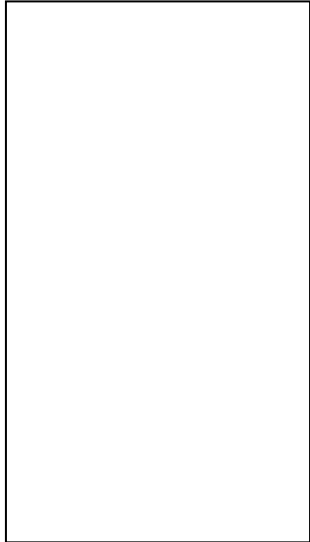


## The TinyJ Virtual Machine's Memory (VM Registers are Not Shown)

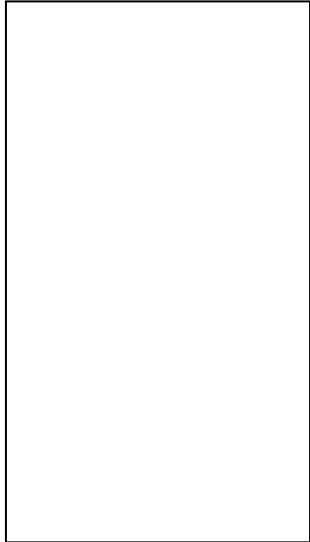
# The TinyJ Virtual Machine's Memory (VM Registers are Not Shown)

STATICALLY  
ALLOCATED  
DATA MEMORY



## The TinyJ Virtual Machine's Memory (VM Registers are Not Shown)

STATICALLY  
ALLOCATED  
DATA MEMORY

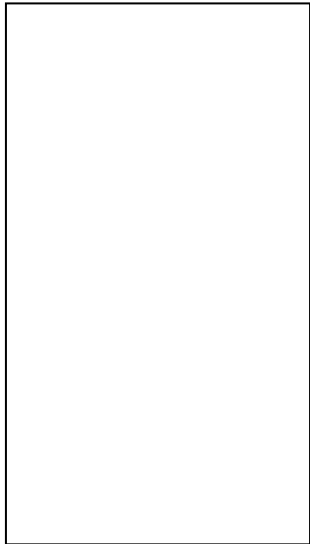


STACK-  
DYNAMICALLY  
ALLOCATED  
DATA MEMORY

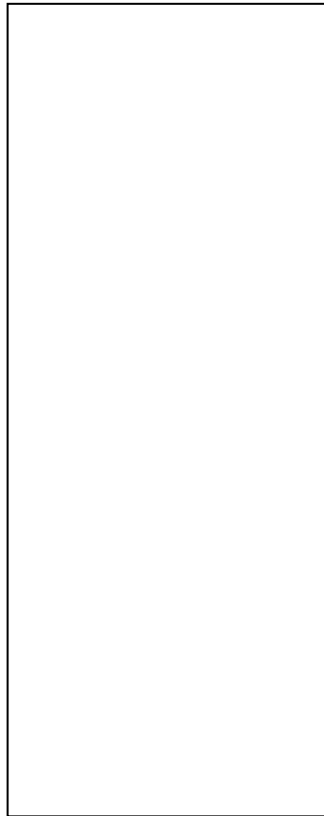


# The TinyJ Virtual Machine's Memory (VM Registers are Not Shown)

**STATICALLY  
ALLOCATED  
DATA MEMORY**



**STACK-  
DYNAMICALLY  
ALLOCATED  
DATA MEMORY**

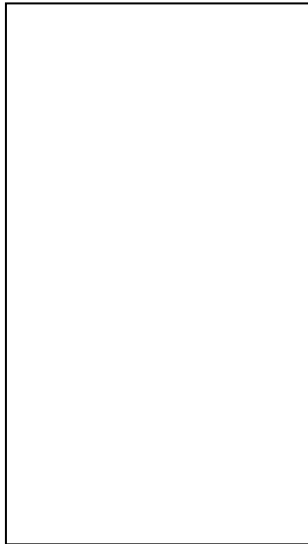


**EXPRSTACK**

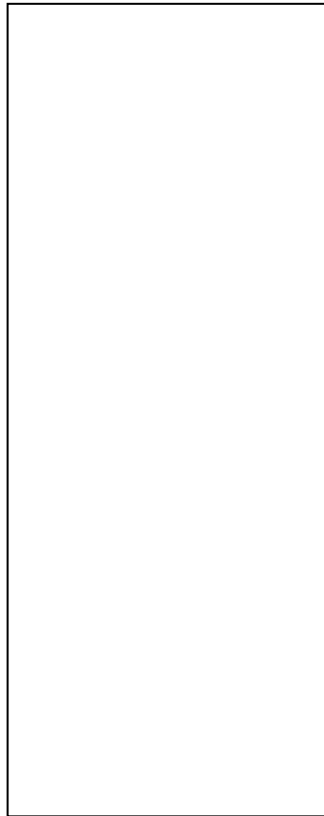


# The TinyJ Virtual Machine's Memory (VM Registers are Not Shown)

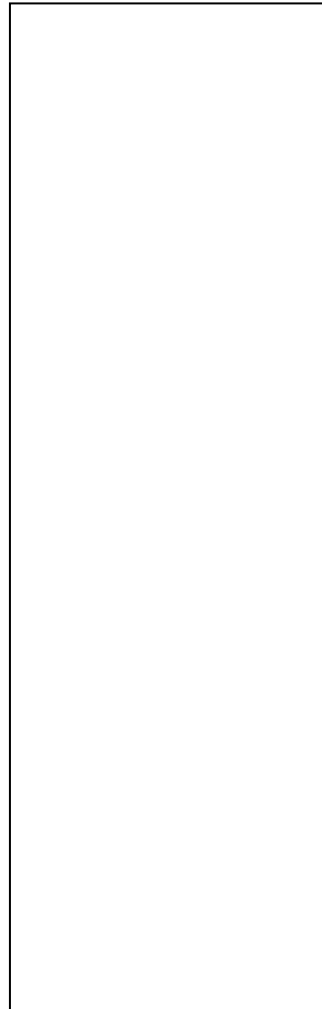
**STATICALLY  
ALLOCATED  
DATA MEMORY**



