Liste des sections traitées

- 0x00000000fffffff0:0x00000000fffffff4 CODE16
- 0x00000000ffffff40:0x00000000ffffff6a CODE16
- 0x00000000ffffffb8:0x00000000ffffffbd GDTR32
- 0x00000000ffffff70:0x00000000ffffffb7 GDT32
- 0x00000000fffffe66:0x00000000fffffe80 CODE32
- 0x00000000fffffe81:0x00000000fffffe93 CODE32
- 0x00000000fffffe94:0x00000000fffffe9a CODE32
- 0x00000000fffffe62:0x00000000fffffe65 DATA32
- 0x00000000fffffe9b:0x00000000fffffea4 CODE32
- 0x00000000fffffea5:0x00000000ffffff12 CODE32

Passage en mode protégé

```
00000000fffffff0 0f09            wbinvd
00000000fffffff2 e94bff          jmp 0xffffff40
```

```
00000000ffffff40 dbe3            fninit
00000000ffffff42 0f6ec0          movd %eax, %mm0
00000000ffffff45 fa              cli
00000000ffffff46 662e0f0116b8ff  o32 lgdt %cs:0xffb8
00000000ffffff4d 0f20c0          mov %cr0, %eax
00000000ffffff50 0c01            or $0x1, %al
00000000ffffff52 0f22c0          mov %eax, %cr0
00000000ffffff55 fc              cld
00000000ffffff56 b80800          mov $0x8, %ax
00000000ffffff59 8ed8            mov %ax, %ds
00000000ffffff5b 8ec0            mov %ax, %es
00000000ffffff5d 8ed0            mov %ax, %ss
00000000ffffff5f 8ee0            mov %ax, %fs
00000000ffffff61 8ee8            mov %ax, %gs
00000000ffffff63 66ea66feffff1000 o32 jmp $0x10, $0xfffffe66
```

```
00000000ffffffb8 0047-ffffff70
```

```
00000000ffffff70 00000000 00000000
00000000ffffff78 0000ffff 00cf9300
00000000ffffff80 0000ffff 00cf9b00
00000000ffffff88 0000ffff 00cf9300
00000000ffffff90 0000ffff 00cf9b00
00000000ffffff98 00000000 00000000
00000000ffffffa0 0000ffff 00cf9300
00000000ffffffa8 0000ffff 00af9b00
00000000ffffffb0 00000000 00000000
```

Le bios commence par sauter à une routine permettant de passer en mode protégé. La GDT est en mode FLAT pour chacun de ses descripteurs de segments. A présent, le processeur est en mode protégé.

Fonction principale du bios

```
00000000fffffe66 b860000080       mov $0x80000060, %eax
00000000fffffe6b 66baf80c         mov $0xcf8, %dx
00000000fffffe6f ef               out %eax, %dx
00000000fffffe70 66bafc0c         mov $0xcfc, %dx
00000000fffffe74 b804000000       mov $0x4, %eax
00000000fffffe79 ef               out %eax, %dx
00000000fffffe7a ed               in %dx, %eax
00000000fffffe7b 0d010000f8       or $0xf8000001, %eax
00000000fffffe80 ef               out %eax, %dx
```

Le controlleur mémoire du processeur est initialisé. Il est configurable à travers l'espace PCI en utilisant les I/O. Son identifiant est : `B0:D0:F0`. Pour obtenir ces informations, il suffit d'appliquer les formules suivantes :

```
address = 0x80000060
B = (address - 0x80000000) >> 16
D = ((address - 0x80000000) >> 11) & 31
F = ((address - 0x80000000) >>  8) & 7
register = (address - 0x80000000) & 255
```

Dans son espace de configuration, le registre `0x60` correspond au `PCIEXBAR`. L'espace MMIO est donc configuré pour être adressé en `0xf8000000`. Le `4` signifie que seulement `64Mo` seront adressables, ce qui est cohérent avec les composants dans le portable et ce qui évite les gaspillage de mémoire. Le pseudo-code est le suivant :

```
io(0xcf8) = 0x80000060
io(0xcfc) = 0x4
io(0xcfc) = io(0xcfc) | 0xf8000001
```

Ces informantions sont disponibles dans le document :

> Mobile 3rd Generation Intel CoreTM Processor Family, Mobile Intel Pentium Processor Family, and Mobile Intel Celeron Processor Family

---

```
00000000fffffe81 bff0800ff8       mov $0xf80f80f0, %edi
00000000fffffe86 c70701c0d1fe     mov $0xfed1c001, (%edi)
00000000fffffe8c bf10f4d1fe       mov $0xfed1f410, %edi
00000000fffffe91 8327fb           and $0xfb, (%edi)
```

L'espace `0xf80f80f0` correspond au composant `B0:D0:F0`. Pour obtenir cette information, il suffit d'appliquer les formules suivantes :

```
address = 0xf80f80f0
PCIEXBAR = 0xf8000000
R = address % 4096
F = ((address - PCIEXBAR) / 4096) % 8
D = ((address - PCIEXBAR) / (4096 * 8)) % 32
B = ((address - PCIEXBAR) / (4096 * 8 * 32))
```

