68000 Stack-Related Instructions

PEA <EA> Push Effective Address

- **Calculates an effective address <ea> and pushes it onto the stack pointed at by address register A7 (the stack pointer, SP).**
- **The difference between PEA and LEA**
  - **LEA loads an effective address in any address register.**
  - **PEA pushes an effective address onto the stack.**
- **PEA <EA> is equivalent to:**

```
LEA      <EA>,Ai
MOVEA.L  Ai,-(A7)
```

Where Ai is an address register other than A7 (A0-A6)

The MOVE Multiple: MOVEM Instruction

- This instruction saves or restores multiple registers.
- Useful in subroutines to save the values of registers not used to pass parameters. MOVEM has two forms:

MOVEM register_list,<ea>

MOVEM <ea>,register_list

- No effect on CCR.

Example: Saving/restoring registers to from memory

```
SUBR1   MOVEM   D0-D7/A0-A6,SAVEBLOCK        SAVE D0-D7/A0-A6
        . . .
        MOVEM   SAVEBLOCK,D0-D7/A0-A6        Restore D0-D7/A0-A6
        RTS
```

Example: Saving/restoring registers using the stack (preferred method).

```
SUBR1   MOVEM   D0-D7/A0-A6,-(SP)      Push D0-D7/A0-A6 onto the stack
        . . .
        MOVEM   (SP)+,D0-D7/A0-A6      Restore D0-D7/A0-A6  from the stack
        RTS
```

