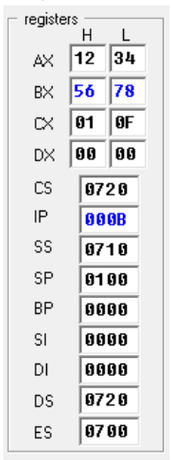
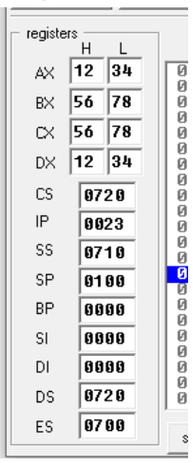
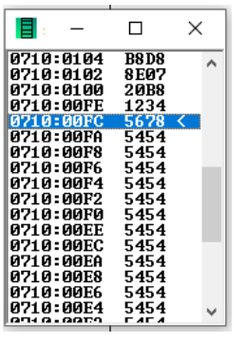
Dave Jhared G. Paduada
BSCpE III - GF

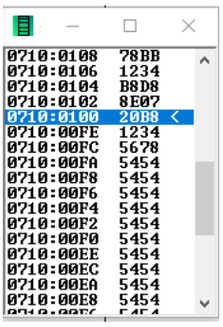
March 25, 2025

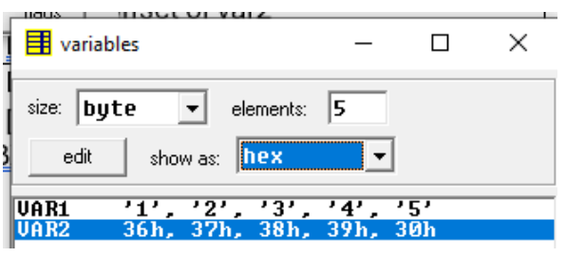
<table>
<tr><th colspan="3">Data Movement Instructions</th></tr>
<tr><th>Code</th><th>Instruction</th><th>Screenshot</th></tr>
<tr><td>.MODEL SMALL
.STACK 100H
.DATA
.CODE
MOV AX, @DATA
MOV DS, AX
mov ax,1234h
mov bx,5678h
push ax
push bx
pop cx
pop dx</td><td>Push and Pop</td><td>Register(Before and After)
Stack(After PUSH and After POP)

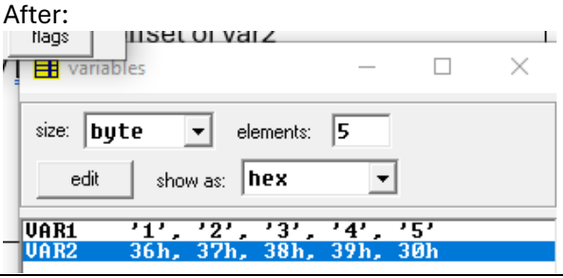
Register Before:
 

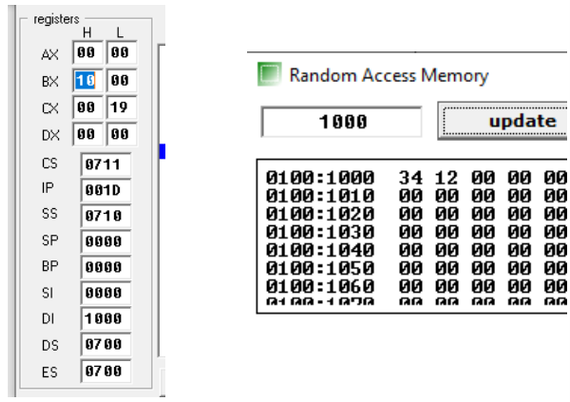
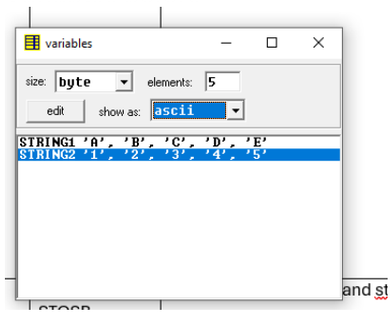
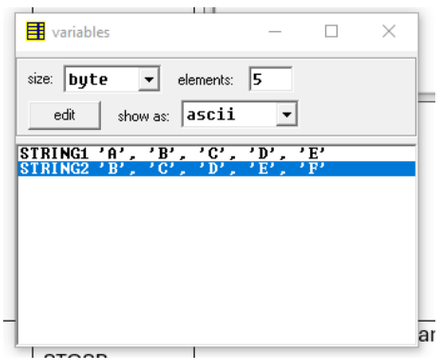
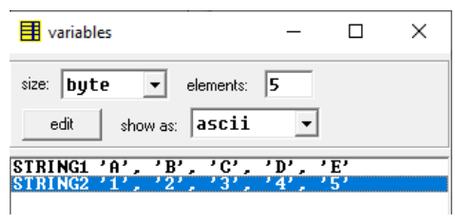
Register After:
 

