COAL MID I SOLUTION

BASIL ALI KHAN
20K-0477.

Question #01:

(i)

DIRECT OFSET OPERAND:

Direct offset operand is used to add displacement to the name of variable. Through this you can access memory location that may not have explicity labels.

Example:

```
arrayB  BYTE  10h, 20h, 30h, 40h
.code
  mov al, arrayB      ; AL = 10h.
  mov al, [arrayB+1]  ; AL = 20h.
  mov al, [arrayB+2]  ; AL = 30h.
```

INDEXED OPERAND:

An Indexed operand add a constant to a register to generate an affective address. Any 32-Bit general purpose registers can be used as indexed registers.

Notation:

```
  constant [reg].
  [constat + reg]
```

Example:

```
arrayB BYTE 10h, 20h, 30h
mov esi, 0
mov al, [arrayB + esi]   ; AL = 10h.
```

DATE

DAY M T W T F S S

(ii)

MOVSX INSTRUCTION :

MOVSX instruction copies the contenst of source operand into destination operand and sign extends the value to 16 or 32 bit. This instruction is used with signed integers.

Notation :

```
MOVSX  reg32 , reg/mem8
MOVSX  reg32 , reg/mem16
MOVSX  reg16 , reg/mem8
```

Example :

```
.data
byteval BYTE 10001111b.
.code
movsx ax, byteval ; AL = 1111111110001111 b
```

(iii)

SIGNED / UNSIGNED INTEGERS :

- All CPU instruction operate exactly same on signed and unsigned integers.
- It cannot distinguish between signed and unsigned integers.
- The programmer are solely responsible for using correct data type with each instruction.

Page No.

(iv)

LABLELS :

A label is an identifier acts as a placemarker for instruction and data. A label placed before an instruction implies instruction's address and a label placed before a variable implies variable's address.

Data Label :

A data label identifies the location of variable provides a way to reference variable in program.

Example:

```
count DWORD 100
```

Code labels:

A label in code area must end with a semi colon character. Code labels are used a targets of Jumping and looping instructions.

Example:

```
target:
    mov ax, 2
    :
    Jmp target
```

(v)

LOOP INSTRUCTION:

- The Loop instruction creates a counting loop.
- Syntax
  Loop target

- Logic:

```
ECX ← ECX - 1
If ECX != 0 , Jump to target.
```

(vi)

VIRTUAL MACHINE CONCEPT:

An effective way to explain how computer hardware and software are related to each other is called virtual machine concept.

Level 4.	High level language.
Level 3	Assembly language
Level 2	Instruction set Architecture (ISA)
Level 1	Digital Logic.

(vii)

LAHF INSTRUCTIONS:

The LAHF instruction copies the low byte of the EFLAGS register into AH. Following flags are copied: Sign, Zero, Auxiliary, carry, Parity and carry.

Example:

```
.data
saveFlags BYTE ?
.code
lahf                    ; load flags to AH.
mov saveFlags, ah.      ; saves them in variable.
```

SAHF INSTRUCTION:

The SAHF instruction copies AH into the low byte of EFlags register.

Example:

```
mov ah, saveFlags   ; load saveflags into AH
sahf                ; copy into flags register.
```

(viii)

REAL ADDRESS MODE:

- Only 1MB of memory can be Accessed.
- Program can access any part of main memory.
- MS DOS runs in real Address mode.

PROTECTED MODE:

- Each program can access address a maximum of 4GB of memory.
- The operating system assigns memory to each running program.
- Programs are preventing from accessing each other's memory.
- Native mode used. by Windows NT, 2000, XP & LINUX

QUESTION # 02

(i)

Address	Value.
32204	42h
32205	E3h
32206	0Eh
32207	00
32208	0Eh
32209	00
3220A	0E
3220B	00.
3220C	0E
3220D.	00

(ii)

```
mov  eax, DWORD PTR X1      ; a) EaX = 00EE342h.
mov  bl,  SIZEOF  X1        ; b) BL = 0Ah
mov  esi,  4
mov  BX , [x1 +ESI]         ; c) BX = 000Eh
```

(iii)

```
mov  AX, 7FF0H
Add  AL, 10H     ; a) CF = 1  SF = 0  ZF = 1  OF= 0

Add AH, 1        ; a) CF = 0  SF = 1  ZF = 0  OF = 1
```

