1.1. Runtime Management

When it comes to runtime management, we use the classic style where we allocate an activation record for each function call, beginning with the initial call to **main**. Each activation record holds the data for the functions' parameters and local variables and constants. We use the **regular frame pointer** to hold the address of the current function call. The activation record also holds the frame pointer of the previous activation record and the return address; that is, the address to jump to when the execution of the current function returns to the calling function. More specifically: the address of the instruction following the call in the calling function.

What makes it a little bit more complicated is that a function may be variadic; that is, it has a variable number of parameters, like **printf** and **scanf**. We introduce a second frame pointer: the **variadic frame pointer** for variadic functions. It holds the address of the activation record plus the size of the extra parameters in the call to a variadic function.

To sum it up: the activation record holds the return address, regular frame pointer, and variadic frame pointer of the calling function as well as the parameters, potential extra parameter in case of a variadic function, and auto and register variables and constants of the current function. The variadic frame pointer is undefined in case of a regular calling function. However, we must always have the variadic frame pointer in the activation record since the function can be called by both variadic and regular functions.

When a function calls another function (or itself recursively) the activation record of the called function is located at the address above the activation record of the calling function. The return address, regular and variadic frame pointers of the calling function are stored at the beginning of the activation record of the called function. When a function returns to the calling function, the regular and variadic frame pointers are restored to the values of the calling function, and a jump back to the return address occurs.

The return value is not part of the activation record. It is stored in a register for integral and pointer types and at the floating-point stack for floating types. In case of a struct or union, its address is returned in a register.

The return address of the activation record for the initial **main** call is initialized to zero. When main return and the return address is zero, an exit of the execution occurs instead of a return. But only for the first call, subsequent recursive calls to **main** have their own regular return addresses since it is quite possible to make recursive calls to **main**. For instance, the following program writes the number one to ten.

```
void main() {
  static count = 1;

  if (count <= 10) {
    printf("%d ", count++);
    main();
  }
}
```

Let us look at the following example.

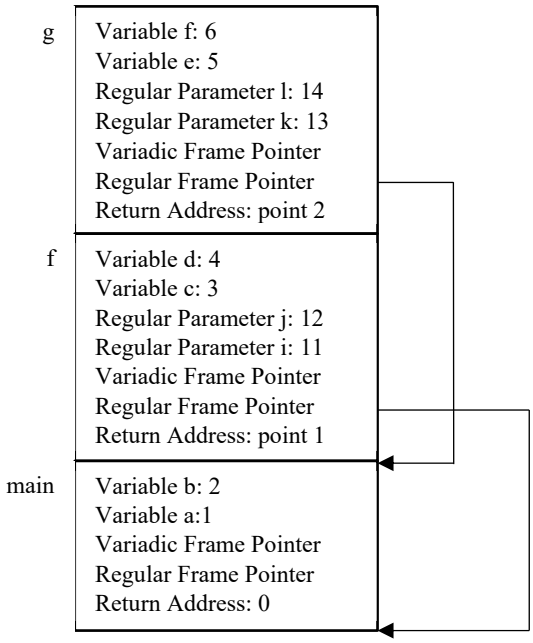
```
void main() {
  int a = 1, b = 2;
  f(11, 12);
}

void f(int i, int j) {
  int c = 3, d = 4;
  g(13, 14);
```

```
}

void g(int k, int l) {
  int e = 5, f = 6;
  // point
}
```

The program above has the call stack as follows when the execution of the program reached the point. The regular parameters refer to parameters defined in the parameter list. In the next example, we look into variadic parameter: extra parameters on calls to variadic functions.

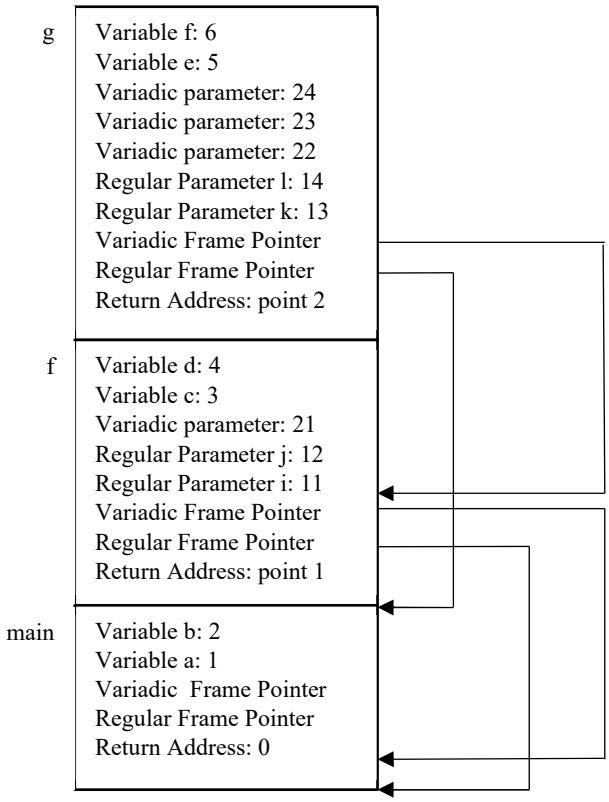


In the example above, the variadic frame pointer is ignored. In the following example, however, we let **f** and **g** be variadic functions. The variadic status is marked by three dots ('…').

```
void main() {
  int a = 1, b = 2;
  f(11, 12, 21);
}

void f(int i, int j, ...) {
  int c = 3, d = 4;
  g(13, 14, 22, 23, 24);
}

void g(int k, int l, ...) {
  int e = 5, f = 6;
  // point
}
```



The variadic parameters are stored after the regular parameters in the activation record. Similar to the previous example, the regular frame pointers point at the beginning of the activation record of the calling function. However, the variadic frame pointers points at the beginning of the activation record of the calling function, plus the size of the variadic parameters. For the sake of argument, let us assume that an integer value allocates four bytes. Then the variadic frame pointer of **f** is the regular frame pointer plus four bytes (one integer value). The variadic frame pointer of **g** the regular frame pointer plus twelve bytes (three integer values). When refereeing to parameters in a variadic function, the regular frame pointer is used, and when refereeing to variables, the variadic frame pointer is used. The variadic parameters are accessible by using the address of the last regular parameter (it has to be at least one) and reading the values by increase the address, which is the task of the macros **va_list**, **va_start**, and **va_arg** in the **stdarg** standard library.