

Questions

1. Write an uninitialized data declaration for a 16-bit signed integer val1

1010 Declare a string variable containing the word "TEST" repeated 500 times.

Initializers

4. Declare a string variable containing the name of your favorite color. Initialize

Ini

Declare a 32

negative decimal value.

3. Declare an unsigned 16-bit integer variable named that uses three

initialize 8-bit signed integer val2 with -11. 2-bit signed integer val3 and initialize it with the smallest possible

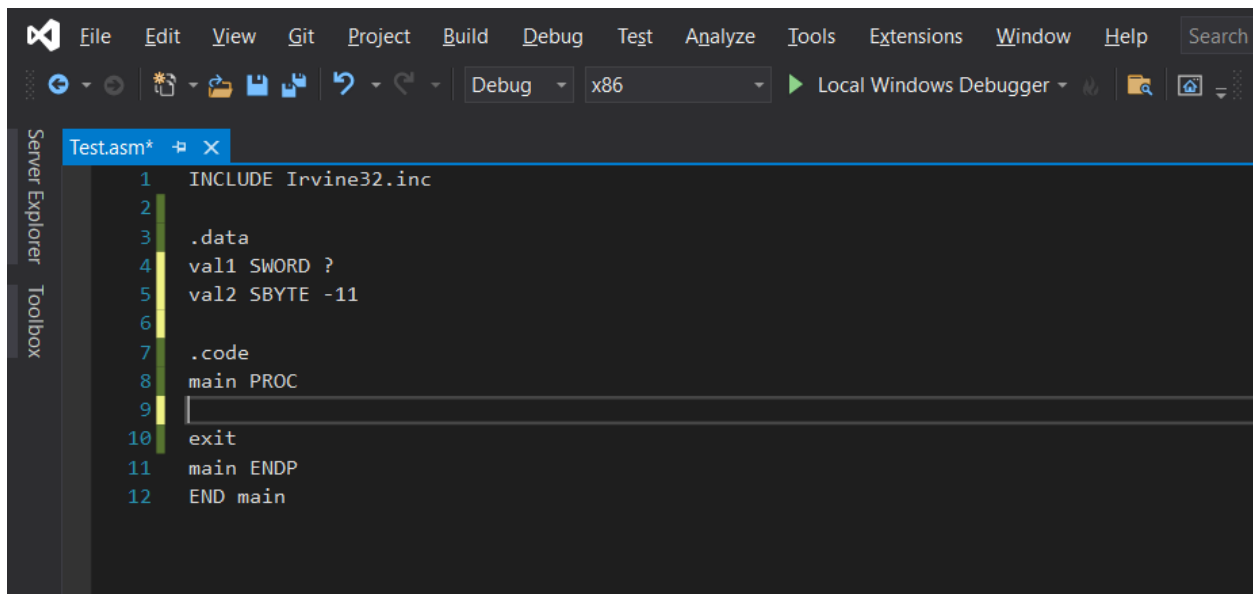
6. Convert the given values of a,b,c,d into binary and then use in 8-bit data definition and implement in the equation.

7. Declare an unsigned 16-bit integer variable named wArray that uses three initializers.

8. Declare an uninitialized array of 50 unsigned doublewords named dArray.

9. Declare a string variable containing the word "TEST" repeated 500 times.

10. Declare an array of 20 unsigned bytes named bArray and initialize all elements



The screenshot shows the Visual Studio Code interface with the file 'Test.asm' open. The code is as follows:

```
1  INCLUDE Irvine32.inc
2
3  .data
4  val1 SWORD ?
5  val2 SBYTE -11
6
7  .code
8  main PROC
9
10 exit
11 main ENDP
12 END main
```

Code:

