KIV Operační systémy

Symetrický multiprocesor

Verze 1.0

Bootstrap procesor x86

- Po zapnutí počítače a inicializaci BIOSu je aktivní pouze jeden procesor, respektive jedno procesorové jádro – tzv. BSP aneb bootstrap procesor
- Zavaděč OS musí zjistit, zda systém obsahuje i další procesorová jádra a aktivovat je
  - Pokud se mu to nepodaří, aktivuje se uniprocesorové jádro

MP Floating Pointer Structure

- Začíná signaturou _MP_ a popisuje dostupné procesory
- Hledá se
  - V 1. kB Extended BIOS Data Area
  - Posledním kB základní paměti
    - „640K ought to be enough for anybody." – Bill Gates, 1981
  - V adresním rozsahu ROM-BIOSu

MP Config

- Jedna z položek MP Floating Pointer Structure je MP Config Pointer ukazující na seznam dostupných procesorů

MP Config

Processor Entry			
Field	Offset	Length	**Description/Use**
Entry Type	0	1B	**Since this is a processor entry, this field is set to 0.**
Local APIC ID	1	1B	**This is the unique APIC ID number for the processor.**
Local APIC Version	2	1B	**This is bits 0-7 of the Local APIC version number register.**
CPU Enabled Bit	3:0	1b	**This bit indicates whether the processor is enabled. If this bit is zero, the OS should not attempt to initialize this processor.**
CPU Bootstrap Processor Bit	3:1	1b	**This bit indicates that the processor entry refers to the bootstrap processor if set.**
CPU Signature	4	4B	**This is the CPU signature as would be returned by the CPUID instruction. If the processor does not support the CPUID instruction, the BIOS fills this value according to the values in the specification.**
CPU Feature flags	**8**	**4B**	**This is the feature flags as would be returned by the CPUID instruction. If the processor does not support the CPUID instruction, the BIOS fills this value according to values in the specification.**

MP Config

- Jedna z položek MP Floating Pointer Structure je MP Config Pointer ukazující na seznam dostupných procesorů
  - BSP
  - AP
    - aka auxiliary processor
    - aka application processor

SMP Bootstrap x86

1. Po zapnutí počítače jsou všechny procesory v reálném režimu
2. BIOS vybere BSP a ostatní procesory zastaví
3. Kód běžící na BSP prohledá paměť, zda najde _MP_
4. Pokud nenajde, zavede se jednoprocesorové jádro
5. Pokud našel, inicializuje APIC BSP
   - Děje se v protected-mode
6. Kód běžící na BSP postupní vzbudí AP pomocí Init-IPI (Inter-Processor Interrupt)

SMP Bootstrap x86

7. Kód běžící na AP ho přepne do protected-mode a začne svoji další činnost synchronizovat s kódem běžícím na BSP
8. Jakmile jsou inicializovány všechny AP, BSP přepne I/O APIC do symetrického I/O režimu
   - Routovací tabulka, která přesměruje přerušení od sběrnic periferií na některý lokální APIC
9. Pokračuje se vlastní inicializací SMP jádra

SMP úskalí plánování

- SMP sice znamená, že každý procesor je stejný, a tudíž že lze každé vlákno SMP OS naplánovat na libovolný procesor, ale v praxi to rozhodně není dobrý nápad
- Každý procesor má svoji cache, ve které jsou uložena data vláken, která na procesoru naposledy běžela
- Cache podstatným způsobem přispívá k rychlému běhu vláken
- Pokud vlákna budeme naivně migrovat mezi procesory, vlákna o tuto výhodu přijdou a celý systém se zpomalí

SMP Synchronizace

- Vlákna běžící na jednotlivých procesorech mají pouze jednu možnost, jak se synchronizovat
  - Atomické instrukce
  - Add, Sub, CompareExchange aka TestAndSet
  - Prefix lock, který zamkne sběrnici
  - x86 – mov strojového slova zarovnaného na adresu beze zbytku dělitelnou velikostí strojového slova (např. sizeof eax či rax)

SpinLock

- Potřebujeme-li, aby pouze jeden z procesorů vykonával kritickou sekci, pak v paměti potřebujeme proměnnou, jejíž stav říká, zda je kritická sekce obsazená, či nikoliv
- Pokud bude obsazená jiným procesorem, pak příchozí procesor čeká ve smyčce, dokud mu ji jiný procesor neodemkne (tj. nemá smysl na uniprocesoru)
  - Pak si ji sám zamkne
- Co kdyby čekalo více procesorů?
  - => Operace s proměnnou musí být atomické

SpinLock - zamčení

```
mov edx, DWORD(-1) //-1 zamčeno
                          //otestujeme stav zámku, 0 = odemčeno
spin:  mov eax, [lockState]
        test eax, eax
         jnz spin

        //zkusíme ho zamknout s -1
      lock cmpxchg [lockState], edx
      //nepředběhl nás jiný procesor?
      //původní lockState je v eax
        test eax, eax
jnz spin
```

SpinLock - odemčení

mov DWORD PTR [lockState], 0

- A je to.

- Odemčení je jednoduché. Ale nedalo by se také udělat něco se zamčením? Přece jenom, soustavné točení se ve smyčce jenom spotřebovává energii. A co když máme mobilní zařízení?

SpinLock – úspora energie

```
//Intel® 64 and IA-32 Architectures Optimization Reference Manual 2011
if (!acquire_lock()){
  /* Spin on pause max_spin_count times before    backing off to sleep */
  for(int j = 0; j < max_spin_count; ++j)
    /* intrinsic for PAUSE instruction*/
    _mm_pause();
    if (read_volatile_lock()) {
      if (acquire_lock()) goto PROTECTED_CODE;
     }
   }

  /* Pause loop didn't work, sleep now */
  Sleep(0);
  goto ATTEMPT_AGAIN;
}
PROTECTED_CODE:
  do_work();
  release_lock();
```