3 Program Sequencing

Overview

Program sequencing on the ADSP-21160 is very similarly to sequencing on the ADSP-2106x family DSPs. The differences between the DSPs sequencing stem mostly from minor differences in their interrupt vector table and modes. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Program Sequencer Operations

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual. For a discussion of dual compute units, see "SIMD Processing" on page 6-1.

Conditional Instruction Execution

The program sequencer evaluates conditions to determine whether to execute a conditional instruction and when to terminate a loop. The conditions are based on information from the arithmetic status registers (ASTATx and ASTATy), mode control 1 register (MODE1), flag inputs and loop counter. The arithmetic ASTATx/y bits are described in "Computation Units" on page 2-1. Conditional execution while the ADSP-21160 operates in SIMD mode as described in "SIMD Processing" on page 6-1.

Each condition that the ADSP-21160 evaluates has an assembler mnemonic and a unique code which is used in a conditional instruction's opcode. For most conditions, the program sequencer can test both true and false states, e.g., equal to zero and not equal to zero. The ADSP-21160 has the same 32 condition and termination codes as the ADSP-2106x.

The bit test flags (BTF) are bit 18 of the ASTATx and ASTATy registers. These flags are set (or cleared) by the results of the BIT TST and BIT XOR forms of the *System Register Bit Manipulation* instruction (instruction Type 18), which can be used to test the contents of the ADSP-21160's system registers. This instruction is described in Appendix A, Group IV--Miscellaneous instructions. Note that this instruction can have asymmetric results across the two processing elements while the ADSP-21160 is operating in SIMD mode (PEYEN bit set in the MODE1 system register), allowing different system registers to be tested in parallel on each processing element (e.g. PEx will be testing a bit in ASTATx while PEy is testing a bit in ASTATy). After either BTF flag bit is set by this instruction, it can be used as the condition in a conditional instruction (with the mnemonic TF;

see Table 3.2 in the ADSP-2106x SHARC User's Manual). For a details of how this instruction works in SIMD mode, see "SIMD Mode and Instruction Support" on page 6-2.

The rest of the details of this feature on the ADSP-21160 operates the same as they do on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Branches

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual. For a discussion of branches during SIMD operation, see "SIMD Processing" on page 6-1.

Loops (DO UNTIL)

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual. For a discussion of loops during SIMD operation, see "SIMD Processing" on page 6-1.

Interrupts

Interrupt Latency

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Interrupt Vector Table

Table 3-1 shows all ADSP-21160 interrupts, listed according to their bit position in the IRPTL and IMASK registers and the LIRPTL register (see "Interrupt Latch Registers (IRPTL and LIRPTL)" on page 3-8). Also shown is the address of the interrupt vector; each vector is separated by four memory locations. The addresses in the vector table represent offsets from a base address. For an interrupt vector table in internal memory, the base address is 0x0004 0000; for an interrupt vector table in external memory, the base address is 0x0080 0000. The third column in Table 3-1 lists a mnemonic name for each interrupt. These names are provided for convenience, and are not required by the assembler.

Table 3-1. Interrupt (IRQ) Vectors & Priority

Register	IRPTL/ IMASK, LIRPTL Bit#	Vector Address[1]	IRQ Name[2]*	Function
IRPTL	0	0x00	EMUI	Emulator (read-only, non-maskable) **HIGHEST PRIORITY**
IRPTL	1	0x04	RSTI	Reset (read-only, non-maskable)
IRPTL	2	0x08	IICDI	Illegal Input Condition Detected
IRPTL	3	0x0C	SOVFI	Status, loop, or mode stack overflow; or PC stack full
IRPTL	4	0x10	TMZHI	Timer=0 (high priority option)
IRPTL	5	0x14	VIRPTI	Vector Interrupt
IRPTL	6	0x18	IRQ2I	asserted
IRPTL	7	0x1C	IRQ1I	asserted
IRPTL	8	0x20	IRQ0I	asserted
IRPTL	9	0x24	-	Reserved
IRPTL	10	0x28	SPR0I	DMA Channel 0 - SPORT0 Receive

Table 3-1. Interrupt (IRQ) Vectors & Priority (Cont'd)

