

Demonstrating TMS320C2xx Pipeline Operation During an Interrupt

APPLICATION BRIEF: SPRA357

James Doublesin

*Digital Signal Processing Solutions
January 1998*



IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE (281) 274-2320

US TMS320 FAX (281) 274-2324

US TMS320 BBS (281) 274-2323

US TMS320 email dsph@ti.com

Contents

Abstract.....	7
Product Support on the World Wide Web.....	8
TMS320C2xx Pipeline Scenarios.....	9
Interrupt Occurring Around SETC and CLRC.....	10
Changing IMR Register Bit to Protect Code without Globally Disabling Interrupts.....	19

Examples

Example 1. C2xx Pipeline Interrupt at XOR.....	10
Example 2. C2xx Pipeline Interrupt at LACL	11
Example 3. C2xx Pipeline Interrupt at SETC.....	12
Example 4. C2xx Pipeline Interrupt at ADD.....	13
Example 5. C2xx Pipeline Interrupt at SACL.....	14
Example 6. C2xx Pipeline Interrupt at CLRC.....	15
Example 7. C2xx Pipeline Interrupt at SUB	16
Example 8. C2xx Pipeline Interrupt at SACH	17
Example 9. C2xx Pipeline Interrupt at AND.....	18
Example 10. Unprotected Code During an IMR Change	19
Example 11. Protecting the Code During an IMR Change.....	20

Demonstrating TMS320C2xx Pipeline Operation During an Interrupt

Abstract

This application brief describes the behavior of the Texas Instruments (TI™) TMS320C2xx pipeline during an interrupt occurring around the SETC and CLRC instructions. This brief also explains how to change the appropriate bit in the IMR register to protect a block of code without globally disabling interrupts.

Each scenario was tested using the TMS320C209SE ('C209SE) DSP and its internal timer as the interrupt source. Pipeline operation was verified using actual code traces on the XDS511/522 emulator.



Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.



TMS320C2xx Pipeline Scenarios

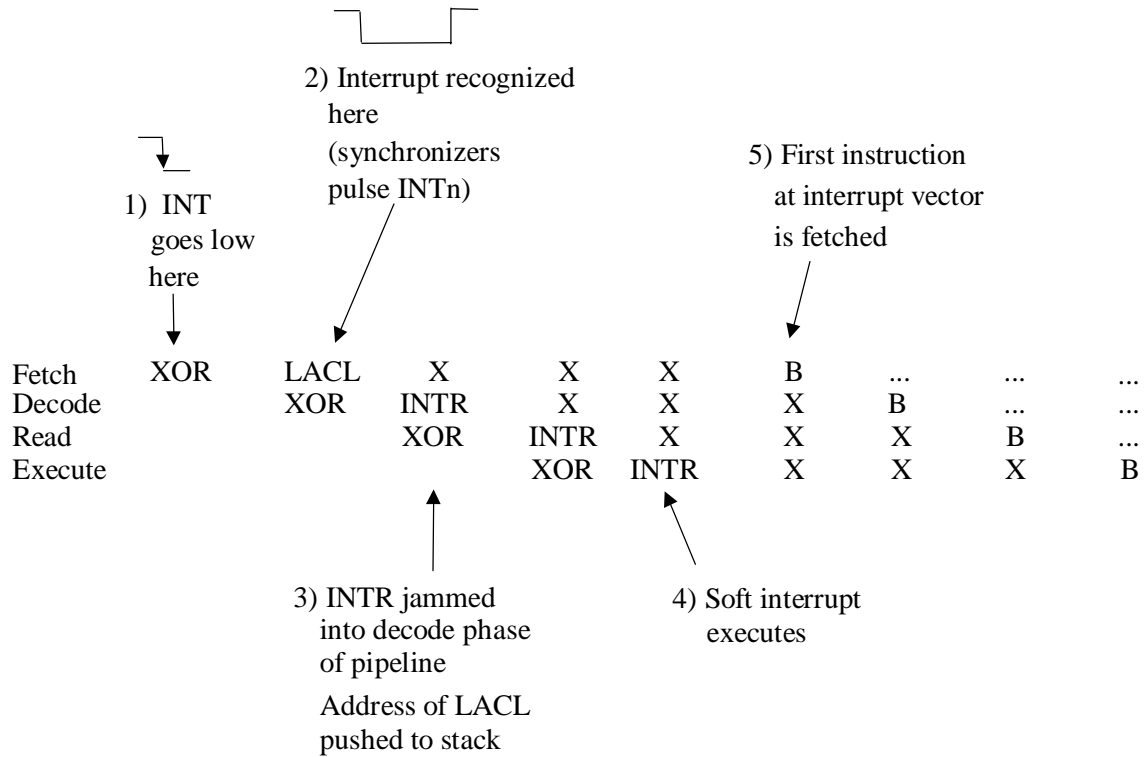
Example 1 through Example 9 show an analysis of the TMS320C2xx DSP pipeline behavior during an interrupt occurring around the SETC and CLRC instructions. Example 10 and Example 11 focus on changing the appropriate bit in the IMR register to protect a block of code without globally disabling interrupts.

