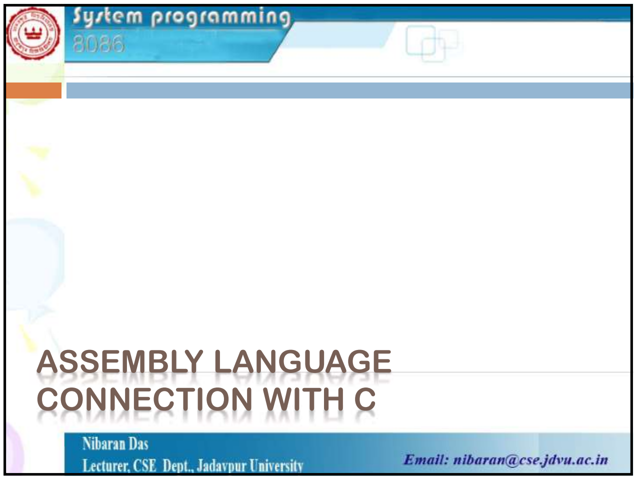
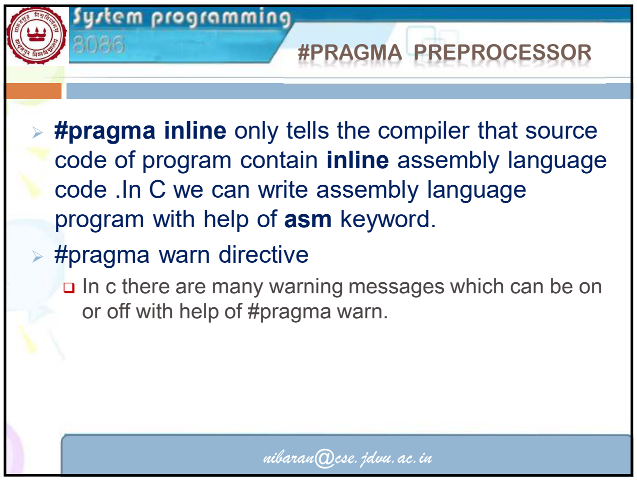


System programming
8086

ASSEMBLY LANGUAGE CONNECTION WITH C

Nibaran Das
Lecturer, CSE Dept., Jadavpur University
Email: nibaran@cse.jdvu.ac.in

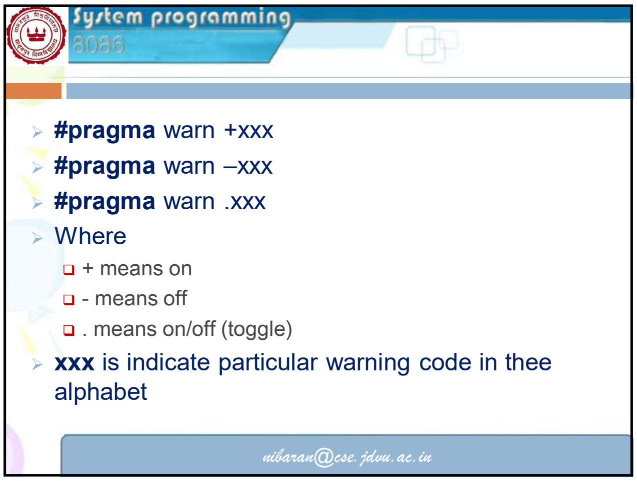


System programming
8086

#PRAGMA PREPROCESSOR

- **#pragma inline** only tells the compiler that source code of program contain **inline** assembly language code .In C we can write assembly language program with help of **asm** keyword.
- #pragma warn directive
  - In c there are many warning messages which can be on or off with help of #pragma warn.

