The stack

The stack

The stack is used during:
Function calls (maybe)
Interrupts
And for storing local variables

The stack

```
int X = 1;
void main()
{
	MyFunction();
	X++;
}
void MyFunction()
{
// do some stuff
	return;
}
```

The stack

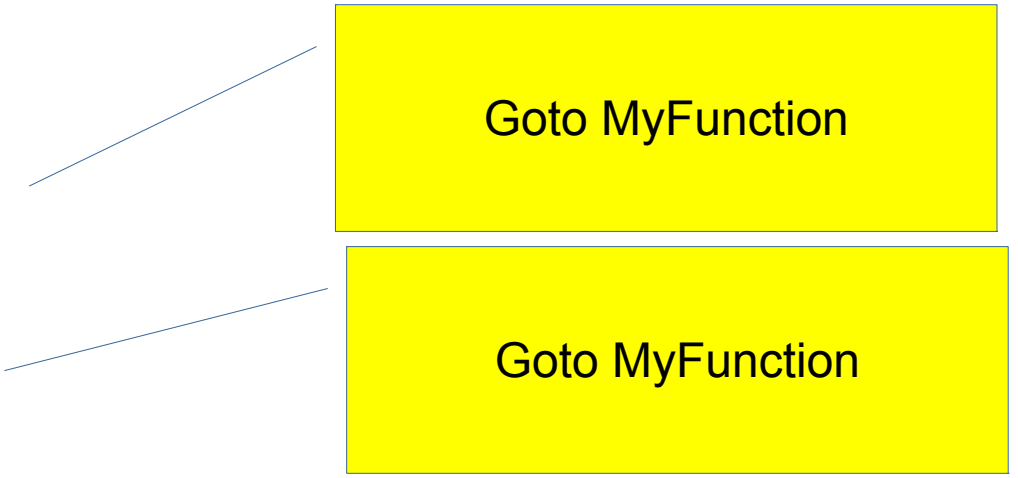
```
int X = 1;
void main()
{
	MyFunction();
	X++;
}
void MyFunction()
{
// do some stuff
	return;
}
```

The stack

```
int X = 1;
void main()
{
	MyFunction();
Next:
	X++;
}
void MyFunction()
{
// do some stuff
	return;
}
```

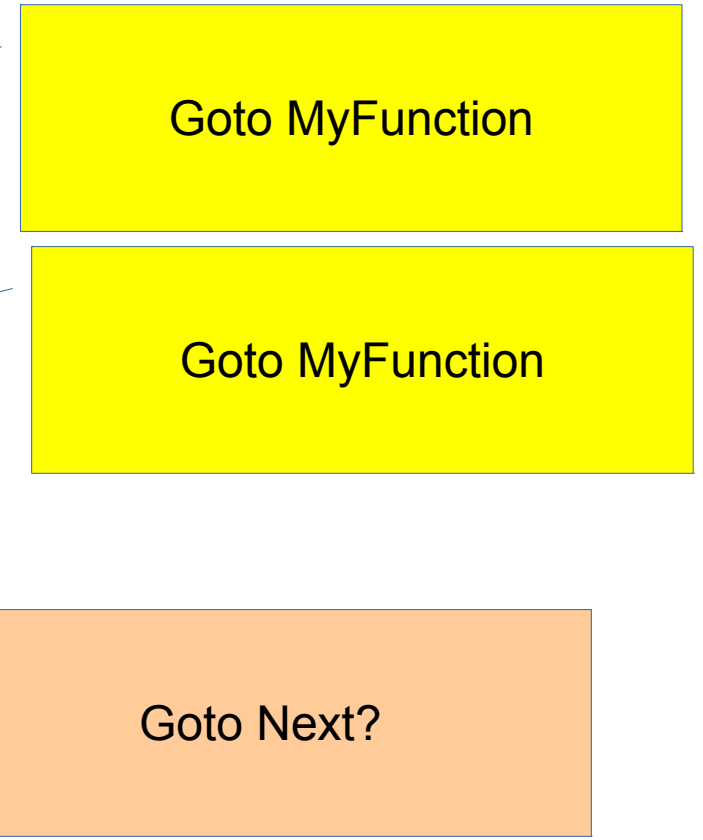
The stack

```
int X = 1;
void main()
{
	MyFunction();
Next1:
	X++;
	MyFunction();
Next2:
	X--;
}
void MyFunction()
{
// do some stuff
	return;
}
```



