Functions in C

Hyung-Sin Kim

Computing Foundations for Data Science

Graduate School of Data Science, Seoul National University

You already know what functions are.

Today we focus more on what happens when you call a function in C

Functions in C

- Declaration (function prototype)
  - Informs the compiler about relevant properties
    - Name, data type of return value, and type of input arguments

```
#include <stdio.h>
int Factorial(int n);

int main(void) {
    int number;
    int answer;
    printf("Input a number: ");
    scanf("%d", &number);
    answer = Factorial(number);
    printf("The factorial of %d is %d\n", number, answer);
}

int Factorial(int n) {
    int result = 1;
    for (int i=1; i <=n; i++)
        result += 1;
    return result;
}
```

Functions in C

- Declaration (function prototype)
  - Informs the compiler about relevant properties
    - Name, data type of return value, and type of input arguments
- Call
  - The caller must transmit proper arguments (number, type) to the callee
  - Calling is possible only when the callee is **declared before**

```
#include <stdio.h>
int Factorial(int n);

int main(void) {
    int number;
    int answer;
    printf("Input a number: ");
    scanf("%d", &number);
    answer = Factorial(number);
    printf("The factorial of %d is %d\n", number, answer);
}

int Factorial(int n) {
    int result = 1;
    for (int i=1; i <=n; i++)
        result += 1;
    return result;
}
```

Functions in C

- Declaration (function prototype)
  - Informs the compiler about relevant properties
    - Name, data type of return value, and type of input arguments
- Call
  - The caller must transmit proper arguments (number, type) to the callee
  - Calling is possible only when the callee is **declared before**
- Definition
  - Formal parameter list, initialized by the argument values passed by the caller
  - Any variable declared in the body is **local** to the function

```
#include <stdio.h>
int Factorial(int n);

int main(void) {
    int number;
    int answer;
    printf("Input a number: ");
    scanf("%d", &number);
    answer = Factorial(number);
    printf("The factorial of %d is %d\n", number, answer);
}

int Factorial(int n) {
    int result = 1;
    for (int i=1; i <=n; i++)
        result += 1;
    return result;
}
```

Functions in C

- Declaration (function prototype)
  - Informs the compiler about relevant properties
    - Name, data type of return value, and type of input arguments
- Call
  - The caller must transmit proper arguments (number, type) to the callee
  - Calling is possible only when the callee is **declared before**
- Definition
  - Formal parameter list, initialized by the argument values passed by the caller
  - Any variable declared in the body is **local** to the function
  - Return value goes back to the caller
    - If there is no return, return type is **void**

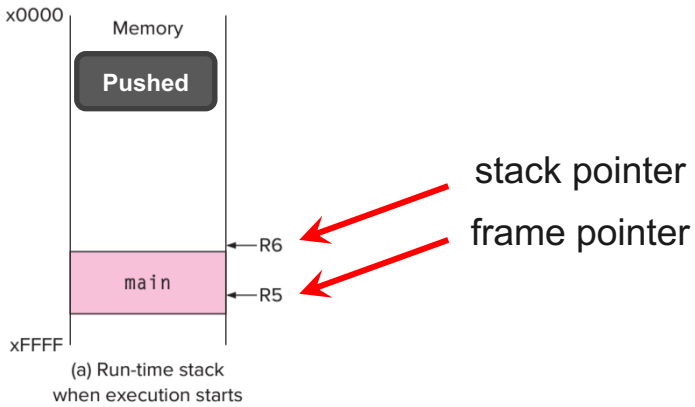
```
#include <stdio.h>
int Factorial(int n);

int main(void) {
    int number;
    int answer;
    printf("Input a number: ");
    scanf("%d", &number);
    answer = Factorial(number);
    printf("The factorial of %d is %d\n", number, answer);
}

int Factorial(int n) {
    int result = 1;
    for (int i=1; i <=n; i++)
        result += 1;
    return result;
}
```

Function Implementation in C – Example

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);    // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);    // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

Function Implementation in C – Overview

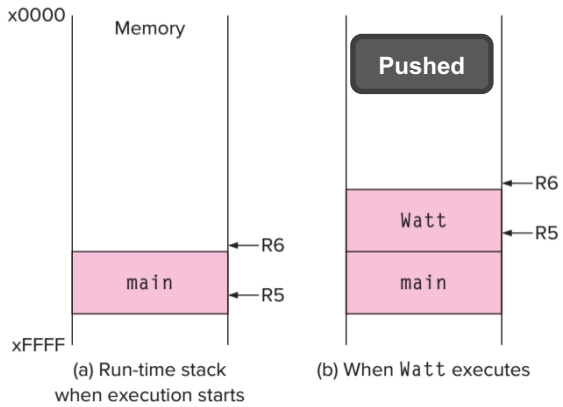
```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```



(a) Run-time stack when execution starts

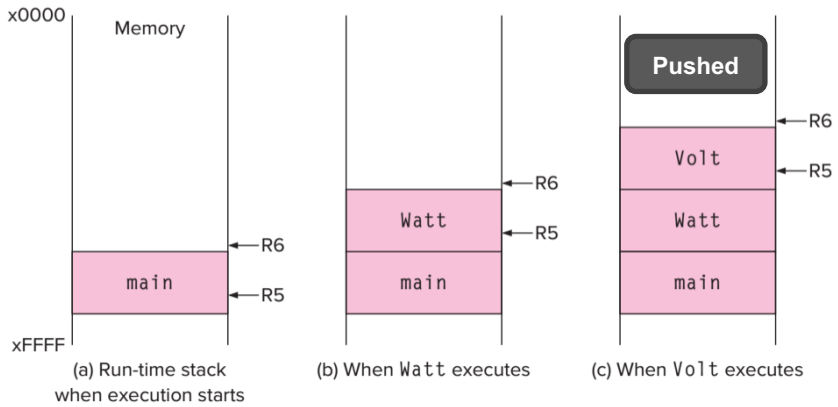
Function Implementation in C – Overview

```
int main(void) {
    int a;
    int b;
    b = Watt(a);    // main calls Watt first
    b = Volt(a, b);    // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);    // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```



