Programmed Segmentation Control and Enhanced Virtual Addressing in MIPS32® Architecture Release 3

This application note describes Programmed Segmentation Control and Enhanced Virtual Adressing in MIPS cores that implement MIPS32® Release 3 of the MIPS® Architecture.

Document Number: MD00952
Revision 01.04
July 26, 2013

Table of Contents

1 Introduction 3

1.1 What was before? 3

1.2 Programmed Segmentation Control 4

2 Programmed Segmentation Control Overview 5

2.1 Virtual Address Segments 5

2.2 Segment Characteristics 5

3 Configuration Registers for Programmed Segmentation Control 8

4 Programmed Segmentation Control Configuration for MIPS Legacy Memory Map 10

5 Programmed Segmentation Control Configuration for Extended Virtual Kernel and User Address Spaces (EVA) 14

6 Programming for Legacy Mode and EVA mode 15

6.1 CP0 Config5 (register 16, Select6) 16

6.2 CP0 Ebase (register 15, Select 1) 17

6.3 Boot Exception Vector Overlay 18

6.4 GCR Core Local Reset Exception Base Register 18

6.5 Core Local Reset Exception Extended Base Register 18

6.6 Overlay Example setting the Physical address to 0xBFC0 0000 22

6.7 Moving I/O Registers and GCR base 24

7 New Load and Store Instructions to support EVA 24

8 Linux Example Using 2GB RAM on a Malta Platform 25

8.1 EVA Memory Map for Linux on Malta FPGA – 2GB RAM Kernel Mode 26

8.2 EVA Memory Map For Linux on Malta FPGA - 2GB RAM User Mode 27

8.3 EVA Memory Map For Linux on Malta FPGA - 2GB RAM EU - Status.ERL = 1 (reset, NMI or cache error) 28

8.4 Adding the memory to Linux 29

8.5 Kernel Files that changed for Segmentation and EVA 29

8.6 Additional Linux Information 30

1 Introduction

This application note describes Programmed Segmentation Control and Enhanced Virtual Addressing (EVA) in the MIPS32® Release 3 Architecture, two additions to the MIPS architecture that increase the flexibility of its segment-based memory scheme.

In Release 3, the memory architecture defines the partitioning of memory space into fixed-size segments, whose access properties and cacheability attributes are programmable. The "mapped" and "unmapped" segment-access properties can now be modified so as to extend system memory space (kseg0/kseg1) to include unmapped access to user space (xkseg), a feature called EVA. Instructions that facilitate system read/write access to xkseg have also been added to the R3 ISA.

1.1 What was before?

Legacy Addressability

User Mode Addressable	Address	Kernel Mode Addressable
Address Error	**4.0 GB**	KSEG3 Fixed by HW as Mapped and Cacheable through TLB and accessible only in Kernel Mode
	3.5 GB	KSEG2 Fixed by HW as Mapped and Cacheable through TLB and accessible only in Kernel Mode
	3.0 GB	KSEG1 Fixed by HW as uncacheable, directly translated to lower memory and accessible only in Kernel Mode
	2.5 GB	KSEG0 Fixed by HW as cacheable, directly translated to lower memory and accessible only in Kernel Mode
USEG Fixed by HW as Mapped and Cacheable through TLB and accessible in all Modes	**2.0 GB**	Upper half of KUSEG Fixed by HW as Mapped and Cacheable through TLB and accessible in all Modes
USEG Fixed by HW as Mapped and Cacheable through TLB and accessible in all Modes	**1.0 GB**	Lower half of KUSEG Fixed by HW as Mapped and Cacheable through TLB and accessible in all Modes

Prior to Programmed Segmentation Control, each segment of an Address Space was classified as "Mapped" or "Unmapped". A "Mapped" address is one that is translated through the TLB or other address translation unit. An "Unmapped" address is one that is not translated through the TLB and that provides a window into the lowest portion of the physical address space, starting at physical address zero, and with a size corresponding to the size of the unmapped segment.

Each segment of an Address Space is associated with one of the three processor operating modes (User, Supervisor, or Kernel). A segment associated with a particular mode is accessible if the processor is running in that mode or a more privileged mode. All modern OS implementations use just two of the modes, User and Kernel. The Kernel mode is the most privileged mode and has access to the full 4 GB of virtual and physical address space. For Kernel mode, there are several segments that divide up the virtual address space, each with a different set of characteristics (some are unmapped, some are cacheable and some are not, some are only accessible through the TLB). The user mode is the least privileged mode and has access to the lowest 2 GB of virtual when a TLB is used, or to the lower 2 GB of virtual address space direct-mapped to the physical memory range 0x4000 0000 through 0xC000 0000 when used with Fixed mapping.

The thing to take away from this is that each segment is defined with specific attributes.

1.2 Programmed Segmentation Control

Addressability with Programmed Segmentation Control

User Mode Addressable	Address	Kernel Mode Accessible
Depends on setting of Segment register 0	4.0 GB	Segment 0 controlled by CP0, SegCtl0, CFG0
Depends on setting of Segment register 0	3.5 GB	Segment 1 controlled by CP0, SegCtl0, CFG1
Depends on setting of Segment register 1	3.0 GB	Segment 2 controlled by CP0, SegCtl1, CFG2
Depends on setting of Segment register 1	2.5 GB	Segment 3 controlled by CP0, SegCtl1, CFG3
Depends on setting of Segment register 2	2.0 GB	Segment 4 controlled by CP0, SegCtl2, CFG4
Depends on setting of Segment register 2	1.0 GB	Segment 5 controlled by CP0, SegCtl2, CFG5

Programmed Segmentation Control divides the Virtual memory map into fixed-size virtual segments whose characteristics are fully programmable. Each segment can be set for mapped or unmapped, Kernel, Supervisor or User mode access, and set for any Cache Attribute. The only characteristic that is fixed is the Virtual Address range of each segment.

The rest of this document will describe the Programmed Segmentation Control memory architecture in greater detail.

2 Programmed Segmentation Control Overview

2.1 Virtual Address Segments

Programmed Segmentation Control divides the Virtual address space into 6 segments with the following virtual address ranges:

Segment Address Table

Segment Number	Size	Starting Virtual Address	Ending Virtual Address
0 (CFG0)	.5 GB	0xE000 0000	0xFFFF FFFF
1 (CFG1)	.5 GB	0xC000 0000	0xDFFF FFFF
2 (CFG2)	.5 GB	0xA000 0000	0xBFFF FFFF
3 (CFG3)	.5 GB	0x8000 0000	0x9FFF FFFF
4 (CFG4)	1 GB	0x4000 0000	0x7FFF FFFF
5 (CFG5)	1 GB	0x0000 0000	0x3FFF FFFF

Notice that the segment size is different for segments 4 and 5.

2.2 Segment Characteristics

A virtual segment can have the following characteristics:

- Access Mode (AM) – In the MIPS architecture there are three privileged modes and two address translation methods. The privileged modes are Kernel, Supervisor and User. The Kernel mode has the highest privileges and can access any address in the 4 GB range. The Supervisor Mode can access any Supervisor or User address segments. User has the lowest privileges and can only access User segments. The two address translation methods are Mapped (through a TLB) or Unmapped (directly translated to a static physical address without a TLB).
- Error Condition (EU) – Setting this characteristic changes the segment to unmapped and uncached when the CPU enters Error condition (Status.ERL = 1: reset, NMI or cache error).
- Cache Coherence (C) – If the segment is mapped, the TLB entry sets the cache attributes; if unmapped, cache attributes are set in the segment configuration.
- Physical Address (PA) - When the segment is unmapped, (either by the Access Mode or Error Condition), it will use this address to select the segment of physical memory that corresponds to the virtual segment. Physical addresses can only be set on a 512 Megabyte boundary so that means only the top 3 bits of the 32 bit physical address can be set and the remaining bits will be 0. The PA field is 7 bits wide however only the first 3 bits of the field are valid for current MIPS cores.

