Model Checking Boot Code in AWS Data Centers

Kareem Khazem

We proved the memory safety of boot code running in AWS data centers (*CAV 2018*).

Byron Cook

Kareem Khazem

Daniel Kroening

Serdar Tasiran

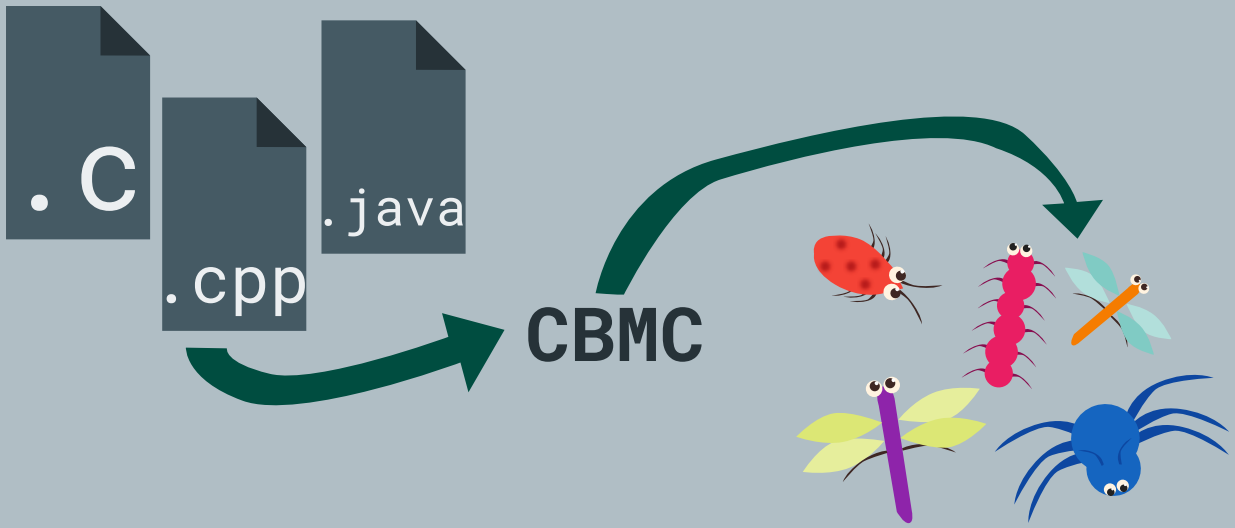
Michael Tautschnig

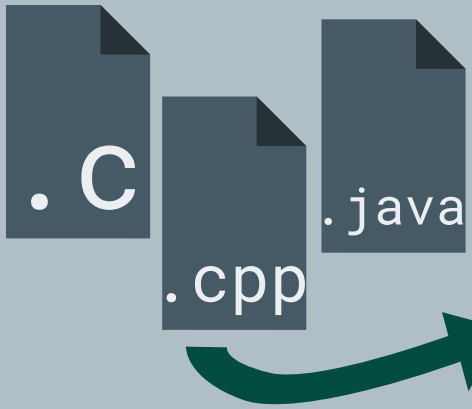
Mark R. Tuttle

Today:

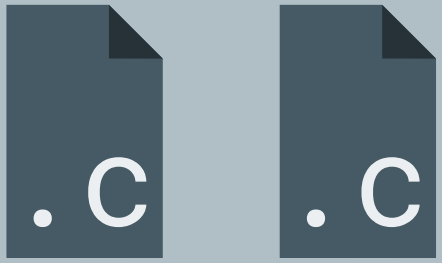
- C Bounded Model Checker
- Boot code & safety
- Challenges
- Solutions

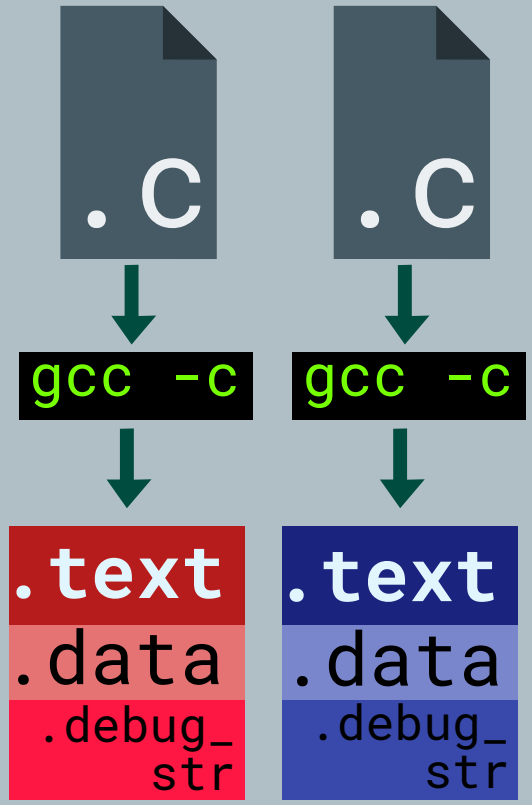
CBMC


.C
.cpp
.java
CBMC

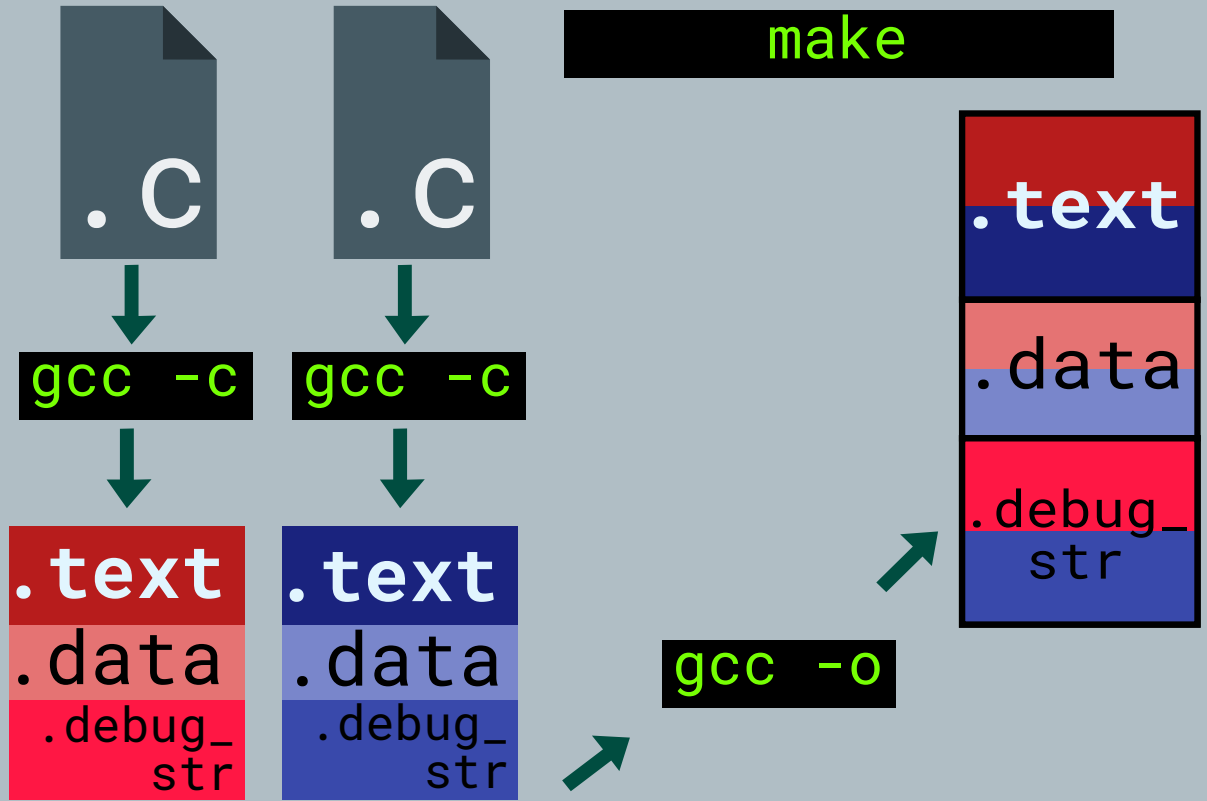


2017 SV-COMP
(falsification)

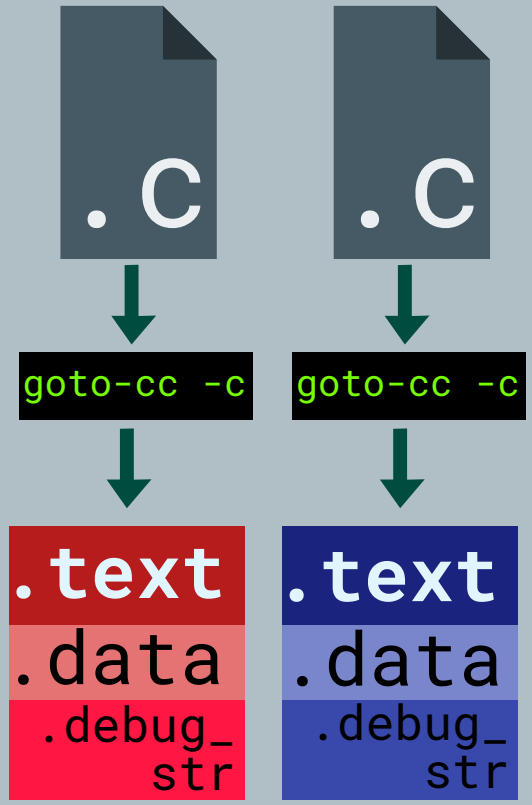

.c
.c



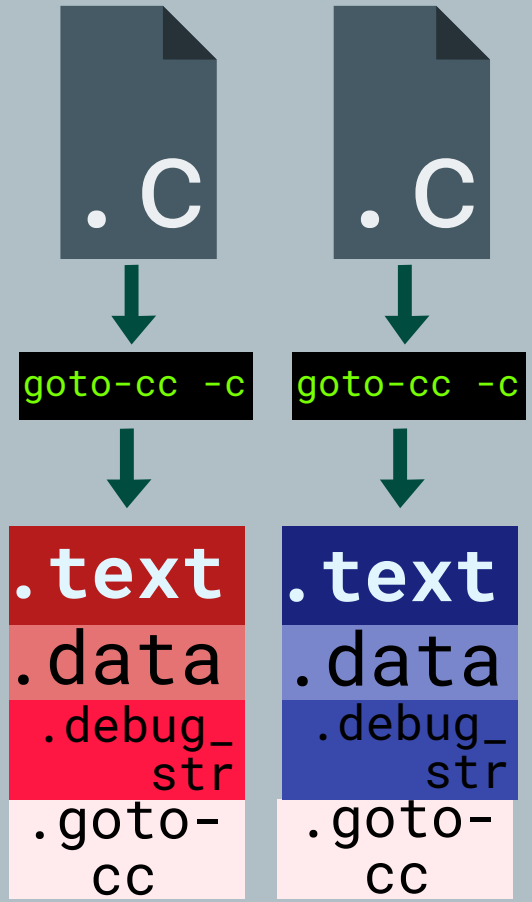
make


make
.c
.c
gcc -c
gcc -c
.text
.data
.debug_
str
.text
.data
.debug_
str
gcc -o
.text
.data
.debug_
str