Function Implementation in C – Overview

```
int main(void) {
    int a;
    int b;
    b = Watt(a);    // main calls Watt first
    b = Volt(a, b);    // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);    // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```



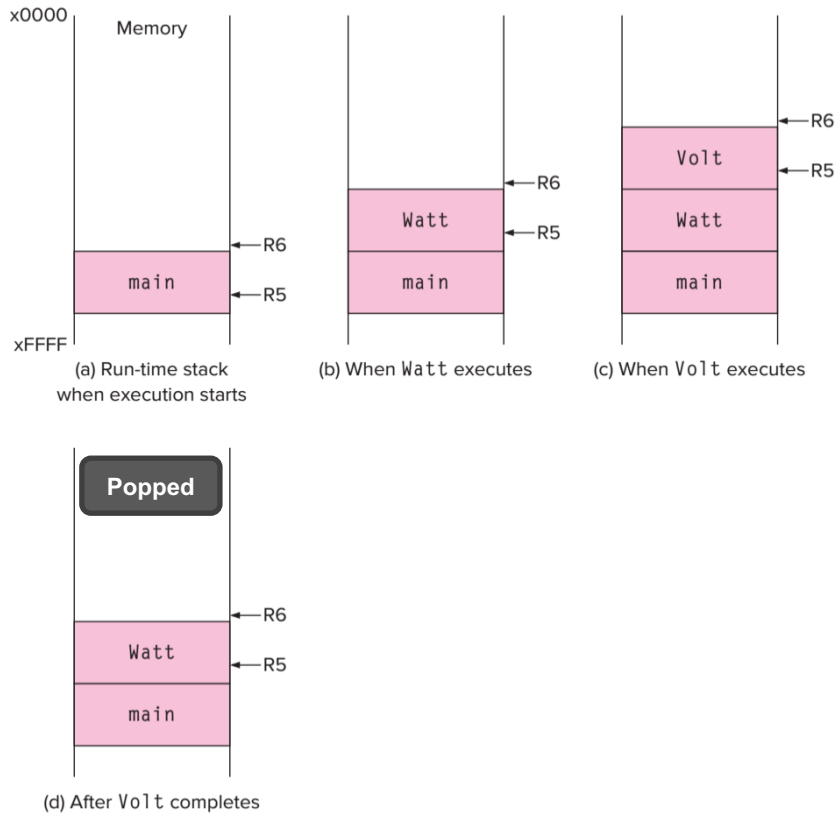
(a) Run-time stack when execution starts

(b) When `Watt` executes

(c) When `Volt` executes

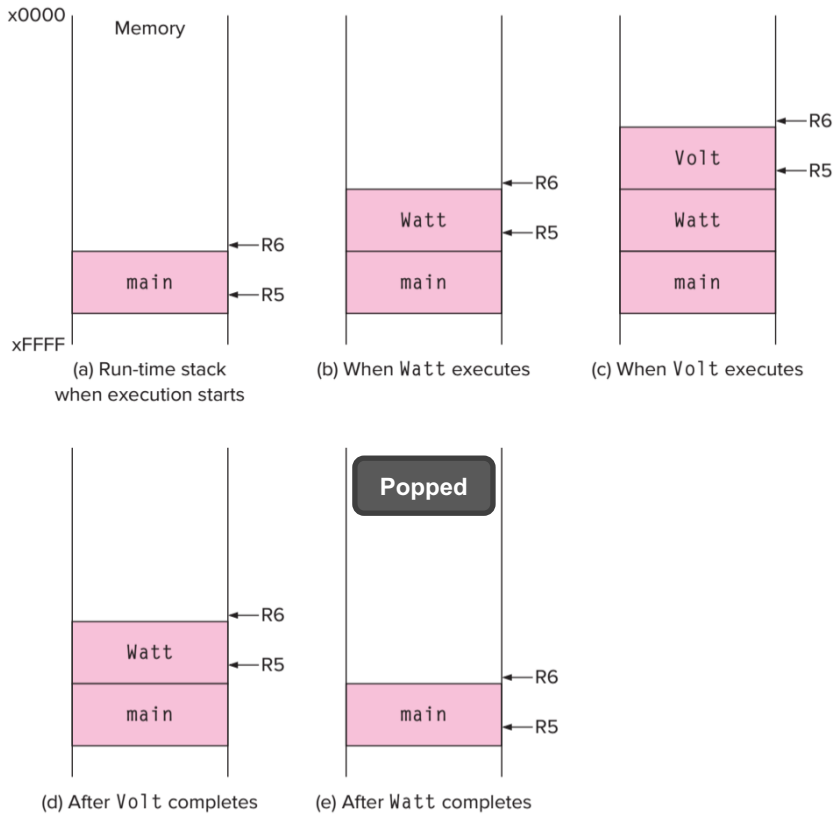
Function Implementation in C – Overview

```
int main(void) {
    int a;
    int b;
    b = Watt(a);    // main calls Watt first
    b = Volt(a, b);    // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);    // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```



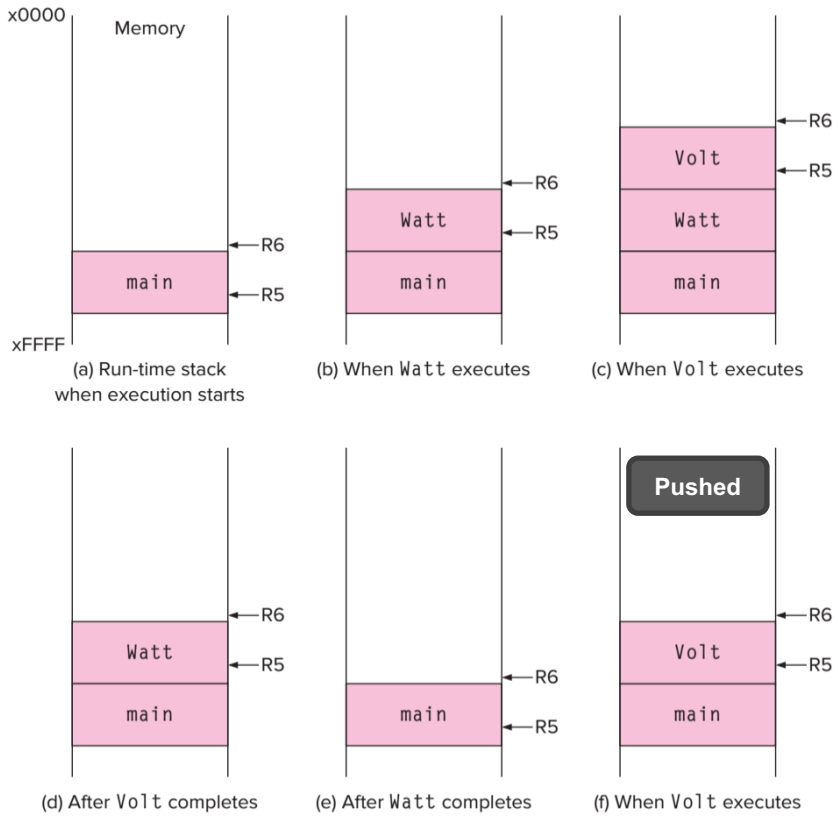
Function Implementation in C – Overview

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```



Function Implementation in C – Overview

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);    // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```



Let's see what happens during a function's lifetime

(1) Calling (Passing arguments to the function)

