A.Y. 2022-2023

PROCESSOR ORGANIZATION AND ARCHITECTURE

AYUSH JAIN

COMPUTER ENGINEERING | TE – B2 | 60004200132

EXPERIMENT – 10

Aim: To implement mixed programming using assembly language and C

Theory:

Mixed Language Programming using Assembly and C
  ● C generates an object code that is extremely fast and compact, but it is not as fast as the object code generated by a good programmer using assembly language.
  ● It is true that the time needed to write a program in assembly language is much more than the time taken in higher level languages like C.
  ● However, there are special cases where a function coded in assembly language reduces execution time.
    Example: The floating point math package must be coded in assembly language as it is used frequently and its execution speed will have great effect on the overall speed of the program that uses it.
  ● There are also occasions where some hardware devices need exact timing and then it is necessary to write assembly level programs to meet such strict timing restrictions.
 ● In addition, certain instructions cannot be executed in higher level languages like C
    Example: C does not have an instruction for performing bitwise rotation operation.

Thus, in spite of C being very powerful, routines must be written in assembly language to:
  ● Increase the speed and efficiency of the routine.
  ● Perform machine specific functions not available in Microsoft C or in Turbo C.
  ● Use 3rd party routines.

A.Y. 2022-2023
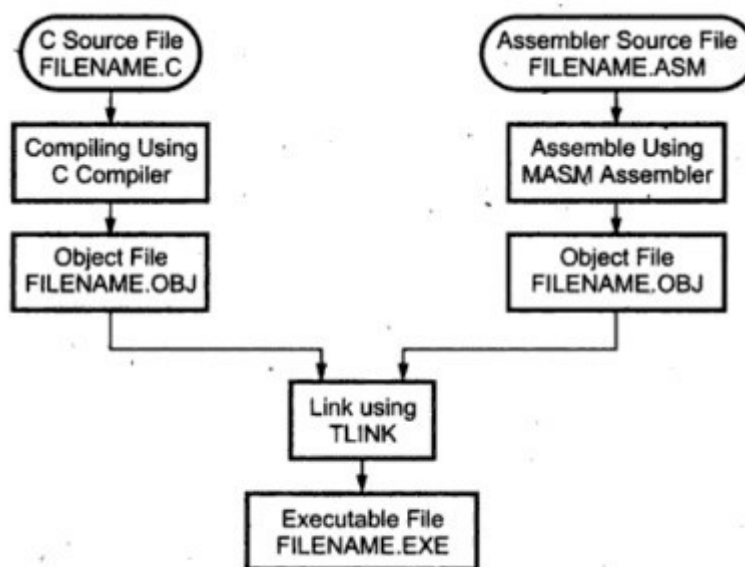
Compiler, assembly and link processes

Rules-

- Register Naming: Register names are prefixed with %, so that registers are %eax, %cl etc, instead of just eax, cl. We use %% for accessing the registers.
- Ordering of operands: Unlike Intel convention (first operand is destination), the order of operands is source(s) first, and destination last. For example, Intel syntax "mov eax, edx" will look like "mov %edx, %eax" in AT&T assembly.

Code:

```c
#include<stdio.h>
int main(void)
{
  int num1 = 10, num2 = 15, sum = 0;
  asm __volatile (
    "addl %%ebx,%%eax"
    :"=a"(num1)
    :"a"(num1), "b"(num2)
  );
  printf("num1 + num2=%d\n", num1);
  asm __volatile (
```

```
    "movl %%eax, %%ecx"
   :"=c"(sum)
   :"a"(num1)
 );
 printf("num1 + num2=%d\n", sum);
 return 0;
}
```

Output:

```
num1 + num2=25
num1 + num2=25


...Program finished with exit code 0
Press ENTER to exit console.
```

Conclusion:
Hence, we implemented mixed mode programming with C and assembly