h

.DATA

INV DB 'Invalid$'

EQU DB 'Equilateral$'

ISO DB 'Isosceles$'

SCA DB 'Scalene$'

NL DB 10, 13, '$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

CALL SCAN ;INPUT TEST CASE

MOV CX, 0 ;CLEAR CX

MOV CL, DL

TESTCASE:

PUSH CX

;INPUT THREE SIDES

CALL SCAN

MOV BH, DL

CALL SCAN

MOV BL, DL

CALL SCAN

MOV CH, DL

;CHECK IF INVALID

XOR AX, AX ;CLEAR AX

MOV AL, BH

ADD AL, BL

CMP AL, CH

JL Invalid

XOR AX, AX ;CLEAR AX

MOV AL, BH

ADD AL, CH

CMP AL, BL

JL Invalid

XOR AX, AX ;CLEAR AX

MOV AL, CH

ADD AL, BL

CMP AL, BH

JL Invalid

;COMPARE THE SIDES

CMP BH, BL

JNE NEQUAL

CMP BH, CH

JE Equilateral ;EVERY SIDE IS EQUAL

JNE Isosceles ;JUST BH AND BL ARE EQUAL

NEQUAL:

CMP BH, CH

JE Isosceles ;JUST BH AND CH ARE EQUAL

CMP BL, CH

JE Isosceles ;JUST BL AND CH ARE EQUAL

JMP Scalene ;NONE OF THEM ARE EQUAL

;OUTPUT

Invalid:

MOV AH, 9

LEA DX, INV

INT 21H

JMP NEWLINE

Equilateral:

MOV AH, 9

LEA DX, EQU

INT 21H

JMP NEWLINE

Isosceles:

MOV AH, 9

LEA DX, ISO

INT 21H

JMP NEWLINE

Scalene:

MOV AH, 9

LEA DX, SCA

INT 21H

NEWLINE:

LEA DX, NL

INT 21H

POP CX

LOOP TESTCASE

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

**16. Assembly Solution :**

;11727 - Cost Cutting

ORG 100H

.DATA

msg1 DB 'Case $'

msg2 DB ': $'

nl DB 10, 13, '$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

CALL SCAN ;Testcase

MOV CX, AX

MOV AX, 1

TESTCASE:

PUSH CX

PUSH AX

CALL SCAN

MOV CX, AX

CALL SCAN

MOV BX, AX

CALL SCAN

CMP AX, BX

JG AXG

CMP BX, CX

JG AXCX

MOV DX, BX ;CX > BX > AX

JMP DONE

AXCX: ;BX > AX, CX

CMP AX, CX

JL CXMID

MOV DX, AX ;BX > AX > CX

JMP DONE

CXMID: ;BX > CX > AX

MOV DX, CX

JMP DONE

AXG:

CMP AX, CX

JG BXCX

MOV DX, AX ;CX > AX > BX

JMP DONE

BXCX: ;AX > BX, CX

CMP BX, CX

JL CXMID

MOV DX, BX ;AX > BX > CX

DONE:

MOV CX, DX ;CX holds result

MOV AH, 9

LEA DX, msg1

INT 21h

POP AX

CALL PRINT

INC AX

PUSH AX

MOV AH, 9

LEA DX, msg2

INT 21h

MOV AX, CX

CALL PRINT

MOV AH, 9

LEA DX, nl

INT 21h

POP AX

POP CX

LOOP TESTCASE

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**17. Assembly Solution :**

;1176

;A Benevolent Josephus

org 100h

.model small

.stack 100h

.data

.code

main proc

start:

mov ah,1

int 21h

add al,3

mov dl,al

mov ah,2

int 21h

;print new line

mov ah,2

mov dl,0dh

int 21h

mov dl,0ah

int 21h

jmp start

main endp

mov ah,4ch

int 21h

end main

**18. Assembly Solution :**

;11854 - Egypt

ORG 100h

.DATA

RIGHT DB 'right', 10, 13, '$'

