PARAMETER PASSING METHODS

How do you pass parameters between the subroutines (or interrupts) and the main routine or other subroutines?

1. Pass the parameter(s) (data or pointer) in the internal registers.

2. Pass the parameter(s) immediately after the call instruction, i.e. in the program memory space. (This requires that the parameter(s) be fixed at assembler time.)

3. Pass a pointer to the location of the parameter(s) immediately after the call instruction.

4. Pass the parameter(s) on the stack prior to the call. (PSH)

5. Pass a pointer to parameter(s) on the stack prior to the call. (PSH)

The Problem: Find the average of two numbers

Solution 1: Pass the parameter(s) in the internal registers.

Solution 1a: Pass the parameter *data* in the internal registers.

```
        ORG     $B600       ;Start program at $B600
START:  LDS     #$0041      ;initialize stack pointer
        LDAA    #$37        ;Load data in the A and B
        LDAB    #$A3        ;   registers
        JSR     AVG         ;Call the subroutine AVG to
                            ;  get average
        ...
*****************************************************
* Get average of inputs in accumulators A and B
* Output in accumulator A

AVG:    ABA                 ;A=A+B
        ASRA                ;Shift A right by 1 bit
                            ;  (divide by 2) keeping
                            ;  bit 7 (for sign
                            ;  extension)
        RTS
```

Solution 1b: Pass the parameters *addresses* in internal pointer registers.

```
        ORG     $0000
DATA:   FCB     $37, $A3

        ORG     $B600       ;Start program at $B600
START:  LDS     #$0041      ;initialize stack pointer
        LDX     #DATA       ;Load X index reg. with
                            ;   address of data
        JSR     AVG1        ;Call the subroutine AVG1
                            ;  to get average
        ...
*************************************************
* Get average of two data bytes in successive memory
* starting at location pointed to by X; Output in A
```

```
AVG1:   LDAA    0,X         ;A = 1st piece of data
        ADDA    1,X         ;A=A + 2nd piece of data
        ASRA                ;Divide by 2
        RTS
```

Solution 2

Pass the parameter(s) immediately after the call instruction, i.e., in the program memory space. (This requires that the parameter(s) be fixed at assemble time.)

Since data follows the call, the return address pushed on the top of the stack by the subroutine call must be corrected before returning from the subroutine.

```
        ORG     $B600       ;Start program at $B600
START:  LDS     #$0041      ;initialize stack pointer
        JSR     AVG2        ;Call the subroutine AVG2
DATA1:  FCB     $37
DATA2:  FCB     $A3
NEXT:   ...
*************************************************
* Get average of two data bytes in program memory at
* location pointed to by data on top of stack
* Output in A
* X affected
* Stack return address corrected
        ORG     $B700
AVG2:   PULX                ;Get address of data
        LDAA    0,X         ;Get first piece of data
                            ;  (at DATA1) into A
        ADDA    1,X         ;A=A + (data at DATA2)
        ASRA                ;Divide by 2;Result in A
        INX                 ;Increment the X index
        INX                 ;  register twice to point
                            ;  to next instruction upon
                            ;  RTS
        PSHX                ;Push the corrected address
                            ;  back on stack
        RTS
```

LABEL	Address	Value
START	$B600	$8E
	$B601	$00
	$B602	$41
	$B603	$BD
	$B604	$B7
	$B605	$00
DATA1	$B606	$37
DATA2	$B607	$A3
NEXT	$B608	…

$003B	
$003C	
$003D	
$003E	
$003F	
$0040	$B6 (Data1 Hi)
$0041	$06 (Data1 Lo)

Solution 3

Pass a pointer to the location of the parameter(s) immediately after the call instruction.