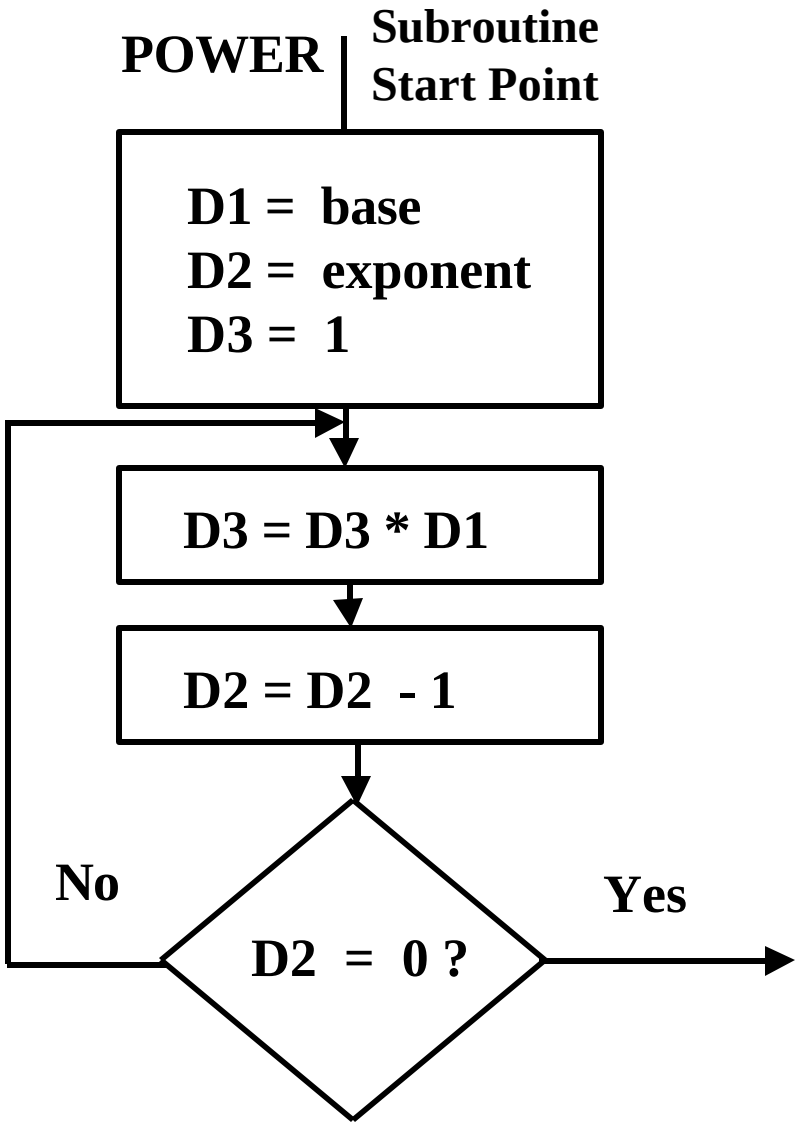
Example: Power Calculation Subroutine

- **A subroutine is needed which accepts two integers as input parameters:**
  - **a base, B (a signed integer), Size = one byte (range: $-128 \leq B \leq 127$)**
  - **an exponent E (a positive integer) Size = one byte,**
  - **and, compute the function B^E size of answer = long word**

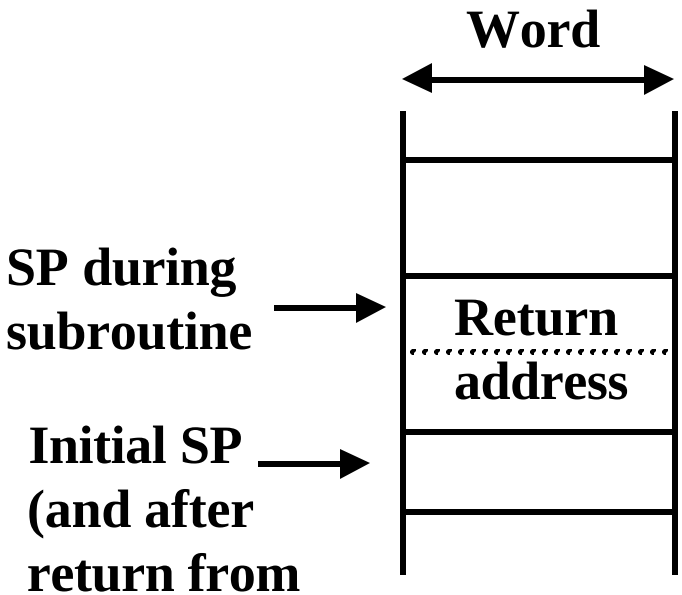
Functional specification (pseudo code) of subroutine POWER:

```
POWER (B, E)

D1  = B                          ;input arguments, base
D2  = E                          ;exponent, a positive integer

initialize D3 to 1               ;answer initialized to 1
while  D2  > 0
     D3 =  D1*D3                 ;compute function  using
     D2 = D2  - 1;               ;continued product of base
end POWER
Return to calling program
```

Basic Flow Chart of Power



Effect on The Stack



POWER: Four Parameter Passing Cases

- **We'll examine four assembly versions of the subroutine POWER and sample Main programs that calls it.**
- **Each version uses a different parameter passing method:**

  - **Case 1: Parameter passing *by value*, using data registers.**
  - **Case 2: Parameter passing *by reference*, using address registers.**
  - **Case 3: Parameter passing *by value*, using the stack.**
  - **Case 4: Parameter Passing *by reference*, using the stack**

POWER Subroutine Example (Case 1)

Parameter Passing *by Value*: Using Data Registers
- Main Program -

```
MAIN    ORG       $400              Main Program origin
        MOVEA.L   #$07FFE,SP        Initialize  Stack Pointer
        MOVE.B    B,D1              Put base number into D1
        EXT.W     D1                Sign extend base to word length
        CLR.W     D2                Clear D2 before loading exponent
        MOVE.B    E,D2              Put exponent number into D2
        BSR       POWER             Call subroutine  POWER
        LEA       A,A5               put address of  answer into A5
        MOVE.L    D3,(A5)           save  answer
        STOP      #$2700            Done
        ORG       $600
B       DC.B      4                 Base number stored here
E       DC.B      2                 Exponent number stored here
A       DS.L      1                 answer to be stored  here
```

POWER Subroutine Example (Case 1)