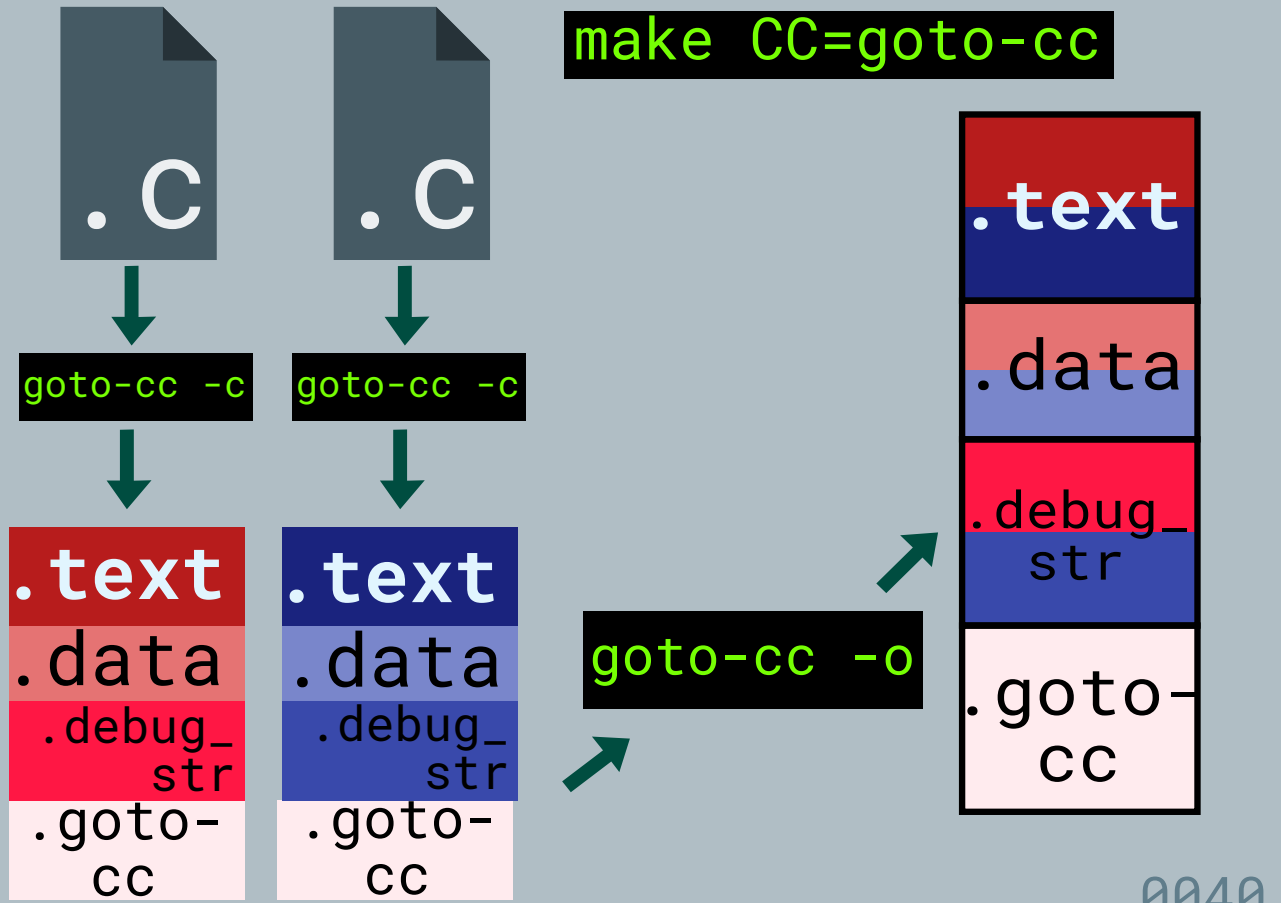

```
make CC=goto-cc
```

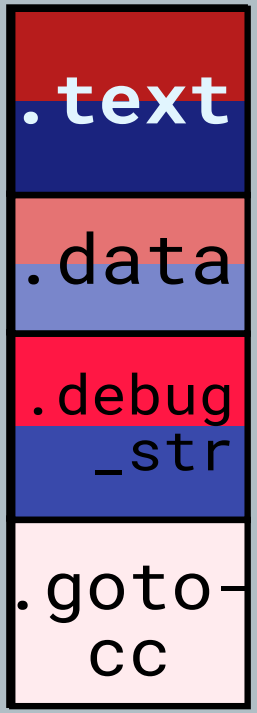


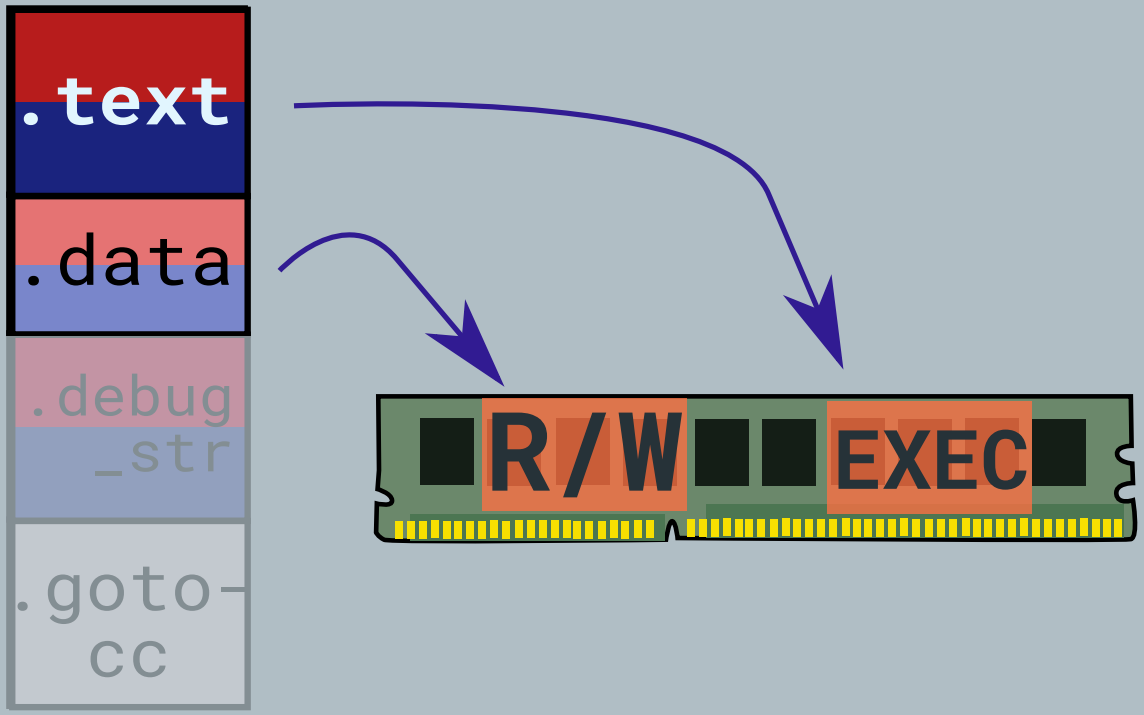

```
make CC=goto-cc
```

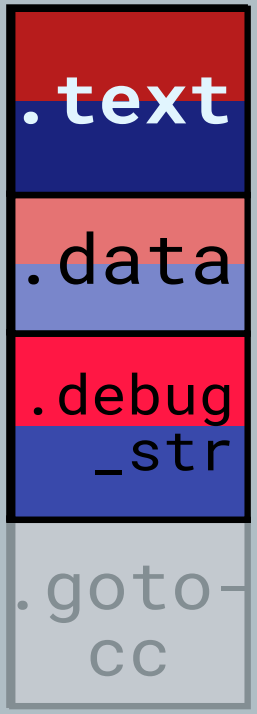



```
make CC=goto-cc
```

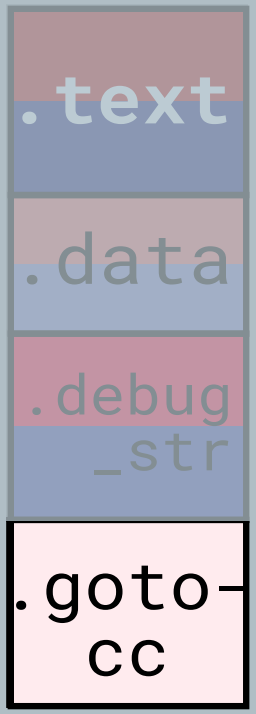

make CC=goto-cc
.c
.c
goto-cc -c
goto-cc -c
.text
.data
.debug_str
.goto-cc
.text
.data
.debug_str
.goto-cc
goto-cc -o
.text
.data
.debug_str
.goto-cc

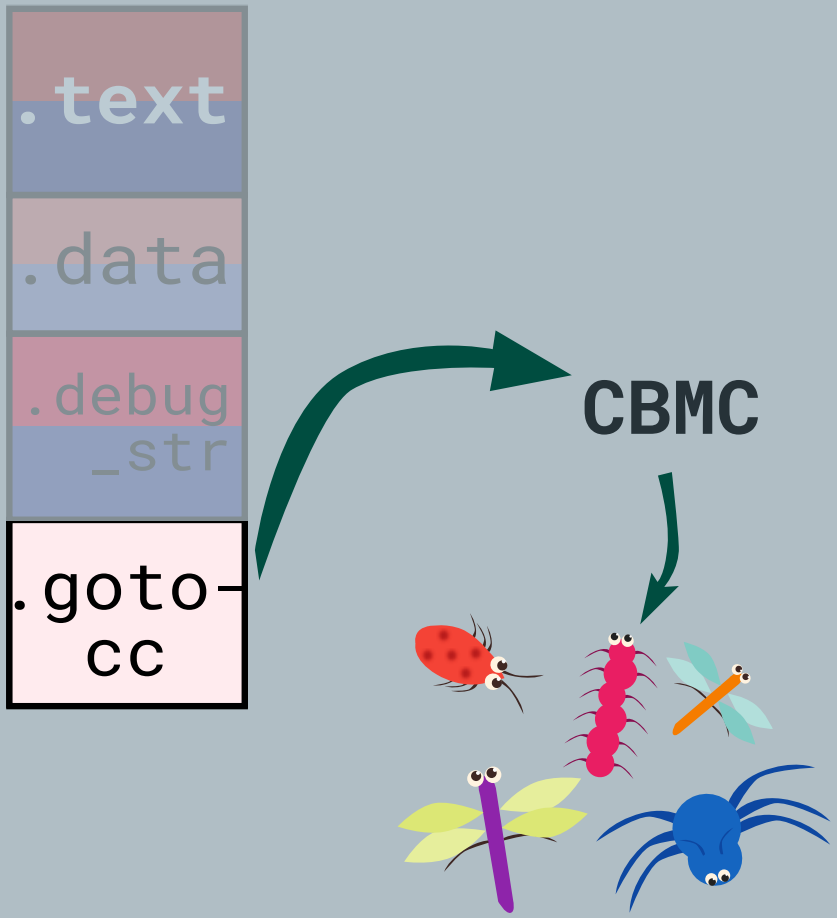

.text
.data
.debug
_str
.goto-
cc

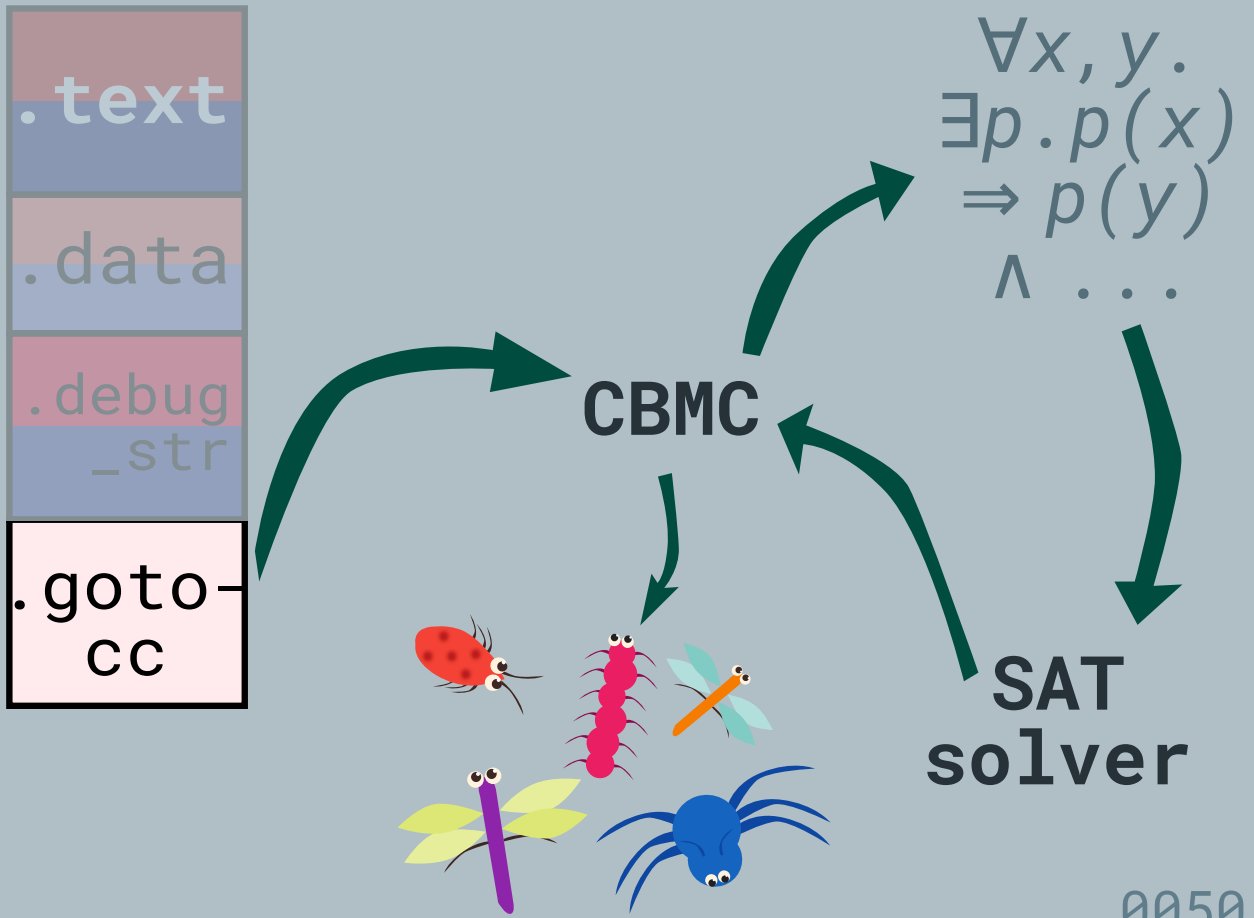

.text
.data
.debug
_str
.goto-
cc
R/W
EXEC



LLDB,
GDB,
...