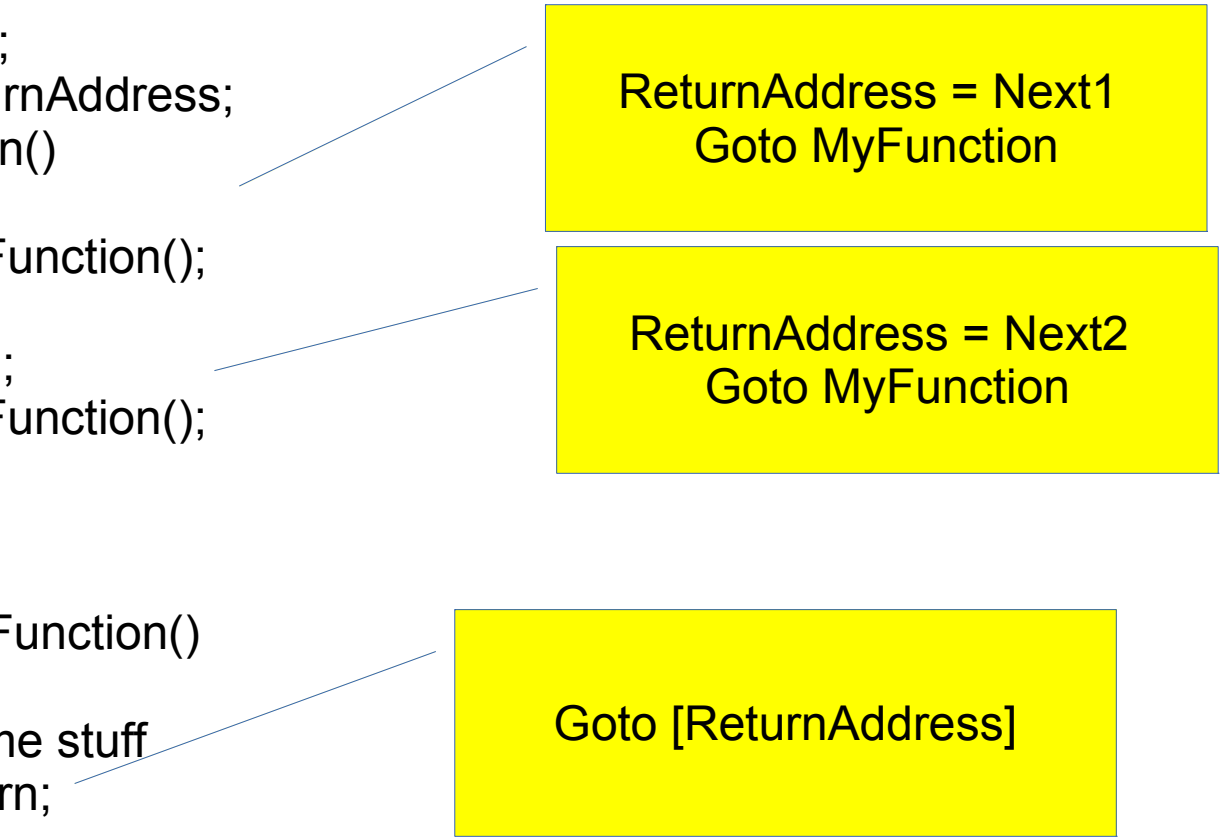
The stack

```
int X = 1;
void main()
{
	MyFunction();
Next1:
	X++;
	MyFunction();
Next2:
	X--;
}
void MyFunction()
{
// do some stuff
	return;
}
```



The stack

```
int X = 1;
int *ReturnAddress;
void main()
{
	MyFunction();
Next1:
	X++;
	MyFunction();
Next2:
	X--;
}
void MyFunction()
{
// do some stuff
	return;
}
```



The stack

```
int X = 1;
int *ReturnAddress;
void main()
{
	MyFunction();
Next1:
	X++;
	MyFunction();
Next2:
	X--;
}
void MyFunction()
{
// do some stuff
	MyOtherFunction()
Next3:
	return;
}
void MyOtherFunction()
{
	return;
}
```

ReturnAddress = Next1
Goto MyFunction

ReturnAddress = Next2
Goto MyFunction

ReturnAddress = Next3
Goto MyOtherFunction

Goto [ReturnAddress]

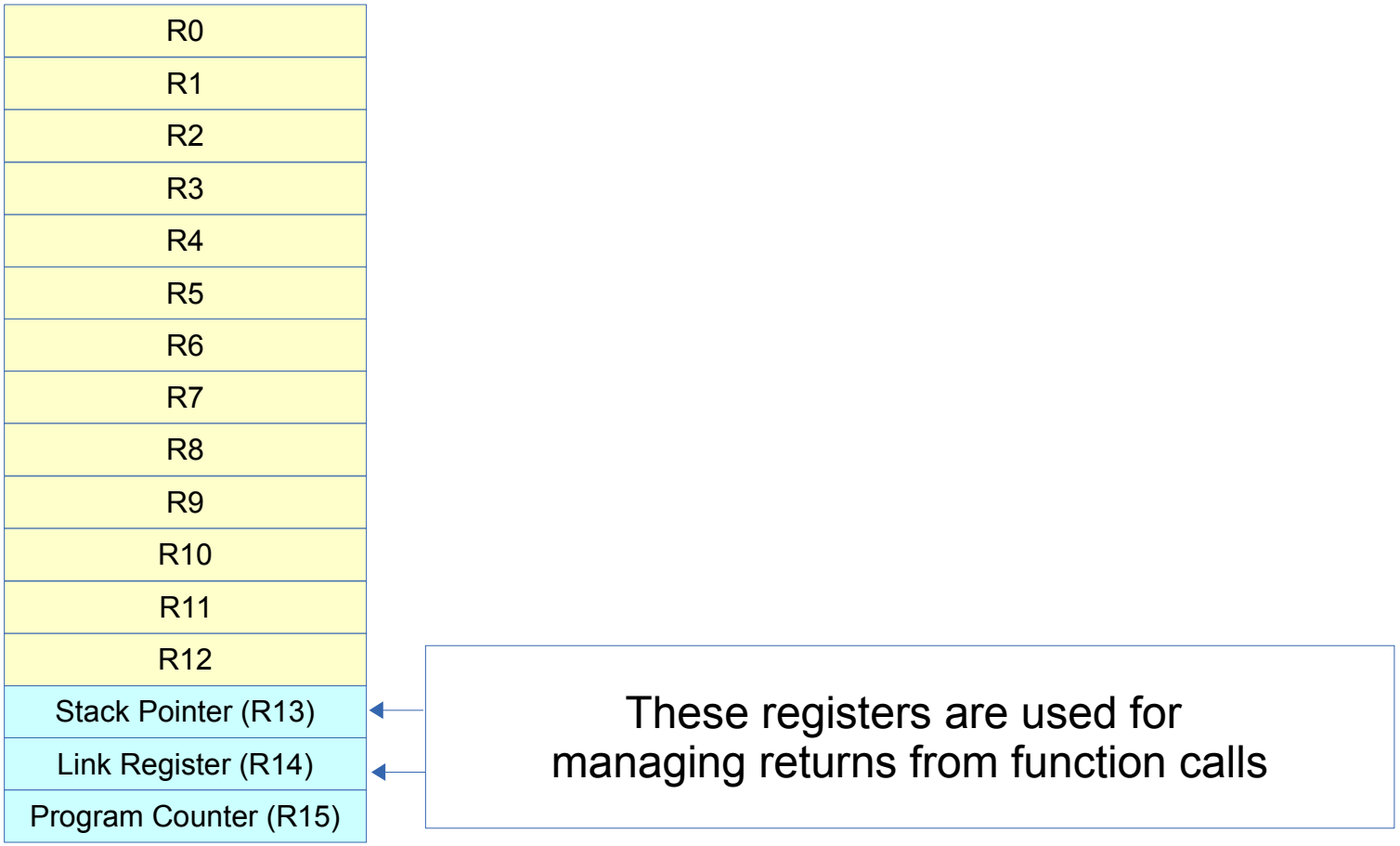
Goto [ReturnAddress]

