Passing Parameters

Data passed to a subroutine is called a parameter.

There are two classes of parameters:

- in (call by value)
  - Original data does not change
  - Copy of original data is used by subroutine
  - Copy may be modified
- in-out (call by reference)
  - Original data can be modified
  - Location of data is passed

Parameters are passed using:

- Registers (data registers used to pass by value)
- Parameter (Memory) block (address register used to pass address of the block)
  - Block may be just one location or a block of memory locations
- Stack (stack can be both used to pass by value, or by reference)

Passing Parameters

Calling Program

```
        move    N,R1            ;R1 serves as a counter – used to pass by value
        move    #NUM1,R2        ; R2 points to the list – used to pass by reference
                                ; address of the first number on the list
        Call    LISTADD         ; call the subroutine
        move    R0,SUM; save result
        ……
```

Subroutine

```
LISTADD     Clear       R0          ; initialize sum to zero
LOOP        Add         (R2)+,R0        ; Add entry from list
            Decrement   R1
            Branch > 0  LOOP
            Return
```

(Fig 2.25 of Hamacher)

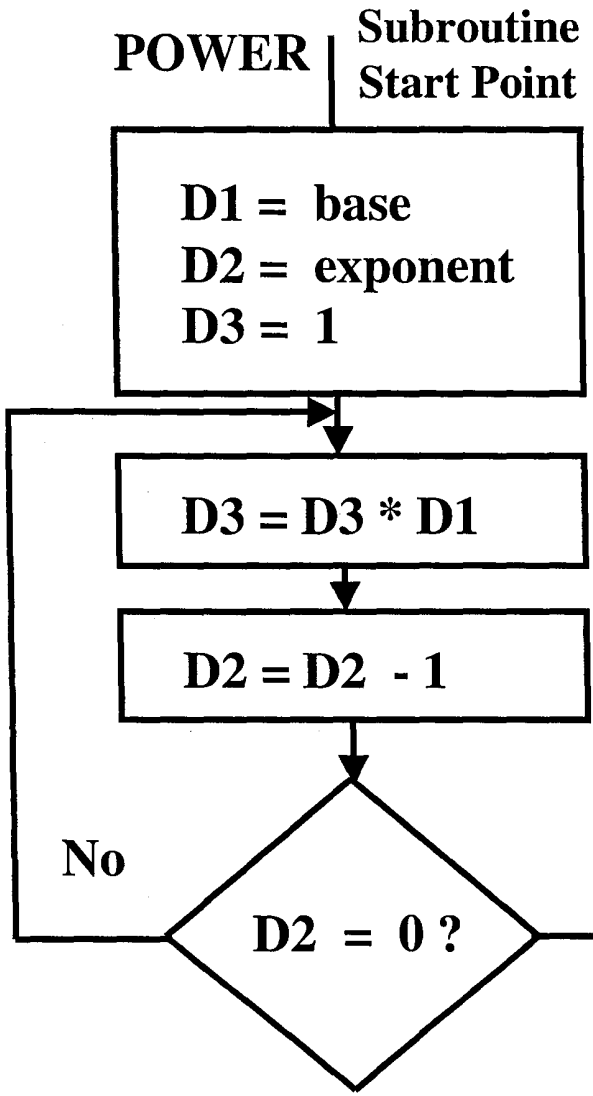
Example: Power Calculation Subroutine

- 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

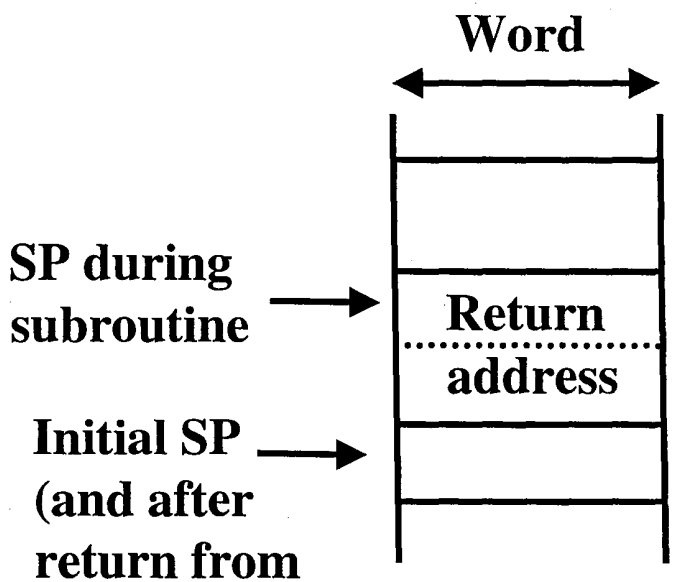
Functional specification (pseudo code) of subroutine POWER:

```
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
	MOVE	#228,D7	Done
	TRAP	#14	
	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

Here D1 and D2 are not true "in" parameters since their values are modified by subroutine

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
        MOVE        #228,D7         Done
        TRAP        #14
        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
```

Example of Passing Parameters by parameter block

; Parameter Blocks have advantage when the number of parameters to be passed is large. If all registers were used for parameter passing, subroutine will have no registers to work with

```
;Main Program
main        equ         *
            ….
            lea           block, a0          ; move address of parameter block to a0
            move.l        #string, (a0)      ; move string address to block
            jsr           convert            ; call subroutine
                        …..                  ; result is in the block (four bytes offset)
                        …..
                        …..                  ; end of code
; Subroutine convert
convert     equ         *
            movea.l     block, a2            ; copy string pointer to a2
            …..
            …..                              ; assume result is in d0
            move.w      d0, 4(a0)            ; save the result in the block
            rts                              ; return
; Data area
;
string      ds.b        20                   ; storage for string
block       ds.l        1                    ; first parameter: base address of the string
            ds.w        1                    ; second parameter: resulted signed integer
            end
```