Internal Cache Architecture of X86 Processors

By: K. Goodnow, Ph.D.

Introduction

This paper is intended as a reference for the commonalities and differences in internal cache structures in X86 style processors. The Intel® 486DX and IBM Blue Lightning processors are studied and compared to highlight the various strategies in cache design. The basic structure of these caches and particular instances are discussed.

All of these processors contain an area of the die dedicated to a static cache memory. This memory is connected directly to the internal processing blocks which allows extremely fast access to their contents. This is especially important in double and triple clocked internal processors where the processing blocks are running two and three times the speed of the external bus.

Cache Architecture

All of the processors studied utilize a unified 4 way set

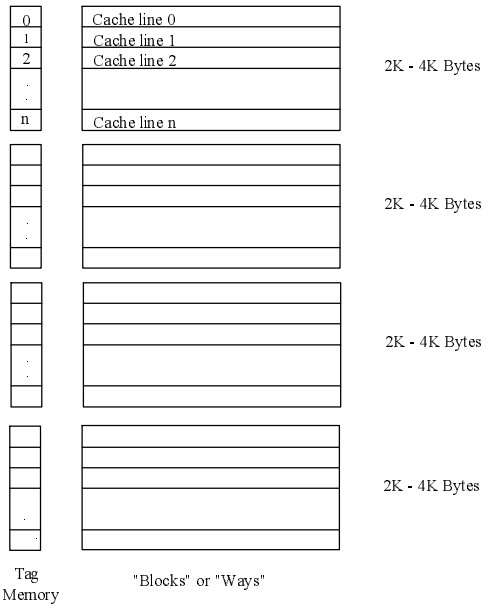


Figure 1. 4-Way Associative Cache

associative cache organization. The unified aspect refers to storing both data and instruction information in the same memory. The four-way set associative aspect refers to a cache organization that is divided into four

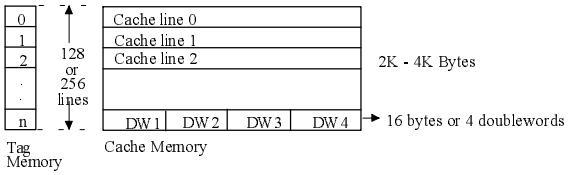


Figure 2: "Way" Organization

blocks or ways of memory (see figure 1). Each block is then divided into individual lines or sets (see figure 2). Each line is associated with the similar lines in the other three blocks of memory. Each line can store a certain number of bytes of information (in the case of these processors 16 bytes). The total size of the cache is determined by the number of bytes on a line or set, the number of sets, and the number of blocks or ways.

Given the specifications for a cache the number of lines can be found. For example, in an 8K cache which is four-way set associative, with a line size of 16 bytes, the number of lines or sets is:

$$\frac{8\text{K bytes}}{4\text{ Way}} = 2\text{K bytes/way}$$

Now using the number of bytes in a line we find:

$$\frac{2\text{K bytes/way}}{16\text{bytes/line}} = 128\text{ lines/way}$$

Therefore each block of memory in the cache would contain 128 lines.

The physical address of the data in memory is used as

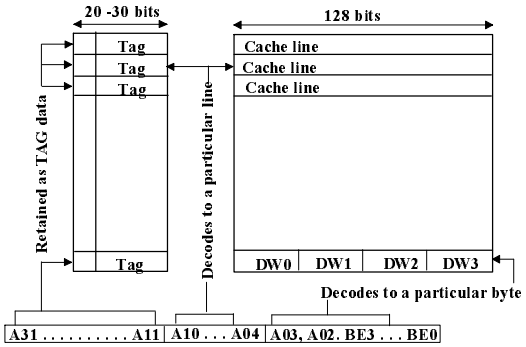


Figure 3: Physical Address Mapping

an index into the cache (see figure 3). Address bits A3-A0 (or A3-A2 and BE3-BE0) defines a particular byte out of the 16 bytes in a line. The next 7-8 bits decode to a particular line or set of the memory block. The top bits are then stored as a tag in the tag memory

CATEGORY	BLSX2/SX3	486DX2	IDX2	IDX4
SIZE	16K	8K	8K	16K
PARITY	YES	NO	NO	NO
TYPE	UNIFIED 4-WAY SET ASSOCIATIVE	UNIFIED 4-WAY SET ASSOCIATIVE	UNIFIED 4-WAY SET ASSOCIATIVE	UNIFIED 4-WAY SET ASSOCIATIVE
WRITE THROUGH	YES	YES	YES	YES
WRITE BACK	NO	YES	NO	NO
WRITE LEVELS	2	8	4	4
LINE SIZE	16 BYTES	16 BYTES	16 BYTES	16 BYTES
FLUSH TYPES	FULL SNOOP ADS FLUSH HOLD FLUSH SOFTWARE	FULL SNOOP FLUSH PIN SOFTWARE	FULL SNOOP FLUSH PIN SOFTWARE	FULL SNOOP FLUSH PIN SOFTWARE
POWER UP STATUS	OFF	OFF	OFF	OFF
FLUSH ON CLOCK STOP	YES	NO	YES	YES
FULL SNOOP ON CLOCK STOP	YES	NO	YES	YES
CACHE TESTING	NO	YES	YES	YES