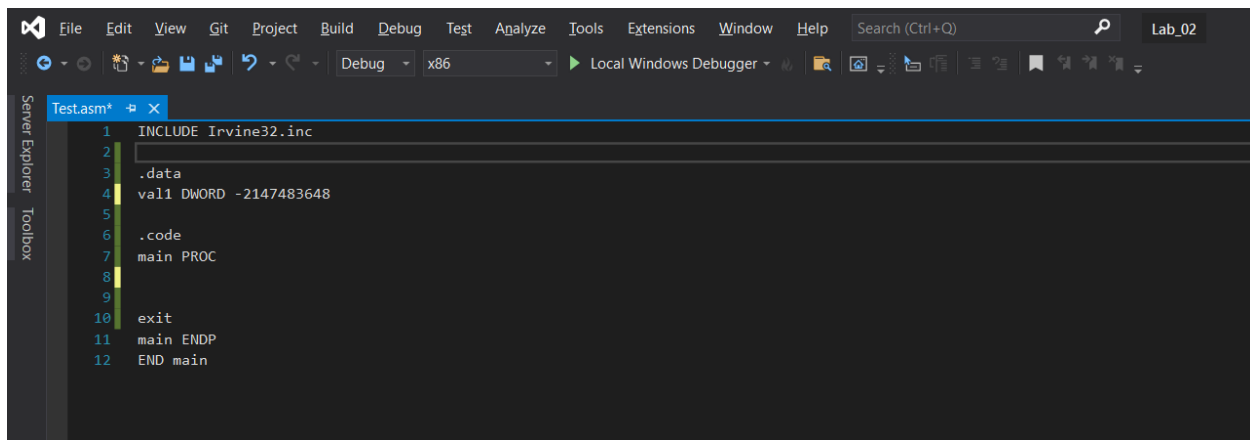
```
INCLUDE Irvine32.inc
```

```
.data
val1 SWORD ?
val2 SBYTE -11
```

```
.code
main PROC
```

```
exit
main ENDP
END main
```

Q2



```
1 INCLUDE Irvine32.inc
2
3 .data
4 val1 DWORD -2147483648
5
6 .code
7 main PROC
8
9
10 exit
11 main ENDP
12 END main
```

Code:

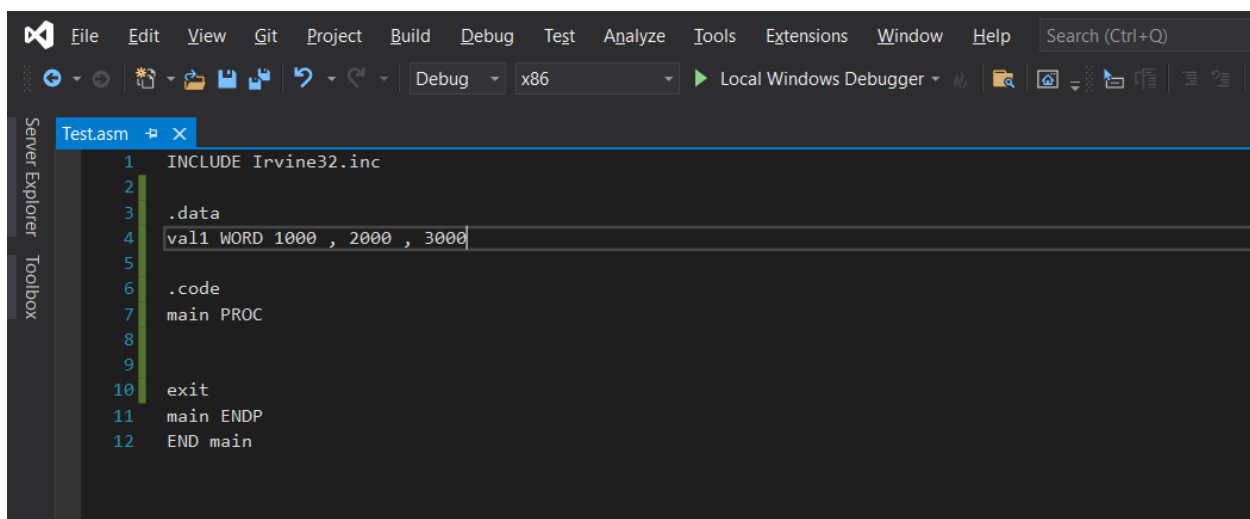
```
INCLUDE Irvine32.inc

.data
val1 DWORD -2147483648

.code
main PROC

exit
main ENDP
END main
```

Q3



```
1 INCLUDE Irvine32.inc
2
3 .data
4 val1 WORD 1000 , 2000 , 3000
5
6 .code
7 main PROC
8
9
10 exit
11 main ENDP
12 END main
```

