Synchronizing Instructions for PowerPC Instruction Set Architecture: Sync vs Isync

The PowerPC Instruction Set provides two types of synchronizing instructions. These are context synchronizing and execution synchronizing instructions. All of the information on this page may be found in more detail in ***PowerPC Microprocessor Family: The Programming Environments for 32-Bit Processors***.

Context Synchronizing Instructions

Context synchronizing instructions for PowerPC include ***isync***, ***sc***, and ***rfi***. In addition, exceptions cause context synchronization. These instructions are used to make sure that the effects of all previously issued instructions are in place before a context switch, and that the context switch takes effect for instructions after the switch. Context synchronization needs to be performed when changing the values in certain fields of certain processor registers, as listed in the table found at the bottom of this page.

The context synchronizing instructions ensure several things:

- Instruction dispatching is halted. The ***sc*** instruction also ensures that no higher priority exception exists.
- All previously issued instructions have completed, at least to a point where they are no longer able to cause an exception. However, memory accesses caused by these instructions need not have completed with respect to other processors and mechanisms. If memory accesses need to be complete, the ***sync*** instruction can be used to ensure this.
- Previously issued instructions will complete in the context in which they were issued (privilege, protection, address translation).
- Instructions issued after the synchronizing instruction will execute in the new context.

In order to make sure context changes take place for instructions following the synchronization, the instruction queue is flushed and all these instructions are refetched with the new context in place.

All context synchronizing instructions are execution synchronizing.

Execution Synchronizing Instructions

While context synchronizing instructions are most useful for ensuring that context switches take effect at the right time, execution synchronizing instructions are useful for providing a sequencing function in a program and for memory access synchronization on multiple processor implementations. These instructions can be used to be sure that previous instructions appear to have completed before subsequent instructions are executed without forcing a flush of prefetched instructions in the instruction queue. This feature is often useful for specifically ordering programs and for debug. Programmers should be aware, however, that the ***sync*** instruction normally takes a significant amount of time to complete, so it should not be used indiscriminately. For many applications wishing to specifically order loads and stores, the ***eieio*** instruction may provide the desired effect at lower performance cost.

Execution-only synchronizing instructions(those that are not also context synchronizing) are different from context synchronizing instructions in that they don't ensure that following instructions execute in the context established by the synchronizing instruction (The instruction queue is not flushed, so instructions after the synchronization in the queue execute in the old context). The new context takes effect sometime after the execution synchronizing instruction has completed. Like context synchronizing instructions, all execution synchronizing instructions halt dispatching and ensure that previous instructions have completed to the point where they may no longer cause an exception.

Some execution synchronizing instructions include ***sync***, ***mtmsr***, and all context synchronizing instructions.

The ***sync*** instruction, in addition to synchronizing execution, can broadcast addresses on the bus and is sometimes used for synchronizing coherent memory with other processors. Unlike ***isync***, ***sync*** forces all external accesses to complete with respect to other processors and mechanisms that access memory.

Table of Instructions Requiring Synchronization

The following table summarizes synchronization requirements for different instructions which alter context. This is a general guide. Individual processors sometimes have additional synchronization requirements or restrictions. Please refer to ***PowerPC Microprocessor Family: The Programming Environments for 32-Bit Microprocessors***, section 2.3.17, "Synchronization Requirements for Special Registers and for Lookaside Buffers" and your specific processor manual for complete information.

In addition to the instructions listed below, many processors have bits in the Hardware Implementation Registers (HIDs) that require some form of synchronization when they are changed. For example, on many PowerPC's, synchronization is required before changing the bits which control cache enabling and locking for the instruction and data caches. Changing the DCE and DLOCK bits in HID0 for many processors requires a ***sync*** prior to a change, and setting the ICE and ILOCK bits in HID0 necessitates an ***isync*** before the change. Please carefully read your processor's manual for further information about synchronization requirements when changing bits in Hardware Implementation Registers.

Please note that some register fields require synchronization differently for data or instruction accesses.

Instruction	Synchronization Required Prior	Synchronization Required After
mtmsr[PR]	none	context
mtmsr[FP](instr access)	none	context
mtmsr[FE0,FE1](instr access)	none	context
mtmsr[SE,BE](instr access)	none	context
mtmsr[ME]	none	context
mtmsr[LE]	implementation dependent	implementation dependent
mtmsr[DR](data access)	none	context
mtmsr[IR](instr access)	none	context
mtmsr[POW](instr access)	implementation dependent	implementation dependent

mtmsr[DABR](data access)	implementation dependent	implementation dependent
mtsr(data access)	context	context
mtsr(instr access)	none	context
mtsrin(data access)	context	context
mtsrin(instr access)	none	context
mtspr[SDR1]	***sync***	context
mtspr[DBAT](data access)	context	context
mtspr[IBAT](instr access)	none	context
mtspr[EAR](data access)	context	context
tlbie(data access)	context	context or ***sync***
tlbie(instr access)	none	context or ***sync***
tlbia(data access)	context	context or ***sync***
tlbia(instr access)	none	context or ***sync***

Sources of Information

Further information on this topic can be found in Motorola's PowerPC Libray and the publicationslisted below:

- ***PowerPC Microprocessor Family: The Programming Environments for 32-Bit Processors***
  Section 4.1.5 Synchronizing Instructions
  Section 6.1.2 Synchronization
  Section 2.3.17 Synchronization Requirements for Special Registers and for Lookaside Buffers
  Section 8.2 PowerPC Instruction Set
  Appendix E Synchronization Programming Examples (see listings for ***sync***, ***isync***, ***rfi***, ***sc***, ***mtmsr***, ***eieio***)
- ***MPC603e & EC603e User's Manual***
  Section 2.3.2.4 Synchronization
  Section 2.3.5.2 Memory Synchronization
  Section 6.3.3.2 Instruction Serialization
- ***MPC750 User's Manual***
  Section 2.3.2.4 Synchronization
  Section 2.3.4.7 Memory Synchronization Instruction
  Section 4.4 Process Switching
  Section 6.3.2.2 Instruction Serialization

The manuals for the individual processors not listed contain details for a particular processor as well. This would be too cumbersome to list here. Refer to your processor's manual for complete information.