Parameter Passing *by Value*: Using Data Registers Continued - Subroutine

```
        ORG     $800        Subroutine  POWER origin
POWER   MOVE.L  #1,D3       initialize result to 1
LOOP    MULS    D1,D3       multiply result with base
        SUB     #1,D2       decrement power by one
        BNE     LOOP        and repeat as long as power > 0
        RTS                 Done, return to calling program
```

POWER Subroutine Example (Case 2)

Parameter Passing *by Reference*: Using Address Registers

- Main Program -

```
MAIN    ORG       $400          Main Program origin
        MOVEA.L   #$07FFE,SP    Initialize  Stack Pointer
        LEA       B,A1          A1  points to base number
        LEA       E,A2          A2  points to exponent
        BSR       POWER         Call subroutine  POWER
        LEA       A,A5           put address of  answer into A5
        MOVE.L    D3,(A5)       save  answer in memory
        STOP      #$2700        Done
        ORG       $600
B       DC.B      4             Base number stored here
E       DC.B      2             Exponent number stored here
A       DS.L      1             answer to be stored  here
```

POWER Subroutine Example (Case 2)

Parameter Passing *by Reference*: Using Address Registers
Continued - Subroutine

```
        ORG     $800        Subroutine  POWER origin
POWER   MOVE.B  (A1),D1     copy base number to D1
        EXT.W   D1          Sign extend base to word length
        CLR.W   D2          Clear D2 before loading exponent
        MOVE.B  (A2),D2     copy exponent to D2
        MOVE.L  #1,D3       initialize result in D3 to 1
LOOP    MULS    D1,D3       multiply result D3 with base D1
        SUB     #1,D2       decrement power in D2 by one
        BNE     LOOP        and repeat as long as power > 0
        RTS                 Done, return to calling program
```

68000 Addressing Modes Revisited:

Address Register Indirect Addressing with Displacement

- **The addressing notation:**

  d16(A0) or (d16,A0)

  - **Refers to the address contained in register A0 plus a signed 16 bit displacement d16**
  - **Some assembles accept only one of the above syntax forms.**

- **Examples:**

MOVE.L (12,A4),D3 or MOVE.L 12(A4),D3

MOVE.W (-$4,A1),D0 or MOVE.W -$4(A1),D0

POWER Subroutine Example (Case 3)

