

PowerPC Instruction Set Extension Guide

*ISA Support for the
PowerPC APU Controller in
Virtex-4*

EDK, Document Revision 2.0, April 28, 2005





"Xilinx" and the Xilinx logo shown above are registered trademarks of Xilinx, Inc. Any rights not expressly granted herein are reserved. CoolRunner, RocketChips, Rocket IP, Spartan, StateBENCH, StateCAD, Virtex, XACT, XC2064, XC3090, XC4005, and XC5210 are registered trademarks of Xilinx, Inc.



The shadow X shown above is a trademark of Xilinx, Inc.

ACE Controller, ACE Flash, A.K.A. Speed, Alliance Series, AllianceCORE, Bencher, ChipScope, Configurable Logic Cell, CORE Generator, CoreLINX, Dual Block, EZTag, Fast CLK, Fast CONNECT, Fast FLASH, FastMap, Fast Zero Power, Foundation, Gigabit Speeds...and Beyond!, HardWire, HDL Bencher, IRL, J Drive, JBits, LCA, LogiBLOX, Logic Cell, LogiCORE, LogicProfessor, MicroBlaze, MicroVia, MultiLINX, NanoBlaze, PicoBlaze, PLUSASM, PowerGuide, PowerMaze, QPro, Real-PCI, Rocket I/O, SelectI/O, SelectRAM, SelectRAM+, Silicon Xpresso, Smartguide, Smart-IP, SmartSearch, SMARTswitch, System ACE, Testbench In A Minute, TrueMap, UIM, VectorMaze, VersaBlock, VersaRing, Virtex-II Pro, Virtex-II EasyPath, Wave Table, WebFITTER, WebPACK, WebPOWERED, XABEL, XACT-Floorplanner, XACT-Performance, XACTstep Advanced, XACTstep Foundry, XAM, XAPP, X-BLOX +, XC designated products, XChecker, XDM, XEPLD, Xilinx Foundation Series, Xilinx XDTV, Xinfo, XSI, XtremeDSP and ZERO+ are trademarks of Xilinx, Inc.

The Programmable Logic Company is a service mark of Xilinx, Inc.

All other trademarks are the property of their respective owners.

Xilinx, Inc. does not assume any liability arising out of the application or use of any product described or shown herein; nor does it convey any license under its patents, copyrights, or maskwork rights or any rights of others. Xilinx, Inc. reserves the right to make changes, at any time, in order to improve reliability, function or design and to supply the best product possible. Xilinx, Inc. will not assume responsibility for the use of any circuitry described herein other than circuitry entirely embodied in its products. Xilinx provides any design, code, or information shown or described herein "as is." By providing the design, code, or information as one possible implementation of a feature, application, or standard, Xilinx makes no representation that such implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of any such implementation, including but not limited to any warranties or representations that the implementation is free from claims of infringement, as well as any implied warranties of merchantability or fitness for a particular purpose. Xilinx, Inc. devices and products are protected under U.S. Patents. Other U.S. and foreign patents pending. Xilinx, Inc. does not represent that devices shown or products described herein are free from patent infringement or from any other third party right. Xilinx, Inc. assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. Xilinx, Inc. will not assume any liability for the accuracy or correctness of any engineering or software support or assistance provided to a user.

Xilinx products are not intended for use in life support appliances, devices, or systems. Use of a Xilinx product in such applications without the written consent of the appropriate Xilinx officer is prohibited.

The contents of this manual are owned and copyrighted by Xilinx. Copyright 1994-2005 Xilinx, Inc. All Rights Reserved. Except as stated herein, none of the material may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of any material contained in this manual may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

PowerPC Instruction Set Extension Guide EDK, Document Revision 2.0, April 28, 2005

The following table shows the revision history for this document.

| | Version | Revision |
|----------|----------------|--|
| 11/24/04 | 1.0 | Initial release. |
| 04/28/05 | 2.0 | Add UDI and FCM load store instructions. |

Table of Contents

Preface: About This Document

| | |
|----------------------------|---|
| Document Conventions | 7 |
|----------------------------|---|