Similar to Solution 2 except now a pointer to the data is passed (instead of the data itself), so that the data does not have to be know at assemble time.

```
        ORG     $0001
LOCAT:  FCB     $37, $A3    ;Define location of data
                            ;  and some default data

        ORG     $B600       ;Start program at $B600
START:  LDS     #$0041      ;initialize stack pointer
        JSR     AVG3        ;Call the subroutine AVG3
DAT_AD: FDB     LOCAT       ;Define the location of
                            ;data
NEXT:   ...
*************************************************
* Get average of two data bytes in program memory at
* location pointed to by pointer on top of stack
* Output in A
* X affected
* Stack return address corrected

        ORG     $B700
AVG3:   PULX                ;Get address of pointer to
                            ;  data
        INX                 ;Increment the X index
        INX                 ;  register twice to point
                            ;   to next instruction
                            ;   upon RTS
        PSHX                ;Push the corrected address
                            ;  back on stack
        DECX                ;Make X point to pointer at
        DECX                ;  DAT_AD
        LDX     0,X         ;Put the pointer into X
        LDAA    0,X         ;Get first piece of data
        ADDA    1,X         ;A=A + (data at DATA2)
        ASRA                ;Divide by 2
RTS
```

LABEL	Address	Value
START	$B600	$8E
	$B601	$00
	$B602	$41
	$B603	$BD
	$B604	$B7
	$B605	$00
DATA_AD	$B606	$00
DATA2	$B607	$01
NEXT	$B608	…

$003B	
$003C	
$003D	
$003E	
$003F	
$0040	$B6 (DAT_AD Hi)
$0041	$06 (DAT_AD Lo)

Solution 4

Pass the parameter(s) on the stack prior to the call. (PSH)

```
        ORG     $B600       ;Start program at $B600
START:  LDS     #$0041      ;initialize stack pointer
        LDAA    #$37        ;Load data onto stack
        PSHA
        LDAA    #$A3
        PSHA
        JSR     AVG4A       ;Call the subroutine AVG4A/AVG4B
        ...
*****************************************************
* Get the average of the inputs on the stack
* Output in accumulator A
* A,B,X affected
* Stack modified
        ORG     $B700
AVG4A:  PULX                ;Save return address
        PULA                ;Get second piece of data
        PULB
        ABA                 ;A=A+B
        ASRA                ;Divide by 2
        PSHX                ;Fix stack for return
                            ;  address
        RTS
```

```
*****************************************************
* Get the average of the inputs on the stack
* Output in accumulator A
* A,X affected
* Stack unmodified
        ORG     $B700
AVG4B:  TSX                 ;IX = SP+1 & IX points to the stack
        LDAA    2,X         ;A = 1st parameter
        ADDA    3,X         ;A = A + 2nd parameter
        ASRA                ;Divide by 2
        RTS
```

Solution 5

Pass the address of the parameter(s) on the stack prior to the call. (PSH)

```
        ORG     $0001
LOCAT1: FCB     $37         ;Define location of data
        ORG     0006H
LOCAT2: FCB     $A3         ;  and some default data
        ORG     $B600       ;Start program at $B600

START:  LDS     #$0041      ;initialize stack pointer
        LDX     #LOCAT2     ;Put location of 2nd data
        PSHX                ;  onto stack
        LDX     #LOCAT1     ;Put location of 1st data
        PSHX                ;  onto stack
        JSR     AVG5A       ;Call the subroutine AVG5A/5B
        ...
*******************************************************
* Get average of numbers pointed to by addresses
*   on stack
* Output in accumulator A
* A,B,X,Y affected; stack modified
        ORG     $B700
AVG5A   PULX                ;Save return address
        PULY                ;Get address of 1st data
        LDAA    0,Y         ;Put first data in A
        PULY                ;Get address of 2nd data
        ADDA    0,Y         ;A=A + 2nd data
        ASRA                ;Divide by 2
        PSHX                ;Fix stack for return add.
        RTS
```

```
*******************************************************
* Get average of numbers pointed to by addresses
*   on stack
* Output in accumulator A
* A,B,X,Y affected; stack unmodified
        ORG     $B700
AVG5B:  TSX                 ;IX = SP+1 & IX points to the stack
        LDY     2,X         ;Get address of 1st data
        LDAA    0,Y         ;Put first data in A
        LDY     4,X         ;Get address of 1st data
        ADDA    0,Y         ;A=A + 2nd data
        ASRA                ;Divide by 2
        RTS
```

- What if you wanted the subroutine to have no effect on any registers?
- Can you think of any other ways to pass data?
  How about using RAM for variables!