For each example:

- ❑ The interrupts have a one-cycle synchronization to the CPU core. When an interrupt occurs, the synchronization circuitry actually recognizes it on the rising edge of the clock (between each cycle). If the interrupt is valid on the clock edge, the synchronization circuitry sends a low pulse to the core (interrupt recognized). There is no interrupt synchronization on the 'C2xlp core, but the diagrams can be adjusted depending on the type of synchronization you have.
- ❑ During a CLRC INTM instruction, the CPU automatically holds off any interrupt through the execution phase for it and the following instruction.
- ❑ An interrupt occurs in the CPU by jamming an INTR instruction into the decode phase of the pipeline. The address of the most recently fetched instruction is pushed to the stack so that normal operation can continue on return from the interrupt.
- ❑ When an interrupt occurs, all instructions in the pipeline will complete through the execute phase.
- ❑ CLRC INTM changes INTM in the execution phase. SETC INTM changes INTM in the decode phase.
- ❑ SACL IMR changes the IMR in the execution phase, and the change in the IMR is not realized until the next cycle.
- ❑ The pipeline is represented with zero wait state operation.

Interrupt Occurring Around SETC and CLRC

Example 1. C2xx Pipeline Interrupt at XOR



Code Example

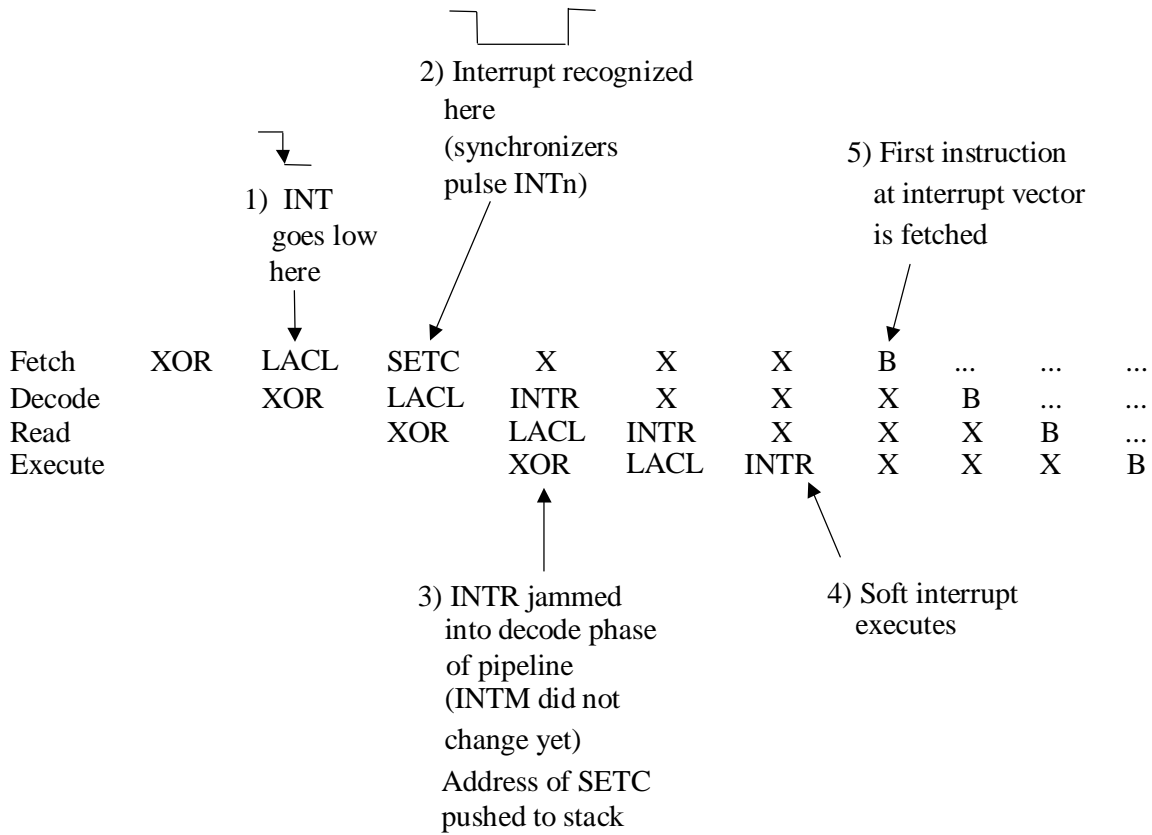
int occurs here →

```

XOR    *
LACL   #0
SETC   INTM
ADD    *
SACL   *
CLRC   INTM
SUB    *
SACH   *
AND    *
OR     *
    
```