Parameter Passing *by Value*: Using The Stack - Main Program -

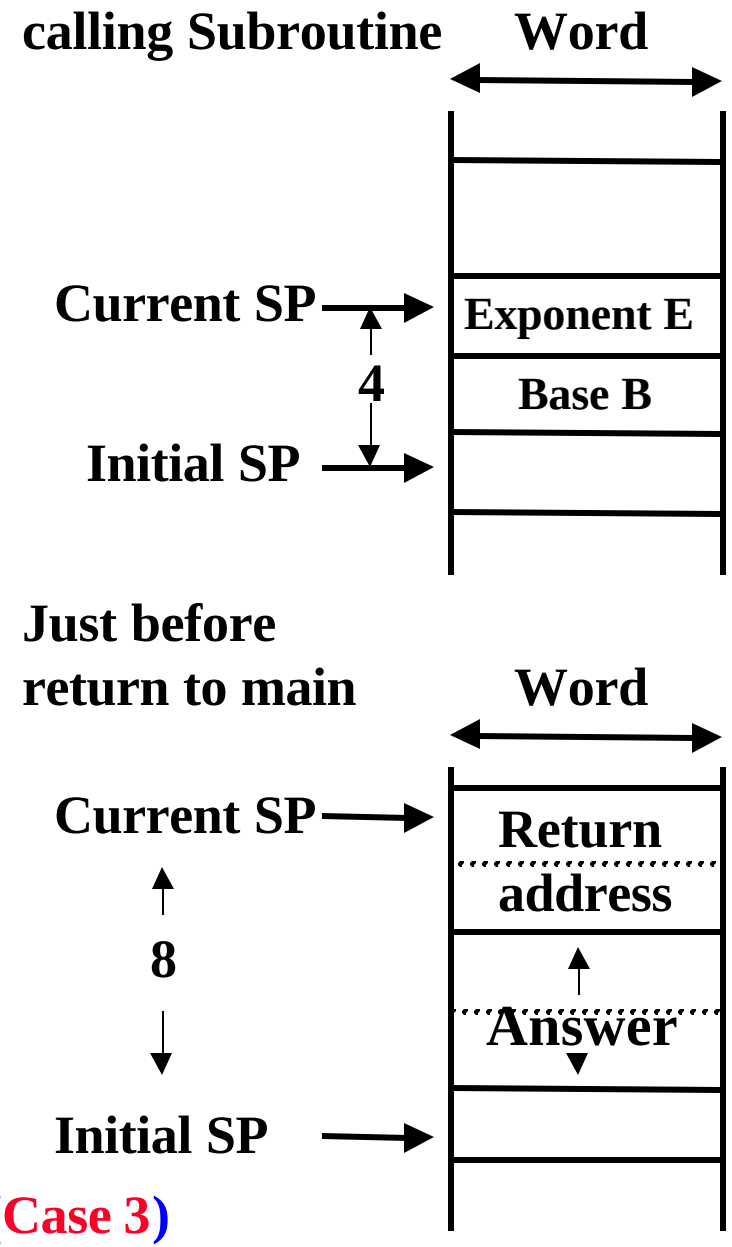
```
MAIN    ORG         $400            Main Program origin
        MOVEA.L     #$07FFE,SP      Initialize  Stack Pointer
        MOVE.B      B,D1            Put base number into D1
        EXT.W       D1              Sign extend base to word length
        MOVE.W      D1,-(SP)        push base  B onto the stack
        CLR.W       D2              Clear D2 before loading exponent
        MOVE.B      E,D2            Put exponent number into D2
        MOVE.W      D2,-(SP)        push exponent E onto the stack
        BSR         POWER           Call subroutine  POWER
        MOVE.L      (SP)+,D3        pop answer from stack resetting SP
        LEA         A,A5             put address of  answer into A5
        MOVE.L      D3,(A5)         save  answer
        STOP        #$2700          Done
        ORG         $600
B       DC.B        4               Base number stored here
E       DC.B        2               Exponent number stored here
A       DS.L        1               answer to be stored  here
```

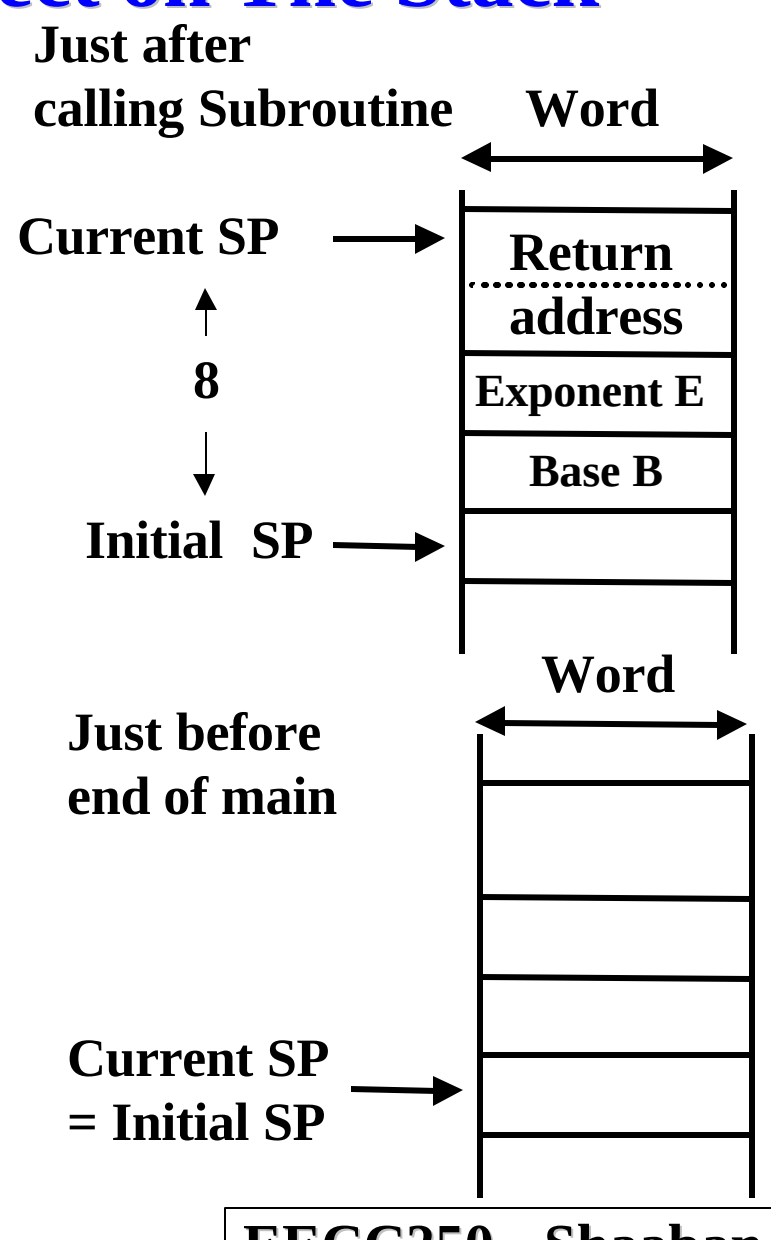
POWER Subroutine Example (Case 3)

Parameter Passing *by Value*: Using The Stack
Continued - Subroutine -

```
        ORG     $800        Subroutine  POWER origin
POWER   MOVE.W  6(SP),D1    copy base  from stack to D1
        CLR.W   D2          Clear D2 before loading exponent
        MOVE.B  4(SP),D2    copy exponent from to D2
        MOVE.L  #1,D3       initialize result in D3 to 1
LOOP    MULS    D1,D3       multiply result D3 with base D1
        SUB     #1,D2       decrement power in D2 by one
        BNE     LOOP        and repeat as long as power > 0
        MOVE.L  D3,4(SP)    Push result onto the stack
        RTS                 Done, return to calling program
```

Effect on The Stack





POWER Subroutine Example (Case 4)

Parameter Passing *by Reference*: Using The Stack

- Main Program -

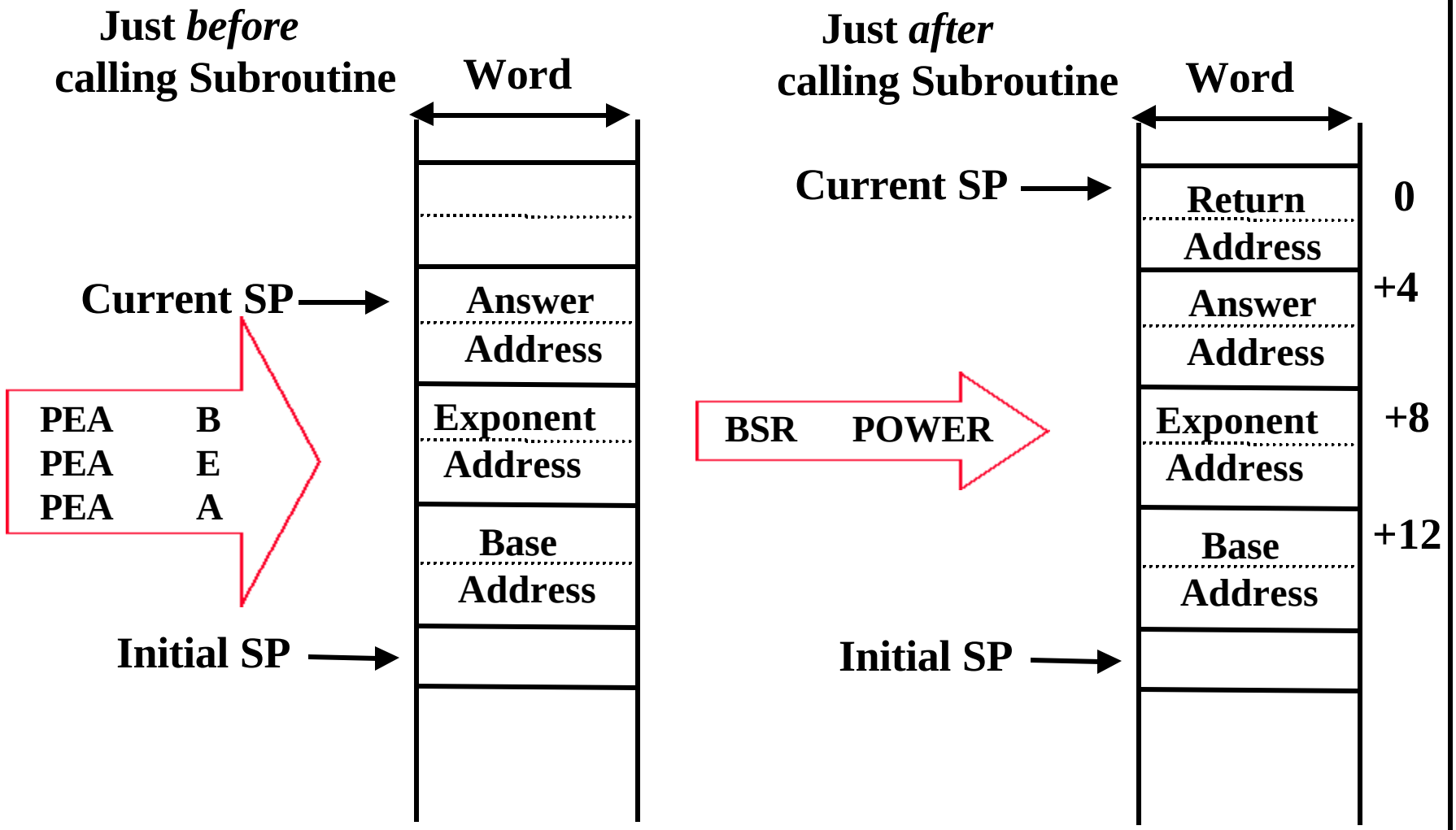
```
MAIN    ORG         $400            Main Program origin
        MOVEA.L     #$07FFE,SP      Initialize  Stack Pointer
        PEA         B               Push address of Base  onto the stack
        PEA         E               Push address of Exponent  onto the stack
        PEA         A               Push address of Answer  onto the stack
        BSR         POWER           Call subroutine  POWER
        LEA         12(SP),SP       Stack clean-up:  stack pointer reset
        STOP        #$2700          Done
        ORG         $600
B       DC.B        4               Base number stored here
E       DC.B        2               Exponent number stored here
A       DS.L        1               answer to be stored  here
```

POWER Subroutine Example (Case 4)