Code:

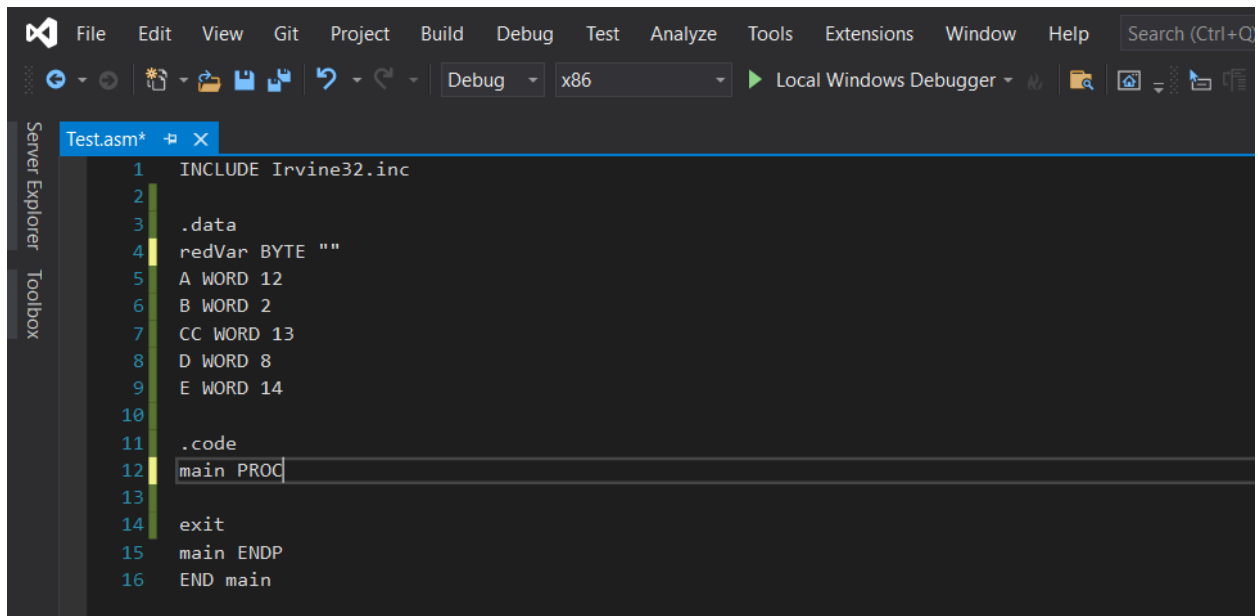
```
INCLUDE Irvine32.inc
```

```
.data
val1 WORD 1000 , 2000 , 3000
```

```
.code
main PROC
```

```
exit
main ENDP
END main
```

Q4



Code:

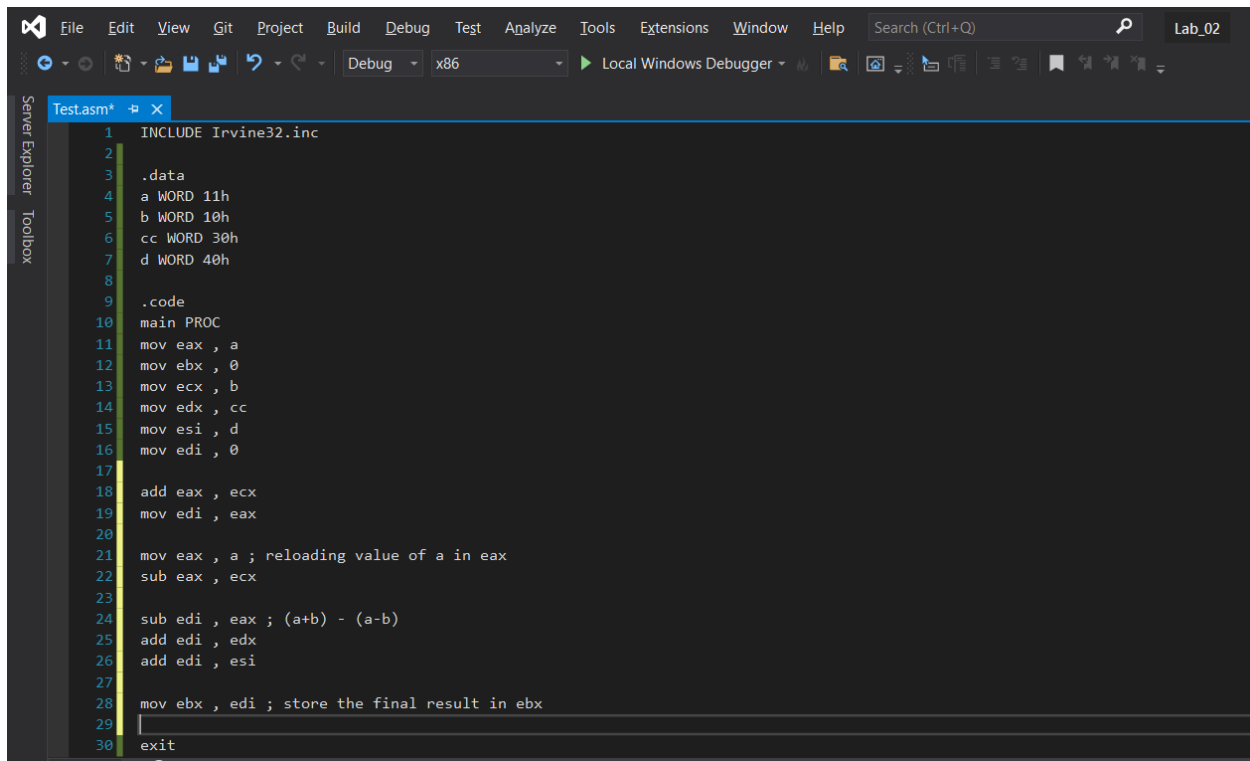
```
INCLUDE Irvine32.inc
```

```
.data
redVar BYTE ""
A WORD 12
B WORD 2
CC WORD 13
D WORD 8
E WORD 14
```

```
.code
main PROC
```

```
exit
main ENDP
END main
```

Q5



```
1 INCLUDE Irvine32.inc
2
3 .data
4 a WORD 11h
5 b WORD 10h
6 cc WORD 30h
7 d WORD 40h
8
9 .code
10 main PROC
11 mov eax, a
12 mov ebx, 0
13 mov ecx, b
14 mov edx, cc
15 mov esi, d
16 mov edi, 0
17
18 add eax, ecx
19 mov edi, eax
20
21 mov eax, a ; reloading value of a in eax
22 sub eax, ecx
23
24 sub edi, eax ; (a+b) - (a-b)
25 add edi, edx
26 add edi, esi
27
28 mov ebx, edi ; store the final result in ebx
29
30 exit
```

