;========== Converting Upper to Lower and Lower to upper =========

ORG C0 ; Originate at position C0

DB 48 ; Put 'H' into position C0 (48 is hex for H)

DB 65 ; Put 'e' into position C0 (65 is hex for e)

DB 4C ; Put 'L' into position C0 (4C is hex for L)

DB 6C ; Put 'l' into position C0 (6C is hex for l)

DB 4F ; Put 'O' into position C0 (4F is hex for O)

DB 77 ; Put 'w' into position C0 (77 is hex for w)

DB 4F ; Put 'O' into position C0 (4F is hex for O)

DB 72 ; Put 'r' into position C0 (72 is hex for r)

DB 4C ; Put 'L' into position C0 (4C is hex for L)

DB 44 ; Put 'D' into position C0 (44 is hex for D)

ORG 00 ; Originate at position 00

MOV AL, A ; Move 9 into AL (9 is length of string)

MOV CL, C0 ; Move C0 into CL

LOOP:

MOV BL, [CL] ; Move positon CL into BL

CMP BL, 5B            ; if the answer is  jump to upper\_to\_lower

JS C\_UPPER\_TO\_LOWER   ; Jump to c\_upper\_to\_lower

JMP C\_LOWER\_TO\_UPPER   ; Jump to c\_lower\_to\_upper

C\_UPPER\_TO\_LOWER:

MOV DL, [CL] ; Move position CL onto DL

ADD DL, 20 ; ADD 20 for DL (Convert Upper case to Lower case)

MOV [CL], DL ; Move DL to position CL

INC CL ; Increment CL

DEC AL ; Decrement AL

JNZ LOOP ; Jump not while AL zero to Loop

JMP END ; Jump to END

C\_LOWER\_TO\_UPPER:

MOV DL, [CL] ; Move position CL onto DL

SUB DL, 20 ; Subtract 20 for DL (Convert Upper case to Lower case)

MOV [CL], DL ; Move DL to position CL

INC CL ; Increment CL

DEC AL ; Decrement AL

JNZ LOOP ; Jump not while AL zero to Loop

JMP END ; Jump to END

END:

END

; ======= IS PRIME =======

MOV AL, 1f

CHECK\_IS\_PRIME:

CMP AL, 0   ; Compare AL to number 0 (30 in hex)

JZ END       ; If true jump to end as number is prime

CMP AL, 1   ; Compare AL to number 1 (31 in hex)

JZ END       ; If true jump to end as number is prime

CMP AL, 2   ; Compare AL to number 2 (32 in hex)

JZ END       ; If true jump to end as number is prime

CMP AL, 3   ; Compare AL to number 3 (33 in hex)

JZ END       ; If true jump to end as number is prime

CMP AL, 5   ; Compare AL to number 5 (35 in hex)

JZ END       ; If true jump to end as number is prime

CMP AL, 7   ; Compare AL to number 7 (37 in hex)

JZ END       ; If true jump to end as number is prime

JMP MOD\_VALUES

MOD\_VALUES:

PUSH AL    ; Push AL

POP BL     ; Pop to BL (Taking the input and copying it to BL)

MOD BL, 2   ; Mod BL by 2

CMP BL, 0   ; Compare BL with 0

JZ CHANGE\_AL  ; If true jump to CHANGE\_AL

PUSH AL    ; Push AL

POP BL     ; Pop to BL (Taking the input and copying it to BL)

MOD BL, 3   ; Mod BL by 3

CMP BL, 0   ; Compare BL with 0

JZ CHANGE\_AL  ; If true jump to CHANGE\_AL

PUSH AL    ; Push AL

POP BL     ; Pop to BL (Taking the input and copying it to BL)

MOD BL, 5   ; Mod BL by 5

CMP BL, 0   ; Compare BL with 0

JZ CHANGE\_AL  ; If true jump to CHANGE\_AL

PUSH AL    ; Push AL

POP BL     ; Pop to BL (Taking the input and copying it to BL)

