Assembly Language LAB

Islamic University – Gaza
Engineering Faculty
Department of Computer Engineering
2013
ECOM 2125: Assembly Language LAB
Created by: Eng. Ahmed M. Ayash
Modified and Presented By: Eihab S. El-Radie

Lab # 9

Advanced Procedures

Objective:

To learn more about procedures.

Part1: Stack Frame (activation record):

Area of the stack set aside for passed arguments, subroutine return address, local variables, and saved registers.

Created by the following steps:

1. Passed arguments, if any, are pushed on the stack.
2. The subroutine is called, causing the subroutine return address to be pushed on the stack.
3. As the subroutine begins to execute, EBP is pushed on the stack.
4. EBP is set equal to ESP. From this point on, EBP acts as a base reference for all of the subroutine parameters.
5. If there are local variables, ESP is decremented to reserve space for the variables on the stack.
6. If any registers need to be saved, they are pushed on the stack.

➢ Explicit Access to Stack Parameters:

A procedure can explicitly access stack parameters using constant offsets from EBP.

- EBP is often called the base pointer or frame pointer because it holds the base address of the stack frame.

- EBP must be restored to its original value when a procedure returns.

➢ RET Instruction:

- Return from subroutine.
- Pops stack into the instruction pointer (EIP or IP). Control transfers to the target address.

Syntax:

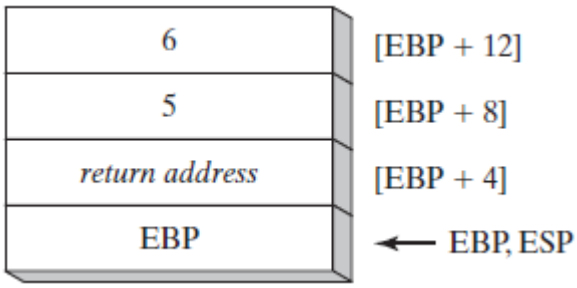
- RET
- RET *n*

- Optional operand *n* causes *n* bytes to be added to the stack pointer after EIP (or IP) is assigned a value.

Example:

```
.data
sum DWORD ?
.code
push 6        ; second argument
push 5        ; first argument
call AddTwo   ; EAX = sum
mov  sum,eax ; save the sum
```

```
AddTwo PROC
push ebp
mov ebp,esp ; base of stack frame
mov eax,[ebp + 12] ; second parameter
add eax,[ebp + 8] ; first parameter
pop ebp
ret
AddTwo ENDP
```



Part2: Local Variables:

- **A local variable** is created, used, and destroyed within a single procedure.
- Local variables are created on the runtime stack, usually below the base pointer (EBP).
- The LOCAL directive declares a list of local variables and immediately follows the PROC directive, each variable is assigned a type.
- Syntax: LOCAL varlist

Syntax:

LOCAL var1:type1, var2:type2, . . .