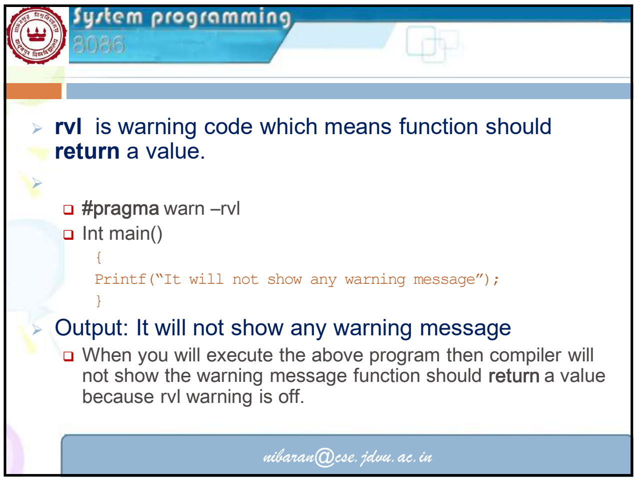
nibaran@cse.jdvu.ac.in



System programming
8086

- **#pragma** warn +xxx
- **#pragma** warn –xxx
- **#pragma** warn .xxx
- Where
  - + means on
  - - means off
  - . means on/off (toggle)
- **xxx** is indicate particular warning code in thee alphabet

nibaran@cse.jdvu.ac.in



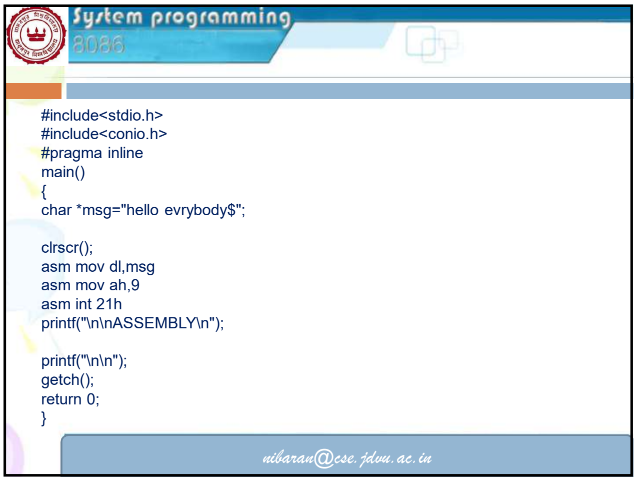
System programming
8086

- **rvl** is warning code which means function should **return** a value.
-
  - #pragma warn –rvl
  - Int main()

```
{
Printf("It will not show any warning message");
}
```

- Output: It will not show any warning message
  - When you will execute the above program then compiler will not show the warning message function should **return** a value because rvl warning is off.

nibaran@cse.jdvu.ac.in



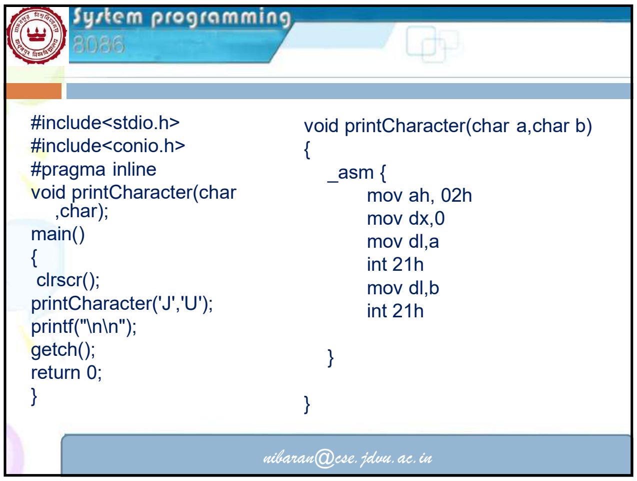
System programming
8086

```
#include<stdio.h>
#include<conio.h>
#pragma inline
main()
{
char *msg="hello evrybody$";

clrscr();
asm mov dl,msg
asm mov ah,9
asm int 21h
printf("\n\nASSEMBLY\n");

printf("\n\n");
getch();
return 0;
}
```

