9. Memory Management Unit

The memory management unit (MMU) is an optional feature of the LX4189 processor. This chapter discusses the implementation of the MMU.

A summary of the features is

- Implements the R2000/R3000 memory management system, with some minor modifications
- Implements fixed 4Kbyte memory pages, with 32-bit virtual and physical addresses and a 6-bit ASID.
- The 4GB address space is broken up into 2GB user space(kuseg), 0.5GB unmappable, cacheable kernel space (kseg0), 0.5GB unmappable, uncacheable kernel space (kseg1), and just less than 1GB of mappable kernel space (kseg2). The remainder of kseg2 (16 MB) is dedicated to specific debug devices.
- The TLB is composed of a configurable number of entries. Configuration options are: 4, 8, 16, 32, 64 entries
- Implements ENTRYHI, ENTRYLO, INDEX, RANDOM registers, based on R2000/R3000 implementation. Implements a read-only WIRED register, according to R4000 implementation.
- Signals the appropriate UTLB exception (TLBL or TLBS) or the appropriate BEV0/BEV1 exception (TLBL, TLBS, MOD)
- Implements the TLBP, TLBR, TLBWI, TLBWR instructions

9.1. References

Please note that this document is an implementation note connected with the Lexra processor series. For a more complete description of MIPS instruction set implementations of Memory Management Hardware, please see *MIPS RISC Architecture* by Kane and Heinrich. For a complementary discussion of the role of Memory Management in operating systems associated with the MIPS instruction set, please see *See MIPS Run* by Sweetman.

9.2. Memory Regions

Region Name	Virtual Address	Physical Address	Cachability	Permission
kuseg	0x0000_0000-0x7fff_ffff	mapped via TLB	set via TLB	user
kseg0	0x8000_0000-0x9fff_ffff	0x0000_0000-0x1fff_ffff	cacheable	kernel
kseg1	0xa000_0000-0xbfff_ffff	0x0000_0000-0x1fff_ffff	uncacheable	kernel
kseg2	0xc000_0000-0xfeff_ffff	mapped via TLB	set via TLB	kernel
upper-kseg2	0xff00_0000-0xffff_ffff	0xff00_0000-0xffff_ffff	uncacheable	kernel

9.3. Registers

9.3.1. TLB Entry

A TLB Entry is a 64-bit quantity which stores a Virtual to Physical Memory Mapping. The fields

and bit locations of a TLB Entry are identical to those in the EntryHi and EntryLo fields. The TLB Entries are read / written via TLB Read (TLBR) / TLB Write (Indexed or Random) (TLBWI or TLBWR) instructions. The TLB Entries can be examined for a match to a specific VPN, ASID pair (also examining the Global bit) using the TLBP instruction. The TLB Entries are also matched to a TLB Request, either from an Instruction or Data Memory request.

9.3.2. CP0 Register 10: EntryHi

The EntryHi register is read/written via the MFC0 / MTC0 instructions (Move From Coprocessor 0 / Move To Coprocessor 0). It is accessed as Coprocessor 0 register 10.

Entryhi is composed of two fields; the Virtual Page Number (VPN) and Application-Specific ID (ASID). The Memory Management Units compares the VPN and ASID fields of EntryHi, together with the global bit of the EntryLo register, to each entry in the TLB when the CPU executes a TLB Probe (TLBP) instruction, and during any mapped Instruction or Data Probe Request. The comparison mechanism is described in the TLBP instruction.

When the processor executes a TLB Read (TLBR) instruction, the TLB entry specified by the INDEX register is loaded into EntryHi and EntryLo.

The VPN and ASID fields of EntryHi are written into the TLB when the CPU executes the TLB Write-Index (TLBWI) or TLB Write Random (TLBWR) instructions.

The format of the EntryHi register is:

31 12	11 6	5 0
VPN	ASID	000000

9.3.3. CP0 Register 2: EntryLo

The EntryLo register is read/written via the MFC0 / MTC0 instructions (Move From Coprocessor 0 / Move To Coprocessor 0). It is accessed as Coprocessor 0 register 2.

EntryLo is composed of the following fields: PFN, N(oncachable), D(irty), V(alid) and G(lobal).

