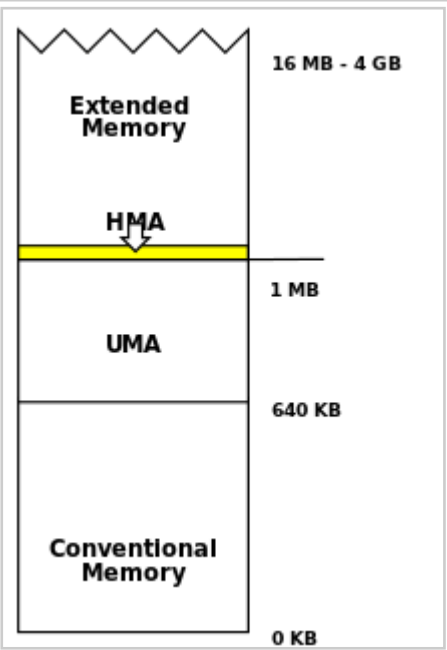
A20 line

From Wikipedia, the free encyclopedia

The **A20** or *addressing line 20* is one of the electrical lines that make up the system bus of an x86-based computer system. The A20 line in particular is used to transmit the 21st bit on the address bus.

A microprocessor will typically have a number of addressing lines equal to the base-two logarithm of its physical addressing space. For example, a processor with 4 gigabytes of physical addressing space requires 32 lines, which are named A0 through A31. The lines are named after the zero-based number of the bit in the address they are transmitting, where the least significant bit is first, therefore numbered bit 0, and signaled on line A0. A20 transmits bit 20 (the 21st bit) and will become active once addresses reach 1 megabyte or 2^{20}.

Contents



The high memory area is available in real mode on 80286 processors if the A20 gate is not disabled

History

The early Intel 8086, Intel 8088, and Intel 80186 processors had 20 address lines, numbered A0 to A19; with these, the processor can access 2^{20} bytes or 1 megabyte. Internal address registers of these processors only had 16 bits. To access a 20-bit address space, an external memory reference was made up of a 16-bit *Offset* address added to a 16-bit *Segment* number, shifted 4 bits so as to produce a 20-bit physical address. The resulting address is equal to Segment * 16 + Offset. There are many combinations of segment and offset that produce the same 20-bit physical address. In consequence there were various ways to address the same byte in memory. For example, here are four of the 4096 different segment:offset combinations, all referencing the byte whose physical address is 0x000FFFFF (the last byte in 1 MB-memory space):

F000:FFFF
FFFF:000F

F555:AAAF
F800:7FFF

Referenced the last way, an increase of one in the offset yields F800:8000, which is a proper address for the processor, but since it translates to the physical address 0x00100000 (the first byte over 1 MB) the processor would need another address-line to actually access this byte. Since such a line doesn't exist on the 8086 line of processors, the 21st bit above, while set, gets dropped, causing the address F800:8000 to "wrap around" and to actually point to the physical address 0x00000000.

In order to improve performance, a trick was used by some DOS programmers, for example, to have one segment that has access to program data (e.g. from F800:0000 to F800:7FFF, pointing to the physical addresses 0x000F8000 - 0x000FFFFF) as well as the I/O data (e.g. keyboard buffer) that was located in the first memory segment (with addresses F800:8000 to F800:FFFF pointing to the physical addresses 0x00000000 to 0x00007FFF).. The address wrap was also used by MS-DOS itself, to implement the CALL 5 entry point in the Program Segment Prefix; the CALL 5 handler was at physical address 0x000000C0 but for CP/M compatibility, its offset had to match the segment size (usually 0xFEF0). The only way to reconcile these was to choose a segment value that, when added to 0xFEF0, resulted in a physical address of 0x001000C0, which would wrap around to 0x000000C0.

When IBM designed the IBM PC AT machine, it decided to use the new higher-performance Intel 80286 microprocessor. The 80286 could address up to 16 megabytes of system memory in protected mode. However, the CPU was supposed to emulate an 8086's behavior in real mode, its startup mode, so it could run operating systems and programs that were not written for protected mode. The 80286 had a bug where it failed to force the A20 line to zero in real mode. Due to this bug, the combination F800:8000 would no longer point to the physical address 0x00000000 but the correct address 0x00100000. As a result, some DOS programs would no longer work. In order to remain compatible with these programs, IBM decided to fix the problem on the motherboard.