Register	IRPTL/ IMASK, LIRPTL Bit#	Vector Address[1]	IRQ Name[2]*	Function
IRPTL	11	0x2C	SPR1I	DMA Channel 1 - SPORT1 Receive
IRPTL	12	0x30	SPT0I	DMA Channel 2 - SPORT0 Trans-mit
IRPTL	13	0x34	SPT1I	DMA Channel 3 - SPORT1 Trans-mit
LIRPTL	0	0x38	LP0I	DMA Channel 4 - Link Buffer 0
LIRPTL	1	0x3C	LP1I	DMA Channel 5 - Link Buffer 1
LIRPTL	2	0x40	LP2I	DMA Channel 6 - Link Buffer 2
LIRPTL	3	0x44	LP3I	DMA Channel 7 - Link Buffer 3
LIRPTL	4	0x48	LP4I	DMA Channel 8 - Link Buffer 4
LIRPTL	5	0x4C	LP5I	DMA Channel 9 - Link Buffer 5
IRPTL	15	0x50	EP0I	DMA Channel 10 - Ext. Port Buffer 0
IRPTL	16	0x54	EP1I	DMA Channel 11 - Ext. Port Buffer 1
IRPTL	17	0x58	EP2I	DMA Channel 12 - Ext. Port Buffer 2
IRPTL	18	0x5C	EP3I	DMA Channel 13 - Ext. Port Buffer 3

Table 3-1. Interrupt (IRQ) Vectors & Priority (Cont'd)

Register	IRPTL/ IMASK, LIRPTL Bit#	Vector Address[1]	IRQ Name[2]*	Function
IRPTL	19	0x60	LSRQI	Link Port Service Request
IRPTL	20	0x64	CB7I	Circular Buffer 7 overflow
IRPTL	21	0x68	CB15I	Circular Buffer 15 overflow
IRPTL	22	0x6C	TMZLI	Timer=0 (low priority option)
IRPTL	23	0x70	FIXI	Fixed-point overflow
IRPTL	24	0x74	FLTOI	Floating-point overflow exception
IRPTL	25	0x78	FLTUI	Floating-point underflow exception
IRPTL	26	0x7C	FLTII	Floating-point invalid exception
IRPTL	27	0x80	SFT0I	User software interrupt 0
IRPTL	28	0x84	SFT1I	User software interrupt 1
IRPTL	29	0x88	SFT2I	User software interrupt 2
IRPTL	30	0x8C	SFT3I	User software interrupt 3
IRPTL	31	0x90	-	Reserved **LOWEST PRIORITY**

[1] Offset from base address: 0x0080 0000 for interrupt vector table in external memory.

[2] These IRPTL/IMASK and LIRPTL bit names are defined in the defhamhd.h include file supplied with the ADSP-21160 Family Development Software. Note that bit16 of IRPTL/IMASK is LPSUM control.

The interrupt vector table may be located in internal memory, at address 0x0004 0000 (the beginning of Block 0), or in external memory at address 0x0080 0000. If the ADSP-21160's on-chip memory is booted from an external source, the interrupt vector table will be located in internal memory. If, however, the ADSP-21160 is not booted (because it will execute from off-chip memory), the vector table must be located in the off-chip memory. For details on booting mode selection, see "Booting" on page 15-21.

Also, if booting is from an external EPROM or host processor, bit 16 of `IMASK` (the `EP0I` interrupt for external port DMA Channel 10) will automatically be set to 1 following reset -- this enables the DMA done interrupt for booting on Channel 10. `IRPTL` is initialized to all zeros following reset.

The `IIVT` bit in the `SYSCON` control `IOP` register can be used to override the booting mode in determining where the interrupt vector table is located. If the ADSP-21160 is not booted (no boot mode), setting `IIVT` to 1 selects an internal vector table while `IIVT=0` selects an external vector table. If the ADSP-21160 is booted from an external source (any mode other than *no boot* mode), then IIVT has no effect.

Interrupt Latch Registers (IRPTL and LIRPTL)

The interrupt latch registers (`IRPTL` and `LIRPTL`) are each 32-bit registers that latch interrupts. They indicate all interrupts currently being serviced as well as any which are pending. Because both registers are readable and writable, any interrupt (except reset) can be set or cleared in software. Do not write to the reset bit (bit 1) in IRPTL because this puts the processor into an illegal state.