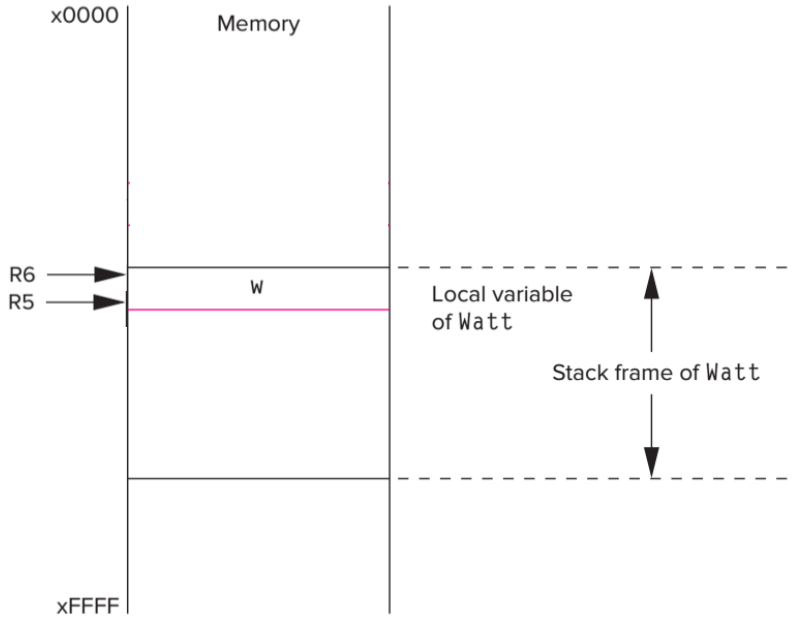
(2) Start Callee (Reserve stack for the function)

(3) End Callee (Return)