The stack

- Using one global variable to store return addresses does not allow nested function calls
- More global variables will help, but how many?

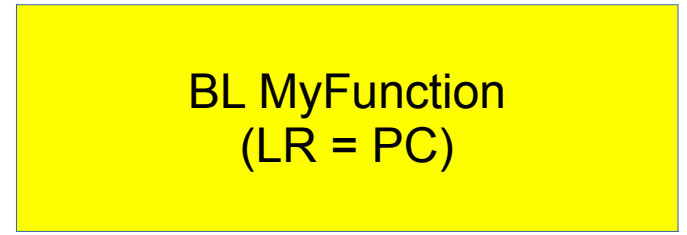
The stack

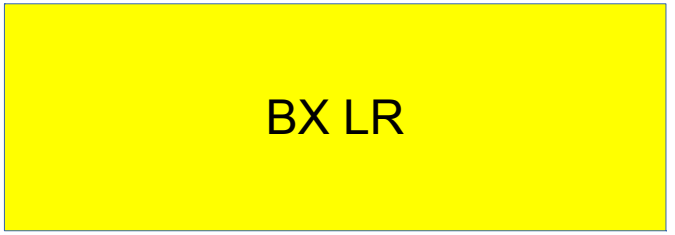
Cortex M0 Registers



The stack

```
int X = 1;
void main()
{
	MyFunction();
Next:
	X++;
}
void MyFunction()
{
// do some stuff
	return;
}
```





PC point to NEXT INSTRUCTION always

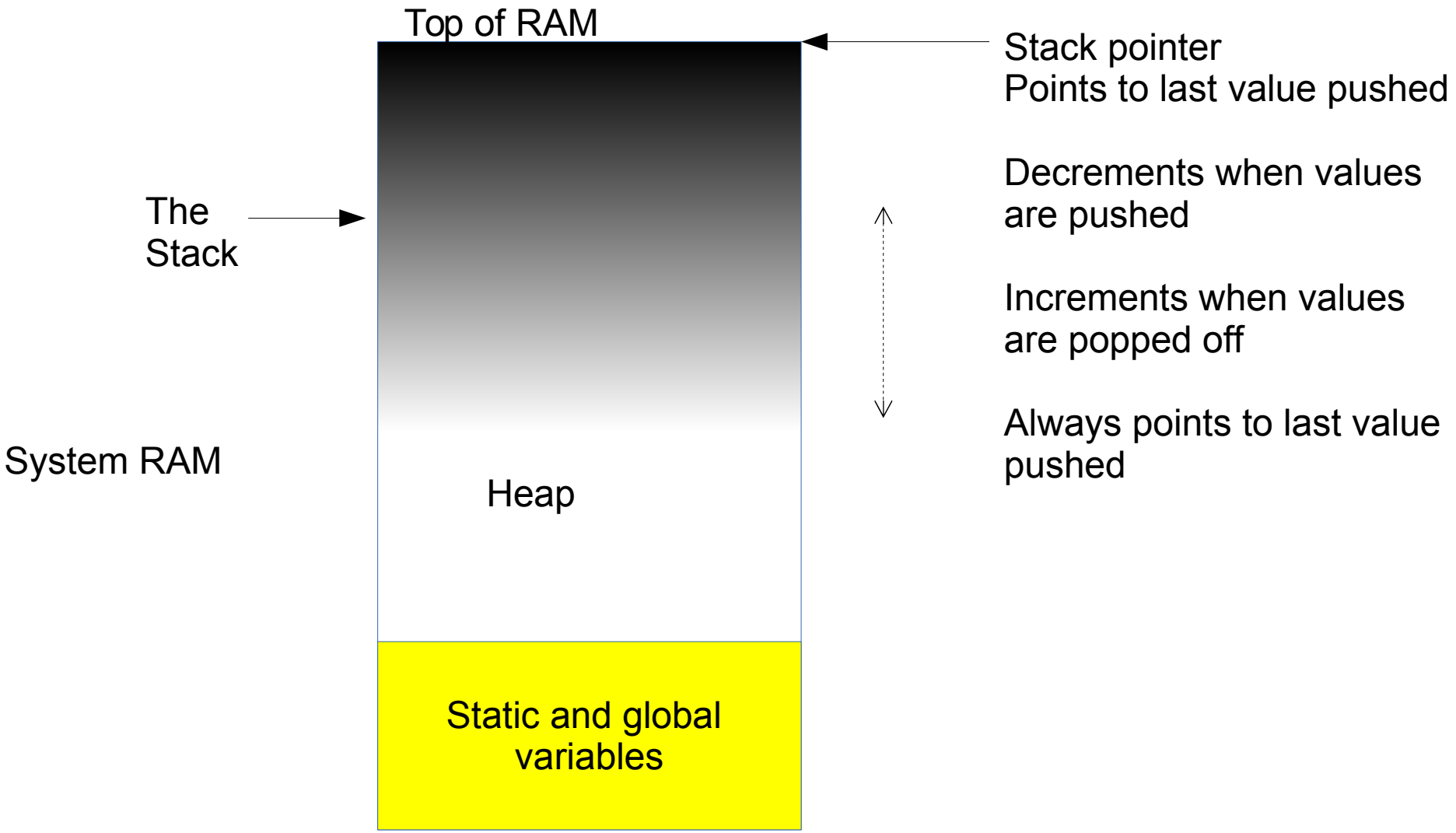
The stack

- The Link Register is a fast way of doing a function call one level deep
- On its own it does not allow for nested function calls.
- Another mechanism is needed to support nesting
- This mechanism requires a storage area, the size of which is difficult to determine up front

