Table 4-3 Address Translation and Cache Coherency Attributes for the kseg0 and kseg1 Segments

Segment Name	Virtual Address Range	Generates Physical Address	Cache Attribute
kseg1	16#BFFF FFFF through 16#A000 0000	16#1FFF FFFF through 16#0000 0000	Uncached
kseg0	16#9FFF FFFF through 16#8000 0000	16#1FFF FFFF through 16#0000 0000	From K0 field of *Config* Register

4.6 Address Translation for the kuseg Segment when $Status_{ERL}$ = 1

To provide support for the cache error handler, the kuseg Segment becomes an unmapped, uncached Segment, similar to the kseg1 Segment, if the ERL bit is set in the *Status* register. This allows the cache error exception code to operate uncached using GPR R0 as a base register to save other GPRs before use.

4.7 Special Behavior for the kseg3 Segment when $Debug_{DM}$ = 1

If EJTAG is implemented on the processor, the EJTAG block must treat the virtual address range 16#FF20 0000 through 16#FF3F FFFF, inclusive, as a special memory-mapped region in Debug Mode. A MIPS32 compliant implementation that also implements EJTAG must:

- explicitly range check the address range as given and not assume that the entire region between 16#FF20 0000 and 16#FFFF FFFF is included in the special memory-mapped region.
- not enable the special EJTAG mapping for this region in any mode other than in EJTAG Debug mode.

Even in Debug mode, normal memory rules may apply in some cases. Refer to the EJTAG specification for details on this mapping.

4.8 TLB-Based Virtual Address Translation

This section describes the TLB-based virtual address translation mechanism. Note that sufficient TLB entries must be implemented to avoid a TLB exception loop on load and store instructions.

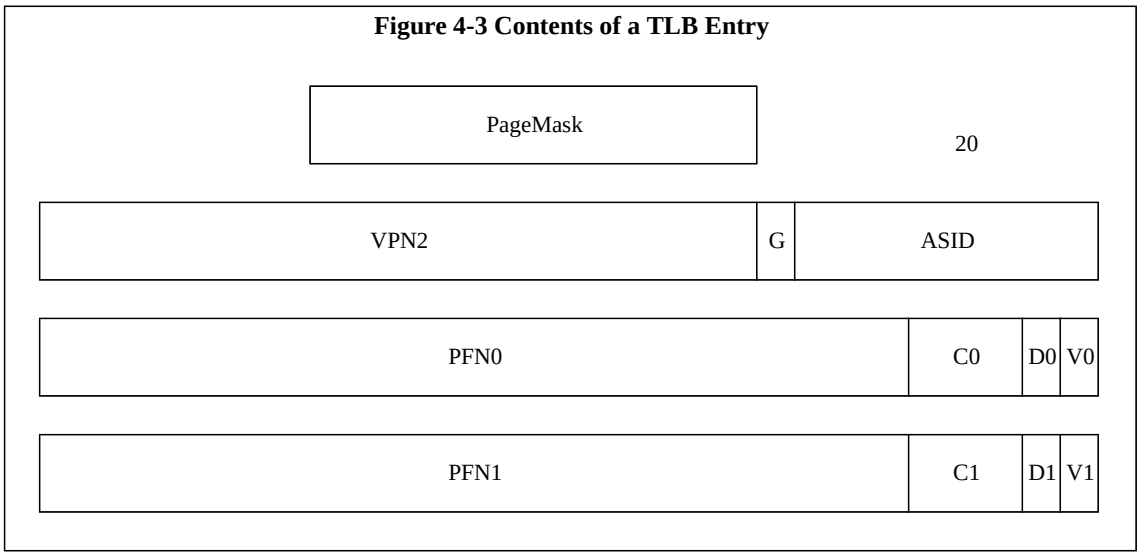
4.8.1 Address Space Identifiers (ASID)

The TLB-based translation mechanism supports Address Space Identifiers to uniquely identify the same virtual address across different processes. The operating system assigns ASIDs to each process and the TLB keeps track of the ASID when doing address translation. In certain circumstances, the operating system may wish to associate the same virtual address with all processes. To address this need, the TLB includes a global (G) bit which over-rides the ASID comparison during translation.

4.8.2 TLB Organization

The TLB is a fully-associative structure which is used to translate virtual addresses. Each entry contains two logical components: a comparison section and a physical translation section. The comparison section includes the virtual page number (VPN2) (actually, the virtual page number/2 since each entry maps two physical pages) of the entry, the ASID, the G(lobal) bit and a recommended mask field which provides the ability to map different page sizes with a single entry. The physical translation section contains a pair of entries, each of which contains the physical page frame number (PFN), a valid (V) bit, a dirty (D) bit, and a cache coherency field (C), whose valid encodings are given in Table 6-6 on page 44. There are two entries in the translation section for each TLB entry because each TLB entry maps an aligned pair of virtual pages and the pair of physical translation entries corresponds to the even and odd pages of the pair. Figure 4-3 shows the logical arrangement of a TLB entry.

Figure 4-3 Contents of a TLB Entry



The fields of the TLB entry correspond exactly to the fields in the CP0 *PageMask*, *EntryHi*, *EntryLo0* and *EntryLo1* registers. The even page entries in the TLB (e.g., PFN0) come from *EntryLo0*. Similarly, odd page entries come from *EntryLo1*.