.text
.data
.debug
_str
.goto-
cc


.text
.data
.debug
_str
.goto-
cc
CBMC


.text
.data
.debug
_str
.goto-cc
CBMC
∀x,y.
∃p.p(x)
⇒ p(y)
∧ ...
SAT
solver

```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

$flag_1 = \bot$

$\wedge\ x_1 = \bot$

```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

$flag_1 = \bot$

$\wedge\ x_1 = \bot$

$\wedge\ a_1 = [0, 0, 0]$

```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

$$\begin{aligned} & \mathrm{flag}_1 = \bot \\ \wedge\ & x_1 = \bot \\ \wedge\ & a_1 = [0,\ 0,\ 0] \\ \wedge\ & y_1 = 2 \end{aligned}$$

```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

$$
\begin{aligned}
& flag_1 = \bot \\
\wedge\ & x_1 = \bot \\
\wedge\ & a_1 = [0,\ 0,\ 0] \\
\wedge\ & y_1 = 2 \\
\wedge\ & x_2 = x_1 + y_1
\end{aligned}
$$

```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

$$
\begin{aligned}
& flag_1 = \bot \\
\wedge\ & x_1 = \bot \\
\wedge\ & a_1 = [0,\ 0,\ 0] \\
\wedge\ & y_1 = 2 \\
\wedge\ & x_2 = x_1 + y_1 \\
\wedge\ & (flag_1 \\
& \quad \wedge\ z_1 = a_1[y_1] \\
& \qquad\qquad)
\end{aligned}
$$

```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

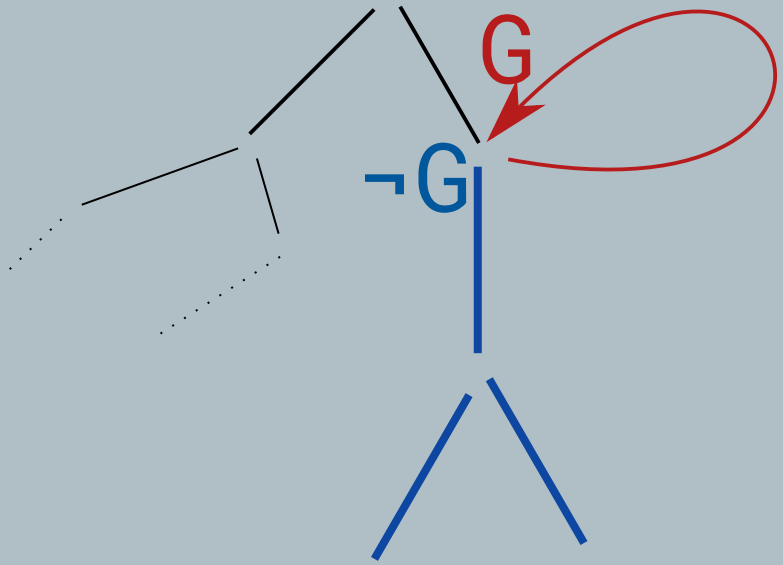
$$
\begin{aligned}
& \mathrm{flag}_1 = \bot \\
\wedge\ & x_1 = \bot \\
\wedge\ & a_1 = [0,\ 0,\ 0] \\
\wedge\ & y_1 = 2 \\
\wedge\ & x_2 = x_1 + y_1 \\
\wedge\ & (\mathrm{flag}_1 \\
& \quad \wedge\ z_1 = a_1[y_1] \\
& \vee\ \neg \mathrm{flag}_1 \\
& \quad \wedge\ z_1 = a_1[x_2])
\end{aligned}
$$