The PFN, N, D, V and G fields of EntryLo are written into the TLB when the CPU executes the TLBWI or TLBWR instructions. During any mapped Instruction or Data Probe Request, and during the execution of a TLBP instruction, the MMU utilizes the G bit to determine if a VPN, ASID combination was a match. A mapped Instruction or Data Probe Request with a TLB match (based on VPN, ASID and G) will examine the D(irty) and V(alid) bits to signal an exception (this process is described in the Exception Priority / Codes section of this chapter). The N(oncacheable) bit is used to determine if the reference should be made to the Cache or directly to the memory.

When the processor executes a TLB Read (TLBR) instruction, the TLB entry specified by the INDEX register is loaded into EntryHi and EntryLo.

The format of the EntryHi register is:

31 12	11	10	9	8	7 0
PFN	N	D	V	G	00000000

9.3.4. CP0 Register 0: Index

The Index register is read/written via the MFC0 / MTC0 instructions (Move From Coprocessor 0 / Move To Coprocessor 0). It is accessed as Coprocessor 0 register 0.

The Index Register contains a 6-bit address into the TLB (INDEX), and a one bit probe failure field (P).

When the processor executes a TLB Write Index (TLBWI) or TLB Read (TLBR) instruction, the TLB entry which is updated / read is pointed at by the INDEX field of the INDEX register.

When the processor executes a TLB Probe (TLBP) instruction: if no TLB match is found, the P bit is set to 1 (Probe Failure) and the results of the INDEX field are undefined. if a single TLB match is found, the P bit is set to 1, and the INDEX field is the INDEX of the TLB entry which matched. The result of the INDEX register following a TLBP instruction with multiple matches is undefined.

The format of the Index register is:

31	30 14	13 8	7 0
P	00000000000000000	INDEX	00000000

9.3.5. CP0 Register 2: Random

The Random register is a read-only register accessed via the MFC0 instruction (Move From Coprocessor 0). It is accessed as Coprocessor 0 register 1.

The Random Register contains a 6-bit address into the TLB (RANDOM), which is decremented on every clock cycle during operation of the processor. It decrements from the largest TLB entry available on the processor (ie, 32 entries) to the value of the WIRED register. This ensures that the lowest WIRED entries (0 to WIRED-1) are safe TLB entries which cannot be randomly overwritten.

When the processor executes a TLB Write Random (TLBWR) instruction, the TLB entry which is updated is pointed at by the RANDOM field of the RANDOM register.

The format of the RANDOM register is:

31 14	13 8	7 0
00000000000000000	RANDOM	00000000

9.3.6. CP0 Register 6: Wired

The Wired register is a read-only register accessed via the MFC0 instruction (Move From Coprocessor 0). It is accessed as Coprocessor 0 register 6.

The WIRED Register contains a 6-bit address of the first non-safe value which may be overwritten by the TLB Write Random (TLBWR) instruction. The Random register decrements from the largest TLB entry available on the processor (ie, 32 entries) to the value of the WIRED register. This ensures that the lowest WIRED entries (0 to WIRED-1) are safe TLB entries which cannot be randomly overwritten.

In the Lexra implementation, the WIRED entry is set by the number of TLB entries which have been selected. The values are:

TLB_ENTRIES	Number Safe Entries	Wired (binary)
64	8	6'b001000
32	4	6'b000100
16	4	6'b000100
8	2	6'b000010
4	2	6'b000010

The format of the WIRED register is:

31 6	5 0
00000000000000000000000000	WIRED

9.4. Instructions

9.4.1. Summary of Instructions

Mnemonic	Operation	Description
TLBP	INDEX <- pointer to TLB entry which matches EntryHi[VPN,ASID], EntryLo[GLOBAL]	Determine if there is an Entry in the TLB which matches EntryHi, EntryLo. Set the Probe Failure bit, and INDEX result.
TLBR	EntryHi, EntryLo <- TLB[INDEX]	Load EntryHi and EntryLo with the TLB entry pointed at by the INDEX register.
TLBWI	TLB[INDEX] <- EntryHi, EntryLo	Update the TLB entry pointed at by the INDEX register with the values in EntryHi, EntryLo
TLBWR	TLB[RANDOM] <- EntryHi, EntryLo	Update the TLB entry pointed at by the RANDOM register with the values in EntryHi, EntryLo

9.4.2. TLB Probe (TLBP) Instruction

The TLB Probe (TLBP) instruction examines each TLB entry to determine if it matches the values contained in EntryHi and EntryLo. The determination of a match for entry i uses the equation:

```
Match[i] = (EntryHi[VPN] == TLBHi[i][VPN]) && (TLBLo[i][GLOBAL] ||
(EntryHi[ASID] == TLBHi[i][ASID]))
```

where TLBHi and TLBLo are the appropriate fields of the TLB entry.