Example 2. C2xx Pipeline Interrupt at LACL



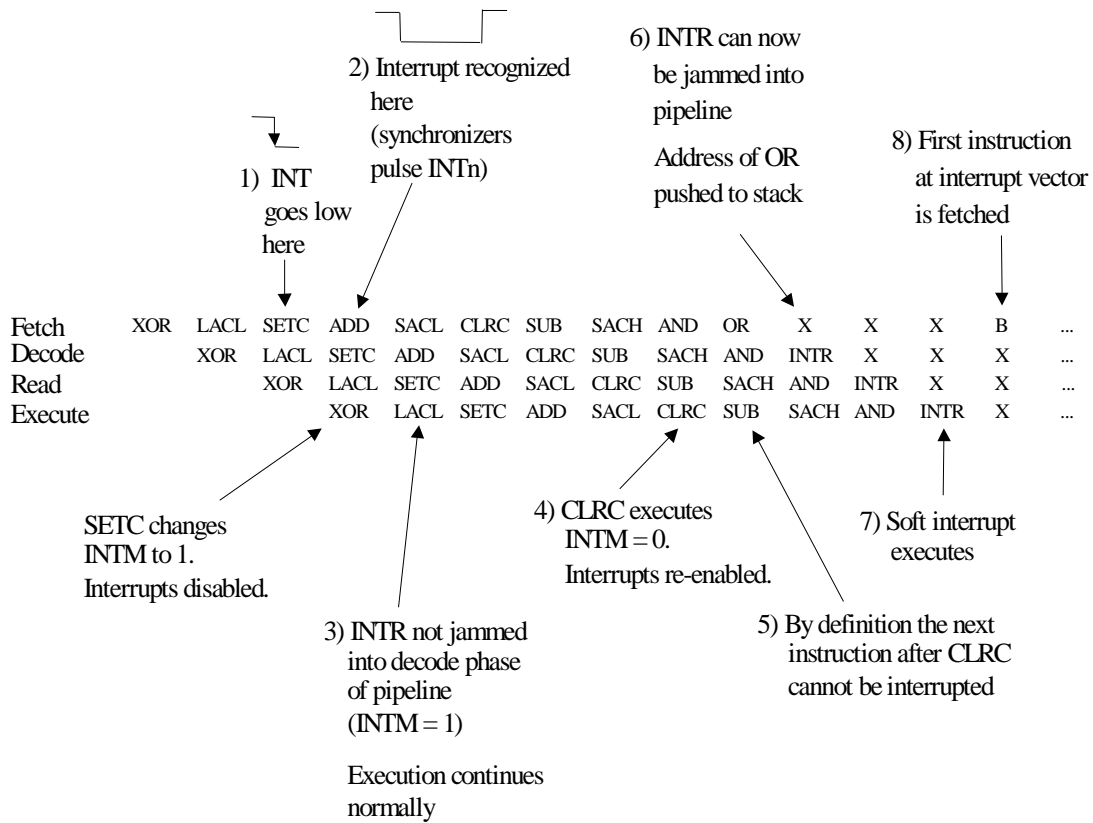
Code Example

```

int occurs here → XOR *
                  LACL #0
                  SETC INTM
                  ADD *
                  SACL *
                  CLRC INTM
                  SUB *
                  SACH *
                  AND *
                  OR *
    
```



Example 3. C2xx Pipeline Interrupt at SETC



Code Example

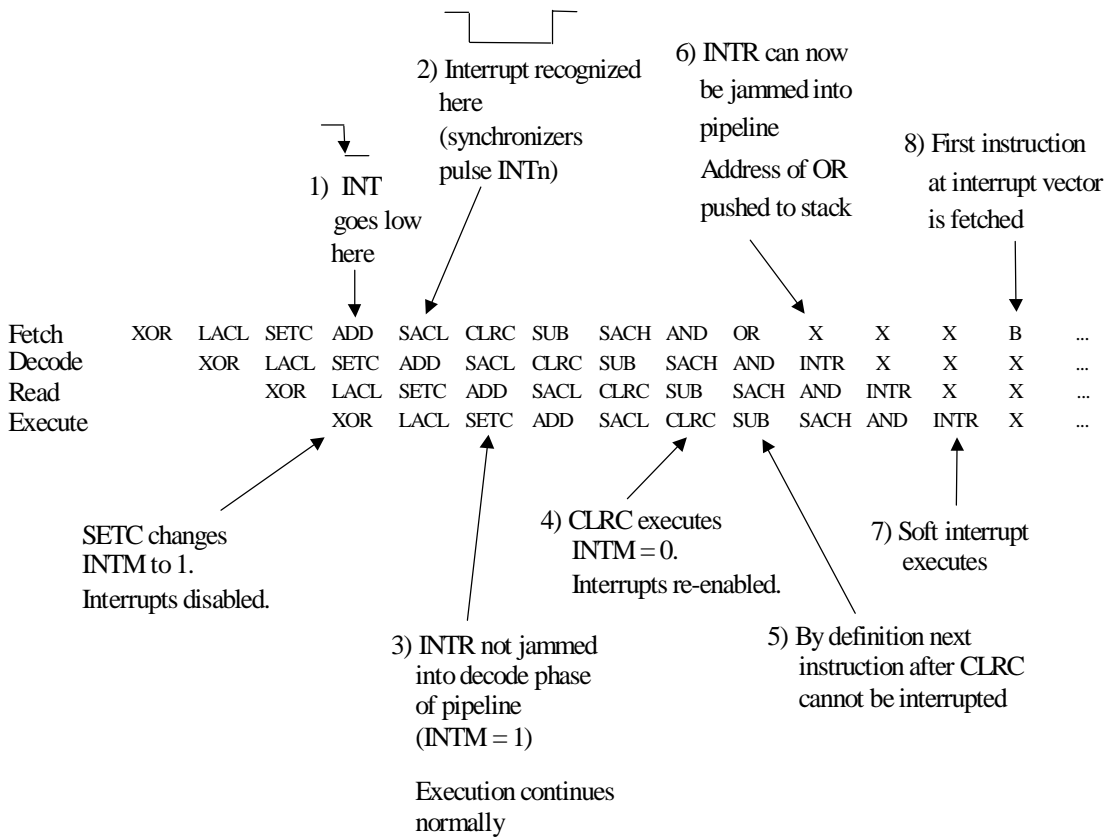
```

XOR      *
LACL     #0
int occurs here → SETC    INTM
ADD      *
SACL     *
CLRC     INTM
SUB      *
SACH     *
AND      *
OR       *

