Assembly Programming Lab. (1)

HelloEverybody.asm

```
.MODEL SMALL
.STACK
.DATA
MESSAGE DB "HELLO EVERYBODY! I AM LEARNING ASSEMBLY LANGUAGE!","$"

.CODE

MAIN PROC
   MOV AX, @DATA
   MOV DS,AX

   MOV AH,09
   LEA DX,MESSAGE
   INT 21H

   MOV AX, 4C00H
   INT 21H
MAIN ENDP
END MAIN
```

AddTwoNumbers.asm

```
; Add two numbers and store the results into the third variable
TITLE       A04ASM1 (EXE)  Move and add operations

.MODEL SMALL

; -----------------------------------------------
.STACK
; -----------------------------------------------
.DATA
     FLDD              DW          215
     FLDE              DW          125
     FLDF              DW          ?
; -----------------------------------------------
.CODE
MAIN                   PROC
            MOV    AX,@DATA              ;Set address of data
            MOV    DS,AX                 ;  segment in DS
            MOV    AX, FLDD              ;Move 0215 to AX
            ADD    AX,FLDE               ;Add  0125 to AX
            MOV    FLDF,AX               ;Store sum in FLDF
            MOV   AX,4C00H               ;End processing
            INT   21H
MAIN                   ENDP                ;End of procedure
            END          MAIN              ;End of program
```


StringCopy1.asm

```
.MODEL SMALL
.STACK
.DATA
STRING1 DB "String Copy","$"
STRING2 DB '?'

.CODE
MAIN        PROC
    MOV AX, @DATA
    MOV DS, AX
    MOV ES, AX

    MOV CX, 13        ; Initialize to move 9 characters
    LEA SI, STRING1    ; Initialize source index register to offset of string 1
    LEA DI, STRING2    ; Initialize destination index register to offset of string 2

 BEGINLOOP:
    MOV AL,[SI]          ; Get a current character from string 1 to AL
    MOV [DI], AL         ; Move it to the current character in string 2
    INC SI               ; Move to the next character in string 1
    INC DI               ; Move to the next character in string 2
    DEC CX               ; Decrease the count for loop
    JNZ      BEGINLOOP     ; Continue to loop if count is not 0

    MOV AH, 09H
    LEA DX, STRING2
    int 21H         ; Display String 2
    .EXIT
MAIN                   ENDP         ;End of procedure
END   MAIN             ;End of program
```

StringCopy2.asm

```
.MODEL SMALL
.STACK
.DATA
.CODE
MAIN        PROC
    MOV AX, @DATA
    MOV DS, AX
    MOV ES, AX

    MOV CX, 12        ; Initialize to move 12 characters
    LEA SI, STRING1   ; Initialize source index register to offset of string 1
    LEA DI, STRING2   ; Initialize destination index register to offset of string 2

 BEGINLOOP:
    MOV AL,[SI]           ; Get a current character from string 1 to AL
    MOV [DI], AL          ; Move it to the current character in string 2
    INC SI                ; Move to the next character in string 1
    INC DI                ; Move to the next character in string 2
    DEC CX                ; Decrease the count for loop
    JNZ      BEGINLOOP    ; Continue to loop if count is not 0

    MOV AH, 09H
    LEA DX, STRING2
    int 21H          ; Display String 2
     .EXIT

STRING1 DB "String Copy","$"
STRING2 DB '?'

MAIN                 ENDP        ;End of procedure
END   MAIN           ;End of program
```


PrintChar.asm

https://en.wikipedia.org/wiki/MS-DOS_API

```
.MODEL SMALL
.STACK
.DATA
.CODE
MAIN  PROC
   MOV  DL,  'A' ;  or  MOV DL, 41h
   MOV  AH,  2
   INT  21H
   MOV  AH,  4CH
   INT  21H
 MAIN  ENDP
END
```

sum.asm

```
.MODEL SMALL
.STACK
.DATA
.CODE
MAIN   PROC
      MOV    CX, 1
      MOV    AX, 0

LOOP1:  ADD    AX, CX
      INC    CX
      CMP    CX, 10
      JBE    LOOP1
      MOV    SUM, AX
      MOV    AH, 4CH
      INT    21H

SUM   DW     ?

MAIN  ENDP
END
```