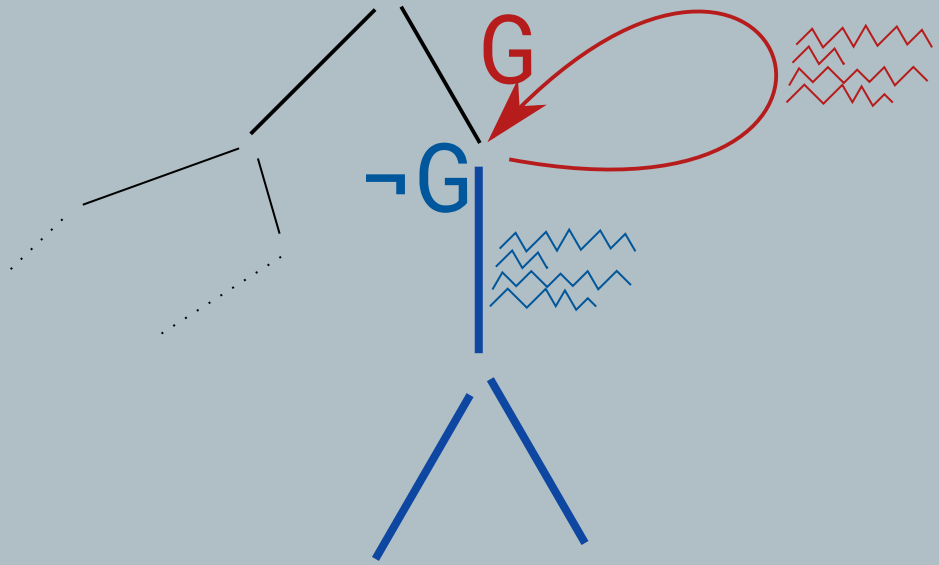
```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

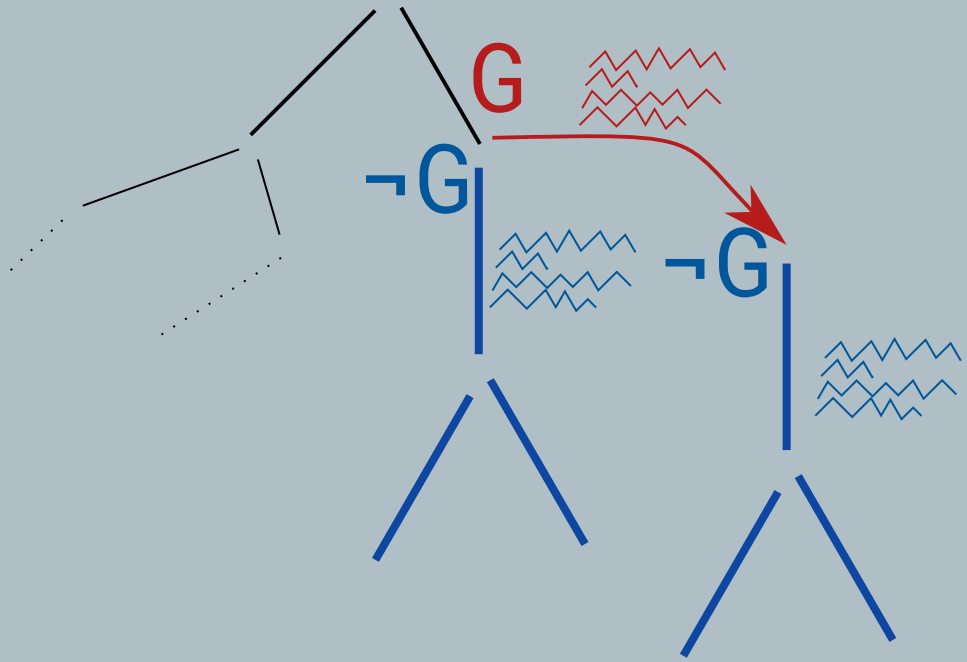
$$
\begin{aligned}
& \mathit{flag}_1 = \bot \\
\wedge\ & x_1 = \bot \\
\wedge\ & a_1 = [0,\ 0,\ 0] \\
\wedge\ & y_1 = 2 \\
\wedge\ & x_2 = x_1 + y_1 \\
\wedge\ & (\mathit{flag}_1 \\
& \quad \wedge z_1 = a_1[y_1] \\
& \vee \neg \mathit{flag}_1 \\
& \quad \wedge z_1 = a_1[x_2]) \\
\wedge\ & (\mathit{bounds\text{-}violated} \\
& \vee \mathit{arith\text{-}overflow}) \\
& \vee \ldots)
\end{aligned}
$$

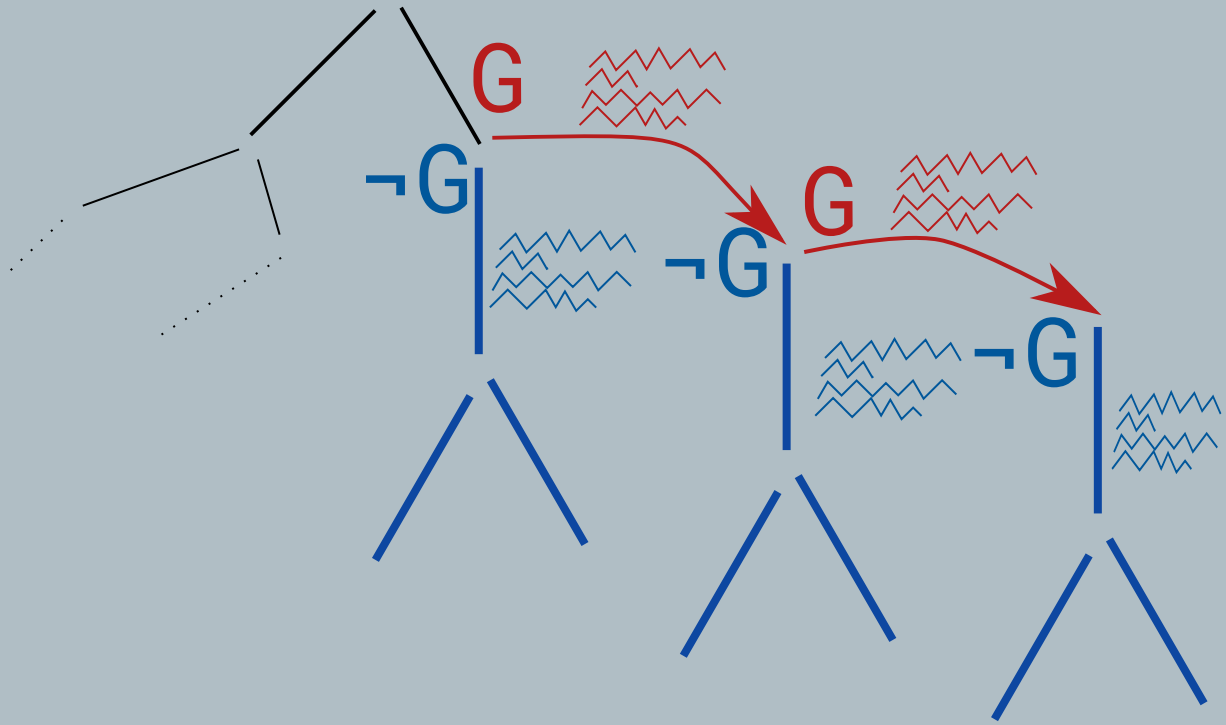
```
int foo(int flag,
        int x)
{
 int a[3] = {0};
 int y = 2;
 x = x + y;
 if(flag)
  int z = a[y];
 else
  int z = a[x];
}
```

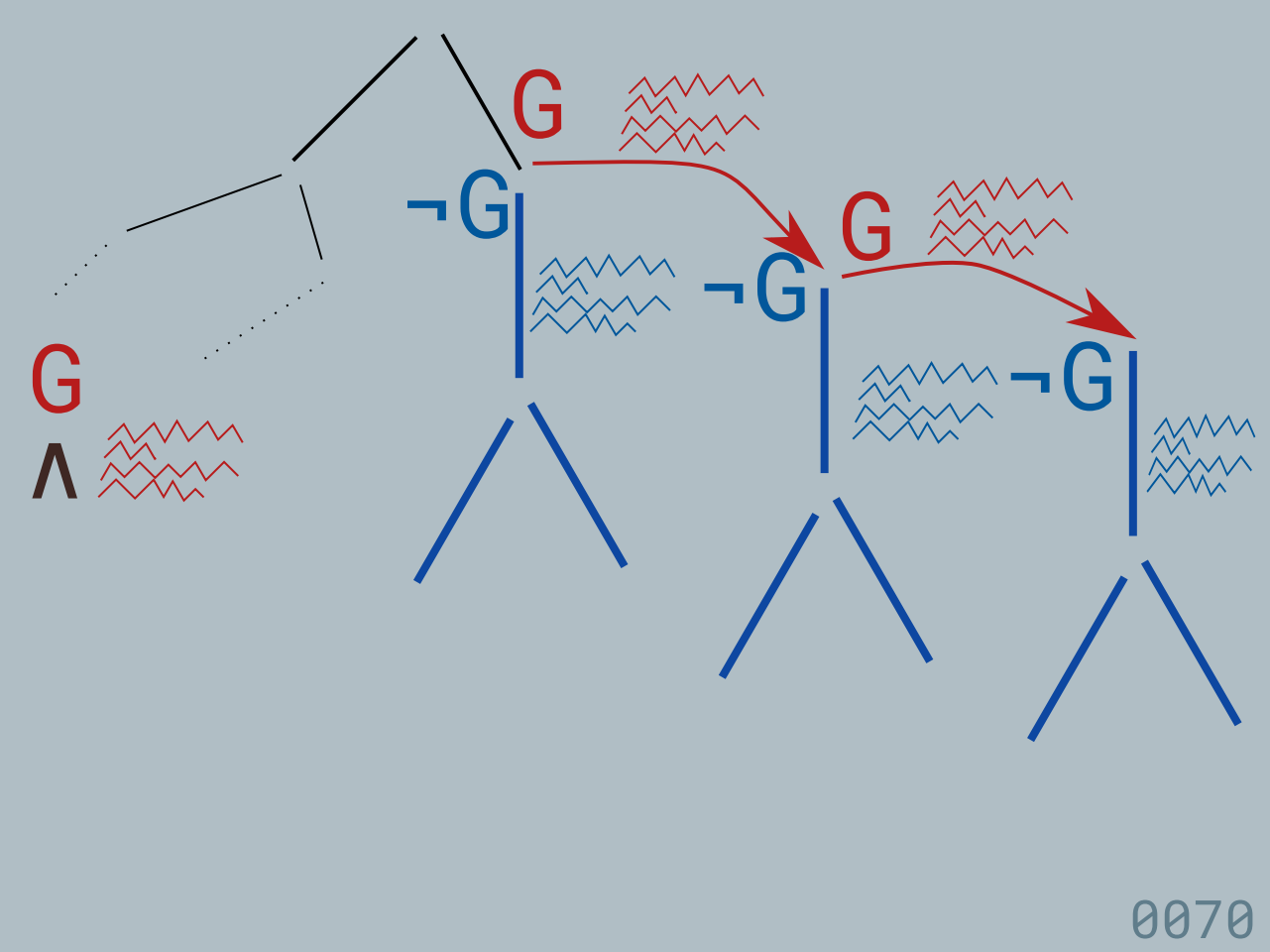
$$
\begin{aligned}
& \mathtt{flag}_1 = \mathbf{1} \\
\wedge\ & x_1 = \mathbf{1} \\
\wedge\ & a_1 = [0,\ 0,\ 0] \\
\wedge\ & y_1 = 2 \\
\wedge\ & x_2 = x_1 + y_1 \\
\wedge\ & (\mathtt{flag}_1 \\
& \quad \wedge z_1 = a_1[y_1] \\
& \ \vee \neg\mathbf{flag}_1 \\
& \quad \wedge\ \mathbf{z_1 = a_1[x_2]}) \\
\wedge\ & (\mathbf{bounds\text{-}violated} \\
& \ \vee \mathtt{arith\text{-}overflow}) \\
& \ \vee \ldots)
\end{aligned}
$$

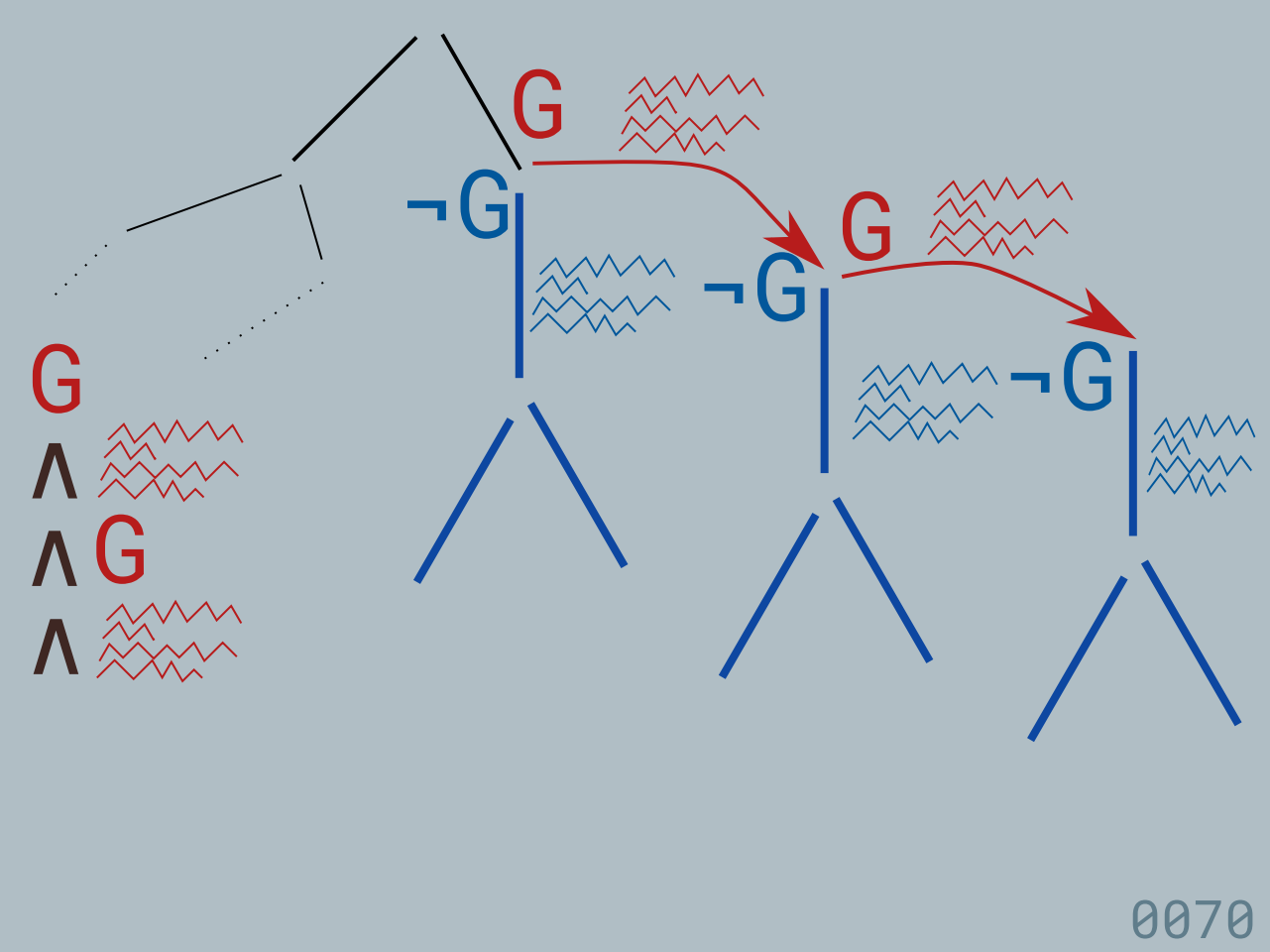

G
¬G

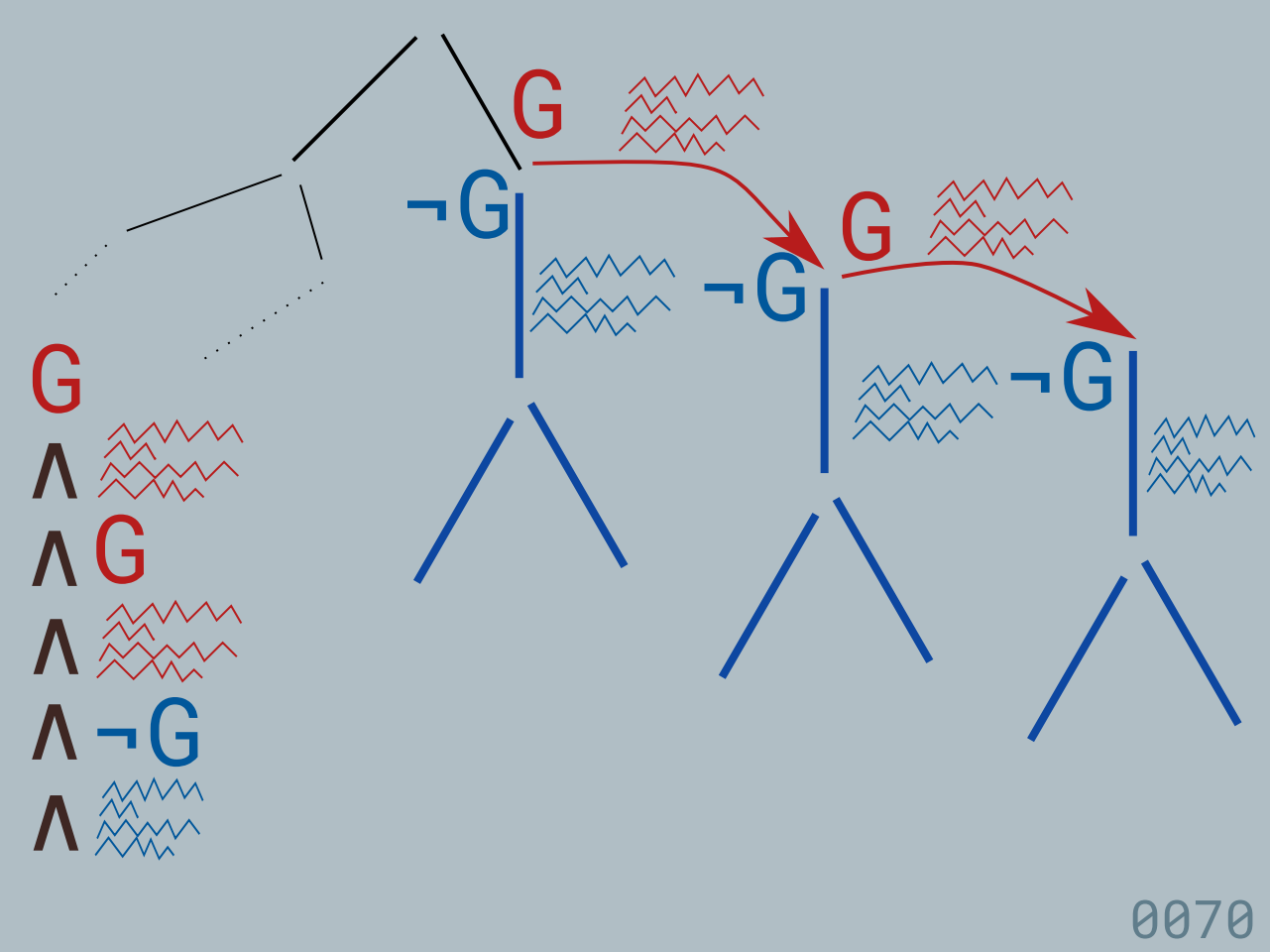

G
¬G

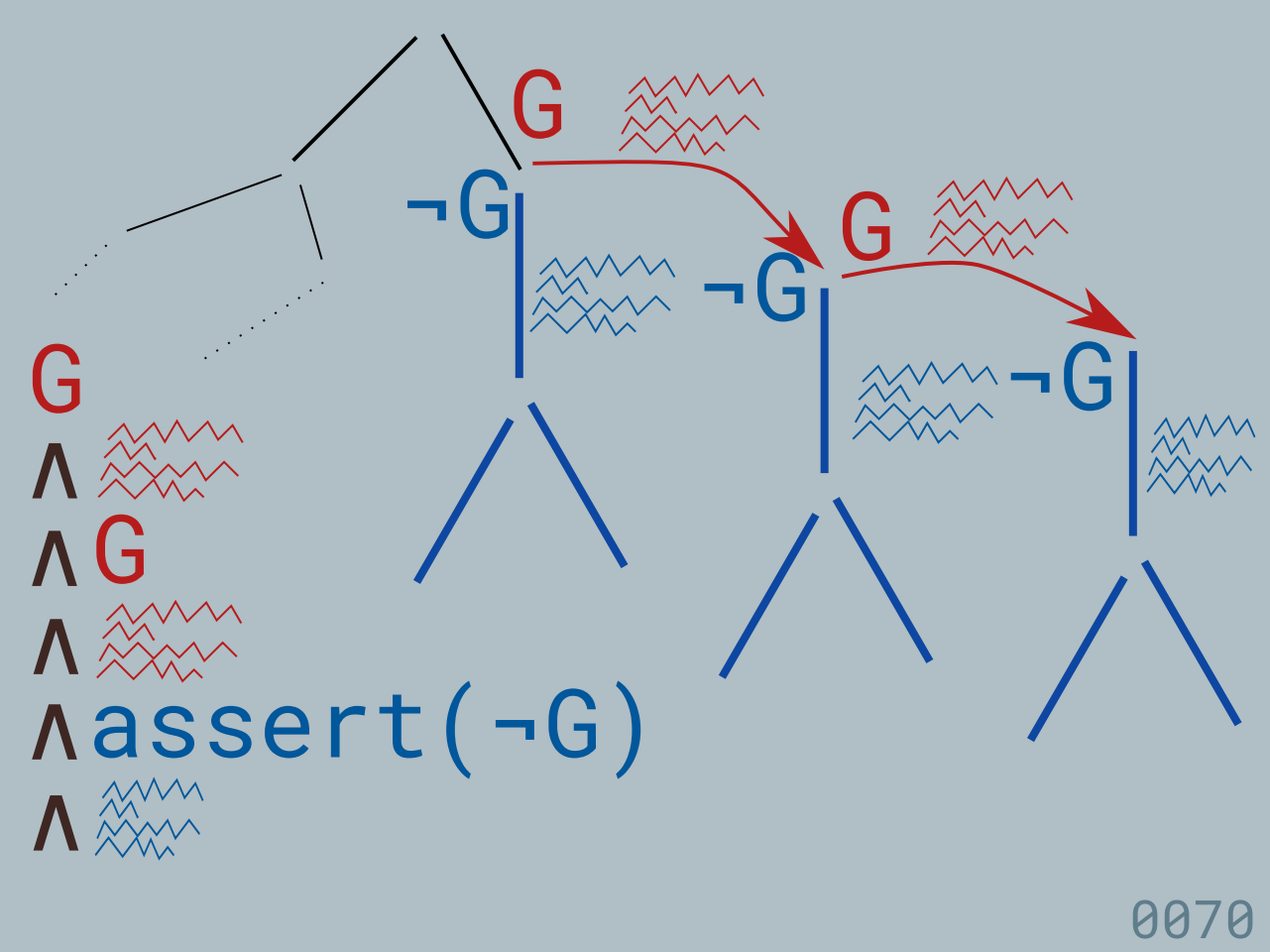

G
¬G
¬G

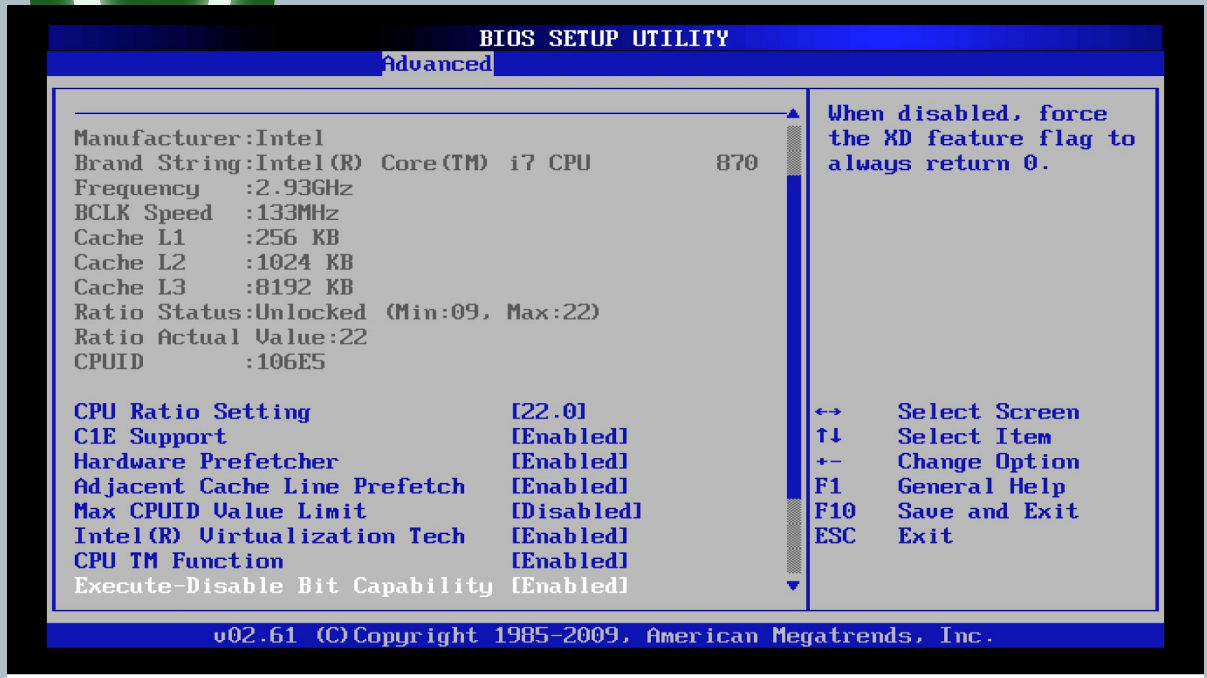

G
¬G
G
¬G
¬G


G
Λ
G
¬G
G
¬G
¬G


G
¬G
G
¬G
G
¬G
G
∧
∧G
∧


G
¬G
G
¬G
¬G
G
∧
∧G
∧
∧¬G
∧

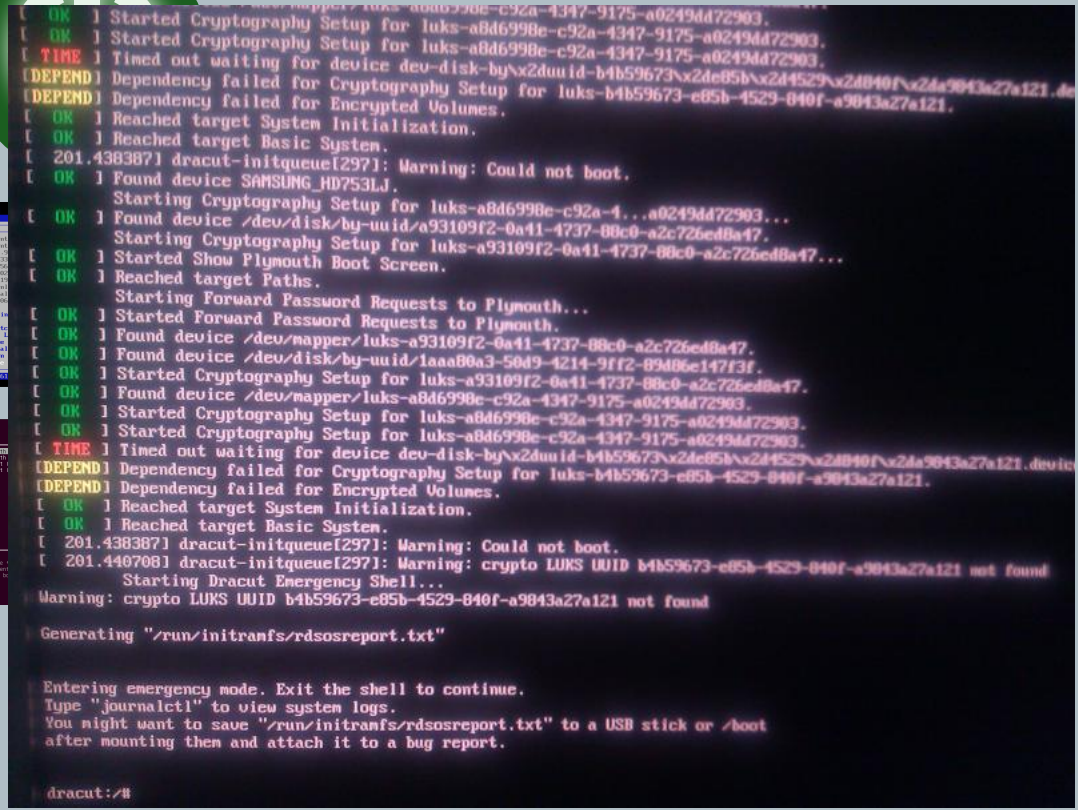

G
¬G
G
¬G
G
¬G
∧
∧G
∧
∧assert(¬G)
∧

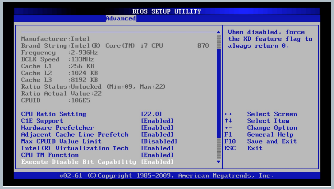

BIOS SETUP UTILITY
Advanced
Manufacturer:Intel
Brand String:Intel(R) Core(TM) i7 CPU 870
Frequency :2.93GHz
BCLK Speed :133MHz
Cache L1 :256 KB
Cache L2 :1024 KB
Cache L3 :8192 KB
Ratio Status:Unlocked (Min:09, Max:22)
Ratio Actual Value:22
CPUID :106E5
CPU Ratio Setting [22.0]
C1E Support [Enabled]
Hardware Prefetcher [Enabled]
Adjacent Cache Line Prefetch [Enabled]
Max CPUID Value Limit [Disabled]
Intel(R) Virtualization Tech [Enabled]
CPU TM Function [Enabled]
Execute-Disable Bit Capability [Enabled]
When disabled, force the XD feature flag to always return 0.
↔ Select Screen
↑↓ Select Item
+- Change Option
F1 General Help
F10 Save and Exit
ESC Exit
v02.61 (C)Copyright 1985-2009, American Megatrends, Inc.

GNU GRUB version 1.99-21ubuntu3

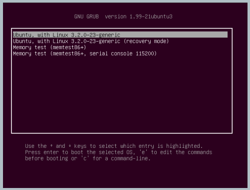
Ubuntu, with Linux 3.2.0-23-generic
Ubuntu, with Linux 3.2.0-23-generic (recovery mode)
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.

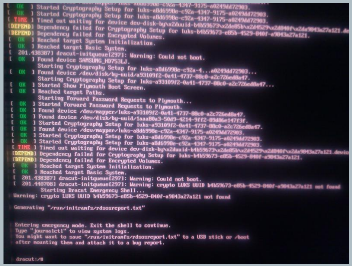

[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[TIME] Timed out waiting for device dev-disk-by\x2duuid-b4b59673\x2de85b\x2d4529\x2d840f\x2da9043a27a121.de
[DEPEND] Dependency failed for Cryptography Setup for luks-b4b59673-e85b-4529-840f-a9043a27a121.
[DEPEND] Dependency failed for Encrypted Volumes.
[OK] Reached target System Initialization.
[OK] Reached target Basic System.
[201.438387] dracut-initqueue[297]: Warning: Could not boot.
[OK] Found device SAMSUNG_HD753LJ.
Starting Cryptography Setup for luks-a8d6998e-c92a-4...a0249dd72903...
[OK] Found device /dev/disk/by-uuid/a93109f2-0a41-4737-88c0-a2c726ed8a47.
Starting Cryptography Setup for luks-a93109f2-0a41-4737-88c0-a2c726ed8a47...
[OK] Started Show Plymouth Boot Screen.
[OK] Reached target Paths.
Starting Forward Password Requests to Plymouth...
[OK] Started Forward Password Requests to Plymouth.
[OK] Found device /dev/mapper/luks-a93109f2-0a41-4737-88c0-a2c726ed8a47.
[OK] Found device /dev/disk/by-uuid/1aaa80a3-50d9-4214-9ff2-89486e147f3f.
[OK] Started Cryptography Setup for luks-a93109f2-0a41-4737-88c0-a2c726ed8a47.
[OK] Found device /dev/mapper/luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[TIME] Timed out waiting for device dev-disk-by\x2duuid-b4b59673\x2de85b\x2d4529\x2d840f\x2da9043a27a121.devic
[DEPEND] Dependency failed for Cryptography Setup for luks-b4b59673-e85b-4529-840f-a9043a27a121.
[DEPEND] Dependency failed for Encrypted Volumes.
[OK] Reached target System Initialization.
[OK] Reached target Basic System.
[201.438387] dracut-initqueue[297]: Warning: Could not boot.
[201.440708] dracut-initqueue[297]: Warning: crypto LUKS UUID b4b59673-e85b-4529-840f-a9043a27a121 not found
Starting Dracut Emergency Shell...
Warning: crypto LUKS UUID b4b59673-e85b-4529-840f-a9043a27a121 not found
Generating "/run/initramfs/rdsosreport.txt"
Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.
dracut:/#



BIOS SETUP UTILITY
Advanced
