# POWER7+™ Processor Programming Model Bulletin

January 25, 2013

# POWER7+ Processor Programming Model Bulletin

## Background

The Power ISA allows an implementation to be fully compliant with an architecture version while implementing a subset of the next version of the architecture. The POWER7+ Processor is compliant with Version 2.06B of the Power ISA. The features described herein are in addition to what is provided by Version 2.06B and will become part of the architecture in Version 2.07.

## Summary

The POWER7+ Processor provides two extensions to the functionality specified in the Power ISA Version 2.06B. The first change is to make the “very low” thread priority that has been available to privileged software also available to non-privileged software. This may be used in spin loops and similar situations in which the little or no useful work is being done. The second change is to enable software to specify the initial aggressiveness with which a new hardware-detected data stream is prefetched. Programs characterized by short, well-predicted data streams often benefit from the first several prefetches being issued quickly, as contrasted with those characterized by longer, less well-predicted data streams.

# “Very Low” Thread Priority

This change enables problem state software to minimize consumption of execution resources when not doing useful work. The description below is an excerpt from v2.06B Book II, Chapter 3 with irrelevant text removed and new text marked with change bars.

## Program Priority Registers

The Program Priority Register (PPR) is a 64-bit register that controls the program’s priority. The PPR provides access to the full 64-bit PPR, and the Program Priority Register 32-bit (PPR32) provides access to the upper 32 bits of the PPR. The Embedded environment only provides access to PPR32. The layouts of the PPR and PPR32 are shown in Figure 1.

**PPR [Category: Server]:**

| | | | | | |
|---|---|---|---|---|---|
| /// | PRI | /// | ??? | /// | ??? |
| 0 | 11 | 14 | 26 | 32 | 44 63 |

**PPR32 [Category: Phased-In]**

| | | | |
|---|---|---|---|
| /// | PRI | /// | ??? |
| 32 | 43 | 46 | 58 63 |

| Bit(s) | Description |
|---|---|
| 11:13 | ***Program Priority (PRI)*** ***($PPR32_{43:45}$)*** <br> 001 very low <br> 010 low <br> 011 medium low (normal) <br> 100 medium <br> If other values are written to this field, the PRI field is not changed. (See Section 4.3.4 of Book III-S for additional information.) |
| 26:31 | Implementation-specific ($PPR32_{58:63}$) |
| 44:63 | implementation-specific |

All other fields are reserved.

**Figure 1. Program Priority Register**

Programs can always set the PRI field to very low, low, medium low, and medium priorities.If other values are written to this field, the PRI field is not changed.

**Programming Note**

The ability to access the low-order half of the PPR (and thus the use of ***mfppr*** and ***mtppr***) might be phased out in a future version of the architecture.

**Programming Note**

By setting the PRI field, a programmer may be able to improve system throughput by causing system resources to be used more efficiently.

E.g., if a program is waiting on a lock (see Section B.2), it could set very low priority, with the result that more processor resources would be diverted to the program that holds the lock. This diversion of resources may enable the lock-holding program to complete the operation under the lock more quickly, and then relinquish the lock to the waiting program.

**Programming Note**

When the system error handler is invoked, the PRI field may be set to an undefined value.

## “or” Instruction Setting the PPR

The *or Rx,Rx,Rx* (see Book I) instruction can be used to set $PPR_{PRI}$ as shown in Figure 2. *or. Rx,Rx,Rx* does not set $PPR_{PRI}$.

| Rx | $PPR_{PRI}$ | Priority |
|---|---|---|
| 31 | 001 | very low |
| 1 | 010 | low |
| 6 | 011 | medium low (normal) |
| 2 | 100 | medium |

**Figure 2. Priority levels for *or Rx,Rx,Rx***

**Programming Note**

**Warning:** Other forms of *or Rx,Rx,Rx* that are not described in this section may also cause program priority to change. Use of these forms should be avoided except when software explicitly intends to alter program priority. If a no-op is needed, the preferred no-op (*ori 0,0,0*) should be used.