HelloWorld1.asm

```
.model small ;
.stack
.data
.code
main PROC ;
   MOV AH, 02h ;

   MOV DL, 48h ; 'H' (0x48)
   INT 21h

   MOV DL, 65h ; 'e' (0x65)
   INT 21h

   MOV DL, 6Ch ; 'l' (0x6C)
   INT 21h

   MOV DL, 6Ch
   INT 21h

   MOV DL, 6Fh ; 'o' (0x6F)
   INT 21h

   MOV DL, 2Ch ; ',' (0x2C)
   INT 21h

   MOV DL, 20h ; ' ' (0x20)
   INT 21h

   MOV DL, 57h ; 'W' (0x57)
   INT 21h

   MOV DL, 6Fh ; 'o' (0x6F)
   INT 21h

   MOV DL, 72h ; 'r' (0x72)
   INT 21h

   MOV DL, 6Ch ; 'l' (0x6C)
   INT 21h

   MOV DL, 64h ; 'd' (0x64)
   INT 21h

   MOV DL, 21h ; '!' (0x21)
   INT 21h

   MOV AH, 4Ch
   INT 21h

main ENDP
END main
```

Exercise. Revise this code with JMP instruction.

HelloWorld2.asm

```
 .model small
.stack
.data
   message DB "Hello, World!", 0
.code
main proc
   mov ax, @data
   mov ds, ax
   mov es, ax

   mov cx, 0

   main_loop:
   mov ax, offset message
   mov si, ax
   add si, cx
   add cx, 1

   mov ah, 02h
   mov dl, [si]
   int 21h

   cmp dl, 0
   jne main_loop

   mov ah, 4ch
   int 21h
main endp
end main
```


HelloWorld3.asm

```
.model small
.stack
.data
   message DB "Hello, World!", 0

.code
main proc
   mov ax, @data
   mov ds, ax
   mov es, ax

   mov si, offset message
   dec si
   main_loop:
   inc si

   mov ah, 02h
   mov dl, [si]
   int 21h

   cmp dl, 0
   jne main_loop

   mov ah, 4ch
   int 21h
main endp
end main
```

HelloWorld4.asm

```
.model small
.stack
.data
   message DB “Hello, World! $”

.code
main proc
   mov ax, @data
   mov ds, ax
   mov es, ax

   main_loop:
   mov ah, 09h
   mov dx, offset message
   int 21h

   mov ah, 4ch
   int 21h
main endp
end main
```


comparison1.asm

```
.model small
.stack
.data
.code
main PROC
   MOV AH, 99
   MOV AL, 1
   CMP AH, AL

   JZ JMP_EQUALS   ; If ZF Flag=1
   JNZ JMP_NEQUALS ; if ZF Flag= 0

   JMP_EQUALS:
   MOV AH, 02h
   MOV DL, 3Dh     ; '='
   INT 21h
   INT 21h
   JMP JMP_EXIT

   JMP_NEQUALS:
   MOV AH, 02h
   MOV DL, 21h     ;  '!'
   INT 21h
   MOV DL, 3Dh     ; '='
   INT 21h
   JMP JMP_EXIT

   JMP_EXIT:
   MOV AH, 4Ch
   INT 21h
main ENDP
end main
```

```
.model small
.stack
.data
.code
main PROC
   MOV AH, 99
   MOV AL, 1
   CMP AH, AL

   JG JMP_GREATER
   JNG JMP_NGREATER  ; (= JLE)

   JMP_GREATER:
   MOV AH, 02h
   MOV DL, 3Eh    ; '>'
   INT 21h
   JMP JMP_EXIT

   JMP_NGREATER:
   MOV AH, 02h
   MOV DL, 3Ch    ; '<'
   INT 21h
   MOV DL, 3Dh    ; '='
   INT 21h
   JMP JMP_EXIT

   JMP_EXIT:
   MOV AH, 4Ch
   INT 21h
main ENDP
end main
```