Stack operations (using PUSH and POP)

```
int mn () {
int s;
s = sm(1,2);
```

```
; values already on the stack
; assume R0=1, R1=2
PUSH R1        ;argument 2
PUSH R0        ;argument 1
JSR sm         ;function
```

stack before call

xEFFF	mn ret value
	? ret address
	? frame pointer
R5	mn local s
	parm = 2
R6	parm = 1
xEFF4	

```
int sm (int a, int b) {
int c;
```

```
PUSH R0        ; ret value = ?
PUSH R7        ; return address
PUSH R5        ; dyn link addr
ADD R5,R6,#-1  ; new DLA
PUSH R0        ; c = ?
```

stack after entering

xEFFF	mn ret value
	? ret address
	? frame pointer
	mn local s
	parm = 2
	parm = 1
	sm return value
	mn return addr
	mn frame pointer
R6,R5	sm local c
xEFF4	

```
int sm (int a, int b) {
int c;
...
return c;
}
```

```
POP R0         ; sm local c
POP R5         ; restore mn R5
POP R7         ; restore mn R7
STR R0,R6,#0   ; store ret value
JMP R7         ; return to mn
```

Stack before exiting

xEFFF	mn ret value
	? ret address
	? frame pointer
R5	mn local s
	parm = 2
	parm = 1
R6	sm return value
	mn return addr
	mn dyn link addr
	sm local c
xEFF4	

```
int mn () {
int s;
...
s=sm(1,2);
}
```

```
POP R0         ; get return value
STR R0,R5,#0   ; store local s
POP R0         ; remove parm
POP R0         ; remove parm
```

stack after return

xEFFF	mn ret value
	? ret address
	? frame pointer
R6,R5	mn local s=ret val
	parm = 2
	parm = 1
	sm return value
xEFF4	

Stack operations (optimized or generated by a compiler)

```
int mn () {
int s;
s = sm(1,2);
```

```
; values already on stack
; assume R0=1, R1=2
ADD R6,R6,#-2  ; push stack
STR R1,R6,#1   ;argument 2
STR R0,R6,#0   ;argument 1
JSR sm         ;function
```

stack before call

xEFFF	mn ret value
	? ret address
	? frame pointer
R5	mn local s
	parm = 2
R6	parm = 1
xEFF4	

```
int sm (int a, int b) {
int c;
```

```
ADD R6,R6,#-4  ; push stack
STR R7,R6,#2   ; return addr
STR R5,R6,#1   ; old FP
ADD R5,R6,#0   ; new FP
```

stack after entering

xEFFF	mn ret value
	? ret address
	? frame pointer
	mn local s
	parm = 2
	parm = 1
	sm return value
	mn return addr
	mn frame pointer
R6,R5	sm local c
xEFF4	

```
int sm (int a, int b) {
int c;
...
return c;
}
```

```
LDR R0,R5,#0   ; sm local c
LDR R7,R5,#2   ; reset mn R7
LDR R5,R5,#1   ; reset mn R5
ADD R6,R5,#-3  ; reset stack
STR R0,R6,#0   ; store ret val
JMP R7         ; return to mn
```

Stack before exiting

xEFFF	mn ret value
	? ret address
	? frame pointer
R5	mn local s
	parm = 2
	parm = 1
R6	sm return value
	mn return addr
	mn frame pointer
	sm local c
xEFF4	

```
int mn () {
int s;
...
s=sm(1,2);
}
```

```
LDR R0,R6,#0   ; get ret value
ADD R6,R5,#0   ; reset stack
STR R0,R5,#0   ; store local s
```

stack after return

xEFFF	mn ret value
	? ret address
	? frame pointer
R6,R5	mn local s=ret val
	parm = 2
	parm = 1
	sm return value
xEFF4	