#WWDC18

What's New in LLVM

Session 409

Jim Grosbach, Senior Manager Platform Technologies
Alex Lorenz, Clang Frontend Engineer
George Karpenkov, Program Analysis Engineer
Ahmed Bougacha, Compiler Backend Engineer

Family of Tools

Clang

Static Analyzer

Sanitizers

LLDB

GPU shader compilers

Family of Tools

Clang

Static Analyzer

Sanitizers

LLDB

GPU shader compilers

Swift

LLVM Open Source

https://llvm.org

Partnership

Partnership

Apple

Intel

Mozilla

Google

ARM

Qualcomm

Facebook

Sony

Partnership

And many more!

Participate!

https://llvm.org

Agenda

Updates on ARC

New diagnostics in Xcode 10

Clang Static Analyzer

Increased security

New instruction set extensions

Updates on ARC

Alex Lorenz, Clang Frontend Engineer

ARC object pointers in C structures!

```
typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;
```

ARC forbids Objective-C objects in struct

```
typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void orderFreeFood(NSString *name) {
  MenuItem item = {
    name,
    [NSNumber numberWithInt:0]
  };
  orderMenuItem(item);
}
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void orderFreeFood(NSString *name) {
  MenuItem item = {
    name,
    [NSNumber numberWithInt:0]
  };


  orderMenuItem(item);
}
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void orderFreeFood(NSString *name) {
  MenuItem item = {
    name,
    [NSNumber numberWithInt:0]
  };
  // [item.name retain];

  orderMenuItem(item);
}
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void orderFreeFood(NSString *name) {
  MenuItem item = {
    name,
    [NSNumber numberWithInt:0]
  };
  // [item.name retain];
  // [item.price retain];
  orderMenuItem(item);
}
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void orderFreeFood(NSString *name) {
  MenuItem item = {
    name,
    [NSNumber numberWithInt:0]
  };
  // [item.name retain];
  // [item.price retain];
  orderMenuItem(item);


}
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void orderFreeFood(NSString *name) {
  MenuItem item = {
    name,
    [NSNumber numberWithInt:0]
  };
  // [item.name retain];
  // [item.price retain];
  orderMenuItem(item);
  // [item.name release];

}
```

```
// ARC Object Pointers in C Structs!

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void orderFreeFood(NSString *name) {
  MenuItem item = {
    name,
    [NSNumber numberWithInt:0]
  };
  // [item.name retain];
  // [item.price retain];
  orderMenuItem(item);
  // [item.name release];
  // [item.price release];
}
```

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;
```

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items = malloc(10 * sizeof(MenuItem));
  orderMenuItems(items, 10);
  free(items);
}
```

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items = malloc(10 * sizeof(MenuItem));
  orderMenuItems(items, 10);
  free(items);
}
```

ARC pointers are not zero-initialized

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items = malloc(10 * sizeof(MenuItem));
  orderMenuItems(items, 10);
  free(items);
}
```

- ARC pointers are not zero-initialized
- ARC pointers are not cleared before deallocation

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items = malloc(10 * sizeof(MenuItem));
  orderMenuItems(items, 10);
  free(items);
}
```

- ARC pointers are not zero-initialized
- ARC pointers are not cleared before deallocation

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items =
  orderMenuItems(items, 10);
  free(items);
}
```

- ARC pointers are not zero-initialized
- ARC pointers are not cleared before deallocation

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items =
  orderMenuItems(items, 10);
  free(items);
}
```

- ARC pointers are not zero-initialized
- ARC pointers are not cleared before deallocation

```
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items = calloc(10, sizeof(MenuItem));
  orderMenuItems(items, 10);
  free(items);
}
```

ARC pointers are not cleared before deallocation

```objc
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items = calloc(10, sizeof(MenuItem));
  orderMenuItems(items, 10);
  // ARC Object Pointer Fields Must be Cleared Before Deallocation
  for (size_t i = 0; i < 10; ++i) {
    items[i].name = nil;
    items[i].price = nil;
  }
  free(items);
}
```

ARC pointers are not cleared before deallocation

```objc
// Structs with ARC Fields Need Care for Dynamic Memory Management

typedef struct {
  NSString *name;
  NSNumber *price;
} MenuItem;

void testMenuItems() {
  // Allocate an array of 10 menu items
  MenuItem *items = calloc(10, sizeof(MenuItem));
  orderMenuItems(items, 10);
  // ARC Object Pointer Fields Must be Cleared Before Deallocation
  for (size_t i = 0; i < 10; ++i) {
    items[i].name = nil;
    items[i].price = nil;
  }
  free(items);
}
```

Objective-C Objects in Structs

Objective-C Objects in Structs

Structs with Objective-C object fields interoperate across language modes

Objective-C Objects in Structs

Structs with Objective-C object fields interoperate across language modes

Unified Objective-C++ ABI between ARC and manual retain/release (MRR)

Objective-C Objects in Structs

Structs with Objective-C object fields interoperate across language modes

Unified Objective-C++ ABI between ARC and manual retain/release (MRR)

- ABI change for some functions passing or returning structs by value

Objective-C Objects in Structs

Structs with Objective-C object fields interoperate across language modes

Unified Objective-C++ ABI between ARC and manual retain/release (MRR)

- ABI change for some functions passing or returning structs by value

When a returned or passed struct has:

ARC object pointer fields and no special members (constructors, destructor)

Objective-C Objects in Structs

Structs with Objective-C object fields interoperate across language modes

Unified Objective-C++ ABI between ARC and manual retain/release (MRR)

- ABI change for some functions passing or returning structs by value

Swift **does not** support importing structs with ARC object pointer fields

New Diagnostics in Xcode 10

Swift and Objective-C Interoperability

Swift and Objective-C Interoperability

Swift code can be imported into Objective-C

Swift and Objective-C Interoperability

Swift code can be imported into Objective-C

Xcode generates a header that exposes Swift interfaces

Importing Swift Closures into Objective-C

Importing Swift Closures into Objective-C

```
@objc protocol Executor {
  func performOperation(handler: () -> Void)
}
```

Importing Swift Closures into Objective-C

```
@objc protocol Executor {
  func performOperation(handler: () -> Void)
}
```

Swift: Closure function arguments are non-escaping by default

Importing Swift Closures into Objective-C

```
@objc protocol Executor {
  func performOperation(handler: () -> Void)
}
```

Swift: Closure function arguments are non-escaping by default

- Should not be called after the function returns

Importing Swift Closures into Objective-C

```
@objc protocol Executor {
  func performOperation(handler: () -> Void)
}
```

Swift: Closure function arguments are non-escaping by default

- Should not be called after the function returns

Easy to forget when conforming to the protocol in Objective-C

Importing Swift Closures into Objective-C

```swift
@objc protocol Executor {
  func performOperation(handler: () -> Void)
}
```

```objc
#import “Executor-Swift.h”
@interface DispatchExecutor : NSObject<Executor>
 - (void) performOperation:(void (^)(void)) handler;
@end
```

Swift: Closure function arguments are non-escaping by default

- Should not be called after the function returns

Easy to forget when conforming to the protocol in Objective-C

```objc
#import “Executor-Swift.h”
@interface DispatchExecutor : NSObject<Executor>
 - (void) performOperation:(void (^)(void)) handler;
@end
```

```
// Warning for Missing Noescape Annotations for Method Overrides

#import "Executor-Swift.h"
@interface DispatchExecutor : NSObject<Executor>
 - (void) performOperation:(void (^)(void)) handler;
@end
```

⚠️ Parameter must be annotated with noescape

```
// Warning for Missing Noescape Annotations for Method Overrides

#import "Executor-Swift.h"
@interface DispatchExecutor : NSObject<Executor>
 - (void) performOperation:(NS_NOESCAPE void (^)(void)) handler;
@end
```

```
// Warning for Missing Noescape Annotations for Method Overrides

#import "Executor-Swift.h"
@interface DispatchExecutor : NSObject<Executor>
 - (void) performOperation:(NS_NOESCAPE void (^)(void)) handler;
@end

@implementation DispatchExecutor
- (void) performOperation:(NS_NOESCAPE void (^)(void)) handler {
}
@end
```

```
// Warning for Missing Noescape Annotations for Method Overrides

#import "Executor-Swift.h"
@interface DispatchExecutor : NSObject<Executor>
 - (void) performOperation:(NS_NOESCAPE void (^)(void)) handler;
@end

@implementation DispatchExecutor
- (void) performOperation:(NS_NOESCAPE void (^)(void)) handler {
  // Programmer must ensure that handler is not called after performOperation returns
}
@end
```

Packing Struct Members with #pragma pack

```
struct Struct {
  uint8_t a, b;
  // 2 byte padding
  uint32_t c;
};
```

Packing Struct Members with #pragma pack

```
struct Struct {
  uint8_t a, b;
  uint32_t c;
};
```

Packing Struct Members with #pragma pack

```
struct PackedStruct {
  uint8_t a, b;
  uint32_t c;
};
```

Packing Struct Members with #pragma pack

```
#pragma pack (push, 1)
struct PackedStruct {
  uint8_t a, b;
  uint32_t c;
};
#pragma pack (pop)
```

Packing Struct Members with #pragma pack

```
#pragma pack (push, 1)
struct PackedStruct {
  uint8_t a, b;
  uint32_t c;
};
// The programmer forgot #pragma pack (pop)
```

Finding Unbalanced #pragma pack Directives

```
#pragma pack (push, 1)
struct PackedStruct {
  uint8_t a, b;
  uint32_t c;
};
// The programmer forgot #pragma pack (pop)
```

Finding Unbalanced #pragma pack Directives

```
#pragma pack (push, 1)
struct PackedStruct {
  uint8_t a, b;
  uint32_t c;
};
// The programmer forgot #pragma pack (pop)
```

⚠️ Unterminated ‘#pragma pack (push, ...)’ at end of file

Finding Unbalanced #pragma pack Directives

```
#pragma pack (push, 1)
struct PackedStruct {
  uint8_t a, b;
  uint32_t c;
};
#pragma pack (pop)
```

⚠️ Unterminated ‘#pragma pack (push, ...)’ at end of file

Finding Unbalanced #pragma pack Directives

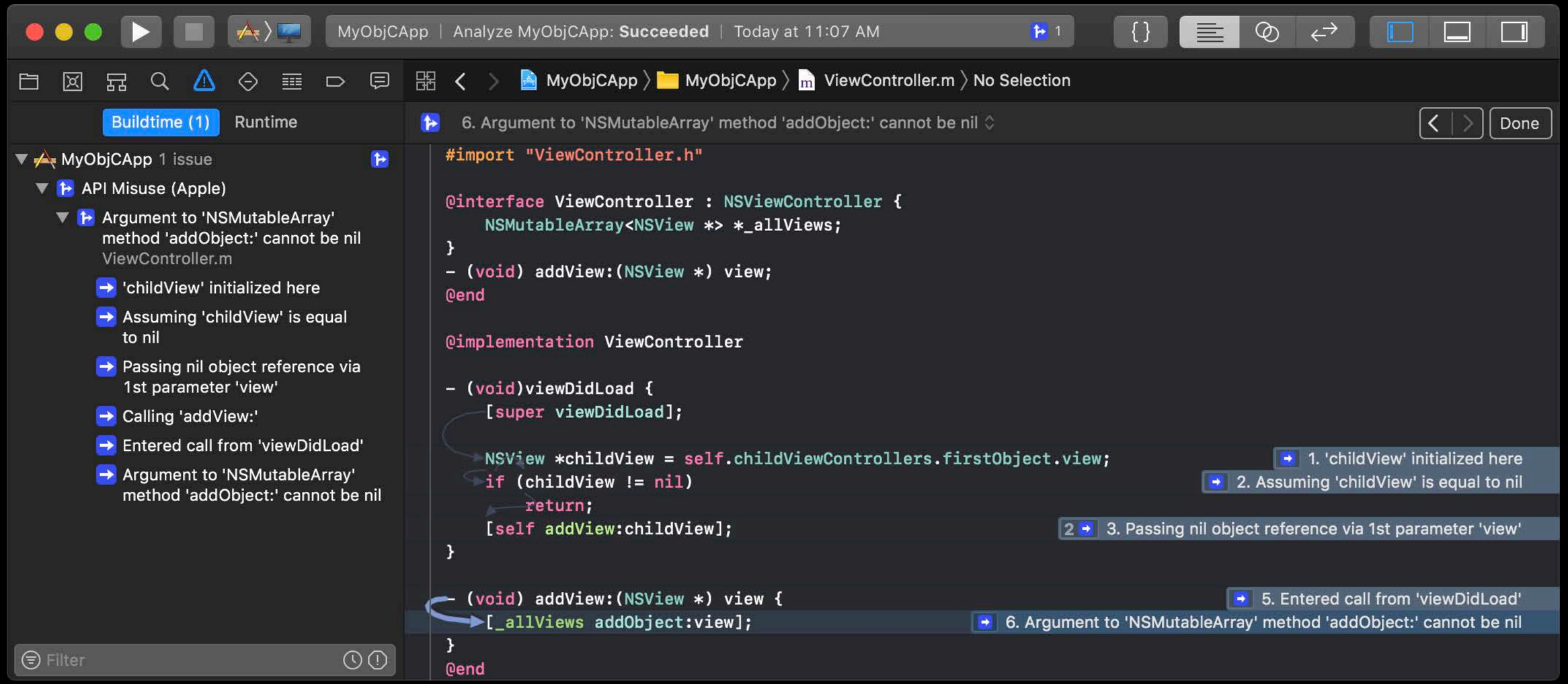
```
#pragma pack (push, 1)
struct PackedStruct {
  uint8_t a, b;
  uint32_t c;
};
#pragma pack (pop)
```

Clang Static Analyzer

George Karpenkov, Program Analysis Engineer

Clang Static Analyzer

Finds edge-case, hard-to-reproduce bugs



Static Analyzer Improvements

NEW

Grand Central Dispatch performance anti-pattern

Autoreleasing variables outliving autorelease pool

Improved performance and report visualizations

Grand Central Dispatch Performance Anti-Pattern

Grand Central Dispatch Performance Anti-Pattern

Performance implications: slowdown and hangs

```
__block NSString *taskName = nil;
dispatch_semaphore_t sema = dispatch_semaphore_create(0);
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
    dispatch_semaphore_signal(sema);
}];
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
return taskName;
```

Grand Central Dispatch Performance Anti-Pattern

Performance implications: slowdown and hangs

```
__block NSString *taskName = nil;
dispatch_semaphore_t sema = dispatch_semaphore_create(0);
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
    dispatch_semaphore_signal(sema);
}];
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
return taskName;
```

Grand Central Dispatch Performance Anti-Pattern

Performance implications: slowdown and hangs

```
__block NSString *taskName = nil;
dispatch_semaphore_t sema = dispatch_semaphore_create(0);
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
    dispatch_semaphore_signal(sema);
}];
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
return taskName;
```

Grand Central Dispatch Performance Anti-Pattern

Performance implications: slowdown and hangs

```
__block NSString *taskName = nil;
dispatch_semaphore_t sema = dispatch_semaphore_create(0);
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
    dispatch_semaphore_signal(sema);
}];
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
return taskName;
```

Grand Central Dispatch Performance Anti-Pattern

Performance implications: slowdown and hangs

```
__block NSString *taskName = nil;
dispatch_semaphore_t sema = dispatch_semaphore_create(0);
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
    dispatch_semaphore_signal(sema);
}];
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
return taskName;
```

Blocks current thread by the execution on a different queue

The task queue usually has lower priority, leads to priority inversion

Spawns useless threads

Grand Central Dispatch Performance Anti-Pattern

Performance implications: slowdown and hangs

```
__block NSString *taskName = nil;
dispatch_semaphore_t sema = dispatch_semaphore_create(0);
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
    dispatch_semaphore_signal(sema);
}];
dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
return taskName;
```

Waiting on a callback

Blocks current thread by the execution on a different queue

The task queue usually has lower priority, leads to priority inversion

Spawns useless threads

Grand Central Dispatch Performance Anti-Pattern

```
__block NSString *taskName = nil;
id remoteObjectProxy = [self.connection synchronousRemoteObjectProxyWithErrorHandler:
    ^(NSError *error) { NSLog(@"Error: %@", error); }];
[remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
}];
return taskName;
```

Grand Central Dispatch Performance Anti-Pattern

```
__block NSString *taskName = nil;
id remoteObjectProxy = [self.connection synchronousRemoteObjectProxyWithErrorHandler:
    ^(NSError *error) { NSLog(@"Error: %@", error); }];
[remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
    taskName = task;
}];
return taskName;
```

If available, use synchronous version of the API

Grand Central Dispatch Performance Anti-Pattern

```
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
  completionHandler(task);
}];
```

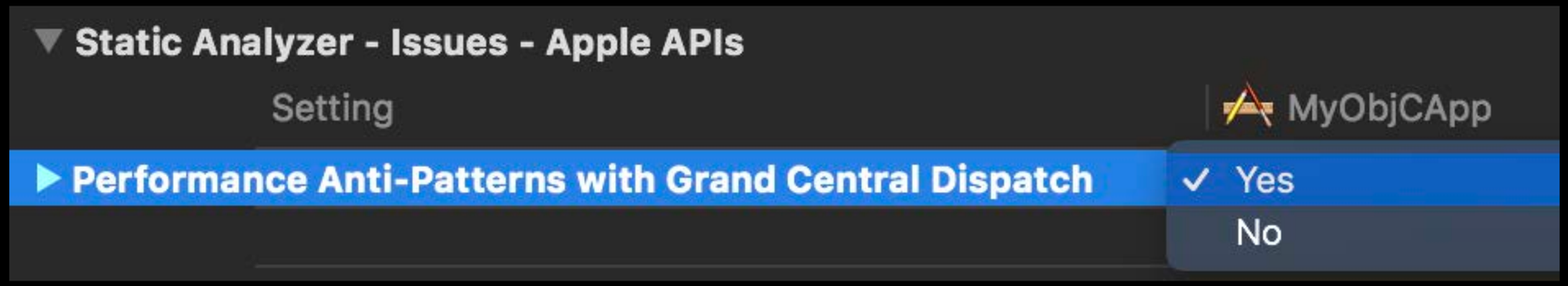
Grand Central Dispatch Performance Anti-Pattern

```
[self.connection.remoteObjectProxy requestCurrentTaskName:^(NSString *task) {
  completionHandler(task);
}];
```

Alternatively, use the API in an asynchronous manner

Detecting Grand Central Dispatch Anti-Pattern

Enabling check in build settings



Autoreleasing Variables Outliving Autorelease Pool

Autoreleasing Variables and Autoreleasing Pools

Using Automatic Reference Counting

```
@autoreleasepool {
    __autoreleasing NSError *err = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
}
```

Autoreleasing qualifier: values released on exit from the autorelease pool

```objc
- (void) validateProperty:(NSError **) error { // implicitly __autoreleasing
   @autoreleasepool {
    if (error)
       *error = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
   }
}
```

Out parameters are implicitly `__autoreleasing`

A subsequent read of error parameter will crash

Out Parameters and Autorelease Pools

A common source of use-after-free crashes

```
- (void) validateProperty:(NSError **) error { // implicitly __autoreleasing
   @autoreleasepool {
    if (error)
       *error = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
   }
}
```

Out parameters are implicitly `__autoreleasing`

A subsequent read of error parameter will crash

Out Parameters and Autorelease Pools

A common source of use-after-free crashes

```
- (void) validateProperty:(NSError **) error { // implicitly __autoreleasing
   @autoreleasepool {
    if (error)
        *error = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
   }
}
```

Out parameters are implicitly `__autoreleasing`

A subsequent read of error parameter will crash

Out Parameters and Autorelease Pools

A common source of use-after-free crashes

```
- (void) validateProperty:(NSError **) error { // implicitly __autoreleasing
   @autoreleasepool {
    if (error)
       *error = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
   }
}
```

Out parameters are implicitly `__autoreleasing`

A subsequent read of error parameter will crash

Capturing Autoreleasing Variables in a Block

Dangerous when library uses autorelease pools

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
    if ([value isEqualToString:@"problem"]) {
      if (error)
        *error =
            [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
    }
  }];
}
```

`-enumerateObjectsUsingBlock:` runs the block inside the autorelease pool

A subsequent read of the error parameter may crash

Capturing Autoreleasing Variables in a Block

Dangerous when library uses autorelease pools

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
    if ([value isEqualToString:@"problem"]) {
      if (error)
        *error =
            [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
    }
  }];
}
```

`-enumerateObjectsUsingBlock:` runs the block inside the autorelease pool

A subsequent read of the error parameter may crash

Capturing Autoreleasing Variables in a Block

Dangerous when library uses autorelease pools

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
    if ([value isEqualToString:@"problem"]) {
      if (error)
        *error =
            [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
    }
  }];
}
```

`-enumerateObjectsUsingBlock:` runs the block inside the autorelease pool

A subsequent read of the error parameter may crash

Capturing Autoreleasing Variables in a Block

Dangerous when library uses autorelease pools

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
    if ([value isEqualToString:@"problem"]) {
      if (error)
        *error =
            [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
    }
  }];
}
```

`-enumerateObjectsUsingBlock:` runs the block inside the autorelease pool

A subsequent read of the error parameter may crash

Xcode 10: Check for __autoreleasing Misuse

Common source of use-after-free

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
    if ([value isEqualToString:@"problem"]) {
      if (error)
        *error =
            [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
    }
  }];
}
```

Xcode 9: introduced a compiler warning requiring explicit `__autoreleasing`

Xcode 10: more targeted static analyzer warning

Xcode 10: Check for __autoreleasing Misuse

Common source of use-after-free

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
    if ([value isEqualToString:@"problem"]) {
      if (error)
        *error =                    Write to autoreleasing out parameter inside autorelease pool
            [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
    }
  }];
}
```

Xcode 9: introduced a compiler warning requiring explicit `__autoreleasing`

Xcode 10: more targeted static analyzer warning

Capturing Autoreleasing Variables

Write to strong variables first

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
    __block NSError *localError;
    [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
        if ([value isEqualToString:@"problem"]) {
            localError = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
        }
    }];
    if (error) {
        *error = localError;
    }
}
```

Capturing Autoreleasing Variables

Write to strong variables first

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  __block NSError *localError;
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
       if ([value isEqualToString:@"problem"]) {
           localError = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
      }
  }];
  if (error) {
      *error = localError;
  }
}
```

Capturing Autoreleasing Variables

Write to strong variables first

```
- (void) findProblems:(NSArray *)arr error:(NSError *__autoreleasing*) error {
  __block NSError *localError;
  [arr enumerateObjectsUsingBlock:^(id value, NSUInteger idx, BOOL *stop) {
       if ([value isEqualToString:@"problem"]) {
           localError = [NSError errorWithDomain:@"domain" code:1 userInfo:nil];
      }
  }];
  if (error) {
       *error = localError;
  }
}
```

Improved Performance and Report Visualizations

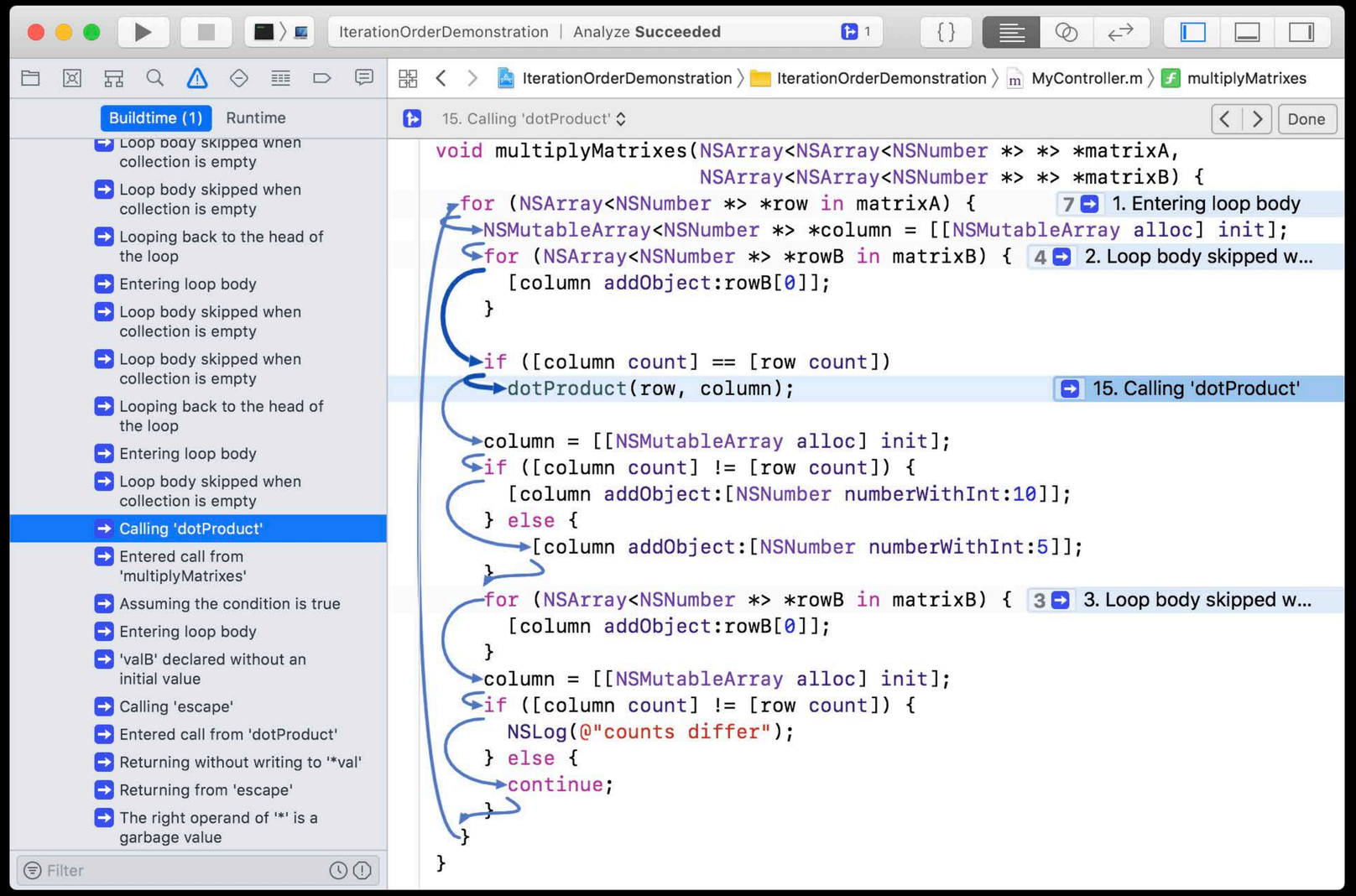
Improved Performance and Report Visualizations

Explores your program in a more efficient way

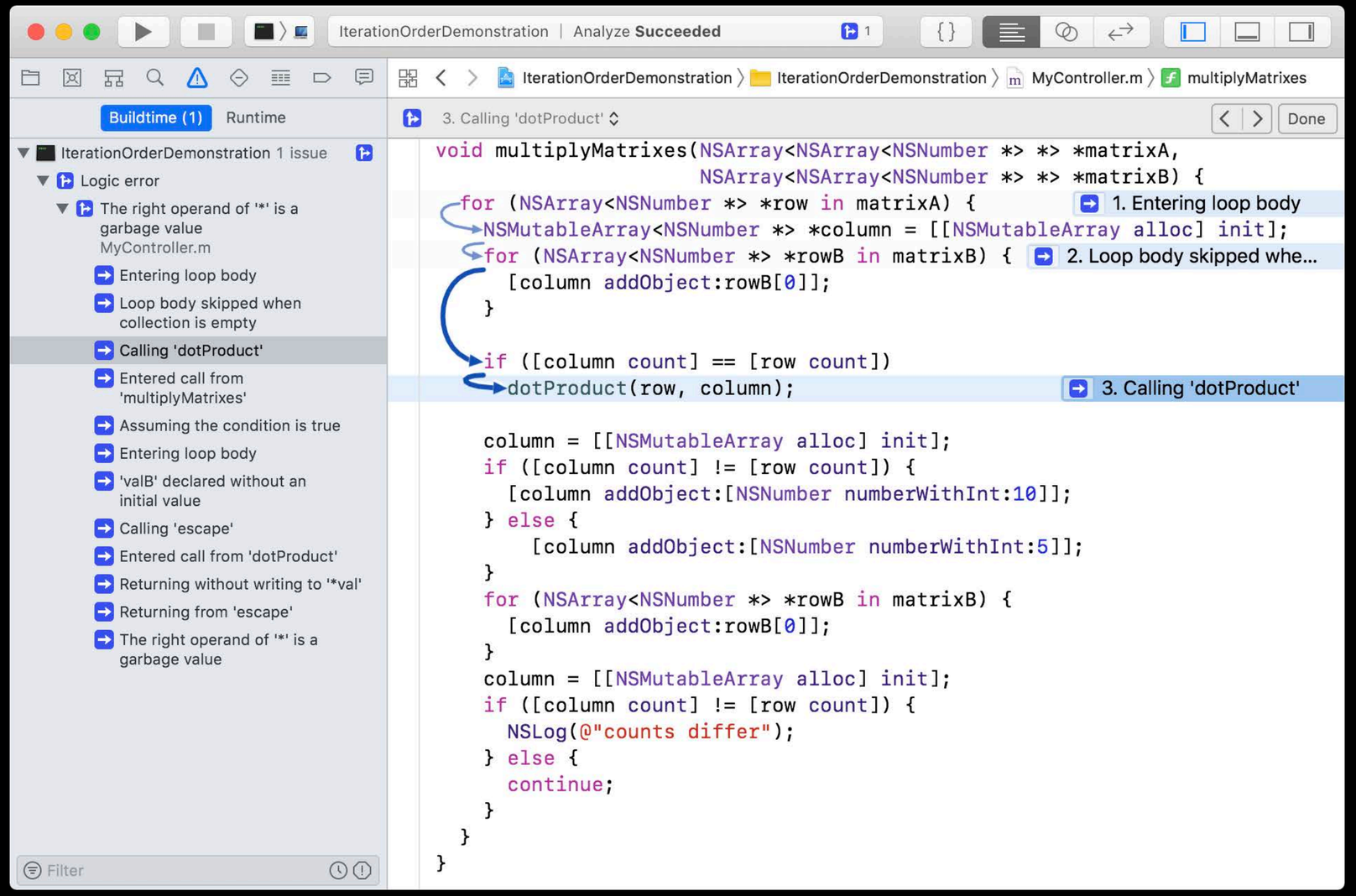
15% average increase in number of found bugs

Within the same analysis time

Xcode 9: Unnecessary Long Error Paths

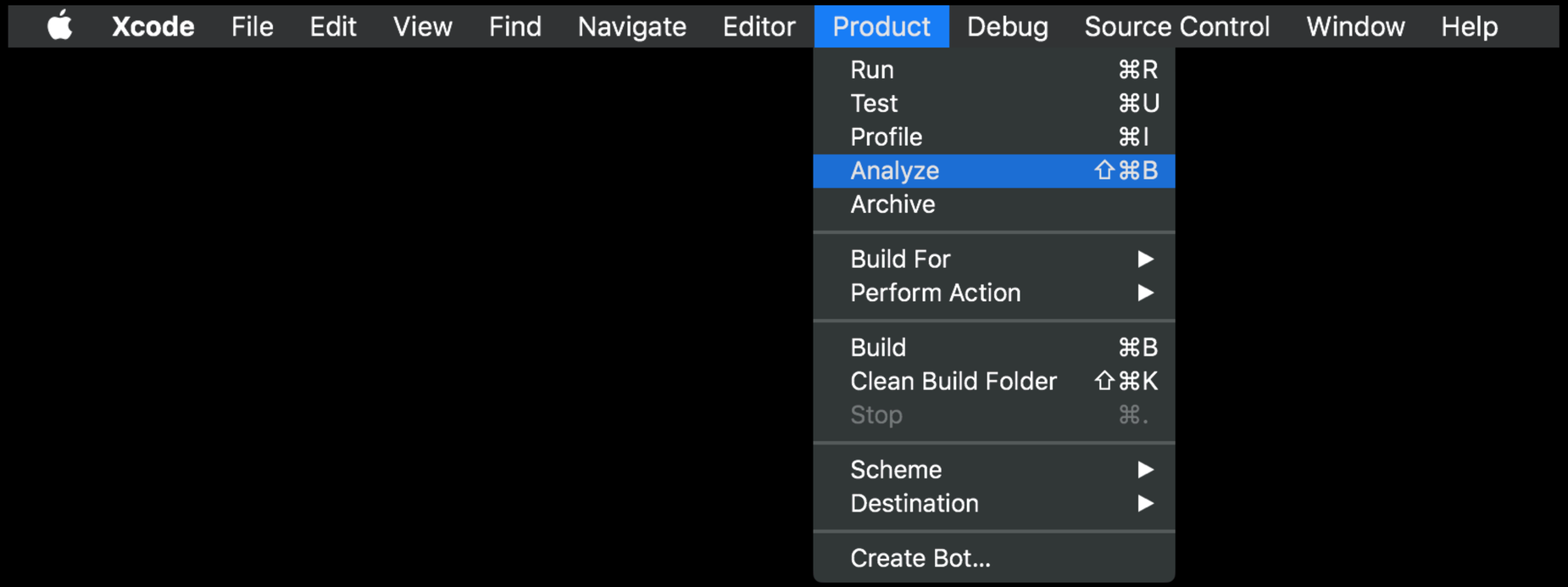


Xcode 10: Improved Error Reporting



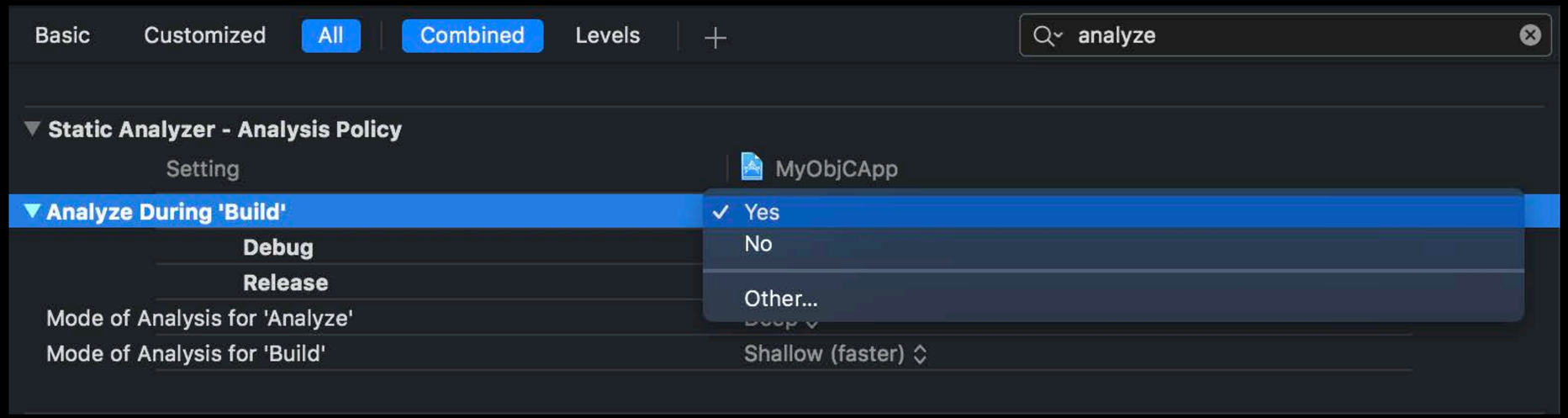
Use Clang Static Analyzer

Finds your bugs before your users do



Use Clang Static Analyzer

Finds your bugs before your users do



Increased Security

Ahmed Bougacha, Compiler Backend Engineer