Chapter 1: APU Instruction Set Extension

| | |
|---------------------------|----|
| Instruction Listing | 9 |
| get | 10 |
| put | 11 |
| udi<n>fcm udi<n>fcm | 12 |
| lbfcmux | 13 |
| lhfcmux | 14 |
| lwfcmux | 15 |
| ldfcmux | 16 |
| lqfcmux | 17 |
| lbfcmx | 18 |
| lhfcmx | 19 |
| lwfcmx | 20 |
| ldfcmx | 21 |
| lqfcmx | 22 |
| stbfcmux | 23 |
| sthfcmux | 24 |
| stwfcmux | 25 |
| stdfcmux | 26 |
| stqfcmux | 27 |
| stbfcmx | 28 |
| sthfcmx | 29 |
| stwfcmx | 30 |
| stdfcmx | 31 |
| stqfcmx | 32 |

About This Document

This document is an addendum to the Instruction Set Architecture (ISA) chapter in the *PowerPC™ Processor Reference Guide* [UG011].

The description herein is focused on the Embedded Development Kit (EDK) extension of the ISA through the Auxiliary Processor Unit (APU) of the PowerPC 405 in Virtex™-4.

The PowerPC relies on custom processing logic implemented in the FPGAS fabric, called Fabric Co-processing Modules (FCM) to execute these instructions. An FCM can be either a user IP or a Xilinx IP. Please refer to the following documents for more information on the use and design of co-processors for the PowerPC APU interface:

- *PowerPC 405 Block Reference Guide* [UG018]
- PPC405_Virtex4 [DS306]
- Fabric Co-processor Bus [DS308]
- FCB2FSL_Bridge [DS309]

Document Conventions

This document follows the same standards as the *PowerPC Processor Reference Guide*.

APU Instruction Set Extension

Instruction Listing

The following pages list the Auxiliary Processor Unit instruction set extension supported by the PPC405.

The notation “**FCM5**” in this document indicates a five bit immediate value. The interpretation of the value is left to the Fabric Coprocessor Module (FCM). Typically this would be the register value on the FCM.

“**FSL**” in the document corresponds to Fast Simple Link. This bus protocol is defined in the *Processor IP Reference Guide*.

Rest of the notations in the document are the same as those defined in the “Document Convention” section of the *PowerPC Processor Reference Guide*.

get

Get from FSL Co-processor Via APU Controller

| | | |
|--------------|----------|---------------------------------------|
| get | rD, FSLx | get data from FSL x (blocking) |
| nget | rD, FSLx | get data from FSL x (non-blocking) |
| cget | rD, FSLx | get control from FSL x (blocking) |
| ncget | rD, FSLx | get control from FSL x (non-blocking) |

X Instruction Form

| | | | | | | | | |
|---|----|-----------|--------|--------|--------|--------|--------|--------|
| 4 | rD | 0 0 0 0 0 | FSL | 4 | n | c | 12 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 2 5 | 2 6 | 2 7 | 3 1 |

Description

This is a privileged instruction.

The processor will read from the FSLx interface and place the result in register rD.

The **get** instruction has four variants.

- The blocking versions (n=0) will stall processor until data is available.
- The non-blocking versions will return even if data is not available. XER[CA] is set to '0' if the data is valid and to '1' if the data was invalid. In case of an invalid access the destination register contents is undefined.
- The **get** and **nget** instructions expect the control bit from the FSL interface to be '0'. If this is not the case, the instruction will set XER[OV] to '1'.
- The **cget** and **ncget** instructions expect the control bit from the FSL interface to be '1'. If this is not the case, the instruction will set XER[OV] to '1'.

Note: The term "blocking" does not correspond to APU blocking. The explanation for FSL blocking instructions is available in the *Processor IP Reference Guide*.

Pseudocode

```
(rD) ← APU(FSL).data
if (n = 1) then
    XER[CA] ← not APU(FSL).exists
if (APU(FSL).control = c) then
    XER[OV] ← 0
else
    XER[OV] ← 1
```

Registers Altered

- rD.
- XER[OV].
- XER[CA] if n=1.

