Helloworld(emulate+run)

```
org 100h
.DATA
MSG DB 'HELLO WORLD!$'
.CODE
MAIN PROC
MOV AX,@DATA
MOV DS,AX
LEA DX,MSG
MOV AH,9
INT 21H
MOV AH,4CH
INT 21H
MAIN ENDP
END MAIN
```

Print Your Name(emulate+run)

```
org 100h
.DATA
MSG DB 'Mary!$'
.CODE
MAIN PROC
MOV AX,@DATA
MOV DS,AX
LEA DX,MSG
MOV AH,9
INT 21H
MOV AH,4CH
INT 21H
MAIN ENDP
END MAIN
```

Addition of 8-bit(emulate+singlestep-4)

```
org 100h
mov al, 5
mov bl, 2
jmp calc
back: jmp stop
calc:
add al, bl
jmp back
stop:
ret
```

Add16-bit(emulate+singlestep-4)

```
org 100h
mov AX, 0FE1CH
mov BX, 0A243H
jmp calc
back: jmp stop
calc:
add AX, BX
jmp back
stop:
ret
```

sub

8-bit(emulate+singlestep-6)

```
org 100h
mov al, 50h
mov bl, 20h
jmp calc
back: jmp stop
calc:
sub al, bl
jmp back
stop:
ret
```

sub16-bit(emu+view+memory+1000: 16,1001:23,1001:43,1002:32+s.s-4)

```
MOV AX,[1000h]
MOV BX,[1002h]
MOV CL,00h
SUB AX,BX
MOV [1004h],AX
JNC jump
INC CL
jump:
MOV [1006h],CL
HLT
```

Mul 8bit(emu+s.s-4)

```
org 100h
mov al, 50h
mov bl, 20h
jmp calc
back: jmp stop
calc:
MUL bl
jmp back
stop:
ret
```

Mul 16-bit(emu+s.s-6)

```
org 100h
mov AX, 0FE1CH
mov BX, 0A243H
jmp calc
back: jmp stop
calc:
MUL BX
jmp back
stop:
ret
```

div 8-bit(emu+s.s-9)

```
org 100h
mov al, 5h
mov bl, 2h
jmp calc
back: jmp stop
calc:
div bl
jmp back
stop:
ret
```

div16bit(emu+s.s-6)

```
org 100h
mov AX, 0FE1CH
mov BX, 0A243H
jmp calc
back: jmp stop
calc:
div BX
jmp back
stop:
ret
```

print a single character on screen(emul+run)

```
org 100h
Main Proc
Mov dl,'A'
Mov ah, 2
INT 21h
Mov ah, 4ch
INT 21h
Main endp
End Main
Ret
```

uppercase to lowercase(emul+run)

```
org 100h
main proc
mov ah, 1
int 21h
mov dl, al
add dl, 32
mov ah, 2
int 21h
mov ah, 4ch
int 21h
main endp
end main
ret
```

print multiple characters on screen(emu+run)

```
org 100h
message db 'WELCOME!$'
main proc
lea dx, [message]
mov ah, 09h
int 21h
mov ah, 4Ch
int 21h
main endp
end main
```

lower-case letter to uppercase(emu+run)

```
main proc
mov ah, 1
int 21h
mov dl, al
sub dl, 32
mov ah, 2
int 21h
mov ah, 4ch
int 21h
main endp
end main
ret
```

take a single-digit integer from the user and print it on the screen.(emu+run)

```
org 100h
.data
msg db 'Enter a
single-digit integer: $'
newline db 0Dh,
0Ah, '$'
result db '?', '$'
.code
main proc
mov ax, @data
mov ds, ax
lea dx, msg
int 21h
mov ah, 01h
int 21h
mov [result], al
lea dx, newline
mov ah, 09h
int 21h
lea dx, result
mov ah, 09h
int 21h
mov ah, 4Ch
int 21h
main endp
end main
```

**display a two-digit number on thescreen.
The two digits number is required to be taken in the program itself.(emul+run)**

```
org 100h
num dw 56
main proc
mov ax, @data
mov ds, ax
mov ax, num
mov cx, 10
div cx
add al, 30h
mov dl, al
mov ah, 02h
int 21h
add ah, 30h
mov dl, ah
mov ah, 02h
int 21h
mov ah, 4Ch
int 21h
main endp
end main
```

print the numbers from 0 to 9.(emu+run)

```
.MODEL SMALL
.STACK 100
.CODE
MOV CX,10
MOV DX,48
L1:
MOV AH,2
INT 21H
INC DX
LOOP L1
MOV AH,4CH
INT 21H
END
```

