x86 od první instrukce s corebootem

Ruik
r.marek@assembler.cz
ruik on #coreboot
(irc.freenode.net)

Osnova přednášky

- Kdo jsme?
- Architektura corebootu
- Jak vypadá PC po zapnutí
- Co vše se musí udělat
- Systemové tabulky
- Boot
- Blobs
- Komentovaná prohlídka zdrojáků

Kdo jsme? - coreboot

- coreboot – svobodný software
- Náhrada za tradiční BIOS
- Startujeme opravdu rychle → 500-1500ms

- Projekt začal v roce 1999 jako odpověd:
  - Q: “Keyboard error press F1 to continue”
  –

Kdo jsme? - coreboot

- coreboot – svobodný software
- Náhrada za tradiční BIOS
- Startujeme opravdu rychle → 500-1500ms

- Projekt začal v roce 1999 jako odpověd:
  - Q: “Keyboard error press F1 to continue”
  - A: krát 1024 stiskutí F1 u clusteru...

Kde jsme k zastižení

- Google Summer of Code
  - Šance pro studenty si slušně vydělat přes léto s corebootem
- FOSDEM
  - Velká konference v Bruselu
  - Setkání opensource vývojářů z celého světa
- Coreboot hackaton
  - Poslední byl v říjnu na ČVUT FIT

Hackton na FIT

Kde je coreboot?

- V chromebook od googlu
- Ve speciálních laptopech
- Různé embedded systémy
- FSF notebooky
  - Nemají rádi binarní bloby (až moc)
  - Vlastní distribuce – libreboot