Manufacturer:Intel
Brand String:Intel(R) Core(TM) i7 CPU 870
Frequency :2.93GHz
BCLK Speed :133MHz
Cache L1 :256 KB
Cache L2 :1024 KB
Cache L3 :8192 KB
Ratio Status:Unlocked (Min:09, Max:22)
Ratio Actual Value:22
CPUID :106E5
CPU Ratio Setting [22.0]
C1E Support [Enabled]
Hardware Prefetcher [Enabled]
Adjacent Cache Line Prefetch [Enabled]
Max CPUID Value Limit [Disabled]
Intel(R) Virtualization Tech [Enabled]
CPU TM Function [Enabled]
Execute-Disable Bit Capability [Enabled]
When disabled, force the XD feature flag to always return 0.
Select Screen
Select Item
Change Option
F1 General Help
F10 Save and Exit
ESC Exit
v02.61 (C)Copyright 1985-2009, American Megatrends, Inc.

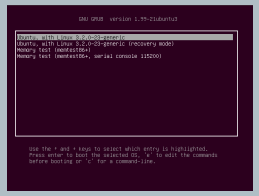


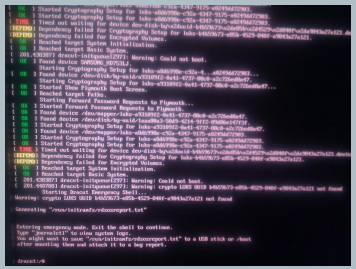
GNU GRUB version 1.99-21ubuntu3
Ubuntu, with Linux 3.2.0-23-generic
Ubuntu, with Linux 3.2.0-23-generic (recovery mode)
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the commands
before booting or 'c' for a command-line.

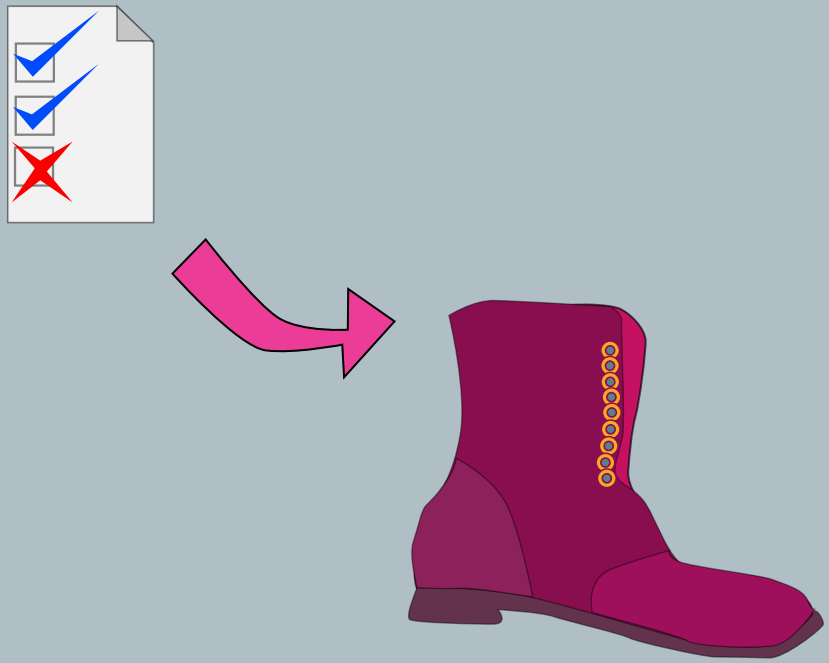


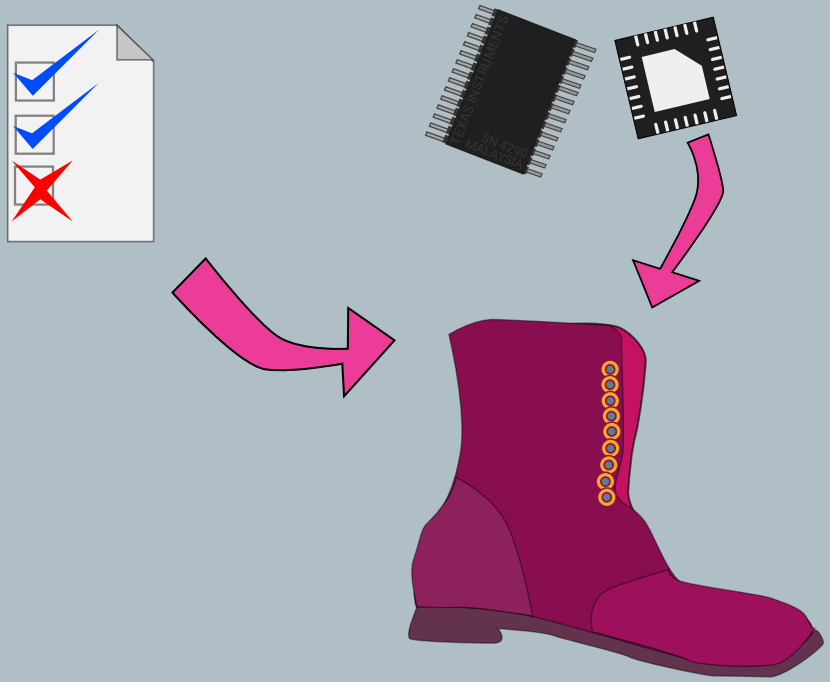
[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[TIME] Timed out waiting for device dev-disk-by\x2duuid-b4b59673\x2de85b\x2d4529\x2d840f\x2da9843a27a121.de
[DEPEND] Dependency failed for Cryptography Setup for luks-b4b59673-e85b-4529-840f-a9843a27a121.
[DEPEND] Dependency failed for Encrypted Volumes.
[OK] Reached target System Initialization.
[OK] Reached target Basic System.
[201.438387] dracut-initqueue[297]: Warning: Could not boot.
[OK] Found device SAMSUNG_HD753LJ.
Starting Cryptography Setup for luks-a8d6998e-c92a-4...a0249dd72903...
[OK] Found device /dev/disk/by-uuid/a93109f2-0a41-4737-88c0-a2c726ed8a47.
Starting Cryptography Setup for luks-a93109f2-0a41-4737-88c0-a2c726ed8a47...
[OK] Started Show Plymouth Boot Screen.
[OK] Reached target Paths.
Starting Forward Password Requests to Plymouth...
[OK] Started Forward Password Requests to Plymouth.
[OK] Found device /dev/mapper/luks-a93109f2-0a41-4737-88c0-a2c726ed8a47.
[OK] Found device /dev/disk/by-uuid/1aaa80a3-50d9-4214-9ff2-89d86e147f3f.
[OK] Started Cryptography Setup for luks-a93109f2-0a41-4737-88c0-a2c726ed8a47.
[OK] Found device /dev/mapper/luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[OK] Started Cryptography Setup for luks-a8d6998e-c92a-4347-9175-a0249dd72903.
[TIME] Timed out waiting for device dev-disk-by\x2duuid-b4b59673\x2de85b\x2d4529\x2d840f\x2da9843a27a121.device
[DEPEND] Dependency failed for Cryptography Setup for luks-b4b59673-e85b-4529-840f-a9843a27a121.
[DEPEND] Dependency failed for Encrypted Volumes.
[OK] Reached target System Initialization.
[OK] Reached target Basic System.
[201.438387] dracut-initqueue[297]: Warning: Could not boot.
[201.440708] dracut-initqueue[297]: Warning: crypto LUKS UUID b4b59673-e85b-4529-840f-a9843a27a121 not found
Starting Dracut Emergency Shell...
Warning: crypto LUKS UUID b4b59673-e85b-4529-840f-a9843a27a121 not found
Generating "/run/initramfs/rdsosreport.txt"
Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.
dracut:/#


BIOS SETUP UTILITY
Advanced
Manufacturer:Intel
Brand String:Intel(R) Core(TM) i7 CPU 870
Frequency :2.93GHz
BCLK Speed :133MHz
Cache L1 :256 KB
Cache L2 :1024 KB
Cache L3 :8192 KB
Ratio Status:Unlocked (Min:09, Max:22)
Ratio Actual Value:22
CPUID :106A5
CPU Ratio Setting 22.01
C1E Support [Enabled]
Hardware Prefetcher [Enabled]
Adjacent Cache Line Prefetch [Enabled]
Max CPUID Value Limit [Disabled]
Intel(R) Virtualization Tech [Enabled]
CPU TM Function [Enabled]
Execute-Disable Bit Capability [Enabled]
When disabled, force the XD feature flag to always return 0.
++ Select Screen
↑↓ Select Item
+- Change Option
F1 General Help
F10 Save and Exit
ESC Exit
v02.61 (C)Copyright 1985-2009, American Megatrends, Inc.

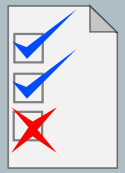
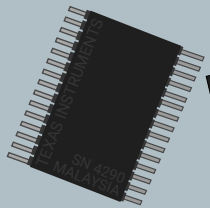
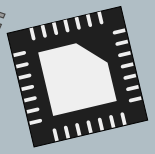
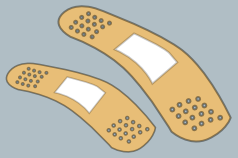

GNU GRUB version 1.99-21ubuntu3
Ubuntu, with Linux 3.2.0-23-generic
Ubuntu, with Linux 3.2.0-23-generic (recovery mode)
Memory test (memtest86+)
Memory test (memtest86+, serial console 115200)
Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, `e' to edit the commands
before booting or `c' for a command-line.







Prove:

∀( ,  ,  , ).

¬(buffer_overflow()

∨ null_dereference()

∨ unallocated_access()
);

Challenges:

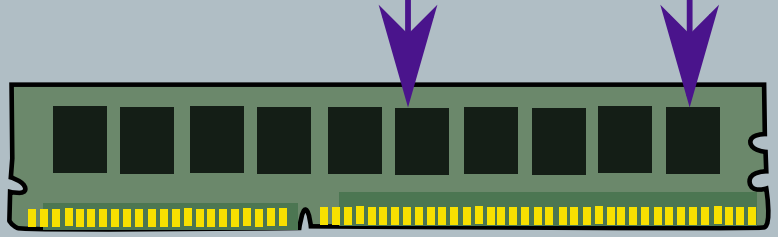
- Memory-mapped I/O
- Device behaviour
- Efficiency
- Linker scripts

MMIO