**STACK-  
DYNAMICALLY  
ALLOCATED  
DATA MEMORY**



**HEAP-  
DYNAMICALLY  
ALLOCATED  
DATA MEMORY**

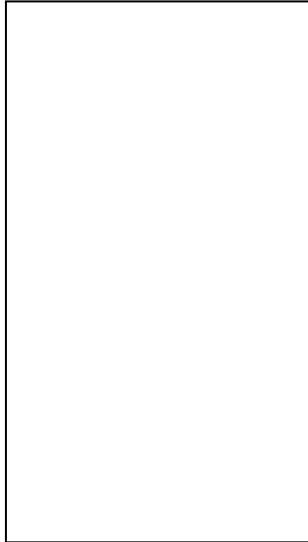


**EXPRSTACK**

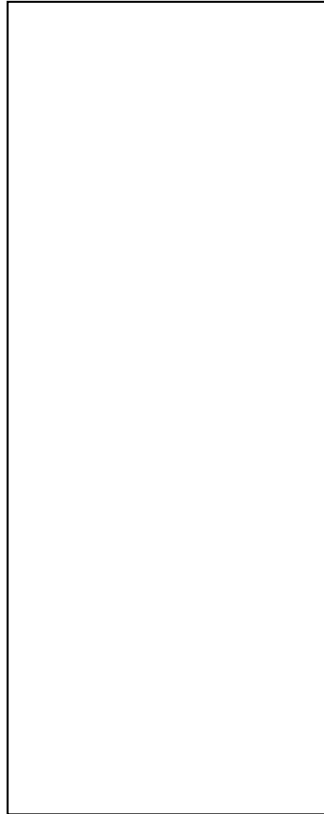


# The TinyJ Virtual Machine's Memory (VM Registers are Not Shown)

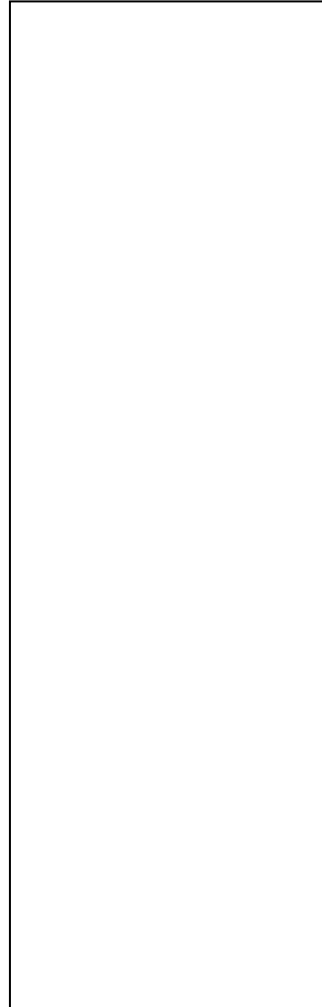
**STATICALLY  
ALLOCATED  
DATA MEMORY**



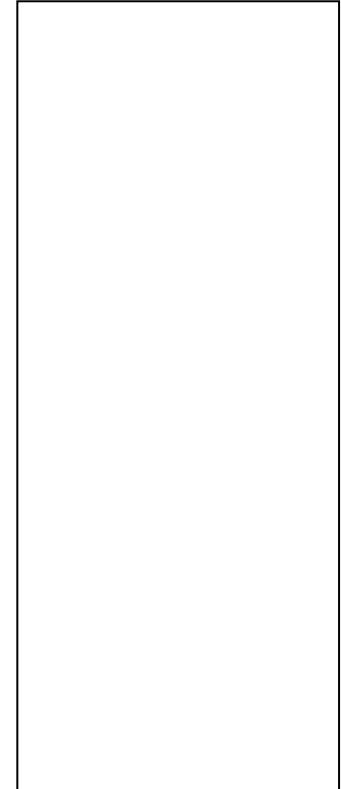
**STACK-  
DYNAMICALLY  
ALLOCATED  
DATA MEMORY**



**HEAP-  
DYNAMICALLY  
ALLOCATED  
DATA MEMORY**



**CODE MEMORY**



**EXPRSTACK**



**An Example:** Which memory locations are statically allocated for the following TinyJ program? What variables / data are they allocated to / for?

**An Example:** Which memory locations are statically allocated for the following TinyJ program? What variables / data are they allocated to / for?

```
import java.util.Scanner;

class Simple {
    static Scanner input
        = new Scanner(System.in);

    static int x, y = 10, z;

    public static void main(String args[])
    {
        System.out.print("Enter x: ");
        x = input.nextInt();
        f(17, y, x-y);
        System.out.println(y + f(21,22,23));
    }

    static int f (int a, int b, int c)
    {
        int v[], w, u = x;
        System.out.print("in f! ");
        return y - a % u;
    }
}
```



**An Example:** Which memory locations are statically allocated for the following TinyJ program? What variables / data are they allocated to / for?

	address	Data Memory	allocated to / for:
<pre>import java.util.Scanner;  class Simple {     static Scanner input         = new Scanner(System.in);     static int x, y = 10, z;     public static void main(String args[])     {         System.out.print("Enter x: ");         x = input.nextInt();         f(17, y, x-y);         System.out.println(y + f(21,22,23));     }     static int f (int a, int b, int c)     {         int v[], w, u = x;         System.out.print("in f! ");         return y - a % u;     } }</pre>			


```
import java.util.Scanner;

class Simple {
    static Scanner input
        = new Scanner(System.in);

    static int x, y = 10, z;

    public static void main(String args[])
    {
        System.out.print("Enter x: ");
        x = input.nextInt();
        f(17, y, x-y);
        System.out.println(y + f(21,22,23));
    }

    static int f (int a, int b, int c)
    {
        int v[], w, u = x;
        System.out.print("in f! ");
        return y - a % u;
    }
}
```

address	Data Memory	allocated to / for:
0		<b>x</b>

```
import java.util.Scanner;

class Simple {
    static Scanner input
        = new Scanner(System.in);

    static int x, y = 10, z;

    public static void main(String args[])
    {
        System.out.print("Enter x: ");
        x = input.nextInt();
        f(17, y, x-y);
        System.out.println(y + f(21,22,23));
    }

    static int f (int a, int b, int c)
    {
        int v[], w, u = x;
        System.out.print("in f! ");
        return y - a % u;
    }
}
```

address	Data Memory	allocated to / for:
0		x
1		y

```
import java.util.Scanner;

class Simple {
    static Scanner input
        = new Scanner(System.in);

    static int x, y = 10, z;

    public static void main(String args[])
    {
        System.out.print("Enter x: ");
        x = input.nextInt();
        f(17, y, x-y);
        System.out.println(y + f(21,22,23));
    }

    static int f (int a, int b, int c)
    {
        int v[], w, u = x;
        System.out.print("in f! ");
        return y - a % u;
    }
}
```

address	Data Memory	allocated to / for:
0		x
1		y
2		z

**An Example:** Which memory locations are statically allocated for the following TinyJ program? What variables / data are they allocated to / for?

```
import java.util.Scanner;

class Simple {
    static Scanner input
        = new Scanner(System.in);
    static int x, y = 10, z;
    public static void main(String args[])
    {
        System.out.print("Enter x: ");
        x = input.nextInt();
        f(17, y, x-y);
        System.out.println(y + f(21,22,23));
    }

    static int f (int a, int b, int c)
    {
        int v[], w, u = x;
        System.out.print("in f! ");
        return y - a % u;
    }
}
```

address	Data Memory	allocated to / for:
0		x
1		y
2		z
3		'E'
4		'n'
5		't'
6		'e'
7		'r'
8		','
9		'x'
10		':'
11		','