Exceptions

None.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction, that uses the PowerPC™ Auxiliary Processor Unit (APU) controller.

put

Put to FSL Co-processor Via APU Controller

| | | |
|--------------|----------|-------------------------------------|
| put | rA, FSLx | put data to FSL x (blocking) |
| nput | rA, FSLx | put data to FSL x (non-blocking) |
| cput | rA, FSLx | put control to FSL x (blocking) |
| ncput | rA, FSLx | put control to FSL x (non-blocking) |

X Instruction Form

| | | | | | | | | |
|---|-----------|--------|--------|--------|--------|--------|--------|--------|
| 4 | 0 0 0 0 0 | rA | FSL | 5 | n | c | 12 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 2 5 | 2 6 | 2 7 | 3 1 |

Description

This is a privileged instruction.

The processor will write the contents of register rA to the FSLx interface.

The **put** instruction has four variants.

- The blocking versions (n=0) will stall the processor until there is space available on the FSL interface.
- The non-blocking versions will return even if there is no space. XER[CA] is set to '0' if space was available and to '1' if no space was available.
- The **put** and **nput** instructions will set the control bit to the FSL interface to '0'.
- The **cput** and **ncput** instruction will set the control bit to '1'.

Note: The term "blocking" does not correspond to APU blocking. The explanation for FSL blocking instructions is available in the *Processor IP Reference Guide*.

Pseudocode

```

APU(FSL).data ← (rA)
APU(FSL).control ← c
if (n = 1) then
    XER[CA] ← APU(FSL).full

```

Registers Altered

XER[CA] if n = 1.

Exceptions

None.

Compatibility

This instruction is defined by Xilinx as a pre defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

udi<n>fcm
udi<n>fcm.

UDI (User Defined Instructions)

| | | |
|-------------------------|----------------|---|
| udi<n>fcm | T, A, B | User defined instructions, not modifying the condition code register. |
| udi<n>fcm. | T, A, B | User defined instructions, modifying condition code |

X Instruction Form

| 4 | T | A | B | 2 | n | 3 | Rcn |
|---|---|--------|--------|--------|--------|--------|-----|
| 0 | 6 | 1 1 | 1 6 | 2 1 | 2 3 | 2 6 | 31 |

Description

The exact operation done by the instruction is determined by the user based on the Fabric Co-processor Module. For more information about user defined instructions, please refer to Chapter 4 (“PowerPC 405 APU Controller”) of the *PowerPC 405 Processor Block Reference Guide*.

Xilinx GNU assembler recognizes 16 User Defined Instructions (UDIs)

- Eight instructions, which will modify the condition code (**Rcn = 0**)
- Eight instructions, which will not modify the condition code (**Rcn = 1**)

The conditional code register (**Rcn**) to be modified is determined by the definition of the instruction as provided by the user.

Special Notations for this instruction

- T : PowerPC GPR (rD)/ FCM Register (FCR)
- A : PowerPC GPR (rD)/ FCM Register (FCR)/ 5 bit Immediate
- B : PowerPC GPR (rD)/ FCM Register (FCR)/ 5 bit Immediate

Five bits each area available for the special notations defined above.

Pseudocode

Dependent on the user operation.

Registers Altered

Dependent on the user operation.

Exceptions

Dependent on the user operation.

Compatibility

This instruction is defined by Xilinx as a user defined instruction (UDI) that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lbfcmux

Load Byte with Update Indexed (Fabric Co-processor Module)

lbfcmux **FCM5, rA, rB**

X Instruction Form

| 31 | FCM5 | rA | rB | 519 | 0 |
|----|------|----|----|-----|---|
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address.

The byte referenced by EA is sign-extended to 32 bits and loaded into register **FCM5**. The EA is loaded into **rA**.

Pseudocode

```

EA          ← (rA) + (rB)
eb          ← 8 * EA28:31
APU(FCM).data      ← undefined
APU(FCM).dataeb:eb+7 ← MEM(EA, 1)
(FCM5)        ← custom_op(APU(FCM).data, (FCM5))
(rA)          ← EA

```

Registers Altered

- **rA**.
- Register inferred by **FCM5**.

Exceptions

- **Data storage:** this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- **Data TLB miss:** this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- **Alignment:** this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- **rA=0**.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lhcmux

Load Half Word with Update Indexed (Fabric Co-processor Module)

lhcmux FCM5, rA, rB

X Instruction Form

| | | | | | |
|----|------|--------|--------|--------|--------|
| 31 | FCM5 | rA | rB | 551 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 3 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address.

The half word referenced by EA is sign-extended to 32 bits and loaded into register **FCM5**. The EA is loaded into **rA**.