```
int x = 7;
int *y = malloc(...);
```

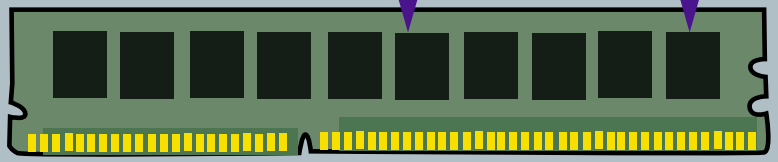
MMIO

```
int x = 7;
int *y = malloc(...);
```



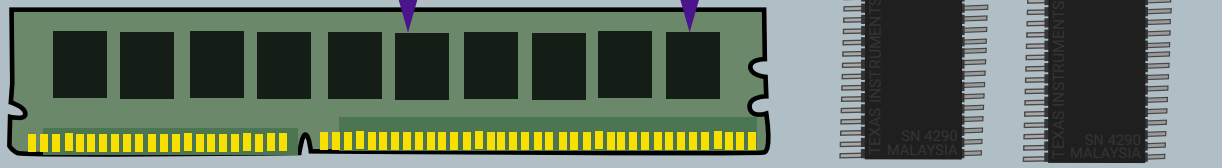
MMIO

```
int x = 7;
int *y = malloc(...);

#define REG_BASE          (0x1000)
#define REG_BOOT_STRAP    (REG_BASE + 0x110)
```



MMIO

```
int x = 7;
int *y = malloc(...);

#define REG_BASE          (0x1000)
#define REG_BOOT_STRAP    (REG_BASE + 0x110)
```



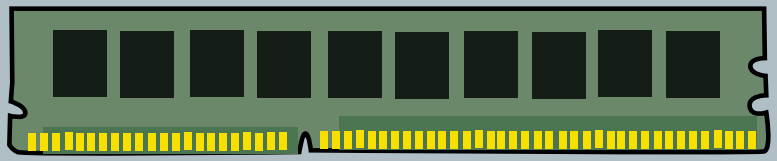
MMIO

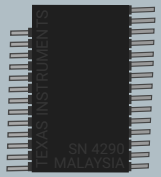
```
#define REG_BASE         (0x1000)
#define REG_BOOT_STRAP   (REG_BASE + 0x110)
#define REG_BOOT_CONF    (REG_BASE + 0x124)
```

```
__CPROVER_allocated_memory(
  REG_BOOT_STRAP, 0x14);
```

Device Behaviour

```
int x = 7;
... = x;
```

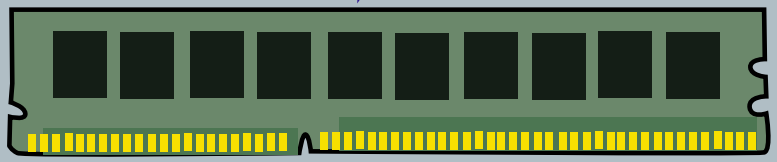




Device Behaviour

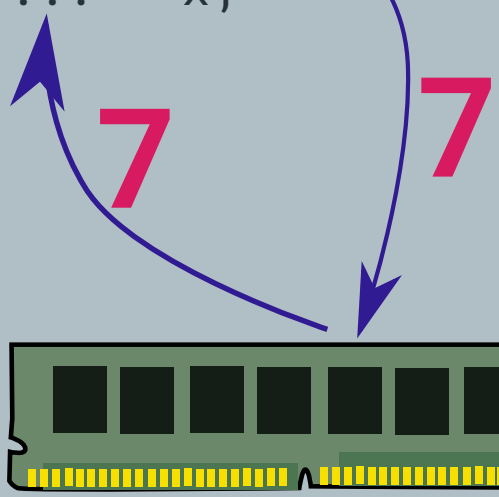
```
int x = 7;
... = x;
```

7



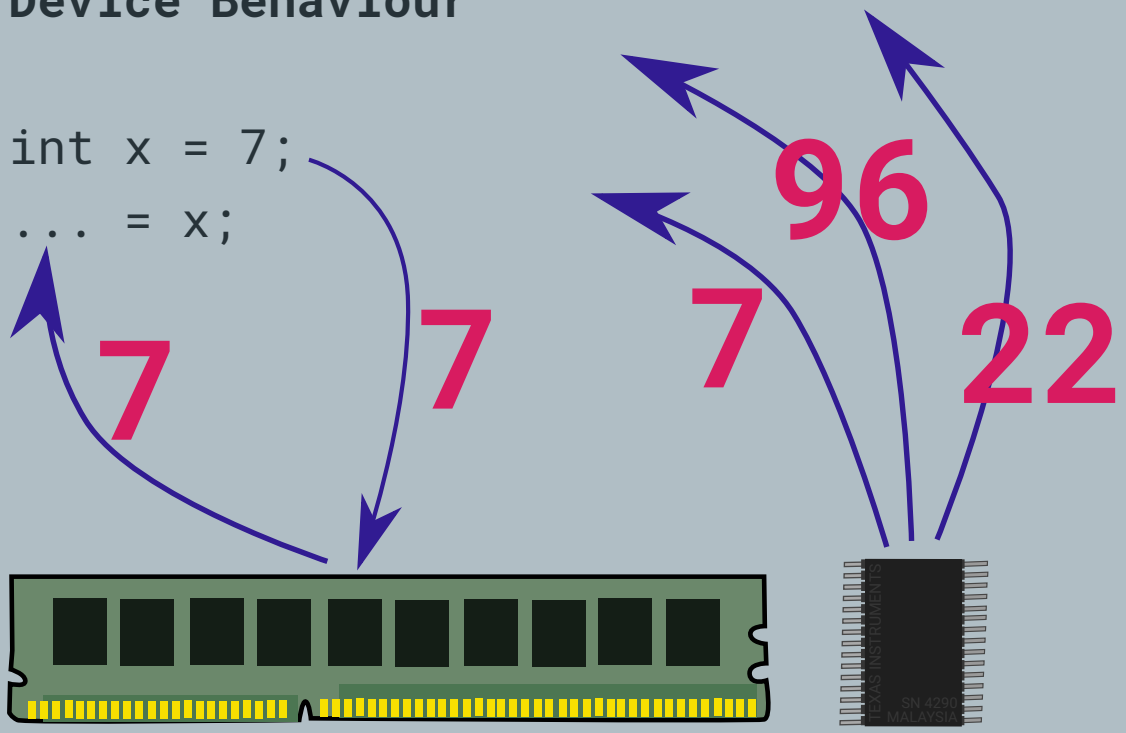
Device Behaviour

```
int x = 7;
... = x;
```



Device Behaviour

```
int x = 7;
... = x;
```



Device Behaviour

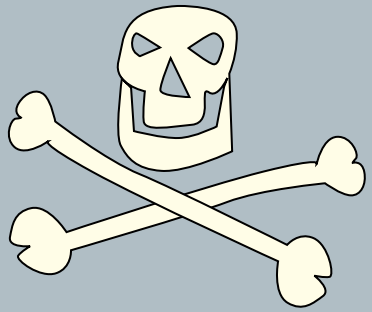
```
int access_register(...)
{

    ...

