6 Cache, Write Buffer and Coprocessors

The chapter describes the ARM processor instruction and data cache, and its write buffer.

Cache, Write Buffer and Coprocessors

6.1 Instruction and Data Cache (IDC)

ARM processor contains a 4Kbyte mixed instruction and data cache. The IDC has 256 lines of 16 bytes (4 words), organized as a 4-way set associative cache, and uses the virtual addresses generated by the processor core. The IDC is always reloaded a line at a time (4 words). It may be enabled or disabled via the ARM processor Control Register and is disabled on **nRESET**.

The operation of the cache is further controlled by the *Cacheable* or C bit stored in the Memory Management Page Table (see the Memory Management Unit chapter). For this reason, in order to use the IDC, the MMU must be enabled. The two functions may however be enabled simultaneously, with a single write to the Control Register.

6.1.1 Cacheable bit

The *Cacheable* bit determines whether data being read may be placed in the IDC and used for subsequent read operations. Typically main memory will be marked as Cacheable to improve system performance, and I/O space as Non-cacheable to stop the data being stored in ARM7500FE's cache. [For example if the processor is polling a hardware flag in I/O space, it is important that the processor is forced to read data from the external peripheral, and not a copy of initial data held in the cache]. The Cacheable bit can be configured for both pages and sections.

6.1.2 IDC operation

In the ARM processor the cache will be searched regardless of the state of the C bit, only reads that miss the cache will be affected.

Cacheable Reads	C = 1
	A linefetch of 4 words will be performed and it will be randomly placed in a cache bank.
Uncacheable Reads	C = 0
	An external memory access will be performed and the cache will not be written.

6.1.3 IDC validity

The IDC operates with virtual addresses, so care must be taken to ensure that its contents remain consistent with the virtual to physical mappings performed by the Memory Management Unit. If the Memory Mappings are changed, the IDC validity must be ensured.

Software IDC flush

The entire IDC may be marked as invalid by writing to the ARM processor IDC Flush Register (Register 7). The cache will be flushed immediately the register is written, but note that the next two instruction fetches may come from the cache before the register is written.

6.1.4 Doubly mapped space

Since the cache works with virtual addresses, it is assumed that every virtual address maps to a different physical address. If the same physical location is accessed by more than one virtual address, the cache cannot maintain consistency, since each virtual address will have a separate entry in the cache, and only one entry will be updated on a processor write operation. To avoid any cache inconsistencies, both doubly-mapped virtual addresses should be marked as uncacheable.

6.2 Read-Lock-Write

The IDC treats the Read-Locked-Write instruction as a special case. The read phase always forces a read of external memory, regardless of whether the data is contained in the cache. The write phase is treated as a normal write operation (and if the data is already in the cache, the cache will be updated). Externally the two phases are flagged as indivisible by asserting the **LOCK** signal.

6.3 IDC Enable/Disable and Reset

The IDC is automatically disabled and flushed on **nRESET**. Once enabled, cacheable read accesses will cause lines to be placed in the cache.

6.3.1 To enable the IDC

To enable the IDC, make sure that the MMU is enabled first by setting bit 0 in Control Register, then enable the IDC by setting bit 2 in Control Register. The MMU and IDC may be enabled simultaneously with a single control register write.

6.3.2 To disable the IDC

To disable the IDC, clear bit 2 in the Control Register and perform a flush by writing to the flush register.

6.4 Write Buffer (Wb)

The ARM processor write buffer is provided to improve system performance. It can buffer up to 8 words of data, and 4 independent addresses. It may be enabled or disabled via the W bit (bit 3) in the ARM processor Control Register and the buffer is disabled and flushed on reset.

The operation of the write buffer is further controlled by one bit, B, or Bufferable, which is stored in the Memory Management Page Tables. For this reason, in order to use the write buffer, the MMU must be enabled.

The two functions may however be enabled simultaneously, with a single write to the Control Register. For a write to use the write buffer, both the W bit in the Control Register, and the B bit in the corresponding page table must be set.

6.4.1 Bufferable bit

This bit controls whether a write operation may or may not use the write buffer. Typically main memory will be bufferable and I/O space unbufferable. The Bufferable bit can be configured for both pages and sections.

6.4.2 Write buffer operation

When the CPU performs a write operation, the translation entry for that address is inspected and the state of the B bit determines the subsequent action. If the write buffer is disabled via the ARM processor Control Register, bufferable writes are treated in the same way as unbuffered writes.

Bufferable write

If the write buffer is enabled and the processor performs a write to a bufferable area, the data is placed in the write buffer at FCLK speeds and the CPU continues execution. The write buffer then performs the external write in parallel. If however the write buffer is full (either because there are already 8 words of data in the buffer, or because there is no slot for the new address) then the processor is stalled until there is sufficient space in the buffer.

Unbufferable writes

If the write buffer is disabled or the CPU performs a write to an unbufferable area, the processor is stalled until the write buffer empties and the write completes externally, which may require synchronization and several external clock cycles.

Read-lock-write

The write phase of a read-lock-write sequence is treated as an Unbuffered write, even if it is marked as buffered.

Note: *A single write requires one address slot and one data slot in the write buffer; a sequential write of n words requires one address slot and n data slots. The total of 8 data slots in the buffer may be used as required. So for instance there could be 3 non-sequential writes and one sequential write of 5 words in the buffer, and the processor could continue as normal: a 5th write or an 6th word in the 4th write would stall the processor until the first write had completed.*

To enable the write buffer

To enable the write buffer, ensure the MMU is enabled by setting bit 0 in the Control Register, then enable the write buffer by setting bit 3 in the Control Register. The MMU and write buffer may be enabled simultaneously with a single write to the Control Register.

To disable the write buffer

To disable the write buffer, clear bit 3 in the Control Register.

Note: *Any writes already in the write buffer will complete normally.*

6.5 Coprocessors

The on-chip FPA is a coprocessor and its operation is described in Chapters 8, 9, and 10.

The ARM processor also has an internal coprocessor designated #15 for internal control of the device.