WRONG DB 'wrong', 10, 13, '$'

.CODE

MAIN PROC

START:

CALL SCAN

MOV BH, DL

CALL SCAN

MOV BL, DL

CALL SCAN

MOV CH, DL

CMP BH, 0

JNE CALC

CMP BL, 0

JNE CALC

CMP CH, 0

JE ENDMAIN

CALC:

XOR AX, AX ;CLEAR AX

MOV AL, BH

IMUL AL

MOV BH, AL

XOR AX, AX ;CLEAR AX

MOV AL, BL

IMUL AL

ADD BH, AL

XOR AX, AX ;CLEAR AX

MOV AL, CH

IMUL AL

CMP AL, BH

JE PRINTRIGHT

LEA DX, WRONG

MOV AH, 9

INT 21H

JMP START ;TAKE INPUT AGAIN

PRINTRIGHT:

LEA DX, RIGHT

MOV AH, 9

INT 21H

JMP START ;TAKE INPUT AGAIN

ENDMAIN:

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

**19. Assembly Solution :**

;11875 - Brick Game

ORG 100H

.DATA

case1 DB 'Case $'

case2 DB ': $'

nl DB 10, 13, '$'

a DW 10 dup (?)

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX ;Initialize data stack

CALL SCAN ;Input testcase

MOV CX, AX

MOV AX, 1 ;Case number

TESTCASE:

PUSH CX

PUSH AX

CALL SCAN ;Input Size

MOV CX, AX

MOV BX, 0

INPUTLOOP:

CALL SCAN

MOV BX[a], AX

ADD BX, 2 ;Increament BX by 2

LOOP INPUTLOOP

SUB BX, 2

SHR BX, 1 ;Point to middle value

MOV AH, 9

LEA DX, case1

INT 21h

POP AX

CALL PRINT ;Print case number

INC AX

PUSH AX

MOV AH, 9

LEA DX, case2

INT 21h

MOV AX, BX[a]

CALL PRINT

MOV AH, 9

LEA DX, nl

INT 21h

POP AX

POP CX

LOOP TESTCASE

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**20. Assembly Solution :**

;11879 - Multiple of 17

ORG 100h

.DATA

input DB 1002 dup(?)

yes DB '1', 10, 13, '$'

no DB '0', 10, 13, '$'

.CODE

MOV AX, @DATA

MOV DS, AX

TESTCASE:

MOV SI, 0

MOV CX, 0

MOV AH, 1

INT 21h

CMP AL, '0'

JE ENDCASE

INPUTLOOP:

MOV SI[input], AL

INC SI

INT 21h

CMP AL, 13

JZ ENDINPUT

JMP INPUTLOOP

ENDINPUT:

MOV SI[input], 36

MOV CX, SI

DEC CX

XOR AX, AX

MOV AL, 0[input]

SUB AL, 30h

MOV SI, 1

CMP CX, 1

JL COMP

DIVISION:

MOV BX, 10 ;muliplier

IMUL BX

ADD AL, SI[input]

INC SI

SUB AL, 30h

CWD ;clear DX for division

MOV BX, 17 ;divisor

IDIV BX

MOV AX, DX

LOOP DIVISION:

COMP:

CMP DX, 0

JZ PRINTYES

MOV AH, 9

LEA DX, no

INT 21h

JMP TESTCASE

PRINTYES:

MOV AH, 9

LEA DX, yes

INT 21h

JMP TESTCASE

ENDCASE:

RET

**21. Assembly Solution :**

;11984

;Thermal change

org 100h

.model small

.stack 100h

.data

i db 0

.code

main proc

mov ah,1

int 21h

sub al,48

mov bl,al

start:

cmp i,bl

jg exit

inc i

;;; float !!! :-(

jmp start

exit:

main endp

mov ah,4ch

int 21h

end main

**22. Assembly Solution :**

;12372 - Packing for holidays

ORG 100H

.DATA

CASE DW ?

A DW ?

