A proposal for the *real* LinuxBIOS

Ron Minnich
LANL

LA-UR-xx-xxxx

Overview

- Where we are
- Why we are there
- The problem
- EFI ... a bright light on the horizon
  - But not why some people think!
- A proposed change
- Conclusions

Where we are

- LinuxBIOS works
- Installed in millions of nodes around the world
- Good vendor buy-in
  - Support from solid companies like LNXI, AMD, etc.
- Rapidly becoming a de-facto standard in parts of the embedded space
  - Fast boot, good control, code quality matter!
- By any measure, we are succeeding

Where we are

- *But it's not where we were meant to be!*
- Look at the name – LinuxBIOS
  - Linux is the BIOS
- What we have is an extremely high quality, open source ("free software") OS loader
  - Can load just about any OS, including windows
- But we could make it better
- If we go back to the original vision

The original vision

- What we call LinuxBIOS, today, was supposed to be about 20 lines of assembly code
- Inspired by the SGI visual workstation
  - Which had no BIOS in the traditional sense
- The idea was to:
  - Put Linux in flash
  - At startup, bcopy it to memory
  - Jump to Linux

What happened to the vision

- PCs got complex
- Linux had less capability than we thought
- Chipsets got complicated
- Sizes went the wrong direction
-

PCs got complex

- In the old days, you turned PC on, memory was up and working – all in hardware
- Now, we have SDRAM, which requires a full PCI enumeration before it can be configured!
- And the memory bus chips tune the bus
- We have SATA, hypertransport topologies to work out, complex bus timings, and so on
- PCs rival old mainframes for complexity

Linux had less capability than we thought

- Originally, we thought we could set PCI_DIRECT and let Linux configure PCI
- Linux can not enumerate a PCI bus if BARs are not initialized
- Linux can not, yet, configure a completely unconfigured PC
  - It's hard to test this, absent a LinuxBIOS machine
- So LinuxBIOS had to do more configuration than planned

Chipsets got complicated

- Chipsets are far more complex than in the beginning
- Complex timing for DDR, PCI-E setup, and so on
- Linux doesn't have the ability to set up these complex timings
- LinuxBIOS has had to take on these tasks too

Sizes went the wrong direction

- BIOS FLASH parts got *smaller*
- Linux got *bigger*
- As a result, ca. 2001, we couldn't put Linux in BIOS FLASH any more
- Had to add IDE-flash, which meant
- We needed a boot loader as the LinuxBIOS payload, not Linux!

The picture today

LinuxBIOS
Turn on memory
Enumerate PCI
Build other tables

Boot loader
Enumerate PCI
Build other tables

Etherboot, FILO,
OpenBIOS

Linux
Enumerate PCI
Build other tables

This picture is not what we had in mind ...

- What's happened here?
- Increasing chipset complexity?
- Smaller FLASH size?
- Bigger Linux?
- What else?

EFI ... a bright spot on the horizon

- But not why you think!
- EFI is a porky piece of software
  - Requires HUGE flash parts – 16 Mbytes
- So when we replace EFI with Linux, we have *lots* of room
- Opens up the possibility of returning to the original vision

Once we solve the FLASH size problem ...

- We've got a lot of work left to do!
- Linux, in its current state, can not configure an unconfigured platform
- It can't do PCI
- It can't do PCI-E
- It can't do a lot of startup work

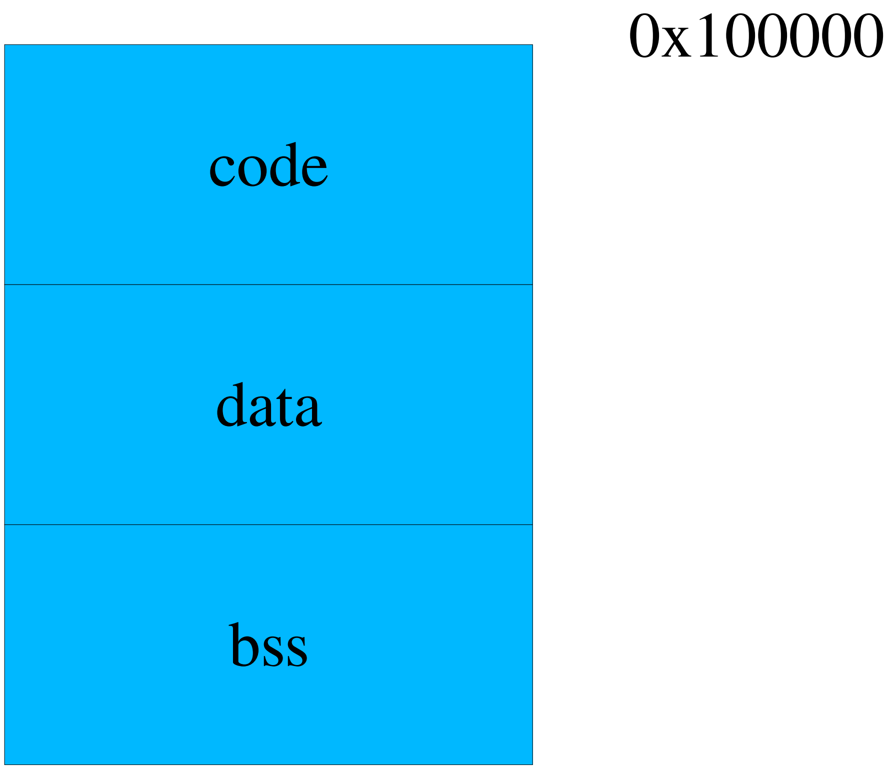
So the real issue is ...