The LIRPTL register is dedicated to the support of the Link Port interrupts. Both interrupt latch bits for each Link Port DMA channel, and their associated mask bits appear in this register. A logical OR'ing of the masked latch state is then summarized in the LPSUM bit in the IRPTL register. The LPSUM bit has a corresponding mask bit in the IMASK register, thus providing a second level of interrupt masking.

When an interrupt occurs, the corresponding bit in either IRPTL or LIRPTL is set. During execution of the interrupt's service routine, this bit is kept cleared -- the ADSP-21160 clears the bit during every cycle, preventing the same interrupt from being latched while its service routine is already executing.

A special method is provided, however, to allow the reuse of an interrupt while it is being serviced. This method is provided by the clear interrupt (CI) modifier of the JUMP instruction. For more information, see "Clearing the Current Interrupt For Reuse" on page 3-15.

Both IRPTL and LIRPTL are cleared by a processor reset.

Interrupt Priority

The interrupts listed in Table 3-1 are ordered by priority. Interrupt priority determines which interrupt is serviced first when more than one occurs in the same cycle. It also determines which interrupts are nested when nesting is enabled. For more information, see "Interrupt Nesting & IMASKP (and LIRPTL)" on page 3-11.

The arithmetic interrupts -- fixed-point overflow and floating-point overflow, underflow, and invalid operation -- for both processing elements are determined from flags in the sticky status registers (`STKYx` and `STKYy`). By reading these flags, the service routine for one of these interrupts can determine which condition caused the interrupt. The routine also has to clear the appropriate `STKYx` or `STKYy` bit so that the interrupt is not still active after the service routine is done.

The timer decrementing to zero causes both the `TMZHI` interrupt and `TMZLO` interrupt. This feature allows you to choose the priority of the timer interrupt. Unmask the timer interrupt that has the priority you want, and leave the other one masked. Unmasking both interrupts results in two interrupts when the timer reaches zero. In this case the processor services the higher priority interrupt first, then the lower priority interrupt.

Interrupt Masking & Control

All interrupts except for reset and the emulator interrupts can be enabled and disabled by the global interrupt enable bit, `IRPTEN`, bit 12 in the `MODE1` register. This bit is cleared at reset. You must set this bit for interrupts to be enabled.

Interrupt Mask Registers (IMASK and LIRPTL)

All interrupts except for reset and the emulator interrupts can be masked. Masked means the interrupt is disabled. Interrupts that are masked are still latched (in `IRPTL` or `LIRPTL`), so that if the interrupt is later unmasked, it is processed.

The IMASK register and LIRPTL register control interrupt masking. The bits in IMASK correspond exactly to the bits in the IRPTL register. Similarly the high order bits of the LIRPTL register correspond exactly to the low order latch bits in the LIRPTL register. For example, bit 10 in IMASK masks or unmasks the same interrupt latched by bit 10 in IRPTL. Similarly, bit 20 in LIRPTL masks or unmasks the same interrupt latched by bit 4 in LIRPTL.

- If a bit in IMASK or the mask field of LIRPTL is set to 1, its interrupt is unmasked (enabled).
- If the bit is cleared (to 0), the interrupt is masked (disabled).

After reset, all interrupts except for the reset and emulator interrupts, and the EP0I interrupt for external port DMA Channel 10 (bit 15 of IMASK) or LP4I interrupt DMA Channel 8 (bits 4/28 of LIRPTL) are masked. The reset interrupt and emulator interrupt are always non-maskable. Either one of the EP0I or LP4I interrupts, or neither is automatically unmasked after reset, depending on whether the ADSP-21160 is booting from EPROM, host, link ports, or from internal memory.

Interrupt Nesting & IMASKP (and LIRPTL)

(Bits 28-24 of LIRPTL comprise the interrupt mask pointer bits that are used during interrupt nesting of the Link Port interrupts.)

Status Stack Save & Restore

For low-overhead interrupt servicing, the ADSP-21160 automatically saves and restores the status and mode contexts of the interrupted program for the following interrupt sources only:

- the three external interrupts (IRQ2-0)
- the timer interrupt
- the VIRPT vector interrupt.

These interrupts cause an automatic push of `ASTATx`, `ASTATy`, and `MODE1` onto the status stack, which is now fifteen levels deep. These registers are automatically popped from the status stack by the return from interrupt instruction, `RTI` and by the `JUMP (CI)` instruction, which is described in "Clearing the Current Interrupt For Reuse" on page 3-15.