```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

```c
int main(void) {
  dlog("Hello");
  ...
}
```

```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

```c
int main(void) {
  dlog("Hello");
  ...
}
```

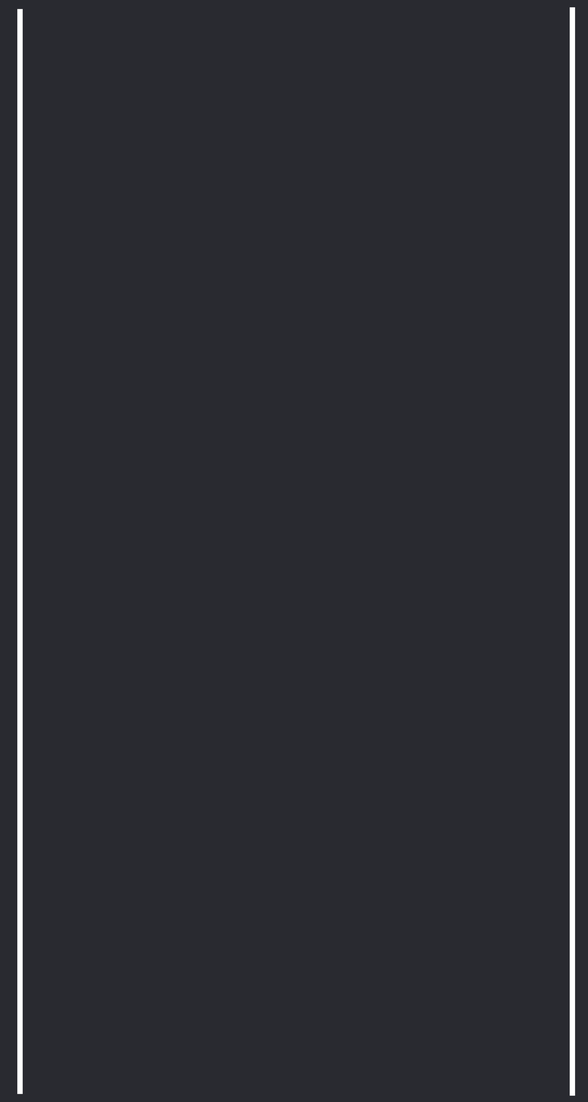
```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

```c
int main(void) {
  dlog("Hello");
  ...
}
```

```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

```c
int main(void) {
  dlog("Hello");
  ...
}
```

```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}

int main(void) {
  dlog("Hello");
  ...
}
```

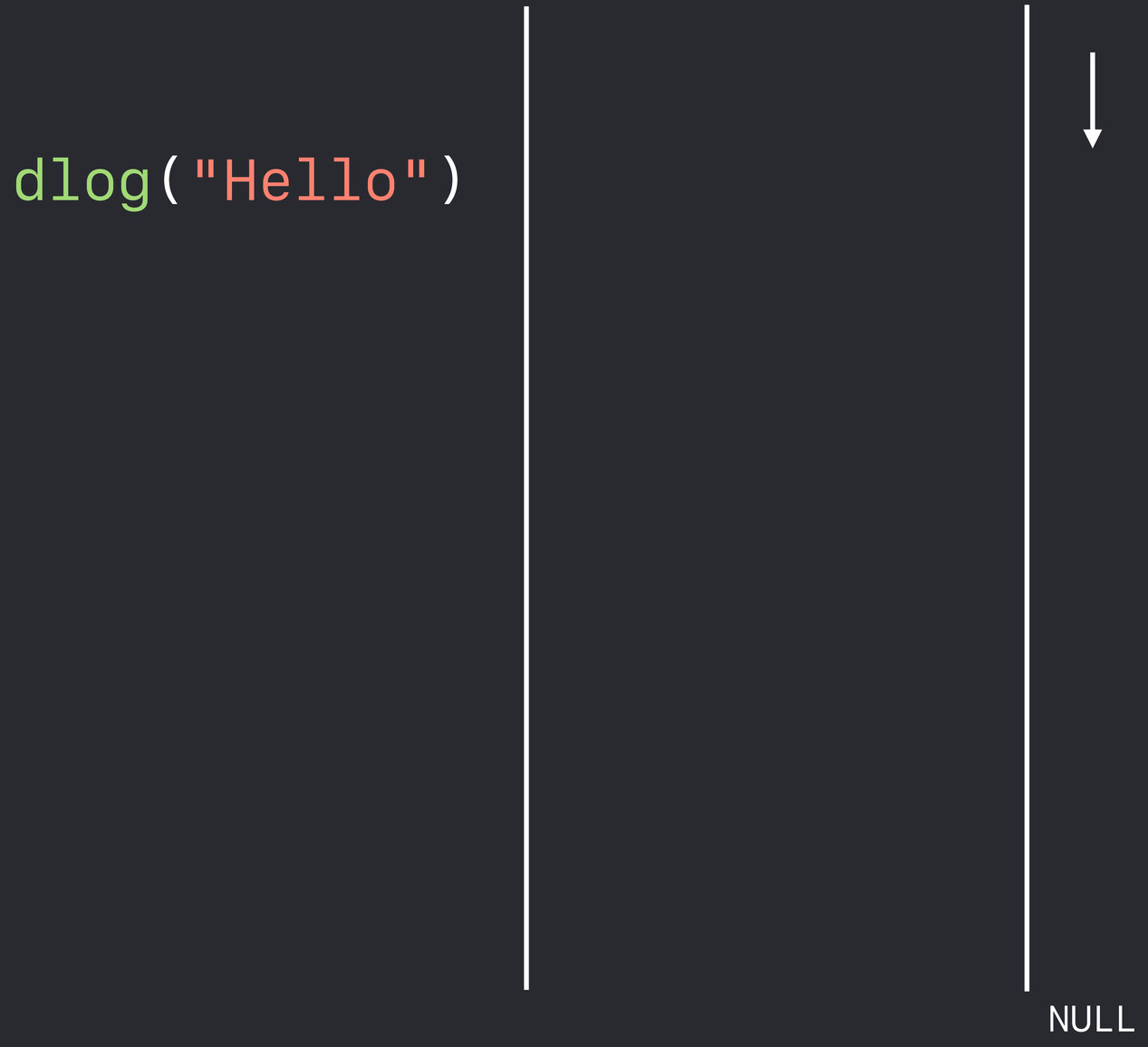


```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

```c
int main(void) {
  dlog("Hello");
  ...
}
```

```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}

int main(void) {
  dlog("Hello");
  ...
}
```

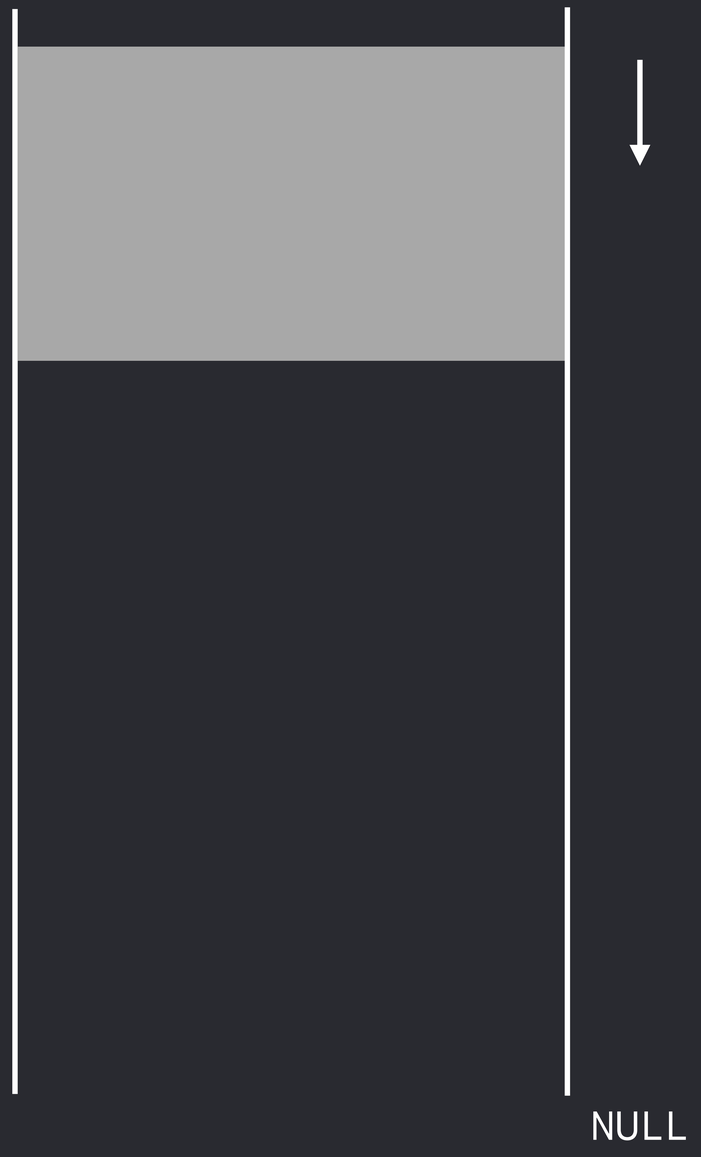
```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```




```c
int main(void) {
  dlog("Hello");
  ...
}
```

```c
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

dlog("Hello")




```c
int main(void) {
  dlog("Hello");
  ...
}
```

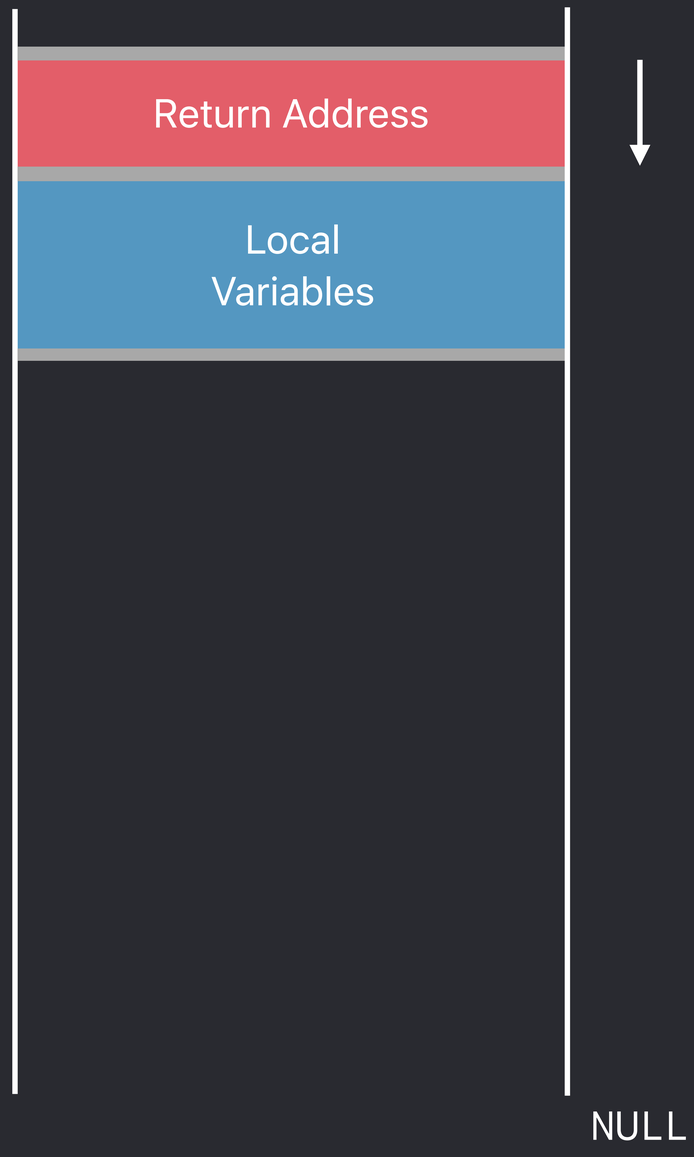
```
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

dlog("Hello")


```
int main(void) {
  dlog("Hello");
  ...
}
```

```
// The Stack: A Refresher

int dlog(const char *str) {
  ...
}
```

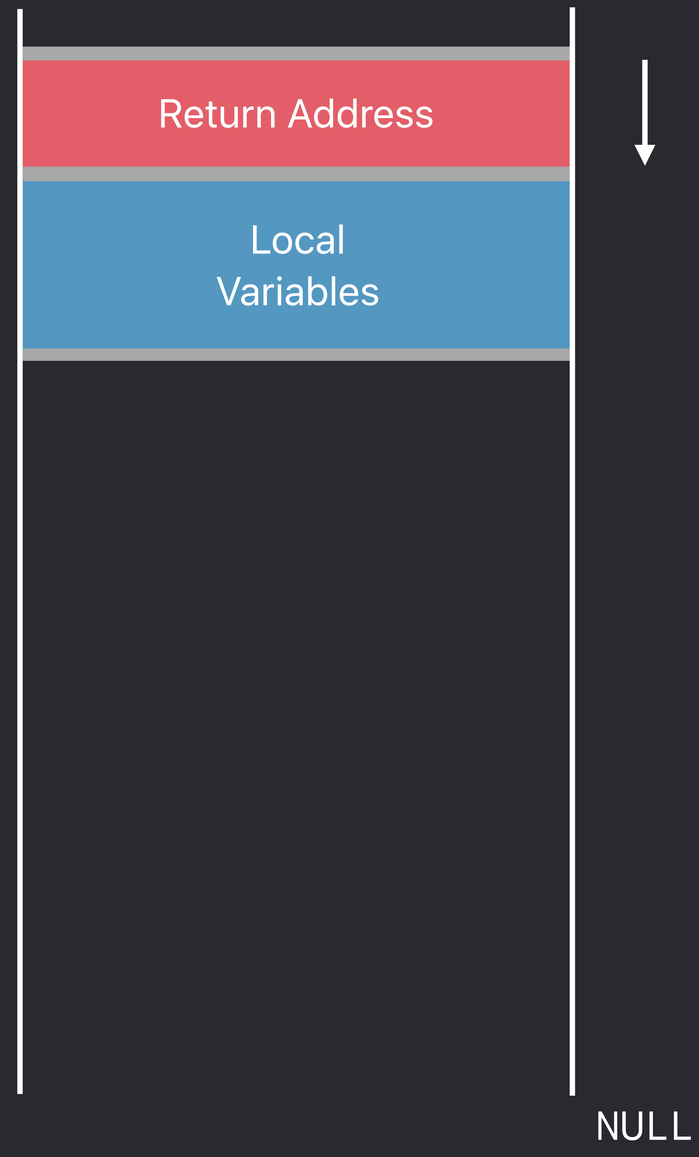
dlog("Hello")




```
int main(void) {
  dlog("Hello");
  ...
}
```

```
// The Stack: A Refresher

int dlog(const char *str) {
  char *logfile_path;
  ...
}
```

```
dlog("Hello")
```

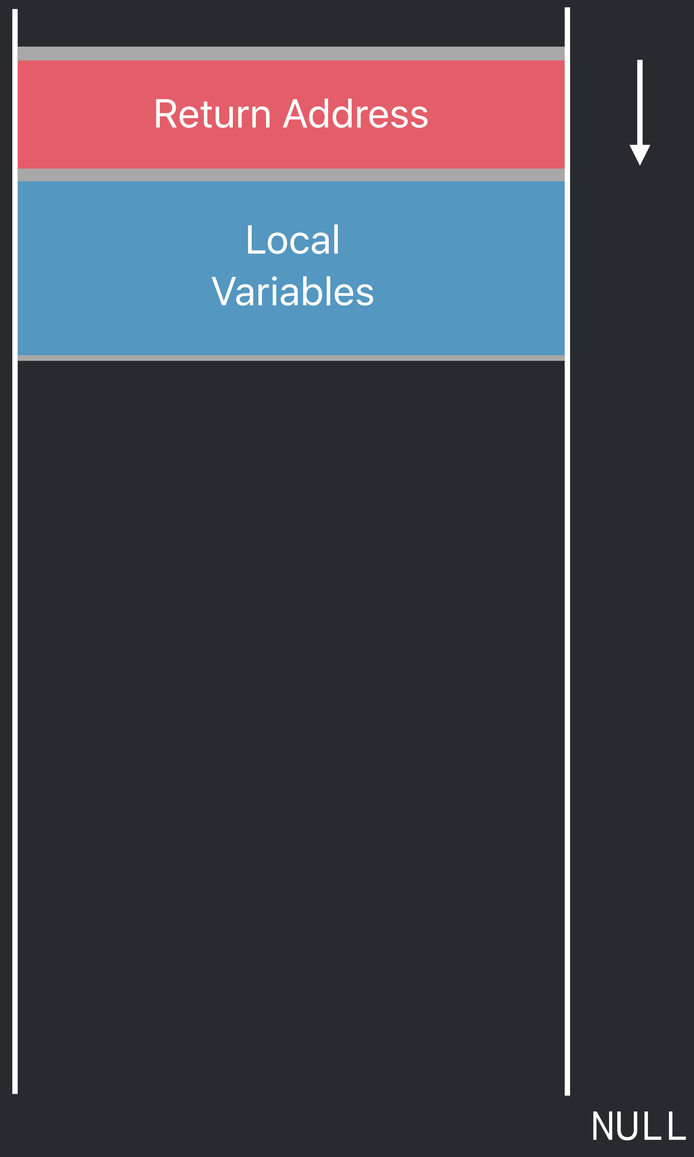


```
// The Stack: A Refresher

int dlog(const char *str) {
  char *logfile_path;
  ...
}
```

```
dlog("Hello")
```

```c
// The Stack: A Refresher

int dlog(const char *str) {
  char *logfile_path;
  logfile_path = dlogfile();
  ...
}

const char *dlogfile() {
  ...
}
```

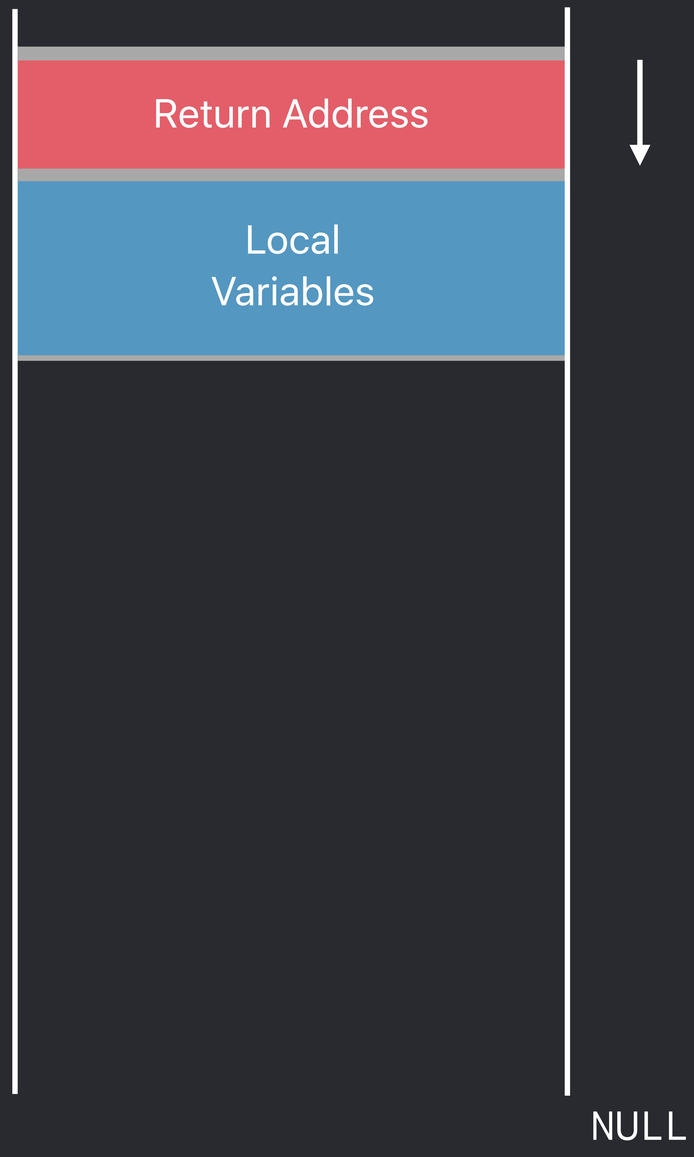
```
dlog("Hello")
```



```c
// The Stack: A Refresher

int dlog(const char *str) {
  char *logfile_path;
  logfile_path = dlogfile();
  ...
}

const char *dlogfile() {
  ...
}
```

dlog("Hello")

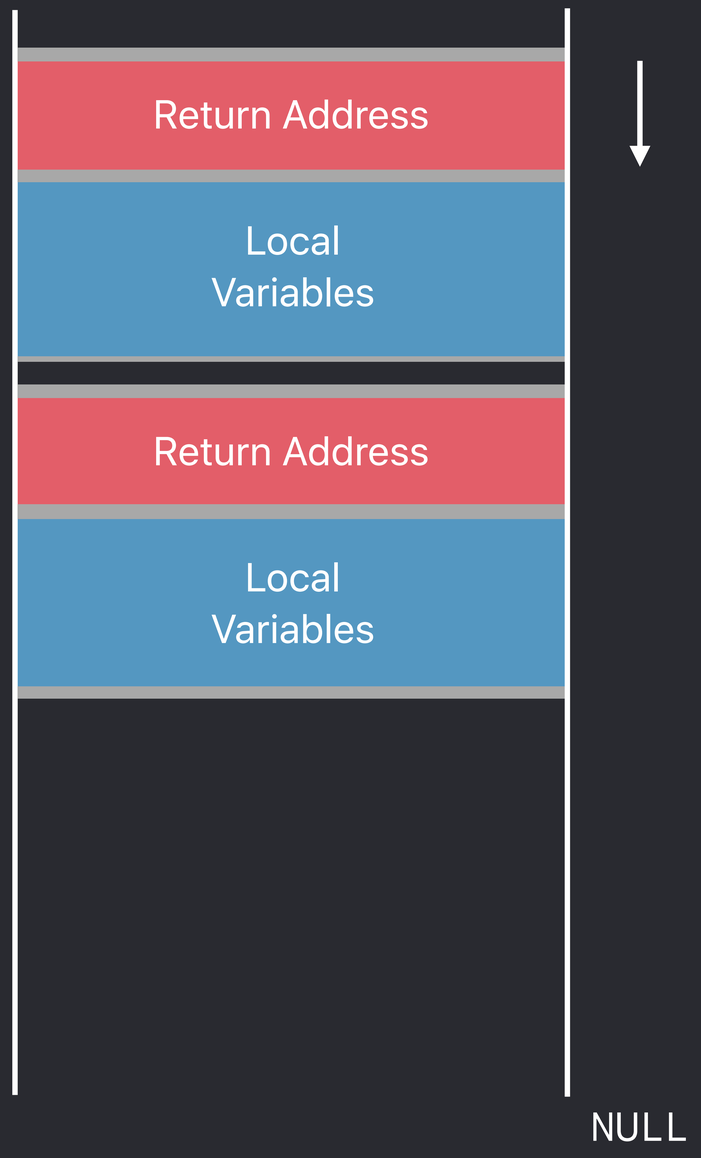
dlogfile()



```c
// The Stack: A Refresher

int dlog(const char *str) {
  char *logfile_path;
  logfile_path = dlogfile();
  ...
}

const char *dlogfile() {
  ...
}
```

dlog("Hello")

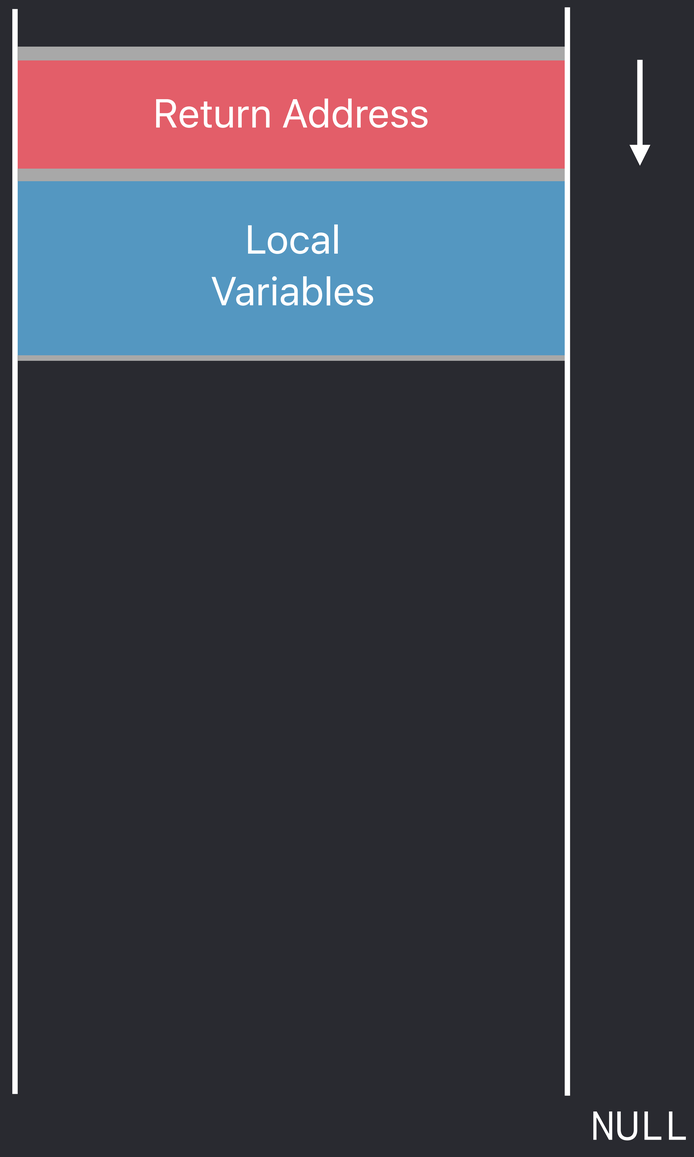
dlogfile()



```c
// The Stack: A Refresher

int dlog(const char *str) {
  char *logfile_path;
  logfile_path = dlogfile();
  ...
}

const char *dlogfile() {
  ...
}
```

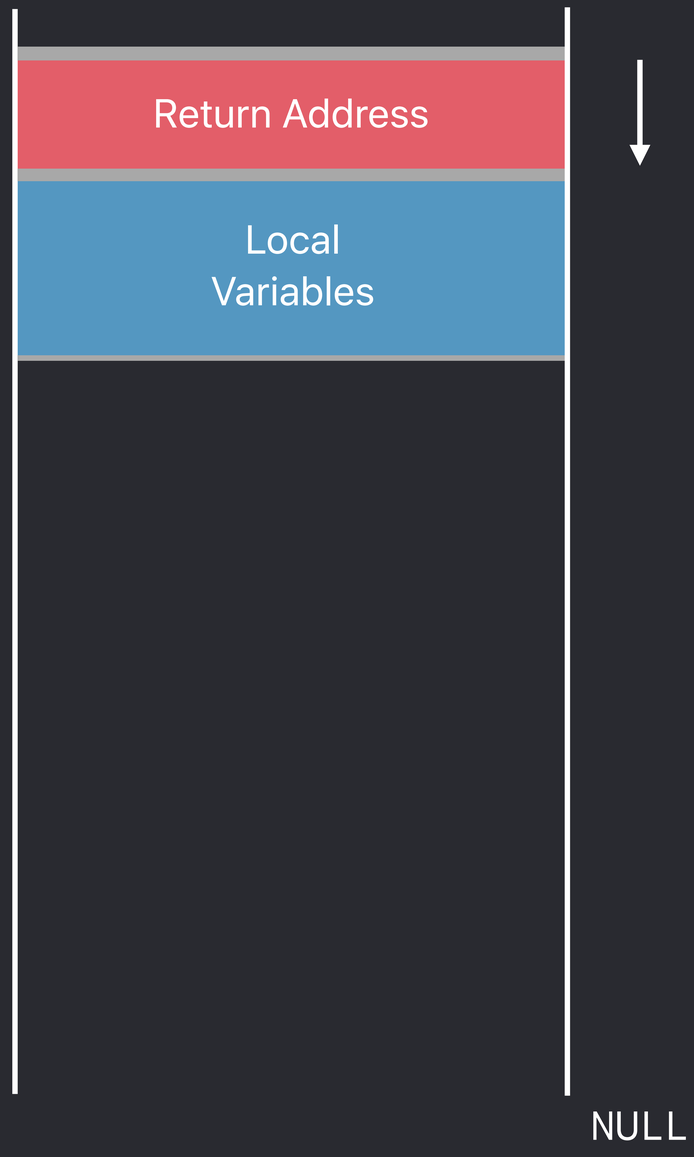
dlog("Hello")

dlogfile()

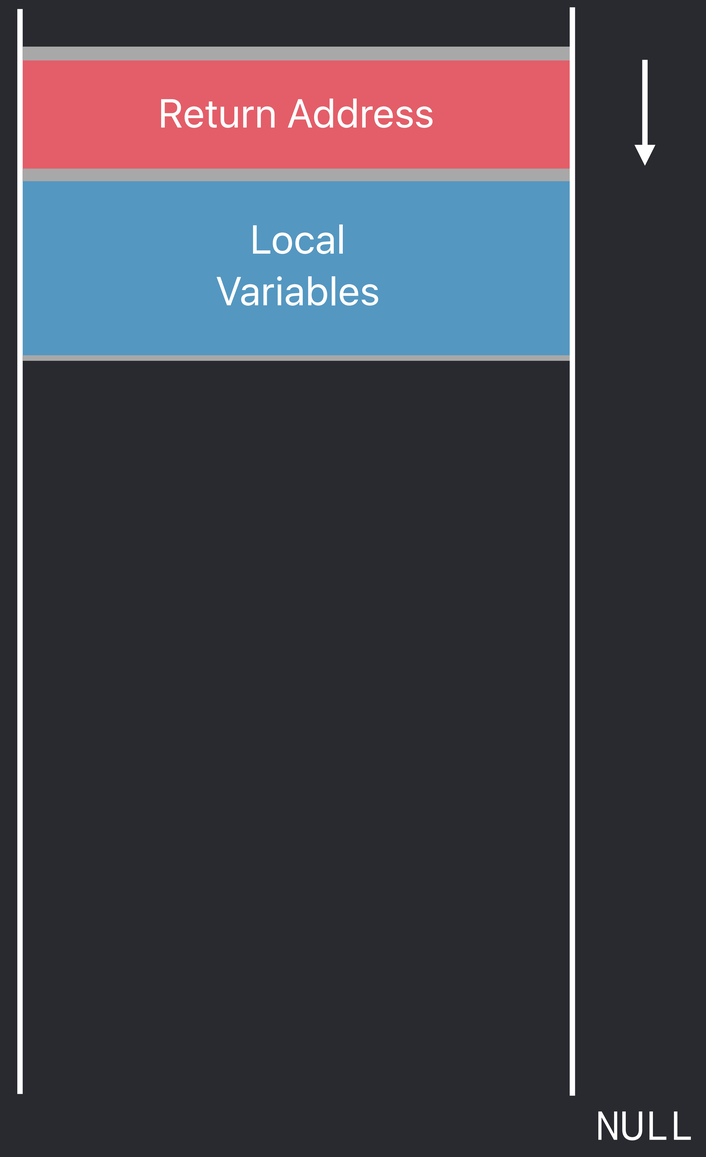


```c
// The Stack: A Refresher

int dlog(const char *str) {
  char *logfile_path;
  logfile_path = dlogfile();
  ...
}

const char *dlogfile() {
  ...
}
```

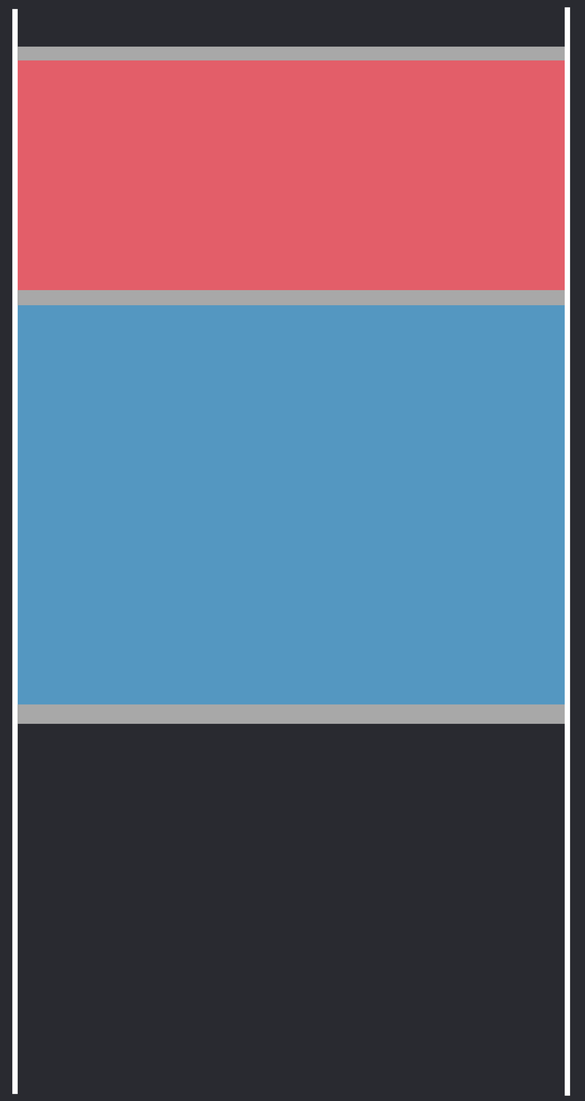
```
dlog("Hello")
```



