coreboot - Bug #98

[AMD Family 10h/15h] CPU model printed with just spaces

03/02/2017 03:37 PM - Paul Menzel

Status:	New	**Start date:**	03/02/2017
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	board support	**Estimated time:**	0.00 hour
Target version:			

Description

Using coreboot on the Asus KGPE-D16, one line for the CPU model name includes just spaces.

```
$ git log --oneline -1 # board status repository
4b4b7ab asus/kgpe-d16/4.5-1093-g308aeff/2017-03-01T16_03_07Z
$ git show | grep "CPU model:" | wc -l
32
$ git show | grep "CPU model:"
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
++CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
++CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
++CPU model: AMD Opteron(tm) Processor 6278
++CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model:
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
+CPU model: AMD Opteron(tm) Processor 6278
```

Here is the code in question.

```
$ nl -ba src/cpu/amd/family_10h-family_15h/processor_name.c
[…]
   236     if (fam15h) {
   237         /* Family 15h or later */
   238         uint32_t dword;
   239         device_t cpu_fn5_dev = dev_find_slot(0, PCI_DEVFN(0x18, 5));
   240         pci_write_config32(cpu_fn5_dev, 0x194, 0);
```

```
241         dword = pci_read_config32(cpu_fn5_dev, 0x198);
242         if (dword == 0) {
243             strcpymax(program_string, sample, sizeof(program_string));
244         } else {
245             /* Assemble the string from PCI configuration register contents */
246             for (i = 0; i < 12; i++) {
247                 pci_write_config32(cpu_fn5_dev, 0x194, i);
248                 p_program_string[i] = pci_read_config32(cpu_fn5_dev, 0x198);
249             }
250 
251             /* Correctly place the null terminator */
252             for (i = (NAME_STRING_MAXLEN - 2); i > 0; i--) {
253                 if (program_string[i] != 0x20)
254                     break;
255             }
256             program_string[i + 1] = 0;
257         }
258     } else {
259         /* variable names taken from fam10 revision guide for clarity */
260         u32 BrandId;    /* CPUID Fn8000_0001_EBX */
261         u8 String1; /* BrandID[14:11] */
262         u8 String2; /* BrandID[3:0] */
263         u8 Model;   /* BrandID[10:4] */
264         u8 Pg;      /* BrandID[15] */
265         u8 PkgTyp;  /* BrandID[31:28] */
266         u8 NC;      /* CPUID Fn8000_0008_ECX */
267         const char *processor_name_string = unknown;
268         int j = 0, str2_checkNC = 1;
269         const struct str_s *str, *str2;
270 
271         /* Find out which CPU brand it is */
272         BrandId = cpuid_ebx(0x80000001);
273         String1 = (u8)((BrandId >> 11) & 0x0F);
274         String2 = (u8)((BrandId >> 0) & 0x0F);
275         Model = (u8)((BrandId >> 4) & 0x7F);
276         Pg = (u8)((BrandId >> 15) & 0x01);
277         PkgTyp = (u8)((BrandId >> 28) & 0x0F);
278         NC = (u8)(cpuid_ecx(0x80000008) & 0xFF);
279 
280         if (!Model) {
281             processor_name_string = Pg ? thermal : sample;
282             goto done;
283         }
284 
285         switch (PkgTyp) {
286         case 0:     /* F1207 */
287             str = String1_socket_F;
288             str2 = String2_socket_F;
289             str2_checkNC = 0;
290             break;
291         case 1:     /* AM2 */
292             str = String1_socket_AM2;
293             str2 = String2_socket_AM2;
294             break;
295         case 3:     /* G34 */
296             str = String1_socket_G34;
297             str2 = String2_socket_G34;
298             str2_checkNC = 0;
299             break;
300         case 5:     /* C32 */
301             str = String1_socket_C32;
302             str2 = String2_socket_C32;
303             break;
304         default:
305             goto done;
306         }
307 
```

```
  308         /* String1 */
  309         for (i = 0; str[i].value; i++) {
  310             if ((str[i].Pg == Pg) &&
  311             (str[i].NC == NC) &&
  312             (str[i].String == String1)) {
  313                 processor_name_string = str[i].value;
  314                 break;
  315             }
  316         }
  317 
  318         if (!str[i].value)
  319             goto done;
  320 
  321         j = strcpymax(program_string, processor_name_string,
  322             sizeof(program_string));
  323 
  324         /* Translate Model from 01-99 to ASCII and put it on the end.
  325         * Numbers less than 10 should include a leading zero, e.g., 09.*/
  326         if (Model < 100 && j < sizeof(program_string) - 2) {
  327             program_string[j++] = (Model / 10) + '0';
  328             program_string[j++] = (Model % 10) + '0';
  329         }
  330 
  331         processor_name_string = unknown2;
  332 
  333         /* String 2 */
  334         for (i = 0; str2[i].value; i++) {
  335             if ((str2[i].Pg == Pg) &&
  336             ((str2[i].NC == NC) || !str2_checkNC) &&
  337             (str2[i].String == String2)) {
  338                 processor_name_string = str2[i].value;
  339                 break;
  340             }
  341         }
  342 
  343  done:
  344         strcpymax(&program_string[j], processor_name_string,
  345             sizeof(program_string) - j);
  346     }
  347 
  348     printk(BIOS_DEBUG, "CPU model: %s\n", program_string);
[…]
```