Only the external, timer, and `VIRPT` interrupts cause a push of the status stack. All other interrupts require an explicit save and restore of the appropriate register to memory, or use of the `PUSH/POP STS` instruction.

Pushing `ASTATx`, `ASTATy`, and `MODE1` preserves the status and control bit settings so that if the service routine alters these bits, the original settings are automatically restored upon the return from the interrupt.

The top of the status stack contains the current values of `ASTATx`, `ASTATy`, and `MODE1`. Reading and writing these registers does not move the stack pointer. The stack pointer is moved, however, by explicit `PUSH` and `POP` instructions.

Mode Mask System Register (MMASK)

The ADSP-21160's Mode Mask (MMASK) system register contains one mask bit for each bit in the MODE1 system register. The mask is applied to the MODE1 system register under the following conditions:

- each time a TYPE 20, PUSH STS instruction is executed. For information on instruction types, see "Instruction Set Reference" on page A-1.
- each time an external interrupt (IRQ), timer interrupt, or VIRPT vector interrupt cause an automatic push of status onto the status stack.

A bit that is set in the MMASK register results in the clearing of the corresponding bit in the MODE1 register immediately following the push of status. In other words, the original MODE1 register contents are first pushed onto the 15 entry status stack, and then the MODE1 register contents are modified based on the state of the MMASK register.

The MMASK register is initialized to 0x0020 0000 (PEYEN bit masked forcing SISD mode on PEx) following chip reset.

An example of where this function may be useful is ISR's that must execute in SISD mode. These ISR's can have the PEYEN bit saved and the PEYEN bit cleared (SISD mode enabled) either automatically, or with one instruction in the ISR preamble by using this feature.

Illegal Input Condition Interrupt

This new interrupt, introduced with ADSP-21160, can detect and signal two types of programming errors (illegal input conditions):

1. inadvertent IOP register access
2. unaligned 64-bit memory reference.

IOP Register Access Control

If the IICDI interrupt is unmasked, and the Mode1 register control bit IIRAE is set, an instruction fetch of or a data write to IOP register space, either local to this ADSP-21160, or on another ADSP-21160 via multiprocessor memory space, will generate an interrupt. This interrupt should be unmasked after the IOP has been initialized, and no further accesses are expected.

Mode1 register space is used for this control enable because Mode1 can be quickly modified by pushing it on the Mode stack. Routines which should have IOP Register space access (e.g. MP communication) can gain “legal” access by setting the corresponding Mode1 Mask bit, then using the PUSH/POP instructions to save/restore Mode1 state before/after calling the IOP access routine. For more information, see “MODE1 Register” on page E-14.

Unaligned 64-bit Memory Access Control

If the core makes an unaligned 64-bit access to memory (internal or external), and the interrupt is unmasked, and the Mode2 register control bit U64MAE is set, an interrupt will be generated. An unaligned 64-bit memory reference is generated only by a data movement instruction with the LW-bit set, and for which the address generated by the instruction is odd (lsb set). There is no compelling reason for this interrupt control to be pushed on the stack, which is why the enable has been allocated in the Mode2 register. For more information, see "MODE2 Register" on page E-19.

Once the sequencer has vectored to the IICDI interrupt, software can poll the Illegal IOP Register Access flag, bit 19, or the Unaligned Memory Access flag, bit 20, in the Sticky X register, to determine the source of the interrupt.

NOTE: This solution does not PREVENT the illegal access from occurring; it only provides a detection mechanism AFTER an illegal access has happened. Additionally, this provides master access protection only; slaves do not detect that an IOP Register space access has occurred. Also note that DMA channels will never generate the interrupt.

Software Interrupts

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Clearing the Current Interrupt For Reuse

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

External Interrupt Timing & Sensitivity

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Multiprocessor Vector Interrupts (VIRPT)

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Timer

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Stack Flags

With one exception, this feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the

ADSP-2106x SHARC User's Manual. The exception is that STKY is replaced with STKYx, stack flags appear in STKYx only and are not duplicated in STKYy.

IDLE & IDLE16

This feature on the ADSP-21160 operates the same as this feature on the ADSP-21061 DSP; supports the IDLE16 instruction. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.

Instruction Cache

This feature on the ADSP-21160 operates the same as this feature on the ADSP-2106x family DSPs. For more information, see the corresponding section in the ADSP-2106x SHARC User's Manual.