However, the ARM7500FE has no external coprocessor bus, so it is not possible to add further external coprocessors to this device. All coprocessor operations other than those implemented by the FPA, or MRC or MCR to registers 0 to 7 on coprocessor #15, will cause the undefined instruction trap to be taken.

7 ARM Processor MMU

This chapter describes the ARM processor Memory Management Unit.

ARM Processor MMU

7.1 Introduction

The MMU performs two primary functions: it translates virtual addresses into physical addresses, and it controls memory access permissions. The MMU hardware required to perform these functions consists of a Translation Look-aside Buffer (TLB), access control logic, and translation table walking logic.

The MMU supports memory accesses based on Sections or Pages:

- **Sections** are comprised of 1MB blocks of memory.
- **Pages** Two different page sizes are supported:
  - **Small Pages** consist of 4KB blocks of memory. Additional access control mechanisms are extended within Small Pages to 1KB Sub-Pages.
  - **Large Pages** consist of 64KB blocks of memory. Additional access control mechanisms are extended within Large Pages to 16KB SubPages. Large Pages are supported to allow mapping of a large region of memory while using only a single entry in the TLB.

The MMU also supports the concept of domains - areas of memory that can be defined to possess individual access rights. The Domain Access Control Register is used to specify access rights for up to 16 separate domains.

The TLB caches 64 translated entries. During most memory accesses, the TLB provides the translation information to the access control logic.

If the TLB contains a translated entry for the virtual address, the access control logic determines whether access is permitted. If access is permitted, the MMU outputs the appropriate physical address corresponding to the virtual address. If access is not permitted, the MMU signals the CPU to abort.

If the TLB misses (it does not contain a translated entry for the virtual address), the translation table walk hardware is invoked to retrieve the translation information from a translation table in physical memory. Once retrieved, the translation information is placed into the TLB, possibly overwriting an existing value. The entry to be overwritten is chosen by cycling sequentially through the TLB locations.

When the MMU is turned off (as happens on reset), the virtual address is output directly onto the physical address bus.

7.2 MMU Program-accessible Registers

The ARM processor provides several 32-bit registers which determine the operation of the MMU. The format for these registers and a brief description is shown in *Figure 7-1:MMU register summary* on page 7-3. Each register will be discussed in more detail within the section that describes its use.

Data is written to and read from the MMUs registers using the ARM CPU's MRC and MCR coprocessor instructions.

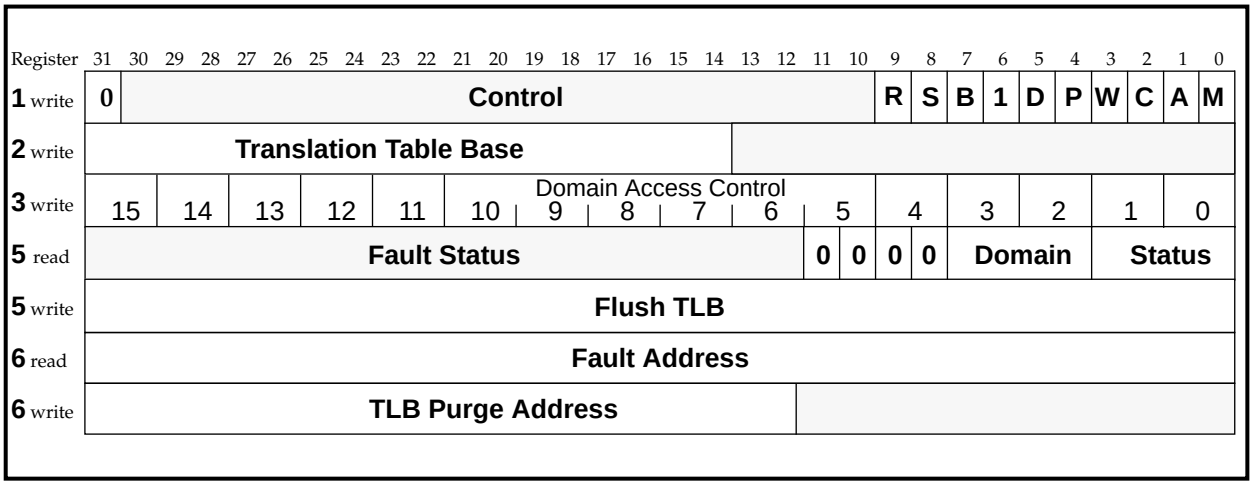

Figure 7-1: MMU register summary

Translation table base register

The Translation Table Base Register holds the physical address of the base of the translation table maintained in main memory. Note that this base must reside on a 16KB boundary.

Domain access control register

The Domain Access Control Register consists of sixteen 2-bit fields, each of which defines the access permissions for one of the sixteen Domains (D15-D0).

Note: *The registers not shown are reserved and should not be used.*

Fault status register

The Fault Status Register indicates the domain and type of access being attempted when an abort occurred. Bits 7:4 specify which of the sixteen domains (D15-D0) was being accessed when a fault occurred. Bits 3:1 indicate the type of access being attempted. The encoding of these bits is different for internal and external faults (as indicated by bit 0 in the register) and is shown in *Table 7-4:Priority encoding of fault status* on page 7-13. A write to this register flushes the TLB.

Fault address register

The Fault Address Register holds the virtual address of the access which was attempted when a fault occurred. A write to this register causes the data written to be treated as an address and, if it is found in the TLB, the entry is marked as invalid. (This operation is known as a TLB purge). The Fault Status Register and Fault Address Register are only updated for data faults, not for prefetch faults.

7.3 Address Translation

The MMU translates virtual addresses generated by the CPU into physical addresses to access external memory, and also derives and checks the access permission. Translation information, which consists of both the address translation data and the access permission data, resides in a translation table located in physical memory. The MMU provides the logic needed to traverse this translation table, obtain the translated address, and check the access permission.

There are three routes by which the address translation (and hence permission check) takes place. The route taken depends on whether the address in question has been marked as a section-mapped access or a page-mapped access; and there are two sizes of page-mapped access (large pages and small pages). However, the translation process always starts out in the same way, as described below, with a Level One fetch. A section-mapped access only requires a Level One fetch, but a page-mapped access also requires a Level Two fetch.