This was accomplished by inserting a logic gate on the A20 line between the processor and system bus, which got named *Gate-A20*. Gate-A20 can be enabled or disabled by software to allow or prevent the address bus from receiving a signal from A20. It is set to non-passing for the execution of older programs which rely on the wrap-around. At boot time, the BIOS first enables Gate-A20 when counting and testing all of the system's memory, and disables it before transferring control to the operating system.

Originally, the logic gate was a gate connected to the Intel 8042 keyboard controller. Controlling it was a relatively slow process. Other methods have since been added to allow for more efficient multitasking of programs which require this wrap-around with programs that access all of the system's memory. There was at first a variety of methods, but eventually the industry settled on the PS/2 method of using a bit in port 92h to control the A20 line.

Disconnecting A20 would not wrap **all** memory accesses above 1 MiB, just those in the 1 MiB-2 MiB, 3 MiB-4 MiB, 5 MiB-6 MiB, etc. ranges. Real mode software only cared about the area slightly above 1 MiB, so Gate-A20 was enough.

Enabling the Gate-A20 line is one of the first steps a protected mode x86 operating system does in the bootup process, often before control has been passed onto the kernel from the bootstrap (in the case of Linux, for example).

Virtual 8086 mode, introduced with the Intel 80386, allows the A20 wrap-around to be simulated by using the virtual memory facilities of the processor: physical memory may be mapped to multiple virtual addresses thus allowing the memory mapped at first mebibyte of virtual memory may be mapped again in the second mebibyte of virtual memory. The operating system may intercept changes to Gate A20 and make corresponding changes to the virtual memory address space, which also makes irrelevant the efficiency of Gate-A20 toggling.

A20 handler

The **A20 handler** is IBM PC memory manager software controlling access to the high memory area (HMA). Extended memory managers usually provide this functionality. A20 handlers are named after the 21st address line of the microprocessor, the A20 line.

In MS-DOS, high memory area managers, such as HIMEM.SYS has the "extra task" of managing A20. HIMEM.SYS provided an API for opening/closing A20. DOS itself could utilize the area for some of its storage needs, thereby freeing up more conventional memory for programs. This functionality was enabled by the "DOS=HIGH" directive in the CONFIG.SYS configuration file.

A20 gate

Controlling the A20 line was an important feature at one stage in the growth of the IBM PC architecture, as it added access to an additional 65520 bytes (64 kilobytes - 16) of memory in real mode without dramatic software changes.

In what was arguably a "hack", the A20 gate was originally part of the keyboard controller on the motherboard, which could open/close it depending on what behavior was desired.[1] The A20 gate is still present on many modern PCs and the gate is initially closed right after boot. Modern protected mode operating systems typically open the A20 gate early during the boot process and never close it again. These operating systems do not have the compatibility reasons for keeping it closed and they gain access to the full range of physical addresses available by opening it.

The Intel 80486 and Pentium added a special pin named *A20M#*, which when asserted low forces bit 20 of the physical address to be zero for all on-chip

cache or external memory accesses. This was necessary since the 80486 introduced an on-chip cache, and therefore masking this bit in external logic was no longer possible. Software still needs to manipulate the gate and must still deal with external peripherals (the chipset) for that.[2]

Support for the A20 gate was changed in the Nehalem microarchitecture (some sources incorrectly claim A20 support was removed). Rather than the CPU having a dedicated A20M# pin which receives the signal whether or not to mask the A20 bit, this has been virtualized so that the information is sent from the peripheral hardware to the CPU using special bus cycles. From a software point of view, the mechanism works exactly as before, and an operating system must still program external hardware (which in-turn sends the aforementioned bus cycles to the CPU) to disable the A20 masking.

Intel no longer supports the A20 gate starting with Haswell. Page 271 of the Intel System Programmers Manual Vol. 3A from June 2013 states: "The functionality of A20M# is used primarily by older operating systems and not used by modern operating systems. On newer Intel 64 processors, A20M# may be absent."[3]

See also

- Computer storage
- High memory area (HMA)

References

1. Tom Shanley, Don Anderson, John Swindle, ISA system architecture*, Addison-Wesley, 1995, ISBN 0-201-40996-8, pp.79-80*
2. Tom Shanley, *Protected mode software architecture*, Taylor & Francis, 1996, ISBN 0-201-55447-X, page 60
3. Intel System Programmers Manual Vol. 3A from June 2013 (http://download.intel.com/products/processor/manual/253668.pdf)

External links

- A20 - a pain from the past (http://www.win.tue.nl/~aeb/linux/kbd/A20.html)

Categories: X86 memory management | Memory expansion | X86 architecture | IBM PC compatibles

---