```



Example 4. C2xx Pipeline Interrupt at ADD

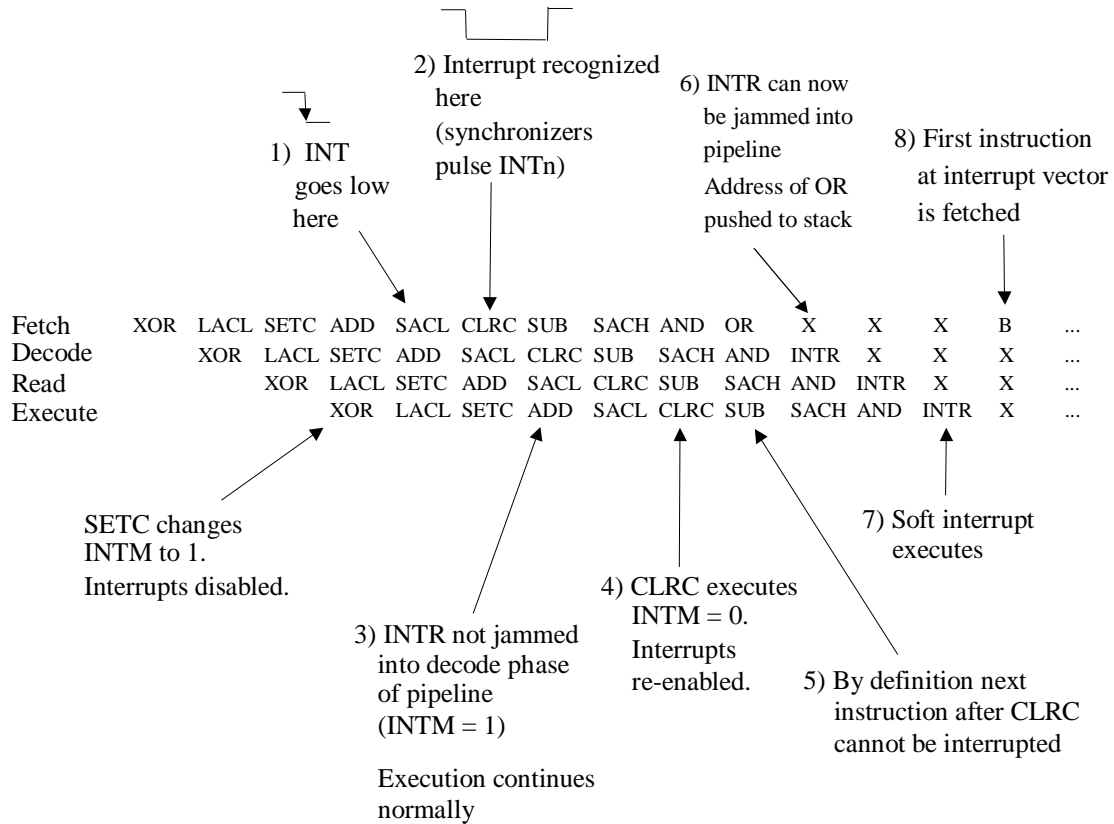


Code Example

```

XOR      *
LACL    #0
SETC    INTM
int occurs here → ADD    *
          *
          SACL    *
          CLRC    INTM
          SUB     *
          SACH    *
          AND     *
          OR      *
    
```

Example 5. C2xx Pipeline Interrupt at SACL



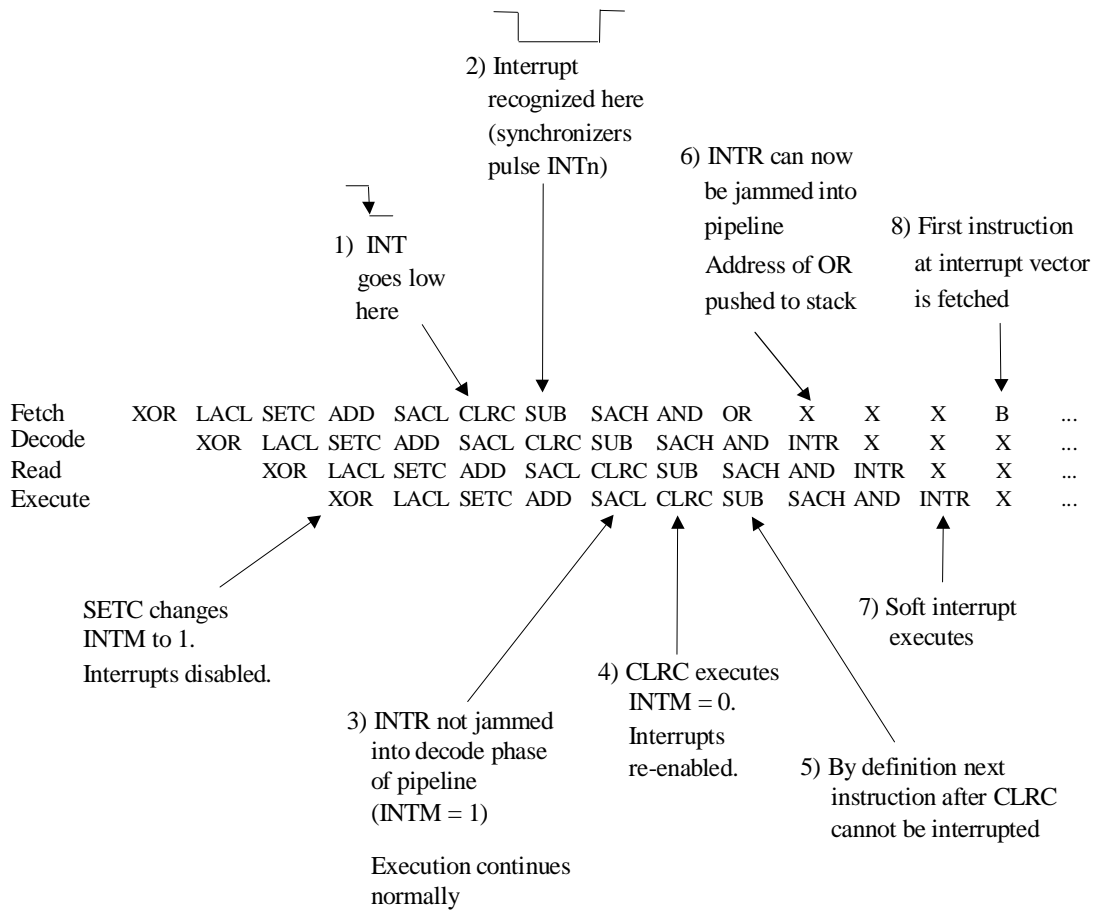
Code Example

```

XOR      *
LACL     #0
SETC     INTM
ADD      *
int occurs here → SACL     *
CLRC     INTM
SUB      *
SACH     *
AND      *
OR       *
    
```



