PowerPC Instruction Set Extension Guide

ISA Support for the PowerPC APU Controller in Virtex-4

EDK 6.3 November 24, 2004
The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/24/04</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
Table of Contents

About This Document

Document Conventions................................................................................................................. 7
General Conventions ...................................................................................................................... 7

Chapter 1: APU Instruction Set Extension

Alphabetical Instruction Listing ......................................................................................................... 9
Preface

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

Document Conventions

General Conventions

This document follows the same standards as the PowerPC Processor Reference Guide.
Chapter 1

APU Instruction Set Extension

Alphabetical Instruction Listing

The following pages list the Auxiliary Processor Unit instruction set extension supported by the PPC405 in alphabetical order.
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)
ncget rD, FSLx  get control from FSL x (blocking)
ncget rD, FSLx  get control from FSL x (non-blocking)

X Instruction Form

<table>
<thead>
<tr>
<th>4</th>
<th>rD</th>
<th>0 0 0 0 0 0</th>
<th>FSL</th>
<th>8</th>
<th>n</th>
<th>c</th>
<th>12</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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’.

Pseudocode

\[
(rD) \leftarrow \text{APU}(FSL).\text{data} \\
\text{if } (n = 1) \text{ then} \\
\quad \text{XER}[CA] \leftarrow \neg \text{APU}(FSL).\text{exists} \\
\text{if } (\text{APU}(FSL).\text{control} == c) \text{ then} \\
\quad \text{XER}[OV] \leftarrow 0 \\
\text{else} \\
\quad \text{XER}[OV] \leftarrow 1
\]

Registers Altered

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

Exceptions

- None.

Compatibility

This instruction is defined by Xilinx as a User Defined Instruction (UDI) that uses the PowerPC Auxiliary Processor Unit (APU) controller.
**put**

Put to FSL co-processor via APU controller

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Pseudocode</th>
</tr>
</thead>
<tbody>
<tr>
<td>put rA, FSLx</td>
<td>put data to FSL x (blocking)</td>
<td>APU(FSL).data ← (rA)</td>
</tr>
<tr>
<td>nput rA, FSLx</td>
<td>put data to FSL x (non-blocking)</td>
<td>APU(FSL).data ← (rA)</td>
</tr>
<tr>
<td>cput rA, FSLx</td>
<td>put control to FSL x (blocking)</td>
<td>APU(FSL).control ← c</td>
</tr>
<tr>
<td>ncput rA, FSLx</td>
<td>put control to FSL x (non-blocking)</td>
<td>APU(FSL).control ← c</td>
</tr>
</tbody>
</table>

**X Instruction Form**

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>rA</th>
<th>FSL</th>
<th>5</th>
<th>n</th>
<th>c</th>
<th>12</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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’.

**Pseudocode**

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

**Registers Altered**

- XER[CA].

**Exceptions**

- None.

**Compatibility**

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