B DW ?

C DW ?

msg DB 'case $'

msg1 DB 'good$'

msg2 DB 'bad$'

.CODE

MAIN PROC

START:

MOV CASE,0

MOV AX,@DATA

MOV DS,AX

CALL SCAN

XOR CX,CX

MOV CX,AX

LOOP1:

INC CASE

CALL SCAN

MOV A,AX

CALL SCAN

MOV B,AX

CALL SCAN

MOV C,AX

CMP A,20

JLE N1

JMP BAD

N1:

CMP B,20

JLE N2

JMP BAD

N2:

CMP C,20

JLE GOOD

JMP BAD

good:

MOV AH,9

LEA DX,msg

INT 21h

MOV AX,CASE

CALL PRINT

MOV AH,2

MOV DX,':'

INT 21h

MOV DX,' '

INT 21h

MOV AH,9

LEA DX,msg1

INT 21h

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

LOOP LOOP1

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

JMP START

BAD:

MOV AH,9

LEA DX,msg

INT 21h

MOV AX,CASE

CALL PRINT

MOV AH,2

MOV DX,':'

INT 21h

MOV DX,' '

INT 21h

MOV AH,9

LEA DX,msg2

INT 21h

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

LOOP LOOP1

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

JMP START

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUT

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUT:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUT

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**23. Assembly Solution:**

;12468 - Zapping

ORG 100H

MAIN PROC

MAINTOP:

CALL SCAN

MOV BX, AX

CALL SCAN

;If a input is zero then terminate

CMP AX, -1

JZ MAINEND

SUB AX, BX

CMP AX, 0

JG ABS

NEG AX

ABS:

CMP AX, 50

JLE PRINTRES

MOV BX, AX

MOV AX, 100

SUB AX, BX

PRINTRES:

CALL PRINT

MOV AH, 2

MOV DL, 10

INT 21h

MOV DL, 13

INT 21h

JMP MAINTOP

MAINEND:

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**24. Assembly Solution :**   
  
org 100h

.model small

.stack 100h

.data

m db "KABIR$"

.code

mov ah,9 ;string output

lea dx,m ; lea points data address of m

int 21h

mov ah,2

mov dl,0dh

int 21h

mov dl,0ah

int 21h

**25. Assembly Solution :**

;12577 - Hajj-e-Akbar

ORG 100H

.DATA

Hajj DB 'Hajj-e-Akbar', 10, 13, '$'

Umrah DB 'Hajj-e-Asghar', 10, 13, '$'

.CODE

MAIN PROC

START:

MOV AH, 1

INT 21H

MOV BL, AL

HUDDAI:

INT 21H

CMP AL, 13

JNE HUDDAI

CMP BL, '\*'

JE ENDMAIN

CMP BL, 'H'

JE AKBAR

LEA DX, UMRAH

MOV AH, 9

INT 21H

JMP START

AKBAR:

LEA DX, Hajj

MOV AH, 9

INT 21H

JMP START

ENDMAIN:

RET

ENDP

**26. Assembly Solution :**

;12646 - Zero or One

ORG 100h

MAIN PROC

START:

MOV AH, 1

INT 21H ;INPUT FOR ALICE

MOV BH, AL

MOV AH, 2

MOV DL, ' '

INT 21H

MOV AH, 1

INT 21H ;INPUT FOR BETO

MOV BL, AL

MOV AH, 2

MOV DL, ' '

INT 21H

MOV AH, 1

INT 21H ;INPUT FOR CLARA

MOV CH, AL

MOV AH, 2

MOV DL, ' '

INT 21H

CMP BH, BL

JNE NEQUAL

CMP BH, CH

JE NONE

MOV AH, 2

MOV DL, 'C' ;CLARA IS DIFFERENT

INT 21H

JMP NEWLINE

NEQUAL:

CMP BH, CH

JNE NEQUAL2

MOV AH, 2

MOV DL, 'B'

INT 21H

JMP NEWLINE

NEQUAL2:

MOV AH, 2

MOV DL, 'A'

INT 21H

JMP NEWLINE

NONE:

MOV AH, 2

MOV DL, '\*'

INT 21H

NEWLINE:

MOV AH, 2

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START

ENDMAIN:

RET

ENDP

**27. Assembly Solution :**

;Little master

;12704

;incomplete

org 100h

.model small

.stack 100h

.data

db i 0

.code

main proc

mov ah,1

int 21h

sub al,48

mov bl,al

repeat:

mov ah,1 ;input x

int 21h

sub al,48

mov cl,al

mov ah,1 ;input y

int 21h

sub al,48

mov ch,al

mov ah,1 ;input r

int 21h

sub al,48

mov bl,al

int i

cmp bl,i

jg exit

jmp repeat

exit:

endp

mov ah,4ch

int 21h

end main

**28. Assembly Solution :**

;12708 - GCD the largest

ORG 100H

MAIN PROC

CALL SCAN

MOV CX, AX

TESTCASE:

PUSH CX

CALL SCAN

MOV CX, 1

MOV BX, 1 ;Initial GCD

OUTERLOOP:

CMP CX, AX

JGE ENDOUTER

MOV DX, CX

INC DX

INNERLOOP:

CMP DX, AX

JG ENDINNER

CMP CX, DX

JZ CONTINUE

PUSH AX

CALL GCD

CMP AX, BX

JLE CONTINUE

MOV BX, AX

CONTINUE:

POP AX

INC DX

JMP INNERLOOP

ENDINNER:

INC CX

JMP OUTERLOOP

ENDOUTER:

MOV AX, BX

CALL PRINT

MOV AH, 2

MOV DL, 10

INT 21h

MOV DL, 13

int 21h

POP CX

LOOP TESTCASE

RET

ENDP

;Stores the GCD of CX & DX in AX

GCD PROC

PUSH CX

PUSH DX

MOV AX, DX

GCDCALC:

CWD

IDIV CX

OR DX, DX

JZ ENDGCD

MOV AX, CX

MOV CX, DX

JMP GCDCALC

ENDGCD:

MOV AX, CX

POP DX

POP CX

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**29. Assembly Solution :**

;136 - Ugly Number

ORG 100h

.DATA

UN DB 'The 1500', 39,'th ugly number is 859963392.', 10, 13, '$'

.CODE

MOV AX, @DATA

MOV DS, AX

LEA DX, UN

MOV AH, 9

INT 21H

RET

**30. Assembly Solution :**

;1727

;Cost Cutting

org 100h

.model small

.stack 100h

.data

newline db 10,13,"$"

a db 0

b db 0

c db 0

.code

main proc

;input first num

mov ah,1

int 21h

sub al,48

mov a,al

;input second num

mov ah,1

int 21h

sub al,48

mov b,al

;input third num

mov ah,1

int 21h

sub al,48

mov c,al

;print a new line

mov ah,9

lea dx,newline

int 21h

; compare a and b

mov bl,a

cmp bl,b

jg aGb

jl bGa

jmp exit:

aGb:

;compare b and c (a>b)

mov bl,b

cmp bl,c

jg abc

jl cGb

jmp exit

cGb:

;compare a and c (a>b , c>b)

mov bl,a

cmp bl,c

jg acb

jl cab

jmp exit

bGa:

;compare a and c (b>a)

mov bl,a

cmp bl,c

jg bac

jl cGa

jmp exit

cGa:

;compare b and c (c>a)

mov bl,b

cmp bl,c

jg bca

jl cba

jmp exit

abc:

; mid = b

mov ah,2

mov dl,b

add dl,48

int 21h

jmp exit

acb:

; mid = c

mov ah,2

mov dl,c

add dl,48

int 21h

jmp exit

bac:

;mid = a

mov ah,2

mov dl,a

add dl,48

int 21h

jmp exit

bca:

; mid = c

mov ah,2

mov dl,c

add dl,48

int 21h

jmp exit

cab:

; mid = a

mov ah,2

mov dl,a

add dl,48

int 21h

jmp exit

cba:

;mid = b

mov ah,2

mov dl,b

add dl,48

int 21h

jmp exit

exit:

main endp

mov ah,4ch

int 21h

end main

**31. Assembly Solution :**

;299 - Train Swapping

ORG 100H

.DATA

msg1 DB 'Optimal train swapping takes $'

msg2 DB ' swaps', 10, 13, '$'

a DB 50 dup(?)