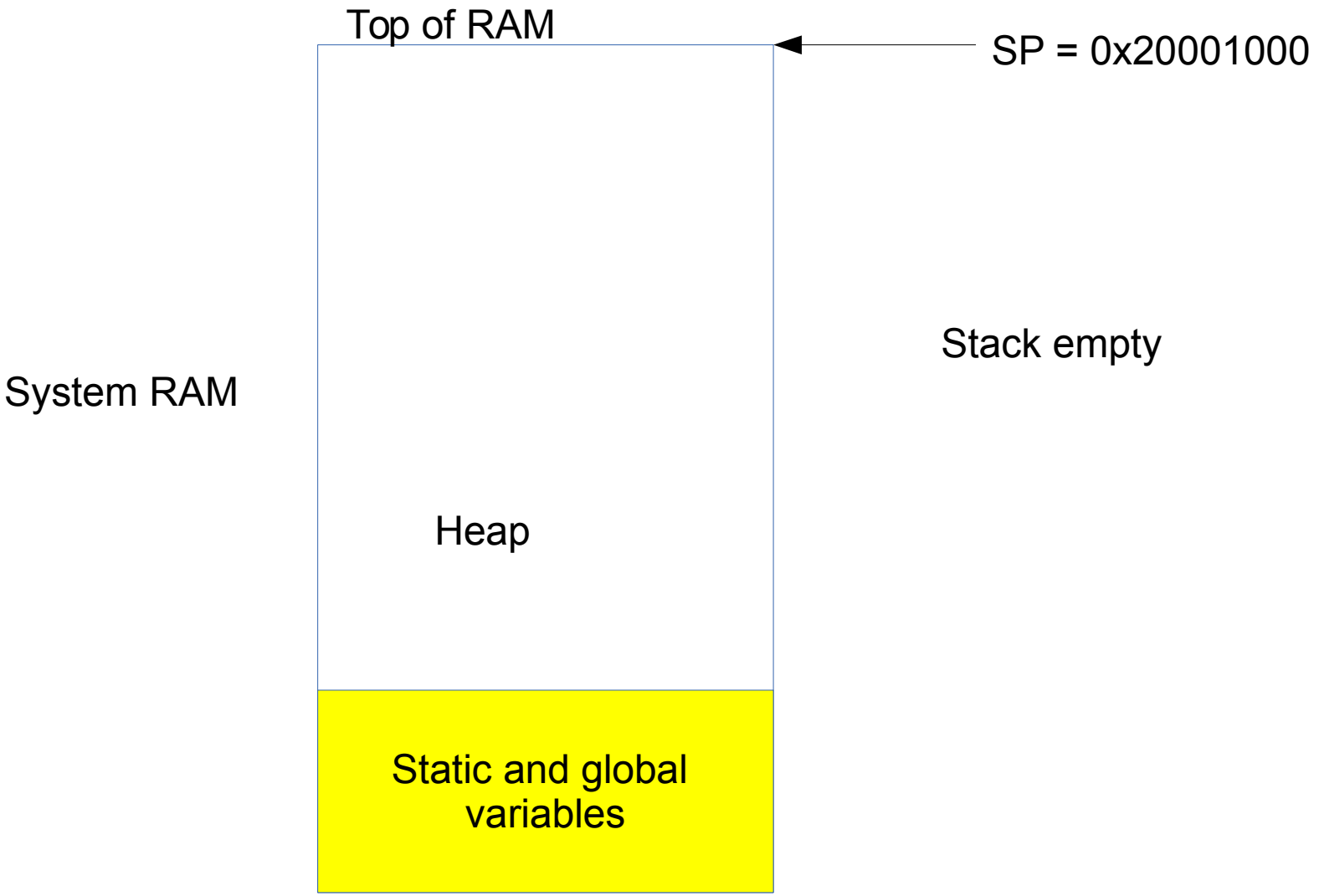
The stack

- Stack instructions
  - PUSH : Subtract 4 from SP. Copies the contents of a register to where SP points.
  - POP : Copy the contents of memory pointed to by SP into target register (4 bytes). Add 4 to SP
  - Push and Pop can work on sets of registers

