Tualatin Celeron の L2 キャッシュを設定してみました

宮崎　敏昌

事の発端は・・・

- ある日、某所のゴミ箱にPC98-NXがじっとこちらを見つめているのに気付く。
- 中を見ると、CPUとMemoryが無い以外は全部そろってるらしい。
- 手持ちのCeleron 300Aとメモリを乗っけたら一応動くようだ！

とういことで、捕獲決定！

どんなものか調べてみる

- スペック

  NEC ValueStar NX PC-VE40H/87D
  CPU: Celeron 400MHz(Mendocino)
  MEM: 64MByte
  HDD: 8GByte
  光学ドライブ:32倍速 CDROM
  Video: ATI RageXL 16MByte
  (On Board)
  PCI: 空き3個(Modemカードなし)
  チップセット:440ZX

素性は良いなあ。
ということで、
パワーアップ開始！

パワーアップの手段は？

(インターネット上にたくさんありました。さすがNEC)

- CPUは、いわゆる下駄の使用で、Celeron1.4GHzまでOkだそうだ。 要改造）
- HDDは、一応8.4GByteまでBIOSで認識可。それ以上でも使用可能なようだ。現在、20GByteをつけてます。
- Memoryの上限は、256MByte らしい。現在、128MByteで使用中。

そこで、問題のCPU・・・

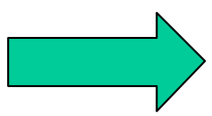
- そのままでは、動作電圧生成の関係でCeleron 533MHzまで。
- LontecのPL-370/T Rev.2.1を使えば、Tualatin Celeron 1.4GHz まで動く。だが、FSB66MHzのため、最高でも933MHzまで。
- MotherBoardのR30抵抗を取っ払うと、FSBが100MHzにアップ！ということでめでたくフルスペックでTualatin Celeronが動作できる？

　　完全に規格外の動作なので、壊れるでしょう。）

しかーし、このCPUを使ってもキャッシュが有効にならないので、大して速くなった気がしない・・。結構大問題！

FreeBSD上でキャッシュコントロール？

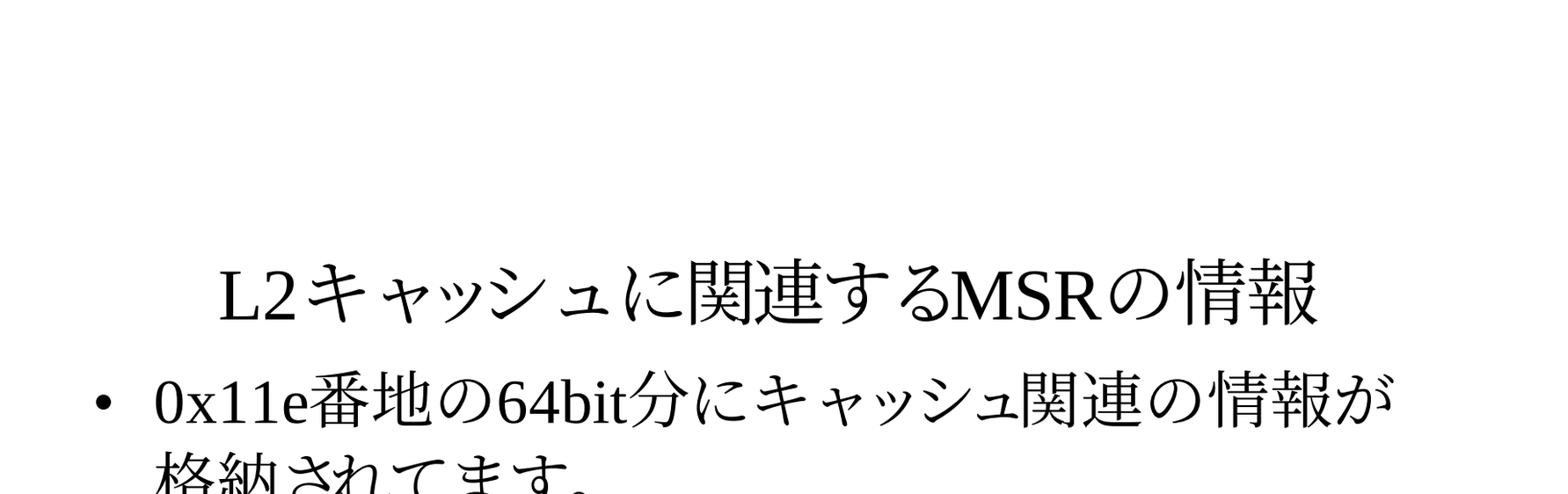
- ここからが本題（笑）
- FreeBSD Press No.1 にPC-9821上でMendocino Celeronのキャッシュ制御の記事を見つける。
- L2キャッシュの情報は、CPUのMSR(Model Specific Register) 上にあるらしい。これは、i686系にあるレジスタ。