- Linux capability, not FLASH size
- FLASH size is pretty easy to fix, given Firmware Hub and LPC flash
- What does Linux need to be able to do?
- Turn on memory
- Turn on a chipset that is not on
  - Topology
  - etc.

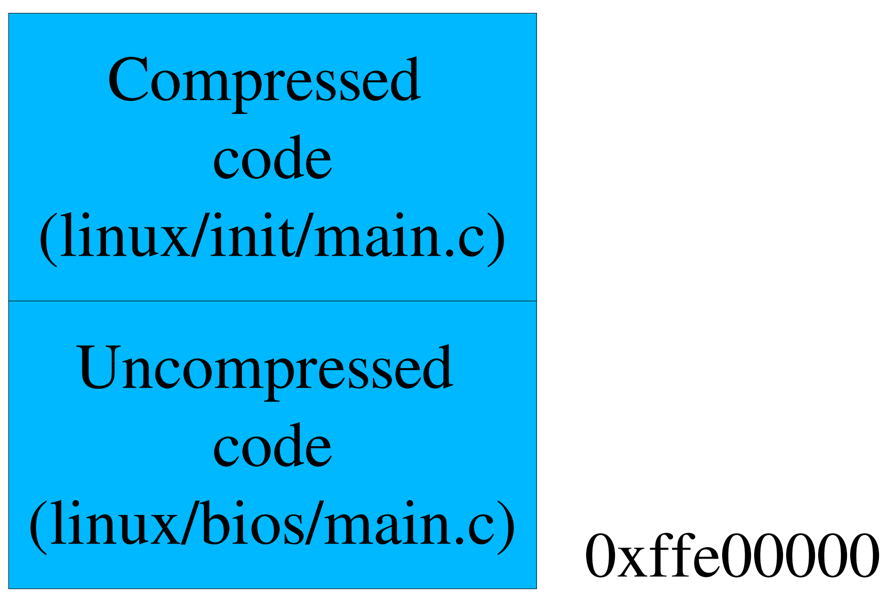
How can Linux do this?

- I think it can, now that we have CAR
  - We have a stack!
  - Local variables and so on
- The kernel startup will need to be reordered a bit
- Basic steps:
  - Establish CAR (assembly)
  - Start up early C code
  - Turn on platform
  - Copy Linux to ram

Linux image today



Linux BIOS image (a.k.a. LinuxBIOS)



What's in uncompressed code

- CAR code (assembly)
- PCI enumeration, data structures stored in CAR
- Any hypertransport setup
- Dram startup
- Turn on dram, save data structures so init/main can pick them up
- Copy linux to ram
- Jump to init/main.c

Result

- Only one PCI enumeration step
- Important chipset code makes it into Linux
- More buy-in from vendors
- Less FUD from companies like I*t**
  - Yeah, there is some: “LinuxBIOS doesn't include Linux anyway!”
- LinuxBIOS subsumed into Linux; set of hackers grows by over 100x

Linux structure changes

- New directories: linux/bios, drivers/north, drivers/south, drivers/superio
- arch/xxx/cpu
  - Not really new, see strongarm
- New config option: LinuxBIOS
- So building a BIOS becomes an option to building a kernel

Other desirable changes

- Many motherboards (e.g. Tyan) have ranges that are very similar
- It would be nice to be able to check the topology, and run the correct init code for the motherboard
- In other words, one bios for a range of motherboards

Summary

- LinuxBIOS has succeeded
  - Without Linux kernel in some cases
- Increasing hardware complexity, drivers, hardware, make subsuming LinuxBIOS into Linux desirable
- We believe that with the advent of
  - CAR
  - The huge FLASH parts needed for EFI
- this is now achievable

Discussion