take two single-digit integers from the user and print the result of addition on the screen.(emul+run)

```
.model small
.stack 100h
.data
msg1 db 'Enter first digit: $'
msg2 db 'Enter second digit: $'
msg3 db 'Result: $'
newline db 0Dh, 0Ah, '$'
.code
main proc
```

```
mov ax, @data
mov ds, ax
mov ah, 09h
lea dx, msg1
int 21h
mov ah, 01h
int 21h
sub al, 30h
mov bl, al
lea dx, newline
mov ah, 09h
int 21h
mov ah, 09h
lea dx, msg2
int 21h
mov ah, 01h
int 21h
sub al, 30h
add bl, al
lea dx, newline
mov ah, 09h
int 21h
add bl, 30h
lea dx, newline
mov ah, 09h
int 21h
mov ah, 09h
lea dx, msg3
int 21h
mov dl, bl
mov ah, 02h
int 21h
mov ah, 09h
lea dx, newline
int 21h
mov ah, 4Ch
int 21h
main endp
end main
```

to take two single-digit numbers as input and display whether they are equal or not.(emul+run)

```
.model small
.stack 100h
.data
msg1 db 'Enter first digit: $'
msg2 db 'Enter second digit: $'
equal_msg db 'Numbers are equal.$'
not_equal_msg db 'Numbers are not equal.$'
newline db 0Dh, 0Ah, '$'
.code
main proc
mov ax, @data
mov ds, ax
mov ah, 09h
lea dx, msg1
int 21h
mov ah, 01h
```

```
int 21h
sub al, 30h
mov bl, al
lea dx, newline
mov ah, 09h
int 21h
mov ah, 09h
lea dx, msg2
int 21h
mov ah, 01h
int 21h
sub al, 30h
mov bh, al
lea dx, newline
mov ah, 09h
int 21h
cmp bl, bh
je equal
lea dx, newline
mov ah, 09h
int 21h
mov ah, 09h
lea dx, not_equal_msg
int 21h
jmp done
equal:
mov ah, 09h
lea dx, equal_msg
int 21h
done:
mov ah, 09h
lea dx, newline
int 21h
mov ah, 4Ch
int 21h
main endp
end main
```

whether a single-digit
number is odd or even(emul+run)

```
.model small
.stack 100h
.data
msg1 db 'Enter a single-digit number: $'
even_msg db 'The number is even.$'
odd_msg db 'The number is odd.$'
newline db 0Dh, 0Ah, '$'
.code
main proc
mov ax, @data
mov ds, ax
mov ah, 09h
lea dx, msg1
int 21h
mov ah, 01h
int 21h
sub al, 30h
lea dx, newline
mov ah, 09h
int 21h
```

```
mov ah, 0
mov bl, 2
div bl
cmp ah, 0
je even
mov ah, 09h
lea dx, odd_msg
int 21h
jmp done
even:
mov ah, 09h
lea dx, even_msg
int 21h
done:
mov ah, 09h
lea dx, newline
int 21h
mov ah, 4Ch
int 21h
main endp
end main
```

print the characters from A to Z inreverse order.(emul+run)

```
.MODEL SMALL
.STACK 100h
.DATA
msg DB 'Z', 0
.CODE
main:
MOV AX, @data
MOV DS, AX
MOV CX, 26
MOV DL, 'Z'
print_loop:
MOV AH, 2
INT 21h
DEC DL
LOOP print_loop
MOV AX, 4C00h
INT 21h
END main
```

print the numbers from 0 to 9 in reverse order.(emul+run)

```
.MODEL SMALL
.STACK 100h
.DATA
msg DB '0', 0
.CODE
main:
MOV AX, @data
MOV DS, AX
MOV CX, 10
MOV DL, '9'
print_loop:
MOV AH, 2
INT 21h
DEC DL
LOOP print_loop
MOV AX, 4C00h
INT 21h
END main
```

print the characters from A to Z.(emul+run)

```
.MODEL SMALL
.STACK 100h
.DATA
msg DB 'A', 0
.CODE
main:
MOV AX, @data
MOV DS, AX
MOV CX, 26
MOV DL, 'A'
print_loop:
MOV AH, 2
INT 21h
INC DL
LOOP print_loop
MOV AX, 4C00h
INT 21h
END main
```

take two single-digit integers from the user and print the result of subtraction on the screen.

(emu+run)

```
.model small
.stack 100h
.data
msg1 db 'Enter first digit: $'
msg2 db 'Enter second digit: $'
msg3 db 'Result: $'
newline db 0Dh, 0Ah, '$'
neg_msg db '-', '$'
.code
main proc
mov ax, @data
mov ds, ax
mov ah, 09h
lea dx, msg1
int 21h
mov ah, 01h
int 21h
sub al, 30h
mov bl, al
lea dx, newline
mov ah, 09h
int 21h
mov ah, 09h
lea dx, msg2
```

```
int 21h
mov ah, 01h
int 21h
sub al, 30h
mov bh, al
lea dx, newline
mov ah, 09h
int 21h
sub bl, bh
cmp bl, 0
jge positive_result
mov ah, 09h
lea dx, neg_msg
int 21h
neg bl
positive_result:
add bl, 30h
lea dx, newline
mov ah, 09h
int 21h
mov ah, 09h
lea dx, msg3
int 21h
mov dl, bl
mov ah, 02h
int 21h
mov ah, 09h
lea dx, newline
int 21h
mov ah, 4Ch
int 21h
main endp
end main
```