## Data Stream Ramp

In an academic dissection of prefetching, one would observe that the depth of the prefetching and the speed with which the mechanism arrives at the desired depth ("speed of the ramp") are orthogonal characteristics. This change enables software to influence the speed of the ramp for hardware-detected data streams. The material below is an excerpt from v2.06B Book II, Section 4.2 with new text marked with change bars.

### Data Stream Control Register (DSCR)

The layout of the Data Stream Control Register (DSCR) is shown in Figure 3 below. .

| | | | | | |
|---|---|---|---|---|---|
| // | URG | LSD | SNSE | SSE | DPFD |
| 0 54 | 55 57 | 58 | 59 | 60 | 61 63 |

**Figure 3. Data Stream Control Register**

**Bit(s) Description**

0:54 Reserved

55:57 ***Depth Attainment Urgency*** (URG)

This field indicates how quickly the prefetch depth should be reached for hardware-detected streams. Values and their meanings are as follows.

0 default
1 not urgent
2 least urgent
3 less urgent
4 medium
5 urgent
6 more urgent
7 most urgent

58 ***Load Stream Disable*** (LSD)

0 No effect.
1 Disables hardware detection and initiation of load streams.

59 ***Stride-N Stream Enable*** (SNSE)

0 No effect.
1 Enables the hardware detection and initiation of load and store streams that have a stride greater than a single cache block. Such load streams are detected only when LSD is also zero. Such store streams are detected only when SSE is also one.

61:63 ***Default Prefetch Depth*** (DPFD)

This field supplies a prefetch depth for hardware-detected streams and for software-defined streams for which a depth of zero is specified or for which ***dcbt/dcbtst*** with TH=1010 is *not* used in their description. Values and their meanings are as follows.

0 default (LPCR$_{DPFD}$)
1 none
2 shallowest
3 shallow
4 medium
5 deep
6 deeper
7 deepest

The contents of the DSCR affect how a processor handles hardware-detected and software-defined data streams.

A move to the DSCR causes all active and nascent data streams to cease to exist.

Access to this SPR is privileged.

**Programming Note**

The URG, LSD, SNSE and SSE fields do not affect the initiation of streams specified using the ***dcbt*** and ***dcbtst*** instructions.

Note that even when SNSE is not set, hardware may detect Stride-N streams in intervals when they access elements that map to sequential cache blocks.

**Programming Note**

The purpose of Depth Attainment Urgency is to regulate the rate of prefetch generation from the cycle at which the hardware first detects an incipient stream until the cycle when the prefetch Depth is reached. A more urgent setting will benefit applications that are dominated by short to medium length streams, because otherwise prefetching does not occur rapidly enough to benefit them. In contrast, applications that frequently cause unproductive prefetches due to stream mispredicts will benefit from a less urgent setting.

Unlike the Depth, the Depth Attainment Urgency applies only to hardware-detected streams. Furthermore, the DSCR provides the only point of control for this parameter. Software-defined streams are assumed not to have the correctness risk associated with hardware streams, and therefore are set to reach their depth relatively quickly.

**Programming Note**

The contents of the DSCR are intended to be managed by application programs. Access to the DSCR is privileged because, when the DSCR was added to the architecture, adding it as non-privileged would have been incompatible with the application binary interface (ABI) of some operating systems. Operating systems will provide a service that allows application programs to manage the contents of the DSCR.

**Programming Note**

The latency-reducing actions taken in response to a program's hints about access to a data stream, including the depth and urgency parameters, may vary based on its behavior and on the behavior of other programs sharing platform resources, as well as on the design of the platform resources they use. Without actually changing the stream specification or DSCR parameters, the processor may adjust its actions (e.g. slow down prefetches or be more selective choosing them) based on their effectiveness and on the availability of storage bandwidth. In general, the goal of this variation is to improve overall system performance and fairness across the set of programs that share resources. There often will be a performance benefit, however, from adjusting stream specifications to the platform and co-resident programs to adjust for these actions by the processor.