**An Example:** Which memory locations are statically allocated for the following TinyJ program? What variables / data are they allocated to / for?

```
import java.util.Scanner;
class Simple {
    static Scanner input
        = new Scanner(System.in);
    static int x, y = 10, z;
    public static void main(String args[])
    {
        System.out.print("Enter x: ");
        x = input.nextInt();
        f(17, y, x-y);
        System.out.println(y + f(21,22,23));
    }
    static int f (int a, int b, int c)
    {
        int v[], w, u = x;
        System.out.print("in f! ");
        return y - a % u;
    }
}
```

address	Data Memory	allocated to / for:
0		x
1		y
2		z
3		'E'
4		'n'
5		't'
6		'e'
7		'r'
8		':'
9		'x'
10		':'
11		':'
12		'i'
13		'n'
14		':'
15		'f'
16		':'
17		':'

**An Example:** Which memory locations are statically allocated for the following TinyJ program? What variables / data are they allocated to / for?

	address	Data Memory	allocated to / for:
<code>import java.util.Scanner;</code>	0		x
<code>class Simple {</code>	1		y
<code>static Scanner input</code>	2		z
<code>    = new Scanner(System.in);</code>	3		'E'
<code>static int x, y = 10, z;</code>	4		'n'
<code>public static void main(String args[])</code>	5		't'
<code>{</code>	6		'e'
<code>    System.out.print("Enter x: ");</code>	7		'r'
<code>    x = input.nextInt();</code>	8		','
<code>    f(17, y, x-y);</code>	9		'x'
<code>    System.out.println(y + f(21,22,23));</code>	10		':'
<code>}</code>	11		','
<code>static int f (int a, int b, int c)</code>	12		'i'
<code>{</code>	13		'n'
<code>    int v[], w, u = x;</code>	14		','
<code>    System.out.print("in f! ");</code>	15		'f'
<code>    return y - a % u;</code>	16		':'
<code>}</code>	17		','

**Answer:** 18 locations are statically allocated as shown above.

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?



**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

**int my\_func(int x, int[] y, int z)     ANSWER:**

```
{
    int a, b[];
    if ( ... ) {
        int c, d[];
        ...
    }
    else {
        int e, f;
        ...
        int g;
        ...
    }
    ...
    int h, i, j, k;
    ...
}
```

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

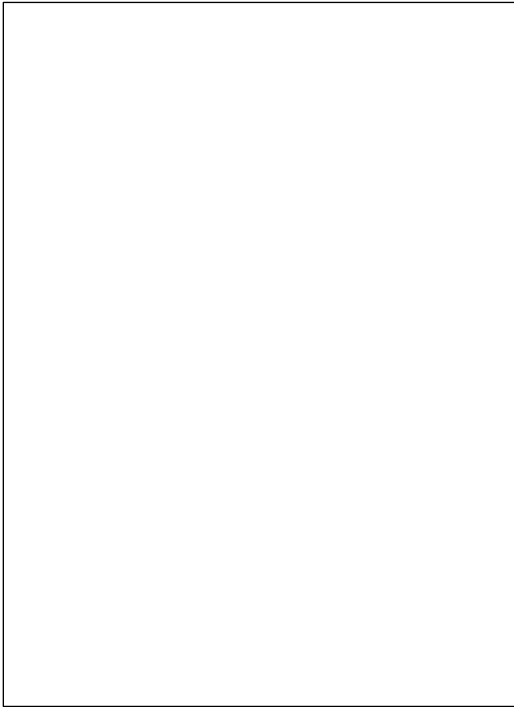
```
{  
    int a, b[];  
    if ( ... ) {  
        int c, d[];  
        ...  
    }  
    else {  
        int e, f;  
        ...  
        int g;  
        ...  
    }  
    ...  
    int h, i, j, k;  
    ...  
}
```

**See pp. 3-4 of:**

<https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

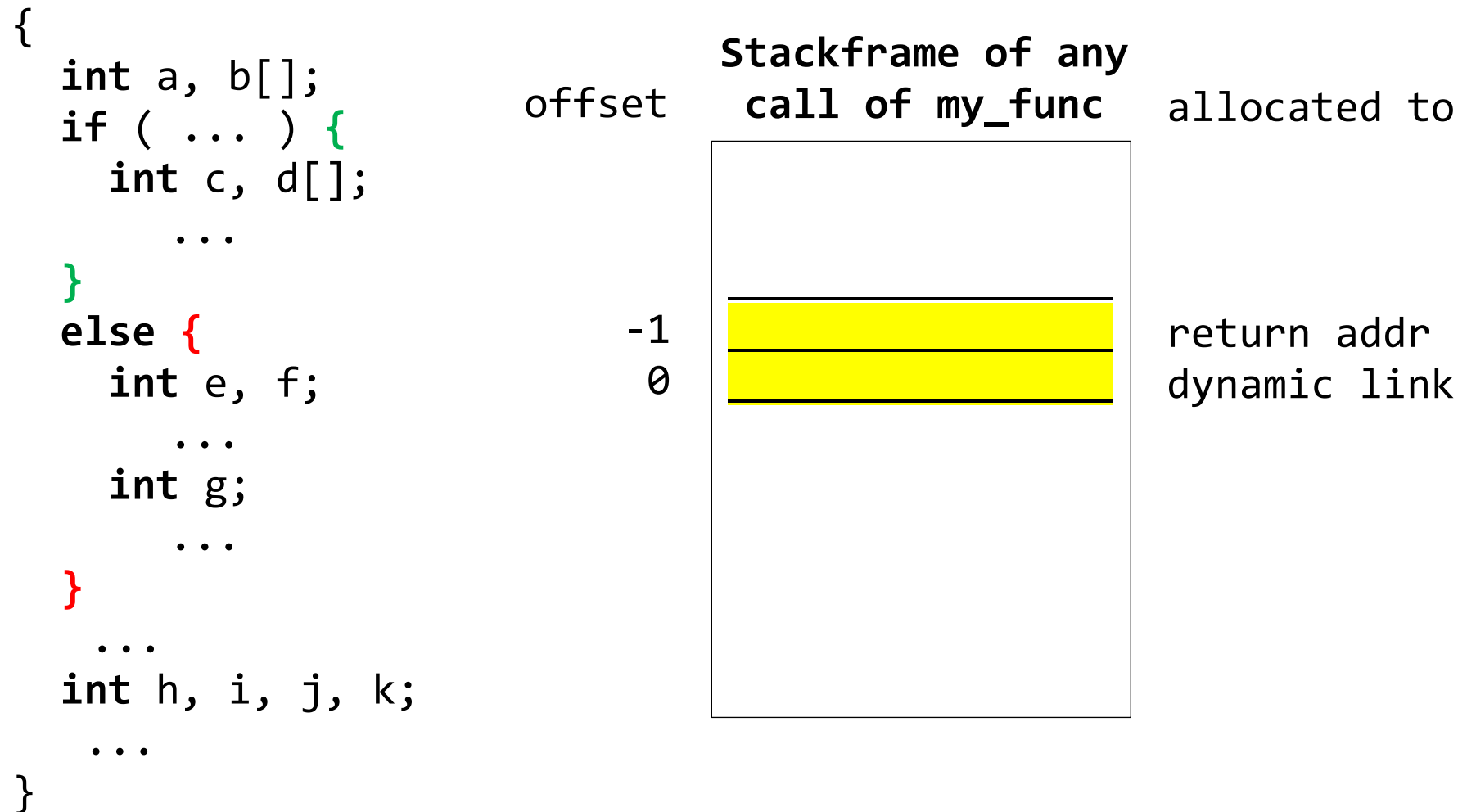
**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