(4) Return to Caller

Function Implementation in C – Calling

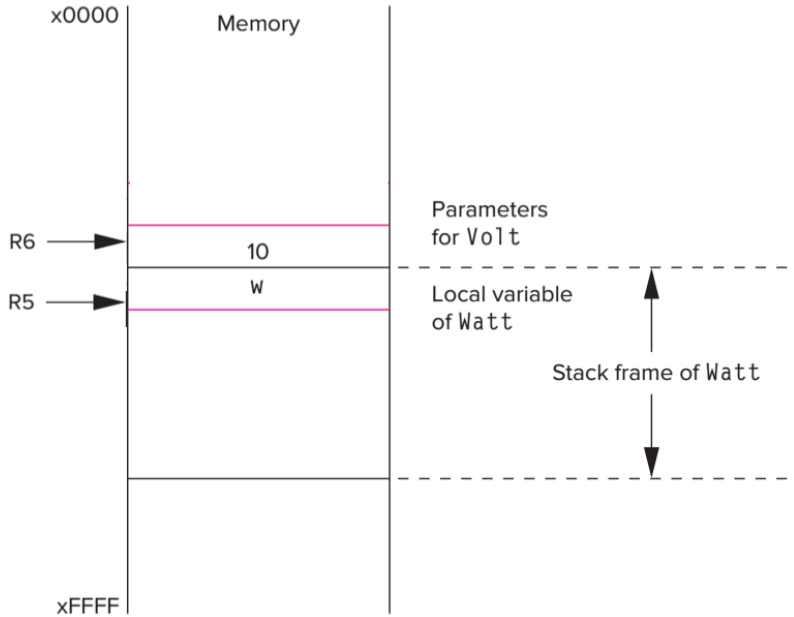
```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);    // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```



Function Implementation in C – Calling

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Decrement R6 and Push 10

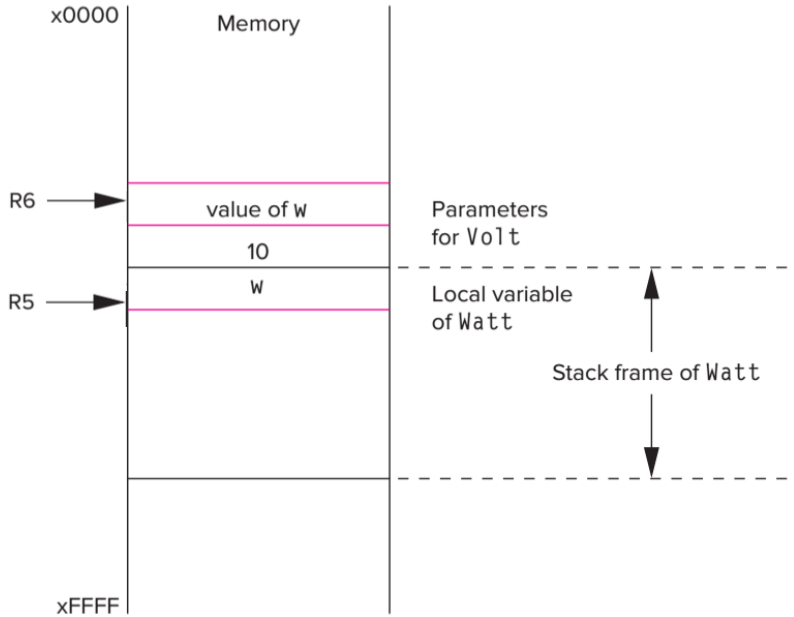


Function Implementation in C – Calling

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Decrement R6 and Push 10

(2) Decrement R6 and Push w value



Function Implementation in C – Calling

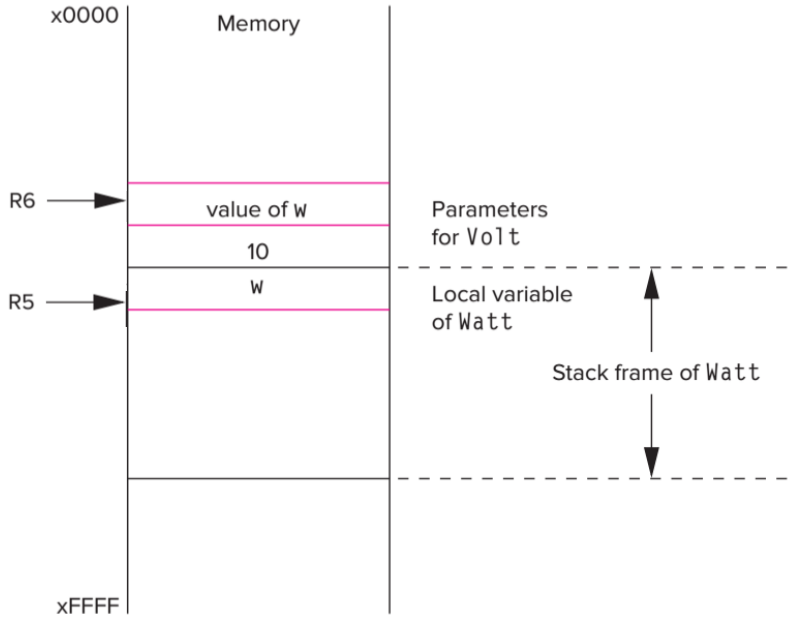
```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}c
```

PC

(1) Decrement R6 and Push 10

(2) Decrement R6 and Push w value

(3) Pass the control to Volt



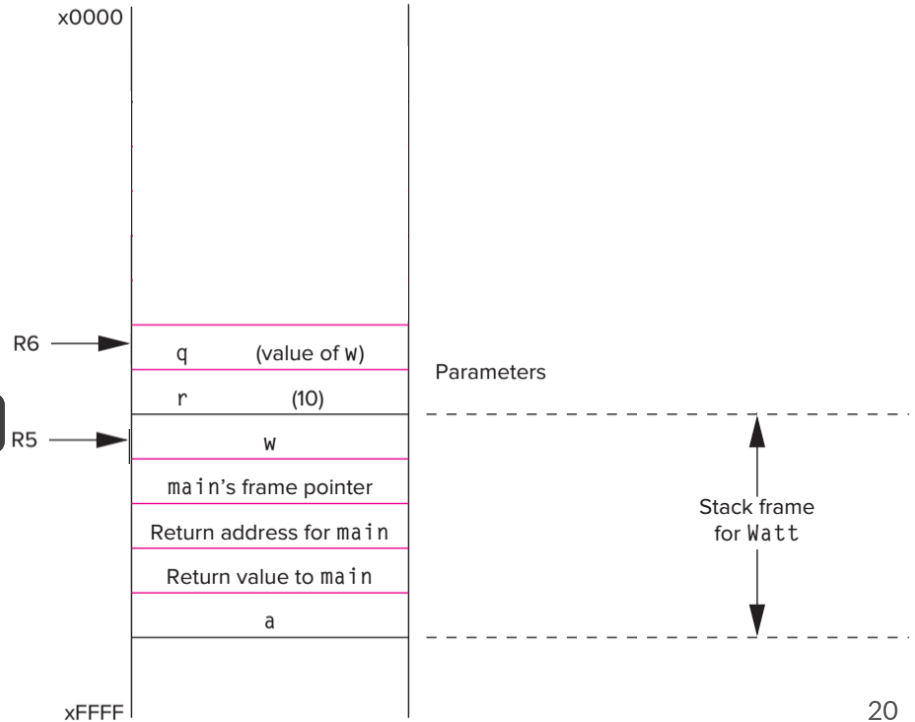
Let's see what happens during a function's lifetime

(1) Calling (Passing arguments to the function)
(2) Start Callee (Reserve stack for the function)
(3) End Callee (Return)
(4) Return to Caller

Function Implementation in C – Start Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);    // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Parameters are already in the stack

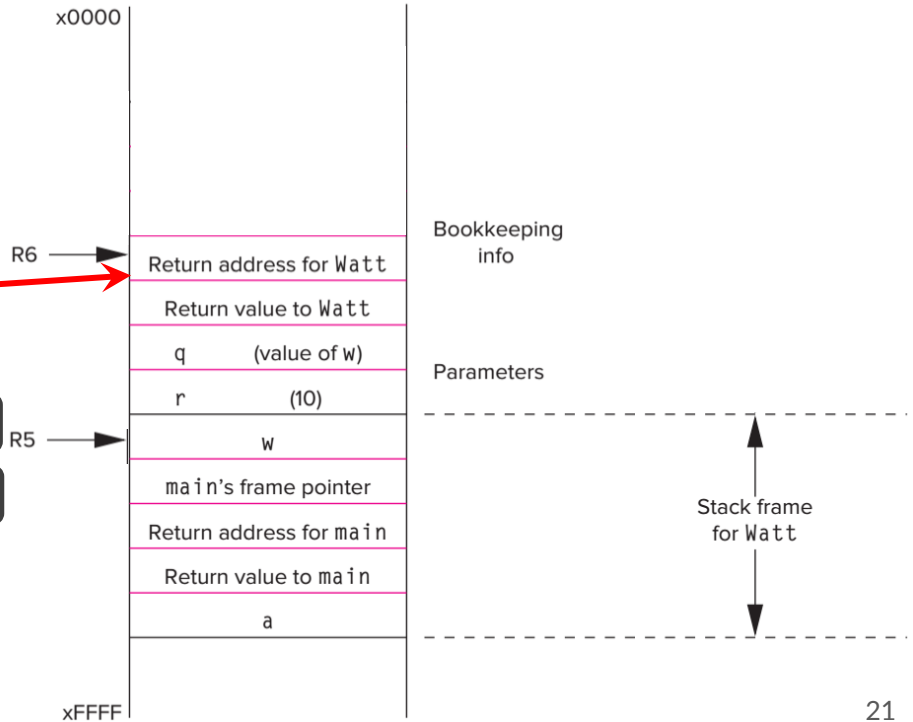


Function Implementation in C – Start Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Parameters are already in the stack

(2) Reserve memory for the return value and store return address



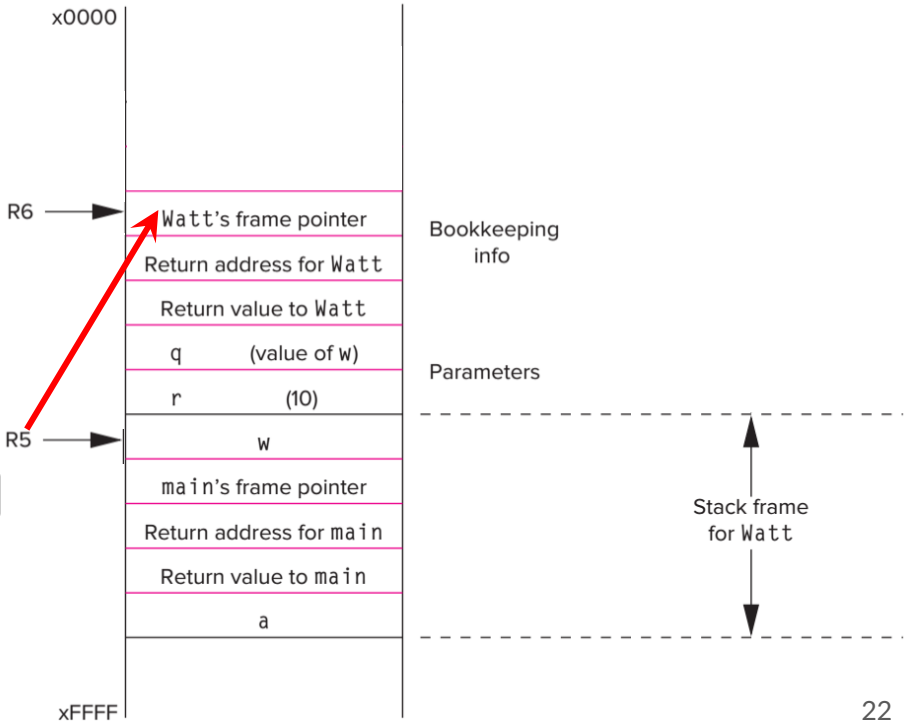
Function Implementation in C – Start Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Parameters are already in the stack

(2) Reserve memory for the return value and store return address

(3) Increase R6 and copy Watt's frame pointer in R5 to where R6 points at



Function Implementation in C – Start Callee

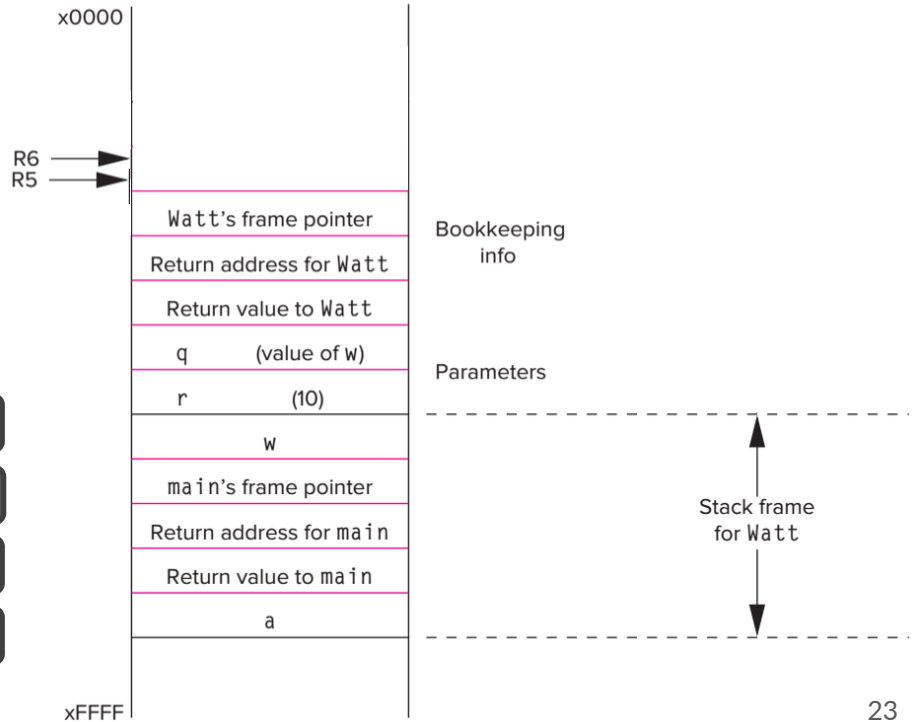
```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Parameters are already in the stack

