floating point or string values. You also may not pass more than three integer values with SCALL. If you say (in Extended XYBASIC):

```
SCALL #8000, I%, J%, K%
```

then XYBASIC will first load registers BC, DE and HL with the values of I%, J% and K%, and then execute an assembly language CALL to the routine at location 8000H. Executing an assembly language RETurn will return control to the next command after the SCALL, and the values in registers BC, DE and HL will be assigned to the variables I%, J% and K%.

As with CALL, the location of the machine language routine may be specified by any formula. However, the parameters of SCALL must be integer variables. The parameters are optional, but a MC (Machine language Call) error will occur if you try to pass more than three parameters, or if you use a non-integer variable as a parameter.

Note that the SCALLed routine must preserve the values in BC, DE and HL if you wish to leave the parameter values unchanged! If you need to use an assembly language routine which does not preserve registers, you can assign the parameters to temporary variables first. In the above example you could for example say:

```
LET ITEMP% = I%
LET JTEMP% = J%
LET KTEMP% = K%
SCALL #8000, ITEMP%, JTEMP%, KTEMP%
```

where ITEMP%, JTEMP% and KTEMP% are integer variables not used elsewhere in your XYBASIC program. The values of these temporaries may be altered by SCALL, but the values of I%, J% and K% will remain unchanged.

Of course the above examples should be written without the % signs in Integer XYBASIC, since only integer variables are allowed.

Example:
Archie Goodwin must write a program to control a high speed stepper motor and print positional information after its rotation is complete. The motor moves one step for each output to port 0, regardless of the value output, and accepts a new step pulse every millisecond. Archie could issue pulses to the motor with an OUT command inside a FOR/NEXT loop. However, his stepper is capable of a much higher step rate; XYBASIC is a high level interpretive language and cannot compete with machine language for speed. But the data manipulation required by the program is very difficult for Archie to perform in machine language.

Archie's solution is to code only the time critical section for motor control in machine language and write the rest of his task in XYBASIC. The following simple machine language routine provides the needed speed.

```
FC00                    ORG     0FC00H
FC00 78         STEPN:  MOV     A,B     ;NUMBER OF STEPS IS IN BC
FC01 B1                 ORA     C       ;ZERO STEPS LEFT?
FC02 C8                 RZ              ;YES - RETURN TO XYBASIC
FC03 D300               OUT     STEP    ;NO - SEND STEP PULSE
FC05 3E83               MVI     A,83H   ;TIMING CONSTANT FOR 1MS
```