**int my\_func(int x, int[] y, int z)     ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<pre>{   int a, b[];   if ( ... ) {     int c, d[];     ...   }   else {     int e, f;     ...     int g;     ...   }   ...   int h, i, j, k;   ... }</pre>			

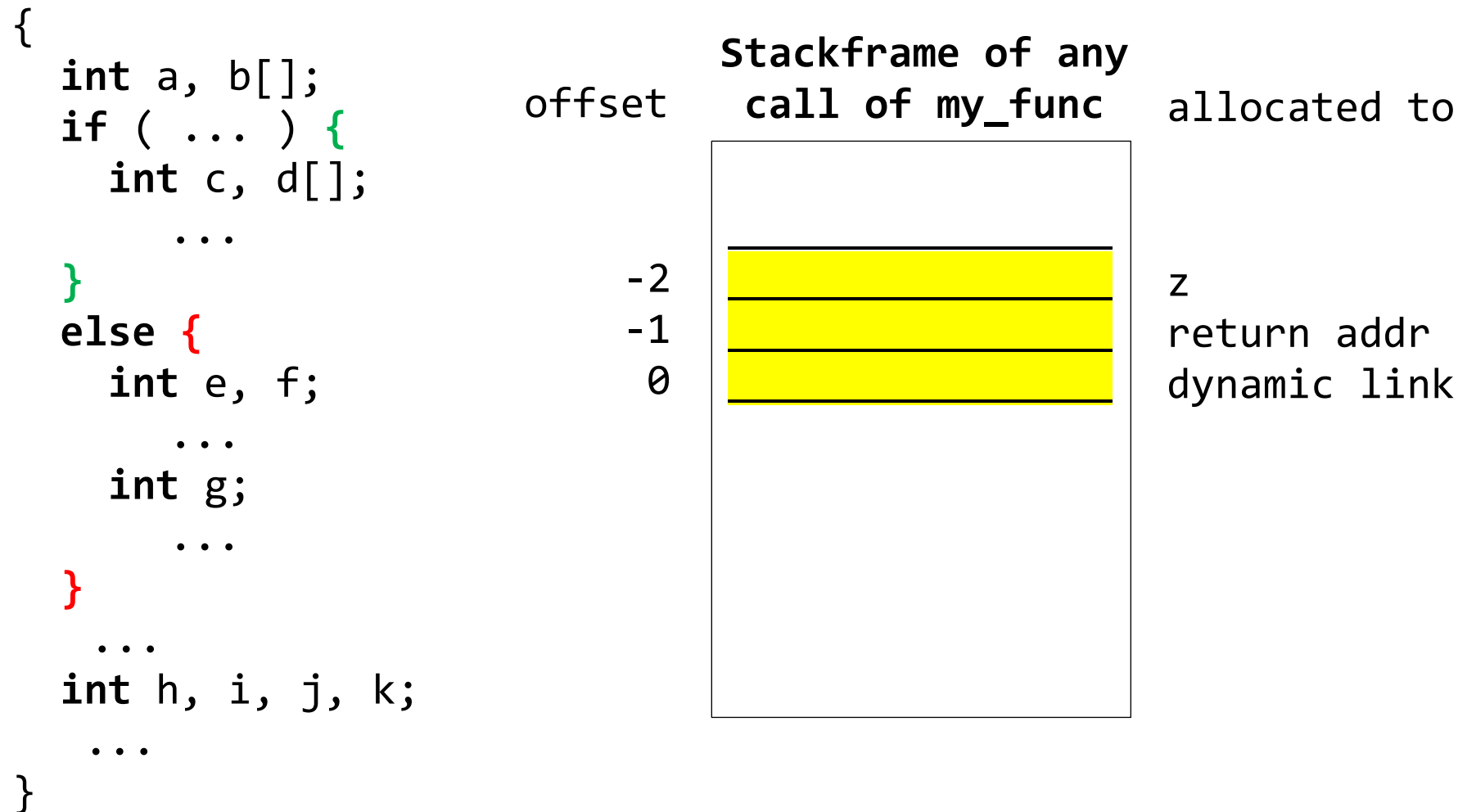
**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**



**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**



**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>			
<code>if ( ... ) {</code>			
<code>int c, d[];</code>	-3		y
<code>...</code>	-2		z
<code>}</code>	-1		return addr
<code>else {</code>	0		dynamic link
<code>int e, f;</code>			
<code>...</code>			
<code>int g;</code>			
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>			
<code>int e, f;</code>			
<code>...</code>			
<code>int g;</code>			
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>			
<code>...</code>			
<code>int g;</code>			
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			



**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>			
<code>int g;</code>			
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c
<code>int g;</code>			
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c
<code>int g;</code>	+4		d
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c, e
<code>int g;</code>	+4		d
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c, e
<code>int g;</code>	+4		d, f
<code>...</code>			
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c, e
<code>int g;</code>	+4		d, f
<code>...</code>	+5		g
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c, e, h
<code>int g;</code>	+4		d, f
<code>...</code>	+5		g
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c, e, h
<code>int g;</code>	+4		d, f, i
<code>...</code>	+5		g
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			



**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c, e, h
<code>int g;</code>	+4		d, f, i
<code>...</code>	+5		g, j
<code>}</code>			
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

**An Example:** How many locations are allocated for each stackframe of the following function, and what stackframe offset is given to each formal parameter and each local variable declared in the body?

`int my_func(int x, int[] y, int z)`      **ANSWER: 11**

	offset	Stackframe of any call of my_func	allocated to
<code>{</code>			
<code>int a, b[];</code>	-4		x
<code>if ( ... ) {</code>	-3		y
<code>int c, d[];</code>	-2		z
<code>...</code>	-1		return addr
<code>}</code>	0		dynamic link
<code>else {</code>	+1		a
<code>int e, f;</code>	+2		b
<code>...</code>	+3		c, e, h
<code>int g;</code>	+4		d, f, i
<code>...</code>	+5		g, j
<code>}</code>	+6		k
<code>...</code>			
<code>int h, i, j, k;</code>			
<code>...</code>			
<code>}</code>			

## Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.gc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

How are stackframes allocated and deallocated during execution of the following sequence of calls and returns?

## Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

How are stackframes allocated and deallocated during execution of the following sequence of calls and returns?

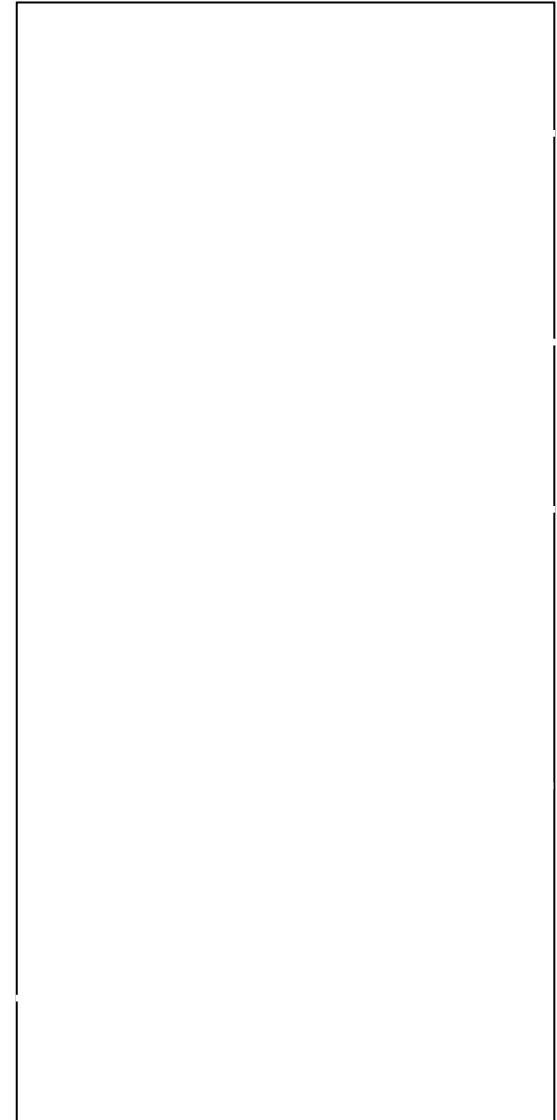
- (1) main() is called
- (2) main() calls f()
- (3) f() calls g()
- (4) g() calls h()
- (5) h() calls f()
- (6) f() returns control to h()
- (7) h() returns control to g()
- (8) g() calls f()

# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

```
(1) main() is called
(2)  main() calls f()
(3)    f() calls g()
(4)      g() calls h()
(5)        h() calls f()
(6)          f() returns control to h()
(7)            h() returns control to g()
(8)              g() calls f()
```



# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

`main()`'s stackframe

---

# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

`main()`'s stackframe

---

`f()`'s stackframe

---

# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

`main()`'s stackframe

---

`f()`'s stackframe

---

`g()`'s stackframe

---

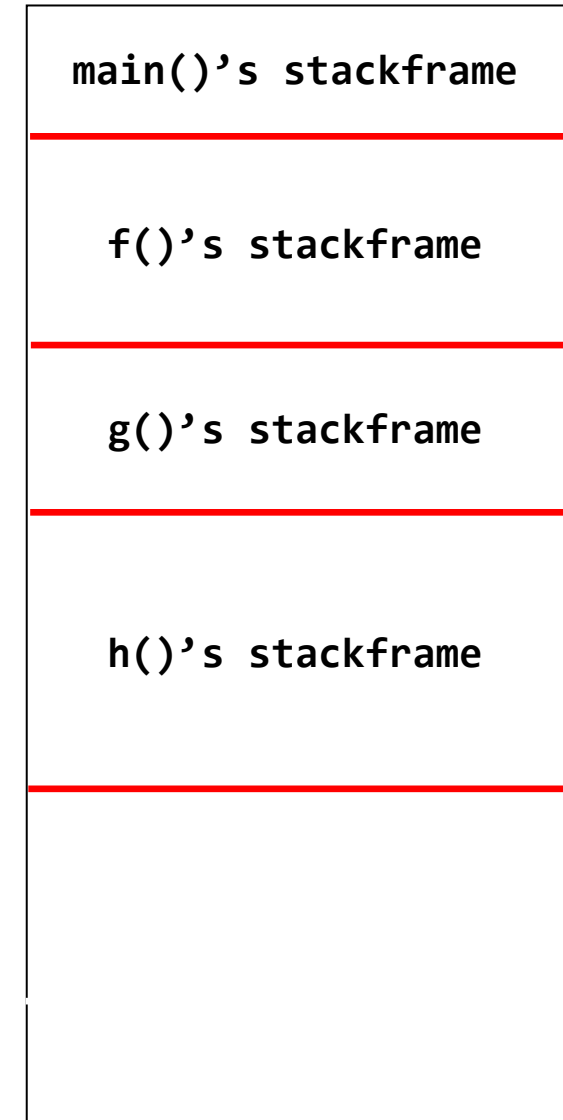


# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

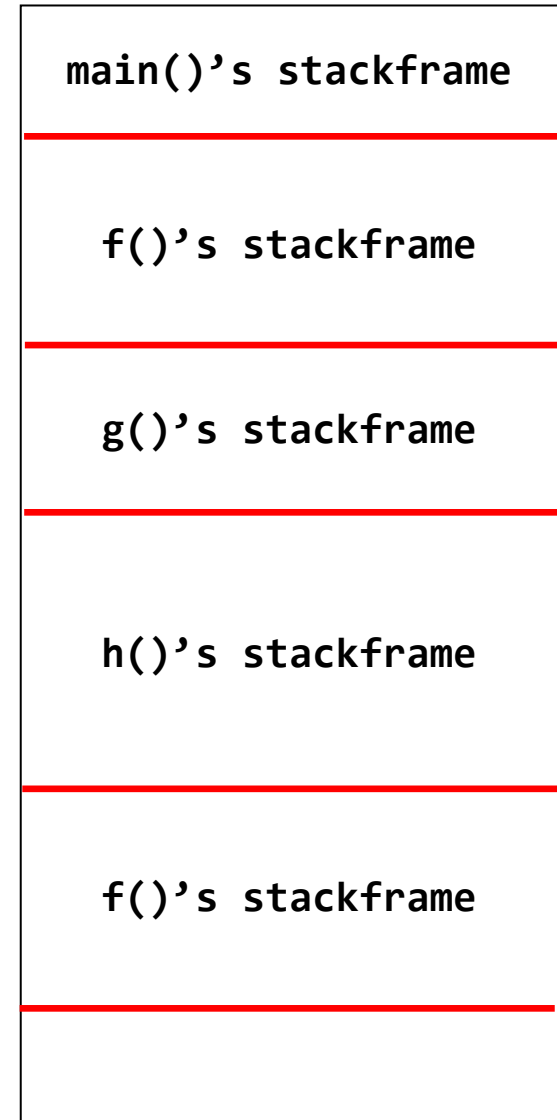


# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

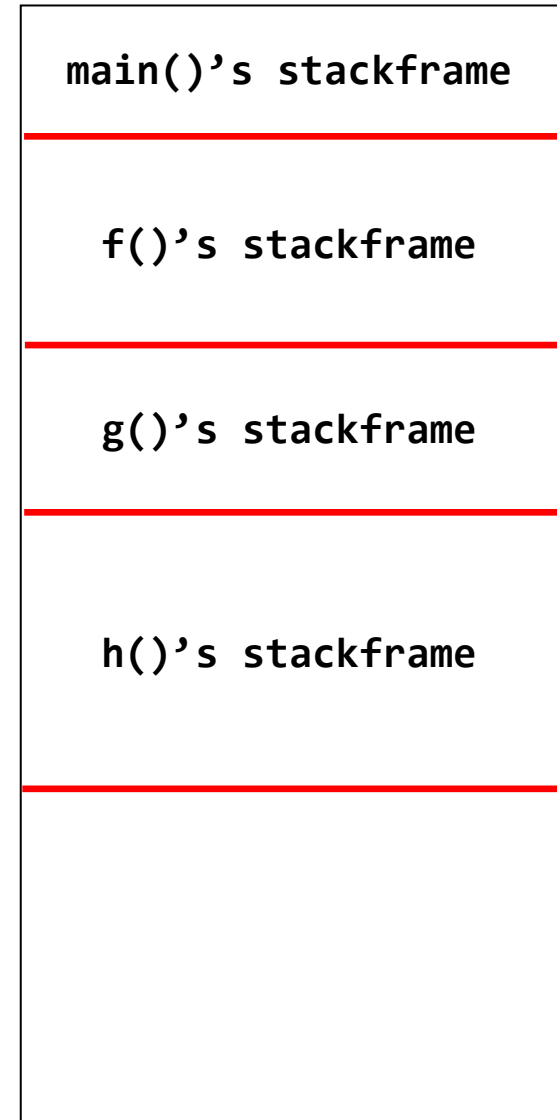


# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

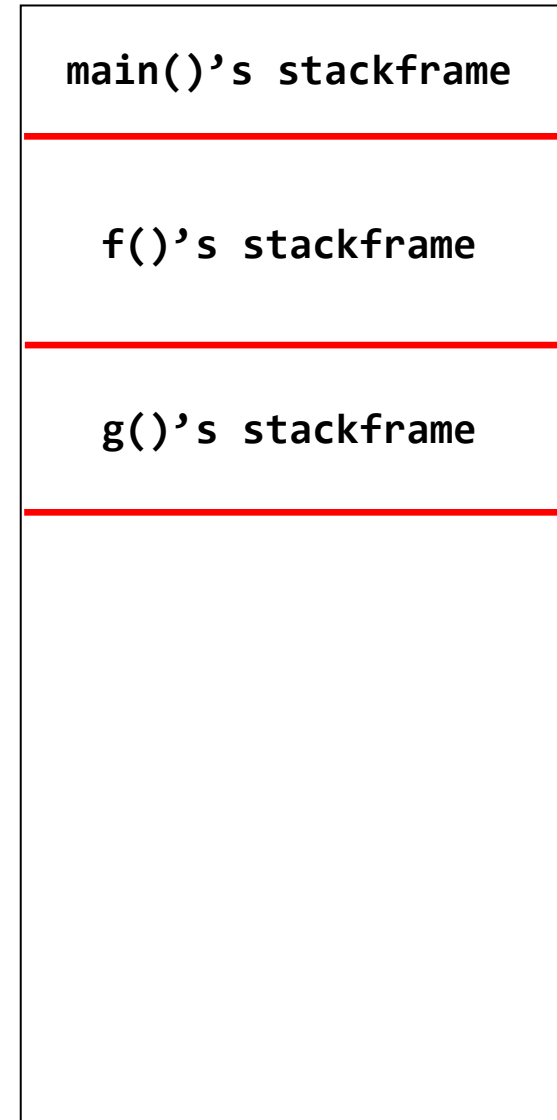


# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

## Stack-Dynamically Allocated Data Memory

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

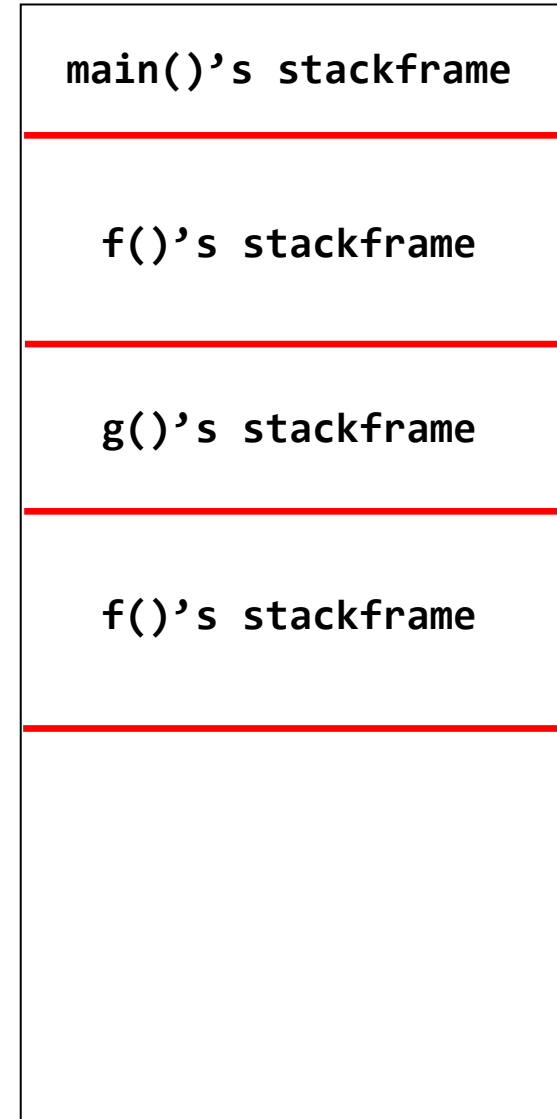


# Example of Stack-Dynamic Allocation

See p. 4 of: <https://euclid.cs.qc.cuny.edu/316/Memory-allocation-VM-instruction-set-and-hints-for-asn-2.pdf>