}
```

Device Behaviour

```
int access_register(...)
{


}
```

Device Behaviour

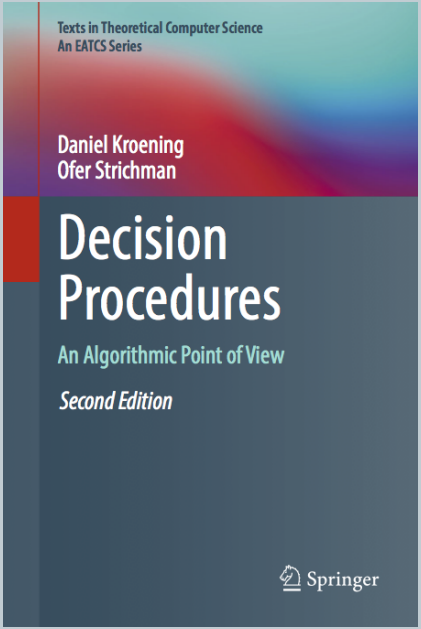
```
int access_register(...)
{


}
```



Device Behaviour

```
__CPROVER_mm_io_r(addr, size)
__CPROVER_mm_io_w(addr, size, val)
```

Efficiency

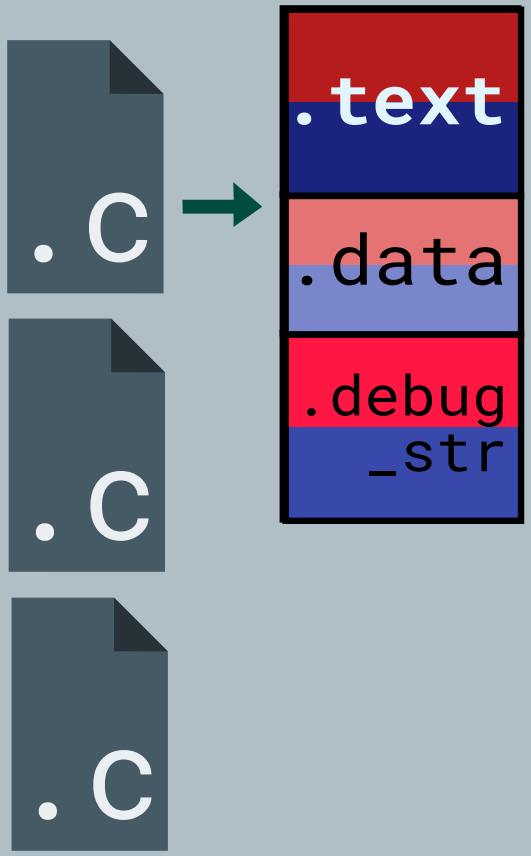


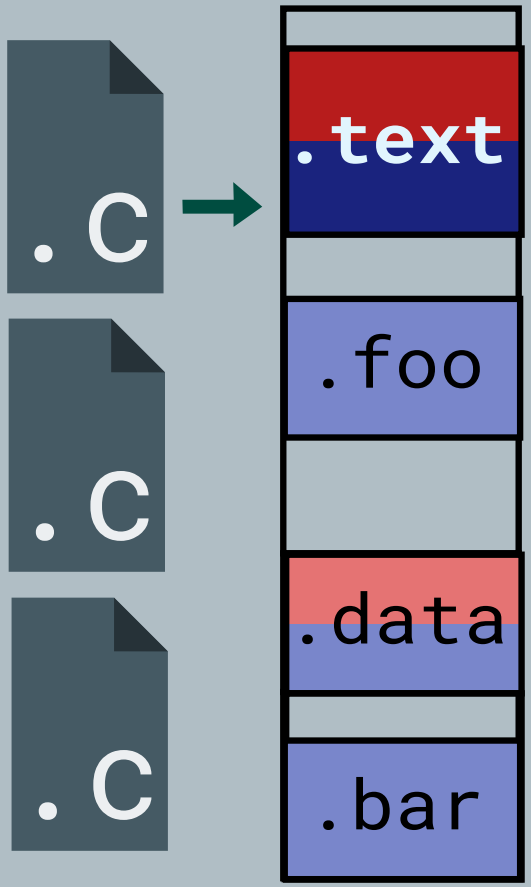
- Byte-level memory access
- memcpy implementation

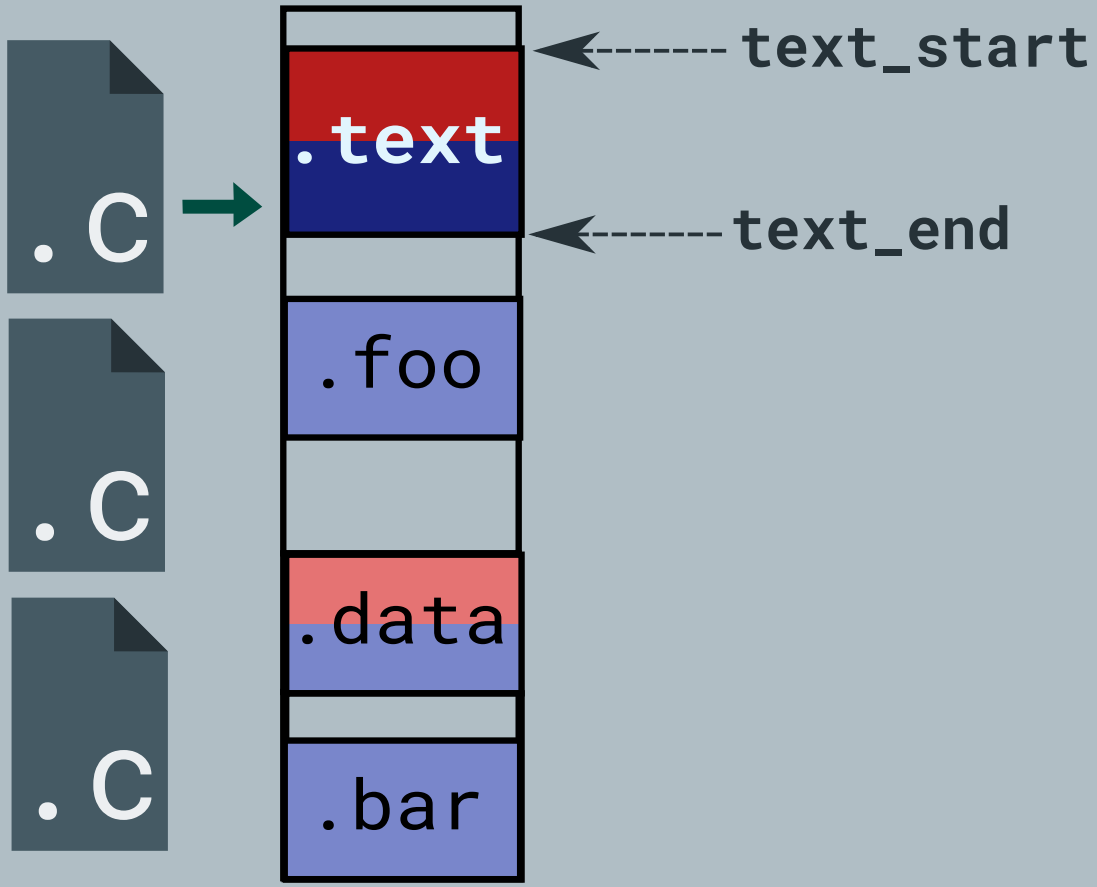

. C

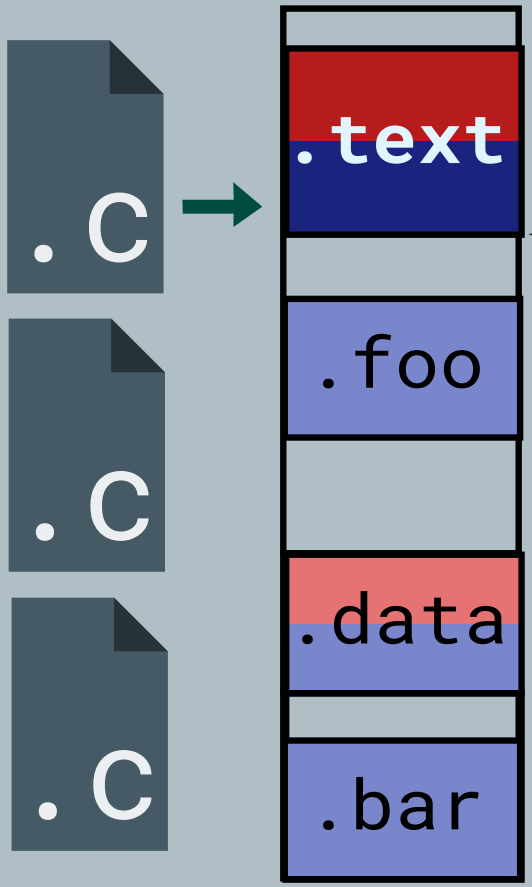
. C

. C

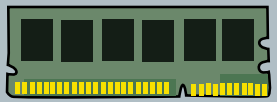

.c
.c
.c
.text
.data
.debug
_str

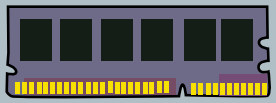

.c
.c
.c
.text
.foo
.data
.bar

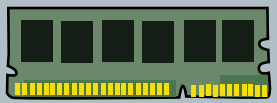

.c
.c
.c
.text
.foo
.data
.bar
text_start
text_end



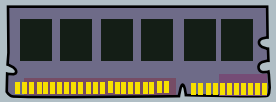
- Tightly-coupled memory
- Device access
- Machine code manipulation





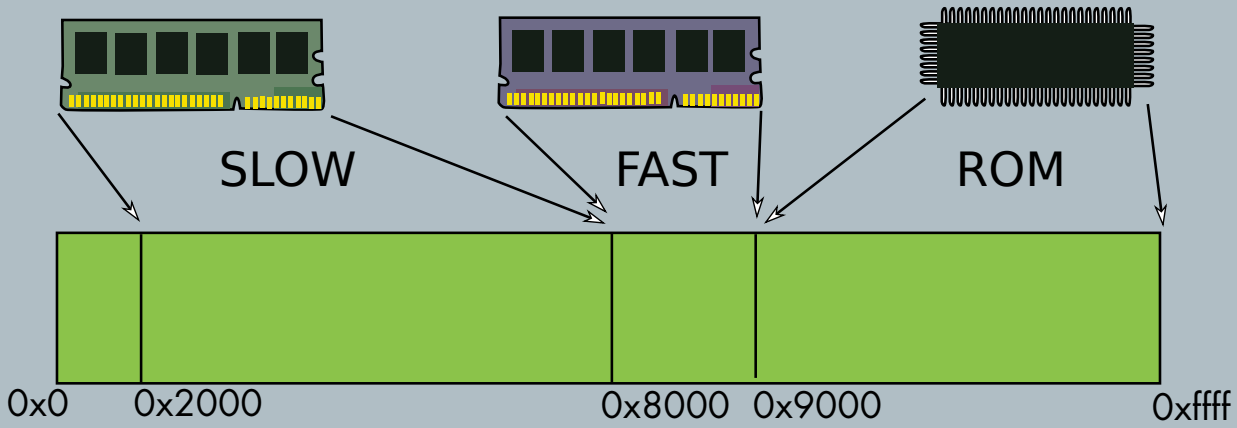


SLOW



FAST

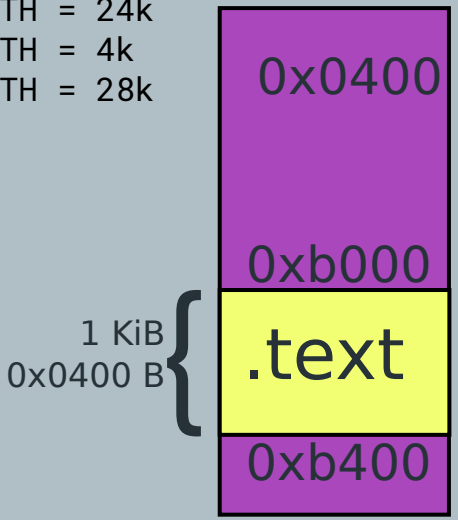
ROM


SLOW
FAST
ROM
0x0
0x2000
0x8000
0x9000
0xffff

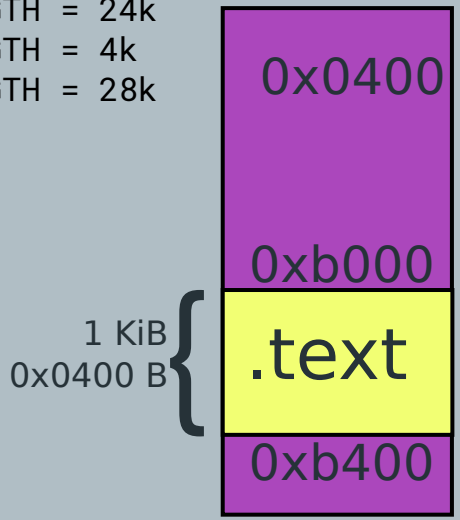

0x0
0x2000
0x8000
0x9000
0xffff


```
MEMORY {
  SLOW: ORIGIN = 0x2000, LENGTH = 24k
  FAST: ORIGIN = 0x8000, LENGTH = 4k
  ROM:  ORIGIN = 0x9000, LENGTH = 28k
}

.text : {
  *(.text*)
} > ROM
```

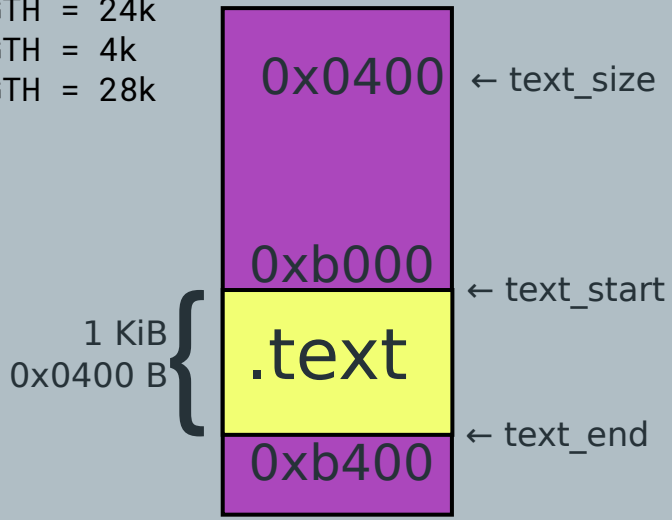

```
MEMORY {
  SLOW: ORIGIN = 0x2000, LENGTH = 24k
  FAST: ORIGIN = 0x8000, LENGTH = 4k
  ROM:  ORIGIN = 0x9000, LENGTH = 28k
}