7.4 Translation Process

7.4.1 Translation table base

The translation process is initiated when the on-chip TLB does not contain an entry for the requested virtual address. The Translation Table Base (TTB) Register points to the base of a table in physical memory which contains Section and/or Page descriptors. The 14 low-order bits of the TTB Register are set to zero as illustrated in *Figure 7-2: Translation table base register*, the table must reside on a 16KB boundary.

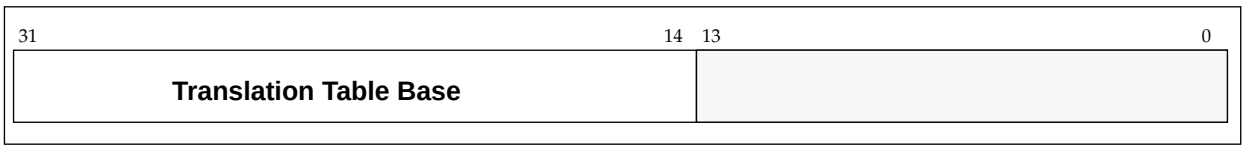

Figure 7-2: Translation table base register

7.4.2 Level one fetch

Bits 31:14 of the Translation Table Base register are concatenated with bits 31:20 of the virtual address to produce a 30-bit address as illustrated in *Figure 7-3:Accessing the translation table first level descriptors* on page 7-5. This address selects a four-byte translation table entry which is a First Level Descriptor for either a Section or a Page (bit1 of the descriptor returned specifies whether it is for a Section or Page).

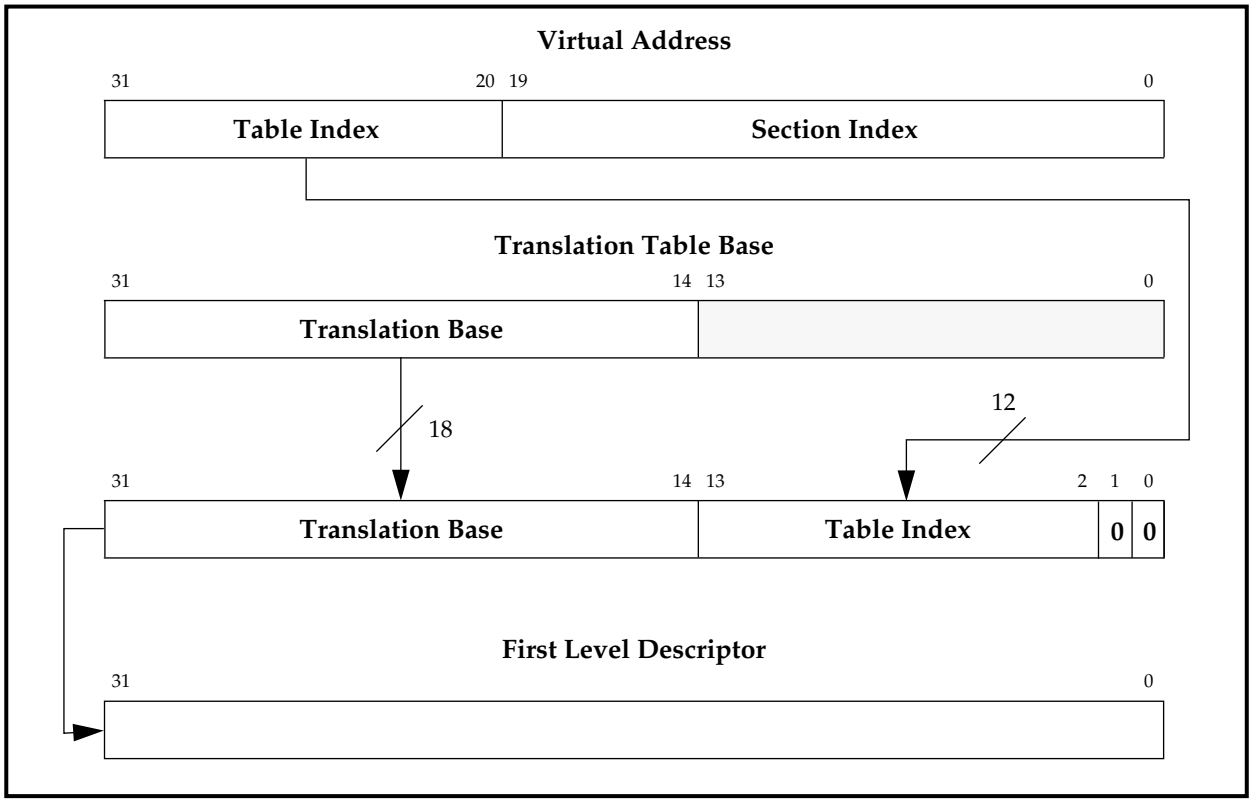

Figure 7-3: Accessing the translation table first level descriptors

7.4.3 Level one descriptor

The Level One Descriptor returned is either a Page Table Descriptor or a Section Descriptor, and its format varies accordingly. The following figure illustrates the format of Level One Descriptors.

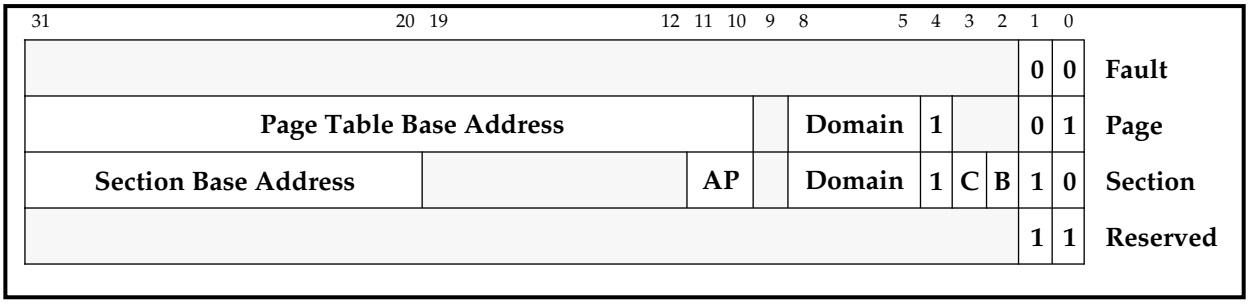

Figure 7-4: Level one descriptors

The two least significant bits indicate the descriptor type and validity, and are interpreted as in *Table 7-1:Interpreting level one descriptor bits [1:0]* on page 7-6.