Table 1: Cache Architecture Comparison

for that line. Each memory block has a 128 or 256 tag line memory associated with it. When checking to see if a particular information byte is stored in the cache, the line addresses (A11-A4) are decoded to determine the tag line memory location to extract the stored address tag and then this tag is compared with the needed physical address upper bits. If there is a match, the lower address bits are decoded to find the right bytes or double word.

The 7-8 address bits that are the index to the line or set of a way are repeated every 2048 or 4096 bytes throughout memory. Every repeat of these 7-8 bits can only be stored in that particular line of one of the four "ways." This is the main limitation of the four way set associative cache memory architecture. If the processor asks for five bytes which are each 2048 or 4096 address locations away from each other, only four can be held in cache memory, even if the rest of cache memory is empty.

The line tag and byte addresses are common across all four of the blocks in the cache, thus the associative nature. A particular piece of information could be in any one of the blocks at that particular line. The tag information is the piece that must be checked to ascertain if the data is present and in which particular block.

This cache architecture is common across all of the processors, the differences are present in the size of the cache, the method of reading and writing to the cache from the external bus, the invalidation and flushing of the cache, and the method of cache coherency during power down mode. These areas will be covered in the following sections. Table 1 shows a comparison of the different cache architectures available in these processors.

Cache Size Differences

The IBM 486DX2 and the Intel 486DX2 each contain an 8K cache. This means a line or set size of 128. The IBM Blue Lightning 486SX2 and 486SX3 and the Intel 486DX4 each contain a 16K cache. This 16K cache has a line or set size of 256.

Cache Line Fill

When the cache is enabled and the processor requests data or instruction information that is not present in the cache, a line fill read is generated. If the line is cacheable the bytes currently requested are fetched from the system along with the rest of the bytes on that line. One of the first places we see a difference in the processors is in the algorithm used to determine the next set of bytes to request. The Intel 486DX2 and 486DX4 use an

Intel specific algorithm that maximizes the double bank memory accessing of most motherboards. These sequences are shown in table 2.

1st	2nd	3rd	4th
0	4	8	C
4	0	C	8
8	C	0	4
C	8	4	0

Table 2: Cache Line Fill Order

The IBM Blue Lightning 486SX2 and 486SX3 use an algorithm that maximizes the hit rate for an access to the next op-code in it's determination of which set of bytes to request next. These sequences are shown in table 3.

1st	2nd	3rd	4th
0	4	8	C
4	8	C	0
8	C	0	4
C	0	4	8

Table 3: IBM Blue Lightning 486SX2 and 486SX3 Cache Line Fill Order

The cacheability of a particular line of information is determined by several methods. The KEN pin is available on all of the processors and allows a direct control of the cacheability of the line. When set, this pin signals to the processor that the current address is cacheable. If any byte in a line is un-cacheable, the entire line is considered un-cacheable.

This "KEN" pin on the Blue Lightning 486SX2 and 486SX3 is shared with a dynamic frequency shifting function and care must be taken in the system design to insure that this pin is used as intended for that particular system.

The cacheability of a line can also be determined through software. All of the processors support page table cache control. This is a table of the physical addresses and their mapping. One bit of this table, the PCD bit, indicates if this particular page is cacheable. The PCD bit can be set or reset by the software of the system. Care must be taken when using the PCD bit that a cache coherency problem does not arise. As stated in the IBM Blue Lightning manual, "If the PCD bit is changed dynamically, the cache needs to be flushed to ensure that the processor will go out to main storage for the data." Most of the time this page table will be under operating system control.

The IBM Blue Lightning 486SX2 and 486SX3 also support the use of a microprocessor specific register (MSR) that controls the cacheability of main memory in 64K regions below 1MB and an upper cacheability limit for memory above 1MB. This allows a coarser control of the caching of memory separate from the other methods. This method is suitable for many DOS type applications.

Cache Line Replacement

If a cache line fill request is valid, the processor attempts to place the data in one of the four "ways" at that particular line. If all four of the "ways" have that line occupied a least-recently-used (LRU) algorithm is used to determine which "way" will be overwritten by the new data. The LRU algorithm uses several bits of memory that are contained in the tag line memory. These bits are changed when there is a memory access on that line on one of the four "ways." The processor will overwrite a line in the cache if it is the least recently used among the four "ways."