(2) Reserve memory for the return value and store return address

(3) Increase R6 and copy Watt's frame pointer in R5 to where R6 points at

(4) Increase R6 and move the frame point (R5) to R6



Function Implementation in C – Start Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

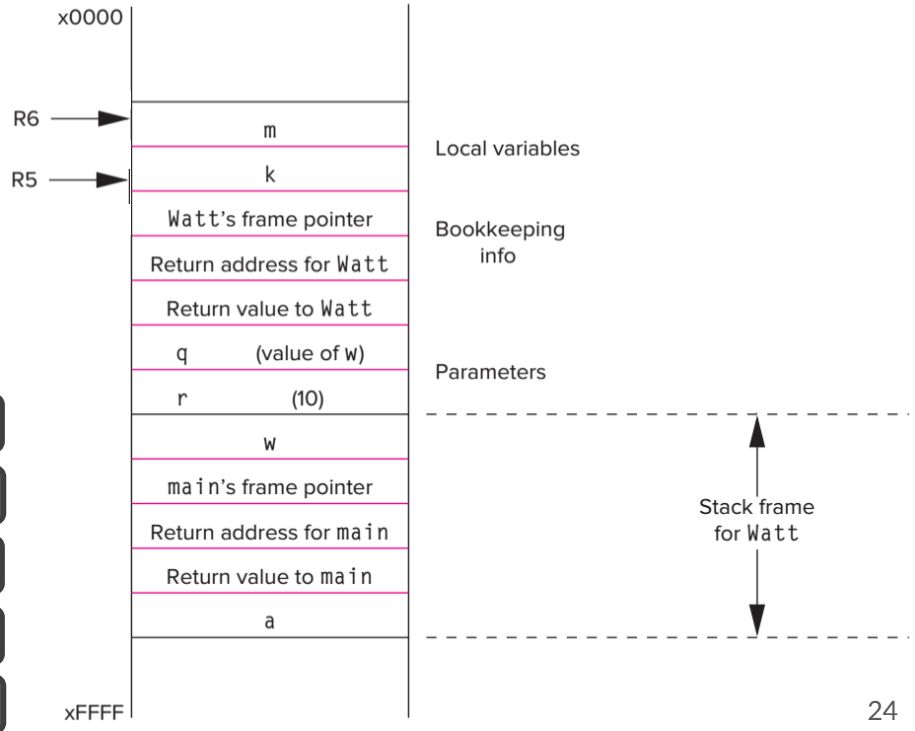
(1) Parameters are already in the stack

(2) Reserve memory for the return value and store return address

(3) Increase R6 and copy Watt's frame pointer in R5 to where R6 points at

(4) Increase R6 and move the frame point (R5) to R6

(5) Reserve memory for Volt's local variables



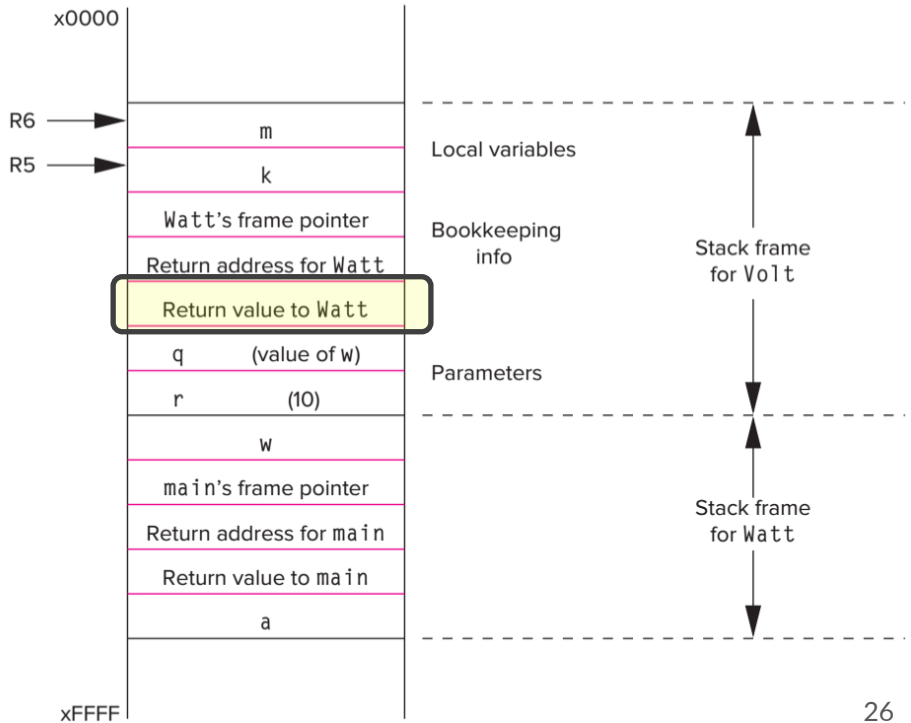
Let's see what happens during a function's lifetime