Example 6. C2xx Pipeline Interrupt at CLRC

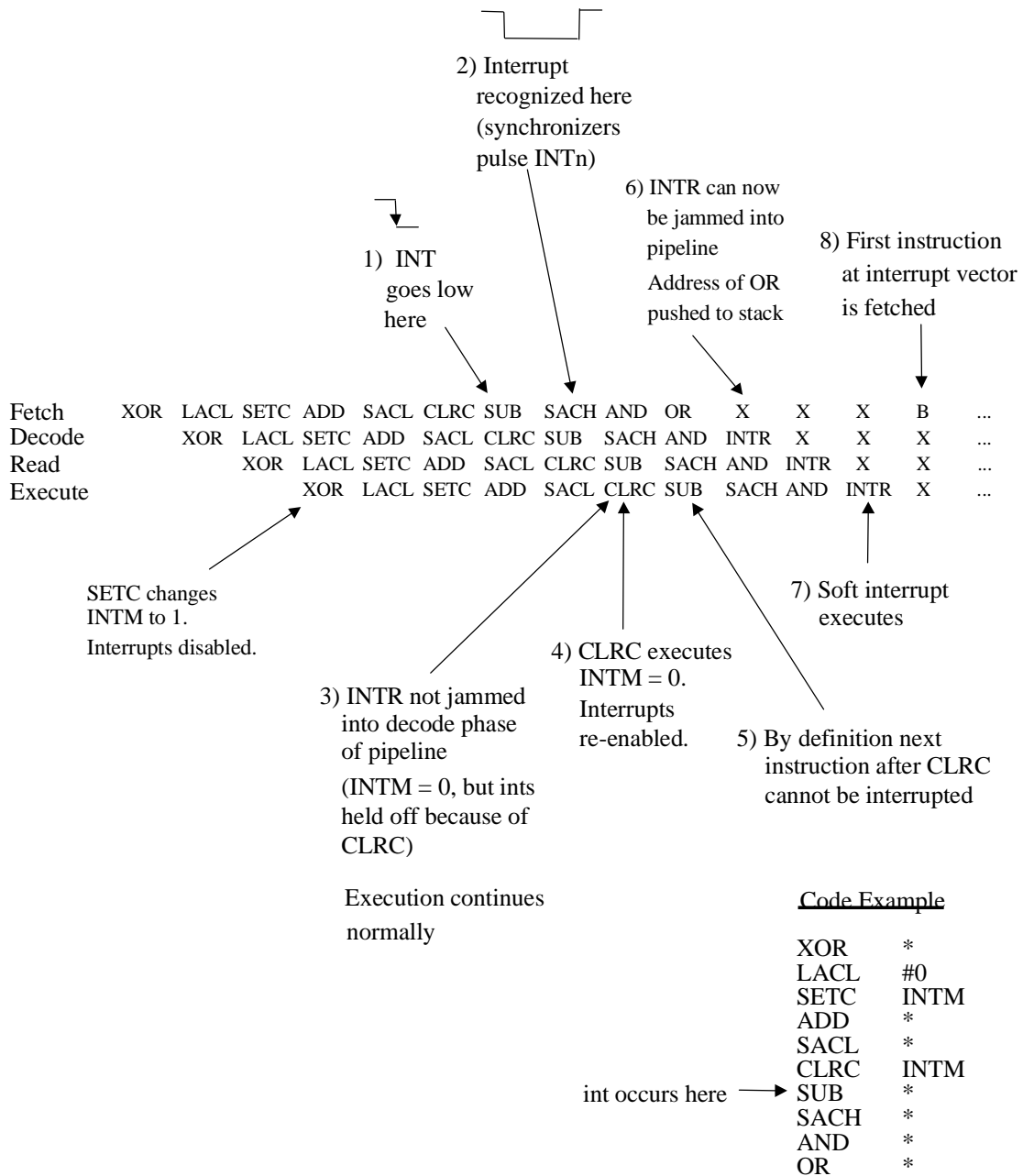


Code Example

```

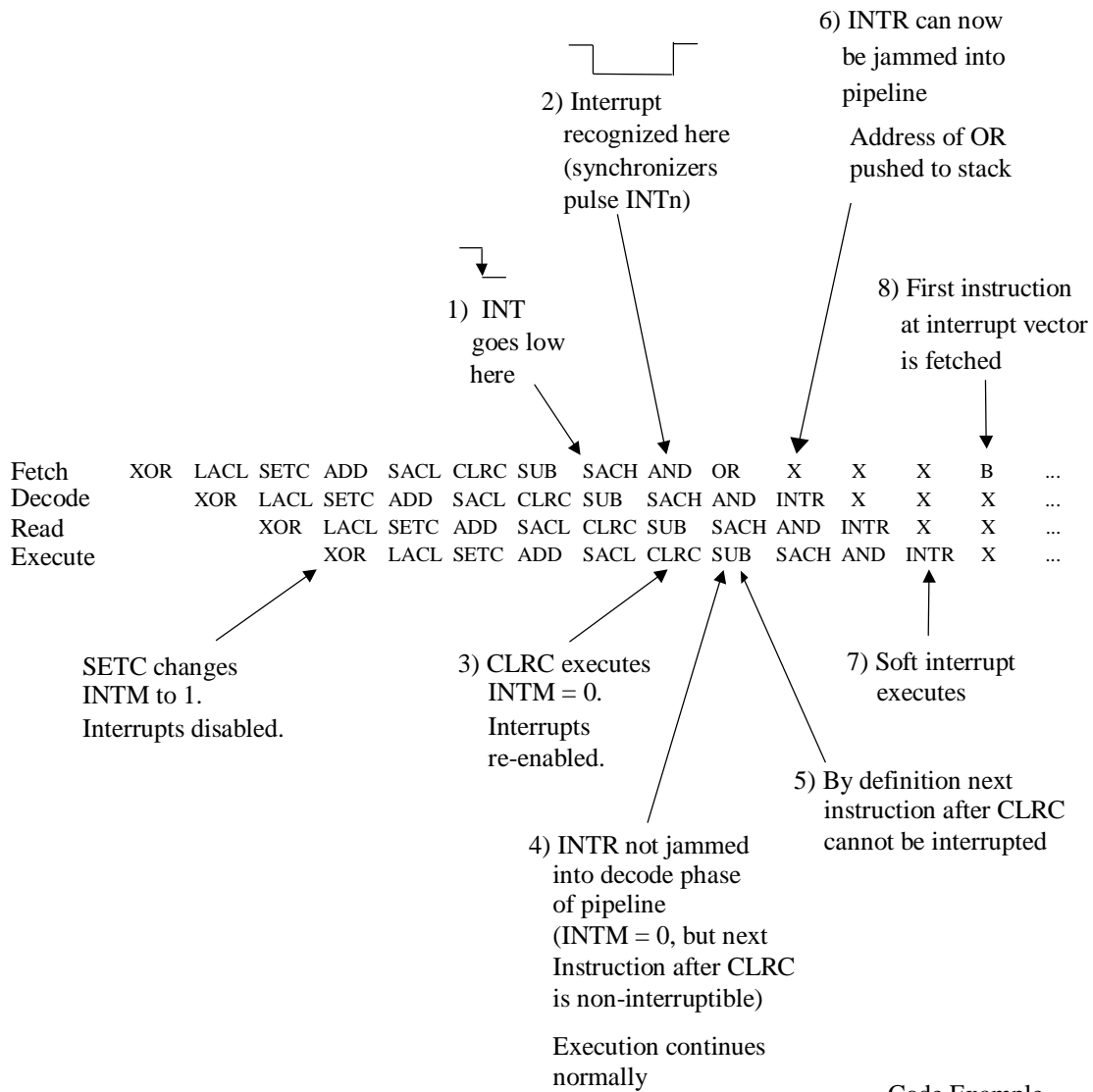
XOR      *
LACL    #0
SETC    INTM
ADD     *
SACL    *
int occurs here → CLRC  INTM
SUB     *
SACH    *
AND     *
OR      *
    
```