The IBM 486DX2 processor has three external pins, RPLVAL#, RPLSET1 and RPLSET0 which indicate the particular "way" that is being invalidated on a new cache line fill. If the pin RPLVAL# is set valid it indicates that the values on pins RPLSET1 and RPLSET0 indicate the particular "way" that is being overwritten. These pins along with the current external address allow an external hardware system to duplicate the internal cache memory data in real time.

Cache Flushing

The ability to maintain cache coherency with different "Bus Masters" writing to memory is handled by invalidating or flushing the internal cache memory. This can be accomplished in two ways, invalidating a particular line of a single "way" or by invalidating the entire cache memory. The first approach of invalidating a single line is accomplished by checking each valid write address that appears on the bus while another bus master controls the bus. Since the entire address needs to be used, this method is called full snooping. This method has the least impact on system performance since only the particular line of that address cycle is invalidated.

The second method invalidates the entire cache memory. This can be accomplished in hardware or software in all of the processors. The software method relies on

the use of the invalidate command in a software program. The hardware method uses an external signal to tell the processor to flush the cache. On the IBM 486DX2, Intel 486DX2 and Intel 486DX4 this external signal is driven on the FLUSH pin.

On the IBM Blue Lightning 486SX2 and 486SX3 a full cache flush is accomplished as either an ADS# or HOLD flush. The type of flush is set as bits in the MSR1000 register. If the full snoop bits are set the processor will accomplish single line invalidates on a hold. With the bits set for a HOLD flush, the assertion of HOLD and HLDA will flush the cache. With ADS# flush set, the processor will flush on HOLD, HLDA, and ADS# being active. This allows a differentiation between bus master and memory refresh cycles.

The CD bit in the control register of the IBM 486DX2, Intel 486DX2, and Intel 486DX4 as well as the CE bit in the MSR1000 registers of the IBM Blue Lightning 486SX2 and 486SX3 enable and disable the cache. Disabling the cache causes new non-cached reads to not enter the cache. If the cache was previously used and contains information and if the current address is in the cache, the information will be read out of the cache not system memory. To disable and not use the cache, the CE or CD bit must be cleared and then the entire cache flushed. This flushing can be accomplished through either the software or hardware methods discussed.

Write Operations

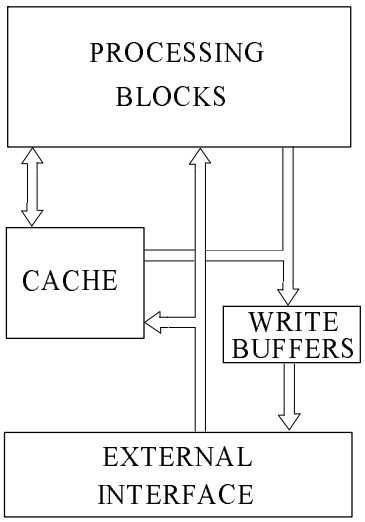
The second type of memory operation is a write into cache from the processing blocks and a write out to system memory from the processor. There are two write modes used in these X86 processors; Write-Through and Write-Back. Write-through operation updates the cache memory (if cacheable) and also writes the data out to system memory at the same time. A write-back operation updates the cache (if cacheable) and does not update the system memory until the modified line is to be overwritten, the entire cache flushed, or an external bus master requires access. This second method means that writes to a cacheable region stay within the processor. This reduces the bus activity and subsequently increases the processor performance.

The IBM Blue Lightning 486SX2 and 486SX3, Intel 486DX2, and Intel 486DX4 processors utilize the write-through mode. The IBM 486DX2 can be set into either write-through or write-back mode. The write-through mode will be discussed first since it is common across all five processors.

Write-Through Mode

This method of writing from the processor blocks is the least complicated in regards to cache coherency. The internal cache and the external system memory are updated at the same time. This method suffers from decreased performance since an external bus cycle (usually much slower than the internal cycle time) must be generated for every write. If the processor were to wait until the external write operation completed before continuing, then a write would effectively halt the operation of the processor. In order to avoid this bottleneck, write buffers are added to the architecture of the processor, see figure 4. These write buffers hold the data, address, and status bits required for the memory cycle. Once the internal processing block passes this information to the write buffer, it can continue processing. The only time it would have to wait is if the write buffers were full. For this reason the addition of write buffers to the architecture becomes a trade-off between performance and silicon area. The IBM Blue Lightning 486SX2 and 486SX3 have two write buffers while the Intel processors have four write buffers apiece. The IBM 486DX2 contains eight write buffers.