Example:

```
MySub PROC
   LOCAL var1:BYTE, var2:WORD, var3:DWORD
```

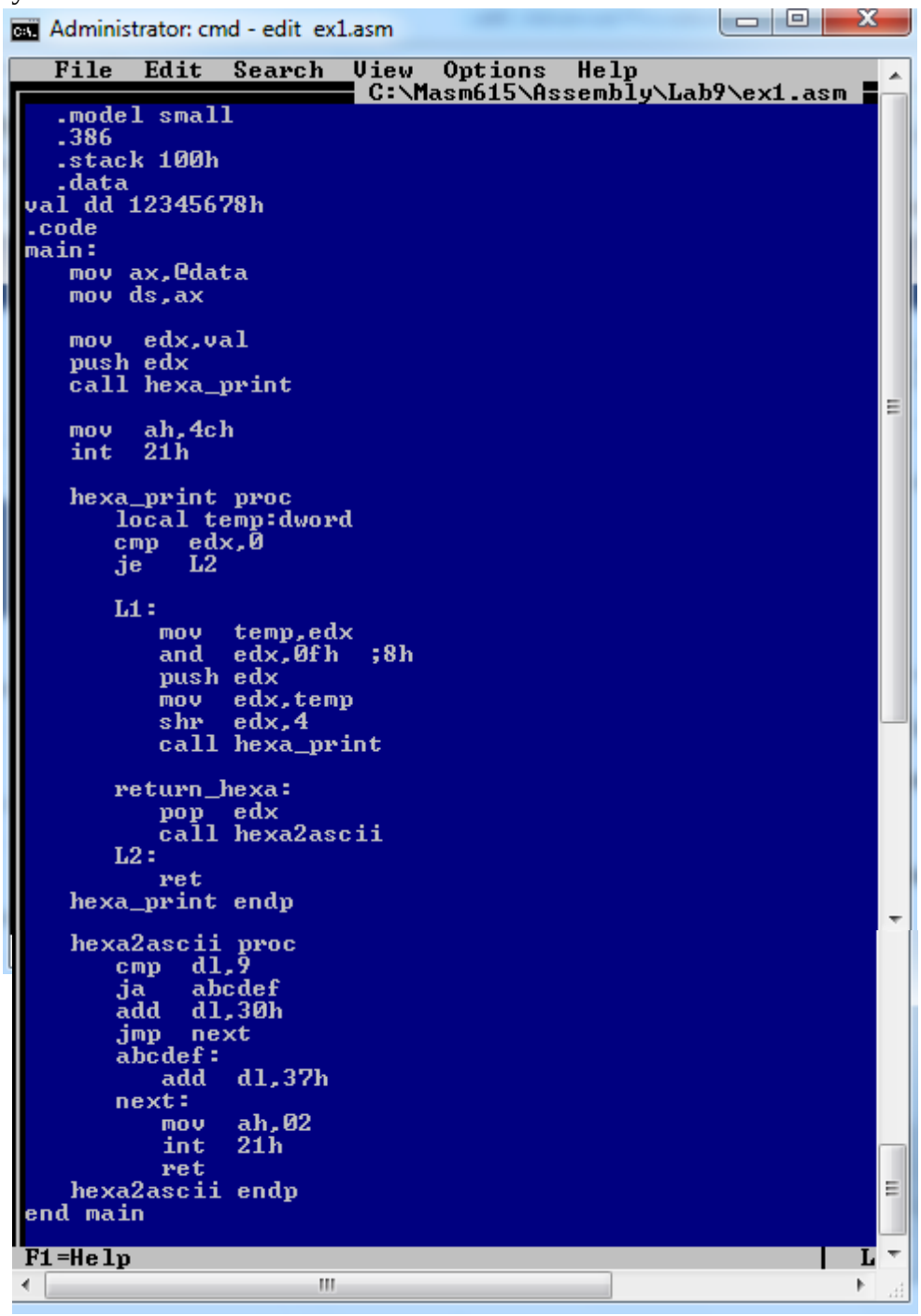
Part3: Recursion:

- A recursive procedure is one that calls itself, either directly or indirectly.
- Recursion, the practice of calling recursive procedures, can be a powerful tool when working with data structures that have repeating patterns.

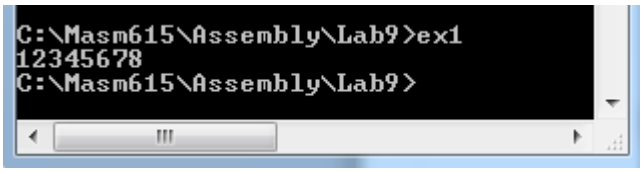
Lab work:

Excercise1:

Write an assembly recursive procedure that prints a hexadecimal number saved in memory on the screen.