Value	Meaning	Notes
0 0	Invalid	Generates a Section Translation Fault
0 1	Page	Indicates that this is a Page Descriptor
1 0	Section	Indicates that this is a Section Descriptor
1 1	Reserved	Reserved for future use

Table 7-1: Interpreting level one descriptor bits [1:0]

7.4.4 Page table descriptor

Bits 3:2 are always written as 0.

Bit 4 should be written to 1 for backward compatibility.

Bits 8:5 specify one of the sixteen possible domains (held in the Domain Access Control Register) that contain the primary access controls.

Bits 31:10 form the base for referencing the Page Table Entry. (The page table index for the entry is derived from the virtual address as illustrated in *Figure 7-7:Small page translation* on page 7-10).

If a Page Table Descriptor is returned from the Level One fetch, a Level Two fetch is initiated, as described below.

7.4.5 Section descriptor

Bits 3:2 (C, & B) control the cache- and write-buffer-related functions as follows:

- **C - Cacheable** data at this address will be placed in the cache (if the cache is enabled).
- **B - Bufferable** data at this address will be written through the write buffer (if enabled).

Bit 4 should be written to 1 for backward compatibility.

Bits 8:5 specify one of the sixteen possible domains (held in the Domain Access Control Register) that contain the primary access controls.

Bits 11:10 (AP) specify the access permissions for this section (see *Table 7-2:Interpreting access permission (AP) bits* on page 7-7). The interpretation depends upon the setting of the S and R bits (control register bits 8 and 9). Note that the Domain Access Control specifies the primary access control; the AP bits only have an effect in client mode. Refer to section on access permissions.

Bits 19:12 are always written as 0.

Bits 31:20 form the corresponding bits of the physical address for the 1MB section.

AP	S	R	Supervisor permissions	User permissions	Notes
00	0	0	No Access	No Access	Any access generates a permission fault
00	1	0	Read Only	No Access	Supervisor read only permitted
00	0	1	Read Only	Read Only	Any write generates a permission fault
00	1	1	Reserved		
01	x	x	Read/Write	No Access	Access allowed only in Supervisor mode
10	x	x	Read/Write	Read Only	Writes in User mode cause permission fault
11	x	x	Read/Write	Read/Write	All access types permitted in both modes.
xx	1	1	Reserved		

Table 7-2: Interpreting access permission (AP) bits

7.5 Translating Section References

Figure 7-6: Section translation illustrates the complete Section translation sequence. Note that the access permissions contained in the Level One Descriptor must be checked before the physical address is generated. The sequence for checking access permissions is described below.

7.5.1 Level two descriptor

If the Level One fetch returns a Page Table Descriptor, this provides the base address of the page table to be used. The page table is then accessed as described in *Figure 7-7: Small page translation*, and a Page Table Entry, or Level Two Descriptor, is returned. This in turn may define either a Small Page or a Large Page access. *Figure 7-5:Page table entry (level two descriptor)* on page 7-8 shows the format of Level Two Descriptors.

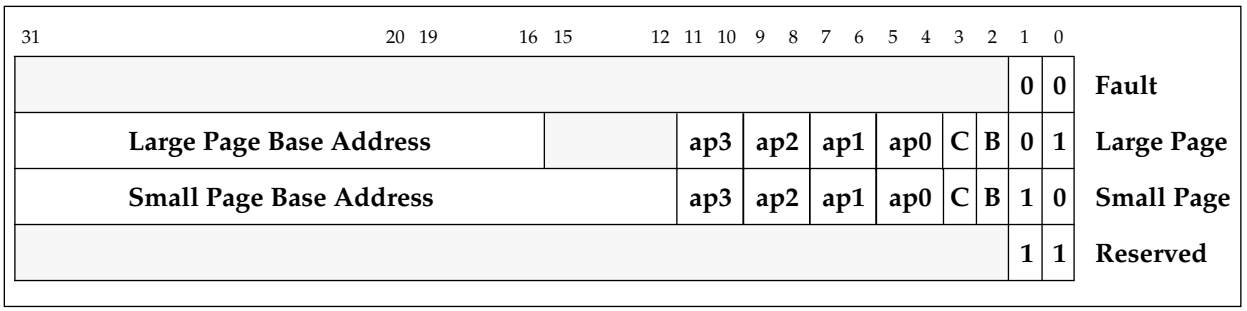

Figure 7-5: Page table entry (level two descriptor)

The two least significant bits indicate the page size and validity, and are interpreted as follows:

Value	Meaning	Notes
0 0	Invalid	Generates a Page Translation Fault
0 1	Large Page	Indicates that this is a 64KB Page
1 0	Small Page	Indicates that this is a 4KB Page
1 1	Reserved	Reserved for future use