Pseudocode

```

EA          ← ((rA) + (rB)) & (~1)
eb          ← 8 * EA28:31
APU(FCM).data      ← undefined
APU(FCM).dataeb:eb+15 ← MEM(EA, 2)
(FCM5)         ← custom_op(APU(FCM).data, (FCM5))
(rA)           ← EA
    
```

Registers Altered

- **rA**.
- Register inferred by **FCM5**.

Exceptions

- **Data storage:** this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- **Data TLB miss:** this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- **Alignment:** this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- **rA=0**.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lwfcmux

Load Word With Update Indexed (Fabric Co-processor Module)

lwfcmux **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|--------|--------|--------|--------|
| 31 | FCM5 | rA | rB | 583 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 3 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address.

The word referenced by EA is loaded into register **FCM5**. The EA is loaded into **rA**.

Pseudocode

```
EA          ← ((rA) + (rB)) & (~3)
eb         ← 8 * EA28:31
APU(FCM).data
(FCM5)     ← custom_op(APU(FCM).data, (FCM5))
(rA)      ← EA
```

Registers Altered

- **rA**.
- Register inferred by **FCM5**.

Exceptions

- **Data storage:** this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- **Data TLB miss:** this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- **Alignment:** this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- **rA=0**.

Compatibility

- This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

ldfcmux

Load Double with Update Indexed (Fabric Co-processor Module)

ldfcmux **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|--------|--------|--------|--------|
| 31 | FCM5 | rA | rB | 775 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 3 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address.

Two words referenced by EA and EA +4 are loaded into register **FCM5**. The EA is loaded into **rA**.

Pseudocode

```
EA          ← ((rA) + (rB)) & (~3)
eb         ← 8 * EA28:31
APU(FCM).data (FCM5) ← MEM(EA, 4)
APU(FCM).data (FCM5) ← MEM(EA+4, 4)
(rA)       ← EA
```

Registers Altered

- **rA**.
- Register inferred by **FCM5**.
 - ◆ The instruction assumes that the register FCM5 is 64-bit.

Exceptions

- **Data storage:** this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- **Data TLB miss:** this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- **Alignment:** this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- **rA=0.**

Compatibility

- This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lqfcmux

Load Quad with Update Indexed (Fabric Co-processor Module)

lqfcmux **FCM5, rA, rB**

X Instruction Form

| 31 | FCM5 | rA | rB | 615 | 0 |
|----|------|----|----|-----|---|
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address.

Four words referenced by EA through EA + 12 are loaded into register **FCM5**. The EA is loaded into **rA**.

Pseudocode

```

EA          ← ((rA) + (rB)) & (~3)
eb         ← 8 * EA28:31
APU(FCM).data
(FCM5)     ← custom_op(APU(FCM).data, (FCM5))
APU(FCM).data
(FCM5)     ← MEM(EA+4, 4)
APU(FCM).data
(FCM5)     ← custom_op(APU(FCM).data, (FCM5))
APU(FCM).data
(FCM5)     ← MEM(EA+8, 4)
APU(FCM).data
(FCM5)     ← custom_op(APU(FCM).data, (FCM5))
APU(FCM).data
(FCM5)     ← MEM(EA+12, 4)
(FCM5)     ← custom_op(APU(FCM).data, (FCM5))
(rA)       ← EA

```

Registers Altered

- **rA**.
- Register inferred by **FCM5**.
 - ♦ The instruction assumes that the register **FCM5** is 128-bit.

Exceptions

- **Data storage**: this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- **Data TLB miss**: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- **Alignment**: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- **rA=0**.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lbfcmx

Load Byte Indexed (Fabric Co-processor Module)

lbfcmx **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|----|----|---|---|
| 31 | FCM5 | rA | rB | 7 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The byte referenced by EA is sign-extended to 32 bits and loaded into register **FCM5**.

Pseudocode

| | |
|-----------------------------------|--|
| EA | $\leftarrow (0 \mid \mathbf{rA}) + (\mathbf{rB})$ |
| eb | $\leftarrow 8 * \mathbf{EA}_{28:31}$ |
| APU(FCM5).data | $\leftarrow \text{undefined}$ |
| APU(FCM5).data _{eb:eb+7} | $\leftarrow \text{MEM}(\mathbf{EA}, 1)$ |
| (FCM5) | $\leftarrow \text{custom_op}(\text{APU}(\text{FCM5}).\text{data}, (\mathbf{FCM5}))$ |

Registers Altered

- Register inferred by **FCM5**.