Here are the valid settings for the field:

PA	Physical Address
000	0x0000 0000
001	0x2000 0000
010	0x4000 0000
011	0x6000 0000
100	0x8000 0000
101	0xA000 0000
110	0xC000 0000
111	0xE000 0000

3 Configuration Registers for Programmed Segmentation Control

There are three CP0 registers that are used to configure the characteristics of the segments. Each register is divided into two 16-bit sections; each section is used to configure one segment.
Here is an expanded Segment Address Table with the CP0 registers and the bit sections that correspond to each segment:

Expanded Segment Address Table

Segment Number	CP0 Register	Segment Bits	Size	Starting Virtual Address	Ending Virtual Address
0 (CFG0)	SegCtl0 5,2	0 - 15	.5 GB	0xE000 0000	0xFFFF FFFF
1 (CFG1)	SegCtl0 5,2	16 - 31	.5 GB	0xC000 0000	0xDFFF FFFF
2 (CFG2)	SegCtl1 5,3	0 - 15	.5 GB	0xA000 0000	0xBFFF FFFF
3 (CFG3)	SegCtl1 5,3	16 - 31	.5 GB	0x8000 0000	0x9FFF FFFF
4 (CFG4)	SegCtl2 5,4	0 - 15	1 GB	0x4000 0000	0x7FFF FFFF
5 (CFG5)	SegCtl2 5,4	16 - 31	1 GB	0x0000 0000	0x3FFF FFFF

Configuration Register Fields

Register Fields		CP0 SegCtl0-2	Reset State
Name	Bits		
PA	11:9 and 27:25	Physical address bits 31:29 for segment. These bits are used when the virtual address space is configured as kernel unmapped or EU is set (and ERL = 1), to select the segment in memory to be accessed.	Configuration Dependent
AM	6:4 and 22:20	Access control mode	Configuration Dependent
EU	3 and 19	Error condition behavior. If set, Configured segment becomes unmapped and uncached when Status.ERL = 1 (reset, NMI or cache error)	Configuration Dependent
C	2:0 and 18:16	Cache Coherency Attribute when Unmapped	Configuration Dependent

Description	**Mode Name**	**Value of bits**
Unmapped Kernel Segment	UK	000
Mapped Kernel Segment	MK	001
Mapped Supervisor and Kernel Segment	MSK	010
Mapped User, Supervisor and Kernel Segment	MUSK	011
Mapped User and Supervisor and Unmapped Kernel	MUSUK	100
Unmapped Supervisor and Kernel Segment	USK	101
Unrestricted Unmapped Segment	UUSK	111

Description	**Mode Name**	**Value of bits**
Uncached, non-coherent	UC	010
Writeback, write-allocate, non-coherent	WB	011
Writeback, write-allocate, coherent, exclusive	CWBE	100
Writeback, write-allocate, coherent, exclusive on write	CWB	101
Uncached accelerated, non-coherent	UCA	111

4 Programmed Segmentation Control Configuration for MIPS Legacy Memory Map

As an example of how to configure all the segments, let's look at how Programmed Segmentation Control is used to configure a core to look just like the memory mapping prior to Programmed Segmentation Control.

CP0 Register	Segment	PA Bits 31:29	AM (Access mode)	EU	C (Cache Coherency Attribute)	VAR (Virtual Address Range)	LM (Legacy Mode Segment)
SegCtl0 (5,2)	CFG0 Bits 15 -0	na	MK - 001 (Mapped Kernel)	0	na (TLB)	0xFFFF FFFF 0xE000 0000	KSEG3
	CFG1 Bits 31 - 16	na	MSK - 010 (Mapped Supervisor/Kernel)	0	na (TLB)	0xDFFF FFFF 0XC000 0000	KSEG2 KSSEG
SegCtl1 (5,3)	CFG2 Bits 15 -0	000	UK - 000 (Unmapped Kernel)	1	010 (uncached)	0xBFFF FFFF 0xA000 0000	KSEG1
	CFG3 Bits 31 - 16	000	UK - 000 (Unmapped Kernel)	1	Config0 K0 bit determines CCA	0x9FFF FFFF 0x8000 0000	KSEG0
SegCtl2 (5,4)	CFG4 Bits 15 -0	010	MUSK - 011 (Mapped User/Supervisor/Kernel)	1	na (TLB)	0x7FFF FFFF 0x4000 0000	KUSEG
	CFG5 Bits 31 - 16	000	MUSK - 011 (Mapped User/Supervisor/Kernel)	1	na (TLB)	0x3FFF FFFF 0x0000 0000	

Let's now look the Programmed Segmentation Control set up for a KUSEG segment:

1	2	3	4	5	6	7	8
CP0 Register	**Segment**	**PA Bits 31:29**	**AM (Access mode)**	**EU**	**C (Cache Coherency Attribute)**	**VAR (Virtual Address Range)**	**LM (Legacy Mode Segment)**
SegCtl2 (5,4)	**CFG4 Bits 15 - 0**	**010**	**MUSK – 011 (Mapped User/Supervisor/Kernel)**	**1**	**NA**	**0x7FFF FFFF 0x4000 0000**	**KUSEG**
	CFG5 Bits 31 - 16	**000**	**MUSK – 011 (Mapped User/Supervisor/Kernel)**	**1**	**NA**	**0x3FFF FFFF 0x0000 0000**	

The legacy KUSEG is a segment accessible in Kernel, Supervisor, or User Mode with an address range that covers the lowest 2 GB of virtual memory and can be mapped through the TLB to any Physical address. When in an error state (Status.ERL = 1), KUSEG becomes direct-mapped (without the TLB) and uncached to the lowest 2 GB of Physical memory, 0x0000 0000 through 0x7FFF FFFF.

In the table above, the SegCtl2 (CP0 register 5, select 4) in Column 1 is used to program KUSEG. This register Controls the configuration for segments 4 and 5 (Column 2) for a virtual address range of 0x0000 0000 to 0x7FFF FFFF (Column 7) the lowest 2 GB of memory. The access mode (Column 4) is 0x011 or Mapped, User, Supervisor and Kernel, which means KUSEG will be accessible through the TLB in all modes for non-error conditions.

EU (Column 5) is set to 1 to indicate that these segments will become unmapped and uncached when the CPU is in an error state (Status.ERL = 1 (reset, NMI or cache error)). The translation in this state is configured by the PA bits (Column 3). Segment 5 (CFG5) sets bits 29 through 31 to 0, causing a mapping of the segment to start at physical address 0, which will cover the first GB of physical memory, 0x0000 0000 through 0x3FFF FFFF. Segment 4 (CFG4) sets bits 29 through 31 to 0x010. This would map the segment to physical address 0x4000 0000 because the top 3 bits (31:29) would be set to 010 and the rest to 0 yielding the 0x4000 0000 address, which covers the second lowest GB of physical memory, 0x4000 0000 through 0x7FFF FFFF.

Now let's look at how KSEG0 is configured using Programmed Segmentation Control:

1	2	3	4	5	6	7	8
CP0 Register	**Segment**	**PA Bits 31:29**	**AM (Access mode)**	**EU**	**C (Cache Coherency Attribute)**	**VAR (Virtual Address Range)**	**LM (Legacy Mode Segment)**
SegCtl1 (5,3)	**CFG3 Bits 15 - 0**	**000**	**UK - 000 (Unmapped Kernel)**	**1**	**Config0 K0 bit determines CCA**	**0x9FFF FFFF 0x8000 0000**	**KSEG0**

The legacy KSEG0 is a segment accessible in Kernel mode with a virtual address range of 0x8000 0000 through 0x9FFF FFFF, 512 MB that is unmapped and cacheable. For the legacy translation unmapped means that KSEG0 translates to physical address 0.

In the table above, SegCtl1 (CP0 register 5, select 3) in Column 1 is used to program KSEG0. Bits 0 – 15 of this register control the configuration of segment 3 (Column 2) for a virtual address range (Column 7) 0x8000 0000 to 0x9FFF FFFF (512 MB). The access mode (Column 4) is 0x000 or unmapped, Kernel. The unmapped Access mode requires the setting of the Physical Address (PA) bits. In this case, the PA bits (Column 3) are set to 0x000, so this segment will directly map to physical address 0 covering the first 512 MB of physical memory, 0x0000 0000 through 0x1FFF FFFF. The cache coherency attribute is determined by the K0 field in the CP0 config0.

EU (Column 5) is set to 1 to indicate that these sections will become uncached when the CPU is in the error state (Status.ERL = 1 (reset, NMI or cache error)).

Let’s now look the Programmed Segmentation Control set up for a KSEG1 configuration:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
CP0 Register	**Segment**	**PA Bits 31:29**	**AM (Access mode)**	**EU**	**C (Cache Coherency Attribute)**	**VAR (Virtual Address Range)**	**LM (Legacy Mode Segment)**
SegCtl1 (5,3)	**CFG2 Bits 31 - 16**	**000**	**UK - 000 (Unmapped Kernel)**	**1**	**010 (uncached)**	**0xBFFF FFFF 0xA000 0000**	**KSEG1**

The legacy KSEG1 is a segment accessible in Kernel mode with a virtual address range of 0xA000 0000 through 0xBFFF FFFF, 512 MB that is unmapped and uncached. For the legacy mapping unmapped means that KSEG1 translates to physical address 0.