(1) Calling (Passing arguments to the function)
(2) Start Callee (Reserve stack for the function)
(3) End Callee (Return)
(4) Return to Caller

Function Implementation in C – End Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Store the return value in R5+3

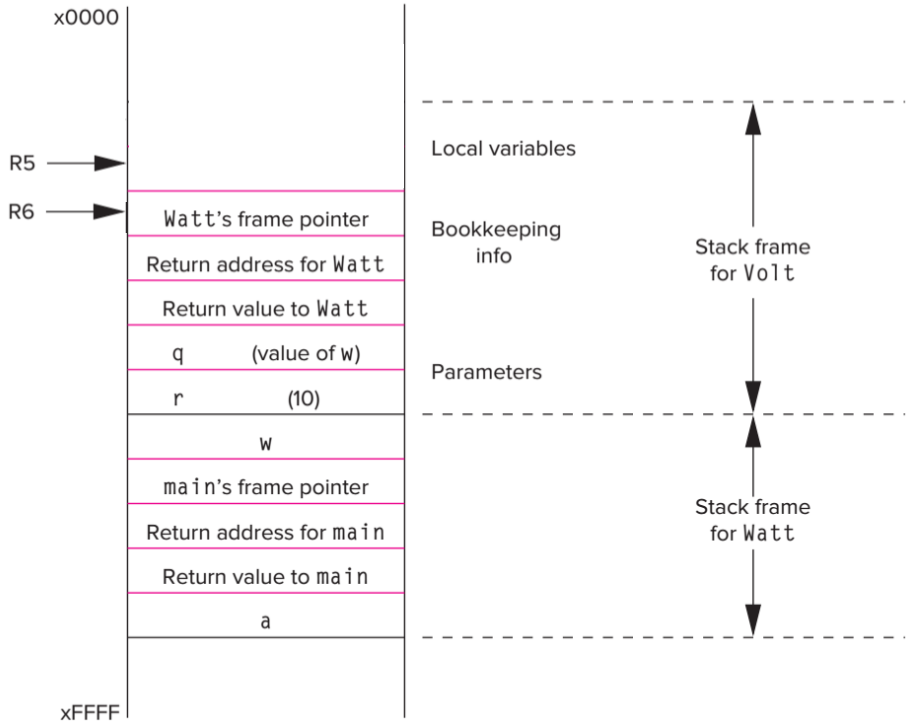


Function Implementation in C – End Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Store the return value in R5+3

(2) Pop the local variables (increase R6 by 2)



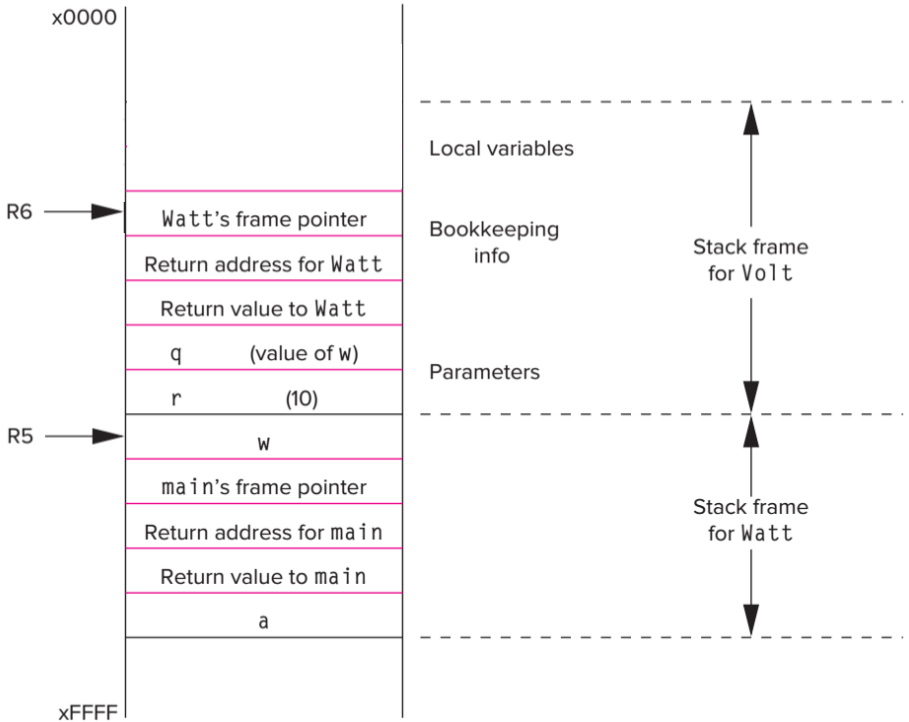
Function Implementation in C – End Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Store the return value in R5+3

(2) Pop the local variables (increase R6 by 2)