If no TLB Match is found, the INDEX register P bit is set to 1, and the INDEX field is undefined.

If a single TLB Match is found, the INDEX register P bit is set to 0, and the INDEX field takes the value of the matching register.

If multiple TLB Matches are found, the INDEX register P bit and INDEX value are both undefined.

9.4.3. TLB Read (TLBR) Instruction

The TLB Read instruction updates the EntryHi and EntryLo registers with the contents of the TLB Entry specified by the INDEX register.

If the value of INDEX register is greater than the number of TLB Entries implemented, the TLB

Read instruction returns a 32-bit zero result to both the EntryHi and EntryLo registers.

9.4.4. TLB Write Index (TLBWI) Instruction

The TLB Write Index instruction updates the entry of the TLB specified by the INDEX field of the INDEX register, with the values specified in EntryHi and EntryLo. If the value of INDEX is greater than the number of registers implemented, no entry is updated.

A TLB Write Index operation invalidates any previously stored copies of the VPN to PFN mapping stored in the instruction fetch sequencer.

9.4.5. TLB Write Random (TLBWR) Instruction

The TLB Write Random instruction updates the entry of the TLB specified by the RANDOM field of the RANDOM register, with the values specified in EntryHi and EntryLo.

A TLB Write Random operation does not invalidate any previously stored copies of the VPN to PFN mapping stored in the instruction fetch sequencer. Therefore, it should not be used to "overwrite" a currently valid mapping. It should be used only to store a new mapping when the existing mappings remain valid.

9.5. Translation Requests

Translation requests can be signalled in three ways: through an instruction fetch, through a data load operation, and through a data store operation. The translation handling mechanism is the same regardless of the nature of the exception, and is described here.

Any memory reference to kseg0, kseg1, or upper kseg2 (see the Memory Region section of this document) it is an unmapped access. An unmapped access does not generate a request to the TLB; instead, the direct map to the PFN is used (note: this specifies both the physical location and cacheability of the request. All of the areas are assumed dirty (writeable))

If the reference is to a kuseg or lower-kseg2 address, a TLB translation request is generated. A TLB match is defined as a single entry (i) matching according to the equation shown below.

```
Match[i] = (VPN(address) == TLBHi[i][VPN]) && (TLBLo[i][GLOBAL] ||
(EntryHi[ASID] == TLBHi[i][ASID]))
```

The list below shows the sequence of decisions, and the results. Note that the VPN comes from the address being tested, while the the ASID is stored in EntryHi.

- (If the processor is in user mode, and the reference is to kseg2, an Address exception is signaled)
- If there is no TLB match, and the reference is to kuseg, generate a UTLB TLBL or TLBS (aka Miss) exception
- If there is no TLB match, and the reference is to lower-kseg2, generate a BEV TLBL or TLBS (aka Miss) exception
- If there is a TLB match and the matching entry is not valid, generate a BEV TLBL or TLBS (aka Miss) exception
- If the request is for a Data Store operation, and the Dirty bit is not set,

generate a BEV Mod exception.

- Generate a cacheable request if the N field is 0, and a noncacheable request if the N field is 1.

9.6. Pipelining / Stalls

9.7. Exception Codes

See the Translation request section for a description of the translation requests / TLB results which cause the exceptions listed in this table. The User TLB (UTLB) exceptions go to a unique exception vector which can utilize a very low level of functionality for user applications only, while the more complete BEV (Boot Exception Vector) versions include handling for privledged (OS) tasks, invalid entries and access violations (modify exceptions). For a more complete discussion of the role of these exceptions, please see *See MIPS Run* by Dominic Sweetman.

Exception Type	BEV0 exception address	BEV1 exception address	Cause ExcCode
UTLB TLBL	0x8000_0000	0xbfc0_0100	2
UTLB TLBS	0x8000_0000	0xbfc0_0100	3
BEV TLBL	0x8000_0080	0xbfc0_0180	2
BEV TLBS	0x8000_0080	0xbfc0_0180	3
BEV MOD	0x8000_0080	0xbfc0_0180	1