```
dlog("Hello")
```

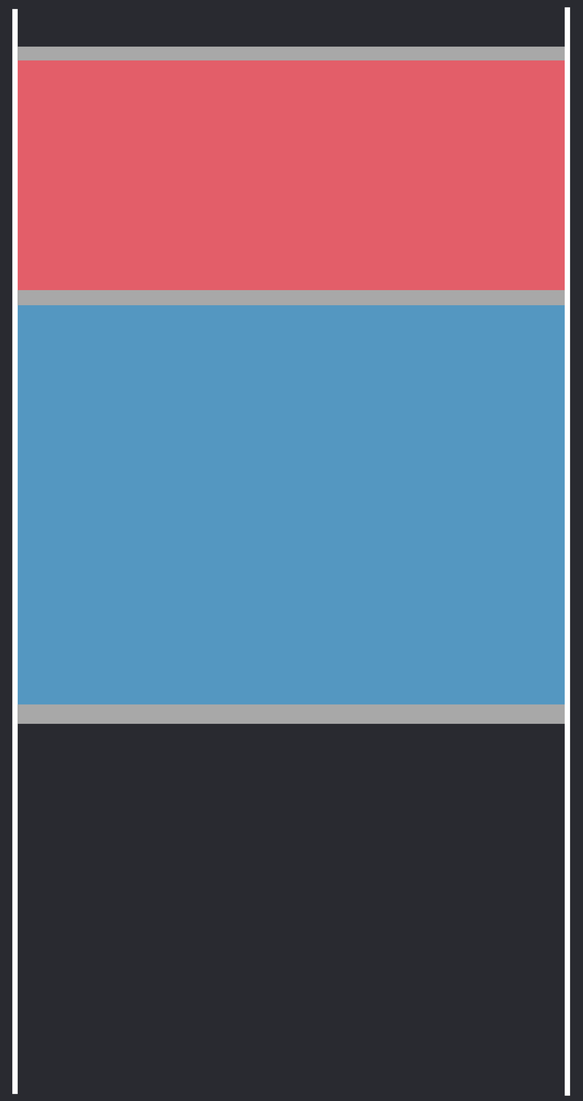


```
dlog("Hello")
```



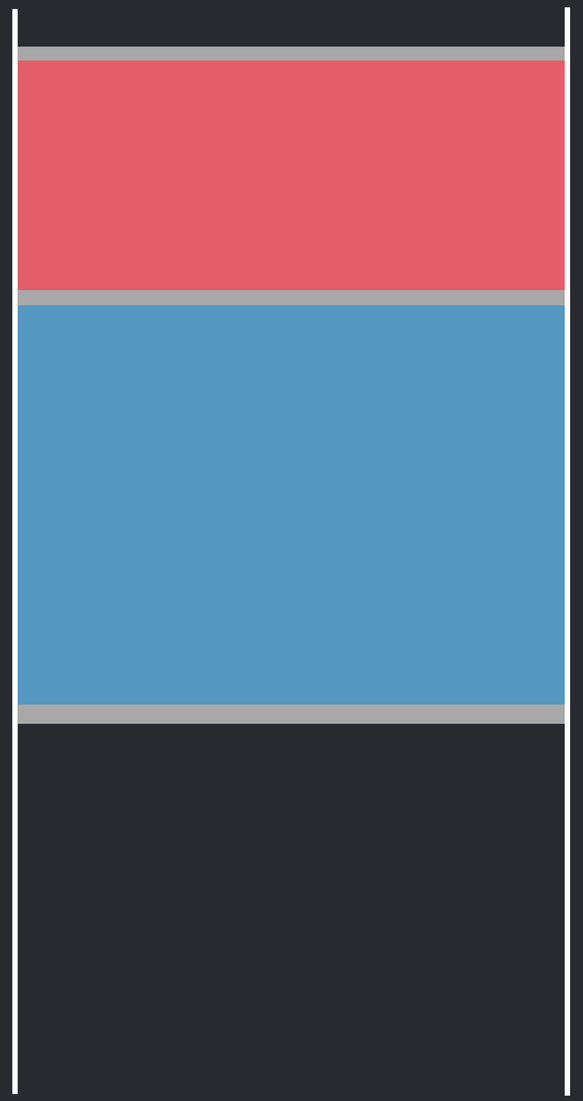
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

dlog("Hello")



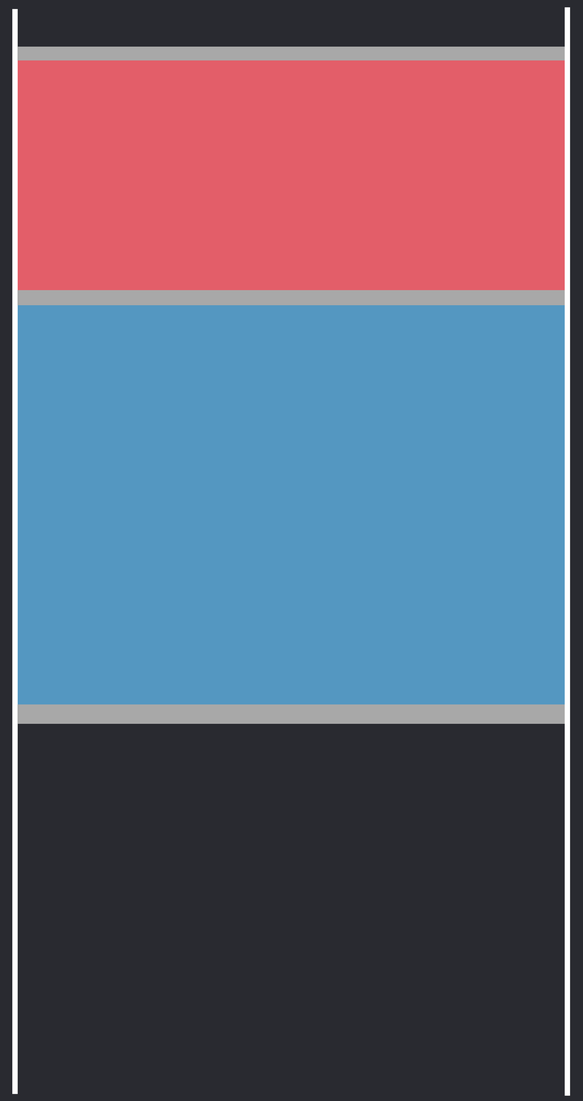
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

dlog("Hello")



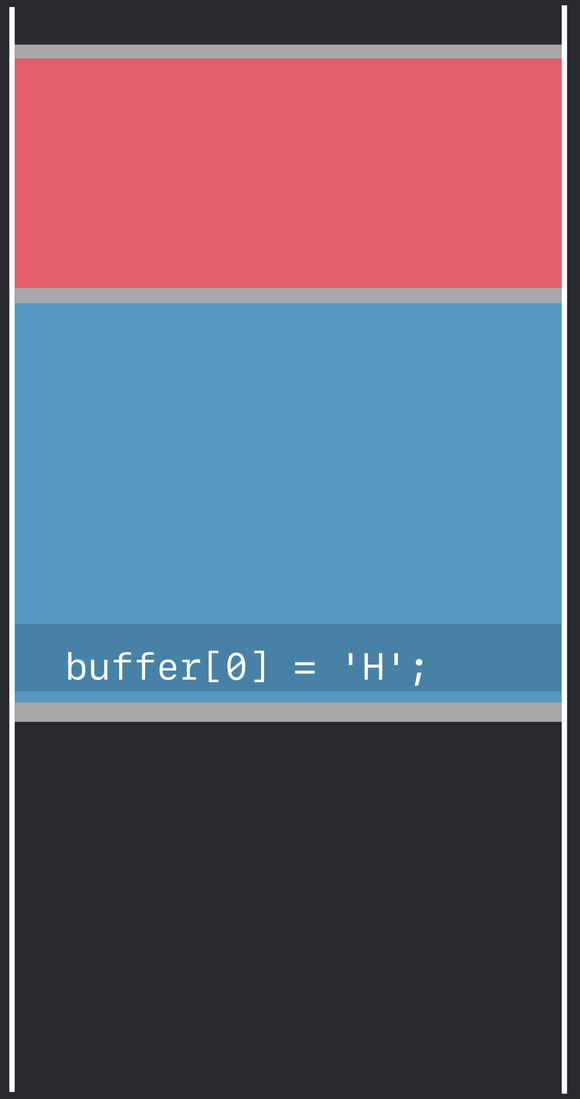
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

dlog("Hello")



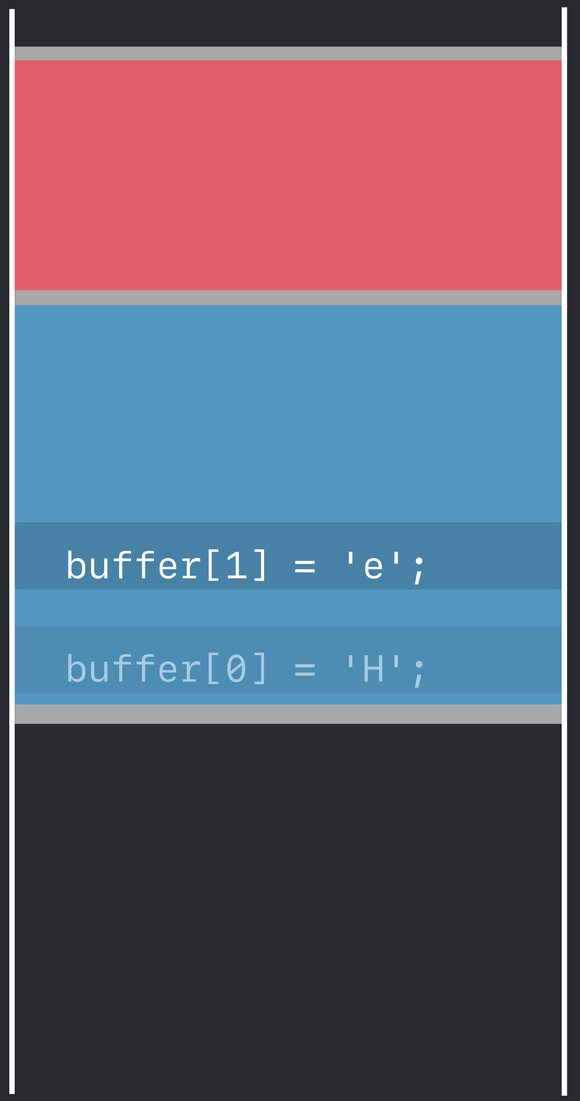
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



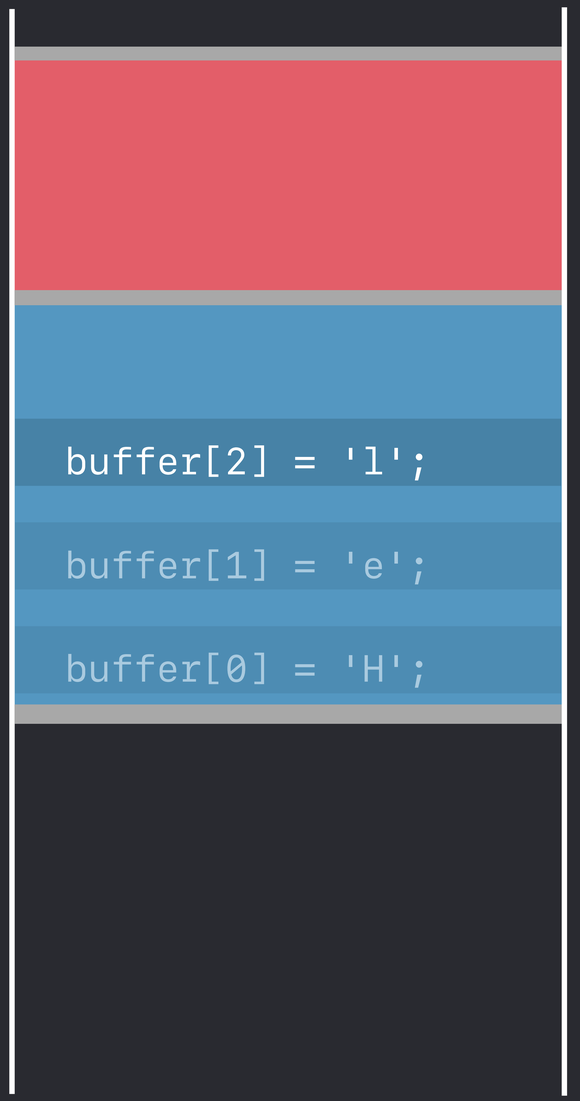
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



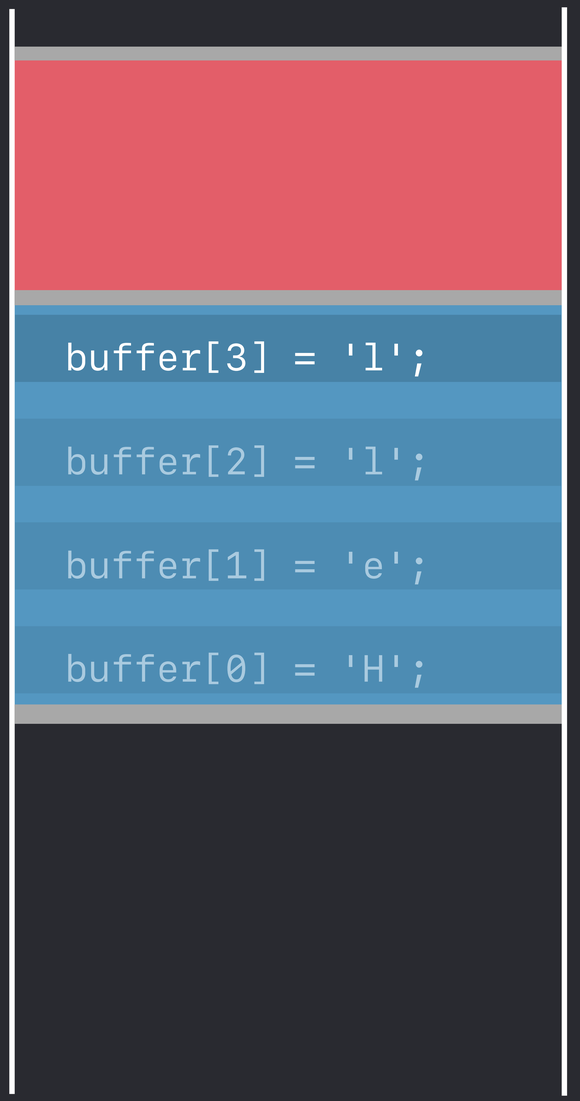
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



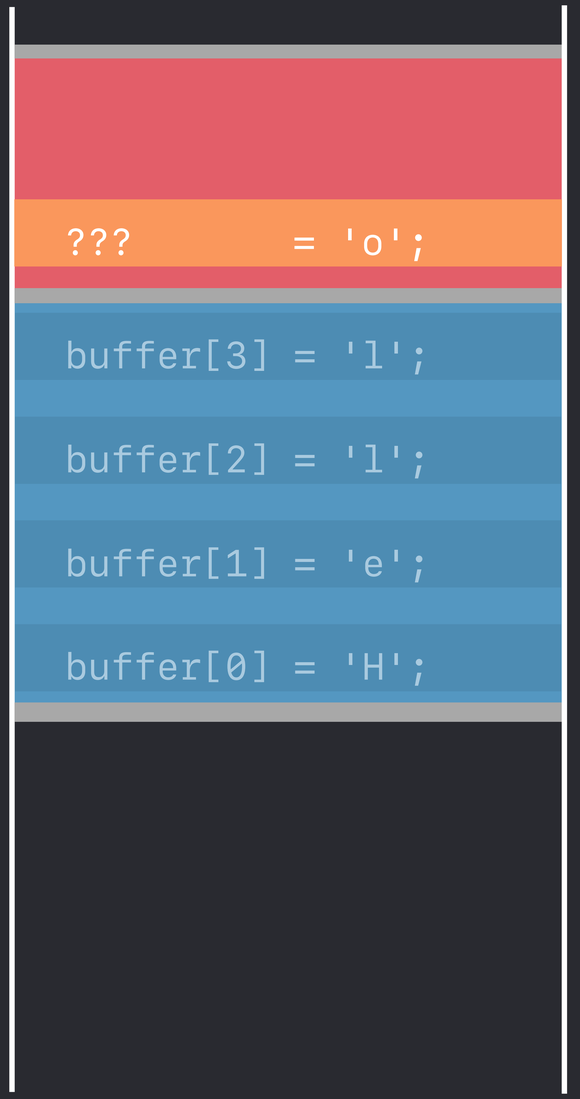
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



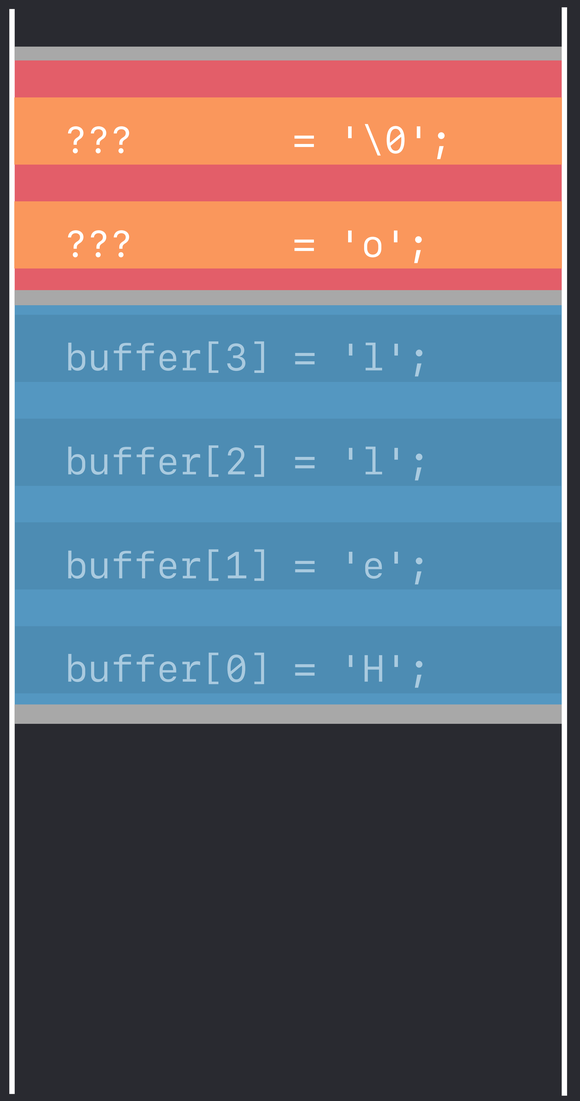
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



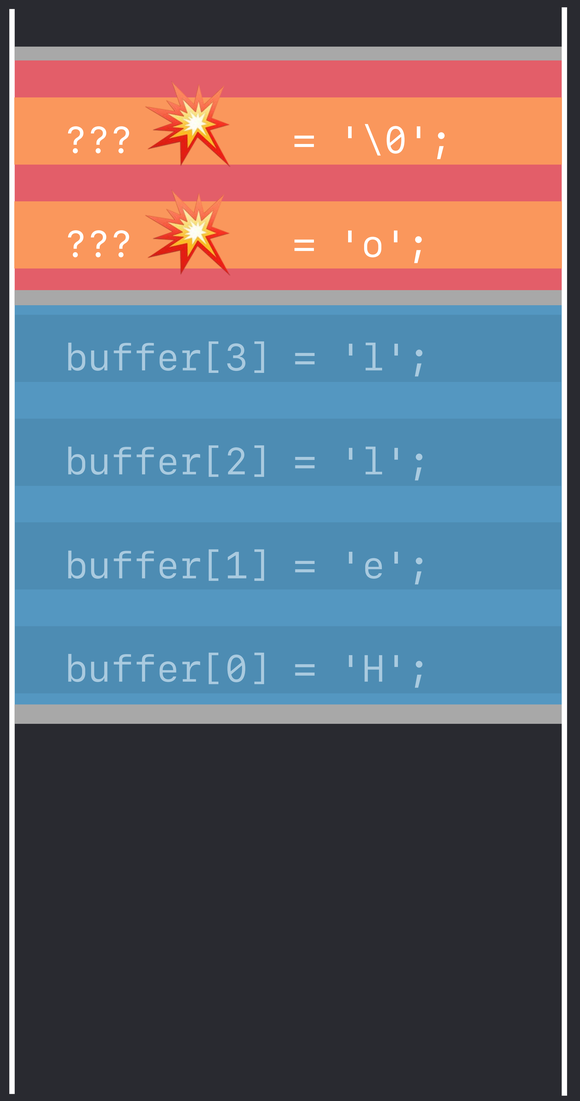
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



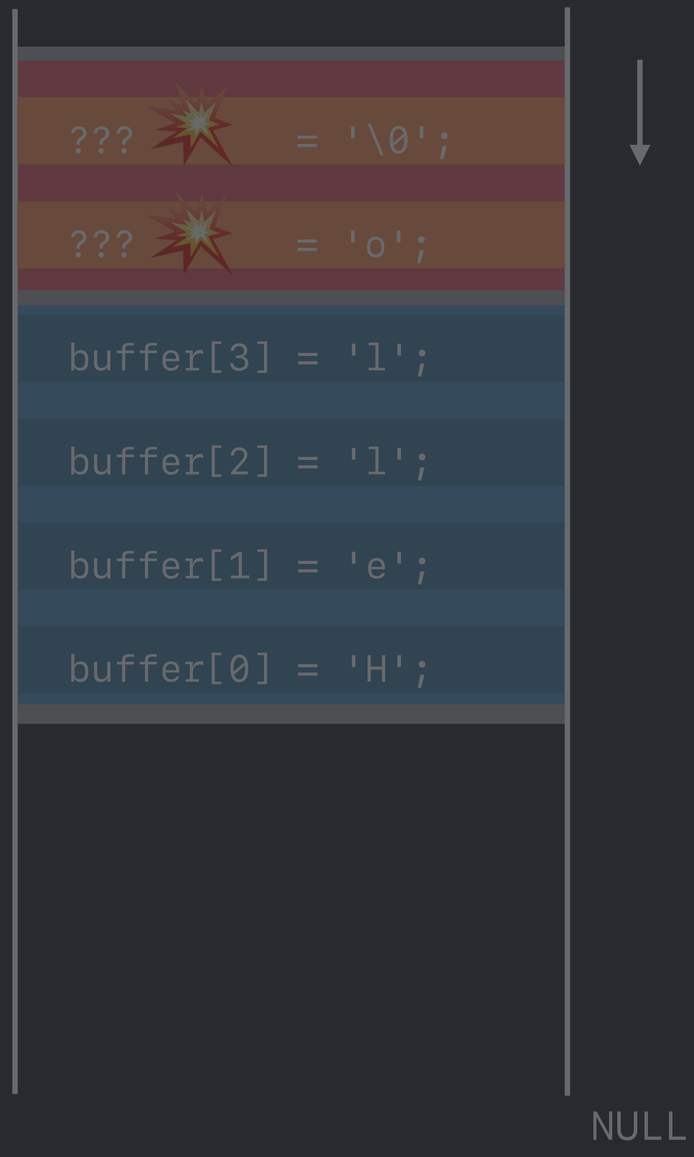
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



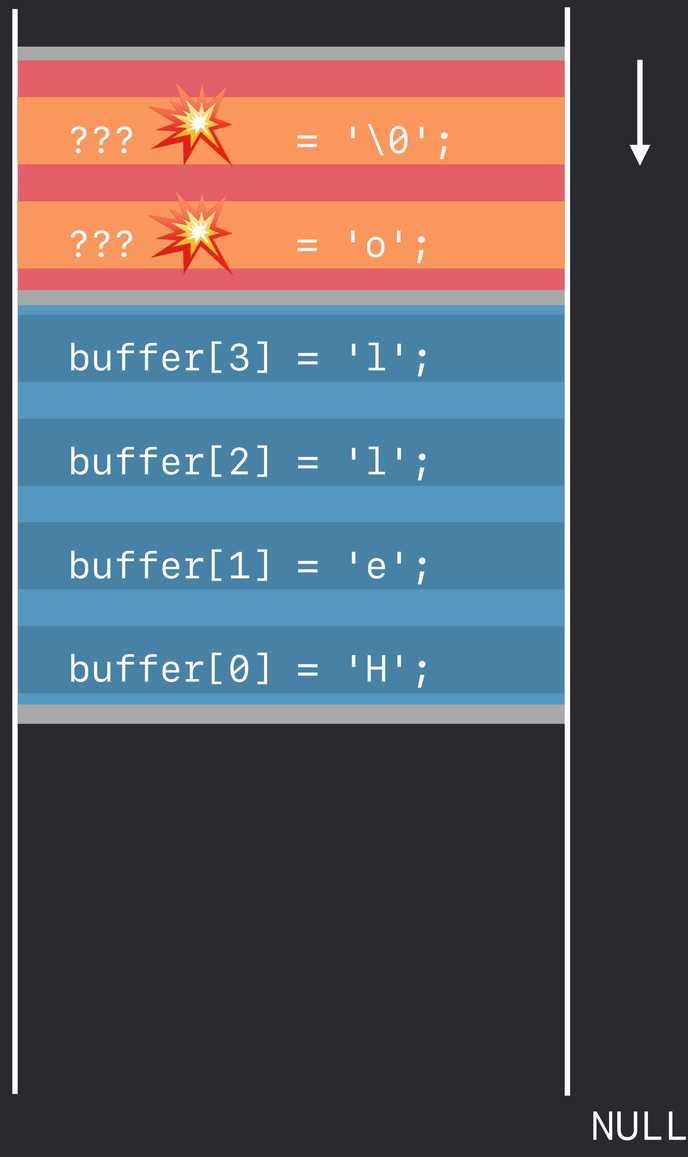
```
// Stack Buffer Overflows: The Fix

int dlog(const char *str) {
  char buffer[4];
  ...
  strncpy(buffer, str, sizeof(buffer));
  ...
}
```

dlog("Hello")



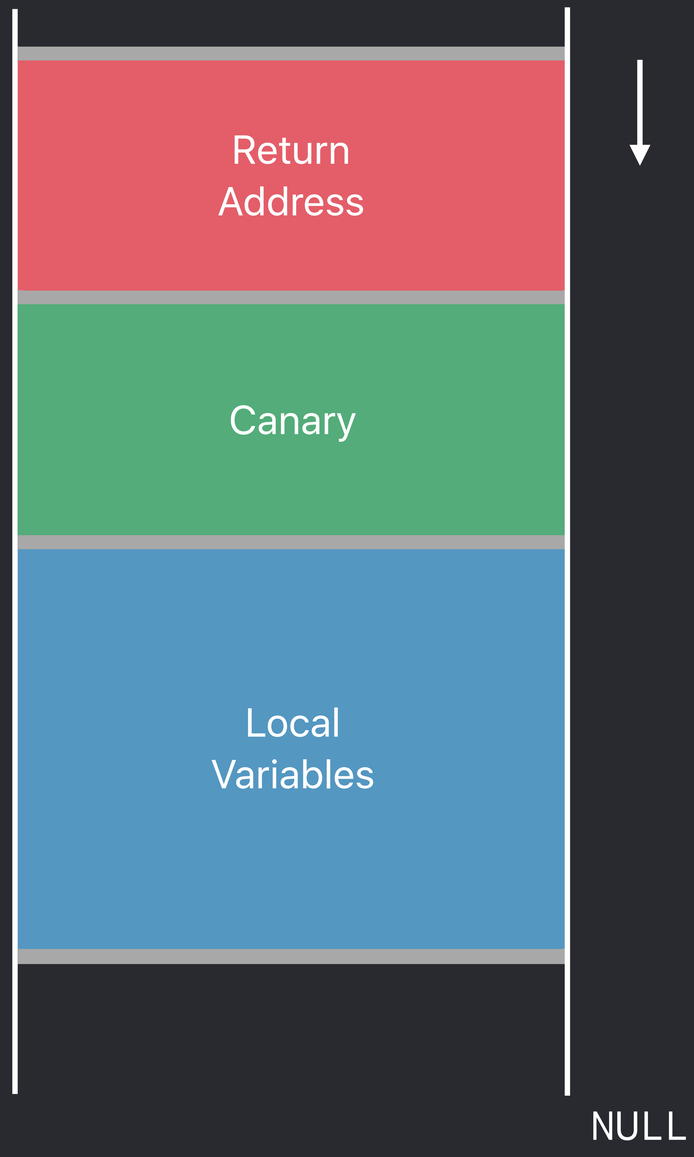
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



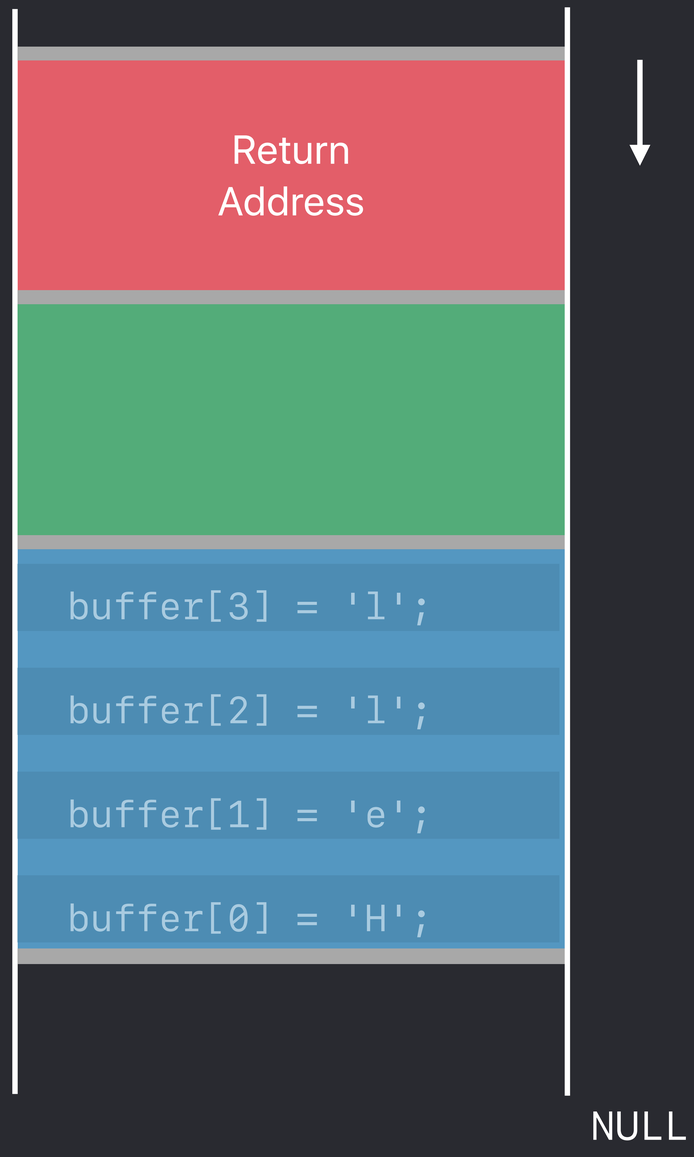
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
}
```

```
dlog("Hello")
```



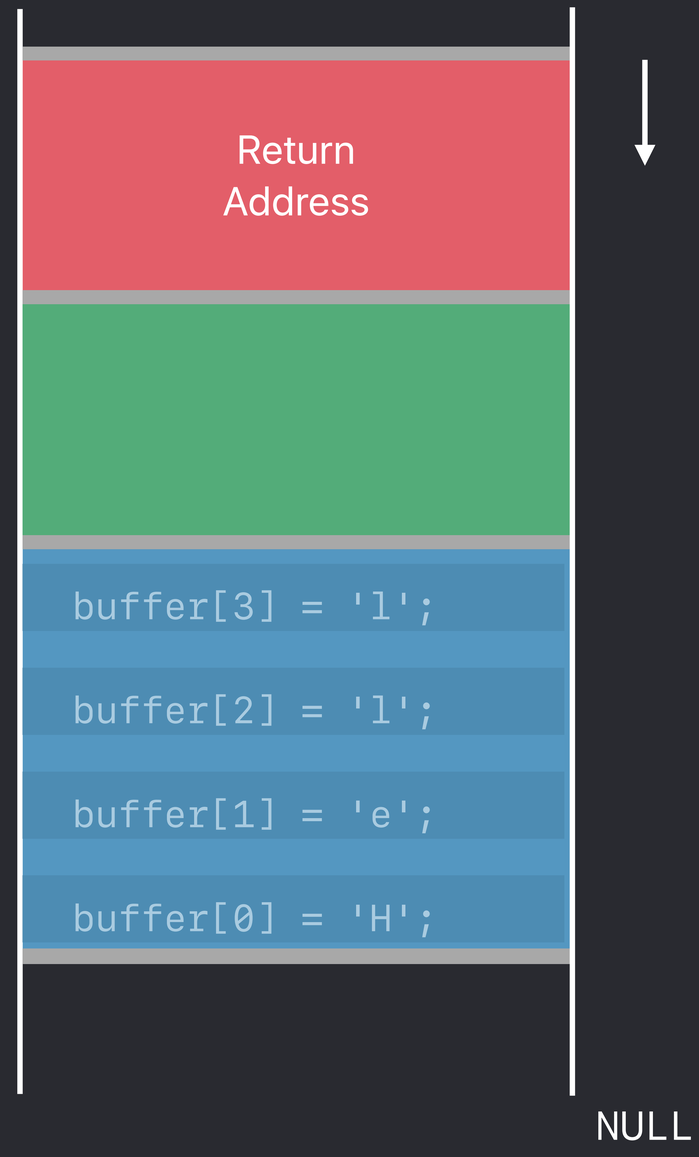
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
  // if (canary is invalid)
  //   abort();
  return ...;
}
```

dlog("Hello")



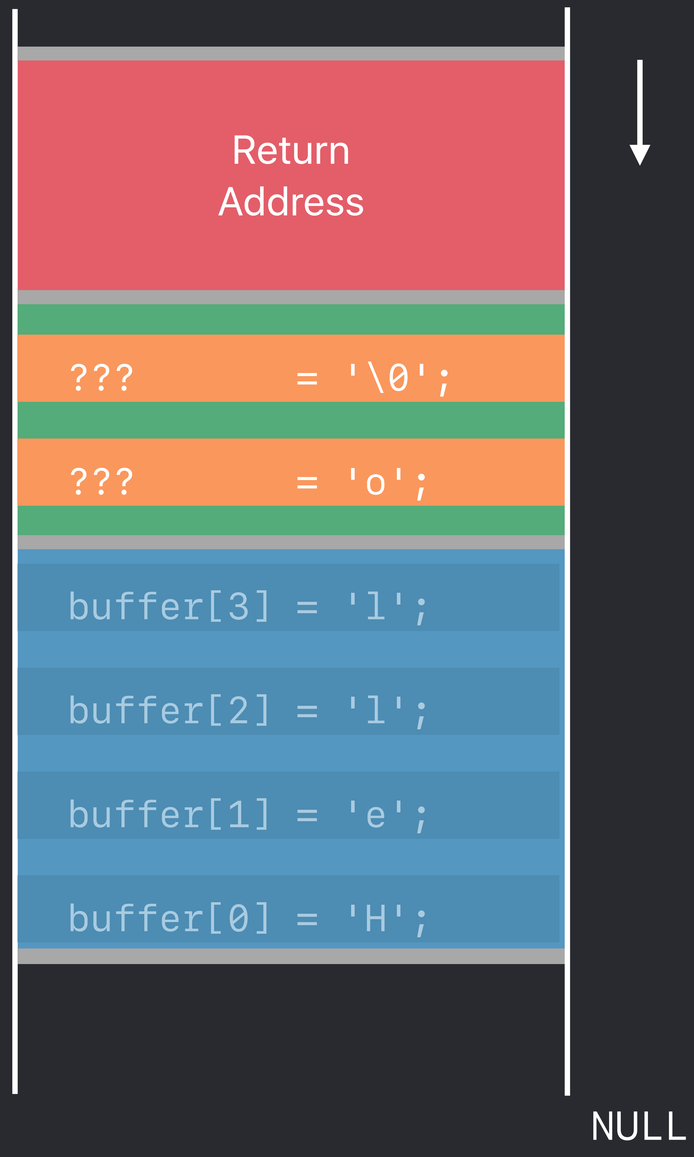
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
  // if (canary is invalid)
  //   abort();
  return ...;
}
```

dlog("Hello")



