INT 15h, AX=E820h - Query System Address Map

Real mode only.

This call returns a memory map of all the installed RAM, and of physical memory ranges reserved by the BIOS. The address map is returned by making successive calls to this API, each returning one "run" of physical address information. Each run has a type which dictates how this run of physical address range should be treated by the operating system.

If the information returned from INT 15h, AX=E820h in some way differs from INT 15h, AX=E801h or INT 15h AH=88h, then the information returned from E820h supersedes what is returned from these older interfaces. This allows the BIOS to return whatever information it wishes to for compatibility reasons.

Input:

EAX	Function Code	E820h
EBX	Continuation	Contains the "continuation value" to get the next run of physical memory. This is the value returned by a previous call to this routine. If this is the first call, EBX must contain zero.
ES:DI	Buffer Pointer	Pointer to an Address Range Descriptor structure which the BIOS is to fill in.
ECX	Buffer Size	The length in bytes of the structure passed to the BIOS. The BIOS will fill in at most ECX bytes of the structure or however much of the structure the BIOS implements. The minimum size which must be supported by both the BIOS and the caller is 20 bytes. Future implementations may extend this structure.
EDX	Signature	'SMAP' - Used by the BIOS to verify the caller is requesting the system map information to be returned in ES:DI.

Output:

CF	Carry Flag	Non-Carry - indicates no error
EAX	Signature	'SMAP' - Signature to verify correct BIOS revision.
ES:DI	Buffer Pointer	Returned Address Range Descriptor pointer. Same value as on input.
ECX	Buffer Size	Number of bytes returned by the BIOS in the address range descriptor. The minimum size structure returned by the BIOS is 20 bytes.
EBX	Continuation	Contains the continuation value to get the next address descriptor. The actual significance of the continuation value is up to the discretion of the BIOS. The caller must pass the continuation value unchanged as input to the next iteration of the E820 call in order to get the next Address Range Descriptor. A return value of zero means that this is the last descriptor. Note that the BIOS indicate that the last valid descriptor has been returned by either returning a zero as the continuation value, or by returning carry.

Address Range Descriptor Structure

Offset in Bytes	Name	Description
0	BaseAddrLow	Low 32 Bits of Base Address
4	BaseAddrHigh	High 32 Bits of Base Address
8	LengthLow	Low 32 Bits of Length in Bytes
12	LengthHigh	High 32 Bits of Length in Bytes
16	Type	Address type of this range.

The *BaseAddrLow* and *BaseAddrHigh* together are the 64 bit *BaseAddress* of this range. The *BaseAddress* is the physical address of the start of the range being specified.

The *LengthLow* and *LengthHigh* together are the 64 bit *Length* of this range. The *Length* is the physical contiguous length in bytes of a range being specified.

The *Type* field describes the usage of the described address range as defined in the table below.

Value	Pneumonic	Description
1	AddressRangeMemory	This run is available RAM usable by the operating system.
2	AddressRangeReserved	This run of addresses is in use or reserved by the system, and must not be used by the operating system.
Other	Undefined	Undefined - Reserved for future use. Any range of this type must be treated by the OS as if the type returned was AddressRangeReserved.

The BIOS can use the AddressRangeReserved address range type to block out various addresses as "not suitable" for use by a programmable device.

Some of the reasons a BIOS would do this are:

- The address range contains system ROM.
- The address range contains RAM in use by the ROM.
- The address range is in use by a memory mapped system device.
- The address range is for whatever reason are unsuitable for a standard device to use as a device memory space.

Assumptions and Limitations

- **1.** The BIOS will return address ranges describing base board memory and ISA or PCI memory that is contiguous with that baseboard memory.
- **2.** The BIOS WILL NOT return a range description for the memory mapping of PCI devices, ISA Option ROM's, and ISA plug & play cards. This is because the OS has mechanisms available to detect them.
- **3.** The BIOS will return chipset defined address holes that are not being used by devices as reserved.
- **4.** Address ranges defined for base board memory mapped I/O devices (for example APICs) will be returned as reserved.
- **5.** All occurrences of the system BIOS will be mapped as reserved. This includes the area below 1 MB, at 16 MB (if present) and at end of the address space (4 gig).
- **6.** Standard PC address ranges will not be reported. Example video memory at A0000 to BFFFF physical will not be described by this function. The range from E0000 to EFFFF is base board specific and will be reported as suits the bas board.
- **7.** All of lower memory is reported as normal memory. It is OS's responsibility to handle standard RAM locations reserved for specific uses, for example: the interrupt vector table(0:0) and the BIOS data area(40:0).