Parameter Passing *by Reference*: Using The Stack
Continued - Subroutine -

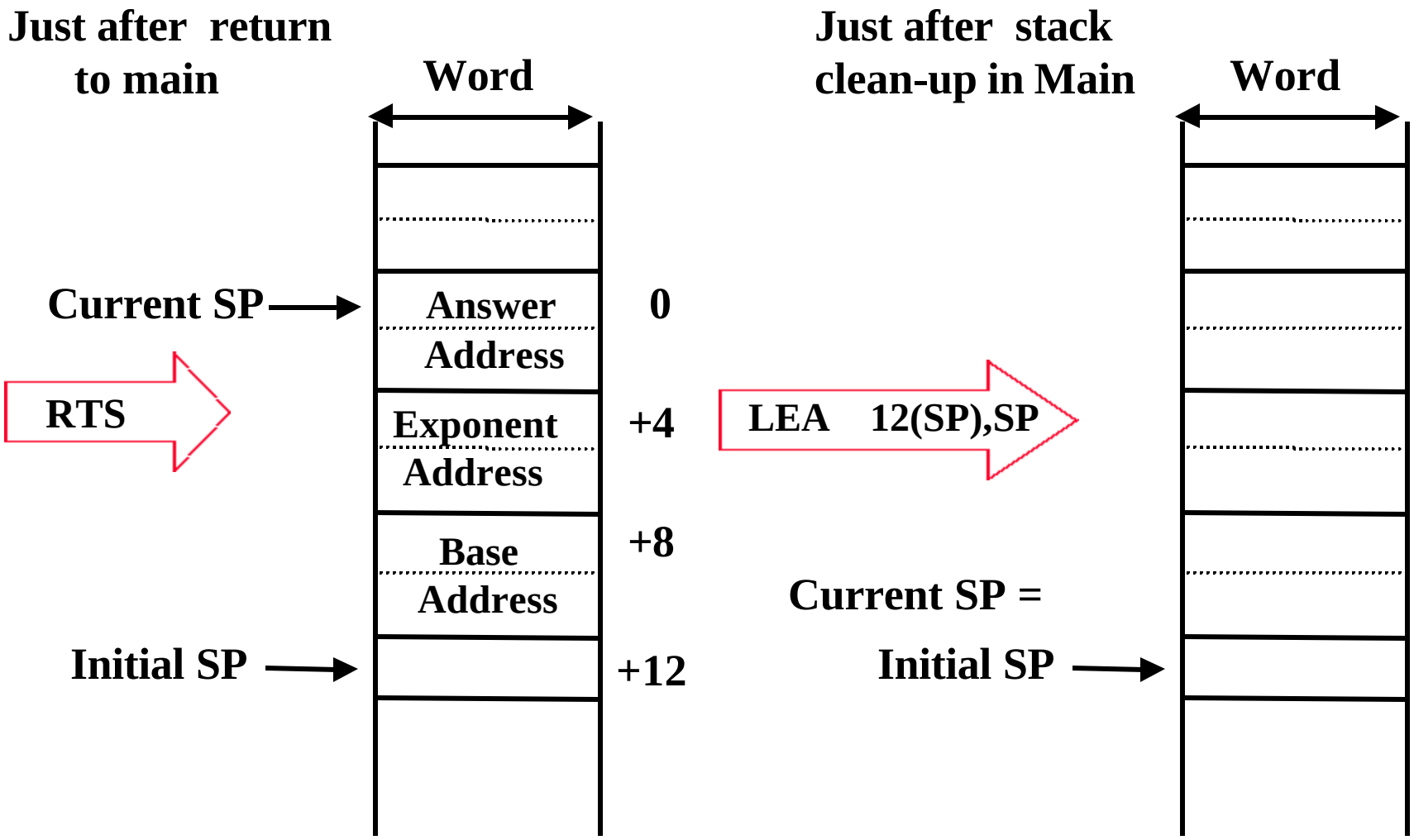
```
        ORG     $800        Subroutine  POWER origin
POWER   MOVEA.L 12(SP),A1   load Base address  in A1
        MOVEA.L 8(SP),A2    load Exponent address  in A2
        MOVEA.L 4(SP),A3    load Answer address address in A3
        MOVE.B  (A1),D1     Put base number into D1
        EXT.W   D1          Sign extend base to word length
        CLR.W   D2          Clear D2 before loading exponent
        MOVE.B  (A2),D2     copy exponent from to D2
        MOVE.L  #1,D3       initialize result in D3 to 1
LOOP    MULS    D1,D3       multiply result D3 with base D1
        SUB     #1,D2       decrement power in D2 by one
        BNE     LOOP        and repeat as long as power  > 0
        MOVE.L  D3,(A3)     Save result in memory
        RTS                 Done, return to calling program
```

Effect on The Stack



(Case 4)

Effect on The Stack



(Case 4)