Exceptions

- Data storage: this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lhcmx

Load Half Word Indexed (Fabric Co-processor Module)

lhcmx FCM5, rA, rB

X Instruction Form

| 31 | FCM5 | rA | rB | 39 | 0 |
|----|------|----|----|----|---|
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The half word referenced by EA is sign extended to 32 bits and loaded into register **FCM5**.

Pseudocode

```
EA          ← ((0 | rA) + (rB)) & (~1)
eb         ← 8 * EA28:31
APU(FCM).data      ← undefined
APU(FCM).dataeb:eb+15 ← MEM(EA, 2)
(FCM5)      ← custom_op(APU(FCM).data, (FCM5))
```

Registers Altered

- Register inferred by **FCM5**.

Exceptions

- Data storage: this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lwfcmx

Load Word Indexed (Fabric Co-processor Module)

lwfcmx **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|----|----|----|---|
| 31 | FCM5 | rA | rB | 71 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The word referenced by EA is loaded into register **FCM5**.

Pseudocode

```
EA          ← ((0 | rA) + (rB)) & (~3)
eb         ← 8 * EA28:31
APU(FCM).data (FCM5) ← MEM(EA, 4)
                    ← custom_op(APU(FCM).data, (FCM5))
```

Registers Altered

- Register inferred by **FCM5**.

Exceptions

- Data storage: this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

ldfcmx

Load Double Indexed (Fabric Co-processor Module)

ldfcmx **FCM5, rA, rB**

X Instruction Form

| 31 | FCM5 | rA | rB | 263 | 0 |
|----|------|----|----|-----|---|
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

Two words referenced by EA and EA + 4 are loaded into register(s) inferred by **FCM5**.

Pseudocode

```
EA          ← ((0 | rA) + (rB)) & (~3)
eb         ← 8 * EA28:31
APU(FCM).data (FCM5) ← MEM(EA, 4)
APU(FCM).data (FCM5) ← MEM(EA+4, 4)
APU(FCM).data (FCM5) ← custom_op(APU(FCM).data, (FCM5))
```

Registers Altered

- Register inferred by **FCM5**.
 - ♦ The exact implementation of the register inferred is to be determined by the user.

Exceptions

- **Data storage:** this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- **Data TLB miss:** this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- **Alignment:** this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

lqfcmx

Load Quad Indexed (Fabric Co-processor Module)

lqfcmx **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 103 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

Four words referenced by EA through EA + 12 are loaded into register(s) inferred by **FCM5**.

Pseudocode

```
EA      ← ((0 | rA) + (rB)) & (~3)
eb      ← 8 * EA28:31
APU(FCM).data
(FCM5)  ← custom_op(APU(FCM).data, (FCM5))
APU(FCM).data
(FCM5)  ← MEM(EA+4, 4)
APU(FCM).data
(FCM5)  ← custom_op(APU(FCM).data, (FCM5))
APU(FCM).data
(FCM5)  ← MEM(EA+8, 4)
APU(FCM).data
(FCM5)  ← custom_op(APU(FCM).data, (FCM5))
APU(FCM).data
(FCM5)  ← MEM(EA+12, 4)
APU(FCM).data
(FCM5)  ← custom_op(APU(FCM).data, (FCM5))
```

Registers Altered

- Register inferred by **FCM5**.
 - ♦ The exact implementation of the register inferred is to be determined by the user.

Exceptions

- Data storage: this exception is raised if the access is prevented by no-access-allowed zone protection. This only applies to accesses in user mode when data relocation is enabled.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stbfcmux

Store Byte with Update Indexed (Fabric Co-processor Module)

stbfcmux FCM5, rA, rB

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 647 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The least-significant byte of register inferred by **FCM5** is stored into the byte referenced by EA. The EA is loaded into the register **rA**.