In the table above, the SegCtl1 (CP0 register 5, select 3) in Column 1 is the register that will be used to program KSEG0. Bits 15 – 31 of this register controls the configuration for segment 2 (Column 2) for a virtual address range of 0xA000 0000 to 0xBFFF FFFF (Column 7) a 512 MB range of memory. The access mode (Column 4) is 0x000 or unmapped, Kernel. The unmapped Access mode requires the setting of the Physical Address (PA) bits. In this case the PA bits (Column 3) are set to 0x000, so this segment will directly map to physical address 0 covering the first 512 MB of physical memory (0x0000 0000 through 0x1FFF FFFF). The cache coherency attribute is uncached so all memory accesses to this segment will go directly to memory.

Note the EU bit is set so this segment will be accessible in error state, Status.ERL = 1 (reset, NMI or cache error).

Now for the last 2 segments KSEG2, KSEG3 or a combination of the two called KSSEG:

CP0 Register	Segment	PA Bits 31:29	AM (Access mode)	EU	C (Cache Coherency Attribute)	VAR (Virtual Address Range)	LM (Legacy Mode Segment)
SegCtl0 (5,2)	CFG0 Bits 31 - 16	na	MK - 001 (Mapped Kernel)	0	na (TLB)	0xFFFF FFFF 0xE000 0000	KSEG3
	CFG1 Bits 15 -0	na	MSK - 010 (Mapped Supervisor/Kernel)	0	na (TLB)	0xDFFF FFFF 0xC000 0000	KSEG2 KSSEG

The names and ranges for these segments usually vary for each OS. Sometimes it is viewed as 2, 512 MB virtual address segments, KSEG2 and KSEG3, starting at 0xC000 0000 and 0xE000 0000, respectively. Sometimes as a segment called KSEG2 or KSSEG that covers 1 GB of virtual address starting at 0xC000 0000. No matter what they are called, the segments are always mapped (only accessible through the TLB), cacheable, and not accessible in User Mode. This example will show two segments of 512 MB and refer to them as KSEG2 and KSEG3.

In the table the SegCtl0 (CP0 register 5, select 2) in Column 1 is the register that will be used to program these segments. This register controls the configuration for segments 1 and 0 (Column 2). Segment 1 with a virtual address range of 0xC000 0000 to 0xDFFF FFFF (Column 7) will cover KSEG2. Segment 0 with a virtual address range of 0xE000 0000 to 0xFFFF FFFF (Column 7) will cover KSEG3. The access mode (Column 4) is 0x010 or Mapped, Supervisor and Kernel for KSEG2, which means that KSEG2 will be accessible through the TLB in Supervisor and Kernel modes for non-error conditions. To show the difference between Supervisor, Kernel and only Kernel access mode for KSEG3, the access mode (Column 4) is 0x001 or Mapped and Kernel, which means KSEG3 will be accessible through the TLB in Kernel modes only for non-error conditions.

Note the EU bit is not set, so these segments will not be accessible in the error state (Status.ERL = 1 (reset, NMI or cache error)).

5 Programmed Segmentation Control Configuration for Extended Virtual Kernel and User Address Spaces (EVA)

The main purpose of Programmed Segmentation Control is to expand the mapped virtual and physical address space available in User mode, to expand the unmapped virtual and physical address space in Kernel mode, and to be able to overlap the two to make it easy for an OS Kernel to access the User address space.

The table below shows just that. It configures a virtual memory from 0x0000 0000 to 0xBFFF FFFF as the first 3 GB of virtual memory, accessible in User mode as a mapped region. For that same virtual range, it sets up Kernel mode access as unmapped and translated to the lower 3 GB of physical memory. It does this by using segments 2 through 5 (CFG2 –CFG5).

The top two virtual memory segments (CFG0 and 1) are set up as mapped Kernel mode only.

CP0 Register	Segment	PA Bits 31:29	AM (Access mode)	EU	C (Cache Coherency Attribute)	VAR (Virtual Address Range)
SegCtl0 (5,4)	CFG0 Bits 31 - 16	na	MK - 001 (Mapped Kernel)	0	na (TLB)	0xFFFF FFFF 0xE000 0000
	CFG1 Bits 15 -0	na	MSK - 001 (Mapped Kernel)	0	na (TLB)	0xDFFF FFFF 0XC000 0000
SegCtl1 (5,3)	CFG2 Bits 31 - 16	101	MUSUK - 100 (Mapped User/Supervisor Unmapped Kernel)	1	011 (cached, write back)	0xBFFF FFFF 0xA000 0000
	CFG3 Bits 15 - 0	100	MUSUK - 100 (Mapped User/Supervisor Unmapped Kernel)	1	011 (cached, write back)	0x9FFF FFFF 0x8000 0000
SegCtl2 (5,2)	CFG4 Bits 31 - 16	010	MUSUK - 100 (Mapped User/Supervisor Unmapped Kernel)	1	011 (cached, write back)	0x7FFF FFFF 0x4000 0000
	CFG5 Bits 15 - 0	000	MUSUK - 100 (Mapped User/Supervisor Unmapped Kernel)	1	011 (cached, write back)	0x3FFF FFFF 0x0000 0000

6 Programming for Legacy Mode and EVA mode

In addition to setting the memory configuration as described above, there are several other registers that control the legacy mode configuration and configuring the core for EVA mode. This section will cover these registers.

Name	Bits	Description	Read/Write	Reset state
K	30	0: Config K0 overrides segment configuration cache mode for CFG3 (Legacy KSEG0) Segment. 1: Config K0 disabled.		
		SI_EVAReset pin deasserted at reset	RW	0
		SI_EVAReset pin asserted at reset,	R	1
CV	29	Cache error exception vector control. Disables logic forcing use of CFG2 (Legacy KSEG1) segment in the event of a Cache Error exception when StatusBEV = 0.	R/W	0

6.1 CP0 Config5 (register 16, Select6)

Config5 is a new CP0 configuration register. There are 3 fields in the register:

EVA	28	This bit is always a logic one to indicate support for enhanced virtual address (EVA).	R	1

- The K bit controls the use of the K0 field in the CP0 Config register. For non-EVA legacy cores, the K0 field controls the cache coherency attribute of the KSEG0 virtual memory segment. To make an EVA core compatible with legacy software, the CFG3 segment that covers the same range of virtual address as did the KSEG0 segment can have its cacheability controlled by the K0 field. If the K bit is 0, then the cacheability setting in the K0 field will override the C field in the CFG3 segment configuration. If the K bit is 1, the setting of the K0 filed will have no effect on the cacheability of the CFG3 segment. The K bit is set by the SI_EVAReset pin on reset. If the pin is deserted, then the reset state is 0, the cacheability of CFG3 is controlled by the K0 field, and the K bit is writable which enables switching from Legacy compatibility to an EVA Setting. If the pin is asserted, the K0 field has no effect, the K field is set to 1 and is read-only, thus disallowing any change from EVA Setting to Legacy compatibility setting.

  In addition to selecting the location of the cache coherency attributes for the CFG3 segment, the CONFIG5.K bit also causes hardware to generate two virtual boot exception overlay segments, one for KSEG0 and one for KSEG1, with the Boot Exception Vector at 0xBfC0 0000 in KSEG1 and mirrored at 0x9FC0 0000 in KSEG0.

- The CV field controls the cache error exception vector. For non-EVA legacy Cores, the cache error exception is forced to virtual address 0xA000 0100 located in the uncached KSEG1 memory segment.

  To make an EVA core compatible with legacy software, when the CV bit is cleared bits 31:29 are forced to a binary 101 thus forcing the address into the CFG2 segment which corresponds to the old KSEG1 segment.

  If the CV bit is set a cache error exception does not force bits 31:29 and uses the full 20 bits of the Ebase register for the vector.

- The EVA bit indicates support for the enhanced virtual addressing and will always be set for interAptiv or proAptiv cores.

6.2 CP0 Ebase (register 15, Select 1)

The Exception Base register is used for the base address for exceptions after the boot process has completed and Status[BEV]is set to 0. For non-EVA legacy Cores, bits 31:30 of the Exception Base address are not writable and are set to a binary 10. This forces the Exception to a virtual address within KSEG0 or KSEG1.

There is a new one-??bit field (11) call WG or the Write Gate bit. To make an EVA core compatible with legacy software, on reset bits 31:30 are set to a binary 10 and these bits are unchanged on writes to Ebase when WG=0 in the value being written. This forces the exception bass address into the CFG2 and CFG3 segments, which correspond to the old KSEG0 and KSEG1 legacy segments.

If WG=1 in the written value, then bits 31:30 are overwritten.

NOTE: The CV bit in the Config5 register as previously described has a different effect on the Exception Base address in that it forces bits 31:29 to a binary 101, regardless of the setting of bit 31:29 in this register.