MOD BL, 7   ; Mod BL by 7

CMP BL, 0   ; Compare BL with 0

JZ CHANGE\_AL  ; If true jump to CHANGE\_AL

JMP END

CHANGE\_AL:

MOV AL, FF  ; Move FF into AL

JMP END

END:

END

;==================== Replace vowels with Z ======================

ORG C0

DB "AEIOUBCU"

ORG 0

MOV AL, 8    ; 8 is the length of the string

MOV BL, 5A   ; 5A is hex for 'Z'

MOV CL, C0   ; Move C0 in to register CL

LOOP:

MOV DL, [CL] ; Move memory location CL in to register DL

CMP DL, 41   ; Compare if DL is ascii 41 (aka 'A')

JZ A\_TO\_Z    ; If the comparison is TRUE we jump to other loop

CMP DL, 45   ; Compare if DL is ascii 45 (aka 'E')

JZ A\_TO\_Z    ; If the comparison is TRUE we jump to other loop

CMP DL, 49   ; Compare if DL is ascii 49 (aka 'I')

JZ A\_TO\_Z    ; If the comparison is TRUE we jump to other loop

CMP DL, 4F   ; Compare if DL is ascii 4F (aka 'O')

JZ A\_TO\_Z    ; If the comparison is TRUE we jump to other loop

CMP DL, 55   ; Compare if DL is ascii 55 (aka 'U')

JZ A\_TO\_Z    ; If the comparison is TRUE we jump to other loop

INC CL       ; Increment postion CL by 1

DEC AL       ; Bring AL down by 1 (incrementaion of the string)

JNZ LOOP     ; JUMP while not zero

JMP END       ; When it is zero jump to end

A\_TO\_Z:

MOV [CL], BL ; Move 'Z' into position CL

JMP LOOP

END:

END

;==================== Push vowels to stack ======================

ORG C0

DB "AHIGOURE"

ORG 0

MOV AL, 8    ; 8 is the length of the string

MOV CL, C0   ; Move C0 in to register CL

LOOP:

MOV DL, [CL] ; Move memory location CL in to register DL

INC CL

CMP DL, 41   ; Compare if DL is ascii 41 (aka 'A')

JZ PUSH\_VOWEL    ; If the comparison is TRUE we jump to other loop

CMP DL, 45   ; Compare if DL is ascii 45 (aka 'E')

JZ PUSH\_VOWEL    ; If the comparison is TRUE we jump to other loop

CMP DL, 49   ; Compare if DL is ascii 49 (aka 'I')

JZ PUSH\_VOWEL    ; If the comparison is TRUE we jump to other loop

CMP DL, 4F   ; Compare if DL is ascii 4F (aka 'O')

JZ PUSH\_VOWEL    ; If the comparison is TRUE we jump to other loop

CMP DL, 55   ; Compare if DL is ascii 55 (aka 'U')

JZ PUSH\_VOWEL    ; If the comparison is TRUE we jump to other loop

DEC AL       ; Bring AL down by 1 (incrementaion of the string)

JNZ LOOP     ; JUMP while not zero

JMP END       ; When it is zero jump to end

PUSH\_VOWEL:

PUSH DL ; Move 'Z' into position CL

DEC AL

JZ END

JMP LOOP

END:

END

;==================== Pop vowels from stack ======================

ORG C0

DB "AHIGOURE"

ORG 0

MOV AL, 8     ; 8 is the length of the string

MOV CL, C0    ; Move C0 in to register CL

LOOP:

MOV DL, [CL] ; Move memory location CL in to register DL

INC CL ; Increment CL

CMP DL, 41    ; Compare if DL is ascii 41 (aka 'A')

JZ PUSH\_VOWEL   ; If the comparison is TRUE we jump to other loop

CMP DL, 45    ; Compare if DL is ascii 45 (aka 'E')