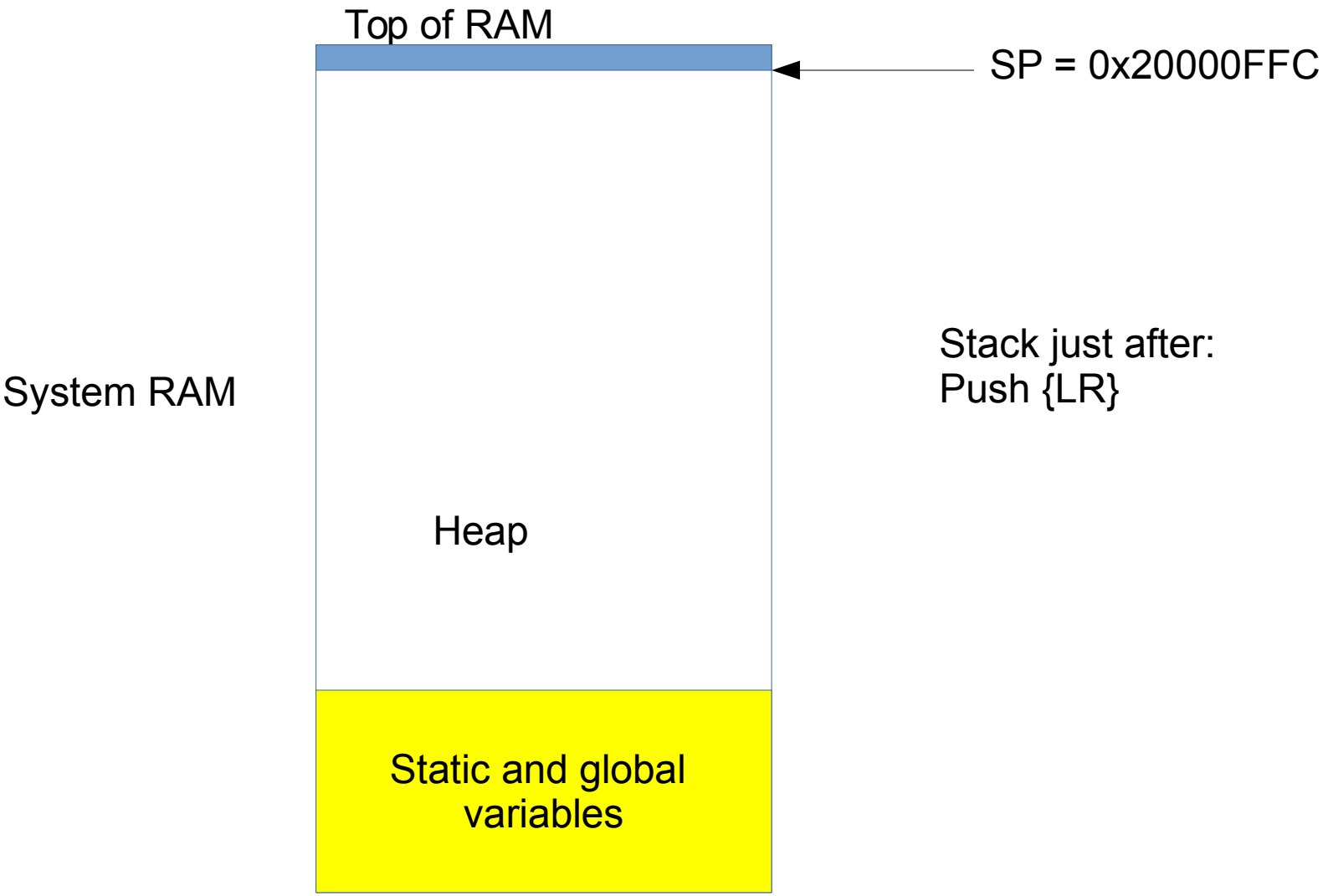
The stack


Top of RAM
Stack pointer
Points to last value pushed
Decrements when values are pushed
Increments when values are popped off
Always points to last value pushed
The Stack
System RAM
Heap
Static and global variables