6.3 Boot Exception Vector Overlay

To add more flexibility to an EVA core, both the virtual and physical address of the Boot Exception Vector can be changed. This is called the Boot Exception Vector Overlay (or BEV Overlay) because it overlays part of the configuration for the memory segment it is in. The BEV Overlay is always present whether or not the core is in EVA mode. The next two GCR registers described below are used to configure the BEV Overlay.

6.4 GCR Core Local Reset Exception Base Register

The BEVExceptionBase field in the Core Local Reset Exception Base Register controls where the CPU will fetch the first instruction on cold reset. For Legacy compatibility mode, the start of the Boot Exception Vectors is located at virtual address 0xBFC0 0000 and is set as the default state for the GCR Core Local Reset Exception Base Register. If you want a Core to cold boot from a different address that is not legacy mode, this register can be configured at IP configuration time for a different address.

This register along with some of the settings in the next register determine the start of the BEV Overlay.

6.5 Core Local Reset Exception Extended Base Register

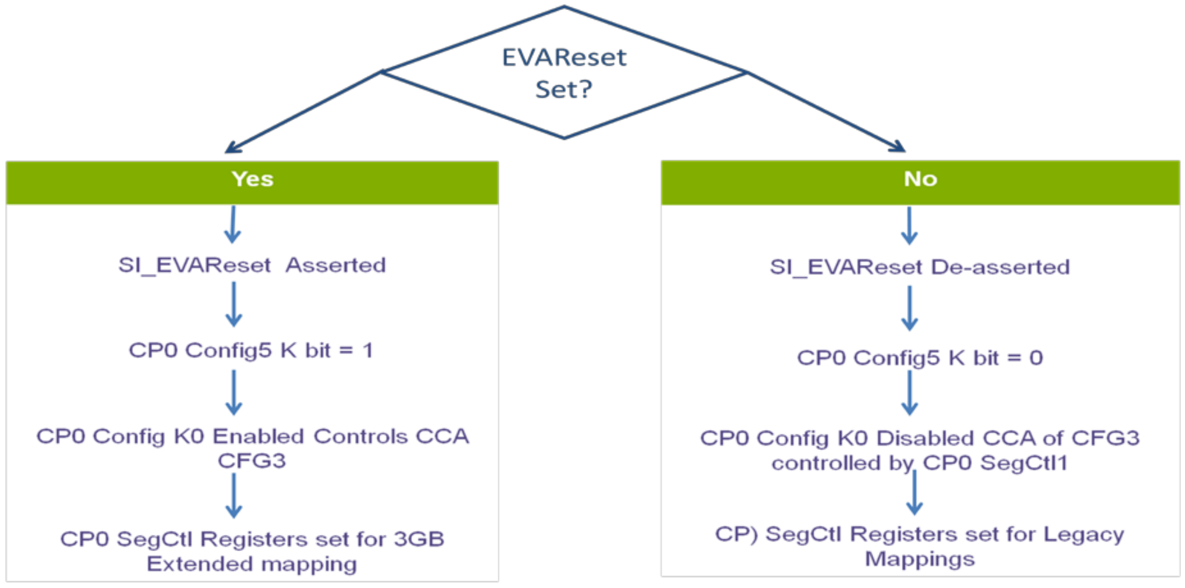
The Core Local Reset Exception Extended Base Register configures the size and location in physical memory of the BEV Overlay. It is also used to control EVA mode and the address of the Exception Vector. This is a per core register, so it is in the Core-Local section of the Global Configuration Registers. The initial values of this register are set at core build time and do not need to be changed.

- The EVAReset Bit (31) indicates if this core will use the legacy virtual boot exception vector addresses.

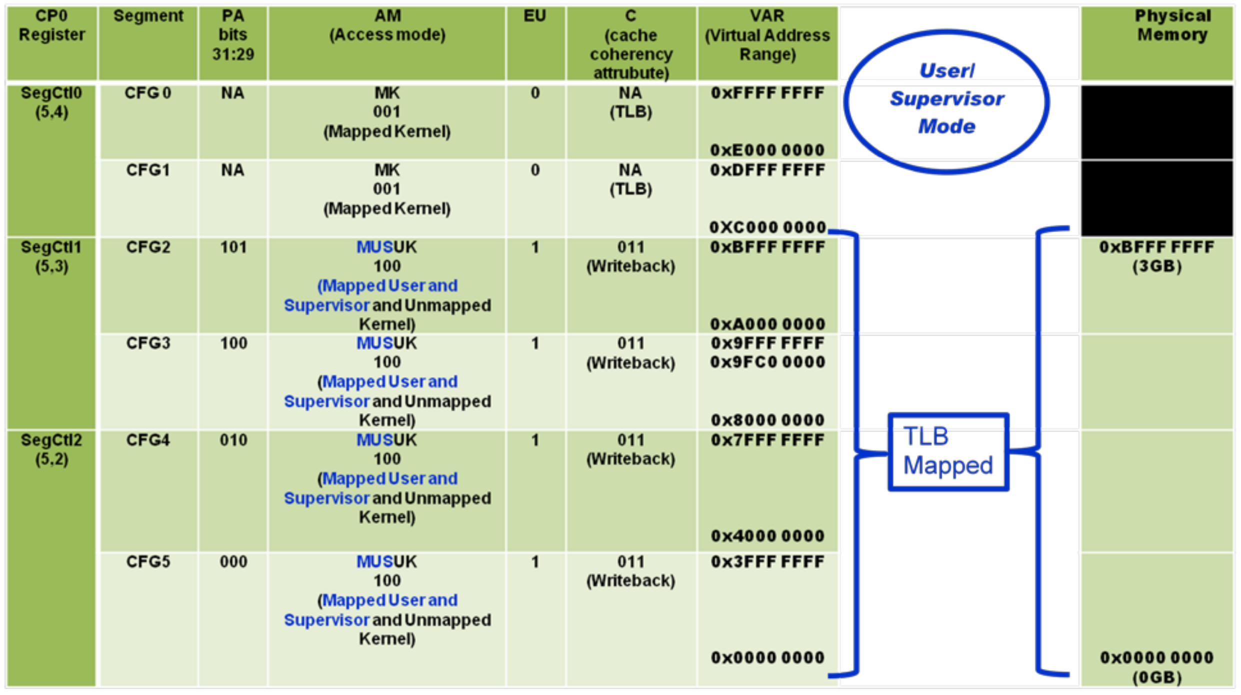
  The initial value of EVAReset is set at IP configuration time. The EVAReset bit controls the SI_EVAReset pin. This pin is driven by the CM to the core.

If EVAReset is 0, the SI_EVAReset pin is de-asserted, which drives the CP0 Config5 K bit to 0. The K bit = 0 enables the K0 field in the CP0 Config register, which will control the CCA of CFG3, and the CP0 segmentation control registers will reflect Legacy mappings for KUSEG, KSEG0, KSEG1, and KSEG2/3. In other words it will map and behave just like a Legacy core.

If EVAReset is 1, the SI_EVAReset pin is asserted, and the CP0 Config5 K bit will be set and be unchangeable. This means that the state of the core is not Legacy-compatible and can never be set for legacy compatibility. The K0 field in the CP0 Config register is disabled and the CCA of CGF3 is controlled by the settings in the SegCtl1 register. The CP0 segmentation control registers will reflect an EVA mapping for a 3GB RAM region.