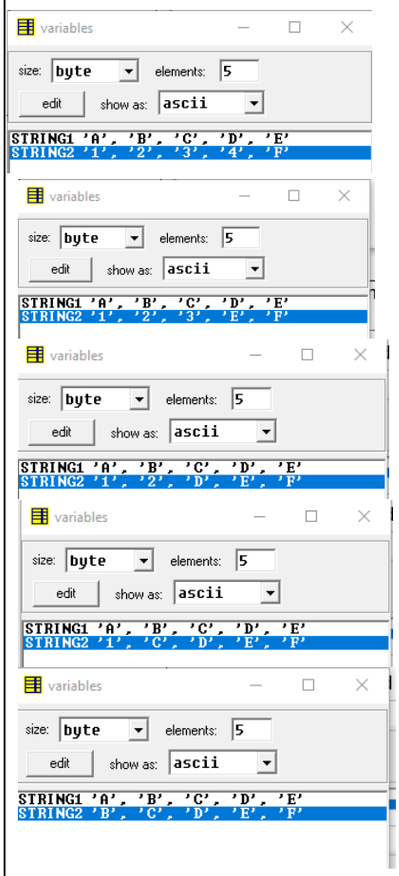
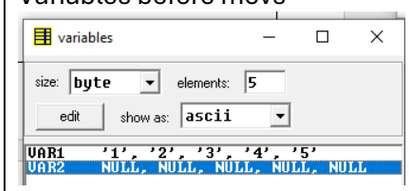
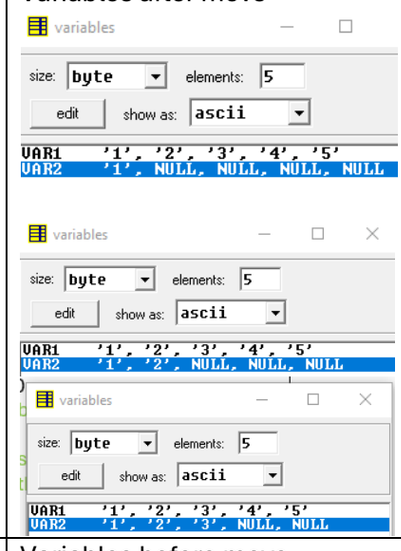
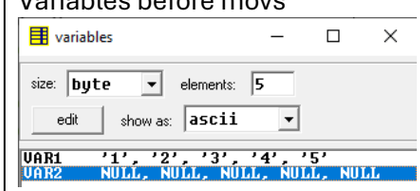
After Push:
 