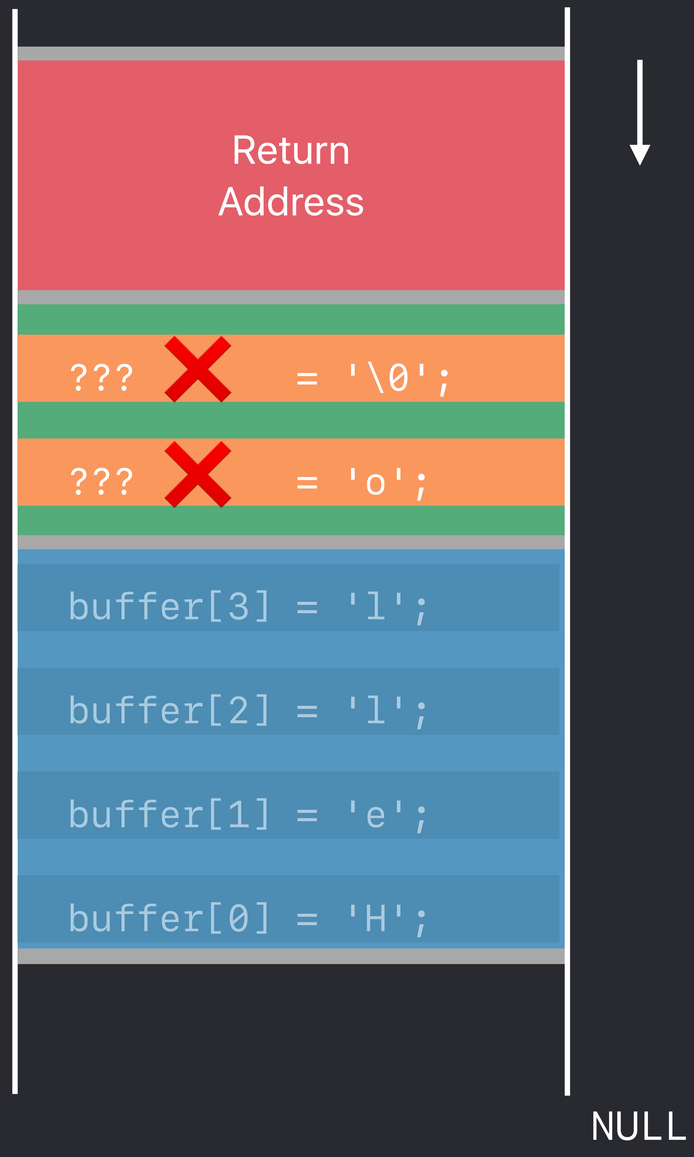
```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
  // if (canary is invalid)
  //   abort();
  return ...;
}
```

dlog("Hello")



```
// Stack Buffer Overflows

int dlog(const char *str) {
  char buffer[4];
  ...
  strcpy(buffer, str);
  ...
  // if (canary is invalid)
  //   abort();
  return ...;
}
```

dlog("Hello")



Stack Protector

Stack Protector

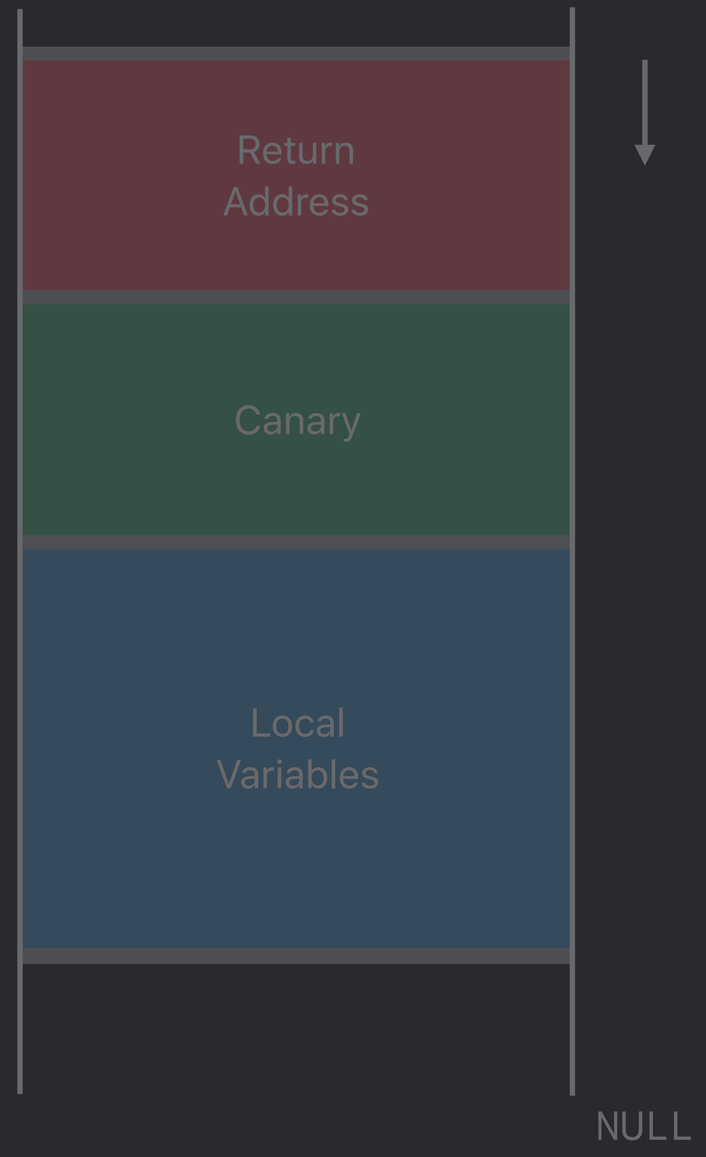
Detects stack buffer overflows

Stack Protector

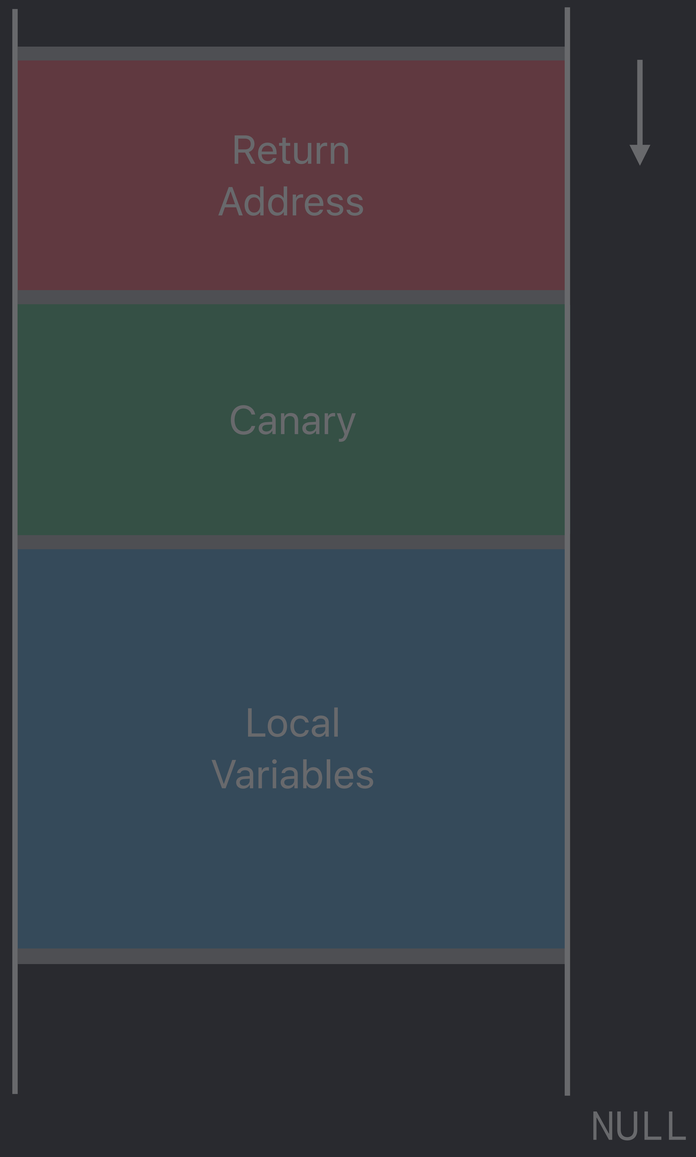
Detects stack buffer overflows

Already enabled by default

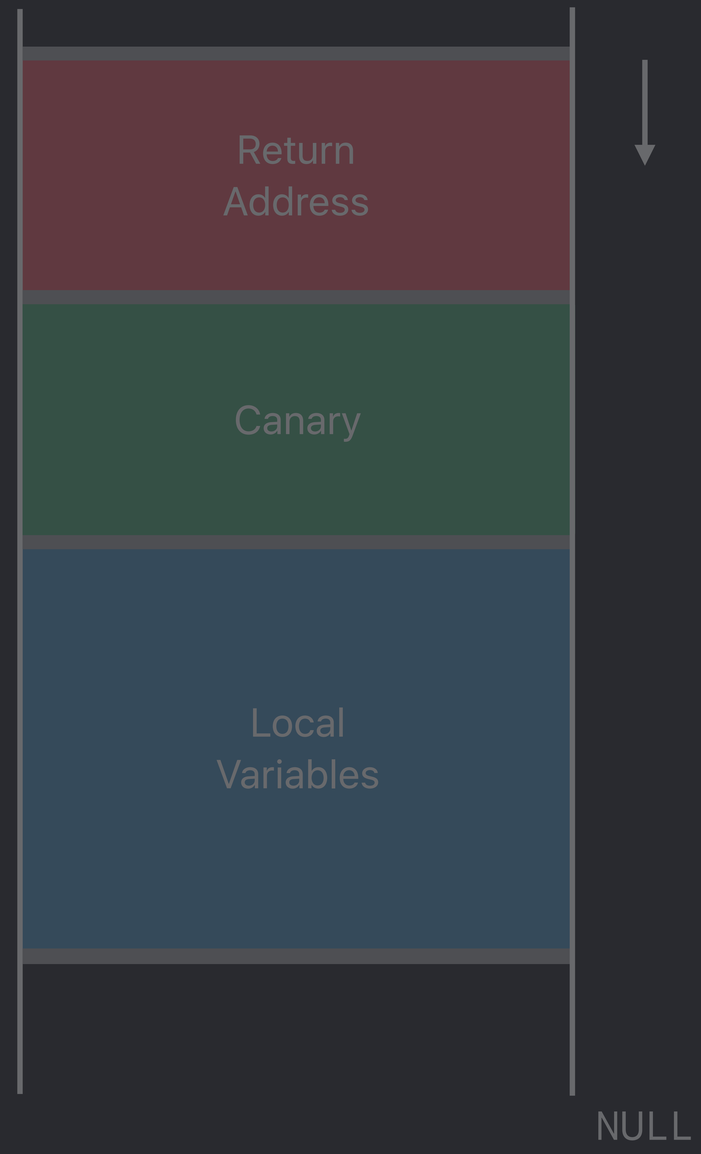
```
// Stack Allocation: Variable Length Arrays

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```



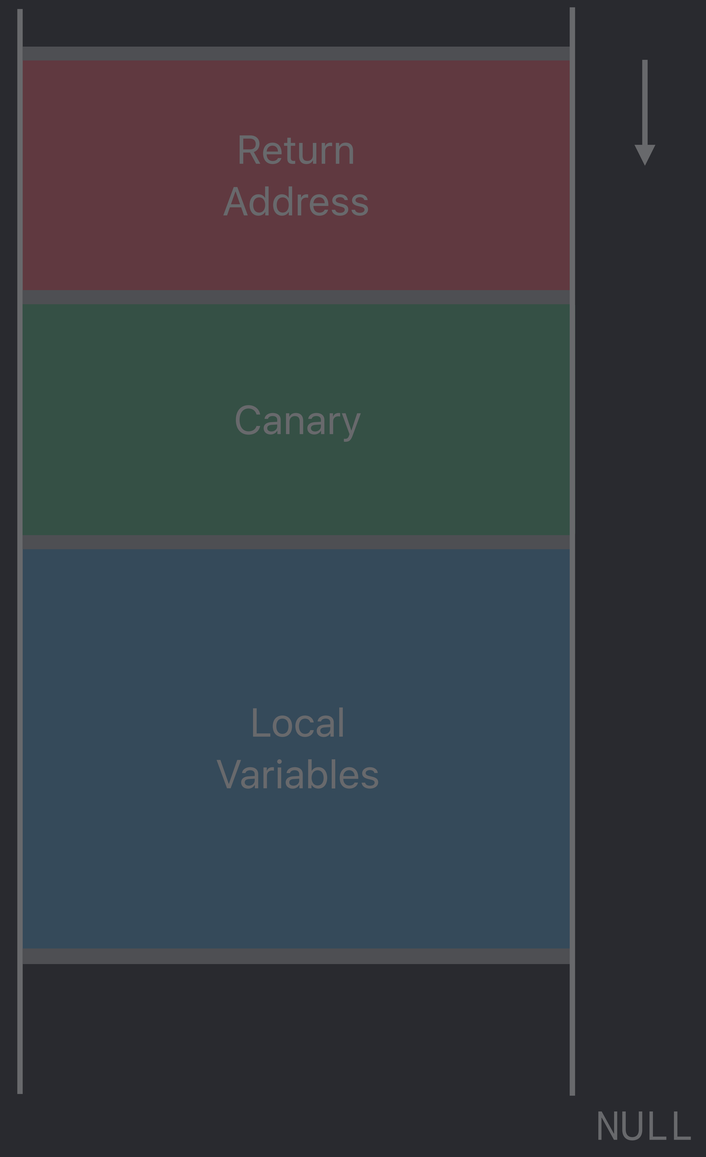
```c
// Stack Allocation: Variable Length Arrays

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```



```
// Stack Allocation: Variable Length Arrays

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

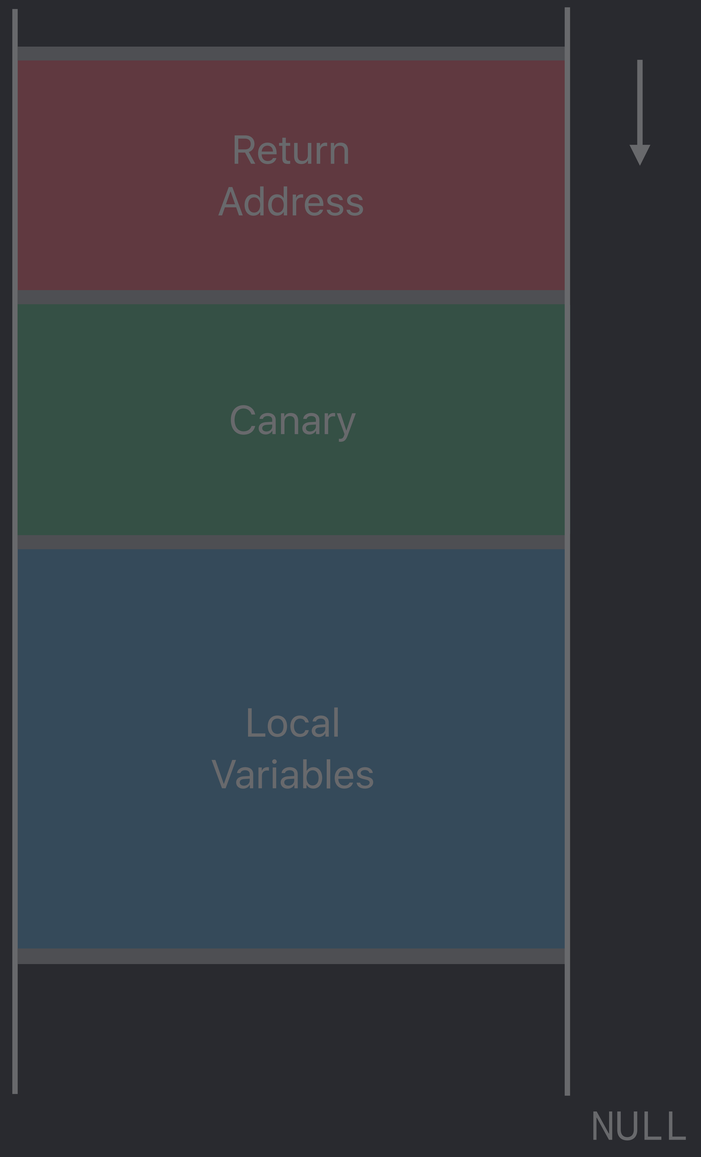


```
// Stack Allocation: Variable Length Arrays

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```



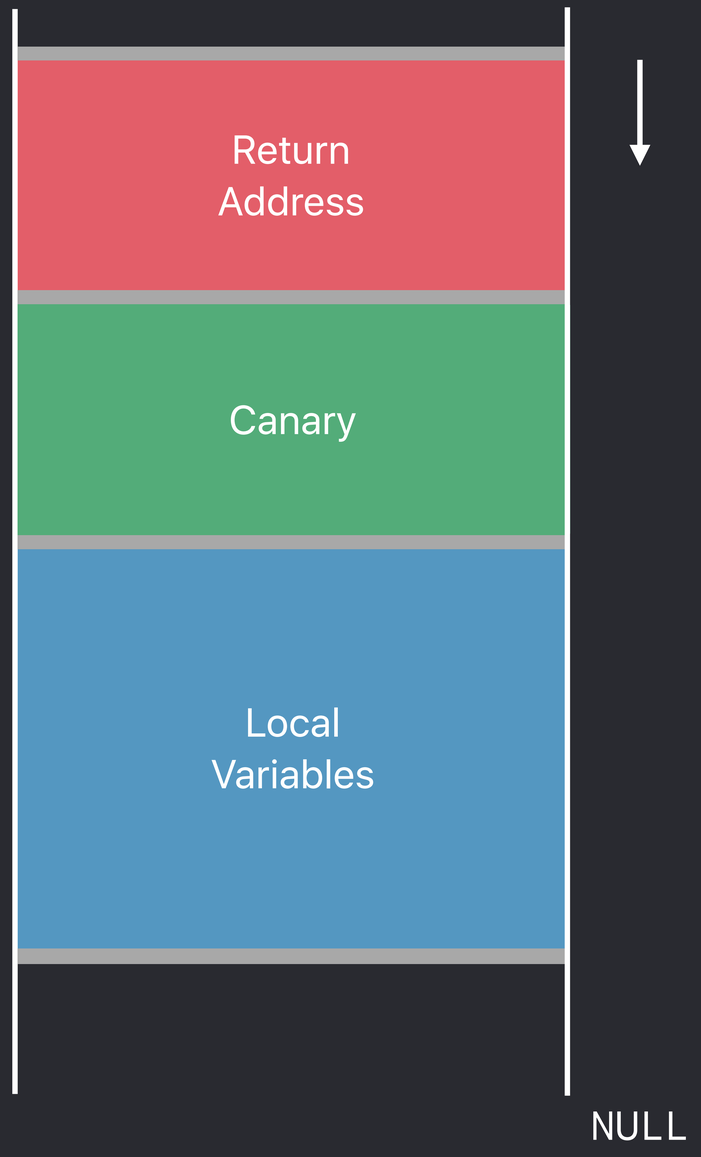
```
// Stack Allocation: Variable Length Arrays

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

```
dlog("Hello", 15000)
```



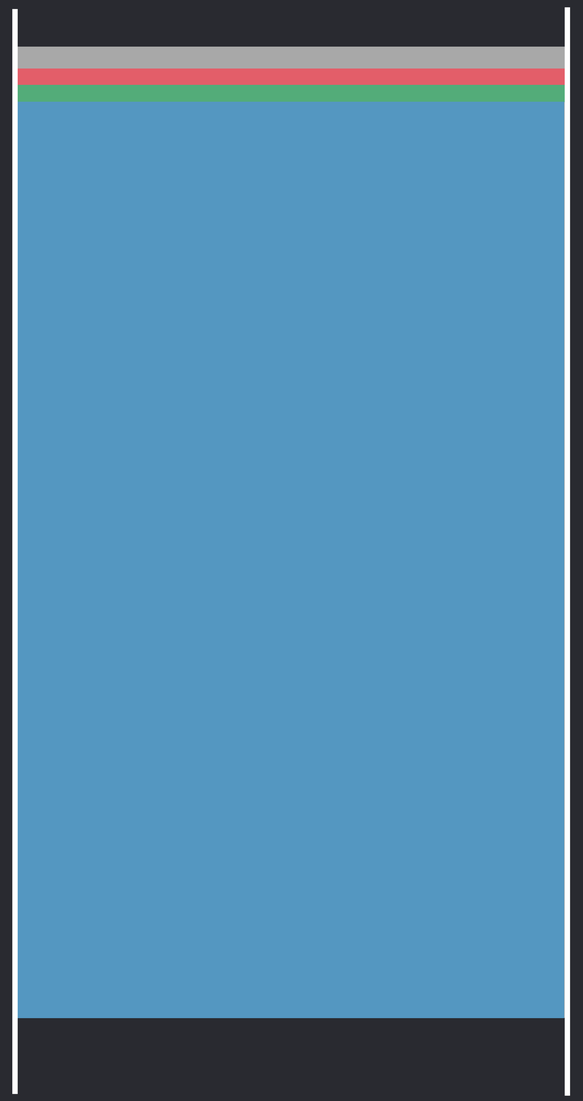
```
// Stack Allocation: Variable Length Arrays

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

```
dlog("Hello", 15000)
```



```c
// Stack Allocation: Variable Length Arrays

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

```c
dlog("Hello", 15000)
```

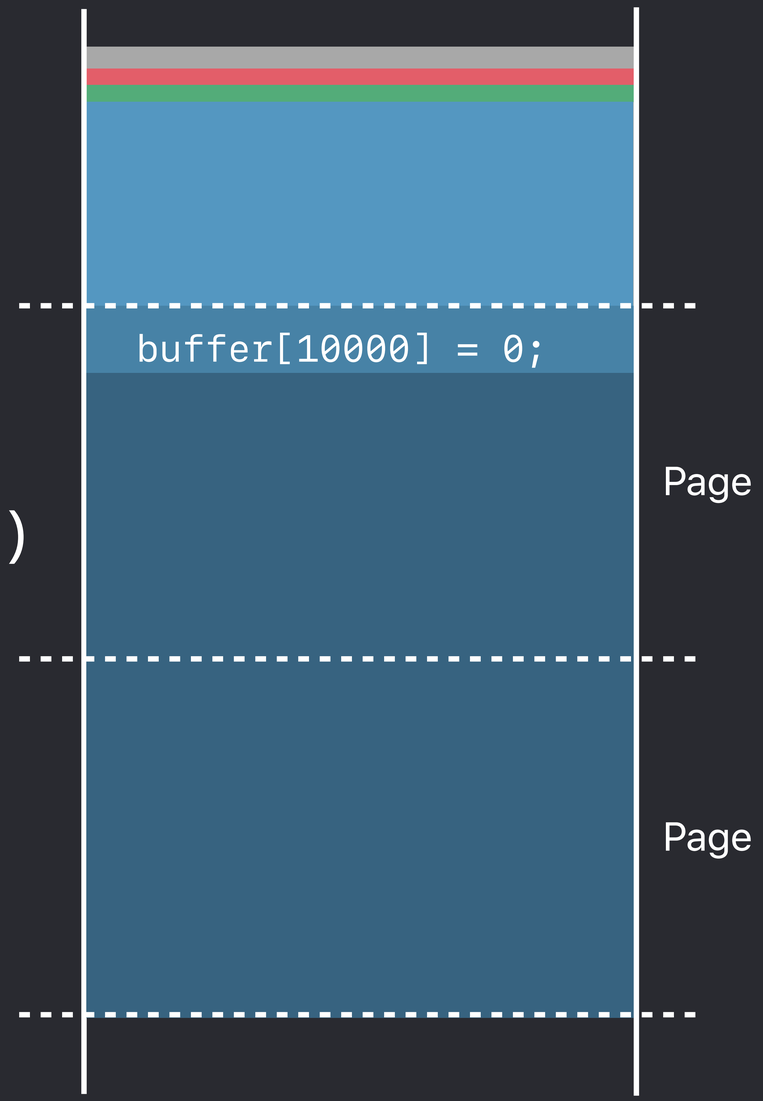


```
// Stack Allocation: Pages

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

```
dlog("Hello", 15000)
```

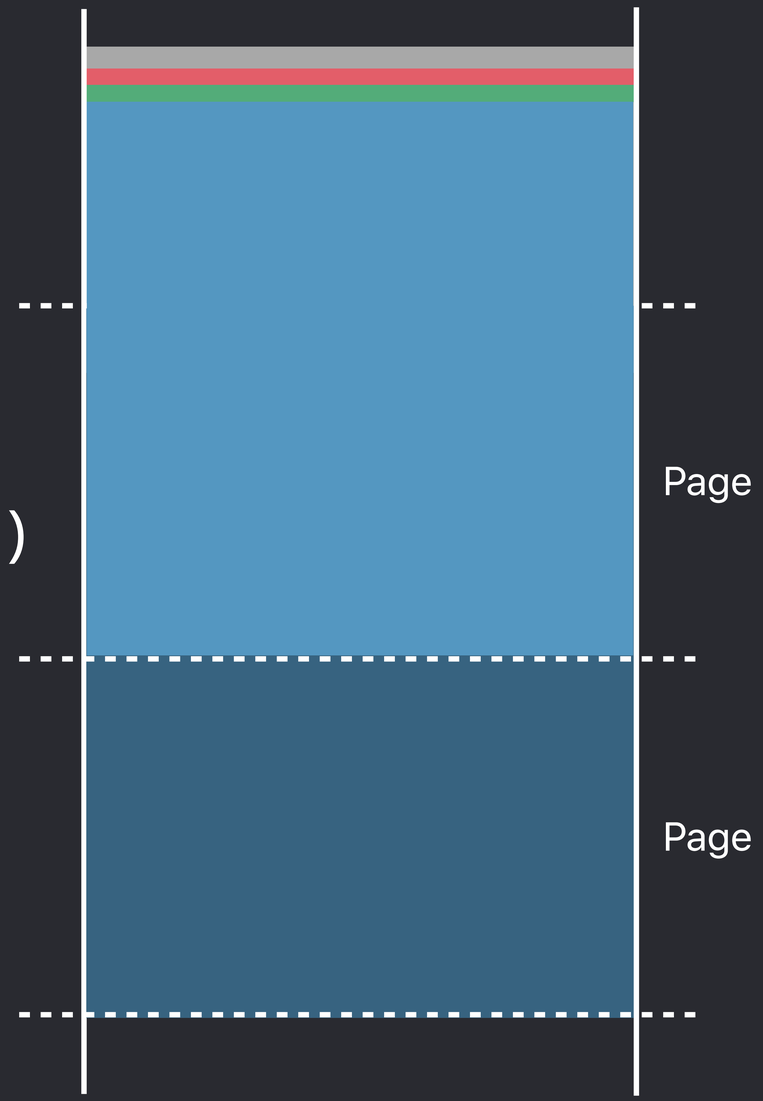
```
// Stack Allocation: Pages

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

```
dlog("Hello", 15000)
```



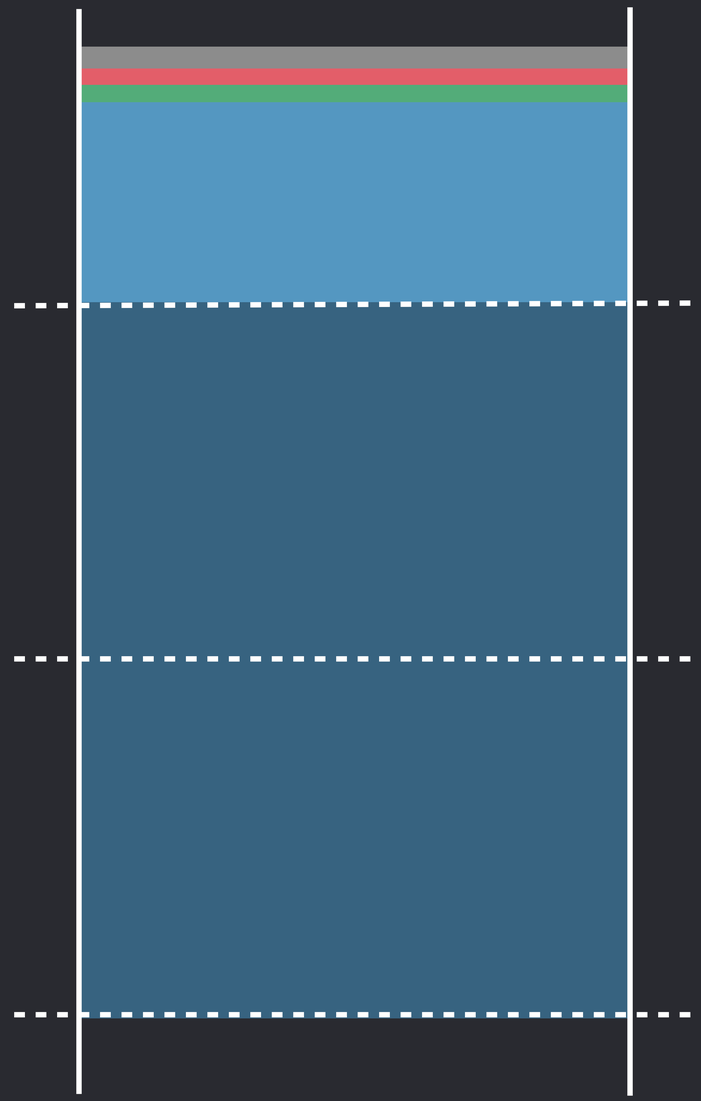
```c
// Stack Allocation: Pages

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

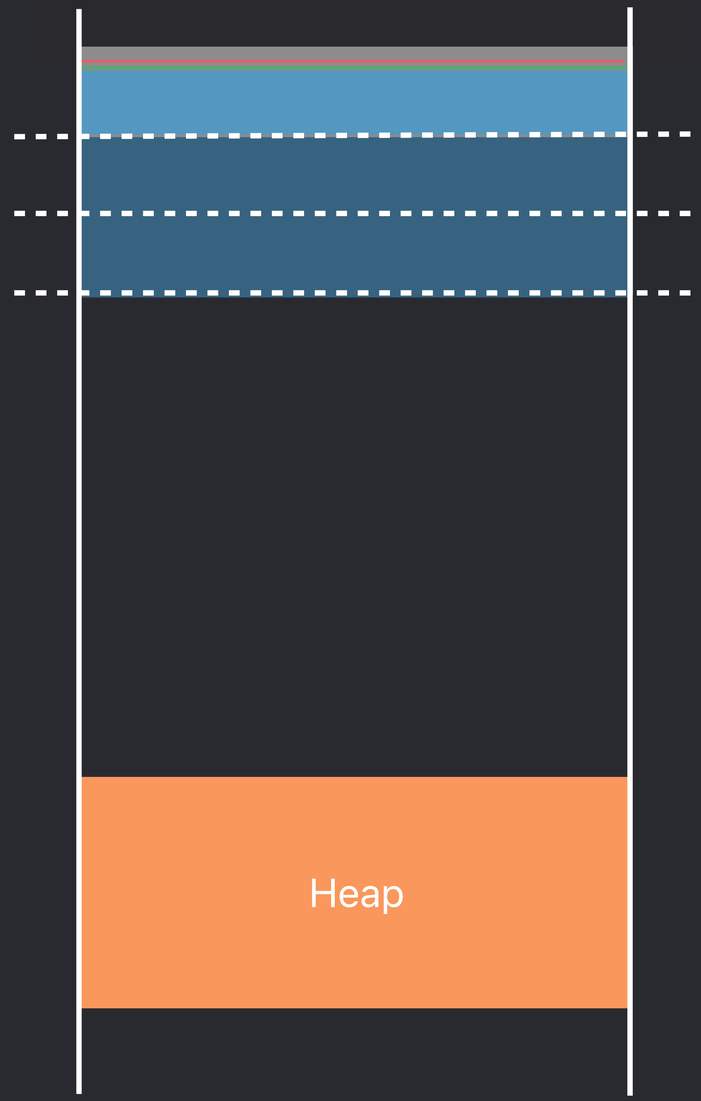
```c
dlog("Hello", 15000)
```

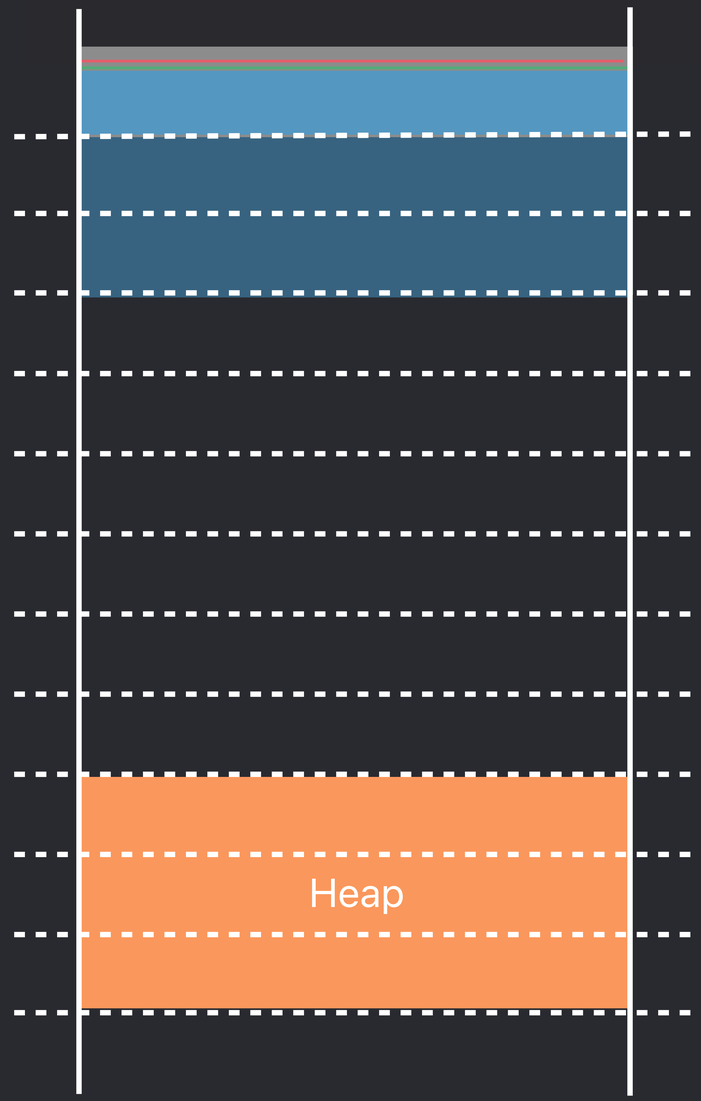


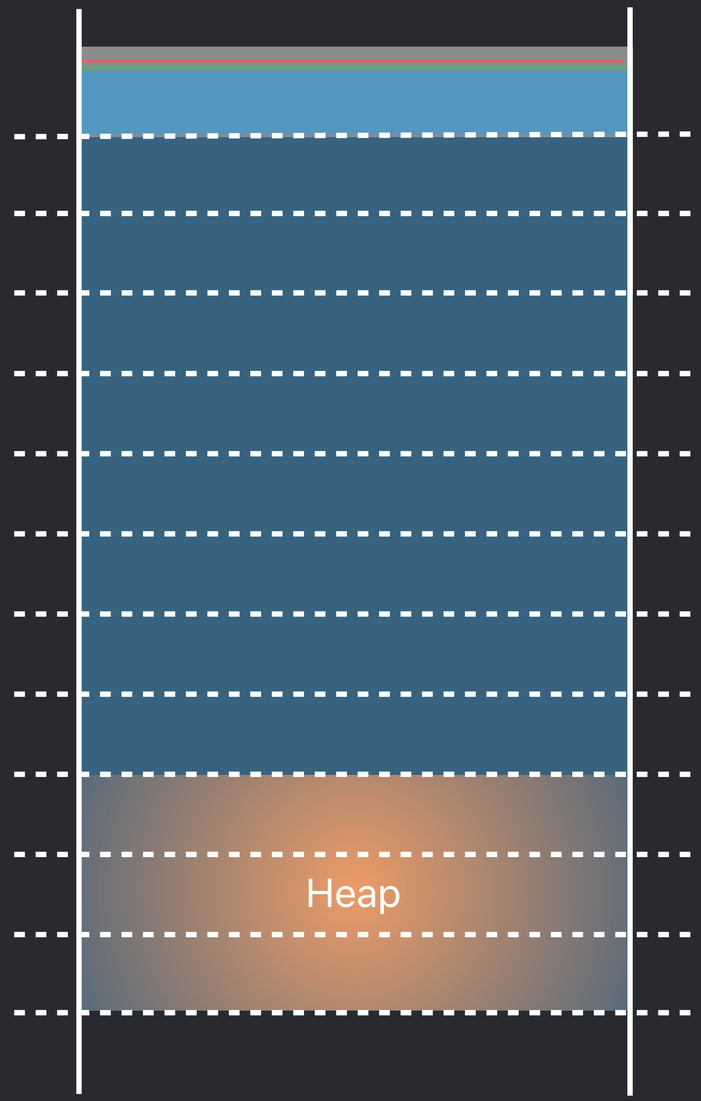
```c
// "Stack Clash"

int dlog(const char *str, size_t len) {
  char buffer[len];
  ...
}
```