Pseudocode

| | |
|---------------|---|
| EA | $\leftarrow (\mathbf{rA}) + (\mathbf{rB})$ |
| eb | $\leftarrow 8 * \mathbf{EA}_{28:31}$ |
| APU(FCM).data | $\leftarrow \mathbf{custom_op}(\mathbf{FCM5})$ |
| MEM(EA, 1) | $\leftarrow \mathbf{APU}(\mathbf{FCM}).\mathbf{data}_{\mathbf{eb}:\mathbf{eb}+7}$ |
| (rA) | $\leftarrow \mathbf{EA}$ |

Registers Altered

- rA

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- $\mathbf{rA} = 0$

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

sthfcmux

Store Half Word With Update Indexed (Fabric Co-processor Module)

sthfcmux **FCM5**, rA, rB

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 679 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register rB are used as the index.
- The contents of register rA are used as the base address. If rA is 0, then 0 is used as the base address.

Two least-significant bytes of register inferred by **FCM5** are stored into the addressed referenced by EA. The EA is loaded into the register rA.

Pseudocode

```
EA          ← ((rA) + (rB)) & (~1)
eb          ← 8 * EA28:31
APU(FCM).data ← custom_op(FCM5)
MEM(EA, 2)  ← APU(FCM).dataeb:eb+15
(rA)       ← EA
```

Registers Altered

- rA

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- rA = 0

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stwfcmux

Store Word With Update Indexed (Fabric Co-processor Module)

stwfcmux FCM5, rA, rB

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 711 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The contents of the register inferred by **FCM5** is stored into the address referenced by EA. The EA is loaded into the register **rA**.

Pseudocode

```
EA          ← ((rA) + (rB)) & (~3)
eb         ← 8 * EA28:31
APU(FCM).data ← custom_op(FCM5)
MEM(EA, 4) ← APU(FCM).data
(rA)       ← EA
```

Registers Altered

- **rA**

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- **rA** = 0

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stdfcmux

Store Double With Update Indexed (Fabric Co-processor Module)

stdfcmux **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 903 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The contents of the register inferred by **FCM5** is stored into the address referenced by **EA** and **EA + 4**. The source register is assumed to be 64-bit. The **EA** is loaded into the register **rA**.

Pseudocode

```
EA      ← ((rA) + (rB)) & (~3)
eb      ← 8 * EA28:31
APU(FCM).data ← custom_op(FCM5)
MEM(EA, 4) ← APU(FCM).data
APU(FCM).data ← custom_op(FCM5)
MEM(EA+4, 4) ← APU(FCM).data
(rA)    ← EA
```

Registers Altered

- **rA**

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the **EA** is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- **rA** = 0

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stqfcmux

Store Quad With Update Indexed (Fabric Co-processor Module)

stqfcmux FCM5, rA, rB

X Instruction Form

| | | | | | |
|----|------|--------|--------|--------|--------|
| 31 | FCM5 | rA | rB | 743 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 3 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register rB are used as the index.
- The contents of register rA are used as the base address. If rA is 0, then 0 is used as the base address.

The contents of the register inferred by FCM5 is stored into the address referenced by EA through EA+12. The source register is assumed to be 128-bit. The EA is loaded into the register rA.

Pseudocode

```
EA                                    ← ((rA) + (rB)) & (~3)
eb                                    ← 8 * EA28:31
APU(FCM) .data                      ← custom_op(FCM5)
MEM(EA, 4)                            ← APU(FCM) .data
APU(FCM) .data                      ← custom_op(FCM5)
MEM(EA+4, 4)                         ← APU(FCM) .data
APU(FCM) .data                      ← custom_op(FCM5)
MEM(EA+8, 4)                         ← APU(FCM) .data
APU(FCM) .data                      ← custom_op(FCM5)
MEM(EA+12, 4)                        ← APU(FCM) .data
(rA)                                  ← EA
```

Registers Altered

- rA

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.
- rA = 0

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stbfcmx

Store Byte Indexed (Fabric Co-processor Module)

stbfcmx **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 135 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The least-significant byte of register inferred by **FCM5** is stored into the byte referenced by EA.