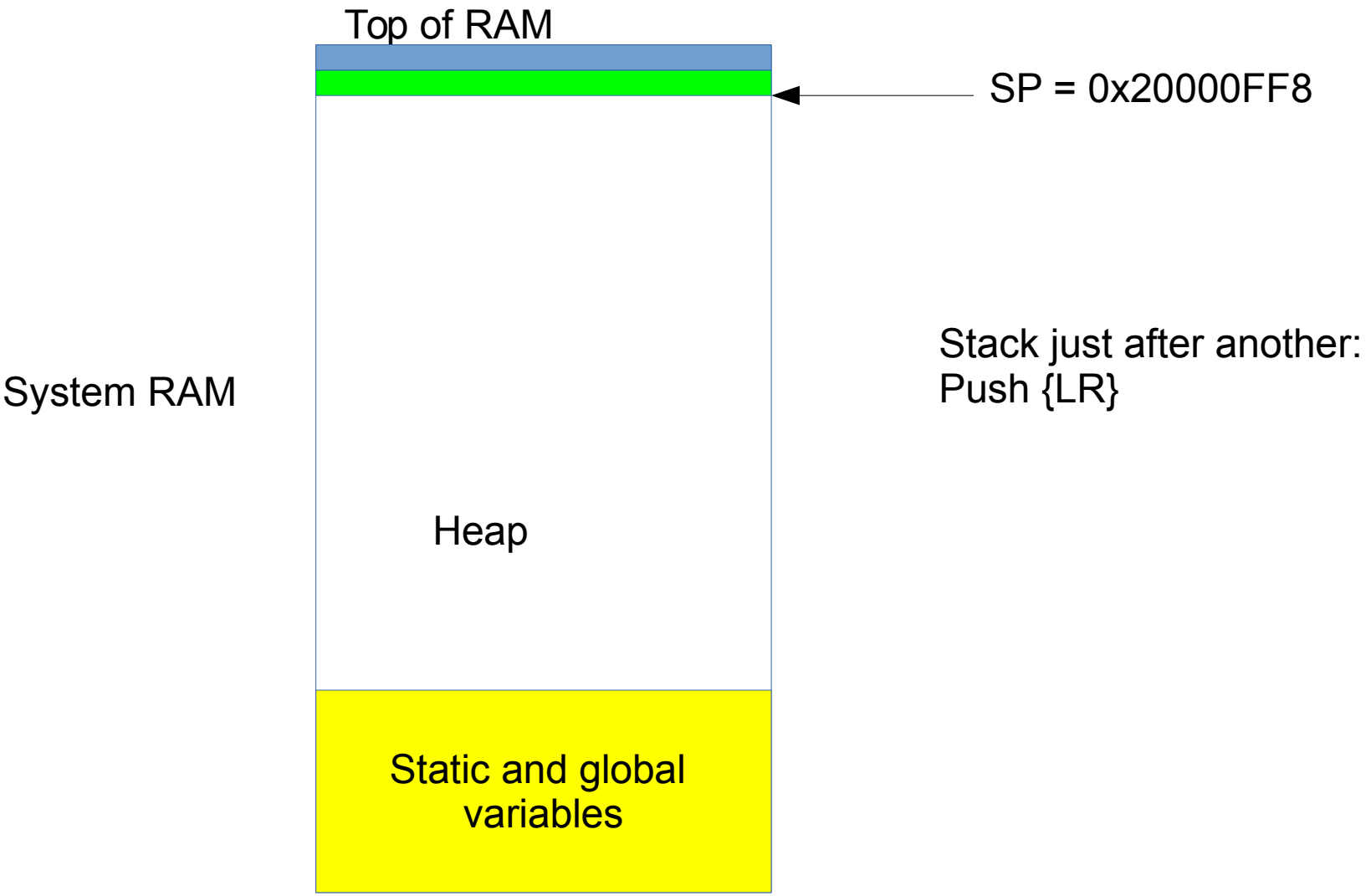
The stack


Top of RAM
SP = 0x20001000
Stack empty
System RAM
Heap
Static and global variables

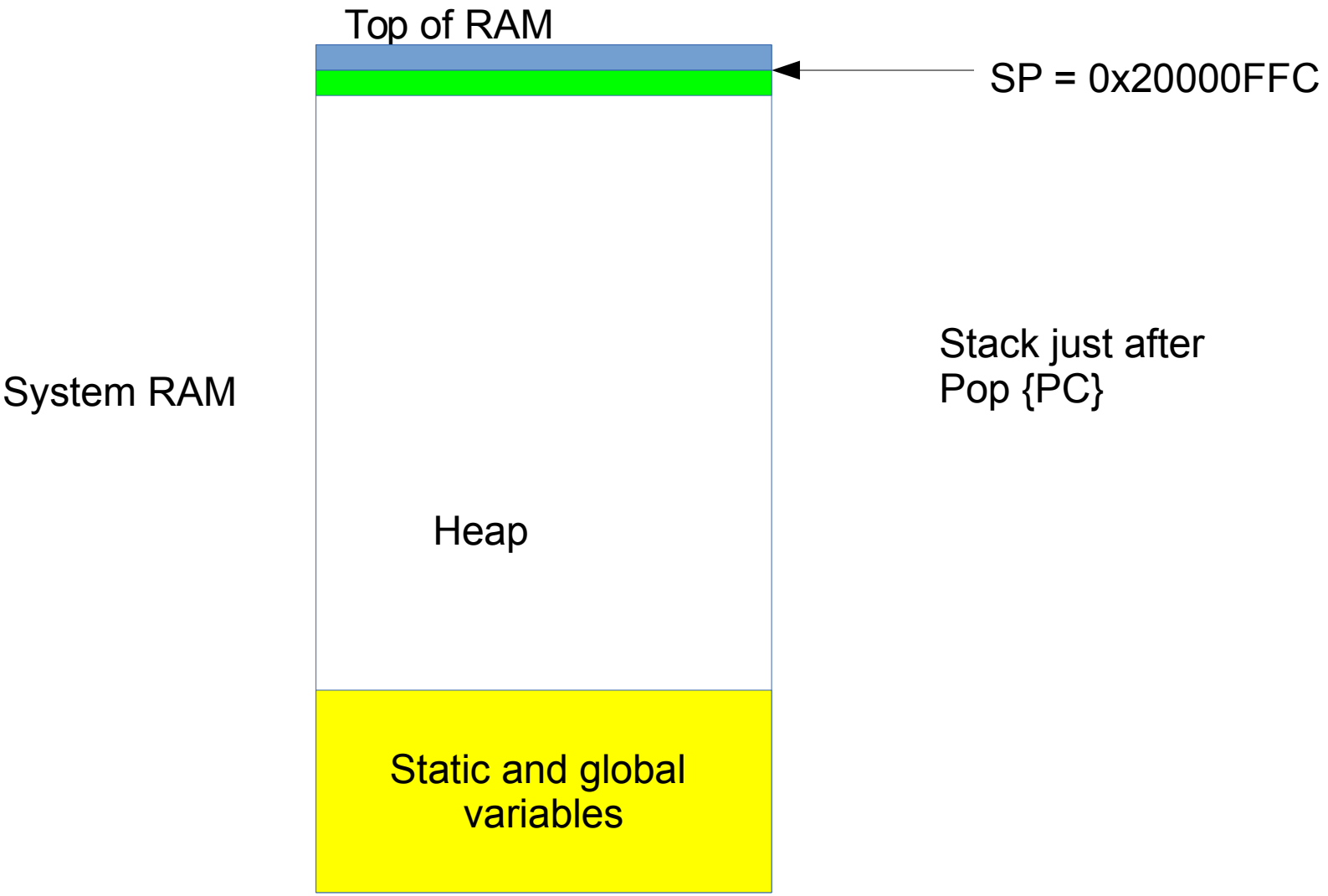
The stack


Top of RAM
SP = 0x20000FFC
System RAM
Stack just after:
Push {LR}
Heap
Static and global
variables