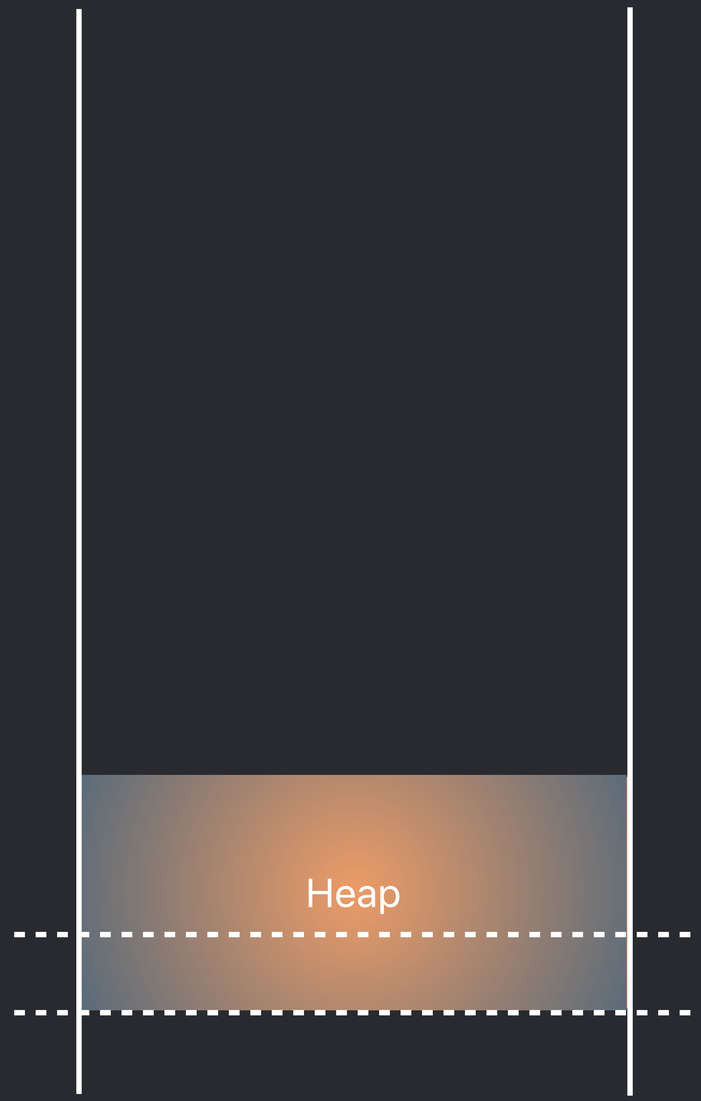
```c
dlog("Hello", /*huge!*/)
```

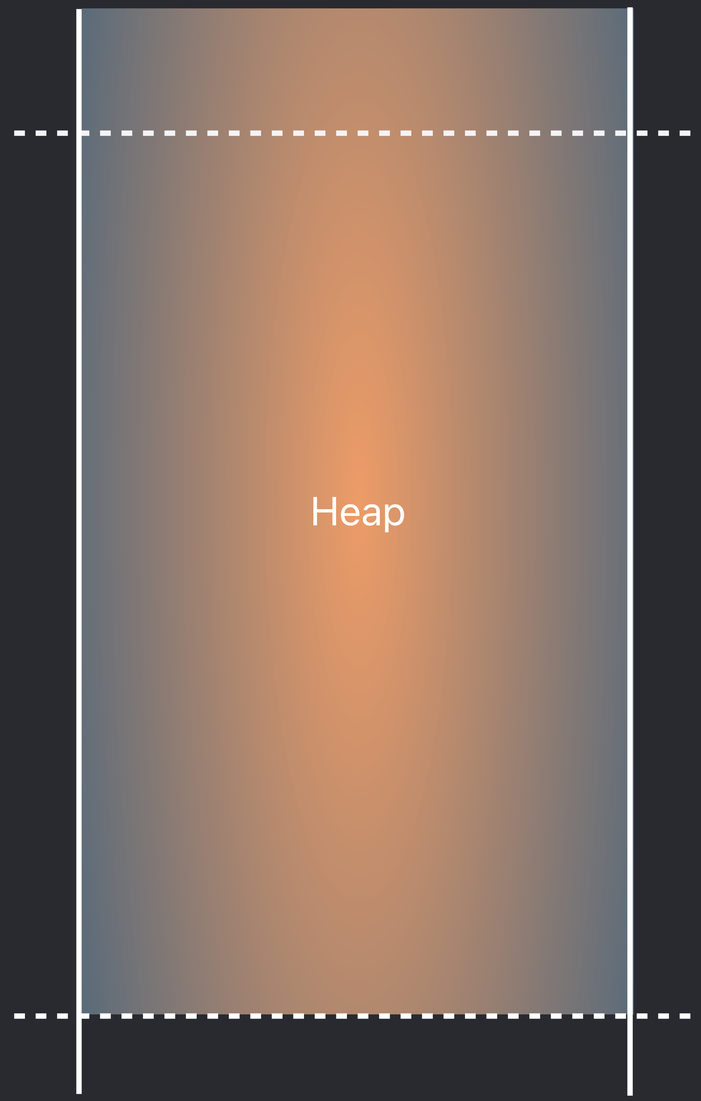


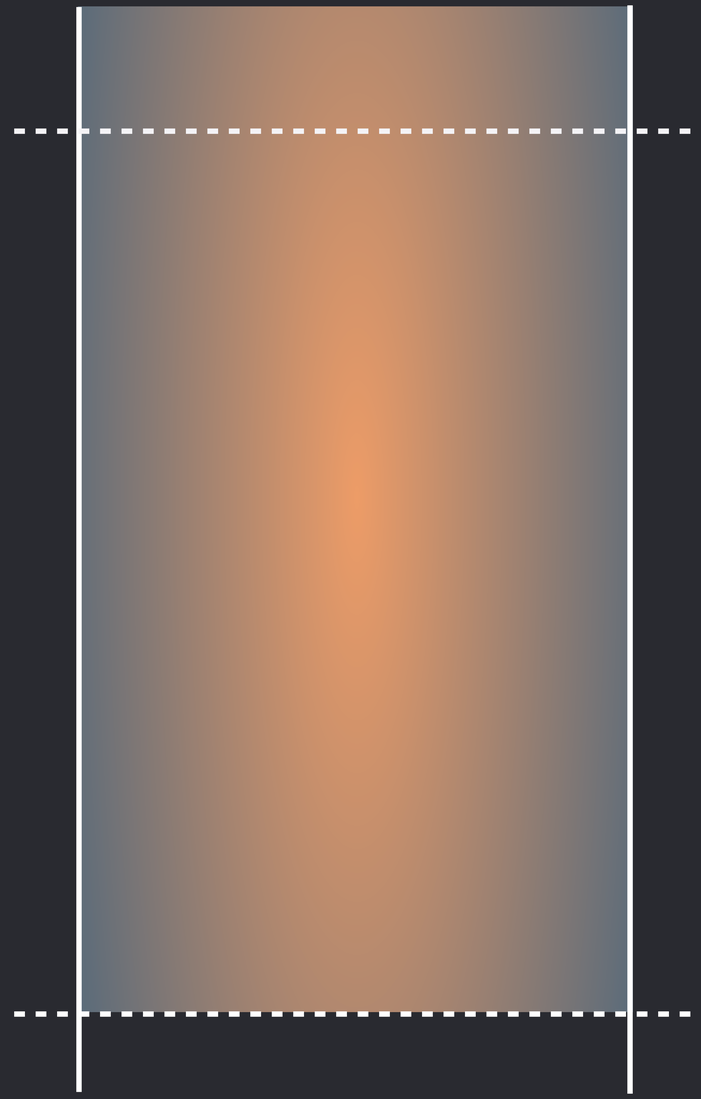

Heap

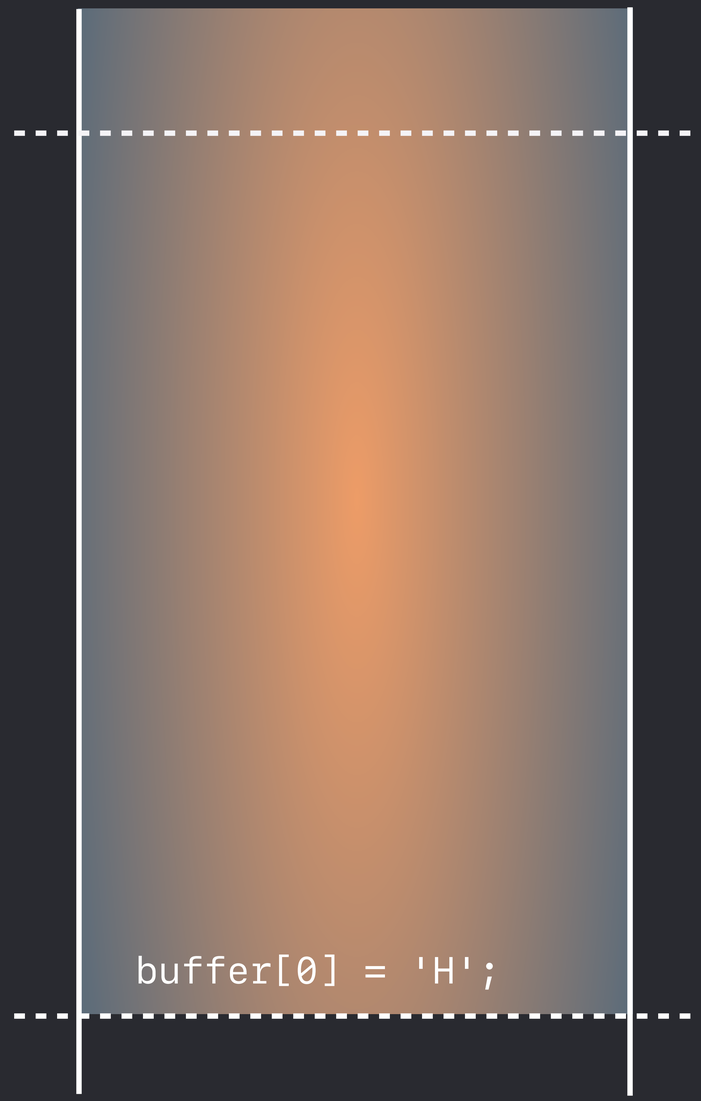

Heap


Heap


Heap

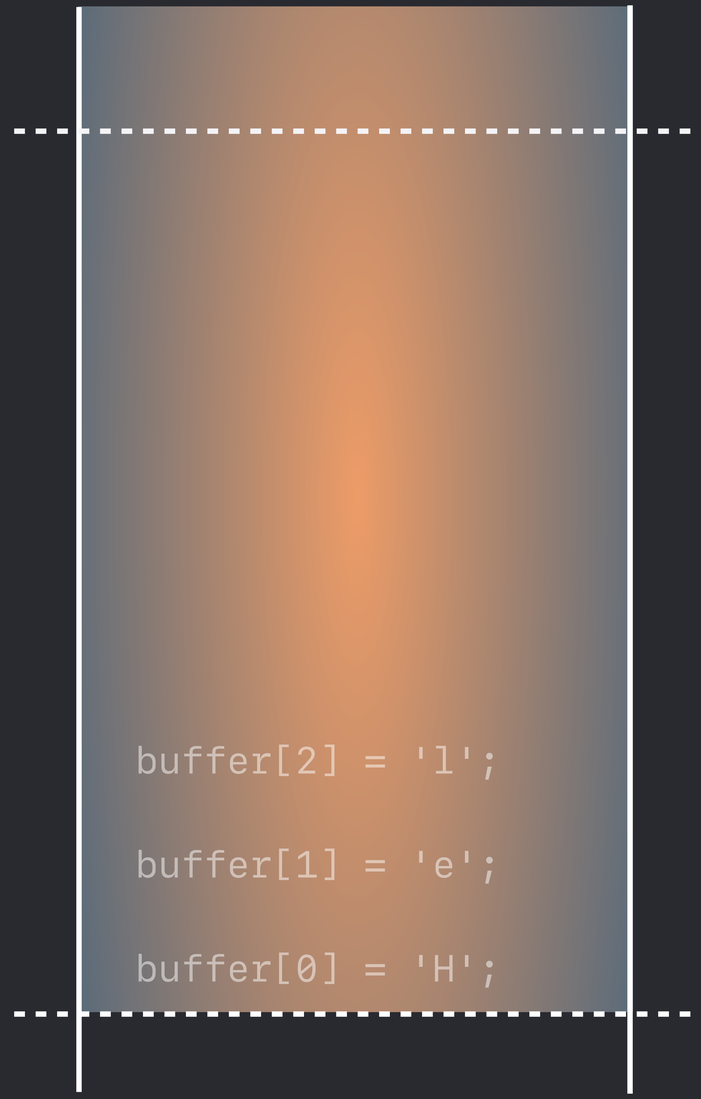

Heap

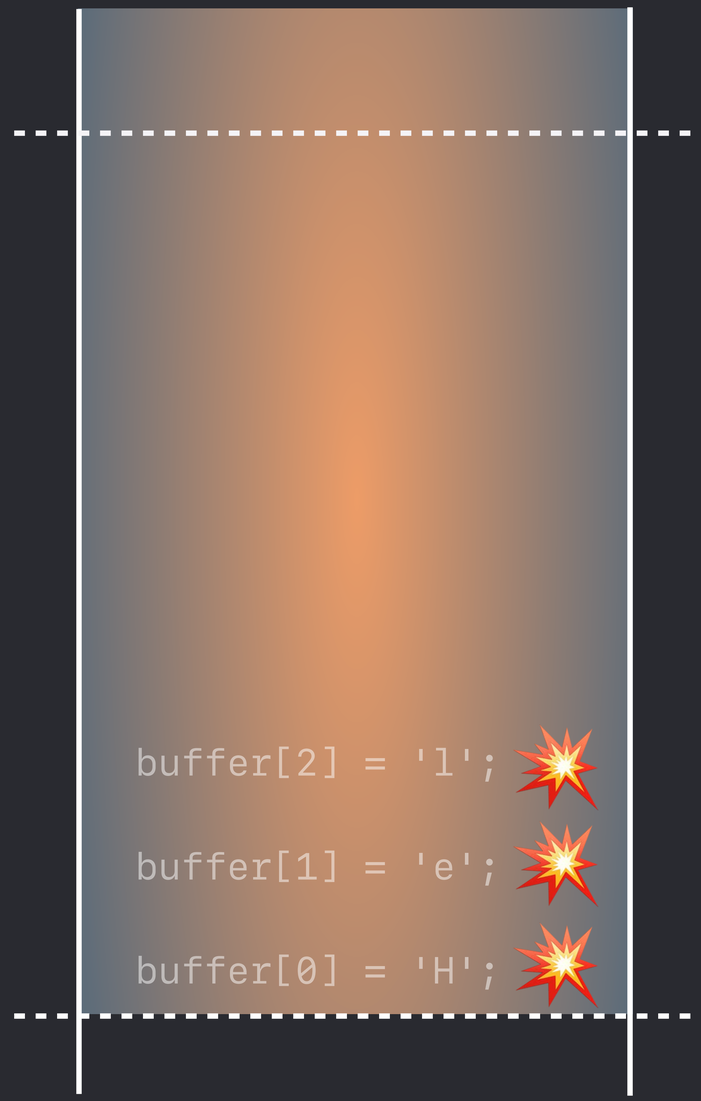


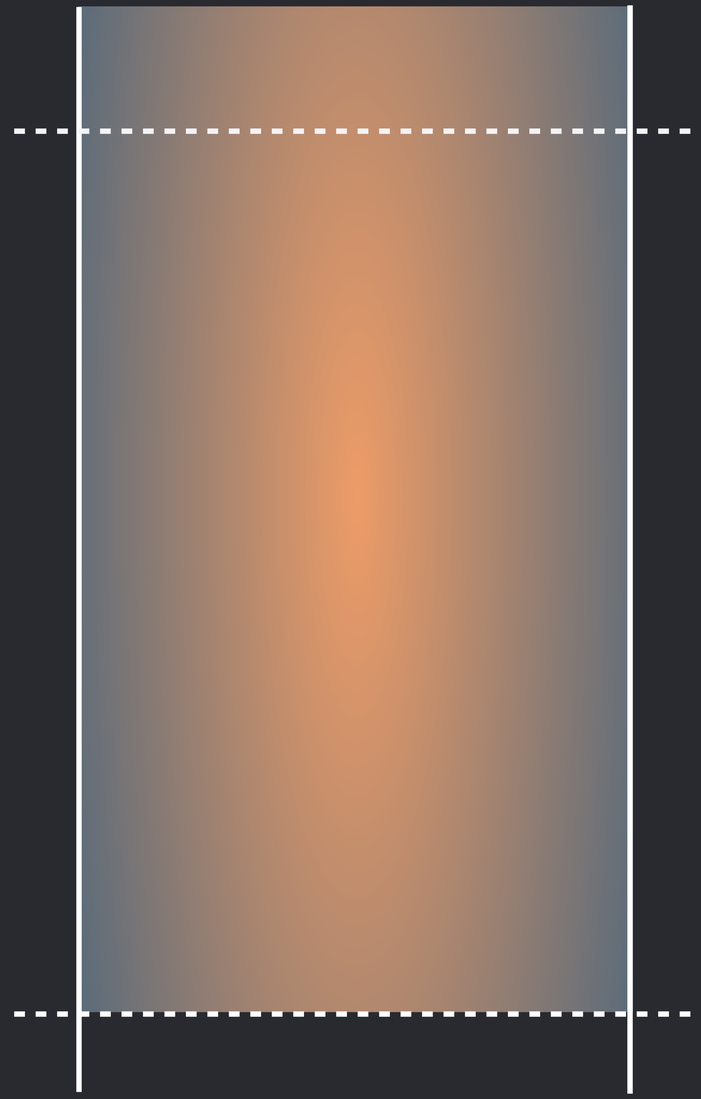

buffer[0] = 'H';

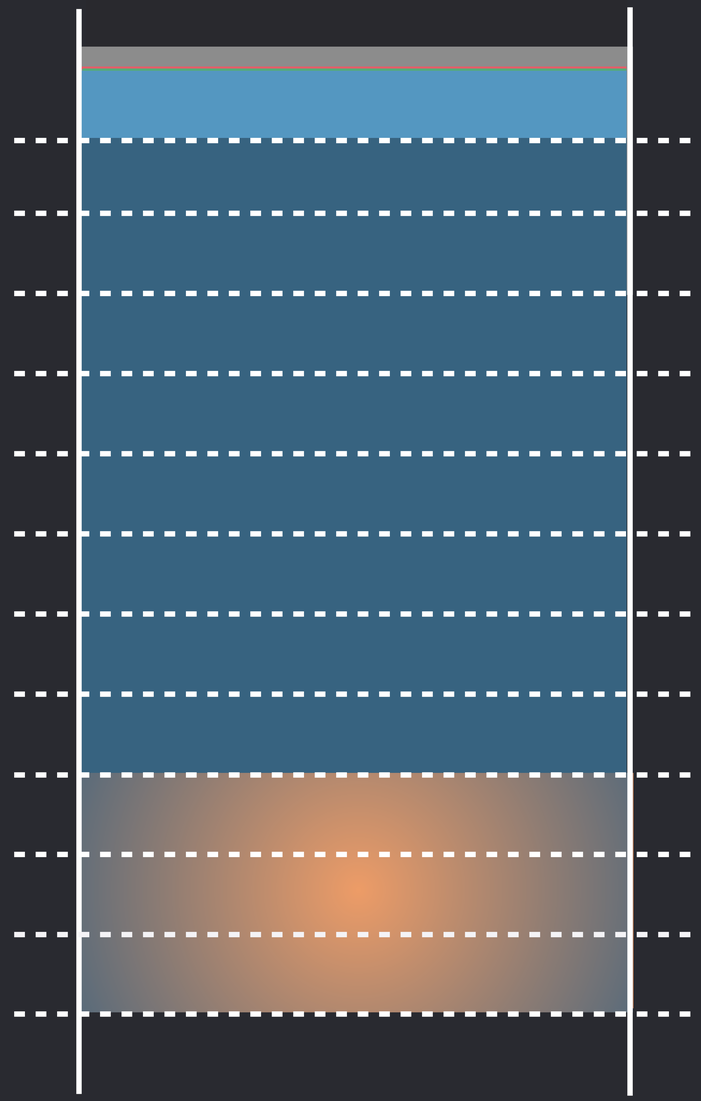

buffer[1] = 'e';
buffer[0] = 'H';


buffer[2] = 'l';
buffer[1] = 'e';
buffer[0] = 'H';

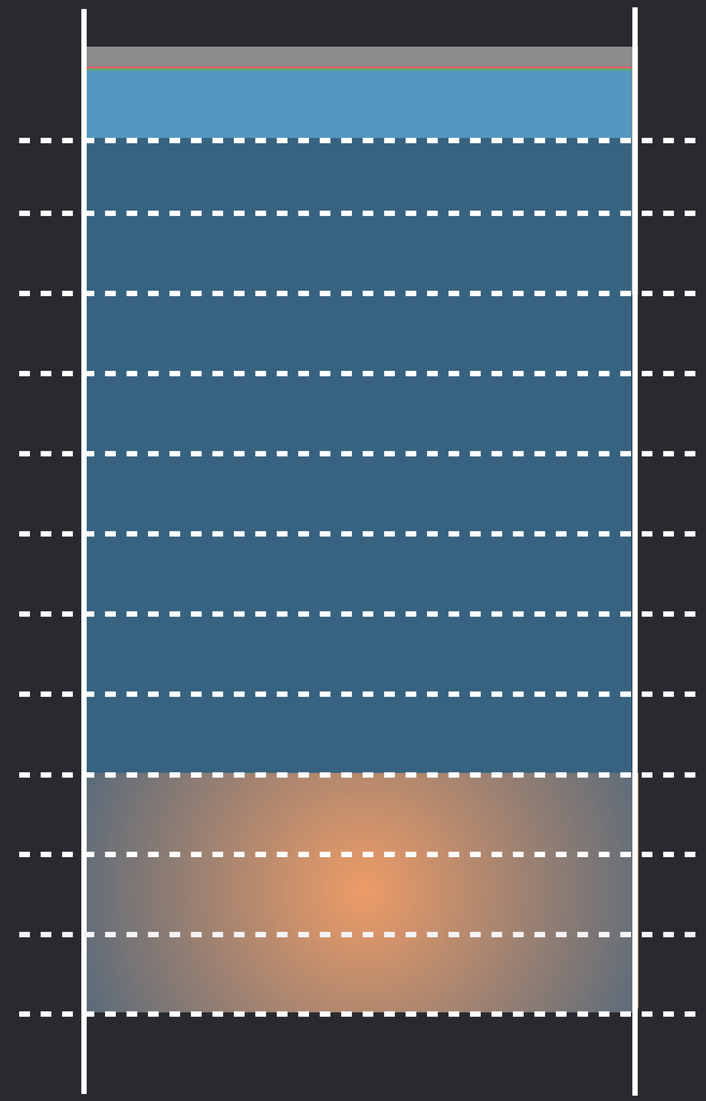

buffer[2] = 'l';
buffer[1] = 'e';
buffer[0] = 'H';


buffer[2] = 'l';
buffer[1] = 'e';
buffer[0] = 'H';

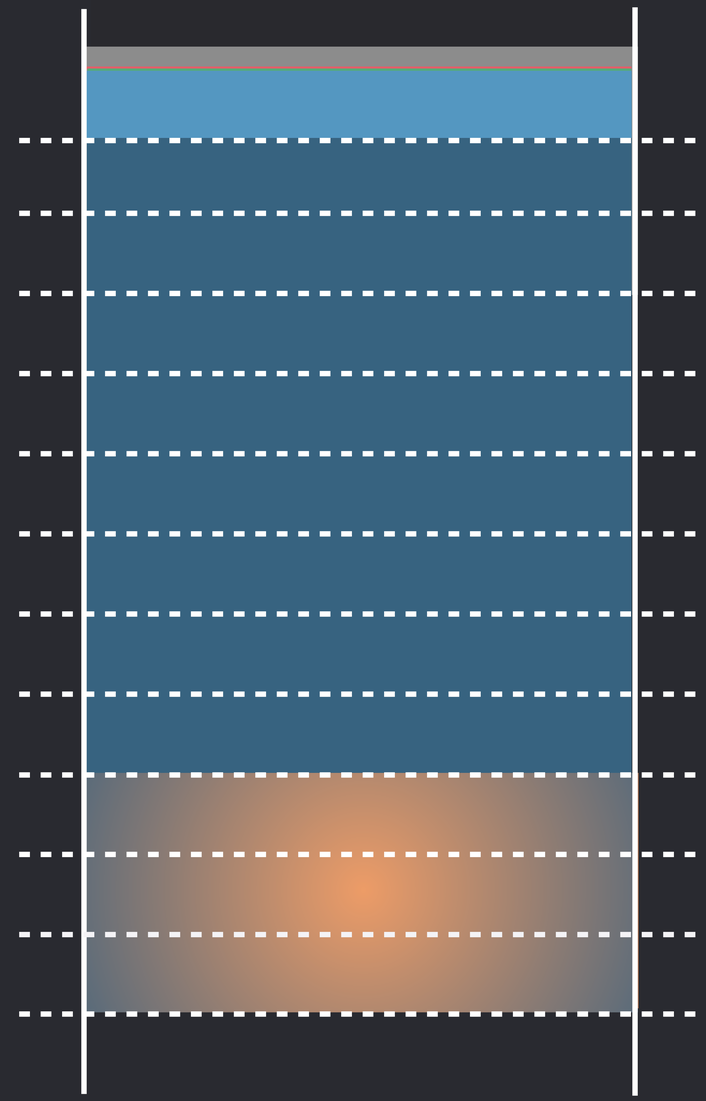




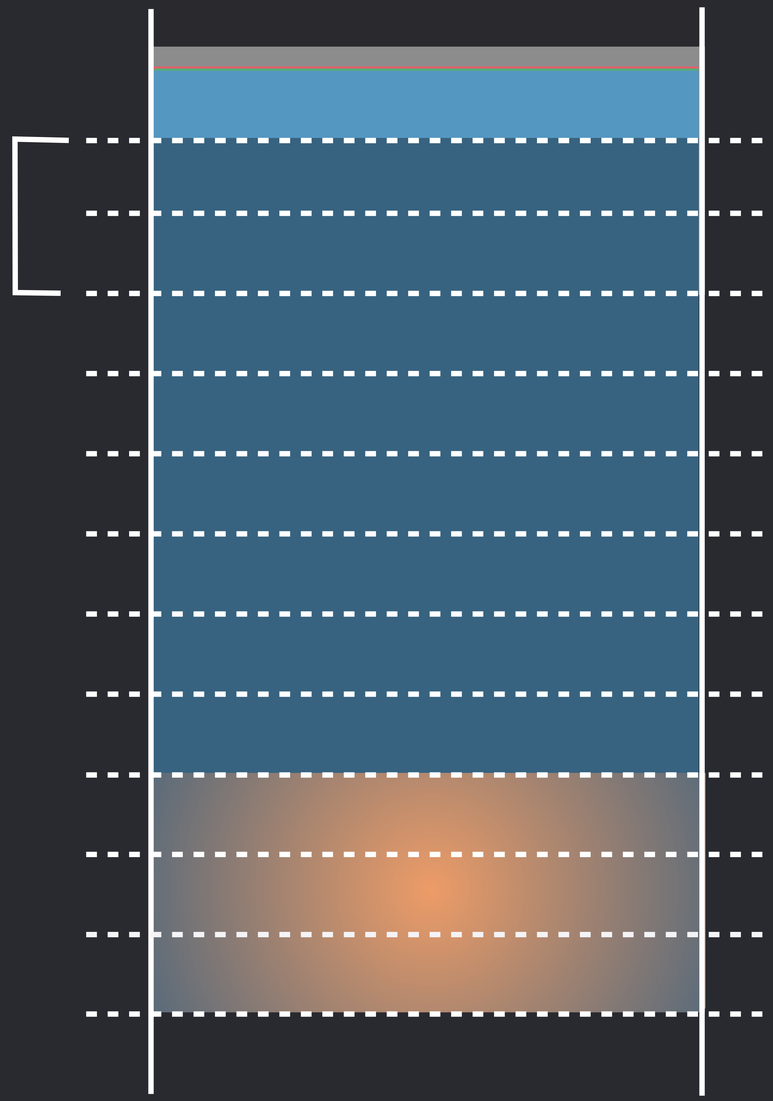
```
// Stack Checking

int dlog(const char *str, size_t len) {
  // if (buffer[len] not in stack)
  //   abort();
  char buffer[len];
  ...
}
```



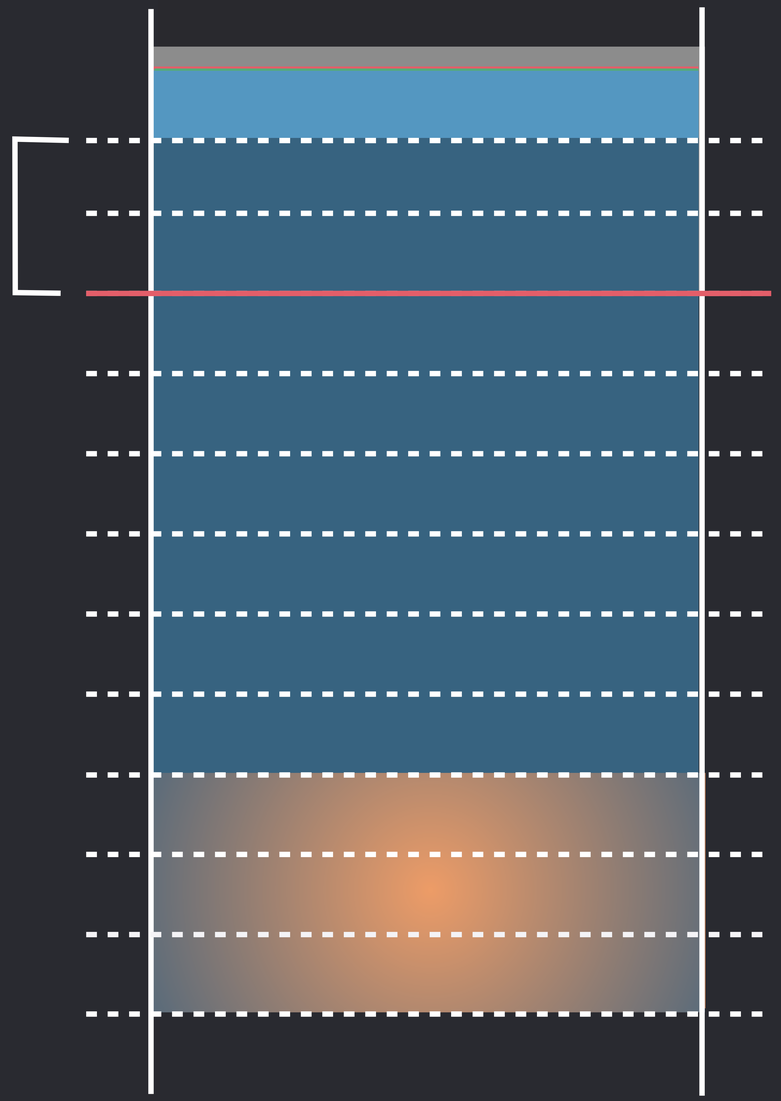
```
// Stack Checking

int dlog(const char *str, size_t len) {
  // if (buffer[len] not in stack)
  //   abort();
  char buffer[len];
  ...
}
```



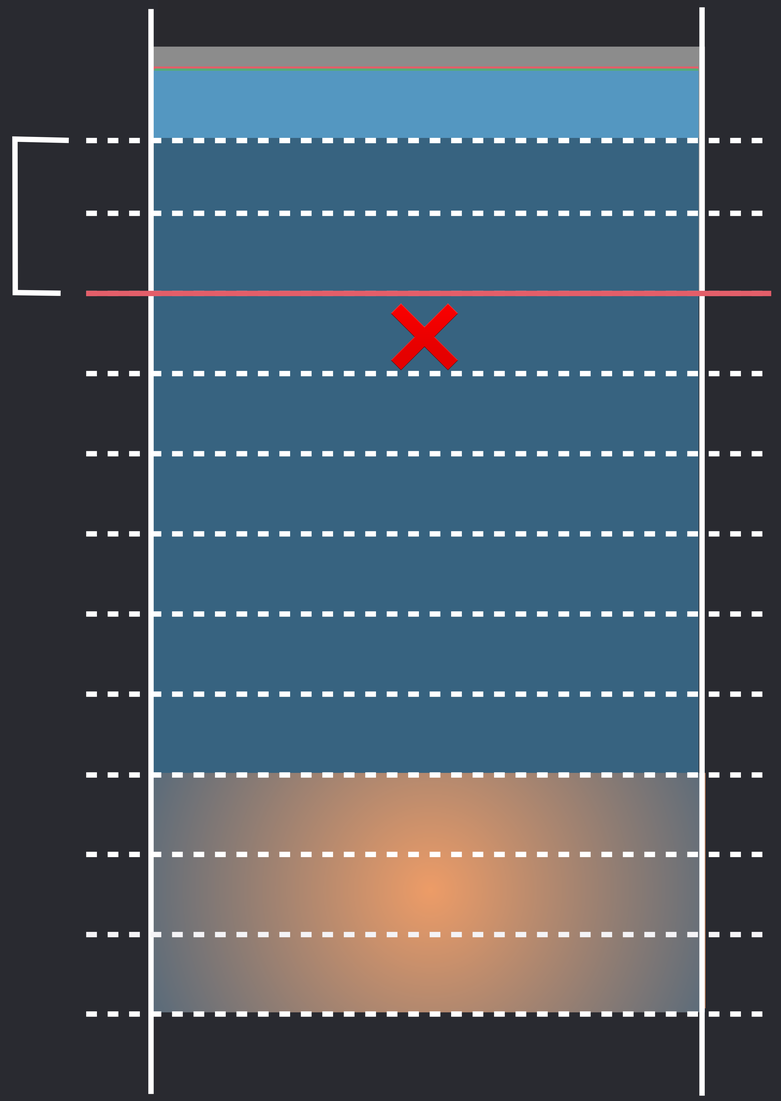
```c
// Stack Checking

int dlog(const char *str, size_t len) {
  // if (buffer[len] not in stack)
  //   abort();
  char buffer[len];
  ...
}
```



```
// Stack Checking

int dlog(const char *str, size_t len) {
  // if (buffer[len] not in stack)
  //   abort();
  char buffer[len];
  ...
}
```



```
// Stack Checking

int dlog(const char *str, size_t len) {
  // if (buffer[len] not in stack)
  //   abort();
  char buffer[len];
  ...
}
```