- (1) `main()` is called
- (2) `main()` calls `f()`
- (3) `f()` calls `g()`
- (4) `g()` calls `h()`
- (5) `h()` calls `f()`
- (6) `f()` returns control to `h()`
- (7) `h()` returns control to `g()`
- (8) `g()` calls `f()`

## Stack-Dynamically Allocated Data Memory



# **EXECUTION OF VARIOUS TINYJ VM INSTRUCTIONS**

**BEFORE** execution of: **PUSHNUM 23**

s t o f f r a m e t	CURRENTLY EXECUTING METHOD ACTIVATION'S STACKFRAME (Part of Data memory)

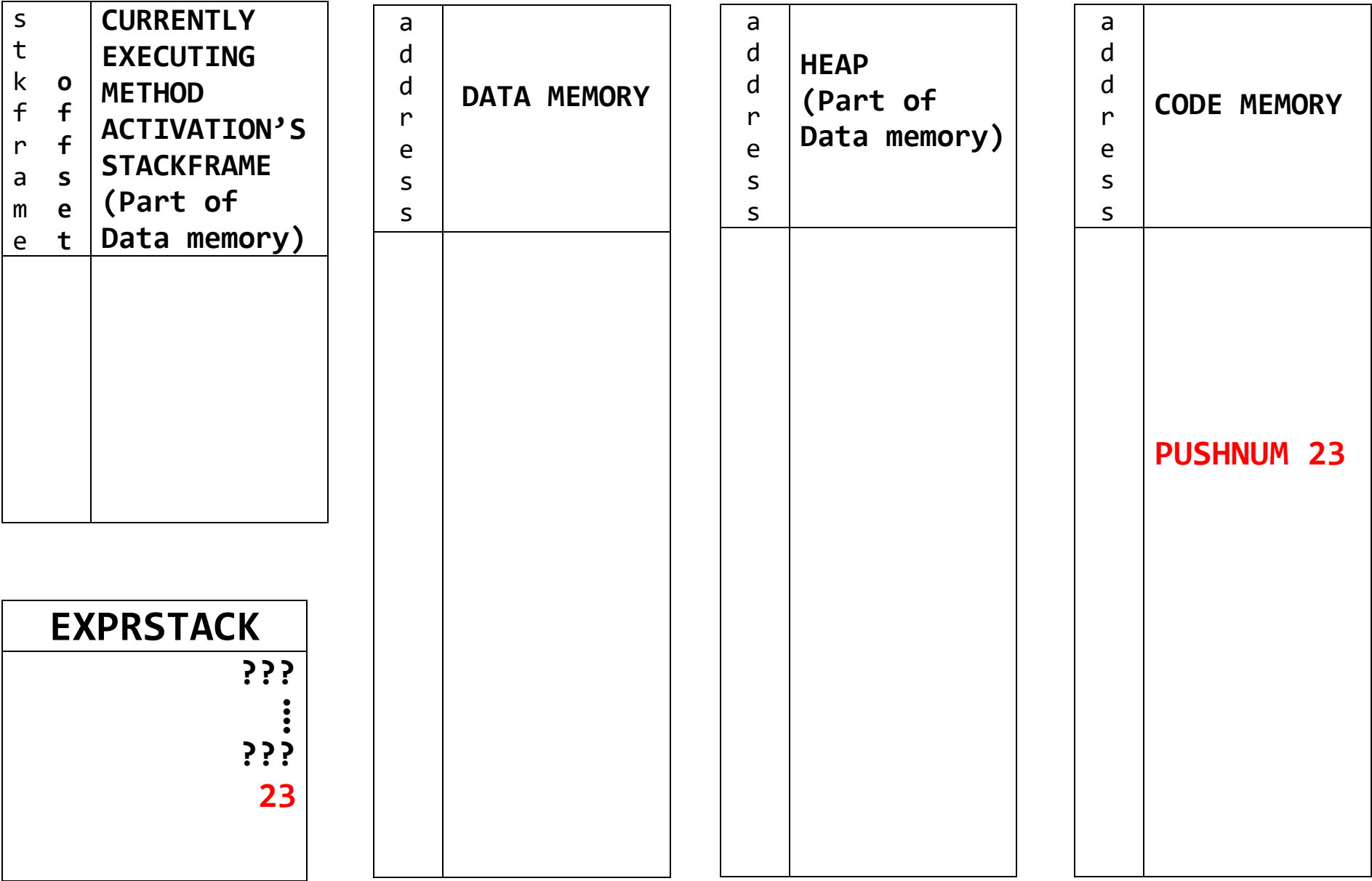
EXPRSTACK
???
⋮
???

a d d r e s s	DATA MEMORY

a d d r e s s	HEAP (Part of Data memory)

a d d r e s s	CODE MEMORY
	<b>PUSHNUM 23</b>

**AFTER** *execution of:* **PUSHNUM 23**





**BEFORE** execution of: **PUSHSTATADDR 17**

s t o c k f r a m e s	CURRENTLY EXECUTING METHOD ACTIVATION'S STACKFRAME (Part of Data memory)