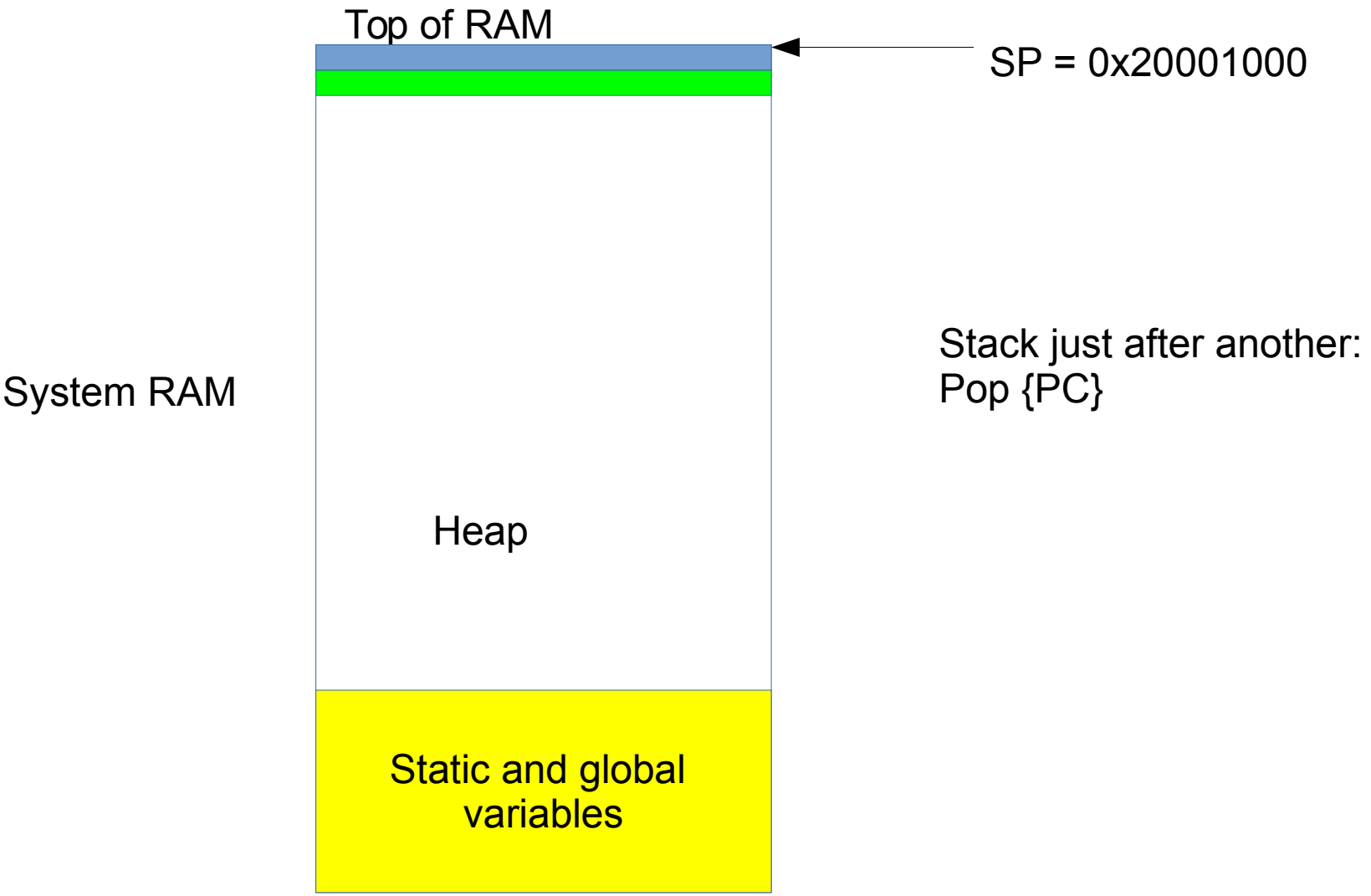
The stack


Top of RAM
SP = 0x20000FF8
System RAM
Stack just after another:
Push {LR}
Heap
Static and global
variables