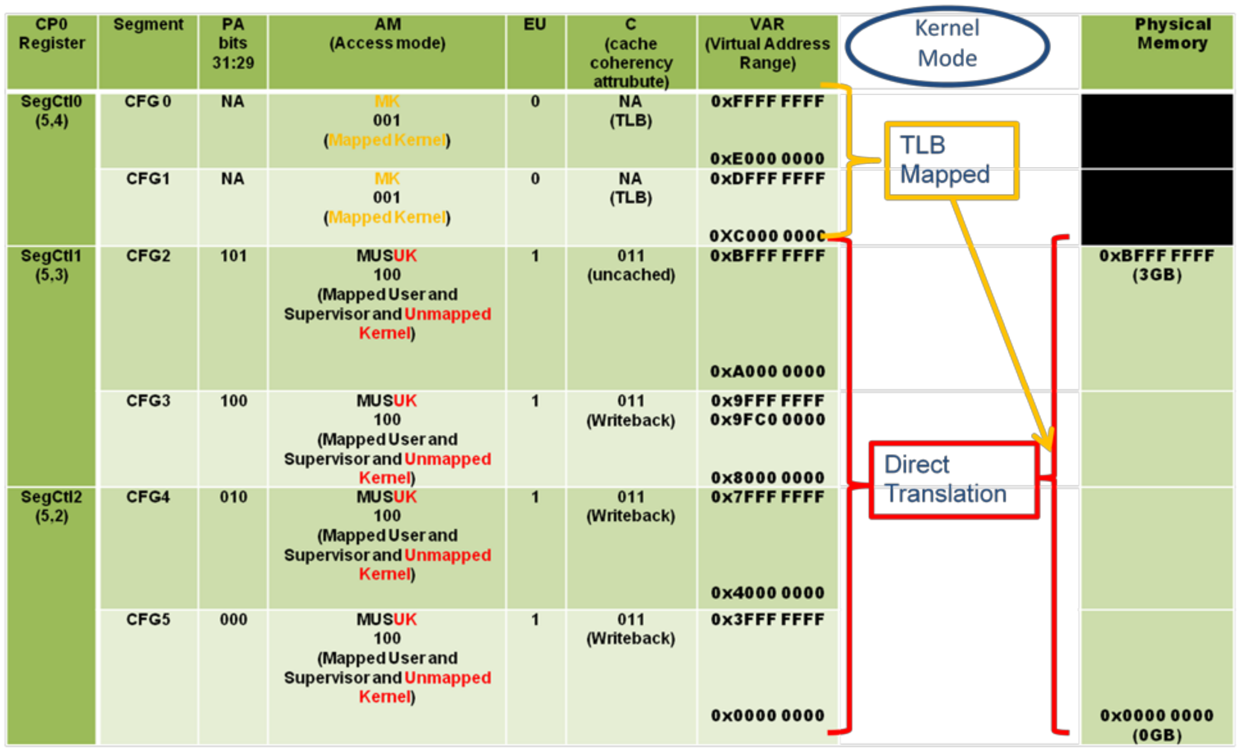

EVAReset Set?
Yes
SI_EVAReset Asserted
CP0 Config5 K bit = 1
CP0 Config K0 Enabled Controls CCA CFG3
CP0 SegCtl Registers set for 3GB Extended mapping
No
SI_EVAReset De-asserted
CP0 Config5 K bit = 0
CP0 Config K0 Disabled CCA of CFG3 controlled by CP0 SegCtl1
CP) SegCtl Registers set for Legacy Mappings

Below are tables that show the preset settings for each addressing mode for the 3GB settings.

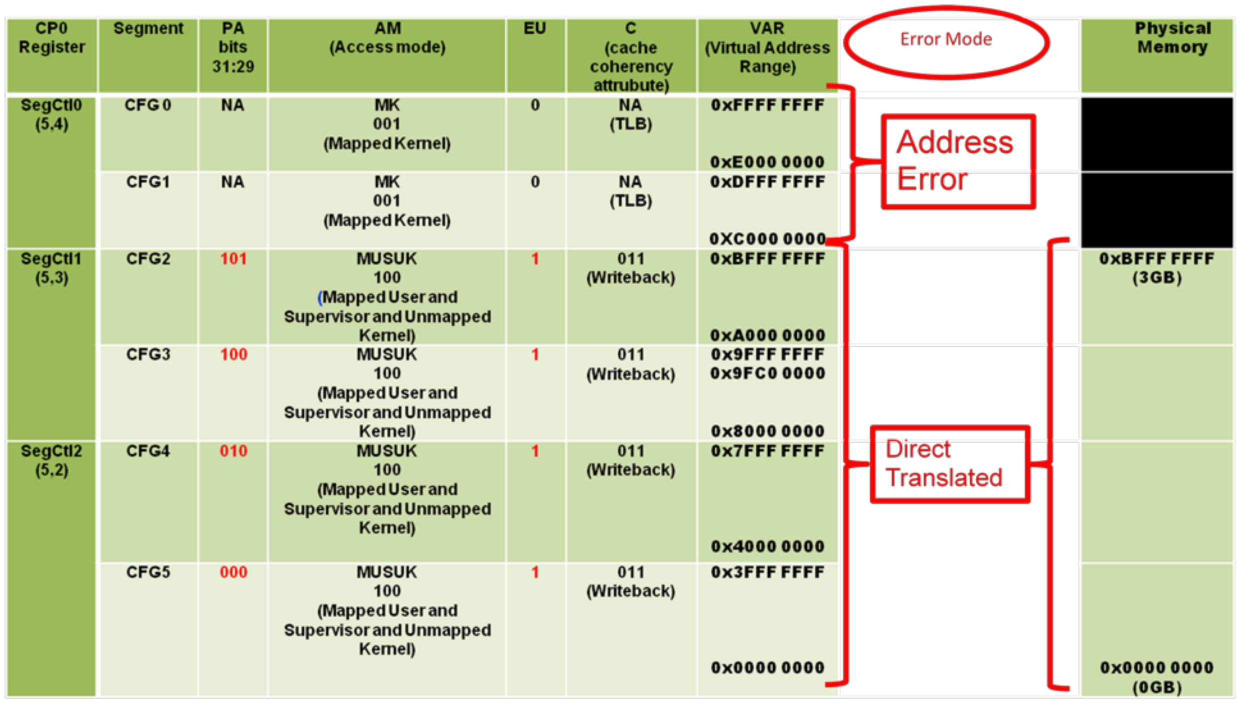


CP0 Register	Segment	PA bits 31:29	AM (Access mode)	EU	C (cache coherency attrubute)	VAR (Virtual Address Range)		Physical Memory
SegCtl0 (5,4)	CFG 0	NA	MK 001 (Mapped Kernel)	0	NA (TLB)	0xFFFF FFFF 0xE000 0000		■
	CFG1	NA	MK 001 (Mapped Kernel)	0	NA (TLB)	0xDFFF FFFF 0XC000 0000		■
SegCtl1 (5,3)	CFG2	101	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0xBFFF FFFF 0xA000 0000		0xBFFF FFFF (3GB)
	CFG3	100	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x9FFF FFFF 0x9FC0 0000 0x8000 0000		
SegCtl2 (5,2)	CFG4	010	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x7FFF FFFF 0x4000 0000		
	CFG5	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x3FFF FFFF 0x0000 0000		0x0000 0000 (0GB)

User/ Supervisor Mode
TLB Mapped



CP0 Register	Segment	PA bits 31:29	AM (Access mode)	EU	C (cache coherency attrubute)	VAR (Virtual Address Range)		Physical Memory
SegCtl0 (5,4)	CFG 0	NA	MK 001 (Mapped Kernel)	0	NA (TLB)	0xFFFF FFFF 0xE000 0000		■
	CFG1	NA	MK 001 (Mapped Kernel)	0	NA (TLB)	0xDFFF FFFF 0XC000 0000		■
SegCtl1 (5,3)	CFG2	101	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (uncached)	0xBFFF FFFF 0xA000 0000		0xBFFF FFFF (3GB)
	CFG3	100	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x9FFF FFFF 0x9FC0 0000 0x8000 0000		
SegCtl2 (5,2)	CFG4	010	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x7FFF FFFF 0x4000 0000		
	CFG5	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x3FFF FFFF 0x0000 0000		0x0000 0000 (0GB)



CP0 Register	Segment	PA bits 31:29	AM (Access mode)	EU	C (cache coherency attrubute)	VAR (Virtual Address Range)	Error Mode	Physical Memory
SegCtl0 (5,4)	CFG 0	NA	MK 001 (Mapped Kernel)	0	NA (TLB)	0xFFFF FFFF 0xE000 0000	Address Error	■
	CFG1	NA	MK 001 (Mapped Kernel)	0	NA (TLB)	0xDFFF FFFF 0XC000 0000		■
SegCtl1 (5,3)	CFG2	101	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0xBFFF FFFF 0xA000 0000	Direct Translated	0xBFFF FFFF (3GB)
	CFG3	100	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x9FFF FFFF 0x9FC0 0000 0x8000 0000		
SegCtl2 (5,2)	CFG4	010	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x7FFF FFFF 0x4000 0000		
	CFG5	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	011 (Writeback)	0x3FFF FFFF 0x0000 0000		0x0000 0000 (0GB)

To aid flexibility in booting a core other than Core 0, this register can also be programmed for other cores to boot at a different boot exception vector addresses when they are powered up. The boot code for each core will need to be linked to the addresses programmed in the GCR Core Local Reset Exception Base Register.

- The LegacyUseExceptionBase bit (30) will force the Boot Exception vector address that is in the Core-Local Reset Exception Base Register to be located in segments CFG2 and CFG3, which correspond to the Legacy KSEG0 and KSEG1 by overriding bits 31:30 and forcing them to be 1 0.

- The BEVExceptionBaseMask bits (27:20) determine the size of the Overlay region from 1 MB to 256 MB in powers of two. The initial value is set at core build time. The size also determines where the Overlay starts. The Overlay will start on a boundary that corresponds to the size of the Overlay.