nibaran@cse.jdvu.ac.in



System programming
8086

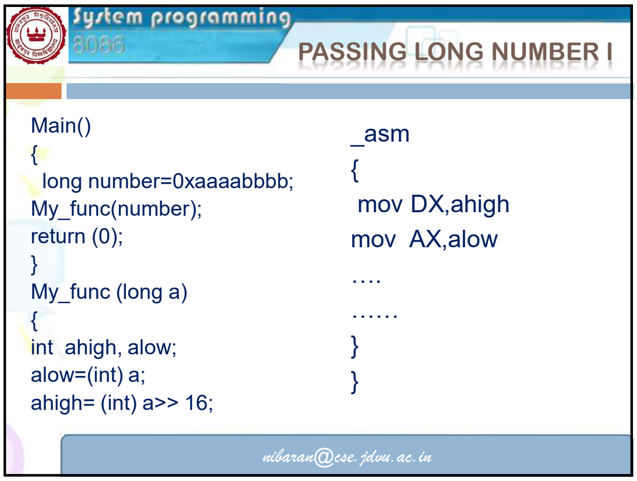
```
#include<stdio.h>
#include<conio.h>
#pragma inline
void printCharacter(char
    ,char);
main()
{
 clrscr();
printCharacter('J','U');
printf("\n\n");
getch();
return 0;
}
```

```
void printCharacter(char a,char b)
{
  _asm {
       mov ah, 02h
       mov dx,0
       mov dl,a
       int 21h
       mov dl,b
       int 21h

  }

}
```

nibaran@cse.jdvu.ac.in



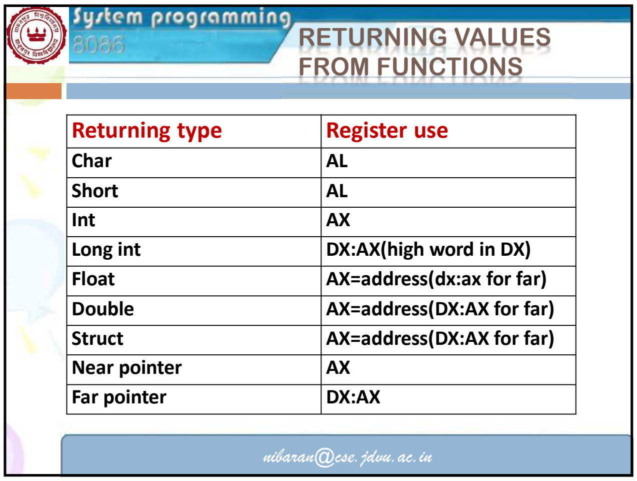
System programming
8086

PASSING LONG NUMBER I

```
Main()
{
 long number=0xaaaabbbb;
My_func(number);
return (0);
}
My_func (long a)
{
int  ahigh, alow;
alow=(int) a;
ahigh= (int) a>> 16;
```

```
_asm
{
 mov DX,ahigh
mov  AX,alow
….
……
}
}
```

nibaran@cse.jdvu.ac.in

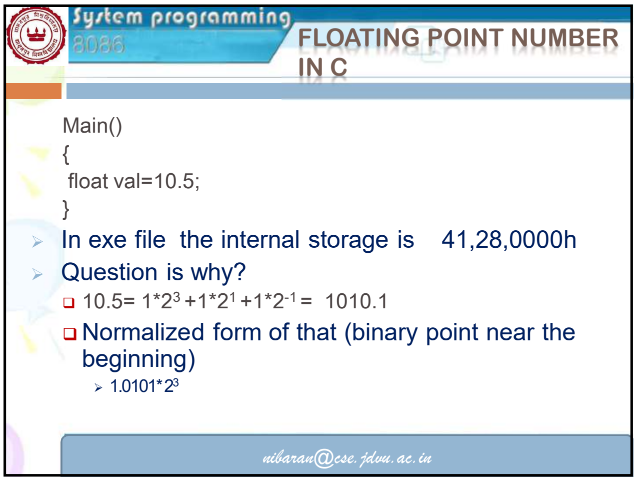


System programming
8086

RETURNING VALUES FROM FUNCTIONS

Returning type	Register use
Char	AL
Short	AL
Int	AX
Long int	DX:AX(high word in DX)
Float	AX=address(dx:ax for far)
Double	AX=address(DX:AX for far)
Struct	AX=address(DX:AX for far)
Near pointer	AX
Far pointer	DX:AX