Table 7-3: Interpreting page table entry bits 1:0

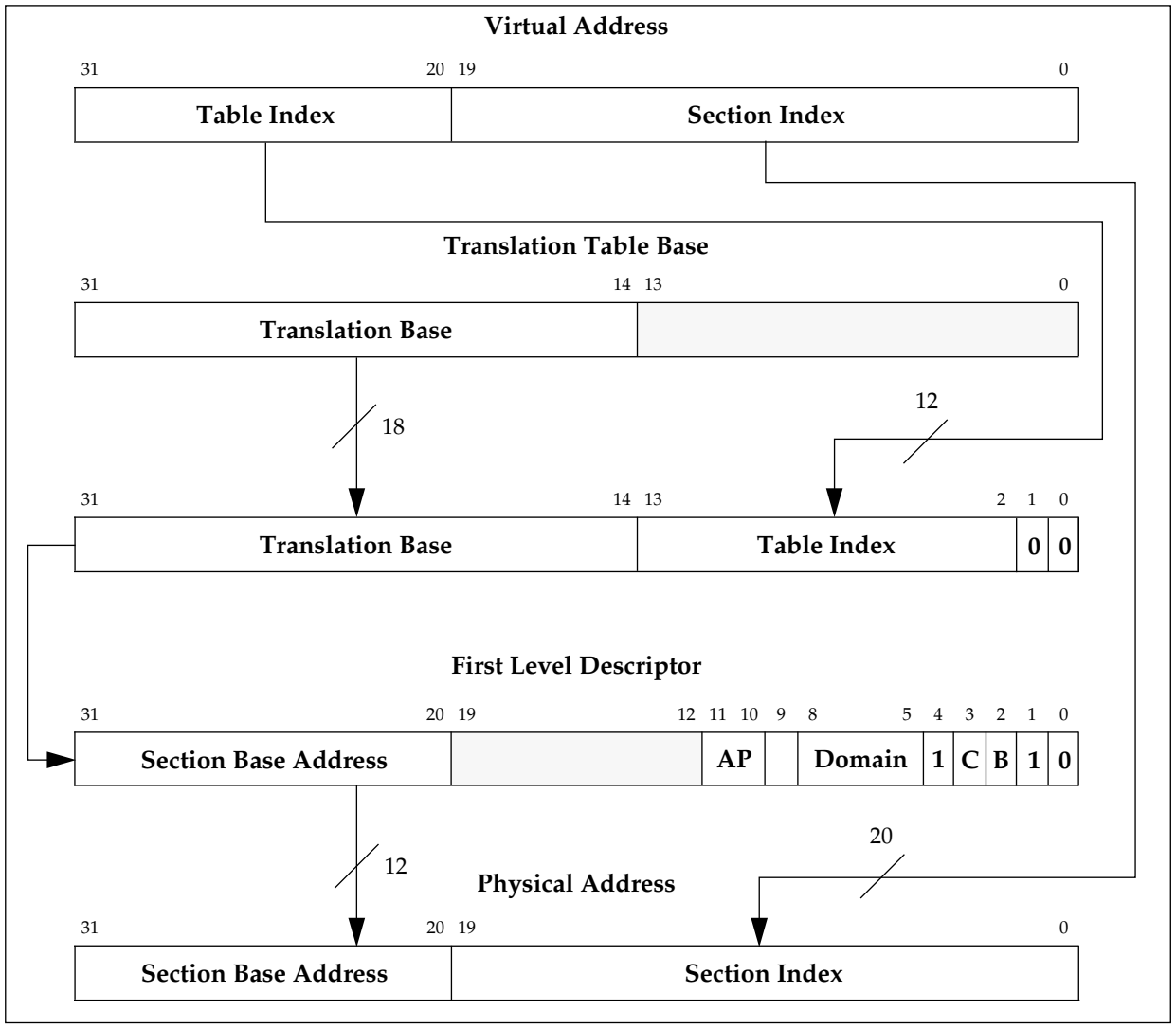

Figure 7-6: Section translation

Bit 2 (B - Bufferable) indicates that data at this address will be written through the write buffer (if the write buffer is enabled).

Bit 3 (C - Cacheable) indicates that data at this address will be placed in the IDC (if the cache is enabled).

Bits 11:4 specify the access permissions (ap3 - ap0) for the four sub-pages and interpretation of these bits is described earlier in *Table 7-1:Interpreting level one descriptor bits [1:0]* on page 7-6.

Bits 15:12 for large pages, these bits are programmed as 0.

Bits 31:12 (small pages) or bits 31:16 (large pages) are used to form the corresponding bits of the physical address - the physical page number. (The page index is derived from the virtual address as illustrated in *Figure 7-7:Small page translation* on page 7-10 and *Figure 7-8:Large page translation* on page 7-11).

7.6 Translating Small Page References

Figure 7-7: Small page translation illustrates the complete translation sequence for a 4KB Small Page. Page translation involves one additional step beyond that of a section translation: the Level One descriptor is the Page Table descriptor, and this is used to point to the Level Two descriptor, or Page Table Entry. (Note that the access permissions are now contained in the Level Two descriptor and must be checked before the physical address is generated. The sequence for checking access permissions is described later).