(3) Restore Watt's frame pointer (Push R6's value to R5) and pop



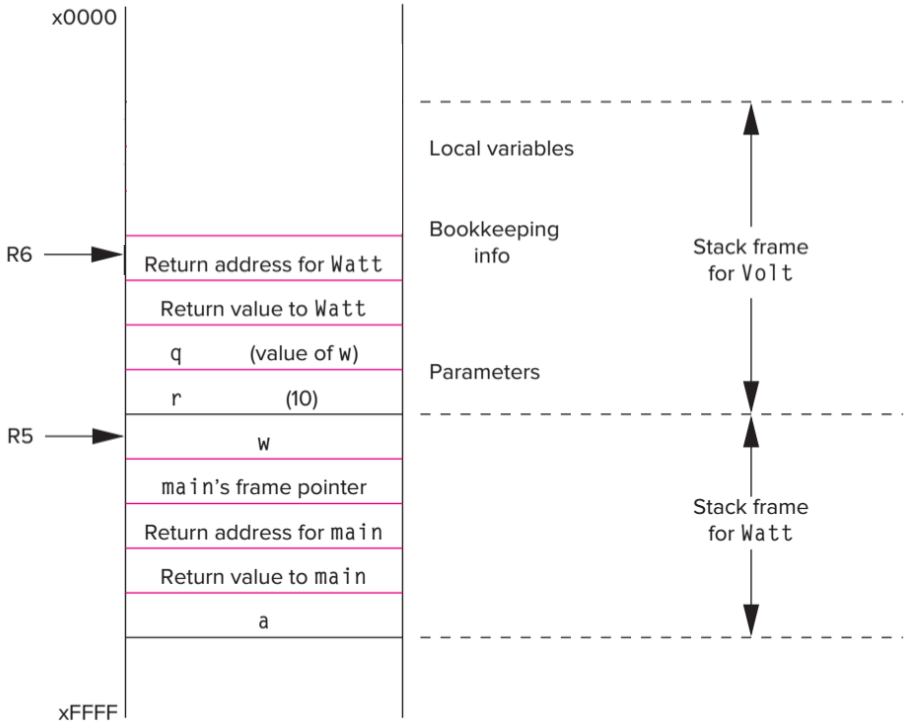
Function Implementation in C – End Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);   // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Store the return value in R5+3

(2) Pop the local variables (increase R6 by 2)