Code:

```
INCLUDE Irvine32.inc

.data
a WORD 11h
b WORD 10h
cc WORD 30h
d WORD 40h

.code
main PROC
mov eax, a
mov ebx, 0
mov ecx, b
mov edx, cc
mov esi, d
mov edi, 0

add eax, ecx
mov edi, eax

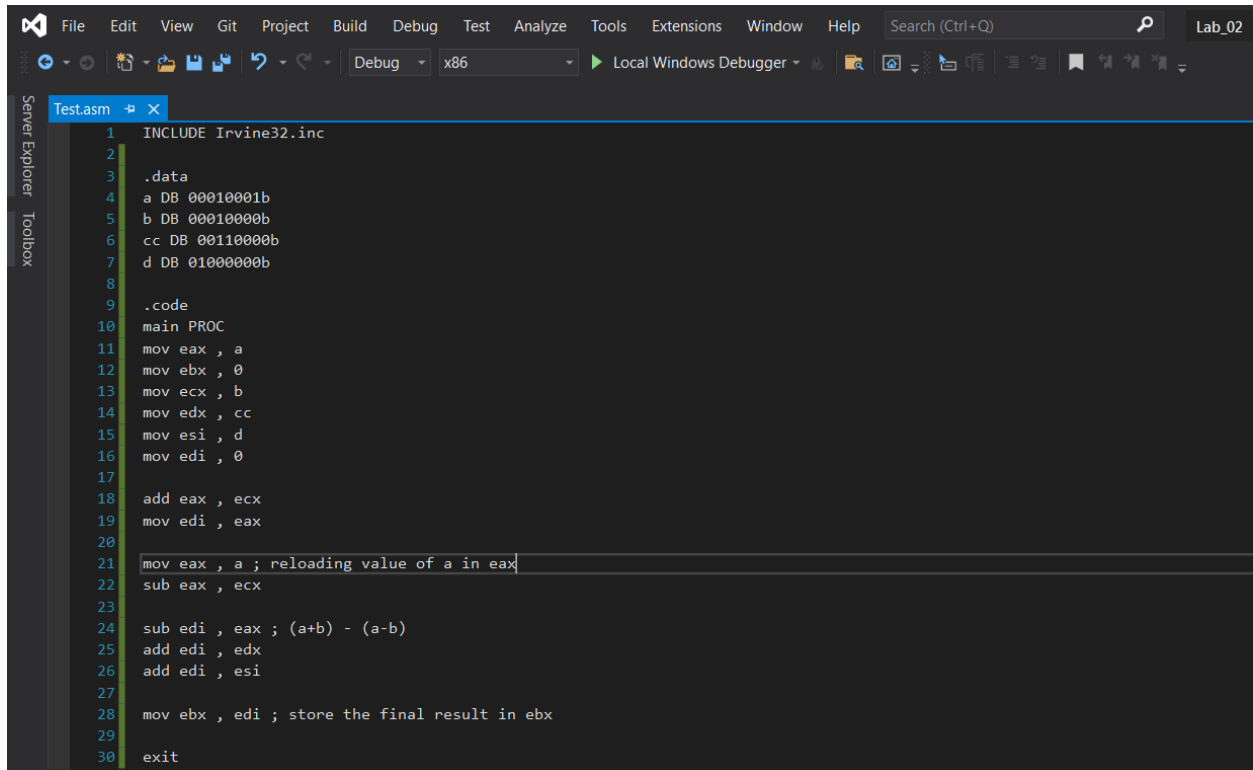
mov eax, a ; reloading value of a in eax
sub eax, ecx

sub edi, eax ; (a+b) - (a-b)
add edi, edx
add edi, esi

mov ebx, edi ; store the final result in ebx
```

```
exit
main ENDP
END main
```

Q6



```
1 INCLUDE Irvine32.inc
2
3 .data
4 a DB 00010001b
5 b DB 00010000b
6 cc DB 00110000b
7 d DB 01000000b
8
9 .code
10 main PROC
11 mov eax , a
12 mov ebx , 0
13 mov ecx , b
14 mov edx , cc
15 mov esi , d
16 mov edi , 0
17
18 add eax , ecx
19 mov edi , eax
20
21 mov eax , a ; reloading value of a in eax
22 sub eax , ecx
23
24 sub edi , eax ; (a+b) - (a-b)
25 add edi , edx
26 add edi , esi
27
28 mov ebx , edi ; store the final result in ebx
29
30 exit
```

