5 Global, Dynamic Local, and Static Local Variables

Global Variables

```
; ex0501.a  Global variables

startup:  bl main
          halt
;============================================================
                           ; #include <stdio.h>
x:        .word 2          ; int x = 2, y;
y:        .word 0

main:     push lr          ; int main()
          push fp          ; {
          mov fp, sp

          ld r0, x         ;    y = x;
          st r0, y

          ld r0, y         ;    printf("%d\n", y);
          dout r0
          nl

          mov r0, 0        ;    return 0;
          mov sp, fp
          pop fp
          pop lr
          ret
                           ; }
```

Default value of global variables is 0

Rule: C global variables for which an initial value is not specified get the default initial value 0.

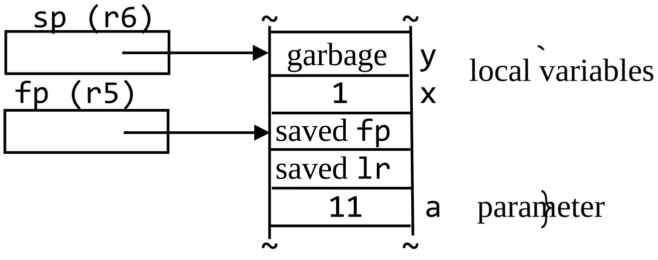
Rule: Global variables are accessed with the `ld` and `st` instructions.

Dynamic Local Variables

```
; ex0502.a  Dynamic local variables

startup:  bl main
          halt
;===========================================================
                            ; #include <stdio.h>
f:        push lr           ; void f(int a)
          push fp           ; {
          mov fp, sp

          mov r0, 1         ;    int x = 1, y;
          push r0
          sub sp, sp, 1

          ldr r0, fp, -1    ;    y = x + a;
          ldr r1, fp, 2
          add r0, r0, r1
          str r0, fp, -2

          ldr r0, fp, -2    ;    printf("%d\n", y);
          dout r0
          nl

          mov sp, fp        ; }
          pop fp
          pop lr
          ret
;===========================================================
main:     push lr           ; int main()
          push fp           ; {
          mov fp, sp

          mov r0, 11        ;    f(11);
          push r0
          bl f
          add sp, sp, 1

          mov r0, 0         ;    return 0;
          mov sp, fp
          pop fp
          pop lr
          ret
                            ; }
```

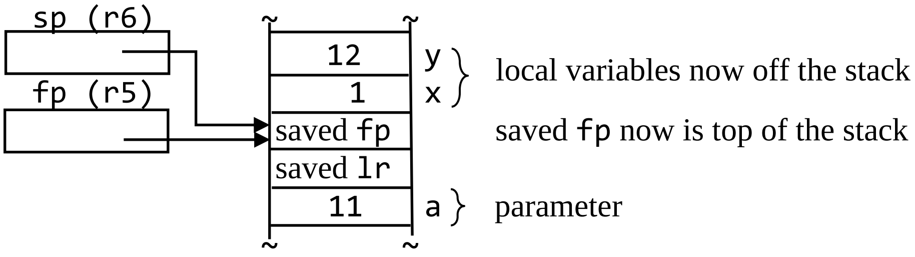
Creates and initializes x

Creates y



Rule: Access parameters with positive offsets; access local variables with negative offsets.

```
            mov sp, fp
```


sp (r6)
fp (r5)
12
1
saved fp
saved lr
11
y
x
a
local variables now off the stack
saved fp now is top of the stack
parameter

```
            pop fp
            pop lr
            ret
```

Rule: A function creates its dynamic local variables on entry and destroys them on exit. The values of dynamic local variables are not retained between calls of a function.

Local variable (no default initialization):

```
     int y;
```

Code is not

```
         mov r0, 0        ; create and initialize y
         push r0
```

In, it is

```
         sub sp, sp, 1    ; reserve slot on stack for y
```

Static Local Variables

```c
// ex0503.c
#include <stdio.h>
int x = 5;
void f()
{
   static int x;
   printf("%d\n", x);   // displays 0
}
//===================
void g()
{
   printf("%d\n", x);   // displays 5
}
//===================
int main()
{
   static int x = 3;
   printf("%d\n", x);   // displays 3
   f();
   g();
   return 0;
}
```

static Local Variables Created with .word Directives

Incorrect!!!

```
x:         .word 5     ; global x
x:         .word 0     ; x in f
x:         .word 3     ; x in main
```

Correct!!!