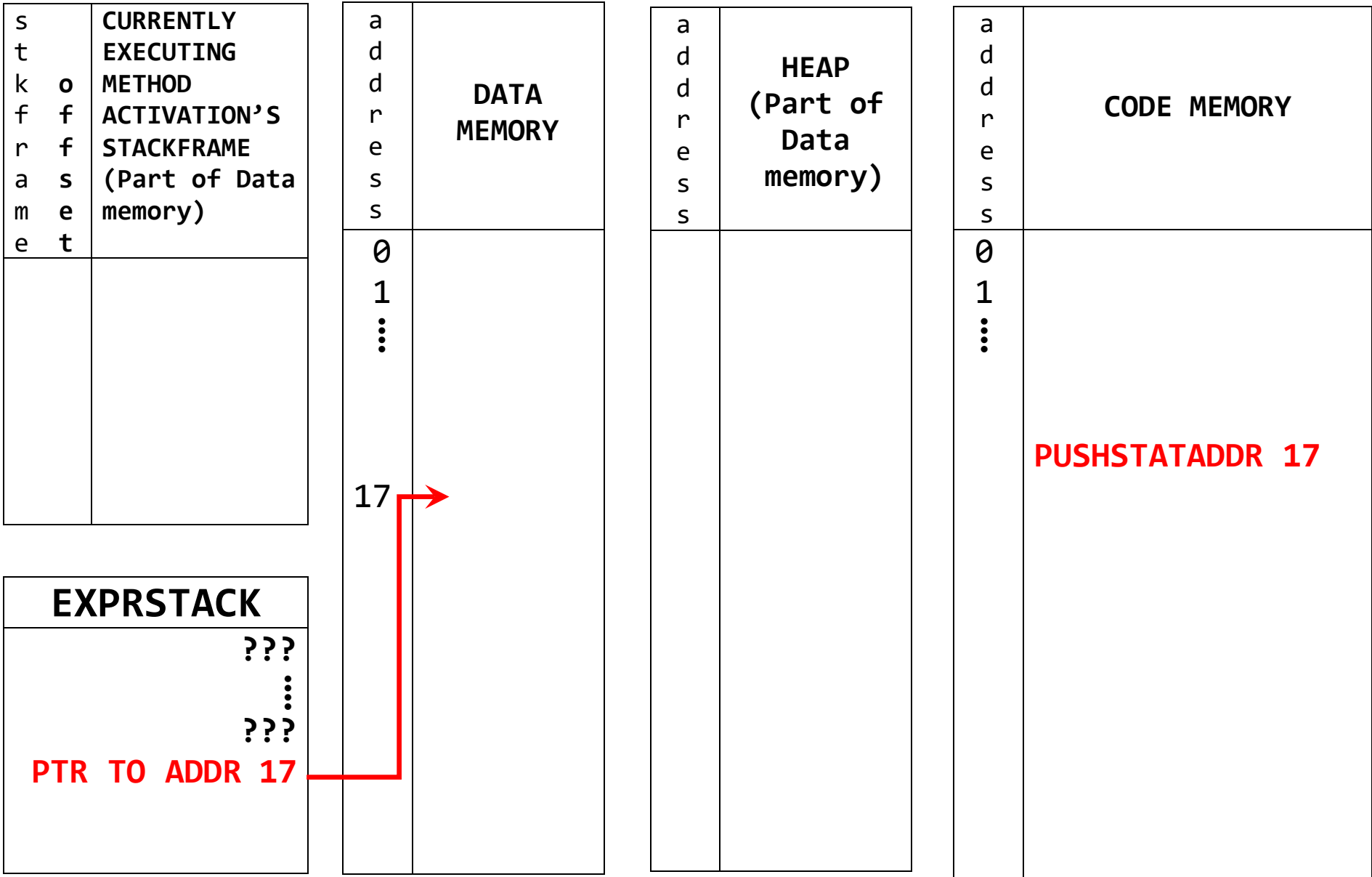
EXPRSTACK
???
⋮
???

a d d r e s s	DATA MEMORY
0 1 ⋮	
17	

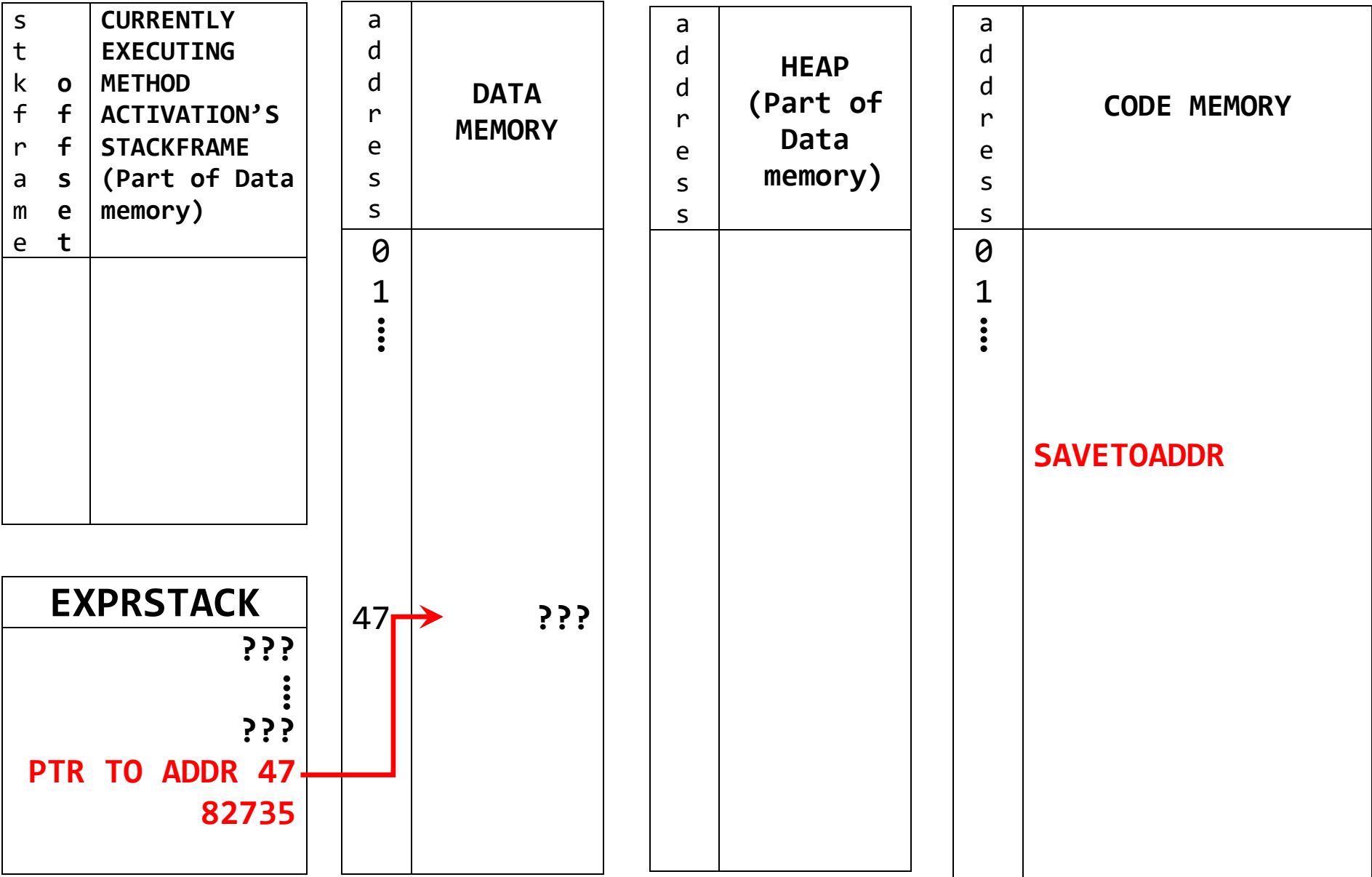
a d d r e s s	HEAP (Part of Data memory)

a d d r e s s	CODE MEMORY
0 1 ⋮	
	<b>PUSHSTATADDR 17</b>

**AFTER** execution of: **PUSHSTATADDR 17**



***BEFORE*** execution of SAVETOADDR



**AFTER** execution of **SAVETOADDR**

s t o c k f r a m e s	CURRENTLY EXECUTING METHOD ACTIVATION'S STACKFRAME (Part of Data memory)

EXPRSTACK
???
⋮
???

a d d r e s s	DATA MEMORY
0 1 ⋮	
47	82735

a d d r e s s	HEAP (Part of Data memory)

a d d r e s s	CODE MEMORY
0 1 ⋮	
	SAVETOADDR