Example address map

This sample address map describes a machine which has 128 MB RAM, 640K of base memory and 127 MB extended. The base memory has 639K available for the user and 1K for an extended BIOS data area. There is a 4 MB Linear Frame Buffer (LFB) based at 12 MB. The memory hole created by the chipset is from 8 M to 16 M. There are memory mapped APIC devices in the system. The IO Unit is at FEC00000 and the Local Unit is at FEE00000. The system BIOS is remapped to 4G - 64K.

Note that the 639K endpoint of the first memory range is also the base memory size reported in the BIOS data segment at 40:13.

Key to types: "ARM" is AddressRangeMemory, "ARR" is AddressRangeReserved.

```
Base (Hex)      Length  Type    Description
0000 0000       639K    ARM     Available Base memory - typically the same
                                value as is returned via the INT 12 function.
0009 FC00       1K      ARR     Memory reserved for use by the BIOS(s).
                                This area typically includes the Extended
                                BIOS data area.
000F 0000       64K     ARR     System BIOS
0010 0000       7M      ARM     Extended memory, this is not limited to
                                the 64 MB address range.
0080 0000       8M      ARR     Chipset memory hole required to support the
                                LFB mapping at 12 MB.
0100 0000       120M    ARM     Base board RAM relocated above a chipset
                                memory hole.
FEC0 0000       4K      ARR     IO APIC memory mapped I/O at FEC00000.  Note
                                the range of addresses required for an APIC
                                device may vary from base OEM to OEM.
FEE0 0000       4K      ARR     Local APIC memory mapped I/O at FEE00000.
FFFF 0000       64K     ARR     Remapped System BIOS at end of address space.
```

Sample operating system usage

The following code segment is intended to describe the algorithm needed when calling the Query System Address Map function. It is an implementation example and uses non standard mechanisms.

```
    E820Present = FALSE;
    Regs.ebx = 0;
    do {
        Regs.eax = 0xE820;
        Regs.es  = SEGMENT (&Descriptor);
        Regs.di  = OFFSET  (&Descriptor);
        Regs.ecx = sizeof  (Descriptor);
       Regs.edx = 'SMAP';

        _int( 0x15, Regs );

        if ((Regs.eflags & EFLAG_CARRY)  ||  Regs.eax != 'SMAP') {
            break;
        }

        if (Regs.ecx < 20  ||  Regs.ecx > sizeof (Descriptor) ) {
            // bug in bios - all returned descriptors must be
            // at least 20 bytes long, and can not be larger then
            // the input buffer.

            break;
        }

        E820Present = TRUE;
        .
        .
```

```
        .
        Add address range Descriptor.BaseAddress through
        Descriptor.BaseAddress + Descriptor.Length
        as type Descriptor.Type
        .
        .
        .

    } while (Regs.ebx != 0);

    if (!E820Present) {
        .
        .
        .
        call INT 15h, AX=E801h and/or INT 15h, AH=88h to obtain old style
        memory information
        .
        .
        .
    }
```

INT 15h, AX=E801h - Get Memory Size for Large Configurations

Real mode only (as far as I know).

Originally defined for EISA servers, this interface is capable of reporting up to 4 GB of RAM. While not nearly as flexible as E820h, it is present in many more systems.

Input:

```
        AX      Function Code   E801h
```

Output:

```
        CF      Carry Flag      Non-Carry - indicates no error
        AX      Extended 1      Number of contiguous KB between 1 and 16 MB,
                                maximum 0x3C00 = 15 MB.
        BX      Extended 2      Number of contiguous 64 KB blocks between
                                16 MB and 4 GB.
        CX      Configured 1    Number of contiguous KB between 1 and 16 MB,
                                maximum 0x3C00 = 15 MB.
        DX      Configured 2    Number of contiguous 64 KB blocks between
                                16 MB and 4 GB.
```

Not sure what this difference between the "Extended" and "Configured" numbers are, but they appear to be identical, as reported from the BIOS.

NOTE: It is possible for a machine using this interface to report a memory hole just under 16 MB (Count 1 is less than 15 MB, but Count 2 is non-zero).

INT 15h, AH=88h - Get Extended Memory Size

Real mode only.

This interface is quite primitive. It returns a single value for contiguous memory above 1 MB. The biggest limitation is that the value returned is a 16-bit value, in KB, so it has a maximum saturation of just under 64 MB even presuming it returns as much as it can. On some systems, it won't return anything above the 16 MB boundary.

The one useful point is that it works on every PC available.

Input:

```
        AH      Function Code   88h
```

Output:

```
        CF      Carry Flag      Non-Carry - indicates no error
        AX      Memory Count    Number of contiguous KB above 1 MB.
```

---

erich@uruk.org