MendocinoだろうがTualatinだろうが
同じように使えるのだろうから。
そこらへんから当たってみることにしましょう。

L2キャッシュに関連するMSRの情報

- 0x11e番地の64bit分にキャッシュ関連の情報が格納されてます。
  - キャッシュの容量 ･方式、有効？、設定済？などなど。
  - 現状のシステムでは、0x07426e3
  - TualatinCeleronのシステムで調べると、0x07427e3



現状では・・・・

- L2キャッシュに関しては、設定済み
- L2キャッシュは無効
- L2キャッシュの容量は、256kByte
- L2キャッシュの方式はReserve

ということで、BIOSとしては、

「これは不明なCPUなので、L2キャッシュは無効にしました。」

ということのようです。ここら辺、BIOSを規格通りきっちり作っていると考えた方がいいのでしょうかねえ。（謎）

えっと、ちなみにPC-9821では、L2キャッシュは何もしなくても有効になります。（これも謎）

キャッシュ制御をソースファイルに追加

- ソースのどこでキャッシュコントロールするか？
  - src/sys/i386/initcpu.c
    - initializecpu(): CPU形式による初期化の振り分け
    - init_tualatin()を追加してキャッシュ制御を実行
- 特定のCPUでキャッシュ制御を有効にするには？
  - Kernel Configに CPU_TUALATINCELERONのOption を追加
  - CPUIDが 0x6B? かつOptionが指定されている時に キャッシュ制御を実行
- キャッシュコントロールの具体的方法は？
  - L2キャッシュを有効にする16bit目を“1”にする。
  - L2キャッシュのレイテンシは、“1”なのでそのまま。

ソース(1) : initializecpu() の一部

```
case CPU_686:
    if (strcmp(cpu_vendor, "GenuineIntel") == 0) {
    switch (cpu_id & 0xff0) {
            case 0x610:
                    init_ppro();
                    break;
            case 0x660:
                    init_mendocino();
                    break;
            case 0x6B0:
                    init_tualatin();
                    break;
    }
} else if (strcmp(cpu_vendor, "AuthenticAMD") == 0) {
```

ここに追加

ソース(2) init_tualatin()

```
/* * Initialize BBL_CR_CTL3 (Control register 3: used to configure the * L2 cache). */
static void init_tualatin(void) {
#ifdef CPU_TUALATINCELERON
        u_long eflags;
        u_int64_t bbl_cr_ctl3;

        eflags = read_eflags();
        disable_intr();
        load_cr0(rcr0() | CR0_CD | CR0_NW);
        wbinvd();
        bbl_cr_ctl3 = rdmsr(MSR_BBL_CR_CTL3);

        /* If the L2 cache is enable, do nothing. */
        if (!(bbl_cr_ctl3 & 0x0100)) {
                bbl_cr_ctl3  |= 1<< 8;
                wrmsr(MSR_BBL_CR_CTL3, bbl_cr_ctl3);
        }
        load_cr0(rcr0() & ~  (CR0_CD | CR0_NW));
        write_eflags(eflags);
#endif /* CPU_TUALATINCELERON */
}
```

結果

- とりあえず、成功!
- 比較ってあまりしてないんですが・・・
  – make depend ： L2 enable 4分程度
    L2 disable 10分程度
    てなところです。
- 体感では、ずいぶん速くなった気がします。