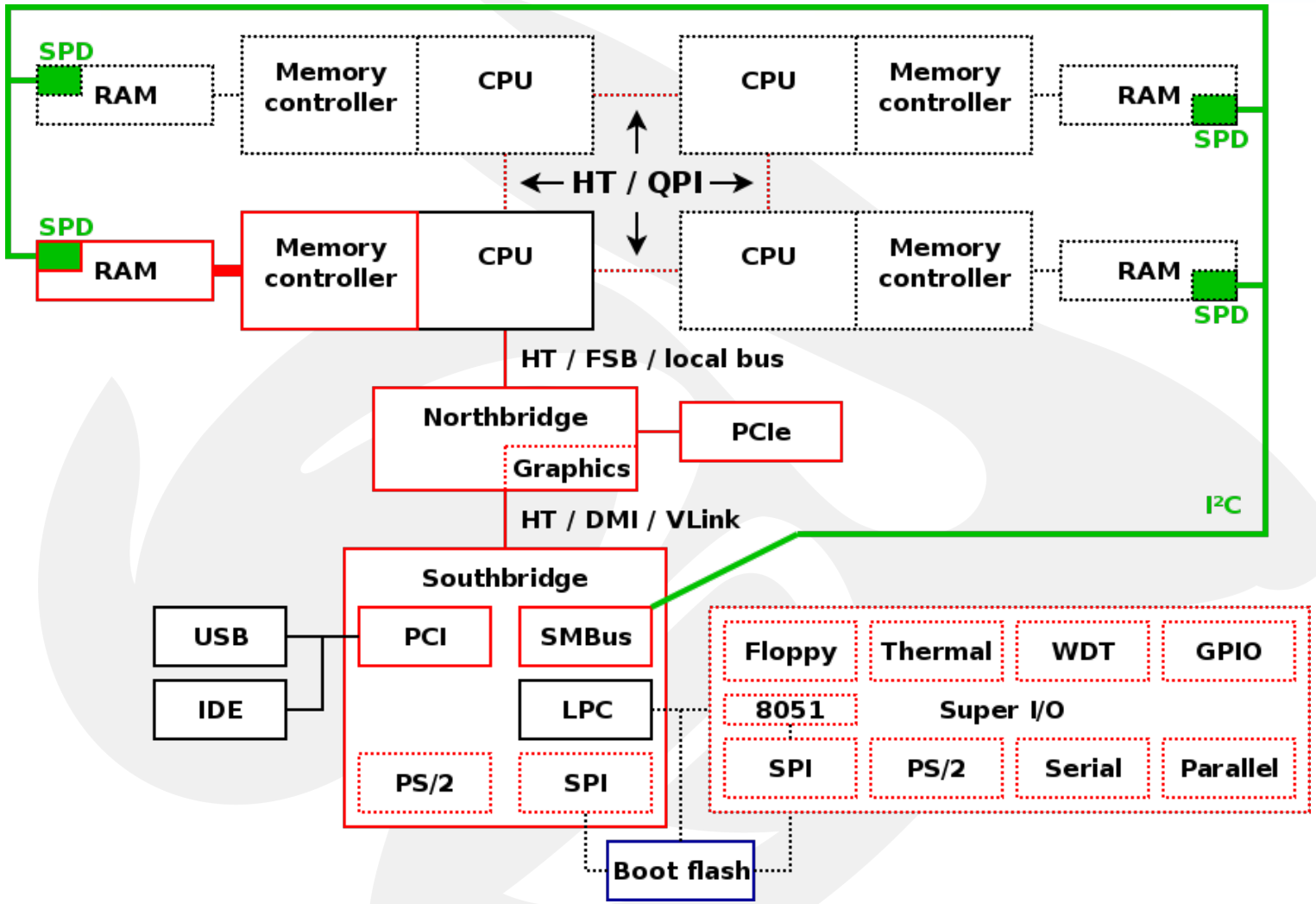
Podporované systémy

- X86 a ARM
- Viz http://www.coreboot.org/Supported_Motherboards
- AMD
  - Momentálně FM2+ Trinity/Richland/Kaveri
  - Plus nové embedded SoC
- Intel
  - Sandy bridge/Ivy bridge/Haswell
  - Pozor některé časti closed source!
  - Intel FSP (Firmware support Package)

Architektura corebootu

- Uložen v paměti flash na motherboardu (CBFS)
- Inicializace probíhá pomocí stages
  - Bootblock – najde romstage
  - Romstage – základní inicializace
  - Ramstage – další inicializace (a další CPU)
- Řízení se předá něčemu dalšímu (payload)
  - Linux kernel
  - Nebo emulace legacy BIOSu – SeaBIOS
  - UEFI – TianoCore
  - GRUB2

Modelový organismus



PC po zapnutí

- Power sequencing
  - Reší jižní můstek
  - Případně z resetu je x86 procesor vyjmut nějakým pomocným procesorem (v laptopu např. EC)
- Někdy se základní nastavení chipsetu bere z flash paměti, někdy ze strap resistors.
- Procesor začne zpracovávat instrukce z reset vectoru, kód běží přímo z fllash paměti

Bylo nebylo...

- CPU staruje v “real” mode
- ano kompatibilní s 8086 (1978)
- CPU nahraje první instrukci z reset vektoru

```
000fffd0  00 00 41 53 52 4f 43 4b  00 39 33 39 41 37 38 35  |..ASROCK.939A785|
000fffe0  47 4d 48 00 2e 00 00 00  27 00 00 00 00 00 10 00  |GMH.....'.......|
000ffff0  e9 11 fc ff ff 00 00 00  e9 5f fc ff de fb ff ff  |.........._......|
00100000
```

```
E911FC jmp word 0xfc04
```

```
objdump -d bootblock.elf
(coreboot/src/cpu/x86/16bit/entry16.inc)
fffffc04 <_start>:
fffffc04:       fa                      cli
fffffc05:       66 89 c5                mov    %ax,%bp
```

Protected mode

- Realný režim má jen 1MB operační paměti
- coreboot napsán v C
- Chceme spustit 32-bit kód
- Přepnutí v src/cpu/x86/32bit/entry32.inc
- Honem do C...
- Ale ne! Kdepak mám zásobník? (esp)

```
data32 lgdt %cs:(%bx)

movl %cr0, %eax
andl $0x7FFAFFD1, %eax
orl $0x60000001, %eax
movl %eax, %cr0
```

Život bez zásobníku

- nemáme RAM, běžíme z ROM (XIP)
- Jak udělat “call” “push” “pop” “ret” ...
- Jak použít proměnné a nebo C funkce?
  - Použít romcc (specialní C compiler který nepoužívá stack)
  - Použít Cache jako RAM (zapneme cache a používáme ji jako 64KB RAM, jen “pár” instruckcí)
- Teď už můžeme do C!

coreboot ROM stage

- první C kód spuštěn z romstage.c
- Inicializace chipsetu a superIO aby fungovala sériová linka
- První fáze inicializace: Chipset/CPU/paměť

