

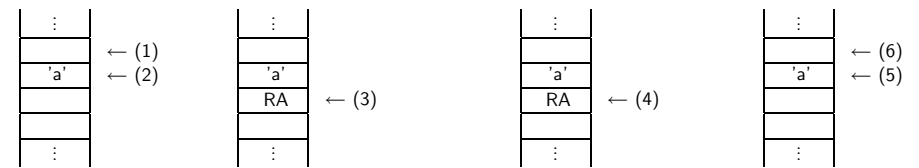
Procedure call – Example without local variables (1)

```

ch    = -1 ; void Write(char ch) {
Write:
(3)  load  !ch ; printf("%c",ch); /* AC=Mem[SP-1] */
      trap #3 ; /* write AC to stdout */
(4)  ret   ; } /* PC=Mem[SP]; SP=SP-1 */
----- Procedure above / Main program below -----
; Write('a');
(1)  load  #'a' ; /* Push('#a') */ /* AC='a' */
      incsp #1 ; /* SP=SP+1 */ /* SP=SP+1 */
      store !0 ; /* Mem[SP]=AC */
(2)  call  Write ; /* SP=SP+1; Mem[SP]=PC;
(5)  ; /* PC=Write */
      incsp #-1 ; /* Pop(1) */ /* SP=SP-1 */
(6)

```

Procedure call – Example without local variables (2)



call (Write) ret
 The numbers in brackets – like (1) – show the position of the stack pointer (SP) at the corresponding position in the program.

Non-void function – Example with one return value (1)

```

RV    = -3 ; int Log10(
x     = -2 ; unsigned int x) {
      ; /* What is at offset -1 ? */
Logx  = 0 ; unsigned int Logx;
NLocs = 1 ;
Log10:
(4)  incsp #NLocs ; /* SP=SP+1 */
(5)  load  !x ; if (x != 0) { /* AC=Mem[SP-2] */
      cmp  #0 ; /* compare AC,0 */
      beq  Lendif ;
      ...
Lendif:
      load  !Logx ; return Logx; /* AC=Mem[SP] */
      store !RV ; /* Mem[SP-3]=AC */
(6)  incsp #-NLocs ; } /*Log10()*/ /* SP=SP-1 */
(7)  ret   ; /* PC=Mem[SP]; SP=SP-1 */
      ...

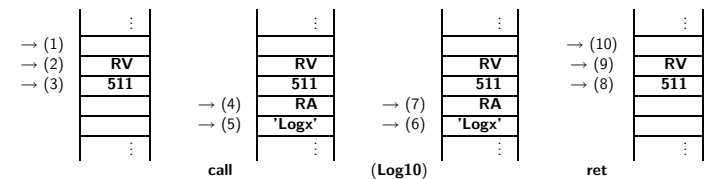
```

Non-void function – Example with one return value (2)

```

log:  block 1 ; int log;
(1)  incsp #1 ; log = Log10( 511 ); /* SP=SP+1 */ /* make RV slot */
(2)  Push  (#511) ; /* AC=511; SP=SP+1; Mem[SP]=AC */
(3)  call  Log10 ; /* SP=SP+1; Mem[SP]=PC; PC=Log10 */
(8)  Pop  (1) ; /* SP=SP-1 */
(9)  load  !0 ; /* AC=Mem[SP] */ /*store RV*/
      store log ; /* Mem[Log]=AC */
      incsp #-1 ; /* SP=SP-1 */ /*pop RV slot*/
(10)

```



Procedure call – Example with two local variables (1)

```

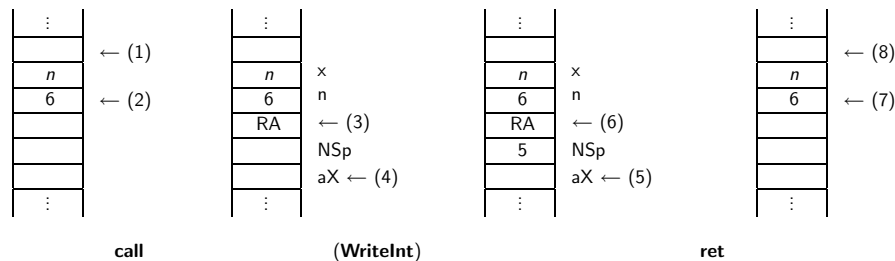
; void WriteInt(
x   = -4 ; int x,
n   = -3 ; unsigned int n) {
;
; /* what is at !-2? */
NSp = -1 ; int NSp; /* # of ' 's to print */
aX  = 0 ; unsigned int aX; /* = |x| */
Nlocs = 2 ;
WriteInt:
(3) incsp #Nlocs ; /* SP=SP+2 */
(4) load !n ; NSp = n - 1; /* AC=Mem[SP-3] */
sub #1 ; /* AC=AC-1 */
store !NSp ; /* Mem[SP-1]=AC */
(5) ...
incsp #-Nlocs ; } /* WriteInt() */ /* SP=SP-2 */
(6) ret ; /* PC=Mem[SP]; SP=SP-1 */
...
    
```

Procedure call – Example with two local variables (2)

```

This is the 'main program'
-----
...
; WriteInt(n, 6);
(1) load n ; /* Push(n) */ /* AC=Mem[n] */
incsp #1 ; /* SP=SP+1 */
store !0 ; /* Mem[SP]=AC */
load #6 ; /* Push(#6) */ /* AC=6 */
incsp #1 ; /* SP=SP+1 */
store !0 ; /* Mem[SP]=AC */
(2) call WriteInt; /* SP=SP+1; Mem[SP]=PC;
(7) ; PC=WriteInt */
incsp #-2 ; /* Pop(2) */ /* SP=SP-2 */
(8)
    
```

Procedure call – Example with two local variables (3)



The numbers in brackets – like (1) – show the position of the stack pointer (SP) at the corresponding position in the program.