Example 7. C2xx Pipeline Interrupt at SUB





Example 8. C2xx Pipeline Interrupt at SACH



Code Example

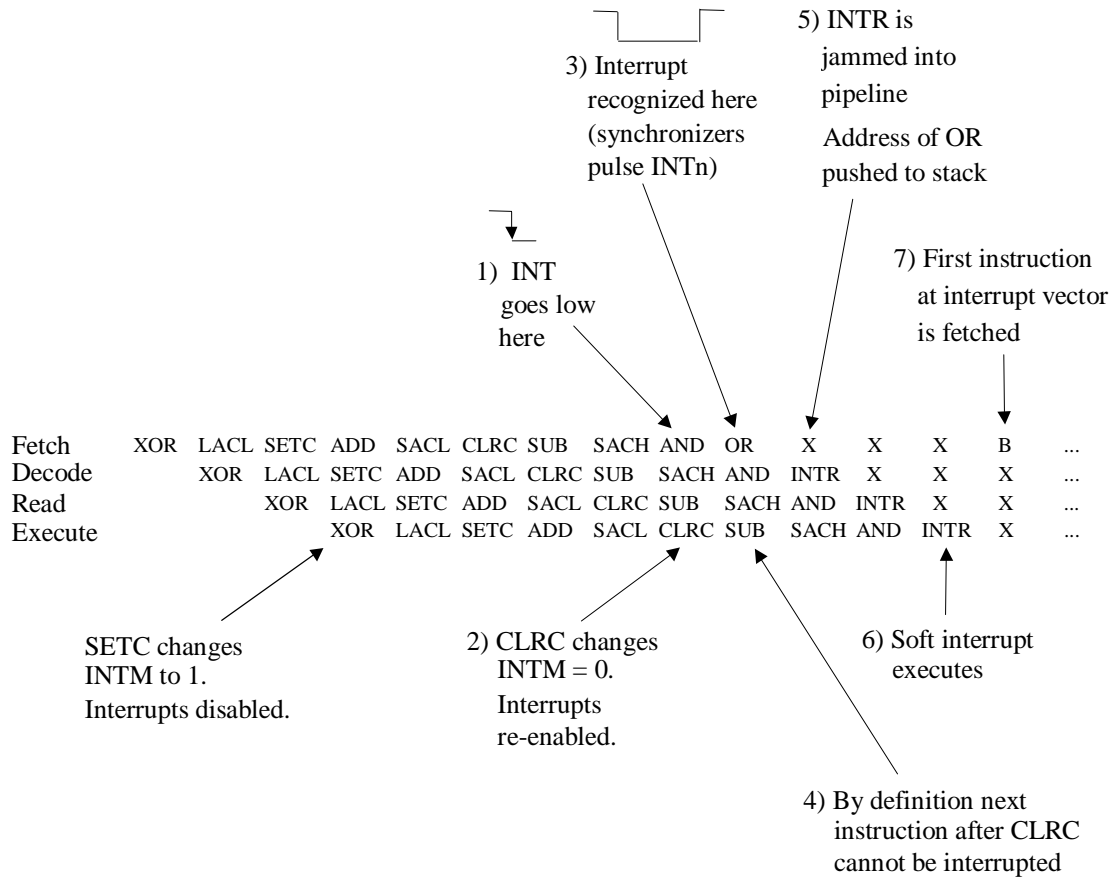
```

XOR      *
LACL     #0
SETC     INTM
ADD      *
SACL     *
CLRC     INTM
SUB      *
SACH     *
AND      *
OR       *
    
```

int occurs here →



Example 9. C2xx Pipeline Interrupt at AND



Code Example

```

XOR      *
LACL     #0
SETC     INTM
ADD      *
SACL     *
CLRC     INTM
SUB      *
SACH     *
AND      *
OR       *

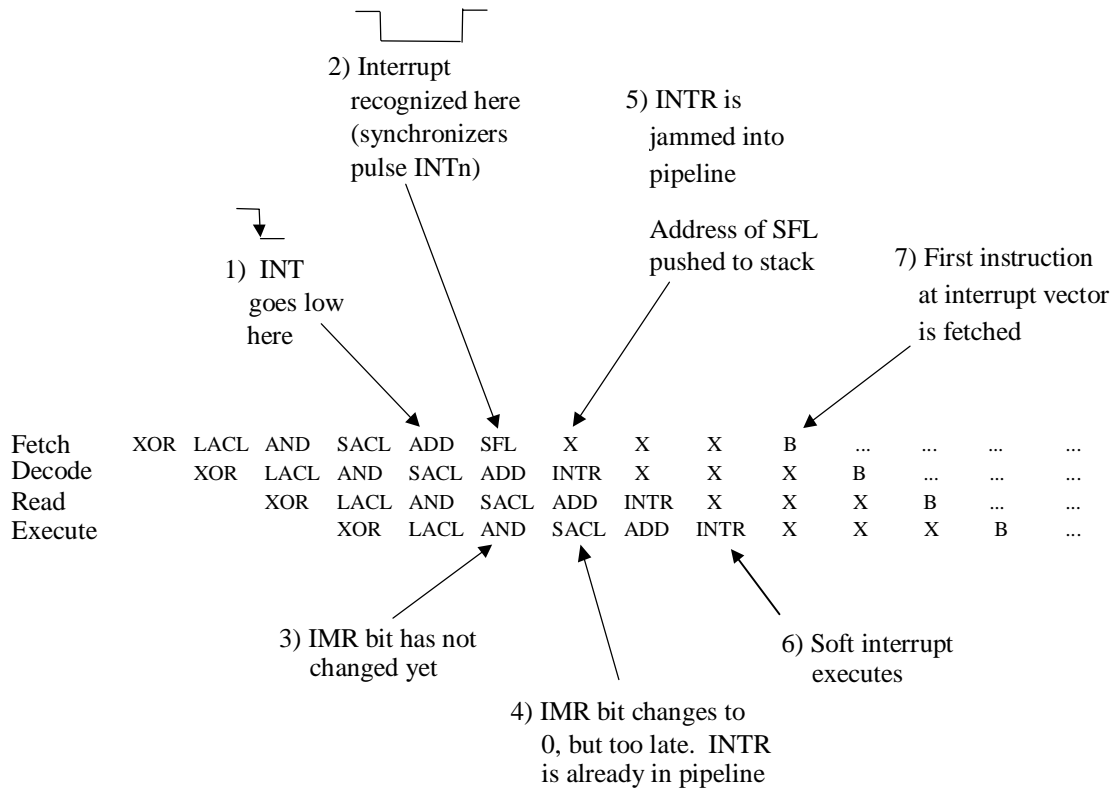
```