Internally, a write from the processing block is handled as a line modify operation if that physical address is



currently stored in the cache. If the data is cacheable and is not currently in the cache, the data is written out to system memory. On the Blue Lightning SX2 and SX3 a line fill request for that physical address is then made. This approach, known as write-allocate, causes

the system memory to be updated first and then that line to be read into the cache.

Although complicated internally, the write-through mode does not normally require any system design considerations since the effect is that of a non-cached write operation.

Write-Back Mode

The write-back mode of operation creates a higher performing processor. Writes to cached memory are at the speed of the internal processing blocks and are not dependent on the external memory cycle time or the number of write buffers. The IBM 486DX2 is placed in this mode by setting the CD bit equal to 0 and the NW bit to 1 in the CR0 register after reset.

The main problem with the write-back mode is cache coherency. The data in the internal cache could be more current then the system memory. If a bus master requires that particular information, the data must be taken out of the internal cache and written to system memory before allowing the bus master access to that data. If the cache is to be invalidated or flushed, all of the modified data must first be written to system memory before invalidating the cache.

When the data is modified on a line in cache memory, if only some of the data is modified it is marked as partially modified. This is done on four byte boundaries. Thus, there are four areas to a line or four double-words that can be identified as modified. This means that when the data needs to be written out to system memory only those particular double-words that were modified need be accessed.

Three methods are available in the IBM 486DX2 processor for cache coherency with the write-back mode. The first two methods require that all modified memory locations be written to system memory before releasing the bus to another bus master. The first method accomplishes this by asserting the FLUSH# pin before asserting HOLD to the processor. The processor will write all previously modified data out of the cache before asserting HLDA. The cache will be retained intact, unless the INVAL pin is asserted when FLUSH# is sampled by the processor in which case the entire cache is invalidated. Otherwise the cache continues to be used as if nothing occurred.
The second method uses the BARB bit in the CCR2 register. If this bit is set all modified cache locations are written out to system memory when a HOLD is asserted. All of the writes will complete before asserting HLDA. The cache is retained as it was before the start of the operation.

One problem of using either of the first two methods for cache coherency is if some bus HOLD cycles are not memory data cycles. If refresh cycles are considered a bus master access, then the modified writes will be purged from the cache at a fairly high rate (every 15µsec). This may result in little performance gain over the write-through mode.

The third method of cache coherency checks each physical address on the bus during other bus master cycles. If a match is found to a cache location that has been previously modified the processor will write just that line back to system memory. This is accomplished by issuing cache inquiry cycles to the IBM 486DX2. The EADS# and INVAL pins are used to issue a cache inquiry to the processor. If the cache contains the physical address currently on the bus and some of the bytes have been modified, the processor will assert HITM# to signal that the line needs to be written to system memory. The processor then waits for HOLD to be de-asserted at which time it will write out the modified bytes to system memory. This method has the least impact on performance since only those data bytes that are needed are written to system memory. This method does require the ability of the bus master to de-assert HOLD before receiving the information requested and then re-generate the read request.

Bus Master Write to a Modified Location

If a full-snoop cache invalidation is being used, and a bus master writes information to a data location that is currently cached and has been marked as modified, the processor can simply mark the line as invalid and accomplish a cache line fill if the location is needed again. The modified data would have been over-written by the bus master in any case.

Power Down Mode Cache Coherency

All of these processors can utilize some type of power management mode. The processor is shut down to some extent in order to save energy. The question arises, how does the processor maintain cache coherency if a bus master accesses memory during the power down. Each of the processors handles this in a different manner.

The IBM Blue Lightning 486SX2 and 486SX3 are able to snoop on the bus and flush the cache in both of their power down modes. The IBM 486DX2 does not support snooping or flush during power down mode. The Intel 486DX2 and 486DX4 processors support both snooping and flushing during power down.

Cache Test Registers

The internal cache test registers TR3, TR4, and TR5 are present in all of the processors except the IBM Blue Lightning 486SX2 and 486SX3. These registers are used to check the viability of the cache. Testing of the cache should be accomplished with the cache turned off.

Conclusion

This paper has shown the architectural decisions made on a variety of issues. The real answer as to the correctness of these decisions is in the performance of the processor and ultimately the marketplace. For more information regarding a particular feature, please consult the appropriate databook.

References

1. **486DX2 Databook**, IBM Microelectronics, 1994.

2. **Blue Lightning Microprocessor Data Sheet**, IBM Microelectronics, 11/1993.

3. **Cache Tutorial**, Intel Corporation, 1991.

4. **IntelDX4 Processor Data Book**, Intel Corportation, 2/1994.

5. **Microprocessors, Vol. 2**, Intel Corporation, 1994.