Pseudocode

```
EA      ← (0 | rA) + (rB)
eb      ← 8 * EA28:31
APU(FCM).data ← custom_op(FCM5)
MEM(EA, 1) ← APU(FCM).dataeb:eb+7
```

Registers Altered

- None

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

sthfcmx

Store Half Word Indexed (Fabric Co-processor Module)

sthfcmx FCM5, rA, rB

X Instruction Form

| | | | | | |
|----|------|--------|--------|--------|--------|
| 31 | FCM5 | rA | rB | 167 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 3 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The two least-significant bytes of register inferred by **FCM5** is stored into the address referenced by EA.

Pseudocode

```
EA                                    ← ((0 | rA) + (rB)) & (~1)
eb                                    ← 8 * EA28:31
APU(FCM).data                       ← custom_op(FCM5)
MEM(EA, 2)                           ← APU(FCM).dataeb:eb+15
```

Registers Altered

- None

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stwfcmx

Store Word Indexed (Fabric Co-processor Module)

stwfcmx FCM5, rA, rB

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 199 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register **rB** are used as the index.
- The contents of register **rA** are used as the base address. If **rA** is 0, then 0 is used as the base address.

The contents of the register inferred by **FCM5** is stored into the address referenced by **EA**.

Pseudocode

```
EA                                    ← ((0 | rA) + (rB)) & (~3)
eb                                    ← 8 * EA28:31
APU(FCM).data                      ← custom_op(FCM5)
MEM(EA, 4)                           ← APU(FCM).data
```

Registers Altered

- None

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stdfcmx

Store Double Indexed (Fabric Co-processor Module)

stdfcmx **FCM5**, rA, rB

X Instruction Form

| | | | | | |
|----|------|----|----|-----|---|
| 31 | FCM5 | rA | rB | 391 | 0 |
| 0 | 6 | 1 | 1 | 2 | 3 |
| | | 1 | 6 | 1 | 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register rB are used as the index.
- The contents of register rA are used as the base address. If rA is 0, then 0 is used as the base address.

The contents of register inferred by **FCM5** is stored into the address referenced by EA. The source register is expected to be 64-bit.

Pseudocode

```
EA      ← ((0 | rA) + (rB)) & (~3)
eb      ← 8 * EA28:31
APU(FCM).data ← custom_op(FCM5)
MEM(EA, 4)  ← APU(FCM).data
APU(FCM).data ← custom_op(FCM5)
MEM(EA+4, 4) ← APU(FCM).data
```

Registers Altered

- None

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.

stqfcmx

Store Quad Indexed (Fabric Co-processor Module)

stqfcmx **FCM5, rA, rB**

X Instruction Form

| | | | | | |
|----|------|--------|--------|--------|--------|
| 31 | FCM5 | rA | rB | 231 | 0 |
| 0 | 6 | 1 1 | 1 6 | 2 1 | 3 1 |

Description

An effective address (EA) is calculated by adding an index to a base address, which is formed as follows:

- The contents of register rB are used as the index.
- The contents of register rA are used as the base address. If rA is 0, then 0 is used as the base address.

The contents of register inferred by **FCM5** is stored into the address referenced by EA. The source register is expected to be 128-bit.

Pseudocode

```
EA      ← ((0 | rA) + (rB)) & (~3)
eb      ← 8 * EA28:31
APU(FCM).data ← custom_op(FCM5)
MEM(EA, 4)   ← APU(FCM).data
APU(FCM).data ← custom_op(FCM5)
MEM(EA+4, 4) ← APU(FCM).data
APU(FCM).data ← custom_op(FCM5)
MEM(EA+8, 4) ← APU(FCM).data
APU(FCM).data ← custom_op(FCM5)
MEM(EA+12, 4) ← APU(FCM).data
```

Registers Altered

- None

Exceptions

- Data storage: this exception is raised if the access is prevented by zone protection when data relocation is enabled.
 - ♦ No-access-allowed zone protection applies only to accesses in user mode.
 - ♦ Read-only zone protection applies to user and privileged modes.
- Data TLB miss: this exception is raised if data relocation is enabled and a valid translation-entry corresponding to the EA is not found in the TLB.
- Alignment: this exception is raised if the access is to a misaligned address or for any access with an Endian attribute not supported in hardware.

Execution of any of the following invalid-instruction forms results in a boundedly undefined result rather than a program exception:

- Reserved bits containing a non-zero value.

Compatibility

This instruction is defined by Xilinx as a pre-defined instruction that uses the PowerPC Auxiliary Processor Unit (APU) controller.