Ce composant correspond à l'interface avec le bus LPC (*LPC Interface Bridge Registers*). Le registre `0xf0` est l'adresse du RCBA (*Root Complex Base Address*). Cette zone contient les registres de configuration du chipset. Elle est à présent accessible à l'adresse `0xfed1c000`. L'offset `0xfed1f410 - 0xfed1c000` correspond au registre `0x3410` de cette zone, le GCS (*General Control and Status*). En masquant ce registre avec `0xfffffffb`, le bios positionne à `0` le bit RPR : **TODO : mieux comprendre ce bit**

> Reserved Page Route (RPR) — R/W. Determines where to send the reserved page registers. These addresses are sent to PCI or LPC for the purpose of generating POST codes. The I/O addresses modified by this field are: 80h, 84h, 85h, 86h, 88h, 8Ch, 8Dh, and 8Eh.
>
> 0 = Writes will be forwarded to LPC, shadowed within the PCH, and reads will be returned from the internal shadow
>
> 1 = Writes will be forwarded to PCI, shadowed within the PCH, and reads will be returned from the internal shadow.
>
> NOTE: if some writes are done to LPC/PCI to these I/O ranges, and then this bit is flipped, such that writes will now go to the other interface, the reads will not return what was last written. Shadowing is performed on each interface. The aliases for these registers, at 90h, 94h, 95h, 96h, 98h, 9Ch, 9Dh, and 9Eh, are always decoded to LPC.

Ces informantions sont disponibles dans le document :

> Intel 6 Series Chipset and Intel C200 Series Chipset

```
00000000fffffe94 66b80100          mov $0x1, %ax
00000000fffffe98 66e780            out %ax, $0x80
```

Le port `0x80` semble être utilisé comment port de diagnostic :

> I/O port 0x80 is traditionally used for POST Codes. (POST = Power On Self Test)

```
00000000fffffe62 fffffea5
```

```
00000000fffffe9b bc62feffff        mov $0xfffffe62, %esp
00000000fffffea0 e9e8feffff        jmp 0xfffffd8d
```

Les instructions aux adresses `0xfffffe9b` et `0xfffffea0` correspondent à un `call`. Par contre, au lieu de laisser le processeur empiler l'adresse de retour, cette dernière est définie statiquement à l'adresse *a0x`fffffe62`*. Après l'exécution de la routine à l'adresse `0xfffffd8d` l'exécution se poursuivra à l'adresse `*0xfffffe62 == 0xfffffea5`.

```
00000000fffffea5 0bc0              or %eax, %eax
00000000fffffea7 740c              jz 0xfffffeb5
00000000fffffea9 b979000000        mov $0x79, %ecx
00000000fffffeae 33d2              xor %edx, %edx
00000000fffffeb0 83c030            add $0x30, %eax
00000000fffffeb3 0f30              wrmsr
00000000fffffeb5 bfdc800ff8        mov $0xf80f80dc, %edi
00000000fffffeba 830f08            or $0x8, (%edi)
```

```
00000000fffffebd b9a0010000       mov $0x1a0, %ecx
00000000fffffec2 0f32             rdmsr
00000000fffffec4 0fbaf016         btr $0x16, %eax
00000000fffffec8 7302             jae 0xfffffecc
00000000fffffeca 0f30             wrmsr
00000000fffffecc b91b000000       mov $0x1b, %ecx
00000000fffffed1 0f32             rdmsr
00000000fffffed3 83e2f0           and $0xf0, %edx
00000000fffffed6 25ff0f0000       and $0xfff, %eax
00000000fffffedb 0d0000e0fe       or $0xfee00000, %eax
00000000fffffee0 0f30             wrmsr
00000000fffffee2 0f20e0           mov %cr4, %eax
00000000fffffee5 0d00060000       or $0x600, %eax
00000000fffffeea 0f22e0           mov %eax, %cr4
00000000fffffeed b003             mov $0x3, %al
00000000fffffeef e680             out %al, $0x80
00000000fffffef1 ba52657250       mov $0x50726552, %edx
00000000fffffef6 b073             mov $0x73, %al
00000000fffffef8 e6b2             out %al, $0xb2
00000000fffffefa e684             out %al, $0x84
00000000fffffefc e684             out %al, $0x84
00000000fffffefe 0ac0             or %al, %al
00000000ffffff00 750c             jnz 0xffffff0e
00000000ffffff02 66baf90c         mov $0xcf9, %dx
00000000ffffff06 b002             mov $0x2, %al
00000000ffffff08 ee               out %al, %dx
00000000ffffff09 b006             mov $0x6, %al
00000000ffffff0b ee               out %al, %dx
00000000ffffff0c ebfe             jmp 0xffffff0c
00000000ffffff0e e9ddfbffff       jmp 0xfffffaf0
```