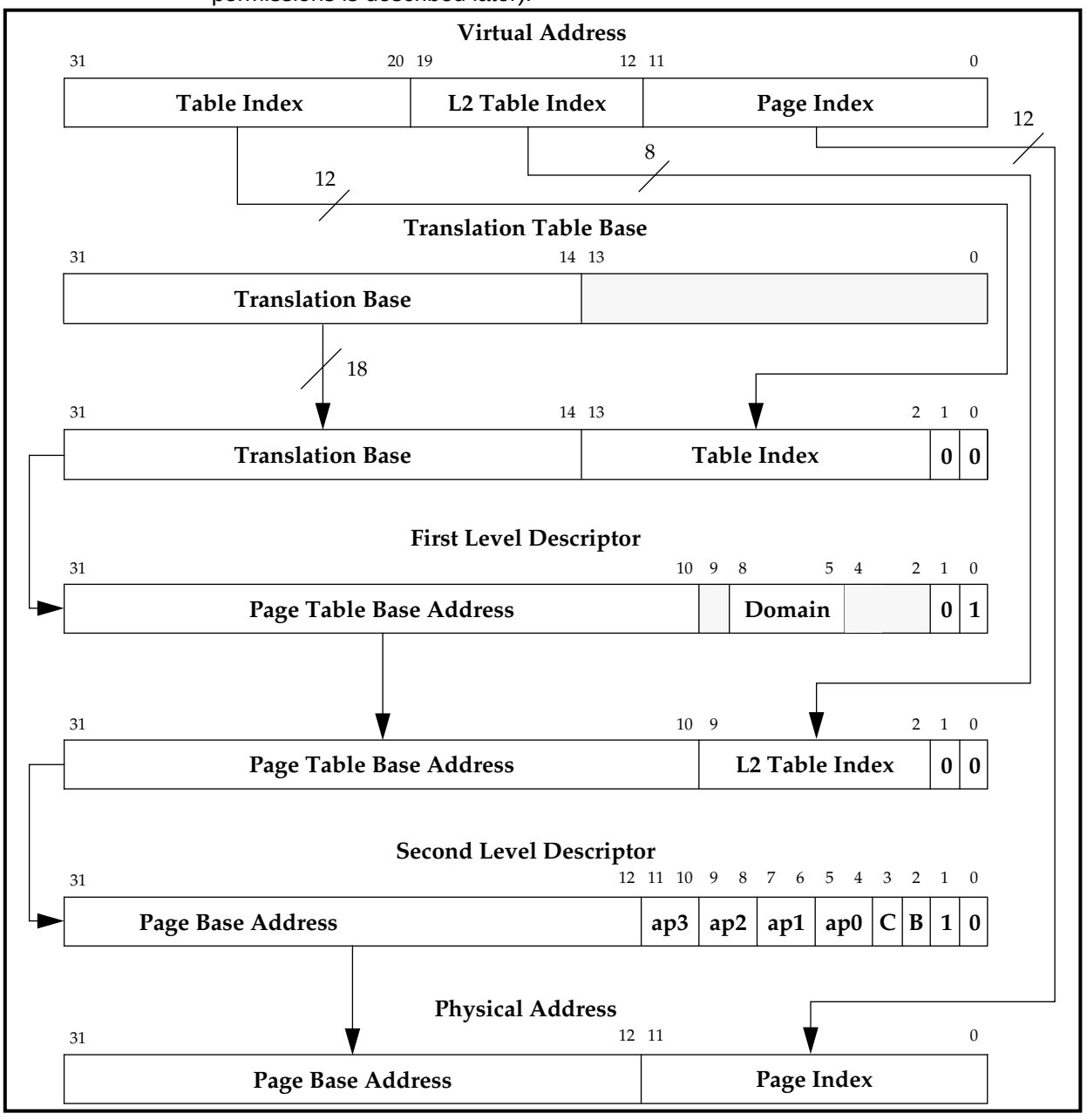

Figure 7-7: Small page translation

7.7 Translating Large Page References

Figure 7-8: Large page translation illustrates the complete translation sequence for a 64KB Large Page. Note that since the upper four bits of the Page Index and low-order four bits of the Page Table index overlap, each Page Table Entry for a Large Page must be duplicated 16 times (in consecutive memory locations) in the Page Table.

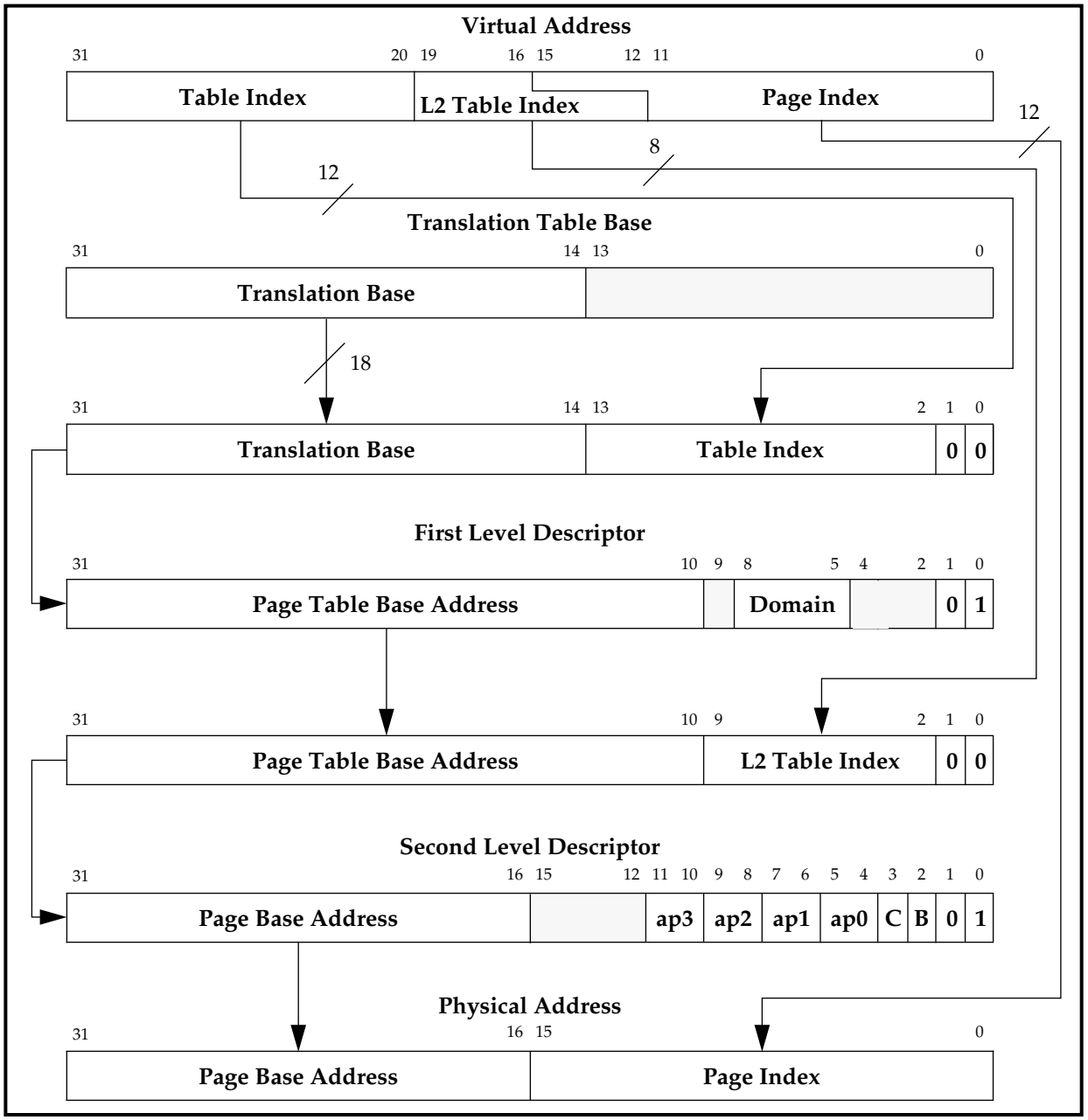

Figure 7-8: Large page translation

7.8 MMU Faults and CPU Aborts

The MMU generates four types of faults:

- Alignment Fault
- Translation Fault
- Domain Fault
- Permission Fault

The access control mechanisms of the MMU detect the conditions that produce these faults. If a fault is detected as the result of a memory access, the MMU will abort the access and signal the fault condition to the CPU. The MMU is also capable of retaining status and address information about the abort. The CPU recognizes two types of abort: data aborts and prefetch aborts, and these are treated differently by the MMU.