1. “@s”   (“@” indicates compiler generated label)
2. a sequence number (0, 1, 2, …)
3. an underscore
4. the name of the variable in the C program

```
@s0_x:     .word 0          ; x in f
@s1_x:     .word 3          ; x in main


           ld r0, @s0_x     ; load the x in f
           dout r0
           nl
```

The `.word` directive for the global `x`, on the other hand, is labeled with `x`:

```
x:         .word 5          ; global x
```

```
; ex0503.a  Static local variables

startup:  bl main
          halt             ; back to operating system
;===========================================================
                           ; #include <stdio.h>
x:        .word 5          ; int x = 5;
f:        push lr          ; void f()
          push fp          ; {
          mov fp, sp

                           ;    static int x;
```

No code for the declaration of a static local variable

```

          ld r0, @s0_x     ;    printf("%d\n", x);
          dout r0
          nl

          mov sp, fp       ; }
          pop fp
          pop lr
          ret
;===========================================================
g:        push lr          ; void g()
          push fp          ; {
          mov fp, sp

          ld r0, x         ;    printf("d\n", x);
```

Accessing global variable using C-level name

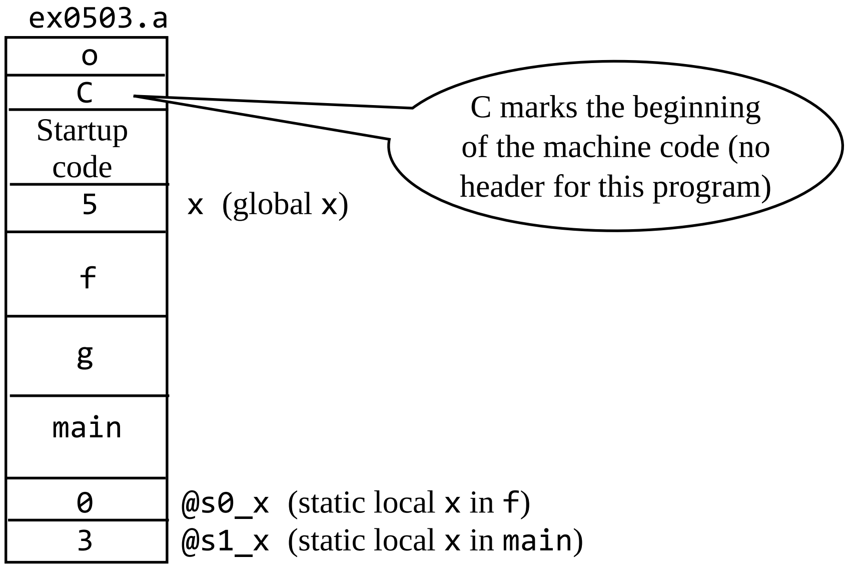
```
          dout r0
          nl

          mov sp, fp       ; }
          pop fp
          pop lr
          ret
;===========================================================
main:     push lr          ; int main()
          push fp          ; {
          mov fp, sp

                           ;    static int x = 3;
```

No code for the declaration of a static local variable

```

          ld r0, @s1_x     ;    printf("%d\n", x);
          dout r0
          nl

          bl f            ;    f();

          bl g            ;    g();

          mov r0, 0        ;    return 0;
          mov sp, fp
```

```
            pop fp
            pop lr
            ret
                           ; }
@s0_x:      .word 0
@s1_x:      .word 3
```

Static local variables created with `.word` directives. Labels start with unique prefix.


ex0503.a
o
C
Startup code
5
x (global x)
f
g
main
0
@s0_x (static local x in f)
3
@s1_x (static local x in main)
C marks the beginning of the machine code (no header for this program)