The stack

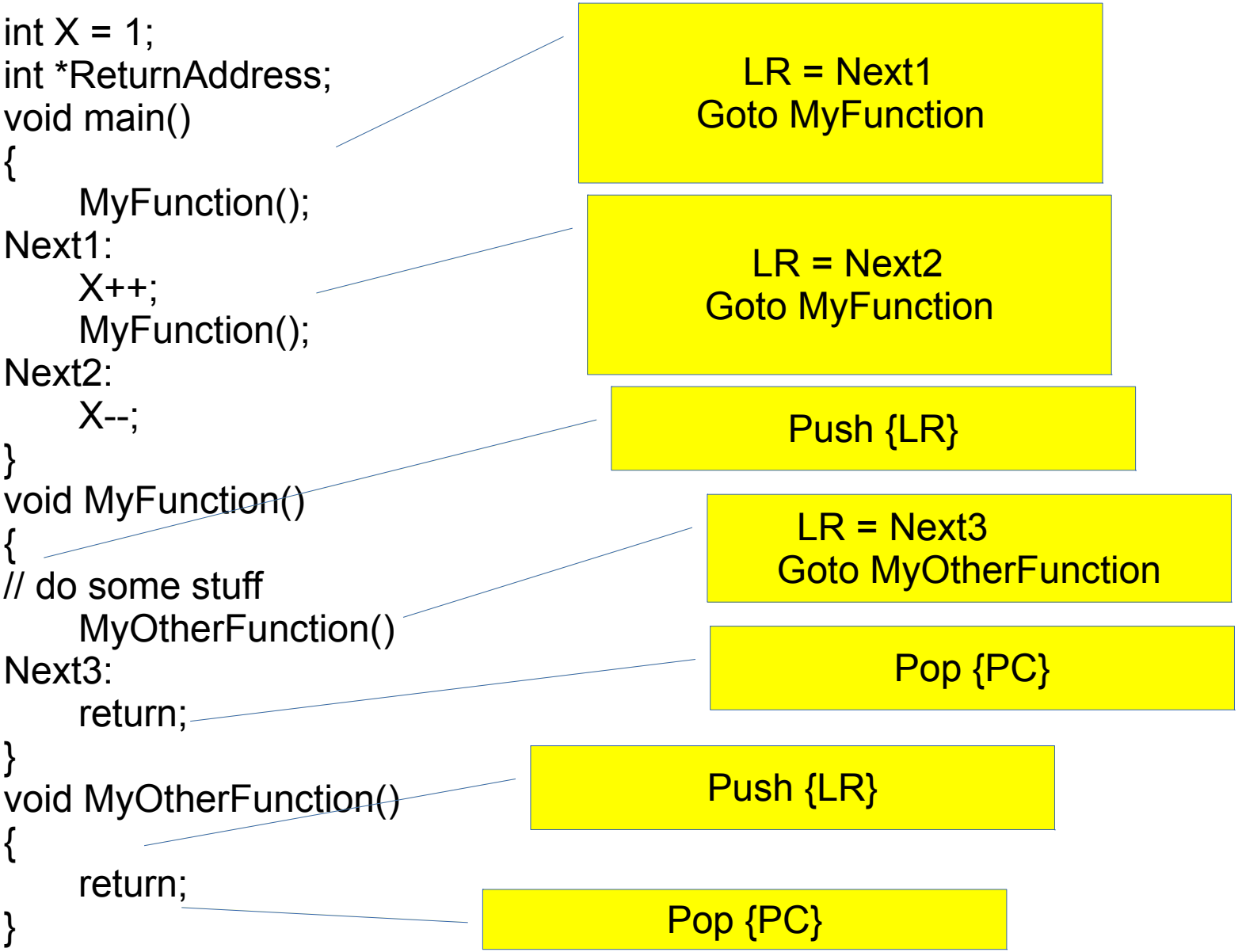

Top of RAM
SP = 0x20000FFC
System RAM
Stack just after
Pop {PC}
Heap
Static and global
variables

The stack

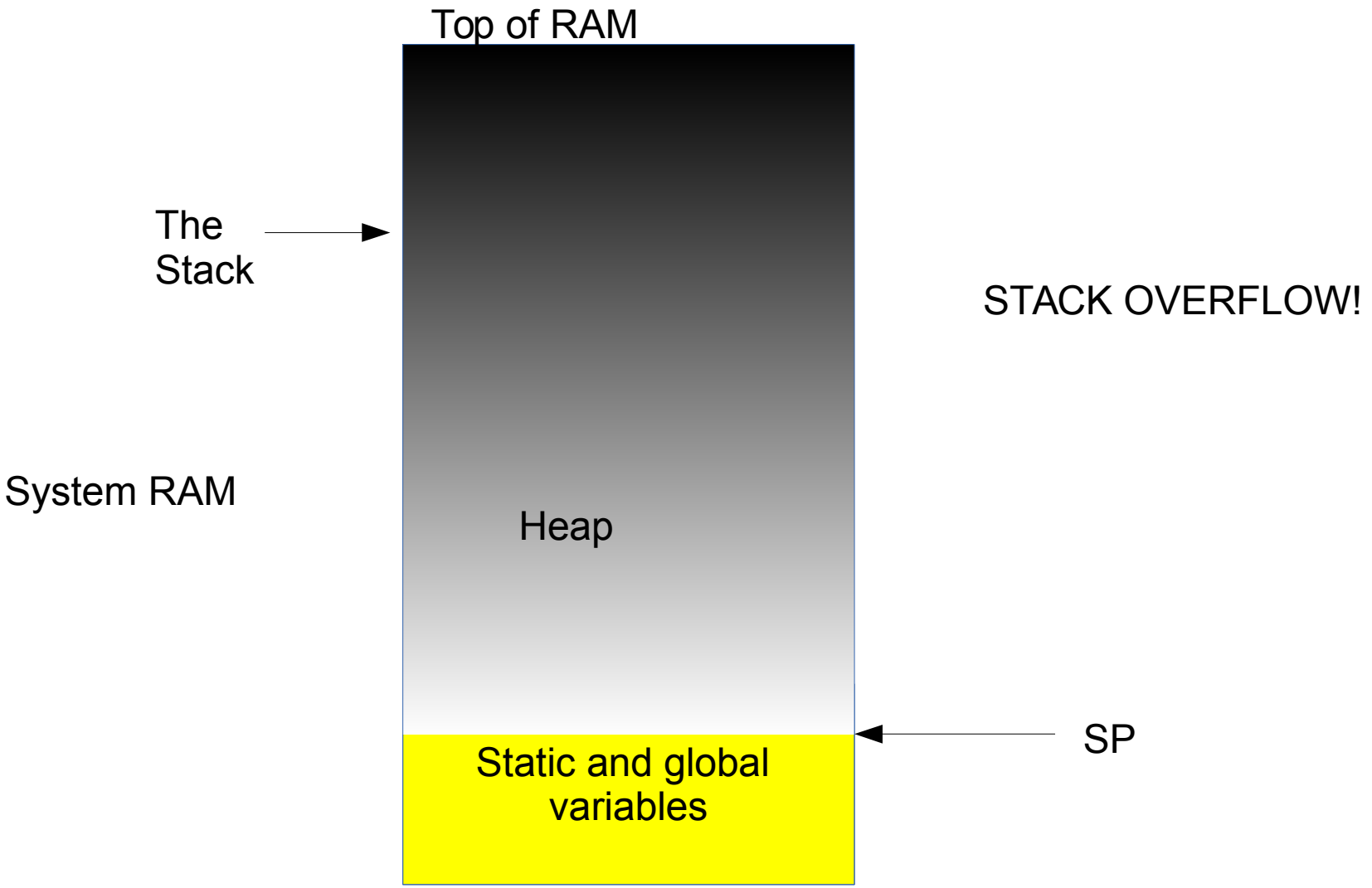

Top of RAM
SP = 0x20001000
System RAM
Stack just after another:
Pop {PC}
Heap
Static and global
variables

The stack

```
int X = 1;
int *ReturnAddress;
void main()
{
	MyFunction();
Next1:
	X++;
	MyFunction();
Next2:
	X--;
}
void MyFunction()
{
// do some stuff
	MyOtherFunction()
Next3:
	return;
}
void MyOtherFunction()
{
	return;
}
```



The stack


Top of RAM
The
Stack
STACK OVERFLOW!
System RAM
Heap
SP
Static and global
variables