(3) Restore Watt's frame pointer (Push R6's value to R5) and pop



Function Implementation in C – End Callee

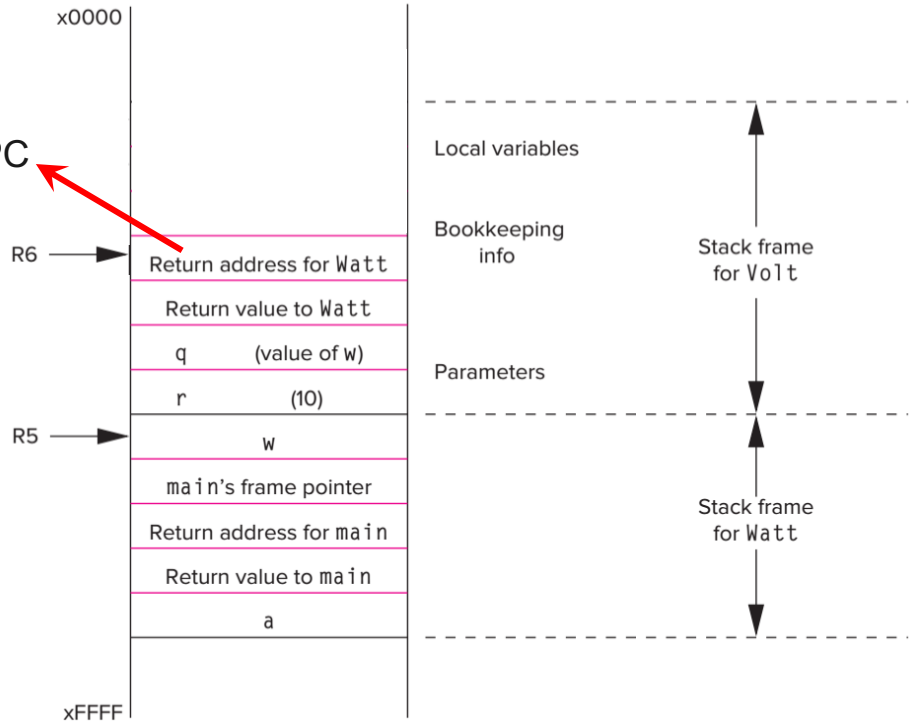
```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Store the return value in R5+3

(2) Pop the local variables (increase R6 by 2)

(3) Restore Watt's frame pointer (Push R6's value to R5)

(4) Transfer the control to Watt (PC = Return value) and pop



Function Implementation in C – End Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

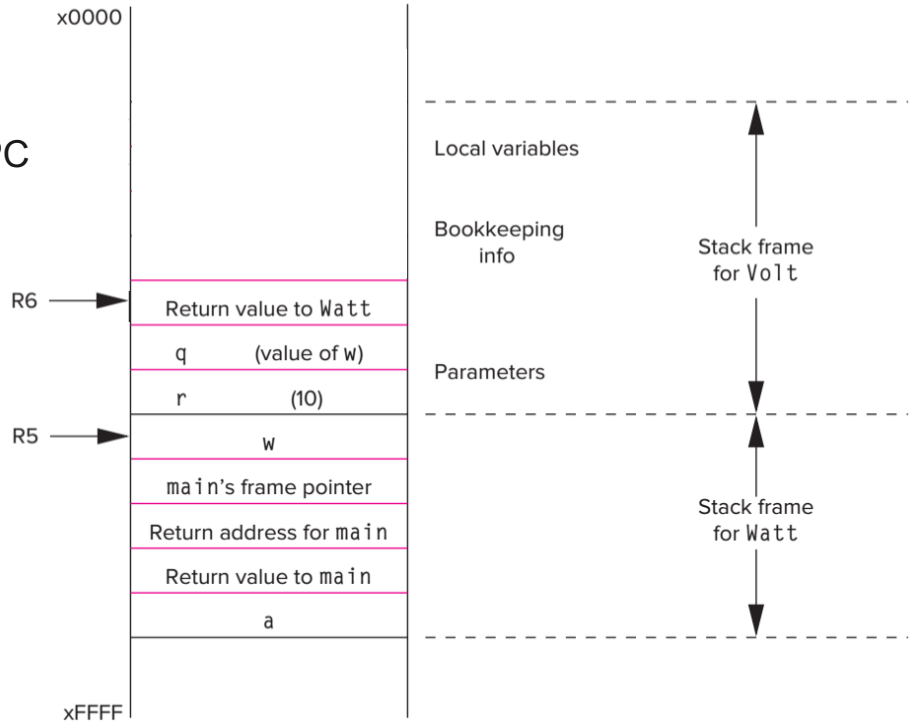
PC

(1) Store the return value in R5+3

(2) Pop the local variables (increase R6 by 2)

(3) Restore Watt's frame pointer (Push R6's value to R5)

(4) Transfer the control to Watt (PC = Return value) and pop



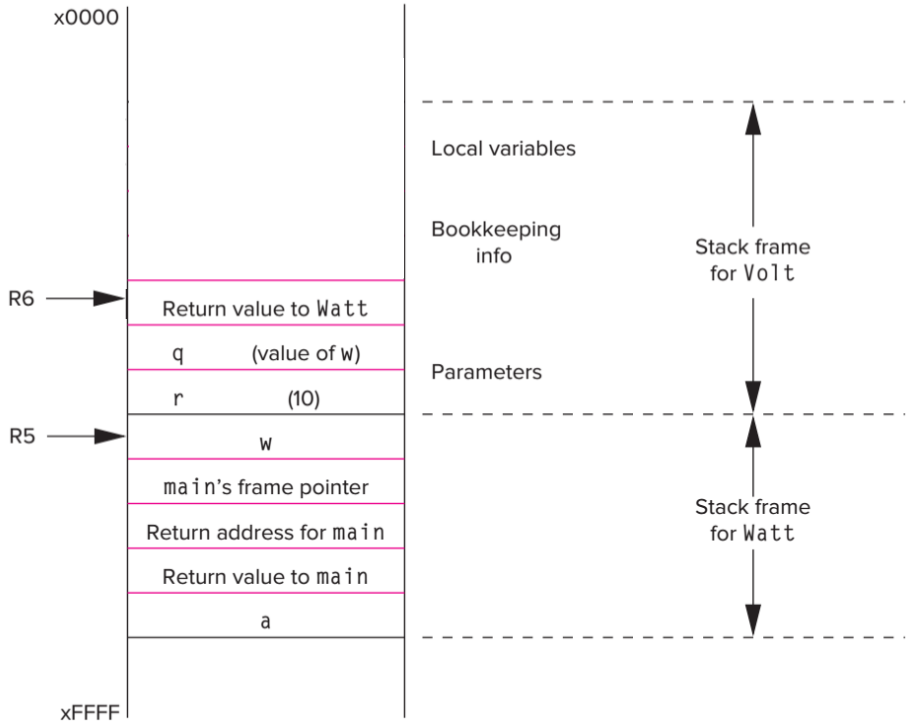
Let's see what happens during a function's lifetime

(1) Calling (Passing arguments to the function)
(2) Start Callee (Reserve stack for the function)
(3) End Callee (Return)
(4) Return to Caller

Function Implementation in C – Return to Caller

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

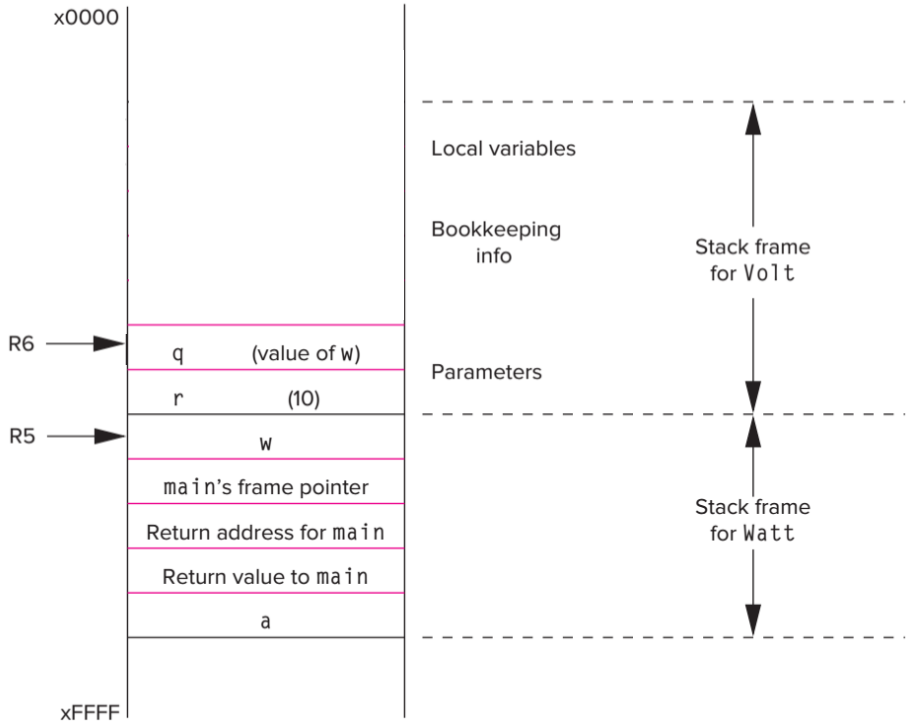
(1) Load the return value (R6) to w and pop



Function Implementation in C – Return to Caller

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Load the return value (R6) to w and pop

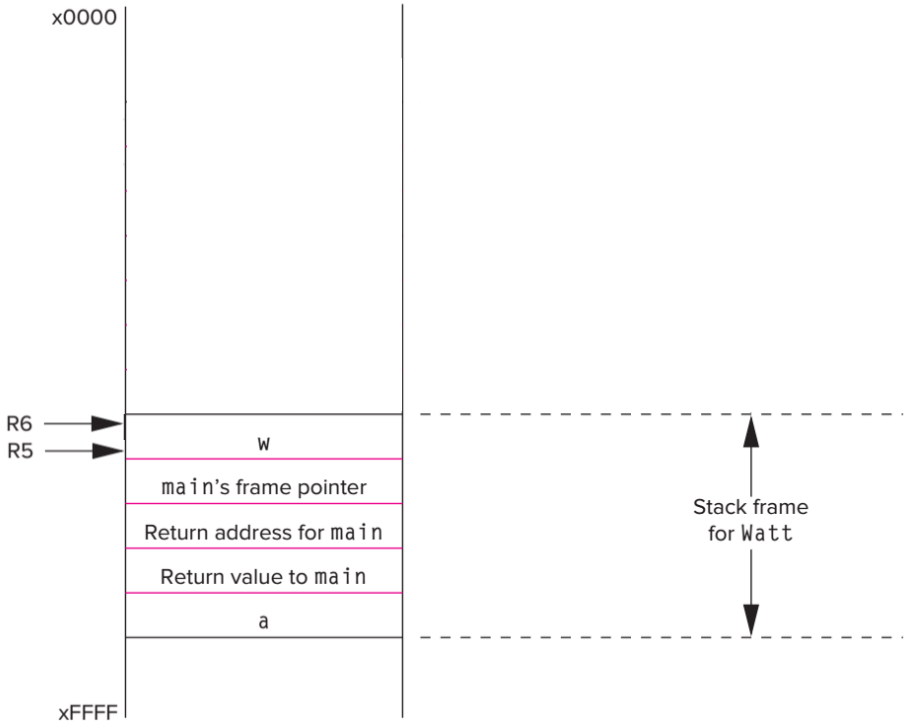


Function Implementation in C – End Callee

```
int main(void) {
    int a;
    int b;
    b = Watt(a);      // main calls Watt first
    b = Volt(a, b);   // then calls Volt
}

int Watt(int a) {
    int w;
    w = Volt(w, 10);  // Watt calls Volt
    return w;
}

int Volt(int q, int r) {
    int k;
    int m;
    return k;
}
```

(1) Load the return value (R6) to w and pop

(2) Pop the arguments



Questions?

Thanks!