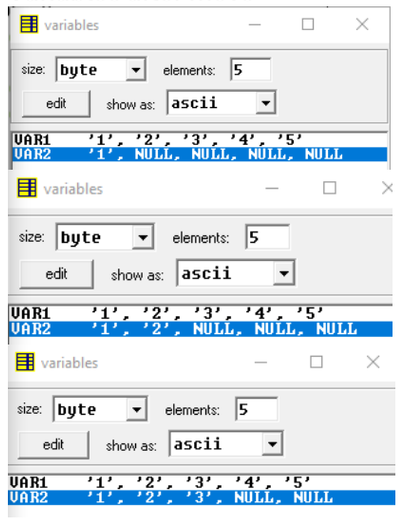
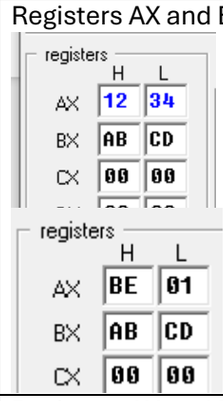
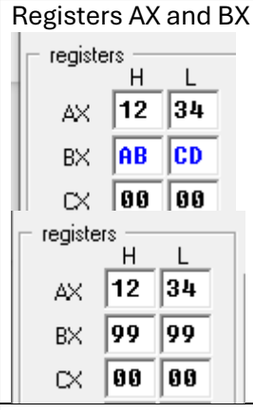
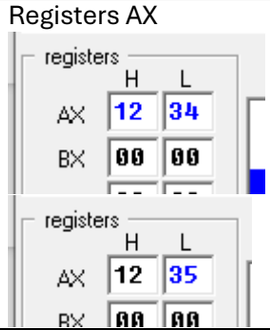
After Pop:
 
</td></tr>
<tr><td>.model small
.data
var1 db '12345',13,10,'$'
var2 db '67890',13,10,'$'
.code
MOV AX, @DATA
MOV DS, AX ; init DS
LEA SI, var1 ; offset of var1
LEA DI, var2; offset of var2
MOV DI,OFFSET VAR2</td><td>LEA</td><td>Register SI and DI

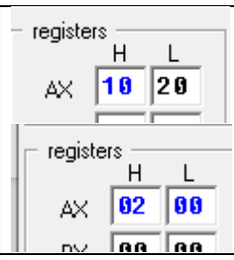
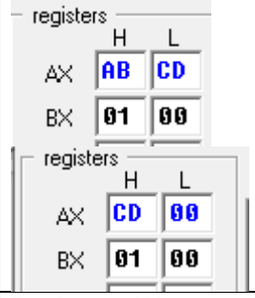
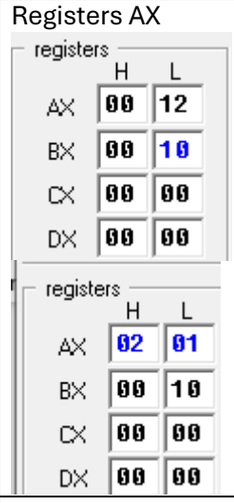
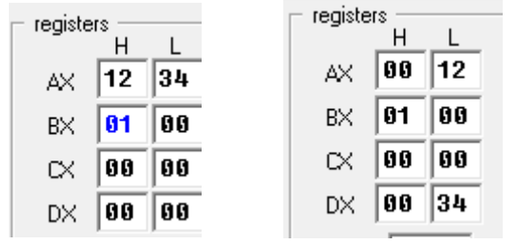
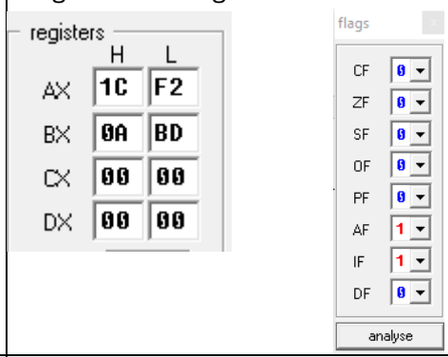
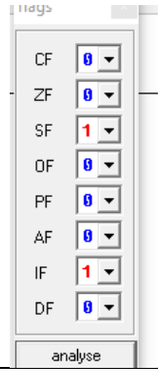
Variable var1 and var2 (Before and After)
Before:
 