Stack Checking

NEW

Stack Checking

NEW

Detects “Stack Clash”

Stack Checking

NEW

Detects “Stack Clash”

Enabled by default in Xcode 10

New Instruction Set Extensions


MacBook


68° PARTLY CLOUDY


9:41
Monday, June 4


MacBook


Precise control over font, text style, & paragraph settings
Measuring tools
- Text spacing
- Alignment
- Grid tool
XCODE

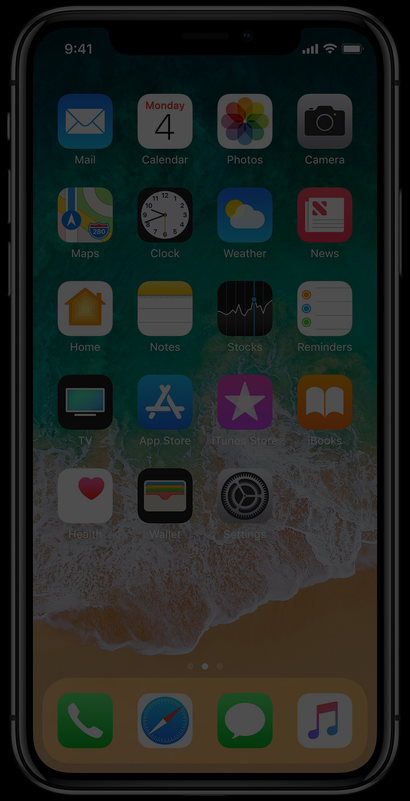

68° PARTLY CLOUDY


9:41
Monday, June 4

AVX-512

ARMv8.1 Atomics
ARMv8.2 16-Bit Float

AVX-512



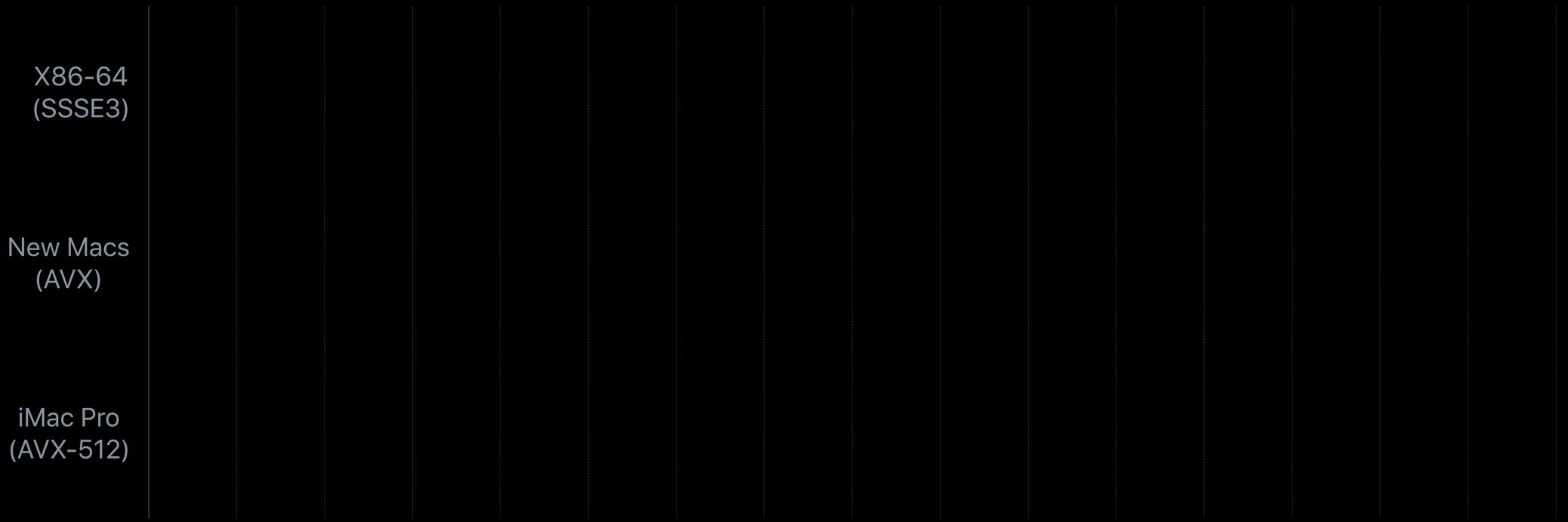
ARMv8.1 Atomics
ARMv8.2 16-Bit Float

AVX-512

512-bit vectors

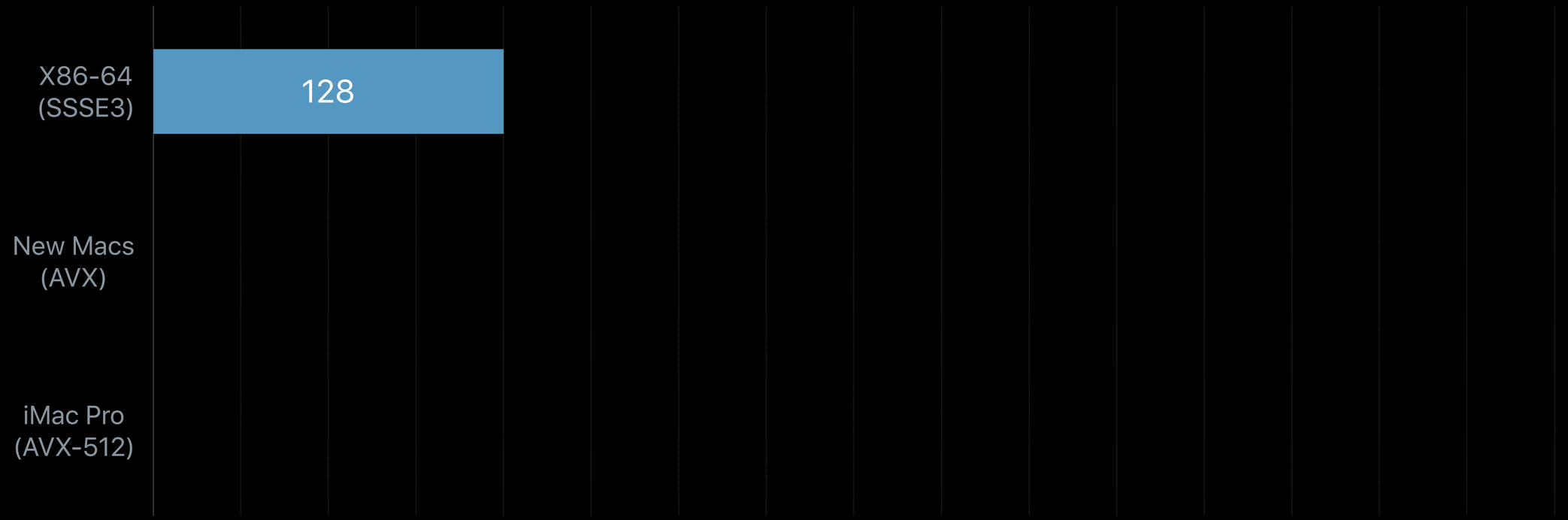
AVX-512

512-bit vectors



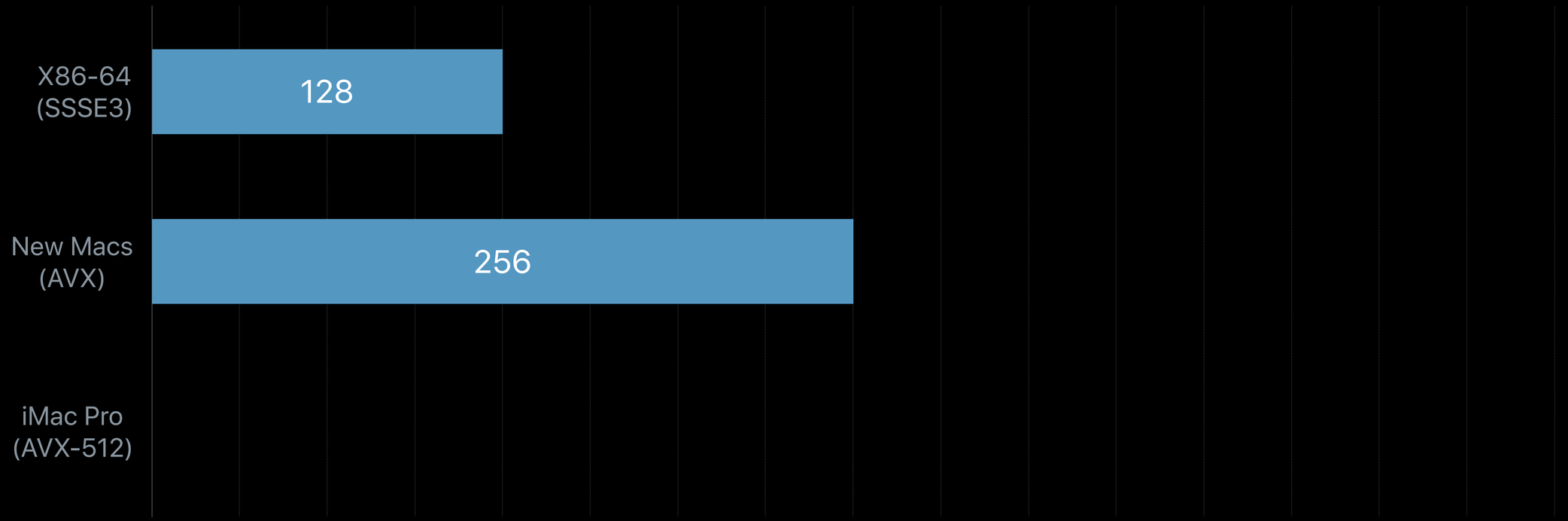
AVX-512

512-bit vectors



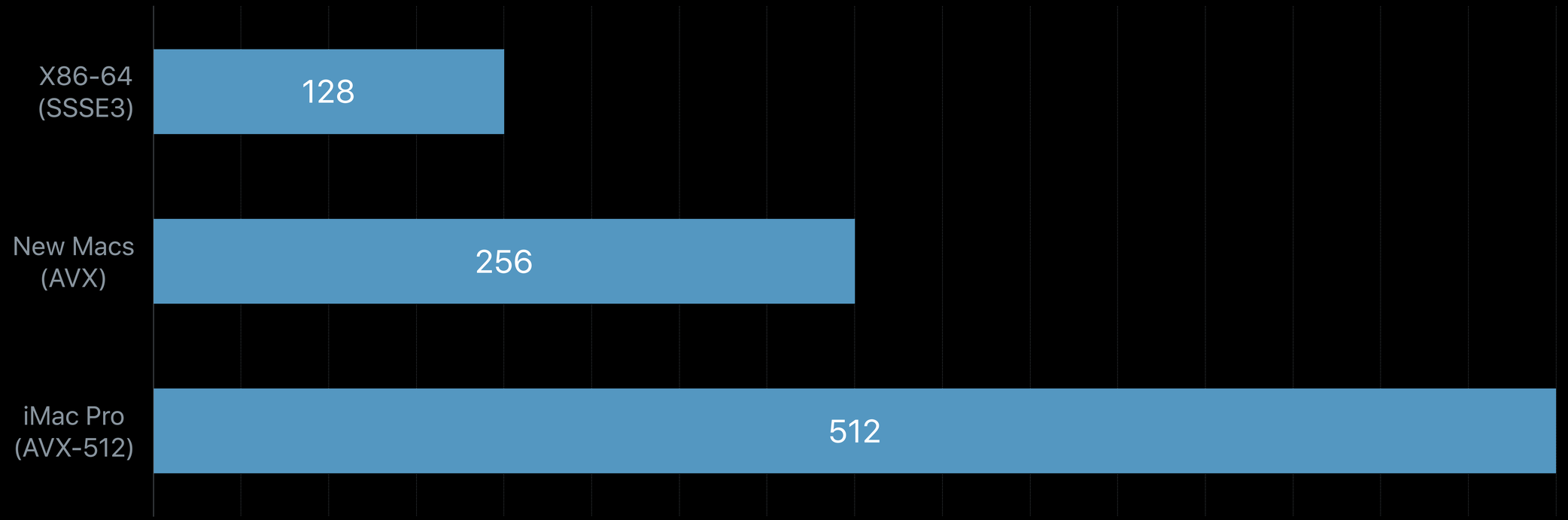
AVX-512

512-bit vectors



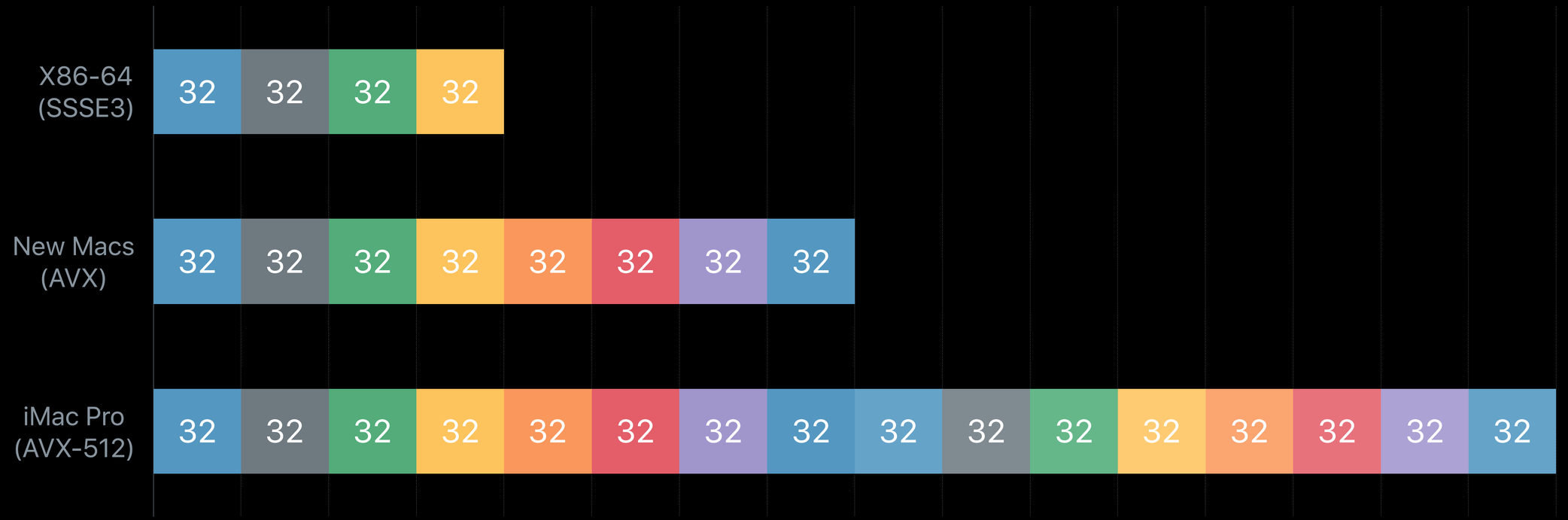
AVX-512

512-bit vectors



AVX-512

512-bit vectors



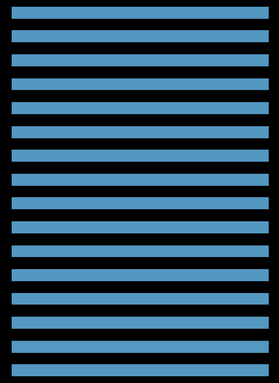
AVX-512

More vector registers

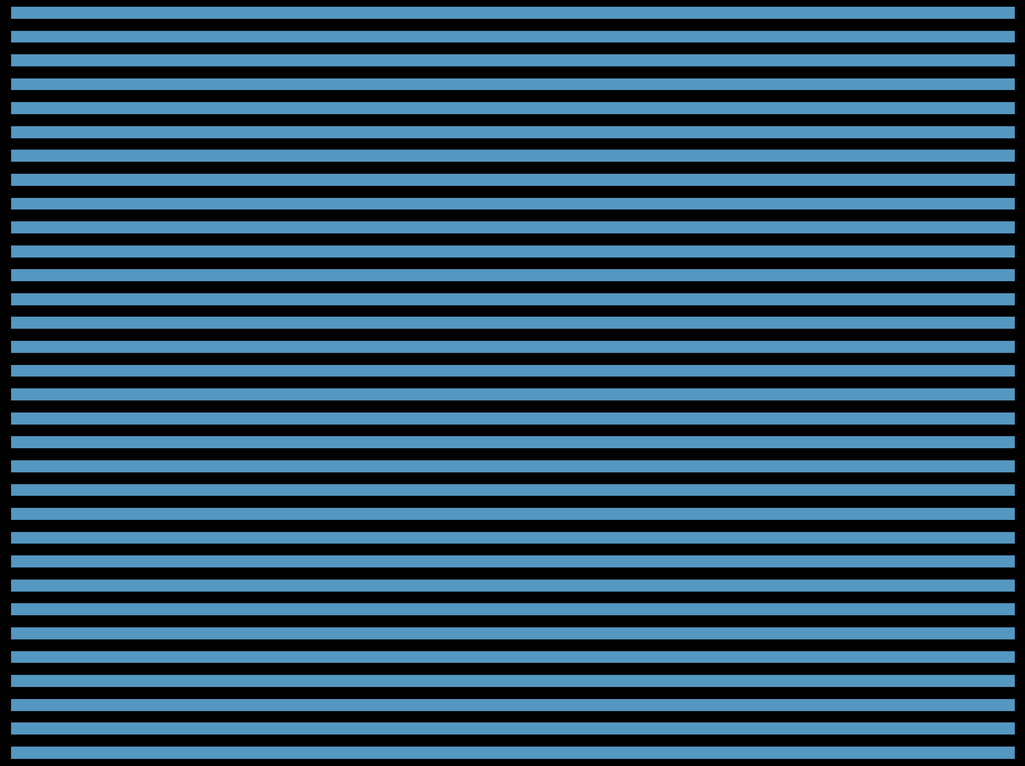
AVX-512

More vector registers

SSSE3



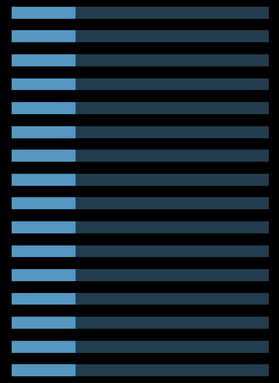
AVX-512



AVX-512

More scalar registers too

SSSE3



AVX-512

```
// Specializing a Function for AVX-512



void compute() {
  // expensive numerical computation
  ...
}
```

```
// Specializing a Function for AVX-512



void compute_generic() {
  // expensive numerical computation
  ...
}
```

```
// Specializing a Function for AVX-512



void compute_generic() {
  // expensive numerical computation
  ...
}


__attribute__((__target__("arch=skylake-avx512")))
void compute_avx512() {
  // expensive numerical computation tuned for AVX-512
  ...
}
```

```
// Specializing a Function for AVX-512



void compute_generic() {
  // expensive numerical computation
  ...
}


__attribute__((__target__("arch=skylake-avx512")))
void compute_avx512() {
  // expensive numerical computation tuned for AVX-512
  ...
}
```

```
// Specializing a Function for AVX-512



void compute_generic() {
  // expensive numerical computation
  ...
}


__attribute__((__target__("arch=skylake-avx512")))
void compute_avx512() {
  // expensive numerical computation tuned for AVX-512
  ...
}
```

```
// Specializing a Function for AVX-512

#include <simd/simd.h>

void compute_generic() {
  simd_float4 vector;
  vector = a + b;
  ...
}

__attribute__((__target__("arch=skylake-avx512")))
void compute_avx512() {
  simd_float4 vector;
  vector = a + b;
  ...
}
```

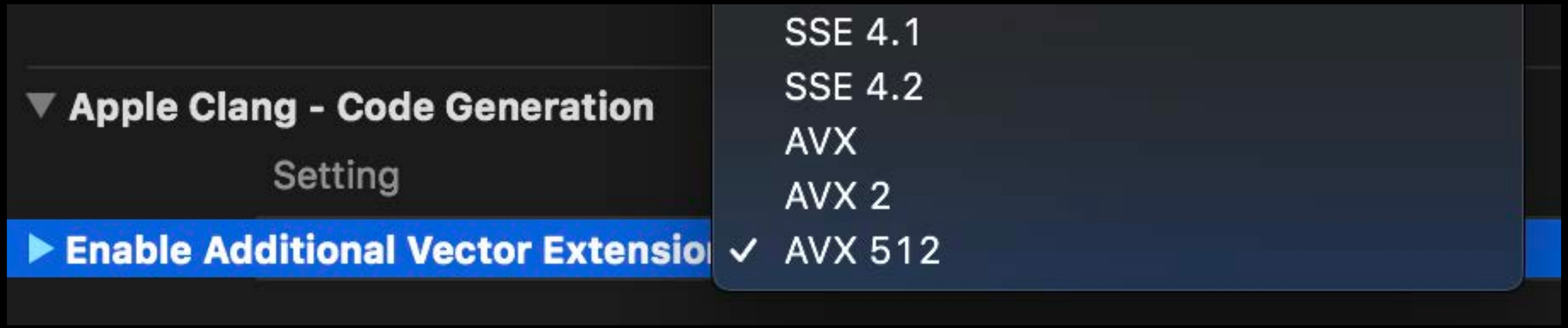
```
// Specializing a Function for AVX-512

#include <simd/simd.h>

void compute_generic() {
  simd_float16 vector;
  vector = a + b;
  ...
}

__attribute__((__target__("arch=skylake-avx512")))
void compute_avx512() {
  simd_float16 vector;
  vector = a + b;
  ...
}
```

```
// Specializing a Function for AVX-512

#include <immintrin.h>

void compute_generic() {
  // expensive numerical computation
  ...
}



__attribute__((__target__("arch=skylake-avx512")))
void compute_avx512() {
  __m512 vector;
  ...
}
```

AVX-512

Enabling in Xcode

NEW



AVX-512

Considerations

AVX-512

Considerations

Only pass large vectors to/from AVX-512 functions

AVX-512

Considerations

Only pass large vectors to/from AVX-512 functions

Align AVX-512 vectors properly

AVX-512

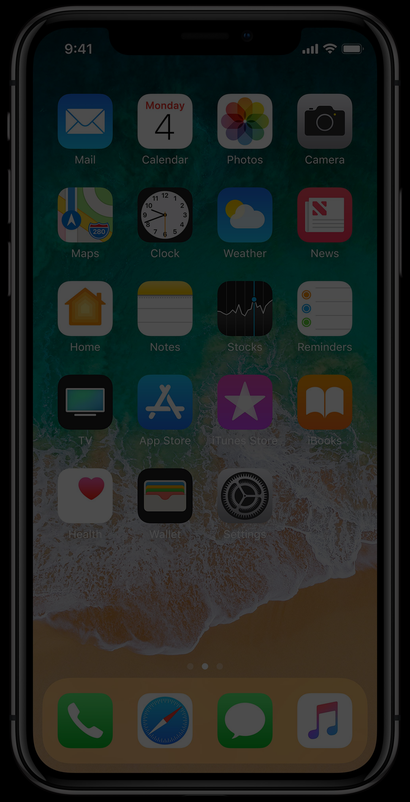
Considerations

Only pass large vectors to/from AVX-512 functions

Align AVX-512 vectors properly

Use Accelerate.framework!