.text : {
  *(.text*)
} > ROM
```



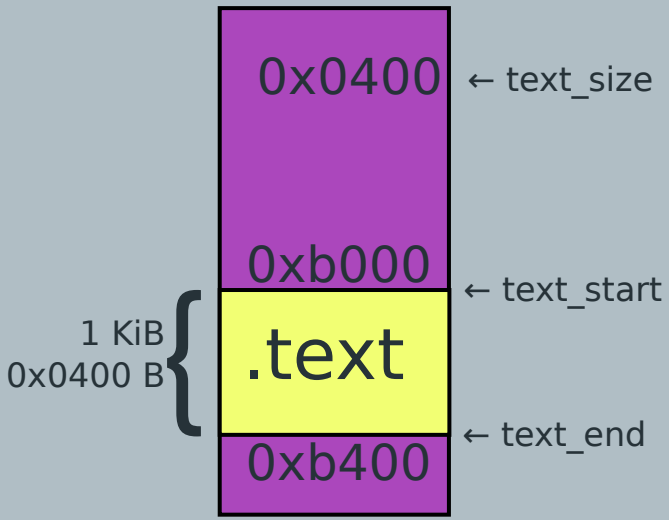

```
MEMORY {
  SLOW: ORIGIN = 0x2000, LENGTH = 24k
  FAST: ORIGIN = 0x8000, LENGTH = 4k
  ROM:  ORIGIN = 0x9000, LENGTH = 28k
}

.text : {
  text_start = .;
  *(.text*)
  text_end = .;
} > ROM
text_size = SIZEOF(.text)
```



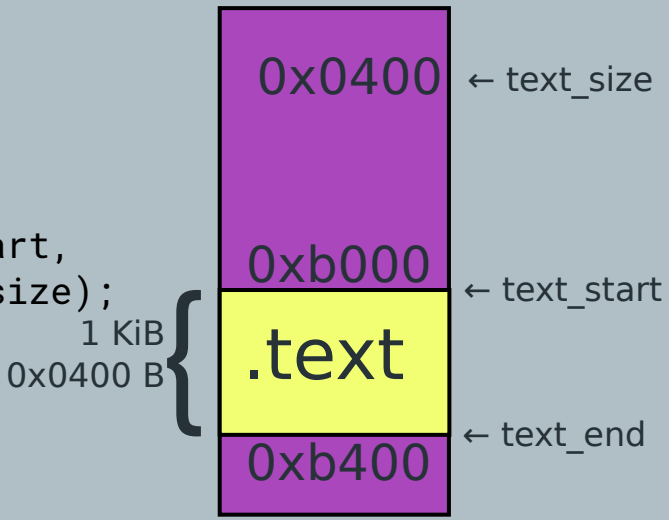

```
MEMORY {
  SLOW: ORIGIN = 0x2000, LENGTH = 24k
  FAST: ORIGIN = 0x8000, LENGTH = 4k
  ROM:  ORIGIN = 0x9000, LENGTH = 28k
}

.text : {
  text_start = .;
  *(.text*)
  text_end = .;
} > ROM
text_size = SIZEOF(.text)
```




```
extern char text_size[];
extern char text_start[];
extern char text_end[];

int main() {
  assert(&text_size ==
         (char *)0x400);

  assert(&text_start ==
         (char *)0xb000);
}
```




```
extern char text_size[];
extern char text_start[];
extern char text_end[];

int main() {
  memcpy(buf,
         (void *)&text_start,
         (size_t *)&text_size);
}
```



The problem

```
extern char text_size[];
extern char text_start[];
extern char text_end[];

int main(){
  memcpy(buf,
         (void *)&text_start,
         (size_t)&text_size);
}
```

The problem

```
extern char text_size[];
extern char text_start[];
extern char text_end[];

int main(){
  memcpy(buf,
         (void *)&text_start,
         (size_t)&text_size);
}
```

harness_1.c

```
char text_size[];
&text_size = 0x0400;
```

0200

The problem

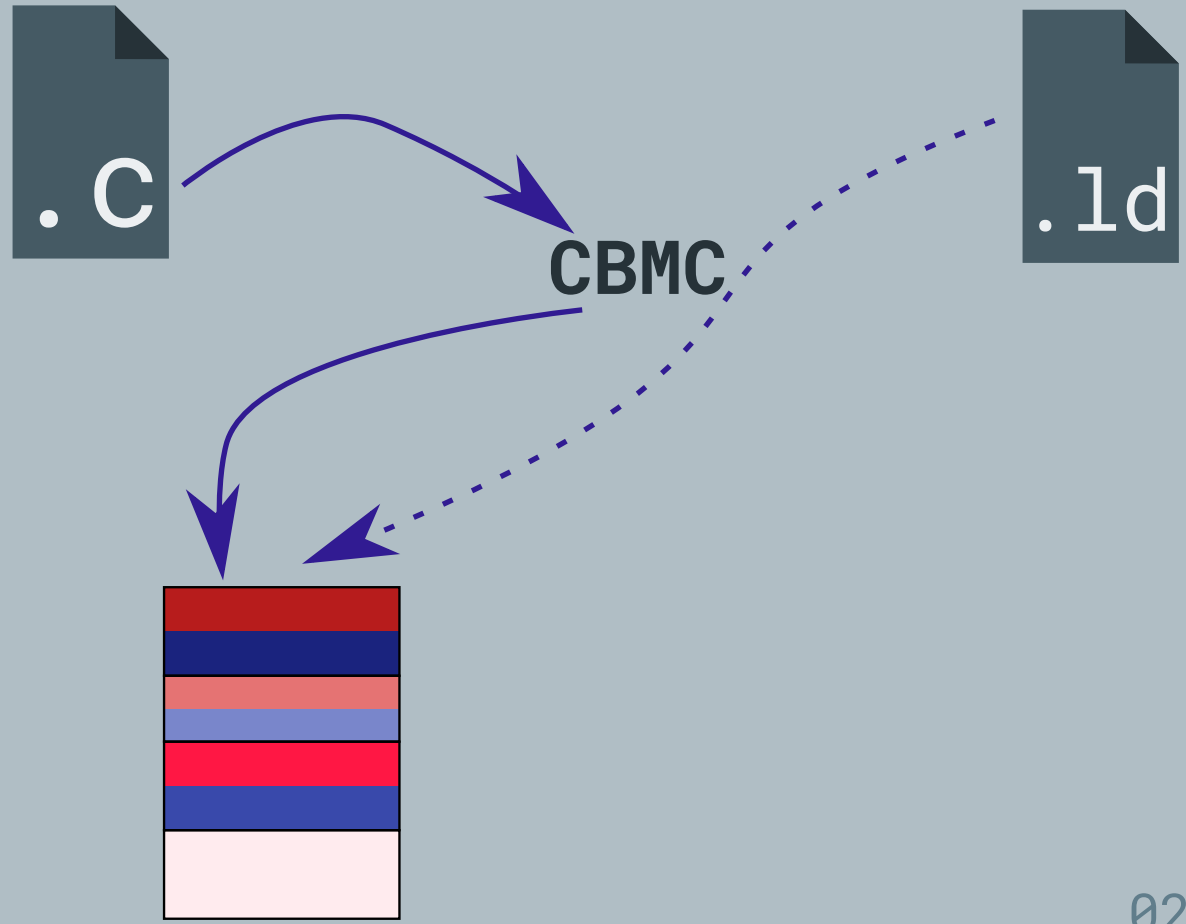
```
extern char text_size[];
extern char text_start[];
extern char text_end[];

int main(){
  memcpy(buf,
         (void *) text_start,
         (size_t) text_size);
}
```

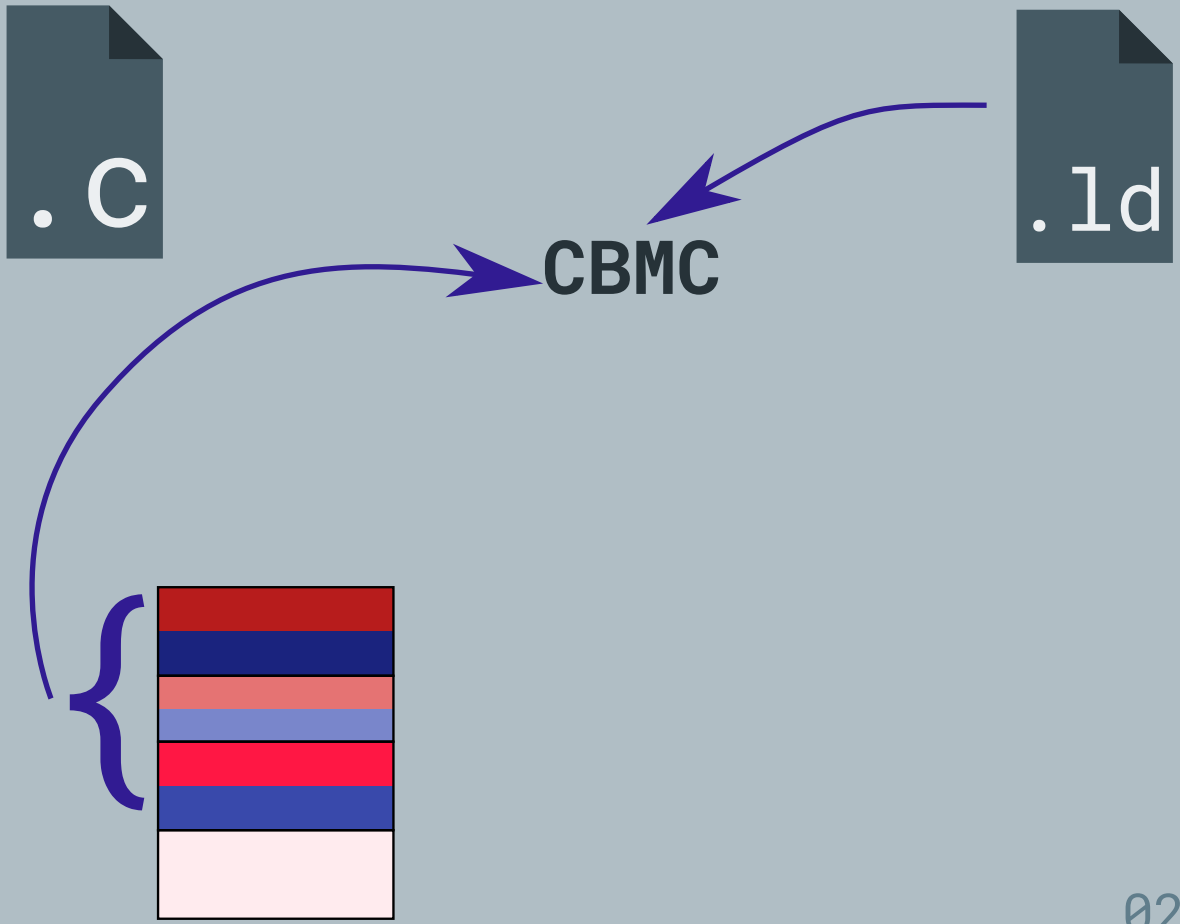
no ampersand

harness_2.c

```
-----------------------------------------
unsigned text_size  = 0x0400u;
unsigned text_start = 0xb000u;
```

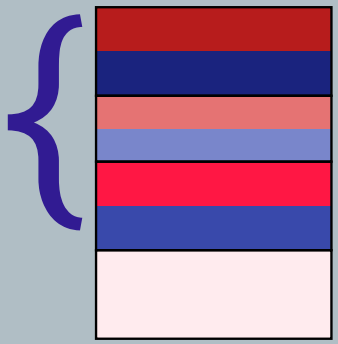
CBMC


.c
CBMC
.ld


.C
.ld
CBMC

CBMC

```
text_start = 0xb000;
text_size  = 0x0400;

__CPROVER_allocated_memory(
  0xb000, 0x0400);
```



Summary

- Wanted to prove absence of bugs in boot code

- Several challenges unique to low-level code

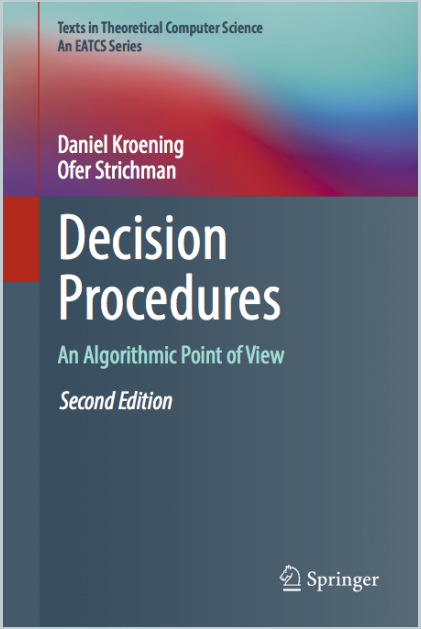
- We enhanced CBMC to overcome these challenges

Model Checking Boot Code in AWS Data Centers

Kareem Khazem

Backup slides

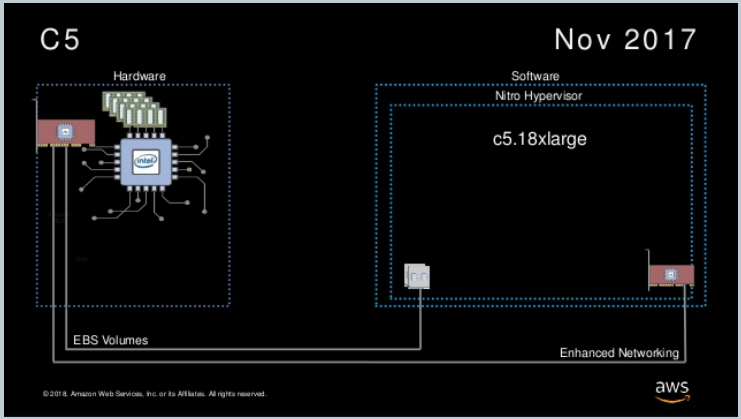
Decision Procedure for Arrays



"Weakly Equivalent Arrays".

Jürgen Christ, Jochen Hoenicke *(FroCos 2015)*.

Nitro Architecture



https://bit.ly/2LtUxys

https://youtu.be/LabltEXk0VQ