If the MMU detects an access violation, it will do so before the external memory access takes place, and it will therefore inhibit the access.

7.9 Fault Address & Fault Status Registers (FAR & FSR)

Aborts resulting from data accesses (data aborts) are acted upon by the CPU immediately, and the MMU places an encoded 4 bit value FS[3:0], along with the 4-bit encoded Domain number, in the Fault Status Register (FSR). In addition, the virtual processor address which caused the data abort is latched into the Fault Address Register (FAR). If an access violation simultaneously generates more than one source of abort, they are encoded in the priority given in *Table 7-4:Priority encoding of fault status* on page 7-13.

CPU instructions on the other hand are prefetched, so a prefetch abort simply flags the instruction as it enters the instruction pipeline. Only when (and if) the instruction is executed does it cause an abort; an abort is not acted upon if the instruction is not used (i.e. it is branched around). Because instruction prefetch aborts may or may not be acted upon, the MMU status information is not preserved for the resulting CPU abort; for a prefetch abort, the MMU does not update the FSR or FAR.

The sections that follow describe the various access permissions and controls supported by the MMU and detail how these are interpreted to generate faults.

In *Table 7-4:Priority encoding of fault status* on page 7-13, *x* is undefined, and may read as 0 or 1.

Notes: *Any abort masked by the priority encoding may be regenerated by fixing the primary abort and restarting the instruction. In fact this register will contain bits[8:5] of the Level 1 entry which are undefined, but would encode the domain in a valid entry.*

Priority	Source	FS[3210]	Domain [3:0]	FAR
Highest	Alignment	00x1	*x*	valid
	Translation (Section)	0101	Note 2	valid
	Translation (Page)	0111	valid	valid
	Domain (Section)	1001	valid	valid
	Domain (Page)	1011	valid	valid
	Permission (Section)	1101	valid	valid
Lowest	Permission (Page)	1111	valid	valid

Table 7-4: Priority encoding of fault status

7.10 Domain Access Control

MMU accesses are primarily controlled via domains. There are 16 domains, and each has a 2-bit field to define it. Two basic kinds of users are supported:

Clients Clients use a domain

Managers Managers control the behavior of the domain.

The domains are defined in the Domain Access Control Register. *Figure 7-9: Domain access control register format* illustrates how the 32 bits of the register are allocated to define the sixteen 2-bit domains.

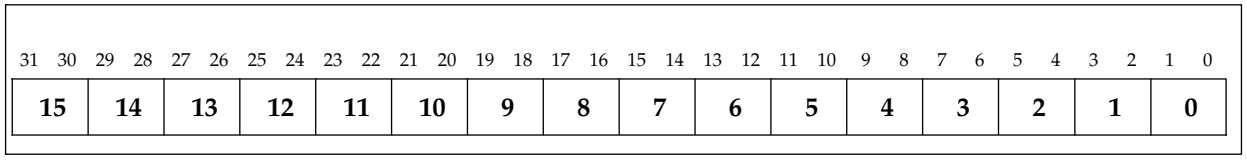

Figure 7-9: Domain access control register format

Table 7-5: Interpreting access bits in domain access control register defines how the bits within each domain are interpreted to specify the access permissions.

Value	Meaning	Notes
00	No Access	Any access will generate a Domain Fault.
01	Client	Accesses are checked against the access permission bits in the Section or Page descriptor.
10	Reserved	Reserved. Currently behaves like the no access mode.
11	Manager	Accesses are NOT checked against the access Permission bits so a Permission fault cannot be generated.

Table 7-5: Interpreting access bits in domain access control register

7.11 Fault-checking Sequence

The sequence by which the MMU checks for access faults is slightly different for Sections and Pages. The figure below illustrates the sequence for both types of accesses. The sections and figures that follow describe the conditions that generate each of the faults.