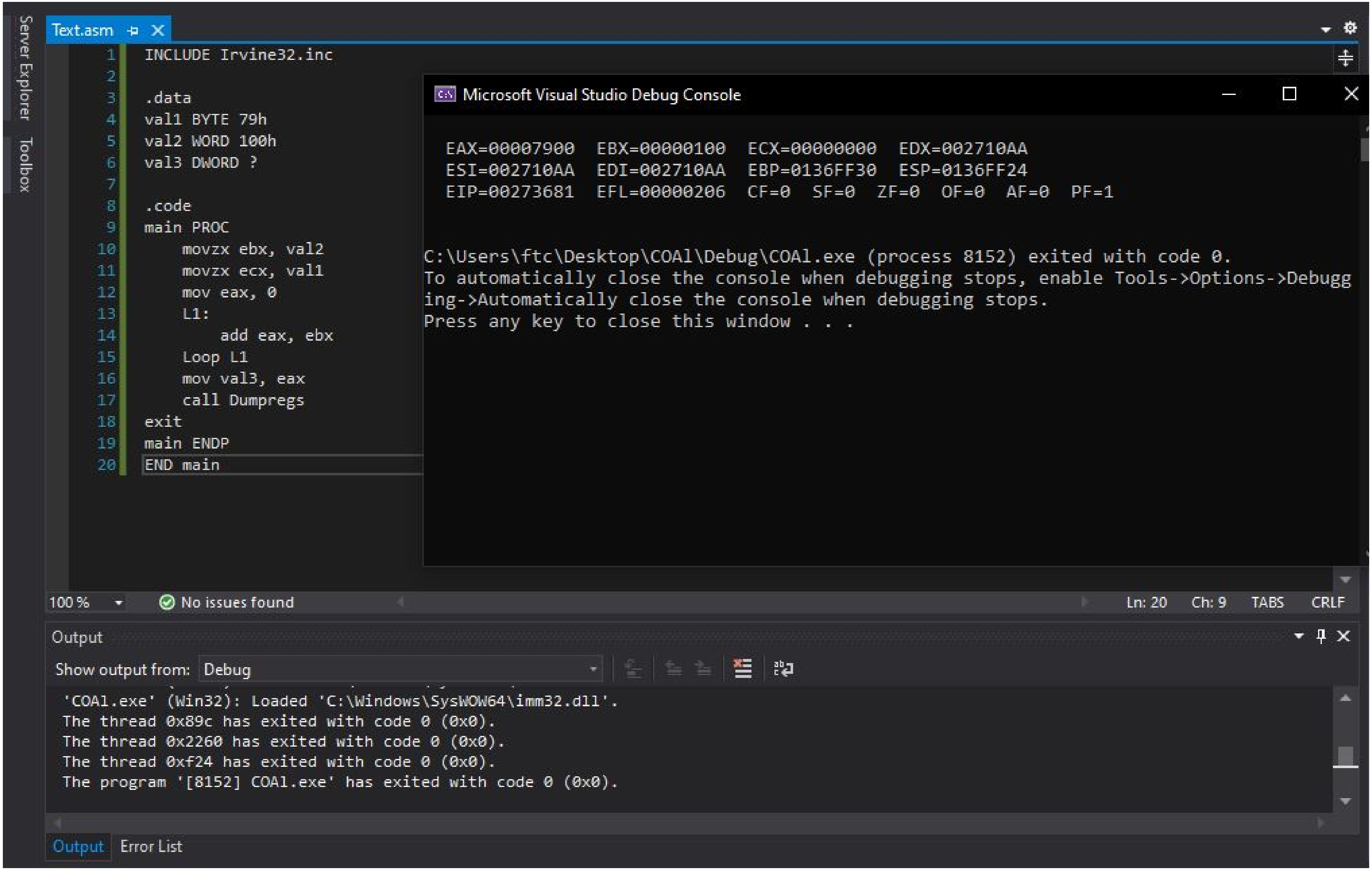
QUESTION # 03:

(i)

```
INCLUDE Irvine32.inc.
.data
 val1  BYTE  79h.
 val2  WORD  100h.
 val3  DWORD  ?
.code
main PROC
       movzx  ebx, val2
       movzx  ecx, val 1
       mov    eax, 0.
       L1 :
           Add  eax, ebx.
       Loop  L1
       mov   val3, eax
       call  Dumpregs.
elit
main ENDP
END main.
```

(ii)

```
INCLUDE Irvine32.inc.
.code
main PROC
    mov ebx, 2
    mov edx, 1
    mov eax, 2
    call Writedec.
    mov ecx, 10-1
    L1:
      mov al, ','
      call Writechar.
      mov eax, edx
      call Writedec.
      add eax, ebx
      mov ebx, edx
      mov edx, eax
    Loop L1
.exit
main ENDP
END main.
```