Code :

```
INCLUDE Irvine32.inc
```

```
.data
```

```
a DB 00010001b ; 8 bit data defining to store binary number
```

```
b DB 00010000b
```

```
cc DB 00110000b
```

```
d DB 01000000b
```

```
.code
```

```
main PROC
```

```
mov eax , a
```

```
mov ebx , 0
```

```
mov ecx , b
```

```
mov edx , cc
```

```
mov esi , d
```

```
mov edi , 0
```

```
add eax , ecx
```

```
mov edi , eax
```

```

mov eax , a ; reloading value of a in eax
sub eax , ecx

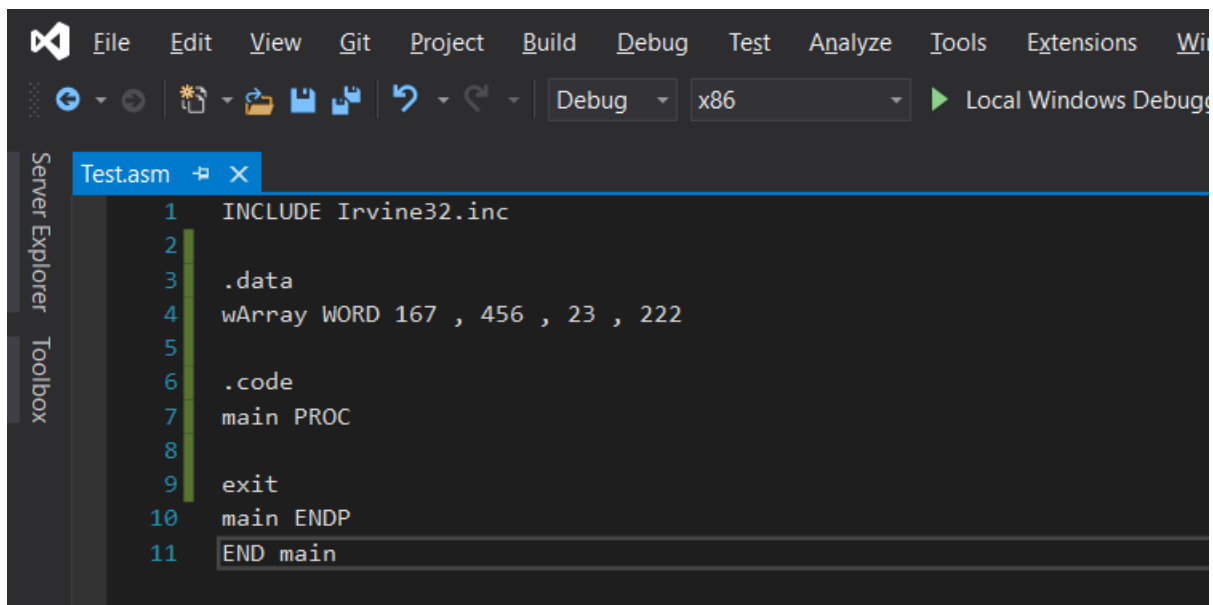
sub edi , eax ; (a+b) - (a-b)
add edi , edx
add edi , esi

mov ebx , edi ; store the final result in ebx

exit
main ENDP
END main

```

Q7



```

1  INCLUDE Irvine32.inc
2
3  .data
4  wArray WORD 167 , 456 , 23 , 222
5
6  .code
7  main PROC
8
9      exit
10 main ENDP
11 END main

```

Code:

```

INCLUDE Irvine32.inc

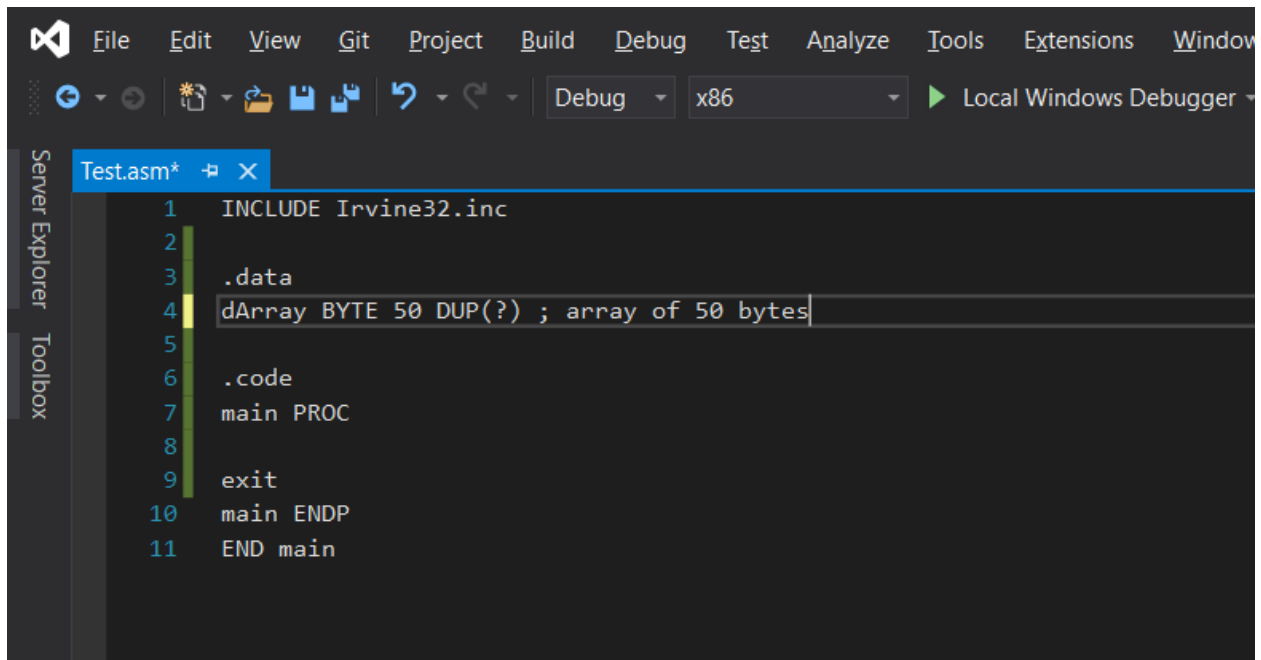
.data
wArray WORD 167 , 456 , 23 , 222

.code
main PROC

    exit
main ENDP
END main