AVX-512



ARMv8.1 Atomics
ARMv8.2 16-Bit Float

AVX-512

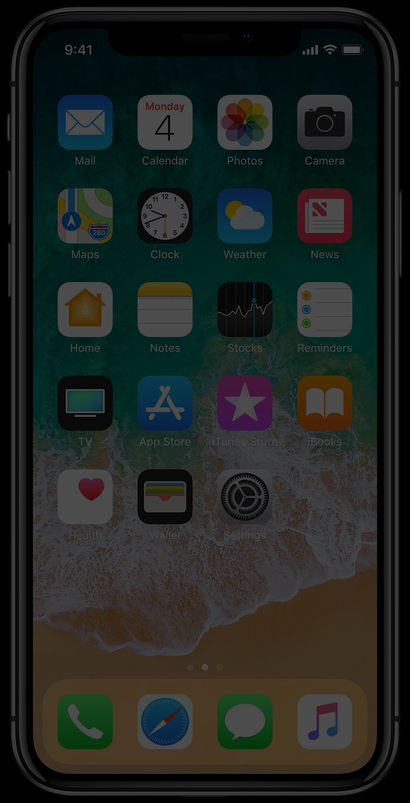
ARMv8.1 Atomics
ARMv8.2 16-Bit Float

AVX-512

ARMv8.1 Atomics

ARMv8.2 16-Bit Float

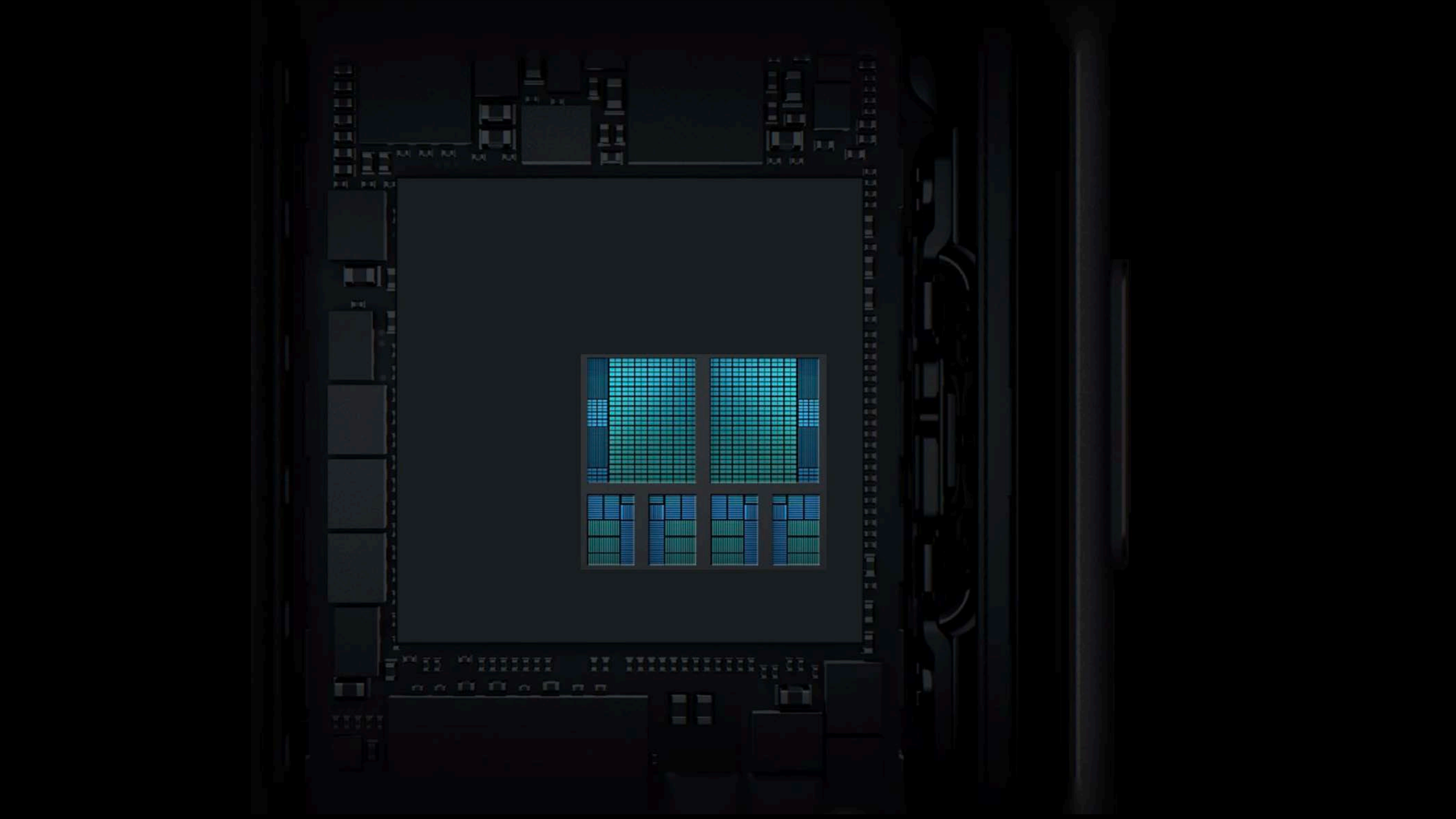
AVX-512



ARMv8.1 Atomics

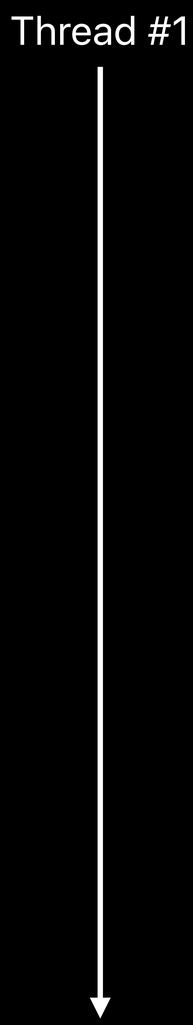
ARMv8.2 16-Bit Float


A11
BIONIC

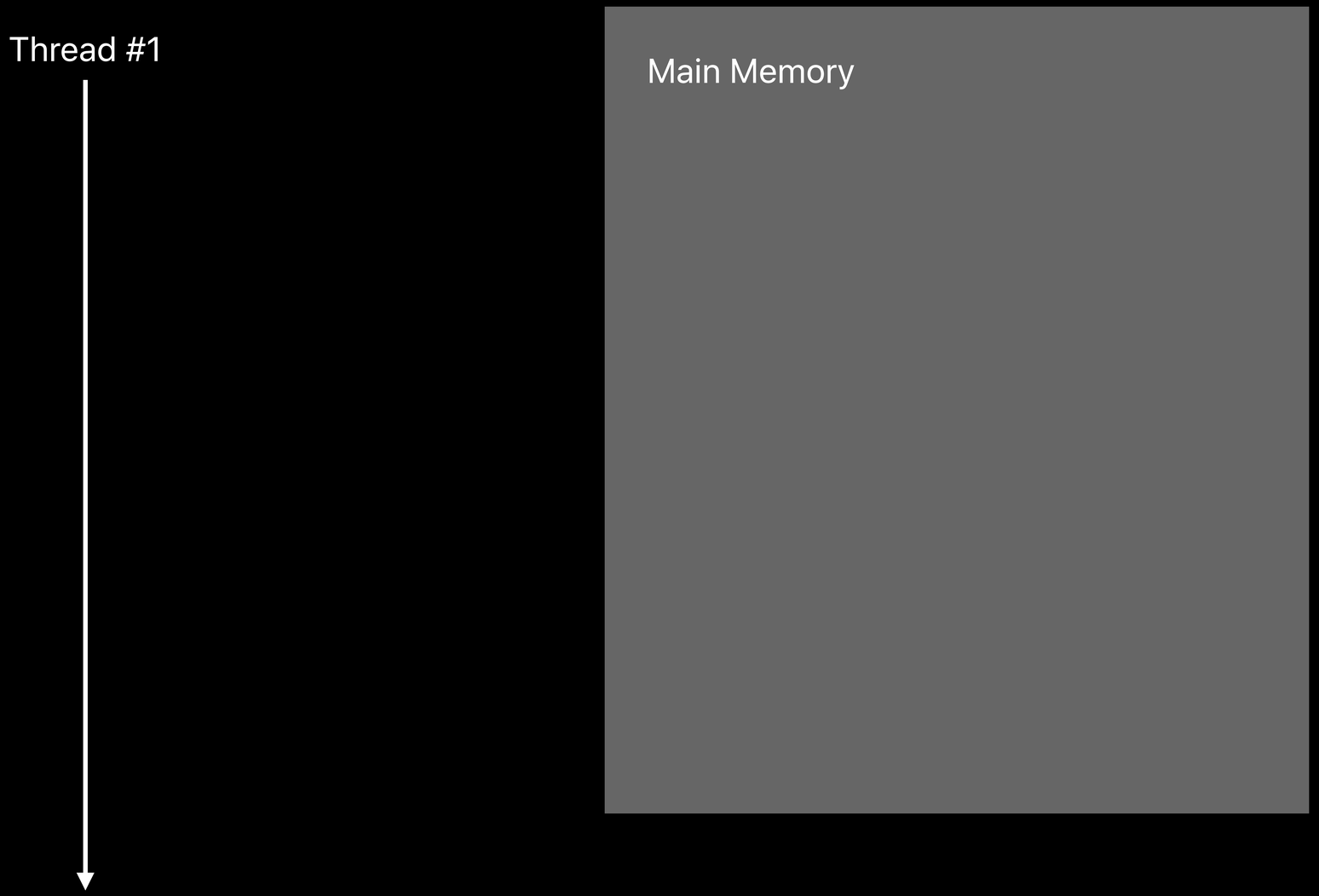


Atomics

Atomics



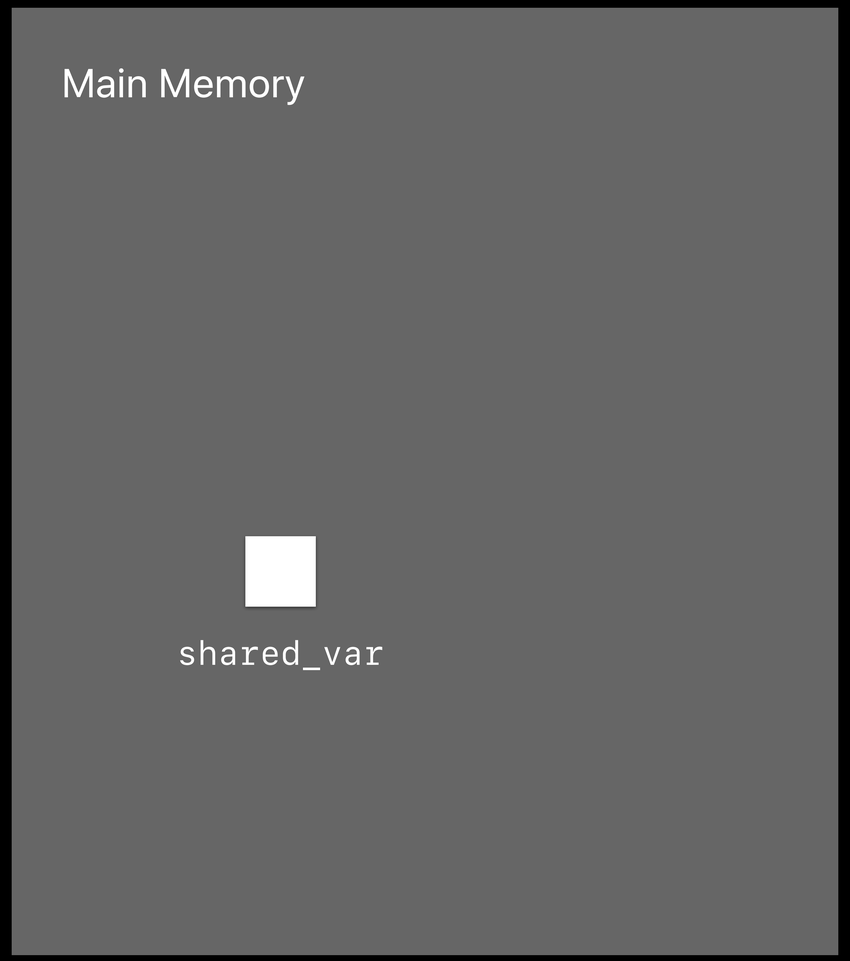
Atomics



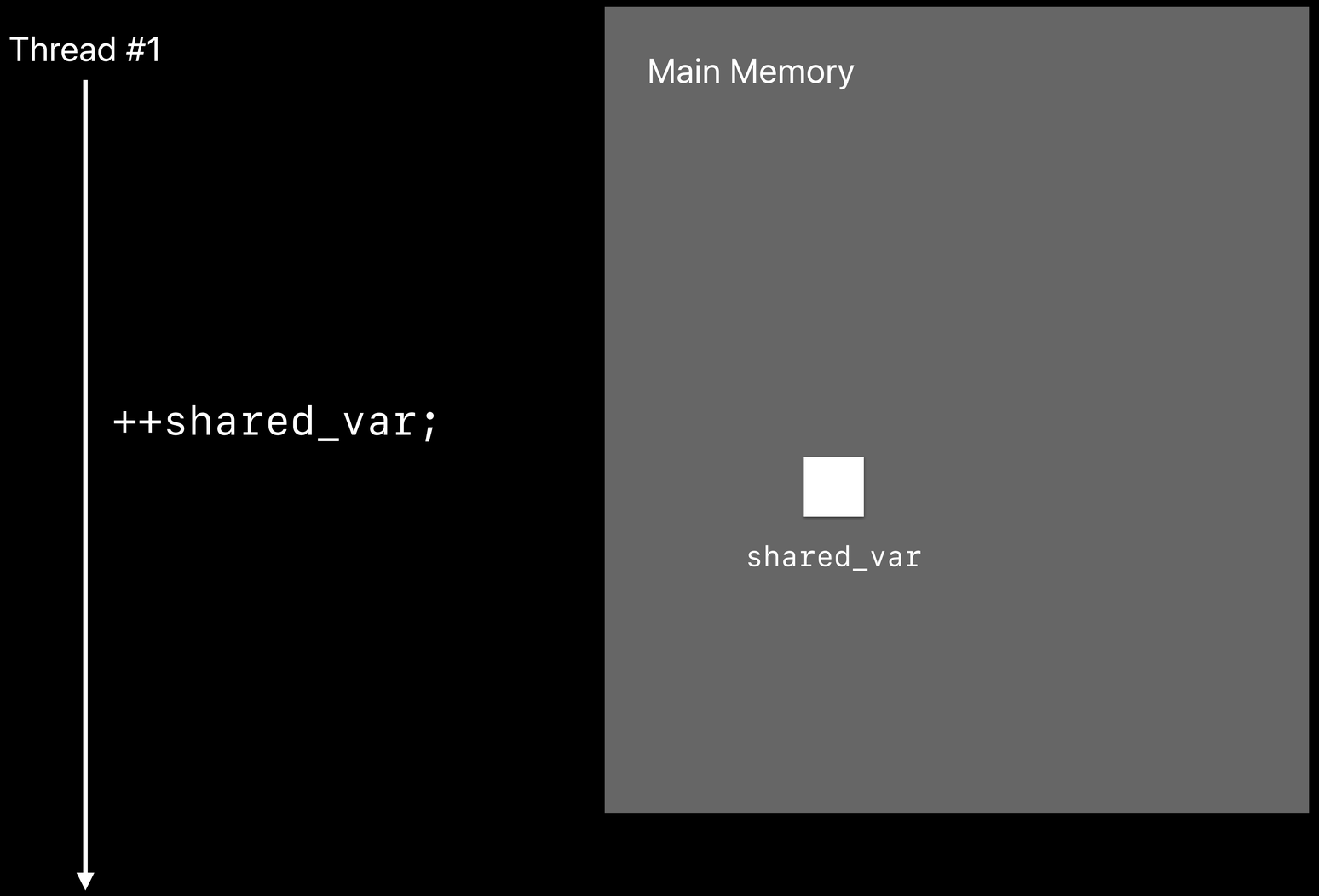
Atomics

Thread #1

```
_Atomic int
  shared_var;
```

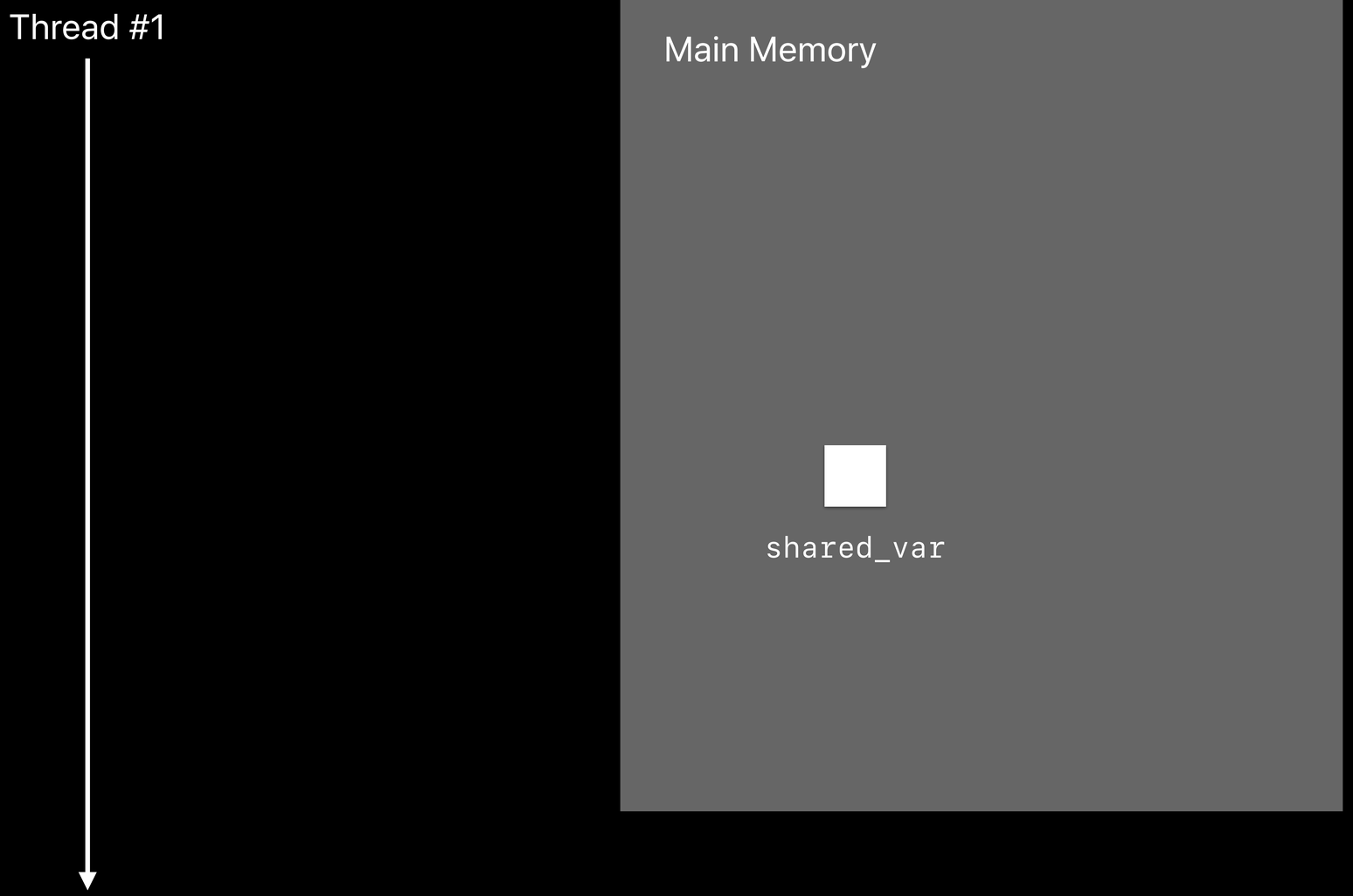


Atomics



Atomics

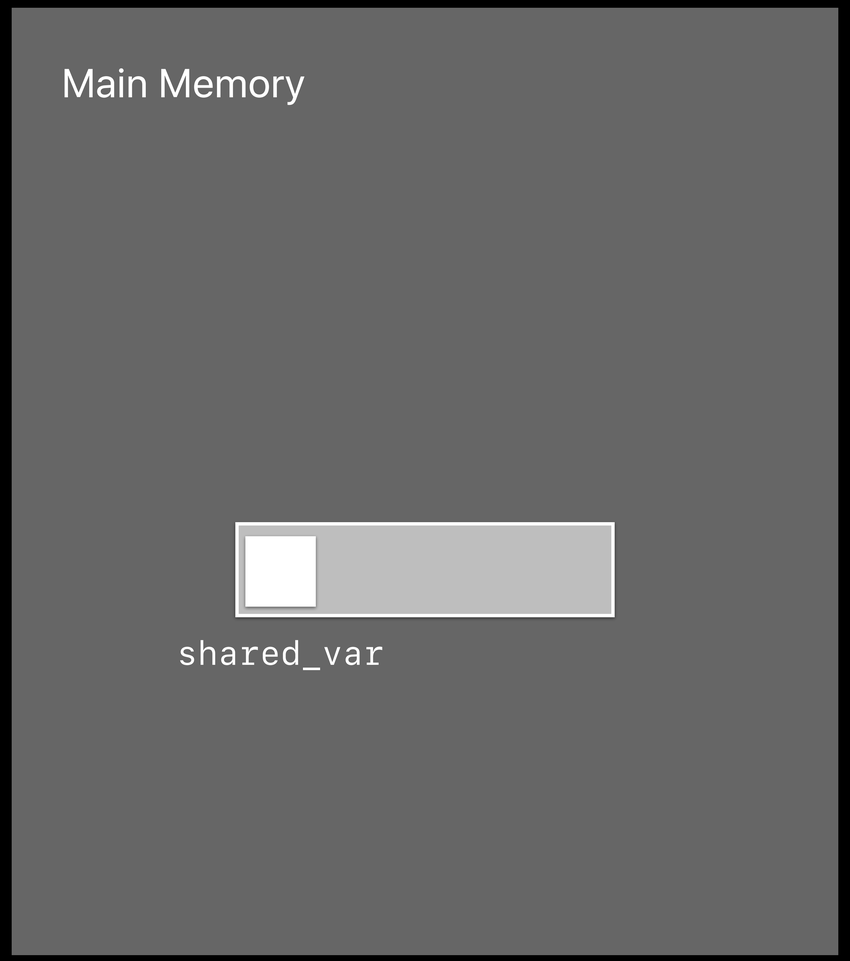
ARM v8



Atomics

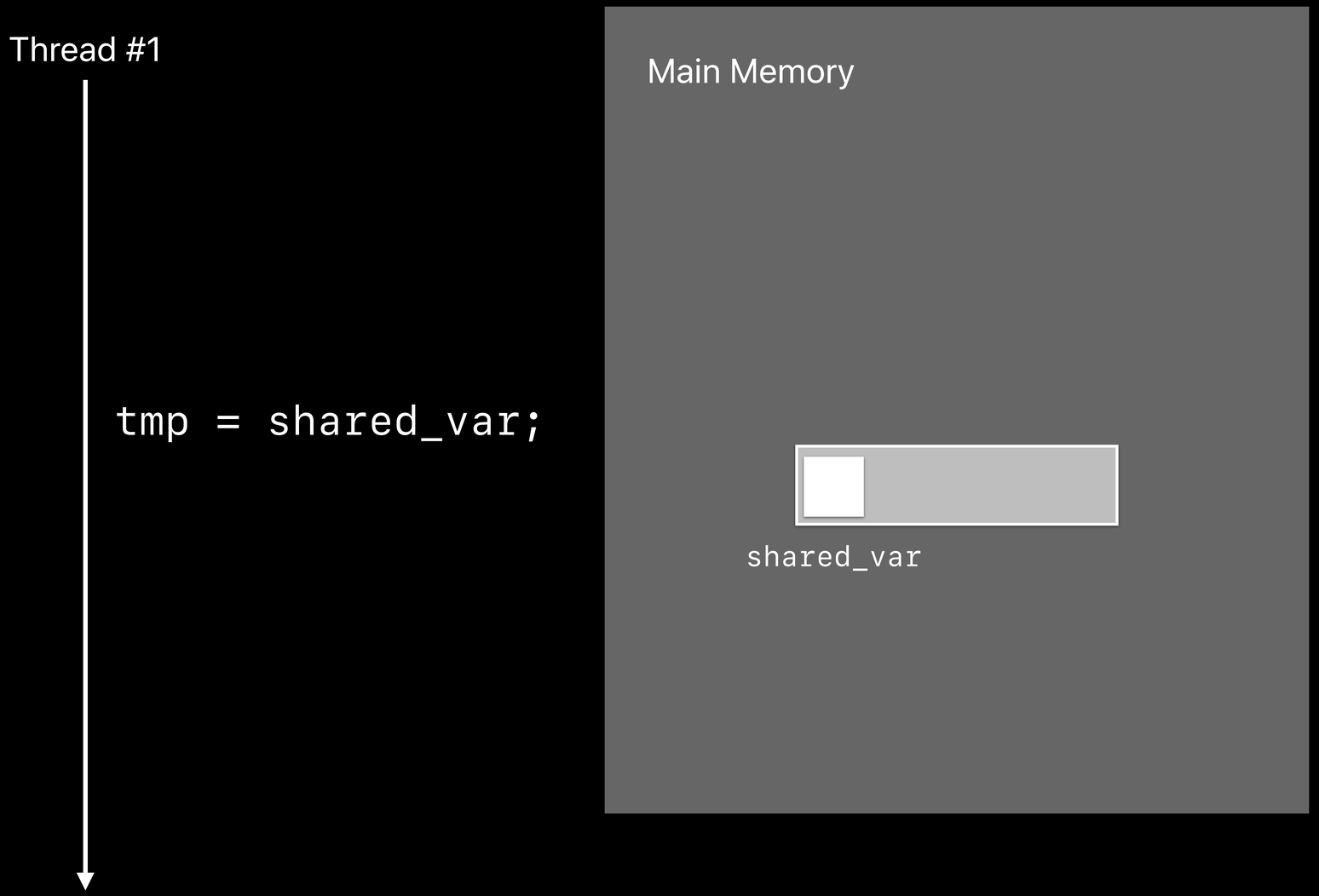
ARM v8

Thread #1



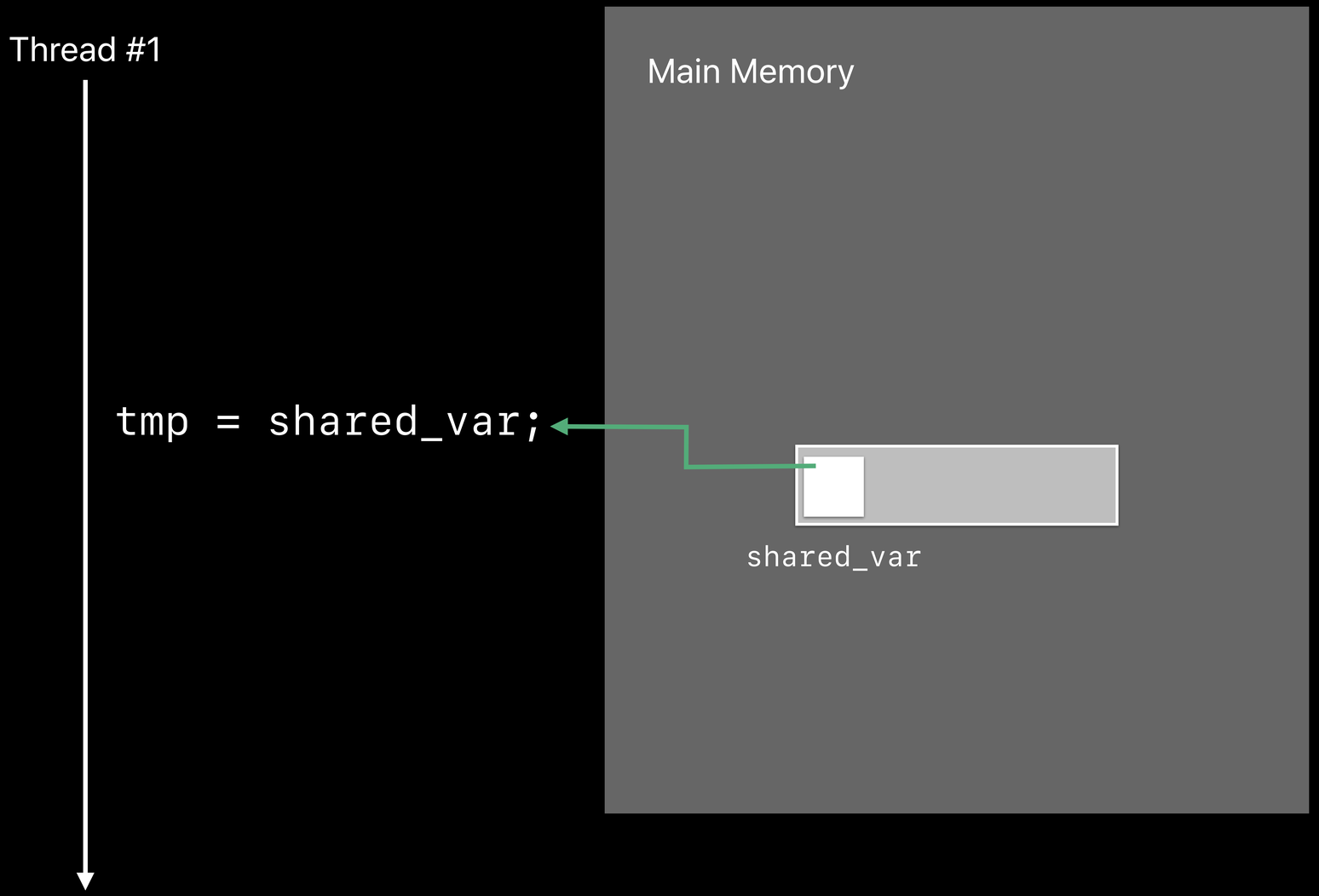
Atomics

ARM v8



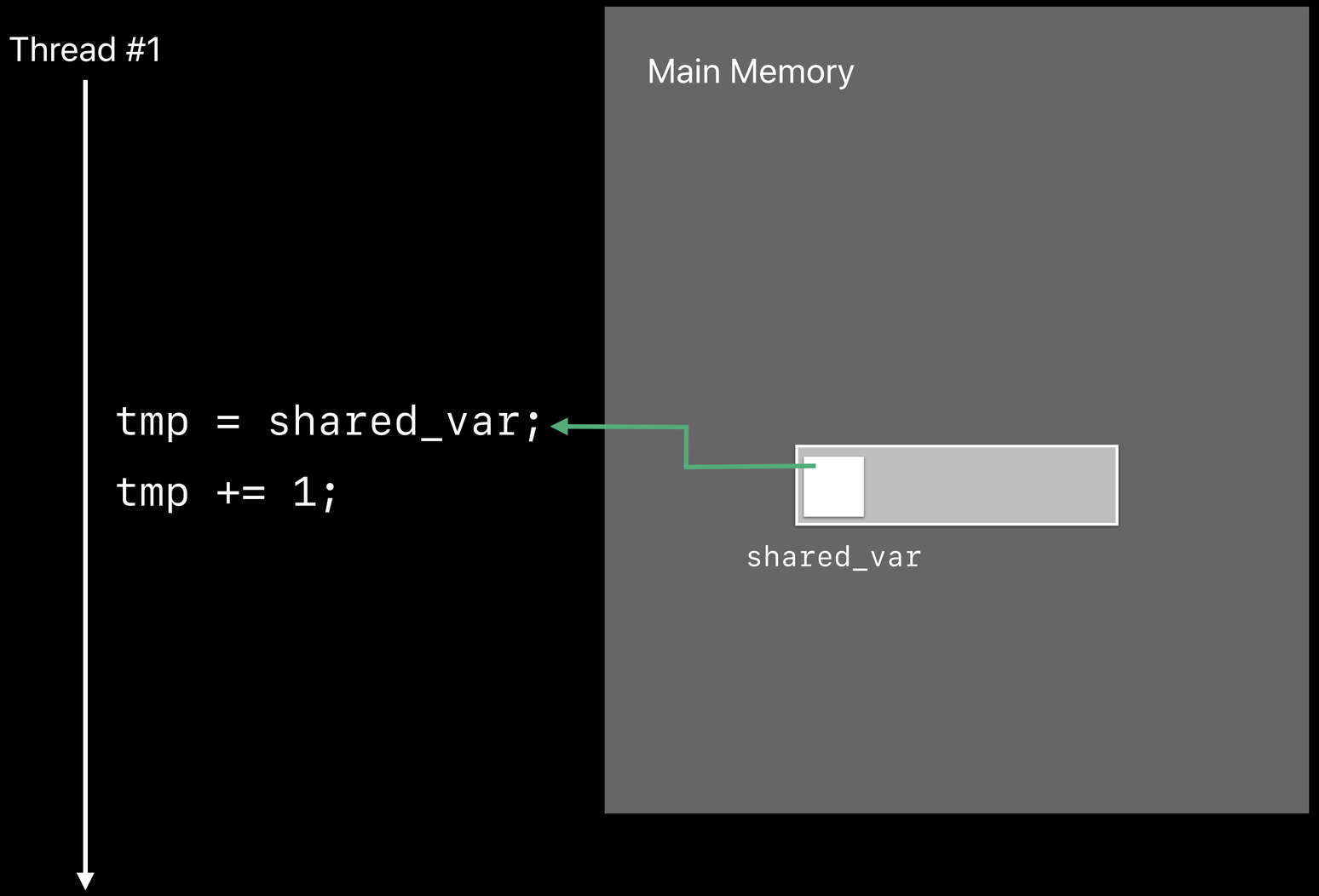
Atomics

ARM v8



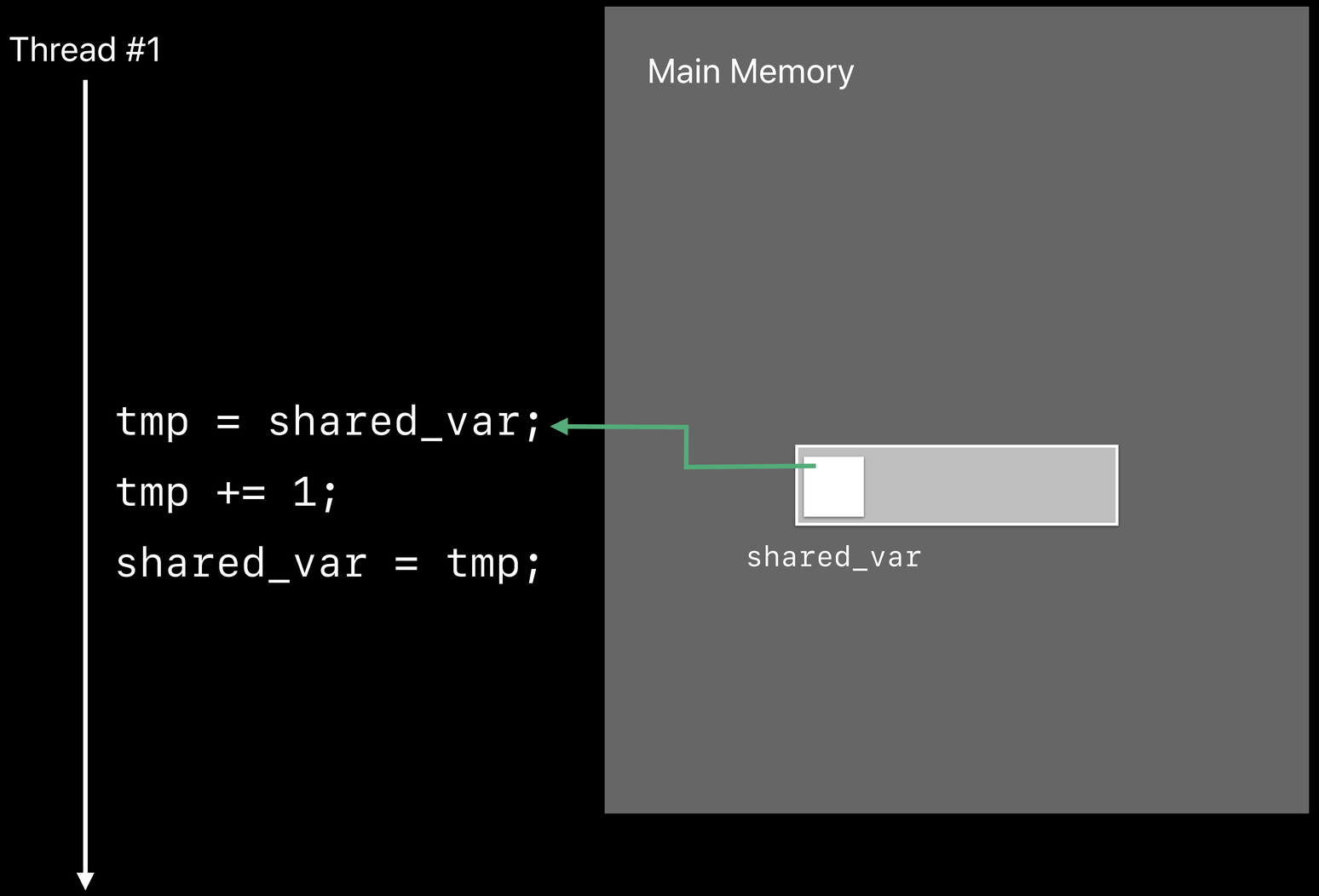
Atomics

ARM v8



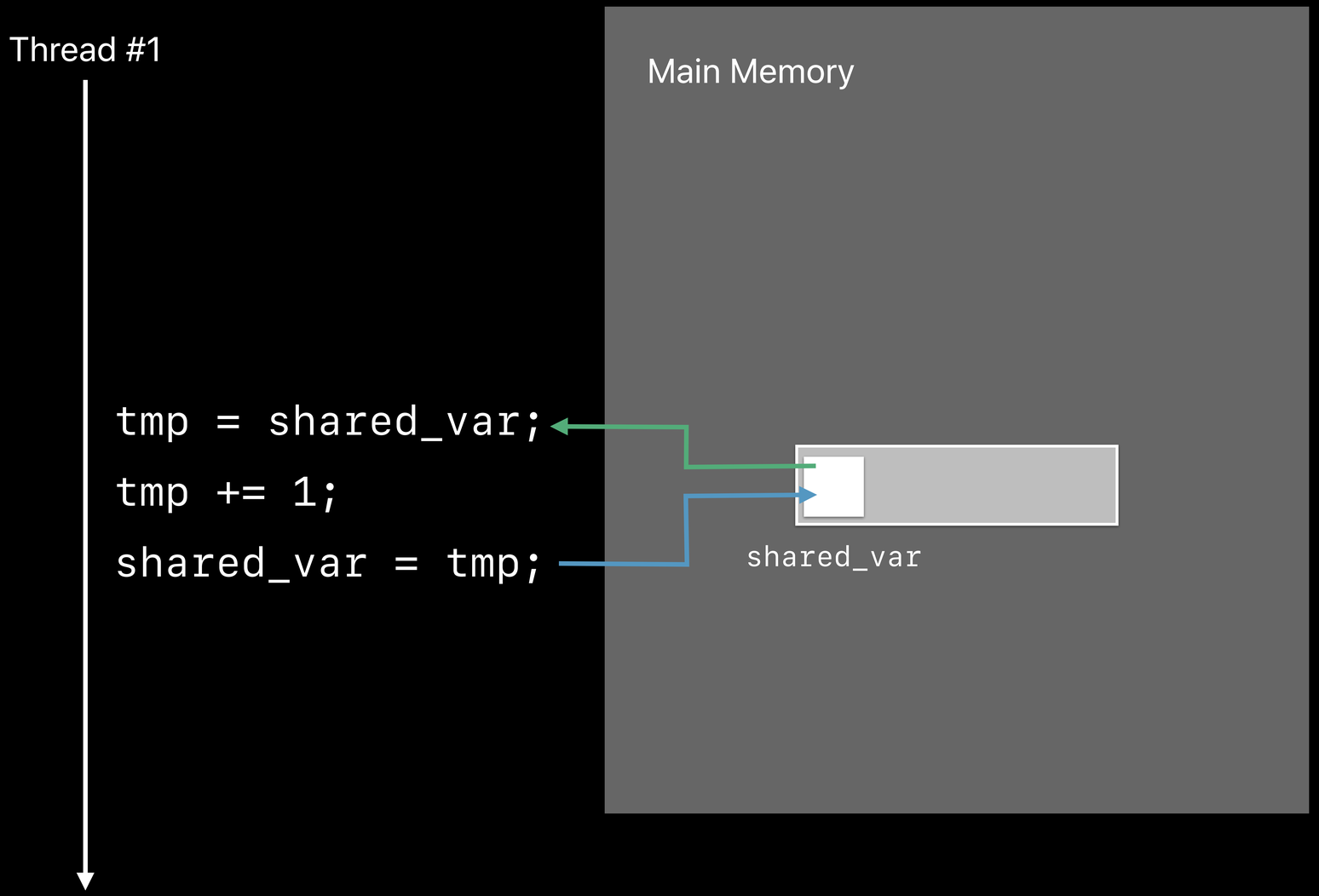
Atomics

ARM v8



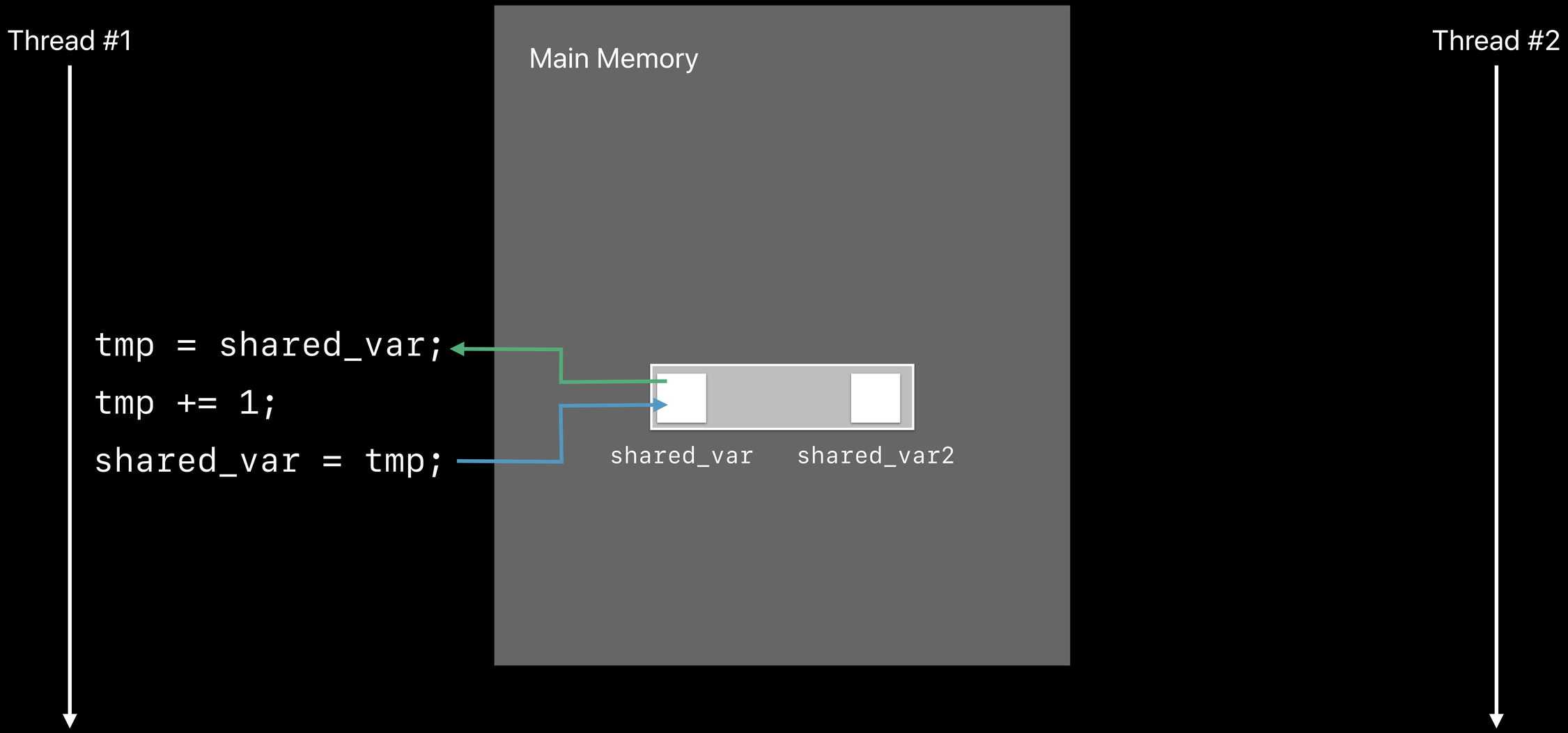
Atomics

ARM v8



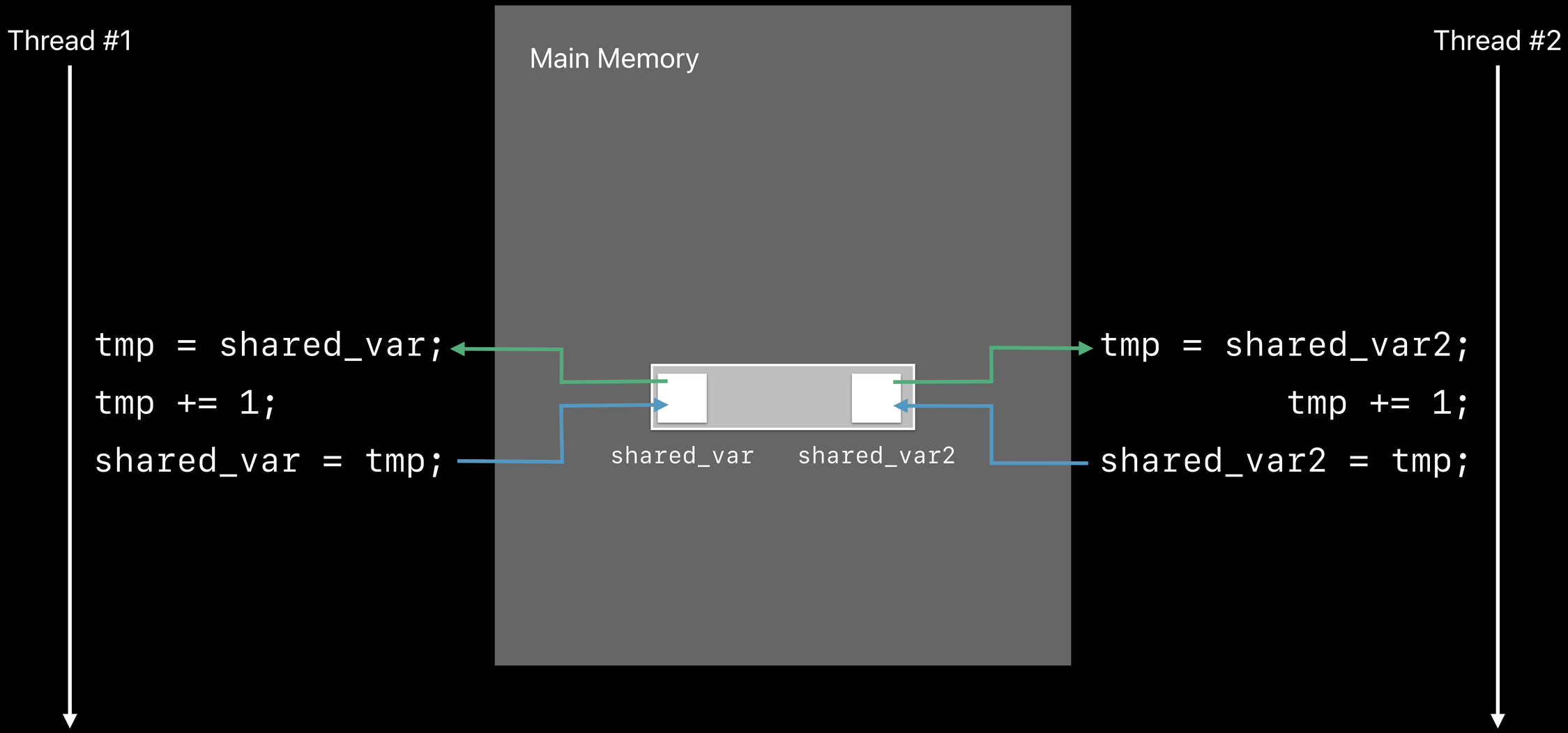
Atomics

ARM v8



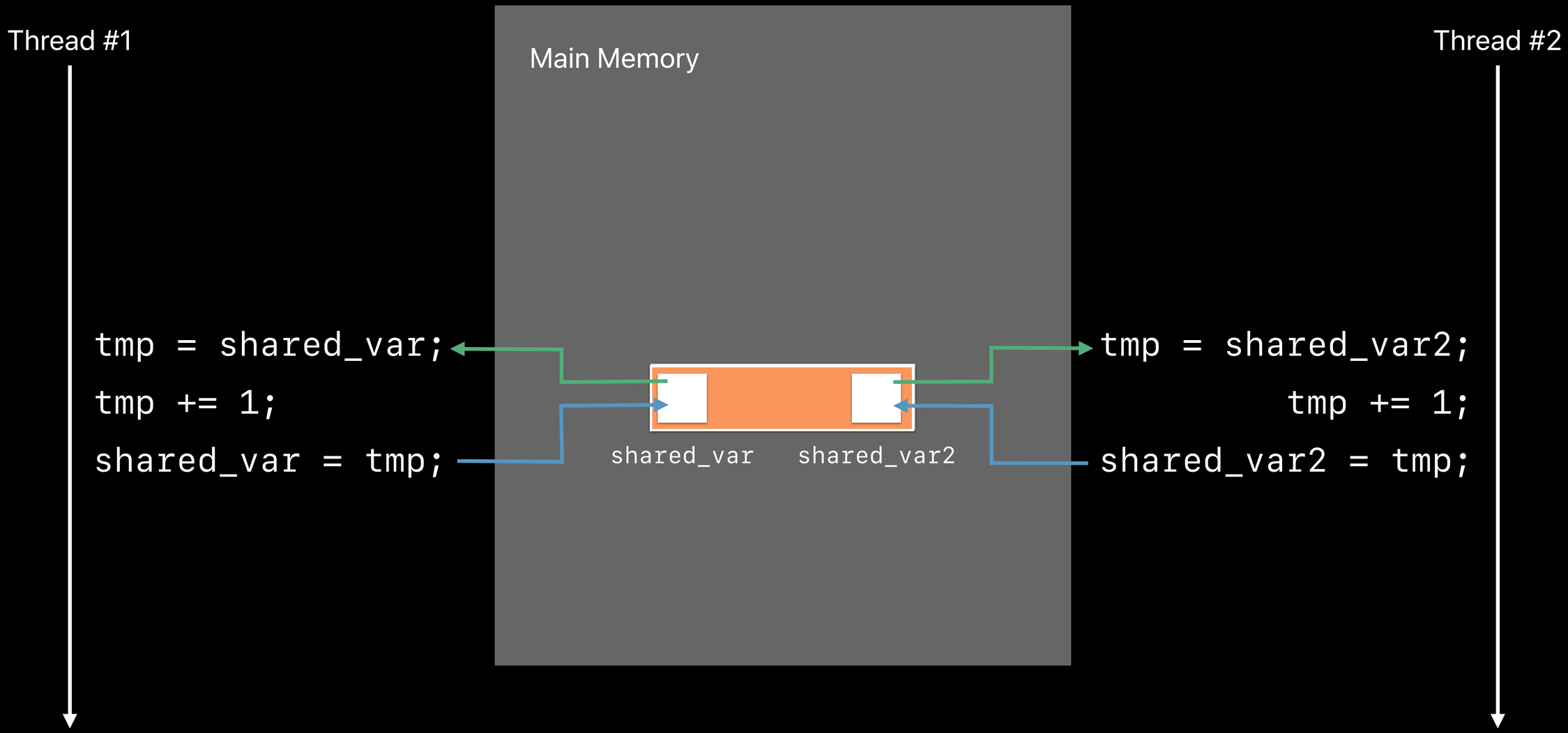
Atomics

ARM v8



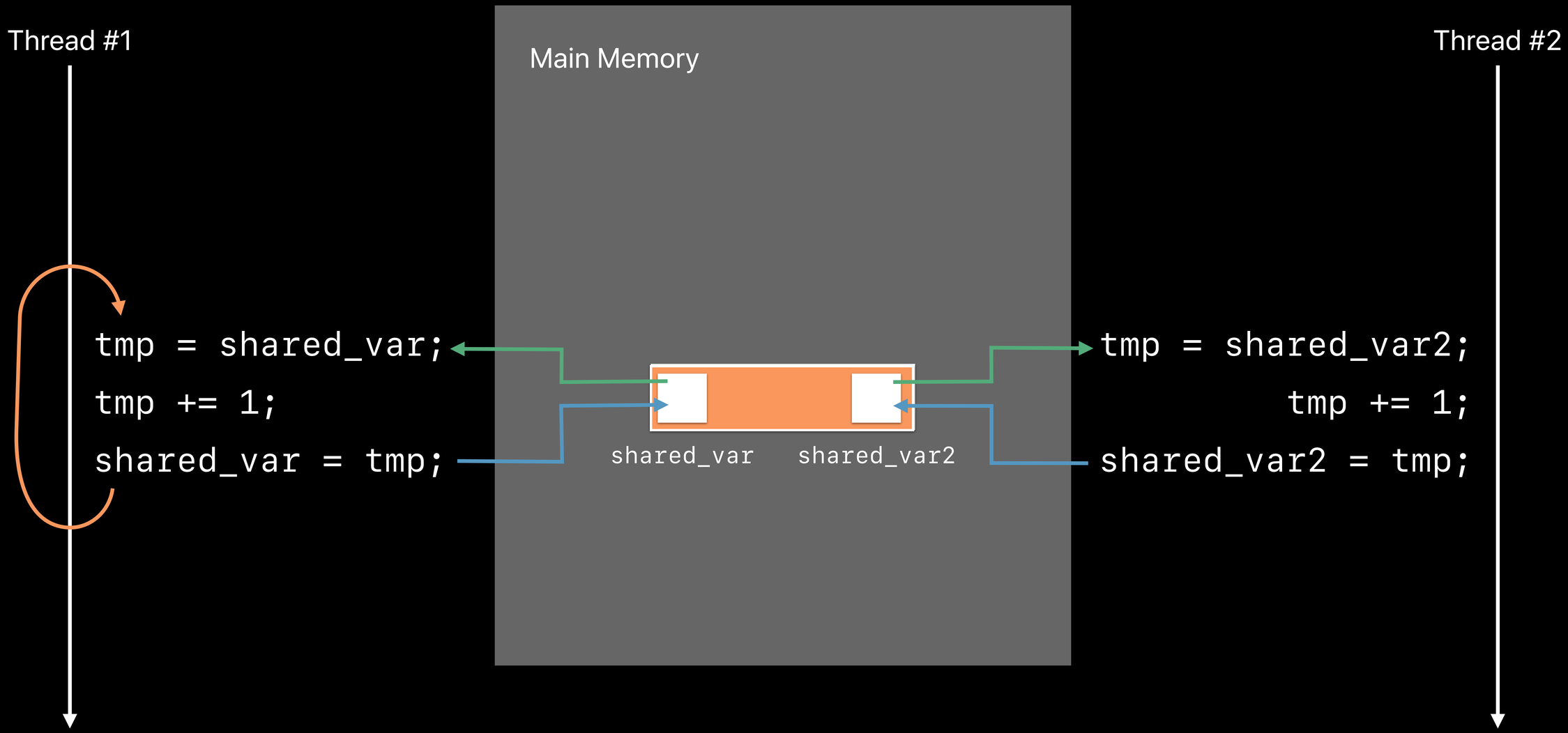
Atomics

ARM v8



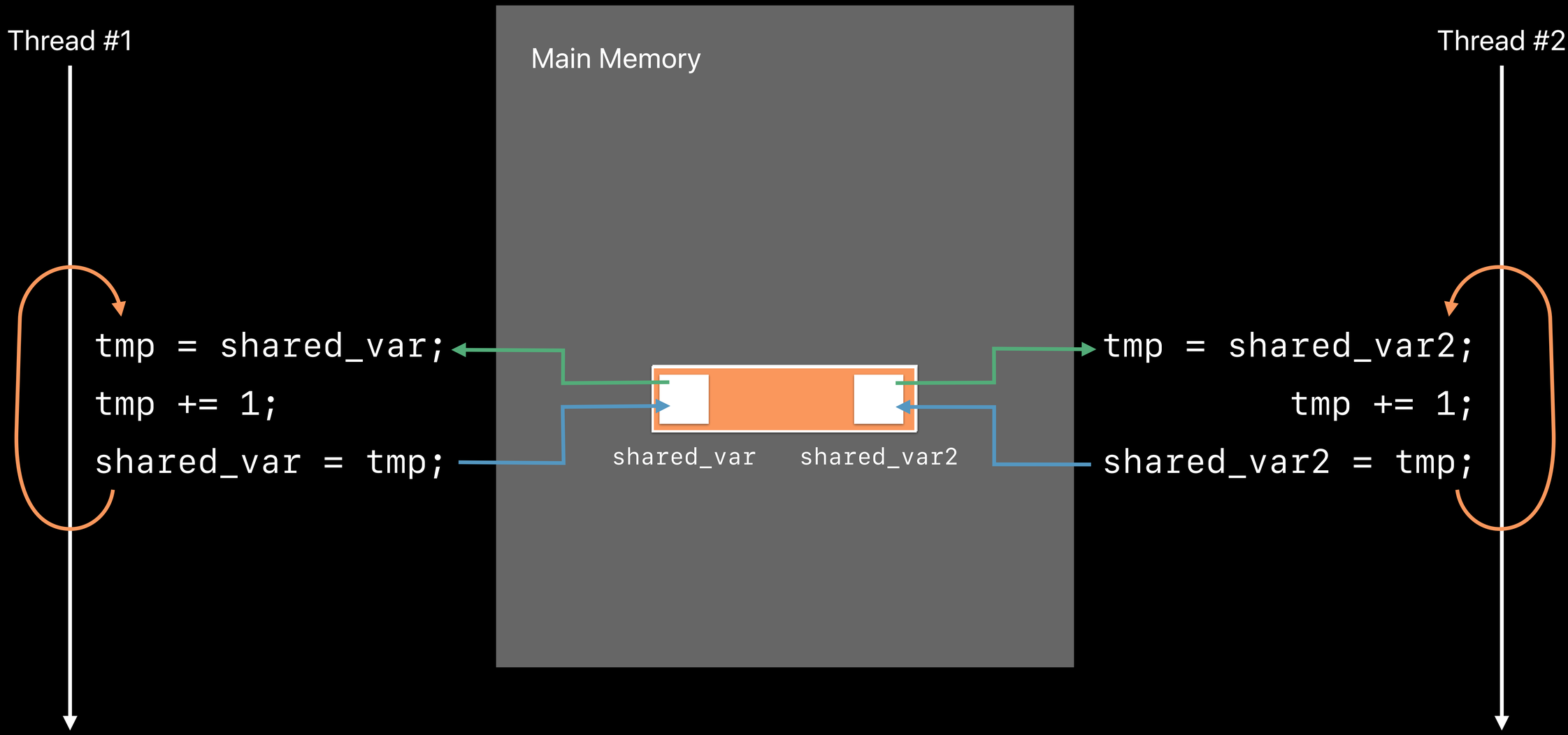
Atomics

ARM v8



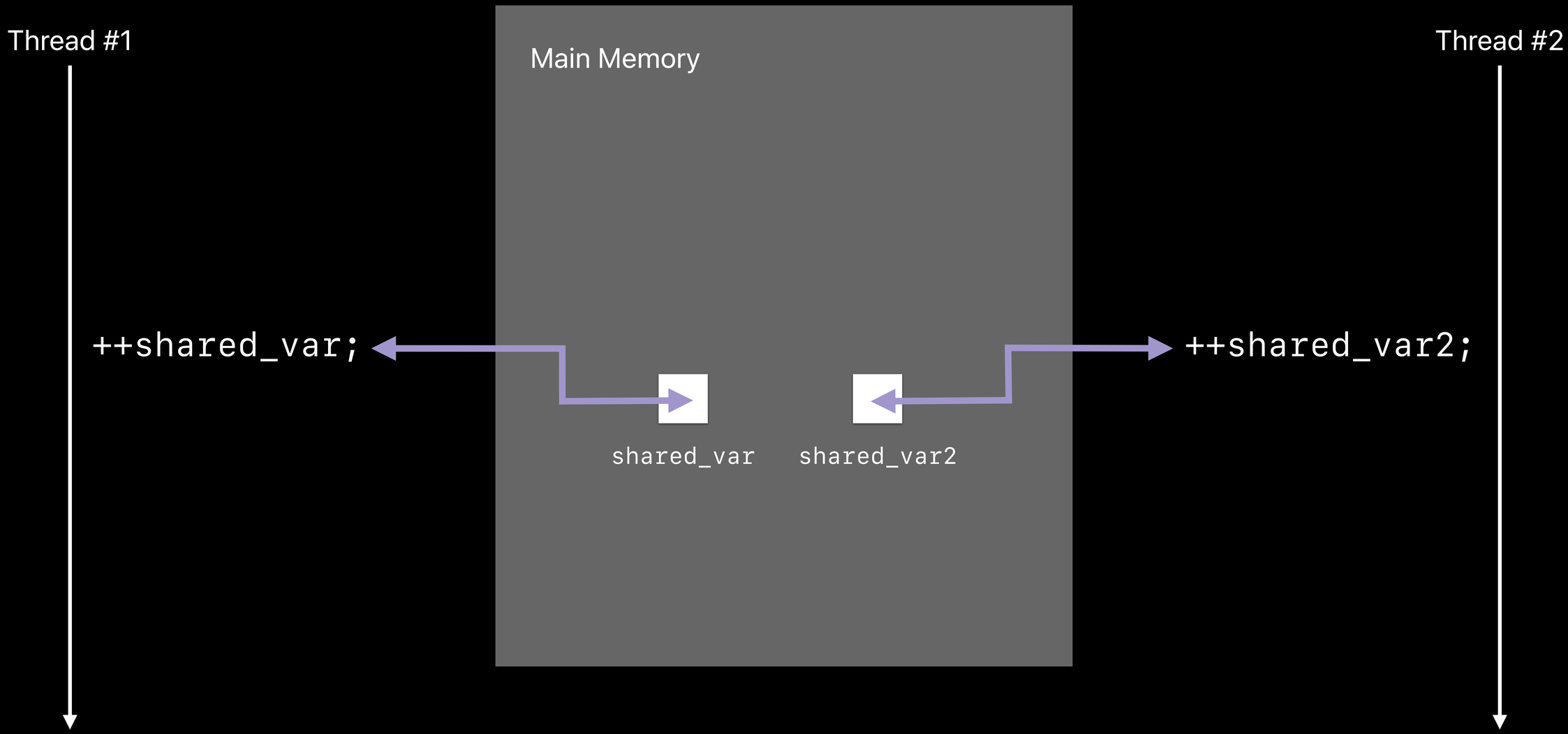
Atomics

ARM v8



ARM v8.1 Atomics

NEW



ARM v8.1 Atomics

Considerations

ARM v8.1 Atomics

Considerations

Only useful for raw C11/C++11 atomics

ARM v8.1 Atomics

Considerations

Only useful for raw C11/C++11 atomics

Use GCD, PThread, `os_unfair_lock`, …!

AVX-512

ARMv8.1 Atomics

ARMv8.2 16-Bit Float

AVX-512

ARMv8.1 Atomics

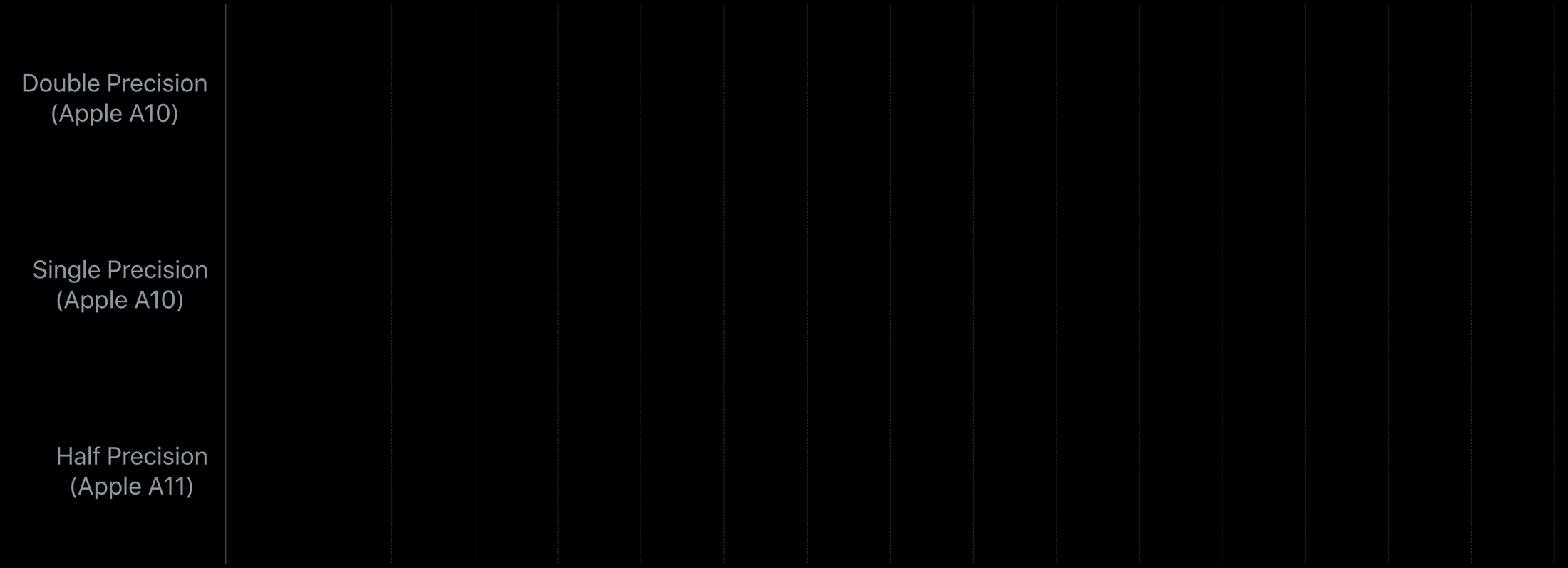
ARMv8.2 16-Bit Float

16-Bit Floating Point

ARM v8.2

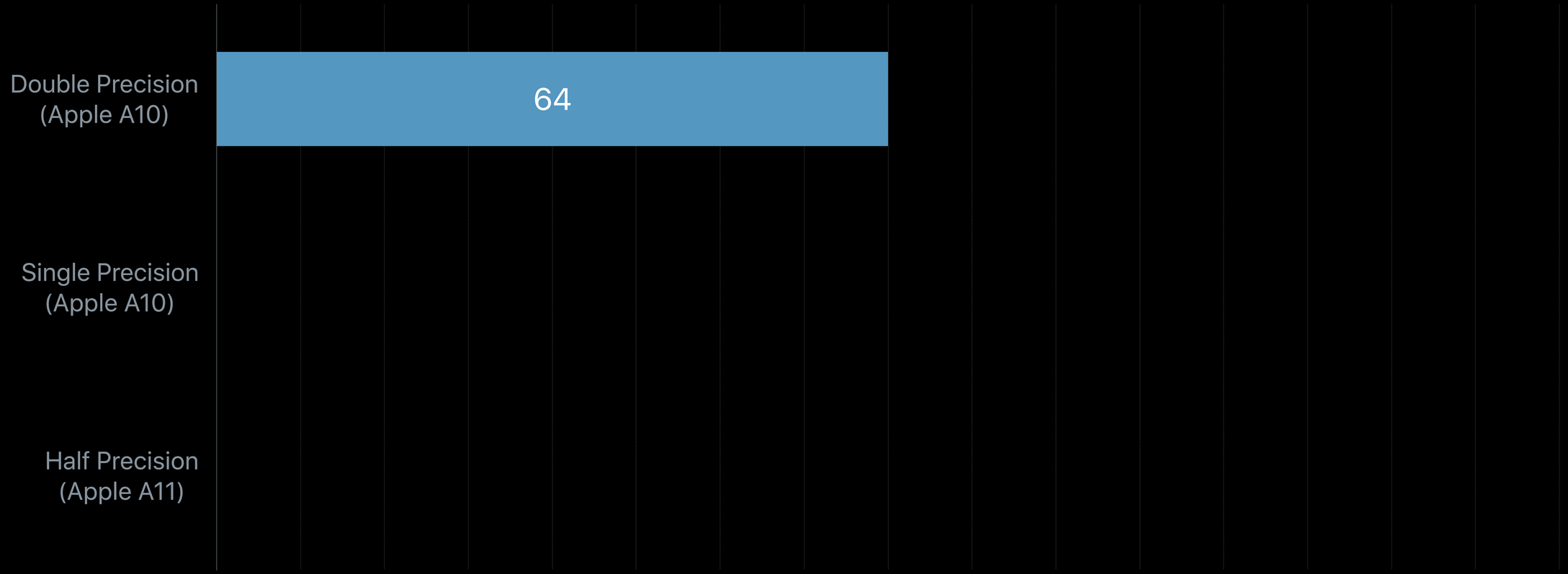
16-Bit Floating Point

ARM v8.2



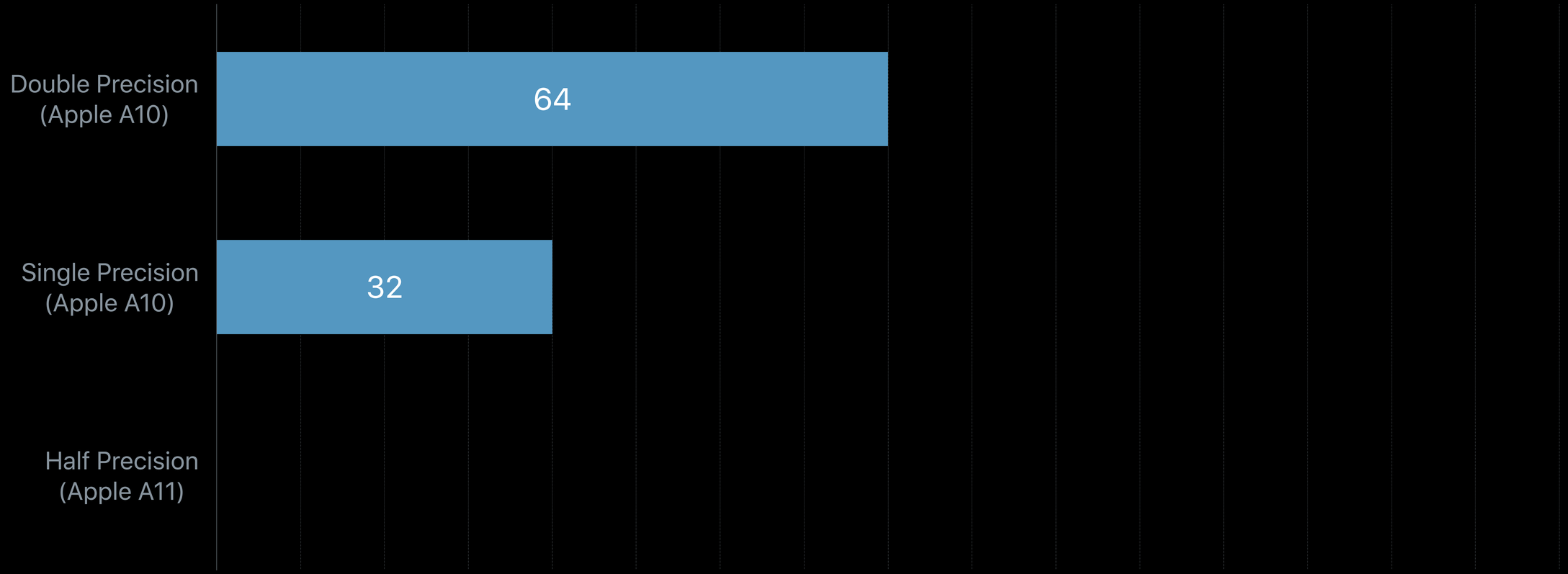
16-Bit Floating Point

ARM v8.2



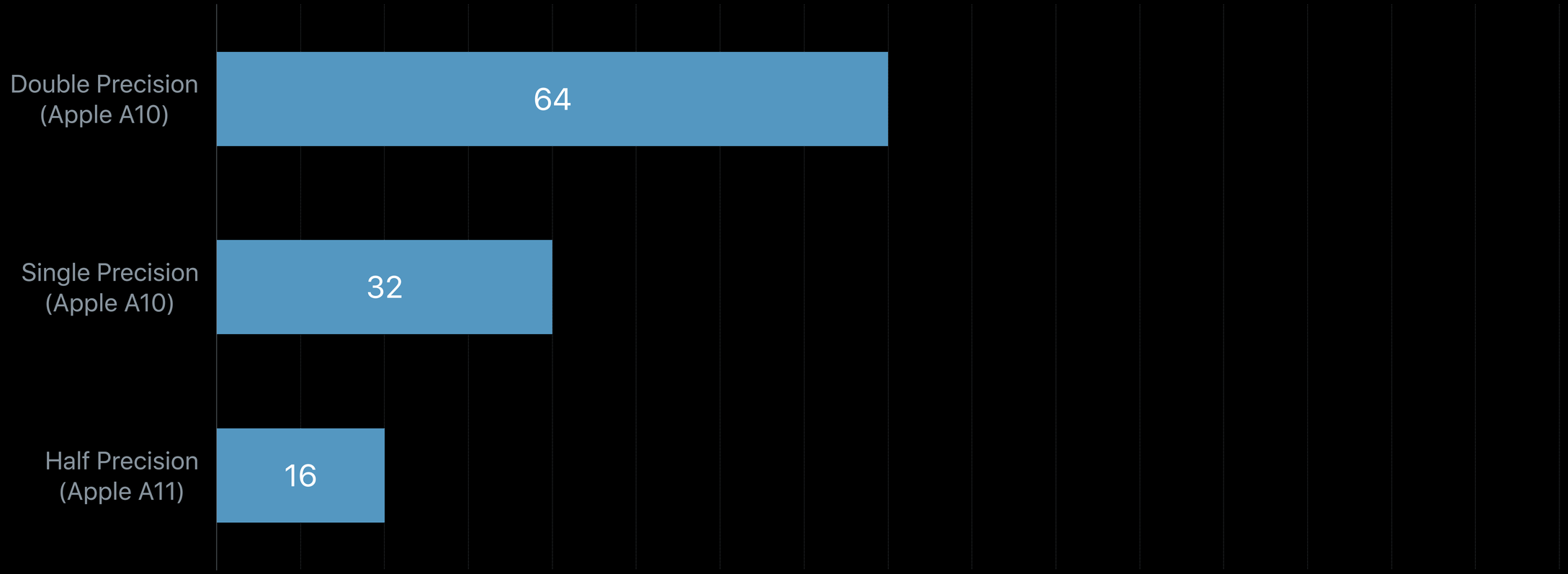
16-Bit Floating Point

ARM v8.2



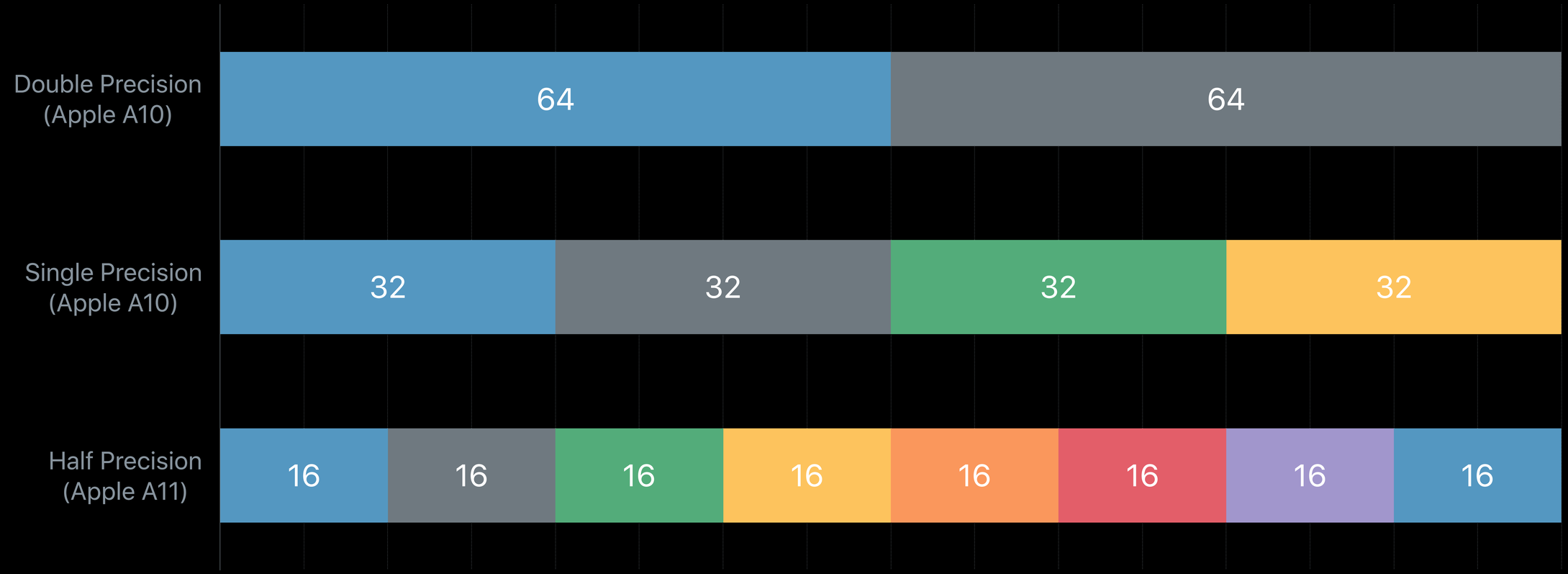
16-Bit Floating Point

ARM v8.2



16-Bit Floating Point

ARM v8.2



AVX-512

ARMv8.1 Atomics

ARMv8.2 16-Bit Float

AVX-512

ARMv8.1 Atomics
ARMv8.2 16-Bit Float

Detecting Instruction Set Extensions