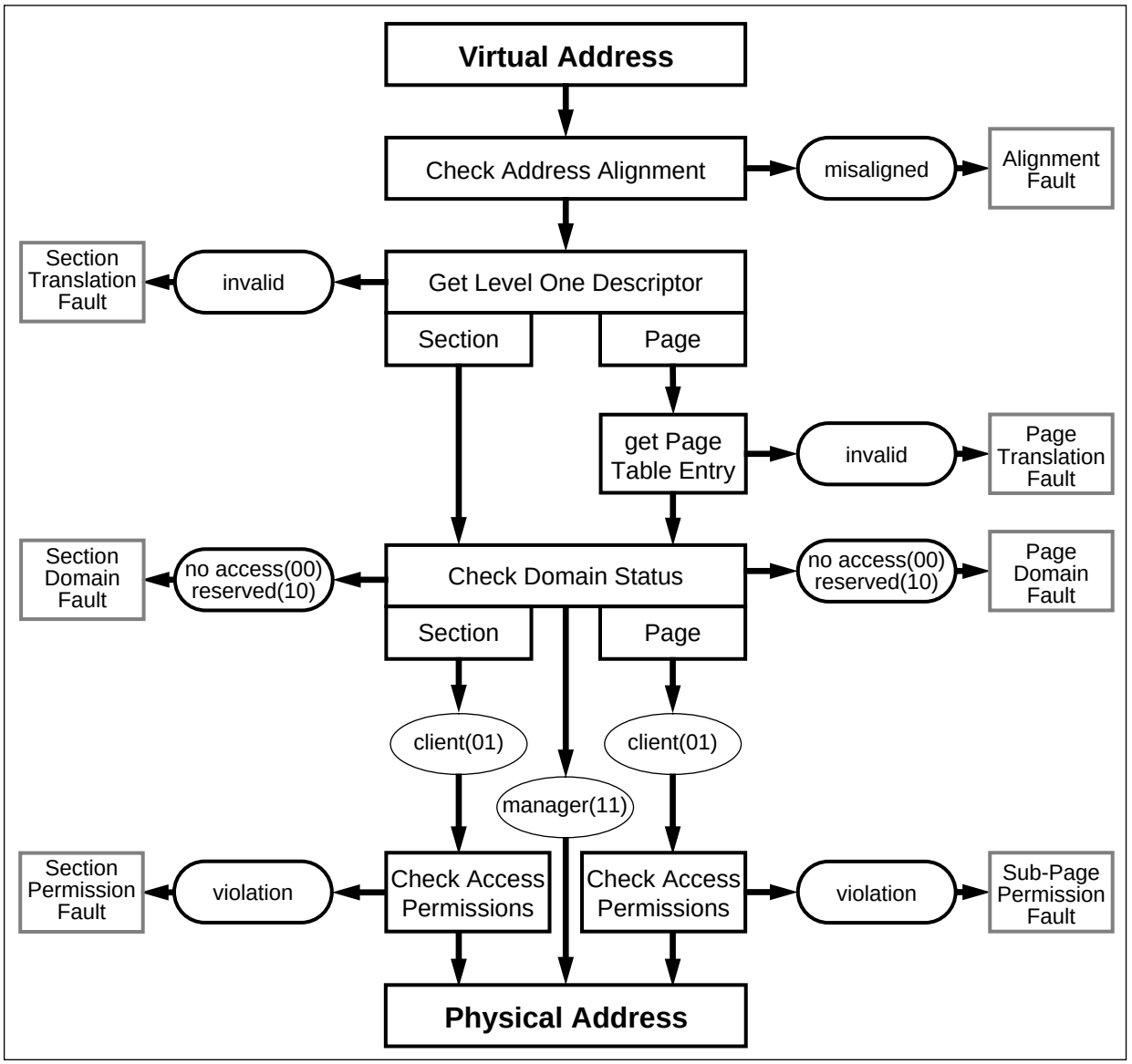

Figure 7-10: Sequence for checking faults

7.11.1 Alignment fault

If Alignment Fault is enabled (bit 1 in Control Register set), the MMU will generate an alignment fault on any data word access the address of which is not word-aligned irrespective of whether the MMU is enabled or not; in other words, if either of virtual address bits [1:0] are not 0.

Alignment fault will not be generated on any instruction fetch, nor on any byte access. Note that if the access generates an alignment fault, the access sequence will abort without reference to further permission checks.

7.11.2 Translation fault

There are two types of translation fault:

Section	is generated if the Level One descriptor is marked as invalid. This happens if bits[1:0] of the descriptor are both 0 or both 1.
Page	is generated if the Page Table Entry is marked as invalid. This happens if bits[1:0] of the entry are both 0 or both 1.

7.11.3 Domain fault

There are two types of domain fault: section and page. In both cases the Level One descriptor holds the 4-bit Domain field which selects one of the sixteen 2-bit domains in the Domain Access Control Register. The two bits of the specified domain are then checked for access permissions as detailed in *Table 7-2:Interpreting access permission (AP) bits* on page 7-7. In the case of a section, the domain is checked once the Level One descriptor is returned, and in the case of a page, the domain is checked once the Page Table Entry is returned.

If the specified access is either No Access (00) or Reserved (10) then either a Section Domain Fault or Page Domain Fault occurs.

7.11.4 Permission fault

There are two types of permission fault: section and sub-page. Permission fault is checked at the same time as Domain fault. If the 2-bit domain field returns client (01), then the permission access check is invoked as follows:

Section

If the Level One descriptor defines a section-mapped access, then the AP bits of the descriptor define whether or not the access is allowed according to *Table 7-2:Interpreting access permission (AP) bits* on page 7-7. Their interpretation is dependent upon the setting of the S bit (Control Register bit 8). If the access is not allowed, a Section Permission fault is generated.

Sub-page

If the Level One descriptor defines a page-mapped access, then the Level Two descriptor specifies four access permission fields (ap3..ap0) each corresponding to one quarter of the page. Hence for small pages, ap3 is selected by the top 1KB of the page, and ap0 is selected by the bottom 1KB of the page; for large pages, ap3 is

selected by the top 16KB of the page, and ap0 is selected by the bottom 16KB of the page. The selected AP bits are then interpreted in exactly the same way as for a section (see *Table 7-2:Interpreting access permission (AP) bits* on page 7-7), the only difference being that the fault generated is a sub-page permission fault.

7.12 External Aborts

The ARM7500FE does not support external aborts.

7.12.1 Interaction of the MMU, IDC and write buffer

The MMU, IDC and WB may be enabled/disabled independently. However there are only five valid combinations. There are no hardware interlocks on these restrictions, so invalid combinations will cause undefined results.

MMU	IDC	WB
off	off	off
on	off	off
on	on	off
on	off	on
on	on	on

Table 7-6: Valid MMU, IDC, and WB combinations

The following procedures must be observed.

To enable the MMU:

1. Program the Translation Table Base and Domain Access Control Registers
2. Program Level 1 and Level 2 page tables as required
3. Enable the MMU by setting bit 0 in the Control Register.

Note: *Care must be taken if the translated address differs from the untranslated address as the two instructions following the enabling of the MMU will have been fetched using "flat translation" and enabling the MMU may be considered as a branch with delayed execution. A similar situation occurs when the MMU is disabled. Consider the following code sequence:*

```
MOV             R1, #0x1
MCR             15,0,R1,0,0     ; Enable MMU
Fetch Flat
Fetch Flat
Fetch Translated
```

To disable the MMU

1 Disable the WB by clearing bit 3 in the Control Register.

2 Disable the IDC by clearing bit 2 in the Control Register.

3 Disable the MMU by clearing bit 0 in the Control Register.

Note: *If the MMU is enabled, then disabled and subsequently re-enabled the contents of the TLB will have been preserved. If these are now invalid, the TLB should be flushed before re-enabling the MMU.*

Disabling of all three functions may be done simultaneously.

7.13 Effect of Reset

See *Chapter 4: The ARM Processor Programmers' Model* .