JZ PUSH\_VOWEL   ; If the comparison is TRUE we jump to other loop

CMP DL, 49    ; Compare if DL is ascii 49 (aka 'I')

JZ PUSH\_VOWEL   ; If the comparison is TRUE we jump to other loop

CMP DL, 4F    ; Compare if DL is ascii 4F (aka 'O')

JZ PUSH\_VOWEL   ; If the comparison is TRUE we jump to other loop

CMP DL, 55    ; Compare if DL is ascii 55 (aka 'U')

JZ PUSH\_VOWEL   ; If the comparison is TRUE we jump to other loop

DEC AL        ; Bring AL down by 1 (incrementaion of the string)

JNZ LOOP      ; JUMP while not zero

JMP WHITE\_SPACE

PUSH\_VOWEL:

PUSH DL       ; Move 'Z' into position CL

DEC AL        ; Decrement AL

JZ WHITE\_SPACE  ; When zero jump to White space loop

JMP LOOP      ; Jump to loop

WHITE\_SPACE:

MOV CL, C0    ; Reset the position of the VDU

MOV AL, 08    ; Move 8 into AL (length of vdu string)

MOV BL, 20    ; Move white space (' ') into BL (20 = Ascii value)

JMP CLEAR\_VDU   ; Jmp to clear VDU

CLEAR\_VDU:

MOV [CL], BL ; Mov BL into position CL on VDU

INC CL        ; Increment CL

DEC AL        ; Decrement AL

JNZ CLEAR\_VDU   ; While AL is not zero jump to Clear\_VDU

MOV CL, C0    ; Move CL to position C0 (VDU)

MOV AL, 5     ; Move 5 into AL register (length of POP)

JMP POP\_VOWEL   ; Jmp to POP\_VOWEL

POP\_VOWEL:

POP BL        ; Pop to BL

MOV [CL], BL   ; Move BL to position CL (in the VDU)

INC CL        ; Increment CL

DEC AL        ; Decrement AL from 5 to 0

JNZ POP\_VOWEL   ; Jump while not Zero to POP\_VOWEL

JMP END      ; Jump to END

END:

END

; ========== Counting from 99 to 0 ========================

Table:

ORG 80

DB CE ; IS FOR 9

DB FE ; IS FOR 8

DB CA ; IS FOR 7

DB 7C ; IS FOR 6

DB DC ; IS FOR 5

DB 4E ; IS FOR 4

DB 9E ; IS FOR 3

DB B6 ; IS FOR 2

DB 0A ; IS FOR 1

DB FA ; IS FOR 0

ORG 90

DB CF ; IS FOR 9

DB FF ; IS FOR 8

DB CB ; IS FOR 7

DB 7D ; IS FOR 6

DB DD ; IS FOR 5

DB 4F ; IS FOR 4

DB 9F ; IS FOR 3

DB B7 ; IS FOR 2

DB 0B ; IS FOR 1

DB FB ; IS FOR 0

ORG 00 ; Originate 00

MOV CL, 80 ; Set CL to 80

RESET\_DL:

MOV DL, 90 ; Set DL to 90

CMP CL, 8A ; Compare CL to 8A

JNZ PUSH\_TO\_SSD ; If its not 8A jump to PUSH\_TO\_SSD

JMP END ; Else jump to END

INCREMENT\_CL:

INC CL ; Increment CL

JMP RESET\_DL ; Jump to RESET\_DL

PUSH\_TO\_SSD:

MOV AL, [CL] ; Move position CL to AL

OUT 02 ; Push out to SSD

MOV AL, [DL] ; Move position DL to AL

OUT 02 ; Push out to SSD

INC DL ; Increment DL

CMP DL,9A ; Compare DL to 9A

JNZ PUSH\_TO\_SSD ; If it is not 9A PUSH\_TO\_SSD

JMP INCREMENT\_CL ; Else jump to INCREMENT\_CL

END:

END

; ========== Counting from 0 to 99 ===============

TABLE:

ORG 80

DB CE ; IS FOR 9

DB FE ; IS FOR 8

DB CA ; IS FOR 7

DB 7C ; IS FOR 6

DB DC ; IS FOR 5

DB 4E ; IS FOR 4

DB 9E ; IS FOR 3

DB B6 ; IS FOR 2

DB 0A ; IS FOR 1

DB FA ; IS FOR 0

ORG 90

DB CF ; IS FOR 9

DB FF ; IS FOR 8

DB CB ; IS FOR 7

DB 7D ; IS FOR 6

DB DD ; IS FOR 5

DB 4F ; IS FOR 4

DB 9F ; IS FOR 3

DB B7 ; IS FOR 2

DB 0B ; IS FOR 1

DB FB ; IS FOR 0

ORG 00 ; Originate 00

MOV CL, 89 ; Set CL to 89

RESET\_DL:

MOV DL, 99 ; Set DL to 99

CMP CL, 7F ; Compare CL to 7F

JNZ PUSH\_TO\_SSD ; If its not 7F jump to PUSH\_TO\_SSD

JMP END ; Else jump to END

INCREMENT\_CL:

DEC CL ; Decrement CL

JMP RESET\_DL ; Jump to RESET\_DL

PUSH\_TO\_SSD:

MOV AL, [CL] ; Move position CL to AL

OUT 02 ; Push out to SSD

MOV AL, [DL] ; Move position DL to AL

OUT 02 ; Push out to SSD

DEC DL ; Decrement DL

CMP DL, 8F ; Compare DL to 8F

JNZ PUSH\_TO\_SSD ; If it is not 9A PUSH\_TO\_SSD

JMP INCREMENT\_CL ; Else jump to INCREMENT\_CL

END:

END

; ======= Calling Functions =======

MOV AL, C0

MOV BL, 4A

CALL 50

MOV AL, FF

CALL 50

HALT

ORG 50

MOV [AL], BL

RET

END

; ====== Counting from 99 to 0 with timer ==============

Setup:

JMP Main

DB 40

Main:

MOV AL, 01 ; Reset SSD

OUT 02

MOV AL, 00

OUT 02

MOV CL, 50

RESET\_DL:

MOV DL, 60 ; Move DL to position A0 (Right side)

CMP CL, 5A ; Cmp CL with position 9A

JNZ PUSH\_TO\_SSD ; IF CL is not at 9A jump to JUMPY

JMP END ; Else jump to END

INCREMENT\_CL: ; Incrementation Left side (Decrementation number)

INC CL ; Increment CL

JMP RESET\_DL ; Jump to JUMP

PUSH\_TO\_SSD:

MOV AL, [CL] ; Move CL/DL to register AL to output

OUT 02 ; Pushes to Seven Segment Display

MOV AL, [DL]

OUT 02

JMP INTR\_ENABLER

INCREMENT\_DL:

INC DL

CMP DL,6A ; CMP DL to end of numbers

JNZ PUSH\_TO\_SSD ; IF DL is not AA jump to JUMPY

JMP INCREMENT\_CL ; ELSE jump to INCY

INTR\_ENABLER: ; ENABLING INTERUPTS

STI ; Initiate timer

JMP LOOP

LOOP:

JMP LOOP ; Timer interrupt

ORG 40 ; Once finished jump to origin 30

POP BL ; Pop of the stack

CLI ; Close the interrupt timer

JMP INCREMENT\_DL

TABLE:

ORG 50

DB CE ; IS FOR 9

DB FE ; IS FOR 8

DB CA ; IS FOR 7

DB 7C ; IS FOR 6

DB DC ; IS FOR 5

DB 4E ; IS FOR 4

DB 9E ; IS FOR 3

DB B6 ; IS FOR 2

DB 0A ; IS FOR 1

DB FA ; IS FOR 0

ORG 60

DB CF ; IS FOR 9

DB FF ; IS FOR 8

DB CB ; IS FOR 7

DB 7D ; IS FOR 6

DB DD ; IS FOR 5

DB 4F ; IS FOR 4

DB 9F ; IS FOR 3

DB B7 ; IS FOR 2

DB 0B ; IS FOR 1

DB FB ; IS FOR 0

END:

END

; ======== Taking in a number and displaying ascii on vdu =======

Main:

IN 00 ; Take in an input

MOV CL, C0 ; Move C0 into CL (top left of VDU)

FirstDigit:

PUSH AL ; Push the input to the Stack

AND AL, F0 ; Split the Binary and take only the right 4 digits

DIV AL, 10

ADD AL, 30 ; Add 30 (Gives the number in the table)

MOV [CL], AL ; Move AL into position CL (Position in VDU)

INC CL ; Increment CL

POP AL ; Pop AL (resets AL to the original input)

SecondDigit:

AND AL, 0F ; Split the Binary and take only the left 4 digits

ADD AL, 30 ; Add 90 (Gives the number in the table)

MOV [CL], AL ; Move AL into position CL (Position in VDU)

End:

END

; ====== Counting from X to 0 using keyboard interrupt ========

Setup:

JMP Main ; Jump to main

DB 0F ; Timer interrupt set at 12

DB 20 ; Keyboard interrupt set at 20

Main:

MOV AL, 01 ; Reset the Left number in SSD

OUT 02 ; Out to SSD

MOV AL, 00 ; Reset the Right number in SSD

OUT 02 ; Out to SSD

STI ; Initiate interrupts

Loopo:

JMP Loopo ; Timer Interrupt

Number1:

ORG 20 ; Originate at 20

CLI ; Close Interrupts

IN 07 ; Take in keyboard interrupt

SUB AL, 30 ; Take 30 from AL

MUL AL, 0A ; Multiply AL by 10

PUSH AL ; Push AL

POP BL ; Pop to BL

MOV AL, 40 ; Move 40 into AL

MOV [03], AL ; Move AL into position 03

STI ; Initialise interrupts

IRET ; Return interrupt

Number2:

ORG 40 ; Originate at 40

CLI ; Close interrupts

IN 07 ; Take keyboard input

SUB AL, 30 ; Subtract 30

ADD BL, AL ; Add AL to BL

MOV AL, 60 ; Move 60 to AL

MOV [02], AL ; Move AL into position 02

PUSH BL ; Push BL

POP AL ; Pop to AL

STI ; Initiate interrupts

IRET ; Return

Count:

ORG 60 ; Originate at 60

CLI ; Close interrupts

PUSH AL ; Push AL to Stack

DIV AL, 0A ; Divide AL by 10

ADD AL, 90 ; Add 90 to AL

MOV AL, [AL] ; Move position AL into AL

SUB AL, 1 ; Subtract 1 from AL

OUT 02 ; Push to the SSD

POP AL ; Pop to AL

PUSH AL ; Push AL to stack

MOD AL, 0A ; Mod AL by 10

ADD AL, 90 ; Add 90 to AL

MOV AL, [AL] ; Move position AL to AL

OUT 02 ; Out to SSD

POP AL ; Pop AL

DEC AL ; Decrement AL

CMP AL, FF ; Check to see if AL is FF

JZ End ; If it is jump to END

STI ; Initiate interrupt

IRET ; Return

TableSetup:

ORG 90

DB FB ; 0

DB 0B ; 1

DB B7 ; 2

DB 9F ; 3

DB 4F ; 4

DB DD ; 5

DB 7D ; 6

DB CB ; 7

DB FF ; 8

DB CF ; 9

End:

END

; ===== Alternating 0 and 1 in VDU using timer =======

ORG 80 ; Originate at 80

DB 30 ; Add 30 (ascii for 0)

DB 31 ; Add 31 (ascii for 1)

ORG 00 ; Originate at 00

JMP MAINCODE ; Jmp to MAINCODE

DB 4A ; Store 4A into db

MAINCODE:

STI ; Initiate interrupts

MOV CL, C0 ; Move C0 into register CL (C0 is top left of vdu)

ADD\_0:

MOV BL, [80] ; Move BL to position 80 (start of the DBs)

MOV [CL], BL ; Move BL into positon CL (move bl into vdu)

CMP CL, FF ; Compare CL to FF, (if CL has reached the bottom right of the VDU)

JZ END ; If it is then jump to END

INC CL ; Else increment CL

JMP ENABLER ; Jump to Enabler (enabling the interrupt)

ADD\_1:

MOV DL, [81] ; Move postion 81 into DL

MOV [CL], DL ; Move DL into Position CL (Adds DL to the VDU

CMP CL, FF ; Compare CL to FF, (if CL has reached the bottom right of the VDU)

JZ END ; If it is then jump to END

INC CL ; Else increment CL

JMP ENABLER ; Jump to Enabler (enabling the interrupt)

ENABLER:

STI ; Allow intterrupts

JMP LOOP ; Jump to LOOP

LOOP:

INC AL ; Counts for 5 seconds by using an infinte loop

JMP LOOP

ORG 4A ; Jump to originate 4A

CLI ; Close intterrupts

POP AL ; Pop AL

PUSH CL ; Push CL to the stack

POP DL ; Pop to DL

MOD DL, 2 ; Check to see if the last number on the VDU was even or not

JZ ADD\_0 ; Jump to ADD\_0

JMP ADD\_1 ; Jump to ADD\_1

END:

END

; ======== Taking in a digit and displaying ascii on SSD ==========

Clear:

MOV AL, 00 ; Reset the Left side of SSD

OUT 02 ; Out to SSD

MOV AL, 01 ; Reset the Right side of SSD

OUT 02 ; Out to SSD

Main:

IN 00 ; Take in an input

CMP AL, 71

JZ End

CMP AL, 30

JS Main

CMP AL, 3A

JNS Main

FirstDigit:

PUSH AL ; Push the input to the Stack

AND AL, F0 ; Split the Binary and take only the right 4 digits

DIV AL, 10

ADD AL, 50 ; Add 90 (Gives the number in the table)

MOV AL, [AL] ; Move position AL to AL

SUB AL, 01 ; Subtract 1 from AL

OUT 02 ; Send to the SSD

POP AL ; Pop AL (resets AL to the original input)

SecondDigit:

AND AL, 0F ; Split the Binary and take only the left 4 digits

ADD AL, 50 ; Add 90 (Gives the number in the table)

MOV AL, [AL] ; Move position AL to AL

OUT 02 ; Send to the SSD

JMP Main

Table:

ORG 50

DB FB ; 0

DB 0B ; 1

DB B7 ; 2

DB 9F ; 3

DB 4F ; 4

DB DD ; 5

DB FD ; 6

DB CB ; 7

DB FF ; 8

DB CF ; 9

DB EE ; A

DB 7C ; B

DB F0 ; C

DB 3E ; D

DB F4 ; E

DB E4 ; F

End:

END

;---Procedure to Add all the odd numbers in table---

ORG 80

DB 03

DB 01

DB 04

DB 06

DB 09

DB 3F

DB 2A

ORG 50 ;IF ODD

MOV DL,0

MOV DL,[AL]

ADD CL,DL

INC AL

RET

ORG 60 ;IF EVEN

INC AL

RET ;its even so do nothing except move next number

ORG 00

MOV AL,80 ;start of table

MOV BL,86 ;end of table

START:

MOV DL,0

MOV DL,[AL] ;Setting DL to first value

MOD DL,2 ;Checking odd or even

JNZ ODD

JMP EVEN

ODD:

CALL 50

CMP AL,BL ;Checking is it finished check

JZ ENDY

JMP START

EVEN:

CALL 60

CMP AL,BL

JZ ENDY

JMP START

ENDY:

END