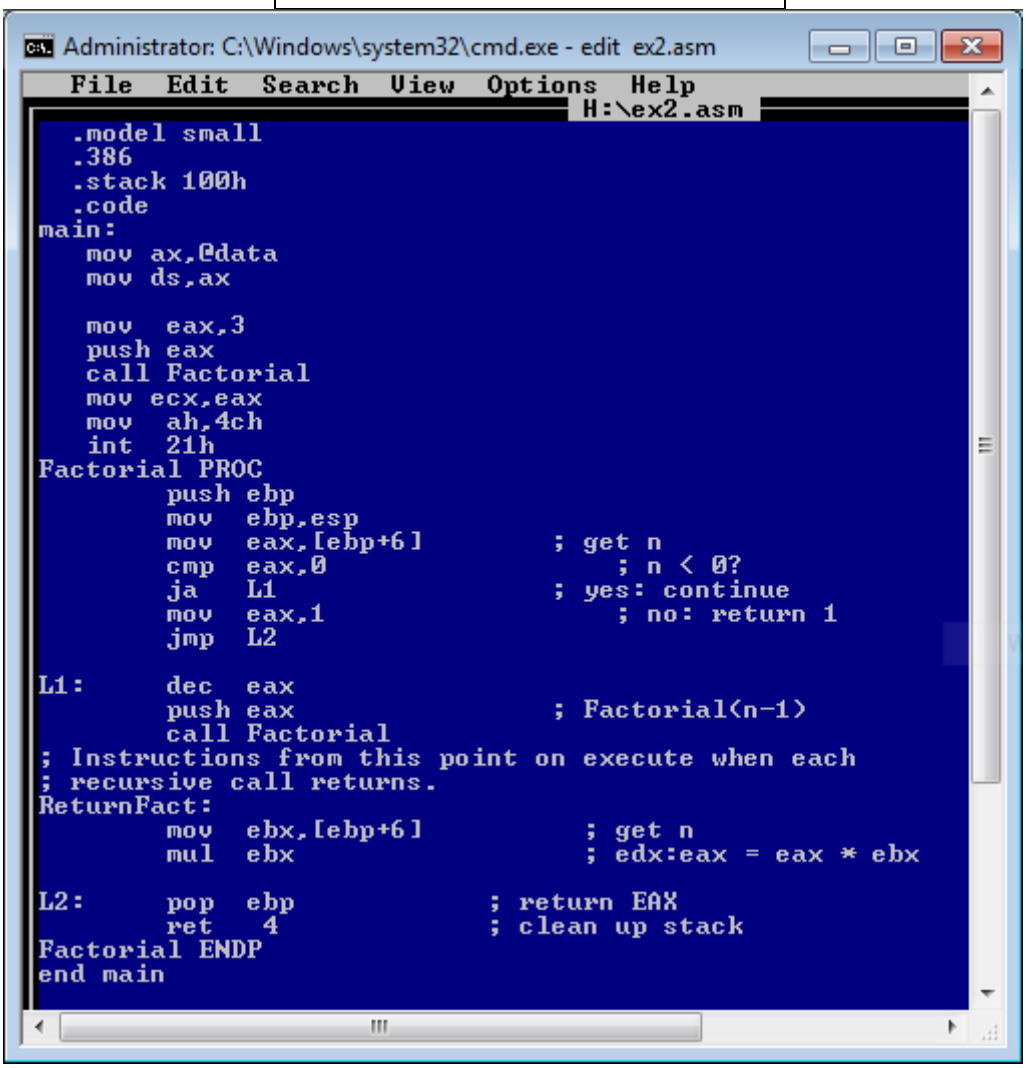
Output:

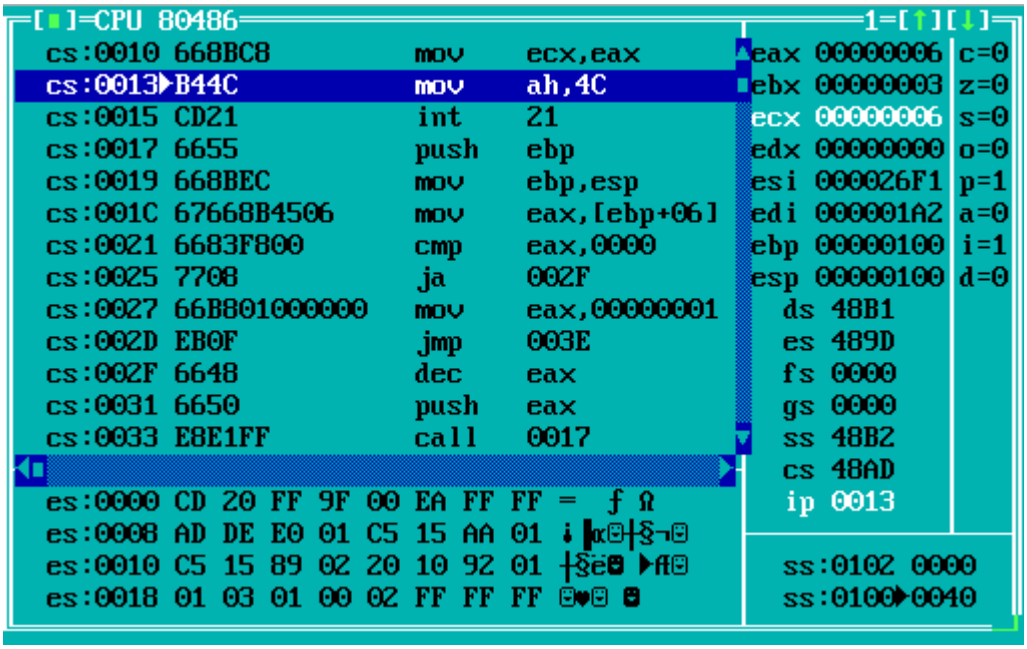


Excercise2:

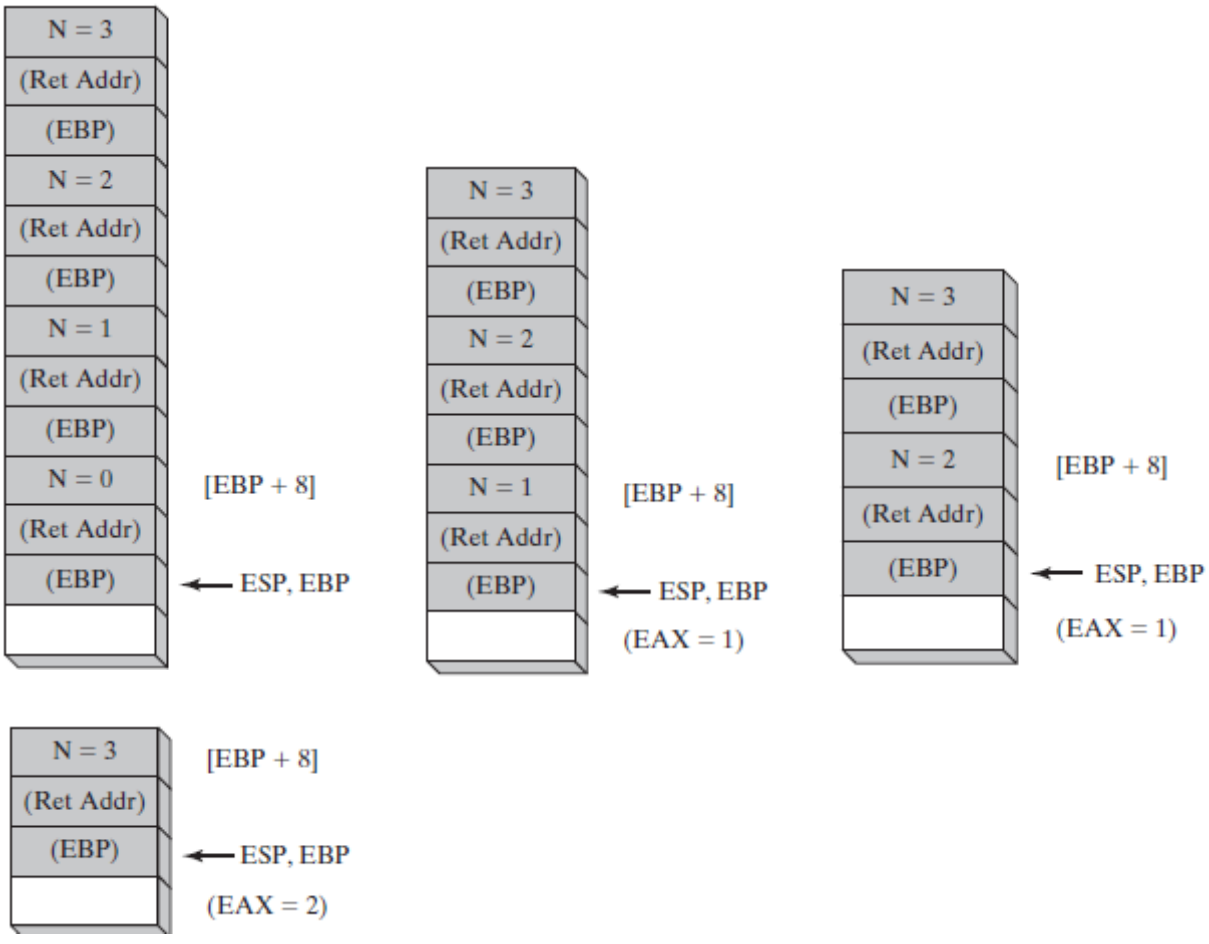
This function calculates the factorial of integer n ≥ 0. A new value of *n* is saved in each stack frame: (use n=3, 5, 12)

```
int function factorial(int n)
{
if(n == 0)
return 1;
else
return n * factorial(n-1);
}
```






```
L2: pop ebp
    ret 4
```



Homework:

1. Write an assembly recursive procedure that calculates the GCD of any two numbers using the following algorithm:

```
GCD(a,b) {
If (a%b==0)
return b;
else
return GCD(b , a%b);
}
```

Find: GCD(72,24), GCD(8,12).

2. Modify Excersise1 so that the output will be 87654321.