```

Q8



```
1  INCLUDE Irvine32.inc
2
3  .data
4  dArray BYTE 50 DUP(?) ; array of 50 bytes
5
6  .code
7  main PROC
8
9  exit
10 main ENDP
11 END main
```

Code:

```
INCLUDE Irvine32.inc
```

```
.data
```

```
dArray BYTE 50 DUP(?) ; array of 50 bytes
```

```
dArray DB 50 DUP(?) ; directive to define an array of 50 bytes
```

```
.code
```

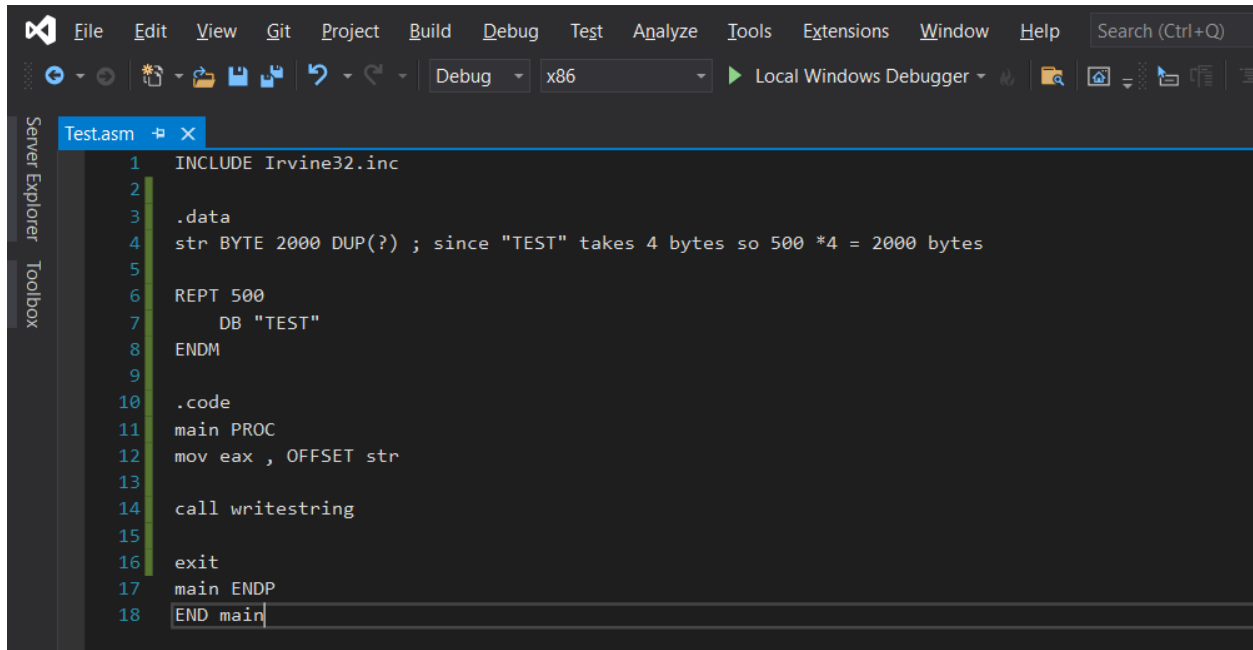
```
main PROC
```

```
exit
```

```
main ENDP
```

```
END main
```