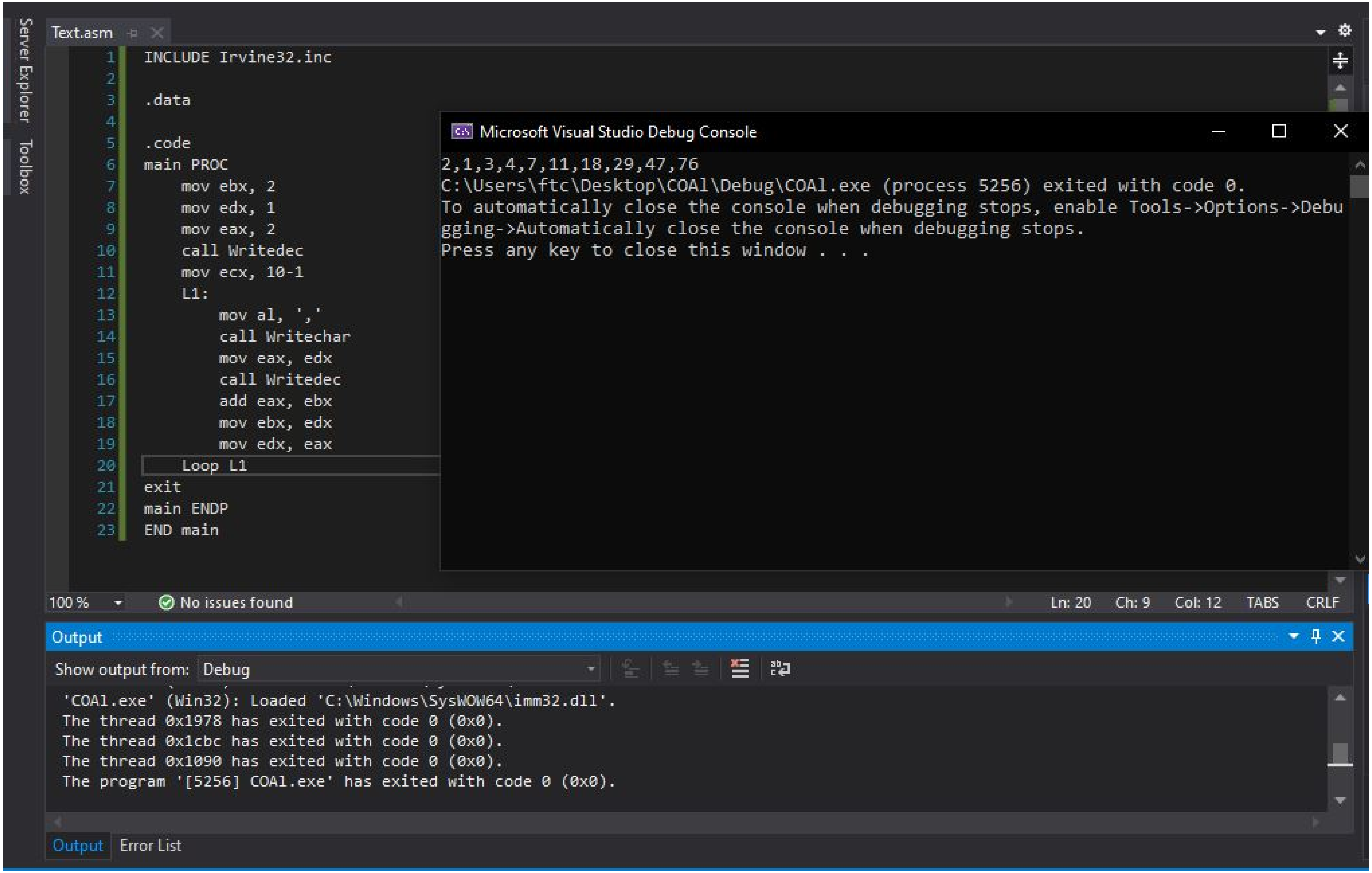


```
INCLUDE Irvine32.inc

.data
val1 BYTE 79h
val2 WORD 100h
val3 DWORD ?

.code
main PROC
    movzx ebx, val2
    movzx ecx, val1
    mov eax, 0
    L1:
        add eax, ebx
    Loop L1
    mov val3, eax
    call Dumpregs
exit
main ENDP
END main
```

```
  EAX=00007900  EBX=00000100  ECX=00000000  EDX=002710AA
  ESI=002710AA  EDI=002710AA  EBP=0136FF30  ESP=0136FF24
  EIP=00273681  EFL=00000206  CF=0  SF=0  ZF=0  OF=0  AF=0  PF=1


C:\Users\ftc\Desktop\COAl\Debug\COAl.exe (process 8152) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugg
ing->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Show output from: Debug

```
'COAl.exe' (Win32): Loaded 'C:\Windows\SysWOW64\imm32.dll'.
The thread 0x89c has exited with code 0 (0x0).
The thread 0x2260 has exited with code 0 (0x0).
The thread 0xf24 has exited with code 0 (0x0).
The program '[8152] COAl.exe' has exited with code 0 (0x0).
```



```
INCLUDE Irvine32.inc

.data

.code
main PROC
    mov ebx, 2
    mov edx, 1
    mov eax, 2
    call Writedec
    mov ecx, 10-1
    L1:
        mov al, ','
        call Writechar
        mov eax, edx
        call Writedec
        add eax, ebx
        mov ebx, edx
        mov edx, eax
    Loop L1
exit
main ENDP
END main
```

```
2,1,3,4,7,11,18,29,47,76
C:\Users\ftc\Desktop\COAl\Debug\COAl.exe (process 5256) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debu
gging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

```
'COAl.exe' (Win32): Loaded 'C:\Windows\SysWOW64\imm32.dll'.
The thread 0x1978 has exited with code 0 (0x0).
The thread 0x1cbc has exited with code 0 (0x0).
The thread 0x1090 has exited with code 0 (0x0).
The program '[5256] COAl.exe' has exited with code 0 (0x0).
```