Mask bits 27-20	Size in MB
0000 0000	1
0000 0001	2
0000 0011	4
0000 0111	8
0000 1111	16
0001 1111	32
0011 1111	64
0111 1111	128
1111 1111	256

- BEV Exception Base PA bits (7:1) sets bits (35:29) of the Physical Base address of the Boot Exception Vectors. The Boot Exception Vector Base Physical address is a 7-bit field, but the current cores only use the first 3 bits. The remaining bits of the address come from the Core Local Reset Exception Base Register. This allows the physical address of the boot exception vector to be placed in any segment. This address and the BEV Exception Base Mask determine the location of the Overlay in physical memory.

- The Present bit (0) is always set if this register is present. It is a read-only bit.

6.6 Overlay Example setting the Physical address to 0xBFC0 0000

This example sets the registers for a core set up at build time to boot in EVA mode with the Boot Exception Vector at 0xBFC0 0000 both for the virtual and Physical with a 16MB region.

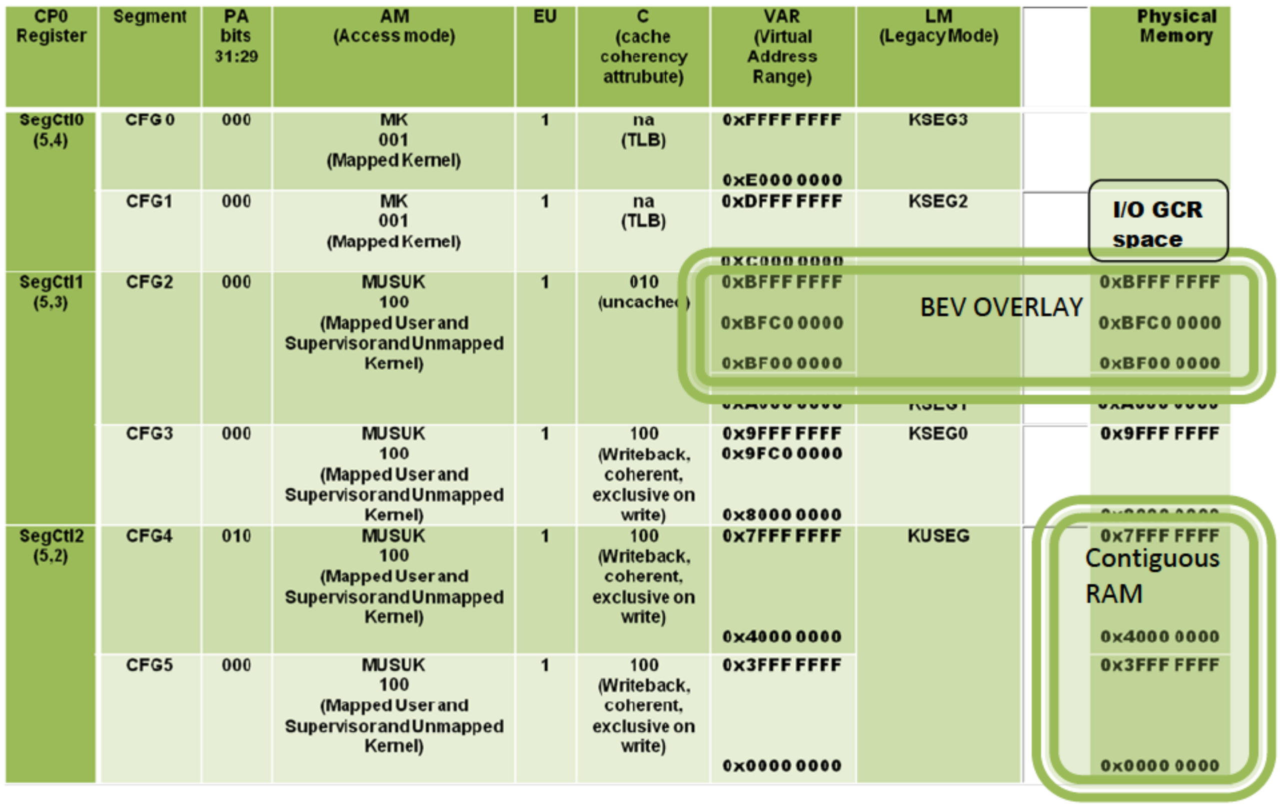
- The BEVExceptionBase field in the Core Local Reset Exception Base Register is set to 0XBFC0 0000, which sets the virtual address of the Boot Exception Vector.
- The Core Local Reset Exception Extended Base Register would be set to 0x80F0 000B. Here is how that value is arrived at:
    - The LegacyUseExceptionBase is cleared, so bits 31:30 will remain as they are in the Core-Local Reset Exception Base Register.

    - The BEVExceptionBaseMask bits 27:20 are set to 0x0F. This sets the size of the address range to map to 16MB.
    - The BEVExceptionBasePA bits 7:1 are set to 0x5. Combining these with bits 28:12 of the Core-Local Reset Exception Base Register sets the Physical address to 0xBFC0 0000.
- Here is the address map for the Overlay section.

Overlay With 16MB Mask and address of BEV of 0xBFC0 0000		
Virtual	Physical	
0xBFFF FFFF	0xBFFF FFFF	End of Overlay Region
0xBFC0 0000	0xBFC0 0000	Boot Exception Vector
0xBF00 0000	0xBF00 0000	Start of Overlay Region

6.7 Moving I/O Registers and GCR base

Below is an example of a 2GB physical RAM memory map when the boot exception vector overlay moves the physical address to 0xBFC0 0000 as setup in the previous section. If the system is set up with an overlay that changes the physical address of the boot exception vector to the BFC0 0000 and if the I/O register placement was also placed at an address above the first 2 GB then there is no longer a hole preventing a contiguous block of RAM memory. NOTE the Default value of the Global Configuration Base register is set to 0x1FBF 8000 this will also need to be changed to a address outside of the RAM memory space.



CP0 Register	Segment	PA bits 31:29	AM (Access mode)	EU	C (cache coherency attrubute)	VAR (Virtual Address Range)	LM (Legacy Mode)		Physical Memory
SegCtl0 (5,4)	CFG 0	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xFFFF FFFF 0xE000 0000	KSEG3		
	CFG1	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xDFFF FFFF 0xC000 0000	KSEG2		I/O GCR space
SegCtl1 (5,3)	CFG2	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	010 (uncached)	0xBFFF FFFF 0xBFC0 0000 0xBF00 0000 0xA000 0000	BEV OVERLAY KSEG1		0xBFFF FFFF 0xBFC0 0000 0xBF00 0000 0xA000 0000
	CFG3	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x9FFF FFFF 0x9FC0 0000 0x8000 0000	KSEG0		0x9FFF FFFF 0x8000 0000
SegCtl2 (5,2)	CFG4	010	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x7FFF FFFF 0x4000 0000	KUSEG		0x7FFF FFFF Contiguous RAM 0x4000 0000
	CFG5	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x3FFF FFFF 0x0000 0000			0x3FFF FFFF 0x0000 0000

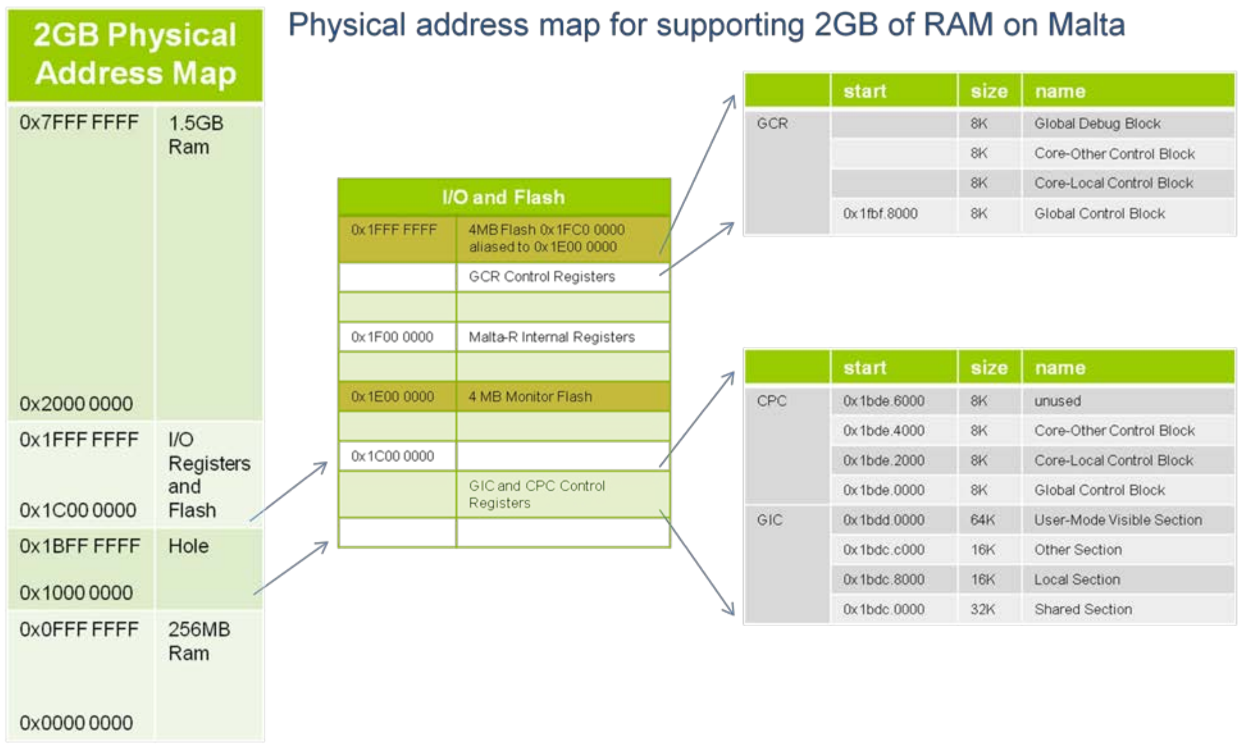
7 New Load and Store Instructions to support EVA