.CODE

MAIN PROC

CALL SCAN ;Testcase

MOV CX, AX

TESTCASE:

PUSH CX

CALL SCAN

MOV CX, AX ;Size of array

XOR BX, BX

INPUTLOOP:

CALL SCAN

MOV BX[a], AL

INC BX

LOOP INPUTLOOP

MOV CX, BX

DEC CX ;CX holds (intial CX) -1

XOR AX, AX ;Holds the number of swaps

XOR BX, BX

BUBBLESORT:

PUSH BX

PUSH CX

INNERLOOP:

MOV DL, BX[a]

INC BX

CMP DL, BX[a]

JL LESS

MOV DH, BX[a]

DEC BX

MOV BX[a], DH

INC BX

MOV BX[a], DL

INC AX ;Increament number of swaps

LESS:

LOOP INNERLOOP

POP CX

POP BX

LOOP BUBBLESORT

PUSH AX

MOV AH, 9

LEA DX, msg1

INT 21h

POP AX

CALL PRINT

MOV AH, 9

LEA DX, msg2

INT 21h

POP CX

LOOP TESTCASE

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**32. Assembly Solution :**

;694 - Collatz Sequenc

ORG 100H

.DATA

msg1 DB 'Case $'

msg2 DB ': A = $'

msg3 DB ', limit = $'

msg4 DB ', number of terms = $'

nl DB 10, 13, '$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV DX, 1

TESTCASE:

PUSH DX

CALL SCAN

MOV BX, AX

CALL SCAN

CMP AX, -1

JE ENDCASE

MOV CX, 1

PUSH BX

CALCULATE:

CMP BX, 1

JZ ENDCULATE

CMP BX, AX

JG LIMIT

TEST BX, 1

JZ EVEN

PUSH AX

MOV AX, 3

IMUL BX

ADD AX, 1

MOV BX, AX

POP AX

JMP INCREAMENT

EVEN:

SAR BX, 1

INCREAMENT:

INC CX

JMP CALCULATE

LIMIT:

DEC CX

ENDCULATE:

POP BX

POP DX

PUSH AX

PUSH DX

MOV AH, 9

LEA DX, msg1

INT 21h

POP AX ;Value of DX

CALL PRINT

PUSH BX

MOV BX, AX ;BX holds case number

MOV AH, 9

LEA DX, msg2

INT 21h

POP AX ;Pops the lower value

CALL PRINT

MOV AH, 9

LEA DX, msg3

INT 21h

POP AX

CALL PRINT

MOV AH, 9

LEA DX, msg4

INT 21h

MOV AX, CX

CALL PRINT

MOV AH, 9

LEA DX, nl

INT 21h

MOV DX, BX

INC DX

JMP TESTCASE

ENDCASE:

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

**33. Assembly Solution :**   
  
;102 - Ecological Been Preminum

ORG 100H

.DATA

f DW ?

a DW ?

c DW ?

d DW ?

x DW ?

.CODE

MAIN PROC

START:

CALL SCAN

XOR CX,CX

MOV CX,AX

loop1:

PUSH CX

MOV x,0

CALL scan

MOV CX,AX

loop2:

CALL scan

MOV a, AX

CALL scan

CALL scan

MOV c, AX

MOV BX, a

MUL BX

ADD x, AX

LOOP loop2

POP CX

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

MOV AX,x

CALL print

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

LOOP loop1

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

JMP start

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUT

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUT:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUT

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP