Using the Stack for Procedure Arguments and Local Variables

- Initialize Y (typically after **PUSH**ing Y)
- Allocate local variables before **PUSH**ing any registers
- Access arguments and local variables with Y + n

Example

Function `Fnc` is passed 3 single byte arguments (arg1, arg2, and arg3), has 3 local variables on the stack (var1 and var3 are word variables and var2 is a byte variable), and needs to preserve registers Y and R16

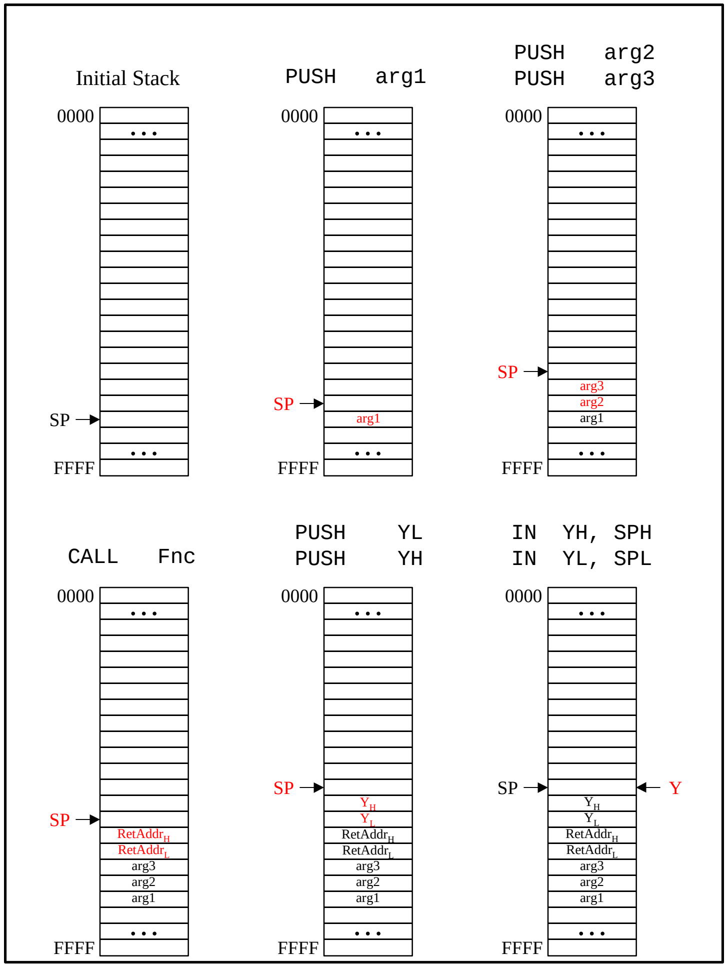
Calling Code

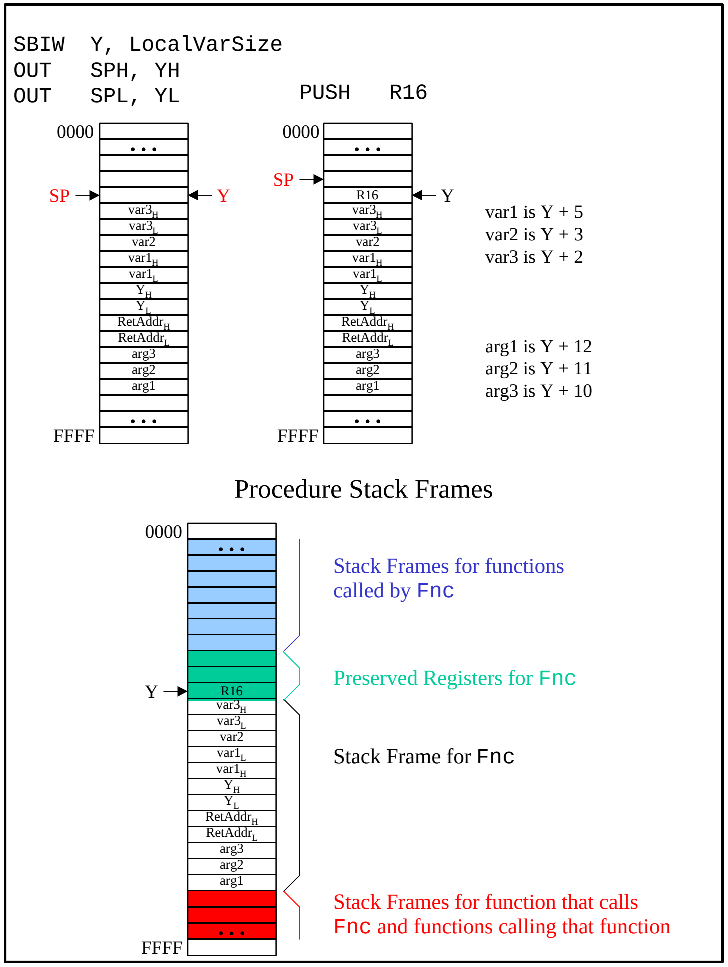
```
PUSH  arg1              ;put arguments on stack
PUSH  arg2
PUSH  arg3
CALL  Fnc               ;call function
POP   R0                ;remove arguments from stack
POP   R0
POP   R0
```

Function Code

```
     .EQU  LocalVarSize = 5  ;5 bytes of local variables

Fnc: PUSH  YL                ;save Y (frame pointer)
     PUSH  YH
     IN    YH, SPH           ;setup stack frame pointer
     IN    YL, SPL
     SBIW  Y, LocalVarSize   ;allocate local variables
     OUT   SPH, YH
     OUT   SPL, YL
     PUSH  R16               ;save trashed registers
       . . .                 ;execute function
     POP   R16               ;restore registers
     ADIW  Y, LocalVarSize   ;deallocate local variables
     OUT   SPH, YH
     OUT   SPL, YL
     POP   YH                ;restore Y (frame pointer)
     POP   YL
     RET                     ;return
```





```
SBIW  Y, LocalVarSize
OUT   SPH, YH
OUT   SPL, YL
```

```
PUSH    R16
```

var1 is Y + 5
var2 is Y + 3
var3 is Y + 2

arg1 is Y + 12
arg2 is Y + 11
arg3 is Y + 10

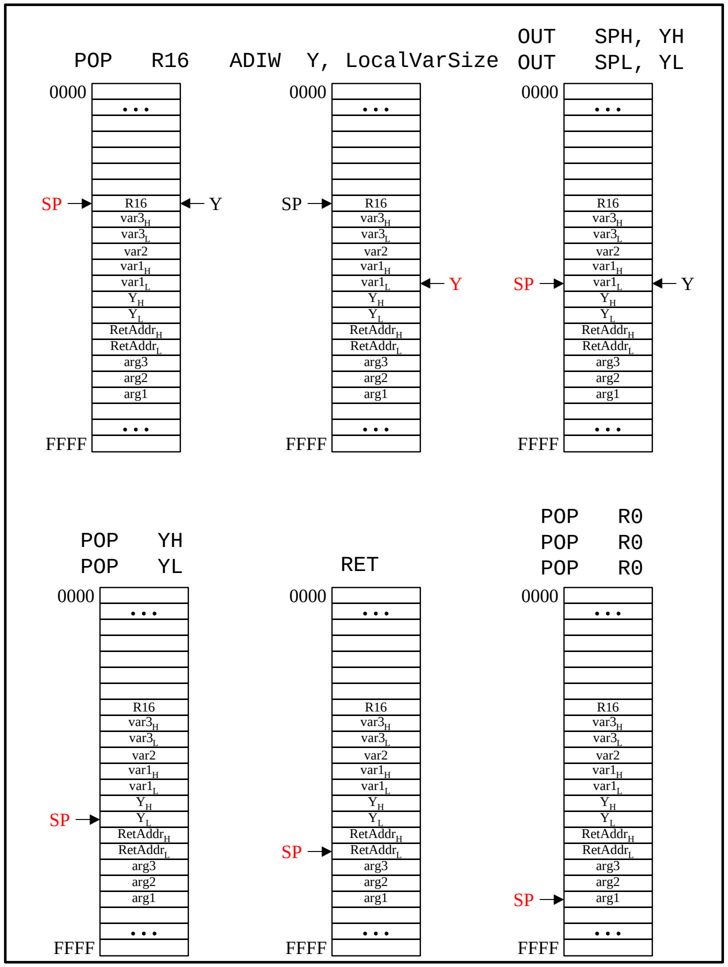
Procedure Stack Frames

Stack Frames for functions called by `Fnc`

Preserved Registers for `Fnc`

Stack Frame for `Fnc`

Stack Frames for function that calls `Fnc` and functions calling that function



```
POP    R16
```

```
ADIW  Y, LocalVarSize
```

```
OUT    SPH, YH
OUT    SPL, YL
```

```
POP    YH
POP    YL
```

```
RET
```

```
POP    R0
POP    R0
POP    R0
```