romstage.c


```
void cache_as_ram_main(unsigned long bist, unsigned long cpu_init_detectedx)
{
        static const u16 spd_addr[] = { DIMM0, DIMM2, 0, 0, DIMM1, DIMM3, 0, 0, };
        int needs_reset = 0;
        u32 bsp_apicid = 0;
        msr_t msr;
        struct cpuid_result cpuid1;
        struct sys_info *sysinfo = (struct sys_info *)(CONFIG_DCACHE_RAM_BASE +
CONFIG_DCACHE_RAM_SIZE - CONFIG_DCACHE_RAM_GLOBAL_VAR_SIZE);

        if (!cpu_init_detectedx && boot_cpu()) {
                /* Nothing special needs to be done to find bus 0 */
                /* Allow the HT devices to be found */
                enumerate_ht_chain();
                /* sb700_lpc_port80(); */
                sb700_pci_port80();
        }

        if (bist == 0)
                bsp_apicid = init_cpus(cpu_init_detectedx, sysinfo);

        enable_rs780_dev8();
        sb700_lpc_init();

        w83627dhg_enable_serial(SERIAL_DEV, CONFIG_TTYS0_BASE);
        uart_init();

#if CONFIG_USBDEBUG
        sb700_enable_usbdebug(CONFIG_USBDEBUG_DEFAULT_PORT);
        early_usbdebug_init();
#endif

        console_init();
        sio_init();

        /* Halt if there was a built in self test failure */
        report_bist_failure(bist);
        printk(BIOS_DEBUG, "bsp_apicid=0x%x\n", bsp_apicid);
```

RAM inicializace I

- Našíst časovací data pro DIMM
- přes I2C bus (SMBus controller)
- Naprogramovat chipset (or CPU) registry
- Poslat JEDEC příkazy RAM
- Hotovo
- Nikolivěk...

RAM inicializace II

- Velmi složité pro DDR2/3
- I když napsáno v C, SLOCcount (circa)
  - DDR (AMD) – 1710 řádek
  - DDR2 (AMD) – 3587 řádek
  - DDR3 (AMD) – 8432 řádek
  - DDR2 (Intel) – 2276 řádek
- Pamět se trénuje, některé parametry se hledají experimentálně - DQS timing!

RAM stage

- Jakmile je romstage u konce a pamět beží nahrahajeme ramstage
- coreboot má něco jako linux kernel “drivery” a sofistikovaný IO/mem resource management system
- Musíme:
  - natrénovat PCIe linky
  - Alokovat BARy pro PCI/PCIe a všechny jiné MMIO
  - Inicializovat chipset (hlavně jižní můstek)
  - Připravit tabulky pro OS

Tabulky pro OS

- Memory map (e820)
  - Mapa RAM pro OS
- Další tabulky pro OS
  - PIR pro IRQ routing
  - MPtable pro IRQ routing, SMP aware
  - ACPI (pro power management, IRQ routing a... vůbec všechno...)

Předání řízení paylodu

- Payload je ve stejné flash paměti
- Může být víc než jeden
  - SeaBIOS – legacy BIOS compatible layer
    - spuští legacy option ROMs
    - A pak boot sektor z bootovacího zařízení
    - Grub/Windows apod...
  - Nebo cokoli jiného
    - Tint – Tetris
    - ROMbasic IBM (z roku 1981)
    - Jiný OS...

Ukázka a bootlog

- Asus F2A85-M
  - FM2 socket
  - AMD Trinity CPU

http://coreboot.org

Komentovaná prohlídka zdrojáků

Chvilka pro free software

- Coreboot je pod GPL v2
- SeaBIOS je pod LGPL v3
- Payload může být pod jakoukoli licencí
- Existují coreboot platformy, kde je všechen x86 kód ve formě zdrojáků
- Problematický VGA BIOS
  - Obsahuje bytekod definovaný výrobcem desky
- Problémy se současným hardwarem co už není moc hardware...

Binární bloby

- Data/kód dodaný výrobcem hardwaru
- Nemusí být nutně pro hlavní processor
  - Mikrokód pro procesor
  - Firmware pro jiné procesory:
    - Intel ME
    - AMD SMU
    - USB3
    - Nebo v jižním můstku i nějaká 8051...
      - AMD IMC
      - Keyboard controller
      - EC v notebooku

Binární bloby II

- Stírá se hranice mezi hardware a software
- Moderní x86 má okolo 5 jiných procesorů
- Intel si chrání svoje knowhow
  - Nezveřejňuje potřebnou dokumentaci
  - Nabízí místo toho FSP – Firmware Support Package
    - Binarní x86 blob který zinicializuje “tajné” části
  - Pro sandybridge existuje už implementace bez FSP
- AMD
  - Až dosud statisíce řádků inicializace pod BSD licencí
  - Pomalu se vydává stejným směrem jako Intel

Binární bloby III

- Nejenom hlavní procesor může být nakažen škodlivým kódem
- BadUSB
  - https://srlabs.de/badusb/
- Škodlivý firmware pro USB controller (POC)
  - Phison 2251-03 (2303) Custom Firmware & Existing Firmware Patches
  - https://github.com/adamcaudill/Psychson

http://coreboot.org

Otázky ?