int occurs here →



Changing IMR Register Bit to Protect Code without Globally Disabling Interrupts

Example 10. Unprotected Code During an IMR Change



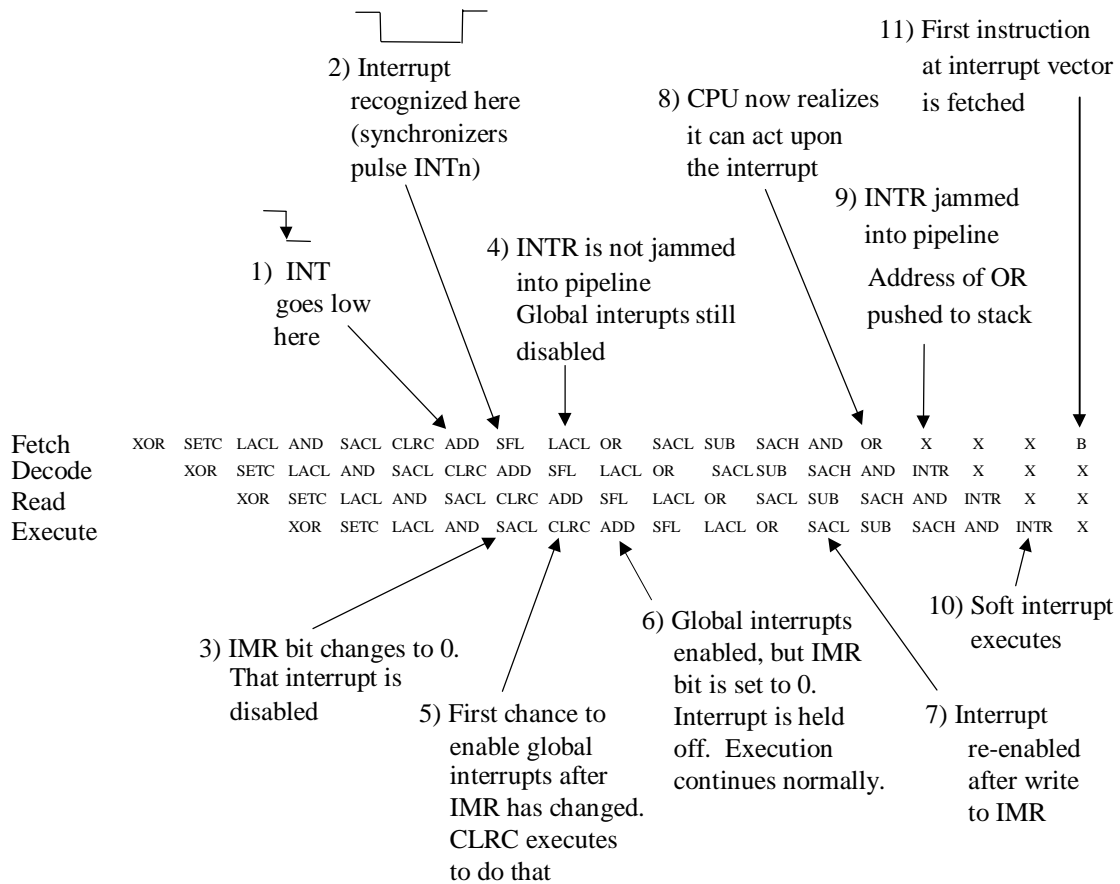
If you want to protect a block of code without globally disabling interrupts, you must change the appropriate bit in the IMR register. This method, however, poses some different problems that the user must be aware of. First, you must realize that a SACL IMR instruction changes the IMR in the execution phase. This is a problem, because if the interrupt comes in as shown above, the interrupt will be executed during the code that you wanted to protect.

Code Example

```

XOR      *
LACL    #0
AND     IMR
SACL    IMR
int occurs here → ADD    *
SFL     *
LACL    #8
OR      IMR
SACL    IMR
SUB     *
SACH    *
AND     *
OR      *
    
```

Example 11. Protecting the Code During an IMR Change



To properly protect the changing of an IMR bit to disable an interrupt, you must include a SETC and CLRC around the code to protect it during the IMR change. The code to the right shows how you can disable an individual interrupt while keeping global interrupts off for the shortest possible time. Moving the CLRC up to take advantage of its delay in re-enabling interrupts will not work because the interrupt will be recognized in between the CLRC INTM and SACL IMR. If you save the status of the accumulator in your interrupt service routine, you can move the SETC INTM instruction between AND IMR and SACL IMR to further reduce the time that global interrupts are disabled. In the code to the right, all instructions except for the first (XOR *) and the last (OR *) are protected from interrupts.

Code Example		
XOR	*	;unprotected code
SETC	INTM	;disable global ints to change IMR
LACL	#0	;setup to disable int in IMR
AND	IMR	
SACL	IMR	;change bit in IMR
CLRC	INTM	;enable global ints
ADD	*	;protected code
SFL	*	;protected code
LACL	#8	;setup to enable int in IMR
OR	IMR	
SACL	IMR	;change IMR to enable int
SUB	*	;protected code
SACH	*	;protected code
AND	*	;protected code
OR	*	;unprotected code

int occurs here →