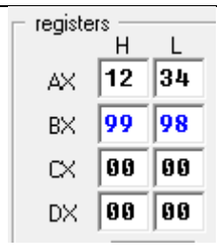
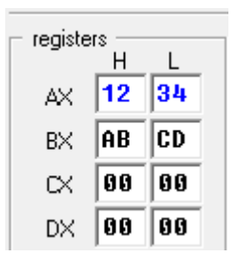
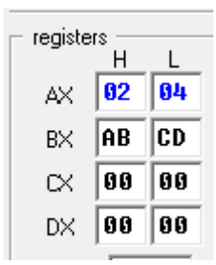
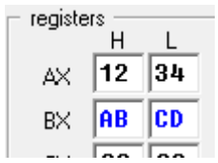
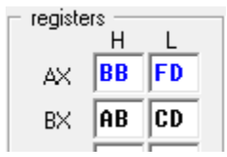
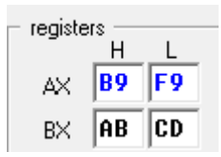
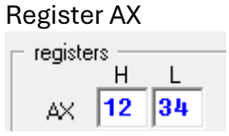
After:
 
</td></tr>
</table>

MOV DI,1000H MOV [DI],1234H LEA BX,[DI]		Register BX Memory pointed by DI 
.model small .data string1 db 'ABCDE',13,10,'$' string2 db '12345',13,10,'$' .code MOV AX, @DATA MOV ES, AX ; initialize ES MOV DS, AX LEA SI, string1 LEA DI, string2 CLD ; read from location 0 mov ax,0000h mov cx,5 ;counter ulit: lodsb ;AL = DS:[SI] inc al stosb ;ES:[DI] = AL loop ulit	LODSB and STOSB W/ CLD	Variables before  Variables after 
.model small .data string1 db 'ABCDE' string2 db '12345' .code MOV AX, @DATA MOV ES, AX ; initialize ES MOV DS, AX LEA SI, string1+04h LEA DI, string2+04h STD ; Read string from end to start mov ax,0000h mov cx,5 ulit: lodsb ;AL = DS:[SI] inc al Increase ASCII value (e.g., 'E' → 'F') stosb ;ES:[DI] = AL loop ulit	LODSB and STOSB W/ STD	Variables (before lods and stos) 

		Variables (after lods and stos) 
.model small .data var1 db '12345' var2 db 3 dup(0) .code MOV AX, @DATA MOV DS, AX ; init DS MOV ES, AX ; init ES LEA SI, var1 ; source LEA DI, var2 ; destination CLD ; DF = 0 MOVSB ; Move byte from DS:[SI] to ES:[DI], SI++, DI++ MOVSB ; Move second byte MOVSB ; Move third byte	MOVSB	Variables before movs  Variables after movs 
.model small .data var1 db '12345' var2 db 3 dup(0) .code MOV AX, @DATA MOV DS, AX ; init DS MOV ES, AX ; init ES LEA SI, var1 ; source LEA DI, var2 ; destination	MOVSB /W REP	Variables before movs 

CLD ; DF = 0 mov cx,3 ; SET COUNTER FOR REP rep MOVSB		Variables after movs 
Code	Instruction	Screenshot
;addition mov ax,1234h mov bx,0ABCDh add ax,bx		Registers AX and BX 
;subtraction mov ax,1234h mov bx,0ABCDH sub bx,ax		Registers AX and BX 
;increment mov ax,1234h inc ax		Registers AX 
;decrement mov ax,1234h dec ax		Registers AX 
;multiply 8-bit mov al,20h mov ah,10h mul ah		Registers AX

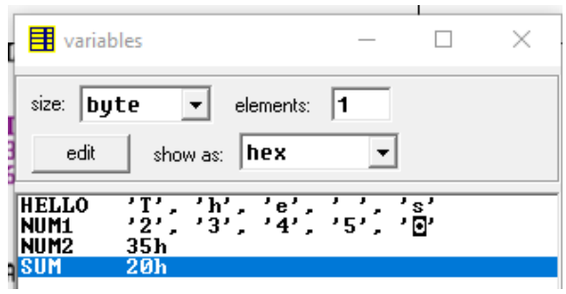
		
``;multiply 16-bit mov ax,0ABCDh mov bx,100h mul bx```		Registers AX and DX 
``;division 8-bit mov al,12h mov bl,10h div bl```		Registers AX 
``;division 16-bit mov ax,1234h mov bx,100h div bx```	1234/100 = 12.34	Registers AX and DX 
``;add w/ carry stc mov ax,1234h mov bx,0ABCDH adc ax,bx```		Register and Flag 
``;sub w/ borrow stc mov ax,1234h mov bx,0ABCDH sbb bx,ax```		Register and Flag 

		
;and mov ax,1234h mov bx,0ABCDH and ax,bx		Register AX and BX  
;or mov ax,1234h mov bx,0ABCDH or ax,bx		Register AX and BX  
;xor mov ax,1234h mov bx,0ABCDH xor ax,bx		Register AX and BX  
;not mov ax,1234h not ax		Register AX  
;neg mov ax,1234h neg ax		Register AX  

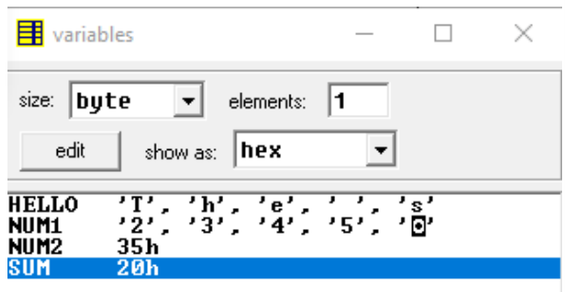
```
.MODEL SMALL
.DATA

hello db 'The sum of the numbers
num1 db '2345',10,13,'$'
num2 db '5678',10,13,'$'
sum db '     ',10,13,'$'

.CODE
mov ax, @DATA
mov ds, ax

mov si, 03h
mov cx, 05h
clc

add_loop:
    mov al, [num1+si]
    adc al, [num2+si]
    aaa
    pushf
    or al, 30h
    popf
    mov [sum+si], al
    dec si
    loop add_loop

mov ah, 09
mov dx, offset hello
int 21h

mov ah, 09
mov dx, offset sum
int 21h
```

Variables after each loop

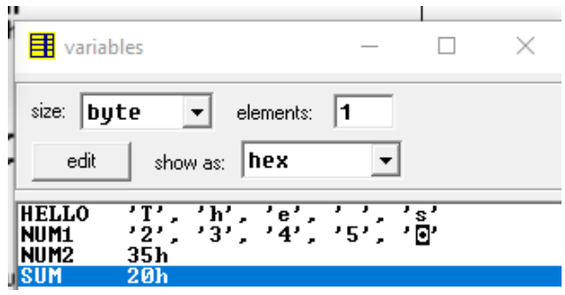
CX 4



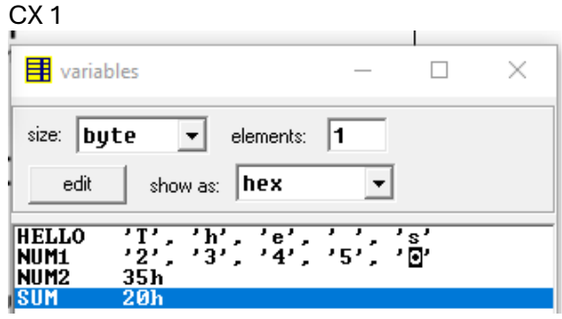
CX 3



CX 2



CX 1



CX 0