F28HS2 Hardware-Software Interface

Lecture 9: ARM Assembly Language 4

Subroutines

- block of code with:
  - single entry point
  - ability to return to instruction after call
- basis of methods/procedures/functions
- subroutine call must:
  - remember address of instruction for return
  - transfer control to address of start of subroutine
- `R14` == `LR`
  - *link* register
  - holds return address

Subroutine call

`BL` *label*

- copy `PC+4` to `LR`
  - i.e. set `LR` to address following `BL` instruction
- branch to *label*
  - i.e. set PC to address for *label*
- so:
  - *label* is name of subroutine
  - after call, `LR` holds return address

Subroutine return

`BX` *label*

- branch indirect on *label*
- i.e. set *PC* to address from *label*

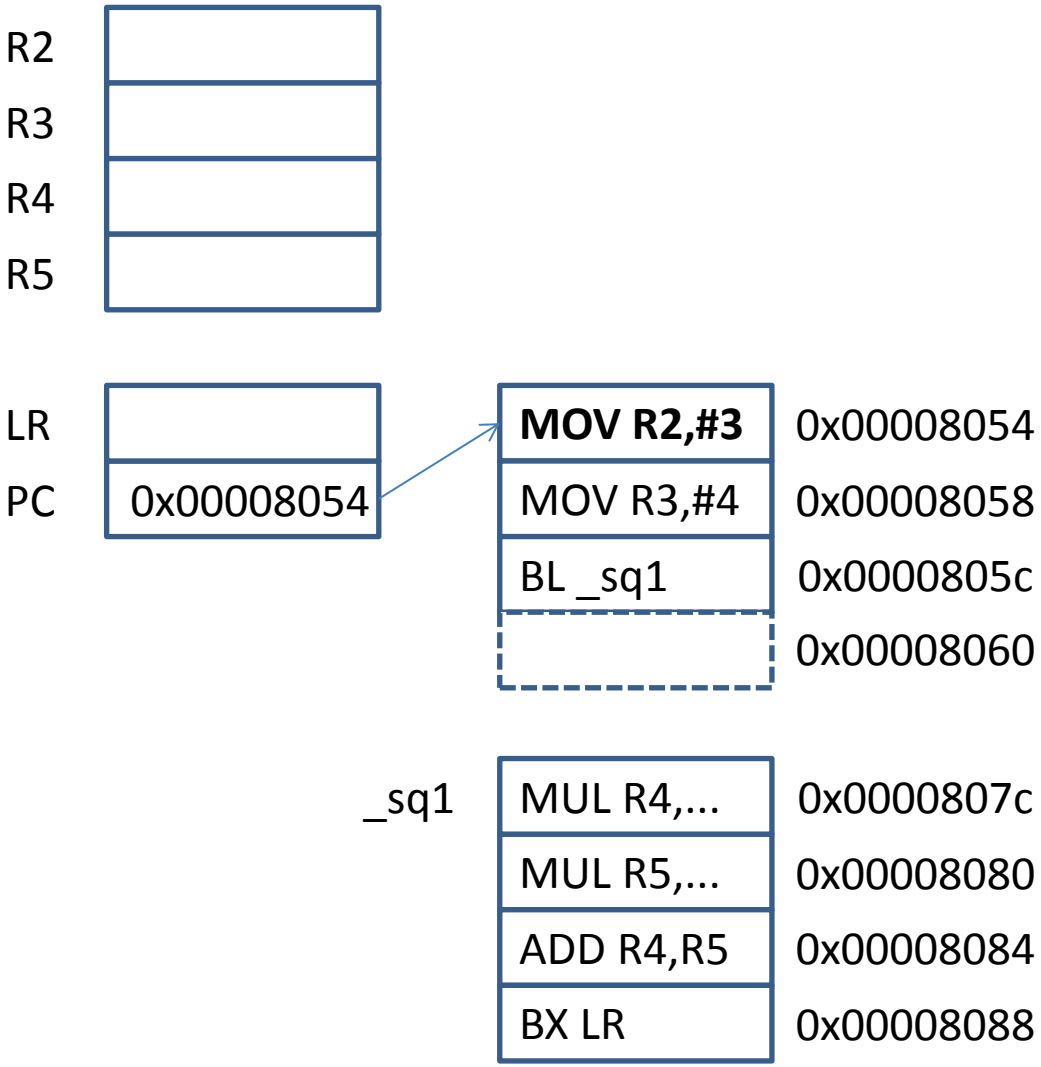
- so:
  - `BX LR` returns to instruction after last `BL`

Example: R4 = R2*R2+R3*R3

```
      MOV R2,#3
      MOV R3,#4
      BL _sq1
      ...
_sq1:
      MUL R4,R2,R2
      MUL R5,R3,R3
      ADD R4,R5
      BX LR
      ...
```

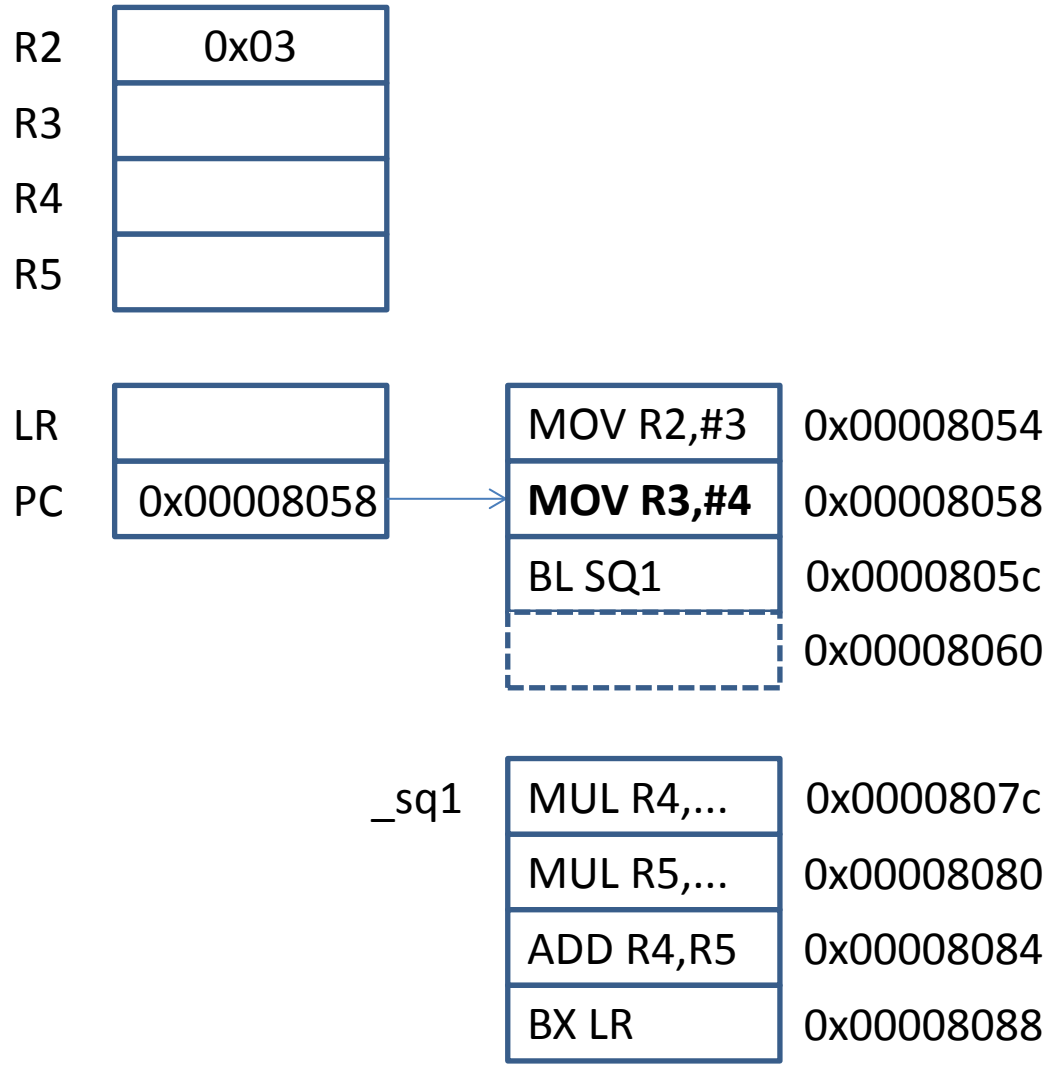
Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL _sq1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



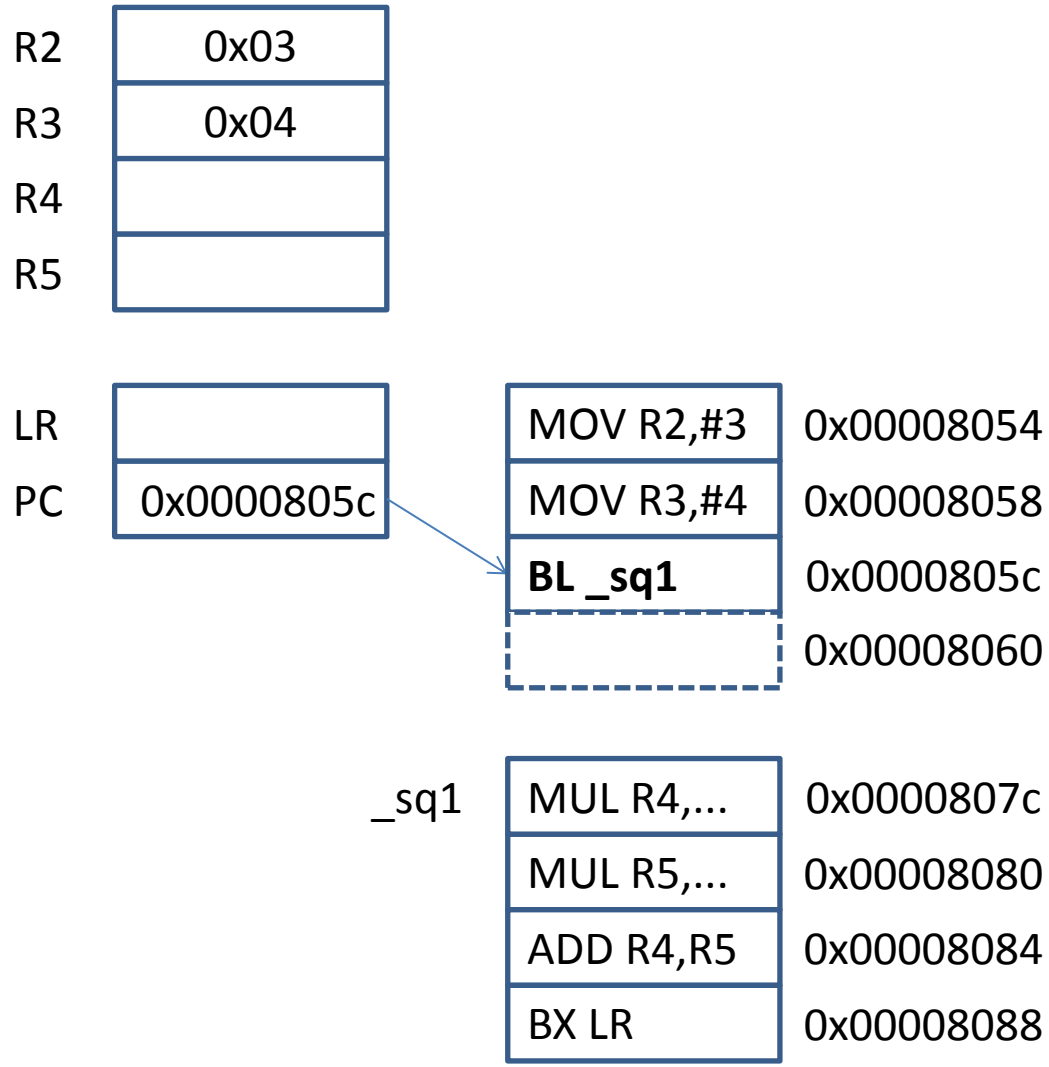
Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL SQ1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



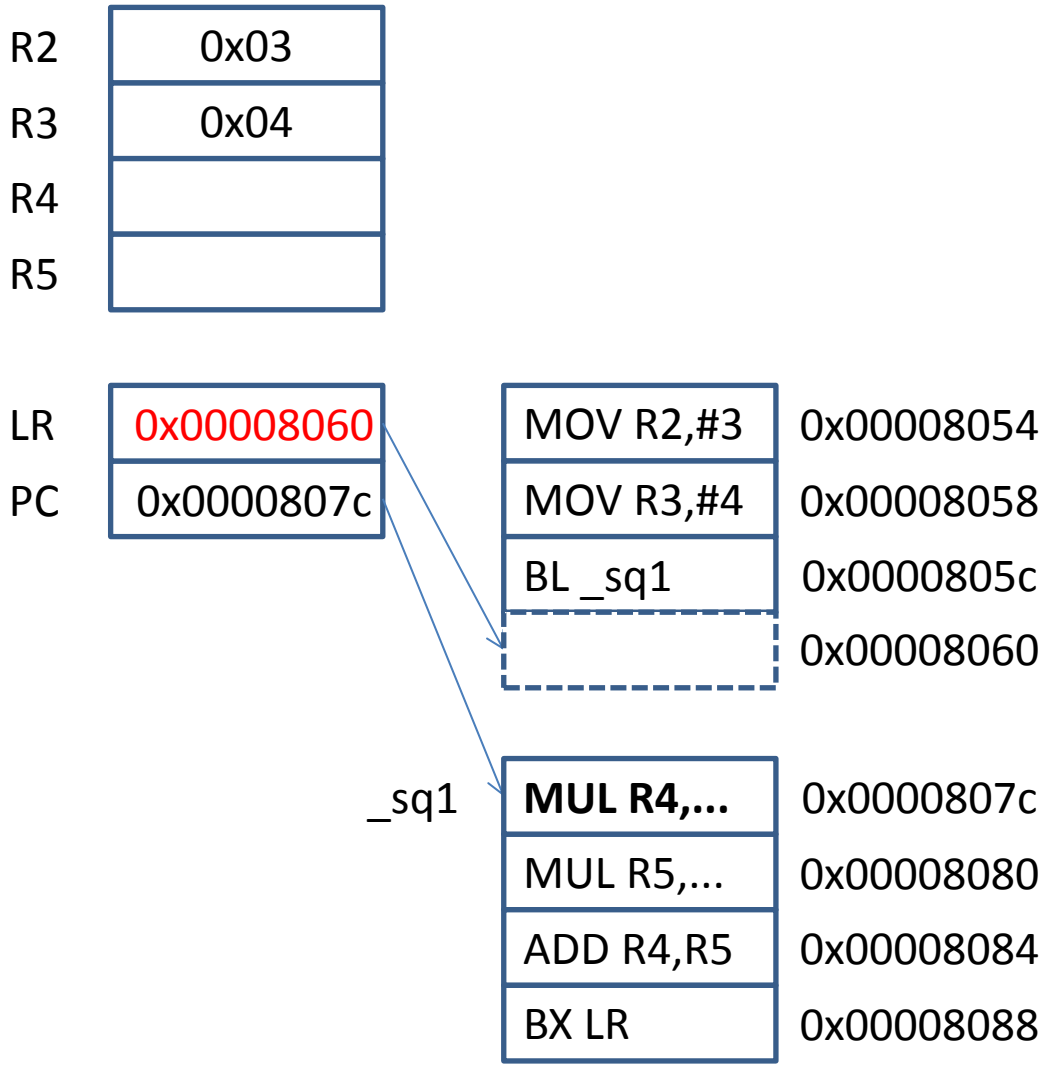
Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL _sq1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



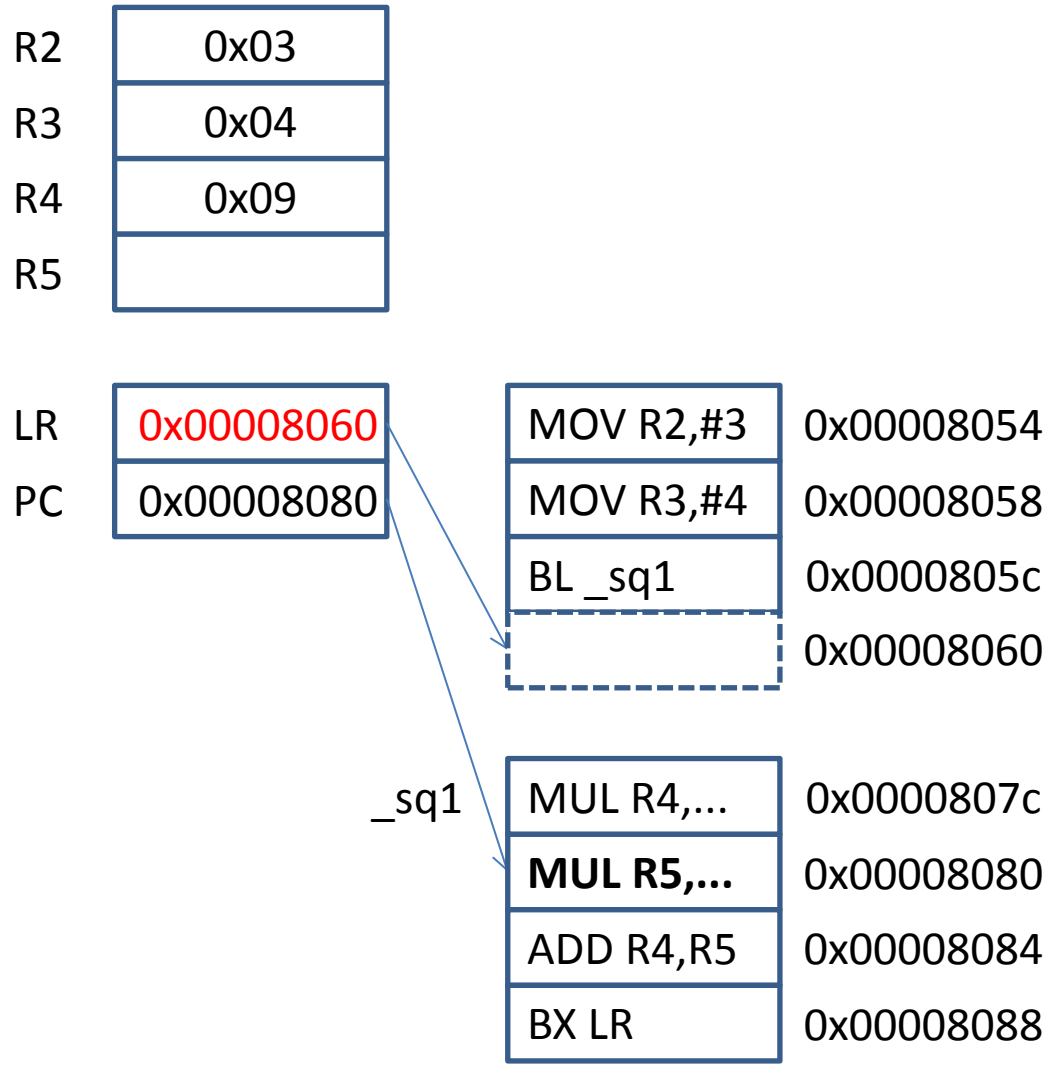
Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL _sq1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



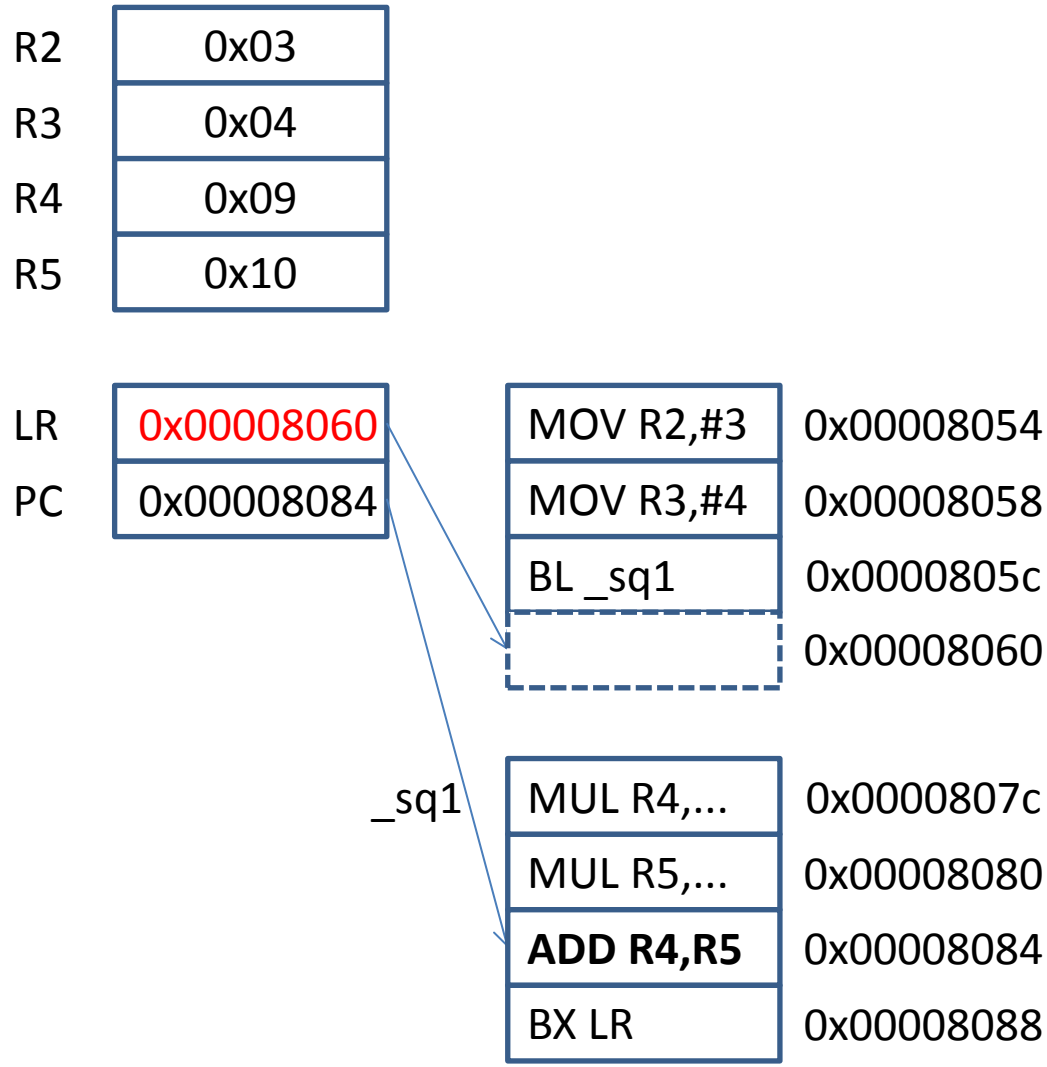
Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL _sq1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



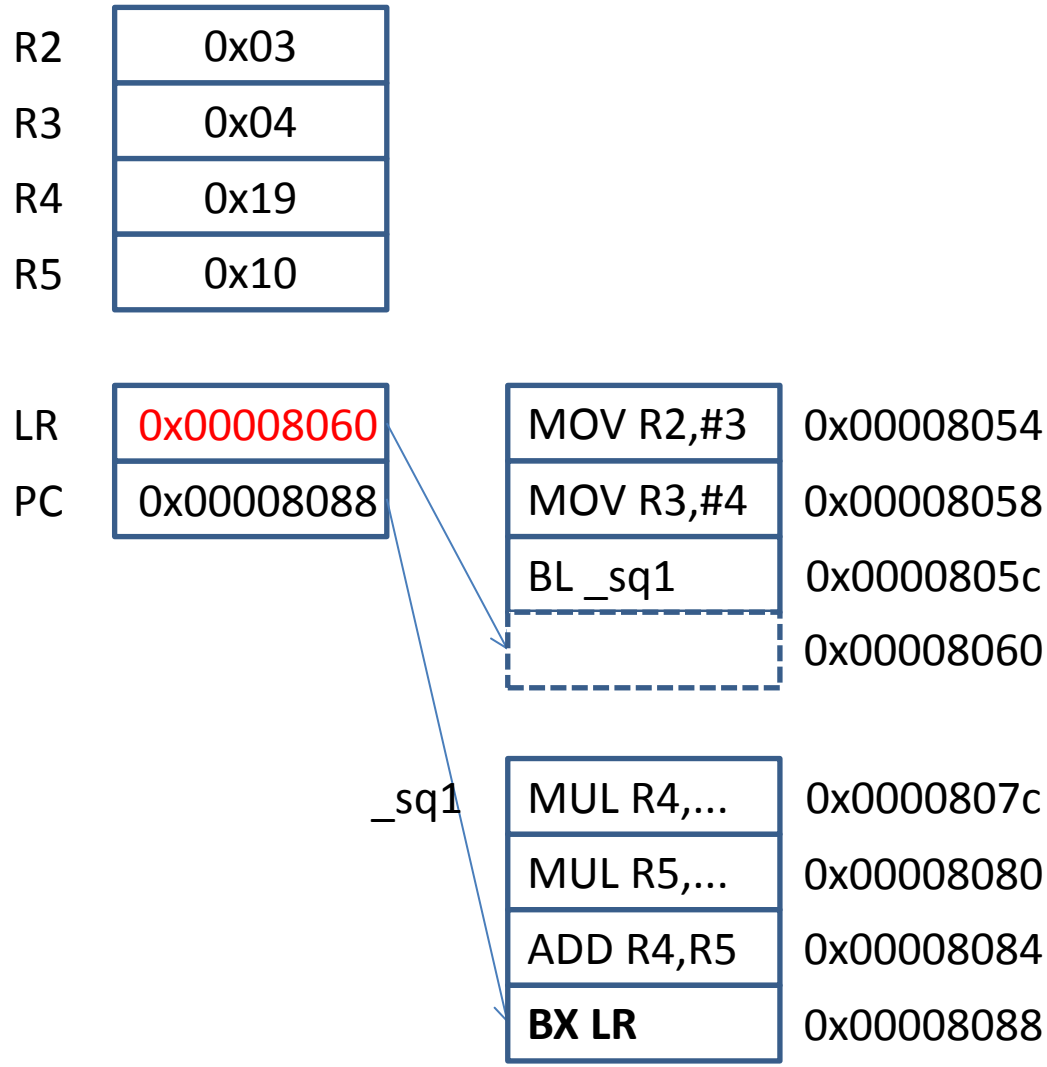
Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL _sq1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



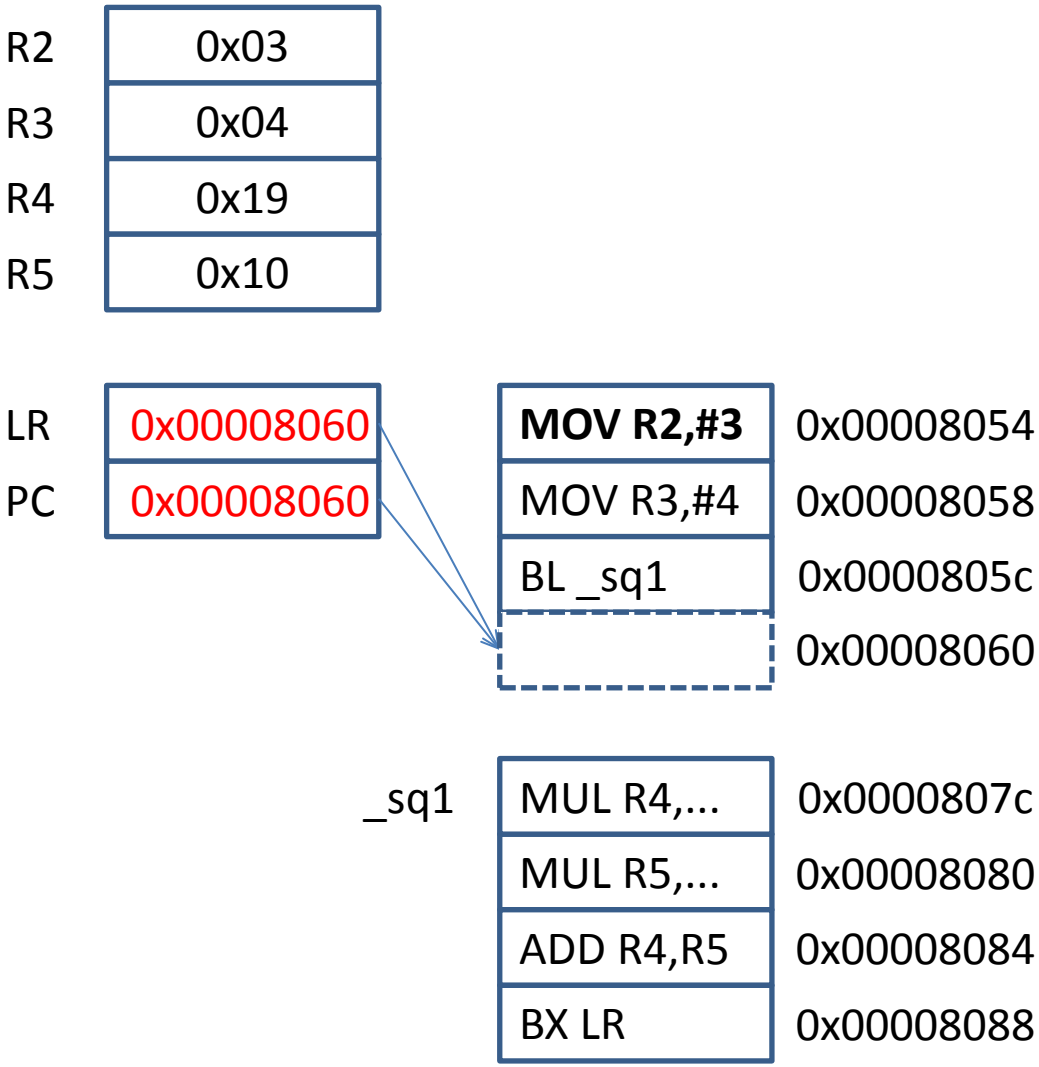
Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL _sq1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



Example: R4 = R2*R2+R3*R3

```
        MOV R2,#3
        MOV R3,#4
        BL _sq1
        ...
_sq1:   MUL R4,R2,R2
        MUL R5,R3,R3
        ADD R4,R5
        BX LR
        ...
```



Parameter passing

- registers
  - fast
  - ties up specific registers
  - may not have enough spare registers
- on stack
  - push from registers before call
  - pop into registers in subroutine
  - not register specific
  - slower

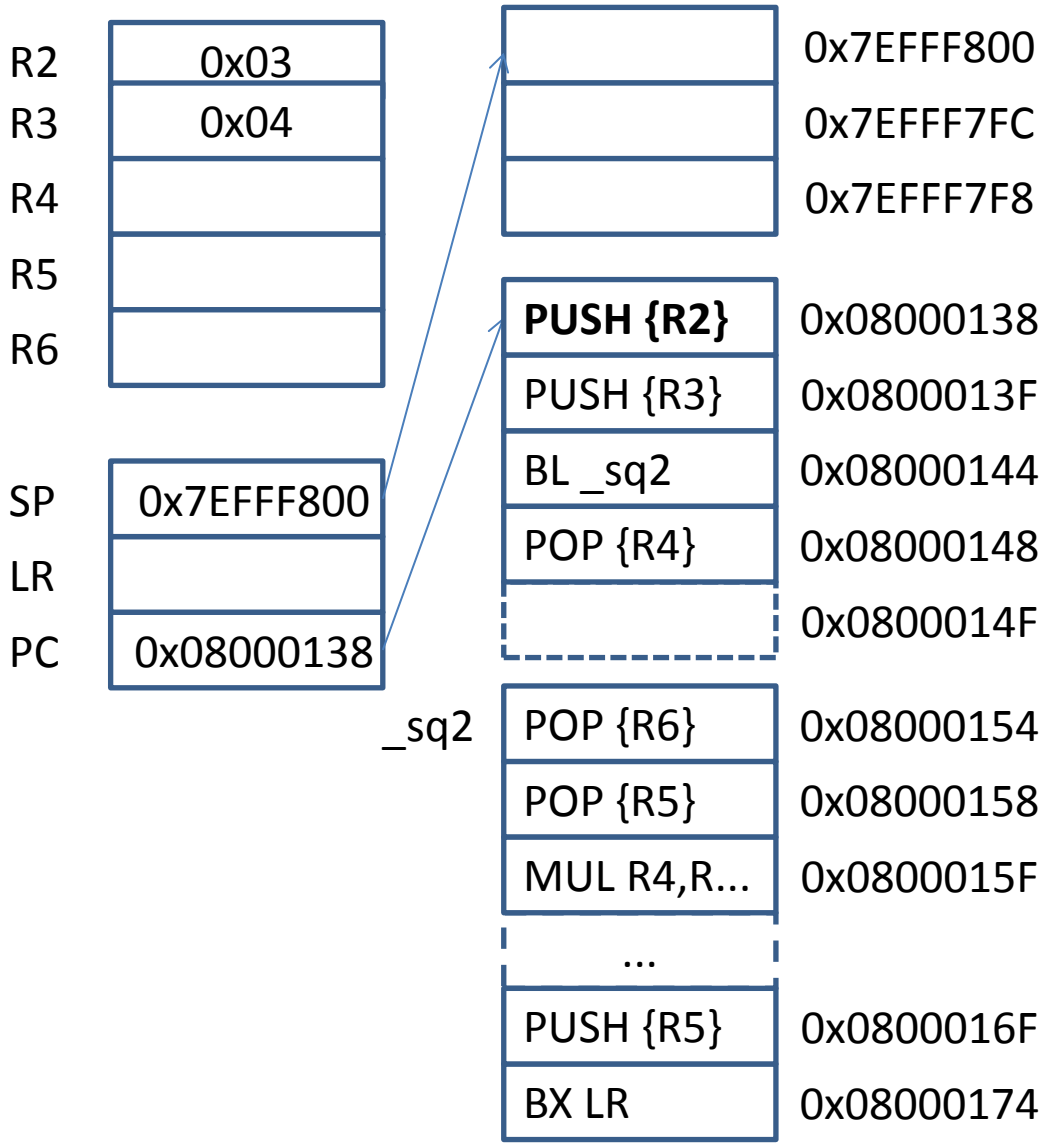
Example: stack parameter passing

```
PUSH {R2}
PUSH {R3}
BL SQ2
POP {R4}
...
```

```
_sq2:POP {R6}
     POP {R5}
     MUL R4,R5,R5
     MUL R5,R6,R6
     ADD R5,R4
     PUSH {R5}
     BX LR
     ...
```

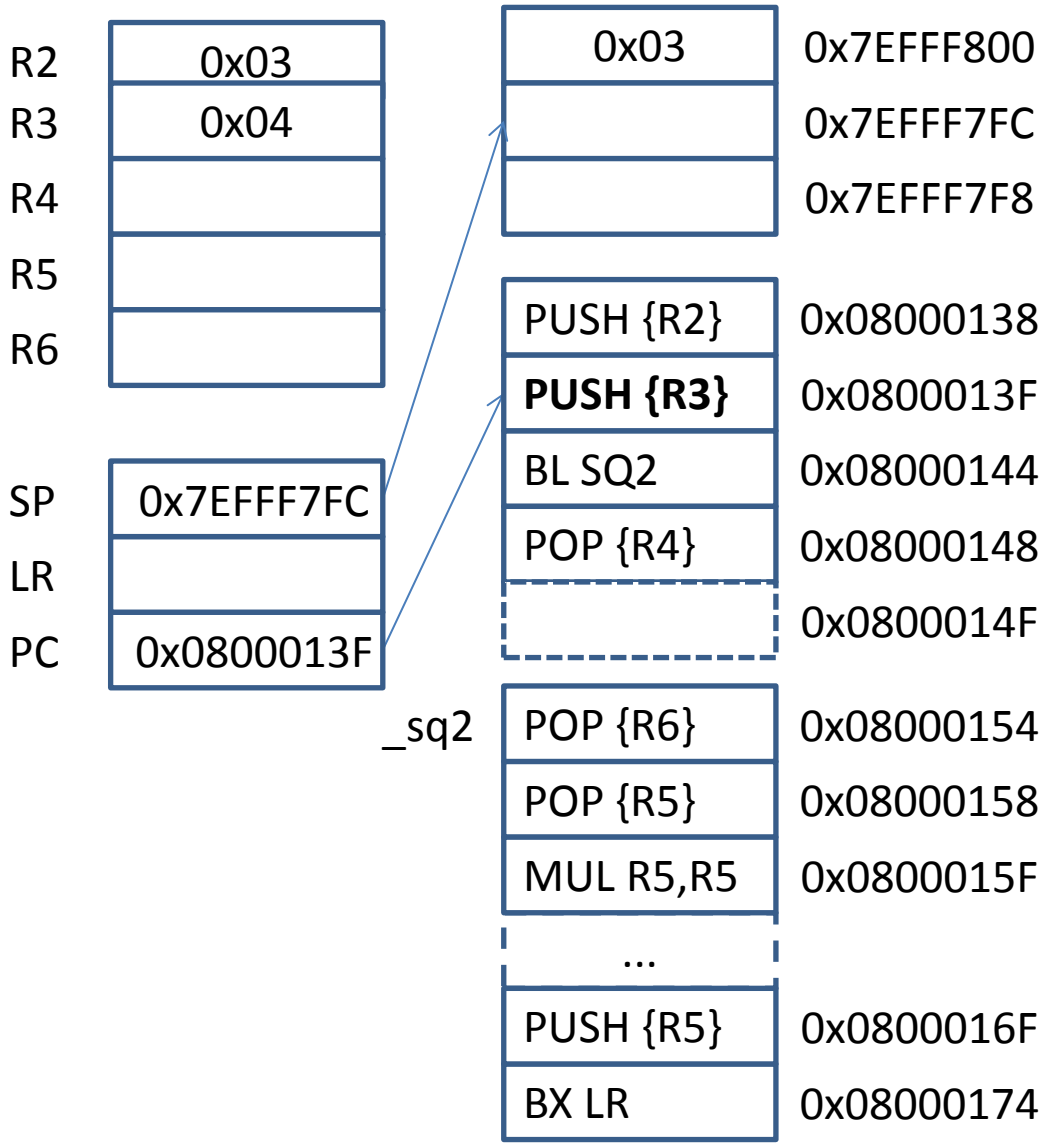
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



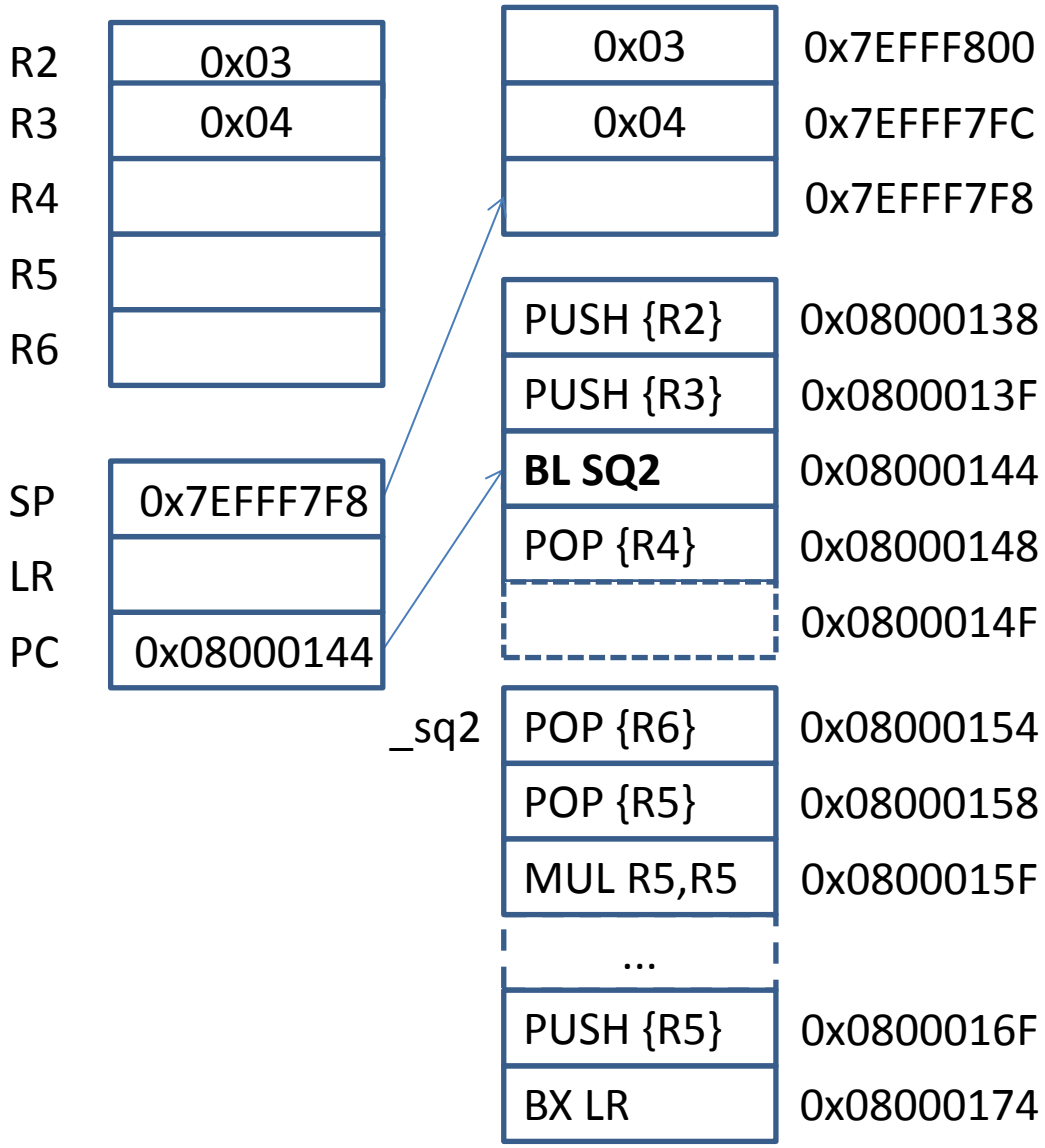
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



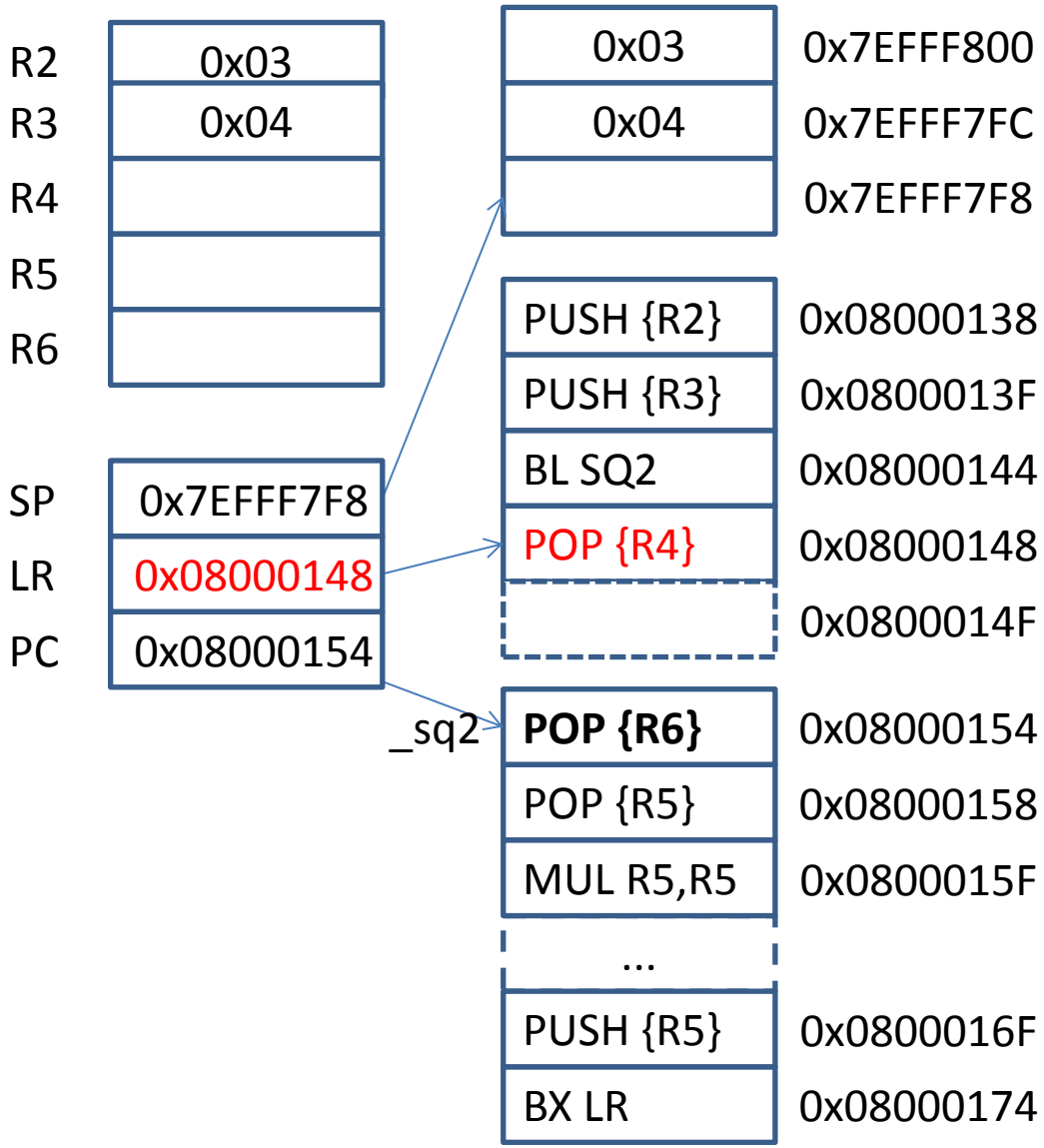
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



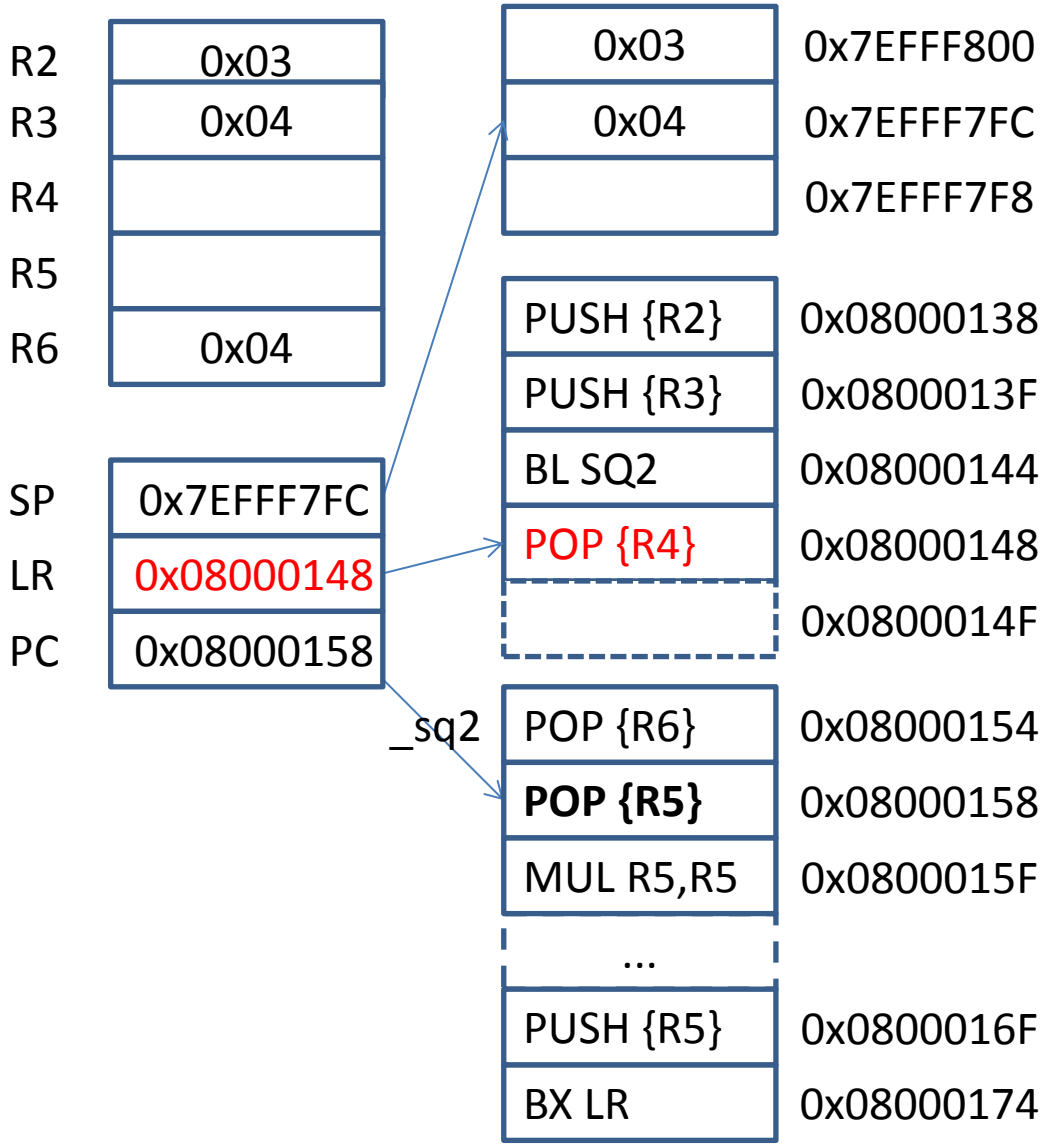
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



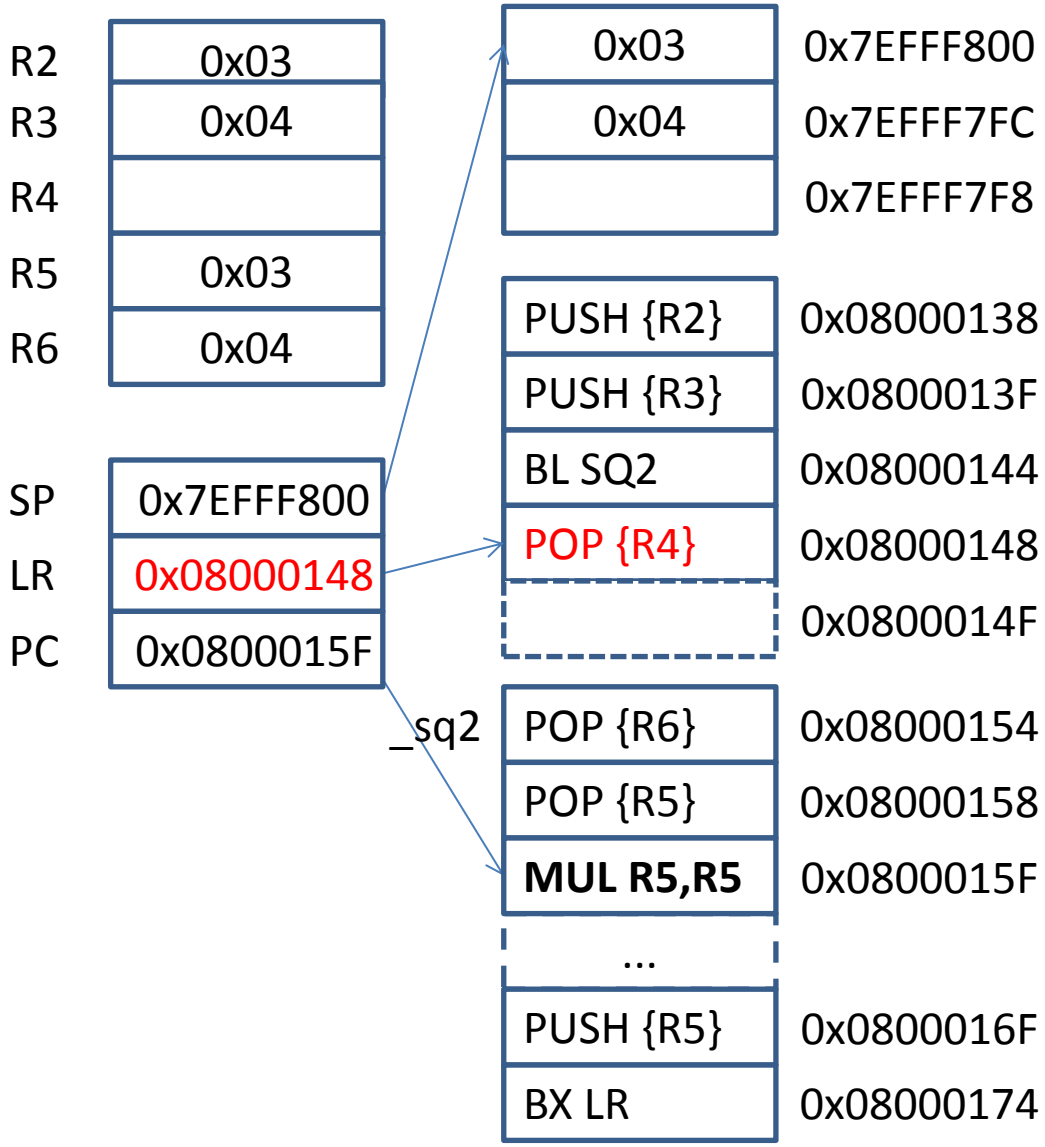
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



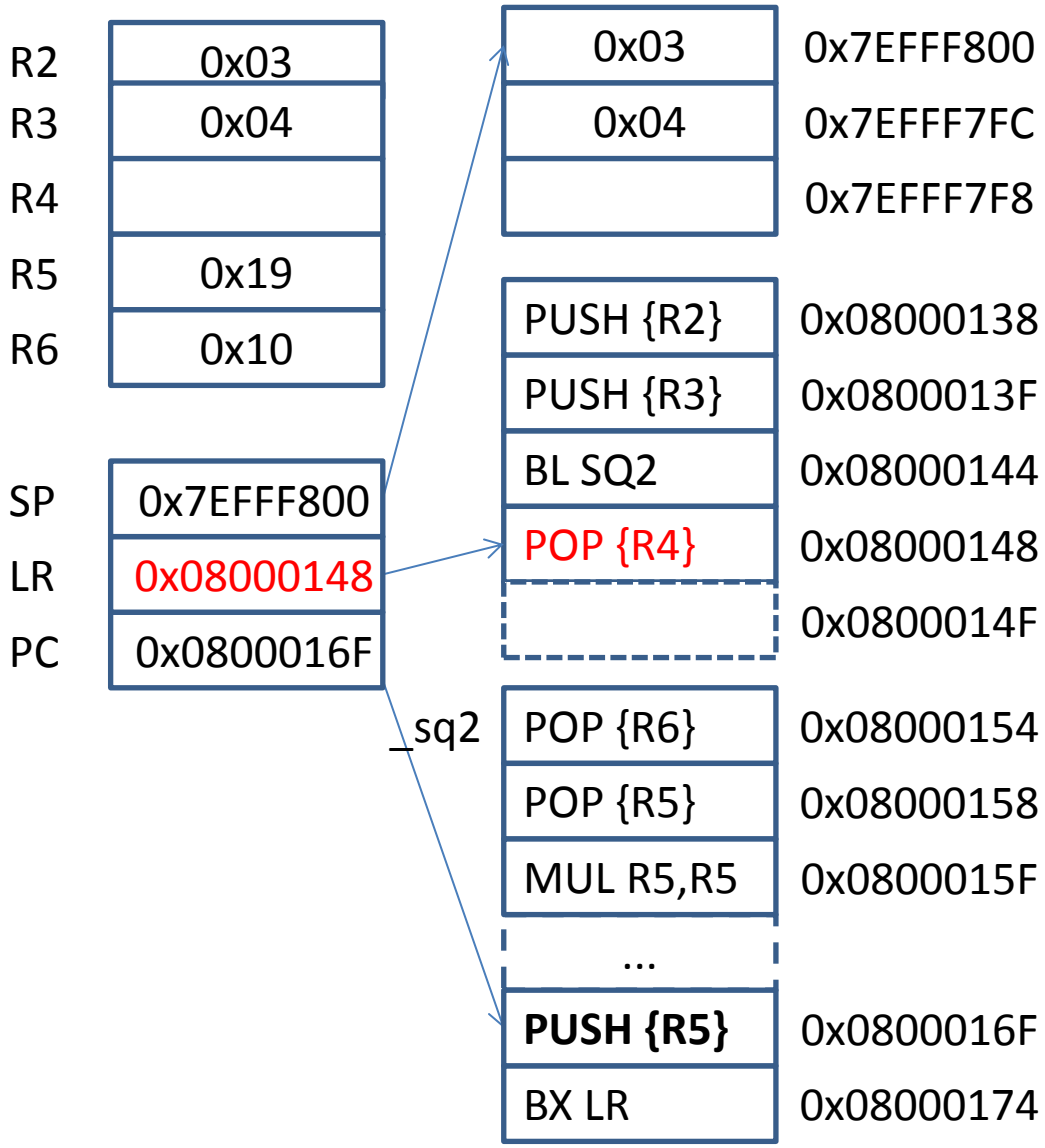
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



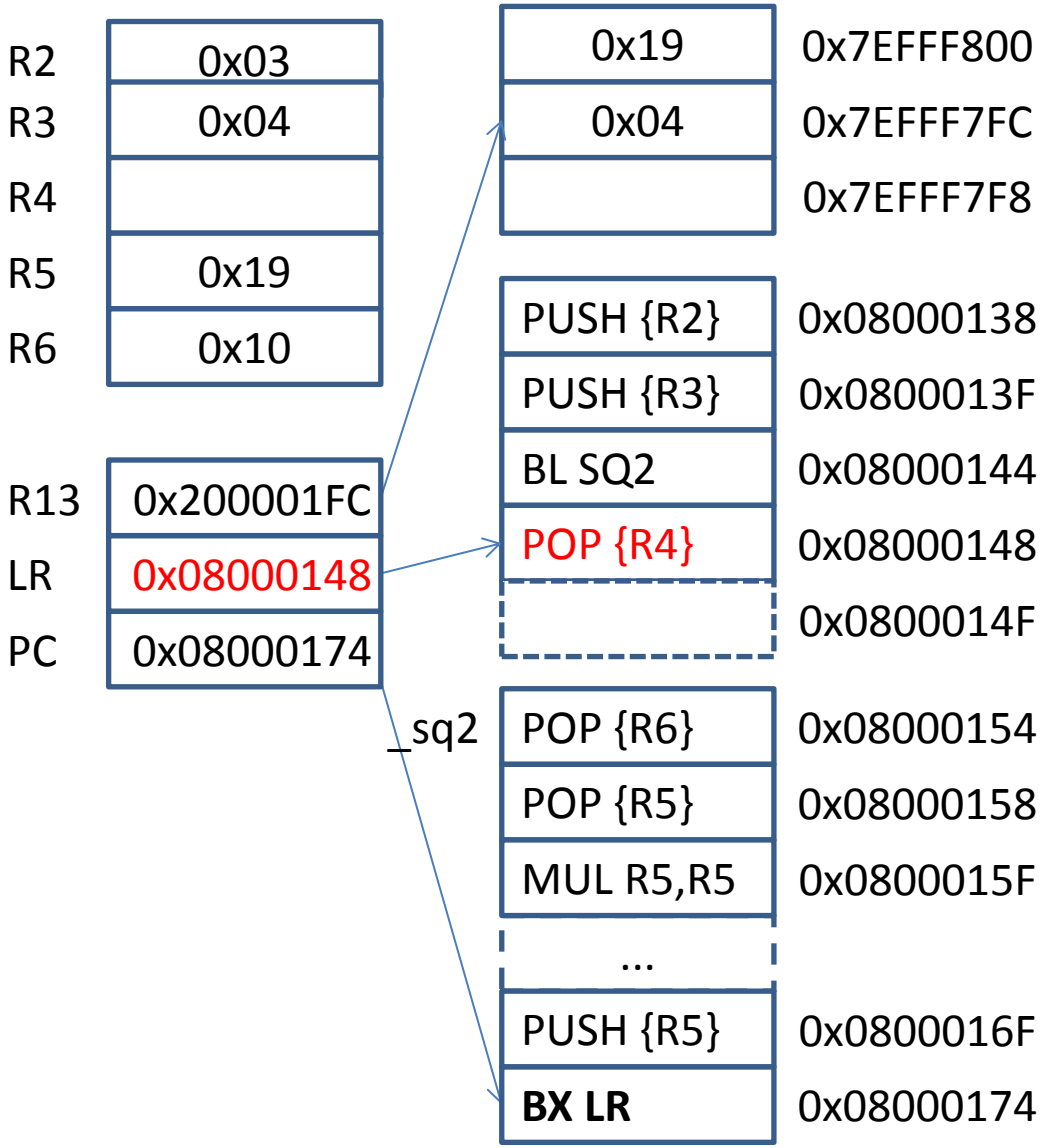
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



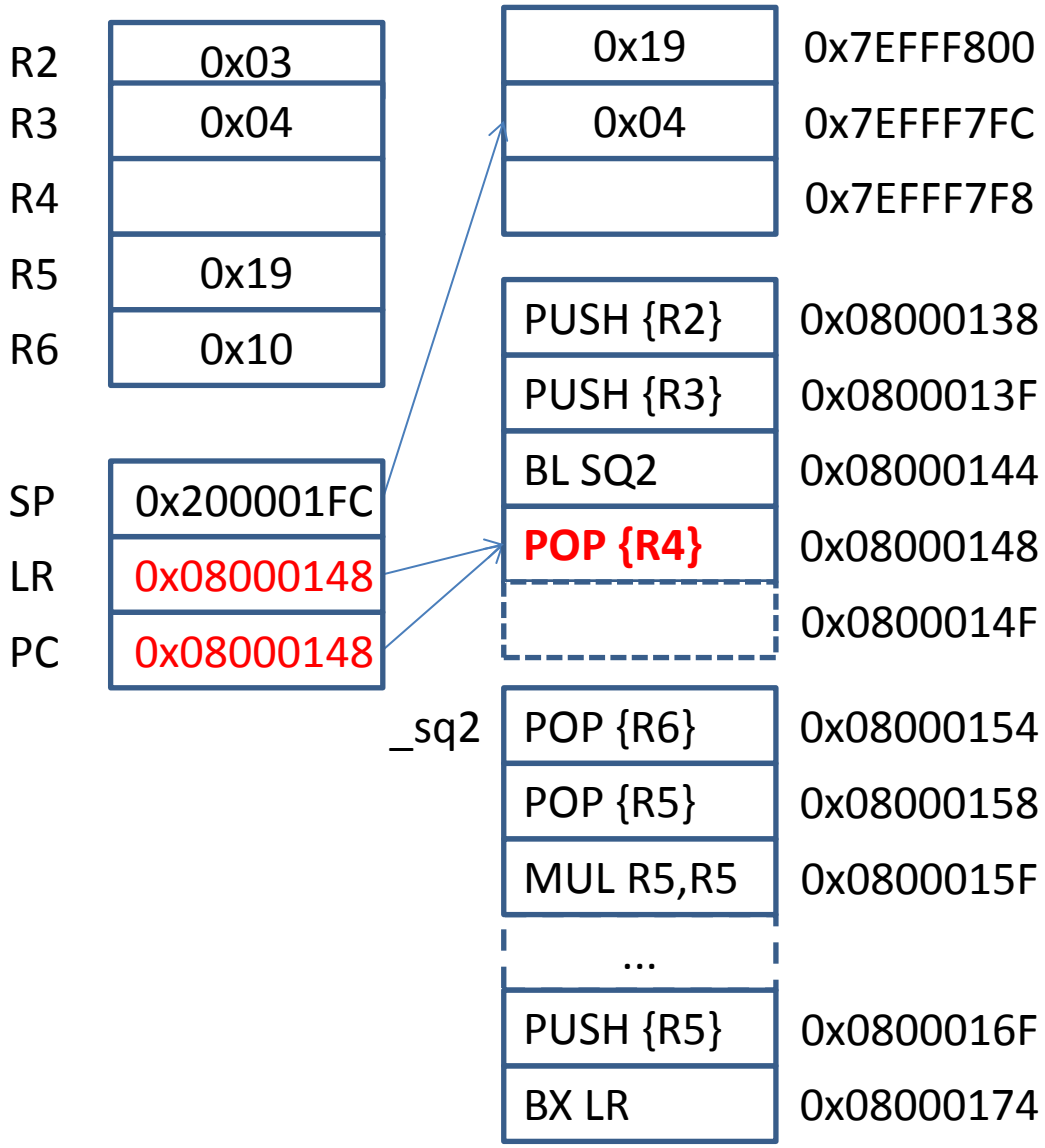
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



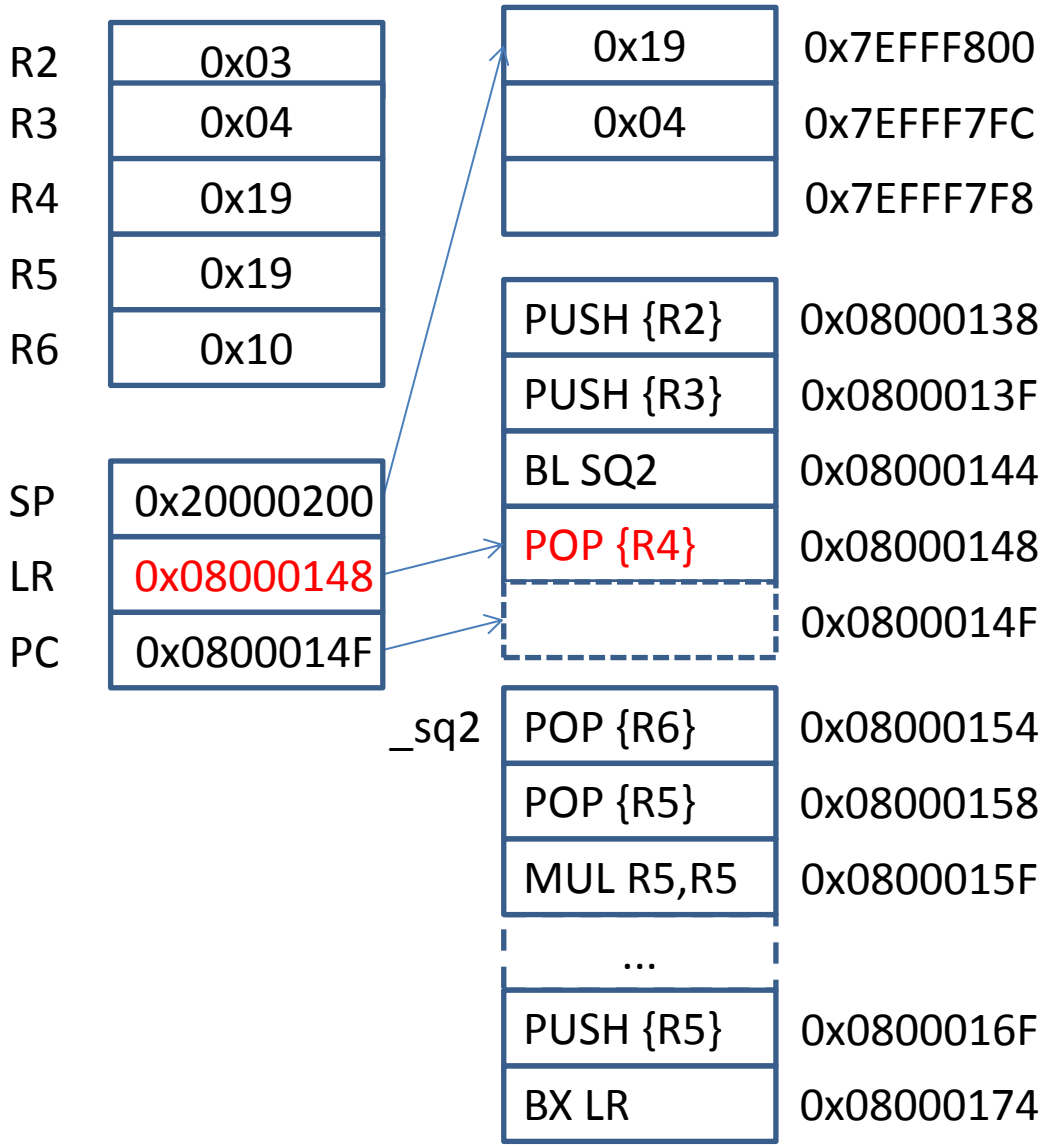
Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```



Example: stack parameter passing

```
        ...
        PUSH {R2}
        PUSH {R3}
        BL SQ2
        POP {R4}
        ...
_sq2:   POP {R6}
        POP {R5}
        MUL R4,R5,R5
        MUL R5,R6,R6
        ADD R5,R4
        PUSH {R5}
        BX LR
        ...
```

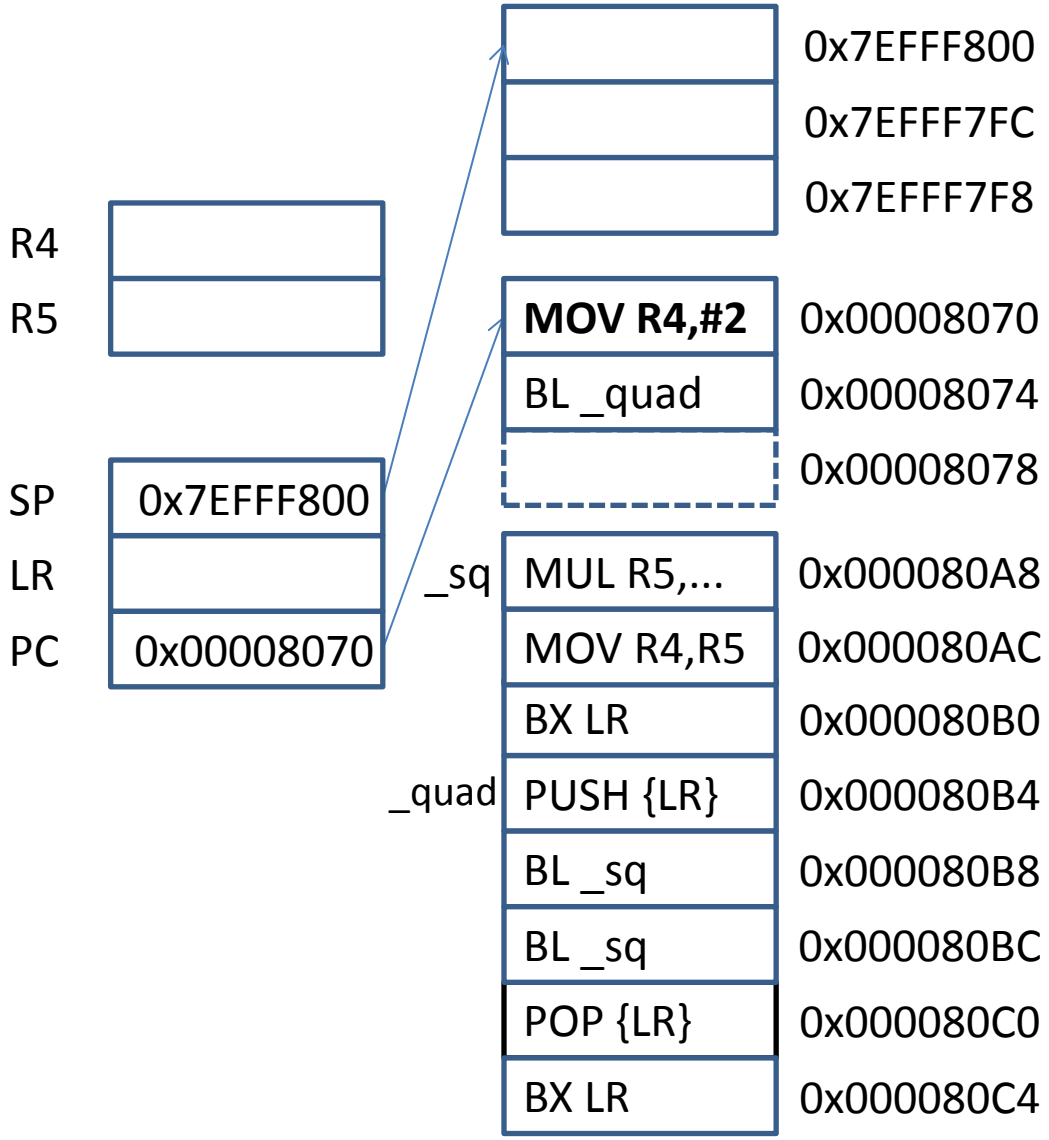


Nested subroutines

- push LR (& working registers) onto stack before nested call
- pop LR (& working registers) off stack after nested return

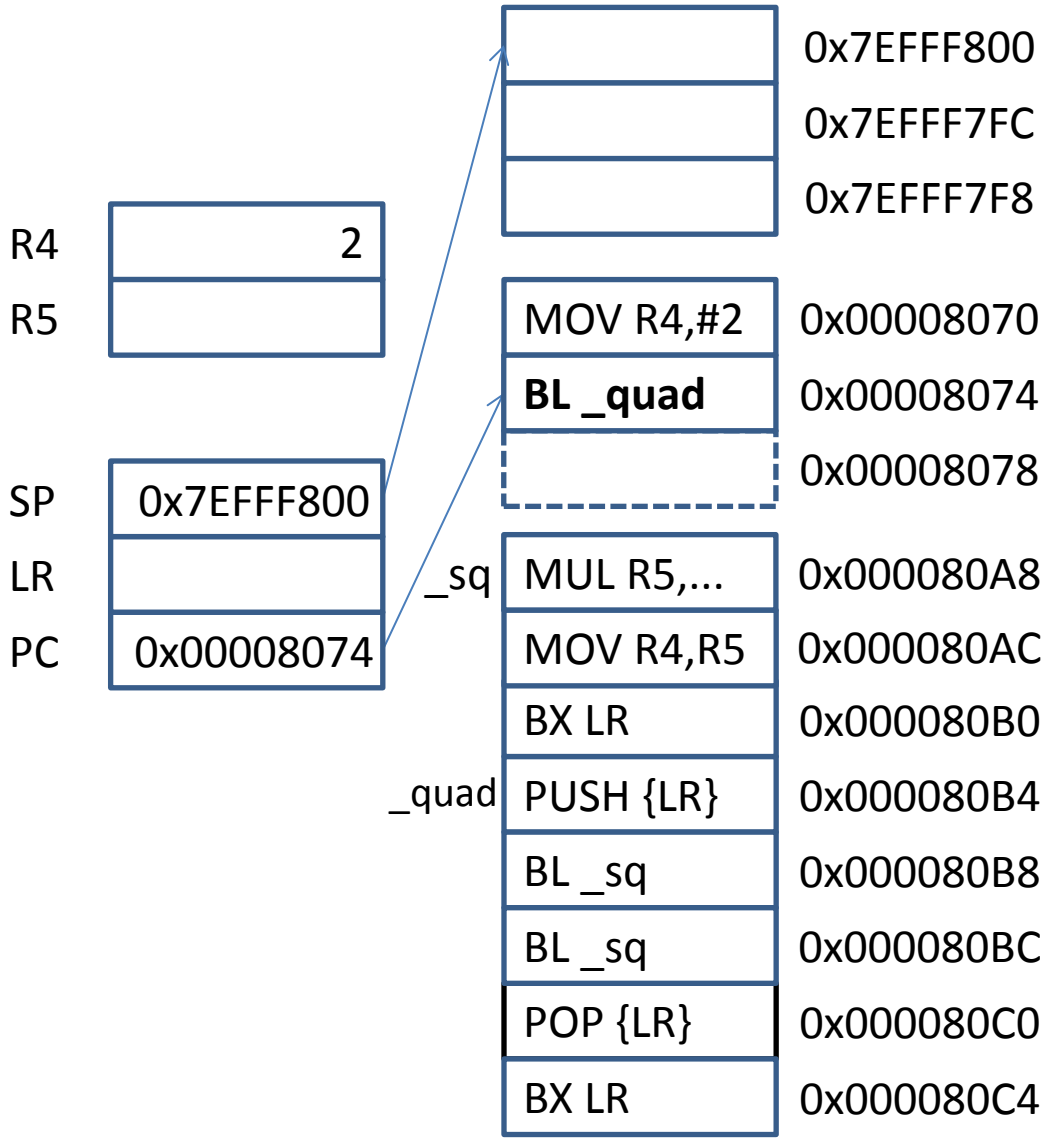
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



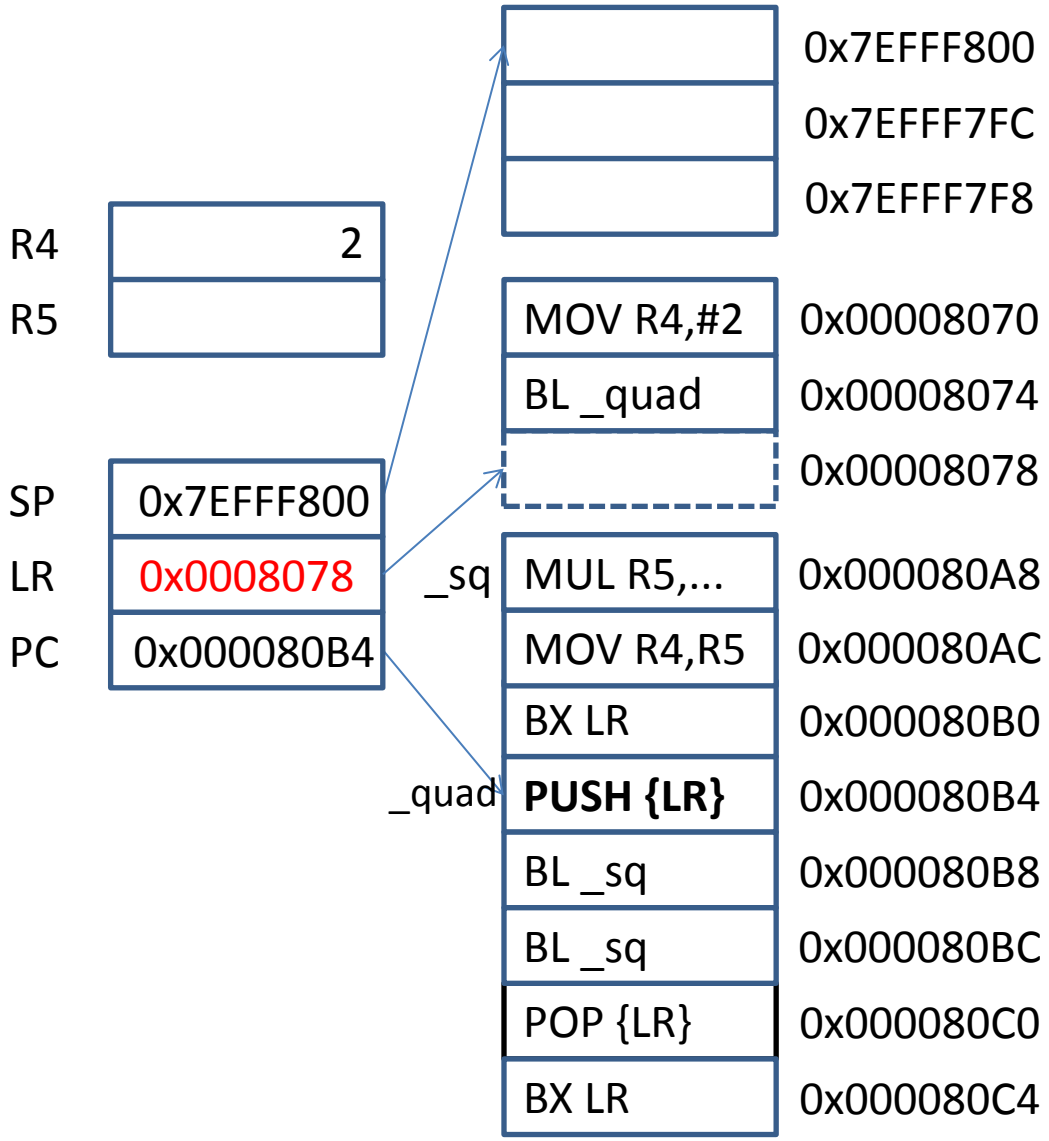
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



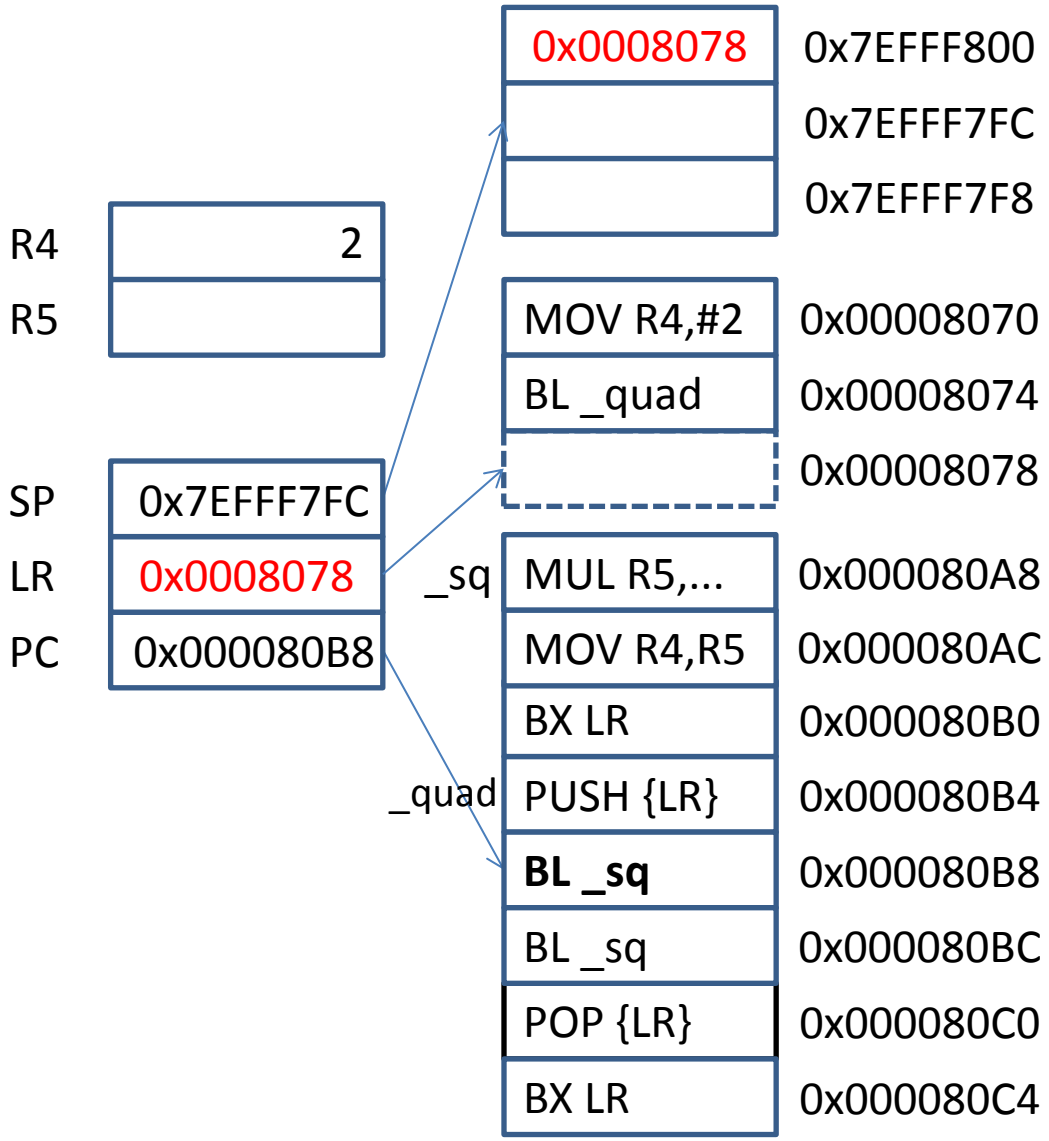
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



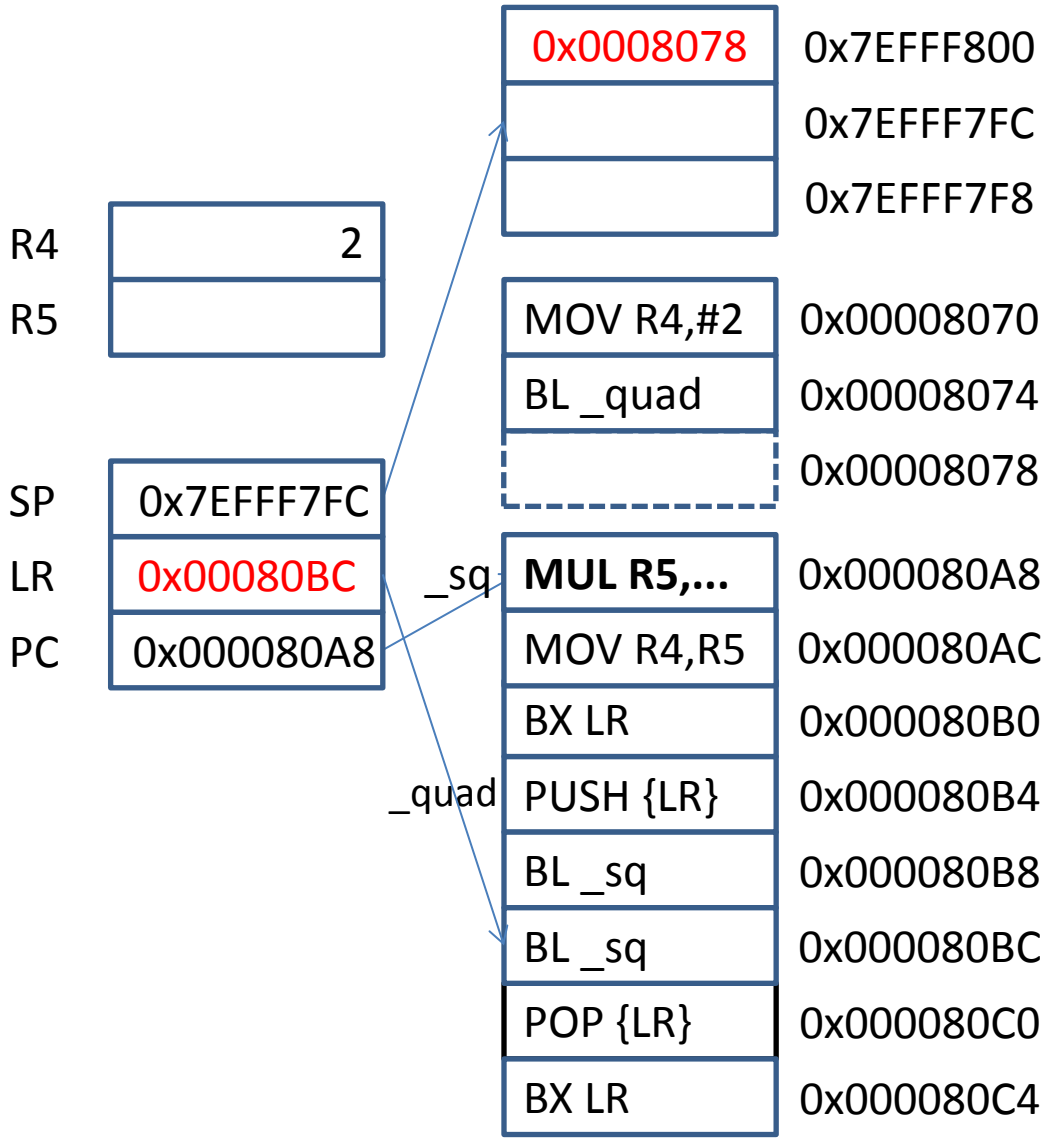
Example: R4 = $R4^4$

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



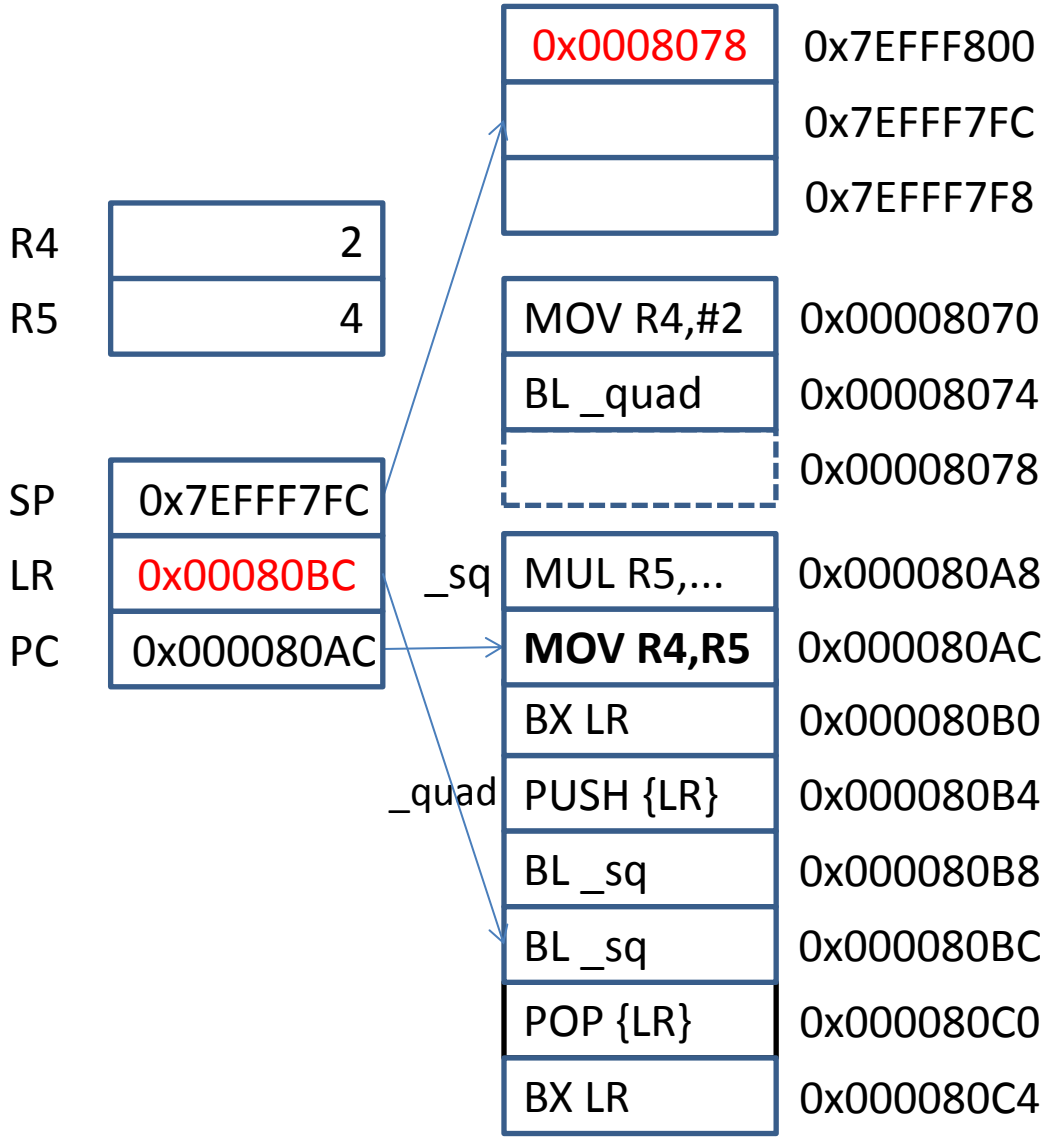
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



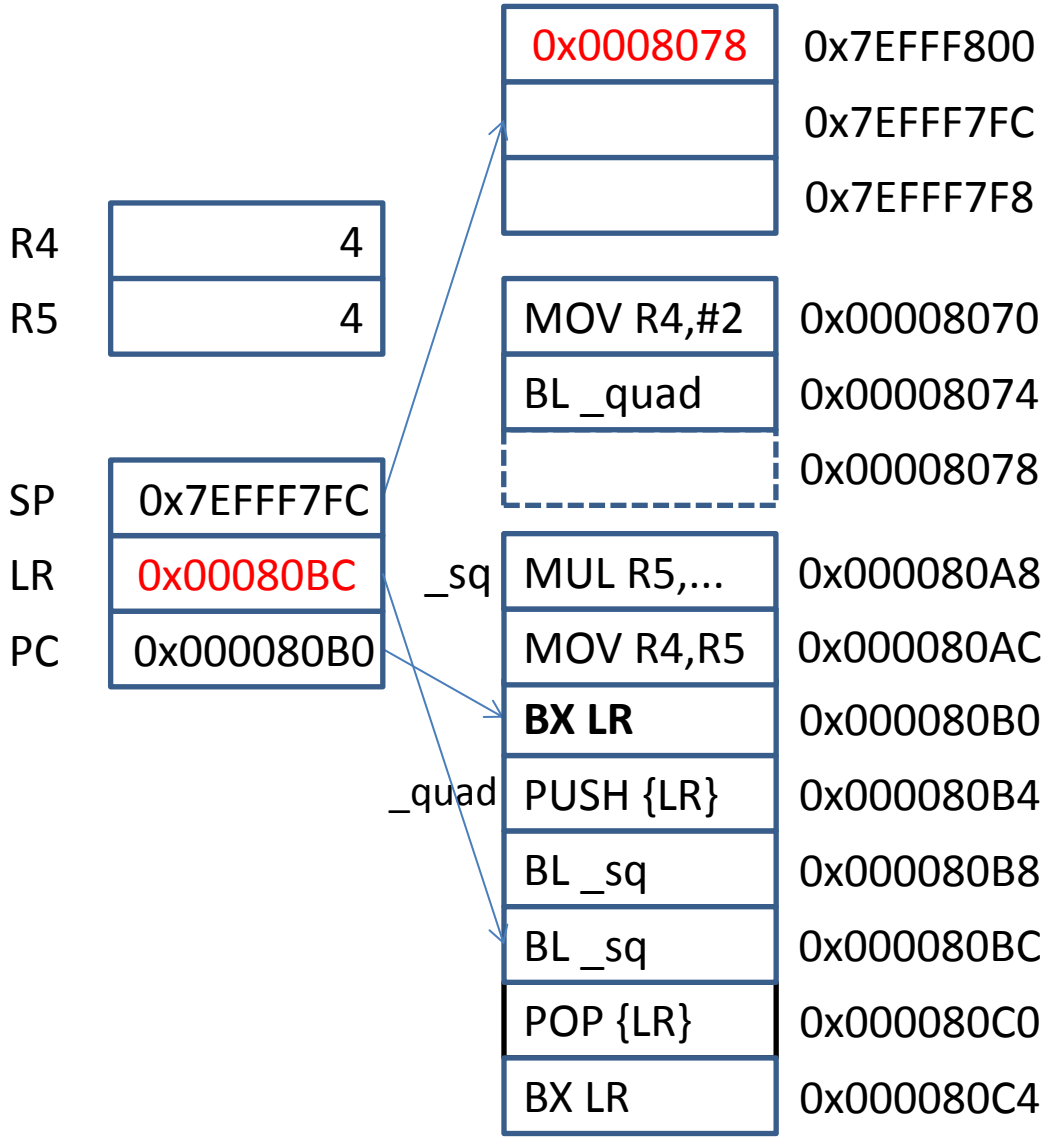
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



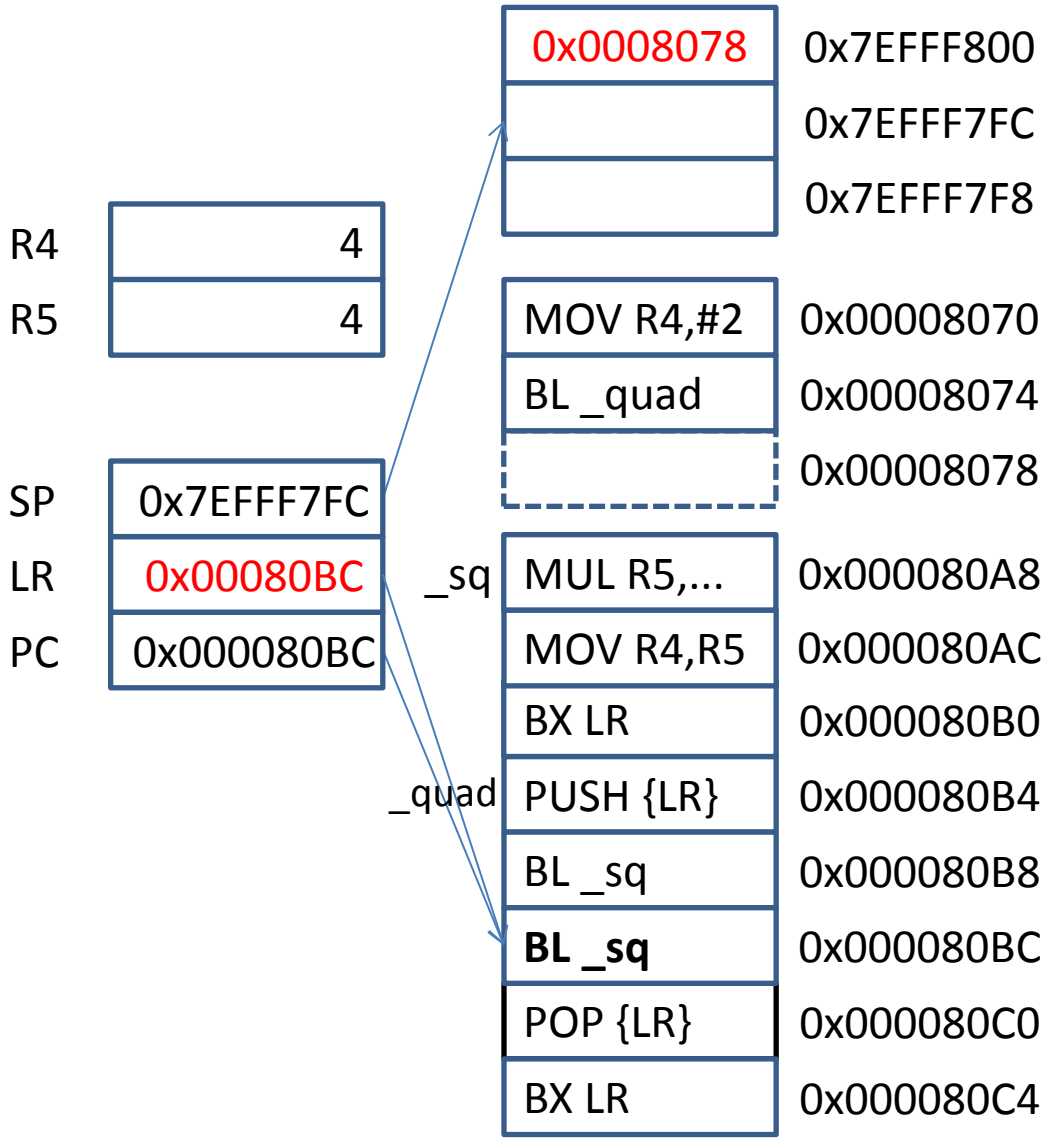
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



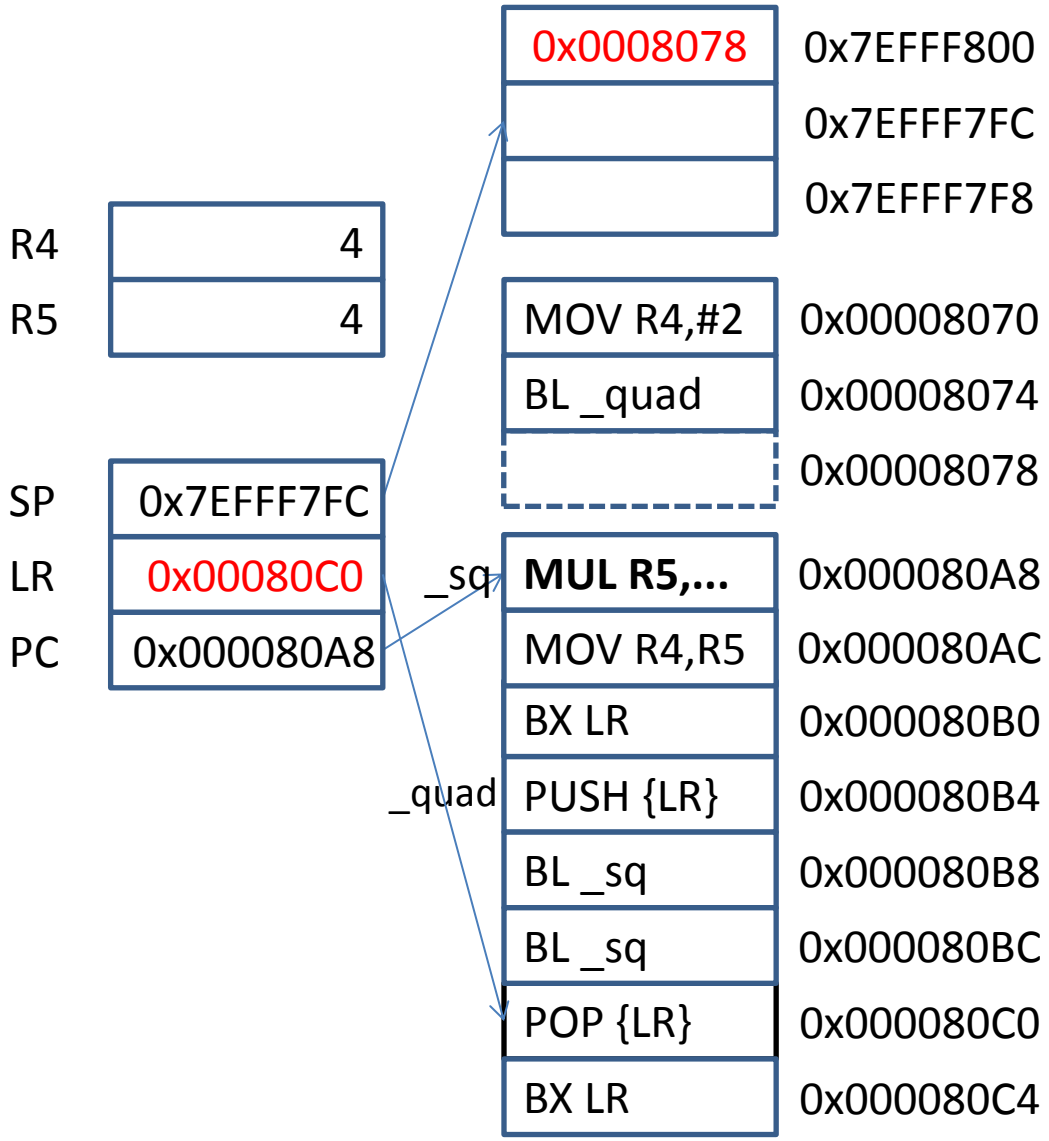
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



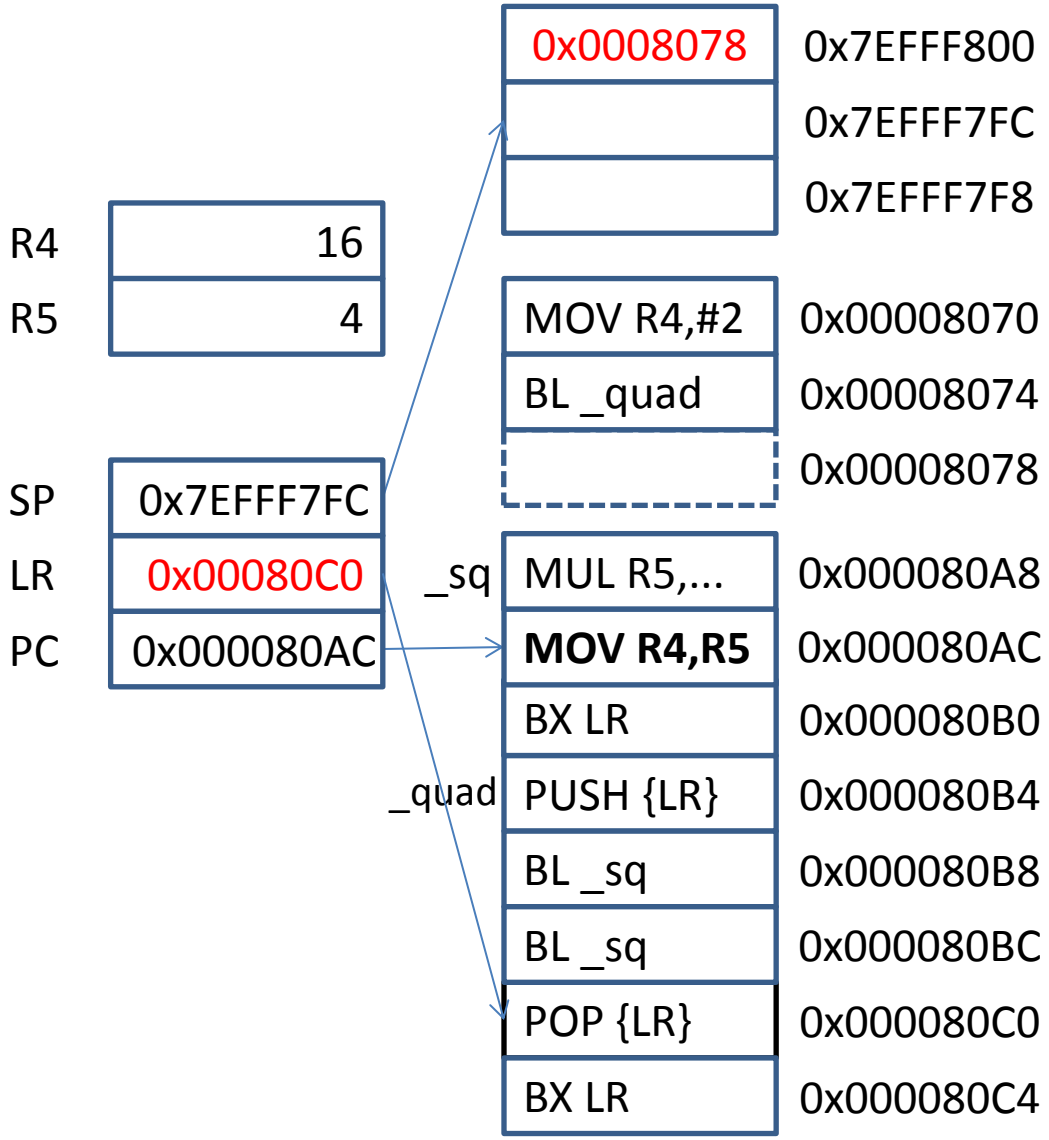
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



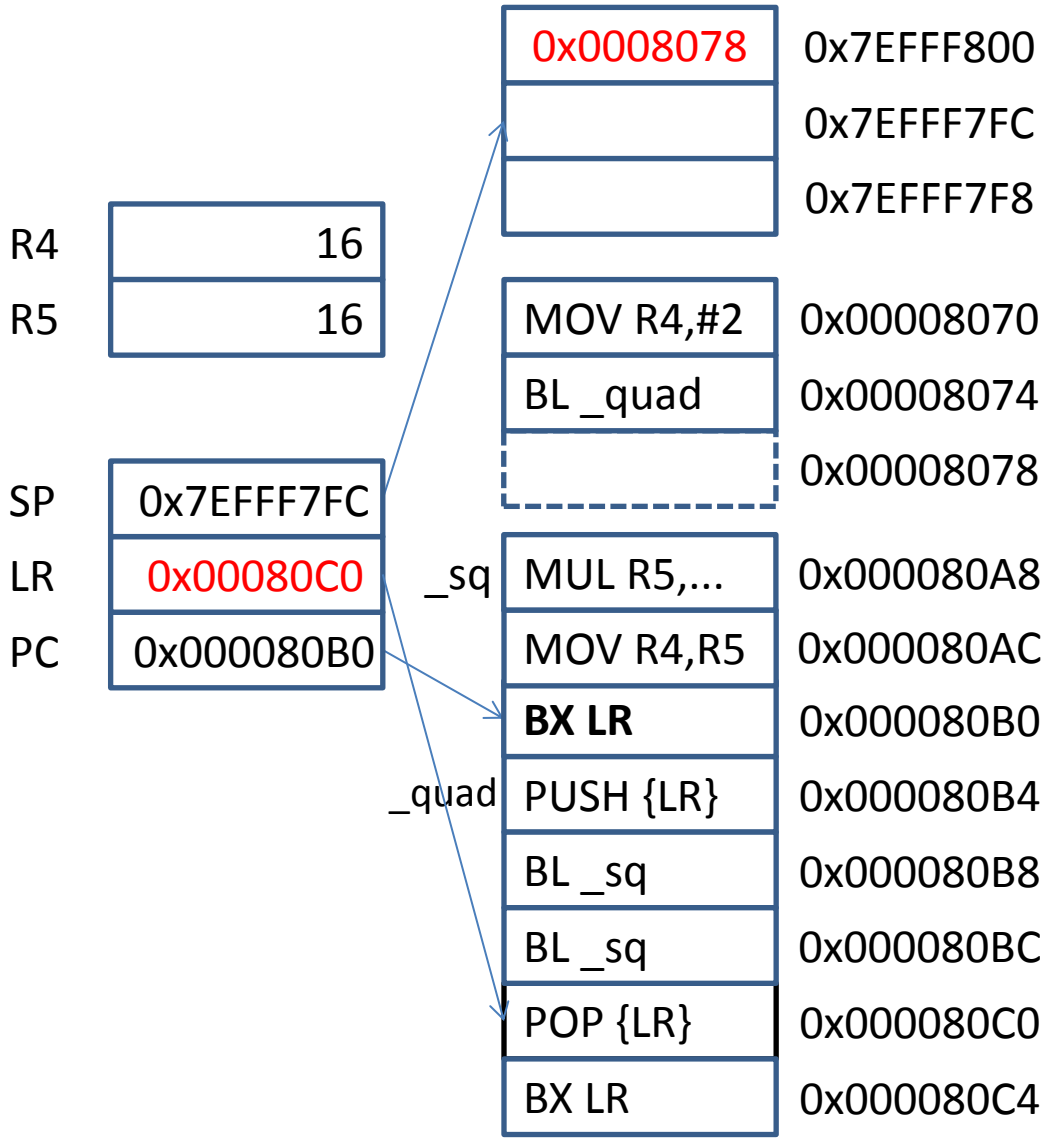
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



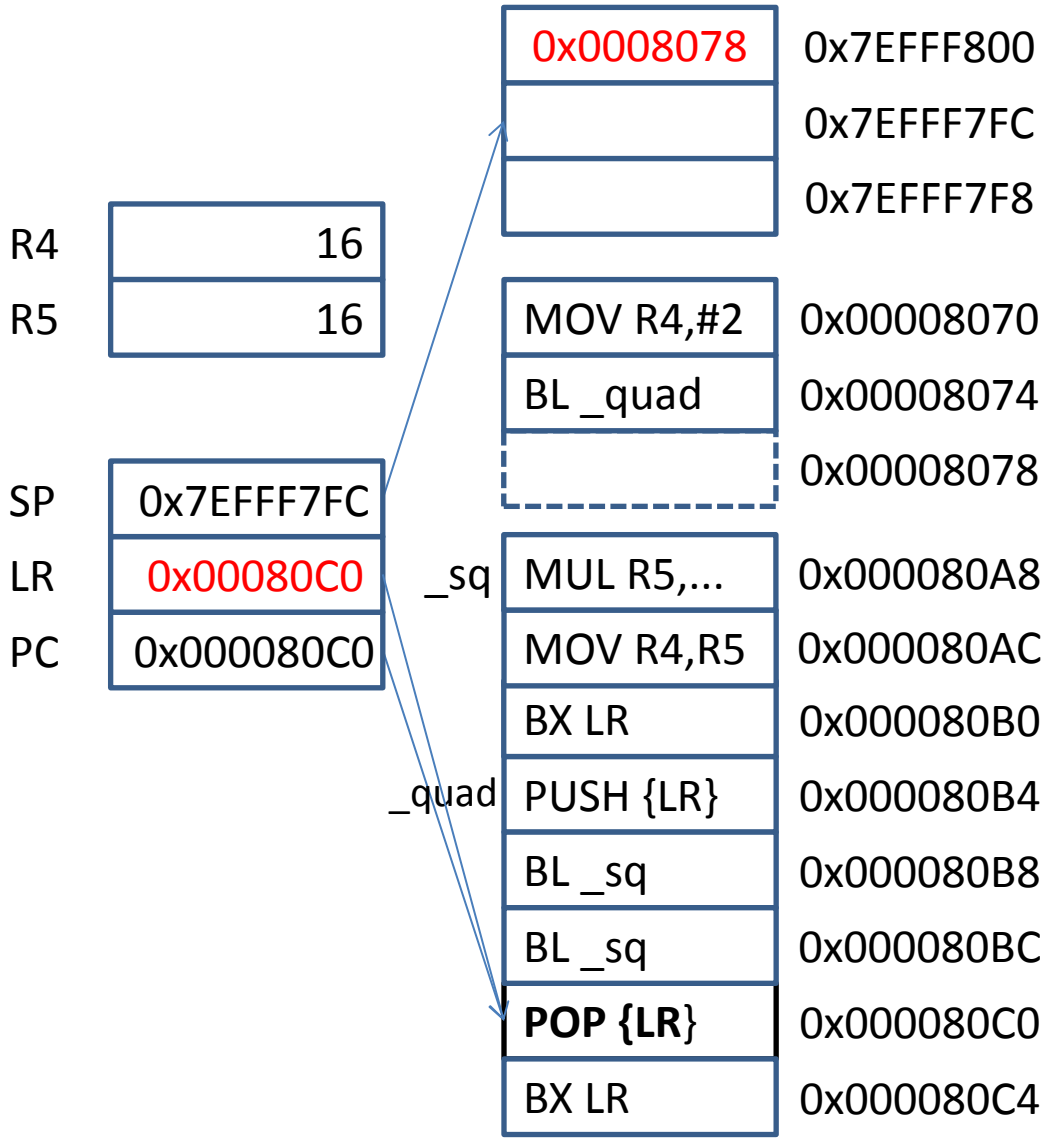
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



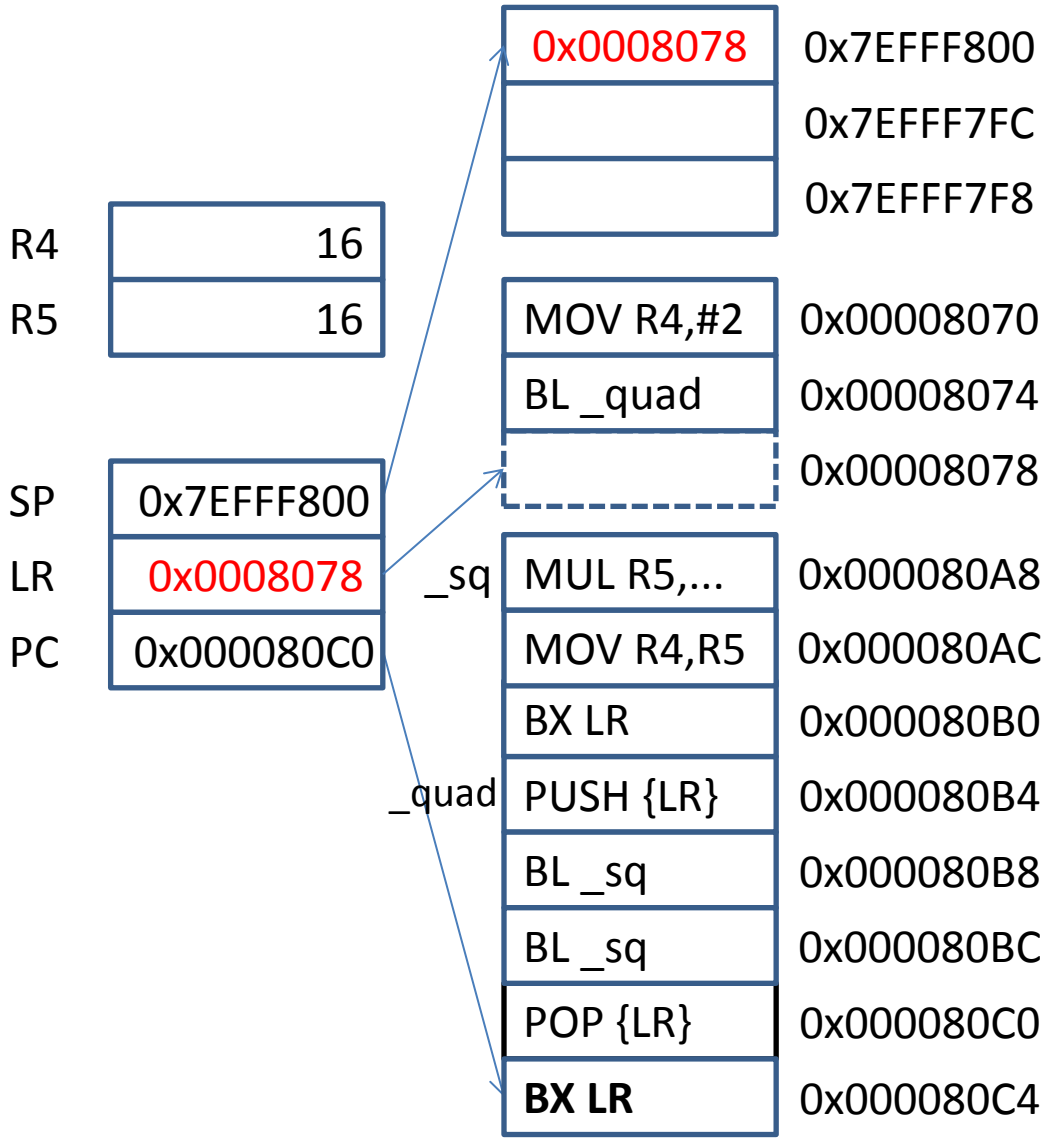
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
   ...
```



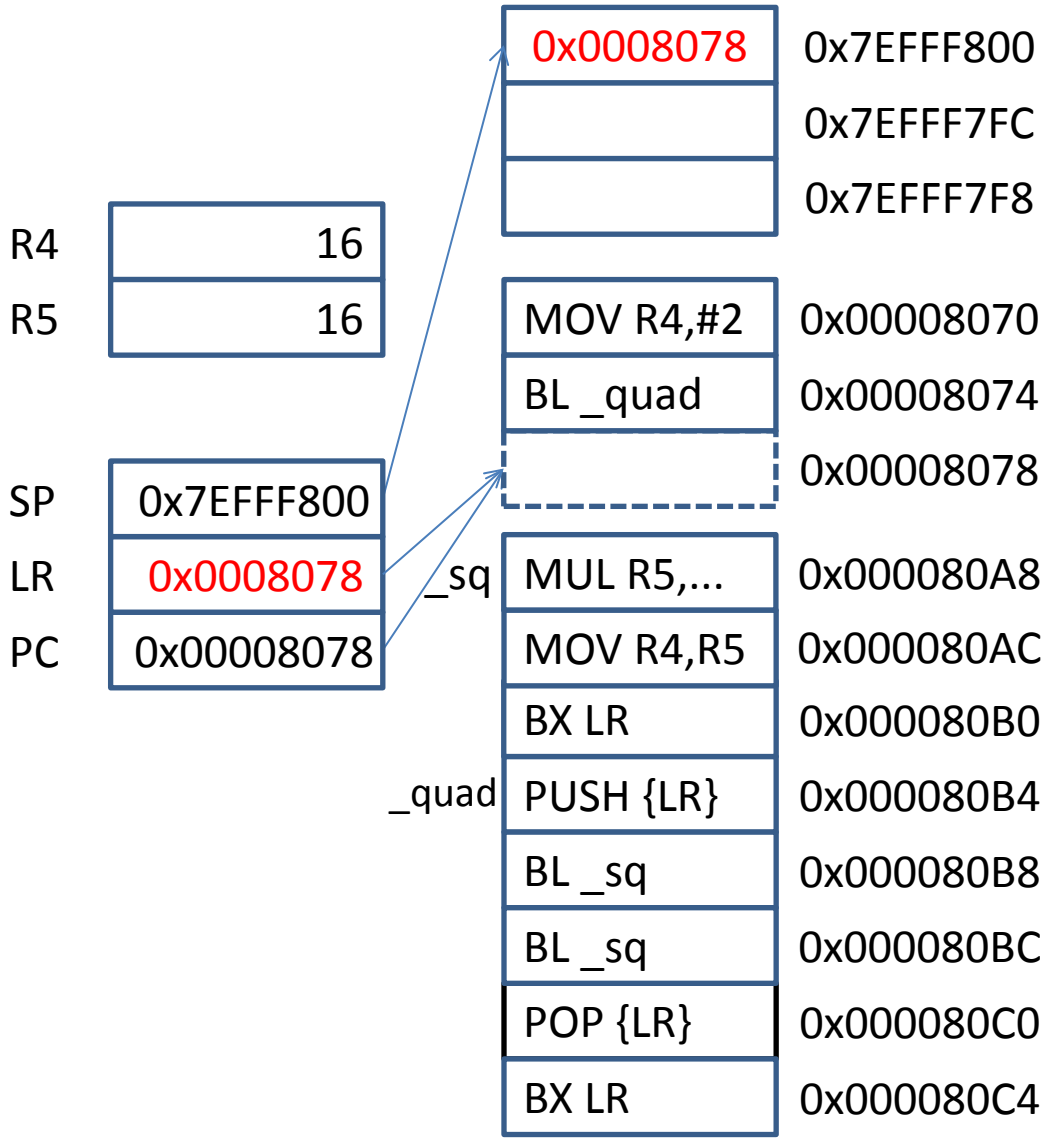
Example: R4 = R4^4

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
    ...
```



Example: R4 = $R4^4$

```
        MOV R4,#2
        BL _quad
        ...
_sq:    MUL R5,R4,R4
        MOV R5,R4
        BX LR
_quad:  PUSH {LR}
        BL _sq
        BL _sq
        POP {LR}
        BX LR
_exit:
```



Recursion: R4 = $R2^{R3}$

- subroutine can call itself
- e.g. $z = x^y$

```
pow(int x, int y, int z)
{   if(y==0)
     return z;
    pow(x,y-1,z*x);
}
```

- e.g. 2^3

```
pow (2,3,1) → pow (2,2,2) →
pow (2,1,4) → pow (2,0,8) → 8
```

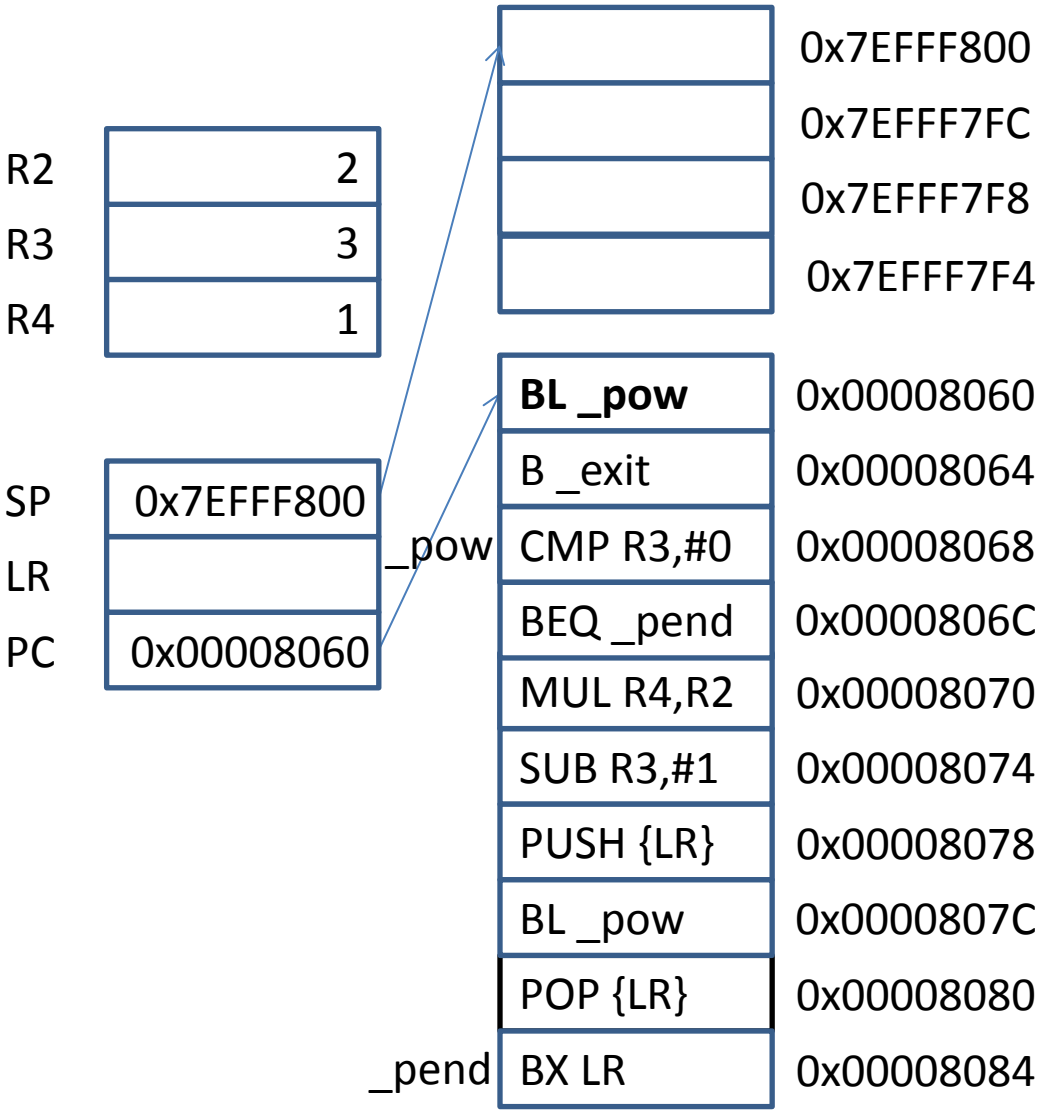
Recursion: R4 = $R2^{R3}$

```
R2 == x
R3 == y
R4 == z
pow(int R2, int R3,
  int R4)
{  if(R3==0)
    return R4;
   pow(R2,R3-1,R4*R2);
}
```

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```

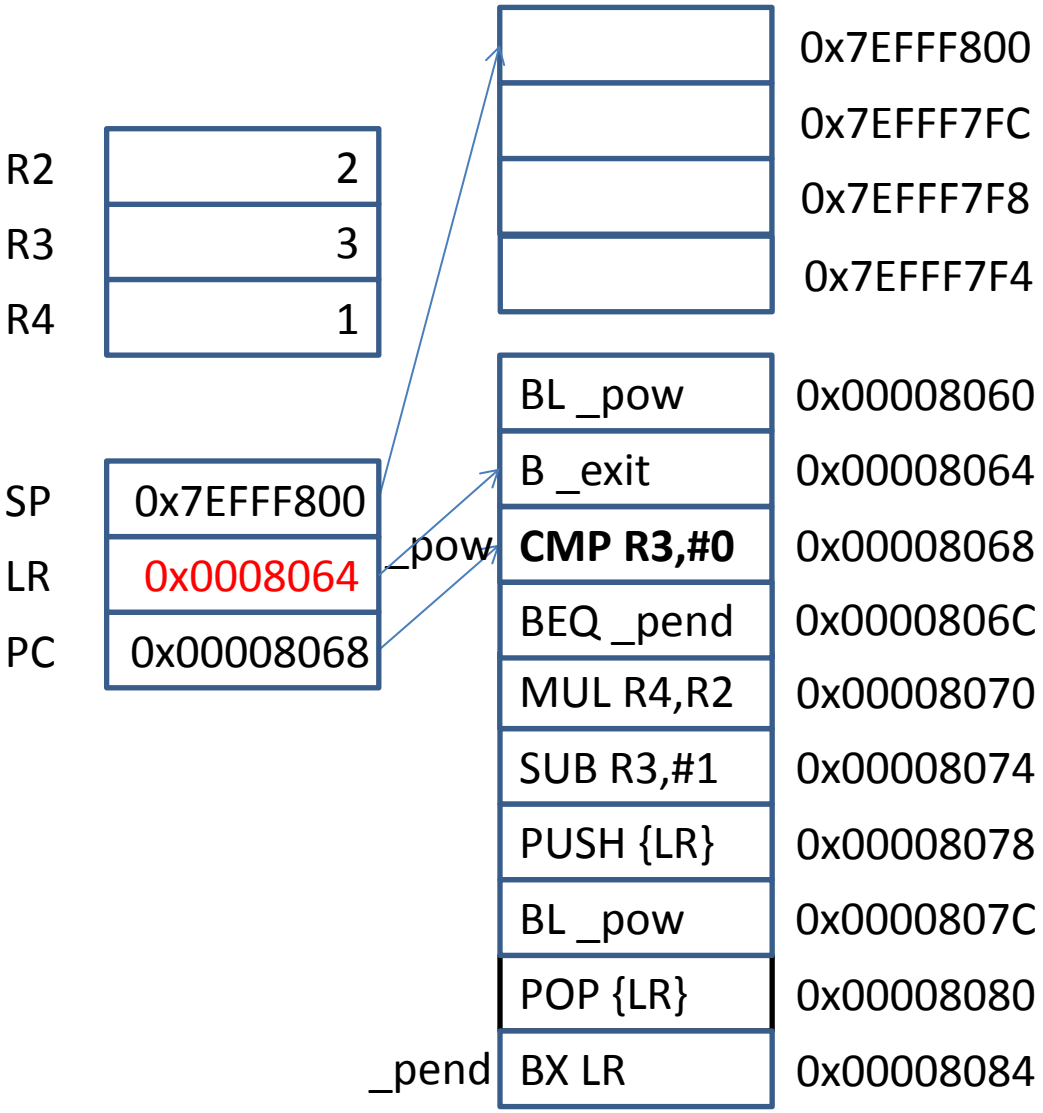
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



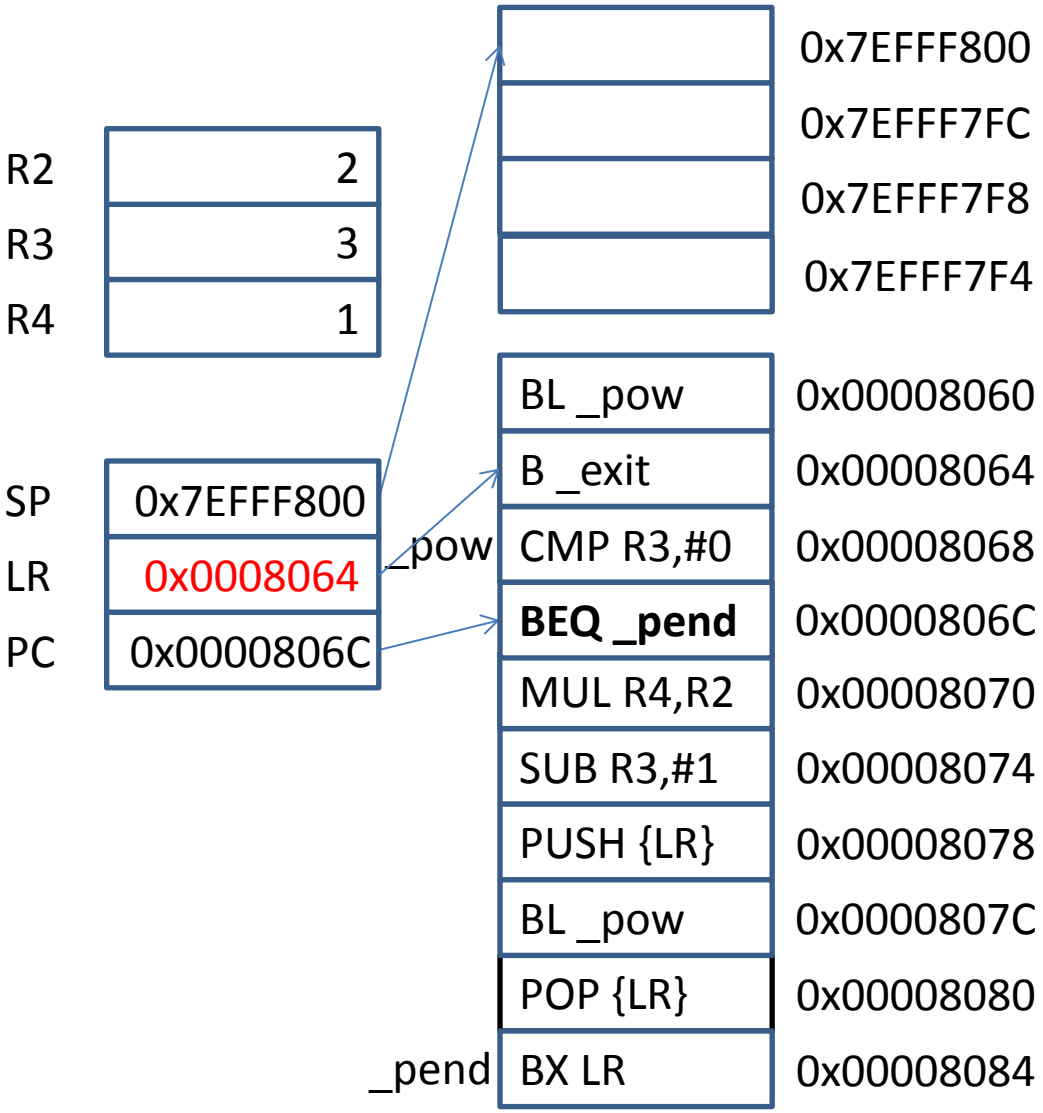
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



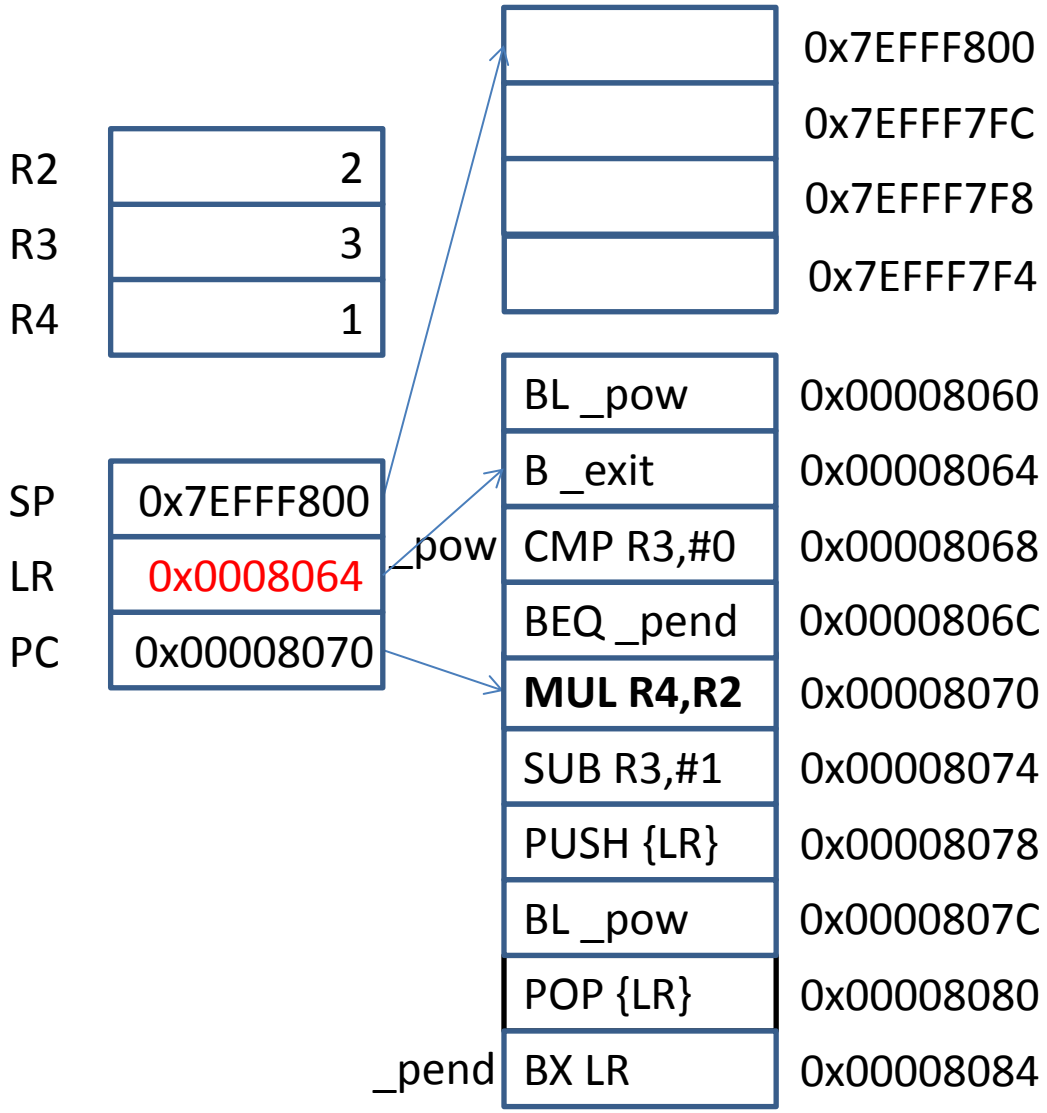
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



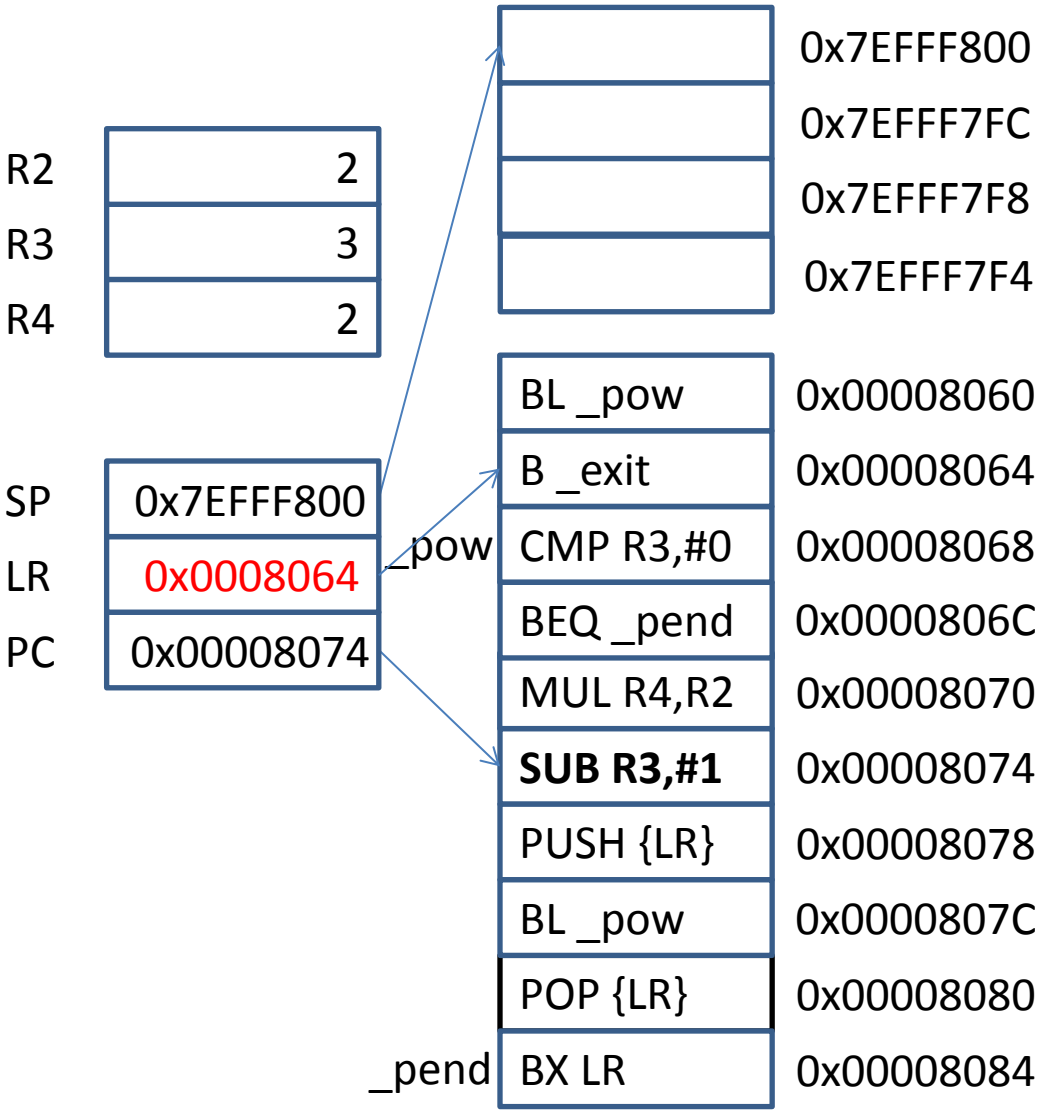
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



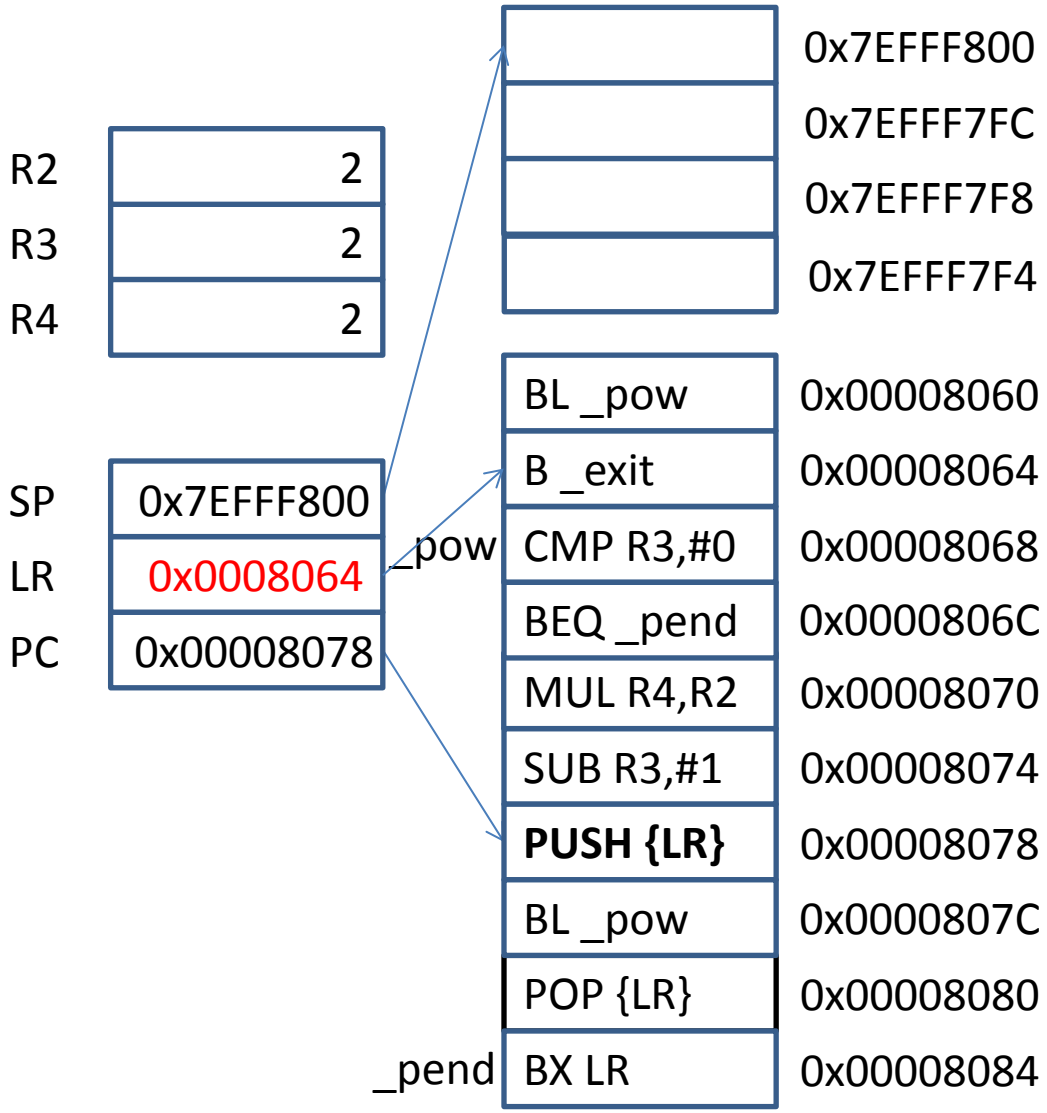
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



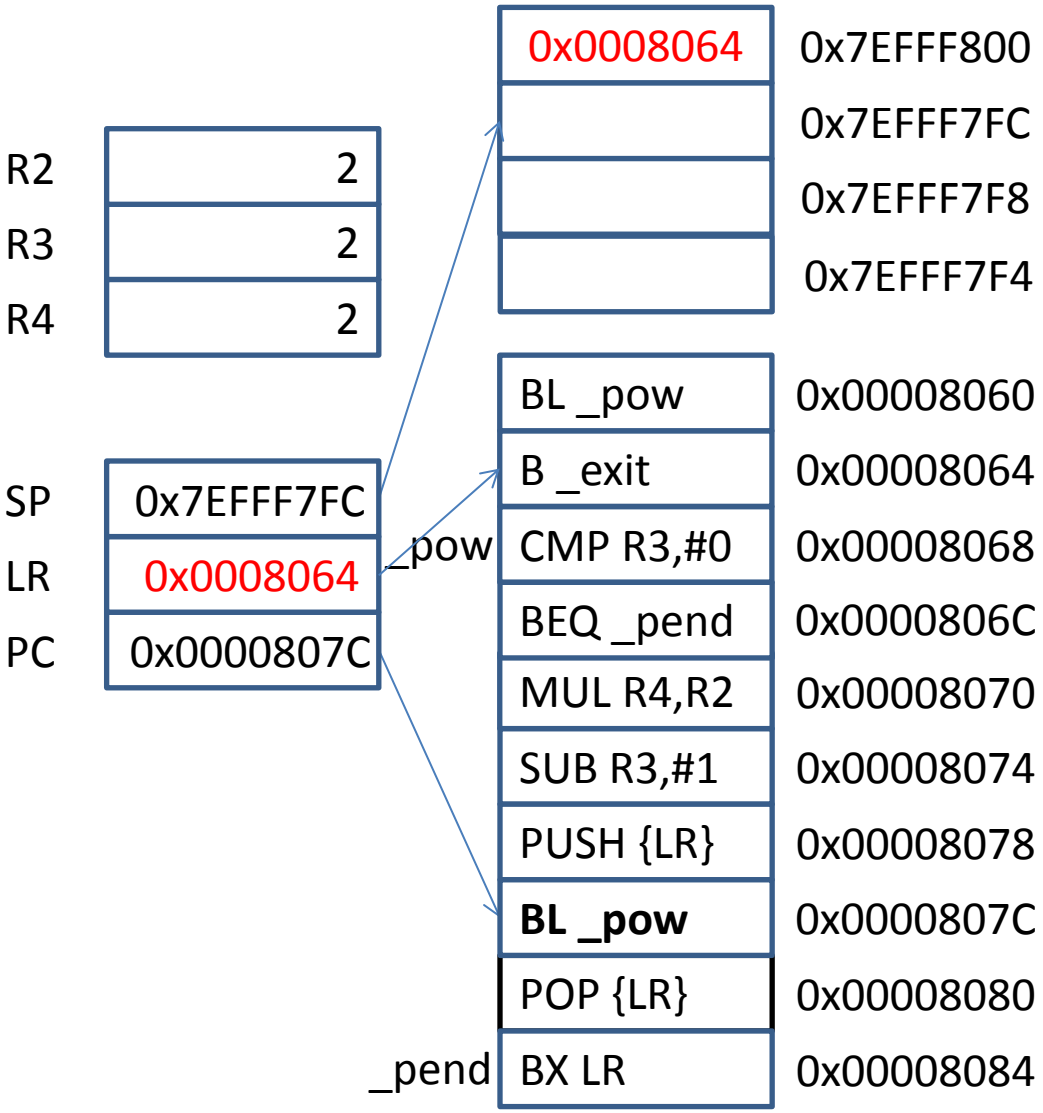
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



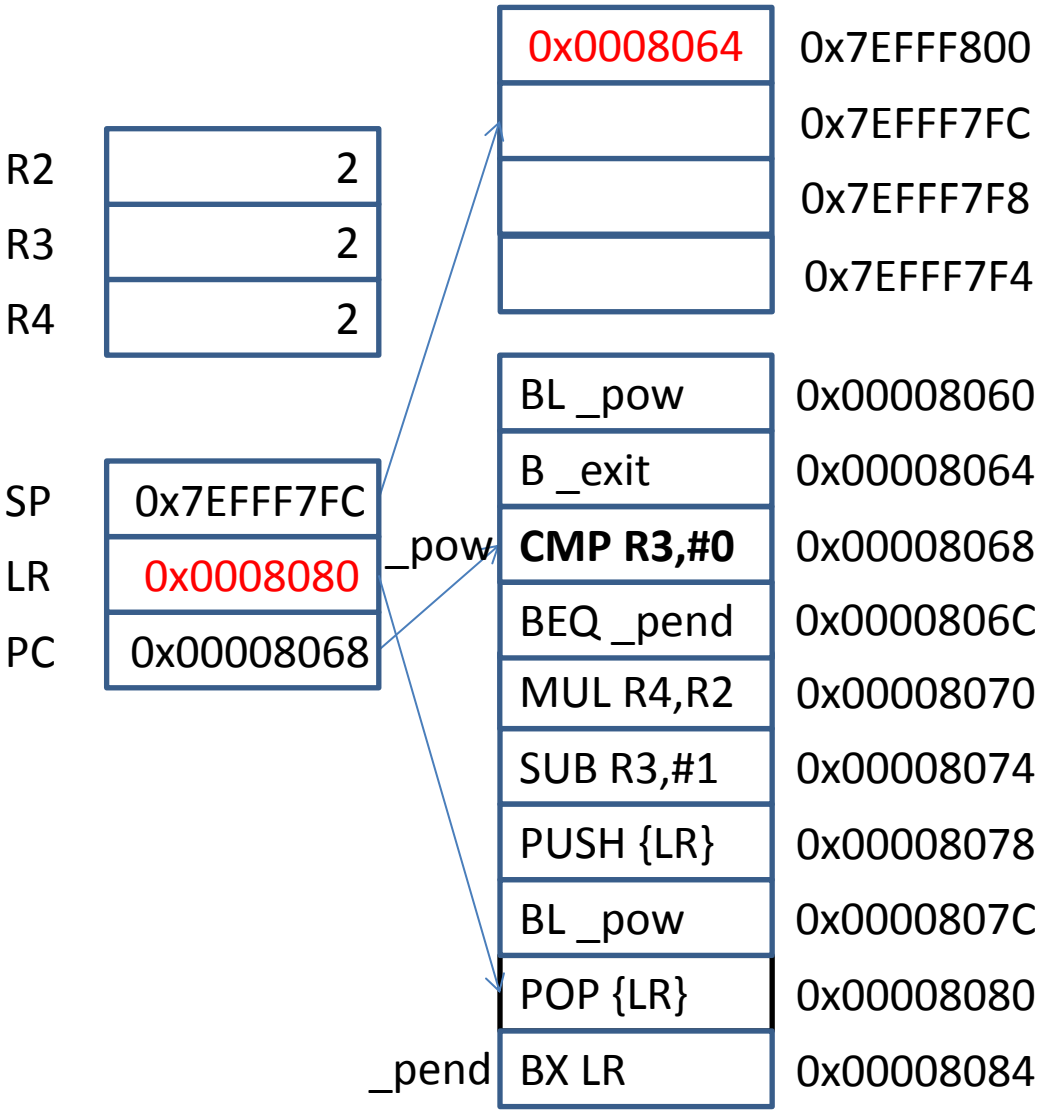
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



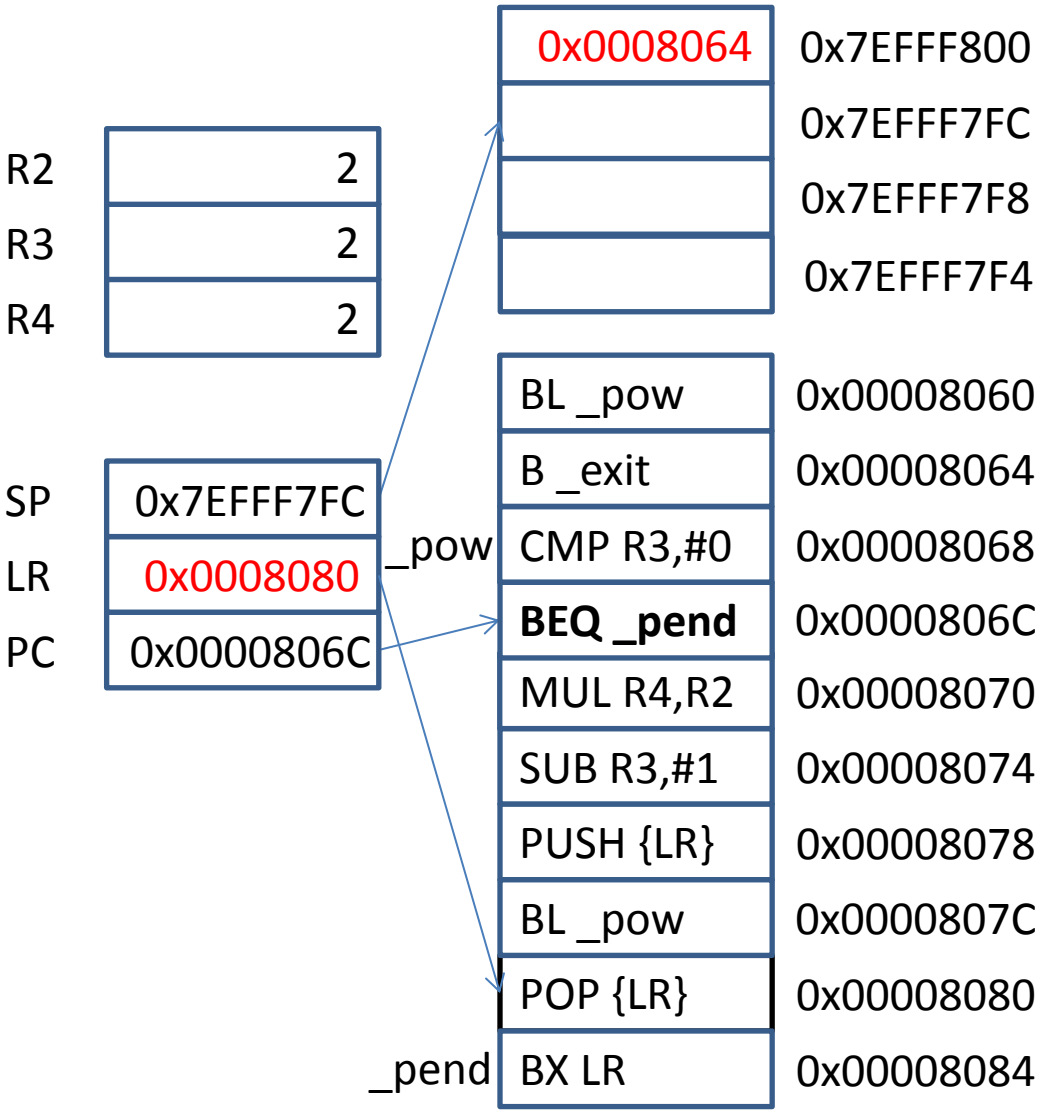
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



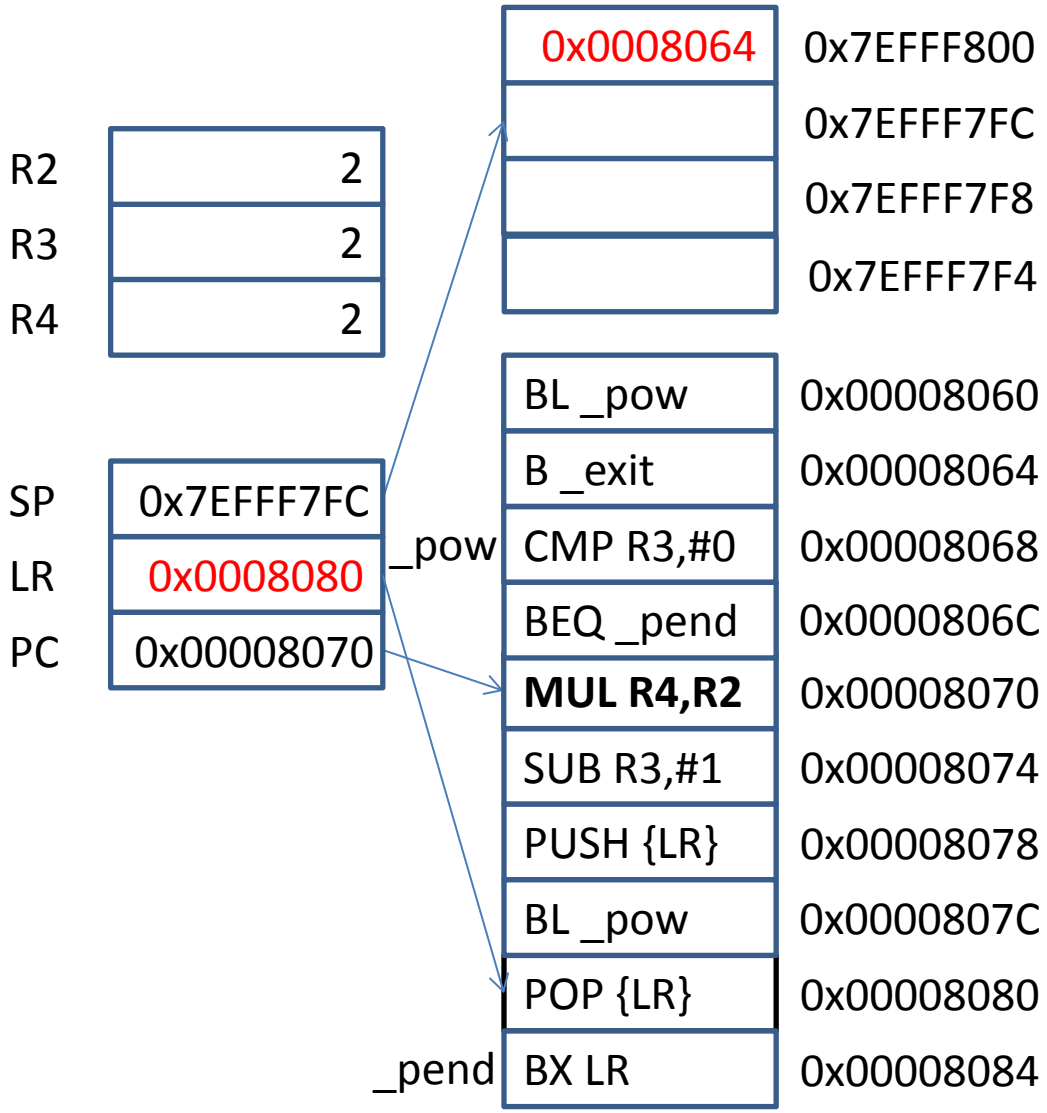
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



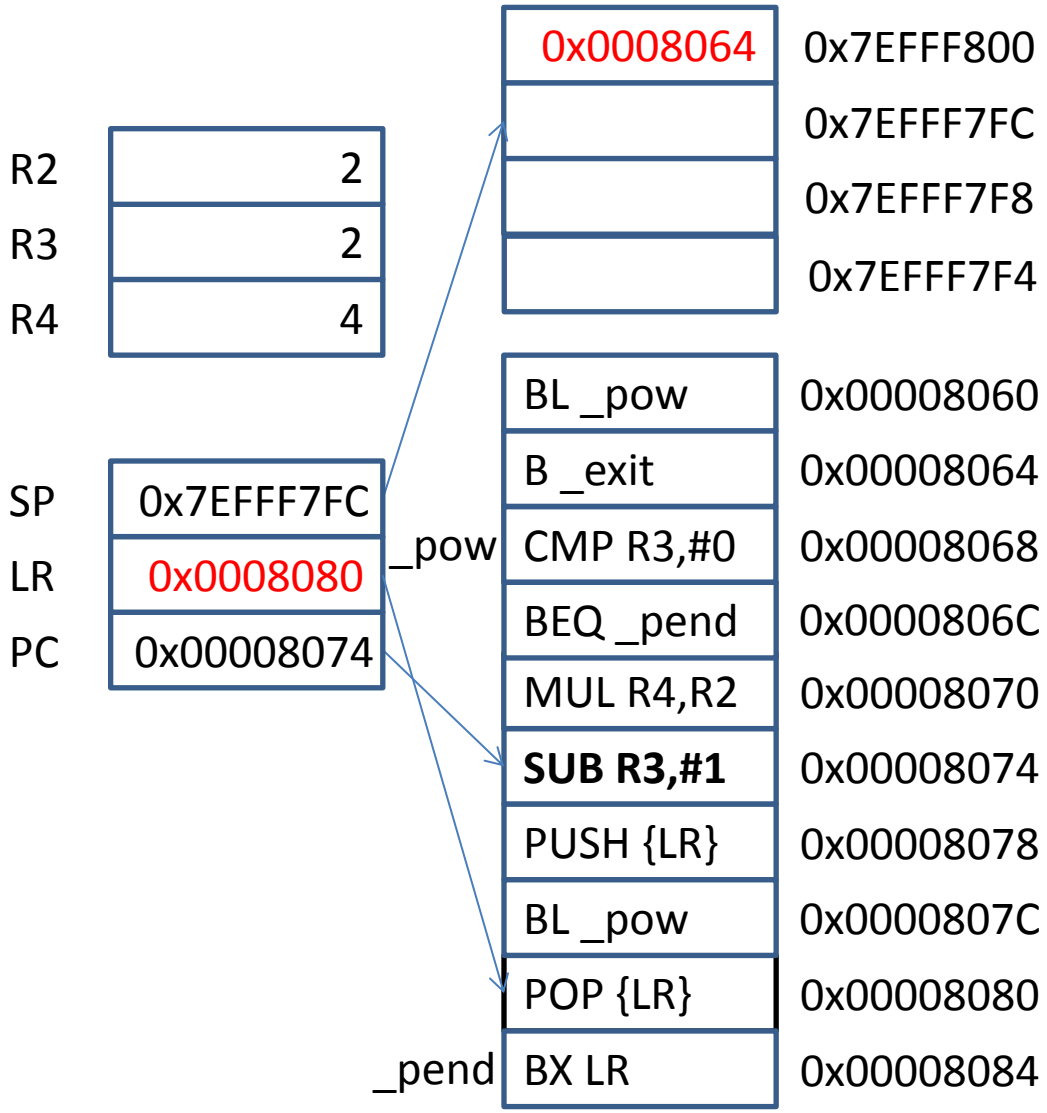
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



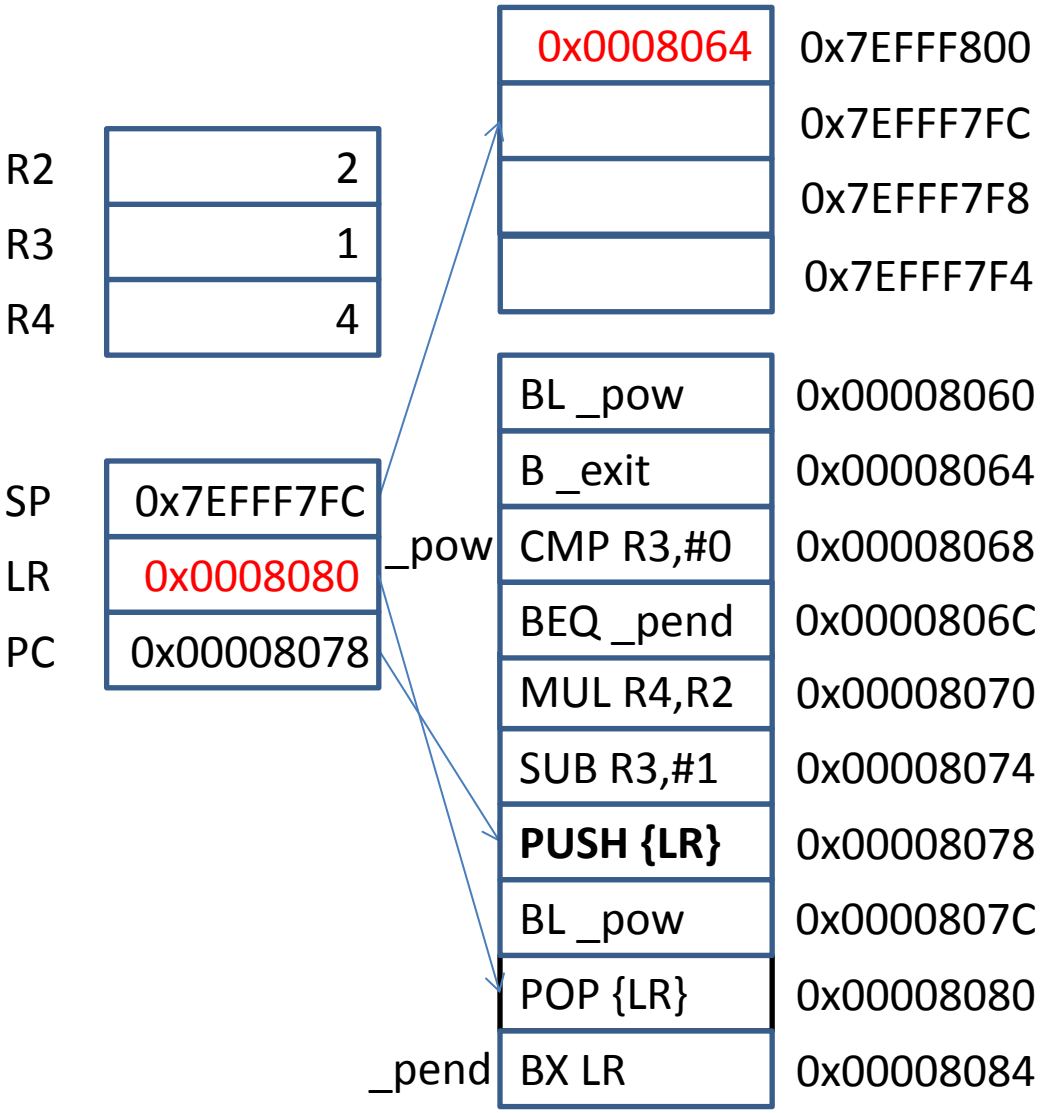
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



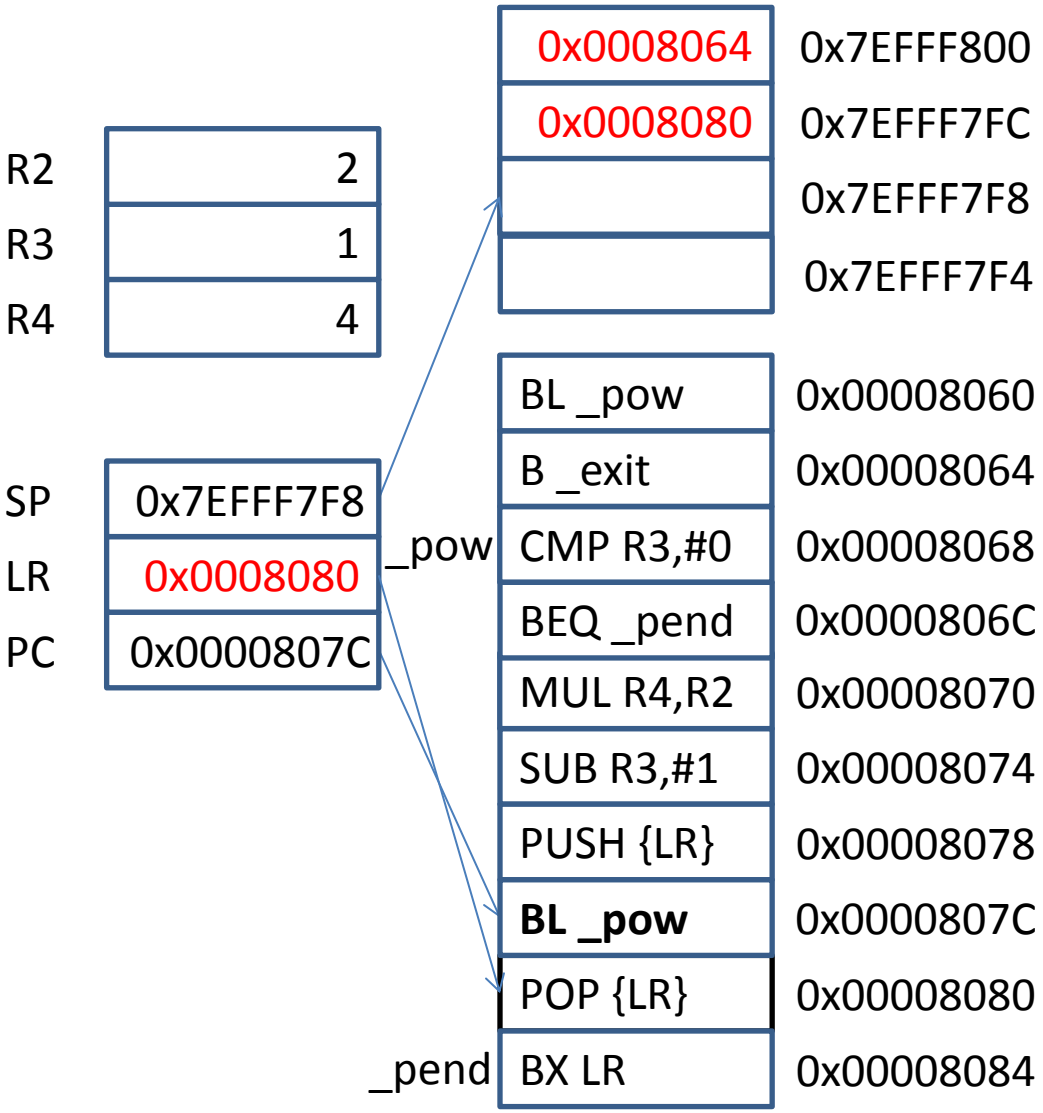
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



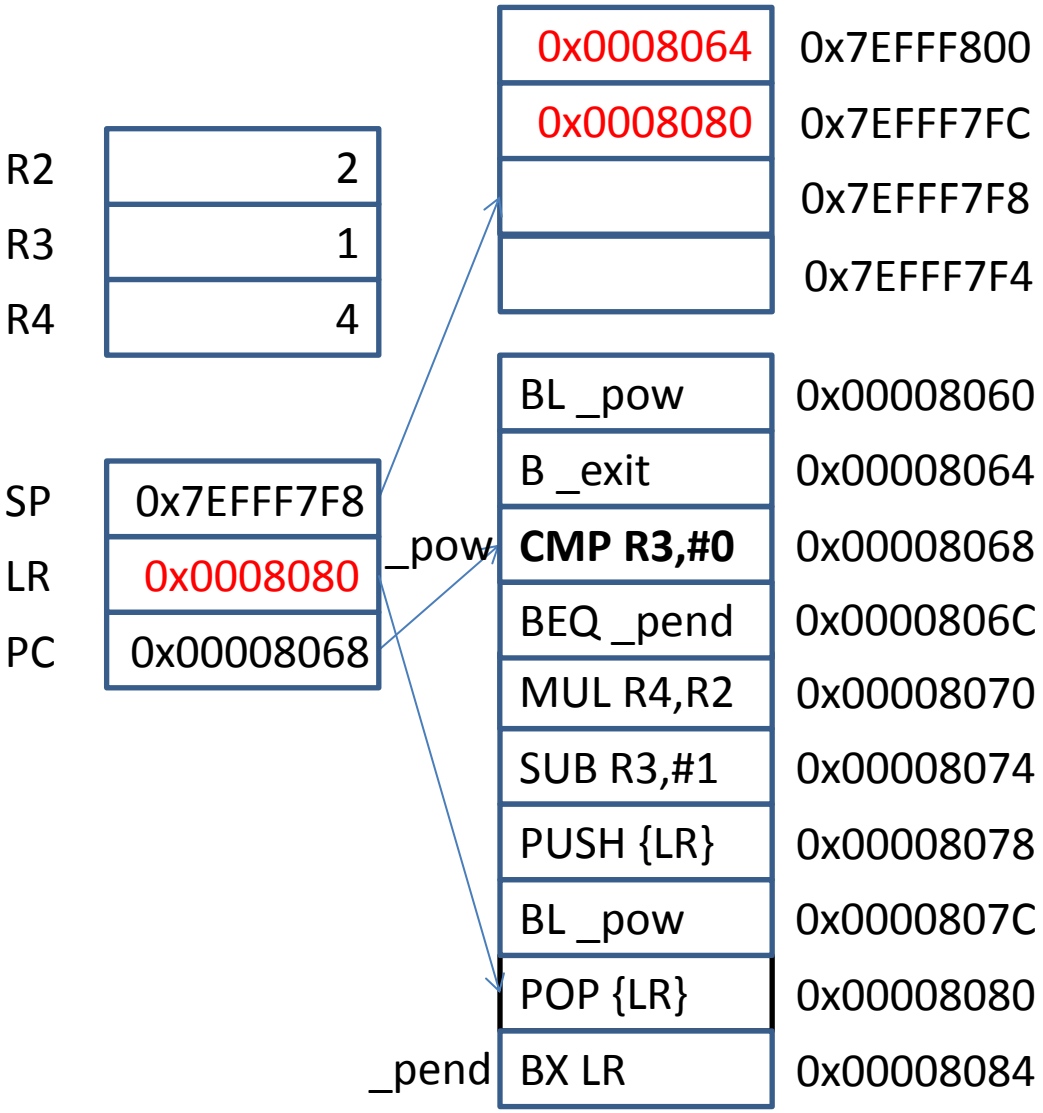
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



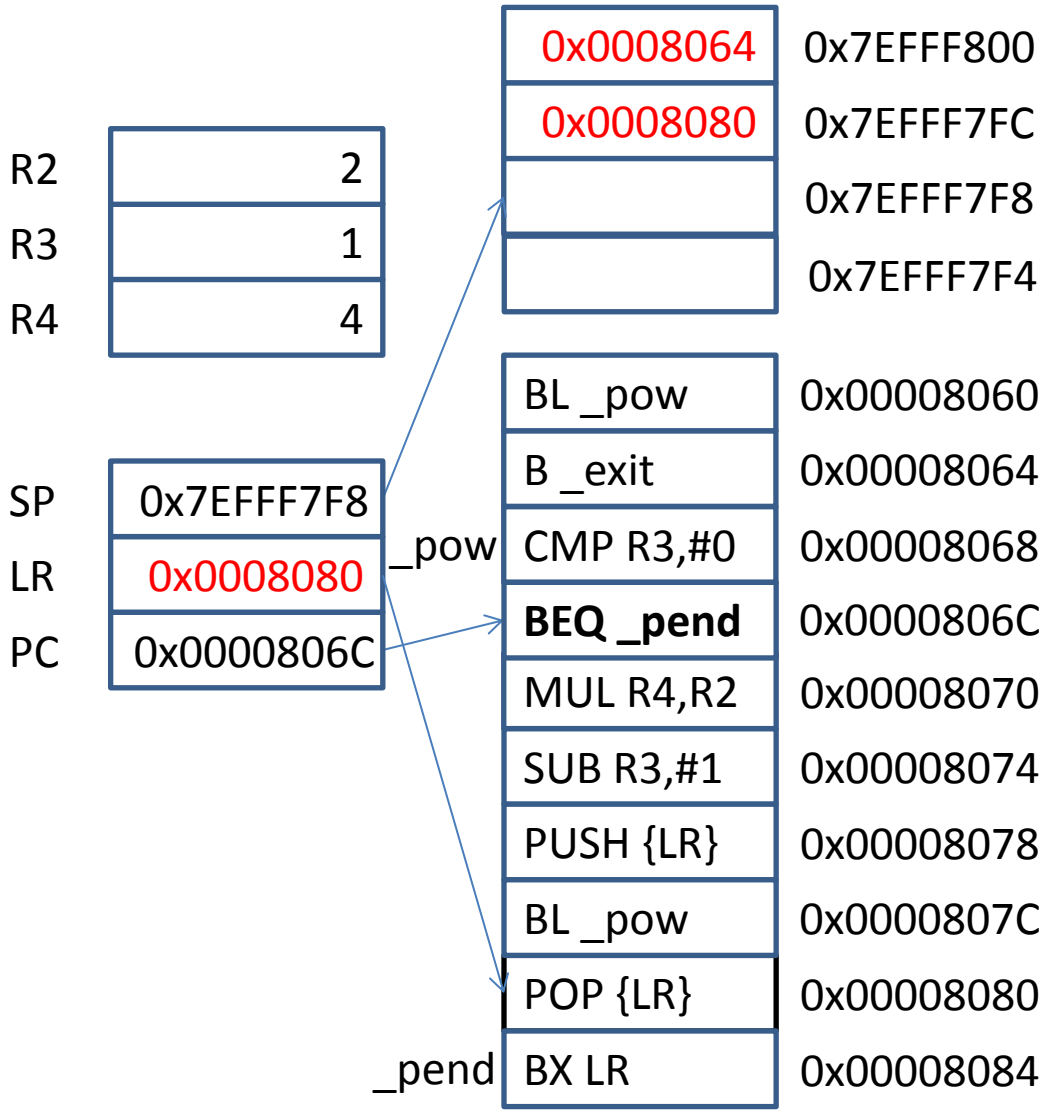
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



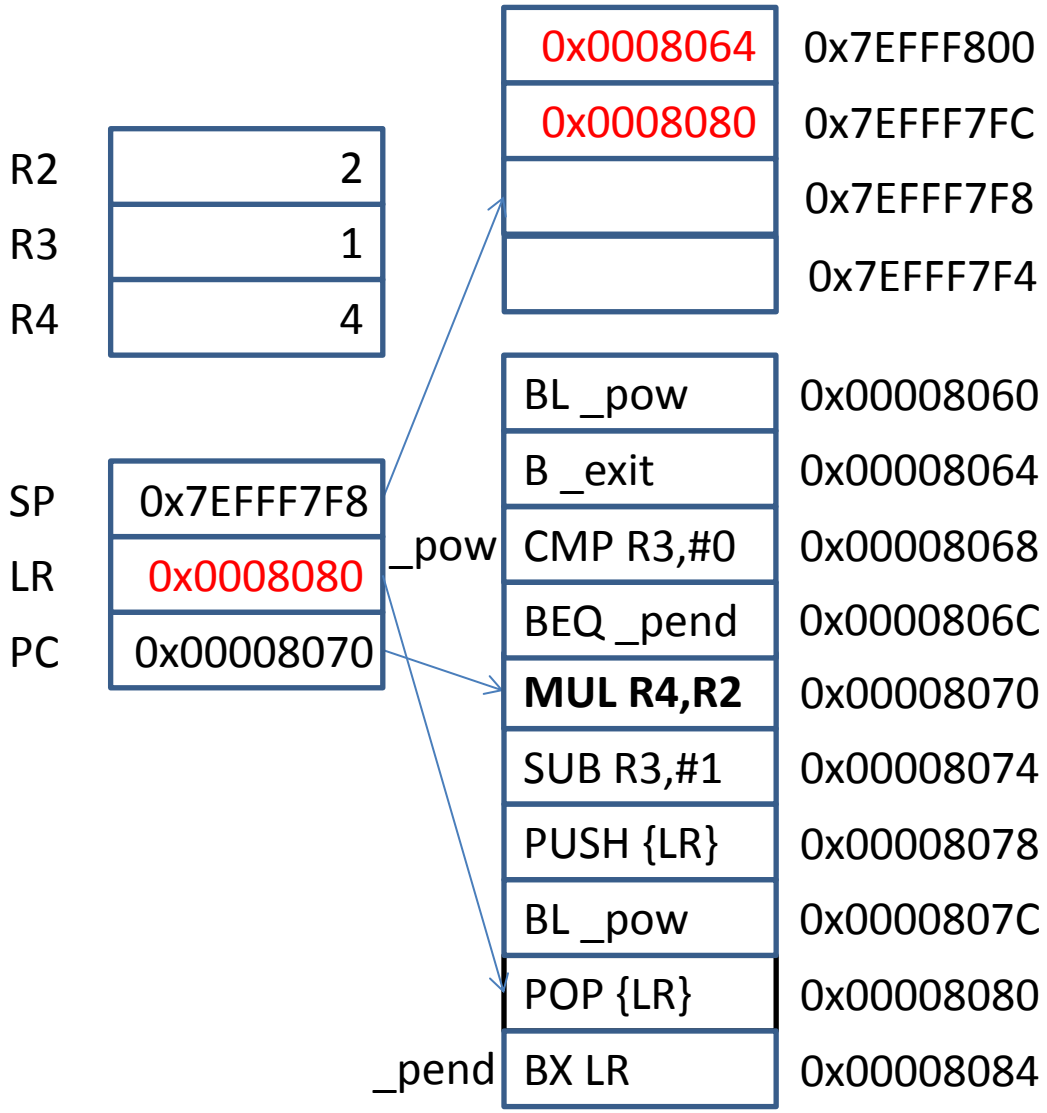
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



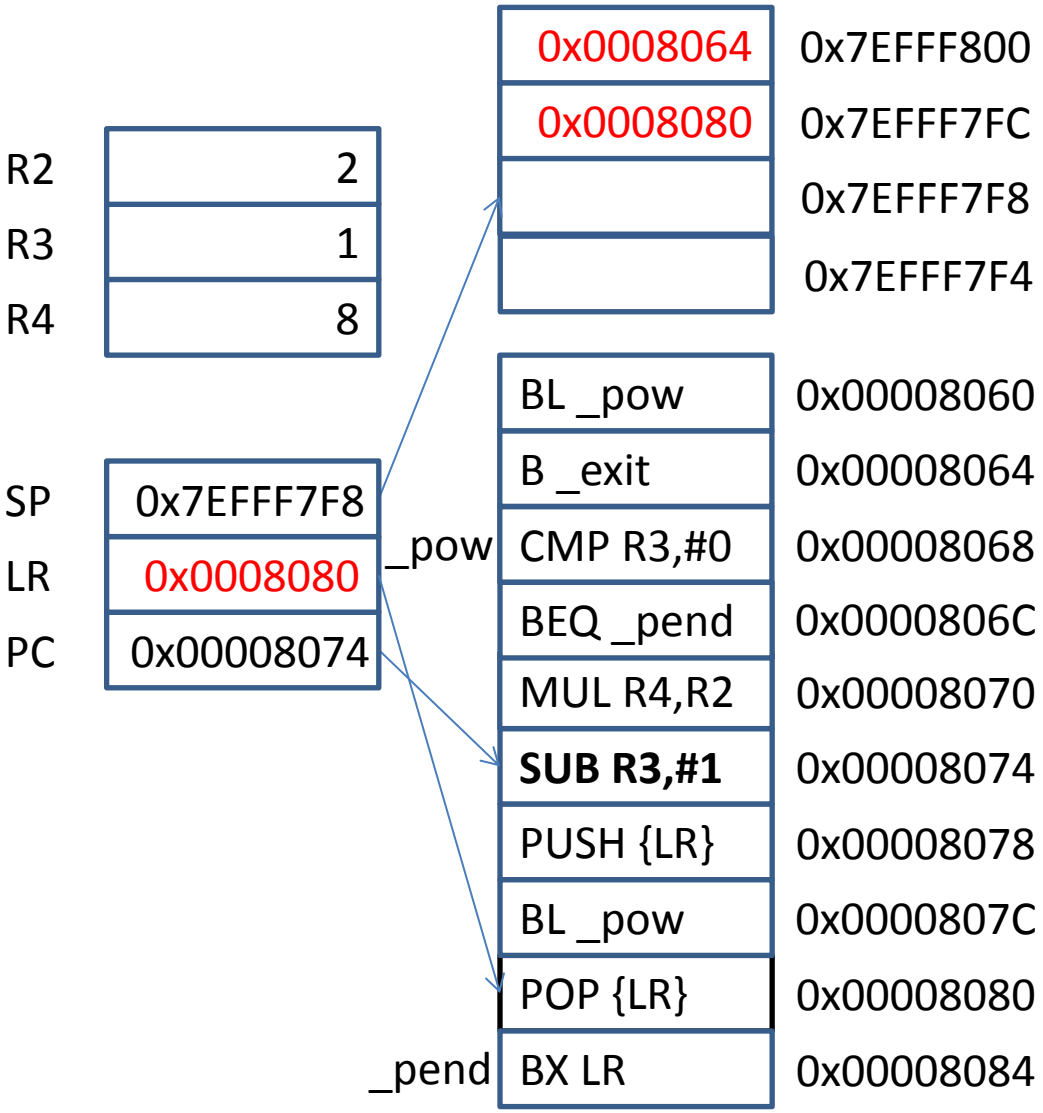
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



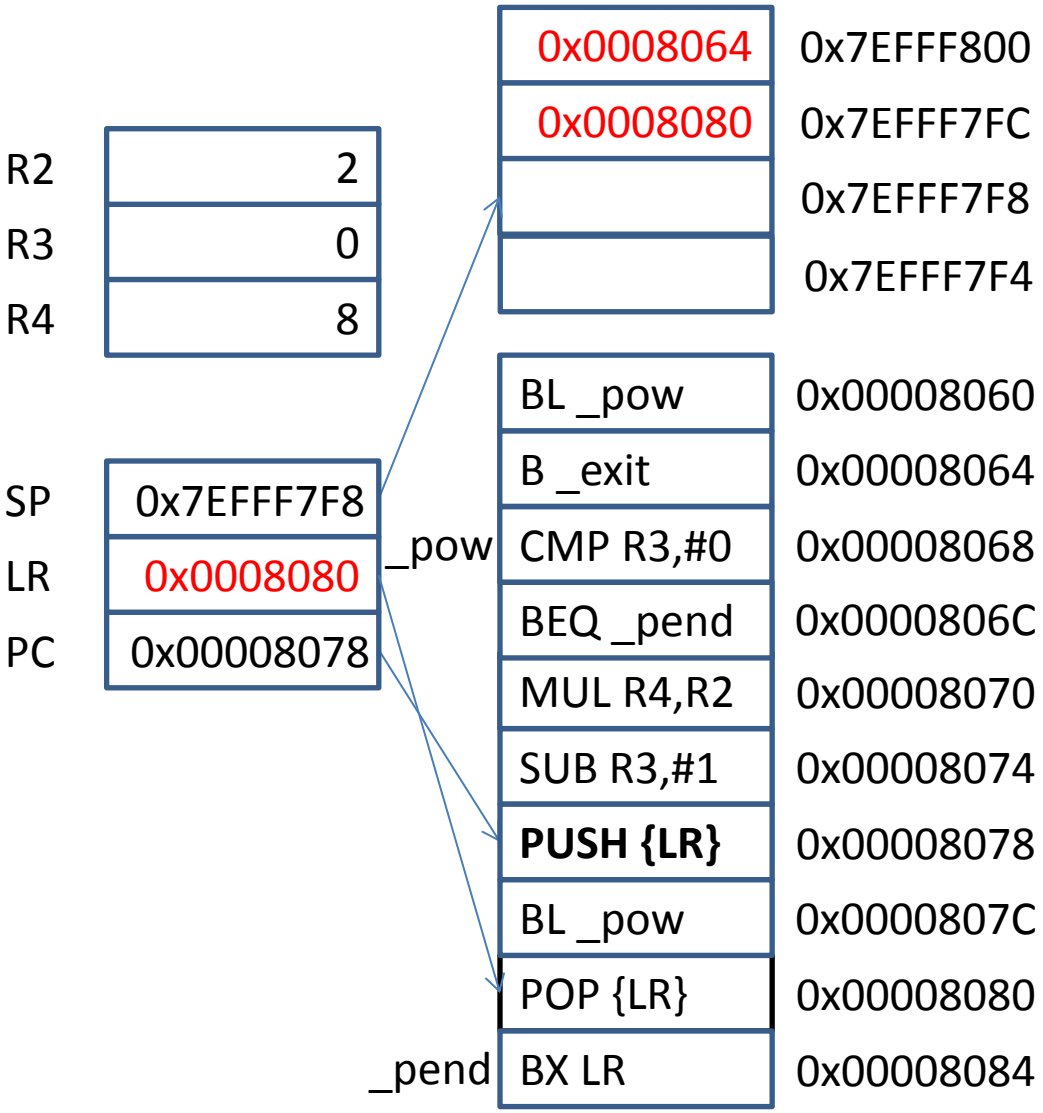
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



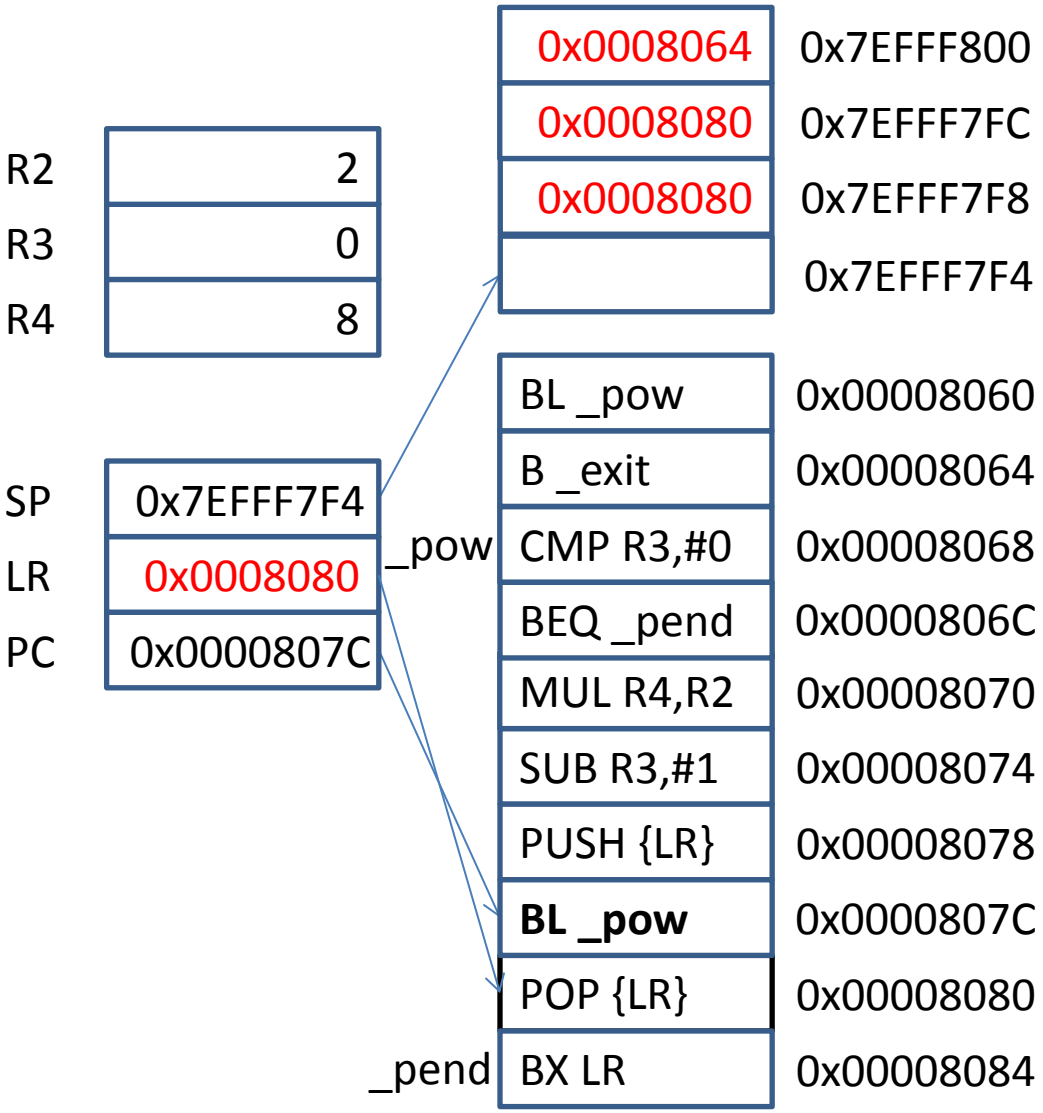
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



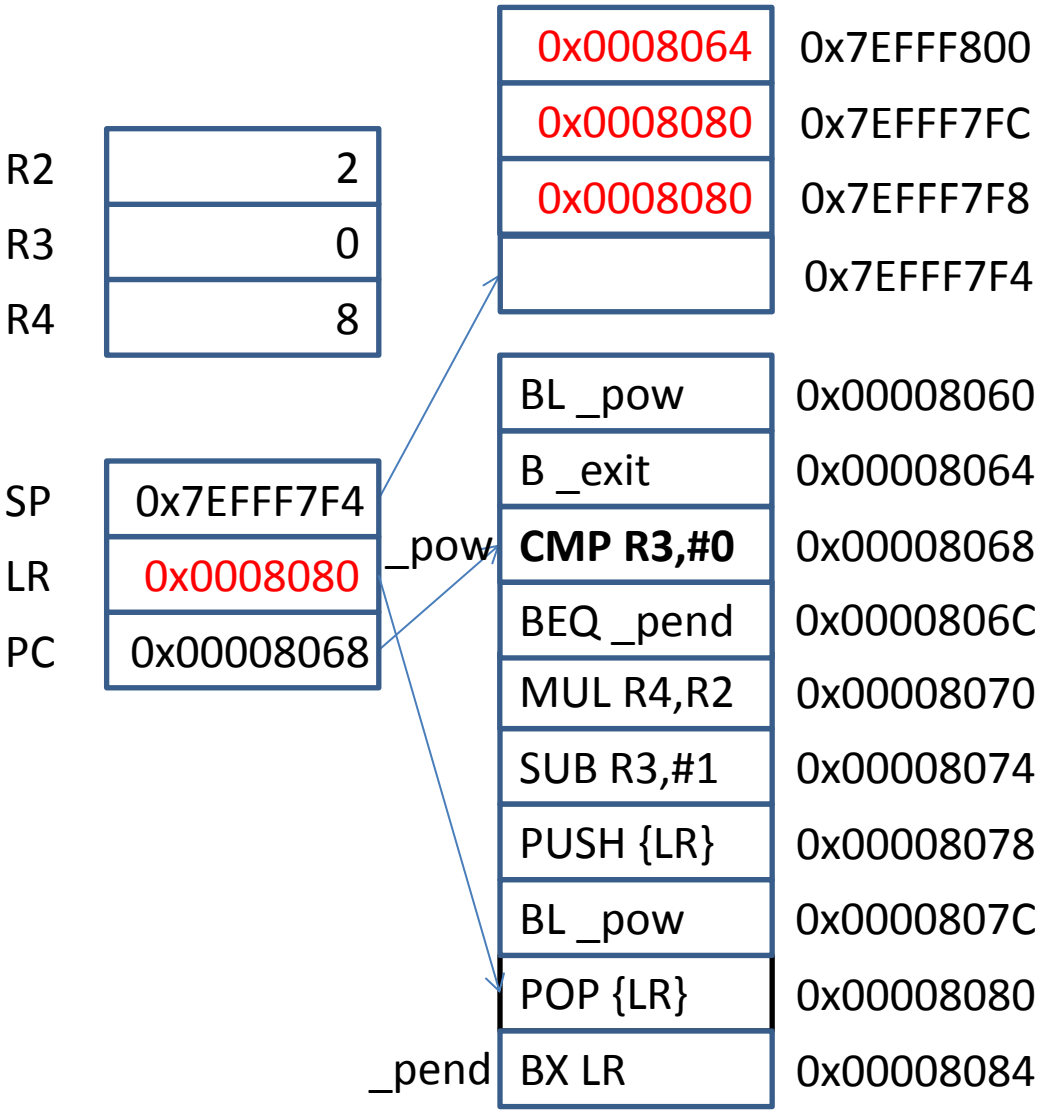
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



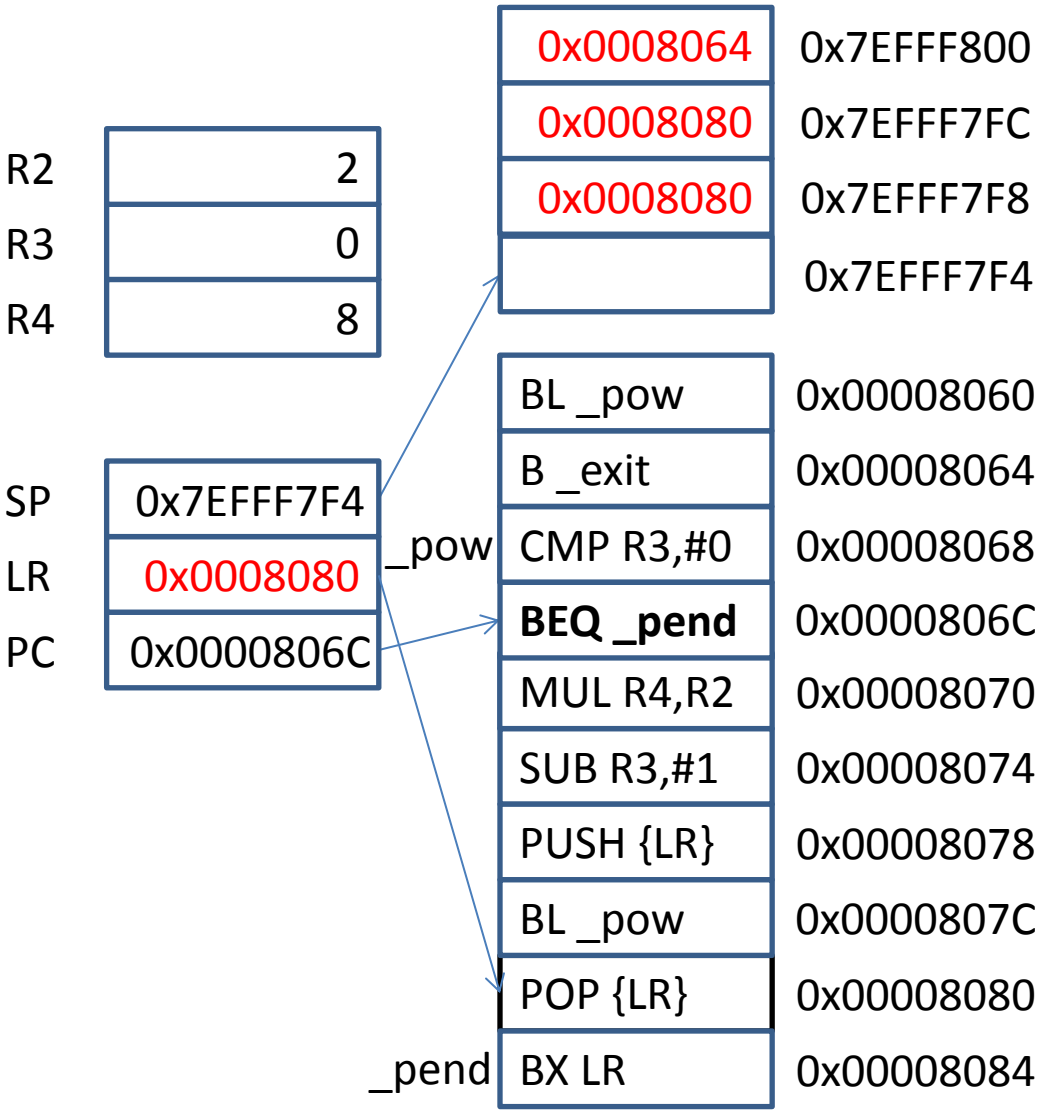
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



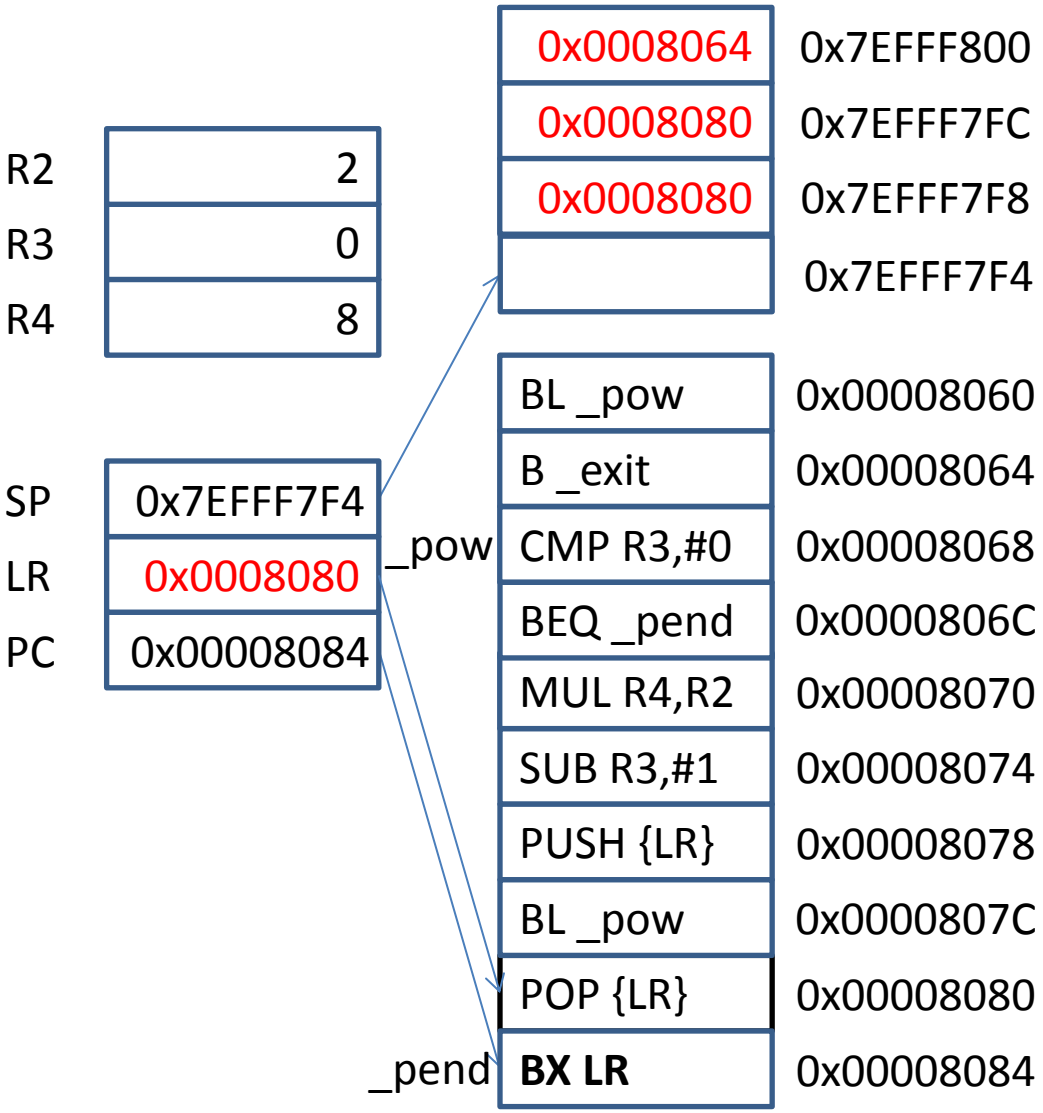
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



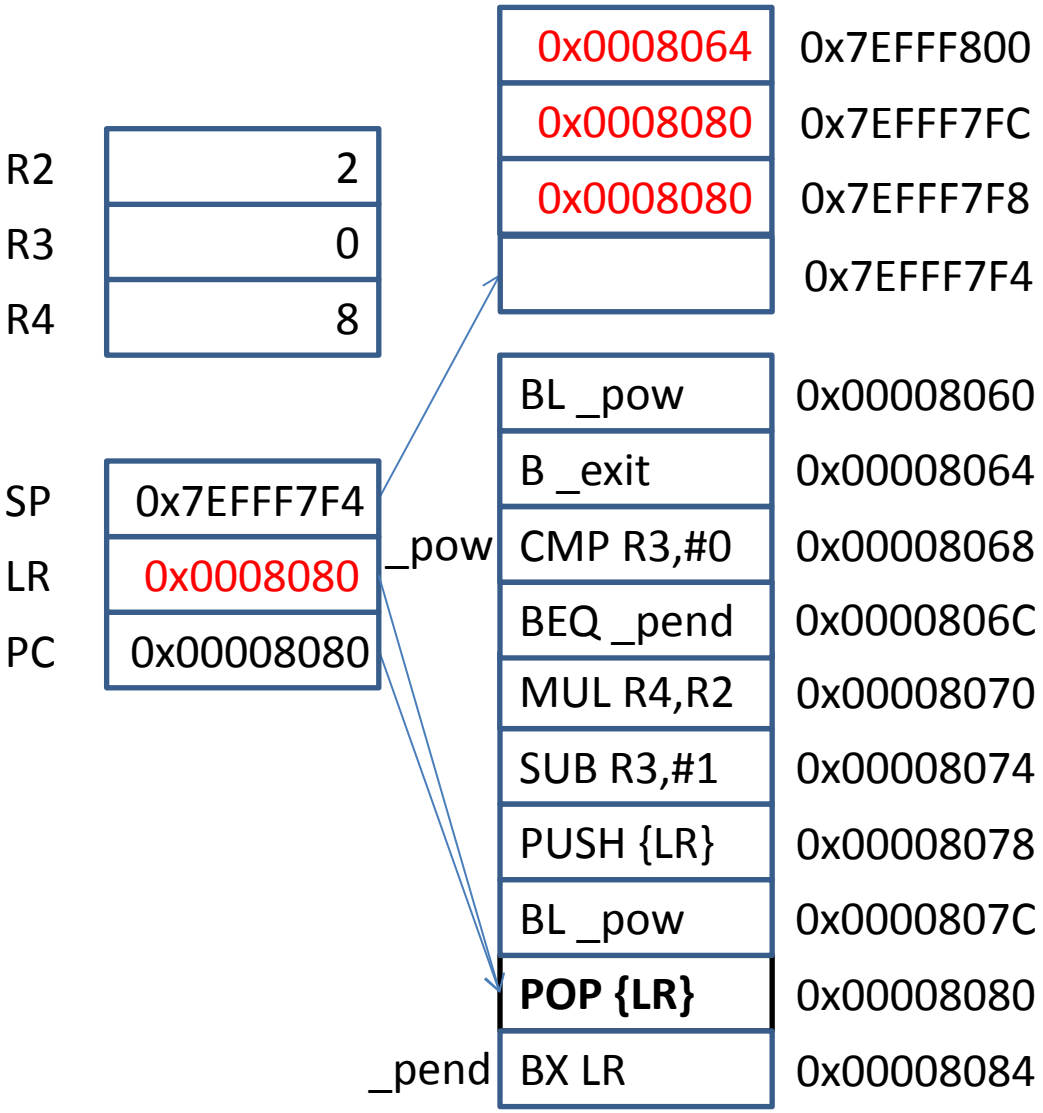
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



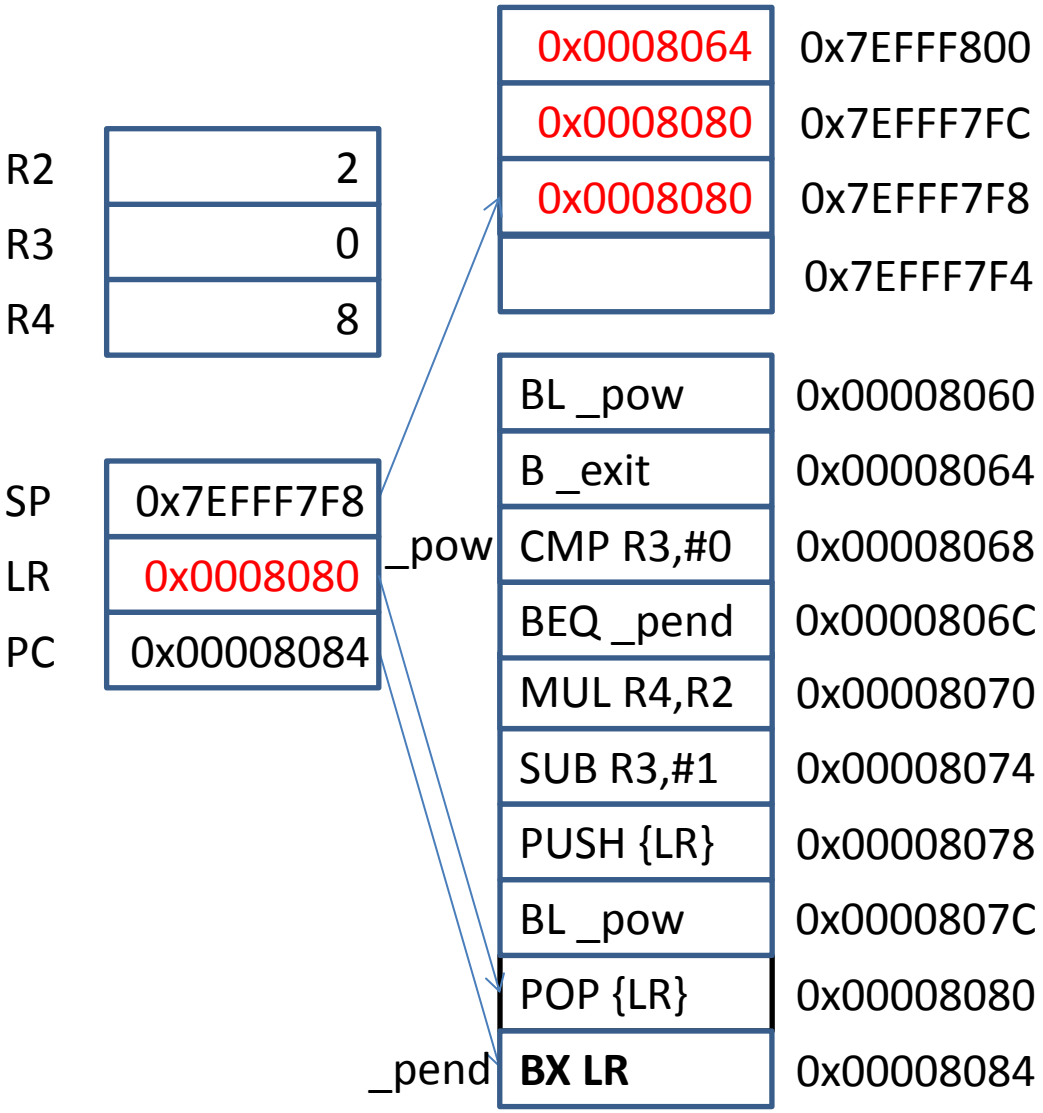
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



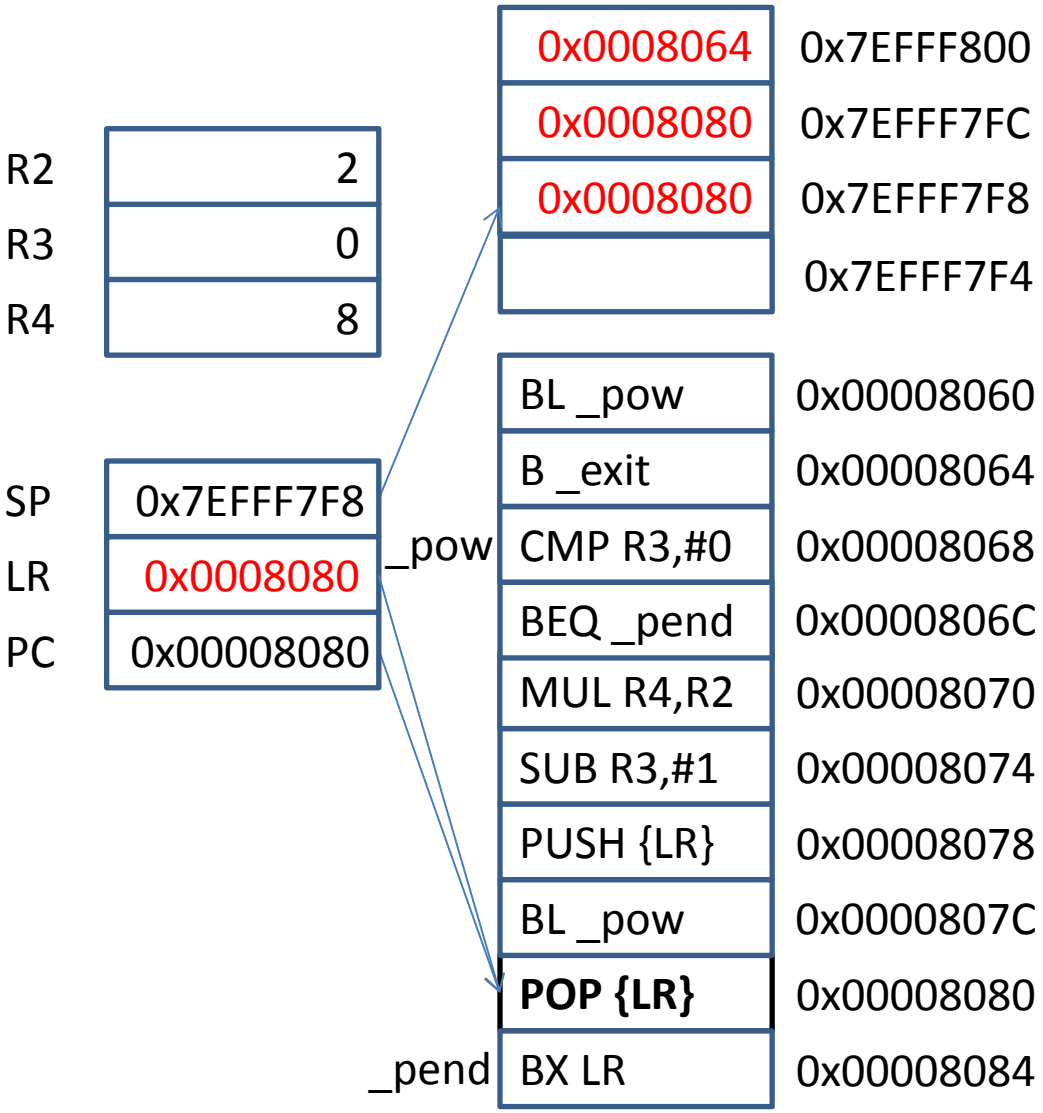
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



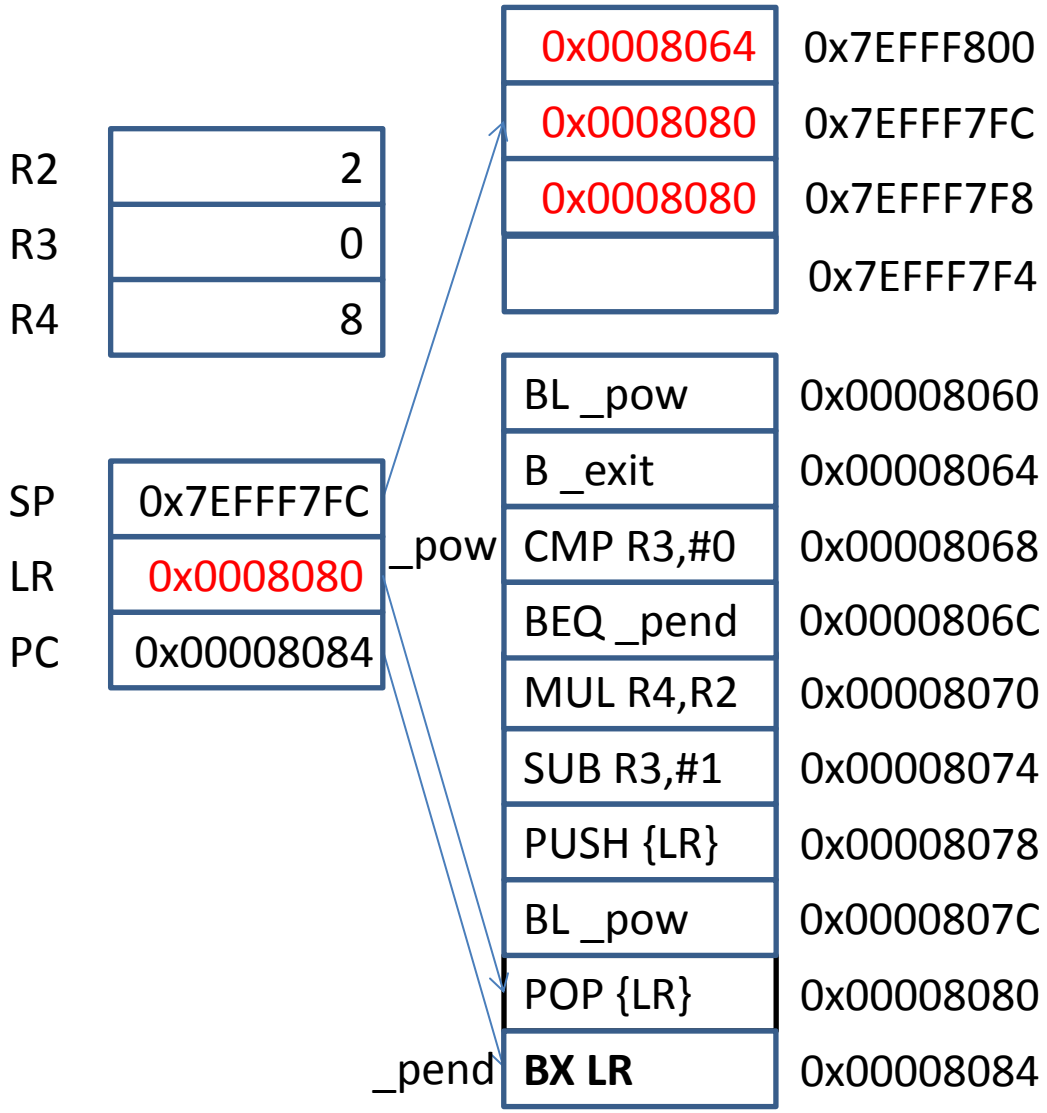
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



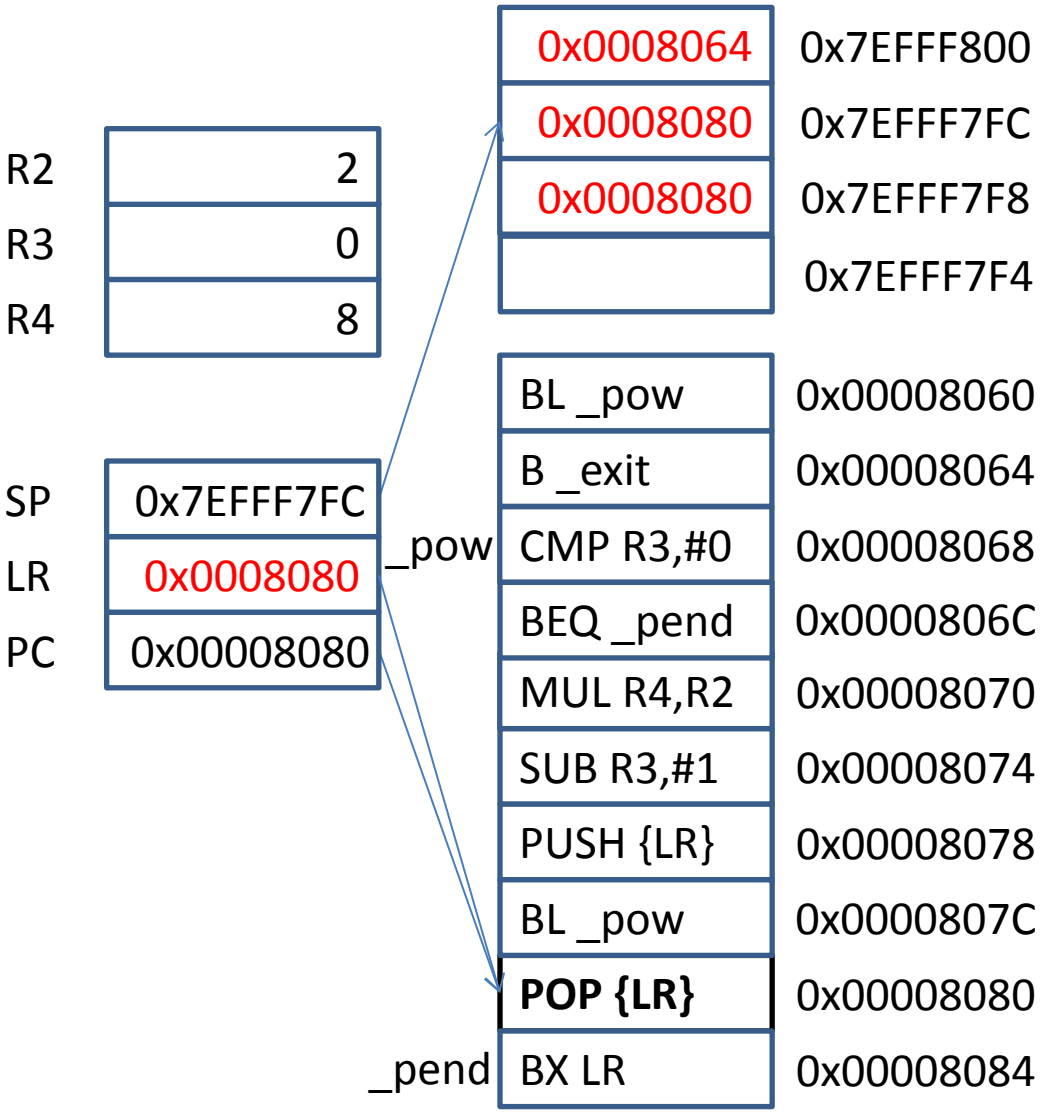
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



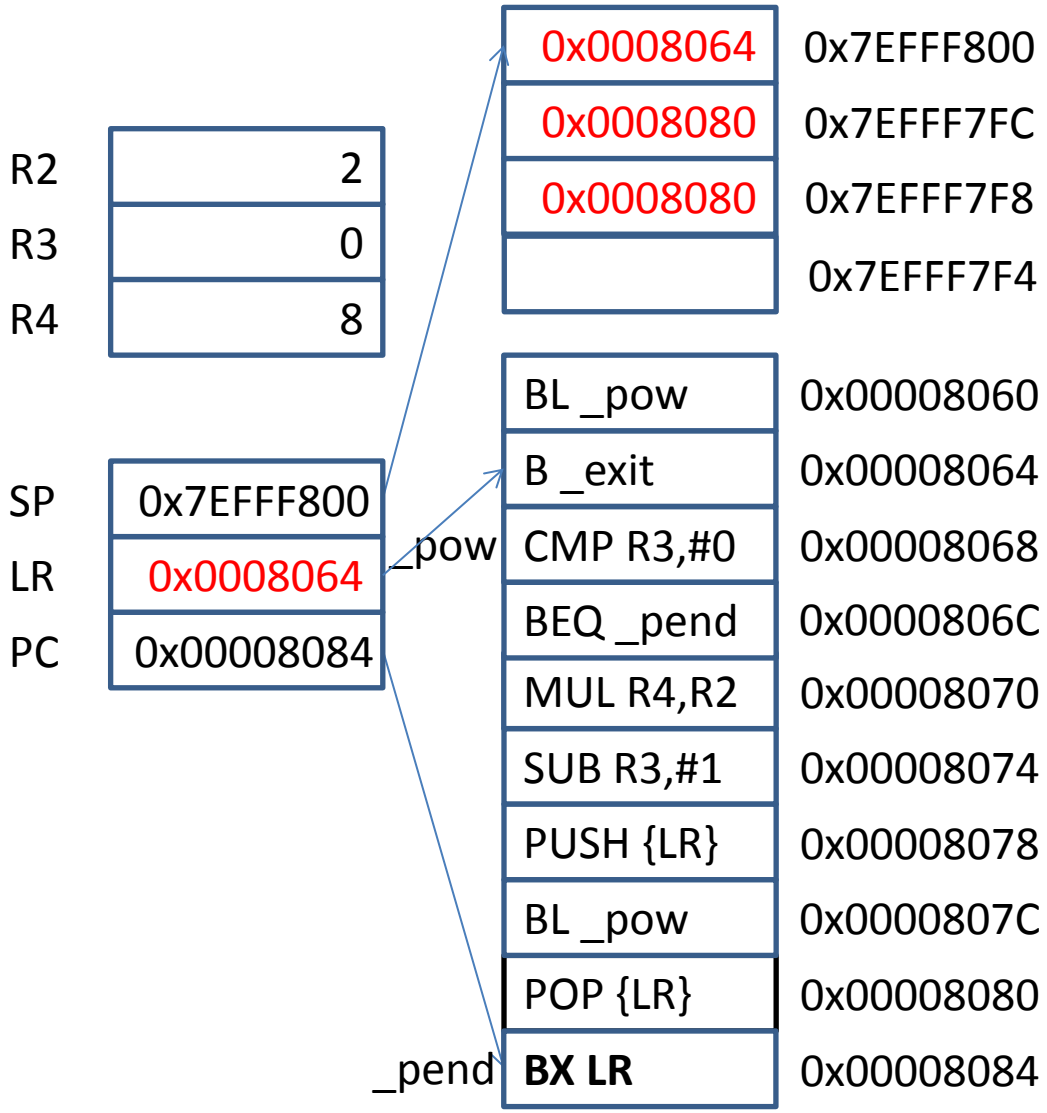
Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



Recursion: R4 = $R2^{R3}$

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```



Recursion: R4 = R2^R3

```
        MOV R2,#2
        MOV R3,#3
        MOV R4,#1
        BL _pow
        B _exit
_pow:   CMP R3,#0
        BEQ _pend
        MUL R4,R2
        SUB R3,#1
        PUSH {LR}
        BL _pow
        POP {LR}
_pend:  BX LR
_exit:  ...
```