The MIPS32 R3 architecture includes load and store instructions that allow the Kernel to access User space as if it were the process whose ASID is currently set in the CP0 EntryHi register. For example, this makes it easier and faster for the Kernel to copy from a User virtual address to a Kernel physical address during a System call. The Kernel can use the new load instructions with the user virtual address passed to it in the call to load from User space, without having to translate the address.

The names for these instructions just have an E appended to the normal non-EVA load/store instruction name. The load instructions are LBE, LBUE, LHE, LHUE, LLE, LWE, LWLE, and LWRE. The store instructions are SBE, SCE, SHE, SWE, SWLE, and SWRE. All of these instructions have the same meaning as their non-EVA counterparts, except that they are Kernel-mode instructions that use the User-mode translation of the address for the load or store, based on the current EntryHi ASID value.

8 Linux Example Using 2GB RAM on a Malta Platform

This is an example of using a 2GB RAM on a MIPS Malta Evaluation Board using an FPGA to expand the available RAM memory in Linux. For a Malta FPGA bitfile that supports EVA, the maximum addressable memory is 1.5GB+256MB, although the board has 2GB module. This is a limitation of the memory controller on the Malta board.


Physical address map for supporting 2GB of RAM on Malta

2GB Physical Address Map	
0x7FFF FFFF 0x2000 0000	1.5GB Ram
0x1FFF FFFF 0x1C00 0000	I/O Registers and Flash
0x1BFF FFFF 0x1000 0000	Hole
0x0FFF FFFF 0x0000 0000	256MB Ram

I/O and Flash	
0x1FFF FFFF	4MB Flash 0x1FC0 0000 aliased to 0x1E00 0000
	GCR Control Registers
0x1F00 0000	Malta-R Internal Registers
0x1E00 0000	4 MB Monitor Flash
0x1C00 0000	
	GIC and CPC Control Registers

	start	size	name
GCR		8K	Global Debug Block
		8K	Core-Other Control Block
		8K	Core-Local Control Block
	0x1fbf.8000	8K	Global Control Block

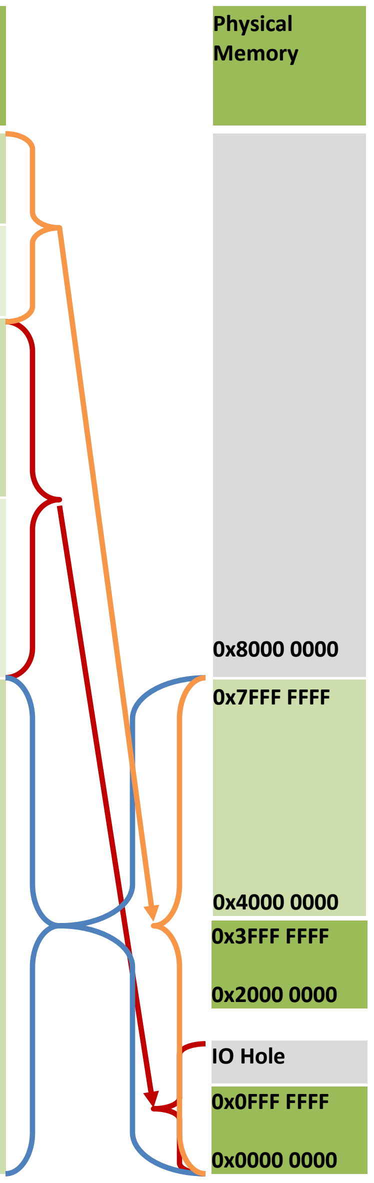
	start	size	name
CPC	0x1bde.6000	8K	unused
	0x1bde.4000	8K	Core-Other Control Block
	0x1bde.2000	8K	Core-Local Control Block
	0x1bde.0000	8K	Global Control Block
GIC	0x1bdd.0000	64K	User-Mode Visible Section
	0x1bdc.c000	16K	Other Section
	0x1bdc.8000	16K	Local Section
	0x1bdc.0000	32K	Shared Section

The 1st RAM region is located at PA=0x0000.0000 – 0x0FFF.FFFF, 256MB. The 2nd RAM region is located at PA=0x2000.0000 – 0x7FFF FFFF 1.5GB. There is an “I/O hole” between the two RAM memory regions used for the boot flash, I/O device registers, and the memory-mapped GCRs.

Since the I/O address used is the same as a legacy core’s bitfile, no device drivers need to change.

8.1 EVA Memory Map for Linux on Malta FPGA – 2GB RAM Kernel Mode

CP0 Register	Segment	PA bits 31:29	AM (Access mode)	EU	C (cache coherency attrubute)	VAR (Virtual Address Range)	LM (Legacy Mode)
SegCtl0 (5,4)	CFG 0	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xFFFF FFFF 0xE000 0000	KSEG3
	CFG1	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xDFFF FFFF 0XC000 0000	KSEG2
SegCtl1 (5,3)	CFG2	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	010 (uncached)	0xBFFF FFFF 0xA000 0000	KSEG1
	CFG3	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x9FFF FFFF 0x8000 0000	KSEG0
SegCtl2 (5,2)	CFG4	010	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x7FFF FFFF 0x4000 0000	KUSEG
	CFG5	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x3FFF FFFF 0x0000 0000	

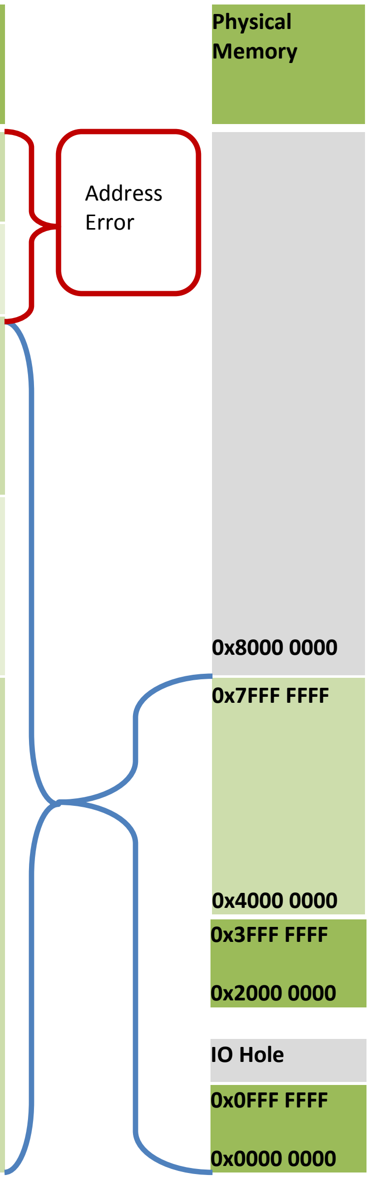


For Kernel Mode Access:

- The segments shown in red, CFG3 and CFG2, correspond to the old KGES0 and KSEG1 and direct map to the lower 512 MB of the physical address space, including the low 256MB of RAM and the I0 Space.
- Segments shown in blue, CFG5 and CFG4, are directly mapped to the lower 2GB of the physical address space, including 2 RAM memory blocks and the I/O space.
- Shown in orange, Segments CFG1 and CFG0 correspond to the old KSEG2 and KSGE3 and are both mapped through the TLB.