nibaran@cse.jdvu.ac.in



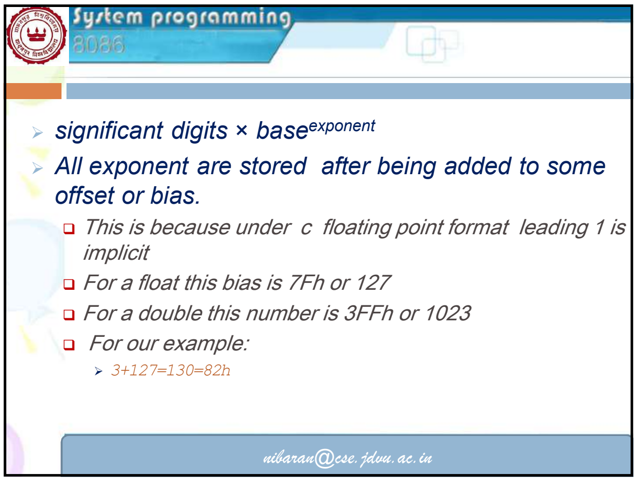
System programming
8086

FLOATING POINT NUMBER IN C

```
Main()
{
 float val=10.5;
}
```

- In exe file the internal storage is 41,28,0000h
- Question is why?
  - 10.5= $1*2^3 +1*2^1 +1*2^{-1}$ = 1010.1
  - Normalized form of that (binary point near the beginning)
    - $1.0101*2^3$

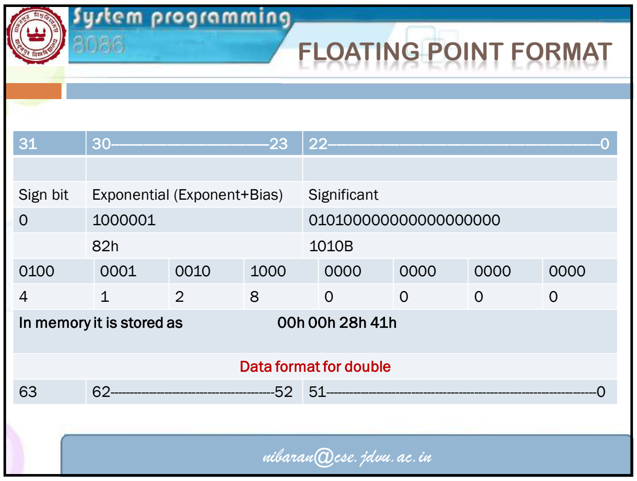
nibaran@cse.jdvu.ac.in



System programming
8086

- *significant digits × base^{exponent}*
- *All exponent are stored after being added to some offset or bias.*
  - *This is because under c floating point format leading 1 is implicit*
  - *For a float this bias is 7Fh or 127*
  - *For a double this number is 3FFh or 1023*
  - *For our example:*
    - *3+127=130=82h*

nibaran@cse.jdvu.ac.in



System programming
8086

FLOATING POINT FORMAT

31	30————————————23			22————————————————————0			
Sign bit	Exponential (Exponent+Bias)			Significant			
0	1000001			01010000000000000000000			
	82h			1010B			
0100	0001	0010	1000	0000	0000	0000	0000
4	1	2	8	0	0	0	0
In memory it is stored as			**00h 00h 28h 41h**				
			Data format for double				
63	62-----------------------------------52			51---0			

nibaran@cse.jdvu.ac.in

System programming
8086

nibaran@cse.jdvu.ac.in