4.8.3 Address Translation

When an address translation is requested, the virtual page number and the current process ASID are presented to the TLB. All entries are checked simultaneously for a match, which occurs when all of the following conditions are true:

- The current process ASID (as obtained from the *EntryHi* register) matches the ASID field in the TLB entry, or the G bit is set in the TLB entry.
- The appropriate bits of the virtual page number match the corresponding bits of the VPN2 field stored within the TLB entry. The “appropriate” number of bits is determined by the PageMask field in each entry by ignoring each bit in the virtual page number and the TLB VPN2 field corresponding to those bits that are set in the PageMask field. This allows each entry of the TLB to support a different page size, as determined by the PageMask register at the time that the TLB entry was written. If the recommended PageMask register is not implemented, the TLB operation is as if the PageMask register was written with a zero, resulting in a minimum 4096-byte page size.

If a TLB entry matches the address and ASID presented, the corresponding PFN, C, V, and D bits are read from the translation section of the TLB entry. Which of the two PFN entries is read is a function of the virtual address bit immediately to the right of the section masked with the PageMask entry.

The valid and dirty bits determine the final success of the translation. If the valid bit is off, the entry is not valid and a TLB Invalid exception is raised. If the dirty bit is off and the reference was a store, a TLB Modified exception is raised.

If there is an address match with a valid entry and no dirty exception, the PFN and the cache coherency bits are appended to the offset-within-page bits of the address to form the final physical address with attributes.

The TLB lookup process can be described as follows:

```
found ← 0
for i in 0...TLBEntries-1
   if ((TLB[i]VPN2 and not (TLB[i]Mask)) = (va31..13 and not (TLB[i]Mask))) and
      (TLB[i]G or (TLB[i]ASID = EntryHiASID)) then
      # EvenOddBit selects between even and odd halves of the TLB as a function of
      # the page size in the matching TLB entry
      case TLB[i]Mask
         2#0000000000000000: EvenOddBit ← 12
         2#0000000000000011: EvenOddBit ← 14
         2#0000000000001111: EvenOddBit ← 16
         2#0000000000111111: EvenOddBit ← 18
         2#0000000011111111: EvenOddBit ← 20
         2#0000001111111111: EvenOddBit ← 22
         2#0000111111111111: EvenOddBit ← 24
         2#0011111111111111: EvenOddBit ← 26
         2#1111111111111111: EvenOddBit ← 28
         otherwise:   UNDEFINED
      endcase
      if vaEvenOddBit = 0 then
         pfn ← TLB[i]PFN0
         v ← TLB[i]V0
         c ← TLB[i]C0
         d ← TLB[i]D0
      else
         pfn ← TLB[i]PFN1
         v ← TLB[i]V1
         c ← TLB[i]C1
         d ← TLB[i]D1
      endif
      if v = 0 then
         SignalException(TLBInvalid, reftype)
      endif
      if (d = 0) and (reftype = store) then
         SignalException(TLBModified)
      endif
      # pfnPABITS-1-12..0 corresponds to paPABITS-1..12
      pa ← pfnPABITS-1-12..EvenOddBit-12 || vaEvenOddBit-1..0
      found ← 1
      break
   endif
endfor
if found = 0 then
   SignalException(TLBMiss, reftype)
endif
```

Table 4-4 demonstrates how the physical address is generated as a function of the page size of the TLB entry that matches the virtual address. The “Even/Odd Select” column of Table 4-4 indicates which virtual address bit is used to select between the even (EntryLo0) or odd (EntryLo1) entry in the matching TLB entry. The “PA generated from” column specifies how the physical address is generated from the selected PFN and the offset-in-page bits in the virtual

address. In this column, PFN is the physical page number as loaded into the TLB from the EntryLo0 or EntryLo1 registers, and has the bit range $PFN_{PABITS\text{-}1\text{-}12..0}$, corresponding to $PA_{PABITS\text{-}1..12}$.

Table 4-4 Physical Address Generation

Page Size	Even/Odd Select	PA generated from
4K Bytes	VA_{12}	$PFN_{PABITS\text{-}1\text{-}12..0} \| VA_{11..0}$
16K Bytes	VA_{14}	$PFN_{PABITS\text{-}1\text{-}12..2} \| VA_{13..0}$
64K Bytes	VA_{16}	$PFN_{PABITS\text{-}1\text{-}12..4} \| VA_{15..0}$
256K Bytes	VA_{18}	$PFN_{PABITS\text{-}1\text{-}12..6} \| VA_{17..0}$
1M Bytes	VA_{20}	$PFN_{PABITS\text{-}1\text{-}12..8} \| VA_{19..0}$
4M Bytes	VA_{22}	$PFN_{PABITS\text{-}1\text{-}12..10} \| VA_{21..0}$
16M Bytes	VA_{24}	$PFN_{PABITS\text{-}1\text{-}12..12} \| VA_{23..0}$
64MBytes	VA_{26}	$PFN_{PABITS\text{-}1\text{-}12..14} \| VA_{25..0}$
256MBytes	VA_{28}	$PFN_{PABITS\text{-}1\text{-}12..16} \| VA_{27..0}$