Local workspace of a subroutine:
A number of temporary memory locations required by the subroutine for temporary private variables used in addition to available data registers.

Recursion and recursive subroutines:
Recursion involves defining the solution of a problem in terms of itself. A recursive subroutine is one that calls itself.

Re-entrant subroutines:
In multi-tasking systems, a subroutine is re-entrant if more than one task or process are allowed to use (call) the subroutine simultaneously without any ill effects.
The Stack and Local Subroutine Variables: Stack Frames

- In order for a subroutine to be *recursive* or *re-entrant*, the subroutine’s local workspace must be *attached* to each use or call of the subroutine.

- A stack frame (SF) of size $d$ bytes is defined as a region of temporary storage in memory of size $d$ bytes at the top of the current stack.

- Upon creating a stack frame:
  - The frame pointer (FP) points to the bottom of the stack frame. Register A6 is normally used as the frame pointer.
  - The stack pointer, SP is updated to point to the top of the frame.

- In 68000 assembly, the LINK and UNLK instructions are used to facilitate the creation/destruction of local subroutine storage using stack frames.
Example: Factorial Using Iteration

**Computation:**
The factorial of a positive integer is defined as:
\[ n! = n \times (n-1) \times (n-2) \times (n-3) \times \ldots \times 1 \]

**Pseudo code:**
```plaintext
Factorial(N)
If N = 1 THEN
    Factorial(N) := 1
ELSE
    Factorial(N) = N * Factorial(N-1)
ENDIF
```

**Assembly Subroutine using iteration:**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVE.L D0,-(SP)</td>
<td>Save the initial value of D0</td>
</tr>
<tr>
<td>MOVE.W D0,D1</td>
<td>Set the result to the input value</td>
</tr>
<tr>
<td>SUBQ.W #1,D0</td>
<td>WHILE ( N &gt; 1 )</td>
</tr>
<tr>
<td>BEQ Exit</td>
<td>( N = N - 1 )</td>
</tr>
<tr>
<td>MULU D0,D1</td>
<td>( \text{Factorial}_N = N \times \text{Factorial}_N )</td>
</tr>
<tr>
<td>BRA Loop</td>
<td></td>
</tr>
<tr>
<td>MOVE.L (SP)+,D0</td>
<td>Restore the value of D0</td>
</tr>
<tr>
<td>RTS</td>
<td></td>
</tr>
</tbody>
</table>
Factorial Using Recursion

- Assembly program to compute factorial of a number using recursive subroutine calls.
- Subroutine parameter passing: by value via data registers.

Main program:

```
ORG           $1000
MAIN     MOVE.W   NUMB,D0    get number
          JSR      FACTOR     go to factorial routine
          MOVE.W   D0,F_NUMB  store result
          STOP #$2700
OR
ORG      $2000
NUMB      DC.W     5         number to be factorialized
F_NUMB    DS.W    1         factorial of number
```
Factorial Using Recursion: Subroutine

* Initial conditions:  D0.W = number to compute factorial of
*                               where  0 < D0.W < 9  (range to avoid overflow)
* Final conditions:  D0.W = factorial of input number
* Register usage:  D0.W destructively used
* Sample case:  Input:  D0.W = 5
*                               Output:  D0.W = 120

FACTOR    MOVE.W     D0,-(SP)    push input number onto stack
     SUBQ.W       #1,D0       decrement number
     BNE             F_CONT  reached 1 yet?
     MOVE.W     (SP)+,D0   yes:  factorial = 1
     RTS                 return
     F_CONT    JSR               FACTOR    no:  recursively call  FACTOR
     MULU          (SP)+,D0    multiply only after stack
*                                                                       contains all numbers
     RETURN    RTS
Factorial Using Recursion Example:

Effect On Stack
Creating A Stack Frame of Size \( d \) Bytes

Stack just after a subroutine call

Current SP
A7

Passed Parameters

Return Address

Stack just after creating a stack frame of size \( d \) by the subroutine

LEA -4(SP),A6

LEA -d(SP),SP

Current SP
A7

Current FP
A6

Stack Frame

Return Address

Passed Parameters

EECC250 - Shaaban
Destroying A Stack Frame

LEA d(SP),SP

Word

Passed Parameters

Stack after destroying the stack frame

Current SP
A7

Current FP
A6

Stack Frame

d

Return Address

Passed Parameters

Current SP
A7

Return Address

Passed Parameters
LINK An,-# d  LINK Instruction

- Allocates or creates a frame in the stack for local use by the subroutine of size $d$ bytes.
- An is an address register serving as the frame pointer (FP); A6 is used.
- Function:
  - Push the contents of address register An onto the stack. (includes pre-decrementing SP by 4).
  - Save the stack pointer in An (An points to bottom of frame)
  - Decrement the stack pointer by $d$ (points to the top of the frame)
  - Similar in functionality to the following instruction sequence:
    
    ```
    MOVEA.L A6,-(SP)
    LEA (SP),A6
    LEA -d(SP),SP
    ```

- After creating the frame:
  - Passed parameters are accessed with a positive displacement with respect to FP, A6 i.e. MOVE.W 8(A6),D0
  - Local temporary storage variables are accessed with negative displacement with respect to A6 i.e. MOVE.L D2,-10(A6)
LINK Instruction Operation

Stack just after a subroutine call before LINK

Current SP
A7

Passed Parameters

Return Address

Current FP
A6

Stack Frame

LINK A6,- # d

Passed Parameters

Return Address

EECC250 - Shaaban
UNLK UNLinK Instruction

UNLK An

• Deallocates or destroys a stack frame. Where An is the address register used as frame pointer (FP); usually A6

• Function:
  – Restore the stack pointer to the value in address register An
    i.e. SP = An or SP = SP + d

  – Restore register An by popping its value from the stack. (includes post-incrementing SP by 4).

Similar in functionality to the following instruction sequence:

```
LEA d(SP),SP
MOVEA.L (SP)+,An
```
UNLK Instruction Operation

Current SP
A7

Stack Frame

UNLK A6

Passed
Parameters

Return Address

Original
A6

Passed Parameters

Current SP
A7

Return Address

Stack just after UNLINK A6

EECC250 - Shaaban
Example: Using
A Stack Frame FP

A segment of a main calling program:

MOVE.W D0,-(SP)     Push parameter #1 onto the stack
MOVE.W D1,-(SP)     Push parameter #2 onto the stack
JSR SBRT             Jump to subroutine SBRT

A segment of a subroutine using a stack frame for local storage:

SBRT LINK A6,-#8      Establish FP and local storage
...                ...
MOVE.W 10(A6),D5     Retrieve parameter #1
...                ...
MOVE.W D4,-4(A6)     local write to stack frame
...                ...
UNLK A6              Deallocate stack frame
RTS
SF Example: Effect On Stack

LINK A6,-#$8
The Effect of Multiple Subroutine Calls on Frame Pointers When Using Local Storage