Q9



```
1  INCLUDE Irvine32.inc
2
3  .data
4  str BYTE 2000 DUP(?) ; since "TEST" takes 4 bytes so 500 *4 = 2000 bytes
5
6  REPT 500
7      DB "TEST"
8  ENDM
9
10 .code
11 main PROC
12     mov eax , OFFSET str
13
14     call writestring
15
16     exit
17 main ENDP
18 END main
```

Code:

```
INCLUDE Irvine32.inc
```

```
.data
```

```
str BYTE 2000 DUP(?) ; Since "TEST" takes 4 bytes so 500*4 = 2000 bytes
```

```
REPT 500
```

```
    DB "TEST"
```

```
ENDM
```

```
.code
```

```
main PROC
```

```
mov eax, OFFSET str
```

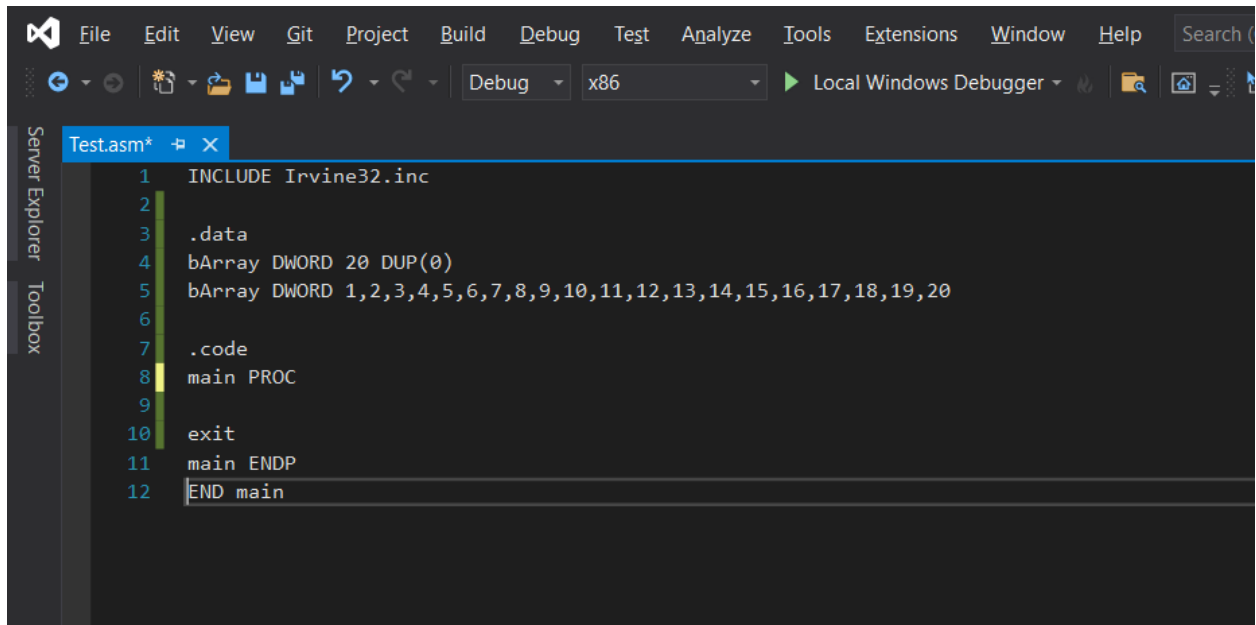
```
call WriteString
```

```
exit
```

```
main ENDP
```

```
END main
```

Q10



```
1  INCLUDE Irvine32.inc
2
3  .data
4  bArray DWORD 20 DUP(0)
5  bArray DWORD 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
6
7  .code
8  main PROC
9
10 exit
11 main ENDP
12 END main
```

Code:

```
INCLUDE Irvine32.inc
```

```
.data
```

```
bArray DWORD 20 DUP(0)
```

```
bArray DWORD 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
```

```
.code
```

```
main PROC
```

```
exit
```

```
main ENDP
```

```
END main
```