Detecting Instruction Set Extensions

New instructions are not available everywhere

Detecting Instruction Set Extensions

New instructions are not available everywhere

Detect available extensions using `sysctlbyname`

Detecting Instruction Set Extensions

New instructions are not available everywhere

Detect available extensions using `sysctlbyname`

Or just let system frameworks handle this for you!

New Instruction Set Extensions

Summary

New Instruction Set Extensions

Summary

	Supported On	Type	`__target__ march=`	Clang Flag	`sysctlbyname`

New Instruction Set Extensions

Summary

	Supported On	Type	`__target__ march=`	Clang Flag	`sysctlbyname`
AVX-512 Vectors		`__m512`	`skylake-avx512`	`-march=skylake-avx512`	`hw.optional.avx512{f,bw,dq,vl,cd}`

New Instruction Set Extensions

Summary

	Supported On	Type	`__target__ march=`	Clang Flag	`sysctlbyname`
AVX-512 Vectors		`__m512`	`skylake-avx512`	`-march=skylake-avx512`	`hw.optional.avx512{f,bw,dq,vl,cd}`
ARM v8.1 Atomics		`_Atomic`	`monsoon`	`-mcpu=monsoon`	`hw.optional.armv8_1_atomics`

New Instruction Set Extensions

Summary

	Supported On	Type	`__target__ march=`	Clang Flag	`sysctlbyname`
AVX-512 Vectors		`__m512`	`skylake-avx512`	`-march=skylake-avx512`	`hw.optional.avx512{f,bw,dq,vl,cd}`
ARM v8.1 Atomics		`_Atomic`	`monsoon`	`-mcpu=monsoon`	`hw.optional.armv8_1_atomics`
ARM v8.2 Float 16		`_Float16`	`monsoon`	`-mcpu=monsoon`	`hw.optional.neon_fp16*`

* Available in a future Beta

ARC Object Pointers in C Structs

Autoreleasing Misuse Analyzer Check

Missing Noescape Annotation

ARM v8.1 Atomics

AVX-512

Suspicious Pragma Pack

Better Static Analyzer Performance

Stack Checking

GCD Anti-pattern Analyzer Check

ARM v8.2 16-Bit Float

ARC Object Pointers in C Structs

Autoreleasing Misuse Analyzer Check

Missing Noescape Annotation

ARM v8.1 Atomics

C++17 `std::optional`

AVX-512

Suspicious Pragma Pack

Faster Compiles for Huge/Generated Functions

Better Static Analyzer Performance

Stack Checking

C++17 `std::variant`

C++17 `std::any`

GCD Anti-pattern Analyzer Check

ARM v8.2 16-Bit Float

More Information

https://developer.apple.com/wwdc18/409

LLVM Compiler, Objective-C, and C++ Lab	Technology Lab 10	Thursday 3:00PM


WWDC18