8.2 EVA Memory Map for Linux on Malta FPGA - 2GB RAM User Mode

CP0 Register	Segment	PA bits 31:29	AM (Access mode)	EU	C (cache coherency attrubute)	VAR (Virtual Address Range)	LM (Legacy Mode)
SegCtl0 (5,4)	CFG 0	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xFFFF FFFF 0xE000 0000	KSEG3
	CFG1	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xDFFF FFFF 0XC000 0000	KSEG2
SegCtl1 (5,3)	CFG2	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	010 (uncached)	0xBFFF FFFF 0xA000 0000	KSEG1
	CFG3	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x9FFF FFFF 0x8000 0000	KSEG0
SegCtl2 (5,2)	CFG4	010	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x7FFF FFFF 0x4000 0000	KUSEG
	CFG5	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x3FFF FFFF 0x0000 0000	

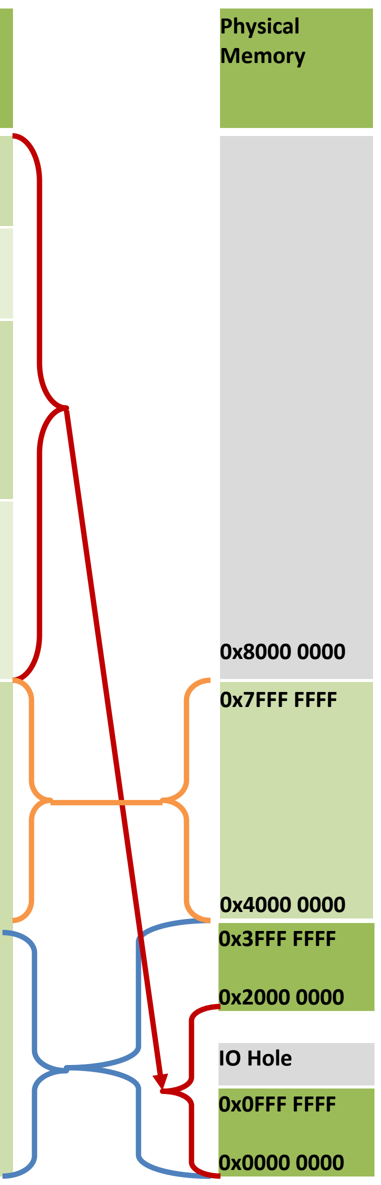


For User Mode Access:

- Segments shown in blue, CFG2, CFG3, CFG4 and CFG5, are all mapped through the TLB.
- Shown in Red, Segments CFG0 and CFG1 are not accessible.

8.3 EVA Memory Map for Linux on Malta FPGA - 2GB RAM EU - Status.ERL = 1 (reset, NMI, or cache error)

CP0 Register	Segment	PA bits 31:29	AM (Access mode)	EU	C (cache coherency attrubute)	VAR (Virtual Address Range)	LM (Legacy Mode)
SegCtl0 (5,4)	CFG 0	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xFFFF FFFF 0xE000 0000	KSEG3
	CFG1	000	MK 001 (Mapped Kernel)	1	na (TLB)	0xDFFF FFFF 0XC000 0000	KSEG2
SegCtl1 (5,3)	CFG2	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	010 (uncached)	0xBFFF FFFF 0xA000 0000	KSEG1
	CFG3	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x9FFF FFFF 0x8000 0000	KSEG0
SegCtl2 (5,2)	CFG4	010	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x7FFF FFFF 0x4000 0000	KUSEG
	CFG5	000	MUSUK 100 (Mapped User and Supervisor and Unmapped Kernel)	1	100 (Writeback, coherent, exclusive on write)	0x3FFF FFFF 0x0000 0000	



For Error Mode:

- Segments shown in red, CFG0, CFG1, CFG2, and CFG3, correspond to the old KGES3, KSEG2, KSEG1 and KSEG0 respectively and all direct map to the lower 512 MB of the physical address space, including the low 256MB of RAM and the I/0 Space.
- Shown in blue, segment CFG5 is directly mapped to the lower 1GB of the physical address space, including three quarters of a GB RAM divided into two parts, with I/O space between.
- Shown in orange, Segment CFG4 is directly mapped to 1GB of physical RAM starting at 0x4000 0000.

8.4 Adding the memory to Linux

For the Linux kernel to use the extra memory, the add_memory_region function is used to register the memory. See arch/mips/mti-malta/malta-memory.c , which shows how this is done for the Malta Board.

Since the mapping and the cache attribute remain the same, existing device driver and the root file system can be used without any modification.

8.5 Kernel Files that changed for Segmentation and EVA

File name	Description
arch/mips/kernel/cpu-probe.c	probe for interaptiv core
arch/mips/kernel/genex.S	add ftlb handler
arch/mips/kernel/r4k_switch.S	change _init_fpu
arch/mips/kernel/scall32-o32.S	setup stack argument in stackargs
arch/mips/kernel/segment.c	create proc entry for segment control
arch/mips/kernel/signal.c	add FPU context switch function call
arch/mips/kernel/smp-cmp.c	add segment check
arch/mips/kernel/spram.c	add interaAptiv SPRAM support
arch/mips/kernel/traps.c	where ftlb handle implemented, add eva trap support
arch/mips/kernel/unaligned.c	emulate lhe, lwe, etc opcode
arch/mips/kernel/mips_ksyms.c	export strncpyxx, copy_from_user, etc
arch/mips/lib/memcpy-inatomic.S	add atomic copy from user, etc function
arch/mips/lib/memcpy.S	__copy_fromuser, __copy_touser, __copy_inuser,
arch/mips/lib/memset.S	add __bzero_user
arch/mips/lib/strlen_user.S	add __strlen_kernel_asm
arch/mips/lib/strncpy_user.S	add __strncpy_from_kernel_asm, __strncpy_from_kernel_nocheck_asm, __strncpy_from_user_asm
arch/mips/lib/strnlen_user.S	add __strnlen_kernel_asm,__strnlen_kernel_nocheck_asm,
arch/mips/mm/cache.c	modify __flush_cache_vmap, __flush_cache_vunmap, export mips_flush_data_cache_range
arch/mips/mm/c-r4k.c	adding D$ flush functions
arch/mips/mm/init.c	modify copy_to_user_page
arch/mips/mm/tlbex.c	add proAptiv support to tlbw function
arch/mips/mm/tlb-r4k.c	use tlbinv to invalidate tlb entry in local_flush_tlb_all()
arch/mips/mti-malta/malta-init.c	program PCI controlller to access 2GB memory
arch/mips/mti-malta/malta-memory.c	add prom_getevamdesc() to setup memory descriptor from bootloader to Linux Kernel

arch/mips/mti-malta/malta-pci.c	shift PCI devices to upper 2GB, to prevent PCI bridges loop
arch/mips/mti-malta/malta-setup.c	define new function plat_eva_setup() to setup segctl register

The changes can be classified as follows:

- Files marked in grey are for generic CPU detection, FPU initialization, and instruction emulation.
- Files marked in peach contain functions that need to copy to user space or from kernel space. These functions have been changed to use the EVA Load/Store instructions. The reason for this is that there is no longer a KUSEG segment, and segmentation control registers are not programmed to be Kernel mapped. In an old legacy core, when data was copied from address space in user mode to address space in kernel mode, lw/sw could be used because both USEG and KUSEG were mapped.
- Files marked in green, have been changed to use the new TLB instructions to invalidate TLB entries.
- Files marked in white are needed by Malta to configure EVA and use the larger RAM memory. This is platform-specific. The key functions for these files is to setup the memory descriptor for Linux to register the appropriate memory that has been detected, register the memory with the add_memory_region function, and setup the segmentation control registers.

8.6 Additional Linux Information

- The macro CKSEG1ADDR in addrspace.h has been changed, depending on settings in the Malta specific spaces.h. This macro should be used when a device driver needs access to an IO control register via an uncached address. The macro supports 3GB RAM space. It is needed because the kernel normally used the Legacy KSEG3 uncached for this purpose, and the corresponding CFG0 segment for 3GB of RAM is configured for use as a DMA zone. It is recommended that you use a similar approach when you need to support 3GB of RAM.
- When the kernel is configured to support 3GB of memory, it will use the CFG0 segment as a DMA zone. This segment will be configured by the kernel as unmapped and uncached, configured to the lower area in physical memory.
- The kernel will configure the CFG2 segment for I/O coherence when there is less than 3 GB of RAM memory in the same way it configures the CFG0 segment when there is 3GB of memory or more.
- In all cases, the CFG1 segment is configured by the kernel as a mapped segment used for vmalloc and loading kernel modules.